Introduction to Multivariate Methods

Classification and Function Approximation

Harrison B. Prosper

Florida State University

Bari Lectures

30, 31 May, 1 June 2016

Outline

- Lecture 1
  - Introduction
  - Classification
  - Grid Searches
  - Decision Trees
- Lecture 2
  - Boosted Decision Trees
- Lecture 3
  - Neural Networks
  - Bayesian Neural Networks

Introduction

Examples

In these lectures, I shall illustrate a few key ideas and methods in multivariate analysis using the following examples:

- Signal/Background Discrimination
- Wine Tasting
- Approximating a 19-parameter Function

Introduction

Two general approaches:

Machine Learning

Given training data $\boldsymbol{T} = (y, \boldsymbol{x}) = (y, x)_1, \ldots (y, x)_N$, a class of functions $\{ f \}$, and some constraint on these functions, teach a machine to learn the mapping

$$y = f(x)$$

Bayesian Learning

This is similar, except the goal is to create a *probability density* over the space of functions $f(x)$, that is, to assign a probability (density) to each function in the space.

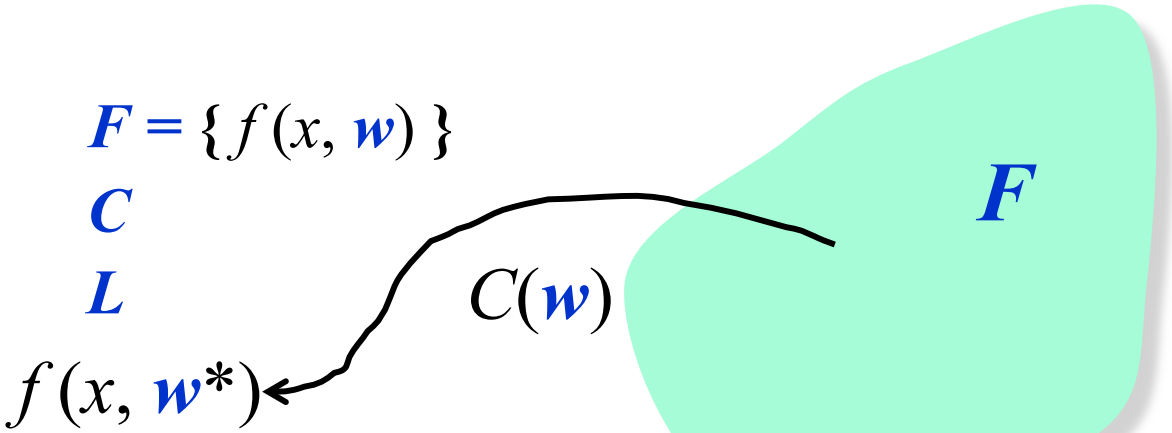
Machine Learning

Choose

Function space $F = \{ f(x, w) \}$

Constraint C

Loss function* L



Method

Find $f(x)$ by minimizing the empirical risk $R(w)$

$$R[f_w] = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w))$$

subject to the constraint $C(w)$

*The loss function measures the cost of making a bad choice of function from the function space.

Machine Learning

Many methods (e.g., neural networks, boosted decision trees, rule-based systems, random forests,…) use the

quadratic loss

$$L(y, f(x, w)) = [y - f(x, w)]^2$$

and choose $f(x, w^*)$ by minimizing the

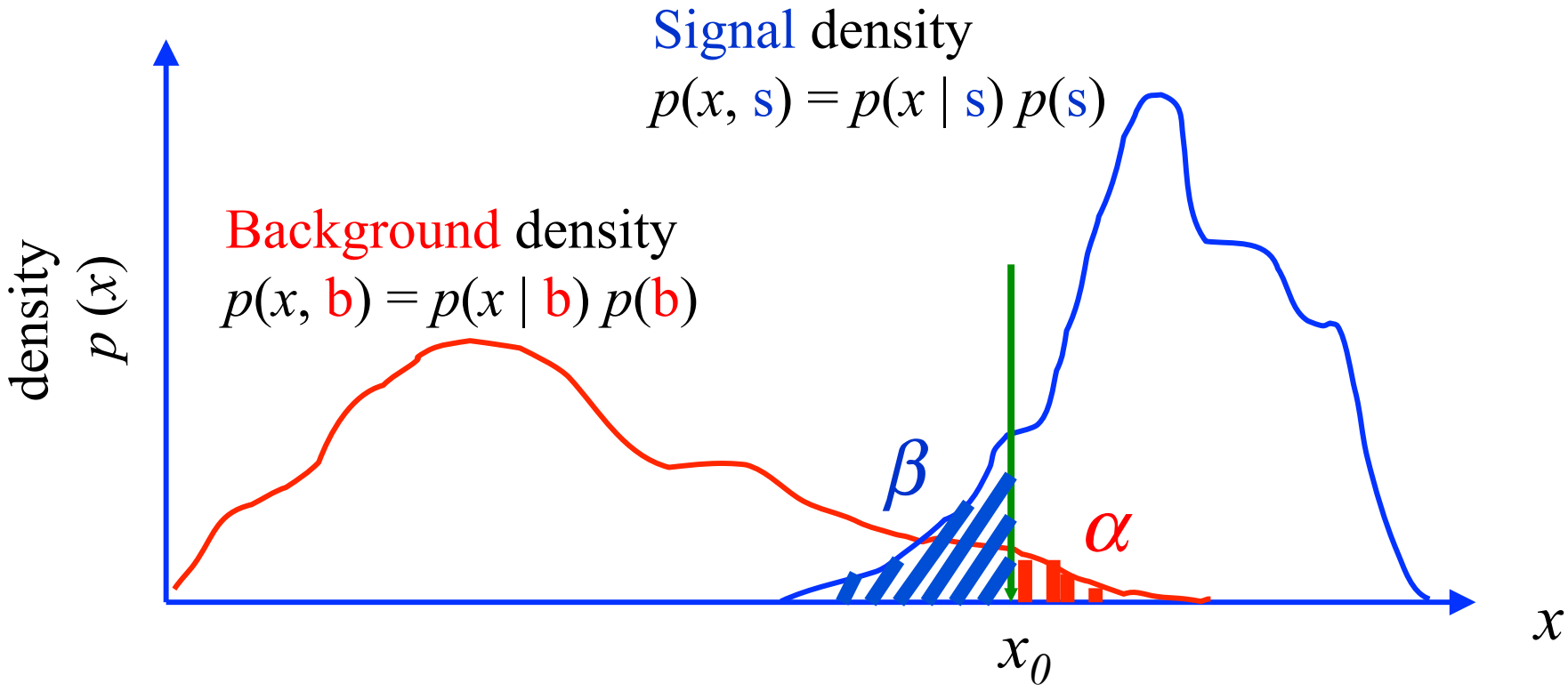
constrained empirical risk (that is, the average loss function)

$$R[f_w] = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w)) + C(w)$$

We shall consider Bayesian learning in Lecture 3.

Classification
In Theory

Classification: Theory



Optimality criterion: minimize the error rate, $\alpha + \beta$

Classification: Theory

The total loss L arising from classification errors is given by

$$L = L_b \int H(f)\, p(x,b)\, dx + L_s \int [1 - H(f)]\, p(x,s)\, dx$$

Cost of background misclassification

Cost of signal misclassification

where $f(x) = 0$ defines a decision boundary
such that $f(x) > 0$ defines the signal acceptance region

$H(f)$ is the Heaviside step function:

$H(f) = 1$ if f > 0, 0 otherwise

Classification: Theory

1-D example

$$L = L_b \int H(x - x_0)\, p(x,b)\, dx + L_s \int [1 - H(x - x_0)]\, p(x,s)\, dx$$

Minimizing the total loss L with respect to the boundary x_0

leads to the result:

$$\frac{L_b}{L_s} = \frac{p(x_0,s)}{p(x_0,b)} = \left[\frac{p(x_0 \mid s)}{p(x_0 \mid b)}\right] \frac{p(s)}{p(b)}$$

The quantity in brackets is just the likelihood ratio. The result, in the context of hypothesis testing (with $p(s) = p(b)$), is called the Neyman-Pearson lemma (1933).

Classification: Theory

The ratio

$$\frac{p(x,s)}{p(x,b)} = \frac{p(s\,|\,x)}{p(b\,|\,x)} \equiv B(x), \quad p(s\,|\,x) = p(x,s) \,/\, p(x)$$

$$p(b\,|\,x) = p(x,b) \,/\, p(x)$$

is called the Bayes discriminant because of its close connection to Bayes' theorem:

$$\frac{B(x)}{1+B(x)} = p(s\,|\,x) = \frac{p(x\,|\,s)p(s)}{p(x\,|\,s)p(s) + p(x\,|\,b)p(b)}$$

Classification: Theory

If the signal class *s* is assigned $y = 1$, while the class *b* is assigned $y = 0$, one obtains the very important result:

$$f(x) = \int y\, p(y \mid x)\, dy = p(1 \mid x) \equiv p(s \mid x)$$

See, Ruck et al., *IEEE Trans. Neural Networks* 4, 296-298 (1990);
Wan, *IEEE Trans. Neural Networks* 4, 303-305 (1990);
Richard and Lippmann, *Neural Computation.* 3, 461-483 (1991)

In summary:

1. Given sufficient training data *T* and
2. a sufficiently flexible function $f(x, w)$, then $f(x, w)$ will approximate $p(s \mid x)$, if $y = 1$ is assigned to objects of class *s* and $y = 0$ is assigned to objects of class *b*

Classification
In Practice

Classification: In Practice

Here is a *short* list of multivariate (MVA) methods that can be used for classification:

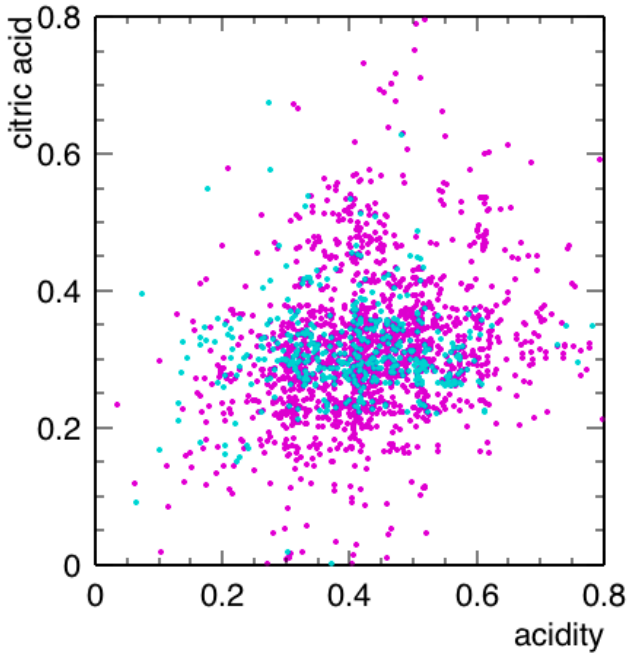
- Random Grid Search
- Fisher Discriminant
- Quadratic Discriminant
- Naïve Bayes (Likelihood Discriminant)
- Kernel Density Estimation
- Support Vector Machines
- Binary Decision Trees
- Neural Networks
- Bayesian Neural Networks
- RuleFit
- Random Forests

Illustrative Examples

Wine Tasting

Wine tasting is big business. But, can we automate it?

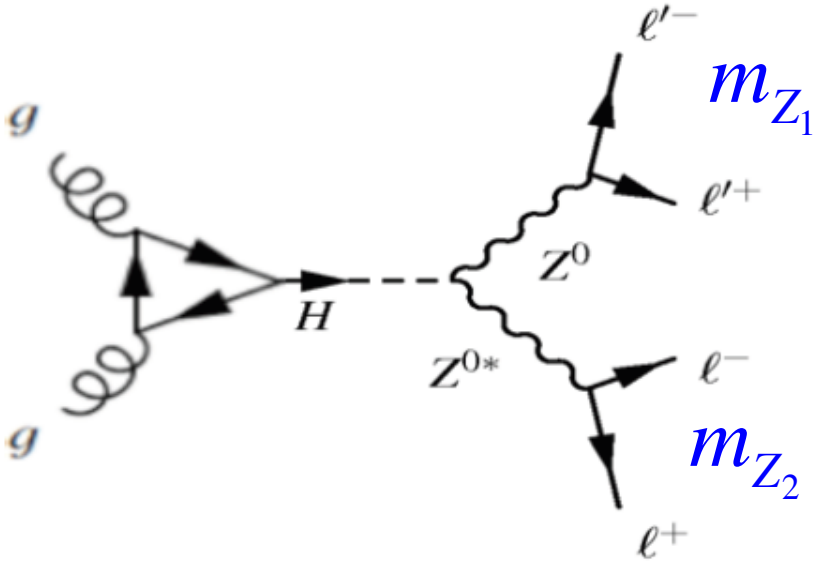
In principle, yes, if we can establish the physical attributes that define "good" wine, such as this one for $117,000 a bottle!



We'll look at this problem tomorrow.

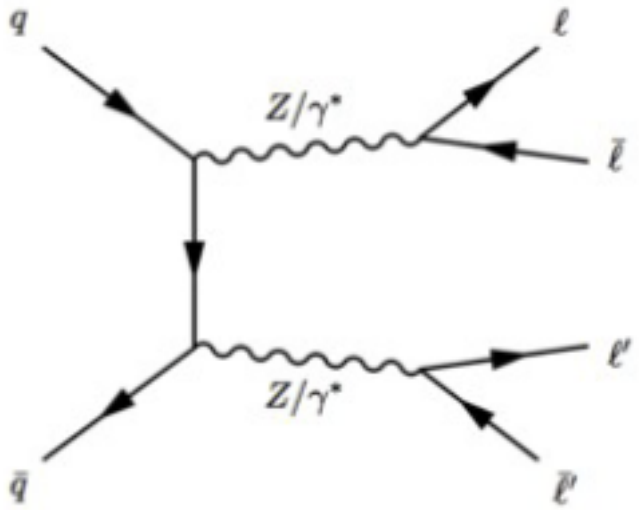
Higgs Boson to ZZ to 4 Leptons

Signal



$pp \to H \to ZZ \to \ell^+\ell^-\ell'^+\ell'^-$

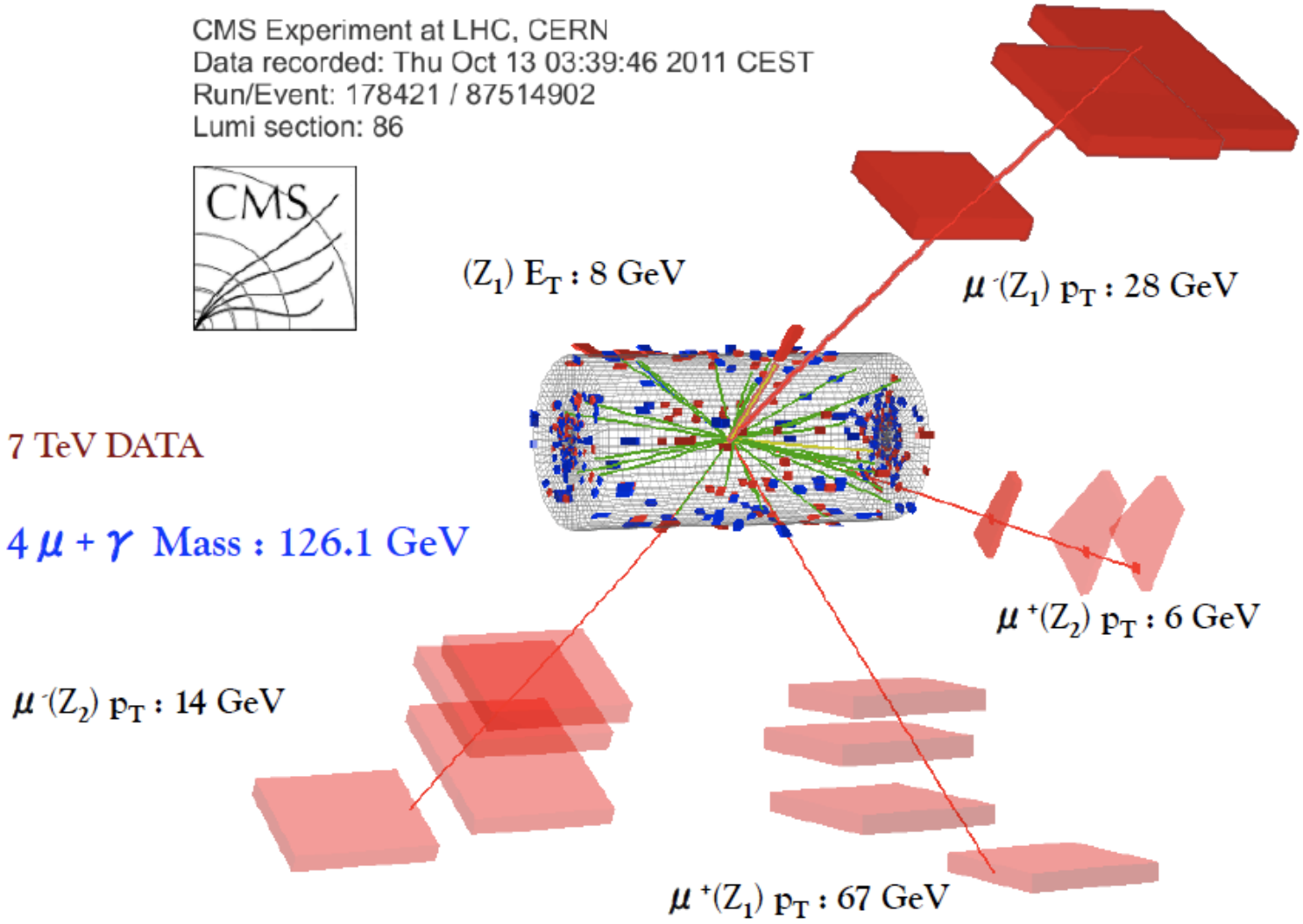
Background



$pp \to ZZ \to \ell^+\ell^-\ell'^+\ell'^-$

We shall use this example to illustrate signal/background discrimination, using the variables $x = (m_{Z1}, m_{Z2})$.

A 4-Lepton Event from CMS



Random Grid Search

Uniform Grid Search

Given the variables $x = (m_{Z1}, m_{Z2})$, the simplest way to try to separate the signal from the background (i.e., the noise) is to consider n thresholds (cuts) on each variable. This means, we would try n^2 pairs of cuts and find the best pair.

But, suppose we have d variables, and do the same thing. Now, we must consider n^d d-tuples of cuts! As d increases, this becomes computationally impossible becomes the number of points to be considered grows extremely rapidly. This is an example of the well-known "curse of dimensionality".

We need to be a bit cleverer…

Random Grid Search

One way to lessen this "curse" is to place cuts where they will do the most good.

The best place is at the *signal* points, since it is the signal that we are most interested in extracting!

We shall call

$$(x_1 \text{ CUT-DIR } m_{Z1}) \text{ AND } (x_1 \text{ CUT-DIR } m_{Z1})$$

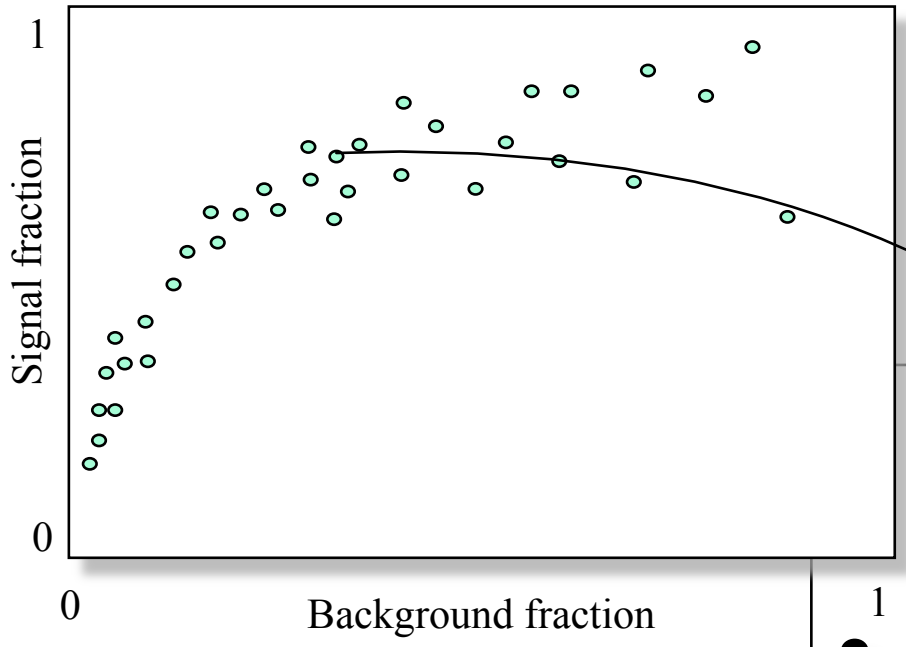
where CUT-DIR: <, >, or ==, a cut-point. In our example, our cut-point is a 2-tuple; in d-dimensions, it is a d-tuple.

(Note: we can also combine cut-points, to form "box" cuts.)

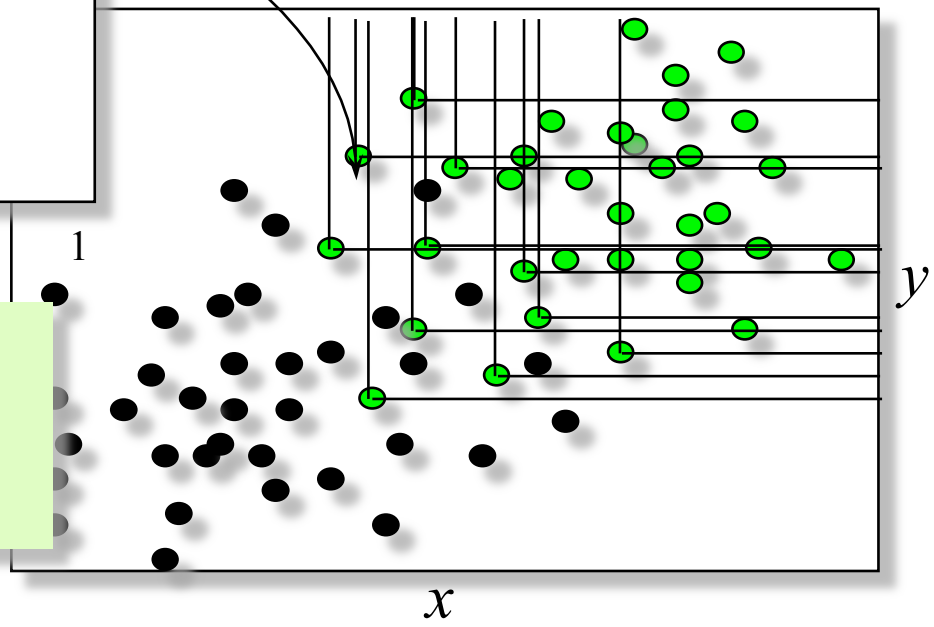
The next slide is a graphical representation of the algorithm.

Random Grid Search (RGS)

Take each point of the signal class as a **cut-point**



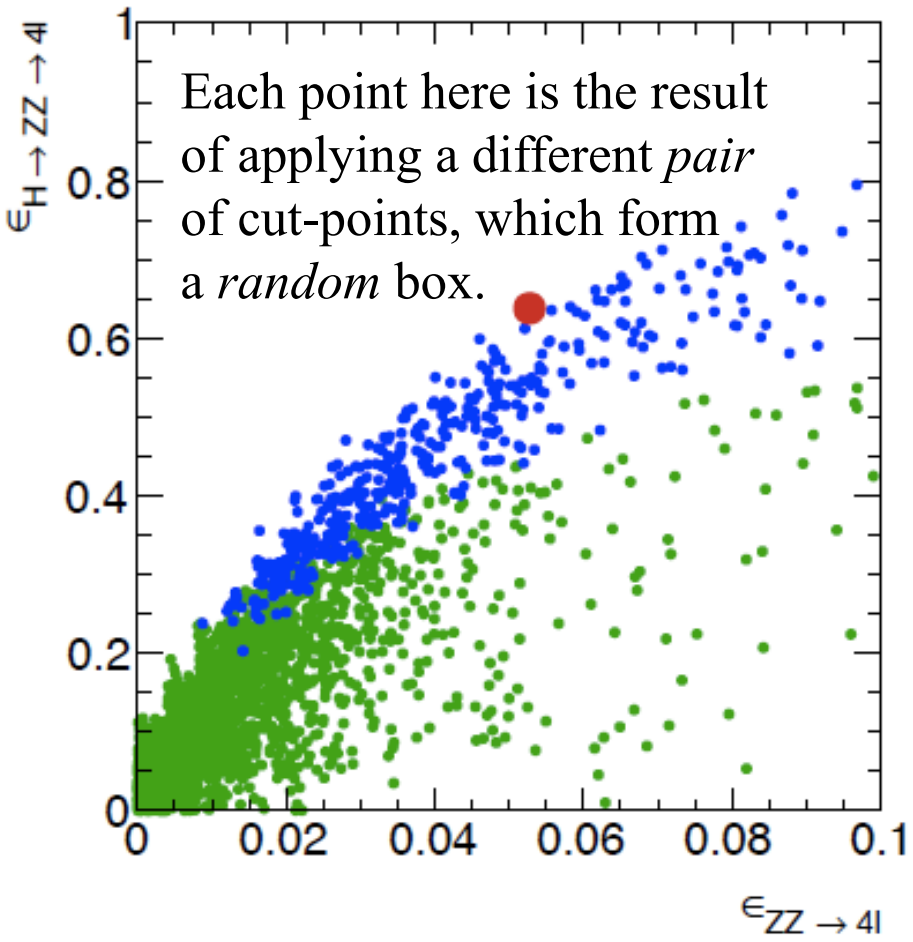
$x > x_i$, $y > y_i$

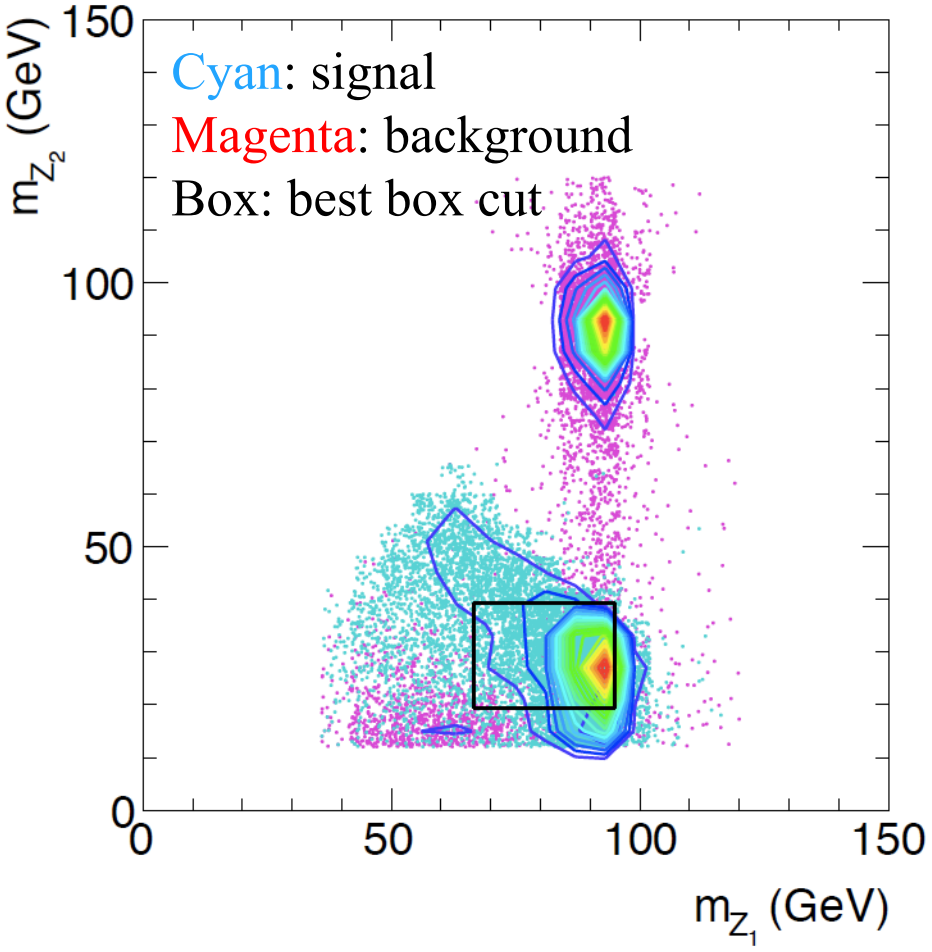


N_{tot} = # events before cuts
N_{cut} = # events after cuts
Fraction = N_{cut}/N_{tot}

H.B.P. et al., Proceedings, CHEP 1995

Higgs Boson to ZZ to 4 Leptons





The red point gives $p(s \mid x) / p(b \mid x) \sim 1 / 1$

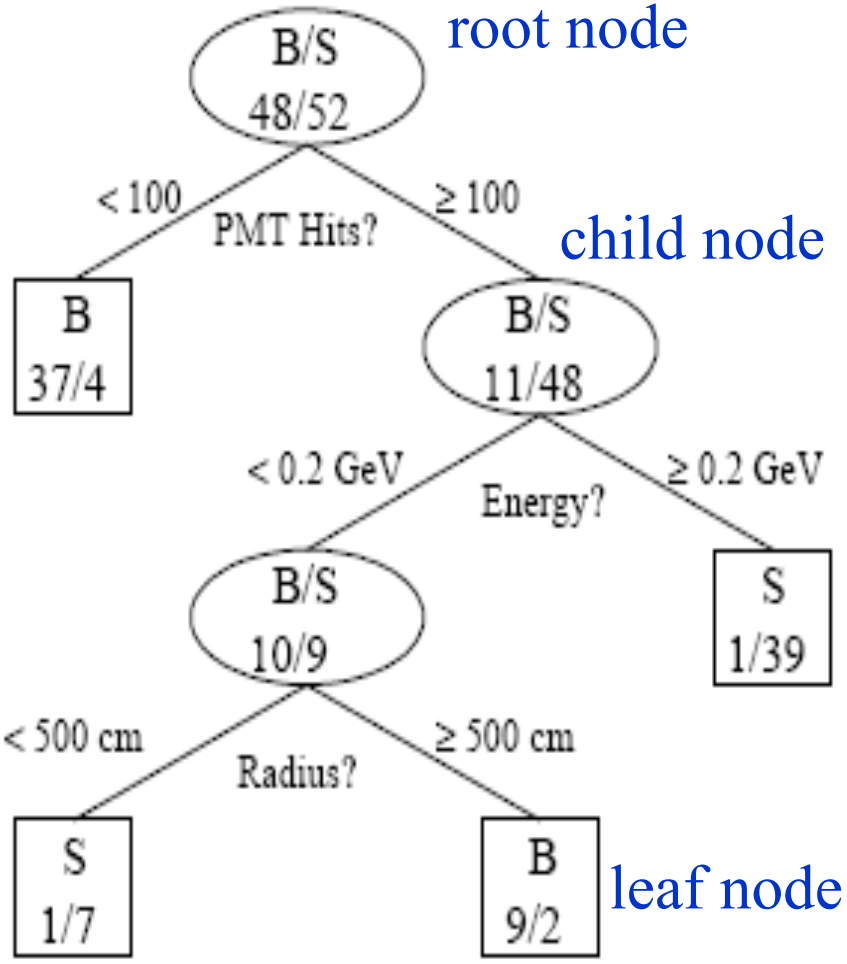
Decision Trees

Decision Trees

Decision tree:
a sequence of if then else statements.

Basic idea: recursively partition the space $\{x\}$ into regions of increasing purity.

Geometrically, a decision tree is a *d-dimensional* histogram in which the bins are built using recursive *binary* partitioning.

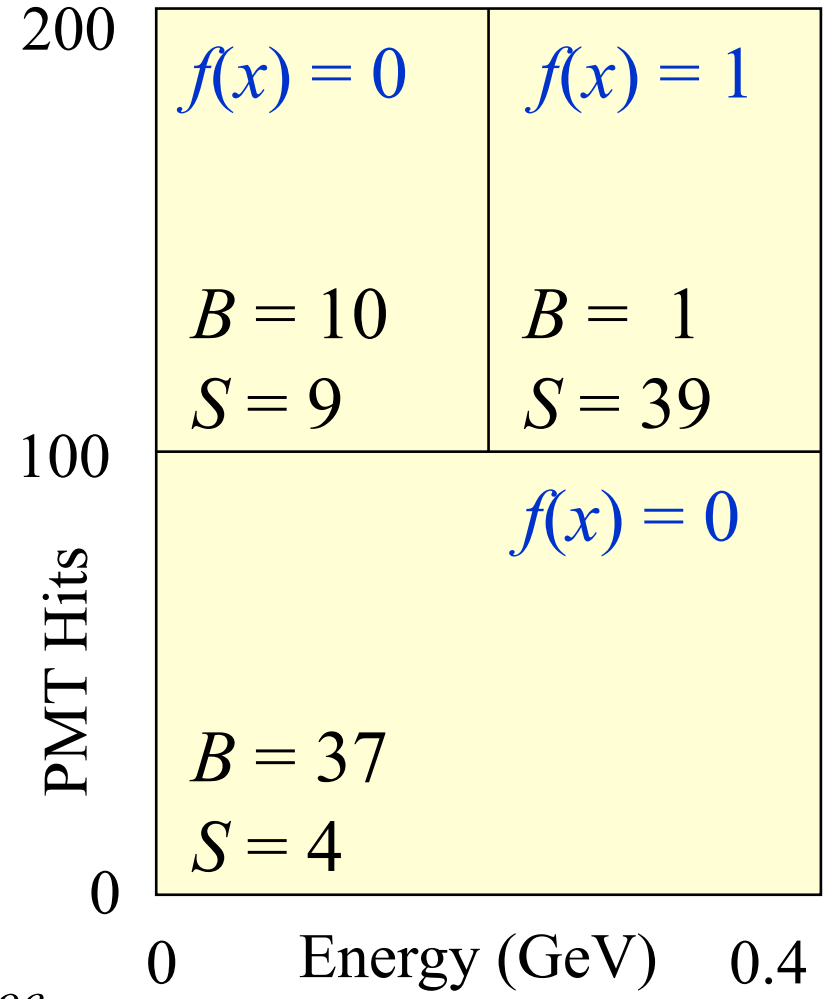


MiniBoone, Byron Roe

Decision Trees

To each bin, we associate the value of the function $f(x)$ to be approximated.

That way, we arrive at a piecewise constant approximation of $f(x)$.



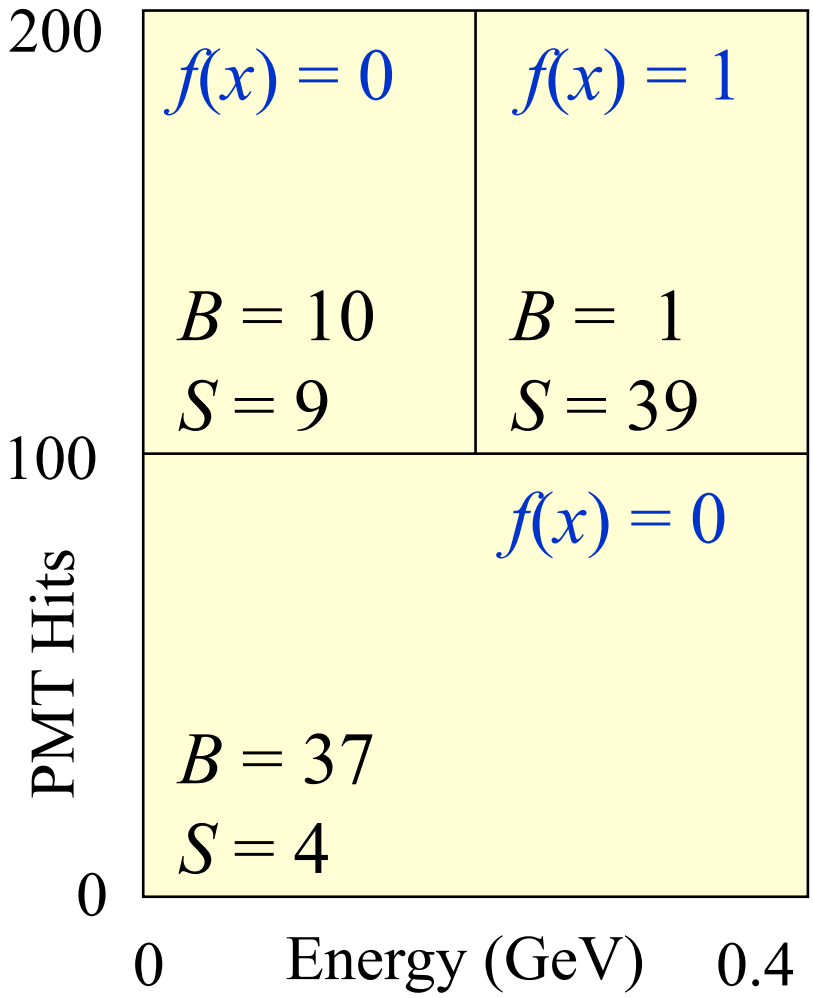
MiniBoone, Byron Roe

Decision Trees

For each variable, find the best partition ("cut"), defined as the one that yields the greatest *decrease* in impurity

= **Impurity** (parent bin)
– **Impurity** ("left"-bin)
– **Impurity** ("right"-bin)

Then choose the best partition among all partitions, and repeat with each child bin.



Decision Trees

The most common impurity measure is the Gini index (Corrado Gini, 1884-1965):

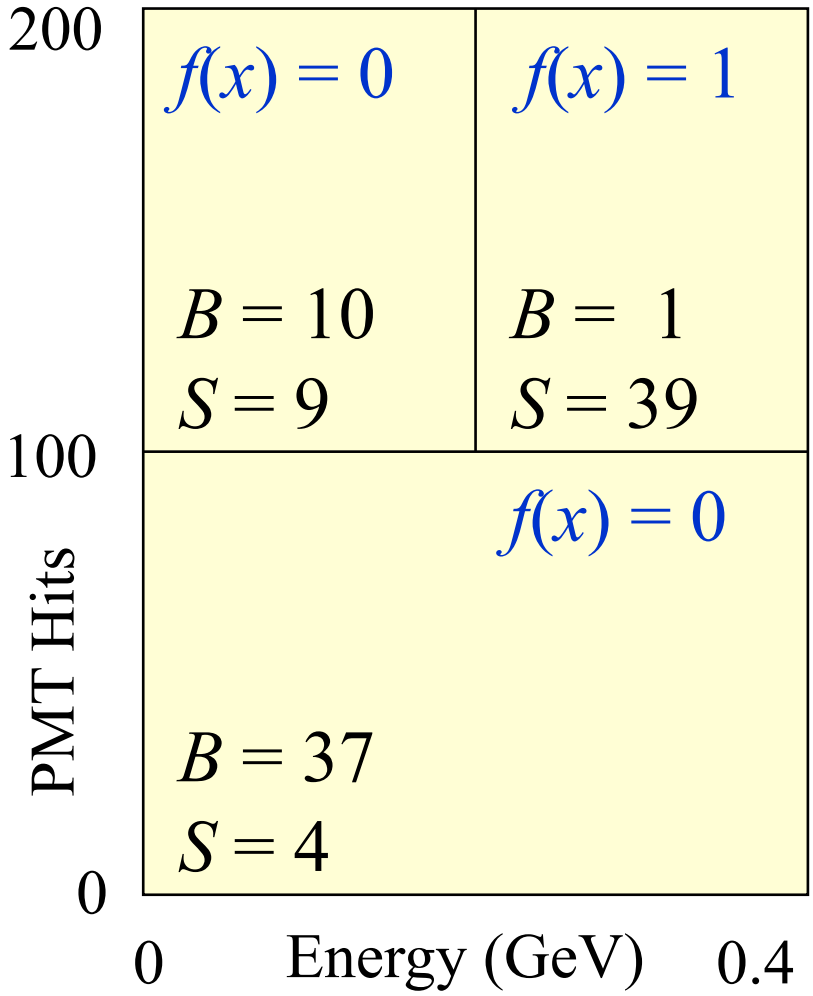
Gini index = $p\,(1 - p)$

where p is the purity

$p = S / (S + B)$

$p = 0$ or 1 = maximal purity

$p = 0.5$ = maximal impurity



Summary

- Multivariate methods can be applied to many aspects of data analysis, including classification and function approximation.
- We considered a simple example of classification using the random grid search and we introduced decision trees.
- In Lecture 2, we shall apply decision trees to the wine tasting problem.