Multi-annotator Probabilistic Active Learning

Marek Herde, Daniel Kottke, Denis Huseljic, Bernhard Sick
{marek.herde | daniel.kottke | dhuseljic | bsick}@uni-kassel.de
Intelligent Embedded Systems Group, University of Kassel, Germany

Motivation

Situation: Training a classifier requires annotations, i.e., class labels, for instances. The corresponding annotation process is often costly due to its execution through human annotators.

Active learning (AL): AL aims at reducing these annotation costs. Therefor, an AL strategy selects instances from which the classifier is expected to learn the most.

Problem: Many AL strategies assume a single omniscient annotator who provides only correct annotations. This assumption conflicts with the error-proneness of human annotators.

Task: Take multiple independent, error-prone, but benevolent annotators into account. Therefore, an AL strategy must not only select instances but annotators, too.

Contributions:

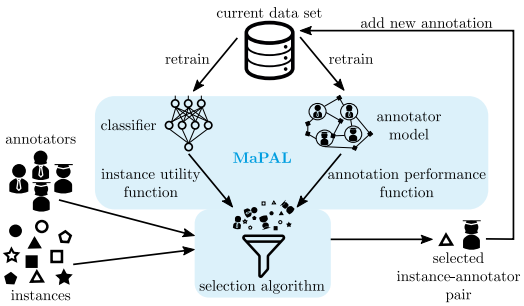
- We propose the AL strategy *multi-annotator probabilistic active learning* **(MaPAL)**. It estimates the annotation performance of error-prone annotators as a function of instances. Based on these annotation performance estimates, MaPAL jointly selects an instance-annotator pair maximizing the classifier's expected performance.
- Comparisons of our AL strategy MaPAL to five related AL strategies in an experimental evaluation over various data sets show MaPAL's superior and robust performance.

Structure of MaPAL

MaPAL's objective: Specification of the optimal data set, such that the classifier's misclassification risk is minimal given a fixed budget of annotation acquisitions. Since it is hardly possible to specify the annotation process for obtaining the optimal data set in advance, MaPAL aims at approximating the optimal solution through a greedy approach.

Cycle iteration: The data set is updated iteratively by executing a cycle, as shown by the plot below. MaPAL selects one instance-annotator pair per cycle. As a result, the selected annotator provides a class label for the selected instance, and the current data set is updated accordingly. To perform this selection, MaPAL consists of the following three components:

- The **instance utility function** quantifies the benefit of annotating an instance for training a classifier. MaPAL builds upon the *probabilistic active learning* (PAL) framework, which estimates the classifier's performance gain when annotating an instance.
- The **annotation performance function** assesses the quality of the available annotators. Therefor, MaPAL uses the proposed *Beta annotator model* (BAM), which estimates an annotator's probability of providing the correct class label for an instance.
- The **selection algorithm** relies on the instance utility and annotation performance function to select an instance-annotator pair maximizing the classifier's expected performance.

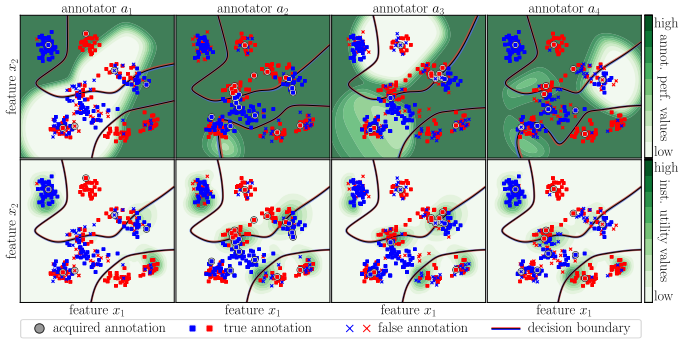


Visualization of MaPAL

The eight plots on the right visualize MaPAL's **annotation acquisition behavior** after 50 actively acquired annotations: All of them show the same two-dimensional toy data set with instances of two classes (blue vs. red). There are four simulated annotators with instance-dependent annotation performances (i.e., the probability of providing a correct annotation dependent on the features of an instance).

The four upper plots show the **annotation performance values** estimated by the BAM. The black lines represent the classifiers' decision boundaries used to assess the quality of the corresponding annotator. We notice low annotation performance estimates in regions where an annotator provided false annotations indicated by encircled crosses.

The four lower plots show the **instance utility values** as a function of each annotator. The black lines represent the decision boundary of the classifier using the current data set. We observe high utility values in dense regions near the classifier's decision boundary, provided that the annotation performance values of the corresponding annotators are high. This way, MaPAL queries annotators in regions with sufficient annotation performance estimates.



Data Sets

We conducted experiments on 29 data sets. Four of these data sets were annotated by **real-world** annotators. For each of the remaining 25 data sets, we employed three techniques to simulate annotators with different assumptions regarding their annotation performances, i.e., **uniform**, **class-dependent**, and **instance-dependent** performances. The number of annotators varied between four to six across all data sets.

In our **repository** `https://github.com/mherde/mapal`, more details on the simulation procedures are given. Moreover, we list all used data sets, including their references, and provide specific characteristics such as the number of instances, features, instances per class, and the annotators' actual mean annotation accuracies.

We compared MaPAL to five related AL strategies, namely **IEThresh**, **IEAdjCost**, **Proactive**, **CEAL**, and **ALIO**. As a baseline AL strategy, we implemented **Random** randomly selecting instance-annotator pairs.

Experimental Setup

Each experiment, i.e., testing an AL strategy on a data set, was run **100 times**. In each run, we randomly split the data set into a training set consisting of 60% of the instances and a test set containing the remaining 40%. The training set did not include any annotations at the start of each run. We stopped the annotation process if a strategy acquired either 40% of the available annotations or reached the limit of 1000 annotation acquisitions.

For classification, we employed a **kernel-based classifier** adjusted to the multi-annotator setting to learn despite partially incorrect annotations.

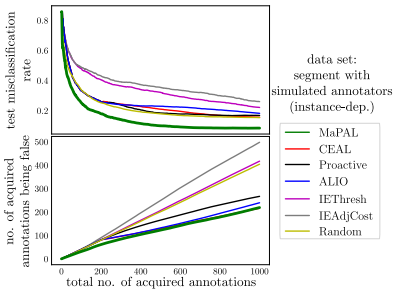
For two data sets, namely compendium and mozilla, we employed the **cosine similarity** kernel because both data sets deal with text classification. For the remaining data sets, the classifier used the **radial basis function** as a kernel.

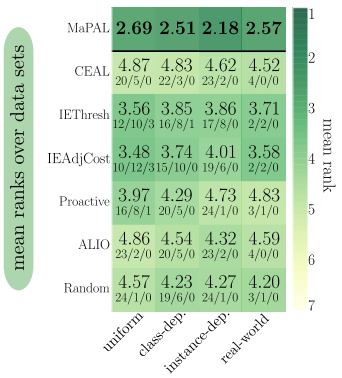
Results

Learning curves represent a common way to compare AL strategies regarding their performances. The right plots show the learning curves for the data set *segment*.

The upper plot of learning curves indicates the **test misclassification rate** after each annotation acquisition. Comparing MaPAL's learning curve to the other strategies, we observe the superiority of MaPAL, which converges fast to a low misclassification rate.

The superior performance of MaPAL is also shown by the lower plot of learning curves counting the **number of false annotations** after each annotation acquisition. The performance estimates of the BAM supported MaPAL in acquiring less false annotations.





We computed the **area under the learning curve (AULC)** of the test misclassification rate for each run of an experiment. As a result, there were 100 AULC values for each AL strategy per data set. We ranked these values between the same runs of the different AL strategies and averaged the ranks for each AL strategy.

We visualize the resulting **mean ranks** taken over all data sets of an annotator group (i.e., uniform, class-dep., instance-dep., and real-world) per AL strategy in the left plot: dark green means good rank, whereas light green means bad rank.

For example, MaPAL has an average rank of 2.18 over the 25 data sets with simulated annotators having instance-dependent performances. The triples below the ranks of MaPAL's competitors count the number of MaPAL's **wins/ties/losses** compared to the corresponding competitor, e.g., MaPAL outperformed CEAL on four of four data sets with real-world annotators.