Homework 4: SVMs

How to view this in nice PDF

`pandoc -s hw4.md -o hw4.pdf`

Unless otherwise stated, 1. all SVMs are hard margin SVM using a linear dot-product kernel.

Hand Computation (1pt each)

1. An SVM is trained using the follow samples:

sample ID	feature a	feature b	feature c	label
1	0.5	0.25	0.125	+1
2	0.4	0.15	0.225	+1
3	0.3	0.75	0.325	-1
4	0.2	0.65	0.425	-1

resuling in the λ's sequentially as $\lambda_1 = 4.5$, $\lambda_2 = 0$, $\lambda_3 = 1.5$, $\lambda_4 = 0$, what is the prediction from the SVM for a new sample $[1, 1, 0]$? Let w_b be 1. Be sure to include steps of estimating $\mathbf{w}$ in your answer. If you have only the final answer, you won't get any point.

2. What are the equations of the two gutters per the $\mathbf{w}$ obtained above and $w_b = 1$?

3. With the $\mathbf{w}$ obtained above, and the assumption that w_b is 1, identify samples that fall into the margin and those do not. A sample falls into the margin if it is between the two gutters. Thus, turn the two equations of the gutters into two inequalities and check them against every sample. Show your steps. If you have only the final answer, you won't get any point.

4. For an SVM, if a (misclassified) sample $\mathbf{x}_i$ is on the outter side (not the margin side) of the gutter for the opposing class, which one of the following conditions holds? And why? You could use proof-by-contradition to eliminate false choices. (If you do not answer the why part, you get no point.)

   1. $y_i(\overbrace{\mathbf{w}^T\mathbf{x} + w_b}^{\text{prediction per SVM}}) \geq -1$
   2. $y_i(\mathbf{w}^T\mathbf{x} + w_b) \leq -1$
   3. $y_i(\mathbf{w}^T\mathbf{x} + w_b) \geq 1$
   4. $y_i(\mathbf{w}^T\mathbf{x} + w_b) \leq 1$
   5. $y_i(\mathbf{w}^T\mathbf{x} + w_b) \geq 0$
   6. $y_i(\mathbf{w}^T\mathbf{x} + w_b) \leq 0$

Programming

Code template: `hw4.py`

For all functions below, every arguments is a 1-D list of floats or integers while every returns is a float or integer.

5. [1pt] **Finding the best C for a soft-margin SVM on fixed a pair of training and test sets** Now we want to study the relationship between C and the performance of an SVM on a real dataset, the Wisconsin Breast Cancer dataset. The data are features manually extracted (e.g., thru rating) by pathologists from biopsy images under a microscope. There are two classes/targets/labels: malignant and benign. The features are quantified descriptions of the images.

The function `study_C_fix_split` takes an argument `C_range` as the sole input which is a list `C` values. For each value in `C_range`, train an SVM and evaluate it using Scikit-learn's SVM classifier functions (functions `fit` and `score` for `sklearn.svm.SVC` class). Finally, return the `C` that yields the highest score.

The code in `study_C_fix_split` already loads the data and split it into the fixed training and test sets. Train an SVM using `X_train` and `y_train` as input and output, and then use `X_test` and `y_test` to get a performance score. The split is at 80% training and 20% test. The data is pre-scambled with fixed `random_state` at 0.

Use default settings for all `sklearn.svm.SVC` functions except the `C` which you should scan, `kernel='linear'`, and `random_state=1` – **it's important to fix the random state to get consistent results**.

6. [1.5pt] **Finding the best C for a soft-margin SVM using cross validation** In the problem above, we used a fixed pair of training and test set. To more holistically study, redo it using cross validation here. Finish the function `study_C_cross_validation` using `sklearn.model_selection.cross_val_score`. Your function should have the same input and output as the function `study_C_fixed_split` above. Use default settings for all parameters unspecified here, e.g., for CV, do default 5-fold CV.

The first argument of `sklearn.model_selection.cross_val_score` is an estimator. In our case, it should be an SVM created using `sklearn.svm.SVC`. You do NOT need to manually `fit` nor `score` as the function `sklearn.model_selection.cross_val_score` does them for you. But note that the function `sklearn.model_selection.cross_val_score` returns a list of floats, which are the scores of all folds.

7. [1.5pt] **Finding the best `C` thru automated grid search** Finally, let's take advantage of Scikit-learn's grid search CV for hyperparameters. Finish the function `study_C_GridCV`. It has the same input and output as the two functions above. Use default settings unless specified here. Note that the

input `C` is a list but in `sklearn.model_selection.GridSearchCV` it needs to be converted into a dictionary like `{'C':[1,2,3,4,]}` whose values must be a 1-D list instead of 1-D numpy array.

8. [2pt] **Finding the best hypermaters for Gaussian kernels thru automated grid search** Expand what you just did above for problem 7 for SVMs with Gaussian kernels. Instead of searching over `C`, you will search over every combination of `C` and σ. The function `study_C_and_sigma_gridCV` takes two inputs, one is a range for `C` and the other is a range for σ. Like in Problem 7, make use of Scikit-learn's grid search CV, grid search CV. This time, your hyperparamter dictionary needs to have two entries, like `{'C':[1,2,3,4,], 'gamma':[5,6,7,8]}` where The `gamma` is how σ is called in Scikit-learn's SVM function.

Bonus

9. [2pt] In the mathematica demo, each SVM is solved into multiple solutions. But many of the solutions are invalidate for their λ's are all zeros. Please modify the code to add some constraints to eliminate invalidate solutions. {Hint]: One class of equations and inequalities are neglected when discussing KKT conditions on our slides. But your can find them under "Necessary Conditions" of the KKT conditions Wikipedia page.

How to submit

For hand computation part, upload one PDF file. For programming part, upload your edited `hw4.py`.

How to view this in nice PDF

```
pandoc -s hw4.md -o hw4.pdf
```