Technology
Arts Sciences
TH Köln

Approximating the optimal threshold for an abstaining classifier based on a reward function with regression

Bachelor Thesis

Jonas Fassbender

jonas@fassbender.dev
11117674

In the course of studies
Computer Science

For the degree of
Bachelor of Science

Technical University of Cologne
Faculty of Computer Science and Engineering

First supervisor: Prof. Dr. Heinrich Klocke
Technical University of Cologne

Second supervisor: Prof. Dr. Fotios Giannakopoulos
Technical University of Cologne

Overath, July 2019

Abstract

This thesis discusses how the abstention rule of an abstaining classifier can be determined based on a reward setting. Abstaining classifiers are part of reliable machine learning, which is applied in real world application domains where wrong predictions are very expensive, for example in medicine or finance. While a cost sensitive setting for determining abstaining classifiers is well known and researched, the reward setting this thesis is based on, is not.

This thesis describes the underlying reward setting and proposes a general and flexible approach to determining abstaining classifiers from reward. Similarities to reinforcement learning—from which the idea of using reward instead of cost comes—are shown.

The proposed method is unstable and can produce non-optimal abstaining classifiers—a direct consequence of its generality and the small scope on which was experimented. The method is intended to be the underlying framework for more specialized and sophisticated methods, which are outlined in this thesis.

While being unstable, empirical tests on real world data sets show that the proposed method can produce approximate optimal abstaining classifiers.

Keywords: Abstaining classifiers, classification with rejection, reliable machine learning

Contents

List of Figures

List of Tables

1. Introduction

An abstaining classifier (see e.g. Vanderlooy et al., 2009)—also called a classifier with reject option (see e.g. Fischer et al., 2016)—is a kind of confidence predictor. It can refuse from making a prediction if its confidence in the prediction is not high enough. An abstaining classifier has an abstention rule based on which it decides, whether it should return a prediction or abstain from it. The abstention rule should be close to optimal. Optimality is dependent on a performance metric—in this thesis called setting—set beforehand.

This thesis introduces a new kind of method for approximating the optimal abstention rule based on a reward function—better known from reinforcement learning than from the supervised learning setting (see e.g. Sutton and Barto, 2018, Chapter 1). The method treats the reward function mostly as unknown and only constraints it in one way; the reward function must assume that all examples it sees are independent and identically distributed (iid). This makes the reward setting a very general approach and gives quite the amount of freedom in designing the reward function, which describes the real world application domain of the abstaining classifier best.

In supervised learning the concept that is closest to reward functions are cost functions. Many abstract types of cost in supervised learning, from which a cost function can be derived, are known (see Turney, 2002).

Probably today's most used methods for obtaining an optimal abstention rule for reducing the expected cost of an abstaining classifier are based on the receiver operating characteristic (ROC) rule (see Tortorella, 2000; Pietraszek, 2005; Vanderlooy et al., 2009; Guan et al., 2018). The method presented in this thesis is more flexible than the methods based on the ROC rule and can—depending on the context of the classification problem—produce results better interpretable than results from a cost setting (see Chapter 2). Also it is more natural with multi-class classification problems than the methods based on the ROC rule, all assuming binary classification problems, wherefore the classifiers generated by these methods must be transformed to multi-class classifiers for non-binary problems.

On the other hand the presented method can suffer from its very general approach and only produces approximations. This can result in non-optimal and unstable abstention rules. Also, the abstention rule of an abstaining classifier that is optimized in the reward setting is weaker than the abstention rule of ROC based abstaining classifiers.

The presented method is intended as an underlying framework for more sophisticated systems and only scratches at the possibilities of generating abstaining classifiers based on a reward setting.

This thesis first presents a motivational example. In Chapter 3 the proposed method is presented and discussed. Also similarities of abstaining classifiers based on a reward setting to reinforcement learning agents are given. After that experiments on data sets from the UCI machine learning repository (see Dua and Graff, 2017) are discussed. Further research ideas and possible optimizations of the presented method are outlined afterwards. At last a conclusion is drawn.

2. Motivational example

This chapter will point out the usefulness of abstaining classifiers in real world application domains where reliability is key. It will show an example why the reward setting can improve readability in some domains. First another example, for which the cost setting—more commonly used in supervised learning—comes more natural is given and the differences are discussed.

Abstaining classifiers—compared to typical classifiers, which classify every prediction, maybe even without a confidence value in it (then called a bare prediction)—can be easily integrated into and enhance processes where they partially replace some of the decision making, since they can delegate the abstained predictions back to the underlying process. The use of abstaining classifiers in domains where reliability—in regard to prediction errors—is important, has an interesting aspect in giving reliability while still being able to decrease work, cost, etc. to some degree. This is a valuable property if there does not exist a typical classifier good enough to fully replace the underlying process.

Many real world application domains for abstaining classifiers can express a cost function associated to the decisions about predicting and abstaining of the classifier—which then chooses the abstention rule with which it produces the least amount of cost, therefore minimizing the cost of introducing the abstaining classifier to the process.

For example, the real world application domain could be a facial recognition system at a company which regulates which employee can enter a trust zone and which can not. The process which should be enhanced with the facial recognition system is a manual process where the employee has to fill out a form in order to receive a key which opens the trust zone.

In this example, the costs of miss-classifying an unauthorized person as authorized can be huge for the company while abstaining or classifying an authorized employee as unauthorized produces quite low costs—the authorized employee just has to start the manual process, which should be replaced by the facial recognition system.

On the other hand, for some real world application domains a reward function based on which the abstaining classifier chooses its abstention rule by maximizing the reward—rather than minimizing the cost—comes more natural.

Such a domain would be the finance industry, where we often can associate a certain amount of money an abstaining classifier can produce or safe by supporting the decision making of an underlying process.

An example for such a process would be the process of a bank for granting a consumer credit. The bank requests information about the consumer from a credit bureau in order to assess the consumer's credit default risk. Now the bank wants to predict the consumer's credit default risk based on information it has about the consumer. If the credit default risk is very high or very low the bank can save money not making a request to the credit bureau for this consumer. The optimal abstention rule for the abstaining classifier making the prediction about the credit default risk can easily be determined by a reward function.

Every correct decision saves the bank the money the request to the credit bureau costs. Every miss-classification costs the bank either the amount of money it would gain by granting the credit, or the money it loses by giving a credit to somebody that does not pay the installments. Abstention cost is the cost of making a request to the credit bureau, if the abstaining classifier rejects the prediction.

Using a reward function—like in the example above—instead of a cost function—has an advantage in readability. One can easily assess the gain of introducing the abstaining classifier to the process. Is the reward generated by the abstaining classifier higher than zero, the process is enhanced by the abstaining classifier. Otherwise the abstaining classifier would produce more cost than it would save and it is not valuable for the bank to introduce it to its process of assessing a consumer's credit default risk.

3. Proposed method based on reward

Let $\mathbf{X}$ be the observation space and $\mathbf{Y}$ the label space. $|\mathbf{Y}| < \infty$, since only classification is discussed. Let $\mathbf{Z}$ be the cartesian product of $\mathbf{X}$ and $\mathbf{Y}$: $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$. $\mathbf{Z}$ is called the example space. Let an example z_i from $\mathbf{Z}$ be: $z_i := (x_i, y_i); z_i \in \mathbf{Z}$. A data set[1] containing examples $z_1, \ldots, z_n$ is annotated as $\wr z_1, \ldots, z_n \int$.

3.1 Scoring classifiers

A classical machine learning predictor—in the previous chapter called a typical classifier—can be represented by a function

$$D : \mathbf{Z}^* \times \mathbf{X} \to \mathbf{Y}. \tag{1}$$

Its first argument being a data set with an arbitrary length the classifier is trained on, while the second is an observation which should be predicted (mapped to a label from $\mathbf{Y}$).

Let $D_{\wr z_1,\ldots,z_n \int}$ be a classical machine learning predictor trained on the data set $\wr z_1, \ldots, z_n \int$ and let $D_{\wr z_1,\ldots,z_n \int}(x)$ be equivalent to (1) with the first argument being $\wr z_1, \ldots, z_n \int$.

The proposed method relies on scoring classifiers. A scoring classifier does not return just a label but instead returns some score for each label from the label space. The scores can be either ascending or descending in order, ascending meaning higher scores are better than lower; descending is the opposite. A score could be a probability or just an uncalibrated confidence value (see Vanderlooy et al., 2009).

Let S be a scoring classifier:

$$S : \mathbf{Z}^* \times \mathbf{X} \to (\mathbf{Y} \to \mathbb{R}). \tag{2}$$

S takes the same arguments as (1) but instead of producing bare predictions it returns a function which maps every label from the label space to a score determined by S.

[1] not an actual set but a multi-set since it can contain the same element more often than one time.

The method proposed is only interested in the highest score and the associated label. For that two functions k and v are defined:

$$
\begin{aligned}
k(S_{\lbag z_1,\ldots,z_n \rbag}, x) &= \operatorname*{arg\,max}_{y \in \mathbf{Y}} S_{\lbag z_1,\ldots,z_n \rbag}(x)(y) \\
v(S_{\lbag z_1,\ldots,z_n \rbag}, x) &= \max_{y \in \mathbf{Y}} S_{\lbag z_1,\ldots,z_n \rbag}(x)(y).
\end{aligned}
$$

The composition kv of k and v returns the tuple with the label mapped to the highest score:

$$
kv(S_{\lbag z_1,\ldots,z_n \rbag}, x) = (k(S_{\lbag z_1,\ldots,z_n \rbag}, x), v(S_{\lbag z_1,\ldots,z_n \rbag}, x)). \tag{3}
$$

3.2 Abstaining classifiers

An abstaining classifier A can be defined as a similar function as (1), with the only difference being the return value:

$$
A : \mathbf{Z} \times \mathbf{X} \to \mathbf{Y} \cup \{\bot\}
$$

A can return a label from $\mathbf{Y}$, but also $\bot$, indicating that A would like to abstain from making a prediction.

Let $\mathbf{S_Z}$ be the set of all scoring classifiers defined like (2) on the example set $\mathbf{Z}$. The proposed method is interested in transforming a scoring classifier $S \in \mathbf{S_Z}$ to an abstaining classifier A. In order to do that a threshold $T \in \mathbb{R}$—the abstention rule of A—is defined and A can be represented as a composition of S and T. Let $S_{\lbag z_1,\ldots,z_n \rbag}$ be a scoring classifier which produces scores in ascending order, T a threshold and x an observation to be predicted. The abstaining classifier A composed of $S_{\lbag z_1,\ldots,z_n \rbag}$ and T predicts x as follows:

$$
A(\lbag z_1,\ldots,z_n \rbag, x) = \begin{cases} k(S_{\lbag z_1,\ldots,z_n \rbag}, x) & \text{if } v(S_{\lbag z_1,\ldots,z_n \rbag}, x) \geq T \\ \bot & \text{if } v(S_{\lbag z_1,\ldots,z_n \rbag}, x) < T \end{cases}. \tag{4}
$$

The equivalent A defined for a scoring classifier $S_{\lbag z_1,\ldots,z_n \rbag}$ that produces scores in descending order just exchanges the comparison operators with their respective opposite:

$$
A(\lbag z_1,\ldots,z_n \rbag, x) = \begin{cases} k(S_{\lbag z_1,\ldots,z_n \rbag}, x) & \text{if } v(S_{\lbag z_1,\ldots,z_n \rbag}, x) \leq T \\ \bot & \text{if } v(S_{\lbag z_1,\ldots,z_n \rbag}, x) > T \end{cases}. \tag{5}
$$

A single threshold as abstention rule is rather unconventional and is one reason the proposed method is unstable. Using a single threshold for all labels is a strong constraint to put onto the scoring classifier, because it must be invariant to the label distribution. Imagine a classification problem where one label makes up 90 percent of all examples and the scoring classifier is not invariant to the label distribution. This could lead the classifier to produce higher scores for observations with the label, which makes up 90 percent. The result could be an abstaining classifier that does not predict any examples which do not have the dominant label, even though with such a distribution predicting the submissive labels would probably be more interesting.

ROC based and other methods for generating abstaining classifiers address this problem by using abstention windows instead of a single threshold as their abstention rule (see Friedel et al., 2006).

Let $\mathbf{Y}$ be a binary problem $\mathbf{Y} := \{P, N\}$, where P is called the positive label and N the negative label. The margin $m : \mathbf{Y} \times \mathbb{R} \to (-1, 1)$ is a function that combines the label with the confidence value and returns a number in the interval of $(-1, 1)$. The closer the return value of m is to the edges of the interval, the more confident the scoring classifier is. -1 means perfectly confident the label is N and 1 means perfectly confident the label is P (see Friedel et al., 2006).

In Guan et al. (2018) a similar method is described, constraining the output of the margin m not on $(-1, 1)$, but instead using only the likelihood of an observation x having the positive label P $(m : \mathbb{R} \to (0, 1))$.

Both Friedel et al. (2006) and Guan et al. (2018) define an abstention window a as a tuple $a := (t_1, t_2); t_1 < t_2$ with two thresholds. An abstaining classifier of the form defined in (4) with an abstention window instead of a threshold predicts an observation x as:

$$A(\lbag z_1, \dots, z_n \rbag, x) = \begin{cases} P & \text{if } m(kv(S_{\lbag z_1, \dots, z_n \rbag}, x)) > t_2 \\ \perp & \text{if } t_1 \leq m(kv(S_{\lbag z_1, \dots, z_n \rbag}, x)) \leq t_2 \\ N & \text{if } m(kv(S_{\lbag z_1, \dots, z_n \rbag}, x)) < t_1 \end{cases} .$$

This addresses the problem of using a single threshold T for predictions on both labels from $\mathbf{Y}$. The constraint of abstention windows is that they are only defined on binary problems and must be transformed in order to use them in a multi-class setting. This could be done with the one-vs-one or the one-vs-all approach, in which multiple binary classifiers are learned (see e.g. Murphy, 2012, Chapter 14.5). But, like stated in Friedel (2005) multi-class problems increase the complexity of ROC based and other methods, because when using a one-vs-one or one-vs-all approach it is possible that more than one label gets predicted by the abstaining classifier (see Friedel, 2005).

On the other hand an arbitrary number of labels can be predicted with a single threshold, though the solution could be sub-optimal and is depending heavily on the underlying scoring classifier.

This thesis does not address the problem of using a single threshold in the empirical study presented in Chapter 4, but a possible solution is given in Chapter 5.

3.3 Abstaining classifiers based on a reward system

The novel approach of this thesis is using a system based on reward which is maximized rather then cost that is minimized in order to determine the optimal abstention rule for

	stateless	stateful
accumulated	true	true
single step	true	false

Table 1: Possible combinations of the two known properties of a reward function. It is not possible to have a stateful single step reward function, because a single step reward function is only dependent on the true and predicted label of one example.

abstaining classifiers. Like stated in Chapter 1, using a reward function—like in reinforcement learning—in a supervised learning setting is rather uncommon. In Chapter 1 and Chapter 2 some reasons why using reward instead of cost are given.

Another aspect of cost, which makes it less flexible than reward, not previously discussed, is that it is only defined on $\mathbb{R}^+$, while reward is defined on $\mathbb{R}$. Reward combines cost with gain.

Let ρ be a reward function:

$$\rho : \mathbf{Y}^* \times \hat{\mathbf{Y}}^* \to \mathbb{R}^*. \tag{6}$$

ρ takes two arbitrary, but equal long vectors with labels from $\mathbf{Y}$ and from $\hat{\mathbf{Y}}$. $\hat{\mathbf{Y}}$ can be equal to $\mathbf{Y}$ or also contain an element indicating abstention $\perp$. The first vector contains the true labels of some sequence of examples, the second contains the predicted labels from some classifier for the same sequence. ρ returns a reward for each tuple of true label and predicted label from the parameter vectors.

The reward function is basically treated as a black box function; the only knowledge we have is, whether ρ produces single-step reward or accumulated reward values and whether ρ is stateful or stateless (see Table 1).

Treating the reward function this way makes it more flexible than a cost setting which uses cost matrices (see Fischer et al., 2016). A cost matrix C for a binary abstaining classifier is defined as

$$C := \begin{pmatrix} C(P,P) & C(P,N) & C(P,\perp) \\ C(N,P) & C(N,N) & C(N,\perp) \end{pmatrix}.$$

A cost function c with the same definition as (6) based on such a cost matrix C would be defined as $c(\vec{t},\vec{p}) = [i = 1,\ldots,|\vec{t}| : C(t_i,p_i)]^T$ and is basically the inverse of a single step reward function—with the difference that $C(P,P)$ and $C(N,N)$ normally do not have a gain associated to them, because then the cost matrix would not be true to its cost setting. A cell of a cost matrix C would provide a gain if its value is smaller than zero.

A reward function that returns already accumulated rewards provides an even more flexible setting than single step reward—which is only dependent on one example's true and predicted label—because it can introduce the concept of state in itself (see Sutton and Barto, 2018, Chapter 1).

For example, the abstaining classifier could be a bettor betting on the outcome of a card game. It starts with a certain amount of money and always bets two thirds of the amount it currently has. Every example is one match and it is possible to derive a certainty measure based on some information about the match. The reward is the amount of money the classifier wins or loses. It gains a certain amount—depending on how much money the classifier owns after the last match it has bet on—if it decides to bet on the current match and does so correctly. It loses two thirds of its reward up to the current bet if the classifier was wrong. A reward function like this is not stateless like a single step reward function and is a commonly used in the reinforcement learning setting (see Sutton and Barto, 2018, Chapter 1).

An interesting question is where to draw the line between an abstaining classifier that maximizes reward and a reinforcement learning agent, because the bettor described above could also be defined as a reinforcement learning agent. This thesis will not declare a clear differentiation between the two concepts, but the interaction with the environment seems to be a good point for differentiation. If the predictions of the abstaining classifier alter reality (the predictions of the bettor above most certainly would change reality), it behaves like an agent, otherwise it is just an abstaining classifier.

An argument for such a differentiation would be, that supervised learning—on which the focus of this thesis lies—is underlined by the assumption, that all observations $x_i \in \mathbf{X}$ observed are independent from the other observations $x_j \in \mathbf{X}$, but that they share the same unknown distribution. This assumption is called the iid assumption (independent and identically distributed) (see Clauset, 2011) and makes the concept of state irrelevant to our observations, which would not be the case if the predictions alter reality. The reward function must always assume, that observations are iid; it can not introduce state to the observation space, only to itself—the reward, like in the bettor example.

3.4 Method for approximating the optimal threshold for abstention based on a reward function

For approximating the optimal threshold—which maximizes the expected reward in the proposed reward setting—an architecture comparable to and influenced by the meta-conformal prediction approach described in Smirnov et al. (2009) is proposed. The architecture of an abstaining classifier based on reward is comparable to the combined classifier used for meta-conformal prediction. A combined classifier *B:M* uses a base classifier *B* defined like (1) and a conformal predictor *M* in order to extend *B* with a confidence measure. *B:M* can then be transformed to an abstaining classifier by defining a threshold *T* in the confidence values generated by *M* using the ROC isometrics approach (see Smirnov et al., 2009; Vanderlooy et al., 2009; Fassbender, 2019).

An abstaining classifier A—defined like (4) or (5)—in the reward based setting described above, can also be described as a combined classifier $A := S{:}Reg$. S is a scoring classifier defined like (2) and Reg is a regressor (defined like (1) with $\mathbf{Y} := \mathbb{R}$). Reg is not necessarily needed and is only used in order to determine the threshold T for A, which can be done by other means described below. Chapter 4 shows how well using different regressors perform in comparison to just taking the threshold which has generated the highest reward on the training set.

The threshold of A that approximates the maximum expected reward is defined during the training phase. Let $\Lbag z_1, \dots, z_n \Rbag$ be a training set. $\Lbag z_1, \dots, z_n \Rbag$ is split into k roughly equal sized partitions using the k-fold method (see Hastie et al., 2009, Chapter 7.10; Algorithm 1, line 2).

For each partition combine the other $k-1$ partitions to a training set; train a scoring classifier S on this set and let it predict on the partition it was not trained on. Add the true labels from the examples in the predicted partition and kv (see Equation 3) of all predictions to a prediction set $P \subseteq \mathbf{Y}^n \times \hat{Y}^n \times \mathbb{R}^n$ (see Algorithm 1, lines 3–11).

After that P is sorted based on the scores, transforming it into a sequence. If S produces scores in ascending order P is also sorted in ascending order, otherwise in descending order. This is done in order to simplify reward accumulation, since the problem of finding the reward at a certain threshold is reduced to just taking all elements from P which come first in the sequence. The reward from the first two columns of P—with a reward function ρ defined like (6)—is computed. Since iid (see previous chapter) is assumed, the parameter vectors for ρ can be provided with any ordering and stateful reward functions must also assume iid on the sequence it sees as its parameters. Every reward is related to an element from P and the reward is combined with the scores from P to build the reward points R. R can be represented as a matrix $R : n \times 2$, where the first column contains all scores and the second column contains the associated reward:

$$R := \begin{pmatrix} s_1 & r_1 \\ \vdots & \vdots \\ s_n & r_n \end{pmatrix}$$

(see Algorithm 1, lines 12, 13).

If ρ is a single step reward function the rewards are accumulated. Since R is sorted based on the scores, the reward at a single point is the sum of all rewards previously seen. The accumulated version of R is $R' := [(s_i, \sum_{j=1}^{i} r_j); i = 1, \dots, n]^T$ (see Algorithm 1, lines 14–16).

At last T is derived from R. If Reg is defined, it is trained on R, with s_i as observation and r_i as the label. T is set equal to the score for which Reg predicts the highest reward; the local maximum of Reg. Only the local maximum is of interest, which means T must lie in the interval derived from the convex hull $C := \operatorname{Conv}(\{i = 1, \dots, n : R_{i1}\})$ of all scores

Algorithm 1 : k-fold method for determining the threshold for an abstaining classifier based on a reward function

```
Input:
  $S$: a socring classifier,
  $\rho$: a reward function defined like (6),
  data set: $\Lbag z_1, \ldots, z_n \Rbag$,
  $k$: the amount of partitions,
  $Reg$: a regressor (optional)
Output:
  T: threshold
$P := \{\}$
split data set into $k$ roughly equal sized partitions $\text{split}_1, \ldots, \text{split}_k$
for all $\text{split}_i, i = 1, \ldots, k$ do
  combine all splits $\neq \text{split}_i$ to a training set
  train $S$ with the training set
  let $S$ predict examples in $\text{split}_i$
  for all elements in prediction of $S \times$ the true labels of $\text{split}_i$ do
    get the label associated with the highest score for the element with (3)
    add the true label, the predicted label and the score to $P$
  end for
end for
sort $P$ based on the scores. In ascending order if $S$ returns ascending scores. Otherwise
  in descending order
$R :=$ scores from $P \times \rho(P_{i1}, P_{i2}); i = 1, \ldots, n$
if $\rho$ is a single step reward function then
  accumulate reward in $R$
end if
reduce $R$ so all scores are unique (optional)
train $Reg$ (optional)
determine $T$ with (7) or (8)
return $T$
```

s_i	r_i		s_i	r_i
0.98	1			
0.98	2		0.98	2
0.97	3			
0.97	2	⇒		
0.97	1		0.97	1
0.96	0		0.96	0

Figure 1: An example for reducing R built from a single-step reward function which gives $+1$ for a correct prediction and -1 for a false one. R is already accumulated. If T would be determined from the unreduced R, 0.97 would be the optimal threshold, because it has the highest associated reward, determined like (8). The problem is that two errors produced by the scoring classifier for examples with the same certainty 0.97 are concealed. Reducing R to only the last element of each score makes certain that no concealing can happen.

generated from S during the training:

$$T := \underset{s|\min C \leq s \leq \max C}{\arg\max} \; Reg(s). \tag{7}$$

If Reg is not defined, T could be derived from R by taking the score which has the highest associated reward

$$T := R_{i1} : \underset{i|1 \leq i \leq n}{\arg\max} f(i) = R_{i2}. \tag{8}$$

For determining T like this, R would be reduced so each score is unique, since T can only split between two scores s_i, s_j, if $s_i \neq s_j$. This step is optional, because maybe for use with Reg an unreduced set of points is wanted (see Algorithm 1, line 17).

Making R unique could be done in different ways, for example—if ρ is a single step reward function—it would make sense to take the last tuple of a sub-sequence where each tuple has the same score, since it contains the most information about the reward (see Figure 1). One could also reduce them by averaging their rewards, etc.

3.5 Equivalences to reinforcement learning

Using a reward system—like described above—to determine an abstaining classifier makes the whole process quite similar to the whole setting of reinforcement learning; not only the reward part. This chapter lists some more aspects which makes a reinforcement learning agent and an abstaining classifier look alike, but also shows where both concepts differ.

reinforcement learning agent	abstaining classifier
state s_i	observation x_i
action a_i	prediction p_i
environment	reality providing examples from $\mathbf{Z}$
action changes state of environment	iid assumption (reality not altered by predictions)
policy π	threshold T
trial and error	examples from reality

Table 2: Comparison of a reinforcement learning agent with an abstaining classifier in the reward setting.

A reinforcement learning agent, also called an autonomous agent, observes—for each (time-)step t—a state s_t from its environment. Based on s_t the autonomous agent takes an action a_t and the environment transitions to a new state s_{t+1}. At each transition the environment provides a reward value; a feedback for the agent on how well it performs. The agent learns a policy with which it maximizes the expected reward (see Arulkumaran et al., 2017).

An abstaining classifier works quite the same way. It observes an observation $x_t \in \mathbf{X}$. For x_t it produces a prediction $p_t \in \mathbf{Y} \cup \{\perp\}$.

The reward system works a little differently than the one used in reinforcement learning. Assuming a direct reward would mean that the abstaining classifier is used in the perfect online setting in which reality provides the correct answer after every prediction, which is seldom the case and would make the abstaining classifier redundant (see Vovk et al., 2005, Chapter 4.3). While the autonomous agent can use trial and error in order to increase the success of its policy, the abstaining classifier is bound to the already observed data and can only try to generalize from the previous observations to unseen ones. The equivalent of the abstaining classifier to the policy of the agent would be its abstention rule—the threshold T (see Table 2).

The most obvious difference is that the agent actively interacts with the environment, while the abstaining classifier should be irrelevant to its environment—reality providing the classifier with examples but without assuming the predictions in any case alter the environment, because it would violate the iid assumption. The only way interaction with the environment can be indirectly represented is through a stateful reward function, which can simulate decisions made by the abstaining classifier (see Chapter 3.3).

4. Experiments

This chapter will show some experiments in which the proposed method, with different configurations, is tested on real-world data sets from the UCI machine learning repository (see Dua and Graff, 2017).

The configuration contains two different scoring classifiers, eighteen reward functions and five regressors—approximating the optimal threshold like (7)—plus the bare threshold derived like (8).

4.1 Data sets

Six data sets where chosen for the experiments. The first being the bank marketing data set[2] (`bank`). This data set contains information about the success of a marketing campaign (phone calls) of a Portuguese bank. The goal is to predict whether a phone call to a potential customer results in success, which means the potential customer subscribes to a term deposit (see Moro et al., 2014). The data set has seventeen features and is a binary classification problem with 41,188 examples. The data set is unbalanced, it contains far less successful phone calls then unsuccessful ones (see Moro et al., 2014).

The second data set tested was `bank-additional`. It is the same data set as `bank`, but has three more features.

The third data set is the car evaluation data set[3] (`car`). It is described in Bohanec and Rajkovič (1988) and contains 1,728 examples with six attributes. Noteworthy is the fact that all features and the label are discrete with just three or four manifestations.

Also tested where the default of credit card clients data set[4] (`credit card`). It contains information about default payments in Taiwan and the goal is to predict whether an observation represents a credible client or not (see Yeh and hui Lien, 2009). This is closely related to the example described in Chapter 2.

Probably the most famous data set used is USPS data set (`usps`). It is used in hundreds of papers and books. It contains 9,298 examples (images) of handwritten digits from real life zip codes collected by the US Postal Service office in Buffalo, NY (see Vovk et al., 2005, Appendix B.1). The observations are a 16×16 matrix where each cell is in the interval of $(-1, 1)$. Each cell represents the brightness of a pixel. The labels are the interval 0 to 9 (see LeCun et al., 1989; Fassbender, 2019).

The last data set tested was the wine quality data set[5] (`wine`). Each example represents a sample of "vinho verde" from northern Portugal. The twelve attributes are physicochemical properties of the sampled wine which are mapped to a sensory output—the quality of the wine from zero to ten. The data set is unbalanced in two ways. It contains only 1,599 red wine samples, but 4,898 white wine samples. The more important imbalance is

[2] `https://archive.ics.uci.edu/ml/datasets/bank+marketing`

[3] `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`

[4] `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

[5] `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`

data set	# examples	# features	$\vert\mathbf{Y}\vert$
`bank`	41,188	17	2
`bank-additional`	41,188	20	2
`car`	1,728	6	4
`credit card`	30,000	24	2
`usps`	9,298	256	10
`wine`	6,497	12	11

Table 3: Characteristics of the tested data sets.

the distribution of the label. Most samples have a quality of five or six (see Cortez et al., 2009).

4.2 Scoring classifiers

Two different underlying scoring classifiers were tested. The first—`cp`—is a conformal predictor based on a 1-nearest neighbor method as its nonconformity measure. A conformal predictor—like its name already suggests—tries to determine a confidence value by predicting how an example conforms to previous seen examples. If it is used in the online setting and $\mathbf{Z}$ is exchangeable, its outputs are valid, in the sense that a conformal predictor Γ^ϵ makes errors at a rate of ϵ or less. ϵ—in the original setting of conformal prediction—is defined beforehand and is called the significance level. $1 - \epsilon$ is the confidence level (see Vovk et al., 2005; Fassbender, 2019).

The conformal predictor `cp` was modified to being a scoring classifier like (2), which produces scores in descending order. Refer to Fassbender (2019) for more information on how to derive an abstaining classifier from a conformal predictor.

A conformal predictor uses an underlying nonconformity measure $N : \mathbf{Z}^* \times \mathbf{Z} \to \mathbb{R}$ to generate a nonconformity score for an example $z := (x, y)$ (see Vovk et al., 2005; Fassbender, 2019). The nonconformity score used in the tests is based on the 1-nearest neighbor method and looks like:

$$N(\wr z_1, \dots, z_n \wr, z) = \frac{\min_{i=1,\dots,n:y_i=y} d(x_i, x)}{\min_{i=1,\dots,n:y_i\neq y} d(x_i, x)}.$$

$d(x_i, x)$ is the euclidean distance in the observation space between x_i and x. It divides the closest distance to an example with the same label with the closest distance to an example with another label (see Vovk et al., 2005; Fassbender, 2019).

The nonconformity score is then transformed to a p-value in order to see how well the example conforms to the previous seen examples. Let $\alpha_i, i = 1, \dots, n$ be the nonconformity scores of $\wr z_1, \dots, z_n \wr$. Let α be the nonconformity score for z. The p-value for z is computed as

$$\frac{|\{i = 1, \ldots, n+1 : \alpha_i \geq \alpha\}|}{n+1}.$$

The higher the p-value, the more conforms z with the previous seen examples and the confidence is high that the prediction is correct.

`cp` is based on the implementation from the libconform library (see Fassbender, 2019).

The second scoring classifier tested was `rf`, a scoring classifier based on a random forest with 100 trees.

The score for an observation x is computed as the mean of the predicted label probabilities of all trees. A tree predicts the label probabilities as the fraction of examples from the leaf in which x falls. For example, let $\mathbf{Y} := \{0, 1, 2\}$ and the leaf in which x falls contains ten examples from the training set. Five examples had label 0, three label 1 and two label 2. The predicted probabilities of the tree are $[(0, \frac{1}{2}), (1, \frac{3}{10}), (2, \frac{1}{5})]$.

`rf` transforms the probability predictions to complementary probabilities in order to make it a scoring classifier that produces scores in descending order, like `cp`.

The scoring classifier is based on the implementation from the scikit-learn library, version 0.21.2 and uses the default parameters (except the amount of trees) (see Buitinck et al., 2013).

It should be noted that the performance of the scoring classifiers are deemed irrelevant for the experiments. The focus of the experiments lies on approximating the optimal threshold and not on finding the best scoring classifier.

4.3 Reward functions

Eighteen reward functions defined like (6) were tested.

In order to have comparable results for the asymmetric reward functions three matrices M^{50}, M^{200} and M^{1000} were generated randomly[6] beforehand; each with a different degree of freedom in range which is denoted by the superscripts. Each matrix was generated with 11×11 elements. Every row and every column is associated to one element in $\mathbf{Y}$ for each data set. Every label in $\mathbf{Y}$ is an integer that can index the matrices. Every row represents a prediction and every column the true label. A binary data set, like `bank`, would index the first two rows and the first two columns, while `wine` uses every row and every column. For M^{50}, M^{200} and M^{1000} refer to Appendix D.

Below, all tested reward functions with their definitions are listed. All, except the reward functions which add an abstention cost to abstained predictions, are single step and therefore stateless reward functions. The reward functions which add an abstention cost are also stateless, but return accumulated reward. Abstaining cost is easily integrated in a reward function, since its input vectors are ordered after the score. Every element

[6] the pseudo random number generator from the Python standard library was used (see van Rossum and the Python development team, 2019, Chapter 9.6).

is assumed to represent the threshold; each element that comes after is abstained and a cost for each is substituted from the reward of the element. In the tested reward function, abstention cost was set to 1, for each abstained example, resulting in abstention costs of $|\vec{t}|-i$, i being the index of the element for which the reward is computed, $\vec{t}$ being the vector of true labels from the parameters, $|\vec{t}|$ the size of the vector.

The tested reward functions are:

- `simple`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -1 & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `simple scaled 1:5`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -5 & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `simple scaled 1:20`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -20 & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `simple scaled 1:100`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -100 & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `asymmetric loss 50`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -M^{50}_{p_i,t_i} & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `asymmetric loss 200`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -M^{200}_{p_i,t_i} & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `asymmetric loss 1000`

$$\rho(\vec{t},\vec{p}) = \left[i = 1, \ldots, |\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -M^{1000}_{p_i,t_i} & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random loss

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random loss scaled 1:5

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -5 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random loss scaled 1:20

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -20 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random loss scaled 1:100

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} 1 & \text{if } t_i = p_i \\ -100 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} RND(0,1) & \text{if } t_i = p_i \\ -RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random scaled 1:5

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} RND(0,1) & \text{if } t_i = p_i \\ -5 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random scaled 1:20

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} RND(0,1) & \text{if } t_i = p_i \\ -20 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- random scaled 1:100

$$\rho(\vec{t},\vec{p}) = \left[i = 1,\ldots,|\vec{t}| : \begin{cases} RND(0,1) & \text{if } t_i = p_i \\ -100 * RND(0,1) & \text{if } t_i \neq p_i \end{cases} \right]^T$$

- `asymmetric 50 abstain`

$$\rho(\vec{t}, \vec{p}) = \left[i = 1, \ldots, |\vec{t}| : -(|\vec{t}| - i) + \sum_{j=1}^{i} \begin{cases} M^{50}_{p_j,t_j} & \text{if } t_j = p_j \\ -M^{50}_{p_j,t_j} & \text{if } t_j \neq p_j \end{cases} \right]^T$$

- `asymmetric 200 abstain`

$$\rho(\vec{t}, \vec{p}) = \left[i = 1, \ldots, |\vec{t}| : -(|\vec{t}| - i) + \sum_{j=1}^{i} \begin{cases} M^{200}_{p_j,t_j} & \text{if } t_j = p_j \\ -M^{200}_{p_j,t_j} & \text{if } t_j \neq p_j \end{cases} \right]^T$$

- `asymmetric 1000 abstain`

$$\rho(\vec{t}, \vec{p}) = \left[i = 1, \ldots, |\vec{t}| : -(|\vec{t}| - i) + \sum_{j=1}^{i} \begin{cases} M^{1000}_{p_j,t_j} & \text{if } t_j = p_j \\ -M^{1000}_{p_j,t_j} & \text{if } t_j \neq p_j \end{cases} \right]^T$$

$RND(0,1)$ is a function returning a pseudo random value from the uniform distribution constrained on the interval $(0,1)$.

4.4 Regressors

Five different kernelized regressors where used in order to test whether using (7) to determine the optimal threshold can produce better approximations than deriving the threshold like (8). (8) is called `bare` in this section.

Three regressors, `GP [1e-3, 1]`, `GP [1e-1, 1]` and `GP [1, 2]` are based on Gaussian processes, while `SVR C1` and `SVR C100` are based on support vector regression (see e.g. Murphy, 2012, Chapters 14, 15).

All regressors based on a Gaussian process are based on a RBF kernel defined like

$$k(x, x') = c * \exp\left(- \frac{\|x - x'\|}{2l^2} \right), \tag{9}$$

c being a constant optimized during training and l is called the length scale, also optimized during training. For each regressor, c is bound to the interval $[0.001, 100]$, while the length scale is bound to the interval in the regressor's name. The different intervals are designed to generate regressors with different bias-variance trade-off. `GP [1, 2]`, for example, has a high variance, but low bias, while `GP [1e-3, 1]` has a low variance and a high bias. c and l are optimized during training with the L-BFGS-B optimization algorithm from the scipy library (see Zhu et al., 1997; Jones et al., 2001–2019).

`SVR C1` and `SVR C100` are both ϵ-support vector machines for regression. They differ in their penalty parameter C. Both are also based on the RBF kernel:

$$k(x, x') = \exp\left(- \frac{\|\psi(x) - \psi(x')\|}{2l^2} \right). \tag{10}$$

(10) is defined like (9), without the c parameter and x, x' replaced by $\psi(x)$ and $\psi(x')$. ψ maps an observation from $\mathbf{X}$ to an enlarged space (see Hastie et al., 2009, Chapter 12.3).

All regressors are based on the implementations from the scikit-learn library (see Buitinck et al., 2013).

4.5 Results

The settings of the experiments were all the same. It was tested how well abstaining classifiers (in this case defined as (5), since both underlying scoring classifiers produce scores with descending order) with different thresholds determined by Algorithm 1 with different settings (described above) performed. k was always set to five and each data set was randomly permuted. The reward points were all reduced like Figure 1 displays. Ten percent of the randomly permuted data set were used as a test set and the rest as training data.

The performance of the underlying scorer was not tested, only how close the threshold approximates the optimal result. Therefore, the results only show how much percent of the best possible reward on the test set the abstaining classifier achieves with the determined threshold.

The results of each experiment are shown in Appendix B. Appendix C shows some example plots where the score is mapped to the reward and the predictions of the regressors are shown. The example plots are from tests on the `usps` data set with `cp` as scoring classifier. In Appendix A metrics for each reward function over every experiment are displayed.

Table 4 shows the average, median and standard deviation of every regressor and `bare` over all data sets. Table 5 shows the same information split by the underlying scoring classifier.

The overall performance of each threshold approximator (7) and (8) is quite good, while (8) (`bare`) on average outperformed the thresholds determined by regressors. It is also more stable, having a standard deviation σ a lot smaller than the other approximators have. While this can be an argument not to use (7), actually `bare` was outperformed 44 percent of the time by a regressor with an average of nearly two percent (0.018) more reward on the test set.

The regressors perform better with `rf` as scoring classifier than with `cp`, while `bare` is constant on both (see Table 5).

While overall the performance of the regressors was good, on all reward functions with an abstaining cost, all regressors somewhat failed utterly, having just around zero percent of the maximum possible reward on the test set (see Appendix B). The regressors based on Gaussian processes also struggled on the higher scaled `simple` and `random loss` reward functions on the `bank` data set. This is why their average is far lower than their mean (see Table 4). `bare` also failed utterly for some reward functions with an abstaining cost, but only on the `car` data set.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
mean	0.975	0.811	0.803	0.803	0.754	0.787
median	0.997	0.972	0.981	0.990	0.988	0.992
σ	0.110	0.344	0.364	0.366	0.403	0.381

Table 4: Metrics over all data sets, scoring classifiers and reward functions.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
cp						
mean	0.981	0.790	0.762	0.756	0.661	0.697
median	0.997	0.964	0.979	0.978	0.967	0.984
σ	0.076	0.363	0.390	0.387	0.440	0.423
rf						
mean	0.970	0.833	0.844	0.851	0.847	0.876
median	0.996	0.974	0.982	0.994	0.992	0.994
σ	0.135	0.325	0.332	0.340	0.339	0.311

Table 5: Metrics for each scoring classifier over all data sets and reward functions.

These utter failures are the reason why this method can not be called stable and more research must be done, in order to acquire more knowledge on how to avoid such faulty behavior.

5. Further research

While the experiments in Chapter 4 substantiate the applicability of the proposed method in its basic setting, many questions are unanswered. There must be more empirical studies on more elaborate reward settings in order to see where the method has its boundaries and how to improve on its stability. The presented method is very general and the goal of this thesis was to present its flexible approach and to discuss its reward based setting. The idea was to present this method as a foundation for more specialized and stable methods, incorporating more knowledge about the domain and more statistical certainty. Also, the context of this method must be improved, adding the performance of the underlying scoring classifier as another point for increasing stability through knowledge of the whole system.

While even for asymmetric and random reward functions the abstaining classifiers trained in Chapter 4 performed well (see Appendix B), using a single threshold can hopefully be improved by an approach again inspired by a method from the conformal prediction framework: Mondrian or conditional conformal predictors (MCPs) (see Vovk et al., 2005; Balasubramanian et al., 2014; Fassbender, 2019).

A MCP partitions $\mathbf{Z}$ into a discrete and finite set $\mathbf{K}$. An element from $\mathbf{K}$ is called a category and a function

$$K : \mathbf{Z}^* \to \mathbf{K}^*$$

is called a taxonomy (see Balasubramanian et al., 2014, Chapter 2).

The idea would be to use such a taxonomy to categorize examples and to define a threshold per category, instead of one threshold for all examples. This would extend the abstention rule of the abstaining classifier to $|\mathbf{K}|$ thresholds. The result could be finer tuned abstaining classifiers, hopefully increasing their performance, which must be tested.

Another way to increase the performance could be to use an ensemble of regressors instead of just one to determine the best threshold, which also remains to be tested. In general, how to chose a good regressor remains an open question not answered in this thesis. Like stated in Chapter 4.4, the regressors tested were just to see how different bias-variance trade-off settings perform. No tuning process was used to find a regressor that performs best, just a few parameters were set before the tests.

A way to decrease training cost would be to replace the k-fold method with a calibration set. The underlying scoring classifier would be trained on a training set once and it would predict on the calibration set instead of splitting the training set into k partitions, resulting in k times the cost of training the underlying scoring classifier. This could be useful approach for big data sets and for scoring classifiers, that have a complex training operation, like support vector machines or neural nets.

6. Conclusion

The goal of this thesis was to discuss the general applicability of a reward based setting in order to define abstaining classifiers. This thesis also proposes a very general and flexible setting for defining such a system and to generate an abstaining classifier from it. The result of the generality of the method can be unstable and non-optimal abstaining classifiers, which is empirically shown on real-world data sets. On the other hand, the empirical study shows, that despite unstable results, the basic method is already applicable to some degree, producing in general abstaining classifiers, which are close to optimal.

In Chapter 5 unanswered questions are listed and ways to improve the proposed method are given.

Hopefully the proposed method can be the general framework on which more specialized and sophisticated approaches are based and the reward setting of this thesis—and the resulting convergence of abstaining classifiers to reinforcement learning agents described in Chapter 3.5—will be further explored.

Appendices

A. Metrics per reward function

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple						
mean	0.986	0.957	0.895	0.907	0.907	0.908
median	0.996	0.968	0.976	0.994	0.993	0.994
σ	0.024	0.044	0.271	0.275	0.275	0.275
simple scaled 1:5						
mean	0.988	0.889	0.979	0.855	0.803	0.834
median	0.994	0.978	0.984	0.955	0.973	0.983
σ	0.017	0.283	0.021	0.238	0.341	0.336
simple scaled 1:20						
mean	0.989	0.890	0.975	0.903	0.740	0.779
median	0.996	0.986	0.986	0.992	0.987	0.990
σ	0.016	0.284	0.027	0.285	0.441	0.396
simple scaled 1:100						
mean	0.993	0.905	0.988	0.908	0.746	0.826
median	0.999	0.996	0.998	0.997	0.994	0.998
σ	0.015	0.286	0.021	0.286	0.444	0.386
asymmetric loss 50						
mean	0.990	0.962	0.888	0.907	0.908	0.908
median	0.999	0.992	0.984	0.994	0.996	0.996
σ	0.024	0.047	0.277	0.279	0.279	0.279
asymmetric loss 200						
mean	0.979	0.952	0.887	0.883	0.875	0.886
median	0.992	0.971	0.980	0.988	0.980	0.978
σ	0.033	0.043	0.282	0.281	0.280	0.281
asymmetric loss 1000						
mean	0.979	0.954	0.895	0.883	0.880	0.890
median	0.991	0.960	0.982	0.973	0.966	0.981
σ	0.025	0.043	0.276	0.275	0.274	0.276

random loss						
mean	0.999	0.957	0.895	0.916	0.917	0.916
median	1.000	0.993	0.990	1.000	1.000	0.999
σ	0.002	0.061	0.279	0.284	0.284	0.284
random loss scaled 1:5						
mean	0.983	0.888	0.975	0.958	0.963	0.885
median	0.989	0.968	0.989	0.977	0.982	0.974
σ	0.023	0.281	0.029	0.048	0.046	0.279
random loss scaled 1:20						
mean	0.976	0.888	0.980	0.872	0.781	0.886
median	0.989	0.984	0.986	0.988	0.982	0.992
σ	0.031	0.284	0.020	0.294	0.379	0.287
random loss scaled 1:100						
mean	0.971	0.896	0.976	0.865	0.800	0.806
median	0.996	0.994	0.996	0.982	0.976	0.992
σ	0.047	0.285	0.039	0.295	0.377	0.379
random						
mean	0.984	0.957	0.906	0.916	0.919	0.887
median	0.995	0.965	0.982	0.996	0.998	0.996
σ	0.026	0.044	0.228	0.231	0.231	0.252
random scaled 1:5						
mean	0.991	0.963	0.893	0.910	0.911	0.880
median	0.996	0.978	0.978	0.998	0.996	0.996
σ	0.013	0.041	0.282	0.287	0.287	0.297
random scaled 1:20						
mean	0.991	0.960	0.895	0.908	0.910	0.885
median	0.994	0.982	0.986	0.996	0.996	0.994
σ	0.012	0.044	0.282	0.286	0.287	0.291
random scaled 1:100						
mean	0.987	0.960	0.899	0.909	0.910	0.881
median	0.992	0.970	0.984	0.994	0.992	0.994

σ	0.015	0.042	0.271	0.274	0.274	0.289
`asymmetric 50 abstain`						
mean	0.998	0.057	0.081	0.246	0.078	0.209
median	1.000	0.012	0.010	0.010	0.000	0.000
σ	0.004	0.127	0.166	0.392	0.175	0.395
`asymmetric 200 abstain`						
mean	0.916	0.169	0.135	0.299	0.132	0.380
median	1.000	0.039	0.007	0.004	0.001	0.005
σ	0.288	0.302	0.308	0.450	0.315	0.482
`asymmetric 1000 abstain`						
mean	0.852	0.403	0.317	0.413	0.397	0.510
median	1.000	0.084	0.043	0.044	0.043	0.575
σ	0.346	0.449	0.437	0.495	0.479	0.496

Table 6: Metrics for each reward function over all data sets and scoring classifiers.

B. Results of the experiments

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.996	0.999	0.999	0.996	0.996	0.996
simple scaled 1:5	0.990	0.961	0.927	0.248	0.248	0.248
simple scaled 1:20	0.997	0.987	0.987	0.000	0.000	0.000
simple scaled 1:100	0.999	0.995	0.995	0.000	0.000	0.000
asymmetric loss 50	0.997	0.999	0.999	0.996	0.996	0.996
asymmetric loss 200	0.992	0.904	0.902	0.898	0.898	0.898
asymmetric loss 1000	0.987	0.915	0.911	0.875	0.875	0.875
random loss	1.000	0.995	0.995	1.000	1.000	1.000
random loss scaled 1:5	0.993	0.920	0.911	0.862	0.862	0.862
random loss scaled 1:20	0.946	0.965	0.965	0.000	0.000	0.000
random loss scaled 1:100	0.999	0.995	0.995	0.000	0.000	0.000
random	1.000	0.995	0.995	1.000	1.000	1.000
random scaled 1:5	1.000	0.991	0.991	1.000	1.000	1.000
random scaled 1:20	1.000	0.991	0.991	1.000	1.000	1.000
random scaled 1:100	1.000	0.991	0.991	1.000	1.000	1.000
asymmetric 50 abstain	1.000	0.001	0.001	0.000	0.000	0.000
asymmetric 200 abstain	1.000	0.001	0.001	0.000	0.000	0.000
asymmetric 1000 abstain	1.000	0.002	0.002	0.000	0.000	0.000

Table 7: Results of tests on the bank data set with cp as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	1.000	0.891	0.950	0.996	1.000	1.000
simple scaled 1:5	0.992	0.969	0.984	0.932	0.992	0.985
simple scaled 1:20	1.000	0.997	0.997	0.998	0.998	1.000
simple scaled 1:100	0.997	1.000	1.000	1.000	0.997	0.997
asymmetric loss 50	0.999	0.888	0.954	0.998	0.999	1.000
asymmetric loss 200	1.000	0.970	1.000	0.999	1.000	0.995
asymmetric loss 1000	0.999	0.959	0.997	0.999	1.000	1.000
random loss	1.000	0.854	0.925	0.992	1.000	0.996
random loss scaled 1:5	0.996	0.946	0.988	0.991	0.991	0.996
random loss scaled 1:20	0.997	0.993	0.966	0.991	0.997	0.997
random loss scaled 1:100	0.998	1.000	1.000	0.992	0.987	0.987
random	0.995	0.903	0.964	0.994	0.996	0.994
random scaled 1:5	0.998	0.904	0.963	0.997	0.998	0.999
random scaled 1:20	0.994	0.899	0.951	0.999	0.994	0.994
random scaled 1:100	0.992	0.904	0.963	0.990	0.992	0.999
asymmetric 50 abstain	1.000	0.057	0.025	0.001	0.000	0.000
asymmetric 200 abstain	1.000	0.063	0.028	0.002	0.001	0.001
asymmetric 1000 abstain	1.000	0.073	0.040	0.006	0.004	0.002

Table 8: Results of tests on the bank data set with rf as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.993	0.893	0.965	0.993	0.992	0.993
simple scaled 1:5	0.991	0.994	0.956	0.495	0.494	0.995
simple scaled 1:20	0.995	0.957	0.957	0.993	0.000	0.995
simple scaled 1:100	0.999	0.997	0.997	0.972	0.000	0.998
asymmetric loss 50	0.996	0.888	0.962	0.995	0.994	0.996
asymmetric loss 200	0.992	0.972	0.997	0.879	0.879	0.992
asymmetric loss 1000	0.990	0.971	0.997	0.878	0.878	0.992
random loss	1.000	0.853	0.935	1.000	1.000	1.000
random loss scaled 1:5	0.991	0.973	0.997	0.873	0.873	0.991
random loss scaled 1:20	0.982	0.962	0.995	0.992	0.000	0.981
random loss scaled 1:100	0.999	1.000	1.000	0.978	0.000	0.999
random	0.992	0.885	0.990	0.999	0.998	0.998
random scaled 1:5	0.992	0.886	0.945	0.999	0.999	0.999
random scaled 1:20	0.993	0.878	0.990	0.999	0.999	0.999
random scaled 1:100	0.990	0.883	0.989	0.999	0.998	0.998
asymmetric 50 abstain	1.000	0.072	0.000	0.000	0.000	0.000
asymmetric 200 abstain	1.000	0.075	0.000	0.000	0.000	0.000
asymmetric 1000 abstain	1.000	0.086	0.000	0.000	0.000	0.000

Table 9: Results of tests on the `bank-additional` data set with `cp` as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.998	0.914	0.958	1.000	0.998	0.999
simple scaled 1:5	0.999	0.989	0.981	0.949	0.999	1.000
simple scaled 1:20	0.995	0.994	0.995	0.995	0.991	0.995
simple scaled 1:100	1.000	0.998	0.998	0.998	0.993	0.993
asymmetric loss 50	1.000	0.914	0.955	0.995	1.000	0.996
asymmetric loss 200	0.997	0.974	0.998	0.989	0.996	0.999
asymmetric loss 1000	0.994	0.954	0.983	0.994	0.988	0.988
random loss	1.000	0.870	0.916	0.991	1.000	0.998
random loss scaled 1:5	0.996	0.960	0.992	0.999	0.996	0.995
random loss scaled 1:20	0.988	0.988	0.986	0.998	0.988	0.993
random loss scaled 1:100	0.982	0.982	0.982	0.973	0.991	1.000
random	0.998	0.909	0.947	0.996	0.998	0.995
random scaled 1:5	0.996	0.904	0.945	0.999	0.996	0.996
random scaled 1:20	0.998	0.896	0.936	0.994	0.998	0.996
random scaled 1:100	0.997	0.902	0.941	0.998	0.997	0.997
asymmetric 50 abstain	1.000	0.051	0.031	0.001	0.000	0.000
asymmetric 200 abstain	1.000	0.056	0.035	0.002	0.000	0.000
asymmetric 1000 abstain	1.000	0.068	0.047	0.005	0.000	0.003

Table 10: Results of tests on the `bank-additional` data set with `rf` as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	1.000	1.000	0.037	0.037	0.037	0.037
simple scaled 1:5	1.000	0.000	1.000	1.000	1.000	0.996
simple scaled 1:20	1.000	0.000	0.986	0.986	0.986	1.000
simple scaled 1:100	1.000	0.000	1.000	1.000	1.000	1.000
asymmetric loss 50	1.000	1.000	0.024	0.024	0.024	0.024
asymmetric loss 200	1.000	1.000	0.000	0.000	0.000	0.000
asymmetric loss 1000	1.000	1.000	0.022	0.022	0.022	0.022
random loss	1.000	1.000	0.015	0.015	0.015	0.015
random loss scaled 1:5	0.987	0.000	0.987	0.987	0.987	1.000
random loss scaled 1:20	1.000	0.000	0.988	0.988	0.988	1.000
random loss scaled 1:100	0.996	0.000	0.996	0.996	0.996	1.000
random	1.000	1.000	0.191	0.191	0.191	0.191
random scaled 1:5	1.000	1.000	0.000	0.000	0.000	0.000
random scaled 1:20	1.000	1.000	0.000	0.000	0.000	0.000
random scaled 1:100	1.000	1.000	0.041	0.041	0.041	0.041
asymmetric 50 abstain	1.000	0.000	0.408	0.408	0.408	1.000
asymmetric 200 abstain	1.000	0.287	0.000	0.000	0.000	1.000
asymmetric 1000 abstain	0.229	1.000	0.000	0.000	0.000	0.229

Table 11: Results of tests on the car data set with cp as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	1.000	1.000	1.000	1.000	0.994	1.000
simple scaled 1:5	0.988	0.852	0.975	0.963	0.975	0.981
simple scaled 1:20	0.952	0.824	0.952	0.945	0.927	0.952
simple scaled 1:100	0.955	0.932	0.932	0.940	0.950	0.950
asymmetric loss 50	0.999	0.999	0.999	0.999	1.000	1.000
asymmetric loss 200	0.998	0.864	0.971	1.000	0.997	0.997
asymmetric loss 1000	0.993	0.843	0.981	0.993	0.990	0.990
random loss	0.995	1.000	1.000	1.000	0.995	1.000
random loss scaled 1:5	0.970	0.994	0.958	1.000	0.970	0.970
random loss scaled 1:20	0.924	0.812	0.929	0.924	0.817	0.912
random loss scaled 1:100	0.860	0.878	0.878	0.878	0.835	0.860
random	0.990	1.000	1.000	1.000	1.000	0.993
random scaled 1:5	0.995	0.995	0.995	0.995	0.995	0.995
random scaled 1:20	0.995	0.993	0.993	1.000	0.995	0.988
random scaled 1:100	0.982	0.990	1.000	1.000	0.982	0.982
asymmetric 50 abstain	0.986	0.024	0.020	0.020	0.020	0.020
asymmetric 200 abstain	0.003	0.992	0.992	0.992	0.992	0.992
asymmetric 1000 abstain	0.007	0.991	0.999	0.999	0.995	0.995

Table 12: Results of tests on the car data set with rf as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.998	0.990	0.990	0.992	0.995	0.998
simple scaled 1:5	0.998	0.993	0.993	1.000	0.042	0.998
simple scaled 1:20	1.000	1.000	1.000	1.000	0.031	1.000
simple scaled 1:100	1.000	1.000	1.000	1.000	0.029	1.000
asymmetric loss 50	0.999	0.990	0.990	0.994	0.997	0.999
asymmetric loss 200	0.907	0.925	0.925	0.953	0.863	0.911
asymmetric loss 1000	0.926	0.986	0.986	0.972	0.928	0.928
random loss	1.000	0.995	0.995	1.000	1.000	1.000
random loss scaled 1:5	0.916	0.933	0.933	0.916	0.961	0.911
random loss scaled 1:20	0.999	1.000	1.000	0.622	0.656	0.999
random loss scaled 1:100	1.000	1.000	1.000	0.602	1.000	1.000
random	0.995	0.975	0.975	0.999	0.999	0.999
random scaled 1:5	1.000	0.978	0.978	1.000	1.000	1.000
random scaled 1:20	0.995	0.984	0.984	0.988	1.000	1.000
random scaled 1:100	0.993	0.979	0.979	0.999	0.994	0.993
asymmetric 50 abstain	1.000	0.002	0.002	1.000	0.000	0.000
asymmetric 200 abstain	1.000	0.002	0.002	1.000	0.000	0.000
asymmetric 1000 abstain	1.000	0.005	0.005	0.999	1.000	0.001

Table 13: Results of tests on the `credit card` data set with `cp` as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.991	0.988	0.988	1.000	0.991	0.991
simple scaled 1:5	0.949	0.996	0.996	0.996	0.971	0.949
simple scaled 1:20	0.996	0.996	0.996	0.996	0.988	0.988
simple scaled 1:100	0.999	0.999	0.999	0.999	0.999	0.999
asymmetric loss 50	1.000	0.995	0.995	1.000	1.000	0.998
asymmetric loss 200	0.989	0.989	0.989	0.989	0.989	0.964
asymmetric loss 1000	0.965	0.962	0.962	0.965	0.957	0.962
random loss	1.000	0.990	0.990	1.000	1.000	0.998
random loss scaled 1:5	0.995	0.990	0.990	0.990	0.990	0.978
random loss scaled 1:20	1.000	1.000	1.000	1.000	1.000	1.000
random loss scaled 1:100	0.998	0.999	0.999	0.999	0.951	0.998
random	0.987	0.999	0.999	0.997	0.997	0.997
random scaled 1:5	0.996	0.987	0.987	1.000	0.996	1.000
random scaled 1:20	1.000	0.988	0.988	1.000	1.000	0.997
random scaled 1:100	0.988	0.995	0.995	0.988	0.988	0.994
asymmetric 50 abstain	0.999	0.001	0.001	0.000	0.000	0.000
asymmetric 200 abstain	0.999	0.003	0.003	0.000	0.000	0.000
asymmetric 1000 abstain	0.999	0.007	0.007	0.000	0.000	0.999

Table 14: Results of tests on the `credit card` data set with `rf` as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.989	0.948	0.989	0.989	0.989	0.989
simple scaled 1:5	0.999	0.972	0.989	0.897	0.971	0.897
simple scaled 1:20	1.000	0.977	0.924	0.966	0.971	0.443
simple scaled 1:100	0.969	0.990	0.990	0.991	0.997	0.981
asymmetric loss 50	0.980	0.956	0.979	0.980	0.980	0.980
asymmetric loss 200	0.987	0.950	0.994	0.987	0.987	0.987
asymmetric loss 1000	0.992	0.945	0.994	0.992	0.992	0.992
random loss	0.995	0.942	0.987	0.995	0.995	0.995
random loss scaled 1:5	0.981	0.963	0.993	0.958	0.991	0.958
random loss scaled 1:20	0.990	0.977	0.985	0.992	0.968	0.773
random loss scaled 1:100	0.948	0.978	0.947	0.978	0.966	0.942
random	0.998	0.955	0.991	0.998	0.998	0.998
random scaled 1:5	0.990	0.956	0.995	0.990	1.000	0.990
random scaled 1:20	0.993	0.954	0.989	0.992	1.000	0.993
random scaled 1:100	0.995	0.962	0.994	0.995	0.995	0.995
asymmetric 50 abstain	1.000	0.005	0.001	1.000	0.000	0.998
asymmetric 200 abstain	1.000	0.001	0.005	1.000	0.004	0.997
asymmetric 1000 abstain	1.000	0.696	0.802	1.000	0.824	0.996

Table 15: Results of tests on the usps data set with cp as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.997	1.000	1.000	0.998	0.997	0.997
simple scaled 1:5	0.999	0.999	0.999	0.999	0.999	0.999
simple scaled 1:20	0.983	0.973	0.929	1.000	1.000	0.983
simple scaled 1:100	1.000	0.963	0.963	1.000	0.993	1.000
asymmetric loss 50	0.996	0.998	0.998	0.993	0.996	0.996
asymmetric loss 200	0.999	1.000	1.000	0.997	0.999	1.000
asymmetric loss 1000	0.999	1.000	1.000	0.994	0.999	0.994
random loss	1.000	1.000	1.000	1.000	1.000	1.000
random loss scaled 1:5	0.986	0.994	0.994	0.976	0.978	0.984
random loss scaled 1:20	0.981	0.988	0.976	0.988	0.981	0.988
random loss scaled 1:100	0.895	0.929	0.929	1.000	0.883	0.890
random	0.996	1.000	1.000	0.994	0.993	0.996
random scaled 1:5	0.997	0.998	0.998	1.000	0.997	0.997
random scaled 1:20	0.993	0.994	0.996	0.998	0.993	0.993
random scaled 1:100	0.991	0.999	0.999	0.992	0.992	0.997
asymmetric 50 abstain	0.998	0.000	0.000	0.000	0.000	0.000
asymmetric 200 abstain	0.997	0.008	0.008	0.005	0.005	0.009
asymmetric 1000 abstain	0.996	0.909	0.909	0.885	0.885	0.921

Table 16: Results of tests on the usps data set with rf as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.944	0.948	0.948	0.948	0.958	0.962
simple scaled 1:5	0.997	0.983	0.983	0.815	0.988	0.003
simple scaled 1:20	0.962	0.984	0.984	0.962	0.996	0.001
simple scaled 1:100	0.996	0.984	0.984	0.996	0.996	0.000
asymmetric loss 50	0.917	0.917	0.808	0.917	0.917	0.917
asymmetric loss 200	0.913	0.908	0.900	0.915	0.920	0.927
asymmetric loss 1000	0.944	0.942	0.940	0.937	0.952	0.962
random loss	0.995	0.995	0.995	0.995	0.995	0.995
random loss scaled 1:5	0.999	0.978	0.957	0.978	0.990	0.007
random loss scaled 1:20	0.914	0.994	0.994	0.986	0.999	0.999
random loss scaled 1:100	0.985	0.993	0.993	0.985	0.997	0.000
random	0.942	0.942	0.902	0.902	0.937	0.565
random scaled 1:5	0.954	0.978	0.947	0.956	0.972	0.627
random scaled 1:20	0.964	0.979	0.958	0.970	0.980	0.707
random scaled 1:100	0.974	0.957	0.934	0.943	0.982	0.592
asymmetric 50 abstain	1.000	0.452	0.462	0.504	0.491	0.493
asymmetric 200 abstain	1.000	0.518	0.525	0.561	0.560	0.561
asymmetric 1000 abstain	1.000	0.921	0.913	0.979	0.979	0.977

Table 17: Results of tests on the `wine` data set with `cp` as scoring classifier.

	bare	SVR C1	SVR C100	GP [1, 2]	GP [1e-1, 1]	GP [1e-3, 1]
simple	0.929	0.914	0.914	0.936	0.936	0.936
simple scaled 1:5	0.960	0.961	0.961	0.961	0.961	0.960
simple scaled 1:20	0.993	0.990	0.990	0.990	0.997	0.993
simple scaled 1:100	1.000	1.000	1.000	0.996	0.999	0.999
asymmetric loss 50	1.000	0.999	0.999	0.989	0.989	0.989
asymmetric loss 200	0.970	0.972	0.972	0.994	0.972	0.968
asymmetric loss 1000	0.957	0.969	0.969	0.974	0.974	0.974
random loss	1.000	0.991	0.991	1.000	1.000	1.000
random loss scaled 1:5	0.987	1.000	1.000	0.968	0.968	0.968
random loss scaled 1:20	0.996	0.981	0.981	0.982	0.982	0.990
random loss scaled 1:100	0.997	0.998	0.998	1.000	1.000	0.996
random	0.918	0.918	0.918	0.920	0.918	0.923
random scaled 1:5	0.976	0.978	0.978	0.990	0.976	0.962
random scaled 1:20	0.971	0.961	0.961	0.959	0.959	0.959
random scaled 1:100	0.945	0.961	0.961	0.961	0.961	0.984
asymmetric 50 abstain	0.997	0.019	0.019	0.019	0.019	0.000
asymmetric 200 abstain	0.997	0.022	0.022	0.022	0.022	0.997
asymmetric 1000 abstain	0.998	0.082	0.082	0.082	0.082	0.998

Table 18: Results of tests on the `wine` data set with `rf` as scoring classifier.

C. Plots from the usps data set with cp as scoring classifier

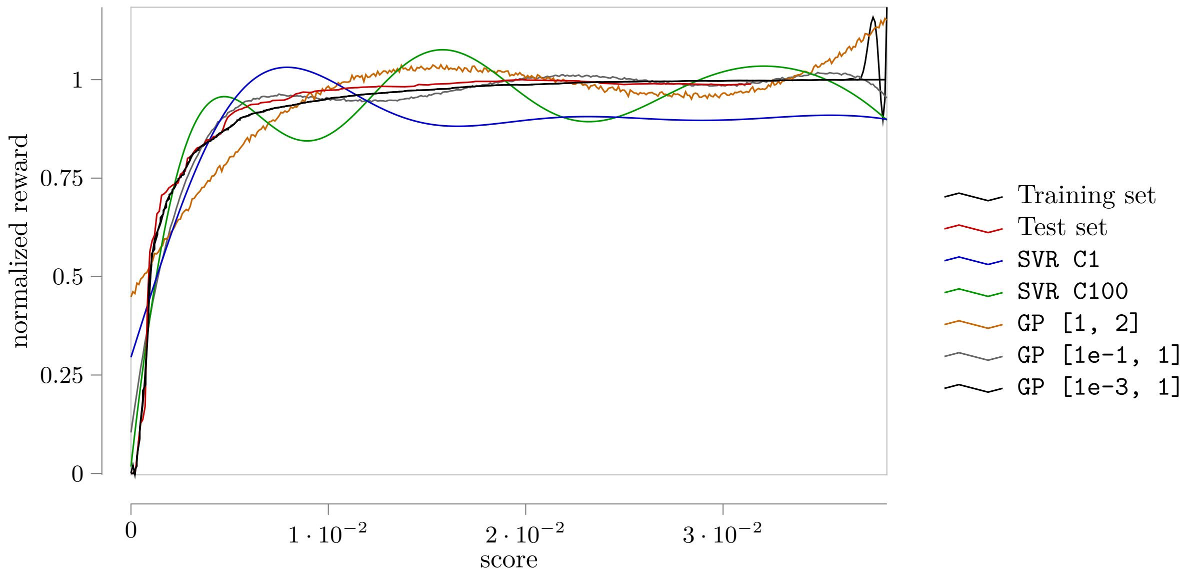

Figure 2: Results from the `simple` reward function.

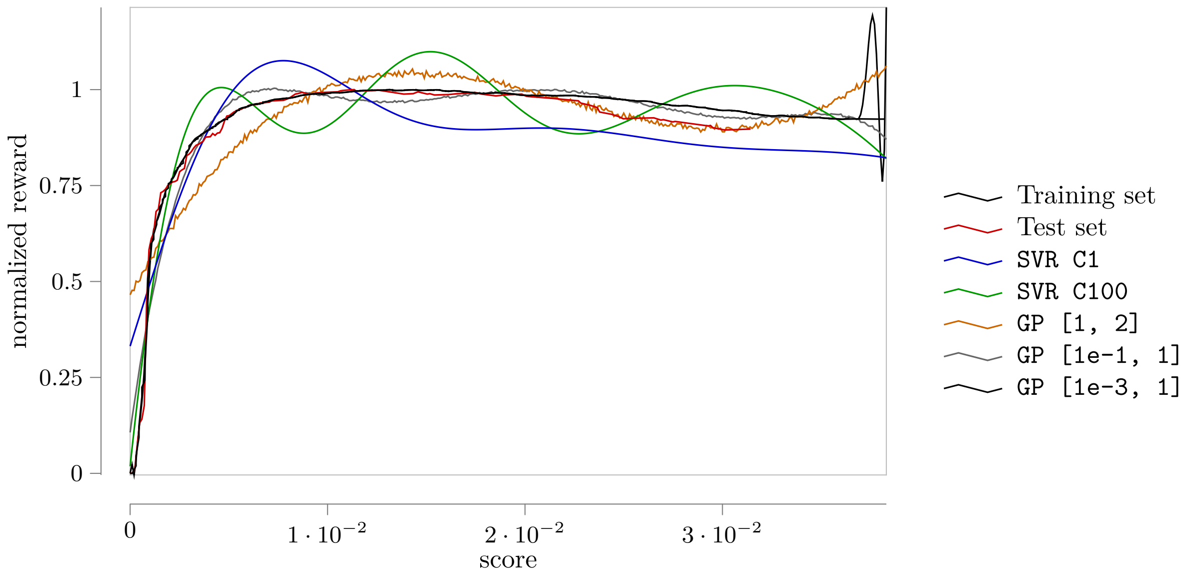

Figure 3: Results from the `simple scaled 1:5` reward function.

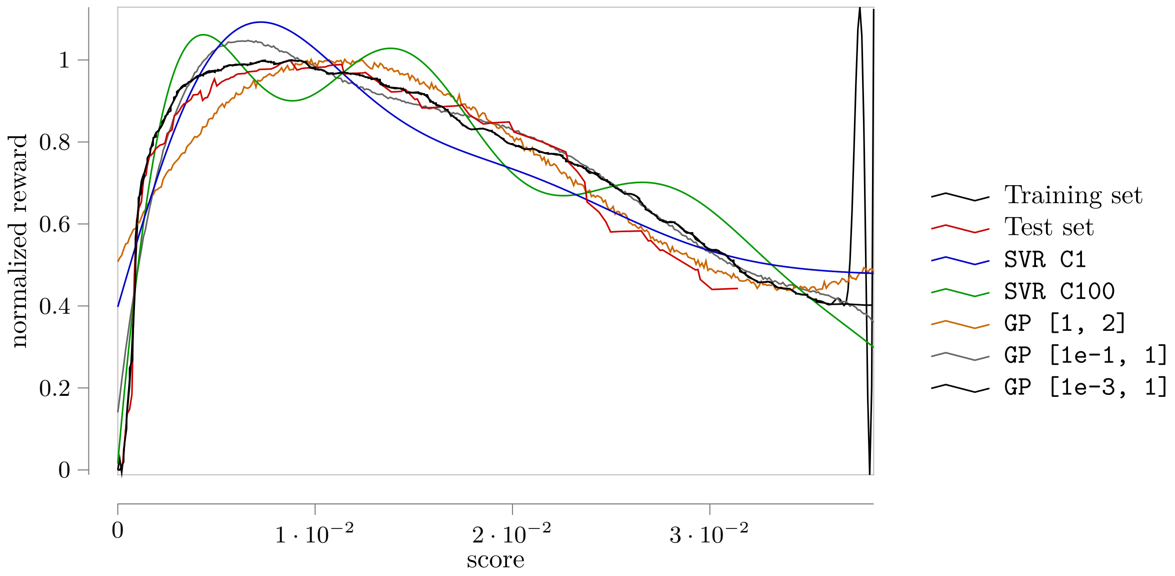

Figure 4: Results from the `simple scaled 1:20` reward function.

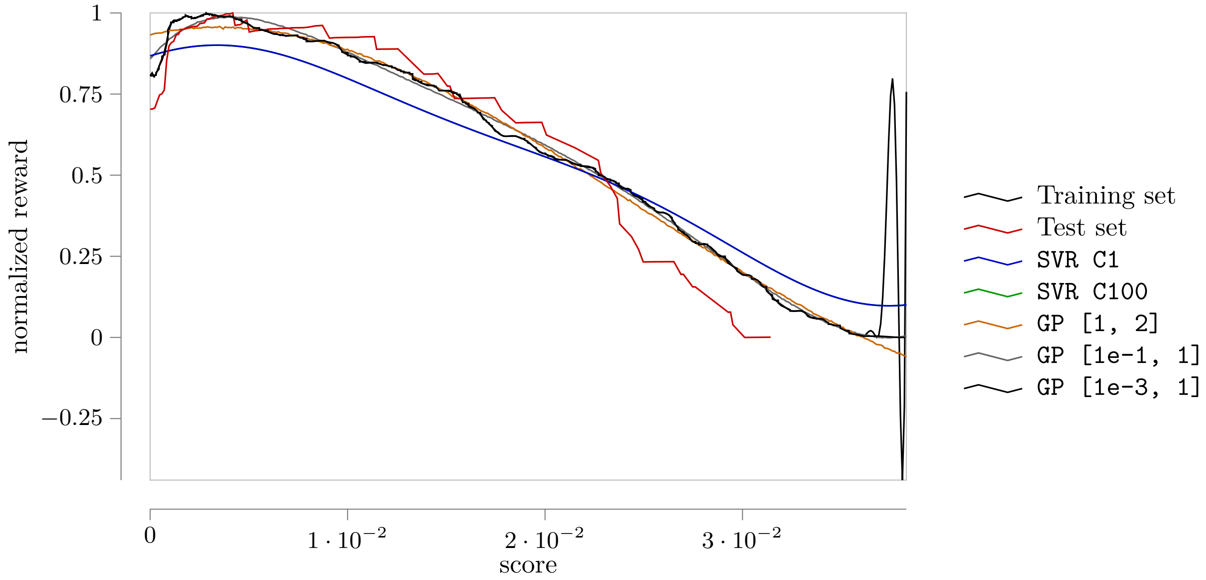

Figure 5: Results from the `simple scaled 1:100` reward function.

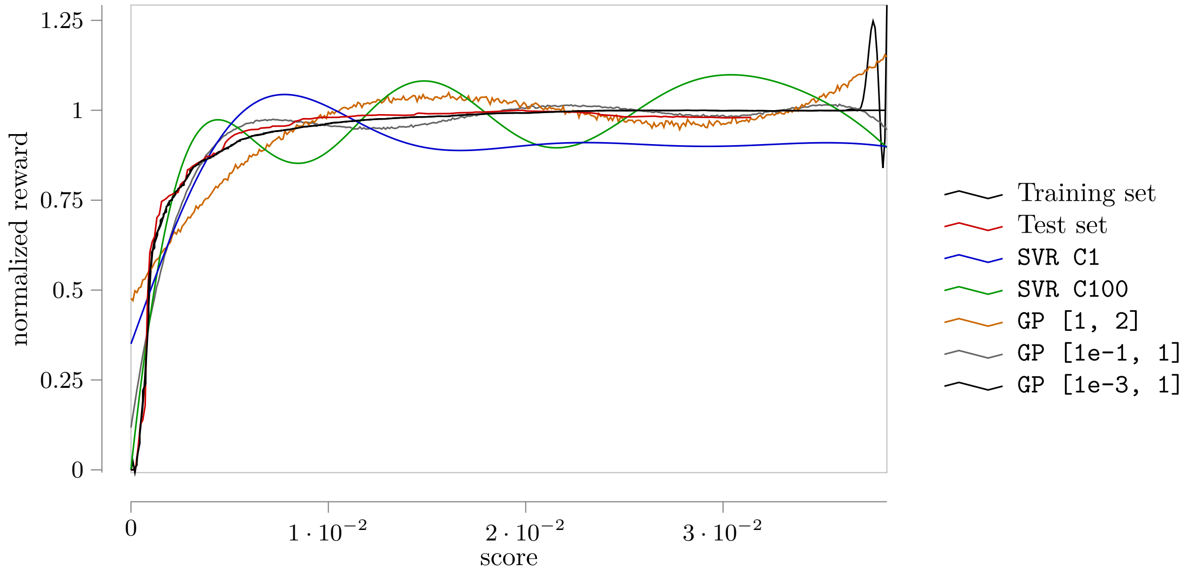

Figure 6: Results from the `asymmetric loss 50` reward function.

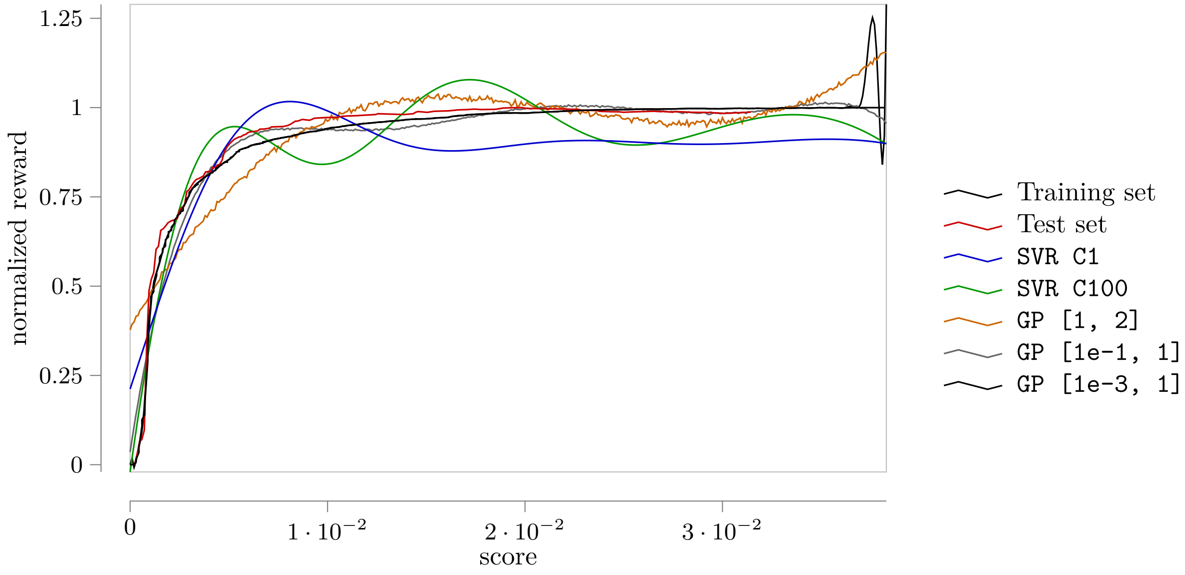

Figure 7: Results from the `asymmetric loss 200` reward function.

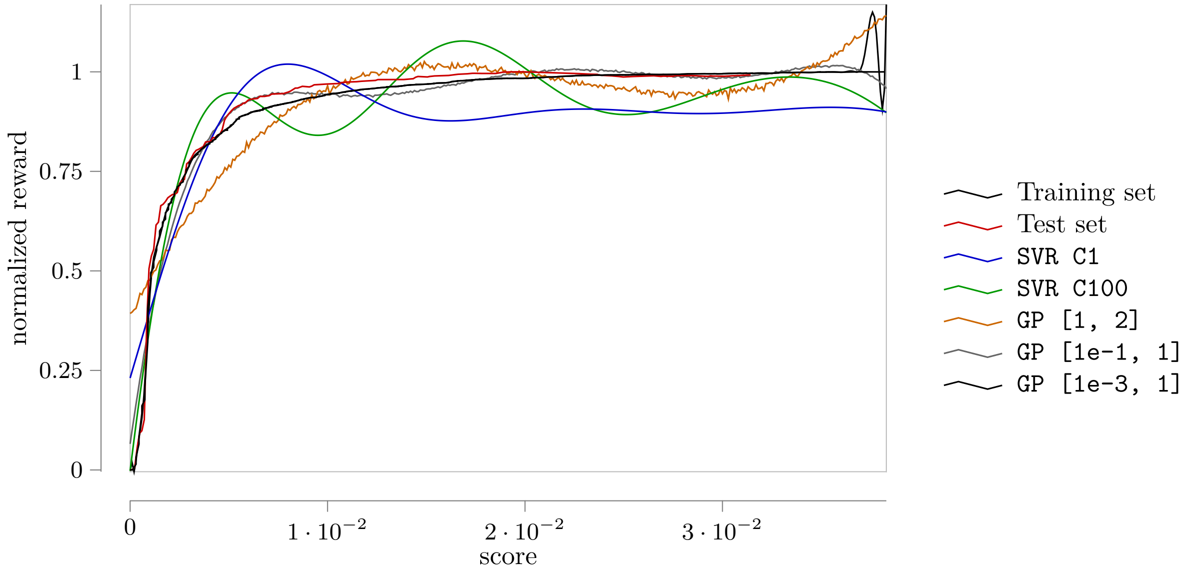

Figure 8: Results from the `asymmetric loss 1000` reward function.

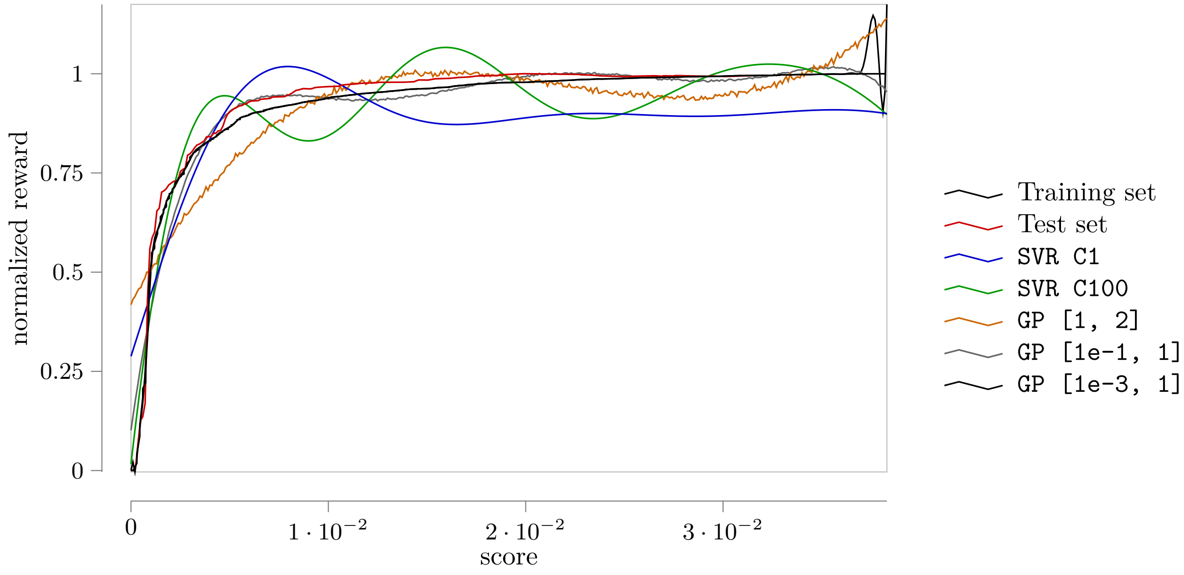

Figure 9: Results from the `random loss` reward function.

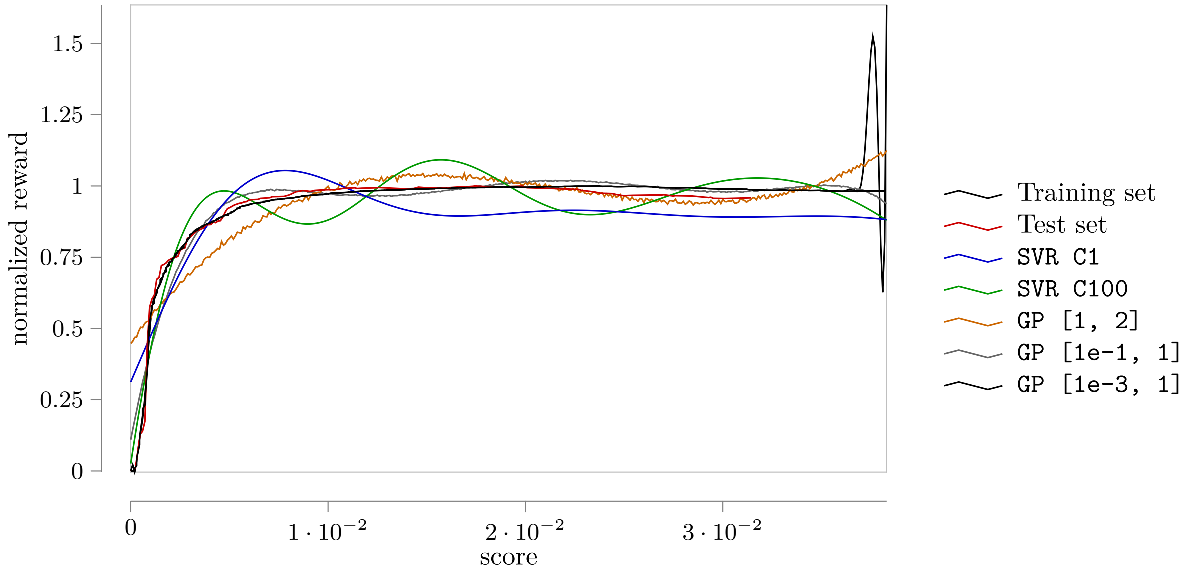

Figure 10: Results from the `random loss scaled 1:5` reward function.

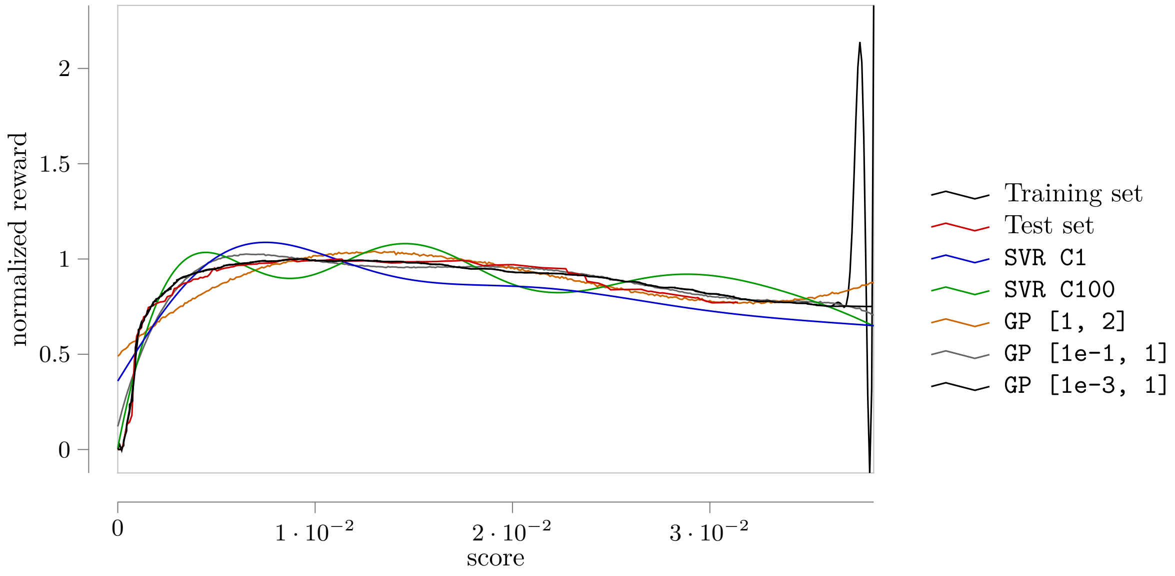

Figure 11: Results from the `random loss scaled 1:20` reward function.

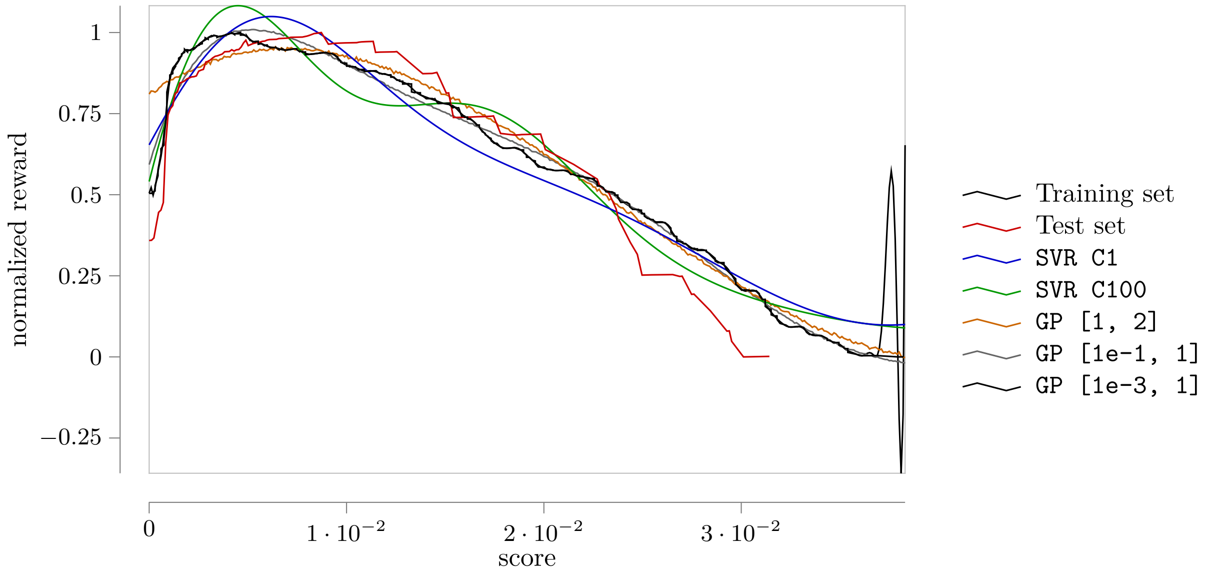

Figure 12: Results from the `random loss scaled 1:100` reward function.

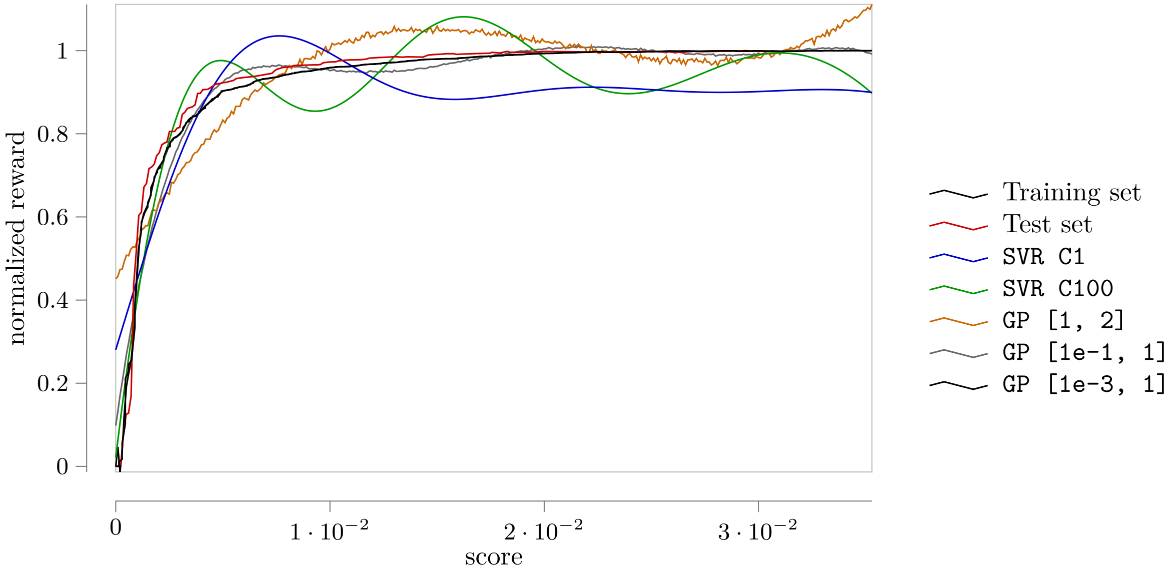

Figure 13: Results from the `random` reward function.

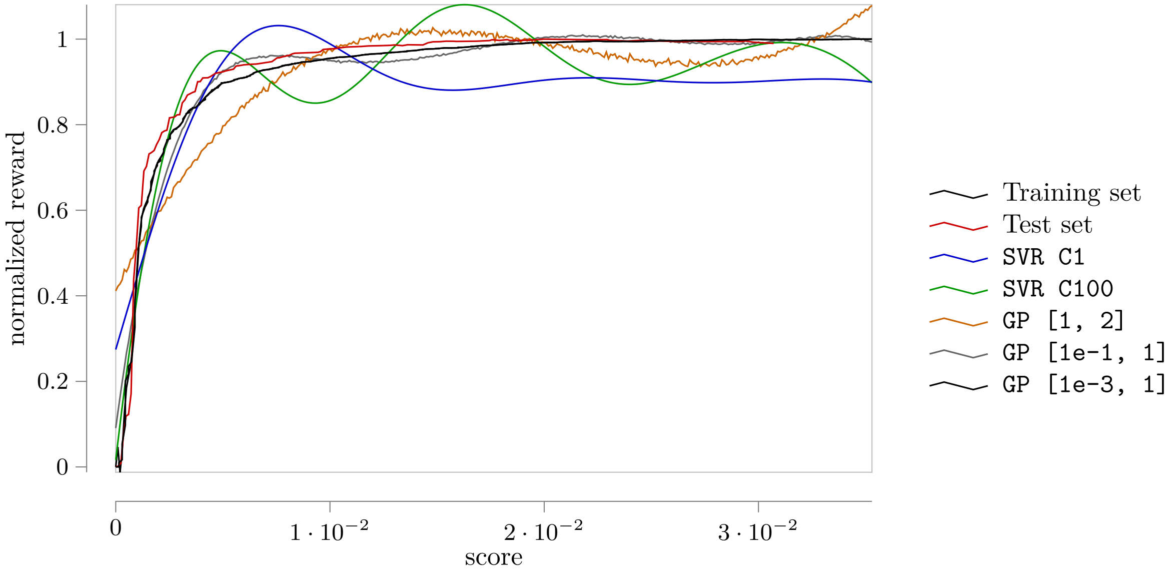

Figure 14: Results from the `random scaled 1:5` reward function.

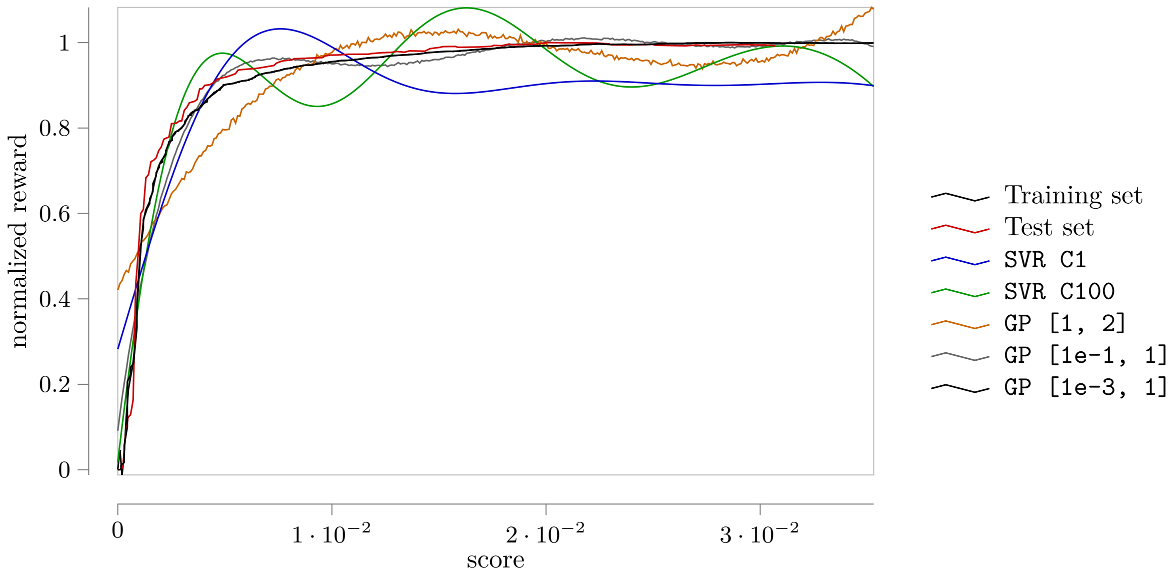

Figure 15: Results from the `random scaled 1:20` reward function.

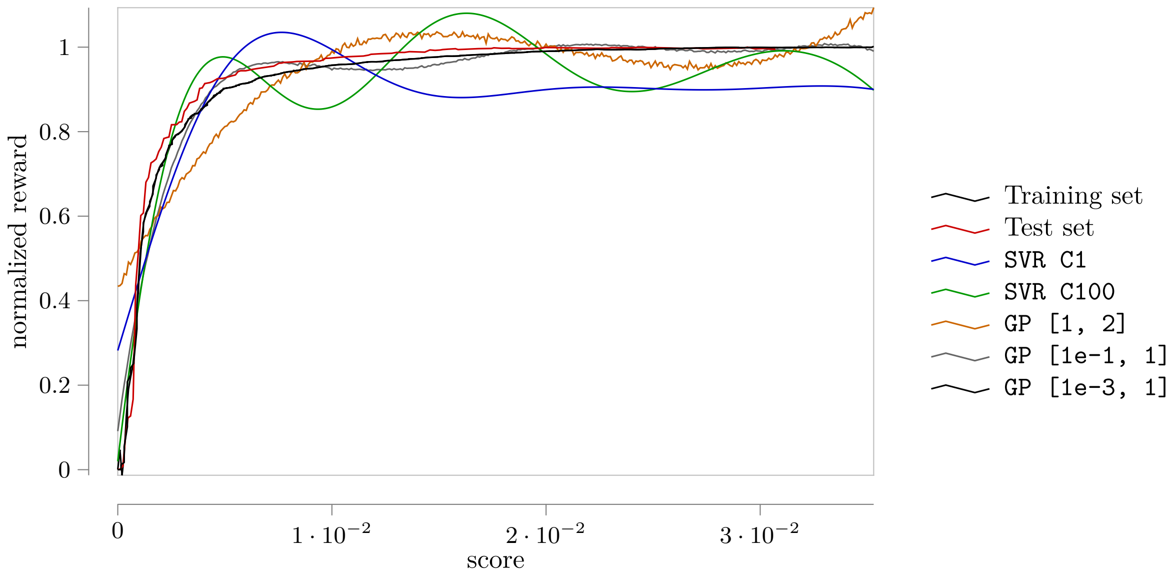

Figure 16: Results from the `random scaled 1:100` reward function.

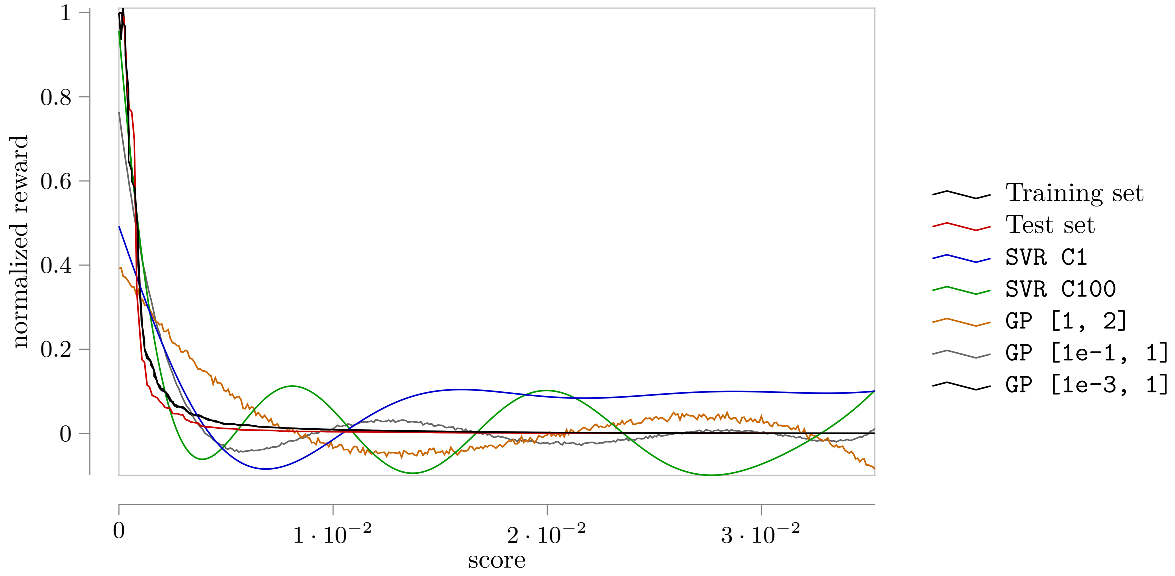

Figure 17: Results from the `asymmetric 50 abstain` reward function.

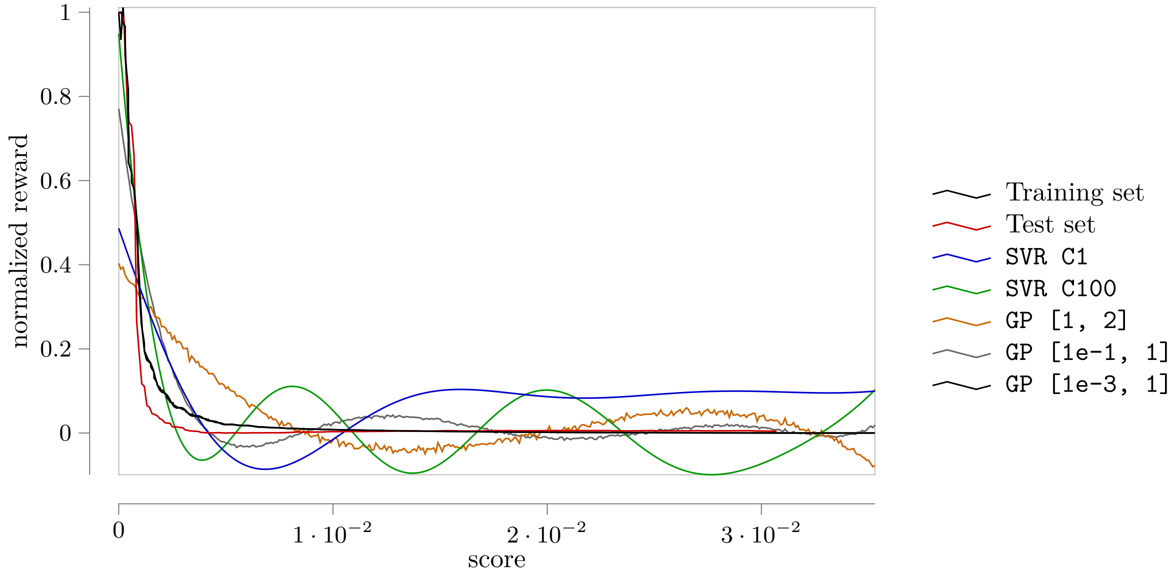

Figure 18: Results from the `asymmetric 200 abstain` reward function.

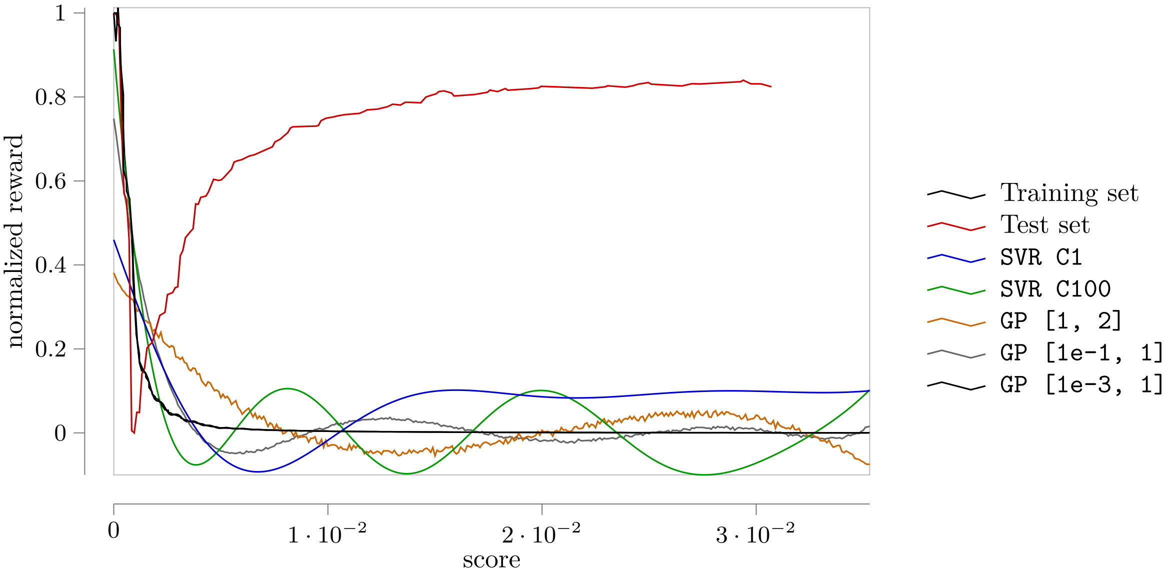

Figure 19: Results from the `asymmetric 1000 abstain` reward function.

D. Asymmetric reward matrices

$$M^{50} := \begin{pmatrix} 37 & 13 & 18 & 13 & 16 & 46 & 43 & 48 & 4 & 39 & 32 \\ 38 & 17 & 40 & 38 & 47 & 0 & 31 & 30 & 44 & 5 & 23 \\ 29 & 50 & 20 & 31 & 48 & 50 & 18 & 11 & 28 & 37 & 10 \\ 20 & 37 & 40 & 31 & 23 & 20 & 13 & 27 & 34 & 34 & 11 \\ 41 & 42 & 37 & 37 & 38 & 40 & 45 & 39 & 24 & 1 & 20 \\ 1 & 36 & 9 & 40 & 8 & 4 & 12 & 6 & 10 & 49 & 50 \\ 10 & 38 & 21 & 34 & 36 & 6 & 43 & 7 & 36 & 22 & 44 \\ 50 & 38 & 6 & 18 & 15 & 39 & 13 & 23 & 43 & 7 & 5 \\ 45 & 24 & 28 & 49 & 11 & 5 & 22 & 31 & 11 & 34 & 9 \\ 50 & 22 & 11 & 25 & 3 & 27 & 25 & 18 & 30 & 1 & 12 \\ 14 & 44 & 37 & 1 & 34 & 17 & 46 & 30 & 50 & 7 & 43 \end{pmatrix}$$

$$M^{200} := \begin{pmatrix} 59 & 199 & 95 & 128 & 126 & 122 & 29 & 104 & 12 & 164 & 200 \\ 81 & 28 & 182 & 40 & 60 & 156 & 49 & 64 & 199 & 180 & 127 \\ 195 & 177 & 200 & 10 & 188 & 198 & 105 & 34 & 48 & 175 & 3 \\ 142 & 96 & 33 & 92 & 80 & 138 & 27 & 199 & 122 & 150 & 56 \\ 186 & 69 & 176 & 143 & 159 & 83 & 132 & 79 & 37 & 26 & 72 \\ 156 & 110 & 59 & 194 & 195 & 94 & 66 & 171 & 180 & 3 & 185 \\ 198 & 151 & 182 & 3 & 156 & 155 & 184 & 187 & 94 & 74 & 44 \\ 156 & 21 & 200 & 47 & 112 & 10 & 112 & 32 & 28 & 125 & 122 \\ 179 & 44 & 147 & 106 & 61 & 80 & 194 & 81 & 78 & 124 & 69 \\ 66 & 23 & 150 & 102 & 11 & 95 & 38 & 150 & 75 & 121 & 7 \\ 137 & 36 & 84 & 119 & 120 & 85 & 117 & 47 & 60 & 96 & 9 \end{pmatrix}$$

$$M^{1000} := \begin{pmatrix} 306 & 977 & 476 & 230 & 744 & 481 & 708 & 701 & 598 & 30 & 954 \\ 637 & 369 & 9 & 685 & 736 & 431 & 288 & 262 & 95 & 394 & 213 \\ 503 & 678 & 878 & 279 & 196 & 100 & 178 & 791 & 636 & 398 & 634 \\ 346 & 667 & 2 & 294 & 172 & 260 & 412 & 211 & 832 & 675 & 756 \\ 278 & 396 & 516 & 674 & 769 & 81 & 422 & 764 & 574 & 341 & 195 \\ 266 & 368 & 983 & 513 & 653 & 537 & 621 & 141 & 512 & 778 & 95 \\ 363 & 716 & 409 & 366 & 96 & 951 & 782 & 573 & 125 & 633 & 623 \\ 922 & 184 & 708 & 153 & 511 & 649 & 621 & 30 & 391 & 795 & 124 \\ 444 & 59 & 123 & 684 & 794 & 223 & 296 & 875 & 506 & 948 & 104 \\ 432 & 123 & 693 & 742 & 541 & 294 & 12 & 1 & 242 & 233 & 104 \\ 774 & 656 & 312 & 996 & 956 & 346 & 603 & 438 & 232 & 504 & 745 \end{pmatrix}$$

References

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *CoRR*, abs/1708.05866, 2017. URL http://arxiv.org/abs/1708.05866.

Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, first edition, 2014. ISBN 0123985374, 9780123985378.

Marko Bohanec and Vladislav Rajkovič. V.: Knowledge acquisition and explanation for multi-attribute decision. In *Making, 8 th International Workshop "Expert Systems and Their Applications*, 1988.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013.

Aaron Clauset. Inference, models and simulation for complex systems: A brief primer on probability distributions. 08 2011. URL http://tuvalu.santafe.edu/~aaronc/courses/7000/csci7000-001_2011_L0.pdf.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547 – 553, 2009. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2009.05.016. URL http://www.sciencedirect.com/science/article/pii/S0167923609001377. Smart Business Networks: Concepts and Empirical Evidence.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Jonas Fassbender. libconform v0.1.0: a python library for conformal prediction. *arXiv preprint arXiv:1907.02015*, 2019. URL https://arxiv.org/abs/1907.02015.

Lydia Fischer, Barbara Hammer, and Heiko Wersing. Optimal local rejection for classifiers. *Neurocomputing*, 214:445 – 457, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.06.038. URL http://www.sciencedirect.com/science/article/pii/S0925231216306762.

Caroline Friedel. On abstaining classifiers. 01 2005.

Caroline C. Friedel, Ulrich Rückert, and Stefan Kramer. Cost curves for abstaining classifiers. In *Proceedings of the ICML 2006 workshop on ROC Analysis in Machine Learning*, 2006.

Hongjiao Guan, Yingtao Zhang, Heng-Da Cheng, and Xianglong Tang. Abstaining classification when error costs are unequal and unknown. *CoRR*, abs/1806.03445, 2018. URL `http://arxiv.org/abs/1806.03445`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–2019. URL `http://www.scipy.org/`. [Online; accessed 14.06.2019].

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL `http://dx.doi.org/10.1162/neco.1989.1.4.541`.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22 – 31, 2014. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2014.03.001. URL `http://www.sciencedirect.com/science/article/pii/S016792361400061X`.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Tadeusz Pietraszek. Optimizing abstaining classifiers using roc analysis. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 665–672, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102435. URL `http://doi.acm.org/10.1145/1102351.1102435`.

Evgueni Smirnov, Georgi Nalbantovi, and A. M. Kaptein. Meta-conformity approach to reliable classification. *Intelligent Data Analysis*, 13, 01 2009. doi: 10.3233/IDA-2009-0400.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

Francesco Tortorella. An optimal reject rule for binary classifiers. pages 611–620, 08 2000. doi: 10.1007/3-540-44522-6_63.

Peter D. Turney. Types of cost in inductive concept learning. *CoRR*, cs.LG/0212034, 2002. URL `http://arxiv.org/abs/cs.LG/0212034`.

Guido van Rossum and the Python development team. *The Python Library Reference*. Python Software Foundation, 3.6th edition, 2019.

Stijn Vanderlooy, Ida G. Sprinkhuizen-Kuyper, Evgueni N. Smirnov, and H. Jaap van den Herik. The roc isometrics approach to construct reliable classifiers. *Intell. Data Anal.*, 13(1):3–37, January 2009. ISSN 1088-467X. URL http://dl.acm.org/citation.cfm?id=1551758.1551760.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

I-Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473 – 2480, 2009. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2007.12.020. URL http://www.sciencedirect.com/science/article/pii/S0957417407006719.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997. ISSN 0098-3500. doi: 10.1145/279232.279236. URL http://doi.acm.org/10.1145/279232.279236.

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum

Rechtsverbindliche Unterschrift