Machine Learning Formulas*

García Prado, Sergio
sergio@garciparedes.me

Taboada Rodero, Ismael José
ismaeljose.taboada@alumnos.uva.es

May 28, 2017

Abstract

[TODO]

1 Introduction

[TODO]

2 Supervised Learning

[TODO]

Overfitting [TODO]

Error Measures [TODO]

- **Error Rate**: [TODO]
- **Resubstitution error**: [TODO]
- **Generalization Error**: [TODO]

Experimental Strategies [TODO]

- **Holdout**: [TODO]
- **Repeated Holdout**: [TODO]
- **Cross Validation**: [TODO]
- **Repeated Cross Validation**: [TODO]

2.1 Decision Trees

Decision trees are classifiers that works splitting the instances between the branches of a tree according to its attributes and assigning the feature's attribute in the leafs of the tree. An scheme of this structure is been included in the figure 1. Decision trees can represent logic functions. It tend to ovefit the data so the advanced heuristics try to avoid this problem.

*This document is being maintained in https://github.com/garciparedes/machine-learning-formulas

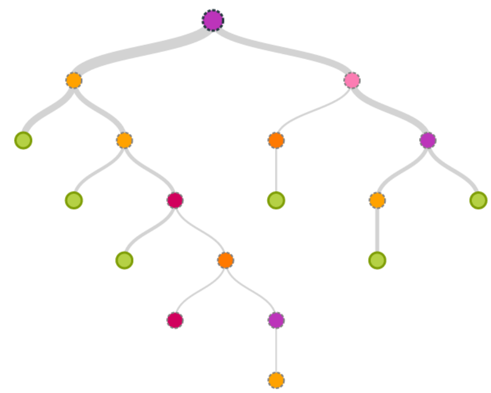

Figure 1: *General concept of Decision Tree. Figure from [ML13]*

2.1.1 Information Theory

Most popular decision trees use information theory heuristics. It's heuristics is based on the information generated by an attribute respect to the feature's attribute. In the equation (3) D is refered to the feaure's attribute and A to the current attribute to be examined. Note that the cardinalities ($card()$) and proportions ($p()$) is been updated according to dataset instances to be examined.

$$I(D) = -\sum_{i \in card(D)} p(d_i) * log_2(p(d_i)) \quad (1)$$

$$Rest(A) = \sum_{j \in card(D|A)} \frac{card(D|A_j)}{card(D)} * I(D|A_j) \quad (2)$$

$$Gain(D, A) = I(D) - Rest(A) \quad (3)$$

There is another heuristic who also uses the cardinality to give advantage to attributes with few categories. This heuristic callec *Information Gain Ratio* is shown at the ecuation (5).

$$GainDivisor(D, A) = -\sum_{j \in card(D|A)} \frac{card(D|A_j)}{card(D)} * log_2\left(\frac{card(D|A_j)}{card(D)}\right) \quad (4)$$

$$GainRatio(D, A) = \frac{Gain(D, A)}{GainDivisor(D, A)} \quad (5)$$

The technique implemented by *C4.5* to deal with numeric attributes is the binary discretization by intervals distributing the values in two categories. The partition is selected picking the point where the attribute $Gain()$ is maximized. One of the points where the feature changes it value.

2.1.2 ID3

ID3 algorithm builds a tree selecting the attribute who maximizes $Gain()$ function respecto de the feature's attribute. It only works with nominal data and removes the attributes that have already been used. When there is no more attributes to select, it label the branch with the feature of the remainder instances.

2.1.3 C4.5

C4.5 is an extension of ID3 tree. It includes some improvements like branch pruning, working with numerical attributes and unknown values. The pruning works in post-pruning mode, i.e. the tree is generated and after is pruned. The heuristic use to determine if branch is pruned is the improve of pesimistic error rate. There are two pruning operations:

- **Subtree Replacement**: Consist of remove the selected subtree and assign the most frecuent feature to this leaf.
- **Subtree raising**: Consis on replace a tree by one of it's childrens and redistribute the instances. The operation is expensive.

Unknown values are trated spliting the selection between branches and assigning probabilities to each of them. The feature where the instance is classified is the one who maximized the probability. During the train period attributes with unknown values are penalized.

2.2 Rule Based Systems

[TODO]

2.2.1 1R

[TODO]

2.2.2 PRISM

[TODO]

2.2.3 IREP

[TODO]

2.2.4 RIPPER

[TODO]

2.2.5 PART

[TODO]

2.3 Instance Based Learning

[TODO]

2.3.1 K - Nearest Neighbors

[TODO]

2.3.2 IB3

[TODO]

2.4 Bayes Learning

[TODO]

2.4.1 Naive Bayes

[TODO]

2.4.2 K2

[TODO]

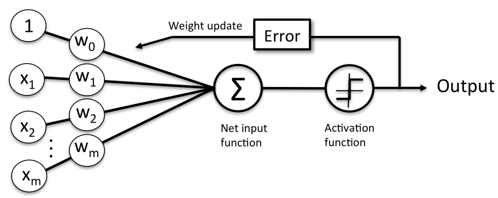

Figure 2: *General concept of perceptron*

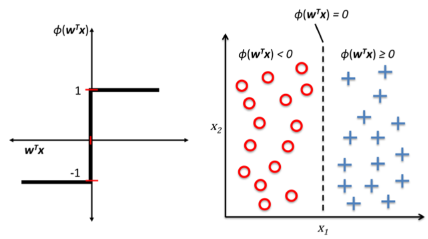

Figure 3: *Activation function*

2.4.3 TAN

[TODO]

2.5 Linear Classifiers

[TODO]

2.5.1 Linear Regression

[TODO]

2.5.2 Logistic Regression

[TODO]

2.5.3 Support Vector Machines

[TODO]

2.6 Neural Networks

2.6.1 Perceptron model

Perceptron's structures

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

Ouput equation

$$z = w_1 x_1 + \dots + w_m x_m = \boldsymbol{w^T x}$$

Activation function

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

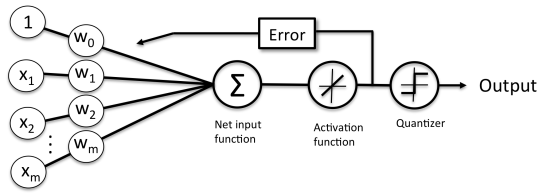

Figure 4: *General concept of adaline perceptron*

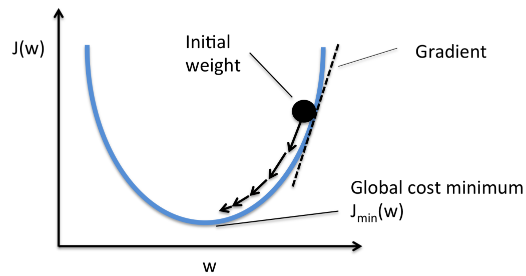

Figure 5: *Gradient descent*

Update of weight vector

$$w_j := w_j + \Delta w_j$$
$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

2.6.2 Adaptive linear neurons (ADALINE)

Cost function

$$J(w) = \frac{1}{2}\sum_i (y^{(i)} - \phi(z^{(i)}))^2$$

Update of weight vector

$$w_j := w_j + \Delta w_j$$
$$\Delta w_j = -\eta \nabla J(w) = \eta \sum_i (y^{(i)} - \phi(z^{(i)}))x_j^{(i)}$$

Features standarization

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

2.6.3 Single Layer Neural Networks

Simple Perceptron [TODO]

ADALINE [TODO]

2.6.4 Multi Layer Perceptron

[TODO]

2.6.5 Radial Basis Functions

[TODO]

2.6.6 Convolutional Neural Networks

[TODO]

2.6.7 Recurrent Neural Networks

[TODO]

3 Unsupervised Learning

[TODO]

References

[CCAG17] Teodoro Calonge Cano and Carlos Javier Alonso GonzáLez. Técnicas de Aprendizaje Autómatico, 2016/17.

[ML13] Big ML. A New Way to Visualize Decision Trees. https://blog.bigml.com/2013/04/19/a-new-way-to-visualize-decision-trees/, 2013.

[Ols16] Dr. Randal S. Olson. Python Machine Learning, 2016.