Project Updates: Image Classification with CIFAR-10 Dataset

Tim Lieb tlieb@wisc.edu

1-December-2020

Progress

I realized that I sort of jumped right into the implementation with only minimal understanding of the dataset, so I did a good amount of background research to learn more. The main takeaways had to do with preprocessing the data, especially for neural networks. The images are not very useful in row vectors, so preprocessing can be done to transform them into three 32 x 32 RGB images. In addition, normalizing the data can be helpful as well, which I implemented for Ridge Regression.

There's a lot of information about setting up neural networks for the CIFAR-10 dataset, but not for other models. This led to the realization that my other methods probably won't work well. This realization shifted my focus from trying to get a perfect working model to instead pushing the performance boundaries and inferring what I could about the data and the model.

I worked more on the Ridge Regression model. I first tested my algorithm using a smaller separate dataset to ensure that it was working properly. I then tested it on a subset of the CIFAR-10 dataset—1000 training images with 1000 test/validation images. This took ~24 to minutes to run with 50 logarithmically spaced lambdas, but the error was very high. This could be due to the limited training data as well as the model limitations—L2 norm also might not be the ideal regularizer either if the solution is sparse, which could make sense since it's likely that not every pixel is important. I also tried running the tests with 10,000 training and 5,000 test images. After 3 hours it's only ~20% of the way done.

I also did some more research into using K Nearest Neighbors and I realized that I had a fundamental misunderstanding. I was thinking that it was still a clustering method that I would use to divide the data into k=10 categories, but that is only how K Means works. In KNN, the data is classified by using distance vectors for the k nearest points. This could potentially be an advantage because k<10 could work fine and run faster, however the speed greatly decreases with larger volumes of data. I then began implementation of the KNN algorithm.

I did a little bit of research into Neural Networks/ some of the background research overlapped with this. More work still needs to be done in this area, including the implementation.

Action Items

- Wrap up Ridge Regression
    - Figure out a way to test for the best lambda value—iterative approach
    - Run the full dataset with only that lambda
    - If this is not feasible, I will likely have to run on a smaller dataset. In this case I would likely separate the data by labels and only use a few in the classifier

    - Could also try other ways to speed it up—making it greyscale or using CVX convex optimization package
- Continue with KNN implementation
    - Perform tests on the algorithm—start small and work up to larger datasets
    - Look into possible speed improvements
    - When looking at PyTorch for NN, found that it could maybe be used for KNN as well—look into this more
- Implement Neural Network Model
    - Use PyTorch, but write my own layers
    - Lots of variations/ parameters to decide on, but can use other research as reference points

17-November-2020

Progress

Since the initial proposal, I have started the implementation of my code. First, I came up with a way to read in the data. I spent some time figuring out the formatting and organization of the data and determined how I would be able to utilize it. The data is already split into 5 training sets and 1 test set, but for cross validation, I will use all 6 sets as training and testing for different iterations. This will allow me to better compare all the different algorithms.

Then I began working on the Ridge Regression algorithm by creating a function to perform the calculations and analysis. I use 5 of the data sets as the training data, and the I split the one hold out set into two sets for the testing. The function first takes a set of lambdas (still need to figure out what set/ range to use) and finds the w for each. Then it chooses the lambda that gives the lowest error rate using the first test set. Then with the best lambda and corresponding w, the last test set is used to find the error rate and the squared error. While my method seems right, the code takes a long time to run, and so I will look into ways to speed it up.

In my project proposal, I proposed using K-means to classify the data. This was a mistake because K-means is used for unsupervised learning which is not what I want. I was thinking of a K-nearest Neighbors approach, but I'm not certain this will work. I will need look into the issue more and determine if it'll fit my needs. If not, I will need to choose another algorithm.

Action Items

- Figure out how to determine the set of lambdas used for Ridge Regression
    - Arbitrary?
    - "Use a logarithmic spaces lambdas"
- Look into approaches to help speed up the Ridge Regression calculations
    - ATA vs AAT → 10,000 x 3072
    - SVD can be expensive
    - Could do a gradient descent type approach?

    - Break into smaller sections?
- Check about using K-nearest neighbors—come up with an alternative algorithm
    - We actually didn't really talk about it in class
    - Other algorithms seem very similar—maybe ok?
        - Same loss function, different regularizer
        - Many are just binary classification—hinge loss, support vectors
- Start implementing the second algorithm once finalized
- Work on Neural Networks aspect once we've overed it in class