9.641 Neural Networks

Problem Set 9: Delta Rule and Gradient Descent

(Due before class on Thursday, Apr. 28)

1. Download the new face dataset and the file `perceptron_delta.m`.

   The face dataset consists of a training and a test set, both including faces and nonfaces. The second file is sample MATLAB code for online gradient training of a perceptron.

   After every 100 examples, the code draws the weight vector as an image, and plots the learning curve. In general, a learning curve is defined as a graph of some measure of performance as a function of time. Here we plot the mean squared error that the perceptron has made up to that point. One can imagine many other measures of performance, such as the mean squared error only on the last 100 examples, etc.

   (a) The program terminates after 10000 example presentations (of course you can change this number if you like). Experiment with the learning rate η to find the value that minimizes the final level of the learning curve. Submit this value of η and the learning curve that the program produces.

   (b) Describe in words what you observe when the the learning rate is either substantially higher or lower than this optimal value.

   (c) In class we wrote the equations for gradient learning of a perceptron without the bias term. Derive the equations for the case with a bias term. You can check your result by comparing with the MATLAB code.

   (d) **OPTIONAL:** Compute the performance of your network on the test set `test`. To do so, threshold the output of your network and compare it to the true label given in `testlabels`. Submit your performance value and a `.mat` file with your w vector via email. **The three highest performances will be rewarded with 10 extra points!**

2. Download the files `mnistabridged.mat`, an abridged version of the MNIST database containing 5000 training examples and 1000 test examples of the handwritten digits.

   Train a collection of ten simple perceptrons to each detect a different digit class. To do this, you could simply add an outer loop around the `perceptron_delta.m` code for the different digits. But it's more elegant

(and better practice in MATLAB) to define a 10×784 matrix $\mathbf{W}$ that holds the 10 perceptron weight vectors in its rows. Then the squared error function is

$$E(\mathbf{W}) = \frac{1}{2} \sum_{\mu} |\mathbf{y}^{\mu} - f(\mathbf{W}\mathbf{x}^{\mu} + \mathbf{b})|^2 \tag{1}$$

where μ is the index labeling the training samples, The 784×1 vector $\mathbf{x}^{\mu}$ contains the μth image in the training set. The function f has been extended to take a vector argument, simply by applying it to each component of the vector.

The desired output $\mathbf{y}^{\mu}$ is now a 10×1 binary vector. The ith component $y_i^{\mu} = 1$ if the input $\mathbf{x}^{\mu}$ belongs to the ith digit class, while all other components are zero. We'll use the convention that the class of "zero"s corresponds to $i = 10$. The bias $\mathbf{b}$ is also a 10×1 vector.

(a) Write the gradient descent updates for W_{ij} and b_i.

(b) Implement these updates in MATLAB using the logistic function $f(x) = 1/(1 + \exp(-x))$ and some constant learning rate. It may help you to use the sample code in `perceptron_delta.m` as a starting point. Modify the visualization segment of the code so that all ten weight vectors are shown as images. Also add a bar graph that shows the outputs of the ten perceptrons, along with an image of the current input vector. This will help you visualize how well your perceptrons are doing.

Find a good learning rate, and run your code until the learning curve becomes approximately flat. Submit your code, along with images of the final synaptic weights of the network, and learning curves.

3. Gradient Descent on a Quadratic Surface.

Although the cost functions that we attempt to minimize using gradient descent rarely have a simple quadratic form over the entire surface, the surface can often be approximated as quadratic in the vicinity of a minima. For this reason, we can study the convergence of the gradient descent algorithm by understanding its performance on a quadratic surface.

Consider the cost function

$$E = \frac{1}{2} \mathbf{w}^T \mathbf{Q} \mathbf{w},$$

where $\mathbf{Q}$ is symmetric and positive definite, and the gradient descent update $\Delta \mathbf{w} = -\eta \frac{\partial E}{\partial \mathbf{w}}$.

(a) Show that the dynamics of w using the gradient update can be written as $\mathbf{w}(t+1) = (\mathbf{I} - \eta \mathbf{Q})\mathbf{w}(t)$.

(b) What is the largest value of η for which this update is stable?

(c) Assume that we select $\eta = \frac{1}{\lambda_{max}}$, where λ_{max} is the largest eigenvalue of $\mathbf{Q}$. Show that as $t \to \infty$, the convergence rate of the error is largely determined by the ratio $\lambda_{min}/\lambda_{max}$, where λ_{min} is the smallest eigenvalue of $\mathbf{Q}$.

4. Gradient Descent Simulation.

   For these problems, plot the trajectories of $\mathbf{w}$ on top of a 2D contour plot of E. Use the domain $x \in [-1, 1], y \in [-1, 1]$, and the two initial conditions $\mathbf{w}(0) = [-0.6, -0.3]^T$ and $[0.8, 0.5]^T$. Simulate until $E < 0.0001$.

   You will probably use some or all of the following MATLAB functions: *plot*, *meshgrid*, *contour*, *hold* and *line*.

   (a) Simulate gradient descent on the cost function

   $$E = \frac{1}{25}(13w_1^2 + 2w_1w_2 + 13w_2^2)$$

   for various initial conditions on $\mathbf{w}$ using a constant η. What are λ_{max} and λ_{min}? Experiment with η. What value of η requires the fewest steps?

   (b) Now simulate gradient descent on the cost function

   $$E = \frac{1}{25}(13w_1^2 + 24w_1w_2 + 13w_2^2).$$

   What are λ_{max} and λ_{min} for this problem? What value of η works best here?

   (c) Provide a geometrical explanation for the relative convergence rates of the two algorithms.

5. Gradient Descent for Linear Perceptrons.

   Consider a linear perceptron $y = \mathbf{w}^T\mathbf{x}$, and the cost function

   $$E = \frac{1}{2}\sum_{\mu}(y^\mu - \mathbf{w}^T\mathbf{x}^\mu)^2, \tag{2}$$

   where the vectors x^μ contain the inputs in the training and/or test data and the scalars y^μ are the corresponding outputs.

   (a) Show that minimizing this cost function is equivalent to minimizing a cost function of the form

   $$E = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} - \mathbf{b}^T\mathbf{w}.$$

   Give $\mathbf{A}$ and $\mathbf{b}$. What is the gradient descent update rule $\Delta\mathbf{w}$ on this cost function?

   (b) Using a change of variables, show that the gradient update in part (a) is equivalent to the gradient update for the cost function

   $$E = \frac{1}{2}\tilde{\mathbf{w}}^T\mathbf{A}\tilde{\mathbf{w}}.$$

   Give $\tilde{\mathbf{w}}$? Based on problem 2, what can we deduce about the convergence rate of E in equation 2?