COMP5318

MACHINE LEARNING AND DATA MINING

ASSIGNMENT 1 REPORT

CLASSIFICATION OF FASHION MNIST DATASET based on Machine Learning Algorithms

GROUP 33:

Haichen Zhu - hzhu8034

Yeseul Yoon - yyoo0548

Kuo Yuan - kyua4202

Tutor: Iwan Budiman

Faculty of Engineering

University of Sydney

October 8, 2019

Abstract

The image classification can be done by many different models. Furthermore, understanding the principles behind each method and carrying out acceptable performance are both important. The aim of this project is exploring image classifiers for the Fashion Mnist dataset which contains 10 different categories and gathering practical experience in doing so. We will implement three different classification models for the given data set and figure out the diversity between them.

Introduction

Object Classification is an important task within the field of computer vision. Based on the labeling of images into one of a number of predefined categories, classification including image sensors, image pre-processing, object detection, object segmentation, feature extraction and object classification.[1] At the same time, demands and uses of image classification methods are rapidly developing in all walks of life. For instance, there is a huge demand for this game-changing technique in the retail industry. For example, quality and price of a product can be compared with the help of an image recognition and classification techniques which compromises of numerous engines spanning across recognition, geometry, quality scoring for building the technology for the stores, range planning of products and many more.[2]

The goal of this study is to explore image classifiers for the Fashion Mnist dataset which contains grayscale images of the size 28x28 into a set of 10 different categories and consists of a training set of 30,000 examples and a test set of 2,000 examples. Here, we will carry out three different classification methods from scratch to classify given dataset and compare their performances.

Methods

Before starting analysing, we may want to transform and have the raw dataset into a more appropriate form so that we will be able to handle the data easier. First of all, we read given test and training data using h5py library and stored them into data_train/test and label_train/test, respectively. Since we were allowed to exploit the 2,000 test examples in the

test dataset, we reassigned the data_test with the first 2,000 test examples.

a. K-Nearest Neighbours

In this project, we have applied several algorithms to classify our tasks. One of them was K-Nearest Neighbours(KNN) algorithm. The basic theory of KNN is simple : "Similar things exist near to each other."[3] Spreading the whole data on a graph, the data points that have similar characteristics would close to each other with relatively shorter distance compared to other data points. Besides, we could catch the K-numbers of data groups naturally through a graph. Euclidian distance, which is also called the straight-line distance, is the most commonly used for calculating distance between data points, see Figure 1. Generally, KNN is

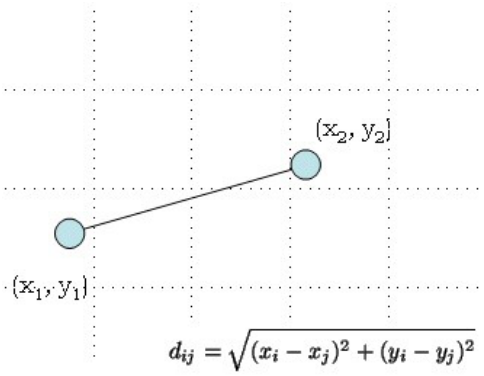

Figure 1: Euclidian distance

among the simplest technique of machine learning since it is relatively intuitive and easy to apply for the dataset.

KNN is non-parametric and also a type of lazy learning. Specifically, the non-parametric algorithm doesnt assume that the structure of a model is fixed and the model grows in size to accommodate the complexity of the data.[4] A parametric algorithm such as linear regression has assumptions that the data has to satisfy beforehand, contrastively. Also, non-parametric classifiers do not calculate class separation with the parameters. What we can see in lazy learning is that the function is only approximated locally and all the computations are performed,[5] when we do the actual classification. Moreover, this algorithm doesnt make a generalisation, hence all of the data for training is used as the test data when performing KNN.

When performing KNN, one of the most important steps for higher accuracy is selecting

the optimal K that rights for the data we have. To do this, we run KNN many times with different K until we find out the optimal K that makes the model has the lowest number of errors, thus improve the quality of the prediction.

Here are some reasons that we chose KNN for the classifier for our project. First, KNN is a quite successful method even with large training sets or the problems that have huge numbers of classes, despite its simplicity. Hence, KNN is generally used for image analysis or character recognition. Secondly, as mentioned earlier, KNN method is intuitive and also simple to implement without any training step. For instance, KNN simply adapts the new data and tags it to the neighbours that has the similarity the most with this new data.

b. Multinomial Logistic Regression

Another algorithm we have used in this project is Multinomial Logistic Regression. Multinomial logistic regression is a simple extension of binary logistic regression that allows for more than two categories of the dependent or outcome variable. Multinomial logistic regression uses maximum likelihood estimation to evaluate the probability of categorical membership[7] as binary logistic regression does so.

There are two common methods to perform multi-class classification using the binary classification logistic regression algorithm: One-vs-All (see figure 2)and One-vs-One (see figure 3).[8] In One-vs-All, we train C separate binary classifier for each class and run all those clas-

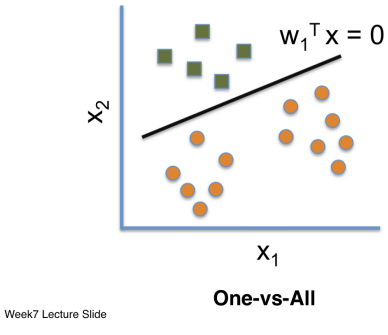

Figure 2: One vs All

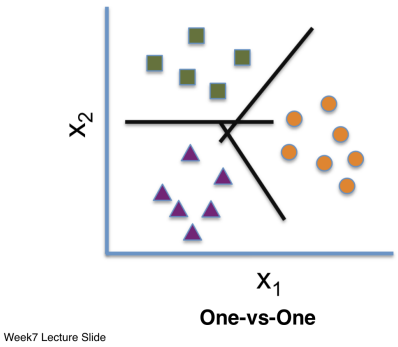

Figure 3: One vs One

sifiers on any new example x we want to predict and take the class with the maximum score. In one-vs-one, we train C choose 2 classifiers = $C(C-1)/2$ one for each possible pair of class and choose the class with maximum votes while predicting for a new example. Compared

with One-vs-All, One-vs-One performs worst when C is large.[9]In binary logistic regression, the value of cost function is limited between 0 and 1 by Sigmoid Function.[7] (see figure 4)
There is a modification of logistic regression using the $softmax$ function instead of the

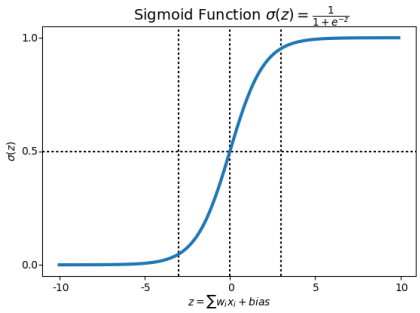

Figure 4: Sigmoid Function

$sigmoid$ function at multinomial logistic regression processing. By $softmax$ function, all values range in $[0,1]$ and the sum of the elements is 1.

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Moreover, cross entropy is a measure of how different 2 probability distributions are to each other. If p and q are discrete, we have :

$$H(p,q) = -\sum^{x} p(x)\, logq(x)$$

This function has a range of $[0, inf]$ and is equal to 0 when $p = q$ and infinity when p is very small compared to q or vice versa. For an example x, the class scores are given by vector $z = \mathbf{w}x + \mathbf{b}$, where $\mathbf{w}$ is a $c * m$ matrix and $\mathbf{b}$ is a length c vector of biases. We define the label y as a one-hot vector equal to 1 for the correct class c and 0 everywhere else. The loss for a training example x with predicted class distribution y and correct class c will be :

$$\hat{y} = softmax(z)$$

$$\begin{aligned} Loss &= H(y,\hat{y}) \\ &= -\sum^{i} y_i log\hat{y}_i \\ &= -log\hat{y}_i \end{aligned}$$

As in the binary case, the loss value is exactly the negative log probability of a single example x having true class label c. Thus, minimizing the sum of the loss over our training examples is equivalent to maximizing the log likelihood. We can learn the model parameters W and b by performing gradient descent on the loss function with respect to these parameters.

c. Artificial Neural Network

Lastly, we implemented Artificial Neural Network(ANN). ANN is inspired by biological neural networks that constitute animal brains.[10] For instance, in our project, the neural network for image recognition is defined as a set of input neurons, which are activated by pixels from the input image. First, the deformation and weighting of the function, which is determined by the person who made the neural network, is applied. And then, the activation of that neuron is transferred to another neuron. This processing is repeated until the last output neuron is activated, which depends on which image has been detected. Similar to other machine learning, neural networks are generally used to solve a wide range of problems, such as computer vision or speech recognition.

According to the model of the data set and result set, the input layer has $28 * 28 = 784$ nodes, and the output layer has 10 nodes, so we set the number of nodes in the hidden layer to 100. Eventually, the number of nodes is $(784, 100, 10)$. It is also possible to determine the size of coefficient (weight) w and intercept b of each layer.

When initializing our parameters, we set the intercept of each layer to 0, but in terms of coefficient matrix w, things are different. This is because if all the parameters are 0, then the output of all neurons will be in the same way, all the neurons in the same layer behave the same during the back propagation. In this case, the gradient is the same, the weight update is the same, so that the model cannot be gathered and the training cannot be completed. Here we used random initialization in a certain interval.

$$\mathbf{w} \sim U\left[-\sqrt{\frac{6}{dim_{n-1}+dim_n}}, \sqrt{\frac{6}{dim_{n-1}+dim_n}}\right], \mathbf{b} \sim U[0,0]$$

Regarding the activation, we don't make any changes in the input layer, it simply pass each sample data directly to the next layer. In the second layer (the hidden layer), the activation function is $tanh$. This is because the $tanh$ function is very sensitive in the interval $[-1,1]$

and it is non-linear, which we can easily see from the plot (figure 5) below.

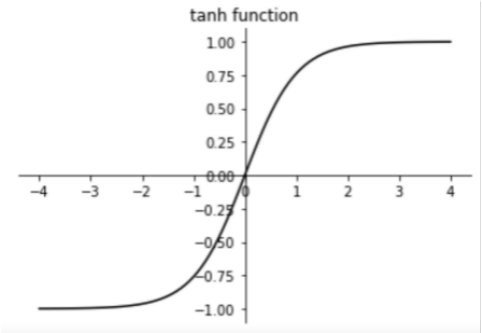

Figure 5: tanh function

The $tanh$ function is similar to $sigmoid$ function. It can be seen from the formula:

$$tanhx = \frac{sinhx}{coshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Their shapes are the same, but the scale and range are different. $Tanh$ is zero-centered, but still can be saturated. Because in the data set provided, the value of each pixel has been normalized, and after such a transformation, the characteristics of each pixel can be enlarged to improve the prediction accuracy of the trained model. In the last layer, the output layer, we let the activation function of this layer be the $softmax$ function.

$$\mathbb{P}\left(y = j \mid \Theta^{(i)}\right) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

Through the formula we can see that the final result is the probability of each predicting label with a sum of 1, which also makes it easier for us to calculate the loss function.Regarding the loss function, since the result of the prediction is a vector (one hot vector) whose length is 10 and sum is equal to 1, and each value represents the probability of each class, we can convert the real result also into a vector where all values are 0 except that the index corresponding to the label with a value of 1, also known as one hot vector. In this way, we can get the square loss of our current model by calculating the Euclidean distance between the predicted and actual vectors.

In forward propagation, since nothing changed in the input layer, the whole process can be defined as below:

$$\mathbf{L}_{0,in} = image$$
$$\mathbf{L}_{0,out} = image$$
$$\mathbf{L}_{n,in} = \mathbf{w}_n \mathbf{L}_{n-1,out} + \mathbf{b}_n$$
$$\mathbf{L}_{n,out} = A_n \mathbf{L}_{n,in}$$

When calculating the gradient, we need to do a forward propagation to calculate and store the input and output of each layer, and then start the reverse iteration. Starting from the last layer, for each layer, we will first Calculate the partial derivative of the loss function, and then multiply it by the partial derivative of the activation function of the layer, and finally multiply to the partial derivative of the input of the layer with variable coefficient and intercept. These can be defined as below:

$$\frac{\partial E_{total}}{\partial \mathbf{w}_{ij}} = \frac{\partial E_{total}}{\partial output_n} * \frac{\partial output_n}{\partial input_n} * \frac{\partial input_n}{\partial \mathbf{w}_{ij}}$$

Where the E_{total} means the total loss of layer n. The process of solving the gradient of b is very similar to w. Through iteration, the gradient calculation of the coefficient and intercept of each layer are completed.

In order to improve the reliability of the calculated gradient, our batch size is defined as 100, which means that every time 100 images are trained together to calculate the average gradient of these 100 images, and then update the parameters.

Experiments

To evaluate the performance of our algorithms, we seperate the train set into two parts: the first 4000 images as validation set and the rest 26000 images as training set.

a. K-Nearest Neighbours

Our KNN model for this project results in 85.05% accuracy with $K = 3$. We run KNN algorithm by repeatedly changing K value from 1 to 30 to capture the highest accuracy and the optimal K. Within the test range, we found that our accuracy fluctuated between 81.95% and

85.05%, and it generally tended to decrease when K is bigger than 3. A graph (see figure 6) below illustrates the changes of the accuracy by changing the number of $neighbours(K)$.

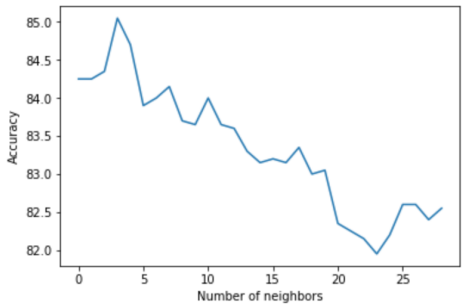

Figure 6: The accuracy by the number of neighbours

b. Multinomial Logistic Regression

Multinomial logistic regression achieves a higher accuracy (83.7%) through $Sigmoid$ Function, compared with using $Softmax$ Function (74.6%). To speed up the model, gradient descent is implemented. Moreover, according to the number of examples is large (30000 for training and 2000 for testing), we used One-vs-All method in order to transfer binary logistic regression method into multinomial logistic regression path.

c. Artificial Neural Network

In ANN processing, in order to get a trained model that optimizes in both time and accuracy, here we tested different learning rates and the number of repeated learning times (epoch numbers). Firstly, we set the learning rate to 0.8 with epoch number equal to 25.(see figure 7)There is a significant drop in the overall loss of validation set and it can achieve over 86 percent accuracy in the end. However, when we zoom into the last 10 epoch, the dramatic fluctuations indicate that our learning rate is too large and it means that we could not get the best solution.

Secondly, we set the learning rate to 0.1 with same epoch number. The validation loss significantly dropped with 82% of accuracy. Moreover, the loss in last 10 learning iterations dropped sharply (see figure 8). Hence, we found that our learning rate is too small and it

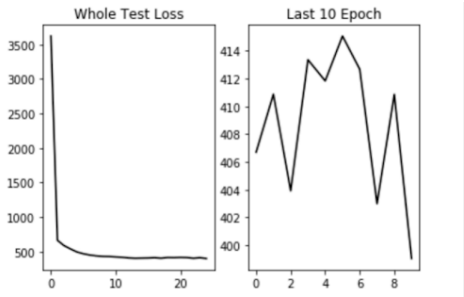

Figure 7: Validation loss and the last 10 epoch with learning rate 0.8

needs more learning to reach local optimal solution.

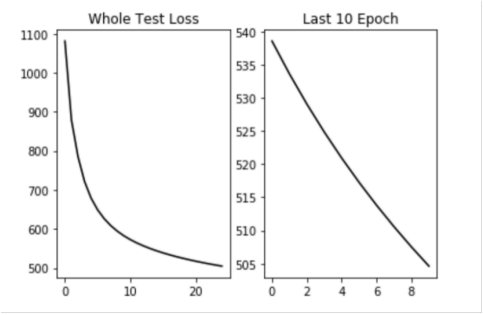

Figure 8: Validation loss and the last 10 epoch with learning rate 0.1

This time, we changed the learning rate to 0.5 with same epoch number (see figure 9). With accuracy reaching 86.8%, the validation loss of the last 10 epoch is still going down.

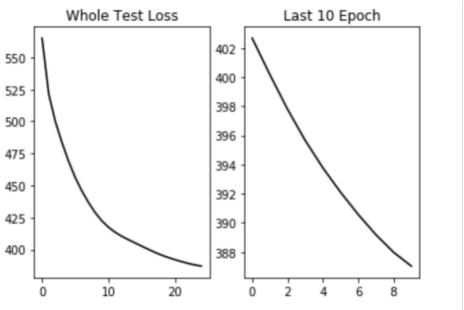

Figure 9: Validation loss and the last 10 epoch with learning rate 0.5

In light of these, we set the learning rate was set to 0.65 here. Interestingly, though the validation loss in last 10 epoch showed downtrend, the accuracy fluctuated in the meanwhile (see figure 10). This means that our predicted results are getting closer to the actual label.

We kept the learning rate at 0.65 and let our algorithm to train more times (epoch number as 50) (see figure 11). Although this configuration can achieve accuracy more than 87%,

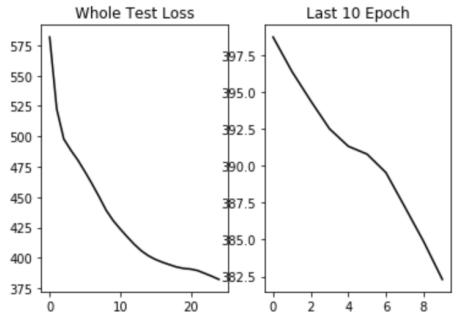

Figure 10: Validation loss and the last 10 epoch with learning rate 0.65

fluctuation with uptrend can be clearly identified in validation loss.

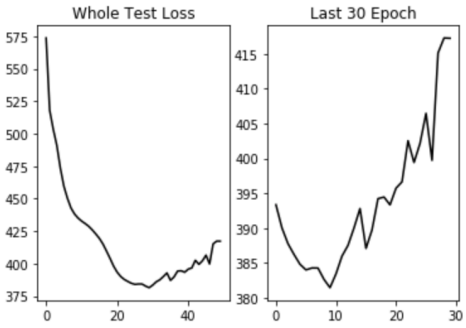

Figure 11: Validation loss and the last 30 epoch with learning rate 0.65

In conclusion, it is reasonable to set our learning rate as 0.5 with epoch time equal to 50. Under this configuration, our ANN algorithm achieved nearly 87% of accuracy within 9 minutes running time.

Algorithm	Highest Accuracy	Running Time
K-Nearest Neighbours	85.05%	under 10 seconds
Multinomial logistic regression	83.70%	around 90 seconds
Artificial Neural Network	87.55%	nearly 9 minutes

Table 1: Comparison of three classifiers

The table 1 above shows the accuracy and the running time for three different classifier methods we have carried out in this project. ANN resulted in the highest accuracy which was 87.55%. Whereas, there are drawbacks of ANN. Although the use of neural networks to train classification models can achieve relatively higher accuracy, the gradient descent method we used can only achieve a regional optimal solution because it cannot be convexly

optimized for its loss function. In this case, the accuracy of the trained classification model is limited. Furthermore, compared to other algorithms, neural networks may need more time to train a model. This may be more advantageous when the data set is larger. Eventually, for this given data, K-Nearest Neighbours algorithm could be more sufficient to meet certain accuracy requirements with relatively faster running time.

The experiments were run on :
Hardware - Processor : 2.6GHz Intel Core i7
Memory : 16GB 2400MHz DDR4,
Graphics : Radeon Pro 560X 4GB, Intel UHD Graphics 630 1536MB
Software - OS : MacOS Mojave 10.14.6
Python 3.7.3 (default, Mar 27 2019, 16:54:48),
IPython 7.6.1, Jupyter Notebook 6.0.0

Discussion

Picking up an optimal K value that works well is always one of the most challenging parts in KNN algorithm. Assuming $K = 1$, (see figure 6) then labeling of the test vector or image is determined by one element in the Train Set and it always results in zero error of the classification in the training set. On the other hand, if K is equal to the number of the training data, (see figure 6) then the label of the test vector is determined by all elements in the Train Set. Moreover, if the class is imbalanced, then every test data gets the same label. According to the experiment we have done here, we found that the error rate of the prediction was minimised and the model worked accurately, when K is 4. We figured out this value intuitively, from the result of the accuracy on independent validation data.

However, cross-validation could take over this experiment for parameter selection. Cross-validation is one of the fundamental and widely-used methods for picking parameters in a prediction. It accomplishes the result starting from partitioning the data into subsets. Next, it makes the analysis on one subset which is the training set and validates on the other subset called test set or validation. Cross-validation method sounds more sophisticated more than the way we performed, but it is computationally expensive in terms of extra training and test steps. Hence, we counted on the simple step as long as it fulfils a reasonable accuracy in prediction.

On the other hand, Multinomial Logistic Regression model runs in a short time, however correspondingly it achieved a lower accuracy compared with KNN. The $Softmax$ blows small differences out of proportion which makes our classifier biased towards a particular class that is not desired. The $Sigmoid$ function, but in the case of binary classification (One-vs- All), we would ultimately reach a better point. Thus, the One-vs-One or One-vs-All is a better approach towards multi-class classification using logistic regression.

Conclusion

This study indicates that the K-Nearest Neighbours algorithm achieved an acceptable accuracy with shortest running time among the classic algorithms. ANN achieves the highest accuracy although within the longest running time. The multinomial logistic regression results in the worst accuracy among these three algorithms. There are aspects for discussion and suggestions as following:

a. K-Nearest Neighbours

Calculating the distance between data points is essential to determine the similarity and dissimilarity. As mentioned before, we carried out the experiment using Euclidian distance and further work is subject to be suggested as the same algorithm progress with other distance calculation methods. A number of distance functions are used in KNN algorithm.

First of all, Euclidian distance measures the distance between two points in Euclidian space and is most widely used. Next, Manhattan distance is the distance between two points measured along axes at right angles.[11] Chebychev distance is calculated as the absolute magnitude of the differences between the coordinate of a pair of data points and is also called as Maximum value distance.[12]

Various distance functions can be used over the dataset and these are likely to vary the results of KNN performance. Thus, we need to take into account the choosing right methods for the distance calculation carefully.

b. Singular Value Decomposition and Principal Component Analysis

Although by experiments, both Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) reduce the accuracy of model due to the size of data is smaller, they still are meaningful machine learning techniques on implements. Especially when facing complex problems with big datasets, SVD and PCA could speed up model efficiently.

c. Artificial Neural Network

Adding hidden layers can be a practical way for improving the performance of Nerual Network algorithm. Another way is to try other activations like Sigmoid, ReLU and Maxout. It is worth mentioning that if the activation is ReLU, we need to make some adjustments to the sample data of the original dataset, because the ReLU function does not change in the positive interval, while the value in the negative interval becomes 0. With this activation, the sample image can be more characterized.

Since three different classifiers have been trained successfully, we can get more accurate prediction by voting from these three classifiers (Stacking of Ensemble Learning). In addition, we can train our final classifier through the AdaBoost (Adaptive Boost) algorithm to assign each original classifier a certain weight. This can make the final predictions more accurate even if our existing classifiers are not accurate enough. Obviously this may cost more time, however, we can lose a little bit accuracy of the original classifier to compensate for the loss of time.

Appendix

We submitted three classifiers in one ipynb file, and each classifier stays in one code box. The first code box is KNN that we suggested as the primary classifier for the accuracy test. As we have already found out what k is suitable in this task, here we directly assigned k as 3. By default, these codes will output the label of the first 5000 images. To change this configuration, the definition of test_number can be found at the top of the code. If the code executes as expected, success information will display below the code with how much time it has been taken.

Bibliography

[1] Pooja K, Sonam S, Sonu A, editors. A Survey on Image Classification Approaches and Techniques. International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 1, January 2013. Available from https://pdfs.semanticscholar.org/9509/c435260fce9dbceaf44b52791ac8fc5343bf.pdf

[2] Ambika C. (2019, January 25). *Uses Cases Of Image Recognition That We See In Our Daily Lives*. Available from https://analyticsindiamag.com/8-uses-cases-of-image-recognition-that-we-see-in-our-daily-lives/

[3] Pooja K, Sonam S, Sonu A. (2013). *A Survey on Image Classification Approaches and Techniques*.

[4] Security Dude. (2012). *Creating my first algorithm from scratch Euclidean distance and Pearson correlation*. Retrived from https://bigsnarf.wordpress.com/2012/03/25/creating-my-first-algorithm-from-scratch-euclidean-distance-and-pearson-correlation/

[5] Wikipedia contributors. (2019, April 17). Nonparametric statistics. In Wikipedia, *The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Nonparametric_statistics&oldid=892923104

[6] *k-Nearest-Neighbor Classifier*. Retrieved from https://www.python-course.eu/k_nearest_neighbor_classifier.php

[7] Jon S, Amanda KM. *Multinomial Logistic Regression*. Available from https://it.unt.edu/sites/default/files/mlr_jds_aug2011.pdf

[8] Bishop CM. (2016) *Pattern recognition and machine learning*. springer.

[9] Nguyen HT.(2019). Machine Learning and Data Mining. *Multiclass Classification* Retrieved from
https://canvas.sydney.edu.au/courses/17650/files/7903630/download?wrap=1

[10] Amari S. The handbook of brain theory and neural networks. MIT press; 2003.

[11] Ayush P. (2019). *Introduction to Logistic Regression*. Retrieved from
https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[12] Paul EB. (11 Fevurary 2019). *Manhattan distance*. Retrieved from
https://www.nist.gov/dads/HTML/manhattanDistance.html

[13] Punam M, Nitin T. (2013).*Analysis of Distance Measures Using K-Nearest Neighbor Algorithm on KDD Dataset*