Multivariate Discriminants II

Harrison B. Prosper
Florida State University

School Of Statistics

Insitut Pluridisciplinaire Hubert Curien, Strasbourg

30 June 2008 – 04 July 2008

Outline

- Introduction

- Support Vector Machines
- Naïve Bayes
- Kernel Density Estimation
- Bayesian Neural Networks

- Issues

- Summary

Introduction

The goal is to approximate the function D(x)

$$\boxed{D(x) = \frac{s(x)}{s(x) + b(x)}}$$

where

$s(x)$	signal density
$b(x)$	background density
$d(x) = \varepsilon\, s(x) + (1 - \varepsilon)\, b(x)$	data density
$\varepsilon = k/(1+k)$	signal fraction
$k = p(S)/p(B)$	signal/background ratio

Introduction

The function D(x) is useful for

- Classification $D(x) > D_0$
- Signal extraction $w(x) = p(S|x) = D/[D+(1-D)/k]$
- Data compression $R^d \rightarrow [0,1] \quad (x \rightarrow D)$

Support Vector Machines

Support Vector Machines

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 dimension, H

$$h : R^d \to R^H$$

Use a hyper-plane to partition the high dimensional space

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

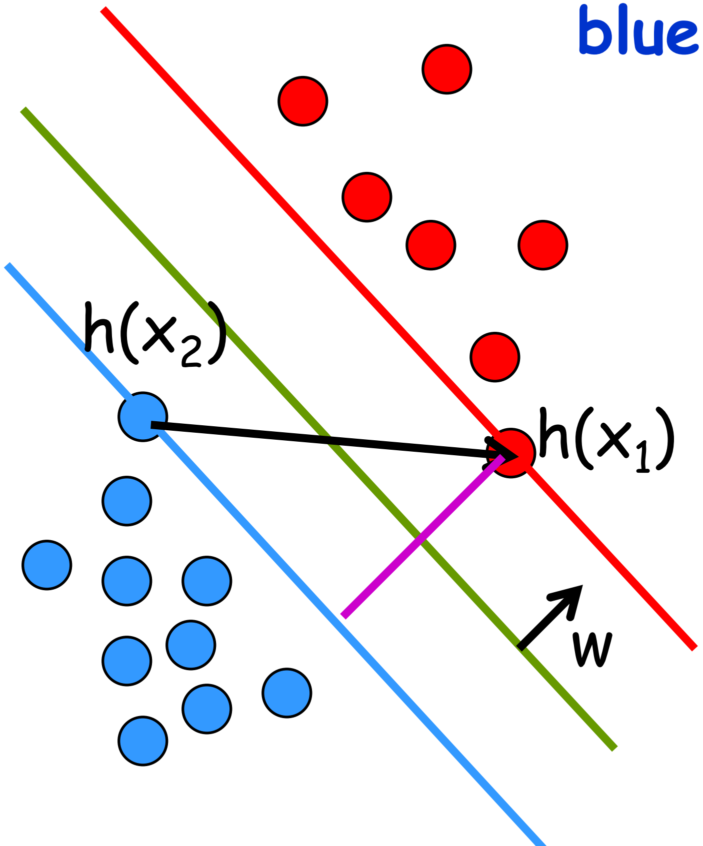
Support Vector Machines

Consider separable data in the high dimensional space

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

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

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



subtract blue from red

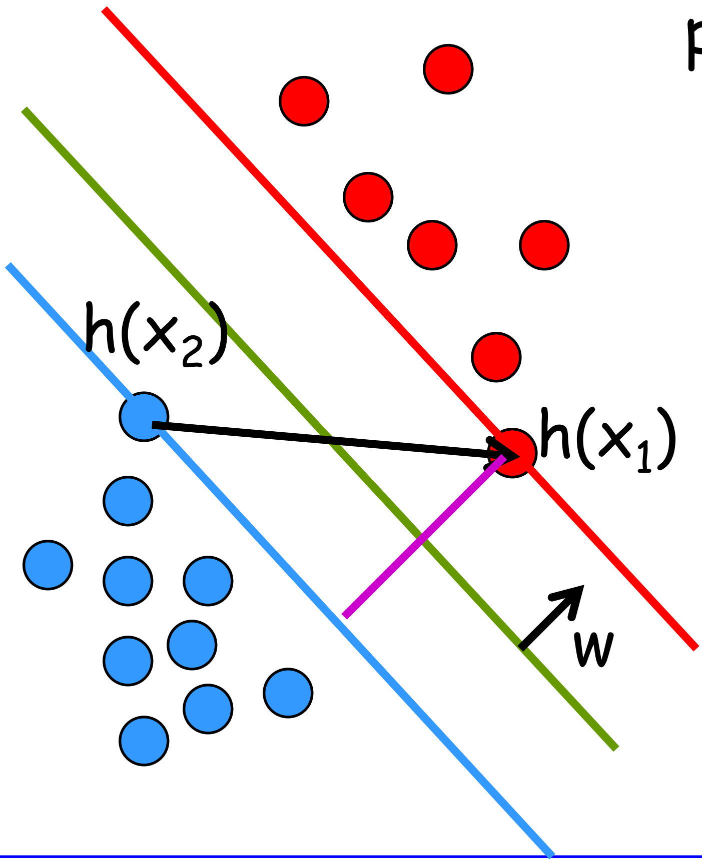
$w.[h(x_1)-h(x_2)] = 2$

and normalize the 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.

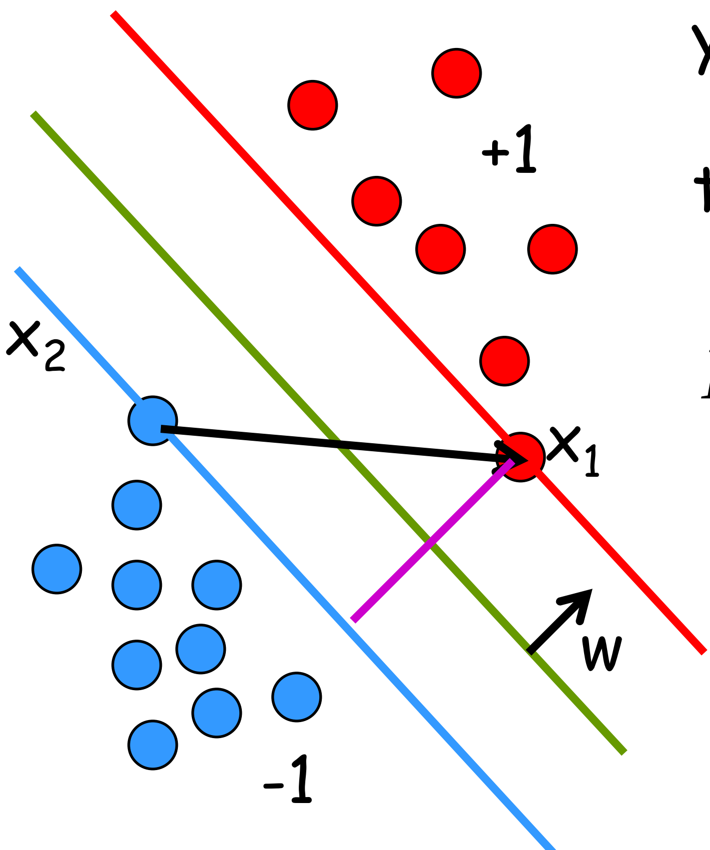


Note: because $m \sim 1/||w||$, maximizing the margin is equivalent to minimizing

$$||w||^2$$

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 constraints



$$y_i\,[w.h(x_i) + b] \geq 1, \quad i = 1 \ldots N,$$

that is, to minimize the function

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

where the $\alpha > 0$ are Lagrange multipliers

Support Vector Machines

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

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

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 **support vectors**.

Support Vector Machines

In general, data are not separable and the constraints have to be relaxed, for example,

$$y_i.(w.x_i + b) \geq 1 - \xi_i$$

by introducing so-called **slack variables** ξ_i .

Important: Because of the scalar product structure one can use **kernels** $K(x_i, x_j) = h(x_i).h(x_j)$ to perform simultaneously the mapping to high dimensions and the scalar product *efficiently*, even in a space of infinite dimensions!

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

SVM – h:R^2 -> R^3

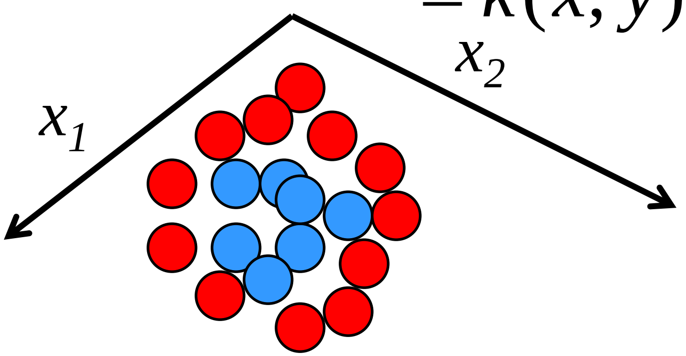
Example

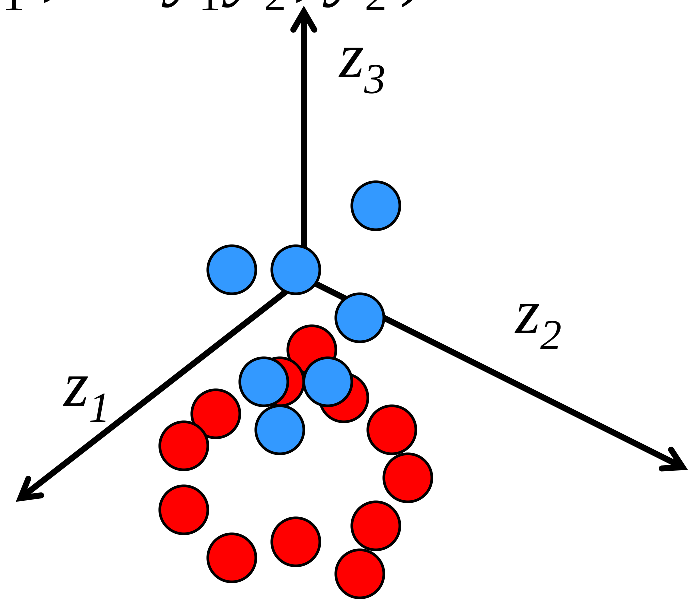
$$h:(x_1, x_2) \rightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$h(x) \cdot h(y) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$= (x \cdot y)^2$$

$$= k(x, y)$$





Since we do not know which mapping h: x -> z is best for a given problem, we must try different kernels.

Naïve Bayes

Naïve Bayes

The method is very simple: ignore the dependencies between variables and approximate the density p(x) by

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

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 then given by

$$\boxed{D(x) = \frac{\hat{s}(x)}{\hat{s}(x) + \hat{b}(x)}}$$

In spite of its name, this method can often yield good results.

It should be tried, because it is easy to compute and the 1-d densities can be approximated with kernel density estimation (KDE), which is the next topic

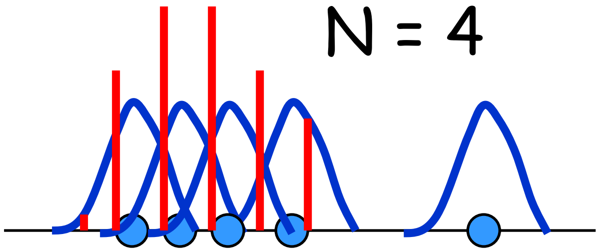
Kernel Density Estimation

Kernel Density Estimation

Basic Idea

Parzen Estimation (1960s)

$$\hat{p}(x) = \frac{1}{N}\sum_{n=1}^{N} K\left(\frac{x - z_n}{h}\right)$$



Mixtures

$$\hat{p}(x) = \sum_{j} w_j \varphi_j(x) \qquad j << N$$

Kernel Density Estimation

Why does it work? In the limit N -> ∞

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

the true density p(x) will be recovered because

$$K\left(\frac{x - z_n}{h}\right) \to \delta^d(x - z), \quad N \to \infty$$

The KDE is therefore a **consistent** estimator of the probability density p(x)

Kernel Density Estimation

In principle, so long as the kernel -> δ-function in the N -> ∞ limit *any* kernel will do.

In practice, the most commonly used kernel is the product of 1-D Gaussians, one for each dimension

$$K\left(\|x-z\|\right) = \exp\left[-\sum_{i=1}^{d}\left(\frac{x-z_i}{h_i}\right)^2 / 2\right] / h_i (2\pi)^{d/2}$$

The h_i are called the **bandwidths**

Kernel Density Estimation

One advantage of a KDE is that the number of adjustable parameters can be made small

Indeed, if the same bandwidth **h** is used for all dimensions, then there will be only a *single* adjustable parameter

$$K(\|x-z\|) = \exp\left[-\sum_{i=1}^{d}\left(\frac{x-z_i}{h}\right)^2/2\right]/h^d(2\pi)^{d/2}$$

SOS '08

Kernel Density Estimation

The **optimal bandwidths** are those yielding the best kernel density estimate of p(x). In principle, this can be found by minimizing the risk function

$$\boxed{R(\hat{p}, p) = \int [\hat{p}(x) - p(x)]^2 dx}$$

In practice, one minimizes some approximation of it. For d = 1, the (approximate) optimal bandwidth is given by

$$\hat{h} = \left(\frac{m_2}{k_2 p_2 N} \right)^{1/5}$$

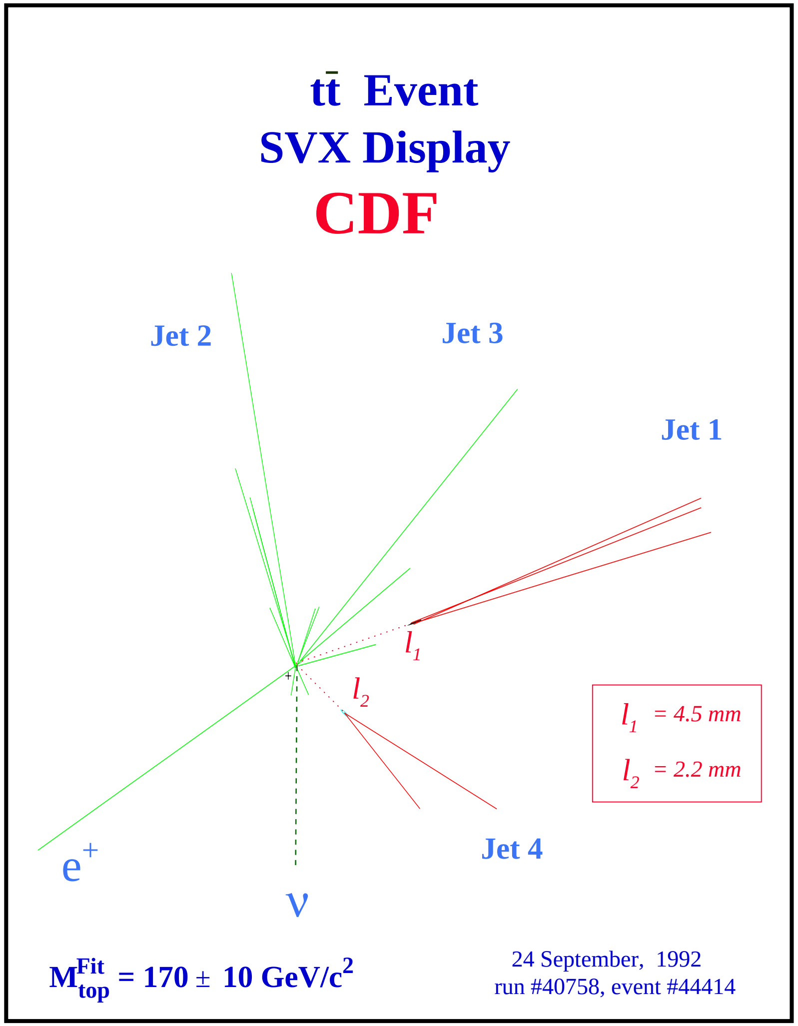
where

$$m_2 = \int x^2 K(x) dx$$

$$k_2 = \int K(x)^2 dx$$

$$p_2 = \int p''(x)^2 dx$$

KDE Example: b-Tagging



Two varieties of jet:

1. **Tagged** (Jet 1, Jet 4)
2. **Untagged** (Jet 2, Jet 3)

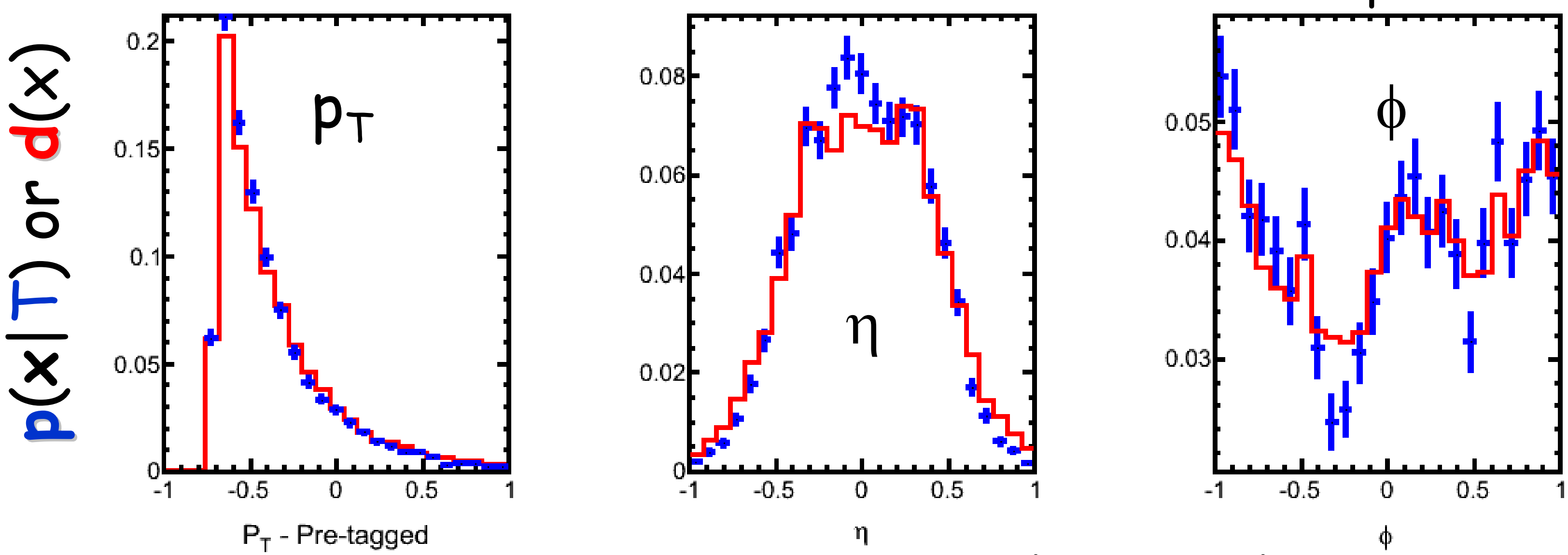
We are often interested in

Pr(Tagged|Jet Variables)

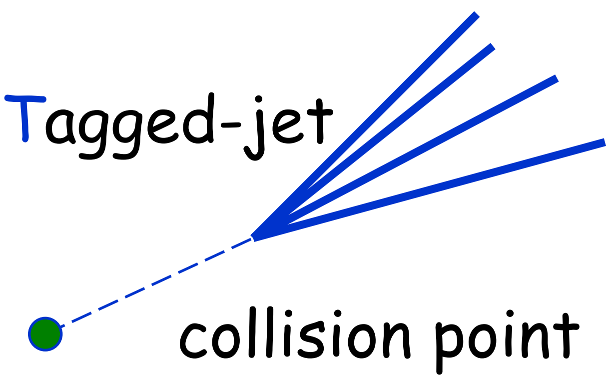
SOS '08

KDE Example: b-Tagging

DØ experiment



$$p(T|x) = p(x|T)\, p(T) \,/\, d(x)$$

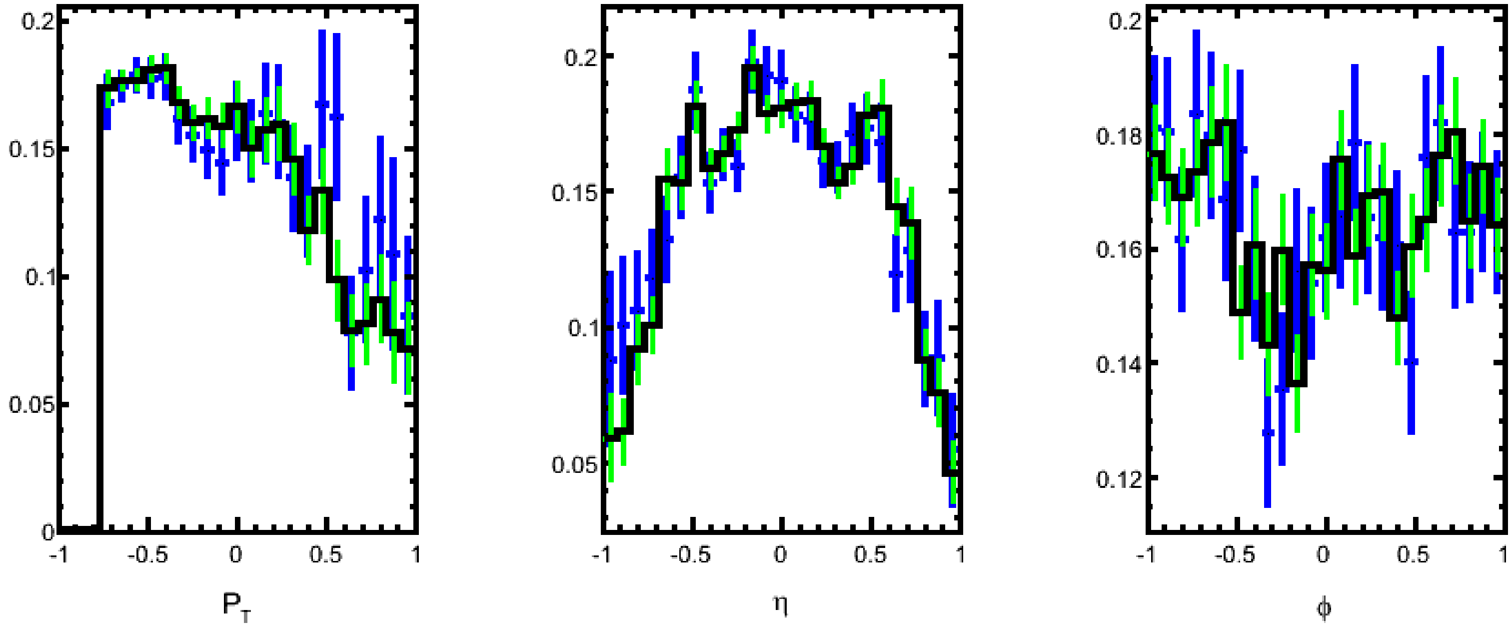


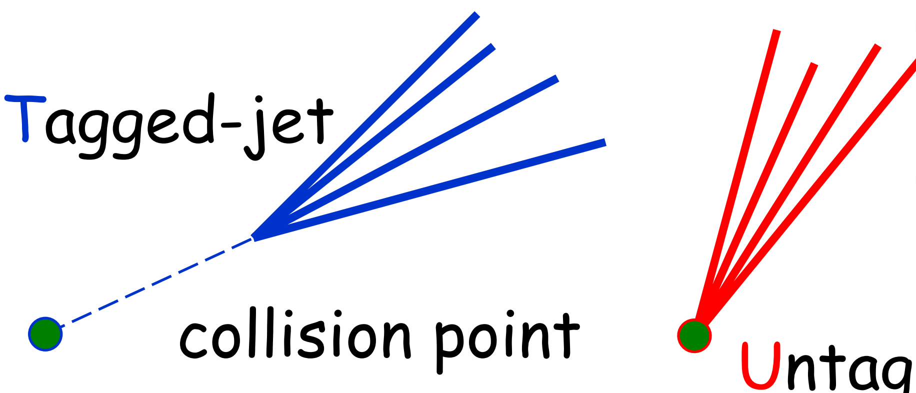
$$d(x) \quad = p(x|T)\, p(T) + p(x|U)\, p(U)$$

$$x = (P_T, \eta, \phi)$$

(red curve is d(x))

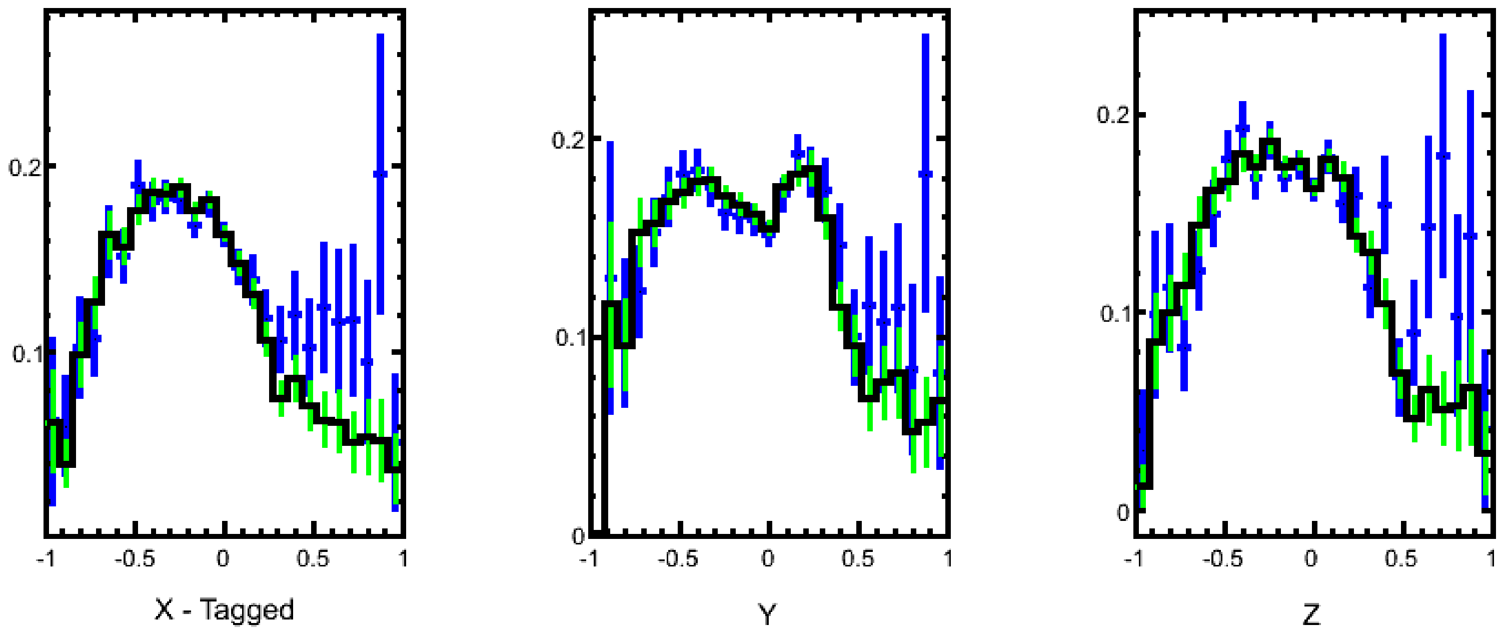
KDE Example: b-Tagging

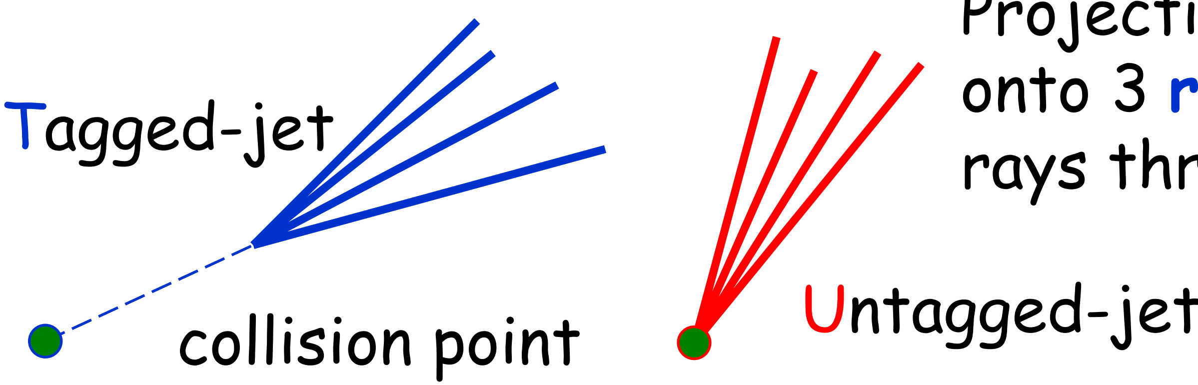




Projections of KDE of p(T|**x**) (black curve) onto the P_T, η and ϕ axes. **Blue points**: ratio of **blue** to **red** histograms (see previous slide)

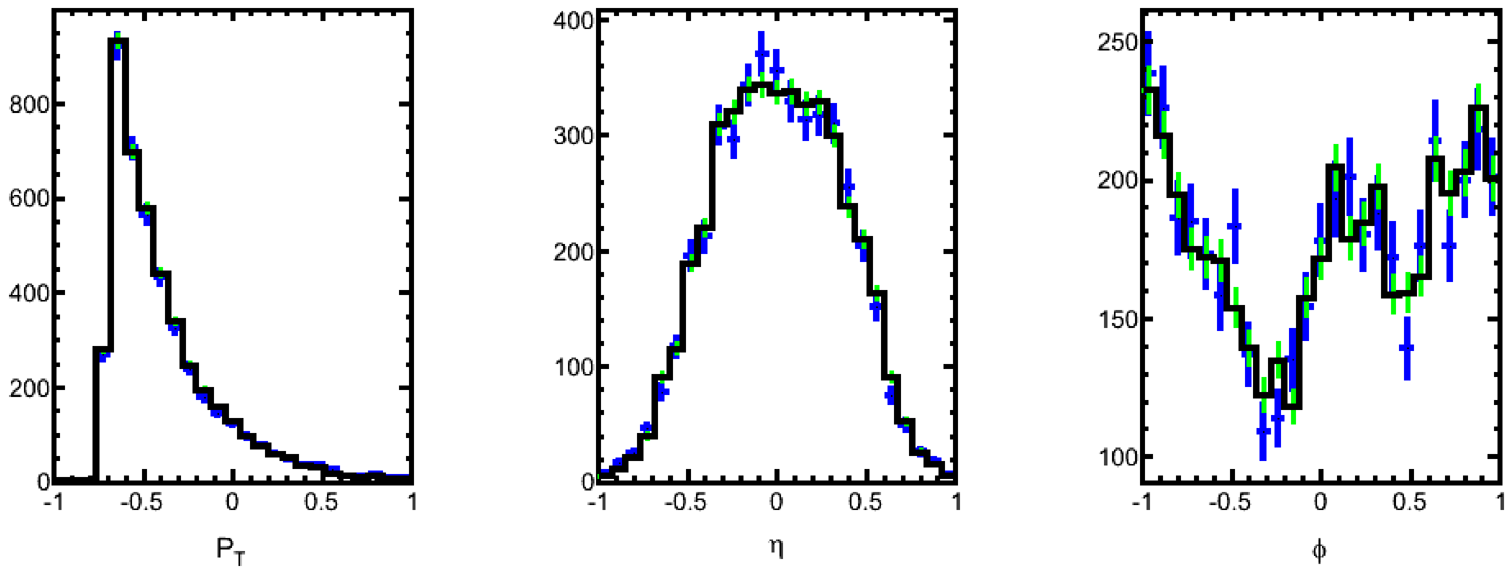
KDE Example: b-Tagging

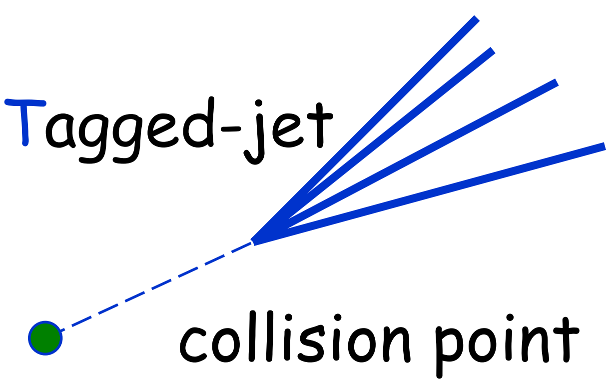


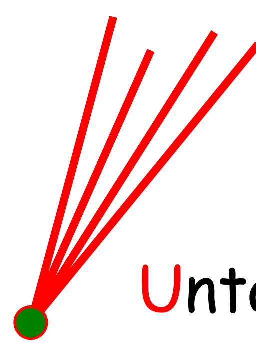


Projections of KDE of p(T|**x**) onto 3 **randomly** chosen rays through the origin.

KDE Example: b-Tagging



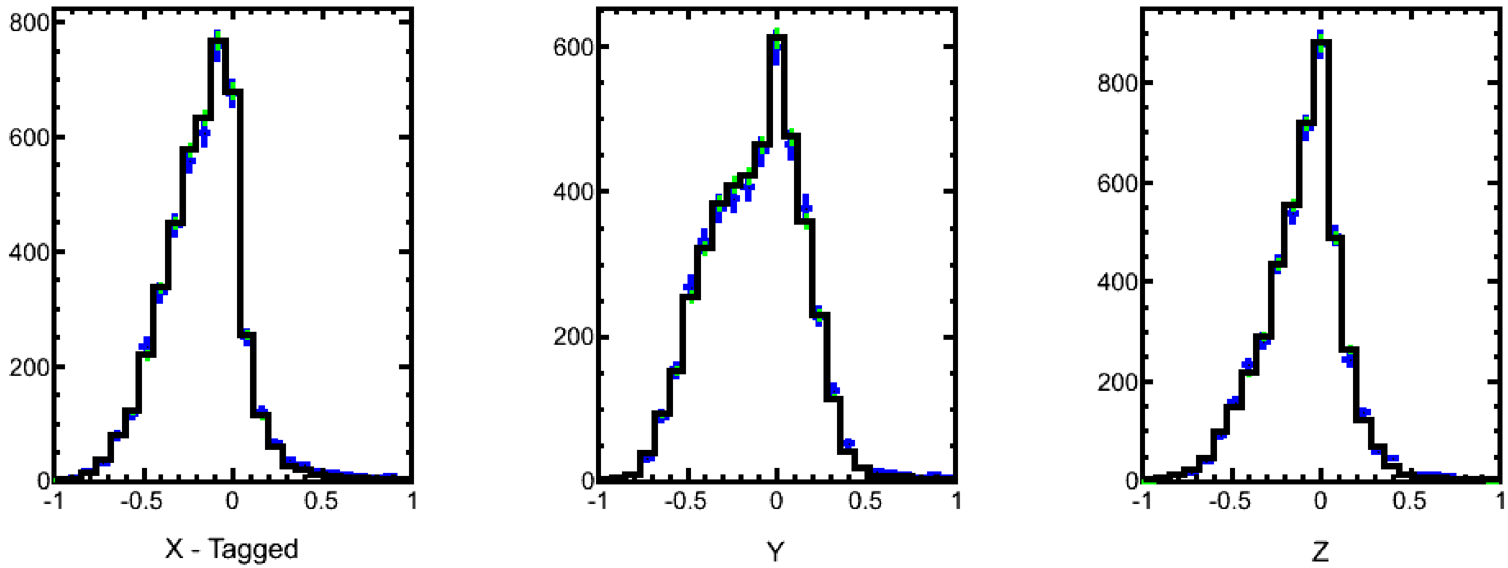




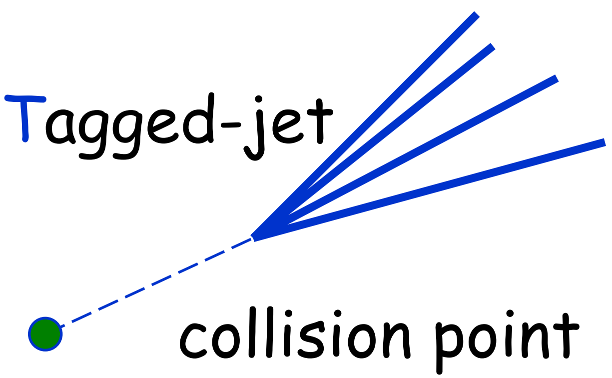
Untagged-jet

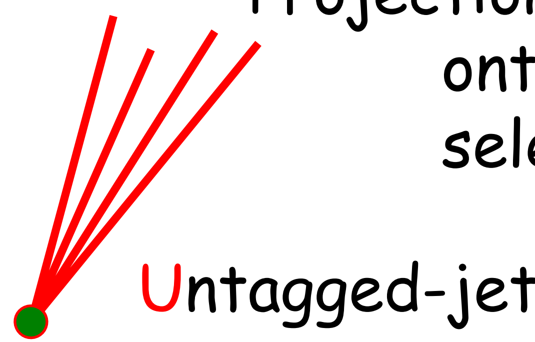
Projections of data weighted by p(T|**x**). Recovers tagged density p(**x**|T).

KDE Example: b-Tagging



Projections of weighted data onto the 3 **randomly** selected rays through the origin





Kernel Density Estimation

Practical Issues

- The choice of bandwidth parameters is crucial.
- In regions where the density of points is low, the kernels will tend to be too far apart.
- A sharp boundary is difficult to model.
- Every evaluation of the KDE requires the evaluation of N, d-dimensional, kernels. If N is large this requires a lot of computation.

Bayesian Neural Networks

Bayesian Neural Networks

Given

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

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

of N training examples and the likelihood function

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

Find

a function $n(\mathbf{x})$ that approximates $D(\mathbf{x})$

Bayesian Neural Networks

For classification, (one form of) the likelihood for the training data is

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

where $y = 0$ for background events

$y = 1$ for signal events

Bayesian Neural Networks

Procedure: Compute

$$p(\mathbf{w}|\mathbf{D}) = p(\mathbf{y}|\mathbf{x},\mathbf{w})\, p(\mathbf{w}) \,/\, \text{const.}$$

using functions of the form

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

from a very large function class and estimate $D(\mathbf{x})$ using

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

The function $n(\mathbf{x})$ is a **Bayesian neural network** (BNN)

Bayesian Neural Networks

Questions:

1. Do sufficiently flexible functions $f(x, \mathbf{w})$ exist?

2. Is there a practical way to do the integral?

Answer 1: Yes!

Hilbert's 13th problem:

Prove that, in general, the following is **impossible**

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

In 1957, Kolmogorov proved the **contrary**: A function $f:R^n \to R$ can be represented as follows

$f(x_1,\ldots,x_n) = \sum_{i=1}^{2n+1} Q_i(\, \sum_{j=1}^{n} G_{ij}(x_j)\,)$

where G_{ij} are independent of f(.)

See Scwindling's talk this afternoon for examples of such functions

Answer 2: Yes!

Computational Method

Generate a sample of N points {w} from the density $p(w|D)$, and average over the last **M** of them.

Do this using methods of statistical mechanics. Generate "states" (p, w) with probability

$\sim \exp(-\beta H)$,

where the "Hamiltonian", H, is

$H = T + V$,

with $T(p) = p^2$ and $V(w) = \ln p(w|D)$

Example 1

Software

Flexible Bayesian Modeling, Radford Neal
http://www.cs.utoronto.ca/~radford/fbm.software.html

Example 1: 1-D

Signal

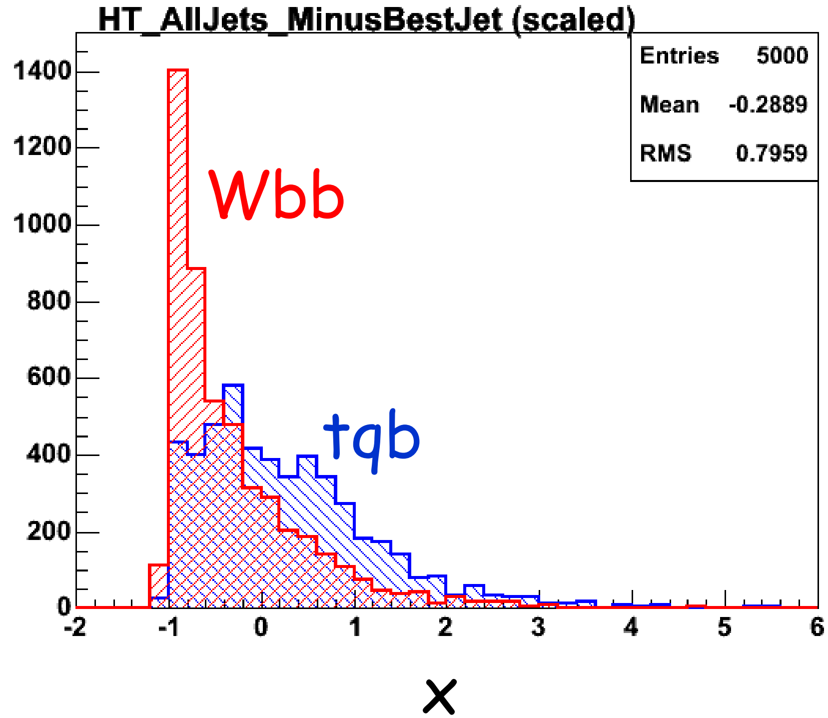
- p+pbar -> **t q b**

Background

- p+pbar -> **W b b**

Function class

- (1, 15, 1)



MCMC

- 500 tqb + Wbb events
- Use last 20 points in a chain of 10,000, skipping every 20th

Example 1: 1-D

Dots

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

H_S, H_B, 1-D histograms

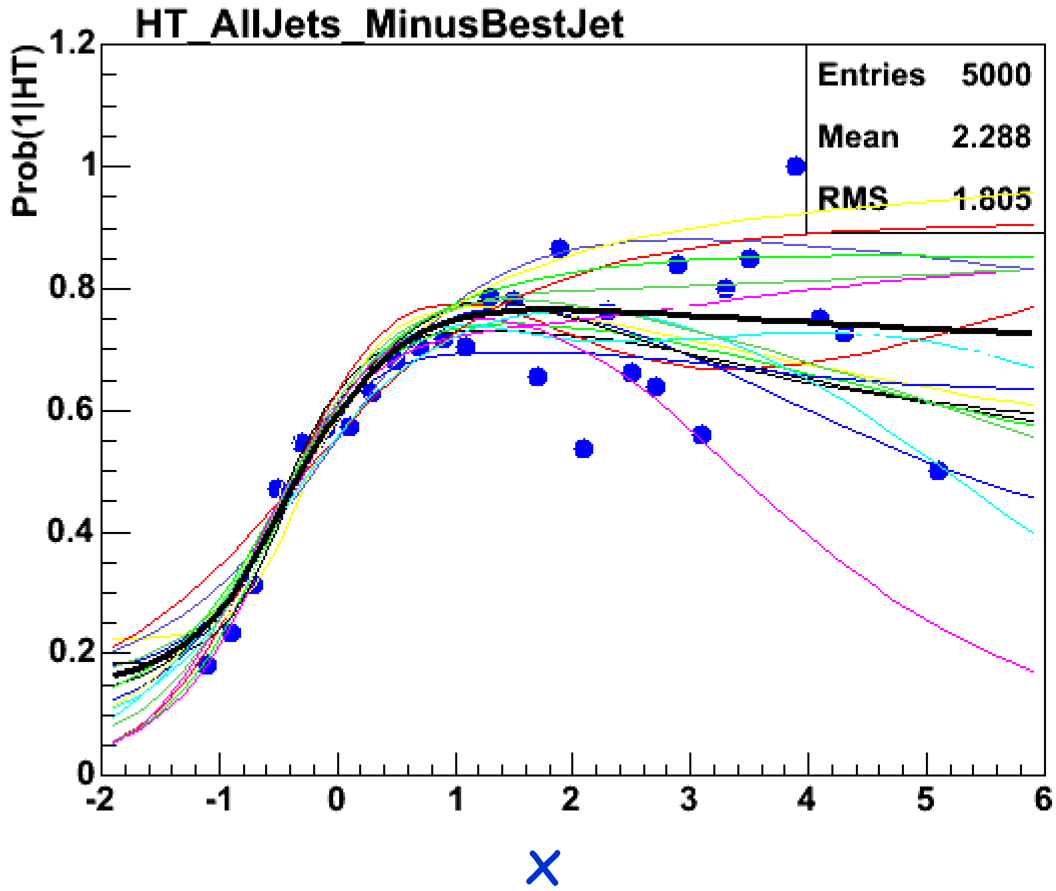
Curves

Individual functions

$n(x, w_k)$

Black curve

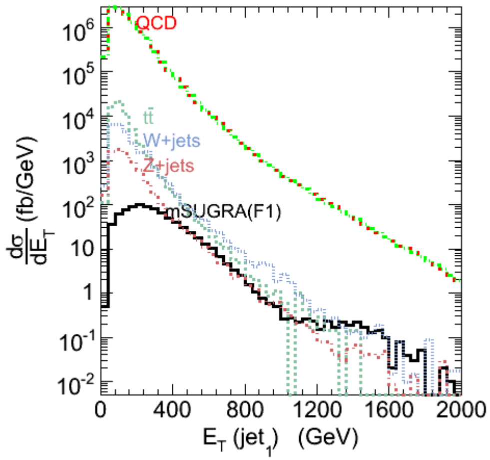
$n(x) = E_w[n(x, w)]$

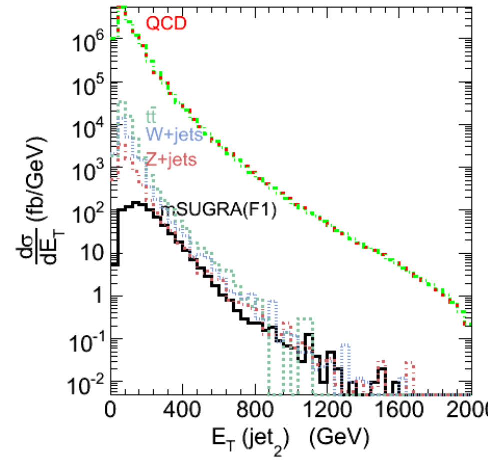


Example 2

Example 2: 14-D

CMS experiment



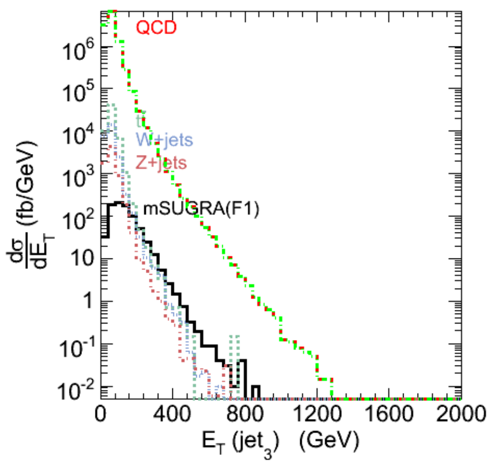


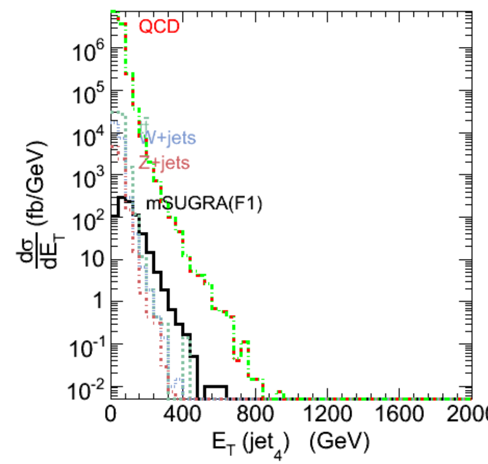
Transverse momentum spectra

SUSY signal: black curve

Signal:Noise

1:25000

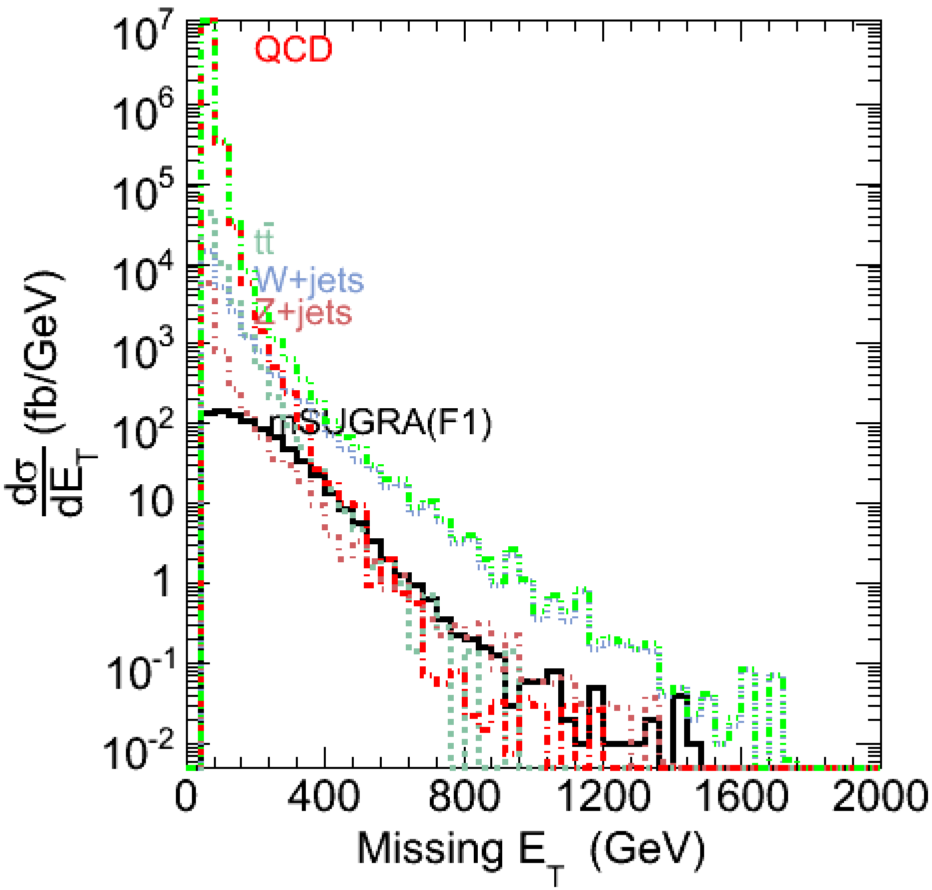




Example 2: 14-D

Missing transverse momentum spectrum

(caused by escape of neutrinos and SUSY particles)



Variables, **x**:

$4 \times (E_T, \eta, \phi)$

$+ \quad (E_T, \phi)$

dim(**x**) = **14**

Example 2: 14-D

Signal

250 p+p -> gluino, gluino (mSUGRA) events

Background

250 p+p -> top, anti-top events

Function class

(14, 40, 1) (dim(w) = 641) !!! ☹

MCMC

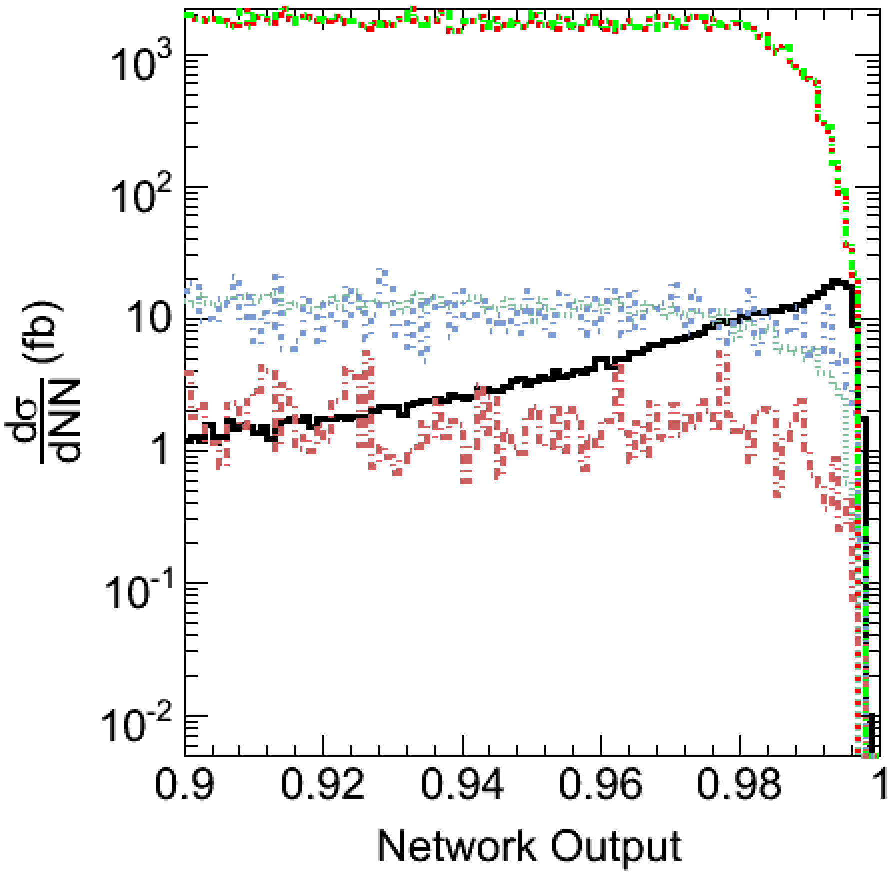
Use last 100 points (that is, networks) in a Markov chain of 10,000, skipping every 20.

Example 2: 14-D

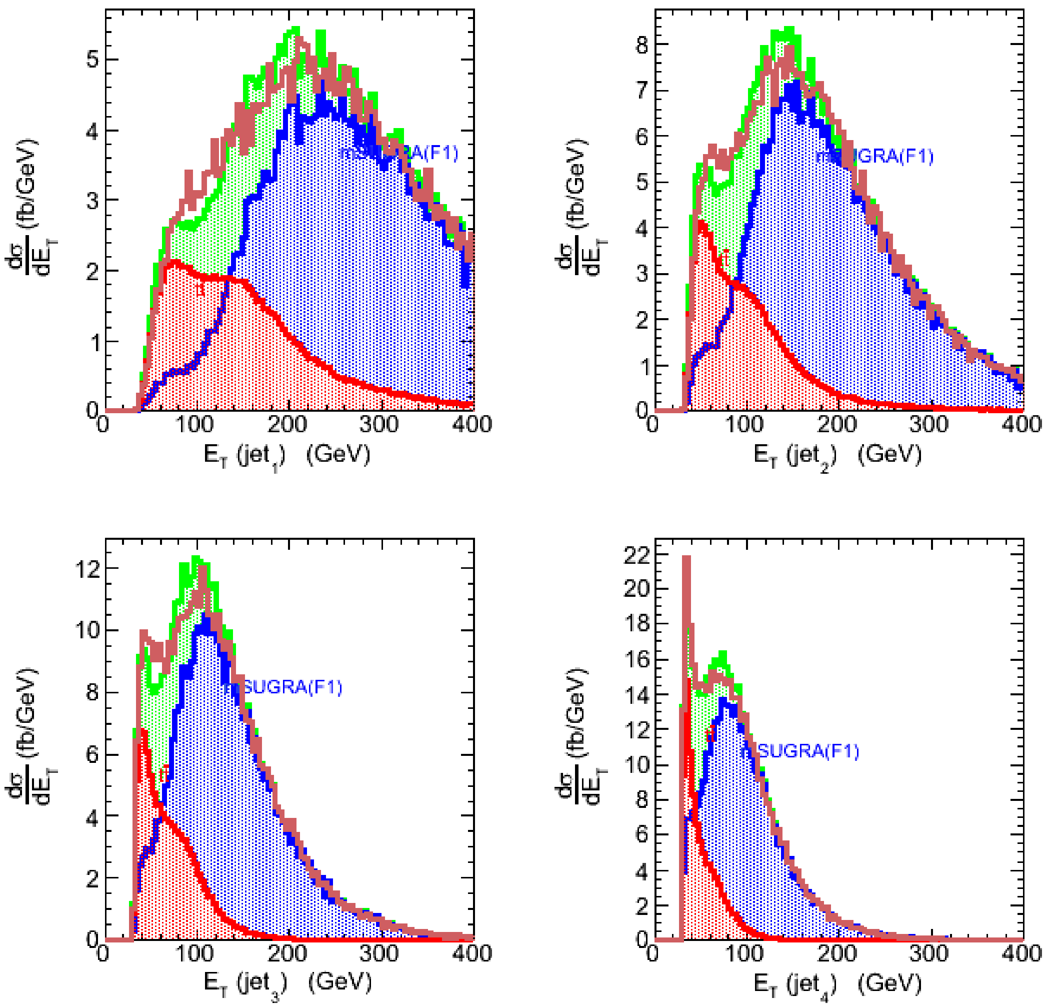
Distribution
beyond n(**x**) > 0.9

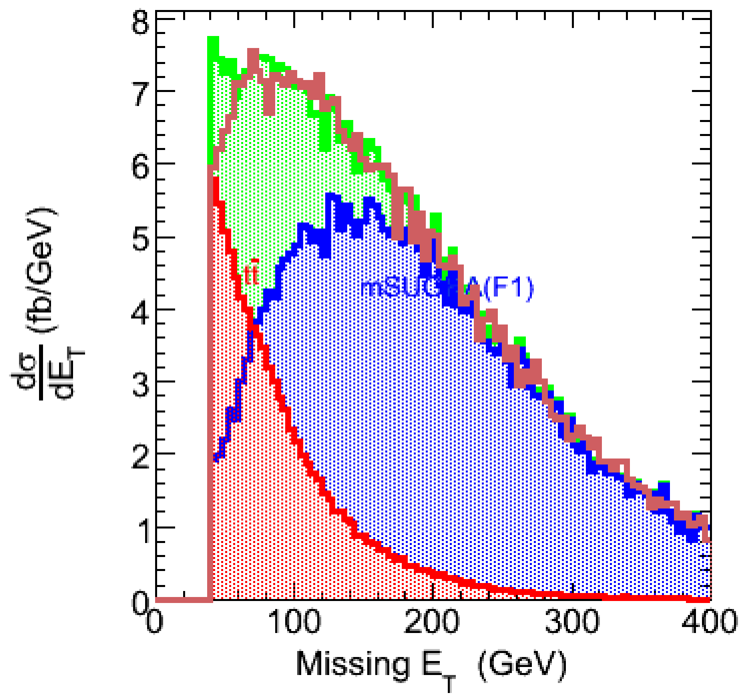
Assuming L = 10 fb^{-1}

Cut	S	B	S/√B
0.90	5×10^3	2×10^6	3.5
0.95	4×10^3	7×10^5	4.7
0.99	1×10^3	2×10^4	7.0



Example 2: 14-D





Verification plots

ça marche! ☺

Issues

- How should one choose the function class?
- How should one verify that a d-dimensional density is well-modeled?
- How should one take into account model uncertainty?
- How should one compute data compression efficiency?

efficiency = **Info**(after compression)/**Info**(before)

Summary

- The function $D(x) = s(x) / [s(x) + b(x)]$ can be applied to many aspects of data analysis
- Moreover, many practical methods, and tools, are available to approximate it
- However, no one method is guaranteed to give the best approximation in all circumstances. So it is good to experiment with a few of them using tools such as **TMVA** or **StatPatternRecognition**