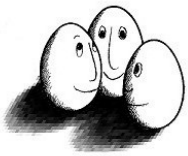
technische universität dortmund

Fakultät Informatik
Lehrstuhl für
Künstliche Intelligenz



fakultät für mathematik m!

Überarbeitete Version der
Masterarbeit im Studiengang Wirtschaftsmathematik
Sommersemester 2018

„LASSO vs. SLOPE: Vergleich und deren praktische Umsetzung anhand von CAMDA- und TCGA-Daten“

Gutachter:
Prof. Dr. Katharina Morik
Prof. Dr. Stefan Turek

Vorgelegt von:
Viktoria Kliewer
viktoria.kliewer@tu-dortmund.de

Danksagung

In erster Linie möchte ich mich ganz herzlich bei Professor Katharina Morik bedanken, insbesondere für ihre Unterstützung und den Glauben an mich, die mir immer wieder Kraft gegeben haben. Professor Morik habe ich in der Vorlesung "Wissensentdeckung in Datenbanken" kennengelernt und war gleich von ihr begeistert. Sie hat mir die großartige Chance gegeben, drei Jahre lang Teil des Lehrstuhls für künstliche Intelligenz, den sie leitet, zu sein.

Professor Stefan Turek möchte ich ebenso meinen Dank aussprechen, vor allem dafür, dass er sich der Begutachtung eines ihm weniger vertrauten Themas angenommen hat. Im Laufe meines Studiums habe ich viele seiner Vorlesungen und Seminare besucht, zu denen ich sehr gerne hingegangen bin.

Sowohl Professor Morik als auch Professor Turek gehören zu den Professoren, die mit besonderer Überzeugung und Begeisterung lehren und zudem den Studierenden auf Augenhöhe begegnen. Beide Professoren haben mein Studium bereichert. Ich bin davon überzeugt, dass sie noch viele andere Studenten inspirieren werden.

Einer im Laufe meines Studiums wichtig gewordenen Person möchte ich ebenfalls meinen Dank aussprechen. Die Ansprechpartnerin für Wirtschaftsmathematiker Frau Kathrin Waßmund war immer geduldig, freundlich, verständnisvoll, zuvorkommend und hat einige Katastrophen verhindern können. Ohne sie hätte sich vieles anders entwickeln können.

Einen ganz besonderen Dank richte ich an meine Eltern, Lilia und Grigorij Kliewer, die mir stets eine Stütze wie aus Stahl waren, sei es in emotionaler oder finanzieller Hinsicht. Ich habe Ihnen sehr viel zu verdanken, nicht zuletzt den Entschluss, die Heimat, die Familie, die Arbeit, das Vertraute, das Heimatland Kasachstan zu verlassen und nach Deutschland auszuwandern, insbesondere um ihren Kindern, meiner Schwester und mir, eine bessere Zukunft zu ermöglichen. Diese Arbeit möchte ich ihnen widmen.

Zum Schluss möchte ich noch einer ganz besonderen Person, meinem Lebensgefährten, danken. Dafür, dass er mich immer unterstützt, mich immer wieder vor völligem Verzweifeln gerettet, mich aufgebaut und mir Halt gegeben hat. Ebenso danke ich für sein Durchhaltevermögen und das Ertragen all meiner Launen.

Es gibt noch viele andere Menschen, die mich während meines Studiums begleitet, mir geholfen und Rückhalt gegeben haben. Allen, die ich hier nicht namentlich erwähne, möchte ich ganz herzlich danken.

Viktoria Kliewer

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmusverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BH-Prozedur	Benjamin-Hochberg-Prozedur
BRCA	Breast Invasive Carcinoma
bzw.	beziehungsweise
bspw.	beispielsweise
ca.	circa
CAMDA	Critical Assessment of Massive Data Analysis
CD	Coordinate Descent
CV	Cross Validation, Kreuzvalidierung
CV-Fehler	Kreuzvalidierungsfehler
d.h.	das heißt
engl.	englisch
FDP	False Discovery Proportion
FDR	False Discovery Rate
FWER	Familywise Error Rate
FP	False Positive
FN	False Negative
GLM	Generalisierte(s) lineare(s) Modell(e) (Singular/Plural)
ISTA	Iterative Soft-Thresholding Algorithms
KKT-Bedingungen	Karush-Kuhn-Tucker-Bedingungen
LARS	Least Angle Regression
LASSO	Least Absolute Shrinkage and Selection Operator
LOOCV	Leave-One-Out-CV
O.B.d.A.	ohne Beschränkung der Allgemeinheit
OLS	Ordinary Least Squares, Summe der kleinsten Quadrate
mRNA	messenger RNA
miRNA	microRNA
NP	nichtdeterministische Polynomialzeit
TP	True Positive
QP	quadratisches Programm
RNA	ribonucleic acid, Ribonukleinsäure
RP	Regularisierungsparameter (Singular und Plural)
S.	Seite
SLOPE	Sorted L-One Penalized Estimation
s.t.	subject to, unter der Nebenbedingung
TCGA	The Cancer Genome Atlas
URL	Uniform Resource Locator
vgl.	vergleiche

Notationsverzeichnis

Falls nicht anders definiert:

$\mathbb{R}$, $\mathbb{R}^n$	$\mathbb{R}$ Menge der reellen Zahlen, $\mathbb{R}^n = \{x = (x_1, ..., x_n)^T \vert x_i \in \mathbb{R}, \forall i \in \{1, ...n\}\}$
$\mathbb{R}_0$, $\mathbb{R}_0^+$	$\mathbb{R}_0 = \mathbb{R} \cup \{0\}$, $\mathbb{R}_0^+ = \{x \mid x \geq 0 \text{ und } x \in \mathbb{R}\}$,
$(\mathbb{R}_0^+)^n$	$(\mathbb{R}_0^+)^n = \{x = (x_1, ..., x_n)^T \vert \ x_i \in \mathbb{R}_0^+ \ \forall i \in \{1, ..., n\}\}$
$\mathbb{N}$, $\mathbb{N}_0$	$\mathbb{N}$ Menge der natürlichen Zahlen ohne die Null, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$
$\mathbb{1}$	$\mathbb{1} \in \mathbb{R}^n$ n-dimensionaler Vektor, dessen Einträge alle Eins sind
X	$X = (x_{ij})_{ij} \in \mathbb{R}^{n \times p}$ die Systemmatrix, $i = 1, ..., n$, $j = 1, ..., p$
y	$y = (y_1, ..., y_n)$ der n-dimensionaler Vektor als die abhängige Variable bzw. der Output, dessen Art vom vorliegenden Optimierungsproblem abhängt
y_i	Output zur Beobachtung x^i, $i = 1, ..., n$
n, p	n Anzahl der Beobachtungen, p Anzahl der Variablen (Merkmale)
x^i	$x^i \in \mathbb{R}^p$, $i = 1, ..., n$ die Beobachtung i, entspricht der Zeile i von X
X_j	Spalte j der Matrix X, $j \in \{1, ..., p\}$
X_E	Spalten von X, die in der Indexmenge $E \subseteq j \in \{1, ..., p\}$ liegen, $X_E = (X_j)_{j \in E}$
β	$\beta = (\beta_1, ..., \beta_p)^T \in \mathbb{R}^p$ ein Regressionsvektor
$\text{supp}(\beta)$	$\text{supp}(\beta) = \{j \in \{1, ..., p\} \vert \ \beta_j \neq 0\}$ der Support von β
β^*	die optimale Lösung der klassischen linearen Regression $y = X\beta^* + \epsilon$ bzw. eines generalisierten linearen Modells
J^*	$J^* = \text{supp}(\beta^*)$
$\bar{y}$	$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ Mittelwert von $y \in \mathbb{R}^n$
$\bar{X_j}$	$\bar{X_j} = \frac{1}{n} \sum_{i=1}^n x_{ij}$ Mittelwert der Spalte X_j
σ_j	$\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (X_j - \bar{X_j})^2}$ (empirische) Standardabweichung von X_j
σ_y	$\sigma_y = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2}$ (empirische) Standardabweichung von y
$\tilde{\beta}_j^*$	$\tilde{\beta}_j^* = \frac{\sigma_j}{\sigma_y} \beta_j^* \ \forall j \in \{1, ..., p\}$ Koeffizienten des standardisierten Systems
$\mathcal{N}(\mu, \sigma^2 I_n)$	Normalverteilung einer n-dimensionalen Zufallsvariable mit Erwartungswert μ und Varianz σ^2
ϵ	$\epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$ der aus der linearen Regression resultierende Fehler
$\mathbb{E}(\cdot)$	Erwartungswert einer Zufallsvariable
$\text{rang}(X)$	das Minimum der Anzahl linear unabhängiger Zeilen bzw. Spalten von X, $\text{rang}(X) = \text{rang}(X^T) \leq \min\{n, p\}$
$\text{Kern}(X)$	$\text{Kern}(X) = \{z \in \mathbb{R}^p \vert Xz = 0\}$
I_n, I_p	n-dimensionale bzw. p-dimensionale Einheitsmatrix

X^{-1}	Inverse von X mit $X^{-1}X = XX^{-1} = I_p$, falls $n = p$ und rang(X)=p gilt
X^+	Pseudoinverse von X
$e^{(\cdot)}$	natürlich Exponentialfunktion
$\|\beta\|_0$	$\|\beta\|_0 = \|\{j \mid \beta_j \neq 0\}\|$ Anzahl der Nichtnulleinträge von β
$\|\beta\|_1$	$\|\beta\|_1 = \sum_{j=1}^p \|\beta_j\|$ L1-Norm von β
$\|\beta\|_2$	$\|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$ euklidische Norm (L2-Norm) von β
$\|\beta\|_\infty$	$\|\beta\|_\infty = \max_{j\in\{1,...p\}} \|\beta_j\|$ der betragsmäßig größte Koeffizient von β
$\|\beta\|_{(j)}$	der betragsmäßig j-größte Eintrag von β
λ, $\boldsymbol{\lambda}$	$\lambda \geq 0$, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_p)$ mit $\lambda_1 \geq ... \geq \lambda_p \geq 0$ Regularisierungsparameter
$J_{\boldsymbol{\lambda}}(\beta)$	$J_{\boldsymbol{\lambda}}(\beta) = \sum_{j=1}^p \lambda_j \|\beta\|_{(j)}$ sortierte L1-Norm
$f_{OLS}(\beta)$	$f_{OLS}(\beta) = \|y - X\beta\|_2^2 = \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij}\beta_j)^2$ Summe der kleinsten Quadrate
β_{OLS}	$\beta_{OLS} \in \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} f_{OLS}$
$f_{LASSO}(\beta)$	Optimierungsfunktion von LASSO, hängt von der Art von y ab
β_L, $\beta_{E,L}$	$\beta_L = (\beta_{1,L}, ..., \beta_{p,L})^T \in \operatorname{argmin}\ f_{LASSO}(\beta)$, $\beta_{E,L} = (\beta_{j,L})_{j\in E}^T$
J	$J = \operatorname{supp}(\beta_L) = \{j \in \{1, ..., p\} \mid \beta_{j,L} \neq 0\}$
$f_{SLOPE}(\beta)$	$f_{SLOPE}(\beta) = \min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + J_{\boldsymbol{\lambda}}(\beta)$ Optimierungsfunktion von SLOPE
β_S, $\beta_{E,S}$	$\beta_S = (\beta_{1,S}, ..., \beta_{p,S})^T \in \operatorname{argmin}\ f_{SLOPE}(\beta)$, $\beta_{E,S} = (\beta_{j,S})_{j\in E}^T$
$\hat{\beta}$	Näherungslösung zu β_L bzw. β_S
$\beta^{(k)}$	Index k oben in Klammern bezeichnet eine in der Iteration k eines Algorithmus berechnete Größe
$S(x, \lambda)$	Soft-Thresholding Operator, $S(x, \lambda) = \operatorname{sign}(x)(\|x\| - \lambda)_+$
$O(\cdot)$	Landau-Symbol; $f(x) = O(g(x))$ bedeutet: $\forall C > 0\ \exists \delta > 0\ \forall \|x\| < \delta$: $\|f(x)\| \leq C \cdot \|g(x)\|$, d.h. $f(x)$ wächst nicht erheblich schneller als $g(x)$
$o(\cdot)$	Landau-Symbol; $f(x) = o(g(x))$ bedeutet: $\forall C > 0\ \exists \delta > 0\ \forall \|x\| < \delta$: $\|f(x)\| < C \cdot \|g(x)\|$, d.h. $f(x)$ wächst langsamer als $g(x)$
$\nabla f(\beta)$	Gradient bzw. Subdifferenzial der Funktion $f(\beta)$
$\nabla f(\hat{\beta})$	Gradient bzw. Subdifferenzial der Funktion $f(\beta)$ ausgewertet in $\hat{\beta}$
$\frac{\partial f(\beta)}{\partial \beta_j}$	partielle Ableitung der Funktion $f(\beta)$ nach β_j
$\frac{\partial f}{\partial \beta_j}(\hat{\beta}_j)$	partielle Ableitung der Funktion $f(\beta)$ nach β_j ausgewertet in $\hat{\beta}_j$
$\|X^TX\|_2$	$\|X^TX\|_2 = \max_{\|b\|_2=1} \|Xb\|_2$ Spektralnorm von $X^TX \in \mathbb{R}^{p\times p}$, wobei $\|X^TX\|_2 = \|\mu_{max}(X^TX)\|$ gilt

$\|X\|_\infty$	$\|X\|_\infty = \max_{b\neq 0} \frac{\|Xb\|_\infty}{\|b\|_\infty}$ Maximumsnorm von X
$\|\mu_{max}(X^TX)\|$	der betragsmäßig größte Eigenwert von X^TX
$\|\beta\|$	falls $\beta \in \mathbb{R}$, so entspricht $\|\beta\| = \max\{0, \beta\}$ dem Betrag von β; ist $\beta \in \mathbb{R}^p$, so ist $\|\beta\| = (\|\beta_1\|, ..., \|\beta_p\|)^T$
$\|E\|$, $\|J\|$, $\|M\|$	für Mengen E, J, M entspricht $\|E\|$, $\|J\|$, $\|M\|$ deren Mächtigkeit
$0_{\|p\|}, 0_{\|p-\|J\|\|}$	$p-$ bzw. $p - \|J\|$-dimensionaler Vektor, dessen Einträge alle Null sind
$\Phi(\alpha)$	$\Phi(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-\frac{1}{2}t^2} dt$ Verteilungsfunktion der Standardnormalverteilung
$\Phi^{-1}(\alpha)$	das α-Quantil der Standardnormalverteilung
$\text{diag}(d_1, ..., d_p)$	$p \times p$-dimensionale Diagonalmatrix mit Diagonaleinträgen d_1,...,d_p
$\boldsymbol{\lambda}_{BH}$	$\boldsymbol{\lambda}_{BH} = (\lambda_{BH}(1), ..., \lambda_{BH}(p)))$ die mittels der BH-Prozedur berechnete Sequenz der Regularisierungsparameter für SLOPE
$\boldsymbol{\lambda}_G$	$\boldsymbol{\lambda}_G = (\lambda_G(1), ..., \lambda_G(p))$ Sequenz basierend auf $\boldsymbol{\lambda}_{BH}$
θ	$\theta \in \mathbb{R}^n$ ein Regressionsvektor
$f^D_{LASSO}(\theta)$	$f^D_{LASSO}(\theta) = -\frac{1}{2}\|\theta\|_2^2 + \theta^T y$ Optimierungsfunktion des dualen Problems zu LASSO
$f^D_{SLOPE}(\theta)$	$f^D_{SLOPE}(\theta) = -\frac{1}{2}\|\theta\|_2^2 + \theta^T y$ Optimierungsfunktion des dualen Problems zu SLOPE
θ^*	optimale Lösung des dualen Problems zu SLOPE
$\hat{\theta}$	Näherungslösung zu θ^*
$C_{\boldsymbol{\lambda}}$	zulässige Menge für das duale Problem zu SLOPE $\theta \in C_{\boldsymbol{\lambda}} \quad :\Longleftrightarrow \quad \sum_{i\leq j} \|\theta\|_{(i)} \leq \sum_{i\leq j} \lambda_i \ \forall j \in \{1, ..., p\}$
$\delta(\hat{\beta})$	Dualitätslücke, $\delta(\hat{\beta}) = (X\hat{\beta})^T(X\hat{\beta} - y) + J_{\boldsymbol{\lambda}}$
$i!$	$i! = i \cdot (i-1) \cdot ... \cdot 2 \cdot 1$ die Fakultät von i, $i \in \mathbb{N}_0$, $0! := 1$
$D^i f$	das i-te Differential einer bis mindestens Ordnung i differenzierbaren Funktion f
K	Anzahl der Klassen bei der multinomialen Regression
N	die für die CV verwendete Anzahl an Teilmengen
T_1,...,T_N	die für die CV verwendeten Teilmengen, $T_i \subseteq \{1, ..., n\}$
$\lfloor \frac{n}{N} \rfloor$	$\lfloor \frac{n}{N} \rfloor = \min\{i \in \mathbb{N} \| \frac{n}{N} \geq i\}$
$\lceil \frac{n}{N} \rceil$	$\lceil \frac{n}{N} \rceil = \min\{k \in \mathbb{N} \| \frac{n}{N} \leq i\}$
$y_{(T_i)}$	$(y_k)_{k\in T_i}$
$X^{(T_i)}$	$X^{(T_i)} = (x_{ij})_{i\in T_i}$, Teilmatrix von X bestehend aus Zeilen in T_i
$\hat{\beta}^{(-T_i)}(\lambda)$	Lösungsvektor, für dessen Bestimmung $\{T_j\}_{j\neq i}$ verwendet wurden

1 Einleitung

Die Möglichkeiten der Aufbewahrung, Speicherung und Zugänglichkeit zu Datenmengen bewirkt viele Durchbrüche in zahlreichen Fachgebieten, sei es Medizin, Unterhaltung oder Industrie. Gleichzeitig stellt die Menge an Daten eine große Herausforderung dar, denn darunter befinden sich viele redundante Informationen. Daher bedarf es effizienter Techniken damit umzugehen, vor allem um an die Kerninformationen zu gelangen.

Die Verfügbarkeit von Daten führt in vielen wissenschaftlichen Problemstellungen zu einem Regressionsproblem, bei dem eine Beziehung zwischen einer oder mehreren unabhängigen Einflussgrößen und einer abhängigen Zielgröße herzustellen ist. Regressionsanalysen erlauben sowohl eine quantitative Beschreibung der Zusammenhänge als auch eine Vorhersage von Werten der abhängigen Variable für neue Beobachtungen. Einige Anwendungen sind die Klassifikation von Texten und Bildern, kombinatorische Chemie, Untersuchung von Genexpressionsdaten, und viele andere. Häufig überschreitet dabei die Anzahl der Merkmale deutlich die der vorhandenen Beobachtungen. Dadurch wird die Regression erschwert und es erfordert das Aussortieren irrelevanter Informationen. Aufgrund dessen gewinnt Variablenselektion immer mehr an Bedeutung.

Die Medizin beschäftigt sich seit langer Zeit intensiv mit der Analyse von Genprofilen von Menschen mit diversen Krankheiten, darunter Krebs. Die Anzahl der Gene je nach Art des Gentyps liegt im Zehntausender-Bereich. Die Vermutung, dass nicht alle Gene bei der Entstehung einer Krankheit eine Rolle spielen, liegt nahe. Das Ziel ist eine möglichst treffende Auswahl von Genen, die Charakteristika von Patientenprofilen erfassen, als Biomarker zur Prognose dienen und somit zur Behandlung eingesetzt werden können.

Diese Arbeit behandelt zwei Ansätze zur Variablenselektion. Zum einen werden diese theoretisch einander gegenübergestellt, zum anderen findet deren praktische Anwendung auf synthetisch generierte Daten und Genexpressionen von Krebspatienten statt. Dazu werden Datensätze von Critical Assessment of Massive Data Analysis (CAMDA) und The Cancer Genome Atlas (TCGA) herangezogen. CAMDA Daten beinhalten Array-based Comparative Genomic Hybridization (Array-CGH) Daten und Genexpressionsdaten in Form von Gesamt-Transkriptom-Shotgun-Sequenzierung (RNA-Seq) für 498 Kinder mit Neuroblastom. TCGA umfasst insgesamt 38 Krebsarten, von denen Genexpressionen in Form von microRNA von Brustkrebs untersucht werden.

1.1 Problemstellung

Es liegt ein Regressionsproblem vor, sodass ein Zusammenhang zwischen unabhängigen Einflussgrößen $x^1, ..., x^n$, $x^i \in \mathbb{R}^p$, $i = 1, ..., n$, $n \geq 1$, und einer abhängigen Zielgröße $y = (y_1, ..., y_n)^T \in \mathbb{R}^n$, herzustellen ist. Hierbei entspricht n der Anzahl der Beobachtungen und p der Anzahl der Variablen bzw. Merkmale. Es gelte die zentrale Annahme, dass ein linearer Zusammenhang besteht.

Die Funktion, die den Zusammenhang der Einflussgrößen und Zielgröße modelliert, hängt von der Art von y ab. Die Zielgröße y ist kontextspezifisch, kann sowohl reellwertig (quantitativ,) als auch diskret mit zwei oder mehr Ausprägungen sein. Manche Probleme, die bspw. aus dem Bereich der Medizin kommen, können einer Zielgröße y bedürfen, die wiederum eine andere Form hat, siehe Kapitel 2.4.2. Ist y reellwertig, so wird die klassische lineare Regression betrachtet, andernfalls liegt ein *generalisiertes lineares Modell* (GLM) vor, siehe Kapitel 2.4. Im Folgenden sei $X := (x^1, ..., x^n)^T = (x_{ij})_{ij} \in \mathbb{R}^{n \times p}$ die Systemmatrix, sodass x_{ij} dem Eintrag der Beobachtung i für das Merkmal j entspricht.

Der Output y sei quantitativ, sodass es sich um das klassische lineare Modell handelt:

$$y = \beta_0^* \mathbb{1} + X\beta^* + \epsilon \ . \tag{1.1}$$

$\beta^* = (\beta_1^*, ..., \beta_p^*)^T \in \mathbb{R}^p$ ist der unbekannte optimale Regressionsvektor, $\mathbb{1}$ bezeichnet den n-dimensionalen Vektor, dessen Einträge alle Eins betragen. Somit bestimmt p die Dimension des Problems. Es bestehe die Annahme, dass das optimale β^* dünnbesetzt ist, sodass nur wenige Einträge nicht Null sind. Die Konstante $\beta_0^* \in \mathbb{R}$ ist der Achsenabschnitt (engl. intercept) und $\epsilon = (\epsilon_1, ..., \epsilon_n)^T \in \mathbb{R}^n$ ist der resultierende Fehlervektor (Residuum) mit der Annahme $\epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$, sodass $y \sim \mathcal{N}(X\beta^*, \sigma^2 I_n)$. Für den Achsenabschnitt β_0^* gilt

$$\beta_0^* = \bar{y} - \sum_{j=1}^{p} \bar{X_j}\beta_j^* \ , \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \ \bar{X_j} = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \ . \tag{1.2}$$

X_j bezeichne die Spalte j von X, sodass $X = (X_1, ..., X_p)$. Es ist üblich, vorab eine Standardisierung der Matrix X vorzunehmen, sodass jede Variable (Spalte) den Erwartungswert Null und Varianz Eins hat[1], $\bar{X_j} = 0$ und $\frac{1}{n}\|X_j\|_2^2 = 1 \ \forall j = 1, ..., p$ [9, 21, 15]. Die Lösungen des ursprünglichen und standardisierten Systems lassen sich problemlos ineinander umrechnen. Bezeichnet σ_j bzw. σ_y die Standardabweichung der Spalte X_j bzw. die Standardabweichung von y vor Standardisierung des Systems, so sind die Koeffizienten des standardisierten Systems durch $\tilde{\beta}_j^* = \frac{\sigma_j}{\sigma_y}\beta_j^*$

[1] Hier entspricht der Erwartungswert dem empirischen arithmetischen Mittel, die Varianz der empirischen Varianz.

$\forall j \in \{1, ..., p\}$ gegeben. Die Standardisierung ist vor allem dann sinnvoll, wenn die Variablen in verschiedenen Einheiten gegeben sind, bspw. in Meter und Zentimeter, da erst durch die Standardisierung die Vergleichbarkeit der Variablen möglich ist [9, 21]. O.b.d.A. gelte zusätzlich $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i = 0$. Damit folgt aus der Gleichung für den Achsenabschnitt (1.2), dass $\beta_0^* = 0$ gilt, und dieser folglich eliminiert werden kann [21]. O.b.d.A. gelte im Folgenden $\beta_0 = 0$. Es ist zu beachten, dass bei GLM der Achsenabschnitt die Gleichung (1.2) nicht erfüllt und daher berücksichtigt werden muss, siehe Kapitel 2.4.

Es ist oft nicht möglich, β^* exakt zu bestimmen, sodass dieser mit einem Lösungsverfahren approximiert werden muss. Eine der gängigsten Vorgehensweisen ist der Einsatz der Methode der kleinsten Quadrate (engl. Ordinary Least Squares, OLS),

$$\min_{(\beta) \in \mathbb{R}^p} \left\{ f_{OLS}(\beta) := \|y - X\beta\|_2^2 \right\} . \tag{1.3}$$

Sei $\beta_{OLS} \in \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} f_{OLS}(\beta)$ eine Lösung der kleinsten Quadrate.

Die Bestimmung von β_{OLS} als Approximation von β^* kann numerisch aufwendig sein, vgl. Kapitel 2.2, außerdem ist β_{OLS} zumeist vollbesetzt. Viele Anwendungen fordern jedoch einen dünnbesetzten Lösungsvektor, sodass Variablenselektion erfolgen soll. Denn ist der j-te Eintrag eines Lösungsvektors Null, so gehört das j-te Merkmal nicht mehr zum Modell, und umgekehrt. Das gesuchte β^* enthält somit das wahre Modell.

Die Notwendigkeit von Variablenselektion ist auf das sogenannte *Verzerrung-Varianz-Dilemma* (engl. bias-variance tradeoff) zurückzuführen [19, 21, 35]. Für ein β_{OLS} ist die L2-Norm des Residuums $\|y - X\beta_{OLS}\|_2^2$ zwar gering, weist hingegen eine hohe Varianz auf. Denn zum einen können die Koeffizienten von β_{OLS} betragsmäßig sehr hoch sein und damit die Varianz steigern, da diese Koeffizienten starke Schwankungen des Outputs selbst bei geringen Änderungen der Eingabedaten verursachen können, falls diese bspw. leicht verrauscht sind. Zum anderen kommt es meist zu Überanpassung (engl. overfitting), insbesondere bei $n \ll p$. Es besteht die Gefahr, dass viele irrelevante Merkmale im Modell verbleiben. Diese Sachverhalte führen zu einer mangelnden Vorhersagegüte für neue Beobachtungen. Es wird eine Balance zwischen den beiden Größen angestrebt, indem möglichst nur diejenigen Merkmale selektiert werden, die y beeinflussen, wodurch sich die hohe Dimension p (enorm) reduzieren kann. Ein Problem hierbei ist die Schätzung der Standardabweichung σ, was in Abhängigkeit des Verhältnisses der Dimensionen n und p recht komplex sein kann. Die Untersuchung, wie die Verteilung von y und damit σ zu schätzen ist, ist kein Bestand der Arbeit.

Diese Arbeit beschäftigt sich mit zwei Methoden zur Variablenselektion, *Least*

Absolute Shrinkage and Selection Operator (LASSO) und *Sorted L-One Penalized Estimation* (SLOPE). LASSO und SLOPE zielen auf einen dünnbesetzten Regressionsvektor ab, der unter Berücksichtigung der Regularisierung (engl. regularization, penalized estimation), die Überanpassung verhindern soll, eine Approximation von β^* (und ebenso von β_{OLS}) darstellt und dessen Koeffizienten betragsmäßig klein sind [5, 35].

Bemerkung: Für LASSO und dessen Lösungsverfahren existiert eine Transformation für das Vorliegen komplexer Daten $X \in \mathbb{C}^{n \times p}$ und $y \in \mathbb{C}^n$ [1, 29], für SLOPE erfolgte die Transformation bislang noch nicht. Durch den Imaginäranteil verdoppelt sich die Dimension auf $2p$ und somit kann sich der numerische Aufwand der Lösungsalgorithmen in Abhängigkeit der Größenordnung von p stark erhöhen. Diese Arbeit beschränkt sich auf $\mathbb{R}$.

1.2 Gliederung

Das Kapitel 2 beschäftigt sich mit LASSO. Die Kapitel 2.1 bis 2.3.3 behandeln die klassische lineare Regression. Zunächst erfolgt eine kurze Vorstellung des Optimierungsproblems. Daraufhin gibt es einen Einblick in die bekanntesten Regularisierungsansätze, die vor LASSO erarbeitet wurden (2.1). Im Kapitel 2.2 steht die Charakterisierung von Lösungen von LASSO und deren Eigenschaften im Vordergrund. Anschließend werden die gängigsten Lösungsalgorithmen im Kapitel 2.3 und deren Vor- und Nachteile vorgestellt. Daraufhin wird LASSO im Kapitel 2.4 in Verbindung mit generalisierten linearen Modellen und möglichen Lösungsverfahren behandelt. Als nächstes werden im Kapitel 2.5 Methoden aufgezeigt, wie Merkmale a priori eliminiert werden können. Abschließend wird im Kapitel 2.6 das Verhalten von LASSO besprochen.

Das Kapitel 3 handelt von SLOPE. Zu Beginn gibt es eine kurze Einführung in das Optimierungsproblem. Kapitel 3.1 setzt sich mit SLOPE bei orthogonalen Designs auseinander, woraufhin im Kapitel 3.2 die Herleitung einer möglichen Sequenz der Regularisierungsparameter erfolgt. Zum Schluss werden Lösungsalgorithmen für SLOPE vorgestellt. Es sei angemerkt, dass SLOPE bisher wenig erforscht ist. Aufgrund dessen kann SLOPE nicht in dem Umfang wie LASSO dargelegt werden.

Kapitel 4 besteht in der praktischen Umsetzung von LASSO und SLOPE in der Programmiersprache R. Dabei wird sowohl das Verhalten der beiden Ansätze vor allem hinsichtlich der Konsistenz der Variablenselektion geprüft als auch der Aufwand als benötigte Rechenzeit miteinander verglichen.

Im Kapitel 5 werden die wichtigsten Ergebnisse zusammengefasst, beide Ansätze kritisch beleuchtet und es gibt einen Ausblick auf mögliche zukünftige Untersuchun-

gen und Verbesserungspotenzial.

2 LASSO

LASSO wurde erstmals von Tibshirani (1996) ausgearbeitet [35]. Die Idee besteht darin, die Summe der kleinsten Quadrate f_{OLS} (1.3) zu minimieren und gleichzeitig die L1-Norm des gesuchten Regressionsvektors durch ein $R > 0$ zu beschränken. Diese Beschränkung dient der Verhinderung von Overfitting und einer möglichst kleinen Auswahl von Merkmalen. Das zu lösende Optimierungsproblem LASSO lautet folglich [35]:

$$\min_{\beta \in \mathbb{R}^p} \left\{ f_{OLS}(\beta) = \|y - X\beta\|_2^2 \right\} \quad \text{s.t.} \quad \|\beta\|_1 \leq R \; . \tag{2.1}$$

Die Nebenbedingung $\|\beta\|_1 \leq R$ entspricht der Regularisierung, die einige Koeffizienten des gesuchten Regressionsvektors auf Null setzt. Die Anzahl der Nulleinträge steigt für jedes $R < \max \|\beta_{OLS}\|_1$ [19, 35]. Graphisch bedeutet die Nebenbedingung, dass der Definitionsbereich für β einen p-dimensionalen Hyperoktaeder[2] darstellt.

LASSO (2.1) lässt sich durch Einführung eines Lagrange-Multiplikators λ äquivalent schreiben als

$$\min_{\beta \in \mathbb{R}^p} \underbrace{\|y - X\beta\|_2^2}_{\text{Ziel-/Verlustfunktion}} + \underbrace{\lambda \|\beta\|_1}_{\text{Regularisierer}} \quad , \quad \lambda > 0 \; , \tag{2.2}$$

wobei λ den sogenannten *Regularisierungsparameter* (RP) darstellt und auf einen fest gewählten Wert gesetzt wird [9, 19, 21]. Demnach ist die Beschränkung der L1-Norm von β durch ein R äquivalent zur L1-Regularisierung, für den Beweis siehe Anhang A3. Diese Äquivalenz folgt aus der Lagrange-Dualität, sodass für jedes R das zugehörige λ eindeutig ist und umgekehrt. Die Lösungen von LASSO als ein Optimierungsproblem mit Nebenbedingungen (2.1) und LASSO in der Lagrange-Form (2.2) stimmen überein [3]. Es gibt allerdings keine Formel, mit der sich λ und R ineinander umrechnen lassen, es kann lediglich die Existenz und die Eindeutigkeit dieser Größen zueinander gezeigt werden.

Die Zielfunktion steht für die Höhe des Fehlers bzw. des Verlusts bei inkorrekten Vorhersagen und wird daher auch Verlustfunktion genannt. Der zweite Term ist der Regularisierer. Ist das λ zu klein, so findet keine Variablenselektion statt. Im Umkehrschluss bewirkt jedes hinreichend große λ Variablenselektion, weil mit der Höhe der RP die Anzahl der Nulleinträge des gesuchten Regressionsvektors steigt,

[2] Ein Hyperoktaeder ist eine Verallgemeinerung eines dreidimensionalen Oktaeders auf Räume beliebiger Dimension.

sodass ab einer bestimmten Höhe dieser dem Nullvektor entspricht [5, 19, 35].

In der Literatur sind alternative Definitionen für die Verlustfunktion zu finden, und zwar

$$\frac{1}{2}\|y - X\beta\|_2^2 \;\; [5,\; 36]\;, \quad \frac{1}{n}\|y - X\beta\|_2^2 \;\; [27,\; 9]\;, \quad \frac{1}{2n}\|y - X\beta\|_2^2 \;\; [21,\; 15]\;. \tag{2.3}$$

Für den gesuchten Lösungsvektor und die Darstellung von LASSO in der ursprünglichen Form (2.1) ist dies irrelevant. Sei λ der Parameter entsprechend der ursprünglichen Verlustfunktion $\|y - X\beta\|_2^2$, vgl. LASSO in der Lagrange-Form (2.2). Die jeweiligen Vorfaktoren der Verlustfunktionen in (2.3) führen wegen

$$\begin{aligned}
\min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda_1\|\beta\|_1 \quad &\Longleftrightarrow \quad \min_{\beta\in\mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \frac{\lambda}{2}\|\beta\|_1 \\
\min_{\beta\in\mathbb{R}^p} \frac{1}{n}\|y - X\beta\|_2^2 + \lambda_2\|\beta\|_1 \quad &\Longleftrightarrow \quad \min_{\beta\in\mathbb{R}^p} \frac{1}{n}\|y - X\beta\|_2^2 + \frac{\lambda}{n}\|\beta\|_1 \qquad (2.4) \\
\min_{\beta\in\mathbb{R}^p} \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda_3\|\beta\|_1 \quad &\Longleftrightarrow \quad \min_{\beta\in\mathbb{R}^p} \frac{1}{2n}\|y - X\beta\|_2^2 + \frac{\lambda}{2n}\|\beta\|_1 \qquad (2.5)
\end{aligned}$$

hingegen für LASSO in der Lagrange-Form zu einer anderen Skalierung des RP's λ. Insbesondere die Verwendung der Optimierungsfunktionen (2.4) und (2.5) erleichtert damit die Interpretation und die Vergleichbarkeit von verschiedenen Werten für den RP für unterschiedliche Anzahl von Beobachtungen [21]. Im Folgenden sei LASSO definiert als

$$\min_{\beta\in\mathbb{R}^p} \left\{ f_{LASSO}(\beta) := \frac{1}{2n}\|y - X\beta\|_2^2 \;+\; \lambda\|\beta\|_1 \right\}. \tag{2.6}$$

Ein Regressionsvektor, der eine Lösung von LASSO (2.6) darstellt, sei definiert als

$$\beta_L \in \operatorname*{argmin}_{\beta\in\mathbb{R}^p} \left\{ f_{LASSO}(\beta) \;= \frac{1}{2n}\|y - X\beta\|_2^2 \;+\; \lambda\|\beta\|_1 \right\}. \tag{2.7}$$

Alle folgenden Ausführungen wurden an die Optimierungsfunktion f_{LASSO} angepasst, indem Formeln um den Vorfaktor $\frac{1}{n}$ bzw. n ergänzt wurden.

2.1 Vorgänger und Elastic Net

Vor LASSO haben zahlreiche Autoren Ansätze, die OLS in Verbindung mit Regularisierung lösen, vorgeschlagen. Bekannte Beispiele sind Subset Selection [8], definiert als

$$\min_{\beta\in\mathbb{R}^p} \left\{ f_{OLS}(\beta) = \|y - X\beta\|_2^2 \right\} \;\; \text{s.t.} \;\; \|\beta\|_0 \le k, \quad k \in \mathbb{N}\;, \tag{2.8}$$

und Ridge Regression [22], definiert als

$$\min_{\beta \in \mathbb{R}^p} \left\{ f_{OLS}(\beta) = \|y - X\beta\|_2^2 \right\} \text{ s.t. } \|\beta\|_2 \leq R \, , R > 0 \, . \tag{2.9}$$

Durch Einführung eines Lagrange-Multiplikators λ lassen sich die Optimierungsprobleme (2.8) und (2.9) umformulieren zu

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_0 \, , \quad \lambda > 0 \, , \tag{2.10}$$

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2 \, , \quad \lambda > 0 \, . \tag{2.11}$$

Zur Lösung der Probleme wird gewöhnlich die jeweilige Lagrange-Form (2.10) bzw. (2.11) herangezogen.

Subset Selection (2.8) erlaubt bis zu k Nichtnulleinträge des resultierenden Lösungsvektors und legt somit die Dünnbesetztheit vorab fest. Das Optimierungsproblem (2.10) ist allerdings wegen $\| \cdot \|_0$ nicht konvex und zudem NP-schwer, sodass für die Rechenzeit $t = t(p)$ gilt: $\nexists m \in \mathbb{N}$: $t = O(p^m)$ [37].

Ridge Regression zielt auf einen Regressionsvektor mit betragsmäßig kleinen Einträgen ab, indem dessen L2-Norm beschränkt wird. Der Vorteil von Ridge Regression liegt in der Schrumpfung der Koeffizienten und damit der Verringerung der Varianz des Outputs. Denn wie bereits in der Problemstellung erläutert bewirken die betragsmäßig hohen Einträge von β_{OLS} Schwankungen des Outputs. Die Zielfunktion in (2.11) ist konvex, differenzierbar und hat eine geschlossene Form der Lösung. Da allerdings nur wenige oder keine Koeffizienten des resultierenden Lösungsvektors den Wert Null haben, findet keine Variablenselektion statt.

Die Erklärung dafür, dass LASSO im Gegenteil zu Ridge Regression einen dünnbesetzten Lösungsvektor liefert, liegt in der Geometrie der L1- und L2-Normen [21, 35, 37]. Abbildung 1 veranschaulicht dies. Der Bereich für zulässige Lösungen von LASSO als ein p-dimensionaler Hyperoktaeder hat Ecken, wohingegen die L2-Regularisierung als p-dimensionale Kugel keine hat. Eine Lösung für LASSO bzw. Ridge Regression ist gefunden, sobald die Kurven der Summe der kleinsten Quadrate in Abhängigkeit von der Entfernung von β_{OLS} den jeweiligen zulässigen Bereich berühren. Wird bei LASSO eine Ecke getroffen, was oft auftritt, so hat β_L viele Nulleinträge und die entsprechenden Merkmale werden eliminiert.

Bei stark korrelierten Merkmalen neigt LASSO dazu, nur wenige oder sogar nur eines dieser Merkmale zu selektieren, sodass der Lösungsvektor β_L in solchen Fällen zu dünnbesetzt ist und dadurch wichtige Informationen verloren gehen können. Zou und Hastie (2005) [46] schlagen daher vor, LASSO und Ridge Regression zu kombinieren, indem der Regularisierungsterm als eine Konvexkombination der L1- und

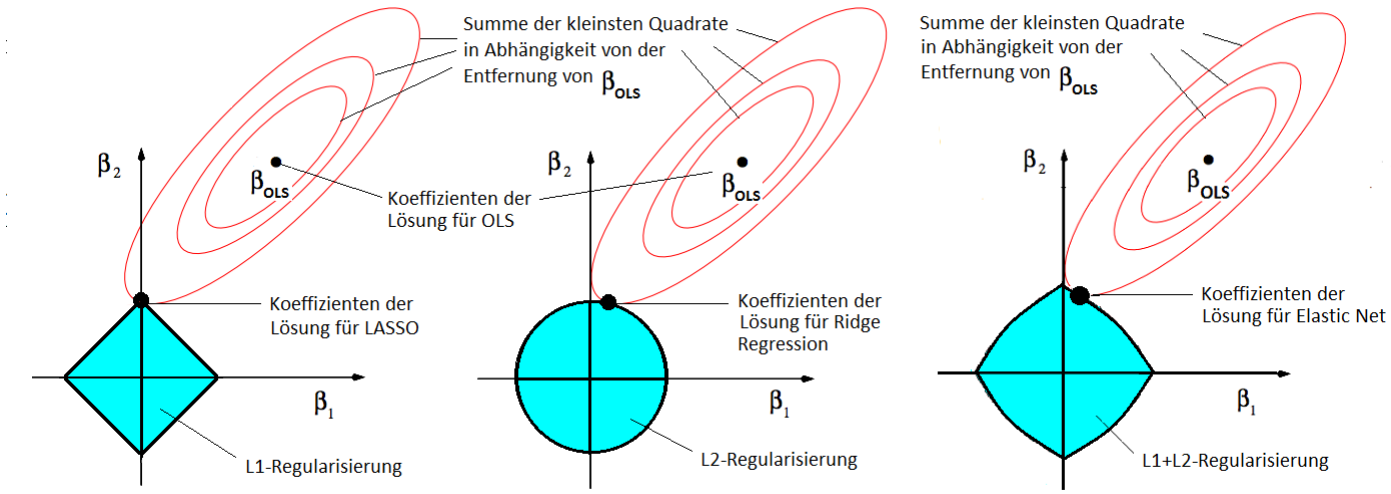


Abbildung 1 Geometrie von LASSO, Ridge Regression und Elastic Net in 2D. LASSO (links), Ridge Regression (Mitte) und Elastic Net (rechts). Modifiziert nach: [21], S.11.

L2-Normen zusammengesetzt wird, woraus das sogenannte *Elastic Net* entsteht

$$\lambda P_\alpha(\beta) \ = \lambda \left((1-\alpha)\|\beta\|_2 + \alpha\|\beta\|_1\right) = \lambda \sum_{j=1}^{p} \left((1-\alpha)|\beta_j|^2 + \alpha|\beta_j|\right), \quad \alpha \in [0,1] \ .$$

Bei $\alpha = 0$ reduziert sich $\lambda P_\alpha(\beta)$ auf die L2-Regularisierung, ist $\alpha = 1$, so liegt L1-Regularisierung vor. Für $\alpha \to 0$ werden die Koeffizienten betragsmäßig kleiner und es werden zunehmend mehr Merkmale selektiert, bei $\alpha \to 1$ hingegen erhöhen sich die Koeffizienten des Regressionsvektors betragsmäßig im Vergleich zu Ridge Regression und die Anzahl der ausgewählten Merkmale sinkt.

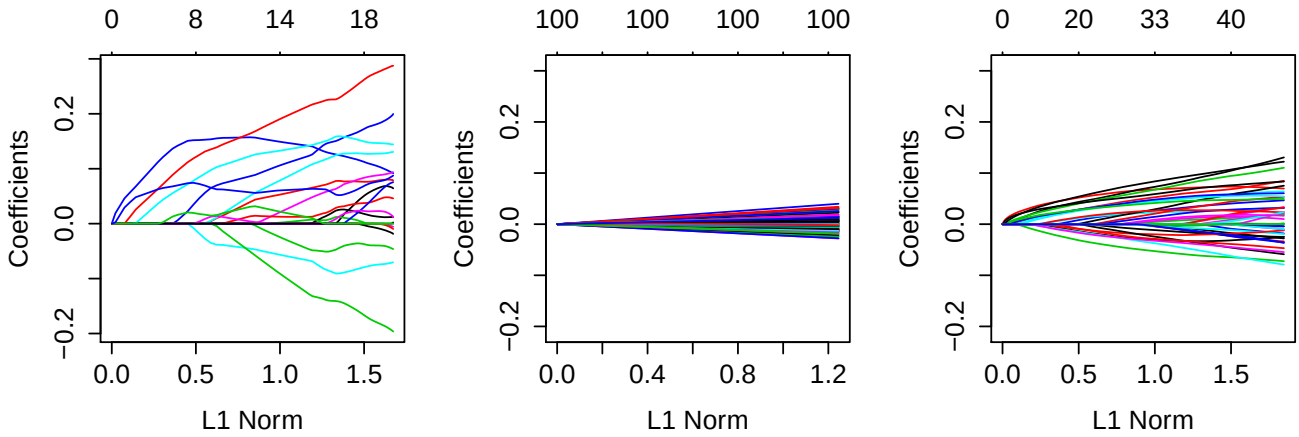


Abbildung 2 Vergleich der Koeffizienten von LASSO, Ridge Regression und Elastic Net. LASSO (links), Ridge Regression (Mitte) und Elastic Net (rechts) mit $\alpha = 0.1$. Die Abbildung stellt jeweils die L1-Norm (horizontale Achse) den Werten der Koeffizienten (vertikale Achse) gegenüber. Oberhalb ist jeweils die Anzahl der Nichtnulleinträge des Regressionsvektors abgebildet.

Dies wird in der Abbildung 2 veranschaulicht. Hier wurden jeweils für eine absteigende Sequenz von RP die Koeffizienten für ein Problem mit $n = 20$ Beobachtungen und $p = 100$ Variablen mittels des R-Pakets **glmnet** berechnet. Die Berechnungen starten bei einem λ_0, für das alle Koeffizienten Null sind. Jeder Pfad entspricht dem Lösungspfad eines Koeffizienten. LASSO liefert Lösungen mit den meisten Null- und betragsmäßig hohen Einträgen. Ridge Regression bringt betragsmäßig sehr kleine Einträge hervor, die Lösungen sind dennoch nicht dünnbesetzt, für jeden RP der Sequenz bis auf λ_0 sind alle 100 Merkmale im Modell enthalten. Die Koeffizienten bei Elastic Net liegen zwischen denen für LASSO und Ridge Regression. Elastic Net liefert zwar auch dünnbesetzte Lösungen, lässt hingegen mehr Merkmale zum Modell zu. Da der Fokus dieser Arbeit auf LASSO und SLOPE liegt, wird Elastic Net nicht weiterhin behandelt.

Der direkte Vorgänger, aus dem die Idee für LASSO entstanden ist, ist Nonnegative Garrote von Breimann [8]

$$\min_{c\in\mathbb{R}^p} \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} c_j\beta_{j,OLS}x_{ji})^2 \text{ , s.t. } c_j \geq 0 \ \forall j = 1, ..., p, \ \sum_{j=1}^{p} c_j \leq R, \ \ R > 0 \ . \quad (2.12)$$

Die Beschränkung der Summe der Koeffizienten $c_j, j = 1, ..., , p$, durch ein $R > 0$ führt zur Skalierung der Koeffizienten $\beta_{j,OLS}$, sodass die skalierten Regressionskoeffizienten von $\tilde{\beta} = (\tilde{\beta}_1, ..., \tilde{\beta}_p)^T = (c_1\beta_{1,OLS}, ..., c_p\beta_{p,OLS})^T$ für hinreichend kleine Werte für R betragsmäßig (viel) kleiner sind als die von β_{OLS}. Denn das Optimierungsproblem (2.12) ist unter der Berücksichtigung der Nebenbedingungen für c äquivalent zu

$$\min_{\tilde{\beta}\in\mathbb{R}^p} \ \|y - X\tilde{\beta}\|_2^2 \quad \text{s.t.} \quad |\sum_{j=1}^{p} \tilde{\beta}_j| \leq s \ , \quad s > 0 \ .$$

Der Parameter s ist von β_{OLS} und R abhängig. Je nach Höhe von R bzw. s erhalten einzelne Einträge von c und damit von $\tilde{\beta}$ den Wert Null, wodurch Variablenselektion ermöglicht wird. Der Nachteil von Nonnegative Garrote liegt jedoch vor allem in der Notwendigkeit der Bestimmung von β_{OLS} und damit der davon unmittelbaren Abhängigkeit der resultierenden Lösung.

2.2 Existenz, Eindeutigkeit und Eigenschaften der Lösung

In diesem Kapitel sollen Lösungen von LASSO charakterisiert und eine explizite Form hergeleitet werden. Es werden Bedingungen aufgezeigt, unter deren Gültigkeit eine eindeutige Lösung vorliegt bzw. von einer eindeutigen Lösung auszugehen ist. Dazu erfolgt zunächst die Charakterisierung von OLS-Lösungen, um unter anderem zu verdeutlichen, weswegen OLS nicht immer das Verfahren erster Wahl sein sollte.

OLS Die folgenden Aussagen gehen auf [17, 23] zurück. Der Regularisierungsterm werde zunächst vernachlässigt, sodass die kleinsten Quadrate (1.3) zu minimieren sind. f_{OLS} ist als quadratische Funktion streng konvex, und zwar bezüglich $\tilde{y} = X\beta$. Daraus folgt, dass eine Lösung $y_{OLS} \in \underset{\tilde{y}\in\mathbb{R}^n}{\operatorname{argmin}} \|y - \tilde{y}\|_2^2$ existiert und y_{OLS} eindeutig ist. Dies wiederum garantiert die Existenz mindestens einer Lösung $\beta_{OLS} \in \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}}\, f_{OLS}(\beta)$.

Vorab sei angemerkt, dass für eine Matrix $X \in \mathbb{R}^{n\times p}$

$$\begin{gathered}\operatorname{rang}(X) = \operatorname{rang}(X^T) = \operatorname{rang}(X^TX) = \operatorname{rang}(X^TX) \leq \min\{n, p\}\ , \\ \{\operatorname{Kern}(X) = \{0\} \iff rang(X) = p\} \text{ bzw. } \{\operatorname{Kern}(X) \neq \{0\} \iff \operatorname{rang}(X) < p\}\end{gathered} \tag{2.13}$$

gilt, wobei $\operatorname{Kern}(X) := \{z \in \mathbb{R}^p |\ Xz = 0\}$.

Die Funktion f_{OLS} ist differenzierbar. Bilden des Gradienten von f_{OLS} nach β und Setzen auf Null als notwendige Bedingung für Extrema ergibt

$$\nabla f_{OLS}(\beta_{OLS}) = 0 \iff X^TX\beta_{OLS} = X^Ty \tag{2.14}$$

$$\iff \beta_{OLS} = (X^TX)^+X^Ty \ \pm\ z,\ z \in \operatorname{Kern}(X) = \{z \in \mathbb{R}^p |\ Xz = 0\}\ , \tag{2.15}$$

wobei β_{OLS} Lösung der Normalengleichung (2.14) mit der Systemmatrix $X^TX \in \mathbb{R}^{p\times p}$ und dem Output $X^Ty \in \mathbb{R}^p$ ist. Die Gleichung (2.15) berücksichtigt die Äquivalenzen (2.13). Die Matrix $(X^TX)^+ \in \mathbb{R}^{p\times n}$ ist die sogenannte *Pseudoinverse* von X^TX. Für die genaue Definition der Pseudoinverse siehe Anhang A2.

Seien β_{OLS}^1 und β_{OLS}^2 zwei beliebige Lösungen der Normalengleichung. Aufgrund der Konvexität ist $\alpha\beta_{OLS}^1 + (1-\alpha)\beta_{OLS}^2$, $\alpha \in [0, 1]$, gleichfalls ein Optimum. Demnach hat die Normalengleichung bei $\operatorname{rang}(X) < p$ unendlich viele Lösungen.

β_{OLS} ist genau dann eindeutig, wenn die Matrix $X^TX \in \mathbb{R}^{p\times p}$ invertierbar ist. Die Matrix X^TX ist folglich symmetrisch positiv definit, sodass wegen $\nabla^2 f_{OLS}(\beta) = H_f(\beta) = 2X^TX$ die Funktion f_{OLS} streng konvex bezüglich β ist[3]. Die eindeutige Lösung lautet hierbei

$$\beta_{OLS} = (X^TX)^{-1}X^Ty\ . \tag{2.16}$$

Die Invertierbarkeit von X^TX ist äquivalent dazu, dass $\operatorname{rang}(X^TX) = p$ gilt. Wegen der Äquivalenzen in (2.13) impliziert dies, dass X vollen Rang hat, $\operatorname{rang}(X) = p$, und kann folglich nur bei $n \geq p$ auftreten. Es werden zwei Fälle unterschieden: $n > p$ und $n \leq p$.

Bei $n > p$ ist $y = X\beta_{OLS}$, sodass die Daten perfekt angepasst werden, selten er-

[3] Eine Funktion ist genau dann streng konvex, wenn die Hesse-Matrix positiv definit ist.

füllt. Sollte dies gelten, so liegen alle Beobachtungspaare auf einer Hyperebene der Dimension $m \leq p-1$ (siehe Beispiel A1 im Anhang). Bei $n > p$ und $\text{rang}(X) < p$ hat die Lösung dieselben Eigenschaften wie für den Fall $n \leq p$. Ist $n \leq p$, so gilt $\text{rang}(X) = m \leq n$, alle Beobachtungspaare liegen demzufolge auf einer (m-1)-dimensionalen Hyperebene, sodass $y = X\beta_{OLS}$ gilt. Das lineare Gleichungssystem $y = X\beta_{OLS}$ ist bei $n < p$ ein System, das weniger Gleichungen als Unbekannte hat, sodass unendlich viele Lösungen existieren. Ist eine Lösung β_{OLS} der Normalengleichung (2.14) gefunden, so ist

$$\beta_{OLS} \pm z \quad \forall z \in \text{Kern}(X)$$

ebenfalls ein Optimum, vgl. (2.15) [21, 37]. Dies hat zur Folge, dass für zwei beliebige Lösungen β^1_{OLS} und β^2_{OLS} mindestens ein $j \in \{1, ..., p\}$ existiert, sodass sich der j-te Koeffizient im Vorzeichen unterscheidet, $\text{sign}(\beta^1_{j,OLS}) = -1$ und $\text{sign}(\beta^2_{j,OLS}) = 1$. Das führt dazu, dass im Fall einer gewünschten Vorhersage des Outputs für eine neue Beobachtung der j-te Koeffizient bei $\beta^1_{j,OLS}$ mit einer negativen und bei $\beta^2_{j,OLS}$ einer positiven Zahl multipliziert wird. Dies hat starke Schwankungen der Vorhersage zur Folge. Jegliche Möglichkeit der Interpretation einer Lösung kann dadurch verloren gehen.

Insgesamt weist die Methode der kleinsten Quadrate eine eher schwache Performance auf, sodass Bedarf nach alternativen Methoden wie bspw. LASSO besteht.

LASSO $f_{LASSO}(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$ ist als Zusammensetzung der streng konvexen Funktion f_{OLS} und der konvexen L1-Norm streng konvex, und zwar wie f_{OLS} bezüglich $\tilde{y} = X\beta$. Demnach existiert eine Lösung $y_L \in \underset{\tilde{y} \in \mathbb{R}^n}{\text{argmin}} \frac{1}{2n}\|y - \tilde{y}\|_2^2 + \lambda\|\beta\|$ und y_L ist eindeutig. Folglich existiert mindestens eine Lösung β_L von LASSO (2.6). Bei $\text{rang}(X) = p$ lässt sich wie bei OLS sofort auf die Eindeutigkeit von β_L schließen. Für $\text{rang}(X) < p$, was bei $n < p$ immer auftritt, kann es unendlich viele Lösungen geben. Es werden Fälle aufgezeigt, in denen trotz $\text{rang}(X) < p$ von einer eindeutigen Lösung auszugehen ist.

Unter der Annahme, dass mehr als eine Lösung existiert, ist die Konvexkombination $\alpha\beta^1_L + (1-\alpha)\beta^2_L \; \forall \alpha \in [0, 1]$ zweier Lösungen β^1_L und β^2_L wegen der Konvexität von f_{LASSO} ebenfalls eine Lösung, deren L1-Norm übereinstimmt [21, 35, 36, 37]. Somit hat LASSO entweder eine oder unendlich viele Lösungen. Die Anzahl aller Möglichkeiten, ein Modell aus der Menge der Merkmale $\{1, ...p\}$ zu selektieren, was der Potenzmenge von $\{1, ...p\}$ entspricht, ist allerdings beschränkt und ist gegeben durch

$$\sum_{k=0}^{p} \binom{p}{k} = 2^p \tag{2.17}$$

Es gelte $\text{rang}(X) < p$ und es sei eine Lösung β_L verfügbar. Ist nicht bekannt, ob diese eindeutig ist, lässt sich diese ohne weitere Überlegungen nur schwer interpretieren und hat somit für die lineare Regression und die Fragestellung zunächst kaum Aussagekraft [36, 37]. Da LASSO vor allem im Fall $p > n$ von Interesse ist, müssen zusätzliche Bedingungen hergeleitet werden, um Lösungen und Eindeutigkeit besser zu charakterisieren. Die folgenden Ergebnisse gehen auf [36, 37, 39] zurück und beziehen sich auf ein festes $\lambda > 0$. Für detaillierte Herleitungen und Beweise sei auf die angegebene Literatur verwiesen.

Wegen der L1-Norm ist LASSO nicht differenzierbar, und zwar in jedem $j \in \{1, ..., p\}$, für das $\beta_j = 0$ gilt. $\beta_j = 0$ bedeutet graphisch, dass es sich um eine Ecke des Hyperoktaeders handelt. Dementsprechend hat die Lösung vorerst keine explizite Form. Aufgrund der Konvexität kann ein *Subdifferential*[4] gebildet werden, sodass die sogenannten *Karush-Kuhn-Tucker-Bedingungen* (KKT-Bedingungen) von LASSO für eine Lösung β_L (2.7) lauten [9, 36, 37, 21]:

$$\frac{1}{n} X^T (y - X\beta_L) = \lambda s, \quad s \in \nabla_\beta \|\beta_L\|_1 \ . \tag{2.18}$$

KKT-Bedingungen sind Optimalitätsbedingugen eines konvexen Problems mit Nebenbedingungen [3]. $s = (s_1, ..., s_p)^T$ ist das Subdifferential der L1-Norm ausgewertet in β_L und ist gegeben durch

$$s_j \in \begin{cases} \text{sign}(\beta_{j,L}), & \beta_{j,L} \neq 0 \\ [-1, 1], & \beta_{j,L} = 0 \end{cases} \qquad \forall j \in \{1, ..., p\} \ . \tag{2.19}$$

Ein β_L ist genau dann eine Lösung von LASSO, wenn (β_L,s) die Gleichung (2.18) löst [21, 36]. Sei $J := \text{supp}(\beta_L)$ als Support einer Lösung β_L definiert, sodass o.B.d.A. $\beta_L = (\beta_{J,L}, \beta_{-J,L}) = (\beta_{J,L}, 0_{|p-|J||})$, wobei $|J|$ die Kardinalität von J ist und $0_{|p-|J||}$ den $p - |J|$-dimensionalen Nullvektor bezeichnet. Aufgrund der Eindeutigkeit von $X\beta_L$ für jede beliebige Lösung β_L liefern die KKT-Bedingungen (2.18) mit der Definition des Subdifferentials s (2.19) dessen Eindeutigkeit für ein festes $\lambda > 0$. Daraus folgt, dass

$$\nexists j \in \text{supp}(\beta_L^1) \cap \text{supp}(\beta_L^2) : \ \text{sign}(\beta_{j,L}^1) \neq \text{sign}(\beta_{j,L}^2)$$

für zwei beliebige Lösungen β_L^1 und β_L^2 mit den entsprechenden Supports gilt [21, 36]. Das Vorzeichen des j-ten Koeffizienten, $j \in \{1, ..., p\}$, stimmt somit im Gegensatz zu Lösungen von OLS bei allen Lösungen von LASSO überein, wodurch die Vorhersage für neue Beobachtungen robuster ist.

[4] Das Subdifferential bezeichnet den verallgemeinerten Gradienten für konvexe Funktionen, die nicht differenzierbar sind.

Mit dem Support J von β_L lassen sich die KKT-Bedingungen (2.18) und das Subdifferential (2.19) als

$$\frac{1}{n}X_J^T(y - X\beta_L) \;=\; \lambda s_J \quad\Longleftrightarrow\quad \frac{1}{n}|X_J^T(y - X\beta_L)| \;=\; \lambda \;, \tag{2.20}$$

$$\frac{1}{n}|X_{-J}^T(y - X\beta_L)| \;\leq\; \lambda|s_{-J}| \;\leq \lambda \tag{2.21}$$

zusammenfassen, wobei $X_J^T := (X_J)^T$, $X_{-J} := (X)_{j\in\{1,...,p\}\setminus J}$, $X_{-J}^T := (X_{-J})^T$ und $s_{-J} := s_{j\in\{1,...,p\}\setminus J}$.

Die Ungleichung (2.21) besagt, dass es Indizes $j \notin J$ geben kann, für die $\frac{1}{n}|X_j^T(y - X\beta_L)| = \lambda$ gilt, die somit (2.20) ebenfalls erfüllen. Folglich sind die Indexmengen, die die Gleichheit in (2.20) und in (2.21) erfüllen, nicht zwingend disjunkt. Daher ist dieses System noch nicht ausreichend, um β_L in einer expliziten Form darzustellen.

Die Indexmenge, für die Gleichheit gilt, sei definiert als

$$E := \{j \in \{1, ..., p\} |\; \frac{1}{n}|X_j^T(y - X\beta_L)| = \lambda\} \;. \tag{2.22}$$

Diese wird in der Literatur als *Equicorrelation Set* bezeichnet, weil E diejenigen Variablen enthält, deren absolutes Skalarprodukt mit dem Residuum $y - X\beta_L$ maximal ist und diese somit die maximale Korrelation mit dem Residuum aufweisen. Aufgrund der Eindeutigkeit des Subdifferentials s ist der Support J einer beliebigen Lösung β_L in der so definierten Indexmenge E enthalten, $J \subseteq E\ \forall\beta_L$. Andernfalls gilt $|s_j| < 1$ für $j \notin E$ und entsprechend $\beta_{j,L} = 0$ [36, 37]. Dies hat zur Folge, dass für ein festes λ von vornherein festgelegt ist, welche Variablen immer aus dem Modell eliminiert werden und welche immer im Modell verbleiben, und zwar ohne Berücksichtigung deren Relevanz.

Aus den KKT-Bedingungen (2.18), der Definition von E (2.22) und der Eindeutigkeit von $X\beta_L$ geht das lineare Gleichungssystem für $\beta_{E,L}$

$$\frac{1}{n}X_E^T(y - X_E\beta_{E,L}) \;=\; \lambda s_E \tag{2.23}$$

$$\Longleftrightarrow\; X_E\beta_{E,L} = X_E(X_E)^+(y - (X_E^T)^+ n\lambda s_E) = y - (X_E^T)^+ n\lambda s_E$$

$$\Longleftrightarrow \beta_{E,L} = (X_E)^+(y - (X_E^T)^+ n\lambda s_E) \tag{2.24}$$

hervor, wobei $(X_E)^+ \in \mathbb{R}^{p\times n}$ bzw. $(X_E^T)^+ \in \mathbb{R}^{n\times p}$ die Pseudoinverse von X_E bzw. X_E^T ist. Mit $(X_E^T X_E)^+$ als die Pseudoinverse von $X_E^T X_E$ lautet nun die äquivalente

Form der KKT-Bedingungen mit $z \in \text{Kern}(X_E)$

$$\begin{aligned}\beta_{E,L} &\overset{(2.23)}{=} (X_E^T X_E)^+ (X_E^T y - n\lambda s_E) \pm z \\ &= (X_E^T X_E)^+ X_E^T y - (X_E^T X_E)^+ n\lambda s_E \pm z = \beta_{E,OLS} - (X_E^T X_E)^+ n\lambda s_E \pm z \ , \end{aligned} \tag{2.25}$$

$$\beta_{-E,L} = 0 \ . \tag{2.26}$$

Somit reduziert sich die Bestimmung von β_L (als Zusammensetzung von $\beta_{E,L}$ und $\beta_{-E,L}$) auf die von $\beta_{E,L}$, das explizit durch das lineare Gleichungssystem (2.25) bzw. (2.24) gegeben ist. Die Gleichungen (2.25) und (2.24) sind laut Herleitung äquivalent, dementsprechend stimmen deren Lösungen überein. Diejenigen $z \in \text{Kern}(X_E)$, für die $\text{sign}(\beta_{E,L}) = s_E$ gilt, liefern eine zulässige Lösung $\beta_{E,L}$ definiert als die Gleichung (2.25). Zusammen mit der Gleichung (2.26) sind die KKT-Bedingungen (2.18) erfüllt und $\beta_{E,L}$ ist folglich eine zulässige Lösung für LASSO (2.6). Für $z = 0$ ist die Bedingung der Übereinstimmung der Vorzeichen immer erfüllt. Eine interessante Beobachtung der Form von $\beta_{E,L}$ ist (unter Vernachlässigung von z) dessen Zusammensetzung aus dessen Lösung der Methode der kleinsten Quadrate $\beta_{E,OLS} = (X_E^T X_E)^+ X_E^T y$ und dem Regularisierungsterm $(X_E^T X_E)^+ n\lambda s_E$, sodass die Regularisierung dadurch deutlich veranschaulicht wird und ausführlich untersucht werden kann.

Das System (2.24) bzw. (2.25) gibt Aufschluss über den Support von $\beta_{E,L}$. Der j-te Eintrag von $\beta_{E,L}$, $j \in E$, ist genau dann Null, wenn die j-te Zeile von $(X_E^T X_E)^+$ bzw. $(X_E)^+$, definiert als $(X_E^T X_E)^+_{[j]}$ bzw. $(X_E)^+_{[j]}$, Null ist oder wenn y in der Nullmenge

$$N = \bigcup_{E,s} \bigcup_{j \in E} \{y \in \mathbb{R}^n : ((X_E)^+)_{[j]}(y - (X_E^T)^+ n\lambda s) = 0\}$$

als Vereinigung von endlich vielen affinen $(n-1)$-dimensionalen Unterräumen liegt. Ist die Spalte $(X_E)^+_{[j]}$ Null, so impliziert dies, dass die j-te Spalte von X_E Null ist, woraus jedoch wegen der Definition des Equicorrelation Set (2.22) $\lambda = 0$ folgt. Dass die Spalte $(X_E)^+_{[j]}$ Null ist, ist daher ausgeschlossen. Für den Support gilt demnach $|\text{supp}(\beta_{E,L})| = |E|$ fast überall.

Aus der Gleichung (2.25) ist ersichtlich, dass $\beta_{E,L}$ eindeutig ist, wenn die Matrix X_E vollen Rang hat, $\text{rang}(X_E) = |E|$, was äquivalent zu $\text{Kern}(X_E) = \{0\}$ ist[5]. Die Matrix $X_E^T X_E$ ist hierbei invertierbar, sodass $(X_E^T X_E)^+ = (X_E^T X_E)^{-1}$, und das

[5] In [36] wurde gezeigt, dass ebenso die umgekehrte Richtung gilt: Aus der Eindeutigkeit von β_L folgt $\text{Kern}(X_E) = \{0\}$.

eindeutige β_L erfüllt

$$\begin{aligned} \beta_{E,L} &= (X_E^T X_E)^{-1}(X_E^T y - n\lambda s_E) = (X_E^T X_E)^{-1} X_E^T y - (X_E^T X_E)^{-1} n\lambda s_E \ , \\ \beta_{-E,L} &= 0 \ . \end{aligned} \tag{2.27}$$

Zudem lässt sich zeigen, dass für den Support J der eindeutigen Lösung β_L gegeben durch (2.27) die Beziehung

$$|\text{supp}(\beta_{E,L})| = |J| \leq \min\{n, p\} \tag{2.28}$$

gilt [12, 36]. Dies hat zur Folge, dass sich der Rechenaufwand vor allem bei $n \ll p$ erheblich reduziert. Außerdem ist dadurch die Anzahl der Möglichkeiten der Wahl eines Modells nun von der Größenordnung $O(\min\{n, p\})$ im Gegensatz zur zuvor hergeleiteten von 2^p (2.17) [9].

Obwohl Eindeutigkeit und Dünnbesetztheit der Lösung von LASSO erwünscht ist, kann diese in praktischen Problemstellungen mit $n \ll p$ wegen (2.28) zu einem unzureichenden Ergebnis führen, denn die Anzahl der durch das eindeutige β_L ausgewählten Merkmale kann hierbei stark die der tatsächlich relevanten unterschreiten. Denn bei stark korrelierten Variablen neigt LASSO dazu, diese Korrelation zu ignorieren und nur eine oder sogar keine der korrelierten Variablen zu selektieren [21, 15]. Ansätze, um dies zu vermeiden, sind bspw. Elastic Net, das in Kapitel 2.1 erwähnt wurde, und Grouped LASSO [43].

Ist die Lösung $\beta_{E,L}$ (2.25) nicht eindeutig, so lässt sich die Kardinalität deren Supports und damit die Anzahl der extrahierten Merkmale nicht beschränken, sodass im Extremfall $|E| = p$ für eine Lösung β_L resultiert und diese somit vollbesetzt ist [36]. Dabei kann dies unabhängig vom Verhältnis von n und p auftreten. Bei $n \ll p$ lässt sich starke Korrelation von Variablen nicht vermeiden und es ist aufgrund der Invarianz von LASSO gegenüber stark korrelierten Variablen dennoch eine dünnbesetzte Lösung zu erwarten. Tibshirani (2013) [36] hat andererseits gezeigt, dass unter der unendlichen Anzahl an Lösungen mindestens eine existiert, deren Support maximal $\min\{n, p\}$ Elemente enthält. Zudem spannen die aus einer beliebigen Lösung $\beta_{J,L}$ resultierenden Spalten X_J denselben affinen Unterraum für fast alle $y \in \mathbb{R}^n$ auf.

Das System (2.25) bzw. (2.27) ist erst dann lösbar, wenn die Indexmenge E und das eindeutige $s = \text{sign}(\beta_L)$ vorliegen. Diese sind in der Regel a priori nicht bekannt und lassen sich erst durch eine bereits vorhandene Lösung β_L bestimmen. Außerdem hängt $\beta_{E,L}$ von der Matrix $X_E^T X_E$, was vor allem wegen schlechter Konditionierung[6]. Probleme bereiten kann. Dennoch ist das System nützlich, zum einen, weil es eine

[6] Für eine invertierbare Matrix $X \in \mathbb{R}^{p \times p}$ ist die Konditionszahl von $X^T X$ das Quadrat der Konditionszahl von X, $\kappa(X^T X) = (\kappa(X))^2$.

ziemlich detaillierte Charakterisierung der Lösung erlaubt, zum anderen bietet dieses einen Ansatz für Lösungsalgorithmen, wie bspw. das LARS, siehe Kapitel 2.3.1.

Nun besteht die Frage, wann X_E vollen Rang hat, sodass $\beta_{E,L}$ und somit die Lösung β_L gegeben als (2.27) eindeutig ist. Im Fall $n \geq p$ mit $\text{rang}(X) = p$ folgt sofort $\text{rang}(X_E) = |E|$. Eine allgemeine und recht oft vorhandene Eigenschaft der Matrix X impliziert vollen Rang von X_E und kann bei einer beliebigen Größenordnung von n und p und deren Verhältnis auftreten. Diese ist das Aufweisen der sogenannten *general position* der Spalten. Die Spalten einer Matrix $X = (X_1, ..., X_p)$ sind in general position, wenn sich höchstens $k+1$ Elemente aus $\{\pm X_1, ..., \pm X_p\}$, bis auf deren Antipoden[7] , in jedem k-dimensionalen, $k \leq \min\{n, p\}$, affinen Unterraum $U \subset \mathbb{R}^n$ befinden. Dies ist äquivalent dazu, dass keine Elemente aus $\{\pm X_j | j \neq j_1, ..., j_{k+1}\}$ in dem durch $\sigma_1 X_{j_1}, ..., \sigma_{j_{k+1}} X_{j_{k+1}}$, wobei $\sigma_1, ..., \sigma_{j_{k+1}} \in \{-1, +1\}$, aufgespannten affinen Unterraum enthalten sind. Unter der Bedingung, dass die Matrixeinträge als Zufallsvariablen aus einer stetigen Wahrscheinlichkeitsverteilung gezogen wurden, kann für beliebige $y \in \mathbb{R}^n$ und $\lambda > 0$ mit einer Wahrscheinlichkeit von Eins gewährleistet werden, dass sich die Spalten in general position befinden und folglich LASSO eine eindeutige Lösung besitzt [36, 21].

2.3 Lösungsalgorithmen

Sei $n \geq p$. Betrachte zunächst den speziellen Fall, in dem die Spalten der Matrix X orthogonal[8] sind. Die Spalten seien zudem orthonormal, sodass deren Norm Eins beträgt und folglich $X^T X = I_p$ gilt. Die Spalten sind insbesondere linear unabhängig, was $\text{rang}(X) = p$ bedeutet und demnach $n \geq p$ voraussetzt. Die Lösung β_L ist hierbei somit eindeutig.

Wegen $X^T(y - X\beta) = X^T y - X^T X\beta = \tilde{y} - \beta$, $\tilde{y} \in \mathbb{R}^p$, reduziert sich LASSO auf

$$\min_{\beta \in \mathbb{R}^p} \left\{ f_{LASSO} = \frac{1}{2n} \|\tilde{y} - \beta\|_2^2 + \lambda \|\beta\|_1 \right\} .$$

Bilden des Subdifferentials und Setzen auf Null als notwendige Bedingung für die Lösung β_L ergibt

$$\nabla f_{LASSO}(\beta_L) = -\frac{1}{n}(\tilde{y} - \beta_L) + \lambda s = 0 \iff \beta_L = \tilde{y} - n\lambda s,$$

wobei $s = (s_1, ..., s_p)$ mit $s_j = \text{sign}(\beta_{j,L})$ für $\beta_{j,L} \neq 0$ und $s_j \in [-1, 1]$ sonst, $j \in \{1, ..., p\}$. Damit ist das β_L gegeben durch

[7] In diesem Zusammenhang sind Antipoden Punkte, die das gegenteilige Vorzeichen haben.

[8] Orthogonalität der Spalten ist nicht mit der Orthogonalität der Matrix gleichzusetzen. Die Matrix X ist genau dann orthogonal, wenn $n = p$ gilt und sowohl Spalten als auch Zeilen orthonormal sind, sodass $X^T X = X X^T = I_p$ folgt.

$$\beta_{j,L} \;=\; \begin{cases} \tilde{y}_j - n\lambda\,, & \tilde{y}_j \;>\; n\lambda \\ 0\,, & \tilde{y}_j \;\leq\; n\lambda \\ \tilde{y}_i + n\lambda\,, & \tilde{y}_j \;<\; -n\lambda\,. \end{cases}$$

Durch Einführung des sogenannten *Soft-Thresholding Operator*'s $S(x,\lambda)$ (auch $S_\lambda(x)$) mit

$$S(x,\lambda) = \text{sign}(x)(|x|-\lambda)_+ = \begin{cases} x-\lambda\,, & x \;>\; 0 \text{ und } \lambda \;<\; |x| \\ 0\,, & |x| \;\leq\; \lambda \\ x+\lambda\,, & x \;<\; 0 \text{ und } \lambda \;<\; |x| \end{cases} \tag{2.29}$$

lässt sich $\beta_{j,L}$ umschreiben als

$$\beta_{j,L} = S\left(\tilde{y}_j\,, n\lambda\right) = S(\beta_{j,OLS}\,,\; n\lambda)\,. \tag{2.30}$$

Es sei angemerkt, dass der Vorfaktor n in der Literatur vernachlässigt wird. Die

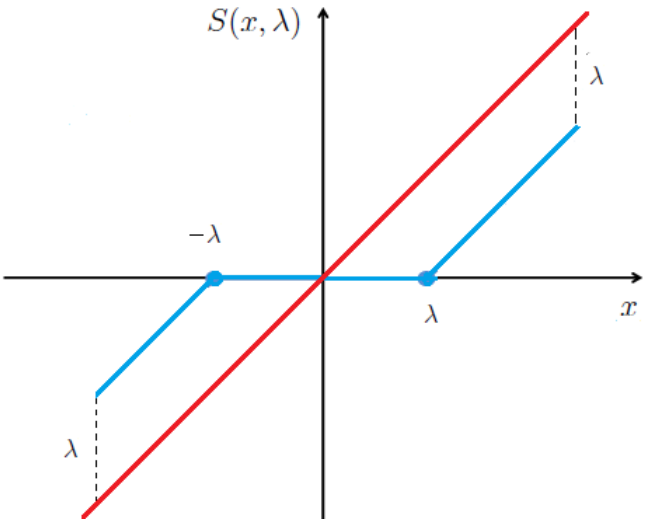


Abbildung 3 Soft-Thresholding Operator. Die rote Linie stellt die Funktion $f(x) = x$ dar. Die blaue Linie ist der Graph zu $S(x,\lambda)$. Liegt x im Intervall $[-\lambda,\lambda]$, so wird $S(x,\lambda)$ auf Null gesetzt, bei $x < \lambda$ wird x um $-\lambda$ nach links, sonst um $+\lambda$ nach rechts verschoben.

zweite Gleichheit in der Gleichung (2.30) folgt aus der Tatsache, dass im Falle orthonormaler Spalten sich die Lösung der kleinsten Quadrate β_{OLS} aufgrund $X^T(y - X\beta) = X^T y - X^T X\beta = \tilde{y} - \beta$ sofort als $\beta_{OLS} = \tilde{y}$ ergibt. Falls die Spalten der Matrix X nicht normiert sind, sodass $X^T X = \text{diag}(d_1, ..., d_p)$, so gilt für die Lösung $\beta_{j,L} = S\left(\tilde{y}_i\,, \frac{n\lambda}{d_j}\right)\;\forall j \in \{1, ..., p\}$.

Sei X nun beliebig. Für LASSO als ein konvexes Optimierungsproblem bieten sich einige attraktive Verfahren zu dessen Lösung an. Dieses Kapitel behandelt die am

häufigsten eingesetzten Lösungsalgorithmen.

2.3.1 LARS

Der Lösungsalgorithmus *Least Angle Regression* (LARS) [12] ist ebenso wie LASSO und SLOPE eine Methode zur Variablenselektion. LARS zieht die klassische lineare Regression mit der Optimalitätsbedingung für die Lösung von OLS (2.15) heran.

LARS lässt sich für die Lösung von LASSO modifizieren. Im Rahmen dieser Arbeit wird nur die modifizierte Version in ihren Grundzügen vorgestellt.

Bisherige Ergebnisse beziehen sich auf ein festes λ. Wie zu Anfang erklärt, regelt der RP λ, wie gut die Daten durch einen Regressionsvektor $\hat{\beta}$ angepasst und welche Variablen selektiert werden. Insbesondere gilt $\hat{\beta} = \hat{\beta}(\lambda)$. Die Wahl von λ ist demgemäß nicht trivial. Aus diesem Grund werden in den meisten Anwendungen Lösungen für eine Reihe von RP gefordert. Dadurch ist es dem Anwender möglich, sich mit einer Sequenz von Lösungen auseinanderzusetzen, und je nach Kontext eine dieser Lösungen zu dem entsprechenden Parameter zu verwenden. Durch CV lässt sich bspw. der Parameter mit dem kleinsten CV-Fehler bestimmen. LARS berechnet Lösungen für LASSO für eine Sequenz von RP $\lambda_0 > \lambda_1 > ... > \lambda_m \geq 0$. Es wird für jedes λ_l das LASSO $\min_{\beta \in \mathbb{R}^p} \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda_l \|\beta\|_1$ gelöst. Der Regressionsvektor β und dessen Equicorrelation Set E werden somit als Funktionen von λ aufgefasst, $\beta = \beta(\lambda)$ und $E = E(\lambda)$, wobei $\lambda \in [0, \infty)$. Damit gilt insbesondere $\beta_j = \beta_j(\lambda)\ \forall j \in \{1, ..., p\}$. Die Funktion $\beta(\lambda)$ ist stetig und stückweise linear und lässt sich als Graph darstellen. Die resultierende Sequenz von Lösungen verdeutlicht die Gegenüberstellung des Fehlers des Residuums und der L1-Norm des Regularisierungsterms. Die Knoten des resultierenden Graphen entsprechen den einzelnen Lösungen für LASSO zu den RP $\lambda_l, l \in \{1, ..., m\}$, wobei jedes λ_l so bestimmt wird, dass dieses den Knoten darstellt, an dem sich der Support $\text{supp}(\beta(\lambda))$ und folglich E ändern, vgl. Abbildung 4.
Aufgrund der Stetigkeit und der stückweisen Linearität lassen sich Lösungen für LASSO für jedes beliebige $\lambda \in [0, \infty)$ durch lineare Interpolation berechnen.

Die Idee des Algorithmus besteht darin, in jeder Iteration l die Lösung $\beta(\lambda_l)$ zu λ_l so zu bestimmen, dass die zugehörigen KKT-Bedingungen für $\beta(\lambda_l)$ gegeben durch $\beta_E(\lambda_l) = (X_E)^+(y - (X_E^T)^+ n\lambda_l s_E)$ (vgl. die Gleichung (2.24)) und $\beta_{-E}(\lambda_l) = 0_{|p-|E||}$ erfüllt sind [36, 37].

Anhand der Form von $\beta_E(\lambda)$ wird durch Ausmultiplizieren dessen lineare Abhängigkeit von λ deutlich:

$$\beta_E(\lambda) = (X_E)^+(y - \lambda(X_E^T)^+ n s_E) = (X_E)^+ y - \lambda(X_E X_E^T)^+ n s_E =: c - \lambda d\ .$$

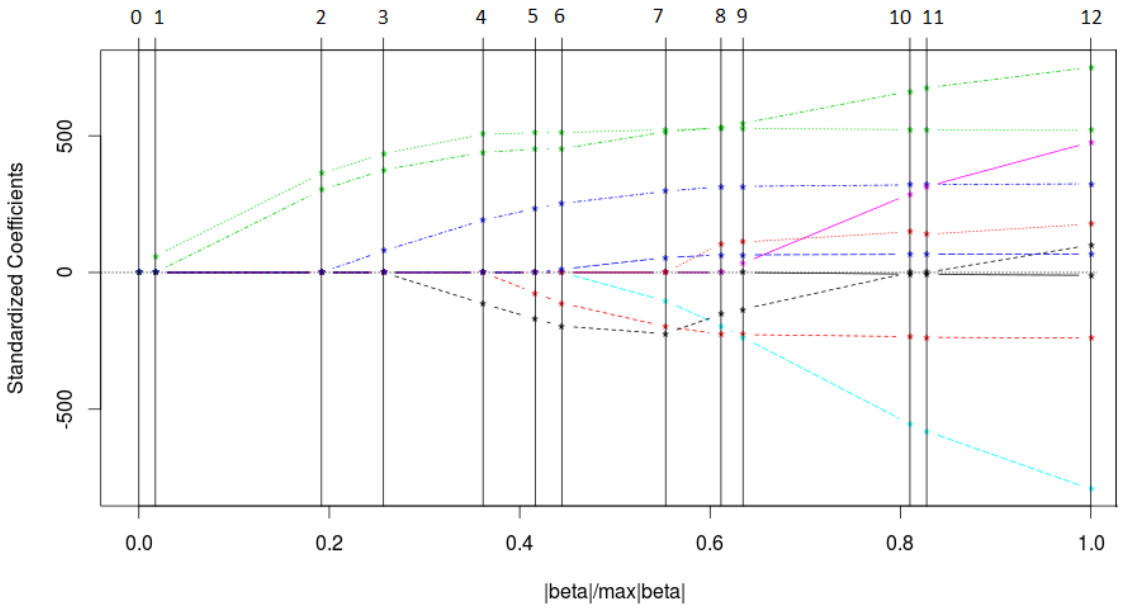


Abbildung 4 Veranschaulichung eines mit LARS berechneten Lösungspfades. Diese Abbildung wurde anhand des von dem R-Paket **lars** vorgeschlagenen Beispiels erstellt und leicht modifiziert. Der Datensatz umfasst insgesamt $p = 12$ Variablen. Die vertikalen Linien entsprechen den Lösungen für einzelne λ_l, $l \in \{0, ..., 12\}$, wobei $\beta(\lambda_0) = 0_{|p|}$. Oberhalb dieser Linien ist die Anzahl der für λ_l im Modell enthaltenen Variablen abgebildet. Fällt der RP, so kommen immer mehr Variablen hinzu, sodass für $\lambda_{12} = 0$ alle p Variablen zum Modell gehören. Auf der horizontalen Achse ist das Verhältnis der L1-Norm des aktuellen $\beta(\lambda_l)$ zu der maximalen L1-Norm von $\beta(\lambda_{12})$ abgebildet, wobei die L1-Norm bei $\beta(\lambda_{12}) = \beta_{OLS}$ mit $\lambda = 0$ ihr Maximum erreicht. Auf der vertikalen Achse sind die standardisierten Werte der einzelnen $\beta_j(\lambda_l)$, $j \in \{1, ..., 12\}$, abgebildet. Die farbigen stückweise linearen Linien entsprechen jeweils dem Lösungspfad eines β_j.

Für $\lambda = \lambda_l$ ist das $\beta_E(\lambda_l)$ folglich gegeben durch

$$\beta_E(\lambda_l) = (X_E)^+ y - \lambda_l (X_E X_E^T)^+ n s_E =: c_l - \lambda_l d_l \ .$$

$\beta_E(\lambda_l)$ kann hierbei als die folgende OLS-Lösung mit der Systemmatrix X_E aufgefasst werden:

$$\beta_E(\lambda_l) = \operatorname{argmin} \left\{ \|\hat{\beta}_E\|_2 : \ \hat{\beta}_E \in \operatorname*{argmin}_{\beta_E \in \mathbb{R}^{|E|}} \|y - (X_E^T)^+ \lambda_l s_E - X_E \beta_E\|_2^2 \right\} \ . \qquad (2.31)$$

Der Algorithmus 1 gibt einen Überblick über die Vorgehensweise von LARS, für Details, ausführliche Herleitungen und Beweise sei auf [12, 36, 37] verwiesen. In [12, 37] wird angenommen, dass X_E vollen Rang hat, sodass die Lösung eindeutig ist, wohingegen [36] vom allgemeinen Fall ausgeht.

Die Gleichung für $\beta_E(\lambda)$ (vgl. Schritt 9) lässt sich durch einfaches Nachrechnen zeigen. Die Koeffizienten von $\beta_E(\lambda_l)$ werden somit in Richtung $(\lambda_l - \lambda)d_l = (\lambda_l -$

Algorithmus 1 LARS Algorithmus für LASSO

1: X, y.
2: $k = 0$, $\lambda_0 = \infty$, $\beta(\lambda_0) = 0_{|p|}$ Startlösung, $E = \emptyset$, $s_E = \emptyset$.
3: **while** $\lambda_l > 0$ **do**
4: Berechne $\beta_E(\lambda_l)$ mit (2.31).
5: $\beta_{-E}(\lambda_l) = 0_{|p-|E||}$.
6: $\beta(\lambda_l) = (\beta_E(\lambda_l), \beta_{-E}(\lambda_l))^T$.
7: $\lambda < \lambda_l$.
8: $\beta_{-E}(\lambda) = 0_{|p-|E||}$.
9: $\beta_E(\lambda) = c_l - \lambda d_l = \beta_E(\lambda_l) + (\lambda_l - \lambda)d_l$.
10: Ermittle die nächste *joining time* λ_{l+1}^{join}.

$$\lambda_{l+1}^{join} = \max_{j \notin E, s_j \in \{-1,+1\}} \frac{X_j^T(y - X_E c_l)}{s_j - X_j^T X_E d_l} = \max_{j \notin E, s_j \in \{-1,+1\}} \frac{X_j^T(I - X_E(X_E)^+)y}{s_j - X_j^T(X_E^T)^+ s_E} . \tag{2.32}$$

11: Extrahiere Indizes j_{l+1}^{join}, für die (2.32) erfüllt ist.
12: Ermittle die nächste *crossing time* λ_{l+1}^{cross}

$$\lambda_{l+1}^{cross} = \max_{j \in E} \frac{[(X_E)^+ y]_j}{[(X_E^T X_E)^+ s]_j} \cdot \mathbb{1}\left\{ \frac{[(X_E)^+ y]_j}{[(X_E^T X_E)^+ s]_j} \leq \lambda_l \right\} . \tag{2.33}$$

13: Extrahiere Indizes j_{l+1}^{cross}, für die (2.33) erfüllt ist.
14: Senke λ bis $\lambda_{l+1} := max\{\lambda_{l+1}^{join}, \lambda_{l+1}^{cross}\}$.
15: **if** $\lambda_{l+1}^{cross} > \lambda_{l+1}^{join}$ **then**
16: $E = E \cup j_{l+1}^{join}$.
17: $s_{l+1}^{join} = \text{sign}(X_{j_{l+1}^{join}}(y - X\beta(\lambda_l))$.
18: $s_E = s_E \cup s_{l+1}^{join}$.
19: **else**
20: $s_E = s_E \setminus s_{j_{l+1}}$.
21: $E = E \setminus j_{l+1}^{cross}$.
22: **end if**
23: $l := l + 1$.
24: **end while**
25: $\hat{\beta} := \beta^{(l)}$ ist die resultierende Lösung.

$\lambda)(X_E X_E^T)^+ n s_E$ verschoben. Der resultierende Output ergibt sich entsprechend als

$$\hat{y} = X\beta_E(\lambda_l) + (\lambda_l - \lambda)X_E(X_E X_E^T)^+ n s_E .$$

Die Richtung $X_E(X_E X_E^T)^+ n s_E$ nennen Efron et al. (2004) [12] als *least angle direc-*

tion, woraus sich der Name für den Algorithmus ableitet.

Die Gleichung für λ_{l+1}^{join} geht aus dem Lösen von $X_j^T(y - X_E(c_l - \lambda d_l)) = s_j\lambda$ für λ für alle $j \notin E$ in der Iteration l hervor.

Wichtig zu bemerken ist die Tatsache, dass mit der Senkung von λ_l bis zu λ_{l+1} sowohl eine Hinzunahme als auch Entfernung einer Variablen aus E möglich ist. Die Anzahl der in E enthaltenen Variablen wächst infolgedessen nicht zwingend mit der fallenden Höhe von λ. Dies ist der wichtigste Unterschied zum klassischen LARS. Der Algorithmus zeigt außerdem auf, dass für alle $\lambda \in (\lambda_{l+1}, \lambda_l]$ die Anzahl der relevanten Variablen identisch ist und sich erst in λ_{l+1} ändert. Der Algorithmus liefert Lösungen für jedes $\lambda \in [0, \lambda_0]$ und diese sind tatsächlich mögliche Lösungen von LASSO, sodass $\beta_{E,L}(\lambda) = (X_E)^+(y - (X_E^T)^+\lambda ns)$ [36].

Der gesamte Aufwand von LARS für LASSO beläuft sich auf $O(np \cdot \min(n, p))$ Rechenoperationen [9]. Ist $n \ll p$, so folgt $O(np \cdot \min(n, p)) = O(p)$, sodass in diesem Fall ein in p linearer Aufwand vorliegt.

Insbesondere ist die Kardinalität des Supports der mit LARS berechneten Lösung von Interesse. Definiere die Menge N als

$$N = \bigcup_{E,s} \bigcup_{j \in E} \left\{ z \in \mathbb{R}^n : ((X_E)^+)_{[j]}(z - (X_E^T)^+\lambda ns) = 0 \right\} .$$

N ist als eine endliche Vereinigung von affinen Unterräumen der Dimension $n - 1$ eine Nullmenge. Liegt $y \in \mathbb{R}^n$ für ein $\lambda > 0$ nicht in N, so weist der Support der mit LARS für das λ berechneten Lösung die maximale Kardinalität unter allen Lösungen für LASSO auf [36]. Bei einer eindeutigen Lösung gilt somit $\text{supp}(\beta_L(\lambda)) = E$ fast überall.

2.3.2 Coordinate Descent

LASSO (2.6) besteht aus einer streng konvexen differenzierbaren Zielfunktion und einem konvexen, nicht differenzierbaren Regularisierungsterm. Zur Lösung solcher Probleme bietet sich das *Coordinate Descent* (CD) an [41]. Die folgenden Ergebnisse stammen aus [41, 15, 21].

Gehe zunächst nur von einer Einflussgröße bzw. Variable j aus, und zwar $X_j = (x_{ij})_i = (x_{1j}, ..., x_{nj})^T \in \mathbb{R}^n$, für ein $j \in \{1, ..., p\}$, bezüglich derer optimiert werden soll. Das LASSO reduziert sich hierbei auf das eindimensionale Problem

$$\min_{\beta_j \in \mathbb{R}} \left\{ f_j(\beta_j) := \frac{1}{2n} \sum_{i=1}^{n} (y_i - x_{ij}\beta_j)^2 + \lambda|\beta_j| \right\} .$$

$f_j(\beta_j)$ ist streng konvex bezüglich β_j, sodass $\hat{\beta}_j = \text{argmin } f(\beta_j)$ eindeutig ist. Die

Optimalitätsbedingung für $\hat{\beta}_j$ lautet

$$\begin{aligned} & \frac{\partial f_j}{\partial \beta_j}(\hat{\beta}_j) = -\frac{1}{n}\sum_{i=1}^{n} x_{ij}(y_i - x_{ij}\hat{\beta}_j) + \lambda s_j = 0 \\ \iff & \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i - \hat{\beta}_j \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = \lambda s_j \ , \end{aligned}$$

wobei $s_j = \text{sign}(\hat{\beta}_j)$, falls $\hat{\beta}_j \neq 0$, $s_j \in [-1, 1]$ sonst. Umstellen nach $\hat{\beta}_j$ unter Berücksichtigung der Standardisierung $\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1$ liefert

$$\hat{\beta}_j \ = \ \begin{cases} \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i - \lambda \ , & \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i \ > \ \lambda \\ 0 \ , & \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i \ \leq \ \lambda \\ \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i + \lambda \ , & \frac{1}{n}\sum_{i=1}^{n} x_{ij}y_i \ < \ -\lambda \ . \end{cases}$$

$\hat{\beta}_j$ ist somit durch den Soft-Thresholding Operator (vgl. (2.29)) gegeben als

$$\hat{\beta}_j = S\left(\frac{1}{n}\sum_{i=1}^{n} y_i x_{ij} \ , \ \lambda\right) \ .$$

Die Optimierung bezüglich einer Variable ist nun auf das vollständige LASSO übertragbar. Für jede Variable $j \in \{1, ..., p\}$ wird in der Iteration k das aufgrund der strengen Konvexität eindeutig lösbare Optimierungsproblem

$$\hat{\beta}_j^{(k+1)} = \ \underset{\beta_j \in \mathbb{R}}{\text{argmin}} \ f_{LASSO}(\hat{\beta}_1^{(k+1)}, ..., \hat{\beta}_{j-1}^{(k+1)}, \beta_j, \hat{\beta}_{j+1}^{(k)}, ..., \hat{\beta}_p^{(k)}) \tag{2.34}$$

bezüglich β_j mittels CD gelöst, während alle anderen Variablen jeweils bei deren aktuellem Wert des Koeffizienten als fest angenommen werden, wobei $\hat{\beta}_l^{(k+1)} = \hat{\beta}_l^{(k)} \ \forall l > j$. Hierbei ist zu beachten, dass alle in der Iteration k berechneten Koeffizienten $\hat{\beta}_1^{(k+1)}$,..., $\hat{\beta}_{j-1}^{(k+1)}$ für die Bestimmung des nächsten Koeffizienten $\hat{\beta}_j^{(k)}$ verwendet werden, vgl. dazu die Iterierte (2.34).

Zur Hervorhebung der Variable j schreibe $f_{LASSO}(\beta)$ in (2.6) als

$$f_{LASSO}(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \sum_{k \neq j} x_{ik}\beta_k - x_{ij}\beta_j)^2 + \lambda \sum_{k \neq j} |\beta_k| + \lambda|\beta_j| \ .$$

Bilden der partiellen Ableitung $\frac{\partial f_{LASSO}}{\partial \beta_j}(\hat{\beta}_j)$, Setzen auf Null und Auflösen nach $\hat{\beta}_j$ ergibt

$$\hat{\beta}_j = \frac{1}{n} x_{ij} \sum_{i=1}^{n}(y_i - \sum_{k \neq j} x_{ik}\beta_k) - \lambda s_j \ . \tag{2.35}$$

Für eine Lösung $\hat{\beta}_j$ ist

$$r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik}\beta_k = y_i - \hat{y_i}^{(j)} = y_i - \hat{y}_i + x_{ij}\hat{\beta}_j = r_i + x_{ij}\hat{\beta}_j$$

das Residuum für die Beobachtung i, bei dem der Beitrag von $\hat{\beta}_j$ nicht berücksichtigt wird. $\hat{y}_i$ ist der aktuelle Output für die Beobachtung i. Einsetzen in die Gleichung (2.35) ergibt für den Koeffizienten $\hat{\beta}_j$

$$\hat{\beta}_j = S\left(\frac{1}{n}\sum_{i=1}^{n} x_{ij}(y_i - \hat{y_i}^{(j)}) \ , \ \lambda\right) = S\left(\frac{1}{n}\sum_{i=1}^{n} x_{ij} r_i^{(j)} \ , \ \lambda\right) \ . \tag{2.36}$$

Bei der Optimierung bezüglich β_j wird somit zunächst die Lösung der kleinsten Quadrate mit dem Residuum $r_i^{(j)}$ bestimmt und anschließend wird auf diese der Soft-Thresholding Operator angewendet. Die Optimierung bezüglich β_j für alle $j \in \{1, ..., p\}$ wird solange durchgeführt, bis Konvergenz erreicht ist. Üblicherweise entspricht das Abbruchkriterium dem Erreichen einer gewünschten Fehlertoleranzgrenze der Differenz von $\hat{\beta}^{(k)}$ und $\hat{\beta}^{(k+1)}$, bspw. gemessen an der L2-Norm.

Das CD besteht somit aus zwei Schleifen. In der Iteration j, $j \in \{1, ..., p\}$, der inneren Schleife des CD erfolgt die Optimierung bezüglich des Koeffizienten β_j (Schritt 6 des Algorithmus 2), in der Iteration k der äußeren Schleife werden die Lösungen der inneren Schleife zu einem $\hat{\beta}^{(k)}$ zusammengesetzt und die Optimalitätsbedingung überprüft. Der Algorithmus zielt darauf ab, eine Approximation $\hat{\beta}$ von $\beta_L \in \operatorname{argmin} f_{LASSO}$ zu bestimmen.

Algorithmus 2 Coordinate Descent für LASSO

1: X, y, λ, $tol \geq 0$ Fehlertoleranzgrenze.
2: $k = 0$, $\hat{\beta}^{(0)} = (\hat{\beta}_1^{(0)}, ..., \hat{\beta}_p^{(0)})^T$ Startlösung.
3: **repeat**
4: $\quad k = k + 1$.
5: $\quad$ **for** $j = 1 : p$ **do**
6: $\quad\quad$ Berechne $\hat{\beta}_j^{(k)}$ mittels der Gleichung (2.36).
7: $\quad$ **end for**
8: $\quad \hat{\beta}^{(k)} = (\hat{\beta}_1^{(k)}, ..., \hat{\beta}_p^{(k)})$.
9: **until** $\|\hat{\beta}^{(k)} - \hat{\beta}^{(k-1)}\| \leq tol$.
10: $\hat{\beta} := \hat{\beta}^{(k)}$ ist die resultierende Lösung.

Es ist üblich, eine stark dünnbesetzte Startlösung $\hat{\beta}^{(0)}$ zu wählen, meist startet der Algorithmus mit $\hat{\beta}^{(0)} = 0_{|p|}$. Den größten Aufwand benötigt der Algorithmus für die Berechnung von $\frac{1}{n}\sum_{i=1}^{n} x_{ij} r_i^{(j)}$. Wegen der Standardisierung der Daten ergibt

sich

$$\frac{1}{n}\sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i^{(j)}) = \frac{1}{n}\sum_{i=1}^{n} x_{ij} r_i \ + \hat{\beta}_j \ . \tag{2.37}$$

Die Summe $\sum_{i=1}^{n} x_{ij} r_i$ lässt sich schreiben als

$$\sum_{i=1}^{n} x_{ij} r_i = \langle X_j, y \rangle \ - \sum_{k:\ \hat{\beta}_k \neq 0} \langle X_j, X_k \rangle \hat{\beta}_k \ .$$

Das Skalarprodukt $\langle X_j, y \rangle_{j \in \{1,\dots,p\}} \in \mathbb{R}^p$ jeder Variable mit dem Output y wird bereits zu Anfang berechnet und gespeichert. Der Aufwand hierfür beträgt einmalig $O(np)$ arithmetische Operationen, und zwar benötigt $\langle X_j, y \rangle$ n Operationen, das Skalarprodukt wird insgesamt p Mal gebildet. Tritt erstmals eine neue Variable k hinzu, so wird das Skalarprodukt $\langle X_k, X_j \rangle \ \forall j \in \{1, ..., k-1, k+1, .., p\}$ berechnet und die Ergebnisse werden ebenso gespeichert. Der Rechenaufwand für diesen Schritt beläuft sich auf $n(p-1) = np - n = O(np)$ Operationen. Für die nächste hinzukommende Variable sind $n(p-2)$ Operationen nötig. In jeder Iteration entfallen somit n Operationen. Falls sich ein Koeffizient in der aktuellen Iteration ändert, so kostet die Anpassung der gespeicherten Subdifferentiale $O(p)$ Operationen. Folglich beläuft sich der gesamte Aufwand für das CD (Algorithmus 2) in jeder Iteration auf $O(np)$ arithmetische Operationen, wobei in jeder Iteration sich der Aufwand um n Operationen reduziert.

Das CD für LASSO ist konvergent. Es gilt $f_{LASSO}(\hat{\beta}^{(k)}) \overset{k\to\infty}{\longrightarrow} f_L$, wobei $f_L := \min f_{LASSO}$ der aufgrund strenger Konvexität eindeutige optimale Funktionswert von f_{LASSO} ist. Daraus folgt, dass die Sequenz $\hat{\beta}^{(k)}$ gegen eine Lösung von LASSO konvergiert, $\hat{\beta}^{(k)} \overset{k\to\infty}{\longrightarrow} \beta_L \in \operatorname{argmin} f_{LASSO}$, denn für jedes $\beta_L \in \operatorname{argmin} f_{LASSO}$ gilt $f_{LASSO}(\beta_L) = f_L$. Im Falle einer eindeutigen Lösung konvergiert das CD folglich gegen diese. Ansonsten hängt die resultierende Lösung $\hat{\beta}$ meist von der Startlösung $\hat{\beta}^{(0)}$ ab.

Die Konvergenz des CD für konvexe, nicht differenzierbare Probleme hat Tseng (2001) [40] gezeigt, wobei insbesondere die Form der zu optimierenden Funktion entscheidend ist. Konvergenz kann nur dann garantiert werden, wenn der nicht differenzierbare Anteil, hier $\lambda\|\beta\|_1$, als Summe von Funktionen der einzelnen Parameter, hier $\lambda|\beta_j|$, völlig separierbar ist. Der Beweis dieser Aussage ist recht technisch, benötigt einige zusätzliche Definitionen und ist daher kein Bestandteil dieser Arbeit. Alle in den folgenden Kapiteln zu behandelnden Optimierungsprobleme haben die für die Anwendung von CD und dessen Konvergenz nötigen Eigenschaften, sodass dies in den folgenden Kapiteln nicht mehr erwähnt wird.

Wichtig zu beachten ist, dass aus einer Startlösung aufgrund des eindeutig lösbaren Problems (2.34) genau eine Lösung resultiert, unabhängig davon, ob LASSO

eine oder unendlich viele Lösungen hat. Dies bedeutet, dass das CD für eine Startlösung genau ein $\hat{\beta}$ liefert, das entweder der Approximation der eindeutigen oder einer der unendlichen vielen Lösungen von LASSO entspricht. Aus zwei verschiedenen Startlösungen geht somit jeweils ein eindeutiger Koeffizientenvektor hervor, wobei die beiden Vektoren bei unendlichen vielen Lösungen von LASSO (meist) nicht übereinstimmen.

Wie bereits erläutert, sind meist Lösungen für mehrere RP von Interesse. Das CD kann für diese Zwecke ebenfalls zum Einsatz kommen, indem es auf das *Pathwise Coordinate Descent* (Pathwise CD, auch *Cyclical Coordinate Descent*) erweitert wird [14].

$\lambda_{\max}$ bezeichne den kleinsten Wert für λ, sodass $\beta_L(\lambda) = 0_{|p|}\ \forall \lambda \geq \lambda_{\max}$. Betrachte die aus den KKT-Bedingungen resultierende Ungleichung $\frac{1}{n}|X^T(y - X\hat{\beta})| \leq \lambda$, vgl. dazu die Gleichung (2.20) und die Ungleichung (2.21). Gilt Ungleichheit für ein $j \in \{1, ..., p\}$, so ist das $\beta_{j,L} = 0$ und somit $\frac{1}{n}|X_j^T y| < \lambda$. Hieraus folgt, dass das $\lambda_{\max}$ gegeben ist durch

$$\lambda_{\max} \ = \ \frac{1}{n} \max_{j \in \{1,...,p\}} |X_j^T y| \ = \ \frac{1}{n} \|X^T y\|_\infty \ . \tag{2.38}$$

Pathwise CD berechnet Lösungen für eine absteigende Sequenz von RP $\lambda_{\max} = \lambda_0 > \lambda_1 > ... > \lambda_r = \lambda_{\min}$ auf der logarithmischen Skala, wobei der resultierende Koeffizientenpfad für eine Startlösung eindeutig ist.

Bühlmann und van de Geer (2011) [9] schlagen die folgende Sequenz auf der logarithmischen Skala vor:

$$\lambda_l = \lambda_{l-1}\exp(-C),\ C > 0 \text{ konstant} \ .$$

Mit dieser Definition ergibt sich für C

$$\begin{aligned} & \lambda_1 = \lambda_0\exp(-C), \quad \lambda_2 = \lambda_1\exp(-C) = \lambda_0\exp(-C)^2 \ , \\ & \quad ... \ , \\ & \lambda_r = \lambda_0\exp(-C)^m \\ \iff \quad & C = \frac{\log(\lambda_0) - \log(\lambda_m)}{m} = \frac{\log(\lambda_{\max}) - \log(\lambda_{\min})}{m} \ . \end{aligned}$$

Die so definierte Sequenz ist äquidistant und unabhängig von den Daten.

Friedman et al. (2010) [15], Hastie et al. (2016) [21] wählen das $\lambda_{\min}$ im Vorhinein als $\lambda_{\min} = \epsilon\lambda_{\max}$ mit $\epsilon = 0.001$ und $m = 100$, λ_1 bis λ_{m-1} werden im Laufe des Algorithmus bestimmt. Die Lösungen zu den einzelnen λ_l, $l = 1, ..., m$, sind somit als $\beta_L(\lambda_l) \in \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda_l\|\beta\|_1$ aufzufassen.

Die Startlösung für das Pathwise CD ist $\hat{\beta}(\lambda_0) = \hat{\beta}(\lambda_{\max}) = 0_{|p|}$ zu $\lambda_{\max}$. Als nächstes wird die Lösung $\hat{\beta}(\lambda_1)$ zu λ_1 mittels CD (Algorithmus 2), ermittelt, wobei die Startlösung das $\hat{\beta}(\lambda_0)$ ist. Ist $\hat{\beta}(\lambda_1)$ berechnet worden, so wird zu λ_2 mit Hilfe des CD die Lösung $\hat{\beta}(\lambda_2)$ bestimmt, wobei hierzu $\hat{\beta}(\lambda_1)$ als Startlösung eingesetzt wird. Dies wird für alle RP durchgeführt. Die äußere Schleife des Pathwise CD durchläuft

Algorithmus 3 Pathwise Coordinate Descent für LASSO

1: X, y, $\lambda_0, ..., \lambda_r$, $tol \geq 0$ Fehlertoleranzgrenze für das CD (Algorithmus 2).
2: $\lambda_0 = \lambda_{\max}$ (2.38), $\hat{\beta}(\lambda_0) = 0_{|p|}$.
3: **for** $l = 1 : m$ **do**
4: $\quad \hat{\beta}(\lambda_{l-1})$ Startlösung.
5: $\quad$ Berechne $\hat{\beta}(\lambda_l)$ mittels Algorithmus 2.
6: **end for**

alle λ_l, $l = 1, ..., m$, die innere Schleife verwendet das CD (Algorithmus 2), indem die einzelnen $\hat{\beta}_j(\lambda_l)$, $j = 1, ..., p$ berechnet und zu einer Lösung $\hat{\beta}(\lambda_l)$ zusammengesetzt werden. Hierbei wird bei jeder Berechnung einer Lösung $\hat{\beta}(\lambda_l)$ zu λ_l, $l \in \{1, ..., m\}$, die vorher bestimmte Lösung $\hat{\beta}(\lambda_{l-1})$ zu λ_{l-1} als Startlösung eingesetzt. Friedmann et al. [15] bezeichnen diese Vorgehensweise als $Warm\ Starts$. Warm Starts führen zur Stabilität und zur Effizienz von Pathwise CD. Zum einen besteht insbesondere in den ersten Iterationen wegen der Dünnbesetztheit der Lösungsvektoren ein geringer Rechenaufwand, zum anderen ist die nach Iteration l resultierende Lösung $\hat{\beta}(\lambda_l)$ eine recht gute Startlösung für das zu berechnende $\hat{\beta}(\lambda_{l+1})$, sodass die Konvergenz des CD (Algorithmus 2) in jeder Iteration schnell erreicht wird. Daher ist es oft effizienter, anstatt eine Lösung zu einem λ Lösungen für eine Sequenz von RP $\lambda_0 > \lambda_1 > ... > \lambda_r$ zu bestimmen.

Um die Konvergenz des Pathwise CD zu beschleunigen, schlagen die Autoren vor, in der Iteration l das CD (Algorithmus 2) nur für die Variablen $j \in J = \text{supp}(\hat{\beta}(\lambda_{l-1}))$ durchzuführen. Sobald der Algorithmus konvergiert, wird für die Variablen $j \notin J$ überprüft, ob diese mit dem aktuellen Residuum r die Ungleichung $\frac{1}{n}|X_j^T r| < \lambda_l$ erfüllen (vgl. die Gleichung (2.38)). λ_l kann in Iteration l demnach als $\lambda_{\max}$ in dieser Iteration aufgefasst werden. Ist dies der Fall, so ist die Lösung $\hat{\beta}(\lambda_l)$ gefunden, mit den berechneten Einträgen in J und sonst Nullen für $j \notin J$. Gibt es $j \notin J$, die diese Bedingung nicht erfüllt, so werden diese Variablen der Indexmenge J hinzugefügt und die Iteration wiederholt. Auf diese Weise verbleiben alle Variablen, dessen Koeffizient entlang der Lösungssequenz jemals zum Support einer Lösung gehörte, bis zum Ende des Algorithmus im Modell. Im Kapitel 2.5 werden weitere Möglichkeiten zur Eliminierung von Variablen vorgestellt.

2.3.3 Proximal Gradient Descent

Die Aussagen dieses Kapitels wurden [21, 30] entnommen.

Das klassische Gradientenverfahren berechnet für eine differenzierbare Funktion $f(\beta)$ eine Folge von Lösungen $\{\beta^{(k)}\}_{k\in\mathbb{N}}$ mit der Vorschrift

$$\beta^{(k+1)} = \beta^{(k)} - t_k \nabla f(\beta^{(k)}),\ k \geq 0\ , \tag{2.39}$$

bis Konvergenz erreicht ist. $\{t_k\}_{k\in\mathbb{N}}$, $t_k > 0\ \forall k \in \mathbb{N}$, ist eine Folge von Schrittweiten. Diese Vorschrift lässt sich umformulieren zu

$$\beta^{(k+1)} = \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ f^{(k)}(\beta) := f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle + \frac{1}{2t_k} \|\beta - \beta^{(k)}\|_2^2 \right\}\ , \tag{2.40}$$

denn Bilden des Gradienten von $f^{(k)}(\beta)$, Setzen auf Null als notwendige Bedingung für Extrema und Auflösen nach $\beta^{(k+1)}$ ergibt gerade die Gleichung (2.39):

$$\nabla f^{(k)}(\beta^{(k+1)}) = \nabla f(\beta^{(k)}) + \frac{1}{s_t}(\beta^{(k+1)} - \beta^{(k)}) = 0 \iff \beta^{(k+1)} = \beta^{(k)} - t_k \nabla f(\beta^{(k)})\ .$$

Der blau markierte Term in der Optimierungsfunktion $f^{(k)}(\beta)$ (2.40) ist das erste Taylor-Polynom und kann als eine lokale lineare Approximation von $f^{(k)}(\beta)$ an der aktuellen Lösung $\beta^{(k)}$ interpretiert werden, siehe Anhang A5. Der zweite Summand kann als L2-Regularisierung mit der quadrierten L2-Norm des Residuums $\beta - \beta^{(k)}$ mit $\frac{1}{2t_k}$ als RP aufgefasst werden, sodass der Definitionsbereich D für das gesuchte $\beta^{(k+1)}$ auf eine konvexe Menge eingeschränkt wird[9]. Somit spielt dieser bei der Suche nach $\beta^{(k+1)}$, das nah am aktuellen $\beta^{(k)}$ liegt, eine entscheidende Rolle.

Die sogenannten *Proximal Gradient Methods* sind eine Erweiterung des klassischen Gradientenverfahrens (engl. Gradient Descent) auf konvexe nicht differenzierbare Probleme wie LASSO, deren Optimierungsfunktion die Form

$$f(\beta) = g(\beta) + h(\beta) \tag{2.41}$$

hat, wobei $g(\beta)$ konvex und differenzierbar und $h(\beta)$ konvex, hingegen nicht differenzierbar ist. Das Proximal Gradient Descent führt dieselbe Approximation wie beim Gradientenverfahren für $g(\beta)$ durch (2.40), $h(\beta)$ bleibt hingegen unverändert,

[9] Zur Erinnerung, wegen der Lagrange-Dualität ist die L2-Regularisierung $\frac{1}{2t_k}\|\beta - \beta^{(k)}\|_2^2$ äquivalent dazu, dass eine eindeutige Konstante C zu $\frac{1}{2t_k}$ existiert, sodass $\|\beta - \beta^{(k)}\|_2^2 \leq C$ gilt.

sodass die nächste Iterierte folgendermaßen bestimmt wird:

$$\begin{aligned}\beta^{(k+1)} &= \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ g(\beta^{(k)}) + \langle \nabla g(\beta^{(k)}), \beta - \beta^{(k)} \rangle + \frac{1}{2t_k}\|\beta - \beta^{(k)}\|_2^2 + h(\beta) \right\} \\ &= \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2t_k}\|\beta^{(k)} - t_k \nabla g(\beta^{(k)}) - \beta\|_2^2 + h(\beta) \right\} . \end{aligned} \tag{2.42}$$

$h(\beta)$ ist nach Voraussetzung konvex, der erste Summand in (2.42) ist streng konvex bzgl. β, woraus Eindeutigkeit von $\beta^{(k+1)}$ folgt.

Definiere den sogenannten *Proximal Operator* (auch *Proximal Mapping*) der Funktion $h(\beta)$ zu b mit einer Konstante t als

$$\operatorname{prox}_{th}(b) := \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2t}\|b - \beta\|_2^2 + h(\beta) .$$

Für $\beta^{(k+1)}$ gegeben durch die Gleichung (2.42) folgt nun die Darstellung

$$\beta^{(k+1)} = \operatorname{prox}_{t_k h}\left(\beta^{(k)} - t_k \nabla g(\beta^{(k)})\right) . \tag{2.43}$$

Für LASSO mit $g(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2$, $\nabla g(\beta) = -\frac{1}{n}X^T(y - X\beta)$ und $h(\beta) = \lambda\|\beta\|_1$ ergibt sich

$$\beta^{(k+1)} = \operatorname{prox}_{t_k h}\left(\beta^{(k)} + t_k \frac{1}{n} X^T(y - X\beta^{(k)})\right) . \tag{2.44}$$

Für $\beta^{(k+1)}$ (2.44) lässt sich eine explizite Form herleiten. Betrachte zunächst den Proximal-Operator $\operatorname{prox}_{\lambda\|\beta\|_1}(b) := \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2}\|b - \beta\|_2^2 + \lambda\|\beta\|_1 \right\}$ von $\lambda\|\beta\|_1$, wobei $t = 1$. Die notwendige Bedingung für das gesuchte Minimum $\hat{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_p)^T := \operatorname{prox}_{\lambda\|\beta\|_1}(b)$ liefert

$$\hat{\beta}_j = \begin{cases} b_j - \lambda, & b_j > \lambda \\ 0, & |b_j| \leq \lambda \\ b_j + \lambda, & b_j < -\lambda . \end{cases}$$

Es folgt $\hat{\beta} = S(b_j, \lambda)$, wobei $S(b_j, \lambda)$ dem Soft-Thresholding-Operator definiert als (2.29) entspricht. Für die nächste Iterierte $\beta^{(k+1)}$ (2.44) resultiert

$$\beta^{(k+1)} = S\left(\beta^{(k)} + t_k \frac{1}{n} X^T(y - X\beta^{(k)}), t_k\lambda\right) , \tag{2.45}$$

wobei die Anwendung von $S(\cdot, \lambda)$ elementweise aufzufassen ist. Algorithmen, die Probleme der Art (2.41) lösen, indem der Soft-Thresholding Operator zum Einsatz kommt, sind auch unter dem Namen *Iterative Soft-Thresholding Algorithms* (ISTA) bekannt [2].

Das Proximal Gradient Descent konvergiert bei einer geeignet gewählten Sequenz

Algorithmus 4 Proximal Gradient Descent für LASSO

1: X, y, λ, $tol \geq 0$ Fehlertoleranzgrenze, $\{t_k\}_{k\in\mathbb{N}_0}$.
2: $k = 0$, $\beta^{(0)}$ Startlösung.
3: **repeat**
4: Berechne $b = \beta^{(k)} + t_k \frac{1}{n} X^T(y - X\beta^{(k)})$.
5: Berechne $\beta^{(k+1)} = \text{prox}_{t_k\lambda\|\beta\|_1}(b) = S(b, t_k\lambda)$.
6: $k = k + 1$.
7: **until** $\|\beta^{(k)} - \beta^{(k-1)}\| \leq tol$.
8: $\hat{\beta} := \beta^{(k)}$ ist die resultierende Lösung.

von Schrittweiten $\{t_k\}_{k\in\mathbb{N}_0}$, und zwar wie bei CD in dem Sinne, dass $f_{LASSO}(\beta^{(k)}) \overset{k\to\infty}{\longrightarrow} f_L$ gilt. Aufgrund dessen konvergiert die Folge $\beta^{(k)}$ gegen eine Lösung von LASSO $\beta_L \in \text{argmin } f_{LASSO}$. Hat LASSO keine eindeutige Lösung, so wirkt sich die Startlösung $\beta^{(0)}$ auf das resultierende β_L aus. Um Konvergenz zu gewährleisten, muss für die Schrittweite $t_k \in (0, 2n/\|X\|_2)$ $\forall k \in \mathbb{N}_0$ gelten, wobei $\|X\|_2$ die Spektralnorm der Matrix X ist. Es gilt $\|X\|_2 = |\mu_{\max}(X^TX)| = \mu_{\max}(X^TX)$, wobei $\mu_{\max}(X^TX)$ der (betragsmäßig[10]) maximale Eigenwert der Matrix X^TX ist [2]. Somit ist eine konstante Schrittweite $t = t_k$ $\forall k \in \mathbb{N}_0$ zulässig, sodass in dem obigen Algorithmus t_k durch t ersetzt werden kann. Ist t geeignet gewählt, so gilt

$$|f_{LASSO}(\beta^{(k)}) - f_L| \leq \frac{L\|\beta^{(0)} - \beta_L\|^2}{2k} \quad \forall \beta_L \in \text{argmin } f_{LASSO} \ , \tag{2.46}$$

sodass die Konvergenzrate $O(\frac{1}{k})$ beträgt [2]. Für den Beweis und Herleitung siehe [2]. $L := \frac{1}{n}\|X^TX\|_2 = \frac{1}{n}\mu_{\max}(X^TX)$ ist dabei die Lipschitz-Konstante des Gradienten von $\frac{1}{2n}\|y - X\beta\|_2^2$, siehe Anhang A4.

Den größten Rechenaufwand in jeder Iteration stellt das Produkt $X^T(y - X\beta^{(k)}) = X^Ty - X^TX\beta^{(k)}$ dar. Allerdings müssen $\tilde{y} = \frac{1}{n}X^Ty$, $O(np)$ arithmetische Operationen, und $\tilde{X} = X^TX$, $O(np \cdot \min(n, p))$ Rechenoperationen, nur einmal zu Anfang durchgeführt werden, sodass im Schritt 4 des Algorithmus 4 der Term $X^T(y - X\beta^{(k)})$ durch $\tilde{y} - \tilde{X}\beta^{(k)}$ ersetzt wird. Zudem ist die Matrix X^TX symmetrisch, daher ist es ausreichend, die Diagonale und die Werte oberhalb bzw. unterhalb der Diagonalen zu bestimmen. Berechnen von $\tilde{X}\beta^{(k)}$ benötigt $O(p^2)$ Operationen, der Rechenaufwand für die Auswertung von $S(b, \lambda)$ ist vernachlässigbar. Der gesamte Aufwand einer Iteration beträgt somit $O(p^2)$ arithmetische Operationen.

Die Berechnung der Lösung ist ähnlich zu CD, vgl. (2.34), insbesondere bei Vorliegen standardisierter Daten und der Schrittweite $t_k = 1$ $\forall k \in \mathbb{N}_0$. Dabei lässt sich nicht verallgemeinern, welches der beiden Verfahren effizienter ist. Das CD berechnet

[10] Da die Matrix X^TX symmetrisch positiv (semi-)definit ist, sind alle Eigenwerte größer (oder gleich) Null.

in der Iteration k die Einträge $\beta^{(k)}$ nacheinander, das Proximal Gradient Descent wendet den Soft-Thresholding-Operator gleichzeitig auf alle Koeffizienten an. Hierbei müssen jedoch die Schrittweiten geeignet gewählt werden, denn ohne die Optimierung ist die Garantie der Konvergenz gegen eine Lösung $\beta_L \in \operatorname{argmin} f_{LASSO}$ nicht gewährleistet. Die Schrittweiten $\{t_k\}_{k\in\mathbb{N}_0}$ hängen zudem von dem maximalen Eigenwert von X^TX ab, dieser ist meist nicht bekannt und dessen Berechnung kann abhängig von der Dimension p sogar den Aufwand des gesamten Algorithmus übersteigen. Das CD hingegen verspricht Konvergenz und profitiert stark von der Dünnbesetztheit der Iterierten. Beim Vergleich des Rechenaufwandes ist im Falle von $n \ll p$ davon auszugehen, dass das CD zumindest hinsichtlich des rechnerischen Aufwandes ein besseres Verhalten aufweist, denn das CD benötigt $O(np)$, Proximal Gradient Descent $O(p^2)$ Operationen. Im Kapitel 3.3 wird eine Version des Proximal Gradient Descent vorgestellt, die eine Konvergenzrate von $O(\frac{1}{k^2})$ aufweist.

2.4 Generalisierte Lineare Modelle und Lösungsalgorithmen

Bisher wurde die klassische lineare Regression behandelt, bei der die Zielvariable y quantitativ ist und als normalverteilt angenommen wird. In der Praxis tauchen jedoch oft Probleme auf, deren Zielvariable y einer anderen Art ist, bspw. diskret. Hierbei handelt es sich um *generalisierte lineare Modelle* (GLM). Unter diesem Namen werden vor allem diejenigen Modelle zusammengefasst, deren Fehlerterm ϵ eine Verteilung aus der Klasse der exponentiellen Familien aufweist, wobei die Normalverteilung zu dieser Klasse gehört. Bei einer diskreten Zielvariable y mit $K = 2$ (binär) bzw. $K > 2$ (multinomial) Ausprägungen besteht die Annahme einer Binomial- bzw. Multinomialverteilung. In der Überlebenszeitanalyse kommt die Cox-Regression zum Einsatz. Da y nicht stetig ist, können das System (1.1) und die Verlustfunktion $f_{OLS} = \|y - X\beta\|_2^2$ in der Form nicht mehr verwendet werden und bedürfen daher einer sachgerechten Transformation. Die jeweiligen Verlustfunktionen fallen bei Vorliegen des klassischen linearen Problems mit der der kleinsten Quadrate zusammen. Wie bisher bezeichnen $(x^i, y_i)_{i=1}^n$ die Beobachtungspaare.

2.4.1 Logistische und multinomiale Regression

Ist die Zielvariable y diskret mit $K = 2$ oder $K > 2$ Ausprägungen, so wird das ursprüngliche Problem der klassischen linearen Regression (1.1) zu einem Klassifikationsproblem. In diesem Fall werden bei zwei Klassen eine, bei mehr als zwei Klassen $K \geq 2$ Hyperebenen gesucht, die die Beobachtungen nach deren Klassenzugehörigkeit bezüglich der jeweiligen Ziel- bzw. Verlustfunktion bestmöglich trennen. Die jeweilige Funktion, die den Zusammenhang modelliert, drückt hierbei die Wahr-

scheinlichkeiten der Zugehörigkeit der Beobachtungen zu der jeweiligen Klasse aus.

Logistische Regression Sei $x \in \mathbb{R}^p$ und o.b.d.A. sei $y \in \{0, 1\}$. Bei einer binären Zielvariable y erfolgt meist der Einsatz der logistischen Regression. Hierbei

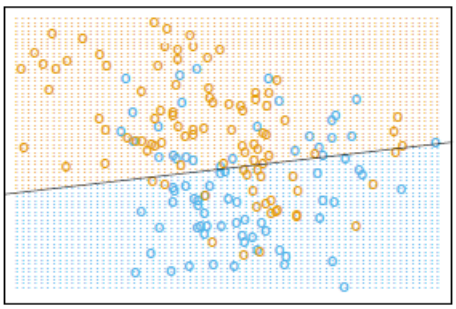


Abbildung 5 Beispiel logistischer Regression in 2D. Die orangefarbenen Punkte entsprechen o.B.d.A. $y_i = 0$, die blauen $y_i = 1$, $i \in \{1, ..., n\}$. Die Gerade entspricht $x^T\hat{\beta} = 0.5$. Anhand der Gerade erhalten alle neuen oberhalb liegenden Beobachtungen die Ausprägung $y_i = 0$, alle unterhalb liegenden Beobachtungen $y_i = 1$, $i > n$. Modifiziert nach [19], S.13.

steht die Approximation der bedingten Wahrscheinlichkeiten im Fokus:

$$\begin{aligned} \Pr(y = 1|x) = \mathbb{E}(y = 1|x) &= \frac{e^{\beta_0+\beta^T x}}{1 + e^{\beta_0+\beta^T x}} = \frac{1}{1 + e^{-(\beta_0+\beta^T x)}} =: p \ \in (0, 1) \ , \\ \Pr(y = 0|x) = \ &\frac{1}{1 + e^{\beta_0+\beta^T x}} = 1 - p \ \in (0, 1) \ . \end{aligned} \tag{2.47}$$

Dies ist äquivalent zu

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta^T x \ . \tag{2.48}$$

Der Bruch $\frac{p}{1-p}$ ist der sogenannte *Odd* (auch Chance), der das Verhältnis der Wahrscheinlichkeit eines Ereignisses zu der Gegenwahrscheinlichkeit unter x ausdrückt. Die bedingten Wahrscheinlichkeiten (2.47) werden folglich so definiert, dass sich der Odd linear verhält, $\text{logit}(p)$ ist der sogenannte *Logit* des Odd's. Die Logarithmusfunktion ist (streng) monoton, sodass die Transformation des Odd's wohldefiniert ist. Die Entscheidung, zu welcher Klasse eine Beobachtung gehört, wird anhand der Hyperebene $H := \{x \ | \beta_0 + \beta^T x = 0\}$ getroffen, die die Grenze zwischen den beiden Klassen darstellt. Auf dieser liegen also gerade alle Punkte, für die $\text{logit}(p)$ (2.48) den Wert 0 annimmt [19].

Die Verlust-/Zielfunktion für LASSO ist hierbei die (mit $\frac{1}{n}$ skalierte) zu maximie-

rende *Log-Likelihood-Funktion*

$$\frac{1}{n}\mathcal{L}(\beta_0, \beta) := \frac{1}{n}\sum_{i=1}^{n}\Big(y_i \log(p(x^i)) + (1 - y_i)\log(1 - p(x^i))\Big)\ , \tag{2.49}$$

Aus den Rechenregeln für den Logarithmus und (2.48) mit $p(x^i) = \Pr(y = 1 \mid x^i)$ und $\tilde{y}_i = \mathbb{I}(y_i = 1)$ folgt

$$\mathcal{L}(\beta_0, \beta) = \sum_{i=1}^{n}\Big(\tilde{y}_i(\beta_0 + \beta^T x^i) - \log\Big(1 + e^{\beta_0 + \beta^T x^i}\Big)\Big)\ .$$

Es ist zwar wünschenswert, dass es keine Überlappungen der Klassen gibt, sodass die Beobachtungen durch die gesuchte Hyperebene perfekt trennbar sind, nur scheitert die logistische Regression in diesem Fall [21]. Die Regressionskoeffizienten des resultierenden β_L, für das die Log-Likelihood-Funktion maximal ist, nehmen keinen reellen Wert an, sondern $\beta_{j,L} \in \pm\infty\ \forall j \in 1, ...p$, da die Log-Likelihood bestrebt ist, für die bedingten Wahrscheinlichkeiten der einzelnen Beobachtungen den Wert Null bzw. Eins zu erreichen [15, 21]. Denn die Exponentialfunktion ist zwar streng konvex, ohne Einschränkung des Definitionsbereiches hat diese dennoch kein globales Minimum. Bei $n \ll p$ lassen sich die Klassen oft perfekt trennen. Außerdem findet hierbei wie zu Anfang bereits erklärt meist starkes Overfitting statt. Daher ist Regularisierung zur Schrumpfung der Koeffizienten erforderlich, um ein interpretierbares Ergebnis zu erhalten. Dabei ist zu beachten, dass der RP λ hinreichend groß sein muss, denn für $\lambda \to 0$ folgt $\beta_{j,L} \to \pm\infty\ \forall j \in \{1, ..., p\}$, und dasselbe Problem bleibt bestehen [15, 21].

Es gibt zwei äquivalente Möglichkeiten, das LASSO für die logistische Regression (analog in den folgenden Kapiteln) zu formulieren, und zwar als eine Maximierungsmit der (mit $\frac{1}{n}$ skalierten) Log-Likelihood-Funktion $\frac{1}{n}\mathcal{L}(\beta_0, \beta)$ oder eine Minimierungsaufgabe mit der negativen (mit $\frac{1}{n}$ skalierten) Log-Likelihood-Funktion $-\frac{1}{n}\mathcal{L}(\beta_0, \beta)$ als Verlustfunktion:

$$\begin{aligned} & \max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left\{ f^{log}_{LASSO} := \frac{1}{n}\mathcal{L}(\beta_0, \beta) - \lambda\|\beta\|_1 \right\} \\ \iff \quad & \min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left\{ f^{log}_{LASSO} := -\frac{1}{n}\mathcal{L}(\beta_0, \beta) + \lambda\|\beta\|_1 \right\}\ . \end{aligned} \tag{2.50}$$

In der Literatur wird LASSO für die logistische Regression meist als die Minimierungsaufgabe (2.50) aufgefasst. Es gilt $y_i(\beta_0 + \beta^T x) \in \{0, \beta_0 + \beta^T x\}$ wegen $y_i \in \{0, 1\}$, 0 und $\beta_0 + \beta^T x$ sind als lineare Funktionen sowohl konkav als auch konvex, $\log(\cdot)$ ist streng konkav, da aber $e^{+(\cdot)}$ streng konvex ist und schneller wächst als $\log(\cdot)$, ist $\log(1 + e^{+(\cdot)})$ streng konvex und folglich $-\log(1 + e^{+(\cdot)})$ streng konkav. Damit ist

die Funktion $\mathcal{L}(\beta_0, \beta)$ als Summe von konkaven und streng konkaven Funktionen streng konkav, und zwar bezüglich $\{\beta_0 + \beta^T x^i\}_{i=1}^n$. Im Umkehrschluss ist die negative Log-Likelihood-Funktion $-\mathcal{L}(\beta_0, \beta)$ streng konvex bezüglich $\{\beta_0 + \beta^T x^i\}_{i=1}^n$. Die zu optimierende Funktion f_{LASSO}^{log} in (2.50) ist damit streng konvex bezüglich $\{\beta_0 + \beta^T x^i\}_{i=1}^n$. Die L1-Regularisierung durch den Parameter λ ist äquivalent zur Beschränkung der L1-Norm, $\sum_{j=1}^p |\beta_j| \leq R \in \mathbb{R}^+$, vgl. Anhang, sodass der Raum der möglichen Lösungen auf einen abgeschlossenen Hyperoktaeder eingeschränkt wird. Aus der strengen Konvexität und der Eingrenzung des Definitionsbereiches folgt die Existenz einer Lösung $(\beta_{0,L}, \beta_L) \in \operatorname{argmin} f_{LASSO}^{log}$. Da $-\mathcal{L}(\beta_0, \beta)$ bezüglich $\{\beta_0 + \beta^T x^i\}_{i=1}^n$ und nicht (β_0, β) streng konvex ist, ist die Lösung nicht zwingend eindeutig.

Zur Lösung des Optimierungsproblems (2.50) kann das Coordinate Descent (CD), vgl. Kapitel 2.3.2, zum Einsatz kommen. Allerdings raten Hastie et al. (2016) [21] davon ab, das CD direkt auf die regularisierte logistische Regression (2.50)] anzuwenden, da die jeweils bezüglich einer Variable optimalen Koeffizienten nicht in expliziter Form angegeben werden können, sodass es eines Liniensuch- bzw. eines numerischen Lösungsverfahrens bedarf.

Um dies zu vermeiden, eignet sich das *Newton-Verfahren*, auch unter dem Namen *Newton-Raphson-Verfahren* bekannt [26]. Dazu wird für eine Näherungslösung $(\hat{\beta}_0, \hat{\beta})$ der optimalen Koeffizienten $(\beta_{0,L}, \beta_L)$ mit Hilfe der Taylor-Entwicklung bis zur zweiten Ordnung eine Approximation von $\mathcal{L}(\beta_0, \beta)$ als

$$\mathcal{L}_Q(\beta_0, \beta) := -\frac{1}{2} \sum_{i=1}^{n} w_i (z_i - \beta_0 - \beta^T x^i)^2 + C(\hat{\beta}_0, \hat{\beta}) \ , \tag{2.51}$$

$$w_i = w_i(\hat{\beta}_0, \hat{\beta}) \ = \ \hat{p}(x^i)(1 - \hat{p}(x^i)) \ , \tag{2.52}$$

$$z_i \ = z_i(\hat{\beta}_0, \hat{\beta}) \ = \ \hat{\beta}_0 + \hat{\beta}^T x^i + \frac{y_i - \hat{p}(x^i)}{w_i} \ , \tag{2.53}$$

hergeleitet. Hierbei sind $\hat{p}(x^i)$ eine Approximation von $\Pr(y = 1|x^i)$, $C(\hat{\beta}_0, \hat{\beta})$ eine Konstante, die nur von $\hat{\beta}_0, \hat{\beta}$ und nicht β_0, β abhängig ist, sodass diese bei der Optimierung keine Rolle spielt, z_i *working responce* und w_i Gewichte. $\mathcal{L}_Q(\beta_0, \beta)$ stellt somit ein gewichtetes OLS-Optimierungsproblem dar, wobei sich die Gewichte nach jeder Iteration ändern. Zur Herleitung von $\mathcal{L}_Q(\beta_0, \beta)$ sei auf [19] verwiesen.

Anstatt $-\frac{1}{n}\mathcal{L}(\beta_0, \beta)$ (vgl. das Optimierungsproblem (2.50)) erfolgt der Einsatz von $-\frac{1}{n}\mathcal{L}_Q$ als Verlustfunktion für die regularisierte logistische Regression, wobei die Konstante $C(\hat{\beta}_0, \hat{\beta})$ eliminiert wird. Das zu lösende Optimierungsproblem lautet nun

$$\min_{(\beta_0, \beta) \in \mathbb{R}^{p+1}} \left\{ \ \tilde{f}_{LASSO}^{log} := -\frac{1}{n}\mathcal{L}_Q(\beta_0, \beta) + \lambda \|\beta\|_1 \ \right\}. \tag{2.54}$$

Ist eine Approximation bzw. Startlösung $(\hat{\beta}_0, \hat{\beta})$ vorhanden, so wird $\mathcal{L}_Q$ (2.51) aktualisiert und anschließend das CD zur Lösung des Problems (2.54) mit dem aktuellen $\mathcal{L}_Q$ verwendet.

Das Pathwise CD startet hier ebenso bei einem $\lambda_0 = \lambda_{\max}$, für das die Lösung $\hat{\beta}(\lambda_0) = 0_{|p|}$ ist. Im Kapitel 2.5 wird das $\lambda_{\max}$ als $\lambda_{\max} = \max\limits_{i \in \{1,...,n\}} |x_i^T(y - \overline{p})|$ hergeleitet, mit $\overline{p} = \mathbb{1}\overline{y}$.

Algorithmus 5 Pathwise Coordinate Descent für die L1-regularisierte logistische Regression

1: X, y, $\lambda_0 > \lambda_1 > ... > \lambda_m > 0$, $tol \geq 0$ Fehlertoleranzgrenze für das CD.
2: $(\hat{\beta}_0(\lambda_0), \hat{\beta}(\lambda_0)) = (0, 0_{|p|})$ Startlösung.
3: **for** $l = 1 : m$ **do**
4: Berechne $w_i(\hat{\beta}_0(\lambda_{l-1}), \hat{\beta}(\lambda_{l-1}))$ mittels (2.52) $\forall i \in \{1, ..., n\}$.
5: Berechne $z_i(\hat{\beta}_0(\lambda_{l-1}), \hat{\beta}(\lambda_{l-1}))$ mittels (2.53) $\forall i \in \{1, ..., n\}$.
6: Aktualisiere $\mathcal{L}_Q$ (2.51) mit $(\hat{\beta}_0(\lambda_{l-1}), \hat{\beta}(\lambda_{l-1}))$.
7: Berechne $(\hat{\beta}_0(\lambda_l), \hat{\beta}(\lambda_l))$ für (2.54) mittels CD (Algorithmus 2).
8: **end for**

Es muss eine Anpassung des Soft-Thresholding Operators (2.36) im CD (Algorithmus 2) für den Koeffizienten $\hat{\beta}_j$, $j \in \{1, ..., p\}$, erfolgen und der Achsenabschnitt $\hat{\beta}_0$ berücksichtigt werden. Der Soft-Thresholding Operator lässt sich analog herleiten, indem das Differential der Optimierungsfunktion $\tilde{f}^{log}_{LASSO}$ in (2.54) nach den entsprechenden Koeffizienten gebildet wird, und wird hier zur Vollständigkeit angegeben. Schreibe $\mathcal{L}_Q(\beta_0, \beta)$ zunächst als

$$\mathcal{L}_Q(\beta_0, \beta) = -\frac{1}{2} \sum_{i=1}^{n} \; w_i \Big(z_i - \beta_0 - x_{ij}\beta_j - \sum_{k \neq j} x_{ik}\beta_k \Big)^2 \; .$$

Für den Koeffizienten $\hat{\beta}_j$, $j \in \{1, ..., p\}$, gilt

$$\begin{aligned}
&\frac{\partial \tilde{f}^{log}_{LASSO}}{\partial \beta_j}(\hat{\beta}_j) = -\frac{1}{n} \sum_{i=1}^{n} w_i \Big(z_i - \beta_0 - x_{ij}\hat{\beta}_j - \sum_{k \neq j} x_{ik}\beta_k \Big) x_{ij} + \lambda s_j = 0 \\
\iff \; & \frac{1}{n} \sum_{i=1}^{n} w_i \Big(z_i - \beta_0 - \sum_{k \neq j} x_{ik}\beta_k \Big) - \hat{\beta}_j \frac{1}{n} \sum_{i=1}^{n} w_i x_{ij}^2 = \lambda s_j \\
\iff \; & \hat{\beta}_j = \frac{\frac{1}{n} \sum_{i=1}^{n} w_i \Big(z_i - \beta_0 - \sum_{k \neq j} x_{ik}\beta_k \Big) - \lambda s_j}{\frac{1}{n} \sum_{i=1}^{n} w_i x_{ij}^2} \; .
\end{aligned}$$

Der Soft-Thresholding Operator ist somit gegeben als

$$\hat{\beta}_j = \frac{S\Big(\frac{1}{n} \sum_{i=1}^{n} w_i \Big(z_i - \beta_0 - \sum_{k \neq j} x_{ik}\beta_k \Big), \lambda \Big)}{\frac{1}{n} \sum_{i=1}^{n} w_i x_{ij}^2} \; . \tag{2.55}$$

Dementsprechend ist im Schritt 7 des Algorithmus 5 die Gleichung (2.55) statt der Gleichung (2.36) im Schritt 6 des Algorithmus 2 zu verwenden.
Das Bilden des Differentials von $\tilde{f}^{log}_{LASSO}$ nach β_0 liefert für den optimalen Achsenabschnitt $\hat{\beta}_0$

$$-\frac{1}{n}\sum_{i=1}^{n} w_i(z_i - \hat{\beta}_0 - \beta^T x^i) = 0 \quad \Longleftrightarrow \quad \hat{\beta}_0 = \frac{\sum_{i=1}^{n} w_i(z_i - \beta^T x^i)}{\sum_{i=1}^{n} w_i} \ .$$

Bemerkung: Im maschinellen Lernen ist es üblich, $y \in \{-1, +1\}$ zu wählen statt $y \in \{0, 1\}$ [21]. Folgend aus der Definition für die Log-Likelihood-Funktion (2.49) hat die Log-Likelihood-Funktion die Form

$$\mathcal{L}(\beta_0, \beta) = \ -\sum_{i=1}^{n} \ \log\Big(1 + e^{-y_i(\beta_0 + \beta^T x^i)}\Big) \ .$$

Hierbei ist $y(\beta_0 + \beta^T x)$ der sogenannte *margin*, der bei einer korrekt klassifizierten Beobachtung positiv und sonst negativ ist. Die Funktion $\tilde{\mathcal{L}}(\beta_0, \beta)$ ist eine bezüglich des margin's (streng) monoton fallende Funktion.

Multinomiale Regression O.b.d.A. sei $y \in \{1, ..., K\}$. Es ist gängig, die multinomiale Regression als Erweiterung der logistischen Regression auf mehr als zwei Klassen aufzufassen, sodass es sich in diesem Fall um die multinomiale logistische Regression handelt [19, 15]. Nach diesem Ansatz werden $K - 1$ Logit's aufgestellt:

$$\log\left(\frac{\Pr(y = k|x)}{\Pr(y = K|x)}\right) = \beta_0^k + (\beta^k)^T x \quad \forall k \in \{1, ..., K-1\} \ . \tag{2.56}$$

Jede Variable $j \in \{1, ..., p\}$ hat folglich $K - 1$ Koeffizienten $\beta_j^1, ..., \beta_j^{K-1}$. Der Logit (2.56) ist äquivalent zu

$$\begin{aligned} \Pr(y = k|x) &= \frac{e^{\beta_0^k + (\beta^k)^T x}}{1 + \sum_{k=1}^{K-1} e^{\beta_0^k + (\beta^k)^T x}} \quad \forall k \in \{1, ..., K-1\} \ , \\ \Pr(y = K|x) &= \ \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_0^k + (\beta^k)^T x}} \quad . \end{aligned}$$

Die entscheidende Rolle spielen wie bei der logistischen Regression die bedingten Wahrscheinlichkeiten der Klassen $1, ..., K$, wobei deren Summe Eins beträgt. Hier wird die Klasse K zum Nenner gewählt, die Wahl des Nenners ist dabei beliebig, weil die zu schätzenden Koeffizienten äquivariant bezüglich dieser sind [19]. Die multinomiale Regression soll hier allerdings in Verbindung mit Regularisierung gelöst werden. Eine schlichte Erweiterung der regularisierten logistischen Regression auf

die regularisierte multinomiale logistische Regression kann hierfür keine Anwendung finden, weil die daraus hervorgehende Lösung nicht äquivariant bezüglich der Wahl des Nenners ist. Daher schlagen einige Autoren stattdessen einen symmetrischen Ansatz für die bedingten Wahrscheinlichkeiten vor,

$$\Pr(y = k|x) = \frac{e^{\beta_0^k + (\beta^k)^T x}}{\sum_{k=1}^K e^{\beta_0^k + (\beta^k)^T x}} := p_k \quad , \tag{2.57}$$

die sich ebenso zu einer Eins aufsummieren, $\sum_1^k p_k = 1$ [15, 21]. Unter dieser Setzung sind nun für jede Variable $j \in \{1, ..., p\}$ jeweils K Koeffizienten zu bestimmen bzw. zu approximieren. Die Log-Likelihood-Funktion lautet

$$\mathcal{L}\left(\{\beta_0^k, \beta^k\}_{k=1}^K\right) = \sum_{i=1}^n \log\Big(\Pr(y = y_i \mid x^i;\ \{\beta_0^k, \beta^k\}_{k=1}^K)\Big) = \frac{1}{n}\sum_{i=1}^n \log(p_{y_i}(x^i)) \ , \tag{2.58}$$

wobei $p_k(x^i) := \Pr(y = k|x^i)$. Mit der bedingten Wahrscheinlichkeit p_k (2.57), Rechenregeln für den Logarithmus und der Definition der sogenannten *Indikatormatrix* $Y \in \mathbb{R}^{n\times K}$ mit den Einträgen $y_{ik} = \mathbb{I}(y_i = k)$ lässt sich die Log-Likelihood-Funktion (2.58) umschreiben als

$$\mathcal{L}\left(\{\beta_0^k, \beta^k\}_{k=1}^K\right) = \sum_{i=1}^n \left[\sum_{k=1}^K y_{ik}(\beta_0^k + (\beta^k)^T x^i) - \log\left(\sum_{k=1}^K e^{\beta_0^k + (\beta^k)^T x^i}\right)\right] \ .$$

Das LASSO für die multinomiale Regression ist nun gegeben als

$$\min_{\{\beta_0^k, \beta^k\}_{k=1}^K \in \mathbb{R}^{K\times(p+1)}} \left\{ f_{LASSO}^{mult} := -\frac{1}{n}\mathcal{L}\left(\{\beta_0^k, \beta^k\}_{k=1}^K\right) + \lambda \sum_{k=1}^K \|\beta^k\|_1 \right\} \ . \tag{2.59}$$

Mit den gleichen Argumenten wie bei der logistischen Regression ist f_{LASSO}^{mult} eine streng konvexe Funktion, und zwar bezüglich $\left\{\{\beta_0^k + (\beta^k)^T x^i\}_{i=1}^n\right\}_{k=1}^K$. Da die L1-Regularisierung den Definitionsbereich auf einen Hyperoktaeder einschränkt, existiert ein Lösungsset $\{\beta_{0,L}^k, \beta_L^k\}_{k=1}^K \in \operatorname{argmin} f_{LASSO}^{mult}$.

Bei Betrachtung der bedingten Wahrscheinlichkeit p_k (2.57) fällt auf, dass ohne weitere Vorgaben oder Nebenbedingungen es nicht möglich ist, die Parameter zu schätzen, denn die Addition bzw. Subtraktion einer beliebigen Funktion $\gamma_0 + \gamma^T x$ zu bzw. von $\beta_0^k + (\beta^k)^T x \ \forall k \in \{1, ..., K\}$ liefert dieselbe bedingte Wahrscheinlichkeit:

$$\frac{e^{\beta_0^k + (\beta^k)^T x \pm (\gamma_0 + c^T x)}}{\sum_{k=1}^K e^{\beta_0^k + (\beta^k)^T x \pm (c_0 + c^T x)}} = \frac{e^{\beta_0^k + (\beta^k)^T x} e^{\pm(\gamma_0 + \gamma^T x)}}{\sum_{k=1}^K e^{\beta_0^k + (\beta^k)^T x} e^{\pm(\gamma_0 + \gamma^T x)}} = \frac{e^{\beta_0^k + (\beta^k)^T x}}{\sum_{k=1}^K e^{\beta_0^k + (\beta^k)^T x}} = p_k \ .$$

Folglich gilt $\mathcal{L}\left(\{\beta_0^k, \beta^k\}_{k=1}^K\right) = \mathcal{L}\left(\{\beta_0^k \pm \gamma_0, \beta^k \pm \gamma\}_{k=1}^K\right)$. Mit Hilfe von Regularisierung lässt sich dieses Problem für γ beheben, da die Addition eines solchen Terms

Einfluss auf die L1-Norm und somit auf den Regularisierungsterm hat. Die Konstante γ_0 bleibt dabei beliebig, weil der Achsenabschnitt β_0 bei dem Regularisierungsterm keine Rolle spielt.

Sei ein Lösungsset $\{\beta_{0,L}^k, \beta_L^k\}_{k=1}^K$ mit $l := \sum_{k=1}^K \|\beta_L^k\|_1$ gegeben. Der Vektor γ wird nun so bestimmt, dass die L1-Norm von $\{\beta_L^k\}_{k=1}^K \pm \gamma$ kleiner als l ist. Für das gesuchte γ muss demnach

$$\gamma = \underset{c\ \in\mathbb{R}^{\text{l}}}{\operatorname{argmin}}\ \lambda \sum_{k=1}^{K} \|\beta_L^k - c\| \quad \Longleftrightarrow \quad \gamma_j = \underset{c_j \in \mathbb{R}}{\operatorname{argmin}}\ \{\ f(c_j) := \lambda \sum_{k=1}^{K} |\beta_{j,L}^k - c_j|\ \}$$

gelten. Bilden des Subdifferentials von $f(c_j)$ und setzen auf Null als notwendige Bedingung für Extrema ergibt für das Optimum γ

$$\sum_{k=1}^{K} -\lambda(s_j^k) = 0 \quad \Longleftrightarrow \quad \sum_{k=1}^{K} s_j^k = 0, \qquad s_j^k = \begin{cases} \text{sign}(\beta_{j,L}^k - \gamma_j), & \beta_{j,L}^k \neq \gamma_j \\ [-1, 1], & \beta_{j,L}^k = \gamma_j\ . \end{cases}$$

Falls $\beta_{j,L}^k = \gamma_j$ gilt, beläuft sich die Summe auf Null. Betrachte nun diejenigen k, für die $\beta_{j,L}^k \neq \gamma_j$ gilt. Die Vorzeichen summieren sich zu einer Null genau dann, wenn γ_j dem Median von $\{\beta_{j,L}^1, ..., \beta_{j,L}^K\}$ entspricht, sodass die Hälfte der Vorzeichen bei $\beta_{j,L}^k < \gamma_j$ den Wert -1 und die andere Hälfte bei $\beta_{j,L}^k > \gamma_j$ den Wert 1 annehmen [15, 21].

Um das Pathwise CD auf das Problem (2.59) anzuwenden, soll analog zur logistischen Regression statt der tatsächlichen Log-Likelihood-Funktion (2.58) eine Approximation dieser ähnlich zu (2.51) eingesetzt werden. Dies würde die Komplexität des Problems stark erhöhen, insbesondere da alle Größen, die vorher einen Vektor darstellten, nun Matrizen repräsentieren würden, wie bspw. die Gewichte. Um dies zu umgehen, wird hier nur eine partielle Approximation je Klasse vorgenommen, sodass pro Iteration bezüglich der Koeffizienten nur einer der Klassen optimiert wird, während alle anderen Koeffizienten bei deren aktuellem Wert fest sind. So entsteht für jede Klasse $k\ \in \{1, ..., K\}$ das L1-regularisierte gewichtete Optimierungsproblem

$$\min_{(\beta_0^k, \beta^k) \in \mathbb{R}^{p+1}} -\frac{1}{n} \mathcal{L}_Q^k(\beta_0^k, \beta^k) + \lambda \|\beta^k\|_1\ , \tag{2.60}$$

das mittels des CD gelöst wird, wobei

$$\mathcal{L}_Q^k(\beta_0^k, \beta^k) := -\frac{1}{2}\sum_{i=1}^{n} w_{ik}(z_{ik} - \beta_0^k - (\beta^k)^T x^i)^2 + C\Big(\{\hat{\beta}_0^k, \hat{\beta}^k\}_{k=1}^K\Big) \ , \tag{2.61}$$

$$w_{ik} = w_{ik}(\hat{\beta}_0^k, \hat{\beta}^k) \ = \hat{p}_k(x^i)(1 - \hat{p}_k(x^i)) \ , \tag{2.62}$$

$$z_{ik} \ = z_{ik}(\hat{\beta}_0^k, \hat{\beta}^k) \ = \hat{\beta}_0^k + (\hat{\beta}^k)^T x^i + \frac{y_{ik} - \hat{p}(x^i)}{w_{ik}} \ . \tag{2.63}$$

Dabei ist $\{\hat{\beta}_0^k, \hat{\beta}^k\}_{k=1}^K$ eine Approximation des optimalen Lösungssets, $C\Big(\{\hat{\beta}_0^k, \hat{\beta}^k\}_{k=1}^K\Big)$ ist eine von $\{\beta_0^k, \beta^k\}_{k=1}^K$ unabhängige Konstante und kann daher aus der Zielfunktion $\mathcal{L}_Q^k(\beta_0^k, \beta^k)$ entfernt werden.

Algorithmus 6 Pathwise Coordinate Descent für die L1-regularisierte multinomiale Regression

1: X, y, $\lambda_0 > \lambda_1 > ... > \lambda_m > 0$, $\{tol_k\}_{k=1}^K$, $tol_k \geq 0 \ \ \forall k \in \{1, .., K\}$, Fehlertoleranzgrenzen für die Klassen 1 bis K für das CD.
2: $\{\hat{\beta}_0^k(\lambda_0), \hat{\beta}^k(\lambda_0)\}_{k=1}^K = \{(0, 0_{|p|})\}_{k=1}^K$ Set von Startlösungen.
3: **for** $l = 1 : m$ **do**
4: **for** k=1:K **do**
5: Berechne $w_{ik}(\hat{\beta}_0^k(\lambda_{l-1}), \hat{\beta}^k(\lambda_{l-1}))$ mittels (2.62) $\ \forall i \in \{1, ..., n\}$.
6: Berechne $z_{ik}(\hat{\beta}_0^k(\lambda_{l-1}), \hat{\beta}^k(\lambda_{l-1}))$ mittels (2.63) $\ \forall i \in \{1, ..., n\}$.
7: Aktualisiere $\mathcal{L}_Q^k$ (2.61) mit $\{(\hat{\beta}_0^k(\lambda_{l-1}), \hat{\beta}^k(\lambda_{l-1}))\}_{k=1}^K$.
8: Berechne $(\hat{\beta}_0^k(\lambda_l), \hat{\beta}^k(\lambda_l))$ für (2.60) mittels CD.
9: **end for**
10: **end for**

Analog zum Pathwise CD für klassische lineare Regression benutzen die Algorithmen 5 und 6 Warm Starts, wodurch die $\{\mathcal{L}_Q^k(\beta_0^k, \beta^k)\}_{k=1}^K$ die eigentlichen Log-Likelihood-Funktionen recht gut approximieren und das CD (Schritt 7 im Algorithmus 5, Schritt 8 im Algorithmus 6) schnell konvergiert. Zur Beschleunigung der Konvergenz werden bei der Anwendung des CD in den Algorithmen 5 und 6 nur die zum Modell gehörenden Variablen herangezogen, vgl. Kapitel 2.3.2.

2.4.2 Cox-Regression

Cox-Regression (auch Cox-Hazard-Regression) findet in der Überlebenszeitanalyse Anwendung und geht auf Cox (1972) [10] zurück. Es werde eine medizinische Studie über einen bestimmten Zeitraum durchgeführt, in der bei einer Erkrankung die Überlebenszeit in Verbindung mit einem Ereignis im Vordergrund steht. Da die Studie einen festen Zeitraum umfasst, leben einige Patienten zum Ende der Studie,

nehmen an der Studie nicht mehr teil oder sind an einer anderen Ursache gestorben. Somit ist für diese Patienten zum Ende der Studie das Ereignis, das als der Tod aufgrund der Krankheit definiert ist, nicht eingetreten, sodass die Daten rechtszensiert (engl. right censored) sind.

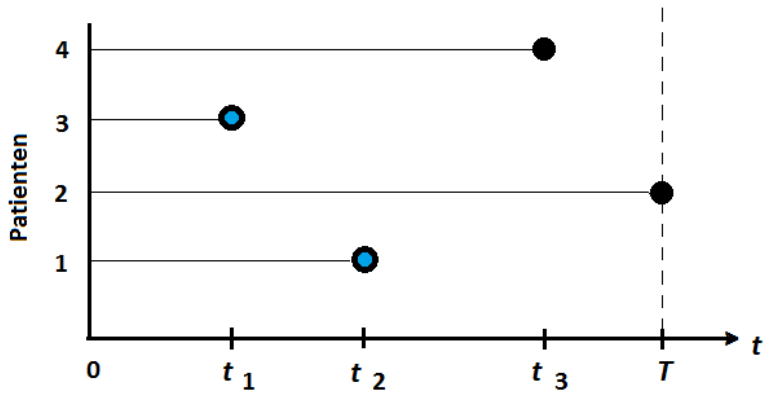


Abbildung 6 Censoring Time. Eine Studie über einen Zeitraum von T Zeiteinheiten mit vier Patienten. Die Patienten 1 und 3 sterben vor dem Ende der Studie an der Krankheit, Patient 2 lebt noch zum Ende der Studie, Patient 4 nimmt an der Studie ab dem Zeitpunkt t_3 nicht mehr teil. Die Überlebenszeiten der Patienten 2 und 4 sind somit rechtszensiert.

Formell liegen zum Ende der Studie n Beobachtungen jeweils als das Tripel (x^i, y_i, δ_i), $i = 1, ..., n$, vor, wobei $\delta_i \in \{0, 1\}$ binär ist, $y_i = \min(t_i, T)$, sodass $y_i = t_i$ der Überlebenszeit (Ausfallzeit, engl. failure time) bei $\delta_i = 1$ (Ereignis ist eingetroffen) und bei $\delta_i = 0$ (Ereignis ist nicht eingetroffen) entweder der Zeitspanne T der Studie oder der Zeit, zu dem der Patient aus anderen Gründen kein Bestandteil der Studie mehr ist (engl. Censoring Time), entspricht [10, 33].

Bei der Cox-Regression steht der sogenannte *Hazard* (auch *Hazardrate*) im Mittelpunkt. Der Hazard des bis zum Zeitpunkt t lebenden Patienten i, bezeichnet als $h_i(t)$, ist die Wahrscheinlichkeit des Ausfalls (des Eintritts des Ereignisses) zum Zeitpunkt t. Die Cox-Regression unterstellt die folgende Form für $h_i(t)$ [10, 33]:

$$h_i(t) := h(t; x^i) = h_0(t)e^{\beta^T x^i} \ . \tag{2.64}$$

$h_0(t)$ ist eine beliebige und unbekannte Funktion und bezeichnet den sogenannten *Baseline-Hazard*. Dieser entspricht dem Hazard bei Beobachtungen mit $x = 0_{|p|}$ zum Zeitpunkt t, sodass diese Einflussgröße sich nicht auf die Ausfallzeit auswirkt.

Die Ausfallzeiten werden in aufsteigender Reihenfolge als $t_1 < ... < t_i < ... < t_q$, $q \leq n$, geordnet. t_1 entspricht somit dem Zeitpunkt, zu dem der erste Patient, t_q zu dem der letzte Patient gestorben ist. O.b.d.A. bestehe die Annahme, dass es keine identischen Überlebenszeiten gibt, $y_i \neq y_k \ \forall i \neq k$, $i, k \in \{1, ..., q\}$, sodass $q = n$ gilt. Die Lösungsalgorithmen lassen sich leicht bei Vorkommen identischer

Überlebenszeiten modifizieren, wobei sich dies kaum auf den Aufwand auswirkt, siehe dazu [33]. $j(i)$ sei der Index desjenigen Patienten, für den zum Zeitpunkt t_i das Ereignis stattgefunden hat. R_i sei die Menge der Patienten, die zum Zeitpunkt t_i noch leben, deren Überlebenszeit folglich größer als t_i ist. R_i wird in der Literatur als *Risk Set* bezeichnet.

Der als (2.64) definierte Hazard lässt jegliche Informationen zwischen den Ausfallzeiten außer Acht, da $h_0(t) = 0$ für $t \neq t_i \ \forall i = 1, ..., q$ möglich ist [10]. Somit können Informationen nur derjenigen Patienten i für die Modellierung des Hazards verwendet werden, für die der Todesfall eingetreten ist, $\delta_i = 1$ und $y_i = t_i$. Dementsprechend können Rückschlüsse bezüglich β nur in den Zeitpunkten $t_1, ..., t_q$ gezogen werden, sodass die Zeit diskret ist. Um eine Analyse für jedes beliebige $h_0(t)$ zu ermöglichen, wird folgende bedingte Wahrscheinlichkeit definiert [10]:

$$p_{j(i)}(\beta) := \frac{e^{\beta^T x^{j(i)}}}{\sum_{j \in R_i} e^{\beta^T x^j}} \ . \tag{2.65}$$

$p_{j(i)}(\beta)$ drückt die bedingte Wahrscheinlichkeit dafür aus, dass gerade der Patient $j(i)$ zum Zeitpunkt t_i unter allen noch lebenden Patienten stirbt. Die bezüglich β zu maximierende Partial Likelihood

$$l(\beta) = \prod_{i=1}^{q} \frac{e^{\beta^T x^{j(i)}}}{\sum_{j \in R_i} e^{\beta^T x^j}} \tag{2.66}$$

ergibt sich als Produkt der bedingten Wahrscheinlichkeiten (2.65). Dieser Ansatz hat zur Folge, dass der Baseline-Hazard keine Rolle spielt und daher nicht geschätzt werden muss. Die folgenden Aussagen gehen auf [33] zurück.

Die Maximierung der Likelihood (2.66) ist äquivalent zur Maximierung der Log-Likelihood-Funktion

$$\mathcal{L}(\beta) := \log(l(\beta)) = \sum_{i=1}^{q} \log\Big(\frac{e^{\beta^T x^{j(i)}}}{\sum_{j \in R_i} e^{\beta^T x^j}} \Big) = \sum_{i=1}^{q} x_{j(i)}^T \beta - \log\Big(\sum_{j \in R_i} e^{\beta^T x^j} \Big) \ . \tag{2.67}$$

Gleichfalls wie bei der logistischen und multinomialen Regression besteht im Falle $p > n$ das Problem, dass die Einträge des Regressionsvektors bei der Maximierung von $l(\beta)$ bzw. $\mathcal{L}(\beta)$ gegen $\pm\infty$ streben, um den Wert Eins für $l(\beta))$ bzw. Null für $\mathcal{L}(\beta)$ zu erreichen. Dieses Problem lässt sich unter anderem mit Hilfe der L1-Regularisierung beseitigen. Das LASSO für die Cox-Regression lautet:

$$\min_{\beta \in \mathbb{R}^p} \ \Big\{ \ f_{LASSO}^{cox} := -\frac{1}{n} \mathcal{L}(\beta) + \lambda \|\beta\|_1 \Big\} \ . \tag{2.68}$$

Die Zielfunktion $\mathcal{L}(\beta)$ von f_{LASSO}^{cox} ist eine bezüglich $\{\beta^T x^i\}_{i=1}^n$ streng konvexe Funk-

tion, die L1-Regularisierung schränkt den Definitionsbereich ein, woraus die Existenz mindestens einer Lösung $\beta_L \in \text{argmin } f^{cox}_{LASSO}$ folgt. Der Parameter λ sollte dabei hinreichend groß sein, denn für $\lambda \to 0$ gilt $\beta_j \to \pm\infty \; \forall j = 1, ..., p$.

Wie bisher erfolgt zum Lösen des Problems (2.68) der Einsatz von CD. Ebenso wie bei der logistischen und multinomialen Regression wird dabei nicht die Log-Likelihood-Funktion $\mathcal{L}$ sondern eine Approximation dieser herangezogen. Mittels der Taylor-Entwicklung von $\mathcal{L}$ in $\tilde{\beta}$ bis zur zweiten Ordnung lässt sich $\mathcal{L}(\beta)$ approximieren als

$$\begin{aligned}\mathcal{L}(\beta) &\approx \mathcal{L}(\tilde{\beta}) + \nabla_\beta \mathcal{L}(\tilde{\beta})(\beta - \tilde{\beta}) + \frac{1}{2}(\beta - \tilde{\beta})^T H_{\mathcal{L}}(\tilde{\beta})(\beta - \tilde{\beta}) \\ &= \mathcal{L}(\tilde{\beta}) + \nabla_{X\beta} \mathcal{L}(\tilde{y})(X\beta - \tilde{y}) + \frac{1}{2}(X\beta - \tilde{y})^T H_{\mathcal{L}}(\tilde{y})(X\beta - \tilde{y}) \; . \end{aligned} \tag{2.69}$$

Dabei ist $\tilde{\beta}$ eine Näherungslösung für das Problem (2.68), $\tilde{y} := X\tilde{\beta}$, $H_{\mathcal{L}}(\tilde{\beta}) \in \mathbb{R}^{p \times p}$ und $H_{\mathcal{L}}(\tilde{y}) \in \mathbb{R}^{n \times n}$ sind die Hessematrizen von $\mathcal{L}$ bezüglich β ausgewertet in $\tilde{\beta}$ bzw. $\tilde{y}$. Ausmultiplizieren von (2.69) und anschließendes Umformen ergibt

$$\mathcal{L}(\beta) \approx -\frac{1}{2}(z(\tilde{y}) - X\beta)^T H_{\mathcal{L}}(\tilde{y})(z(\tilde{y}) - X\beta) + C(\tilde{\beta}, \tilde{y}) \; ,$$

wobei $z(\tilde{y}) := \tilde{y} - H_{\mathcal{L}}(\tilde{y})^{-1} \nabla_{X\beta} \mathcal{L}(\tilde{y})$. Die Konstante $C(\tilde{\beta}, \tilde{y})$ ist unabhängig von β und ist für die Optimierung daher irrelevant, unter anderem fällt diese bei der Bildung von Differentialen bezüglich β oder β_j, $j \in \{1, ..., p\}$, weg.

Der Nachteil dieser Approximation besteht in der Notwendigkeit der Berechnung der vollbesetzten Hessematrix $H_{\mathcal{L}}(\tilde{y})$ mit $O(n^2)$ Einträgen und anschließender Bestimmung derer Inversen. Um dies zu umgehen, ersetzen die Autoren die Hessematrix durch die Diagonalmatrix

$$W(\tilde{y}) := \text{diag}(w(\tilde{y})_1, ..., w(\tilde{y})_n) = \text{diag}(H_{\mathcal{L}}(\tilde{y})) \; ,$$

deren Inverse als $W^{-1} = \text{diag}(w(\tilde{y})_1^{-1}, ..., w(\tilde{y})_n^{-1})$ gegeben ist. Hastie et al. (1990) [18] haben gezeigt, dass sich der gesuchte Regressionsvektor dadurch nicht ändert, unter anderem weil die Diagonaleinträge der Hessematrix im Vergleich zu den restlichen Einträgen betragsmäßig klein sind. Die Größe $z(\tilde{y})$ wird nun berechnet als

$$z(\tilde{y}) := \tilde{y} - W^{-1} \nabla_{X\beta} \mathcal{L}(\tilde{y}) \; .$$

Die Approximation von $\mathcal{L}(\beta)$ ist gegeben durch:

$$\mathcal{L}_Q(\beta) := -\frac{1}{2}\sum_{i=1}^{n} w(\tilde{y})_i (z(\tilde{y})_i - \beta^T x^i)^2 + C(\tilde{\beta}, \tilde{y}) \ , \tag{2.70}$$

$$w(\tilde{y})_i = H_{\mathcal{L}}(\tilde{y})_{ii} = \sum_{k \in C_i} \Big(\frac{e^{\tilde{y}_i} \sum_{j \in R_i} e^{\tilde{y}_j} - (e^{\tilde{y}_i})^2}{(\sum_{j \in R_k} e^{\tilde{y}_j})^2} \Big) \ , \tag{2.71}$$

$$z_(\tilde{y}_i) \ = \tilde{y}_i + \frac{1}{w(\tilde{y})_i} \left(\delta_i - \sum_{k \in C_i} \Big(\frac{e^{\tilde{y}_i}}{\sum_{j \in R_k} e^{\tilde{y}_j}} \Big) \right) . \tag{2.72}$$

C_i bezeichnet die Menge der Patienten k mit $t_k < y_i$.

Das LASSO für die Cox-Regression mit der approximierten Zielfunktion $\mathcal{L}_Q(\beta)$ lautet nun

$$\min_{\beta \in \mathbb{R}^p} \ \Big\{ \ \tilde{f}^{cox}_{LASSO} := -\frac{1}{n} \mathcal{L}_Q(\beta) + \lambda \|\beta\|_1 \ \Big\} \ . \tag{2.73}$$

Dies ist ein L1-regularisiertes gewichtetes Optimierungsproblem, das mittels CD gelöst wird. In der inneren Schleife des CD erfolgt in der Iteration j, $j \in \{1, ..., p\}$, die Optimierung bezüglich β_j. Die Optimalitätsbedingung ergibt

$$\begin{aligned} & \frac{\partial \tilde{f}^{cox}_{LASSO}}{\partial \beta_j}(\hat{\beta}_j) = -\frac{1}{n} \sum_{i=1}^{n} w(\tilde{y})_i \Big(z(\tilde{y})_i - x_{ij}\hat{\beta}_j - \sum_{k \neq j} x_{ik}\beta_k \Big) x_{ij} + \lambda s_j = 0 \\ \iff \ & \hat{\beta}_j = \frac{\frac{1}{n}\sum_{i=1}^{n} w(\tilde{y})_i \Big(z(\tilde{y})_i - \sum_{k \neq j} x_{ik}\beta_k \Big) - \lambda s_j}{\frac{1}{n}\sum_{i=1}^{n} w(\tilde{y})_i x_{ij}^2} \ , \end{aligned}$$

wobei $s_j = \text{sign}(\hat{\beta}_j)$ für $\hat{\beta}_j \neq 0$, $s_j \in [-1, 1]$ sonst. Das optimale $\hat{\beta}_j$ ist somit durch den Soft-Thresholding Operator $S(x, \lambda) = \text{sign}(x)(x - |\lambda|)_+$ (2.29) mit $x := \frac{1}{n}\sum_{i=1}^{n} w(\tilde{y})_i \Big(z(\tilde{y})_i - \sum_{k \neq j} x_{ik}\beta_k \Big)$ gegeben als

$$\hat{\beta}_j = \frac{S\Big(\frac{1}{n}\sum_{i=1}^{n} w(\tilde{y})_i \Big(z(\tilde{y})_i - \sum_{k \neq j} x_{ik}\beta_k \Big), \lambda \Big)}{\frac{1}{n}\sum_{i=1}^{n} w(\tilde{y})_i x_{ij}^2} \ . \tag{2.74}$$

Wie bislang soll das LASSO, hier das Problem (2.73), mittels des Pathwise CD für eine Sequenz von RP $\lambda_0 > ... > \lambda_m \geq 0$ gelöst werden. Um das $\lambda_0 = \lambda_{\max}$ zu bestimmen, nehme an, dass $\hat{\beta} = 0_{|p|}$ die optimale Lösung zu einem λ_0 ist. Einsetzen in die Gleichung (2.74) ergibt $\frac{1}{n}\sum_{i=1}^{n} w(0)_i x_{ij} z(0)_i < \lambda$, sodass das $\lambda_{\max}$ bestimmt werden kann als

$$\lambda_{\max} = \max_{j \in \{1, ..., p\}} \frac{1}{n} \sum_{i=1}^{n} w(0)_i x_{ij} z(0)_i \ .$$

Das λ_m wird als $\lambda_m = \lambda_{\min} = \delta \lambda_{\max}$, $\delta > 0$ gewählt. Bei $n \geq p$ setzen die Autoren das δ auf $\delta = 0.0001$, bei $n < p$ auf $\delta = 0.05$. Alle Größen sind Funktionen von λ, insbesondere $\hat{\beta} = \hat{\beta}(\lambda)$ und $\hat{y} = \hat{y}(\lambda) = X\hat{\beta}(\lambda)$. Der Algorithmus verwendet

gleichfalls Warm Starts.

Algorithmus 7 Pathwise Coordinate Descent für die L1-regularisierte Cox-Regression

1: X, y, $\lambda_0 > \lambda_1 > ... > \lambda_m \geq 0$, $tol \geq 0$ Fehlertoleranzgrenze für das CD.
2: $\hat{\beta}(\lambda_0) = \tilde{\beta}(\lambda_0) = 0_{|p|}$, $\hat{y}(\lambda_0) = X\hat{\beta}(\lambda_0) = X\tilde{\beta}(\lambda_0) = \tilde{y}(\lambda_0) = 0_{|p|}$ Startlösung.
3: **for** $l = 1 : m$ **do**
4: Berechne $w(\tilde{y}(\lambda_{l-1}))_i$ mittels (2.71) $\forall i \in \{1, ..., n\}$.
5: Berechne $z(\tilde{y}(\lambda_{l-1}))_i$ mittels (2.72) $\forall i \in \{1, ..., n\}$.
6: Setze $w(\tilde{y}(\lambda_{l-1}))_i$, $z(\tilde{y}(\lambda_{l-1}))_i$, $\tilde{\beta}(\lambda_{l-1})$ in $\mathcal{L}_Q$ (2.70) ein.
7: Berechne $\hat{\beta}(\lambda_l)$ für das Problem (2.73) mittels CD (Algorithmus 2) unter Verwendung der Gleichung (2.74) im Schritt 6 des Algorithmus 2.
8: Setze $\tilde{\beta}(\lambda_l) = \hat{\beta}(\lambda_l)$, $\tilde{y}(\lambda_l) = \hat{y}(\lambda_l)$.
9: **end for**

Eine Schwierigkeit des Algorithmus besteht bei der Berechnung von $w(\tilde{y})_i$ (2.71) und $z(\tilde{y})_i$ (2.72), $i = 1, ..., n$, da diese Größen einer Aktualisierung des Risk Set R_i benötigen. Dazu muss die Berechnung von $\sum_{j \in R_k} e^{\tilde{y}_j}$ für alle k in C_i erfolgen. Die Anzahl der Elemente C_i und R_k beträgt $O(n)$, sodass sich der dafür benötigte Aufwand zunächst auf $O(n^2)$ beläuft. Es gilt allerdings

$$\sum_{j \in R_{k+1}} e^{\tilde{y}_j} = \sum_{j \in R_k} e^{\tilde{y}_j} - \sum_{j \in R_k \ \& \ j \notin R_{k+1}} e^{\tilde{y}_j} \ .$$

Folglich ist die Berechnung der vollständigen Summe $\sum_{j \in R_k} e^{\tilde{y}_j}$ nur für den ersten Index k in der Menge C_i notwendig, für jeden weiteren Index werden die bereits berechneten Beiträge der Patienten zwischen den Zeitpunkten $k-1$ und k abgezogen. Auf diese Weise reduziert sich der Aufwand auf $O(n)$ Operationen.

Residuum bei GLM In der klassischen linearen Regression ist die Norm des Residuums ein Maß dafür, wie gut sich die Zielvariable durch die Einflussgrößen erklären lässt. Bei einem GLM wie der logistischen, multinomialen und Cox-Regression ist die Devianz (engl. deviance) eine zentrale Kenngröße für die Bewertung der Güte der Anpassung. Im klassischen linearen Modell stimmt diese mit der Norm des Residuums überein. Die Devianz in Abhängigkeit eines Regressionsvektors β ist definiert als

$$D(\beta) = 2\log\left(\frac{\mathcal{L}_{sat}}{\mathcal{L}(\beta)}\right) = 2(\mathcal{L}_{sat} - \mathcal{L}(\beta)) \ .$$

$\mathcal{L}_{sat}$ ist die maximale Log-Likelihood, $\mathcal{L}(\beta)$ ist die Log-Likelihood in β. Die Log-Likelihood erreicht ihr Maximum bei dem gesättigten (engl. saturated) Modell, das alle Variablen enthält. Die maximale Likelihood (2.66) beträgt eins, sodass für die

Log-Likelihood $\mathcal{L}_{sat} = 0$ gilt, vgl. (2.67). Die Null-Devianz

$$D_{null} = D(0) = 2(\mathcal{L}_{sat} - \mathcal{L}_{null}) \ ,$$

wobei $\mathcal{L}_{null} = \mathcal{L}(0_{|p|})$, ist die Devianz des Null-Modells, das keine Merkmale einbezieht und damit entweder Null (Cox-Regression) oder dem Achsenabschnitt β_0 (logistische, multinomiale Regression) entspricht. Einsetzen von $0_{|p|}$ in die Log-Likelihood (2.67) ergibt $\mathcal{L}_{null} = \mathcal{L}(0_{|p|}) = -\sum_{i=1}^{q} \log(|R_i|)$. Das Pathwise CD kann bereits vor λ_m gestoppt werden, indem von dem Abbruchkriterium

$$D(\hat{\beta}(\lambda_k)) - D_{null} \geq 0.99 D_{null}$$

Gebrauch genommen wird. Das Pathwise CD bricht somit ab, wenn das aus dem für λ_k resultierende Modell 99 Prozent der Null-Devianz erklärt, sodass es meist vor λ_m zum Abbruch kommt. Dieses Abbruchkriterium wird ebenfalls bei allen anderen vorgestellten Problemen (klassische lineare Regression, logistische und multinomiale Regression) verwendet.

Es sei angemerkt, dass die Cox-Regression bei allen Problemen, die Studien über einen festen Zeitraum umfassen und zum Ende Beobachtungen der Form (x^i, y_i, δ_i), $i = 1, ...n$, zur Verfügung stehen, einsetzbar ist.

2.5 SAFE und Strong Rules zur Eliminierung von Variablen

Eines der wichtigsten Ziele von Selektionsverfahren ist das Extrahieren aller tatsächlich relevanten Merkmale, sodass kein relevantes Merkmal fälschlicherweise den irrelevanten zugeordnet wird. In diesem Kapitel geht es darum, mögliche Regeln aufzuzeigen, wie Merkmale a priori aus dem Modell eliminiert werden können, und zwar so, dass möglichst gewährleistet werden kann, dass keines dieser Merkmale für das Modell von Wichtigkeit ist. Hier bedeutet dies, dass mittels der vorliegenden Daten und des RP's λ eine Teilmenge $M \subseteq \{1, ..., p\}$ von Merkmalen aussortiert wird, sodass $\beta_M = 0_{|M|}$. Dies geschieht vor der Approximation von $\beta_L \in \mathbb{R}^p$. Nach der Eliminierung wird LASSO mit der Systemmatrix $X_{-M} = (x_{ij})_{\{j|j \notin M\}} \in \mathbb{R}^{n \times (p-|M|)}$ gelöst. Die Dimensionsreduktion kann den Speicher- und numerischen Aufwand für die Berechnung von $\tilde{\beta}_L \in \mathbb{R}^{p-|M|}$ erheblich senken. Die Lösung des ursprünglichen Systems lautet $\beta_L = (\tilde{\beta}_L, 0_{|M|}) \in \mathbb{R}^p$. Die Validierung der Lösung erfolgt mittels der KKT-Bedingungen. Dieses Kapitel behandelt Regeln für die klassische lineare und die logistische Regression. Mit derselben Vorgehensweise lassen sich diese Regeln auf allgemeine regularisierte konvexe Probleme übertragen.

2.5.1 Klassische lineare Regression

Der Support einer Lösung β_L liegt im Equicorrelation Set (2.22), die entsprechenden Variablen weisen somit das maximale absolute Skalarprodukt mit dem Residuum $y - X\beta_L$ auf. Dies gilt ebenso für den Lösungsalgorithmus LARS 1 in jeder Iteration l für den jeweiligen RP λ_l, vgl. Kapitel 2.3.1, $l \in \{1, ..., m\}$. Dieser Tatsache entsprang die Vermutung, dass die Variablen, deren absolutes Skalarprodukt mit dem Residuum gering ist, eher aus dem Modell ausgeschlossen werden können als diejenigen mit einer hohen Korrelation mit dem Residuum.

$\lambda_{\max} = \frac{1}{n} \max_{j \in \{1,...,p\}} |X_j^T y| = \frac{1}{n} \|X^T y\|_\infty$ ist der kleinste Wert für λ, für das $\beta_L(\lambda) = 0_{|p|}\ \forall \lambda \geq \lambda_{\max}$ gilt, vgl. Kapitel 2.3.2. Aus der Cauchy-Schwarz-Ungleichung, die besagt, dass für zwei beliebige Vektoren x, y derselben Dimension $|x^T y| \leq \|x\|\|y\|$ gilt, folgt

$$\begin{aligned} \lambda_{\max} = \frac{1}{n} \max_{j \in \{1,...,p\}} |X_j^T y| \ &\leq \ \frac{1}{n} \max_{j \in \{1,...,p\}} \|X_j\|_2 \|y\|_2 = \frac{1}{n} \|y\|_2 \\ \iff \quad & \frac{1}{n} \frac{\|y\|_2}{\lambda_{\max}} \geq 1 \ . \end{aligned} \tag{2.75}$$

SAFE Rule El Ghaoui et al. (2010) [13] haben mittels des duales Problems von LASSO (siehe Anhang A6) die sogenannte *SAFE Rule* hergeleitet, die diejenigen Merkmale $j \in \{1, ...p\}$ für $\lambda < \lambda_{\max}$ aussortiert, für die gilt

$$\frac{1}{n} |X_j^T y| < \lambda - \frac{1}{n} \|X_j\|_2 \|y\|_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}} \quad . \tag{2.76}$$

Die Autoren haben gezeigt, dass für jedes j, das die Ungleichung (2.76) erfüllt, der Koeffizient $\beta_{j,L}$ in der Lösung tatsächlich den Wert Null annimmt und die entsprechende Variable aus den Daten entfernt werden kann. Diese Regel lässt sich auf Lösungsalgorithmen wie LARS und CD, die Lösungen für eine Sequenz von RP berechnen, übertragen, sodass in der Iteration l alle Variablen j aussortiert werden können, für die

$$\frac{1}{n} |X_j^T (y - X\beta(\lambda_{l-1}))| < \lambda_{l-1} - \frac{1}{n} \|X_j\|_2 \|y\|_2 \frac{\lambda_l - \lambda_{l-1}}{\lambda_l} \quad , \tag{2.77}$$

gilt, wobei $\lambda_{l-1} < \lambda_l$ [13]. Die Ungleichung (2.77) bezeichnet die *sequentielle* bzw. *rekursive SAFE Rule*. Für Details und Herleitung siehe [13].

Strong Rules Auf Basis der SAFE Rule haben Tibshirani et al. (2010) [38] die sogenannten *Strong Rules* aufgestellt, mit dem Ziel, möglichst viele Variablen auszusortieren, sodass die Lösung noch effizienter gewonnen werden kann. Die Autoren

unterscheiden wie bei der SAFE Rule zwischen der *Basic Strong Rule*

$$\frac{1}{n}|X_j^T y| < \lambda - (\lambda_{\max} - \lambda) = 2\lambda - \lambda_{\max} \ , \tag{2.78}$$

die durch Setzung $\frac{1}{n}\frac{\|X_j\|_2\|y\|_2}{\lambda_{\max}} = 1$ aus der SAFE Rule entsteht, und der *sequentiellen* bzw. *rekursiven Strong Rule* mit $\lambda_0 = \lambda_{\max}$

$$c_j(\lambda) := \frac{1}{n}|X_j^T(y - X\beta(\lambda_{l-1}))| < 2\lambda_l - \lambda_{l-1} \ . \tag{2.79}$$

Im ersten Schritt eines Lösungsalgorithmus reduziert sich die sequentielle auf die Basic Strong Rule. Bei standardisierten Daten schließt die Basic Strong Rule wegen der Ungleichungen in (2.75) mehr Variablen aus dem Modell aus als die SAFE Rule. Praktische Untersuchungen zeigen, dass dies auch im Falle eines nicht standardisierten Systems überwiegend zutrifft.

Da die meisten Lösungsalgorithmen darauf abzielen, Lösungen für eine Sequenz von RP zu bestimmen, sind vor allem die rekursiven Regeln (2.77) und (2.79) von Interesse. Die sequentielle Strong Rule eliminiert eine große Anzahl an Merkmalen, wobei keine oder nur wenige dieser Merkmale fälschlicherweise den redundanten zugewiesen werden. Zur Validierung der Strong Rules werden die KKT-Bedingungen herangezogen. Die folgenden Ergebnisse stammen aus [21, 38].

Mit der Definition von $c_j(\lambda)$ (2.79) lassen sich die KKT-Bedingungen schreiben als

$$\forall j \in \{1, ..., p\} : \quad c_j(\lambda) \in \begin{cases} \lambda s_j(\lambda), & \beta_{j,L} \neq 0 \\ [-\lambda, \lambda], & \beta_{j,L} = 0 \ , \end{cases} \tag{2.80}$$

wobei $s_j(\lambda) = \text{sign}(\beta_{j,L}(\lambda))$.

Betrachte zunächst die Basic Strong Rule, die der rekursiven Strong Rule im ersten Schritt eines Lösungsalgorithmus entspricht. Die KKT-Bedingungen besagen, dass $\frac{1}{n}|X_j^T y| < \lambda_{\max}$ für alle Koeffizienten j, die nicht im Modell enthalten sind, gilt. Die Basic Strong Rule legt fest, dass $|c_j(\lambda)| \leq \lambda_{\max} - \lambda$ für ein $\lambda < \lambda_{\max}$ gelten soll. Gilt $|c_j(\lambda)| < \lambda - (\lambda_{\max} - \lambda) = 2\lambda - \lambda_{\max}$, so gilt wegen der KKT-Bedingungen (2.80) die Ungleichung $|c_j(\lambda)| < \lambda$, sodass $\beta_{j,L}(\lambda) = 0$ folgt.

Der sequentiellen Strong Rule liegt die folgende Annahme zugrunde:

$$|c_j(\hat{\lambda}) - c_j(\tilde{\lambda})| \leq |\hat{\lambda} - \tilde{\lambda}| \quad \forall j \in \{1, ..., p\}, \ \forall \hat{\lambda}, \tilde{\lambda} \geq 0 \ . \tag{2.81}$$

Es wird somit gefordert, dass die Funktion $c_j(\lambda)$ Lipschitz-stetig ist, wobei die Lipschitz-Konstante Eins beträgt. Aus der Analysis ist bekannt, dass dies dazu äquivalent ist, dass $c_j(\lambda)$ fast überall differenzierbar und das Differential (falls existent)

beschränkt ist. $c_j(\lambda)$ ist stückweise linear, stetig und differenzierbar für alle λ, bis auf $\lambda \in \{\lambda_{\max}, \lambda_1, ..., \lambda_l, ...\}$. Die Sequenz der RP ist eine abzählbare Menge und somit eine Nullmenge, sodass $c_j(\lambda)$ differenzierbar fast überall folgt. Es muss demnach gelten:

$$|\partial_\lambda c_j(\lambda)| \leq 1 \quad \text{fast überall} \; . \tag{2.82}$$

Die Dreiecksungleichung, die Ungleichungen (2.79) und (2.81) ergeben

$$\begin{aligned} |c_j(\lambda_l)| &= |c_j(\lambda_l) - c_j(\lambda_{l-1}) + c_j(\lambda_{l-1})| \\ &\leq |c_j(\lambda_l) - c_j(\lambda_{l-1})| + |c_j(\lambda_{l-1})| \\ &\overset{(2.81)}{\leq} (\lambda_l - \lambda_{l-1})|c_j(\lambda_{l-1})| \\ &\overset{(2.79)}{<} (\lambda_l - \lambda_{l-1}) + (2\lambda_l - \lambda_{l-1}) = \lambda_l \; , \end{aligned}$$

sodass aus den KKT-Bedingungen $\beta_j(\lambda_l) = 0$ folgt und die Variable j tatsächlich aus dem Modell ausgeschlossen werden kann [38].

Nun bleibt die Frage, ob und wann die Voraussetzung (2.81) bzw. (2.82) gewährleistet werden kann. Aus den Regeln der Differentialrechnung folgt [21]

$$\partial_\lambda c_j(\lambda) = \begin{cases} s_j(\lambda), & \beta_{j,L} \neq 0 \\ s_j(\lambda) + \lambda \partial_\lambda s_j(\lambda) \; , & \beta_{j,L} = 0 \; . \end{cases}$$

Für alle Variablen $j \in J = \text{supp}(\beta(\lambda_l))$ ist diese Voraussetzung erfüllt. Für ausgeschiedene Variablen ist dies nicht zwingend gegeben. Insbesondere bei stark korrelierten Merkmalen zwischen $j \in J$ und $j \notin J$ oder bei einer (sehr) kleinen Schrittweite zwischen den RP kann diese Bedingung scheitern. Hierbei kann die sequentielle Strong Rule relevante Variablen fälschlicherweise ausschließen. Sind die Variablen hingegen unkorreliert, so begeht die sequentielle Strong Rule nur selten Fehler. Bei einem Problem mit $n \ll p$ weist diese umso besseres Verhalten auf, hinsichtlich sowohl der aus dem Modell ausgeschlossenen hohen Anzahl an Variablen als auch der Gewährleistung von deren Irrelevanz. Die Strong Rules sind eine sehr effiziente Heuristik, die zudem eine sehr einfache Form haben.

Die Überprüfung, ob das berechnete $\beta_L = (\tilde{\beta}_L, 0_{|M|}) \in \mathbb{R}^p$ eine mögliche Lösung von LASSO ist, erfolgt anhand der KKT-Bedingungen. Ist dies der Fall, so stellt β_L eine Lösung dar. Existieren unter den aussortierten Variablen $j \notin M$ solche, die die KKT-Bedingungen (2.80) nicht erfüllen, so werden diese dem Support hinzugefügt und das LASSO erneut gelöst. Sobald alle Variablen den KKT-Bedingungen genügen, ist eine zulässige Lösung gefunden. Wie jedoch bereits erklärt, sind die Strong Rules recht stabil, sodass Fehler (vor allem bei $n \ll p$) nur selten vorkommen. [21]

2.5.2 Logistische Regression

Die folgenden Ausführungen stammen aus [38]. Betrachte das LASSO für die logistische Regression, das durch das Optimierungsproblem (2.50) gegeben ist. Die KKT-Bedingungen für eine Lösung $(\hat{\beta}_0, \hat{\beta})$ lauten

$$\frac{1}{n} X^T (y - p(x; \hat{\beta}_0, \hat{\beta})) = \lambda s, \quad s \in \nabla_\beta \|\hat{\beta}\|_1 \ ,$$

wobei $p(x; \hat{\beta}_0, \hat{\beta}) = \Pr(y = 1 \mid x) = \frac{1}{1+e^{-(\hat{\beta}_0 + \hat{\beta}^T x)}} \in (0, 1)$ ist, vgl. die Definition (2.47). Da $\hat{\beta}_0$ und $\hat{\beta}$ als Funktionen von λ aufgefasst werden, $\hat{\beta}_0 = \hat{\beta}_0(\lambda)$ $\hat{\beta} = \hat{\beta}(\lambda)$, folgt $p = p(x; \hat{\beta}_0(\lambda), \hat{\beta}(\lambda))$, sodass p ebenso eine Funktion in λ ist. Definiere $c_j = X_j^T (y - p(x; \hat{\beta}_0(\lambda), \beta(\lambda)))$ und es gelte die Ungleichung (2.81). Gilt $\hat{\beta}(\lambda) = 0_{|p|}$ für ein λ, so ist $p(\hat{\beta}_0(\lambda), \hat{\beta}(\lambda)) = \frac{1}{1+e^{-\hat{\beta}_0(\lambda)}}$, wobei $\frac{1}{1+e^{-\hat{\beta}_0(\lambda)}} \leq 1$. Für alle $y_i, i = 1, ..., n$, gilt $y_i \in \{0, 1\}$, sodass $\overline{y} = \frac{1}{n} \sum_1^n y_i \leq 1 \ \forall \lambda$. Das $\lambda_{\max}$ ist folglich durch

$$\lambda_{\max} = \max_{i \in \{1,...,n\}} |x_i^T (y - \overline{p})|$$

gegeben, wobei $\overline{p} = \mathbb{1}\overline{y}$. Analog zur Ungleichung (2.78) lautet die Basic Strong Rule für die logistische Regression

$$\frac{1}{n} |X_j^T (y - \overline{p})| < 2\lambda - \lambda_{\max} \ .$$

Für die sequentielle Strong Rule für die logistische Regression folgt

$$\frac{1}{n} |X_j^T (y - p(\hat{\beta}_0(\lambda_{l-1}), \hat{\beta}(\lambda_{l-1}))| < 2\lambda_l - \lambda_{l-1} \ .$$

Für die multinomiale und die Cox-Regression lassen sich die Basic Strong Rule und die sequentielle Strong Rule ähnlich herleiten.

2.6 Performance von LASSO

Dieses Kapitel soll das Verhalten von LASSO in erster Linie hinsichtlich der Korrektheit der Variablenselektion aufzeigen und untersuchen, wie verlässlich eine gefundene Lösung ist.

2.6.1 Familywise Error Rate

Fasse die Variablenselektion zunächst als einen Hypothesentest auf. Die Nullhypothese entspreche der Irrelevanz der Variable j, sodass $\beta_j = 0$, die Gegenhypothese sei $\beta_j \neq 0$, $j \in \{1, ..., p\}$. Die sogenannte *Familywise Error Rate* (FWER) entspricht

der Wahrscheinlichkeit, die Nullhypothese fälschlicherweise zurückzuweisen, sodass in diesem Zusammenhang eine redundante Variable den relevanten zugeordnet wird,

$$\text{FWER} := P(\#\{\text{false positives}\} \geq 1) \ .$$

Die FWER entspricht somit dem Fehler 1.Art (auch α-Fehler). Unter bestimmten Voraussetzungen kann die Kontrolle der FWER zu einem gewünschten Niveau $\alpha \in [0, 1]$ garantiert werden.

Die Spalten der Matrix X seien orthogonal, sodass $X^T X = \text{diag}(d_1, ..., d_p)$, $d_j \in \mathbb{R}\ \forall j \in \{1, ..., p\}$, gelte. Bei einer Matrix mit orthogonalen Spalten lässt sich die FWER kontrollieren. Nehme zudem an, dass die Spalten normiert sind, sodass $X^T X = I_p$ gilt.

Betrachte das Ausgangsproblem der linearen Regression $y = X\beta^* + \epsilon$. Es bestehe weiterhin die Annahme, dass der Fehlerterm ϵ unabhängig identisch normalverteilt ist, $\epsilon \ \sim \ \mathcal{N}(0, \sigma^2 I_n)$, wobei die Standardabweichung σ bekannt ist oder mit nur wenig Aufwand bestimmt werden kann. Aus der Orthonormalität der Spalten folgt

$$\tilde{y} = X^T y = X^T X\beta^* + X^T\epsilon = \beta^* + X^T\epsilon = \beta^* + \tilde{\epsilon} \ \sim \ \mathcal{N}(\beta^*, \sigma^2 I_p) \ .$$

Das Ziel, die relevanten Merkmale korrekt zu bestimmen, besteht hierbei in der Gegenüberstellung und Überprüfung von p Hypothesen: die Hypothese $H_{0,j}$ sei $\beta_j = 0$ und die Gegenhypothese $H_{1,j}$ entspreche $\beta_j \neq 0$, $j \in \{1, ..., p\}$ [5]. Zur Kontrolle der FWER kann hierbei die sogenannte $Bonferroni - Methode$ Einsatz finden. Die Hypothese $H_{0,j}$ wird genau dann zurückgewiesen, wenn

$$|\tilde{y}_j| > \sigma \cdot \Phi^{-1}\left(1 - \frac{\alpha}{2p}\right)$$

gilt, wobei $\Phi^{-1}(\alpha)$ das α-Quantil der Standardnormalverteilung ist. Die Zurückweisung hängt nur von der Anzahl der Merkmale p und der Standardabweichung σ ab und ist somit unabhängig von X.

Bei einer Matrix X mit orthonormalen Spalten ist die Lösung von LASSO β_L durch den Soft-Thresholding Operator (2.30) gegeben. Wird folglich der RP gesetzt als

$$\lambda_{\text{bonf}} = \frac{1}{n}\sigma \cdot \Phi^{-1}\left(1 - \frac{\alpha}{2p}\right) \ , \tag{2.83}$$

so ist FWER $\leq \alpha$ erfüllt [5].

FWER ist eine recht restriktive Größe, denn ein niedriger Wert für α führt zwar dazu, dass nur wenige oder keine irrelevanten Merkmale zum Modell gehören, dadurch besteht allerdings die Gefahr, dass viele relevante Merkmale nicht selektiert

werden. Es hat sich in der Praxis herausgestellt, dass die Verwendung von λ_{bonf} in Verbindung mit LASSO in den meisten Fällen dazu führt, dass zu wenige Merkmale oder keine extrahiert werden. Außerdem geht die Kontrolle über die FWER im Falle von Matrizen, deren Spalten nicht orthogonal sind, sehr schnell verloren [5]. Daher ist diese Größe vor allem bei $n \ll p$ nicht von Interesse und wird in dieser Arbeit nicht weiter behandelt.

2.6.2 Konsistenz und Irrepresentable Condition

Es gibt verschiedene Herangehensweisen, eine Lösung β_L zu beurteilen. Zum einen kann dazu bspw. die L2-Norm der Differenz zwischen der wahren Lösung β^* (1.1) und β_L, $\|\beta_L - \beta^*\|_2^2$, oder der Vorhersagefehler als $\|X(\beta_L - \beta^*)\|_2^2$, zum anderen die Konsistenz der Variablenselektion herangezogen werden. Die Konsistenz ist die Fähigkeit, die richtigen Merkmale zu extrahieren.

Bühlmann und van de Geer (2011) [9] haben gezeigt, dass der Vorhersagefehler $\|X(\beta_L - \beta^*)\|_2^2$ für $p, n \to \infty$ mit einer hohen Wahrscheinlichkeit die folgende Ungleichung erfüllt:

$$\frac{1}{n}\|X(\beta_L - \beta^*)\|_2^2 \leq C \cdot \|\beta^*\|_1 \sqrt{\frac{\log(p)}{n}} \ .$$

Ist $\|\beta^*\|_1 = o(\sqrt{\frac{n}{\log(p)}})$, so lassen sich mittels der Lösungen von LASSO recht präzise Vorhersagen treffen. Dabei gibt es keine Anforderungen an die Matrix X.

LASSO wird in erster Linie zur Variablenselektion eingesetzt, daher spielt die Konsistenz eine wesentlich größere Rolle. Denn ist $\|\beta_L - \beta^*\|_2$ oder der Vorhersagefehler $\frac{1}{\sqrt{n}}\|X(\beta_L - \beta^*)\|_2$ gering, so bedeutet dies nicht, dass β_L und β^* denselben oder ähnlichen Support haben, wobei dies insbesondere auf den Vorhersagefehler zutrifft [21, 45]. Selbst wenn dieser sehr gering ist, können sich die Supports von β_L und β^* stark unterscheiden.

J^* sei der Support der optimalen Lösung β^*, J der Support von β_L. Nun ist das Ziel, dass J und J^* übereinstimmen, sodass durch β_L alle relevanten Merkmale und nur diese selektiert werden. Es sei angemerkt, dass für den Fall, dass J^* korrekt bestimmt wurde, die Norm $\|\beta_L - \beta^*\|_2$ verringert werden kann, indem die Methode der kleinsten Quadrate auf die reduzierte Systemmatrix $X_{J^*} \in \mathbb{R}^{n \times |J^*|}$ angewendet wird [21]. Wie bislang seien die Spalten der Matrix standardisiert und σ bezeichne die Standardabweichung des Fehlerterms, vgl. (1.1).

Eine Eigenschaft der Matrix, die Konsistenz gewährleistet, ist die sogenannte *Irrepresentable Condition* [45]. Diese verlangt, dass ein $\gamma \in (0,\ 1]$ existiert, so-

dass gilt

$$\max_{j \notin J^*} \|(X_{J^*}^T X_{J^*})^{-1} X_{J^*}^T X_j\|_1 \leq 1 - \gamma \ . \tag{2.84}$$

Verglichen mit der Lösung (2.16) kann $(X_{J^*}^T X_{J^*})^{-1} X_{J^*}^T X_j$ als ein Lösungsvektor von OLS mit der Systemmatrix X_{J^*} und dem Output X_j interpretiert werden, dessen L1-Norm durch $1 - \gamma$ beschränkt ist. Die Irrepresentable Condition besagt, dass alle Spalten, die nicht im Support J^* liegen, mit allen Spalten, die zum Support J^* gehören, ein geringes Skalarprodukt aufweisen. Es wird demnach verlangt, dass die Spalten des Supports J^* mit denjenigen außerhalb J^* möglichst wenig korreliert sind. Es ist wünschenswert, dass alle Spalten $\{X_j\}_{j \notin J^*}$, orthogonal zu allen Spalten von X_{J^*} sind, sodass $\gamma = 1$ folgt. Die Irrepresentable Condition ist somit bei Matrizen mit orthogonalen Spalten erfüllt.

Ist $n \ll p$, so ist davon auszugehen, dass die Spalten, die nicht zu J^* gehören, zu den Spalten J^* nicht orthogonal sind. Es besteht dennoch die Hoffnung, dass deren Skalarprodukt gering ist. Außerdem setzt die Bedingung die Beziehung (2.84) voraus, sodass $(X_{J^*}^T X_{J^*})$ invertierbar ist. Die Matrix $X_{J^*}^T X_{J^*}$ ist folglich symmetrisch positiv definit, sodass alle Eigenwerte positiv sind und für den kleinsten Eigenwert

$$\frac{1}{n} \mu_{min} \left(X_{J^*}^T X_{J^*} \right) \geq C_{min}$$

mit einer festen Konstante $C_{min} > 0$ gilt. Wähle einen RP λ mit

$$\lambda \geq \frac{8\sigma}{\gamma} \sqrt{\frac{\log(p)}{n}} \ . \tag{2.85}$$

Je nach Höhe von γ und σ kann dies $\lambda \gg \sqrt{\frac{\log(p)}{n}}$ bedeuten. Ist die Irrepresentable Condition erfüllt, so gelten mit dieser Wahl des RP's mit einer Wahrscheinlichkeit größer als $1 - c_1 e^{-c_2 n \lambda^2}$, $c_1 \geq 0$, $c_2 \geq 0$ feste Konstanten, die folgenden Aussagen [21]:

1) β_L ist eindeutig. Besonders bei $n \ll p$ ist dies von Bedeutung, denn wie im Kapitel 2.2 erklärt ist Eindeutigkeit in hohen Dimensionen sehr selten gegeben.

2) $J \subseteq J^*$, J enthält somit keine falsch selektierten Merkmale.

3)

$$\|\beta_{J^*,L} - \beta_{J^*}^*\| \leq \lambda \left(\frac{4\sigma}{\sqrt{C_{min}}} + \left\| \left(\frac{1}{n} X_{J^*}^T X_{J^*} \right)^{-1} \right\|_\infty \right) =: B(\lambda, \sigma, X) \ ,$$

wobei $\|A\|_\infty := \max_{b \neq 0} \frac{\|Ab\|_\infty}{\|b\|_\infty}$ die Maximumsnorm für eine Matrix A bezeichnet.

4) Gilt $\min_{j \in J^*} |\beta_j^*| > B(\lambda, \sigma, X)$, so ist LASSO konsistent, da J alle Indizes, die dies erfüllen, enthält, sodass aus 2) $J^* = J$ mit einer hohen Wahrscheinlichkeit folgt. Diese Bedingung an die Einträge von β^* bedeutet, dass diese betragsmäßig hinrei-

chend groß sein sollten, damit J den Support J^* gänzlich trifft. Existieren Einträge β_j^*, die unter der Schranke $B(\lambda, \sigma, X)$ liegen, so liegen die entsprechenden Indizes nicht in J, sodass $J \subset J^*$.

Für den Beweis dieser Aussagen siehe [21, 45].

Konsistenz der Variablenselektion bedeutet zudem, dass mit steigender Anzahl an Beobachtungen n die Wahrscheinlichkeit, exakt den Support J^* zu selektieren, gegen Eins strebt. Zhao und Yu (2006) [45] haben gezeigt, dass unter der Voraussetzung der Irrepresentable Condition dies erfüllt ist. Es folgt

$$P(J = J^*) \to 1 \quad (n \to \infty) \; .$$

Zur Herleitung der Irrepresentable Condition zeigen Zhao und Yu (2006) [45] unter anderem, dass im Falle starker Korrelation einer irrelevanten mit einer relevanten Variable LASSO unabhängig von der Größenordnung von n, p und deren Verhältnis kaum in der Lage ist, dies zu unterscheiden. Die Autoren belegen, dass LASSO bis auf wenige Ausnahmen genau dann konsistent ist, wenn die Irrepresentable Condition gilt. Für Konsistenz ist diese Bedingung somit (fast immer) notwendig. Zudem stellt diese nach der Argumentation der Autoren ebenfalls eine hinreichende Bedingung dar.

2.6.3 Instabilität der Kreuzvalidierung

Das vorige Kapitel gibt eine mögliche Wahl des RP's zur Kontrolle der FWER, wobei dieser lediglich von p und dem gewünschten Niveau α abhängt. Soll der RP in Abhängigkeit von den Daten gewählt werden, so bietet sich Kreuzvalidierung (engl. cross validation, CV) an.

N-fache CV besteht darin, die Menge der Beobachtungen $\{1, ..., n\}$ zufällig in N Teilmengen T_1, ..., T_N mit der Mächtigkeit $|T_i| = \lfloor \frac{n}{N} \rfloor$ oder $|T_i| = \lceil \frac{n}{N} \rceil$, $i \in \{1, ...N\}$, zu zerlegen, jeweils $N-1$ Teilmengen als Trainingsset und die übrige Teilmenge als Testset zu gebrauchen. Dies wird N Mal durchgeführt, sodass jede Teilmenge einmal als Testset eingesetzt wird. Die maximale Anzahl an Teilmengen ist $N = n$, sodass es sich hierbei um Leave-One-Out-CV (LOOVC) handelt. Das Gütemaß für die CV ist in der Regel der Vorhersagefehler. Im Falle der klassischen linearen Regression wird somit der CV-Fehler für λ folgendermaßen berechnet:

$$CV(\lambda, X, y) = \frac{1}{N} \sum_{i=1}^{N} \| y_{(T_i)} - X^{(T_i)} \hat{\beta}^{(-T_i)}(\lambda) \|_2^2 \; . \tag{2.86}$$

Für die hier vorgestellten GLM (2.54), (2.60) und (2.73) ist der CV-Fehler jeweils

definiert als

$$
\begin{aligned}
CV(\lambda, X, y) &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_Q\left(\hat{\beta}_0(\lambda), \hat{\beta}^{(-T_i)}(\lambda)\right) \ , \\
CV(\lambda, X, y) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \mathcal{L}_Q^k\left(\hat{\beta}_0^k(\lambda), \hat{\beta^k}^{(-T_i)}(\lambda)\right) \ , \\
CV(\lambda, X, y) &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_Q\left(\hat{\beta}^{(-T_i)}(\lambda)\right) \ .
\end{aligned} \tag{2.87}
$$

Dabei ist $y_{(T_i)}$ der Output der Beobachtungen in T_i, $X^{(T_i)}$ eine Teilmatrix von X (die Testmenge), die die Zeilen T_i enthält, und $\hat{\beta}^{(-T_i)}(\lambda)$ ist der resultierende Vektor, für dessen Bestimmung die Teilmengen $\{T_j\}_{j\neq i}$ benutzt wurden.

Wie in den Kapiteln 2.3.2 und 2.4 beschrieben, werden Lösungen für LASSO $\hat{\beta}(\lambda_1)$,..., $\hat{\beta}(\lambda_m)$ zu einer absteigenden Folge an RP $\lambda_0 > ... > \lambda_m$ berechnet, wobei die Startlösung zu λ_0 ein Nullvektor ist. Mit dem sinkenden RP steigt die Anzahl der Nichtnulleinträge der einzelnen Iterierten, bis der Lösungsvektor vollbesetzt ist. $CV(\lambda_l, X, y)$ bezeichne den CV-Fehler zu dem RP λ_l, $l \in \{1, ..., m\}$. Der RP mit dem kleinsten CV-Fehler definiert als $\lambda_{\min}$ ergibt sich schließlich als

$$
\lambda_{\min} = \underset{\lambda_l}{\operatorname{argmin}}\ CV(\lambda_l, X, y) \ . \tag{2.88}
$$

Bei LASSO ist die Wahl der zu verwendenden Anzahl an Teilmengen N für CV nicht trivial. Denn es hat sich herausgestellt, dass das durch LASSO zu einem λ selektierte Modell anfällig gegenüber der zum Trainieren verwendeten Teilmenge und damit gegenüber N ist, sodass LASSO folglich hinsichtlich der Variablenselektion instabil ist [42, 32]. Wie N zu wählen ist, wird in dieser Arbeit nicht untersucht. Kapitel 4 wird die Instabilität in der praktischen Anwendung aufzeigen. Ebenso bei der Anzahl der selektierten Merkmale zeigt LASSO hohe Varianz auf. Roberts und Nowak (2014) [32] führen das sogenannte *Percentile-LASSO* ein, das das LASSO stabilisieren soll. Für Details siehe [32].

3 SLOPE

Dieses Kapitel richtet sich nach [4, 5].

Der Ansatz SLOPE wurde erstmals von Bogdan et al. (2013) vorgestellt [4]. Das Optimierungsproblem SLOPE hat die Form

$$
\min_{\beta\in\mathbb{R}^p} \left\{ f_{SLOPE}(\beta) := \underbrace{\frac{1}{2}\|y - X\beta\|_2^2}_{\text{Ziel-/Verlustfunktion}} + \underbrace{J_{\boldsymbol{\lambda}}(\beta)}_{\text{Regularisierer}} \right\} . \tag{3.1}
$$

$J_{\boldsymbol{\lambda}}(\beta)$ stellt die sortierte L1-Norm[11] dar, die definiert ist als

$$J_{\boldsymbol{\lambda}}(\beta) := \lambda_1|\beta|_{(1)} + ... + \lambda_p|\beta|_{(p)} \ ,$$

wobei $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_p)$, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$ und $|\beta|_{(1)} = \max_{j\in\{1,...,p\}} |\beta_j| \geq ... \geq |\beta|_{(p)} = \min_{j\in\{1,...,p\}} |\beta_j|$. Wie bei LASSO stellt die Verlustfunktion die (mit $\frac{1}{2}$ skalierte) Summe der kleinsten Quadrate f_{OLS} dar und der zweite Term $J_{\boldsymbol{\lambda}}(\beta)$ von $f_{SLOPE}(\beta)$ ist der Regularisierer. Die Koeffizienten von β erhalten somit unterschiedliche RP. Dies geschieht in der Weise, dass je höher der betragsmäßige Eintrag $|\beta|_{(j)}$, je bedeutsamer das entsprechende Merkmal für das Modell ist, desto höher ist der RP λ_j [4, 5]. Dies hat unter anderem zur Folge, dass sich anhand einer Lösung die Wichtigkeit der Merkmale ablesen lässt. Die Sequenz der RP $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$ ist eine monoton fallende Folge. Dadurch ist die Regularisierung bei SLOPE mit $J_{\boldsymbol{\lambda}}$ sanfter als bei LASSO mit einem hohen RP λ, denn für $\lambda \geq \lambda_1$ gilt $J_{\boldsymbol{\lambda}}(\beta) \leq \lambda\|\beta\|_1$.

Gilt $\lambda_k = \lambda_j \ \forall \ k, j \in \{1, ..., p\}$, so reduziert sich SLOPE unter Vernachlässigung des Vorfaktors $\frac{1}{n}$ auf LASSO (2.6). SLOPE kann demnach als eine Erweiterung von LASSO betrachtet werden. Bogdan et al. (2013) [4] bezeichnen das Problem (3.1) zunächst als *ordered LASSO*, bevor die Autoren es in [5] in SLOPE umbenennen.

Ein Regressionsvektor, der SLOPE (3.1) löst, sei definiert als

$$\beta_S \ \in \ \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \ \frac{1}{2}\|y - X\beta\|_2^2 \ + J_{\boldsymbol{\lambda}}(\beta) \ . \tag{3.2}$$

Bogdan et al. (2015) [5] bezeichnen β_S (3.2) als SLOPE. In dieser Arbeit wird das Optimierungsproblem (3.1) als SLOPE aufgefasst.

SLOPE ist ebenso wie LASSO als Zusammensetzung einer streng konvexen (Zielfunktion) und konvexen (sortierte L1-Norm) Funktion streng konvex. Ähnlich wie die L1-Regularisierung bei LASSO schränkt hier der Regularisierungsterm $J_{\boldsymbol{\lambda}}(\beta)$ den Definitionsbereich von f_{SLOPE} ein, vgl. Abbildung 7, sodass eine Lösung $\beta_S \in \operatorname{argmin} f_{SLOPE}$ existiert. Da f_{SLOPE} wie LASSO bezüglich $X\beta$ und nicht β streng konvex ist, ist $X\beta_S$ zwar für jede Lösung β_S eindeutig, β_S ist es nicht zwingend. Mit ähnlichen Argumenten wie für LASSO, vgl. Kapitel 2.2, kann die Eindeutigkeit der Lösung nur unter bestimmten Voraussetzungen garantiert werden.

Bogdan et al. (2015) [5] charakterisieren SLOPE unter anderem als einen Hypothesentest, wobei die Nullhypothese der Irrelevanz des Merkmals j, somit $\beta_j = 0, \ j \in \{1, ..., p\}$, und die Gegenhypothese $\beta_j \neq 0$ entspricht. Die Nullhypothese wird genau dann zurückgewiesen, wenn der j-te Eintrag des Lösungsvektors β_S Null ist,

[11] Nachrechnen der Axiome zeigt, dass die sortierte L1-Norm insbesondere eine Norm ist, siehe Anhang A7.

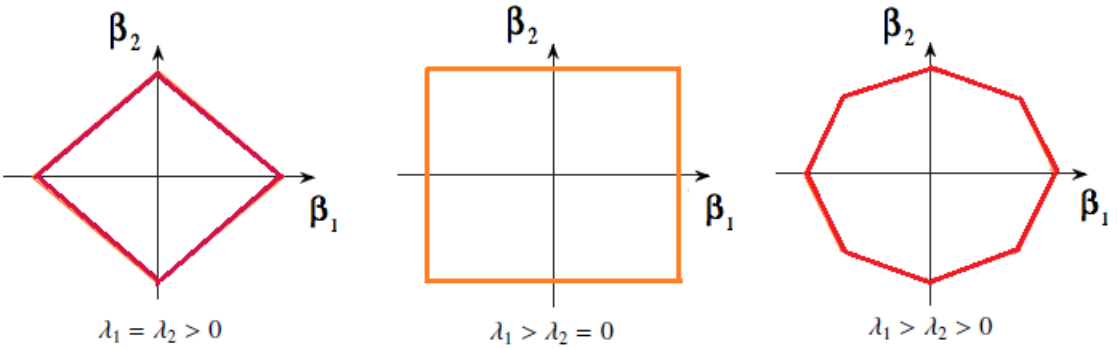


Abbildung 7 Geometrie von SLOPE. Definitionsbereich von SLOPE in zwei Dimensionen. Gilt $\lambda_1 = \lambda_2$ (links), so liegt L1-Regularisierung vor, ist $\lambda_2 = 0$ (Mitte), so bedeutet dies, dass der betragsmäßig maximale Koeffizient von β einen bestimmten Wert nicht überschreiten darf, die sortierte L1-Regularisierung entspricht hier $\lambda_1 \|\beta\|_\infty$. Sind beide Parameter ungleich Null, so ist der Definitionsbereich in zwei Dimensionen auf ein Oktagon beschränkt. Modifiziert nach [25], S.8.

$\beta_{j,S} = 0$. Dies bedeutet somit, dass das Merkmal j genau dann als relevant gilt, wenn der Annahme über dessen Irrelevanz, und zwar $\beta_{j,S} = 0$, widersprochen wird.

Die Ausgangsannahme sei: eine Lösung β_S sei verfügbar und unter den selektierten Merkmalen $j \in \{1, ...p \mid \beta_{j,S} = 0\}$ befinden sich alle relevanten Merkmale. Das Ziel von SLOPE ist in erster Linie die Selektion korrekter Merkmale, somit solcher, die die Zielvariable tatsächlich beeinflussen. Die Absicht besteht zum einen in der Anpassung an die tatsächliche Anzahl relevanter Merkmale (Dünnbesetztheit der wahren Lösung), zum anderen in der Kontrolle des erwarteten Verhältnisses der nicht relevanten Merkmale zu allen ausgewählten Merkmalen. Die Größe, die dies beschreibt, ist die sogenannte *False Discovery Rate* (FDR) definiert als

$$\text{FDR} := E\left(\frac{V}{\max\{R, 1\}}\right) .$$

V bezeichnet die Anzahl falsch zurückgewiesener Nullhypothesen (Anzahl der ausgewählten irrelevanten Merkmale), R die Anzahl aller zurückgewiesenen Nullhypothesen (Anzahl aller ausgewählten Merkmale). FDR ist eine aussagekräftige globale Größe im multiplen Testen zur Bewertung des Begehens von Fehlern erster Art, die eine Zurückweisung der Nullhypothese, obwohl diese wahr ist, bezeichnet. Um das Ziel der Kontrolle der FDR zu erreichen, muss eine entsprechende Sequenz $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$ bestimmt werden. Der Quotient

$$\frac{V}{\max\{R, 1\}} \tag{3.3}$$

wird als *False Discovery Proportion* (FDP) bezeichnet, sodass FDR $= E$(FDP) gilt.

3.1 SLOPE bei orthogonalen Designs

Die Spalten der Matrix X seien orthonormal, sodass $X^T X = I_p$ gelte.

Für diesen speziellen Sachverhalt lassen sich einige Eigenschaften und Garantien herleiten. Betrachte das Ausgangsproblem der linearen Regression $y = X\beta^* + \epsilon$. Wie bislang gelte für den Fehlerterm $\epsilon \sim \mathcal{N}(0, \sigma^2 I_n)$, wobei die Standardabweichung σ bekannt ist oder nur einen geringen Aufwand zu deren Berechnung benötigt. Unter der Berücksichtigung der Orthonormalität der Spalten resultiert

$$\tilde{y} = X^T y = X^T X\beta^* + X^T\epsilon = \beta^* + X^T\epsilon = \beta^* + \tilde{\epsilon} \ \sim \ \mathcal{N}(\beta^*, \sigma^2 I_p) \ .$$

Wie im Kapitel 2.6.1 bereits besprochen wurde, findet hierbei die Bestimmung der relevanten Merkmale, indem die folgenden p Hypothesen einander gegenübergestellt und überprüft werden: die Hypothese $H_{0,j}$ entspreche $\beta_j = 0$ und die Gegenhypothese $H_{1,j}$ sei $\beta_j \neq 0$, $j \in \{1, ..., p\}$.

Es soll die FDR zum Niveau $q \in [0, 1]$ kontrolliert werden. Dies lässt sich mittels der sogenannten *Benjamin-Hochberg-Prozedur* (BH-Prozedur) erreichen. Die Vorgehensweise ist die folgende: zunächst wird die Zielvariable $\tilde{y}$ in betragsmäßig absteigender Reihenfolge sortiert, $|\tilde{y}_{(1)}| \geq ... \geq |\tilde{y}_{(p)}|$, sodass daraus die dazu entsprechenden Hypothesen $H_{(1)}, ..., H_{(p)}$ resultieren. Als nächstes fällt die Entscheidung, welche Merkmale den relevanten zugeordnet werden. Für alle Indizes $j \leq j_{BH}$ mit

$$j_{BH} = \max\left\{ j \ \middle| \ \frac{|\tilde{y}|_{(j)}}{\sigma} \geq \Phi^{-1}(1 - q_j) \right\}, \quad q_j = j\frac{q}{2p} \ ,$$

werden die Hypothesen $H_{(j)}$ zurückgewiesen. Unter der BH-Prozedur gilt für die FDR

$$\text{FDR} \ = \ q\frac{p_0}{p} \ ,$$

wobei $p_0 = |\{j \in 1, ...p| \ \beta_j = 0\}| = p - \|\beta\|_0$ die Anzahl wahrer Nullhypothesen bezeichnet.

Diese Vorgehensweise zeigt zum einen, dass eine Kontrolle der FDR zum Niveau q möglich ist und zum anderen, dass die Zurückweisung von $H_{(j)}$ und damit die Auswahl relevanter Merkmale in Abhängigkeit von den Daten geschieht. Bogdan et al. (2015) [5] haben gezeigt, dass bei der Sequenz der RP $\boldsymbol{\lambda}_{BH} \in \mathbb{R}^p$, deren Einträge gegeben sind als

$$\boldsymbol{\lambda}_{BH}(j) = \Phi^{-1}(1 - q_j), \quad q_j = j\frac{q}{2p} \ , \quad j \in \{1, ...p\} \ , \tag{3.4}$$

für das Optimierungsproblem

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \sigma \sum_{j=1}^{p} \lambda_{BH}(j)|\beta|_{(j)} \tag{3.5}$$

bei einer Matrix X mit orthogonalen Spaltenvektoren die FDR folgendes erfüllt:

$$\text{FDR} \ \leq \ q\frac{p_0}{p} \ \leq \ q \ .$$

$\boldsymbol{\lambda}_{BH}$ hängt nur von p ab, sodass $\boldsymbol{\lambda}_{BH}$ eine unabhängig vom Verhältnis der Dimensionen n und p monoton fallende Folge ist. Kontrolle der FDR hat zudem zur Folge, dass gleichzeitig eine Anpassung an die tatsächliche Anzahl relevanter Merkmale erfolgt.

Es sei angemerkt, dass im Falle $\beta^* = 0$, falls keine Merkmale relevant sind, die FWER (vgl. Kapitel 2.6.1) und FDR übereinstimmen.

3.2 Ableitung einer möglichen λ-Sequenz

Sind die Spalten nicht orthogonal, so kann die Sequenz $\boldsymbol{\lambda}_{BH}$ zwar trotzdem zum Einsatz kommen, die Kontrolle der FDR ist nun nicht mehr gewährleistet. Außerdem argumentieren die Autoren, dass $\boldsymbol{\lambda}_{BH}$ zu einer zu schwachen Regularisierung führen kann. Die Herleitung einer möglichen alternativen Sequenz geschieht auf der Basis der Ergebnisse für $\boldsymbol{\lambda}_{BH}$ und unter Vornahme einiger teilweise grober Abschätzungen. Das Verhältnis der Dimensionen n und p sei beliebig.

Betrachte die aus den KKT-Bedingungen für LASSO (2.18) resultierende Gleichung (2.27) für das eindeutige β_L mit $E = \text{supp}(\beta_L)$. Nehme an, dass E und $\text{supp}(\beta^*)$ und die Vorzeichen der Koeffizienten von β_L und β^* übereinstimmen. O.b.d.A. gelte $\beta_{j,L} \geq 0 \ \forall j \in \{1, ..., p\}$. Unter Vernachlässigung des Vorfaktors n ist $\beta_{E,L}$ damit durch $\beta_{E,L} = (X_E^T X_E)^{-1}(X_E^T y - \lambda \mathbb{1}_E)$ gegeben. Nehme an, dass die Sequenz der RP λ_1 bis λ_p für SLOPE gerade so gewählt wurde, dass die resultierende Lösung β_S (3.2) den Support E hat, sodass in diesem Fall das $\beta_{E,S}$ in etwa gegeben ist durch

$$\beta_{E,S} \approx \hat{\beta}_{E,S} = (X_E^T X_E)^{-1}(X_E^T y - \boldsymbol{\lambda}_E) = \beta_{OLS} - (X_E^T X_E)^{-1}\boldsymbol{\lambda}_E \ , \tag{3.6}$$

wobei $\boldsymbol{\lambda}_E = (\lambda_1, ..., \lambda_{|E|})^T$ und $\beta_{E,S} = (\beta_{(1)}, ..., \beta_{(|E|)})^T$ sind. Aus (3.6) und der Linearität des Erwartungswertes[12] resultiert

$$\mathbb{E}(\beta_E^* - \beta_{E,S}) = \mathbb{E}(\beta_E^*) - \mathbb{E}(\beta_{E,S}) \approx \ \beta_{OLS} - \beta_{OLS} + (X_E^T X_E)^{-1}\boldsymbol{\lambda}_E = (X_E^T X_E)^{-1}\boldsymbol{\lambda}_E \ ,$$

[12] Für zwei Zufallsvariablen X_i, X_j gilt $\mathbb{E}(aX_i + bX_j) = a\mathbb{E}(X_i) + b\mathbb{E}(X_j)$, $a, b \in \mathbb{R}$ beliebig.

sodass

$$\mathbb{E}\Big(X_j^T X_E(\beta_E^* - \beta_{E,S})\Big) \approx \mathbb{E}\Big(X_j^T X_E (X_E^T X_E)^{-1} \boldsymbol{\lambda}_E\Big) \ , \ j \in E \ .$$

Nehme an, dass $x_{ij} \sim \mathcal{N}(0,1)$ gilt, sodass $\mathbb{E}\Big((X_j^T X_E)^2\Big) = 1$ für $j \notin E$ folgt. Sind die Einträge der Matrix normalverteilt, so sprechen die Autoren von *Gaussian designs*. Mit dieser Annahme und der Linearität des Erwartungswertes ergibt sich

$$\mathbb{E}\Big((X_j^T X_E (X_E^T X_E)^{-1} \boldsymbol{\lambda}_E)^2\Big) = \boldsymbol{\lambda}_E^T \mathbb{E}\Big((X_E^T X_E)^{-1}\Big)\boldsymbol{\lambda}_E = \frac{\|\boldsymbol{\lambda}_E\|_2^2}{n - |E| - 1} \ . \tag{3.7}$$

Die zweite Gleichheit in (3.7) folgt daraus, dass $X_E^T X_E \in \mathbb{R}^{|E| \times |E|}$ Wishart-verteilt ist. Die Wishart-Verteilung ist die Entsprechung der Chi-Quadrat-Verteilung für Matrizen. Die Matrix $(X_E^T X_E)^{-1}$ ist damit invers Wishart-verteilt, deren Erwartungswert $\frac{1}{n-|E|-1} I_{|E|}$ ist, für Details siehe [11, 31].

Daraus lässt sich eine mögliche Sequenz $\boldsymbol{\lambda}_G \in \mathbb{R}^p$ bei Gaussian designs ableiten, wobei diese von $\boldsymbol{\lambda}_{BH}$ (3.4) abhängig ist. Die Sequenz $\boldsymbol{\lambda}_G$ wird folgendermaßen berechnet:

$$\boldsymbol{\lambda}_G(1) = \boldsymbol{\lambda}_{BH}(1), \quad \boldsymbol{\lambda}_G(j) = \boldsymbol{\lambda}_{BH}(j)\sqrt{1 + \frac{\sum_{k<j} \boldsymbol{\lambda}_G(k)^2}{n-j}}, \quad j = 2, ..., p \ . \tag{3.8}$$

Es muss sichergestellt werden, dass die Sequenz eine monoton fallende Folge darstellt, andernfalls ist f_{SLOPE} (3.1) nicht konvex. Um zu verhindern, dass die Werte der Folge ab einem bestimmten Index ansteigen, schlagen die Autoren in diesem Fall vor, die Sequenz $\hat{\boldsymbol{\lambda}}_G \in \mathbb{R}^p$ mit Einträgen

$$\hat{\boldsymbol{\lambda}}_G(j) = \begin{cases} \boldsymbol{\lambda}_G(j) \ (3.8) \ , & j \leq \hat{k} \\ \boldsymbol{\lambda}_G(\hat{k}) \ , & j > \hat{k} \end{cases} \tag{3.9}$$

einzusetzen, wobei $\hat{k}$ der Index des globalen Minimums der Folge $\boldsymbol{\lambda}_G$ ist. Somit gilt $\hat{\boldsymbol{\lambda}}_G(j) = \hat{\boldsymbol{\lambda}}_G(\hat{k}) \ \forall j > \hat{k}$. Zwar wird mittels der Definition von $\hat{\boldsymbol{\lambda}}_G$ (3.9) gesichert, dass die Folge monoton fallend bleibt, $\boldsymbol{\lambda}_G$ und damit $\hat{\boldsymbol{\lambda}}_G$ hängen allerdings von q, siehe Definition von $\boldsymbol{\lambda}_{BH}$ (3.4), und n ab. Insbesondere bei $n \ll p$ und einem kleinen q kann dies schnell zum Anstieg der Folge $\boldsymbol{\lambda}_G$ führen, sodass sogar $\hat{k} = 1$ möglich ist. Infolgedessen gilt $\hat{\boldsymbol{\lambda}}_G(j) = \hat{\boldsymbol{\lambda}}_G(k) \ \forall j, k \in \{1, ..., p\}$, sodass $\hat{\boldsymbol{\lambda}}_G$ eine konstante Folge ist und sich SLOPE damit auf LASSO reduziert. Mit der so definierten Sequenz $\hat{\boldsymbol{\lambda}}_G$ ist nun das folgende Optimierungsproblem zu lösen:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - X\beta\|_2^2 + \sigma \sum_{j=1}^{p} \hat{\boldsymbol{\lambda}}_G(j)|\beta|_{(j)} \ . \tag{3.10}$$

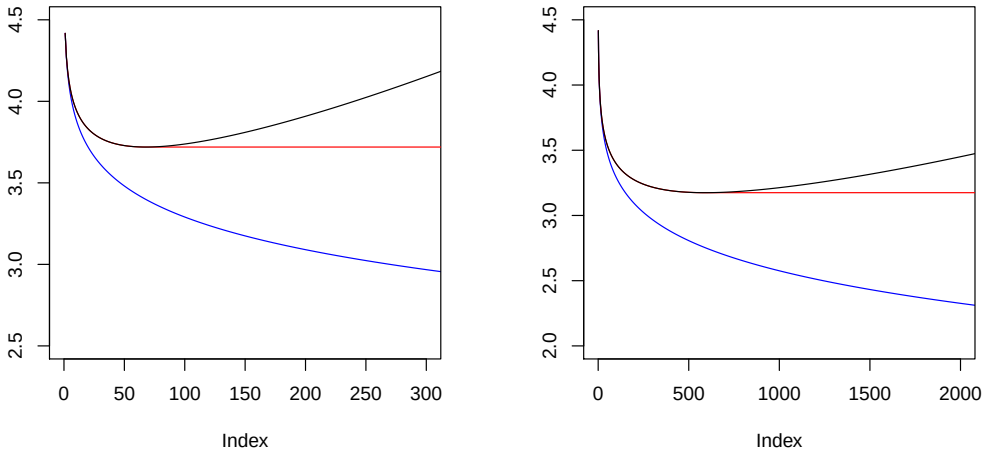


Abbildung 8 Verschiedene Sequenzen der Regularisierungsparameter für SLOPE. Sequenzen resultierend aus der BH-Prozedur (3.4) (blau), (3.8) (schwarz) und (3.9) (rot) mit $q = 0.1$ und $p = 10000$. Links wurde $n = \frac{p}{2} = 5000$, recht $n = 2p = 20000$ gesetzt. Links steigt $\boldsymbol{\lambda}_G$ ab $\hat{k} = 68$, rechts ab $\hat{k} = 589$ an.

3.3 Proximal Gradient Descent

SLOPE lässt sich wie LASSO als ein konvexes Problem mit einer konvexen differenzierbaren Zielfunktion $g(\beta) = \frac{1}{2}\|y - X\beta\|_2^2$ und einem konvexen, nicht differenzierbaren Regularisierungsterm $h(\beta) = J_{\boldsymbol{\lambda}}(\beta)$ unter anderem mit Hilfe des Proximal Gradient Descent lösen.

Betrachte dazu zunächst den Proximal Operator bei LASSO (2.44) für die Iterierte $\beta^{(k+1)}$. Für SLOPE wird lediglich $t_k\lambda\|\beta\|_1$ durch $t_k J_{\boldsymbol{\lambda}}$ ersetzt, sodass der Proximal Operator bei SLOPE die folgende Form hat:

$$\beta^{(k+1)} = \text{prox}_{t_k J_{\boldsymbol{\lambda}}}\left(\beta^{(k)} - t_k X^T(y - X\beta^{(k)})\right) \ . \tag{3.11}$$

Für die Berechnung des Proximal Operators kann $\lambda\|\beta\|_1$ allerdings nicht schlicht durch $J_{\boldsymbol{\lambda}}$ in der Gleichung (2.45) ersetzt werden. Daher ist eine sachgerechte Anpassung nötig. Denn die Aufstellung und Kontrolle der KKT-Bedingungen für SLOPE ist wegen der sortierten L1-Norm nicht trivial. Daher schlagen die Autoren eine alternative Vorgehensweise zur Lösung von (3.11) vor, siehe das nächste Kapitel 3.3.1.

Es ist zu beachten, dass in jeder Iteration die Einträge zu sortieren sind. Der rechnerische Aufwand hierfür beträgt $O(p \cdot \log(p)) \approx O(p)$ Operationen und ist somit (beinahe) linear. Im Übrigen gelten dieselben Aussagen bezüglich der Wahl der Schrittweiten $\{t_k\}_{k\in\mathbb{N}_0}$, und zwar $t_k \in (0, 2/\mu_{\max}(X^TX))\ \forall k \in \mathbb{N}_0$, bezüglich der Konvergenz (2.46) mit der Konvergenzrate $O(\frac{1}{k})$ und des Aufwandes wie im Kapitel 2.3.3 vorgestellt. Berücksichtige hierbei, dass in den entsprechenden Formeln der

Vorfaktor n bzw. $\frac{1}{n}$ wegfällt.

3.3.1 Berechnung des Proximal Operators für SLOPE

Betrachte den wegen der strengen Konvexität eindeutigen Proximal Operator von SLOPE zu einem y

$$\text{prox}_{J_{\boldsymbol{\lambda}}}(y) := \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \ \frac{1}{2}\|y - \beta\|_2^2 + \sum_{j=1}^{p} \lambda_j |\beta|_{(j)} \ . \tag{3.12}$$

Dieser Proximal Operator entspricht unter anderem der Lösung von SLOPE (3.1) mit $X = I_p$. Ist die Matrix X orthogonal, $X^T X = I_p$, so löst $\text{prox}_{J_{\boldsymbol{\lambda}}}(X^T y)$ das Optimierungsproblem SLOPE (3.1).

Die Vorzeichen der Lösung $\beta_S := \text{prox}_{J_{\boldsymbol{\lambda}}}(y)$ stimmen mit denen von y überein, $\text{sign}(\beta_{j,S}) = \text{sign}(y_j) \ \forall j \in \{1, ..., p\}$. Somit ist es von Vorteil, die Vorzeichen von y zu speichern, den Proximal Operator für $|y|$ zu bestimmen und anschließend die Vorzeichen anzupassen. Sei P eine Permutationsmatrix, die die Einträge von $|y|$ sortiert, sodass der Proximal Operator (3.12) mit $P|y|$ statt $|y|$ gelöst wird, und zwar $\text{prox}_{J_{\boldsymbol{\lambda}}}(P|y|)$. Der ursprüngliche Proximal Operator (3.12) lässt sich anschließend als $\beta_S = \text{prox}_{J_{\boldsymbol{\lambda}}}(y) = \text{sign}(y) P^{-1} \text{prox}_{J_{\boldsymbol{\lambda}}}(P|y|)$ berechnen, wobei $\text{sign}(y) = (\text{sign}(y_1), ..., \text{sign}(y_p))^T$ ist. Damit kann o.b.d.A. die Annahme $y_1 \geq ... \geq y_p$ getroffen werden. Aufgrund dessen gilt ebenso $\beta_{1,S} \geq ... \geq \beta_{p,S} \geq 0$. Folglich ist β_S ebenfalls die Lösung des quadratischen Programms (QP)

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|y - \beta\|_2^2 + \sum_{j=1}^{p} \lambda_j \beta_j, \quad \text{s.t.} \ \ \beta_1 \geq ... \geq \beta_p \geq 0 \ , \tag{3.13}$$

sodass die Optimierungsprobleme (3.12) und (3.13) äquivalent sind, für den Beweis siehe Anhang A8.

Die Herleitung dieser Äquivalenz dient vor allem dazu, sich von der sortierten L1-Norm zu lösen. Die Autoren haben gezeigt, dass der Proximal Operator (3.12) durch $\text{prox}_{J_{\boldsymbol{\lambda}}}(y) = (y - \boldsymbol{\lambda})_+$ gegeben ist, falls $(y - \boldsymbol{\lambda})_+$ monoton fallend ist. Zum Beweis dieser Aussage können aufgrund der Äquivalenz der Optimierungsprobleme (3.12) und (3.13) die KKT-Bedingungen des QP herangezogen werden, für den Beweis siehe [5]. Daraus lässt sich der Algorithmus 8 ableiten, dessen Richtigkeit ebenfalls die KKT-Bedingungen des QP sicherstellen.

Fast Proximal SLOPE weist eine wichtige Eigenschaft auf, und zwar

$$\text{prox}_{J_{\boldsymbol{\lambda}}}(y) = \text{prox}_{J_{\boldsymbol{\lambda}^{(+)}}}(y^{(+)}) \ ,$$

Algorithmus 8 Fast Proximal SLOPE

1: $y = (y_1, ..., y_p)^T$ mit $y_1 \geq ... \geq y_p$, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_p)^T$ mit $\lambda_1 \geq ... \geq \lambda_p$.
2: **while** $y - \boldsymbol{\lambda}$ monoton steigend **do**
3: Finde Indizes $j_1 < ... < j_2$: $y_{j_1} - \lambda_{j_1} < y_{j_1+1} - \lambda_{j_1+1} < ... < y_{j_2} - \lambda_{j_2}$.
4: **for** $k = j_1 : j_2$ **do**

$$y_k = \frac{1}{j_2 - j_1 + 1} \sum_{j_1 \leq k \leq j_2} y_k \ , \qquad \lambda_k = \frac{1}{j_2 - j_1 + 1} \sum_{j_1 \leq k \leq j_2} \lambda_k \ .$$

5: **end for**
6: **end while**
7: $\beta_S = (y - \boldsymbol{\lambda})_+$ ist die resultierende Lösung, $J = \text{supp}(\beta_S)$ die Menge der selektierten Variablen.

wobei $y^{(+)}$ und $\lambda^{(+)}$ die in jeder Iteration nach Schritt 4 berechneten aktualisierten Werte für die anfänglichen y und λ sind. Dies bedeutet, dass die Lösung $\beta_S = \text{prox}_{J_{\boldsymbol{\lambda}}}(y)$ für alle im Laufe des Algorithmus bestimmten $y^{(+)}$ und $\lambda^{(+)}$ identisch ist. Diese Eigenschaft stellt sicher, dass der Algorithmus nach einer endlichen Anzahl an Iterationen terminiert und tatsächlich die Lösung β_S berechnet.

Das Fast Proximal SLOPE ist in dem Algorithmus 8 in seiner einfachsten Form dargestellt. Wegen der Abhängigkeit von y_k und λ_k von allen vorher bestimmten Koeffizienten, müssen alle im Laufe des Algorithmus berechneten y und λ gespeichert werden. Dies kann die Komplexität des Algorithmus je nach Größenordnung von p stark erhöhen. Es kann allerdings gezeigt werden, dass der Proximal Operator für alle Sequenzen der Form $y_{j_1} - \lambda_{j_2} \leq ... \leq y_{j_1} - \lambda_{j_2}$, wie im Schritt 3 des Algorithmus 8, konstant ist, sodass $\beta_{i,S} = \beta_{j,S}\ \forall i, j \in \{j_1, ..., j_p\}$. Dies ermöglicht eine stackbasierte (engl. stack-based) Implementierung des Fast Proximal SLOPE, siehe Algorithmus 9.

Der Algorithmus verwendet die Notation eines Tupels als $(a, b)_i = (c, d)$ für $a_i = c,\ b_i = d$. In den Schritten 3 bis 11 werden insgesamt p Tupel $\{(i, j, s, w)\}_{t=1}^{p}$ erstellt, wobei jedes Tupel $(i, j, s, w)_t$ in das nachfolgende $(i, j, s, w)_{t+1}$ höchstens einmal integriert wird. Aufgrund dessen weist der Algorithmus eine Komplexität von $O(p)$ auf.

3.3.2 Accelerated Proximal Gradient Descent

Da die Schrittweiten von dem größten Eigenwert $\mu_{\max}(X^T X)$ von $X^T X$ abhängig sind, und sich dieser bei hohen Dimensionen selten ohne großen Aufwand bestimmen lässt, haben Beck und Teboulle (2009) [2] die sogenannte *backtracking stepsize rule* hergeleitet, sodass in der Iteration k die Schrittweite t_k als eine Approximation

Algorithmus 9 Stack-based Algorithm für Fast Proximal SLOPE

1: $y = (y_1, ..., y_p)^T$ mit $y_1 \geq ... \geq y_p$, $\lambda = (\lambda_1, ..., \lambda_p)^T$ mit $\lambda_1 \geq ... \geq \lambda_p$.
2: $t = 0$.
3: **for** $k = 1 : p$ **do**
4: $\quad t = t + 1$.
5: $\quad (i, j, s, w)_t = (k, k, y_i - \lambda - i, (y_i - \lambda_i)_+)$.
6: $\quad$ **while** $t > 1$ && $w_{t-1} \leq w_t$ **do**
7: $\quad\quad (i, j, s, w)_{t-1} = \left(i_{t-1}, j_t, s_{t-1} + s_t, (\frac{j_{t-1} - i_t + 1}{j_t - i_{t-1} + 1} s_{t-1} + \frac{j_t - i_t + 1}{j_t - i_{t-1} + 1} s_t)_+\right)$.
8: $\quad\quad$ Lösche $(i, j, s, w)_t$.
9: $\quad\quad t = t - 1$.
10: $\quad$ **end while**
11: **end for**
12: **for** $l = 1 : t$ **do**
13: $\quad$ **for** $k = i_l : j_l$ **do**
14: $\quad\quad \beta_{k,S} = w_l$.
15: $\quad$ **end for**
16: **end for**
17: β_S ist die resultierende Lösung, $J = \text{supp}(\beta_S)$ die Menge der selektierten Variablen.

der Lipschitz-Konstante L von $\frac{1}{2}\|y - X\beta\|_2^2$ (vgl. Anhang A4) bestimmt wird. Für diese Modifikation des Algorithmus 9 wird der Startwert für $t_0 = 0$ gesetzt und ein zusätzlicher Zwischenschritt vor der Berechnung von $\beta^{(k+1)}$ eingefügt. Da diese Regel zur Approximation von L einige weitere Definitionen und Aussagen benötigt, ist deren Vorstellung kein Bestandteil dieser Arbeit.

Es ist möglich, durch leichte Abänderung des Proximal Gradient Descent eine Konvergenz von $O(\frac{1}{k^2})$ zu erreichen [28, 2][13]. Dazu werden zwei zusätzliche Größen, eine Folge von Vektoren $\{a_k\}_{k \in \mathbb{N}_0}$ und eine Folge von Skalaren $\{\theta_k\}_{k \in \mathbb{N}_0}$ benötigt. Die Vorgehensweise zeigt Algorithmus 10.

Der Unterschied zum Proximal Gradient Descent liegt vor allem darin, im Schritt 7 des Algorithmus 10 in der Iteration $k + 1$ den Proximal Operator nicht in Abhängigkeit des aus der Iteration k resultierenden $\beta^{(k)}$, sondern einer bestimmten Linearkombination der letzten beiden Iterierten $\beta^{(k-1)}$ und $\beta^{(k)}$ (Schritt 9 des Algorithmus 10) zu berechnen.

Falls die Lipschitz-Konstante nicht abgeschätzt werden soll, weil diese bspw. vorliegt oder schnell berechnet werden kann, so wird Schritt 4 des Algorithmus 10 nicht durchgeführt und es gilt $\theta_k = t_k \; \forall k \in \mathbb{N}_0$.

Für die Differenz von f_{SLOPE} ausgewertet in der aktuellen Iterierten $\beta^{(k)}$ zu dem

[13] Dies gilt ebenso für LASSO.

Algorithmus 10 Accelerated Proximal Gradient Descent für SLOPE

1: X, y, $\lambda_1 \geq ... \geq \lambda_p$, $tol \geq 0$ Fehlertoleranzgrenze.
2: $k = 0$, $\beta^{(0)}$ Startlösung, $a^{(0)} = \beta^{(0)}$, $t_0 = 1$, $\theta_0 = 0$.
3: **repeat**
4: Bestimme t_k mittels backtracking stepsize rule.
5: Berechne $b = a^{(k)} - t_k X^T(y - Xa^{(k)})$.
6: Sortiere die Einträge von b, sodass $b_1 \geq .... \geq b_p$.
7: Berechne $\beta^{(k+1)} = \text{prox}_{t_k J_{\boldsymbol{\lambda}}}(b)$ mittels Algorithmus 9.
8: $\theta_{k+1} = \frac{1}{2}(1 + \sqrt{1 + 4/\theta_k^2})$.
9: $a^{(k+1)} = \beta^{(k+1)} + \frac{(\theta_{k-1}-1)}{\theta_k}(\beta^{(k+1)} - \beta^{(k)})$.
10: $k = k + 1$.
11: **until** convergence.
12: $\hat{\beta} := \beta^{(k)}$ ist die resultierende Lösung, $J = \text{supp}(\hat{\beta})$ die Menge der selektierten Variablen.

globalen optimalen Funktionswert von f_{SLOPE} gilt die Abschätzung [2]

$$|f_{SLOPE}(\beta^{(k)}) - f_{SLOPE}(\beta_S)| \leq \frac{2L\|\beta^{(0)} - \beta_S\|_2^2}{(k+1)^2} \quad \forall \beta_S \in \text{argmin } f_{SLOPE} \ .$$

Für die Herleitung und den Beweis, dass dieser Algorithmus zum einen konvergiert und zum anderen eine Konvergenzrate von $O(\frac{1}{k^2})$ aufweist, sei auf [28] verwiesen.

Bei der Betrachtung der Optimierungsprobleme (3.5) und (3.10) fällt auf, dass zur Berechnung einer geeigneten $\boldsymbol{\lambda}$-Sequenz die Standardabweichung σ des Fehlerterms ϵ erforderlich ist. σ ist jedoch selten bekannt und muss daher approximiert werden. Falls $n \geq p$ gilt, ist eine Schätzung von σ mit Hilfe der Stichprobenvarianz möglich. Der Algorithmus 11 bildet das Vorgehen bei einem unbekannten σ ab. Der Algorithmus startet mit $\text{RSS} = \sigma^{(1)} = \|y\|_2^2$ als Approximation der tatsächlichen Standardabweichung σ. Diese ist eine recht grobe Approximation, deren Wert im Normalfall zu hoch ist. Daher wird die Schätzung im Laufe des Algorithmus verbessert, um dadurch eine umso genauere Näherungslösung für β_S zu erhalten. Schritt 7 des Algorithmus 11 macht jedoch deutlich, dass σ nur dann geschätzt werden kann, wenn für den Support J der aktuellen Approximation der Lösung $|J| < n - 1$ gilt. Das mit SLOPE zu bestimmende Modell erlaubt folglich maximal $n - 2$ Merkmale, was die Anzahl relevanter Merkmale bei $n \ll p$ stark unterschätzen kann. Der Algorithmus bricht entweder ab, sobald der Support zweier aufeinander folgenden Lösungen übereinstimmt, oder wenn in der Iteration k $|J| \leq n-1$ gilt. Denn in dem zweiten Fall sind weitere Berechnungen nicht mehr möglich.

Algorithmus 11 Algorithmus für SLOPE bei einem unbekannten σ

X, y, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_p)^T$ Sequenz berechnet mittels (3.9), *tol* Fehlertoleranzgrenze.
$k = 0$, $\sigma^{(0)} = 1$, $J_+ = \emptyset$.
repeat
    $k = k + 1$.
    $J = J_+$.
    Berechne RSS resultierend aus der Lösung linearer Regression unter Verwendung der in J enthaltenen Variablen mit Output y:
    $\hat{b} \in \underset{b \in \mathbb{R}^{|J|}}{\operatorname{argmin}} \|X_J b - y\|_2^2$, RSS $= \|X\hat{b} - y\|_2^2$.
    $\sigma^{(k)} = \text{RSS}/(n - |J| - 1)$.
    Berechne $\hat{\beta}$ mit der Sequenz $\sigma^{(k)}\boldsymbol{\lambda}$ mittels Algorithmus 10.
    $J_+ = \text{supp}(\hat{\beta})$.
until $J = J_+$.
$\hat{\beta}$ ist die resultierende Lösung, $J = \text{supp}(\hat{\beta})$ die Menge der selektierten Variablen.

3.3.3 Dualitätslücke als Abbruchkriterium

Anstatt die Norm der Differenz zweier aufeinanderfolgenden Iterierten $\beta^{(k)}$ und $\beta^{(k+1)}$ für den Abbruch des Lösungsalgorithmus zu verwenden, schlagen die Autoren den Einsatz der Dualitätslücke vor. Definiere $r := y - X\beta$, sodass SLOPE formuliert werden kann als

$$\min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \frac{1}{2}\|r\|_2^2 + J_{\boldsymbol{\lambda}} \text{ s.t. } \{ r = y - X\beta \iff r - y + X\beta = 0 \} .$$

Aufgrund der Äquivalenz eines Optimierungsproblems mit Nebenbedingungen zur Lagrange-Dualität, siehe Anhang A3, existiert ein eindeutiger Lagrange-Multiplikator $\theta \in \mathbb{R}^n$ zur Nebenbedingung $r - y + X\beta = 0$, sodass (3.14) sich umschreiben lässt als

$$\min_{\beta \in \mathbb{R}^p, r \in \mathbb{R}^n} \Big\{ f(\beta, r, \theta) := \frac{1}{2}\|r\|_2^2 + J_{\boldsymbol{\lambda}} - \theta^T(r - y + X\beta) = \frac{1}{2}\|r\|_2^2 - \theta^T r - \theta^T X\beta + J_{\boldsymbol{\lambda}} + \theta^T y \Big\} . \tag{3.14}$$

$\theta^T y$ spielt bei der Minimierung keine Rolle. Es ist ersichtlich, dass $f(\beta, r, \theta)$ bezüglich β und r unabhängig voneinander minimiert werden kann. Die optimalen Lösungen

können sofort bestimmt werden:

$$\min_{\beta \in \mathbb{R}^p} \; -\theta^T X\beta + J_{\boldsymbol{\lambda}} = \begin{cases} 0, & \theta \in C_{\boldsymbol{\lambda}} \\ -\infty, & \text{sonst} \ , \end{cases} \tag{3.15}$$

$$\min_{r \in \mathbb{R}^n} \; \frac{1}{2}\|r\|_2^2 - \theta^T r \; = \; \theta^T\theta \quad \text{mit } r = \theta \ , \tag{3.16}$$

wobei

$$\theta \in C_{\boldsymbol{\lambda}} \quad :\Longleftrightarrow \quad \sum_{i \le j} |\theta|_{(i)} \le \sum_{i \le j} \lambda_i \ \forall j \in \{1, ..., p\} \ .$$

Nach Einsetzen von (3.15) und (3.16) in die Optimierungsfunktion (3.14) ergibt sich das duale Problem zu SLOPE als

$$\max_{\theta \in \mathbb{R}^n} \left\{ f^D_{SLOPE}(\theta) := -\frac{1}{2}\|\theta\|_2^2 + \theta^T y \right\} \quad \text{s.t.} \;\; X^T\theta \in C_{\boldsymbol{\lambda}} \ .$$

Es gilt starke Dualität, sodass das Minimum des primalen mit dem Maximum des duales Problems übereinstimmt, siehe Anhang A3. Es gilt somit $f_{SLOPE}(\beta_S) = f^D_{SLOPE}(\theta^*)$, wobei $\theta^* \in \text{argmax}\ f^D_{SLOPE}(\theta)$ die Lösung des dualen Problems ist. Daraus lassen sich Abbruchkriterien ableiten. Laut (3.16) gilt $\theta^* = r = y - X\beta_S$, sodass sich $\hat{\theta} = r = y - X\hat{\beta}$ als Näherungslösung zu θ^* anbietet, wobei $\hat{\beta}$ eine Approximation von β_S ist. Die Differenz von $f_{SLOPE}(\hat{\beta})$ und $f^D_{SLOPE}(\hat{\theta})$ ergibt

$$\delta(\hat{\beta}) = (X\hat{\beta})^T(X\hat{\beta} - y) + J_{\boldsymbol{\lambda}} \tag{3.17}$$

als Dualitätslücke in $\hat{\beta}$. Das Erreichen einer Fehlertoleranzgrenze für $\delta(\beta)$ als Abbruchkriterium ist noch nicht ausreichend, da zudem gewährleistet werden muss, dass $\hat{\theta} \in C_{\boldsymbol{\lambda}}$ gilt und $\hat{\theta}$ somit für das duale Problem zulässig ist. Dazu wird eine zusätzliche Schranke eingeführt, die einen gewissen Grad an Unzulässigkeit von $\hat{\theta}$ erlaubt:

$$\text{unzul}(\theta) = \max\Big\{0, \; \max_j \sum_{i \le j} (|\hat{\theta}|_{(i)} - \lambda_i)\Big\}$$

Ein Algorithmus bricht folglich genau dann ab, wenn sowohl $\delta(\hat{\beta})$ (3.17) als auch unzul(θ) die vorgegebene(n) Fehlertoleranzgrenze(n) erfüllen, oder die Anzahl der selektierten Variablen $n - 2$ überschreitet, falls der Algorithmus 11 zum Einsatz kommt.

4 Praktische Umsetzung in der Programmiersprache R

Dieses Kapitel besteht in der Anwendung der R-Pakete **glmnet** [16] für LASSO und **SLOPE** [6] für SLOPE sowohl auf synthetische als auch auf Genexpressionen von CAMDA Patienten mit Neuroblastom und Genexpressionen von TCGA Patienten. Die CAMDA-Daten sind unter `http://camda2017.bioinf.jku.at/doku.php/contest_dataset` verfügbar. Um diese einzusehen und herunterladen zu können, muss ein Account angelegt werden. Unter dieser URL sind außerdem Informationen über CAMDA zu finden. Für Informationen zu TCGA siehe `https://cancergenome.nih.gov/` und Zugang zu TCGA-Daten `https://portal.gdc.cancer.gov/`. Das Portal stellt einige Möglichkeiten zur Verfügung, die Daten herunterzuladen. Seit 15.07.2016 sind die TCGA-Daten auf diesem Portal gespeichert. Vorher erfolgte der Zugriff über das Portal `https://tcga-data.nci.nih.gov/docs/publications/tcga/?`, das einen R-Client hatte. Dieser kann dennoch weiterhin verwendet werden. Unter anderem mittels des R-Pakets **EasyTCGA**[14] [24] können die gewünschten Genexpressionsdaten als eine einzige Matrix für alle Patienten eines Krebstyps und alle verfügbaren Gene in der gewünschten Form wie in der Abbildung 4 dargestellt heruntergeladen werden. Zudem stellt **EasyTCGA** die wichtigsten Eingabedaten zur Verfügung. Diese sind die Barcodes der Patienten und die Namen der Gene, sodass der Anwender diese nicht im Vorhinein auf den angegebenen Portalen heraussuchen muss. Der Nachteil besteht darin, dass diese Datensätze keine Daten enthalten, die seit 15.07.2016 aktualisiert und hinzugefügt wurden.

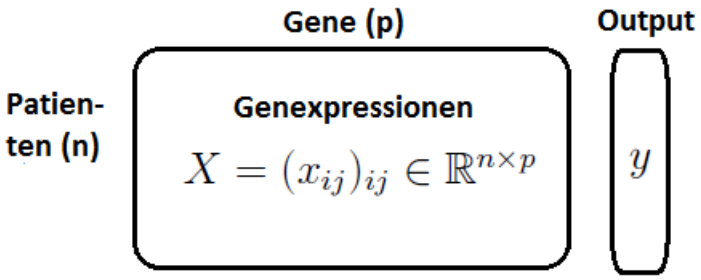


Abbildung 9 Genexpressionsmatrix. Die Zeilen stellen die Patienten und die Spalten die Gene dar. Somit entspricht n der Anzahl der Patienten und p der Anzahl der Gene. Der Eintrag x_{ij} drückt die Genexpression des Gens j für den Patienten i aus, $i \in \{1, ..., n\}$, $j \in \{1, ...p\}$.

Die R-Pakete **glmnet** und **SLOPE** sind auf GitHub verfügbar und können mit den folgenden Befehlen heruntergeladen werden:

```
> devtools::install_github("cran/glmnet")
> devtools::install_github("cran/SLOPE")
```

[14] `https://github.com/sanglee/EasyTCGA`

Keine der Matrizen der realen Datensätze hat mehr Zeilen als Spalten, die Anzahl der Beobachtungen (Patienten) ist folglich immer geringer als die der Merkmale (Gene). Somit weisen die Matrizen keine der gewünschten Eigenschaften wie einen vollen Rang oder gar Orthogonalität auf. Die Genexpressionen der TCGA-Datensätze enthalten allerdings fehlende Werte, sodass nach deren Bereinigung und Imputation die Anzahl der Patienten die der Gene übersteigen kann.

Die Schätzung der Verteilung ist vor allem bei $n \ll p$ kaum möglich. Weder die Verteilung der Matrixeinträge noch die des Outputs der Genexpressionen sind bekannt. Darüber, welche Gene relevant sind, und über die Größenordnung derer Anzahl gibt es ebenso keine Informationen. Deswegen können keine Aussagen über die FWER und die FDR getroffen werden. Um das Verhalten anhand realer Daten dennoch beurteilen zu können, kommt CV zum Einsatz, und zwar je nach Beispiel sowohl unter Einsatz aller Beobachtungen gleichzeitig als Trainings- und Testdatensatz als auch deren Aufteilung in N möglichst gleichgroße Teildatensätze, sodass jeweils $N-1$ Teildatensätze zum Trainieren und einer zum Testen verwendet wird.

In dieser Arbeit wird bei CAMDA der Datensatz der RNA-Seq Genexpressionen untersucht. Diese Genexpressionen sind unglücklicherweise verrauscht. Die Genexpressionsmatrix umfasst $n = 498$ Patienten und $p = 60249$ Gene. Diese Matrix wird im Folgenden als `CAMDA` definiert. Bei den TCGA-Daten werden microRNA Genexpressionen von Brustkrebspatienten (Breast Invasive Carcinoma, BRCA) herangezogen. Die microRNA Genexpressionsmatrix umfasst $n = 1097$ Patienten und $p = 2588$ Gene, wobei diese viele fehlende Werte aufweisen. Daher werden zunächst alle Zeilen und Spalten, bei denen mehr als die Hälfe der Einträge aus fehlenden Werten besteht, eliminiert. Die resultierende Matrix ist von der Dimension 755×552, hat somit mehr Zeilen als Spalten. Die restlichen fehlenden Werte werden mittels Imputation vervollständigt. Dazu wird das R-Paket **impute** eingesetzt [20]. Dieses kann folgendermaßen heruntergeladen werden:

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("impute")
```

Die nach der Imputation resultierende Matrix, die im Folgenden als `TCGA` bezeichnet wird, hat einen vollen Rang, woraus die Eindeutigkeit der Lösungen von LASSO für jedes beliebige $\lambda \geq 0$ und SLOPE für jede beliebige absteigende Folge $\boldsymbol{\lambda}$ folgt.

Da für SLOPE bislang keine Transformation auf generalisierte lineare Modelle vorgenommen wurde, lassen sich LASSO und SLOPE nur im Falle der klassischen linearen Regression mit einem stetigen Output y vergleichen. Bei den realen Datensätzen kann der Vergleich von LASSO und SLOPE daher lediglich anhand der Überlebenszeiten, die als normalverteilt angenommen werden, stattfinden. Die Analyse anhand aller Patienten hätte jedoch zur Folge, dass die Censoring Time aller zum

Ende der Studie lebenden Patienten deren Überlebenszeit gleichgesetzt würde. Daher können nur die Genprofile der während der Studie an der Krankheit gestorbenen Patienten verwendet werden, sodass die jeweiligen Datensätze auf diese Patienten reduziert werden müssen. Dadurch sinkt die Anzahl der für die Analyse verfügbaren Beobachtungen in den Datensätzen erheblich. Die Verteilung des Vitalstatus der Patienten ist dabei ungleichmäßig, siehe Tabelle 1.

Datensatz	n	**Vitalstatus = 1**	**Vitalstatus = 0**
`TCGA`	775	106	649
`CAMDA`	498	393	105

Tabelle 1 Verteilung des Vitalstatus von CAMDA Patienten und TCGA Brustkrebspatienten des microRNA Datensatzes. Vitalstatus = 1 bedeutet, dass der Patient während der Studie an der Krankheit gestorben ist, sonst gilt Vitalstatus = 0.

Die Berechnung eines Lösungsvektors für SLOPE bei CAMDA Genexpressionen nahm für verschiedene Sequenzen der RP viel Zeit in Anspruch, wobei nach einigen Stunden Rechenzeit die Fehlermeldung kam, dass mehr als $n - 2$ Merkmale selektiert wurden und zudem die Anzahl der Iterationen erreicht ist. Die Änderung der Eingabeparameter hat das Problem nicht gelöst. Aufgrund dessen und der damit verbundenen Problemen von RStudio wird für den Vergleich von LASSO und SLOPE die Genexpressionsmatrix `CAMDA` auf 1000 Gene mit der höchsten Standardabweichung reduziert, sodass $p = 1000$ folgt. Aus biologischer Hinsicht ist dies allerdings weniger korrekt, unter anderem, weil die Daten von CAMDA verrauscht sind.

Der Aufbau dieses Kapitels ist folgend:

- Kapitel 4.1 behandelt LASSO in Verbindung mit den im Kapitel 2.4 vorgestellten GLM:
  1. Logistische Regression als Klassifikation mit `CAMDA` & `TCGA` und Vitalstatus als Zielvariable.
  2. Logistische Regression als Klassifikation mit `CAMDA` und Alter der CAMDA Patienten als Zielvariable.
  3. Multinomiale Regression als Klassifikation mit `CAMDA` & `TCGA` und Stadium der Patienten als Zielvariable.
  4. Cox-Regression mit `CAMDA` & `TCGA` und Überlebenswahrscheinlichkeit als Zielvariable.
- Kapitel 4.2 behandelt LASSO und SLOPE:
  1. Kapitel 4.2.1: lineare Regression mit synthetischen Daten mit einer orthogonalen Systemmatrix, diese wird im Folgenden als `X.orth` bezeichnet.

2. Kapitel 4.2.1: lineare Regression mit synthetischen Daten, wobei die Einträge der Matrix $x_{ij} \sim \mathcal{N}(0, \frac{1}{n})$ (gaussian designs) erfüllen, diese Matrix wird im Folgenden als `X.gauss` bezeichnet.
3. Kapitel 4.2.2: lineare Regression mit den auf die gestorbenen Patienten reduzierten Matrizen der CAMDA und TCGA microRNA Genexpressionen mit Überlebenszeit in Tagen als Zielvariable, diese Matrizen werden als `CAMDA_dead` und `TCGA_dead` bezeichnet. Die Matrix `CAMDA_dead` wird zusätzlich auf 1000 Gene mit der höchsten Standardabweichung reduziert.

Die Tabelle 2 veranschaulicht die Vorgehensweise.

Kapitel	**Datensatz**	$\boldsymbol{n}$	$\boldsymbol{p}$	**Methode**	**Regression**	**Zielvariable**
4.1.1	`CAMDA`	498	60249	LASSO	log. Reg.	Vitalstatus
4.1.1	`TCGA`	755	552	LASSO	log. Reg.	Vitalstatus
4.1.1	`CAMDA`	498	60249	LASSO	log. Reg.	Alter
4.1.1	`CAMDA`	498	60249	LASSO	mult. Reg.	Stadium
4.1.1	`TCGA`	755	552	LASSO	mult. Reg.	Stadium
4.1.2	`CAMDA`	498	60249	LASSO	Cox-Reg.	Überlebenswahrs.
4.1.2	`TCGA`	755	552	LASSO	Cox-Reg.	Überlebenswahrs.
4.2.1	`X.orth`	1000	1000	LASSO	lin. Reg.	$y \in \mathbb{R}$
4.2.1	`X.orth`	1000	1000	SLOPE	lin. Reg.	$y \in \mathbb{R}$
4.2.1	`X.gauss`	1000	5000	LASSO	lin. Reg.	$y \in \mathbb{R}$
4.2.1	`X.gauss`	1000	5000	SLOPE	lin. Reg.	$y \in \mathbb{R}$
4.2.2	`CAMDA_dead`	393	1000	LASSO	lin. Reg.	# Tage
4.2.2	`CAMDA_dead`	393	1000	SLOPE	lin. Reg.	# Tage
4.2.2	`TCGA_dead`	106	552	LASSO	lin. Reg.	# Tage
4.2.2	`TCGA_dead`	106	552	SLOPE	lin. Reg.	# Tage

Tabelle 2 Lernaufgaben und Datensätze für die praktische Analyse.

Die Rechenzeit wird folgendermaßen bestimmt:

```
> start = proc.time()
> ...
> proc.time() - start
```

Dabei kann sich bei SLOPE die Laufzeit je nach verwendeter Sequenz der RP aufgrund der Sortierung von Einträgen unterscheiden, falls sich bspw. bei einer Sequenz alle RP, $\lambda_1 > ... > \lambda_p$, und bei einer anderen nur wenige Einträge bis zu einem gewissen Index j unterscheiden, $\lambda_1 > ... > \lambda_j = ... = \lambda_p$. Außerdem ist zu beachten, dass die Laufzeit für dieselben Berechnungen, ob bei LASSO oder SLOPE, verschieden ausfallen kann, was unter anderem durch den vorhandenen Speicherplatz

oder die verwendete API (Application Programming Interface) bedingt sein kann. Daher wird die nach 10-facher Anwendung derselben Berechnungen, teilweise zu unterschiedlichen Tageszeiten, durchschnittliche Laufzeit (in Sekunden) angegeben.

4.1 Anwendung von glmnet auf generalisierte lineare Modelle

Das R-Paket **glmnet** setzt Elastic Net, vgl. Kapitel 2.1, somit insbesondere LASSO, sowohl für die klassische lineare Regression als auch für alle im Kapitel 2.4 vorgestellten GLM um. Als Lösungsalgorithmus kommt dabei das CD bzw. das Pathwise CD zum Einsatz, siehe Kapitel 2.3.2 und 2.4.

Die wichtigsten Funktionen von **glmnet** sind `glmnet` und `cv.glmnet`. `glmnet` wendet das Pathwise CD an. Dabei ist `alpha` ein Eingabewert der Funktionen und dieser ist als `alpha=1` voreingestellt, sodass ohne dessen Änderung LASSO gelöst wird. Es wird für 100 verschiedene RP $\lambda_1, ..., \lambda_{100}$, die im Laufe des Pathwise CD bestimmt werden, jeweils eine Lösung berechnet, wodurch ein Lösungspfad für jeden Koeffizienten entsteht. Bei der Anwendung von `cv.glmnet` findet zusätzlich für jedes λ_l, $l \in \{1, ..., 100\}$, 10-fache CV statt, sodass (2.86) für die klassische lineare Regression und (2.87) für das jeweilige GLM berechnet werden. Das R-Paket bietet noch weitere Möglichkeiten als Gütekriterium für die CV an. Die Anzahl der RP m und der für die CV zu verwendenden Teilmengen N kann der Anwender ebenso selbst bestimmen. Die maximale Anzahl an Teilmengen ist n, sodass es sich in diesem Fall um LOOCV handelt. Um die Regressionskoeffizienten des Lösungsvektors $\hat{\beta}(\lambda)$ als Approximation von $\beta_L(\lambda)$ (2.7) zu einem $\lambda \geq 0$ aus einem mittels `glmnet` oder `cv.glmnet` berechneten Objekt zu extrahieren, stellt das R-Paket die Funktion `coef` zur Verfügung. Für Vorhersagen wird die Funktion `predict` verwendet. `glmnet` bzw. `cv.glmnet` standardisieren die Systemmatrix X und den Output y, geben die Koeffizienten des Lösungsvektors jedoch auf der Originalskala zurück. Für Details siehe [16].

Aus der CV gehen zwei bestimmte RP hervor, `lambda.min` und `lambda.1se`, diese werden in den folgenden Tabellen als $\lambda_{\min}$ und $\lambda_{1\text{se}}$ bezeichnet. Das `lambda.min` entspricht dem RP mit dem kleinsten gemittelten CV-Fehler (2.88), der dem Vorhersagefehler entspricht, siehe Kapitel 2.6.3. Wie im Kapitel 2.6 erklärt, ist ein geringer Fehler nicht mit optimaler Variablenselektion gleichzusetzen. Das `lambda.min` ist zudem in der Regel zu klein, sodass zu viele Merkmale selektiert werden und damit die Gefahr besteht, dass sich darunter viele oder gar nur falsche Merkmale befinden. Der RP `lambda.1se` entspricht dem RP, aus dem das am meisten regularisierte Modell hervorgeht, sodass dessen CV-Fehler in etwa einen Standardfehler (auch Stichprobenfehler) von dem minimalen CV-Fehler entfernt ist. In der Regel gilt `lambda.1se`

> `lambda.min`, sodass für `lambda.1se` weniger Variablen ausgewählt werden, in einigen Anwendungen stimmen `lambda.1se` und `lambda.min` überein. Im Kapitel 2.3.2 wurde gezeigt, dass jede Variable, die im Laufe des Pathwise CD zu einem λ_l ins Modell eingetreten ist, bis zum Ende im Modell verbleibt. Dies bedeutet, dass die zu `lambda.1se` selektierten Merkmale eine Teilmenge der zu `lambda.min` ausgewählten Merkmale sind.

Die mit `glmnet` bzw. `cv.glmnet` berechneten Objekte können mittels `plot` graphisch dargestellt werden. Oberhalb der Graphiken wird die Anzahl der Nichtnulleinträge des Koeffizientenvektors in Abhängigkeit von λ abgebildet. Der Plot des mittels `glmnet` berechneten Objektes zeigt den Pfad der einzelnen Koeffizienten in Abhängigkeit von λ. Der CV Plot des mittels `cv.glmnet` berechneten Objektes stellt die CV-Kurve in Abhängigkeit von λ dar, wobei zwei vertikale Linien `lambda.min` und `lambda.1se` markieren.

glmnet wendet die im Kapitel 2.5 vorgestellte sequentielle Strong Rule (2.79) an, die sowohl zu Anfang als auch im Laufe des Algorithmus vor der Berechnung einer Lösung zu λ_l, $l \in \{1, ..., m\}$, a priori eine hohe Anzahl an Merkmalen (Spalten der Datenmatrix) eliminiert. Dies kann die Dimension p (stark) reduzieren und dadurch die Effizienz deutlich steigern, was bei $n \ll p$ von großer Bedeutung ist. Dies spiegelt sich in der Rechenzeit wider. Die Dimensionsreduktion kann zudem dazu führen, dass die reduzierte Datenmatrix mehr Zeilen (Beobachtungen) als Spalten (Merkmale) aufweist und dadurch Eigenschaften wie bspw. einen vollen Rang besitzt.

Für die Darstellung der jeweiligen Ergebnisse in Tabellen werden alle Größen, falls nötig, auf drei Nachkommastellen gerundet.

4.1.1 Logistische und multinomiale Regression

Die in diesem Unterkapitel ausführlich vorgestellte Vorgehensweise von **glmnet** ist beispielgebend für den Rest dieses Kapitels.

Der Output y sei diskret mit $K = 2$ oder $K > 2$ Ausprägungen. O.b.d.A sei $y \in \{0, 1\}$ bei einer binären und $y \in \{1, ..., K\}$ bei einer multinomialen Outputvariable. Die Ausprägungen können beliebig gesetzt werden.

Ein oft auftretendes Problem bei der logistischen und multinomialen Regression ist die ungleichmäßige Verteilung der Klassen. Liegt bspw. ein Problem mit zwei Klassen vor, bei dem die Anzahl der Beobachtungen einer Klasse deutlich die der anderen übersteigt, so kann die CV in Abhängigkeit der dafür verwendeten Anzahl $N \leq n$ der Teilmengen instabil sein.

Zur Beurteilung der Güte der Klassifikation werden drei Größen berechnet, sowohl für den resultierenden Regressionsvektor zu `lambda.min` als auch `lambda.1se`. Diese

Größen sind Error $\in [0,1]$, Precision $\in [0,1]$ und Recall $\in [0,1]$, die definiert sind als:

$$\text{Error} = \frac{|\{i \in \{1,...,n\} | y_i \neq \hat{y}_i|}{n}, \quad \text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}, \quad \text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|} .$$

Für die Klasse mit der Ausprägung 1 bezeichnen bei einer binären Variable TP die korrekt als $y = 1$, FP die fälschlicherweise als $y = 1$ und FN die fälschlicherweise als $y = 0$ klassifizierten Beobachtungen. Analog lassen sich diese Größen für die Klasse der Beobachtungen mit der Ausprägung 0 definieren. Bei einer multinomialen Outputvariable für die Klasse $k \in \{1,...,K\}$ bezeichnen TP die korrekt als $y = k$, FP die fälschlicherweise als $y = k$ und FN die fälschlicherweise als $y \neq k$ klassifizierten Beobachtungen. Dabei soll Error möglichst gering sein, Precision und Recall für die Zielklasse möglichst hoch sein.

Logistische Regression Bei Vitalstatus der Patienten entspreche $y = 1$ dem Eintritt des Ereignisses, $y = 0$ sonst. Die Ergebnisse der Analyse der CAMDA- und TCGA-Daten unter Verwendung von Vitalstatus als Output y weisen sehr starke Schwankungen und Instabilitäten auf, sowohl in Bezug auf die Genselektion als auch die Klassifikationsgüte. Dies liegt an der stark ungleichmäßigen Verteilung der zum Ende der Studie lebenden und gestorbenen Patienten, vgl. Tabelle 1. Bei CAMDA kommt außerdem ein hoher Unterschied in den Dimensionen, $n = 498 \ll p = 60249$, hinzu. Somit sind zwar jeweils Precision und Recall für die Klasse mit der Ausprägung 0 hoch, für die Klasse mit der Ausprägung 1 hingegen sehr gering, sodass keine sinnvollen Schlussfolgerungen gezogen werden können. Bei den TCGA Brustkrebspatienten wird sogar unabhängig vom Regularisierungsparameter λ für alle Patienten Vitalstatus = 0 vorausgesagt, sodass Precision = Recall = 0 für die Klasse mit der Ausprägung 1 folgt.

Statt des Vitalstatus wird bei CAMDA Genexpressionen für die logistische Regression daher das Alter verwendet. Bei Neuroblastom gibt es eine Risikogrenze von 18 Monaten. Bei Patienten unter 18 Monaten bestehen gute Chancen auf Heilung, bei älteren Patienten ist die Chance deutlich geringer. Die Verteilung von Patienten unter und über 18 Monaten ist in etwa gleich, das Verhältnis ist 3:2. Die Anzahl der Patienten unter 18 Monaten beträgt 300, die der über 18 Monaten 198, sodass von stabileren Ergebnissen auszugehen ist.

Der Output y ist binär mit Ausprägungen 0 und 1, wobei 0 dem Alter unter 18 und 1 über 18 Monaten entspricht. Die folgende Analyse zielt folglich darauf ab, die Patienten ($\leq$ 18 Monate, $>$ 18 Monate) anhand ihrer Genprofile nach ihrem Alter zu klassifizieren. Berechne den Lösungspfad der Koeffizienten mittels `glmnet`:

```
> obj = glmnet(CAMDA, y=age, family="binomial")
```

Führe nun mittels `cv.glmnet` (2-fache) CV durch:

```
> cv.obj1 = cv.glmnet(CAMDA, y=age, family="binomial")
> cv.obj2 = cv.glmnet(CAMDA, y=age, family="binomial")
```

Die Abbildung 10 zeigt die Ergebnisse. In der rechten Graphik ist der Pfad der Koeffizienten abgebildet. Die mittlere und rechte Graphiken zeigen die CV Plots von `cv.obj1` und `cv.obj2`.

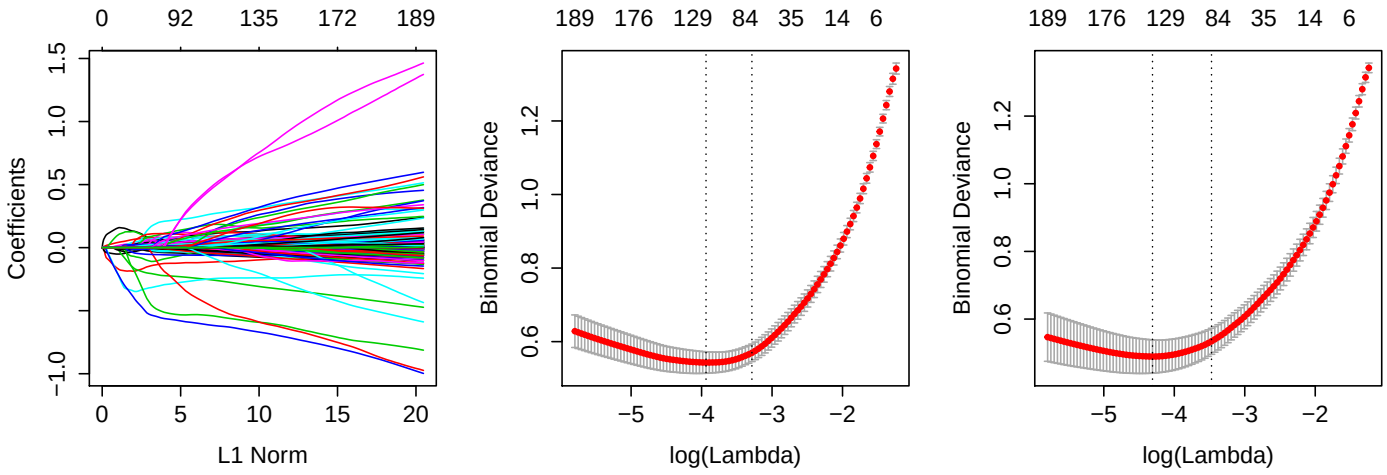


Abbildung 10 CAMDA: Koeffizientenpfad (links) und zwei CV Plots für die logistische Regression mit Alter als Output.

Die Lösungskoeffizienten zu `lambda.min` und `lambda.1se` und die Anzahl der dazugehörigen selektierten Gene werden folgendermaßen bestimmt:

```
> coef.min1 = coef(cv.obj1, s="lambda.min")
> length(coef.min1@i[-1])
1] 119
> coef.1se1 = coef(cv.obj1, s="lambda.1se")
> length(coef.1se1@i[-1])
[1] 81
> coef.min2 = coef(cv.obj2, s="lambda.min")
> length(coef.min2@i[-1])
1] 102
> coef.1se2 = coef(cv.obj2, s="lambda.1se")
> length(coef.1se2@i[-1])
[1] 76
```

Die bspw. zu `coef.1se2` selektierten Gene mit den dazugehörigen 10 betragsmäßig größten Einträgen des resultierenden Regressionsvektors sind:

```
> x = coef.1se2@x[-1]
```

```
> names(x) = colnames(CAMDA)[coef.1se2@i[-1]]
> idx = order(abs(x), decreasing=TRUE)
> x = x[idx,drop=FALSE]
> x = x[1:10]
> data.frame(x)
                                 x
SRGAP3:Gene_AceView   -0.5570722
SESTD1:Gene_AceView   -0.5032119
FOXP1:Gene_AceView    -0.3000080
HES4:Gene_AceView     -0.2202686
ST8SIA3:Gene_AceView   0.2045004
NGLY1:Gene_AceView    -0.1594933
ADRBK2:Gene_RefSeq    -0.1285919
woygey:Gene_AceView    0.1115762
TRMT61A:Gene_AceView   0.1070190
TTLL5:Gene_RefSeq      0.1034779
```

Zunächst wird zur Überprüfung der Stabilität von CV bezüglich der Bestimmung von `lambda.min` und `lambda.1se` und somit der Anzahl der dazugehörigen selektierten Gene `cv.glmnet` wie oben 10-fach auf `CAMDA` mit Alter als Output y angewendet:

```
> coef.min = list()
> coef.1se = list()
> for(j in 1:10){
    cv.obj = cv.glmnet(CAMDA, y=age, family="binomial")
    coef.min[[j]] = coef(cv.obj, s=cv.obj$lambda.min)
    coef.1se[[j]] = coef(cv.obj, s=cv.obj$lambda.1se) }
```

Die Anzahl der zu `lambda.min` selektierten Merkmale schwankt zwischen 102 und 129, die zu `lambda.1se` zwischen 45 und 86, siehe Tabelle 3. Der Schnitt der zu `lambda.min` selektierten Gene enthält dabei 98, der zu `lambda.1se` 39 Gene. Unter Beachtung der Schwankungen lässt sich daher sagen, dass zumindest erste Schlussfolgerungen möglich sind.

Die interne CV von `glmnet` dient vor allem dazu, `lambda.min` und `lambda.1se` zu bestimmen. Um die Stabilität von LASSO in diesem Beispiel zu überprüfen, wird nun mit der eigens geschriebenen Funktion `cv.lasso(X, y, N, family)` zusätzlich (äußere) 10-fache CV durchgeführt. Die Menge der vorhandenen Beobachtungen wird folglich in 10 Teilmengen unterteilt, sodass jede einmal zum Testen verwendet wird. In jeder Iteration wird `cv.glmnet` somit auf eine Teilmatrix von `CAMDA`, sodass die Teilmatrix der Dimension 448×60249 oder 449×60249 ist, angewendet. Dabei werden die zu `lambda.min` und `lambda.1se` gehörigen Gene extrahiert.

```
> cv.object = cv.lasso(CAMDA, y=age, N=10, family="binomial")
```

Die Anzahl der daraus resultierenden Variablen zu `lambda.min` liegt zwischen 85 und 149, zu `lambda.1se` zwischen 47 und 93. Dies deutet auf ein recht stark von der verwendeten Trainingsmenge abhängiges Ergebnis hin, was sich ebenso in anderen Größen widerspiegelt, siehe Tabelle 3.

Es wurde zusätzlich der Durchschnitt der ausgewählten Gene gebildet, wobei der Schnitt der Gene zu `lambda.min` mit dem der Gene zu `lambda.1se` übereinstimmt und gegeben ist durch:

```
> cv.object$schnitt.1se
 [1]  "APOB:Gene_AceView"          "HES4:Gene_AceView"
 [3]  "FOXP1:Gene_AceView"         "LOC100288568:Gene_RefSeq"
 [5]  "MAGEA9B:Gene_AceView"       "SESTD1:Gene_AceView"
 [7]  "SRGAP3:Gene_AceView"        "bloytoy:Gene_AceView"
 [9]  "LOC100287128:Gene_AceView" "C6orf126:Gene_AceView"
 [11] "TTC22:Gene_AceView"         "woygey:Gene_AceView"
 [13] "LOC645545:Gene_RefSeq"
```

Somit liegen nun jeweils nur 13 Gene im Durchschnitt.

Die Ergebnisse von LASSO für die logistische Regression in Verbindung mit CAMDA Genexpressionen und Alter als Output sind in der Tabelle 3 zusammengefasst. Die Größen resultierend aus der (äußeren) CV unterliegen deutlich höheren Schwan-

Datensatz	Laufzeit **glmnet**	Laufzeit **cv.glmnet**	#Gene $\lambda_{min}/\lambda_{1se}$	Error $\lambda_{min}/\lambda_{1se}$	Precision $\lambda_{min}/\lambda_{1se}$	Recall $\lambda_{min}/\lambda_{1se}$
CAMDA	7.561	82.652	$102-129/$ $45-86$	$0-0.012/$ $0.032-0.064$	$0.995-1/$ $0.946-0.98$	$0.975-1/$ $0.889-0.94$
	5.174	62.355	$85-149/$ $47-93$	0.04-0.24	$0.667-1/$ $0.727-1$	0.583-0.947

Tabelle 3 CAMDA: Ergebnisse der logistischen Regression mit Alter als Output. Die erste Zeile bildet den Sachverhalt ab, dass die Matrix `CAMDA` 10-fach sowohl als Trainings- als auch Testdatensatz verwendet wurde. Die zweite Zeile zeigt die Ergebnisse nach 10-facher (äußerer) CV.

kungen als unter Verwendung der vollständigen Matrix `CAMDA`.

Multinomiale Regression Als nächstes wird LASSO in Verbindung mit dem Stadium untersucht. Die Beschreibung der Stadien von Neuroblastom kann unter `https://www.cancer.gov/types/neuroblastoma/patient/neuroblastoma-treatment-pdq#section/_21` oder `https://www.cancer.net/cancer-types/neuroblastoma-childhood/stages-and-groups`, der Stadien von Brustkrebs unter `https://www.krebsinformationsdienst.de/tumorarten/brustkrebs/stadieneinteilung.php` nachgelesen werden. Für die

Analyse werden hier bei den TCGA Brustkrebspatienten die Stadien IA und IB zu Stadium 1, IIA und IIB zu Stadium 2, IIIA, IIIB und IIIC zu Stadium 3 zusammengefasst, 4 bezeichnet das Stadium IV.

LASSO ist in diesem Fall instabil, weil die Verteilung der Anzahl der zu dem jeweiligen Stadium gehörenden Patienten ungleichmäßig ist, siehe Tabelle 4. Insbesondere bei TCGA ist dieser Sachverhalt problematisch, da mehr als die Hälfte der Patienten Stadium 2 aufweist, dies wird sich in Ergebnissen widerspiegeln.

	Stadium 1	**Stadium 2**	**Stadium 3**	**Stadium 4**	**Stadium 4S**
CAMDA	121	78	63	183	53
TCGA	137	430	171	9	

Tabelle 4 Verteilung der Stadien von CAMDA Patienten und TCGA Brustkrebspatienten.

Die Analyse erfolgt analog zur logistischen Regression. y habe nun die Ausprägungen 1 bis 5 bei CAMDA Genexpressionen, wobei 5 dem Stadium 4S entspricht, und 1 bis 4 bei TCGA Genexpressionen. Berechne jeweils den Lösungspfad der Koeffizienten mittels `glmnet` und führe mittels `cv.glmnet` CV durch:

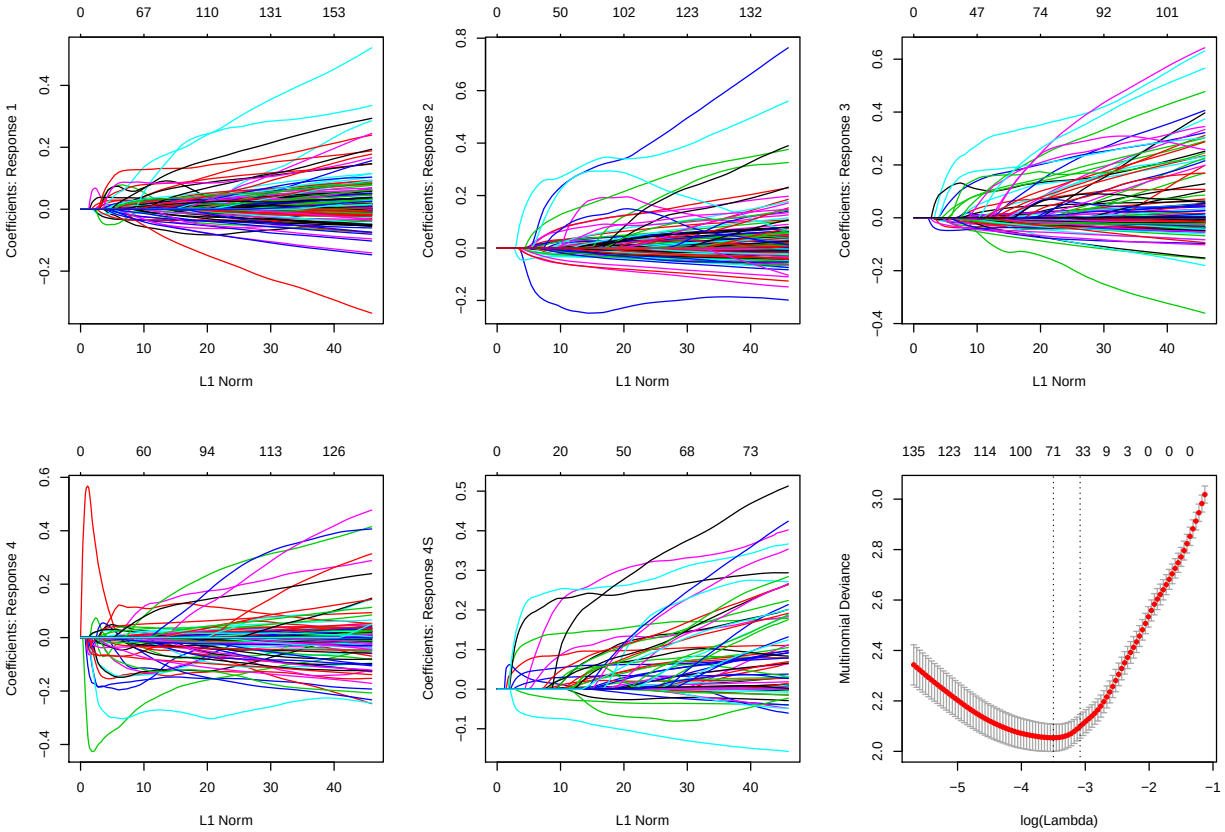
```
> obj.camda = glmnet(CAMDA, y=stage.camda, family="multinomial")
> cv.obj.camda = cv.glmnet(CAMDA, y=stage.camda,
                           family="multinomial")
> obj.tcga = glmnet(TCGA, y=stage.brca, family="multinomial")
> cv.obj.tcga = cv.glmnet(TCGA, y=stage.brca,
                          family="multinomial")
```

Wie im Kapitel 2.4.1 besprochen wurde, sind bei der multinomialen Regression insgesamt K Hyperebenen, hier folglich fünf bei CAMDA und vier bei TCGA, zu bestimmen. Der Koeffizientenpfad der Lösungen für die Hyperebenen CAMDA bzw. TCGA inklusive des CV Plots von `cv.obj.camda` bzw. `cv.obj.tcga` ist in den Abbildungen 11a bzw. 11b dargestellt.

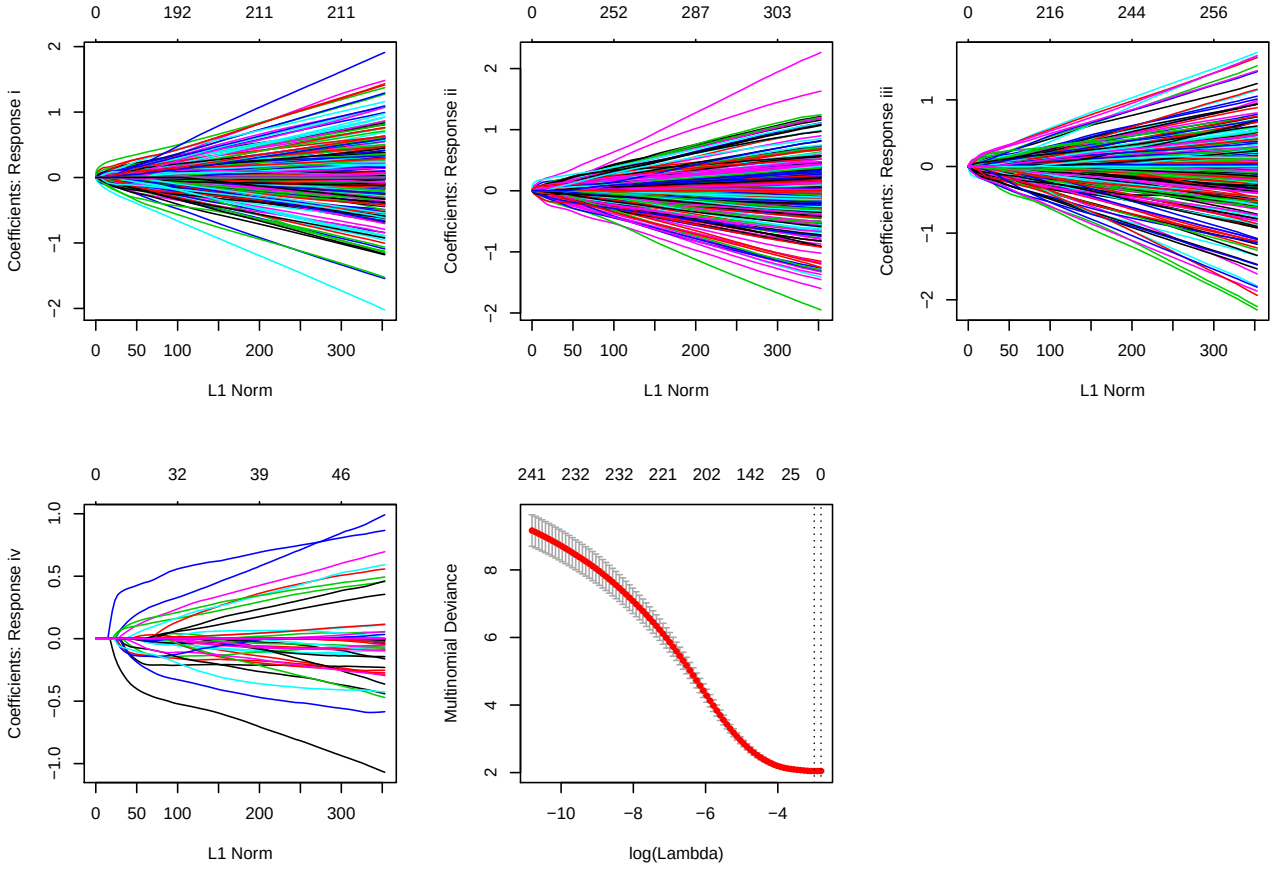
Führe nun 10-fache (äußere) CV mittels der Funktion `cv.lasso`:

```
> cv.object.camda = cv.lasso(CAMDA, y=stage.camda, N=10
                             family="multinomial")
> cv.object.tcga = cv.lasso(TCGA, y=stage.brca, N=10,
                            family="multinomial")
```

Bei der multinomialen Regression werden aufgrund der Bestimmung von K Hyperebenen, was der Berechnung von K Regressionsvektoren entspricht, Gene für jede Klasse, hier für jedes Stadium selektiert. Bspw. enthält der Schnitt der selektierten Gene zu `lambda.min` für das Stadium 4 nach 10-facher (äußerer) CV die folgenden Gene:



(a) CAMDA.



(b) TCGA.

Abbildung 11 CAMDA und TCGA: Koeffizientenpfad der multinomialen Regression mit Stadium als Output inklusive eines CV Plots.

```
> cv.object.camda$schnitt.min[[4]]
[1] "AGRN:Gene_AceView"    "ALDH3A2:Gene_AceView"
[3] "MRPL11:Gene_AceView" "nukame:Gene_AceView"
> cv.object.tcga$schnitt.min[[4]]
[1] "hsa-mir-22-3p"
```

Die Ergebnisse der multinomialen Regression mit Stadium als Output nach 10-facher (äußerer) CV sind in der Tabelle 5 abgebildet. Dabei wurden Precision und Recall für jede Klasse berechnet.

Datensatz	**Stadium**	**Precision** $\lambda_{min}/\lambda_{1se}$	**Recall** $\lambda_{min}/\lambda_{1se}$	**CV-#Gene** $\lambda_{min}/\lambda_{1se}$	**CV-Error** $\lambda_{min}/\lambda_{1se}$	**Laufzeit** **glmnet/cv.glmnet**
CAMDA	1	$0.25-0.786/$ $0.315-0.733$	$0.364-0.923/$ $0.545-1$	$52-96/$ $27-53$	$0.3-0.52/$ $0.32-0.469$	27.064 / 241.144
	2	$0.125-0.333/$ $0-0.5$	$0.083-0.25/$ $0-0.333$	$35-69/$ $7-44$		
	3	$0-0.75/$ $0-1$	$0-0.375/$ $0-0.25$	$36-59/$ $8-41$		
	4	$0.68-0.842/$ $0.65-0.808$	$0.833-1/$ $0.87-1$	$55-70/$ $36-53$		
	4S	$0.667-1/$ $0.75-1$	$0.399-0.715/$ $0.4-0.715$	$18-40/$ $9-17$		
TCGA	1	0	0	$0-6/$ $0-1$	$0.379-0.493/$ $0.379-0.493$	3.821 / 33.5
	2	0.507-0.623	1	$1-14/$ $1-2$		
	3	0	0	$1-5/$ $0-1$		
	4	0	0	$1-2/$ 1		

Tabelle 5 CAMDA und TCGA: Ergebnisse der multinomialen Regression mit Stadium als Output nach 10-facher (äußerer) CV. Die Laufzeit entspricht der durchschnittlichen Laufzeit zur Berechnung eines Objektes mittels `glmnet` bzw. `cv.glmnet` für einen Trainingsdatensatz.

Die Ergebnisse zeigen ganz deutlich, wie sich die ungleichmäßige Verteilung der Stadien auswirkt. Fast alle Größen weisen starke Unregelmäßigkeiten auf, was darauf schließen lässt, dass die berechneten Objekte stark von dem dazu jeweils verwendeten Trainingsdatensatz abhängt. Die hohen Werte für Error sind durch die oft falsche Zuweisung von Stadium 4 bei CAMDA bzw. Stadium 2 bei TCGA zu Patienten des

Testdatensatzes bedingt. Die Werte von Precision und Recall von 0 für die Stadien 1, 3 und 4 bei TCGA sagen aus, dass die meisten Patienten des Testdatensatzes offenbar dem Stadium 2 zugewiesen werden, sodass keine TP für die Stadien 1, 3 und 4 existieren. Ebenso variiert die Anzahl der selektierten Gene. Bei TCGA fällt auf, dass oftmals $\hat{\beta} \equiv 0_{|p|}$, folglich das Nullmodell, den kleinsten CV-Fehler liefert.

4.1.2 Cox-Regression

In diesem Kapitel wird die Cox-Regression praktisch umgesetzt, siehe Kapitel 2.4.2. Der Output y besteht aus den Komponenten `time`, die entweder der Überlebenszeit oder der Censoring Time der Patienten entspricht, und `status`. Dabei ist Vitalstatus = 1, falls das Ereignis eingetreten ist (der Patient ist an der Krankheit gestorben), sonst Vitalstatus = 0 zu setzen.

Die Log-Likelihood-Funktion für die Cox-Regression 2.73 enthält keinen Achsenabschnitt β_0, zur Vollständigkeit wird bei der Ausführung von `glmnet` und `cv.glmnet` daher `intercept=FALSE` gesetzt.

Berechne wie bisher den Lösungspfad der Koeffizienten mittels `glmnet` und ein CV-Objekt mittels `cv.glmnet`:

```
> object.camda = glmnet(CAMDA, y=Surv(time1,status1),
                    family="cox", intercept=FALSE)
> cv.obj.camda = cv.glmnet(CAMDA, y=Surv(time1,status1),
                       family="cox", intercept=FALSE)
> object.tcga = glmnet(TCGA, y=Surv(time2,status2),
                    family="cox", intercep=FALSE)
> cv.obj.tcga = cv.glmnet(TCGA, y=Surv(time2,status2),
                       family="cox", intercept=FALSE)
```

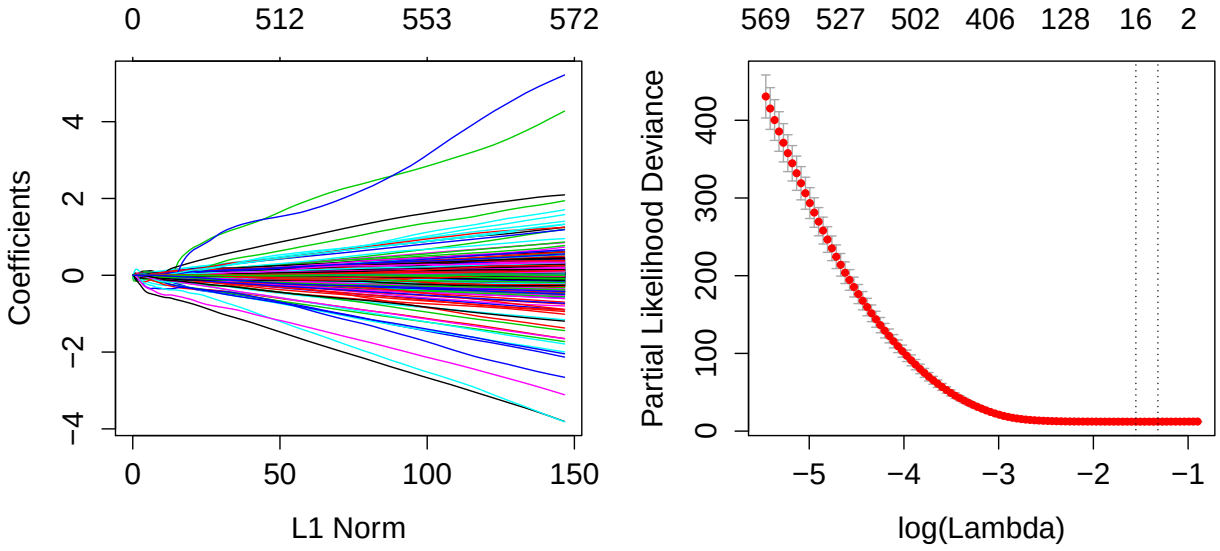
Die Abbildungen 12a und 12b zeigen die Plots.

Nach 10-facher Anwendung von `cv.glmnet` mit der vollständigen Matrix `CAMDA` liegt die Spanne der Anzahl der selektierten Gene zu `lambda.1se` zwischen 14 und 17, zu `lambda.1se` zwischen sieben und 11, siehe Tabelle 6. Der Schnitt der Gene zu `lambda.1se` ist gegeben durch:

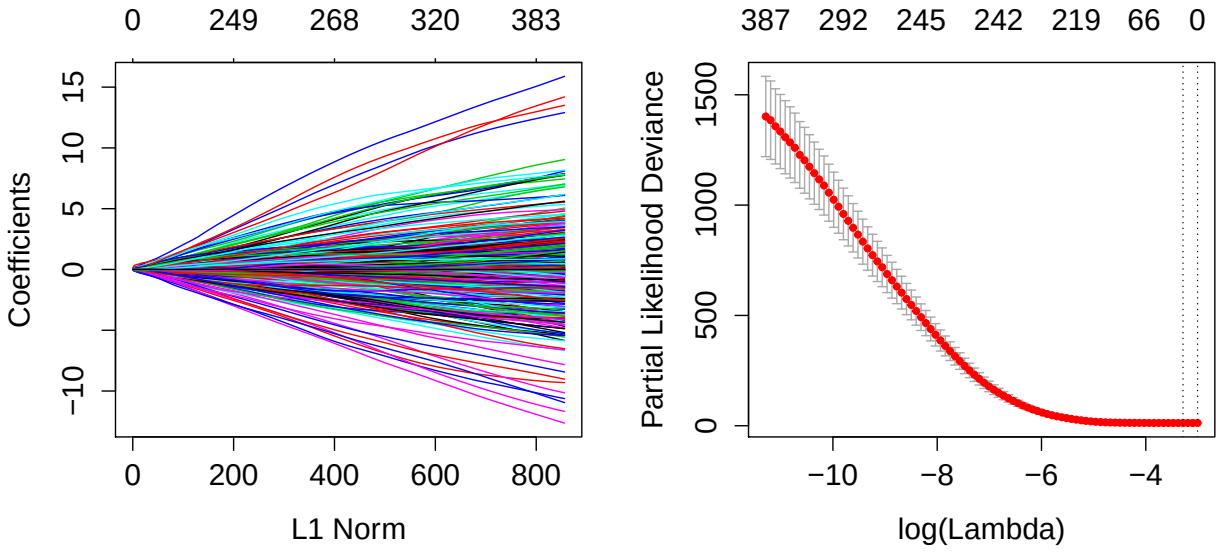
```
[1] "UXT:Gene_AceView"    "MPRIP:Gene_AceView"
[3] "RPH3A:Gene_AceView" "MRPL10:Gene_AceView"
[5] "STX10:Gene_AceView" "korgu:Gene_AceView"
[7] "plusey:Gene_AceView"
```

Die Schnitt der zu `lambda.min` ausgewählten Gene enthält 13 Gene. Bei TCGA liegen sechs Gene im Schnitt der zu `lambda.min` selektierten Gene:

```
[1] "hsa-mir-143-3p"   "hsa-mir-15a-3p"  "hsa-mir-181c-3p"
[4] "hsa-mir-29b-1-5p" "hsa-mir-3187-3p" "hsa-mir-374a-3p"
```



(a) CAMDA.



(b) TCGA.

Abbildung 12 CAMDA und TCGA: Koeffizientenpfad (links) der Cox-Regression mit Output y als Zusammensetzung von `time` und `status` inklusive eines CV Plots (rechts).

Die Wahl der Gene mit den vollständigen Genexpressionsmatrizen scheint somit recht konsistent zu sein.

Bei der 10-fachen (äußeren) CV ändert sich dies jedoch. Sowohl unterliegt die Anzahl der selektierten Gene viel stärkeren Schwankungen als auch die Auswahl der Gene:

```
> cv.object.camda = cv.lasso(CAMDA, y=stage.camda, N=10
                             family="cox")
```

```
> cv.object.tcga = cv.lasso(TCGA, y=stage.brca, N=10,
                            family="cox")
> cv.object.camda$schnitt.min
[1] "CNIH4:Gene_AceView" "DKC1:Gene_RefSeq"
[3] "snawjarby:Gene_AceView"
> cv.object.tcga$schnitt.min
[1] "hsa-mir-374a-5p"
```

Die Ergebnisse für die Cox-Regression sind in der Tabelle 6 dargestellt. Die Spalte

Datensatz	**Laufzeit glmnet**	**Laufzeit cv.glmnet**	**#Gene $\lambda_{min}/\lambda_{1se}$**	**CV-#Gene $\lambda_{min}/\lambda_{1se}$**
CAMDA	23.524	573.178	$14-17/$ $7-11$	$27-43/$ $10-24$
TCGA	21.511	202.375	$6-16/$ 0	$1-22/$ 0

Tabelle 6 CAMDA und TCGA: Ergebnisse der Cox-Regression mit Output y als Zusammensetzung von `time` und `status`.

"#Gene" gibt die Spanne der Anzahl der ausgewählten Gene nach 10-facher Anwendung von `cv.glmnet` auf die vollständigen Genexpressionsmatrizen, die Spalte "CV-#Gene" die Spanne der Anzahl selektierter Gene nach 10-facher (äußerer) CV.

4.2 LASSO vs. SLOPE

Dieses Kapitel vergleicht LASSO und SLOPE anhand künstlich generierter Daten und Genexpressionen.

Zunächst eine kurze Einführung in das R-Paket **SLOPE**. Dieses wird zur Lösung des Problems 3.1 eingesetzt. Die wichtigste Funktion von **SLOPE** ist `SLOPE`. Diese setzt alle anderen Funktionen des R-Pakets ein. Das R-Paket stellt für die RP die Sequenzen $\boldsymbol{\lambda}_{BH}$ (3.4) und $\hat{\boldsymbol{\lambda}}_G$ (3.9) zur Verfügung, `lambda=c("bhq","gaussian")`, der Anwender kann ebenso eine eigene Sequenz wählen. Die wesentlichen Rückgabewerte von `SLOPE` sind `selected` als die Indizes (Spalten) der ausgewählten Variablen bzw. Merkmale, `beta` als der resultierende Regressionsvektor, der eine Approximation zu β_S (3.2) darstellt, und `sigma` als eine Schätzung der Standardabweichung der abhängigen Variable y. Ein wichtiger Eingabewert von `SLOPE` ist `normalize`, der als `TRUE` voreingestellt ist. Es ist wichtig zu beachten, dass der berechnete Lösungsvektor bei Setzen von `normalize=TRUE` nicht auf der Originalskala zurückgegeben wird und somit, falls nötig, einer Umrechnung bedarf. Handelt es sich bei der Systemmatrix um keinen Spezialfall wie bspw. orthogonale Spalten oder den im Kapitel 3.2

betrachteten Fall, oder es bestehen fehlende Informationen über die Verteilung des Outputs, der Merkmale oder Matrixeinträge, so ist `normalize=TRUE` zu setzen.

`SLOPE` verwendet den Algorithmus 11, falls der Eingabewert `sigma` unspezifiziert bleibt, ansonsten kommt der Algorithmus 10 zum Einsatz, wobei als Abbruchkriterium die Dualitätslücke benutzt wird, siehe Kapitel 3.3.3.

4.2.1 Synthetische Daten

Bei künstlich generierten Problemen, bei denen der Anwender die Lösung vorgibt oder diese problemlos bestimmt werden kann, kann nach der Approximation dieser Lösung die FDP (3.3) als der Anteil der ausgewählten irrelevanten zu allen ausgewählten Variablen berechnet werden. In diesem Unterkapitel ist β^* der im Vorhinein vorgegebene optimale Lösungsvektor. Das Ziel der Analyse dieses Unterkapitels ist zum einen, zu überprüfen, ob LASSO und SLOPE die im Vorhinein anhand von β^* bekannten relevanten Variablen finden und ob bzw. wie viele irrelevante Variablen selektiert werden, zum anderen, zu untersuchen, wie gut LASSO und SLOPE das β^* approximieren.

Wähle $k = 50$ als Anzahl der relevanten Variablen, die den Nichtnulleinträgen des Regressionsvektors β^* entsprechen. Der Support von β^* sei J^*.

Im Folgenden wird $\beta^*_{J^*} = 5\sqrt{2\log(p)}$ gesetzt. Dabei erhält ein beliebiges $j \in J^*$ den Wert $\beta^*_j = 10\sqrt{2\log(p)}$, sodass dieser dem (betragsmäßig) größten Eintrag und j damit der wichtigsten Variable entspricht. Das `beta.stern` bezeichne β^*. In diesem Beispiel ist $j = 581$ die wichtigste Variable.

Es folgt die Anwendung von LASSO und SLOPE mit der Matrix `X.orth` mit $n = p = 1000$, generiert mit R. Der Output sei $y = X\beta^* + \epsilon$ mit $\epsilon \sim \mathcal{N}(0, 1)$. Die Standardabweichung von y ist folglich durch $\sigma = 1$ gegeben. Für das β^* gilt $\beta^*_{581} \approx 37.16922$, die restlichen Einträge betragen ca. 18.58461. Da `X.orth` als orthogonale Matrix invertierbar ist, sind die Lösungen von LASSO für jedes beliebige $\lambda \geq 0$ und SLOPE für jede beliebige Sequenz $\boldsymbol{\lambda}$ eindeutig.

Zunächst die Anwendung von `glmnet` und `cv.glmnet`.

```
> obj = glmnet(X.orth, y, family="gaussian")
> cv.obj = cv.glmnet(X.orth, y, family="gaussian")
```

Die Abbildung 13 stellt das Ergebnis graphisch dar. Anhand des CV Plots ist ersichtlich, dass die Anzahl der sowohl zu `lambda.min` als auch `lambda.1se` selektierten Variablen deutlich höher ist als deren tatsächliche Anzahl, die hier 50 beträgt.

Extrahiere die für `lambda.min` und `lambda.1se` berechneten Koeffizienten:

```
> coef.min = coef(cv.obj, s=cv.obj$lambda.min)
> coef.1se = coef(cv.obj, s=cv.obj$lambda.1se)
```

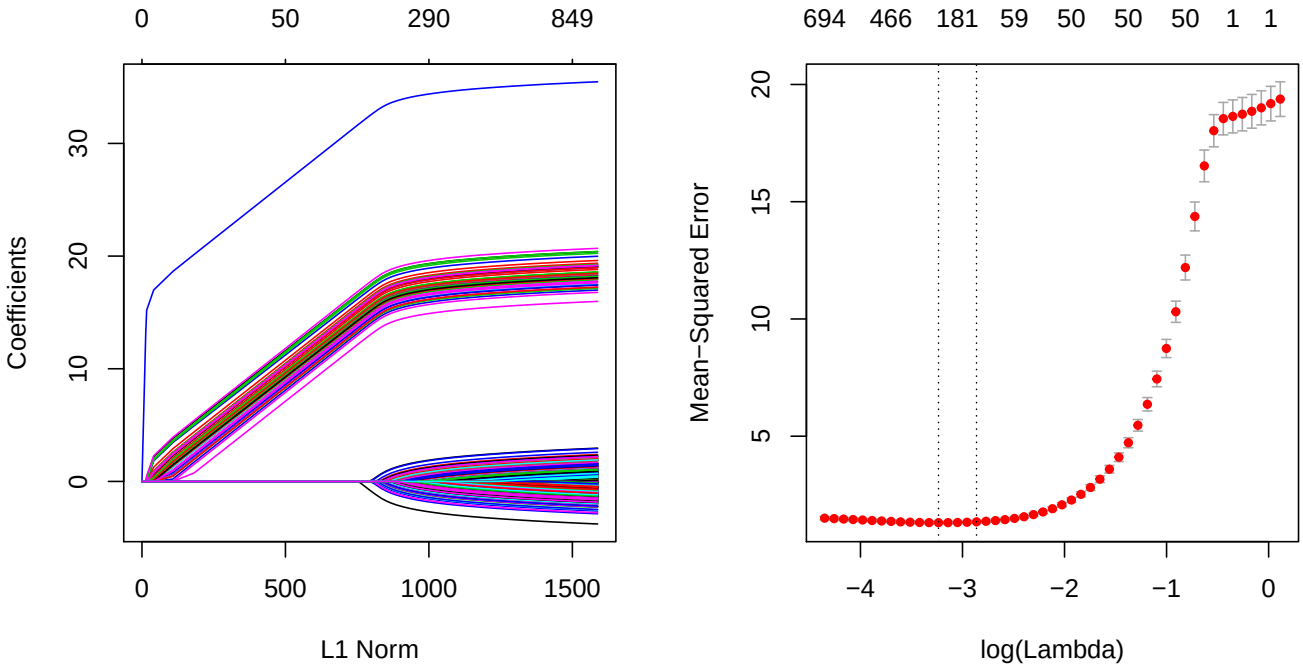


Abbildung 13 LASSO mit einer orthogonalen Systemmatrix. Koeffizientenpfad (links) und CV Plot (rechts).

Die Anzahl der für `lambda.min` extrahierten Variablen ist 260, für `lambda.1se` wurden 129 Variablen ausgewählt, die Anzahl übersteigt damit die Anzahl der relevanten Variablen erheblich. Nun wird überprüft, ob die zu `lambda.1se` gehörenden Variablen alle relevanten Variablen enthält[15].

```
> selected.stern = which(beta.stern!=0)
> selected.1se = coef.1se@i[-1]
> length(intersect(selected.1se,selected.stern))
[1] 50
```

Somit wurden neben 79 irrelevanten alle 50 relevanten Variablen ausgewählt. Der (betragsmäßig) größte Eintrag wurde korrekt der Variable 581 zugewiesen, anhand der Abbildung 13 wird deutlich, dass dies für den ganzen Koeffizientenpfad, d.h. für jedes λ gilt. Außerdem selektiert LASSO bis zum Überschreiten von 50 Nichtnulleinträgen ab einem λ nur die tatsächlich relevanten Variablen. Der Koeffizientenpfad zeigt ebenfalls, dass diese im Gegensatz zu den relevanten Koeffizienten betragsmäßig klein sind. Einerseits werden zu viele Variablen selektiert, andererseits sind die Einträge der sich darunter befindenden relevanten Variablen betragsmäßig wesentlich höher, sodass dies dem Anwender auffallen muss. Obwohl 79 Variablen zu viel selektiert wurden, ist der resultierende Vektor eine nicht allzu schlechte Approxima-

[15] Wie zu Anfang dieses Kapitels erklärt, sind die zu `lambda.1se` selektierten eine Teilmenge der zu `lambda.min` ausgewählten Variablen.

tion von `beta.stern`:

```
> norm(beta.stern[coef.1se@i]-coef.1se@x[-1], type="2")
[1] 15.15355
```

Die Norm des Residuums wird folgendermaßen bestimmt:

```
> predict.1se = predict(cv.obj, newx=X.orth, s="lambda.1se",
                        type="response")
> norm(predict.1se - y, type="2")
[1] 31.73266
```

Anhand des Koeffizientenpfades ist ersichtlich, dass die Werte der relevanten Variablen mit fallendem λ gegen die tatsächlichen Werte konvergieren.

Ein möglicher Wert für λ für LASSO bei einem Problem, bei dem die Systemmatrix orthogonale Spalten aufweist und die Größenordnung der relevanten Merkmale bekannt ist, ließe sich bspw. mit Hilfe der Graphik ablesen. Eine mögliche Wahl für dieses Beispiel ist e^{-2}, vgl. Abbildung 13.

```
> coef = coef(cv.obj, s=exp(-2))
> selected = coef@i[-1]
> length(selected)
[1] 50
> length(intersect(selected,selected.stern))
[1] 50
```

Für diesen Wert für den RP werden genau 50 Variablen ausgewählt, wobei diese mit den optimalen übereinstimmen.

Da die Matrix orthogonal ist, gilt die Irrepresentable Condition (2.84). Im Kapitel 2.6 wurde bei Gelten der Irrepresentable Condition für den RP $\lambda \geq \frac{8\sigma}{\gamma}\sqrt{\frac{\log(p)}{n}}$ (2.85) vorgeschlagen. In diesem Beispiel ist $\gamma = 1$ und $\sigma = 1$, sodass sich als möglicher RP

```
> lambda.irr = 8*sqrt(log(p)/n)
```

ergibt. Die Koeffizienten und die Anzahl aller und korrekt ausgewählter Merkmale zu `lambda.irr` sind:

```
> coef.irr = coef(cv.obj, s=lambda.irr)
> selected.irr = coef.irr@i[-1]
> length(selected.irr)
[1] 7
> length(intersect(selected.irr,selected.stern))
[1] 7
```

Wie bereits in der Theorie beschrieben, sind die zu `lambda.irr` selektierten Variablen in den relevanten enthalten. Insgesamt wurden jedoch nur sieben der 50 relevanten Variablen ausgewählt. Der RP `lambda.irr` ist somit zu hoch.

Nun folgt die Anwendung von SLOPE. Wegen der Orthogonalität und der Verteilung von y ist aus Kapitel 3.1 bekannt, dass SLOPE mit der Sequenz $\boldsymbol{\lambda}_{BH}$ (3.4) die FDR, die ein zwingender Eingabewert der Funktion `SLOPE` ist, zu einem gewünschten Level q kontrolliert. Bei der Anwendung von `SLOPE` ist wie bereits erklärt zu beachten, dass im Falle der Standardisierung des Systems die resultierende Lösung einer Umrechnung auf die Originalskala bedarf. In diesem Beispiel muss wegen der Eigenschaften von `X.orth` keine Standardisierung vorgenommen werden, sodass `normalize=FALSE` gesetzt wird. Es wird jeweils ein Objekt mit `fdr=0.1` und `fdr=0.2` berechnet.

```
> slope1 = SLOPE(X.orth, y, fdr=0.1, lambda="bhq", sigma=1,
                 normalize=FALSE)
> slope2 = SLOPE(X.orth, y, fdr=0.2, lambda="bhq", sigma=1,
                 normalize=FALSE)
```

Die Nichtnulleinträge der resultierenden Lösungsvektoren sind in der Abbildung 14 dargestellt.

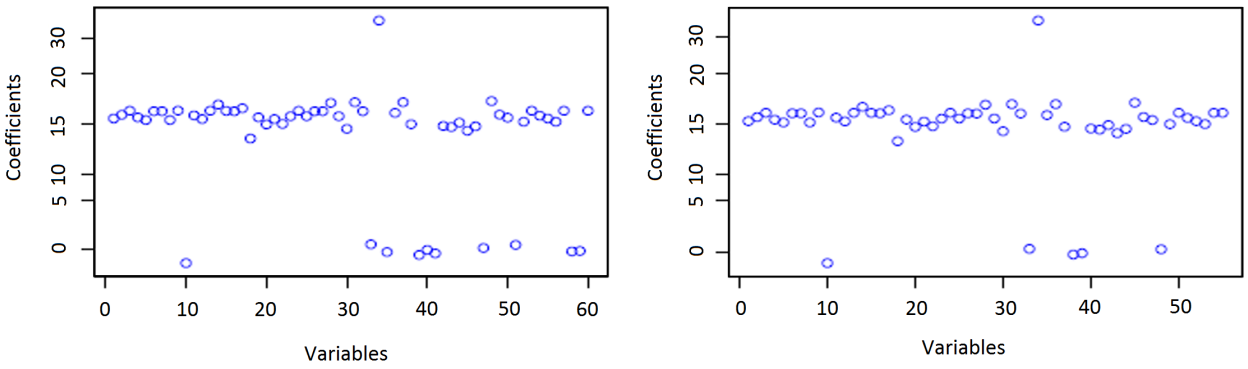


Abbildung 14 SLOPE: Nichtnulleinträge von zwei berechneten Lösungsvektoren mit einer orthogonalen Systemmatrix. Links: `fdr=0.1`; rechts: `fdr=0.2`.

Mit den folgenden Befehlen wird überprüft, wie viele Variablen selektiert wurden, wie hoch die Norm der Differenz von `beta.stern` und des jeweils berechneten `beta` ist:

```
> length(slope1$selected)
[1] 55
> length(slope2$selected)
[1] 60
> length(intersect(slope1$selected,selected.stern))
[1] 50
> length(intersect(slope2$selected,selected.stern))
[1] 50
> norm(beta.stern-slope1$beta, type="2")
```

```
[1] 22.52735
> norm(beta.stern-slope2$beta, type="2")
[1] 21.09586
```

`slope` hat somit bei `fdr=0.1` fünf, bei `fdr=0.2` 10 irrelevante Variablen den relevanten zugeordnet, wobei die wichtigste Variable, hier 581, erkannt wurde. Außerdem sind die Koeffizienten der irrelevanten Variablen nah bei Null. Die Norm des jeweiligen Residuums ist:

```
> norm(X.orth%*%slope1$beta - y, type="2")
[1] 37.96322
> norm(X.orth%*%slope2$beta - y, type="2")
[1] 36.96659
```

Interessant zu bemerken ist, dass bei `fdr=0.2` die Approximation von `beta.stern`, gemessen an der L2-Norm, etwas besser und Norm des Residuum etwas kleiner als bei `fdr=0.1` ausfallen. Dies liegt daran, dass SLOPE auf die Selektion von Merkmalen einen größeren Wert legt.

Sowohl LASSO als auch SLOPE haben in diesem Beispiel alle relevanten Variablen gefunden und die wichtigste Variable erkannt, wobei bei beiden die zu Unrecht als relevant eingeordneten Variablen betragsmäßig viel kleinere Einträge im Vergleich zu den relevanten haben. Daher lässt sich vermuten, dass der Anwender diese ebenso als irrelevant einstufen würde. Dieser Sachverhalt kann durch Rundungsfehler bedingt sein, sodass es möglich ist, dass beide Verfahren bei exakter Rechnung weniger oder keine irrelevanten Variablen selektieren.

Es folgt ein Beispiel mit der Matrix `X.gauss` mit $n = 1000$ und $p = 5000$, deren Einträge normalverteilt sind, wobei diese $x_{ij} \sim \mathcal{N}(0, \frac{1}{n})$ erfüllen. Die Anzahl der relevanten Variablen sei weiterhin 50. Die Koeffizienten von β^* sind gegeben durch $5\sqrt{2\log(p)} \approx 20.63637$, wobei ein Koeffizient den Wert $10\sqrt{2\log(p)} \approx 41.27273$ erhält. Der Output sei $y = X\beta^* + \epsilon$ mit $\epsilon \sim \mathcal{N}(0, \frac{1}{n})$, sodass die Standardabweichung von y durch $\sigma = \frac{1}{n}$ gegeben ist. Aufgrund der Eigenschaften von `X.gauss` muss für die Anwendung von `SLOPE` keine Standardisierung erfolgen. Es sei angemerkt, dass `X.gauss` die Irrepresentable Condition nicht erfüllt.

Wie im Kapitel 3.2 beschrieben, kann für derartige Systemmatrizen $\hat{\boldsymbol{\lambda}}_G$ als Sequenz zum Einsatz kommen.

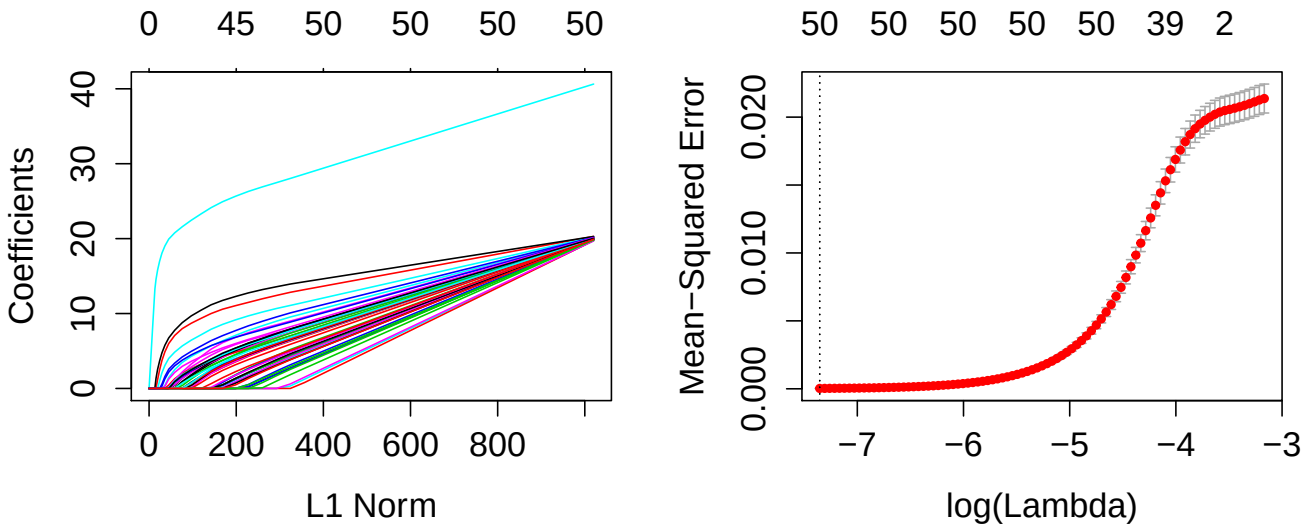
```
> lambda1 = create_lambda(n, p, fdr=0.1, method="gaussian")
> lambda2 = create_lambda(n, p, fdr=0.2, method="gaussian")
> which(lambda.gauss1==min(lambda.gauss1))[1]
[1] 10
> which(lambda.gauss2==min(lambda.gauss2))[1]
[1] 12
```

Die Sequenz `lambda1` ist bis zum Eintrag 10, `lambda2` bis zum Eintrag 12 streng monoton fallend, ab dem Index 10 bzw. 12 stimmen alle Werte überein.

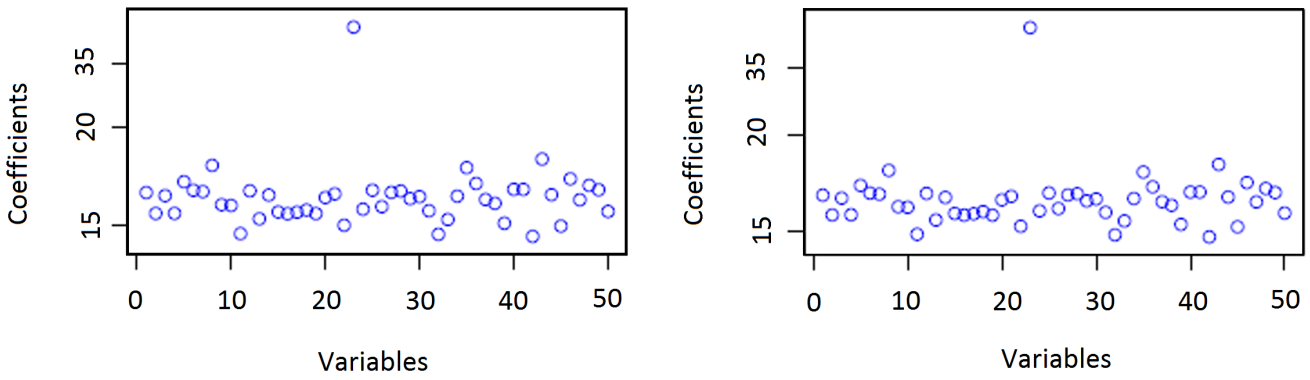
Berechne den Koeffizientenpfad und CV mittels `glmnet` und `cv.glmnet`, und jeweils ein Objekt für die FDR von 0.1 bzw. 0.2 mittels `SLOPE`:

```
> obj = glmnet(X.gauss, y, family="gaussian")
> cv.obj = cv.glmnet(X.gauss, y, family="gaussian")
> slope1 = SLOPE(X.gauss, y, fdr=0.1, lambda=lambda1,
                 sigma=1/n, normalize=FALSE)
> slope2 = SLOPE(X.gauss, y, fdr=0.2, lambda=lambda2,
                  sigma=1/n, normalize=FALSE)
```

Die Abbildung 15 stellt die Ergebnisse graphisch dar. Die Anzahl der selektierten



(a) LASSO: Koeffizientenpfad (links) und CV Plot (rechts).



(b) SLOPE: Nichtnulleinträge von zwei berechneten Lösungsvektoren. Links: `fdr=0.1`; rechts: `fdr=0.2`.

Abbildung 15 Graphische Darstellung der Ergebnisse von LASSO und SLOPE mit einer Systemmatrix, deren Einträge normalverteilt sind.

Variablen ist sowohl bei LASSO zu `lambda.1se` als auch bei SLOPE jeweils 50, wobei diese den relevanten Variablen entsprechen. Bei LASSO gibt es maximal 50 Nichtnulleinträge. `lambda.min` und `lambda.1se` stimmen überein und diese entsprechen dem kleinsten $\lambda \approx 0.00064$. Die Abbildung zeigt ganz klar, dass bei LASSO die Werte der Koeffizienten mit fallendem λ gegen den tatsächlichen Wert der Nichtnulleinträge konvergieren. Da hier `lambda.min`, `lambda.1se` und das kleinste λ gleich sind, ist die zugehörige Lösung eine gute Approximation von β^*, siehe Tabelle 7. Die optimale Lösung für LASSO entspricht hierfür somit (fast) der Lösung von OLS. Bei SLOPE sind viele Nichtnulleinträge (bis auf den maximalen Wert) teilweise deutlich kleiner als der tatsächliche Wert von 20.63637, sodass SLOPE das β^* etwas schlechter approximiert. Bei beiden Methoden ist die Norm des Residuums klein, wobei die Norm bei LASSO kleiner ist, siehe Tabelle 7.

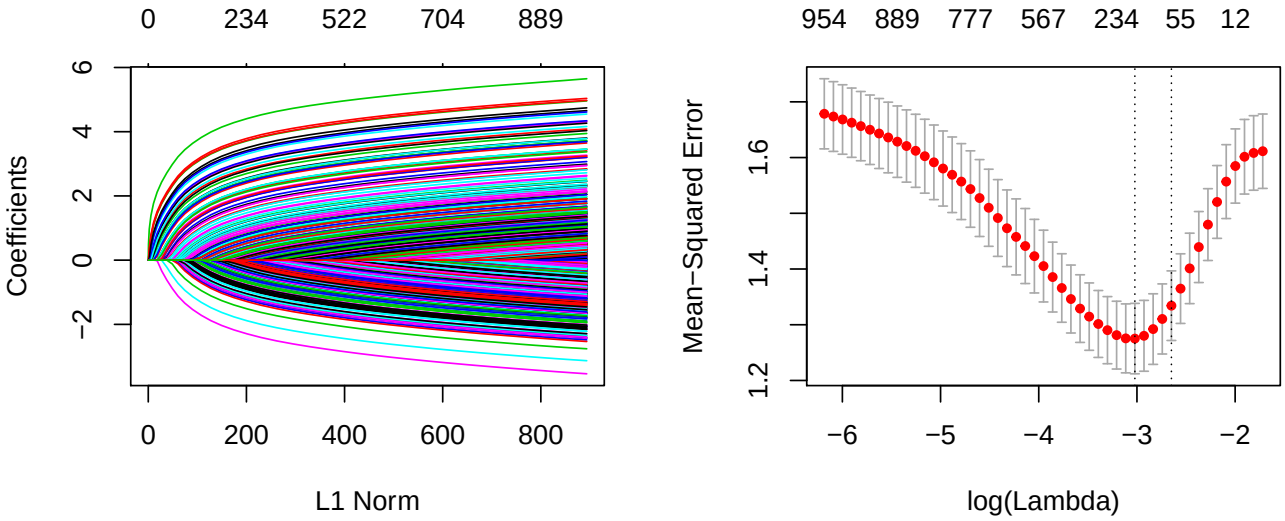
weak signals Die gute Performance sowohl von LASSO als auch von SLOPE bei den obigen Beispielen ist unter anderem auf die Höhe der Nichtnulleinträge von `beta.stern` zurückzuführen. Bogdan et al. (2015) zeigen diesen Sachverhalt. Die Autoren sprechen von *strong signals* und *weak signals*. Alle Nichtnulleinträge von β^* erhalten nun den Wert $\sqrt{2\log(p)}$ und es erfolgt die Analyse mit denselben Matrizen und dem entsprechend generierten Output y.

Ziehe zunächst `X.orth` heran. Alle Einträge von β^* haben hierbei den (gerundeten) Wert von 3.717. Berechne analog die jeweiligen Objekte:

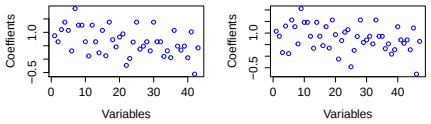
```
> obj = glmnet(X.orth, y, family="gaussian")
> cv.obj = cv.glmnet(X.orth, y, family="gaussian")
> slope1 = SLOPE(X.orth, y, fdr=0.1, lambda="bhq", sigma=1,
                 normalize=FALSE)
> slope2 = SLOPE(X.orth, y, fdr=0.2, lambda="bhq", sigma=1,
                 normalize=FALSE)
```

Der Koeffizientenpfad von `obj` und der CV Plot von `cv.obj` sind in der Abbildung 16a zu finden. Der CV Plot von `cv.obj` 16a zeigt, dass der Wert der RP `lambda.min` und `lambda.1se` von `cv.obj` zu niedrig ist, wodurch zu viele Variablen selektiert werden. Für `lambda.1se` sind es 72 Variablen, darunter 46 relevante. Die Koeffizienten sind zerstreut und einige haben sogar einen negativen Wert. `slope1` (`slope2`) liefert insgesamt 43 (47) Variablen, darunter 3 (5) irrelevante, sodass die FDR von `0.1` (`0.2`) nicht überschritten wird. Die Nichtnulleinträge der berechneten Lösungsvektoren von `slope1` und `slope2` sind in der Abbildung 16b dargestellt. Die Einträge sind ebenso wie bei LASSO zerstreut und im Vergleich zu den tatsächlichen Werten von β^* von 3.717 kleiner, wenige Koeffizienten haben ebenso einen negativen Wert.

Dieses Beispiel verdeutlicht, dass selbst im optimalen Fall einer orthogonalen Ma-



(a) LASSO: Koeffizientenpfad (links) und CV Plot (rechts).



(b) SLOPE: Nichtnulleinträge von zwei berechneten Lösungsvektoren. Links: `fdr=0.1`; rechts: `fdr=0.2`.

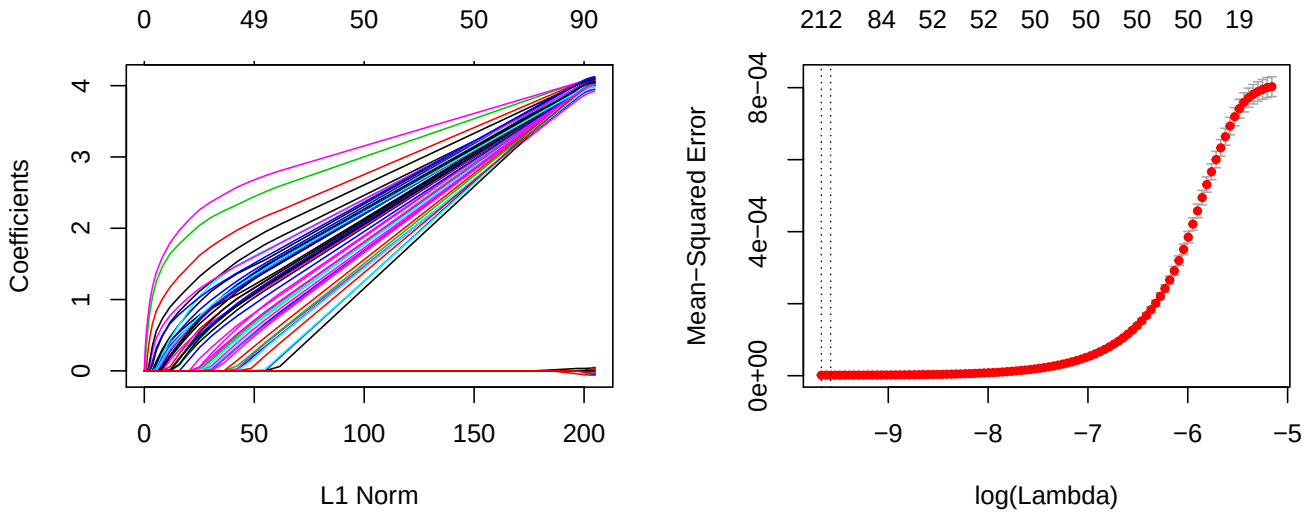
Abbildung 16 Graphische Darstellung der Ergebnisse von LASSO und SLOPE mit einer orthogonalen Systemmatrix bei weak signals.

trix bei weak signals beide Verfahren weniger gut performen. Anhand der Nichtnulleinträge des Koeffizientenvektors lässt sich nicht mehr ablesen, welche Variablen falsch als relevant einsortiert wurden. Da jedoch SLOPE bei Vorliegen orthogonaler Spalten und bekanntem σ die Kontrolle der FDR garantiert, ist es verlässlicher als LASSO, auch wenn es nicht zwingend alle relevanten Variablen findet.

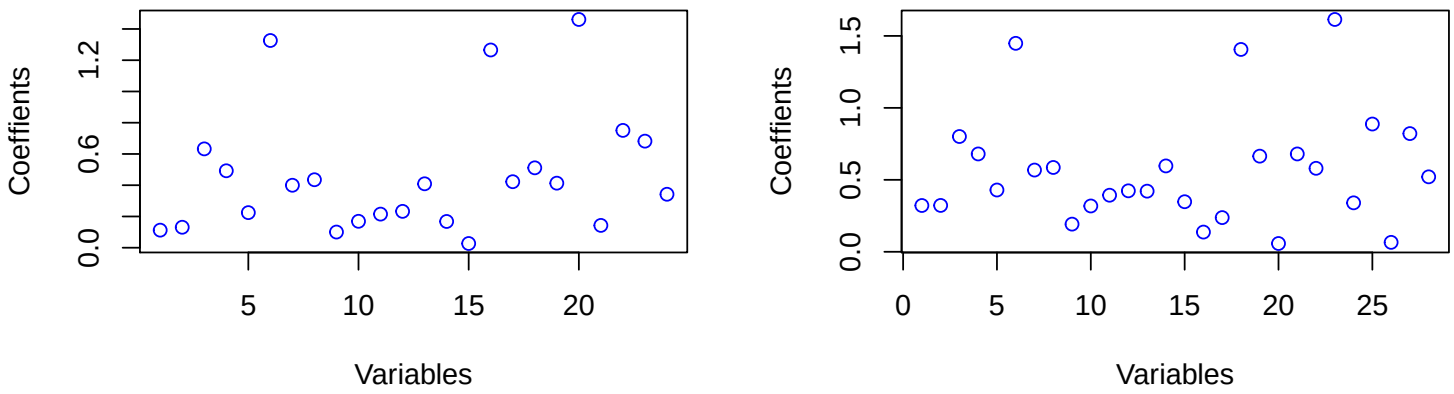
Als nächstes die analoge Analyse mit `X.gauss`. Der Wert der Nichtnulleinträge von β^* ist hierbei 4.127. Berechne wie bislang die jeweiligen Objekte:

```
> obj = glmnet(X.gauss, y, family="gaussian")
> cv.obj = cv.glmnet(X.gauss, y, family="gaussian")
> slope1 = SLOPE(X.gauss, y, fdr=0.1, lambda="gaussian",
                 sigma=1/n, normalize=FALSE)
> slope2 = SLOPE(X.gauss, y, fdr=0.2, lambda="gaussian",
                 sigma=1/n, normalize=FALSE)
```

Abbildung 17 stellt die Ergebnisse graphisch dar. Der CV Plot von `cv.obj` 17a zeigt, dass die Anzahl der zu RP `lambda.min` und `lambda.1se` gehörenden Variablen viel höher als 50 ist. Der Regressionsvektor zu `lambda.1se` hat 150 Nichtnulleinträge,



(a) LASSO: Koeffizientenpfad (links) und CV Plot (rechts).



(b) SLOPE: Nichtnulleinträge von zwei berechneten Lösungsvektoren. Links: `fdr=0.1`; rechts: `fdr=0.2`.

Abbildung 17 Graphische Darstellung der Ergebnisse von LASSO und SLOPE bei weak signals mit einer Systemmatrix, deren Einträge normalverteilt sind.

wobei die relevanten Variablen darin enthalten sind. Der betragsmäßig größte Wert der unter den als falsch relevant ausgewiesenen Variablen beträgt allerdings 0.05951, wobei die Werte der relevanten Variablen zwischen 3.9033 und 4.1092 liegen. Somit heben sich diese stark hervor, sodass der Anwender sofort die 50 relevanten unter den 150 selektierten Variablen finden sollte. `slope1` wählt lediglich 24, `slope2` 28 Variablen, wobei diese zu den relevanten Variablen gehören. Die FDP bei SLOPE ist damit zwar Null, es wurden jedoch 26 bzw. 22 relevante Variablen den irrelevanten zugeordnet. Zudem sind die Werte der Koeffizienten bedeutend kleiner als 4.127.

Diese Sachverhalte haben zur Folge, dass LASSO in diesem Beispiel sowohl bzgl. der Variablenselektion als auch der Vorhersage und der Approximation von β^* ein (wesentlich) besseres Verhalten aufweist.

Dass SLOPE so viele relevante Variablen nicht als solche erkennt, liegt hauptsächlich an der Höhe der Werte der Sequenz $\boldsymbol{\lambda}_G$. Der Einsatz der Sequenz $\boldsymbol{\lambda}_{BH}$

```
> slope = SLOPE(X.gauss, y, fdr=0.2, lambda="bhq",
                sigma=1/n, normalize=FALSE)
```

hat zur Folge, dass nun 37 Variablen, die alle relevant sind, selektiert werden, sodass 13 Variablen den irrelevanten zugeordnet wurden. Da die Regularisierung mit $\boldsymbol{\lambda}_{BH}$ sanfter als mit $\boldsymbol{\lambda}_G$ ist, werden mehr Variablen selektiert. Die Einträge sind jedoch wie bei $\boldsymbol{\lambda}_G$ wesentlich kleiner als der tatsächliche Wert von 4.127.

Die Ergebnisse dieses Unterkapitels sind in der Tabelle 7 zusammengefasst. Der

	Datensatz	**fdr**	**fdp**	**Laufzeit**	**#Variablen**	$\|\hat{\beta} - \beta^*\|$	$\|\hat{y} - y\|$
LASSO	`X.orth`		0.612	0.313 / 4.311	$151 - 260/$ $106 - 129$	15.154	31.733
SLOPE	`X.orth`	0.1	0.09	0.606	55	22.527	37.963
SLOPE	`X.orth`	0.2	0.167	0.182	60	21.096	36.967
LASSO	`X.gauss`		0	0.83 / 7.417	50	4.866	0.147
SLOPE	`X.gauss`	0.1	0	48.794	50	30.127	0.941
SLOPE	`X.gauss`	0.2	0	47.312	50	31.848	0.89
LASSO	`X.orth`		0.361	0.459 / 6.912	$130 - 201/$ $72 - 93$	17.904	33.871
SLOPE	`X.orth`	0.1	0.07	0.203	43	22.469	37.25
SLOPE	`X.orth`	0.2	0.106	0.218	47	21.324	36.476
LASSO	`X.gauss`		0.583	1.147 / 25.175	$201 - 213/$ $150 - 182$	0.803	0.032
SLOPE	`X.gauss`	0.1	0	14.505	24	27.732	0.838
SLOPE	`X.gauss`	0.2	0	14.452	28	27.091	0.815

Tabelle 7 Ergebnisse der Analyse von LASSO und SLOPE für synthetische Daten, wobei 50 relevante Variablen vorliegen.

obere Teil der Tabelle bezieht sich auf $\beta^*_{J^*} = 5\sqrt{2\log(p)}$ und $\beta^*_j = 10\sqrt{2\log(p)}$ für ein $j \in J^*$ (strong signals), der untere Teil auf $\beta^*_{J^*} = \sqrt{2\log(p)}$ (weak signals). Bei LASSO ist die Laufzeit als die für die Berechnung des Koeffizientenpfades mittels `glmnet` und zusätzlicher Durchführung von CV mittels `cv.glmnet` (Laufzeit `glmnet` / `cv.glmnet`) zu verstehen. Die Anzahl der Variablen ist in Abhängigkeit

von `lambda.min` / `lambda.1se` nach 10-facher Anwendung von `cv.glmnet` angegeben. Die letzten beiden Spalten sind bei LASSO in Abhängigkeit von `lambda.1se` aufgeführt, wobei dafür das mittels `cv.glmnet` berechnete Objekt mit der kleinsten Anzahl an Variablen verwendet wurde, hier somit 106 (Zeile 1 des oberen Blocks), 72 (Zeile 1 des unteren Blocks) und 150 (Zeile 4 des unteren Blocks).

4.2.2 Genexpressionen

Bei SLOPE erfolgt der Einsatz der Sequenz $\hat{\boldsymbol{\lambda}}_{BH}$, denn $\hat{\boldsymbol{\lambda}}_{G}$ ergibt für die vorliegenden Genexpressionsmatrizen eine konstante Folge. Außerdem erfolgen die Berechnungen nur für die FDR von 0.2, weil bereits bei den synthetischen Daten in einigen Fällen die Setzung `fdr=0.1` dazu geführt hat, dass zu wenige Variablen selektiert wurden.

Da für die Genexpressionen keine Informationen über deren Verteilung oder über die Verteilung des Outputs vorhanden sind, muss eine Standardisierung des Systems vorgenommen werden, wobei dies intern in den jeweiligen Funktionen geschieht. Um Vorhersagen machen zu können, bedarf die Lösung bei SLOPE daher einer Umrechnung auf die Originalskala. Der Output `os` (`os.camda, os.brca`) steht hier für Overall Survival. Die Überlebenszeit in Tagen liegt bei CAMDA zwischen 12 und 3809, bei TCGA zwischen 116 und 6593.

Zunächst folgt die Analyse der CAMDA Genexpressionen.

```
> obj = glmnet(CAMDA_dead, os.camda, family="gaussian")
> cv.obj = cv.glmnet(CAMDA_dead, os.camda, family="gaussian")
> slope = SLOPE(CAMDA_dead, os.camda, fdr=0.2, lambda="bhq")
```

Die graphische Darstellung der Ergebnisse zeigt die Abbildung 18. `slope` liefert 49 Gene, `cv.obj` 35 zu `lambda.min` und 8 zu `lambda.1se`, wobei `cv.obj` zu `lambda.min` und `slope` 27 gemeinsame Gene enthalten. Anhand der Abbildung ist ersichtlich, dass die Koeffizienten der Lösungsvektoren von LASSO betragsmäßig deutlich größer sind als die des Lösungsvektors von SLOPE.

Die äußere CV findet 10-fach mittels eigens geschriebener Funktionen `cv.lasso` und `cv.slope` statt, wobei dieselben CV-Teilmengen für LASSO und SLOPE benutzt werden. Der Schnitt ausgewählter Gene zu `lambda.min` bei LASSO nach 10-facher (äußerer) CV enthält lediglich ein Gen:

```
[1] "sneetoy:Gene_AceView"
```

Bei SLOPE tauchte in einer CV-Iteration die folgende Fehlermeldung auf:

```
Error in SLOPE(X, y, lambda="gaussian", :
  Selected >= n-1 variables. Cannot estimate variance.
  In addition: Warning message:
```

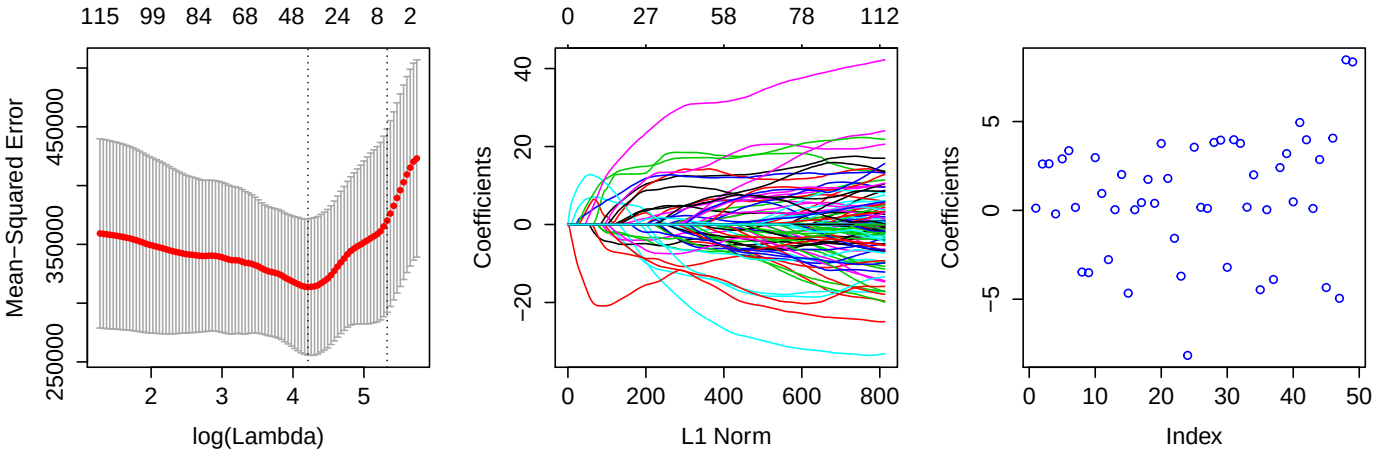


Abbildung 18 CAMDA und TCGA: CV Plot (links) und Koeffizientenpfad (Mitte) resultierend aus LASSO, Nichtnulleinträge des Lösungsvektors (rechts) resultierend aus SLOPE.

```
In SLOPE_solver(...): SLOPE solver reached iteration limit
```

Wird diese außer Acht gelassen, so enthält der Schnitt von SLOPE nach 10-facher CV hingegen 15 Gene:

```
 [1] "KCNH5:Gene_AceView"      "rawju:Gene_AceView"
 [3] "sneetoy:Gene_AceView"    "XAGE1D:Gene_RefSeq"
 [5] "XAGE1C:Gene_AceView"     "kitumu:Gene_AceView"
 [7] "PNMA5:Gene_RefSeq"       "storjoyby:Gene_AceView"
 [9] "MTNR1A:Gene_AceView"     "slerroy:Gene_AceView"
[11] "zuder:Gene_AceView"      "TOP3B2:Gene_AceView"
[13] "gleyglar:Gene_AceView"   "watemu:Gene_AceView"
[15] "ratemu:Gene_AceView"
```

Damit ist SLOPE in diesem Beispiel bezüglich der Variablenselektion konsistenter, trotz der nicht adaptiv gewählten Sequenz $\hat{\boldsymbol{\lambda}}_{BH}$.

Nun zur Analyse des microRNA Datensatzes für Brustkrebs.

```
> cv.obj = cv.glmnet(BRCA_dead, os.brca, family="gaussian")
> slope = SLOPE(BRCA_dead, os.brca, fdr=0.2, lambda="bhq")
```

Der sowohl aus `cv.obj` zu `lambda.min` als auch `slope` resultierende Lösungsvektor ist $0_{|p|}$, sodass keine Gene selektiert wurden. Dies erklärt die sehr geringe Laufzeit, siehe Tabelle 8. Bei `cv.obj` stimmen `lambda.min` und `lambda.1se` überein und entsprechen dem λ_0, mit dem die Sequenz der RP bei `cv.glmnet` startet. Bei 10-facher Anwendung von `cv.glmnet` wurden in vier CV-Iterationen Gene zu `lambda.min` selektiert, und zwar jeweils zwei und fünf Gene. Diese sind:

```
> selected.lasso[1:2]
```

```
[[1]]
[1] "hsa-mir-216a-5p" "hsa-mir-628-3p"
[[2]]
[1] "hsa-mir-125a-3p" "hsa-mir-216a-5p" "hsa-mir-301b-3p"
[4] "hsa-mir-30a-3p"  "hsa-mir-628-3p"
```

Nun wird 10-fache (äußere) CV durchgeführt, um zu überprüfen, ob hierbei ebenso keine oder wenige Gene ausgewählt werden. Bei der Anwendung von `cv.lasso` schwankt die Anzahl der ausgewählten Gene zwischen 0 und 8. Bei der CV mittels `cv.slope` wurde in zwei CV-Iterationen ein Gen selektiert, und zwar:

```
[1] "hsa-mir-216a-5p"
```

Da sowohl bei LASSO als auch bei SLOPE nur wenige oder keine Gene ausgewählt wurden, gilt für die Norm des Residuums der Testmenge $\|y_{(T_i)} - X^{(T_i)}\hat{\beta}^{(-T_i)}\|_2 \approx \|y_{(T_i)}\|_2$, sodass die vorhergesagte Überlebenszeit (fast) aller Patienten Null Tage beträgt. Die Vorhersage der Überlebenszeit für den jeweiligen Testdatensatz bei der CV hat daher wenig Sinn ergeben, was anhand der Fehlermaße umso ersichtlicher ist.

Die Ergebnisse der Analyse der Genexpressionen fasst die Tabelle 8 zusammen. Die Spalten "Laufzeit" und "#Gene" sind analog zur vorherigen Analyse. Die Spalte

	Datensatz	**Laufzeit**	**#Gene**	**CV-#Gene**	**CV-Error**
LASSO	CAMDA	0.042 / 0.458	$14-80$/ $0-12$	$10-100$/ $0-10$	$407.073-987.190$/ $328.496-1009.697$
SLOPE	CAMDA	9.423	49	$43-58$	$630.08-1393.466$
LASSO	TCGA	0.028 / 0.325	$0-5$/ 0	$0-8$/ 0	$909.355-1884.11$/ $933.94-2757.623$
SLOPE	TCGA	0.02	0	$0-1$	$933.94-2757.623$

Tabelle 8 Performance von LASSO und SLOPE bei CAMDA und TCGA Genexpressionen.

"CV-#Gene" gibt die Spanne der in der 10-fachen (äußeren) CV Anzahl selektierter Gene, bei LASSO in Abhängigkeit von `lambda.min`/`lambda.1se`. "CV-Error" für die Vorhersage sowohl für LASSO als auch für SLOPE wurde folgendermaßen berechnet:

$$\text{CV-Error} = \frac{1}{\sqrt{|T_i|}}\|y_{(T_i)} - X^{(T_i)}\hat{\beta}^{(-T_i)}\|_2 \ . \tag{4.1}$$

Schlussfolgerung Als Schlussfolgerung dieser Analyse lässt sich sagen, dass im

Falle idealisierter Daten beide Ansätze recht gut performen, wobei die Größenordnung der relevanten Merkmale bekannt sein sollte. Es gab jedoch Beispiele, in denen bei künstlich generierten Daten LASSO ein besseres Verhalten als SLOPE gezeigt hat. Wie in der Theorie beschrieben, ist SLOPE bei orthogonalen Designs konsistent bezüglich Variablenselektion. Dies wurde anhand der Beispiele bestätigt.

Insbesondere bei realen Datensätzen ergab sich, dass LASSO bezüglich der Merkmalsselektion teilweise chaotisch agierte. Dessen Instabilität wurde in den meisten Beispielen bestätigt. Dies ist unter anderem auf die hohen Dimensionen zurückzuführen. Wenn nur wenige Beobachtungen im Vergleich zu Merkmalen vorliegen, kann sich die zur Analyse verwendete Teilmenge stark auf das Ergebnis auswirken. Dies wurde ebenfalls belegt. Das Kapitel 2.6 besagte zudem, dass für LASSO die Irrepresentable Condition eine wichtige Rolle für die Konsistenz der Merkmalsselektion spielt. Da die Variablenselektion bei LASSO bei den Genexpressionen meist starken Schwankungen unterlag, lässt vermuten, dass die Irrepresentable Condition für diese Datensätze nicht erfüllt ist.

Bei der praktischen Gegenüberstellung von LASSO und SLOPE wurde umso deutlicher, dass SLOPE in erster Linie auf die Variablenselektion und LASSO bei der Bestimmung von `lambda.min` und `lambda.1se` eher auf die Minimierung der Norm des Residuums abzielt. Daher ist der Vorhersagefehler bei LASSO zwar zum Teil geringer, SLOPE ist jedoch hinsichtlich der Merkmalsselektion verlässlicher.

Eine wichtige Komponente ist die für eine Lösung benötigte Rechenzeit. In dieser Hinsicht braucht LASSO viel weniger Aufwand als SLOPE. Das R-Paket **glmnet** bindet eine effiziente Implementierung in der Programmiersprache Fortran ein, **SLOPE** ist hingegen gänzlich in R geschrieben. Hinzu kommt bei LASSO die Eliminierung einer hohen Anzahl an Merkmalen a priori und der Einsatz von Warm Starts, siehe Kapitel 2.5. Außerdem werden Lösungen für eine vom Anwender vorgegebene Anzahl m (voreingestellt sind $m = 100$) an RP berechnet, für jedes beliebige andere $\lambda \notin \{\lambda_1, ..., \lambda_m\}$ sind die dazugehörigen Lösungen durch Interpolation bestimmbar. Mit `cv.glmnet` erfolgt zudem neben der Berechnung des Koeffizientenpfades zusätzlich CV. `SLOPE` im Gegenteil berechnet nur eine Lösung für eine Sequenz an RP.

Die praktische Analyse bestätigte die Theorie, die besagt, dass die Systemmatrix und der Output gewisse Kriterien erfüllen müssen, um stabile Aussagen über eine gefundene Lösung und die zugehörigen Merkmale treffen zu können.

5 Fazit und Ausblick

Diese Arbeit hat LASSO und SLOPE gegenübergestellt, sowohl theoretisch als auch in der praktischen Anwendung.

In der Theorie wurde diskutiert, über welche Charakteristika das Optimierungsproblem verfügen muss, um ein aussagekräftiges Ergebnis zu erhalten. Die meisten vorgestellten möglichen Kriterien, die bestimmte Eigenschaften zusichern, gehen von einer gewissen Struktur, Verteilung und konkreten Annahmen aus, sodass von der Richtigkeit dieser Eigenschaften für praktische Probleme, die diese Kriterien aufweisen, ausgegangen werden kann. Die Eindeutigkeit der Lösung ist oft eine wichtige Voraussetzung, was nur bei $n \geq p$ und $\text{rang}(X) = p$ zutrifft. Der Fokus hier liegt auf Problemen, bei denen die Anzahl der Merkmale stark die der Beobachtungen übersteigt, sodass diese Voraussetzung und die damit geltenden Eigenschaften nichtig sind.

SLOPE erlaubt unterschiedliche Regularisierung von Merkmalen, sodass im Gegensatz zu LASSO Merkmale hinsichtlich ihrer Wichtigkeit behandelt werden können. Das Verfahren scheint in einigen Fällen vielversprechend zu sein, bis jetzt existieren allerdings nur wenige konkrete Ergebnisse. SLOPE garantiert Stabilität bei Optimierungsproblemen mit Matrizen, deren Spalten orthogonal sind, und solchen, deren Einträge normalverteilt sind. Dabei ist jedoch die Standardabweichung σ des Outputs zu berücksichtigen, weil diese eine große Rolle spielt. Schließlich hängen die Sequenzen $\sigma\boldsymbol{\lambda}_{BH}$ und $\sigma\hat{\boldsymbol{\lambda}}_G$ von σ ab. Wenn die Systemmatrix die gewünschten Eigenschaften hat, σ problemlos bestimmt werden kann, so sollte SLOPE LASSO vorgezogen werden. FDR, die von SLOPE in solchen Problemen kontrolliert wird, ist eine bedeutende Größe in statistischen Analysen.

Eine Schlussfolgerung aus der praktischen Anwendung für SLOPE ist dessen meist wesentlich höhere Konsistenz bezüglich der Merkmalsselektion. LASSO bietet wiederum bessere Vorhersagen. Bis jetzt wurden bei SLOPE keine Möglichkeiten erarbeitet, wie Dimensionsreduktion im Vorhinein durchgeführt werden kann. Dadurch könnte SLOPE viel effizienter sein und den benötigten Aufwand zur Berechnung einer Lösung reduzieren. Im Kapitel 2.5 wurden für LASSO hingegen Methoden präsentiert, wie Merkmale a priori mit einer gewissen Sicherheit entfernt werden können. Zudem bewältigt LASSO, auch wenn es instabil ist, hohe Dimensionen besser als SLOPE. Mit LASSO lassen sich zumindest erste Schlussfolgerungen ziehen. Außerdem kann sich der Anwender mit einer Reihe von Lösungen auseinandersetzen.

Es bleiben viele offene Fragen, wie die Feststellung der Abhängigkeit und Anzahl relevanter Merkmale. Bei dem Beispiel der Genexpressionen ist die Gefahr aufgrund einer im Vergleich zur Anzahl der Gene geringen Zahl von Patienten recht hoch,

dass sich die Verteilung bei Hinzufügen des Genprofils eines weiteren Patienten ändert. Außerdem gibt es bis zum jetzigen Zeitpunkt kaum Ergebnisse bezüglich der Performance und Interpretierbarkeit von LASSO und SLOPE bei nicht linearem Verhalten.

Um SLOPE gänzlich als ein gängiges Verfahren zu etablieren, muss dessen Verhalten bei allgemeinen Optimierungsproblemen untersucht und müssen entsprechende Sequenzen der RP erarbeitet werden. Ein interessanter Aspekt dabei ist das Umgehen mit korrelierten Merkmalen, bei dem LASSO scheitert. LASSO setzt Regression um, bei der der Fehlerterm ϵ eine Verteilung aus der Klasse der exponentiellen Familien aufweist. SLOPE ist nur für die klassische lineare Regression einsetzbar und bedarf einer Transformation auf GLM.

Zahlreiche Wissenschaftler haben bislang großartige Arbeit geleistet, die in vielen Gebieten zu Durchbrüchen geführt hat. Ein Verfahren, dass alle Eventualitäten abdeckt und den Fall $n \ll p$ problemlos meistert, existiert jedoch noch nicht. Die Regression mit unverhältnismäßiger Anzahl an Beobachtungen zu Merkmalen bleibt weiterhin eine Herausforderung.

Literatur

[1] Andrade JF, de Campos MLR, Apolinario JA (2010), A complex version of the LASSO algorithm and its application to beamforming, 7th International Telecommunications Symposium.

[2] Beck A, Teboulle M (2009), A fast iterative shrinkage-thresholding algorithm for linear inverse problems, Society for Industrial and Applied Mathematics, Vol. 2, No. 1, 183–202.

[3] Bertsekas D (1999), Nonlinear Programming, Athena Scientific, Belmont, MA, 2nd ed..

[4] Bogdan M, van den Berg E, Su W, Candes EJ (2013), Statistical Estimation and Testing via the Ordered l_1 Norm, arXiv:1310.1969.

[5] Bogdan M, van den Berg E, Sabatti C, Su W, Candes EJ (2015), SLOPE - Adaptive Variable Selection via Convex Optimization, The Annals of Applied Statistics, Vol. 9, No. 3, 1103-1140.

[6] Bogdan M, van den Berg E, Sabatti C, Su W, Candes EJ, Patterson E (2015), Sorted L1 Penalized Estimation (SLOPE, R Package Version 0.1.3, https://CRAN.R-project.org/package=SLOPE.

[7] Boyd S, Vandenberghe L (2004), Convex Optimization, Cambridge University Press.

[8] Breimann L (1995), Better Subset Selection Using the Nonnegativ Garrote, Technometrics, Vol. 37, No. 4, 373-384.

[9] Bühlmann P, van de Geer S (2011), Statistics for High-Dimensional Data: Methods, Theory and Applications, Springer-Verlag Berlin Heidelberg.

[10] Cox D (1972), Regression models and life tables, Journal of the Royal Statistical Society, Series B, Vol. 34, No. 2, 187–220.

[11] Cook RD (2011), On the mean and variance of the generalized inverse of a singular Wishart matrix, Electronic Journal of Statistic, Vol. 5, 146–158.

[12] Efron B, Hastie T, Johnstone I, Tibshirani R (2004), Least angle regression, The Annals of Statistics, Vol. 22, No. 4, 1947-1975.

[13] El Ghaoui L, Viallon V, Rabbani T (2010), Safe feature elimination in sparse supervised learning, Technical Report UC/EECS-2010-126, EECS Dept., University of California at Berkeley.

[14] Friedman J, Hastie T, Hoefling H, Tibshirani R (2007), Pathwise Coordinate Optimization, The Annals of Applied Statistics, Vol. 2, No. 1, 302–332.

[15] Friedman J, Hastie T, Tibshirani R (2010), Regularization Paths for Generalized Linear Models via Coordinate Descent, Journal of Statistical Software, Vol. 33, No. 1, 1-21.

[16] Friedman J, Hastie T, Tibshirani R (2018), glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models, R Package version 2.0-16, https://CRAN.R-project.org/package=glmnet.

[17] Hämmerlin G, Hohmann A (1989), Numerische Mathematik, Springer.

[18] Hastie T, Tibshirani R (1990), Generalized Additive Models, Chapman and Hall, London.

[19] Hastie T, Tibshirani R, Friedman J (2009), The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Series in Statistics, 2nd ed..

[20] Hastie T, Tibshirani R, Narasimhan B, Chu G (2018), impute: Imputation for microarray data, R package version 1.54.0.,https://bioconductor.org/packages/release/bioc/html/impute.html.

[21] Hastie T, Tibshirani R, Wainwright M (2016), Statistical Learning with Sparsity: The Lasso and Generalizations, CRC Press.

[22] Hoerl A, Kennard R (1970), Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics, Vol. 12, No. 1, 55-67.

[23] Ipsen I (2009), Numerical matrix analysis : linear systems and least squares, Society for Industrial and Applied Mathematics.

[24] Kliewer V, Lee S (2016), EasyTCGA: An R package for easy batch downloading of TCGA data from FireBrowse, No. 4, TU Dortmund.

[25] Kremer P J, Lee S, Bogdan M, Paterlini S (2017), Sparse Portfolio Selection via the sorted l1-Norm, arXiv:1710.02435.

[26] Lee J, Sun Y, Saunders M (2014), Proximal newton-type methods for minimizing composite functions, Society for Industrial and Applied Mathematics, Vol. 24, No. 3, 1420–1443.

[27] Leng C, Lin Y, Wahba G (2006), A note on the Lasso and related procedures in model selection, Statistica Sinica, Vol. 16, 1273-1284.

[28] Nesterov Y E (1983), A method for solving the convex programming problem with convergence rate $O(1/k2)$, Soviet mathematics - Doklady, Vol. 27, No. 2, 543–547.

[29] Ollila E (2016), Direction of arrival estimation using robust complex Lasso, 10th European Conference on Antennas and Propagation.

[30] Parikh N, Boyd Stephen (2013), Proximal Algorithms, Foundations and Trends in Optimization, Vol. 1, No. 3, 123-231.

[31] Prokhorov AV (2002) Wishart distribution, Encyclopaedia of Mathematics, Springer-Verlag Berlin.

[32] Roberts S, Nowak G (2014), Stabilizing the lasso against cross-validation variability, Computational Statistics and Data Analysis, Vol. 70, 198–211.

[33] Simon N, Friedman J, Hastie T, Tibshirani R (2011), Regularization Paths for Cox's Proportional Hazard Model via Coordinate Descent, Journal of Statistical Software, Vol. 39, No. 5, 1-13.

[34] Stoer J (2002), Numerische Mathematik 1, Springer.

[35] Tibshirani R (1996), Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society, Series B (Methodological), Vol. 58, No. 1, 267-288.

[36] Tibshirani RJ (2013), The Lasso Problem and Uniqueness, Electronic Journal of Statistics, Vol. 7, 1456-1490.

[37] Tibshirani RJ (with Wasserman L) (2015), Sparsity and the Lasso, Statistical Machine Learning, Spring.

[38] Tibshirani R, Bien J, Friedman J, Hastie T, Simon N, Taylor J, Tibshirani RJ (2010), Strong Rules for Discarding Predictors in Lasso-type Problems, Journal of the Royal Statistical Society, Series B (Methodological), 245–266.

[39] Tibshirani RJ, Taylor J (2012), Degrees of freedom in lasso problems, The Annals of Statistics, Vol. 40, No. 2, 1198-1232.

[40] Tseng P (2001), Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimazation, Journal of Optimization Theory and Applications, Vol. 109, No. 3, 475-494.

[41] Wright SJ (2015), Coordinate descent algorithms, Journal of Mathematical Programming, Vol. 151, No. 1, 3-34.

[42] Xu H, Caramanis C, Mannor S (2011), Sparse algorithms are not stable: A no-free-lunch theorem, Pattern Analysis and Machine Intelligence, IEEE Transactions, Vol. 34, No. 1, 187–193.

[43] Yuan M, Lin Y (2006), Model selection and estimation in regression with grouped variables, Journal of the Royal Statistical Society, Series B (Methodological), Vol. 68, No.1, 49–67.

[44] Zhang Y, Ray S, Guo W (2016), On the Consistency of Feature Selection With Lasso for Non-linear Targets, 33rd International Conference on Machine Learning, Vol. 1, 322-330.

[45] Zhao P, Yu B (2006), On Model Selection Consistency of Lasso, Journal of Machine Learning Research, Vol. 7, 2541-2563.

[46] Zou H, Hastie T (2005), Regularization and Variable Selection via the Elastic Net, Journal of the Royal Statistical Society, Series B (Methodological), Vol. 67, No. 2, 301–320.

A1 Beispiele linearer Regression

In diesem Kapitel werden Beispiele klassischer linearer Regression ohne Regularisierung gezeigt.

Die Hyperebene entspricht in zwei Dimensionen einer Gerade, in drei Dimensionen entweder einer Gerade oder einer Ebene. Die Abbildung 19 zeigt Beispiele, wann in zwei und drei Dimensionen die vorliegenden Beobachtungen perfekt durch eine Gerade oder Ebene angepasst werden können. In zwei Dimensionen ist eine Gerade gesucht, die die vorhandenen Datenpunkte am besten anpasst. Im Beispiel 1 in der Abbildung 19 ist ein einziger Datenpunkt vorhanden, der auf unendlich vielen Geraden liegt. Im Beispiel 2 sind zwei Datenpunkte vorhanden, die auf genau einer Gerade liegen. Beispiel 3 veranschaulicht dieselben Sachverhalte in drei Dimensionen. Sind nur die roten Datenpunkte gegeben, so liegen diese auf unendlich vielen Ebenen, bspw. auf der roten und blauen Ebene, und auf genau einer Gerade, kommt ein dritter Datenpunkt dazu, hier der blaue Punkt, so liegen die drei Datenpunkte auf genau einer Ebene, und zwar auf der blau eingezeichneten Ebene.

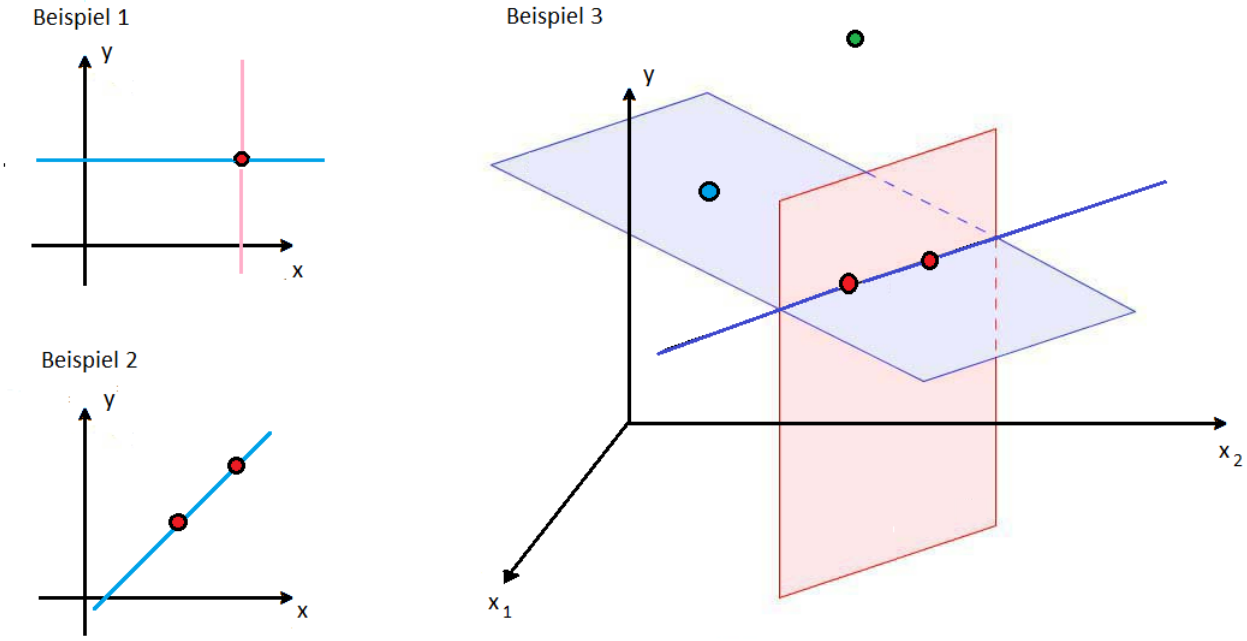


Abbildung 19 Beispiele linearer Regression in 2D und 3D mit $n \leq p$.

Solchen Beispielen (in beliebiger Dimension) liegt somit ein lineares Gleichungssystem mit einer Matrix $X \in \mathbb{R}^{n \times p}$ zugrunde, das weniger Gleichungen ($n < p$, System ist unterbestimmt) als Unbekannte oder genauso viele Unbekannte ($n = p$) besitzt. Ein unterbestimmtes System hat unendlich viele Lösungen. Bei einem quadratischen System, dessen Systemmatrix invertierbar ist, existiert genau eine Lösung. Diese einfachen Beispiele sollen veranschaulichen, dass sich bei Hinzufügen

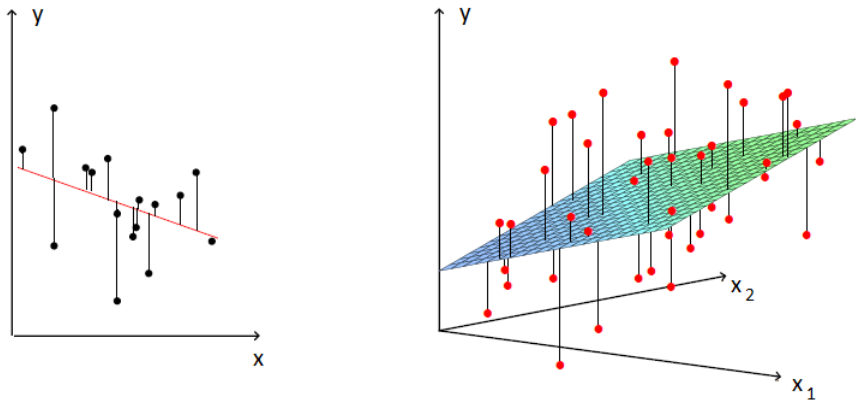


Abbildung 20 Beispiele linearer Regression in 2D und 3D. Die Graphik links wurde mit R erstellt, die Graphik rechts ist aus [19], S.45.

einer neuen Beobachtung die gesuchte Hyperebene eine völlig andere sein kann.

Anhand dieser einfachen Beispiele wird deutlich, dass ausreichend Beobachtungen nötig sind, um eine aussagekräftige Analyse zu erlauben. Kommt in dem dreidimensionalen Beispiel der Abbildung 19 zu dem blauen und den roten Datenpunkten eine Beobachtung hinzu, hier der grüne Punkt, so ist die Ebene, die die 4 Punkte am besten anpasst, eine völlig andere. Die Abbildung 20 zeigt Beispiele in zwei und drei Dimensionen, in denen n deutlich größer als p ist. In solchen Fällen lässt sich zumindest eine Behauptung über den Zusammenhang der Daten aufstellen. In der Abbildung 20 ist deutlich zu erkennen, dass ein linearer Zusammenhang der Daten postuliert werden kann, sodass die bestimmte Gerade bzw. Ebene die Daten recht gut approximiert und somit verlässliche Vorhersagen für neue Beobachtungen erlaubt.

A2 Definition der Pseudoinverse

Die Definition geht auf [34] zurück. Für eine Matrix $X \in \mathbb{C}^{n\times p}$, die singulär und/oder nicht quadratisch ist, bezeichnet X^+ die Pseudoinverse von X, die bei quadratischen regulären Matrizen ihrer Inversen X^{-1} entspricht. Die Pseudoinverse existiert für jede beliebige Matrix und ist eindeutig. Diese ist gekennzeichnet durch die folgenden Eigenschaften:

1. $XX^+X = X$
2. $X^+XX^+ = X^+$
3. $(XX^+)^* = XX^+$
4. $(X^+X)^* = X^+X$

$(XX^+)^*$ bezeichnet die hermitesch adjungierte Matrix von XX^+. Ist rang(X)=p, so ist $X^+ = (X^TX)^{-1}X^T$ die Linksinverse von X mit $X^+X = I_p$, ist rang(X)=n,

so ist $X^+ = X^T(XX^T)^{-1}$ die Rechtsinverse von X mit $XX^+ = I_n$.

A3 Äquivalenz eines Optimierungsproblems mit Nebenbedingungen zur Lagrange-Dualität

Der Beweis richtet sich nach [7].

Es liege ein allgemeines nicht zwingend konvexes Optimierungsproblem mit Nebenbedingungen vor,

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ s.t. \quad & g(x) \leq 0, \\ & h(x) = 0 \ , \end{aligned} \tag{A3.1}$$

wobei $g(x) = (g_1(x), ..., g_p(x))^T$ und $h(x) = (h_1(x), ..., h_m(x))^T$ sind. Die Nebenbedingungen sind somit koeffizientenweise zu verstehen. Es bestehe die Annahme, dass der zulässige Definitionsbereich D für x, sodass die Nebenbedingungen erfüllt sind, nicht leer ist. Folglich existiert (mindestens) eine Lösung $x^* \in \text{argmin } f(x)$ von (A3.1). $f^* = f(x^*)$ sei der optimale Wert von $f(x)$.

Die zu zeigende *Lagrange-Dualität* besagt, dass die Nebenbedingungen mittels der sogenannten *Lagrange-Multiplikatoren* bzw. *dualen Variablen* λ und μ, in die Zielfunktion aufgenommen werden können. Die zu optimierende *Lagrange-Funktion* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$ lautet

$$L(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x) = f(x) + \sum_{j=1}^{p} \lambda_j g_j(x) + \sum_{i=1}^{m} \mu_i h_i(x)$$

Der zulässige Definitionsbereich von $L(x, \lambda, \mu)$ ist somit durch $D \times \mathbb{R}^m \times \mathbb{R}^p$ gegeben. λ_j ist der zugehörige Lagrange-Multiplikator zur Nebenbedingung $g_j(x) \leq 0$, $j = 1, ..., p$, $\mu_i(x)$ zur Nebenbedingung $h_i(x) = 0$, $i = 1, ..., m$.

Die *duale Funktion* zu (A3.1) ist definiert als

$$q(\lambda, \mu) = \inf_{x\in D} L(x, \lambda, \mu) = \inf_{x\in D} \Big(f(x) + \lambda^T g(x) + \mu^T h(x) \Big) \ .$$

Ist $q(\lambda, \mu)$ nicht beschränkt, so $q(\lambda, \mu) = -\infty$.

Untere Schranke für $\mathbf{f}^*$ Sei $\lambda \geq 0$, $\tilde{x} \in D$ beliebig, sodass $g_j(\tilde{x}) \leq 0 \ \forall j \in \{1, ...p\}$ und $h_i(\tilde{x}) = 0 \ \forall i \in \{1, ...m\}$. Daraus folgt

$$\sum_{j=1}^{p} \lambda_j g_j(\tilde{x}) + \sum_{i=1}^{m} \mu_i h_i(\tilde{x}) \ \leq 0 \ .$$

Für die Lagrange-Funktion ergibt sich

$$L(\tilde{x}, \lambda, \mu) = f(\tilde{x}) + \sum_{j=1}^{p} \lambda_j g_j(\tilde{x}) + \sum_{i=1}^{m} \mu_i h_i(\tilde{x}) \ \leq f(\tilde{x}) \ .$$

Dies wiederum führt zu

$$q(\lambda, \mu) = \inf_{x \in D} L(x, \lambda, \mu) \ \leq \ L(\tilde{x}, \lambda, \mu) \ \leq \ f(\tilde{x}) \ . \tag{A3.2}$$

Da die Ungleichung (A3.2) für alle $\tilde{x} \in D$ gilt, folgt

$$q(\lambda, \mu) \leq f^* \ , \tag{A3.3}$$

sodass die duale Funktion eine untere Schranke für f^* darstellt. Allerdings hat die untere Schranke nur dann Aussagekraft, wenn $q(\lambda, \mu) \neq -\infty$ gilt, was $\lambda \geq 0$ voraussetzt. Der Definitionsbereich für $q(\lambda, \mu)$ ist somit (x, λ, μ) $D \times (\mathbb{R}_0^+)^p \times \mathbb{R}^m$.

Das duale Problem zu (A3.1) lautet nun

$$\max q(\lambda, \mu) \ \text{ s.t. } \ \lambda \geq 0 \ .$$

$q^* = q(\lambda^*, \mu^*)$ bezeichne den optimalen Funktionswert von $q(\lambda, \mu)$, wobei λ^* und μ^* die optimalen Lösungsparameter des dualen Problems sind. Laut (A3.3) gilt $q^* \leq f^*$.

Nun bleibt die Frage, wann starke Dualität gilt, sodass Gleichheit $q^* = f^*$ erfüllt ist. Das primale Problem (A3.1) sei konvex. Eine mögliche Bedingung für Gleichheit ist die sogenannte *Slater-Bedingung*, die verlangt, dass ein $x \in D$ existiert, sodass $g(x) < 0$ gilt. Der Regularisierungsterm von LASSO bzw. SLOPE ist äquivalent dazu, dass der Definitionsbereich von f_{LASSO} bzw. f_{SLOPE} auf einen p-dimensionalen Hyperoktaeder bzw. ein symmetrisches konvexes p-dimensionales Polygon (ein Oktaeder ist ein Spezialfall eines konvexen Polygons) eingeschränkt ist. Die Gleichheit in den Nebenbedingungen gilt jeweils nur am Rand des Hyperoktaeders bzw. Polygons, für alle anderen zulässigen (engl. feasible) Punkte gilt strikte Ungleichheit, sodass die Slater-Bedingung erfüllt ist. Demzufolge kann zur Lösung von LASSO und SLOPE das jeweilige duale Problem eingesetzt werden.

A4 Lipschitz-Konstante der (skalierten) kleinsten Quadrate

Seien $f = \frac{1}{2n}\|y - X\beta\|_2^2$, $\nabla f = -\frac{1}{n}X^T(y - X\beta)$ und $\beta^1 \neq \beta^2 \in \mathbb{R}^p$. Es gilt

$$\begin{aligned}\|\nabla f(\beta^1) - \nabla f(\beta^2)\|_2 &= \| - \frac{1}{n}X^T(y - X\beta^1) + \frac{1}{n}X^T(y - X\beta^2)\|_2 \\ &= \frac{1}{n}\|X^T(X\beta^1 - X\beta^2)\|_2 = \frac{1}{n}\|X^TX(\beta^1 - \beta^2)\|_2 \\ &\leq \frac{1}{n}\|X^TX\|_2\|\beta^1 - \beta^2\|_2 = \frac{1}{n}\mu_{\max}(X^TX)\|\beta^1 - \beta^2\|_2 \ .\end{aligned}$$

Das letzte Gleichheitszeichen folgt aus der Symmetrie von X^TX. Folglich ist $L := \frac{1}{n}\mu_{\max}(X^TX)$ die Lipschitz-Konstante von ∇f.

Ohne den Vorfaktor $\frac{1}{n}$ ist die Lipschitz-Konstante für $f = \frac{1}{2}\|y - X\beta\|_2^2$ durch $L := \|X^TX\|_2 = \mu_{\max}(X^TX)$ gegeben.

A5 Taylor-Entwicklung

Es Funktion $f(\beta)$: $D \to \mathbb{R}$ sei auf dem Definitionsbereich D glatt, d.h. unendlich oft differenzierbar, wobei $D \subseteq \mathbb{R}^p$. Die Taylorreihe von f in $\hat{\beta} \in D$ lautet

$$Tf(\beta, \hat{\beta}) = \sum_{i=0}^{\infty} \frac{D^i f(\hat{\beta})}{i!}(\beta - \hat{\beta})^i = f(\hat{\beta}) + (\beta - \hat{\beta})^T\nabla f(\hat{\beta}) + \frac{1}{2}(\beta - \hat{\beta})^T H_f(\hat{\beta})(\beta - \hat{\beta}) + \dots \ ,$$

wobei $i! = i \cdot (i-1) \dots \cdot 2 \cdot 1$ die Fakultät von i und $D^i f$ das i-te Differential von f ist. Dabei sind $D^0 f := f$, $D^1 f := \nabla f$ der Gradient von f und $D^2 f := H_f$ die Hesse-Matrix von f. Das Taylor-Polynom n-ten Grades von f in $\hat{\beta}$ ist gegeben als

$$\begin{aligned}T_n f(\beta, \hat{\beta}) =& \sum_{i=0}^{\infty} \frac{D^i f(\hat{\beta})}{i!}(\beta - \hat{\beta})^i \\ &= f(\hat{\beta}) + (\beta - \hat{\beta})^T\nabla f(\hat{\beta}) + \frac{1}{2}(\beta - \hat{\beta})^T H_f(\hat{\beta})(\beta - \hat{\beta}) + \\ &\quad \dots + \frac{1}{n!}D^n f(\hat{\beta})(\beta - \hat{\beta})^n \ .\end{aligned}$$

Taylor-Polynome stellen eine lokale Approximation von f um $\hat{\beta}$ dar, sodass

$$f(\beta) \approx T_n f(\beta, \hat{\beta}) \ .$$

A6 Das duale Problem zu LASSO

Die folgenden Ergebnisse gehen auf [21] zurück.

Das Optimierungsproblem (2.6) sei das primale Problem zu LASSO. Das duale Problem kann in einigen Fällen von Vorteil sein, insbesondere lassen sich aus diesem

einige Schlussfolgerungen ziehen. Um das duale Problem aufzustellen, wird zunächst das Residuum $y - X\beta$ als r definiert, $r := y - X\beta$, sodass LASSO äquivalent ist zu

$$\min_{\beta\in\mathbb{R}^p, r\in\mathbb{R}^n} \quad \frac{1}{2n}\|r\|_2^2 + \lambda\|\beta\|_1 \;\text{ s.t. }\; \{\; r = y - X\beta \iff r - y + X\beta = 0 \;\} \; .$$

Aufgrund der Lagrange-Dualität lässt sich (A6.1) umformulieren zu

$$\begin{aligned}\min_{\beta\in\mathbb{R}^p, r\in\mathbb{R}^n} \Big\{ f(\beta, r, \theta) &:= \frac{1}{2n}\|r\|_2^2 + \lambda\|\beta\|_1 - \theta^T(r - y + X\beta) \\ &= \frac{1}{2n}\|r\|_2^2 - \theta^T r - \theta^T X\beta + \lambda\|\beta\|_1 + \theta^T y \Big\} \; , \end{aligned} \tag{A6.1}$$

wobei $\theta \in \mathbb{R}^n$ der eindeutige Lagrange-Parameter zur Nebenbedingung $r = y - X\beta$ ist. $\theta^T y$ hängt nicht von β oder r ab und wird daher zunächst außer Acht gelassen. Die Minimierung von $f(\beta, r, \theta)$ hinsichtlich β und r kann unabhängig voneinander stattfinden, wobei die Lösungen unmittelbar bestimmt werden können:

$$\min_{\beta\in\mathbb{R}^p} \; -\theta^T X\beta + \lambda\|\beta\|_1 = \begin{cases} 0, & \{\|X^T\theta\|_\infty = \max\limits_{j\in\{1,\dots,p\}} |X_j^T\theta|\} \;\le \lambda \; , \\ -\infty, & \text{sonst} \; , \end{cases} \tag{A6.2}$$

$$\min_{r\in\mathbb{R}^n} \; \frac{1}{2n}\|r\|_2^2 - \theta^T r \quad = -\frac{n-2}{2}\|\theta\|_2^2 \quad \text{mit } r = n\theta \; . \tag{A6.3}$$

Nach Einsetzen von (A6.2) und (A6.3) in die Optimierungsfunktion (A6.1) ist das duale Problem zu LASSO gegeben durch

$$\max_{\theta\in\mathbb{R}^n} \; \Big\{ f^D_{LASSO}(\theta) := -\frac{n-2}{2}\|\theta\|_2^2 + \theta^T y \Big\} \quad s.t. \;\; \|X^T\theta\|_\infty \le \lambda \; . \tag{A6.4}$$

In der Literatur wird das duale Problem ohne den Vorfaktor $\frac{1}{n}$ aufgestellt. Unter Vernachlässigung von $\frac{1}{n}$ ändern sich die entscheidenden Gleichungen (A6.3) und (A6.4) zu [13, 38]

$$\min_{r\in\mathbb{R}^n} \; \frac{1}{2}\|r\|_2^2 - \theta^T r \; = -\frac{1}{2}\theta^T\theta \quad \text{mit } r = \theta$$

$$\max_{\theta\in\mathbb{R}^n} \; \Big\{ f^D_{LASSO}(\theta) := -\frac{1}{2}\|\theta\|_2^2 + \theta^T y = \frac{1}{2}(\|y\|_2^2 - \|y - \theta\|_2^2) \Big\} \;\text{ s.t. }\; \|X^T\theta\|_\infty \le \lambda \; .$$

Das duale Problem projiziert y auf die Menge $F_\lambda = \{\theta \in \mathbb{R}^n \,|\|\, X^T\theta\|_\infty \le \lambda\}$, die ein konvexes n-dimensionales Polytop darstellt und aus dem Schnitt von $2p$ Halbräumen $\{|X_j\theta| \le \lambda\}_{j=1}^p$ entsteht.

A7 Beweis: Sortierte L1-Norm ist eine Norm

Durch Nachrechnen der Axiome soll gezeigt werden, dass die sortierte L1-Norm

$$J_\lambda(\beta) = \lambda_1|\beta|_{(1)} + ... + \lambda_p|\beta|_{(p)}$$

tatsächlich eine Norm ist, wobei $\lambda_1 \geq ... \geq \lambda_p \geq 0$.

Definitheit+Homogenität: Wegen $\lambda_j \geq 0 \ \forall j \in \{1, ...p, \}$ und der Eigenschaften des Betrags $|\cdot|$ ist sofort ersichtlich, dass $J_\lambda(\beta) = 0 \iff \beta = 0$ (Definitheit) und $J_\lambda(a\beta) = |a|J_\lambda(\beta) \ \forall a \in \mathbb{R}$ (Homogenität) gilt.

Konvexität: Seien α, $\beta \in \mathbb{R}^p$

$$\begin{aligned} J_\lambda(\alpha + \beta) &= \lambda_1|\alpha + \beta|_{(1)} + ... + \lambda_p|\alpha + \beta|_{(p)} \\ &\leq \lambda_1(|\alpha|_{(1)} + |\beta|_{(1)}) + ... + \lambda_p(|\alpha|_{(p)} + |\beta|_{(p)}) \\ &= \lambda_1|\alpha|_{(1)} + ... + \lambda_p|\alpha|_{(p)} + \lambda_1|\beta|_{(1)} + ... + \lambda_p|\beta|_{(p)} \\ &= J_\lambda(\alpha) + J_\lambda(\beta) \end{aligned}$$

A8 Beweis der Äquivalenz des Proximal Operators von SLOPE zu einem QP

Der folgende Beweis richtet sich nach [5].

$f(\beta) := \frac{1}{2}\|y - \beta\|_2^2 + J_\lambda(\beta)$ sei die in (3.12) zu optimierende Funktion, $\hat{\beta}$ sei die eindeutige Lösung, sodass $f(\hat{\beta})$ das globale Optimum der Funktion f ist, $f(\hat{\beta}) < f(\beta) \ \forall \beta \neq \hat{\beta}$. Es gelte $y_1 \geq ... \geq y_p$. Nehme an, dass es zwei Indizes i und j mit $i < j$ gibt, sodass $\hat{\beta}_i < \hat{\beta}_j$ und $y_i > y_j$. Vertausche die Einträge $\hat{\beta}_i$ und $\hat{\beta}_j$ und speichere diesen Vektor als $\tilde{\beta} = (\hat{\beta}_1, ..., \hat{\beta}_j, ..., \hat{\beta}_i, ..., \hat{\beta}_p)$ ab. Es gilt $J_\lambda(\hat{\beta}) = J_\lambda(\tilde{\beta})$. Es folgt

$$\begin{aligned} f(\hat{\beta}) - f(\tilde{\beta}) &= \frac{1}{2}\|y - \hat{\beta}\|_2^2 + J_\lambda(\hat{\beta}) - \frac{1}{2}\|y - \tilde{\beta}\|_2^2 - J_\lambda(\tilde{\beta}) \\ &= \frac{1}{2}\left(\sum_{k \notin \{i,j\}} (y_k - \hat{\beta}_k)^2 + (y_i - \hat{\beta}_i)^2 + (y_j - \hat{\beta}_j)^2 - \sum_{k \notin \{i,j\}} (y_k - \hat{\beta}_k)^2 - (y_i - \hat{\beta}_j)^2 + (y_j - \hat{\beta}_i)^2\right) \\ &= \frac{1}{2}\left((y_i - \hat{\beta}_i)^2 + (y_j - \hat{\beta}_j)^2 - (y_i - \hat{\beta}_j)^2 - (y_j - \hat{\beta}_i)^2\right) \\ &= \beta_j y_i - \beta_i y_i + \beta_i y_j - \beta_j y_j \\ &= (\beta_j - \beta_i)(y_i - y_j) \ > 0 \\ \iff \quad & f(\hat{\beta}) > f(\tilde{\beta}) \ , \end{aligned}$$

was einen Widerspruch zur Optimalität der eindeutigen Lösung $\hat{\beta}$ darstellt.

Eidesstattliche Versicherung
(Affidavit)

Name, Vorname
(Last name, first name)

Matrikelnr.
(Enrollment number)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem folgenden Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present Bachelor's/Master's* thesis with the following title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution.

Titel der Bachelor-/Masterarbeit*:
(Title of the Bachelor's/ Master's* thesis):

*Nichtzutreffendes bitte streichen
(Please choose the appropriate)

Ort, Datum
(Place, date)

Unterschrift
(Signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to €50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, section 63, subsection 5 of the North Rhine-Westphalia Higher Education Act (*Hochschulgesetz*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:**

Ort, Datum
(Place, date)

Unterschrift
(Signature)

**Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.