Machine learning methods applied to the analysis of central exclusive production events in ALICE

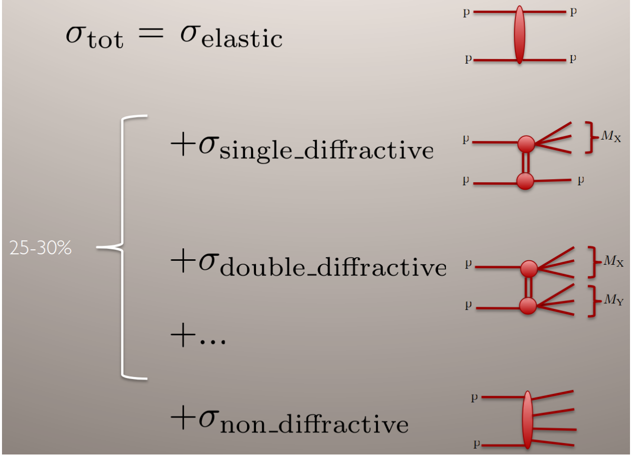
Sebastian Ratzenböck[1]

[1]Stefan Meyer Institut
Österreichische Akademie der Wissenschaften

26. April 2018

Outline

Introduction

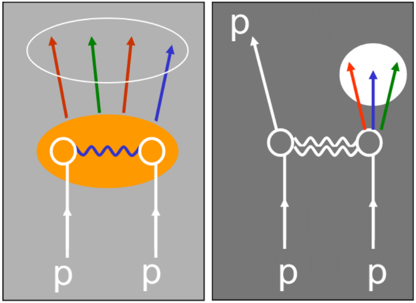


Cross section contribution at LHC energies

Diffraction definition

Diffractive events are reactions where **no quantum numbers are exchanged** $\rightarrow$ leads to special topology

Incident hadrons acquire color
$\rightarrow$ break apart



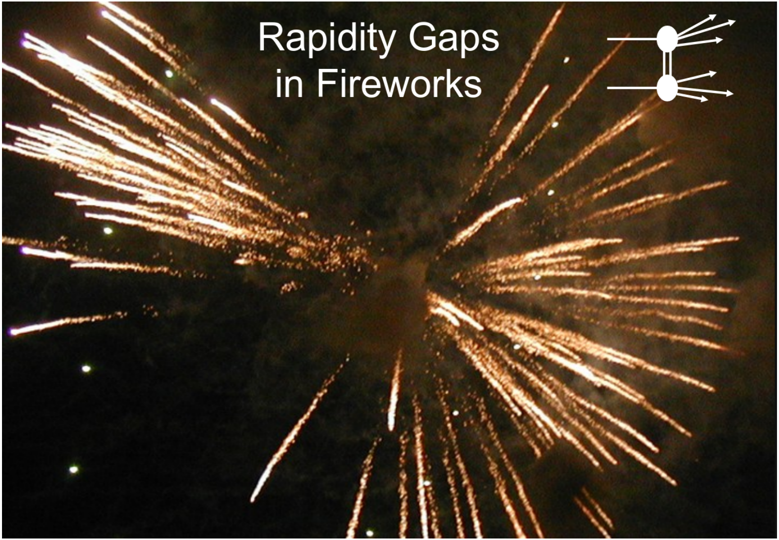
Colorless exchange

$$\eta = -ln(tan(\frac{\theta}{2}))$$

η gap exponentially suppressed

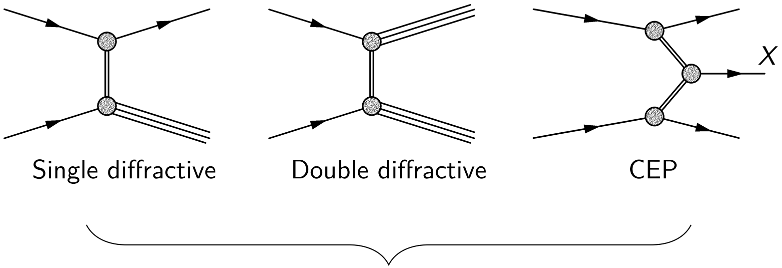
η gap not suppressed

Rapidity gap



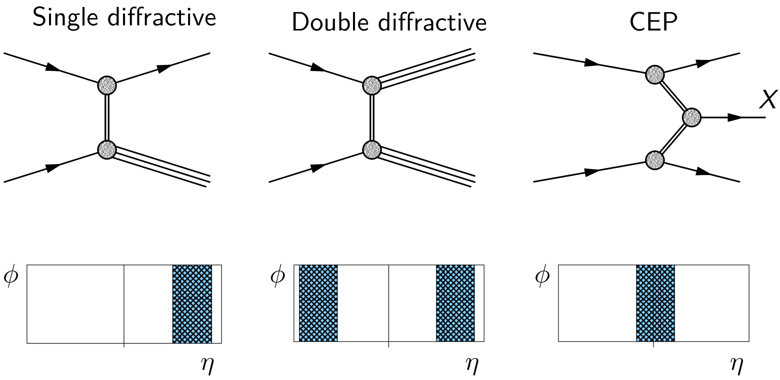
Diffractive processes

Different types of diffractive events are distinguished

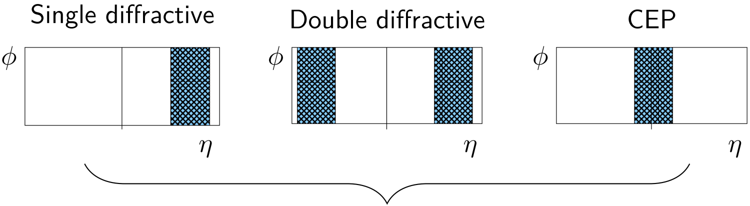


Described by *Regge* theory

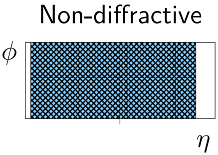
Event topology



Event topology

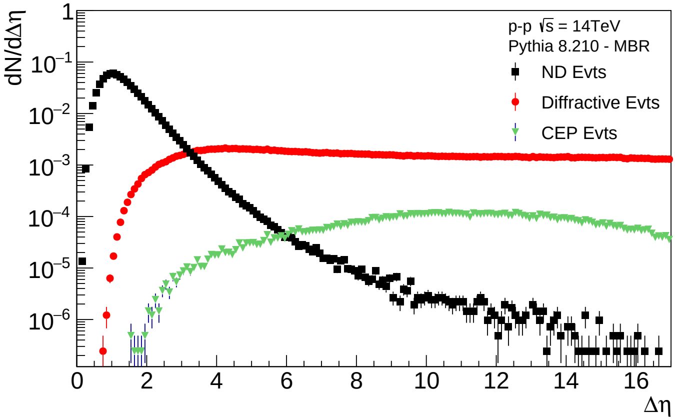


Large *eta* gap compared to non-diffractive event



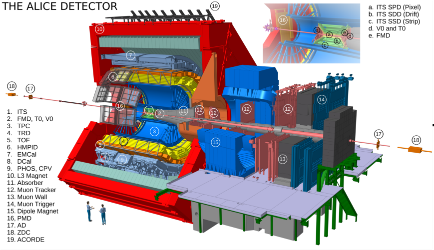
Event selection

Large rapidity gaps in *non-diffractive* events are exponentially suppressed



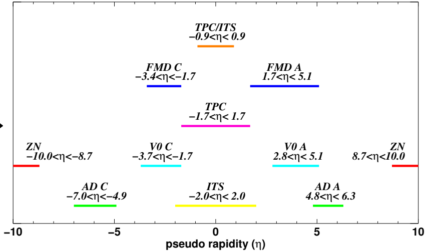
The ALICE detector system

ALICE detector



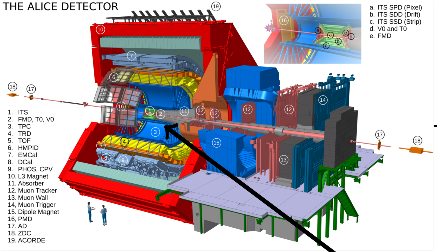
→

η coverage

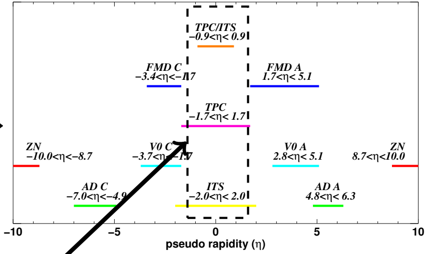


The ALICE detector system

ALICE detector



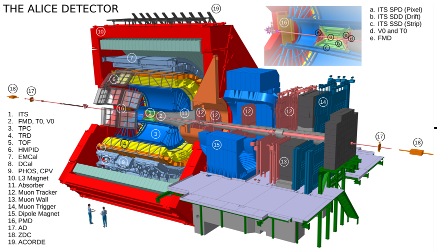
η coverage



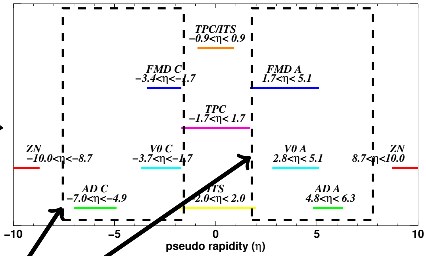
Central barrel $\rightarrow$ determine p^{μ}

The ALICE detector system

ALICE detector



η coverage



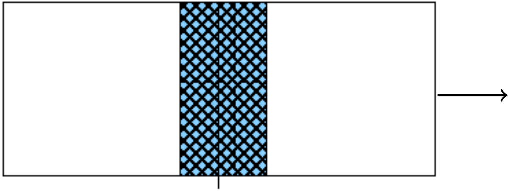
Small detectors for global event characterization

Central exclusive production at ALICE

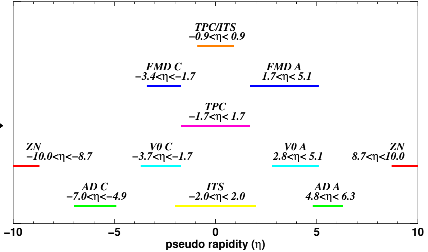
To study the CEP events a η gap condition is used

1. Signal in the central barrel
2. No activity in veto detectors

CEP event



η coverage

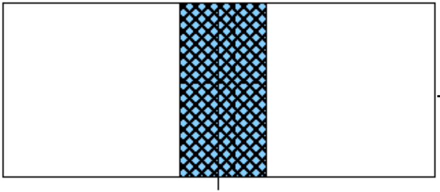


Central exclusive production at ALICE

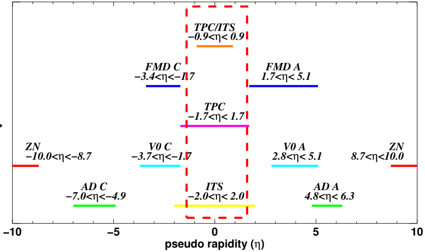
To study the CEP events a η gap condition is used

1. Signal in the central barrel
2. No activity in veto detectors

CEP event



η coverage

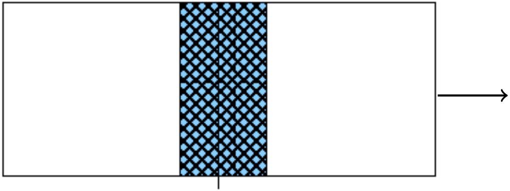


Central exclusive production at ALICE

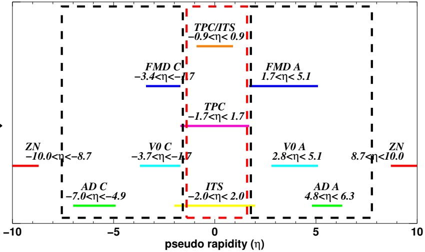
To study the CEP events a η gap condition is used

1. Signal in the central barrel
2. No activity in veto detectors

CEP event

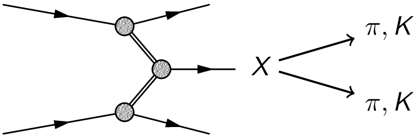


η coverage



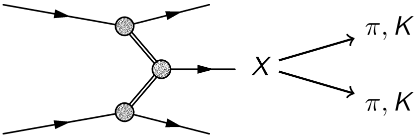
Regge theory - short overview

- Reactions characterized by a color neutral t -channel exchange carrying vacuum quantum numbers
  $\rightarrow$ Quantum number filter on $J^{PC} = (\text{even})^{++}$ states
- CEP $\rightarrow$ hadron spectroscopy for < 2.5 GeV
  - Study mass spectrum
  - Lightest *glueball* predicted in that region $J^{PC} = (0)^{++}$ state



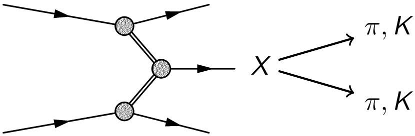
Regge theory - short overview

- Reactions characterized by a color neutral t-channel exchange carrying vacuum quantum numbers
  $\rightarrow$ Quantum number filter on $J^{PC} = (\text{even})^{++}$ states
- CEP $\rightarrow$ hadron spectroscopy for < 2.5 GeV
  - Study mass spectrum
  - Lightest *glueball* predicted in that region $J^{PC} = (0)^{++}$ state



Regge theory - short overview

- Reactions characterized by a color neutral t -channel exchange carrying vacuum quantum numbers
  $\rightarrow$ Quantum number filter on $J^{PC} = (\text{even})^{++}$ states
- CEP $\rightarrow$ hadron spectroscopy for < 2.5 GeV
  - Study mass spectrum
  - Lightest *glueball* predicted in that region $J^{PC} = (0)^{++}$ state



Regge theory - short overview

particle	IG(JPC)
η	0+(0-+)
$f_0(500)$ or σ was $f_0(600)$	0+(0++)
$\rho(770)$	1+(1–)
$K^*_0(800)$ or κ	1/2(0+)
$\omega(782)$	0-(1–)
$K^*(892)$	1/2(1-)
$\eta'(958)$	0+(0-+)
$f_0(980)$	0+(0++)
$a_0(980)$	1-(0++)
$\phi(1020)$	0-(1–)
$h_1(1170)$	0-(1+-)
$K_1(1270)$	1/2(1+)
$b_1(1235)$	1+(1+-)
$a_1(1260)$	1-(1++)
$f_2(1270)$	0+(2++)
$f_1(1285)$	0+(1++)
$\eta(1295)$	0+(0-+)
$\pi(1300)$	1-(0-+)
$a_2(1320)$	1-(2++)
$f_0(1370)$	0+(0++)
$\pi_1(1400)$	1-(1-+)
$K_1(1400)$	1/2(1+)
$\eta(1405)$	0+(0-+)
$K^*(1410)$	1/2(1-)
$f_1(1420)$	0+(1++)
$\omega(1420)$	0-(1–)
$K^*_0(1430)$	1/2(0+)
$K^*_2(1430)$	1/2(2+)
$a_0(1450)$	1-(0++)
K(1460)	1/2(0-)
$\rho(1450)$	1+(1–)
$\eta(1475)$	0+(0-+)
$f_0(1500)$	0+(0++)
$f'_2(1525)$	0+(2++)
$K_2 2(1580)$	1/2(2-)
$\pi_1(1600)$	1-(1-+)
$\eta_2(1645)$	0+(2-+)
$\omega(1650)$	0-(1–)
$K_1(1650)$	1/2(1+)
$\omega_3(1670)$	0-(3–)
$\pi_2(1670)$	1-(2-+)
$\phi(1680)$	0-(1–)
$K^*(1680)$	1/2(1-)
$\rho_3(1690)$	1+(3–)
$\rho(1700)$	1+(1–)
$f_0(1710)$	0+(0++)
$\pi(1800)$	1-(0-+)
$\phi_3(1850)$	0-(3–)

Quantum number filter

particle	IG(JPC)
$f_0(500)$ or σ was $f_0(600)$	0+(0++)
$f_0(980)$	0+(0++)
$f_2(1270)$	0+(2++)
$f_0(1370)$	0+(0++)
$f_0(1500)$	0+(0++)
$f'_2(1525)$	0+(2++)
$f_0(1710)$	0+(0++)
$f_2(1950)$	0+(2++)

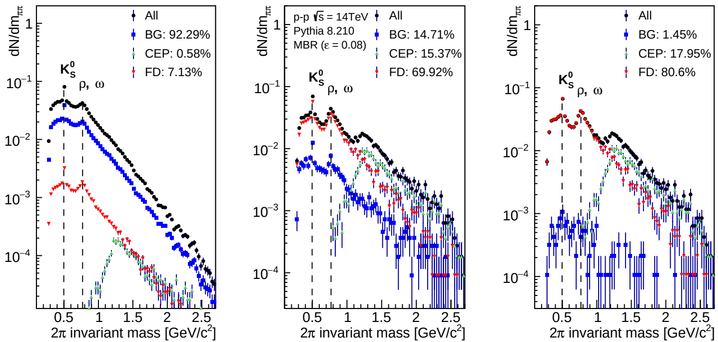
Glue ball ? $J^{PC} = 2^{++}$

Invariant mass spectrum

Studying *Pythia-8* simulations yield

- Enforcing η gap cut reduces non-diffractive almost entirely
- Remaining background are partially reconstructed CEP events - **feed down**

$\rightarrow$ increasing $\Delta\eta$

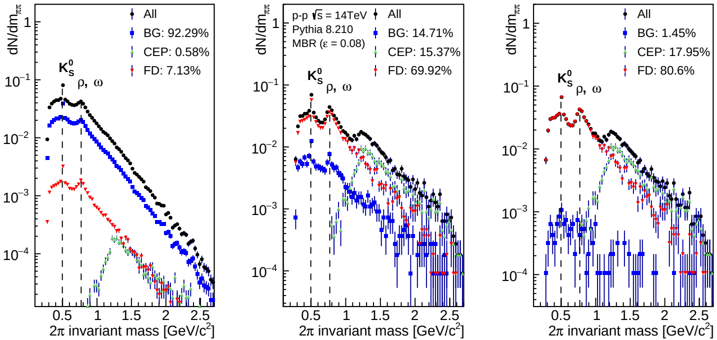


Invariant mass spectrum

Studying *Pythia-8* simulations yield

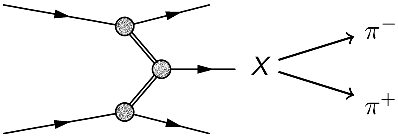
- Enforcing η gap cut reduces non-diffractive almost entirely
- Remaining background are partially reconstructed CEP events - **feed down**

$\rightarrow$ increasing $\Delta\eta$

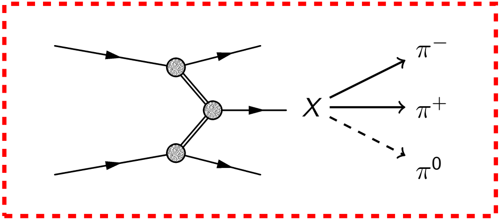


Feed down

Feed down *vs* fully reconstructed events



Total mass of X
recontructable



π^0 not detectable
$\rightarrow$ missing mass

Motivation

- Reduce dominant background contribution: **feed down**
  $\rightarrow$ Try multivariate approach instead of classical cut methods

- Next up
  - Comparison: Multi-variate *vs.* single-variate analysis

ML: an overview

In general ML represents a contrast to a *rule based systems*

Rule-based system

System that uses rules to make deductions or choices

- Domain-specific expert system
- Knowledge base: facts & rules (if $\rightarrow$ then statement)
- Rules manually specified (by expert) $\rightarrow$ expensive, incomplete

ML: an overview

In general ML represents a contrast to a *rule based systems*

Rule-based system

System that uses rules to make deductions or choices

- Domain-specific expert system
- Knowledge base: facts & rules (if $\rightarrow$ then statement)
- Rules manually specified (by expert) $\rightarrow$ expensive, incomplete

ML: an overview

In general ML represents a contrast to a *rule based systems*

Rule-based system

System that uses rules to make deductions or choices

- Domain-specific expert system
- Knowledge base: facts & rules (if $\rightarrow$ then statement)
- Rules manually specified (by expert) $\rightarrow$ expensive, incomplete

ML: an overview

In general ML represents a contrast to a *rule based systems*

Machine learning

- Algorithms that learn from *data* & make predictions on *data*
- Automatic methods $\rightarrow$ no human needed
- Human work required for defining problem & assessing the data

ML: an overview

In general ML represents a contrast to a *rule based systems*

Machine learning

- Algorithms that learn from *data* & make predictions on *data*
- Automatic methods $\rightarrow$ no human needed
- Human work required for defining problem & assessing the data

ML: an overview

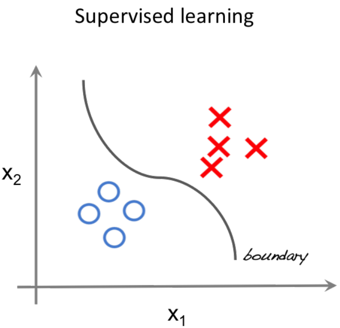
In general ML represents a contrast to a *rule based systems*

Machine learning

- Algorithms that learn from *data* & make predictions on *data*
- Automatic methods $\rightarrow$ no human needed
- Human work required for defining problem & assessing the data

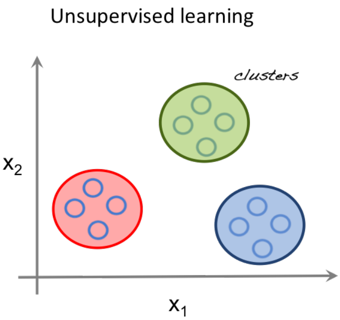
Types of ML

- Supervised
  - Classification
  - Regression
- Unsupervised



Types of ML

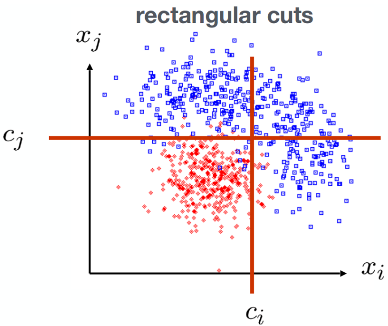
- Supervised
  - Classification
  - Regression
- Unsupervised



Rectangular cuts

Standard cut in one variable

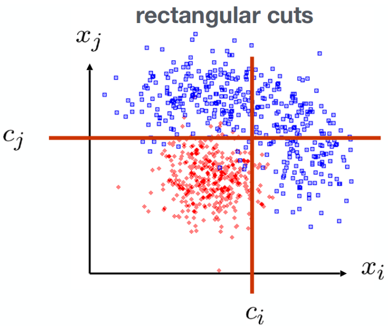
- Cuts only in lower-dimensional sub-spaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables
  $\rightarrow$ misclassification



Rectangular cuts

Standard cut in one variable

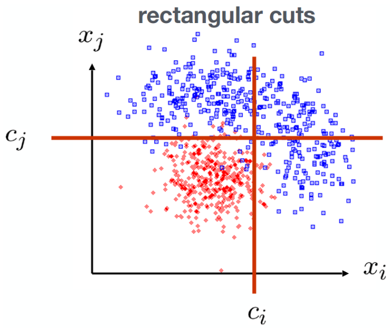
- Cuts only in lower-dimensional sub-spaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables
  $\rightarrow$ misclassification



Rectangular cuts

Standard cut in one variable

- Cuts only in lower-dimensional sub-spaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables $\rightarrow$ misclassification

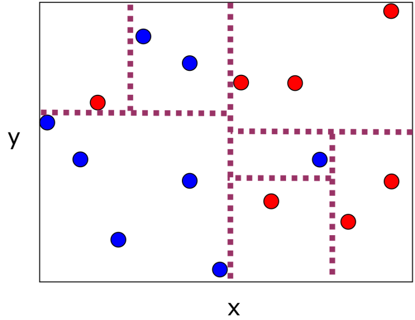


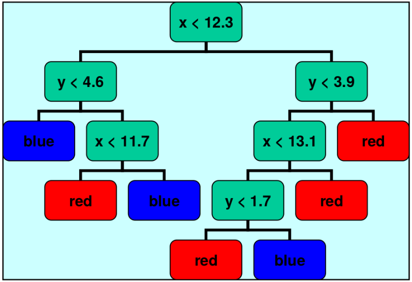
Rectangular cuts with *decision trees*

- Tree-like graph $\rightarrow$ flowchart
- Easy to understand
- Either be manually modeled by experts or learned from training data

Decision tree learning

Recursively split feature space into sub-spaces until they are pure

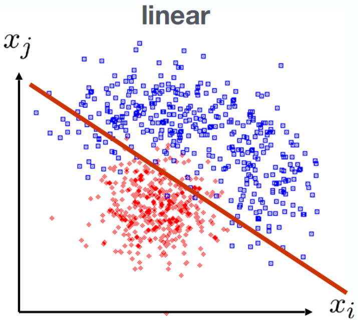




Linear cuts

More degrees of freedom than rectangular cut

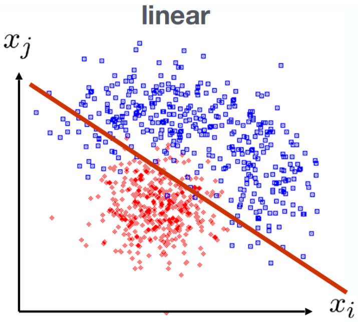
- Simple white box methods
- Very fast classification
- Can become very powerful by using *kernel trick*



Linear cuts

More degrees of freedom than rectangular cut

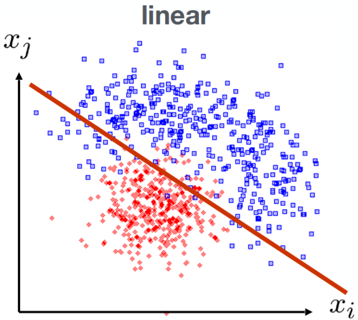
- Simple white box methods
- Very fast classification
- Can become very powerful by using *kernel trick*



Linear cuts

More degrees of freedom than rectangular cut

- Simple white box methods
- Very fast classification
- Can become very powerful by using *kernel trick*



Linear models

Takes a *linear function* of its inputs $\boldsymbol{x} = (x_1, \ldots x_n)$ to base its decision on.

$$y : \mathbb{R}^n \to \mathbb{R} \mid \boldsymbol{x} \mapsto y = f(\boldsymbol{w} \cdot \boldsymbol{x}) = f(\sum_j w_j x_j)$$

$\boldsymbol{w} \ldots$ weight vector

Simplest case

$$y = f(x) = \Theta(x) = \begin{cases} 0 & x < 0 \quad \text{background} \\ 1 & x \geq 0 \quad \text{signal} \end{cases}$$

$\rightarrow$ Function can be approximated by **single layer perceptron**

Linear models

Takes a *linear function* of its inputs $\boldsymbol{x} = (x_1, \dots x_n)$ to base its decision on.

$$y : \mathbb{R}^n \to \mathbb{R} \mid \boldsymbol{x} \mapsto y = f(\boldsymbol{w} \cdot \boldsymbol{x}) = f(\sum_j w_j x_j)$$

$\boldsymbol{w} \dots$ weight vector

Simplest case

$$y = f(x) = \Theta(x) = \begin{cases} 0 & x < 0 \quad \text{background} \\ 1 & x \geq 0 \quad \text{signal} \end{cases}$$

$\rightarrow$ Function can be approximated by **single layer perceptron**

Linear models

Takes a *linear function* of its inputs $\boldsymbol{x} = (x_1, \ldots x_n)$ to base its decision on.

$$y : \mathbb{R}^n \to \mathbb{R} \mid \boldsymbol{x} \mapsto y = f(\boldsymbol{w} \cdot \boldsymbol{x}) = f(\sum_j w_j x_j)$$

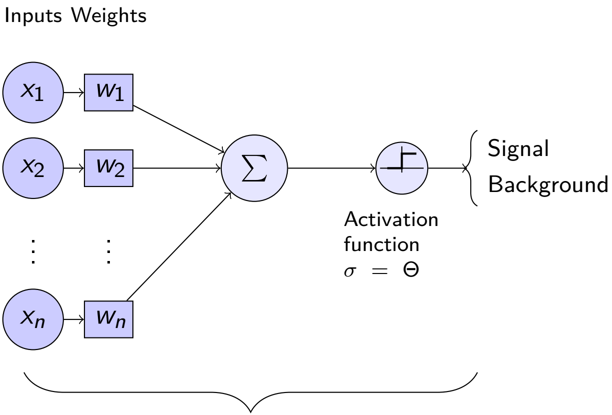
$\boldsymbol{w} \ldots$ weight vector

Simplest case

$$y = f(x) = \Theta(x) = \begin{cases} 0 & x < 0 \quad \text{background} \\ 1 & x \geq 0 \quad \text{signal} \end{cases}$$

$\rightarrow$ Function can be approximated by **single layer perceptron**

Single layer perceptron (SLP)

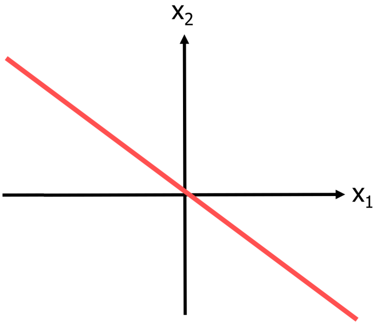


$$y = \sigma\left(\sum_j w_j x_j\right)$$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
    1. Present training sample $\boldsymbol{x}$
    2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
    3. If $\Delta \neq 0 \;\; \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$

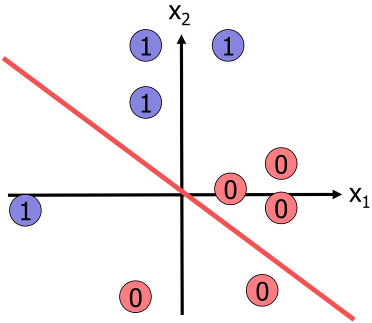


$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
    1. Present training sample $\boldsymbol{x}$
    2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
    3. If $\Delta \neq 0 \ \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$



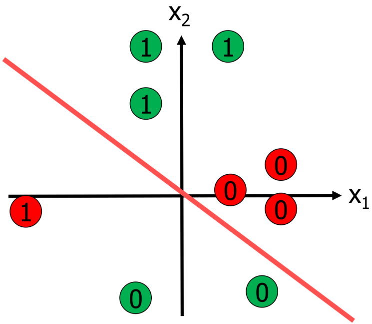
$$y = w_1x_1 + w_2x_2$$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
   1. Present training sample $\boldsymbol{x}$
   2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
   3. If $\Delta \neq 0 \ \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha \Delta \boldsymbol{x}$

$\alpha \ldots$ learning rate



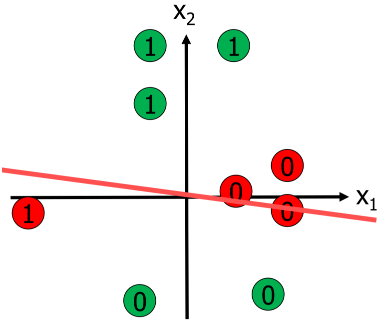
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
   1. Present training sample $\boldsymbol{x}$
   2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
   3. If $\Delta \neq 0 \ \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$

$\alpha \ldots$ learning rate



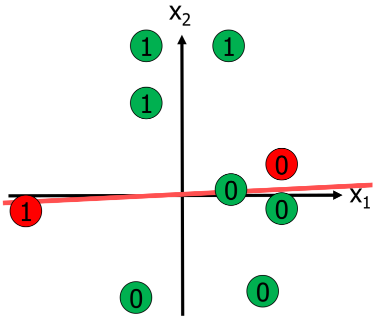
$y = w_1 x_1 + w_2 x_2$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
   1. Present training sample $\boldsymbol{x}$
   2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
   3. If $\Delta \neq 0 \ \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$

$\alpha \ldots$ learning rate



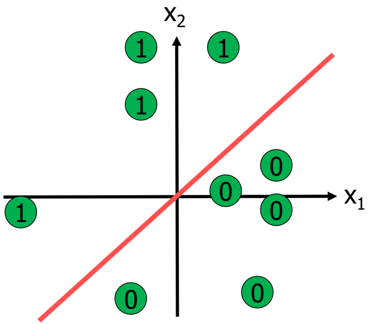
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
   1. Present training sample $\boldsymbol{x}$
   2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
   3. If $\Delta \neq 0 \ \rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$

$\alpha \ldots$ learning rate



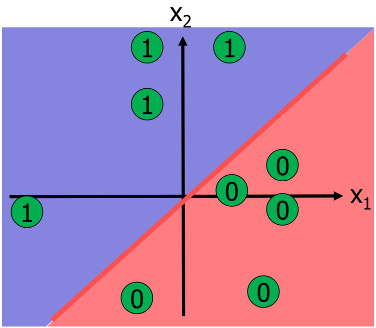
$y = w_1 x_1 + w_2 x_2$

SLP training

Algorithm

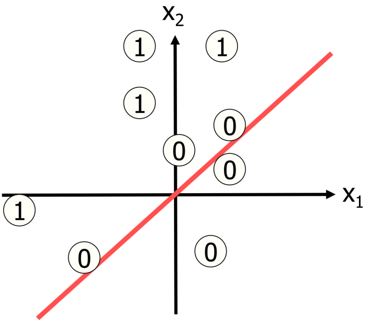
1. Initialize weights $\boldsymbol{w}$
2. Repeat until $y_{predict} = y_{target}$:
   1. Present training sample $\boldsymbol{x}$
   2. Predict sample label $y = \Theta(\boldsymbol{x}\boldsymbol{w}_{init})$ and compute error $\Delta = y_{target} - y$
   3. If $\Delta \neq 0 \;\rightarrow$ update weights $\boldsymbol{w}' = \boldsymbol{w} + \alpha\Delta\boldsymbol{x}$

$\alpha \ldots$ learning rate

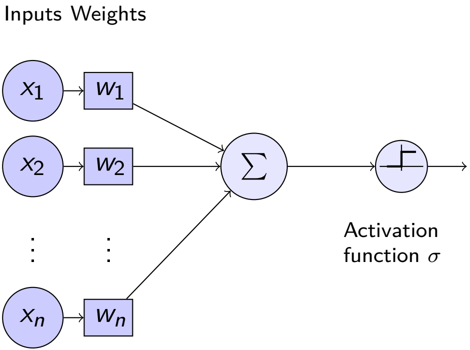


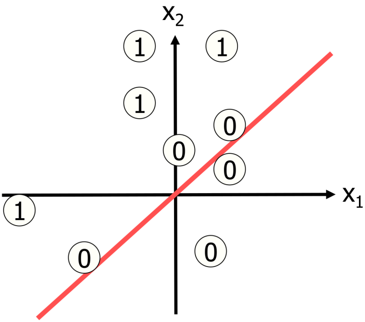
$$y = w_1x_1 + w_2x_2$$

SLP bias term

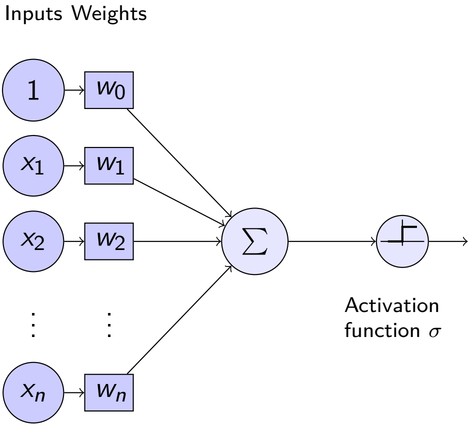


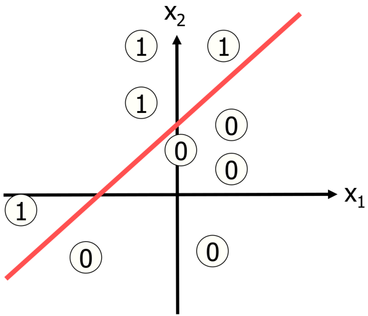
SLP bias term



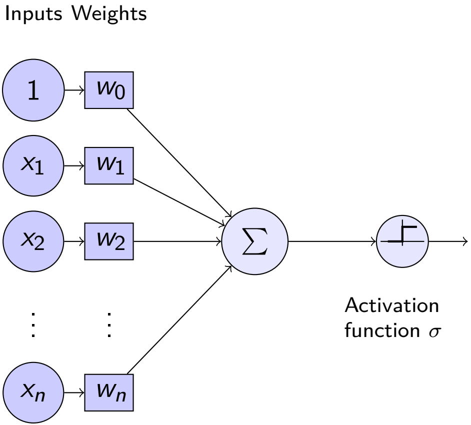


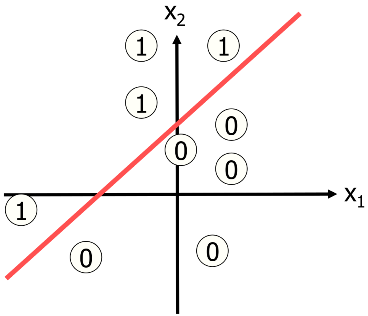
SLP bias term





SLP bias term

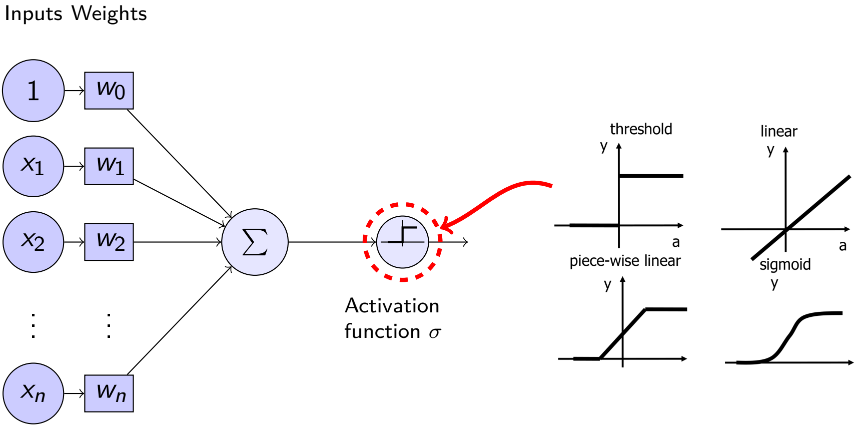




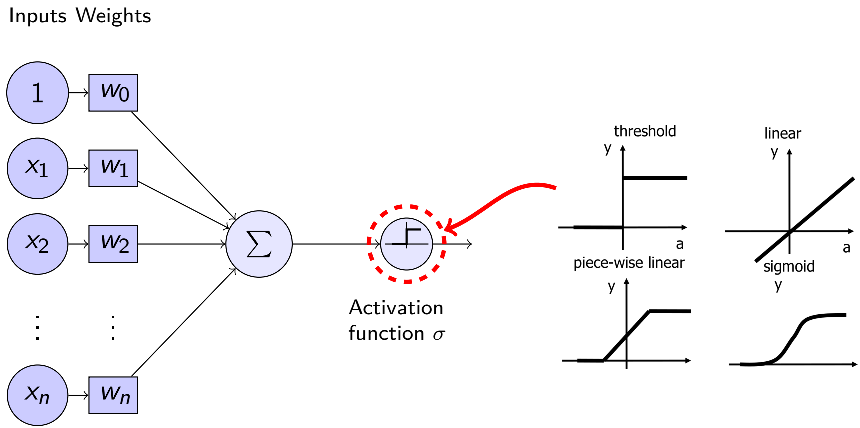
Bias weight w_0 learned just as the other weights

$$y = \sigma\left(w_0 + \sum_{j=1}^{n} w_j x_j\right)$$

SLP activation functions



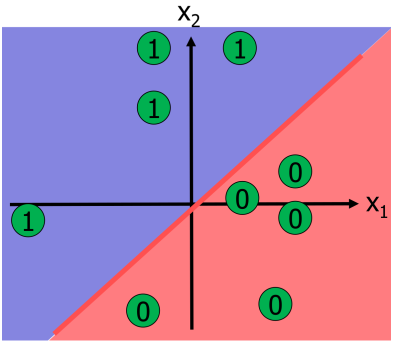
SLP activation functions



Often used: *sigmoid* $\rightarrow$ output $\in (0, 1)$

Non-linear activation functions: output

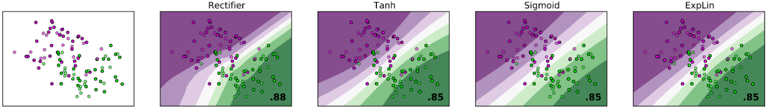
Distance to decision boundary comes into play

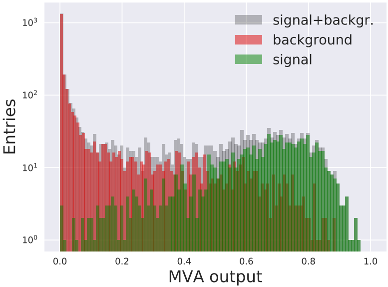


MVA output $y = 0$ *or* 1

Non-linear activation functions: output

Distance to decision boundary comes into play





MVA output
$y \in (0, 1)$

Improving linear methods

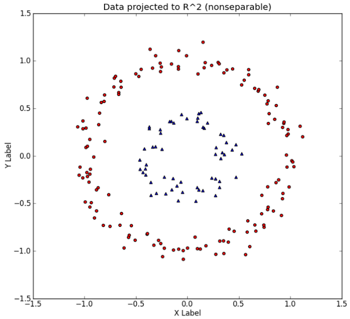
Kernel trick

Map data to a higher dimensional space where linear hyper-plane can again be found

Improving linear methods

Kernel trick

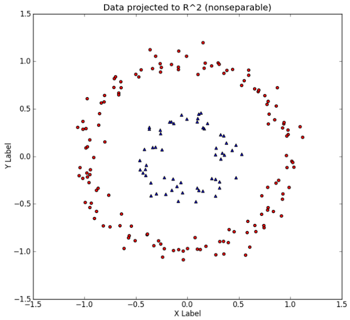
Map data to a higher dimensional space where linear hyper-plane can again be found

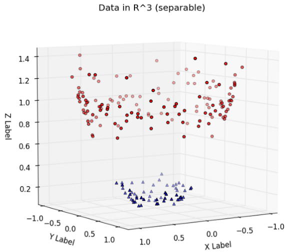


Improving linear methods

Kernel trick

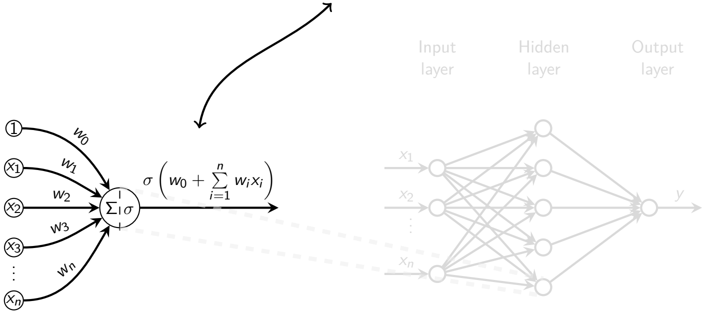
Map data to a higher dimensional space where linear hyper-plane can again be found





Non-linear models

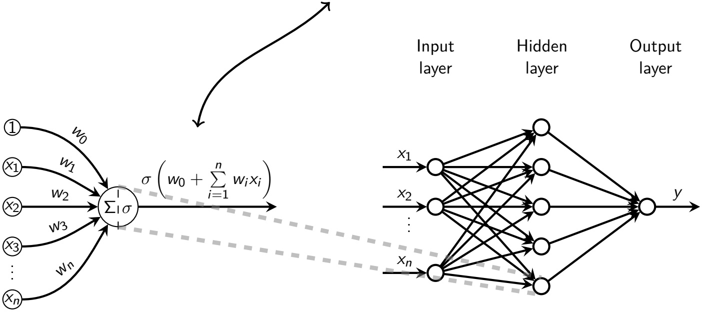
$$y = \sigma[w_0^{(2)} + \overbrace{\boldsymbol{w}^{(2)}}^{\mathbb{R}^m} \cdot \underbrace{\sigma(\overbrace{\boldsymbol{w}_0^{(1)}}^{\mathbb{R}^m} + \overbrace{\boldsymbol{W}^{(1)}}^{\mathbb{R}^{m\times n}} \cdot \boldsymbol{x})}_{\text{Output of hidden layer: } \mathbb{R}^m}]$$



→ The hidden layer *learns* a representation so that the data is linearly separable

Non-linear models

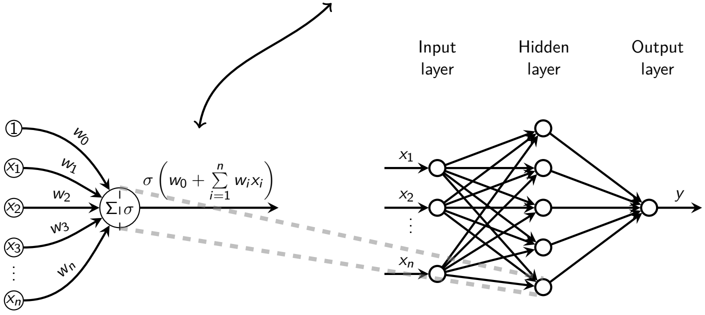
$$y = \sigma\Big[w_0^{(2)} + \overbrace{\boldsymbol{w}^{(2)}}^{\mathbb{R}^m} \cdot \underbrace{\sigma\big(\overbrace{\boldsymbol{w}_0^{(1)}}^{\mathbb{R}^m} + \overbrace{\boldsymbol{W}^{(1)}}^{\mathbb{R}^{m\times n}} \cdot \boldsymbol{x}\big)}_{\text{Output of hidden layer: } \mathbb{R}^m}\ \Big]$$



→ The hidden layer *learns* a representation so that the data is linearly separable

Non-linear models

$$y = \sigma[w_0^{(2)} + \overbrace{\boldsymbol{w}^{(2)}}^{\mathbb{R}^m} \cdot \underbrace{\sigma(\overbrace{\boldsymbol{w}_0^{(1)}}^{\mathbb{R}^m} + \overbrace{\boldsymbol{W}^{(1)}}^{\mathbb{R}^{m\times n}} \cdot \boldsymbol{x})}_{\text{Output of hidden layer: } \mathbb{R}^m}\ ]$$

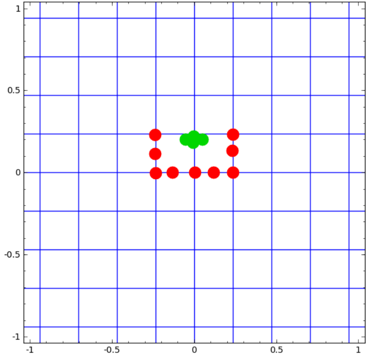


→ The hidden layer *learns* a representation so that the data is linearly separable

Visualizing the hidden layer

1. Linear transformation: $\boldsymbol{W}\boldsymbol{x}$
2. Translation: $\boldsymbol{w}_0$
3. Non-linear activation: σ

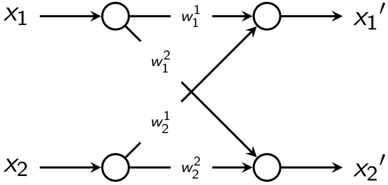
$$\tilde{\boldsymbol{x}} = \boldsymbol{x}$$

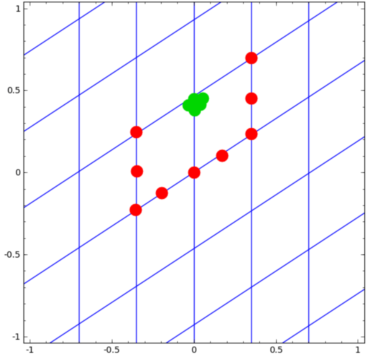


Visualizing the hidden layer

1. Linear transformation: $\boldsymbol{W}\boldsymbol{x}$
2. Translation: $\boldsymbol{w}_0$
3. Non-linear activation: σ

$$\tilde{\boldsymbol{x}} = (\boldsymbol{W}^{(1)} \cdot \boldsymbol{x})$$

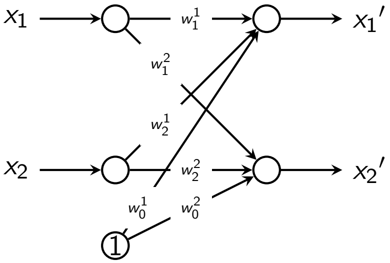


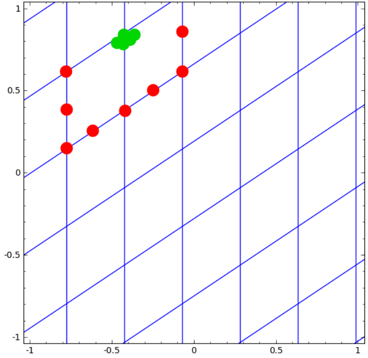


Visualizing the hidden layer

1. Linear transformation: $\boldsymbol{W}\boldsymbol{x}$
2. Translation: $\boldsymbol{w}_0$
3. Non-linear activation: σ

$$\tilde{\boldsymbol{x}} = (\boldsymbol{w}_0^{(1)} + \boldsymbol{W}^{(1)} \cdot \boldsymbol{x})$$

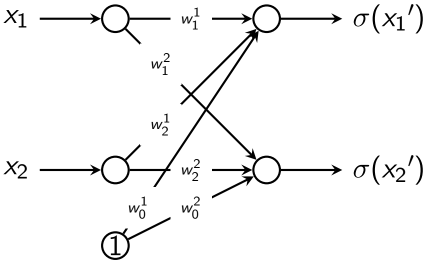


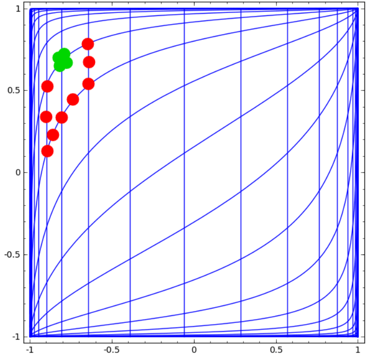


Visualizing the hidden layer

1. Linear transformation: $\boldsymbol{Wx}$
2. Translation: $\boldsymbol{w}_0$
3. Non-linear activation: σ

$$\tilde{\boldsymbol{x}} = \sigma(\boldsymbol{w}_0^{(1)} + \boldsymbol{W}^{(1)} \cdot \boldsymbol{x})$$

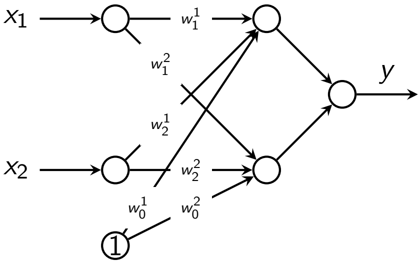


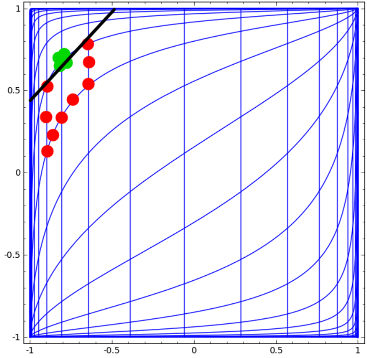


Visualizing the hidden layer

1. Linear transformation: $\boldsymbol{W}\boldsymbol{x}$
2. Translation: $\boldsymbol{w}_0$
3. Non-linear activation: σ

$$y = \sigma(w_0^{(2)} + \boldsymbol{w}^{(2)} \cdot \tilde{\boldsymbol{x}})$$





Update weights via back-propergation

Back-propagation

Adapt weights by going backwards in the network

- Initialize weights
- Evaluate $y_{predict} = f(\boldsymbol{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss $L = L(y_{true}, \boldsymbol{w}^{(1)}, \boldsymbol{W}^{(2)})$

Update weights via back-propergation

Back-propagation

Adapt weights by going backwards in the network

- Initialize weights
- Evaluate $y_{predict} = f(\boldsymbol{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss $L = L(y_{true}, \boldsymbol{w}^{(1)}, \boldsymbol{W}^{(2)})$

Update weights via back-propergation

Back-propagation

Adapt weights by going backwards in the network

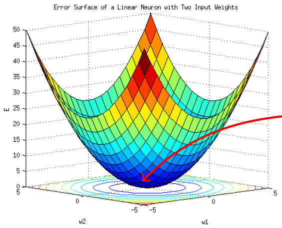
- Initialize weights
- Evaluate $y_{predict} = f(\boldsymbol{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss $L = L(y_{true}, \boldsymbol{w}^{(1)}, \boldsymbol{W}^{(2)})$

Update weights via back-propergation

Back-propagation

Adapt weights by going backwards in the network

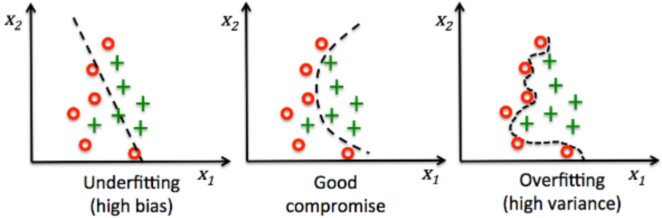
- Initialize weights
- Evaluate $y_{predict} = f(\boldsymbol{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss $L = L(y_{true}, \boldsymbol{w}^{(1)}, \boldsymbol{W}^{(2)})$



Minimize loss $\rightarrow$ find best fit

Bias-variance trade-off

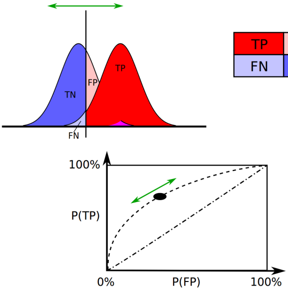
- Small variance: Classifiers with low degrees of freedom are less prone to statistical fluctuations
  $\rightarrow$ different training samples result in similar classification boundaries
- However: if data contain features that a model with few degrees of freedom cannot describe, a bias is introduced



Quantify MVA output

Evaluate classifier performance

- ROC curve: continuously scan y & plot background acceptance (FPR) vs. signal efficiency (TPR)



TP	FP
FN	TN

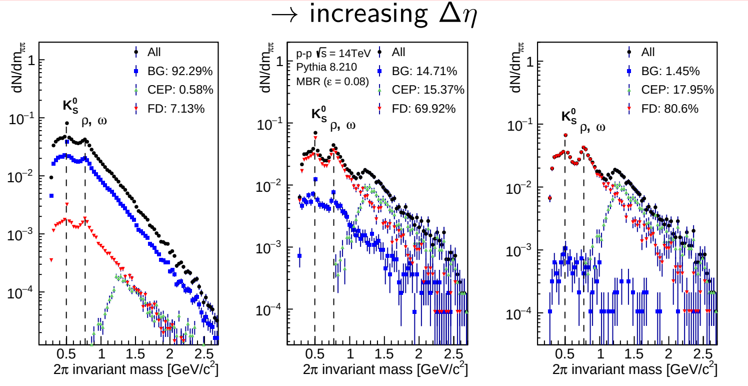
$$FPR = FP/(FP+TN)$$

$$TPR = TP/\ (TP + FN)$$

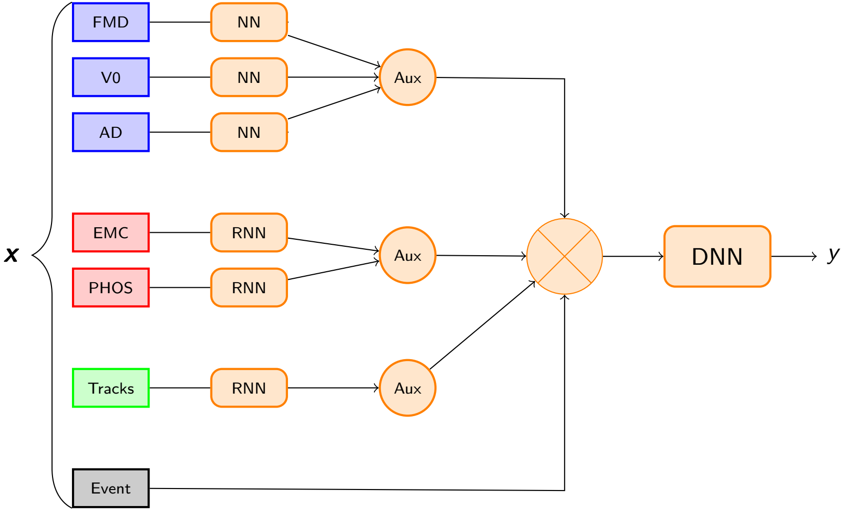
ROC-AUC: probability that classifier ranks randomly chosen positive sample higher than randomly chosen negative one

MVA applied to CEP data

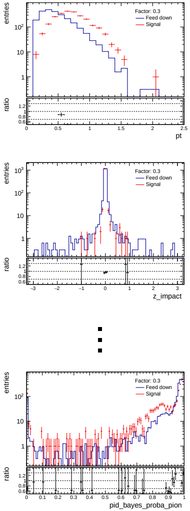
In order to reduce the main background component - **feed down** - ML is applied on simulated data.



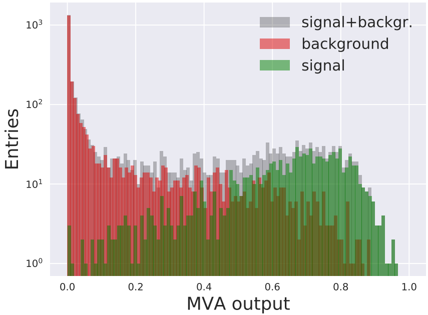
Neural network structure



Results

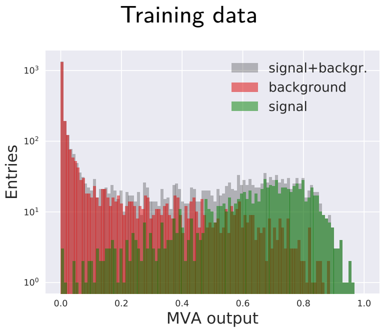


$$\mathbb{R}^n \to \mathbb{R}$$



Results

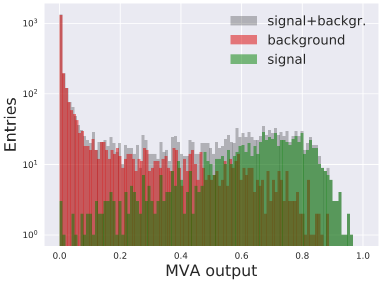
ROC curve



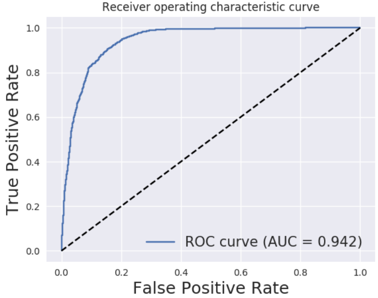
Results

ROC curve

Training data



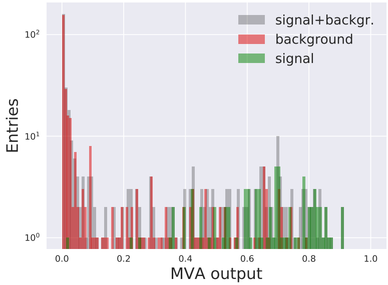
ROC train



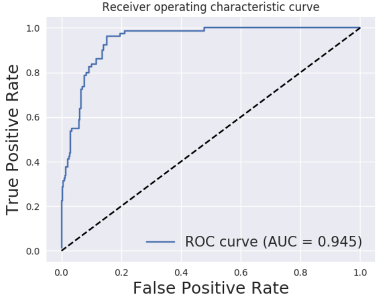
Results

ROC curve

Test data

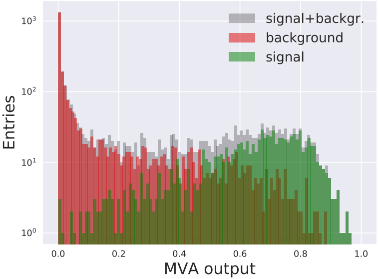


ROC test



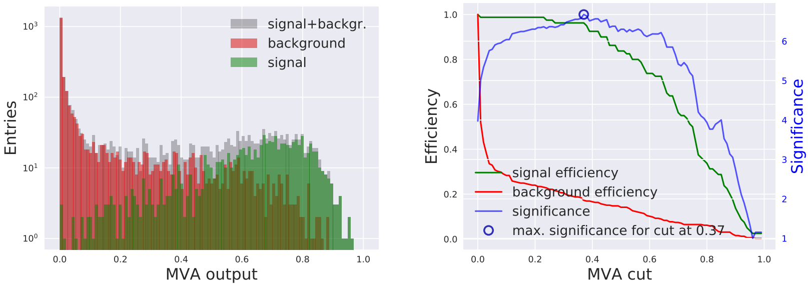
Results

To find the optimal cut on MVA output we evaluate significance along y



Results

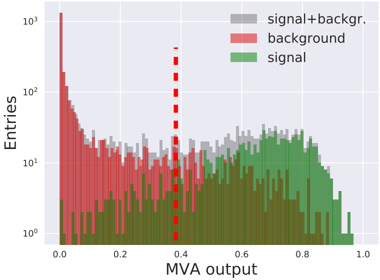
To find the optimal cut on MVA output we evaluate significance along y

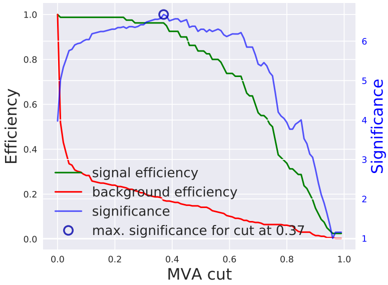


$$\text{Significance } S = \frac{N_{sig}}{\sqrt{N_{sig}+N_{BG}}}$$

Results

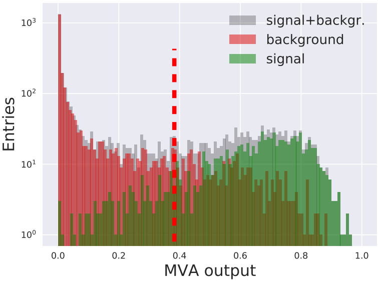
To find the optimal cut on MVA output we evaluate significance along y



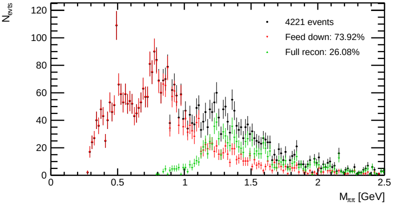


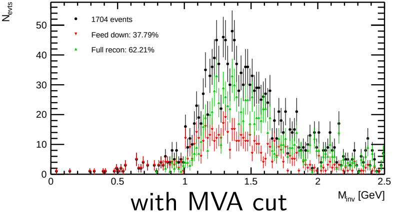
$$\text{Significance } S = \frac{N_{sig}}{\sqrt{N_{sig}+N_{BG}}}$$

Results



no MVA cut



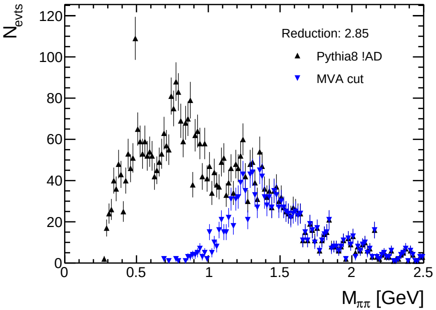


with MVA cut

Results

Spectrum comparison yields MVA cut $\leftrightarrow$ mass cut

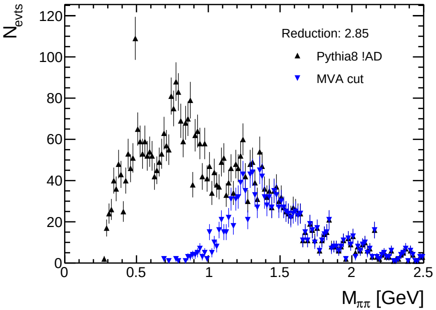
Pythia-8 (training & test)



Results

Spectrum comparison yields MVA cut $\leftrightarrow$ mass cut

Pythia-8 (training & test)



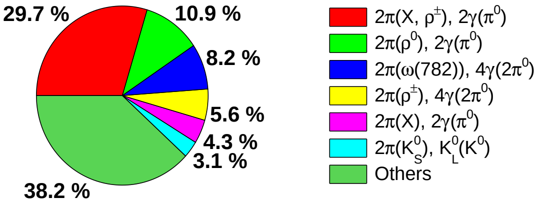
Challenges

- Need higher statistics to train models
- Missing components in MC data
- Feature distributions real $\leftrightarrow$ MC data not always overlapping
- CEP simulations active field of research
- EMCal corrections

EMCal corrections

- Currently: emcal data are not making a difference
  $\rightarrow$ however, they should!
- EMCal correction framework should (hopefully) fix that

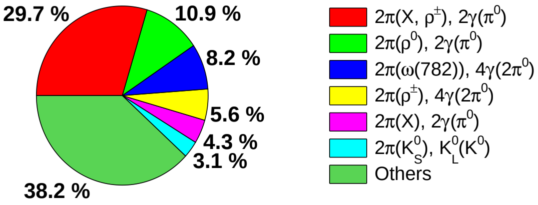
Feed down contributions



EMCal corrections

- Currently: emcal data are not making a difference
  $\rightarrow$ however, they should!
- EMCal correction framework should (hopefully) fix that

Feed down contributions



Conclusion

- *Pythia-8* simulations show η gap condition drastically reduces non-diffractive background
- Dominant remaining background is composed of partially reconstructed (**feed down**) events
- MVA provides reasonable results on simulated data $\rightarrow$ needs further steps on real data