Statistical analysis of experimental data
Classification

Aleksander Filip Żarnecki

FACULTY OF PHYSICS
UNIVERSITY OF WARSAW

Lecture 11

January 12, 2023

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Hypothesis Testing

Problem

So far, we focused on the problem of extracting model parameters from the collected data sample. We used maximum likelihood approach (or χ^2 minimization, which is a special case).

However, what we often want to do is to "make choice", discriminate between two (or more) hypothesis based on the collected data.

We already addressed this problem (partially) when discussing limits (lecture 06) and consistency of the fit (lecture 07).

The general formulation of the problem: how to discriminate between two model hypothesis H_0 and H_1 based on the collected data D?

Common case:
H_0 - Standard Model is valid, H_1 - SM + additional BSM contribution
D - the whole collected data sample, subset, or a single measurement

Neyman–Pearson Lemma

According to Neymann and Pearson, the optimal, "most powerful" method to discriminate between the two hypothesis is to look at likelihood ratio

$$Q(D) = \frac{L(D|H_1)}{L(D|H_0)}$$

When considering single measurements, making a cut on Q(x) is the optimal way to classify events. By using likelihood ratio, multi-dimensional measurements (whole events) are also presented as single number...

When we consider the whole sample of collected data, value of Q(D) is the best discriminant between the two hypothesis.

Still, one needs to compare the value of $Q(D)$ resulting from the measurement, with the expected Q distributions for the two hypothesis.

CL$_s$ method

The two hypothesis we consider in this case:
H_0 - Standard Model without Higgs contribution - "background" only (b)
H_1 - SM with Higgs contribution - "signal+background" (s+b)
where we can consider different masses of the Higgs, m$_H$

Instead of using Q, it is more convenient to use

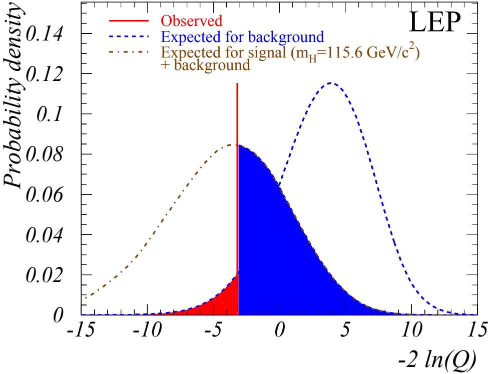
$$q \;=\; -2\ln Q \;=\; -2\ell(D|H_1) + 2\ell(D|H_0) \;=\; \chi^2(D|H_1) - \chi^2(D|H_0)$$

where:

- positive q values are expected for data more in agreement with background only hypothesis (H_0)
- negative q values indicate that data are better described by signal+background hypothesis (H_1)

CL$_s$ method

Value of q from LEP, q_{dat}, was compared with distribution obtained with multiple Monte Carlo experiments for $m_H = 115.6$ GeV.



We can define

$$CL_{s+b} = \int_{q_{dat}}^{+\infty} dq\, f^{H1}(q)$$

⇐ indicated as blue area

$$CL_b = \int_{q_{dat}}^{+\infty} dq\, f^{H0}(q)$$

⇐ indicated as red is $1 - CL_b$

Hypothesis Testing

$\mathrm{CL_s}$ method

Experiments at LEP, running with energy up to $\sqrt{s} = 210$ GeV, could only observe Higgs bosons with mass of up to about 118 GeV

For higher masses, signal+background hypothesis (H_1) becomes indistinguishable from background only one (H_0)...

Hypothesis Testing

$\mathbf{CL_s}$ method

Experiments at LEP, running with energy up to $\sqrt{s} = 210$ GeV, could only observe Higgs bosons with mass of up to about 118 GeV

For higher masses, signal+background hypothesis (H_1) becomes indistinguishable from background only one (H_0)...

With tight event selection, experiments observed 4 candidate events with $m_H^{rec} > 109 GeV$. Expectations of background only hyposthesis: $b = 1.2$

In strictly frequentiest approach we could exclude (on 95%CL) not only the SM, but also all Higgs scenarios (H_1) with $m_H > 118 GeV$!..

CL$_s$ method

Experiments at LEP, running with energy up to $\sqrt{s} = 210$ GeV, could only observe Higgs bosons with mass of up to about 118 GeV For higher masses, signal+background hypothesis (H_1) becomes indistinguishable from background only one (H_0)...

With tight event selection, experiments observed 4 candidate events with $m_H^{rec} > 109 GeV$. Expectations of background only hyposthesis: $b = 1.2$

In strictly frequentiest approach we could exclude (on 95%CL) not only the SM, but also all Higgs scenarios (H_1) with $m_H > 118 GeV$!..

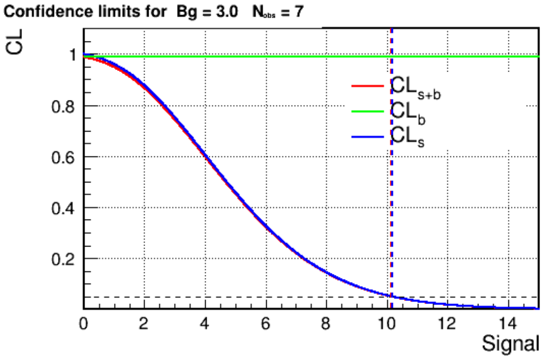
Frequentist approach gives us result which is correct (from statistical point of view) but not very useful... Too sensitive to background fluctuations?

Solution is to look for confidence level of H_1 relative to H_0:

$$CL_s \quad = \quad \frac{CL_{s+b}}{CL_b}$$

$\mathrm{CL_s}$ method example

Counting experiment with expected background $\mu_{bg} = 3$ and $N_{obs} = 7$



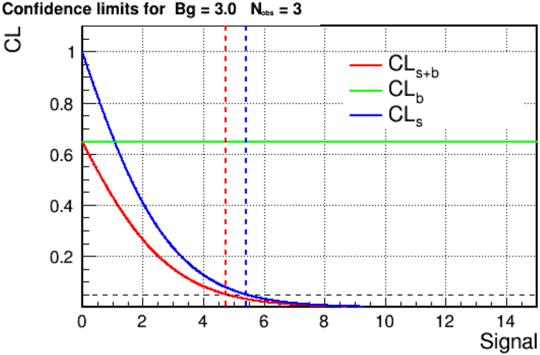
Probability of background hypothesis to result in $N_{obs} \leq 7$ is 98.8%

⇒ $\mathrm{CL_s}$ limit on number of signal events is 10.17 (95% CL)
almost the same as the Frequentist limit ($\mathrm{CL_{s+b}}$): 10.15

Hypothesis Testing

$\mathrm{CL_s}$ method example

Counting experiment with expected background $\mu_{bg} = 3$ and $N_{obs} = 3$



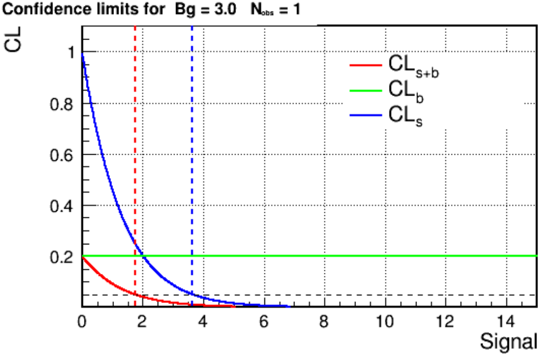
Probability of background hypothesis to result in $N_{obs} \leq 3$ is 64.7%

⇒ $\mathrm{CL_s}$ limit on number of signal events is 5.40 (95% CL)
only slightly higher than the Frequentist limit ($\mathrm{CL_{s+b}}$): 4.75

Hypothesis Testing

CL$_s$ method example

Counting experiment with expected background $\mu_{bg} = 3$ and $N_{obs} = 1$



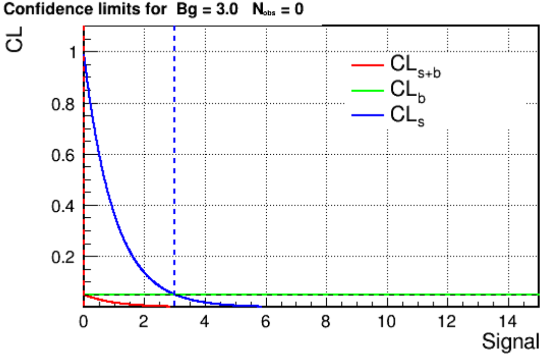
Probability of background hypothesis to result in $N_{obs} \leq 1$ is 19.9%

$\Rightarrow$ CL$_s$ limit on number of signal events is 3.64 (95% CL)
significantly higher than the Frequentist limit (CL$_{s+b}$): 1.74

Hypothesis Testing

CL$_s$ method example

Counting experiment with expected background $\mu_{bg} = 3$ and $N_{obs} = 0$



Probability of background hypothesis to result in $N_{obs} = 0$ is 4.98%

⇒ CL$_s$ limit on number of signal events is 3.00 (95% CL)
while all signal hypothesis are excluded in Frequentist approach (CL$_{s+b}$)!

CL$_s$ method

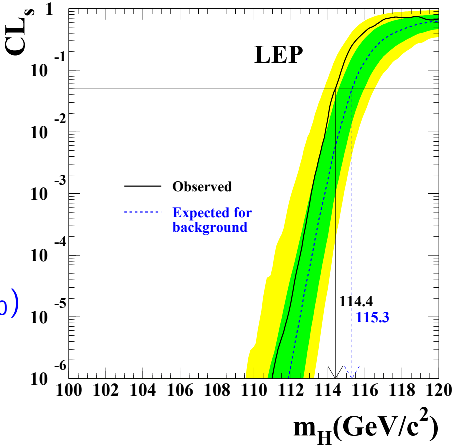
In the modified approach, we exclude (at 95% CL) all scenarios with

$$CL_s \quad < 0.05$$

This means that the probability of H_1 to reproduce the collected data is less than 5% of the SM probability:

$$P(q > q_{dat}|H_1) \; < \; 0.05 \; P(q > q_{dat}|H_0)$$

Final Higgs limits from LEP



Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Event classification

Problem definition

The problem is similar to the one discussed in lecture 10: we want to discriminate between two model hypothesis H_0 and H_1 based on the collected data D.

Common case - interpretation of measurement results:

- H_0 - Standard Model is valid,
- H_1 - SM has to be extended by adding BSM contribution
- D - the whole collected data sample

According to Neymann and Pearson, the optimal, “most powerful” method to discriminate between the two hypothesis is to look at likelihood ratio

$$Q(D) = \frac{L(D|H_1)}{L(D|H_0)}$$

Event classification

Problem definition

The problem is similar to the one discussed in lecture 10: we want to discriminate between two model hypothesis H_0 and H_1 based on the collected data D.

Different case - **classification of collected measurements**:

- H_0 - measurement can be attributed to the Standard Model,
- H_1 - measurement is due to BSM contribution,
- D - single measurement ("event" in HEP experiments)

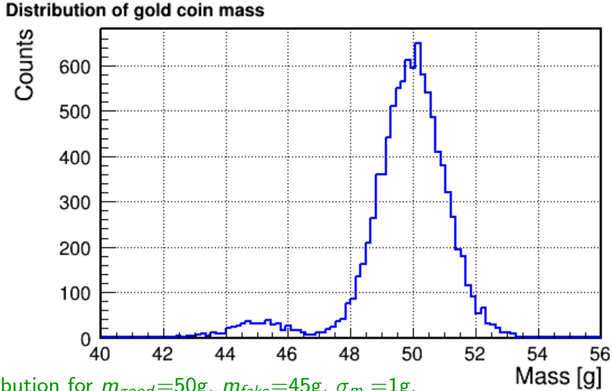
According to Neymann and Pearson, the optimal, "most powerful" method to discriminate between the two hypothesis is to look at likelihood ratio

$$Q(D) = \frac{L(D|H_1)}{L(D|H_0)}$$

Event classification

Simple example

Fake gold coins have lower mass than the true ones.
We can select good coins (reject fake coins) by measuring the mass...

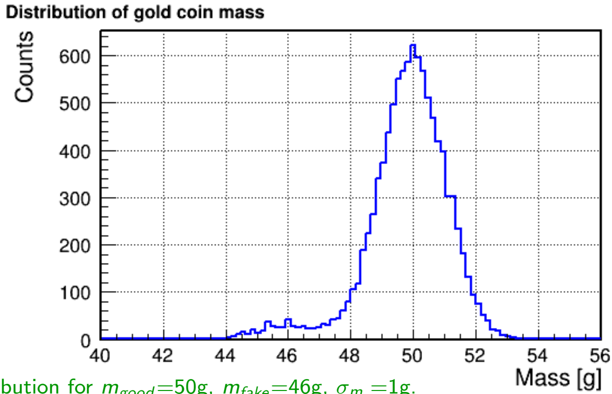


Example distribution for m_{good}=50g, m_{fake}=45g, σ_m =1g.

Fake coin contribution (5%) clearly visible (5σ separation)...

Simple example

Fake gold coins have lower mass than the true ones.
We can select good coins (reject fake coins) by measuring the mass...



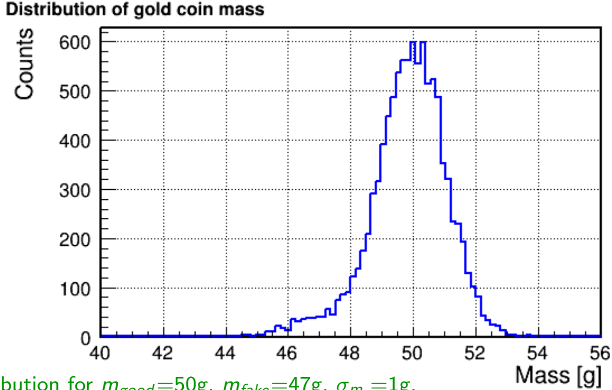
Example distribution for m_{good}=50g, m_{fake}=46g, σ_m =1g.

Not so obvious when the mass difference is smaller...

Event classification

Simple example

Fake gold coins have lower mass than the true ones.
We can select good coins (reject fake coins) by measuring the mass...



Example distribution for m_{good}=50g, m_{fake}=47g, σ_m =1g.

Not so obvious when the mass difference is smaller... How to describe it?

Event classification

Simple example

We use the Neymann–Pearson Lemma directly:

$$Q(m) = \frac{L(m|H_1)}{L(m|H_0)} \quad \text{with} \quad \begin{array}{l} H_1 - \text{good coin} \\ H_0 - \text{fake coin} \end{array}$$

Assuming Gaussian uncertainties of the mass measurement

$$Q(m) = \frac{G(m; m_1, \sigma)}{G(m; m_0, \sigma)}$$

Event classification

Simple example

We use the Neymann–Pearson Lemma directly:

$$Q(m) \quad = \quad \frac{L(m|H_1)}{L(m|H_0)} \quad \text{with} \quad \begin{array}{l} H_1 - \text{good coin} \\ H_0 - \text{fake coin} \end{array}$$

Assuming Gaussian uncertainties of the mass measurement

$$\begin{aligned} Q(m) \quad &= \quad \frac{G(m; m_1, \sigma)}{G(m; m_0, \sigma)} \\ &= \quad \exp\left[-\frac{1}{2}\left(\frac{m - m_1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{m - m_0}{\sigma}\right)^2 \right] \end{aligned}$$

Event classification

Simple example

We use the Neymann–Pearson Lemma directly:

$$Q(m) = \frac{L(m|H_1)}{L(m|H_0)} \quad \text{with} \quad \begin{array}{l} H_1 - \text{good coin} \\ H_0 - \text{fake coin} \end{array}$$

Assuming Gaussian uncertainties of the mass measurement

$$\begin{aligned} Q(m) &= \frac{G(m; m_1, \sigma)}{G(m; m_0, \sigma)} \\ &= \exp\left[-\frac{1}{2}\left(\frac{m-m_1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{m-m_0}{\sigma}\right)^2 \right] \\ &= \exp\left[\left(\frac{m_1-m_0}{\sigma^2}\right)\left(m - \frac{m_1+m_0}{2}\right) \right] \end{aligned}$$

Event classification

Simple example

Q is not very convenient to use, changes from very small to very large values. We already considered (LEP Higgs limits):

$$q(m) \;=\; -2\;\log Q(m) \;\;=\;\; -2\left(\frac{m_1 - m_0}{\sigma^2}\right)\left(m - \frac{m_1 + m_0}{2}\right)$$

Event classification

Simple example

Q is not very convenient to use, changes from very small to very large values. We already considered (LEP Higgs limits):

$$q(m) = -2 \log Q(m) = -2\left(\frac{m_1 - m_0}{\sigma^2}\right)\left(m - \frac{m_1 + m_0}{2}\right)$$

One can also consider discriminator function: $-1 < y < +1$

$$y(m) = \frac{1-Q}{1+Q} = \frac{L(m|H_1) - L(m|H_0)}{L(m|H_1) + L(m|H_0)}$$

$$= \tanh\left(-\frac{q(m)}{4}\right) = \tanh\left(\left(\frac{m_1 - m_0}{2\sigma^2}\right)\left(m - \frac{m_1 + m_0}{2}\right)\right)$$

Event classification

Simple example

Q is not very convenient to use, changes from very small to very large values. We already considered (LEP Higgs limits):

$$q(m) = -2 \log Q(m) = -2 \left(\frac{m_1 - m_0}{\sigma^2} \right) \left(m - \frac{m_1 + m_0}{2} \right)$$

One can also consider discriminator function: $-1 < y < +1$

$$y(m) = \frac{1-Q}{1+Q} = \frac{L(m|H_1) - L(m|H_0)}{L(m|H_1) + L(m|H_0)}$$

$$= \tanh\left(-\frac{q(m)}{4} \right) = \tanh\left(\left(\frac{m_1 - m_0}{2\sigma^2} \right) \left(m - \frac{m_1 + m_0}{2} \right) \right)$$

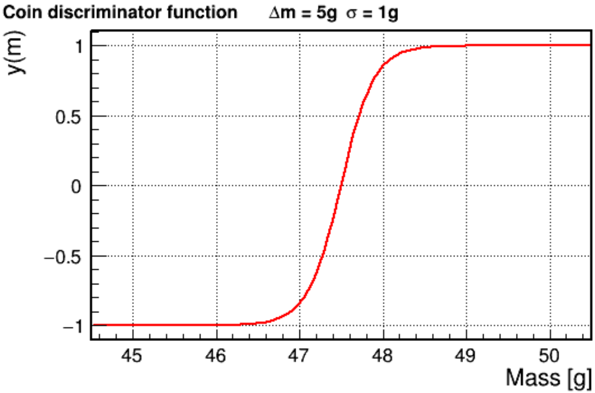
Final selection should be based on a cut: $y(m) > y_{cut}$ or $Q(m) > Q_{cut}$ ...

Event classification

Simple example

Good vs fake coin discriminator function

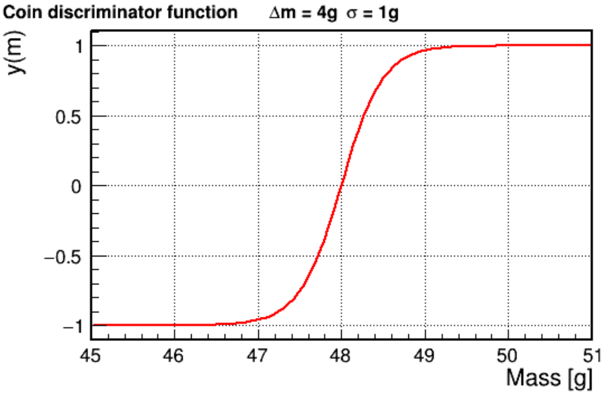
We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



Simple example

Good vs fake coin discriminator function

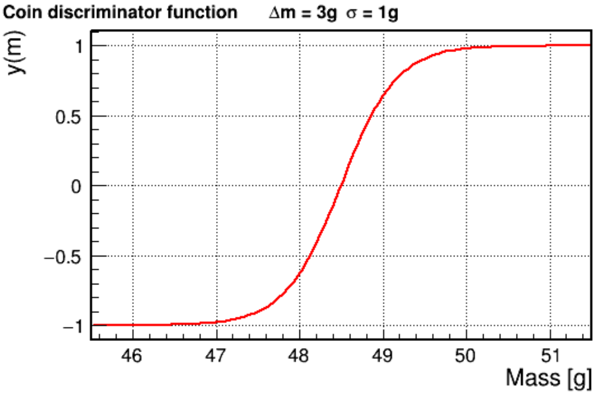
We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



Simple example

Good vs fake coin discriminator function

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin

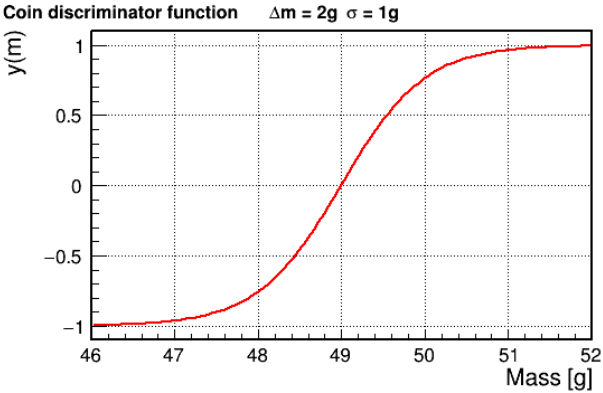


Event classification

Simple example

Good vs fake coin discriminator function

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



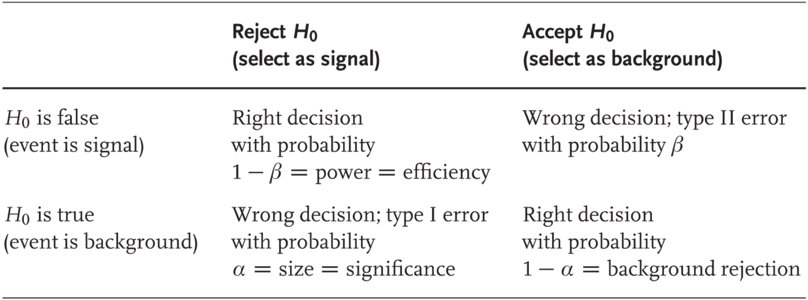
Classification gets more and more difficult with decreasing Δm...

Event classification

Classification errors

O.Behnke et. al, *Data Analysis in High Enegy Physics*

Selecting the classification cut, two types of error need to be considered



	Reject H_0 (select as signal)	**Accept H_0 (select as background)**
H_0 is false (event is signal)	Right decision with probability $1-\beta$ = power = efficiency	Wrong decision; type II error with probability β
H_0 is true (event is background)	Wrong decision; type I error with probability α = size = significance	Right decision with probability $1-\alpha$ = background rejection

Probability of accepting fake

$$\alpha \;=\; \int\limits_{y(m)>y_{cut}} dm\; p(m|H_0)$$

Probability of rejecting good

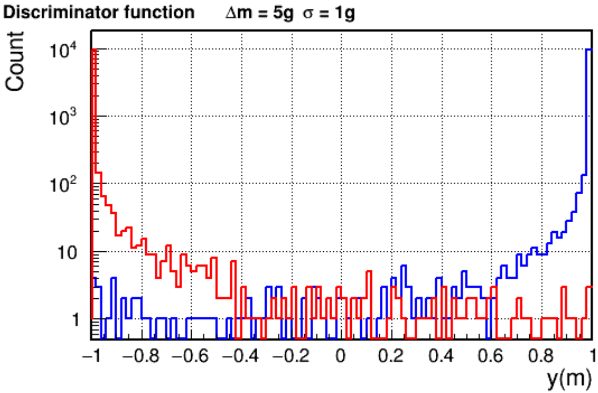
$$\beta \;=\; \int\limits_{y(m)<y_{cut}} dm\; p(m|H_1)$$

Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



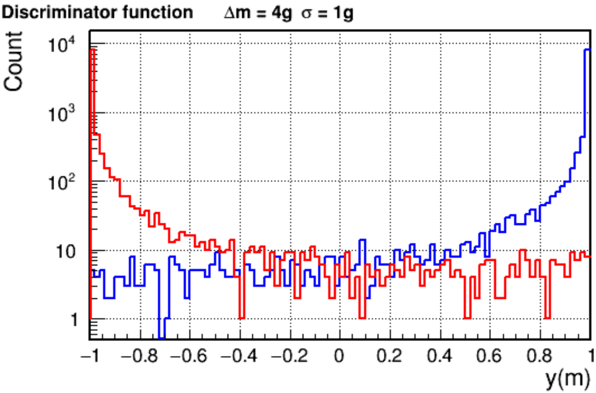
Large separation in $\Delta m \Rightarrow$ very efficient classification possible

Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin

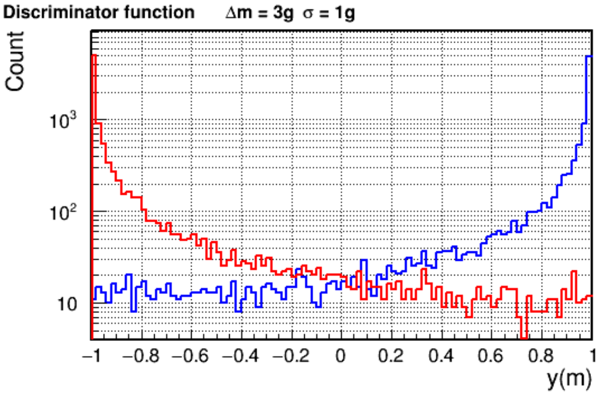


Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



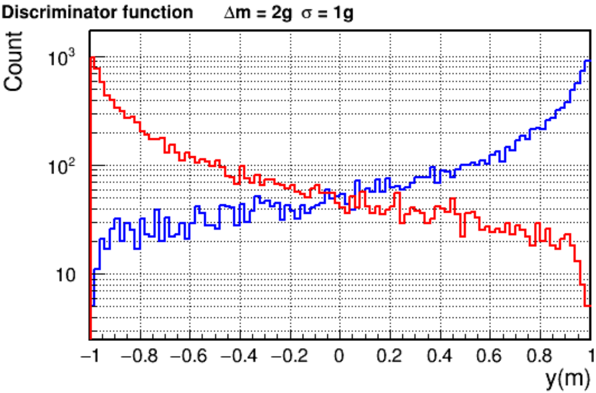
Classification still possible, but error rate substantial

Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



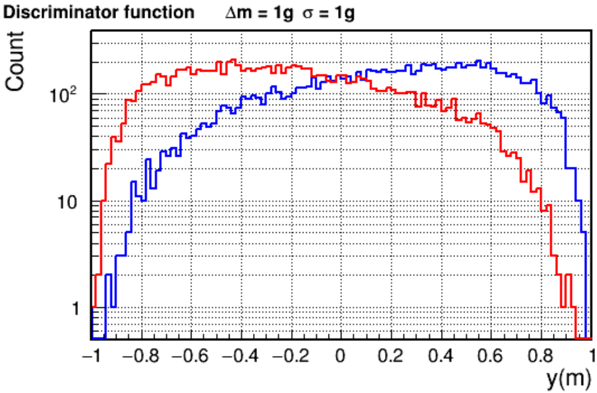
Classification still possible, but error rate substantial

Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



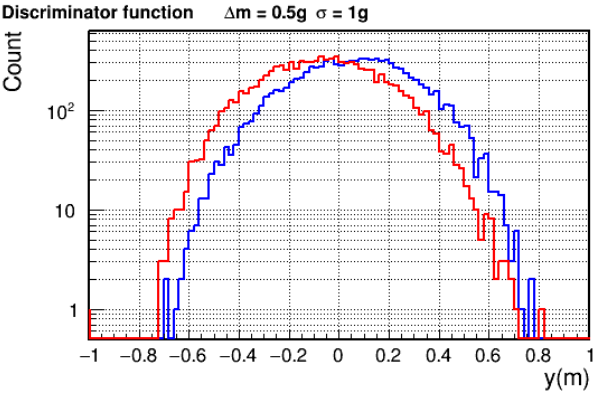
Efficient classification "coin by coin" no longer possible...

Event classification

Simple example

Discriminator function distribution

We expect $y \to -1$ for fake coin, $y \to +1$ for good coin



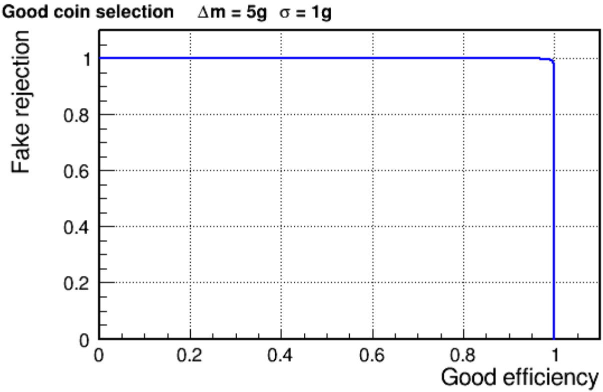
Efficient classification “coin by coin” no longer possible...

Event classification

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:

$1-\alpha$

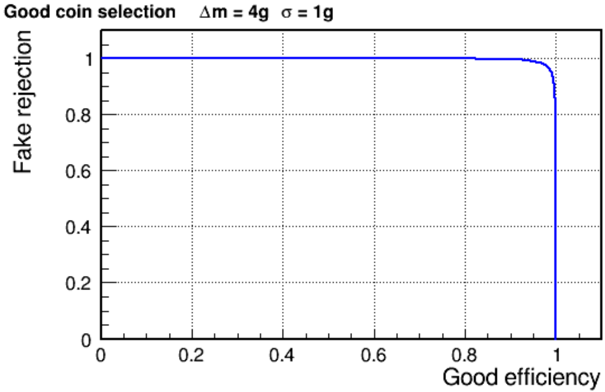


$1-\beta$

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:

$1 - \alpha$



$1 - \beta$

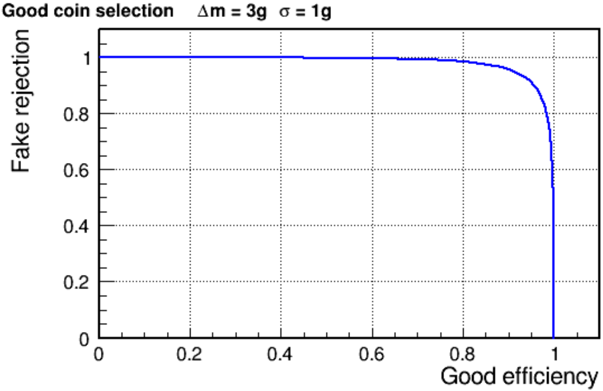
In the realistic case, we can not have $\alpha \to 0$ and $\beta \to 0$ at the same time...

Event classification

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:

$1 - \alpha$



$1 - \beta$

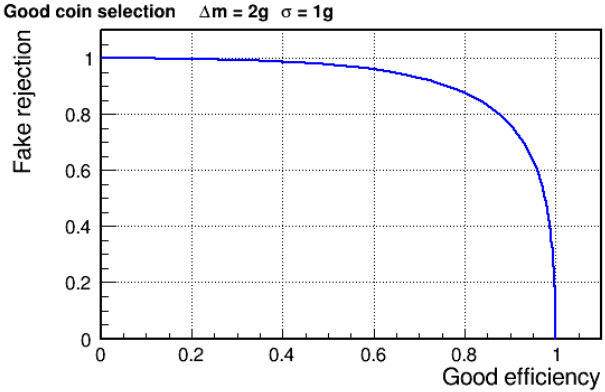
In the realistic case, we can not have $\alpha \rightarrow 0$ and $\beta \rightarrow 0$ at the same time...

Event classification

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:

$1 - \alpha$



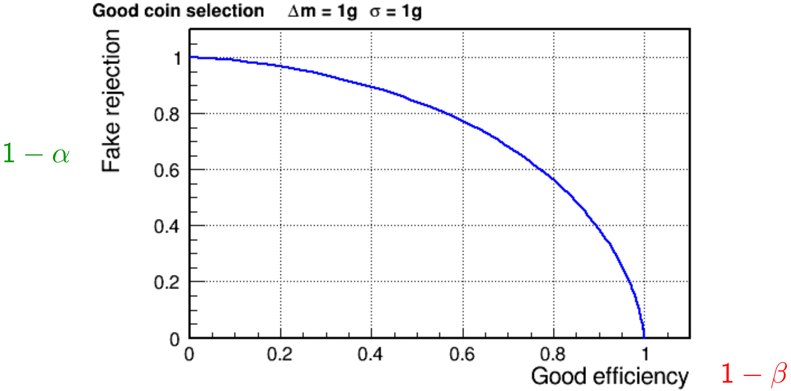
$1 - \beta$

Optimal cut value strongly depends on the actual goal of the analysis...

Event classification

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:



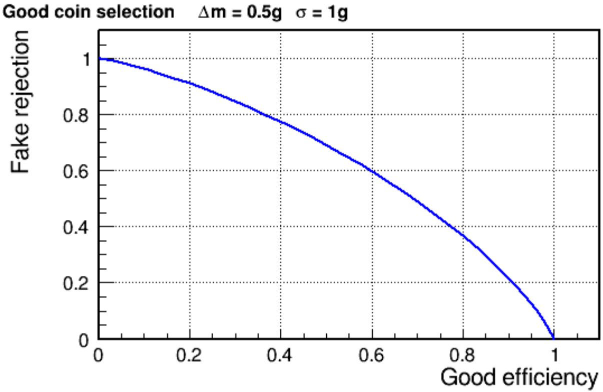
Optimal cut value strongly depends on the actual goal of the analysis...

Event classification

ROC curve

For both good and fake coins, efficiency depends on the assumed y_{cut} value. All possible choices on a Receiver-Operating-Characteristic curve:

$1-\alpha$



$1-\beta$

Optimal cut value strongly depends on the actual goal of the analysis...

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Naive Bayes Classifier

Problem definition

According to Neymann and Pearson, the optimal, "most powerful" method to discriminate between the two hypothesis is to look at likelihood ratio

$$Q(\mathbf{x}) = \frac{L(\mathbf{x}|H_1)}{L(\mathbf{x}|H_0)}$$

where we now consider classification of single measurement **x**.

However, this can be directly used only, if the likelihoods are known.
For example (in particle physics), if we know the differential cross sections for the considered signal and background processes, and detector effects can be neglected...

Naive Bayes Classifier

Problem definition

According to Neymann and Pearson, the optimal, “most powerful” method to discriminate between the two hypothesis is to look at likelihood ratio

$$Q(\mathbf{x}) = \frac{L(\mathbf{x}|H_1)}{L(\mathbf{x}|H_0)}$$

where we now consider classification of single measurement **x**.

However, this can be directly used only, if the likelihoods are known.
For example (in particle physics), if we know the differential cross sections for the considered signal and background processes, and detector effects can be neglected...

In most cases, we need to decide on the selection procedure based on the data (or pseud-data from Monte Carlo simulation) itself, try to use it to 'reconstruct' the likelihood ratio dependence on **x**...

Naive Bayes Classifier

Bayes' Theorem

refer to lecture 01

We can try to apply Bayes' Theorem to the classification problem.
We can ask for the "probability" of the considered hypothesis H for given outcome $\mathbf{x}$ (data) of the measurement:

$$P(H|\mathbf{x}) \quad = \quad \frac{P(\mathbf{x}|H)}{P(\mathbf{x})} \cdot P(H)$$

Naive Bayes Classifier

Bayes' Theorem

refer to lecture 01

We can try to apply Bayes' Theorem to the classification problem.
We can ask for the "probability" of the considered hypothesis H for given outcome $\mathbf{x}$ (data) of the measurement:

$$P(H|\mathbf{x}) \;=\; \frac{P(\mathbf{x}|H)}{P(\mathbf{x})} \cdot P(H)$$

Let us assume that we know the probability density functions (properly normalized) for the two considered hypothesis:

$$p_0(\mathbf{x}) \;=\; P(\mathbf{x}|H_0) \qquad p_1(\mathbf{x}) \;=\; P(\mathbf{x}|H_1)$$

and the expected fraction of events corresponding to H_1: f_1.

Naive Bayes Classifier

Bayes' Theorem

refer to lecture 01

We can try to apply Bayes' Theorem to the classification problem.
We can ask for the "probability" of the considered hypothesis H for given outcome $\mathbf{x}$ (data) of the measurement:

$$P(H|\mathbf{x}) \;=\; \frac{P(\mathbf{x}|H)}{P(\mathbf{x})} \cdot P(H)$$

Let us assume that we know the probability density functions (properly normalized) for the two considered hypothesis:

$$p_0(\mathbf{x}) \;=\; P(\mathbf{x}|H_0) \qquad p_1(\mathbf{x}) \;=\; P(\mathbf{x}|H_1)$$

and the expected fraction of events corresponding to H_1: f_1. Then:

$$P(H_1|\mathbf{x}) \;=\; \frac{f_1 \cdot p_1(\mathbf{x})}{f_1 \; p_1(\mathbf{x}) + (1 - f_1) \; p_0(\mathbf{x})}$$

Naive Bayes Classifier

Likelihood classifier

Single measurement (event) often corresponds to a set of observables:

$$\mathbf{x} \quad = \quad (x_1, x_2, \ldots x_N)$$

If N is large, it is difficult to reconstruct probability density function of $\mathbf{x}$.

Likelihood classifier

Single measurement (event) often corresponds to a set of observables:

$$\mathbf{x} \;=\; (x_1, x_2, \ldots x_N)$$

If N is large, it is difficult to reconstruct probability density function of $\mathbf{x}$.

We usually start from considering probabilities for single variable:

$$p_k^{(j)}(x_j) \;=\; P(x_j|H_k) \;=\; \int \cdots \int_{i \neq j} dx_i \; P(\mathbf{x}, H_k) \qquad k = 1, 2$$

Naive Bayes Classifier

Likelihood classifier

Single measurement (event) often corresponds to a set of observables:

$$\mathbf{x} \;=\; (x_1, x_2, \ldots x_N)$$

If N is large, it is difficult to reconstruct probability density function of **x**.

We usually start from considering probabilities for single variable:

$$p_k^{(j)}(x_j) \;=\; P(x_j|H_k) \;=\; \int \cdots \int_{i \neq j} dx_i \; P(\mathbf{x}, H_k) \qquad k = 1, 2$$

We can then apply the Bayes' Theorem to single variable distribution:

$$P(H_1|x_j) \;=\; \frac{f_1 \cdot p_1^{(j)}(x_j)}{f_1 \; p_1^{(j)}(x_j) + (1 - f_1) \; p_0^{(j)}(x_j)}$$

Naive Bayes Classifier

Likelihood classifier

Assuming the absence of correlations between the observables, treating different observables as independent random variables, multi-deminsional pdf can be calculated as a product of variable pdfs.

Likelihood of hypothesis k for measured event $\mathbf{x}$ is then given by

$$L_k(\mathbf{x}) \; = \; L(H_k|\mathbf{x}) \;\; = \;\; \prod_j P(H_k|x_j)$$

Naive Bayes Classifier

Likelihood classifier

Assuming the absence of correlations between the observables, treating different observables as independent random variables, multi-deminsional pdf can be calculated as a product of variable pdfs.

Likelihood of hypothesis k for measured event $\mathbf{x}$ is then given by

$$L_k(\mathbf{x}) = L(H_k|\mathbf{x}) = \prod_j P(H_k|x_j)$$

We can then construct the classifier based on the likelihood ratio:

$$\gamma(\mathbf{x}) = \frac{L_1(\mathbf{x})}{L_0(\mathbf{x}) + L_1(\mathbf{x})}$$

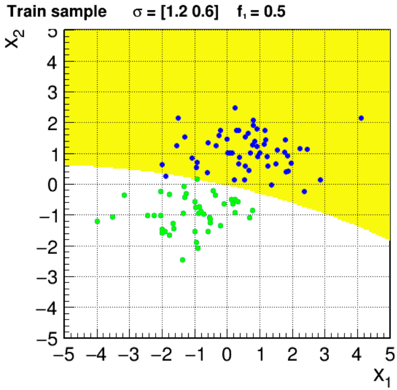
which should be equivalent to the Neyman-Pearson classifier.

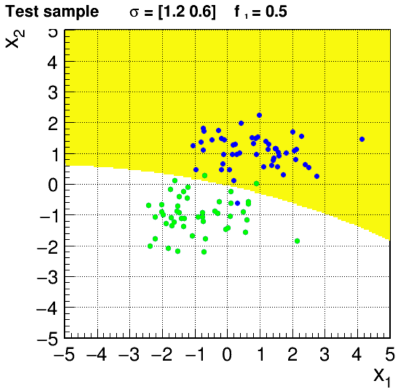
Assuming correlations can be neglected and in the limit of large training samples.

Naive Bayes Classifier

Example separation of 2D Gaussian distributions

We first use the train sample of events to reconstruct individual $p_k^{(j)}$.
Training results can be then applied to the independent test sample...

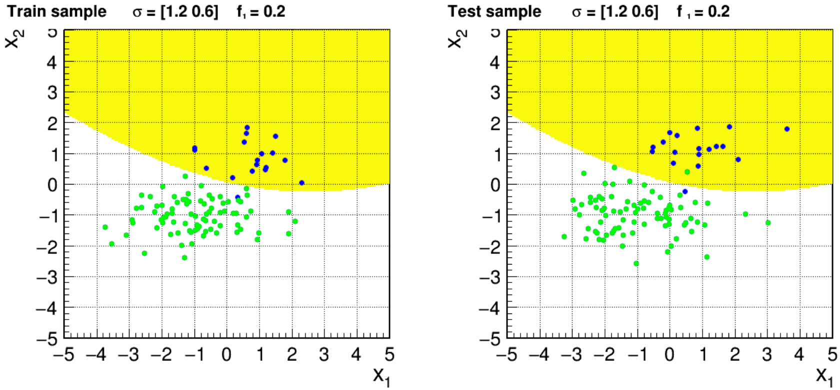

Train sample σ = [1.2 0.6] f_1 = 0.5


Test sample σ = [1.2 0.6] f_1 = 0.5

Implementation of the Gaussian Naive Bayes Classifier in **sklearn**.

Example separation of 2D Gaussian distributions

We first use the train sample of events to reconstruct individual $p_k^{(j)}$.
Training results can be then applied to the independent test sample...

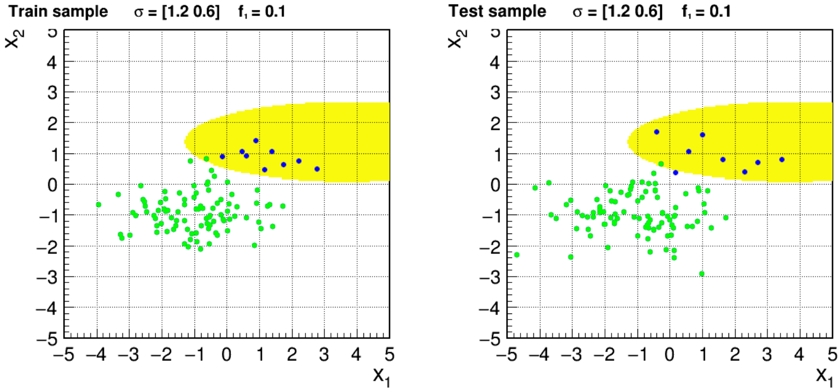


Implementation of the Gaussian Naive Bayes Classifier in **sklearn**.

Naive Bayes Classifier

Example separation of 2D Gaussian distributions

We first use the train sample of events to reconstruct individual $p_k^{(j)}$. Training results can be then applied to the independent test sample...

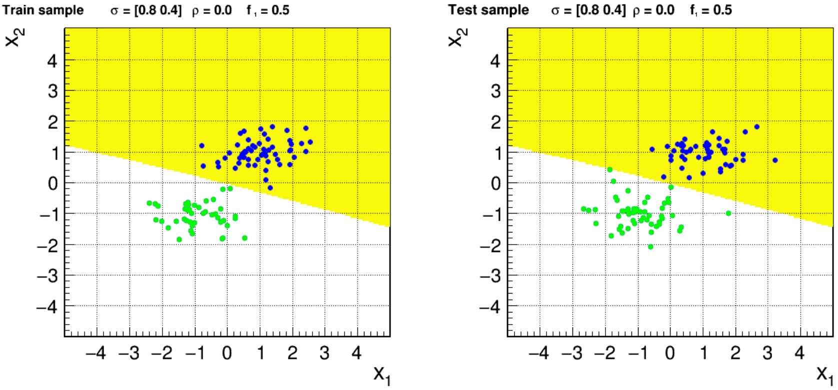


Implementation of the Gaussian Naive Bayes Classifier in **sklearn**.

Naive Bayes Classifier

Example

Efficient classification can be obtained for uncorrelated variables.



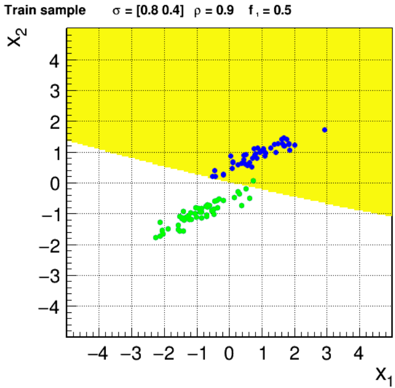
Implementation of the Gaussian Naive Bayes Classifier in **sklearn**.

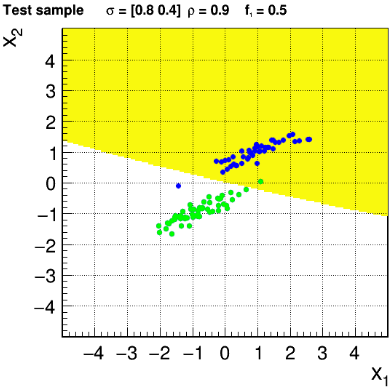
Naive Bayes Classifier

Example

Efficient classification can be obtained for uncorrelated variables.

However, it is clearly far from optimal in case of correlations!





Implementation of the Gaussian Naive Bayes Classifier in **sklearn**.

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Classifier based on the linear combination of input variables:

$$F(\mathbf{x};\mathbf{w}) \;=\; w_0 + \sum_{j=1}^{N} w_j\, x_j \;=\; w_0 + \mathbf{w}\cdot\mathbf{x}$$

Resulting decision boundaries, $F(\mathbf{x}) = F_{cut}$, are hyperplanes in N dim.

How to find vector **w** giving best separation between two classes of events?

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Classifier based on the linear combination of input variables:

$$F(\mathbf{x};\mathbf{w}) \;=\; w_0 + \sum_{j=1}^{N} w_j\, x_j \;=\; w_0 + \mathbf{w}\cdot\mathbf{x}$$

Resulting decision boundaries, $F(\mathbf{x}) = F_{cut}$, are hyperplanes in N dim.

How to find vector **w** giving best separation between two classes of events?

For the general case, to use numerical optimization procedure, one needs to define the "loss function". Possible choice: (similar to χ^2)

$$L(\mathbf{w}) \;=\; \sum_{\text{events } i} \left[t^{(i)} - y(F(\mathbf{x}^{(i)};\mathbf{w})) \right]^2$$

where y is the decision ("activation") function (eg. step function or tanh), $t^{(i)}$ is true class of training event $\mathbf{x}^{(i)}$ (-1 for H_0 and $+1$ for H_1).

Iterative procedure can be applied to minimize $L(\mathbf{w})$.

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Weight vector **w** defines the direction, on which all events are projected.
Projection “reduces” the N variable problem to single variable $F(\mathbf{x})$.

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Weight vector **w** defines the direction, on which all events are projected. Projection "reduces" the N variable problem to single variable $F(\mathbf{x})$.

If we assume Gaussian variable distributions, we can look at the direction which maximizes the relative distance between the two hypothesis in F:

$$D(\mathbf{w}) \;=\; \frac{(h_1 - h_0)^2}{\sigma_1^2 + \sigma_0^2}$$

h_k and σ_k^2 are the expected values and variances of $F(\mathbf{x})$ for hypothesis k:

$$h_k \;=\; \mathbb{E}(F(\mathbf{x})|H_k) \qquad \text{and} \qquad \sigma_k^2 \;=\; \mathbb{V}(F(\mathbf{x})|H_k)$$

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Weight vector **w** defines the direction, on which all events are projected. Projection "reduces" the N variable problem to single variable $F(\mathbf{x})$.

If we assume Gaussian variable distributions, we can look at the direction which maximizes the relative distance between the two hypothesis in F:

$$D(\mathbf{w}) \;=\; \frac{(h_1 - h_0)^2}{\sigma_1^2 + \sigma_0^2}$$

h_k and σ_k^2 are the expected values and variances of $F(\mathbf{x})$ for hypothesis k:

$$h_k \;=\; \mathbb{E}(F(\mathbf{x})|H_k) \qquad \text{and} \qquad \sigma_k^2 \;=\; \mathbb{V}(F(\mathbf{x})|H_k)$$

How to relate h_k and σ_k^2 to the properties of the **x** distribution for H_k:

$$\boldsymbol{\mu}_k \;=\; \mathbb{E}(\mathbf{x}|H_k) \qquad \text{and} \qquad \mathbb{C}_{\mathbf{x}} \quad ?$$

Linear discriminant

(Behnke)

One can note that:

$$h_1 - h_0 \;=\; \mathbf{w}\cdot\boldsymbol{\mu}_1 \;-\; \mathbf{w}\cdot\boldsymbol{\mu}_0 \;=\; \mathbf{w}\cdot(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

and so the nominator of $D(\mathbf{w})$ can be written as:

$$(h_1 - h_0)^2 \;=\; \mathbf{w}^\mathsf{T}\,\mathbb{B}\,\mathbf{w} \qquad \text{where} \qquad \mathbb{B} \;=\; (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\mathsf{T}$$

is the so-called between-class matrix.

Fisher Linear Discriminant

Linear discriminant (Behnke)

One can note that:

$$h_1 - h_0 \;=\; \mathbf{w}\cdot\boldsymbol{\mu}_1 \;-\; \mathbf{w}\cdot\boldsymbol{\mu}_0 \;=\; \mathbf{w}\cdot(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

and so the nominator of $D(\mathbf{w})$ can be written as:

$$(h_1 - h_0)^2 \;=\; \mathbf{w}^{\mathsf{T}}\,\mathbb{B}\,\mathbf{w} \qquad \text{where} \qquad \mathbb{B} \;=\; (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathsf{T}}$$

is the so-called between-class matrix.

Introducing also the within-class matrix

$$\mathbb{W} \;=\; \mathbb{C}_{\mathbf{x}}^{(H_1)} + \mathbb{C}_{\mathbf{x}}^{(H_0)}$$

one can directly write *F* distance between two hypothesis in terms of **w**:

$$D(\mathbf{w}) \;=\; \frac{\mathbf{w}^{\mathsf{T}}\,\mathbb{B}\,\mathbf{w}}{\mathbf{w}^{\mathsf{T}}\,\mathbb{W}\,\mathbf{w}}$$

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Matrices $\mathbb{B}$ and $\mathbb{W}$ depend only on **x** pdfs for the two hypothesis.
They do not depend on the classifier weights **w**.

Maximum relative distance requirement corresponds to $\nabla D(\mathbf{w}) = 0$ condition. The solution is:

$$\mathbf{w} \;=\; a\ \mathbb{W}^{-1}\,(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

where a is an arbitrary scaling factor. w_0 is the second free parameter

Fisher Linear Discriminant

Linear discriminant

(Behnke)

Matrices $\mathbb{B}$ and $\mathbb{W}$ depend only on $\mathbf{x}$ pdfs for the two hypothesis.
They do not depend on the classifier weights $\mathbf{w}$.

Maximum relative distance requirement corresponds to $\nabla D(\mathbf{w}) = 0$ condition. The solution is:

$$\mathbf{w} \;=\; a\ \mathbb{W}^{-1}\,(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$

where a is an arbitrary scaling factor. w_0 is the second free parameter

When parameters of the pdfs are not known, they can be derived from the properties of the training sample:

$$\mathbf{w} \;=\; a\ \left(\hat{\mathbb{C}}_{\mathbf{x}}^{(H_1)} + \hat{\mathbb{C}}_{\mathbf{x}}^{(H_o)}\right)^{-1}\ \left(\bar{\mathbf{x}}^{(H_1)} - \bar{\mathbf{x}}^{(H_0)}\right)$$

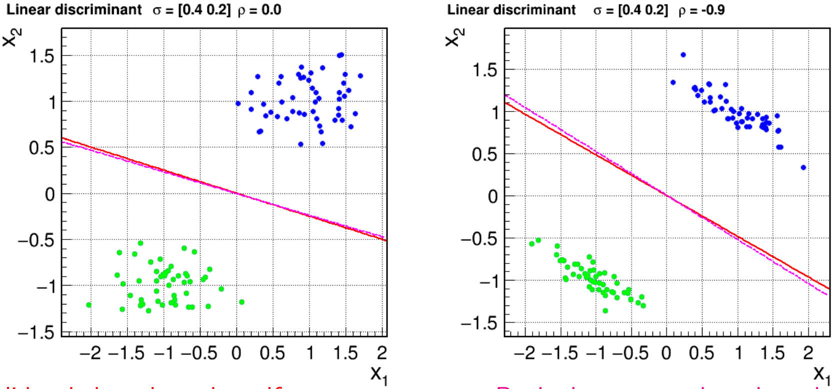
where $\bar{\mathbf{x}}$ and $\hat{\mathbb{C}}$ are the mean and covariance matrices for the training data

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for well separated data



Linear discriminant σ = [0.4 0.2] ρ = 0.0

Linear discriminant σ = [0.4 0.2] ρ = -0.9

Solid red: based on the pdf parameters Dashed magenta: based on data

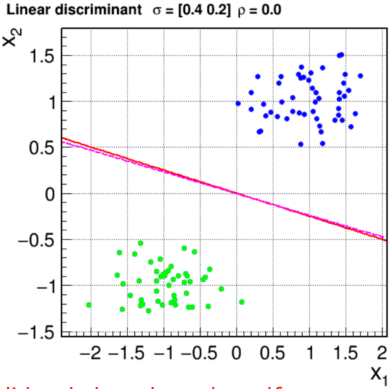
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

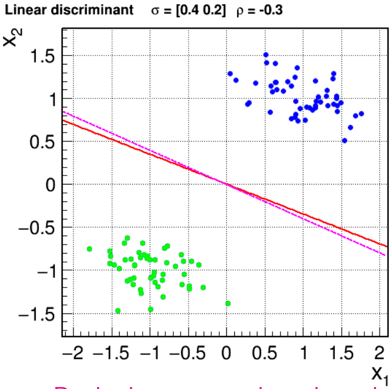
Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for well separated data





Solid red: based on the pdf parameters Dashed magenta: based on data

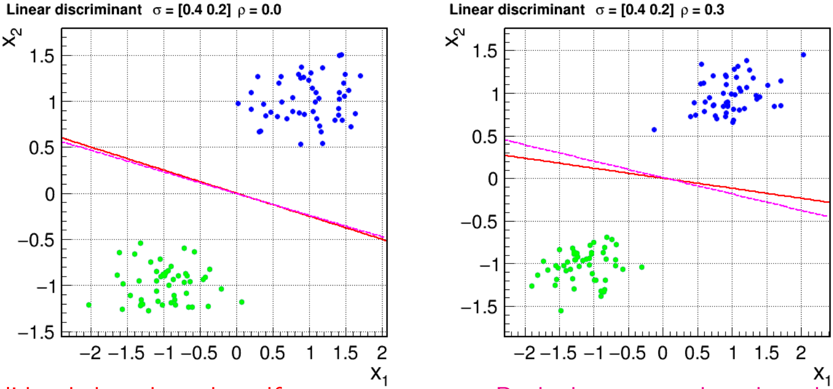
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for well separated data



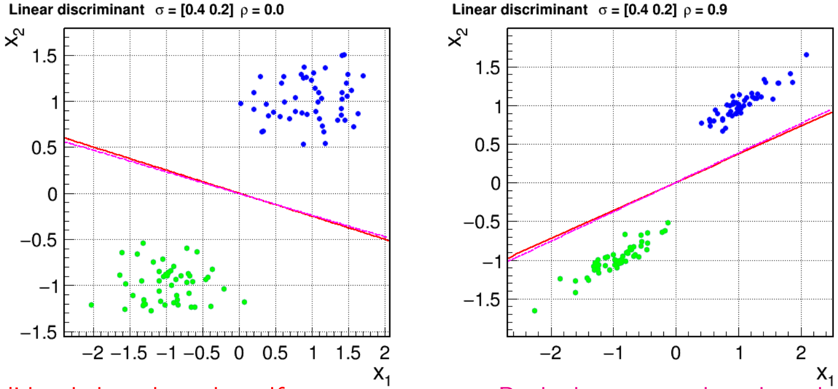
Solid red: based on the pdf parameters Dashed magenta: based on data

Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account
Different correlation coefficients ρ for well separated data



Solid red: based on the pdf parameters Dashed magenta: based on data

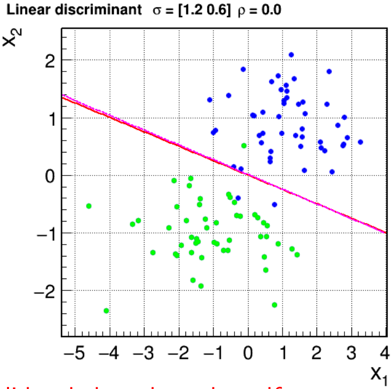
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

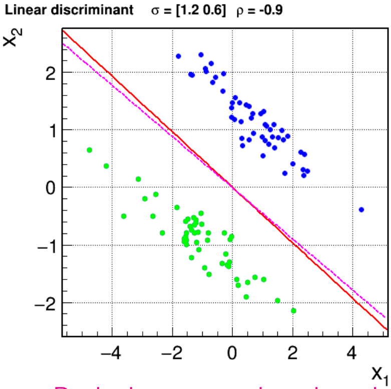
Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for less separated data





Solid red: based on the pdf parameters Dashed magenta: based on data

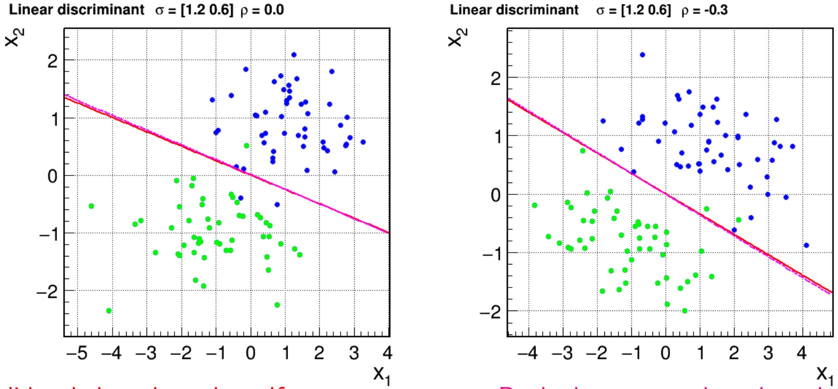
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for less separated data



Linear discriminant σ = [1.2 0.6] ρ = 0.0

Linear discriminant σ = [1.2 0.6] ρ = -0.3

Solid red: based on the pdf parameters Dashed magenta: based on data

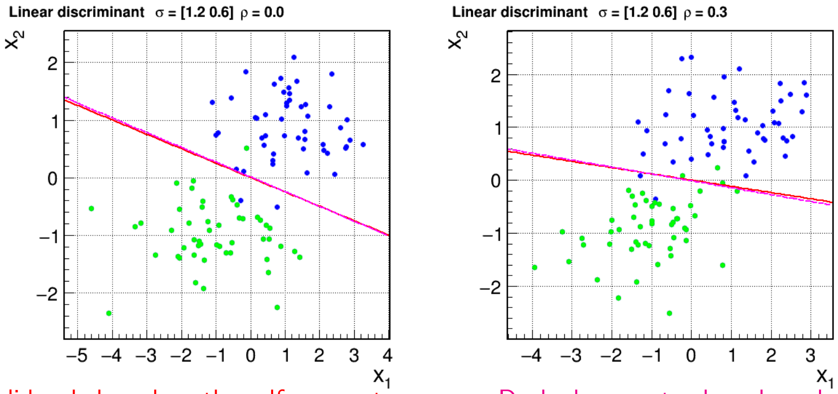
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for less separated data



Solid red: based on the pdf parameters Dashed magenta: based on data

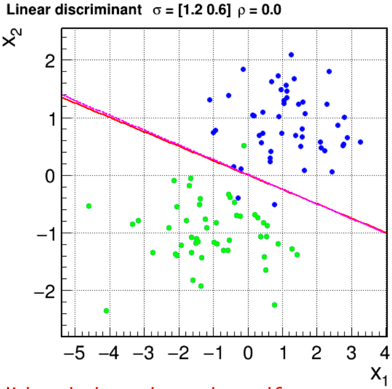
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

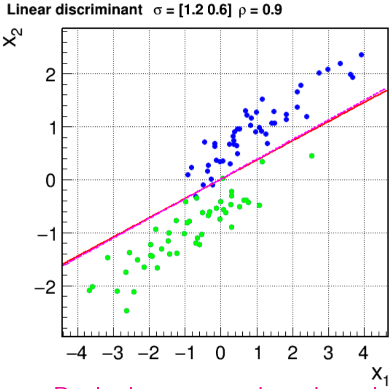
Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different correlation coefficients ρ for less separated data





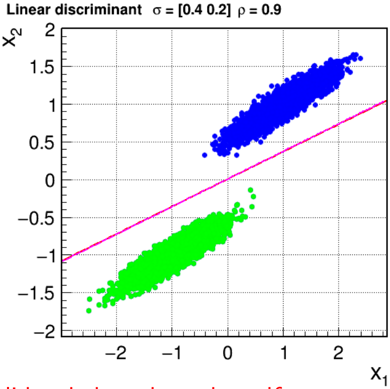
Solid red: based on the pdf parameters Dashed magenta: based on data

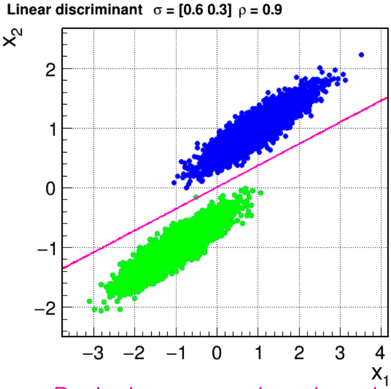
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account
Different relative separation for high correlation coefficient $\rho = 0.9$


Linear discriminant σ = [0.4 0.2] ρ = 0.9


Linear discriminant σ = [0.6 0.3] ρ = 0.9

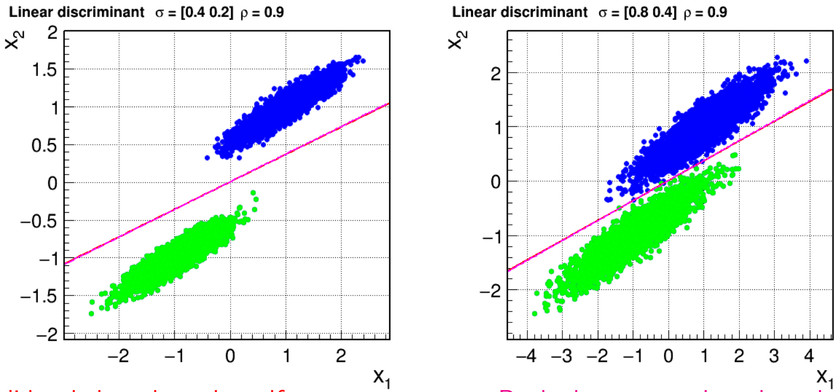
Solid red: based on the pdf parameters Dashed magenta: based on data
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different relative separation for high correlation coefficient $\rho = 0.9$



Solid red: based on the pdf parameters Dashed magenta: based on data

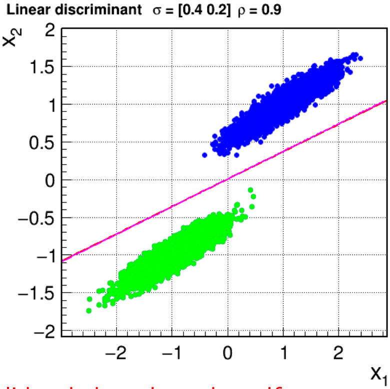
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

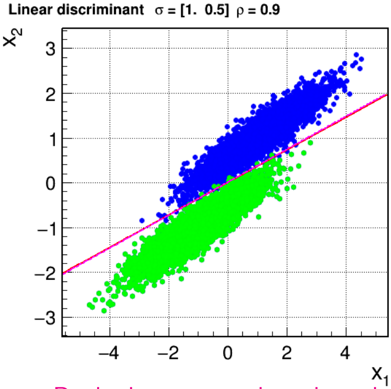
Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different relative separation for high correlation coefficient $\rho = 0.9$





Solid red: based on the pdf parameters Dashed magenta: based on data

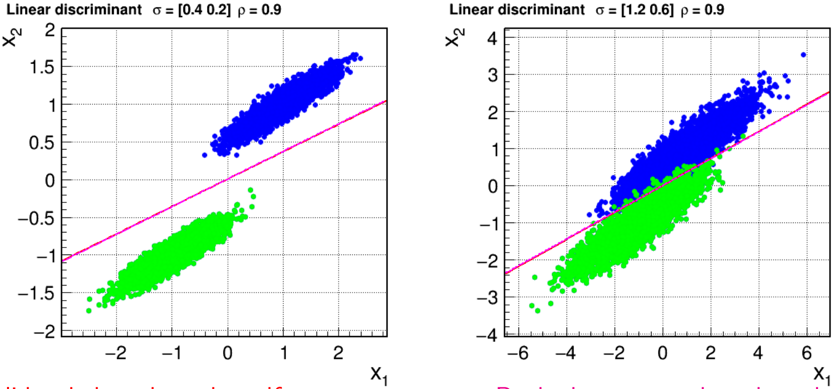
Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Fisher Linear Discriminant

Example separation of 2D Gaussian distributions

Fisher discriminant takes variable correlations properly into account

Different relative separation for high correlation coefficient $\rho = 0.9$



Solid red: based on the pdf parameters Dashed magenta: based on data

Data driven analysis: implementation of the Linear Discriminant Analysis in **sklearn**.

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Nearest neighbors classifier

Principle

This classifier refers directly to the Neymann and Pearson Lemma.

It is based on the expectation, that the likelihood ratio can be related to the ratio of the expected event densities:

$$Q(\mathbf{x}) \;=\; \frac{L(\mathbf{x}|H_1)}{L(\mathbf{x}|H_0)} \;=\; \frac{1}{N_1}\frac{dN_1}{d\mathbf{x}} \left(\frac{1}{N_0}\frac{dN_0}{d\mathbf{x}}\right)^{-1} \;=\; \frac{N_0}{N_1}\,\frac{dN_1}{dN_0}$$

where dN_k represent the expected number of events for hypothesis k, in a small variable space volume $d\mathbf{x}$ (in the limit $d\mathbf{x} \rightarrow 0$, $N_k \rightarrow \infty$)

Nearest neighbors classifier

Principle

This classifier refers directly to the Neymann and Pearson Lemma.

It is based on the expectation, that the likelihood ratio can be related to the ratio of the expected event densities:

$$Q(\mathbf{x}) \;=\; \frac{L(\mathbf{x}|H_1)}{L(\mathbf{x}|H_0)} = \frac{1}{N_1}\frac{dN_1}{d\mathbf{x}} \left(\frac{1}{N_0}\frac{dN_0}{d\mathbf{x}}\right)^{-1} = \frac{N_0}{N_1}\,\frac{dN_1}{dN_0}$$

where dN_k represent the expected number of events for hypothesis k, in a small variable space volume $d\mathbf{x}$ (in the limit $d\mathbf{x} \to 0$, $N_k \to \infty$)

The idea is to replace the expected event densities dN_k by numbers of events in the actual data (training sample including H_0 and H_1 events):

$$dN_k(\mathbf{x}) \;\to\; n_k(\mathbf{x}) = \sum_{\mathbf{x}' \in \Delta(\mathbf{x})} \mathbf{x}' \subset H_k$$

The key point is how to define "neighborhood" region $\Delta(\mathbf{x})$ of point $\mathbf{x}$

Nearest neighbors classifier

Principle

Two possible approaches are commonly used.

One can define $\Delta(\mathbf{x})$ by specifying maximum distance d between points:

$$\Delta(\mathbf{x}) \;=\; \{x' \;:\; d(\mathbf{x}', \mathbf{x}) < R_{max}\}$$

However, R_{max} has to be sufficiently large to always accept a sample of training events, also in the regions of lowest probability density...
That is why this approach is not very efficient...

Nearest neighbors classifier

Principle

Two possible approaches are commonly used.

One can define $\Delta(\mathbf{x})$ by specifying maximum distance d between points:

$$\Delta(\mathbf{x}) \;=\; \{x' \;:\; d(\mathbf{x}', \mathbf{x}) < R_{max}\}$$

However, R_{max} has to be sufficiently large to always accept a sample of training events, also in the regions of lowest probability density...
That is why this approach is not very efficient...

This problem is "solved" in the "k nearest neighbors" (kNN) classification. We sort training events **x**' according to their distance from the test point **x** and take the closest k points:

$$\Delta(\mathbf{x}) \;=\; \{x' \;:\; d(\mathbf{x}', \mathbf{x}) < R(\mathbf{x})\} \quad \text{and} \quad R(\mathbf{x}) \;:\; \sum_{\mathbf{x}'} 1 = k$$

Nearest neighbors classifier

Principle

Two possible approaches are commonly used.

One can define $\Delta(\mathbf{x})$ by specifying maximum distance d between points:

$$\Delta(\mathbf{x}) \ = \ \{x' \ : \ d(\mathbf{x}', \mathbf{x}) < R_{max}\}$$

However, R_{max} has to be sufficiently large to always accept a sample of training events, also in the regions of lowest probability density...
That is why this approach is not very efficient...

This problem is "solved" in the "k nearest neighbors" (kNN) classification.
We sort training events **x**' according to their distance from the test point **x** and take the closest k points:

$$\Delta(\mathbf{x}) \ = \ \{x' \ : \ d(\mathbf{x}', \mathbf{x}) < R(\mathbf{x})\} \quad \text{and} \quad R(\mathbf{x}) \ : \ \sum_{\mathbf{x}'} 1 = k$$

Where we still need to define the distance measure $d(\mathbf{x}', \mathbf{x})$...

Nearest neighbors classifier

k Nearest Neighbors

We can then define our classifier as

$$\gamma(\mathbf{x}) \;=\; \frac{n_1(\mathbf{x})}{n_0(\mathbf{x}) + n_1(\mathbf{x})} = \frac{n_1(\mathbf{x})}{k}$$

where $\gamma_{cut} = 0.5$ is assumed in most cases...

Nearest neighbors classifier

k Nearest Neighbors

We can then define our classifier as

$$\gamma(\mathbf{x}) \;=\; \frac{n_1(\mathbf{x})}{n_0(\mathbf{x}) + n_1(\mathbf{x})} = \frac{n_1(\mathbf{x})}{k}$$

where $\gamma_{cut} = 0.5$ is assumed in most cases...

For the distance measure, standard Euclidean metric is usually assumed:

$$d(\mathbf{x}', \mathbf{x}) \;=\; |\mathbf{x}' - \mathbf{x}| = \left(\sum_j (x'_j - x_j)^2 \right)^{\frac{1}{2}}$$

which, however, neglects differences in value scales of different variables and possible correlations between them

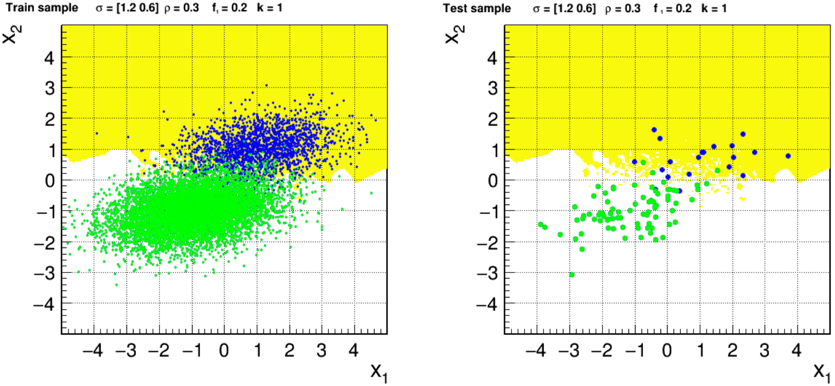
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Works already for $k = 1$: decision based on the nearest train event.

Very large fluctuations, reflecting fluctuations in the training sample...



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

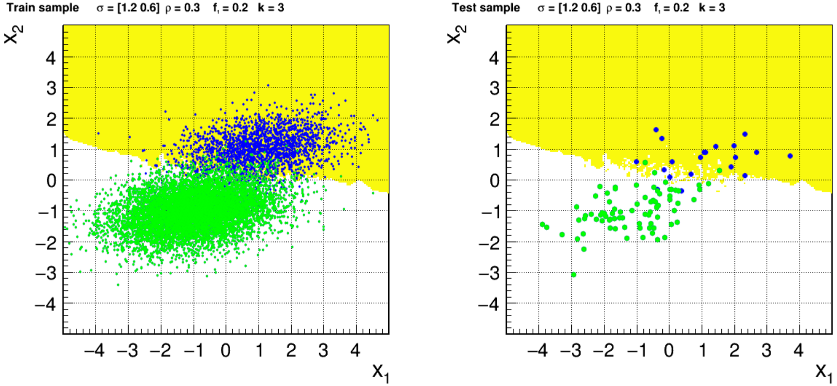
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Classification results more stable for larger k.

Impact of fluctuations still visible...



Implementation of the k Nearest Neighbors classifier in **sklearn**.

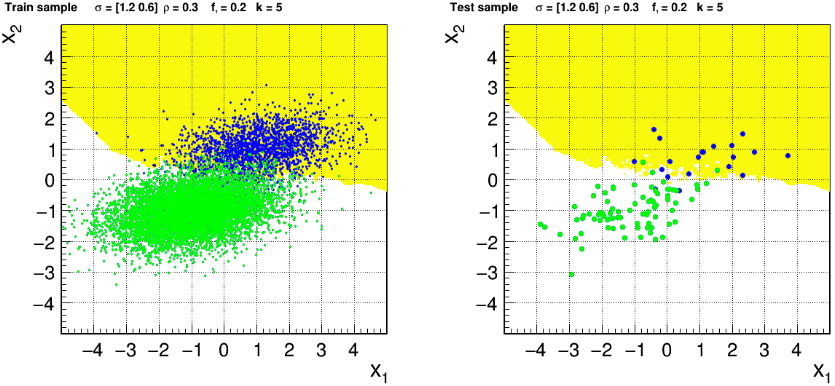
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Classification results more stable for larger *k*.

Impact of fluctuations still visible...



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

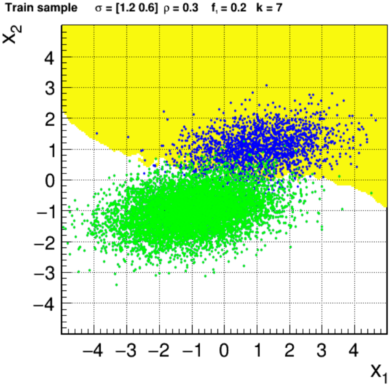
Nearest neighbors classifier

*k*NN example

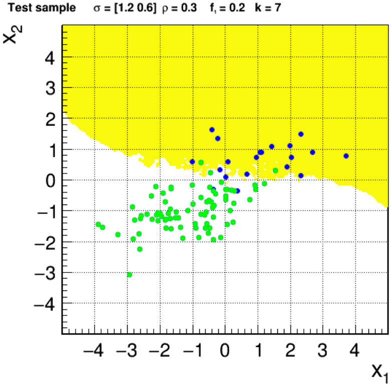
separation of 2D Gaussian distributions

Classification results more stable for larger k.

Impact of fluctuations still visible...



Train sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 7



Test sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 7

Implementation of the k Nearest Neighbors classifier in **sklearn**.

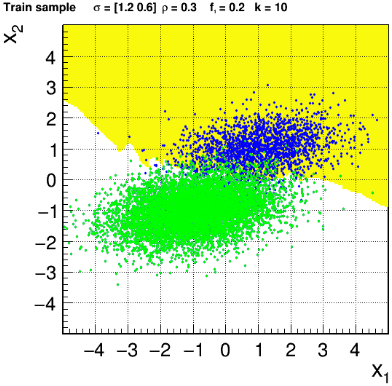
Nearest neighbors classifier

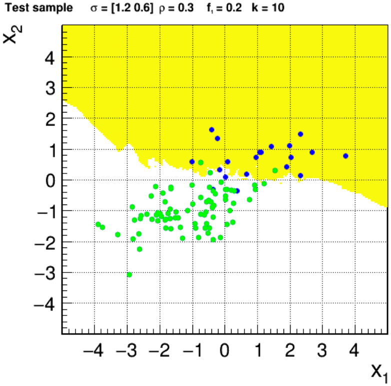
*k*NN example

separation of 2D Gaussian distributions

Classification results more stable for larger *k*.

Impact of fluctuations still visible...


Train sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 10


Test sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 10

Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

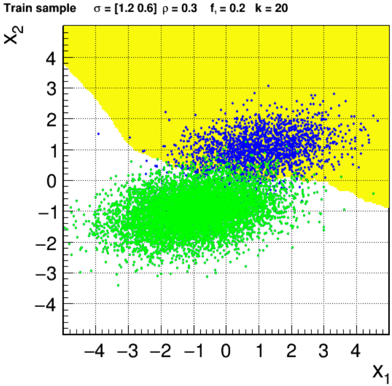
Nearest neighbors classifier

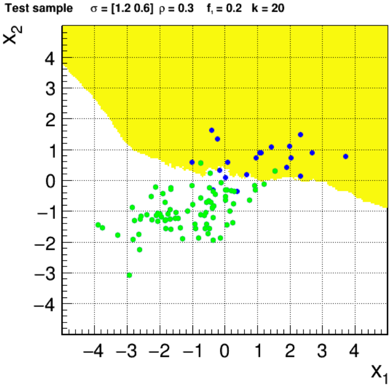
*k*NN example

separation of 2D Gaussian distributions

Smooth classification boundary for $k > 10$, but details can be lost...

Classification seems not to be optimal? (linear discriminator)





Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

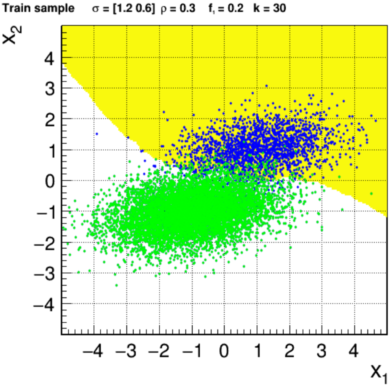
Nearest neighbors classifier

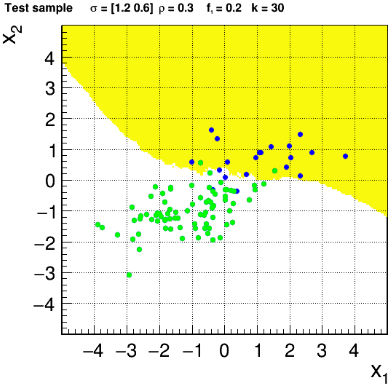
*k*NN example

separation of 2D Gaussian distributions

Smooth classification boundary for $k > 10$, but details can be lost...

Classification seems not to be optimal? (linear discriminator)





Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

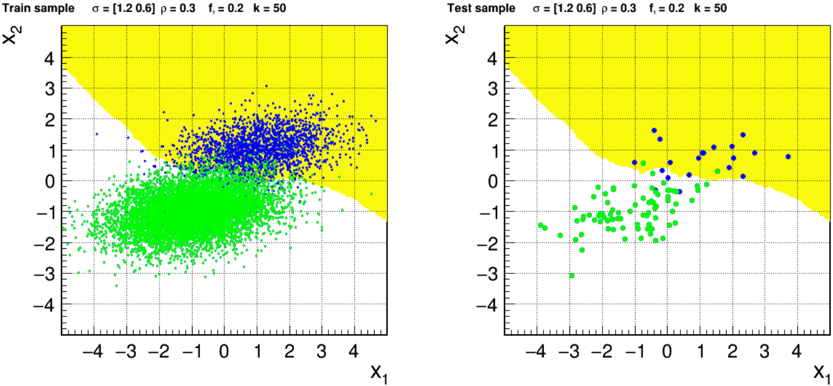
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Smooth classification boundary for $k > 10$, but details can be lost...

Classification seems not to be optimal? (linear discriminator)



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

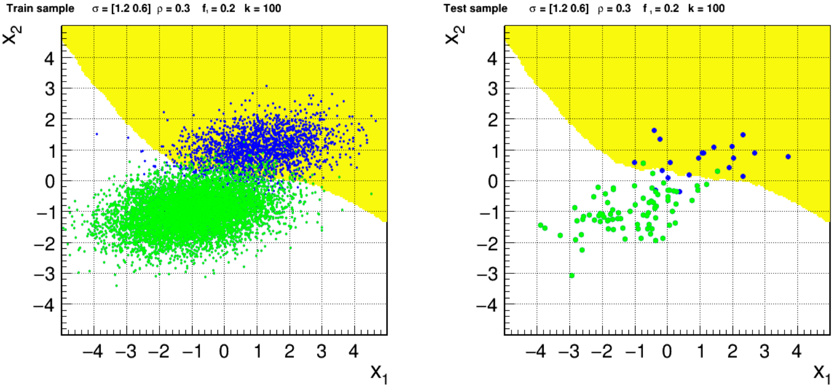
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Smooth classification boundary for $k > 10$, but details can be lost...

Classification seems not to be optimal? (linear discriminator)



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

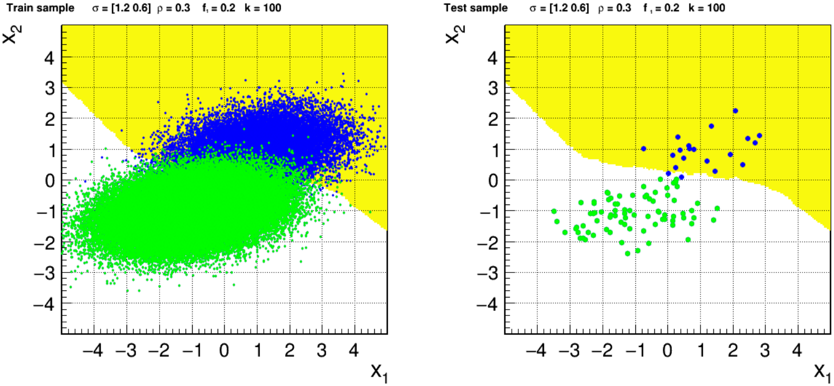
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Precision of classification improves with size of training sample

But it also gets more and more time consuming...



Train sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 100

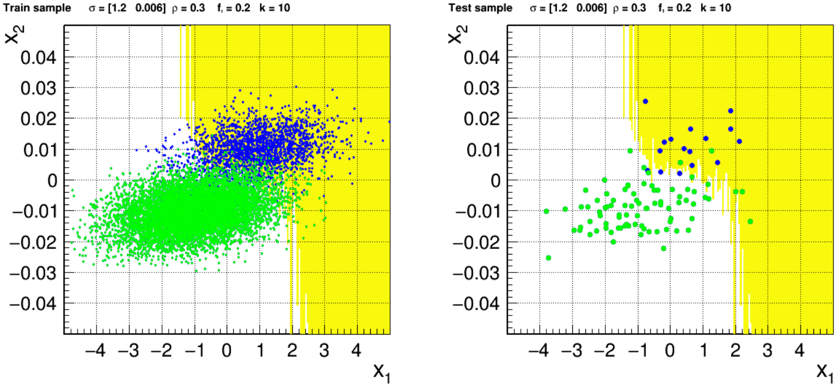
Test sample σ = [1.2 0.6] ρ = 0.3 f_1 = 0.2 k = 100

Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

*k*NN example

separation of 2D Gaussian distributions

Default procedure, assuming Euclidean metric, clearly fails when scales of the considered observables are very different...



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

Nearest neighbors classifier

k Nearest Neighbors

To solve the problem of different variable scales, one could redefine considered set of variables to span the same numerical range or to have same variances. This however will still neglect possible correlations.

Nearest neighbors classifier

k Nearest Neighbors

To solve the problem of different variable scales, one could redefine considered set of variables to span the same numerical range or to have same variances. This however will still neglect possible correlations.

In the general case, distance measure properly reflecting properties of the data set should be used. Frequent choice:

$$d^2(\mathbf{x}', \mathbf{x}) \;=\; (\mathbf{x}' - \mathbf{x})^{\intercal}\, \mathbb{C}_{\mathbf{x}}^{-1}\, (\mathbf{x}' - \mathbf{x}) \;=\; \sum_{jk} (x'_j - x_j)\, (\mathbb{C}_{\mathbf{x}}^{-1})_{jk}\, (x'_k - x_k)$$

where $\mathbb{C}_{\mathbf{x}}$ is the covariance matrix of the measurement.

This is so-called "Mahalanobis distance" measure.

Similar to the calculation of the χ^2 value between two points

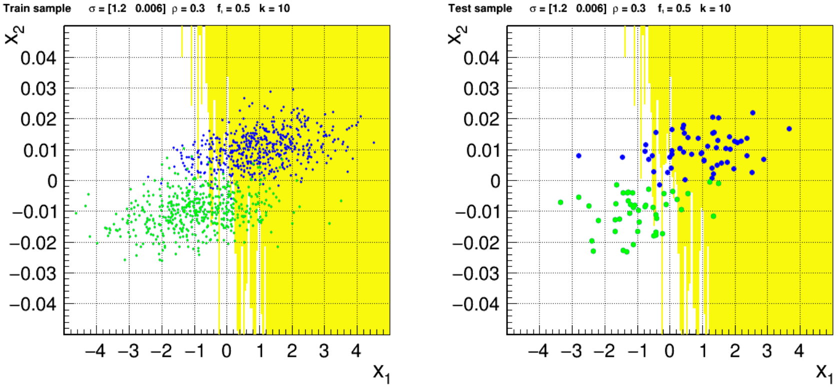
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Two-dimensional data set with large scale difference between variables

With default distance measure (Euclidean metric)



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

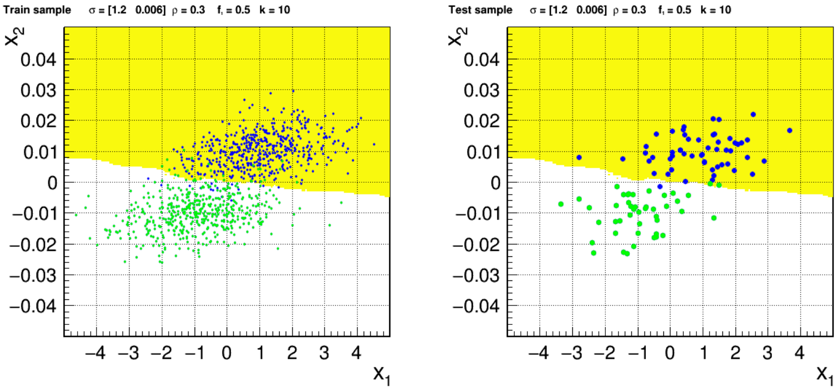
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Two-dimensional data set with large scale difference between variables

With Mahalanobis distance measure (including correlations)



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

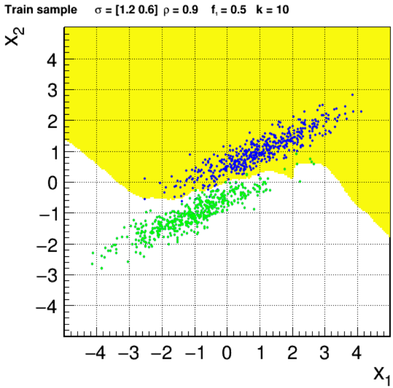
Nearest neighbors classifier

*k*NN example

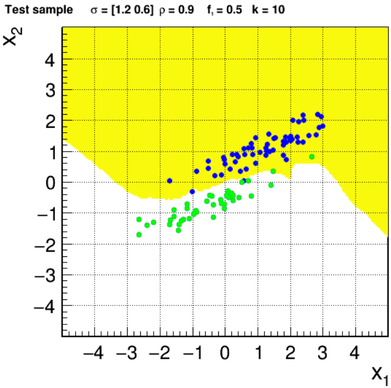
separation of 2D Gaussian distributions

Two-dimensional data set with large correlation between variables

With default distance measure (Euclidean metric)



Train sample σ = [1.2 0.6] ρ = 0.9 f_1 = 0.5 k = 10



Test sample σ = [1.2 0.6] ρ = 0.9 f_1 = 0.5 k = 10

Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

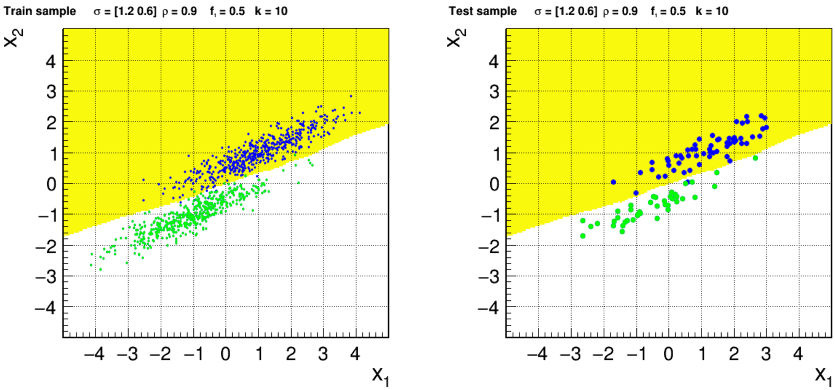
Nearest neighbors classifier

*k*NN example

separation of 2D Gaussian distributions

Two-dimensional data set with large correlation between variables

With Mahalanobis distance measure (including correlations)



Implementation of the *k* Nearest Neighbors classifier in **sklearn**.

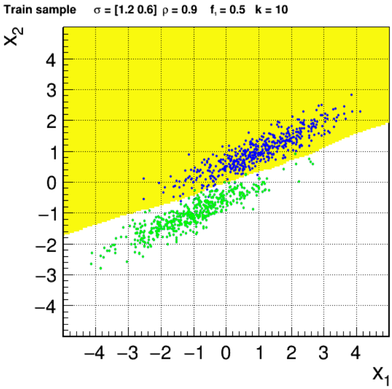
Nearest neighbors classifier

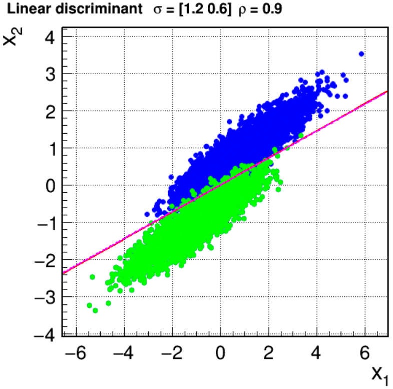
*k*NN example

separation of 2D Gaussian distributions

Two-dimensional data set with large correlation between variables

With Mahalanobis distance measure (including correlations)



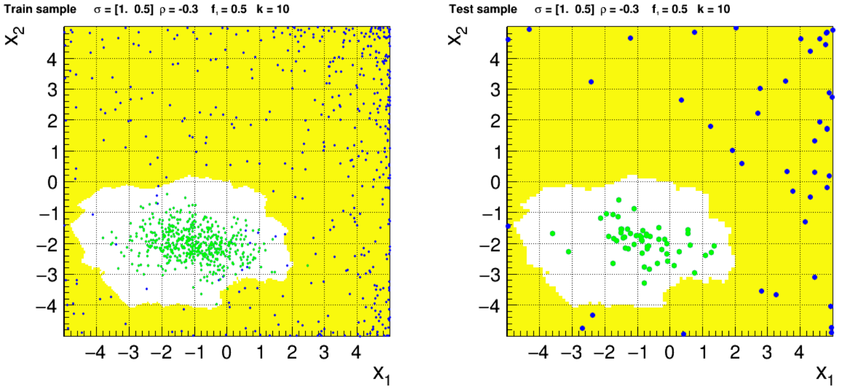


Results consistent with those obtained with Fisher linear discriminant.

Nearest neighbors classifier

k Nearest Neighbors

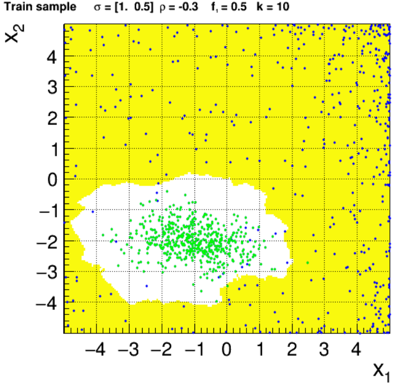
While Naive Bayes and Fisher Linear Classifiers are based on modeling the likelihood distribution, nearest neighbors classifier is very general, can be used in (almost) any case. Example application:



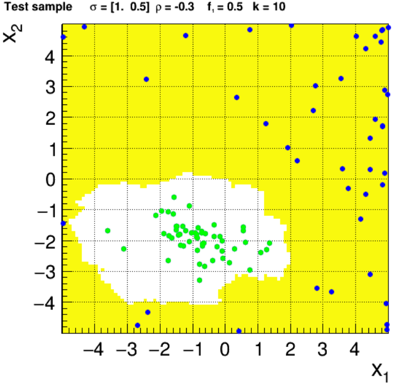
Nearest neighbors classifier

k Nearest Neighbors

While Naive Bayes and Fisher Linear Classifiers are based on modeling the likelihood distribution, nearest neighbors classifier is very general, can be used in (almost) any case. Example application:



Train sample σ = [1. 0.5] ρ = -0.3 f_1 = 0.5 k = 10



Test sample σ = [1. 0.5] ρ = -0.3 f_1 = 0.5 k = 10

Unfortunately, it is also very slow and requires large training samples...

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Iterative procedure

Linear discriminant

In the standard approach, weights optimizing classifier based on the linear combination of input variables:

$$F(\mathbf{x};\mathbf{w}) \;=\; w_0 + \sum_{j=1}^{N} w_j\, x_j \;=\; w_0 + \mathbf{w}\cdot\mathbf{x}$$

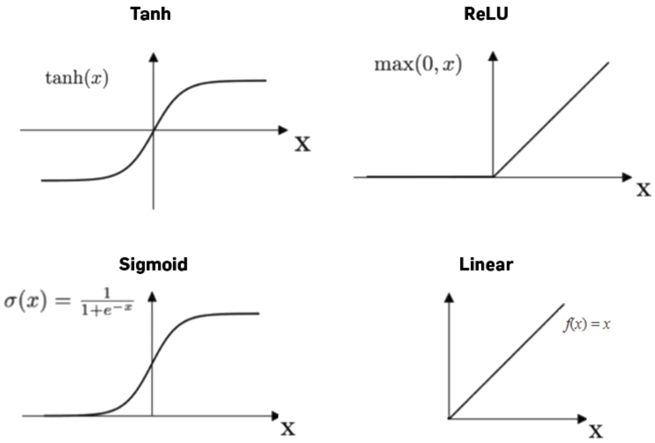
are found based on the properties of the considered (training) samples.

However, the problem can be also solved without looking at the global properties, by minimizing the "loss function". Possible choice, "distance":

$$L(\mathbf{w}) \;=\; \sum_{\text{events } i} \left[t^{(i)} - y(F(\mathbf{x}^{(i)};\mathbf{w})) \right]^2$$

where y is the "activation function", $t^{(i)}$ is true class of event $\mathbf{x}^{(i)}$.
Many choices are possible for the activation function...

Activation function



Source: Artificial Intelligence Wiki

Iterative procedure

Perceptron Learning

Let activation function be defined in such a way that:

$$\lim_{x\to-\infty} y(x) \;=\; t(H_0) \qquad \text{and} \qquad \lim_{x\to+\infty} y(x) \;=\; t(H_1)$$

so that for events far from division boundary $(y = 0)$ we have:

$$y^{(i)} \;\equiv\; y\left(F(\mathbf{x}^{(i)};\mathbf{w})\right) \;\approx\; t^{(i)}$$

These events hardly contribute to the loss function.

Iterative procedure

Perceptron Learning

Let activation function be defined in such a way that:

$$\lim_{x\to-\infty} y(x) \; = \; t(H_0) \qquad \text{and} \qquad \lim_{x\to+\infty} y(x) \; = \; t(H_1)$$

so that for events far from division boundary $(y = 0)$ we have:

$$y^{(i)} \; \equiv \; y\left(F(\mathbf{x}^{(i)}; \mathbf{w})\right) \;\; \approx \;\; t^{(i)}$$

These events hardly contribute to the loss function.

Events which are incorrectly classified contribute most to loss function.

We can notice that the sign of $y^{(i)} - t^{(i)}$ indicates the reason:

- if $y^{(i)} - t^{(i)} > 0 \Rightarrow$ weights used were too large
- if $y^{(i)} - t^{(i)} < 0 \Rightarrow$ weights used were too small

Iterative procedure

Perceptron Learning

One can consider the iterative procedure of adjusting the weights:

$$\mathbf{w}^{(n+1)} \quad = \quad \mathbf{w}^{(n)} + \eta \sum_{i} \left(t^{(i)} - c^{(i)} \right) \cdot \mathbf{x}^{(i)}$$

where η is the learning rate parameter.

This approach was first proposed by M. Rosenblatt in 1958.

Weight correction can be applied on event by event basis (starting from the beginning when event loop completed) or calculating global correction for the whole sample.

Iterative procedure

Perceptron Learning

One can consider the iterative procedure of adjusting the weights:

$$\mathbf{w}^{(n+1)} \;=\; \mathbf{w}^{(n)} + \eta \sum_i \left(t^{(i)} - c^{(i)} \right) \cdot \mathbf{x}^{(i)}$$

where η is the learning rate parameter.

This approach was first proposed by M. Rosenblatt in 1958.

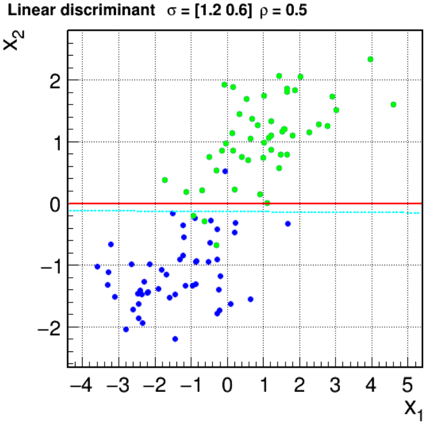
Weight correction can be applied on event by event basis (starting from the beginning when event loop completed) or calculating global correction for the whole sample.

Surprisingly, with proper choice of η this procedure works, results in classification optimization, even without referring to the loss function...

Perceptron Learning example

Example results for linear discriminant, starting from random weights:

N=100

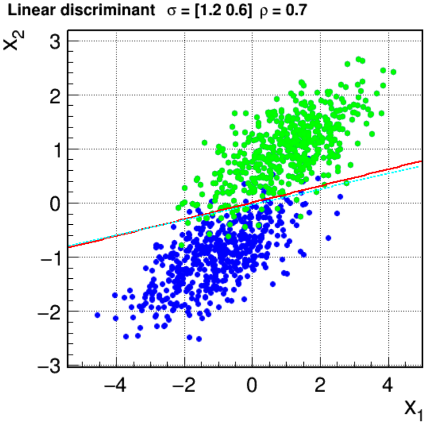


Iterative procedure (dashed cyan) compared with Fisher discriminant (solid red)

Perceptron Learning example

Example results for linear discriminant, starting from random weights:

N=1000



Iterative procedure (dashed cyan) compared with Fisher discriminant (solid red)

Classification

1. Event classification
2. Naive Bayes Classifier
3. Fisher Linear Discriminant
4. Nearest neighbors
5. Iterative procedure
6. Homework

Homework

Solutions to be uploaded by January 26.

You ordered 1000 silicon sensors, 36×24 mm^2, in the factory. The order was processed on two machines (50% each) and it turns out that:

- all sensors from the first machine are working OK.
  They also have proper size tolerance $\sigma_1 = 0.05$ mm
- the second machine was faulty, produced only faulty sensors
  and with worse size tolerance $\sigma_2 = 0.25$ mm

Unfortunately, the two samples were mixed. When collecting your order, you can make selection based on the measured dimensions of the sensor and pay only for the sensors you select...

- find the optimal selection approach
- calculate the corresponding ROC curve
- what is the optimal selection cut, if you pay 10$ for each collected sensor, and you can sell good ones for 12$...

assume no correlation between the two dimensions