MAT 345 - PROJECT #5

due Wednesday, December 12, 2018 at 12:00PM.

OBJECTIVE: In this project, you will implement a neural network.

GRADING: The assignment is worth 5% of your course grade.

INSTRUCTIONS: Students will work individually on this project, but they may ask questions and clarification from classmates and the instructor. Students must submit their projects on Moodle.

SUBMIT THE FOLLOWING: A copy of your code and a report.

PROJECT: In this project, you will build a neural network for digit recognition. Download the MNIST data set from http://yann.lecun.com/exdb/mnist/. There should be four files: the *training set* contains 60000 examples, and the *test set* contains 10000 examples.

0. Map output values into vectors. For example, if $y = 3$, let $\mathbf{y} = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]^T$, with a 1 in the 3rd row of a 10-dimensional vector. Note that $y = 0$ is mapped to $\mathbf{y} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]^T$, with 1 in the 10th row.

I. Set up the network architecture: we will use 3 layers

   (a) The input layer will have $28 \times 28 = 784$ units since we are using images of size 28x28

   (b) The output layer will have 10 units (1 for digit 1, $\cdots$, 9 for digit 9, 10 for digit 0)

   (c) The hidden layer will have $s_2$ units. Your code should work for different values of $s_2$.

II. Train the neural network using $\mathcal{D}_1$ = *training set*:

   Step 1. Implement the activation function, for which we use the sigmoid $\theta(x)$ discussed in class

$$\theta(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}.$$

   Recall that implementation for a vector $\theta(\mathbf{z})$ is done coordinate-wise.

   Step 2. Initialize the weights in $W^{(1)}$ and $W^{(2)}$ with random values in $[-\epsilon, \epsilon]$, for some small $\epsilon$.

   Note that the dimensions of the weight matrices are:

$$\begin{array}{cc} W^{(1)} & W^{(2)} \\ s_2 \times 785 & 10 \times (s_2 + 1) \end{array}$$

Step 3. For each data point $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_1$:

(1). Implement the **Feed Forward** algorithm:

(a) Let $\mathbf{a}^{(1)} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$, that is, add the bias term $\mathbf{a}_0^{(1)} = 1$.

(b) Let $\mathbf{z}^{(2)} = W^{(1)}\mathbf{a}^{(1)}$ (it has dimension $s_2 \times 1$)

(c) Let $\mathbf{a}^{(2)} = \theta(\mathbf{z}^{(2)})$ and add $\mathbf{a}_0^{(2)} = 1$, the bias term

(d) Let $\mathbf{z}^{(3)} = W^{(2)}\mathbf{a}^{(2)}$ (it has dimension $10 \times 1$)

(e) Let $\mathbf{a}^{(3)} = \theta(\mathbf{z}^{(3)})$

(2). Implement the **Back Propagation** algorithm

(a) Let $\delta^{(3)} = \mathbf{a}^{(3)} - \mathbf{y}$ (it has dimension $10 \times 1$)

(b) Let $\tilde{W}^{(2)}$ be the matrix $W^{(2)}$ with the column of ones removed, so it has size $10 \times s_2$.

Let $\delta^{(2)} = [\tilde{W}^{(2)}]^T \delta^{(3)} \odot \theta'(\mathbf{z}^{(2)})$ (it has dimension $s_2 \times 1$).

(c) Compute the set of partial derivatives with respect to the weights, for $k = 1, 2$:

$$\begin{aligned} gW^{(1)}(\mathbf{x}, \mathbf{y}) &= \delta^{(2)}[\mathbf{a}^{(1)}]^T \quad &\rightarrow \quad \text{size } s_2 \times 785. \\ gW^{(2)}(\mathbf{x}, \mathbf{y}) &= \delta^{(3)}[\mathbf{a}^{(2)}]^T \quad &\rightarrow \quad \text{size } 10 \times (s_2 + 1). \end{aligned}$$

Notes: Recall that $\odot$ stands for coordinate-wise multiplication, i.e, if $\mathbf{u} = [u_1, u_2, u_3]$ and $\mathbf{v} = [v_1, v_2, v_3]$, then

$$\mathbf{u} \odot \mathbf{v} = [u_1 v_1, u_2 v_2, u_3 v_3]$$

Also, we have derived in class that

$$\theta'(\mathbf{z}^{(k)}) = \mathbf{a}^{(k)} \odot (\mathbf{1} - \mathbf{a}^{(k)}).$$

Step 4. Compute the gradients for the *training set* $\mathcal{D}_1$: for $k = 1, 2$

$$\text{gradW}^{(k)} = \frac{1}{|\mathcal{D}_1|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_1} gW^{(k)}(\mathbf{x}, \mathbf{y}).$$

Step 5. Run the **Gradient Descent** algorithm to find the weights that minimize the cost function.

(a) Weights were initialized in Step 2.

(b) Choose $\eta$.

(c) Update rule: for $k = 1, 2$

$$W_{t+1}^{(k)} = W_t^{(k)} - \eta \cdot \text{gradW}_t^{(k)}$$

Note that the matrices and gradients can be "unrolled" as discussed in class, so one can work with vectors instead of matrices, but that is not necessary.

III. Test the network using $\mathcal{D}_2 =$ *test set*: compute the accuracy for this network

   (a) Use the Feed Forward part of your program to predict the output for data points in $\mathcal{D}_2 =$ *test set*. Make sure you predict by choosing the label with the highest activation value of at least 0.5.

   (b) Count how many data points from $\mathcal{D}_2$ are accurately predicted.

IV. In your Report, include:

   (a) Your name

   (b) The programming language you used for the project

   (c) A discussion on which values of $\eta$ lead to a reasonable performance in the gradient descent algorithm (for example, try values between 0.01 and 5.)

   (d) Train the network with $s_2 = 30$ and output the resulting weights $W^{(1)}$, $W^{(2)}$. Test your network and output the accuracy.

   (e) Try different number of units in the hidden layer $s_2$ and output the accuracy rate for each case (try a variety of sizes for $s_2$, such that $30, 100, 300$ etc.) Do more units in the hidden layer lead to better accuracy?

   (f) Include any additional information, such as if you are using regularization, if you are using stochastic gradient descent rather than gradient descent, if you are doing a gradient check for back propagation, etc.