EE2211 Tutorial 12

Question 1: The convolutional neural network is particularly useful for applications related to image and text processing due to its dense connections.
a) True
b) False

Question 2: In neural networks, nonlinear activation functions such as sigmoid, and ReLU
a) speed up the gradient calculation in backpropagation, as compared to linear units
b) are applied only to the output units
c) help to introduce non-linearity into the model
d) always output values between 0 and 1

Question 3: A fully connected network of 2 layers has been constructed as

$$\boldsymbol{F}_{\mathbf{w}}(\mathbf{X}) = f(f(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2)$$

where $\mathbf{X} = \begin{bmatrix} 1 & 1 & 3.0 \\ 1 & 2 & 2.5 \end{bmatrix}$, $\mathbf{W}_1 = \mathbf{W}_2 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$.

Suppose the Rectified Linear Unit (ReLU) has been used as the activation function (f) for all the nodes. Compute the network output matrix $\boldsymbol{F}_{\mathbf{w}}(\mathbf{X})$ (up to 1 decimal place for each entry) based on the given network weights and data.

$$\boldsymbol{F}_{\mathbf{w}}(\mathbf{X}) = \begin{bmatrix} blank1 & blank2 & blank3 \\ blank4 & blank5 & blank6 \end{bmatrix}$$

Question 4: A fully connected network of 3 layers has been constructed as

$$\boldsymbol{F}_{\mathbf{w}}(\mathbf{X}) = f([\mathbf{1}, f([\mathbf{1}, f(\mathbf{X}\mathbf{W}_1)]\mathbf{W}_2)]\mathbf{W}_3)$$

where $\mathbf{X} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 5 & 1 \end{bmatrix}$, $\mathbf{W}_1 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix}$, $\mathbf{W}_2 = \mathbf{W}_3 \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 1 \end{bmatrix}$.

Suppose the **Sigmoid** has been used as the activation function (f) for all the nodes. Compute the network output matrix $\boldsymbol{F}_{\mathbf{w}}(\mathbf{X})$ (up to 1 decimal place for each entry) based on the given network weights and data.

$$\boldsymbol{F}_{\mathbf{w}}(\mathbf{X}) = \begin{bmatrix} blank1 & blank2 & blank3 \\ blank4 & blank5 & blank6 \end{bmatrix}$$

(MLP classifier, find the best hidden node size, assuming same hidden layer size in each layer, based on cross-validation on the training set and then use it for testing)

Question 5:

Obtain the data set "`from sklearn.datasets import load_iris`".

(a) Split the database into two sets: 80% of samples for training, and 20% of samples for testing using random_state=0

(b) Perform a 5-fold Cross-validation **using only the training set** to determine the best 3-layer `MLPClassifier(from sklearn.neural_network import MLPClassifier` with `hidden_layer_sizes=(Nhidd,Nhidd,Nhidd) for Nhidd in range(1,11))`* for prediction. In other words, partition the **training set** into two sets, 4/5 for training and 1/5 for validation; and repeat this process until each of the 1/5 has been validated. Provide a plot of the average 5-fold training and validation accuracies over the different network sizes.

(c) Find the size of `Nhidd` that gives the best validation accuracy for the training set.

(d) Use this `Nhidd` in the `MLPClassifier` with `hidden_layer_sizes=(Nhidd,Nhidd,Nhidd)` to compute the prediction accuracy based on the 20% of samples for testing in part (a).

* The assumption of `hidden_layer_sizes=(Nhidd,Nhidd,Nhidd)` is to reduce the search space in this exercise. In field applications, the search should take different sizes for each hidden layer.

(An example of handwritten digit image classification using CNN)

Question 6:

Please go through the baseline example in the following link to get a feel of how the Convolutional Neural Network (CNN) can be used for handwritten digit image classification.
https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

Note: This example assumes that you are using standalone Keras running on top of TensorFlow with Python 3 (you might need `conda install -c conda-forge keras tensorflow` to get the Keras library installed).

The following codes might be useful for warnings suppression if you find them annoying:

```
import warnings
warnings.filterwarnings("ignore",category=UserWarning)
```

As the data size and the network size are relatively large comparing with previous assignments, the codes can take quite some time to run (e.g., several minutes running on the latest notebook).