Statistical Methods for Machine Learning
Assignment 2: Basic Learning Algorithms

Philip Pickering
pgpick@gmx.at

Marco Eilers
eilers.marco@googlemail.com

Thomas Bracht Laumann Jespersen
ntl316@alumni.ku.dk

1 Regression

1.1 Maximum Likelihood solution

We are using a linear model:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D$$

and we let $\phi_i(\mathbf{x}) = x_i$ for $i = 1, \ldots, D$ and $\phi_0(\mathbf{x}) = 1$.

1.1.1 Selection 1

For our first selection S_1 our design matrix becomes a 200×5 matrix.

$$\mathbf{\Phi}_{S_1} = \begin{bmatrix} 1 & \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \mathbf{x}_{1,3} & \mathbf{x}_{1,4} \\ & & \vdots & & \\ 1 & \mathbf{x}_{i,1} & \mathbf{x}_{i,2} & \mathbf{x}_{i,3} & \mathbf{x}_{i,4} \\ & & \vdots & & \\ 1 & \mathbf{x}_{N,1} & \mathbf{x}_{N,2} & \mathbf{x}_{N,3} & \mathbf{x}_{N,4} \end{bmatrix}$$

where the notation $\mathbf{x}_{i,j}$ indicates the j'th entry in the i'th vector.

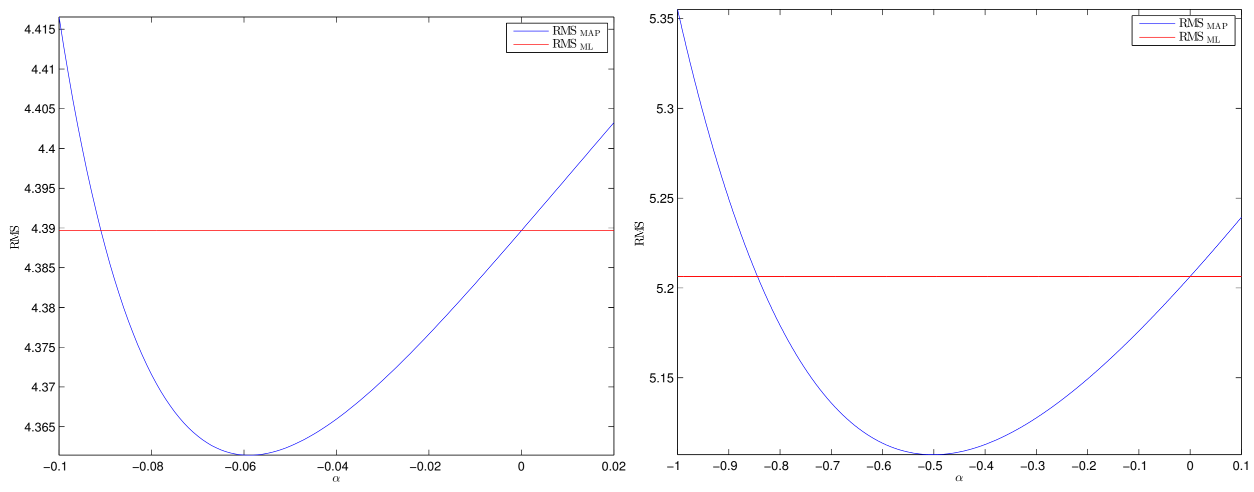
Finding the ML estimate of our parameters for S_1 gives

$$\mathbf{w}_{S_1} = \begin{bmatrix} -43.0947 \\ -0.1299 \\ 0.0352 \\ 0.9335 \\ -0.0433 \end{bmatrix} \quad \text{and} \quad \text{RMS}_{S_1} = 4.3897$$

1.1.2 Selection 2

Our second selection S_2 consists only of the data from the 'Abdomen 2' column, giving a design matrix $\mathbf{\Phi}_{S_2}$ of dimensions 200×2. Training the model on the same training data yields:

$$\mathbf{w}_{S_2} = \begin{bmatrix} -37.4085 \\ 0.6133 \end{bmatrix} \quad \text{and} \quad \text{RMS}_{S_2} = 5.2064$$



(a) (b)

Figure 1: Plot of RMS against varying values of α

1.1.3 Discussion

Just looking at the root mean square values of the two selections, it appears that S_1 performs better than S_2, but not by a lot. This could suggest that either the variable 'Abdomen 2' is the most descriptive in terms of body fat, or that the linear model simply is a poor fit no matter how many variables we include. It could be a combination of the two.

It could probably be argued that the linear regression model is a poor predictor, but including more variables should improve the results.

1.2 Maximum a posteriori solution

For S_1 the lowest root mean square value is 4.3614, the same as the ML solution differing on the second digit. The difference is larger for S_2 in which we obtain the value 5.1071, differing from the ML solution on the first digit.

In fig. 1 two plots are found of the root mean square values for varying values of α. In both plots we set $\beta = 1$. The RMS value of the ML solution is plotted as a straight line.

1.2.1 Comparison

Firstly, we can observe for both plots that when $\alpha = 0$, we obtain the same RMS error for the MAP estimate as for the ML solution. This is expected and demonstrates that when our prior precision parameters are set to zero, the MAP estimate becomes the ML estimate.

Fig. 1(a) is the plot for S_1, and it can be seen that the RMS_{MAP} error drops below the RMS_{ML} in the interval $[-0.091, 0]$. In fig. 1(b) the plot for S_2 similarly gives us that the RMS_{MAP} error is lower in the interval $[-0.844, 0]$.

The intervals in which the MAP estimate outperforms the ML estimate are very small, and the obtained difference likewise small.

1.3 Theory

We have from CB (1.44) that:

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w}) \times p(\mathbf{w}),$$

which reads that the probability distribution of the parameters given the target vector $\mathbf{t}$ is proportional to the probability of the target vector given the parameters multiplied by the probability of the parameters.

This is also Bayes' Theorem in words, namely the posterior distribution is proportional to the likelihood times the prior.

With this in mind we state the likelihood function defined in CB (3.10)

$$\begin{aligned} p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \mathcal{N}(\mathbf{T}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I}) \qquad \text{(from the lecture slides)} \end{aligned}$$

and the prior is given by:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0).$$

As is written in CB page 153, the product of the two Gaussians will again be Gaussian, but we want to verify the result of the multiplication, so we want to 'complete the square' in the exponential. In order to verify the result we would then need to find the normalization coefficient using the standard result for a normalized Gaussian.

$$\begin{aligned} &\exp\left\{-\frac{1}{2}(\mathbf{T} - \boldsymbol{\Phi}\mathbf{w})^T \beta\mathbf{I}(\mathbf{T} - \boldsymbol{\Phi}\mathbf{w})\right\} \times \exp\left\{-\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0)\right\} \\ =& \exp\left\{-\frac{\beta}{2}(\mathbf{T} - \boldsymbol{\Phi}\mathbf{w})^T (\mathbf{T} - \boldsymbol{\Phi}\mathbf{w})\right\} \times \exp\left\{-\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0)\right\} \\ =& \exp\left\{-\frac{\beta}{2}(\mathbf{T} - \boldsymbol{\Phi}\mathbf{w})^T (\mathbf{T} - \boldsymbol{\Phi}\mathbf{w}) - \frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^T \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0)\right\} \qquad (1) \end{aligned}$$

Multiplying out the expression in (1) we find (excluding the exp part):

$$= -\frac{1}{2}\left(\beta\mathbf{T}^T\mathbf{T} - \beta\mathbf{T}\boldsymbol{\Phi}\mathbf{w} - \beta\mathbf{w}^T\boldsymbol{\Phi}^T\mathbf{T} + \beta\mathbf{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{w} + \mathbf{w}^T\mathbf{S}_0^{-1}\mathbf{w} - \mathbf{w}^T\mathbf{S}_0^{-1}\mathbf{m}_0 - \mathbf{m}_0^T\mathbf{S}_0^{-1}\mathbf{w} + \mathbf{m}_0^T\mathbf{S}_0^{-1}\mathbf{m}_0\right)$$

Rearranging in terms of $\mathbf{w}$ and $\mathbf{w}^T$ gives us:

$$= -\frac{1}{2}\left(\mathbf{w}^T(\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{S}_0^{-1})\mathbf{w} - \mathbf{w}^T(\beta\boldsymbol{\Phi}^T\mathbf{T} + \mathbf{S}_0^{-1}\mathbf{m}_0) - (\beta\mathbf{T}^T\boldsymbol{\Phi} + \mathbf{m}_0^T\mathbf{S}_0^{-1})\mathbf{w} + \mathbf{m}_0^T\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{T}^T\mathbf{T}\right) \qquad (2)$$

The last two terms in (2) we can ignore, because they are independent of $\mathbf{w}$ and we will treat them as constant. The expression can be simplified if we set $\mathbf{A} = \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{S}_0^{-1}$ and $\mathbf{B} = \beta\boldsymbol{\Phi}^T\mathbf{T} + \mathbf{S}_0^{-1}\mathbf{m}_0$. We also make use of the fact that $(\beta\mathbf{T}^T\boldsymbol{\Phi} + \mathbf{m}_0^T\mathbf{S}_0^{-1}) = (\beta\boldsymbol{\Phi}^T\mathbf{T} + \mathbf{S}_0^{-1}\mathbf{m}_0)^T = \mathbf{B}^T$, i.e. that $\mathbf{S}_0$ is symmetric, and correspondingly for its inverse it holds that $(\mathbf{S}_0^{-1})^T = \mathbf{S}_0^{-1}$.

$$\begin{aligned} (2) &= -\frac{1}{2}\left(\mathbf{w}^T\mathbf{A}\mathbf{w} - \mathbf{w}^T\mathbf{B} - \mathbf{B}^T\mathbf{w}\right) \\ &= -\frac{1}{2}(\mathbf{w} - \mathbf{A}^{-1}\mathbf{B})^T\mathbf{A}(\mathbf{w} - \mathbf{A}^{-1}\mathbf{B}) \end{aligned}$$

from which we can clearly see that $\mathbf{S}_N^{-1} = \mathbf{A}$ and $\mathbf{m}_N = \mathbf{A}^{-1}\mathbf{B} = \mathbf{S}_N(\beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \mathbf{S}_0^{-1})$

$$= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_N)^T\mathbf{S}_N(\mathbf{w} - \mathbf{m}_N)$$

which is the result that we wanted. Note also we've made use of the fact that $\mathbf{S}_N$ is symmetric. This is true because $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ and $\mathbf{S}_0^{-1}$ both are symmetric and any given linear combination of symmetric matrices will itself be symmetric.

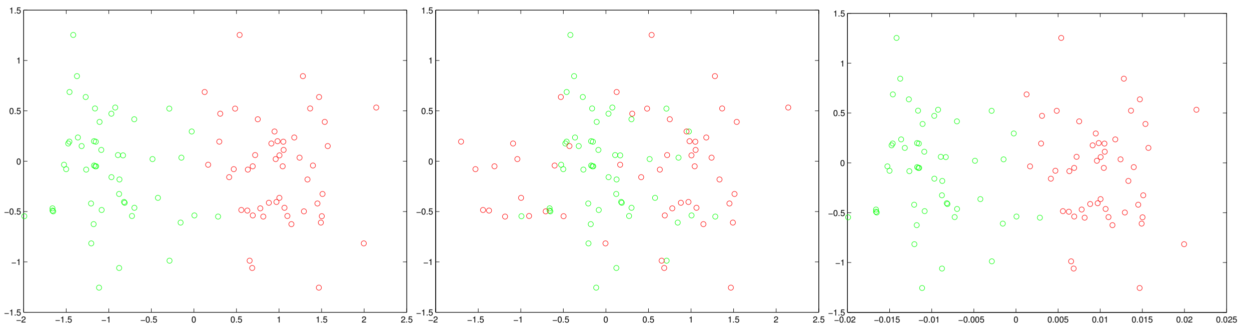

(a) knollA (b) knollB (c) knollC

Figure 2: Visualisation of the training data for each of the knoll problems

2 Linear Discriminant Analysis

2.1 Observations regarding the data

In fig. 2 you can see plots of all three training data sets. It is obvious that while knollA and knollC contain well-separable groups of data points, the distributions in knollB seem to overlap. We can therefore expect that LDA will perform relatively well on knollA and knollB, whereas the results for knollB will likely be more erratic.

2.2 Results of LDA

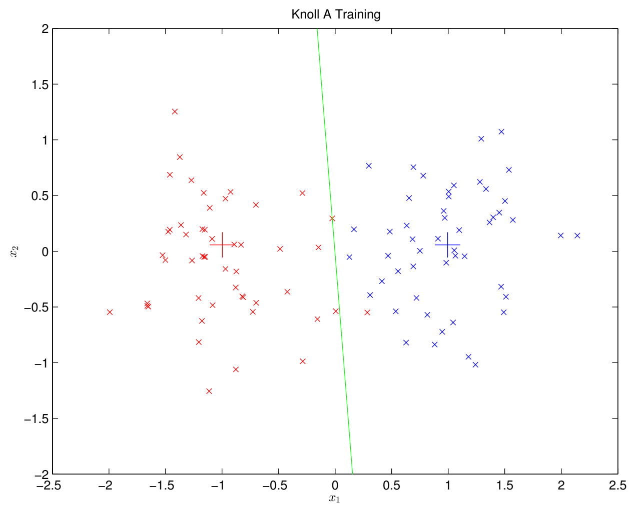
If we run LDA on all three data sets and plot them with their respective decision boundaries (fig. 3), we find that the algorithm indeed performs well on the knollA and knollC sets. In table 1 we can see that although the error count is greater than zero for all training and test sets, which is also implied by the margin always being slightly negative, the number of errors is relatively low for both training and test sets. The training set performs slightly better than the test set in both cases, which is to be expected, but there is not over- or underfitting problem.

For knollB, however, the results are much worse. The properties of the underlying distribution simply do not allow them to be separated by a straight line, which results in a high error percentage and a highly negative margin.

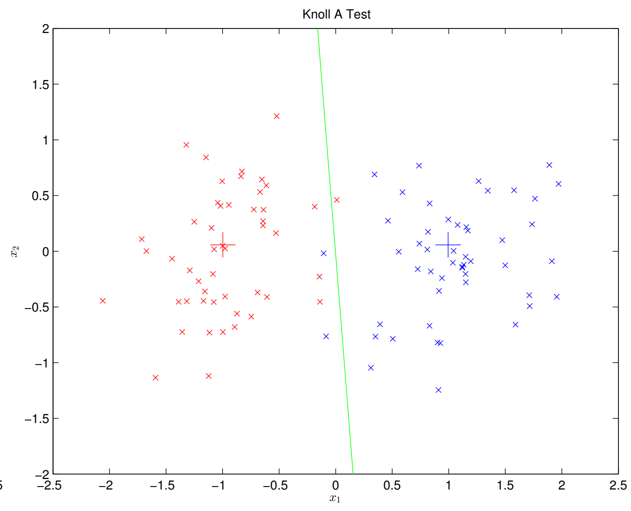
Since the distributions in knollB overlap to a high degree, it is generally not possible to find a function of any kind to separate both classes accurately. However, a Quadratic Discriminant Analysis might be able to adapt to the fact that one class is more concentrated in $-0.5 \leq x_1 \leq 0.5$, whereas the other one has a higher variance and is therefore spread out more widely. It could therefore assign all points in this range to one class and the rest to the other. While this will still result in a high error percentage, such an algorithm might find the Bayes Optimal Solution or a good approximation.

	knollA		knollB		knollC	
	Training	Test	Training	Test	Training	Test
Errors	0.01	0.03	0.40	0.49	0.01	0.03
Margin	-0.242763	-0.142026	-1.629922	-2.061411	-0.002435	-0.001425

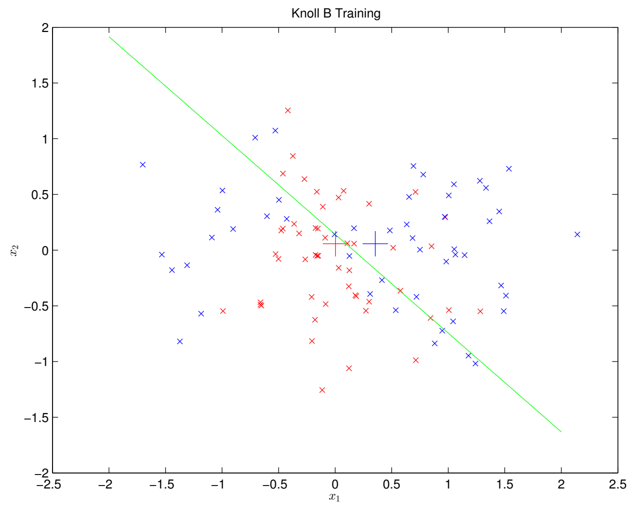
Table 1: Error percentages and margins for each of the knoll problems



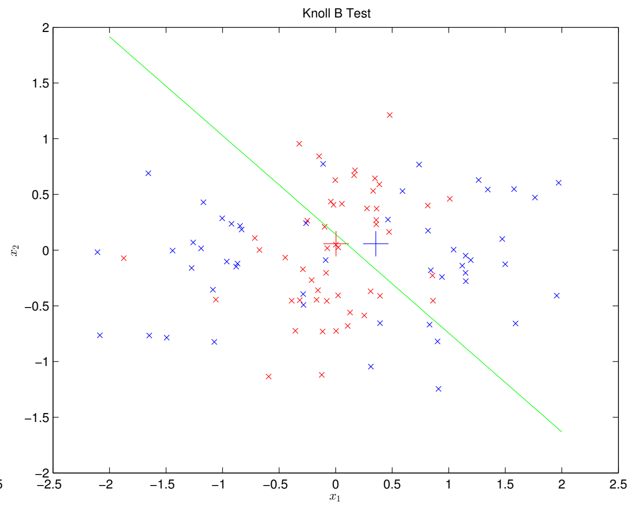
(a)



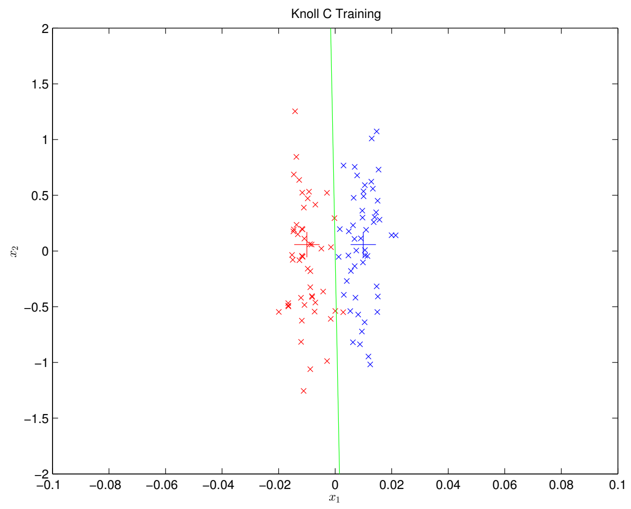
(b)



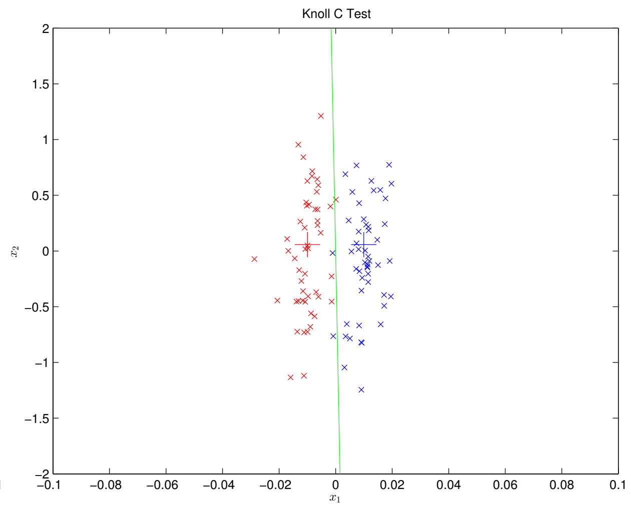
(c)



(d)



(e)



(f)

Figure 3: Visualisation of the training data for each of the KNOLL problems

	KNOLLA		KNOLLB		KNOLLC	
k	test	train	test	train	test	train
1	0.04	-	0.23	-	0.20	-
3	0.02	0.01	0.20	0.13	0.32	0.16
5	0.03	0.01	0.19	0.19	0.34	0.21
7	0.04	0.01	0.19	0.19	0.42	0.21
9	0.03	0.02	0.23	0.20	0.46	0.33
	3	3	5	3	1	3

Table 2: Table of error percentages for k-NN classifier on all three KNOLL problems.

3 Nearest Neighbor Classification

3.1 Nearest Neighbor Classification with Euclidian Metric

The results of the runs of the classifier for different values of k are found in table 2. The bottom field of each column indicates the k for which the classifier gave the lowest error rate.

The error rate is computed by adding up the errors made by the classifier, and then dividing by the test set size. As we have the actual class of a given point (for both test and training sets), this is straightforward to implement.

As a general note, classifying the training set itself will always give us $k = 1$ as the optimal value, which makes sense, because the closest point to a point in both the training and test sets is the point itself, hence the distance will always be zero (and the classification perfect). But it's interesting to see how the classifier behaves when more than one neighbor has to be considered. If a point belonging to one class is surrounded by points in the other class, then it will be misclassified. As can be seen from the results, the optimal value for k in all the training sets is 3, although it should be noted that the error rate for $k = 3, 5$ and 7 in KNOLLA are all 0.1.

In KNOLLA, the error rates are very low (< 0.05) for all the values of k, being lowest when $k = 3$. In KNOLLB and KNOLLC the error rates increase dramatically (≥ 0.19), for KNOLLB the lowest error rates are obtained when $k = 5$ or 7. In KNOLLC the best results are unanimously obtained when $k = 1$.

We can compute an accuracy for the k-NN on the same training data, which for $k = 1$ always gives an error degree of zero, because the nearest neighbour of a given point in the data set is the point itself. But for higher values of k, we can get errors for points surrounded by points from the opposite class.

Both LDA and k-NN perform very well on KNOLLA, as would be expected because the points of the two classes seem cleanly separated in visualisation (fig. 2). They also both perform quite bad on KNOLLB, but here k-NN outperforms LDA, because it only misclassifies about half as many points as does the LDA. This seems reasonable, because k-NN is able to to make use of the fact that the range of $-0.5 \leq x_1 \leq 0.5$ is more densely populated by points of one class, whereas outside of this range the other class is more dominant. This is the kind of distinction that cannot be achieved by a linear decision boundary.

On KNOLLC, on the other hand, LDA clearly outperforms k-NN, achieving near perfect results, while k-NN only performs about as good as it does for KNOLLB. The data in KNOLLC is such that the separation induced by a linear decision boundary gives a higher accuracy than just observing your k nearest neighbours. In other words, a portion of the points lie so close to the decision boundary that they are geometrically closer to the points in the opposite class than points of their own class.

Another comment one could make in terms of computational performance, is that k-NN suffers from the fact that distances must be recomputed for every new point which for a large data set means a lot of computation. LDA has a overhead just in the training phase when computing $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and their common covariance $\boldsymbol{\Sigma}$—after that classification becomes simply a few multiplications and additions, which is nowhere near as computationally demanding as k-NN.

3.2 Changing the Metric

To prove that d is a metric, given

$$d(\mathbf{x}, \mathbf{z}) = \|\mathbf{Mx} - \mathbf{Mz}\|, \text{ where } \mathbf{M} = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix}$$

and $\|\cdot\|$ is the standard L_2-norm (in $\mathbb{R}^2$), we need to verify $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ that 1) $d(\mathbf{x}, \mathbf{y}) \geq 0$; 2) $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$; 3) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (symmetry) and 4) $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^2 : d(\mathbf{x}, z) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

3.2.1 Proof

We need only to observe that $\mathbf{M}$ is a projection $m : \mathbb{R}^2 \to \mathbb{R}^2$, i.e. onto $\mathbb{R}^2$ itself, given by $m(\mathbf{x}) = \mathbf{Mx}$. This immediately gives us all the properties we need, because L_2 is itself a (complete) metric on $\mathbb{R}^2$.

For instance, if we let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x'}, \mathbf{y'}, \mathbf{z'}$ be the result of applying m on $\mathbf{x}, \mathbf{y}, \mathbf{z}$ respectively, we can prove the triangle inequality:

$$\begin{aligned} d(\mathbf{x}, \mathbf{z}) &= \|\mathbf{Mx} - \mathbf{Mz}\| \\ &= \|\mathbf{Mx} - \mathbf{My} + \mathbf{My} - \mathbf{Mz}\| \\ &= \|(\mathbf{x'} - \mathbf{y'}) + (\mathbf{y'} - \mathbf{z'})\| \\ &\leq \|\mathbf{x'} - \mathbf{y'}\| + \|\mathbf{y'} - \mathbf{z'}\| \quad \text{(by property of } L_2 \text{ norm)} \\ &= d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{3}$$

3.2.2 Results

Using d as metric on KNOLLC we obtain the results shown in table 3.

k	training set	test set
1	0.04	-
3	0.02	0.01
5	0.03	0.01
7	0.04	0.01
9	0.03	0.02
	3	3

Table 3: Results for KNOLLC using d as metric

The error rates drop dramatically. This is because our metric "spreads out" the points along the x_1-axis, meaning that the nearest neighbours are more likely to be located along the x_2 axis instead of the x_1. One could also think of it as adding a penalty to the distance along the x_1 axis. This is important because the distance along the x_1 axis is actually what separates the two classes. We've visualised the effect of transforming KNOLLC test set using $\mathbf{M}$ in fig. 4.

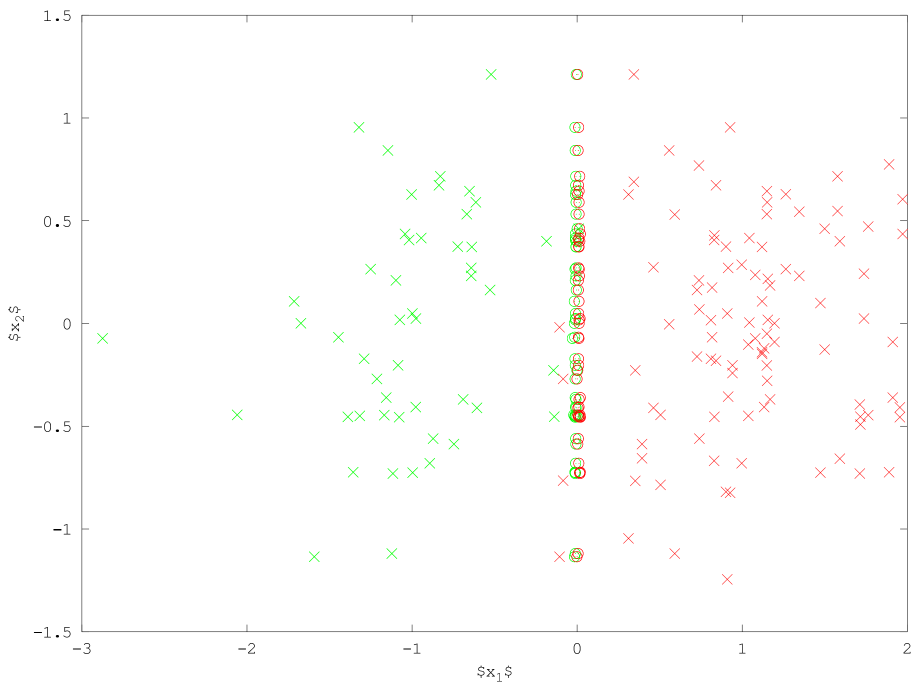


Figure 4: Visualisation of the KNOLLC test set, before and after transforming it using $\mathbf{M}$