Jagannathan Govindan

IST 718

Lab #3

**Introduction**

The MNIST dataset is a collection of images created by the National Institute of Standards and Technology (NIST). Each handwritten image in the dataset consists of 28 x 28 pixels. With the provided 60,000 images for training, and 10,000 for testing. As part of this assignment, different models are going to be tried to find the accuracy of the predicting them.

Like the original MNIST dataset, the fashion variant consists of the same number of train and test images, each being the same dimension. But each image in the collection is a series of ten possible clothing types.

We will try Neural networks and SVM algorithm on this dataset and the results will be shared.

**Obtain and Scrub:**

The dataset used for this study was obtained directly from the fashion-mnist repository. Then each corresponding file were committed into a dedicated code crepository:

* t10k-images-idx3-ubyte.gz
* t10k-images-idx1-ubyte.gz
* train-images-idx3-ubyte.gz
* train-labels-idx3-ubyte.gz

Since the datasets were downloaded locally, two different functions were used to load the datasets into python. First, the input\_data.read\_data\_sets function was used, and needed by the corresponding neural network. Specifically, the tensorflow implementation requires the input object to be the base.Dataset type. Next the load\_mnist was used for the svm modelling. This function simply returns both images and labels. Since the fashion-mnist repository was not cloned, the latter function was copied locally into the svm.py.

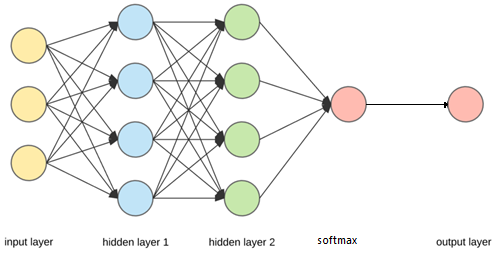
**Models:**

Once the required dataset was loaded into python, a brief exploratory verified that the dataset is the expected fashion MNIST:

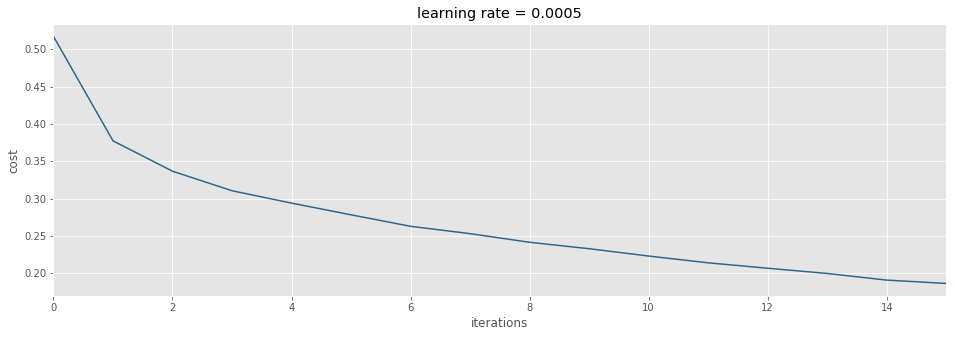


A neural network was trained using two hidden layers each implementing a linear function, followed by a ReLU function. Finally, a softmax function creates an output for each fashion target class.

An initialize\_parameters function was defined for the input and output of each neural network layer. The aggregation of each of these layer attributes was used to define the forward propagation behavior of the overall neural network:

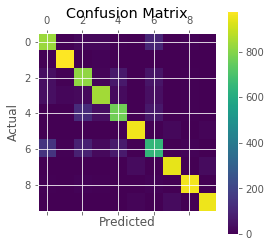


In addition to the forward propagation, the AdamOptimizer[[1]](#footnote-1) was used for each epoch iteration to specify the type of backward propagation. Default values, with a learning rate of 0.0005 portrayed that the number of iterations stabilize at about 14 iterations.



Finally, the overall train accuracy for the generated neural network was 0.935, while the test accuracy was 0.889.

**Note:** the implemented neural network was an adaptation from Vivian Rajkuma[[2]](#footnote-2).



SVM Trained Classifier Accuracy: 0.8723

Predicted Values: [9 2 1 ..., 8 1 5]

Classifier on Validation Images: 0.8723

Confusion Matrix:

[[834 5 21 20 4 2 104 0 9 1]

[ 4 981 1 8 3 0 3 0 0 0]

[ 39 6 819 12 66 0 54 0 4 0]

[ 41 16 25 846 29 0 39 0 4 0]

[ 2 1 121 39 769 0 63 0 5 0]

[ 0 0 0 1 0 960 0 22 3 14]

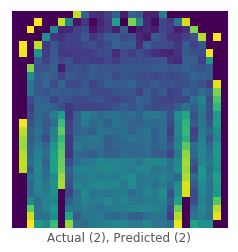
[147 4 91 26 72 0 650 0 10 0]

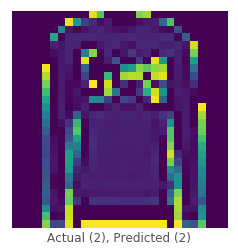
[ 0 0 0 0 0 27 0 944 0 29]

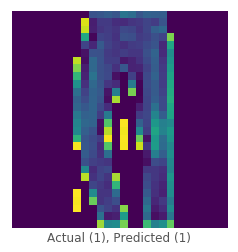
[ 6 0 10 4 2 3 11 2 962 0]

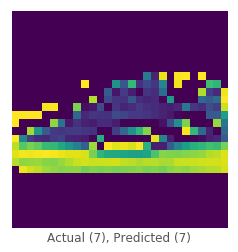
[ 0 1 0 0 0 12 1 28 0 958]]

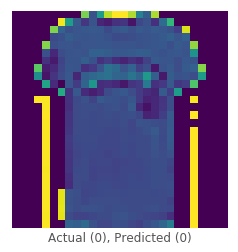
The results of the svm prediction matches earlier test accuracy:

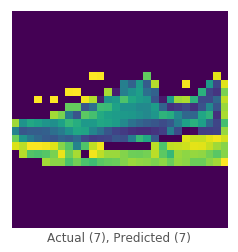


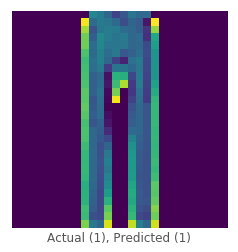


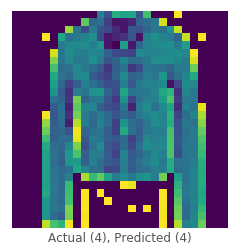


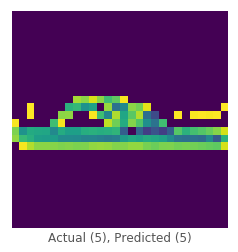


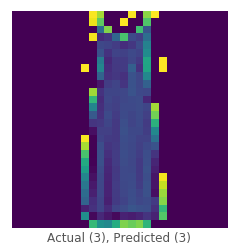


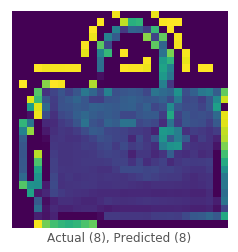


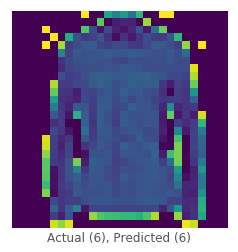


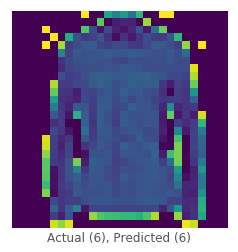


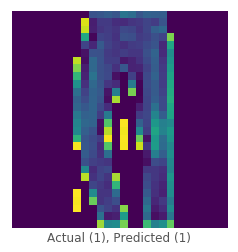


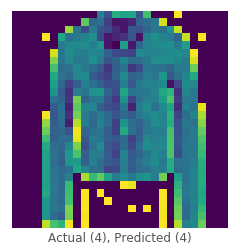












It is generally known that svm is much less performant than the neural network. For this reason, the corresponding benchmark has been omitted as an exercise. However, this study clearly illustrates that the neural network implementation is better performing. However, the degree of performance is not significantly large. Therefore, it would be interesting to see if adjustments in the neural network weights, number of layers and different backward propagation techniques could increase the difference between the two implementations. Similarly, changing the parameters for the svm, including the kernel and margin of the separating plane would be an interesting future study.

Compared to the traditional MNIST number classification, the fashion variant is not as accurate. Specifically, the MNIST techniques have been found in the upper 98-99%. However, as stated earlier, more clever techniques could be imposed to draw higher results. This could involve a series of ensemble learners. Though a simple additional approach could involve appending to the provided dataset. The additional data could be as simple as a set of images not belonging to any of the target classes. For example, additional images of non-fashion images could be used during the train. This would be analogous to extending the one versus rest, taken advantage of within the sklearn framework.

**Conclusions:**

From the above results we can see that a multilayer neural network can outperform the traditional support vector machine (svm). However, as stated in the results, it would be interesting to explore the adjustment of parameters for both techniques. Specifically, adjusting the penalty and changing the kernel to radial in the svm. However, this may not improve the performance gap between the two approaches, since the neural network implements linear regression.

**Python** **code**:



1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)