COMP.4220 - Machine Learning

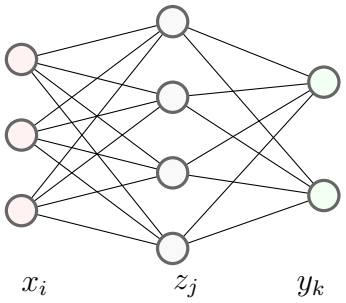
Homework 5

student name:

---

1. Consider the following feed-forward neural network with three inputs (including the bias term), one hidden layer, and two outputs. Assume that the output units have linear activation functions, so that $y_k = a_k$, while the hidden units have sigmoidal activation functions given by $h(a) = \tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$. We also consider a standard sum-of-squares error function, so that for pattern (i.e., data point) n the error is given by $E_n = \frac{1}{2}\sum_{k=1}^{K}(y_k - t_k)^2$, where y_k is the activation of the output unit k, and t_k is the corresponding target, for a particular input patter x_n.

   ⇒**Note:** This question is meant to be solved on paper for a better understanding of the forward and backward propagation. Hence, for all the calculations in this question, showing values rounded to 1 decimal digit would be sufficinet.



   (a) Calculate the network output by performing forward-propagation with the following inputs and weight values:

$$\mathbf{x} = [1, -2, -1]^\top$$

$$W^{(1)} = \begin{bmatrix} 0.5 & -1 & 1.5 \\ -0.5 & -1 & 1 \\ 1 & -0.5 & 0 \\ -2 & 1 & 1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 1 & -2 & -0.5 & 0 \\ 0.5 & 0.5 & 1 & -1 \end{bmatrix}$$

$$\mathbf{y} = ?$$

(b) Now that you calculated the output $\mathbf{y}$ which is the prediction of the network for the given input, perform a backpropagation pass. Calculate δ_k and δ_j, knowing that for the given training input, the target value is $\mathbf{t} = [-1, 1]$.

(c) Finally, use the results from previous parts to calculate the gradients for the first and second layer:

$$\frac{\partial E}{\partial w_{ji}^{(1)}}, \frac{\partial E}{\partial w_{kj}^{(2)}}$$

(12 marks)

2. (Programming) In this section, you build a model for detecting and classifying hand-written images of digits. For this part, we make use of the existing data set "digits". This data set contains images of hand-written digits in 10 classes where each class refers to a digit. With 1797 samples, this data set was collected from 43 people, 30 of whom contributed to the training set and 13 others contributed to the test set. Each sample is an 8×8 image of pixels in range $[0, 16]$.

   ⇒**Note:** You are only allowed to use `numpy`, `sklearn`, and `matplotlib` modules for this question.

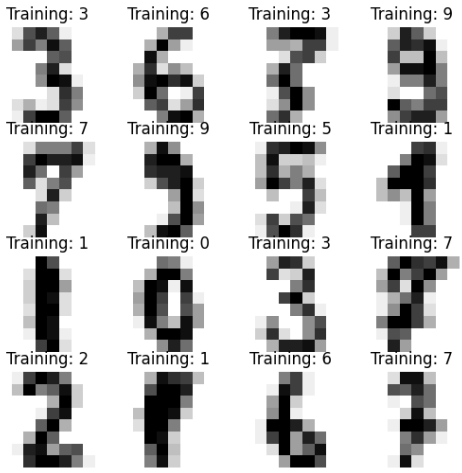
   (a) Import the data set using the following line and check the details of the data set by printing the description using the `load_digits()` function.

```
from sklearn import datasets

```

   Verify the number of samples and shape of each image.

   (b) Generate a plot with 4×4 subplots. Each of the subplots should show a random image from the data set. The title for each image should indicate the class to which this sample belongs. Make sure you remove the xy axes for each subplot. This can be done using the `set_axis_off` in `matplotlib`. Your plot should look like this:



   (c) In order to pass each sample which is an 8×8 image to our neural network, we should first transform it to a 1D array of size 64. Apply this operation to all the images in the data set and verify that the transformed data set has correct shape ($n_{\text{samples}}, n_{\text{features}} = 64$). Note that This transformation should not require a `for` loop if you do it the right (i.e., efficient) way.

(d) Divide the data set into the training and test sets using the `train_test_split` function from the `model_selection` sub-module in `sklearn`. Use a 70%-30% proportion to get $X_{\text{train}}, t_{\text{train}}, X_{\text{test}}, t_{\text{test}}$. To perform consistent testing avoid shuffling the data while splitting. When completed, verify the size of each set.

(e) Instantiate a neural network class from `sklearn` to perform classification. To do this you need to use the `MLPClassifier` with proper initialization inputs. Then fit the model to the training set.

(f) After training is complete, use the trained model on the test set to get predictions y_{pred}. Then to compare model predictions with the targets from the test set, plot a *Confusion Matrix*. A confusion matrix, generated using `confusion_matrix` from the `metrics` sub-module, is a table that visualizes the performance of a classification model by comparing predicted and actual values. It is structured such that rows represent actual classes and columns represent predicted classes. By looking at each row and the corresponding column, you should be able to understand the behavior of your model and the quality of its predictions. A perfect model would have maximum score on the diagonal of the matrix.

(g) Generate another plot with 4×4 subplots. Each of the subplots should show a random image from the test set. The title for each image should indicate model's prediction (y_{pred}). By comparing the predictions against the corresponding target values determine whether the prediction is correct or not, and set the color of the image title accordingly. Use 'blue' and 'red' for correct and incorrect predictions, respectively. Make sure you remove the xy axes for each subplot.

(h) Tune the parameters of the network and explain your tuning process. Which parameters did you choose to tune? How did you tune them? Why did you select a certain option for each of the arguments? Note that without increasing the training time significantly, you should be able to get very good results.

(i) (Extra Credit) By considering conditions such as required training time, overfitting, size of the network, extra credit will be given to models that can perform better than others. You are encouraged to explain why your model should be selected to receive extra credits.

Tips: refer to the `Scikit-learn` documentation to learn about the functions and their features. (13 marks)