Classify Fashion Products from Fashion-MNIST Thumbnail Images

Leo Hu and **Natnael Mulat**
{lehu,namulat}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

We aim to train machine learning models that can determine the category of a fashion product based on thumbnail images. Specifically, we trained logistic regression models and *k*-nearest neighbors classifiers. To reduce the run time, we compressed the images using Principle Component Analysis (PCA). For logistic regression, we tuned our models by varying the optimizing methods and strength of the penalty term. For *k*-NN implementation, we ran both *l*1 and *l*2 distance metric measurements each with variations of *k*. We determined our best model is the *k*-NN classifier with *l*2 distance with *k* value of 6. Our best *K*-NN classifier produced an accuracy score of 0.851 on the test set.

1 Introduction

We aim to train machine learning models that can determine the category of a fashion product based on thumbnail images. Our data set contains thumbnail images of 70,000 unique products. Each fashion product can be grouped under one of these categories: T-shirt/top, trouser, pullover, dress, coat, sandals, shirt, sneaker, bag or ankle boots. The fashion products come from different gender and age groups: men, women, kids and neutral. The images are stored in matrices of pixels. Our model aims to classify a fashion product based on its matrix of pixels.

The same problem has been studied by Xiao, Rasul and Vollgraf. In their work, they compared the predictive power of decision tree classifiers, *k*-nearest neighbors classifiers, logistic regressions, random forest classifiers, support vector machines, etc. They determined that support vector machines (SVM) were the best classifiers with an average accuracy score of 0.897, which outperformed logistic regressions and *k*-nearest neighbors classifiers. Logistic regression and k-nearest neighbors classifiers yield similar performances, having average accuracy scores of 0.842 and 0.854 respectively (Xiao, Rasul, and Vollgraf 2017). Based on what we learned in class, we aim to produce logistic regression models and k-nearest neighbors classifiers with balanced complexity that will not overfit or underfit and perform similarly to those produced by Xiao, Rasul and Vollgraf.

The paper is organized as follows: Section 2 introduces the dataset used to develop the models and the methods employed to prepare the dataset before fitting the models. Section 3 discusses our experimental setup and related techniques. Section 4 presents our findings and the analysis of our results. Section 5 points out the broader impact of our research, and Section 6 summarizes the research questions, our results, and ways to improve or build upon this research. Finally, section 7 summarizes the contributions of each researcher, and section 8 contains our acknowledgements for other researchers who have helped us develop our models.

2 Data Preparation

Dataset

We used the publicly available dataset titled "Fashion-MNIST", a new dataset comprised of 28×28 grayscale images of 70,000 fashion products thumbnails from 10 categories, with 7,000 images per category (Xiao, Rasul, and Vollgraf 2017). The original dataset consists of a training dataset having 60,000 images and a test dataset having 10,000 images. Every image has a corresponding label that indicates its category. Each image is stored in a 1 x 784 array of pixels to represent a 28×28 matrix. Each element in the array corresponds to a pixel value, ranging from 0 to 255. As a result, our training dataset is a 60,000 x 784 matrix, and our test dataset is a 10,000 x 784 matrix. The labels for our training dataset are stored in a 60,000 x 1 matrix, and the ones for our test dataset are stored in a 10,000 x 1 matrix. Possible labels are T-shirt/top, trouser, pullover, dress, coat, sandals, shirt, sneaker, bag or ankle boots. Integers ranging from 0 to 9 represent each of these labels respectively.

For our machine learning models, the input is a 1 x 784 pixel array of a fashion product image, while the output is an integer representing the category of the fashion product. Therefore, we have 784 features, with each feature representing a pixel value.

Reprocessing

Before the experiments, we shuffled the given training dataset and its corresponding label set. Out of the given training dataset, we used a train set of 80 % and a validation set of 20 % to train our model. Additionally, to help our gradient descent algorithm converge faster, we did min-max scaling on our train set and validation set. We divide all pixel values by 255, such that each value in our matrix ranges from 0 to 1.

Feature Selection

Since using all 784 features to develop our models took a longer time than expected, we compressed our images using Principle Component Analysis (PCA). The PCA process uses Singular Value Decomposition (SVD) to project the dataset to a lower dimensional space, such that the base vectors in the lower dimensional space explain most of the variances in the original dataset. Let vector $x = [x_1, x_2, x_3 ... x_{784}]$ be the collection of 784 pixels. Specifically, PCA on image compression works by finding a new list of basis B on each pixel such that it preserves a portion of the original variances. Then, we took the mean $\bar{X}$, and added back a few pixel values to reconstruct the bulk of the image using the B (VanderPlas 2017). To find PCA of image J:

$$PCA(J) = \bar{X} + x_1 \cdot (B_1) + x_2 \cdot (B_2) + x_3 \cdot (B_3) \dots$$

Initially, we compressed our images to preserve 95% of the variances. We got 187 features out of the original 784, with a significant loss of image quality. Then, we compressed our images to preserve 99% of the variances, which gave us 458 features to work with, without a significant loss of image quality. *Figure 1* presents comparison of the original and compressed images using PCA by inverse transforming the data to view how the data changes because of the PCA. As can be seen, the two pictures very similar, but the quality of the 458 features is slightly lower than the image that has 784 features. However, for our analysis, retaining 99% of the variances makes training our model easier without any comparable loss in performance.

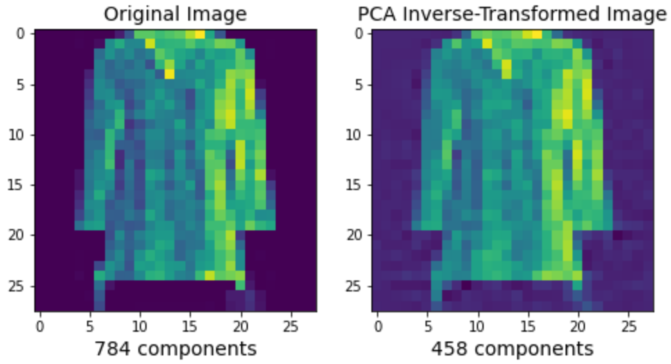

Figure 1 : Images of Fashion Products before and after Compression Using PCA

3 Experiments

To develop an optimal classification model that can predict the categories for Fashion MNIST dataset, we trained 3 models with logistic regression with variation of 3 gradient descent solvers, and k-nearest neighbors algorithm with 50 variation of k to choose the best model. Each developed model used a training set where each prediction is computed with an estimator fitted on the corresponding validation set. The rest of this section has the experiments for logistic regression and k-nearest neighbors with specifications set-up and the details of algorithms used.

Logistic Regression

For log regressions, the cost function, $J(\theta)$ we optimized is equal to

$$J(\theta) = \frac{1}{m}(\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^i) + (1 - y^{(i)}) \log(1 - h_\theta(x^i)))$$

where m is the number of features, i is the number of examples, and $h(\theta)$ is our hypothesis function defined as

$$h(\theta) = \frac{1}{1 + e^{-\sum_{i=1}^{m} \theta_i x_i}}$$

In order to overcome the problem of over-fitting and under-fitting, we used $l2$ or Ridge regularization, such that if the coefficients take large values, the optimization function is penalized. To account for $l2$, a regularization term is added to the cost function such that

$$\frac{1}{m}(\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^i) + (1 - y^{(i)}) \log(1 - h_\theta(x^i))) + \frac{1}{m}(\lambda(\sum_{j=1}^{n} \theta_j^2))$$

where the regularization parameter λ is a control on the fitting parameters. As the magnitudes of the fitting parameters increase, there will be an increasing penalty on the cost function. This penalty is dependent on the squares of the parameters as well as the magnitude of λ.

To optimise $J(\theta)$ we used Stochastic Average Gradient(SAG) and quasi-Newton method known as limited-memory Broyden–Fletcher–Goldfarb–Shanno(L-BFGS-B). In SAG optimization, for each feature, we reset $\theta_j \leftarrow \theta_j - \alpha \nabla J(\theta)$ where α is the learning rate that we chose for the descent step(Schmidt and Bach 2017). This process is run until it converges to a tolerance t. However, in SAG the step-size is set to

$$\frac{1}{(1/n) + L + i}$$

where n is the number of samples/examples in the data set, while L is the max sum of squares for over all examples, and i is a constant either 1 or 0 depending on whether bias or intercept is added to the decision function(Schmidt and Bach 2017).

Stochastic average gradient descent is a first-order optimization method, as it uses first-order information (ie. the gradient) to find the minimum. On the other hand, L-BFGS-B optimization method takes into account the objective function's second-order behavior in addition to its first-order behavior in order to determine the step size. With this optimization method, there is no longer a need to set a learning rate parameter since our step size is determined exactly by the distance to the minimum of the fitted parabola at that point(Rafati and Marica 2020). Thus, we reset θ_j as

$$\theta_j \leftarrow \theta_j - \frac{\frac{\partial}{\partial j(\theta)} J(\theta)}{\frac{\partial^2}{\partial \theta^2} J(\theta)}$$

The first derivative is replaced by the gradient $\nabla J(\theta)$ and the second derivative is replaced by the *Hessian H*, which is rewritten as

$$\theta_j \leftarrow \theta_j - \frac{\nabla J(\theta)}{H(J(\theta))}$$

Although, SAG and L-BFGS-B optimize the cost function for $l2$ regularization, we used SAGA, variant of SAG,that supports the non-smooth $l1$ regularization option to optimize the cost function(Defazio, Bach, and Lacoste-Julien 2014). The models developed for $l1$ regularization optimize the cost function $J(\theta)$:

$$\frac{1}{m}(\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^i) + (1-y^{(i)}) \log(1-h_\theta(x^i))) + \frac{1}{m}(\lambda|\sum_{j=1}^{n} \theta_j|)$$

k-Nearest Neighbors (k-NN)

k-NN implementations developed in this paper use both Euclidean distance($l2$-distance) and Manhattan Distance($l1$-distance). Both metrics measure the distance between observation x_a and x_b for all j features. For $l2$-distance, we calculated the distance by finding the sum of differences between the two observations squared. In other words, the Euclidean distance reflects the linear distance between two points in Euclidean space. On the other hand, $l1$-distance metric is the absolute difference between two observations. For our models we ran both variations to pick the best model that had the highest accuracy score. Further, another specification of our k-NN model is that if two points are equidistant from the unclassified observation, then the algorithm takes the mode of the variable. In other words, if in the dataset, number of datapoints for class 1 is more, then 1 would be used on the tie. If on the other hand, the mode of the two classes are the same, then the algorithm randomly chooses a label between the tied neighbors to assign for the variable.

Hyper-Parameter Tuning

We tested our models by varying the optimizing methods and λ strength of the penalty term for logistic regression. For k-NN implementation, we ran $l1$ and $l2$ distance metric measurements each with variation of k, which is the number of nearest neighbors to include in the majority of the voting process.

For each optimization method, we ran variations of λ for $l2$ regularization such that $\lambda = \{0.05, 0.1, 0.5, 1\}$. The k-NN implementation varied in the number of nearest neighbours in the space of $k = \{1, 2, 3, ..., 50\}$ for Euclidean and Manhattan distance measurement to find the best k value and distance measurement method. All models used the validation set to provide an unbiased evaluation of a model fit on the training data set while tuning model hyper-parameters. We used accuracy score to compare the models developed since we were using a balanced data set. After comparing the models using the accuracy score, we selected the best model to run our test data set.

4 Results

To assess the predictive ability of our models we set up a benchmark model that predicted the correct figure with an accuracy score of 0.1. Our benchmark model always predicted the same category, such as" trouser", for any arbitrary figure input. Since the 10 categories are perfectly balanced, we would get the correct category label with a probability of 0.1.

We developed logistic regression models using different optimization and regularization methods with different penalty strengths. The accuracy score results for the logistic regressions are summarized on *Table* 1.

Model	λ=0.05	λ=0.1	λ =0.5	λ=1
$SAG, p = L2$	0.8525	0.8523	0.8510	0.8509
L-BFGS-B, $p = L2$	0.8524	0.8525	0.8517	0.8515
$SAGA, p = L1$	0.8450	0.8519	0.8527	0.8521

Table 1 : Logistic Regression Models' Accuracy Score Results

The first row of *Table* 1 has the accuracy scores for SAG optimization method with different levels of penalty terms with $l2$ regularization for over-fitting. We found out that λ of 0.05 has a slightly better accuracy score than the other models developed using SAG. The second row of *Table*1 shows the accuracy scores for L-BFGS-B optimization method, and interestingly, we got the same accuracy score of 0.8525 as the best model with SAG optimization method. However, our best model from L-BFGS-B optimization method uses λ of 0.1 rather than 0.05. The top two models we developed using the SAG and L-BFGS-B optimization method use $l2$ regularization, to check if $l1$ regularization better fit the data, we ran models with SAGA optimization that also gave us the option of developing $l1$ regularized models. The results we got from $l1$ regularization have a slightly lower accuracy score than the $l2$ regularization. The best model we got using SAGA optimization with $l1$ regularization has an accuracy score of 0.8519 which is lower than the best model for SAG and L-BFGS-B optimization methods with $l2$ regularization.

For $l2$ regularization, a general trend of decreasing the strength of penalty λ gave us a better accuracy score. However, for $l1$ regularization, as the strength of penalty increased, the models performed better in terms of accuracy score.

We also varied the classification methods to develop models. Besides logistic regression, we ran k-NN classification with different distance measurement methods and plotted the accuracy score for the number of neighbors k, which varies from 1 to 50. The result for Manhattan distance measurement is summarized in *Figure* 2. The maximum accuracy score we got for Manhattan distance method is 0.8494 with k value of 4. As we increased the number of neighbors beyond 4, the accuracy score declined. As k increases, the k-NN fits a smoother curve to the data. This is because a higher value of k reduces the edginess by taking more data into account, thus reducing the overall complexity and flexibility of the model.

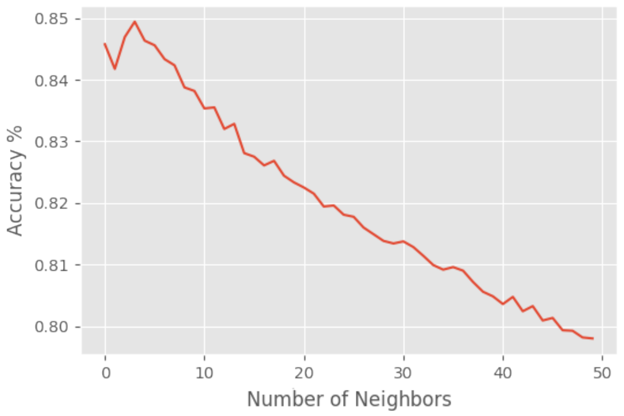

Figure 2 : KNN using Manhattan Distance - Accuracy Score

We also ran similar experiments of *k*-NN classifiers with the Euclidean distance method. The results are summarized on Figure 3. From those models, we learned that the maximum accuracy score is 0.8578 when the number of neighbors *k* is set to 6, which is a better prediction score than the *k*-NN classifier with Manhattan distance measurement. Overall, the *k*-NN classifier with 6 number of neighbours and Euclidean distance measurement performed better than the other models developed. When we compared our best model with the benchmark of 0.1, it performed much better with an accuracy score of 0.8578.

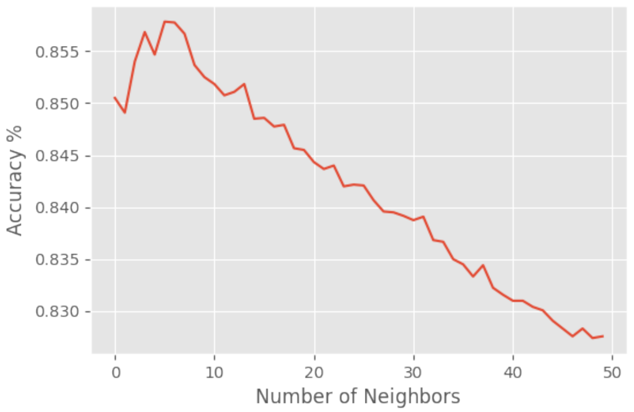

Figure 3 : KNN using Euclidean Distance - Accuracy Score

With the models we developed, we expected the test set to perform as similar to our best model developed with *k*-NN classifiers using Euclidean distance measurement with 6 neighbours. We expected the accuracy score to be similar with range of 0.84 to 0.86. The results of our test set are summarized in *Figure* 3.

The accuracy score of the test set is 0.851, which is similar to most of the logistic regression models we developed. For instance, SAG and L-BFGS-B optimization methods with a penalty term of 0.5 had the same results with an accuracy score of 0.851. This model can predict correctly with accuracy score that is around 75 percentage points more than our benchmark model. From the confusion matrix on *Figure* 4, we can easily separate some classes that our model can predict correctly. Classes such as bags, boots, sneakers, trousers, and sandals are the easiest classes to predict for our model. On the contrary, it seems hard to separate shirts from t-shirts, coats and pullovers.

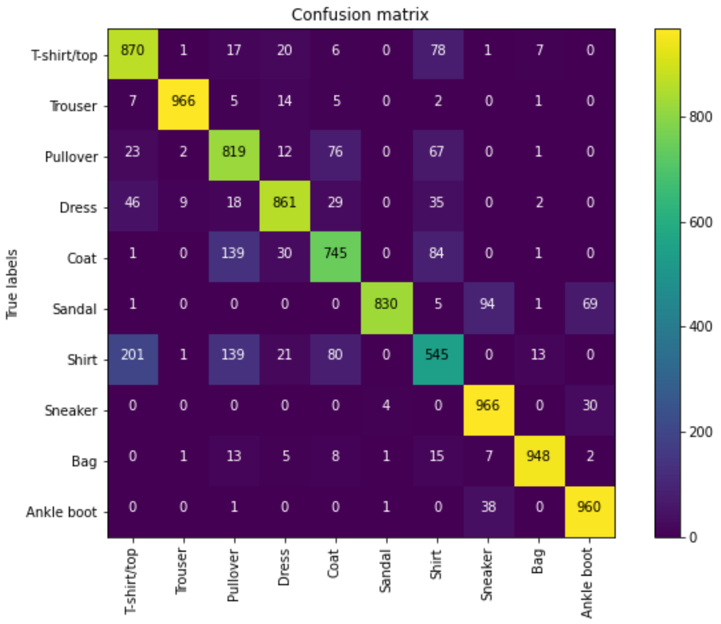

Figure 4 : Confusion Matrix for Test Set

Overall, the results we got from the test set seem to match the work of Xiao, Rasul, and Vollgraf. Our logistic regression results slightly outperform the results they got since their best logistic regression model had an accuracy score of 0.842 with $l1$ penalty and λ of 1. Our best logistic model had an accuracy score of 0.8525, which we got by using SAG and L-BFGS-B optimization methods with $l2$ penalty and λ of 0.1 and 0.05. Their best *k*-NN classifier model had an accuracy score of 0.854 with $l1$-distance and k value of 9 and 5 had similar accuracy score. However, the accuracy score we got was 0.851 using $l2$ distance with k value of 6.

5 Broader Impacts

Similar to the model we trained, computer vision systems that can recognize objects is a topic of frequent debate, especially when deployed in high stakes environments. A *NewYorkTimes* article reported how Tesla cars crashed using machine learning as they are not trained for each instance (Lohr 2016). As a result, when the car deals with an object that has not been trained on, it might ignore it. Although that is a good approach since most of the objects that a car passes by are not necessarily supposed to be recognized since they are not in front of the car. But there are unusual instances where those unrecognized objects may appear on the road causing an accident since the autopilot mode of the car is not trained to recognize that object.

Furthermore, image recognition models can not contextually understand images as humans do. The same $NewYorkTimes$ article also pointed out that Facebook algorithm once took down an image of a naked, 9-year-old girl fleeing napalm bombs posted by a Norwegian author (Lohr 2016). The algorithm saw child pornography instead of an iconic photo of the Vietnam War and human suffering. Understanding the intricacies of everyday life are inherent in human visual intelligence, but beyond the reach of current machine learning technology.

As a result, we should seriously reconsider the deployment of image recognition technologies in environments where understanding your surrounding is crucial and can have real life impacts.

6 Conclusions

In this paper, we aimed to find a model that predicts the class of images given a training set of images with labels. In order to reduce the training time we normalized and ran a PCA algorithm on the dataset. We trained logistic regression with $l1$ and $l2$ regularization with SAG, SAGA, and L-BFGS-B optimization method and $k-$NN classifiers to develop a model that can predict the class of an image. After comparing the accuracy score of our models, k-NN classifier with $l2$ distance and k value of 6 was chosen as it fits the data set better than the other models specified. Using that model, we used our test set to get an accuracy score of 0.851. Our results show that, the model is able to *exactly* predict 85% of the true image classes, which is much better than our benchmark model with an accuracy score of 0.1.

In addition, we also found out that the Euclidean distance measurement better suited the problem at hand. In addition, our analysis of the results indicates that classes, such as shirts are harder to predict for our model. This may be because there are similarities between shirt, t-shirt, pullover, and coat. While classes such as bags, boots, sneakers,trousers, and sandals are the easiest classes to predict.

7 Contributions

Leo Hu wrote the Abstract, Introduction, Data Preparation, and Broader Impacts section, while Natnael Mulat prepared the data for the models by normalizing and running PCA on the data to produce $Figure$ 1. In addition, Natnael Mulat worked on developing the models using Scikit-learn's buitin regression solver and k-nn classifier, and produced the figures and tables associated with the results of the models. Natnael Mulat wrote the Experimentation and Results. Leo Hu signed up appointments with Dr. Raghuram Ramanujan and the Writing Center tutor Ben Caldwell $21'$. Both partners wrote the Conclusion section, and proofread every sections together with writing center tutor Ben Caldwell ‘21.

8 Acknowledgements

We would like to thank Dr. Raghuram Ramanujan for his generous advice about our drafts and coding. We also would like to extend our acknowledgements to Ben Caldwell ‘21 for reviewing and proofreading our paper.

References

Defazio, A.; Bach, F. R.; and Lacoste-Julien, S. 2014. *SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives*.

Lohr, S. 2016. A lesson of tesla crashes? computer vision can't do it all yet.

Rafati, J., and Marica, R. 2020. *Quasi-Newton Optimization Methods for Deep Learning Applications*. 9–38.

Schmidt, M., and Bach, F. 2017. *Minimizing Finite Sums with the Stochastic Average Gradient*.

VanderPlas, J. T. 2017. *Python data science handbook: essential tools for working with data*. O'Reilly.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.