SGN-41007 Pattern Recognition and Machine Learning

- ▷ The group assignment is linked to the Kaggle competition.
- ▷ All items of this document are mandatory except item 7.
- ▷ Each step should be documented in a written report that has to be returned no later than Friday, March 10th at 23:59. **Send PDF to `pattern@tut.fi`**.
- ▷ We recommend `http://www.overleaf.com/` for preparing the report.
- ▷ If you have technical questions regarding the assignment, we will organize sessions where you may ask questions collectively.

1. *Preparations: log in to Kaggle, load data and Python template.*

   The competition and related data is available at

   `https://inclass.kaggle.com/c/gene-expression-prediction`

   Log into the system either using your existing Google/Facebook/Yahoo account or create new account for Kaggle.com. This allows you to load the data and submit solutions.

   Download the data and the Python template for opening the data. A Python template is available at

   `https://github.com/mahehu/SGN-41007/tree/master/assignment`

2. *Add a classifier to the template script.*

   The template does not process the loaded data in any way. Continue the script by the following functionalities

   - Train a *Logistic regression* classifier with the training data.
   - Predict class probabilities for the test samples.
   - Write Kaggle submission file in the following format[1]. Each line contains the gene ID and the probability of the gene being active.

     ```
     GeneId,Prediction
     1,0.200000
     2,0.400000
     ...
     3871,0.400000
     ```

     Submit this to Kaggle to check that the file format is good.

     Add to the report the required code lines and the accuracy score of Kaggle.

[1] `https://docs.python.org/2/tutorial/inputoutput.html#reading-and-writing-files`

3. *Experiment with parameter* `C` *and penalty of the LogReg classifier.*

   As seen in the class, the parameter `C` has a large effect on the accuracy. Add a loop in your code that estimates the AUC score for different values of `C`; for example, $C = 10^{-6}, \ldots, 10^{0}$. Easiest approach is to use `sklearn` function `cross_val_score`.

   Do the above CV experiment for L1 and L2 penalty.

   Next, train a model with all data using the best C and penalty combination. Submit to Kaggle and check the score.

   Add to the report the required code lines and a table of accuracies for selected values of C for the two penalties (for example, like Table 1) and the Kaggle score for best combination.

4. *Add preprocessing and cross-validate.*

   The dimensionality of the data is relatively high: $5 \times 100 = 500$ features. Try to reduce the number of dimensions using Principal Component Analysis. This is most conveniently done by creating a pipeline of the preprocessing and classification steps. Then, each step will be executed in a sequence with only a single fit() operation. This is particularly convenient when cross-validating a sequence of operations.

   Prepare a pipeline of PCA and Logistic regression[2]. Experiment with different PCA dimensionalities (PCA transforms the original 500 dimensional data into a smaller number of dimensions as specified by user). Iterate over a list of dimensions (*e.g.,* $[10, 20, 30, \ldots, 100]$) like in task 3 above, or use `sklearn.model_selection.GridSearchCV` as described in the example[2].

   Add to the report the required code lines and a table of accuracies for selected values of PCA dimensions and the Kaggle score for best choice.

[2] `http://scikit-learn.org/stable/auto_examples/plot_digits_pipe.html`

Table 1: Cross-validated AUC scores of Logistic Regression classifier for different hyperparameter combinations.

C	L1 penalty	L2 penalty
10^{-6}	80%	80%
10^{-5}	85%	90%
$\vdots$	$\vdots$	$\vdots$
10^{0}	84%	80%

5. *Try different classifiers.*

   Iterate the above procedure for **at least two additional classifiers** of your choice. Their hyperparameters may be different, but adjust the search procedure to vary those parameters you feel are important.

   Add to the report the required code lines and a table of accuracies for selected classifiers (and parameters) and the Kaggle score for best choice.

6. *Communicate your progress to the customer.*

   Attend the weekly meetings with the customer. In each meeting, tell the customer what you have done since previous meeting and what you plan before the next meeting. Describe which tasks of this document are done and what is your leaderboard position in Kaggle. Ask any questions regarding biology that may help you.

   Add to the report the required code lines and a table of accuracies for selected classifiers (and parameters) and the Kaggle score for best choice.

7. *Try deep learning (optional)*

   If you have access to an NVidia GPU, you may try to train a convolutional network (CNN) for the original 5×100 blocks. The benefit of a CNN this is that it preserves the original shape of each sample instead of vectorizing to a 500-dimensional vector. You may try to emulate the structure that the authors of the original paper [1] used[3]. Or you may construct a structure of your own.

   For those who wish to experiment on TUT GPU cluster, contact the teacher for getting an account and a quick introduction to its use.

   Add to the report the required code lines and and cross-validated accuracy and the Kaggle score.

References

[1] R. Singh, J. Lanchantin, G. Robins, and Y. Qi, "Deepchrome: deep-learning for predicting gene expression from histone modifications," *Bioinformatics*, vol. 32, no. 17, pp. i639–i648, 2016.

[3] The code is available at `https://github.com/QData/DeepChrome`; although in LUA language used by the Torch platform