Johns Hopkins University

665.645.8VL: Artificial Intelligence for Robotics

Homework and PA #3 [300 points]

PART A: Theory and Algorithms *[100 points]* ** See PART B Programming Assignment on Page 3*

Please - clearly write your **full name** on the first page. Submit a single PDF file as your answer for Part A. Please provide brief but complete explanations, using diagrams where necessary, and suitably using your own words. While presenting calculations and equations, explain the variables and answers in words.

Study Chapter 18 of Russel AI textbook – selected sections only, plus propositional Logic notes provided. Answer the below: (AIMA 3rd Edition)

1. **From textbook p.763** [40 points]
   18.1
   18.3
   18.17
   18.19
2. Study the Home Credit Default Risk Kaggle competition and data sets. [10 points]

https://www.kaggle.com/c/home-credit-default-risk

Specify 4 machine learning questions relevant to this use case.
Against each ML question also list one or more algorithms which would help answer it.

3. Study the SVM Python coding/tutorials at the below links: [25 points]
   https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/
   https://towardsdatascience.com/support-vector-machine-python-example-d67d9b63f1c8

Then, use the below data set and apply SVM using Py libraries to classify (SEPARATELY) (i) Flowers and (ii) Mushrooms using the Iris and Mushroom data sets.
https://archive.ics.uci.edu/ml/datasets/Iris
https://archive.ics.uci.edu/ml/datasets/Mushroom

4. Implement the above classification problem from #3 using Artificial Neural Networks (ANN) in Python. Compare results. [25 points]

Do not implement the NN from scratch – use libraries like ANN().
https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
https://www.pyimagesearch.com/2016/09/26/a-simple-neural-network-with-python-and-keras/

<u>Section 2</u>:

Ridge and Lasso Regression for Regularization [75 points]

In this assignment, you might need Jupyter notebook, as you will play around with the models. For this, you would need to install it. **Study and refer** to this implementation:
http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10-py.html

Install Jupyter notebook and ipynb

with conda:

```
    conda install ipython jupyter
```

with pip:

```
    # first, always upgrade pip!

    pip install --upgrade pip

    pip install --upgrade ipython jupyter
```

To open the ipynb files, you can open it using the command ipython notebook filename.ipynb from t
he directory it is downloaded on to. After running this command, you will be navigated to the Juypter notebook running on localhost:8888.

Zip file contents:

Train.csv
House-Price-Prediction-Ridge-Lasso.ipynb

The Jupyter notebook is trying to solve the below problem statement

Problem Statement for the example provided is below:
A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. The company wants to know

- Which variables are significant in predicting the price of a house, and
- How well those variables describe the price of a house.

Assignment Question:
Based on the above example, you pick a data set and a problem of your choice and come up with some prediction scenario and use ridge and lasso regression to come up with the factors that influence your dataset the most.

A good place for Data sets (not needed for this PA; just a reference)
https://www.kaggle.com/datasets

Part 3: Implement a CNN

Instructions: Please make changes as needed and complete an implementation of CNN. [125 points]

- Make a copy of this notebook in your own Colab and complete the implementation.
- You can add more cells if necessary. You may also add descriptions to your code, though it is not mandatory.
- Make sure the notebook can run through by *Runtime* -> *Run all*. **Keep all cell outputs** for grading.
- Install PyTorch and Skorch.

```
pip install -q torch skorch torchvision torchtext
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
import skorch
import sklearn
import numpy as np
import matplotlib.pyplot as plt
```

Implement a convolutional neural network for image classification on CIFAR-10 dataset.

CIFAR-10 is an image dataset of 10 categories. Each image has a size of 32x32 pixels. The following code will download the dataset, and split it into train and test. For this question, we use the default validation split generated by Skorch.

```
train = torchvision.datasets.CIFAR10("./data", train=True, download=True) test =
torchvision.datasets.CIFAR10("./data", train=False, download=True)
```

The following code visualizes some samples in the dataset. You may use it to debug your model if necessary.

```
def plot(data, labels=None, num_sample=5):
  n = min(len(data), num_sample)
  for i in range(n):
    plt.subplot(1, n, i+1)
    plt.imshow(data[i], cmap="gray")
    plt.xticks([])
    plt.yticks([])
    if labels is not None:
      plt.title(labels[i])

train.labels = [train.classes[target]for target in train.targets]
plot(train.data, train.labels)
```

1) Basic CNN implementation

Consider a basic CNN model

- It has 3 convolutional layers, followed by a linear layer.
- Each convolutional layer has a kernel size of 3, a padding of 1.
- ReLU activation is applied on every hidden layer.

Please implement this model in the following section. You will need to tune the hyperparameters and fill the results in the table.

a) Implement convolutional layers

Implement the initialization function and the forward function of the CNN.

```
class CNN(nn.Module):
  def __init__(self):
    super(CNN, self).__init__()
    # implement parameter definitions here

  def forward(self, images):
    # implement the forward function here
    return None
```

b) Tune hyperparameters

Train the CNN model on CIFAR-10 dataset. Tune the number of channels, optimizer, learning rate and the number of epochs for best validation accuracy.

```
#    implement    hyperparameters    here    model    =    skorch.NeuralNetClassifier(CNN,
criterion=torch.nn.CrossEntropyLoss,                         device="cuda") # implement input
normalization & type cast here model.fit(train.data, train.targets)
```

Write down **validation accuracy** of your model under different hyperparameter settings. Note the validation set is automatically split by Skorch during model.fit().

Hint: You may need more epochs for SGD than Adam.

#channel for each layer \ optimizer	**SGD**	**Adam**
(128, 128, 128)		
(256, 256, 256)		
(512, 512, 512)		

2) Full CNN implementation - Basic

Based on the CNN in the previous question, implement a full CNN model with max pooling layer.

- Add a max pooling layer after each convolutional layer.
- Each max pooling layer has a kernel size of 2 and a stride of 2.

Please implement this model in the following section. You will need to tune the hyperparameters and fill the results in the table. You are also required to complete the questions.

a) Implement max pooling layers

Copy the CNN implementation in previous question. Implement max pooling layers.

```
class CNN_MaxPool(nn.Module):
  def __init__(self):
    super(CNN_MaxPool, self).__init__()
    # implement parameter definitions here


  def forward(self, images):
    # implement the forward function here
    return None
```