Department of Computer Science
Technical University of Cluj-Napoca

Intelligent Systems

Laboratory activity 2019-2020

Name: Nitu Cristian-Mihai
Group: 30235
Email: cristian.nitu85@gmail.com

Profesor indrumator: Alexandru Ghiurutan

Contents

Chapter 1

Project description

This project scope is to implement some Machine Learning algorithms on different datasets, and to analyze how these algorithms performs on them. We will start by the study of a simple OR perceptron, will create a neuronal network with different numbers of hidden layers, and different numbers of nodes per layer, will see how CNN, decision trees, K-Means, KNN, RNN or SVM performs. Upon doing these, we can form an idea based on what does the term of "Machine Learning" means.

Chapter 2

A short introduction to ML

The term of "Machine Learning" can be defined as the process of teaching a computer to perform a task without needing to provide explicit instructions. How can one do this? Well, it's simple: we have to select a model (algotithm) which our machine can apply on a give dataset for trying to "learn" what is happening there (not enough learning: "underfitting", too much learning: "overfitting") and try to apply these knowledge on other data. We could reduce this process in 4 simply steps: preprocess model data, construct the model, train the model on the dataset ("learn") and evaluate how useful that model is.

We can divide these algorithms into categories according to their purpose:

- **Supervised learning**: the algorithms here try to determine the relationship between the inputs in the dataset and the predicted output, so that, it should be able to make new predictions based on other "unseen" inputs. (ex: predict whether a person has cancer based on different attributes).
- **Unsupervised learning**: there is no output to associate with the inputs from the dataset, so that the computer has to define "its own rules", to determine what the output should be. (ex: pattern detection).
- **Semi-supervised learning**: it stands between the 2 aforementioned categories. In this case, the output is being given just for a small part of the present inputs from the

dataset, for the others, it is the computer's responsability to find out the answer. (ex: speech recognition from an audio file).

- **Reinforcement learning**: it is based on the interaction with the environment. In the process, it learns from its experiences of the environment until it explores full range of possible states to reach its goal. (ex: escape from a labyrinth: at each step, the computer will receive a feedback based on its current position (a reward), so that it can deduce whether it is more closer to its goal (getting out of the labyrinth)).

Chapter 3

Perceptron: the first step in ML

What is a perceptron? Well, a perceptron is a kind of structure which has a number of inputs (depending on its size), and produce a binary output (its activation function). Each of these inputs has a weight (you can associate this with the importance of that perceptron). For finding its activation function output, we need to sum all the products between the inputs and their weights, and to compare it with the bias. If the sum is bigger than this value, the perceptron will be active.

We will find out how to implement an OR gate with 3 inputs using a perceptron, but, before that, we need 2 define some new concepts. The epoch determines the number of times we iterate through the entire training set. Threshold represents the max number of epochs used for training our model (or perceptron in our case). Learning rate represents how much we the weights will be updated, so that the machine will determine the actual output based on some inputs. For implementing our OR gate we need to make 3 steps: initialization where we set the threshold, the learning rate and the weights; training: where we update the weights based on the neuron predictions and prediction: where we find out whether the perceptron is active or not (the result of the OR gate).

Chapter 4

Neural Network

In this chapther we will describe a neural network, together with an example of application. A neural network is formed by more layers: an input layer, 0, 1 or more hidden layers and one output layer. Each layer is made of one or more neurons (a structure which can give a value between 0 and 1). The number of hidden layers and the one of neurons per layer influences the knowledge of the input dataset (we have to take care of other aspects too, like learning rate, number of epochs, dataset size and number of parameters for obtaining the best predictions).

In our example, we will create a neural network for predicting whether a person is diabetic or not (binary classification problem). We will use the PimaDiabetesDataset for this purpose. This dataset has 8 attributes which influences the existence of the diabet (like glucose), and a binary output representing the presence of this disease. We will split the dataset into a training set (the machine will learn from it) and a validation set (where we test how accurate the predictions are). We will scale our inputs (between -1 and 1), for better results. We will test with and without mini-batch (bunches of data), will test with different number of hidden layers and different number of neurons on each layer, applying different functions on each layer (like ReLU, Sigmoid). We will plot, comparatively, the loss functions of training and validation set, and their accuracy. This way we can have an overview on how to improve our model predictions.

Chapter 5

Convolutional Neural Network

A convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNN are versions of multilayer perceptrons (which means that each neuron in one layer is connected to all neurons in the next layer). The pass from a layer to another is made by a kernel filter which maps a region of its size from the input layer to a value on the output layer. The number of input, kernel and output layers may vary.

We will test how a three layers CNN works on the CIFAR10 dataset, we will apply ReLU function, we will train and test the dataset, and plot the loss and accuracy for both parts of the dataset (red for train, blue for accuracy).

Chapter 6

Decision trees

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. It makes a decision, at each step, based on an heuristic (information gain, gini ratio or gini index). The main advantage is that we can see what is happening at every step, being able to modify how depth it should go for increasing our accuracy.

We will apply this decision tree for finding out whether a person has breast cancer or not. For that, we will choose a bunch of representative columns like Age, BMI, Glucose, Insulin, HOMA, Leptin and MCP.1, called featured columns (one of them represent an internal node at every step). Will split our dataset into training and validation sets (we will choose a higher training set) and will apply a decision tree classifier. Although we obtain a good accuracy (0.72), we will try to improve that one by applying the gini index and reducing the max depth size, (accuracy = 0.77).

Chapter 7

K-Nearest Neighbor (KNN)

The KNN is an algorithm used for classification which main advantage is the less time consuming training phase (no need to train for generalization). It works by associating a new element to the group which has a number of K members closer to the target element (based on euclidian distance). Because of this, it does not need to "learn" based on the existing data.

We will classify some seeds (which are split in 3 categories), based on some inputs like Area, Perimeter, Compactness, Length of Kernel, Width of Kernel, Assimetry Coefficient, Length of Kernel Groove. We will test this classifier with different values of k parameter and will study the given accuracy, which is a very good one (over 90 percents in some cases).

Chapter 8

Support Vector Machines (SVM)

SVM is an algorithm which separates data points using a hyperplane with the largest amount of margin. We can think at a hyperplane like a rectangle with a width equal to the margin component, which realize a perfect separation between the elements of 2 classes. This separation can be made, if needed, in a space with more dimensions (so that there is a delimitation between the elements).

We will use the breast cancer detection dataset for this algorithm; it will be split into a training set of 0.7 and a test set of 0.3. We will use a so-called SVM kernel which transforms an input data space into the required form (a higher dimensional space if needed). We will make the prediction using a linear kernel, which brings us a good accuracy of 0.8 .

Chapter 9

K-Means Clustering

Is an unsupervised learning algorithm. It takes K centroids, and split the data between those (by measuring a manhattan distance, as a basic example). This algorithm works based on some metrics

We will study the Titanic problem of survival. First of all, we will process the missing values, as this algorithm does not accept those ones (by replacing those ones with the mean on that column, for example). Other necessary data processings are: the removal of some features that are not significant, the removal of the output column (remember: unsupervised learning algorithm), and the conversion of all the fields to numeric ones. We will set the number of clusters to 2 as we get 2 possibile outputs (a person has survived or not). In order to enhance the performance, we tweak some parameters of the model itself, like the algorithm and the max number of iterations. By applying the algorithm, we will notice an accuracy of 0.62

Chapter 10

Bibliography

- https://medium.com/@thomascountz/perceptron-implementing-and-part-2-84bfb1f46597
- https://victorzhou.com/blog/intro-to-neural-networks/
- https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- https://towardsdatascience.com/conv2d-to-finally-understand-what-happens-in-the-forward-pass-1bbaafb0b148
- https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
- https://www.datacamp.com/community/tutorials/k-means-clustering-python
- https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn