Statistische Methoden der Datenanalyse

Wintersemester 2012/2013

Albert-Ludwigs-Universität Freiburg

Prof. Markus Schumacher, Dr. Stan Lai

Physikalisches Institut Westbau 2 OG

E-Mail: Markus.Schumacher@physik.uni-freiburg.de

stan.lai@cern.ch

Kapitel 8

Ereignisklassifizierung (Fortsetzung)

Lösungsansätze zum “Fluch der Dimensionalität”

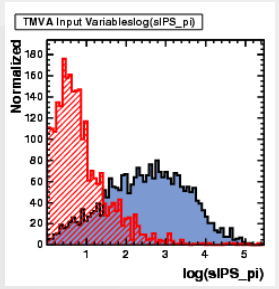
a) schätze n-dimenionale WDFs für Signal und Untergrund
- Nächste Nachbarn-Methode
- Kernel-WDF-Schätzern ….

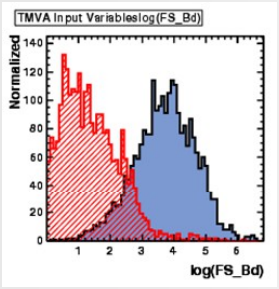
immer noch ein wenig FdD, da jedes zu klassifizierende Ereignis mit allen Trainingsereignissen verglichen werden muss

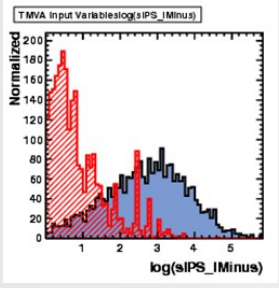
b) konstruierte Teststatistik als Abbildung von $R^n \rightarrow R$
= analytische Funktion in Observablenwerten,
die ähnlich optimal ist wie ***t(x)*** aus Neyman-Pearson-Lemma

- Projektive Likelihood
- Fishers lineare Teststatistik
- Künstliche Neuronale Netzwerke (Artificial Neural Network ANN)
- Verstärkte Entscheidungsbäume (Boosted Decision Tree BDT)

Formulierung des Problems





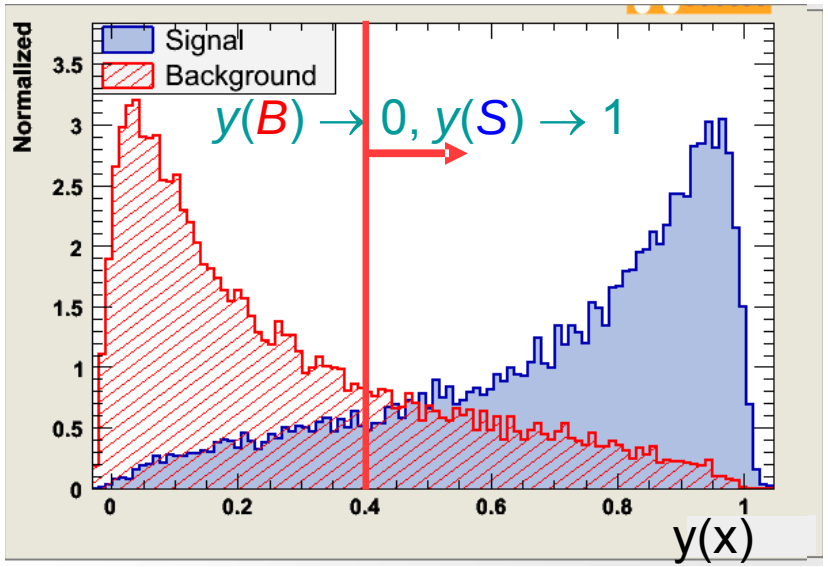


für jedes Ereignis, ob Signal oder Untergrund sind D Observablen gemessen

suche Abbildung y aus dem D-dimensionalen Observablenraum auf eindimensionale Teststatistik

$\mathbb{R}^D$

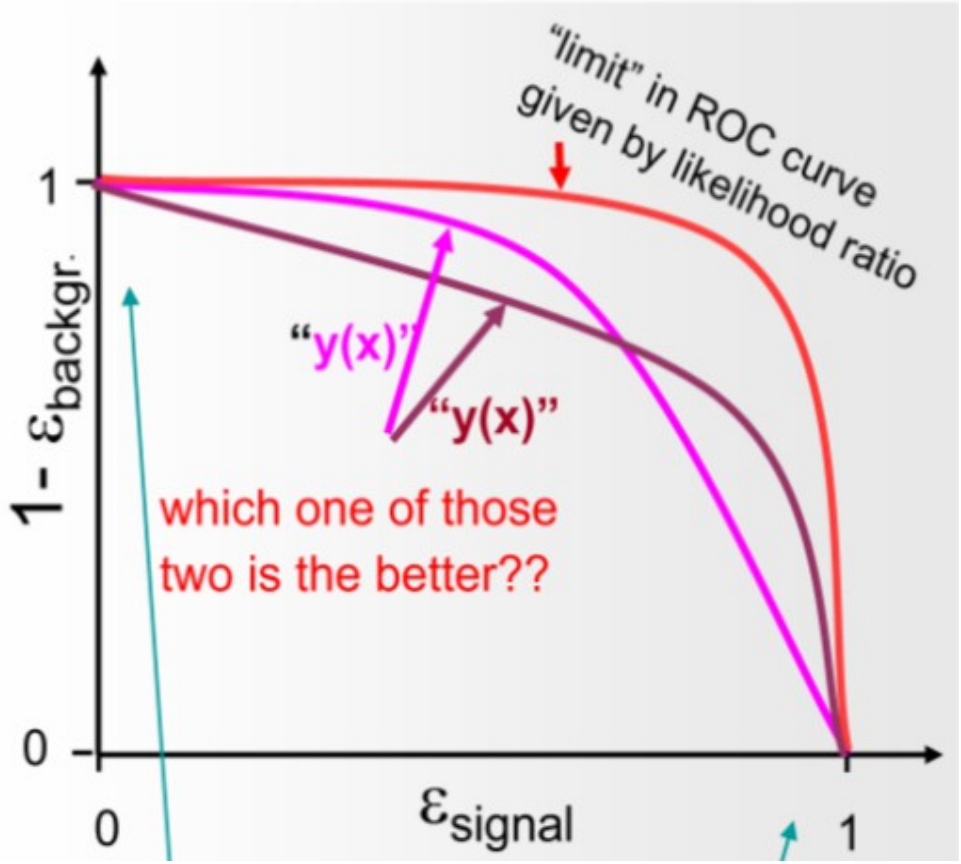
$y(x): R^n \rightarrow R$:



Ereignise werden dann selektiert durch einen Schnitt auf die Größe von y

$y > y_{Schnitt}$

Optimale Trennkraft für Näherungen an NPL



für konstruierte Teststatistiken ist Annäherung an NPL-Trennkraft nicht für alle Werte der Entscheidungsgrenze gleich

Wahl des Arbeitspunktes kann Auswahl der Teststatistik beinflussen.

gr. Untergrund-unterdrückung
kl. Signaleffizienz

kl. Untergrund-unterdrückung
gr. Signaleffizienz

Projektive Likeihood

Im Falle unabhängiger Observablen faktorisiert gemeinsame WDF im n-dim. Observablenraum in Produkt von n eindimensionalen WDF

$$p(\mathbf{x}) \cong \prod_{i=0}^{D} p_i(\mathbf{x})$$

PDFs

discriminating variables

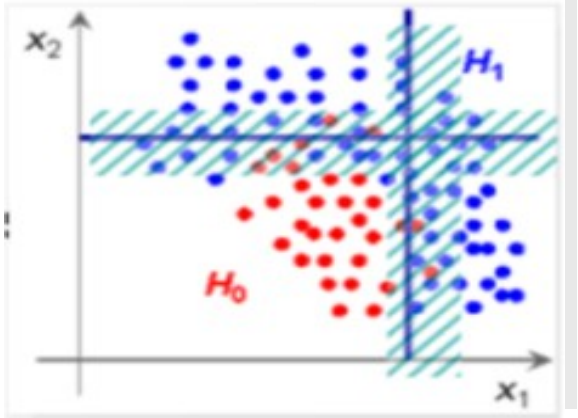
Likelihood ratio for event *event*

$$y(x_{\mathrm{PDE},k_{\mathrm{event}}}) = \frac{\prod_{i\in\{\mathrm{variables}\}} p_i^{\mathrm{signal}}(x_{i,k_{\mathrm{event}}})}{\sum_{C\in\{\mathrm{classes}\}} \left(\prod_{i\in\{\mathrm{variables}\}} p_i^{C}(x_{i,k_{\mathrm{event}}}) \right)}$$

Classes: signal, background types

sehr einfache Konstruktion: benötigt nur n eindimenionale Histogramme

an Stelle von “harten Schnitten” auf Observablen wird eine gewisse “Unschärfe” in Entscheidungs-grenze eingebaut

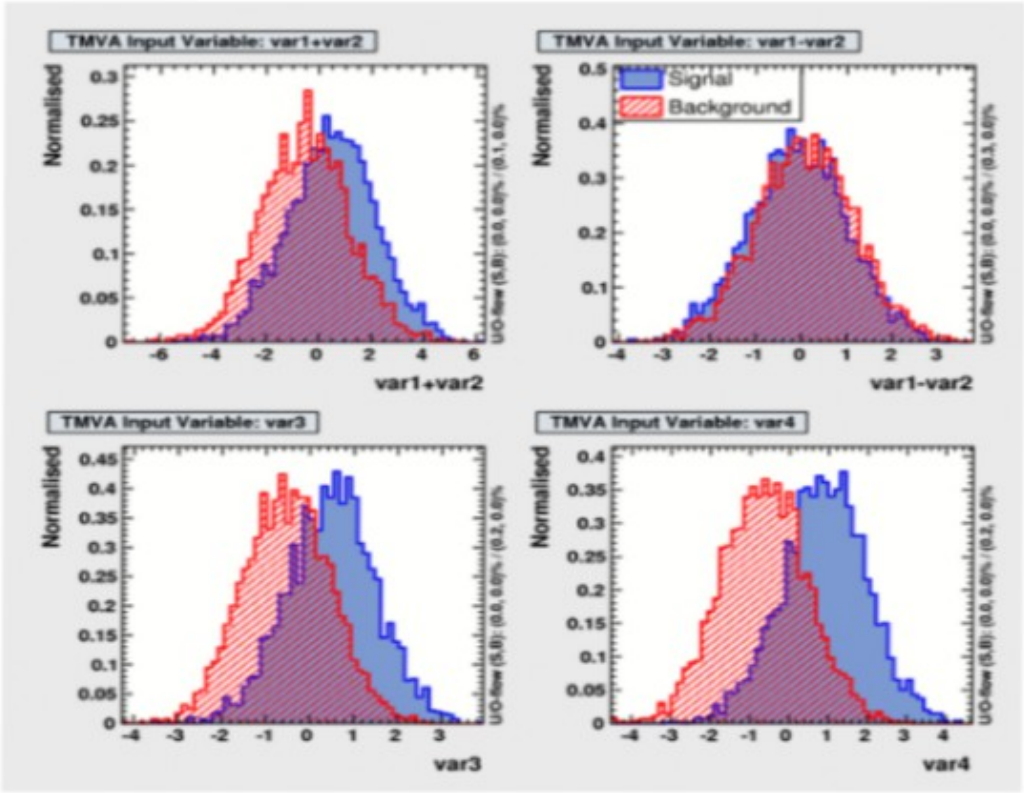


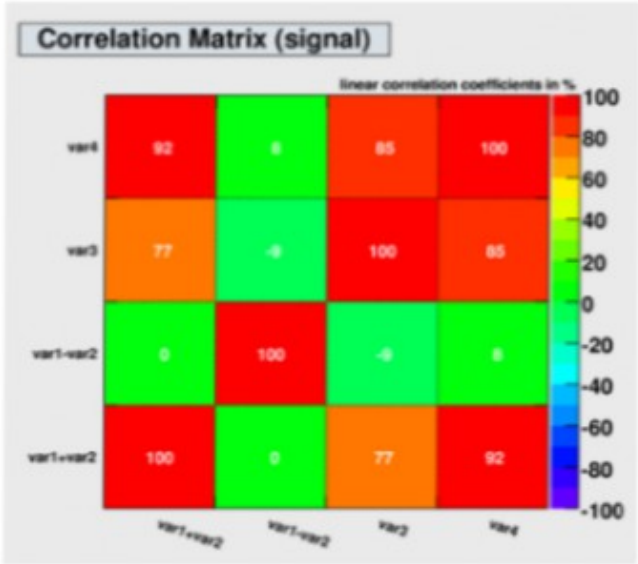
für unkorrelierte Observablen optimal, d.h.

äquivalent zu NPL-Teststatistik

Projektive Likelihood im Falle von Korrelationen

Korrelation: $C_{ij}=\mathrm{cov}[x_i, x_j]=E[x_i x_j]-E[x_i]E[x_j]\neq 0 \quad (i\neq j)$

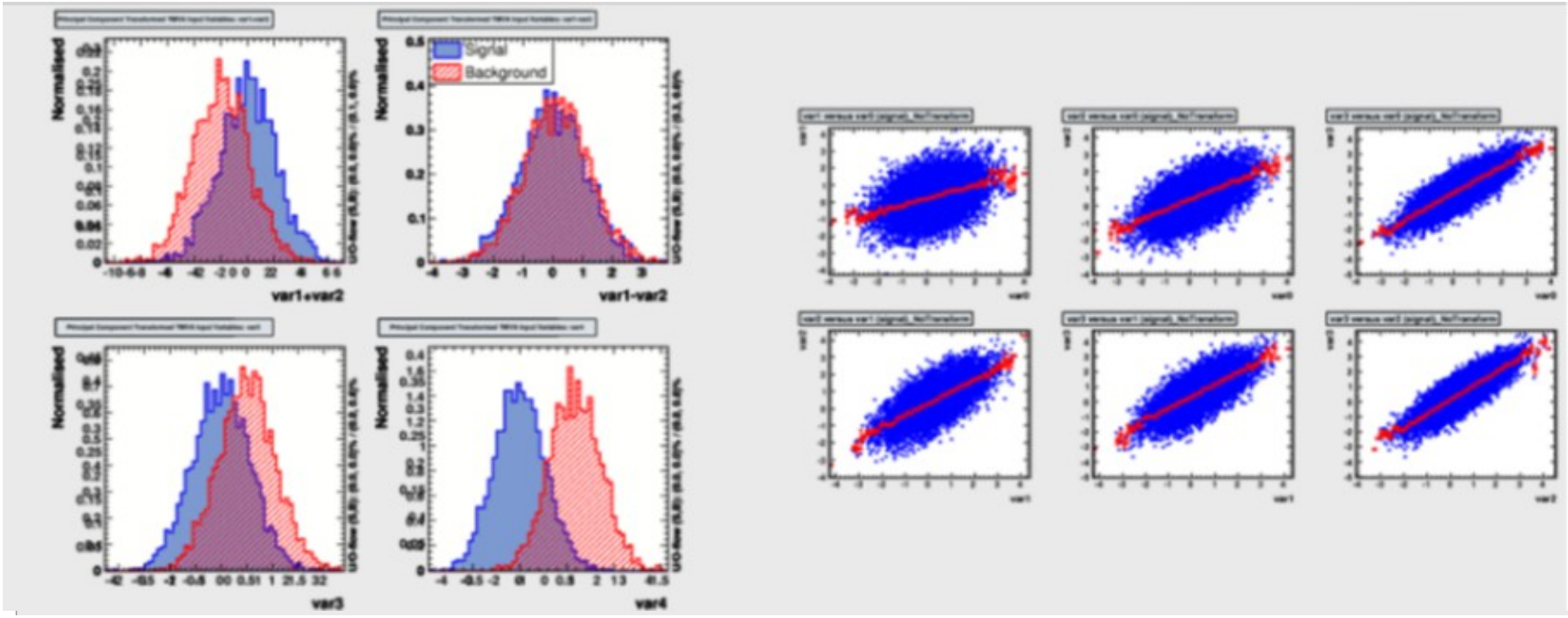




Dekorrelation:
wähle nichtlineare Transformation der Eingangsvariablen=Observablen, so dass für transformierte Observable die Kovarianzmatrix diagonal ist

Projektive Likelihood im Falle von Korrelationen

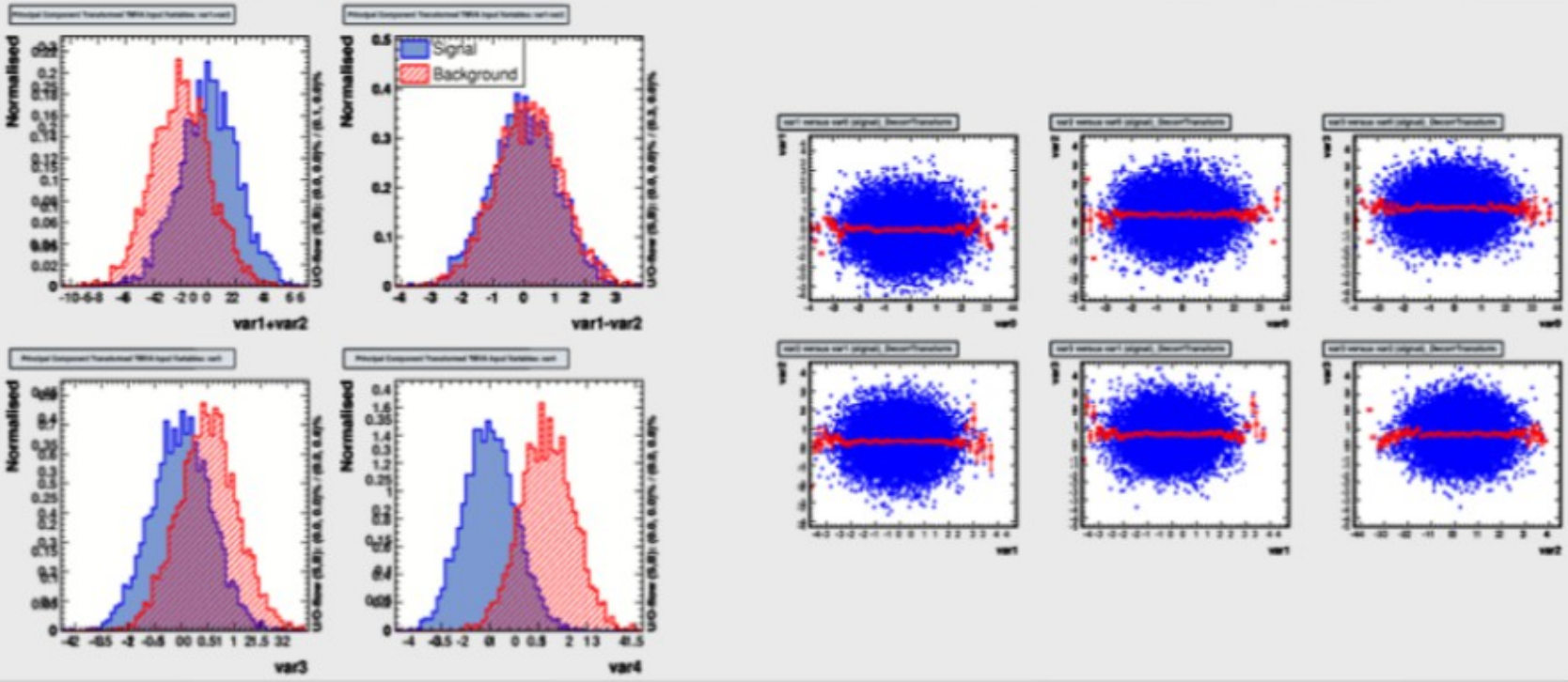
- Determine *square-root* C' of correlation matrix C, *i.e.*, $C = C'C'$
  - compute C' by diagonalising C: $D = S^T C S \Rightarrow C' = S\sqrt{D}S^T$
  - transformation from original (x) in de-correlated variable space (x′) by: $x' = C'^{-1}x$



Achtung: funktioniert optimal nur für lineare/gausssche Korrelationen

Projektive Likelihood im Falle von Korrelationen

- Determine *square-root* C' of correlation matrix C, *i.e.*, $C = C'C'$
  - compute C' by diagonalising C: $D = S^T C S \Rightarrow C' = S\sqrt{D}S^T$
  - transformation from original (x) in de-correlated variable space (x′) by: $x' = C'^{-1}x$



Achtung: funktioniert optimal nur für lineare/gausssche Korrelationen

Anwendung der Dekorrelation

Bei Projektiver Likelihood:
dekorreliere Observablen separat unter Signal- und Untergrundhypothesen

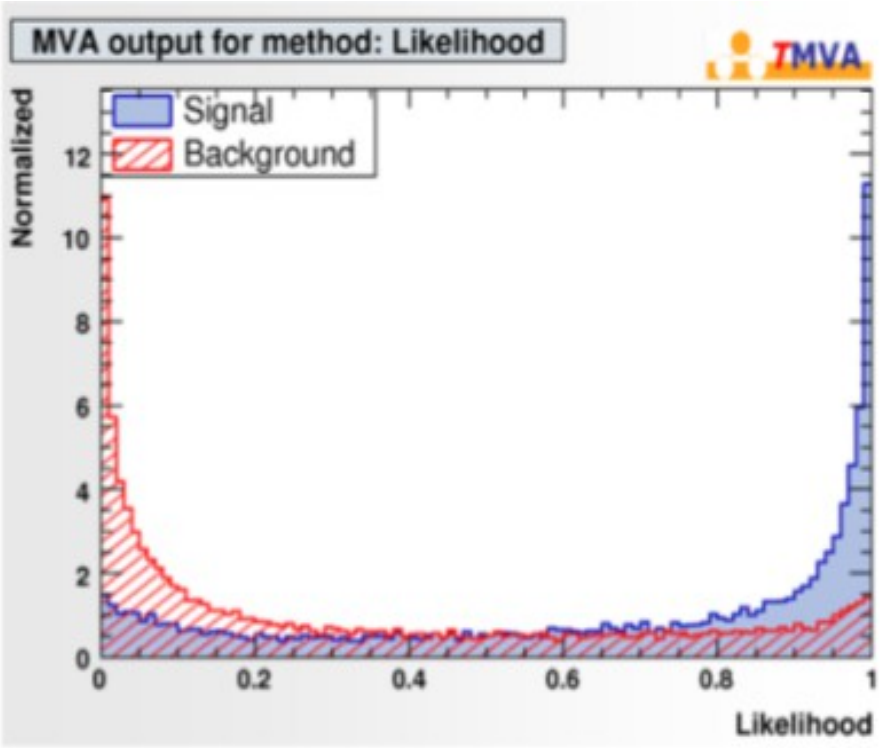
$$y_L\left(i_{\text{event}}\right) = \frac{\prod_{k\in\{\text{variables}\}} p_k^S\left(x_k(i_{\text{event}})\right)}{\prod_{k\in\{\text{variables}\}} p_k^S\left(x_k(i_{\text{event}})\right) + \prod_{k\in\{\text{variables}\}} p_k^B\left(x_k(i_{\text{event}})\right)}$$

$$y_L^{\text{trans}}\left(i_{\text{event}}\right) = \frac{\prod_{k\in\{\text{variables}\}} p_k^S\left(\hat{T}^S x_k(i_{\text{event}})\right)}{\prod_{k\in\{\text{variables}\}} p_k^S\left(\hat{T}^S x_k(i_{\text{event}})\right) + \prod_{k\in\{\text{variables}\}} p_k^B\left(\hat{T}^B x_k(i_{\text{event}})\right)}$$

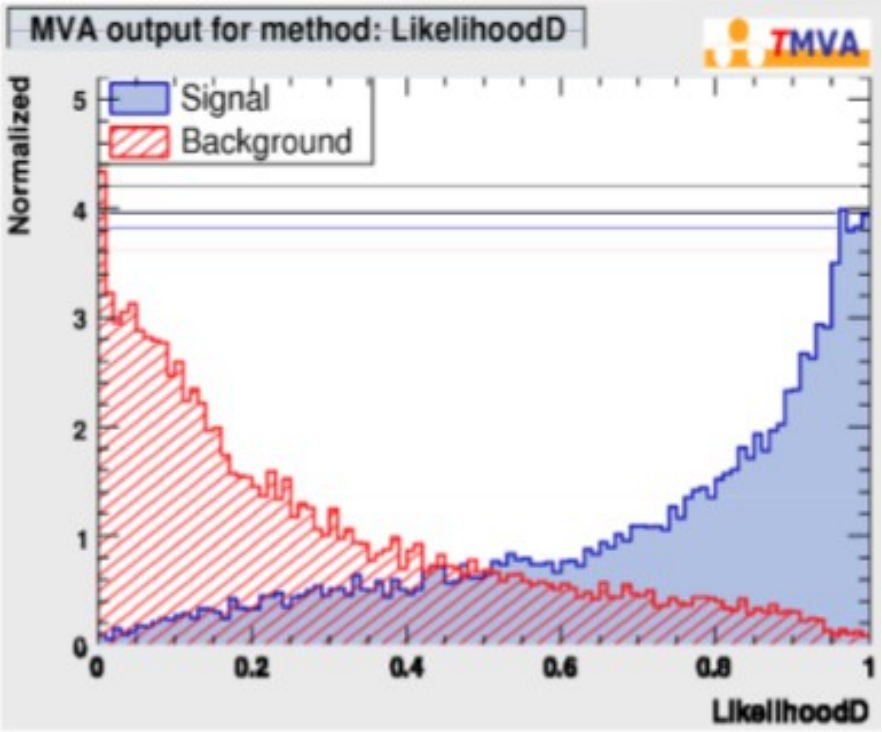
Projektive Likelihood: Effekt der Dekorrelation

Beispiel:
linear korrelierte Gaussverteilungen → Dekorrelation funktioniert zu 100%
1-dim. Proj. Likelihood auf dekorrelierten Größen liefert optimale Trennkraft

korrelierte Größen



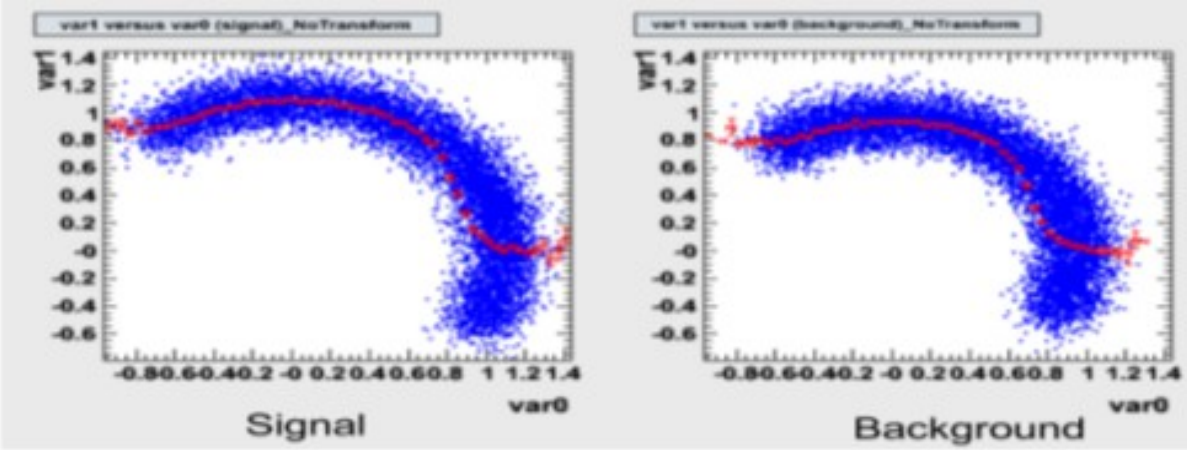
nach Dekorrelation



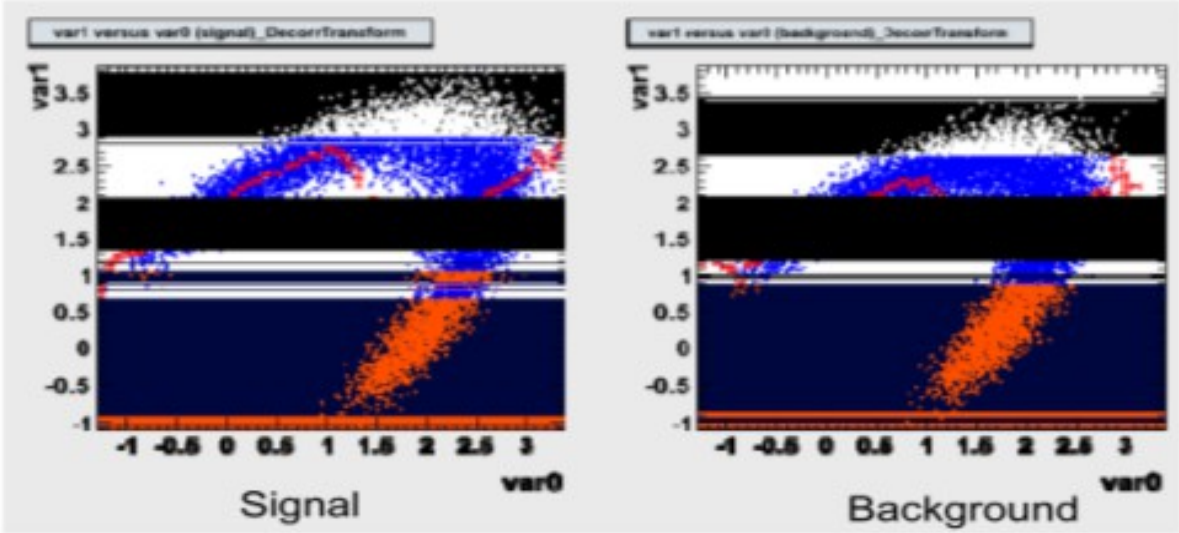
weniger “Peaks” bei 0 und 1. Trennkraft besser, eventuell optimal.

Dekorrelation bei nicht linearen Korrelationen

Originalkorrelation



nach Dekorrelation



Achtung: bei nicht gaussverteilten Größen und nicht linear korrelierten Größen kann Dekorrelation die Trennkraft auch verschlechtern!

Transformation auf Gaussfunktionen

Ziel: transformiere Eingangsgrößen, sodass nach Transformation verteilt nach Gauss-WDF

1. Schritt: transformiere auf gleichförmige Verteilung

$$x_k^{\text{flat}}(i_{\text{event}}) = \int_{-\infty}^{x_k(i_{\text{event}})} p_k(x'_k)\,dx'_k \quad , \forall k \in \{\text{variables}\}$$

Rarity transform of variable k | Measured value | PDF of variable k

2. Schritt: transformiere durch inverse Fehlerfunktion auf Gauss-WDF

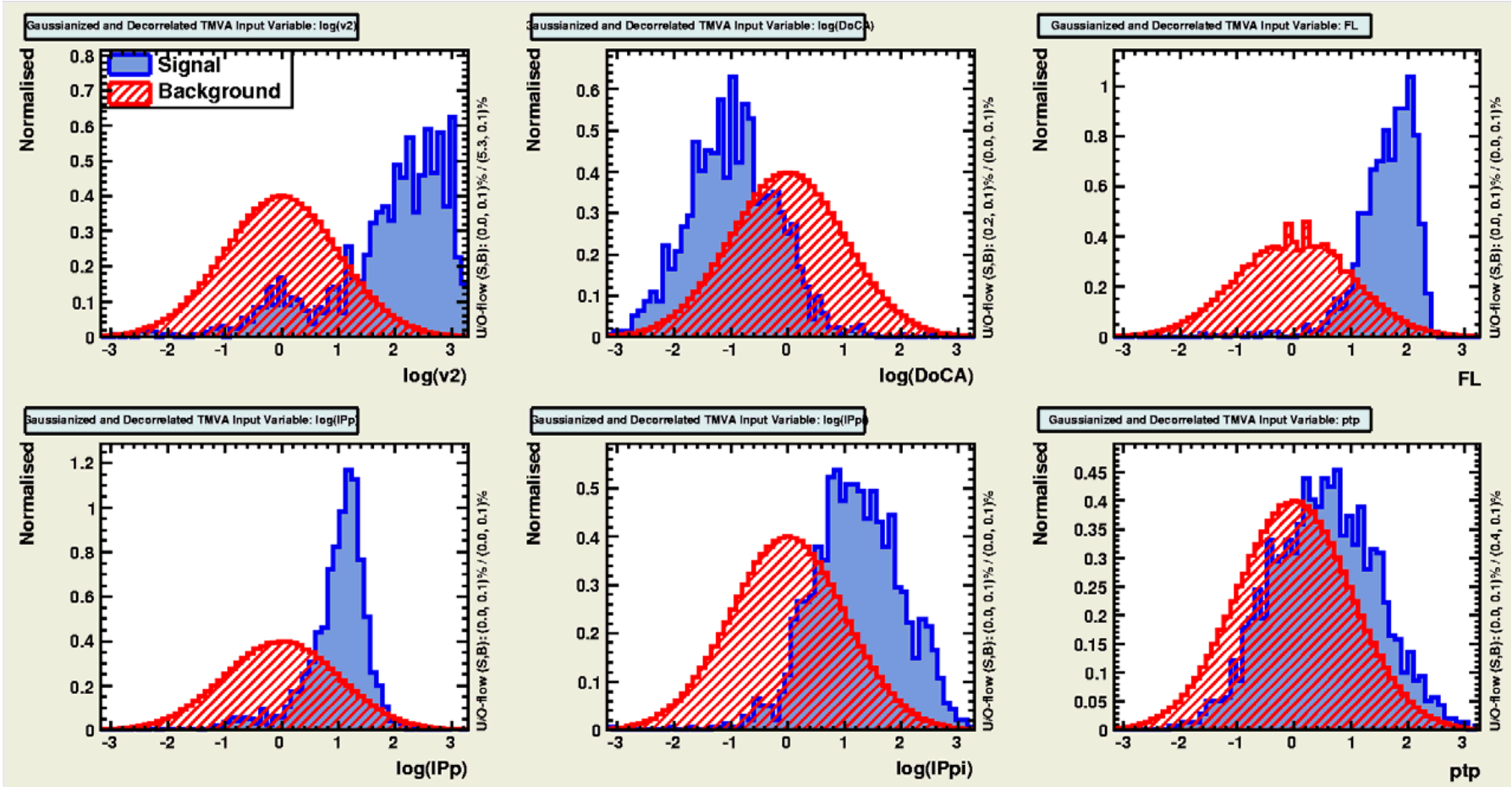
$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

$$x_k^{\text{Gauss}}(i_{\text{event}}) = \sqrt{2} \cdot \text{erf}^{-1}\left(2x_k^{\text{flat}}(i_{\text{event}}) - 1\right) \quad , \forall k \in \{\text{variables}\}$$

3. Schritt: dekorreliere Gauss-transformierte Observablen und probiere es nochmal

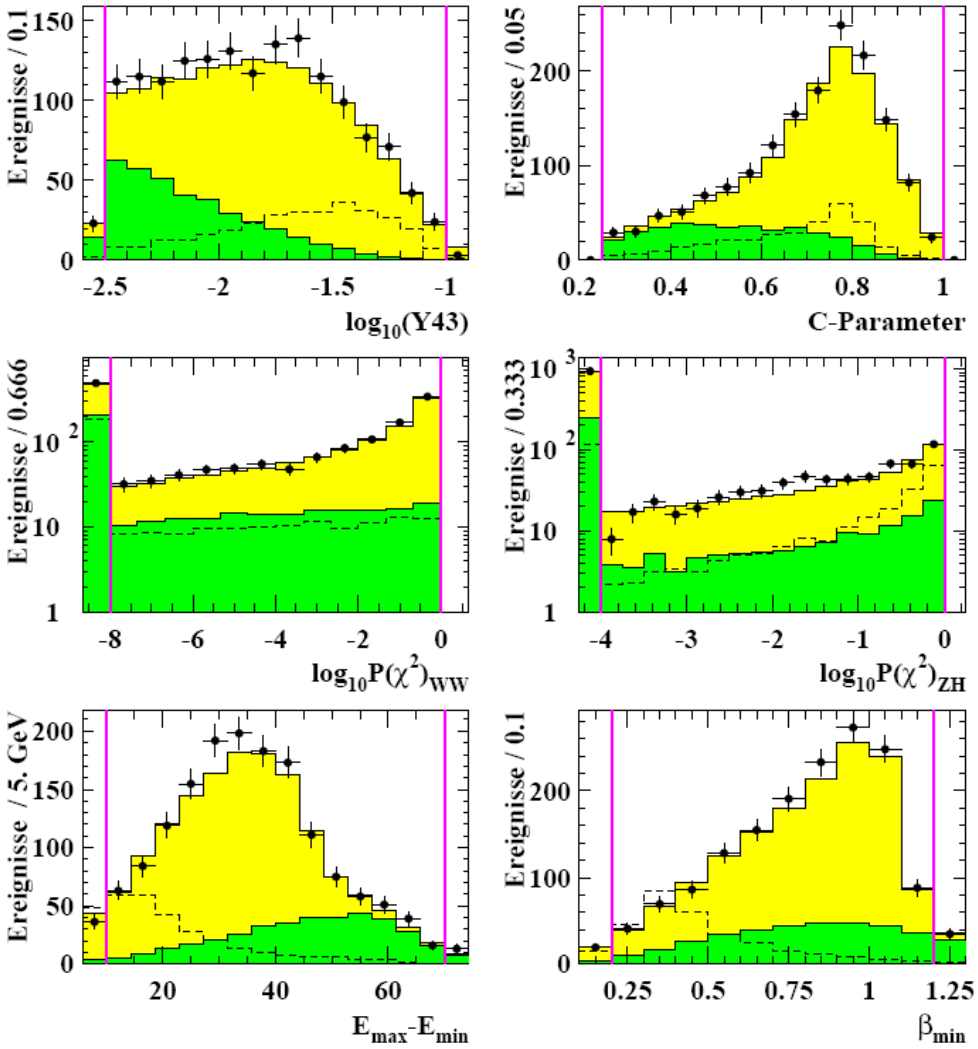
Transformation auf Gaussfunktionen

Es kann nur Signal- oder Untergrund WDF in Gauss-Verteilung überführt werden

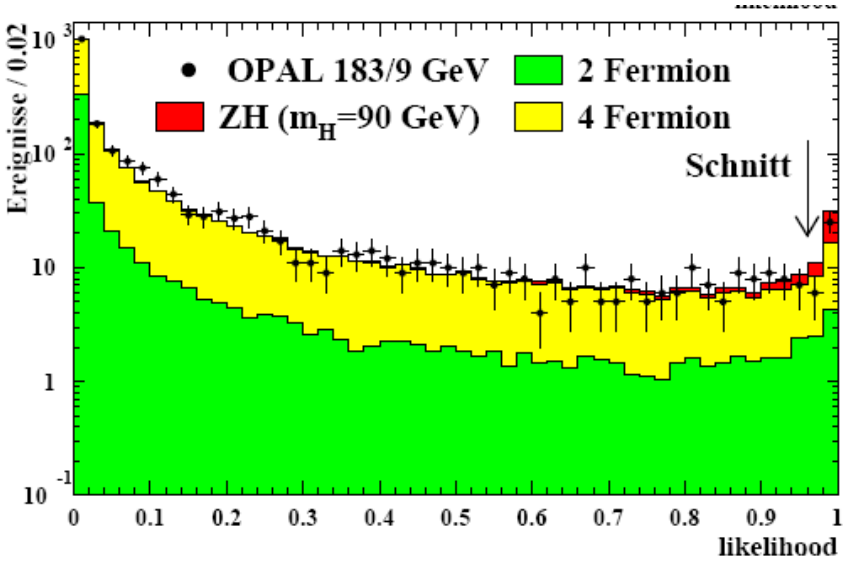


Likelihoodselektion für Higgs-Suche bei LEP

Eingangsvariablen



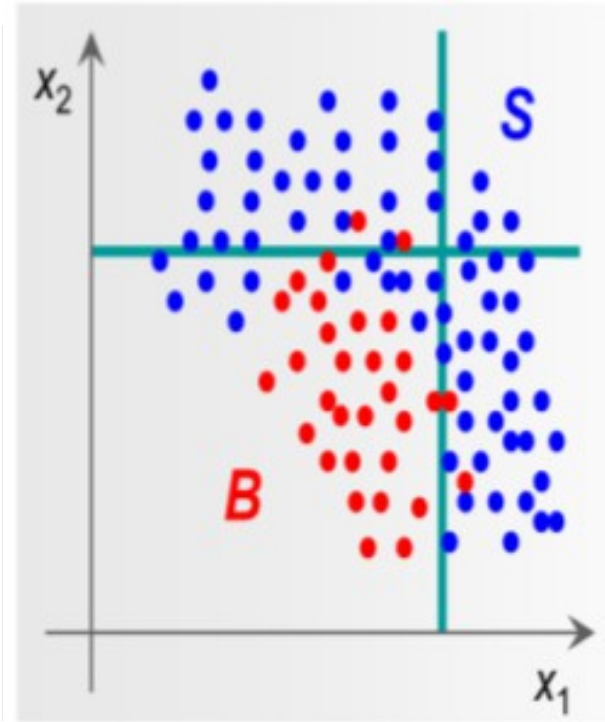
Likelihood-Output



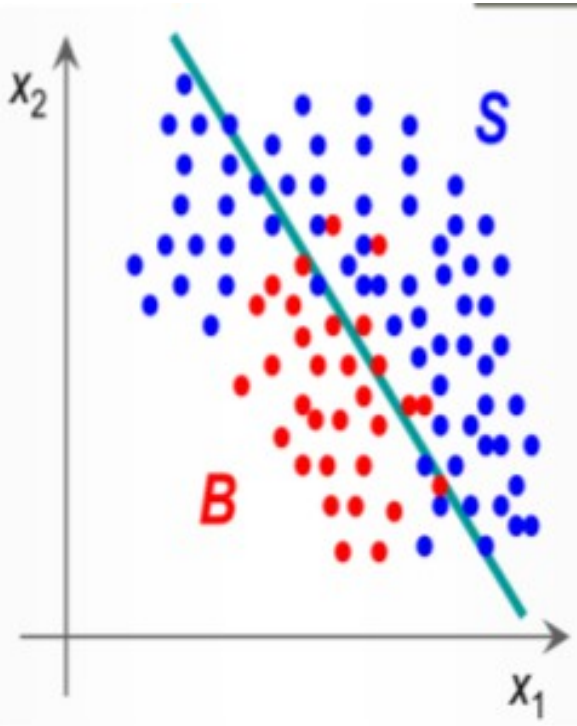
Andere Möglichkeiten Ereignisse zu selektieren?

Verwende eine andere Art der Entscheidungsgrenze

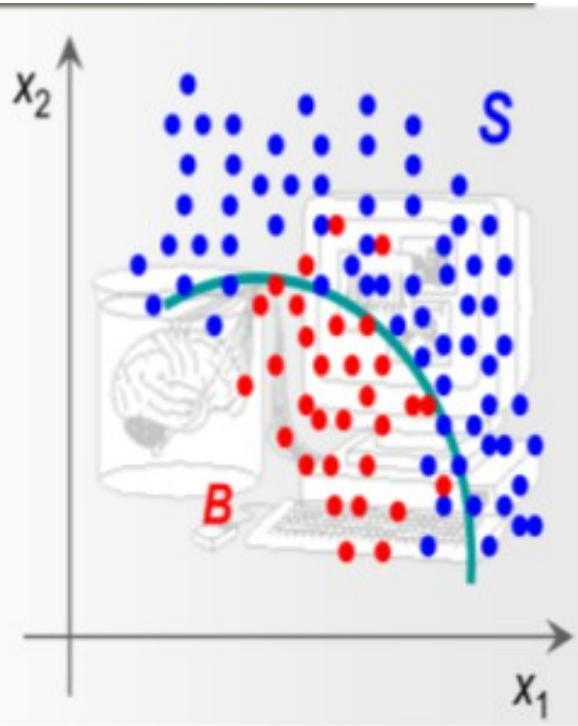
Schnitte



linear



nichtlinear

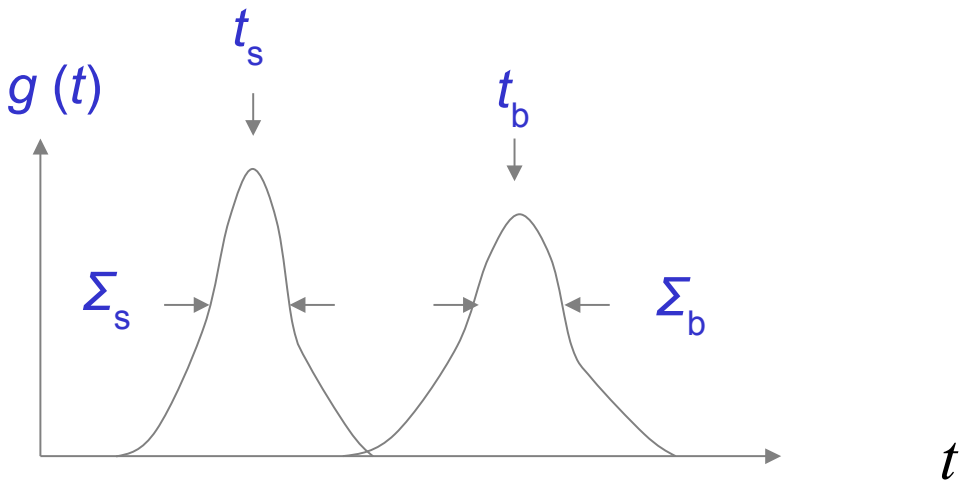


Lineare Teststatistik: Fisherdiskriminante

Ansatz: $$t(\vec{x}) = \sum_{i=1}^{n} a_i x_i$$

Wähle die Parameter a_1, ..., a_n so, dass die WDFs unter den beiden Hypothesen "Signal" und "Untergrund" maximale Trennkraft besitzen: $g(t|\mathrm{s}),\ g(t|\mathrm{b})$

Wir wollen:
- großer Abstand zwischen Erwartungswerten
- kleine Breite/Varianz der WDFs



→ Fishers Ansatz: maximiere: $$J(\vec{a}) = \frac{(\tau_\mathrm{s} - \tau_\mathrm{b})^2}{\Sigma_\mathrm{s}^2 + \Sigma_\mathrm{b}^2}$$

Bestimmung der Koeffizienten a_i

Wir haben

$$(\mu_k)_i = \int x_i f(\vec{x}|H_k)\, d\vec{x}$$

$$(V_k)_{ij} = \int (x-\mu_k)_i (x-\mu_k)_j f(\vec{x}|H_k)\, d\vec{x}$$

mit

$$k = 0,1 \quad \text{(hypothesis)}$$

$$i,j = 1,\ldots,n \quad \text{(component of } \vec{x}\text{)}.$$

ausgedrückt in Erwartungswert und Kovariant der Testatistik $t(\vec{x})$ lauten diese

$$\tau_k = \int t(\vec{x}) f(\vec{x}|H_k) d\vec{x} = \vec{a}^T \vec{\mu}_k \,,$$

$$\Sigma_k^2 = \int (t(\vec{x}) - \tau_k)^2 f(\vec{x}|H_k)\, d\vec{x} = \vec{a}^T V_k \vec{a} \,.$$

Bestimmung der Koeffizienten a_i (2)

Der Zähler von $J(\boldsymbol{a})$ ist dann

$$(\tau_0 - \tau_1)^2 = \sum_{i,j=1}^{n} a_i a_j (\mu_0 - \mu_1)_i (\mu_0 - \mu_1)_j$$

'zwischen' Klassen

$$= \sum_{i,j=1}^{n} a_i a_j B_{ij} = \vec{a}^T B \vec{a} \,,$$

'innerhalb' Klassen

und der Nenner ist

$$\Sigma_0^2 + \Sigma_1^2 = \sum_{i,j=1}^{n} a_i a_j (V_0 + V_1)_{ij} = \vec{a}^T W \vec{a}$$

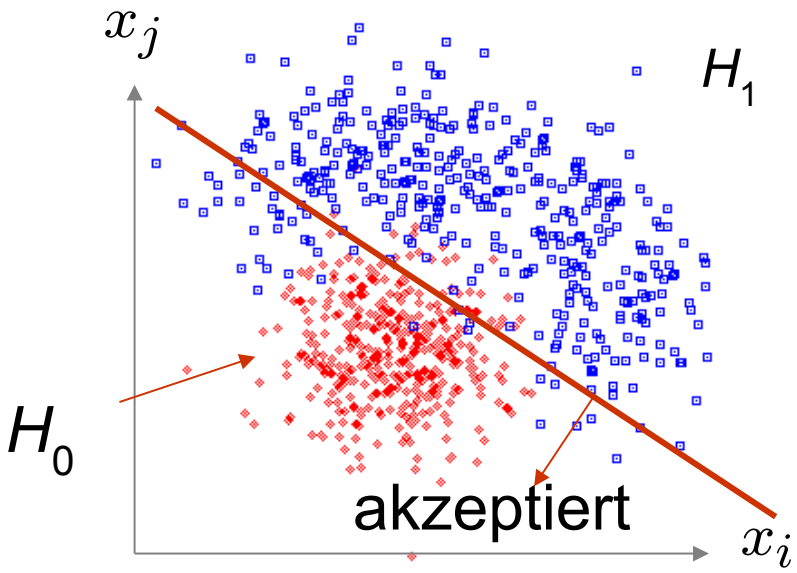
→ maximiere

$$J(\vec{a}) = \frac{\vec{a}^T B \vec{a}}{\vec{a}^T W \vec{a}} = \frac{\text{separation between classes}}{\text{separation within classes}}$$

Fisher-Diskriminante

Nullsetzen $\frac{\partial J}{\partial a_i} = 0$ ergibt Fishers lineare Discriminantenfunktion:

$$t(\vec{x}) = \vec{a}^T \vec{x}, \qquad \text{with } \vec{a} \propto W^{-1}(\vec{\mu}_0 - \vec{\mu}_1)$$



Entspricht linearer Entscheidungsgrenze
= (n-1) dimensionaler Hyperebene
im n-dimensionalen Observablenraum.

Vector ***a*** ist Normalenvektor auf Ebene

Fisher-Diskriminante: Bemerkung zu KQ

Wir erhalten die gleiche Trennkraft zwischen den Hypothesen wenn wir die Koeffizienten a_i mit gemeinsamen Skalierungsfaktor multiplizieren und einen beliebigen Offset a_0 addieren:

$$t(\vec{x}) = a_0 + \sum_{i=1}^{n} a_i x_i$$

Daher können wir die Mittelwerte t_0 and t_1 unter der Null- und Alternativhypothese vorgeben, z.B. 0 und 1.

Maximierung von $J(\vec{a}) = (\tau_0 - \tau_1)^2/(\Sigma_0^2 + \Sigma_1^2)$

ist dann äquivalent zur Minimierung von

$$\Sigma_0^2 + \Sigma_1^2 = E_0[(t-\tau_0)^2] + E_1[(t-\tau_1)^2]$$

Maximierung $J(\boldsymbol{a})$
→ 'Kleinste Quadrate'

In der Praxis werden Erwartungswerte durch die Stichprobenmittelwerte aus Traningsdatensätzen mit MC-Simulation ersetzt/geschätzt.

Fisher-Diskriminante u. Gausssche Daten

Annahme $f(\vec{x}|H_k)$ ist Multivariate n-dimensionale-Gauss-WDF mit Erwartungswerten

$$E_0[\vec{x}] = \vec{\mu}_0 \text{ for } H_0 , \qquad E_1[\vec{x}] = \vec{\mu}_1 \text{ for } H_1 ,$$

und identischen Kovarianz-Matrizen *V* für beide Hypothesen. Dann können wir die Fisherdiskriminate (mit Offset) schreiben als.

$$t(\vec{x}) = a_0 + (\vec{\mu}_0 - \vec{\mu}_1)^T V^{-1} \vec{x} .$$

Das Neyman-Pearson-Likelihoodverhältnis wird dann zu:

$$\begin{aligned} r &= \frac{f(\vec{x}|H_0)}{f(\vec{x}|H_1)} \\ &= \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu})_0^T V^{-1}(\vec{x}-\vec{\mu}_0) + \frac{1}{2}(\vec{x}-\vec{\mu})_1^T V^{-1}(\vec{x}-\vec{\mu}_1)\right] \\ &\propto e^t \end{aligned}$$

Fisher-Diskriminante u. Gausssche Daten (2)

Dies bedeutet $t \propto \ln r + \text{ const.}$

Fisherdiskriminante ist monotone Funktion der Neyman-Pearson-Teststatistik

Die Fisher-Diskriminante ist äquivalent zur Verwendung des Likelihoodverhältnisses und liefert somit den optimalen Test, maximiert Reinheit für gegebene Nachweiswahrscheinlichkeit.

Für nicht-gaussverteilte Daten ist dies nicht mehr gültig.
Aber lineare Teststatistik ist möglicherweise sehr gute praktische Lösung.

Oft transformiert man die Daten, so dass die WDFs besser durch Gauss-WDF angenähert werden und wendet dann Fisher-Methode an.

Fisher-Diskriminante u. Gausssche Daten (3)

Multivariate Gaussverteilte Daten mit gleicher Kovarianzmatrix ergeben ebenfalls einen einfachen Ausdruck für Posterior-Wkt., z.B.

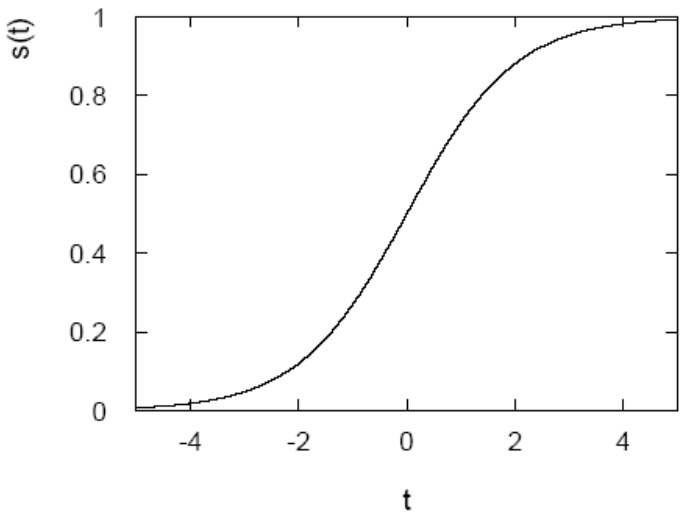
$$P(H_0|\vec{x}) = \frac{f(\vec{x}|H_0)\pi_0}{f(\vec{x}|H_0)\pi_0 + f(\vec{x}|H_1)\pi_1} = \frac{1}{1+\frac{\pi_1}{\pi_0 r}} \,.$$

Für spezifische Wahl des "Offset" a_0 kann man dies schreiben als:

$$P(H_0|\vec{x}) = \frac{1}{1+e^{-t}} \equiv s(t) \,,$$

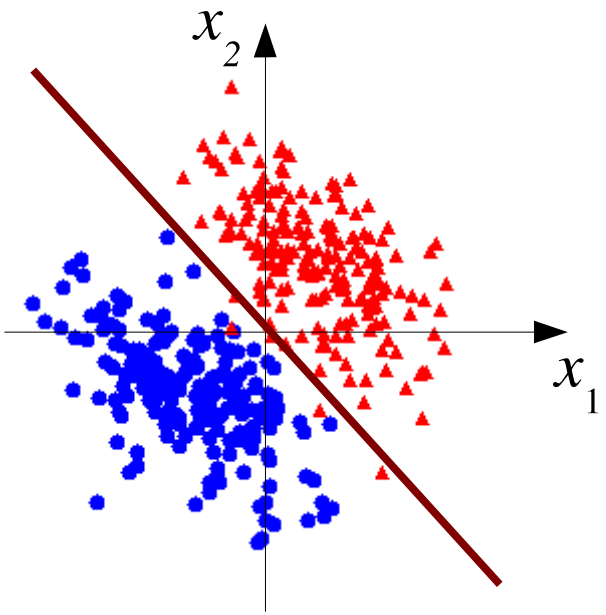
die ist die Logistische Sigmoidfunktion:

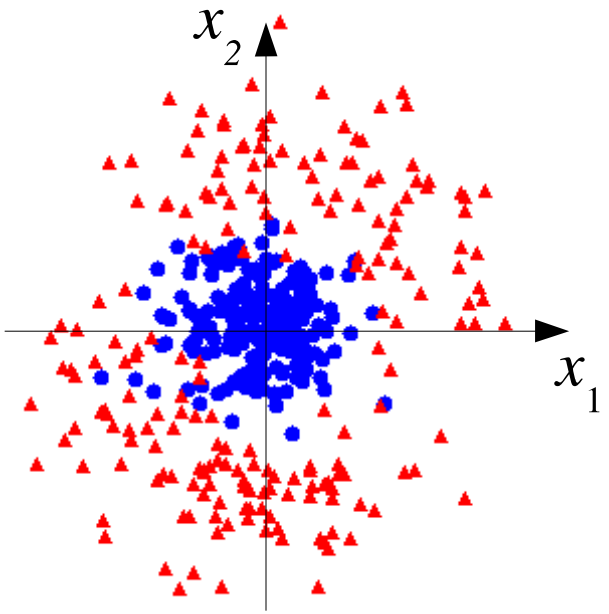
(taucht später bei ANN wieder auf)



Fisher-Diskriminante: Beispiele

Fisher: nur optimal
für Gaussverteilte Variablen
mit identischer Kovarianzmatrix





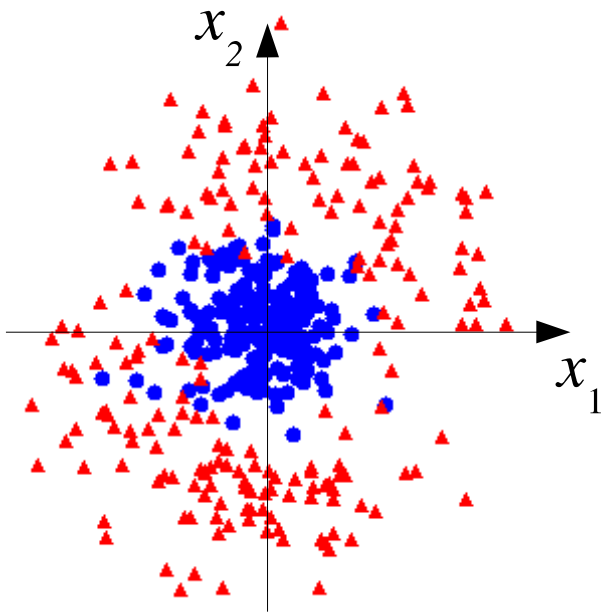
Für gewisse Fälle ist
lineare Entscheidungsgrenze
fast komplett nutzlos.

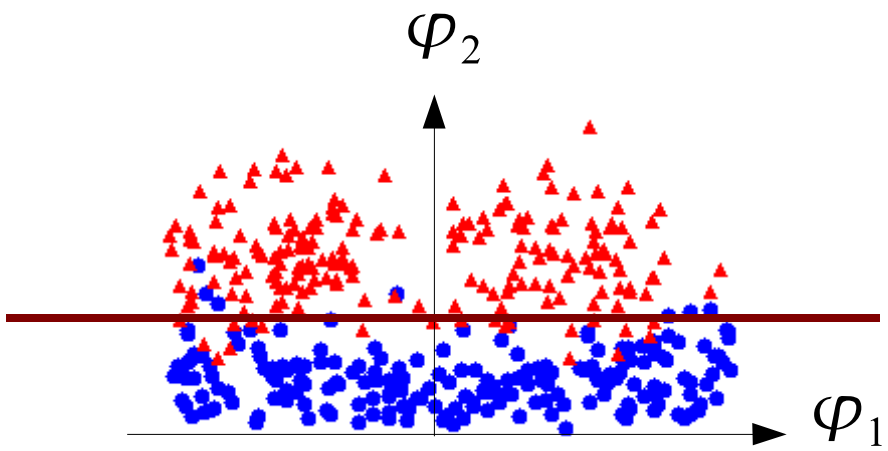
Nichtlineare Transformation der Observablen

Transformation: $x_1, \ldots, x_n \rightarrow \varphi_1(\vec{x}), \ldots, \varphi_m(\vec{x})$

Nach Transformation ist lineare Entscheidungsgrenze geeigneter.

Für unser Beispiel: $\varphi_1 = \tan^{-1}(x_2/x_1) \quad \varphi_2 = \sqrt{x_1^2 + x_2^2}$



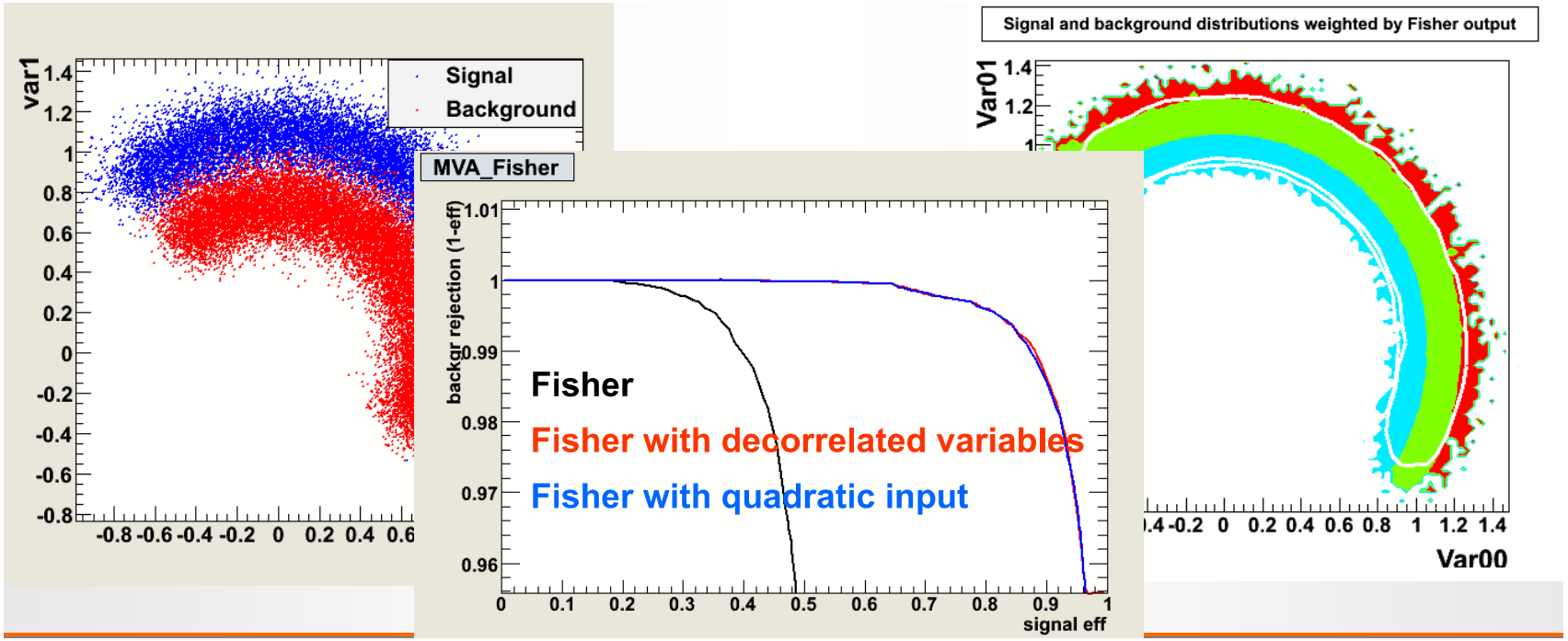


Fisherdiskriminate mit quadratischem Input

einfache Erweiterung von linearer zu quadratischer Entscheidungsgrenze

VAR0, VAR1 → Fisherkonstruktion → lin. Grenze in VAR0 und VAR1

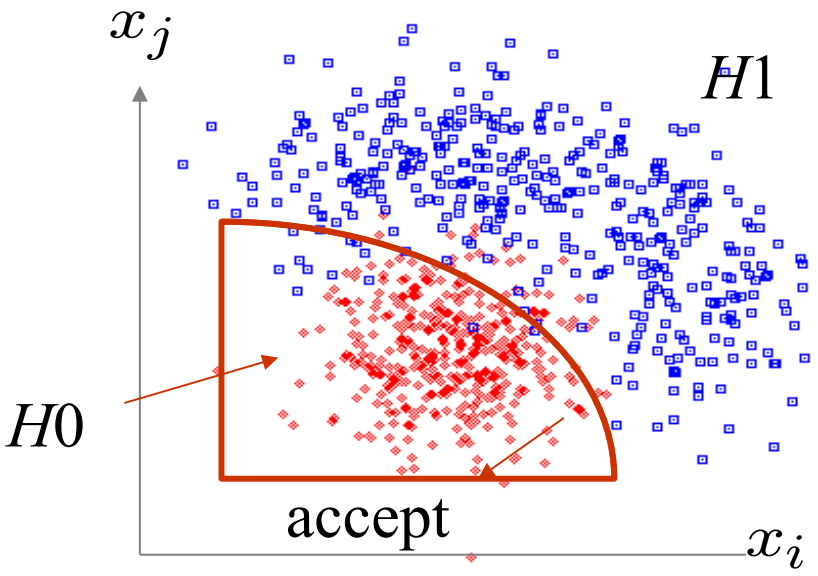
Quadratische Eingangsgrößen: VAR0*VAR0, VAR1*VAR0, VAR1*VAR1
→ Fisherkonstruktion → quadratische Grenze in VAR0 und VAR1



Nichtlin. Entscheidungsgrenze u. Training

Optimale Entscheidungsgrenze oft nicht Hyperebene

→ suche Teststatistik mit nichtlinearer Entscheidungsgrenze



Bisher: Teststatistik berechenbar bei 1 Durchlauf über simulierte Ereignisse
kein Trainingsprozess = Anpassung von Parametern notwendig

Jetzt: Funktionelle Form der Teststatsitik zwar vorgegegen aber
Bestimmung der optimalen Parameter in Basisfunktionen durch Trainingsprozess → viele Durchläufe über simulierte Ereignissätze = Trainingszyklen

Training und Fehlerfunktion

$Y(x_1,\ldots,x_n)$ von $R^n \rightarrow R$ sei zu bestimmende Teststatistik

Vorgaben von Zielwerten für korrekte Klassifizierung:
meist 1 für Signal 0 für Untergrund

Fehlerfunktion: misst Abweichung der Ausgangswerte der Teststatistik von Zielwerten

$$EPE(y(x)) = E(y - y(x))^2$$

Fehler = mittlerer quadratische Abweichung der Klassifikation von vorgebenen Zielwerten

wird im Trainingsprozess minimiert → Anpassung der Parameter

Training und Fehlerfunktion

wenn wir nichtlineare Entscheidungsgrenze haben wollen,
dann muss $y(x_1,\ldots,x_n)$ nichtlinear in Eingangsgrößen konstruiert werden

$$y(\vec{x}) = \sum_i^M \left(w_i \, h_i(\vec{x}) \right)$$

$h_i()$ nicht lineare Funktion der Eingangsgrößen

gewichten mit w_i

Betrachte $h_i(x)$ als Basisfunktionen

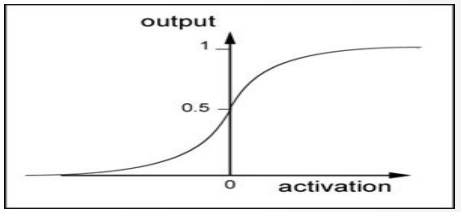
Wenn $h_i(x)$ ausreichend allgemein, (i.e. nichtlinear) dann liefert eine

Linearkombination ausreichend vieler Basisfunktionen optimale Trennkraft/Klassifizierungsgüte (Weierstrass-Theorem)

In "Künstlichen Neuronalen Netwerken (ANN)" meist verwendet:

$$\mathbf{y(x)} = \sum_i^M w_{0i} A\left(w_{i0} + \sum_{j=1}^{D} w_{ij} \cdot x_j \right)$$

$$A(x) = \frac{1}{1-e^{x}}$$



y(x) ist Linearkombination von nichtlinearen Funktionen von Linearkombinationen der Eingangsgrößen/Observablenwerten

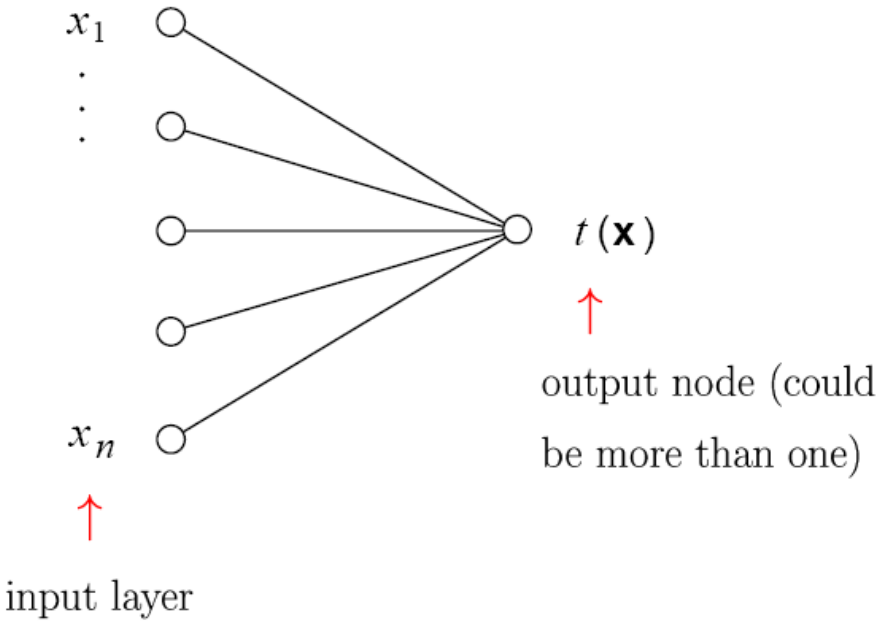
Einlagen Perzeptron

Motivation: aus Neuron-Synapsenstruktur im Gehirn
Neuron wird aktiviert durch Reize auf Eingangsneuronen
dies durch Synapsen verbunden sind

t(x): nichtlineare Antwort auf lineare Kombination der Eingangsgrößen

$$t(\vec{x}) = s\left(a_0 + \sum_{i=1}^{n} a_i x_i\right) \quad , \text{ where } s(u) \equiv (1+e^{-u})^{-1}$$ .Sigmoidftk.

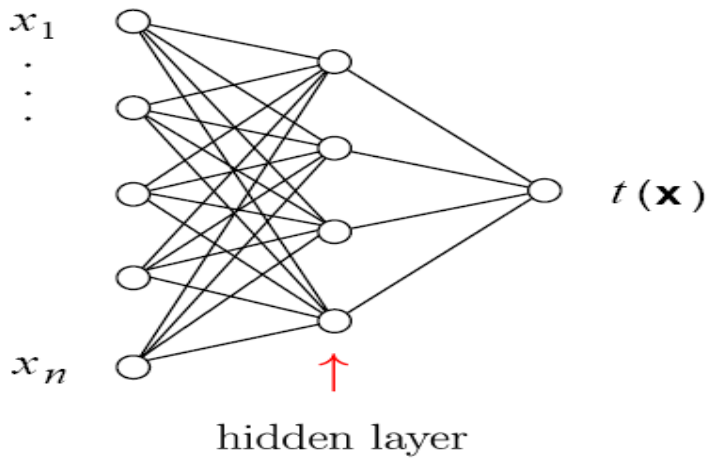
Diese Abbildung wird
Einlagen-Perzeptron genannt



$s(\cdot)$ ist monoton in
Fishers t(x)
→ äquivalent dazu

Das Multilagen-Perzeptron

Verallgemeinerung auf mehrlagiges Perzeptron:



Die Werte der "Neuronen" in der versteckten Lage sind:

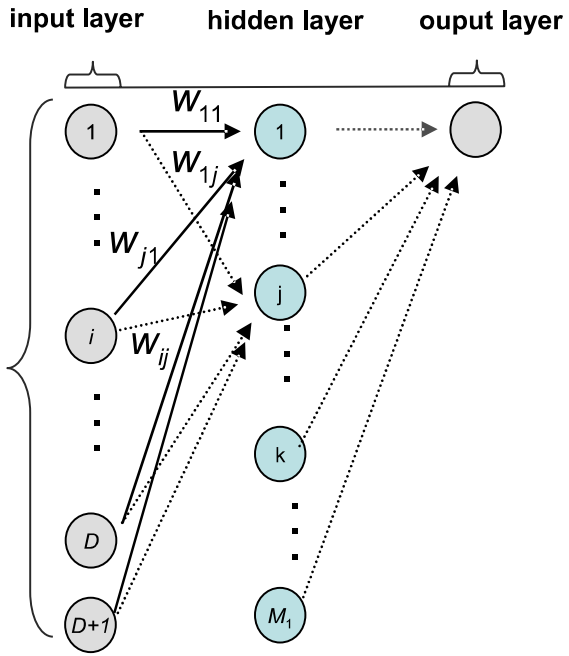
$$h_i(\vec{x}) = s\left(w_{i0} + \sum_{j=1}^{n} w_{ij}x_j\right) ,$$

Der Ausgangswert/Output des Netzwerks ist gegeben durch:

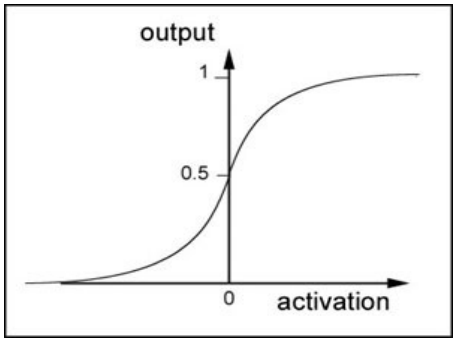
$$t(\vec{x}) = s\left(a_0 + \sum_{i=1}^{n} a_i h_i(\vec{x})\right)$$

$a_i, w_{ij} =$ Gewichte der Synapsen (Verbindungsstärken)

Das Multilagen-Perzeptron



output: $$y(x)=\sum_{i}^{M} w_{0i} A\left(w_{i0} + \sum_{j=1}^{D} w_{ij} \cdot x_j \right)$$



"Activation" function
e.g. sigmoid:

$$A(x) = \left(1 + e^{-x}\right)^{-1}$$

or tanh

Bestimmung der optimalen Gewichte im Trainingsprozess

Nur Vorwärtspropagation der Information ("Feed Forward Network")

Es könne auch mehrere versteckte Lagen vewendet werden.
In der Praxis maximal 2. Theorie sagt, dass 1 Lage im Prinzip schon ausreicht.

Training des Multilagen-Perzeptron

Für jedes Ereignis a kennen wir Observablenwerte und Zielwert des Output

$$\vec{x}_a = (x_1, \ldots, x_n) \qquad t_a = 0, 1$$

Sei ***w*** der Vektor aller anzupassenden Gewichte

Bestimmung der optimalen Gewichte durch Minimierung der Fehlerfunktion:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{a=1}^{N} |y(\vec{x}_a, \boldsymbol{w}) - t_a|^2 = \sum_{a=1}^{N} E_a(\boldsymbol{w})$$

Im Prinzip: auf allen Ereignissen auswerten und versuchen einen Multiparameterfit durchzuführen.
Liefert in Praxis kein gutes Ergebnis (sehr viele Nebenminima)

Alternative: Online-Lernen = Update der Gewichte nach Auswertung jedes Ereignisses

Training des Multilagen-Perzeptron

Zu Beginn: zufällige Zuweisung von Gewichten an Synapsen

Nach Prüfung der Ereignisse werden die Gewichte geändert gemäß:

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E\left(\boldsymbol{w}^{(\tau)}\right)$$

Lernrate $\eta > 0$

Erstaunlich schlechte Leistungsgüte wenn alle Ereignisse berücksichtigt werden.

In Praxis besser: Anpassung der Gewichte nach jedem Ereignis a:

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_a\left(\boldsymbol{w}^{(\tau)}\right)$$

Fehlerrückpropagation

Fehlerrückpropagation = Algorithmus zum Auffinden des Gradientenvektors

Die Netzwerkantwort kann geschieben werden als: $y(\boldsymbol{x}) = h(u(\boldsymbol{x}))$

wobei: $u(\vec{x}) = \sum_{j=0} w_{1j}^{(2)} \varphi_j(\vec{x})$ $\varphi_j(\vec{x}) = h\left(\sum_{k=0} w_{jk}^{(1)} x_k\right)$

Mit $\phi_0 = x_0 = 1$ die Summe erstreckt sich über alle Knoten/Neuronen in der vorherige Lage (+offset w_0)

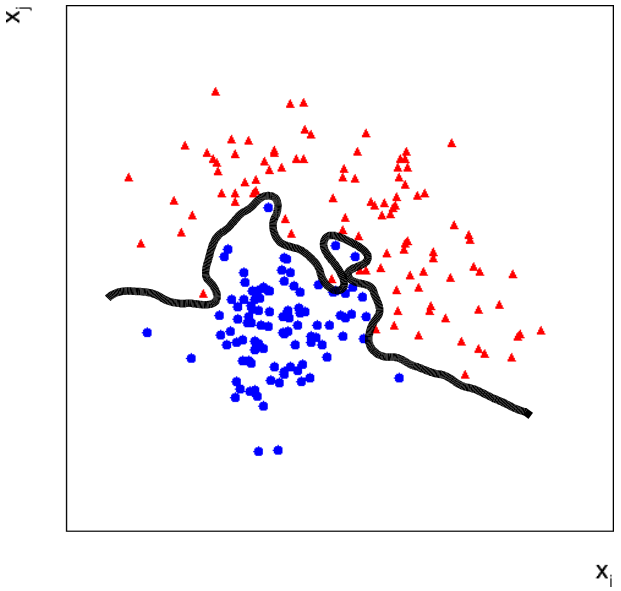
Zunächst Update der Synapsen zwischen versteckter Lage und Ausgang z.B. für Ereignis a:

$$\frac{\partial E_a}{\partial w_{1j}^{(2)}} = (y_a - t_a)\, h'(u(\vec{x}))\, \varphi_j(\vec{x})$$

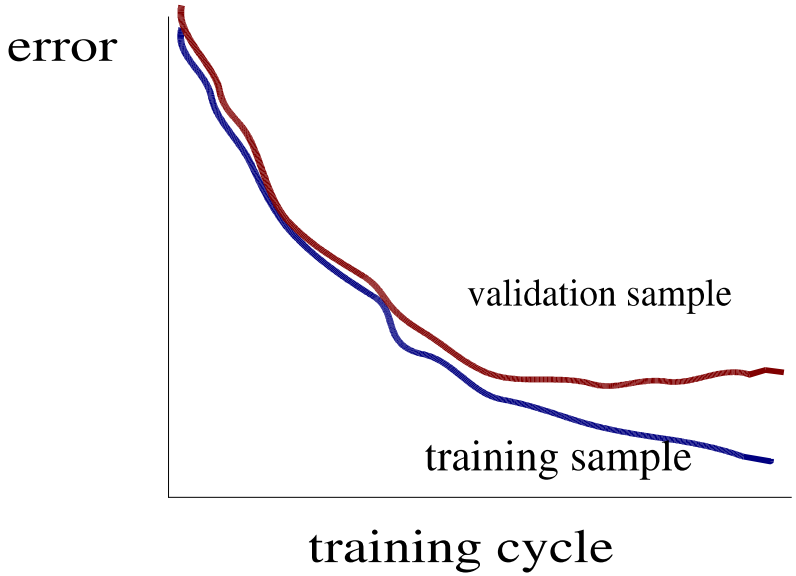
Dann Update der Synapsen zwischen Einganglage und versteckter Lage (mit Kettenregel)

Übertraining

Falls Netzwerk zu viele Knoten besitzt, und zu lange trainiert wird, wird sich Entscheidungsgrenze zu sehr an statistische Fluktuationen in Trainingsdatensatz anpassen:



Monitoring des Training:

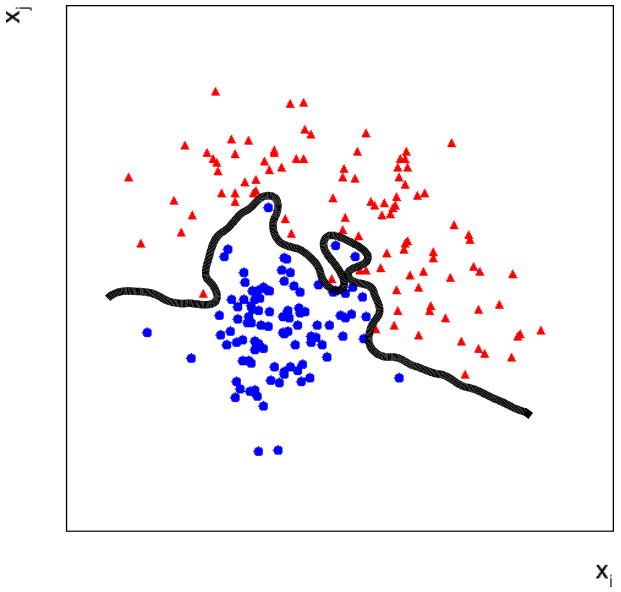


Überprüfe Fehlerrate mit Hilfe von statistisch unabhängigen Ereignissen, die nicht im Training verwendet werden.

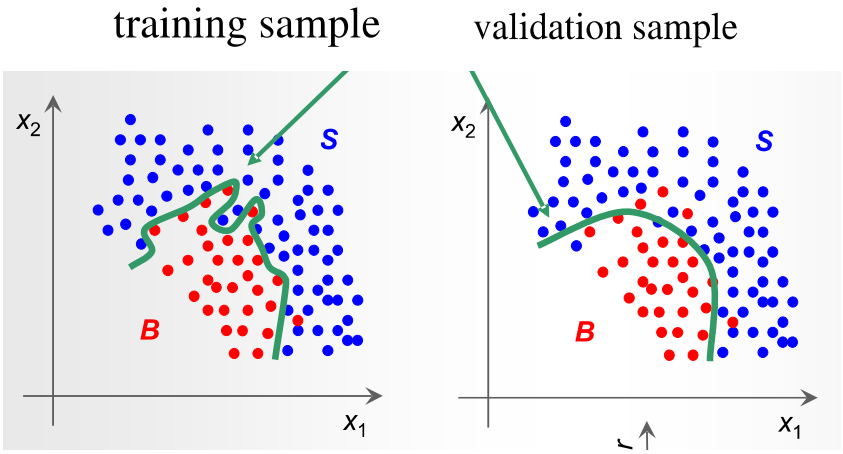
Beende Training, wenn beide Kurven sich voneinander trennen oder die Fehlerrate im Validierungssample wieder steigt.

Übertraining

Falls Netzwerk zu viele Knoten besitzt, und zu lange trainiert wird, wird sich Entscheidungsgrenze zu sehr an statistische Fluktuationen in Trainingsdatensatz anpassen:



Monitoring des Training:



Überprüfe Fehlerrate mit Hilfe von statistisch unabhängigen Ereignissen, die nicht im Training verwendet werden.

Beende Training, wenn beide Kurven sich voneinander trennen oder die Fehlerrate im Validierungssample wieder steigt.

Validierung und Tests

The validation sample can be used to make various choices about the network architecture, e.g., adjust the number of hidden nodes so as to obtain good "generalization performance" (ability to correctly classify unseen data).

If the validation stage is iterated may times, the estimated error rate based on the validation sample has a bias, so strictly speaking one should finally estimate the error rate with an independent test sample.

Rule of thumb if data not too expensive (Narsky):	train : validate : test 50 : 25 : 25

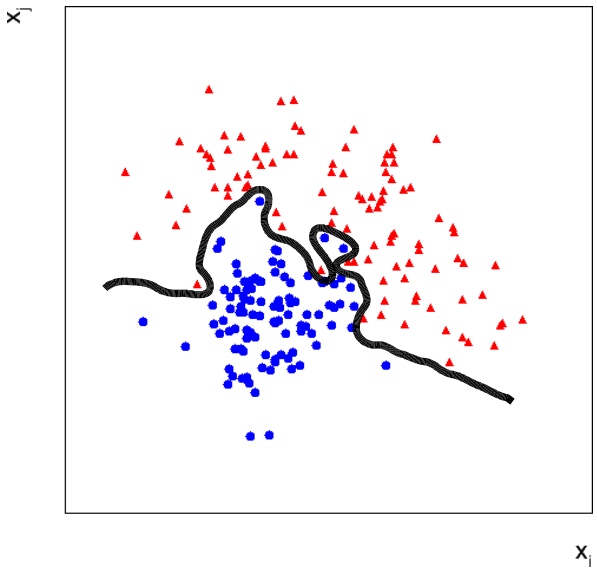
But this depends on the type of classifier. Often the bias in the error rate from the validation sample is small and one can omit the test step.

“Trade Off” zwischen “Bias” und “Varianz”

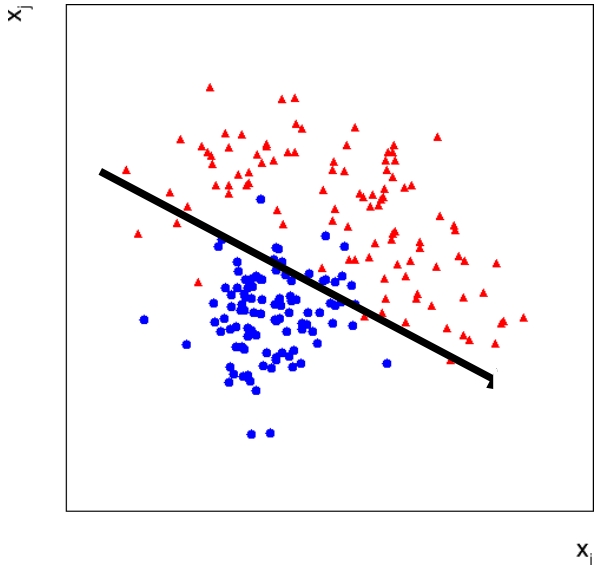
Für endliche Anzahl an Trainigsereignissen und steigender Anzahl an Neuronen haben gewichte große stat.Fehler (→ großer Varianz, Übertraining)

Für zu kleine Anzahl von Neuronen kann das Netwerk nicht ausreichend die Nichtlinearitäten berücksichtigen (→ Bias)

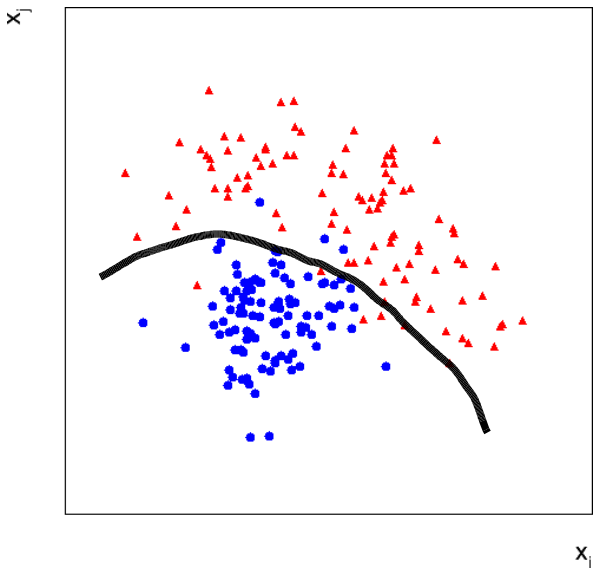
Bias und Varianz sind wiederrum gegenläufige Kriterien.



high variance



high bias



good trade-off

Struktur des Netzwerkes

Easy to generalize to arbitrary number of layers.

Feed-forward net: values of a node depend only on earlier layers, usually only on previous layer ("network architecture").

More nodes → neural net gets closer to optimal $t(\boldsymbol{x})$, but more parameters need to be determined.

Parameters usually determined by minimizing an error function,

$$\mathcal{E} = E_0[(t - t^{(0)})^2] + E_1[(t - t^{(1)})^2] \,,$$

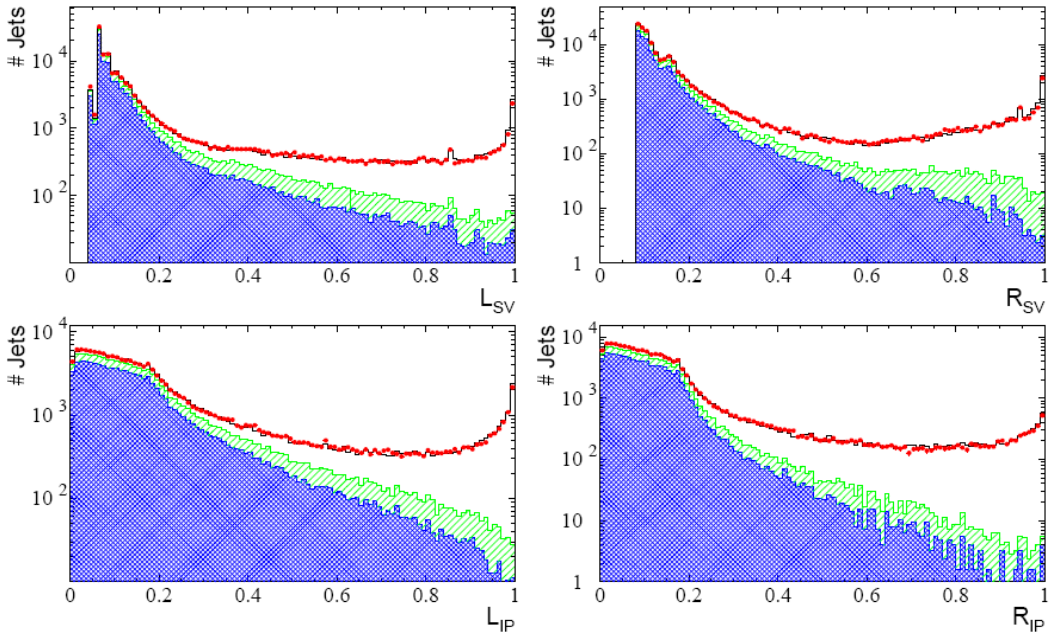
where t (0) , t (1) are target values, e.g., 0 and 1 for logistic sigmoid.

Expectation values replaced by averages of training data (e.g. MC).

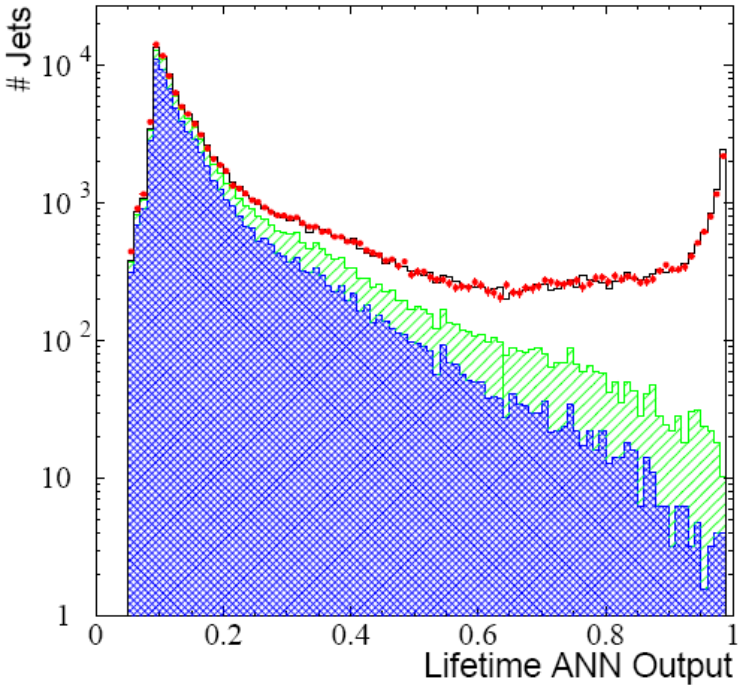
In general training can be difficult; standard software available.

Beispiel von ANN bei LEP2 (OPAL)

ANN input variables



ANN output



Struktur des Netzwerkes

Theorem: An MLP with a single hidden layer having a sufficiently large number of nodes can approximate arbitrarily well the Bayes optimal decision boundary. K. Weierstrass-Theorem

Holds for any continuous non-polynomial activation function

Leshno, Lin, Pinkus and Schocken (1993), *Neural Networks* **6**, 861—867

In practice often choose a single hidden layer and try increasing the the number of nodes until no further improvement in performance is found.

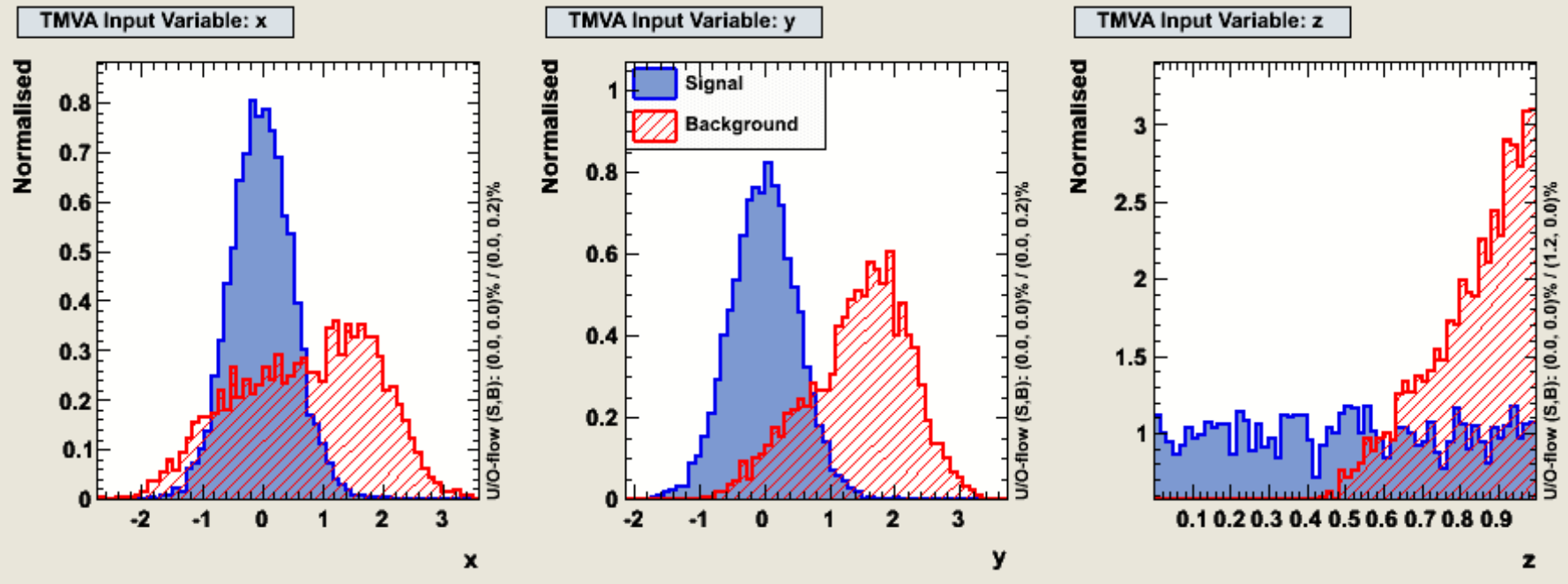
Struktur des Netzwerkes

"Relatively little is known concerning the advantages and disadvantages of using a single hidden layer with many units (neurons) over many hidden layers with fewer units. The mathematics and approximation theory of the MLP model with more than one hidden layer is not well understood."

"Nonetheless there seems to be reason to conjecture that the two hidden layer model may be significantly more promising than the single hidden layer model, ..."

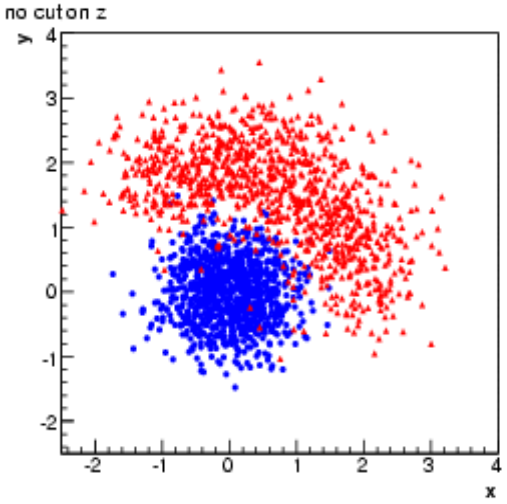
A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numerica (1999), pp. 143—195.

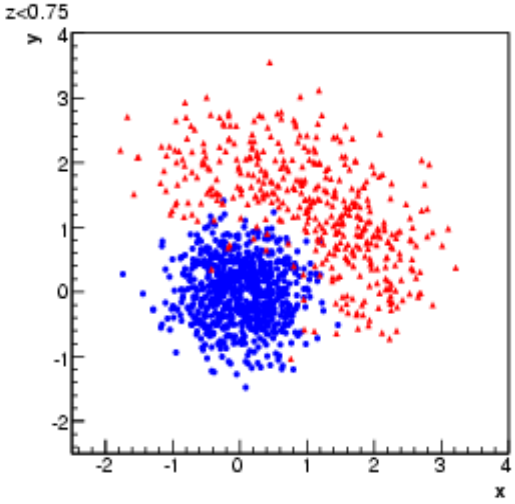
Ein vergleichendes Beispiel



Ein vergleichendes Beispiel

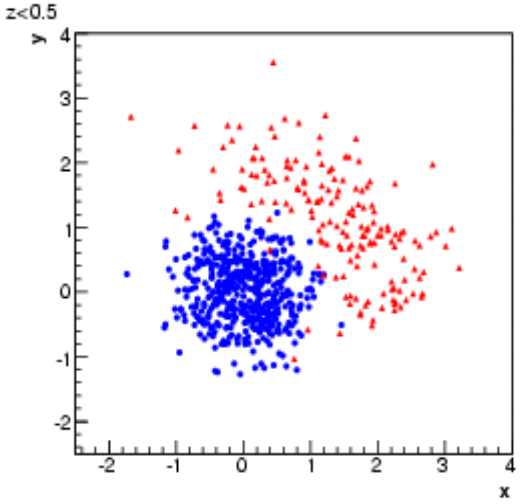
no cut on z

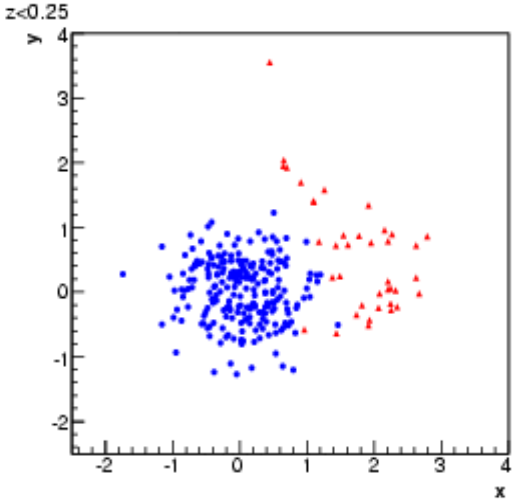




$z < 0.75$

$z < 0.5$

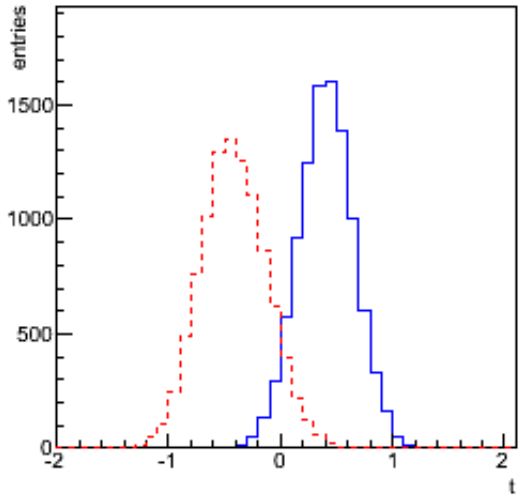


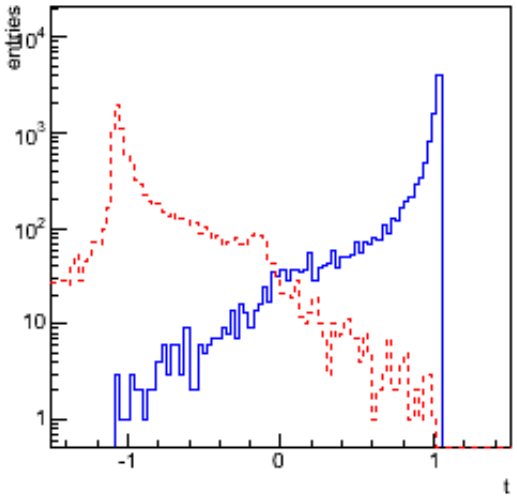


$z < 0.25$

Ein vergleichendes Beispiel

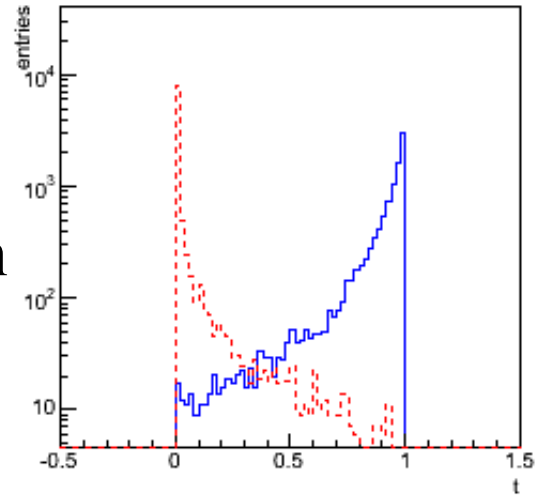
Fisher

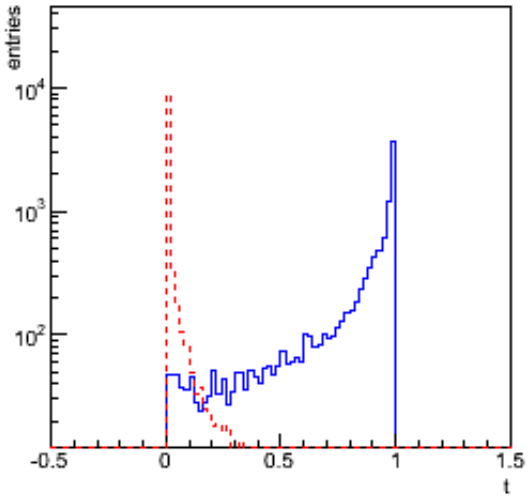




ANN

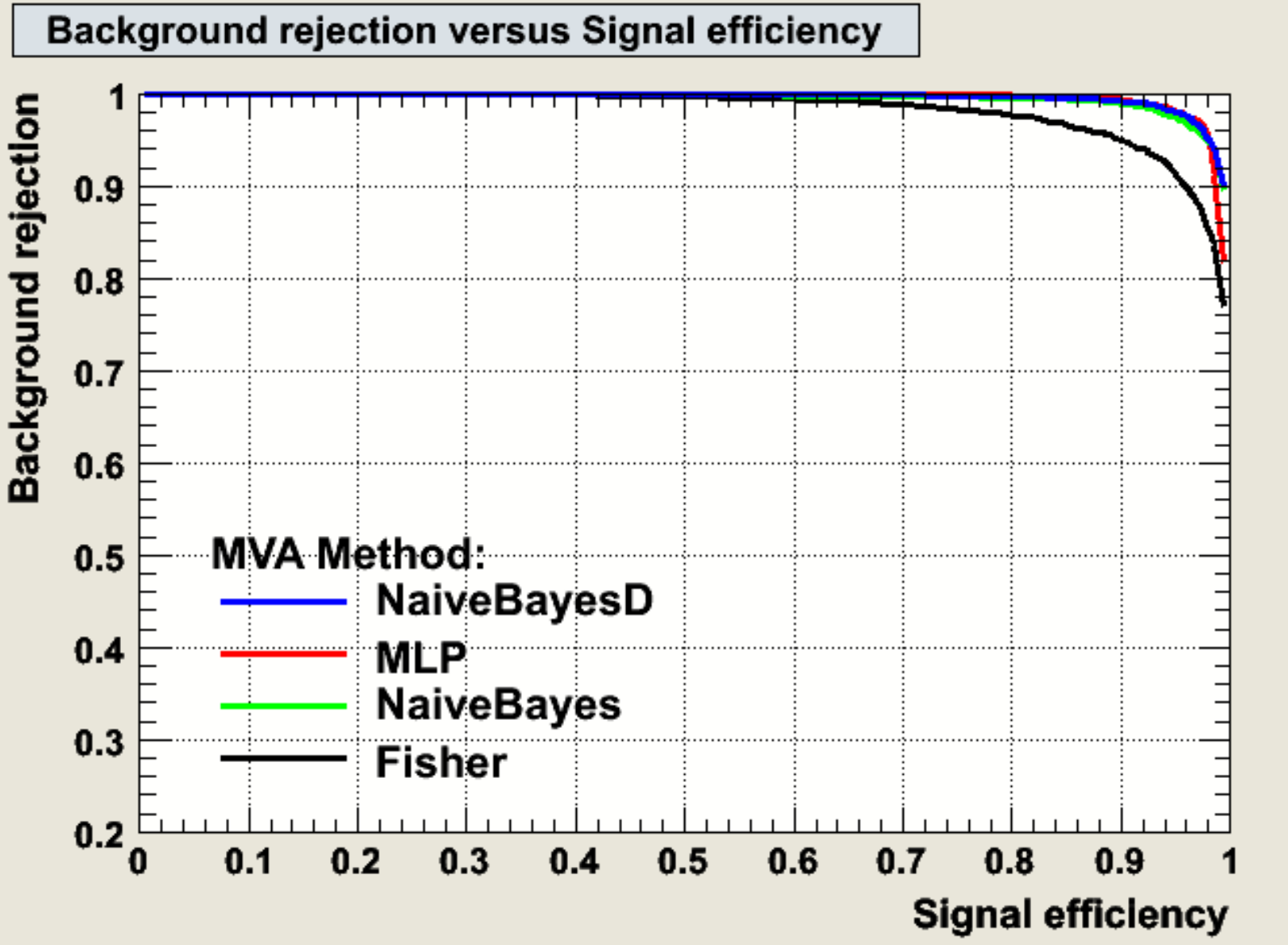
Likelihood ohne Dekorrelation





Likelihood mit Dekorrelation

Ein vergleichendes Beispiel



Ein vergleichendes Beispiel