1 Regression

1.1 Maximum Likelihood Solution

First we load the dataset, give it reasonable column names, and split it up into a training and a testing subset.

We attempt to predict body fat using two models. The first one is based on weight, and chest, abdomen, and hip circumferences, and the second is based solely on abdominal circumference.

We make a design matrix for every selection, and then find $\mathbf{w}_{\mathrm{ML}}$ by $\mathbf{w}_{\mathrm{ML}} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$ where $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}$ is the pseudo-inverse of $\mathbf{\Phi}$.

Once we have found $\mathbf{w_{ML}}$ we can apply it to the test set, via $y(\mathbf{x}, \mathbf{w}) = \mathbf{w^T}\phi(\mathbf{x})$, and we can calculate the RMS via

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(t_n - y(\mathbf{x_n}, \mathbf{w}))^{\mathbf{2}}}$$

Thus we get an RMS of 4.07 using the first model, and an RMS of 4.4 for the second model. The first, complex, model yields slightly better results, though it seems that using the abdominal circumference alone will also yield fair results.

For posterity, we have also calculated the error on the training set. For the two models, we get RMS of 4.52, and 4.81. The fact that the using the train error is not smaller than the test error, likely points to the fact that we are not over-fitting.

1.2 Maximum a posteriori Solution

The random assignment into training and test data implies that each run will generate different results. However generally speaking, model 1 (weight, chest, abdomen and hip) provides better out-of-sample prediction relative to model 2 (abdomen) in terms of lower values for the root mean square deviation (RMS). This is reflected in figure 1.

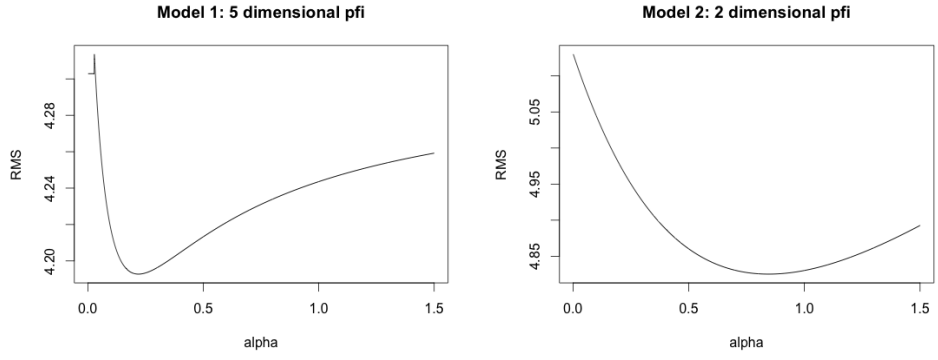

Figure 1: Maximum a posteriori estimations of model 1 and model 2

Using MAP, with an isotropic uniform prior, we obtained an RMS error of 4.19 for model 1, and 4.84 for model 2 as shown in figure 1. We did not perform cross-validation, and

variations in the RMS based on the choice of training and test data set, were greater than differences between MAP, and ML methods, though MAP methods seemed to perform, on average, better (data not shown).

We will also note that in the majority of cases, we did not observe a parabolic curve with some minimum, between RMS and α.

1.3 Theory behind MAP

We will attempt to show $p(w|t) = N(w|m_n, S_n)$, where, $m_n = S_n(S_0^{-1}m_0 + \beta\Phi^T\mathbf{t})$ and $S_n = S_0^{-1} + \beta\Phi^T\Phi$. In general $p(w|t) = \frac{p(t|w)p(w)}{p(t)}$, as per Bayes' Rule.

$$p(w|t) \propto p(t|w)p(w) = \prod_n^N N(t_n|w^T\phi(x_n), \beta^{-1}) \cdot N(w|m_0, S_0^{-1})$$

The constant denominator being absorbed in the proportionality sign. Explicitly writing the multivariate gaussians above:

$$p(w|t) \propto \prod_n^N \exp[-1/2(t_n - w^T\phi(x_n))^T\beta^{-1}(t_n - w^T\phi(x_n))] \cdot \exp[-1/2(w-m_0)^T S_0^{-1}(w-m_0)]$$

The product of the exponents is equivalent to the exponent of the sum, so we take the exp outside the product sign.

$$p(w|t) \propto \exp \sum_n^N [-1/2(t_n - w^T\phi(x_n))^T\beta^{-1}(t_n - w^T\phi(x_n))] \cdot \exp[-1/2(w-m_0)^T S_0^{-1}(w-m_0)]$$

The sum is with respect to n. Transforming the target into a target vector, and the basis function $\phi(x)$ into the design matrix Φ we obtain:

$$p(w|t) \propto \exp[-1/2(\mathbf{t} - w^T\Phi)^T\beta^{-1}(\mathbf{t} - w^T\Phi)] \cdot \exp[-1/2(w - m_0)^T S_0^{-1}(w - m_0)]$$

Combining the exponents:

$$p(w|t) \propto \exp[-1/2(\mathbf{t} - w^T\Phi)^T\beta^{-1}(\mathbf{t} - w^T\Phi) - 1/2(w - m_0)^T S_0^{-1}(w - m_0)]$$

expanding the brackets:

$$p(w|t) \propto \exp[-1/2(\mathbf{t}^T\beta^{-1} - w\Phi^T\beta^{-1})(\mathbf{t} - w^T\Phi) - 1/2(w^T S_0^{-1} - m_0^T S_0^{-1})(w - m_0)]$$

expanding further:

$$p(w|t) \propto$$

$$\exp[-1/2(\mathbf{t}^T\beta^{-1}\mathbf{t} - w\Phi^T\beta^{-1}\mathbf{t} - \mathbf{t}^T\beta^{-1}w^T\Phi + w\Phi^T\beta^{-1}w^T\Phi)) - 1/2(w^T S_0^{-1}w - m_0^T S_0^{-1}w + w^T S_0^{-1}m_0 - m_0^T S_0^{-1}m_0)]$$

We can now absorb everything not dependent on w into the proportionality sign as it is a constant, and complete the squares showing that

$$p(w|t) \propto \exp[-1/2w^T(S_0^{-1})w + w^T(S_0^{-1} + \beta\Phi^T\Phi)(S_0 m_0 + \beta\Phi\mathbf{t})]$$

where via completing the squares again:

$$p(w|t) \propto -1/2(w - \mu)^T\Sigma^{-1}(x - \mu)$$

with (μ,Σ^{-1}) are specified by (m_n,S_n).

2 Linear Discriminant Analysis

We load the data KnollA, KnollB, and KnollC, and create 2-dimensional plots, while coloring by index/classification as seen in figure 2. KnollA and KnollC are very similar sets, with however, their coordinate points are differently scaled in terms of their x values. As is apparent from the training data, KnollA and KnollC are divided by classification on either side of more or less exactly the value $x = 0$, while KnollB has a more scattered representation of the data points where points of both indexes appear on either sides of the x axis.

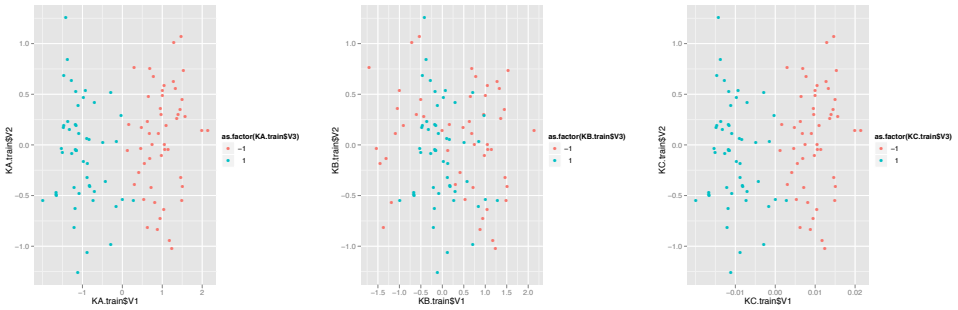

Figure 2: Plotted points of the training sets for KnollA, KnollB and KnollC

To implement the LDA on all of the datasets, the R function `lda` from the `MASS` library was used. Priors were set to 0.5 for both classes, as those were the empirical frequencies of the classes in the training set:

$$\hat{\pi}_k = \frac{\sum \text{samples in class } k}{\sum \text{total samples}}$$

We get 98% accuracy on datasets KnollA and KnollC, and accuracy of 58% for KnollB. Looking at figure 2, it seems obvious a linear classifier would fare poorly with the KnollC data. In figure 3, we show the linear classification boundary of the three datasets.

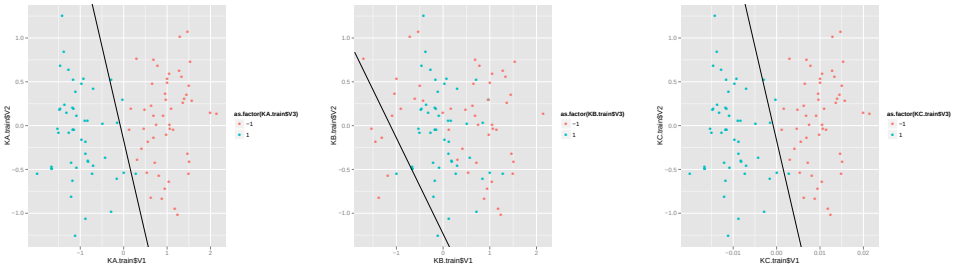

Figure 3: Plotted points of the training sets for KnollA, KnollB and KnollC

3 Nearest Neighbor Classication

3.1 Nearest Neighbor Classication with Euclidean Metric

The k-nearest neighbor classifier (k-NN) is implemented by calculating the distances for each point in the test set from each point in the training set. Said distances are then mapped as

key-value pairs to the classifications of their respective points in the training set, where the distance is the key, and the classification, the value mapped to it. Once the mappings are obtained, the map is sorted to ascend by distance. Once this is done, the sum of the first k (where k is an odd integer) classifications (which are either -1 or 1) is computed, yielding the result as to which class the test point falls into by the classifications of the majority of its neighbors (the points with the lowest distances from said point). Please reference the code for the actual algorithm implemented.

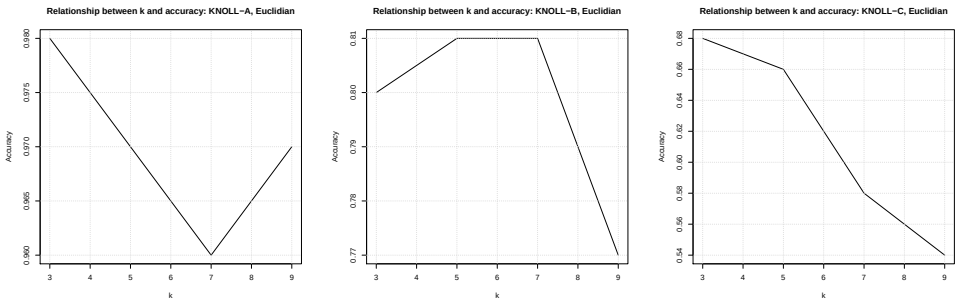

Figure 4: k-NN applied to datasets KnollA, KnollB and KnollC using the Euclidian metric

As is apparent from the graphs (see figure 4), a k value of around 3 seems optimal. A low k-value results in a complex model which captures the variation in the data well, but tends to over fit, while a large k-value is less likely to exhibit over fitting, but is more likely to miss local features of the data. This is known as the bias-variance tradeoff, and it haunts many machine learning algorithms.

We also see that KNN outperforms LDA in the KnollB data set , but performs poorly on the Knoll C dataset; it appears that a straightforward Euclidean distance measure is a poor fit when the data is rescaled.

3.2 Changing the Metric

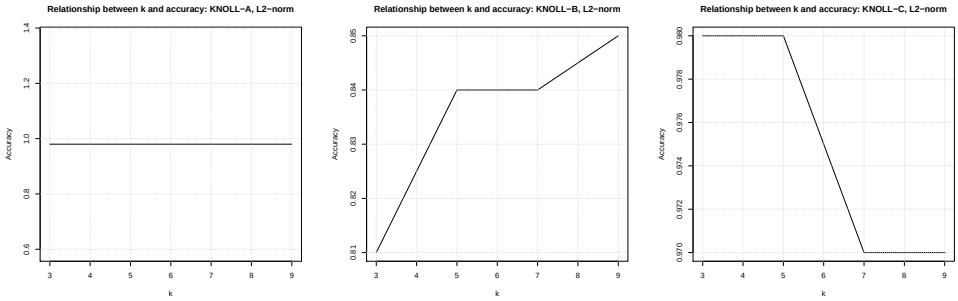

Figure 5: k-NN applied to datasets KnollA, KnollB and KnollC using the L_2-norm metric

Here we see noticeable improvement in the Knoll C dataset.

Proving the metric:

As a metric is defined as a function with the following properties:

1. $d(x,y) \geq 0$ with $d(x,y) = 0 \iff x = y$
2. $d(x,y) = d(y,x)$
3. $d(x,y) \leq d(x,z) + d(z,y)$

the proof of the metric is as follows, in terms of the above properties:

ad1)

The proof of property 1 follows directly from the fact that the matrix M has full rank. This implies that the $d(x,y)$ can only be obtained with $x - z = 0 \Leftrightarrow x = z$.

ad2)

$d(x,y) =$

$$\left|\left|\begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} - \begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}\right|\right| = \sqrt{(100_{x_1})^2 + (x_2)^2 + (-100_{y_1})^2 + (-y_2)^2} =$$

$100(x_1 + y_1) + x_2 + y_2$

$d(y,x) =$

$$\left|\left|\begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} - \begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}\right|\right| = \sqrt{(100_{y_1})^2 + (y_2)^2 + (-100_{x_1})^2 + (-x_2)^2} =$$

$100(x_1 + y_1) + x_2 + y_2$

$Q.E.D.$

ad3)

$$d(x,y) \leq (x,z) + d(z,y) \Leftrightarrow \left|\left|\begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} - \begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}\right|\right| =$$

$$100(x_1{+}y_1){+}x_2{+}y_2 \leq \left|\left|\begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} - \begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix}\right|\right| + \left|\left|\begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}z_1\\ z_2\end{bmatrix} - \begin{bmatrix}100 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}\right|\right| =$$

$$\sqrt{(100_{y_1})^2 + (y_2)^2 + (100_{z_1})^2 + (-z_2)^2} + \sqrt{(100_{z_1})^2 + (z_2)^2 + (100_{y_1})^2 + (-y_2)^2} =$$

$$(100(x_1 + z_1) + x_1 + z_2) + (100(z_1 + y_1) + z_1 + y_2) =$$

$$100(x_1{+}y_1{+}2_{z_1}){+}(x_2{+}y_2{+}2_{z_2}) \Leftrightarrow 100(x_1{+}y_1){+}x_2{+}y_2 \leq 100(x_1{+}y_1{+}2_{z_1}){+}(x_2{+}y_2{+}2_{z_2})$$

$\Leftrightarrow 0 \leq 200z_1 + 2z_2$, where $z_1 \geq 0$ and $z_2 \geq 0$

Q.E.D.