Midterm 2 Practice for COMP6321 Fall 2019

The questions in this practice midterm are suggestive only of the *style* and *difficulty* of questions that will be asked on on the real midterm. The length and the particular course content evaluated will be different. Not all topics are covered by this practice midterm.

Q1. [5 marks total] You are given a very large historical data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ of residential property sales. It has the following features $\mathbf{x}_i$ and targets y_i:

- x_1: $\text{residence\_type} \in \{\text{condo}, \text{house}\}$, the type of property;
- x_2: $\text{num\_bedroom} \in \{0, 1, 2, 3, 4\}$, the number of bedrooms;
- x_3: $\text{num\_bathroom} \in \{1, 2, 3\}$, the number of bathrooms;
- x_4: $\text{year\_of\_sale} \in \{1980, 1981, \ldots, 2018, 2019\}$, the year the property was sold; and
- y: $\text{sale\_price} \in \mathbb{R}_{\geq 0}$, the price at which the property actually sold.

You are building a system to predict a property's potential sale price in 2020, *i.e.* where for a property with current features $\{x_1, x_2, x_3\}$ you set $x_4 = 2020$ and predict its extrapolated price.

a) [3 marks] Suppose you applied the *sklearn.model_selection.train_test_split* function to create a training set for your model and a held-out test set for evaluating it. Would this test set provide an *over*estimate or an *under*estimate of your model's squared error for the intended use? Explain.

b) [2 marks] Propose a training and testing split that, if used, would provide a more accurate estimate of your model's test-time accuracy.

Q2. [8 marks] Given a data set $(\mathbf{X}, \mathbf{y})$, use scikit-learn to do the following:

- Split $(\mathbf{X}, \mathbf{y})$ into 75% training and 25% testing, and normalize the features.
- Train a *GradientBoostingClassifier* using random hyperparameter search with
  - 5-fold cross validation,
  - the maximum tree depth (for weak learners) sampled uniformly from range $\{1, \ldots, 10\}$,
  - the gradient boosting learning rate sampled logarithmically from the interval $[0.01, 10]$, and
  - 30 total hyperparameter settings evaluated.
- Print the training and testing accuracy of your classifier.

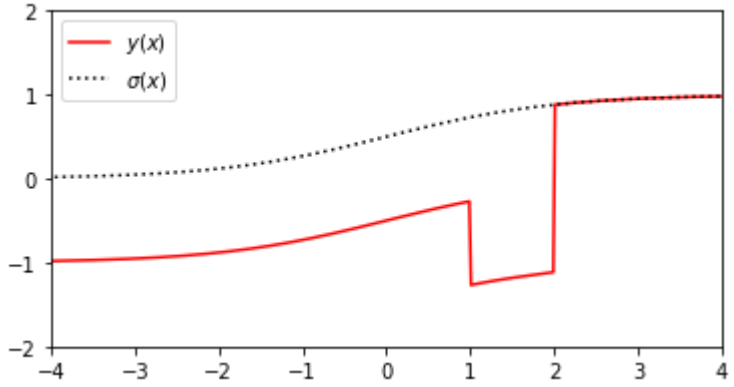
Answer by completing the script below. (Scipy/sklearn documentation will be provided on last page of exam.)

```
import scipy
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
X, y = load_dataset()

# Your code below. Aim for 8-12 lines.
```

Q3. [16 marks total] This question is about *neural networks*.

a) [3 marks] Give a sigmoidal neural network, including coefficients, that would closely approximation the function $y(x)$ shown below. Use as few hidden units as possible. The sigmoid $\sigma(x)$ is shown for reference.



b) [4 marks] Consider the neural network below for a 3-dimensional input $\boldsymbol{x}$:

$$
\begin{aligned}
a_1 &= w_1 x_1 + w_2 x_2 + w_3 \\
a_2 &= w_1 x_2 + w_2 x_3 + w_3 \\
z_1 &= \tanh(a_1) \\
z_2 &= \tanh(a_2) \\
y &= w_4 z_1 + w_5 z_2 + w_6
\end{aligned}
$$

Suppose we compute the squared error loss $\ell = \frac{1}{2}(y - t)^2$ with respect to a target t. Derive the gradient $\nabla_{\boldsymbol{w}} \ell$ with respect to $\boldsymbol{w} = \begin{bmatrix} w_1 & \cdots & w_6 \end{bmatrix}^T$. Note that $\tanh'(a) = 1 - \tanh(a)^2$. Show your steps.

c) [3 marks] Bishop uses the symbols δ_k to and δ_j to represent the "errors" that are computed during the backpropagation algorithm. What is the formula for each "error" computed in your answer to (b), and how many times is each quantity re-used when computing the gradient?

d) [1 marks] Is the neural network in part (b) a convolutional neural network? You must explain your answer.

e) [5 marks] You will use PyTorch to write a *build_convnet* function that can be used like this:

```
>>> convnet = build_convnet(32, 20, 5, 32, 3, 100, 4)
>>> x = torch.rand((5, 3, 20, 32))  # Create a batch of 5 random RGB images of size 32x20
>>> y = convnet(x)                  # Generate output activations
>>> torch.softmax(y, dim=1)         # Convert each row of outputs into class probabilities
tensor([[0.2271, 0.2611, 0.2132, 0.2987],
        [0.2312, 0.2519, 0.2127, 0.3042],
        [0.2188, 0.2627, 0.2169, 0.3017],
        [0.2242, 0.2588, 0.2092, 0.3078],
        [0.2299, 0.2538, 0.2166, 0.2998]])
```

Write PyTorch code to implement the *build_convnet* function below.

```
import torch

def build_convnet(in_width, in_height, num_filters, filter_size, pool_size, num_hidden, num_output):
    """
    Returns a convnet suitable for 4-way classification of images. Specifically:
    * one convolutional layer having num_filters of size filter_size, tanh activations, and
      accepting an RGB image of size in_width x in_height;
    * one max-pooling layer with pool_size regions of stride 1;
    * one fully-connected hidden layer with num_hidden units and rectified linear outputs; and
    * one fully-connected output layer with num_output outputs.
    """
    # Your code here
```

Q4. [2 marks] Give the formula for the Kullbach-Leibler divergence $D_{\mathrm{KL}}(p(\boldsymbol{z}) \mid\mid q(\boldsymbol{z}))$.

Q5. [4 marks] You are given the following function for building and training a variational autoencoder:

```
def train_vae(X, num_layers, num_hidden, activation, beta_coefficient):
    """
    Returns a new VAE trained on trained on (N,D) matrix X.
    The encoder and decoder both have num_layers with num_hidden units per layer.
    Each layer calls activation(a) to compute its outputs, where 'a' is a tensor of activations.
    """
    ...
```

Can this function be used to perform principal component analysis on the first *num_components* of *X*? If not, explain why any such VAE is different from PCA. If yes, show how.