0907531 Machine Learning (Spring 2018)
Final Exam

الاسم: ..... **رقم التسجيل:** ..... **رقم التسلسل:** .....

==========

Instructions: Time **60** min. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. No questions are allowed. There are eight problems. Each problem has 5 points.

==========

P1. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?

If a model performs great on the training data but generalizes poorly to new instances, the model is likely overfitting the training data (or we got extremely lucky on the training data).

Possible solutions to overfitting are getting more data, simplifying the model (selecting a simpler algorithm, reducing the number of parameters or features used, or regularizing the model), or reducing the noise in the training data.

P2. Complete the following code to find and print the model's RMSE on the test set.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

train_set, test_set = train_test_split(housing, test_size=0.3)
X_train = train_set.drop("y", axis=1)
y_train = train_set["y"].copy()
X_test = test_set.drop("y", axis=1)
y_test = test_set["y"].copy()

…   # some code is omitted

forest_reg = RandomForestRegressor(random_state=42)
forest_reg.fit(X_train_prepared, y_train)
X_test_prepared = full_pipeline.transform(X_test)

housing_predictions = forest_reg.predict(X_test_prepared)

mse = mean_squared_error(y_test, housing_predictions)
print("RMSE = ", np.sqrt(mse))
```

P3. The following code is used in the MNIST classification problem. What is the main purpose of this code and how many training jobs it includes?

```
from sklearn.model_selection import GridSearchCV

param_grid = [{'weights': ["uniform", "distance"],
               'n_neighbors': [3, 4, 5]}]

knn_clf = KNeighborsClassifier()
grid_search = GridSearchCV(knn_clf, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

This is grid search to find best hyper-parameters for the KNN classifier.

It fits 5 folds for each of the 2 × 3 = 6 candidates, totaling 30 fits.

P4. Suppose you have an MLP composed of one input layer with 12 passthrough neurons, followed by one hidden layer with 30 artificial neurons, and finally one output layer with 10 artificial neurons. All artificial neurons use the ReLU activation function. Assume that the batch size is 40.

- What is the shape of the input matrix **X**?

  The shape of **X** is 40 × 12

- What about the shape of the hidden layer's weight vector $\mathbf{W}_h$, and the shape of its bias vector $\mathbf{b}_h$?

  The shape of $\mathbf{W}_h$ is 12 × 30 and the length of $\mathbf{b}_h$ is 30

- What is the shape of the output layer's weight vector $\mathbf{W}_o$?

  The shape of $\mathbf{W}_o$ is 30 × 10

- What is the shape of the network's output matrix **Y**?

  The shape of **Y** is 40 × 10

P5. How many neurons do you need in the output layer if you want to classify tumor into benign and malignant? What activation function should you use in the output layer? If instead you want to tackle English letter classification, how many neurons do you need in the output layer, using what activation function? Answer the same questions for getting your network to predict oil prices.

Problem	Output Layer Size	Activation Function
Classify tumor	**One neuron**	**Logistic**
Letter classification	**26 neurons**	**Softmax**
Oil prices	**One neuron**	**None**

P6. For the following two-layer convolutional network, what is the number of output feature maps of this network and what is the size of each map?

```
height = 28
width = 28
channels = 1
n_inputs = height * width

with tf.name_scope("inputs"):
    X = tf.placeholder(tf.float32, shape=[None, n_inputs], name="X")
    X_reshaped = tf.reshape(X, shape=[-1, height, width, channels])

conv1 = tf.layers.conv2d(X_reshaped, filters=32, kernel_size=3,
                         strides=1, padding="SAME",
                         activation=tf.nn.relu, name="conv1")
conv2 = tf.layers.conv2d(conv1, filters=64, kernel_size=3,
                         strides=2, padding="SAME",
                         activation=tf.nn.relu, name="conv2")
```

There are 64 feature maps.

Each map is 14 by 14.

P7. Consider a convolutional layer with 3 × 3 filters, outputting 100 feature maps of size 50 × 50, with stride 1 and SAME padding. If the input is RGB image (three channels), what is the number of parameters that need to be trained for this layer?

The number of parameters is (3 × 3 × 3 + 1) × 100 = 2,800

P8. Complete the following code to implement an RNN that has two LSTM layers. The first layer should have 100 neurons and the second layer should have 50 neurons. Use TensorFlow's dynamic unrolling through time function.

```
n_steps = 28
n_inputs = 28

X = tf.placeholder(tf.float32, [None, n_steps, n_inputs])

# your code goes here

layers = [tf.contrib.rnn. BasicLSTMCell(num_units=100),
          tf.contrib.rnn. BasicLSTMCell(num_units=50)]

multi_layer_cell = tf.contrib.rnn.MultiRNNCell(layers)

outputs, states = tf.nn.dynamic_rnn(multi_layer_cell, X,
          dtype=tf.float32)
```

<Good Luck>