Assignment 2

Neural Networks and Deep Learning

CSCI 5922

Spring 2017

Assigned Tue Sep 12, 2017

Due: Tue Sep 26, 2017

Assignment submission

Please read instructions here.

Goal

The goal of this assignment is to build a one-hidden-layer back propagation network to process real data. For this assignment, I want you to implement the neural net (activation function and training code) yourself, not using tensorflow or other software tools. The purpose is for you to understand the nitty gritty of what these tools are doing for you before we switch over to using the tools.

Data Set

I picked a data set from the UCI Machine Learning Repository, a nice source of data set. It consists of experimental data used for binary classification of room occupancy (i.e., room is occupied versus empty) based on temperature, humidity, light, and CO_2 sensors. The training and test data sets are each collected over a week period. Information about the data set and how it has been used in academic publications can be found here.

The data set includes time stamps with date and hour/minute/second within the day. You are not to use time stamp features for predicting occupancy. Since this is a commercial office building, the time stamp is a strong predictor of occupancy. Rather, the goal is to determine

whether occupancy can be sensed from: (1) temperature, expressed in degrees Celsius, (2) relative humidity, expressed as a %, (3) light, in lux, (4) CO_2, in ppm, and (5) the humidity ratio, which is derive from the temperature and the relative humidity.

The training data are to be found here. The test data are to be found here. There are 8144 training examples and 9753 test examples.

Part 1

Using the perceptron code you wrote for Assignment 1, train a perceptron (linear activation function with a binary threshold) using the training set. Your perceptron should have the 5 input variables described above.

(1a) Report the training and test set performance in terms of % examples classified correctly.

Remember an important property of the perceptron algorithm: it is guaranteed to converge only if there is a setting of the weights that will classify the training set perfectly. (The learning rule corrects errors. When all examples are classified correctly, the weights stop changing.) With a noisy data set like this one, the algorithm will not find an exact solution. Also remember that the perceptron algorithm is not performing gradient descent. Instead, it will jitter around the solution continually changing the weights from one iteration to the next. The weight changes will have a small effect on performance, so you'll see training set performance jitter a bit as well.

Part 2

Write your own code to implement a feedforward neural net with a single hidden layer. You should have 5 input units, H hidden units, and 1 output unit. Write your own code to train the network with back propagation. Write your code so that it loops over training epochs, and within a training epoch it chooses mini-batches of size N, where N can range from 1 to the total number of examples in the training set. As a way of testing your code, set the learning rate very small and set N to be the training set size (i.e., you're doing batch training). In this situation, you should be guaranteed that the error monotonically decreases over epochs of training.

(2a) Decide what error function you wish to use. Two obvious candidates are squared error and cross entropy. Report which you have picked.

(2b) As a way of verifying that your network learns something beyond prior statistics of the training set, let's compute a measure of baseline performance. As a measure of baseline, use the training set to determine the constant output level of the network, call it C, that will minimize your error measure. That is, assume your net doesn't learn to respond to the inputs, but rather gives its best guess of the output without regard to the input. Then for any example where the target is 0 the network will output C and for any example where the target is 1 the network will

output C. Using your error measure, solve for C and compute the baseline error. Report the baseline error.

(2c) Using a network with H=5 hidden units, and mini-batches of size N=100, select a learning rate (or a learning rate schedule) that results in fairly consistent drops in error from one epoch to the next, make a plot of the training error as a function of epochs. On this graph, show a constant horizontal line for the baseline error. If your network doesn't drop below this baseline, there's something going awry. For now, train your net until you're pretty sure the training error isn't dropping further (i.e., a local optimum has been reached).

(2d) Report the learning rate (or learning rate schedule) you used to produce the plot in (2c)

(2e) Report training and test set performance in terms of % examples classified correctly.

(2f) Now train nets with varying size, H, in {1, 2, 5, 10 20}. You may have to adjust your learning rates based on H, or use one of the heuristics in the text for setting learning rates to be independent of H. Decide when to stop training the net based on training set performance. Make a plot, as a function of H, of the training and test set performance in terms of % examples classified correctly.

Part 3 (Extra Credit)

See how much adding information about time of day helps the network. Add a new set of inputs that represent the time of day. (Don't add information about day of week or absolute date.)

(3a) Determine an appropriate representation for the time of day. Describe the representation you used. For example, you might add one unit with a value ranging from 0 to 1 for times ranging from 00:00 to 23:59. Report the representation you selected.

(3b) Train your net with H=5 hidden and compare training and test set performance to the net you built in (2e)

https://www.cs.colorado.edu/~mozer/Teaching/syllabi/DeepLearningFall2017/assignments/assignment2.html