

# Multivariate Methods An Overview

Harrison B. Prosper  
Florida State University

11 February, 2008  
Caltech MVA Workshop

# Outline

- Introduction
- Multivariate Methods
- A Few Issues
- Summary

# Introduction

Multivariate methods can be useful in:

- Probability density estimation
- Signal/background discrimination
- Detecting regions of interest
- Function approximation
- Data compression
- Variable selection
- Optimization
- Model comparison

# Introduction

**Example:** Multivariate methods can be used, in principle, to construct the likelihood function for a dataset comprising  $N$  events

$$p(x | \theta, \varphi) \propto \exp(-d(\theta, \varphi)) \prod_{i=1}^N d(x_i, \theta, \varphi)$$

where  $d(\theta, \varphi) = \int d(x, \theta, \varphi) dx$

and  $d(x, \theta, \varphi) = \alpha s(x | \theta, \varphi_s) + \beta b(x | \varphi_b)$

$$\int s(x | \theta, \varphi_s) dx = 1, \quad \int b(x | \varphi_b) dx = 1$$

# Introduction

In practice, it is usually easier to reduce the dimensionality of the data by approximating the function

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

and build a likelihood function using the 1-D signal and background distributions of  $p(s|x)$ .

Finally, one derives an estimator  $y = f(x)$  for the quantity of interest

# Introduction

It could be argued, however, that since the estimator

$$y = f(x)$$

is the ultimate goal, why not approximate it directly?

One way to approximate  $f(x)$  is to evolve an ensemble of functions  $\{ f(x) \}$ , using some genetic algorithm, until one arrives at suitable descendents.

# Introduction

## A Short List of Multivariate Methods

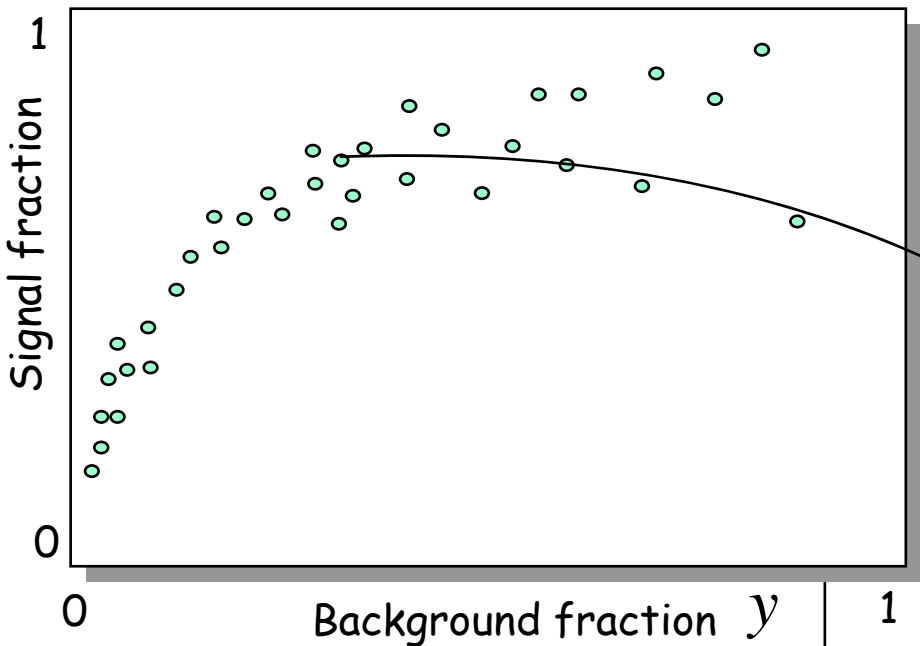
- Random Grid Search
- Quadratic & Linear Discriminants
- Support Vector Machines
- Naïve Bayes
- Kernel Density Estimation
- Neural Networks
- Bayesian Neural Networks
- Binary Decision Trees

# Multivariate Methods

---

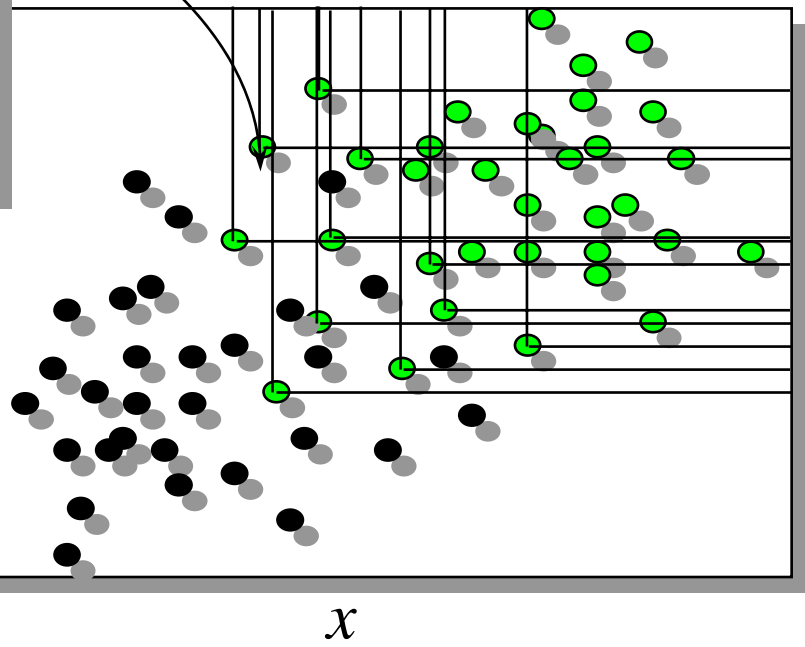
# Random Grid Search

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



$$x > x_i$$

$$y > y_i$$



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

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

# Quadratic & Linear Discriminants

---

# Quadratic Discriminants

An approximation to  $p(s|x)$  can be derived by replacing  $s(x)$  and  $b(x)$  with multivariate Gaussians in the **Bayes discriminant**:

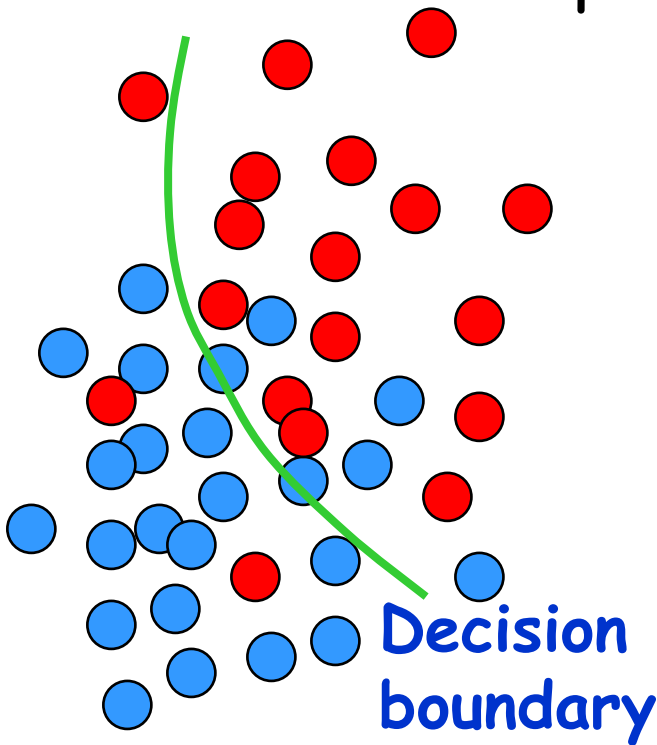
$$B(x) = \frac{p(s|x)}{p(b|x)} = \frac{s(x)p(s)}{b(x)p(b)}$$

One usually considers  $\lambda(x) = \ln B(x)$ , which, dropping non-essential constants, yields

$$\lambda(x) = (x - \mu_B)^T \Sigma_B^{-1} (x - \mu_B) - (x - \mu_S)^T \Sigma_S^{-1} (x - \mu_S)$$

# Quadratic Discriminant

A fixed value of  $\lambda(x)$  defines a quadratic surface that partitions the  $n$ -dimensional input space  $\{x\}$  into signal-rich and background-rich regions.



# Linear Discriminant

If, in the quadratic function  $\lambda(x)$ , we use the same covariance matrix for each class of events

e.g.,  $\Sigma = \Sigma_S + \Sigma_B$

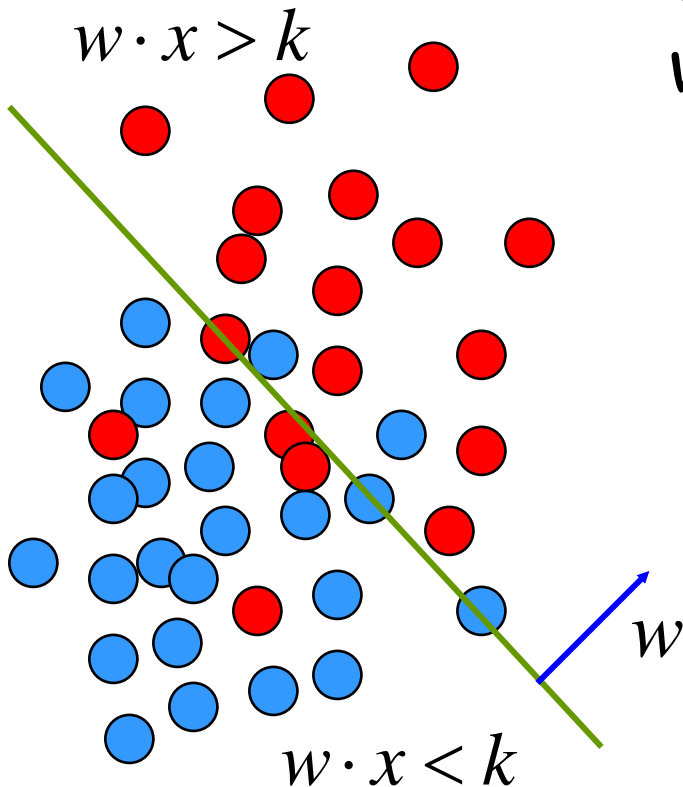
we arrive at

## Fisher's Discriminant

$$\lambda(x) \rightarrow w \cdot x$$

where  $w$  is a vector given by

$$w \propto \Sigma^{-1} (\mu_S - \mu_B)$$



# Naïve Bayes

---

# Naïve Bayes

This method is very simple: each density  $p(x)$  is approximated by

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

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  $p(s|x)$  is given by

$$\hat{p}(s | x) = \frac{\hat{s}(x) p(s)}{\hat{s}(x) p(s) + \hat{b}(x) p(b)}$$

In spite of its name, this method can sometimes provide surprisingly good results. The reason is that  $p(s|x)$  is usually a much gentler function than either of the densities  $s(x)$  or  $b(x)$ .

# Support Vector Machines

---

# Support Vector Machines

Generalization of the Fisher discriminant  
(Boser, Guyon and Vapnik, 1992).

## Basic Idea

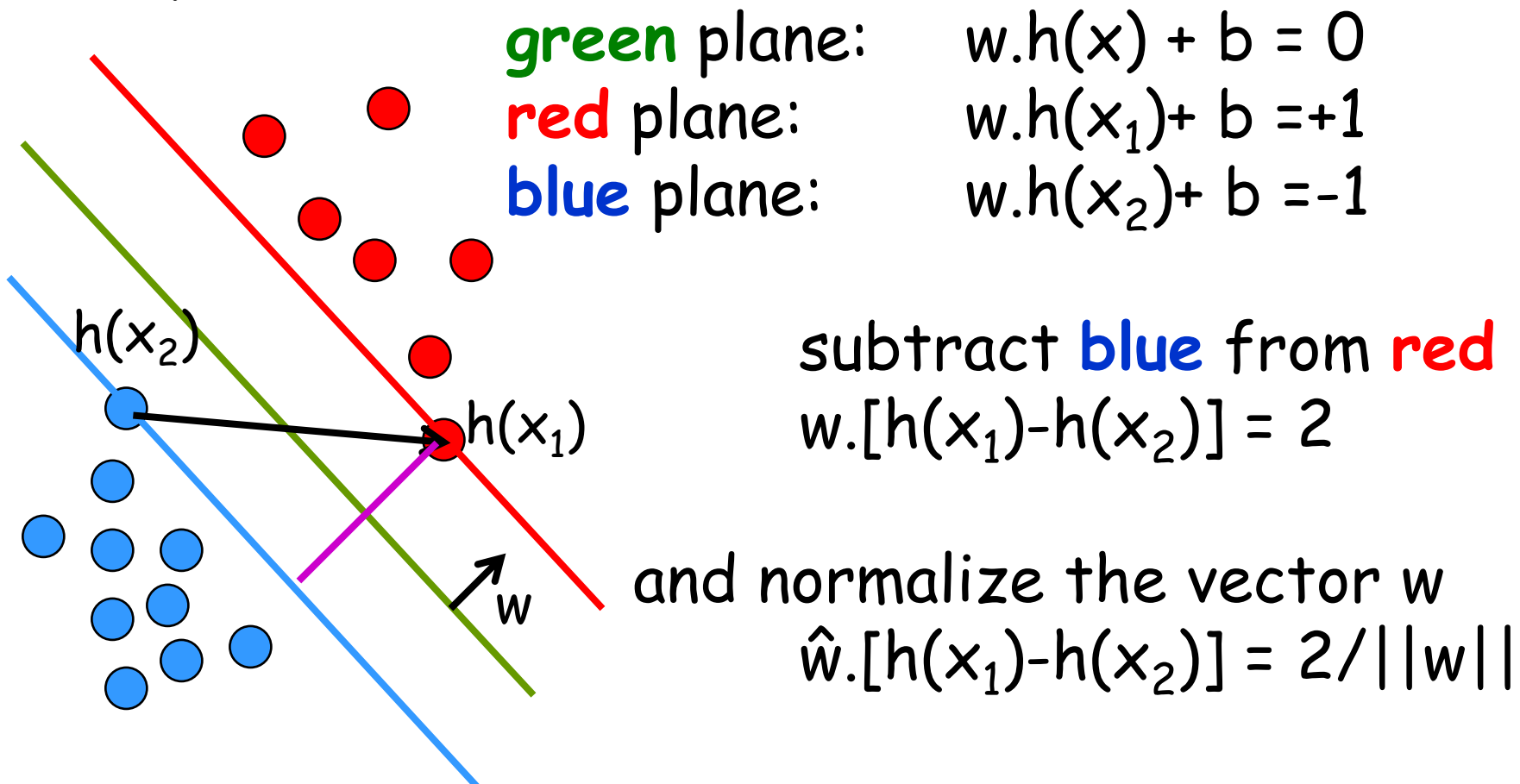
Data that are **non-separable** in  $n$ -  
dimensions may be better separated if  
mapped into a space of higher dimension

$$h : \mathcal{R}^n \rightarrow \mathcal{R}^{\text{Huge}}$$

Use a hyper-plane to partition the high  
dimensional space  $f(x) = w \cdot h(x) + b$

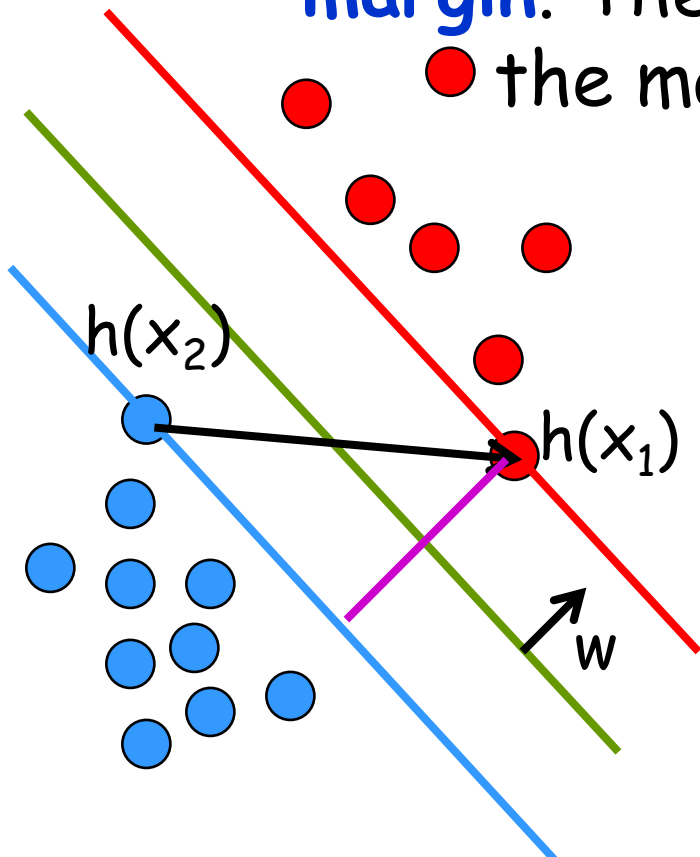
# Support Vector Machines

Consider **separable** data in the high dimensional space



# Support Vector Machines

The quantity  $m = \hat{w} \cdot [h(x_1) - h(x_2)]$ , 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

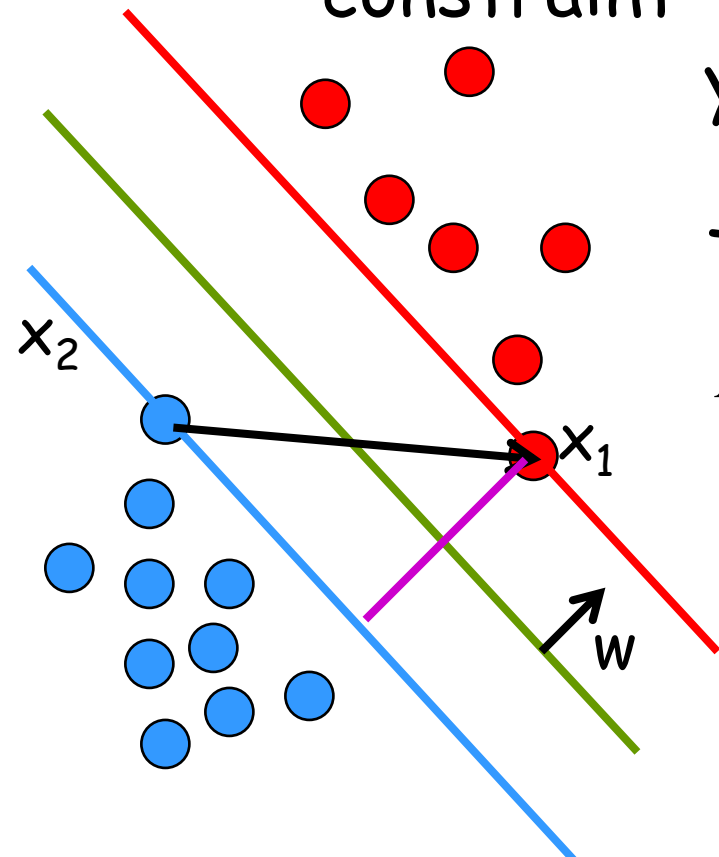
$$y_i (w \cdot h(x_i) + b) \geq 1, \quad i = 1 \dots N$$

That is the task is to minimize

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2$$

$$- \sum_{i=1}^N \alpha_i [y_i (w \cdot h(x_i) + b) - 1]$$

where the  $\alpha > 0$  are Lagrange  
multipliers



# Support Vector Machines

When  $L(w, b, \alpha)$  is minimized with respect to  $w$  and  $b$ , the Lagrangian  $L(w, b, \alpha)$  can be transformed to its dual form

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

At the minimum, the only non-zero coefficients  $\alpha$  are those corresponding to points *on the red and blue planes*: that is, the **support vectors**.

**Key idea**: replace scalar product  $h(x_i) \cdot h(x_j)$  by a kernel function  $k(x_i, x_j)$ .

# Kernel Density Estimation

---

# Kernel Density Estimation

## Basic Idea

### Parzen Estimation (1960s)

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

### Mixtures

$$p(x) = \sum_j \varphi(x, j) q(j) \quad j \ll N$$

# Kernel Density Estimation

Why does it work? In the limit  $N \rightarrow \infty$

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

the true density  $p(x)$  will be recovered  
provided that

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

# Neural Networks

---

# Neural Networks

Given

$$D = \mathbf{x}, \mathbf{y}$$

$$\mathbf{x} = \{x_1, \dots, x_N\}, \quad \mathbf{y} = \{y_1, \dots, y_N\}$$

of  $N$  training examples

Find

a function  $n(\mathbf{x}, \mathbf{w})$ , with parameters  $\mathbf{w}$ , by maximizing a likelihood function

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w})$$

# Neural Networks

A typical likelihood (for classification) is

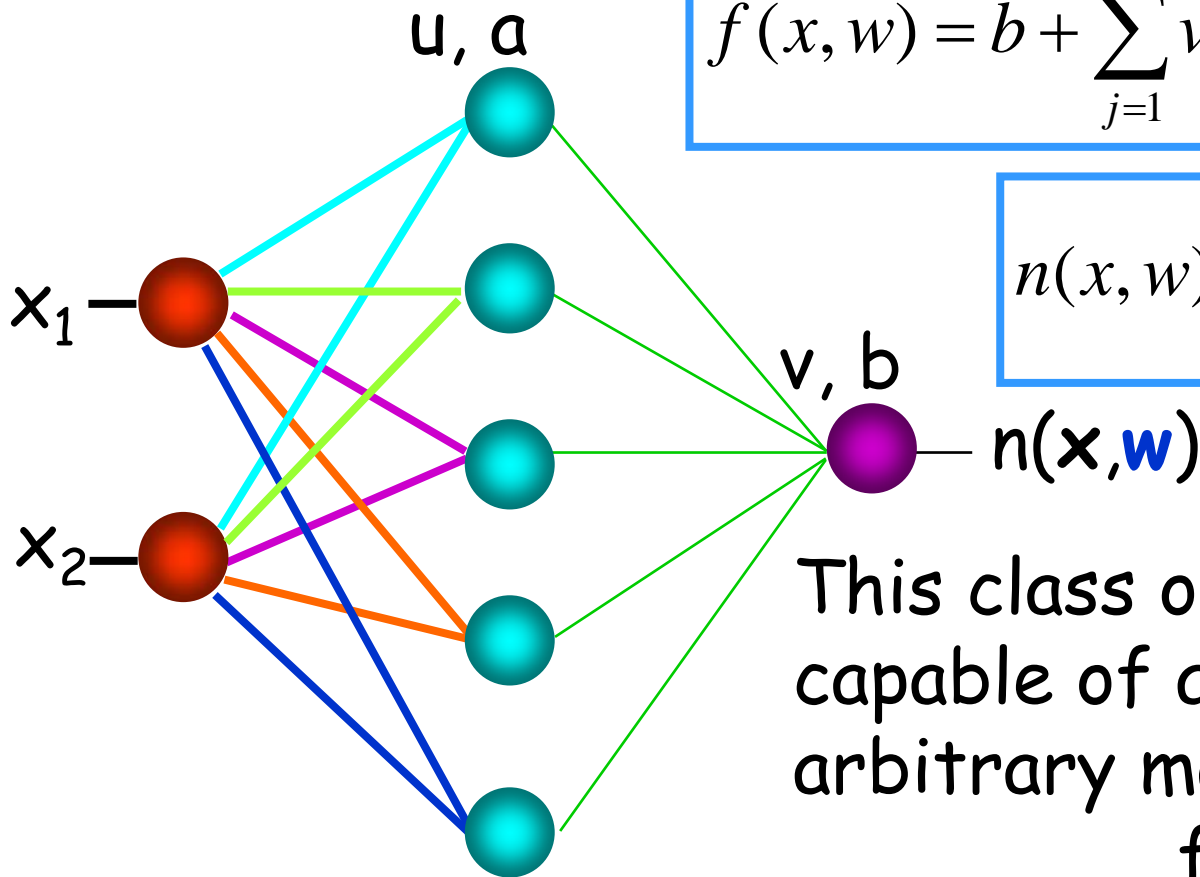
$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_i n(\mathbf{x}_i, \mathbf{w})^y [1 - n(\mathbf{x}_i, \mathbf{w})]^{1-y}$$

where  $y = 0$  for background events  
 $y = 1$  for signal events

**If**  $n(\mathbf{x}, \mathbf{w})$  is sufficiently flexible, then the maximum likelihood solution for  $\mathbf{w}$  yields

$n(\mathbf{x}, \mathbf{w}) \rightarrow p(s|\mathbf{x})$ , *asymptotically*.

# Neural Networks



$$f(x, w) = b + \sum_{j=1}^H v_j \tanh \left[ a_j + \sum_{i=1}^P u_{ij} x_i \right]$$

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

This class of functions is capable of approximating arbitrary mappings  
 $f: \mathbb{R}^P \rightarrow \mathbb{R}$

# Bayesian Neural Networks

Given

$$p(\mathbf{w}|\mathbf{D}) = p(\mathbf{D}|\mathbf{w}) p(\mathbf{w}) / p(\mathbf{D})$$

over the parameter space of the functions

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

estimate  $p(s|\mathbf{x})$  as follows

$$p(s|\mathbf{x}) \sim n(\mathbf{x}) = \int n(\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathbf{D}) d\mathbf{w}$$

$n(\mathbf{x})$  is a **Bayesian Neural Network (BNN)**

# BNN - 1-D Example

Signal

$p+p\bar{p} \rightarrow t q b$

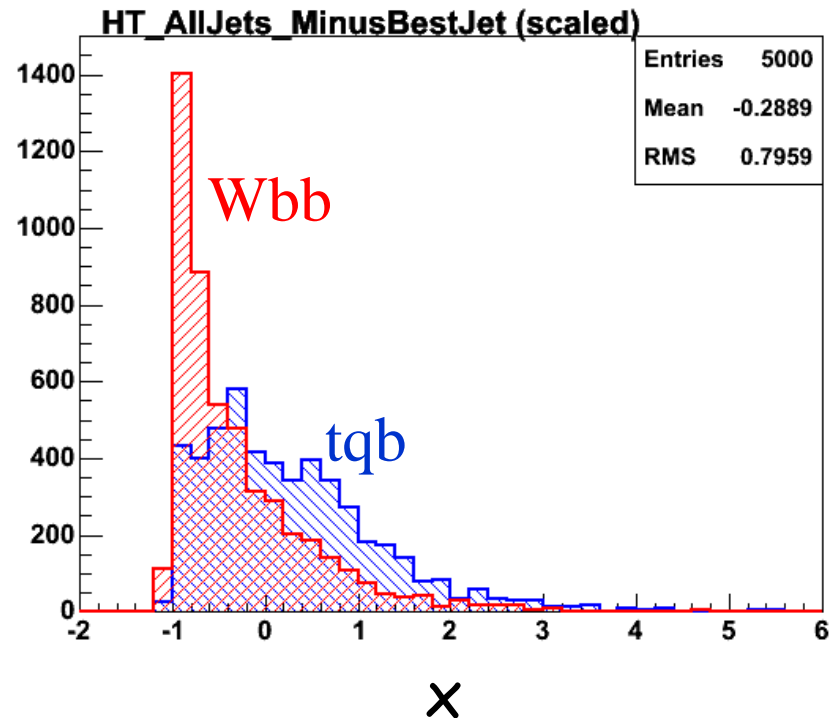
Background

$p+p\bar{p} \rightarrow W b b$

Function Class

$(1, 15, 1)$

Sample  $p(w|D)$  using MCMC



# BNN - 1-D Example

## Dots

$$p(S|x) = H_S / (H_S + H_B)$$

$H_S, H_B$ , 1-D histograms

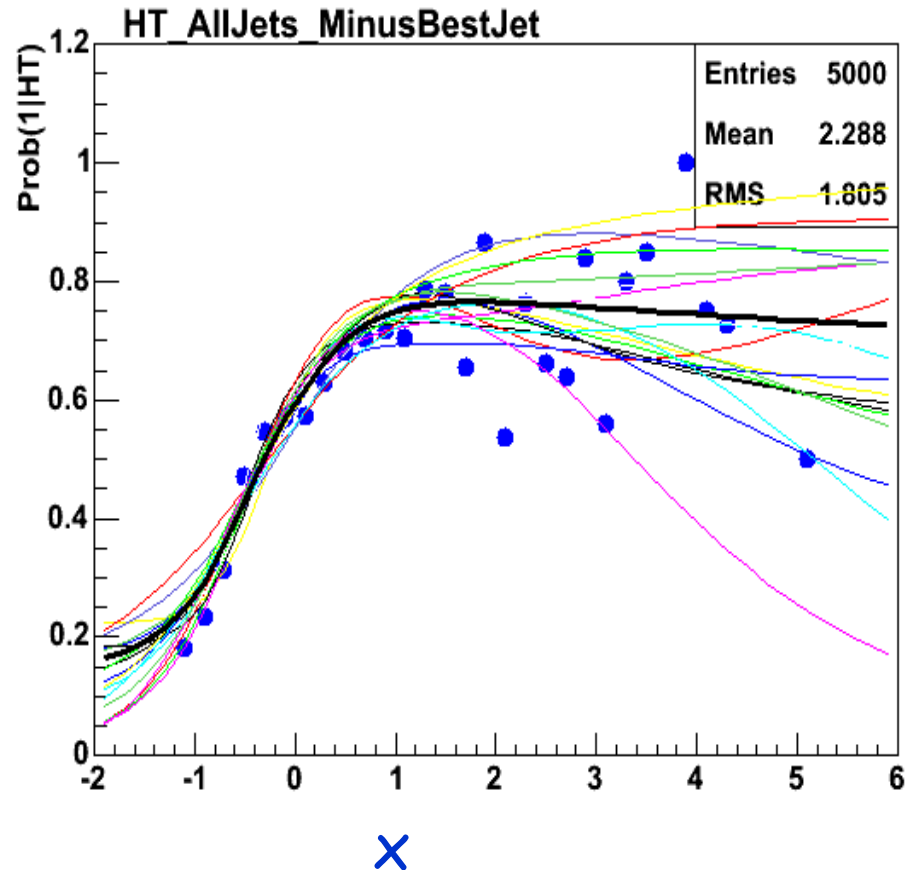
## Curves

Individual NNs

$$n(x, \mathbf{w}_k)$$

## Black curve

$$n(x) = (1/N) \sum_k n(x, \mathbf{w}_k)$$



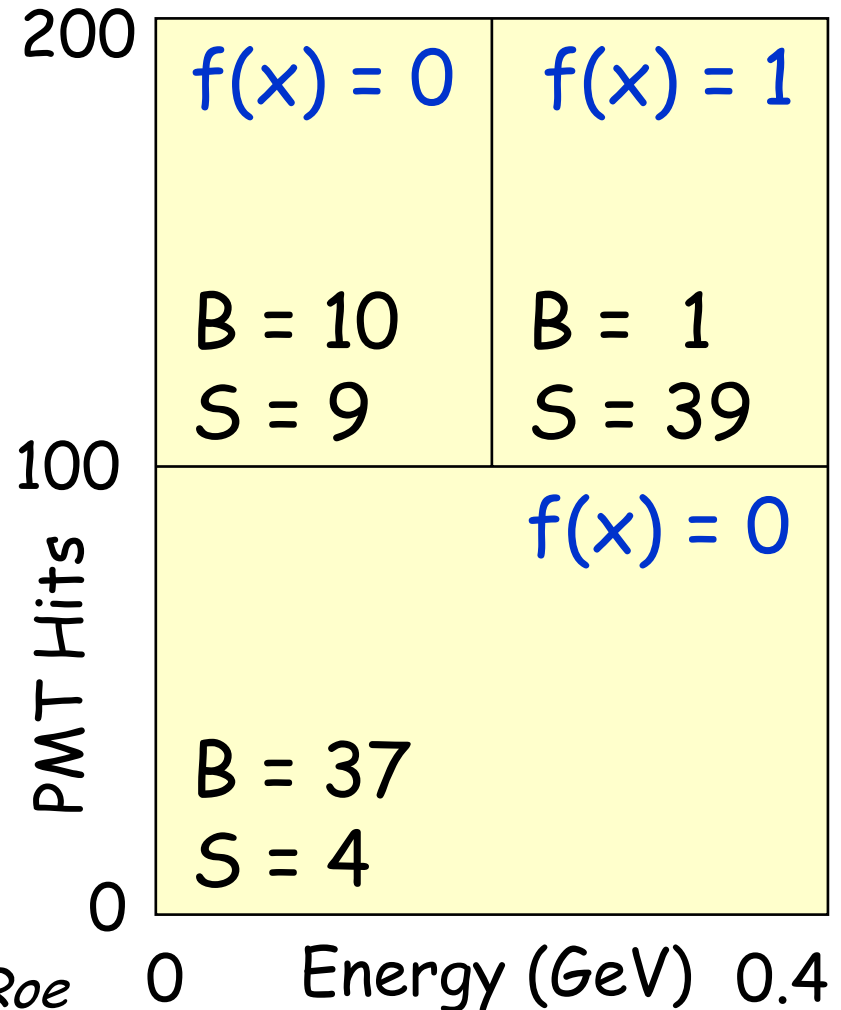
# Decision Trees

---

# Decision Trees

A decision tree is an **n-dimensional histogram** whose bins are constructed recursively

Each bin is associated with a value of the desired function  **$f(x)$**



*MiniBoone, Byron Roe*

# Decision Trees

For each variable find the **best** cut:

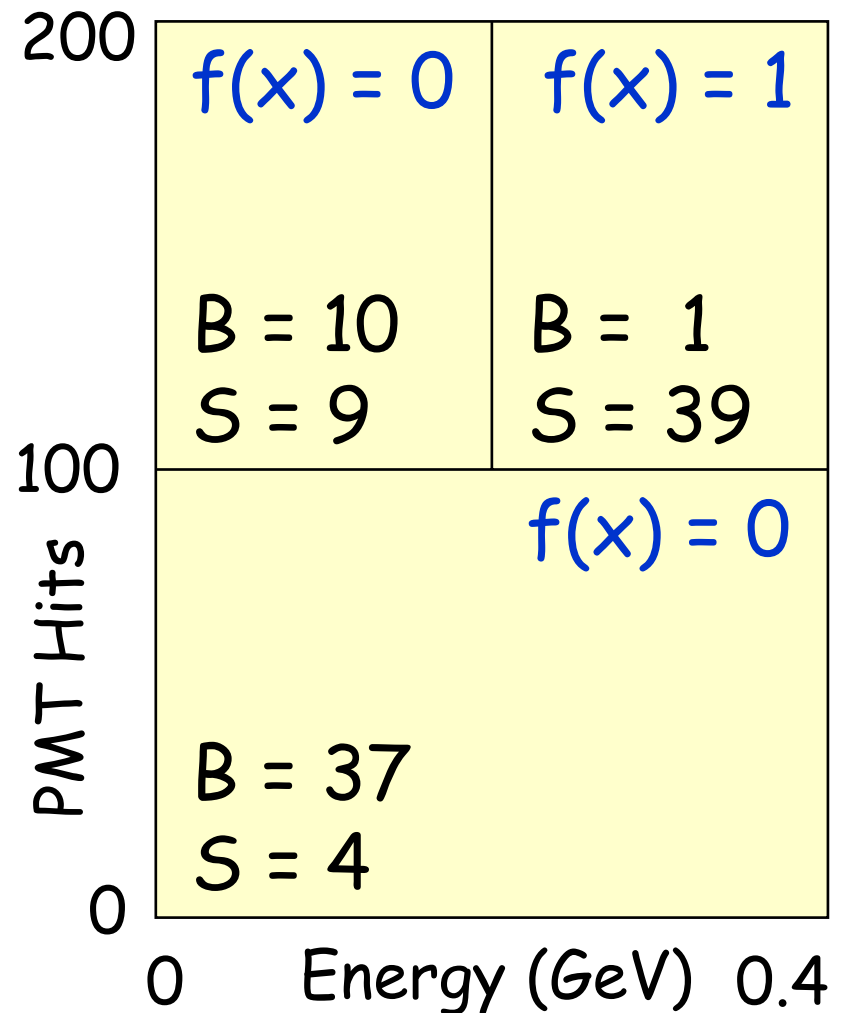
Decrease in impurity

= **Impurity**(parent)

- **Impurity**(leftChild)

- **Impurity**(rightChild)

and partition using the **best** of the best



# Decision Trees

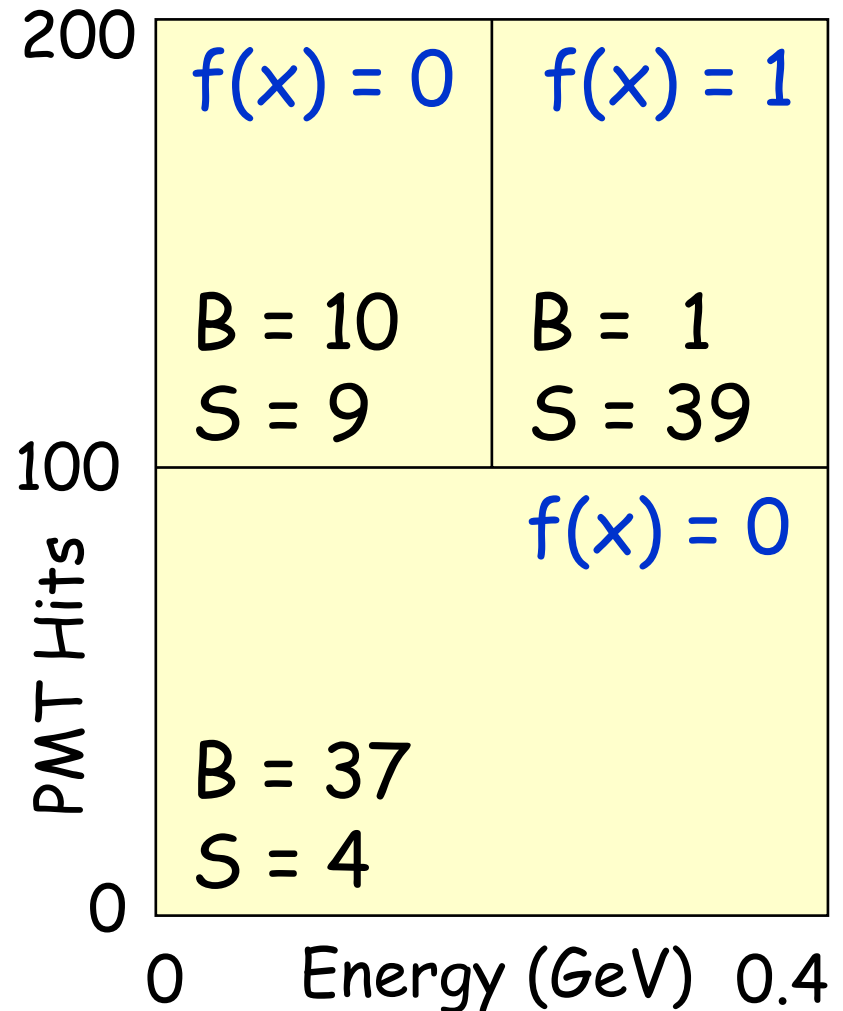
A common impurity measure (Gini):

$$\text{Impurity} = N * p*(1-p)$$

where

$$p = S / (S+B)$$

$$N = S + B$$



# Issues

- How should one choose the function class (number of leaves, number of trees, number of hidden nodes etc.)?
- How does one verify that an  $n$ -density is well-modeled?
- How should one protect against model uncertainty when compressing data?

# Summary

- Multivariate methods can be applied to many aspects of data analysis
- Many practical methods are available to compress data to 1-dimension with little information loss
- No one method is guaranteed to be the best in all circumstances, so experiment with a few of them