CS 444 Assignment-1

February 13, 2024

1 (Optional) Colab Setup

If you aren't using Colab, you can delete the following code cell. This is just to help students with mounting to Google Drive to access the other .py files and downloading the data, which is a little trickier on Colab than on your local machine using Jupyter.

[1]:
```
# you will be prompted with a window asking to grant permissions
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
# fill in the path in your Google Drive in the string below. Note: do not␣
 ↪escape slashes or spaces
import os
datadir = "/content/drive/MyDrive/CS444/assignment1/"
# if not os.path.exists(datadir):
#   !ln -s "/drive/MyDrive/CS444/assignment1/" $datadir
os.chdir(datadir)
!pwd
```

```
/content/drive/MyDrive/CS444/assignment1
```

[3]:
```
# downloading Fashion-MNIST
import os
os.chdir(os.path.join(datadir,"fashion-mnist/"))
!chmod +x ./get_data.sh
!./get_data.sh
os.chdir(datadir)
```

```
--2024-02-13 23:37:05--
https://raw.githubusercontent.com/zalandoresearch/fashion-
mnist/master/data/fashion/t10k-images-idx3-ubyte.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
```

```
HTTP request sent, awaiting response… 200 OK
Length: 4422102 (4.2M) [application/octet-stream]
Saving to: ‘t10k-images-idx3-ubyte.gz.19’

t10k-images-idx3-ub 100%[===================>]   4.22M  15.6MB/s    in 0.3s

2024-02-13 23:37:06 (15.6 MB/s) - ‘t10k-images-idx3-ubyte.gz.19’ saved
[4422102/4422102]

--2024-02-13 23:37:06--
https://raw.githubusercontent.com/zalandoresearch/fashion-
mnist/master/data/fashion/t10k-labels-idx1-ubyte.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.109.133, 185.199.111.133, 185.199.108.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 5148 (5.0K) [application/octet-stream]
Saving to: ‘t10k-labels-idx1-ubyte.gz.19’

t10k-labels-idx1-ub 100%[===================>]   5.03K  --.-KB/s    in 0.02s

2024-02-13 23:37:06 (212 KB/s) - ‘t10k-labels-idx1-ubyte.gz.19’ saved
[5148/5148]

--2024-02-13 23:37:06--
https://raw.githubusercontent.com/zalandoresearch/fashion-
mnist/master/data/fashion/train-images-idx3-ubyte.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 26421880 (25M) [application/octet-stream]
Saving to: ‘train-images-idx3-ubyte.gz.19’

train-images-idx3-u 100%[===================>]  25.20M  25.2MB/s    in 1.0s

2024-02-13 23:37:08 (25.2 MB/s) - ‘train-images-idx3-ubyte.gz.19’ saved
[26421880/26421880]

--2024-02-13 23:37:08--
https://raw.githubusercontent.com/zalandoresearch/fashion-
mnist/master/data/fashion/train-labels-idx1-ubyte.gz
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.110.133, 185.199.109.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
```

```
HTTP request sent, awaiting response… 200 OK
Length: 29515 (29K) [application/octet-stream]
Saving to: ‘train-labels-idx1-ubyte.gz.19’

train-labels-idx1-u 100%[===================>]  28.82K  --.-KB/s    in 0.004s

2024-02-13 23:37:08 (7.00 MB/s) - ‘train-labels-idx1-ubyte.gz.19’ saved
[29515/29515]
```

2 Imports

[4]:
```
import random
import numpy as np
from data_process import get_FASHION_data, get_RICE_data
from scipy.spatial import distance
from models import Perceptron, SVM, Softmax, Logistic
from kaggle_submission import output_submission_csv
%matplotlib inline

# For auto-reloading external modules
# See http://stackoverflow.com/questions/1907993/
  ↪autoreload-of-modules-in-ipython
%load_ext autoreload
%autoreload 2
```

3 Loading Fashion-MNIST

In the following cells we determine the number of images for each split and load the images. TRAIN_IMAGES + VAL_IMAGES = (0, 60000] , TEST_IMAGES = 10000

[5]:
```
# You can change these numbers for experimentation
# For submission we will use the default values
TRAIN_IMAGES = 50000
VAL_IMAGES = 10000
normalize = True
```

[6]:
```
data = get_FASHION_data(TRAIN_IMAGES, VAL_IMAGES, normalize=normalize)
X_train_fashion, y_train_fashion = data['X_train'], data['y_train']
X_val_fashion, y_val_fashion = data['X_val'], data['y_val']
X_test_fashion, y_test_fashion = data['X_test'], data['y_test']
n_class_fashion = len(np.unique(y_test_fashion))
```

4 Loading Rice

[7]:
```
# loads train / test / val splits of 80%, 20%, 20%
data = get_RICE_data()
X_train_RICE, y_train_RICE = data['X_train'], data['y_train']
X_val_RICE, y_val_RICE = data['X_val'], data['y_val']
X_test_RICE, y_test_RICE = data['X_test'], data['y_test']
n_class_RICE = len(np.unique(y_test_RICE))

print("Number of train samples: ", X_train_RICE.shape[0])
print("Number of val samples: ", X_val_RICE.shape[0])
print("Number of test samples: ", X_test_RICE.shape[0])
```

```
Number of train samples:  10911
Number of val samples:  3637
Number of test samples:  3637
```

4.0.1 Get Accuracy

This function computes how well your model performs using accuracy as a metric.

[8]:
```
def get_acc(pred, y_test):
    return np.sum(y_test == pred) / len(y_test) * 100
```

5 Perceptron

Perceptron has 2 hyperparameters that you can experiment with: ### Learning rate The learning rate controls how much we change the current weights of the classifier during each update. We set it at a default value of 0.5, but you should experiment with different values. Here is a guide to help you find a right learning rate: - Try values ranging from 5.0 to 0.0005 to see the impact on model accuracy. - If the accuracy fluctuates a lot or diverges, the learning rate is too high. Try decreasing it by a factor of 10 (e.g. from 0.5 to 0.05). - If the accuracy is changing very slowly, the learning rate may be too low. Try increasing it by a factor of 10. - You can also try adding a learning rate decay to slowly reduce the learning rate over each training epoch. For example, multiply the learning rate by 0.95 after each epoch. - Plot training and validation accuracy over epochs for different learning rates. This will help you visualize the impact of the learning rate. - Here is a detailed guide to learning rate.

5.0.1 Number of Epochs

An epoch is a complete iterative pass over all of the data in the dataset. During an epoch we predict a label using the classifier and then update the weights of the classifier according to the perceptron update rule for each sample in the training set. You should try different values for the number of training epochs and report your results.

You will implement the Perceptron classifier in the **models/perceptron.py**

The following code: - Creates an instance of the Perceptron classifier class - The train function of the Perceptron class is trained on the training data - We use the predict function to find the

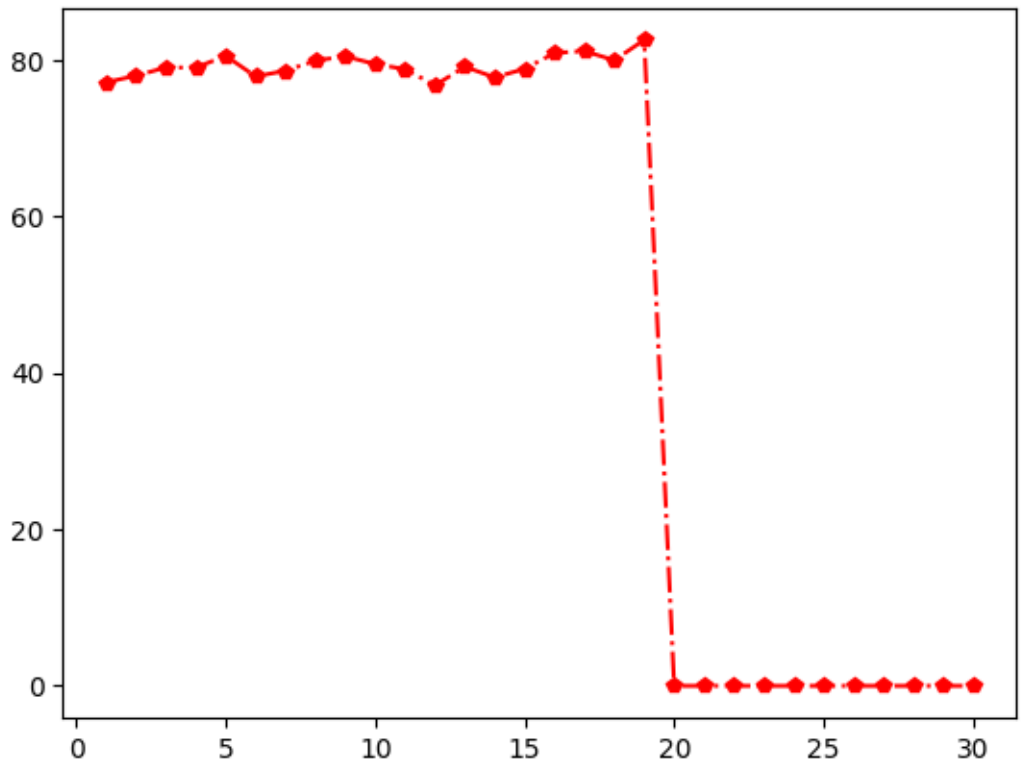
training accuracy as well as the testing accuracy

5.1 Train Perceptron on Fashion-MNIST

[9]:
```
### Experiment
lr = 0.005
n_epochs = 30

import matplotlib.pyplot as plt

x = np.linspace(1, n_epochs, n_epochs)
y = np.zeros(n_epochs)

percept_fashion = Perceptron(n_class_fashion, lr, 1)
for epoch in range(n_epochs):
  print("training epoch {}:".format(epoch))
  percept_fashion.train(X_train_fashion, y_train_fashion)
  pred_percept = percept_fashion.predict(X_val_fashion)
  print('The validation accuracy is: %f' % (get_acc(pred_percept,␣
 ↪y_val_fashion)))
  y[epoch] = get_acc(pred_percept, y_val_fashion)
  if y[epoch] > 82:
    break

plt.plot(x, y, 'r-.p')
plt.show()
```

```
training epoch 0:
The training accuracy is 77.806
The validation accuracy is: 77.120000
training epoch 1:
The training accuracy is 79.032
The validation accuracy is: 77.950000
training epoch 2:
The training accuracy is 79.822
The validation accuracy is: 79.010000
training epoch 3:
The training accuracy is 80.684
The validation accuracy is: 78.990000
training epoch 4:
The training accuracy is 81.554
The validation accuracy is: 80.490000
training epoch 5:
The training accuracy is 80.166
The validation accuracy is: 77.860000
training epoch 6:
The training accuracy is 80.138
The validation accuracy is: 78.590000
```

```
training epoch 7:
The training accuracy is 81.992
The validation accuracy is: 79.880000
training epoch 8:
The training accuracy is 82.098
The validation accuracy is: 80.430000
training epoch 9:
The training accuracy is 81.492
The validation accuracy is: 79.440000
training epoch 10:
The training accuracy is 81.146
The validation accuracy is: 78.790000
training epoch 11:
The training accuracy is 78.776
The validation accuracy is: 76.730000
training epoch 12:
The training accuracy is 81.07799999999999
The validation accuracy is: 79.090000
training epoch 13:
The training accuracy is 80.28999999999999
The validation accuracy is: 77.740000
training epoch 14:
The training accuracy is 81.596
The validation accuracy is: 78.830000
training epoch 15:
The training accuracy is 83.31
The validation accuracy is: 80.860000
training epoch 16:
The training accuracy is 83.62
The validation accuracy is: 81.100000
training epoch 17:
The training accuracy is 82.66799999999999
The validation accuracy is: 79.930000
training epoch 18:
The training accuracy is 84.734
The validation accuracy is: 82.530000
```



[10]:
```
opt_epoches = np.argmax(y) + 1
opt_acc = np.max(y)
print("optimal n_epoches =", opt_epoches)
print("optimal valid accuracy =", opt_acc)
```

```
optimal n_epoches = 19
optimal valid accuracy = 82.53
```

[11]:
```
### Output Optimal
percept_fashion = Perceptron(n_class_fashion, lr, opt_epoches)
percept_fashion.train(X_train_fashion, y_train_fashion)
```

```
The training accuracy is 77.806
The training accuracy is 79.032
The training accuracy is 79.822
The training accuracy is 80.684
The training accuracy is 81.554
The training accuracy is 80.166
The training accuracy is 80.138
The training accuracy is 81.992
The training accuracy is 82.098
The training accuracy is 81.492
The training accuracy is 81.146
```

```
The training accuracy is 78.776
The training accuracy is 81.07799999999999
The training accuracy is 80.28999999999999
The training accuracy is 81.596
The training accuracy is 83.31
The training accuracy is 83.62
The training accuracy is 82.66799999999999
The training accuracy is 84.734
```

[12]:
```
pred_percept = percept_fashion.predict(X_train_fashion)
print('The training accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_train_fashion)))
```

```
The training accuracy is given by: 84.734000
```

5.1.1 Validate Perceptron on Fashion-MNIST

[13]:
```
pred_percept = percept_fashion.predict(X_val_fashion)
print('The validation accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_val_fashion)))
```

```
The validation accuracy is given by: 82.530000
```

5.1.2 Test Perceptron on Fashion-MNIST

[14]:
```
pred_percept = percept_fashion.predict(X_test_fashion)
print('The testing accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_test_fashion)))
```

```
The testing accuracy is given by: 81.820000
```

5.1.3 Perceptron_Fashion-MNIST Kaggle Submission

Once you are satisfied with your solution and test accuracy, output a file to submit your test set predictions to the Kaggle for Assignment 1 Fashion-MNIST. Use the following code to do so:

[15]:
```
output_submission_csv('kaggle/perceptron_submission_fashion.csv',␣
 ↪percept_fashion.predict(X_test_fashion))
```

5.2 Train Perceptron on Rice

[16]:
```
lr = 0.000001
n_epochs = 8

percept_RICE = Perceptron(n_class_RICE, lr, n_epochs)
percept_RICE.train(X_train_RICE, y_train_RICE)
# print(y_train_RICE)
```

```
The training accuracy of epoch 0 is 58.50059572908074
The training accuracy of epoch 1 is 98.13949225552196
The training accuracy of epoch 2 is 94.9042250939419
The training accuracy of epoch 3 is 98.62524058289799
The training accuracy of epoch 4 is 99.6333974887728
The training accuracy of epoch 5 is 99.56924204930803
The training accuracy of epoch 6 is 99.01017321968656
The training accuracy of epoch 7 is 99.70671799101824
```

[17]:
```
pred_percept = percept_RICE.predict(X_train_RICE)
print('The training accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_train_RICE)))
```

```
The training accuracy is given by: 99.706718
```

5.2.1 Validate Perceptron on Rice

[18]:
```
pred_percept = percept_RICE.predict(X_val_RICE)
print('The validation accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_val_RICE)))    ### !!! y_val_RICE in {0, 1} form
```

```
The validation accuracy is given by: 99.670058
```

5.2.2 Test Perceptron on Rice

[19]:
```
pred_percept = percept_RICE.predict(X_test_RICE)
print('The testing accuracy is given by: %f' % (get_acc(pred_percept,␣
 ↪y_test_RICE)))
```

```
The testing accuracy is given by: 99.615067
```

6 Support Vector Machines (with SGD)

Next, you will implement a "soft margin" SVM. In this formulation you will maximize the margin between positive and negative training examples and penalize margin violations using a hinge loss.

We will optimize the SVM loss using SGD. This means you must compute the loss function with respect to model weights. You will use this gradient to update the model weights.

SVM optimized with SGD has 3 hyperparameters that you can experiment with: - **Learning rate** - similar to as defined above in Perceptron, this parameter scales by how much the weights are changed according to the calculated gradient update. - **Epochs** - similar to as defined above in Perceptron. - **Regularization constant** - Hyperparameter to determine the strength of regularization. In this case it is a coefficient on the term which maximizes the margin. You could try different values. The default value is set to 0.05.

You will implement the SVM using SGD in the **models/svm.py**

The following code: - Creates an instance of the SVM classifier class - The train function of the SVM class is trained on the training data - We use the predict function to find the training accuracy as well as the testing accuracy

6.1 Train SVM on Fashion-MNIST

[20]:
```
### Experiment ###
lr = 500
n_epochs = 80
reg_const = 0.001

x = np.linspace(1, n_epochs, n_epochs)
y = np.zeros(n_epochs)

svm_fashion = SVM(n_class_fashion, lr, 1, reg_const)
for epoch in range(n_epochs):
  print("training epoch {}:".format(epoch))
  svm_fashion.train(X_train_fashion, y_train_fashion)
  pred_svm = svm_fashion.predict(X_val_fashion)
  print('The validation accuracy is: %f' % (get_acc(pred_svm, y_val_fashion)))
  y[epoch] = get_acc(pred_svm, y_val_fashion)
  if get_acc(pred_svm, y_val_fashion) > 81.7:
    break

plt.plot(x, y, 'r-.p')
plt.show()
```

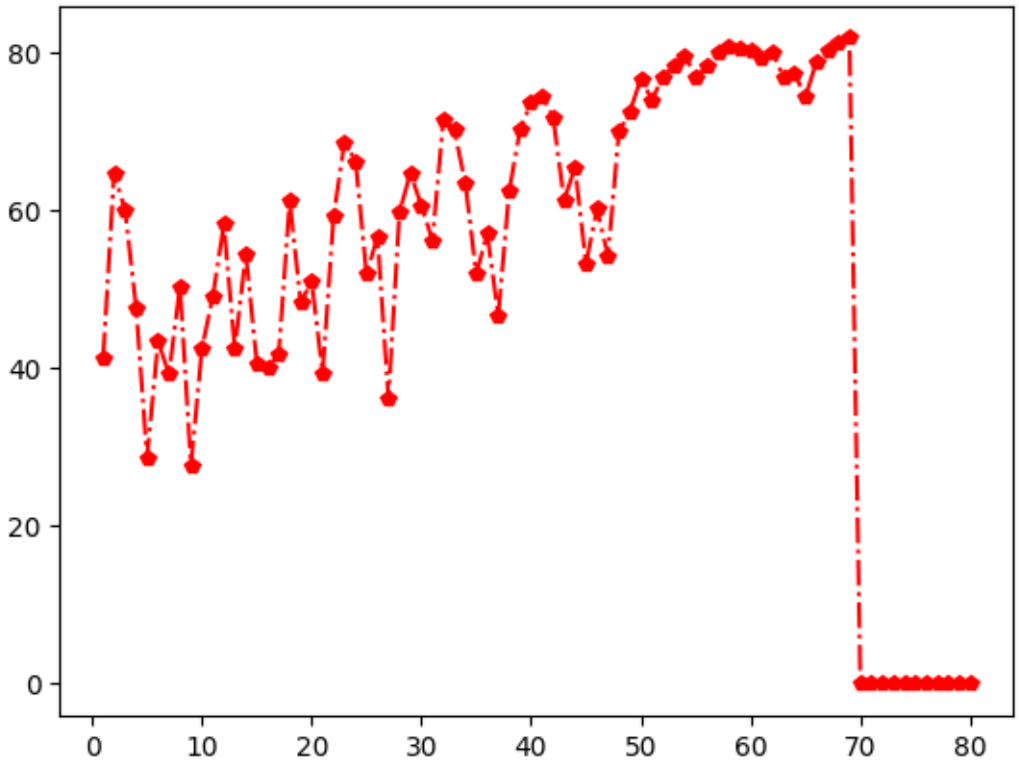
```
training epoch 0:
The training accuracy is 40.716
The validation accuracy is: 41.300000
training epoch 1:
The training accuracy is 65.032
The validation accuracy is: 64.760000
training epoch 2:
The training accuracy is 60.751999999999995
The validation accuracy is: 59.950000
training epoch 3:
The training accuracy is 47.446
The validation accuracy is: 47.720000
training epoch 4:
The training accuracy is 29.24
The validation accuracy is: 28.680000
training epoch 5:
The training accuracy is 44.152
The validation accuracy is: 43.370000
training epoch 6:
The training accuracy is 39.391999999999996
The validation accuracy is: 39.230000
```

```
training epoch 7:
The training accuracy is 50.246
The validation accuracy is: 50.370000
training epoch 8:
The training accuracy is 27.886
The validation accuracy is: 27.650000
training epoch 9:
The training accuracy is 42.344
The validation accuracy is: 42.490000
training epoch 10:
The training accuracy is 49.402
The validation accuracy is: 49.130000
training epoch 11:
The training accuracy is 57.644
The validation accuracy is: 58.200000
training epoch 12:
The training accuracy is 43.442
The validation accuracy is: 42.420000
training epoch 13:
The training accuracy is 54.449999999999996
The validation accuracy is: 54.490000
training epoch 14:
The training accuracy is 41.124
The validation accuracy is: 40.610000
training epoch 15:
The training accuracy is 39.934
The validation accuracy is: 40.100000
training epoch 16:
The training accuracy is 41.958
The validation accuracy is: 41.780000
training epoch 17:
The training accuracy is 61.288
The validation accuracy is: 61.200000
training epoch 18:
The training accuracy is 48.404
The validation accuracy is: 48.310000
training epoch 19:
The training accuracy is 51.158
The validation accuracy is: 50.930000
training epoch 20:
The training accuracy is 39.648
The validation accuracy is: 39.380000
training epoch 21:
The training accuracy is 60.150000000000006
The validation accuracy is: 59.270000
training epoch 22:
The training accuracy is 69.162
The validation accuracy is: 68.660000
```

```
training epoch 23:
The training accuracy is 66.52
The validation accuracy is: 66.220000
training epoch 24:
The training accuracy is 52.172
The validation accuracy is: 51.890000
training epoch 25:
The training accuracy is 56.864000000000004
The validation accuracy is: 56.650000
training epoch 26:
The training accuracy is 36.58
The validation accuracy is: 36.120000
training epoch 27:
The training accuracy is 60.232
The validation accuracy is: 59.770000
training epoch 28:
The training accuracy is 65.632
The validation accuracy is: 64.730000
training epoch 29:
The training accuracy is 60.88
The validation accuracy is: 60.540000
training epoch 30:
The training accuracy is 57.504
The validation accuracy is: 56.220000
training epoch 31:
The training accuracy is 71.76
The validation accuracy is: 71.390000
training epoch 32:
The training accuracy is 71.004
The validation accuracy is: 70.150000
training epoch 33:
The training accuracy is 63.861999999999995
The validation accuracy is: 63.420000
training epoch 34:
The training accuracy is 52.23799999999999
The validation accuracy is: 51.910000
training epoch 35:
The training accuracy is 57.065999999999995
The validation accuracy is: 57.030000
training epoch 36:
The training accuracy is 46.844
The validation accuracy is: 46.680000
training epoch 37:
The training accuracy is 62.739999999999995
The validation accuracy is: 62.460000
training epoch 38:
The training accuracy is 71.06
The validation accuracy is: 70.250000
```

```
training epoch 39:
The training accuracy is 73.996
The validation accuracy is: 73.580000
training epoch 40:
The training accuracy is 74.95
The validation accuracy is: 74.460000
training epoch 41:
The training accuracy is 72.47200000000001
The validation accuracy is: 71.770000
training epoch 42:
The training accuracy is 61.907999999999994
The validation accuracy is: 61.300000
training epoch 43:
The training accuracy is 66.13
The validation accuracy is: 65.390000
training epoch 44:
The training accuracy is 53.644000000000005
The validation accuracy is: 53.230000
training epoch 45:
The training accuracy is 60.773999999999994
The validation accuracy is: 60.150000
training epoch 46:
The training accuracy is 54.568000000000005
The validation accuracy is: 54.250000
training epoch 47:
The training accuracy is 70.87400000000001
The validation accuracy is: 69.910000
training epoch 48:
The training accuracy is 73.68400000000001
The validation accuracy is: 72.470000
training epoch 49:
The training accuracy is 77.338
The validation accuracy is: 76.670000
training epoch 50:
The training accuracy is 74.876
The validation accuracy is: 73.990000
training epoch 51:
The training accuracy is 76.98
The validation accuracy is: 76.780000
training epoch 52:
The training accuracy is 79.318
The validation accuracy is: 78.370000
training epoch 53:
The training accuracy is 80.464
The validation accuracy is: 79.550000
training epoch 54:
The training accuracy is 77.69200000000001
The validation accuracy is: 76.890000
```

```
training epoch 55:
The training accuracy is 78.846
The validation accuracy is: 78.190000
training epoch 56:
The training accuracy is 81.07799999999999
The validation accuracy is: 80.110000
training epoch 57:
The training accuracy is 81.316
The validation accuracy is: 80.650000
training epoch 58:
The training accuracy is 81.136
The validation accuracy is: 80.540000
training epoch 59:
The training accuracy is 80.83399999999999
The validation accuracy is: 80.350000
training epoch 60:
The training accuracy is 80.174
The validation accuracy is: 79.250000
training epoch 61:
The training accuracy is 80.484
The validation accuracy is: 80.040000
training epoch 62:
The training accuracy is 77.726
The validation accuracy is: 76.800000
training epoch 63:
The training accuracy is 77.656
The validation accuracy is: 77.250000
training epoch 64:
The training accuracy is 75.198
The validation accuracy is: 74.390000
training epoch 65:
The training accuracy is 79.11200000000001
The validation accuracy is: 78.770000
training epoch 66:
The training accuracy is 81.37
The validation accuracy is: 80.350000
training epoch 67:
The training accuracy is 82.08
The validation accuracy is: 81.240000
training epoch 68:
The training accuracy is 82.584
The validation accuracy is: 81.820000
```



[21]:
```
opt_epoches = np.argmax(y) + 1
opt_acc = np.max(y)
print("optimal n_epoches =", opt_epoches)
print("optimal valid accuracy =", opt_acc)
```

```
optimal n_epoches = 69
optimal valid accuracy = 81.82000000000001
```

[22]:
```
### Output Optimal
svm_fashion = SVM(n_class_fashion, lr, opt_epoches, reg_const)
svm_fashion.train(X_train_fashion, y_train_fashion)
```

```
The training accuracy is 40.716
The training accuracy is 65.032
The training accuracy is 60.751999999999995
The training accuracy is 47.446
The training accuracy is 29.24
The training accuracy is 44.152
The training accuracy is 39.391999999999996
The training accuracy is 50.246
The training accuracy is 27.886
The training accuracy is 42.344
The training accuracy is 49.402
```

```
The training accuracy is 57.644
The training accuracy is 43.442
The training accuracy is 54.449999999999996
The training accuracy is 41.124
The training accuracy is 39.934
The training accuracy is 41.958
The training accuracy is 61.288
The training accuracy is 48.404
The training accuracy is 51.158
The training accuracy is 39.648
The training accuracy is 60.150000000000006
The training accuracy is 69.162
The training accuracy is 66.52
The training accuracy is 52.172
The training accuracy is 56.864000000000004
The training accuracy is 36.58
The training accuracy is 60.232
The training accuracy is 65.632
The training accuracy is 60.88
The training accuracy is 57.504
The training accuracy is 71.76
The training accuracy is 71.004
The training accuracy is 63.861999999999995
The training accuracy is 52.23799999999999
The training accuracy is 57.065999999999995
The training accuracy is 46.844
The training accuracy is 62.739999999999995
The training accuracy is 71.06
The training accuracy is 73.996
The training accuracy is 74.95
The training accuracy is 72.47200000000001
The training accuracy is 61.907999999999994
The training accuracy is 66.13
The training accuracy is 53.644000000000005
The training accuracy is 60.773999999999994
The training accuracy is 54.568000000000005
The training accuracy is 70.87400000000001
The training accuracy is 73.68400000000001
The training accuracy is 77.338
The training accuracy is 74.876
The training accuracy is 76.98
The training accuracy is 79.318
The training accuracy is 80.464
The training accuracy is 77.69200000000001
The training accuracy is 78.846
The training accuracy is 81.07799999999999
The training accuracy is 81.316
The training accuracy is 81.136
```

```
The training accuracy is 80.83399999999999
The training accuracy is 80.174
The training accuracy is 80.484
The training accuracy is 77.726
The training accuracy is 77.656
The training accuracy is 75.198
The training accuracy is 79.11200000000001
The training accuracy is 81.37
The training accuracy is 82.08
The training accuracy is 82.584
```

[23]:
```
pred_svm = svm_fashion.predict(X_train_fashion)
print('The training accuracy is given by: %f' % (get_acc(pred_svm,␣
 ↪y_train_fashion)))
```

```
The training accuracy is given by: 82.584000
```

6.1.1 Validate SVM on Fashion-MNIST

[24]:
```
pred_svm = svm_fashion.predict(X_val_fashion)
print('The validation accuracy is given by: %f' % (get_acc(pred_svm,␣
 ↪y_val_fashion)))
```

```
The validation accuracy is given by: 81.820000
```

6.1.2 Test SVM on Fashion-MNIST

[25]:
```
pred_svm = svm_fashion.predict(X_test_fashion)
print('The testing accuracy is given by: %f' % (get_acc(pred_svm,␣
 ↪y_test_fashion)))
```

```
The testing accuracy is given by: 81.370000
```

6.1.3 SVM_Fashion-MNIST Kaggle Submission

Once you are satisfied with your solution and test accuracy output a file to submit your test set predictions to the Kaggle for Assignment 1 Fashion-MNIST. Use the following code to do so:

[26]:
```
output_submission_csv('kaggle/svm_submission_fashion.csv', svm_fashion.
 ↪predict(X_test_fashion))
```

6.2 Train SVM on Rice

[27]:
```
lr = 0.5
n_epochs = 30
reg_const = 0.001

svm_RICE = SVM(n_class_RICE, lr, n_epochs, reg_const)
svm_RICE.train(X_train_RICE, y_train_RICE)
```

```
Doing binary classification.
The training accuracy is 16.689579323618368
Doing binary classification.
The training accuracy is 33.66327559343782
Doing binary classification.
The training accuracy is 72.55980203464394
Doing binary classification.
The training accuracy is 97.60791861424251
Doing binary classification.
The training accuracy is 99.2209696636422
Doing binary classification.
The training accuracy is 99.79836861882504
Doing binary classification.
The training accuracy is 99.91751443497388
Doing binary classification.
The training accuracy is 99.96333974887727
Doing binary classification.
The training accuracy is 99.98166987443864
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
Doing binary classification.
The training accuracy is 99.99083493721932
```

```
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
Doing binary classification.
The training accuracy is 100.0
```

[28]:
```
pred_svm = svm_RICE.predict(X_train_RICE)
print('The training accuracy is given by: %f' % (get_acc(pred_svm,␣
  ↪y_train_RICE)))
```

```
The training accuracy is given by: 100.000000
```

6.2.1 Validate SVM on Rice

[29]:
```
pred_svm = svm_RICE.predict(X_val_RICE)
print('The validation accuracy is given by: %f' % (get_acc(pred_svm,␣
  ↪y_val_RICE)))
```

```
The validation accuracy is given by: 99.917514
```

6.3 Test SVM on Rice

[30]:
```
pred_svm = svm_RICE.predict(X_test_RICE)
print('The testing accuracy is given by: %f' % (get_acc(pred_svm, y_test_RICE)))
```

```
The testing accuracy is given by: 100.000000
```

7 Softmax Classifier (with SGD)

Next, you will train a Softmax classifier. This classifier consists of a linear function of the input data followed by a softmax function which outputs a vector of dimension C (number of classes) for each data point. Each entry of the softmax output vector corresponds to a confidence in one of the

C classes, and like a probability distribution, the entries of the output vector sum to 1. We use a cross-entropy loss on this sotmax output to train the model.

Check the following link as an additional resource on softmax classification: http://cs231n.github.io/linear-classify/#softmax

Once again we will train the classifier with SGD. This means you need to compute the gradients of the softmax cross-entropy loss function according to the weights and update the weights using this gradient. Check the following link to help with implementing the gradient updates: https://deepnotes.io/softmax-crossentropy

The softmax classifier has 3 hyperparameters that you can experiment with: - **Learning rate** - As above, this controls how much the model weights are updated with respect to their gradient. - **Number of Epochs** - As described for perceptron. - **Regularization constant** - Hyperparameter to determine the strength of regularization. In this case, we minimize the L2 norm of the model weights as regularization, so the regularization constant is a coefficient on the L2 norm in the combined cross-entropy and regularization objective.

You will implement a softmax classifier using SGD in the **models/softmax.py**

The following code: - Creates an instance of the Softmax classifier class - The train function of the Softmax class is trained on the training data - We use the predict function to find the training accuracy as well as the testing accuracy

7.1 Train Softmax on Fashion-MNIST

[31]:
```
### Experiment ###
lr = 3.5
n_epochs = 200
reg_const = 0.001

import matplotlib.pyplot as plt

x = np.linspace(1, n_epochs, n_epochs)
y = np.zeros(n_epochs)

softmax_fashion = Softmax(n_class_fashion, lr, 1, reg_const)
for epoch in range(n_epochs):
  print("training epoch {}:".format(epoch))
  softmax_fashion.train(X_train_fashion, y_train_fashion)
  pred_softmax = softmax_fashion.predict(X_val_fashion)
  print('The validation accuracy is: %f' % (get_acc(pred_softmax,␣
 ↪y_val_fashion)))
  y[epoch] = get_acc(pred_softmax, y_val_fashion)
  if get_acc(pred_softmax, y_val_fashion) >= 84:
    break

plt.plot(x, y, 'r-.p')
plt.show()
```

```
training epoch 0:
The training accuracy is 36.608000000000004
The validation accuracy is: 36.910000
training epoch 1:
The training accuracy is 55.17999999999999
The validation accuracy is: 55.030000
training epoch 2:
The training accuracy is 53.1
The validation accuracy is: 53.420000
training epoch 3:
The training accuracy is 59.102
The validation accuracy is: 58.900000
training epoch 4:
The training accuracy is 62.388
The validation accuracy is: 61.980000
training epoch 5:
The training accuracy is 64.044
The validation accuracy is: 63.220000
training epoch 6:
The training accuracy is 66.01
The validation accuracy is: 65.460000
training epoch 7:
The training accuracy is 66.45
The validation accuracy is: 65.910000
training epoch 8:
The training accuracy is 63.798
The validation accuracy is: 63.790000
training epoch 9:
The training accuracy is 66.712
The validation accuracy is: 66.530000
training epoch 10:
The training accuracy is 72.066
The validation accuracy is: 71.750000
training epoch 11:
The training accuracy is 72.162
The validation accuracy is: 71.310000
training epoch 12:
The training accuracy is 75.008
The validation accuracy is: 74.520000
training epoch 13:
The training accuracy is 71.932
The validation accuracy is: 71.320000
training epoch 14:
The training accuracy is 72.46000000000001
The validation accuracy is: 71.730000
training epoch 15:
The training accuracy is 76.184
The validation accuracy is: 75.780000
```

```
training epoch 16:
The training accuracy is 77.06
The validation accuracy is: 76.520000
training epoch 17:
The training accuracy is 72.934
The validation accuracy is: 72.340000
training epoch 18:
The training accuracy is 67.94
The validation accuracy is: 67.680000
training epoch 19:
The training accuracy is 68.958
The validation accuracy is: 68.760000
training epoch 20:
The training accuracy is 70.92399999999999
The validation accuracy is: 70.290000
training epoch 21:
The training accuracy is 68.894
The validation accuracy is: 69.100000
training epoch 22:
The training accuracy is 68.786
The validation accuracy is: 67.930000
training epoch 23:
The training accuracy is 74.51
The validation accuracy is: 74.100000
training epoch 24:
The training accuracy is 74.534
The validation accuracy is: 73.620000
training epoch 25:
The training accuracy is 71.368
The validation accuracy is: 70.900000
training epoch 26:
The training accuracy is 75.688
The validation accuracy is: 75.210000
training epoch 27:
The training accuracy is 78.598
The validation accuracy is: 77.960000
training epoch 28:
The training accuracy is 77.77199999999999
The validation accuracy is: 76.950000
training epoch 29:
The training accuracy is 75.99000000000001
The validation accuracy is: 75.570000
training epoch 30:
The training accuracy is 75.66000000000001
The validation accuracy is: 74.750000
training epoch 31:
The training accuracy is 74.654
The validation accuracy is: 73.570000
```

```
training epoch 32:
The training accuracy is 73.63
The validation accuracy is: 73.160000
training epoch 33:
The training accuracy is 76.28
The validation accuracy is: 75.630000
training epoch 34:
The training accuracy is 79.012
The validation accuracy is: 78.440000
training epoch 35:
The training accuracy is 79.976
The validation accuracy is: 79.480000
training epoch 36:
The training accuracy is 80.824
The validation accuracy is: 80.140000
training epoch 37:
The training accuracy is 77.312
The validation accuracy is: 76.360000
training epoch 38:
The training accuracy is 77.102
The validation accuracy is: 76.610000
training epoch 39:
The training accuracy is 72.08200000000001
The validation accuracy is: 71.140000
training epoch 40:
The training accuracy is 74.824
The validation accuracy is: 74.550000
training epoch 41:
The training accuracy is 74.012
The validation accuracy is: 73.350000
training epoch 42:
The training accuracy is 76.92
The validation accuracy is: 76.220000
training epoch 43:
The training accuracy is 77.424
The validation accuracy is: 76.650000
training epoch 44:
The training accuracy is 78.5
The validation accuracy is: 77.970000
training epoch 45:
The training accuracy is 81.296
The validation accuracy is: 80.390000
training epoch 46:
The training accuracy is 81.19800000000001
The validation accuracy is: 80.260000
training epoch 47:
The training accuracy is 77.922
The validation accuracy is: 76.710000
```

```
training epoch 48:
The training accuracy is 78.896
The validation accuracy is: 78.500000
training epoch 49:
The training accuracy is 75.368
The validation accuracy is: 73.820000
training epoch 50:
The training accuracy is 75.196
The validation accuracy is: 74.620000
training epoch 51:
The training accuracy is 78.378
The validation accuracy is: 77.510000
training epoch 52:
The training accuracy is 81.11
The validation accuracy is: 80.490000
training epoch 53:
The training accuracy is 80.46600000000001
The validation accuracy is: 79.470000
training epoch 54:
The training accuracy is 78.55199999999999
The validation accuracy is: 78.000000
training epoch 55:
The training accuracy is 76.548
The validation accuracy is: 75.760000
training epoch 56:
The training accuracy is 77.13799999999999
The validation accuracy is: 76.010000
training epoch 57:
The training accuracy is 81.036
The validation accuracy is: 80.320000
training epoch 58:
The training accuracy is 81.43
The validation accuracy is: 80.650000
training epoch 59:
The training accuracy is 77.026
The validation accuracy is: 76.520000
training epoch 60:
The training accuracy is 74.66199999999999
The validation accuracy is: 73.880000
training epoch 61:
The training accuracy is 75.28
The validation accuracy is: 74.650000
training epoch 62:
The training accuracy is 78.40599999999999
The validation accuracy is: 77.690000
training epoch 63:
The training accuracy is 80.448
The validation accuracy is: 79.870000
```

```
training epoch 64:
The training accuracy is 78.472
The validation accuracy is: 77.420000
training epoch 65:
The training accuracy is 78.51599999999999
The validation accuracy is: 77.880000
training epoch 66:
The training accuracy is 77.578
The validation accuracy is: 76.560000
training epoch 67:
The training accuracy is 79.712
The validation accuracy is: 79.040000
training epoch 68:
The training accuracy is 80.05799999999999
The validation accuracy is: 78.930000
training epoch 69:
The training accuracy is 81.448
The validation accuracy is: 80.750000
training epoch 70:
The training accuracy is 78.456
The validation accuracy is: 77.510000
training epoch 71:
The training accuracy is 78.718
The validation accuracy is: 78.190000
training epoch 72:
The training accuracy is 76.244
The validation accuracy is: 75.490000
training epoch 73:
The training accuracy is 77.114
The validation accuracy is: 76.700000
training epoch 74:
The training accuracy is 82.126
The validation accuracy is: 81.400000
training epoch 75:
The training accuracy is 81.22
The validation accuracy is: 80.430000
training epoch 76:
The training accuracy is 80.82000000000001
The validation accuracy is: 79.810000
training epoch 77:
The training accuracy is 80.428
The validation accuracy is: 79.570000
training epoch 78:
The training accuracy is 76.91
The validation accuracy is: 75.960000
training epoch 79:
The training accuracy is 75.534
The validation accuracy is: 75.050000
```

```
training epoch 80:
The training accuracy is 82.006
The validation accuracy is: 81.490000
training epoch 81:
The training accuracy is 82.364
The validation accuracy is: 81.330000
training epoch 82:
The training accuracy is 81.43599999999999
The validation accuracy is: 80.760000
training epoch 83:
The training accuracy is 82.15400000000001
The validation accuracy is: 80.990000
training epoch 84:
The training accuracy is 82.116
The validation accuracy is: 81.470000
training epoch 85:
The training accuracy is 83.02000000000001
The validation accuracy is: 81.940000
training epoch 86:
The training accuracy is 81.496
The validation accuracy is: 80.710000
training epoch 87:
The training accuracy is 79.616
The validation accuracy is: 78.880000
training epoch 88:
The training accuracy is 78.684
The validation accuracy is: 77.940000
training epoch 89:
The training accuracy is 81.244
The validation accuracy is: 80.480000
training epoch 90:
The training accuracy is 79.95
The validation accuracy is: 78.960000
training epoch 91:
The training accuracy is 81.604
The validation accuracy is: 80.960000
training epoch 92:
The training accuracy is 78.542
The validation accuracy is: 77.370000
training epoch 93:
The training accuracy is 79.364
The validation accuracy is: 79.060000
training epoch 94:
The training accuracy is 81.388
The validation accuracy is: 80.350000
training epoch 95:
The training accuracy is 82.53399999999999
The validation accuracy is: 81.870000
```

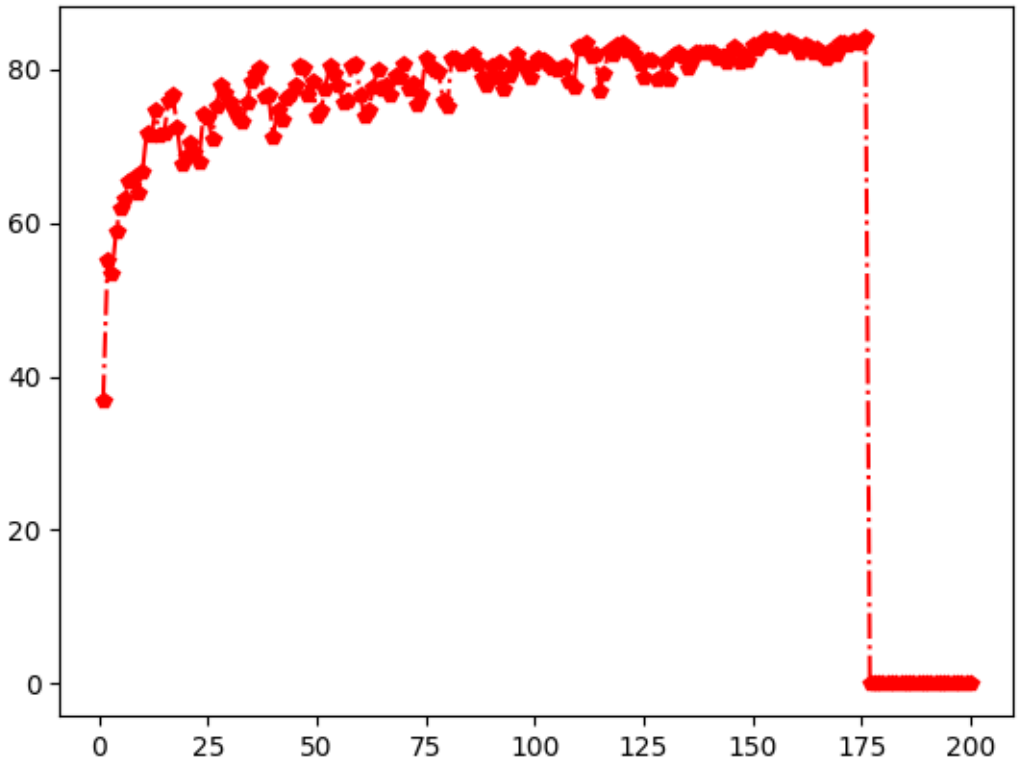
```
training epoch 96:
The training accuracy is 81.684
The validation accuracy is: 80.700000
training epoch 97:
The training accuracy is 80.47800000000001
The validation accuracy is: 79.810000
training epoch 98:
The training accuracy is 80.10199999999999
The validation accuracy is: 78.930000
training epoch 99:
The training accuracy is 81.21199999999999
The validation accuracy is: 80.610000
training epoch 100:
The training accuracy is 82.416
The validation accuracy is: 81.380000
training epoch 101:
The training accuracy is 81.838
The validation accuracy is: 81.070000
training epoch 102:
The training accuracy is 81.53399999999999
The validation accuracy is: 80.570000
training epoch 103:
The training accuracy is 80.58
The validation accuracy is: 80.150000
training epoch 104:
The training accuracy is 80.732
The validation accuracy is: 80.000000
training epoch 105:
The training accuracy is 80.622
The validation accuracy is: 80.180000
training epoch 106:
The training accuracy is 81.04599999999999
The validation accuracy is: 80.310000
training epoch 107:
The training accuracy is 79.14999999999999
The validation accuracy is: 78.450000
training epoch 108:
The training accuracy is 77.936
The validation accuracy is: 77.520000
training epoch 109:
The training accuracy is 83.756
The validation accuracy is: 82.810000
training epoch 110:
The training accuracy is 83.682
The validation accuracy is: 82.730000
training epoch 111:
The training accuracy is 84.11800000000001
The validation accuracy is: 83.280000
```

```
training epoch 112:
The training accuracy is 82.39
The validation accuracy is: 81.560000
training epoch 113:
The training accuracy is 82.324
The validation accuracy is: 81.860000
training epoch 114:
The training accuracy is 77.754
The validation accuracy is: 77.090000
training epoch 115:
The training accuracy is 79.688
The validation accuracy is: 79.400000
training epoch 116:
The training accuracy is 83.536
The validation accuracy is: 82.420000
training epoch 117:
The training accuracy is 82.91
The validation accuracy is: 82.010000
training epoch 118:
The training accuracy is 83.966
The validation accuracy is: 83.190000
training epoch 119:
The training accuracy is 84.148
The validation accuracy is: 83.300000
training epoch 120:
The training accuracy is 83.61800000000001
The validation accuracy is: 82.340000
training epoch 121:
The training accuracy is 83.48400000000001
The validation accuracy is: 82.660000
training epoch 122:
The training accuracy is 82.78999999999999
The validation accuracy is: 81.890000
training epoch 123:
The training accuracy is 81.53
The validation accuracy is: 80.690000
training epoch 124:
The training accuracy is 79.51
The validation accuracy is: 78.850000
training epoch 125:
The training accuracy is 82.004
The validation accuracy is: 81.190000
training epoch 126:
The training accuracy is 82.03
The validation accuracy is: 81.030000
training epoch 127:
The training accuracy is 79.654
The validation accuracy is: 78.730000
```

```
training epoch 128:
The training accuracy is 79.84599999999999
The validation accuracy is: 78.950000
training epoch 129:
The training accuracy is 81.648
The validation accuracy is: 80.900000
training epoch 130:
The training accuracy is 79.69800000000001
The validation accuracy is: 78.670000
training epoch 131:
The training accuracy is 82.458
The validation accuracy is: 81.780000
training epoch 132:
The training accuracy is 83.134
The validation accuracy is: 82.080000
training epoch 133:
The training accuracy is 82.318
The validation accuracy is: 81.610000
training epoch 134:
The training accuracy is 81.416
The validation accuracy is: 80.230000
training epoch 135:
The training accuracy is 81.774
The validation accuracy is: 81.040000
training epoch 136:
The training accuracy is 83.256
The validation accuracy is: 82.200000
training epoch 137:
The training accuracy is 83.312
The validation accuracy is: 82.150000
training epoch 138:
The training accuracy is 83.166
The validation accuracy is: 82.190000
training epoch 139:
The training accuracy is 83.356
The validation accuracy is: 82.260000
training epoch 140:
The training accuracy is 83.018
The validation accuracy is: 82.020000
training epoch 141:
The training accuracy is 82.384
The validation accuracy is: 81.480000
training epoch 142:
The training accuracy is 82.578
The validation accuracy is: 81.580000
training epoch 143:
The training accuracy is 82.12
The validation accuracy is: 81.000000
```

```
training epoch 144:
The training accuracy is 83.09
The validation accuracy is: 82.220000
training epoch 145:
The training accuracy is 83.974
The validation accuracy is: 82.850000
training epoch 146:
The training accuracy is 81.83
The validation accuracy is: 80.770000
training epoch 147:
The training accuracy is 82.542
The validation accuracy is: 81.900000
training epoch 148:
The training accuracy is 82.408
The validation accuracy is: 81.160000
training epoch 149:
The training accuracy is 83.994
The validation accuracy is: 83.060000
training epoch 150:
The training accuracy is 83.67599999999999
The validation accuracy is: 82.590000
training epoch 151:
The training accuracy is 84.446
The validation accuracy is: 83.500000
training epoch 152:
The training accuracy is 84.868
The validation accuracy is: 83.950000
training epoch 153:
The training accuracy is 84.454
The validation accuracy is: 83.670000
training epoch 154:
The training accuracy is 84.504
The validation accuracy is: 83.790000
training epoch 155:
The training accuracy is 84.25200000000001
The validation accuracy is: 83.510000
training epoch 156:
The training accuracy is 83.664
The validation accuracy is: 82.960000
training epoch 157:
The training accuracy is 84.402
The validation accuracy is: 83.530000
training epoch 158:
The training accuracy is 84.074
The validation accuracy is: 83.290000
training epoch 159:
The training accuracy is 84.304
The validation accuracy is: 83.210000
```

```
training epoch 160:
The training accuracy is 83.172
The validation accuracy is: 82.060000
training epoch 161:
The training accuracy is 84.098
The validation accuracy is: 83.060000
training epoch 162:
The training accuracy is 84.06
The validation accuracy is: 82.910000
training epoch 163:
The training accuracy is 83.042
The validation accuracy is: 82.040000
training epoch 164:
The training accuracy is 83.818
The validation accuracy is: 82.720000
training epoch 165:
The training accuracy is 82.94399999999999
The validation accuracy is: 81.850000
training epoch 166:
The training accuracy is 82.376
The validation accuracy is: 81.330000
training epoch 167:
The training accuracy is 83.226
The validation accuracy is: 82.280000
training epoch 168:
The training accuracy is 82.748
The validation accuracy is: 81.910000
training epoch 169:
The training accuracy is 84.276
The validation accuracy is: 83.310000
training epoch 170:
The training accuracy is 84.47200000000001
The validation accuracy is: 83.450000
training epoch 171:
The training accuracy is 84.402
The validation accuracy is: 83.150000
training epoch 172:
The training accuracy is 84.82600000000001
The validation accuracy is: 83.740000
training epoch 173:
The training accuracy is 84.53399999999999
The validation accuracy is: 83.460000
training epoch 174:
The training accuracy is 84.02
The validation accuracy is: 83.400000
training epoch 175:
The training accuracy is 84.958
The validation accuracy is: 84.020000
```



[32]:
```
opt_epoches = np.argmax(y) + 1
opt_acc = np.max(y)
print("optimal n_epoches =", opt_epoches)
print("optimal valid accuracy =", opt_acc)
```

```
optimal n_epoches = 176
optimal valid accuracy = 84.02
```

[33]:
```
### Output Optimal
softmax_fashion = Softmax(n_class_fashion, lr, opt_epoches, reg_const)
softmax_fashion.train(X_train_fashion, y_train_fashion)
```

```
The training accuracy is 36.608000000000004
The training accuracy is 55.17999999999999
The training accuracy is 53.1
The training accuracy is 59.102
The training accuracy is 62.388
The training accuracy is 64.044
The training accuracy is 66.01
The training accuracy is 66.45
The training accuracy is 63.798
The training accuracy is 66.712
The training accuracy is 72.066
```

```
The training accuracy is 72.162
The training accuracy is 75.008
The training accuracy is 71.932
The training accuracy is 72.46000000000001
The training accuracy is 76.184
The training accuracy is 77.06
The training accuracy is 72.934
The training accuracy is 67.94
The training accuracy is 68.958
The training accuracy is 70.92399999999999
The training accuracy is 68.894
The training accuracy is 68.786
The training accuracy is 74.51
The training accuracy is 74.534
The training accuracy is 71.368
The training accuracy is 75.688
The training accuracy is 78.598
The training accuracy is 77.77199999999999
The training accuracy is 75.99000000000001
The training accuracy is 75.66000000000001
The training accuracy is 74.654
The training accuracy is 73.63
The training accuracy is 76.28
The training accuracy is 79.012
The training accuracy is 79.976
The training accuracy is 80.824
The training accuracy is 77.312
The training accuracy is 77.102
The training accuracy is 72.08200000000001
The training accuracy is 74.824
The training accuracy is 74.012
The training accuracy is 76.92
The training accuracy is 77.424
The training accuracy is 78.5
The training accuracy is 81.296
The training accuracy is 81.19800000000001
The training accuracy is 77.922
The training accuracy is 78.896
The training accuracy is 75.368
The training accuracy is 75.196
The training accuracy is 78.378
The training accuracy is 81.11
The training accuracy is 80.46600000000001
The training accuracy is 78.55199999999999
The training accuracy is 76.548
The training accuracy is 77.13799999999999
The training accuracy is 81.036
The training accuracy is 81.43
```

```
The training accuracy is 77.026
The training accuracy is 74.66199999999999
The training accuracy is 75.28
The training accuracy is 78.40599999999999
The training accuracy is 80.448
The training accuracy is 78.472
The training accuracy is 78.51599999999999
The training accuracy is 77.578
The training accuracy is 79.712
The training accuracy is 80.05799999999999
The training accuracy is 81.448
The training accuracy is 78.456
The training accuracy is 78.718
The training accuracy is 76.244
The training accuracy is 77.114
The training accuracy is 82.126
The training accuracy is 81.22
The training accuracy is 80.82000000000001
The training accuracy is 80.428
The training accuracy is 76.91
The training accuracy is 75.534
The training accuracy is 82.006
The training accuracy is 82.364
The training accuracy is 81.43599999999999
The training accuracy is 82.15400000000001
The training accuracy is 82.116
The training accuracy is 83.02000000000001
The training accuracy is 81.496
The training accuracy is 79.616
The training accuracy is 78.684
The training accuracy is 81.244
The training accuracy is 79.95
The training accuracy is 81.604
The training accuracy is 78.542
The training accuracy is 79.364
The training accuracy is 81.388
The training accuracy is 82.53399999999999
The training accuracy is 81.684
The training accuracy is 80.47800000000001
The training accuracy is 80.10199999999999
The training accuracy is 81.21199999999999
The training accuracy is 82.416
The training accuracy is 81.838
The training accuracy is 81.53399999999999
The training accuracy is 80.58
The training accuracy is 80.732
The training accuracy is 80.622
The training accuracy is 81.04599999999999
```

```
The training accuracy is 79.14999999999999
The training accuracy is 77.936
The training accuracy is 83.756
The training accuracy is 83.682
The training accuracy is 84.11800000000001
The training accuracy is 82.39
The training accuracy is 82.324
The training accuracy is 77.754
The training accuracy is 79.688
The training accuracy is 83.536
The training accuracy is 82.91
The training accuracy is 83.966
The training accuracy is 84.148
The training accuracy is 83.61800000000001
The training accuracy is 83.48400000000001
The training accuracy is 82.78999999999999
The training accuracy is 81.53
The training accuracy is 79.51
The training accuracy is 82.004
The training accuracy is 82.03
The training accuracy is 79.654
The training accuracy is 79.84599999999999
The training accuracy is 81.648
The training accuracy is 79.69800000000001
The training accuracy is 82.458
The training accuracy is 83.134
The training accuracy is 82.318
The training accuracy is 81.416
The training accuracy is 81.774
The training accuracy is 83.256
The training accuracy is 83.312
The training accuracy is 83.166
The training accuracy is 83.356
The training accuracy is 83.018
The training accuracy is 82.384
The training accuracy is 82.578
The training accuracy is 82.12
The training accuracy is 83.09
The training accuracy is 83.974
The training accuracy is 81.83
The training accuracy is 82.542
The training accuracy is 82.408
The training accuracy is 83.994
The training accuracy is 83.67599999999999
The training accuracy is 84.446
The training accuracy is 84.868
The training accuracy is 84.454
The training accuracy is 84.504
```

```
The training accuracy is 84.25200000000001
The training accuracy is 83.664
The training accuracy is 84.402
The training accuracy is 84.074
The training accuracy is 84.304
The training accuracy is 83.172
The training accuracy is 84.098
The training accuracy is 84.06
The training accuracy is 83.042
The training accuracy is 83.818
The training accuracy is 82.94399999999999
The training accuracy is 82.376
The training accuracy is 83.226
The training accuracy is 82.748
The training accuracy is 84.276
The training accuracy is 84.47200000000001
The training accuracy is 84.402
The training accuracy is 84.82600000000001
The training accuracy is 84.53399999999999
The training accuracy is 84.02
The training accuracy is 84.958
```

[34]:
```
pred_softmax = softmax_fashion.predict(X_train_fashion)
print('The training accuracy is given by: %f' % (get_acc(pred_softmax,␣
 ↪y_train_fashion)))
```

```
The training accuracy is given by: 84.958000
```

7.1.1 Validate Softmax on Fashion-MNIST

[35]:
```
pred_softmax = softmax_fashion.predict(X_val_fashion)
print('The validation accuracy is given by: %f' % (get_acc(pred_softmax,␣
 ↪y_val_fashion)))
```

```
The validation accuracy is given by: 84.020000
```

7.1.2 Testing Softmax on Fashion-MNIST

[36]:
```
pred_softmax = softmax_fashion.predict(X_test_fashion)
print('The testing accuracy is given by: %f' % (get_acc(pred_softmax,␣
 ↪y_test_fashion)))
```

```
The testing accuracy is given by: 83.030000
```

7.1.3 Softmax_Fashion-MNIST Kaggle Submission

Once you are satisfied with your solution and test accuracy output a file to submit your test set predictions to the Kaggle for Assignment 1 Fashion-MNIST. Use the following code to do so:

[37]:
```
output_submission_csv('kaggle/softmax_submission_fashion.csv', softmax_fashion.
  ↪predict(X_test_fashion))
```

7.2 Train Softmax on Rice

[38]:
```
lr = 0.0005
n_epochs = 10
reg_const = 0.001

softmax_RICE = Softmax(n_class_RICE, lr, n_epochs, reg_const)
softmax_RICE.train(X_train_RICE, y_train_RICE)
```

```
Doing binary classification.
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
The training accuracy is 90.83493721931995
```

[39]:
```
pred_softmax = softmax_RICE.predict(X_train_RICE)
print('The training accuracy is given by: %f' % (get_acc(pred_softmax,␣
  ↪y_train_RICE)))
```

```
The training accuracy is given by: 90.834937
```

7.2.1 Validate Softmax on Rice

[40]:
```
pred_softmax = softmax_RICE.predict(X_val_RICE)
print('The validation accuracy is given by: %f' % (get_acc(pred_softmax,␣
  ↪y_val_RICE)))
```

```
The validation accuracy is given by: 90.184218
```

7.2.2 Testing Softmax on Rice

[41]:
```
pred_softmax = softmax_RICE.predict(X_test_RICE)
print('The testing accuracy is given by: %f' % (get_acc(pred_softmax,␣
  ↪y_test_RICE)))
```

```
The testing accuracy is given by: 92.438823
```

8 Logistic Classifier

The Logistic Classifier has 2 hyperparameters that you can experiment with: - **Learning rate** - similar to as defined above in Perceptron, this parameter scales by how much the weights are changed according to the calculated gradient update. - **Number of Epochs** - As described for perceptron. - **Threshold** - The decision boundary of the classifier.

You will implement the Logistic Classifier in the **models/logistic.py**

The following code: - Creates an instance of the Logistic classifier class - The train function of the Logistic class is trained on the training data - We use the predict function to find the training accuracy as well as the testing accuracy

8.0.1 Training Logistic Classifer

[42]:
```
### Experiment ###
learning_rate = 0.000005
n_epochs = 10
threshold = 0.1

lr = Logistic(learning_rate, n_epochs, threshold)
lr.train(X_train_RICE, y_train_RICE)
```

```
/content/drive/MyDrive/CS444/assignment1/models/logistic.py:30: RuntimeWarning:
overflow encountered in exp
  return 1 / (1 + np.exp(-z))

The training accuracy is 68.89377692237191
The training accuracy is 65.60351938410778
The training accuracy is 79.81853175694253
The training accuracy is 95.45412886078269
The training accuracy is 94.26267069929429
The training accuracy is 94.7209238383283
The training accuracy is 96.78306296398131
The training accuracy is 97.80954999541747
The training accuracy is 98.83603702685363
The training accuracy is 99.3584456053524
```

[43]:
```
learning_rate = 0.5
n_epochs = 10
threshold = 0.5

lr = Logistic(learning_rate, n_epochs, threshold)
lr.train(X_train_RICE, y_train_RICE)
```

```
The training accuracy is 69.36119512418661
The training accuracy is 66.07093758592247
The training accuracy is 70.65346897626249
The training accuracy is 71.1483823664192
The training accuracy is 94.83090459169645
```

```
The training accuracy is 97.59875355146183
The training accuracy is 99.20263953808085
The training accuracy is 99.2209696636422
The training accuracy is 98.90935752909907
The training accuracy is 97.20465585189258
```

[44]:
```
pred_lr = lr.predict(X_train_RICE)
print('The training accuracy is given by: %f' % (get_acc(pred_lr,␣
 ↪y_train_RICE)))
```

```
The training accuracy is given by: 97.204656
```

8.0.2 Validate Logistic Classifer

[45]:
```
pred_lr = lr.predict(X_val_RICE)
print('The validation accuracy is given by: %f' % (get_acc(pred_lr,␣
 ↪y_val_RICE)))
```

```
The validation accuracy is given by: 97.360462
```

8.0.3 Test Logistic Classifier

[46]:
```
pred_lr = lr.predict(X_test_RICE)
print('The testing accuracy is given by: %f' % (get_acc(pred_lr, y_test_RICE)))
```

```
The testing accuracy is given by: 97.360462
```

Generate PDF

[47]:
```
%%capture

from google.colab import drive
drive.mount('/content/drive')
# install tex; first run may take several minutes
! apt-get install texlive-xetex
# file path and save location below are default; please change if they do not␣
 ↪match yours
! jupyter nbconvert --output-dir='/content/drive/MyDrive/' '/content/drive/
 ↪MyDrive/Colab Notebooks/CS444_Assignment1.ipynb' --to pdf
```