CS 830 Intro to AI, Spring 2025

Written Assignments, Sushma Akoju

Assignment 10

Electronically submit your solution using the instructions on the course web page, including your source code as well as a transcript of your program running with the tester and brief answers to the following questions:

1. Describe any implementation choices you made that you felt were important. If you implement anything beyond the assignment as written, please be sure to discuss it. Clearly explain any aspects of your program that aren't working. Mention anything else that we should know when evaluating your work.

I chose Numpy library thinking it would optimize.
But it is taking longer than a minute given that we cannot use other known forms of parallelization/tensors or neural networks.

I also attempted to use PCA and NMF for dimensionality reduction. I included both basic KNN and PCAKNN. But KNN works on a smaller dataset. For this reason, my make.sh contains virtualenv and installing scikit-learn. But KNN is the original one and works on the Colab.
K > 16 works better for this example. K = 20+ or even 34 could work. We could have used gridsearch to find an appropriate k.

I found out that there are other data structures such as BallTree and they could be used to measure the distances for KNN. But I did not include it since it requires some additional libraries such as scikit-learn.

For Online Linear LMS : I used lms update step and it works for digits on my colab. But label prediction accuracy as well as F1 score is not good. F1 score is preferred.

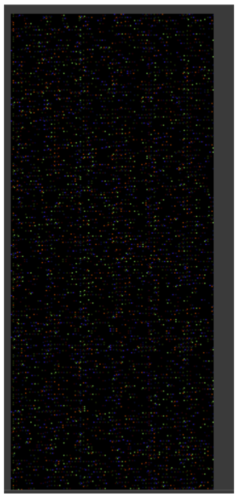
For Naive Bayes (NB), I implemented 4 versions: Naive bayes basic version works fine.

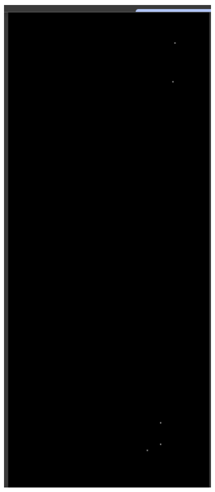
NBC I tried Non Matrix Factorization for dimensionality reduction to reduce the components to 50 even though there may not be many features.
I also implemented MAP Maximum a posteriori (`argmax P(theta|D)`) and MLE with `P(D|theta)` Expectation Maximization - but I did not test them. I was hoping they might improve the speed as they are Naive Bayes relevant variations and make independent assumptions.

I implemented an NMF version of Naive Bayes, since PCA has the potential to generate negative numbers which can throw the NB as it is based on counts and cannot be negative.

I also implemented a Single Layer ADALINE for LMS update with Gradient descent. But it was slow. I then implemented a 2 layer network for LMS with activations, softmax since Online Linear Regression LMS requires to convert results to logits to get max probability argmax and the prediction label. This was slow on agate but it runs very fast on Colab due to the GPU.

Just curious, so I printed the image out one with RGB and other as it is:





All algorithms work on small data. I think additional data structures choices or buffering or parallelism (use of threads) would be better approach for this.

2. What can you say about the time and space complexity of your program?

KNN: For n samples, m classes, m^n -1 entries for each label so K labels => K (m^n -1) => exponential in n. -> space complexity for KNN approximately.

For n samples, d features, O(nd) is the time complexity. KNN seems expensive in terms of computation.

LMS/Linear: For n samples, X -> n * d, Weights = n. It is a training step update for LMS. The computational complexity seems lower for LMS. The convergence rate depends on the step size, and the eigenvalues. The data we have is sparse as noted by the professor in the class. I was hoping Online Linear LMS would work for the digits image, due to the sparse data and the filter detection.

NB: n samples, d features and m classes, O(ndm) since O(dm) per prediction. Space complexity is O(dc). NB suffers from the curse of dimensionality.

3. For each algorithm you implemented, describe a learning problem that you think it will perform poorly on and explain why.

KNN: KNN is known for classification problems. The choice of k can affect model accuracy and can influence bias and variance. But it has a high computational cost. Mainly any problems that have high dimensional data, KNN would be expensive and performs poorly due to aforementioned reasons.

Linear/LMS: perform poorly on non-linear data and can also lead to slower convergence. Stability can be impacted is step size is not chose appropriately.
Naive Bayes: assumes all features are independent. It is known for poor estimation of probabilities. It can easily show correlation leads to causation which is dangerous. Problems such as causal event weather/rain predictions, causal data are known to perform poorly with NB.

4. What suggestions do you have for improving this assignment in the future?

Syllabus says “You may use any programming language you wish in this course.”
But assignment 10 says “Python might not be the best choice for this assignment, as runtime efficiency is important. It might be helpful to keep in mind that the number of instances is much larger than the number of attributes which is larger than the number of classes.”
I still proceeded with Python as I had only 1.5 days available for the homework and do not have other programming language environments.