Multivariate Discriminants
Lectures 4.1, 4.2

Harrison B. Prosper

Florida State University

INFN School of Statistics 2013

Vietri sul Mare (Salerno, Italy)

3 – 7 June, 2013

Outline

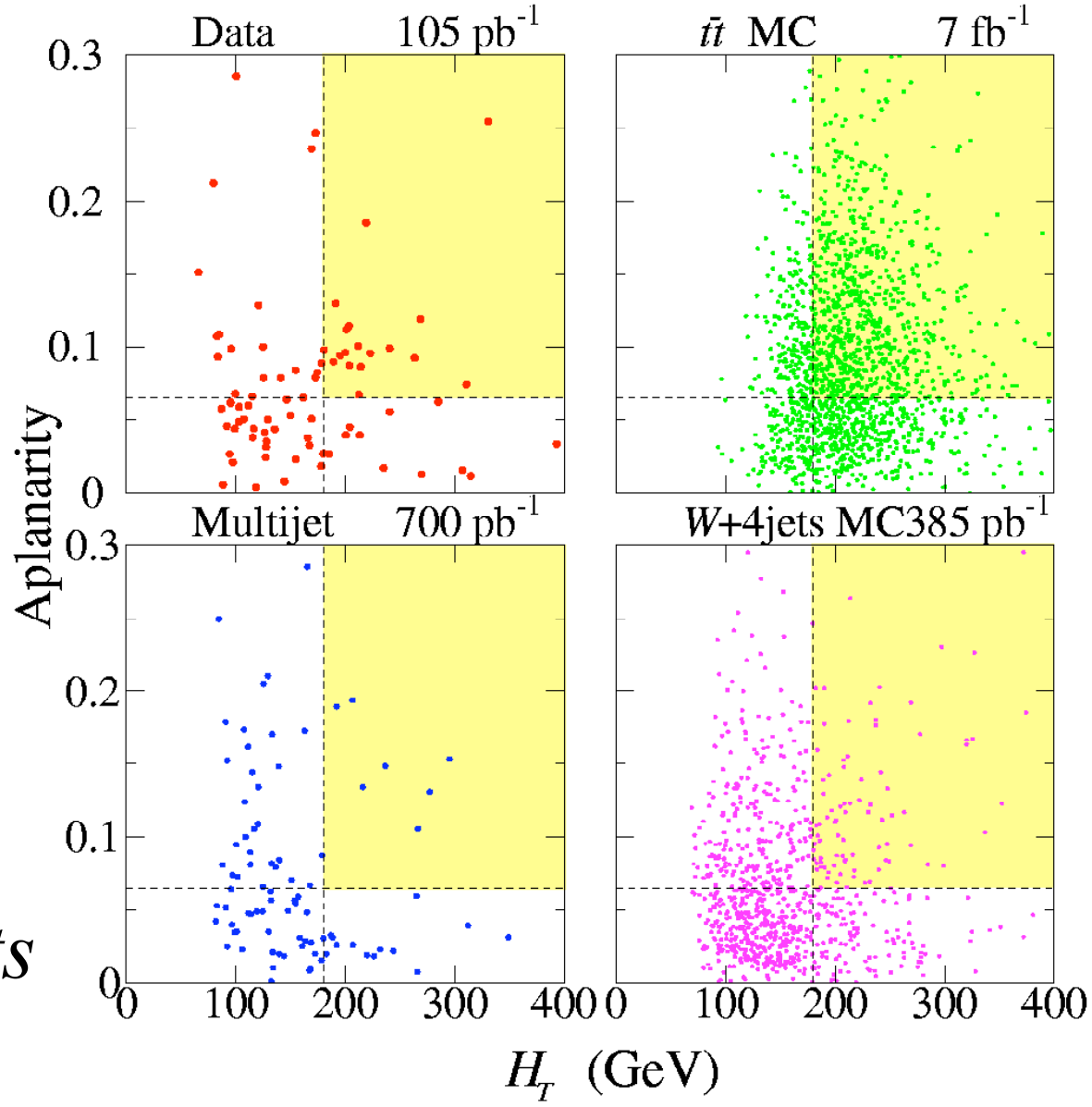
- Lecture 4.1
  - Introduction
  - Classification
    - In Theory
    - In Practice
- Lecture 4.2
  - Classification
    - In Practice

Introduction

Introduction – Multivariate Data

DØ 1995
Top quark discovery

$x = (A, H_T)$



$p\bar{p} \rightarrow t\bar{t} \rightarrow l + jets$

Introduction – General Approaches

Two general approaches:

Machine Learning

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

Bayesian Learning

Given training data $\boldsymbol{T}$, a function space $\{f\}$, the likelihood of the training data, and a prior defined on the space of functions, infer the mapping $y = f(x)$.

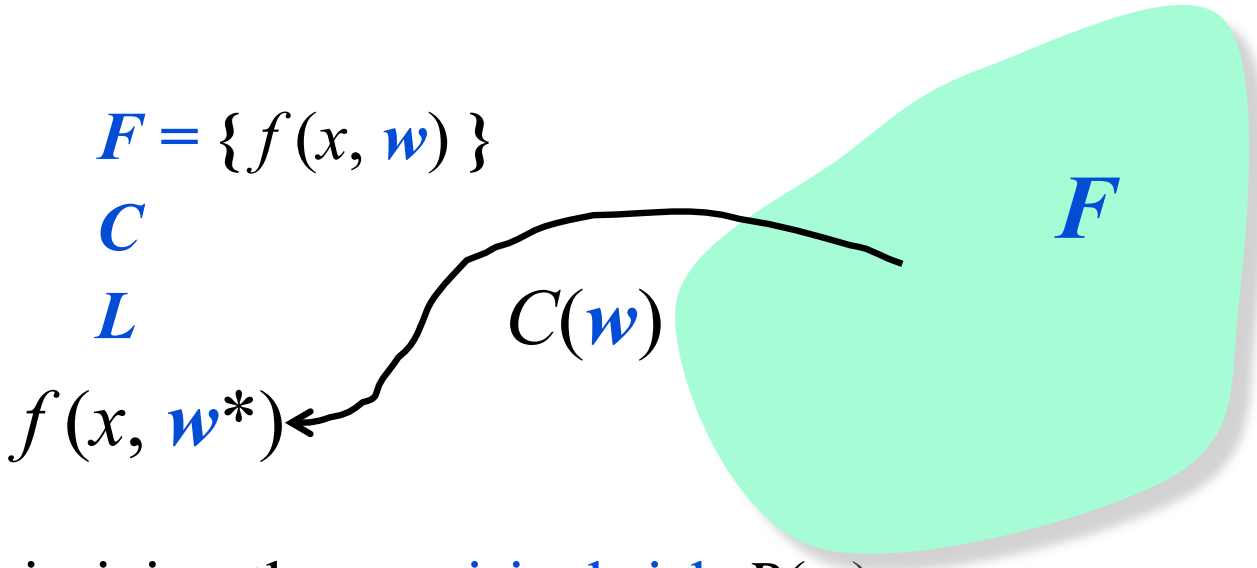
Machine Learning

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 choosing badly

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 mean square empirical risk

$$R[f_w] = \frac{1}{N}\sum_{i=1}^{N}[y_i - f(x_i, w)]^2 + C(w)$$

Bayesian Learning

Choose

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

Likelihood $p(T \mid w)$, $T = (y, x)$

Loss function L

Prior $p(w)$

Method

Use Bayes' theorem to assign a probability (density)

$$p(w \mid T) = p(T \mid w)\, p(w) / p(T)$$
$$= p(y \mid x, w)\, p(x \mid w)\, p(w) / p(y \mid x)\, p(x)$$
$$\sim p(y \mid x, w)\, p(w) \qquad \text{(assuming } p(x \mid w) = p(x)\text{)}$$

to *every* function in the function space.

Bayesian Learning

Given, the posterior density $p(\boldsymbol{w} \mid T)$, and *new* data x one computes the (predictive) distribution

$$p(y \mid x, T) = \int p(y \mid x, w)\, p(w \mid T)\, dw$$

If a definite value for y is needed for every x, this can be obtained by minimizing the (risk) function,

$$R[f_w] = \int L(y, f)\, p(y \mid x, T)\, dy$$

which for $L = (y - f)^2$ approximates $f(x)$ by the average

$$f(x) \simeq \bar{y}(x, T) \equiv \int y\, p(y \mid x, T)\, dy$$

Bayesian Learning

Suppose that ***y*** has only two values **0** and **1**, for every x, then

$$f(x) = \int y\, p(y \mid x, T)\, dy$$

reduces to

$$f(x) = p(1 \mid x, T)$$

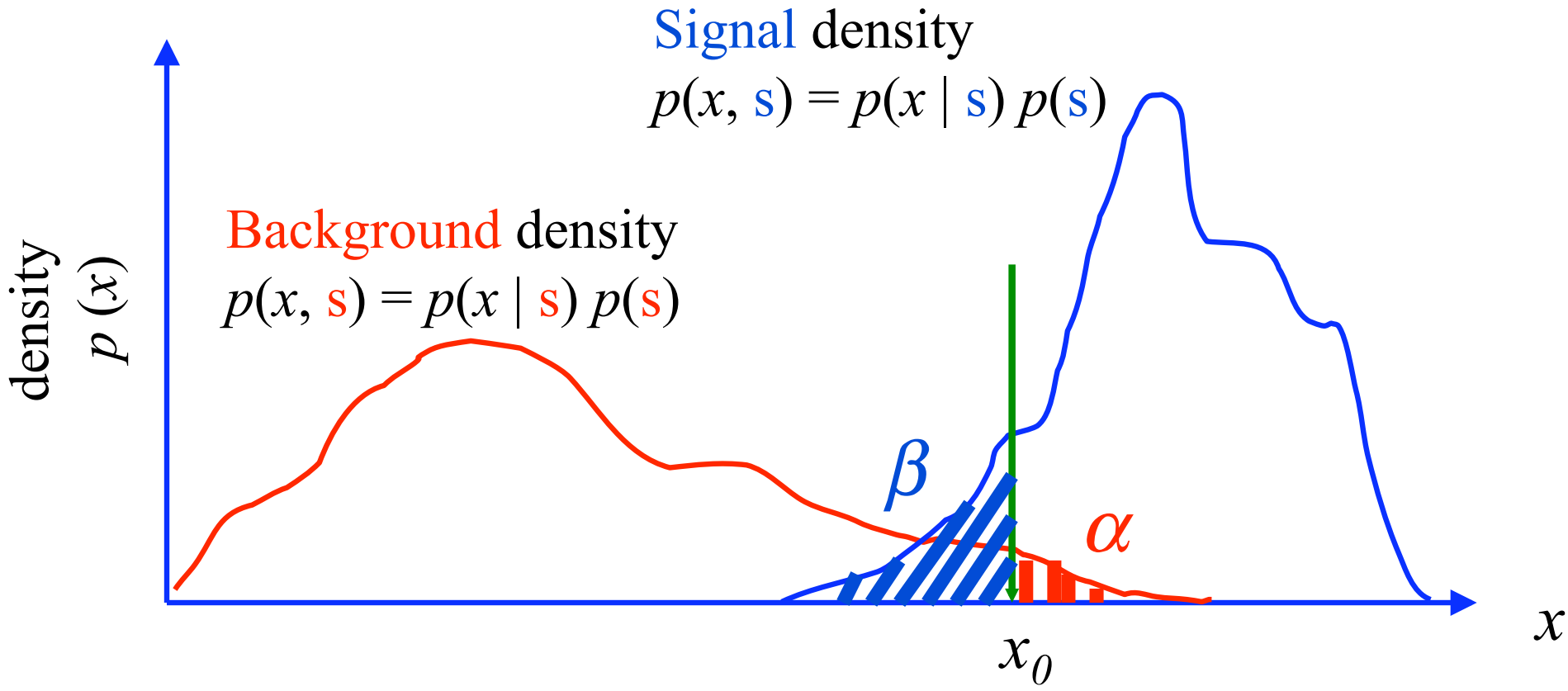
where $y = \mathbf{1}$ is associated with objects to be kept and $y = \mathbf{0}$ with objects to be discarded. For example, in an e-mail filter, we can reject junk e-mail using the (*complement of the*) rule

if $p(1 \mid x, T) > q$ *accept* x

which is called the **Bayes classifier**

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 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: The Bayes Connection

Consider the mean squared risk in the limit $N \to$ infinity,

$$R[f] = \frac{1}{N}\sum_{i=1}^{N}[y_i - f(x_i, w)]^2 + C(w)$$

$$\to \int dx \int dy [y - f(x, w)]^2 \, p(y, x)$$

$$= \int dx \, p(x) \left[\int dy \, (y - f)^2 \, p(y \mid x) \right]$$

where we have written $p(y \mid x) = p(y, x) / p(x)$ and where we have assumed that the effect of the constraint (in this limit) is negligible.

Classification: The Bayes Connection

Now minimize the functional $R[f]$ with respect to f. If the function f is sufficiently flexible, then $R[f]$ will reach its absolute minimum. Then for any small change δf in f

$$\delta R[f] = 2\int dx\, p(x)\delta f\left[\int dy\,(y - f)\,p(y \mid x)\right] = 0$$

If we require the above to hold for all variations δf, for all x, then the term in brackets must be zero.

$$\int dy\,(y - f)\,p(y \mid x) = 0$$

Classification: The Bayes Connection

Since for the signal class *s*, $y = 1$, while for the background, *b*, $y = 0$, we obtain the important result:

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

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: The Discriminant

In practice, we typically do not use $p(s \mid x)$ directly, but rather the **discriminant**

$$D(x) = \frac{p(x \mid s)}{p(x \mid s) + p(x \mid b)} = \frac{\exp(\lambda)}{1 + \exp(\lambda)},$$

$$\text{where} \quad \lambda(x) \equiv \ln[p(x \mid s) / \ p(x \mid b)]$$

This is fine because $p(s \mid x)$ is a *one-to-one* function of $D(x)$ and therefore both have the same discrimination power

$$p(s \mid x) = \frac{D(x)}{D(x) + [1 - D(x)] / a}, \quad a = p(s) / \ p(b)$$

Classification
In Practice

Classification: In Practice

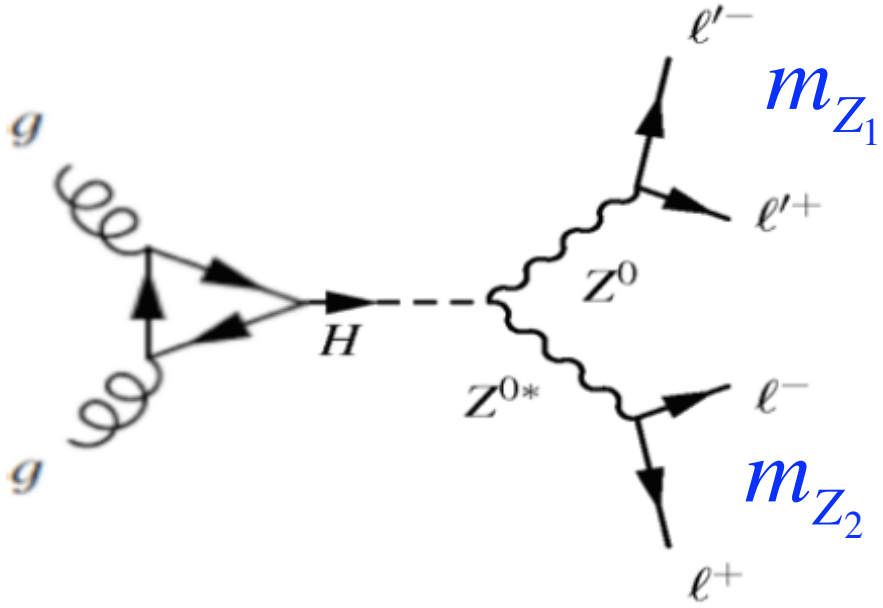
Here is a *short* list of multivariate (MVA) methods that can be used for classification:

- 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 Example

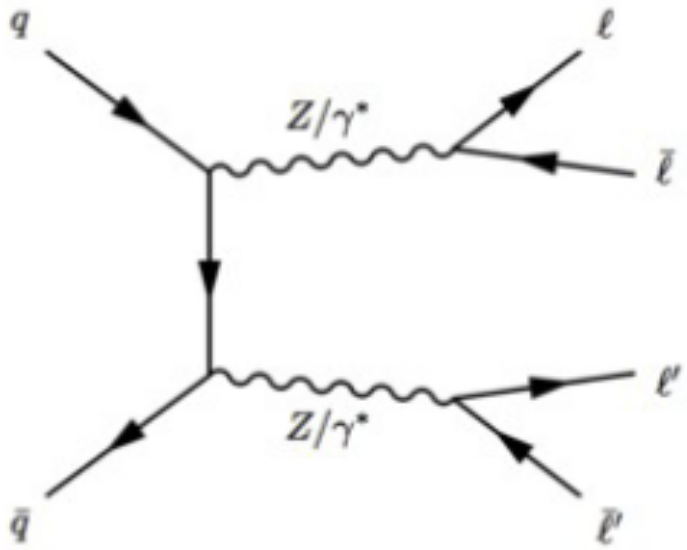
Example – H to ZZ to 4 Leptons

Signal



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

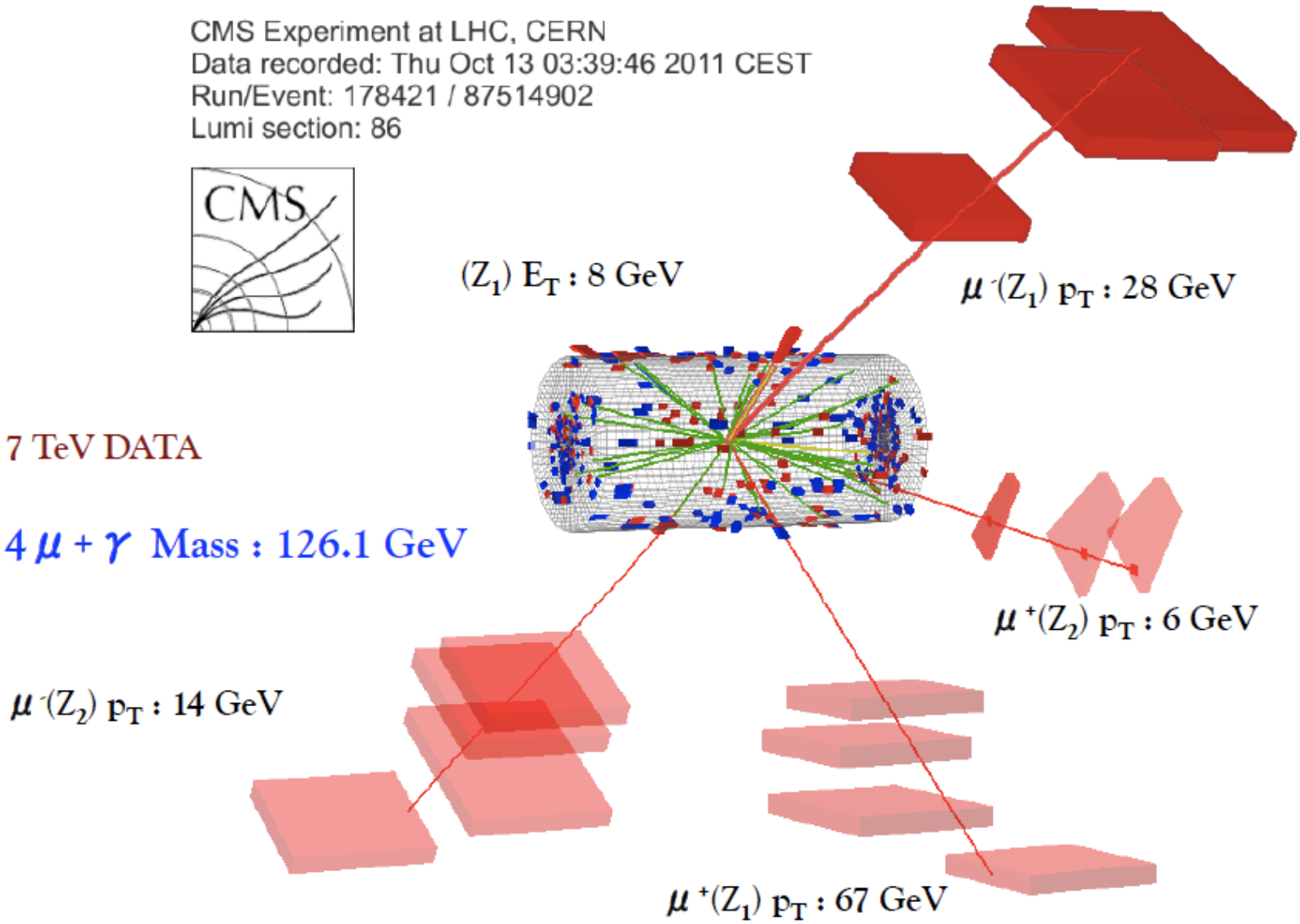
Background



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

We shall use this example to illustrate a few of the methods.
We start with $p(s) / p(b) \sim 1 / 20$ and use $x = (m_{Z1}, m_{Z2})$

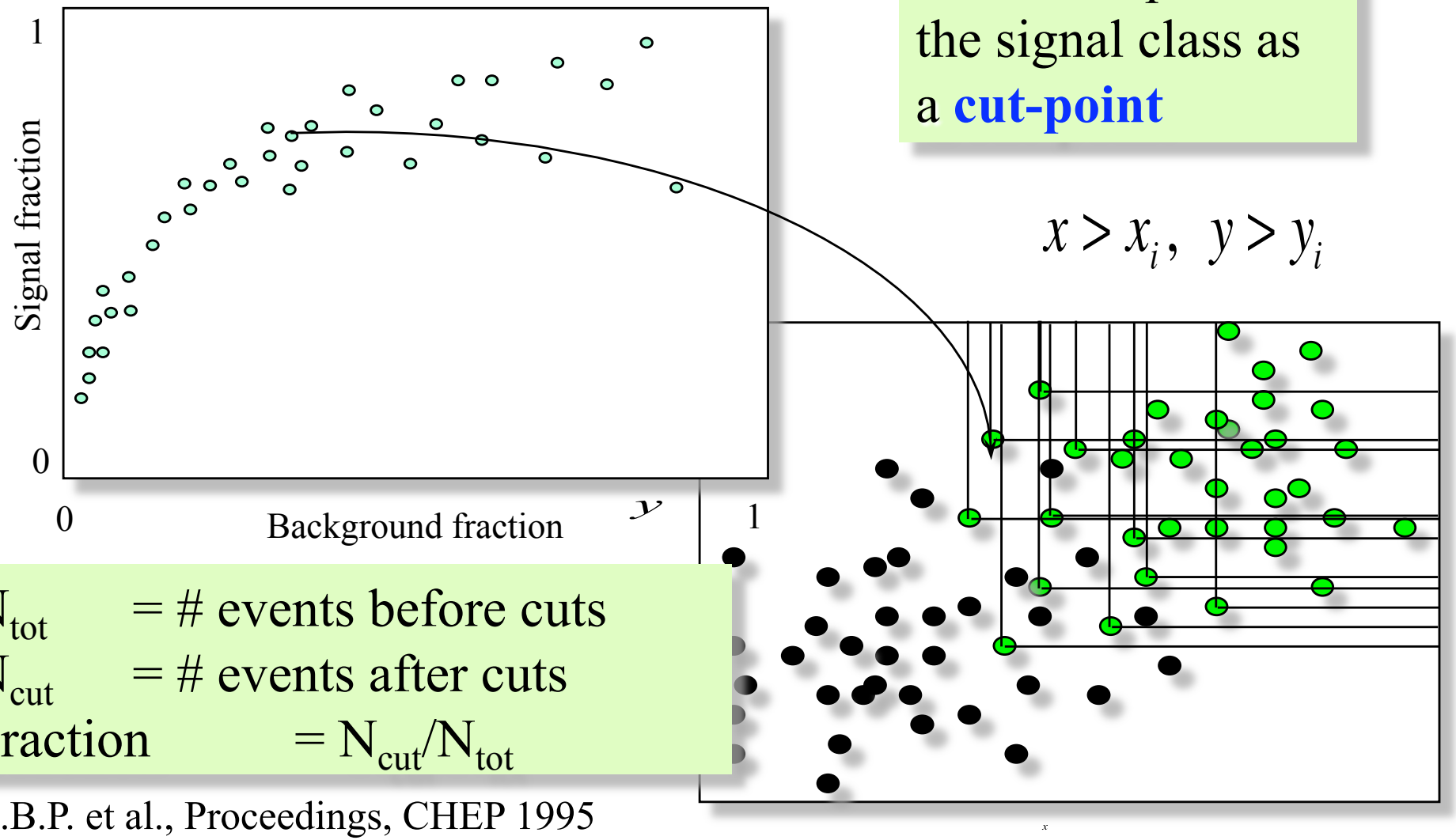
A 4-Lepton Event from CMS



Random Grid Search

Random Grid Search (RGS)

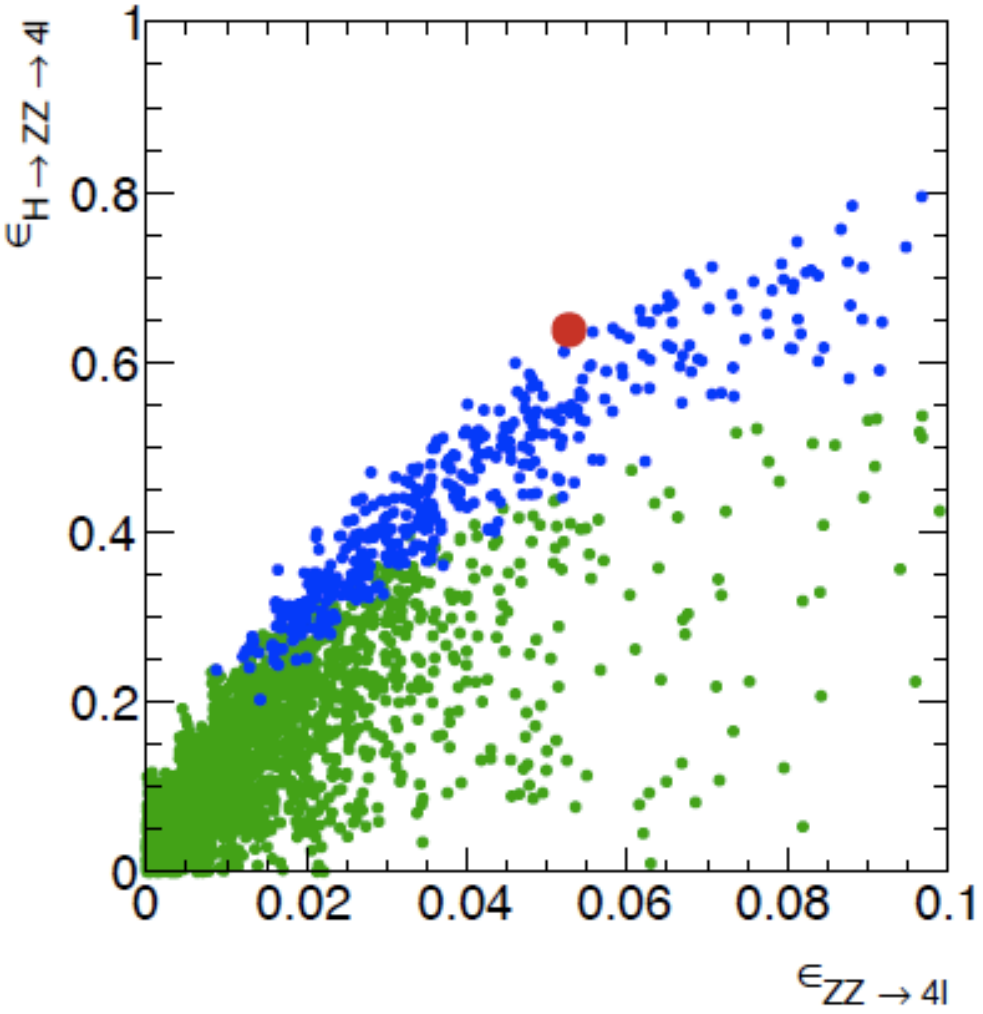
Take each point of the signal class as a **cut-point**

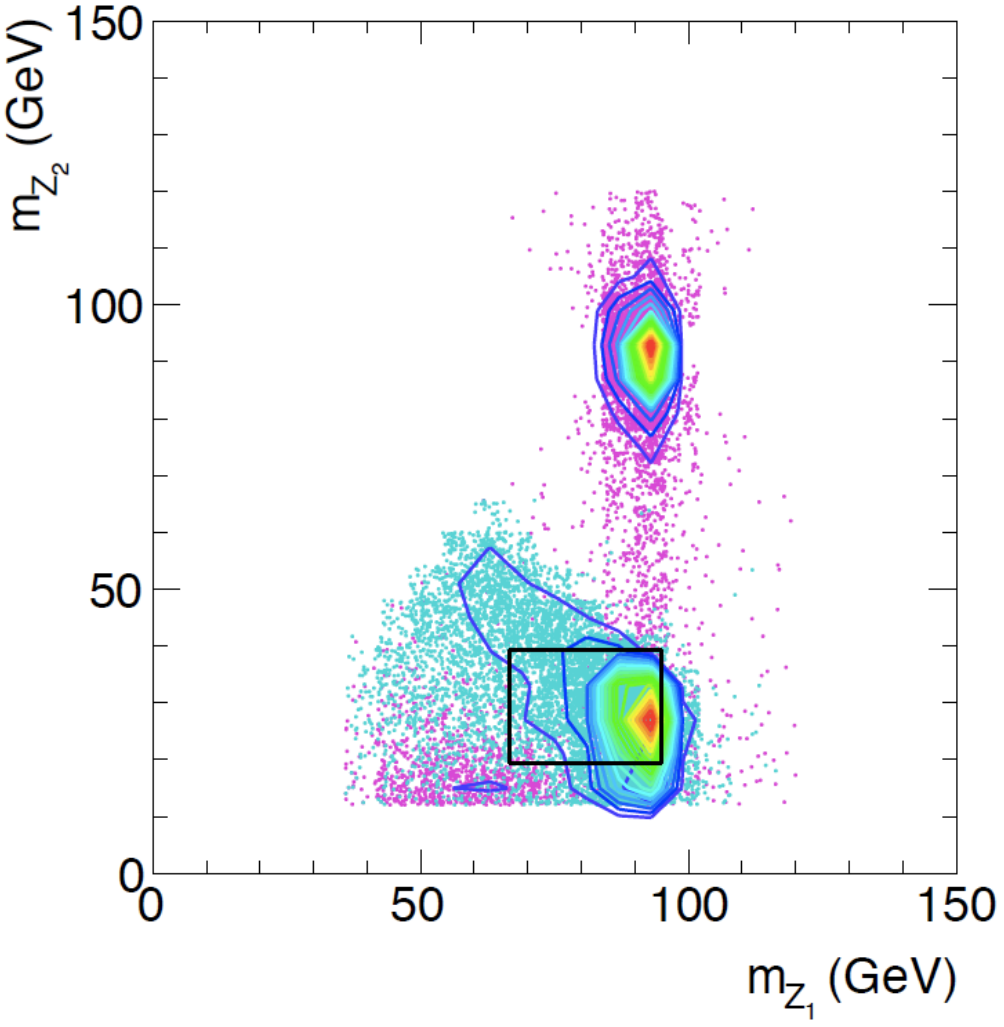


N_{tot} = # events before cuts
N_{cut} = # events after cuts
Fraction = N_{cut}/N_{tot}

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

Example – H to ZZ to 4Leptons





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

Linear & Quadratic Discriminants

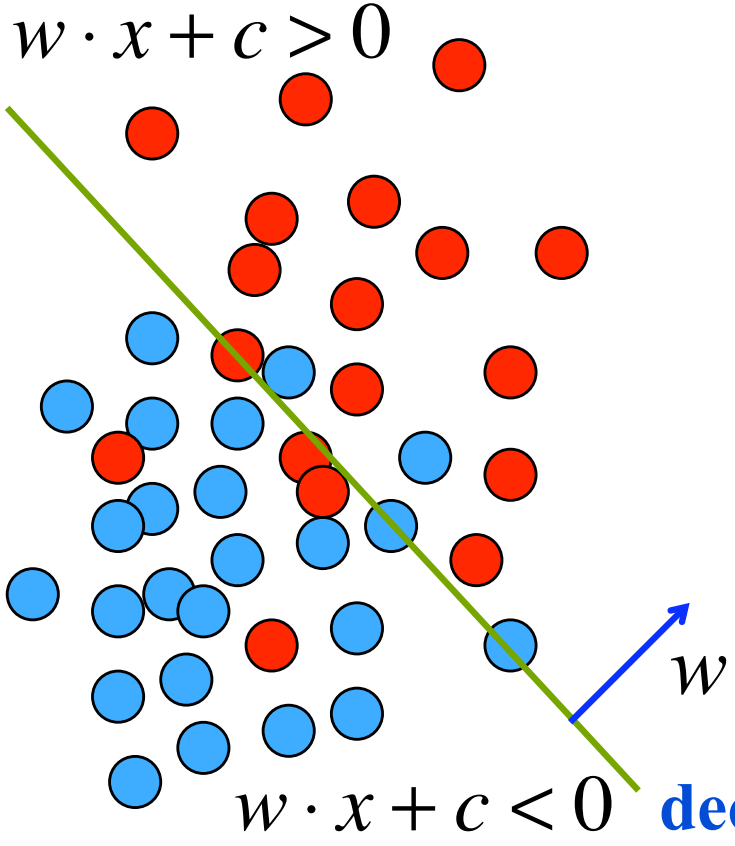
Fisher (Linear) Discriminant

$$B(x) = \frac{p(x \mid s)\,p(s)}{p(x \mid b)\,p(b)}$$

Take $p(x \mid s)$ and $p(x \mid b)$ to be Gaussian (and dropping the constant term) yields

$$\lambda(x) = \ln \frac{G\left(x \mid \mu_s, \Sigma\right)}{G\left(x \mid \mu_b, \Sigma\right)} \rightarrow w \cdot x + c$$

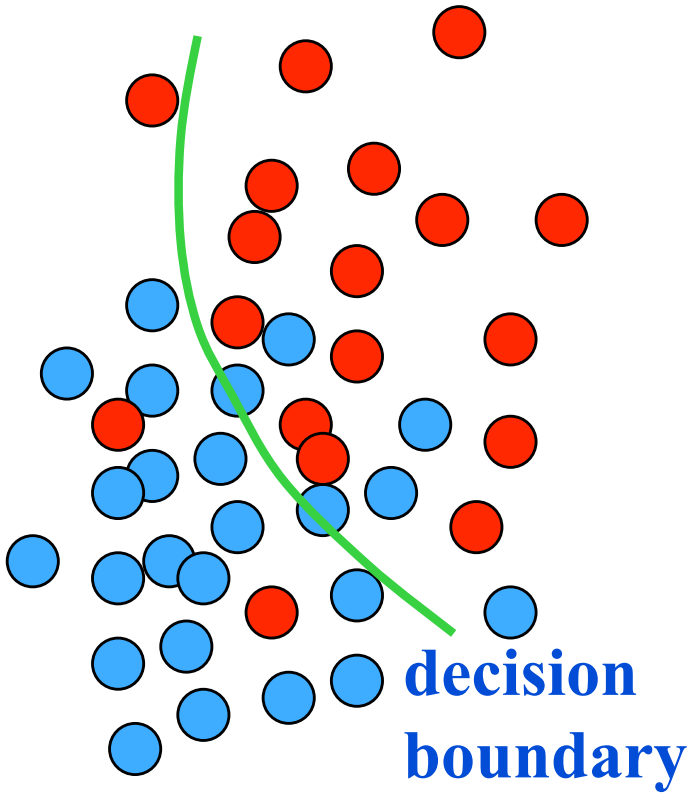
$$w \propto \Sigma^{-1}(\mu_s - \mu_b)$$



decision boundary

Quadratic Discriminant

If we use *different* covariance matrices for the signal and the background densities, we obtain the **quadratic discriminant**:



$$\lambda(x) = (x - \mu_b)^T \Sigma_b^{-1} (x - \mu_b) - (x - \mu_s)^T \Sigma_s^{-1} (x - \mu_s)$$

a fixed value of which defines a curved surface that partitions the space $\{x\}$ into signal-rich and background-rich regions

Naïve Bayes
(Likelihood Ratio Discriminant)

Naïve Bayes

In this method the d-dimensional density $p(x)$ is replaced by the *approximation*

$$\hat{p}(x) = \prod_{i=1}^{d} q(x_i)$$

where $q(x_i)$ are the 1-D marginal densities of $p(x)$

$$q(x_i) = \int_{\{x_j : x_j \neq x_i\}} p(x)dx$$

Naïve Bayes

The naïve Bayes estimate of $D(x)$ is given by

$$D(x) = \frac{\exp(\hat{\lambda})}{1+\exp(\hat{\lambda})}, \quad \hat{\lambda} = \sum_i \ln[q(x_i \mid s) / q(x_i \mid b)]$$

In spite of its name, this method can sometimes provide very good results.

And, of course, if the variables truly are statistically independent then this approximation is expected to be good.

Kernel Density Estimation

Kernel Density Estimation

Basic Idea

Place a kernel function at each point and adjust their widths to obtain the best approximation

Parzen Estimation (1960s)

$$p(x) = \frac{1}{N} \sum_n \varphi\left(\frac{x - x_n}{h} \right) \qquad 1 \le n \le N$$

Mixtures

$$p(x) = \sum_j \varphi(x, j)\, q(j) \qquad j << N$$

Kernel Density Estimation

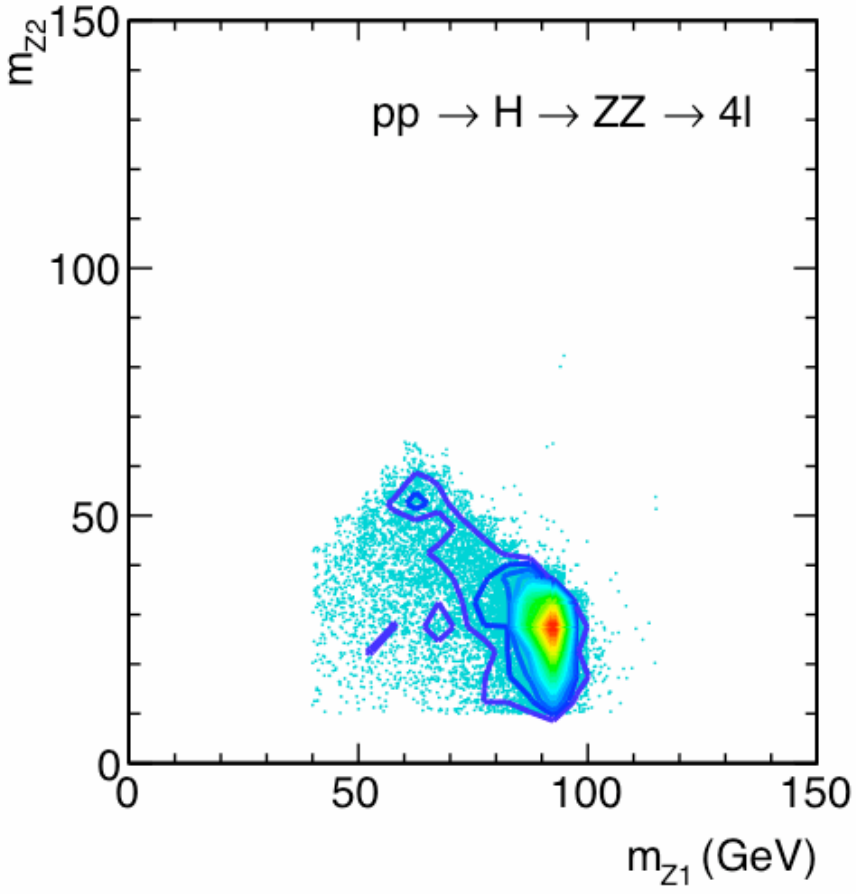
Why does it work? In the limit N goes to infinity

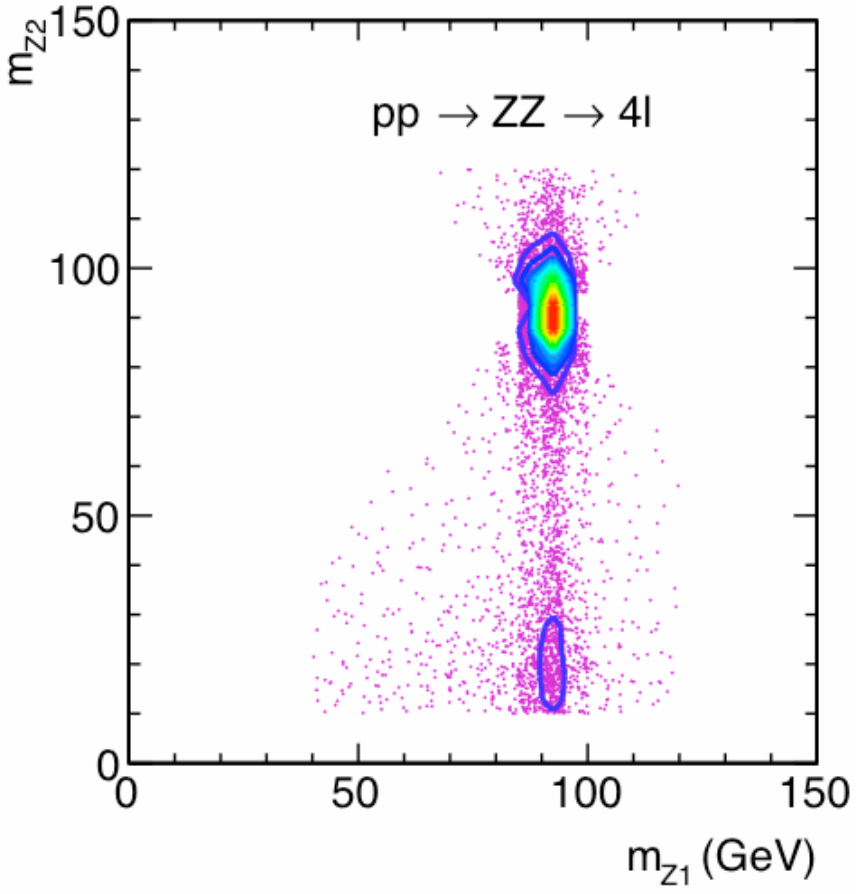
$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\varphi\left(\frac{x - x_n}{h}\right) \to \int \varphi\left(\frac{x - z}{h}\right) p(z)dz$$

the true density $p(x)$ will be recovered provided that the kernel converges to a d-dimensional δ-function:

$$\varphi\left(\frac{x - x_n}{h}\right) \to \delta^d(x - z)$$

KDE of Signal and Background





3000 points / KDE

PART II

Support Vector Machines

Support Vector Machines

This is a generalization of the Fisher discriminant (Boser, Guyon and Vapnik, 1992).

Basic Idea

Data that are non-separable in *d*-dimensions may be better separated if mapped into a space of higher (usually, infinite) dimension

$$h : \Re^d \rightarrow \Re^\infty$$

As in the Fisher discriminant, a hyper-plane is used to partition the high dimensional space

$$f(x) = w \cdot h(x) + c$$

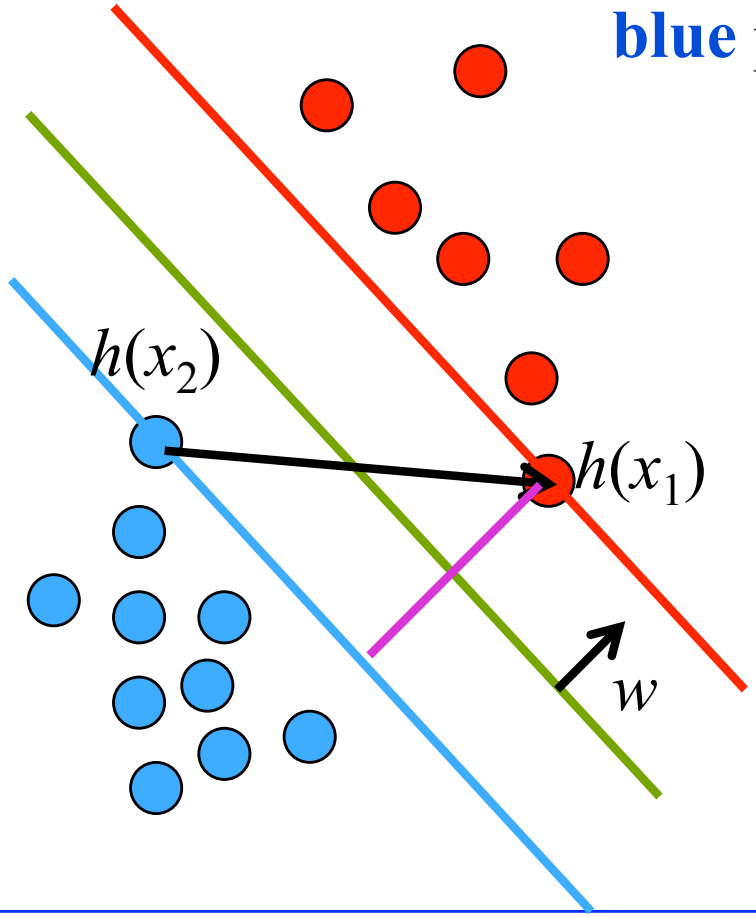
Support Vector Machines

Consider *separable* data in the high dimensional space

green plane: $w.h(x) + c = 0$

red plane: $w.h(x_1) + c = +1$

blue plane: $w.h(x_2) + c = -1$



subtract **blue** from **red**

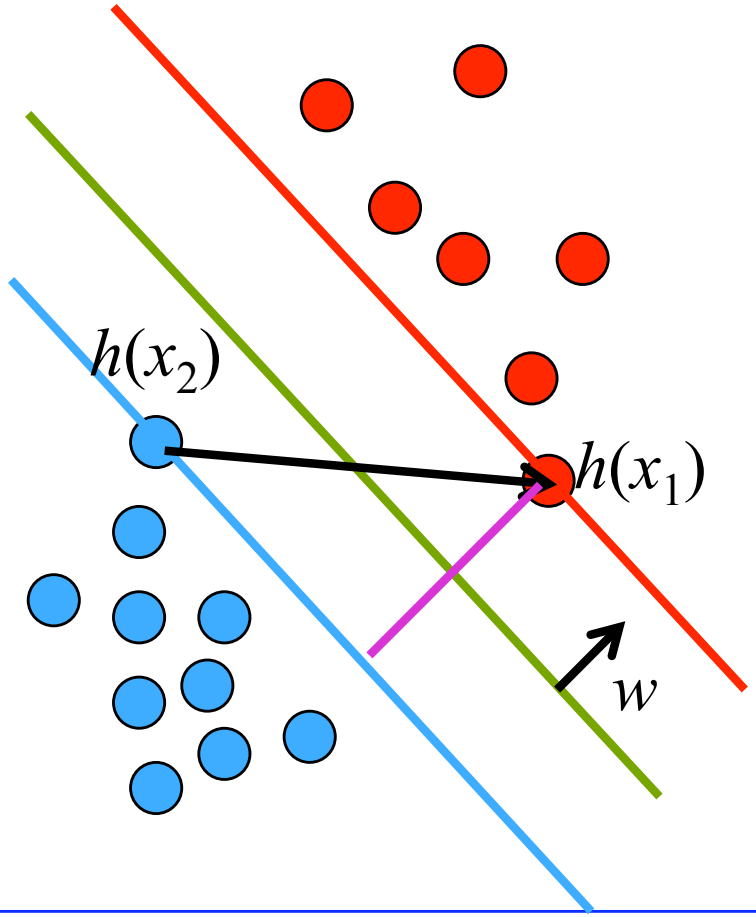
$w.[h(x_1) - h(x_2)] = 2$

and normalize the high dimensional vector w

$\hat{w}.[h(x_1) - h(x_2)] = 2/||w||$

Support Vector Machines

The quantity $m = \hat{w}.[h(x_1) - h(x_2)]$, the distance between the **red** and **blue** planes, is called the **margin**. The best separation occurs when the margin is as large as possible.



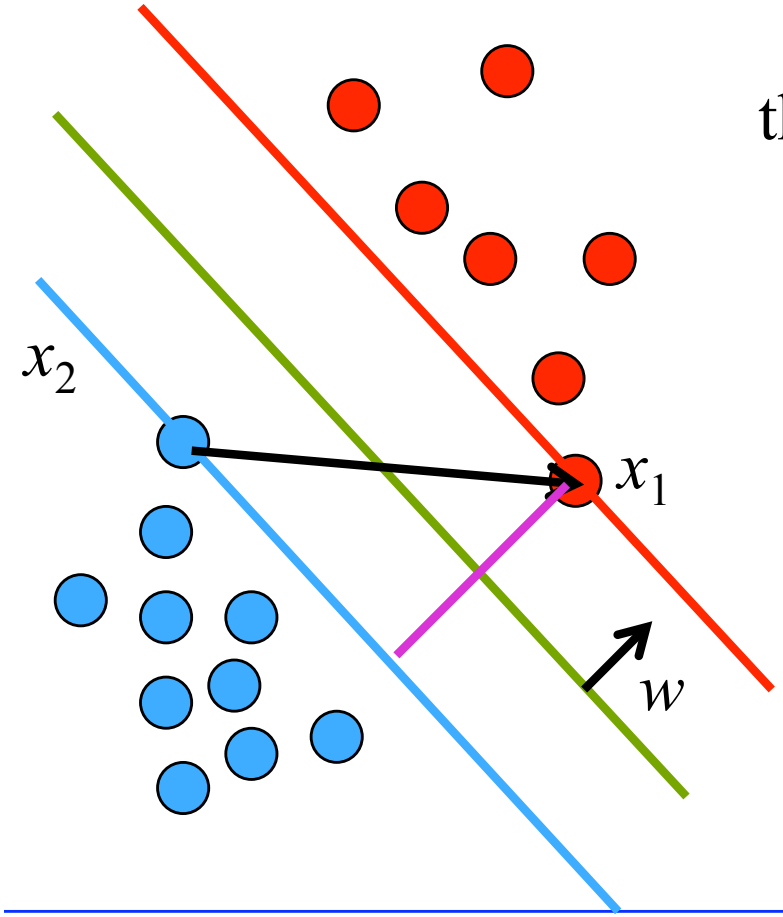
Note: because $m \sim 1/||w||$, maximizing the margin is equivalent to minimizing

$$||w||^2$$

Support Vector Machines

Label the **red** dots $y = +1$ and the **blue** dots $y = -1$. The task is to minimize $\|w\|^2$ subject to the constraint

$$y_i\,(w.h(x_i) + c) \geq 1, \qquad i = 1 \ldots N$$



that is, the task is to minimize

$$L(w,c,\alpha) = \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(w \cdot h(x_i) + c \right) - 1 \right]$$

where the $\alpha > 0$ are Lagrange multipliers

Support Vector Machines

When $L(\mathrm{w},\mathrm{c},\alpha)$ is minimized with respect to w and c, the function $L(\mathrm{w},\mathrm{c},\alpha)$ can be transformed to

$$E(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j h(x_i) \cdot h(x_j)$$

At the minimum of $E(\alpha)$, the only non-zero coefficients α are those corresponding to points *on* the **red** and **blue** planes: that is, the so-called **support vectors**. The key idea is to replace the scalar product $h(x_i).h(x_j)$ between two vectors of infinitely many dimensions by a **kernel function** $K(x_i, x_j)$. The (unsolved) problem is how to choose the correct kernel for a given problem?

Neural Networks

Neural Networks

Given

$\boldsymbol{T} = \boldsymbol{x}, \boldsymbol{y}$

$\boldsymbol{x} = \{x_1, \ldots x_N\}, \boldsymbol{y} = \{y_1, \ldots y_N\}$

of N training examples

Find

Find a function $f(\boldsymbol{x}, \boldsymbol{w})$, with parameters $\boldsymbol{w}$, using **maximum likelihood**

Neural Networks

A typical likelihood for classification is

$$p(\boldsymbol{y} \mid \boldsymbol{x}, w) = \Pi_i \, n(\boldsymbol{x}_i, w)^y \, [1 - n(\boldsymbol{x}_i, w)]^{1-y}$$

where $y = 0$ for background events (e.g, junk e-mail)

$y = 1$ for signal events (e.g., "good" e-mail)

As noted in the previous lecture, **IF** $n(\boldsymbol{x}, \boldsymbol{w})$ is sufficiently flexible, then the maximum likelihood solution for $\boldsymbol{w}$ is $n(\boldsymbol{x}, \boldsymbol{w}) = p(s \mid \boldsymbol{x})$, *asymptotically*, that is, as we acquire more and more data

Historical Aside – Hilbert's 13th Problem

Problem 13: Prove the conjecture

In general, it is ***impossible*** to do the following:

$$f(x_1,\ldots,x_n) = F(\, g_1(x_1),\ldots,\, g_n(x_n)\,)$$

But, in 1957, Kolmogorov ***disproved*** Hilbert's conjecture!

Today, we know that functions of the form

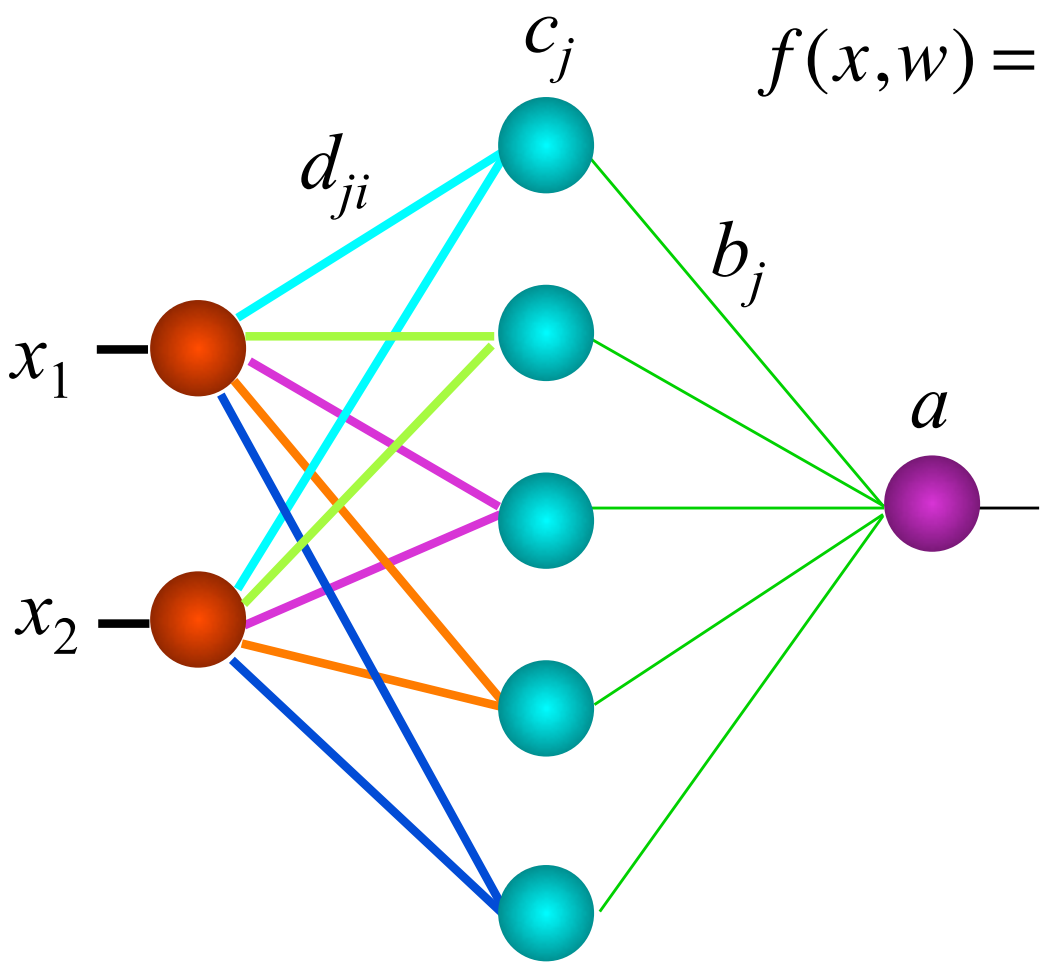
$$f(x_1,\cdots,x_I) = a + \sum_{j=1}^{H} b_j \tanh\left[\, c_j + \sum_{i=1}^{I} d_{ji} x_i \,\right]$$

can provide arbitrarily accurate approximations.

(Hornik, Stinchcombe, and White,

Neural Networks **2**, 359-366 (1989))

NN – Graphical Representation



$$f(x,w) = a + \sum_{j=1}^{H} b_j \tanh\left[c_j + \sum_{i=1}^{I} d_{ji} x_i \right]$$

$$n(x,w) = \frac{1}{1+\exp[-f(x,w)]}$$

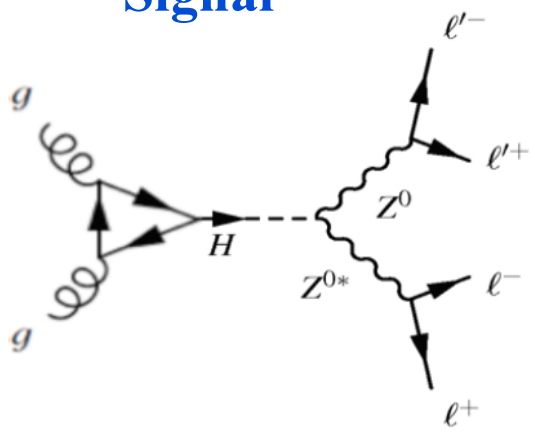
f is used for regression

n is used for classification

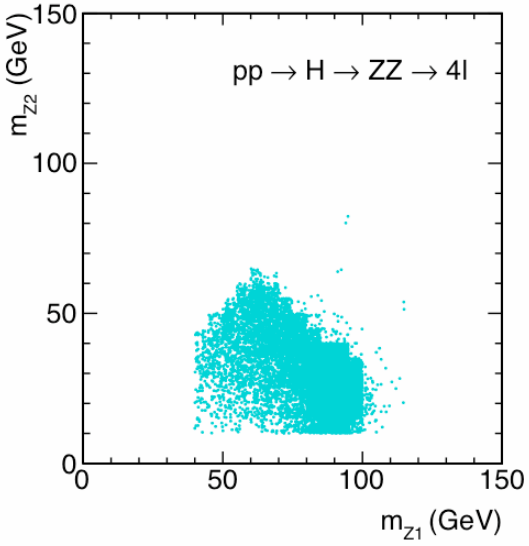
$w = a, b, c, d$

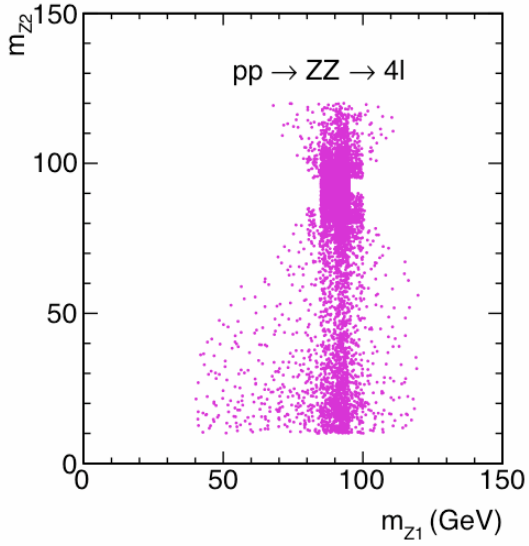
Example – H to ZZ to 4Leptons

Signal

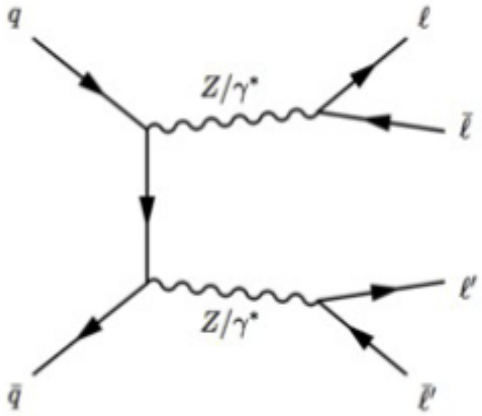


$pp \to H \to ZZ \to \ell^+\ell^-\ell'^+\ell'^-$

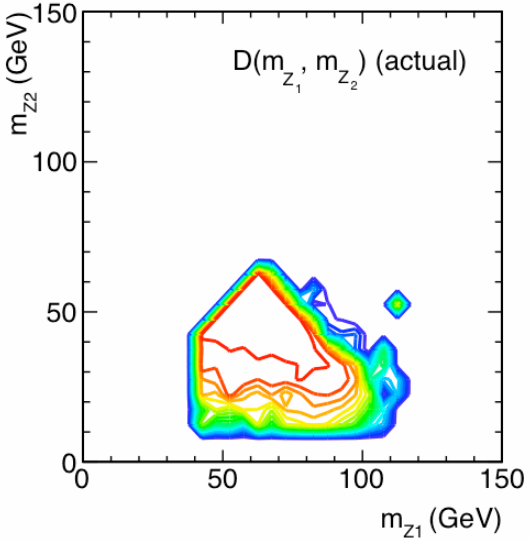


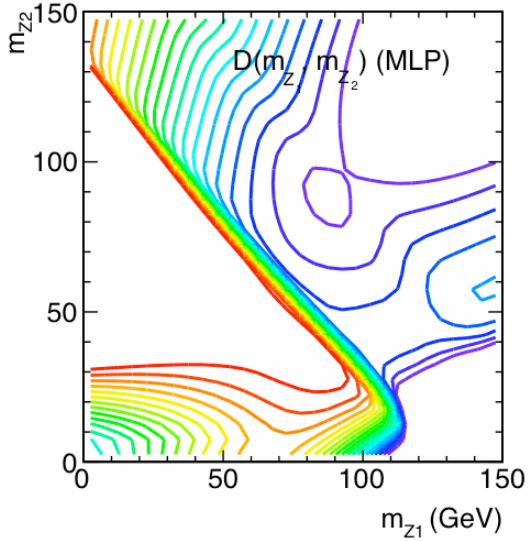


Background



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



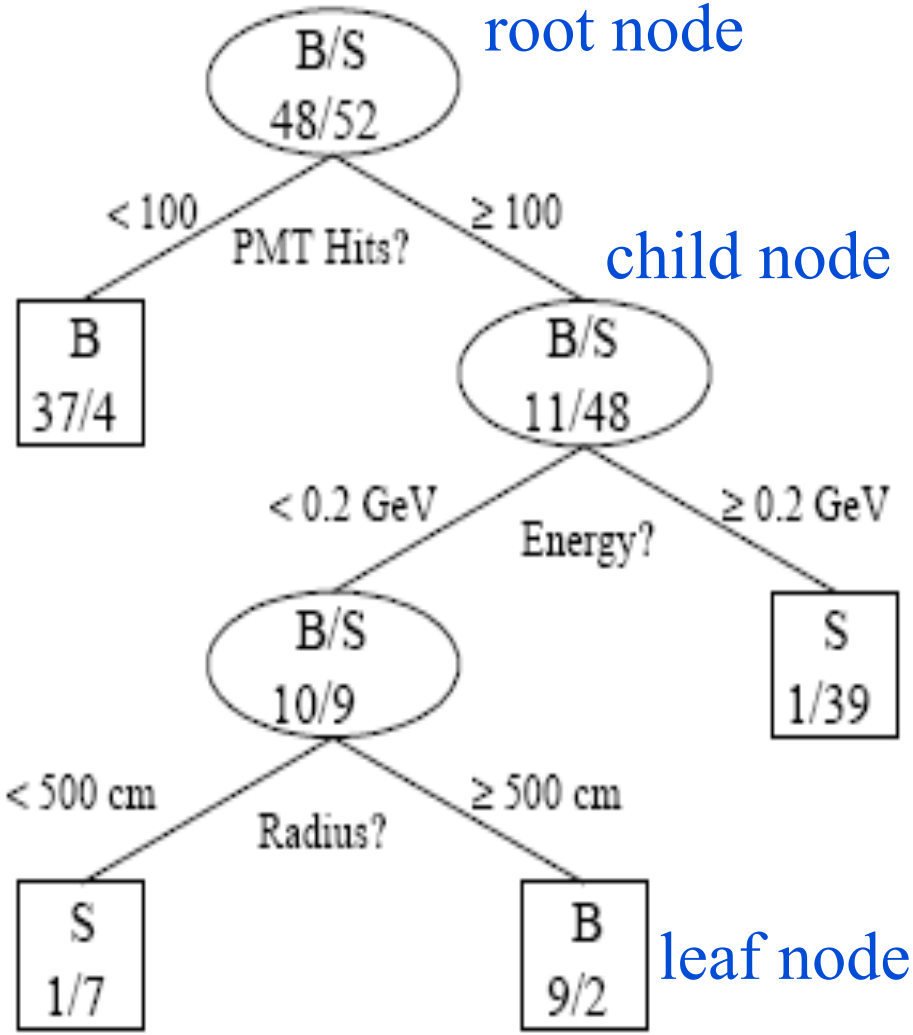


Decision Trees

Decision Trees

A decision tree is
a sequence of if then else
statements, that is, cuts

Basic idea: choose cuts that
partition the space $\{x\}$ into
regions of increasing purity
and do so recursively



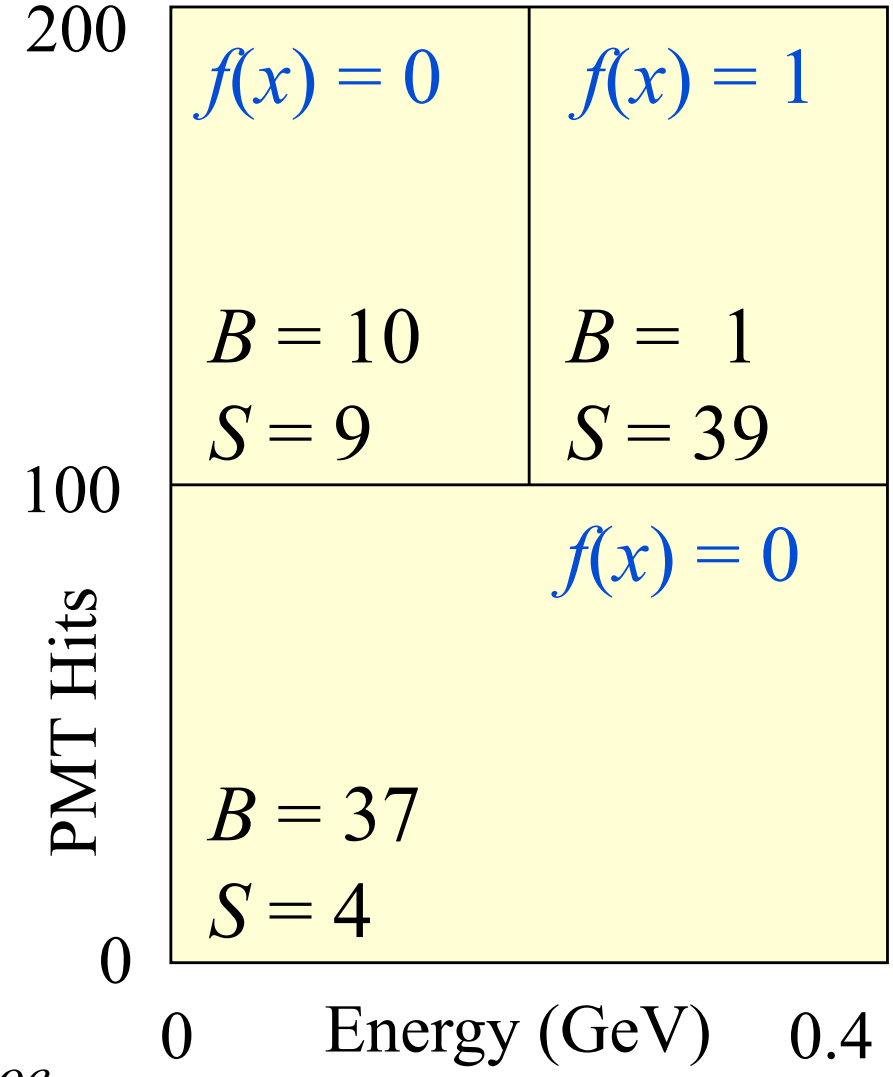
MiniBoone, Byron Roe

Decision Trees

Geometrically, a decision tree is simply a ***d*-dimensional histogram** whose bins are constructed recursively

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

This provides a **piecewise constant** approximation of $f(x)$.



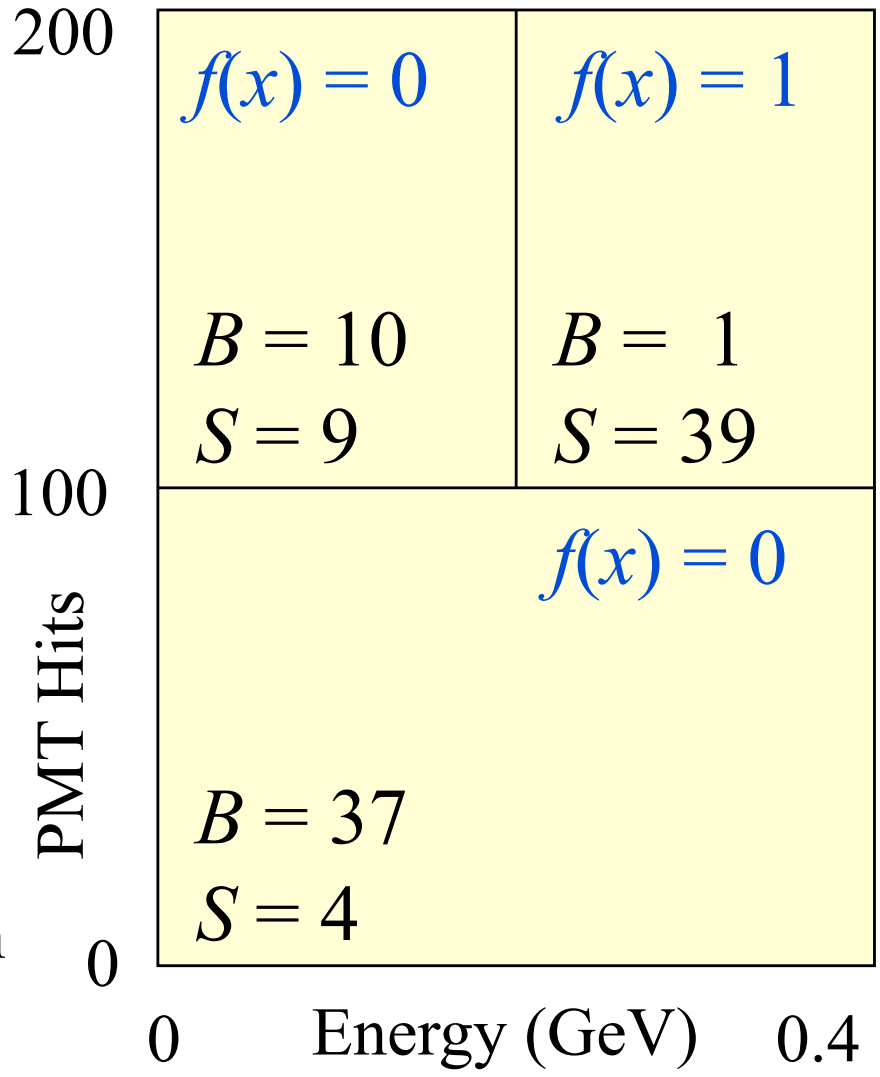
MiniBoone, Byron Roe

Decision Trees

For each variable find the best cut, defined as the one that yields the biggest

decrease in impurity
= **Impurity** (parent bin)
– **Impurity** (“left”-bin)
– **Impurity** (“right”-bin)

Then choose the best cut among these cuts, partition the space, and repeat with each child bin



Decision Trees

The most common impurity measure is the Gini index (Corrado Gini, 1912):

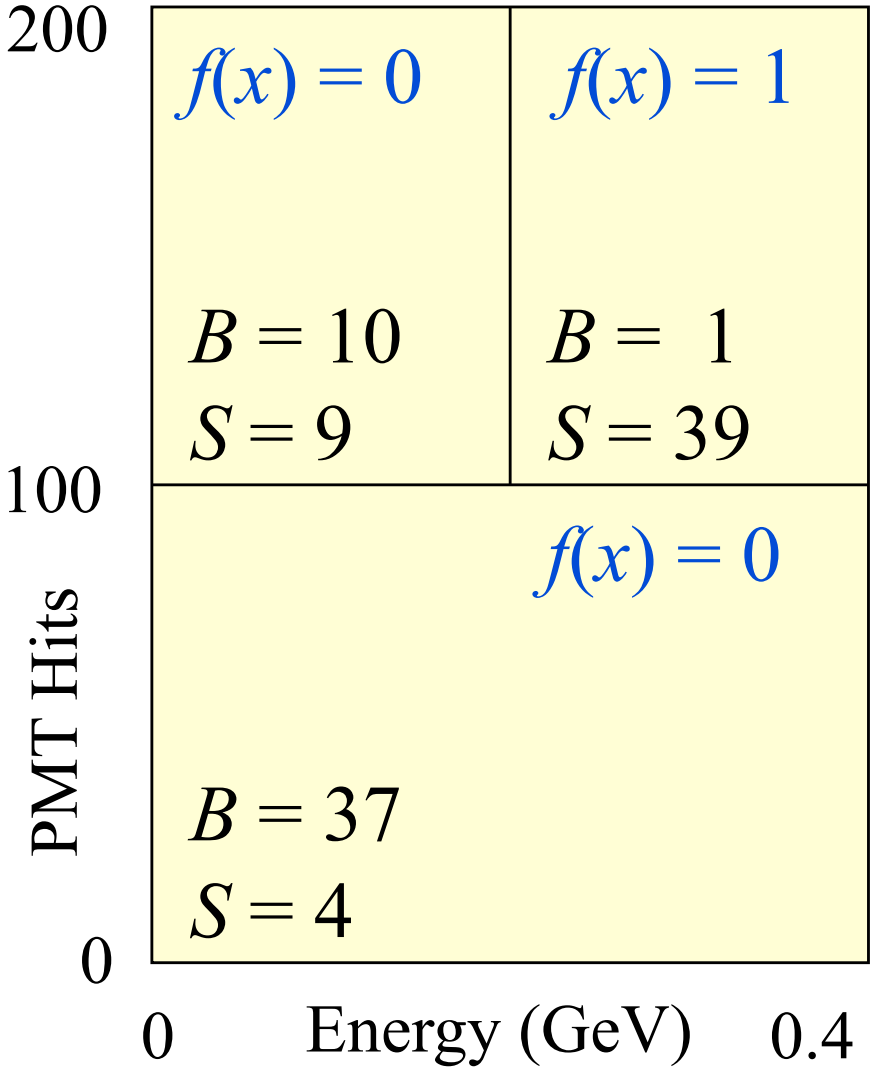
Impurity = $p\,(1 - p)$

where

$p = S / (S + B)$

$p = 0$ or 1 = maximal purity

$p = 0.5$ = maximal impurity



Ensemble Methods

Ensemble Methods

Suppose that you have a collection of discriminants $f(x, w_k)$, which, individually, perform only marginally better than random guessing.

It is possible to build from such discriminants (weak learners) highly effective ones by averaging over them:

$$f(x) = a_0 + \sum_{k=1}^{K} a_k f(x, w_k)$$

Jeromme Friedman & Bogdan Popescu (2008)

Ensemble Methods

The most popular methods (used mostly with decision trees) are:

- **Bagging**: each tree trained on a **bootstrap sample** drawn from training set
- **Random Forest**: bagging with **randomized** trees
- **Boosting**: each tree trained on a **different weighting** of full training set

Adaptive Boosting

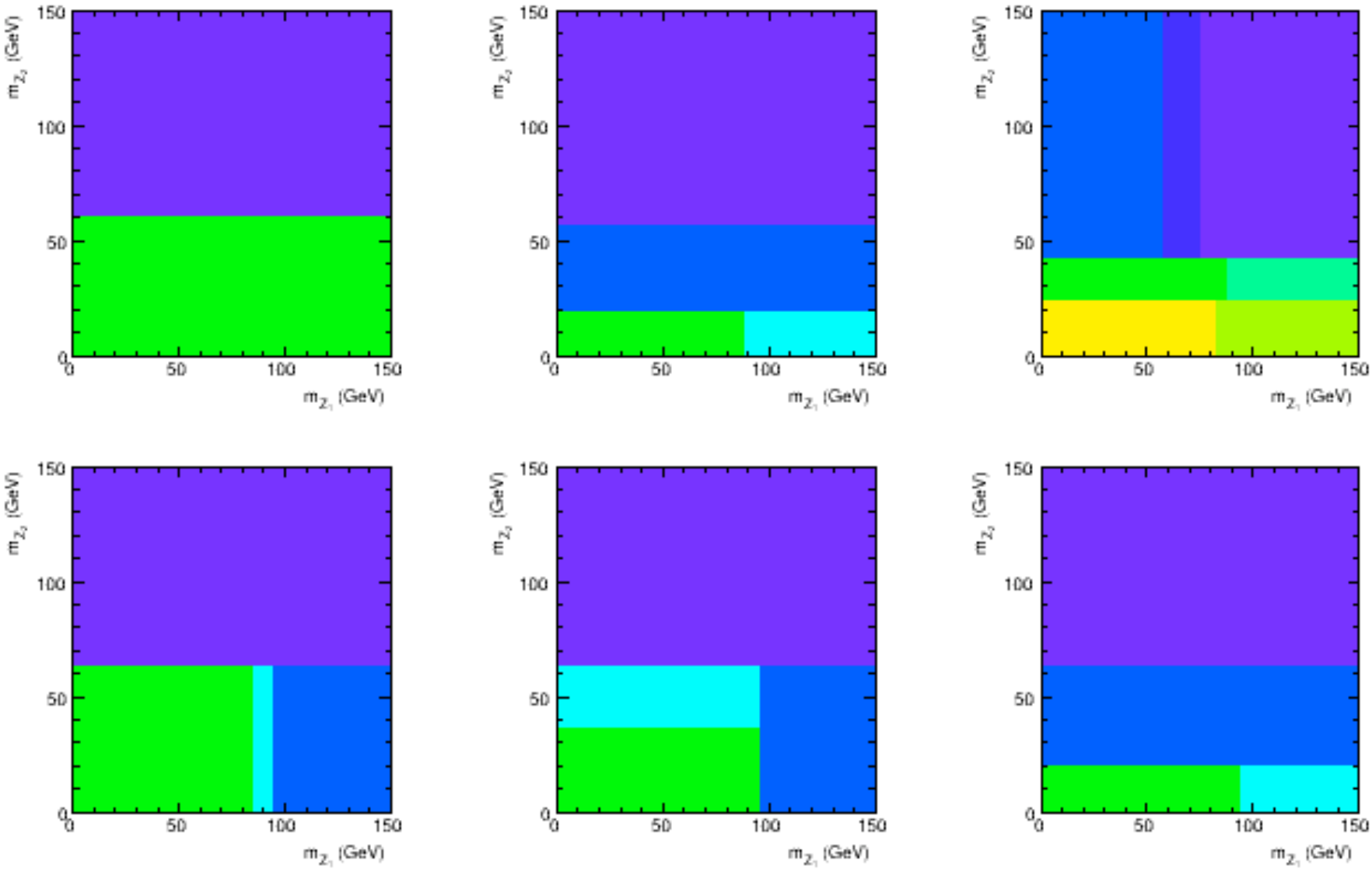
Repeat *K* times:

1. Create a decision tree $f(x, \boldsymbol{w})$
2. Compute its error rate ε on the ***weighted*** training set
3. Compute $\alpha = \ln(1-\varepsilon) / \varepsilon$
4. Modify training set: ***increase weight*** of ***incorrectly classified examples*** relative to the weights of those that are correctly classified

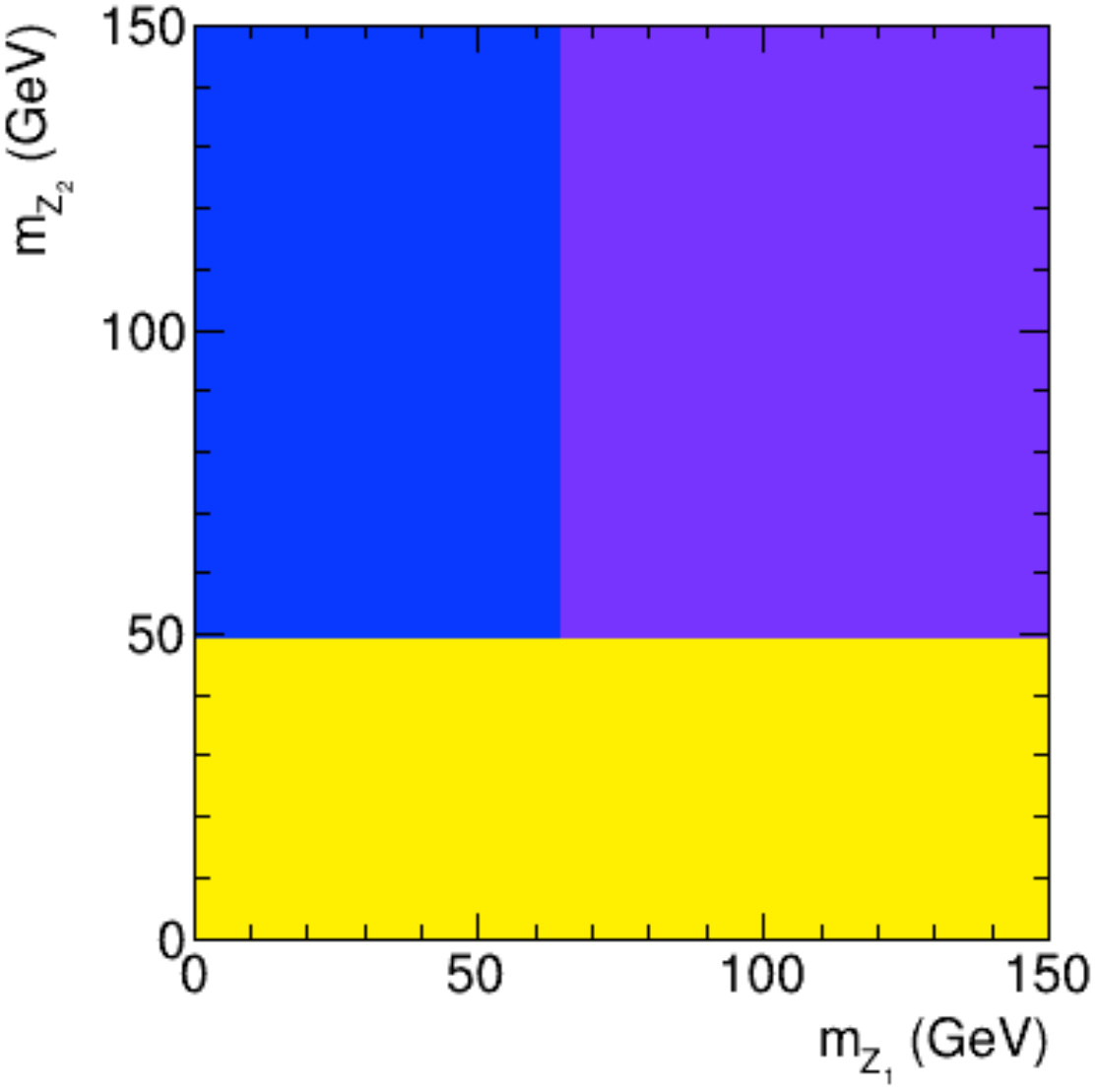
Then compute weighted average $f(x) = \sum \alpha_k \ f(x, w_k)$

Y. Freund and R.E. Schapire.
Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

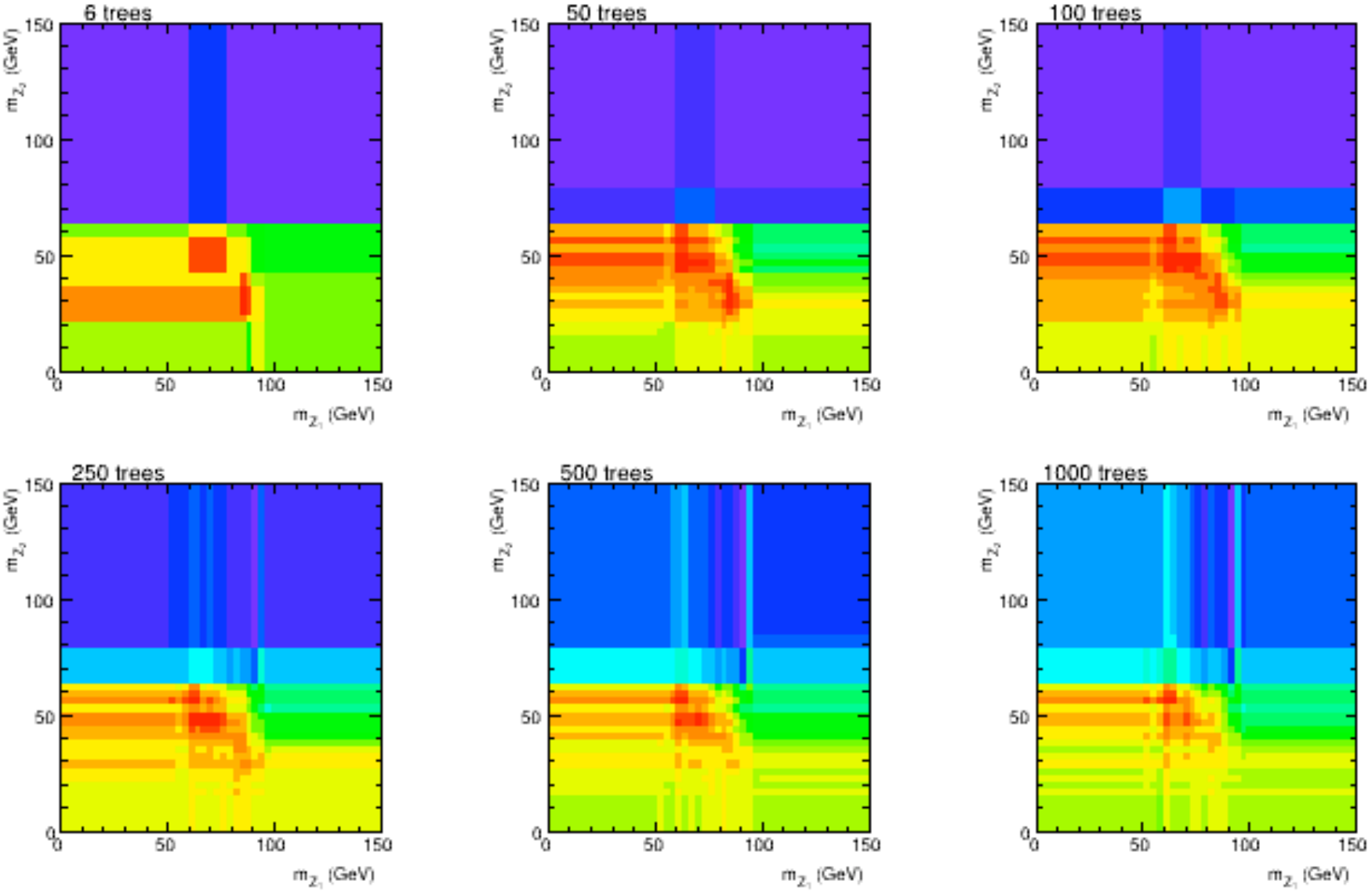
First 6 Decision Trees



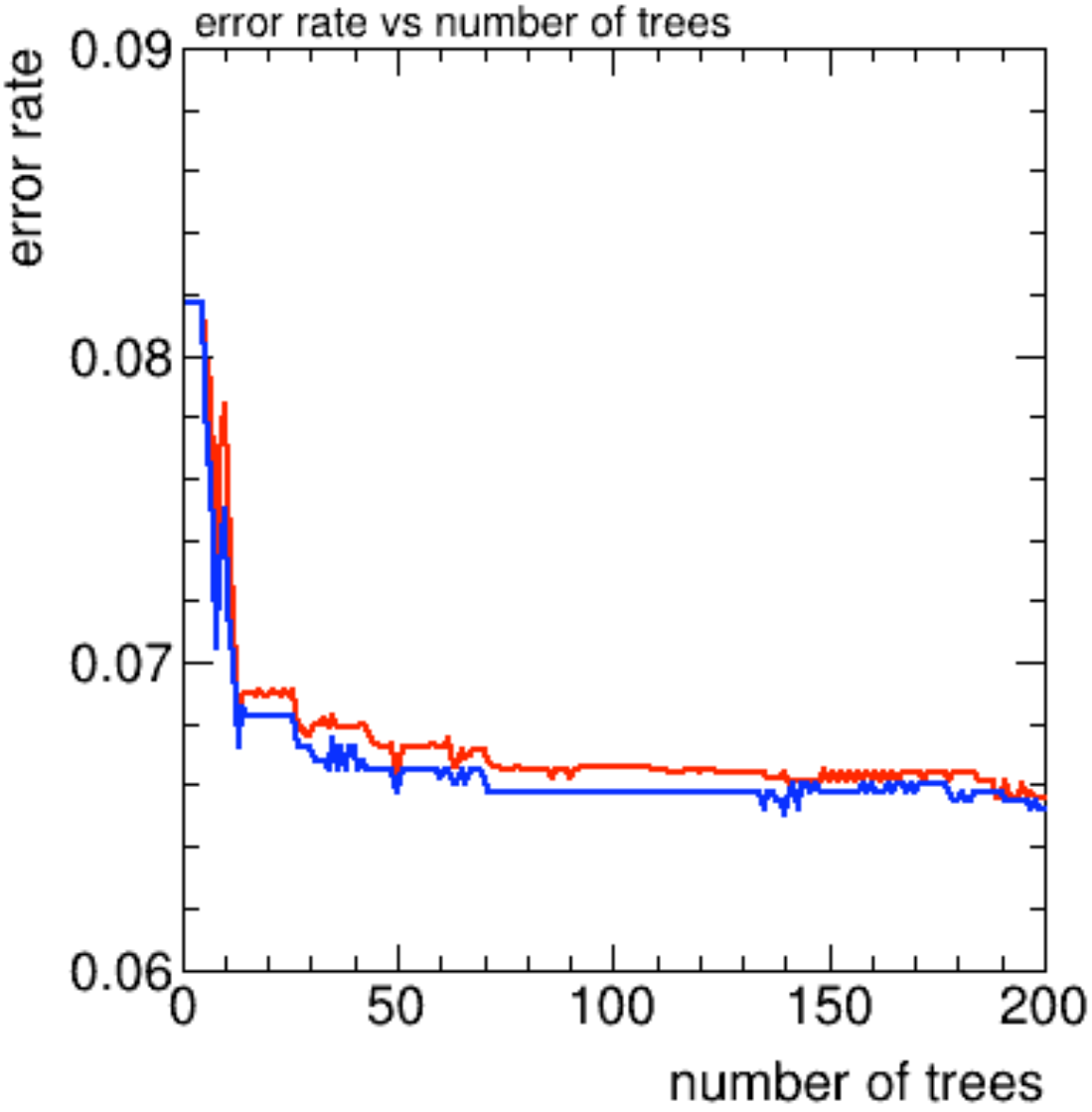
First 100 Decision Trees

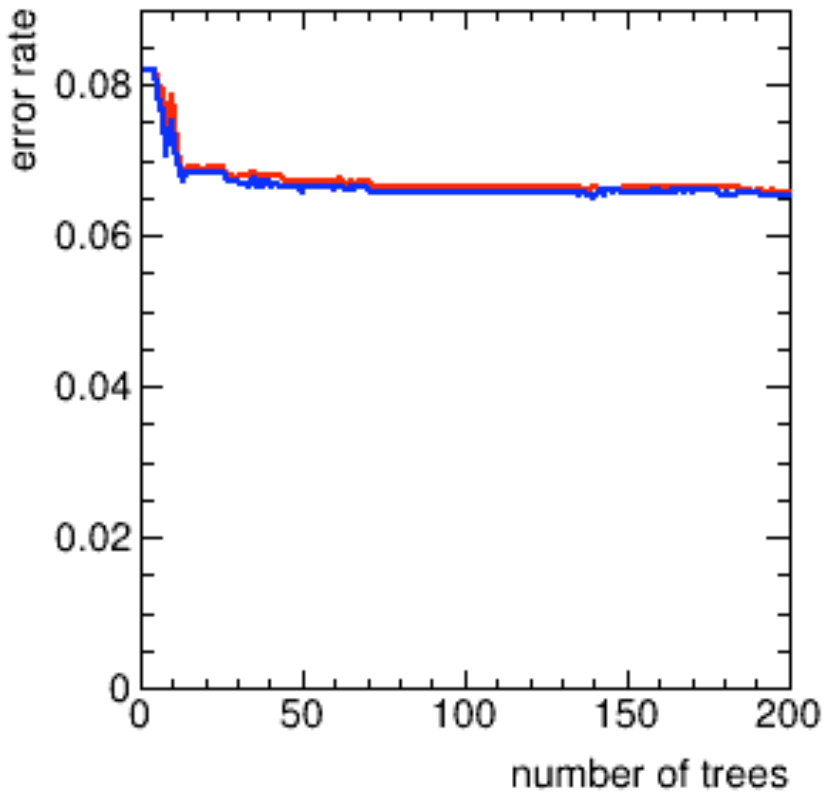


Averaging over a Forest



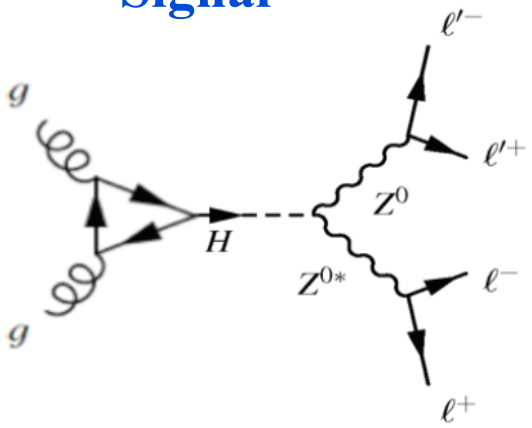
Error Rate vs Number of Trees



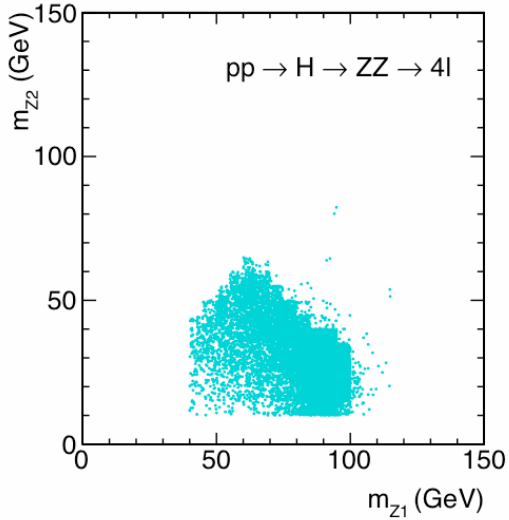


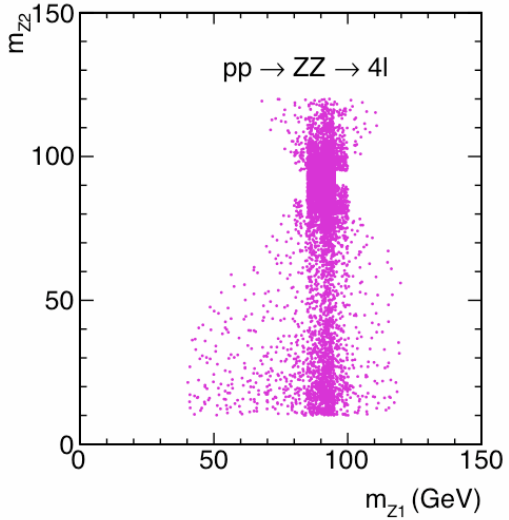
Example – H to ZZ to 4Leptons

Signal



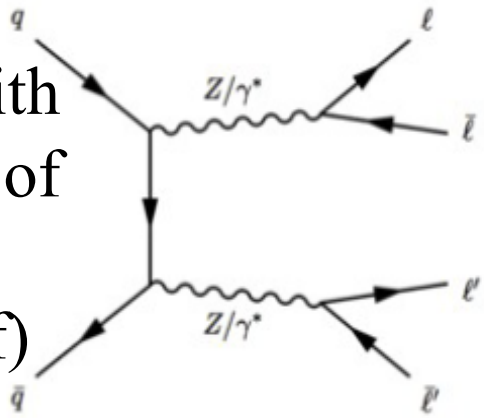
$pp \to H \to ZZ \to \ell^+\ell^-\ell'^+\ell'^-$



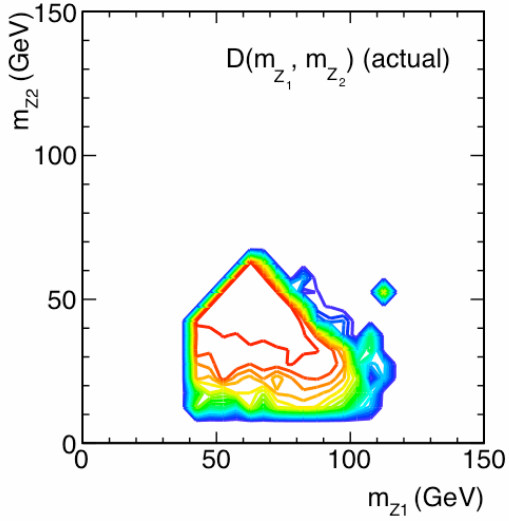


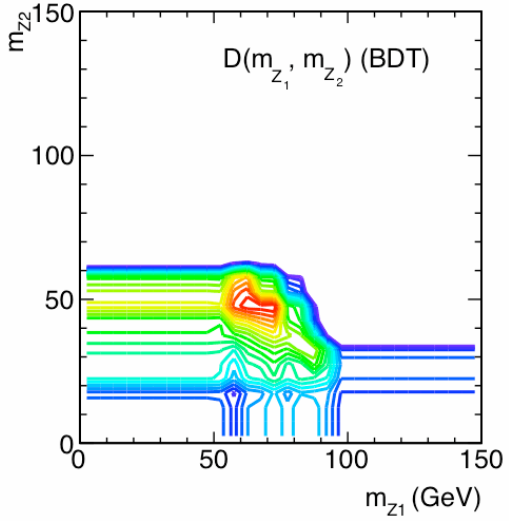
Background

200 trees with a minimum of 100 counts per bin (leaf)



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





Bayesian Neural Networks

Bayesian Neural Networks

Given

$$p(w \mid T) = \mathrm{p}(T \mid w)\ \mathrm{p}(w) / \mathrm{p}(T)$$

over the parameter space of the functions

$$\mathrm{n}(\boldsymbol{x}, w) = 1 / [1+\exp(-f(\boldsymbol{x}, w))]$$

one can estimate $\mathrm{p}(s \mid \boldsymbol{x})$ as follows

$$\mathrm{p}(s \mid \boldsymbol{x}) \sim \mathrm{n}(\boldsymbol{x}) = \int \mathrm{n}(\boldsymbol{x}, w)\ \mathrm{p}(w \mid T)\ \mathrm{d}w$$

$\mathrm{n}(\boldsymbol{x})$ is called a **Bayesian Neural Network** (BNN)

Bayesian Neural Networks

Generate Sample

N points $\{w\}$ from $p(w \mid T)$ using a Markov chain Monte Carlo (MCMC) technique (see talk by Glen Cowan) and average over the last M points

$$n(\boldsymbol{x}) = \int n(\boldsymbol{x}, w)\, p(w \mid T)\, dw$$

$$\sim \sum n(\boldsymbol{x}, w_i) / M$$

Example – H to ZZ to 4Leptons

Dots

$p(s \mid x) = H_s / (H_s+H_b)$

H_s, H_b, 1-D histograms

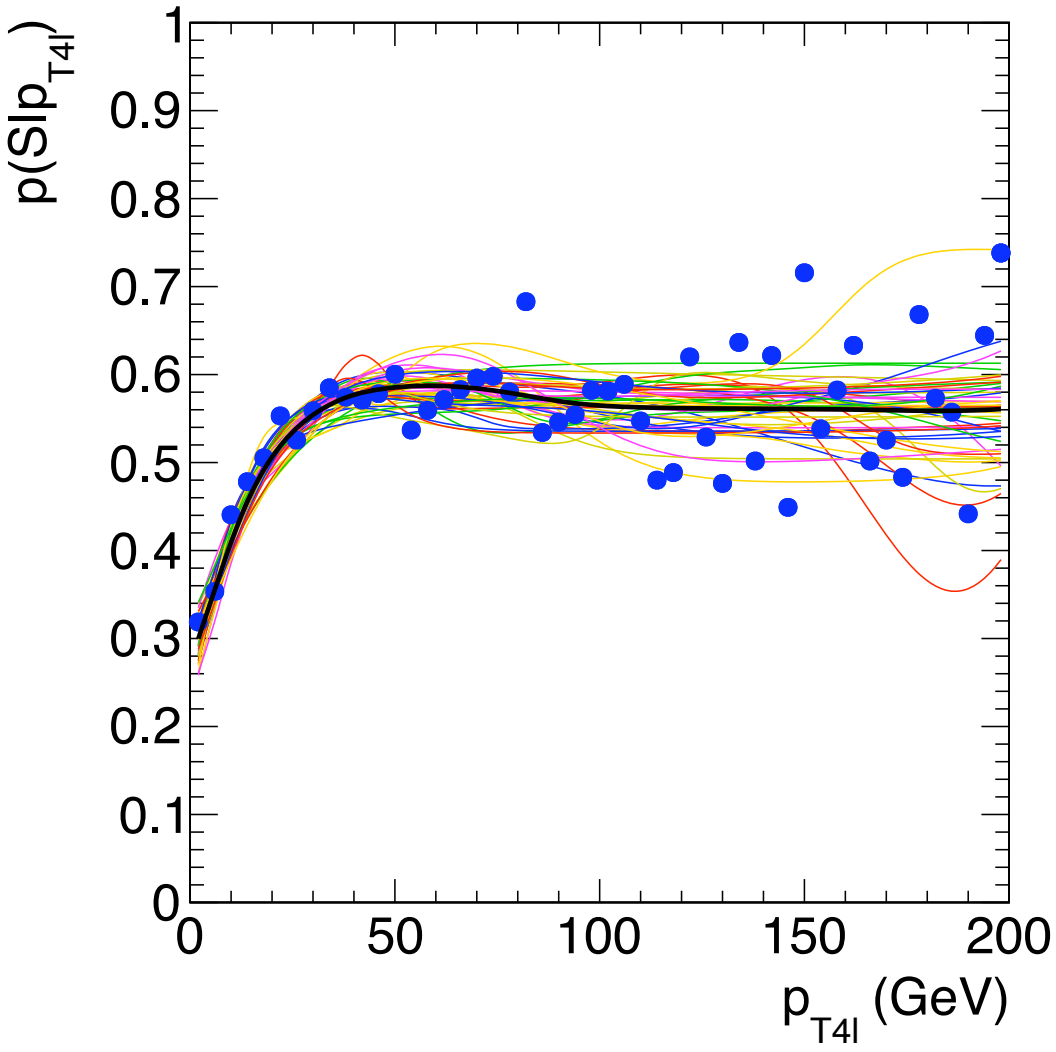
Curves

Individual NNs

$n(x, w_k)$

Black curve

$n(x) = < n(x, w) >$



Modeling Response Functions

Consider the observed jet p_{T} spectrum:

$$f_{\mathrm{obs}}(p_T \mid \nu,\omega) = \int R(p_T \mid z,\nu,\omega)\, f(z \mid \nu,\omega)\, dz$$

In order to avoid unfolding, we need to publish the response function,

$$R(p_T \mid z,\nu,\omega)$$

and the prior

$$\pi(\nu,\omega)$$

The latter can be supplied as a sample of points (ν, ω) over which one would average. But what of the response function?

Modeling Response Functions

The response function can be written as

$$R(p_T \mid z, \nu, \omega) = p(p_T, z, \nu, \omega) \,/\, p(z, \nu, \omega)$$

Therefore, in principle, it could be modeled as follows:

1. Build a discriminant D between the original events and the events in which the jet p_{T} is replaced by a value sampled from a known distribution $u(p_{\mathrm{T}})$
2. Approximate R using

$$R(p_T \mid z, \nu, \omega) \simeq u(p_T) D \,/\, (1 - D)$$

Summary

- Multivariate methods can be applied to many aspects of data analysis.
- Many practical methods, and convenient tools such as TMVA, are available for regression and classification.
- All methods approximate the same mathematical entities, but no one method is guaranteed to be the best in all circumstances. So, just as is true of statistical methods, it is good practice to experiment with a few of them!

MVA Tutorials

1. Download (the 25M!) file **tutorials.tar.gz** from

   http://www.hep.fsu.edu/~harry/INFNSOS2013

2. Unpack

   tar zxvf tutorials.tar.gz

3. Check

   cd tutorials-cowan
   python expFit.py