CS 532 (Sec. 4) – Project Proposal

Alex Guo

Due Date: October 22, 2020

1 Project Dataset

The dataset I'm using contains images of handwritten, chinese characters (which represent numbers). The link is: https://www.kaggle.com/gpreda/chinese-mnist/notebooks. There are 15 classes for the data, since there are 15 different numbers – 0–10, 100, 10^3, 10^4, 10^8 – that are each represented with a single chinese character. The dataset contains 15,000, 64×64 images of a single character, and there are 1,000 images for each type of character.

2 Algorithms that will be investigated

I am not planning on using an unsupervised learning method like PCA for dimensionality reduction to pre-process the data (I don't think it will be needed). But I will probably use some python library for the feature extraction. The three supervised learning algorithms I will use are:

1. **k-NN** (k-nearest neighbors): The k-NN method is non-parametric, so I don't need to worry about messing with a loss function. Since the method is based on distance, I will normalize the training data to be on similar scales.

2. **SVM** (support vector machines): SVM (or at least the kernel-based SVM) uses the kernel trick to implicitly map the inputs to a higher dimensional feature space. The big decision I need to make is if I should use a linear kernel or a non-linear kernel to transform the data. I guess I will first try a linear kernel (since it's computationally faster), and then see if I need a non-linear kernel for better accuracy (since they allow for curved hyperplanes).

3. **NN** (neural networks): The main two things that I will need to experiment with are the number of layers and the activation functions.

I will shuffle the data and then take 90% of it for training and leave 10% for testing (i.e. 10% is hold-out data). Within the training data, I will use k-fold validation to get the average misclassification error (I will probably use $k = 10$ or something). I will use this average error to further tweak each algorithm. I will then test my algorithms on the testing data and hope some of them do well. I might even write some chinese characters myself and test the algorithms on those as well. To compare the three algorithms, I will just see which one is more accurate when applied to the testing data.

3 Project Github

Here's the link: https://github.com/ag262/CS532.git

4 Project timeline

10/22 - 11/4: write code to train the first algorithm

11/5 - 11/16: investigate other two algorithms and start coding them

11/17 (milestone): first update due

11/18 - 11/30: finish coding all three algorithms and start doing k-fold validation and testing them on the hold-out data

12/1 (milestone): second update due

12/2 - 12/11: do last-minute corrections/testing and write up the final report

12/12 (milestone): final report due