DeepLearning on FPGAs

Introduction to Data Mining

Sebastian Buschjäger

Technische Universität Dortmund - Fakultät Informatik - Lehrstuhl 8

October 18, 2016

Structure of this course

Goals:

→ Learning the basics of Data Mining

→ Learning the basics of Deep Learning

→ Learning the basics of FPGA programming

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

Structure of this course

Goals:

→ Learning the basics of Data Mining
→ Learning the basics of Deep Learning
→ Learning the basics of FPGA programming

Small lecture-phase in the beginning

- **Week 1 - 4:** Data Mining and Deep Learning
- **Week 4 - 6:** FPGAs and Software

[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

Structure of this course

Goals:

→ Learning the basics of Data Mining

→ Learning the basics of Deep Learning

→ Learning the basics of FPGA programming

Small lecture-phase in the beginning

- **Week 1 - 4:** Data Mining and Deep Learning
- **Week 4 - 6:** FPGAs and Software

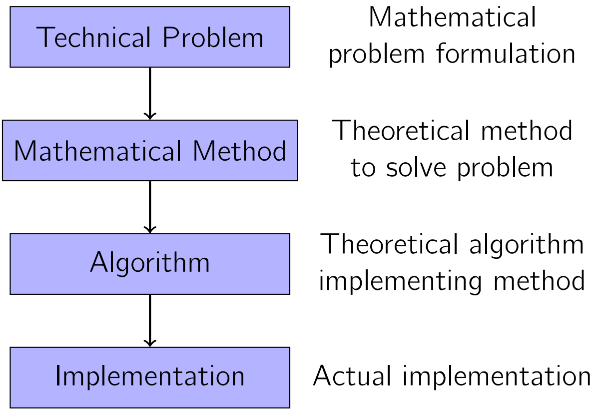
Goal: Dogs vs. Cats Kaggle competition[1]

- Image classification on FPGA with Deep Learning
- Train classifier on FPGA with Deep Learning

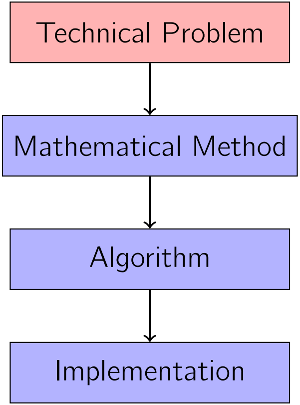
[1] https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/

The Goal: Predict dogs and cats

Overall Computer Science Approach



Overall Computer Science Approach: Example



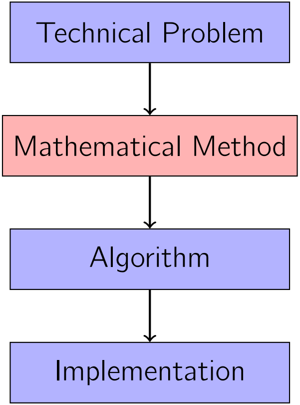
Best route from v_s to n_e in graph

Theoretical method to solve problem

Theoretical algorithm implementing method

Actual implementation

Overall Computer Science Approach: Example



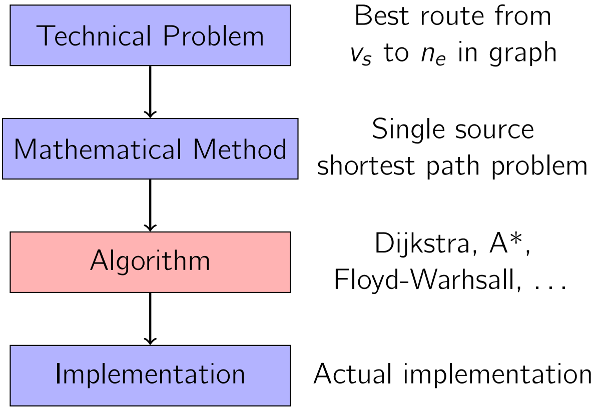
Best route from v_s to n_e in graph

Single source shortest path problem

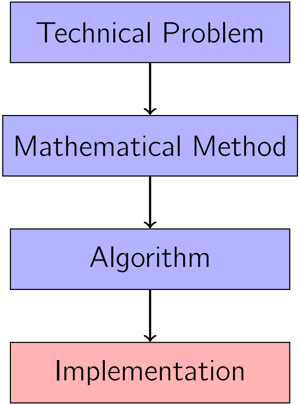
Theoretical algorithm implementing method

Actual implementation

Overall Computer Science Approach: Example



Overall Computer Science Approach: Example



Best route from
v_s to n_e in graph

Single source
shortest path problem

Dijkstra, A*,
Floyd-Warhsall, ...

`C,Java,Python,...`

Data Mining Basics

What is Data Mining?

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Data Mining Basics

"The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use."

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Example: Find all cats on the given pictures
$\rightarrow$ What is a mathematical representation of a cat?

Data Mining Basics

> “The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.”

Fact: Data Mining follows the same general approach
But: Some problems are hard to be exactly formalised and thus need some special treatment

Example: Find all cats on the given pictures
$\rightarrow$ What is a mathematical representation of a cat?

Idea: Formalise given problem by positive and negative examples
$\rightarrow$ That is our data

Data Mining Basics

Problem 1: Data needs to be gathered and pre-processed
→ crawling the web for images with tag "cat"

Data Mining Basics

Problem 1: Data needs to be gathered and pre-processed

$\rightarrow$ crawling the web for images with tag "cat"

Problem 2: Totally unclear what knowledge our data might contain

$\rightarrow$ cats and dogs can be on the same picture

$\Rightarrow$ We have to "mine" data and knowledge from it

Data Mining Basics

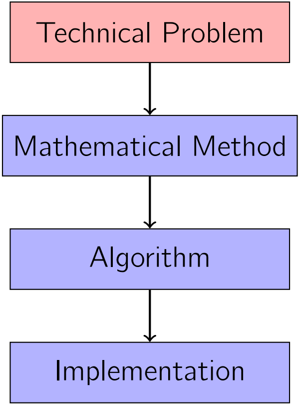
Problem 1: Data needs to be gathered and pre-processed
→ crawling the web for images with tag "cat"
Problem 2: Totally unclear what knowledge our data might contain
→ cats and dogs can be on the same picture
⇒ We have to "mine" data and knowledge from it

Data Mining is an interdisciplinary field of:

- computer science: algorithm, theory, data structure, algorithm implementation, data warehousing, ...
- statistics: algorithm, theoretical insights, modelling, ...
- domain specifics: theoretical and practical insights, special knowledge, ...

Our focus: Mostly implementation and algorithms

Overall Computer Science Approach



Data Mining problems

Theoretical method
to solve problem

Theoretical algorithm
implementing method

Actual implementation

Data Mining: Problems

Our focus: Classification

Given:

- Set of possible classes $\mathcal{Y}$, e.g. $\mathcal{Y} = \{-1, +1\}$
- Set of labelled training examples / data
  $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N) \mid (\vec{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$
- A model $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameter $\theta \in \Theta$

Find: $\widehat{\theta}$, so that $f_{\widehat{\theta}}(\vec{x}) = \widehat{f}(\vec{x})$ that predicts class y for given $\vec{x}$

Data Mining: Problems

Our focus: Classification

Given:

- Set of possible classes $\mathcal{Y}$, e.g. $\mathcal{Y} = \{-1, +1\}$
- Set of labelled training examples / data
  $\mathcal{D} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N) \mid (\vec{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$
- A model $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameter $\theta \in \Theta$

Find: $\widehat{\theta}$, so that $f_{\widehat{\theta}}(\vec{x}) = \widehat{f}(\vec{x})$ that predicts class y for given $\vec{x}$

Note 1: If $|\mathcal{Y}| = 2$ its called binary classification
Note 2: If $\mathcal{Y} = \mathbb{R}$ its called regression
Our focus: Binary classification: $\mathcal{Y} = \{0, +1\}$ or $\mathcal{Y} = \{-1, +1\}$

Data Mining: Notation

Note: The input space can be (nearly) everything
Our focus: $d-$dimensional vectors: $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^n$

$\mathcal{D}$	Feature 1	Feature 2	...	Feature d	Label
Example 1	x_{11}	x_{12}	$\dots$	x_{1d}	y_1
Example 2	x_{21}	x_{22}	$\dots$	x_{2d}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Example N	x_{N1}	x_{N1}	$\dots$	x_{Nd}	y_N

Data Mining: Notation

Note: The input space can be (nearly) everything

Our focus: $d-$dimensional vectors: $\vec{x} \in \mathcal{X} \subseteq \mathbb{R}^n$

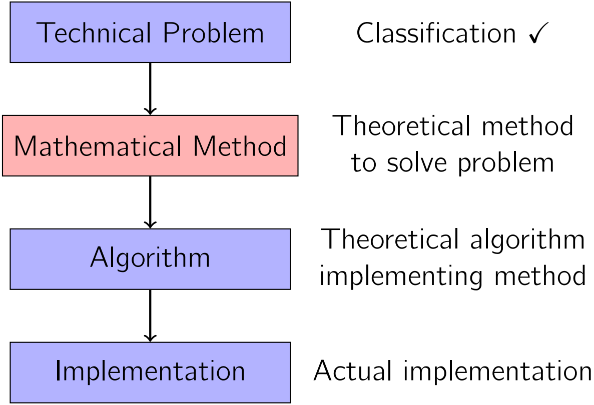
$\mathcal{D}$	Feature 1	Feature 2	...	Feature d	Label
Example 1	x_{11}	x_{12}	$\dots$	x_{1d}	y_1
Example 2	x_{21}	x_{22}	$\dots$	x_{2d}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Example N	x_{N1}	x_{N1}	$\dots$	x_{Nd}	y_N

Matrix $X \in \mathbb{R}^{d \times N}$

Vector $\vec{y} \in \mathcal{Y}^N$

then: in short $\mathcal{D} = (X, \vec{y})$

Overall Computer Science Approach



Data Mining: K nearest neighbour method

Obviously: We want a prediction method $\widehat{f}(\vec{x})$
Observation: Examples $\vec{x}_i$ and $\vec{x}_j$ which are similar probably have the same label $y_i = y_j$

Data Mining: K nearest neighbour method

Obviously: We want a prediction method $\widehat{f}(\vec{x})$
Observation: Examples $\vec{x}_i$ and $\vec{x}_j$ which are similar probably have the same label $y_i = y_j$

Idea: Given new and unseen observation $\vec{x}$

- use distance function $dist\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- calculate $d(\vec{x}, \vec{x}_i)$ for all $i = 1, \ldots, N$
- find k nearest neighbours of $\vec{x}$ $S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_k, y_k)\}$
- predict most common label in S

Data Mining: K nearest neighbour method

Obviously: We want a prediction method $\widehat{f}(\vec{x})$
Observation: Examples $\vec{x}_i$ and $\vec{x}_j$ which are similar probably have the same label $y_i = y_j$

Idea: Given new and unseen observation $\vec{x}$

- use distance function $dist\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$
- calculate $d(\vec{x}, \vec{x}_i)$ for all $i = 1, \ldots, N$
- find k nearest neighbours of $\vec{x}$ $S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_k, y_k)\}$
- predict most common label in S

Note: If S has equal number of positive and negative examples, take a random class

Data Mining: K-NN (Some Notes)

Note 1: K-NN has no real model θ, we just use the data directly

Data Mining: K-NN (Some Notes)

Note 1: K-NN has no real model θ, we just use the data directly

K-NN has two parameters

- *dist* Models the distance of neighbours. This must fit the data given! Usually euclidean norm is a good start:

$$dist(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_i - \vec{x}_j)^T \cdot (\vec{x}_i - \vec{x}_j)}$$

- K Models the number of neighbours we want to look at.

Data Mining: K-NN (Some Notes)

Note 1: K-NN has no real model θ, we just use the data directly

K-NN has two parameters

- *dist* Models the distance of neighbours. This must fit the data given! Usually euclidean norm is a good start:

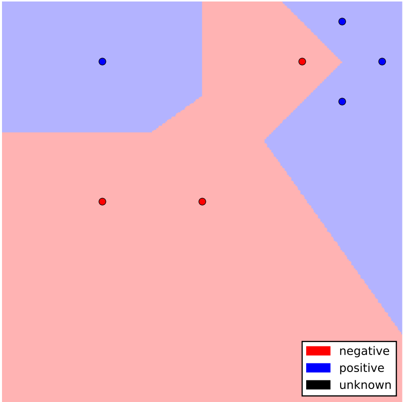
$$dist(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_i - \vec{x}_j)^T \cdot (\vec{x}_i - \vec{x}_j)}$$

- K Models the number of neighbours we want to look at.

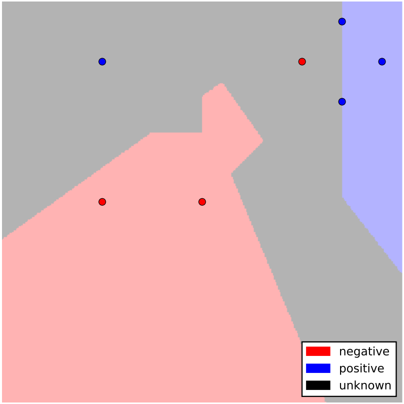
Note 2: K-NN can be used for regression as well. Just average the labels in S :

$$\widehat{f}(\vec{x}) = \frac{1}{k} \sum_{y \in S} y$$

Data Mining: K-NN Examples

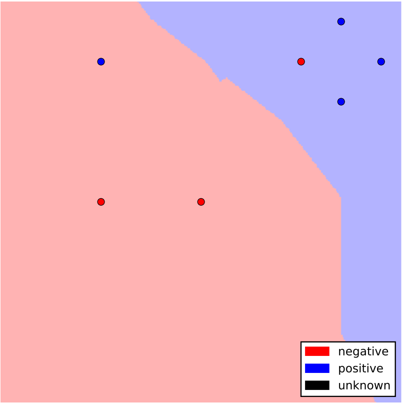


$k = 1$

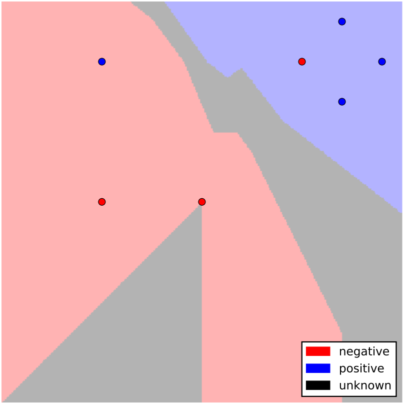


$k = 2$

Data Mining: K-NN More examples

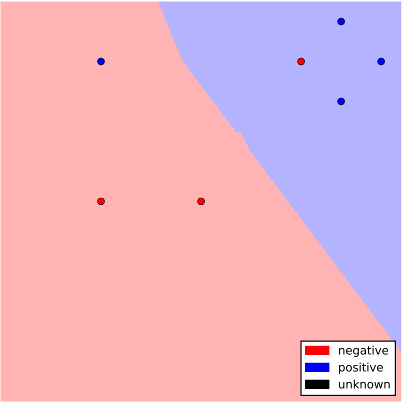


$k = 3$

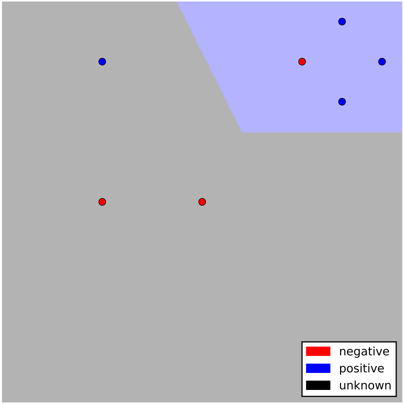


$k = 4$

Data Mining: K-NN Even more examples

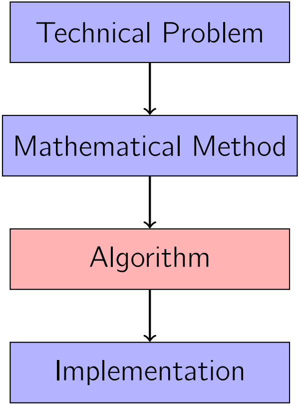


$k = 5$



$k = 6$

Overall Computer Science Approach



Classification ✓

K-NN ✓

Theoretical algorithm implementing method

Actual implementation

Data Mining: Naive K-NN algorithm

Let: $\vec{x}^*$ be new unobserved data to be classified

$S = \emptyset$
for $i = 1, \ldots, K$ **do**
    for $\vec{x} \in X$ **do**
        if $d(\vec{x}^*, \vec{x}) < min$ and $\vec{x} \notin S$ **then**
            $min = d(\vec{x}^*, \vec{x})$
            $\vec{x}_{min} = \vec{x}$
        end if
        $S = S \cup \{\vec{x}_{min}\}$
    end for
end for

Data Mining: Naive K-NN algorithm

Let: $\vec{x}^*$ be new unobserved data to be classified

```
S = ∅
for i = 1, ..., K do
    for x⃗ ∈ X do
        if d(x⃗*, x⃗) < min and x⃗ ∉ S then
            min = d(x⃗*, x⃗)
            x⃗_min = x⃗
        end if
        S = S ∪ {x⃗_min}
    end for
end for
```

Computation in $O(d)$ — refers to $d(\vec{x}^*, \vec{x})$

Lookup in $O(K)$ — refers to $\vec{x} \notin S$

Data Mining: Naive K-NN algorithm

Let: $\vec{x}^*$ be new unobserved data to be classified

$S = \emptyset$
for $i = 1, \ldots, K$ **do**
  for $\vec{x} \in X$ **do**
    if $d(\vec{x}^*, \vec{x}) < min$ and $\vec{x} \notin S$ **then**
      $min = d(\vec{x}^*, \vec{x})$
      $\vec{x}_{min} = \vec{x}$
    end if
    $S = S \cup \{\vec{x}_{min}\}$
  end for
end for

Computation in $O(d)$

Lookup in $O(K)$

Worst Case runtime: $O(K^2Nd)$ for every new example!

Data Mining: More intelligent K-NN algorithm (1)

We want: Extract model $\widehat{\theta}$ once, then apply it

Thus: Model extraction can be slow, but application should be fast

Data Mining: More intelligent K-NN algorithm (1)

We want: Extract model $\widehat{\theta}$ once, then apply it
Thus: Model extraction can be slow, but application should be fast

Often: $k \leq 20, d \approx 100 - 1000, N \geq 1000$
Observation 1: Our K-NN algorithm does not really compute a model. It just uses the data $\mathcal{D}$ $\rightarrow$ really fast model computation

Data Mining: More intelligent K-NN algorithm (1)

We want: Extract model $\widehat{\theta}$ once, then apply it
Thus: Model extraction can be slow, but application should be fast

Often: $k \leq 20, d \approx 100 - 1000, N \geq 1000$
Observation 1: Our K-NN algorithm does not really compute a model. It just uses the data $\mathcal{D} \rightarrow$ really fast model computation

But: Application is really slow, since we search over all examples
Observation 2: It is enough to only look at examples "near" $\vec{x}^*$

Data Mining: More intelligent K-NN algorithm (1)

We want: Extract model $\widehat{\theta}$ once, then apply it
Thus: Model extraction can be slow, but application should be fast

Often: $k \leq 20, d \approx 100 - 1000, N \geq 1000$
Observation 1: Our K-NN algorithm does not really compute a model. It just uses the data $\mathcal{D} \rightarrow$ really fast model computation

But: Application is really slow, since we search over all examples
Observation 2: It is enough to only look at examples "near" $\vec{x}^*$

Idea: Pre-process $\mathcal{D}$ ($\rightarrow$ data structures), so that fast retrival of neighbours is possible $\Rightarrow$ "Fast nearest neighbour search"
Thus: Training time increases, but queries are faster

Data Mining: More intelligent K-NN algorithm (2)

Fact: There are many algorithms realising this idea

- **Tree structures:** k-d tree, quadtree, range tree, ...
- **Locality Sensitive Hashing:** Random projection, TLSH, ...
- **Approximative Nearest Neighbour:** Best bin first, LSH, ...

Data Mining: More intelligent K-NN algorithm (2)

Fact: There are many algorithms realising this idea

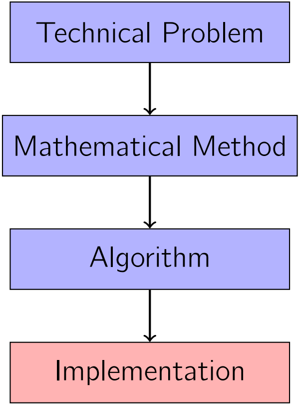
- **Tree structures:** k-d tree, quadtree, range tree, ...
- **Locality Sensitive Hashing:** Random projection, TLSH, ...
- **Approximative Nearest Neighbour:** Best bin first, LSH, ...

Usually we expect for the average case:

- **Pre-processing:** $O(Nd\log(Nd))$
- **Queries:** $O(Kd\log(N))$

Bottom line: The runtime not only depends on the method, but also the algorithm realising it

Overall Computer Science Approach



Classification ✓

K-NN ✓

Brute force
trees, hashing ✓

Actual implementation

Data Mining: Implementation of K-NN

Obviously: Implementation also influences the runtime!

Data Mining: Implementation of K-NN

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Data Mining: Implementation of K-NN

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Usually: Use language and system we know
But: Some systems / hardware is better at certain tasks
→ e.g. graphics cards are built to do matrix-vector multiplication

Data Mining: Implementation of K-NN

Obviously: Implementation also influences the runtime!

Fact: We need to take the underlying system into account

- **System:** CPU, GPU, FPGA, ...
- **Hardware:** Word length, cache sizes, vectorization, ...
- **Software:** Paging in OS, (Multi-) Threading, Swapping, ...
- **Language:** `C` vs. `Java` vs. `Haskell` ...

Usually: Use language and system we know
But: Some systems / hardware is better at certain tasks
→ e.g. graphics cards are built to do matrix-vector multiplication

Thus: Choose method and algorithm depending on system
Our focus: Mostly methods and algorithms, later implementation

Overall Computer Science Approach

Technical Problem ↓	Classification ✓
Mathematical Method ↓	K-NN ✓
Algorithm ↓	Brute force trees, hashing ✓
Implementation	System and language ✓

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
$\rightarrow$ Integer operations are fast, but less accurate than floating point

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
→ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
→ Brute force K-NN vs. indexing vs. approximated K-NN ...

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
→ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
→ Brute force K-NN vs. indexing vs. approximated K-NN ...

Bottom line: Comparing specific methods is difficult
Thus: Compare performance of **computed** model

Data Mining: Measure Model quality

Fact 1: Prediction quality also depends on the algorithm, the implementation and the data
→ Integer operations are fast, but less accurate than floating point

Fact 2: There are many different models, even more algorithms and even more implementations
→ Brute force K-NN vs. indexing vs. approximated K-NN ...

Bottom line: Comparing specific methods is difficult
Thus: Compare performance of **computed** model

Important: There is no free lunch (**Wolpert, 1996**)
→ Some methods work better on some problems, but no method works well on all problems

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

1. how well explains the model training data?
2. can we give any guarantees for new predictions?
3. how well generalises the model to new and unseen data?

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

1. how well explains the model training data?
2. can we give any guarantees for new predictions?
3. how well generalises the model to new and unseen data?

1: K-NN just saves the data
$\rightarrow$ does not explain the data at all

Data Mining: Measure Model quality (2)

Question: So, what is model quality?

1. how well explains the model training data?
2. can we give any guarantees for new predictions?
3. how well generalises the model to new and unseen data?

1: K-NN just saves the data
→ does not explain the data at all

2: K-NN assumes similarity depending on the distance function
→ no guarantees at all, especially if distance function does not fit

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
$\rightarrow$ 2 possible wrong predictions and 2 possible correct predictions

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
→ 2 possible wrong predictions and 2 possible correct predictions
Visualization: Confusion matrix

	Predicted value	
True value	True positive (TP)	False negative (FN)
	False positive (FP)	True negative (TN)

Data Mining: Measure Model quality (3)

Fact: In binary classification we have two choices: predict 0 or 1
$\rightarrow$ 2 possible wrong predictions and 2 possible correct predictions
Visualization: Confusion matrix

	Predicted value	
True value	True positive (TP)	False negative (FN)
	False positive (FP)	True negative (TN)

Accuracy: $Acc = \frac{TP+TN}{N}$

Big Remark: The accuracy only tells us something about the data $\mathcal{D}$ we know! There are no guarantees for new data

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we use $k = 1$, then $Acc = 1$ (perfect!)

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we use $k = 1$, then $Acc = 1$ (perfect!)

Question: Is that what we want?
Clear: This is just memorizing the training data, no real learning!
Question: How well deals our model with new, yet unseen data?

Data Mining: Measure Model quality (4)

Obviously: The best model has $Acc = 1$, the worst has $Acc = 0$
Observation: If we use $k = 1$, then $Acc = 1$ (perfect!)

Question: Is that what we want?
Clear: This is just memorizing the training data, no real learning!
Question: How well deals our model with new, yet unseen data?

Idea: Split data into training $\mathcal{D}_{Train}$ and test data $\mathcal{D}_{Test}$
Then: $\mathcal{D}_{Test}$ is new to the model $f_{\widehat{\theta}}$
Question: How to split $\mathcal{D}$?

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for “bad” splits.
$\rightarrow$ Model quality might be over- or under-estimated

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for "bad" splits.
$\rightarrow$ Model quality might be over- or under-estimated

2) Leave-One-Out: Use every example once for testing and train model on the remaining data. Average results.
$\rightarrow$ N models are computed, but insensitive for "bad" splits.
$\rightarrow$ Usually impractical

Data Mining: Measure Model quality (5)

1) Test/Train: Split $\mathcal{D}$ by size, e.g. 80% training and 20% test data
$\rightarrow$ Fast and easy to compute, but sensitive for "bad" splits.
$\rightarrow$ Model quality might be over- or under-estimated

2) Leave-One-Out: Use every example once for testing and train model on the remaining data. Average results.
$\rightarrow$ N models are computed, but insensitive for "bad" splits.
$\rightarrow$ Usually impractical

3) K-fold cross validation: Split data into k buckets. Use every bucket once for testing and train model on the rest. Average results.
$\rightarrow$ Insensitive for "bad" splits and practical. Usually $k = 10$.

Summary

Important concepts:

- **Classification** is one data mining task
- **Training data** is used to define and solve the task
- **A Method** is a general approach / idea to solve a task
- **A algorithm** is a way to realise a method
- **A model** forms the extracted knowledge from data
- **Accuracy** measures the model quality given the data

Summary

Important concepts:

- **Classification** is one data mining task
- **Training data** is used to define and solve the task
- **A Method** is a general approach / idea to solve a task
- **A algorithm** is a way to realise a method
- **A model** forms the extracted knowledge from data
- **Accuracy** measures the model quality given the data

Note: Runtime and model quality depend on method, algorithm and implementation

So far: K-NN is one method with many different algorithms and implementations to solve classification problems

Some administration stuff

Requirements to pass this course:

- Implement your own neural network for the FPGA
- Apply it to the data of the kaggle competition
- Give a small presentation / review about your approach

Thus: After the lecture phase you are free to do what you want until the end of the semester → you work in self-organizing groups

Question: When will we meet again for lectures?

Homework: I give some simple homeworks to get you started more easily → We will use the MNIST dataset for that

- 32×32 pixel grayscaled images of numbers $0 - 9$ (10 labels)
- already pre-processed in CSV format
- test/train split plus a smaller sample for development

Homework

Homework until next meeting

- Implement a simple CSV-Reader
  - First column contains the label $(0 - 9)$
  - Remaining 784 columns contain grayscale value $(0 - 255)$
- Implement accuracy computation for Test/Train split
  - We discussed the binary confusion matrix (4 entries)
  - Here 10 classes: Only diagonal of the confusion matrix needed for the accuracy → just count correct classifications and divide it by the total number of test examples
- Implement K-NN with distance function of your choice
  - Euclidean distance is a good start

Note 1: We will later use `C`, so please use `C` or a `C`-like language

Note 2: Use the smaller split for development and the complete data set for testing → What's your accuracy?