Karlsruhe Institute of Technology

Statistische Methoden der Datenanalyse

Hans Dembinski
IEKP, KIT Karlsruhe

Topics for today

- Probability density estimation
- Unfolding of resolution effects from data distributions
- Multivariate classification

Probability density estimation

Probability density estimation

We now know two estimators of the probability density *f(x)*

Bootstrap estimator

$$\hat{f}_B(x) = \frac{1}{N}\sum_i \delta(x - x_i)$$

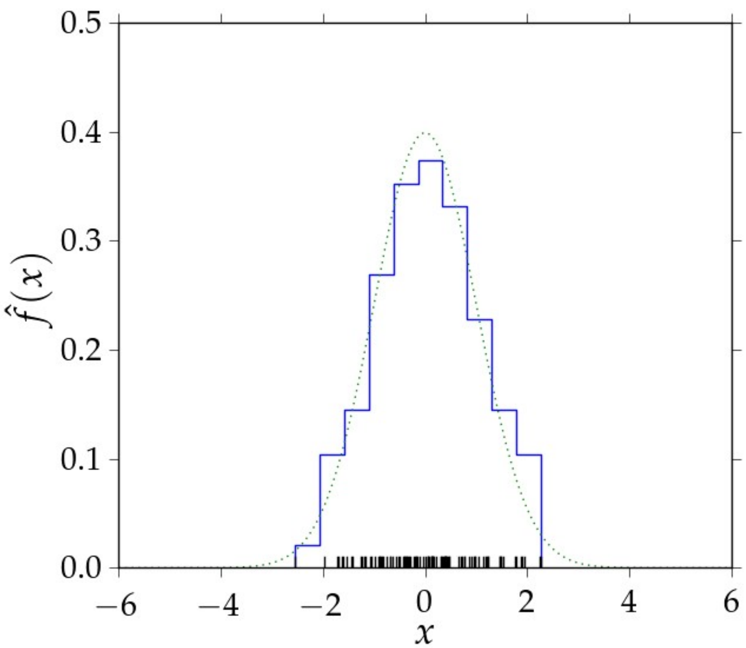
with data points x_i

Can only be used in integrals

Histogram estimator

$$\hat{f}(x) = \sum_l \frac{k_l/N}{x_{l+1} - x_l} H(x - x_l)\, H(x_{l+1} - x)$$

with k_l being the number of data points that fall into the interval (x_l, x_{l+1})



Bin sizes are usually selected ad hoc

Objective, optimal choice?

Probability density estimation

Criterion: integrated squared error (ISE), combines variance and bias of estimator

$$\mathrm{ISE} = \int \mathrm{d}x\,[\hat{f}(x) - f(x)]^2 = \int [\hat{f}(x)]^2 \mathrm{d}x - 2 \underbrace{\int \hat{f}(x)\, f(x) \mathrm{d}x} + \underbrace{\cancel{\int [f(x)]^2 \mathrm{d}x}}_{\text{constant}}$$

cross-validation

$$E[\hat{f}(x)] \simeq \frac{1}{N} \sum_i f_{(i)}(x_i)$$

Result for uniform histograms using cross-validation

$(x_{l+1} - x_l = h)$

$$\mathrm{ISE}(h) = \frac{2}{(n-1)h} - \frac{n+1}{n^2(n-1)h} \sum_l k_l^2$$

Minimize to get optimal *h*

Bootstrap and histogram estimates are useful, but fail at least if derivatives $f'(x),\ f''(x),\ \ldots$ are needed

Can we construct some kind of *smooth* histogram?

Kernel density estimation (KDE)

Idea: convolute bootstrap estimate with smooth kernel function *K(x)*

$$\hat{f}(x) = \int \mathrm{d}x' K\left(\frac{x-x'}{h}\right) \hat{f}_B(x') = \frac{1}{Nh}\sum_l K\left(\frac{x-x_i}{h}\right)$$

with $K(x) \geq 0, \int \mathrm{d}x K(x) = 1$

bandwidth

+ derivatives are well defined

– kernel density estimators are biased by construction

Many choices for *K(x)*, two stand out

Epanechnikov $K(x) = \begin{cases} \frac{3}{4}(1-x^2) & -1 \leq \mathrm{x} < 1 \\ 0 & \text{otherwise} \end{cases}$

+ optimal kernel
+ fast to compute
– finite support

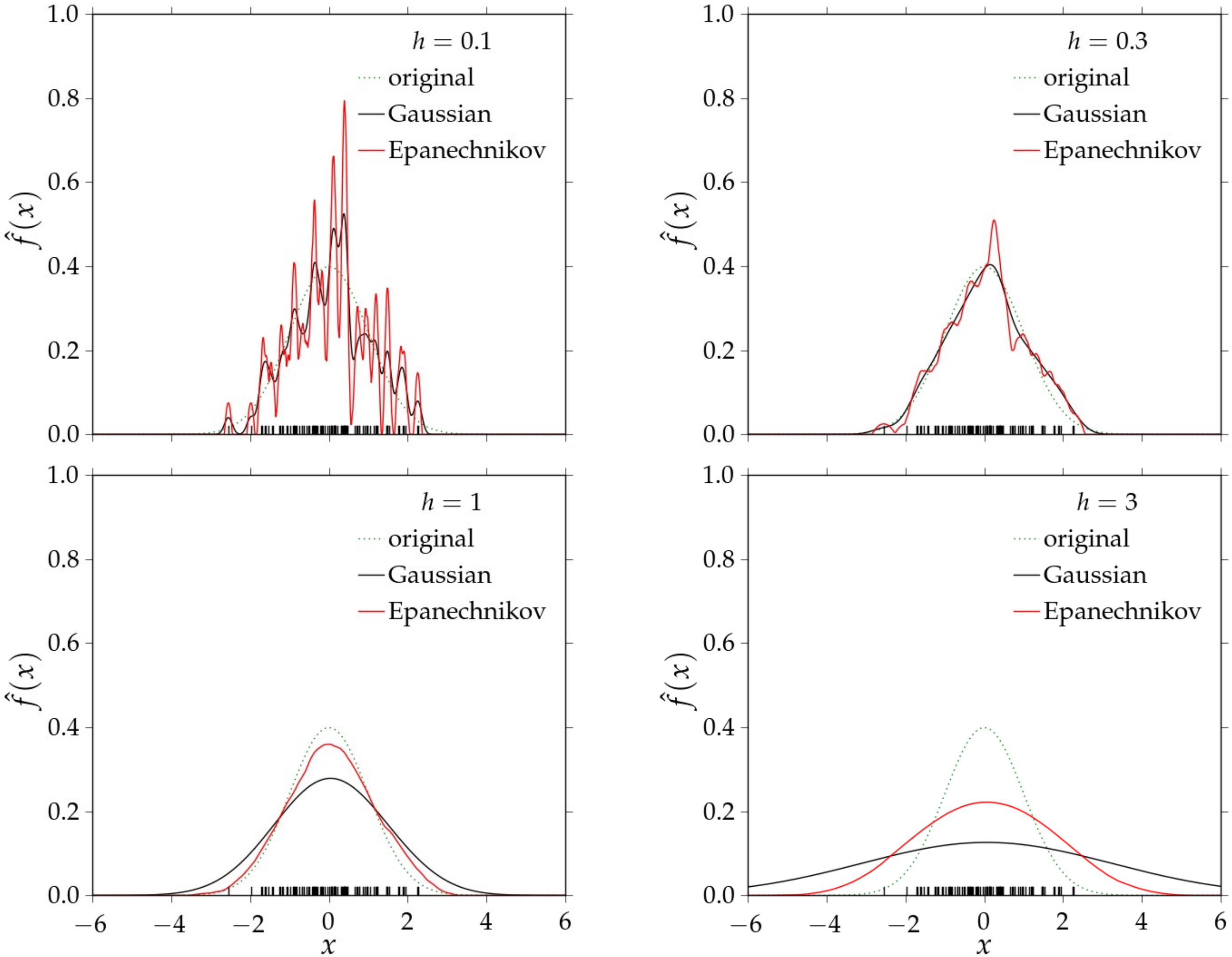
Gaussian $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$

+ infinite support
– slow to compute

In practice, the choice of the kernel does not matter much for the statistical quality
Important is the optimal choice of bandwidth *h* for the given data set

KDE examples

100 data points from Normal distribution
$\mu = 0,\ \sigma = 1$



KDE asymptotic properties

Mean integrated squared error (MISE) approaches zero at rate $O(N^{-4/(r+4)})$
dimension

$$\mathrm{MISE} = \int \mathrm{d}x\, E\Big[\big(\hat{f}(x) - f(x)\big)^2\Big] = \int V[\hat{f}(x)] + \big(E[\hat{f}(x) - f(x)]\big)^2 \mathrm{d}x$$

Convergence rate is slower than for parametric density estimates, e.g. $O(N^{-4/5})$ for $r = 1$

approximate asymptotic variance

$$V[\hat{f}(x)] \approx \frac{R(K)f(x)}{nh} - \frac{[f(x)]^2}{n}$$

approximate asymptotic bias

$$E[\hat{f}(x) - f(x)] \approx \frac{1}{2}\sigma_K^2 h^2 f''(x)$$

with $R(K) = \int \mathrm{d}x [K(x)]^2$ and $\sigma_K^2 = \int \mathrm{d}x\, x^2 K(x)$

→ optimal *h* balances variance and bias

$$h = \left(\frac{R(K)}{\sigma_K^4 R(f'')}\right)^{1/5} n^{1/5}$$ minimizes asymptotic MISE

$$n \to \infty \Rightarrow h \to 0 \Rightarrow E[\hat{f}(x) - f(x)] \to 0$$ KDEs are asymptotically unbiased

h depends on *f''(x)*, for normal distributed data $h_0 = 1.06\, s\, N^{-1/5}$ s = sample standard deviation

In practice, *h* can be obtained by leave-one-out cross validation
Another way is to use a surgurate estimate like h_0 to get $R(f'')$ in order to calculate *h*
Sheather and Jones found best plug-in estimate so far, see literature

Unfolding of resolution effects from data distributions

Resolution effects and unfolding

Special case of density estimation: unfolding of resolution effects from data distributions

Experimentalist wants to measure real-valued observable x, however detector does not measure true x but $y = x + \delta$ ⟵ random detector generated offset

Example

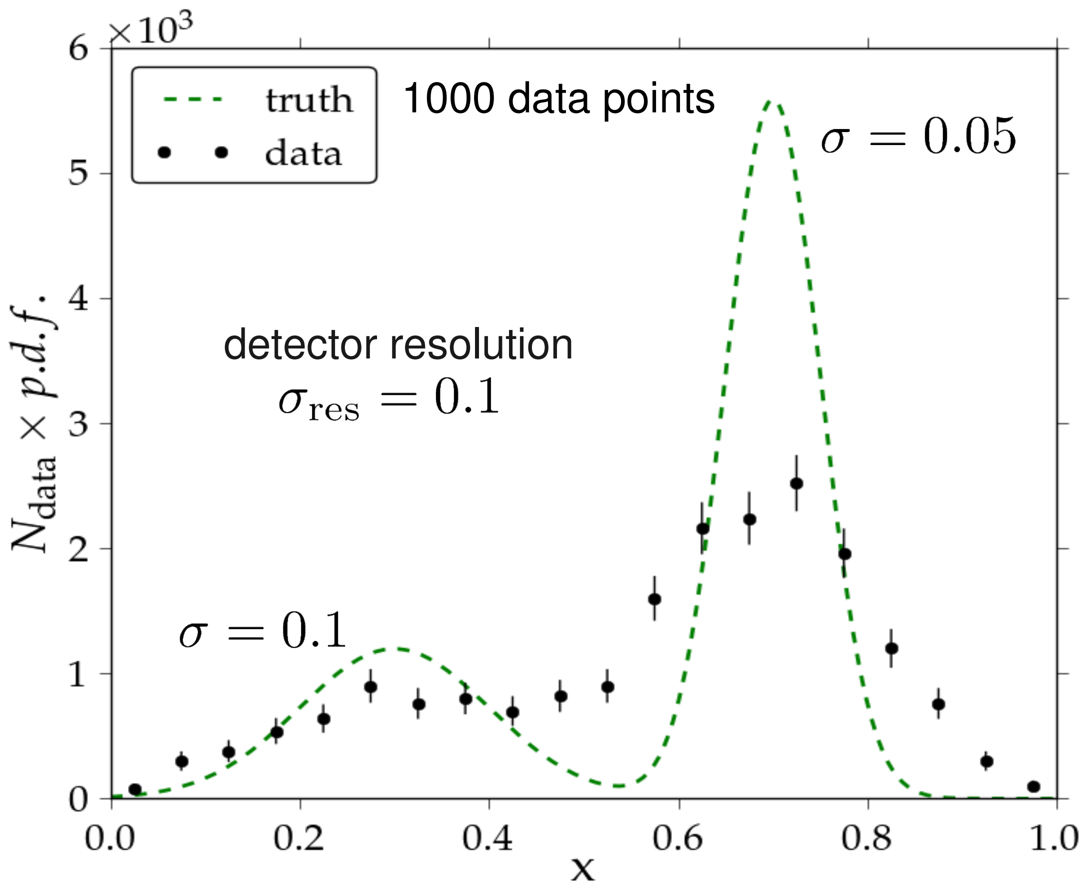
Two Gaussian peaks with random Gaussian resolution offset

How to obtain non-parametric estimate of $f(x)$ from observed y_i?

Approach: fit convoluted

$$g(y) = \int \mathrm{d}x\, \underbrace{K(y,x)}_{\text{resolution kernel}}\, f(x)$$

to data, using a flexible parameterization for $f(x)$

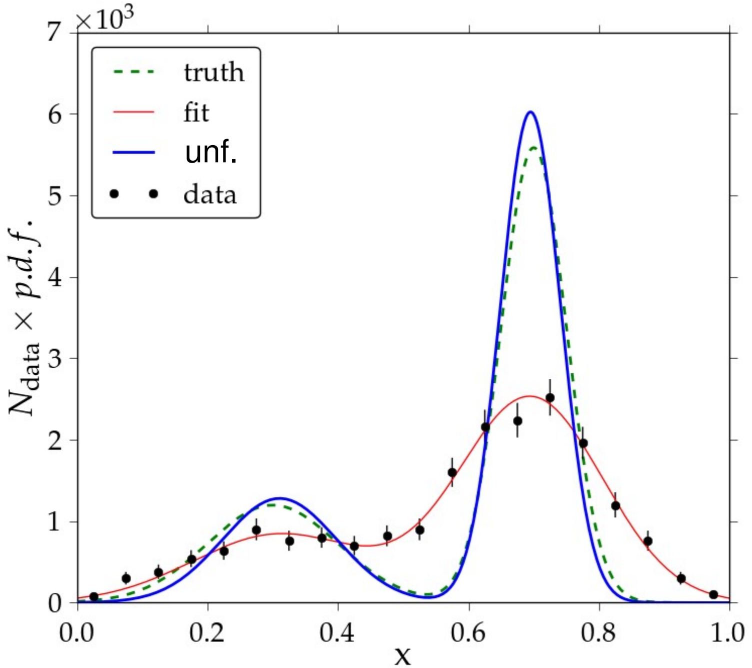


Parametric vs. non-parametric unfolding

Parametric case – model known

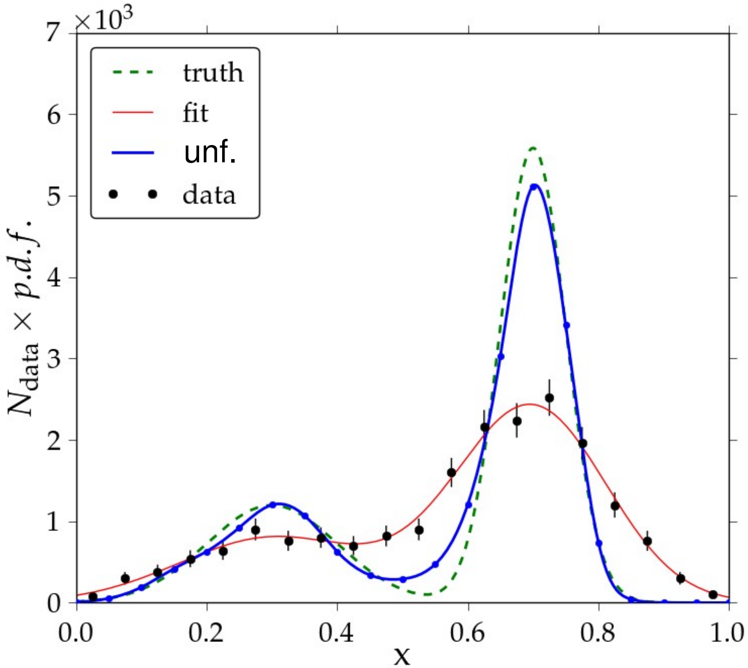
Prior information:
Solution = sum of 2 Gaussians

f(x) has 6 free parameters,
parametric model avoids over-fitting



Non-parametric case – model unknown

Prior information:
Solution is *regularized* in some way,
e.g. has to be smooth
f(x) has **many** free parameters,
regularization avoids over-fitting



Unfolding problem is ill-posed

Represent general solution as Fourier sum with many coefficents

$$f(x) = \frac{a_0}{2} + \sum_k a_k \cos\left(k\frac{2\pi}{\Delta x}x\right) + b_k \sin\left(k\frac{2\pi}{\Delta x}x\right)$$

Fold with kernel *K(y,x)*

$$g(y) = \int \mathrm{d}x\, K(y,x)\, f(x)$$

Fit to data

$$g(y) = \frac{\tilde{a}_0}{2} + \sum_k \tilde{a}_k \cos\left(k\frac{2\pi}{\Delta x}y\right) + \tilde{b}_k \sin\left(k\frac{2\pi}{\Delta x}y\right)$$

In case of Gaussian kernel

$$K(y,x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y-x)^2}{2\sigma}\right)$$

coefficient relation can be calculated analytically

$$\boxed{\begin{aligned} a_k &= \tilde{a}_k \exp\left(\frac{1}{2}k^2\sigma^2\right) \\ b_k &= \tilde{b}_k \exp\left(\frac{1}{2}k^2\sigma^2\right) \end{aligned}}$$

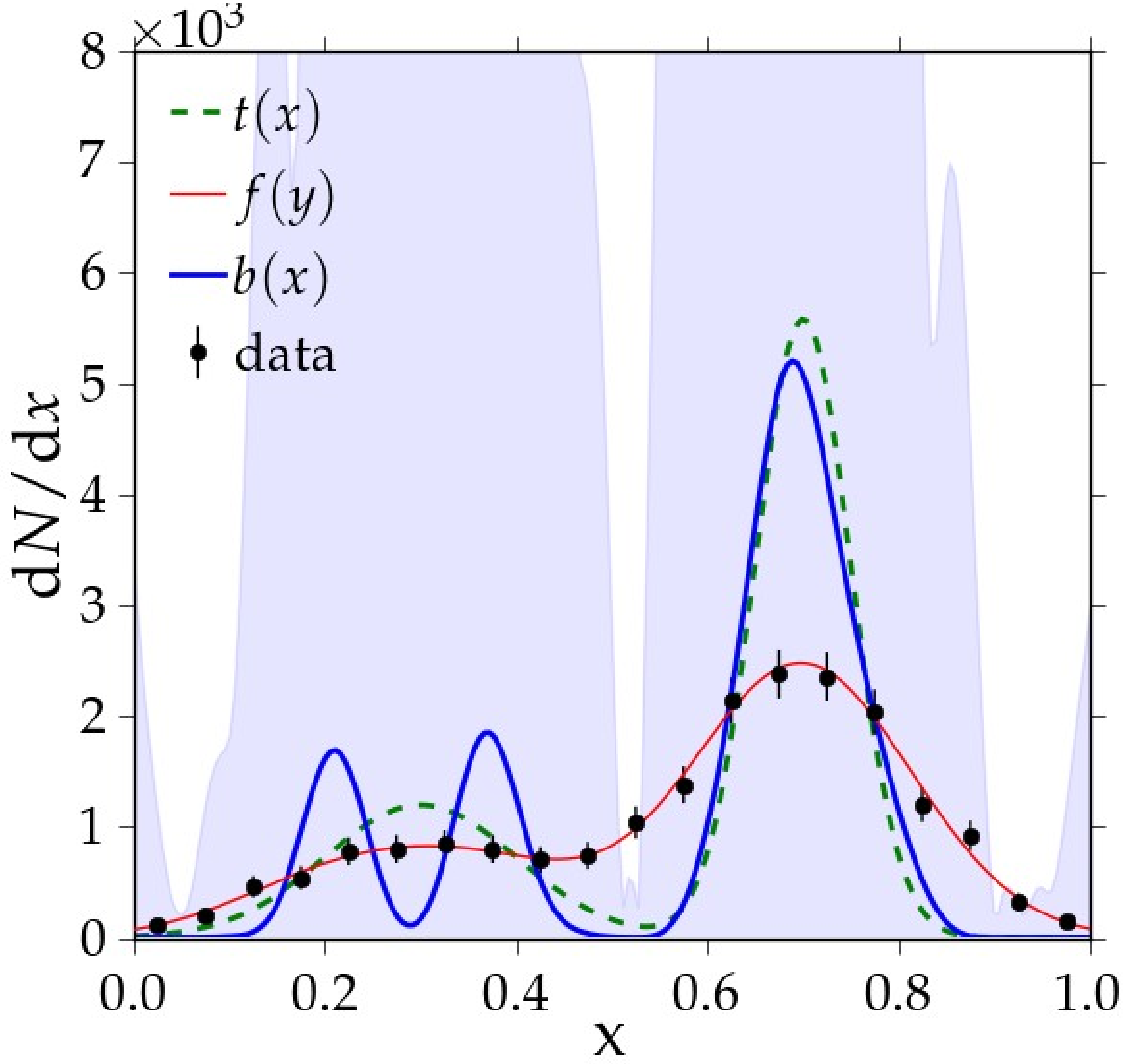
Folding acts as low-pass filter

Unfolding acts as high frequency amplifier
But: high frequencies are mostly noise

Unfolding w/o regularization

Without regularization, fit prefers solution dominated by high frequencies

wild oscillations
huge uncertainties and
strong correlations

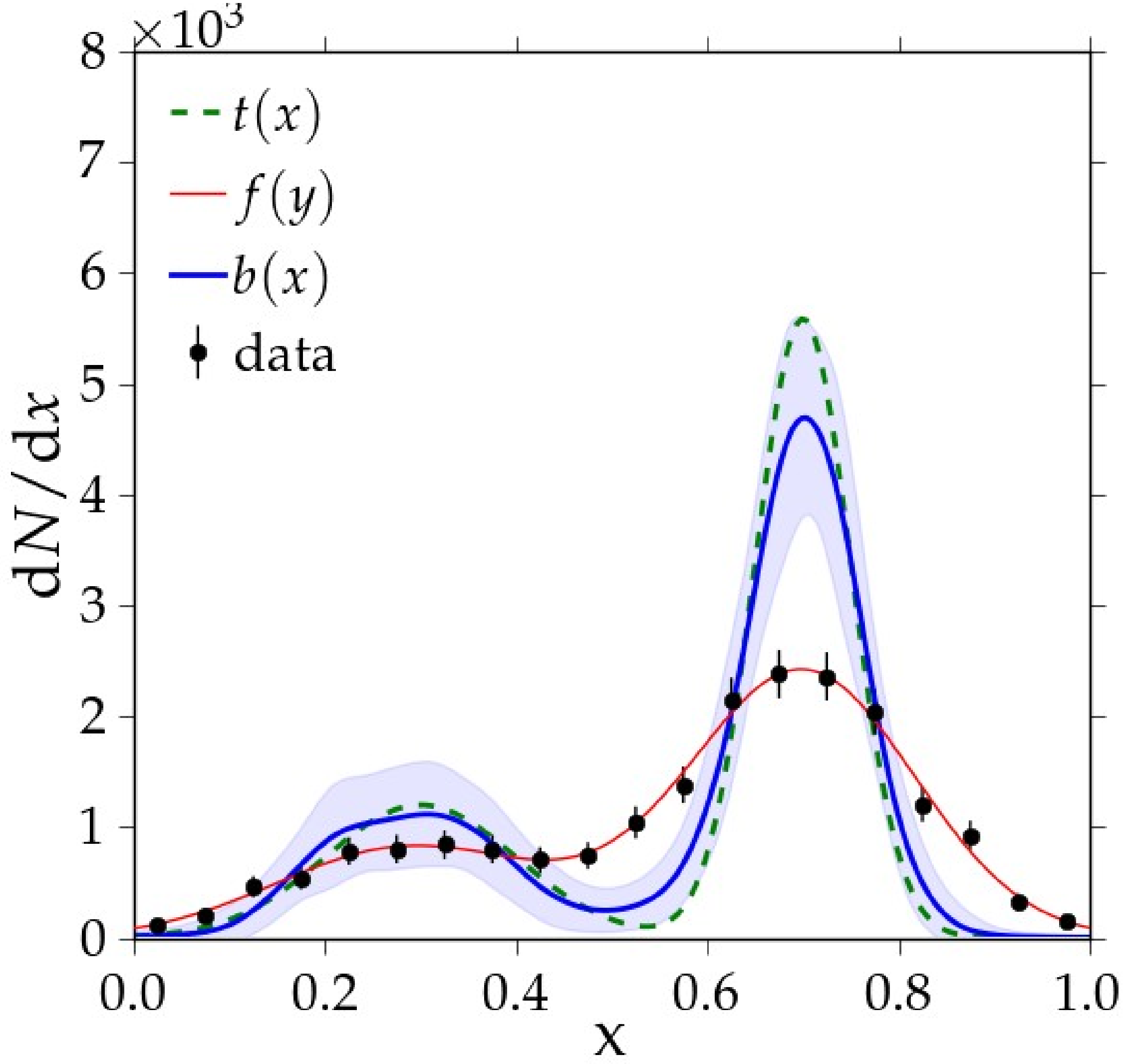


Unfolding with regularization

Regularization penalizes high frequencies in fit
→ smooth solution

But: solution is biased

Challenge: trade-off
bias vs. suppression of high frequencies



Unfolding algorithms

- Unfolding algorithms in Physics
  - Gold (1964)
  - Blobel (1984) – RUN algorithm
  - D'Agostini (1992) – Bayesian approach
  - Schmelling (1994) – Maximum entropy
  - SVD-based unfolding (1996)

Based on binned data, no automatic choice of regularization weight

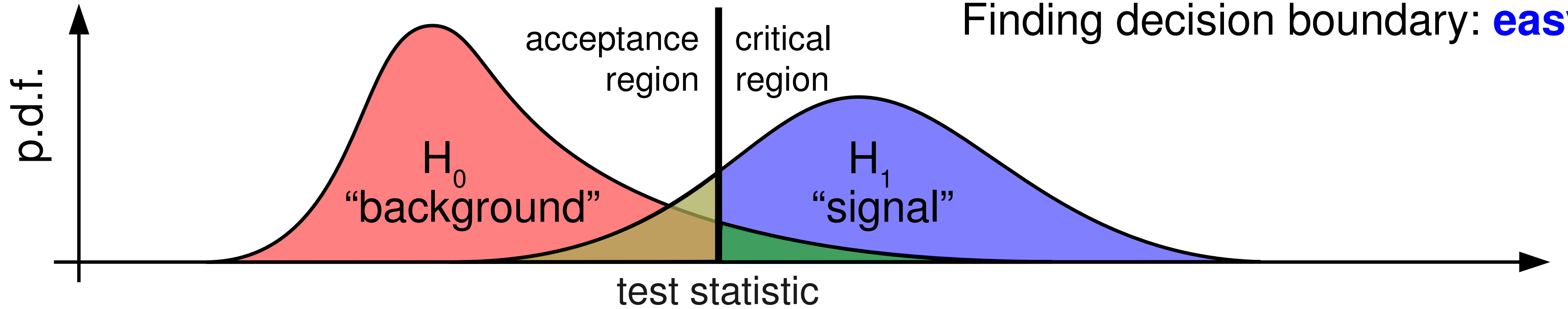
  - ARU (2011) – http://projects.hepforge.org/aru
    - Unbinned maximum-Likelihood fit
    - Regularization based on distance to original data distribution
      - Invariant to data transformations
      - Strength inverse to local density
    - Automatic choice of regularization weight
      - Criterion: minimum MISE
    - Full analytic uncertainty calculation

Multivariate classification

Multivariate classification

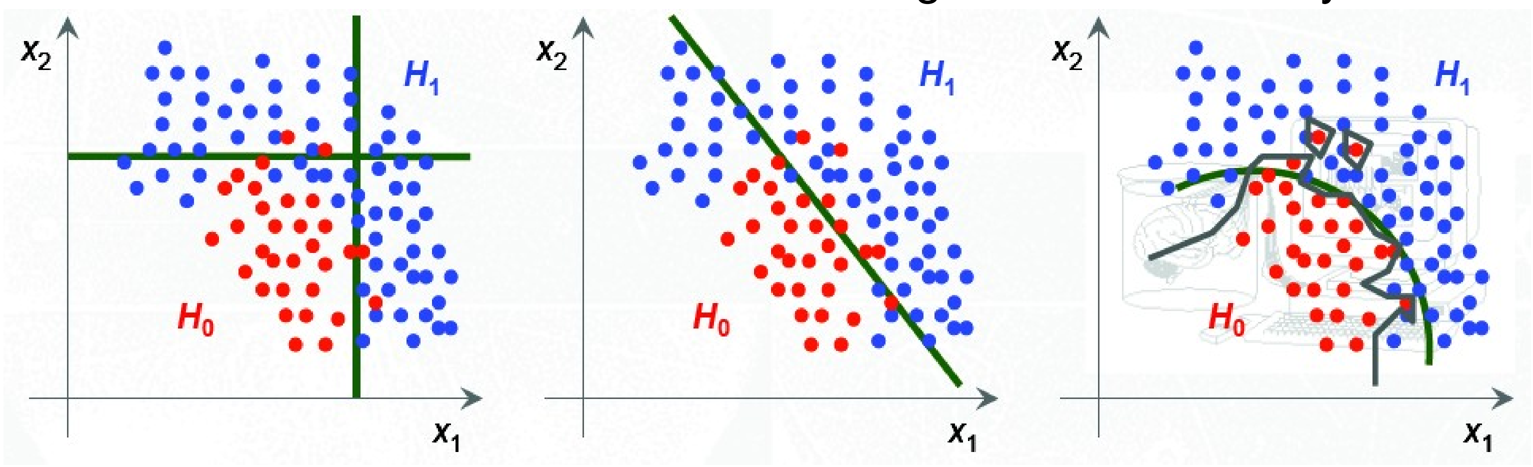
Hypothesis test between two fully defined models in **1d**

Finding decision boundary: **easy**



Hypothesis test in **nd** – typically p.d.f.s not available, only training data sets

Finding decision boundary: **challenging**



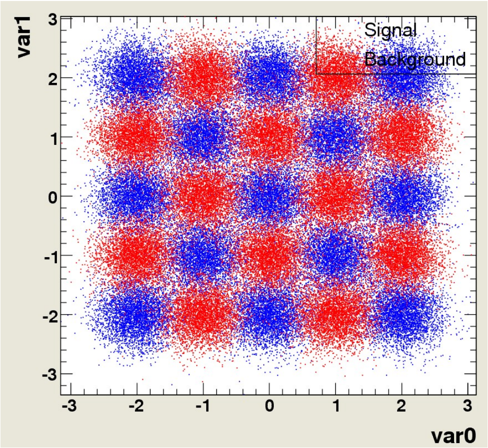
Manually placed rectangular cuts may be inefficient and nd-data hard to visualize

Classification methods

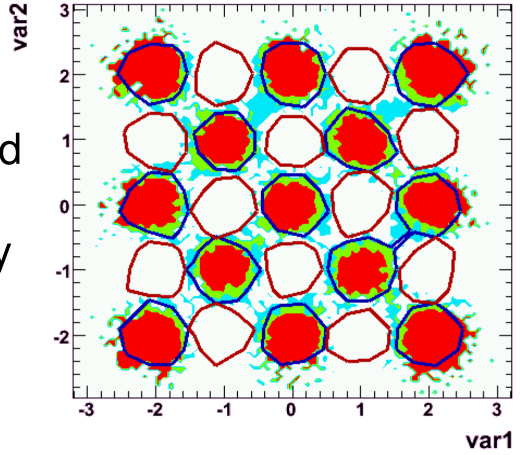
Power of the hypothesis test now depends on form of decision boundary
Classification methods find good decision boundary automatically

Methods can be compared in **purity vs. efficiency plot**

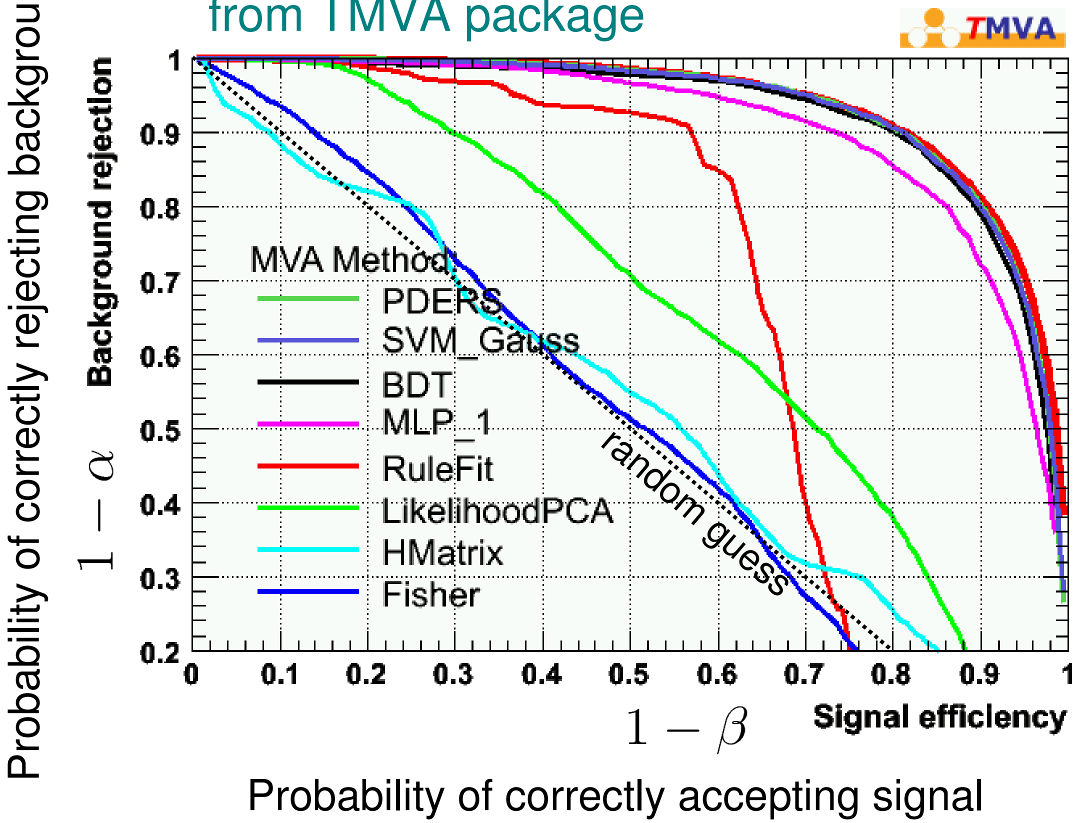
training data



computed decision boundary



Checkerboard example from TMVA package



Popular classification methods

- Probability density estimator range-search (PDERS)
  - Estimate joint p.d.f. of signal and background with KDEs
  - Use likelihood ratio of p.d.f.s to find decision boundary
- Boosted decision tree (BDT)
  - Repeat two steps many times
    - Build tree of optimal rectangular cuts from weighted data set → save tree
    - Increase weight of misclassified events (Boosting)
  - Decide via majority vote over all cut trees
- Artificial neural network (ANN)
  - Fit flexible parametrization of decision boundary to training data set
- Support Vector Machine (SVM)
  - Find hyper-plane that best separates two event classes after projecting data points into higher dimensional space

Comparison of classification methods

Criteria		Classifiers						
		Cuts	Projected likelihood	PDERS/ k-NN	Fisher	MLP	BDT	SVM
Perfor-mance	no / linear correlations	😐	☺	☺	☺	☺	😐	☺
	nonlinear correlations	😐	☹	☺	☹	☺	☺	☺
Speed	Training	☹	☺	☺	☺	😐	☹	😐
	Response	☺	☺	☹/😐	☺	☺	☹	😐
Robust-ness	Over-training	☺	😐	😐	☺	☹	☹	😐
	Weak input variables	☺	☺	☹	☺	😐	😐	😐
Curse of dimensionality		☹	☺	☹	☺	😐	☺	😐
Transparency		☺	☺	😐	☺	☹	☹	☹

table adapted from Andreas Hoecker,
Lecture on multivariate analysis techniques, Karlsruhe 2009

SVMs are recommended

Artificial neural networks

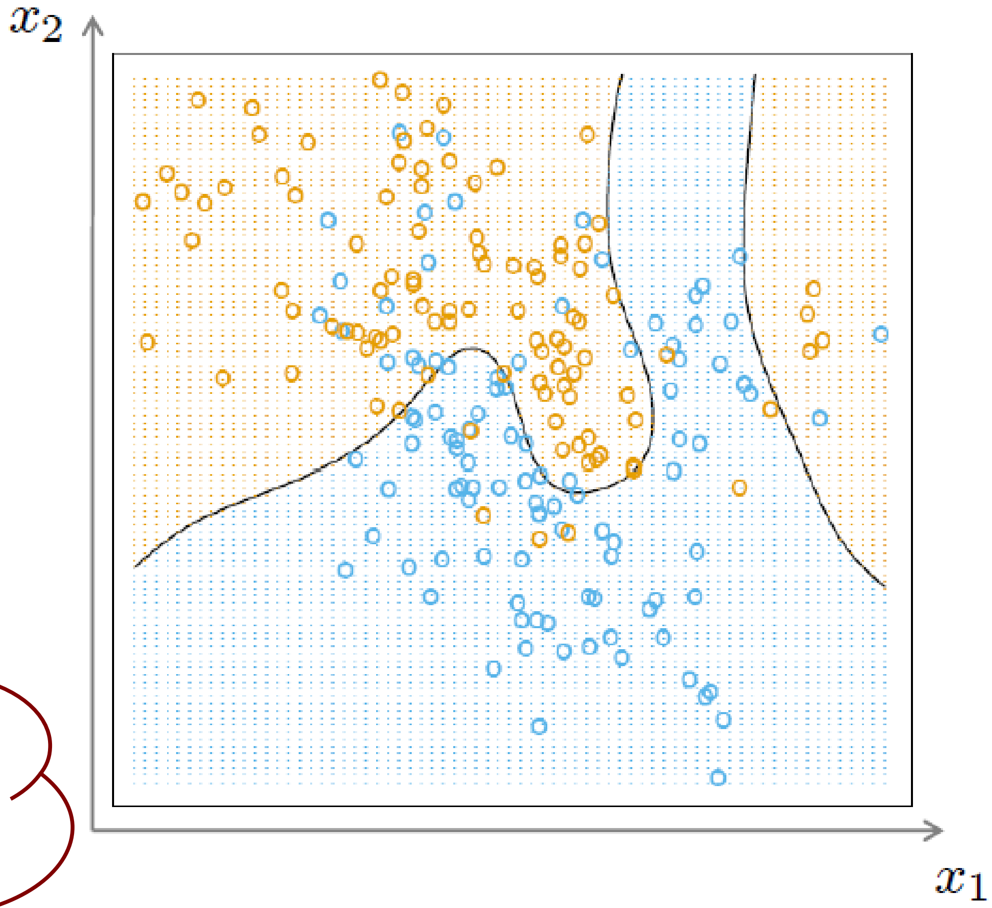
General problem

Find decision boundary
to best separate
two classes A, B of events

– based on a finite training sample
– and in such a way that
boundary generalizes well
to new samples

If you are thinking about “fitting a empirical separation function”, you are on the right track

example with non-linear decision boundary



slide adapted from Jan Therhaag, TMVA workshop, CERN 2011

Linear classifier

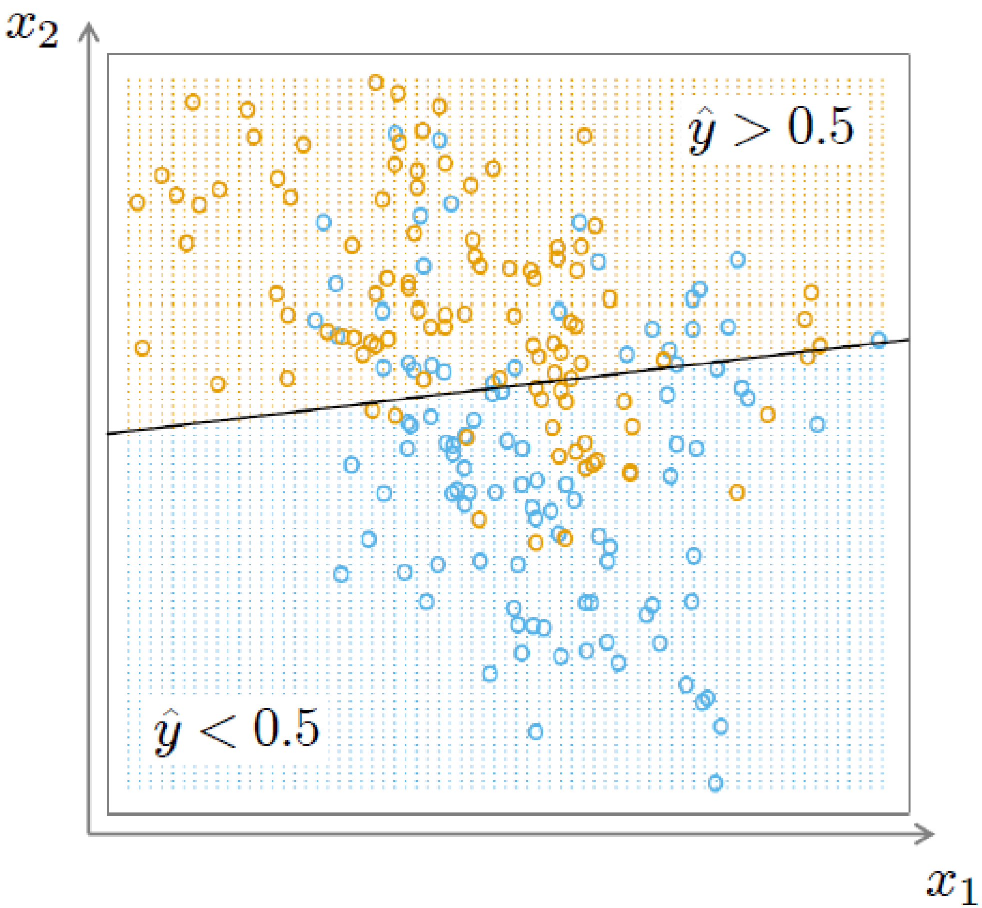
Code classes A, B in binary variable

$$y \in \{0, 1\}$$

A B

$$\hat{y} = \sum_i w_i \, x_i + w_0 = \vec{w} \cdot \vec{x}$$

with $\vec{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_N \end{pmatrix}$



Fit this to data to obtain weight vector

Then, decision boundary given by

$$\hat{y} = \vec{w} \cdot \vec{x} \begin{cases} \leq 0.5 \rightarrow \text{class A} \\ > 0.5 \rightarrow \text{class B} \end{cases}$$

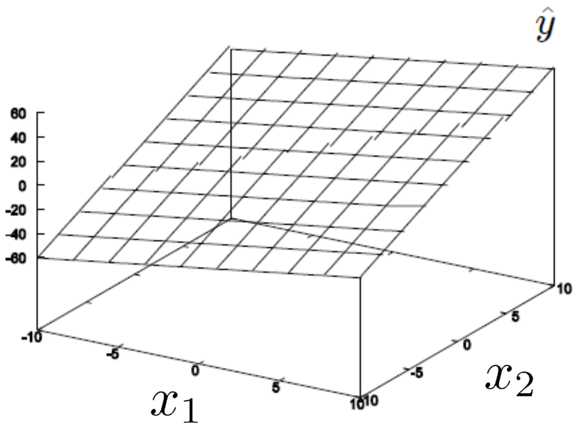
slide adapted from Jan Therhaag, TMVA workshop, CERN 2011

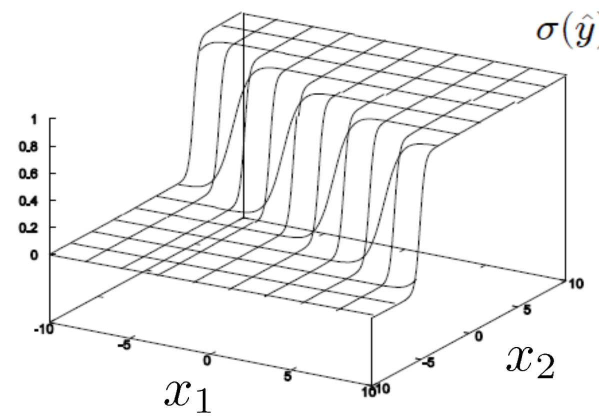
Linear classifier → Perceptron

We want to interpret $\hat{y}$ as a probability → apply **sigmoid transformation**

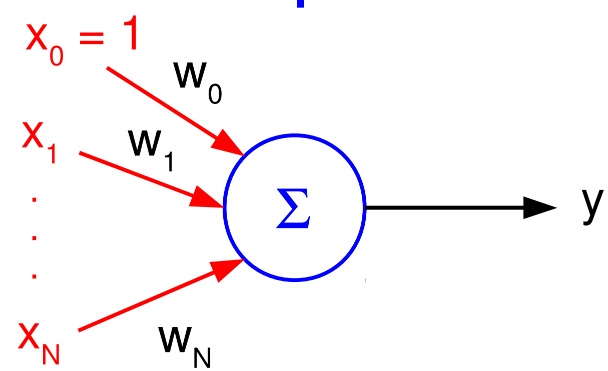
$$\hat{y} \to \sigma(\hat{y}) = \frac{1}{1+\exp(-\hat{y})} = \frac{1}{1+\exp(-\vec{w}\cdot\vec{x})}$$

bounded between [0,1]





Perceptron



$\sigma(\hat{y})$ can be interpreted as $P(\text{class B}|\vec{x})$

$$P(\text{class A}|\vec{x}) = 1 - P(\text{class B}|\vec{x})$$
$$= \sigma(-\hat{y})$$

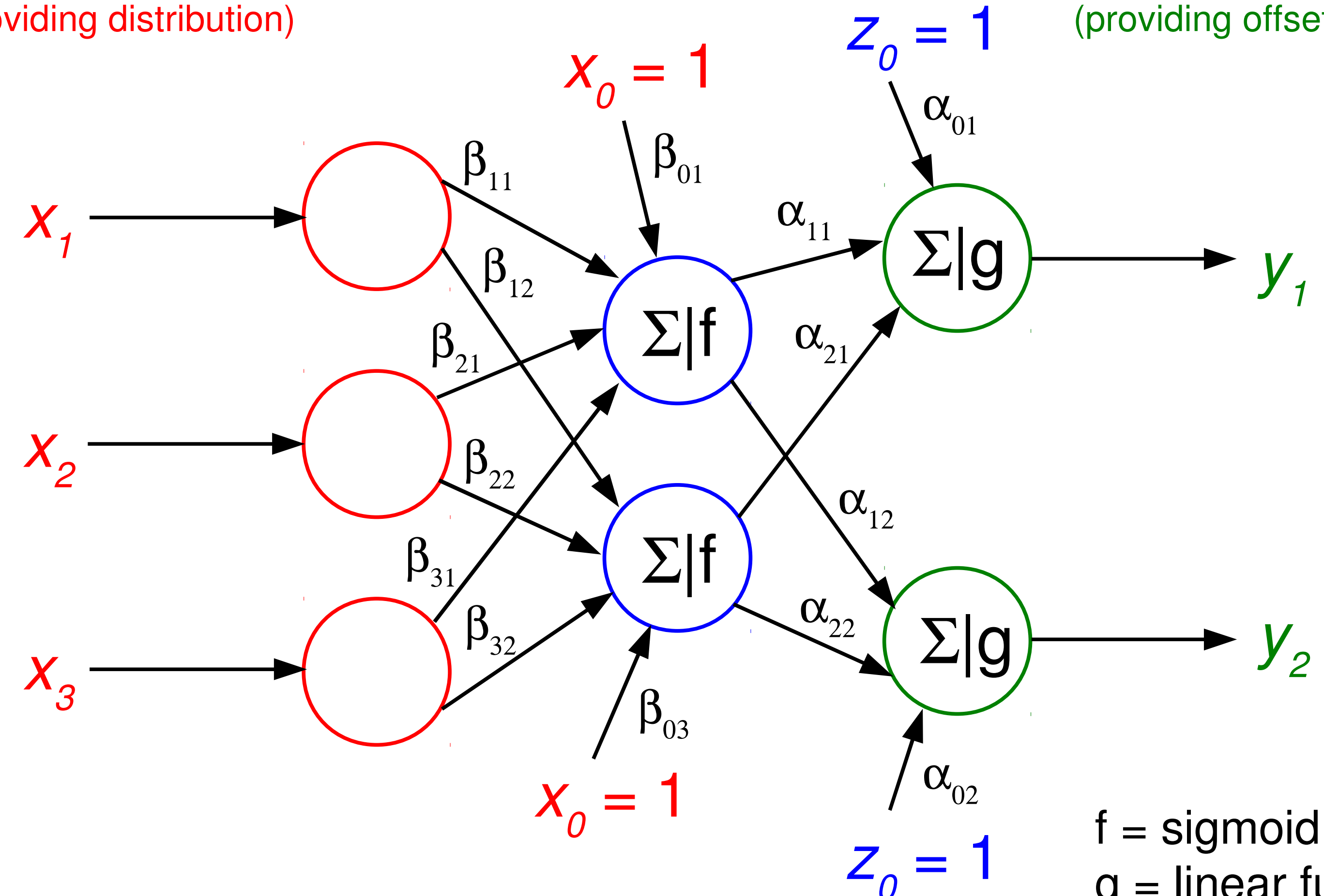
slide adapted from Jan Therhaag, TMVA workshop, CERN 2011

Neural networks

input layer
number of nodes =
dimension of input
(providing distribution)

hidden layer(s)
any number of nodes

output layer
number of nodes
= dimension of output
(providing offset and scaling)



f = sigmoid function
g = linear function

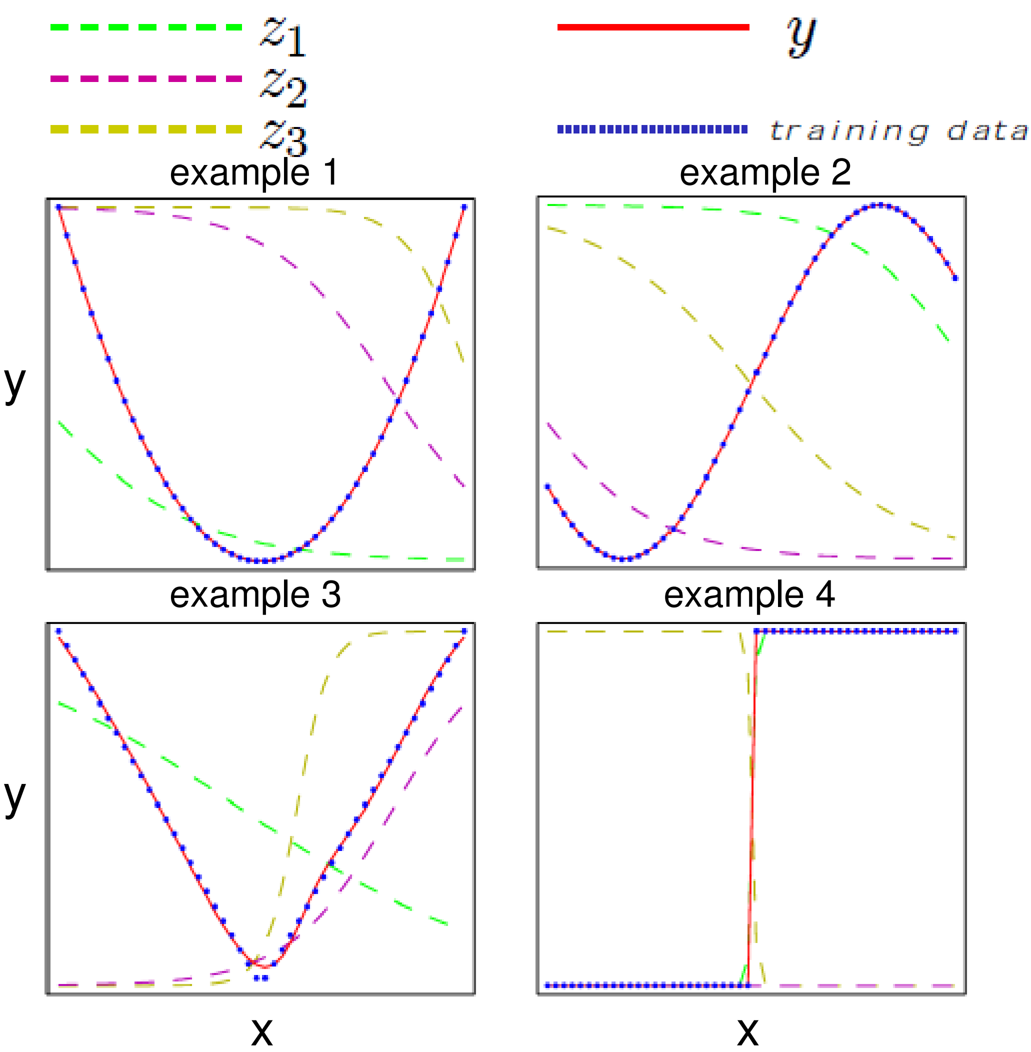
Universal approximators

Kolmogorov's universal approximation theorem

Neural network with one hidden layer of sufficient size can approximate any continuous function to arbitrary precision

Non-linear optimization problem with several local minima
→ use global optimization schemes (time consuming training)

Unknown required number of hidden nodes for a given problem
→ use as many nodes as computing power permits
→ avoid overfitting via regularization



slide adapted from Jan Therhaag, TMVA workshop, CERN 2011

Overtraining

Decision boundary can become overly complex for many hidden nodes
→ bad generalization
→ overconfident prediction

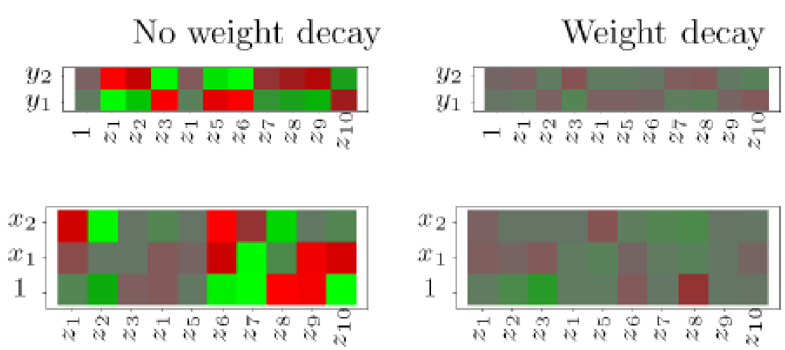
Weight magnitude ~ smoothness

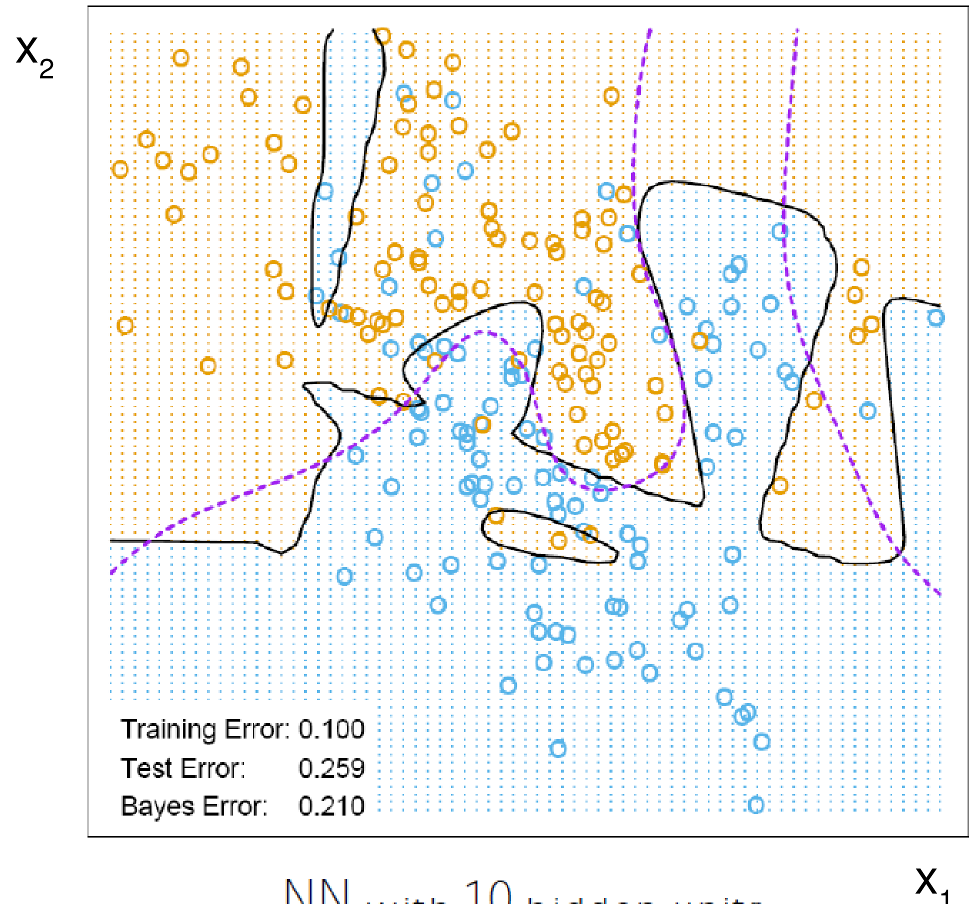
↓

Regularization
Penalize large weights during global optimization

$$\tilde{E}(w) = E(w) + \lambda w^T w$$

hyper parameter

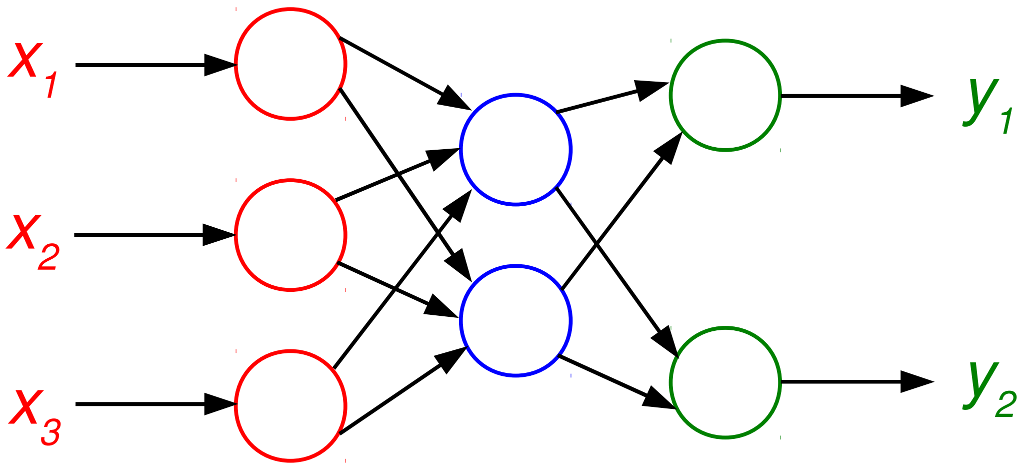




NN with 10 hidden units

slide adapted from Jan Therhaag, TMVA workshop, CERN 2011

Neural networks – summary



- Neural networks are universal approximators
  - Can be used for regression and classification
- ANNs are difficult and time-consuming to train, but provide fast prediction
- ANNs become very powerful in Bayesian context
  - Use MCMC simulation to obtain global maximum
  - Maximize *Bayesian evidence* to obtain optimal hyperparameters (no overfitting)
- NN are nothing special, any flexible mapping $R^i \rightarrow R^o$ would do the same