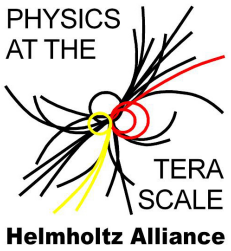


Introduction to Machine Learning

Harrison B. Prosper

Florida State University, USA

Terascale Statistics School 2017

DESY, Hamburg, Germany

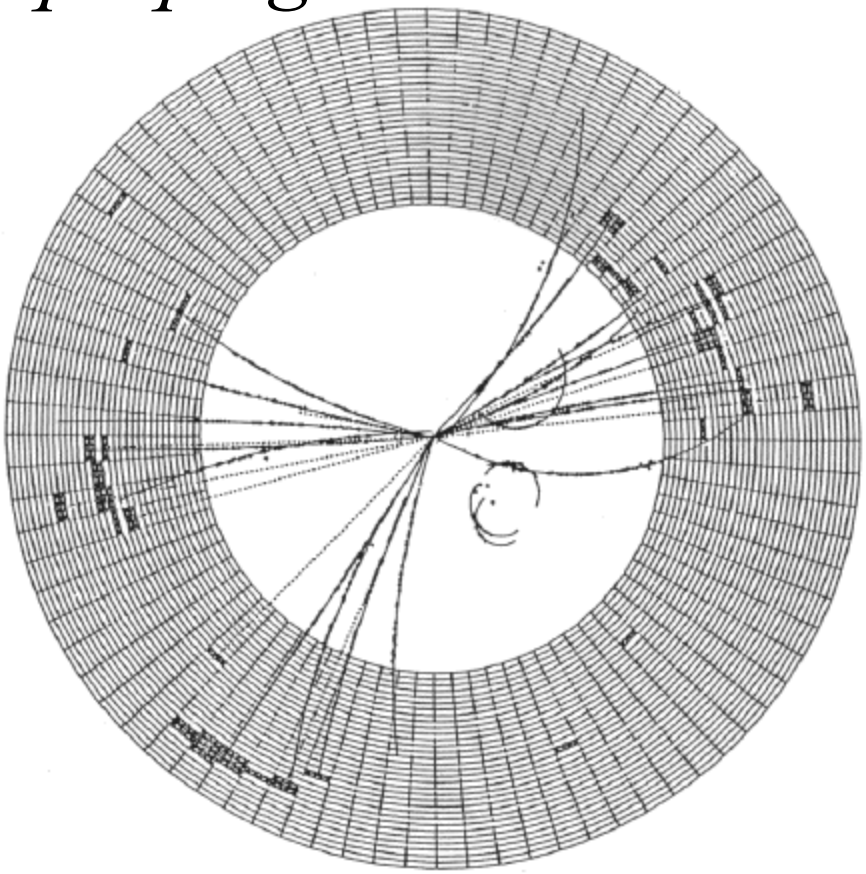
6 – 13 March, 2017

My Connection with DESY

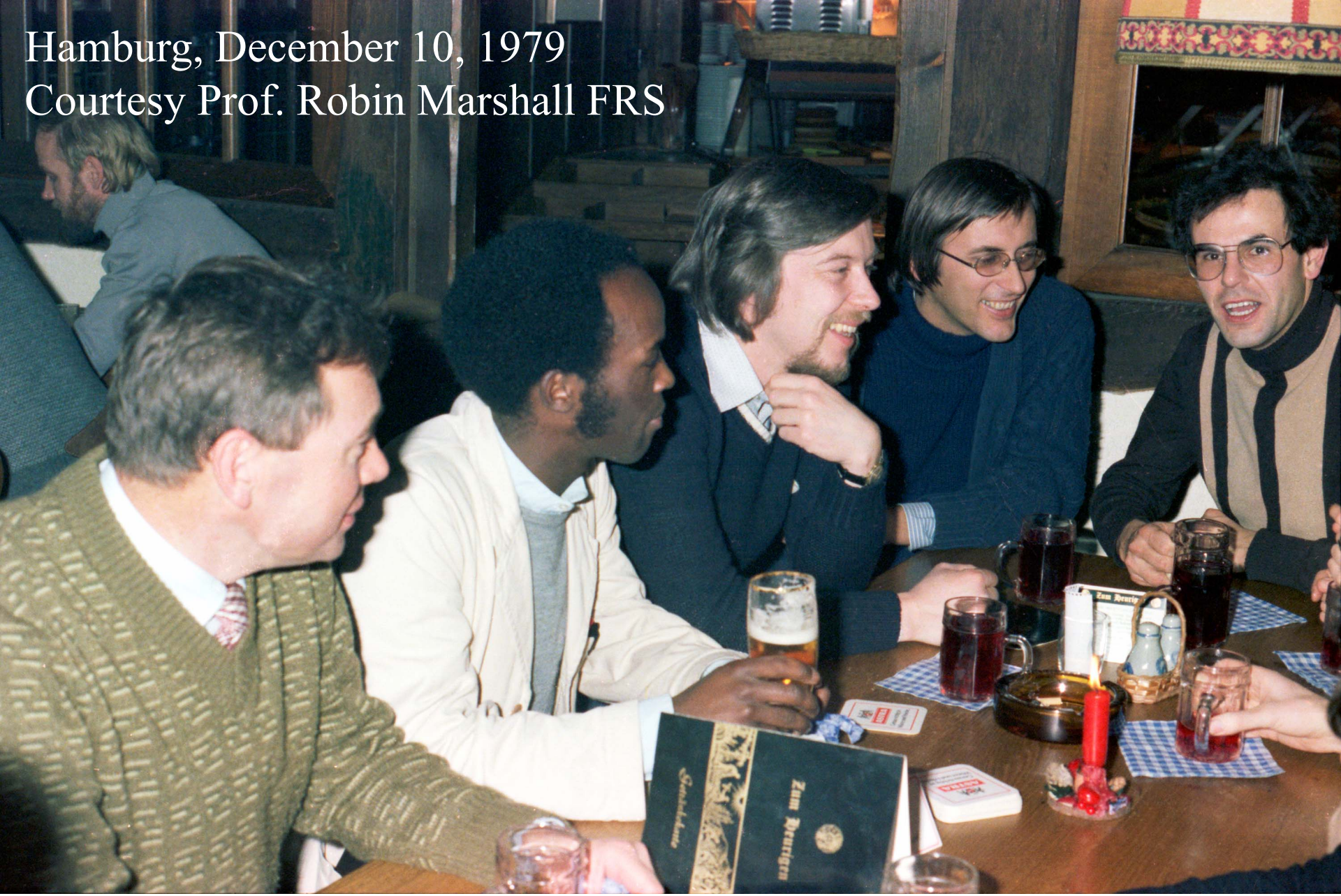
$$e^+ + e^- \rightarrow q + \bar{q} + g$$

DESY
Hamburg,
Germany

1979
TASSO
MARK-J
JADE
PLUTO



*** SUMS (GEV) *** PTOT 35.768 PTRANS 29.964 PLONG 15.700 CHARGE -2
TOTAL CLUSTER ENERGY 15.169 PHOTON ENERGY 4.893 NR OF PHOTONS 11


Hamburg, December 10, 1979
Courtesy Prof. Robin Marshall FRS

Outline

- Lecture 1
  - Introduction
  - Theory
  - Practice
    - Grid Searches
    - Fisher Discriminant
    - Boosted Decision Trees
- Lecture 2
    - Neural Networks, Shallow, Deep, Bayesian

INTRODUCTION

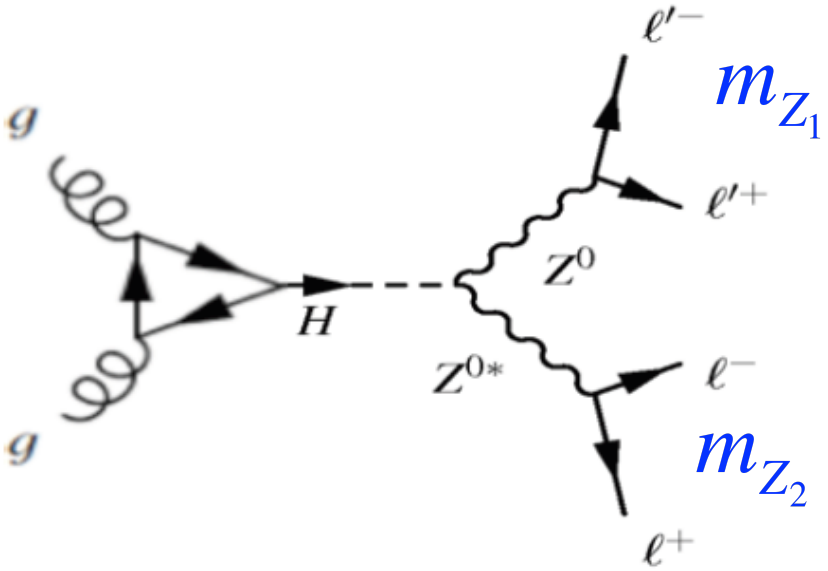
Introduction

In these lectures, I shall illustrate a few key ideas in machine learning using the following examples:

- Example 1: Higgs to ZZ to 4 leptons vs. ZZ to 4 leptons
- Example 2: Wine Tasting

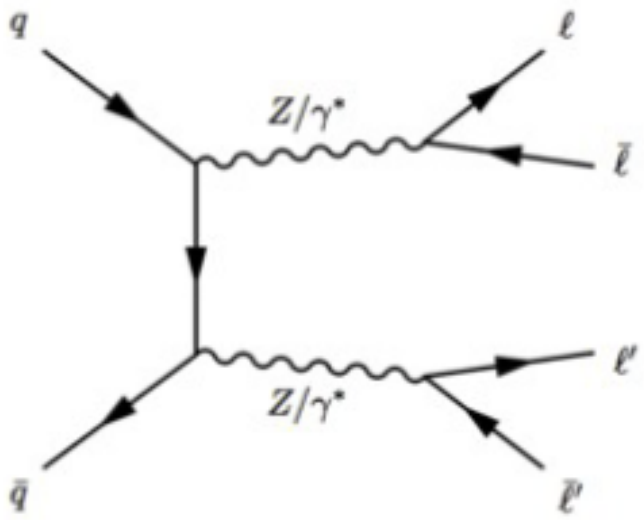
Example 1: Higgs to ZZ to 4 Leptons

Signal



$pp \rightarrow H \rightarrow ZZ \rightarrow l^+ \, l^- l'^+ l'^-$

Background

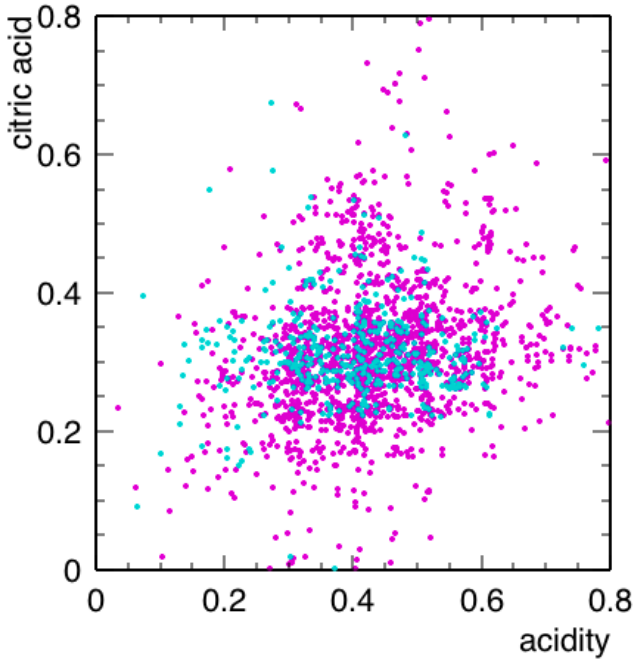


$pp \rightarrow ZZ \rightarrow l^+ \, l^- l'^+ l'^-$

Example 2: Wine Tasting

Wine tasting is big business. But, can it be automated?

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



Introduction: What is Machine Learning?

The use of computer-based algorithms for constructing useful *models* of data.

Machine learning algorithms fall into four broad categories:

1. Supervised Learning
2. Semi-supervised Learning
3. Unsupervised Learning
4. Reinforcement Learning

The two lectures and the tutorials are about supervised learning.

Introduction: Approaches

Sometimes people distinguish between:

Machine Learning

Given training data $\boldsymbol{T} = (\boldsymbol{y}, \boldsymbol{x}) = (y_1, x_1), \ldots (y_N, x_N)$, a class of functions $\{ f \}$, and some constraint C on these functions, find the best approximation to the unknown mapping

$$y = f(x)$$

Bayesian Learning

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

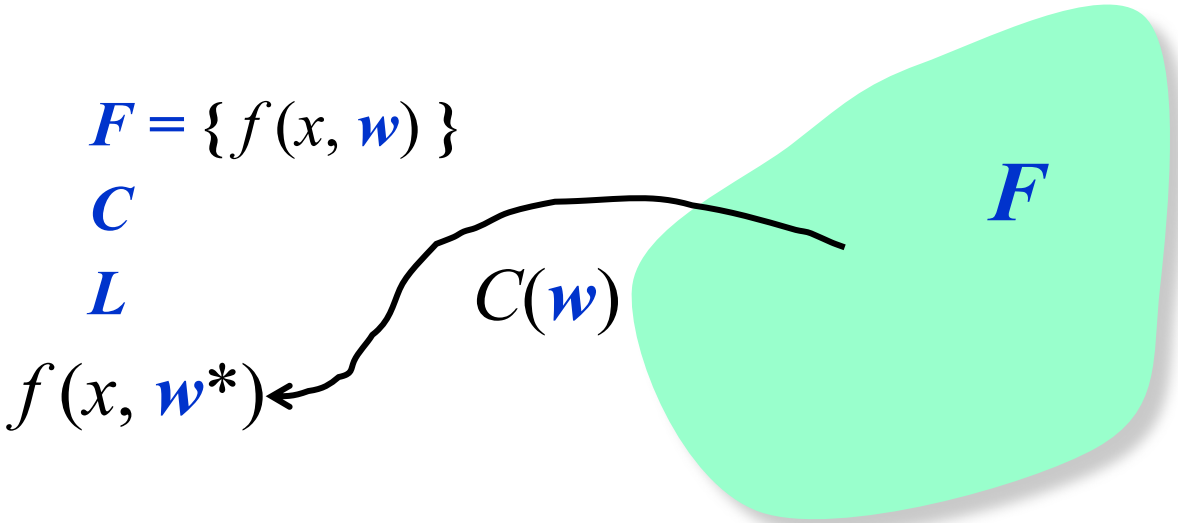
Introduction: Machine Learning

Choose

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

Constraint C

Loss function* L



Method

Find $f(x)$ by minimizing the average loss, sometimes referred to as the empirical risk

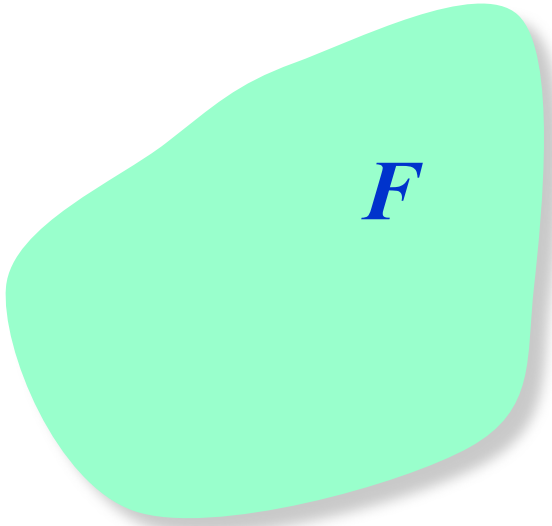
$$R(w) = \frac{1}{N} \sum_{i=1}^{N} L\big(y_i, f(x_i, w)\big) + C(w)$$

*The loss function is a measure of the cost of making a bad choice.

Introduction: Bayesian Learning

Choose

Function space	$\boldsymbol{F} = \{ f(x, \boldsymbol{w}) \}$
Likelihood	$p(\boldsymbol{T} \mid \boldsymbol{w})$, $\boldsymbol{T} = (\boldsymbol{y}, \boldsymbol{x})$
Loss function	$\boldsymbol{L}$
Prior	$p(\boldsymbol{w})$



Method

Use Bayes' theorem to assign a probability density

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

to every parameter point, and therefore every function in the function space.

THEORY

WHAT YOU WANTED TO KNOW BUT WERE AFRAID TO ASK!

Machine Learning

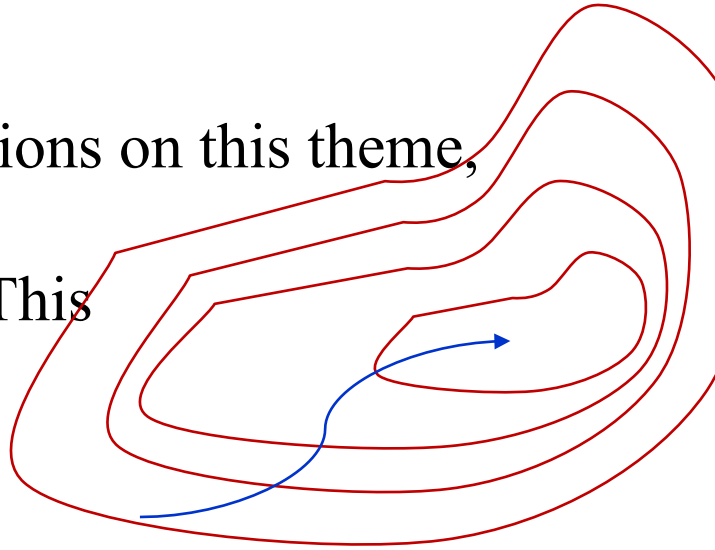
Minimization via Gradient Descent

The empirical risk defines a "landscape" in the space of parameters, or equivalently in the space of functions.

The goal is to find the lowest point in the landscape, usually by moving in the direction of the local *negative* gradient:

$$w_i \leftarrow w_i - \rho \frac{\partial R(w)}{\partial w_i}$$

Most minimization algorithms are variations on this theme, the most popular of which is called Stochastic Gradient Descent (SGD). This uses constantly changing subsets of the training data to provide *noisy* estimates of the gradient.



Machine Learning

Quadratic Empirical Risk

Many methods (decision trees, random forests, neural networks, deep neural networks…) use the quadratic loss function

$$L(y, f) = (y - f)^2$$

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

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

The quadratic loss is popular because it is mathematically convenient. However, it is sensitive to outliers. If outliers are a problem $L(y, f) = \sqrt{(y - f)^2}$ may be a better choice.

Machine Learning

Consider the quadratic risk function in the limit $N \to \infty$

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

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

$$= \int dx\, p(x)\Big[\int dy (y - f)^2 p(y|x)\Big]$$

where $p(y|x) = p(y, x)/p(x)$ and where we have assumed that the influence of the constraint (in this limit) is negligible. Notice that R is a *functional* $R[f]$ of $f(x, w)$, that is, R depends on (infinitely) many values of f.

Machine Learning

If we change the function f by a small *arbitrary* function δf a small change

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

will be induced in R. In general, $\delta R \neq 0$.

But, if the function f is flexible enough we shall be able to reach the minimum of R, where $\delta R = 0$.

We want this to hold for all variations δf and for all values of x. The only way that can happen is if the quantity in brackets is *zero*. This implies

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

Machine Learning: Classification

Suppose that y has only two values, 1 and 0, associated with two distinct classes, s and b, respectively. Then

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

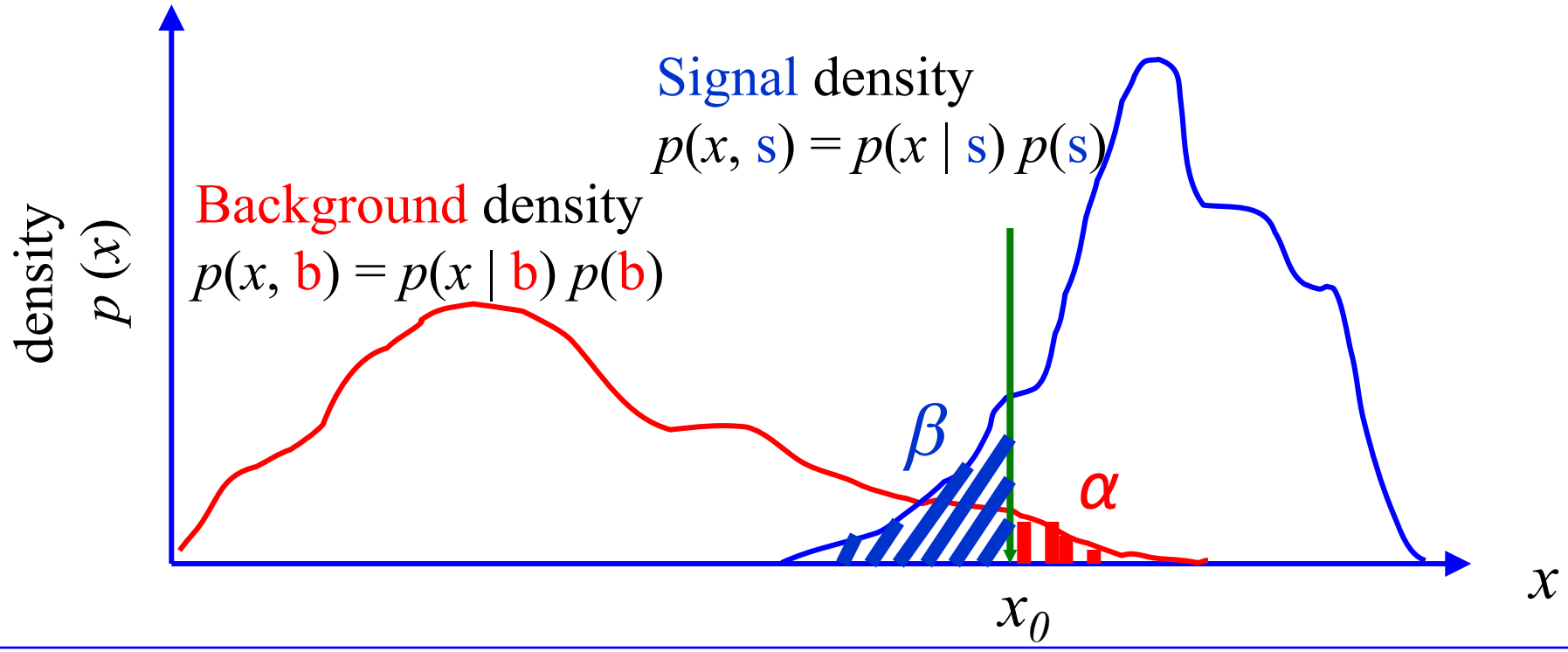
See, Ruck et al., *IEEE Trans. Neural Networks* 4, 296-298 (1990);
Wan, *IEEE Trans. Neural Networks* 4, 303-305 (1990);
Richard and Lippmann, *Neural Computation.* 3, 461-483 (1991)

We therefore conclude the following:

1. given sufficient training data T, with labels $y = 1$ for objects of class s and $y = 0$ otherwise, and
2. given a sufficiently flexible function $f(x, w)$, we can approximate $p(s \mid x)$, *by minimizing the quadratic risk.* Note: this conclusion is independent of the function class!

Machine Learning: Classification

Consider the classification rule: if $f(x) > 0$ assign the object to class s (e.g., the signal class) otherwise assign it to class b (e.g., the background class). Unless the classes are completely separable, misclassifications are inevitable.

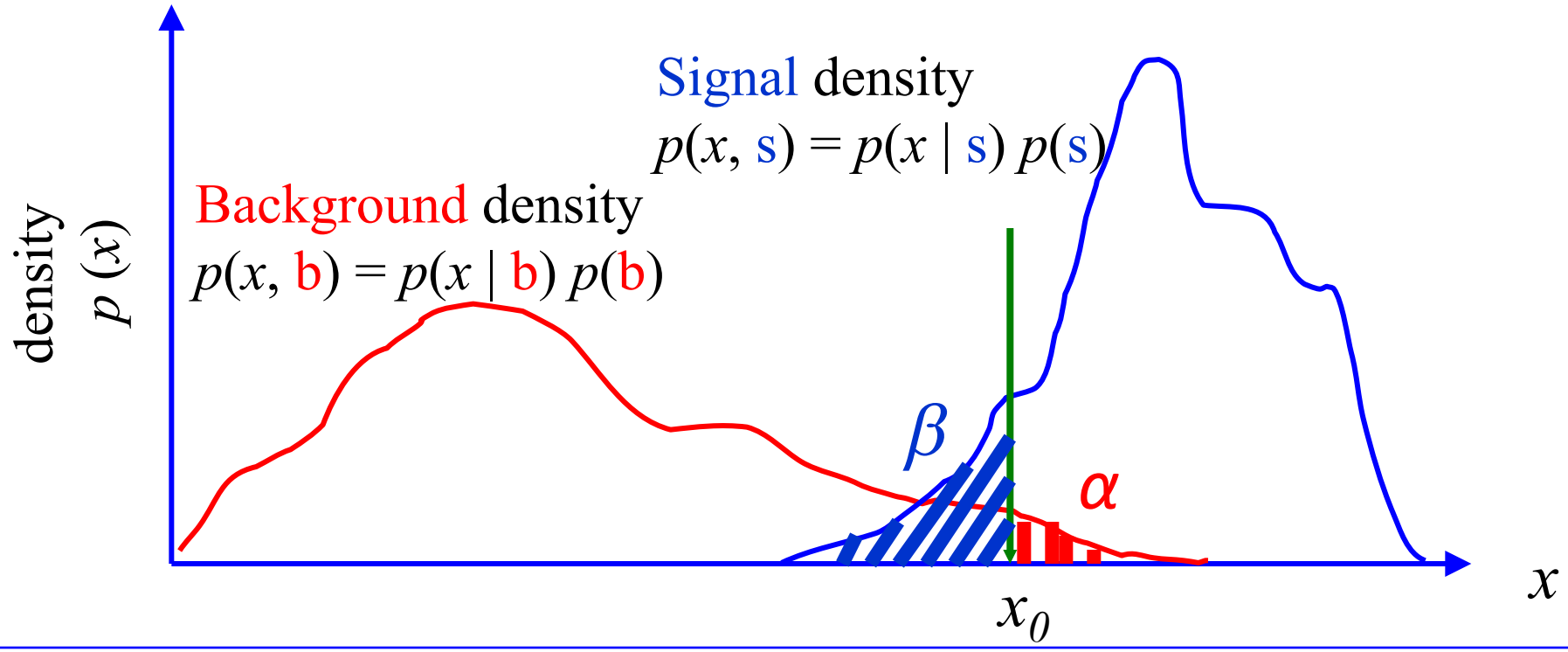


Machine Learning: Classification

Let L_b and L_s, respectively, be the losses incurred due to misclassification. If x is 1-dimensional, the error rate is given by

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

where $H(x, x_0) = 1$ if $x > x_0$, and zero otherwise.



Machine Learning: Classification

Now minimize the error rate with respect to the boundary x_0. This yields

$\frac{L_b}{L_s} = \exp(\lambda(x_0))\frac{p(s)}{p(b)}$, where $\lambda(x) = \ln\frac{p(x|s)}{p(x|b)}$ is the log likelihood ratio. Defining, the discriminant

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

we obtain, finally,

$$p(s \mid x) = \frac{D(x)}{D(x) + [1 - D(x)]/A}$$

where $A = p(s) / p(b)$ is the (prior) signal to noise ratio.

Machine Learning: Classification

We see that the direct minimization of the error rate gives the same answer as minimizing the quadratic risk, namely,

$$f(x) \approx p(s \mid x) = \frac{D(x)}{D(x) + [1 - D(x)]/A}$$

$$A = p(s) / p(b)$$

a result that holds even when x is multidimensional.

In general, different loss functions yield different solutions. But the most commonly used loss functions yield solutions that can be written in terms of the log likelihood ratio:

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

Bayesian Learning

Recall: the goal is to approximate the posterior density $p(w \mid T)$.

Given *new* data x we can compute the predictive distribution

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

that is, the probability (density) of y given x.

If a definite value for y is needed for every x, we can achieve this by minimizing the risk function,

$$R(f) = \int L(y,f) p(y|x,T) dy.$$

For $L = (y-f)^2$, we get the same answer as before, namely,

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

Exercise 1: How is $f(x)$ approximated if $L(y,f) = \sqrt{(y-f)^2}$?

PRACTICE

MAKES PERFECT!

Machine Learning in Practice

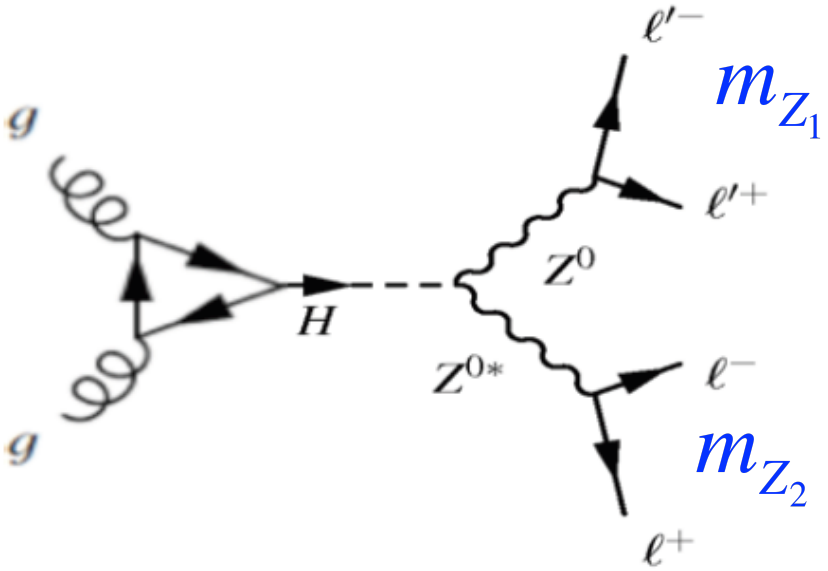
Here is a *very short* list of supervised machine learning methods:

- Grid Searches: Uniform, Random
- Fisher & Quadratic Discriminants
- Decision Trees: Single, Boosted
- Neural Networks: Shallow, Deep, Bayesian
- Naïve Bayes
- Kernel Density Estimation
- Support Vector Machines
- Random Forests

GRID SEARCHES

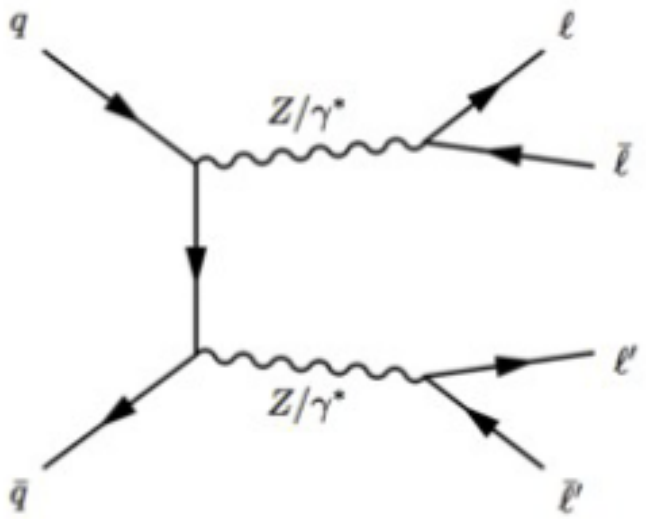
Example 1: Higgs to ZZ to 4 Leptons

Signal



$pp \to H \to ZZ \to l^+\, l^- l'^+ l'^-$

Background



$pp \to ZZ \to l^+\, l^- l'^+ l'^-$

Consider the task of separating signal from background using the variables $x = (m_{Z1}, m_{Z2})$.

Uniform Grid Search

Given the variables $x = (m_{Z1}, m_{Z2})$, the simplest way to try to separate signal from background is to consider n thresholds (cuts) on each variable. In 2 dimensions, we would have to try n^2 tuples of cuts.

But, suppose we have d variables, and wish to do the same thing. Now, we must consider n^d d-tuples of cuts! As d increases, the task becomes computationally prohibitive because the number of cuts to be considered grows rapidly. This is an example of the "*curse of dimensionality*".

We need to be a bit smarter…

Random Grid Search

One way to break the "curse" (or at least to tame it!) is to place cuts where they are more likely to do the most good.

A good place is at the *signal* points, since it is the signal that we wish to extract!

We shall call

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

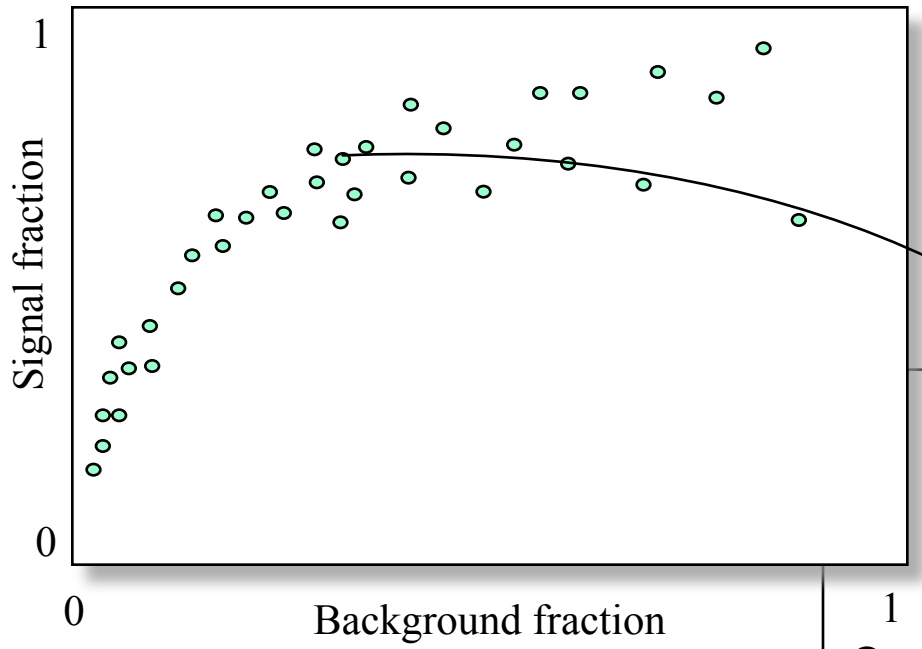
where CUT-DIR: <, >, or ==, a cut-point. In our example, our cut-point is a 2-tuple; in d-dimensions, it is a d-tuple.

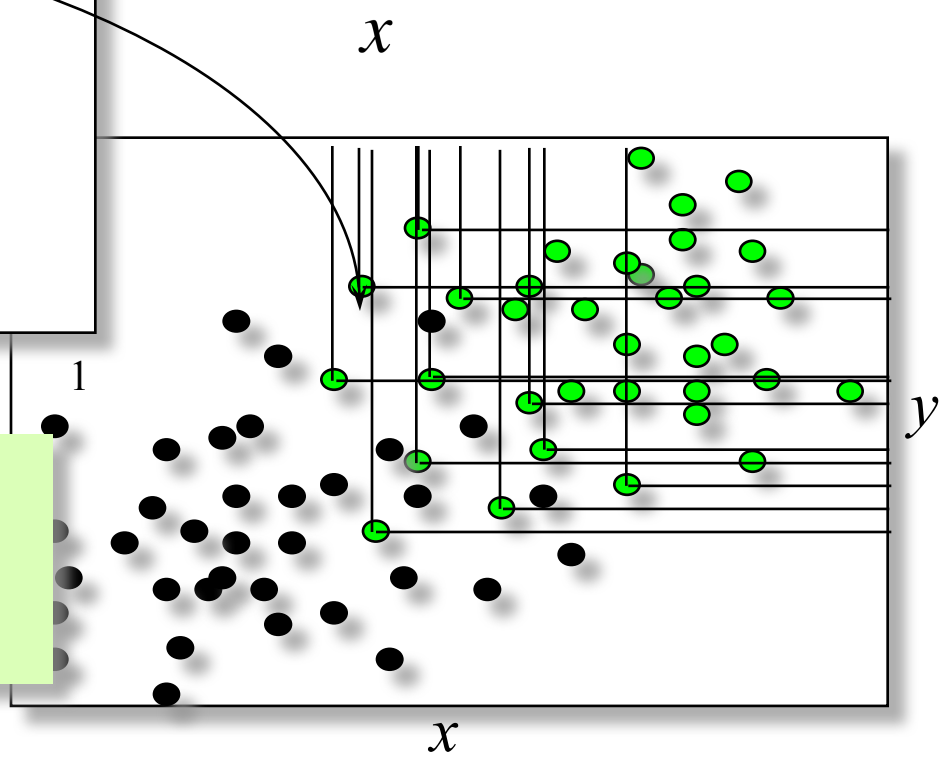
(Note: we can also combine cut-points, to form "box" cuts.)

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**

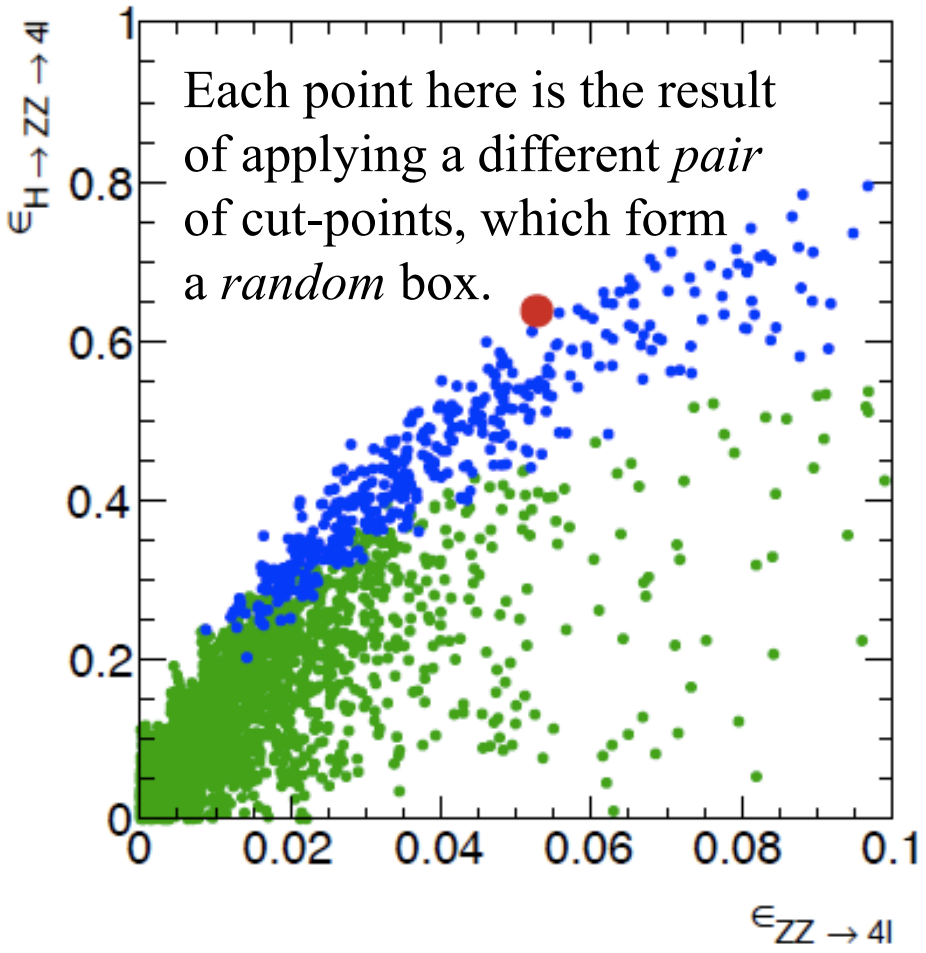


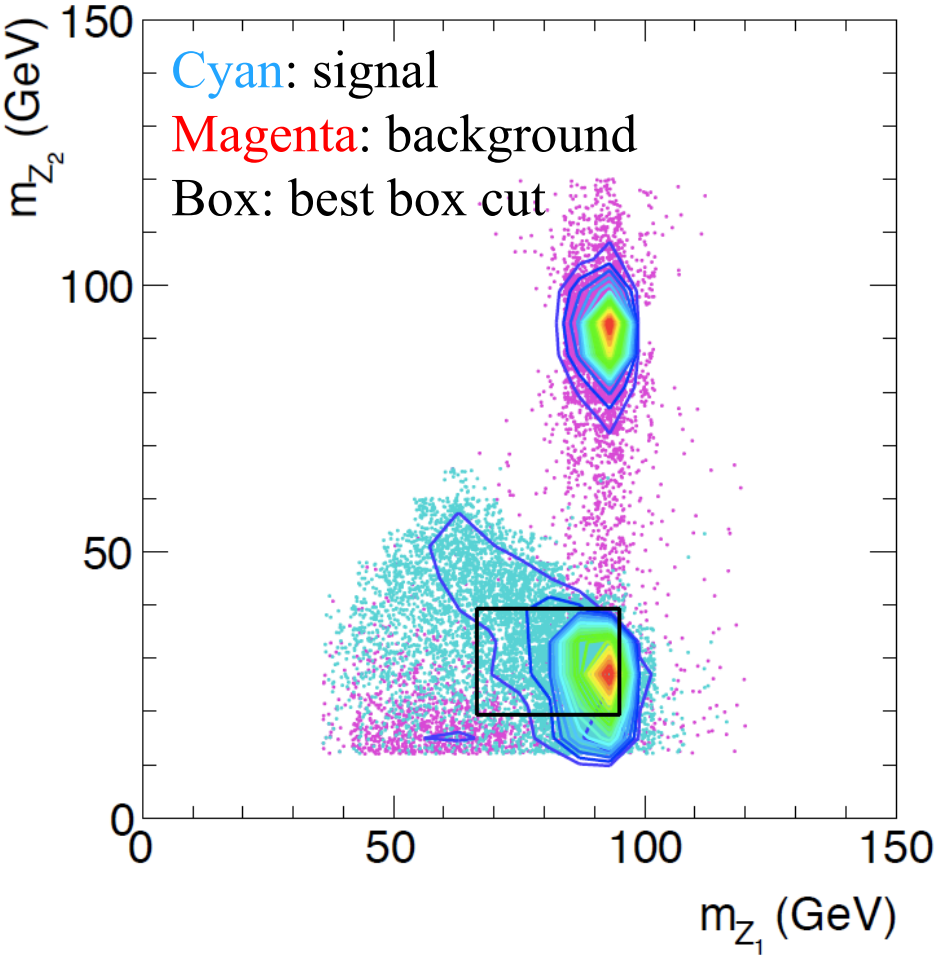


N_{tot} = # events before cuts
N_{cut} = # events after cuts
Fraction = N_{cut}/N_{tot}

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

Example 1: Higgs to ZZ to 4 Leptons





The red point is the best cut, defined in some "*reasonable*" way.

Example 1: Higgs to ZZ to 4 Leptons

But what is "reasonable"?
Let us assume that "*reasonable*" = *minimize average loss*.
But, now, we need a loss function! Define,

$$G(x, s, b) = 2 \ln \frac{p(x|b+s)}{p(x|b)}$$

where, here, s and b denote the *total* mean signal and background counts, respectively, resulting from the cuts.

We can think of $G(x, s, b)$ as a *negative* loss, that is, a gain.
Minimizing average loss is equivalent to maximizing average gain (call it the benefit)

$$B(s, b) = 2 \int p(x|b+s) \ln \frac{p(x|b+s)}{p(x|b)} dx$$

Example 1: Higgs to ZZ to 4 Leptons

Apart from the factor of 2, the function

$$B(s,b) = 2\int p(x|b+s)\ln\frac{p(x|b+s)}{p(x|b)}dx$$

is an instance of the Kullback-Leibler divergence, a measure of the "dissimilarity" between two densities $p(x)$ and $q(x)$,

$$D(p||q) = \int p(x)\ln\frac{p(x)}{q(x)}dx$$

or, when x is discrete,

$$D(p||q) = \sum_x p(x)\ln\frac{p(x)}{q(x)}$$

Example 1: Higgs to ZZ to 4 Leptons

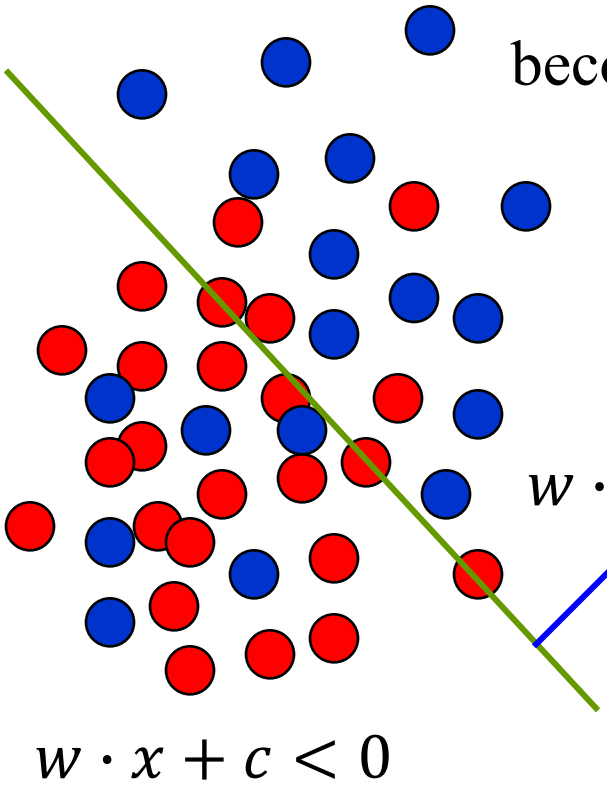
Exercise 2: Show that $B(s,b) = 2bf(s/b)$, where $f(z) = (1+z)\ln(1+z) - z$, when $p(x|b+s)$ and $p(x|b)$ are Poisson distributions and $x = n$ is an integer.

Exercise 3: Show that $B(s,b) = \frac{s}{\sqrt{b}}$ in the limit $s \ll b$.

FISHER & QUADRATIC DISCRIMINANTS

Fisher Discriminant

Suppose we can approximate $p(x \mid s)$ and $p(x \mid b)$ by Gaussian densities with differing means, but with the same covariance matrix, Σ, then the log likelihood ratio $\lambda(x) = \ln \frac{p(x|s)}{p(x|b)}$



becomes

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

which is *linear* in x and where

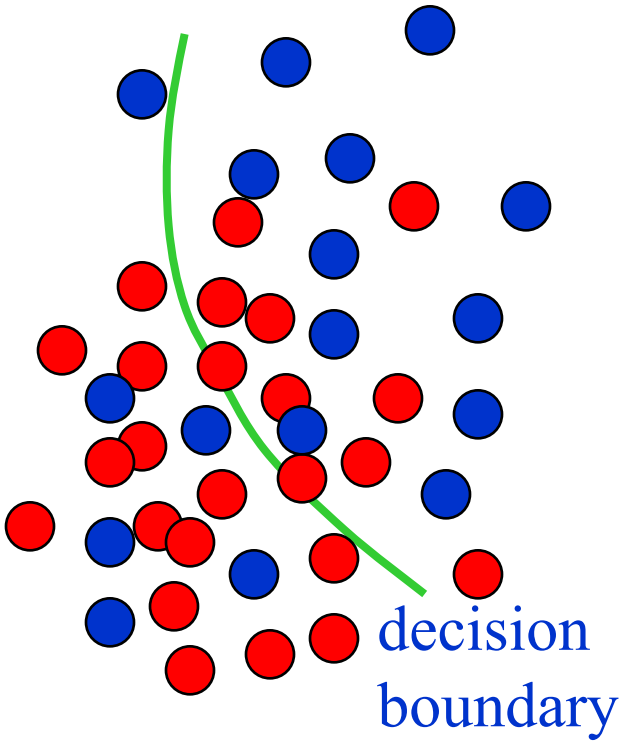
$$w = \Sigma^{-1}(\mu_s - \mu_b)$$

Exercise 4: Show that $\lambda(x)$ is linear in x.

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 defines a *curved* surface in x-space that partitions the space into signal-rich and background-rich regions.

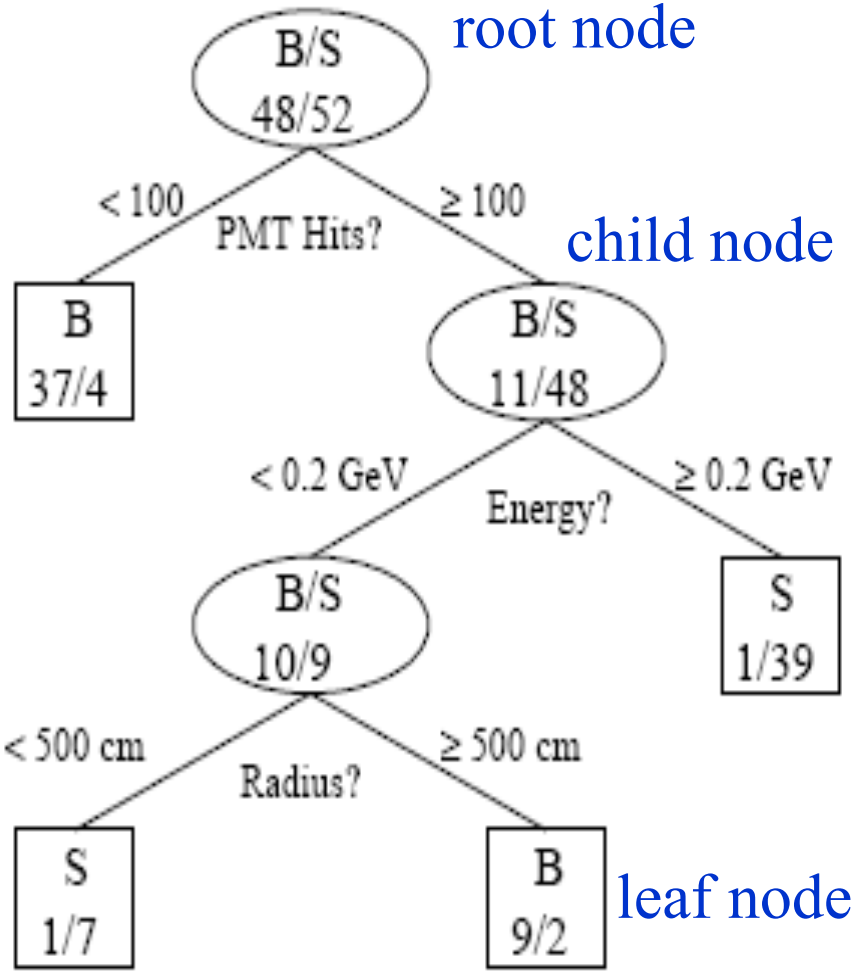
DECISION TREES

Decision Trees

Decision tree:
a sequence of if then else statements.

Basic idea: recursively partition the x-space into regions of diminishing *impurity*.

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

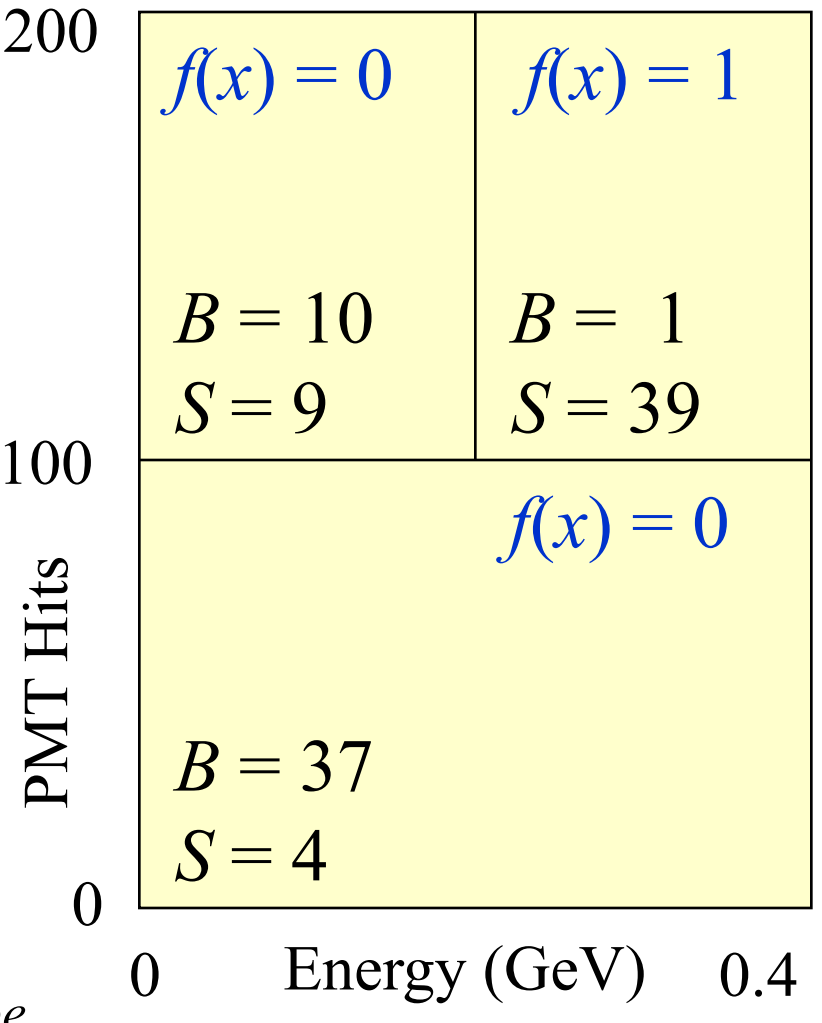


MiniBoone, Byron Roe

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 $f(x)$.



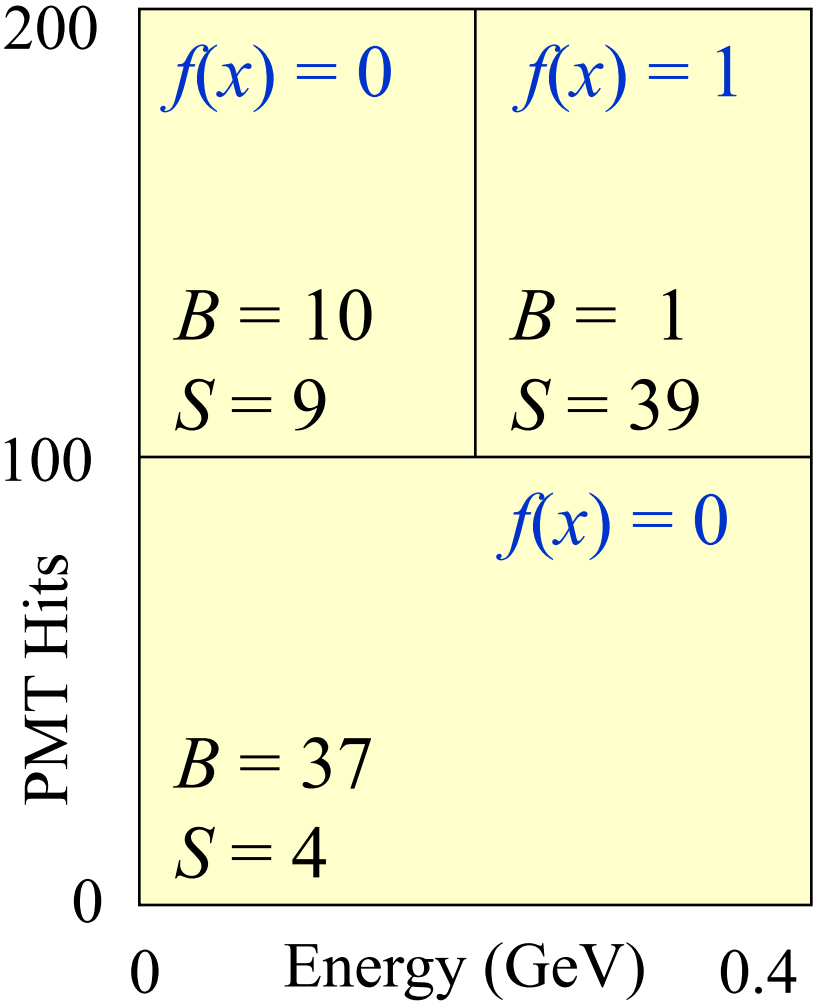
MiniBoone, Byron Roe

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):

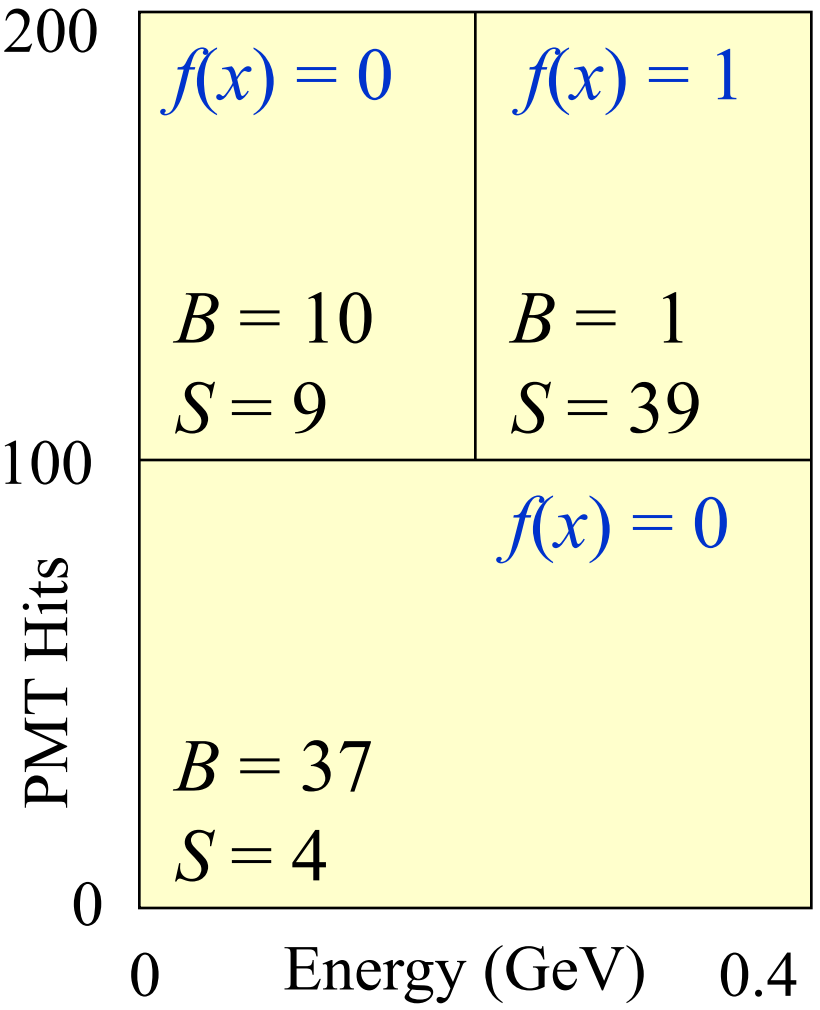
Gini index = $p\,(1-p)$

where p is the purity

$p = S / (S + B)$

$p = 0$ or 1 = maximal purity

$p = 0.5$ = maximal impurity

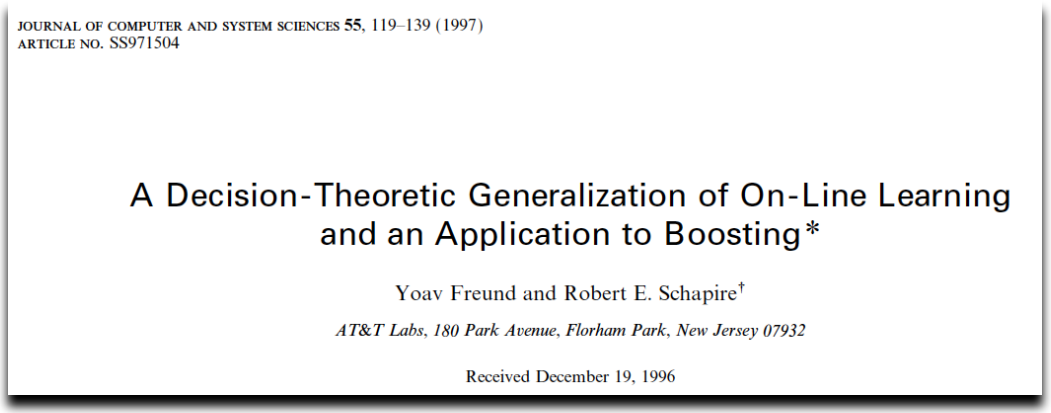


BOOSTED DECISION TREES

Making a Silk Purse from a Pig's Ear!

In 1997, AT&T researchers Freund and Schapire [Journal of Computer and Sys. Sci. **55** (1), 119 (1997)], showed that it was possible to build highly effective classifiers by combining a large number of mediocre ones!

The Freund-Schapire algorithm, which they called AdaBoost, was the first successful method to *boost* (i.e., enhance) the performance of poorly performing classifiers by averaging their outputs.


JOURNAL OF COMPUTER AND SYSTEM SCIENCES 55, 119–139 (1997)
ARTICLE NO. SS971504

A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*

Yoav Freund and Robert E. Schapire†

AT&T Labs, 180 Park Avenue, Florham Park, New Jersey 07932

Received December 19, 1996

Averaging Weak Learners

Suppose that you have a collection of classifiers $f(x, w_k)$, which, individually, perform only marginally better than random guessing. Such classifiers are referred to as weak learners.

A decade later, the idea of Freund and Schapire was developed further by Friedman and Popescu who showed it is possible to build powerful classifiers from mediocre ones through different kinds of averaging:

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

Jerome Friedman & Bogdan Popescu (2008)

Averaging Weak Learners

The most popular averaging methods (mostly used with decision trees as weak learners) are:

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

*A bootstrap sample is a random sample of size N drawn, *with replacement*, from another of the same size. Duplicates can occur and are allowed.

Adaptive Boosting

Each object in the training set is labeled either with $y_n = +1$ or -1 and has weight $w_{(1),n}$, where $\sum_n w_{(1),n} = 1$.

For $k = 1$ to K:

1. Create a decision tree $f(x, w)$ using current weights.
2. Compute its error rate ε on the *weighted* training set.
3. Compute $\alpha_k = \ln(1-\varepsilon) / \varepsilon$
4. Modify training sample:

   $w_{(k+1),n} \leftarrow w_{(k),n} \exp\left[-\frac{\alpha_k f(x_n, w_{(k)}) y_n}{2}\right]$ and

   normalize weights $\sum_n w_{(k+1),n} = 1$.

The BDT is the weighted average $f(x) = \sum \alpha_k f(x, w_{(k)})$

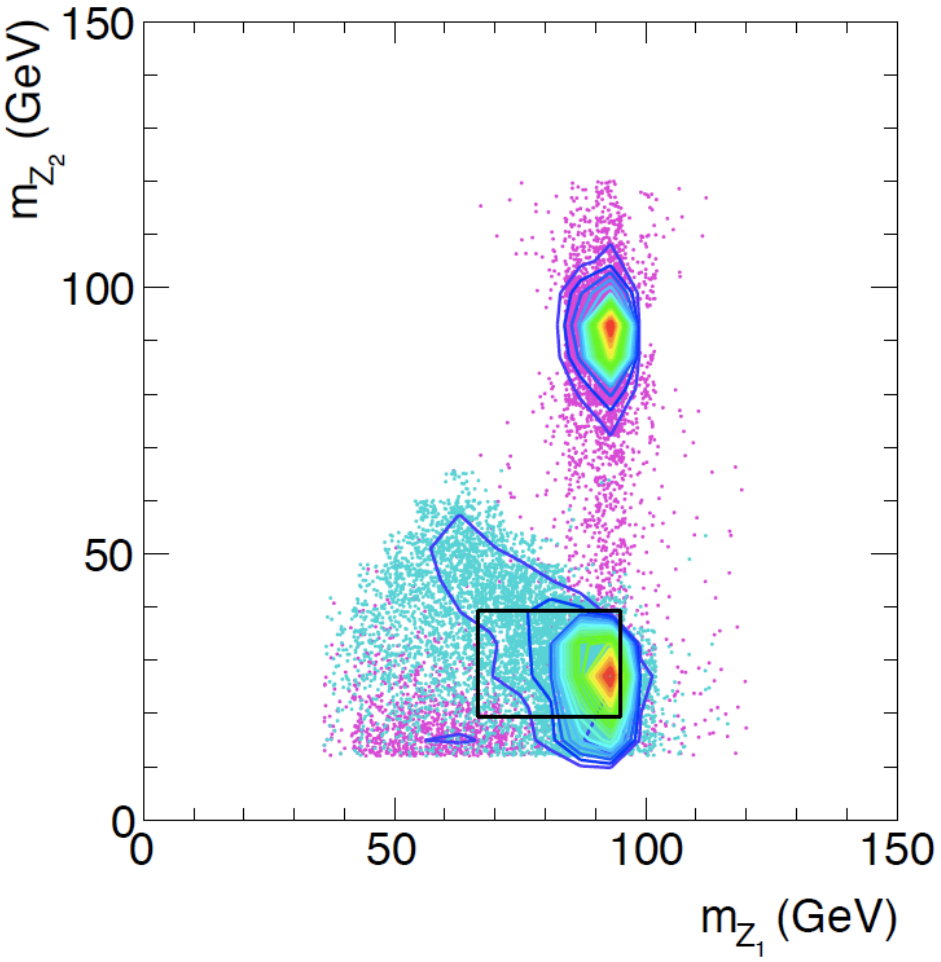
Y. Freund and R.E. Schapire, Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

Example 1: Higgs to ZZ to 4 Leptons

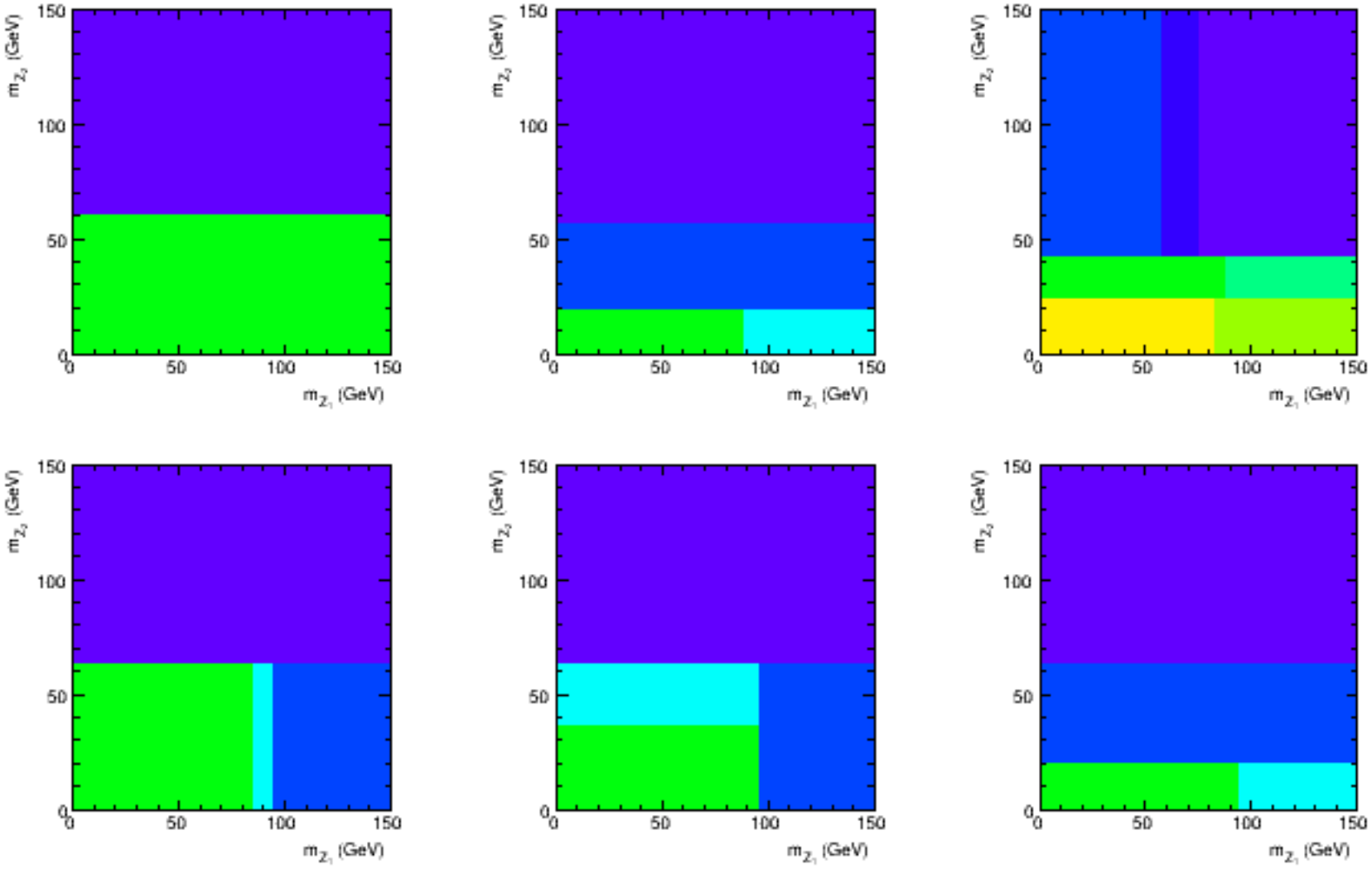
Apply AdaBoost using
the same variables as before:

$$x = (m_{Z1}, m_{Z2}).$$

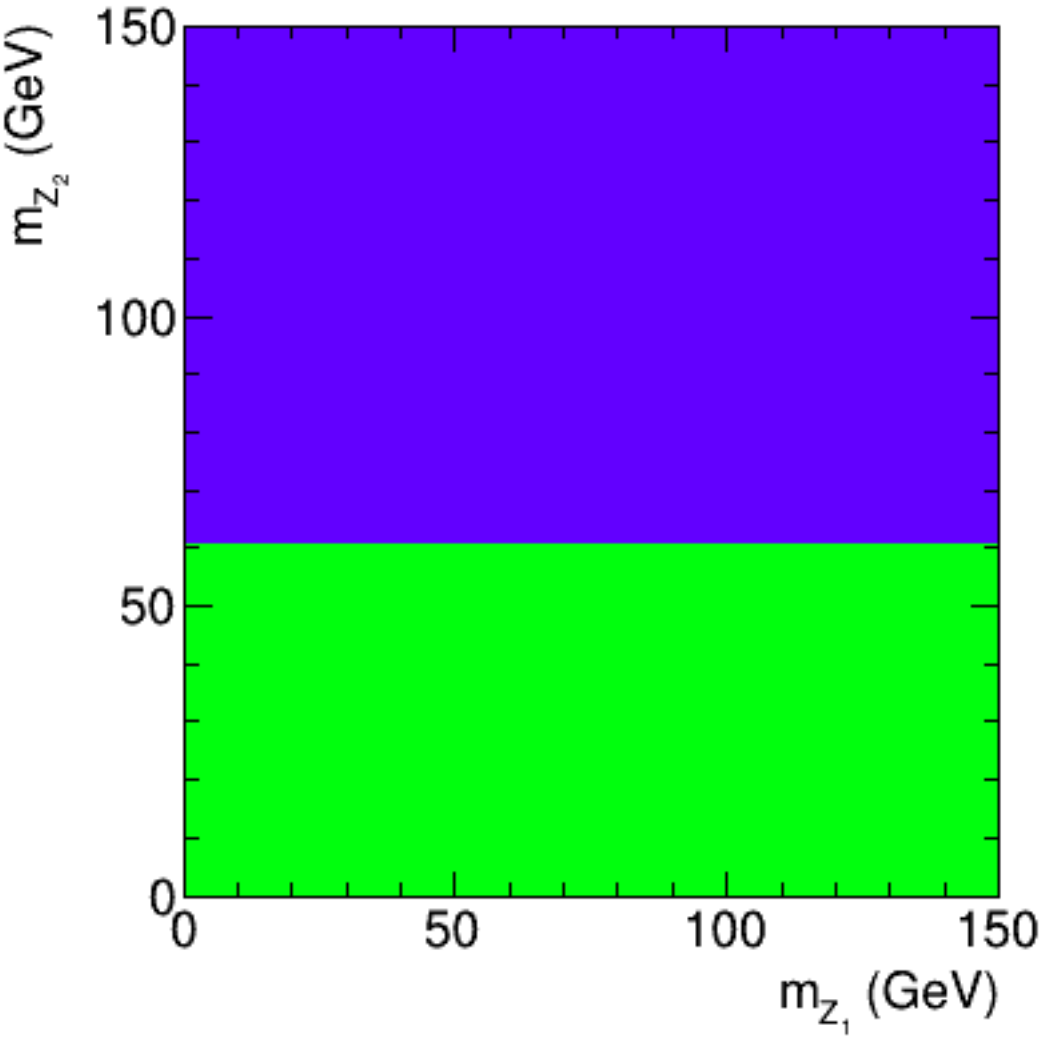
Magenta is the $pp \to ZZ \to 4l$
background and cyan
the $pp \to H \to ZZ \to 4l$
signal.



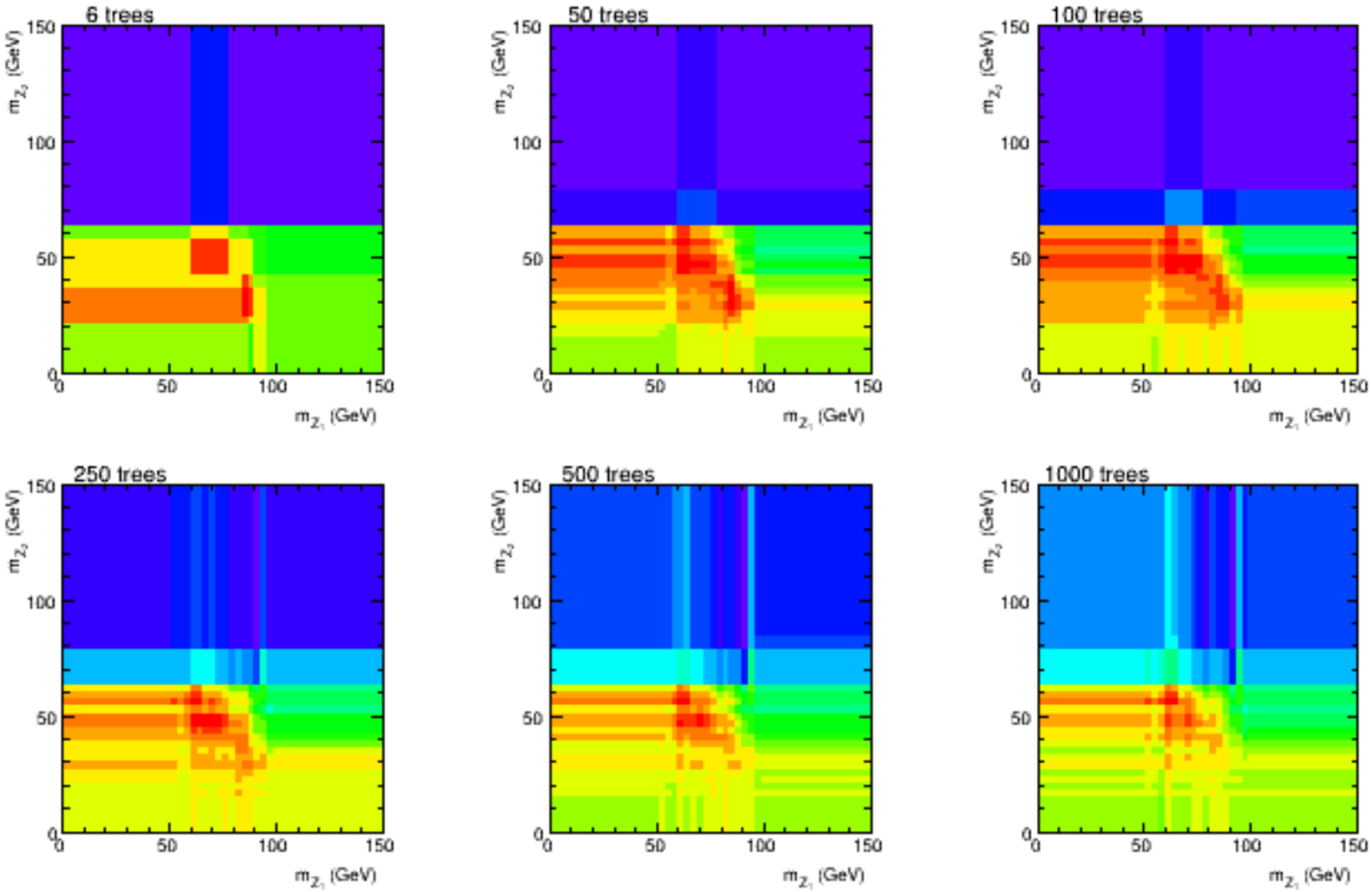
First 6 Decision Trees



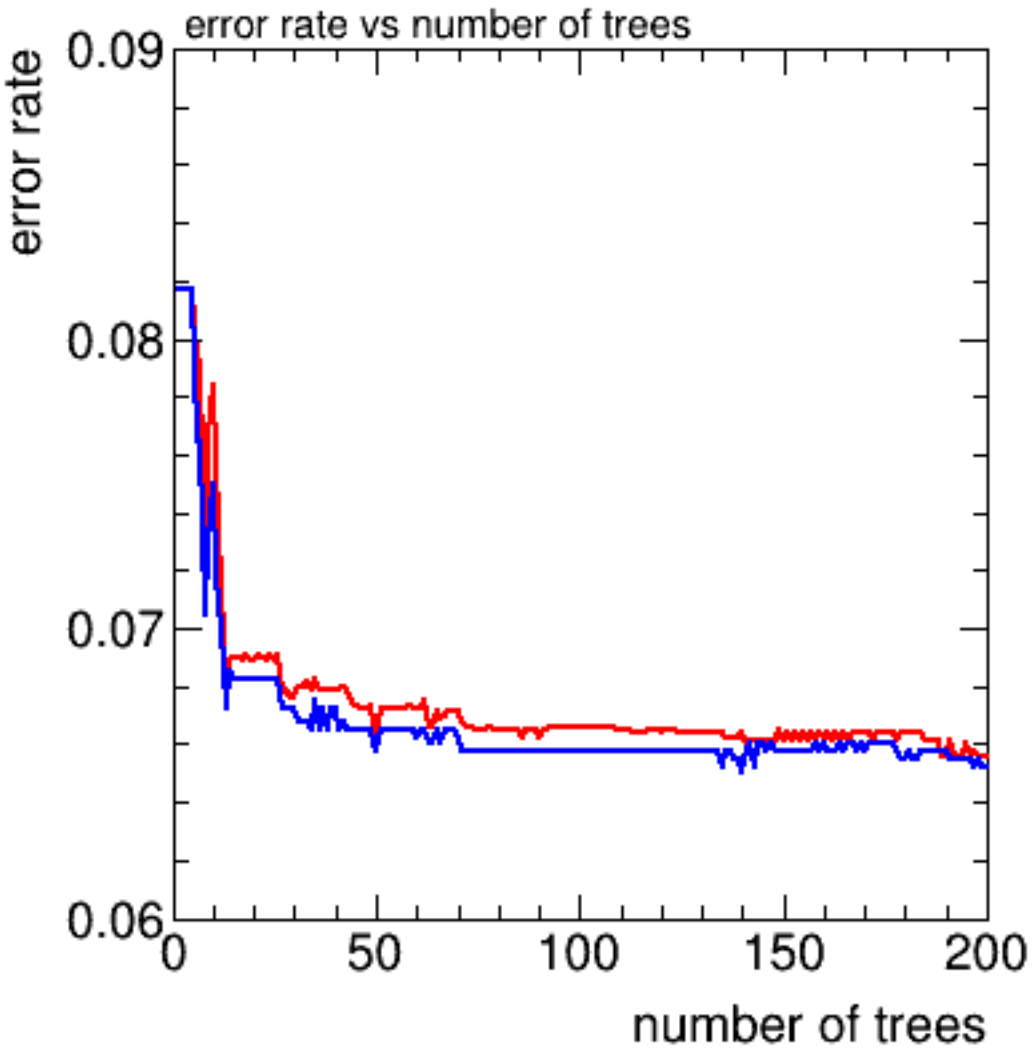
First 100 Decision Trees

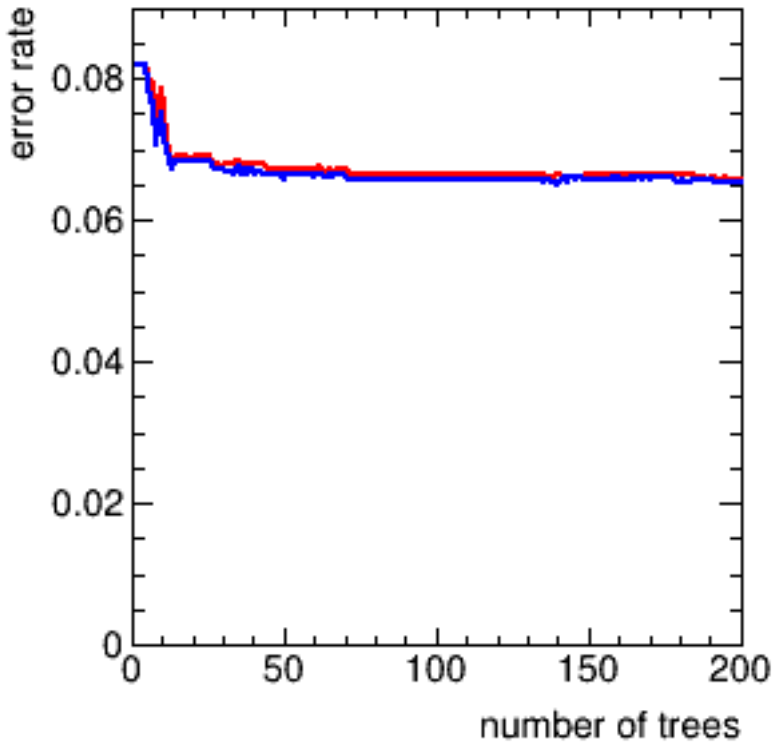


Averaging over (a Forest of!) Trees



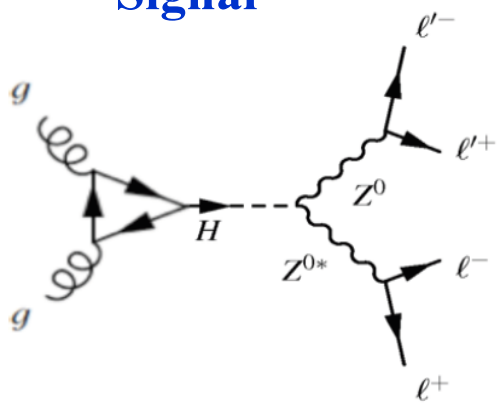
Error Rate vs. Number of Trees





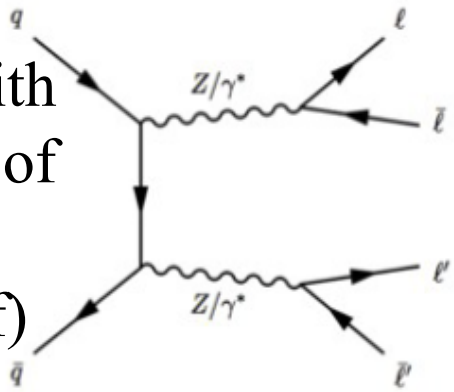
Example 1: Higgs to ZZ to 4 Leptons

Signal



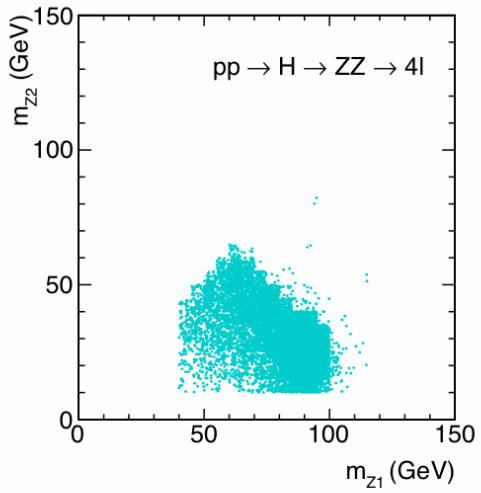
pp

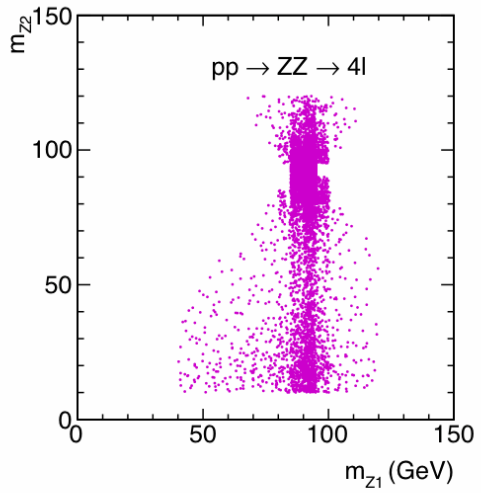
Background

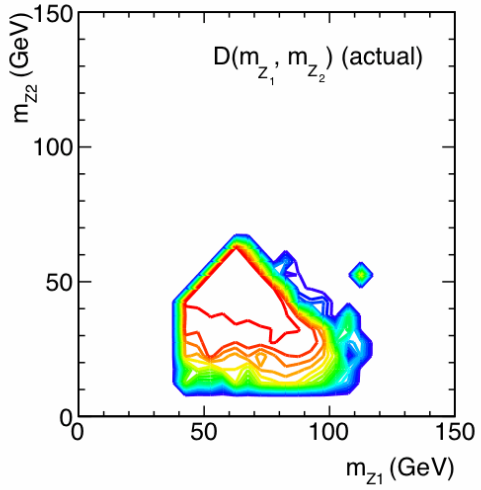


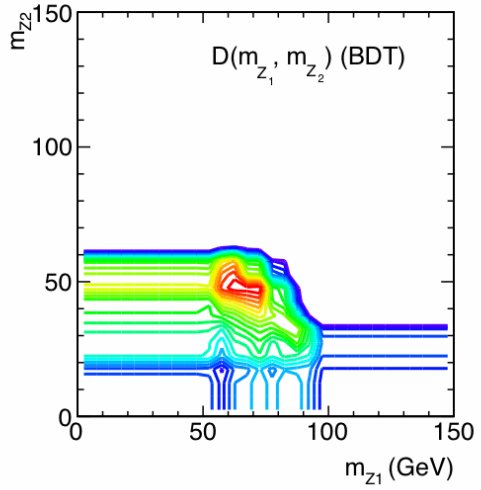
pp

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









AdaBoost: Why Does It Work?

AdaBoost is a rather non-intuitive algorithm. Why it works as well as is does remains a matter of debate.

There is considerable evidence that AdaBoost is highly resistant to *overtraining*.

Indeed, it is possible for the error rate on the training set to go to zero, yet remain roughly constant on the test set.

Also, AdaBoost does not seem to fit into the theory we discussed earlier: in particular, it is not clear what loss function, if any, it is minimizing.

AdaBoost: Why Does It Work?

Soon after its invention, however, Friedman, Hastie, and Tibshirani showed that the algorithm is mathematically equivalent to minimizing the following risk function

$$R(f) = \int \exp(-yF(x))\, p(x,y)dxdy,$$

where $F(x) = \sum_{k=1}^{K} \alpha_k f(x, w_k)$.

This is interesting because is implies that the BDT output $F(x)$ can be mapped to

$$D(x) = 1/(1 + \exp(-2F(x))$$

and interpreted as probability!

J. Friedman, T. Hastie and R. Tibshirani, ("Additive logistic regression: a statistical view of boosting," The Annals of Statistics, 28(2), 377-386, (2000))

Example 2: Wine Tasting

Let's try our hand at automated "wine tasting" using the wine tasting example based on data by Cortez et al.*

http://www.vinhoverde.pt/en/history-of-vinho-verde

* P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

Example 2: Wine Tasting

AdaBoost will be used to build a classifier that can distinguish between good wines and bad wines from the Vinho Verde area of Portugal using the data from Cortez *et al.*

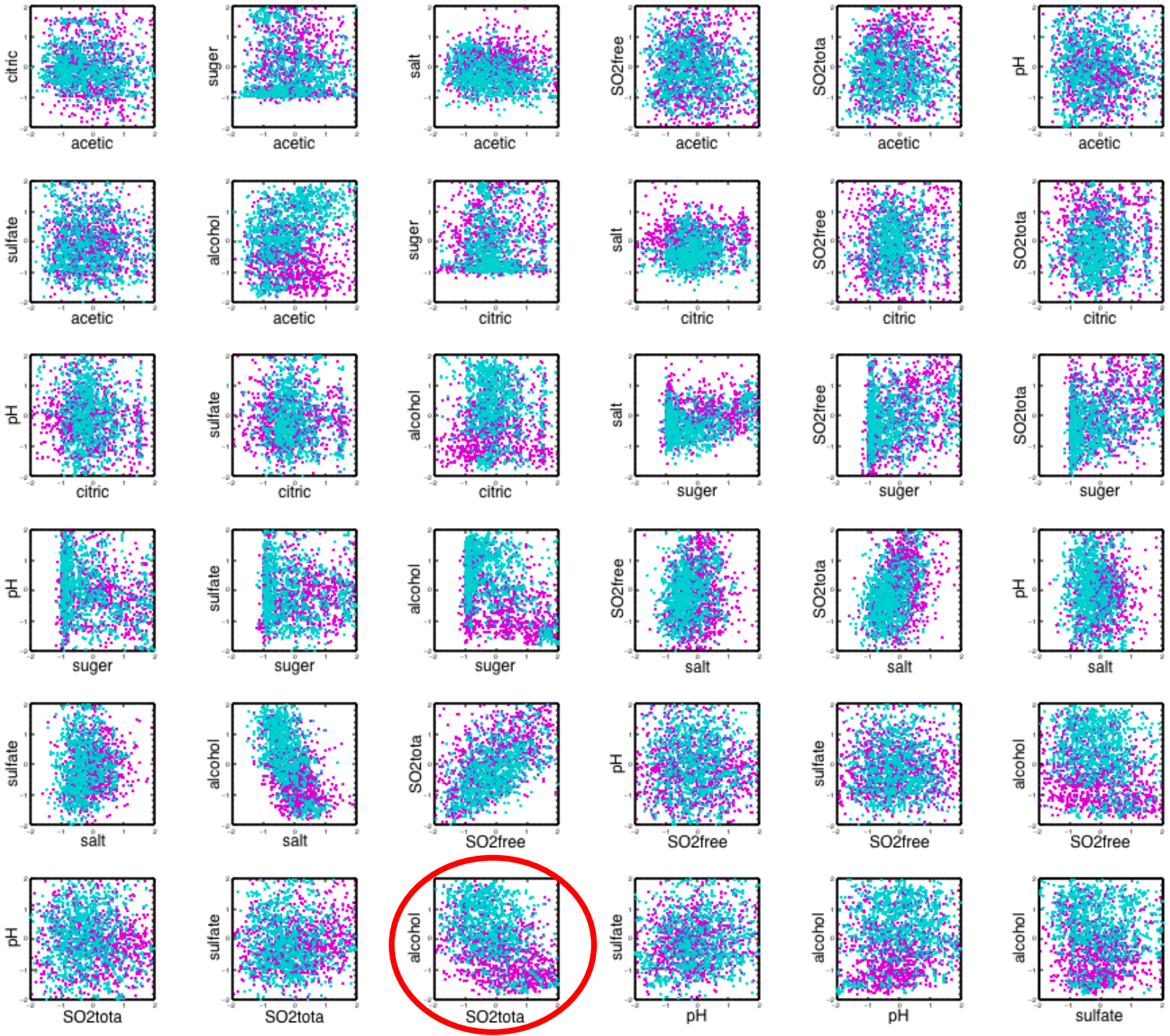
We'll define a good wine as one with expert rating ≥ 0.7 on a scale from 0 to 1, where 1 is a wine from Heaven and 0 is a wine from Hell!

First, let's look at the training data…

Example 2: Wine Tasting

Data: [Cortez *et al.*, 2009].

variables	description
acetic	acetic acid
citric	citric acid
sugar	residual sugar
salt	NaCl
SO2free	free sulfur dioxide
SO2tota	total sulfur dioxide
pH	pH
sulfate	potassium sulfate
alcohol	alcohol content
quality	(between 0 and 1)

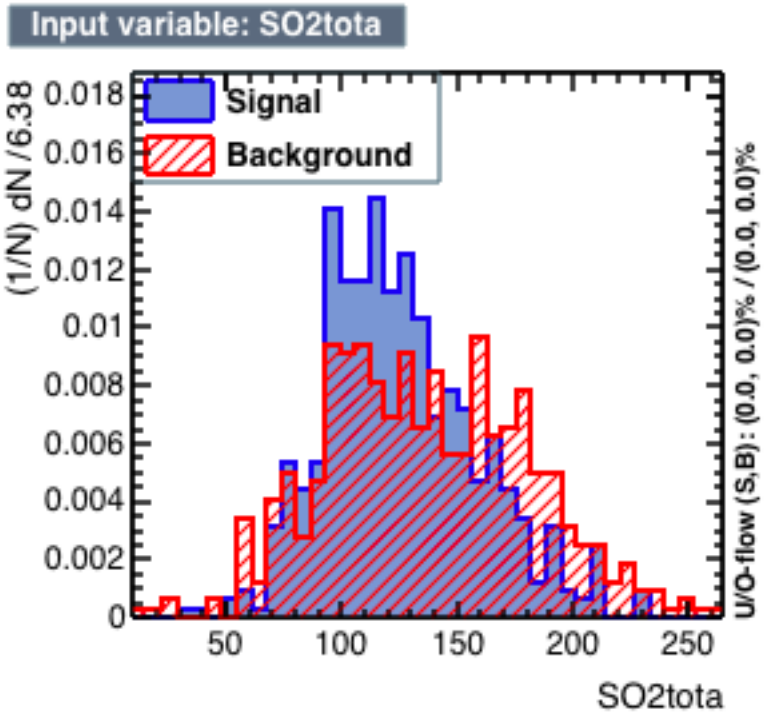

citric acetic
suger acetic
salt acetic
SO2free acetic
SO2tota acetic
pH acetic
sulfate acetic
alcohol acetic
suger citric
salt citric
SO2free citric
SO2tota citric
pH citric
sulfate citric
alcohol citric
salt suger
SO2free suger
SO2tota suger
pH suger
sulfate suger
alcohol suger
SO2free salt
SO2tota salt
pH salt
sulfate salt
alcohol salt
SO2tota SO2free
pH SO2free
sulfate SO2free
alcohol SO2free
pH SO2tota
sulfate SO2tota
alcohol SO2tota
sulfate pH
alcohol pH
alcohol sulfate

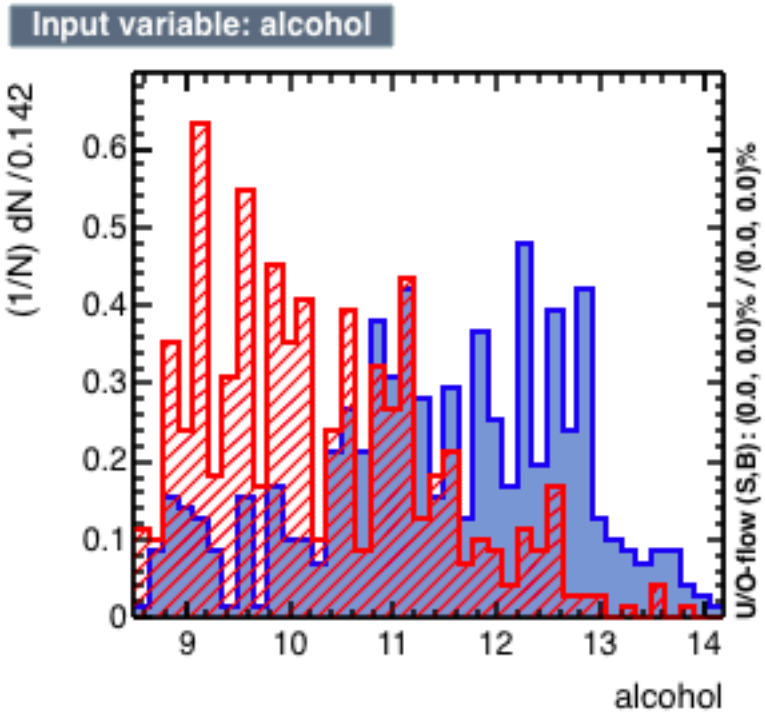
Example 2: Wine Tasting

To make visualization easier, we'll use only two variables:

SO2tota: the total sulfur dioxide content (mg/dm^3)

alcohol: alcohol content (% volume)

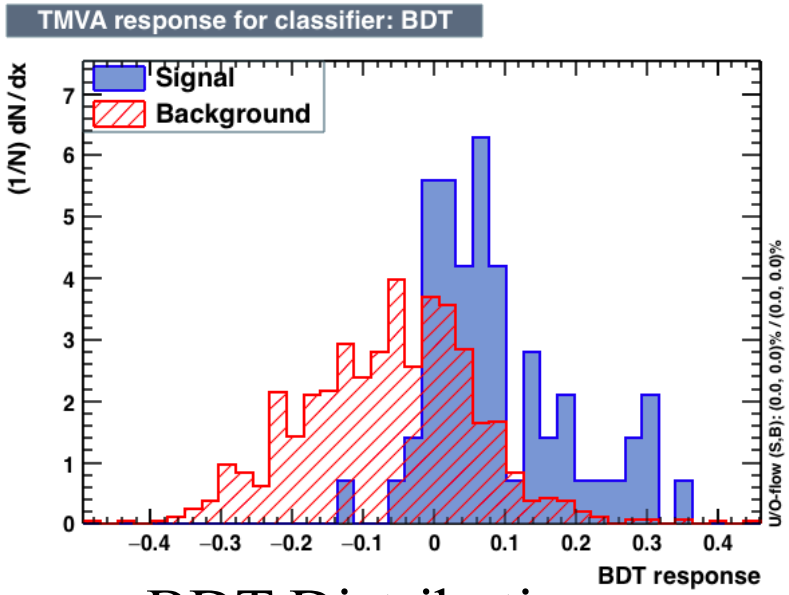




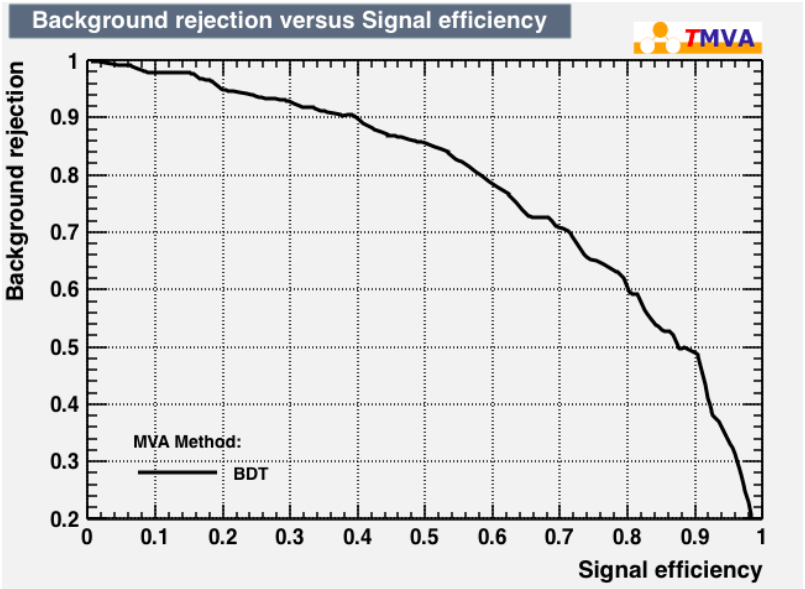
Example 2: Wine Tasting – Results

x = SO2tota

y = alcohol



BDT Distribution



Fraction of bad wine rejected for a given fraction of good wine accepted.

$$BDT(x,y) = \sum_{k=0}^{99} \alpha_k f(x,y,w_k)$$

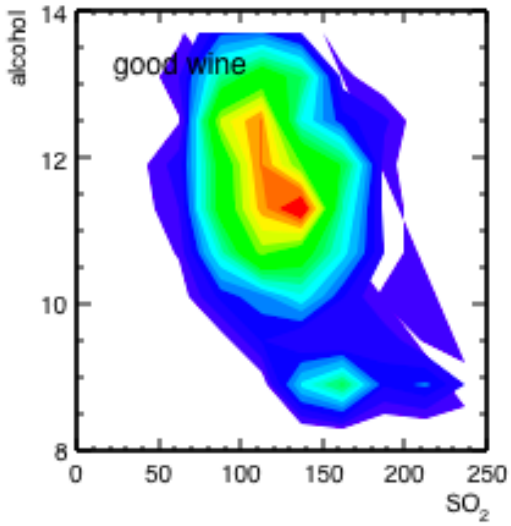
Example 2: Wine Tasting – Results

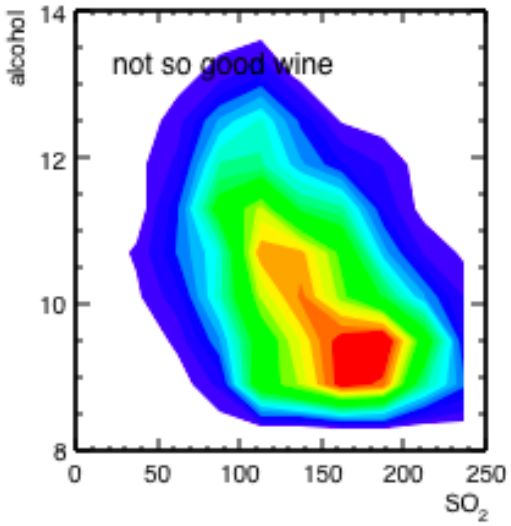
The upper figures are density plots of the training data.

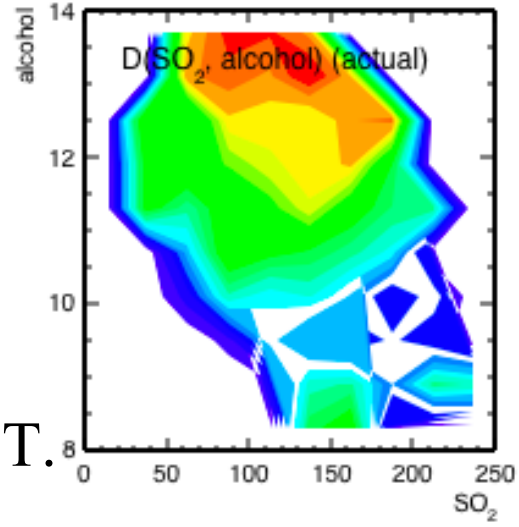
The lower plots are approximations of the discriminant

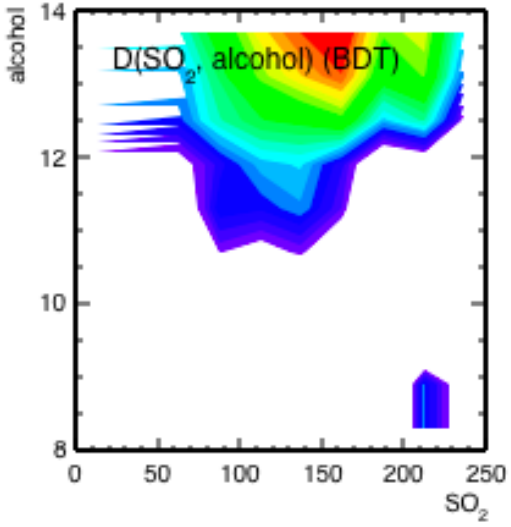
$D(x, y)$

The left, uses 2-D histograms, the right uses the remapped BDT.









Summary

- Machine learning has the reputation of being all about creating black boxes that do amazing things, such as automatically tag faces on *Facebook*.
- In fact, machine learning is simply the use of computers to build *automatically* useful models of data.
- After a bit of theory, we considered simple examples of classification using the random grid search and boosted decision trees.
- Tomorrow, we consider neural networks.