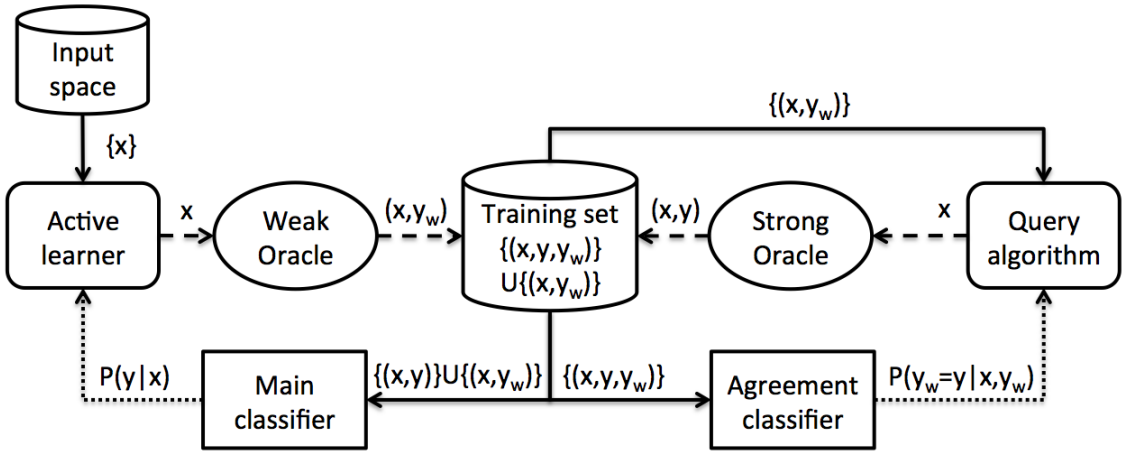
CHALMERS
UNIVERSITY OF TECHNOLOGY



Active learning of neural network from weak and strong oracles

Master's thesis in Complex Adaptive Systems

BJÖRN MATTSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:EX006

Active learning of neural network from weak and strong oracles

BJÖRN MATTSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Active learning of neural network from weak and strong oracles
BJÖRN MATTSSON

Supervisor: Daniel Langkilde, Recorded Future
Examiner: Lennart Svensson, Department of Signals and Systems

Master's Thesis 2017:EX006
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Flowchart describing the algorithm proposed in this thesis to do active learning from both a strong and a weak oracle.

Typeset in LaTeX
Gothenburg, Sweden 2016

Active learning of neural network from weak and strong oracles
BJÖRN MATTSSON
Department of Signals and Systems
Chalmers University of Technology

Abstract

When implementing deep learning techniques in real-world applications acquiring labeled data can be the most difficult and expensive part. Sometimes in this process there is both a weak (and cheap to ask) oracle as well as a strong (but expensive to ask) oracle available to which one can ask for labels to examples. The weak oracle can for example be non-expert mechanical turks or a rule-based system, whereas the strong oracle may be human experts that provide very reliable labels.

We propose an algorithm to do active learning in the presence of a weak and a strong oracle. A central part of the algorithm is an agreement classifier which predicts the probability of the weak oracle knowing the correct label for an example. However, at prediction time the agreement classifier is only assumed to be able to sort the examples after how much the weak oracle can be trusted, which is believed to be a key reason behind why the algorithm works in practice. A second key idea in the algorithm is that the agreement classifier does not only condition the classification on the information from the input space, but also on the label proposed by the weak oracle. A third idea that is examined is to leverage the probabilities supplied by the agreement classifier directly in the cost function of the main classifier.

To test the algorithm we built a test environment based on binary sentence classification as well as three types of synthetic weak oracles. The algorithm performs well with these three synthetic oracles. It manages to decrease the cross entropy on the learning task more per example queried to the strong oracle than what standard active learning do.

Keywords: active learning, deep learning, artificial neural network, natural language processing, machine learning

Acknowledgements

I want to express my gratitude to Recorder Future who gave me the opportunity to carry out this study, and who also offered me a fun and friendly working environment while doing so. Furthermore I want to give special thanks to Daniel Langkilde, supervisor of this thesis, for invaluable guidance and support throughout the process. I want to thank Lennart Svensson, examiner at Chalmers, for many insightful discussions and advices during the thesis. I also want to give thanks to David Lidberg, opponent for the thesis, for providing me with thorough feedback. Finally I want to thank Olof Mogren, PhD student of the machine learning research group at Chalmers, for helping me in the early stages of the project.

Björn Mattsson, Gothenburg, February 2017

Contents

List of Figures

List of Tables

1
Introduction

When designing and implementing methods based on machine learning for solving real-world problems, the task can be divided into two sub-tasks: first of all to construct a data set that can be used to train the model, and second to choose and implement a method (eg. SVM or deep-learning) that suits the problem. It is the second of these two sub-tasks that academia traditionally pays most attention to (assuming well defined data sets exist). In many cases how to define and create the data set can be what is most troublesome and costly in the industry.

At Recorded Future they have specialized in analyzing vast quantities of text, found through various sources on the Internet, and with that data deliver insights, in particular cyber security insights. One step in this process is to decide whether a sentence describes a cyber attack or not. Traditionally this has been done with a rule-based system, but recently a convolutional neural network model has been implemented based on the architecture in [4]. When moving from a rule-based system to a machine learning model, a data set on which the machine learning model can be trained has to be created. A data set for classification is traditionally created by letting a human assign labels to some examples taken from the data stream which the classifier will be applied on. This is expensive since many examples are needed which implies large human effort. This has motivated Recorded Future to invest in new techniques for building the data sets.

A first step in decreasing the cost of creating a data set is to realize that choosing examples to label randomly is inefficient. Many similar examples may be labeled which creates redundancy in the data set, and too much redundancy in the data set is unnecessary. A better approach is to use active learning. With active learning the idea is to iteratively build the training data set by choosing examples to label which the classifier thinks are difficult. The redundancy is thus reduced since when the classifier has learnt the label for some type of examples it will not ask for labels to similar examples.

At Recorded Future there is a rule-based system available which today does the classification. Therefore the question was asked whether this rule-based system could be leveraged when building the data set. This question was the motivation behind this thesis. Various approaches to this problem were considered in the beginning. The approach that was finally chosen was to modify the algorithm proposed by [11]. We propose an algorithm to do active learning in the presence of a strong oracle (which is expensive to ask but always supplies the correct label) and a weak oracle (which is cheaper to ask but sometimes supplies incorrect labels). Furthermore we assume that the errors made by the weak oracle are not distributed over the input space in any particular way. The weak oracle can be correct (or incorrect) both

close to and far away from the true decision boundary. Applied to the problem faced by Recorded Future the rule-based system takes the role as a weak oracle, and the annotators the role as a strong oracle.

1.1 Purpose

The purpose of this thesis was to investigate whether a preexisting rule-based system can be leveraged when doing active learning, and what characteristics this rule-based system has to have to do this successfully. The goal was to help Recorded Future develop methods that save money when building data sets for training machine learning models.

1.2 Objective

The main objective of this thesis was to define and implement a algorithm to do active learning in the presence of a strong and weak oracle. The algorithm proposed by [11] is theoretically complicated and would be difficult to implement as is. It contains many assumptions that probably do not hold in real-world applications. Therefore this thesis aimed at developing a practical implementation using key ideas from that paper.

To be able to develop the algorithm a test environment had to be built, in which different implementations could be tested. To do this, a good and large enough, data set to do the experiments had to be found and preprocessed for the task. Also, synthetic weak oracles that works with this data set had to be constructed.

1.3 Scope

This project considered active learning from a strong and a weak oracle when the task is binary classification. Experiments have been performed in a setting where the examples are sentences. The classification was done with a convolutional neural network. We assumed that the cost of asking the weak oracle was zero, thus we only considered how to minimize the number of label queries made to the strong oracle. Moreover, the strong oracle was assumed to always answer with the correct label.

1.4 Formal problem description

Here follows a formal description of the problem that also will serve as an introduction to the notation used in this report.

We are given unlabeled samples $\boldsymbol{x}$ distributed according to U that come from an input space $\mathcal{X}$. We can sample examples from this distribution $p_U(\boldsymbol{x})$. In this particular case the examples $\boldsymbol{x}$ are sentences. The examples $\boldsymbol{x}$ can be mapped to desired outputs y according to $p(y|\boldsymbol{x})$. The outputs belong to a binary output space

$\mathcal{Y} = \{-1, +1\}$. These labels y tells us which category the corresponding sentence belongs to. The joint distribution of examples and labels is defined as:

$$p_D(\boldsymbol{x}, y) = p(y|\boldsymbol{x})p_U(\boldsymbol{x}) \tag{1.1}$$

and is jointly distributed as D.

We have a classifier $f_{\boldsymbol{w}}$ whose weights, $\boldsymbol{w}$, we want to train, ideally so that $f_{\boldsymbol{w}}(y|\boldsymbol{x}) = p(y|\boldsymbol{x})$[1]. This means that for a given sentence $\boldsymbol{x}$ the prediction our classifier gives about whether a particular sentence describes a cyber-attack or not would perfectly reflect the reality. Since reaching a perfect mapping is impossible on most real world problems we have to measure the performance of the classifier. This will be done by using the cross entropy and the accuracy of $f_{\boldsymbol{w}}$. The cross entropy is defined as:

$$\mathrm{L}_D(f_{\boldsymbol{w}}) = -\,\mathbb{E}_{(\boldsymbol{x},y)\sim D}[\log f_{\boldsymbol{w}}(y|\boldsymbol{x})] \tag{1.2}$$

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\,\frac{1}{N}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{S}} \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i), \tag{1.3}$$

where the first equation represents the case where we have access to the underlying distribution D. The second equation represents the real world scenario where we only have N samples of $(\boldsymbol{x}_i, y_i) \in \mathcal{S}$. The accuracy of $f_{\boldsymbol{w}}$ is defined as:

$$\begin{aligned}\mathrm{acc}_D(f_{\boldsymbol{w}}) =& \mathbb{E}_{(\boldsymbol{x},y)\sim D}[\mathcal{I}\{f_{\boldsymbol{w}}(y|\boldsymbol{x}) > 0.5\}] \\ \mathrm{acc}_{\mathcal{S}}(f_{\boldsymbol{w}}) =& \frac{1}{N}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{S}} \mathcal{I}\left\{f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i) > 0.5\right\},\end{aligned}$$

where $\mathcal{I}\{\cdot\}$ equals 1 when the expression inside it is true and 0 otherwise. As for the cross entropy the two equations describes the cases where we have access to D and only have samples from it respectively. For simplicity we will from now let $y|\boldsymbol{x}$ denote the label y corresponding to $\boldsymbol{x}$ even when the context is not a conditional probability.

To be able to reach high accuracy we need to label enough examples sampled from U with their corresponding y so that $f_{\boldsymbol{w}}$ can learn an internal representation of how to distinguishes between the two classes. We can query the annotators (from now on called the strong oracle) S about the label, y_S of a specific sample, $\boldsymbol{x}$. This strong oracle knows the ground-truth but is expensive to ask. That it knows the ground-truth can be expressed as that it has the same conditional distribution as the real underlying mapping, i.e. $p(y_S|\boldsymbol{x}) = p(y|\boldsymbol{x})$. We can also ask the rule-based system (which we call the weak oracle) W about the label of a sample, y_W. The second oracle can be asked for free but does not always provides us with the right answer. This means that its conditional distribution does not agree with the real world and $p(y_W|\boldsymbol{x}) \neq p(y|\boldsymbol{x})$ on parts of the input space. Important to the definition of this problem is that the errors made by the weak oracle W are not uniformly noisy, nor necessarily more frequent close to the true decision boundary. This means that the errors made by the weak oracle can be more and less frequent in different regions.

[1]We will let $f_{\boldsymbol{w}}(y|\boldsymbol{x})$ represent the conditional probability returned by the classifier. When the y i omitted we assume the positive class, i.e. $f_{\boldsymbol{w}}(\boldsymbol{x}) = f_{\boldsymbol{w}}(y = +1|\boldsymbol{x})$

Furthermore the regions where the weak oracle frequently is wrong can occur far from the decision boundary separating the classes $\mathcal{Y}$ in the input space $\mathcal{X}$. The opposite is also true, regions where the weak oracle with high probability supplies the correct label could contain parts of the decision boundary in them.

Now, we are ready to formally define the problem. The aim is to develop an algorithm for creating a data set by sampling $\boldsymbol{x}$ from U and labeling some of them by using S and W so that we can reach a sufficiently good performance of $f_{\boldsymbol{w}}$ with as few queries to the strong oracle as possible. We want the performance of $f_{\boldsymbol{w}}$ to rise as quickly as possible for the queries we make to S.

1.5 Related work

An initial overview of the field of active learning can be found in [3]. In [1] they present a more theoretically well founded algorithm called agnostic active learning.

The active learning problem where different oracles, or labelers, exist has been analyzed from different perspectives. [6] for example discuss a problem arising when using crowdsourcing platforms to annotate data: how many times should examples be relabeled?

Some previous work analyzed the setting where two different types of labelers exist: both a strong one (that always supplies the correct label) and a weak one (which sometimes is wrong). The work presented in [10] porpose a theoretical algorithm to do this type of active learning. They assume that the weak oracles are more likely to be correct in label-homogeneous areas than close to the decision boundary. In [7] they analyze the problem where there is both a weak and a strong annotator in an online active learning setting.

The previous work that has largest resemblance with the algorithm proposed in this thesis is [11]. First of all the problem description is identical, with a weak oracle that rather than being uniformly noisy can be incorrect both close to and far away from the decision boundary. Both that paper and the algorithm proposed here use a separate classifier which has the purpose of learning the discrepancy between the strong and weak oracle. Finally, classical active learning is in both approaches considered to be an integral part in how to sample new points from the input space.

However the algorithm proposed in [11] is a theoretical algorithm with many assumptions that would be difficult to apply in practice. For example central to their algorithm is the assumption of knowledge about the Vapnik–Chervonenkis (VC) dimension (see [9] for an overview). This thesis has been an attempt to take the core ideas from from the algorithm they propose and apply it in practice for a real neural network. Therefore the structure of the algorithm proposed in chapter 4 deviates much from what was proposed by [11].

1.6 Thesis outline

In chapter 2 and 3 important background theory is introduced. The former describing how convolutional neural networks are used in this thesis to classify text, and the latter describing a standard active learning procedure. Chapter 4 contains the

contribution of this thesis, which is an algorithm to do active learning in the presence of both a strong and weak oracle. Chapter 5 describes the method used when evaluation the algorithm. In chapter 6 the results from the experiments are presented, these are later discussed in chapter 7. Finally the conclusions are presented in chapter 8.

2

Sentence classification with convolutional neural networks

The convolutional neural network for sentence classification used in this report is based on the architecture in [4]. In this chapter follows a description of that architecture. In figure 2.1 we can see a illustration of it.

The goal is to given a sentence $\boldsymbol{x}$ from an input space $\mathcal{X}$ predict the output label y belonging to an output space $\mathcal{Y}$. We thus want our classifier f_w to be able to represent the true conditional probability of the data $p(y|\boldsymbol{x})$ as good as possible by adjusting the weights $\boldsymbol{w}$.

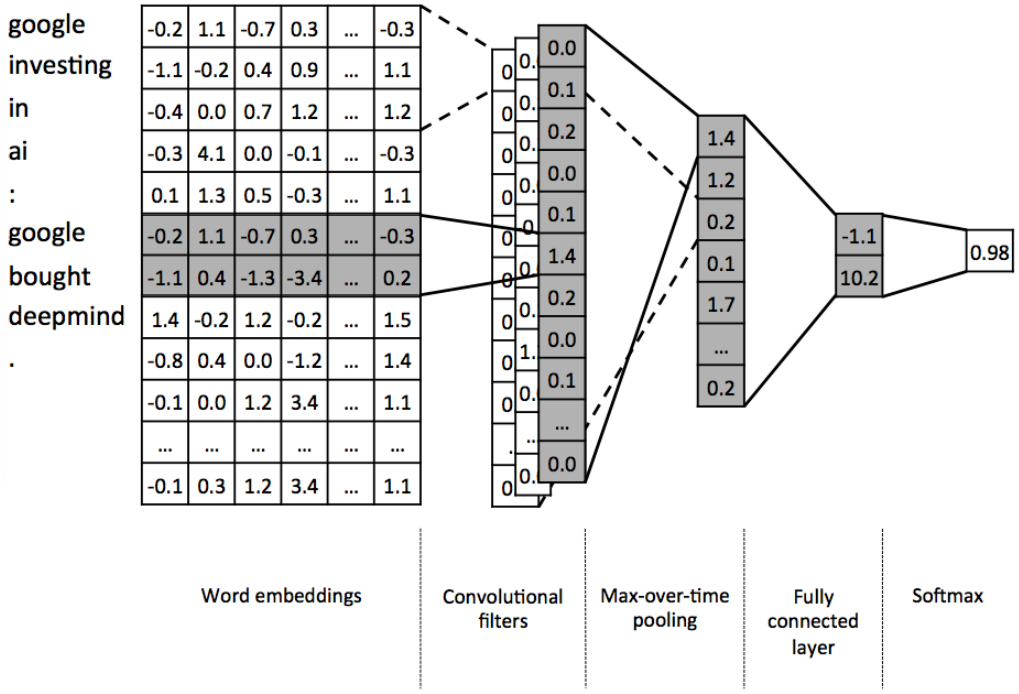


Figure 2.1: Illustration over the convolutional neural network architecture used in this thesis. For each layer a specific usage of the corresponding function is showed, the input values is marked as grey cells and the location of output value is showed with solid lines.

2.1 Word tokenization and word embedding

Before we can apply the methods offered by deep learning we need to do some preprocessing of the sentences. We want to transform the sentences in our data set to a standardized form and thus help the algorithm generalize to new examples. We put all characters in lower-case. We also tokenize the sentence into its constituting words and interpunctuations. With tokenization of a sentence we mean that we from $\boldsymbol{x}$ create a time ordered list, where each element either is a word or some interpunctuation.

Now we let the token i in our vocabulary be represented by a k-dimensional real-valued vector $\boldsymbol{w}_i^e$, commonly called word embedding. This representation will be shared between all sentences in our data set, so 'company' for example will always be represented by the same vector for all sentences. The word embeddings are trainable so that in the backpropagation step the vector $\boldsymbol{w}_i^e$ representing a specific word i can be changed. In total the number of trainable parameters in the word embedding is $k \cdot D$, where k is the word embedding dimension and D is the size of our vocabulary. When the word embeddings are trained their parameters have been found to change so that words with similar meanings move closer to each other in the embedding space. This property will help us in analyzing the sentences. Word embeddings also serve as a way to reduce the dimensionality of the space of tokens in the vocabulary.

We now have represented our sentence as a list of vectors, which effectively is a matrix. We denote this representation $\boldsymbol{x} \in \mathbb{R}^{n \times k}$ (for simplicity we let this representation keep the same notation as before), where n is the number of words in the sentence and k is the dimensionality of the word embeddings. We want all matrix representations to be of equal size, so sentences of varying length will either be truncated or padded to the length n.

2.2 Convolutional filters

We want to find patterns in the matrix representation $\boldsymbol{x} \in \mathbb{R}^{n \times k}$ of our sentence that can convey information on which class y it belongs to. Since words can mean different things when surrounded by different words we do not want to look at single words, but rather groups of words. We create a convolutional filter with a weight matrix and a bias term:

$$\boldsymbol{w}^f \in \mathbb{R}^{h \times k}, \quad b^f \in \mathbb{R},$$

where h is the filters window size. We want to use this filter to consider h consecutive words in our matrix representation of the sentence. Features c_i are extracted from our sentence by applying:

$$c_i = g\left(\sum_{j=1}^{h}\sum_{l=1}^{k} x_{i+j-1,l} \cdot w_{j,l}^f + b^f\right),$$

here g is a non-linear function. In this thesis a rectified linear unit is used:

$$g(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Since words with similar meanings will be trained to have word embeddings that are similar to each other the extracted feature c_i can be expected to be similar when applied to chunks of sentences with similar meanings. If we continuously apply the filter c_i to all possible consecutive sequences of words for a sentence we get a feature map:

$$\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-h+1}]^T \in \mathbb{R}^{n-h+1}.$$

This feature map represents how much of the pattern detected by the filter defined by $\boldsymbol{w}^f$ and b^f can be found in different parts of the sentence. If we want our algorithm to detect different kinds of patterns in our sentence we can create many different filters. They can have different window sizes to detect patterns of different size. Each filter should have its own trainable parameters $\boldsymbol{w}^f$ and b^f. The total amount of trainable parameters introduced in the convolution step equals:

$$\sum_{i=1}^{m}(h_i k + 1),$$

where h_i is the window size for filter i out of m filters.

2.3 Max-over-time pooling

When classifying sentences there, to some degree, exists translational-invariance (or time-ivariance). If you for example want to decide if a movie review is positive or negative, it does not matter if you read "I loved this movie" as first or last statement in the movie review, the meaning will be the same. With this in mind it is natural to only consider the most prominent feature found by each filter in the convolutional step. If the filter that has been tuned to detect "I loved this movie" finds a similar pattern in one part of the sentence it is of marginal value to know how similar all other chunks of the sentence were to that pattern. To only consider the pattern most similar to the filter we have applied is called max-over-time pooling. Mathematically we express this as that we take the maximum value of the vector $\boldsymbol{c}$ belonging to a specific filter:

$$\hat{c} = \max(\boldsymbol{c})$$

2.4 Fully connected layer and softmax

Finally we want to synthesize the information in the features extracted by the convolution and max-over-time pooling layer. The features extracted by the convolution layer are concatenated in the vector $\boldsymbol{z}$ as:

$$\boldsymbol{z} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m]^T \in \mathbb{R}^m,$$

where m is the number of different filters we have used. To transform this vector into output values we use the output layer weights:

$$\boldsymbol{w}^o \in \mathbb{R}^{2\times m}, \quad \boldsymbol{b}^o \in \mathbb{R}^2,$$

where we have used 2 since we do binary classification. The number of trainable parameters in this final step is $2 \cdot (m + 1)$.

We multiply the feature vector with the output weights and add the bias to obtain the vector of output values $\boldsymbol{y}$ as:

$$\boldsymbol{y} = \boldsymbol{w}^o \cdot \boldsymbol{z} + \boldsymbol{b}^o.$$

The values in $\boldsymbol{y}$ here represents how much the algorithm believes the input example $\boldsymbol{x}$ belongs to a specific class. A more easily interpretable measure would be the probability that the it belongs to the class. We can transform the value into a probability by using the softmax function:

$$p(y = y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}.$$

We have now reached a point where we have an algorithmic and mathematical representation of $p(y|\boldsymbol{x}, \boldsymbol{w})$ where $\boldsymbol{w}$ are all the trainable parameters, or weights, in the model. We will denote this neural network model as $f_{\boldsymbol{w}}(y|\boldsymbol{x})$. The next step is how we can modify the weights $\boldsymbol{w}$ so that this algorithmic representation approximates the true distribution $p(y|\boldsymbol{x})$.

2.5 Cost function and backpropagation

To train our neural network we use the cross entropy, as defined earlier:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\frac{1}{N} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}} \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i), \tag{1.3}$$

where $\mathcal{S}$ is the training data set that we have available. A lower value of the cost function $\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}})$ equals a better fit of our parameters $\boldsymbol{w}$ to the data set $\mathcal{S}$. If there are large imbalances in the number of examples of the different classes in $\mathcal{S}$ there is a risk that the neural network only will learn the distribution between the classes in $\mathcal{S}$. To make sure that this does not happen we can add a class-specific weight c_y to the cost function (1.3) as:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\frac{1}{N} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}} c_{y_i} \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i). \tag{2.1}$$

The value of c_y is defined as:

$$c_y = \frac{\sum_y N_y}{2N_y},$$

where N_y represents the number of examples of class y in the training data. The 2 in the denominator is used to make $\mathbb{E}[c_j] = 1$.

Minimizing our cost function is difficult since it is non-convex. However when optimizing neural networks it is often good enough to find a local minimum. To find a local minimum in a neural network it is common to use the gradient of the cost function. The gradient is defined as:

$$\nabla \mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = \left[\frac{\partial \mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}})}{w_1}, \frac{\partial \mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}})}{w_2}, \ldots, \frac{\partial \mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}})}{w_n} \right]^T.$$

To find the derivative of $\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}})$ in all directions of $\boldsymbol{w}$ we use backpropagation. The idea is to use the chain-rule to expand an arbitrarily complex function into atomic parts which are easy to differentiate and then multiply these parts together. If we for example have the function $f(g(\boldsymbol{w}))$ we could find the the derivative $\frac{\partial f}{w_i}$ as:

$$\frac{\partial f}{w_i} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w_i}.$$

An interesting property with backpropagation is that to calculate the derivative in different directions i we only have calculate $\frac{\partial g}{\partial w_i}$ in each direction, it suffices to calculate $\frac{\partial f}{\partial g}$ once. This leads to that it is efficient to calculate the gradient in all directions of $\boldsymbol{w}$.

When we have found the gradient there are many different ways for how to do the optimization, the most simple way is by gradient descent, in this report however we use a variant of that called Adam [5].

2.6 Early stopping and regularization

One major risk with our training procedure is that the model learns to represent the distribution in our training set $\mathcal{S}_{\text{train}}$ well but does not generalize well to the real data distribution D of $\boldsymbol{x}$ and y. When this happens we say that the model overfits. To stop this from happening we use a validation set $\mathcal{S}_{\text{val}}$ separated from the training set $\mathcal{S}_{\text{train}}$. On the validation set we continuously evaluate the cross entropy but we do not use it for training. When the cross entropy starts to increase on $\mathcal{S}_{\text{val}}$ we can conclude that the model is overfitting and we stop training it at that point. We call this procedure for early stopping.

Besides doing early stopping we can also avoid overfitting by regularizing the network. We will in this thesis use two regularization techniques: dropout and l_2-regularization. Applying dropout to a layer means to multiply each node in that layer with a random variable R_d (drawn independently for each node) defined as:

$$R_d = \begin{cases} 1/P_d & \text{with probability } P_d, \\ 0 & \text{with probability } 1 - P_d, \end{cases}$$

where P_d is the probability of keeping each node. The nodes that are kept are multiplied with $1/P_d$ to ensure that $\mathbb{E}[R_d] = 1$, this to avoid introducing any scaling of the signal. Dropout is applied to the layer that we get as output from the max-over-time pooling layer. In reality we as input to the fully connected layer thus use:

$$\hat{c}_i^d = R_{d,i} \cdot \hat{c}_i.$$

Dropout is only used at training time and not when we evaluate the performance on the validation set or use the model to do predictions. Intuitively dropout can be understood as that we restrict the model to only have a subset of the convolutional filters available at training time. If it can only access a random subset of feature extracting convolutional filters at training time the classification of an example has

to depend on many different filters. Consequently we avoid creating features that are highly specialized to just some specific examples which would lead to overfitting.

l_2-regularization is applied as an extra penalty to the cost function that penalizes large weights. The full cost function after l_2-regularization is:

$$C(\boldsymbol{w}) = \mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) + \lambda \frac{1}{2} ||\boldsymbol{w}^o||_2^2,$$

where we have applied the penalty to the output weights $\boldsymbol{w}^o$, and λ is a penalty parameter that we use to decide how much we should penalize large weights. We use $\frac{1}{2}$ to get $\lambda \boldsymbol{w}^o$ when differentiating the l_2-penalty with respect to $\boldsymbol{w}^o$, which is a nice and intuitive form. This cost term effectively pulls the gradient in the direction of the origin, and weights further out will be pulled harder. The l_2-regularization can be interpreted as that we do not want any single feature to be too dominant when deciding the class of examples. Each time the training algorithm does not find a specific convolutional filter useful it will pull the corresponding weights slightly towards zero. As for dropout this will create filters that are general and applicable to many different examples. It should be noted that in [4] they instead of l_2-regularization use a different approach to avoid too large values of $\boldsymbol{w}^o$.

3

Active learning

In this chapter a brief overview of active learning is given, as it plays a central role in the algorithm proposed in chapter 4. For a more thorough discussion of active learning see [3]. Active learning in its most distilled form only considers one oracle, which always supplies the correct label. The problem can be stated as: We want to create a data set by sampling $\boldsymbol{x}$ from $\mathcal{X}$ distributed as U and assigning labels y to some of them by querying the oracle, so that we can reach as good performance on $f_{\boldsymbol{w}}$ as possible with as few queries to the oracle as possible. We want the performance of $f_{\boldsymbol{w}}$ to improve as quickly as possible with the number of queries made to the oracle.

3.1 Learning by random sampling

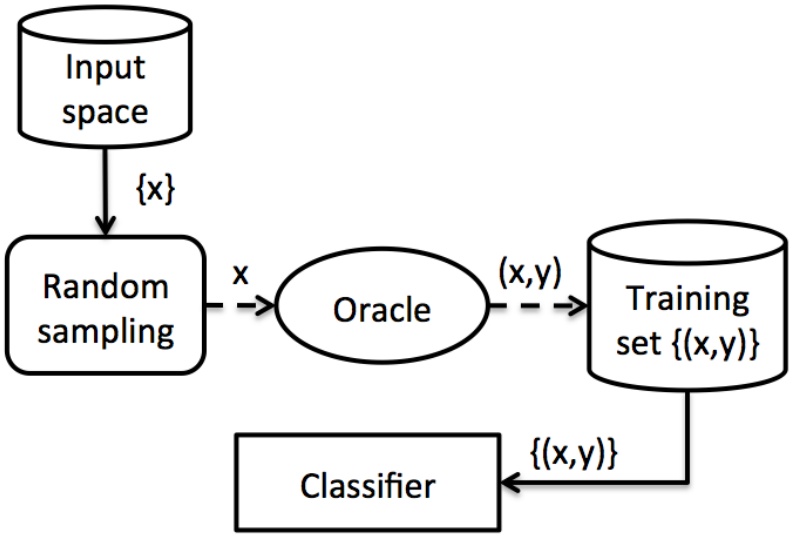


Figure 3.1: Flowchart describing the random sampling procedure to construct a training data set for the classifier. Lines represent: single data points (dashed), data set (solid). Boxes represent: data sets and distributions (cylinder), classifier (square, sharp edges), algorithm (square, rounded edges), labeling oracle (ellipse).

If we disregard the cost of asking the oracle and only care about the final performance of our classifier $f_{\boldsymbol{w}}$ we could sample U randomly and ask for the labels of all the examples. In figure 3.1 we can see a flowchart of how this would be done. From the input space $\mathcal{X}$ we randomly choose samples $\boldsymbol{x}$ according to U which we then pass along to our oracle that label according to $p(y|\boldsymbol{x})$. We put the example $\boldsymbol{x}$

and its corresponding label y in our training data set. Finally, this training data set is used when training our classifier $f_{\boldsymbol{w}}$. When the classifier is trained we measure its performance on some separate data set, if its performing well enough we are done. If not we continue the sampling procedure as in the flowchart until the classifier performs well enough.

The random sampling algorithm above works well for problems were little data is needed and/or the labeling procedure is cheap. In most real-world applications however that is not the case. One example when random sampling does not work well is when for most of the examples $\boldsymbol{x}$ it is easy to predict their label y, and only for a tiny fraction of the examples distributed as U the prediction is difficult. For the classifier to be able to learn these difficult examples one would have to sample and label so many examples $\boldsymbol{x}$ from U such that enough examples from the tiny fraction of examples that are difficult are present in the training data set. This leads to many label queries to the oracle which can be expensive.

3.2 Active learning

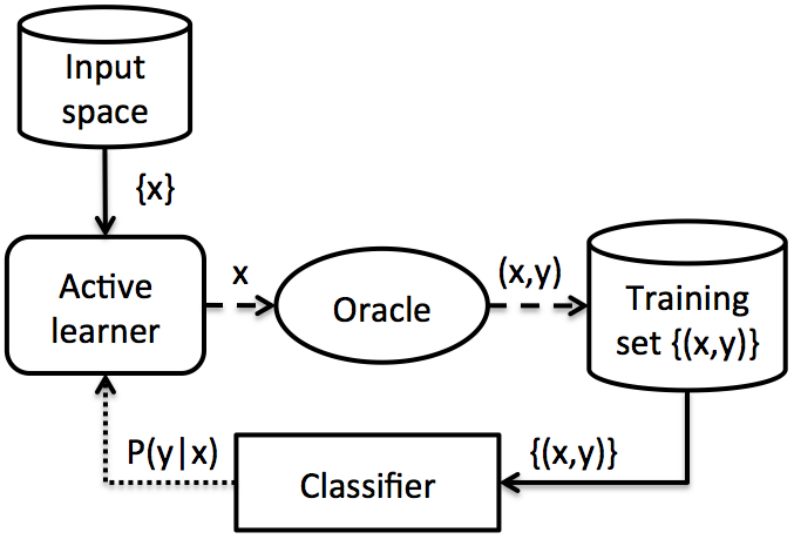


Figure 3.2: Flowchart describing the active learning algorithm. Lines represent: single data points (dashed), data set (solid), classifier predictions (dotted). Boxes represent: data sets and distributions (cylinder), classifier (square, sharp edges), algorithm (square, rounded edges), labeling oracle (ellipse).

The idea behind active learning is to choose as difficult examples as possible, as they probably contain information that is more valuable to the classifier. In figure 3.2 we can see a simple flowchart describing a simple active learning procedure. Instead of just randomly sample examples from the input distribution U we let the classifier $f_{\boldsymbol{w}}$ assign probabilities of them belonging to the different classes $\mathcal{Y}$. We then choose the examples which the classifier is most uncertain of. In the case of binary classification being uncertain of an example $\boldsymbol{x}$ means to have a low value of $|f_{\boldsymbol{w}}(\boldsymbol{x}) - 0.5|$. In algorithm 1 we see a description of this process. If U is infinite it

Algorithm 1 Active learning algorithm (for binary classification) AL($U, f_{\boldsymbol{w}}, n$)

1: Input: distribution of unlabeled samples U, classifier $f_{\boldsymbol{w}}$, number of samples to query n
2: Sort $\boldsymbol{x} \in U$ after $|f_{\boldsymbol{w}}(\boldsymbol{x}) - 0.5|$ in ascending order
3: $Q \leftarrow \{$the n first examples $\boldsymbol{x}$ from the sorted list$\}$
4: **return** Q

is not feasible to sort all $\boldsymbol{x} \in U$. In that case we would use some sampled subset of U. An alternative is to accept points for which $f_{\boldsymbol{w}}(\boldsymbol{x})$ lies in some interval around 0.5.

Instead of randomly sampling U we say that we sample from the decision boundary of $f_{\boldsymbol{w}}$. Consequently the distribution over examples in the training data set and the input space will differ. The difference is that the training data set is more heavily concentrated around difficult regions of $\mathcal{X}$, for which the classifier needs more examples to learn.

Now we return to the example discussed above, where just a tiny fraction of the input distribution U are difficult. We ask ourselves how the active learning algorithm would sample data in this case? In the beginning all examples in U would be close to the decision boundary of the classifier as it has no training set to train on. Therefore it would choose example from all over U. Quite quickly though it would learn how to predict these easy examples and it would assign them high probability of belonging to one of the classes. However, the classifier would continue to struggle with the tiny subset of difficult examples, and it would not assign them high probabilities of belonging to either class. The active learning algorithm would thus be able to cherry-pick these examples and pass them along to the oracle to let them form part of the training set.

When constructing the data set ideally one would pick and label one example from $\mathcal{X}$, add it to the training data set, retrain the classifier and then repeat the procedure. This is in practice not a feasible approach since all those steps takes time. In practice active learning is done batch-wise. This means that instead of just choosing one new example each iteration a batch of new examples are chosen from $\mathcal{X}$, labeled, and added to the training data set.

Above the simplest available active learning version was introduced. This is the method that will be used in this thesis and that forms a part of the algorithm introduced in section 4. There are more advanced and theoretically complicated versions. An introductory overview can be found in [3]. One algorithm they discuss is called "Query by Committee". This algorithm maintains a set of classifiers which are trained on the training data set. After training they will reach slightly different minima. Therefore they will also make predictions that differ. Examples for which the classifiers disagree about which label it should be given are chosen for labeling by the strong oracle and then added to the training data set. In [1] active learning is discussed from a more theoretical approach.

4

Bi-oracle learning algorithm

This chapter will introduce the algorithm developed in this thesis to solve the active learning problem from section 1.4. We will refer to this algorithm as *bi-oracle learning algorithm*. First an overview over the algorithm will be given. After that the design of the different core parts of the algorithm will be detailed.

4.1 Overview of the algorithm

To make $f_{\boldsymbol{w}}$ learn how to classify the examples correctly we have to label enough examples of $\boldsymbol{x}$. The choice of examples to label does not necessarily have to be distributed according to U. A more efficient way of choosing examples to label can be achieved through active learning as described in chapter 3. In this report active learning will be done simply by sampling from the decision boundary of $f_{\boldsymbol{w}}$.

Since the weak oracle W is not uniformly noisy but rather have different probabilities of being correct in different regions we can describe this behaviour with the distribution $p(y = y_W|\boldsymbol{x})$. In this thesis we assume that the cost of querying the weak oracle is zero. We can then query the weak oracle for the labels to all examples, thus y_W will always be available. Therefore we will condition the distribution above also on y_W, and we obtain:

$$p(y = y_W|\boldsymbol{x}, y_W). \tag{4.1}$$

We will train a separate classifier to learn this distribution, or in other words to predict whether the weak and strong oracles agree. We will denote this agreement classifier as $f^{ag}_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ are trainable weights. We want to train this classifier to a high similarity with the distribution in (4.1) so that $f^{ag}_{\boldsymbol{\theta}} \approx p(y = y_W|\boldsymbol{x}, y_W)$.

A key idea in the proposed algorithm (shared with [11]) is that if we manage to achieve a good approximation $f^{ag}_{\boldsymbol{\theta}} \approx p(y = y_W|\boldsymbol{x}, y_W)$ we can use this classifier to predict whether a label provided by the weak oracle have a high probability of coinciding with the true label or not. If they have a high probability of being equal (i.e. $p(y = y_W|\boldsymbol{x}, y_W) \approx 1$) we do not have to ask the strong oracle for the label. When we are uncertain about whether they are equal or not (i.e. $p(y = y_W|\boldsymbol{x}, y_W) \approx 0.5$) we have not got enough faith in the label provided by the weak oracle and we will have to ask the strong oracle for the label of that example. A third possibility where $p(y = y_W|\boldsymbol{x}, y_W) \approx 0$ exists. This means that the labels differs with high probability, in this case we can assign the opposite label of what the weak oracle suggested. Whether this last case will occur or not in real-world problems is questionable. It effectively means that we have a weak oracle that for some regions of the input space $\mathcal{X}$ always are wrong about the label. However, as

Algorithm 2 Bi-oracle learning algorithm

Input: distribution of unlabeled samples U, strong oracle S, weak oracle W, main classifier $f_{\boldsymbol{w}}$, agreement classifier $f^{ag}_{\boldsymbol{\theta}}$, number of samples to query n_t, share of examples to put in the validation set s_v and desired share of correct labels s_c in the train set.
Let $t \leftarrow 0$, $\mathcal{S}_{\text{train}} \leftarrow \emptyset$ and $\mathcal{S}_{\text{val}} \leftarrow \emptyset$
repeat
    $Q_t \leftarrow \text{AL}(U, f_{\boldsymbol{w}}, n_t)$ ▷ see algorithm 1
    Randomly split Q_t into $Q_{\text{val},t}$ and $Q_{\text{train},t}$ according to s_v
    Let $\mathcal{S}_{\text{val}} \leftarrow \mathcal{S}_{\text{val}} \cup \{\{\boldsymbol{x}, y_S, y_W\}\ \forall \boldsymbol{x} \in Q_{\text{val},t}\}$ by querying S and W
    Choose the trust-threshold $T \leftarrow T(s_c, f^{ag}_{\boldsymbol{\theta}}, \mathcal{S}_{\text{val}})$ ▷ see section 4.2
    Let $\mathcal{S}_{\text{train}} \leftarrow \mathcal{S}_{\text{train}} \cup \{\{\boldsymbol{x}, y_W\}\ \forall \boldsymbol{x} \in Q_{\text{train},t}\}$ by querying W
    for each $\boldsymbol{x} \in \mathcal{S}_{\text{train}}$ **do**
        if $y_S|\boldsymbol{x} \in \mathcal{S}_{\text{train}}$ **then**
            Continue
        else if $f^{ag}_{\boldsymbol{\theta}}(y = y_W|\boldsymbol{x}, y_W) > T$ **then**
            Let $\mathcal{S}_{\text{train}} \leftarrow \mathcal{S}_{\text{train}} \cup y_{S'} = y_W|\boldsymbol{x}$
        else if $f^{ag}_{\boldsymbol{\theta}}(y \neq y_W|\boldsymbol{x}, y_W) > T$ **then**
            Let $\mathcal{S}_{\text{train}} \leftarrow \mathcal{S}_{\text{train}} \cup y_{S'} = \neg y_W|\boldsymbol{x}$
        else
            Let $\mathcal{S}_{\text{train}} \leftarrow \mathcal{S}_{\text{train}} \cup y_S|\boldsymbol{x}$ by querying S
        end if
    end for
    Train $f_{\boldsymbol{w}}$ on $\{y_S, \boldsymbol{x}\} \cup \{y_{S'}, \boldsymbol{x}\}$ using $\mathcal{S}_{\text{train}}$ for training and $\mathcal{S}_{\text{val}}$ for supervision
    Train $f^{ag}_{\boldsymbol{\theta}}$ on $\{y_S = y_W, \boldsymbol{x}\}$ using $\mathcal{S}_{\text{train}}$ for training and $\mathcal{S}_{\text{val}}$ for supervision
    Calculate performance metrics
    $t \leftarrow t + 1$
until Performance is good enough
return Main classifier $f_{\boldsymbol{w}}$, train set $\mathcal{S}_{\text{train}}$ and val set $\mathcal{S}_{\text{val}}$

we will see it is natural to include the possibility of this case in our algorithm so therefore we will do that.

The main algorithm is outlined in Algorithm 2 and visualized in figure 4.1. As input it takes the distribution U over the input space $\mathcal{X}$ from which we can sample unlabeled examples, the strong and the weak oracles S and W, the main classifier $f_{\boldsymbol{w}}$ that we want to achieve a high accuracy on predicting $p(y|\boldsymbol{x})$, and the agreement classifier $f^{ag}_{\boldsymbol{\theta}}$ described above. There are also three settings that the user have to choose: n_t which is the number of samples to query at time step t, the share of samples to be put in the validation set, s_v, and the desired share of correct labels in the training set, s_c. In section 4.2 we present the motivation behind s_c and how it is used.

For each iteration through the algorithm it first chooses a set Q_t of new examples. This is done through some active learning method. In this report it is simply done by choosing the examples which the main classifier $f_{\boldsymbol{w}}$ is most uncertain how to label. It then proceeds to split up this set of previously unseen examples between

the validation set and the training set. For all the examples in the validation set, the algorithm queries both the strong and weak oracle. Next, the algorithm chooses a trust threshold, T. The trust threshold reflects for what levels of confidence we should trust our agreement classifier. How this is chosen is discussed in section 4.2.

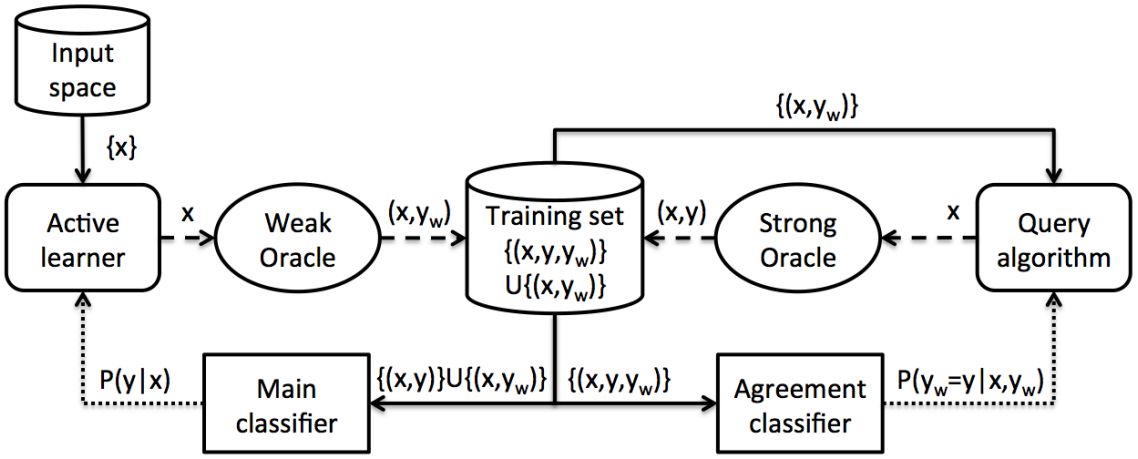


Figure 4.1: Flowchart describing a simplified version the proposed algorithm. The path of the examples in the validation set it not included. The details regarding s_c and T are left out. Lines represent: batches of data (dashed), access to data sets and distributions (solid), classifier predictions (dotted). Boxes represent: data sets and distributions (cylinder), classifiers (square, sharp edges), decision algorithms (square, rounded edges), labeling oracles (ellipse).

Having chosen a trust threshold, T, for our agreement classifier we now proceed to process the the training set. This is done on lines 8-19. First all new examples are added, but only the weak oracle is queried about their label. When this has been done, the examples for which the agreement classifier is certain enough is either correctly labeled or incorrectly labeled by the weak oracle are temporarily labeled as either $y_{S'} = y_W$ or $y_{S'} = \neg y_W$ respectively. We can see here that it is easy to include the second case, where the agreement classifier is certain the label from the weak oracle is incorrect. The examples for which the agreement classifier is uncertain the strong oracle is queried and their true labels y_S are stored.

Next step is to retrain the two classifiers on the training set. This is done on line 20 and 21. The agreement classifier is trained only on the subset of examples for which we have queried the strong oracle. The process will be described more in detail in section 4.3. The main classifier on the other hand is trained on all $\boldsymbol{x} \in \mathcal{S}_{\text{train}}$. The examples for which y_S is unknown we use $y_{S'}$ as a substitute. How to define this cost function when training the main classifier is discussed in section 4.4.

Finally some performance metrics, such as the cross entropy and accuracy are evaluated. This should be calculated using a data set separate from the rest of the algorithm.

4.2 Controlling the error in the data

We want to be able to avoid querying the strong oracle when possible but at the same time maintain a training data set that does not contain too many errors. If our training data set contains too many errors our main classifier will learn a poor representation of the true decision boundary and will not achieve good performance. To prevent this from happening we want to be able to control the share of examples that are labeled correctly in the training data set, we call this s_c.

Given the total size of our training set, $|\mathcal{S}_{\text{train}}|$, and s_c we can calculate how many erroneous labels we can afford as:

$$E = (1 - s_c)|\mathcal{S}_{\text{train}}|. \tag{4.2}$$

For the examples which the agreement classifier is more certain belongs to any of the two classes than T the strong oracle will not be queried. This will introduce errors in the training data set. The number of errors introduced can be estimated as:

$$\sum\nolimits_{\{\boldsymbol{x} \in \mathcal{S}_{\text{train}} \,:\, p(y_{S'} = y_S|\boldsymbol{x}) > T\}} \mathbb{E}[p(y_{S'} \neq y_S|\boldsymbol{x})].$$

Since we wanted the total number of errors to be smaller than E we can combine the two previous equations to:

$$\sum\nolimits_{\{\boldsymbol{x} \in \mathcal{S}_{\text{train}} \,:\, p(y_{S'} = y_S|\boldsymbol{x}) > T\}} \mathbb{E}[p(y_{S'} \neq y_S|\boldsymbol{x})] < (1 - s_c)|\mathcal{S}_{\text{train}}|. \tag{4.3}$$

Now we can choose a T as large as possible without violating the inequality.

Now we have to find a way to calculate the left hand side of equation (4.3) in practice. A naive approach would be to use the approximation $p(y = y_W|\boldsymbol{x}, y_W) \approx f^{ag}_{\boldsymbol{\theta}}$. $p(y = y_W|\boldsymbol{x}, y_W)$ can then easily be remapped to $p(y_{S'} = y_S|\boldsymbol{x})$ and $p(y_{S'} \neq y_S|\boldsymbol{x})$. However, while this approach might seem plausible in theory it turns out that in practice it does not work. In practice the value of the agreement classifier is a bad approximation of the probability $p(y_W = y_S|\boldsymbol{x}, y_W)$, as we will see in section 6.2.

We also have a validation set with samples distributed just as the training set but with both the labels y and y_W assigned. A possibility is to use this set to obtain T given a fixed s_c on the validation set. We create a hypothetical training set of the data in the validation set. Then we go through all the examples in this hypothetical training set, starting with the ones the agreement classifier is most certain about. This can be done by passing all the examples to $f^{ag}_{\boldsymbol{\theta}}$ and sorting them. For each example that the weak and strong label differs we increment the counter of the numbers of hypothetical errors. When we hit the critical point of $(1 - s_c)|\mathcal{S}_{\text{val}}|$ number of errors we stop and choose the value of $f^{ag}_{\boldsymbol{\theta}}$ for that example to be our trust threshold, T. All the examples the agreement classifier is less certain about are assigned correct strong labels in this hypothetical training set and will thus not be processed at the next iteration when it is time to choose a new trust threshold. Because $\mathcal{S}_{\text{val}}$ and $\mathcal{S}_{\text{train}}$ are sampled identically we will get similar values of s_c when calculated like this for the validation set. In conclusion we will in reality not keep s_c fixed on the training set, but rather s_c on the validation set. However,

they will be sufficiently close to each other. When looking at the results in section 6.2 we will see that this method works very well in practice.

We are now able to control s_c, which is the share of labels used in training the main classifier that are correct. A high value of s_c should lead to a conservative algorithm that avoids errors in the training data set. A lower value of s_c should lead to a more aggressive algorithm that exploits the availability of the weak oracle to a larger extent but will also lead to more errors in the training data set. This parameter can of course be changed as the learning process proceeds.

4.3 Agreement classifier

The agreement classifier plays a central role in the algorithm. To train it we use all examples for which we have queried the strong label. The examples that end up in the training set of the agreement classifier are those which both the main classifier and agreement classifier are uncertain about. This could be understood as that we implicitly do active learning of the agreement classifier, but only on the parts of the input space which are relevant for the main classifier. Because of this, the distribution of the training set for the agreement classifier differs from that of the main classifier. Their validation sets on the other hand are the same.

The purpose of the agreement classifier is to predict whether the weak and strong oracle will provide the same label for a given example or not. To perform this prediction the agreement classifier can use two types of information: the example $\boldsymbol{x}$, and the label predicted by the weak oracle y_W. Remember that we conditioned the probability of agreement on both $\boldsymbol{x}$ and y_W in equation (4.1) to allow for this. This leads us to that we could write the classifier as $f^{ag}_{\boldsymbol{\theta}}(\boldsymbol{x}, y_W)$, however we will mostly continue to use $f^{ag}_{\boldsymbol{\theta}}$ for simplicity.

In the following two subsections we will discuss why it is important to feed the classifier both with the information obtained through $\boldsymbol{x}$ and through y_W. This discussion will be based on two different types of characteristics that a weak oracle can have. Most weak oracles will of course rather have a combination of these than being clear-cut cases.

4.3.1 High accuracy on parts of the input space

One characteristic that a weak oracle can have is to be expert on a particular region of the input space, where it knows how to classify the examples. Formally this means that $p(y|\boldsymbol{x}) \approx p(y_W|\boldsymbol{x})$ when $\boldsymbol{x} \in \mathcal{X}_W \subset \mathcal{X}$. On other parts of the input space it does not know how to classify the examples and therefore $p(y|\boldsymbol{x}) \neq p(y_W|\boldsymbol{x})$ when $\boldsymbol{x} \notin \mathcal{X}_W \subset \mathcal{X}$. If the weak oracle is a rule-based system it will have these properties if it has been configured very well to a particular problem (either intentionally or unintentionally). In this case the new classifier seeks to replace that rule-based system but also be able to solve other problems. One example would be a credit fraud detection system fine-tuned on one specific type of cardholder, say females aged 25-35, whereas the new machine learning method should work for all types of customers.

For weak oracles with these property the task of the agreement classifier is to detect when a new sample, $\boldsymbol{x}$, is inside the region $\mathcal{X}_W$ where the weak oracle is certain. In the example above, just a simple classifier relying on whether the cardholder $\boldsymbol{x}$ falls in that specific category would of course suffice. However, in many cases the weak oracle has been built organically over a long time period and it is hard to tell on which parts of the input space it performs well and on which parts it does not. This leads to the weak oracle being unintentionally rather than intentionally configured to perform well on $\mathcal{X}_W$. This is the argument behind why as agreement classifier use a classifier able to learn complex decision boundaries in $\mathcal{X}$.

4.3.2 High accuracy for some output class

A second type of trait that a weak oracle can have is that it has a high precision on a particular class. So that if the weak oracle proposes that particular class we can conclude that it is correct with high probability. The cost of false positives and false negatives usually differs, and real-world rule-based systems to do prediction are usually configured to avoid one of them. For example false positives is what is avoided at Recorded Future; too many false positives would lead to its clients getting spammed with unimportant data. In health care on the other hand false negatives is what is costly, because you do not want to tell a patient she is healthy if she in the end turns out to have terminal disease.

From this we can conclude that important information of the accuracy of the weak oracle can be inherent in the label y_W that the weak oracle produces. To take advantage of this information only it would of course suffice with a simple classifier only relying on y_W. However important information regarding the accuracy of the weak oracle might also be encoded in the sample $\boldsymbol{x}$, which would be lost if the classifier only took y_W as input.

4.4 Cost function with soft labels

When training neural networks to do binary classification the cross entropy is commonly used as cost function. The cross entropy was defined in equation (1.3) as:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\frac{1}{N}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{S}} \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i),$$

where $\mathcal{S}$ is the data set we use to evaluate it. At training time this cost is calculated and the errors are backpropagated. The cost function can be extended to the case where we have probabilities on the values of the examples. We start of with the formal definition of the cross entropy from equation (1.2):

$$\mathrm{L}_D(f_{\boldsymbol{w}}) = -\mathbb{E}_{(\boldsymbol{x},y)\sim D}[\log f_{\boldsymbol{w}}(y|\boldsymbol{x})].$$

This can be written as:

$$\mathrm{L}_D(f_{\boldsymbol{w}}) = -\int_{\mathcal{X},\mathcal{Y}} p_D(\boldsymbol{x},y)\log f_{\boldsymbol{w}}(y|\boldsymbol{x}) = -\int_{\mathcal{X},\mathcal{Y}} p(y|\boldsymbol{x})p_U(\boldsymbol{x})\log f_{\boldsymbol{w}}(y|\boldsymbol{x}),$$

where we have used equation (1.1). If we have samples of $\boldsymbol{x}_i$ distributed according to $\mathcal{S}$ the loss over those examples can thus be evaluated as:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\sum_{y}\sum_{\boldsymbol{x}_i \in \mathcal{S}} p(y|\boldsymbol{x}_i) \log f_{\boldsymbol{w}}(y|\boldsymbol{x}_i).$$

In the binary case where y only can take two values it can be easier to understand the expanded expression:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\sum_{\boldsymbol{x}_i \in \mathcal{S}} \left[p(y=-1|\boldsymbol{x}_i) \log f_{\boldsymbol{w}}(y=-1|\boldsymbol{x}_i) + p(y=+1|\boldsymbol{x}_i) \log f_{\boldsymbol{w}}(y=+1|\boldsymbol{x}_i)\right]. \tag{4.4}$$

Applied to our problem we can for the examples where only y_W is available use the agreement classifier $f^{ag}_{\boldsymbol{\theta}}$ to approximate $p(y = y_W|\boldsymbol{x}_i)$. With this approximation we can use equation (4.4) to define the loss as:

$$\begin{aligned}\mathrm{L}_{\mathcal{S}_S \cup \mathcal{S}_W}(f_{\boldsymbol{w}}) = &-\sum_{(y_i,\boldsymbol{x}_i)\in\mathcal{S}_S} \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i) \\ &-\sum_{(y_i,\boldsymbol{x}_i)\in\mathcal{S}_W} [f^{ag}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \log f_{\boldsymbol{w}}(y_i|\boldsymbol{x}_i) + (1 - f^{ag}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \log f_{\boldsymbol{w}}(\neg y_i|\boldsymbol{x}_i)]. \end{aligned} \tag{4.5}$$

In conclusion we now have two possibilities for calculating the cross entropy when we train our classifier. Either we round off the values given by $f^{ag}_{\boldsymbol{\theta}}$ and feed the examples directly into equation (1.3), or we take the information given by $f^{ag}_{\boldsymbol{\theta}}$ into account and use equation (4.5) instead. Both possibilities are equal for the examples where we have the strong label available, but they differ for the examples where we only have the label from the weak oracle available.

4.5 Usage on finite input space

It is important to understand that this algorithm assumes the input space $\mathcal{X}$ to be infinite. In almost all practical applications this is the case, but in lab settings where a limited data set is used this is not the case (as for the results in this report for examples). When the input space is not infinite the active learning algorithm will sooner or later be forced to choose points which the main classifier is certain about. This leads to a larger error budget through equation (4.2). At the same time the new data points are added far from the decision boundary, and more errors will be allowed close to the decision boundary. The consequence of this is that the performance of the main classifier will decrease. This can however easily be countered by at each iteration monitoring the cross entropy of the main classifier on the old validation set $\mathcal{S}_{\text{val}}$ and the new examples in the validation set $Q_{\text{val},t}$. When the cross entropy is lower on the latter we should start to increase the desired share of labels correct in the training data, s_c, instead of asking for new data points.

5
Method

In this chapter we will outline the experimental setup used to evaluate the algorithm proposed in chapter 4.

5.1 Data set and preprocessing

To do the experiment we needed a labeled data set. AG's corpus over news articles was used[1]. The data set contains 496,835 news articles. Each news article contains a title, a description, a category, a source and some other information. We used four of these categories, *Business* was the category that our classifier should be able to find, i.e. the class with positive label. The categories *World*, *Sports* and *Sci/Tech* acted as negative examples. To do the prediction the classifier was given the information in the description field, cases for which that field was empty were discarded. Statistics for these four categories can be seen in table 5.1. Summarizing the data set consisted of 239,014 examples of which 54,432 or 22.8% were positive. It was important to have such a large data set. With a smaller data set it would have been hard to discern the difference between number of queries made to the strong oracle for the different algorithms.

Category	Label	No. of examples
Business	+1	54,432
World	−1	81,299
Sports	−1	62,151
Sci/Tech	−1	41,132

Table 5.1: Categories used from AG's corpus.

For preprocessing BeautifulSoup was used to remove html tags that some of the descriptions contained. All characters were put in lower case. Word tokenization was made with the toolkit NLTK [2]. Finally all descriptions that were longer than 60 tokens were truncated to that length.

[1]The data set was downloaded from `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

5.2 Paremeter settings for convolutional neural network

As main classifier the architecture described in section 2 was implemented. Parameters were chosen to be as in [4] since they worked well. The word-embedding dimension was set to $k = 128$, and initialized by sampling a uniform random variable on the range $(-0.1, 0.1)$. Filter sizes $h = 3, 4$ and 5 with 100 different filters each were used. For regularization we set the dropout probability to $P_d = 0.5$ and the l_2-regularization parameter to $\lambda = 3$. A mini-batch size of 50 was used. The performance of the classifier was evaluated against the validation set once every 50 iterations. Whenever the cross entropy on the validation set had not decreased for 5 consecutive evaluations the training was interrupted, and the model with lowest cross entropy for the validation set was saved.

5.3 Active learning method

Since there is a diminishing return in how much valuable information each example contains when doing active learning it is more important with small batches early on in the learning procedure. We also wanted to avoid using too large batches late in the learning procedure due to time-constraints of how long time an experiment reasonably could take. How many examples to include in a batch was calculated as:

$$\max(1000, 0.75 \cdot (|\mathcal{S}_{\text{train}}| + |\mathcal{S}_{\text{val}}|), \tag{5.1}$$

where $|\mathcal{S}_{\text{train}}| + |\mathcal{S}_{\text{val}}|$ is the size of the training and validation set at that point. In the case of bi-oracle learning the number of queries to the weak oracle was used when evaluating $|\mathcal{S}_{\text{train}}| + |\mathcal{S}_{\text{val}}|$. The active learning schedule described here was, besides a separate algorithm, also used in the bi-oracle learning algorithm.

To save time the neural network was trained from the point where it had been stopped with early stopping on the latest batch. It then starts training from a point where it ended up when training with less information than currently available. A risk with this that early training procedures may end up in a local minimum which will be difficult to escape later. To avoid this we, with 25% probability, retrained it completely from scratch after each training procedure. When a complete retraining had been done we chose the model which had lowest cross entropy on the validation set.

5.4 Bi-oracle learning algorithm

When constructing the algorithm as described in section 4 many different parameter settings were tried out in the development process. For the results presented in this thesis if not specified explicitly the settings as described in this paragraph were used. As cost function the most simple form of the cross entropy, as defined in equation (1.3), was used:

$$\mathrm{L}_{\mathcal{S}}(f_{\boldsymbol{w}}) = -\frac{1}{N} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}} \log f_{\boldsymbol{w}}(y_i | \boldsymbol{x}_i).$$

As described in section 4.5 we increased the parameter s_c as the input space $\mathcal{X}$ was depleted of difficult examples. Three different steps of increasingly high value of s_c were chosen as 0.95, 0.99 and 1 (in practice most of the times the algorithm took the step from 0.95 to 0.99 and from 0.99 to 1 directly after one-another).

Next we describe how the convolutional neural network was configured to work as agreement classifier. After that how the synthetic weak oracles that were investigated in this report were designed.

5.4.1 Design of agreement classifier

The agreement classifier used largely the same design and parameters as the main classifier, but with one small but important tweak. Appended to the output from the max-pooling layer was the weak label y_W. Including this value makes the algorithm able to take advantage of differing precision of the weak oracle between the subclasses. The weights connecting this value to the output neurons were also exempted from the l_2-regularization. The reason behind this is that we want the classifier to learn the precision of the weak oracle for the different classes completely and are not worried about overfitting to the weak label.

Some of the experiments performed are designed so that it in the data exists a high imbalance between the number of examples for which the strong and weak oracles agree or disagree. Therefore we used the modified cost function from equation (2.1) when training the agreement classifier.

5.4.2 Random expert weak oracle

As discussed in section 4.3.2 a first hypothesis was that the algorithm would perform well in conjunction with a weak oracle that has high precision on one class. In this case the value of the label from the weak oracle y_W would encode information valuable for the agreement classifier to learn which examples the weak and strong oracle often agree upon. The easiest imaginable weak oracle of this type is one that knows the correct label for some random share of the data. A weak oracle like this was constructed, we call it for *random expert weak oracle* (REWO). For the examples it did not know the correct label it always answered with the negative label -1. This lead to that it had a precision of 1.0 on the label $+1$. The recall on label $+1$ could be controlled.

Whether this weak oracle knows the correct label or not is completely independent of the example $\boldsymbol{x}$. It has an equally large probability of knowing the correct label close to the decision boundary as far from the decision boundary. This allows the algorithm to leverage the weak oracle both early on and late in the active learning process. However since it only got valuable information through y_W it also probably made the task a bit more difficult for the agreement classifier used in this report since its architecture is not constructed with this structure of the data in mind.

Weak oracle	Information from $\boldsymbol{x}$	Information from y_W
REWO		✓
CEWO	✓	✓
CEWOr	✓	

Table 5.2: Overview over the three types of weak oracles studied and where the agreement classifier will get the information needed to learn when used with them.

5.4.3 Category expert weak oracle

The next step is to include the information provided in the example $\boldsymbol{x}$ as motivated in section 4.3.1. To do this a another weak oracle was constructed, which we call *category expert weak oracle* (CEWO). Instead of knowing the correct label for some random share of the data this weak oracle knows the correct label for each example $\boldsymbol{x}$ that contains one of a specified set of words, $\mathcal{W}_{\text{correct}}$. If the sentence does not contain any of those words the weak oracle always provides the negative label -1.

Theoretically this weak oracle should provide more information to the agreement classifier than the category expert weak oracle. If one of the words in $\mathcal{W}_{\text{correct}}$ is included in $\boldsymbol{x}$ the agreement classifier should be able to predict agreement in both the cases when $y_W = +1$ and when $y_W = -1$. This in contrast with the random expert weak oracle for which the agreement classifier only is promised agreement when $y_W = +1$. Furthermore the architecture of the agreement classifier used is constructed with this structure of the data in mind, where the information is contained in $\boldsymbol{x}$. It is however not designed for data where the information is contained in a single word in $\boldsymbol{x}$, but rather in groups of words through the use of convolutional windows. But since the algorithm turned out to work well even when the window sizes were 3, 4 and 5 as described in section 5.2 this was not changed. $\mathcal{W}_{\text{correct}}$ contained business related words and thus $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1) > p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = -1)$.

We will also introduce a variant of this weak oracle, one which when an example does not contain any of the words specified in $\mathcal{W}_{\text{correct}}$ assigns a random label. The probability of a specific label being assigned was equal to the share of that class in the data. We call this weak oracle for CEWOr, where r stands for random. When the agreement classifier is used with this type of weak oracle the relevant information will be contained in $\boldsymbol{x}$, and the information from y_W will not be very useful as the precision on the two classes will be low.

In table 5.2 we see an overview over the three types of weak oracles introduced which will be used to generate results. The CEWO weak oracle should be the most useful one, since it provides information both in the form of $\boldsymbol{x}$ and y_W, whereas the other two only provides information through one of those two variables.

For all three weak oracle that have been introduced we have assumed perfect labeling on some subset of the data. Whenever the REWO proposed the label $y_W = +1$ the true label was also always $+1$. And when an example $\boldsymbol{x}$ contained a word from $\mathcal{W}_{\text{correct}}$ the CEWO and CEWOr always supplied the correct label. The assumption that given some condition on the example $\boldsymbol{x}$ and the weak label y_w one can be sure that the answer always is correct is very strong. To test what happens when this assumption is loosened the CEWO was modified so that one could choose the probability of it responding with the correct label given that $\boldsymbol{x}$ contained some

word from $\mathcal{W}_{\text{correct}}$.

5.5 Experimental design and visualization

Most experiments described in the results have been generated using 10-fold cross-validation. 90% of the data for each fold were used as the input space $\mathcal{X}$ in the algorithm. The algorithms had this $\mathcal{X}$ available to sample from in the active learning process. The algorithms had to subdivide the chosen examples into both a training set and a validation set and assign them labels according to the algorithm. The test set, $\mathcal{S}_{\text{test}}$, which consisted of 10% of the data for each fold, was used for generating results presented in section 6. Besides for generating results for later inspection the algorithm never used any information from $\mathcal{S}_{\text{test}}$. As validation set we for all algorithms randomly picked 10% of the data chosen in each active learning batch (i.e. $s_v = 0.1$ in algorithm 2).

Since the goal as stated in the beginning of the thesis is to reach a good performance on some metric with as few queries to the strong oracle as possible the results are most easily visualized as a graph with number of queries to the strong oracle on the x-axis and performance for the chosen metric on the y-axis. To do these types of visualizations the relevant metrics were evaluated after each batch and subsequent training of the classifiers. Furthermore these values were calculated as mean values over the 10-fold cross-validation. Standard errors of the mean were also calculated and are in the figures in section 6 visualized as dashed lines at a distance of one standard error of the mean from the main line. For the bi-oracle learning algorithm the amount of queries to the strong oracle made after a specific batch varies between different runs since it is dependent on how much the agreement classifier has learned up until that point. This becomes a problem when calculating the averages since we instead of getting multiple values on the y-axis for a specific value on the x-axis will get both different values for the y- and x-axis. To overcome this we did linear interpolation between each two consecutive batches for a specific run. For each value on the x-axis we could now calculate the mean and standard error over all the runs even though all of them did not have a value measured at that specific point.

6

Results

This chapter contains the results generated with the experimental setup described in chapter 5. The first section presents results from standard active learning. In the second section metrics from a single run with the bi-oracle learning algorithm is presented and analyzed. The remaining sections compare the performance of the bi-oracle learning algorithm for some different specifications of the weak oracle.

6.1 Active learning

In figure 6.1 we can see how the cross entropy and accuracy changes for our classifier when we either use active learning to choose new examples to label or when we choose new examples randomly. The black line labeled *All at once* is there for reference and shows the value reached when the classifier immediately was trained on all training examples. The figure has been generated by doing a cross-validation of ten runs for each method, the dashed lines are placed at one standard error of the mean away from the main line.

As we can see it is more efficient to do active learning than to just randomly choose new samples to label. The difference is more notable for accuracy than it is for cross entropy. We can also note that the active learning algorithm actually reaches a better final value than what was reached when we trained with all examples at once or when the algorithm chose new examples randomly. This is counter intuitive since the algorithms at this stage should have exactly the same training data available. The explanation behind this result lies in the experimental design. In all cases we use early stopping on a separate validation set. Since the two algorithms that are continuously fed with more and more data restarts training from the up until that best point after each new batch of data they have more chances of escaping local minima. Therefore they can be expected to reach better performance. But why does then the active learning algorithm and the algorithm that chooses examples randomly differ so much in the end? The reason behind this is probably that the algorithm that randomly chooses data will get many relevant examples late in the training procedure. Consequently it will not have as many chances to train with all the relevant data as the active learning algorithm, which early on will have most of the important examples available.

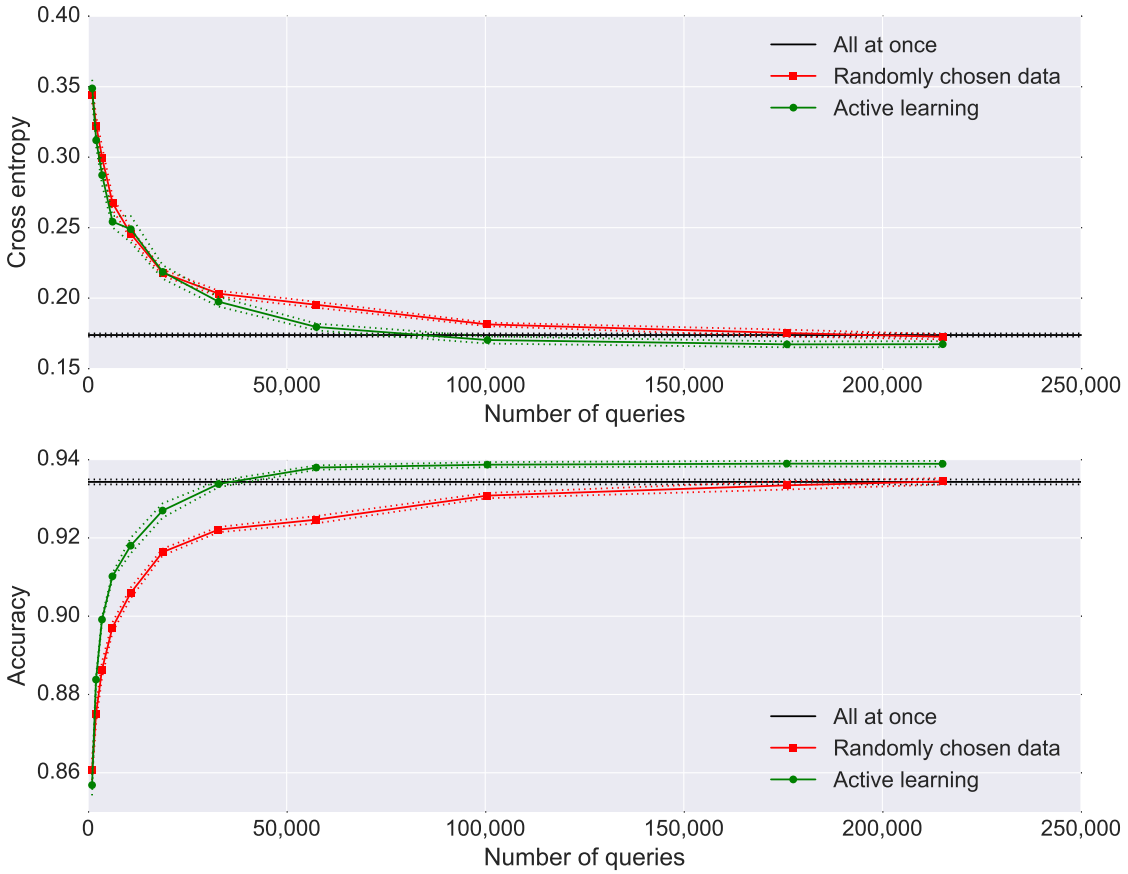


Figure 6.1: Accuracy and Cross entropy after different amount of queried examples for both an algorithm that chooses examples at random and examples chosen by active learning. The black line is there for reference. Dashed lines are one standard error of the mean away from the main line.

6.2 Bi-oracle learning algorithm

In figure 6.2 we can see some important metrics for one run with the bi-oracle learning algorithm. This particular run was performed with a random expert weak oracle as described in section 5.4.2 with precision of 1.0 and recall of 0.75 on class $y = +1$. The x-axis in both figures describe how many queries to the strong oracle the algorithm has performed up until that point. In the upper figure we see how the cross entropy changes during the run for three different data sets: the test set, the validation set (which the algorithm continously builds up through active learning), and the latest batch added to this validation set. In the lower figure we see two different types of values: three different measures of the share correctly labeled examples s_c, and also the trust threshold, T.

We can in the figure see some interesting signs of how the algorithm works. The first thing one notices is that the data points arrive with longer and longer distance between them. The reason behind this is that the batch size of queries to the weak oracle is decided by the growing function in equation (5.1). What we measure on the x-axis is the number of queries to the strong oracle, since that is somewhat

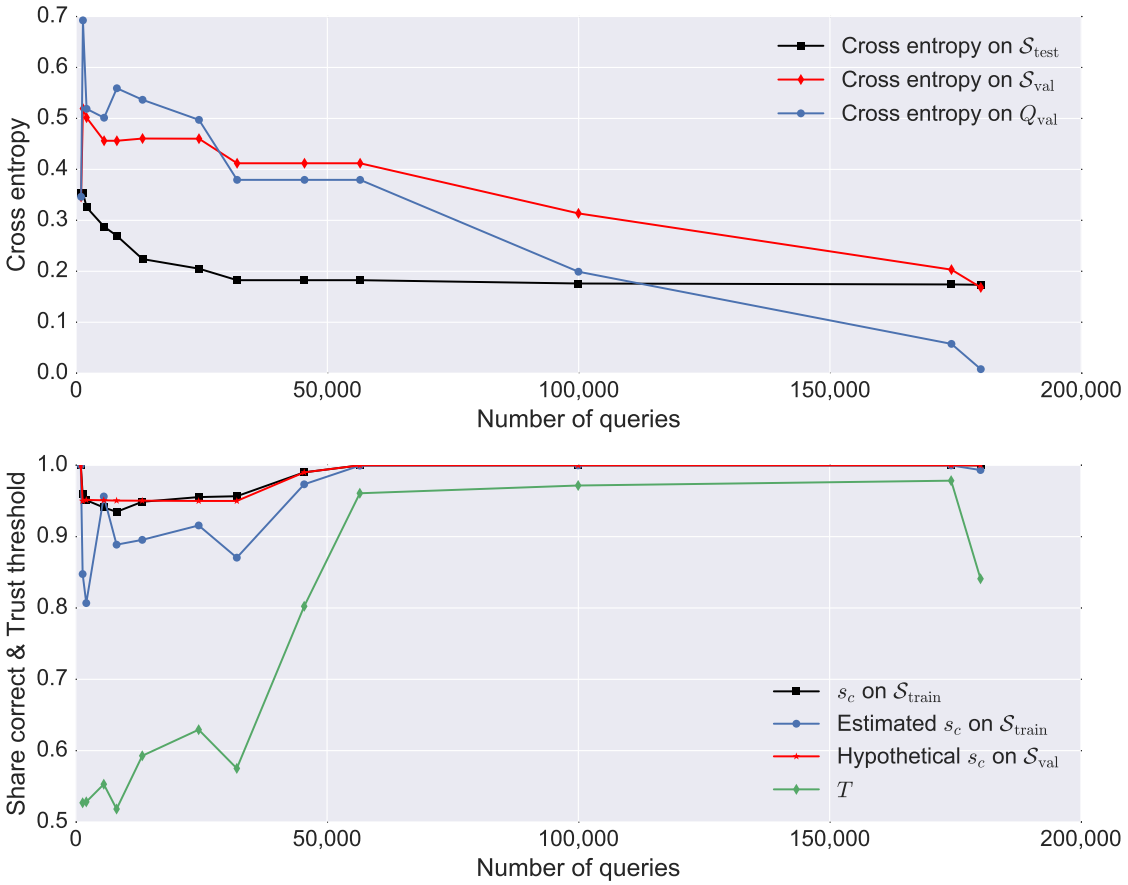


Figure 6.2: Relevant metrics for the execution and evaluation of the bi-oracle learning algorithm for one run with a weak oracle as described in section 5.4.2 with precision of 1.0 and recall 0.75 for class $y = +1$. The upper figure shows the cross entropy on the test set, the validation set and the set of new examples to the validation set. The lower figure shows the share correctly labeled examples on the training set, an estimation of that, and that share on a hypothetical training set formed by the validation set. It also shows the trust threshold, T, of the algorithm.

correlated with the number of queries to the weak oracle it will also be growing. This is not always true since we do not expand our data set with new examples $\boldsymbol{x}$ when increasing s_c.

As described in sections 4.5 and 5.4 the algorithm will increase the desired s_c whenever we measure a better cross entropy on Q_{val} (the new data points in the validation set) than on the entire validation set $\mathcal{S}_{\text{val}}$. This happens when the only examples left in $\mathcal{X}$ are more easy to classify than the ones we have already chosen. We can see this occurring in the upper figure after around 30,000 queries. The two following steps we increase the desired s_c which we see in the lower figure, the move along the x-axis for these steps is only due to the algorithm querying the strong oracle for some points in $\mathcal{S}_{\text{train}}$, no new examples $\boldsymbol{x}$ are added.

In the lower part of figure 6.2 we see three curves for s_c. The curve labeled s_c *on* $\mathcal{S}_{train}$ describes how large share of the labels y in our training data set that are correct. This value would not be available in a real-world use case. To calculate

it we need all the true labels for all examples in $\mathcal{S}_{\text{train}}$, and the purpose of this algorithm is avoid needing them all. However it is this value that we wish to control somehow. As described in section 4.2 the approach to control it used here is by in reality controlling s_c on a hypothetical training set created with the validation set. The value of this is visualized with the curve *Hypothetical* s_c *on* $\mathcal{S}_{val}$. By comparing this curve with s_c *on* $\mathcal{S}_{train}$ we see that this works well in practice, the two curves are close to each other. An alternative solution is to assume that $f_{\boldsymbol{\theta}}^{\text{ag}}$ is a good approximation of $p(y_W = y|\boldsymbol{x}, y_w)$ and use equation (4.3). The value obtained by this approximation is showed as the curve *Estimated* s_c *on* $\mathcal{S}_{train}$. As we can see in the figure this approximation does not seem to work well in practice. It for almost all time steps overestimates the number of errors that we have in $\mathcal{S}_{\text{train}}$, often by a factor of two.

We can in the lower part of figure 6.2 also observe the trust threshold, T, which is the value we control to maintain the hypothetical s_c on $\mathcal{S}_{\text{val}}$ at the desired value. All examples for which $f_{\boldsymbol{\theta}}^{ag}(\boldsymbol{x}, y_w) > T$ or $f_{\boldsymbol{\theta}}^{ag}(\boldsymbol{x}, y_w) < 1 - T$ the algorithm will not ask the strong oracle for the corresponding label. This value fluctuates heavily during the run of the algorithm, it starts close to 0.5, and then increases to finally reach a value closer to 1.0 (ignoring the final dip).

One interesting thing to note when looking at how the cross entropy changes over time for the three different data sets is how the improvement in them differs. The cross entropy on the test set $\mathcal{S}_{\text{test}}$ decreases much more rapidly than on $\mathcal{S}_{\text{val}}$. The reason behind this is that the active learning procedure deliberately chooses as difficult examples as possible from $\mathcal{X}$ when building the training and validation set. Therefore, the cross entropy for most parts of the run will be worse on the validation set $\mathcal{S}_{\text{val}}$. In the end when all points in $\mathcal{X}$ have been queried the distribution of points in $\mathcal{S}_{\text{val}}$ will be identical to the one in $\mathcal{S}_{\text{test}}$ and the cross entropy for those two sets should converge to the same value, which we also see happen.

6.3 Random expert weak oracle

$p(y_W = y\|y = +1)$	0.5	0.75	0.95
Recall $y = +1$	0.5	0.75	0.95
Precision $y = +1$	1	1	1
Recall $y = -1$	1	1	1
Precision $y = -1$	0.87	0.93	0.99
Accuracy	0.9	0.95	0.99

Table 6.1: Recall, precision on the two classes and accuracy for the weak oracle used to generate the results in figure 6.3.

Now we turn to the results achieved by using a random expert weak oracle as described in section 5.4.2. This type of weak oracle knows the label for a random subset of the examples, for the ones it does not know the label it will always answer with the label -1. Thus it has a precision of 1.0 for the label $+1$, and a recall of

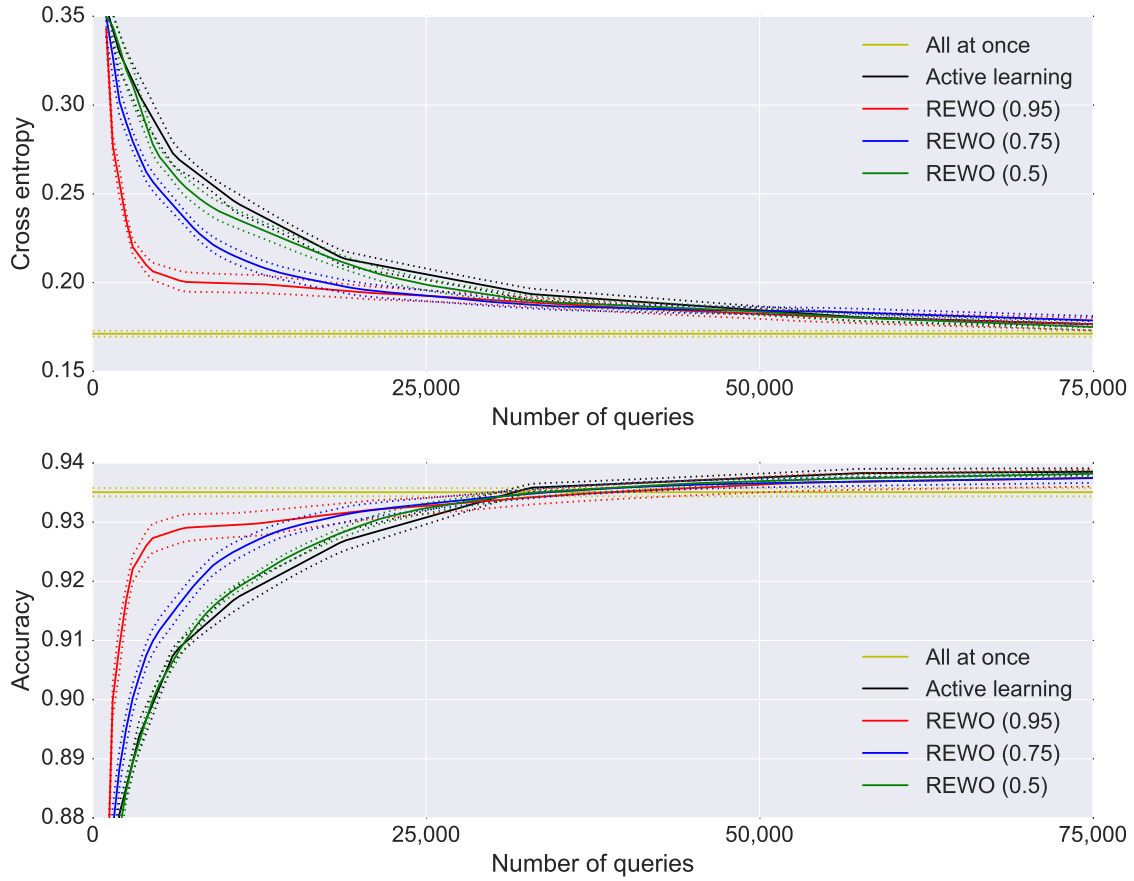


Figure 6.3: Accuracy and cross entropy after different amount of queried examples to the strong oracle for both the standard active learning algorithm and the bi-oracle learning algorithm used with a REWO. The number in parenthesis is equal to $p(y_W = y|y = +1)$ for that weak oracle.

1.0 for label -1. This weak oracle was used with three different levels of recall for label $+1$. In table 6.1 we can see other relevant metrics for the weak oracle.

In figure 6.3 we can see the performance of the bi-oracle learner for these weak oracles compared with the active learner. We can clearly see that the higher recall the weak oracle has the better the algorithm performs, which is intuitive since the weak oracle in that case has knowledge about more examples. Furthermore we see that the gains are most significant in the beginning. The bi-oracle learning algorithm differs most from the active learning algorithm early on, later the performance of the two algorithms converge more and more. This is reasonable since when the algorithm chooses to only ask the weak oracle some errors will be introduced in the training data set. To achieve the last performance all errors need to be forced out of the training set which requires many queries to the strong oracle and hinders us from being able to leverage the weak oracle any more.

In the case where the recall is 0.95 the increase in performance is extremely rapid. The reason behind this is that a recall this high leads to an almost perfect classifier as we can see in table 6.1. By analyzing the numbers behind this graph we found out that almost all queries made to the strong oracle up until the point where

the cross entropy flattens out are made up of examples in the validation set. Thus the bi-oracle learning algorithm in this case feels so certain about the performance of the weak oracle that it does not ask the strong oracle for almost any examples in the training set.

For the two other weak oracles, they also have high precision on class $y = -1$ and high accuracy as we can see in table 6.1. But for these weak oracles the precision on class $y = -1$ is less than 0.95, which is the share correctly labeled examples, s_c, that we force our training set to have. Therefore the agreement classifier has to learn a more difficult decision boundary in this case.

6.4 Category expert weak oracle

$p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}} \mid y = +1)$	0.3	0.5
Recall $y = +1$	0.3	0.5
Precision $y = +1$	1	1
Recall $y = -1$	1	1
Precision $y = -1$	0.83	0.87
Accuracy	0.87	0.9

Table 6.2: Recall, precision on the two classes and accuracy for the weak oracle used to generate the results in figure 6.4.

The second type of weak oracle that has been investigated in this thesis was introduced in section 5.4.3. This weak oracle has perfect knowledge of the labels y in some subspace of $\mathcal{X}$ defined as those examples that contain any of a given set of words $\mathcal{W}_{\text{correct}}$. For the examples which does not contain any of those words it will always answer with label -1. As for the previous described weak oracle the precision on class $+1$ will thus be 1.0 and the recall on class -1 will also be 1.0. In table 6.2 we can see some metrics for this weak oracle for the two set of words $\mathcal{W}_{\text{correct}}$ that were used in this thesis. The words in $\mathcal{W}_{\text{correct}}$ were chosen carefully to achieve a weak oracle with similar classification properties as for the random expert weak oracle used in section 6.3, to facilitate comparison between the two.

In figure 6.4 we can see the results obtained for the bi-oracle learning algorithm with the category expert weak oracle. For both the weak oracle with a recall of 0.3 and 0.5 on the negative examples we can see that they perform better than when doing standard active learning. With higher recall the algorithm was able to do less queries to the strong oracle to reach a given performance. As for the previous weak oracle we can in this experiment as well also notice that the effect decreases as we allow for more and more queries to the strong oracle.

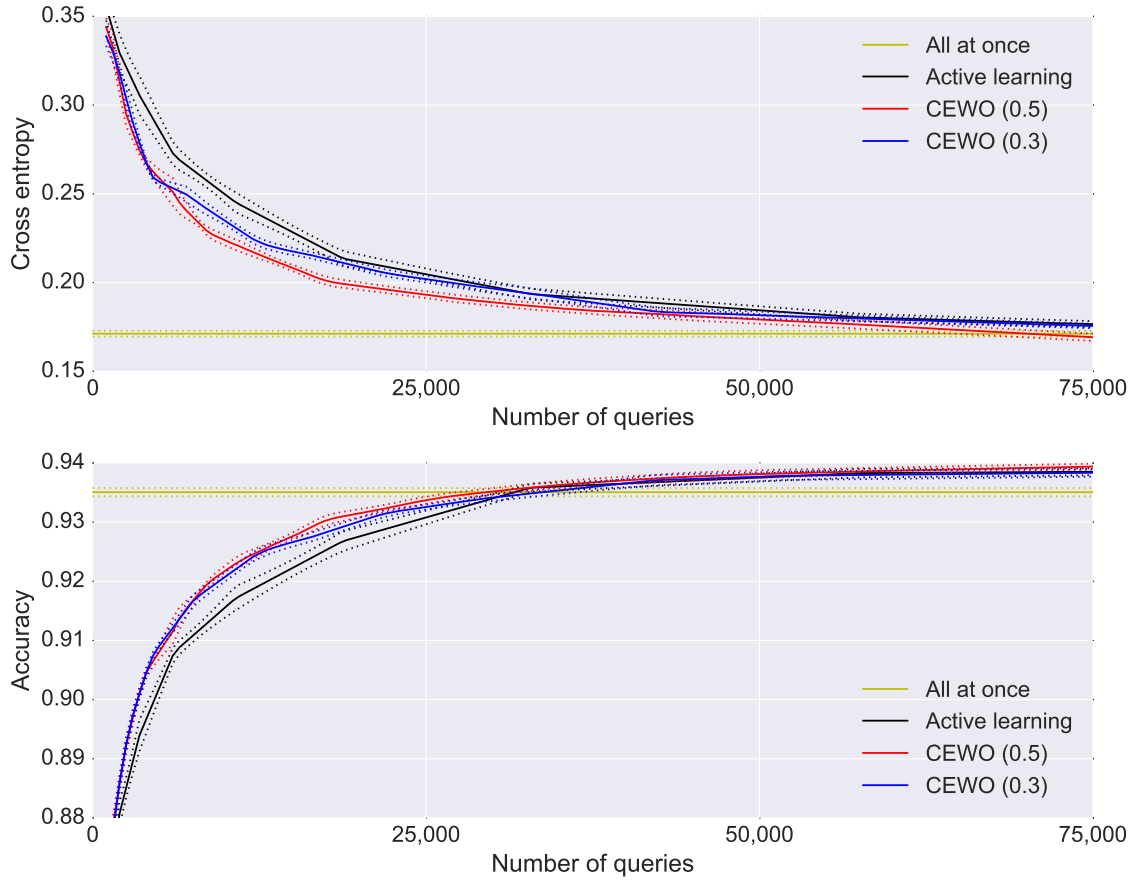


Figure 6.4: Accuracy and cross entropy after different amount of queried examples to the strong oracle for both the standard active learning algorithm and the bi-oracle learning algorithm used with a CEWO. The number in parenthesis is equal to $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1)$ for that weak oracle.

$p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}\|y = +1)$	0.3	0.5
Recall $y = +1$	0.46	0.61
Precision $y = +1$	0.38	0.49
Recall $y = -1$	0.78	0.81
Precision $y = -1$	0.83	0.88
Accuracy	0.73	0.79

Table 6.3: Recall, and precision on the two classes and accuracy for the CEWOr used to generate the results in figure 6.5.

6.5 Category expert weak oracle with randomly assigned unknown labels

A variant of the weak oracle presented in section 5.4.3 is the category expert weak oracle with randomly assigned unknown labels (CEWOr). This weak oracle also knows exactly how to assign labels when an example contains any one of a set of words $\mathcal{W}_{\text{correct}}$. However, when an example does not contain any of those words it

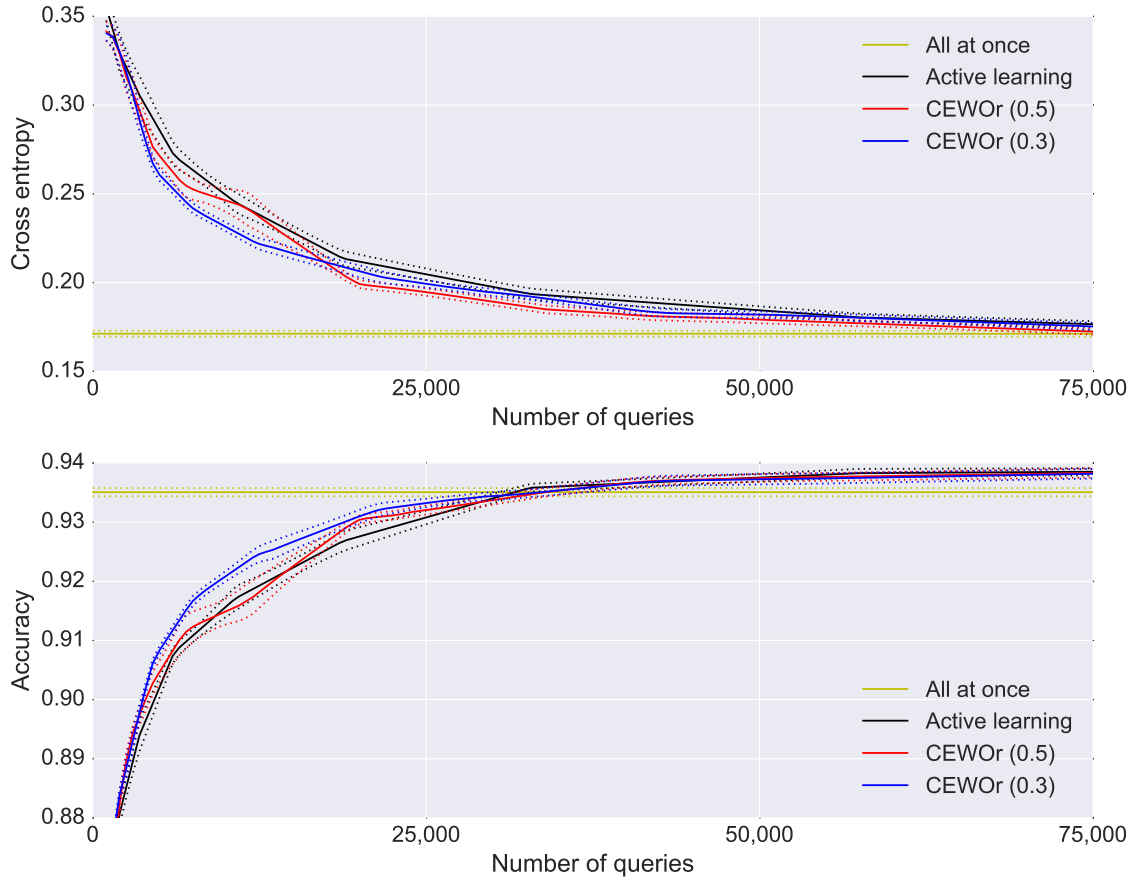


Figure 6.5: Accuracy and cross entropy after different amount of queried examples to the strong oracle for both the standard active learning algorithm and the bi-oracle learning algorithm used with a CEWOr. The number in parenthesis is equal to $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1)$ for that weak oracle.

will assign the label randomly instead of always giving it a negative label. Thus it will not have equally high precision as was the example for CEWO and REWO. In table 6.3 we see some relevant metrics for the performance of this weak oracle. We can see a big difference in performance between this weak oracle and the two other weak oracles used. This weak oracle has much worse performance on all metrics compared with the other types of weak oracles.

In figure 6.5 the results generated for this types of weak oracles are displayed. We see that it is clearly possible for the bi-oracle learning algorithm to save on queries to the strong oracle when used for this type of weak oracle. We can also see that the weak oracle with $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1) = 0.5$ actually performed worse in average than the active learning algorithm for parts of the curve. However at the same place the standard error widens considerably so this is probably due to one or two of the ten cross-validation runs taking an improbable path. That the agreement classifier with the weak oracle $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1) = 0.5$ should be worse than with $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1) = 0.3$ is not a possible explanation since $\mathcal{W}_{\text{correct}}$ for the latter contains a subset of words of $\mathcal{W}_{\text{correct}}$ for the former.

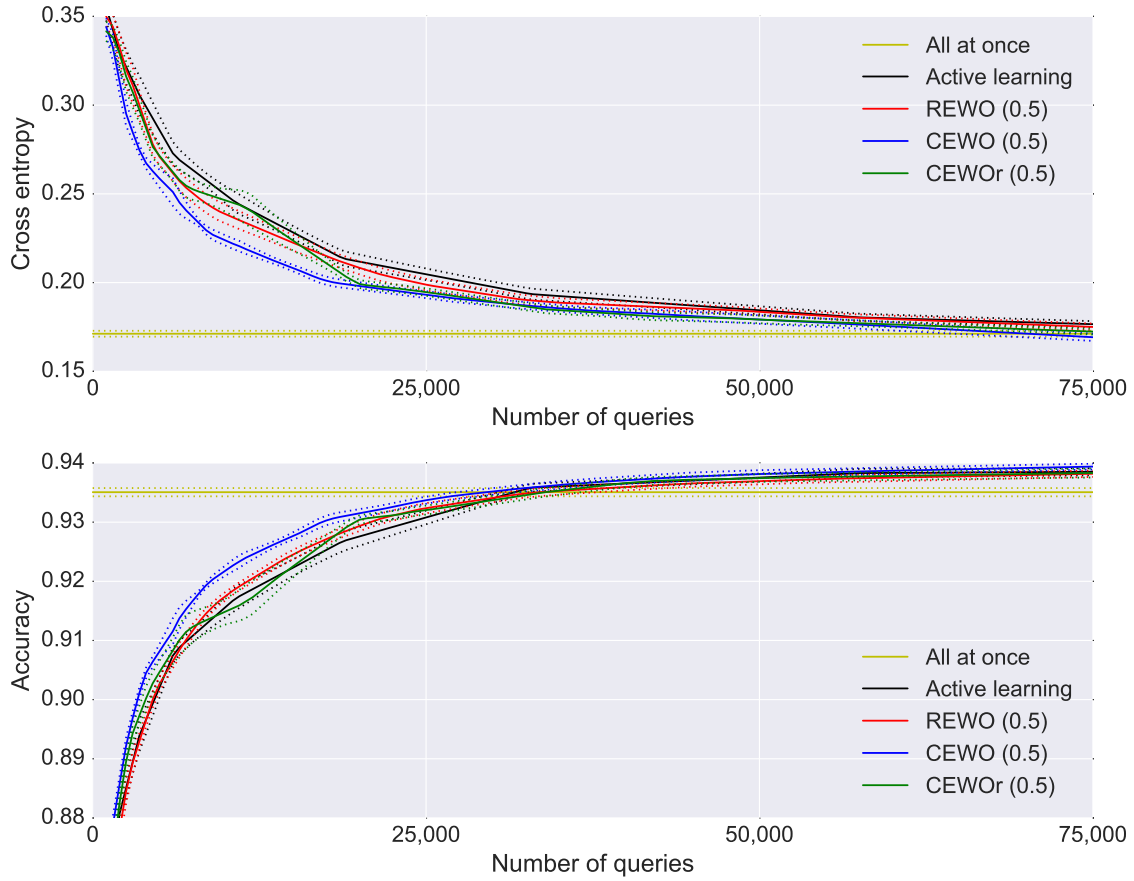


Figure 6.6: Comparison between three runs with the bi-oracle learning algorithm with different types of weak oracles. The meaning of the value in the parenthesis differs, but are as in figures 6.3, 6.4 and 6.5.

6.6 Comparing the different weak oracles

The results discussed in the previous three sections are generated on the same cross validation splits to facilitate comparison. In figure 6.6 we plot a comparison between bi-oracle algorithm running with a REWO, CEWO or CEWOr. They have all been configured so that 50% of the examples belonging to class +1 should be easy for the agreement classifier to learn.

Comparing the weak oracles we see that the CEWO performs best, which should not be surprising since it obtains information through both $\boldsymbol{x}$ and y_W as we can see in table 5.2. For most parts of the curve it seems like CEWOr outperforms REWO, even though it is a bit difficult to draw any clear conclusions since the CEWOr curve has large standard error of the mean around 10,000 queries to the strong oracle.

6.7 Lowering precision of weak oracle

In this section we investigate what happens when we loosen the assumption that the category expert weak oracle always will provide the correct label given that $\boldsymbol{x}$

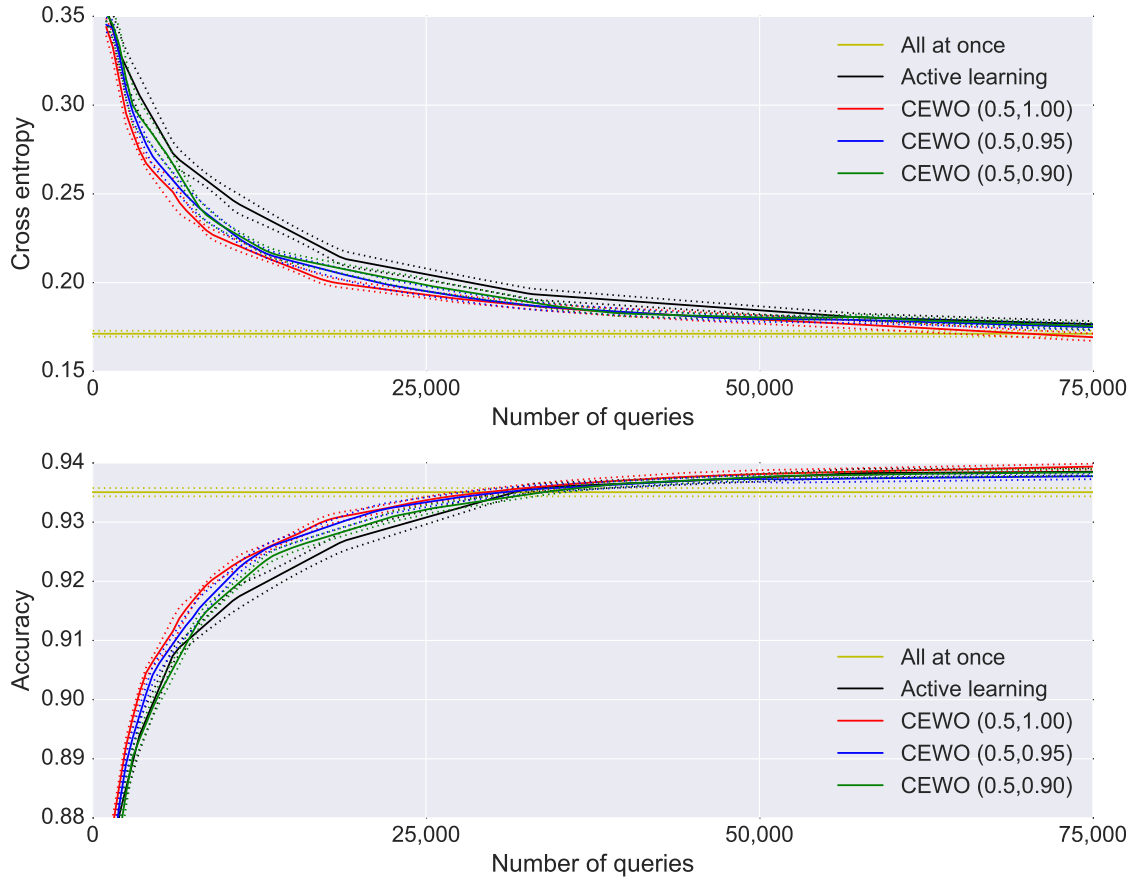


Figure 6.7: Comparison between three runs with the bi-oracle learner algorithm with different CEWO. The first value in the parenthesis equals $p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}}|y = +1)$ and the second value equals $p(y_W = y|\boldsymbol{x} \in \mathcal{W}_{\text{correct}})$.

contains any of the words in $\mathcal{W}_{\text{correct}}$. We thus instead let the probability $p(y_W = y|\boldsymbol{x} \in \mathcal{W}_{\text{correct}})$ vary. We let this value be 1.0, 0.95 and 0.9. How this changes the performance can be seen in table 6.4. The reason behind that the precision on $y = +1$ decrease more rapidly than $p(y_W = y|\boldsymbol{x} \in \mathcal{W}_{\text{correct}})$ should be the class imbalance. Even though the words in $\mathcal{W}_{\text{correct}}$ are chosen to be such that they are more frequent in the positive class there are much more negative examples in the data set.

In figure 6.7 we plot the performance of the algorithm used with the CEWO with settings as in table 6.4. We can see a visible decrease in performance even as $p(y_W = y|\boldsymbol{x} \in \mathcal{W}_{\text{correct}})$ only decreases slightly. The probability of a label being correct given that the example is in the weak oracles expert region seems to influence the performance of the algorithm much.

6.8 Cost function with soft labels

In section 4.4 an alternative cost function was introduced. It uses the prediction made by the agreement classifier to supply soft labels on the examples for which

$p(\boldsymbol{x} \in \mathcal{W}_{\text{correct}} \mid y = +1)$	0.5	0.5	0.5
$p(y_W = y \mid \boldsymbol{x} \in \mathcal{W}_{\text{correct}})$	0.9	0.95	1.0
Recall $y = +1$	0.45	0.47	0.5
Precision $y = +1$	0.89	0.94	1
Recall $y = -1$	0.98	0.99	1
Precision $y = -1$	0.86	0.86	0.87
Accuracy	0.87	0.88	0.9

Table 6.4: Recall, precision on the two classes and accuracy for the category expert weak oracle used to generate the results in figure 6.7.

only a weak label exists. To test this cost function results were generated using a category expert weak oracle with $\mathcal{W}_{\text{correct}}$ including 50% of the examples in class $+1$. This is the same weak oracle that was used in section 6.4. In figure 6.8 we can see how the algorithm behaves for this cost function compared with the standard cost function with only hard labels. The soft label cost function was tested both using a fixed value of s_c, and thus varying T, as well as instead keeping T fixed to 0.7. We can see that using the soft label cost function in both cases leads to similar performance as the hard label cost function.

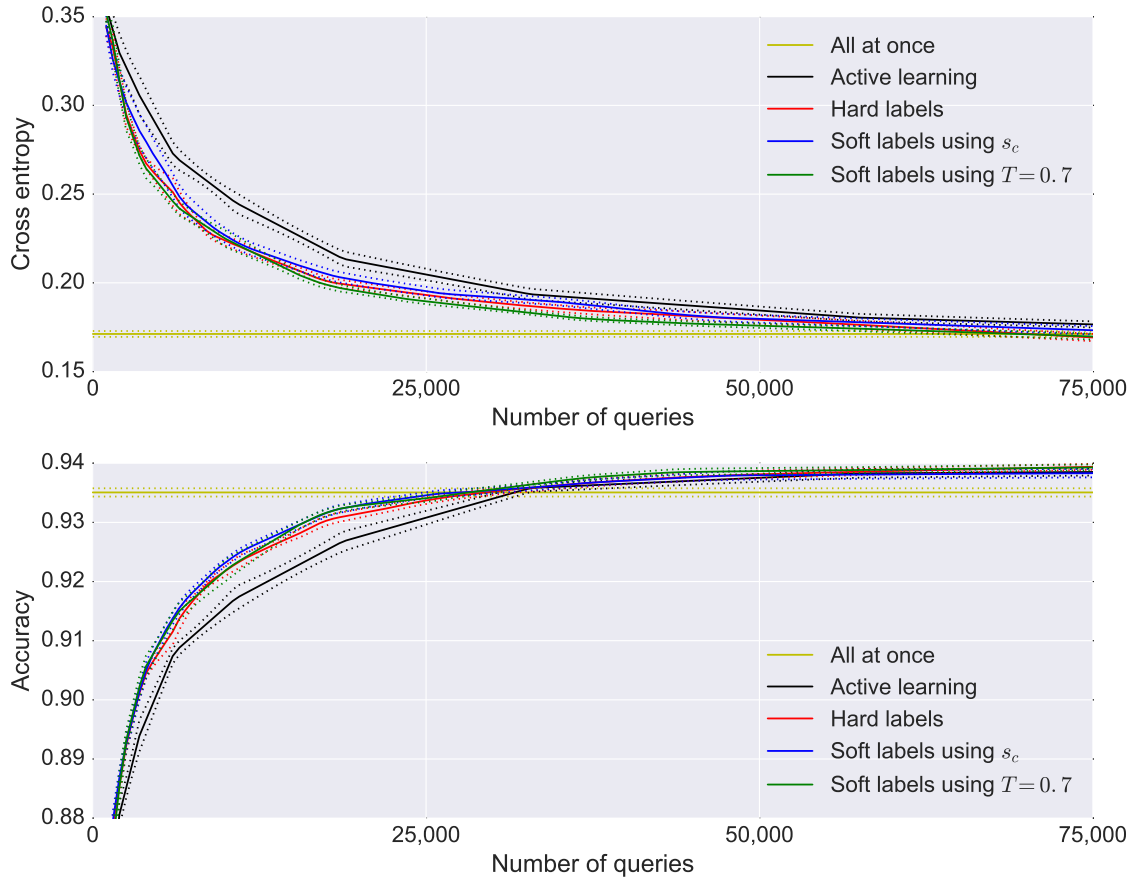


Figure 6.8: Comparison of the algorithm with different cost functions. *Hard labels* means that the cost function in equation (1.3) was used, whereas for the two other bi-oracle learning algorithm runs the cost function defined in equation (4.5) was used. For *Soft labels using* s_c, the algorithm was apart from the cost function unchanged, for *Soft labels using* $T = 0.7$ a fixed trust threshold was used throughout the run.

7
Discussion

In this chapter the implications of the results presented in chapter 6 will be discussed in a larger scope. First, the key features of our algorithm will be discussed and motivated. The last two sections will discuss how it performs for the different synthetical weak oracles and with what real-world weak oracles it can be expected to work.

7.1 Structure of the algorithm

In the proposed algorithm a central parameter is the trust threshold, T. T decides for what values of the agreement classifier we should trust the weak oracle. A first approach for how to control the algorithm would be to set this parameter to some fixed value. While this idea might seem natural it did not work well in practice. In section 4.2 it was proposed that instead of using a fixed value of T we let it vary as more examples becomes available in the training set. What is kept fixed then instead is the share correctly labeled examples, s_c, in the training set, or at least a good approximation of that. By observing figure 6.2 we can see how much T varies during the run. If we instead would have kept T fixed one can imagine that s_c would vary significantly during the run. There would be a risk of s_c becoming dangerously low.

As described in section 4.2 there exists some different possibilities for how to control s_c on the training set. One possibility is to approximate $p(y = y_W)$ with $f^{ag}_{\boldsymbol{\theta}}$, another one is to let the validation set hypothetically play the role of the training set. In this algorithm we have used the second alternative since the first one did not work well in practice. The first one is believed to work poorly since the approximation did not hold up in practice which we can see in figure 6.2. When we use the second alternative we do not assume that agreement classifier can produce a good approximation of $p(y = y_W)$, but merely that it can order the examples after how much we should trust them. As we will see this is a much more reasonable assumption, and makes for a robust algorithm.

It is worth diving deeper into why $f^{ag}_{\boldsymbol{\theta}}$ has not been a good approximation of $p(y = y_W)$. The theory suggests that a neural network if trained with a cross entropy cost function is a good approximation of the probability of an example having a specific label [8]. However [8] finds that this approximation only holds if sufficiently many training examples exists. This is an assumption one cannot make in active learning! If we have sufficiently many training examples there would be no need for active learning. Furthermore when estimating s_c we sum over the approximations

(see equation (4.3)), this magnifies potential errors in the approximation of $p(y_W = y)$. If the estimation of $p(y_W = y)$ is biased to low or high values, these errors will be reinforced. This explains the large difference in figure 6.2 between the estimated and real value of s_c on $\mathcal{S}_{\text{train}}$.

The proposed algorithm uses just one hyperparameter, which needs to be decided when using it. This is the desired share correctly labeled examples in the training set, s_c. All the results in this report use an increasing value of s_c of 0.95, 0.99 and finally 1.0 as the difficult examples are depleted from $\mathcal{X}$ in accordance with section 4.5. Some experiments were made with different series of increasing values. Nevertheless, none are presented amongst the results since the necessity of having a series of increasing values of s_c originates from the fact that $\mathcal{X}$ is finite, and this is not something that would occur in a real-world application of the algorithm. When $\mathcal{X}$ is infinite the active learning process leads to that examples closer and closer to the decision boundary of the main task are added to the training set. Therefore the errors in the training set allowed by s_c ought to move closer and closer to the decision boundary as well. This leads to the conclusion that the algorithm should be robust to choices of s_c in real-world applications. It is important to point out that this is a speculative conclusion founded on intuition rather than results.

The proposed algorithm is heavily dependent on the availability of the validation set $\mathcal{S}_{\text{val}}$. The validation set is used at three different places in the algorithm: when training the agreement classifier, when deciding the T given a desired s_c, and when training the main classifier. The prominent role of the validation set in the proposed algorithm cannot be too emphasized.

In figure 6.8 we can see the soft label cost function (proposed in equation (4.5)) tested with both a fixed s_c and a fixed T. We cannot see any significant difference between the performance of the algorithm with these settings or with the soft label cost function. We can see that when we used the soft label cost function the algorithm worked even when we used a fixed T. This is interesting and is something that could be worth to investigate more in the future. From this we can conclude that the approximation $f^{ag}_{\boldsymbol{\theta}} \approx p(y = y_W)$ is not completely off. When we use the approximation as a part of the soft label cost function we do not sum over it, as we did when estimating s_c in equation (4.3). That might be the reason why the approximation works well when used in the soft label cost function.

7.2 Performance on different types of weak oracles

As discussed in section 4.3 the information fed to the agreement classifier can arrive through the example $\boldsymbol{x}$ and/or the weak label y_W. Our algorithm have been tested and worked on synthetic oracle that feed information through either one of those channels, or through both.

The classification the agreement classifier had to make in these three examples was easy, the decision boundaries were extremely simplified. Either it was just to recognize the label of y_W, or learning whether $\boldsymbol{x}$ contained a specific word or not, or a combination. What was not studied in this report is for how difficult

learning problems faced by the agreement classifier it is possible to successfully use the algorithm. Studying this is hard since it is difficult to construct realistic synthetic weak oracles. Nevertheless, by doing some reasoning we can come to some conclusions. There has to exist some subspace $\Omega \subset (\mathcal{X}, \mathcal{Y})$ where it is more easy for the agreement classifier to learn when the weak and strong oracle can be expected to agree than what it is for the main classifier to learn $p(y|\boldsymbol{x})$. If this is not the case the main classifier will learn its task more quickly than what the agreement classifier learns for which examples it can skip querying the strong oracle. Thus the active learning algorithm will stop adding new examples $\boldsymbol{x} \in \Omega$ to the data set before the agreement classifier learns which it can skip asking the strong oracle about. The problem the agreement classifier faces only has to be more easy on some subset $\Omega \subset (\mathcal{X}, \mathcal{Y})$, since it to sucesfully use the algorithm is enough to be able to avoid querying the strong oracle for some examples.

In section 6.7 it was studied how the performance of the algorithm decreased when changing the probability of the weak oracle supplying the correct label when $\boldsymbol{x} \in \mathcal{W}_{\text{correct}}$. We saw that even a slight decrease in performance of the weak oracle in its expert region made it more difficult for the algorithm to leverage the weak oracle. This is reasonable since if the agreement classifier cannot be certain enough about its predictions many errors will be introduced when only the weak oracle is asked and subsequently s_c will be reached early after few included weak labels. This leads us to the conclusion that the accuracy of the weak oracle on some subspace $\Omega \subset (\mathcal{X}, \mathcal{Y})$ has to be close to 1 (or 0) for the proposed algorithm to be able to avoid doing queries to the strong oracle on that subspace. It is not enough that the weak oracle performs quite well over the entire $(\mathcal{X}, \mathcal{Y})$. There has to exist subspaces on which its performance is very high.

The conclusions drawn in this section can be summarized as: to be able to leverage the weak oracle with this algorithm $p(y_W = y|\boldsymbol{x}, y_W)$ has to be close to 1 (or 0) on some region of $(\boldsymbol{x}, y_W) \in (\mathcal{X}, \mathcal{Y})$ which the agreement classifier can learn to identify faster than the main classifier learns $p(y|\boldsymbol{x})$ on the same region.

7.3 Outlook and real-world applications

This thesis have focused on binary classification of text. It is apparent that it should be possible to extend to other types of binary classification. It should also be possible to extend the algorithm to multiclass classification without large changes since the agreement classifier still could be trained as the probability of two labels agreeing. In this case however the prediction of two labels disagreeing would not add any particularly useful information, since the example then may belong to one of many other classes. Applying this method to regression would be more difficult. At least two possible formulations of the agreement classifier exists: one where it is trained to predict $\mathbb{E}[y_W - y]$ (it would in this case be a regression problem and not a classification problem), and another one where it is trained to predict $p(|y_W - y| < \epsilon)$ for some ϵ. Evaluating these, and other approaches is left to future work.

Another extension which is left to future work is how to modify the algorithm to allow for more than one weak oracle. A tricky part in doing so will probably be how to define s_c when multiple weak oracles exist; how many errors should each

weak oracle be allowed to introduce in to the training set?

A third extension which has not be considered at all in this thesis is the setting where there exist multiple oracles that have different performances, as well as different costs of querying. In this case it is hard to see how this algorithm could be applied. Instead of always asking the weak oracle we might for some examples want to directly query the strong oracle. Introducing this feature to the proposed algorithm would be quite unnatural. Creating an algorithm more similar to the one proposed by [11] might be a possibility. They use a separate training set for the agreement classifier (or disagreement classifier as they call it), and then only ask one of the oracles for each example when building the training set for the main classifier.

In section 7.2 we discussed what structure a weak has to have for it to be feasible to leverage the proposed algorithm. Now we will instead discuss two types of real-world settings which this algorithm suits well (based on the discussions in sections 4.3.1 and 4.3.2). One possible use case is when there is an already existing rule-based system which one hopes to improve by introducing a machine learning method. To do so one would have to build a new data set. If the rule-based system has a high performance on one class, or if the subspace for which it has high agreement with the strong oracle has an easy to learn decision boundary it should be possible to use the rule-based system as a weak oracle. Another use case is when there exists two categories of annotators, one very expensive expert, and another cheaper non-expert annotator (of negligible querying cost in comparison to the expert). They could then play the roles of strong and weak oracles respectively.

Independently of what active learning problem one faces there is a method that could help the researcher to get a feeling for whether it would be possible to leverage this algorithm or not. Early on in the process of building the data set one could just try to learn the classification problem $p(y = y_W|\boldsymbol{x}, y_W)$. By evaluating the performance of this classifier on some test set it should be possible to draw initial conclusions of the feasibility of this approach.

8

Conclusion

In this chapter the conclusions drawn in this thesis are summarized. More details concerning the conclusions and the justification behind them can be found in chapters 6 and 7.

- The proposed algorithm to do active learning in the presence of a strong and weak oracle has been shown to be working for training a neural network to do sentence classification with some different types of synthetic weak oracles.
- The algorithm uses an agreement classifier, which is trained to predict the probability of the weak oracle supplying the correct label, but when used it is only assumed to be able to sort the examples after how much trust we can put in them. That only this weak assumption is made about the agreement classifier is believed to be critical to the algorithm.
- Instead of conditioning the response of the agreement classifier only on $\boldsymbol{x}$ this thesis proposes that it also should be conditioned on y_W, which can contain useful information.
- The desired share correctly labeled examples in the training set, s_c, is the only hyperparameter that has to be set by the user when using the proposed algorithm.
- Experiments have successfully been performed using a soft label cost function. In this case when we only have the weak label we use the probability that the agreement classifier supplies when training the main classifier.
- Important for the use of the algorithm is the structure of the weak oracle. It should have high accuracy on some region of the input and/or output space, and the agreement classifier should be able to learn this region faster than what the main classifier can learn the label distribution on that region.
- This thesis has focused on binary sentence classification. It should be easy to extend the work to other types of classification problems. Extending it to regression problems is probably possible but more difficult.

Bibliography

[1] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 65–72, New York, NY, USA, 2006. ACM.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.

[3] M. Hasenjäger and H. Ritter. In Lakhmi C. Jain and Janusz Kacprzyk, editors, *New Learning Paradigms in Soft Computing*, chapter Active Learning in Neural Networks, pages 137–169. Physica-Verlag GmbH, Heidelberg, Germany, Germany, 2002.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[6] Christopher H Lin, Daniel S Weld, et al. To re (label), or not to re (label). In *Second AAAI conference on human computation and crowdsourcing*, 2014.

[7] Luigi Malago, Nicolo Cesa-Bianchi, and J Renders. Online active learning with strong and weak annotators. In *NIPS Workshop on Learning from the Wisdom of Crowds*, 2014.

[8] Michael D Richard and Richard P Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483, 1991.

[9] Eduardo D Sontag. Vc dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, 168:69–96, 1998.

[10] Ruth Urner, Shai Ben-David, and Ohad Shamir. Learning from weak teachers. In Neil D. Lawrence and Mark A. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, volume 22, pages 1252–1260, 2012.

[11] Chicheng Zhang and Kamalika Chaudhuri. Active learning from weak and strong labelers. *CoRR*, abs/1510.02847, 2015.