

4 Coding

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

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

- Solving the dual problem:

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

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.

```

1
2
3 # train your svm in the primal
4 svm_model = svm_train_primal( data_train, label_train, regularisation_para_C)
5
6 # test on your test data
7 test_accuracy = svm_predict_primal(data_test, label_test, svm_model)
8
9 # ...
10
11 # train your svm in the dual
12 svm_model_d = svm_train_dual( data_train, label_train, regularisation_para_C)
13
14 # test on your test data
15 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 determined 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. Recommended 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.