Homework Assignment 5

COGS 118A: Introduction to Machine Learning I

Due: 11:59pm, Wednesday, February 21st, 2018 (Pacific Time).

Instructions: Answer the questions below, attach your code, and insert figures to create a PDF file; submit your file via TritonEd (ted.ucsd.edu). You may look up the information on the Internet, but you must write the final homework solutions by yourself.

Late Policy: 5% of the total points will be deducted on the first day past due. Every 10% of the total points will be deducted for every extra day past due.

System Setup: You can install Anaconda to setup the Jupyter Notebook environment. Most packages have been already installed in Anaconda. If some package is not installed, you can use `pip` to install the missing package, that is, just type `pip install PACKAGE_NAME` in the terminal.

Grade: ____ out of 100 points

1 (15 points) Shattering

Use shattering to derive the VC-dimension for classifiers below. Show your work.

1) $f(x; w, b) = \text{sign}(x \times w + b)$

2) $f(x; q, b) = \text{sign}(q \times x \times x + b)$

3) $f(x; w, b) = \text{sign}((x \times w + b)^2)$

where $x, w, q, b \in \mathbb{R}$, and w, q and b are free parameters.

2 (85 points) Support Vector Machine

In this problem, you are required to solve a series of questions using support vector machine (SVM). You will use Arrhythmia dataset that contains 452 data points. Each data point has a 279-dimensional feature vector and an 1-dimensional label (either 0 or 1), which means it is a binary classification task and can be solved by SVM. Please download the `arrhythmia.npy` as data source and `hw5-q1-svm.ipynb` to fill the blanks. You can use the functions from `sklearn` in your implementation unless in some case we ask you to implement a few built-in functions by yourself.

2.1 (35 points) Linear SVM

In this sub-problem, you need to use the linear SVM to conduct the binary classification.

1) Load data from `arrhythmia.npy` and shuffle the data points.

2) Select 80% of the data points as your **training and validation set**. The rest 20% is regarded as your **test set**. Actually, in the cross-validation, the training and validation set can be called as "training set". However, in order to be consistent with the code, we still call it "training and validation set" here.

3) Train the SVM classifier using a linear kernel. In linear SVM, there is a parameter C which adjusts the cost of outliers. You would need to use a grid search method to find the best parameter C^*. In fact, such grid search will utilize the cross-validation (3-fold) to get all the **average training accuracies** and **average validation accuracies** from the linear SVM model with different parameter C on training and validation set. The parameter $C = C^*$ which maximizes the **average validation accuracy** will be selected as the best. In fact, here "average" means the average accuracy over the folds in cross-validation, not the average accuracy over the different parameter C.

   Hint 1: You are allowed to use `svm.SVC()` and `GridSearchCV()` in your code.

   Hint 2: You can perform grid search on the following list of C:

$$C \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$$

4) Draw heatmaps for the result of grid search and find the best C^* for average validation accuracy. Report the heatmaps and best C^*.

5) Use the the best C^* to train a linear SVM classifier on training and validation set. Then, use the trained classifier to calculate the accuracy on test set. Report the test accuracy.

2.2 (20 points) SVM with the RBF Kernel

In this sub-problem, you need to use the SVM with the radial basis function (RBF) kernel to conduct the binary classification.

1) Train the SVM classifier using a RBF kernel. In SVM with the RBF kernel, there is a parameter C and a parameter γ which can be tuned. Again you would need to use a grid search method with cross-validation (3-fold) to find the best combination of parameter C^* and γ^* for current SVM model on training and validation set.

Hint 1: You are allowed to use `svm.SVC()` and `GridSearchCV()` in your code.

Hint 2: You can perform grid search on the following list of C and γ:

$$C \in \{0.1, 1, 10, 100\}, \quad \gamma \in \{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$$

2) Draw heatmaps for the result of grid search and find the best C^* and γ^* for average validation accuracy. Report the heatmaps and best C^* and γ^*.

3) Use the the best C^* and γ^* to train a SVM classifier with a RBF kernel on training and validation set. Then, use the trained classifier to calculate the accuracy on test set. Report the test accuracy.

2.3 (30 points) Implement Grid Search and Cross-validation

In this sub-problem, you need to implement the grid search and cross-validation functions by yourself. You are **NOT** allowed to use `GridSearchCV()` here.

1) Implement a cross-validation function. In this function, you should divide your training and validation set into several subsets which have roughly the same size (the number of subsets is given by variable `fold`). Train the SVM with RBF kernel for `fold` rounds and each round choose one different subset as validation set and all the other data points (all the other `fold - 1` subsets) as training set. Calculate the **training accuracy** and **validation accuracy** every round. Finally, return the **average training accuracy** and **average validation accuracy** over all rounds. For more details you can refer to page 19 in `COGS118A_2018_Lecture10.pdf`, where the notations are slightly different.

2) Implement a grid search function. In this function you need to traverse all combinations of C and γ. For each combination of C and γ, you should call your implemented cross-validation function above to get the average training accuracy and average validation accuracy. Finally, you need to return **average training accuracy matrix** and **average validation accuracy matrix** for all combinations of C and γ.

3) Like what you have done in SVM with the RBF kernel, perform your implemented grid search with cross-validation (3-fold) to find the best combination of parameter C^* and γ^*. Draw heatmaps for result of grid search and get the best C^* and γ^*. Report the heatmaps and the best C^* and γ^*.

Hint: You can compare your heatmaps with the heatmaps from `GridSearchCV()` in the above sub-problem to confirm the correctness of your implementation. Both heatmaps should share similar behavior.

2.4 (Bonus 5 points) Implement Linear SVM

In this sub-problem, you need to implement the linear SVM using gradient descent by yourself. Same dataset in HW4 Q2 logistic regression is still used here: $\{(\mathbf{x}^{(i)}, y^{(i)})\}$, $y^{(i)} \in \{0, 1\}$ and $\mathbf{x}^{(i)} = [x_0^{(i)}, x_1^{(i)}, \ldots, x_K^{(i)}]^\top$ where $x_0 = 1$ is added as a bias. Your implementation of SVM should minimize the loss function:

$$\mathcal{L}(\theta) = ||\theta||^2 + \lambda \sum_i \max(0, 1 - y^{(i)} f(\mathbf{x}^{(i)}; \theta))$$

where $f(\mathbf{x}^{(i)}; \theta)) = \sum_{k=0}^{K} \theta_k x_k^{(i)}$ and $\lambda = 1$. You are **NOT** allowed to use `svm.SVC()` here. Train the linear SVM model and report the code with following results:

1) The optimal θ^*.

2) Training accuracy and test accuracy.

3) Plot of training data along with decision boundary.

4) Plot of test data along with decision boundary.