PY-599 (Fall 2018): Applied Artificial Intelligence

Dimension Reduction of Training Data for Faster Training

Deadline to Submit: Thursday 3:00 PM, Oct 18th (right before the Thursday session)
Submission Method: Please share your Colab notebook

Training a model can be a very time-consuming task. Imagine your training data is composed of millions of data points, and each data point itself can be a large image, a voice signal, or a video. Storing the training data, moving it from the memory to the processor, and processing it can require a lot of time and a lot of electricity power ($) – and depending on the situation we may not afford either of them! But imagine we can afford to lose a few points of accuracy in exchange for a faster/cheaper training phase.

There are certain methods to trade a little bit of accuracy for speed. A good example would be reducing the size of your data. For example, rather than having a 4K video as a data point, we can reduce the video to a 480p-resolution video. Granted we are going to lose some information by compressing the data, but the resulting data would be smaller, therefore easier to store, transfer, and process. And the neural network to handle the data would be smaller, and as a result we will need less computation for training the model. And therefore, the process of training would be faster and cheaper.

In the class we discussed what dimension reduction means, and we worked with multiple different dimension reduction algorithms such as PCA.

In this homework we like to combine PCA with a classification problem. In the class we talked about how we can design a feedforward neural network to classify single-digit images (MNIST dataset). In this homework we want to use PCA (or any other dimension reduction method that you like) to reduce and project the images to a lower dimensional space. And then we train the neural network on this reduced dataset.

Basically we just need to integrate "Session 8: Principal Component Analysis" code (or the homework that you did on dimension reduction) into "Session 3: MNIST Classification, Feedforward NN" code. You don't have to use these codes, you can write your own code – or just copy and paste these two components and combine them together.

Examine different values for the number of principal components in PCA (or any other reduction method that you choose) and see how much accuracy you lose for those values, and find the fewest number of principal component (which results in maximum amount of reduction) with reasonable amount of reduction in the performance of the classifier. To evaluate the performance of the model always apply it to the test data.

Heads-up tip: check the difference between `fit_transform` instruction and `fit` and `transform` as two separate instructions when you are using scikit-learn's PCA implementation. Which one should you use? Please note that you have to apply the trained model to the test data or future unseen data.