MATH 156S HOMEWORK 6

Due May. 13

Exercise 1 (Derivative of softmax-cross-entropy)**.** The softmax function $\sigma : \mathbb{R}^n \to \mathbb{R}^n$, is defined by

$$\mathbf{x} = [x_1, \ldots, x_n]^T \mapsto \sigma(\mathbf{x}) = [\sigma_1(\mathbf{x}), \ldots, \sigma(\mathbf{x}_n)] = \left[\frac{\exp(x_1)}{\sum_{i=1}^n \exp(x_i)}, \ldots, \frac{\exp(x_n)}{\sum_{i=1}^n \exp(x_i)} \right]^T. \tag{1}$$

Define the following softmax-cross-entropy loss

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) := -\sum_{i=1}^{\kappa} \mathbf{1}(\mathbf{y} = \text{class } i) \log \sigma_i(\hat{\mathbf{y}}) \tag{2}$$

for each $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^n$.

(i) Show that

$$\frac{\partial}{\partial x_j} \sigma_i(\mathbf{x}) = \sigma_i(\mathbf{x}) \left(\mathbf{1}(i = j) - \sigma_j(\mathbf{x}) \right). \tag{3}$$

(ii) Show that

$$\frac{\partial}{\partial \hat{\mathbf{y}}} \ell(\mathbf{y}, \hat{\mathbf{y}}) = \sigma(\hat{\mathbf{y}}) - \mathbf{y}. \tag{4}$$

Exercise 2. Formulate an algorithm that computes the gradient of a general L-layer neural network. You may need to extend the error backpropagation algorithm for 2-layer NN we discussed above to the general L-layer case and show its correctness.

Exercise 3 (2-layer FFNN on MNIST)**.** Below we show the train and test accuracy of classifying MNIST dataset (digits 0 through 4) using the 2-layer FFNN discussed in Example 3.3.1 in LN. Here the accuracy for multiclass classification is computed by the total number of correctly classified examples divided by the total number of examples. (Ref: Jupyter Notebook)

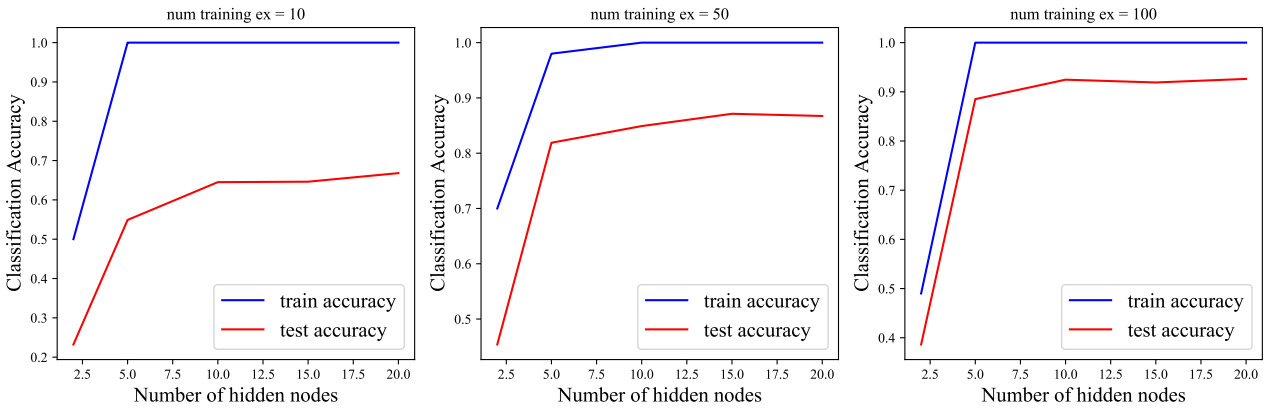

FIGURE 1. Classification accuracy of 2-layer FFNN for MNIST data with digits from 0 up to 4. The number of hidden nodes are varied within $\{2, 5, 10, 15, 20\}$ and the number of training examples are varied within $\{10, 50, 100\}$.

(i) What can you conclude from Figure 1? Optimal number of hidden nodes for each size of training set? Overfitting in terms of the number of hidden nodes and the size of training set?

(ii) Make similar plots for (1) digits 5 through 9 and (2) all digits while keeping other parameters the same. (You may increase the number of iterations for training for classifing all digits.) Do your observations in (ii) still hold in these experiments?

Exercise 4. Consider the following L_2-regularized 2-layer neural network training problem:

$$\hat{\mathbf{w}} := \operatorname{argmin}_{\mathbf{w}=[\mathbf{W}^{(1)},\mathbf{W}^{(2)}]} \left[\ell_{NN}(\mathbf{w};\lambda) := \left(\sum_{s=1}^{n} \ell(\mathbf{y}_s, \hat{\mathbf{y}}(\mathbf{x}_s;\mathbf{w})) \right) + \lambda \left(\|\mathbf{W}^{(1)}\|_F^2 + \|\mathbf{W}^{(2)}\|_F^2 \right) \right], \tag{5}$$

where $\lambda \geq 0$ is a L_2-regularization constant. This will penalize the weight matrices $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ to have large L_2-norm.

(i) How does this change the training algorithm? Derive the gradient of the above loss function and formulate the modified training algorithm.

(ii) Modify the Multilayer Perceptron implementation `MLP` in Jupyter Notebook so that it has the option to use L_2-regularization. Reproduce Figure 1 with varing levels of regularization. Report a reaonable value of the regularization constant for the experiment in Figure 1.