Imports

In [1]:
```
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
```

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

In [2]:
```
# You can change these numbers for experimentation
# For submission we will use the default values
TRAIN_IMAGES = 50000
VAL_IMAGES = 10000
normalize = True
```

In [3]:
```
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))
```

Loading Rice

In [4]:
```
# 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
```

Get Accuracy

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

In [5]:
```
def get_acc(pred, y_test):
    return np.sum(y_test == pred) / len(y_test) * 100
```

Perceptron

Perceptron has 2 hyperparameters that you can experiment with:

- **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. We recommend changing the learning rate by factors of 10 and observing how the performance of the classifier changes. You should also try adding a **decay** which slowly reduces the learning rate over each epoch.
- **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

Train Perceptron on Fashion-MNIST

In [6]:
```
lr = 0.5
n_epochs = 10

percept_fashion = Perceptron(n_class_fashion, lr, n_epochs)
percept_fashion.train(X_train_fashion, y_train_fashion)
```

```
Epoch 0 Accuracy 10.646
Epoch 1 Accuracy 83.71
Epoch 2 Accuracy 84.006
Epoch 3 Accuracy 84.08200000000001
Epoch 4 Accuracy 84.08200000000001
Epoch 5 Accuracy 84.08200000000001
Epoch 6 Accuracy 84.08200000000001
Epoch 7 Accuracy 84.08200000000001
Epoch 8 Accuracy 84.08200000000001
Epoch 9 Accuracy 84.08200000000001
```

In [7]:
```
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: 84.082000
```

Validate Perceptron on Fashion-MNIST

In [8]:
```
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: 82.020000
```

Test Perceptron on Fashion-MNIST

In [9]:
```
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: 81.880000
```

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:

In [10]:
```
output_submission_csv('kaggle/perceptron_submission_fashion.csv', percept_fashion.pred
```

Train Perceptron on Rice

In [11]:
```
lr = 0.3
n_epochs = 20

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

```
Epoch 0 Accuracy 54.779580240124645
Epoch 1 Accuracy 94.94088534506461
Epoch 2 Accuracy 99.8166987443864
Epoch 3 Accuracy 99.89918430941252
Epoch 4 Accuracy 99.89918430941252
Epoch 5 Accuracy 99.89918430941252
Epoch 6 Accuracy 99.89918430941252
Epoch 7 Accuracy 99.89918430941252
Epoch 8 Accuracy 99.89918430941252
Epoch 9 Accuracy 99.89918430941252
Epoch 10 Accuracy 99.89918430941252
Epoch 11 Accuracy 99.89918430941252
Epoch 12 Accuracy 99.89918430941252
Epoch 13 Accuracy 99.89918430941252
Epoch 14 Accuracy 99.89918430941252
Epoch 15 Accuracy 99.89918430941252
Epoch 16 Accuracy 99.89918430941252
Epoch 17 Accuracy 99.89918430941252
Epoch 18 Accuracy 99.89918430941252
Epoch 19 Accuracy 99.89918430941252
```

In [12]:
```
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: 99.899184
```

Validate Perceptron on Rice

In [13]:
```
pred_percept = percept_RICE.predict(X_val_RICE)
print('The validation accuracy is given by: %f' % (get_acc(pred_percept, y_val_RICE)))
```

```
The validation accuracy is given by: 99.835029
```

Test Perceptron on Rice

In [14]:
```
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: 99.835029
```

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

Train SVM on Fashion-MNIST

In [15]:
```
lr = 1
n_epochs = 30
reg_const = 0.05

svm_fashion = SVM(n_class_fashion, lr, n_epochs, reg_const)
svm_fashion.train(X_train_fashion, y_train_fashion)
```

```
Epoch 0 Accuracy 9.693999999999999
Epoch 1 Accuracy 77.282
Epoch 2 Accuracy 74.698
Epoch 3 Accuracy 82.054
Epoch 4 Accuracy 79.814
Epoch 5 Accuracy 83.834
Epoch 6 Accuracy 82.296
Epoch 7 Accuracy 83.97
Epoch 8 Accuracy 84.218
Epoch 9 Accuracy 84.372
Epoch 10 Accuracy 84.298
Epoch 11 Accuracy 84.24000000000001
Epoch 12 Accuracy 84.186
Epoch 13 Accuracy 84.314
Epoch 14 Accuracy 84.296
Epoch 15 Accuracy 84.38600000000001
Epoch 16 Accuracy 84.348
Epoch 17 Accuracy 84.294
Epoch 18 Accuracy 84.298
Epoch 19 Accuracy 84.32600000000001
Epoch 20 Accuracy 84.32
Epoch 21 Accuracy 84.32
Epoch 22 Accuracy 84.336
Epoch 23 Accuracy 84.328
Epoch 24 Accuracy 84.322
Epoch 25 Accuracy 84.316
Epoch 26 Accuracy 84.316
Epoch 27 Accuracy 84.314
Epoch 28 Accuracy 84.31
Epoch 29 Accuracy 84.308
```

In [16]:
```
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: 84.308000
```

Validate SVM on Fashion-MNIST

In [17]:
```
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: 82.840000
```

Test SVM on Fashion-MNIST

In [18]:
```
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: 82.140000
```

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:

In [19]:
```
output_submission_csv('kaggle/svm_submission_fashion.csv', svm_fashion.predict(X_test_
```

Train SVM on Rice

In [20]:
```
lr = 1
n_epochs = 50
reg_const = 0.05

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

```
Epoch 0 Accuracy 54.779580240124645
Epoch 1 Accuracy 59.160480249289705
Epoch 2 Accuracy 62.423242599211804
Epoch 3 Accuracy 74.60361103473558
Epoch 4 Accuracy 77.49977087343048
Epoch 5 Accuracy 79.08532673448812
Epoch 6 Accuracy 78.41627715149849
Epoch 7 Accuracy 78.7278892860416
Epoch 8 Accuracy 78.27880120978828
Epoch 9 Accuracy 79.0944917972688
Epoch 10 Accuracy 78.98451104390065
Epoch 11 Accuracy 78.94785079277793
Epoch 12 Accuracy 79.13115204839153
Epoch 13 Accuracy 79.07616167170745
Epoch 14 Accuracy 79.08532673448812
Epoch 15 Accuracy 79.05783154614609
Epoch 16 Accuracy 79.00284116946202
Epoch 17 Accuracy 79.00284116946202
Epoch 18 Accuracy 79.00284116946202
Epoch 19 Accuracy 79.0120062322427
Epoch 20 Accuracy 79.03950142058474
Epoch 21 Accuracy 79.04866648336541
Epoch 22 Accuracy 79.04866648336541
Epoch 23 Accuracy 79.04866648336541
Epoch 24 Accuracy 79.04866648336541
Epoch 25 Accuracy 79.04866648336541
Epoch 26 Accuracy 79.04866648336541
Epoch 27 Accuracy 79.03033635780406
Epoch 28 Accuracy 79.04866648336541
Epoch 29 Accuracy 79.04866648336541
Epoch 30 Accuracy 79.04866648336541
Epoch 31 Accuracy 79.04866648336541
Epoch 32 Accuracy 79.04866648336541
Epoch 33 Accuracy 79.04866648336541
Epoch 34 Accuracy 79.04866648336541
Epoch 35 Accuracy 79.04866648336541
Epoch 36 Accuracy 79.04866648336541
Epoch 37 Accuracy 79.04866648336541
Epoch 38 Accuracy 79.04866648336541
Epoch 39 Accuracy 79.04866648336541
Epoch 40 Accuracy 79.04866648336541
Epoch 41 Accuracy 79.04866648336541
Epoch 42 Accuracy 79.04866648336541
Epoch 43 Accuracy 79.04866648336541
Epoch 44 Accuracy 79.04866648336541
Epoch 45 Accuracy 79.04866648336541
Epoch 46 Accuracy 79.04866648336541
Epoch 47 Accuracy 79.04866648336541
Epoch 48 Accuracy 79.04866648336541
Epoch 49 Accuracy 79.04866648336541
```

In [21]:

```
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: 79.048666
```

Validate SVM on Rice

In [22]:
```
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: 78.581248
```

Test SVM on Rice

In [23]:
```
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: 79.323618
```

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

Train Softmax on Fashion-MNIST

In [24]:
```
lr = 0.0005
n_epochs = 200
reg_const = 1

softmax_fashion = Softmax(n_class_fashion, lr, n_epochs, reg_const)
softmax_fashion.train(X_train_fashion, y_train_fashion)
```

```
Epoch 0 Accuracy 6.39
Epoch 1 Accuracy 75.994
Epoch 2 Accuracy 78.276
Epoch 3 Accuracy 78.798
Epoch 4 Accuracy 79.994
Epoch 5 Accuracy 80.05
Epoch 6 Accuracy 81.186
Epoch 7 Accuracy 79.10199999999999
Epoch 8 Accuracy 81.462
Epoch 9 Accuracy 82.592
Epoch 10 Accuracy 78.572
Epoch 11 Accuracy 82.238
Epoch 12 Accuracy 80.822
Epoch 13 Accuracy 81.734
Epoch 14 Accuracy 85.356
Epoch 15 Accuracy 83.084
Epoch 16 Accuracy 82.988
Epoch 17 Accuracy 83.248
Epoch 18 Accuracy 82.35
Epoch 19 Accuracy 82.462
Epoch 20 Accuracy 81.41000000000001
Epoch 21 Accuracy 78.676
Epoch 22 Accuracy 77.53999999999999
Epoch 23 Accuracy 82.446
Epoch 24 Accuracy 82.91
Epoch 25 Accuracy 83.978
Epoch 26 Accuracy 83.8
Epoch 27 Accuracy 82.048
Epoch 28 Accuracy 77.84400000000001
Epoch 29 Accuracy 82.734
Epoch 30 Accuracy 76.94
Epoch 31 Accuracy 84.3
Epoch 32 Accuracy 83.364
Epoch 33 Accuracy 85.112
Epoch 34 Accuracy 85.504
Epoch 35 Accuracy 77.45
Epoch 36 Accuracy 80.444
Epoch 37 Accuracy 79.164
Epoch 38 Accuracy 81.498
Epoch 39 Accuracy 83.492
Epoch 40 Accuracy 76.86
Epoch 41 Accuracy 82.812
Epoch 42 Accuracy 82.798
Epoch 43 Accuracy 85.524
Epoch 44 Accuracy 82.95
Epoch 45 Accuracy 83.366
Epoch 46 Accuracy 84.574
Epoch 47 Accuracy 84.922
Epoch 48 Accuracy 85.91
Epoch 49 Accuracy 85.658
Epoch 50 Accuracy 85.48
Epoch 51 Accuracy 84.084
Epoch 52 Accuracy 83.692
Epoch 53 Accuracy 83.096
Epoch 54 Accuracy 86.22800000000001
Epoch 55 Accuracy 84.77
Epoch 56 Accuracy 85.464
Epoch 57 Accuracy 85.58
Epoch 58 Accuracy 81.77600000000001
Epoch 59 Accuracy 86.65
```

```
Epoch 60 Accuracy 85.992
Epoch 61 Accuracy 86.824
Epoch 62 Accuracy 86.92
Epoch 63 Accuracy 87.238
Epoch 64 Accuracy 87.24
Epoch 65 Accuracy 87.148
Epoch 66 Accuracy 85.274
Epoch 67 Accuracy 87.16000000000001
Epoch 68 Accuracy 88.016
Epoch 69 Accuracy 86.982
Epoch 70 Accuracy 86.19
Epoch 71 Accuracy 87.786
Epoch 72 Accuracy 87.646
Epoch 73 Accuracy 88.05199999999999
Epoch 74 Accuracy 87.31400000000001
Epoch 75 Accuracy 87.96199999999999
Epoch 76 Accuracy 87.64999999999999
Epoch 77 Accuracy 87.28
Epoch 78 Accuracy 88.008
Epoch 79 Accuracy 87.972
Epoch 80 Accuracy 87.756
Epoch 81 Accuracy 88.064
Epoch 82 Accuracy 88.03999999999999
Epoch 83 Accuracy 88.05
Epoch 84 Accuracy 87.688
Epoch 85 Accuracy 88.168
Epoch 86 Accuracy 88.03
Epoch 87 Accuracy 87.66199999999999
Epoch 88 Accuracy 88.108
Epoch 89 Accuracy 88.212
Epoch 90 Accuracy 88.13
Epoch 91 Accuracy 88.32
Epoch 92 Accuracy 88.22
Epoch 93 Accuracy 88.276
Epoch 94 Accuracy 88.0
Epoch 95 Accuracy 88.39399999999999
Epoch 96 Accuracy 88.164
Epoch 97 Accuracy 87.83800000000001
Epoch 98 Accuracy 88.228
Epoch 99 Accuracy 88.03800000000001
Epoch 100 Accuracy 88.354
Epoch 101 Accuracy 88.348
Epoch 102 Accuracy 88.22
Epoch 103 Accuracy 88.262
Epoch 104 Accuracy 88.292
Epoch 105 Accuracy 88.30799999999999
Epoch 106 Accuracy 88.318
Epoch 107 Accuracy 88.304
Epoch 108 Accuracy 88.31599999999999
Epoch 109 Accuracy 88.27000000000001
Epoch 110 Accuracy 88.366
Epoch 111 Accuracy 88.372
Epoch 112 Accuracy 88.372
Epoch 113 Accuracy 88.31599999999999
Epoch 114 Accuracy 88.336
Epoch 115 Accuracy 88.402
Epoch 116 Accuracy 88.426
Epoch 117 Accuracy 88.412
Epoch 118 Accuracy 88.426
Epoch 119 Accuracy 88.20400000000001
```

```
Epoch 120 Accuracy 88.408
Epoch 121 Accuracy 88.354
Epoch 122 Accuracy 88.432
Epoch 123 Accuracy 88.388
Epoch 124 Accuracy 88.46000000000001
Epoch 125 Accuracy 88.332
Epoch 126 Accuracy 88.376
Epoch 127 Accuracy 88.434
Epoch 128 Accuracy 88.478
Epoch 129 Accuracy 88.446
Epoch 130 Accuracy 88.378
Epoch 131 Accuracy 88.426
Epoch 132 Accuracy 88.472
Epoch 133 Accuracy 88.414
Epoch 134 Accuracy 88.466
Epoch 135 Accuracy 88.44
Epoch 136 Accuracy 88.416
Epoch 137 Accuracy 88.498
Epoch 138 Accuracy 88.426
Epoch 139 Accuracy 88.442
Epoch 140 Accuracy 88.414
Epoch 141 Accuracy 88.446
Epoch 142 Accuracy 88.412
Epoch 143 Accuracy 88.426
Epoch 144 Accuracy 88.44999999999999
Epoch 145 Accuracy 88.442
Epoch 146 Accuracy 88.456
Epoch 147 Accuracy 88.474
Epoch 148 Accuracy 88.414
Epoch 149 Accuracy 88.416
Epoch 150 Accuracy 88.402
Epoch 151 Accuracy 88.414
Epoch 152 Accuracy 88.42
Epoch 153 Accuracy 88.426
Epoch 154 Accuracy 88.442
Epoch 155 Accuracy 88.46199999999999
Epoch 156 Accuracy 88.454
Epoch 157 Accuracy 88.428
Epoch 158 Accuracy 88.434
Epoch 159 Accuracy 88.472
Epoch 160 Accuracy 88.464
Epoch 161 Accuracy 88.44800000000001
Epoch 162 Accuracy 88.41
Epoch 163 Accuracy 88.414
Epoch 164 Accuracy 88.41799999999999
Epoch 165 Accuracy 88.444
Epoch 166 Accuracy 88.4
Epoch 167 Accuracy 88.41799999999999
Epoch 168 Accuracy 88.412
Epoch 169 Accuracy 88.416
Epoch 170 Accuracy 88.40400000000001
Epoch 171 Accuracy 88.442
Epoch 172 Accuracy 88.42
Epoch 173 Accuracy 88.428
Epoch 174 Accuracy 88.456
Epoch 175 Accuracy 88.426
Epoch 176 Accuracy 88.432
Epoch 177 Accuracy 88.44
Epoch 178 Accuracy 88.426
Epoch 179 Accuracy 88.436
```

```
Epoch 180 Accuracy 88.438
Epoch 181 Accuracy 88.438
Epoch 182 Accuracy 88.436
Epoch 183 Accuracy 88.432
Epoch 184 Accuracy 88.438
Epoch 185 Accuracy 88.44
Epoch 186 Accuracy 88.44
Epoch 187 Accuracy 88.436
Epoch 188 Accuracy 88.436
Epoch 189 Accuracy 88.42999999999999
Epoch 190 Accuracy 88.42999999999999
Epoch 191 Accuracy 88.432
Epoch 192 Accuracy 88.436
Epoch 193 Accuracy 88.428
Epoch 194 Accuracy 88.432
Epoch 195 Accuracy 88.436
Epoch 196 Accuracy 88.432
Epoch 197 Accuracy 88.436
Epoch 198 Accuracy 88.436
Epoch 199 Accuracy 88.438
```

In [25]:

```
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: 88.436000
```

Validate Softmax on Fashion-MNIST

In [26]:

```
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: 84.540000
```

Testing Softmax on Fashion-MNIST

In [27]:

```
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: 83.530000
```

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:

In [28]:

```
output_submission_csv('kaggle/softmax_submission_fashion.csv', softmax_fashion.predict
```

Train Softmax on Rice

In [29]:

```
lr = 0.5
n_epochs = 200
reg_const = 1

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

```
Epoch 0 Accuracy 47.28255888552837
Epoch 1 Accuracy 45.220419759875355
Epoch 2 Accuracy 54.779580240124645
Epoch 3 Accuracy 54.779580240124645
Epoch 4 Accuracy 45.220419759875355
Epoch 5 Accuracy 45.220419759875355
Epoch 6 Accuracy 45.220419759875355
Epoch 7 Accuracy 60.141141966822474
Epoch 8 Accuracy 45.220419759875355
Epoch 9 Accuracy 54.779580240124645
Epoch 10 Accuracy 64.89780954999543
Epoch 11 Accuracy 45.220419759875355
Epoch 12 Accuracy 76.711575474292
Epoch 13 Accuracy 54.779580240124645
Epoch 14 Accuracy 45.220419759875355
Epoch 15 Accuracy 75.55677756392632
Epoch 16 Accuracy 76.253322335258
Epoch 17 Accuracy 45.220419759875355
Epoch 18 Accuracy 56.29181559893686
Epoch 19 Accuracy 73.71459994500962
Epoch 20 Accuracy 56.695078361286775
Epoch 21 Accuracy 45.220419759875355
Epoch 22 Accuracy 78.40711208871781
Epoch 23 Accuracy 78.33379158647237
Epoch 24 Accuracy 60.05865640179635
Epoch 25 Accuracy 54.779580240124645
Epoch 26 Accuracy 51.76427458528091
Epoch 27 Accuracy 45.220419759875355
Epoch 28 Accuracy 45.220419759875355
Epoch 29 Accuracy 45.220419759875355
Epoch 30 Accuracy 45.220419759875355
Epoch 31 Accuracy 52.01173128035927
Epoch 32 Accuracy 72.61479241132803
Epoch 33 Accuracy 45.220419759875355
Epoch 34 Accuracy 57.61158463935479
Epoch 35 Accuracy 57.061680872513975
Epoch 36 Accuracy 45.220419759875355
Epoch 37 Accuracy 67.61066813307671
Epoch 38 Accuracy 54.779580240124645
Epoch 39 Accuracy 77.82054807075428
Epoch 40 Accuracy 59.618733388323705
Epoch 41 Accuracy 79.15864723673357
Epoch 42 Accuracy 45.220419759875355
Epoch 43 Accuracy 55.21950325359729
Epoch 44 Accuracy 55.622766015947214
Epoch 45 Accuracy 76.54660434423975
Epoch 46 Accuracy 71.78993676106681
Epoch 47 Accuracy 50.09623315919715
Epoch 48 Accuracy 63.46805975620933
Epoch 49 Accuracy 63.633030886261565
Epoch 50 Accuracy 55.476125011456325
Epoch 51 Accuracy 66.75831729447347
Epoch 52 Accuracy 56.695078361286775
Epoch 53 Accuracy 55.41196957199157
Epoch 54 Accuracy 68.06892127211071
Epoch 55 Accuracy 66.92328842452571
Epoch 56 Accuracy 50.2520392264687
Epoch 57 Accuracy 69.41618550087068
Epoch 58 Accuracy 60.59939510585648
Epoch 59 Accuracy 72.09238383282926
```

```
Epoch 60 Accuracy 50.78361286774814
Epoch 61 Accuracy 74.3286591513152
Epoch 62 Accuracy 57.12583631197874
Epoch 63 Accuracy 59.04133443314087
Epoch 64 Accuracy 72.45898634405646
Epoch 65 Accuracy 49.39968838786546
Epoch 66 Accuracy 70.6443039134818
Epoch 67 Accuracy 63.183942810008254
Epoch 68 Accuracy 63.733846576849054
Epoch 69 Accuracy 74.01704701677207
Epoch 70 Accuracy 79.25946292732105
Epoch 71 Accuracy 77.72889744294748
Epoch 72 Accuracy 58.12482815507286
Epoch 73 Accuracy 62.20328109247548
Epoch 74 Accuracy 56.319310787278894
Epoch 75 Accuracy 72.20236458619742
Epoch 76 Accuracy 76.03336082852168
Epoch 77 Accuracy 70.66263403904317
Epoch 78 Accuracy 74.69526166254239
Epoch 79 Accuracy 75.87755476125011
Epoch 80 Accuracy 74.02621207955275
Epoch 81 Accuracy 50.18788378700394
Epoch 82 Accuracy 72.26652002566217
Epoch 83 Accuracy 75.45596187333882
Epoch 84 Accuracy 77.01402254605443
Epoch 85 Accuracy 76.16167170745119
Epoch 86 Accuracy 79.24113280175969
Epoch 87 Accuracy 78.94785079277793
Epoch 88 Accuracy 74.72275685088444
Epoch 89 Accuracy 70.58014847401705
Epoch 90 Accuracy 79.04866648336541
Epoch 91 Accuracy 74.87856291815599
Epoch 92 Accuracy 48.60232792594629
Epoch 93 Accuracy 71.7991018238475
Epoch 94 Accuracy 74.28283383741179
Epoch 95 Accuracy 59.28879112821923
Epoch 96 Accuracy 73.18302630373019
Epoch 97 Accuracy 73.9528915773073
Epoch 98 Accuracy 56.56676748235725
Epoch 99 Accuracy 63.596370635138854
Epoch 100 Accuracy 73.70543488222894
Epoch 101 Accuracy 78.09549995417468
Epoch 102 Accuracy 79.25029786454037
Epoch 103 Accuracy 67.26239574741088
Epoch 104 Accuracy 79.67189075245166
Epoch 105 Accuracy 77.11483823664193
Epoch 106 Accuracy 65.94262670699294
Epoch 107 Accuracy 75.29099074328659
Epoch 108 Accuracy 60.26028778297131
Epoch 109 Accuracy 61.79085326734488
Epoch 110 Accuracy 74.98854367152416
Epoch 111 Accuracy 61.77252314178352
Epoch 112 Accuracy 69.97525433049216
Epoch 113 Accuracy 78.28796627256897
Epoch 114 Accuracy 79.58024012464485
Epoch 115 Accuracy 75.43763174777747
Epoch 116 Accuracy 80.13930895426634
Epoch 117 Accuracy 78.76454953716433
Epoch 118 Accuracy 58.92218861699202
Epoch 119 Accuracy 71.45999450096234
```

```
Epoch 120 Accuracy 79.24113280175969
Epoch 121 Accuracy 58.75721748693978
Epoch 122 Accuracy 78.88369535331317
Epoch 123 Accuracy 79.48858949683806
Epoch 124 Accuracy 63.550545321235454
Epoch 125 Accuracy 78.58124828155073
Epoch 126 Accuracy 61.28677481440747
Epoch 127 Accuracy 68.1697369626982
Epoch 128 Accuracy 79.35111355512785
Epoch 129 Accuracy 80.8450187883787
Epoch 130 Accuracy 70.31436165337732
Epoch 131 Accuracy 67.41820181468243
Epoch 132 Accuracy 79.2869581156631
Epoch 133 Accuracy 76.4366235908716
Epoch 134 Accuracy 72.0190633305838
Epoch 135 Accuracy 61.66254238841537
Epoch 136 Accuracy 72.81642379250299
Epoch 137 Accuracy 67.83062963981304
Epoch 138 Accuracy 54.128860782696364
Epoch 139 Accuracy 80.95499954174686
Epoch 140 Accuracy 74.89689304371736
Epoch 141 Accuracy 71.73494638438274
Epoch 142 Accuracy 73.36632755934377
Epoch 143 Accuracy 76.64742003482723
Epoch 144 Accuracy 72.29401521400422
Epoch 145 Accuracy 72.58729722298598
Epoch 146 Accuracy 63.00064155439464
Epoch 147 Accuracy 69.84694345156265
Epoch 148 Accuracy 71.74411144716342
Epoch 149 Accuracy 72.3948309045917
Epoch 150 Accuracy 60.86518192649619
Epoch 151 Accuracy 77.27064430391349
Epoch 152 Accuracy 78.0405095774906
Epoch 153 Accuracy 76.88571166712492
Epoch 154 Accuracy 85.23508386032445
Epoch 155 Accuracy 84.91430666300064
Epoch 156 Accuracy 67.5190175052699
Epoch 157 Accuracy 56.56676748235725
Epoch 158 Accuracy 76.95903216937036
Epoch 159 Accuracy 88.2870497662909
Epoch 160 Accuracy 65.14526624507377
Epoch 161 Accuracy 85.14343323251764
Epoch 162 Accuracy 88.67198240307947
Epoch 163 Accuracy 77.59142150123728
Epoch 164 Accuracy 93.53863073962057
Epoch 165 Accuracy 83.6495279992668
Epoch 166 Accuracy 99.18430941251948
Epoch 167 Accuracy 99.84419393272844
Epoch 168 Accuracy 99.92667949775455
Epoch 169 Accuracy 99.78003849326367
Epoch 170 Accuracy 99.89918430941252
Epoch 171 Accuracy 99.53258179818532
Epoch 172 Accuracy 99.44093117037852
Epoch 173 Accuracy 99.79836861882504
Epoch 174 Accuracy 99.71588305379892
Epoch 175 Accuracy 99.85335899550913
Epoch 176 Accuracy 99.89001924663185
Epoch 177 Accuracy 99.83502886994776
Epoch 178 Accuracy 99.89001924663185
Epoch 179 Accuracy 99.84419393272844
```

```
Epoch 180 Accuracy 99.88085418385117
Epoch 181 Accuracy 99.91751443497388
Epoch 182 Accuracy 99.89918430941252
Epoch 183 Accuracy 99.89918430941252
Epoch 184 Accuracy 99.9083493721932
Epoch 185 Accuracy 99.91751443497388
Epoch 186 Accuracy 99.87168912107049
Epoch 187 Accuracy 99.9083493721932
Epoch 188 Accuracy 99.86252405828981
Epoch 189 Accuracy 99.89918430941252
Epoch 190 Accuracy 99.84419393272844
Epoch 191 Accuracy 99.89918430941252
Epoch 192 Accuracy 99.89001924663185
Epoch 193 Accuracy 99.84419393272844
Epoch 194 Accuracy 99.74337824214096
Epoch 195 Accuracy 99.85335899550913
Epoch 196 Accuracy 99.9083493721932
Epoch 197 Accuracy 99.89918430941252
Epoch 198 Accuracy 99.9083493721932
Epoch 199 Accuracy 99.9083493721932
```

In [30]:

```
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: 99.908349
```

Validate Softmax on Rice

In [31]:

```
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: 99.835029
```

Testing Softmax on Rice

In [32]:

```
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: 99.862524
```

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

Training Logistic Classifer

In [33]:
```
learning_rate = 0.2
n_epochs = 10
threshold = 0

y_train_RICE = np.where(y_train_RICE == 0, -1, y_train_RICE)

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

```
Epoch 0 Accuracy 54.779580240124645
Epoch 1 Accuracy 69.36119512418661
Epoch 2 Accuracy 71.91824763999634
Epoch 3 Accuracy 68.49967922280268
Epoch 4 Accuracy 91.74227843460727
Epoch 5 Accuracy 78.38878196315645
Epoch 6 Accuracy 94.84923471725781
Epoch 7 Accuracy 97.35129685638346
Epoch 8 Accuracy 95.8665566859133
Epoch 9 Accuracy 98.47859957840711
```

In [34]:
```
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: 99.624232
```

Validate Logistic Classifer

In [35]:
```
y_val_RICE = np.where(y_val_RICE == 0, -1, y_val_RICE)
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: 99.615067
```

Test Logistic Classifier

In [36]:
```
y_test_RICE = np.where(y_test_RICE == 0, -1, y_test_RICE)
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: 99.642563
```