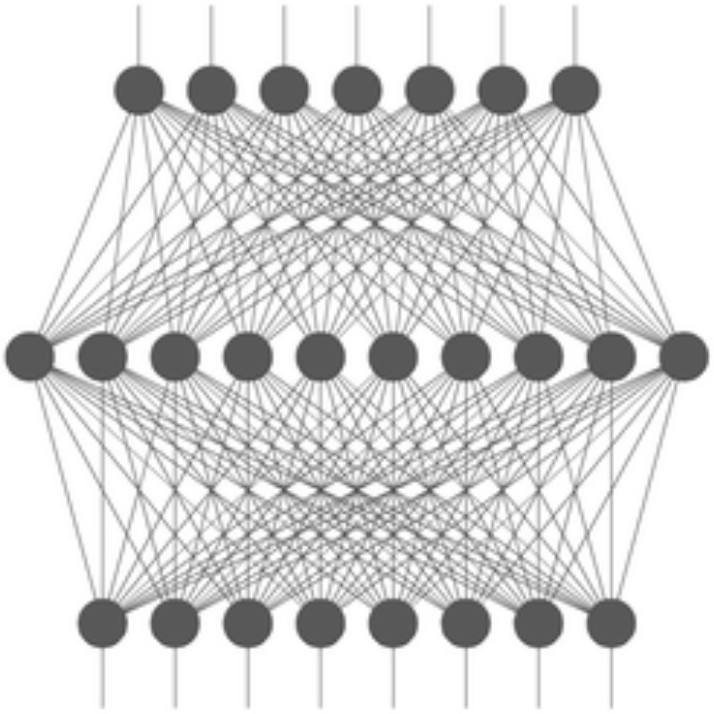
Karlsruhe Institute of Technology

Neural Networks and NeuroBayes® in High Energy Physics

Lectures at the School of Statistics, Autrans, 2010

Prof. Dr. Michael Feindt

IEKP KCETA Karlsruhe Institute of Technology KIT



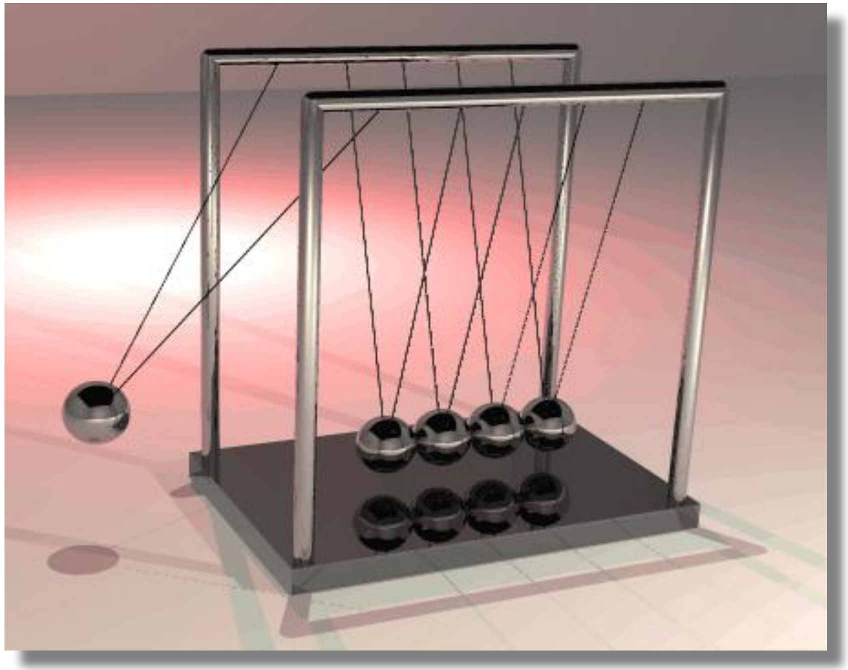
Predictable

The result of simple classical physics processes is exactly predictable

(one cause leads to one definite unique result, determinism)

Examples:

pendulum, planets, billard, electromagnetism…



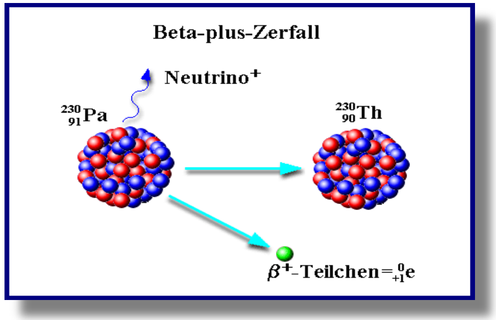
Unpredictable

Purely random processes are not predictable at all
(even if the initial conditions are completely known!)

Examples:

Lottery
(Too many tiny influences and branchings, deterministic chaos)

radioactive decay
(quantum mechanics)

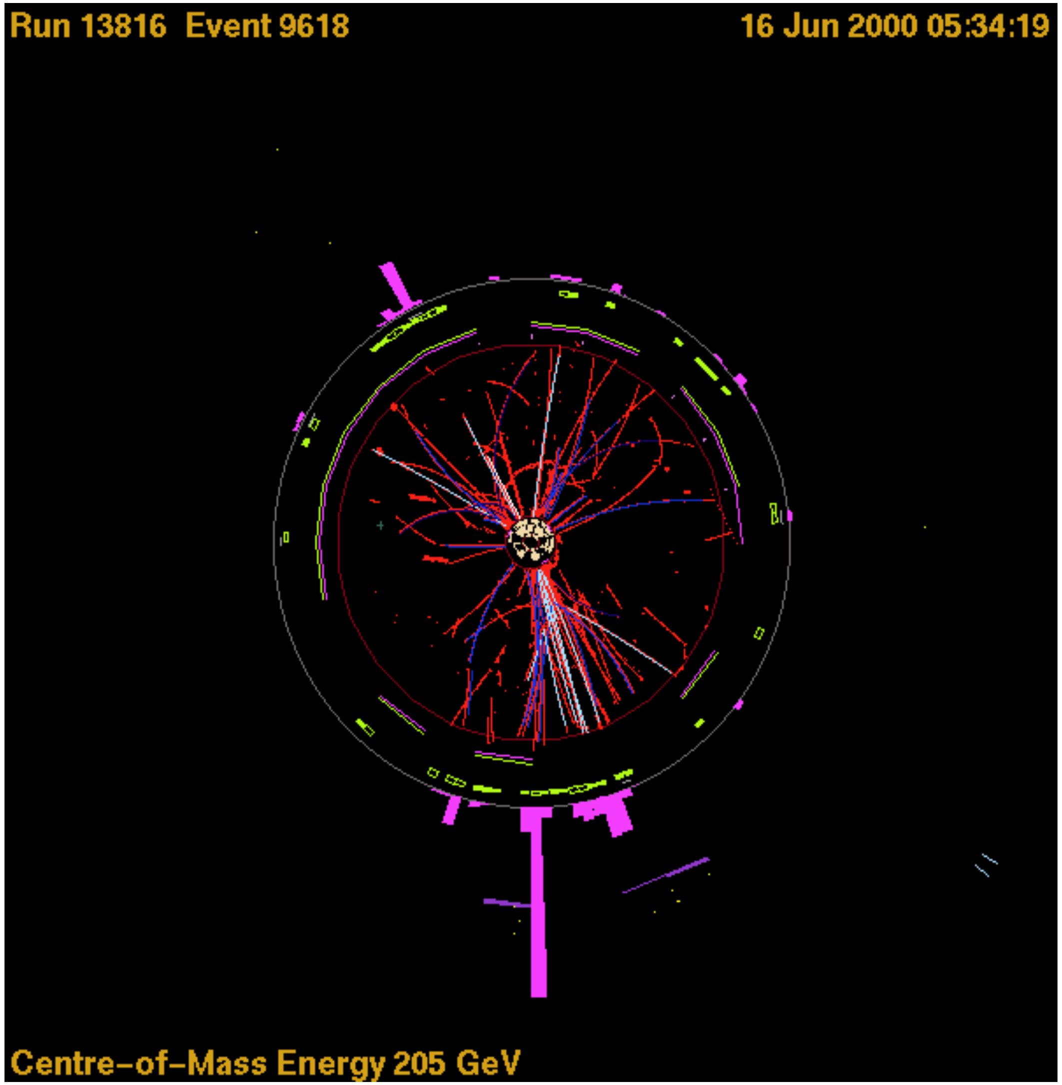


electronic noise

Quantum Mechanics:
In every collision
something else
happens!

Experiments:
Observe mean
values, distributions,
correlations,
determine parameters
(mean lifetime, spin,
parity etc) from that.

OPAL experiment at LEP



Probability

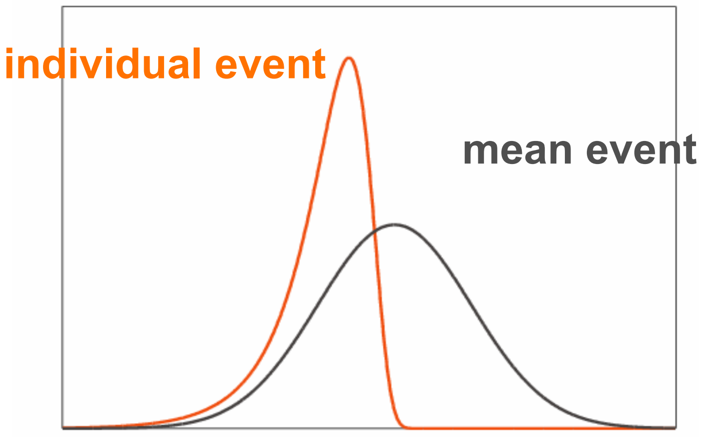
Many systems in nature and life:
Mixture of predictable and unpredictable
(quasi-) random or chaotic components.

→Probability statements, statistics.

NeuroBayes core technology:

Extraction of a predictable component from empirical data (or Monte Carlo simulations)

Statistically relevant predictions for future events



Individualisisation of probability statements:

conditional probabilities: f(t|x), dependent on individual event with properties x instead of general (a priori) probability f(t)

Bayes' Theorem (1)

Conditional Probabilities:

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Because of $P(A \cap B) = P(B \cap A)$ it follows that

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)}$$

Bayes´ Theorem

Bayes' Theorem (2)

Extremely important due to the interpretation A=theory B=data

Likelihood

Prior

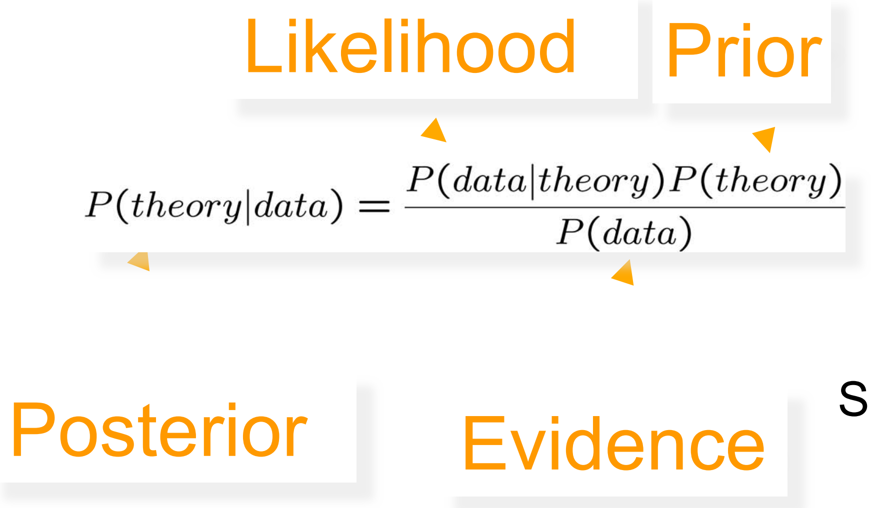
$$P(theory|data) = \frac{P(data|theory)P(theory)}{P(data)}$$

Posterior

Evidence

Bayesian vs. classical statistics

Classical statistics is just a special case of Bayesian statistics:



Likelihood

Prior

$$P(theory|data) = \frac{P(data|theory)P(theory)}{P(data)}$$

Posterior

Evidence

Maximisation of likelihood instead of a posteriori probability means:

Implicit assumption that prior probability is flatly distributed, i.e. each value has same probability.

Sounds reasonable, but is in general wrong!
Does not mean that one knows nothing!

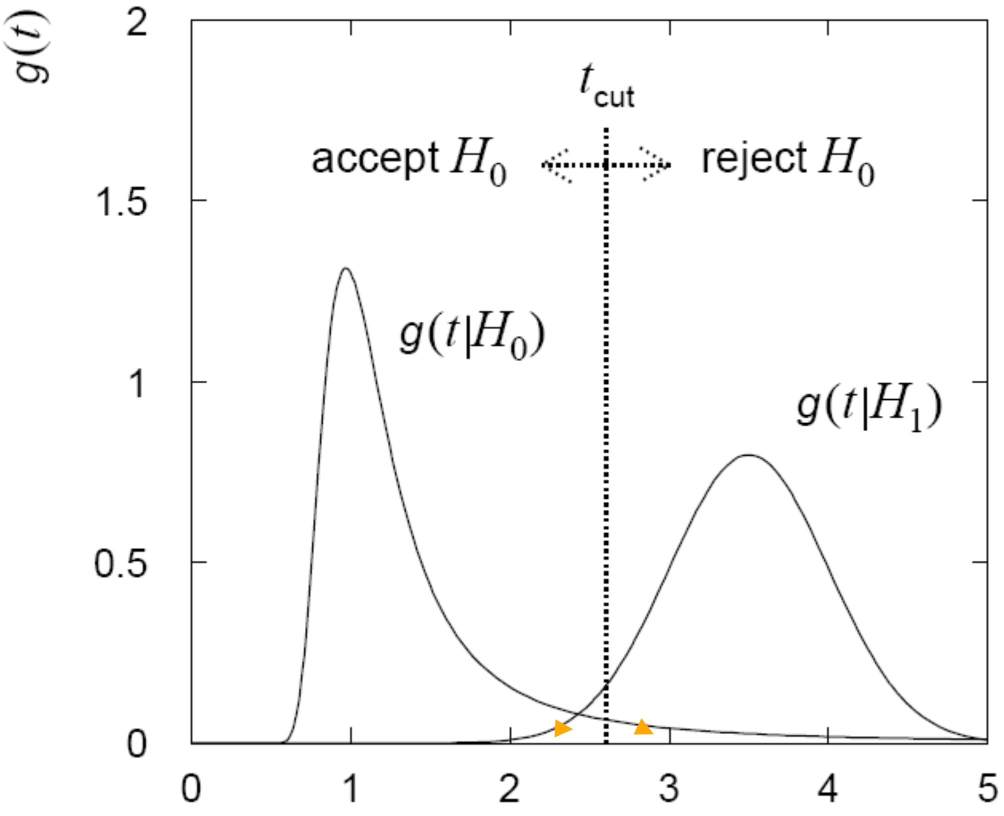
Classification == Hypothesis testing

Cut in a 1-dimensional real test-statistic which is correlated to the probability of hypothesis H0:

Accept hypothesis H0,
if t<t(cut)

Error of 1. kind:
P1(true hypothesis will be rejected)

Error of 2. kind:
P2(wrong hypothesis is accepted)



Hypothesis testing

Important quantities for all classification tasks

$$\boxed{\text{Efficiency:}} \quad \varepsilon = \frac{P(\text{selected and true})}{P(\text{true})} = 1 - P1$$

$$\boxed{\text{Purity:}} \quad \mathcal{P} = \frac{P(\text{selected and true})}{P(\text{selected})} = 1 - P2$$

Also use $\boxed{\text{dilution}}$ $D = 2\mathcal{P} - 1$.

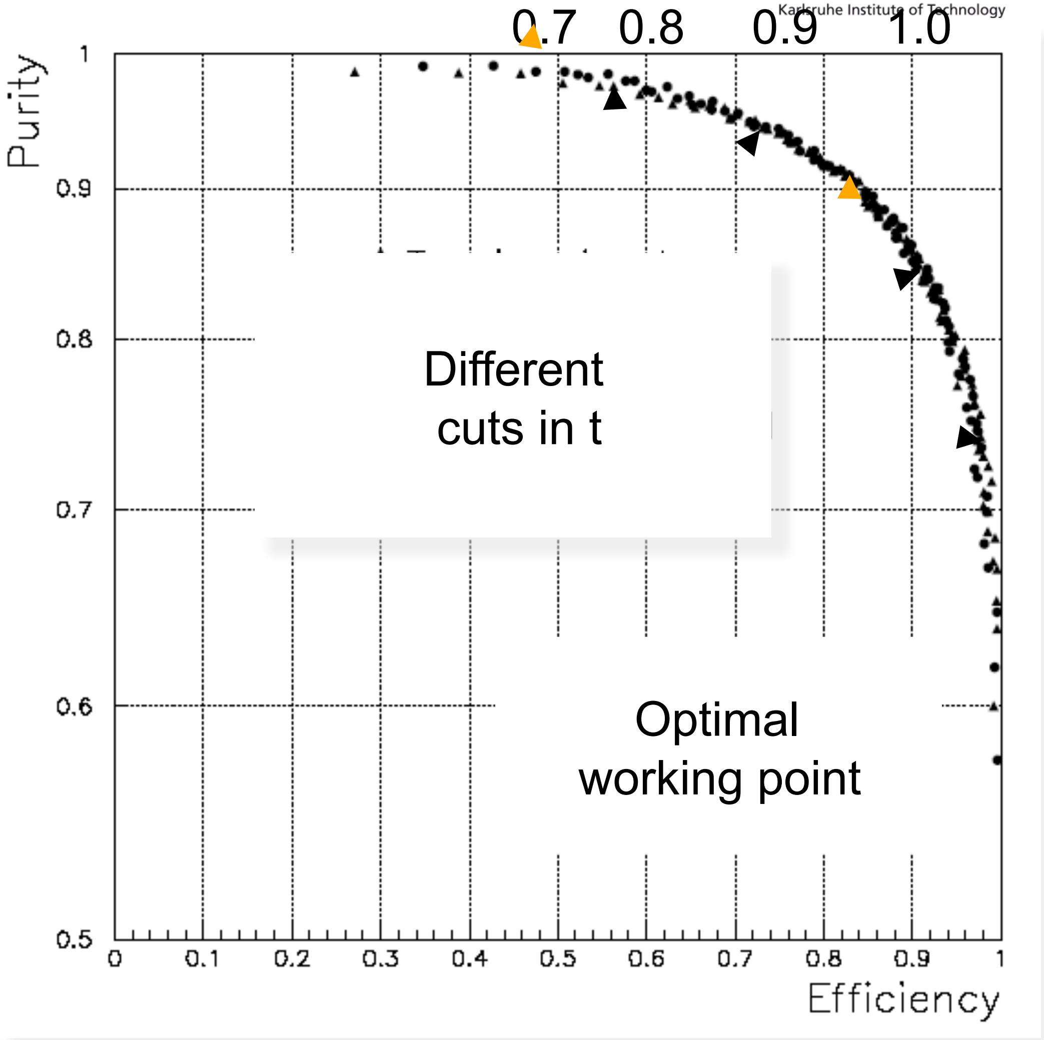
Choice of working point (i.e. cut-value) according to application.

Good test statistic maximizes area in $\mathcal{P} - \varepsilon$ plane.

Hypothesis testing

A statistical method is the better the nearer it reaches the point (1,1) in the purity-efficiency-plot

Optimal choice of working point according to particular task: How does the total error of the analysis scale with ε und P?

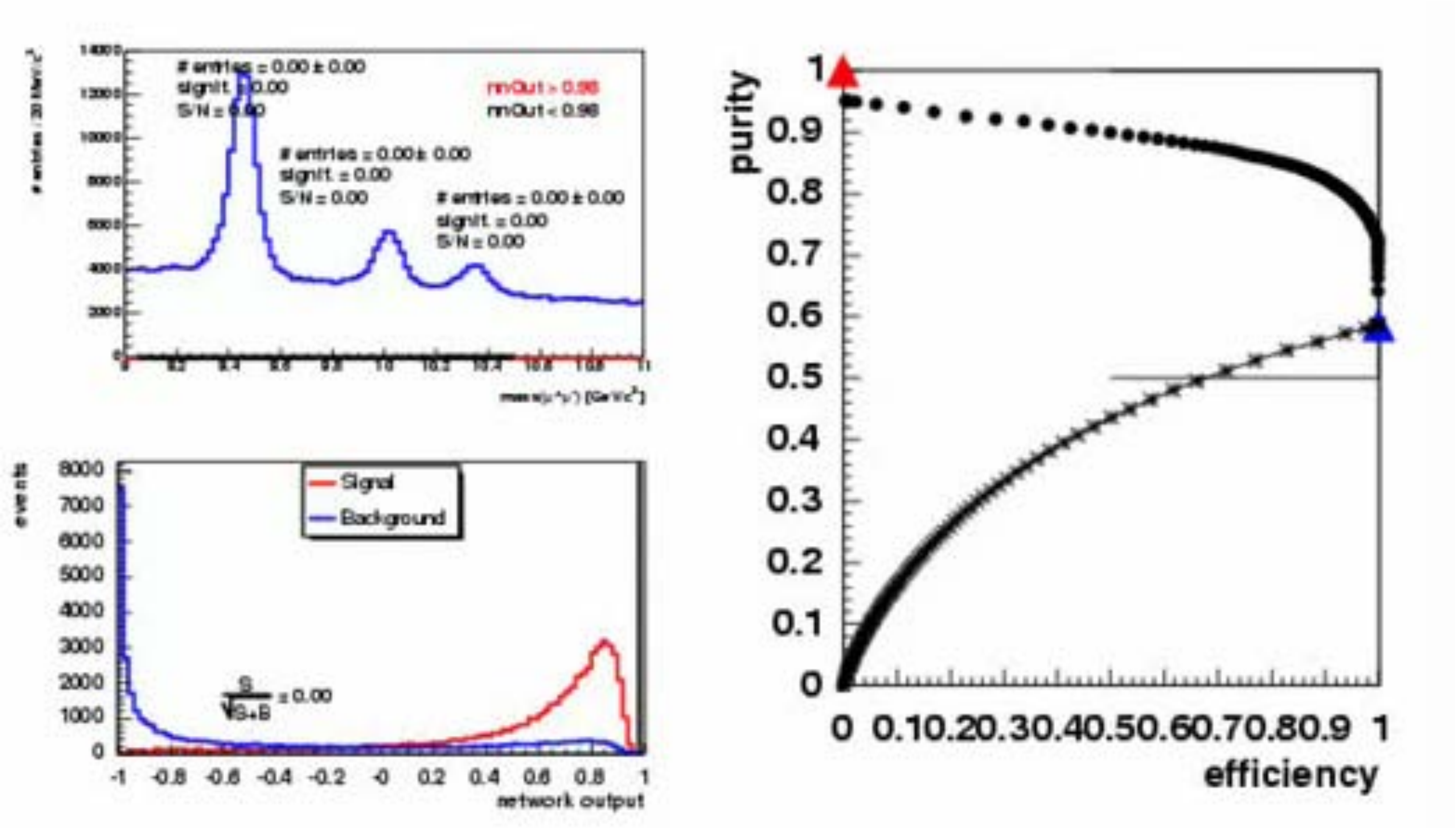


Different cuts in t

Optimal working point

Flavour-tagging in oscillation analyses: $Signif. \propto \sqrt{\varepsilon} \cdot (2P-1)$

Determining the working point
(scan through cuts on network output)



Construction of a test statistic:
How to make 100 dimensions one real number...

Sequential cuts: simple

Linear separation of correlated input variables by hyperplane in n-dim. space:

Fisher-discriminant: maximises separation of expectation values of two classes in units of the sum of their variances

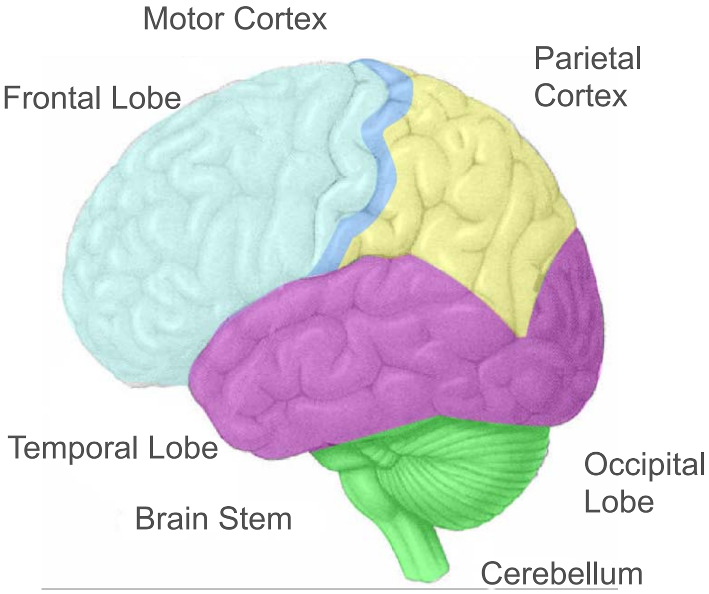
Neyman-Pearson-Lemma: n uncorrelated variables are separated optimally by the likelihood ratio:

$$t = \sum_{i}^{n} \frac{L_{H_0}(x_i)}{L_{H_1}(x_i)}$$

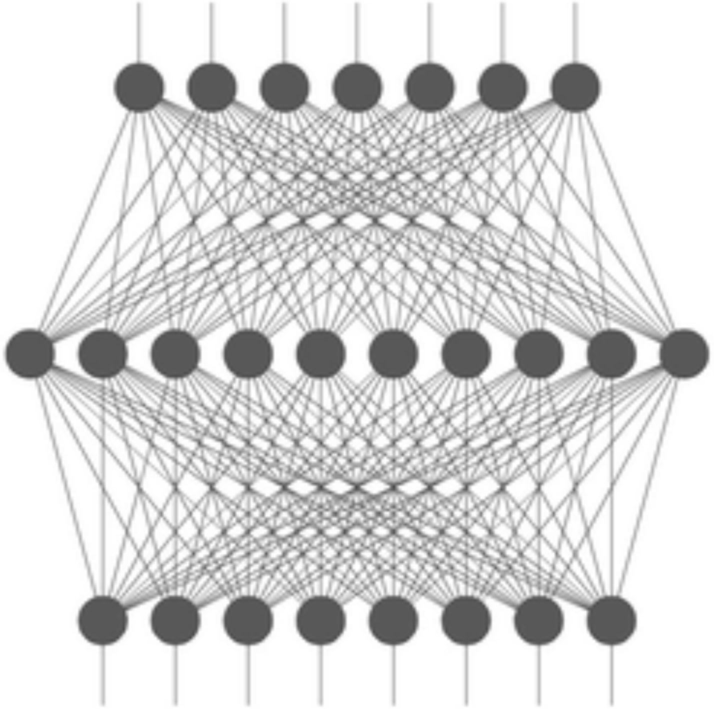
Or: neural networks (or support vector machines...)

Neural networks

Neural networks:
Self learning procedures, copied from nature



Neural networks



The information
(the knowledge, the expertise)
is coded in the connections
between the neurons

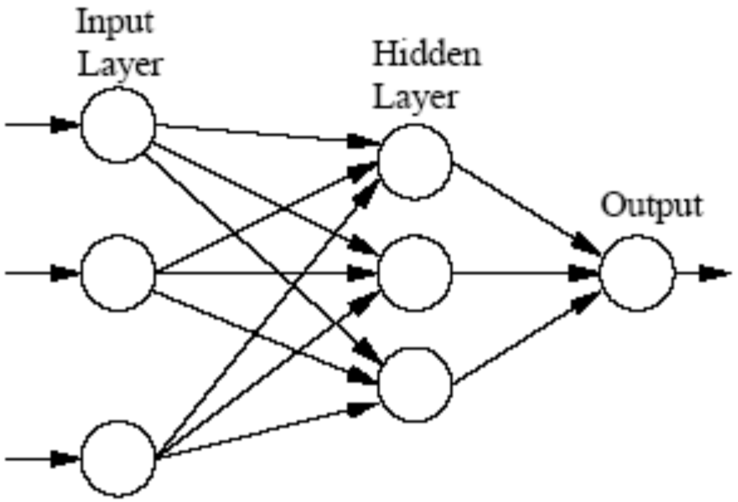
Each neuron performs fuzzy decisions

A neural network can learn from examples

Human brain: about 100 billion (10^{11}) neurons
about 100 trillion (10^{14}) connections

Neural Network

basic functions



The output of node j in layer n is calculated from weighted sum of outputs in layer $n-1$:

$$x_j^{(n)} = f(\sum_i w_{i,j}^{(n)} x_i^{(n-1)} + w_{0,j}^{(n)})$$

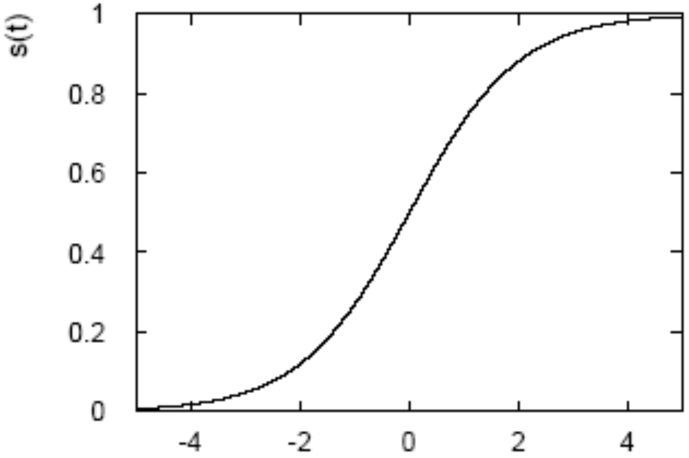
Each connection has associated a weight $w_{i,j}^{(n)}$, each node a bias $w_{0,j}^{(n)}$.

Neural network transfer functions

A non-linear monotonuous transfer function $f(x)$ is applied at the output of each node, e.g. the sigmoid function:

$$f(x) = \frac{1}{1 + exp(-x)}$$

It maps the intervall $(-\infty, \infty)$ to the compact $(0, 1)$.



Neural network training

Training is the minimisation process of a loss function, during that the network weights are changed such that the deviation of the wanted output for a set of input vectors is minimisesd.

Possible loss functions:
Sum of quadratic deviations
or entropy (maximum likelihood)

Backpropagation (Rumelhardt et al. 1986):
Calculate gradient backwards by applying chain rule
Optimise using gradient descent method. Step size??

Neural network training

Difficulty: find global minimum of highly non-linear function in high (~ >100) dimensional space.

Imagine task to find deepest valley in the Alps (just 2 dimensions)

Easy to find the next local minimum...

but globally...
...impossible!

Naïve neural networks and criticizm

We‘ve tried that but it didn‘t give good results

- Stuck in local minimum
- Learning not robust

We‘ve tried that but it was worse than our 100 person-years analytical high tech algorithm

- Selected too naive input variables
- Use your fancy algorithm as INPUT !

We‘ve tried that but the predictions were wrong

- Overtraining: the net learned statistical fluctuations

Yeah but how can you estimate systematic errors?

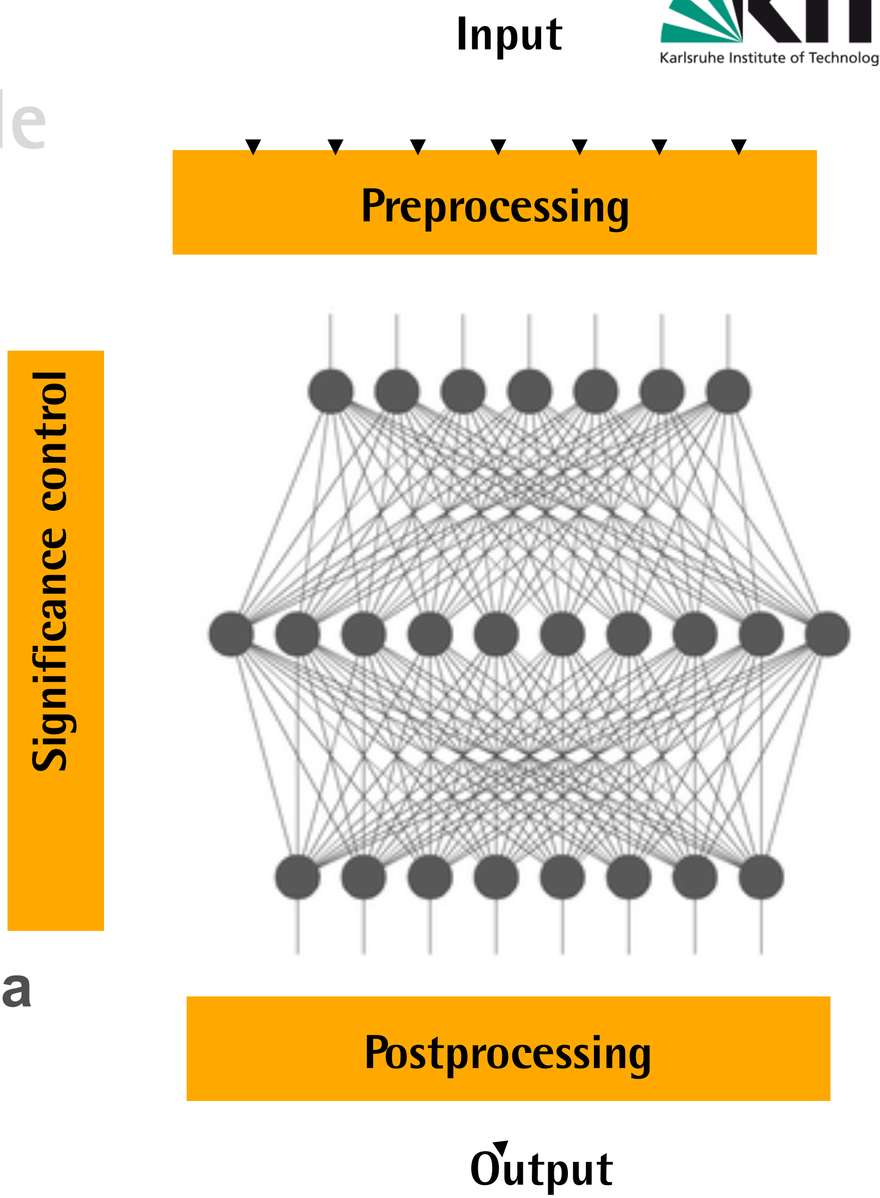
- How can you with cuts when variables are correlated?
- Tests on data, data/MC agreement possible and done.

Address all these topics and build a professional robust and flexible neural network package for physics, insurance, bank and industry applications: NeuroBayes®

NeuroBayes® principle

NeuroBayes® Teacher:
Learning of complex relationships from existing data bases (e.g. Monte Carlo)

NeuroBayes® Expert:
Prognosis for unknown data



Input

Preprocessing

Significance control

Postprocessing

Output

How it works: training and application

Historic or simulated data

Data set
a = ...
b = ...
c = ...
....
t = ...!

NeuroBayes® Teacher

Expert system

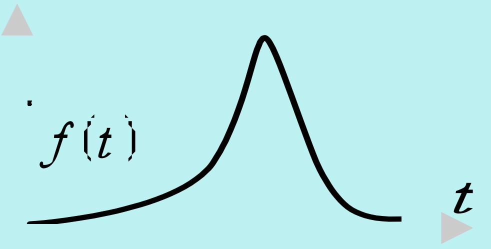
Expertise

Actual (new real) data

Data set
a = ...
b = ...
c = ...
....
t = ?

NeuroBayes® Expert

Probability that hypothesis is correct (classification) or probability density for variable t



NeuroBayes® task 1:
Classifications

Classification:
Binary targets: Each single outcome will be "yes" or "no"
NeuroBayes output is the Bayesian posterior probability that answer is "yes" (given that inclusive rates are the same in training and test sample, otherwise simple transformation necessary).

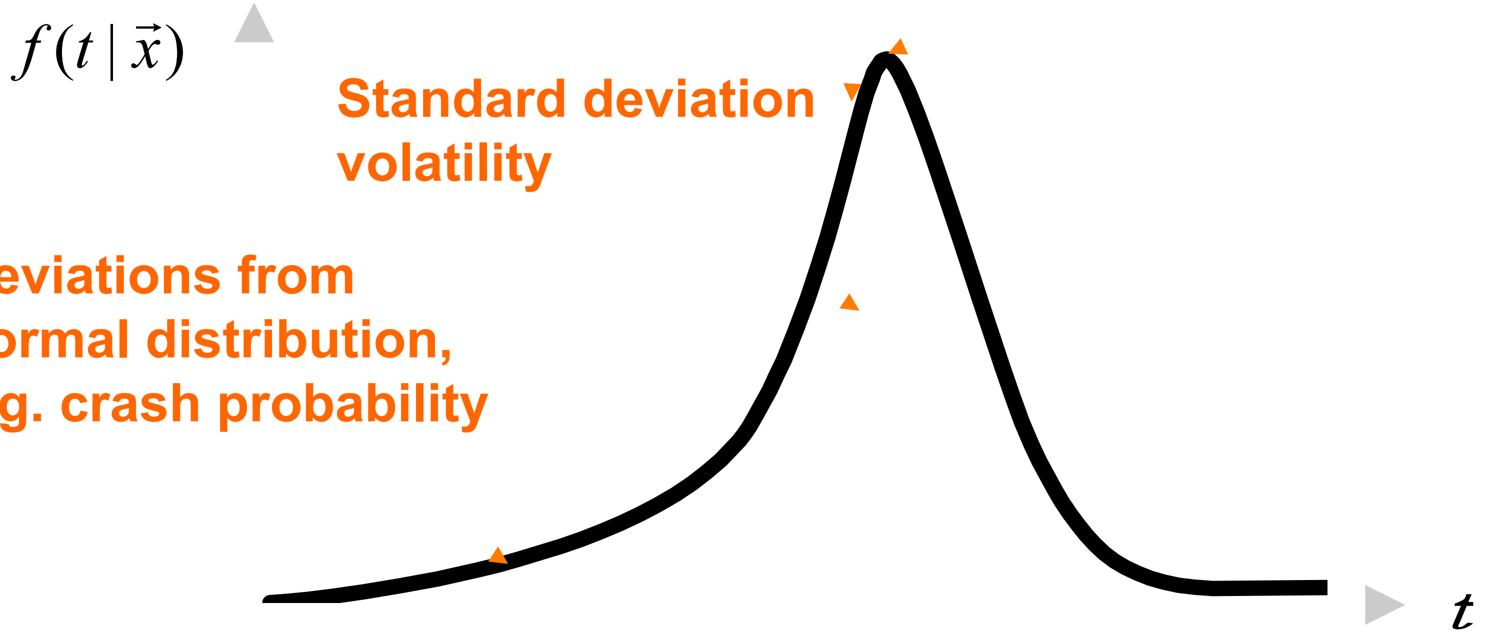
Examples:
> This elementary particle is a K meson.
> This event is a Higgs candidate.
> Germany will become soccer world champion in 2010.
> Customer Meier will have liquidity problems next year.
> This equity price will rise.

NeuroBayes® task 2:
Conditional probability densities

Probability density for real valued targets:
For each possible (real) value a probability (density) is given.
From that all statistical quantities like mean value, median, mode, standard deviation, percentiles etc can be deduced.

Examples:
> Energy of an elementary particle
  (e.g a semileptonically decaying B meson with missing neutrino)
> Q value of a decay
> Lifetime of a decay
> Price change of an equity or option
> Company turnaround or earnings

Prediction of the complete probability distribution – event by event unfolding –



Expectation value

Mode

Standard deviation volatility

Deviations from normal distribution, e.g. crash probability

Conditional probability densities in particle physics

What is the probability density of the true B momentum in this semileptonic B candidate event taken with the CDF II detector

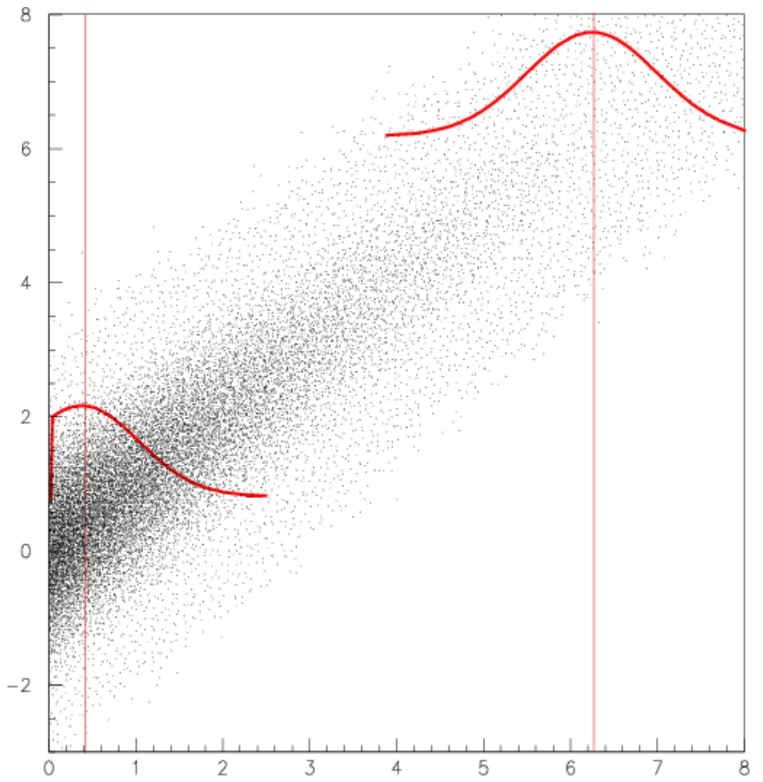
with these n tracks with those momenta and rapidities in the hemisphere, which are forming this secondary vertex with this decay length and probability, this invariant mass and transverse momentum, this lepton information, this missing transverse momentum, this difference in Phi and Theta between momentum sum and vertex topology, etc pp ?

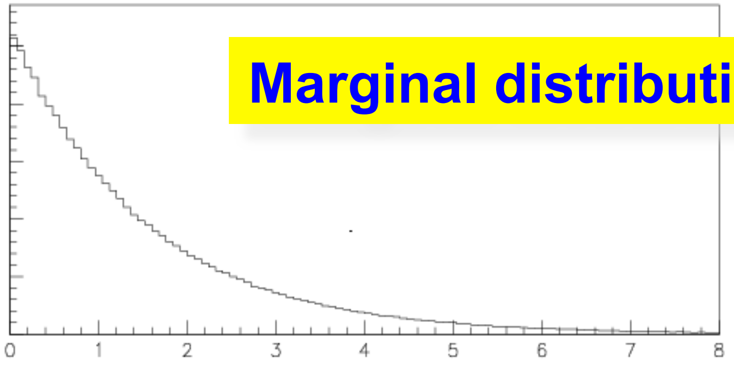
t

$\vec{x}$

$f(t \mid \vec{x})$

Conditional probability densities f(t|x)



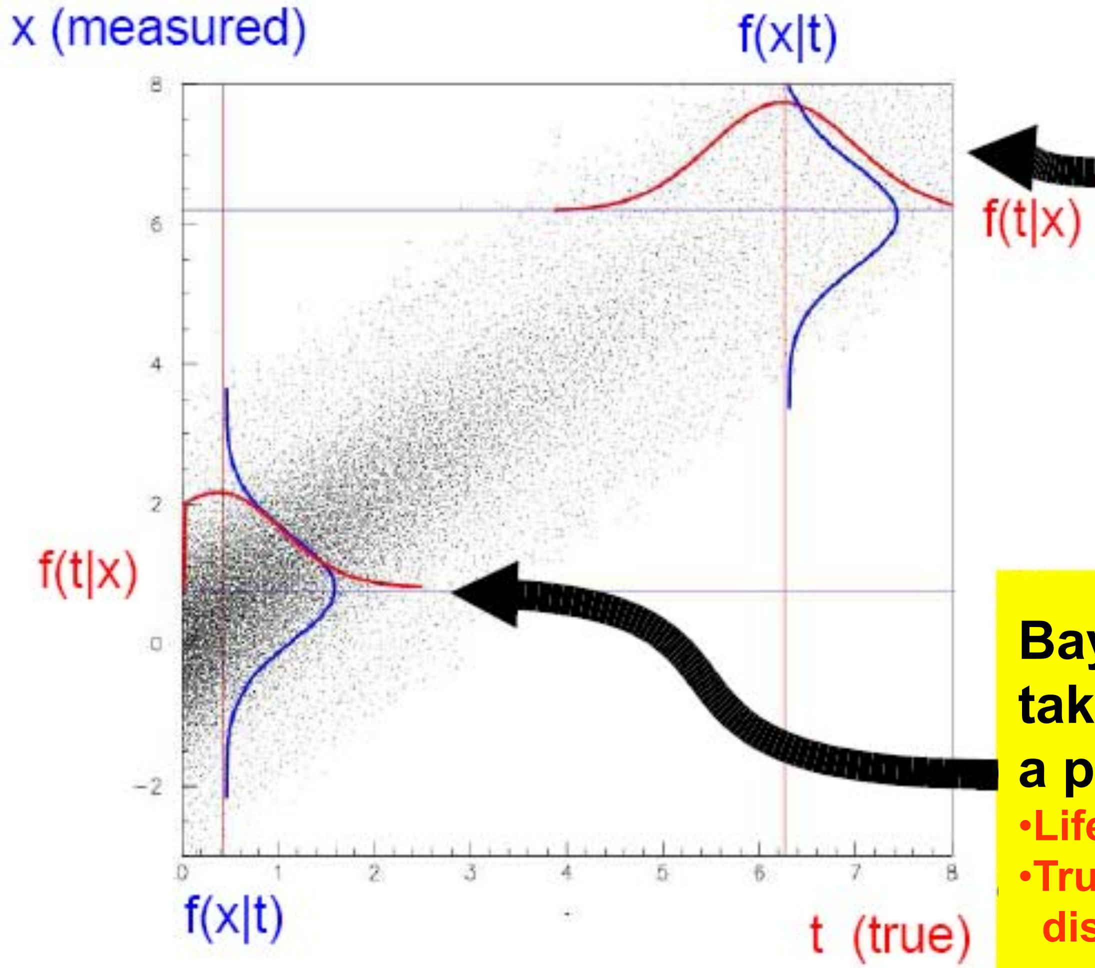


Conditional probability density for a special case x (Bayesian Posterior)

Conditional probability densities f(t|x) are functions of x, but also depend on marginal distribution f(t).

Marginal distribution f(t)

Inclusive distribution (Bayesian Prior)



Classical ansatz:
f(x|t)=f(t|x)
approximately correct
at good resolution
far away from
physical boundaries

Bayesian ansatz:
takes into account
a priori- knowledge f(t):

- **Lifetime never negative**
- **True lifetime exponentially distributed**

The aim:

Aim:
Bayesian estimator $f(t\,|\,\vec{x})$ for a single multidimensional measurement $\vec{x}$.

- Components of $\vec{x}$ may be correlated.
- Components of $\vec{x}$ should be correlated to t or its uncertainty.
- All this should be learned automatically in a robust way from data bases containing Monte-Carlo simulations or historical data.

Note:
Conditional probability density contains much more information than just the mean value, which is determined in a regression analysis.
It also tells us something about the uncertainty and the form of the distribution, in particular non-Gaussian tails.

Main message:

NeuroBayes is a very powerful algorithm

- robust – (unless fooled) does not overtrain, always finds good solution - and fast
- can automatically select significant variables
- output interpretable as Bayesian a posteriori probability
- can train with weights and background subtraction
- has potential to improve many analyses significantly
- in density mode it can be used to improve resolutions (e.g. lifetime in semileptonic B decays)

NeuroBayes is easy to use

- Examples and documentation available
- Good default values for all options →fast start!
- Direct interface to TMVA available
- Introduction into root planned
- To use from C,C++, Fortran, Python etc.
- Two code generators available

<phi-t> NeuroBayes®

> is based on neural 2nd generation algorithms,
  Bayesian regularisation,
  optimised preprocessing with transformations
  and decorrelation of input variables and
  linear correlation to output.

> learns extremely fast due to 2nd order methods and
  0-iteration mode

> is extrmely robust against outliers

> is immune against learning by heart statistical noise

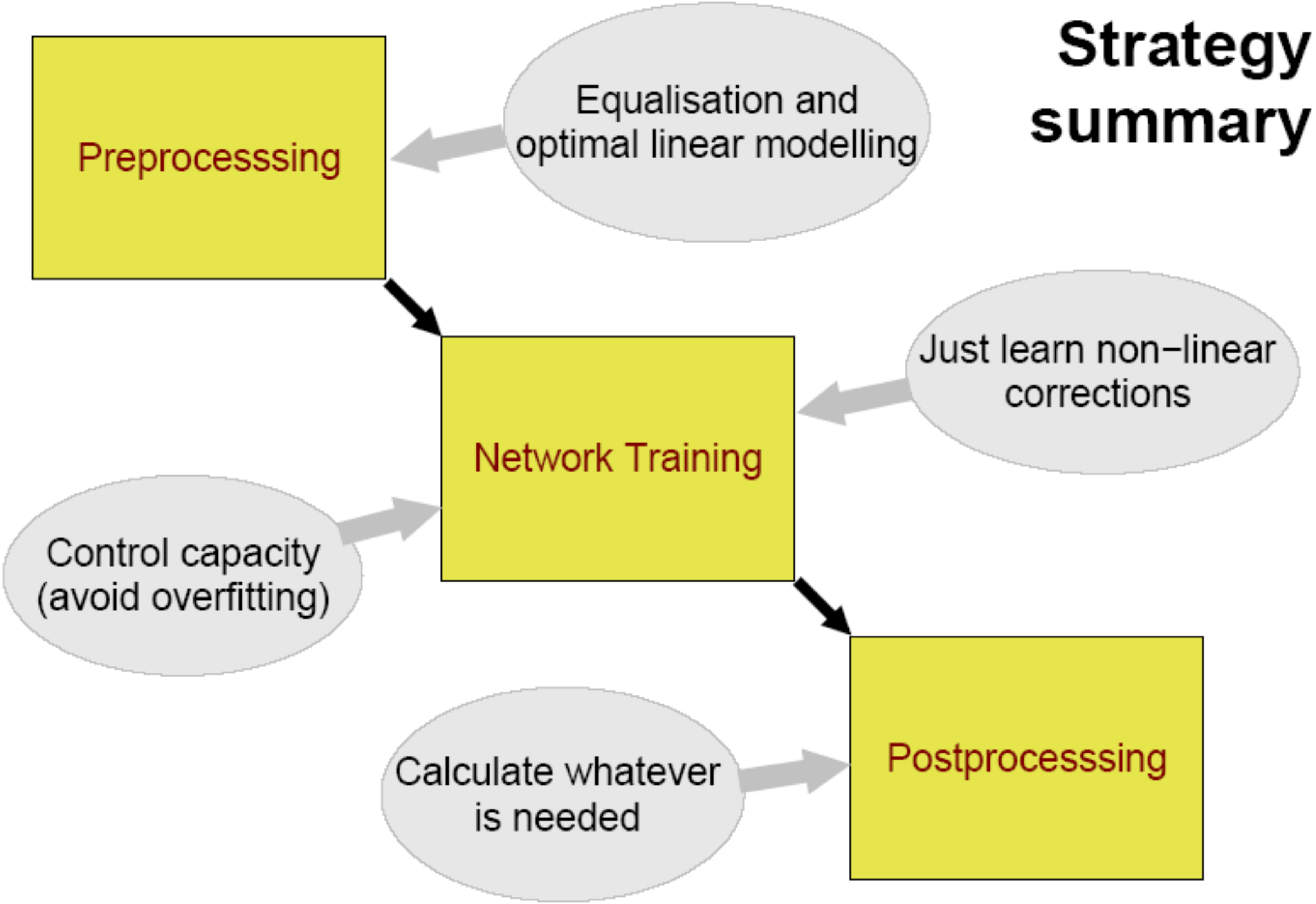
> tells you if there is nothing relevant to be learned

> delivers sensible prognoses already with small statistics

> has advanced boost and cross validation features

> is steadily further developed

Strategy summary



Preprocesssing

Equalisation and optimal linear modelling

Network Training

Just learn non−linear corrections

Control capacity (avoid overfitting)

Postprocesssing

Calculate whatever is needed

Bayesian Regularisation

Use Bayesian arguments to regularise network learning:

Likelihood

Prior

$$P(theory|data) = \frac{P(data|theory)P(theory)}{P(data)}$$

Posterior

Evidence

Learn only statistically relevant information, suppress statistical noise

Conditional probability densities in particle physics

What is the probability density
of the true B momentum
in this semileptonic B candidate event
taken with the CDF II detector

t

with these n tracks with those momenta and
rapidities in the hemisphere,
which are forming this secondary vertex
with this decay length and probability, this
invariant mass and transverse momentum,
this lepton information, this missing
transverse momentum, this difference in Phi
and Theta between momentum sum and
vertex topology, etc pp

$\vec{x}$

$f(t \mid \vec{x})$

NeuroBayes solution ansatz

Discretize f(t) into N intervals of same area by equalisation (nonlinear transformation t –> s)

↓

Train a neural network with N output nodes to the N binary decisions: The true t is larger than / lower than threshold i

↓

Fit smooth function (cubic spline) through N net outputs: = cumulated conditional probability in transformed variable s

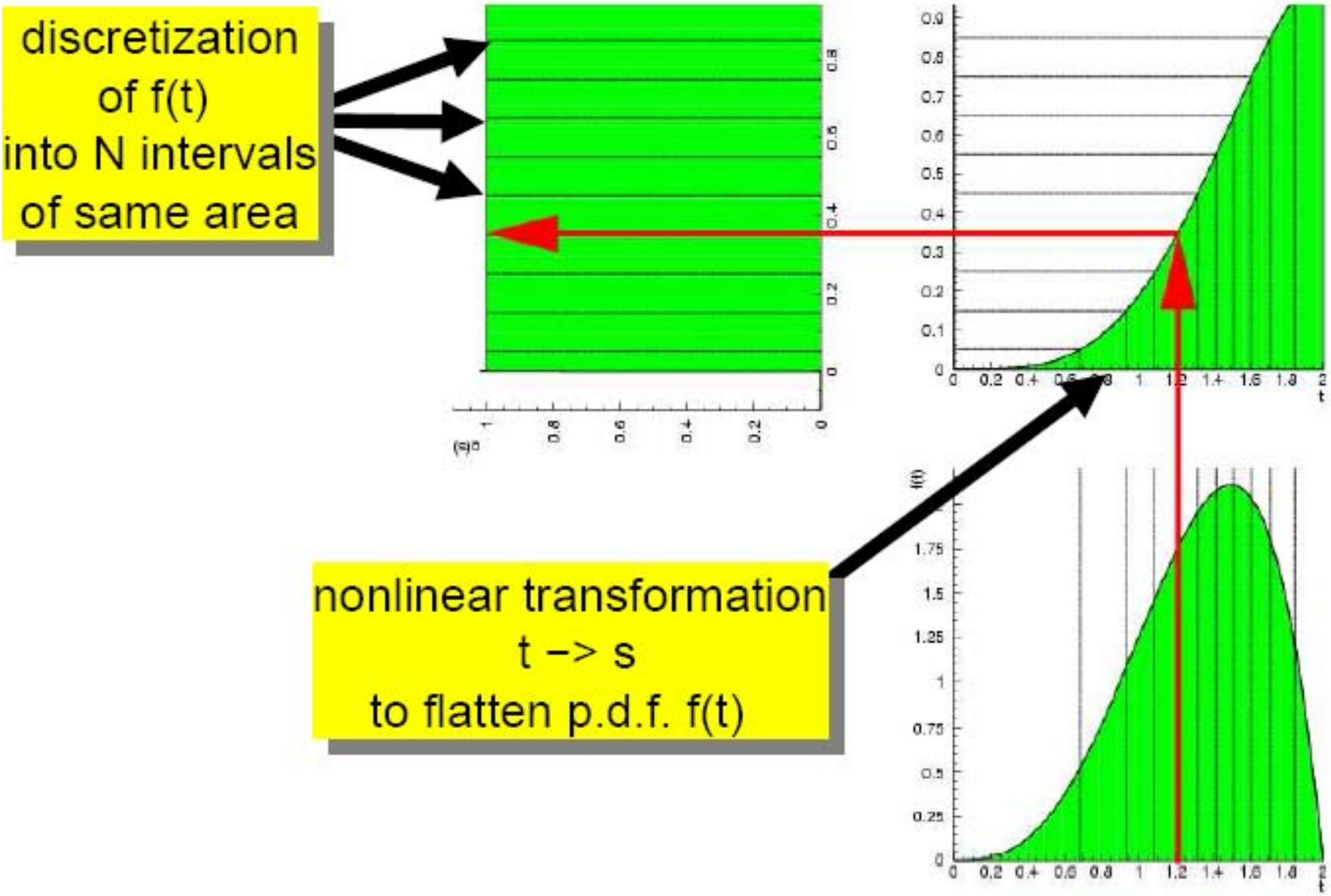
↓

Analytic differentiation returns probability density function in transformed variable s

↓

Back transformation to variable t returns f(t|x)

Equalisation and discretisation



discretization
of f(t)
into N intervals
of same area

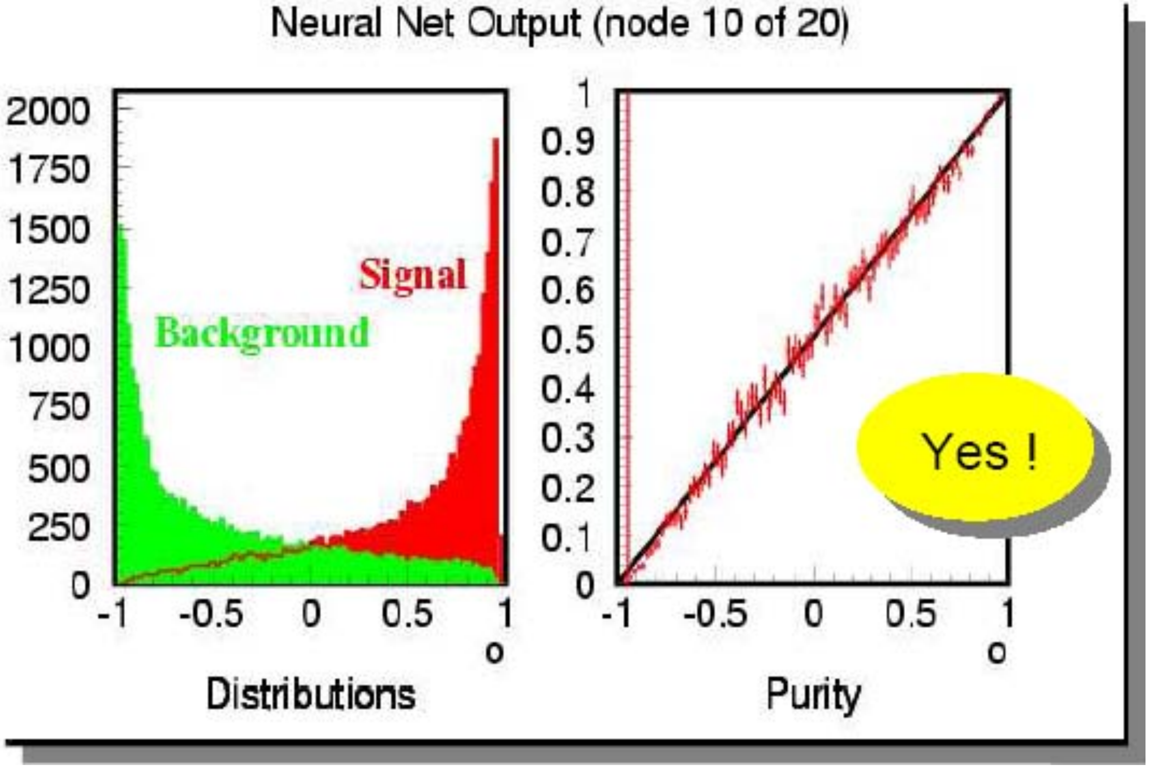
nonlinear transformation
t –> s
to flatten p.d.f. f(t)

Theoretical basis

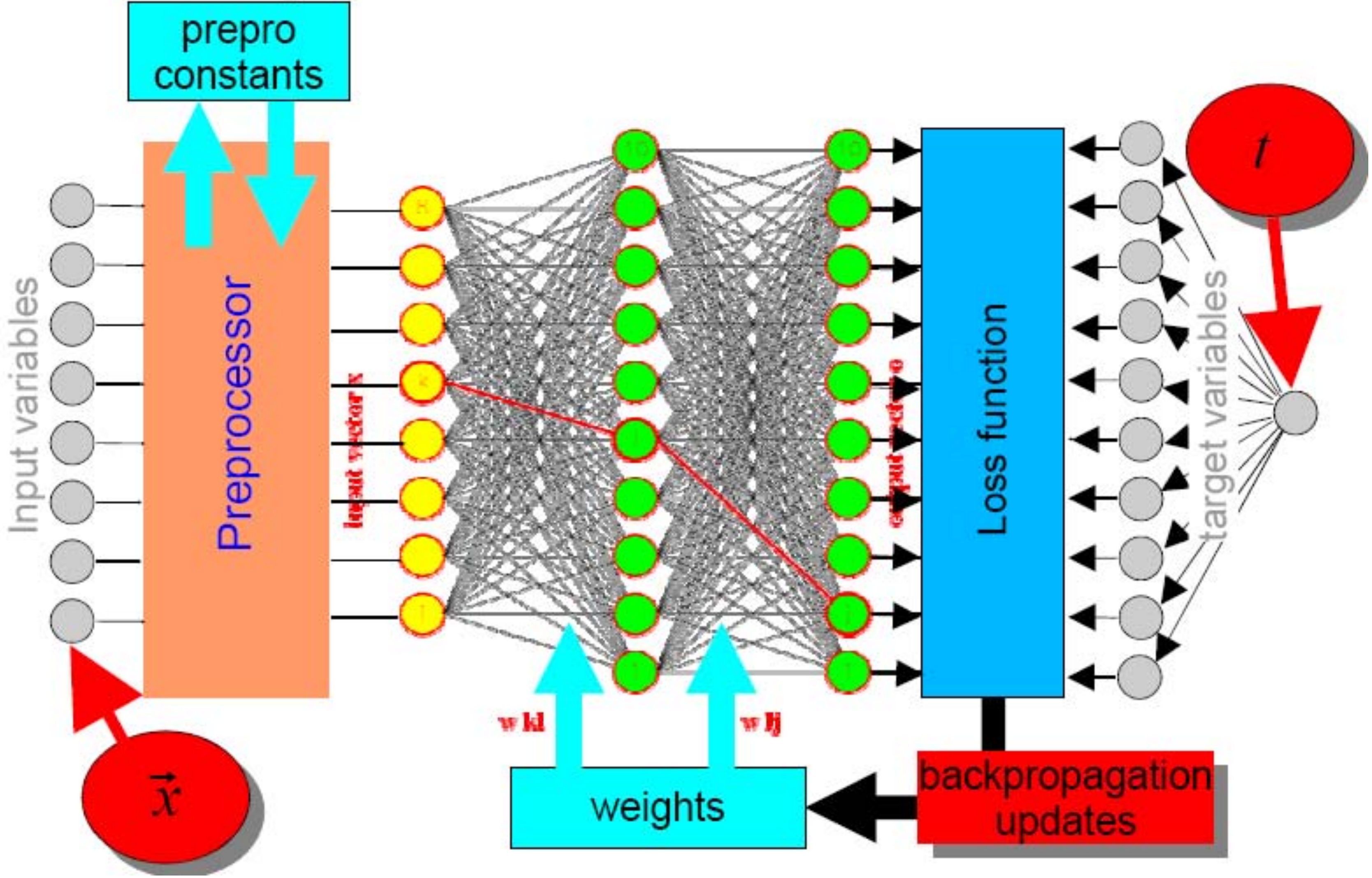
(Under some contollable conditions...)

neural network outputs can be interpreted as Bayesian a posteriori probability that the classification is correct

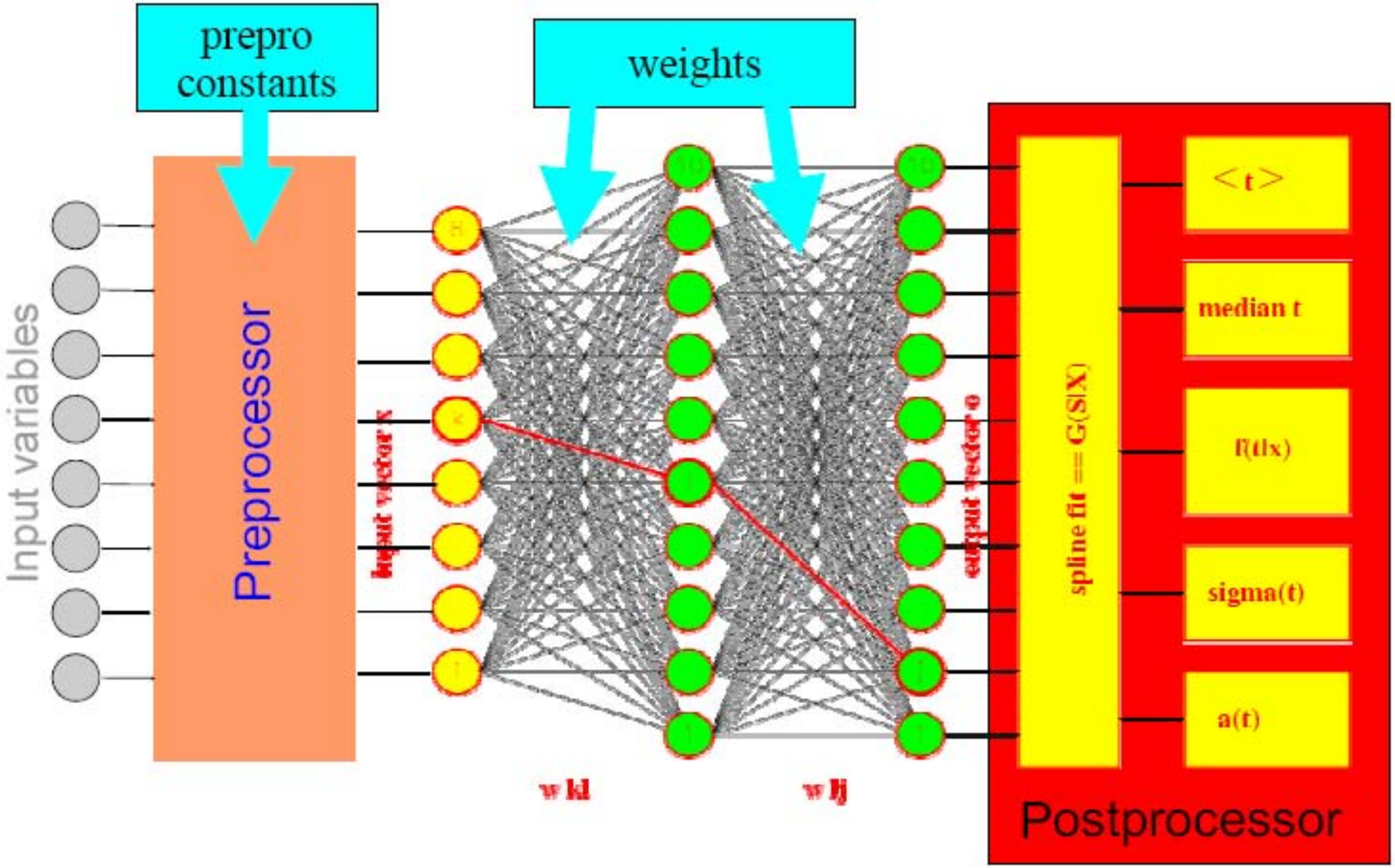
Purity of a given output is linear function of the output value



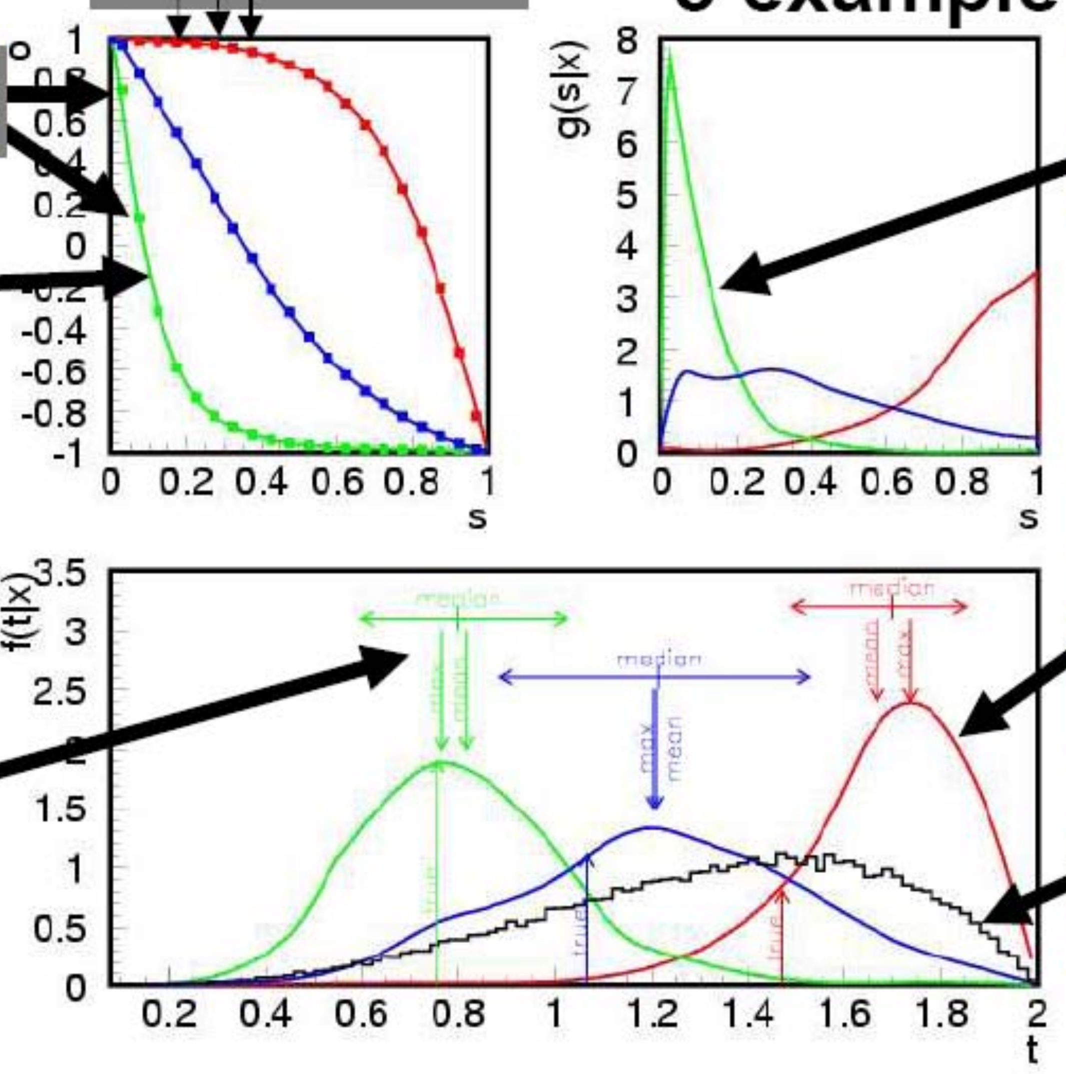
NeuroBayes Network architecture: Teacher



NeuroBayes network architecture: Expert



3 example events



20 net output nodes

20 net outputs

spline fit through outputs

analytic derivative of spline fit

back-transformed distributions $f(t|\vec{x})$

median, mean, max likelihood, error intervals

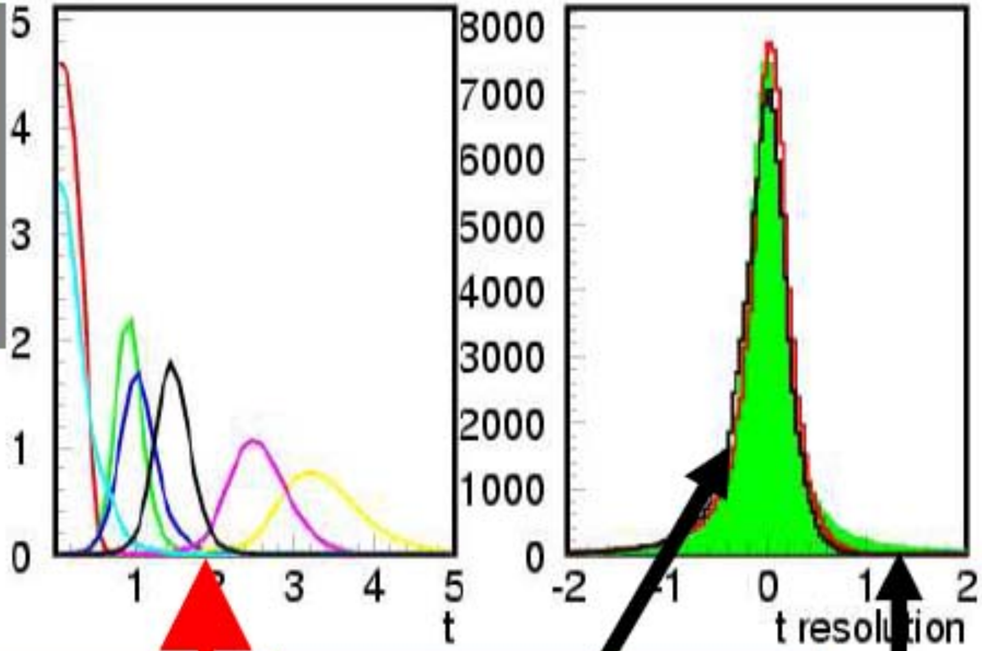
inclusive distribution

automatic error propagation

toy experiment :
measure (with errors)
- decay length d
- momentum p

and train for proper time t:

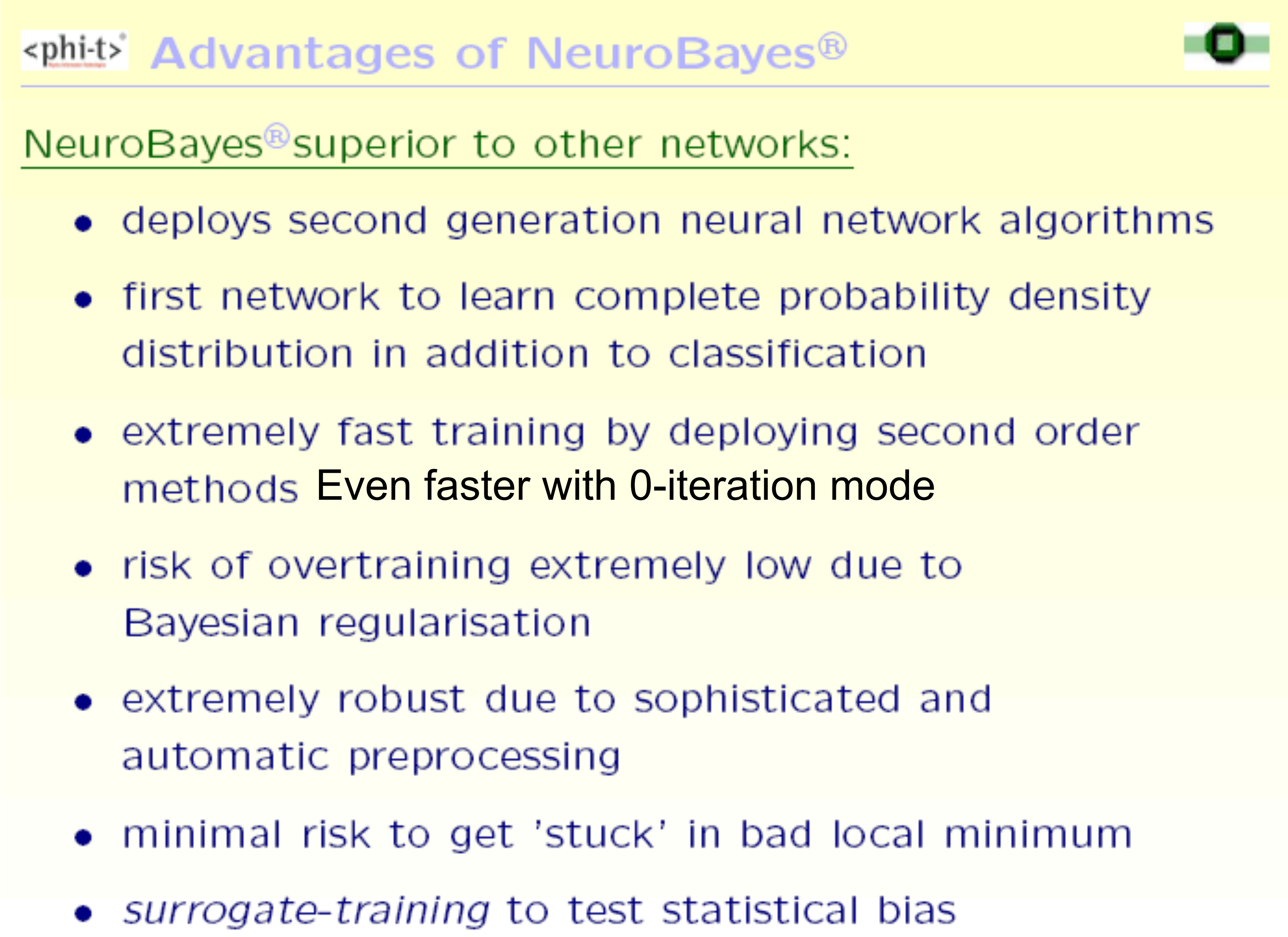
$$t = \frac{m}{c} \cdot \frac{d}{p}$$



Result :
networks learns automatically from data :
- that it should divide d by p
- how it should propagate errors
- true lifetimes are never negative (although both measured d and p can be)

Max likelihood estimate
median estimate

classical approach:
tail from negative
lifetimes



<phi-t>

Advantages of NeuroBayes®

NeuroBayes® superior to other networks:

- deploys second generation neural network algorithms
- first network to learn complete probability density distribution in addition to classification
- extremely fast training by deploying second order methods Even faster with 0-iteration mode
- risk of overtraining extremely low due to Bayesian regularisation
- extremely robust due to sophisticated and automatic preprocessing
- minimal risk to get 'stuck' in bad local minimum
- *surrogate-training* to test statistical bias

Preprocessing I

Why preprocess input variables?
Shouldn't the network learn it all?

Yes, but ...

Optimisation in many dimensions difficult

Example (2D): Deepest valley in Swiss Alps

Isn't the next valley deeper ?

→ difficult to find out once you are down there.

Now try to find minimum in $\mathcal{O}(1000)$ dimensions . . .

Preprocessing: "Guide" network to best minimum

<phi-t> Preprocessing II

Global preprocessing:

- normalisation and decorrelation
  → new covariance matrix is unit matrix
- rotate such that first variable contains all linear information about mean, second about width, etc.
- automatically recognise binary and discrete variables
- direct connection between input and output layer
  → network learns deviations from best linear estimate (for shape reconstruction)

Preprocessing of input variables I

Non−linear mapping to flat distribution between −1 and 1 (by ranking)

↓

Non−linear mapping to standard Gaussian

↓

De−correlation by diagonalisation of covariance matrix

↓

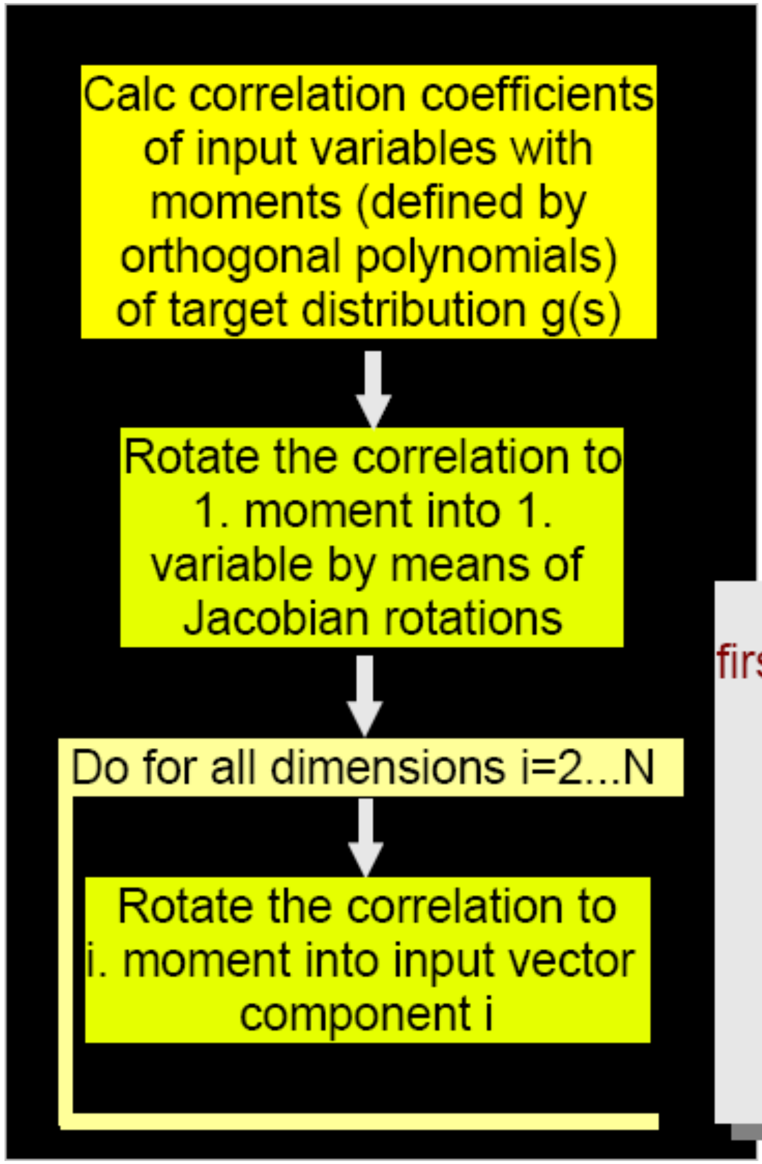
Normalisation by dividing by sqrt of eigenvalues of covariance matrix

At this stage all input variables are
- independent,
- centered at 0
- σ=1
- Gaussian (if no discrete input variables)

New covariance matrix is unit matrix

completely automatic
absolutely robust

Preprocessing of input variables II



Calc correlation coefficients of input variables with moments (defined by orthogonal polynomials) of target distribution g(s)

Rotate the correlation to 1. moment into 1. variable by means of Jacobian rotations

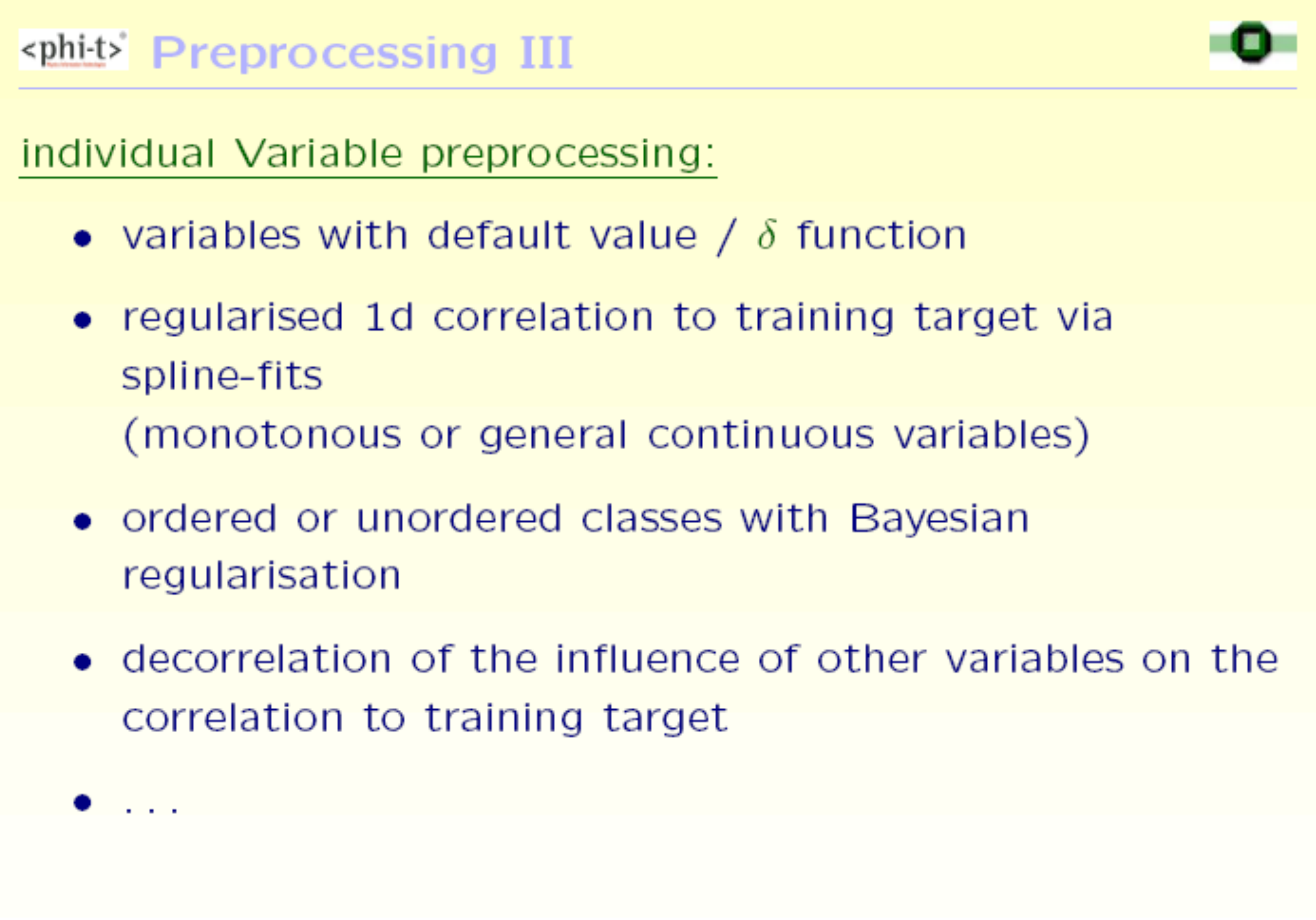
Do for all dimensions i=2...N

Rotate the correlation to i. moment into input vector component i

Degeneracy of new covariance matrix allows
arbitrary rotation of n−dimensional basis

Rotate such that
first variable contains all linear information on the mean value,
second all linear information on width,
i−th variable on i−th moment.

The larger i, the more statistical uncertainty:
high frequency oscillations in solution
(ill−posed problem)



Preprocessing III

individual Variable preprocessing:

- variables with default value / δ function
- regularised 1d correlation to training target via spline-fits
  (monotonous or general continuous variables)
- ordered or unordered classes with Bayesian regularisation
- decorrelation of the influence of other variables on the correlation to training target
- ...

NeuroBayes® Teacher output ($e^{\pm}$ ID)

analyse correlations:

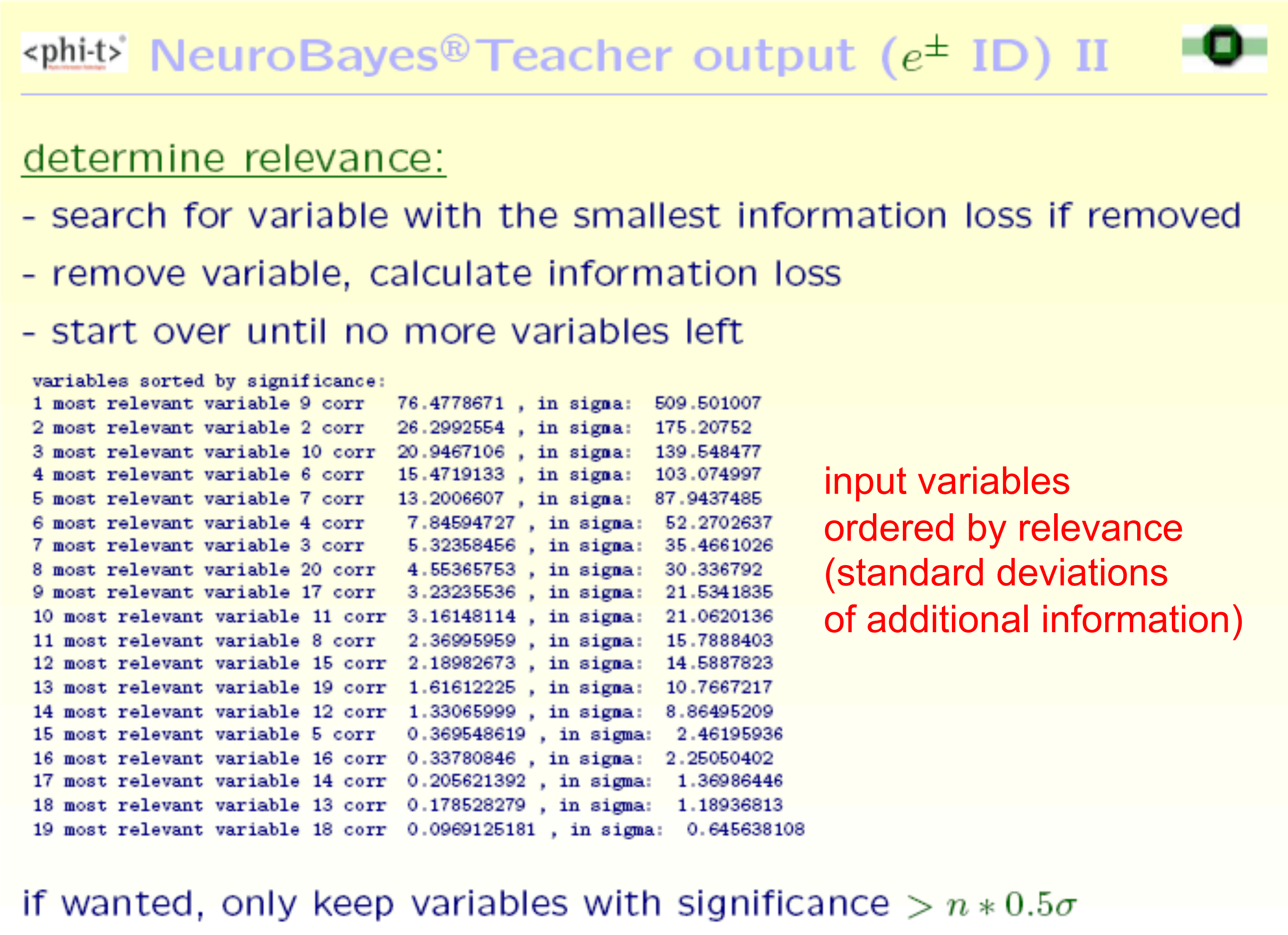
covariance matrix: $V_{ij} = \frac{1}{N}\sum_{events}(x_i - < x_i >) * (x_j - < x_j >)$

correlation matrix: $\rho_{ij} = \frac{V_{ij}}{\sigma_i \sigma_j}$

after preproc: $< x_i >= 1$ and $\sigma_i = 1$

training target

```
COVARIANCE MATRIX (IN PERCENT) (truncated at variable 15)
  0    1.0    2.0    3.0    4.0    5.0    6.0    7.0    8.0    9.0   10.0   11.0   12.0   13.0   14.0   15.0
  0    0.7    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0    1.0
  1  100.0   66.0   48.6   31.1   39.3   51.1   77.3   57.0   76.5   61.2   48.7    6.1    8.9    5.5    9.2
  2   66.0  100.0   41.5   25.7   45.8   37.6   64.0   45.4   58.4   44.2   48.9    6.5    8.9    5.4    8.7
  3   48.6   41.5  100.0   33.5   23.5   27.7   41.8   33.0   48.3   31.6   34.1    7.3    9.4    4.9    9.7
  4   31.1   25.7   33.5  100.0   30.7   17.3   26.4   14.7   24.1   13.4   50.5   17.9   19.7    8.5   20.5
  5   39.3   45.8   23.5   30.7  100.0   23.6   38.6   25.2   37.5   22.3   66.1    8.7    9.9   13.3   14.8
  6   51.1   37.6   27.7   17.3   23.6  100.0   40.8   43.2   41.5   29.2   27.7   11.5   10.8    8.1   12.0
  7   77.3   64.0   41.8   26.4   38.6   40.8  100.0   57.6   82.9   51.6   47.2    6.8    9.5    7.4   10.3
  8   57.0   45.4   33.0   14.7   25.2   43.2   57.6  100.0   55.4   47.8   28.1    7.4    9.4    7.9   11.7
  9   76.5   58.4   48.3   24.1   37.5   41.5   82.9   55.4  100.0   50.8   44.4    8.7   11.0    5.1   10.1
 10   61.2   44.2   31.6   13.4   22.3   29.2   51.6   47.8   50.8  100.0   27.5   -1.2    2.6    1.7    2.7
 11   48.7   48.9   34.1   50.5   66.1   27.7   47.2   28.1   44.4   27.5  100.0   11.2   12.8   10.8   16.4
 12    6.1    6.5    7.3   17.9    8.7   11.5    6.8    7.4    8.7   -1.2   11.2  100.0   71.2    4.6   45.6
 13    8.9    8.9    9.4   19.7    9.9   10.8    9.5    9.4   11.0    2.6   12.8   71.2  100.0    5.4   63.1
 14    5.5    5.4    4.9    8.5   13.3    8.1    7.4    7.9    5.1    1.7   10.8    4.6    5.4  100.0   72.6
 15    9.2    8.7    9.7   20.5   14.8   12.0   10.3   11.7   10.1    2.7   16.4   45.6   63.1   72.6  100.0
 16    1.7    1.8    1.7    1.0    3.2    2.5    2.3    6.2    2.7   -0.3    1.1    1.0    1.2    0.9    1.2
 17    7.7    4.0    3.0    2.1    2.4    1.0    5.7    1.2    4.9    3.5    2.9    0.4    0.5    0.3    0.5
 18    2.2    1.8    1.5   -1.0    0.1    0.9    4.3   10.8    3.4   -0.9    0.2   -1.1   -0.8   -1.1   -1.2
 19    4.9    2.6    2.0    0.8    1.3    0.5    3.7    1.5    3.6    2.3    1.7    0.2    0.4    0.1    0.2
 20   15.5    9.5    6.1    1.2    6.4    6.4   13.4   10.6   11.4    8.9    5.9    1.2    1.9    2.4    2.3
```



<phi-t> NeuroBayes® Teacher output ($e^{\pm}$ ID) II

determine relevance:

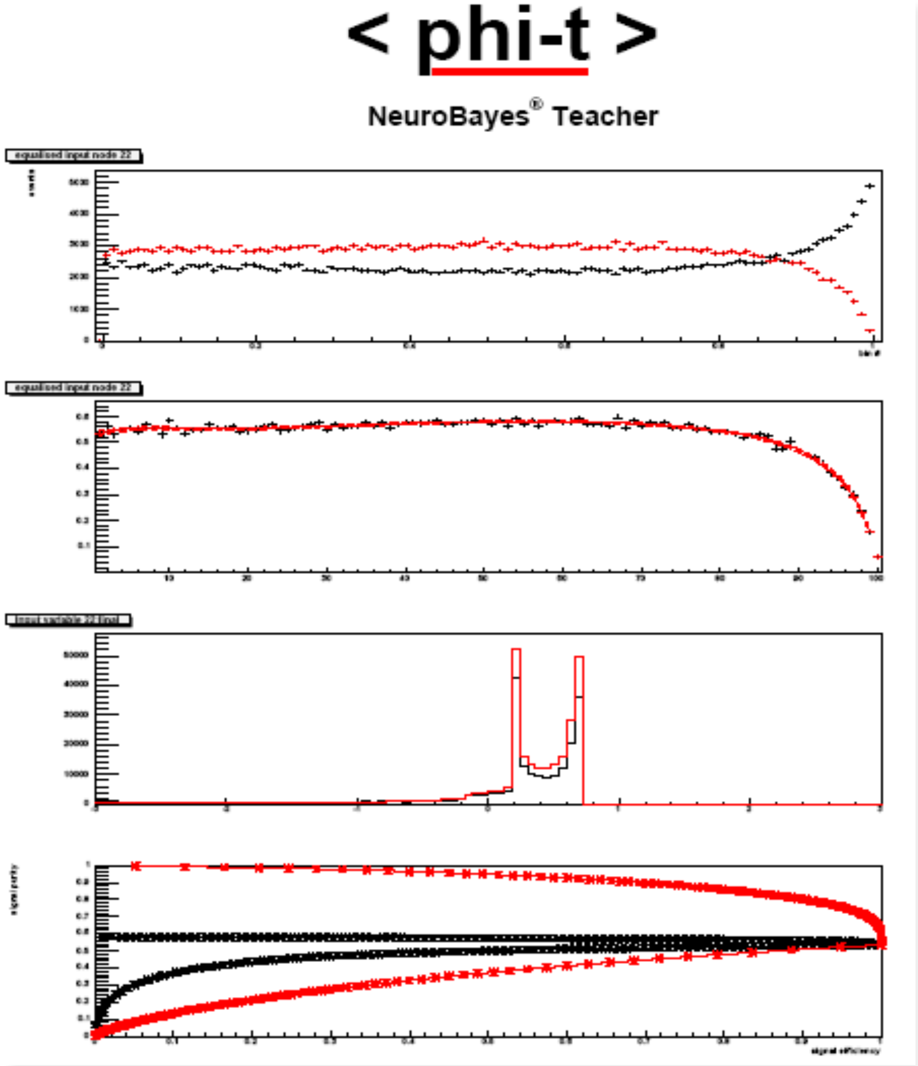
- search for variable with the smallest information loss if removed
- remove variable, calculate information loss
- start over until no more variables left

```
variables sorted by significance:
1 most relevant variable 9 corr   76.4778671 , in sigma:  509.501007
2 most relevant variable 2 corr   26.2992554 , in sigma:  175.20752
3 most relevant variable 10 corr  20.9467106 , in sigma:  139.548477
4 most relevant variable 6 corr   15.4719133 , in sigma:  103.074997
5 most relevant variable 7 corr   13.2006607 , in sigma:  87.9437485
6 most relevant variable 4 corr    7.84594727 , in sigma:  52.2702637
7 most relevant variable 3 corr    5.32358456 , in sigma:  35.4661026
8 most relevant variable 20 corr   4.55365753 , in sigma:  30.336792
9 most relevant variable 17 corr   3.23235536 , in sigma:  21.5341835
10 most relevant variable 11 corr  3.16148114 , in sigma:  21.0620136
11 most relevant variable 8 corr   2.36995959 , in sigma:  15.7888403
12 most relevant variable 15 corr  2.18982673 , in sigma:  14.5887823
13 most relevant variable 19 corr  1.61612225 , in sigma:  10.7667217
14 most relevant variable 12 corr  1.33065999 , in sigma:  8.86495209
15 most relevant variable 5 corr   0.369548619 , in sigma:  2.46195936
16 most relevant variable 16 corr  0.33780846 , in sigma:  2.25050402
17 most relevant variable 14 corr  0.205621392 , in sigma:  1.36986446
18 most relevant variable 13 corr  0.178528279 , in sigma:  1.18936813
19 most relevant variable 18 corr  0.0969125181 , in sigma:  0.645638108
```

input variables
ordered by relevance
(standard deviations
of additional information)

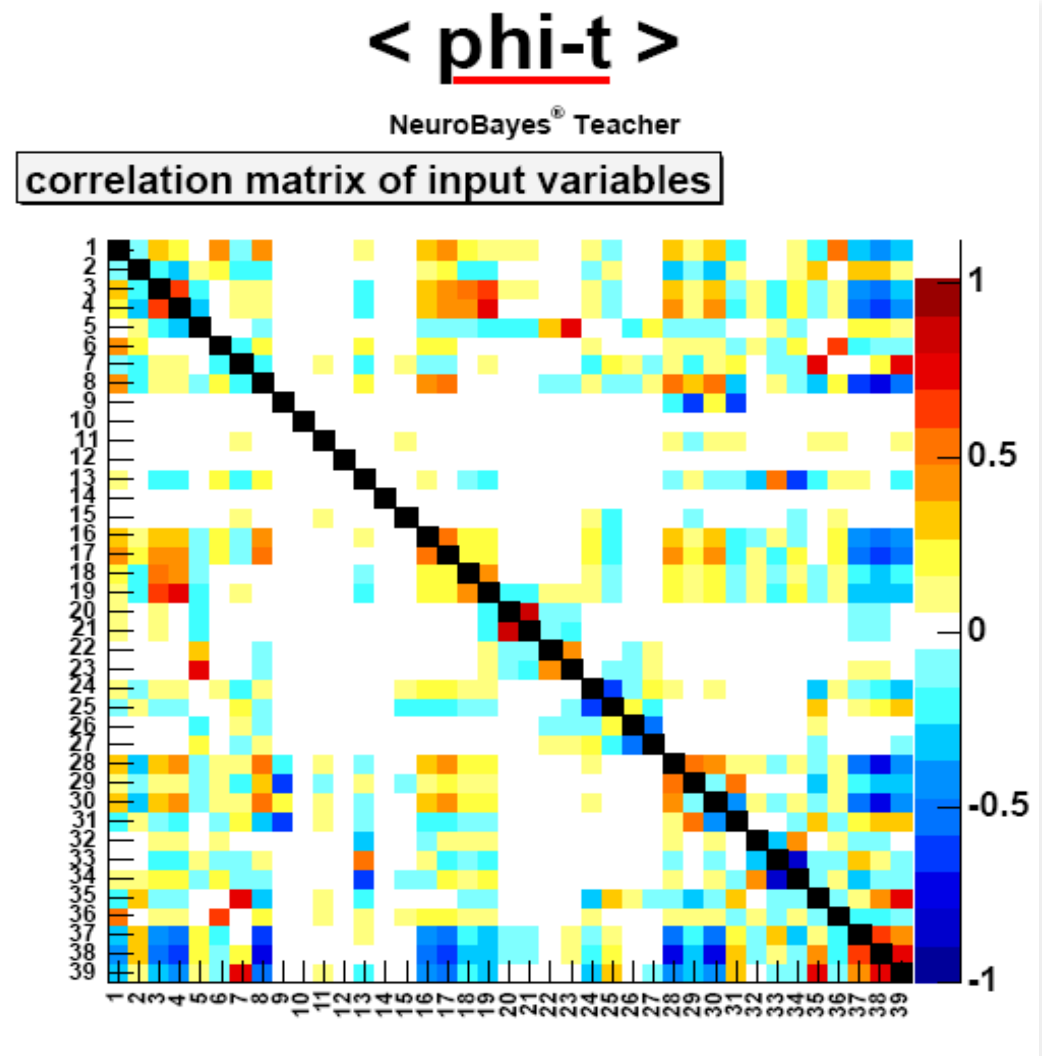
if wanted, only keep variables with significance $> n * 0.5\sigma$

Visualisation of single input-variables

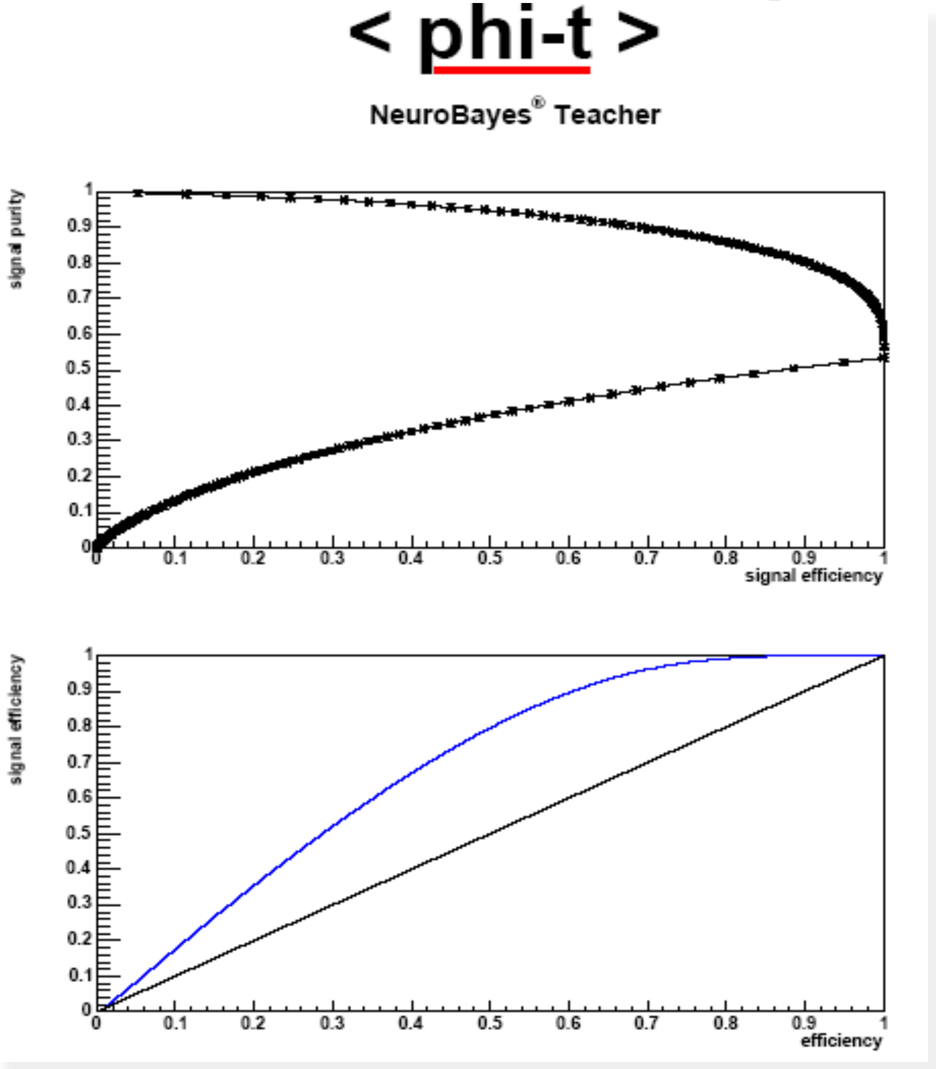


Visualisation of correlation matrix

Variable 1: Training target



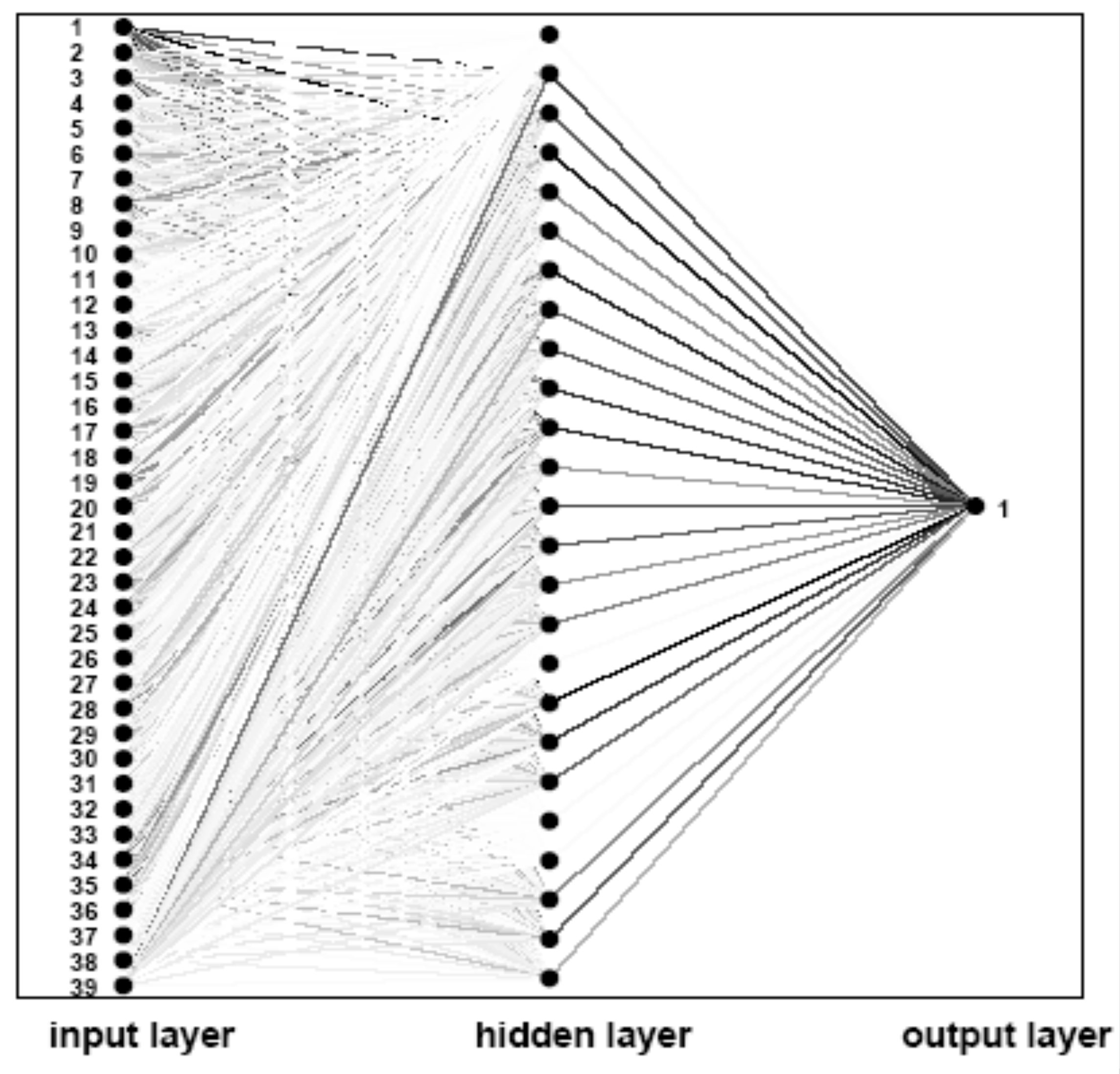
Visualisation of network performance

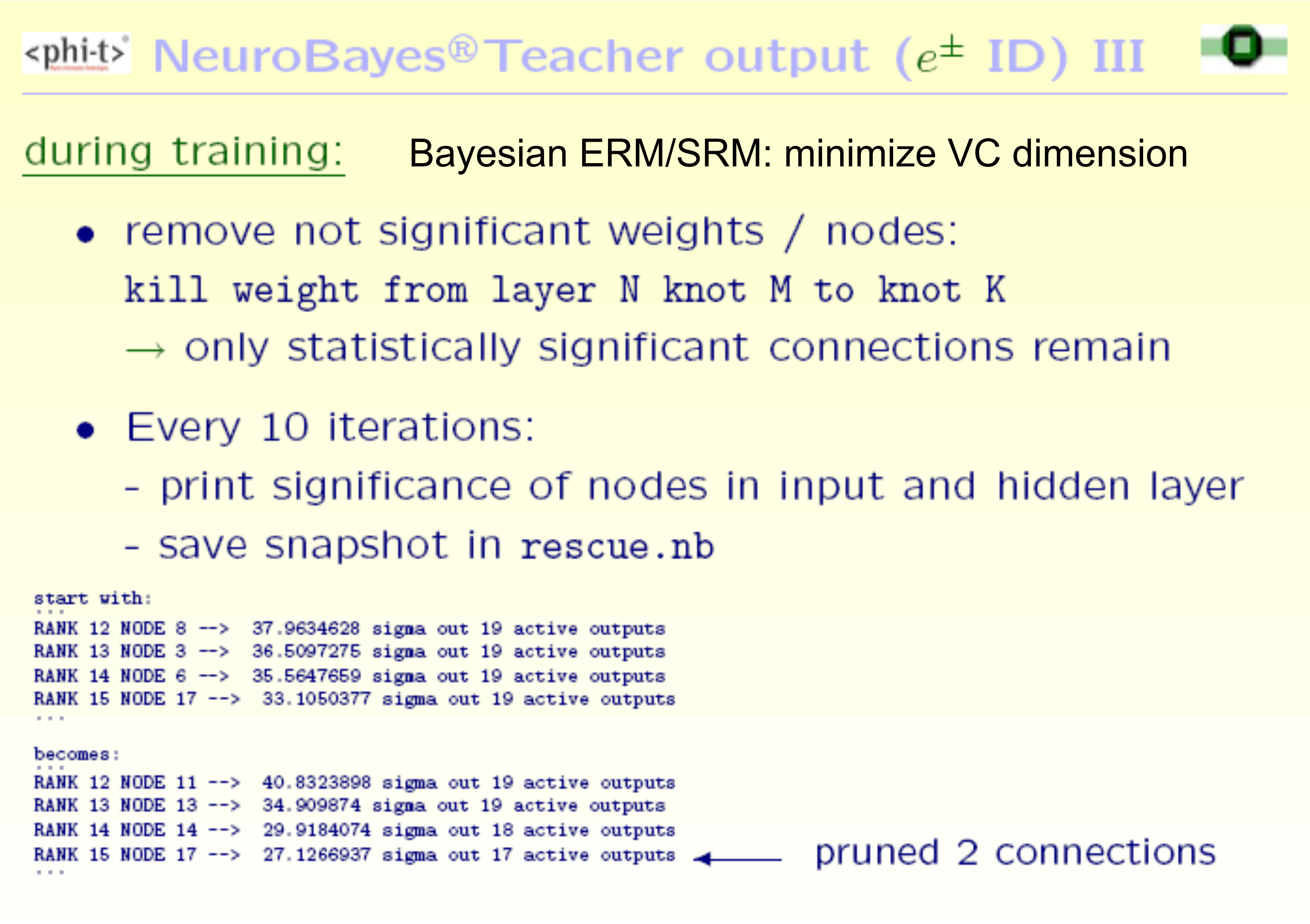


Purity vs. efficiency

Signal-effiziency vs.
total efficiency
(Lift chart)

Visualisation of NeuroBayes network topology





<phi-t>

NeuroBayes® Teacher output ($e^{\pm}$ ID) III

during training: Bayesian ERM/SRM: minimize VC dimension

- remove not significant weights / nodes:

  `kill weight from layer N knot M to knot K`

  → only statistically significant connections remain

- Every 10 iterations:
  - print significance of nodes in input and hidden layer
  - save snapshot in `rescue.nb`

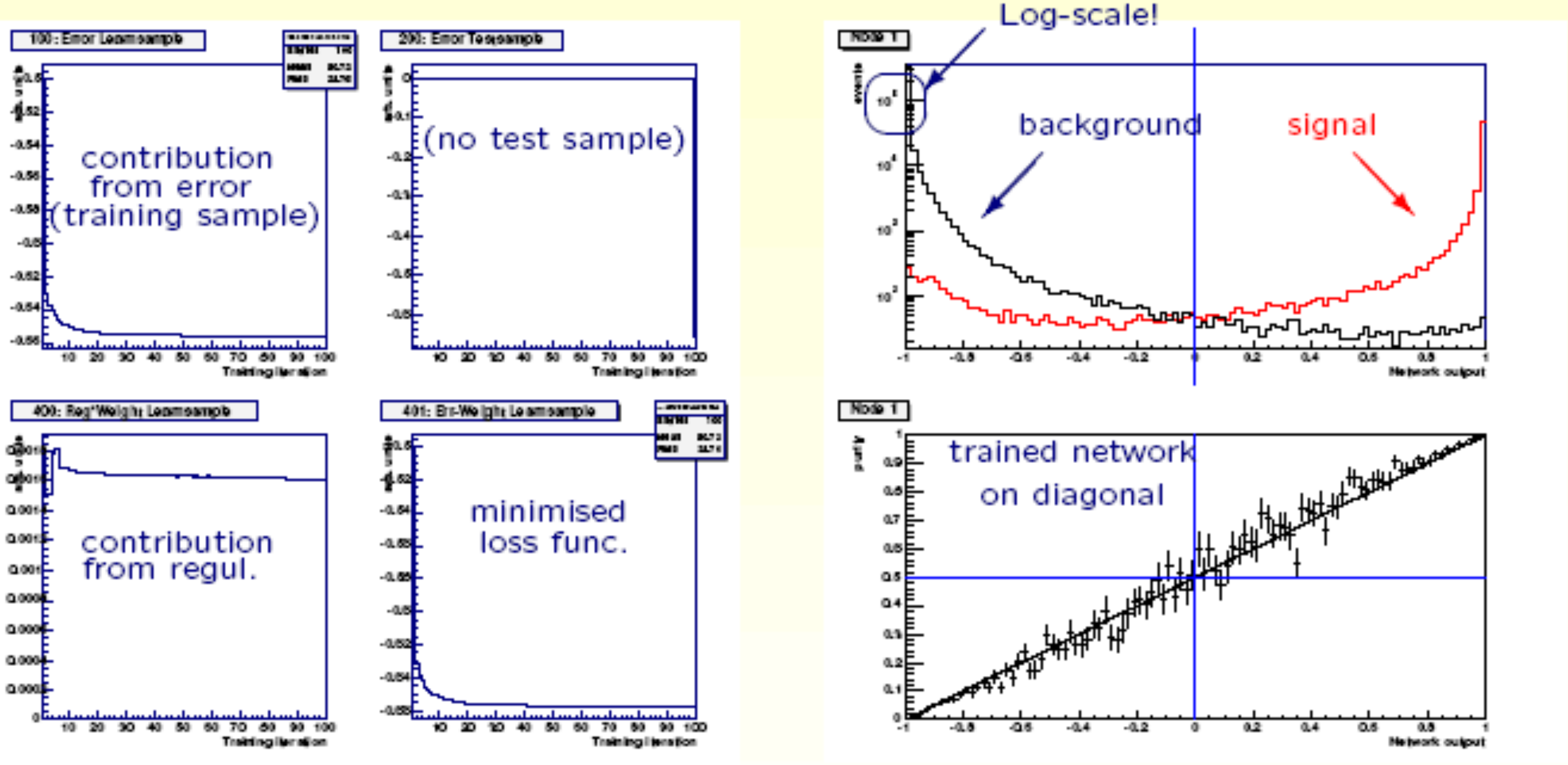
```
start with:
...
RANK 12 NODE 8 -->  37.9634628 sigma out 19 active outputs
RANK 13 NODE 3 -->  36.5097275 sigma out 19 active outputs
RANK 14 NODE 6 -->  35.5647659 sigma out 19 active outputs
RANK 15 NODE 17 -->  33.1050377 sigma out 19 active outputs
...

becomes:
...
RANK 12 NODE 11 -->  40.8323898 sigma out 19 active outputs
RANK 13 NODE 13 -->  34.909874 sigma out 19 active outputs
RANK 14 NODE 14 -->  29.9184074 sigma out 18 active outputs
RANK 15 NODE 17 -->  27.1266937 sigma out 17 active outputs
...
```

← pruned 2 connections

NeuroBayes® Teacher output ($e^{\pm}$ ID) IV

after training: Analysis of control plots



contribution from error (training sample)

(no test sample)

contribution from regul.

minimised loss func.

Log-scale!

background

signal

trained network on diagonal

More than 50 diploma and Ph.D. theses…

from experiments DELPHI, CDF II, AMS, CMS and Belle used NeuroBayes® or predecessors very successfully.

Many of these can be found at
www.neurobayes.de

Talks about NeuroBayes® and applications:
www-ekp.physik.uni-karlsruhe.de/~feindt → Forschung

Recent highlights using NeuroBayes (all CDF II):

Discovery of orbitally excited B+ und B_s**- mesons**

First observation of particle antiparticle oscillations of B_s- mesons

Measurement of lifetime difference of short and long lived B_s mesons and limits on CP-violating parameters

Spin-parity determination and most precise mass determination of X(3872) (exotic, not a normal meson)

Discovery of single top quark production mechanism

First exclusion of a 160-170 GeV standard model Higgs boson

Press (Die Welt, April 21, 2006)

DIE WELT Seite 31

Ur-Schlange hatte Hüfte und Beine

WISSENSCHAFT KOMPAKT

RAUMFAHRT

Start von „Cloud-Sat"

Antimaterie wandeln

ch – Software aus Deutschland ermöglichte die Messungen

Sekunde 2,8 Bil-
-B-Mesonen um
ede Sekunde et-

Tempo dieses Tanzes gemessen", sagt Jacobo Konigsberg, Sprecher der CDF-Kollaboration.

kenntnisse über die Eigenschaften der Elementarteilchen, sondern auch über die Entwicklung des

Materie kann sich in Antimaterie wandeln

Amerikanische Elementarteilchenphysiker melden Durchbruch – Software aus Deutschland ermöglichte die Messungen

„Wenn man sich Materie vorstellt, die mit Antimaterie tanzt, dann haben wir das unglaubliche Tempo dieses Tanzes gemessen"

Some applications in high energy physics

DELPHI: (mainly predecessors of NeuroBayes in BSAURUS)

Kaon, proton, electron id
Optimisation of resolutions inclusive B- E, ϕ, θ, Q-value
B^{**}, B_s^{**} enrichment
B fragmentation function
Limit on B_s-mixing
B^0-mixing
B- F/B-asymmetry
B-> wrong sign charm

Some applications in high energy physics

CDF II:

Electron ID, muon ID, kaon/proton ID
Optimisation of resonance reconstruction in many channels (X, Y, D, D_s , D_s**, B, B_s, B**,B_s**)
Spin parity analysis of X(3182)
Inclusion of NB output in likelihood fits
B-tagging for high pt physics (top, Higgs, etc.)
B-Flavour tagging for mixing analyses (new combined tagging)
B0, B_s-lifetime, $\Delta\Gamma$, mixing, CP violation
Discovery of single top quark production
Higgs search, first high energy Standard Model exclusion limits

Some applications in high energy physics

CMS:

B-tagging
single top physics
Higgs searches

Belle:

Continuum suppression
B full reconstruction
B flavour tagging
KEKB accelerator optimisation

H1:

Calorimeter response optimisation

LHCb, ATLAS

First studies

More than 50 diploma and Ph.D. theses…

from experiments DELPHI, CDF II, AMS, CMS and Belle used NeuroBayes® or predecessors very successfully.

Many of these can be found at

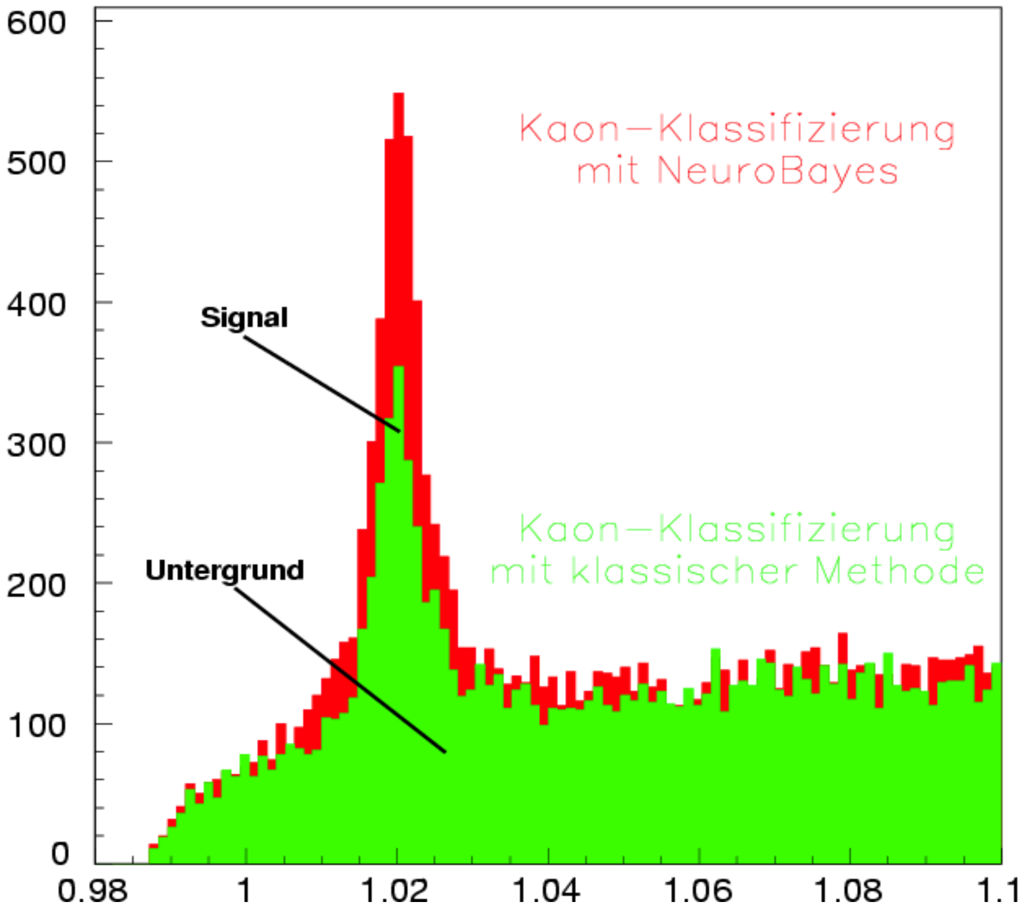
www.phi-t.de → Wissenschaft → NeuroBayes

Talks about NeuroBayes® and applications:

www-ekp.physik.uni-karlsruhe.de/~feindt → Forschung

Early examples (DELPHI)

$\Phi \rightarrow K^+K^-$



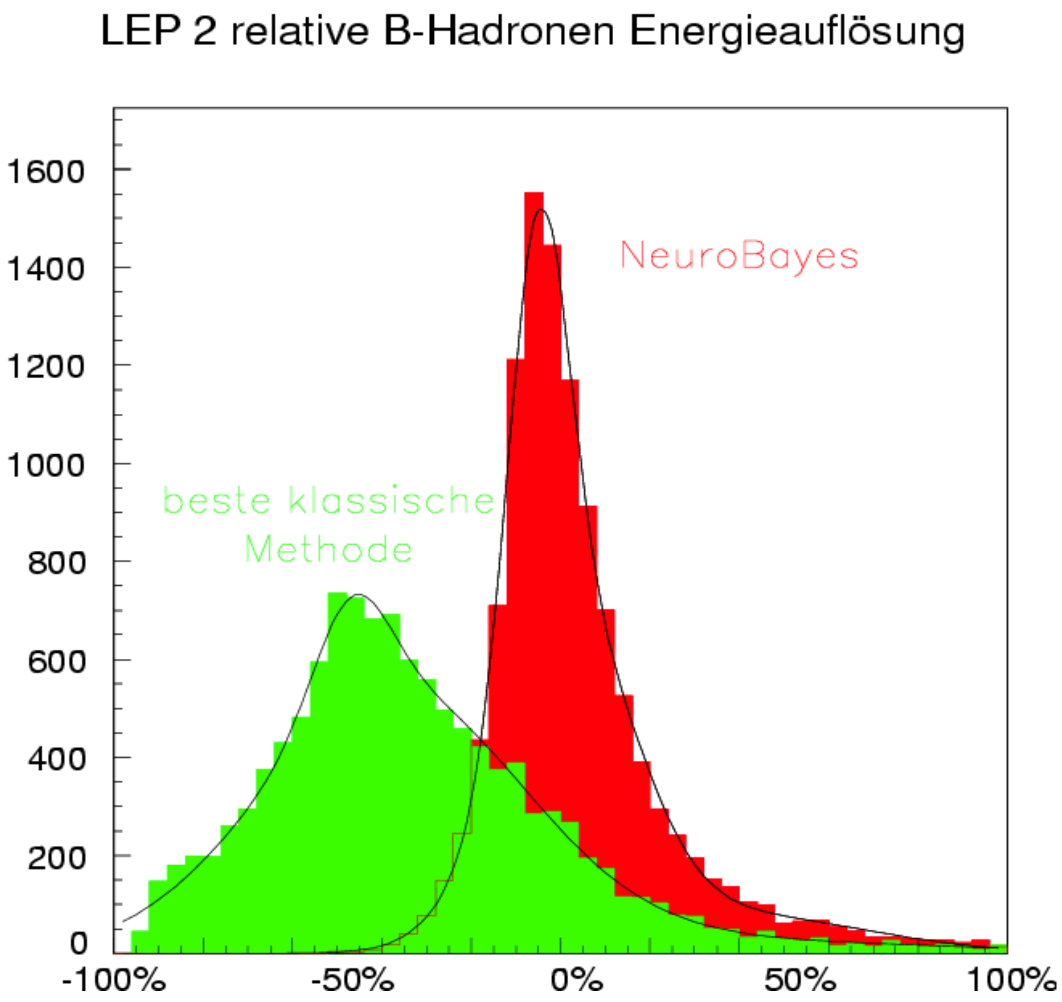
Classification:
Hadron Identification
(DELPHI at CERN):

Doubled signal strength at constant background level by neural network classification

original method : several 10 millions CHF cost

NeuroBayes predecessor: Additional factor of 2 with very limited additional effort

Density training, mean (DELPHI, CERN)



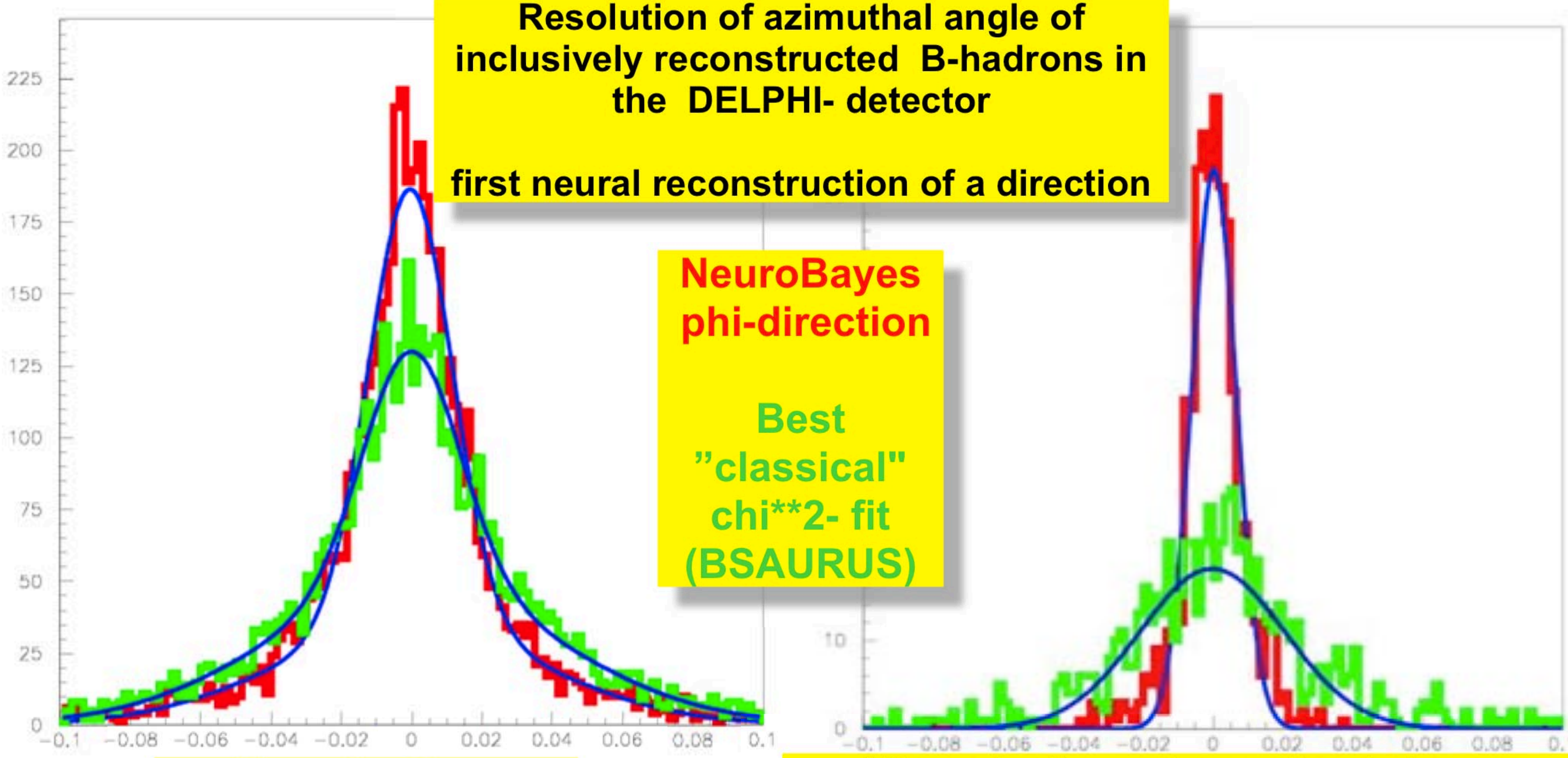
Optimised reconstruction of real valued quantities: extended regression

much improved resolution

(narrow peak around +-0)

by NeuroBayes-technology

Direction of B-mesons (DELPHI)



Resolution of azimuthal angle of inclusively reconstructed B-hadrons in the DELPHI- detector

first neural reconstruction of a direction

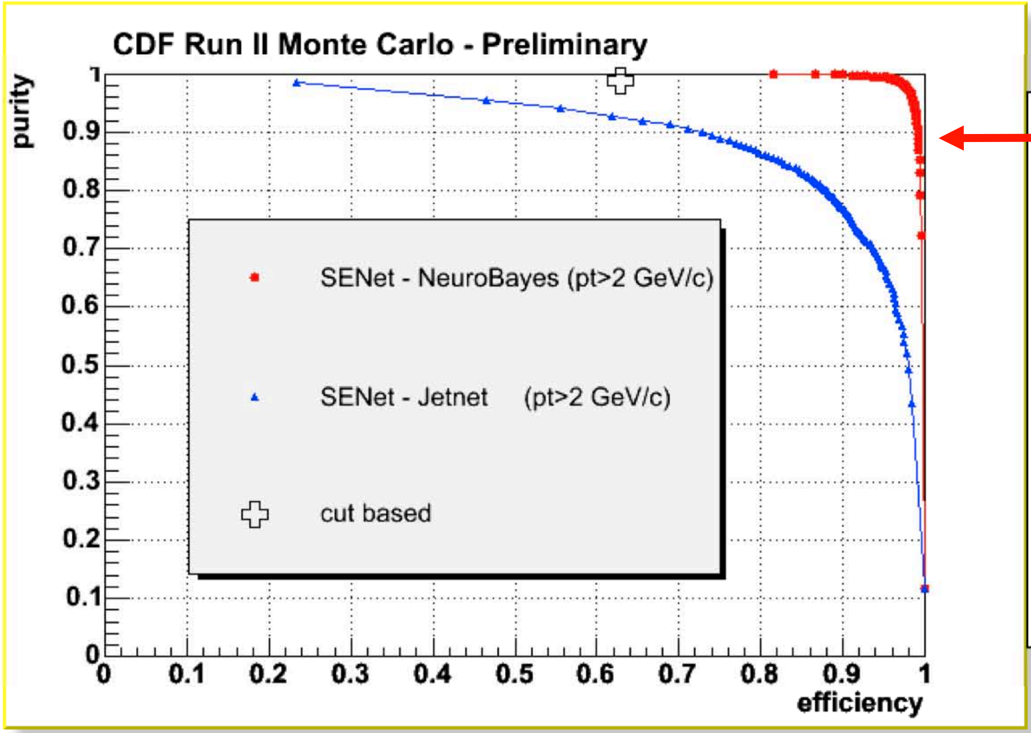
NeuroBayes phi-direction

Best ”classical" chi2- fit (BSAURUS)**

No selection:
Improved resolution

After selection cut on estimated error:
Resolution massively improved, no tails
==> allows reliable selection of good events

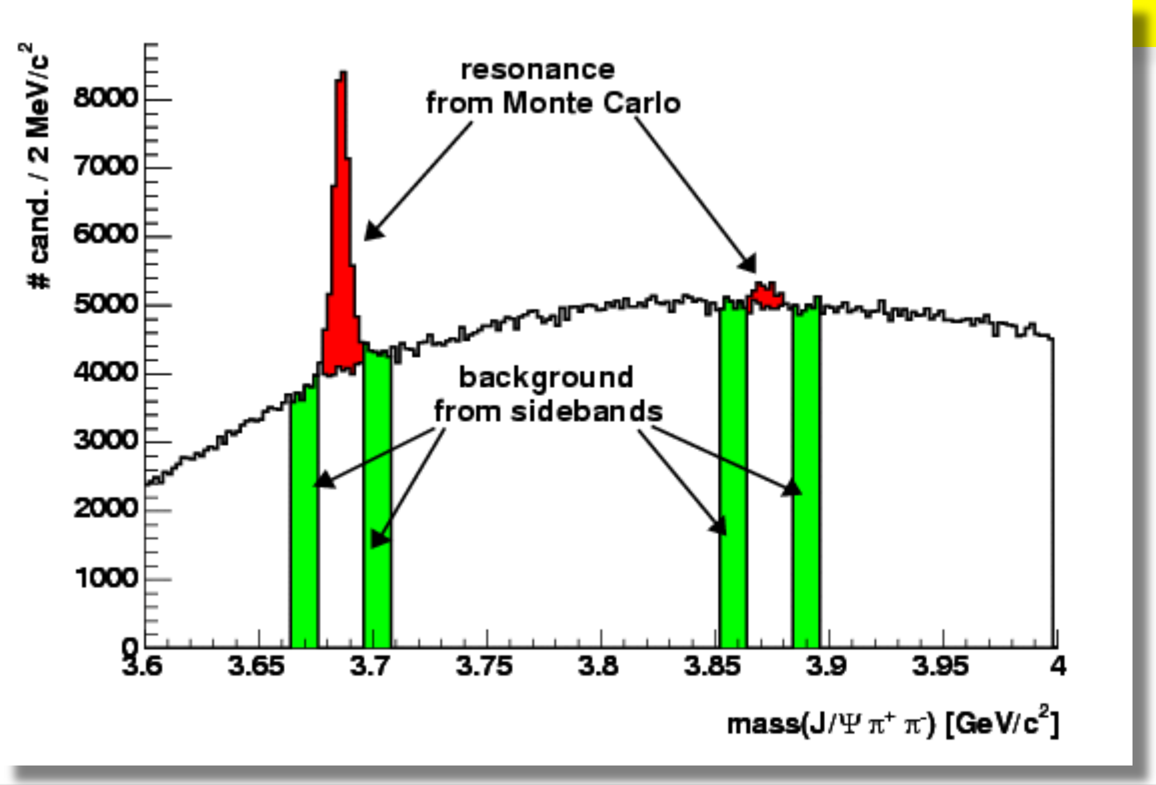
Particle identification (soft electrons in CDF II)



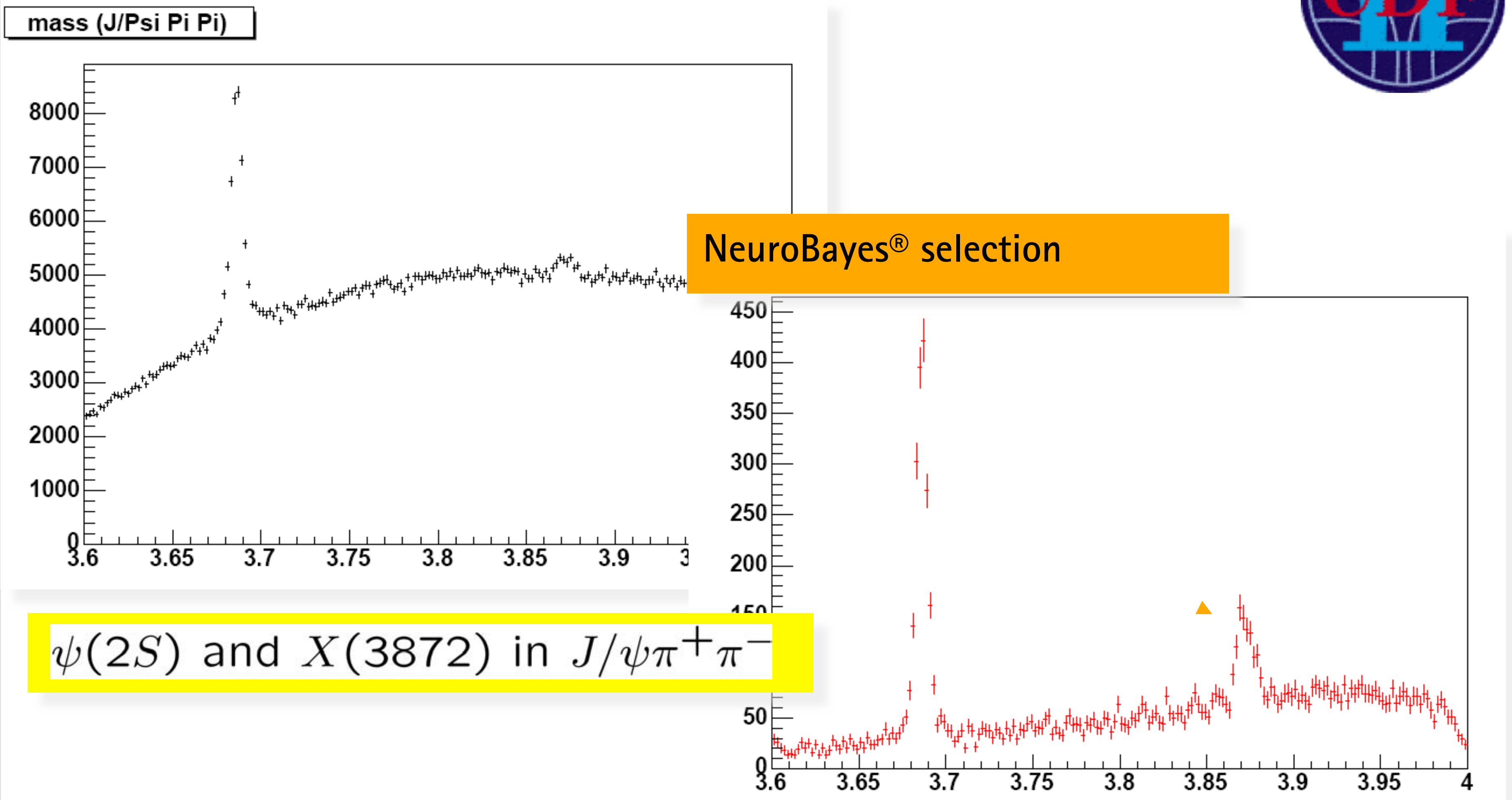
Thesis U. Kerzel:
on basis of Soft Electron Collection
(much more efficient than
cut selection
or JetNet with same inputs)
- after clever preprocessing by hand
  and careful learning parameter
  choice this could also be as good
  as NeuroBayes®

Hadron collider: No good MC for backgrounds available
MC for resonance production with different J^{PC} assumptions

Idea: take background from sidebands in data
check that network cannot learn mass

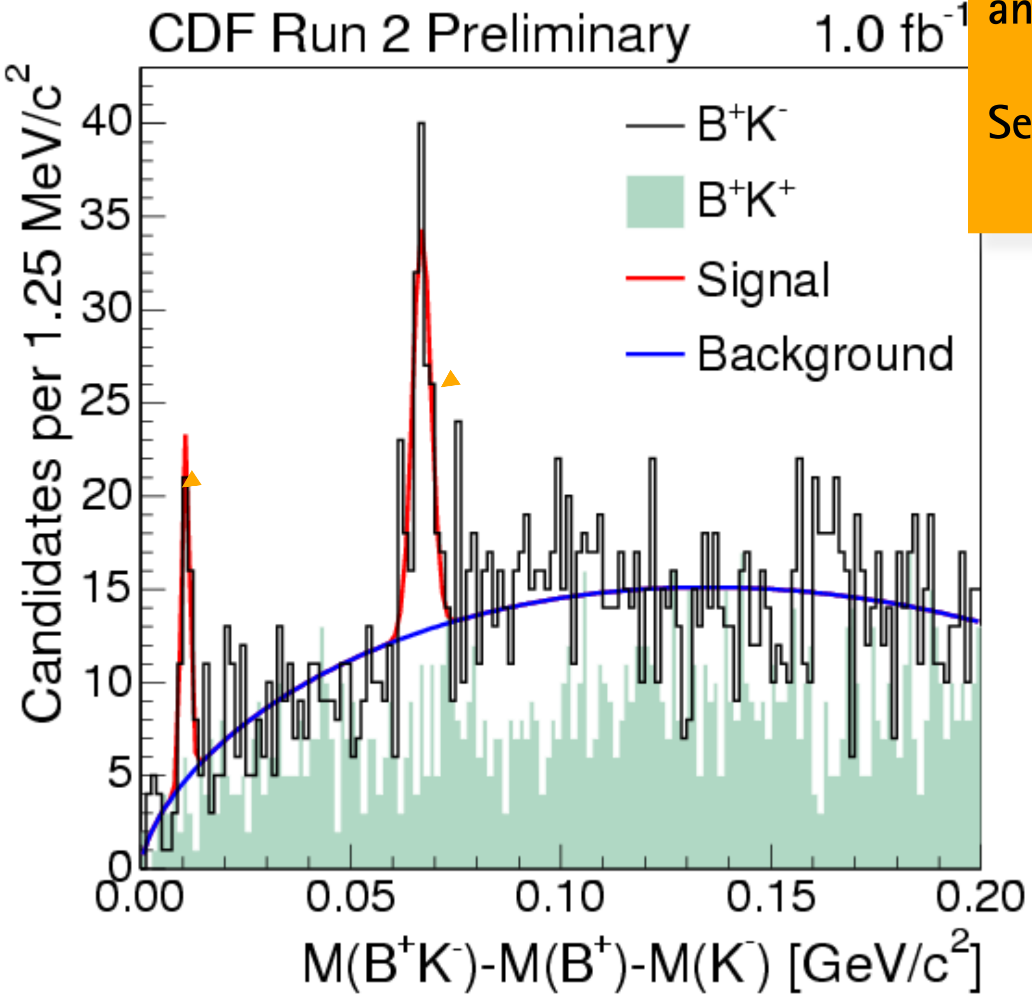


Just a few examples...



$\psi(2S)$ and $X(3872)$ in $J/\psi\pi^+\pi^-$

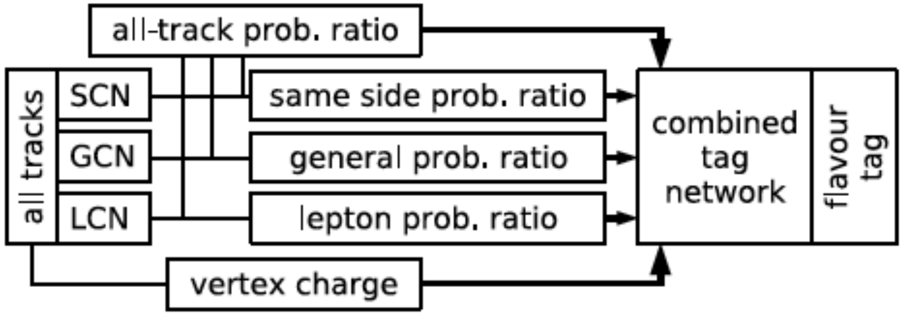
Just a few examples...



First observation of B_s1
and most precise of B_s2*

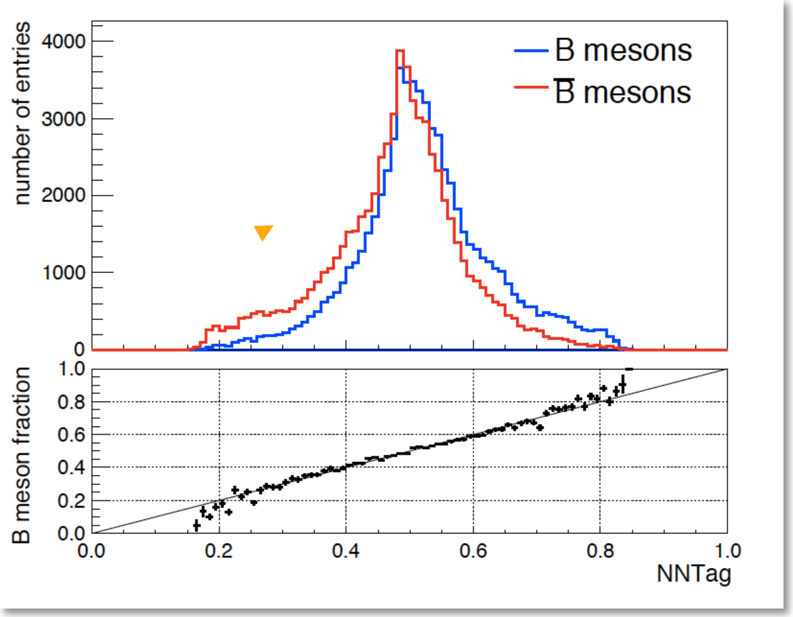
Selection using NeuroBayes®

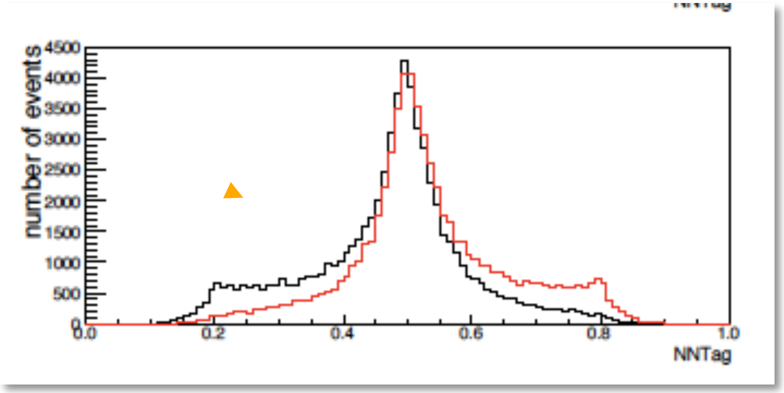
New CDF NeuroBayes B_s flavour tagger



$\mathcal{T} = \epsilon D^2 = 4.6\%$ **Without particle ID**

$\mathcal{T} = \epsilon D^2 = 6.7\%$ **Including particle ID**





Nice new methods…

Training with weighted events (e.g for J^{PC}-determination)

Data-only training with sideband subtraction (i.e. negative weights) and sPlot

Construction of weights for MC phase space events such that they are distributed like real data

Interpretation of NeuroBayes output as Bayesian a posteriori probability allows to avoid cuts on output variable but instead
-- inclusion into likelihood-fits (B-mixing, CP-violation)
-- usage with sPlot to produce “background free“ plots

Research on finding signals in data without having good background model

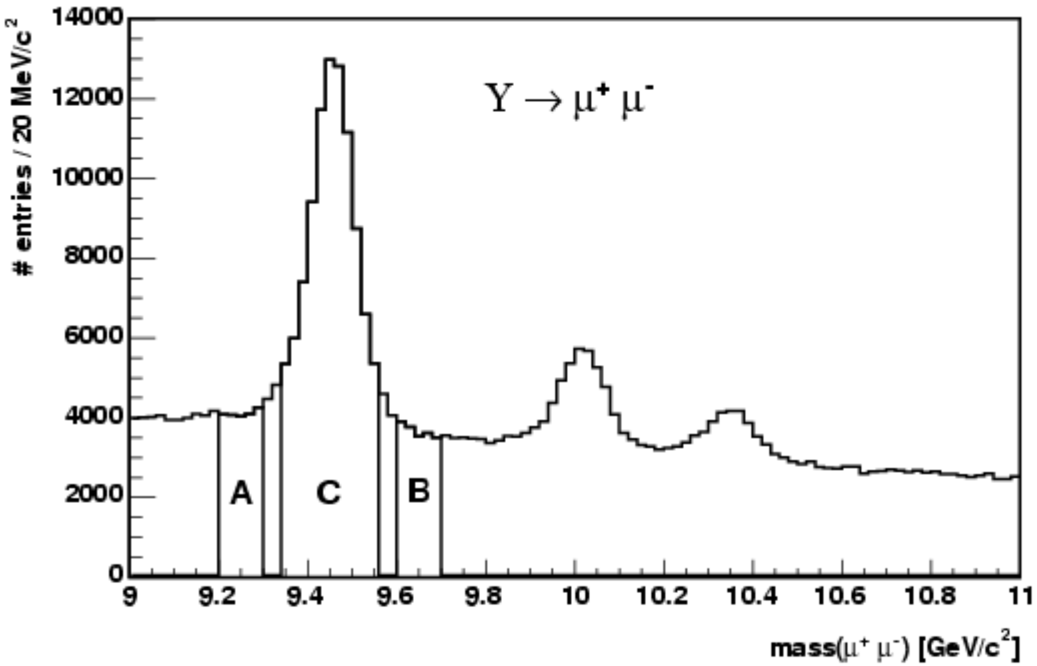
Hadron collider: Fast resonance S/B optimisation without MC:

Idea: **Training with background subtraction**

Signal: Peak region weight 1

Sideband region with weight -1

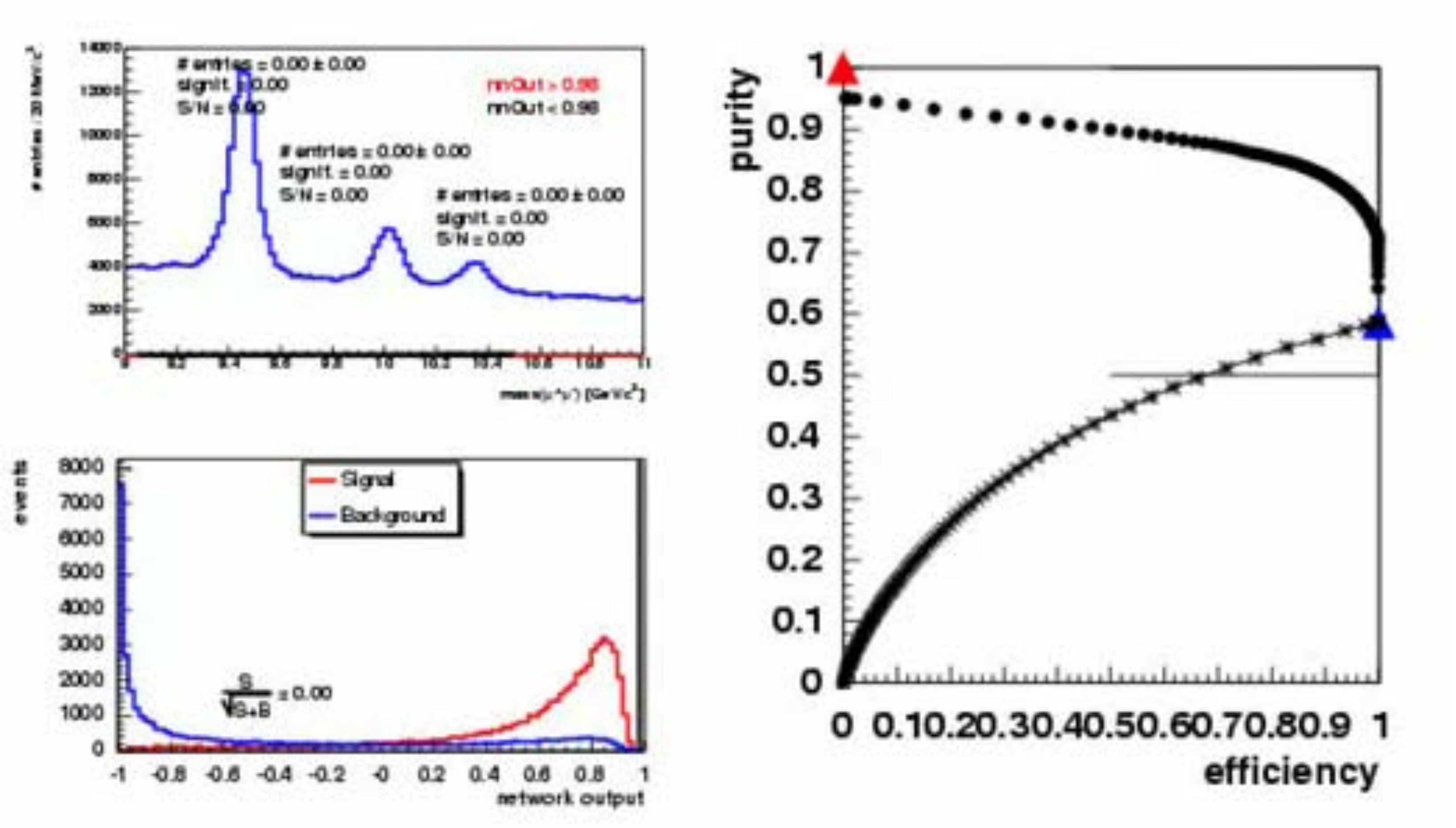
Background: Sideband region with weight 1



works very well!

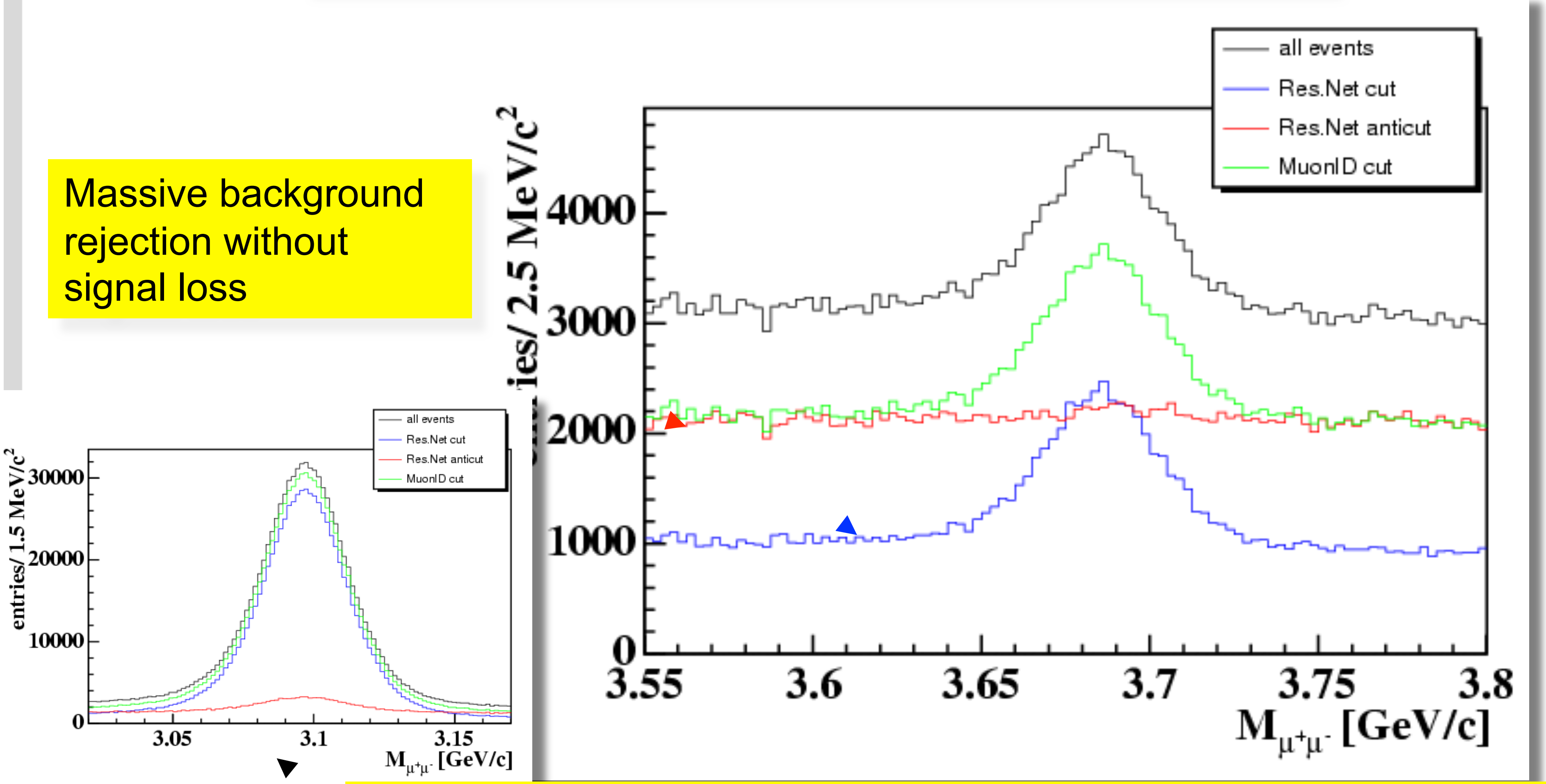
also for Y(2S) and Y(3S) !
Although just trained on Y(1S)

Example for data-only training (on1.resonance) (scan through cuts on network output)



NeuroBayes muon identification: ψ(2S) signal (CDF II)

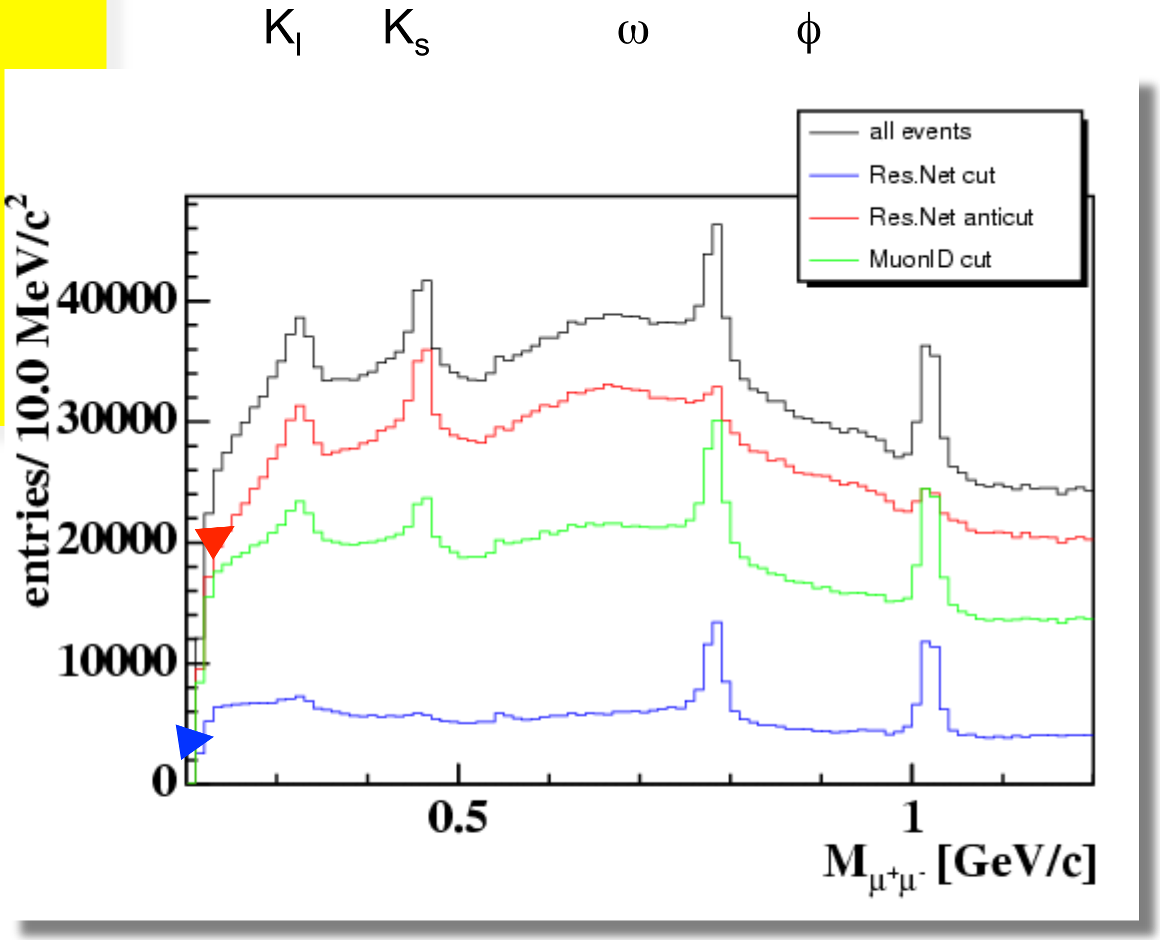
Massive background rejection without signal loss



also works on J/ψ (small effect due to already good S/B)

Low mass μμ resonances (CDF II)

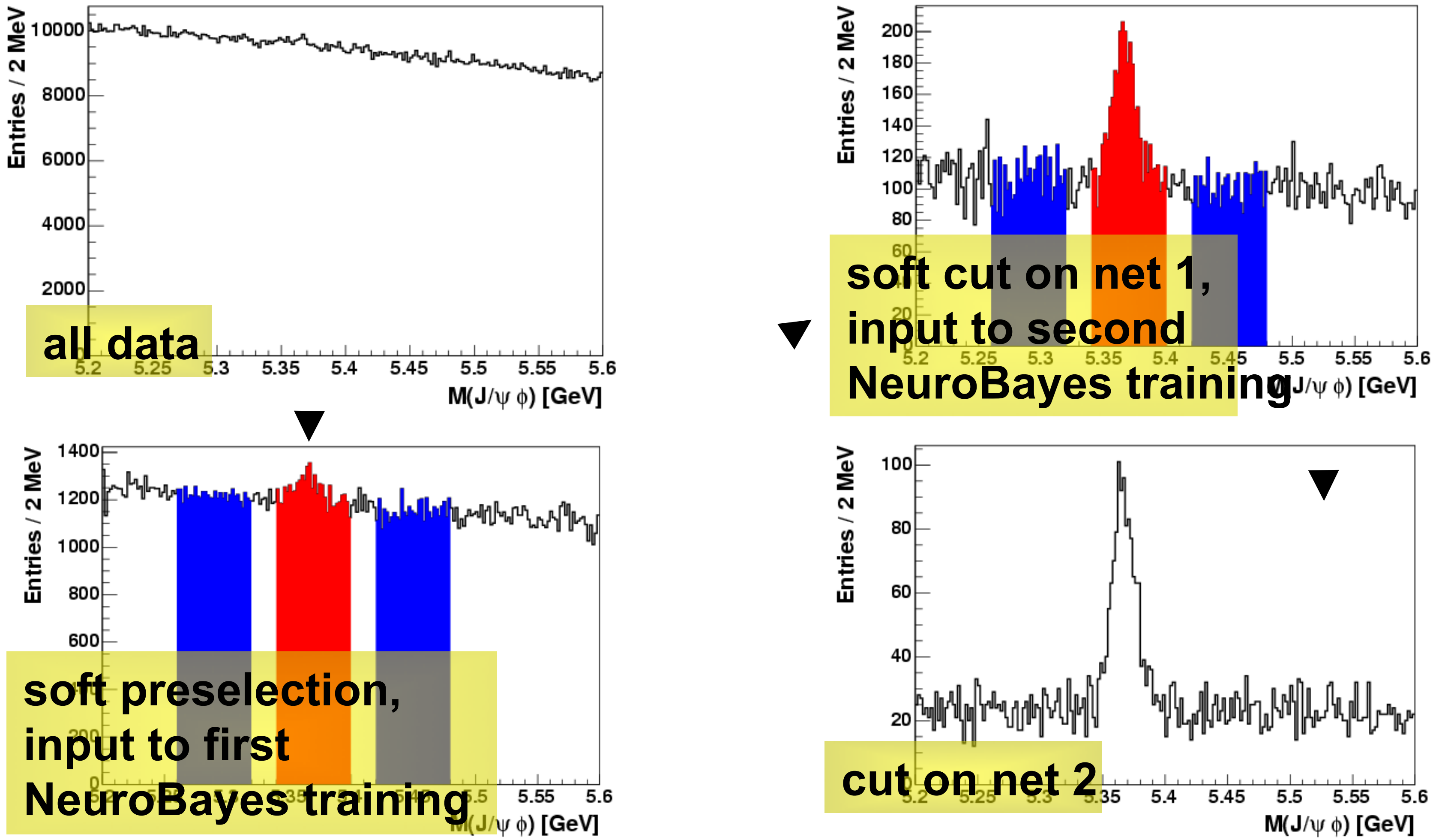
Cut on J/ψ network (distinguishes a resonance decaying into two muons from all the rest (non-muons, combinatorics))



reject

accept

NeuroBayes B_s to J/ψ Φ selection without MC (CDF II)



all data

▼

soft preselection, input to first NeuroBayes training

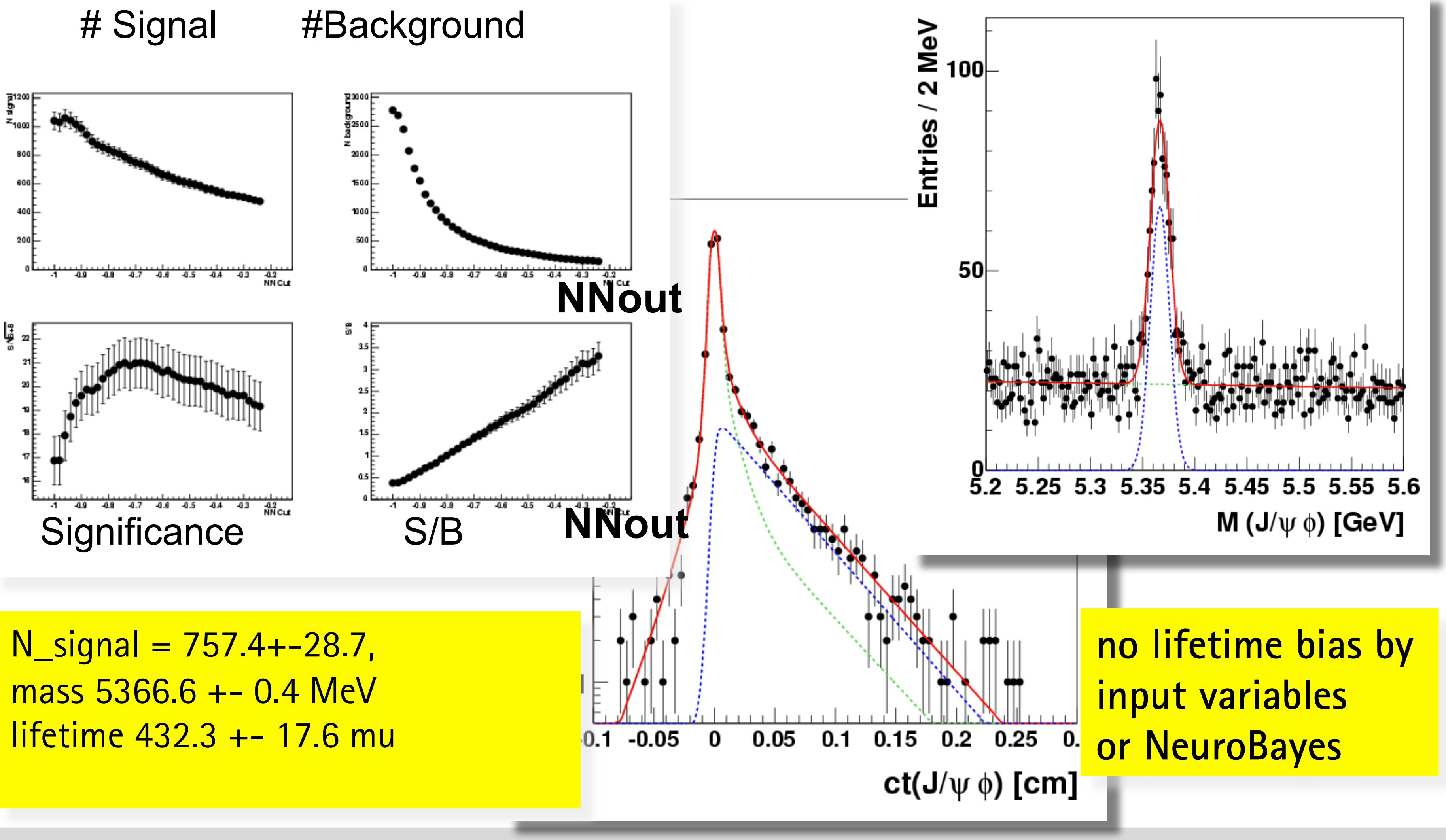
▼

soft cut on net 1, input to second NeuroBayes training

▼

cut on net 2

NeuroBayes B_s to J/ψ Φ selection without MC (CDF II)



N_signal = 757.4+-28.7,
mass 5366.6 +- 0.4 MeV
lifetime 432.3 +- 17.6 mu

no lifetime bias by input variables or NeuroBayes

Making MC for hadronic background without specific model: Multidimensional correlated regression using NeuroBayes

Use data in non-resonance region as signal
Use phase space MC as background

Train NeuroBayes network.
NN output O is Bayesian a posteriori probability that event stems from signal (i.e. data distribution) rather than phase space MC:

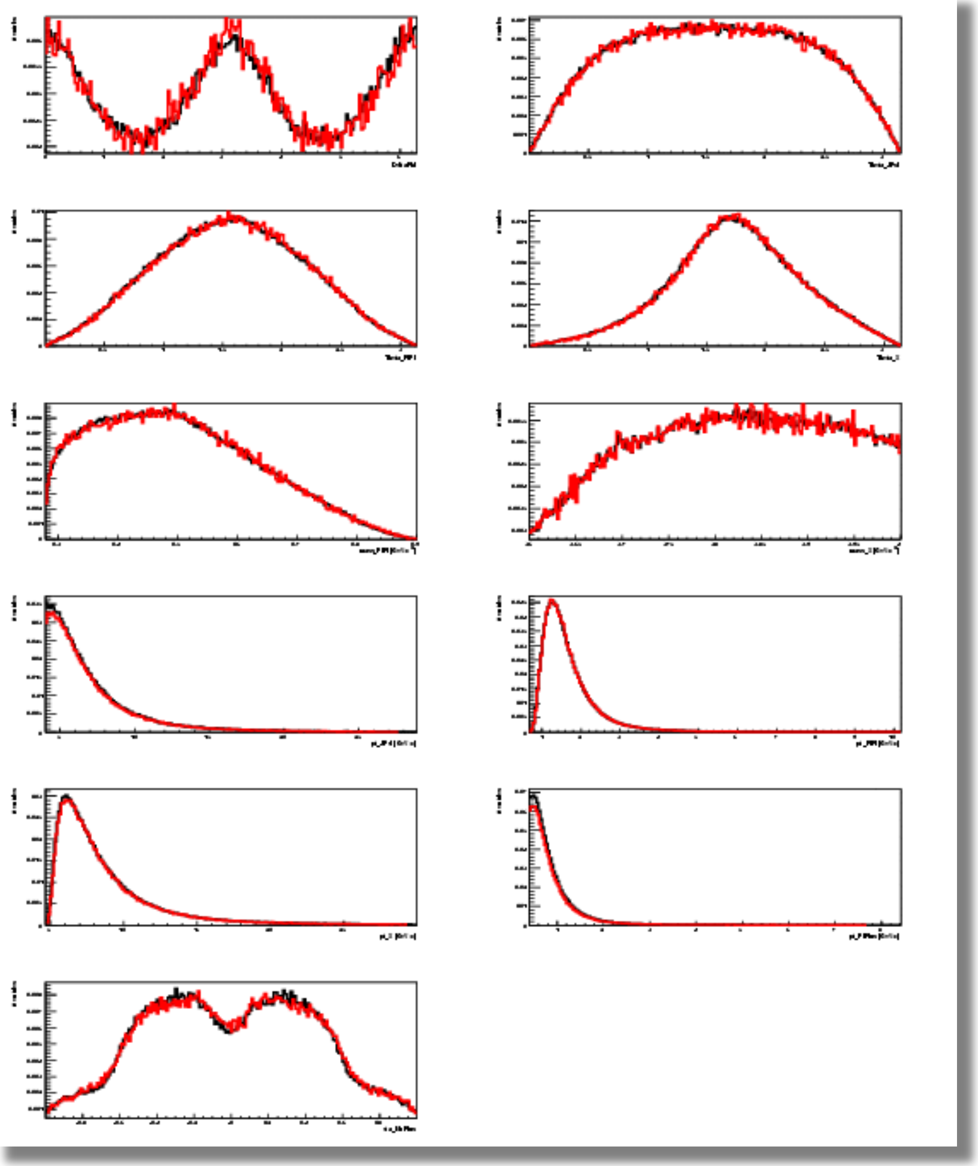
$$O=P(S) \text{ with } P(S)+P(B)=1$$

Calculate weight $W= P(S)/P(B) = O/(1-O)$

Phase space MC events with this weight W look like data!
MC modelling of complicated background is possible!
Opens new roads for likelihood fits

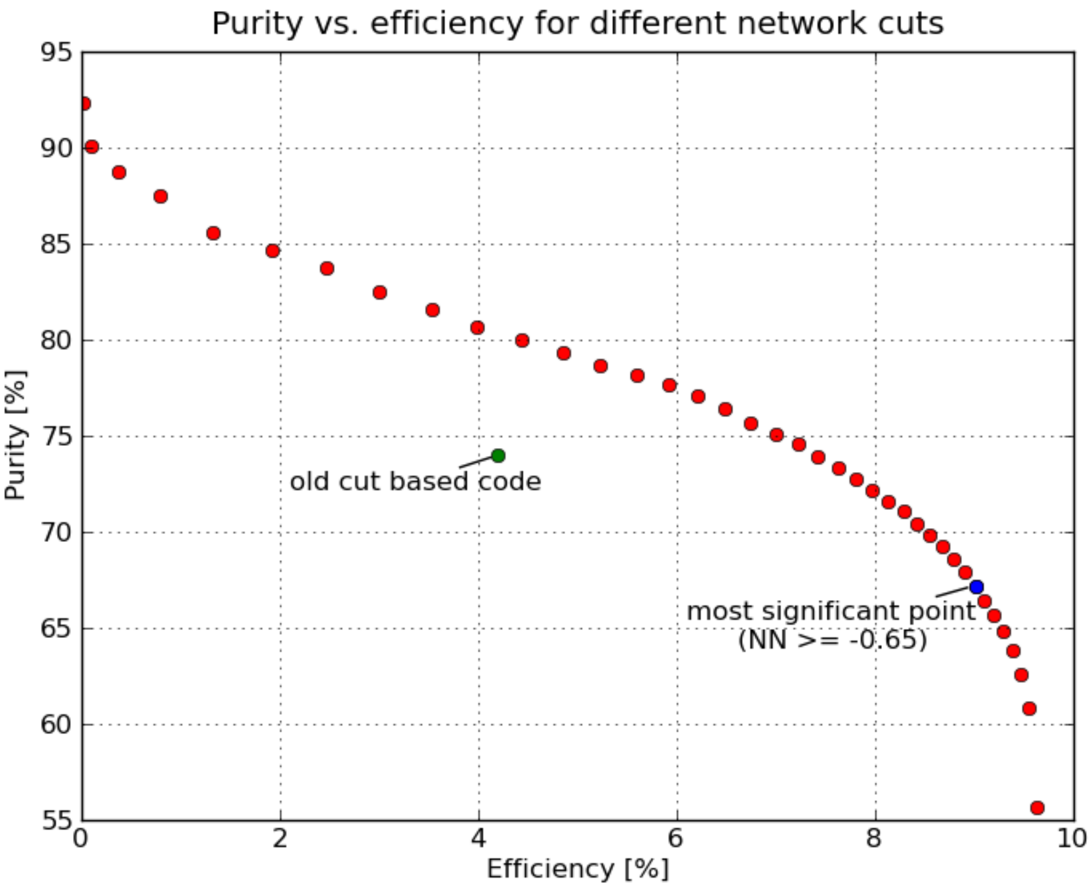
Some kinematical variable distributions
(CDF II $J/\psi\ \pi^+\pi^-$ selection)

Black: real data
Red: weighted phase space MC



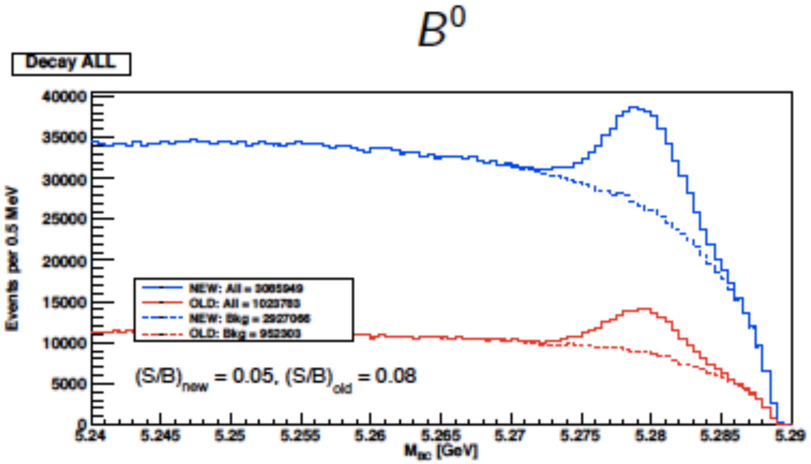
Belle B-factory
running very successfully since 2000.

KIT joined Belle Collaboration in 2008 and introduced NeuroBayes.

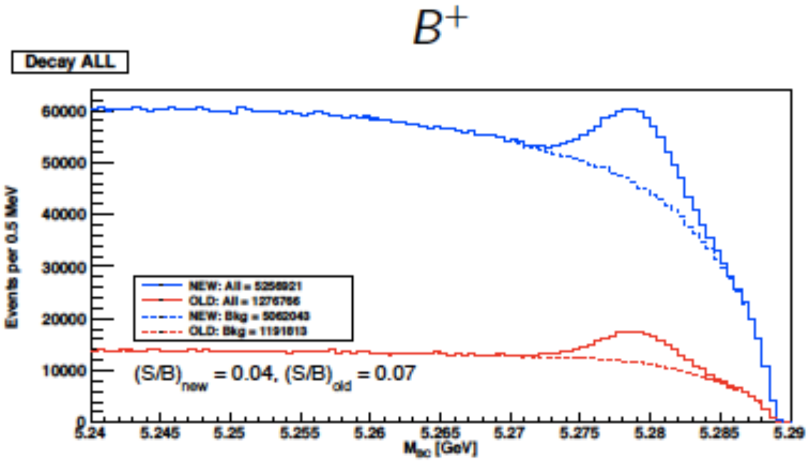


NeuroBayes enhances efficiency of flavour tagging calibration reaction B-> D* l ν by 71% at same purity

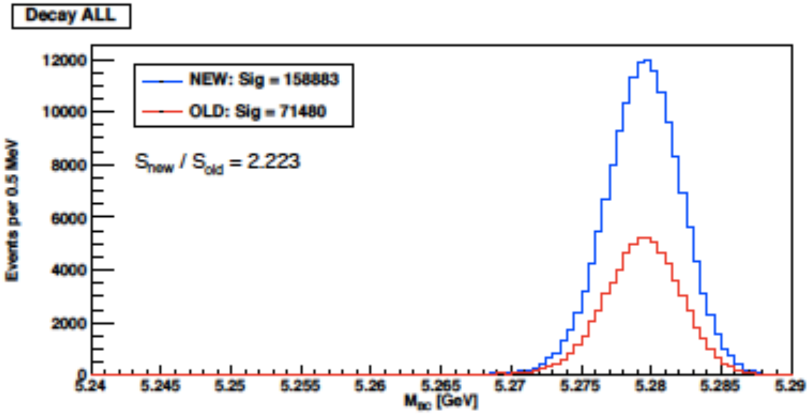
Belle: Full reconstruction of B mesons in >1000 decay chains.
Hierarchical system with > 100 NeuroBayes networks, fully automatic.
Preliminary gain about factor 2 compared to classical algorithm.



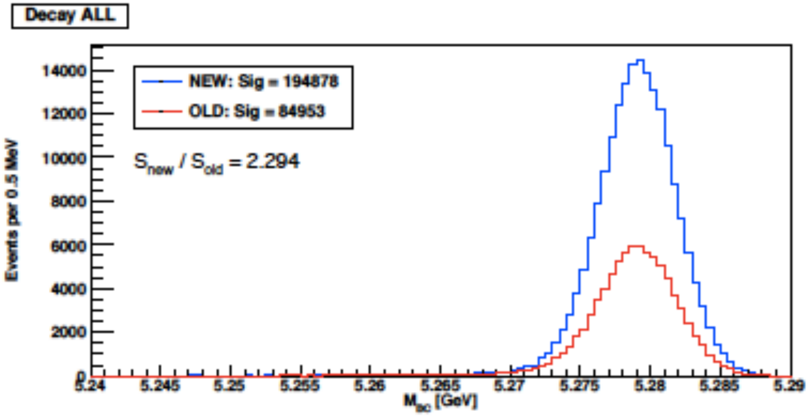
All candidates



All candidates

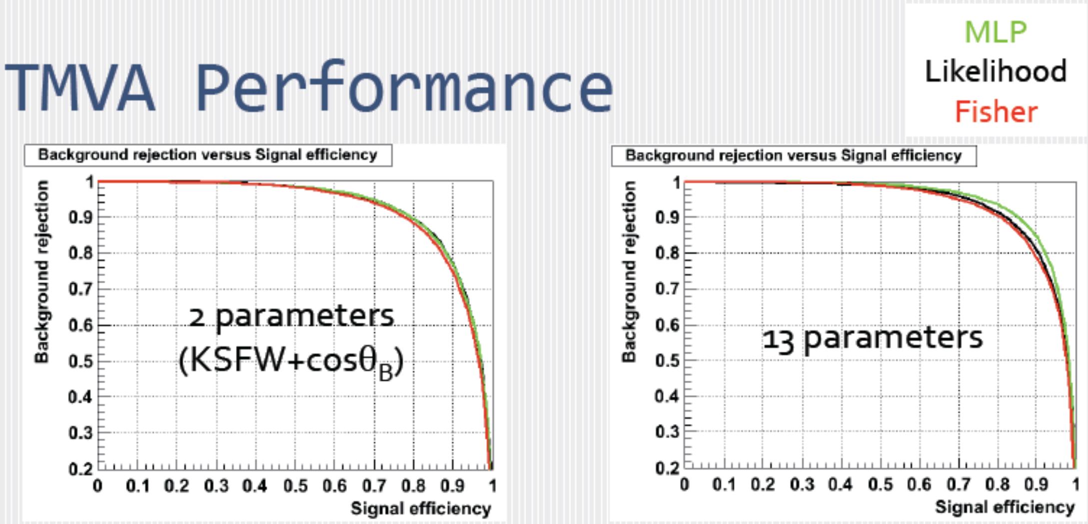


Correctly tagged candidates



Correctly tagged candidates

TMVA Performance



Belle: Non-BB- continuum-suppression for reconstruction of a rare B decay mode

Summary table for MLP results

BG rejection	Sig. efficiency (2 Pars.)	Sig. efficiency (13 Pars.)	Rate (13 Pars./2 Pars.)
99 %	43%	54%	1.26
95%	69%	76%	1.10
90%	79%	85%	1.08
80%	88%	92%	1.05

rformance

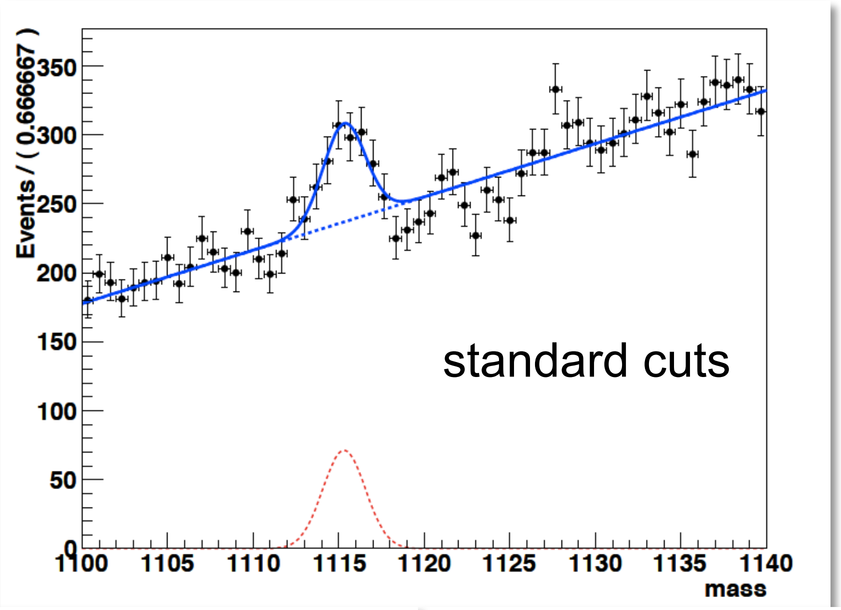
Improvement is larger especially for the analysis of small signal.

For DK ADS mode,
41% increase of signal is obtained with the same BG rejection!
(Of course we will re-optimize cut.)

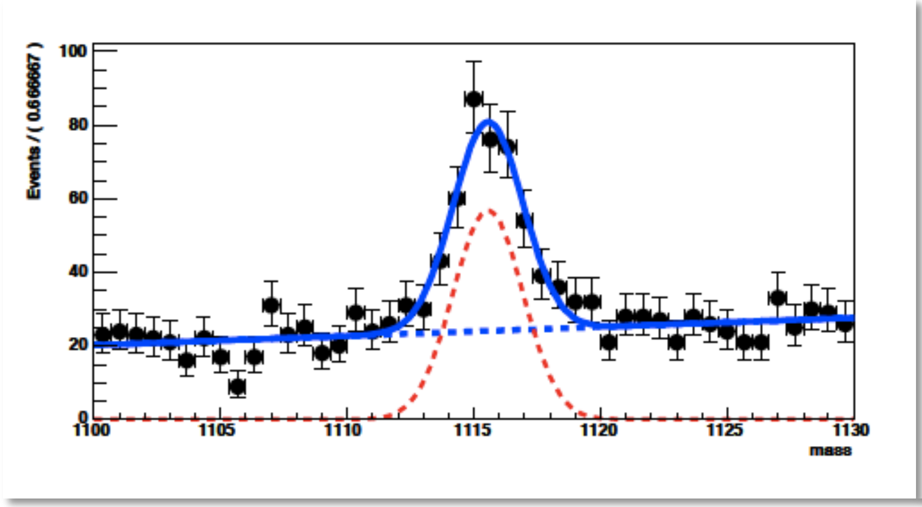
Comparison with TMVA algorithms shows advantage of NB

BG rejection	Sig. efficiency (2 Pars.)	Sig. efficiency (12 Pars.)	Rate (12 Pars./2 Pars.)
99 %	44%	62%	1.41
95%	69%	81%	1.17
90%	79%	89%	1.13
80%	88%	95%	1.08

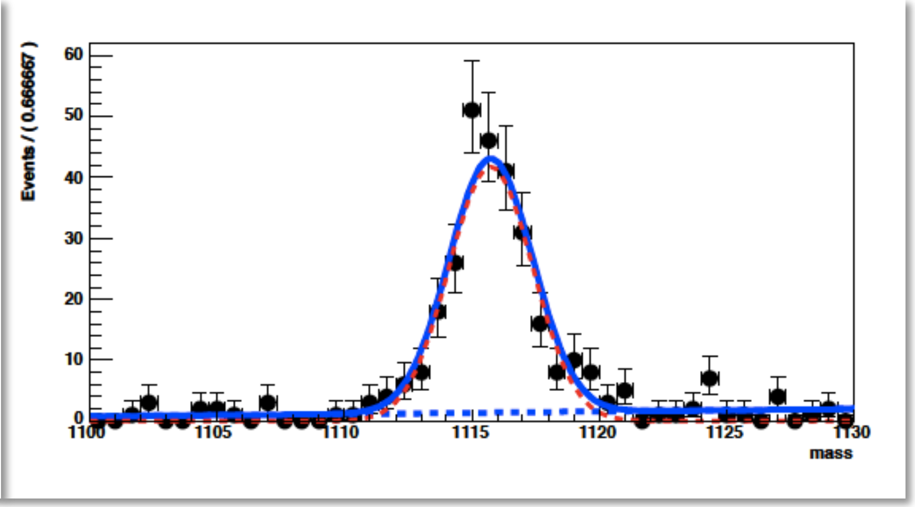
First application in LHCb: Λ baryon selection.



soft NB selection



hard NB selection



Bindings and Licenses

NeuroBayes® is commercial software belonging to Phi-T GmbH.
License files needed.
Special rates for public research.
Essentially free for high energy physics research.

NeuroBayes is nowunderway to become an officially supported CERN tool. It can be found at
/afs/cern.ch/sw/lcg/external/neurobayes/10.4

NeuroBayes core code written in Fortran.
Libraries for many platforms (Linux, Windows, …) available.
Bindings exist for C++, C, Fortran, java, lisp, python, etc.

Two code generators for easy usage exist.
New: Interface to root-TMVA available (classification only).

C++ NeuroBayes Teacher code fragment (1)

```
#include "NeuroBayesTeacher.hh“

//create NeuroBayes instance
  NeuroBayesTeacher* nb = NeuroBayesTeacher::Instance();
  const int  nvar = 14;     //number of input variables

  nb->NB_DEF_NODE1(nvar+1);       // nodes in input layer
  nb->NB_DEF_NODE2(nvar);         // nodes in hidden layer
  nb->NB_DEF_NODE3(1);            // nodes in output layer
  nb->NB_DEF_TASK("CLA");         // binominal classification
  nb->NB_DEF_ITER(10);            // number of training iterations

  nb->SetOutputFile("BsDsPiKSK_expert.nb");       // expertise file
  nb->SetRootFile("BsDsPiKSK_expert.root");       // histogram file
```

C++ NeuroBayes Teacher code fragment (2)

```
// in training event loop
    nb->SetWeight(1.0);  //set weight of event
    // set Target
    nb->SetTarget(0.0) ; // set Target, this event is BACKGROUND,
                         // else set to 1.

   InputArray[0] = GetValue(back,"BsPi.Pt");     // define input variables
   InputArray[1] = TMath::Abs(GetValue(back,"Bs.D0"));
...
   nb->SetNextInput(nvar,InputArray);
//end of event loop

nb->TrainNet();        //perform training
```

Many options existing, but this simple code usually already gives very good results.

C++ NeuroBayes expert code fragment

```
#include "Expert.hh"
...
   Expert* nb = new Expert("../train/BsDsPiKSK_expert.nb",-2);
...
   InputArray[0] = GetValue(signal,"BsPi.Pt");
   InputArray[1] = TMath::Abs(GetValue(signal,"Bs.D0"));
...
   Netout = nb->nb_expert(InputArray);
```

Documentation

Basics:

M. Feindt, A Neural Bayesian Estimator for Conditional Probability Densities, E-preprint-archive physics 0402093

M. Feindt, U. Kerzel, The NeuroBayes Neural Network Package, NIM A 559(2006) 190

Web Sites:

www.phi-t.de (Company web site, German & English)
www.neurobayes.de (English site on physics results with NeuroBayes & all diploma and PhD theses using NeuroBayes, talks, Manuals, FAQ and discussion forum)
www-ekp.physik.uni-karlsruhe.de/~feindt (some NeuroBayes talks can be found here under -> Forschung)

Summary

Neural networks are flexible and versatile multivariate tools.
However, some problems are known (step size dependence, long CPU time, possibility of overtraining)
All these problems are overcome in NeuroBayes.
NeuroBayes is meanwhile much more than a neural network.

Easy to use
Robust
Fast to ultra-fast
Produces only small calibration files (expertises)
Finds complicated multidimensional relationships with high probability
Generalises very well
Award winning performance
Steadily further developed professionally
(e.g. recent development n-dimensional probability densities in time O(n))

Use it and improve your analysis!
Knowledge important also outside physics (see talk this night).