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 PaaÃ§en
Theoretical Computer Science
Bielefeld University

Supervisors Prof. Dr. Barbara Hammer and Benjamin PaaÃ§en

October 19, 2015

Johannes Kummert

Optimum reject options for multiclass classification

Bachelor Thesis, October 19, 2015

Reviewers: Prof. Dr. Barbara Hammer and Benjamin PaaÃ§en

Supervisors: Prof. Dr. Barbara Hammer and Benjamin PaaÃ§en

Abstract

Contents

1 Introduction

2 Reject Options

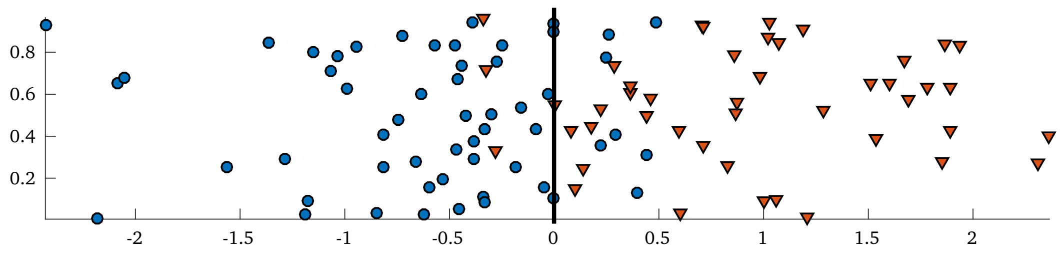
2.1 Two Classes

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.

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

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 θ 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 θ

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 (by y) data set X is divided into the two sets L and E by applying f as a classifier with:

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

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

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

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

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

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), there is no single optimal choice of θ in general. Still there is a set of values that we consider optimal. As shown above each θ corresponds to a tuple $(|T_\theta|, |F_\theta|)$. θ is optimal if

$$\nexists \theta' : |F_\theta'| \leq |F_\theta|, |T_\theta'| \geq |T_\theta|$$

and at least one inequality holds.

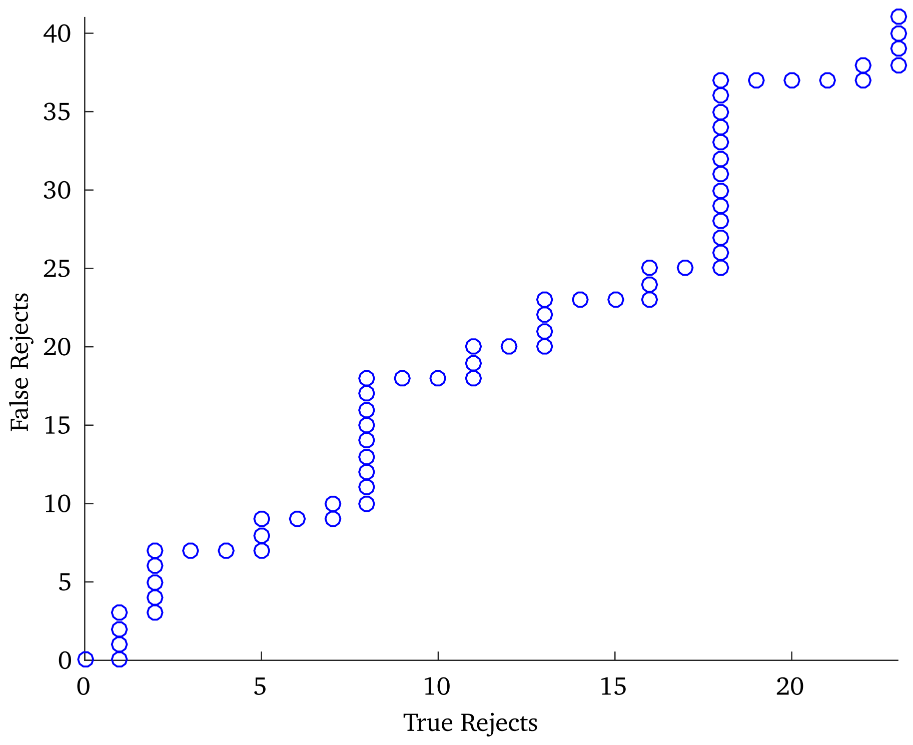
2.1.3 Finding θ

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

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

is a set of possible thresholds. But it would leave us with an infinite search area (TODO: wording?) but this is easily reduced when recognized that it is unnecessary to consider multiple thresholds in between two points that are next to each other (when sorted according to 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 θ. We want $|T_\theta|$, the amount of true rejects, to be large so we should skip cluster of true rejects since choosing a point in the middle of such cluster can not be better than the point at the end of it. Additionally we want $|F_\theta|$, the amount of false rejects, to be small, so we should skip cluster of false rejects since a point in the middle of

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.



such cluster can not be better than a point at the beginning of it. In conclusion this means that it is sufficient to consider only points at the beginning of clusters of false rejects, so

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

where i is the index of the points in X when sorted according to r. This gives us 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_i like in chapter 2.1 and an according measure of confidence r_i. Points are now classified in the class where confidence is maximal among all classes. This gives us a multi class classifier f with

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

$$\bar{x} \mapsto \arg\max_i r_i(\bar{x}) \tag{2.8}$$

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

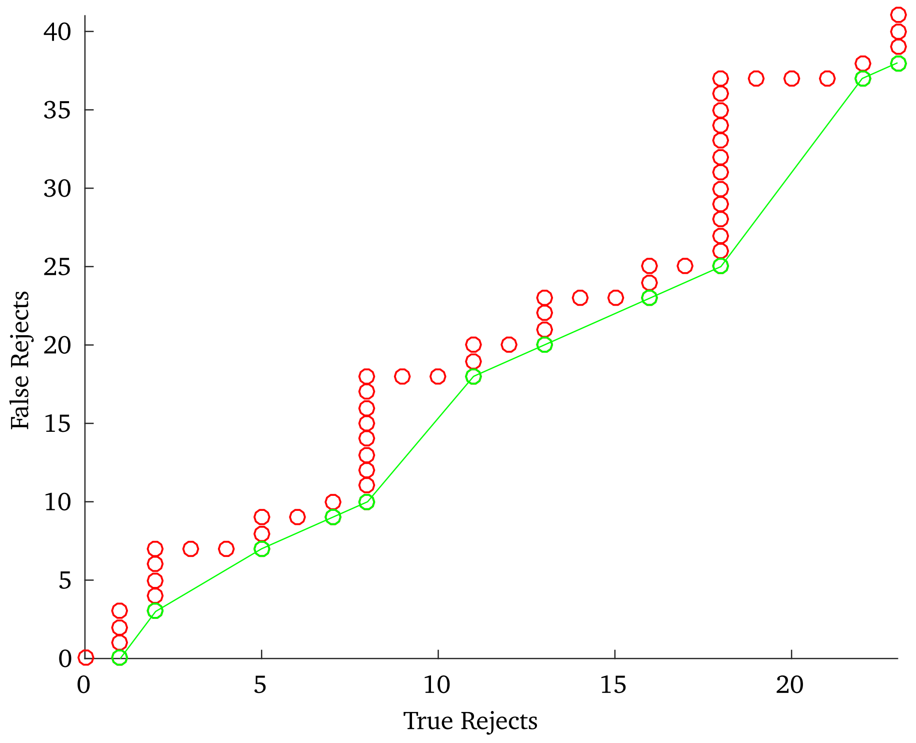
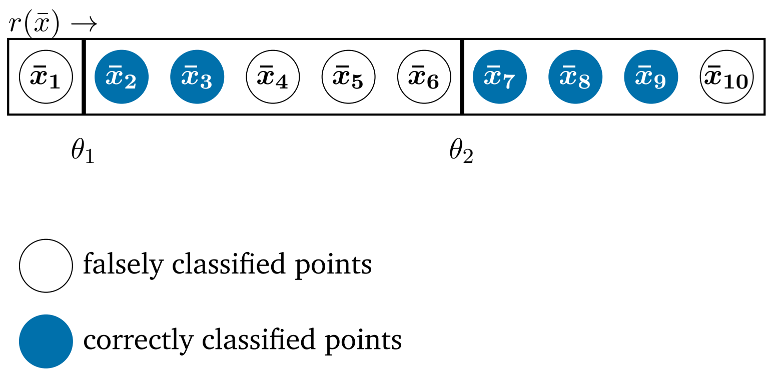


Fig. 2.4: This is an example of classified points sorted by a measure of confidence. You can see that $r(\bar{x}_2)$ and $r(\bar{x}_7)$ are the only sensible thresholds. Choosing $r(\bar{x}_8)$ instead of $r(\bar{x}_7)$ would only add an additional false reject and choosing $r(\bar{x}_6)$ over $r(\bar{x}_7)$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



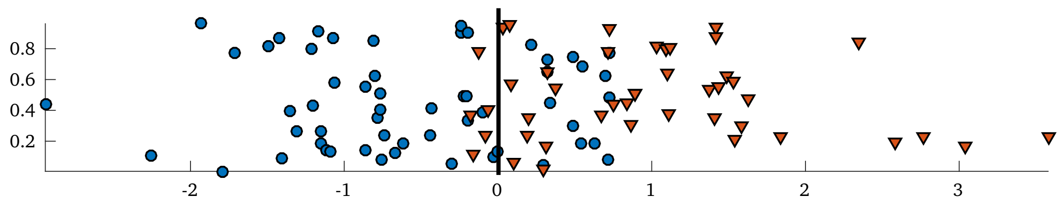
2.2.1 Global Reject

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

$$\max_i r_i(\bar{x}) < \theta \ : \ \bar{x} \text{ rejected}$$

We look again to choose θ 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_i 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 farer 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 each one now has a local reject threshold θ_i. A point is rejected if

$$r_i(\bar{x}) < \theta_i \mid \forall \bar{x} \text{ where } f(\bar{x}) = i$$

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

2.2.3 Optimal Local Reject

For each threshold θ_i 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

$$T^i_{\theta_i} = R^i_{\theta_i} \cap E_i \tag{2.9}$$
$$F^i_{\theta_i} = R^i_{\theta_i} \cap L_i \tag{2.10}$$

where $R^i_{\theta_i}$ is the amount of rejected points in class i given the threshold θ_i, E_i the amount of falsely classified points in class i and L_i the amount of correctly classified points in i. As before we consider a certain set Θ_i of possible thresholds for each class (see chapter 2.1.3). We conclude that the threshold vector $\bar{\theta} = (\theta_1, ..., \theta_N)$ is optimal if each θ_i is optimal. Conversely an optimal θ_i 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 Θ_i. 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 clusters. But our problem is equivalent to a multiple choice knapsack problem (MCKP) where thresholds within a class correspond to items, false rejects correspond to costs, and true rejects correspond to their value (TODO:link). This allows us a faster solution that still maintains optimal results. Let

$$opt(n,j,i) = \max_{\bar{\theta}} \left\{ |T_{\bar{\theta}}| \; s.t. \begin{array}{rc} |F_{\bar{\theta}}| \leq & n \\ \theta_k \in & \Theta_k \forall k < j \\ \theta_j \in & \{\theta_j(0), ..., \theta_j(i)\} \\ \theta_k \in & \theta_k(0) \forall k > j \end{array} \right\}$$

(TODO: define $|T_{\bar{\theta}}|$ and $|F_{\bar{\theta}}|$??? and range of n,j,i)
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 2.11 to efficiently compute opt:

$$opt(n,j,i) = \begin{cases} \text{if } n = 0 : & \sum_{k=1}^{N} \left| T^k_{\Theta_k(0)} \right| & \text{(2.11a)} \\ \text{if } n > 0,\, j = 0 : & \sum_{k=1}^{N} \left| T^k_{\Theta_k(0)} \right| & \text{(2.11b)} \\ \text{if } n > 0,\, j > 0,\, i = 0 : & opt\left(n, j-1, |\Theta_{j-1}| - 1\right) & \text{(2.11c)} \\ \text{if } 0 < n < \left| F^j_{\Theta_j(i)} \right|,\, j > 0 : & opt\left(n, j, i-1\right) & \text{(2.11d)} \\ \text{if } n \geq \left| F^j_{\Theta_j(i)} \right| > 0,\, j > 0 : & max\left\{ opt\left(n, j, i-1\right), \right. & \\ & opt\left(n - \left| F^j_{\Theta_j(i)} \right|, j-1, |\Theta_{j-1}| - 1\right) + & \\ & \left. \left| T^j_{\Theta_j(i)} \right| - \left| T^j_{\Theta_j(0)} \right| \right\} & \text{(2.11e)} \end{cases}$$

Each case is explained as follows:

- Case 2.11a: $n = 0$ 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 2.11b: $j = 0$ means that no class is under consideration for a threshold. So we stay with the first threshold in each class (see above).

- Case 2.11c: $i = 0$ means that no threshold is under consideration in class j. So we look in the previous cell with the strictest threshold possible.

- Case 2.11d: The chosen threshold i in class j exceeds the allowed amount of false rejects, so the next less strict threshold is considered.

- Case 2.11e: Here the ith 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 $|F^j_{\Theta_j(i)}|$ 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 equal in effort to the previous method in (TODO:ref to reject paper) 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 unfeasible 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

$$\triangle T^i_{\theta_i(j)} = T^i_{\theta_i(j)} - T^i_{\theta_i(j-1)} \tag{2.12}$$

$$\triangle F^i_{\theta_i(j)} = F^i_{\theta_i(j)} - F^i_{\theta_i(j-1)} \tag{2.13}$$

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

$$g = \triangle T^i_{\theta_i} - \triangle F^i_{\theta_i}$$

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 $\left(T^i_{\theta_i}, F^i_{\theta_i}\right)$ 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 $\Theta_i$
**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 ;
$\bar{\theta} = \{\theta_0(0), ..., \theta_N(0)\}$ ;
$F :=$ 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 $\theta_i(j) \in \bar{\theta}$ do
        if $\exists \theta_i(j+1)$ then
            $g(i) = \triangle T^i_{\theta_i(j+1)} - \triangle F^i_{\theta_i(j+1)}$ ;
        else
            $g(i) = -\infty$ ;
        end
    end
    $\hat{i} = \arg\max_i g(i)$ ;
    set threshold $\theta_{\hat{i}}(j)$ in $\bar{\theta}$ to $\theta_{\hat{i}}(j+1)$ ;
    $T = T + \triangle T^i_{\theta_{\hat{i}}(j+1)}$ ;
    $F = F + \triangle F^i_{\theta_{\hat{i}}(j+1)}$ ;
    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.

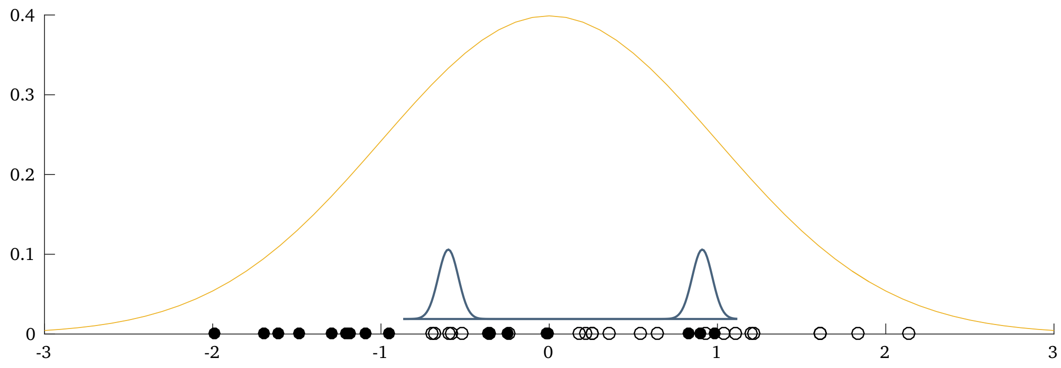
2.2.7 Evaluation

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. Furthermore we want to find out how close the greedy approach is to being optimal.

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 generated data set with one cluster of falsely classified points in each class.



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. (TODO: ref to arc paper).

DP vs Brute Force

Since there is no formal 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.

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 2.8 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 2.9) we can observe that the greedy reject options lead to a very similar quality of classification as the ideal ones (TODO: again wording).

Fig. 2.7: 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.

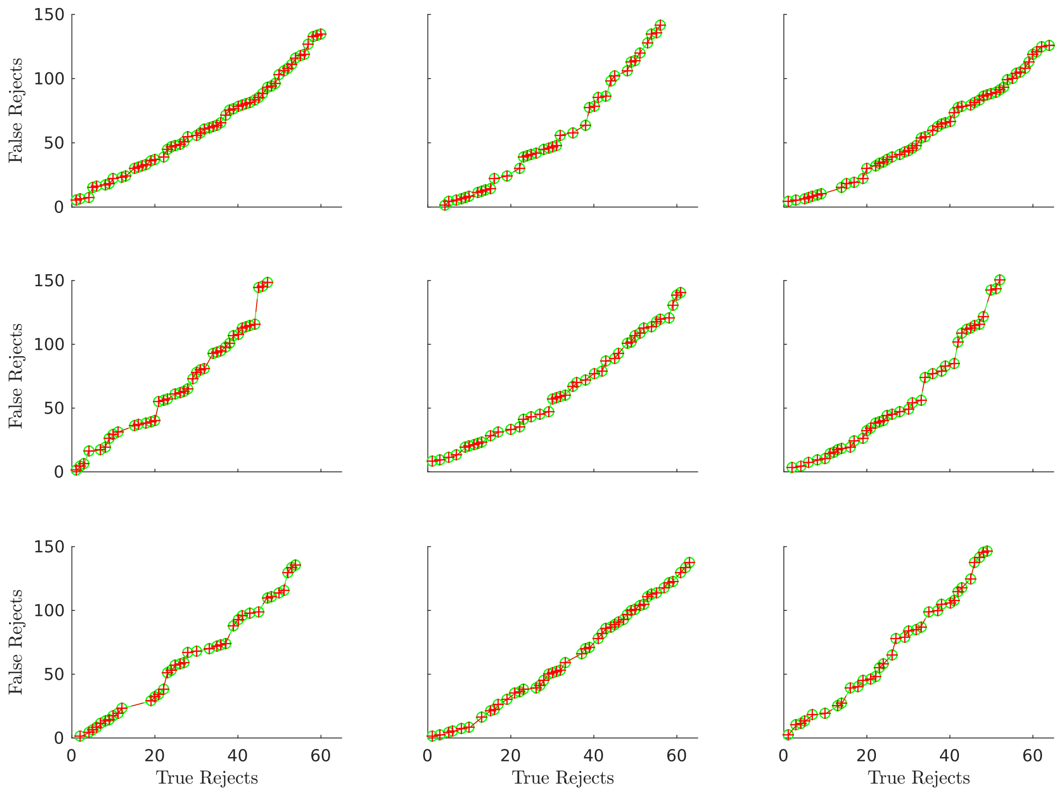


Fig. 2.8: Comparison of the thresholds computed greedily (red crosses) and the optimal ones computed by dynamic programming (green circles) on randomly generated data sets. We can see that the greedy strategy is close to optimal and sometimes falls off slightly if a lot of points are rejected.

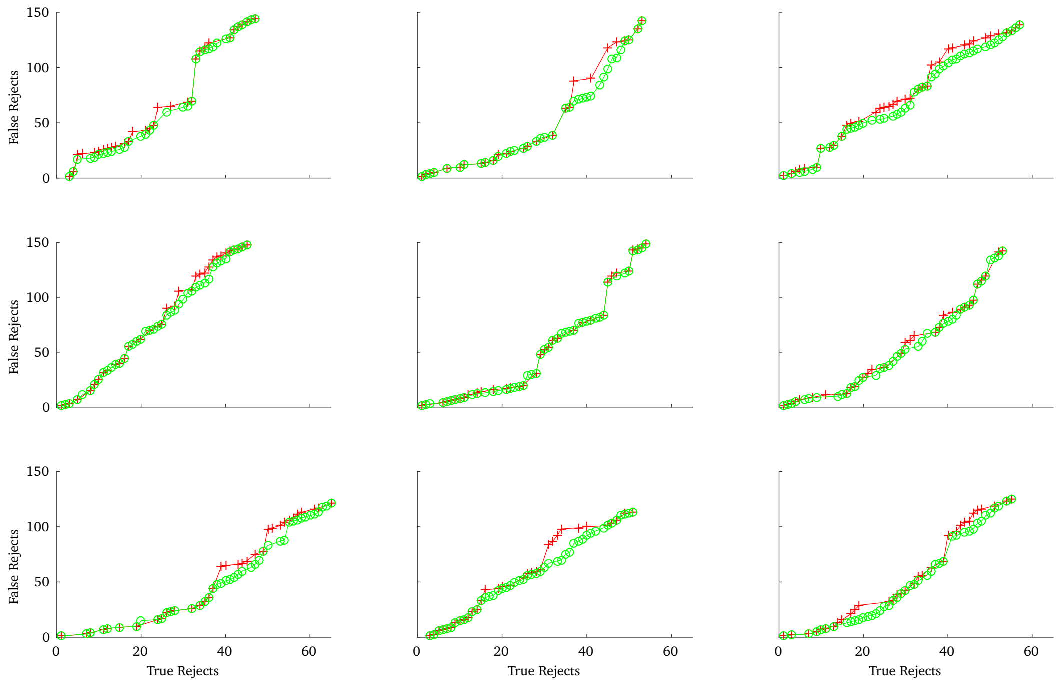
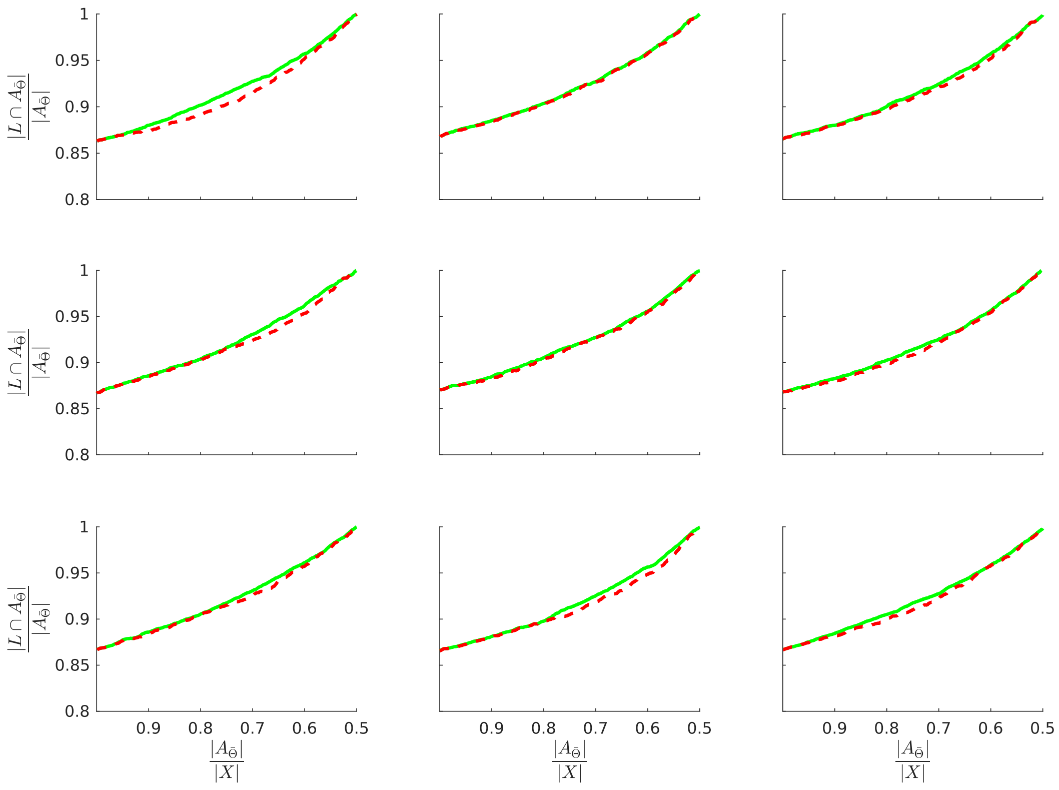


Fig. 2.9: ARCs for data sets classified with optimal rejects (green curve) and with greedy rejects (red dashed curve).



Application for SVM

3

4 Conclusions

List of Figures

List of Tables