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 and k runs over units in second 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.

Answer: By chain rule, the derivative wrt V_k is:

$$2(\hat{y} - y)s_k$$

the derivative wrt U_{kj} is:

$$2(\hat{y} - y)V_k f'(U_k h)h_j$$

and finally the derivative wrt W_{ji} is:

$$2(\hat{y} - y)\sum_k V_k f'(U_k h)U_{kj} f'(W_j x)x_i$$

Thus in the backward pass we

(a) compute $2(\hat{y} - y)$, at which point we can get derivative wrt V_k,

(b) then compute $2(\hat{y} - y)V_k f'(U_k h)$, at which point we can get derivative wrt U_{kj},

(c) finally compute $2(\hat{y} - y)\sum_k V_k f'(U_k h)U_{kj} f'(W_j x)$ at which point we can get derivative wrt W_{ji}.

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 $p_{k|i}$.

Answer: Given the number of data vectors at node t, each data vector at node t has attribute $A(x) = a_i$ with probability q_i. If the data vectors are independent, then the number with level a_i is multinomial with parameters $q_1, ..., q_L$. Since there are just two classes in this problem, similar reasoning gives $N^k|N = n \sim \text{Bin}(n, p_k)$ and $N_i^k|N_i = n_i \sim \text{Bin}(n_i, p_{k|i})$.

(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.

Answer:

$$\Delta_{\text{Gini}} = 2p_1(1 - p_1) - 2\sum_{i=1}^{L} q_i p_{1|i}(1 - p_{1|i})$$

and we will form

$$\hat{\Delta}_{\text{Gini}} = \frac{2n^1}{n}\left(1 - \frac{n^1}{n}\right) - \sum_{i=1}^{L} \frac{n_i}{n}\frac{2n_i^1}{n_i}\left(1 - \frac{n_i^1}{n_i}\right).$$

We compute $E(\hat{\Delta}_{\text{Gini}})$ with variation over n^1, n_i^1, and n_i, $i = 1, ..., L$. Identities $\sum np^n = p\,(d/dp)\sum p^n$ are handy if you cant recall the sums.

$$\begin{aligned} E\left(2\frac{N^1}{n}\left(1 - \frac{N^1}{n}\right)\right) &= \frac{2}{n^2}E(N^1(n - N^1)) \\ &= \frac{2}{n^2}\sum_{n^1=0}^{n}\binom{n}{n_1} n^1(n - n^1)p_1^{n_1}(1 - p_1)^{n-n_1} \\ &= 2\frac{(n-1)}{n}p_1(1 - p_1) \end{aligned}$$

$$E\left(\frac{N_i}{n}\left(2\frac{N_i^1}{N_i}\left(1-\frac{N_i^1}{N_i}\right)\right)\right) = \frac{2}{n}E\left(\frac{1}{N_i}E(N_i^1(N_i-N_i^1)|N_i)\right)$$
$$= \frac{2p_{1|i}(1-p_{1|i})}{n}E(N_i-1)$$
$$= \frac{2p_{1|i}(1-p_{1|i})}{n}(nq_i-1)$$

so

$$E(\hat{\Delta}_{\text{Gini}}) = 2\frac{(n-1)}{n}p_1(1-p_1) - 2\sum_{i=1}^{L}\frac{p_{1|i}(1-p_{1|i})}{n}(nq_i-1).$$

(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

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

Answer: In this case $p_{1|i} = p_1$ and

$$E(\hat{\Delta}_{\text{Gini}}) = 2n(n-1)p_1(1-p_1) - 2\sum_{i=1}^{L}\frac{p_{1|i}(1-p_{1|i})}{n}(nq_i-1)$$
$$= 2\frac{(n-1)}{n}p_1(1-p_1) - 2p_1(1-p_1)\frac{n-L}{n}$$
$$= 2p_1(1-p_1)\frac{(L-1)}{n}$$

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

Answer: Yes. If we have two uninformative attributes, A with levels $1, ..., L$ and A' with fewer levels $1, ..., L'$, $L' < L$, selection on the Δ_{Gini}-criterion will be biased towards A, as A has more attribute-levels so its Δ_{Gini} will typically be larger.

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

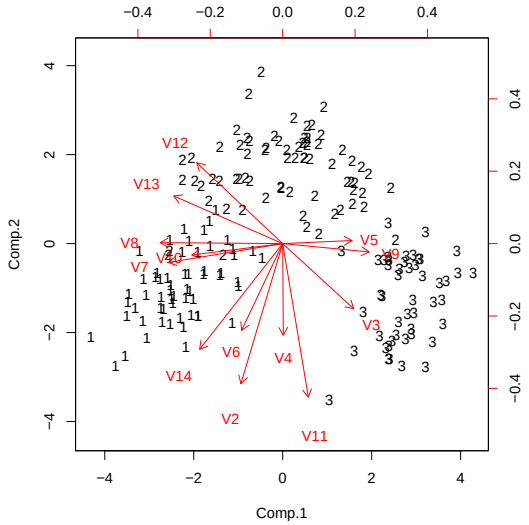
```
td <- read.table(
        "http://www.stats.ox.ac.uk/%7Eteh/teaching/datamining/wine.data",
        sep=",")
```

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:



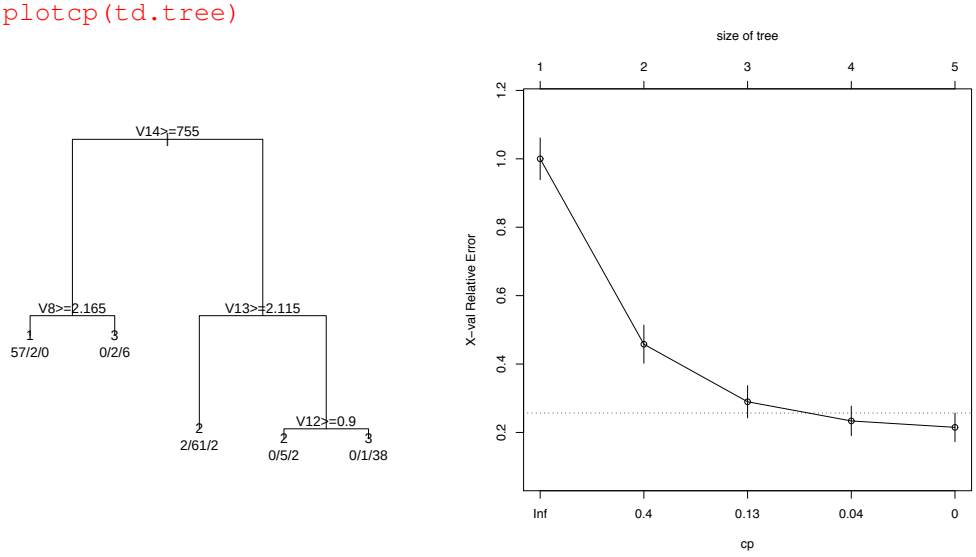
Answer:

```
library(rpart)
library(MASS)
td <- read.table("http://www.stats.ox.ac.uk/%7Eteh/teaching/datam
names(td)[1] <- "Type"
td[,1] <- as.factor(td[,1])

td.pc <- princomp(td[,2:14],scale=0,cor=T)
biplot((td.pc),xlabs=as.numeric(td$Type))
```

(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.

Answer:

```
td.tree <- rpart(Type~.,td,parms=list(split='gini'),
                 control = rpart.control(xval=10),method="class")
plot(td.tree)
text(td.tree,use.n=TRUE,digits=4,pretty=0);
plotcp(td.tree)
```



The tree was pruned at CP=0.04 (reading from the graph, and choosing the smallest tree withing 1 sd of the minimum). The relative cross validation error rate was 0.23 relative to an apparent/training root error rate of 0.6.

Referring to the biplot, variable V14 increases towards the 1's. The tree routes TRUE to the left. Splitting on large V14 we capture all the 1's and also a few 3's from small values of the 2nd principal component (PC). Variable V8 is aligned with the 1st PC (or rather, it's -ve), so splitting on large values of V8 (-ve 1st PC) removes these residual 3's. Variable 13 has a component in the direction of positive 2nd PC, which splits the 2's and 3's routed to the right at the first split. Note that this is really just a consistency check - we cannot trust that the points at the end of a particular variable vector in a biplot will have a large value of that component (since there are more than two variable vectors in the biplot, they are linearly dependent).

(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.

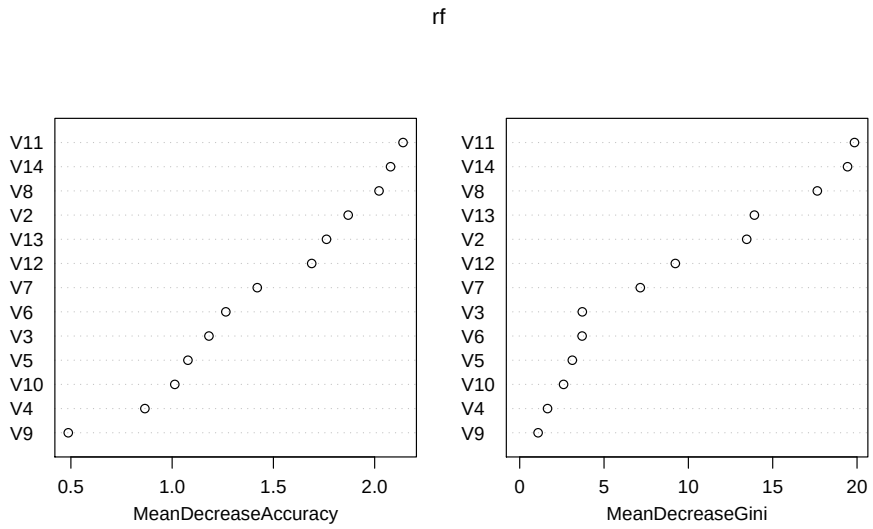
Answer: The out-of-bag estimation error is lower than for a single tree, as the following output shows

```
> print(rf)

Call:
 randomForest(x = td[, 2:14], y = td[, 1], importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.69%
Confusion matrix:
   1  2  3 class.error
1 59  0  0  0.00000000
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

The variable importance plot is obtained by

```
> varImpPlot(rf)
```

Two measures of importance are computed, which give a way of interpreting the forest (which is harder than for a single tree). The left panel shows the decrease in accuracy if permuting variable x. A larger decrease in accuracy means a higher "importance".



Experimenting with the number of variables over which the best splitpoint is searched,

one obtains in all cases better values than with a single tree, even though the choice matters. If searching over 9 variables, we get

```
> rf <- randomForest(td[,2:14],td[,1],mtry=9)
> print(rf)
> > >
Call:
 randomForest(x = td[, 2:14], y = td[, 1], mtry = 9)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 2.25%
Confusion matrix:
   1  2  3 class.error
1 58  1  0  0.01694915
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

```
rf <- randomForest(td[,2:14],td[,1],mtry=6)
print(rf)
```

while searching over 3 variables (the default) gives a similar answer as the initial Random Forests (Remember that the trees are random as the variables over which the best splitpoint is searched are chosen randomly. The answer will only be the same if the number of trees is chosen very large.)

```
> rf <- randomForest(td[,2:14],td[,1],mtry=6)
> print(rf)
> > >
Call:
 randomForest(x = td[, 2:14], y = td[, 1], mtry = 6)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 2.25%
Confusion matrix:
   1  2  3 class.error
1 58  1  0  0.01694915
2  1 68  2  0.04225352
3  0  0 48  0.00000000
```

This search can be done more systematically but the default value is close to optimal for this dataset.