Learning from Label Proportions by Optimizing Cluster Model Selection

Marco Stolpe[1] and Katharina Morik[1]

Technical University of Dortmund, Artificial Intelligence Group
Baroper Strasse 301, 44227 Dortmund, Germany
marco.stolpe@tu-dortmund.de and katharina.morik@tu-dortmund.de
http://www-ai.cs.tu-dortmund.de/

Abstract. In a supervised learning scenario, we learn a mapping from input to output values, based on labeled examples. Can we learn such a mapping also from groups of unlabeled observations, only knowing, for each group, the proportion of observations with a particular label? Solutions have real world applications. Here, we consider groups of steel sticks as samples in quality control. Since the steel sticks cannot be marked individually, for each group of sticks it is only known how many sticks of high (low) quality it contains. We want to predict the achieved quality for each stick **before** it reaches the final production station and quality control, in order to save resources. We define the problem of learning from label proportions and present a solution based on clustering. Our method empirically shows a better prediction performance than recent approaches based on probabilistic SVMs, Kernel k-Means or conditional exponential models.

1 Introduction

Consider a steel factory where charges of steel sticks are processed sequentially at several production stations. The quality of sticks is assessed at the end of the process. For each stick though, we are given sensor measurements and other parameters during its being processed. Based on this information, we want to predict the quality of individual sticks as early as possible, *before* they reach the final production station and quality control. This saves resources, because sticks that can no longer reach the desired quality can be locked out. The steel sticks cannot be marked and tracked. Therefore, the available quality information is not related to single sticks, but charges of sticks. For each charge, we know how many sticks had a certain type of error (quality). We want to learn a prediction function from the sensor measurements of the process and the error type counts of charges. The learned model is used to predict the error type for individual sticks at intermediate production stations.

We generalize this learning problem. It deviates from that of supervised learning, where we learn from individually labeled training examples. It is different from semi-supervised learning [5], where we are given at least some labeled examples. Since we have *some* label information, it is not strictly unsupervised learning. Multiple instance learning [23] is a special case, because the bags of examples are either labeled positive or negative, where we have proportions of labels for each charge.

In this paper, we contribute a clustering approach for the problem which has the following properties:

– It empirically shows good prediction performance.
– Learning has linear running time in the number of observations.
– Its prediction models are fast to apply.
– It can handle the case of multiple classes.
– It can handle additional labeled observations, if they exist.
– It can weight the relevance of features.
– It is independent of a certain clustering method.

To the best of our knowledge, no other method exists yet which shares *all* of these properties.

The paper is structured as follows. Section 2 formally defines the learning task and accompanying error measures. We analyze best and worst case from a Bayesian perspective. Section 3 presents a new method for learning from label proportions, LLP. In Sect. 4, we compare the prediction performance and runtime of LLP with other existing methods. In Sect. 5, we shortly discuss related work and conclude.

2 Learning from Label Proportions

In the following, we will first define the task of learning from label proportions. Then, we introduce accompanying measures for evaluating the performance of learners and discuss the problem of model selection. In the last subsection, we explore best and worse case by analyzing the problem from a Bayesian perspective.

2.1 The Learning Task

The task of learning from label proportions can be defined as follows.

Definition 1 (Learning from label proportions). *Let X be an instance space composed of a set of features $X_1 \times \ldots \times X_m$ and $Y = \{y_1, \ldots, y_l\}$ be a set of categorical class labels. Let $P(X, Y)$ be an unknown joint distribution of observations and their class label. Given is a sample of unlabeled observations $U = \{x_1, \ldots, x_n\} \subset X$, drawn i.i.d. from P, partitioned into h disjunct groups $G_1, \ldots, G_h$. Further given are the proportions $\pi_{ij} \in [0, 1]$ of label y_j in group G_i, for each group and label. Based on this information, we seek a function (model) $g : X \to Y$ that predicts $y \in Y$ for observations $x \in X$ drawn i.i.d. from P, such that the expected error*

$$Err_P = E[L(Y, g(X))] \tag{1}$$

Labeled examples (unknown)

$G_1 = \{(x_1, 1), (x_3, 1), (x_7, 0)\}$
$G_2 = \{(x_2, 0), (x_4, 0), (x_5, 1), (x_6, 1)\}$
$G_3 = \{(x_8, 0), (x_9, 0)\}$

Sample U (known)

$G_1 = \{x_1, x_3, x_7\}$ $\quad n = 9$
$G_2 = \{x_2, x_4, x_5, x_6\}$ $\quad h = 3$
$G_3 = \{x_8, x_9\}$ $\quad l = 2$

Label proportions (known)

$Y = \{0, 1\}$

$$\Pi = \begin{array}{cc} y_1 & y_2 \\ \end{array}$$
$$\Pi = \begin{pmatrix} 0.33 & 0.67 \\ 0.50 & 0.50 \\ 1.00 & 0.00 \end{pmatrix} \begin{array}{l} |G_1| = 3 \\ |G_2| = 4 \\ |G_3| = 2 \end{array}$$

$\eta \quad 0.56 \quad 0.44$

Fig. 1: Example for a given label proportion matrix Π

for a loss function $L(Y, g)$ is minimized. The loss penalizes the deviation between the known and predicted label value for an individual observation x.

The main difference to a supervised learning scenario is that the labels of individual observations are unknown or hidden, i.e., there is *no* set of labeled training instances $(x_i, y_i) \in X \times Y$.

The label proportions π_{ij} can be conveniently written as a $h \times l$ matrix $\Pi = (\pi_{ij})$, where the values in a row $\Pi_{i,\cdot} = (\pi_{i1}, \ldots, \pi_{il})$ sum up to one. (see Fig. 1). The proportion of label y_j over sample U can be calculated from Π:

$$\eta(\Pi, y_j) = \frac{1}{n} \sum_{i=1}^{h} |G_i| \cdot \pi_{ij} \tag{2}$$

By multiplication of π_{ij} with its respective group size $|G_i|$, one gets the frequency counts μ_{ij} of observations with label $y_j \in Y$ in group G_i.

2.2 Training and Test Error

In a supervised learning scenario, a learner can adjust its current hypothesis based on the average loss on the training set. In contrast, when learning from label proportions, one can only measure how well the given proportions are matched. Applying the learned model $g(X)$ to all $x_i \in U$, the resulting label proportions can be calculated, i.e., in each group one counts the number of observations x_i with $g(x_i) = y_j$ for each label $y_j \in Y$ and divides the counts by the size of their respective group. This leads to a new matrix Γ_g, containing the model-based label proportions:

$$\Gamma_g = (\gamma_{ij}^g), \quad \gamma_{ij}^g = \frac{1}{|G_i|} \sum_{x \in G_i} I(g(x), y_j), \quad I = \begin{cases} 1 : g(x) = y_j \\ 0 : g(x) \neq y_j \end{cases} \tag{3}$$

Similarly to defining a loss function for individual observations, it is now possible to define a loss function for individual matrix entries, for example by the squared error $(\pi_{ij} - \gamma^g_{ij})^2$. The total deviance between Π and Γ_g can then be defined as the average squared error over all matrix entries:

$$\mathrm{Err}_{MSE}(\Pi, \Gamma_g) = \frac{1}{hl} \sum_{i=1}^{h} \sum_{j=1}^{l} (\pi_{ij} - \gamma^g_{ij})^2 \tag{4}$$

Calculating Err_{MSE} for a function $g(X)$ on sample U might be seen as an analogon to the training error in supervised learning. However, the loss in Err_{MSE} uses aggregated label information, although by Definition 1, we really need to minimize the loss over individual observations. The mismatch between the two measures can lead to problems, because many labelings of U can minimize Err_{MSE}. For example, when randomly sampling μ_{ij} many observations from G_i and assigning them label y_j, the model-based label proportions will always match exactly the given proportions. Hence, labelings that minimize Err_{MSE} don't necessarily minimize the average loss over individual observations, already on sample U. Therefore, obtaining a good estimate of Err_P is difficult. In supervised learning, one may select the model which has the lowest average loss over one or several test sets. But without labels for individual observations in the test set, only knowing its label proportions, a low Err_{MSE} on the test set is no reliable indicator for a good model. As for the training error, many different labelings can lead to the same label proportions, but only a few labelings will minimize Err_P. However, if given a labeled test set, it is possible to evaluate different models as in the supervised case.

For the experiments in Sect. 4, the error between Π and Γ_g wasn't measured by Err_{MSE}, but by Err_{Π}, a combination of two different error measures:

$$\mathrm{Err}_{\Pi}(\Gamma_g) = \sqrt{\mathrm{Err}_{weight}(\Pi, \Gamma_g) \cdot \mathrm{Err}_{prior}(\Pi, \Gamma_g)} \quad \text{with} \tag{5}$$

$$\mathrm{Err}_{weight}(\Pi, \Gamma_g) = \frac{1}{hl} \sum_{i=1}^{h} \sum_{j=1}^{l} \eta(\Pi, y_j) \frac{|G_i|}{n} (\pi_{ij} - \gamma^g_{ij})^2 \quad \text{and} \tag{6}$$

$$\mathrm{Err}_{prior}(\Pi, \Gamma_g) = \frac{1}{l} \sum_{j=1}^{l} \left(\eta(\Pi, y_j) - \eta(\Gamma_g, y_j)\right)^2 \tag{7}$$

Err_{weight} weights the squared error of individual matrix entries by their relative group and class size. Err_{prior} catches situations where two hypotheses g_1 and g_2 are indistinguishable from each other, because the total error sum over all matrix entries is the same. In such cases, they may be distinguished by their column differences, as calculated by η.

If in addition to the label proportions, the labels $c_1, \ldots, c_t$ of individual observations $T = \{a_1, \ldots, a_t\} \subseteq U$ are given, error criterion (5) can be easily extended by including the average loss Err_T over these training examples:

$$\mathrm{Err}_{\Pi} = \sqrt{\mathrm{Err}_{weight} \cdot \mathrm{Err}_{prior} \cdot \mathrm{Err}_T} \quad \text{with} \quad \mathrm{Err}_T = \frac{1}{t} \sum_{i=1}^{t} L(c_i, g(a_i)) \tag{8}$$

2.3 Best and Worst Case from a Bayesian Perspective

From a Bayesian perspective, a good model can be obtained by estimating the conditional class density $P(Y|X)$. Applying Bayes theorem, one recognizes that $P(Y|X)$ may also be estimated from other unknown densities—the class-conditional density $P(X|Y)$ and the class prior density $P(Y)$:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{9}$$

$P(X)$ doesn't need to be estimated, since it can be calculated from $P(X|Y)$ and $P(Y)$. When learning from label proportions, the class prior $P(y_j)$ for label y_j can be estimated as $\eta(\Pi, y_j)$, the proportion of y_j over sample U. However, finding a good estimate for $P(X|Y)$ depends on the distribution of observations over the given groups and the form of matrix Π.

In the *best case*, each group G_i only contains observations from a single class and at least l groups contain observations from different classes. If $\pi_{ij} = 1$ appears in a row, all observations in the group must have the same label, which can be assigned to all group members. We then are in a familiar supervised learning scenario and can choose from many well-known classifiers for training.

However, without further knowledge about the distribution of observations over the groups, the best we can assume is a uniform distribution. Here, in the *worst case*, all π_{ij} are equal. Then, if we interpret π_{ij} as an estimate for the class prior $P(y_j|G_i)$ of group G_i, it equals the estimated class prior $P(y_j)$. Since each observation has the same probability of being sampled into group G_i, and we assumed all priors to be equal, we can only guess the correct label with probability $1/l$. In general, if the number of groups remains constant, $P(y_j|G_i)$ will approximate $P(y_j)$ for large sample sizes n. This can be seen if we imagine each group G_i to be an independent data set with observations sampled from the same distribution $P(X, Y)$.

For cases where all $P(y_j)$ are different, a better performance can be achieved than in the worst case. One can at least predict the majority class. The question is if one can get any better. Except for the best case, the estimation of $P(X|Y)$ is difficult, because observations with the same label are spread over all groups. The LLP algorithm, introduced in the next section, is based on the idea that observations sharing the same label might also have similar feature values.

3 Learning from Label Proportions by Clustering

The k-Nearest-Neighbour classifier predicts the majority label of k known observations closest to a given search point. It is presupposed that observations lying close together in local regions of the input space also share the same class label. If we could somehow identify these local groups of observations, which is the problem of cluster analysis, the only problem left was to assign the correct labels to the clusters.

Definition 2 (Cluster analysis). *Given a sample U of n unlabeled observations $x_1, \ldots, x_n$ and a measure $d : X \times X \to \mathbb{R}^+$ for the dissimilarity of observations, the aim of cluster analysis is to determine a set $\mathcal{C} = \{C_1, \ldots, C_k\}$ (clustering) of subsets $C_i \subset U$ (clusters), such that observations within the same cluster are more similar to each other than those in different clusters, as measured by a quality function $q : 2^{2^X} \to \mathbb{R}^+$.*

Many algorithms have been developed for solving this task. We focus on those returning disjunct clusters, like the well-known k-Means algorithm [16], which was also used for the experiments in Sect. 4. Given a clustering, it must be found out which cluster best represents which class. The problem is solved by assigning each cluster a label such that Err_Π (5) is minimized (see Sect. 3.3).

In how far similar observations share the same class label not only depends on $P(X, Y)$, but also on the chosen similarity measure. According to Hastie et al. [13], the relevance of features can have an enormous influence on the clustering results. Therefore, the similarity measure should respect weights $w_f \in [0, 1]$ for each feature, as given by a vector $\boldsymbol{w} = (w_1, \ldots, w_m)$. In unsupervised learning, such weights are usually specified by a domain expert. Here, the relevance weights can be approximated automatically (see Sect. 3.2), based on criterion Err_Π. In the next section, the accompanying optimization problem is stated. Then, the LLP algorithm for solving it is described.

3.1 Optimization Problem

Let the vector $\boldsymbol{\lambda}_\mathcal{C} = (\lambda_1, \ldots, \lambda_k)$ with $\lambda_j \in Y$ represent a labeling for a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let $m_{\boldsymbol{\lambda}_\mathcal{C}} : U \to Y$ be a mapping that returns for a given observation $x \in C_i$ the label λ_i. Given a clustering $\mathcal{C}$, we are searching for a labeling $\boldsymbol{\lambda}_\mathcal{C}$ of the clusters that minimizes the error (5) between the model-based label proportions $\Gamma_{m_{\boldsymbol{\lambda}_\mathcal{C}}}$ and the known label proportions:

$$\min_{\boldsymbol{\lambda}_\mathcal{C}} \mathrm{Err}_\Pi(\Gamma_{m_{\boldsymbol{\lambda}_\mathcal{C}}}) \tag{10}$$

Let $q_{\boldsymbol{w}}$ be a function which is able to assess the quality of a clustering based on a similarity measure that respects feature weights. We are searching for a clustering which maximizes $q_{\boldsymbol{w}}$ and whose labeling most minimizes Err_Π, for all possible weight vectors $\boldsymbol{w}$. This optimization problem can be stated as follows:

$$\min_{\boldsymbol{w}} \mathrm{Err}_\Pi(\Gamma_{m_{\boldsymbol{\lambda}^*_\mathcal{C}}}), \quad \boldsymbol{\lambda}^*_\mathcal{C} = \operatorname*{argmin}_{\boldsymbol{\lambda}_{\mathcal{C}^*}} \mathrm{Err}_\Pi(\Gamma_{m_{\boldsymbol{\lambda}_{\mathcal{C}^*}}}), \quad \mathcal{C}^* = \operatorname*{argmax}_{\mathcal{C}} q_{\boldsymbol{w}}(\mathcal{C}) \tag{11}$$

3.2 The LLP Algorithm

The LLP algorithm solves problem (11) by an evolutionary strategy. For each weight vector $\boldsymbol{w}$, the sub-optimization problem of maximizing $q_{\boldsymbol{w}}$ is solved by an inner clustering algorithm, where the particular $q_{\boldsymbol{w}}$ depends on the algorithm.

Algorithm 1 The LLP algorithm.

```
Input: Label proportion matrix $\mathit{\Pi}$, sample $U$, groups $\mathcal{G} = \{G_1, \ldots, G_h\}$,
  labels $Y = \{y_1, \ldots, y_l\}$, clustering algorithm clusterer, labeling algorithm labeler,
  parameters maxgen, psize, mutvar, crossprob, tournsize
Output: Clustering $\mathcal{C}$, labeling $\boldsymbol{\lambda}_\mathcal{C}$, weight vector $\boldsymbol{w}$
  $best\_fit := -\infty$; $generation := 0$
  Randomly initialize a population $P$ of psize normalized weight vectors
  while $generation < maxgen$ do
    for $\boldsymbol{w} \in P$ do
      $\mathcal{C} := clusterer(\ U, \boldsymbol{w}\ )$
      $(\boldsymbol{\lambda}_\mathcal{C}, \mathrm{Err}_\mathit{\Pi}) := labeler(\ \mathcal{C}, \mathcal{G}, \mathit{\Pi}, Y\ )$
      if $best\_fit < -\mathrm{Err}_\mathit{\Pi}$ then
        $best\_fit := -\mathrm{Err}_\mathit{\Pi}$;   $best\_\mathcal{C} := \mathcal{C}$;   $best\_\boldsymbol{\lambda}_\mathcal{C} := \boldsymbol{\lambda}_\mathcal{C}$;   $best\_\boldsymbol{w} := \boldsymbol{w}$
      end if
    end for
    $generation := generation + 1$
    if $generation < maxgen$ then
      $Pcopy := P$
      Gaussian mutation of all individual weights in $Pcopy$ with variance mutvar
      $Pchildren :=$ Uniform crossover on $P \cup Pcopy$ with probability crossprob
      $P :=$ Tournament selection with size tournsize on $P \cup Pcopy \cup Pchildren$
    end if
  end while
  return $best\_\mathcal{C}$, $best\_\boldsymbol{\lambda}_\mathcal{C}$, $best\_\boldsymbol{w}$
```

The only prerequisite for the clusterer is that it returns disjunct clusters and respects different feature weights. The sub-optimization problem (10) is independent from the clusterer and currently can be solved by two different labeling heuristics introduced in Sect. 3.3. Using an evolutionary strategy as a wrapper has the advantage that it is not necessary to integrate criterion $\mathrm{Err}_\mathit{\Pi}$ into the optimization problem of the inner clustering algorithm. For example, we already have run LLP successfully with Kernel k-Means [10], DBSCAN [12] and PROCLUS [1], without modification. Moreover, LLP can be used with different error measures, for instance with criterion (8) that can respect individually labeled examples.

LLP (see Alg. 1) takes a clustering algorithm *clusterer* and a labeling algorithm *labeler* as parameters, in addition to $\mathit{\Pi}$, U, $G_1, \ldots, G_h$ and Y, which are related to the label proportions learning task. LLP then approximates the optimal weight vector $\boldsymbol{w}$ and returns $\boldsymbol{w}$, the related clustering $\mathcal{C}$ and labels $\boldsymbol{\lambda}_\mathcal{C}$ for the clusters.

The evolutionary strategy starts with a random population P of normalized weight vectors, $\boldsymbol{w}_i \in [0,1]$. For each individual in P, the clustering algorithm *clusterer* is called. The clusters are labeled according to the given labeling algorithm *labeler* and the fitness is evaluated by criterion $\mathrm{Err}_\mathit{\Pi}$. If the fitness is higher than the best fitness seen so far, the newly found clustering, labeling and weight vector are memorized as the new best ones. In each generation, the

Algorithm 2 The greedy labeling algorithm.

```
Input: Clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, groups $\mathcal{G} = \{G_1, \ldots, G_h\}$,
  label proportion matrix $\mathit{\Pi}$, labels $Y = \{y_1, \ldots, y_l\}$
Output: Labeling $\boldsymbol{\lambda}_{\mathcal{C}} = (\lambda_1, \ldots, \lambda_k)$
  Initialize components of $\boldsymbol{\lambda}_{\mathcal{C}}$ with $y_1$;
  for $i := 1$ to $k$ do
    lowest_error := 0; best_label := $y_1$;
    for $j := 1$ to $l$ do
      $\boldsymbol{\lambda}_{\mathcal{C}}[i] := y_j$;
      $\mathit{\Gamma}_m$ := count_labels( $\mathcal{G}$, $\mathcal{C}$, $\boldsymbol{\lambda}_{\mathcal{C}}$ );
      current_error := $\mathrm{Err}_{\mathit{\Pi}}(\mathit{\Gamma}_m)$;
      if current_error < lowest_error then
        lowest_error := current_error;
        best_label := $y_j$;
      end if
    end for
    $\boldsymbol{\lambda}_{\mathcal{C}}[i]$ := best_label;
  end for
  return $\boldsymbol{\lambda}_{\mathcal{C}}$
```

weight values in a copy of P are mutated by a Gaussian distribution and, with a certain probability, exchanged with P by a crossover operator. Then, the individuals take part in a tournament and only the best ones are kept in the next generation. This process is repeated until the maximum number of generations as specified by the user is reached.

3.3 Labeling Heuristics

The following two labeling algorithms solve the sub-optimization problem (10) heuristically.

Greedy Labeling As shown in Alg.2, in the initial step, all clusters get label y_1. Then, consecutively for each cluster, we calculate $\mathit{\Gamma}_m$ for all labels and memorize the label that most reduces $\mathrm{Err}_{\mathit{\Pi}}(\mathit{\Gamma}_m)$. The strategy has runtime $k \cdot l$.

Exhaustive Labeling Since k can be restricted to a small number and $l = 2$ for a binary classification problem, trying l^k possible labelings for a clustering $\mathcal{C}$ is no problem. In our experiments (see section 4), good solutions often were found for $k \leq 6$. For each labeling, we need to calculate $\mathrm{Err}_{\mathit{\Pi}}$, which takes linear time in the number of observations n. The calculations only involve basic operations like count, addition, multiplication and division (see (5)).

3.4 Run-Time Analysis

The user-specified parameters *maxgen*, *psize* and *tournsize* are constants. They do not depend on the number of observations n and limit the number of iterations to be constant. As discussed in Sect. 3.3, the asymptotic run-time of the heuristic labeling strategies is linear in n, as k and l are constants and the evaluation of (5) takes linear time. The asymptotic run-time of LLP will otherwise depend on the used cluster algorithm. For example, if we allow for approximate solutions and limit the number of iteration steps, k-Means has linear run-time. Hence, LLP has linear run-time.

3.5 Generating a Prediction Model

The LLP algorithm returns labeled clusters of sample U. It is then possible to assign labels to individual observations $x_i \in U$ with $m_{\boldsymbol{\lambda}_C}$. To predict the labels of new observations, the clustering must be transformed into a prediction model. The way to do this depends on the used clustering algorithm. In the case of k-Means, one can simply assign new observations to their closest cluster mean and predict the corresponding cluster label. A big advantage of the cluster mean model is that it is usually very small, as $k \ll n$, and therefore fast to apply. Another option for getting a prediction model is to train a classifier like Naïve Bayes [14] or a Support Vector Machine [22] in a subsequent step, based on the now labeled observations. However, this increases the training time.

4 Experiments

We have compared the LLP algorithm to three state-of-the-art methods for learning from label proportions: The Mean Map method [19], Inverse Calibration (Invcal) [21] and AOC Kernel k-Means (AOC-KK) [6]. For a further discussion of these methods, see Sect. 5. LLP has been implemented in Java. As inner clustering algorithm, we have used Fast k-Means [11], which is a variant of k-Means utilizing the triangle inequality for faster distance calculations. As distance measure, we have used the weighted Euclidean distance. We have implemented AOC-KK using a combination of Java and Matlab. For Mean Map and Invcal, we used R scripts provided by the author of Invcal [21].

4.1 Prediction Performance Experiments

The prediction performance (accuracy) of LLP, AOC-KK, Invcal and Mean Map has been assessed on the eight UCI [3] data sets shown in Table 1. We have mapped each possible value of a nominal feature to a binary numerical feature with values 0 or 1. Numerical features were normalized to the $[0, 1]$ interval. Table 1 shows the number of features m after this preprocessing step.

In each single experiment, the accuracy has been assessed by a 10-fold cross-validation. For learning from label proportions, we have partitioned the training set of a particular fold into groups of size σ, by uniform sampling of observations. We tried several group sizes σ: 2, 4, 8, 16, 32, 64 and 128 (with the last group smaller than σ, if necessary). The label proportions were calculated and the

Table 1: UCI data sets used for the experiments.

Dataset	n	m	Dataset	n	m
CREDITA	690	42	SONAR	208	60
VOTE	435	16	DIABETES	768	8
COLIC	368	60	BREAST CANCER	286	38
IONOSPHERE	351	34	HEARTC	303	22

individual labels removed. In each fold, the accuracy of the learned prediction model has been calculated on a labeled test set.

The kernel methods Mean Map, Invcal and AOC-KK have been tested with the linear kernel, polynomial kernels of degree 2 and 3 and radial basis kernels ($\gamma = 0.01$, 0.1 and 1.0). LLP has been tested with both labeling heuristics (see Sect. 3.3), for cluster sizes $k \in [2, 12]$. As parameters for the evolutionary strategy, we used $maxgen = 10$, $psize = 25$, $mutvar = 1.0$, $crossprob = 0.3$ and $tournsize = 0.25$. By running LLP with k-Means, we get a prediction model consisting of cluster means. The same is true for AOC-KK. However, the cluster methods also assign labels to each observation in sample U, allowing for a subsequent training of other classifiers. Based on the clustering results, we have trained models for Naïve Bayes [14], kNN [2], Decision Trees [20], Random Forests [4], and the SVM [22] with linear and radial basis kernel. The model parameters have been optimized by a grid or evolutionary search.

The combination of all datasets, group sizes, classifiers, their variants and parameters results in a total of 13.216 experiments: 672 for Mean Map and Invcal, 2.688 for AOC-KK and 9.856 for LLP. For group sizes 16, 32, 64 and 128 on the datasets COLIC and SONAR, and for group size 128 on CREDITA, we conducted additional experiments with LLP for $maxgen = 5$ and $psize = 100$. In some cases, we got a better prediction accuracy. All experiments took about three weeks. They were run in parallel on up to six machines with an AMD Dual-Core or Quad-Core Opteron 2220 processor and a maximum of 4 GB main memory.

4.2 Prediction Performance Results

Figure 2 contains plots of the highest achieved accuracies for all data sets and group sizes, based on the best performing models of LLP, AOC-KK, Invcal and MeanMap, over all conducted experiments. LLP shows a higher accuracy than Invcal for many group sizes on the data sets CREDITA, VOTE, COLIC, SONAR and BREAST CANCER. On CREDITA, VOTE, IONOSPHERE, SONAR and DIABETES, the variance of accuracy between group sizes is smaller for LLP in comparison to the other methods. Mean Map performs worse than LLP and Invcal in many cases. The performance of AOC-KK varies, depending on the data set. It shows good performance on BREAST CANCER and HEARTC, but not on the others. Except

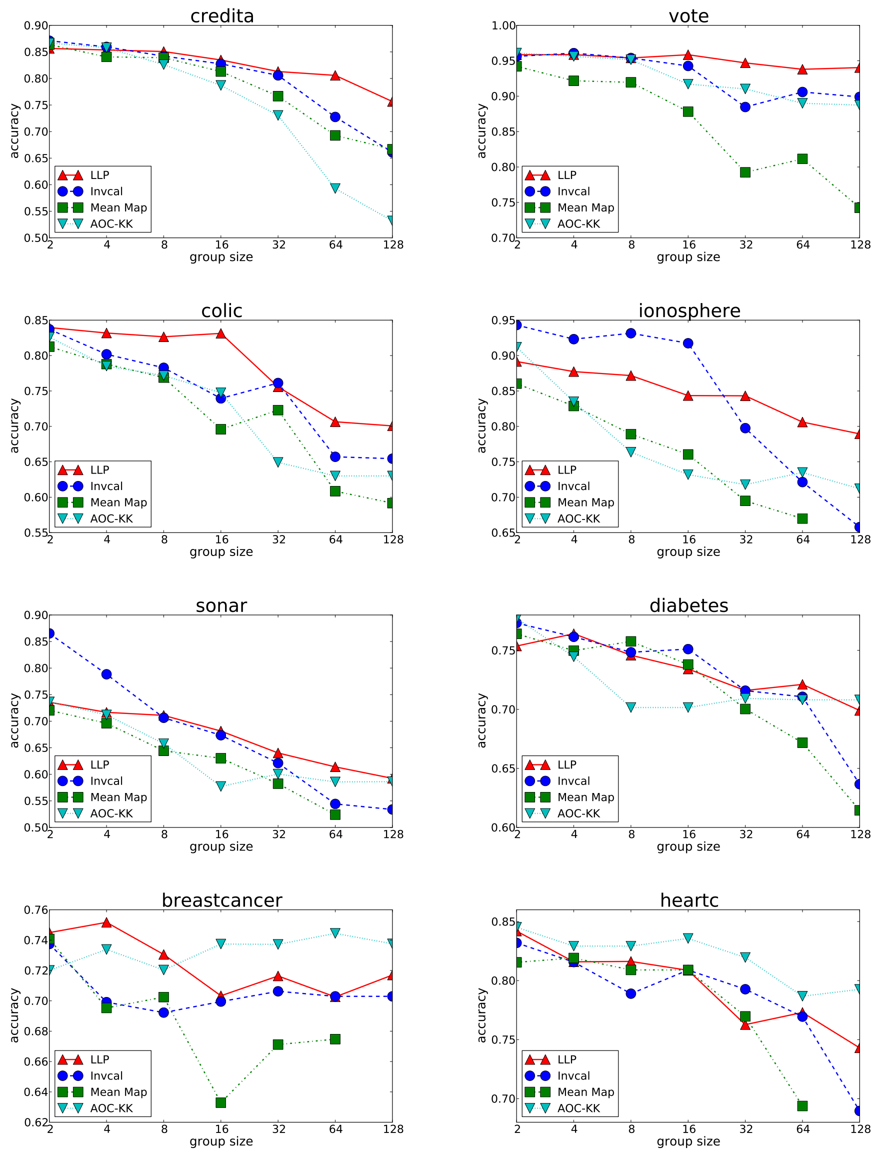


Fig. 2: Highest accuracies for all data sets and group sizes, over all 13.216 runs of LLP, AOC-KK, Invcal and MeanMap (plus the additional runs of LLP with $maxgen = 5$ and $psize = 100$). Some values for Mean Map and group size 128 are missing in the plots, due to an error in the R script.

Table 2: Average ranks of classifiers by group size, and their difference to LLP's rank, based on the best models for each data set and group size. Positive difference values indicate a better performance of LLP. Highest ranks and significant differences (higher than CD) at the 10%-level are marked in bold.

σ	2	4	8	16	32	64	128
	AVERAGE RANKS						
LLP	2.500	**1.875**	**1.500**	**1.875**	**1.625**	**1.375**	**1.375**
AOC-KK	**2.000**	2.750	3.000	2.875	2.625	2.375	2.000
Invcal	**2.000**	**1.875**	2.375	2.125	2.125	2.275	2.625
Mean Map	3.500	3.500	3.125	3.125	3.625	3.875	-
	DIFFERENCES, $CD_{<128}$=1.4317, CD_{128}=0.98						
AOC-KK	-0.500	0.875	**1.500**	1.000	1.000	1.000	0.625
Invcal	-0.500	0.000	0.875	0.250	0.500	1.000	**1.250**
Mean Map	1.000	**1.625**	**1.625**	1.250	**2.000**	**2.500**	-

for the BREAST CANCER and VOTE data set and a few other accuracy values, the overall accuracy of *all* methods decreases with a larger group size.

4.3 Statistical Significance

For the comparison of multiple classifiers over multiple data sets, Demsar [9] proposes the Friedman test, which is a non-parametric equivalent of ANOVA. We use the adjusted version, with a test statistic distributed according to the F-distribution (see [9]). The test ranks the classifiers for each data set separately. Under the null-hypothesis, the average ranks of the classifiers should be equal. In case of a critical difference, the null-hypothesis can be rejected. The test yielded significant differences for all group sizes. One can then proceed with a post-hoc test. We have decided for the two-tailed Bonferroni-Dunn test (again, see [9]), which is for comparing a single classifier (here, LLP) to all others.

Table 2 shows the average ranks of the compared classifiers and their difference to LLP's rank. Each rank was calculated based on the best performing models (including the standard classifiers), over all conducted experiments. The table also shows the critical difference (CD) values for the Bonferroni-Dunn test. The CD for $\sigma = 128$ is different, because Mean Map was not included in the comparison, due to missing values. LLP has the highest rank in six cases, for $\sigma > 2$. At the 10%-level, LLP is significantly better than AOC-KK for $\sigma = 8$, better than Invcal for $\sigma = 128$ and better than Mean Map for $\sigma = 4, 8, 32$ and 64. In all other cases, LLP performs equivalently.

The ranks in Table 3 are based on different models than those in Table 2. For LLP and AOC-KK, we have only included the best performing cluster mean models. We have compared them to the best performing models of Invcal and Mean Map, i.e. to different kernels. The cluster mean models perform significantly better than Mean Map for $\sigma = 64$ and better than Invcal for $\sigma = 128$. In

Table 3: Average ranks of classifiers by group size, and their difference to LLP's rank. Ranks are based on the best performing models of Invcal and Mean Map and the best performing cluster mean models of LLP and AOC-KK. Positive difference values indicate a better performance of LLP. Highest ranks and significant differences (higher than CD) at the 10%-level are marked in bold.

σ	2	4	8	16	32	64	128
	Average Ranks						
LLP	2.375	2.375	**2.000**	2.250	2.250	**1.750**	**1.375**
AOC-KK	3.750	3.250	3.125	2.875	2.875	2.375	2.125
Invcal	**2.125**	**1.625**	2.125	**1.750**	**1.625**	2.000	2.500
Mean Map	2.750	2.750	2.750	3.125	3.250	3.875	-
	Differences, $CD_{<128}$=1.4317, CD_{128}=0.98						
AOC-KK	1.375	0.875	1.125	0.625	0.625	0.625	0.750
Invcal	-1.000	-0.750	0.125	-0.500	-0.625	0.250	**1.125**
Mean Map	0.125	0.375	0.750	0.875	1.000	**2.125**	-

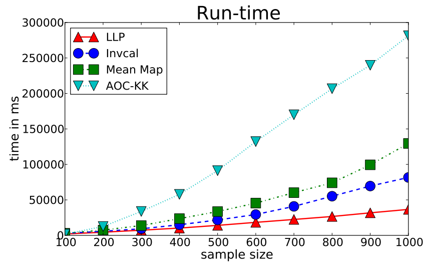
all other cases, they show an equivalent prediction performance, but are faster to train and apply, as discussed in Sects. 3.4 and 4.4. In the same way, we have separately compared the exhaustive and greedy labeling strategies to the best performing models of all other classifiers. The exhaustive strategy performed better, in the sense that it showed more significant differences to the other methods.

Concerning the performance and significance of the standard classifiers, which were trained based on the LLP and AOC-KK cluster models, Decision Trees performed significantly better than Invcal for $\sigma = 128$, better than Mean Map for $\sigma = 64$ and better than AOC-KK for $\sigma = 4$ and 32. Naive Bayes, k-NN and Random Forests had a performance similar to the cluster mean models. The linear SVM and the SVM with radial basis kernels showed no significant differences to Invcal, MeanMap or AOC-KK.

4.4 Run-Time Comparison

For an empirical run-time comparison of LLP, Invcal, Mean Map and AOC-KK, we have generated random data for a two Gaussian mixture classification problem (10.000 observations and 10 features, with values normalized to $[0, 1]$). Then, the average run-time for training and the accuracy of the classifiers over 10 folds of a cross-validation has been assessed for different samples of the data, with varying sizes (see Fig. 3). The group size for learning from label proportions has been $\sigma = 16$ for all runs. A radial basis kernel with $\gamma = 0.1$ has been used for the kernel methods. LLP has been run with the exhaustive labeling strategy and Fast k-Means ($k = 6$), with parameters $maxgen = 3$, $psize = 25$, $mutvar = 1.0$, $crossprob = 0.3$ and $tournsize = 0.25$ for the evolutionary optimization. Both LLP and AOC-KK used the cluster mean model for prediction.

LLP shows a high prediction performance for all sample sizes. Moreover, LLP has the lowest run-time. However, since the methods are implemented in



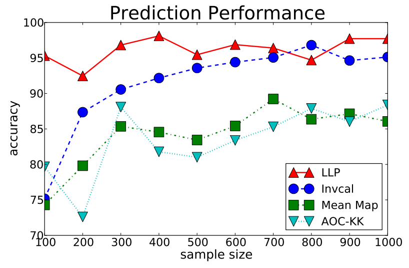


Fig. 3: Average run-time and accuracy of 10-fold cross-validations with LLP, Invcal, Mean Map and AOC-KK on several samples of random data. The data was generated for a two Gaussian mixture classification problem ($n = 10000$, $m = 10$, feature values normalized to $[0, 1]$).

different programming languages (Java, Matlab, R), one should not compare the absolute times, but the slope of the curves. The curve of LLP's run-time is a straight line, while the other curves indicate a polynomial run-time.

5 Related Work

The problem of learning from label proportions has gained attention in the machine learning community, only recently. Musicant et al. [18] formally defined the problem of learning from aggregate values for regression and classification tasks. They modified well-known methods like k-NN [2], backpropagation neural networks [17] and the linear SVM [22] to respect the given label proportions. Their experimental results focus on regression tasks, while we are mainly interested in classification.

Quadrianto et al. [19] have proposed the Mean Map method which estimates the conditional class probability $P(Y|X, \theta)$ by conditional exponential models, using a feature map $\Phi(X, Y)$ and a normalization function g:

$$P(Y|X, \theta) = \exp(\, \langle \Phi(X, Y), \theta \rangle - g(\theta|X)\,) \tag{12}$$

The parameter θ is estimated by solving a convex maximization problem for the conditional log-likelihood $\log P(Y|X, \theta)$. This depends on the unknown labels only in terms of the empirical mean μ_{XY}, which they approximate by the observation means for each group and its given label proportions. They compare Mean Map to kernel density estimation, discriminative sorting, and MCMC [15]. Mean Map outperformed the related techniques. For this reason, we have compared LLP only to Mean Map. Although LLP and Mean Map can both handle multi class problems, for easier comparison with Inverse Calibration, we have restricted our experiments to binary classification problems. During training, Mean Map needs to solve a general convex optimization problem. In contrast, LLP's worst-case training time is linear in n for the cluster mean models. As was shown in Sect. 4, these models achieved equivalent accuracy. Moreover, over all trained models, LLP's accuracy has been significantly higher than Mean Map's for several group sizes.

Rueping [21] proposes the inverse calibration method. The author converts the regression SVM (SVR) into a probabilistic classifier by applying a scaling function σ to the outputs $y = f(x)$, such that $\sigma(y)$ is a good estimate for $p = P(y = 1|x)$. Since no individual estimates p for each observation x are given, it is only required that f predicts $y = \sigma^{-1}(p)$ well on average. This is equivalent to approximating the given label proportions well. The constraints are integrated as auxiliary conditions into the standard SVR optimization problem. LLP outperformed the inverse calibration for $\sigma = 128$, also with the cluster mean models. It achieved equivalent results on smaller group sizes, but in shorter time.

For a semi-supervised learning case, Dara et al. [7] cluster the data first with SOMs and then label the clusters. However, they have labeled observations, which we do not. Demiriz et al. [8] adapt the k-Means optimization problem to respect labeled data. Again, this is a semi-supervised setting, with labeled observations. The idea is similar though to the AOC Kernel k-Means algorithm by Chen et al. [6], who integrate the loss function (4) into the optimization problem of Kernel k-Means clustering [10]. In comparison to AOC-KK, LLP has achieved significantly better accuracy for $\sigma = 8$ over all conducted experiments. For $\sigma > 2$, LLP had a higher average rank than AOC-KK. LLP needs only linear training time, while in contrast, AOC-KK solves a quadratic optimization problem in each iteration step of Kernel k-Means.

6 Conclusions and Future Work

We have presented a new approach for learning from label proportions, the LLP algorithm. With k-Means as the clustering algorithm, LLP has only linear worst-case training time and its cluster mean models are small and fast to apply. In comparison to state-of-the-art methods, which need more training time, the cluster mean models have shown significantly better or equivalent prediction accuracy. By training other classifiers on the labeled clusters, the highest achieved accuracy of LLP was significantly different in even more cases, and LLP had the highest average rank for all $\sigma > 2$. In the future, we want to evaluate LLP's performance on data from the steel factory, as mentioned in the introduction. Moreover, it would be interesting to assess LLP's prediction performance with multi class problems and additional labeled observations. Another direction is to use different clustering algorithms with LLP and compare their performance.

Acknowledgements This work has been supported by the DFG, Collaborative Research Center SFB876, project B3.

References

1. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering. In: Proc. of the Int. Conf. on Management of Data. pp. 61–72. SIGMOD '99, ACM, New York, NY, USA (1999)

2. Aha, D.: Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. Int. J. of Man-Machine Studies 36(2), 267–287 (1992)
3. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
4. Breimann, L.: Random forests. Machine Learning 45, 5–32 (2001)
5. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge, MA (2006)
6. Chen, S., Liu, B., Qian, M., Zhang, C.: Kernel k-Means based framework for aggregate outputs classification. In: Proc. of the Int. Conf. on Data Mining Workshops (ICDMW). pp. 356–361 (2009)
7. Dara, R., Kremer, S., Stacey, D.: Clustering unlabeled data with SOMs improves classification of labeled real-world data. In: Proc. of the 2002 Int. Joint Conf. on Neural Networks (IJCNN). vol. 3, pp. 2237–2242 (2002)
8. Demiriz, A., Bennett, K., Bennett, K.P., Embrechts, M.J.: Semi-supervised clustering using genetic algorithms. In: Proc. of Artif. Neural Netw. in Eng. (ANNIE). pp. 809–814. ASME Press (1999)
9. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
10. Dhillon, I., Guan, Y., Kulis, B.: Kernel k-Means: spectral clustering and normalized cuts. In: Proc. of the 10th Int. Conf. on Knowl. Discov. and Data Mining. pp. 551–556. SIGKDD '04, ACM, New York, NY, USA (2004)
11. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proc. of the 20th Int. Conf. on Machine Learning (ICML) (2003)
12. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int. Conf. on Knowl. Discov. and Data Mining. pp. 226–231. AAAI Press (1996)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Statistics, Springer, 2nd edn. (2009)
14. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. of the 11th Conf. on Uncertainty in Artif. Int. pp. 338–345. Morgan Kaufmann, San Francisco (1995)
15. Kueck, H., de Freitas, N.: Learning about individuals from group statistics. In: Uncertainty in Artif. Int. (UAI). pp. 332–339. AUAI Press, Arlington, Virginia (2005)
16. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Symp. Math. stat. & prob. pp. 281–297 (1967)
17. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
18. Musicant, D.R., Christensen, J.M., Olson, J.F.: Supervised learning by training on aggregate outputs. In: Proc. of the 7th Int. Conf. on Data Mining (ICDM). pp. 252–261. IEEE Computer Society, Washington, DC, USA (2007)
19. Quadrianto, N., Smola, A.J., Caetano, T.S., Le, Q.V.: Estimating labels from label proportions. J. Mach. Learn. Res. 10, 2349–2374 (December 2009)
20. Quinlan, J.R.: Induction of decision trees. Machine Learning 1(1), 81–106 (1986)
21. Rüping, S.: SVM classifier estimation from group probabilities. In: Proc. of the 27th Int. Conf. on Machine Learning (ICML) (2010)
22. Vapnik, V.: The nature of statistical learning theory. Springer, New York, 2nd edn. (1999)
23. Witten, I.H., Eibe, F., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. Data Management Systems, Elsevier, Inc., Burlington, MA, 3rd edn. (2011)