Application of Machine Learning and Pattern Recognition

Pedro Carvalho
Department of Electrical and Computer Engineering, Temple University
tun22685@temple.edu

Introduction: In the last years, the technology had a great advancement in pattern recognition systems. In nowadays almost every kind of information can be digitalized and stored in a virtual data format. The improvement in electronics components as microprocessors had allowed the management of large amounts of data with less effort of computational time, however with a higher energy consumption. Then, with the capability of processing huge data, the importance of the data analysis field raised immensely given the innumerous applications that can be performed from that, for example, sound and image processing applied to speech recognition and cancer diagnostics, respectively. In order to develop systems that could automatize data identification based only on its characteristics, machine learning came as a field of statistical and probabilistic solutions which algorithms can learn by themselves how to better classify a dataset, by analyzing its features components and returning some useful outputs based on these predictions.

This work aims to explore all the concepts learned throughout the semester related to machine learning and pattern recognition. Bayes Theorem is one of the most important concepts that is present in the majority of the machine learning algorithm solutions. This theorem describes a posterior probability of an event given its prior knowledge and an estimated likelihood for the current event. This concept is applied directly in classifications algorithms as Maximum Likelihood Classifier, in parameter estimators, and also in complex functions optimization (i.e. Bayesian Optimization). Moreover, the course provided the knowledge behind the mainly discriminative and generative algorithms present in the literature and currently widely used, based on supervised and unsupervised learning techniques. This basis allowed us to understand the full implementation of those algorithms, including the mathematical formulation, and how each parameter can influence in the data prediction. Therefore, this foundation grants a reasonable discernment for a decision of which parameters can be applied to an algorithm given a specific data.

It was provided a training and a development two-dimensional data set with the following distribution shown in figure 1. In like manner, it was provided a larger data set with a five-dimensional features space. Then, it was defined two discriminative algorithms for an appropriate classification of these data. Given the non-linearity of the two-dimensional data and inferring that the five-dimensional data follow a similar distribution, the non-neural algorithm chosen was the Support Vector Machine. Additionally, for the artificial neural network solution approach, it was chosen the Multilayer Perceptron in which allows a bunch of initialization learning parameters possible to tune, and consequently, provide an acceptable result.

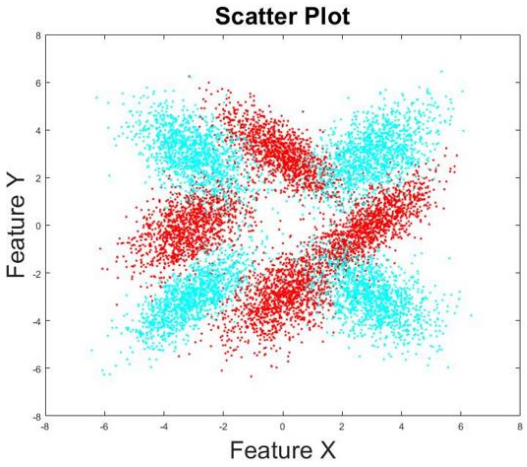

Figure 1. Scatter Plot 2D Data

Support Vector Machine (SVM - MATLAB): The SVM is a supervised learning that basically splits the data through a hyperplane which is defined by support vectors from the training set samples. This hyperplane separates the data and provides the prediction based on the region where the observation point is placed. This algorithm can be adapted for multi-classes classification which is not the focus on this work. The basis of this algorithm solution is to find specific key samples from the training set (support vectors) that can "support" a linear boundary in the data that maximize the distance between these support vectors and the hyperplane. This distance is known as margin. Note that, a very large margin can provide overfitting

which is not good for a classification model. A characteristic that makes the SVM a special classifier model is that, even though its basis is defined by a linear hyperplane boundary, we can still apply nonlinear data given the kernel trick. It essentially transforms the data by a dot product in a higher-dimensional space in which is possible to describe a linear hyperplane with an acceptable margin.

Multilayer Perceptron (MLP- Scikit Learn): An artificial neural network classifier takes the data features as an input layer and output a prediction for each sample. Between the input and output layers we have the hidden-layers. Which one of these hidden layers consists in a specific number of neurons or states that receive a weighted sum of all outputs from the previous layer and provides the output of this sum associated with a bias, given an activation function. The activation function is simply a function that each neuron uses to output the weighted input that it receives. Based on a loss function in which measures how well the network is predicting the data given a training set, the proper neural network can update its weights parameters by backpropagation in order to reduce the loss function value and reach a better prediction rate. Thus, the model can learn which are the specific input components (features) that provide better accuracy for some data, and then, define a larger weight for those components than for others. The learning rate is an important parameter that dictates how much the model should change at each iteration. The most common approach applied to neural network models' optimization is the Stochastic Gradient Descent (SGD) which computes the gradient of the loss function based on its weights and updates the model iteratively. A more recent optimization method is the ADAM algorithm which provides an adaptative learning rate based on a computation of the average of the gradient and the squared gradient.

Results: The first evaluations were done in the two-dimensional data for the following reasons; this data is smaller than the five-dimensional, so the computational time was proportionally lower. Also, this data set provides a visualization of the model classifier boundary by plotting its decision surface over the data scatter. This visual analysis gives us a better idea of how different choices of parameters affect the model. The first algorithm trained was the non-neural SVM. Figure 2 presents a scatter plot of SVM classification. As mentioned before, the model should perform a non-linear classification, and for that, it was applied the Kernel trick. Thus, an evaluation of which better Kernel function choice was done with a solution based on Bayesian Optimization and a 5-Fold cross validation. Bayesian Optimization is a good optimization approach in cases that the function is not analytically well-defined or not deterministic. This method takes an assumption of some first random outputs and defines a mean function for that. With that mean function and the prior hyper-parameters results information, the algorithm estimates where should be located the input vector of parameters that provides a minimum or maximum expected output function value, known as Gaussian Process. This estimation is repeated iteratively until it reaches the stop conditions set by the user. The Radial Basis Function (RBF) Kernel function provided the best classification performance among second, third, fourth order Polynomial and Sigmoid functions. Once defined the Kernel function, another optimization process was repeated in order to tune properly the remaining hyperparameters through more iterations of Bayesian Optimization. The performance results for this data was shown in table 1. The same hyperparameters optimization procedure was done for the five-dimensional data with just one more step of fine-tuning. After the convergence of 40 Bayesian Optimization iterations, also with RBF Kernel function, a Grid Search was performed around the optimal point. The final performance of the tuned model was

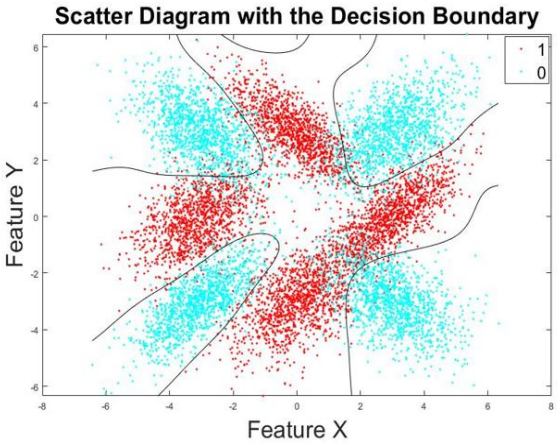

Figure 2. Decision Surface Based on SVM Classifier (RBF Kernel Function) in the Training Data.

shown in table 1. One curious observation is that, a high value of the parameter C provided a highly overfitted model with zero percent of train classification error, and a bad dev prediction. So, to avoid overfitting, this parameter was limited in a range with small values (1e-4, 1e-1) in the optimization process.

Algorithm	Data Dimension	Data Set		
		Train	Dev Test	Eval
SVM (MATLAB)	2D	7.76 %	7.90 %	7.80 %
	5D	36.61 %	36.47%	36.59%
MLP (ScikitLearn)	2D	8.00 %	7.95 %	7.75 %
	5D	36.52 %	36.50 %	36.97%

Table 1. Classification Performance

The multilayer perceptron algorithm was also tuned through a hybrid approach with Grid Search and Bayes Optimization. The hyperparameters evaluated in the optimization were: the number of layers, the number of neurons in each layer, the batch size, the L2 regularization factor (controls overfitting), the learning rate initialization value and the momentum. The activation function was fixed as ReLu. The number of epochs was fixed in 100 but it was also set the early stoping. As the number of layers and neurons are naturally positive integers, they were tuned first through three differents Grid Searches. The first one searched the optimal number of neurons for each one of the two hidden layers. The second did the same search now for three hidden layers, and the last one, for four hidden layers. The best case was four layers with 80 nodes each one. Then, the batch size was tuned manually with the remaining default parameters, presenting a good performance for a batchsize of 20 samples. Then, the remaining float parameters, the L2 regularization, learning rate initial value and the momentum, were tuned by Bayesian Optimization with 20 initial random parameters and 100 more iterations. These two first parameters were restricted in a range of (1e-5, 1e-2) and, (0.5 , 1) for the momentum. Outside of the optimization algorithm, it was also verified the best type of backpropagation optimizer, SGD or ADAM. The best performance was provided with SGD's backpropagation optimizer. This full optmization procedure was done for both data set types. The classification performance for Multilayer Perceptron Algorithm is shown in table 1.

For the five-dimensional data, the elapsed computational time for a Bayes Optimization with 40 iterations in the SVM training was 32 hours, and 65 hours for the 200 different combinations in the Grid Search fine-tuning. MLP presented a faster training process. The Grid Search for the number of layers decision took approximately 40 hours, after the layers' parameters defined, the Bayes Optimization for 20 random points plus 150 estimated iterations, spent 13 hours. All these optimization rounds were repeated several times during the past three weeks working on this project in order to reach the results shown in the table 1.

Conclusions: This work presented a brief background of the current field applied to Machine Learning focused on the concepts acquired throughout the course. It has shown that for a good tuning performance we should have a deep knowledge of the algorithm basis which will allow us to define the possible best choices of the model hyperparameters, and consequently, avoid wasting unnecessary time during the training. It is important to highlight that besides the final algorithms' implementations, it was also previously evaluated the performance of other classification algorithms, namely Random Forest, K-Nearest Neighbors, and Recurrent Neural Network (LSTM), with their default parameters, in order to decide for the most adequate algorithm choice. Moreover, the tuning process automation done by Grid Search, Random Search, and Bayes Optimization, requires a reasonable insight for choosing the hyperparameters' range in their initialization. The advantage of the automation in the tuning process is that it provides less manual job during the training. However, it brings more processing time. An important observation: this tuning process for both classifier algorithms reveled that a deep search for all possible hyperparameters are impracticable, due the computational time to fit the model for a large data set. Thus, we are not able to verify all possible combinations of hyperparameters as we are supposed to do. Then, we should tweak some of them separately and find some smart search solutions as Bayesian Optimization in order to avoid use only blind searches as trial and error method. Finally, we noticed that both algorithms performed similar results which states that the tuning process is what really defines the performance of the classifier model.