Name of the thesis

Konstantinos Papadimos

Contents

1 Theoritical Overview

1.1 Special Relativity (invariant mass)

1.2 The standard model

1.3 Collisions and resonances

1.4 Maybe some elements of the theory behind our monte carlo simulations

2 Detector physics, the CMS detecor

3 Machine Learning for Classification

The main goal Machine Learning is set to achieve, is the development of algorythms equiped with the capabillity of learning from data automatically. In particular, an artificially intelligent system must have the ability to identify objects in its surroundings, as well as anticipate the actions of its environment, in order to make informed decisions. Due to this, machine learning techniques tend to be more oriented towards forecasting, rather than prediction.

3.1 Decision trees and Supervised Learning

The distinction between different particles, can be regarded as a classification problem where the target, is the prediction of a categorical output variable(i.e. lepton, boson), based on one or more input variables(i.e momenta components). Classification problems in machine learning can be solved with supervised learning. In such procedure, a training data set is being used for the development(training) of a model that is able to perform the classification task. The output of the model is then beeing tested and evaluated on previously unseen data.

Before presenting any specific method of solving classification problems It is important to present an overview of the key elements in supervised learning.

3.1.1 Supervised learning

Let us pose the following problem: Given a data set $D = (\vec{X}, \vec{y})$, where $\vec{X}$ is a matrix of the independet variables and $\vec{y}$ is a vector of dependent variables, we want to find a model $f(\vec{x}; \vec{\theta})$, that can predict an output from a set of input variables. Moreover, we want to be able to judge the performance of the model on a given data set. To do that we need to define a cost function $C(\vec{y}, f(\vec{X}; \vec{\theta}))$, such that the model will have to find the parameters θ that minimize the cost function.(1803.08823)

This the mathimatical pustulation of a supervised learning problem. I will now, in brief, discuss the role and interpretation of each of the 'ingredients' stated above.

- Model

  The model, is a mathematical function $f : \vec{x} \to y$ of the parameters θ. Given a set of parameters, the output of the function, the prediction y_i, is derived from the input variables $\vec{x}$. The parameters are undefined. The task of the training is to estimate the set of parameters from the training data set. In a classification problem(something is of type a or it is not), it is possible to use the logistic transformation of the function output, to obtain the probability of the positive class.

- Cost function

  The cost function, also known as an objective function, is represented by mathematical function and it measures how well a model fits the training data. The cost function is used to train the model by finding the best set of parameters θ that minimize the function. In machine learning, the objective function, usually consists of two parts: a training loss function (L) and a regularization term (Ω).

  $$obj(\theta) = L(\theta) + \Omega(\theta)$$

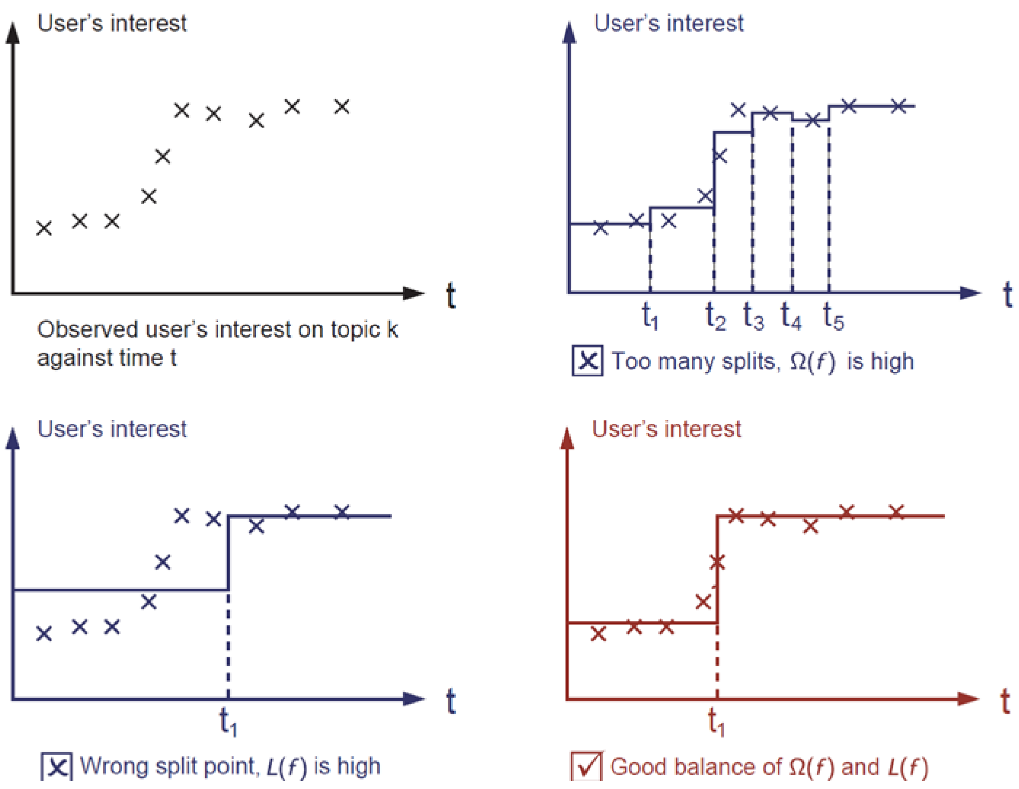
  The training loss function measures how predictive the model is with respect to the training data. A common choice of training loss function is the logistic loss, which is used for logistic regression(classification) and is given by

  $$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i \ln(1 + e^{\hat{y}_i}))]$$

  where y_i is the true label and $\hat{y}_i$ is the predicted label.

  The regularization term, $\Omega(\theta)$, controls the complexity of the model, which helps to avoid overfitting. Overfitting occurs when a model is too complex and starts to extract local features from the training data. The model thus, looses its generalization power to new unseen data. Regularization helps to prevent overfitting by adding a penalty term to the cost function, which discourages the model from having too many parameters or too complex a structure.

  The following figure gives an example of overfitting due to a very complex and very simple model.



3.1.2 Decision Trees

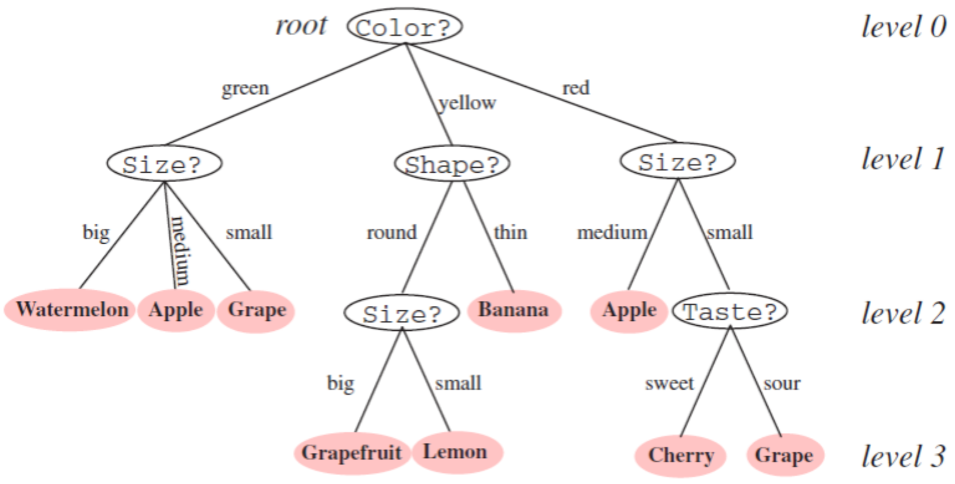
A decision tree is a flowchart-like tree structure, where each internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

Formally, a decision tree can be represented as a set of rules or conditions in the form of:

`f(X) = {condition1, condition2,..condition_n}`

where each condition is a tuple of the form (feature, threshold, comparison operator) and the final outcome is represented by the leaf node.

For example, consider the decision tree of figure x that classify fruits based on color, shape, size, and taste. Let X be the input `X = {"red", "smal", "sour"}`. Then `f(X) = "grape"`



- Decision Tree Ensembles

  The tree ensemble model consists of a set of classification and regression trees (CART). Let $\mathcal{F}$ be the set of all possible CART's and $f_k \in \mathcal{F}$, a function that represents a CART. The model in discussion then, can be written as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathcal{F}$$

If $\hat{y}_i$ represents the prediction of the tree, given an input variable x_i, the real label of x_i will be denoted as y_i . The objective function will be of the form:

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{i=1}^{t} \omega(f_i)$$

where $\omega(f_i)$ is the complexity of a given tree and l is the loss function.

- Tree boosting

  As stated earlier, the model is beeing trained, to learn those trees f_k that minimize the objective. The resulting model then, will be an ensemble of those functions f_k. The optimization of the objective, is a problem that cannot be solved with the traditional methods. Instead, the model is being iteratively trained in an additive manner.(1603.02754) let the prediction value at the t-th iteration be $\hat{y}_i^{(t)}$. In the next iteration(t+1), the chosen function f_{t+1}, is such that if added to the model, the resulting prediction $\hat{y}_i^{(t+1)}$ will minimize the cost function:

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = \hat{y}_i^{(1)} + f_2(x_i)$$

$$\dots$$

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) = \sum_{k=1}^{K} f_k(x_i)$$

  The objective at step t is:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t)}) + \sum_{i=1}^{t} \omega(f_i) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_i(t))$$

  Taylor expanding the loss function $l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$, around f_t, up to the second order and neglecting terms, referring to previous rounds, the specific objective becomes:

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t)$$

  Where

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$
$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

  This is the minimization goal for f_t . (xgboost-readthedocs-io-en-stable)