# Optimum reject options for multiclass classification

Johannes Kummert

*October 19, 2015*

Bachelor Thesis

# Optimum reject options for multiclass classification

Johannes Kummert

*1. Reviewer* Prof. Dr. Barbara Hammer
Theoretical Computer Science
Bielefeld University

*2. Reviewer* Benjamin Paaßen
Theoretical Computer Science
Bielefeld University

*Supervisors* Prof. Dr. Barbara Hammer and Benjamin Paaßen

October 19, 2015

**Johannes Kummert**

*Optimum reject options for multiclass classification*

Bachelor Thesis, October 19, 2015

Reviewers: Prof. Dr. Barbara Hammer and Benjamin Paaßen

Supervisors: Prof. Dr. Barbara Hammer and Benjamin Paaßen

# Abstract

# Contents

# 1 Introduction

## 1.1 Scope

In 2008 Nature wrote that Google was "[g]oing from a collection of donated servers housed under a desk to a global network of dedicated data centres processing information by the petabyte" (TODO:cite) and former Google CEO Eric Schmidt said himself that "There were 5 exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days." Since then countless more companies are specializing in mining and, more importantly, analyzing large heaps of data, the phrase "Big Data" is prevalent. Machine Learning provides central techniques for this analysis and classification is one of its basic applications [Alp10, p. 5]. Big Data demands new approaches to classification and has different requirements in quality than just the accuracy. Life long learning [Sut14, p. 3] and real-time capability, one of the design principles of Industry 4.0 [MH15, p. 12], call for more flexible and adaptable models. One way is to provide a classification confidence [SJDZ05] to add more information than just a class label. Based on that a classification model can be easily changed to account for shifting demands, e.g. only accept points with high confidence in critical applications. Thus reject options and error handling are becoming more important. We aim to find a method that provides optimal reject options based on classification confidence.

## 1.2 Related Work

First we want to look at two different state of the art methods for reject options and explain how we base our work on them and where we diverge.
There are several methods that build upon the knowledge of the probability density of the classes as proposed in [Cho70]. This approach commonly yields a global reject threshold hence the assumption of similar class structures is made. While we consider the possibility of a global reject we will decide on local thresholds (see chapter 2.2.2). Furthermore we allow any real valued confidence measure not just probabilities.
Approach [SB13] detects outliers within a class by comparing the distance of a point to other points in his class, thus introducing a confidence measure that can be computed within a class and does not take intuitive measures into account, like

the distance to the decision hyperplane. While the examples in this thesis focus on conventional confidence measures, in theory this is an applicable way to formulate confidence that works with the methods proposed below.

## 1.3 Focus

The focus of this work will be to formulate a reject strategy that finds optimal thresholds regardless of the structure of the data and classes. This strategy will be largely based on concepts introduced in the paper [LF15]. In order to expand on these concepts we try to find a more efficient solution by reducing the number of considered thresholds, thus leading to a faster computation time while maintaining optimality.

## 1.4 Structure

We will first work on formulating a reject strategy and a definition of an optimal reject in the context of a two class classifier. Then we will expand on that to develop an optimal local reject option for multiple classes and find ways to compute this option efficiently. Based on a brute force approach we will define a dynamic programming scheme that works in polynomial runtime. Subsequently we devise a greedy strategy and evaluate whether this approach can compete with optimal solutions.

# 2 Reject Options

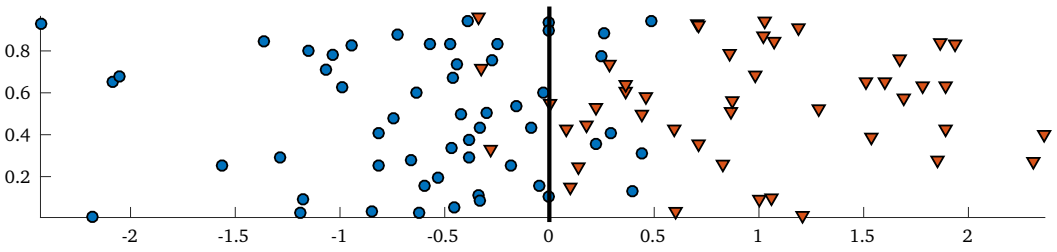
## 2.1 Two Classes

To make our way towards optimal rejects for multi class classification, we start of small by looking at a general two class classifier $f$ that divides the space via a decision boundary.

$$f : \mathbb{R}^n \to \{1, 2\}$$

Let $r$ be a measure of confidence that a point is part of its respective class, e.g. distance to the decision boundary. If $r(\bar{x})$ is large it means that $\bar{x}$ is likely in the class it was assigned to.

**Fig. 2.1:** Example of two classes separated by a decision boundary. You can see falsely classified points near the boundary which we would like to reject.




### 2.1.1 Reject Strategy

We look for a threshold $\theta$ that defines for a given point $\bar{x}$ whether it is rejected or not:

$$r(\bar{x}) < \theta : \bar{x}\ rejected$$

### 2.1.2 Optimal $\theta$

In order to find an optimal threshold we need to decide how to evaluate it. Optimally only wrongly classified data points are rejected (now referred to as true rejects). But generally there will be rejected, although correctly classified, points (false rejects).

A given labeled data set $X = \{\bar{x}_1, ..., \bar{x}_n\}$ (by $y(\bar{x}) \mapsto \{1, 2\}$) is divided into the two sets $L$ and $E$ by applying $f$ as a classifier with:

$$L = \{\forall \bar{x} \in X \mid f(\bar{x}) = y(\bar{x})\} : \text{correctly classified points} \tag{2.1}$$

$$E = \{\forall \bar{x} \in X \mid f(\bar{x}) \neq y(\bar{x})\} : \text{incorrectly classified points} \tag{2.2}$$

And by applying our reject strategy with the threshold $\theta$, $X$ is divided into $A$ and $R$ with:

$$A_\theta = \{\forall \bar{x} \in X \mid r(\bar{x}) \geq \theta\} : \text{accepted points} \tag{2.3}$$

$$R_\theta = \{\forall \bar{x} \in X \mid r(\bar{x}) < \theta\} : \text{rejected points} \tag{2.4}$$

The set $T$ contains all true rejects and $F$ all false rejects.

$$T_\theta = R \cap E \tag{2.5}$$

$$F_\theta = R \cap L \tag{2.6}$$

Naturally we want $|T_\theta|$ to be large and $|F_\theta|$ to be small. Since these two goals often contradict each other, e.g. more true rejects most times bring more false ones (see figure 2.2), in general there is no single optimal choice of $\theta$. Still there is a set of values that we consider optimal. As shown above each $\theta$ corresponds to a tuple $(|T_\theta|, |F_\theta|)$. $\theta$ is optimal if

$$\nexists \theta' : |F'_\theta| \leq |F_\theta|, |T'_\theta| \geq |T_\theta|$$

and at least one inequality holds, thus being Pareto optimal.

### 2.1.3 Finding $\theta$

Now that we know how to evaluate our thresholds we still need to know where to look for them. Obviously $\theta$ needs to be in the range of $r$ so

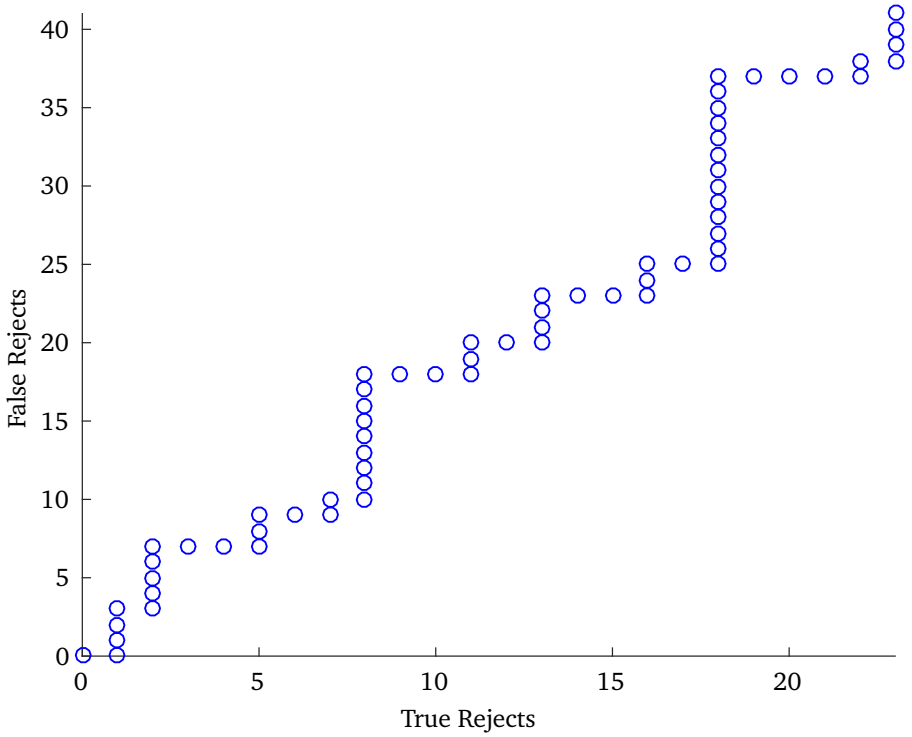
$$\Theta := \{\theta \in \mathbb{R} \mid \theta < \max_{\bar{x}} r(\bar{x})\}$$

is the set of possible thresholds. Because this set is infinitely large we need to reduce the its size (TODO:wording). We recognize that it is sufficient to only consider the discreet set of the values of $r$. So with $\Theta := \{r(\bar{x}) | \forall \bar{x} \in X\}$ we have a feasible set of possible thresholds. It can be refined by looking again at our criteria for an optimal $\theta$. Let

$$\bar{r} = (r_1, ..., r_n) \mid r_i = r(\bar{x}_i) \forall \bar{x}_i \in X, r_j < r_{j+1} \forall j \in \{1, ..., n-1\}$$

be the vector of thresholds in $\Theta$ sorted by the measure of confidence in ascending order. Consider $\bar{r}(i)$ as threshold with $\bar{x}_i \in E$. We now consider different cases:

**Fig. 2.2:** In this example we can see the corresponding true and false rejects if $r(\bar{x})$ is chosen as a threshold for each $\bar{x} \in X$. $r(\bar{x})$ is increasing from the bottom left corner to the upper right. As we can see, the number of false and true rejects are monotonically increasing with a more strict threshold.




- $\bar{x}_{i-1} \in E$: $\bar{r}(i-1)$ cannot be a better choice since it would result in one less true reject.
- $\bar{x}_{i-1} \in L$: $\bar{r}(i-1)$ is a better threshold since it gives an additional true reject.
- $\bar{r}(i+1)$ cannot be a better threshold since it only adds a false reject.

In conclusion this means that it is sufficient to consider only points at the beginning of groups of false rejects, giving us

$$\Theta := \{r(\bar{x}_i) \mid \bar{x}_i \in L,\ \bar{x}_{i-1} \in E\}$$

as an easily computed set of thresholds to consider to be optimal (see figures 2.4 and 2.3.)

## 2.2 Multi class Classification

Let us now expand the problem to a classifier for $N$ multiple classes. By using a one vs all strategy, we get $N$ binary classifiers $f_k$ like in chapter 2.1 and an according

**Fig. 2.3:** This figure shows the thresholds from before (see figure 2.2). The green marked points correspond to threshold we consider optimal.

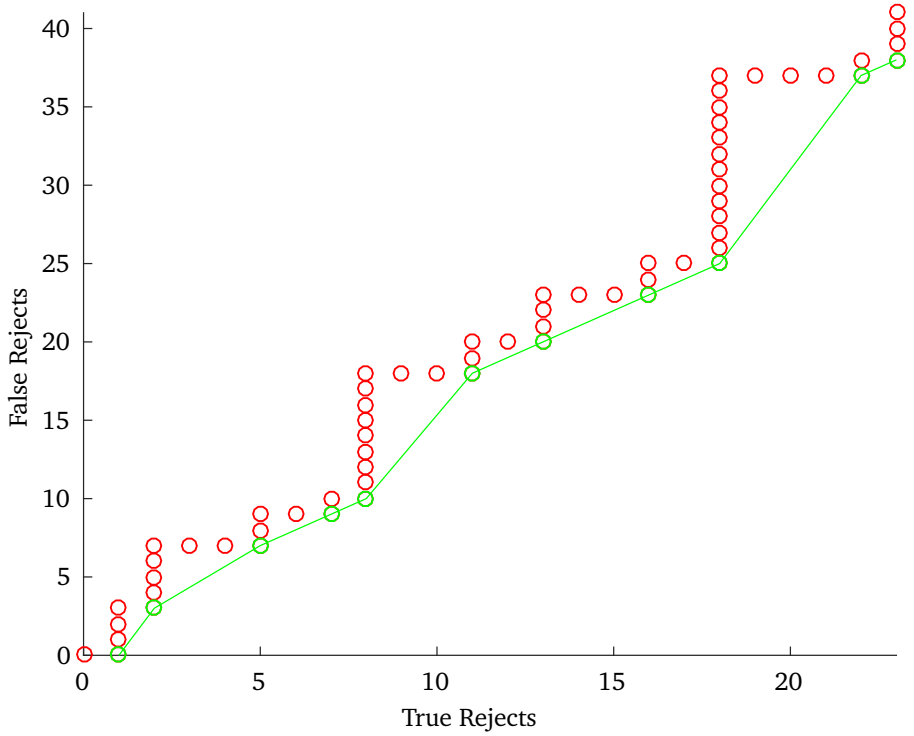
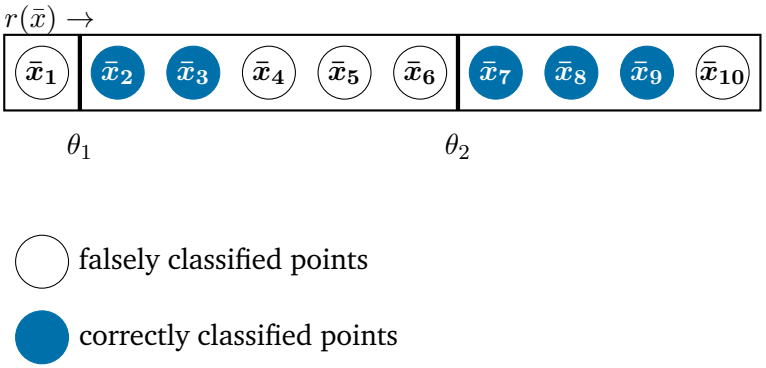



**Fig. 2.4:** This is an example of classified points sorted by a measure of confidence. You can see that $r(\bar{x}_2)$ and $r(\bar{x}_7)$ are the only sensible thresholds. Choosing $r(\bar{x}_8)$ instead of $r(\bar{x}_7)$ would only add a false reject and choosing $r(\bar{x}_6)$ over $r(\bar{x}_7)$ would result in one less true reject.



measure of confidence $r_k$. Points are now classified in the class where confidence is maximal among all classes. This gives us a multi class classifier $f$ with

$$f : \mathbb{R}^n \to \{1, ..., N\} \tag{2.7}$$

$$\bar{x} \mapsto \arg\max_k r_k(\bar{x}) \tag{2.8}$$

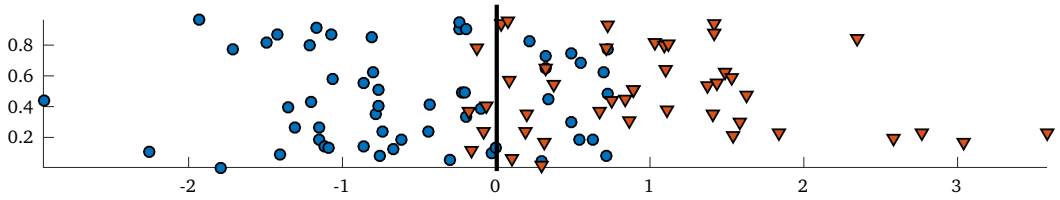
### 2.2.1 Global Reject

To adapt our reject strategy from before, we search for a threshold $\theta$ and reject according to our measures of confidence:

$$\max_k r_k(\bar{x}) < \theta \ : \ \bar{x} \text{ rejected}$$

We look again to choose $\theta$ so that it meets our requirements for an optimal threshold (see chapter 2.1.2). This strategy comes with the problem that it relies on all measures $r_k$ to be scaled the same way. While there might be a workaround for this, e.g. using probabilities as scale, a global reject is still imprecise when the internal structures of the classes differ a lot(see figure 2.5). A reasonable threshold for a class with most points and also most classification errors near the decision plane is probably not well suited for a class with a larger variance in its set of data points. This leads to the idea of having individual thresholds for each class.

**Fig. 2.5:** In this two-class classification example one class (blue circles) has few errors near the decision boundary and the other (red triangles) multiple further away. A global reject optimal for the first class would result in a lot of unrejected errors in class 2. Conversely an optimal global reject for class 2 would result in many false rejects in class 1.




### 2.2.2 Local Reject

To account for differences in class structure we define a local reject threshold $\theta_k$. A point is rejected if

$$r_k(\bar{x}) < \theta_k \mid \forall \bar{x} \text{ where } f(\bar{x}) = k$$

This gives us an N-dimensional threshold vector $\bar{\theta} = (\theta_1, ..., \theta_N)$. Each $\theta_k$ regulates the reject practice only in their respective class.

### 2.2.3 Optimal Local Reject

For each threshold $\theta_k$ the same criteria apply to determine if it is optimal as for a global threshold (see chapter 2.1.2). With the difference that we look at each class individually. So the true and false rejects are now given by

$$T^k_{\theta_k} = R^k_{\theta_k} \cap E_k \tag{2.9}$$

$$F^k_{\theta_k} = R^k_{\theta_k} \cap L_k, \tag{2.10}$$

$$\forall \theta_k \in \Theta_k := \{r_k(\bar{x}_i) \mid \bar{x}_i \in L_k,\ \bar{x}_{i-1} \in E_k\},\ \forall k \in \{1, ...N\} \tag{2.11}$$

where $R^k_{\theta_k}$ is the amount of rejected points in class $k$ given the threshold $\theta_k$, $E_k$ the set of falsely classified points in class $k$ and $L_k$ the set of correctly classified points in $k$. We conclude that the threshold vector $\bar{\theta} = (\theta_1, ..., \theta_N)$ is optimal if each $\theta_k$ is optimal. Conversely an optimal $\theta_k$ is not necessarily a component of an optimal vector $\bar{\theta}$.

### 2.2.4 Computation by Brute Force

To now find the optimal local reject vectors $\bar{\theta}$ with a brute force approach we need to consider every combination of thresholds in the sets $\Theta_k$. Let

$$\mathbb{P} = \left\{\bar{\theta} = (\theta_1, ..., \theta_N) \in \Theta_1 \times ... \times \Theta_N\right\}$$

be the set of all choices from the sets of optimal thresholds of each class. Algorithm 1 describes in pseudo code how to find all optimal $\bar{\theta}$.

**Data**: $\mathbb{P}$
**Result**: set of optimal local reject vectors $\Theta_{opt}$
initialization;
array $opt$ where $opt(n)$ is the maximum number of true rejects with n false ones ;
array $theta$ where $theta(n)$ is the optimal threshold vector with n false rejects ;
**foreach** $\bar{\theta} \in \mathbb{P}$ **do**
    $F :=$ amount of false rejects for $\bar{\theta}$ ;
    $T :=$ amount of true rejects for $\bar{\theta}$ ;
    **if** $opt(F) < T$ **then**
        $opt(F) := T$ ;
        $theta(F) = \bar{\theta}$ ;
    **end**
**end**
**return** *theta*

**Algorithm 1:** Computing optimal local reject options by brute force.

### 2.2.5 Computation by Dynamic Programming

Computation by brute force scales exponentially with the number of classes and is therefore not a feasible solution for large data sets with lots of classes. However our problem is equivalent to a multiple choice knapsack problem [PS79] where thresholds within a class correspond to items, false rejects correspond to costs, and true rejects correspond to their value. Thus a DP-Scheme is available that allows computation in polynomial time. Let

$$opt(n,j,i) = \max_{\bar{\theta}} \left\{ |T_{\bar{\theta}}| \;\; s.t. \begin{array}{ll} |F_{\bar{\theta}}| \leq & n \\ \theta_l \in & \Theta_l \forall l < j \\ \theta_j \in & \left\{ \bar{\theta}_j(0), ..., \bar{\theta}_j(i) \right\} \\ \theta_l \in & \bar{\theta}_l(0) \forall l > j \end{array} \right\}$$

with

$$n \in \{0, ..., |L|\} \tag{2.12}$$
$$j \in \{1, ..., N\} \tag{2.13}$$
$$i \in \{0, ..., |\Theta_j| - 1\} \tag{2.14}$$
$$\bar{\theta}_k = (\theta_1, ..., \theta_n) \mid \theta_i \in \Theta_k,\ \theta_j < \theta_{j+1} \forall j \in \{1, ..., n-1\} (TODO) \tag{2.15}$$

be the maximum number of true rejects with $n$ false rejects while considering all thresholds in classes before class $j$ and the first $i$ thresholds in class $j$. All $\bar{\theta}$ corresponding to $opt(n, N, |\Theta_N| - 1) \forall n < |L|$ fulfill our criteria for an optimal threshold vector.
This results in the Bellmann equation 2.16 [Bel57] to efficiently compute $opt$:

$$opt(n,j,i) = \begin{cases} \text{if } n = 0: & \sum_{k=1}^{N} \left| T^k_{\bar{\theta}_k(0)} \right| & \text{(2.16a)} \\ \text{if } n > 0,\ j = 0: & \sum_{k=1}^{N} \left| T^k_{\bar{\theta}_k(0)} \right| & \text{(2.16b)} \\ \text{if } n > 0,\ j > 0,\ i = 0: & opt\left(n, j-1, |\Theta_{j-1}| - 1\right) & \text{(2.16c)} \\ \text{if } 0 < n < \left| F^j_{\bar{\theta}_j(i)} \right|,\ j > 0: & opt\left(n, j, i-1\right) & \text{(2.16d)} \\ \text{if } n \geq \left| F^j_{\bar{\theta}_j(i)} \right| > 0,\ j > 0: & max \Bigg\{ opt\left(n, j, i-1\right), \\ & opt\left(n - \left| F^j_{\bar{\theta}_j(i)} \right|, j-1, \left| \bar{\theta}_{j-1} \right| - 1\right) + \\ & \left| T^j_{\bar{\theta}_j(i)} \right| - \left| T^j_{\bar{\theta}_j(0)} \right| \Bigg\} & \text{(2.16e)} \end{cases}$$

Each case is explained as follows:

- Case 2.16a: $n = 0$ means that no false rejects are allowed so the first threshold (index 0) in each class is chosen. The resulting amount of true rejects is summed up over all classes.

- Case 2.16b: $j = 0$ means that no class is under consideration for a threshold. So we stay with the first threshold in each class (see above).

- Case 2.16c: $i = 0$ means that no threshold is under consideration in class $j$. So we look in the previous class with the strictest threshold possible.

- Case 2.16d: The chosen threshold $i$ in class $j$ exceeds the allowed amount of false rejects, so the next less strict threshold is considered.

- Case 2.16e: Here the $i$th threshold in $j$ is a possible threshold but it is not clear whether it is optimal. We consider both cases. If it is not the optimal threshold, we take the next less strict one. If it is optimal, we continue our search in the previous class but with $|F^j_{\tilde{\theta}_j(i)}|$ less allowed false rejects in consequence to choosing this threshold. The other consequence is that this threshold results in a number of gained true rejects compared to the least strict threshold and this gain is added.

This Bellmann equation can be computed by dynamic programming (see algorithm TODO). We use three nested loops over $n$, $j$ and $i$. This results in a processing time of $\mathbb{O}(|L| \cdot N \cdot \max_k |\Theta_k|)$. Note that this is equal in effort to the previous method in [LF15] but that the worst case is far less likely to occur here since it would mean that there are no clusters of classification errors. Every grouping of falsely classified points means less computation time than the previous method.

### 2.2.6 Greedy Computation

Computation by dynamic programming gives us an optimal local reject option, but it still might be infeasible in some cases since it "scales quadratically with the number of data" (TODO: quote reject paper). Hence we are looking for a greedy approximation with linear running time. For this approach we define

$$\triangle T^k_{\tilde{\theta}_k(j)} = T^k_{\tilde{\theta}_k(j)} - T^k_{\tilde{\theta}_k(j-1)} \tag{2.17}$$

$$\triangle F^k_{\tilde{\theta}_k(j)} = F^k_{\tilde{\theta}_k(j)} - F^k_{\tilde{\theta}_k(j-1)} \tag{2.18}$$

as the amount of true and false rejects gained by a threshold compared to its predecessor and

$$g = \triangle T^k_{\tilde{\theta}_k} - \triangle F^k_{\tilde{\theta}_k}$$

as the local gain. The greedy approach lies within choosing the thresholds with the best local gain. Initially all thresholds are set to the most tolerant in each class. Looking at the next strict one in each class we pick the one with the highest local gain until the strictest possible thresholds are reached. The pair of true and false rejects $\left(T^k_{\bar{\theta}_k}, F^k_{\bar{\theta}_k}\right)$ is saved in each step if it is an improvement to an existing solutions. Algorithm 2 details this procedure in pseudo code.

**Data**: sets of thresholds $\Theta_k$
**Result**: set of optimal local reject vectors $\Theta_{opt}$
initialization;
array $opt$ where $opt(n)$ is the maximum number of true rejects with n false ones ;
array $theta$ where $theta(n)$ is the optimal threshold vector with n false rejects ;
$\bar{\theta} = \left\{\bar{\theta}_0(0), ..., \bar{\theta}_N(0)\right\}$ ;
$F :=$ amount of false rejects for $\bar{\theta}$ ;
$T :=$ amount of true rejects for $\bar{\theta}$ ;
**while** *a more strict threshold in at least one class exists* **do**
  **foreach** $\theta_k(j) \in \bar{\theta}$ **do**
    **if** $j + 1 < |\Theta_k|$ **then**
      $g(k) = \triangle T^k_{\bar{\theta}_k(j+1)} - \triangle F^k_{\bar{\theta}_k(j+1)}$ ;
    **else**
      $g(k) = -\infty$ ;
    **end**
  **end**
  $\hat{k} = \arg\max_k g(k)$ ;
  set threshold $\theta_{\hat{k}}(j)$ in $\bar{\theta}$ to $\theta_{\hat{k}}(j+1)$ ;
  $T = T + \triangle T^{\hat{k}}_{\bar{\theta}_{\hat{k}}(j+1)}$ ;
  $F = F + \triangle F^{\hat{k}}_{\bar{\theta}_{\hat{k}}(j+1)}$ ;
  **if** $opt(F) < T$ **then**
    $opt(F) := T$ ;
    $theta(F) = \bar{\theta}$ ;
  **end**
**end**
**return** *theta*

**Algorithm 2:** Computing optimal local reject options by greedy evaluation.

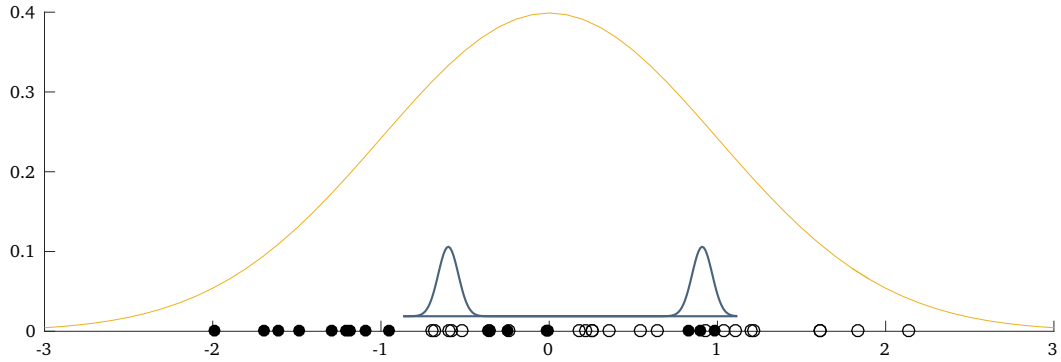
## 2.3 Evaluation

In this chapter we evaluate the described methods. We try to confirm the optimality of the dynamic programming method by comparing its result to the brute force algorithm. Additionally we will compare the run time of both algorithms. Furthermore we want to find out how close the greedy approach is to being optimal.

### 2.3.1 Data Sets

For our evaluation we use randomly generated data sets. Using a normal distribution we get random samples of points divided into two classes, above and below $0$. We now introduce further normal distributions at random locations to get cluster of falsely classified points (see figure 2.6).

**Fig. 2.6:** Example of a small 2-classes generated data set with one group of falsely classified points in each class.




To simulate multi class data sets we use multivariate normal distributions to generate random points for data sets with four classes (2D) and with eight classes (3D). The points are separated into the respective class by the coordinate axis (one quadrant corresponds to a class).

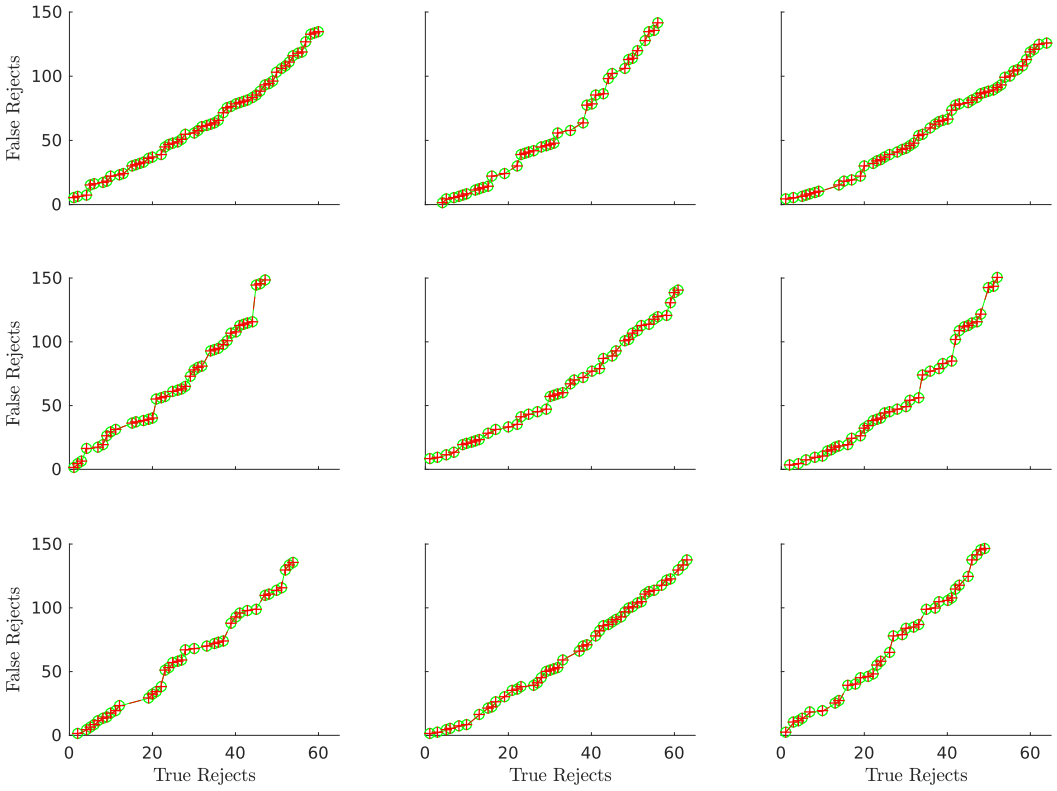
### 2.3.2 Methods

In regards to our definition of an optimal threshold (see chapter 2.1.2) we use the true and false rejects of each threshold considered to be optimal by the respective method. To further see the behavior of the reject options we introduce a measure of quality of the classification in a second evaluation method. Based on [ B.10] we compare the ratio of accepted points to all points $\left(\frac{|A_{\bar{\Theta}}|}{|X|}\right)$ as reject accuracy to the ratio of correctly classified points to all accepted points $\left(\frac{|L \cap A_{\bar{\Theta}}|}{|A_{\bar{\Theta}}|}\right)$ as classification accuracy.

### 2.3.3 DP vs Brute Force

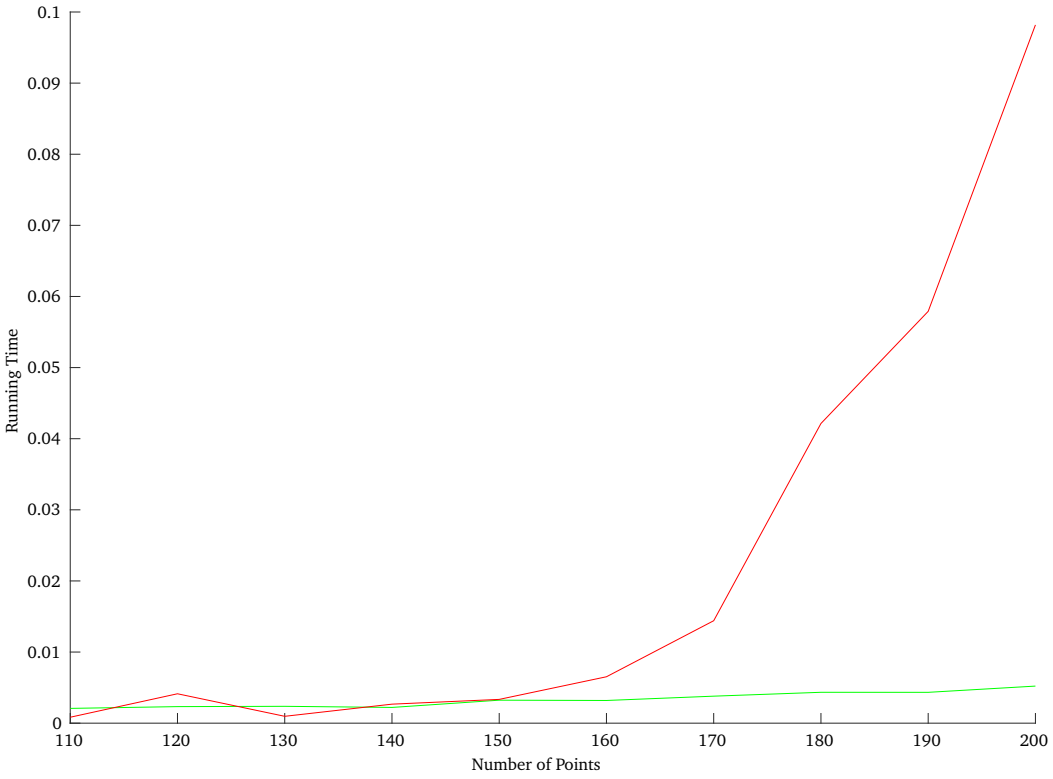
Since we formally did not proof that the Bellman equation (see chapter 2.2.5) finds every optimal reject option we compare its result to the outcome of the brute force method. We can see in figure 2.7 that the results match for nine different generated data sets. Although this is, of course, no proof we conclude for now that the dynamic programming algorithm finds optimal thresholds.

**Fig. 2.7:** This figure shows the true and false rejects for the thresholds considered optimal by dynamic programming(red crosses) and by brute force(red circles) on different randomly generated data sets. We can see that the results coincide in each case.



When comparing the running time of both methods we can see that for multiple classes (eight) we observe the expected result. The brute force method's running time grows exponentially (see figure 2.8).

**Fig. 2.8:** Here we can see the running time in seconds of the brute force (red) compared to the dynamic programming algorithm (green). A small amount of points lets the brute force method be competitive due to constant factors in the implementation. However the time increases exponentially with larger data sets.




### 2.3.4 DP vs Greedy

We now compare the results of the greedy strategy (see chapter 2.2.6) to the optimum. If the results are nearly accurate it is a feasible solution for big data sets since its running time is linear. We can observe in figure 2.9 that the results of the greedy computation are mostly close to being optimal or even optimal and that far outliers are rare and not extreme (TODO:wording). Using our second evaluation method (see figure 2.10) we can observe that the greedy reject options lead to a very similar quality of classification as the ideal ones (TODO: again wording).

**Fig. 2.9:** Comparison of the thresholds computed greedily (red crosses) and the optimal ones computed by dynamic programming (green circles) on randomly generated data sets. We can see that the greedy strategy is close to optimal and sometimes falls off slightly if a lot of points are rejected. There are no noticeable differences when the number of classes is increased.

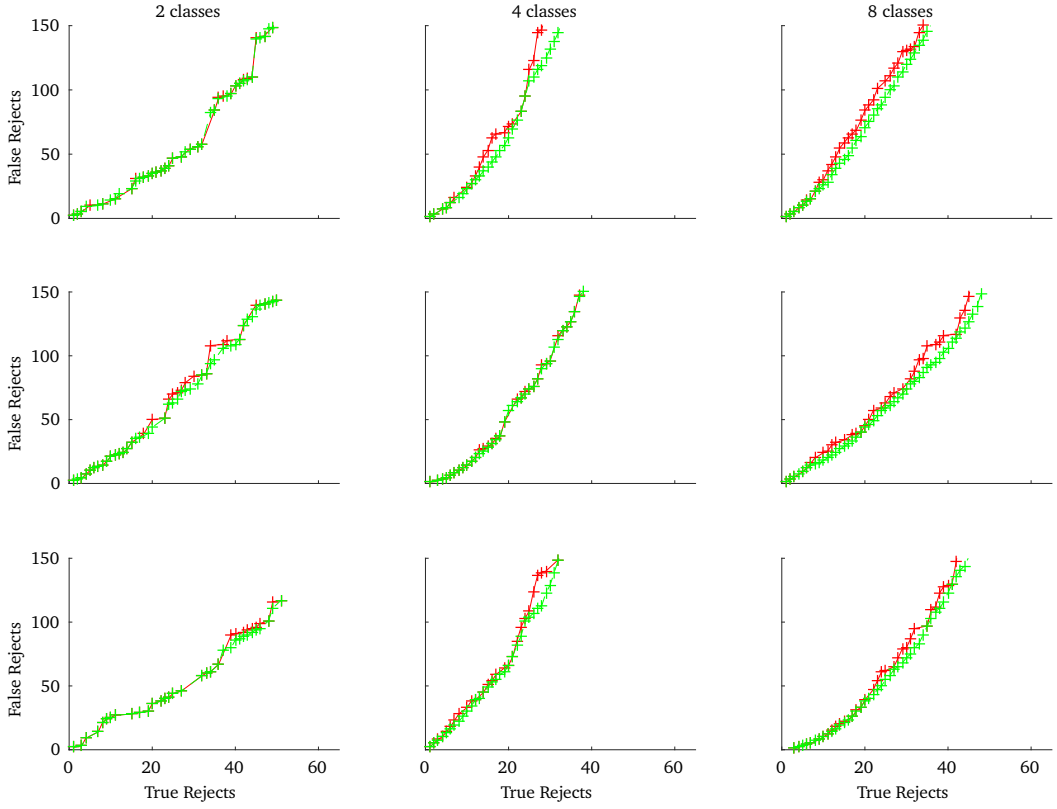
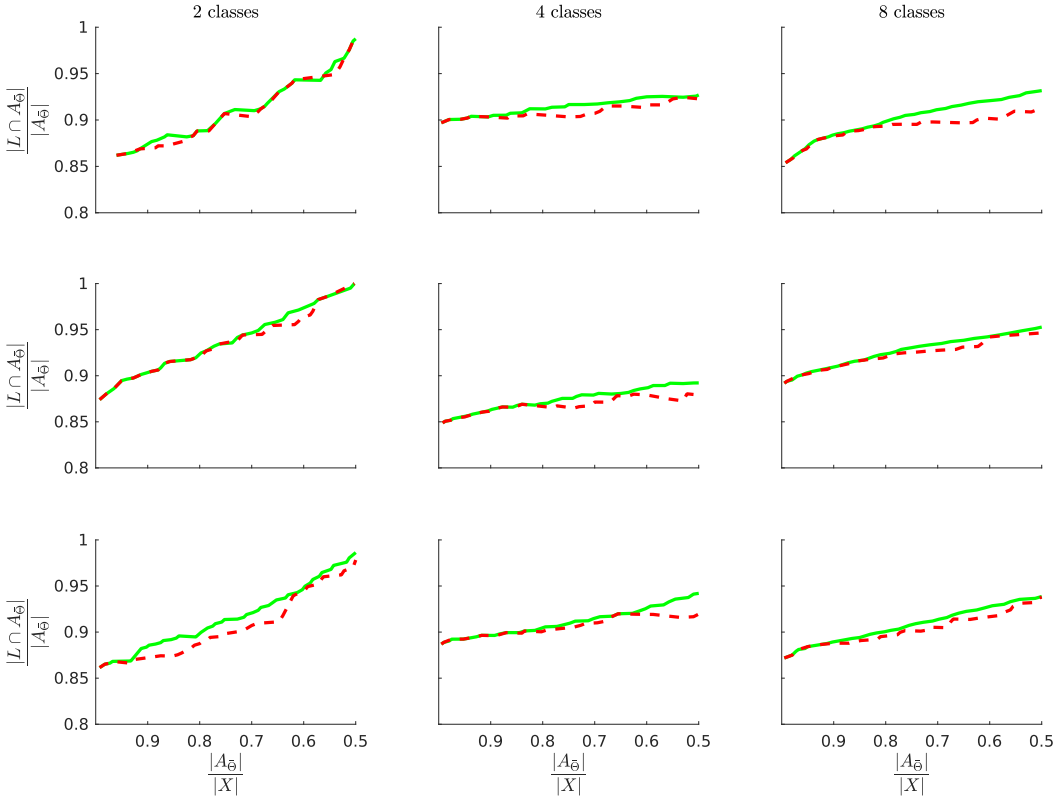


**Fig. 2.10:** ARCs for data sets classified with optimal rejects (green curve) and with greedy rejects (red dashed curve). This supports our observation from the previous test (see figure 2.9) in that the greedy approach can compete with the optimal solutions.



# 3 Application for SVM

# 4 Conclusions

Coming from our goals we set in the introduction (see chapter 1.3) we managed to formulate a reject strategy that expanded on paper [LF15] in reducing the number of thresholds that are considered in the algorithm. The results are still optimal and can be computed in quadratic running time (see chapter 2.3.3). With the prevalence of Big Data in mind we developed a greedy approach that approximated the reject option in linear running time thus feasible for large sets of data. Our evaluation (see chapter 2.3.4) showed that this strategy yields near optimal results on our randomly generated data sets.

## 4.1 Future Works or SVM

# Bibliography


[Alp10] Ethem Alpaydın. *Introduction to Machine Learning*. 2010 (cit. on p. 1).

[Bel57] R. Bellman. „Dynamic Programming“. In: *Princeton University Press* (1957) (cit. on p. 9).

[Cho70] C. K. Chow. „On Optimum Recognition Error and Reject Tradeoff“. In: *IEEE Transactions in Information Theory* (1970) (cit. on p. 1).

[LF15] Heiko Wersing Lydia Fischer Barbara Hammer. „Optimum Reject Options for Prototype-based Classification“. 2015 (cit. on pp. 2, 10, 19).

[MH15] Boris Otto Mario Hermann Tobias Pentek. „Design Principles for Industrie 4.0 Scenarios: A Literature Review“. 2015 (cit. on p. 1).

[PS79] Andris A. Zoltners Prabhakant Sinha. „The Multiple-Choice Knapsack Problem“. 1979 (cit. on p. 9).

[SB13] M. Sugiyama and K. M. Borgwardt. „Rapid Distance-Based Outlier Detection via Sampling.“ In: *Neural Information Processing Systems (NIPS)* (2013) (cit. on p. 1).

[SJDZ05] Donal Doyle Sarah Jane Delany Padraig Cunningham and Anton Zamolotskikh. „Generating Estimates of Classification Confidence for a Case-Based Spam Filter“. In: *Proceedings of the 6th international conference on Case-Based Reasoning Research and Development* (2005) (cit. on p. 1).

[Sut14] Shan Suthaharan. „Big data classification: problems and challenges in network intrusion prediction with machine learning“. In: *ACM SIGMETRICS Performance Evaluation Review* (2014) (cit. on p. 1).

[ B.10] M. S. A. Nadeem J. D. Zucker B. Hanczar. „Accuracy-Rejection Curves (ARCs) for Comparing Classification Methods with a Reject Option.“ In: *Workshop on Machine Learning in Systems Biology (MLSB)* (2010) (cit. on p. 12).

# List of Figures

# List of Tables