MTAT.03.227 Machine Learning
Spring 2016 / Exercise session XV
Nominal score: 10p
Maximum score: 15p
Deadline: 24th of May 16:15 EET

Exercises 2,3,5,6,8,9,10 are by Raivo Kolde with minor modifications by Meelis Kull, exercises 1,4,7,11,12,13 are by Meelis Kull.

Preliminaries

The goal of this homework is to get acquainted with some ensemble methods like bagging, boosting and random forests. For testing we use the dataset about handwritten digits. In order to make the implementation easier we have stripped it down to only 2 digits: 2 and 3. The datasets are in file Ensemble.RData and can be read into R using command

```
print(load("Ensemble.RData"))
```

There are 2 datasets test and train, both with the same structure, first column indicates the class and all other columns are the features. Single digits from the dataset can be visualised with the help of this function

```
library(ggplot2)
draw_digit = function(data,index) {
  d = expand.grid(1:28,28:1)
  d$pixel = as.numeric(data[index,2:ncol(data)])
  p = ggplot(d)
  p = p + geom_tile(aes(x=Var1,y=Var2,fill=pixel))
  print(p)
}
```

using the following command:

```
draw_digit(train,13)
draw_digit(train,42)
```

In all the models we use classification trees that in R are implemented in package rpart. For building one tree on training data, predicting values on test data and calculating misclassification rate, the code looks roughly like that:

```
library(rpart)
train$y = factor(train$y)
m = rpart(y ~ ., data = train, method = "class")
p = predict(m, newdata = test, type = "class")
t = table(test$y, p)
print(t)
print(1 - sum(diag(t))/sum(t))
```

Bagging (4p)

Bagging is a method where an ensemble of classifiers is trained on bootstrap samples of training data. To generate a bootstrap sample of a dataset with n rows one has to sample with replacement n rows from training data. To predict a response. a prediction is made with all members of the ensemble and the result with most "votes" is reported.

1. Train a bagging classifier on training data using 50 bootstrap samples. Measure its misclassification rate. Use the function bagging from the R package ipred. (**1p**)

2. Solve the same task as in Exercise 1 but implement bagging yourself, training the base classifier with rpart. (**2p**)

3. Draw a figure that shows change in misclassification rate of bagging classifier if we add samples. (x-axis: number of datasets 1-50; y-axis: misclassification rate) (**1p**)

Random Forest (4p)

Bagging estimate can be improved, if we decrease the correlation between the trees in ensemble. One way to do it, is to use random subsets of data when constructing trees.

4. Train a random forest classifier using the R package randomForest, using 50 trees and sampling 50 features as candidates for each split decision. Measure its misclassification rate. (**1p**)

5. Implement a simple random forest type classifier, where each tree is constructed using only a subset of m = 50 variables. Note that this is different from the original random forest because the original method samples the subset of variables at each decision node again. Use 50 bootstrap samples. Measure its misclassification rate and compare to the random forest from the previous exercise. (**2p**)

6. Try values 10, 50 and 300 for m and draw similar figure as in Excercise 3. What can be said about the results? (**1p**)

Boosting (4p)

Boosting works a bit differently that bagging. Instead of taking samples of the data, the training dataset is re-weighted every step, to concentrate efforts on the points that were misclassified.

7. Train Adaboost classifier on our example doing 50 steps using the function ada from the R package ada. Measure its misclassification rate. (**1p**)

8. Implement Adaboost yourself using rpart as the base model learner. The instance weights can be specified using the 'weights' parameter to rpart. Iterate 50 steps and measure its misclassification rate. Compare to the result of the previous exercise. (**2p**)

9. The default tree parameters might provide too rich model for boosting and therefore induce overfitting. To avoid this, limit the tree depth to 2 (can be set as rpart.control(maxdepth=2) in rpart and ada) and train again the classifier for 50 steps. You can modify the code from either of the two previous exercises. Measure its misclassification rate. (**1p**)

Interpretation (3p)

10. Interpret the results from all previous tasks.

    (a) What methods perform the best? (**1p**)

    (b) How does the result change with the number of models in the ensemble? (**1p**)

    (c) How does changing the model complexity affect the ensemble performance in bagging and in boosting? (**1p**)

Exercises on a toy dataset (5p)

Answer the questions below for the dataset D consisting of the following two-dimensional data points and two-class labels: $x_1 = (1, 2)$, $x_2 = (0, 0)$, $x_3 = (-1, -1)$, $y_1 = +1$, $y_2 = -1$, $y_3 = +1$.

11. A decision tree with a single decision node is known as the *decision stump*. Let us define a learning algorithm $\mathcal{A}$ as a decision stump learner which selects its decision node as the one with minimal error. If several such exist then it chooses the one with the most positive predictions. Apply the AdaBoosting algorithm on the above data D with $T = 3$ iterations and the decision stump learner $\mathcal{A}$. Determine all the obtained instance weights w_{ti} and model weights α_t. Determine the final output score $M(x)$ of each of the instances. Are all instances classified correctly when thresholding $M(x)$ at 0? If not, how many iterations T does it take to reach correct classifications? (**1.5p**)

12. Consider the bagging algorithm applied on the above data D, ensemble size $T = 3$, and the decision stump learner $\mathcal{A}$ as defined in the previous exercise.

    (a) What is the probability that the majority vote of the obtained ensemble gives correct predictions to all instances? (**1p**)

(b) What is the probability if $T = 1$? (**0.5p**)

(c) $T = 5$? (**0.5p**)

13. The AdaBoost algorithm uses several "magical" constants: $1/N$ in defining $D_1(i)$; $1/2$ in defining α_t; $1/Z_t$ in defining $D_{t+1}(i)$.

(a) What would happen with the results of the algorithm if we changed $1/N$ into the constant 1? Would the output of the new algorithm be the same as the output of the old algorithm for all possible inputs and choices of base learner? (**0.5p**)

(b) What would happen with the results of the algorithm if we changed $1/2$ into the constant 1? (**0.5p**)

(c) What would happen with the results of the algorithm if we changed $1/Z_t$ into the constant 1? (**0.5p**)