Homework 4: SVMs

Important issues

1. all SVMs are hard-margin using a linear dot-product kernel.
2. do NOT truncate a number on your own.

How to view this in nice PDF

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

A precompiled PDF is here Weblinks are in pink.

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

   Suppose (they may not satisfy KKT conditions) the λ's are sequentially: $\lambda_1 = 4.5$, $\lambda_2 = 0$, $\lambda_3 = 1.5$, $\lambda_4 = 0$, 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, i.e.,

   $$-1 < \mathbf{w}^T\mathbf{x} + w_b < 1$$

   where $\mathbf{x}$ is the (unaugmented) feature vector of the sample.

Show your steps, especially the value of the prediction $\mathbf{w}^T\mathbf{x} + w_b$. If you have only the final answer, you won't get any point.

Please check over all four samples, as the λ's above are toy examples and do not satify KKT conditions.

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, what three conditions below hold? 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 argument is a 1-D list of floats or integers while every return 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. More information about this dataset can be found in Section 7.2.7 of this Scikit-learn document.

   The anatomy of the function

   Finish the function `study_C_fix_split` which scans `C` over a range provided as the input `C_range`, tracks the best performance score of the SVM so far, and finally returns the `C` that yields the highest performance score. Use Scikit-learn's SVM functions (See below).

   Training and test sets The code template 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 inputs/feature vectors and labels/targets, 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 1.

   How to train and test an SVM in Scikit-learn? To train an SVM, use the function `sklearn.svm.SVC.fit`. To get the performance score of a trained SVM on a test set, use the function `sklearn.svm.SVC.score`.

Configurations 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**. By default, Sklearn will use non-weighted accuracy as the score.

6. [1.5pt] **This problem is now a free problem. Everyone gets points for free**

   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**

   Redo Problem 5 in Scikit-learn's grid search CV . An example is provided on the linked webpage. Finish the function `study_C_GridCV`. It has the same input and output as the function in Problem 5.

   How to provide `C_range` to `sklearn.model_selection.GridSearchCV`?
   The input `C` is a list or Numpy array, but in `sklearn.model_selection.GridSearchCV` it needs to be converted into a dictionary that maps a hyperparameter name to a sequence of values `{'C':[1,2,3,4,]}`.

   How to make use of the function `sklearn.model_selection.GridSearchCV`?
   In our case, the function `GridSearchCV` needs two arguments. The first is an instance of `sklearn.svm.SVC` with proper configurations (see below). The second the parameter dictionary mentioned above.

   How to fetch the result from `sklearn.model_selection.GridSearchCV`?
   Its return has a member called `best_params_` which is a dictionary. `best_params_['C']` is what you need in the end.

   Configureations For your SVM instances, be sure that `C` is NOT set, `kernel` is set to `\'linear\'`, and `random_state` is set to `1`. Use default values for all other settings and parameters.

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.

   Finish the function `study_C_and_sigma_gridCV` that 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 `gamma` is how σ is called in Scikit-learn's SVM classifier.

   For your SVM instances, be sure that the `kernel` is set to `rbf` and `random_state` is set to `1`. Do NOT set `C` nor `gamma` when initializing the SVC instance. Use default values for all other settings and parameters.

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.

   Mathematica can be download from this link.

How to submit

For hand computation part, upload one PDF file. For programming part, upload your edited `hw4.py`. Do NOT delete contents in the template, especially those about importing modules.