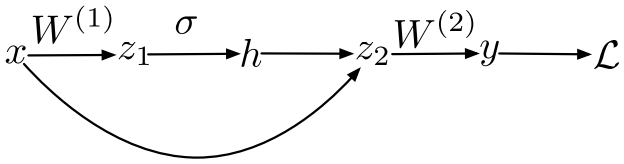
HOMEWORK 3

CSC2515 Fall 2021

- **Deadline:** Friday, November 26, 2021 at **16:59**.
- **Submission:** You need to submit two files through MarkUs. One is a PDF file including all your answers and plots. The other is a source file (Python script or Jupyter Notebook) that reproduces your answers. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc) as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. If the code does not run, you may lose most/all of your points for that question.
- **Late Submission:** 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.
- **Collaboration:** You can discuss the assignment with up to two other students (group of three). You can work on the code together. But each of you need to **write your homework report individually**. You must mention the name of your collaborators clearly in the report and the source code.

1. Backpropagation – 20pts.

The goal of this exercise is to help you practice how backpropagation works. We consider a simple variation of the feedforward fully-connected network. In the usual feedforward fully-connected network, each layer is connected to its previous layer. The main difference here is that one of the hidden layers in this network is connected to the input too. The computation graph and how each computation is performed is as follows:



$$
\begin{aligned}
z_1 &= W^{(1)}x \quad \text{with } x \in \mathbb{R}^d \\
h &= \sigma(z_1) \quad \text{with } h \in \mathbb{R}^d \\
z_2 &= h + x \\
y &= W^{(2)}z_2 \\
\mathcal{L} &= \frac{1}{2}(y-t)^2 \text{ with } t \in \mathbb{R}.
\end{aligned}
$$

Here σ is the activation function, and you can assume that it is differentiable. Answer the following questions:

(a) **[4pt]** Determine the dimensions of $W^{(1)}$, $W^{(2)}$, z_1, and z_2.

(b) **[2pt]** Calculate the number of parameters in this network, as a function of d.

(c) **[14pt]** Compute the gradient of loss $\mathcal{L}$ with respect to all variables. That is, compute

- $\bar{y} = \frac{\partial \mathcal{L}}{\partial y} = \dots$
- $\bar{W}^{(2)} = \frac{\partial \mathcal{L}}{\partial W^{(2)}} = \dots$
- $\bar{z}_2 = \dots$
- $\bar{h} = \dots$
- $\bar{z}_1 = \dots$
- $\bar{W}^{(1)} = \dots$
- $\bar{x} = \dots$

2. Multi-Class Logistic Regression – 10pts.

The goal of this exercise is to verify the formula on Slide 91 of Lecture 3. Consider

$$\begin{aligned}\mathbf{z} &= W\mathbf{x}\\ \mathbf{y} &= \mathrm{softmax}(\mathbf{z})\\ \mathcal{L}_{\mathrm{CE}}(\mathbf{t},\mathbf{y}) &= -\mathbf{t}^\top \log \mathbf{y} = -\sum_{k=1}^{K} t_k \log y_k\end{aligned}$$

Note that if $x \in \mathbb{R}^d$, the dimension of W is $K \times d$. We denote its k-th row by $\mathbf{w}_k$. The vector $\mathbf{y}$ is a function of W and x. And the output $\mathbf{t}$ is a one-hot encoding of the output.

Recall that the k-th component of $\mathbf{y}$ is

$$y_k = \mathrm{softmax}(z_1, \ldots, z_K)_k = \frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z_{k'})}.$$

(a) **[5pt]** Compute

$$\frac{\partial y_k}{\partial z_{k'}},$$

for any $k, k' = 1, \ldots, K$ (note that k and k' may or may not be the same). Try to write it in a compact form (no $\exp(\cdots)$ would be needed).

(b) **[5pt]** Compute

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(\mathbf{t}, \mathbf{y}(\mathbf{x}; W))}{\partial \mathbf{w}_k}.$$

You need to show all the derivations in order to get the full mark. The final solution alone will not give you any mark, as it is already shown on the slide.

3. Handwritten Digit Classification – 70 points.

For this question you will build classifiers to label images of handwritten digits. Each image is 8 by 8 pixels and is represented as a vector of dimension 64 by listing all the pixel values in raster scan order. The images are grayscale and the pixel values are between 0 and 1. The labels y are $0, 1, 2, \cdots, 9$ corresponding to which character was written in the image. There are 700 training cases and 400 test cases for each digit; they can be found in a3digits.zip.

Starter code written in Python is provided to help you load the data. A skeleton is also provided for each question that you should use to format your solution. You are welcome to use any other programming language, as long as your code is functional and modular.

Please note that if you are asked to report/compute quantities these should be clearly displayed in your written report. It is not sufficient to simply print these as an output of your code. The same applies to plots and figures.

0. **[3pt]** Load the data and plot the means for each of the digit classes in the training data (include these in your report). Given that each image is a vector of size 64, the mean will be a vector of size 64 which needs to be reshaped as an 8×8 2D array to be rendered as an image. Plot all 10 means side by side using the same scale.

3.1. *K-NN Classifier – 12pt.*

1. **[6pt]** Build a simple K nearest neighbour classifier using Euclidean distance on the raw pixel data.
   (a) For $K = 1$ report the train and test classification accuracy.
   (b) For $K = 15$ report the train and test classification accuracy.
2. **[1pt]** For $K > 1$, K-NN might encounter ties that need to be broken in order to make a decision. Choose any (reasonable) method you prefer and explain it briefly in your report.
3. **[5pt]** Use 10 fold cross validation to find the optimal K in the 1-15 range. You may use the KFold implementation in sklearn. Report this value of K along with the train, validation, and test set classification accuracies, averaged across folds where applicable.

3.2. *Classifiers comparison – 30pt.* In this section, you will design three different classifiers for the provided hand-written digits data set. You are free to implement your own classifier or to use any package you prefer as long as you will provide readable modular code. We recommend exploring well-known packages such as PyTorch, TensorFlow and scikit-learn. Here is the list of classifiers you are to implement

1. **MLP - Neural Network Classifier – 10pt.** Design a Multi-Layer Perceptron Neural Network. We are asking for a fully connected network with one-hot encoding output. Your input layer needs one unit for each pixel; given that your are distinguishing among 10 classes, the output layer will need 10 units. Other than the input and the output layers, you are free to deign the best network that provide the least possible error rate, taking overfitting into consideration.

2. **SVM classifier – 10pt.** Design an SVM classifier. We briefly covered a few kernels in the class. Since you are using external packages, you have a pretty good chance to try kernel beyond what described. Also, there are useful grid search tools that helps you reach the optimal set of hyper-parameters
3. **AdaBoost Classifier – 10pt.** You will have a chance to turn a weak-learner into a strong performing classifier. Again, you have total freedom of the architecture as long as you provide original, readable and modular code.

3.3. *Model Comparison – 25pt.* Briefly summarize the performance of each model, including the K-NN model with the optimal K you found. This will be a good chance to study different ways to measuring a classifier performance. Measuring MSE or error rate is not enough, you need to provide, at least, the following metrics for each classifier:

1. ROC (Receiver Operating Characteristics) curve
2. Confusion Matrix
3. Accuracy
4. Precision
5. Recall

While we did not cover all these metrics in class, it is a very good idea to read and understand what each one measures. A good starting point is perhaps this Wikipedia link Receiver operating characteristic.
Which classifier performed best? Which performed worst? Did this match your expectations? The most important is to show good understanding of why a certain classifier did better in the provided dataset.