American sign language understanding

Identify using SVM and neural network algorithm

Diogo Guedes, 76318, Gisela Pinto, 76397,
UC: Aprendizagem Automatica (Machine Learning), course instructor: Petia Georgieva

***Abstract*—Sign language is the most commonly used method for deaf people to communicate with others. However, understanding it is not an universal skill. For this reason, building a system that recognizes hand gestures and sign language can be very useful to facilitate the communication gap.**
The machine learning algorithms used were State Vector Machines, Logistic Regression and Artificial Neural Networks. Each algorithm and factors that provided the best performance are described in this paper.

***Index Terms*—Hand gesture, sign language,State Vector Machine, Logistic Regression, Artificial Neural Network.**

I. Introduction

Deaf is a disability that impairs the hearing, while mute is a disability that impairs the speaking. For the people affected by one of these disabilities the only way to communicate with others is by using sign language, where each gesture has a specific meaning.
Still the general public doesn't give the necessary importance in learning Sign Language. One solution to tackle the communication gap is by using the service of sign language interpreter, however it's very costly since people with this training are not easily found. So a cheap and easy solution is required so that deaf-mute and normal people can communicate normally.
Nowadays, researchers try to solve the problem as cheaply and as efficiently as possible. The breakthrough for this problem is the Sign Language Recognition System. This system aims to recognize sign language, and translate it to the local language via text or speech. However, building this system is very expensive and is difficult to for daily use. Another advancement was the use of classification algorithms using machine learning. This was used on pictures or real time images. However, it still has some shortcomings, especially in the tracking of hand movements.
The problem of developing sign language recognition ranges from the image to the classification process. Researchers are still trying to find the best method for the images. Gathering images using a camera gives challenges in regards to image pre-processing. Classification methods also give researches some drawbacks, by having to recognize that there are several sign languages.
This report aims to discuss the Sign Language Recognition using Machine Learning Classification algorithms. This report explains the data set used and various Machine Learning algorithms used for Sign Language recognition.

II. Data Description

The American sign language data set used is from a GitHub [1][2], and is accessible through the drive link here.
This data set is comprised in several images from A to Y. The letters J and Z aren't in this data set because they have motion. Every letter is represented with 240 to 250 images. In total, the data set has an average of 5832 images.
These images were taken with a smartphone camera, some images represent some problems purposely because they are cross validation images. This problems are:

- Quality of images: some images are pixelated or poor on light;
- Aren't centered: some images are on the left or right side;
- Images rotated: some are more rotated that others.

In this data set, 30% of the images of each letter are for testing, and 70% are for training.

In order to reduce the impact of processing the dataset raw, the following pre-processing steps were applied to each image:

- image resize
- grayscaling

These operations are equally applied to the test images.

III. Algorithms used

Classification machine learning techniques like SVM, Neural Networks and Logistic Regression [3] are used for supervised learning, which involves labeling the data set before feeding it into the algorithm for training. Feature extraction algorithms are used for dimensionality reduction in order to create a subset of the initial features such that only important data is passed to the algorithm. Grayscaling and image resize were used for this purpose.

A. State Vector Machine

Support vector machines is a supervised learning method with learning algorithms that analyze data used for classification, regression analysis and outliers detection.[4]
In SVM, each data point is plotted in an n-dimensional space (n is the number of features) with the value of each feature being the value of a particular coordinate. The classification is done by finding a hyperplane that better differentiates the classes. A hyperplane is a decision plane which separates a set of objects having different class memberships.
The advantages of SVM are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

- Versatile: different Kernel functions can be specified for the decision function.

The disadvantages of SVM include:

- If the number of features is much greater than the number of samples, avoiding over-fitting in choosing Kernel functions and the regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

For this project a python version of SVM was created and uses the sklearn library (in use is sklearn.svm). SVC has the kernel set to Linear. A kernel transforms an input data space into the required form. A linear kernel can be used as a normal dot product.
The implementation is based on libsvm and multi-class support is handled according to a one-vs-one scheme. The objective of a Linear SVC (Support Vector Classifier) is to fit to the data provided, returning a best fit hyperplane that divides, or categorizes, the data. From here, after getting the hyperplane, it's important then feed some features to the classifier to see what the predicted class is.
The implementation was done as follows [5]:

- Load Data: First the data set was loaded with all the transformation on each image.
- Splitting Data: It was necessary to divide the data set into a training set and a test set. The separation of the dataset was done by the function train test split on model selection library from sklearn. The dataset is broken down in two parts with a ratio of 70:30. It means 70% of the data will be used for model training and 30% for model testing.
- Generating Model: The SVM module was imported and the support vector classifier object was created by passing the argument kernel as the linear kernel in SVC() function. Then, the model was fited on train set and performed prediction on the test set.
- Evaluating the Model: The accuracy was computed by comparing the actual test set value and predicted value Figure 1.

In the end, was calculated accuracy of the model, Precision from each letter, Recall and F1-score. With this implementation, the accuracy of this model is 96,29%. The Precision, Recall and F1-score are in the Table I.

It was also calculated the Micro Average, Macro Average and Weighted Average. So, Micro Average is a method that sums up the individual true positives, false positives and false negatives of the model for different sets and then applies them to get the statistics. In this case the result for precision, recall and f1 score in Micro average is 99%. Macro Average method take the average of the precision and recall of the system. In this case the result is equally 99%. Weighted Average is a method that calculate metrics for each label and find their average weighted by support (the number of true instances for each label). In this case the result for precision, recall and f1 score in Weighted average is 99%.

The support for each letter is on figure 2.

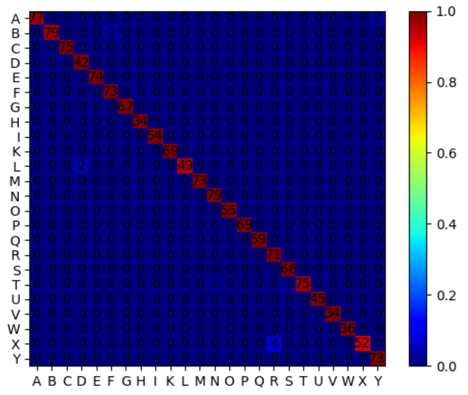

Fig. 1. Confusion matrix on SVM model

Letter	Precision	Recall	F1-score
A	1.00	0.97	0.99
B	1.00	0.98	0.99
C	1.00	1.00	1.00
D	0.95	1.00	0.98
E	1.00	1.00	1.00
F	0.97	1.00	0.99
G	1.00	1.00	1.00
H	1.00	1.00	1.00
I	1.00	1.00	1.00
K	1.00	1.00	1.00
L	1.00	0.96	0.98
M	1.00	1.00	1.00
N	0.99	1.00	0.99
O	1.00	1.00	1.00
P	1.00	1.00	1.00
Q	1.00	1.00	1.00
R	0.95	1.00	0.97
S	0.99	1.00	0.99
T	1.00	0.97	0.99
U	1.00	1.00	1.00
V	1.00	1.00	1.00
W	1.00	1.00	1.00
X	1.00	0.93	0.96
Y	0.97	1.00	0.99

TABLE I
RESULTS ON SVM

B. Logistic Regression

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables. [6]

The advantages of Logistic Regression are:

- Efficient and Straightforward nature.
- Doesn't require high computation power.

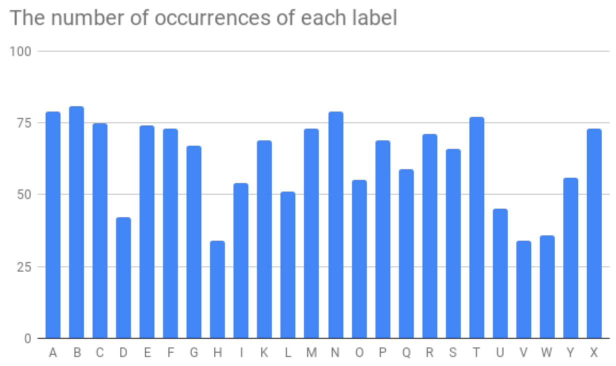

Fig. 2. The number of occurrences of each label

- Easy to implement.
- Easily interpretable.
- Used widely by data analyst and scientist.
- Doesn't require scaling of features.

The disadvantages of Logistic Regression include:

- Is not able to handle a large number of categorical features/variables.
- It is vulnerable to overfitting.
- Can't solve the non-linear problem.
- Will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to eeach other.

So, in the multiclasse case, we set the parameter multi class on Logistic Regression function sklearn from linear model to multinomial [7]. This means that the training algorithm uses the cross- entropy loss. This class, also, implements regularized regression using the 'liblinear' library with 'lbfgs' solvers. L-BFGS is an optimization algorithm in the family of quasi-Newton method that approximates the Broyden-Fletcher-Goldfarb-Shanno(BFGS) algorithm using a limited amount of computer memory. It is a popular algorithm for parameter estimation in machine learning. The algorithm finds a local minimum of an objective function, making use of objective function values and the gradient of the objective function. That level of description covers many optimization methods. [8]
The sigmoid function, also called logistic function gives an S shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO.
The implementation is very similar to SVM algorithm. The following are the implementation steps:

- Load Data: First of all, it was loaded the data set with all the transformation on each image.
- Splitting Data: It was necessary dividing the data set into a training set and a test set. Split the data set by using the function train test split on model selection library from sklearn (same as SVM model). The dataset is broken into two part in a ratio of 70:30. It means 70% data will be used for model training and 30% for model testing.
- Model Development and Prediction: Imported the Logistic Regression module and created a Logistic Regression classifier object using LogistiRegression() function from sklearn. Then, the model was fited on the train set using fit() and perorm prediction on the test set using predict().
- Model Evaluation using Confusion Matrix: A confusion matrix is a table that is used to evaluate the performance of a classification model. It also allows visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.
- Visualizing Confusion Matrix (Figure 3) using Heatmap: The result of the model in the form of a confusion matrix using matplotlib and seaborn.

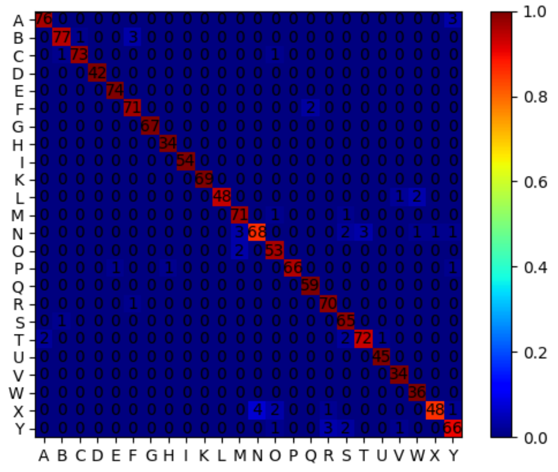

Fig. 3. Confusion Matrix for Logistic Regression model

In the end, was calculated accuracy of the model, Precision from each letter, Recall and F1-score. With this implementation the accuracy of this model is 91,37%. The Precision, Recall and F1-score are in Table II.

It was also calculated Micro Average, Macro Average and Weighted Average here too. In Table III are the result:

The support for each letter is in figure 4.

C. Artificial Neural Network

Artificial Neural Network is a Machine Learning paradigm that tries to replicate the data processing that's performed by the brain. [9] A Neural Network architecture is commonly divided into 3 parts:

- Input Layer, which receives the training and testing data for processing
- Hidden Layer, which is comprised of 1 or more layers

Letter	Precision	Recall	F1-score
A	0.97	0.96	0.97
B	0.97	0.95	0.96
C	0.99	0.97	0.98
D	1.00	1.00	1.00
E	0.99	1.00	0.99
F	0.95	0.97	0.96
G	1.00	1.00	1.00
H	0.97	1.00	0.99
I	1.00	1.00	1.00
K	1.00	1.00	1.00
L	1.00	0.94	0.97
M	0.93	0.97	0.95
N	0.94	0.86	0.90
O	0.91	0.96	0.94
P	1.00	0.96	0.98
Q	0.97	1.00	0.98
R	0.95	0.99	0.97
S	0.90	0.98	0.94
T	0.96	0.94	0.95
U	0.98	1.00	0.99
V	0.94	1.00	0.97
W	0.92	1.00	0.96
X	0.98	0.86	0.91
Y	0.92	0.90	0.91

TABLE II
RESULTS ON LOGISTIC REGRESSION

	Precision	Recall	F1-score
Micro Average	0.96	0.96	0.96
Macro Average	0.96	0.97	0.97
Weighted Average	0.96	0.96	0.96

TABLE III
AVERAGE RESULTS ON LOGISTIC REGRESSION

- Output Layer, which the number of nodes depends on the Machine Learning problem

The classification problem being analysed in this research involves the distinction between 24 different types of images, so the number of output nodes must be 24.
The number of Input Layer nodes corresponds to the size of the images in the dataset. Originally it was decided to pre-process the dataset to be of size 30x30 pixels. Even though this resulted in faster processing, important details of the images were removed, which could result in less accurate classifications for images not in the dataset. To avoid this problem, the 100x100 pixels was used instead.

The image data was normalized and fed to the neural network for training.

Implementation

The neural networks training and testing was done with the class Matlab code as a base. In order to the image dataset in matlab, it was necessary to upload and convert all the images into a matrix X and create a matrix y. The matrix X has an image for each of its rows, and each columns corresponds to one pixel. Before create the matrix X, the images needed to be rotated in order to be visualized properly with the function for displaying one hundred images.

The activation function used was the one from the class, the Sigmoid Activation.

All the comparison tests made had, when feasible, the same initial weights and all the parameters the same except the one being tested.

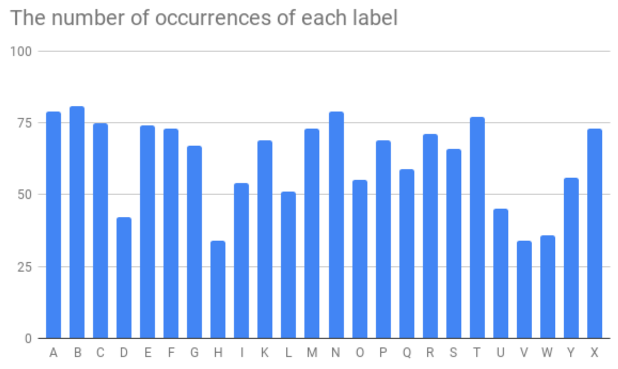

Fig. 4. The number of occurrences of each label

Note: By distraction I forgot to split the dataset matrices X and y in two parts (70:30), so the tests where the global accuracy was measured used the dataset that was used to train the neural network.

Mini-Batch Learning

The training dataset used has almost 5000 examples. To improve the training performance of the neural network, the mini-batch learning process was used. To decide on the size to use for each batch, 3 alternatives were tested:

- Batch of 32 examples
- Batch of 128 examples
- Batch of 256 examples

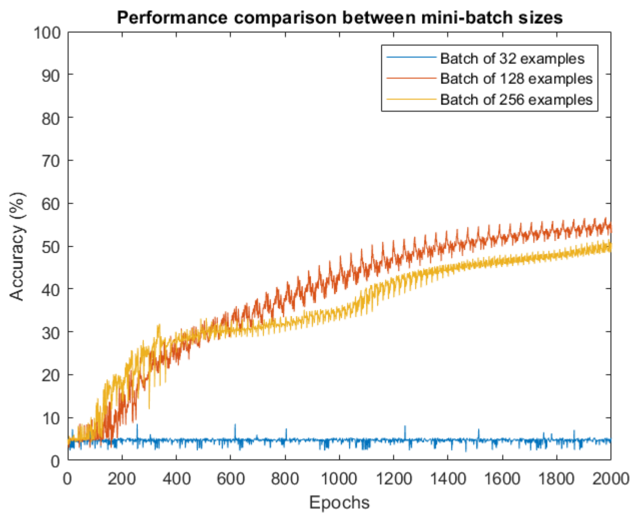

Fig. 5. NN performance for different batch sizes

Note: To compare the impact of the batch sizes, the same initial weights were used. In this graph each epoch correspond to the optimization of one batch of the dataset.
According to the graph in Figure 5, the batch that resulted in the greatest accuracy after 2000 epochs was the batch with 128 examples.
It can also be concluded that the more times the entire dataset is passed through the optimization step, the more accurate the results will be. In the case of the 32 examples batch, in 2000 epochs, the entire dataset only passes 12 times through

optimization. In the case of the 128 examples, the entire datasets passes 48 time and for the 256 examples batches it passes 96 times. However there's a limit which can be seen for the 256 examples batch. If the size of a batch surpass a certain threshold, the learning performance of the NN will begin to suffer.

Dataset Organization

In batch learning the impact of the dataset organization is less pronounced because the entire dataset is optimized at the same time, at each optimization iteration, but that's not the case for mini-batch learning. The following graph plots the evolution of the accuracy (each epoch correspond to the optimization of one batch of the dataset) through 2000 epochs for a dataset which has the the images of the same letters all following each other and a dataset with the dataset randomized.

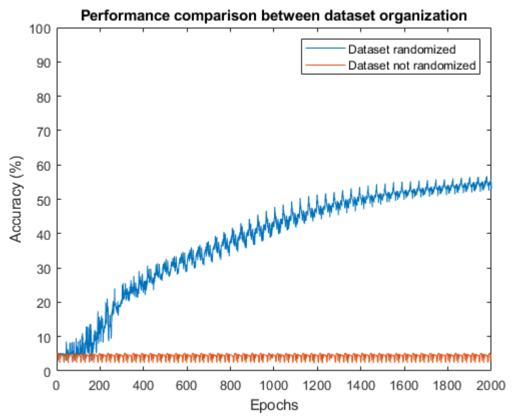

Fig. 6. NN performance for randomized and not randomized training datasets

It can be seen that the NN learning rate is stuck for the non-randomized dataset. This is because the NN optimizes the network in one iteration for one type of image. In the next iteration a new type of image may be fed to the optimization algorithm, which will drastically affect the cost and the accuracy and the network will optimize only for this new type.

By randomizing the entries of the dataset, it becomes more homogeneous. This proves to be effective due to various types of images being fed, in one batch, into the optimization algorithm. This allows for the optimization of the network for various types of images at each optimization iteration.

Neural Network regularization

To understand how regularization affects the performance of the neural network, various training simulations were made with different lambda values. In the graph of Figure 7 it can be seen that regularization has a relatively big impact on the resulted accuracy of the network, where the biggest accuracy was around 62% and the smallest was around 52%.

Number of Hidden Layer nodes

To test the impact of the number of hidden layer nodes on the learning rate of the network, for dimensions were tested: 25, 50, 100 and 400 nodes. The graph of Figure 8 shows the accuracy of the network for the batches of 125 examples over 1000 epochs.

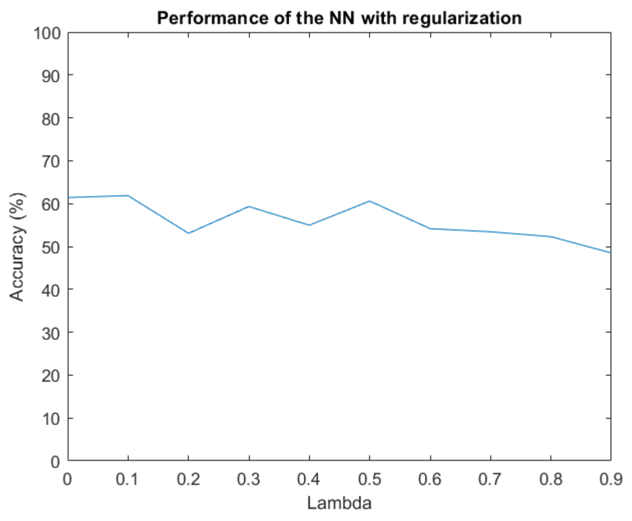

Fig. 7. NN performance with regularization

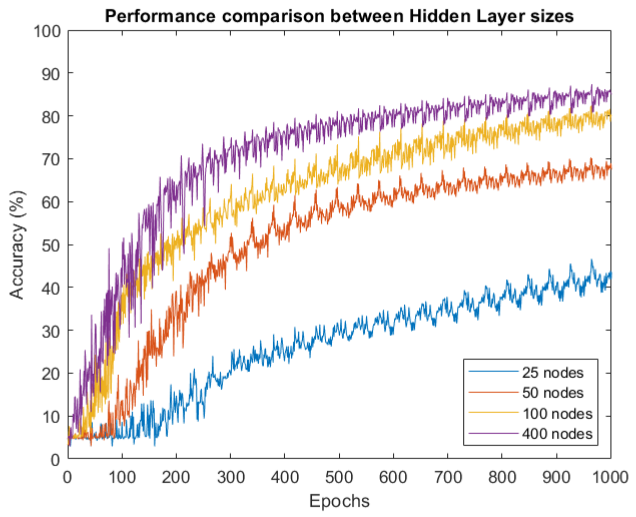

Fig. 8. NN performance with different number of HL nodes

The obtained accuracy for the entire dataset of each simulation can be seen in the following table:

Hidden Layer Size	Accuracy(%)
25 nodes	53.027560
50 nodes	67.431100
100 nodes	79.863207
400 nodes	84.952726

TABLE IV
NEURAL NETWORK ACCURACY FOR EACH HL SIZE

So even though the accuracy of the hidden layer increases significantly as the number of hidden layer nodes increases, at each step the number of nodes necessary for a significant improvement also increases.

IV. Model Selection

After running all the algorithms and analyzing the results it was concluded that the best algorithm is the State Vector Machine.
In summary:

- SVM has an accuracy of 96,29%.
- Logistic Regression has an accuracy of 91,37%.
- Neural Network has an accuracy of 84.9%.

V. Conclusion

From this report, one can extract several conclusion. The first one is that, when using a small set of features, using logistic Regression is a bit more efficient than using Neural Network or SVM. For a more bigger data set in this case SVM was better. Neural Network has a more efficiency for much bigger data set because de computing. In this case Neural Network was not a good classification algorithm since it has the worst accuracy, it can be because it's images and a low data set or because it was implemented in matlab and not python.
Another important conclusion is that state vector machines attained optimal and worst performances with the same number of relevant features.
In a future implementation, maybe we can expand the data set to a real time image data set instead of static image data set. In this case Neural Network can be more efficient since the data set is bigger and with more details to keep in attention.

VI. Work Division

Gisela Pinto: Report, Algorithms(SVM and Logistic Regression Python), Presentation Slides.
Diogo Guedes: Report, Algorithm(Neural Network Matlab), Presentation Slides.

References

[1] A. Singh. [Online]. Available: https://github.com/Anmol-Singh-Jaggi/Sign-Language-Recognition.
[2] [Online]. Available: https://github.com/hthuwal/sign-language-gesture-recognition.
[3] [Online]. Available: https://www.statisticssolutions.com/what-is-logistic-regression/.
[4] A. Navlani. [Online]. Available: https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python.
[5] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.htmln.
[6] [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression.
[7] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
[8] [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046403000340.
[9] [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.