Statistical Methods for Machine Learning Assignment 1

Kristina Aluzaite (ljg630)
Mads Ølsgaard (gtw631)
Morten Winther Olsson (hkv518)

18. February 2014

Introduction

II.1 Classification

II.1.1 Linear discriminant analysis & II.1.2 LDA and normalization

For the LDA we are using the mlpy implementation because an error in our own implementation caused all data points to be classified to all classes.

Error rate is simply calculated as 1 minus number of correct classifications over total datapoints. See table

Label	error
Non-normalized, test	0.2105
Non-normalized, train	0.14
Normalized, test	0.2105
Normalized, train	0.14

Table 1: Results of Iris classification

Given that the model is fitted around the training data, it is no surprise that the training data has a lower error rate than when testing on the test set.

We see no change in normalizing the data. This is because LDA assumes all classes to be normally distributed and with the same covariance. So normalizing data won't affect the covariance matrix.

II.1.3 Bayes optimal classification and probabilistic classification

The Bayes Optimal is defined as the classifier that gives the lowest risk.

$argmax(v_j \in V) = \sum_{h_i \in H} p(v_j|h_i)P(h_i|D)$

System that classifies new instances based on the equation above are called Bayes Optimal Classifiers.

In our case, the hypothesis set $H = \{h_1, h_2\}$, where $h_1 = 0$, $h_2 = 1$, and $P(h_1) = 0.25$, while $P(h_2) = 0.75$.

Let $V = \{\bigoplus, \bigodot\}$

suppose: $\bigoplus$ `if` 1
$\bigodot$ `if` 0

Given the data set D:
$P(h_1, D) = 0.25$

$P(h_2, D) = 0.75$

Probabilities of classifying new instances as $\bigoplus$ or $\bigodot$ for the h_1 or h_2
$(\bigodot, h_1) = 1$
$(\bigodot, h_2) = 0$
$(\bigoplus, h_1) = 0$
$(\bigoplus, h_2) = 1$

Therefore, multiplying and summing all the the conditionals we get:
$\sum_{h_i \in H} P(\bigoplus |h_i) P(h_i|D) = 0.75$
$\sum_{h_i \in H} P(\bigodot |h_i) P(h_i|D) = 0.25$

And the most probable classification of the new instance is:
$argmax(v_j \in \{\bigoplus, \bigodot\}) = \sum_{h_i \in H} p(\bigoplus |h_i) P(h_i|D) = \bigoplus$

The Bayes risk is defined as a minimum risk over all possible measurable functions H.
For the classifier above, the bayes risk is $R = 0.25$, as the $P(h_2 = 0) = 0.25$

The risk of the probabilistic classifier given the data is a combination of probabilities of classification for different states. We sum the products of the event probability times the risk of the false prediction.

for $h_1 = 0$, the probability of a false prediction is 0.75 and the hypothesis $h(x) = 0$ occurs 0.25 of the time; and
for $h_0 = 1$ the probability of a false prediction is 0.25 and the hypothesis is true 0.75 of the time.

Hence, the risk of the classifier is $0.75 * 0.25 + 0.25 * 0.75 = 0.375$.

II.2 Regression: Sunspot Prediction

II.2.1 Maximum likelihood solution

We use a simple linear model for our regression of the sunspot data

$$y(\vec{x}, \vec{w}) = w_0 + w_1 x_1 + ... + w_D x_D \tag{1}$$

To facilitate this we set the basis function $\phi_j(x) = x_j$ where j denotes the j'th value in the observation vector x. So $j \in \{0 \dots D\}$. For $j = 0$, 1 is returned. In the code this is facilitated by adding a column of 1's to the training and testing data.

This gives us the design matrix Φ,

$$\Phi = \begin{pmatrix} 1 & X_{1,1} & X_{1,2} & \cdots & X_{1,j} \\ 1 & X_{2,1} & X_{2,2} & \cdots & X_{2,j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{l,1} & X_{l,2} & \cdots & X_{l,j} \end{pmatrix} \tag{2}$$

Where l is the number of observations.

$\vec{W}$ is found by taking the pseudo inverse of the design matrix using numpy's `np.linealg.pinv()` function.

Results are tested using the *root mean square* (RMS) error seen in table . Selection 3 yields the lowest error and is therefore the best selection.

This makes good sense, since we know that sunspots follow an 11 year cycle and selection 3 is the only selection that has data from before and after 11 years before any given year.

Selection	RMS
1	35.47
2	28.83
3	18.77

Table 2: Results of sun-spot prediction using Maximum likelihood solution

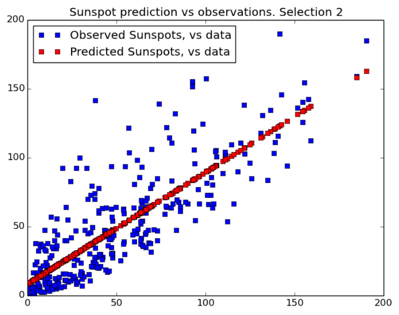


Figure 1: Showing predictions and actual observations as a function of data from selection 2. We clearly see the linearity of the model, but also that the observations do in fact center around a linear line.

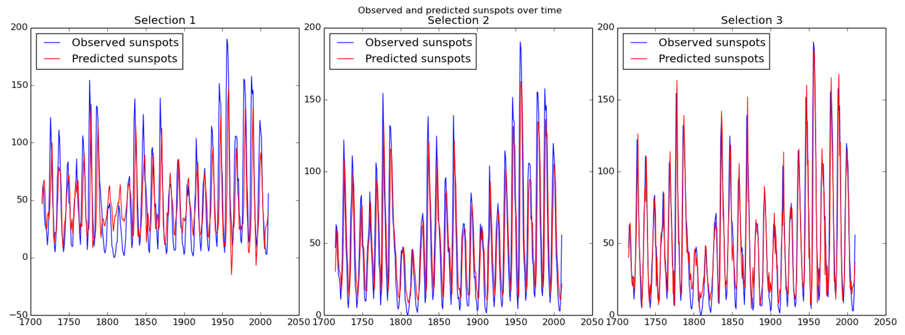
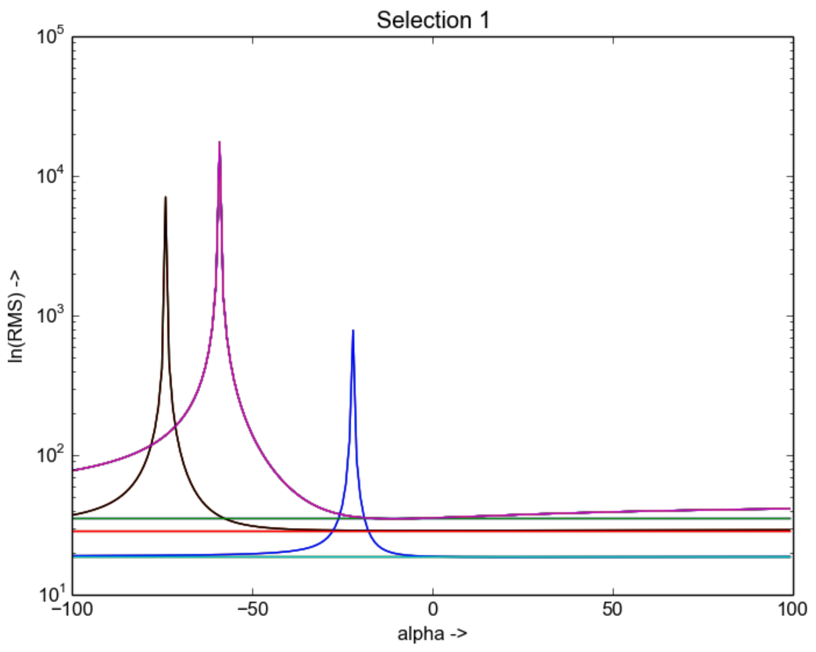


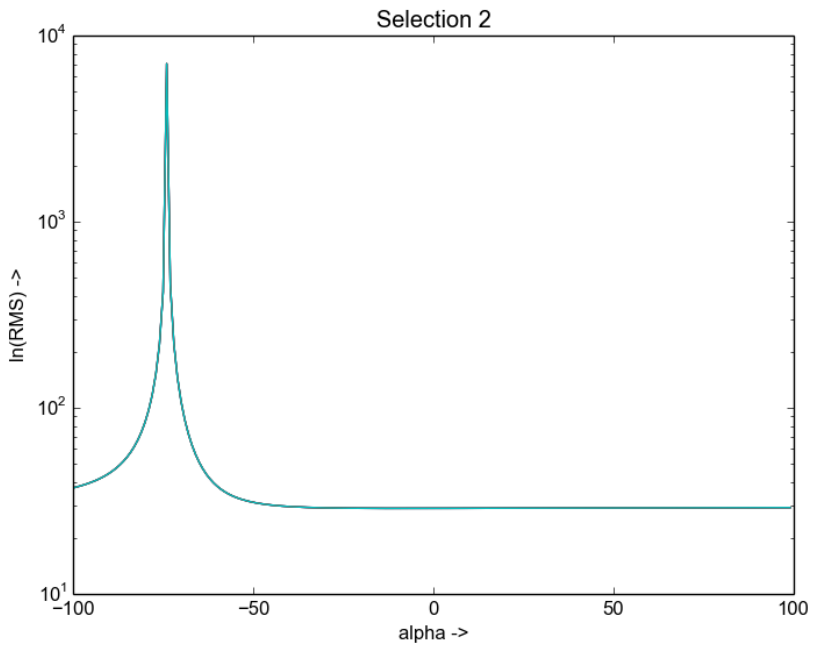
Figure 2: Showing how the different selections fit with acutal observations. We see all models account for the recurring ups and downs, but only selection 3 matches all hills and valleys well.

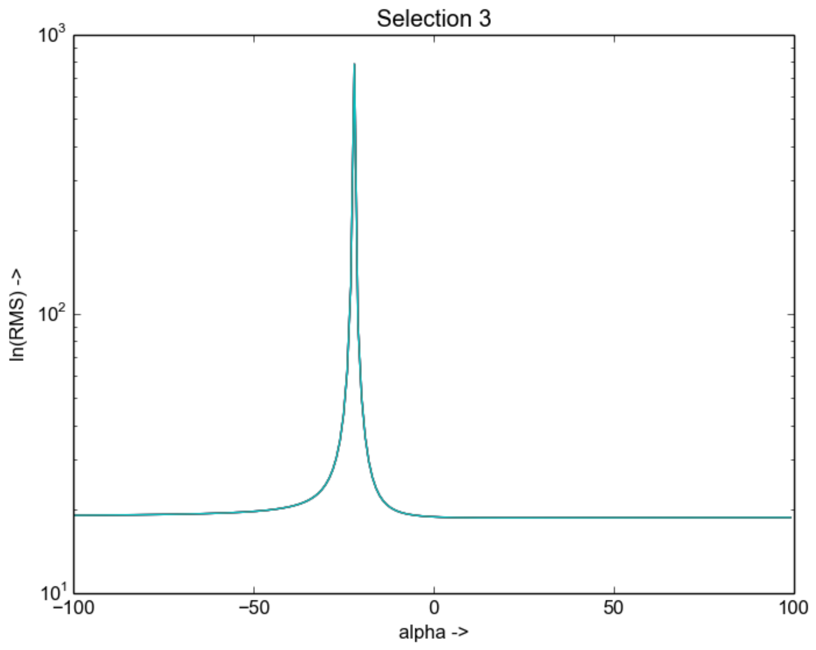
II.2.2 Maximum a posteriori solution

Done, see `src/II.2.2.py`.

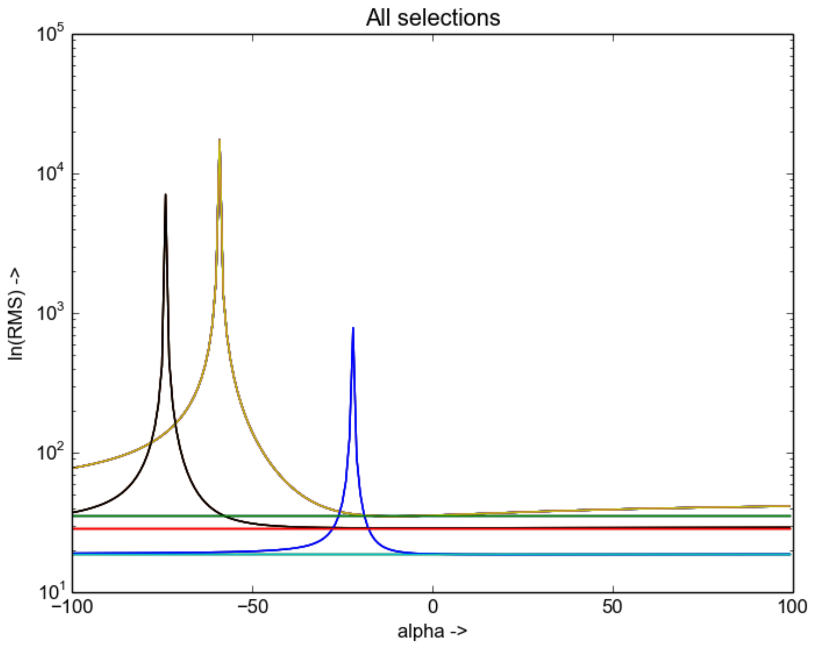
The plots are:







and combined, along with the RMS of the ML results:



It seems clear that selection 3 provides the best prediction, and selection 1 the worst. The ML solution scores slightly worse for a narrow range of alpha (-11 for Selection 1, -8 for Selection 2 and 22 for Selection 3), but much better for most choices of alpha.

II.2.3 Weighted sum-of-squares (based on CB Ex. 3.3)

$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} r_n\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$

To find the minimum we solve $\frac{\delta}{\delta w_i}E_D = 0$ for all i:

$\frac{\delta}{\delta w_i}E_D = \sum_{n=1}^{N} r_n(t_n - \mathbf{w}^T\phi(\mathbf{x}_n)) \cdot (-\phi_i(\mathbf{x}_n)) = 0$ for all i

Since $\phi(\mathbf{x})^T = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots, \phi_{n-1}(\mathbf{x}))$, we get

$\sum_{n=1}^{N} r_n\mathbf{w}^T\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T - \sum_{n=1}^{N} r_n t_n\phi(\mathbf{x}_n)^T = 0$

or

$0 = \mathbf{w}^T\sum_{n=1}^{N} r_n\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T - \sum_{n=1}^{N}(r_n t_n\phi(\mathbf{x}_n)^T)$ (*)

Setting $\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \vdots & & & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$ we rewrite (*) as

$0 = \mathbf{w}^T(\mathbf{\Phi}^T\mathbf{r}\mathbf{\Phi}) - \mathbf{r}\mathbf{t}^T\mathbf{\Phi}$

$\Rightarrow \mathbf{w}^T(\mathbf{\Phi}^T\mathbf{r}\mathbf{\Phi}) = \mathbf{r}\mathbf{t}^T\mathbf{\Phi}$

$\Rightarrow (\mathbf{\Phi}^T\mathbf{r}\mathbf{\Phi})^T\mathbf{w} = (\mathbf{\Phi}^T\mathbf{r}\mathbf{\Phi})\mathbf{w} = \mathbf{\Phi}^T\mathbf{r}\mathbf{t}$

$\Rightarrow \mathbf{w}^* = (\mathbf{\Phi}^T\mathbf{r}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{r}\mathbf{t}$

.1 Data dependent noise variance

If a data point is far from the expected value, the weighting can be used to give that point less importance when minimizing the error.

.2 Replicated data points

Replicated data points will usually hold undue importance, but the weights can be chosen lower for the replicated points to counter this.