HOMEWORK 1

CMU 10-707: DEEP LEARNING (FALL 2017)
https://piazza.com/cmu/fall2017/10707
OUT: Sept 11, 2017
DUE: Sept 25, 2017 11:59 pm
TAs: Dheeraj Rajagopal, Yao-Hung (Hubert) Tsai

Problem 1 (6 pts)

This question will test your general understanding of overfitting as it relates 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.

1. 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; choose your horizontal range so that this line is neither on the extreme left nor on the extreme right. Indicate on your vertical axis where zero error is and draw your graphs with increasing error upwards and increasing complexity rightwards.

2. 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 axis where zero error is and draw your graphs with increasing error upwards and increasing training set size rightwards.

3. One of the commonly used regularization methods in neural networks is *early stopping*. Argue qualitatively why (or why not) early stopping is a reasonable regularization metric.

Problem 2 (8 pts)

Consider N training points (x_i, y_i) drawn i.i.d from a distribution that is characterized as:

$$x_i \sim h(x) \tag{1}$$
$$y_i = f(x_i) + \epsilon_i \tag{2}$$
$$\epsilon_i \sim (0, \sigma^2) \tag{3}$$

where f is the regression function. An estimator for this data, *linear* in y_i is given by

$$\hat{f}(x^*) = \sum_{i=1}^{N} l_i(x^*; \mathcal{X}) y_i, \tag{4}$$

where $l_i(x^*; \mathcal{X})$ depends on the entire training sequence of x_i (denoted by $\mathcal{X}$) but do not depend on y_i. Show that k-nearest-neighbor regression and linear regression are members of this class of estimators. What would be the $l_i(x^*; \mathcal{X})$ in both these regressions (knn and linear)?

Problem 3 (6 pts)

The form of Bernoulli distribution, given by:

$$\text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x},$$

is not symmetric between the two values of $x \in \{0, 1\}$. Often, it will be convenient to use an equivalent formulation for which $x \in \{-1, 1\}$, in which case the distribution can be written as:

$$p(x|\mu) = \left(\frac{1-\mu}{2}\right)^{(1-x)/2}\left(\frac{1+\mu}{2}\right)^{(1+x)/2},$$

where $\mu \in [-1, 1]$. Show that this new distribution is normalized and compute its mean, variance, and entropy.

Problem 4 (12 pts)

Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a neural network output $y(x, w)$ that represents $p(t = 1|x; w)$, 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. What is the error function when $\epsilon = 0$? Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.

Problem 5 (8 pts)

Consider a two-layer network function in which the hidden-unit nonlinear activation functions are given by logistic sigmoid functions of the form:

$$\sigma(a) = \frac{1}{1+\exp(-a)} \tag{5}$$

Show that there exists an equivalent network, which computes exactly the same function, but with hidden unit activation functions given by $\tanh(a)$ where the tanh function is defined by:

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{6}$$

Problem 6 (60 pts)

For this question you will write your own implementation of the 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/10707/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 the 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 (e.g. with architecture: $784 \rightarrow 100 \rightarrow 10$). You should use the initialization scheme discussed in class and choose a reasonable learning rate (i.e. 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 [5 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 [5 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 [5 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) Best performing single-layer network [10 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)

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 % incorrect) 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).

g) Extension to multiple layers [10 points]

Implement a 2-layer neural network, starting with a simple architecture containing 100 hidden units in each layer (e.g. 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, 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 the 1-layer network outperform a 2-layer model in term of generalization capabilities?

h) Batch Normalization [10 points]

For your two-layer network, implement batch normalization for a batch size of 32. Describe how batch norm helps (or doesn't help) in terms of speed and accuracy (train and validation).

i) Different Activation Functions [5 points]

Now, try changing the activation function to ReLU and tanh instead of the original sigmoid. Do you see any difference in terms of performance or accuracy ? Report your findings.

Write up

Hand in answers to all questions above. For Problem 6, 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 to the point – do not write long paragraphs or only vaguely explain results.

- The answers to all questions should be in pdf form (please use LaTeX).
- 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 README document using `zip` or `tar.gz` in a file called `10707-A1-*yourandrewid*.[zip|tar.gz]`.
- Submit your PDF write-up to the Gradescope assignment "Homework 1" and your packaged code to the Gradescope assignment "Code for Homework 1."