# MSA220 Project 2, Method Comparisons for Classification

Elouan Tardivel
elouan@student.chalmers.se

Louisa Schlachter
louisaschlachter@gmail.com

## 1 Classifier 1: Optimal Weighted Nearest Neighbour (OWNN)

Suppose that our data consists of pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ in $\mathbb{R}^d \times \{1, 2\}$. This means that there are $d$ features, $n$ observations and 2 classes. In our case, we have a high dimensional data set and $d > n$. We reorder the training data such that it fulfills:

$$\|X_1 - x\| \leq \ldots \leq \|X_n - x\|,$$

for some element $x \in \mathbb{R}^d$ from the test data and where $\|\,.\,\|$ is for example the $L_2$-Norm (euclidean distance).

We recall the functionality of the **$k$-nearest neighbour classifier**: It assigns the class to $x$ which is the major class among the $k$ nearest training points of $x$. This means:

$$\begin{cases} \text{Class 1}, & \text{if } \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}_{\{Y_i=1\}} \geq 1/2 \\ \text{Class 2}, & \text{otherwise.} \end{cases}$$

A **weighted nearest neighbour classifier** gives each neighbour of $x$ a weight. Therefore the $i$th nearest neighbour has a weight $w_i$, whereas all weights have to sum up to 1. This means we assign classes according the following rule:

$$\begin{cases} \text{Class 1}, & \text{if } \frac{1}{k}\sum_{i=1}^{k} w_i \mathbb{1}_{Y_i=1} \geq 1/2 \\ \text{Class 2}, & \text{otherwise.} \end{cases}$$

To find optimal weights, we want to minimize the misclassification error rate, ie. the risk. The Bayes Classifier gives an optimal risk, denoted by $R_{\mathcal{R}}(C^B)$,

but it can not be used in practice. If $R_{\mathcal{R}}(C^{wnn})$ defines the risk of the optimal weighted $k$-nearest neighbour method, we obtain the following excess risk:

$$\mathcal{R}_{\mathcal{R}}(C_n^{wnn}) - \mathcal{R}_{\mathcal{R}}(C^{Bayes}) = \left(B_1 s_n^2 + B_2 t_n^2\right)\{1 + o(1)\},$$

for some constants $B_1,\ B_2$ and with $t_n = n^{-2/d}\sum_{i=1}^{n} w_{ni}(i^{1+2/d} - (i-1)^{1+2/d})$, $s_n^2 = \sum_{i=1}^{n} w_i^2$.

Minimization leads according to [Sam12] to the following **optimal weights**:

$$w_i^* = \begin{cases} \frac{1}{k^*}\left(1 + \frac{d}{2} - \frac{d}{2k^{*2/d}}(i^{1+2/d} - (i-1)^{1+2/d})\right), & i = 1, 2, \ldots, k^* \\ 0, & i = k^*+1, \ldots, n, \end{cases}$$

with $k^* = \lfloor Bn^{\frac{4}{d+4}} \rfloor$, where the constant $B$ depends on $B_1$, $B_2$ and $d$. Therefore only a proportion of $\mathcal{O}(n^{-d/(d+4)})$ of the weights is non-zero. Moreover, the weights get smaller the more far away the neighbours get from $x$. This means, that the classes of close training data points have more influence on the prediction. Since in our case $d$ is large for the high-dimensional data, this relationship can be roughly described with "value weight $\approx$ C/neighbour order".

**Tuning of parameters**: An analysis in [Gus09] of 10-fold cross-validations of 31 data sets leads to the result, that the performance of $k$-nearest neighbour methods has its maximum between $k = 5$ and $k = 11$. Outside of this interval, the accuracy decreases. Moreover, the null hypothesis of no difference between different values of $k$ has been rejected in all the cases.

We note, that a choice of $k$ which is too small tends to overfitting. On the other hand, a value of $k$ which is too big ignores locally interesting behaviour.

We will apply this method to two data sets. At first we consider the **Arcene** data. It provides data of three mass-spectrometry data sets for the two classes cancer (1) and no cancer (-1). Arcene contains 200 instances with 10000 features. 88 of the 200 patients have cancer and no values are missing.

We read the data in R and perform a 10-fold cross-validation. We define the training data as *X.train* with labels *Y.train*. The function `myownn` of the package `snn` needs the following inputs:

- A matrix which contains of the training data. The columns represent the features whereas the last column are the labels. The rows represent the instances.
- The test data.

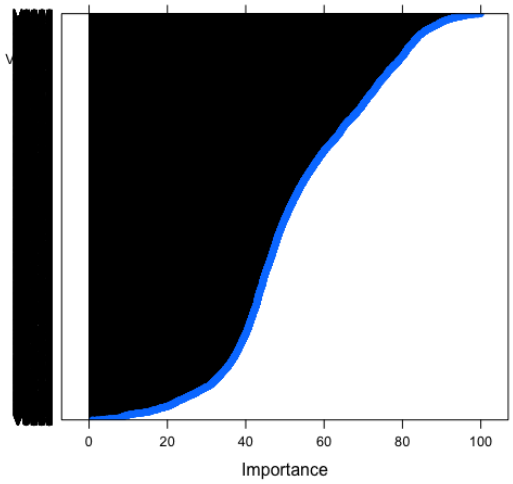


Figure 1: Variable importance of Arcene with OWNN.

- The parameter $k$ to perform the optimal weighted $k$-nearest neighbour method.

We apply the `cv.tune` function of the same package to obtain a good choice for $k$. This is done by a 5-fold cross-validation and leads to $k = 5$.

The predicted classes are now calculated by:

```
out <- myownn(cbind(X.train,Y.train), X.test, k)
```

At last, we compute the misclassification error and average it over the cross-validation groups. In our case, we get:

Misclassification error rate: 14%

Now, we will have a look on the **variable importance** of the OWNN-method. We run the `varImp` function from the `caret` package on the first training group of our cross-validation. The results are shown in figure 1.

The importance-curve takes the form of a "S"-shape, whereas the slope is not significantly big or small. It is quite centered around 50%, which means that the importance is equally spread out. Only a few percentage of the features is very important ($> 80\%$) or very unimportant ($<20\%$).

The 20 most important variables are shown in the following table. Note, that the second row contains the variable names (V#column) and the third one contains the importance of the features in percentage.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| V2556 | V2309 | V8368 | V3365 | V4960 | V312 | V9617 | V9092 | V5005 | V7891 |
| 100.00 | 99.81 | 99.80 | 99.53 | 99.53 | 99.47 | 99.38 | 99.30 | 99.19 | 99.09 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| V414 | V1148 | V376 | V9027 | V2866 | V8585 | V4147 | V1748 | V3288 | V2227 |
| 98.86 | 98.44 | 98.32 | 98.16 | 97.98 | 97.87 | 97.86 | 97.66 | 97.63 | 97.25 |

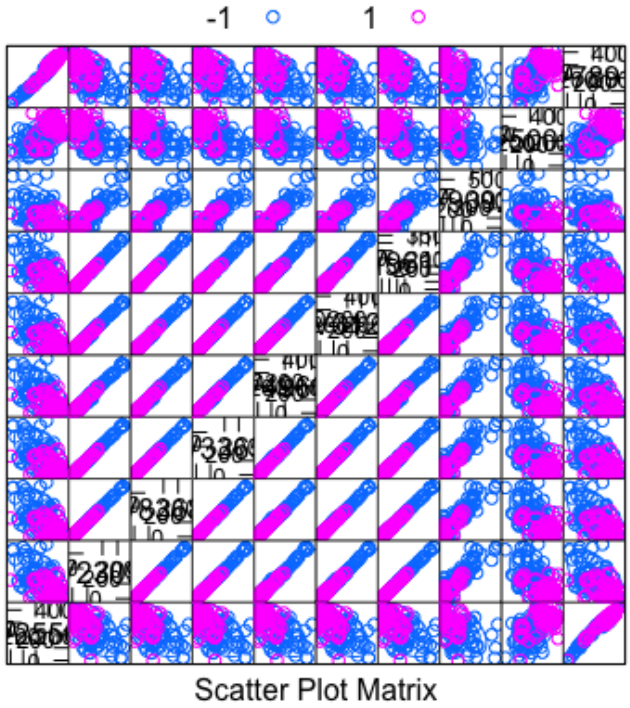


Figure 2: Scatter plot of the 10 most important variables computed with the caret package

We can observe from the 10 most important variables that the two classes are linearly separable even though a lot of overlapping occurs.

The second data set is called **Gisette**. It contains data about the handwritten digits four and nine, therefore we again have two classes. The handwritten data has been normalized to an image of dimension 28×28. Our data set has 5000 features, containing information about the pixels, and 1000 instances (ie. im-

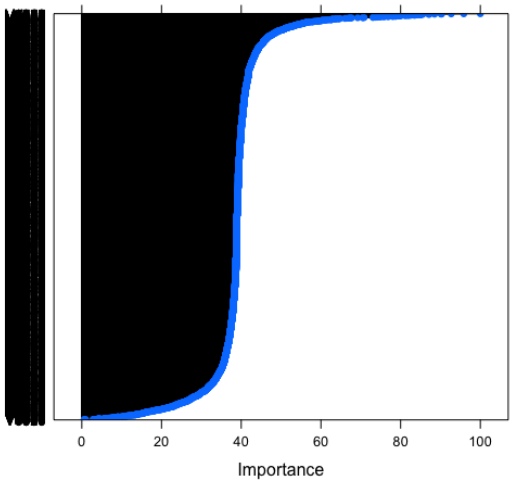


Figure 3: Variable importance of Gisette with OWNN.

ages).

The important features of this data set are the features, that describe pixels which are crucial to distinguish between the two numbers. We observe, that the written digits look very similar. The most difference is the upper "curve" that closes the "circle" of the nine, which doesn't exist in four. Furthermore, the given data is sparse, which means that many entries are zero. This is the case, because there is no writing on different spots of the image. In our data set, 286 of the 5000 columns (ie. 286 features) have only zero-entries in each row. These features would give no information which could help to distinguish between the two digits.

We run the same procedure of the 10-fold cross-validation as for the Arcene-data. It gives on average:

Misclassification error rate: 6.5%.

This value is significantly better than for the Arcene data. This could be the case, since the classes of the Gisette data should be more clear to classify. It seems easier to distinguish between the two handwritten digits than to decide if someone has cancer, based on different body characteristics.

Additionally, we perform the `varImp` function for the first training group of the Gisette data set from the cross-validation. The result is shown in figure 6.

Compared to the Arcene data, this graph has a much bigger slope. Almost all the variables show an importance around 40%. Again, only a few features are very (un-)important. This makes sense since except of some crucial and unimportant (white) pixels, most of the spots should be equally important.

The 20 most important variables are shown in the following table with the same properties as before.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| V3657 | V558 | V512 | V2743 | V4508 | V3003 | V3976 | V215 | V905 | V3722 |
| 100.00 | 95.84 | 92.69 | 90.31 | 90.13 | 88.51 | 87.23 | 87.03 | 85.54 | 85.17 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| V339 | V1229 | V4876 | V1772 | V4165 | V1600 | V4387 | V2302 | V4879 | V1559 |
| 84.95 | 84.95 | 84.91 | 84.91 | 84.12 | 84.01 | 83.64 | 83.64 | 83.35 | 82.54 |

## 2 Classifier 2: Sparse LDA

Since we have already discussed this method in the lecture, we will only apply it on our data sets:

**Arcene:** Normalization of the training data reduces the feature set. In our case, the 10000 features are reduced to 9950. We get the predicted values of the test data with the following procedure:

```
Xc <- normalize(X.train)
Xn <- Xc$Xc

out <- sda(Xn, Y.train, lambda = 1e-6, stop = -1,
                        maxIte = 25, trace = TRUE)

  X.test <- normalizetest(X.test, Xc)
  test <- predict(out, X.test)
```

A 10-fold cross-validation gives an average misclassification error of 35%.

An analogous analysis for the **Gisette** data set leads to an average misclassification error of 17%.

Moreover, the variable importance analysis by `varImp` gave the exact same results as before.

# 3 Classifier 3: Support Vector Machine (SVM)

The method SVM is about finding a decision boundary between two classes that maximises the margin.

In two dimension a linear decision boundary can be represented as a straight line.

This line or hyperplane in higher dimension, can be expressed by the following equation :

$$w^T x + b = 0$$

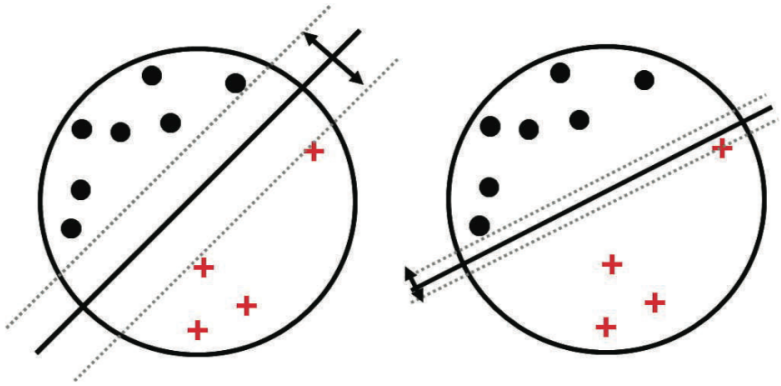

**Figure 14.11** Illustration of the large margin principle. Left: a separating hyper-plane with large margin. Right: a separating hyper-plane with small margin.

Figure 4: Margin and class boundary representation from [Mur12]

## Linearly separable class with hard margin

We want to maximize the distance from the closest point to the boundary. Hence, maximising the margin $\gamma = \frac{1}{\|w\|}$.

That is equivalent to minimizing $\|w\|$ or $\frac{1}{2}\|w\|^2$.

Suppose that we have $n$ observations $x_i \in \mathbb{R}^d$ from two classes. To ease the notation we consider the class labels $y_i$ as -1 or +1. In the hard margin case, we want all the occurrences of one class to be on one side of the boundary and the rest on the other side. Therefore, we have all these constraints

$$\forall i \in [\,|1, n|] \qquad y_i(w^T x_i + b) \geqslant 1$$

NB : In order to solve this optimization problem, Lagrange multipliers are often used. This is called the dual problem and it is used reduce the number of variables. In a high dimensional setting, it is easier to solve the dual problem because the number of constraints (i.e $n$) in the primal problem becomes the number of variables of the dual problem .

Once the optimization problem is solved we can compute the hyperplane. For a new observation $x_new$ the class prediction will be given by :

$$sign(w^T x_{new} + b)$$

## Linearly separable class with soft margin

Introduction of slack variables If the data are not completely linearly separable

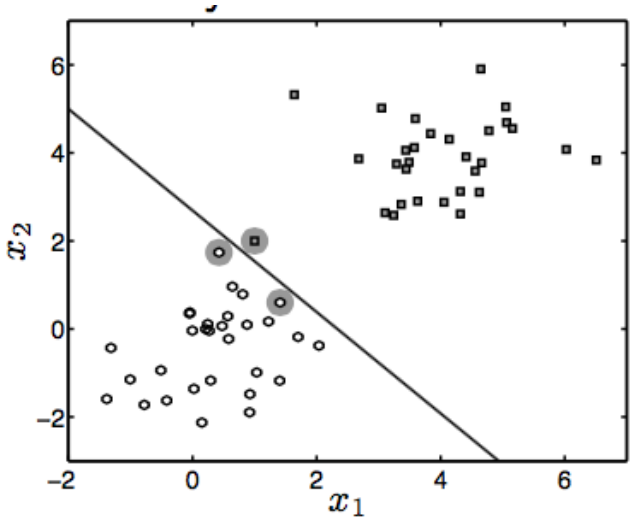


Figure 5: Soft margin justification

of if some data points from one class are outliers and lie closer to the other class then the hard margin seems not to give the right classifier. That is why we can introduce slack variables in the constraints in order to a soft constraints problem.

$$\forall i \in [\,|1, n|] \qquad y_i(w^T x_i + b) \geqslant 1 - \xi_i$$
$$\xi_i \geqslant 0$$

The objective becomes

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

The parameter C allows us to control how many miss-classify observations we want during the training. If C is too small the model is under-fitting and we lose sparsity. However, if C is too big we over-fit the noise. The choice of this regularization parameter is decisive for the quality of the model, cross-validation is often used to find the proper value.

## Non Linearly separable class

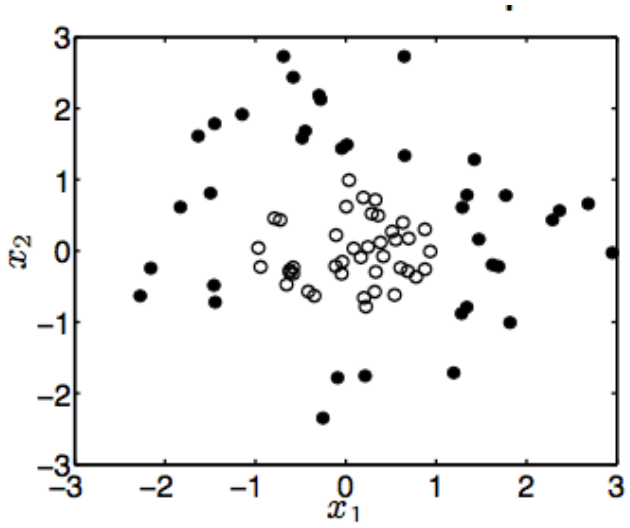


Figure 6: Non linearly separable dataset

When the class are not linearly separable, a simple solution can be using a transformation function that projects the points in a space where they are linearly separable. One can show that the problem can be written as follow :

$$\min_{\alpha} \sum_{i} \alpha_i + \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

This lets appear the inner product $x_i^T x_j$, we can use the "kernel trick" and replace this inner product by a kernelized function.

$$\min_{\alpha} \sum_{i} \alpha_i + \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

Many off-the-shelf kernels exist depending on which problem you are facing. The Gaussian kernel and polynomial one are two of them.

We tried the Radial and Gaussian kernel on the Arcene dataset, it is notable that the results is worse than with linear kernel. As the scatter plot matrix from the figure 2 shows, the two classes seems to be linearly separable and that could explain the performances of Radial and Gaussian kernel.

NB: We will not deal with multiple class problems, but the general idea relies on the "one-against-one" or "one-against-all" approaches.

# 4 Comparison

**Arcene**:
2 classes, 200 instances and 10000 features.

**Gisette**:
2 classes, 1000 instances and 5000 features.
Handwritten digits 4 and 9, normalized to an image of dimension 28×28.

Apply **10-fold cross-validation** to get **misclassification error** as an average of the cross-validation cases:

| Method | OWNN | SVM | sparseLDA |
|---|---|---|---|
| Error Arcene | 14% | 12% | 35% |
| Error Gisette | 6.5% | 4.7% | 17% |

# References


[Gus09] Diego Furtado Silva Gustavo E.A.P.A. Batista. *How k-Nearest Neighbor Parameters Affect its Performance?* 2009.

[Mur12] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspective*. The MIT Press, 2012.

[Sam12] Richard J. Samworth. *Optimal Weighted Nearest Neighbour Classifiers*. 2012.