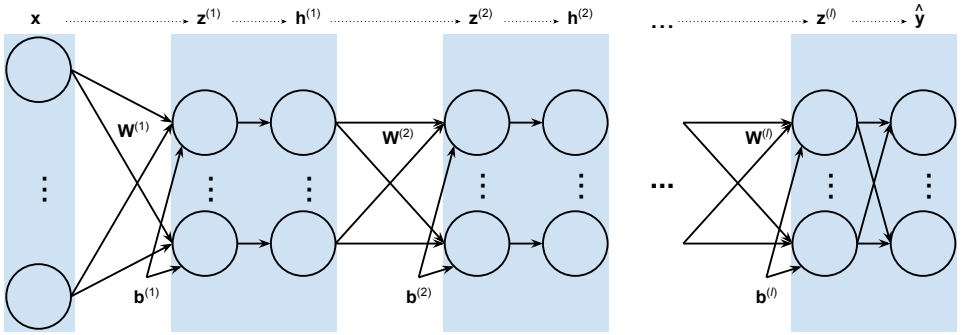
Homework 4 – Deep Neural Networks (CSDS/541, Whitehill, Spring 2019)

You may complete this homework assignment either individually or in teams up to 2 people.

1. **Feed-forward neural network** [40 points]: In this problem you will train a multi-layer neural network to classify images of hand-written digits from the MNIST dataset. Similarly to Homework 3, the input to the network will be a 28×28-pixel image; the output will be a real number. Specifically, the network you create should implement a function $f : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$, where:

$$\begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{h}^{(1)} &= \text{relu}(\mathbf{z}^{(1)}) \\ \mathbf{z}^{(2)} &= \mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)} \\ \vdots & \quad \vdots \\ \mathbf{z}^{(l)} &= \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \\ \hat{\mathbf{y}} &= \text{softmax}(\mathbf{z}^{(l)}) \end{aligned}$$

The network specified above is shown in the figure below:



As usual, the (unregularized) cross-entropy cost function should be

$$f_{\text{CE}}(\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{10} y_k^{(i)} \log \hat{y}_k^{(i)}$$

where m is the number of examples.

Hyperparameter tuning: In this problem, there are several different hyperparameters that will impact the network's performance:

- Number of hidden layers (suggestions: $\{3, 4, 5\}$)
- Number of units in each hidden layer (suggestions: $\{30, 40, 50\}$)
- Learning rate (suggestions: $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$)
- Minibatch size (suggestions: $16, 32, 64, 128, 256$)
- Number of epochs
- L_2 Regularization strength applied to the weight matrices (but not bias terms)

In order not to "cheat" – and thus overestimate the performance of the network – it is crucial to optimize the hyperparameters **only** on the **validation** set; do **not** use the test set. (The training set would be ok but typically leads to worse performance.)

Your task: Use stochastic gradient descent to minimize the cross-entropy with respect to all the weight matrices and bias vectors. Specifically:

(a) Implement stochastic gradient descent (SGD; see Section 5.9 and Algorithm 6.4 in the *Deep Learning* textbook) for the multi-layer neural network shown above. **Important**: your backprop algorithm must work for *any* number of hidden layers (not just 1 hidden layer). [25 points]

(b) Optimize the hyperparameters by training on the **training** set and selecting the parameter settings that optimize performance on the **validation** set. **You should *systematically* (i.e., in code) try at least 10 (in total, not for each hyperparameter) different hyperparameter settings**; accordingly, make sure there is a method called `findBestHyperparameters` (and please name it as such to help us during grading) [10 points]. **Include a screenshot** showing the progress and final output (selected hyperparameter values) of your hyperparameter optimization.

(c) After you have optimized your hyperparameters, then run your trained network on the **test set** and report the accuracy (percent correctly classified images). **Include a screenshot** showing both these values during the last 20 epochs of SGD. **The accuracy (percentage correctly classified test images) should be at least** 96.5%**.** [5 points]

In addition to your Python code (`homework4_WPIUSERNAME1.py` or `homework4_WPIUSERNAME1_WPIUSERNAME2.py` for teams), create a PDF file (`homework4_WPIUSERNAME1.pdf` or `homework4_WPIUSERNAME1_WPIUSERNAME2.pdf` for teams) containing the screenshots described above.