Name(s): Md Ridwan Hossain Talukder
NetID(s): mtalukd
Team name on Kaggle leaderboard: mtalukd

For each of the sections below, your reported test accuracy should approximately match the accuracy reported on Kaggle.

Perceptron

Initially, I tried to use a learning rate without decay and used mini-batch gradient descent but the accuracy was around 0.31, and using the learning decay improved the accuracy by almost 6%. So yes adding a learning rate decay helped. I used the common method for learning rate decay which is
$\alpha = (1/(1 + decayRate \times epochNumber)) * \alpha 0,\ where\ the\ decayRate\ =\ 0.01.$
For hyperparameter tuning I tried some different combinations of the learning rate, number of epochs, and batch size (as I have used mini-batches so tuning the batch size helped). I have listed the combinations I used below:

lr	n_epochs	Tr_acc	Val_acc	Test_acc	Batch Size
0.5	10	37.435	33.77	33.77	100
0.5	20	37.435	33.77	33.77	100
0.01	30	37.7275	33.7	34.23	100
0.5	20	40.015	34.12	34.5	200
0.5	40	40.015	34.12	34.5	200
0.2	40	40.2	34.16	34.48	200
0.2	20	40.2	34.16	34.48	200
0.2	10	40.2	34.16	34.48	200
0.1	10	40.0825	34.18	34.4	200
0.1	10	40.0825	34.18	34.4	200
0.01	10	40.095	34.5	34.29	200
0.01	10	40.095	34.5	34.29	200
0.5	50	40.015	34.12	34.5	200
0.5	20	42.76	37.28	37.21	400
0.5	30	42.3975	37.05	37.39	400
0.5	50	41.9675	37.05	36.03	400
0.5	30	43.5575	38.14	37.97	800
0.8	30	43.5475	38.06	37.51	800

lr	n_epochs	Tr_acc	Val_acc	Test_acc	Batch Size
0.5	10	37.435	33.77	33.77	100
0.5	20	37.435	33.77	33.77	100
0.01	30	37.7275	33.7	34.23	100
0.3	30	43.6575	38.22	37.77	800
0.4	30	43.82	38.45	37.63	800
0.005	25	43.4475	37.64	37.47	800
0.001	30	43.2925	38.39	37.78	800
0.0001	30	43.5475	38.03	37.34	800

CIFAR DATASET

Optimal hyperparameters:	lr = 0.5, n_epochs = 30
Training accuracy:	43.5575
Validation accuracy:	38.14
Test accuracy:	37.97

SVM

I have used learning rate decay here as well.

lr	n_epochs	reg_const	Tr_acc	Val_acc	Test_acc
0.5	10	0.05	37.79	35.67	35.93
0.1	10	0.05	37.83	35.65	35.86
0.01	10	0.05	37.7825	35.57	35.97
0.001	10	0.05	37.775	35.76	35.89
0.0001	10	0.05	35.13	33.38	33.59
0.001	10	0.01	37.825	35.75	35.81
0.001	10	0.1	37.805	35.88	35.82
0.001	30	0.05	38.0025	36.02	35.17
0.001	50	0.05	38.1425	35.52	35.55

CIFAR DATASET

Optimal hyperparameters:	lr = 0.001, n_epochs = 30, reg_const = 0.05
Training accuracy:	38.002500
Validation accuracy:	36.020000
Test accuracy:	35.170000

Softmax

lr	n_epochs	reg_const	Tr_acc	Val_acc	Test_acc
0.5	10	0.05	34.19	33.32	34.09
0.1	10	0.05	34.2	33.36	34.08
0.01	10	0.05	34.1875	33.29	34.11
0.001	10	0.05	33.985	33.07	33.81
0.0001	10	0.05	35.13	33.38	33.59
0.01	10	0.01	34.1725	33.29	34.07
0.01	10	0.1	34.18	33.28	34.12
0.01	10	0.5	34.155	33.23	34.09
0.01	20	0.1	34.2775	33.31	34.2
0.01	30	0.1	34.2775	33.19	34.2
0.01	20	0.05	34.31	33.28	34.25

CIFAR DATASET

Optimal hyperparameters:	lr = 0.01 n_epochs = 20 reg_const = 0.05
Training accuracy:	34.310000
Validation accuracy:	33.280000
Test accuracy:	34.250000

2. Support Vector Machine (bonus: 20 points)

Part-1:

I have used my SVM to check if the CIFAR-10 is linearly separable or not. For that I trained the SVM with all the training data of specific two classes and choose a subset of that data for testing to see whether it can classify correctly those data but it seems it can not classify them correctly even though the test data is coming from the data set containing only those two classes. If it was linearly separable then the accuracy would have been 100%. Which is not the case. The code is in the ipynb file but I am listing it here as well:

```
lr = 0.001
n_epochs = 30
reg_const = 0.05

classes = np.unique(y_train_CIFAR)
for class_label in classes:
  for i in range(class_label+1, len(classes)):
    x = np.where((y_train_CIFAR == class_label) | (y_train_CIFAR ==
i))[0]
    y_sample = np.take(y_train_CIFAR, x)
    x_sample = np.zeros((y_sample.shape[0], X_train_CIFAR.shape[1]))
    for ind, val in enumerate(x):
      x_sample[ind] = X_train_CIFAR[val]
    x_train,x_test,y_train,y_test =
train_test_split(x_sample,y_sample,test_size=0.1)
    svm_CIFAR = SVM(n_class_CIFAR, lr, n_epochs, reg_const)
    svm_CIFAR.train(x_sample, y_sample)
    pred_svm = svm_CIFAR.predict(x_test)
    acc = get_acc(pred_svm, y_test)
    print(f'The training accuracy for class {class_label, i} is
{acc}%')
```

```
Class (0, 1) are not linearly separable as the accuracy is 77.81954887218046
Class (0, 2) are not linearly separable as the accuracy is 73.00995024875621
Class (0, 3) are not linearly separable as the accuracy is 75.4077791718946
Class (0, 4) are not linearly separable as the accuracy is 78.5982478097622
Class (0, 5) are not linearly separable as the accuracy is 78.04265997490589
Class (0, 6) are not linearly separable as the accuracy is 87.39076154806492
Class (0, 7) are not linearly separable as the accuracy is 79.27590511860176
Class (0, 8) are not linearly separable as the accuracy is 68.33541927409262
Class (0, 9) are not linearly separable as the accuracy is 76.28607277289838
Class (1, 2) are not linearly separable as the accuracy is 78.35820895522389
Class (1, 3) are not linearly separable as the accuracy is 72.39648682559599
Class (1, 4) are not linearly separable as the accuracy is 83.4793491864831
Class (1, 5) are not linearly separable as the accuracy is 79.79924717691343
Class (1, 6) are not linearly separable as the accuracy is 85.0187265917603
Class (1, 7) are not linearly separable as the accuracy is 81.5230961298377
Class (1, 8) are not linearly separable as the accuracy is 79.09887359198999
Class (1, 9) are not linearly separable as the accuracy is 68.75784190715181
Class (2, 3) are not linearly separable as the accuracy is 71.51741293532339
Class (2, 4) are not linearly separable as the accuracy is 60.9181141439206
Class (2, 5) are not linearly separable as the accuracy is 67.12328767123287
Class (2, 6) are not linearly separable as the accuracy is 67.65799256505576
Class (2, 7) are not linearly separable as the accuracy is 75.12376237623762
Class (2, 8) are not linearly separable as the accuracy is 80.8695652173913
Class (2, 9) are not linearly separable as the accuracy is 81.3200498132005
Class (3, 4) are not linearly separable as the accuracy is 70.08760951188987
Class (3, 5) are not linearly separable as the accuracy is 57.66331658291457
Class (3, 6) are not linearly separable as the accuracy is 65.29338327091136
Class (3, 7) are not linearly separable as the accuracy is 73.78277153558052
Class (3, 8) are not linearly separable as the accuracy is 79.34918648310388
Class (3, 9) are not linearly separable as the accuracy is 79.17189460476789
Class (4, 5) are not linearly separable as the accuracy is 66.66666666666666
Class (4, 6) are not linearly separable as the accuracy is 66.0024906600249
Class (4, 7) are not linearly separable as the accuracy is 68.74221668742216
Class (4, 8) are not linearly separable as the accuracy is 83.125
Class (4, 9) are not linearly separable as the accuracy is 83.35419274092615
Class (5, 6) are not linearly separable as the accuracy is 67.375
Class (5, 7) are not linearly separable as the accuracy is 74.625
Class (5, 8) are not linearly separable as the accuracy is 79.9498746867168
Class (5, 9) are not linearly separable as the accuracy is 80.52763819095478
Class (6, 7) are not linearly separable as the accuracy is 80.99378881987577
Class (6, 8) are not linearly separable as the accuracy is 91.52119700748129
Class (6, 9) are not linearly separable as the accuracy is 87.625
Class (7, 8) are not linearly separable as the accuracy is 86.65835411471322
Class (7, 9) are not linearly separable as the accuracy is 78.4019975031211
```

Class (8, 9) are not linearly separable as the accuracy is 76.06516290726817

Part-2:

In a synthetic dataset with two classes that are highly non-linearly separable, we can combine sine and cosine functions to produce features for each class. The data can be classified as class 1 or class 2 depending on whether a point is above or below a predetermined threshold by using sine and cosine functions to generate two features, x1 and x2.

The Python code to generate such a dataset is provided here (also provided in ipynb file):

```
lr = 0.001
n_epochs = 30
reg_const = 0.05

X = np.random.uniform(0, 2*np.pi, size=(1000, 2))
X1 = np.column_stack([np.sin(X[:,0]), np.cos(X[:,1])])
X2 = np.column_stack([np.cos(X[:,0]), np.sin(X[:,1])])
threshold = 0.5
y1 = np.where(X1[:,0] + X1[:,1] < threshold, 0, 1)
y2 = np.where(X2[:,0] + X2[:,1] < threshold, 1, 0)
X = np.vstack([X1, X2])
y = np.concatenate([y1, y2])
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.1)
svm_CIFAR = SVM(n_class_CIFAR, lr, n_epochs, reg_const)
svm_CIFAR.train(x_train, y_train)
pred_svm = svm_CIFAR.predict(x_test)
acc = get_acc(pred_svm, y_test)
print(f'The accuracy for class {class_label, i} is {acc}%')
```

And the accuracy of the SVM for this data set is 33.0%