Homework 1

Pan Li
CSCI-GA 3033-090: Deep Reinforcement Learning

September 15, 2020

Solution 1. *The linear regression model would be* $y = 0.5x$

Solution 2. *K-means is a clustering method which aims to partition n observations into k clusters, where each observation belongs to the cluster with the nearest mean or the cluster centers. It is used to identify similar groups which have not been explicitly labeled in the data. Meanshift is also a clustering method, and it is a centroid-based algorithm, meaning that it works by updating candidates for centroids to be the mean of the points within a given region. The differences between the two methods lie in that Meanshift does not rely upon a priori knowledge of the number of clusters whereas K-Means algorithm is an unsupervised clustering algorithm that requires explicit input of the cluster number.*

Solution 3. *Softmax is a generalization of logistic regression that can be used for multi-class classification. It is a function that turns a vector of real values into another vector that sums to 1. In this way, it converts the scores to a normalized probability distribution. During the neural network computation, softmax would become unstable with exploding gradient or vanishing gradient.*

Solution 4. *In CNNs, regularization is a method that controls the model complexity and to prevent overfitting. The popular regularization techniques include L1 regularization, L2 regularization and Dropout (randomly ”dropping out” units during the training process of a neural network).*

Solution 5. *GAN, or Generative Adversarial Network consists of two neural networks (generator/discriminator) that contest with each other in a zero-sum game, where one agent's gain is another agent's loss. Given a training set, GAN learns to generate new data with the same distribution as the training set. "Mode collapse" refers to the phenomenon that when a generator produces an especially plausible output, it may learn to produce only that output. As a result, the generators rotate through a small set of output types.*

Solution 6. *To compute PCA for the data points, first I will standardize and normalize the dataset. Then I will compute the covariance matrix based on the given features. I will calculate the eigenvalues and eigenvectors for the covariance matrix subsequently, and then sort eigenvalues and their corresponding eigenvectors based on their values. I will pick the top-k eigenvalues and form the matrix of eigenvectors accordingly. And finally, I will transform transform the original matrix.*

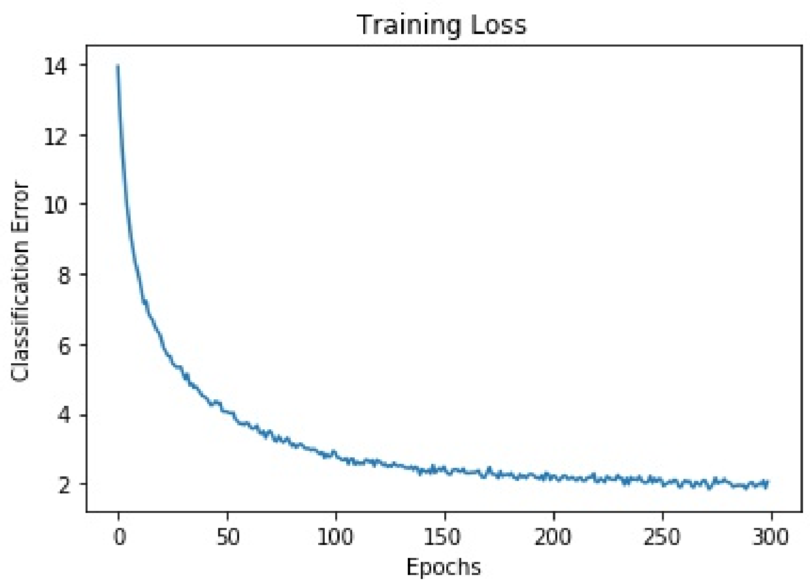

Figure 1: Training Curve

Solution 7. *Some common initialization methods include Zero initialization, Random initialization, Xavier initialization and Kaiming Initialization.*

Solution 8. *Training curve is shown in Figure 1. The final cross-entropy classification loss is 1.8711. Classification accuracy is 100% on the training set, so the visualization of errors cannot be shown. In my model, I have used a fully-connected network as the input layer, and another fully-connected network as the output layer. To improve the classification performance, I used the ReLU activation function as well as the Dropout technique. I also choose the learning rate carefully in order to keep a consistent performance.*

```
from torch.utils.data import DataLoader, Dataset
import pandas as pd
import torch
import torch.nn as nn
from torch.autograd import Variable
import matplotlib.pyplot as plt

class Trainset(Dataset):
    def __init__(self, data):
        feature, group = [], []
        for _, row in data.iterrows():
            feature.append(row['feature'])
            group.append(row['class'])
        group = [int(x/2-1) for x in group]
        self.feature = torch.FloatTensor(feature)
        self.group = torch.LongTensor(group)
    def __getitem__(self, index):
        return self.feature[index], self.group[index]
    def __len__(self):
        return self.feature.size(0)

class Classifier(nn.Module):
    def __init__(self, hidden_size, dropout=0.1):
        super(Classifier, self).__init__()
        self.hidden_size = hidden_size
        self.dropout = dropout

        self.linear = nn.Linear(feature_size, hidden_size)
        self.affine_linear = nn.Linear(hidden_size, 2)

        self.initialize_param()

    def initialize_param(self, initrange=0.1):
        # Initialize Linear Weight
        self.linear.weight.data.uniform_(-initrange, initrange)
        self.linear.bias.data.fill_(0)
        self.affine_linear.weight.data.uniform_(-initrange, initrange)
        self.affine_linear.bias.data.fill_(0)

    def forward(self, feature):
        vector = self.linear(feature)
        vector = torch.nn.Dropout(p=0.1)(vector)
        vector = torch.nn.ReLU()(vector)
        vector = self.affine_linear(vector)
        return vector

names = ['code_number','clump_thickness','cell_size','cell_shape',
        'marginal_adhesion','single_epithelial','bare_nuclei',
        'bland_chromatin','normal_nucleoli','mitoses','class']
batch_size = 32
epoch = 300
hidden_size = 128
feature_size = len(names)-3
lr = 0.01

data = pd.read_csv('breast-cancer-wisconsin.data',names=names)

data['feature'] = data[['clump_thickness','cell_size','cell_shape',
        'marginal_adhesion','single_epithelial',
        'bland_chromatin','normal_nucleoli','mitoses']].values.tolist()
train_loader = DataLoader(Trainset(data=data), batch_size=batch_size,
    shuffle=True)
model = Classifier(hidden_size, feature_size)
loss_list = []
for epoch_id in range(epoch):
   model.train()
   total_loss = 0
   optimizer = torch.optim.SGD(model.parameters(), lr=lr)
   crit = torch.nn.CrossEntropyLoss()
   for batch in train_loader:
       optimizer.zero_grad()
       feature, group = Variable(batch[0]), Variable(batch[1])
       group_pred = model.forward(feature)
       loss = crit(group_pred, group)
       loss.backward()
       optimizer.step()
       total_loss += loss
   loss_list.append(total_loss.detach().numpy())

x = range(epoch)
plt.plot(x,loss_list)
plt.xlabel('Epochs')
plt.ylabel('Classification Error')
plt.title('Training Loss')
plt.savefig("training_epoch.jpg",bbox_inches='tight')
```