EVALUATION OF MACHINE LEARNING CLASSIFICATION ALGORITHMS USING MNIST

Saba Kanwal(18L-1860)

October 8, 2018

Abstract

In this report, I evaluated performance of some machine learning classification algorithms on MNIST Dataset of Hand Written Digits. Algorithms discussed in this report are KNN, Perceptron learning, Gradient Descent using Sigmoid Unit, and Multi-layer neuron with one hidden layer. In this report, these algorithms are evaluated based on how much time they take to classify an example and how many unseen examples are correctly classified by them. KNN gives highest accuracy of 98% on testing data with value of k=1. The value of hyper parameter k is selected with the help of validation data. But KNN is lazy algorithm it takes about 60 hours for 1 value of k. But the other algorithms return result in constant time for one example if their tune parameters are computed.
Keywords— KNN, lazy algorithm, Gradient Descent, Sigmoid unit, Percepton, Multi-layer Neural Network,hyper parameter

Chapter 1

Introduction

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In this report, we studied different classification algorithms to recognize numeric digits from 0 to 9. So, our dataset consist of 10 classes or labels. The classification algorithms that are analyzed in this report are described below.

1.1 K-Nearest Neighbor

KNN is one of the highly used classification algorithm. In this, an object is classified with the help of its k neighbors. If we have k neighbors, then majority voting is used to assign a label to an unseen instance. But there are two questions. How to measure the neighbors of an object and what value of k will be used to predict an object?

1.1.1 Distance Metric

For determining neighbors, many distance metrics are used. We only discussed two of them that are used for our classification task. One is Euclidean distance and the other is cosine similarity.

Euclidean distance: Euclidean distance function is mostly used to measure distance between two vectors in vector space. In our task, images are vectors. So, we can measure distance of an unseen instance to all of the images used for training. Then k images are selected that give minimum distance. if more than one classes inside k nearest images then class with majority vote is selected. Let A and B are two images of m dimensions the Euclidean distance between these

can be computed as

$$dist(A,B) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{1.1}$$

Cosine Similarity: Cosine similarity measure is typically used to calculate similarity between two vectors. The higher value means most similar vectors. The cosine similarity between two vectors can be computed as

$$sim(A,B) = \frac{\vec{A}.\vec{B}}{|\vec{A}||\vec{B}|} \tag{1.2}$$

1.1.2 Parameter Selection

The best choice of k depends upon the data; generally, larger values of k reduces effect of the noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques. We find value of k by splitting training data into training and validation data. Then different values of k are used to find accuracy on validation data. Then we select that value that give highest accuracy on validation data.

1.2 Perceptron Learning

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. Perceptron is a function that maps its input x (a real-valued vector) to an output value f(x)(a single binary value).

$$\sum_{i=1}^{m} w_i x_i > 0 \; then \; f(x) = 1 \; else \; f(x) = 0 \tag{1.3}$$

In the above equation,w is a weight vector and m is the number of inputs to the perceptron (plus bias term). The complete algorithm is as follows:

1. Initialize the weights and threshold to small random numbers
2. Present a vector x to the neuron inputs and calculate the output.
3. Update the weights according to:

$$w_j(t+1) = w_j(t) + \eta(d-y)x \tag{1.4}$$

   where

   (a) d is the desired output,

   (b) t is the iteration number

   (c) eta is the step size or learning rate

4. repeat steps 2 and 3 until:

   (a) the iteration error is less than a user-specified error threshold or

   (b) a predetermined number of iterations have been completed.

1.3 Gradient Descent using Sigmoid Unit

Gradient decent is like perceptron learning algorithm. But instead of updating weight vector after each misclassification it update weights after one pass of training examples. The weights are updated by the following formula

$$w_j(t+1) = w_j(t) + \eta \sum_{i=1}^{n} (y_i - o_i) o_i (1 - o_i) X_{ij} \tag{1.5}$$

In above equation, yi is the desired output, oi is the predicted output using sigmoid function, and eta is the learning rate and take its value very small. Termination condition is the same as in the perceptron learning. The sigmoid function that used in place of threshold function is as follows

$$output = \frac{1}{\frac{1}{1+e^{-sum}}} \tag{1.6}$$

1.4 Multi-layer Neural Network

Perceptron learning is a single layer network. If the network contains more than one layers, the it is called multi-layer neural network. We implemented a two layer neural network that has inputs, outputs same as perceptron but there is also a hidden layer that takes input from the input examples and output layer use results of hidden layer as input. This network can learn many complicated function that cannot be learned by perceptron.

Chapter 2

Dataset

The dataset that are used for measuring and comparing performance of machine learning classification algorithm is taken from the Institute of Standards and Technology database (MNIST). This dataset is composed of 60,000 training images and 10,000 testing images. Each image is of 28x28 dimensions. Total pixels in a single image is 784. And pixel values ranges from 0 to 255 in a digit image. For Finding value of hyper parameters training data set is split in to two parts. Training data and validation data. Training data contains 50,000 images and validation data contains 10,000 images.

Chapter 3

Experiments and Results

First of all all of the algorithms are implemented in python. The detail of experiments performed on the algorithms are described in below sections

3.1 K-Nearest Neighbour

3.1.1 KNN using Euclidean distance

For determining, what value of k will be used for measuring accuracy of KNN. The training data is divided into two parts training and validation dataset. Now, training dataset contains 50,000 training images and validation dataset contains 10,000 images. For odd values of k from 1 to 15, its accuracy measured using training dataset and validation dataset. The result of this experiment for different values of k are given below:

K	Size of training	Size of validation	Accuracy	Time
1	50,000	10,000	98.82%	60 hour
3	10,000	100	96.62%	30 min
5	10,000	100	96.52%	30 min
7	10,000	100	96.63%	30 min
11	10,000	100	95%	30 min
13	10,000	100	95%	30 min
15	10,000	100	95%	30 min

KNN takes too much time to compute results. So, its not possible for me to wait a lot of time for results. For faster results, i reduce the training dataset size and validation dataset size as shown in above table. The value of k that gives highest accuracy on the validation dataset is k=1. So, value of k=1, the testing dataset accuracy is given below in table.

K	Size of training	Size of testing	Accuracy	Time
1	40,000	5,000	97%	28 hour

3.1.2 KNN using Cosine Similarity

The results of accuracy using cosine similarity on validation data set is given below

K	Size of training	Size of validation	Accuracy	Time
1	10,000	100	99%	20 min
3	10,000	100	99%	20 min
5	10,000	100	99%	20 min
7	10,000	100	98%	20 min
11	10,000	100	98.7%	20 min
13	10,000	100	99%	20 min
15	10,000	100	99%	20 min

For different values of k, the accuracy on validation dataset remains approximate same. So, for measuring accuracy of Cosine similarity, the value of k used is also k=1. The accuracy of Cosine similarity on testing data is shown in table below

K	Size of training	Size of testing	Accuracy	Time
1	60,000	100	98.9%	25 min

3.2 Perceptron Learning

Perceptron gives binary output. But we have to classify data into 10 classes. To use perceptron learning, we perform some preprocessing on our training data. Then we train 10 perceptrons. Each perceptron is trained for individual digit. Some difficulties faced in perceptron leaning are

1. Algorithm not terminate because error not reduces to zero. After that i included step size to 50 in termination condition.

2. After converging to some certain value, the misclassification values repeats again and again in order

For training each perceptron, I used 40,000 examples and step size of 20. it takes more than one day to give results. The trained perceptron misclassification on training data is given in table below

Perceptron	Size of training	Misclassification on training
0	40,000	565
1	40,000	234
2	40,000	390
3	40,000	600
4	40,000	520
5	40,000	300
6	40,000	450
7	40,000	230
8	40,000	90
9	40,000	600

The total misclassifications of all perceptron value is about 3,444. The correctly classified instances of training at last step 36,556. Means it fits about 86% of training dataset. The accuracy on testing data set is given below

Size of training	Size of testing	Accuracy	Time to Predict
40,000	10,000	80%	1 to 2 min

3.3 Gradient Descent using Sigmoid function

The issues faced during sigmoid function:

1. learning rate = 0.5 its exponent goes out of range after 2 to 3 iteration. After no convergence
2. For weight values greater than 0.5, exponent goes out of range and after no convergence

To avoid the above issues, I used value of learning rate = 0.1 and weight values less than 0.1. But this converges too much slowly. I repeat this process for 50 steps, it takes about 20 hours to compute results.The results of its on testing data is so bad than other algorithms.

Size of training	Size of testing	Accuracy	Time to Predict
40,000	10,000	63.5%	1 to 3

3.4 2-layer Neural Network

Some preprocessing is performed on training data set to make it compatible for 2-layer neural network. The result produces for single example from this neural network is vector of size 10. So, for computations, the training labels of each example is also converted to vector of size 10. The accuracy of this two layer neural network is given below:

Size of training	Size of testing	No. of Steps	Accuracy
10,000	10,000	20	53.5%

Chapter 4

Discussion and Conclusion

If we compare the above algorithms, by using accuracy value KNN gives highest value of accuracy of about 98%. KNN is also a simplest algorithm and easy to understand. But it is lazy loading algorithm means when a test examples comes it compute result by computing distance to all of the training examples. Its response time will be very high as compared to other algorithms. Other algorithms results affected by different values of initial weights and learning rate. These will be choose very carefully, otherwise results will not converge. But if we have training data, then we can compute weight vector and save it somewhere. When new example comes, its result can be computed by using these weight vector. This can respond in secs. And the accuracy of these algorithms can also be improved much more by choosing above mentioned parameters very carefully. And also the number of times these procedures repeats themselves for updating weight vector will be maximum as well as possible.

Perceptron converges much more faster than gradient descent. Because it update its weight after each misclassification. But it can fail if more than one local minimum exists. Then it can stuck in local minimum not in global minimum. But gradient descent can avoid this local minimum issue, because it uses all of the training examples to update their weights. But if their is very complicated function that cannot be fit by using percepron learning and gradient descent. Then concept of multi-layer neural network comes. It can fit any type of data accurately.

We can conclude that, No one algorithm is better in all of the situations. Choice of algorithm changed according to complexity of data, accuracy and time requirements. If accuracy and simplicity is of interest then KNN will be used for classification of anything. But if time to respond is also important, then we have to some other according to complexity of data.