Spoken digit recognition in Arabic: SVM vs MLP

Pablo Quinoa
pablo.quinoa@city.ac.uk

Abstract

This paper aims to present a critical evaluation and comparison of two algorithms, Support Vector Machines (SVM) and Multilayer Perceptron (MLP), on a supervised classification time series task which consists on classifying digits (0 to 9) that are spoken in Arabic. The evaluation methods used to find the best fit for this task consists on measuring the accuracy, comparing their confusion matrices and time to predict.

1. Introduction

Speech recognition plays a very important role nowadays, and engineers all around the world are trying very hard to improve such technology for its multiple applications across all domains. In this paper we are going to focus on the recognition of single digits (0 to 9) being spoken in Arabic. Developing systems to automatically recognise someone speaking on the phone their passport number or credit card number accurately is crucial, therefore these systems need to be very accurate.

The purpose of this paper is to find or critically compare which algorithm is better for the task of recognising spoken digits. For this, we will analytically compare given a dataset of spoken digits in Arabic, both Support Vector Machine (SVM) and Multilayer Perceptron (MLP).

2. About the dataset

The dataset used to analytically compare both SVM and MLP is one where the data was collected by the Laboratory of Automatic and Signals in the University of Badji-Mokhtar in Algeria [1]. This dataset contains time series of mel-frequency cepstrum coefficients (MFCCs) corresponding to the 0 to 9 spoken digits in Arabic. There are a total of 8800 spoken digits or rows, where each row consists of up to 93 frames, where each frame contains 13 MFCCs. This makes a total of 1209 attributes (13x93) where each attribute is a discrete variable. The dataset is clean (no missing values) and is also balanced, the 8800 rows consist of 880 rows for each of the 10 digits and those were recorded by an even amount of men and women.

2.1. Pre-processing on the dataset

Although the dataset was clean and well-structured for analysis, some pre-processing was required before we could implement a time series digit recognition classifier. First, we created a column for the labels (the corresponding digit for each row) as this one was missing (the documentation for the dataset explains how the first x rows are for 0, next block of rows for digit 1…).

Due to the nature of time series data and in this case speech, some rows or spoken digits were longer than the others. Either because someone speaks faster or because the digit is a shorter word, as we can see in Figure 1 below.

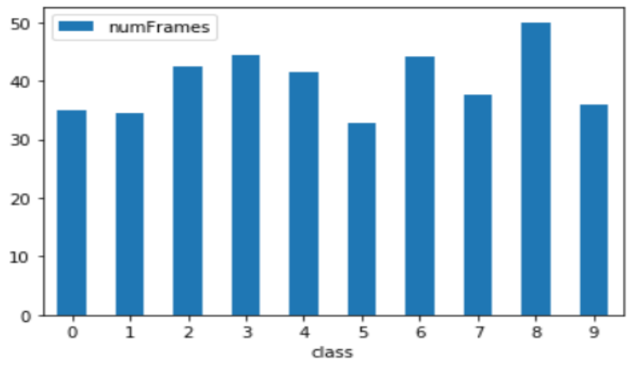


Figure 1 - Average number of frames per spoken Arabic digit in the dataset

The shorter row in the dataset consists of 4 frames (4 x 13MFCCs = 52 attributes) and the longest consists of 93 frames (93 x 13MFCCs = 1209 attributes), therefore we padded 0's on all of those rows that were shorter than the longest row to make all rows equivalent in number of columns or attributes. Therefore, we ended with a dataset consisting of 8800 rows and 1209 columns, each row properly labelled with a 0 to 9 digit.

3. Models

3.1. Support Vector Machines (SVM)

SVM is a supervised model that can be used for classification or regression, and it works by separating two classes that might be linearly separable, but if they are not it can use what it's called *Kernel trick* mapping the inputs to higher dimensions where these two classes might be separable. Although it works by separating only two classes (binary classification), techniques have been developed to cope with multi class classification, techniques such as One-Against-All or One-Against-One. The objective of SVM is to find the best hyperplane with the highest margin distance between the two nearest points of each class, this way given the largest margin we end with a lower generalization error.

PROS	**CONS**
- Good performance on big number of features. - They are very accurate on high dimensional spaces. - The outliers have less impact.	- Slow on big datasets. - Can cause overfitting when the number of features is much higher than the number of samples. - Returns the classified class, instead of probability estimates of belonging to a class.

3.2. Multilayer Perceptrons (MLP)

MLP is a neural network which is composed of more than one perceptron. A perceptron tries to reproduce one of our brain neurons, and basically it takes some inputs which are then multiplied by some weights (weights to represent the strength of a particular input). Then we add all those inputs weights together to get a weighted sum, which is then applied to an activation function of our choice (hyperparameter of MLP) to get some outputs. Hence a perceptron classifies input by separating two categories with a straight line. The impossibility to solve nonlinear problems, such as the XOR problem, introduced what we know by Multilayer Perceptrons which in fact is just a network with more than one perceptron.

The training of an MLP consists of adjusting the weights and biases (using backpropagation) in all the perceptrons of the network in order to minimize the error, so to find the best dependencies between the inputs and outputs.

PROS	CONS
- Computationally expensive. - Re-uses pre trained layers. The lower layer weights of a model can be re-used so that a higher layer can use the model of what a lower layer already trained. - Uses fast optimizers like the Momentum, to adjust all weights in the network. - Neuron weights can be regularized to fix overfitting problems.	- Is a black box. So, it is difficult to analyse what happens inside a network. - Computationally expensive. - Requires a big amount of data to give some decent results. - Large number of hyperparameters to set and tune.

4. Hypothesis statement

There is no such thing as one best algorithm when it comes to comparing SVM with MLP, not even for specific applications. Different results in performance have been observed across different papers, some stating MLP had higher performance and some saying the opposite. Thus, training the best models of each algorithm on a given dataset and comparing them like we do in this paper is the only way to find out.

What seems consistent across all studies on using MLP or SVM for digit recognition in any language, is that they achieve accuracies in the order of 90%. [2] Achieves an average of 93% accuracy on all digits on the same dataset used in this paper. On the other hand [3] achieve its better performance at 94.9% using a Gaussian Kernel SVM, this time using a different dataset with 0 to 9 digits in English.

SVM is probably the most used algorithm when it comes to speech recognition or pattern recognition, but MLP and neural networks in general are increasing its popularity and are widely used as well for the studied domain.

5. Training, evaluation methodology, choice of parameters and experimental results

First, we went through a phase of hyperparameter tuning for each of the algorithms compared in this paper. We used a 70-30 split partition, where 70% of the data was used to find the best model hyperparameters and later train the best models, and 30% was kept out for later testing and evaluation.

For SVM a grid search using cross validation was performed to find the combination of hyperparameters that would give us the lowest validation error. Grid search was our choice for hyperparameter optimization because in the case of SVM the number of parameters that we want to train on is small, thus creating manually a grid of parameters was preferred.

In the case of MLP, Bayesian optimization was used to find the best hyperparameters. Bayesian optimization can handle more hyperparameter than grid search, it is just faster and more efficient. Also, it keeps track of past evaluations (avoids wasting time on low performing hyperparameters) and will consider the last combination's performance to try to build another combination within that region of parameters where the best ones are. We set the Bayesian Optimizer to 300 function evaluations, and as we can see in Figure 3 below after around 175 iterations we reach a plateau where the validation error stays at around 0.03.

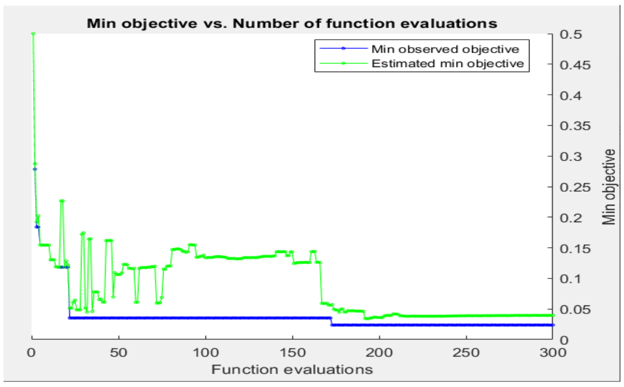


Figure 2 – Min objective error vs number of function evaluations for Bayesian Optimization used in MLP

For SVM we created a grid of parameters which consists of 3 kernel possible functions (linear, rbf or polynomial), 5 constraint box values between 0.1 and 100, and finally 5 gammas between 0.001 and 100. The combination of these parameters with a lower cross validation error was chosen as the best hyperparameters, in this case a polynomial function of order 2, with a box constraint of 1 and a gamma of 10. In addition to the grid search, we run an experiment to find which coding technique (one vs one or one vs all to allow multi class classification) resulted in better performance, and one vs all was found to be better.

In MLP we set 6 optimizable variables to fit into Bayes Optimizer. Network depth, number of hidden neurons, learning rate, momentum, 6 different train functions (traingda,trainrp, traingdm, traingdx, trainscg, trainoss) and 4 transfer functions (softmax,logsig,purelin,tansig). The result of running Bayes Optimizer with these values, returned the best combination of hyperparameters, which in this case was 2 layers, 18 hidden neurons per layer, a learning rate of 0.01, a momentum of 0.85, trainscg train function and tansig transfer function. In addition to the Bayes Optimization, in other experiments we found that 1000 epochs were the optimal training parameter for MLP, so we used this value for training.

Finally, in a last experiment we plotted some learning curves (see Figure 4 below) with the trained and validation accuracy for both SVM and MLP using 8 different size of samples, the smaller being 250 samples all the way up to 6600 which was all the training data. We observed how with 4500 samples of data the accuracy reaches its maximum. We can also notice how MLP overfits a bit, has a higher generalization error on validation on unseen data.

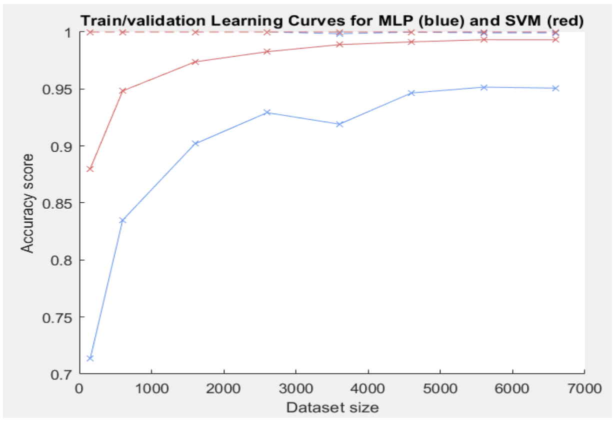
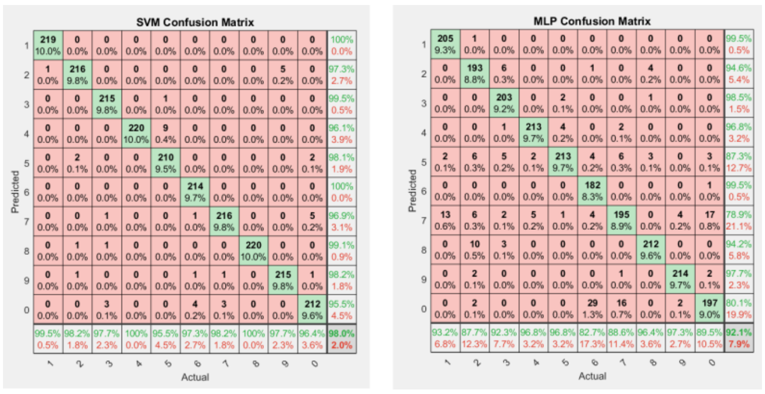


Figure 3 – Validation (full lines) and train (broken lines on the top close to 1 accuracy) learning curves for MLP in blue and red for SVM across different dataset sizes

6. Analysis and critical evaluation of results

After training a model with the best hyperparameters for both SVM and MLP using the training data (70%), we evaluated the two models using the test data this time (30%) in order to see how they perform on some unseen data.

Accuracy was found to be slightly different between the two, SVM achieved a 98.04% accuracy across all the test set whereas MLP got a 92.13%.



SVM Confusion Matrix

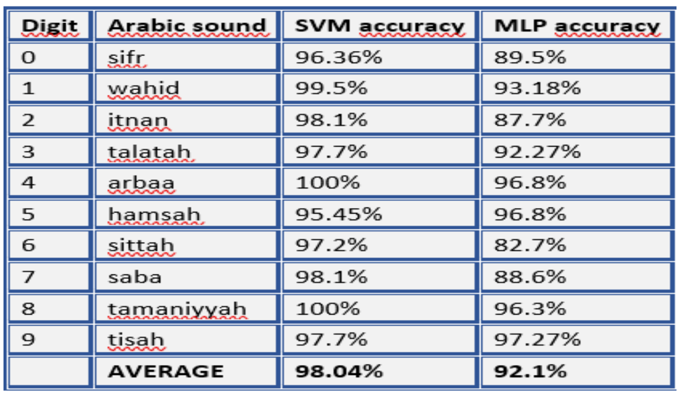
Predicted \ Actual	1	2	3	4	5	6	7	8	9	0	
1	219 10.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	1 0.0%	216 9.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.2%	0 0.0%	97.3% 2.7%
3	0 0.0%	0 0.0%	215 9.8%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
4	0 0.0%	0 0.0%	0 0.0%	220 10.0%	9 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96.1% 3.9%
5	0 0.0%	2 0.1%	0 0.0%	0 0.0%	210 9.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	98.1% 1.9%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	214 9.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	1 0.0%	216 9.8%	0 0.0%	0 0.0%	5 0.2%	96.9% 3.1%
8	0 0.0%	1 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	220 10.0%	0 0.0%	0 0.0%	99.1% 0.9%
9	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	215 9.8%	1 0.0%	98.2% 1.8%
0	0 0.0%	0 0.0%	3 0.1%	0 0.0%	0 0.0%	4 0.2%	3 0.1%	0 0.0%	0 0.0%	212 9.6%	95.5% 4.5%
	99.5% 0.5%	98.2% 1.8%	97.7% 2.3%	100% 0.0%	95.5% 4.5%	97.3% 2.7%	98.2% 1.8%	100% 0.0%	97.7% 2.3%	96.4% 3.6%	98.0% 2.0%

MLP Confusion Matrix

Predicted \ Actual	1	2	3	4	5	6	7	8	9	0	
1	205 9.3%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
2	0 0.0%	193 8.8%	6 0.3%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	4 0.2%	0 0.0%	0 0.0%	94.6% 5.4%
3	0 0.0%	0 0.0%	203 9.2%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	98.5% 1.5%
4	0 0.0%	0 0.0%	1 0.0%	213 9.7%	4 0.2%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	96.8% 3.2%
5	2 0.1%	6 0.3%	5 0.2%	2 0.1%	213 9.7%	4 0.2%	6 0.3%	3 0.1%	0 0.0%	3 0.1%	87.3% 12.7%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	182 8.3%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	99.5% 0.5%
7	13 0.6%	6 0.3%	2 0.1%	5 0.2%	1 0.0%	4 0.2%	195 8.9%	0 0.0%	4 0.2%	17 0.8%	78.9% 21.1%
8	0 0.0%	10 0.5%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	212 9.6%	0 0.0%	0 0.0%	94.2% 5.8%
9	0 0.0%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	214 9.7%	2 0.1%	97.7% 2.3%
0	0 0.0%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	29 1.3%	16 0.7%	0 0.0%	2 0.1%	197 9.0%	80.1% 19.9%
	93.2% 6.8%	87.7% 12.3%	92.3% 7.7%	96.8% 3.2%	96.8% 3.2%	82.7% 17.3%	88.6% 11.4%	96.4% 3.6%	97.3% 2.7%	89.5% 10.5%	92.1% 7.9%

Figure 4 – Confusion matrices for SVM on the left, MLP on the right

As we can see in the confusion matrices from Figure 4 above, SVM has most of the test samples in the diagonal of the matrix (correctly classified). Although MLP also has most of its samples in the diagonal, it has many more samples above and below the diagonal (misclassified).



Digit	Arabic sound	SVM accuracy	MLP accuracy
0	sifr	96.36%	89.5%
1	wahid	99.5%	93.18%
2	itnan	98.1%	87.7%
3	talatah	97.7%	92.27%
4	arbaa	100%	96.8%
5	hamsah	95.45%	96.8%
6	sittah	97.2%	82.7%
7	saba	98.1%	88.6%
8	tamaniyyah	100%	96.3%
9	tisah	97.7%	97.27%
	AVERAGE	**98.04%**	**92.1%**

Figure 5 - Accuracies for each digit on both SVM and MLP

When we analysed the accuracies for each different digit, we can also see in Figure 4 how for basically all digits SVM is more precise, and probably for the nature of the syllables in Arabic we found that 0 and 6 were the ones that the system found harder to recognize (especially when using MLP).

Also, the prediction time was monitored as speech recognition systems are usually used on real time, thus time becomes a very important factor. Both SVM and MLP were very fast on predicting 2200 test samples, it took around 1 second on the SVM trained model, and for MLP it took 0.23 seconds (5 times faster). When it comes to training MLP was found to be 3 times slower than SVM, but training time would usually be offline on real time systems, then training time is less important to consider.

7. Conclusions

We can conclude that for spoken digit recognition and in this particular case the Arabic one studied in this paper SVM is more precise (around 6% more) than MLP, and offers a precision that makes it suitable for any type of speech recognition application. Although MLP was found to be faster to classify, SVM is fast enough, thus SVM would still be our preferred choice.

Given that SVM would be our choice for any system of this nature, in future investigations we would like to apply Platt's probabilistic outputs for SVM like explained in [4]. This way we would not only get the predicted class for a given sample but also the probability of belonging to that class. Therefore, we would be able to study for each digit what are the other digits that the system is confusing with (e.g. 'arbaa' sounds very similar to 'saba').

8. References

[1] Prof.Mouldi Bedda, 2008. *Spoken Arabic Digit Dataset from UCI Repository of machine learning databases* [http://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit].
[2] Nacereddine Hammami, Mouldi Bedda, 2010. *Improved Tree Model for Arabic Speech Recognition.*
[3] Issam Bazzi and Dina Katabi, 2000. *Using Support Vector Machines for Spoken Digit Recognition.*
[4] Hsuan-Tien Lin, Chih-Jen Lin, Ruby C. Weng, 2007. *A Note on Platt's Probabilistic Outputs for Support Vector Machines.*

Appendix 1 – glossary

Mel-frequency cepstrum coefficients (MFCCs): representation of the short-term power spectrum of a sound. They are common features in speech recognition.

Generalization error: is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

Activation or transfer function: in neural networks, the activation function of a node defines the output of that node given an input or set of inputs.

Box constraint: a parameter that controls the maximum penalty imposed on margin-violating observations, and aids in preventing overfitting (used in SVM).

Gamma (SVM parameter): a parameter that defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.

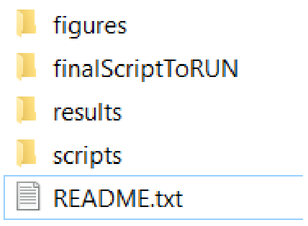
Momentum: momentum methods in the context of neural networks refers to a group of techniques designed to speed up the convergence optimization methods like gradient descent.

Epoch: in neural networks, an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs.

Perceptron: In neural networks, a perceptron is an artificial neuron using the step function as the activation function. The perceptron algorithm is also termed the single-layer perceptron, to differentiate from multilayer perceptron.

Appendix 2 – Implementation details

Together with this paper, the scripts and data used for all experiments can be found. There are 4 folders:



figures
finalScriptToRUN
results
scripts
README.txt

1. **figures:** all matlab figures of interest extracted during the execution of the scripts.

2. **finalScriptToRUN:** this folder contains a script 'performanceEvaluation_SVM_MLP.m' which should print you in the console accuracy results for both SVM and MLP. Finally, 2 figures will display, a confusion matrix for both SVM and MLP (should take no more than 1 minute to run the script). This folder also contains the test data 'allTest.csv'.

3. **results:** some results of some of the scripts previously run.

4. **scripts:** all other scripts are here. 'BayesOptimization_MLP.m' and 'gridSearch_SVM.m' are the hyperparameter tuning scripts for both SVM and MLP. 'PlotLearningCurves_SVM_MLP.m' this script draws a learning curve with the training/validation scores. 'PREPROCESSING_digitRecognition.m' is where we rearranged the dataset, so be studied. Finally, 'trainBestModels_SVM_MLP.m' does the training of the best models given the results of the hyperparameter tuning scripts.

All the scripts on this project are to be run with Matlab.