PBSB 5022.03 2016 Assignment 1

Course website: https://piazza.com/mskcc.org/spring2016/pbsb502203/home
R-Project: https://www.r-project.org/

Machine Learning for Visual Perception

The first part of this assignment is designed to get familiar with random forests. The second part is a real world visual perception problem from computational pathology.
The example code is shown in R but you can use your preferred programming language and random forest package exist for all modern languages (e.g. scikit-learn in Python, Sherwood in C# and C++, etc.).

1. Train and test a random forest model on the Iris dataset

1.) Load the Iris data set provided on Plaza and investigate its format. Every sample is described by 4 different features and assigned a Species as class label.

```
iris
head(iris)
```

```
       Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
  1            5.1         3.5          1.4         0.2     setosa
  2            4.9         3.0          1.4         0.2     setosa
  3            4.7         3.2          1.3         0.2     setosa
  …
```

2.) Investigate the class label distribution:

```
summary(iris)
table(iris$Species)
```

3.) Install and load a random forest package/library.

```
install.packages("randomForest")
library(randomForest)
```

4.) Note 2: To see all possible parameters type
"?<function name>" in R, e.g. `?randomForest`

5.) Define the design matrix and the classification target

```
X <- iris[,1:4]
Y <- iris$Species
```

6.) Learn a random forest model

```
rf <- randomForest(x=X, y=Y, ntree=200, do.trace=T)
```

7.) Print results

```
rf
```

8.) Plot results

```
plot(rf)
```

9.) Investigate the Variable Importance

```
importance(rf)
barplot(t(importance(rf)), col=4)
varImpPlot(rf)
```

10.) Investigate how the model parameters influence the classification performance

Note 1: This is not a binary but a multi-class classification task, since there are three distinct species.
Note 2: To see all possible parameters type "?<function name>" in R, e.g. `?randomForest`
Note 3: To train two equal randomForests for a dataset X, use a defined random number seed with `set.seed(<number>)`

The input of your algorithm

- Iris data

The output of your algorithm

- Out Of Bag Error convergence plot
- Variable Importance plot

Optional Bonus Task

- Train a different classifier, e.g. a SVM (package `e1071`) or a Naïve Bayes classifier and compare its performance to random forests.

2. Learn a classification model to differentiate malignant from benign cell nuclei.

Overview: Single cell classification is an important task in daily clinical practice. E.g. in pathology or oncology, variations in cancer cell types and density in the tissue can make differences for diagnosis, prognosis and treatment. While traditional pathology fulfills this task on a daily basis with bright field microscopes and the naked eye with the years of experiences of a pathologist, research is ongoing to automate this task with computer vision methods and machine learning algorithms. Thus, reproducibility, throughput and objectivity is hoped to be improved. Since the various tissues differ tremendously in shape, morphology and general appearance, these classifiers in general tend to be tuned to individual tissues. In this example, you will train a classifier to differentiate cancer nuclei from benign nuclei in Renal Clear Cell Carcinoma (RCC) tissue micro array images. As training labels, a pathologist has pointed every cell nucleus in two such images (Fig. 1, 2).

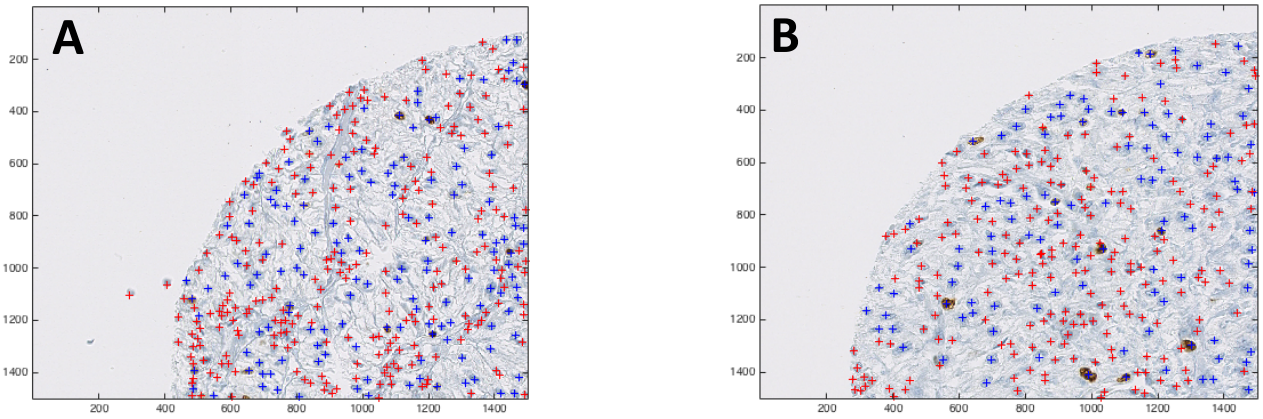

Figure 1: **A:** Labeled TMA spot 1 from a pathologist. Red are cancer cell nuclei, blue are benign cell nuclei. **B:** Labeled TMA spot 2 from a pathologist. **D:** Patches of labeled surface features from 9P/Tempel. **E:** Locally normalized patches, constituting the positive class for training.

Figure 2: **LEFT:** Three examples of cancerous nuclei. **RIGHT:** Three examples of benign nuclei.

Exercise Goal: Train a random forest classifier to differentiate between cancerous and benign nucleus features (labeled by a pathology expert). Learn how to extract features and how to test their impact on classification.

1. Download and unzip the nucleus feature dataset from Plaza. We provide csv files with the raw images and labels for use in R and in other languages.

2. Load the data of spot 1 into R

```
spot1 <- read.csv("<yourPath>/spot1.csv", header=TRUE, sep=";")

# Look at one nucleus, first 5 columns are metadata
image(matrix(as.numeric(spot1[1, 6:ncol(spot1)]), nrow=60))
```

3. Extract the target variable Y from spot 1

```
Y <- spot1[, "Labeler1"]
Y <- as.factor(Y)
table(Y)
```

4. Create a design matrix **X** by extract features from the grayscale patches. Start by using a vector of raw grayscale values. For example:

```
X <- c()
for (i in 1:nrow(spot1)) {
   X <- rbind(X, spot1[i, 6:ncol(spot1)])
}
```

5. Train a random forest "rf1" to differentiate malignant from benign nucleus features

6. Investigate the feature importance to decide which features make sense and which do not.

7. Report the OOB error of your model to your TA to compare with other students.

8. Add new features to the design matrix, e.g. a histogram of the image patch, and repeat from step 5.

Bonus Tasks:

1. Plot an ROC curve based on the confidence of the classifier to see its performance.

2. Compare the ROC curves of different models in the same plot.

3. Can the nuclei of spot 2 be classified with the classifier trained on spot 1?
   For this,
   - load the data of spot 2 into R (see step 2)
   - Extract the target Variable Y2 from spot 2 (see step 3)
   - Create a similar design matrix X2 as you did for spot 1 (see step 4)
   - Classify the nuclei of spot 2 with their design matrix and the classifier of step 5.
   ```
   Y2_hat <- predict(rf1, newdata=X2)
   ```
   - Compare the predictions with the real label
   ```
   table(Y2, Y2_hat)
   ```

Hints:

- ROC curves can be plotted with the "pROC", "Epi" or the "ROCR" package.
- Image patches and matrices can be visualized with the "image()" function.
  ```
  image(spot1[1, 7: ncol(spot1)])
  ```

The input of your algorithm

- Image features from the nucleus patch.

The output of your algorithm

- A random forest model
- OOB generalization error estimates
- ROC curve

General guidelines for problem sets involving software

- Send all the software required via PIAZZA to the TA. It is most helpful if you package code, data, and answers in a .zip or .tgz file. https://piazza.com/mskcc.org/spring2016/pbsb502203/home
- We expect your solutions to be in the spirit of "reproducible research" (http://reproducibleresearch.net/). Include instructions/scripts that allow reproducing your experiments with relatively little effort. For example, include a script "main.r" or "main.m" that calls the other files.
- Your software is evaluated on clarity and elegance as well as correctness. Comment your code.
- Insert a comment for everything that is not immediately obvious; and do not comment what is obvious. Think whether you would still understand the code you write after locking it up in a drawer for a year.
- You will be given code examples in R, but you are free to use any language with which you are comfortable (Matlab, C#, C++, Python, JS, etc.).
- You are encouraged to use available libraries for reading/writing files and analogous tasks. However, you cannot use functions which the homework implies you have to write yourself.
- You are not required to follow the "implementation guidelines" perfectly. If you know a better alternative for a certain step, it is ok to use it.
- You are responsible for the parameters you choose. If we give you a "reasonable" value for a parameter that does not appear to work, you should try other values.
- You cannot share code for homework or look at other people's code. You are free to discuss general ideas about the problem (reading aloud your code does not count as discussion).

Notes on R

If you use R, consider using a good IDE. RStudio is open source and an excellent option.
R is available for any OS from the local UCLA mirror: http://cran.stat.ucla.edu/
RStudio is available at http://www.rstudio.com/products/RStudio/
Additional R links will be posted on Plaza.
Don't hesitate to ask if you have questions.