10807, Assignment 1

Out Sep 13th, Due Sep 28th, 2016 at midnight.

Write up

Hand in answers to all the questions in the parts above. For the last question, the goal of your write-up is to document the experiments you have done and your main findings. So be sure to explain the results. Be concise and up to the point – do not write long paragraphs, vaguely explaning results.

- The answers to all the questions should be in pdf form. Please use latex. You can find style files on the course webpage. Do not forger to include you name, your andrew ID, and your e-mail.
- Please include a README file with instructions on how to execute your code. Your code should implement a backpropagation algorithm with cross-entropy error function for one-layer and two-layer neural networks.
- Package your code and a copy of the write-up `pdf` document using `zip` or `tar.gz` in a file called `10807-A1-*yourandrewid*.[zip|tar.gz]`.
- Submit your assignemt to gradescope.

Problem 1 (6 pts)

This question will test your general understanding of overfitting as they relate to model complexity and training set size. Consider a continuous domain and a smooth joint distribution over inputs and outputs, so that no test or training case is ever duplicated exactly.

- For a fixed training set size, sketch a graph of the typical behavior of training error rate (y-axis) versus model complexity (x-axis). Add to this graph a curve showing the typical behavior of the corresponding test error rate versus model complexity, on the same axes. (Assume that we have an infinite test set drawn independently from the same joint distribution as the training set). Mark a vertical line showing where you think the most complex model your data supports is; chose your horizontal range so that this line is neither on the extreme left nor on the extreme right. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing complexity rightwards.

- For a fixed model complexity, sketch a graph of the typical behavior of training error rate (y-axis) versus training set size (x-axis). Add to this graph a curve showing the typical behavior of test error rate versus training set size, on the same axes (again on an iid infinite test set). Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing training set size rightwards.

Problem 2 (12 pts)

In this question we will study the expected loss for regression problems. Consider the expected L_q loss function given by:

$$\mathbb{E}[L_q] = \iint |y(\mathbf{x}) - t|^q p(\mathbf{x}, t) \mathrm{d}t \mathrm{d}\mathbf{x} \tag{1}$$

Show that:

a) The minimum expected loss for $q = 1$ is given by the conditional median (i.e. by the function $y(\mathbf{x})$ such that the probability mass for $t < y(\mathbf{x})$ is the same as for $t \geq y(\mathbf{x})$).

b) The minimum expected loss for $q \to 0$ is given by the conditional mode (i.e. by the function $y(\mathbf{x})$ equal to the value of t that maximizes $p(t|\mathbf{x})$ for each $\mathbf{x}$).

Problem 3 (10 pts)

Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a neural network output $f(\mathbf{x}, \mathbf{w})$ that represents $p(t = 1|\mathbf{x})$, and suppose that there is a probability ϵ that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the error function corresponding to the negative log-likelihood. Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.

Problem 4 (12 pts)

Consider following distribution

$$p(x|\sigma, q) = \frac{q}{2(2\sigma^2)^{1/q}\Gamma(1/q)} \exp\left(-\frac{|x|^q}{2\sigma^2}\right)$$

which is a generalization of the univariate Gaussian distribution.

- (6 pts) Show that this distribution is normalized so that it represents a valid probability distribution and that it reduces to the Gaussian when $q = 2$.
- (6 pts) Consider a regression model in which the target variable is given by $t = y(x, w) + \epsilon$ and ϵ is a random noise variable drawn from the above distribution. Derive the log likelihood function over w and σ^2, for an observed data set of input vectors $X = \{x_1, ..., x_N\}$ and corresponding target variables $t = (t_1, ..., t_N)^\top$.

Problem 5 (60 pts)

For this question you will write your own implementation of backpropagation algorithm for training your own neural network. Please do not use any toolboxes. We recommend that you use MATLAB or Python, but you are welcome to use any other programming language if you wish.

The goal is to label images of 10 handwritten digits of "zero", "one",..., "nine". The images are 28 by 28 in size (MNIST dataset), which we will be represented as a vector $\mathbf{x}$ of dimension 784 by listing all the pixel values in raster scan order. The labels t are 0,1,2,...,9 corresponding to 10 classes as written in the image. There are 3000 training cases, containing 300 examples of each of 10 classes, 1000 validation (100 examples of each of 10 classes), and 3000 test cases (300 examples of each of 10 classes). they can be found in the file digitstrain.txt, digitsvalid.txt and digitstest.txt:

http://www.cs.cmu.edu/∼rsalakhu/10807_2016/assignments.html
Format of the data: digitstrain.txt contains 3000 lines. Each line contains 785 numbers (comma delimited): the first 784 real-valued numbers correspond to the 784 pixel values, and the last number denotes the class label: 0 corresponds to digit 0, 1 corresponds to digit 1, etc. digitsvalid.txt and digitstest.txt contain 1000 and 3000 lines and use the same format as above. As a warm up question, load the data and plot a few examples. Decide if the pixels were scanned out in row-major or column-major order.

Backpropagation Algorithm

Implement backpropagation algorithm with sigmoid activation function in a single-layer neural network. The output layer should be a softmax output over 10 classes corresponding to 10 classes of handwritten digits. Your backprop code should minimize the cross-entropy entropy function for multi-class classification problem.

a) Basic generalization [5 points]

Train a single layer neural network with 100 hidden units (with architecture: $784 \rightarrow 100 \rightarrow 10$). You should use initialization scheme discussed in class as well as choose a reasonable learning rate (e.g. 0.1). Train the network repeatedly (more than 5 times) using different random seeds, so that each time, you start with a slightly different initialization of

the weights. Run the optimization for at least 200 epochs each time. If you observe underfitting, continue training the network for more epochs until you start seeing overfitting.

Plot the average training cross-entropy error (sum of the cross-entropy error terms over the training dataset divided by the total number of training example) on the y-axis vs. the epoch number (x-axis). On the same figure, plot the average validation cross-entropy error function.

Examine the plots of training error and validation error (generalization). How does the network's performance differ on the training set versus the validation set during learning? Use the plot of error curves (training and validation) to support your argument.

b) Classification error [3 points]

You should implement an alternative performance measure to the cross entropy, the mean classification error. You can consider the output correct if the correct label is given a higher probability than the incorrect label. You should then count up the total number of examples that are classified incorrectly (divided by the total number of examples) according to this criterion for training and validation respectively, and maintain this statistic at the end of each epoch. Plot the classification error (in percentage) vs. number of epochs, for both training and validation. Do you observe a different behavior compared to the behavior of the cross-entropy error function?

c) Visualizing Parameters [3 points]

Visualize your best results of the learned W as 100 28x28 images (plot all filters as one image, as we have seen in class). Do the learned features exhibit any structure?

d) Learning rate [4 points]

Try different values of the learning rate ϵ. You should start with a learning rate of 0.1. You should then reduce it to $.01$, and increase it to 0.2 and 0.5. What happens to the convergence properties of the algorithm (looking at both average cross entropy and % Incorrect)? Try momentum of $\{0.0, 0.5, 0.9\}$. How does momentum affect convergence rate? How would you choose the best value of these parameters?

e) Number of hidden units [5 points]

Set the learning rate ϵ to $.01$, momentum to 0.5 and try different numbers of hidden units on this problem. You should try training a network with 20, 100, 200, and 500 hidden units. Describe the effect of this modification on the convergence properties, and the generalization of the network.

f) Dropout [8 points]

Implement backpropagation with dropout by setting each hidden unit to 0 with probability 0.5. Try training a larger network with dropout. Describe the effect of this modification on the convergence properties, and the generalization of the network.

g) Best performing single-layer network [12 points]

Cross-validation: Explore various model hyper-parameters, including

- learning rates
- momentum
- number of hidden units in each layer
- number of epochs (early stopping)
- L_2 regularization (weight decay)
- Dropout values

to achieve the best validation accuracy. Briefly describe your findings.

Given the best found values, report the final performance of your 1-layer neural network (both average cross entropy and % Inorrect) on the training, validation, and test sets. Visualize your best results of the learned W as 28x28 images (plot all filters as one image, as we have seen in class).

h) Extension to multiple layers [20 points]

Implement a 2-layer neural network, starting with a simple architecture containing 100 hidden units in each layer (with architecture: $784 \rightarrow 100 \rightarrow 100 \rightarrow 10$).

Cross-validation: Explore various model hyper-parameters, including learning rates, momentum, number of hidden units in each layer, number of epochs, dropout, and weight decay to achieve the best validation accuracy. Briefly describe your findings.

Given the best found values, report the final performance of your 2-layer neural network (both average cross entropy and % Incorrect) on the training, validation, and test sets. Visualize your best results of the learned 1st-layer W as 28x28 images (plot all filters as one image, as we have seen in class). How do these filters compare to the ones you obtained when training a single-layer network?

Does 1-layer network outperform a 2-layer model in term of generalization capabilities?