Jimmy Wallace

Dr. Fosler-Lussier

CSE 5526: Introduction to Neural Networks

14 April 2020

Lab 3 Report - Collaborative Filtering

Introduction:

In this lab, students were tasked with the goal of designing and creating a Restricted Boltzmann Machine to perform collaborative filtering on a provided data set. The goal of collaborative filtering is to correlate visible input features with hidden features so that for a given visible input feature, it can be reconstructed by the hidden feature with a high degree of correlation. These machines inherit their correlation abilities from their unique, fully connected, bipartite structure and weight update rule derived from contrastive divergence. The data set provided in this lab contains numerous samples of ice cream preferences according to flavor and brand. We wish to determine a set of hidden features from this data set that is highly correlated to the ice cream preference input features to demonstrate the correlating abilities of Restricted Boltzmann Machines. We also wish to measure and analyze the efficacy of the Restricted Boltzmann Machine with varying architectures to determine the effect of architecture and hyperparameters on the performance of Restricted Boltzmann Machines.

Procedure:

Part I

To perform collaborative filtering, we must first initialize and train a Restricted Boltzmann Machine. For Part I, we are instructed to use a hidden layer size of 4 nodes. So, because each ice cream preference sample contains 10 data points, the weight matrix of this Restricted Boltzmann Machine will be 10 x 4. Because these types of machines are restricted to containing to disjoint sets of nodes, their data structure can be easily represented by a two-dimensional matrix of weights. To begin, each weight is initialized to a uniformly distributed real value between 0 and 1. Once the weights have been initialized, we can start the training process to update the weights provided in the matrix and maximize the likelihood of the visible input feature from the generated hidden feature. For this lab, I decided to implement a batch training process to help execute computations quickly. So once training begins, the Restricted Boltzmann Machine will continue to train for a given number of epochs. With each epoch, the provided features matrix will be shuffled to ensure randomness and then batch processed. Each visible input feature will be used to generate a hidden feature and each hidden feature will be used to regenerate the visible input feature. The reconstructed visible input feature, not necessarily identical to the original visible input feature, is then used to generate the second hidden feature. The generative and regenerative process simply computes the weighted product of the given feature with the bidirectional weights matrix to compute its sigmoid, or respective likelihood. Then, the likelihood is randomly sampled to return either a -1 or 1, the domain of the features. After each epoch, the weights matrix will be updated by the weight update rule derived by contrastive divergence given below.

After the Restricted Boltzmann Machine has trained for a desired number of epochs, its weight matrix is trained and ready to start correlating visible input features and hidden features. To determine the efficacy of our model, we decided to compare absolute mean errors. The absolute mean error measures the average number of disagreeing feature nodes from given visible input features and the generated visible features. A Restricted Boltzmann Machine with a small absolute mean error correlates visible input features with hidden features well whereas a large absolute mean error implies low correlation between layers.

Part II

For this portion of the lab, students were tasked with experimenting with varying Restricted Boltzmann Machine architectures and hyperparameters to compare with the base model defined in Part I. I chose to vary the number of hidden nodes in the hidden layer and implement search and converge learning to compare to the base model. To implement a Restricted Boltzmann Machine with an arbitrary number of hidden nodes, one must accordingly construct the weights matrix with correct dimensions. Aside from the weight matrix initialization, the training and predicting process is identical to the base model. I used a Restricted Boltzmann Machine Class to represent this type of neural network with general input and hidden layer dimensions. So, implementing the architecture change of hidden layer size was trivial. On the other hand, I also implemented search and converge learning which uses a dynamic learning rate that is given by the equation seen below.

To implement this design change, I had to refactor my training method to optionally use search and converge learning if desired. This dynamic learning rate eagerly searches for a solution first and then calms down and starts to focus on minimizing error and taking small steps rather than large jumps towards its desired weights.

After implementing a Restricted Boltzmann Machine Class capable of varying architectures, I tested each unique machine with various learning rates for a given number of epochs and plotted their results. These graphs visually represent the efficacy of Restricted Boltzmann Machines with varying architectures and hyperparameters.

Results:

To test the efficacy of each Restricted Boltzmann Machine model, each model was trained with the learning rates 0.001, 0.01, and 0.1 for a given number of epochs to explore the effects of learning rates on training. Each graph listed below details the efficacy results of one Restricted Boltzmann Machine with given architecture detailed in the graph. Moreover, some graphs utilize a search and converge constant of 0; by design, this implies that the given Restricted Boltzmann Machine does not use search and converge learning.

Base Model

A screenshot of a cell phone

Description automatically generated

The base model featured an architecture with four hidden nodes and stochastic learning. The model converged towards an absolute mean error of roughly 3.1 between each respective learning rate. So, this implies that on average a reconstructed visible input feature will have three errors with respect to the original visible input feature. Also, as deduced from the graph above, as the learning rate of the model increased, the initial slope increased. This is because larger learning rates imply model confidence in weight update rules.

Increased Number of Hidden Nodes Model

A screenshot of a cell phone

Description automatically generated

The next model features an increased number of hidden nodes. This model utilizes a hidden layer size of 8 nodes to increase the hypothesis space served by this model. With added complexity by the means of additional hidden nodes, more complex state spaces can be defined. So, from the results above, we see that each model converged towards an average absolute error of 2.5.

Search and Converge Learning Model

A screenshot of a cell phone

Description automatically generated

The next model features search and converge learning which is a method of learning that features a dynamic learning rate. To start, search and converge training eagerly searches for solutions with large steps and as training progresses begins to take smaller steps and focus on minimizing error. This model seemed to converge towards an absolute average error of 3.2.

Increased Number of Hidden Nodes and Search and Converge Learning Model

A screenshot of a cell phone

Description automatically generated

This model features increased number of hidden nodes and search and converge learning. As you can see, this model converges towards an absolute average error of roughly 2.5. It is important to note that this model with an aggressive learning rate of 0.1 approached an absolute average error of 1.6, which is exceedingly low given the results of the other models.

Discussion:

As briefly mentioned in the introduction, the purpose of this experiment is to measure and compare the efficacy of varying Restricted Boltzmann Machines. From the results produced in the subsequent section, we deduce that the addition of more hidden nodes provides the most dramatic increase in model accuracy and performance. This follows logic though because by increasing the number of hidden nodes in the hidden layer, more parameters are added to the weight matrix provided more degrees of freedom to define state spaces. So, it is understandable that increasing the size of the hidden layer increases the accuracy of the model. Moreover, search and converge learning provided a great method for quickly arriving at a solution but, search and converge tends to oscillate enthusiastically as it continues to train which is undesirable. Although the two models defined in Part II seemed to outperform the base model, the base model provided a clear training process to its desirable weights. Also, the goal of collaborative filtering is not necessarily to reconstruct identically the input feature but determine correlated features between input features to determine patterns. This concept is quite important in marketing practices for many companies today.

Lastly, it is important to note that none of these models converged to a single value because of the inherent randomness in the sampling process. The inherent randomness ensures that guaranteed convergence is unlikely.

Conclusion:

The purpose of this experiment was to familiarize yourself with Restricted Boltzmann Machines and their collaborative filtering abilities. By training varying models of Restricted Boltzmann Machines, I was able to determine the effects of architecture on the performance of these machines. After looking at the results of training, I deduced that adding additional hidden nodes provided the most dramatic change to model performance whereas search and converge training seemed to reach a solution quickly then oscillate. However, the base model also provided valuable knowledge because it was able to correlate input features to other input features with an absolute average error less than that of random sampling. So, the base model could quite possibly be the best model to determine ice cream preferences from liked ice cream flavors.