Homework Assignment 1

CSE 253: Neural Networks

Winter 2017

Instructions

Due Wednesday, January 18^{th}:

1. Please hand in your assignment via Vocareum. This has not been set up yet for this class, more detailed instructions will be posted here later. We prefer a report written using LaTeXin NIPS format for each assignment. You are free to choose an alternate method (Word, etc.) if you want, but we still prefer NIPS format.

2. You may use a language of your choice (Python and MATLAB are recommended); You should submit your code on Vocareum along with your report. For your own purposes, keep your code clean with explanatory comments, as it may be reused in the future.

3. Using the MATLAB toolbox for neural networks or any off-the-shelf code is strictly prohibited except where noted.

4. For the "Problems from Bishop" part, please do your own work. You can discuss the assignment with other classmates, as long as you follow the Gilligan's Island Rule (see below). Books, notes, and Internet resources can be consulted, but not copied from. Working together on homeworks must follow the Gilligan's Island Rule (Dymond, 1986): No notes can be made during a discussion, and you must watch one hour of Gilligan's Island or equivalently insipid activity before writing anything down. Suspected cheating will be reported to the Dean.

5. For the programming part (Logistic and Softmax regression), please work in pairs. In *extraordinary circumstances* (e.g., you have a highly contagious disease and are afraid of infecting your teammate), we will allow you to do it on your own. Please discuss your circumstances with your TA, who will then present your case to me.

Problems from Bishop (20 points)

Work problems 1-4 (5 points each) on pages 28-30 of Bishop. *Note: In Equation 1.51, the argument of exp should be* $(-\epsilon^2/\sigma^2)$. This should be done individually, and each team member should turn in his or her own work separately

Logistic and Softmax Regression (35 points)

This part will be done in pairs. If you want to do it with three people, the work better be extraordinarily good. In this problem, we will classify handwritten digits from Yann LeCun's website for the MNIST Database. Please download the four files found there, these will be used for this problem. To reduce computation time, we will use only a subset of these files. Please use only the first 20,000 training images/labels and only the first 2,000 testing images/labels. These are easier patterns than the second half of the test set.

Logistic Regression

Although we consider logistic regression to be a classification technique, it is called "regression" because it is used to fit a continuous variable: the probability of the category, given the data. Logistic regression can be modeled as using a single neuron reading in an input vector $(1, x) \in \mathbb{R}^{d+1}$ and parameterized by weight vector $w \in \mathbb{R}^{d+1}$. d is the dimensionality of the input, and we tack on a "1" at the beginning for a bias parameter, w_0. The neuron outputs the probability that x is a member of class C_1.

$$P(x \in C_1|x) = g_w(x) = \frac{1}{1 + \exp(-w^\top x)} \tag{1}$$

$$P(x \in C_2|x) = 1 - P(x \in C_1|x) = 1 - g_w(x), \tag{2}$$

where $g_w(x)$ simply notes that the function g is parameterized by w. Note we identify the output y^n of the "network" for a particular example, x^n, with $g_w(x^n)$, i.e., $y^n = g_w(x^n)$. With the hypothesis function defined, we now use the cross entropy loss function (Equation 3) for two categories over our training examples. This equation measures how well our hypothesis function g does over the N data points,

$$E(w) = -\sum_{n=1}^{N} \{t^n \ln y^n + (1 - t^n) \ln(1 - y^n)\}. \tag{3}$$

Here, t^n is the *target* or *teaching signal* for example n. Our goal is to optimize this cost function via gradient descent. This cost function is minimized at 0 when $t^n = y^n$ for all n. One issue with this cost function is that it depends on the number of training examples. For reporting purposes in this assignment, a more convenient measure is the average error:

$$E(w) = -\frac{1}{N} \sum_{n=1}^{N} \{t^n \ln y^n + (1 - t^n) \ln(1 - y^n)\}. \tag{4}$$

Softmax Regression

Softmax regression is the generalization of logistic regression for multiple (c) classes. Now given an input x^n, softmax regression will output a vector y^n, where each element, y_k^n represents the probability that x^n is in class k.

$$y_k^n = \frac{exp(a_k^n)}{\sum_{k'} exp(a_{k'}^n)} \tag{5}$$

$$a_k^n = w_k^T x^n \tag{6}$$

Here, a_k^n is called the *net input* to output unit y_k. Note each output has its own weight vector w_k. With our model defined, we now define the *cross-entropy* cost function for multiple categories in Equation 7:

$$E = -\sum_n \sum_{k=1}^{c} t_k^n \ln y_k^n \tag{7}$$

Again, taking the average of this over the number of training examples normalizes this error over different training set sizes.

Gradient Descent

Recall the gradient descent iterative algorithm shown in **Algorithm 1**.

Problem (35 points)

1. **Derive the gradient for Logistic Regression.** *(2 points)* We need the gradient of the cost function, Equation 3, with respect to the parameter vector w. Show that for the logistic regression cost function, the gradient is:

Algorithm 1 Batch Gradient Descent

procedure BATCH GRADIENT DESCENT
  $w_0 \leftarrow 0$
  for t = 0, . . . , m **do**
    $w_{t+1} = w_t - \eta \sum_{n=1}^{N} \nabla E^n(w)$
  return w

where η is the step size and $\nabla E^n(w)$ is the gradient of the cost function with respect to the weights w on the n^{th} example.

$$-\frac{\partial E^n(w)}{\partial w_j} = (t^n - y^n)x_j^n \tag{8}$$

Show work.

2. **Derive the gradient for Softmax Regression.** *(3 points)* For softmax regression cost function, Equation 7, show that the gradient is:

$$-\frac{\partial E^n(w)}{\partial w_{jk}} = (t_k^n - y_k^n)x_j^n \tag{9}$$

Show work. **Note:** Here we are departing from Bishop's notation. w_{jk} is the weight from unit j to unit k, not vice-versa.

Hint: Recall your logarithm rules, such as $\ln(\frac{a}{b}) = \ln a - \ln b$. The hardest part here is the derivative of the softmax. Most of the derivations are already done for you in Bishop Chapter 6. You just have to fill in any missing steps.

3. **Read in Data**. Read in the data from the files. Each image is 28×28 pixels, so now our unraveled $x \in \mathbb{R}^{784}$, where each vector component represents the greyscale intensity of that pixel. For each image, append a '1' to the beginning of each x-vector; this represents the input to the bias, which we represent as w_0. There is matlab code for reading in MNIST available from this page. There is one in python here. We do not guarantee that this code will work for you! Again, you may save time (and space) by using the first 20,000 training images and the first 2,000 test images. If you use more, please document this in your report. Note that the first half of these files contain "clean" images, and the second half are by high school students.

4. **Logistic Regression via Gradient Descent.** *(10 points)* Now, using the gradient derived for Logistic Regression cross entropy loss, use gradient descent to classify $x \in \mathcal{R}^{785}$ (there is one extra dimension for the bias term) for two categories: 2's and 3's. The target is 1 if the input is from the "2" category and 0 if it is from the other category. To do this, extract from the dataset the patterns for 2's and 3's. Again, please write your own code for this!

   Experiment with different learning rates and try "annealing" the learning rate by reducing it over time. A possible annealing schedule might be to use the following: $\eta(t) = \eta(0)/(1 + t/T)$, where $\eta(0)$ is an initial learning rate, and T is a new metaparameter. The initial learning rate should be small (e.g., 0.001, for example). Experiment with different initial learning rates to find one that works well for you.

   Early stopping: How do you decide how many iterations of training you need? This is hard to decide in advance, and there is a danger of over-fitting the training set. To avoid this, divide your training set into two parts, a part used to actually change the weights, and a part (usually 10%) as a "hold-out" set. The hold-out set is a stand-in for the unseen test set. As you train on the part used to change the weights, test the network on the hold-out set at the end of every epoch (an "epoch" is one pass through all of the training data, i.e., for each t in Algorithm 1). If the error on the hold-out set begins to go up consistently over, say, 3 epochs, use the weights with the minimum error on the hold-out set as your final answer, and then test on the test set (but see detailed instructions below).

   For your report, please:

   (a) Here, we want you to Plot the loss function (E) over training for the training set and the hold-out set. The loss function should be computed over the entire training set *after* the epoch. I.e., after changing the weights, run every pattern through the network and compute the loss.

Testing on the test set is typically not available in the real world (and is considered "cheating" if the test set *is* available). However, since we aren't in the real world, but are stuck here in academia for the time being, let's check how well the hold-out set actually models the test set by plotting all three on the same plot. *(1.5 points)*

If your classifier learns this task in one or two epochs, take a closer look by plotting these numbers over the entire sets every 1/10th epoch, or even more frequently, so that you can see gradual progress. To do this, you will change the weights after a "minibatch" of 10% of the patterns until you've gone through them all. We're looking for a relatively smooth curve. Does the hold-out set "work" as a good stand-in for the test set? Discuss. *(1 point)*

(b) Plot the percent correct classification over training for the training set, the hold-out set, and the test set, again on the same plot, again, after each epoch. This can be done simultaneously with computing the loss. Count a classification as correct if the input is a "2" pattern, and the output is >= 0.5, and vice-versa for the other pattern. (Again, if your classifier learns everything in one epoch, check progress more frequently). *(1.5 points)*

(c) Repeat the above exercise for 2's versus 8's. *(4 points)*

(d) Display the weights (without the bias) as an image for your two classifiers (2 vs. 3 and 2 vs. 8). Plot the *difference* of the weights between the two classifiers as an image as well. Explain the difference image. Could the difference of the weights be useful in some way? *(2 points)*

5. **Regularization** *(10 points)* One way to improve generalization is to use *regularization*. Regularization is a way to "smooth" the model - to make it ignore idiosyncratic differences between items. It follows William of Ockham's principle, paraphrased as: "The explanation requiring the fewest assumptions is most likely to be correct." In this case, replace "assumption" with "weights" and you get the idea. Regularization is the idea that we should penalize the model for being too complex. It is carried out by a new objective function that includes a term to make the model "smaller" by minimizing the weights in some way:

$$J(w) = E(w) + \lambda C(w) \tag{10}$$

where $C(w)$ is the complexity penalty and λ is the strength of regularization. For $\lambda = 0$, J reduces to E. For infinite (or simply, very large) values of λ, the model ignores the error and only reduces complexity. For the following versions of the complexity penalty, it would reduce all of the weights to 0. There are many possible forms for C, such as L_2 regularization:

$$C(w) = ||w||^2 = \sum_{i,j} w_{i,j}^2, \tag{11}$$

where w is the weight vector of the model, or L_1 regularization:

$$C(w) = |w| = \sum_{i,j} |w_{i,j}|. \tag{12}$$

(a) Derive the update term for logistic regression for gradient descent in J with respect to w, for L_2 and L_1 regularization. All you have to do is figure out $\partial C/\partial w$ in each case. *(2 points)*

(b) Implement these two methods, and train logistic regression for just 2 vs. 3, for several different values of λ, e.g., 0.01, 0.001, 0.0001, using early stopping. Plot the percent correct for different values of λ on the same graph over training. What do you observe? *(2 points)*

(c) For the same points in training as you use to plot the percent correct, plot the length of the weight vector for each λ as a reality check. Discuss. *(2 points)*

(d) Make a plot of final test set error (y axis) versus the log of λ on the x-axis. Discuss. *(2 points)*

(e) Plot the weights in each case as an image. What do you observe? *(2 points)*

6. **Softmax Regression via Gradient Descent.** *(10 points)* Now, using the gradient derived for softmax regression loss, use gradient descent to perform 10-way classification.

(a) Again, use a hold-out set, and use regularization, for the best value of λ and type of regularization ($L_2 vs. L_1$ from the previous experiment. Check a couple of other values of λ to see if you get better results, and use the best results in your report. *(6 points)*

(b) Plot the loss function over the number of training iterations for the training, hold-out, and test set. You don't need to make plots based on different values of λ, just report the best result you get, but document which λ you use. *(2 points)*

(c) Also plot the percent correct over the number of training iterations for the training, hold-out and test set for the same λ. *(2 points)*

7. **What to turn in** *(5 points)*

For the Bishop problems part, each teammate should submit a separate report.

For the programming part, please submit a pdf of your report (again, NIPS format preferred) with

(a) An informative title and list of authors with email addresses. A title more descriptive than "CSE 253 Programming Assignment 1" would be good.

(b) An abstract that provides a short, high-level description of the assignment, and briefly describes what you did with numerical results highlighted at the end (e.g., "Using these methods, we achieved 90% correct on discriminating handwritten digits.").

(c) An introduction to the problem, your methods, results, and discussion (these should be separate sections). Since there are at least two parts, you can repeat these sections for each part. Make sure your discussion includes answers to thought questions posed in the assignment, i.e., where it says "Explain...". Your results sections should include the figures requested, with figure captions.

(d) A final paragraph summarizing the results and what you learned from the assignment.

(e) A section with separate paragraphs stating the contributions of each team member to the project. People who have not pulled their weight have gotten very low scores compared to their teammates in the past. It is best to be clear at the beginning what each teammate is expected to do, and to try to divide the tasks equally. Pair programming is encouraged!

(f) References, if you used them.

Please submit your code on Vocareum.

Nota Bene: Feel free to use momentum, if you know what that is! A typical value of the momentum parameter is 0.9. Document this in your report if you do. The momentum described in the CSE 250a assignment 5 from Fall 2016 is not the same as the one we expect you to use. See page 262, section 7.5.2 (page 282 of the pdf) of Bishop for a description. If you use Nesterov Momentum, document that.