Introduction to Statistical Machine Learning
The University of Adelaide
Assignment 1

Due date: 11:59pm, 2 **Sept.,** 2020

1 Instructions and submission guidelines:

- This is not a group assignment. Everyone is asked to complete the assignment individually.
- The assignment consists of a report and matlab (or Python) implementation of binary-class linear Support Vector Machines (SVMs).
- Explain the key concepts of linear SVM in the report.
- Make sure that your writing is legible and clear, and the mathematical symbols are consistent.
- Make sure that your Matlab or Python code is well commented and can be executed directly. Please include a `README.txt` file to show how to run your code.
- You should sign an assessment declaration coversheet to submit with your assignment. The assessment declaration coversheet is included in the zip file.
- Submit your report and all your code via myuni on the course web page.

2 Reading

We have briefly covered soft margin binary SVMs in Lecture 3. Please read the SVM tutorial [1] and the guide [3] for more details to complete the assignment.

3 Report

Please write down your understanding of binary class linear SVMs (within 3 pages) and experiment comparison of your code and an existing implementation of SVMs such as libsvm [2] or scikit-learn (within 3 pages) in the report. So in total, you have at most 8 pages for the report. This is no strict format of the report (rather than the page limit). The purpose of the report is to show what you have understood about SVMs and what you have done to make your code correct. The report should at least cover the following key points (not limited to) :

- The primal form and its dual form for both hard margin and soft margin case;
- Concept of support vectors;
- Concept of max margin;
- Concepts of generalisation/test error;
- Experimental results including comparison between your implementation and a 3rd-party SVM implementation.

4 Coding

- Please implement soft margin binary class linear SVMs by:
  - Solving the primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \|\mathbf{w}\|_2^2 + C \cdot \frac{1}{n} \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, ..., n \end{aligned} \tag{1}$$

  - Solving the dual problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned} \tag{2}$$

  Your code should strictly implement the functions that are in the form of the prototypes below. Otherwise you may not get marks even if your experiment results are correct.

  Other necessary functions such as data IO, plotting, analysis of results etc. and anything else can be in any form.

```

# train your svm in the primal
svm_model  = svm_train_primal( data_train, label_train, regularisation_para_C)

# test on your test data
test_accuracy = svm_predict_primal(data_test, label_test, svm_model)

# ...

# train your svm in the dual
svm_model_d  = svm_train_dual( data_train, label_train, regularisation_para_C)

# test on your test data
test_accuracy_d = svm_predict_dual(data_test, label_test, svm_model_d)
```

  NB: 'regularisation_para_C' is the hyper-parameter C in the above two equations and typically need to be determinted by cross-validation if you want to achieve good accuracy on test data.

  Examples that do **not** follow the coding requirement:

  a) Input and output arguments are not the same as above prototypes, such as:

  ... ... `svm_train_primal( data_train , label_train , regularisation_para_C, kernel_para )`

  b) Implement SVM as a CLASS. Do not implement SVM as a CLASS!

- You are encouraged to use the matlab optimisation tool

  http://cvxr.com/cvx/

  or Python tool https://cvxopt.org/

  to solve the above two optimisation problems (the primal and dual)

- You are asked to call third-party SVM implementations to compare the results against your implementation. Recommeded third-party SVM implementations:

  SVM in scikit:

  https://scikit-learn.org/stable/modules/svm.html

  SVM in matlab:

  https://au.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html

  LibSVM: see [2].

- The data for training and testing are included. Please check the README file inside the zip file. You need to run your SVM using this provided dataset.

5 Data

Both Training and Test data are provided already. However validation dataset is not provided. So if you want to have a validation dataset for cross validation of the hyper-parameter C, you will need to randomly split the provided Training data into Training and Validation.

NB: Please note that in the SVM formulations we have studied, the labels are either -1 or $+1$. For real-world datasets if the labels are not -1, $+1$, such as 0, 1 or 1, 2, you can just convert the labels into -1, $+1$.

6 Marking criteria

- Define variables before you use them [5 points]
- Primal and dual forms of hard and soft margin SVMs [5 points]
- Concept of support vectors [5 points]
- Concept of max margin [5 points]
- Experiments

  We are intentionally leaving this part open-ended to allow for experimentation. Besides the following, you are encouraged to run more experiments and show your thoughts. You might get extra points.

  - compare your w, b obtained via solving the primal problem, with the w, b reconstructed by the dual variables obtained via solving the dual problem (both the results and the reconstruction formulation) [10 points]
  - compare your results (both training and testing errors) with those of a 3rd-party SVM implementation [10 points]
  - code [60 points]

Please note that all responses/answers to all above checkpoints should be included in the report, not in the code.

References

[1] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[2] Chih-chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines," 2001. software available at https://www.csie.ntu.edu.tw/~cjlin/libsvm/. 2012.

[3] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.