MS1b: SDM - Problem Sheet 7

1. Consider a neural network with two hidden layers. Let x be an observation vector with corresponding label y. The unit activations in first hidden layer are computed as

$$h_j = f(W_j x)$$

where W is a matrix of parameters, W_j is the jth row, f is some non-linear differentiable function and j runs over indices of units in first hidden layer. Similarly the second layer activations are

$$s_k = f(U_k h)$$

with U_k the kth row of weight matrix U, k runs over units in second hidden layer and h the vector of activations in first hidden layer. Finally the predicted output is then

$$\hat{y} = Vs$$

where V is a row vector of parameters.

Suppose we use the squared-loss $(y - \hat{y})^2$. Calculate the derivative of the loss with respect to parameters V_k, U_{kj}, and W_{ji}, showing that these derivatives can be computed recursively using a back-ward pass through the network.

2. Consider selecting between multi-level attributes, in order to make a multi-way split (at node t say), using the expected decrease in the Gini index to decide the choice of attribute. Consider a generic attribute $A \in \{a_1, a_2, ..., a_L\}$ with L levels, for data in two classes, so that $K = 2$, and $p_k,\ k = 1, 2$, is the class distribution at node t. The Gini index of impurity is $2p_1(1-p_1)$. If we split using attribute A (and are not using dummy variables) we will have an L-way split.

Consider data (X, Y)=(data-vector, class label) passing through the tree, and denote by X_t, Y_t a generic pair reaching node t, so that $p_k = \Pr(Y_t = k)$. Denote by $q_i = \Pr(A(X_t) = a_i),\ i = 1, 2, ..., L$, the distribution of the attribute-level of X_t. Let $p_{k|i} = \Pr(Y_t = k | A(X_t) = a_i)$ be the distribution of class labels conditioned on the attribute level of X_t. Suppose in fact $N_i = n_i$ of $N = n$ data vectors reaching node t possess attribute-level a_i, $N^k = n^k$ are in class k, and $N_i^k = n_i^k$ are in class k with attribute level a_i.

   (a) Explain why $N_i|N = n$, $N^k|N = n$ and $N_i^k|N_i = n_i$ have respectively multinomial, binomial and binomial distributions, and give the parameters of these distributions in terms of q_i, p_k and $p_{k|i}$.

   (b) The parameters p_k, q_i and $p_{k|i}$ are unknown. The Gini index is computed using the plug-in estimates $\hat{p}_k = n^k/n,\ \hat{q}_i = n_i/n$ and $\hat{p}_i^k = n_i^k/n_i$ respectively. Calculate

the expected change in the Gini index between node t and its L child-nodes, conditioned on $N = n$ data vectors at node t.

(c) Suppose the attribute-levels are actually uninformative of class, so that $p_{k|i} = p_k$. Show that, conditioned on $N = n$, the expected decrease in the Gini index is equal

$$2p_1(1 - p_1)(L - 1)/n.$$

(d) Is this attribute selection criterion biased in favor of multi-level attributes?

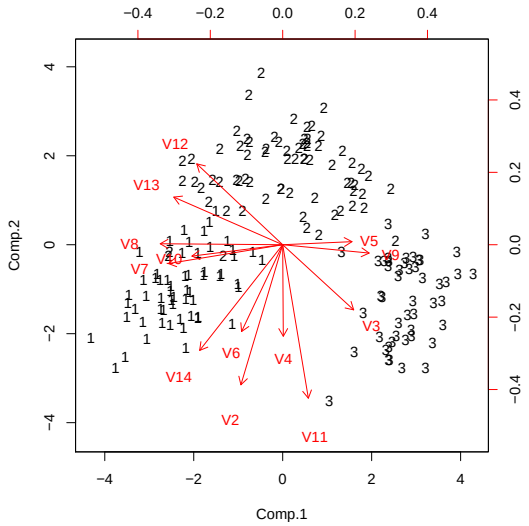
3. The files `wine.txt` is available on the course website. You can again read it directly with

```
td <- read.table(
        "http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/wine.data",
        sep=",")
```

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The first column contains the class label (1, 2 or 3), which is denoting the grower the wine came from. The goal is to predict the grower (the class 1, 2 or 3) given, some predictor variables. These are, in columns 2-14,
2) Alcohol
3) Malic acid
4) Ash
5) Alcalinity of ash
6) Magnesium
7) Total phenols
8) Flavanoids
9) Nonflavanoid phenols
10) Proanthocyanins
11) Color intensity
12) Hue
13) OD280/OD315 of diluted wines
14) Proline.

(a) Make a biplot using the `scale=0` option, and then use the `xlabs=as.numeric(td$Type)` option in `biplot()` to label points by their `$Type`. The output should look like:



(b) Now make a classification tree analysis, and relate the decision rules discovered there to the projections of the original variable axes displayed in the biplot.

(c) Now produce a Random Forest fit, calculating the out-of-bag estimation error and compare with the tree analysis. You could start like:

```
library(randomForest)
rf <- randomForest(td[,2:14],td[,1],importance=TRUE)
print(rf)
```

Experiment with the parameter `mtry`, the random number of variables the best splitpoint is searched over. Use `varImpPlot` to determine what are the most important variables.