EE 526X Deep Machine Learning: Theory and Practice — Homework 2
Assigned: 10/01. Due: 10/09

Problem 1. Prove that the softplus function

$$f(z) = \log(1 + e^z) \tag{1}$$

is convex in z by showing that its second derivative is positive for all z.

Problem 2. Let $\mathbf{z}$ denote a vector $\mathbf{z} = [z_1, z_2, \ldots, z_n]^T$. Let $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$, such that

$$p_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}. \tag{2}$$

That is $\mathbf{p}$ is the output when the softmax function is applied to $\mathbf{z}$. Derive the Jacobian matrix

$$\frac{\partial \mathbf{p}}{\partial \mathbf{z}} = \left[\frac{\partial p_j}{\partial z_i}\right]_{i,j=1,1}^{n,n}. \tag{3}$$

Problem 3. Let $\mathbf{z}$ denote a "logit" vector $\mathbf{z} = [z_1, z_2, \ldots, z_n]^T$. Let $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$, such that

$$p_i = \frac{e^{z_i}}{\sum_{i=1}^{n} e^{z_i}}. \tag{4}$$

Let $\mathbf{y}$ denote a probability vector $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ such that $y_i \geq 0, \forall i \in [1, n]$, and $\sum_{i=1}^{n} y_i = 1$. Let J denote the cross entropy between $\mathbf{p}$ and $\mathbf{y}$:

$$J(\mathbf{z}) = -\sum_{i=1}^{n} y_i \log p_i, \tag{5}$$

where log is natural logarithm.

Derive the gradient vector

$$\frac{\partial J}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial J}{\partial z_1} \\ \vdots \\ \frac{\partial J}{\partial z_n} \end{bmatrix} \tag{6}$$

Hint: You can either use the Jacobian, or directly take the gradient. If you take the derivative directly, without using the Jacobian matrix, then it is helpful to write $\log(p_i) = z_i - \log\left(\sum_{i=1}^{n} e^{z_i}\right)$.

Problem 4. Let a neural network be such that it has two neurons in one single layer. The neurons has two common inputs. The model can be described as

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{7}$$

We are given two training points:

(a) When $\mathbf{x} = [1, 0]^T$, $\mathbf{y} = [1, 0]^T$.

(b) When $\mathbf{x} = [0, 1]^T$, $\mathbf{y} = [0, 1]^T$.

Note that $\mathbf{y}$, the output has been one-hot encoded. Let $\mathbf{X}$ and $\mathbf{Y}$ both be the 2×2 identity matrix, denoting the input and output for the training data. Use softmax on $\mathbf{z}$ and use cross-entropy as the cost function, run the forward and backward propagation *by hand calculation* to update $\mathbf{W}$ and $\mathbf{b}$ for two iterations, with the following conditions:

(a) Both initialized to all zeros.

(b) Learning rate is $\eta = 1$.

(c) Use gradient descent.

That is, make two updates of $\mathbf{W}$ and $\mathbf{b}$ by hand calculation. You need to show the intermediate steps (values of $\mathbf{z}$, $d\mathbf{z}$, $\mathbf{p}$, $d\mathbf{W}$, $d\mathbf{b}$, etc).

Problem 5. Design a three layer neural network that has the following specifications:

(a) Input dimensions: 57 (all real numbers);

(b) Output dimension: 1 (binary);

(c) Layer 1: d_1 neurons, with ReLU (rectified Linear Unit) nonlinearity;

(d) Layer 2: d_2 neurons, with ReLU nonlinearity;

(e) Output Layer: 1 neuron, with logistic nonlinearity;

(f) The objective function is cross-entropy (logistic regression).

Use the same training and test data from the Spambase Data Set. Train the neural network using forward and backward propagation and gradient descent.

You can use the `DNN.py` program available on Canvas. You need to submit the source program, in Python. Also the results on the learning rate(s) used, and training and test errors, and running time should be reported.

Problem 6.

Using the code available on canvas `DNN.py`, experiment with doing classification with the MNIST data set, using the following settings:

(a) Single layer, 10 neurons, softmax + cross entropy objective function.

(b) Three layers, [(50, ReLU), (50, ReLU), (10, Linear)], softmax + cross entropy objective function.

(c) Experiment with other neuron settings, with no more than 3 layers, and no more than 150 neurons in total.

Submit your final codes. Report both training and testing errors for the different settings.

END OF ASSIGNMENT