Assignment 5: K-Nearest Neighbor Classifier and Support Vector Machines

UVA CS 4774
Machine Learning Foundation, Deep Learning and Good Uses

January 20, 2022

a *The assignment should be submitted in the PDF format through Collob. If you prefer hand-writing QA parts of answers, please convert them (e.g., by scanning or using an app like Genuis Scan) into PDF form.*

b *For questions and clarifications, please post on Slack.*

c *Policy on collaboration:*

Homework should be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the midnight on the due date. Each student has 10 *extension days to be used at his or her own discretion throughout the entire course. You could use the days in whatever combination you like. For example, all late days on 1 assignment or 1 each day over 10 assignments (for a maximum grade of 90% on each). After you've used all 10 days, you cannot get credit for anything turned in late.*

e *: we will use the EXTRA points in YOUR final grade calculation...*

e *Policy on grading:*
1: *50 points in total. 30 points for code submission (and able to run). 20 points for displaying results in the report.*
2: *50 points in total. 25 points for code submission (and able to run). 25 points for the results in your report.*
3: *10 extra points if you win the competition! i.e. if you get Top 20 accuracy in the leaderboard)*
The overall grade will be divided by 10 and inserted into the grade book. Therefore, you can earn 11 out of 10.

1 KNN and Model Selection (k) (programming)

In this problem, we will implement k-Nearest Neighbor classifier and select k using cross validation. We will use the same movie review dataset we used in HW4. Please use the "knn_template.ipynb" template to guide your work. Please follow the instructions and the function names/descriptions in the template. Feel free to cross-check your implementation against sci-kit's KNN. Other requirements or recommendations are the same as HW1 and HW2. This problem covers two main tasks:

- (1): To implement a very simple classifier, k-nearest neighbors (kNN), from scratch.
- (2): To implement a **4-fold** Cross Validation strategy for selecting the best k for kNN classification. Feel free to reuse the cross-validation code in the previous HWs.

Detailed Instructions:

1. You are required to fill-in the function

$$findBestK(x, y, klist, nfolds)$$

following the guidance in the template. You are given a list of $k \in \{3, 5, 7, 9, 11, 13\}$. The goal is to find the best k using $nfolds = 4$ fold cross validation.

2. The first step is implementing the cross validation method that splits data into folds:

$$x\_train, y\_train, x\_test, y\_test = fold(x, y, i, nfolds)$$

.i indicates the fold number (iterated from 0 to $nfolds - 1$). This function returns the ith train and test fold from x and y. Feel free to try other folds for cross validation, kfold=3-fold or kfold=10-fold.

3. After getting a train and test fold, the next step is implementing kNN classification method using the function:

$$y\_predict = classify(x\_train, y\_train, x\_test, k)$$

Where k is the number of data points that kNN takes into consideration when labeling an unlabeled data point. Note here the training data is part of the testing (classify) algorithm. In kNN, we label a new data point based on the k points that are the closest to it in our train dataset. In this function, for each testing point, find its k nearest neighbors in the training set (those having the smallest Euclidean distance to the testing point). As reviewed in class, kNN requires a distance metric to compute distance between samples. We will use **Euclidean distance** as the measurement of distance in kNN. Then we label the testing point according to the majority voting rule. The predictions $y\_predict$ is a list of 0 or 1 predictions obtained from the classify method.

4. Once predictions are obtained, the next step is to evaluate the prediction. For this, implement the calc_accuracy method:

$$acc = calc\_accuracy(y\_predict, y)$$

Where $y\_predict$ is the list of 0 or 1 predictions obtained from the classify method and y is the true label for the testing points (the last column of the test data). We repeat classify and evaluation methods for all folds, to get cross validation accuracy for each k.

5. **Input Features:** We will use three types of input features for this HW.

   - BOW representation: These are the features we used for naiveBayes in HW4.You can reuse the data processing code for this HW. For this HW, you will also compare to a $MAX\_VOCAB = 768$ to make the number of input features comparable to BERT features.
   - TF-IDF representation: Instead of the vanilla Bag of Words representation, we will use the Term Frequency-Inverse Document Frequency representation. For this you can use the

     $$sklearn.feature\_extraction.text.TfidfTransformer$$

     that converts a count matrix into TF-IDF representation.
   - BERT Embeddings: Similar to the previous HW, we will use embeddings, or sentence level representations extracted from a pretrained model and directly use as input for the kNN classifier. Note that because we do not need to train this model, we will extract the sentence level embeddings from the $TFBertModel$. Most helper methods have been provided for this part of the HW.

6. **Task 1:** Report the cross validation accuracy for all values of k as well as the best k, for all three types of input features. Discuss why some values work better than others. Also report the test accuracy for each type of input features using the best k.

7. **Task 2:** Implement function $barplot(klist, accuracy\_list)$. A bar graph is recommended to show how accuracy varies with k. Use k as x-axis and accuracy as y-axis. Put the bar plot in your report. You can use matplotlib for this question. Att: we will not consider the computational speed in grading your kNN code.

2 Support Vector Machines with Scikit-Learn and preprocessing

This question covers two main tasks:

- (1) For this assignment, you will implement an SVM classifier. Given a proper set of attributes, your program will be able to determine whether an individual makes more than 50,000 USD/year.
- (2) You will use scikit-learn's C-Support Vector Classifier: sklearn.svm.SVC. Install the latest stable version of scikit-learn following directions available at `http://scikit-learn.org/stable/install.html`. You can use builtin functions for feature processing, like sklearn OneHotEncoder and pandas feature processing functions.

Dataset Details: Two dataset files are provided, labeled sample set "salary.labeled.csv" and unlabeled sample set "salary.2Predict.csv". Make sure to download them from Collab! The "salary.labeled.csv" is the total training data available to you. The unlabeled sample set "salary.2Predict.csv" is a text file in the same format as the labeled dataset "salary.labeled.csv", **except that its last column includes a fake field for class labels**. The details of the column names and feature properties are provided in the svm_template.py. You are required to generate/ predict labels for samples in "salary.2Predict.csv" (including about $10k$ samples).

Note: We will evaluate your output 'predictions' - an array of strings (">50K" or "<=50K") according to the true labels of these test samples in "salary.2Predict.csv" (ATT: you don't have these labels !!!). This simulates a Kaggle-competition in which test labels are always held out and only team-ranking will be released after all teams have submitted their predictions. When grading this assignment, we will rank all students' predictions. So please try to submit the best performing model that you can! **Best groups in the tournament will receive bonus credit.**

Detailed Steps:

1. The first step is loading and preprocessing the data:
   - The dataset contains both categorical and continuous features(See sample data in Table 1. First, implement load_data function in svm_template.py. This function takes as input filename *training_csv* and processes into numerical vector representation. For example, to start, you can try normalization on continuous features and one-hot encoding on categorical data.
   - For preprocessing, use techniques mentioned in class!
   - Be creative. A good data preprocessing can help you win the tournament.
2. The next step is to choose a good hyperparameter for SVM classifier via cross validation. You are required to write a wrapper code for cross-validation as part of $train\_and\_select\_model(training\_csv)$, and iterate through different hyperparameters (just like we did in HW1/2).
   - The different hyperparameters you can choose include SVM kernels (basic linear kernel / polynomial kernel / RBF kernel), C value, ... A few examples are provided in the svm_template.py.
   - Call sklearn.svm.SVC with your parameter to get a classification!
   - You need to include the CV classification accuracy results in your pdf report by performing 3-fold cross validation (CV) on the labeled set "salary.labeled.csv" (including about $38k$ samples) using at least three different SVM kernels you pick. Please provide details about the kernels you have tried and their performance (e.g. 3CV classification accuracy) including results on both CV train accuracy and CV test accuracy into the writing. For instance, you can summarize the results into a table with each row containing kernel choice, kernel parameter, CV train accuracy and CV test accuracy.
3. Next, also as part of $train\_and\_select\_model(training\_csv)$, select the best hyperparameter, train a model using the selected hyperparameter and also return the corresponding cross validation score as well as the trained SVC model.
4. Finally, use the trained model to make predictions on the test set using $predict(\text{``}salary.2Predict.csv\text{''}, trained\_model)$ and save the output. The svm_template.py provides $output\_results(predictions)$ to save the predictions in the desired format. **Remember not to shuffle your prediction outputs!**

Submission Instructions: You are required to submit the following :

1. A python program svm.py (filled in svm_template.py):

   It should be able to train and select a model using a set of hyperparameters on the training data, these hyperparameters should be hard coded.

2. We should be able to use the learned/ trained_model to classify samples in an unlabeled test set using the following function:

   ```
   predictions = predict(‘‘salary.2Predict.csv”,trained_model)
   ```

3. A file “predictions.txt” generated by:

   ```
   output_results(predictions)
   ```

   Please do not archive the file or change the file name for the automated grading.

4. In your PDF submission,

   - A table reporting classification accuracy (score) averaged over the test folds, along with details of the best performing kernel, C etc.
   - You need to include the CV classification accuracy results in your pdf report. Please provide details about the hyperparameters you have tried and their performance (e.g. 3CV classification accuracy) including results on both train and test folds into the writing. For instance, you can summarize the results into a table with each row containing kernel choice, kernel parameter, CV train accuracy and CV test accuracy.
   - Please also discuss in the written submission about how you preprocessed the data, and how you chose your SVM hyperparameters.

Classes: >50K, <=50K.
Attributes:
age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Table 1: About the data in SVM coding.