Homework 2
600.482/682 Deep Learning
Spring 2019

March 1, 2019

Due 2019 Mon. 02/25 11:59pm.
Please submit a latex generated PDF and python jupyter notebook to Gradescope with entry code MYRR74

1. Given the following function,

$$f(x_1, x_2, w_1, w_2) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}} + 0.5(w_1^2 + w_2^2)$$

(a) Please calculate $\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}$.

Ans:

$\frac{\partial f}{\partial w_1} = (1 - \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}}) * \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}} * x_1 + w_1$

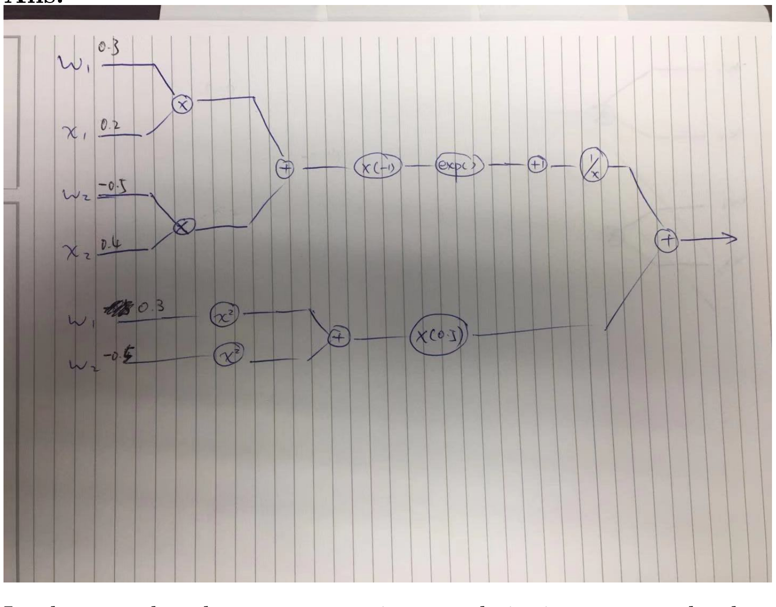
$\frac{\partial f}{\partial w_2} = (1 - \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}}) * \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}} * x_2 + w_2$

$\frac{\partial f}{\partial x_1} = (1 - \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}}) * \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}} * w_1$

$\frac{\partial f}{\partial x_2} = (1 - \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}}) * \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2)}} * w_2$

(b) Start with the following initialization: $w_1 = 0.3, w_2 = -0.5, x_1 = 0.2, x_2 = 0.4$, draw the computation graph. Please use the back propagation node as what we have done in class. (Insert your result as a figure in latex submission. Do NOT need to include the intermediate values during the forward and backward pass.)

Ans:



(c) Implement the above computation graph in jupyter notebook using numpy. Please initialize the weights and use the input scalar value as defined in (b). Fix the step size as 0.01. Plot the weight value w_1 and w_2 trend for 30 iterations in a single figure.

2. Your goal in this exercise is to implement a linear classifier and to submit the results of training on the given datasets. (Hint: You can derive a training method as $f(x, W) = Wx$, pass through a softmax and use Kullback-Leibler divergence as loss function.)
TODO: Write a python program that iteratively computes a small step in the direction of negative gradient. Apply your code to data that you can download here (`https://piazza.com/class_profile/get_resource/jr28pso2atzom/js1984xa8pr1be`). **Please go through the README.txt carefully before you do experiments.** Start your optimization at the origin ($W = 0$). If your algorithm converges to a correct solution that can linearly separate the given data samples, submit that solution; if it does not, indicate where it did not converge or whether it could not separate the data set.

3. In this problem, we want to compare the performance of single-layer and multi-layer classifiers.

   (a) Please download the dataset here (`https://piazza.com/class_profile/get_resource/jr28pso2atzom/js1984xa8pr1be`). Use the same model and test on this dataset. Does it classify as well?

   (b) Please add more layer perceptron (1-2 more).* Add activation functions after each layer (you can use sigmoid or ReLU). Does it give you better performance?

   (c) Please visualize your model's prediction and the groundtruth label as separate figures in (a) and (b). Can you explain the difference of these two models?

   * If for a single layer, the output is $f(x, W)$, then adding another layer means $g(f(x, W))$, which connects the output of the former layer as input to the next layer.

4. Now, we want to test what we learned on more complex dataset. Please download MNIST dataset (`http://yann.lecun.com/exdb/mnist/`), which has been divided into training and test dataset. Please compute an average "image" on training data set, (an average matrix of all training inputs). Use this as rows in a weight matrix to implement linear classifier. Add softmax to the output.

   (a) Visualize the "templates" and validate on test data. What is the test accuracy?

   (b) Think about and implement a simple feature extractor* and apply it to your image. Use the extracted features to train a classifier as in Question (a) to classify the data. Does this perform better or worse than (a)? Why?

   (c) Generate your own handwritten image, fix the size the same as MNIST, and test your model to your image. Does it work? Describe the performance.

   * If you cannot think of something, try using "Histogram of gradients". E.g. at every point in the image, filter with gaussian model using $\sigma = 2.0$, compute the gradient in (x, y) direction, then construct a histogram of gradient orientations with n bins. Rather than just counting numbers, try accumulating the gradient magnitude ($\sqrt{g_x^2 + g_y^2}$) to make it more robust to noise. (python script of gaussion filter can refer to `https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html`; gradient image calculation can refer to `https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.gradient.html`, and please pay attention to the gradient direction (axiss) when using numpy.)