

Fall 2019: STAT 445 / CSCI 555 Statistical Learning and Data Mining

Assignment #3 To be completed in either Matlab or Python

This assignment is focused on multiclass classification. We will use a more realistic example of character recognition than that used in the previous assignments, considering all digits 0-9 instead of just a selection of two of them. Download (from moodle), `NumberRecognitionBigger.mat`, which was provided in assignment 2. Note the dataset includes data samples for all handwritten digits 0 to 9. You can implement your assignment in either Matlab or python, with details to follow. Note that python is by far the most common environment for coding ANNs and CNNs. Traditionally I have indicated that Matlab is easier on the students, however, python may in fact be easier for questions 3 and 4 in this assignment.

Coding

Example Matlab and Python functions that can be relied upon are already outlined in Assignments 1 & 2. Assignment 3 may also benefit from the following commands. You are expected to read documentation on the commands available and try to get them working, prior to asking for assistance. Please address questions to the course Teaching Assistant, Derek Berger (dberger@stfx.ca, PS 1094).

Matlab

Creating an artificial neural network (ANN) in Matlab requires the definition of a basic net architecture. This can be done with the following command:

```
net = patternnet(5);
```

which will establish an ANN with 5 hidden layer neurons. You can then train your ANN with this command:

```
[net]=train(net, trainingData, trainingLabels);
```

And test it with this command

```
[predictions]=net(testingData);
```

Creating a convolutional neural network (CNN) in Matlab requires the definition of a layer structure. This defines the configuration/architecture of the CNN.

For example:

```
Layers = [ ...  
imageInputLayer([28 28 1]) %defines lowest layer as taking in a 28x28 image  
convolution2dLayer(3,5) %convolutional filter bank with 5 3x3 learned filters  
reluLayer %a rectified linear unit layer  
fullyConnectedLayer(10) %parameter: # of classes to output  
softmaxLayer %softmax layer  
classificationLayer %final stage: bit odd for Matlab to put this after softmax  
];
```

You can also set many learning options, like the learning rate, number of epochs etc.:

```
options = trainingOptions('sgdm','InitialLearnRate',0.001,'MaxEpochs',15);
```

Then you can train your model:

```
net = trainNetwork(trainingImages,trainingLabels,layers,options);
```

Note that trainingImages will be $m \times m \times 1 \times n$, where n is the total number of training images and m is the input x and y dimensions of the image itself. You can test it with:

```
predictions = predict(net,testingImages);
```

Python

ANN

Basic ANNs in Python are most easily implemented in `sklearn.neural_network` as the `MLPClassifier` class. The calls are nearly identical to MATLAB, but different default activation functions and solvers are used. To create an ANN with similar defaults to the MATLAB ANN above, we can use

```
ann = MLPClassifier(hidden_layer_sizes=5, activation="tanh", solver="lbfgs")
```

The ANN is trained with:

```
ann.fit(X_train, y_train)
```

and test predictions are generated with:

```
ann.predict(X_test)
```

CNN

Writing a CNN in Python is done most easily using the Keras interface to Tensorflow.

```
from keras import losses, optimizers
from keras.layers import Conv2D, Flatten, BatchNormalization # we have 2D
images
from keras.layers import Dense, ReLU
from keras.models import Sequential
```

A Model can be built sequentially via `Sequential()`.

```
model = Sequential()
```

In MATLAB, the `imageInputLayer` implicitly handles normalization. In Python, this is explicit.

```
model.add(BatchNormalization())
model.add(
    Conv2D(5, kernel_size=3, input_shape=(28, 28, 1),
          activation="linear", # only to match MATLAB defaults
          data_format="channels_last"
    )
)
```

Add a relu layer:

```
model.add(ReLU())
```

What MATLAB calls a `fullyConnectedLayer`, Tensorflow calls a `Dense` layer. You sometimes need to flatten the output of a convolutional layer before inputting it to a `Dense` layer

```
model.add(Flatten())
model.add(
```

```
Dense(units=10, activation="softmax") # 10 units, 10 digits
) # multiclass classification output, use softmax
```

Options can also be set in Python. Some are handled when building the network, some when you finally compile the network for fitting.

```
model.compile(
    optimizer=optimizers.SGD(momentum=0.9, lr=0.001),
    loss=losses.mean_squared_error,
    metrics=["accuracy"],
)
```

The model is trained and predictions are made in the usual manner.

```
history = model.fit(X_train, y_train, epochs=15, verbose=1)
y_pred = model.predict(X_test)
```

Questions

Question 1: Reuse K-Fold cross validation (K=5) from assignment #2, but use all digits 0-9 in the dataset. Within the validation, you will train and compare a CNN with the basic layer architecture defined above, as well as a K-NN classifier with (K=1, K=5 and K=10). The validation will train these models for predicting 0s through 9s. NOTE: for a fair comparison, K-Fold randomization should only be performed once, with any selected samples for training applied to the creation of all classifier types (CNN, KNN) in an identical manner (i.e. the exact same set of training data will be used to construct each model being compared to ensure a fair comparison).

Provide a K Fold validated error rate for each of the classifiers. Provide a copy of your code (Matlab or python). Answer the following questions:

- Which classifier performs the best in this task?
- Why do you think the underperforming classifiers are doing more poorly?

It was previously announced on multiple occasions that each student is required to assemble their own dataset compatible with supervised learning based classification (i.e. a collection of measurements across many samples/instances/subjects that include a group of interest distinct from the rest of the samples). If you are happy with your choice from assignment 1 or 2, then re-provide your answer to Assignment 1 or 2 Question 2 below. If you want to change your dataset for this assignment, for a future assignment or for your graduate project, you are free to do so, but you have to update your answer to Question 2 based on your new dataset choice.

Question 2: (Repeat) Describe the dataset you have collected: total number of samples, total number of measurements, brief description of the measurements included, nature of the group of interest and what differentiates it from the other samples, sample counts for your group of interest and sample count for the group not of interest. Write a program that analyzes each measurement/feature individually. For each measurement, compute Cohen's d statistic (the difference between the average value of the group of interest and the average value of the group not of interest, divided by the standard deviation of the joint distribution that includes both groups). Provide the leading 10 leading measurements (d statistic furthest from zero), with their corresponding d statistic, making it clear what those measurements represent in your dataset (these are

the measurements with the most obvious potential to inform prediction in any given machine learning algorithm). Provide your code.

Question 3: Adapt your code from Question 1 to apply the ANN alongside the KNN ($K=1$, $K=5$, $K=10$) on your personal dataset. Note you do not need to implement a CNN here as they are applied to images and many personal datasets are not image based. Experiment with different training options for the ANN (such as changing the number of hidden layers), and report on error rate differences between those options. Provide the error rates for the different ANN classifiers and the KNN classifier and your code. Answer the following questions: What ANN configuration that you tried worked best in your application? Why do you think it worked better than the other options you evaluated?

Question 4: There are infinite possible implementations of a CNN. You have a lot of potential flexibility, especially in your choice of how to structure the architecture and the many trainingOptions you can set (see above example and online documentation). In this final assignment question, you are challenged to create the best CNN architecture (lowest error rate) that you are capable of. Report the final K Fold error rate your system produces, and submit the code of your best architecture comparing against the models built in question 1, all compared in a fair manner (same K-Fold training samples). Also, download the NumberRecognitionTesting.mat file, which contains a single set of example images without labels. Predict those testing samples' classes with a trained model using your best architecture. You will have to save these predictions and submit them with your assignment. Derek will provide detailed submission instructions. You are REQUIRED to follow them to the letter, especially with respect to the submission of these testing sample predictions.

Deadline: November 28th, 2019.