Supervised Learning

Mariajose Franco Orozco
mfrancoo@eafit.edu.co
EAFIT University
Mathematical Engineering
Medellín, Colombia

I. Introduction

Supervised Learning is a very important methodology in machine learning. It consists of training algorithms with labeled data. With this, the idea is to map a set of input variables and an output variable. And then, applying this mapping to predict the outputs of new unseen data [1].

Examples of supervised learning algorithms include linear regression, decision trees, random forests, support vector machines, and neural networks. Supervised learning is commonly used in a variety of applications, including image recognition, speech recognition, natural language processing, and predictive modeling.

In this project, the objective is to train 5 algorithms: linear regression, decision tree, SVM with a linear kernel, SVM with a polynomial kernel, and SVM with a radial base kernel. The idea is to train this model with the known data in order to reach some guarantees of learning.

II. Theoretical Framework

This section will explain the theory of the concepts used in this project.

A. *Learning Machines*

1) ***Linear Regression****:* The idea of linear regression is to find the relation or dependencies between variables, usually between one dependent variable and one or more independent variables in charge of 'explaining' the dependent variable. This is also called features or observed data [2].

2) ***Decision Tree****:* Is a classifier that partitions the space into two or more sub-spaces according to a certain discrete function of the input value nodes [2]. Each partition is called a node; the nodes are connected between branches. Each node represents a decision and each branch represents the possible consequence of that decision.

3) ***Support Vector Machine****:* It is a tool for both classification and regression analysis. The idea of this algorithm is to find a hyperplane that best separates the input data or features into different classes. The hyperplane selected will be the one that maximizes the distance between the hyperplane and the closest data point of each class, this is called the margin [3]. Different kernels can be used for the Support Vector machines, for example, linear kernels, polynomial kernels, radial base kernels, logistic kernels, and more.

B. *Generalization Error*

It refers to the performance of the model in unseen or new data. The idea is to make a model that with known data can predict, with a good performance, unseen data, which is the reason why the desired generalization error would be small. The generalization error is defined as follows:

$$err_p(h) = prob(h\Delta C_i) < \epsilon \tag{1}$$

If we define C_i as a target concept, then it will be paired with a subset of x and consequently will produce a subset $h \subset x$ called hypothesis with the distribution p.

C. *Training Error*

The idea of supervised learning is to build models that satisfy a low training error. This means that the classification obtained with the built models would likely agree with the real labels of the data points. the training error is the measure of the disagreement between the output of our models and the real labels, and it is defined as follows:

$$err_S(h) = \frac{|S \cap (h\Delta C_i)|}{|S|} < \delta \tag{2}$$

D. *VC dimension*

The Vapnik-Chervonenkis (VC) dimension is a concept in machine learning and statistics that measures the capacity or complexity of a model class. It represents the maximum number of data points that can be shattered or separated by a given model class. It is useful for guiding the selection of appropriate models and for evaluating their performance on new data.

E. *Optimal Training Set*

It is possible to guarantee learning in our machines with the following equations:

$$n \geq \frac{1}{\epsilon}\left(ln(|H| + ln\left(\frac{1}{\delta}\right)\right) \tag{3}$$

and for the decision tree, the equation is the following:

$$n \geq \frac{ln(2)}{2\epsilon^2}\left((2^k - 1)(1 + log_2(m)) + 1 + ln\left(\frac{1}{\delta}\right)\right) \tag{4}$$

where k is the depth and m is the number of features.
With these equations and defining the ϵ and δ desired, we can obtain the number of data we need in our sample in order to guarantee learning.

III. Methodology

First, a toy dataset was used for analyzing how the algorithms worked. The toy dataset considered was the Iris dataset.

Then, the real dataset used in this implementation was the *Rice Dataset Cammeo and Osmancik* which can be downloaded from **here**. It consists of a study made in 2019 by [**?**] in which 2 types of rice: Cammeo and Osmancik, were analyzed and some characteristics were extracted from each type of them resulting in 3810 data and 7 attributes for each one of them [4]. The attributes are:

- **Area:** Returns the number of pixels within the boundaries of the rice grain.
- **Perimeter:** Calculates the circumference by calculating the distance between pixels around the boundaries of the rice grain.
- **Major Axis Length:** The longest line that can be drawn on the rice grain, i.e. the main axis distance, gives.
- **Minor Axis Length:** The shortest line that can be drawn on the rice grain, i.e. the small axis distance, gives.
- **Eccentricity:** It measures how round the ellipse, which has the same moments as the rice grain, is.
- **Convex Area:** Returns the pixel count of the smallest convex shell of the region formed by the rice grain.
- **Extent:** Returns the ratio of the region formed by the rice grain to the bounding box pixels.

The output of the dataset is the class of rice, which is 'Cammeo' or 'Osmancik'. This was a categorical output, for making it a numerical output, 2 dummy variables were considered for the output:

- **Cammeo:** 1 if it is and 0 if it is not.
- **Osmancik:** 1 if it is and 0 if it is not.

This way, the resultant dataset consists of 7 inputs and 2 outputs, all of them numerical variables, ready to be normalized.

For this project, we will find the optimal training set for $\epsilon = \delta = 0.01, 0.05, 0.1$, and then, with the amount of data needed, we will train the 5 different learning machines mentioned above. Then, the score will be obtained when passing to the machines the test dataset in order to see if these machines were well trained.

Then, the idea was to extend and reduce the dimension of the problem and make the same process with these new datasets. For extending the dimension of the dataset, an autoencoder was used and the high-dimension dataset resultant is 12 dimensional. For the reduction of the space, the UMAP algorithm was implemented and the low-dimension dataset resultant is 3 dimensional. Finally, the 5 learning machines were also applied to these new datasets.

IV. Experimentation

The optimal n for the training set was obtained with the equations 3 and 4, and the results for the original dataset were the folowing:

TABLE I
Optimal Training Set for the Original Dataset

Optimal training set size considering differents generalization and training error			
	epsilon=delta=0.01	**epsilon=delta=0.05**	**epsilon=delta=0.1**
Linear Regression	668.46	101.50	43.82
Decision Tree	903188.69	35904.43	8952.08
SVM Linear Kernel	668.46	101.50	43.82
SVM Polynomial Kernel	680.23	103.85	44.99
SVM Radial Base Kernel	inf	inf	inf

Then, the same equations were applied to the high-dimension dataset obtained with the autoencoder. The results obtained for the optimal size for the training set are the following:

TABLE II
Optimal Training Set for the High Dimensional Dataset

Optimal training set size considering differents generalization and training error			
	epsilon=delta=0.01	**epsilon=delta=0.05**	**epsilon=delta=0.1**
Linear Regression	717.01	111.21	48.67
Decision Tree	903188.69	35904.43	8952.08
SVM Linear Kernel	717.01	111.21	48.67
SVM Polynomial Kernel	724.42	112.69	49.41
SVM Radial Base Kernel	inf	inf	inf

Finally, the optimal n for the low-dimension dataset are the following:

TABLE III
Optimal Training Set for the Low Dimensional Dataset

Optimal training set size considering differents generalization and training error			
	epsilon=delta=0.01	**epsilon=delta=0.05**	**epsilon=delta=0.1**
Linear Regression	599.14	87.64	36.88
Decision Tree	903188.69	35904.43	8952.08
SVM Linear Kernel	599.14	87.64	36.88
SVM Polynomial Kernel	621.46	92.10	39.12
SVM Radial Base Kernel	inf	inf	inf

V. Results

In this section, the results obtained with every model are presented. First, the learning was performed for the original dataset, and in order to get the scores obtained, the test dataset was used. In second place, the learning of the machines was performed with the high-dimensional dataset and finally the learning for the low-dimensional dataset.
In the following table, the results obtained for all the models with linear regression are presented.

TABLE IV
Results obtained for the linear regression

Linear Regression					
	Dataset	**epsilon**	**delta**	**n**	**score**
Model 1	Original	0.01	0.01	668.46	0.704
Model 2	Original	0.05	0.05	101.50	0.653
Model 3	Original	0.1	0.1	43.82	0.653
Model 4	High Dim	0.01	0.01	717.01	0.733
Model 5	High Dim	0.05	0.05	111.21	0.689
Model 6	High Dim	0.1	0.1	48.67	0.626
Model 7	Low Dim	0.01	0.01	599.14	0.705
Model 8	Low Dim	0.05	0.05	87.64	0.708
Model 9	Low Dim	0.1	0.1	36.88	0.675

We can see that the model with the best performance was model 4: with high-dimensional dataset, $\epsilon = \delta = 0.01$ and $n = 717$. The worst performance was obtained with the model 6: with high-dimensional dataset, $\epsilon = \delta = 0.1$ and $n = 48$.
The results obtained for all the models with the decision tree are presented in the following table.

TABLE V
Results obtained for the decision tree

Decision Tree					
	Dataset	**epsilon**	**delta**	**n**	**score**
Model 1	Original	0.01	0.01	903188.69	0.908
Model 2	Original	0.05	0.05	35904.43	0.905
Model 3	Original	0.1	0.1	8952.08	0.906
Model 4	High Dim	0.01	0.01	903188.69	0.916
Model 5	High Dim	0.05	0.05	35904.43	0.904
Model 6	High Dim	0.1	0.1	8952.08	0.905
Model 7	Low Dim	0.01	0.01	903188.69	0.913
Model 8	Low Dim	0.05	0.05	35904.43	0.916
Model 9	Low Dim	0.1	0.1	8952.08	0.910

It is possible to evidence that the models with the best performance were models 4: with high-dimensional dataset, $\epsilon = \delta = 0.01$ and $n = 903188$ and model 8: with low-dimensional dataset, $\epsilon = \delta = 0.05$ and $n = 35904$. The worst performance was obtained with the model 5: with high-dimensional dataset, $\epsilon = \delta = 0.05$ and $n = 35904$.
In the following table, the results obtained for all the models with the SVM and a linear kernel are presented.

TABLE VI
Results obtained for the SVM Linear Kernel

SVM Linear Kernel					
	Dataset	**epsilon**	**delta**	**n**	**score**
Model 1	Original	0.01	0.01	668.46	0.927
Model 2	Original	0.05	0.05	101.50	0.925
Model 3	Original	0.1	0.1	43.82	0.921
Model 4	High Dim	0.01	0.01	717.01	0.923
Model 5	High Dim	0.05	0.05	111.21	0.916
Model 6	High Dim	0.1	0.1	48.67	0.914
Model 7	Low Dim	0.01	0.01	599.14	0.925
Model 8	Low Dim	0.05	0.05	87.64	0.919
Model 9	Low Dim	0.1	0.1	36.88	0.910

We can evidence that the model with the best performance was model 1: with the original dataset, $\epsilon = \delta = 0.01$ and $n = 668$. The worst performance was obtained with the model 9: with low-dimensional dataset, $\epsilon = \delta = 0.1$ and $n = 36$.
In the following table, the results obtained for all the models with the SVM and a polynomial kernel are presented.

TABLE VII
Results obtained for the SVM Polynomial Kernel

SVM Polynomial Kernel					
	Dataset	**epsilon**	**delta**	**n**	**score**
Model 1	Original	0.01	0.01	680.23	0.927
Model 2	Original	0.05	0.05	103.85	0.913
Model 3	Original	0.1	0.1	44.99	0.879
Model 4	High Dim	0.01	0.01	724.42	0.931
Model 5	High Dim	0.05	0.05	112.69	0.923
Model 6	High Dim	0.1	0.1	49.41	0.876
Model 7	Low Dim	0.01	0.01	621.46	0.931
Model 8	Low Dim	0.05	0.05	92.10	0.921
Model 9	Low Dim	0.1	0.1	39.12	0.854

We can see that the models with the best performance were models 4: with high-dimensional dataset, $\epsilon = \delta = 0.01$ and $n = 724$ and model 7: with low-dimensional dataset, $\epsilon = \delta = 0.01$ and $n = 621$. The worst performance was obtained with the model 9: with low-dimensional dataset, $\epsilon = \delta = 0.1$ and $n = 39$.
Finally, in the following table, the results obtained for all the models with the SVM and a radial base kernel are presented.

TABLE VIII
Results obtained for the SVM Radial Base Kernel

SVM Radial Base Kernel					
	Dataset	**epsilon**	**delta**	**n**	**score**
Model 1	Original	0.01	0.01	inf	0.930
Model 2	Original	0.05	0.05	inf	0.930
Model 3	Original	0.1	0.1	inf	0.930
Model 4	High Dim	0.01	0.01	inf	0.926
Model 5	High Dim	0.05	0.05	inf	0.926
Model 6	High Dim	0.1	0.1	inf	0.926
Model 7	Low Dim	0.01	0.01	inf	0.926
Model 8	Low Dim	0.05	0.05	inf	0.926
Model 9	Low Dim	0.1	0.1	inf	0.926

Finally, we can evidence that the first 3 models obtained the same score, with this score being the best score within these models, and the worst score was obtained with the rest of the models.

VI. Conclusions

The best model for the linear regression is model $\#4$ with a score of 0.733. The best model for the decision tree is model $\#4$ instead of model $\#8$ because the ϵ and δ are lower in model $\#4$ with a score of 0.916. The best model for the SVM with a linear kernel is model $\#1$ with a score of 0.927. The best model for the SVM with a polynomial kernel can be any model between $\#4$ and $\#7$ because they have the same ϵ and δ and the same score of 0.931. The best model for the SVM with a radial base kernel is model $\#1$ because is the one with lower ϵ and δ with a score of 0.930. The best performance between all the machines was obtained with the support vector machine using a polynomial Kernel, but the performance, in general, was very good, all the machines were able to learn from the training dataset in order to get a great score with the test dataset.

It is also evident that the majority of the best performance was obtained with the high-dimension dataset. This could also mean that our data variables considered were not as representative of the data and it is needed to increment the space to obtain more features about them.

References

[1] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. [Online]. Available: https://doi.org/10.1007/978-3-540-75171-7_2

[2] V. Nasteski, "An overview of the supervised machine learning methods," *HORIZONS.B*, 2017.

[3] A. Kumari, J. Prem Kumar, V. Prakash, and K. Divya, "Supervised Learning Algorithms: A Comparison," *Kristu Jayanti Journal of Computational Sciences*, 2021.

[4] U. M. L. Repository, "Rice (Cammeo and Osmancik) Data Set," 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Rice+{%}28Cammeo+and+Osmancik{%}29{#}