Statistical Machine Learning HT 2019 - Problem Sheet 4

Please send any comments or corrections to Pier Palamara (email on course website).

---

1. **Tree Learning: Gini.** Consider a binary classification problem with $\mathcal{Y} = \{1, 2\}$. We are at a node t in a decision tree and would like to split it based on Gini impurity. Consider a categorical attribute A with L levels, i.e., $x^{(A)} \in \{a_1, a_2, ..., a_L\}$. For a generic example (X_i, Y_i) reaching node t, denote:

$$\begin{aligned} p_k &= \mathbb{P}\left(Y_i = k\right), \ k = 1, 2, \\ q_\ell &= \mathbb{P}\left(X_i^{(A)} = a_\ell\right), \ \ell = 1, \ldots, L, \\ p_{k|\ell} &= \mathbb{P}\left(Y_i = k | X_i^{(A)} = a_\ell\right), \ k = 1, 2, \text{ and } \ell = 1, \ldots, L. \end{aligned}$$

Thus, the population Gini impurity is given by $2p_1(1 - p_1)$. Further, assume $N = n$ examples $\{(X_i, Y_i)\}_{i=1}^n$ have reached the node t, and denote

$$\begin{aligned} N^k &= \left|\{i : \ Y_i = k\}\right|, \ k = 1, 2, \\ N_\ell &= \left|\left\{i : \ X_i^{(A)} = a_\ell\right\}\right|, \ \ell = 1, \ldots, L, \\ N_{k|\ell} &= \left|\left\{i : \ Y_i = k \text{ and } X_i^{(A)} = a_\ell\right\}\right|, \ k = 1, 2, \text{ and } \ell = 1, \ldots, L. \end{aligned}$$

   (a) Assuming data vectors reaching node t are independent, explain why $N_\ell | N = n$, $N^k | N = n$ and $N_{k|\ell} | N_\ell = n_\ell$ have respectively multinomial, binomial and binomial distributions with parameters q_ℓ, p_k and $p_{k|\ell}$.

   (b) If we split using attribute A (and are not using dummy variables) we will have an L-way split and the resulting impurity change will be

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

   The parameters p_k, q_ℓ and $p_{k|\ell}$ are unknown, however. The Gini impurity estimate $\hat{\Delta}_{\text{Gini}}$ is thus computed using the plug-in estimates $\hat{p}_k = N^k/N$, $\hat{q}_\ell = N_\ell/N$ and $\hat{p}_{k|\ell} = N_{k|\ell}/N_\ell$ respectively. Calculate the expected estimated impurity change $\mathbb{E}[\hat{\Delta}_{\text{Gini}} | N = n]$ between node t and its L child-nodes, conditioned on $N = n$ data vectors reaching node t.

   (c) Suppose the attribute-levels are actually uninformative about the class label, so that $p_{k|\ell} = p_k$. Show that, conditioned on $N = n$, the expected estimated Gini impurity change is then equal

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

   (d) Is this attribute selection criterion biased in favor of attributes with more levels?

2. **Neural networks: backpropagation.** Recall the definition of a one-hidden layer neural network for binary classification in the lectures. The objective function is L_2-regularized log loss:

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) + \frac{\lambda}{2} \left(\sum_{jl} (w_{jl}^h)^2 + \sum_{l} (w_l^o)^2 \right)$$

and the network definition is:

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_l^o h_{il} \right), \qquad h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij} \right),$$

with transfer function $s(a) = \frac{1}{1+e^{-a}}$.

(a) Verify that the derivatives needed for gradient descent are:

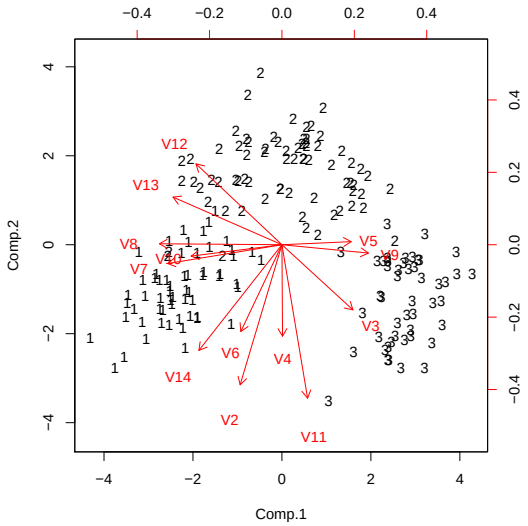
$$\frac{\partial J}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{il},$$
$$\frac{\partial J}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) w_l^o h_{il} (1 - h_{il}) x_{ij}.$$

(b) Suppose instead that you have a neural network for binary classification with L hidden layers, each hidden layer having m neurons with logistic transfer function. Give the parameterization for each layer, and derive the backpropagation algorithm to compute the derivatives of the objective with respect to the parameters. For simplicity, you can ignore bias terms.

3. **AdaBoost (ESL).** Solve exercises 10.1 (optimal β) and 10.2 (connection between exponential loss and log-odds) of the "Elements of Statistical Learning" book, which is freely available at `https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf`.

4. ***Coding*: Biplots, Trees, Random Forests.** Download the wine dataset from `https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data` and load it using `read.table("wine.data",sep=",")`. Description of the dataset is given at `https://archive.ics.uci.edu/ml/datasets/Wine`.

(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 train a classification tree using `rpart`, and relate the decision rule discovered there to the projections of the original variable axes displayed in the biplot. Give the plots of the tree as well as of the cross-validation results in `rpart` object using `plotcp`.

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

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

Use `tuneRF` to find an optimal value of `mtry`, the number of attribute candidates at each split. Use `varImpPlot` to determine what are the most important variables.

5. ***Coding*: Neural networks.** In this question you will investigate fitting neural networks using the `nnet` library in R. We will train a neural network to classify handwritten digits 0-9. Download files `usps_trainx.data`, `usps_trainy.data`, `usps_testx.data`, `usps_testy.data` from
`http://www.stats.ox.ac.uk/~palamara/teaching/SML19/`.

First, you'll need to make sure that the nnet library is installed. Use `install.packages("nnet")` and select a package repository (CRAN mirror) nearby.

You can load the data files using `read.table(...)`. If you need more help getting started, consult slide 59 of the notes available at `http://www.stats.ox.ac.uk/~evans/statprog/prog_slid_ho.pdf`.

Each handwritten digit is 16×16 in size, so that data vectors are $p = 256$ dimensional and each entry (pixel) takes integer values 0-255. There are 2000 digits (200 digits of each class) in each of the training set and test set.

You can view the digits with:

```
plot_digit = function(x,y) {
  # input x should be a numeric vector of length 256
  image(matrix(as.matrix(x), 16,16), col=grey(seq(0,1,length=256)),
```

```
        main=sprintf("Label: %d",y))
}
plot_digit(trainx[500,],trainy[500,])
```

Download the R script `nnetusps.R` from `http://www.stats.ox.ac.uk/~palamara/teaching/SML19/nnetusps.R`. The script trains a 1-hidden layer neural network with $S = 10$ hidden units for $T = 10$ iterations, reports the training and test errors, runs it for another 10 iterations, and reports the new training and test errors. To make computations quicker, the script down-samples the training set to 200 cases, by using only one out of every 10 training cases. You will find the documentation for the `nnet` library useful: `http://cran.r-project.org/web/packages/nnet/nnet.pdf`.

(a) Edit the script to report the training and test error after every iteration of training the network. Use networks of size $S = 10$ and up to $T = 100$ iterations. Plot the training and test errors as functions of the number of iterations. Discuss the results and the figure.

(b) Edit the script to vary the size of the network, reporting the training and test errors for network sizes $S = 1, 2, 3, 4, 5, 10, 20, 40$. Use $T = 25$ iterations. Plot these as a function of the network size. Discuss the results and the figure.

(c) **Optional.** Can you get a significant performance increase by adding multiple layers? To investigate, you'll need to switch to the package `neuralnet`. Here is code to run `neuralnet` with one hidden layer, check the documentation (in R type `?neuralnet`) to see how to add more layers.

```
library(neuralnet)
data = cbind(class.ind(trainy),trainx)
names(data)[1:10] = paste0("label",names(data[1:10]))
fml <- as.formula(paste(paste(paste0('label',0:9),collapse=" + "),
  ' ~ ' ,paste(paste0("V",1:256),collapse='+')))

net <- neuralnet(fml, data, err.fct="ce", linear.output=FALSE)
```

6. ***Coding (optional)*: Tensor Flow.** This problem is going to give you the chance to try out TensorFlow, one of a number of popular deep learning packages. TensorFlow must be installed in python, but once installed there is an R library. Instructions:

   - Install TensorFlow using the instructions here: `https://www.tensorflow.org/get_started/os_setup`
   - Verify that you can use TensorFlow in python:

     ```
     $ python
     >>> import tensorflow
     >>>
     ```

   - If you would like to use R, follow the instructions here: `https://github.com/rstudio/tensorflow`

   Returning to the digit classification task from the previous problem, try once again to improve performance using a multi-layer neural network. Hint: use a convolutional layer (`conv2d` in TensorFlow).