CS 441

HW 1: Instance-based Methods

Due Date: Feb 5, 2024 11:59pm

In this assignment we will explore instance-based methods on two problems: handwritten digit recognition and temperature prediction. The aims of this homework are:

1. Gain experience with machine learning methods based on retrieval, particularly K-means clustering, KNN classification, and KNN regression
2. Get experience working with FAISS, which provides much faster search and is very useful in practice.
3. Learn how feature representations can impact performance
4. Get practice selecting parameters using a validation set (stretch goals)

Read this document, the report template, and tips and tricks before beginning to code.

- Report template
- Tips and Tricks
- Starter code and data

We are finalizing the submission procedure. The problems will not substantially change, but some details of reporting/submission may change, e.g. whether you enter results directly into Gradescope or submit a PDF of the report.

MNIST Digit Dataset

MNIST is a dataset of handwritten digits created in 1998. The task is to learn to classify an image into a digit from “0” to “9”. The samples are collected from employees of the US Census Bureau and high school students. Each sample is a 28x28 image with values from 0 to 255. We can reshape and rescale it to a 784 dimensional vector with values from 0 to 1. There are 60,000 training images, and 10,000 test images. In this homework, we will treat 10,000 images of the training images as validation for initial testing and parameter selection. **For parts a and b, use the validation set to measure error, not the test set.**

Efficient querying of high-dimensional data is important for many applications, including retrieval, clustering, and K nearest-neighbor classification. In the first two parts, we will use the Faiss (Facebook AI Similarity Search) library to explore kmeans and KNN on the MNIST data.

1. Retrieval, Clustering, and Nearest Neighbor Classification [30 points]

To start, we're going to write our own code to solve these problems, without worrying about speed. Some code structure has been provided in the starter code. Edit it to complete the homework.

Retrieval: The core task is retrieval. Implement the function `get_nearest` using Euclidean (L2) distance. Check that `get_nearest(x_test[0], x_train)` returns `i=53843`. *Report the index of the closest example in `x_train` to `x_test[1]`.*

K-means: Now, using your get_nearest function, write a function kmeans that iteratively assigns each data point to the nearest cluster center. This will take a long time if you apply it to the full training set, so apply it to only the first 1000 examples, `x_train[:1000]`. Try this with K = 10 and K = 30, and display the cluster centers after each iteration. *Include the displays from after the 1st and 10th iteration for K=30 in your report.* Do any of the clusters seem to change identity?

1-NN: Now, use your get_nearest function to perform 1-nearest neighbor. For each test sample, find the index of the closest sample in the training data to predict its label. Since your retrieval code is slow, we're not going to use the full dataset yet. To check your method, calculate the error for the first 100 test samples using only the first 1,000 training samples; the error should be 17%. *Report the percent error for the **first 100 test samples** using the **first 10,000 training samples**.*

2. Make it fast [35 points]

We'll now use the FAISS library to speed up retrieval, K-means, and 1-NN.

Retrieval: Exact search can be performed using the code below.

```
index = faiss.IndexFlatL2(X.shape[1])  # set for exact search

index.add(x_train) # add the data

dist, idx = index.search(x_test[:2],1) # returns index and sq err for each sample
```

Check that `idx` matches your retrieved indices from Part 1.

K-means: Complete fast_kmeans using FAISS for the retrieval instead of your get_nearest function. In each iteration, create a new index, add the cluster centers, and find the nearest center to all samples. You may want to disable any print or display functions inside the kmeans loop.

Record the root mean squared error (RMSE) at the start of each iteration, and *plot the RMSE for each iteration for K=10, K=30, and K=100 when clustering the full training set with 20 iterations.*

1-NN: Use FAISS to evaluate 1-NN on the full training and test sets. Try this with both the exact search and LSH approximate search. The only difference is how you set up the index. For LSH, use:

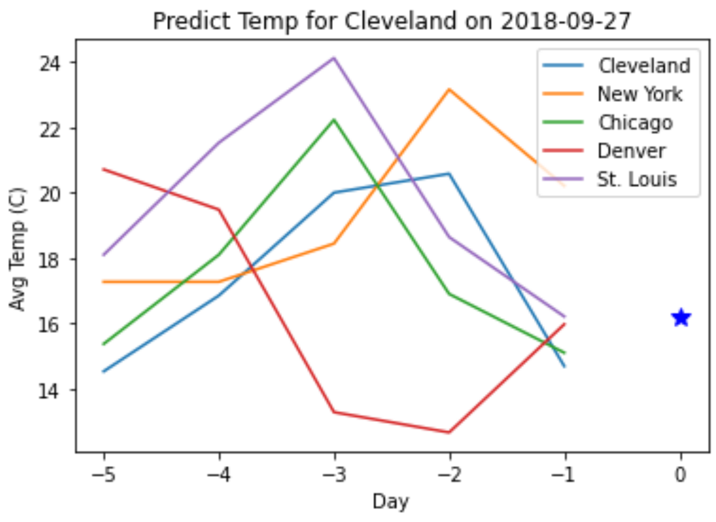
```
dim = X.shape[1]

index = faiss.IndexLSH(dim, dim)
```

Evaluate 1-NN using each search method, while varying the number of training samples: `s in [100,1000, 10000, 60000]`. In each case, use `x_train[:s]` as the train set. *Plot number of samples vs. 3percent error on a semilogx plot for both exact and LSH (on the same plot). Also, record timings using `time.time()` and plot samples vs. time on a semilogx plot.*

It is often useful to look at which labels are confused with which other labels, e.g. by looking at a confusion matrix, which you can generate with `sklearn.metrics.confusion_matrix`. In your report, *indicate which label is most often confused with '2'* when using the full training set and exact search for 1-NN.

3. **Temperature Regression** [20 points]

The Temperature Regression problem is derived from a [Kaggle dataset](). The task is to predict the average temperature in Cleveland for the next day given the average temperature of major US cities in the preceding five days.



Implement and test k-NN for regression

Perform 5-NN regression, *reporting RMSE for two variants*:

1. Original features
2. Normalize the features by subtracting the previous day's Cleveland temperature. I.e., if previous day's Cleveland temperature is c, features are X, and value to predict is y, then predict `y_query-c = NN(X_query, X-c, y-c)`

For these experiments, train on (x_train, y_train) and test on (x_test, y_test). To validate your method, if you set K=3, you should get an RMSE of 3.403 for the original features.

Do you get better results with the normalized features? Think about why.

4. **Test your understanding** [15 points]

Answer the questions in this section of your report.

5. **Stretch Goals**

A big part of machine learning development is selecting parameters through experimental comparisons on a validation set. Each of these comparisons is worth 15 points.

a. Compare K-NN on the MNIST classification for N=1, 3, 5, 11, 25. For these tests, use `x_train[:50000]` as a training set and `x_train[50000:]` as a validation set. Report error on the validation set for all parameters. Then performance on the test set for the best parameter using the full training set.

   When K is greater than 1, return the most common label of the nearest samples. If there is a tie, return the most common label with the closest sample.

b. Compare K-NN on the temperature regression dataset for N=1, 3, 5, 11, 25 using both feature types. Report all results on the validation set, and then run the single best setting on the test set.

c. For K-means (on MNIST), does running with multiple re-runs or running a single run longer tend to provide lower RMSE? For this, you can use the FAISS `kmeans` function. Syntax is below.

   ```
   kmeans = faiss.Kmeans(x_train.shape[1], 30, niter=10, nredo=1, seed=int(t))
   sum_sq_err = kmeans.train(x_train)
   ```

   Compare (niter=10, nredo=5) vs. (niter=50, nredo=1) for K=30. Repeat this test five times and report the mean and standard deviation of the RMSE.

   Compare (niter=4, nredo=5) vs. (niter=20, nredo=1) for K=30. Repeat this test five times and report the mean and standard deviation of the RMSE.

Submission Instructions

We are finalizing the submission procedure. Some details may change, e.g. whether you enter results directly into Gradescope or submit a PDF of the report.

Submit three files: (1) completed report template; (2) an html or pdf version of your Jupyter notebook; and (3) a zip of your notebook and any other python files needed to run your code. Be sure to include your name, number of points claimed, and acknowledgments. **The report is your primary deliverable**. The notebook and code are included for verification/clarification by the grader.