

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

Sebastian Ratzenböck¹

¹Stefan Meyer Institut
Österreichische Akademie der Wissenschaften

26. April 2018

Outline

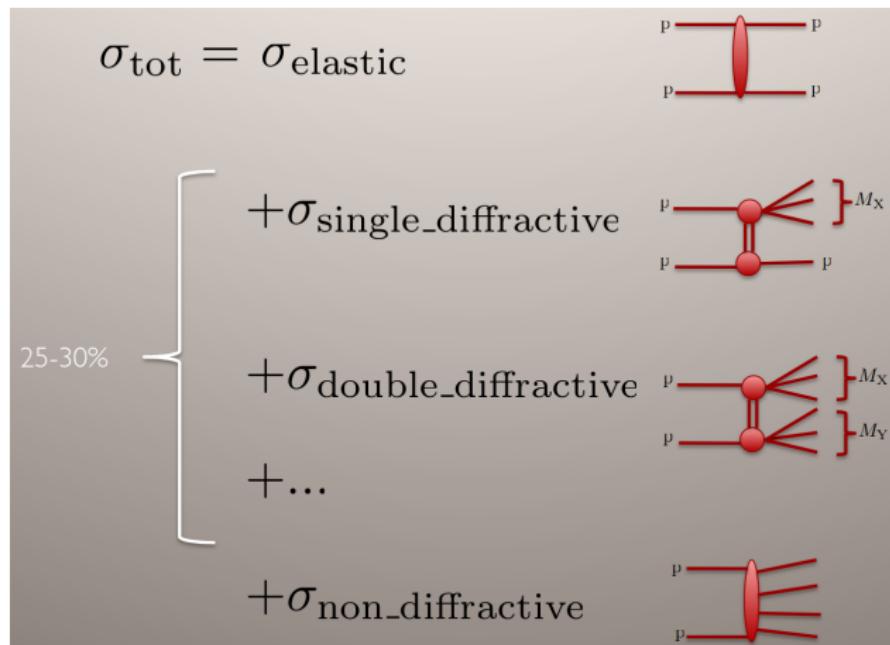
1 Central exclusive production at ALICE

2 ML: an overview

- Rectangular cuts
- Linear cuts
- Non-linear cuts

3 Results & Conclusion

Introduction

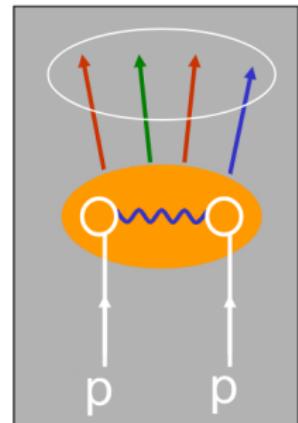


Cross section contribution at LHC energies

Diffractive definition

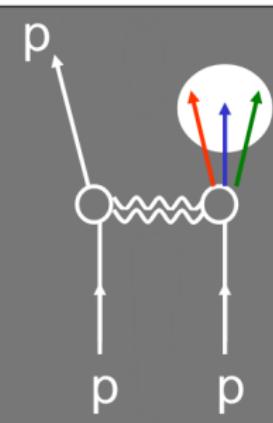
Diffractive events are reactions where **no quantum numbers are exchanged** → leads to special topology

Incident hadrons
acquire color
→ break apart



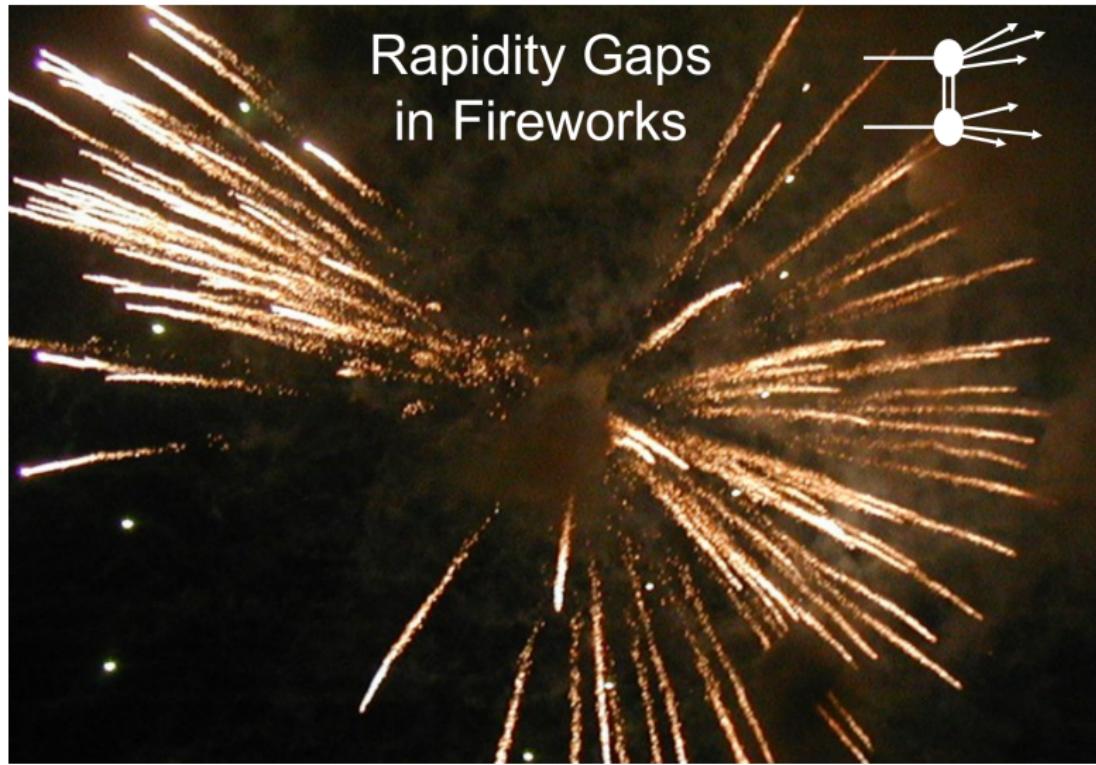
$$\eta = -\ln(\tan(\frac{\theta}{2}))$$

η gap exponentially suppressed



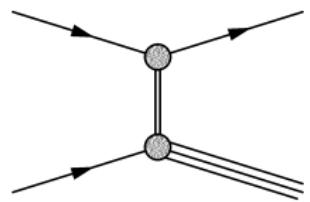
η gap not suppressed

Rapidity gap

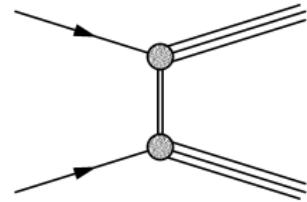


Diffractive processes

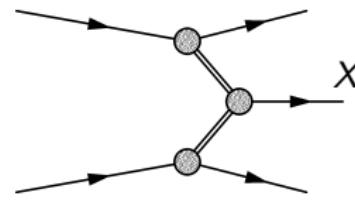
Different types of diffractive events are distinguished



Single diffractive



Double diffractive

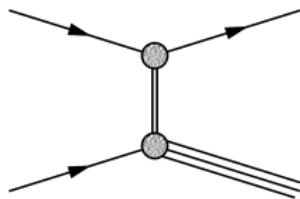


CEP

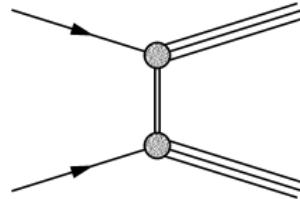
Described by *Regge theory*

Event topology

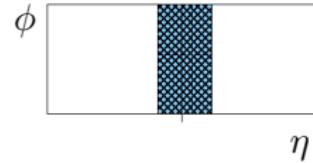
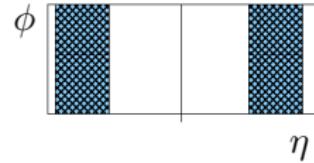
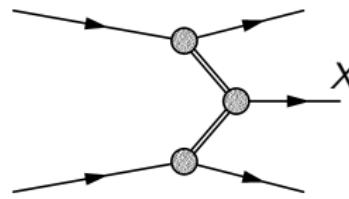
Single diffractive



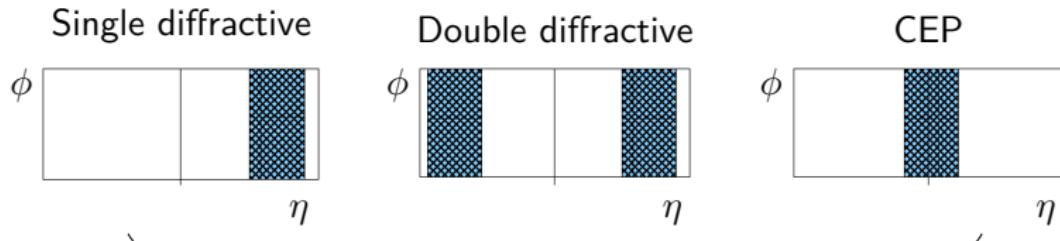
Double diffractive



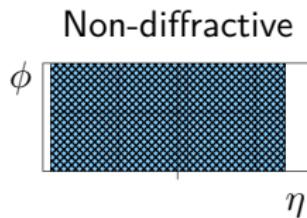
CEP



Event topology

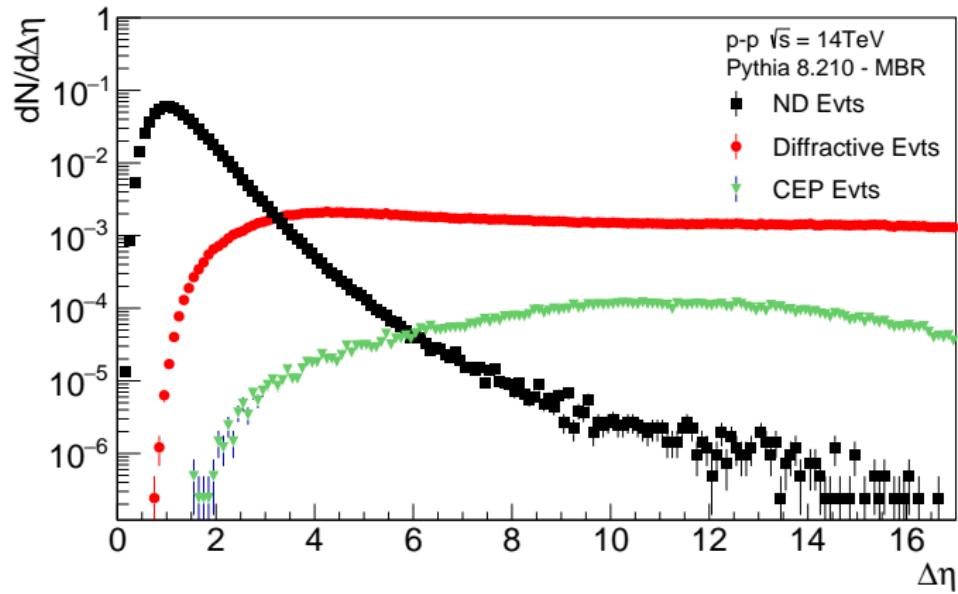


Large η gap compared to non-diffractive event



Event selection

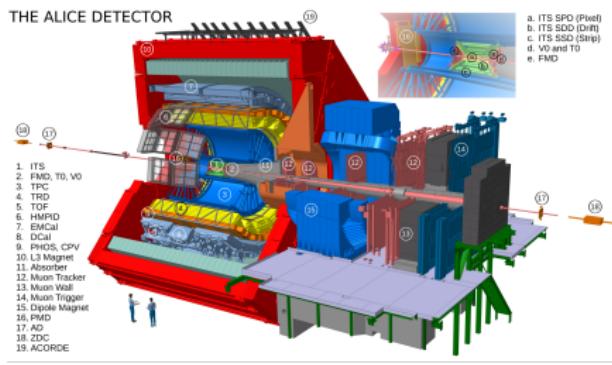
Large rapidity gaps in *non-diffractive* events are exponentially suppressed



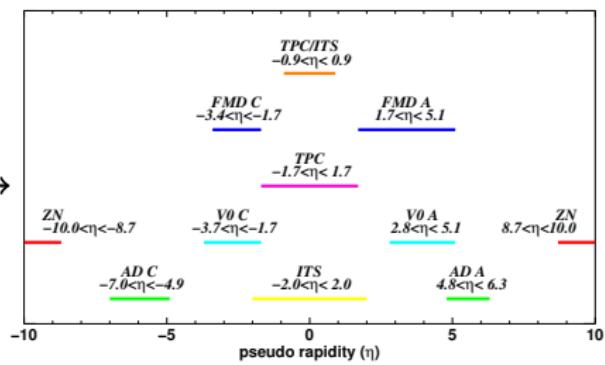
The ALICE detector system

ALICE detector

THE ALICE DETECTOR



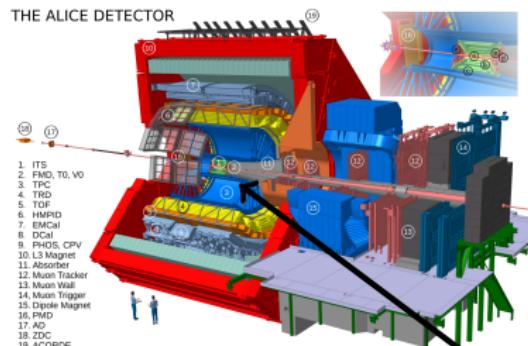
η coverage



The ALICE detector system

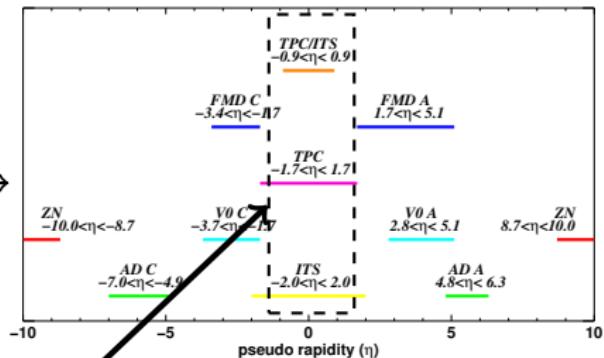
ALICE detector

THE ALICE DETECTOR



a.
b.
c.
d.
e.

η coverage

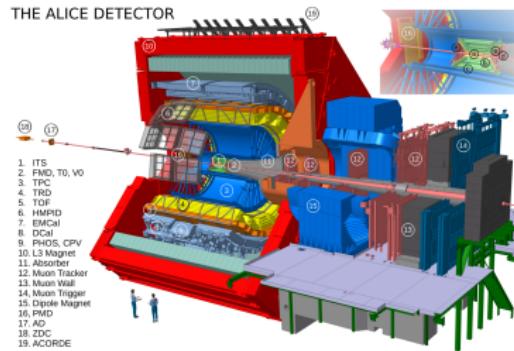


Central barrel \rightarrow determine p^μ

The ALICE detector system

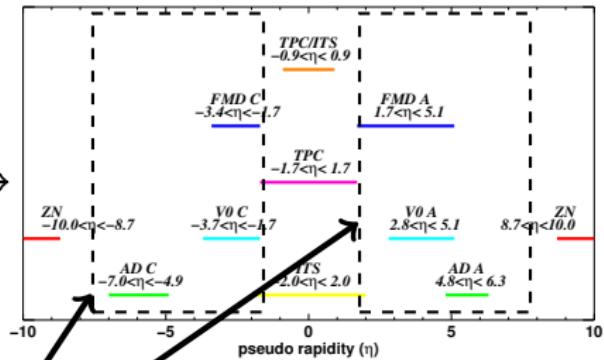
ALICE detector

THE ALICE DETECTOR



- a. ITS SPD (Pixel)
b. ITS TPC (Drift)
c. ITS SSD (Stripl)
d. V0 and T0
e. FMD

η coverage

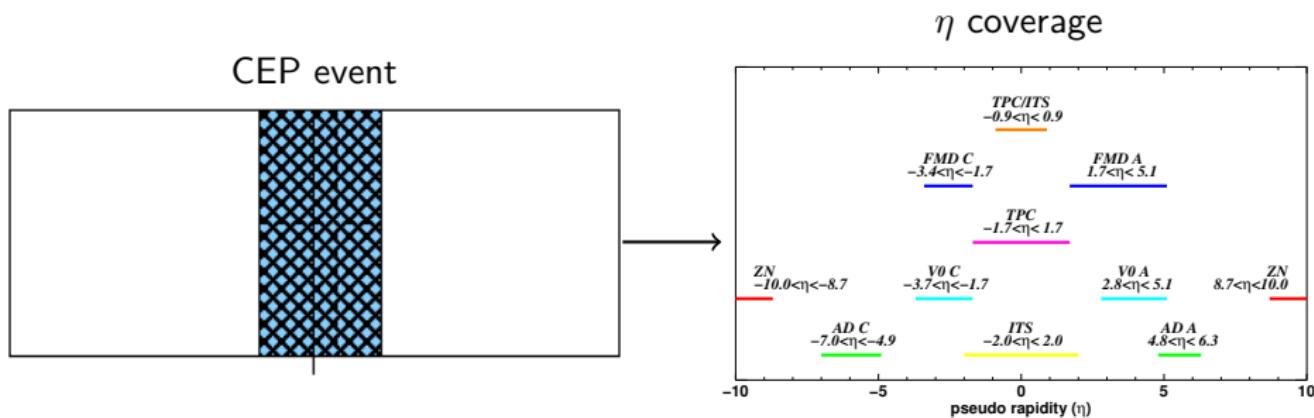


Small detectors for global event characterisation

Central exclusive production at ALICE

To study the CEP events a η gap condition is used

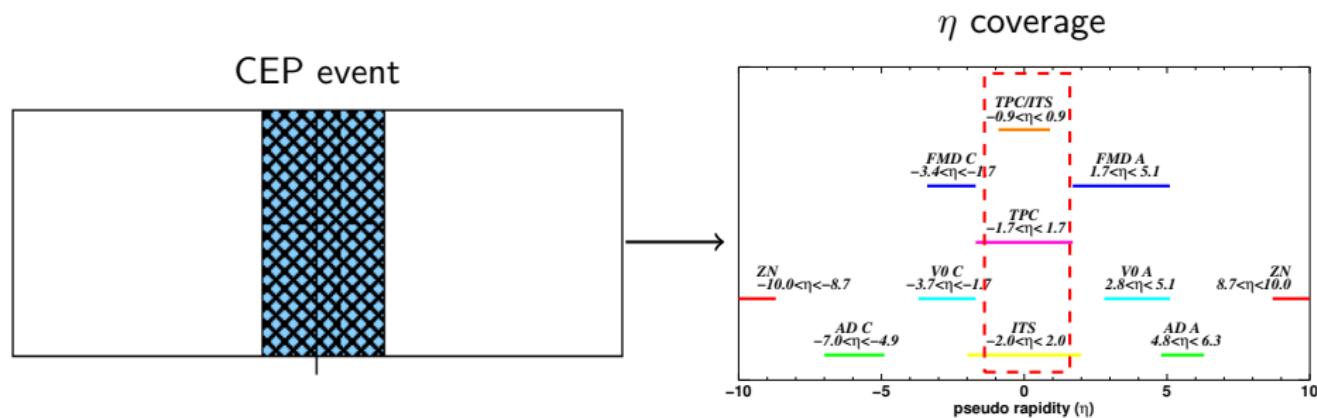
- ① Signal in the central barrel
- ② No activity in veto detectors



Central exclusive production at ALICE

To study the CEP events a η gap condition is used

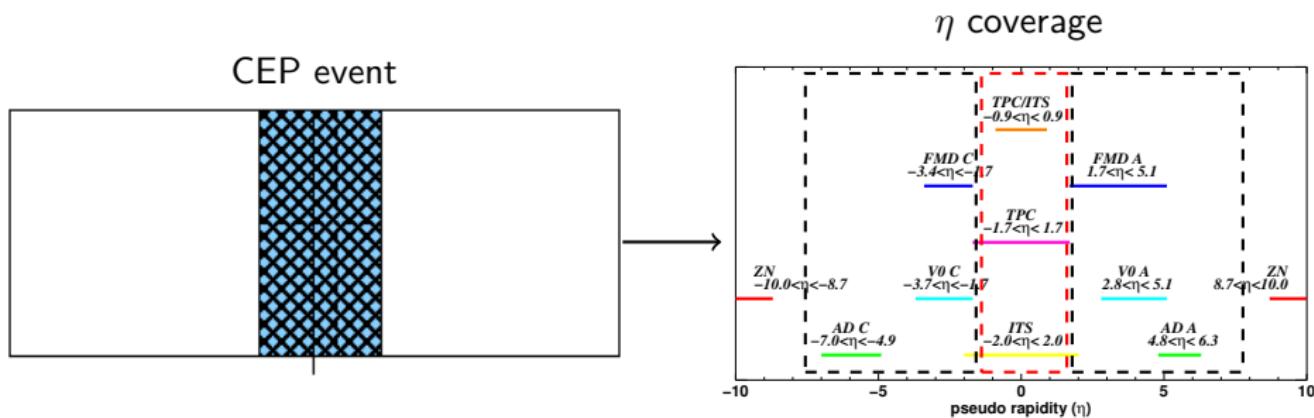
- ➊ Signal in the central barrel
- ➋ No activity in veto detectors



Central exclusive production at ALICE

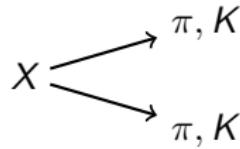
To study the CEP events a η gap condition is used

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



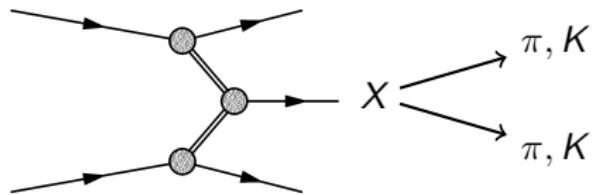
Regge theory - short overview

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



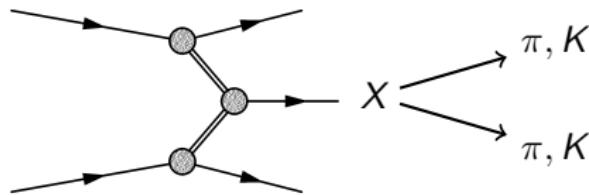
Regge theory - short overview

- Reactions characterized by a color neutral t -channel exchange carrying vacuum quantum numbers
→ Quantum number filter on $J^{PC} = (\text{even})^{++}$ states
- CEP → 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
→ Quantum number filter on $J^{PC} = (\text{even})^{++}$ states
- CEP → 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++)
$\rho'_0(770)$	1+(1-)
$K^*(800)$ or K^*	1/2(0+)
$\psi(782)$	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+-)
$K_*(1270)$	1/2(1+)
$b_1(1275)$	1+(1+-)
$a_1(1260)$	1-(1++)
$f_2(1270)$	0+(2++)
$f'_0(1285)$	0+(1++)
$\eta J(1285)$	0+(0-)
$\pi(1300)$	1-(0+)
$a_2(1320)$	1-(2++)
$f_0(1370)$	0+(0++)
$\pi^*(1400)$	1-(1-)
$K_*(1400)$	1/2(1+)
$\psi(1405)$	0+(0-+)
$K^*(1410)$	1/2(1-)
$f_1(1420)$	0-(1++)
$\omega(1420)$	0-(1-)
$K^*(1430)$	1/2(0+)
$K^*_0(1430)$	1/2(2++)
$a_0^*(1450)$	1-(0++)
$K(1460)$	1/2(0-)
$\rho(1450)$	1+(1-)
$\eta J(1475)$	0+(0-)
$f_0(1500)$	0+(0++)
$f_2(1525)$	0+(2++)
$K_{21}(1580)$	1/2(2-)
$\pi^*(1600)$	1-(1-)
$\eta_s(1645)$	0+(2-)
$\omega(1650)$	0-(1-)
$K_1(1650)$	1/2(1+)
$\omega(1670)$	0-(3-)
$\pi^*(1670)$	1-(2-)
$\phi(1680)$	0-(1-)
$K^*(1680)$	1/2(1-)
$\rho_2(1690)$	1+(3-)
$\rho(1700)$	1+(1-)
$f_0(1710)$	0+(0++)
$\pi(1800)$	1-(0+)
$\phi(1850)$	0-(3-)

Quantum number filter

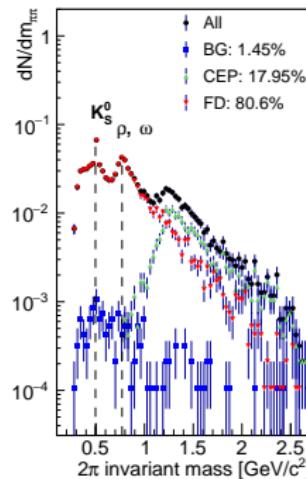
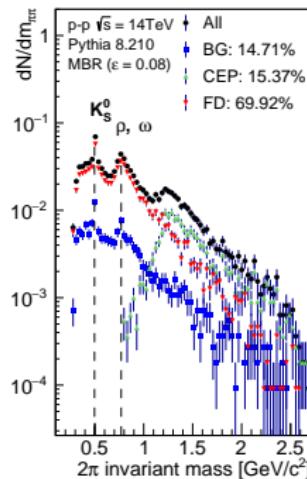
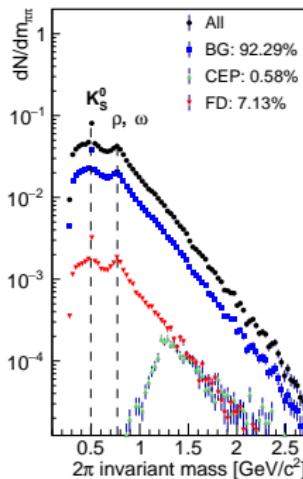
Glue ball? $J^{PC} = 2^{++}$

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

→ increasing $\Delta\eta$

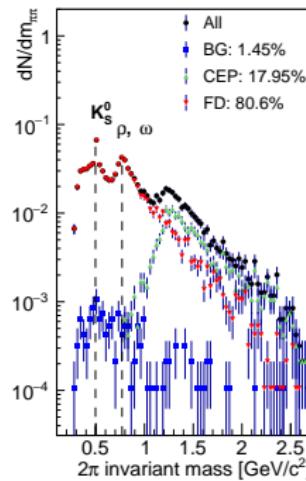
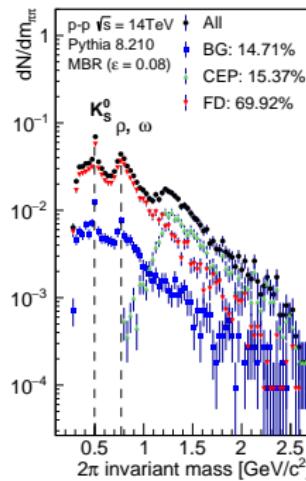
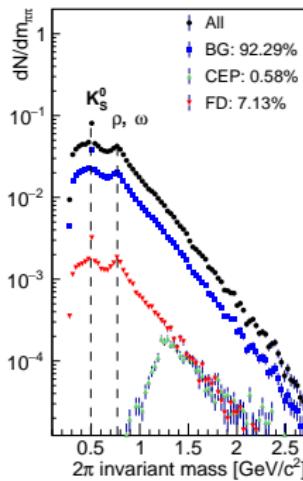


Invariant mass spectrum

Studying *Pythia-8* simulations yield

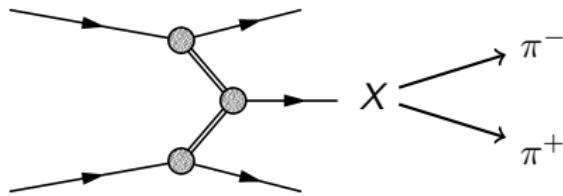
- Enforcing η gap cut reduces non-diffractive almost entirely
- Remaining background are partially reconstructed CEP events - **feed down**

→ increasing $\Delta\eta$

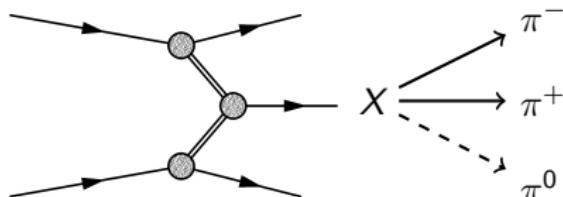


Feed down

Feed down vs fully reconstructed events



Total mass of X
reconstructable



π^0 not detectable
→ missing mass

Motivation

- Reduce dominant background contribution: **feed down**
 - 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 → then statement)
- Rules manually specified (by expert) → 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 → then statement)
- Rules manually specified (by expert) → 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 → then statement)
- Rules manually specified (by expert) → 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 → 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 → no human needed
- Human work required for defining problem & assessing the data

ML: an overview

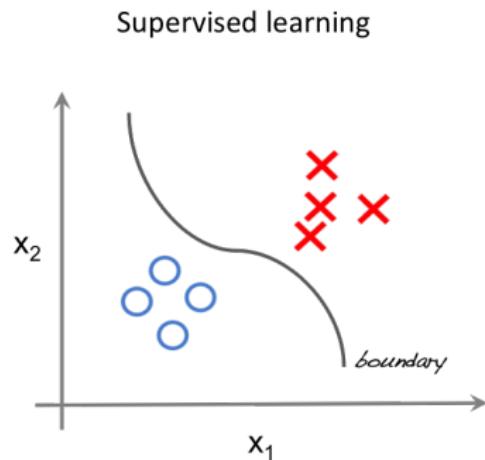
In general ML represents a contrast to a *rule based systems*

Machine learning

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

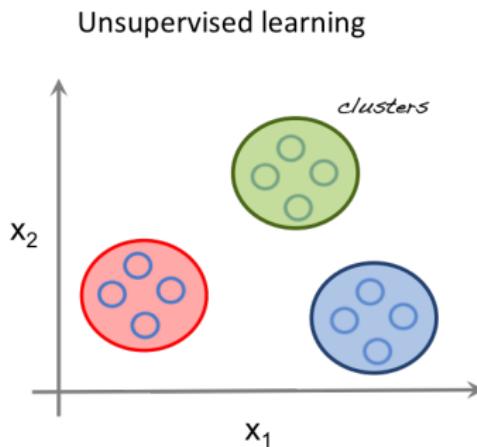
Types of ML

- Supervised
 - ▶ Classification
 - ▶ Regression
- Unsupervised



Types of ML

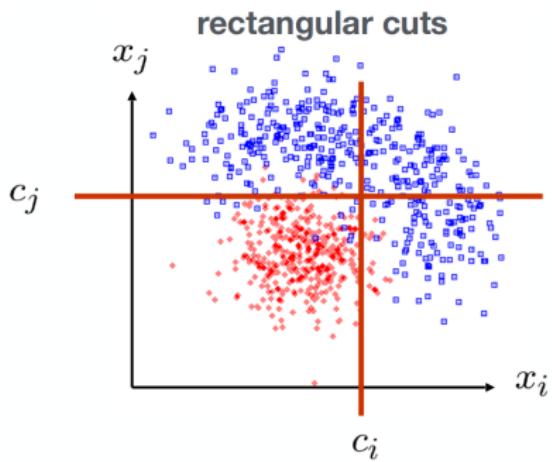
- Supervised
 - ▶ Classification
 - ▶ Regression
- Unsupervised



Rectangular cuts

Standard cut in one variable

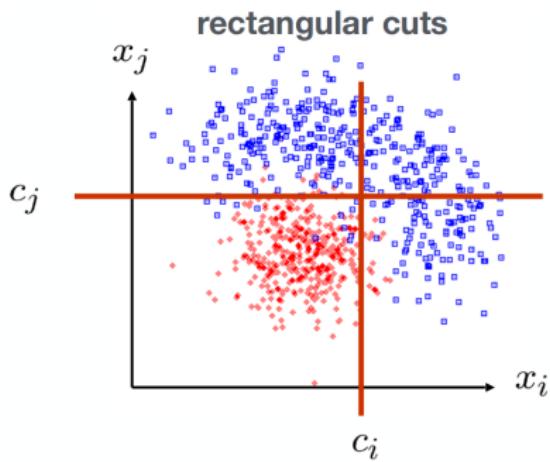
- Cuts only in lower-dimensional subspaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables
→ misclassification



Rectangular cuts

Standard cut in one variable

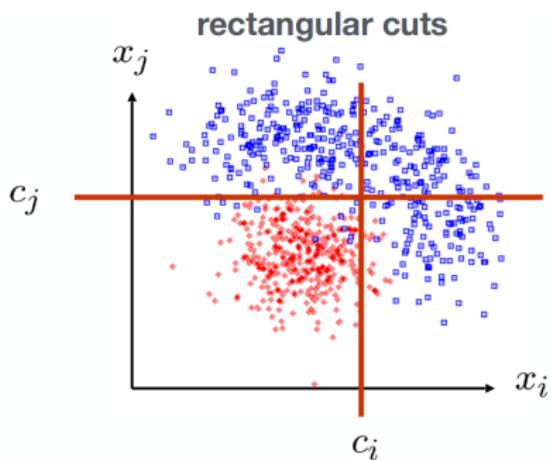
- Cuts only in lower-dimensional subspaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables
→ misclassification



Rectangular cuts

Standard cut in one variable

- Cuts only in lower-dimensional subspaces
- Ignores possible dependencies between the input variables
- Signal might behave like BG in several observables
→ misclassification

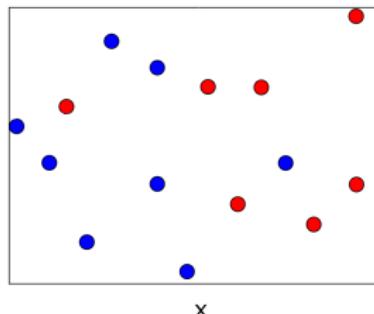


Rectangular cuts with *decision trees*

- Tree-like graph → flowchart
- Easy to understand
- Either be manually modelled by experts or learned from training data

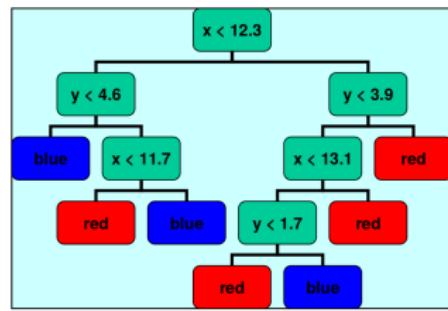
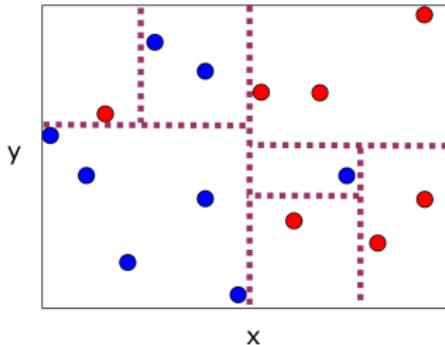
Decision tree learning

Feature space



Classification

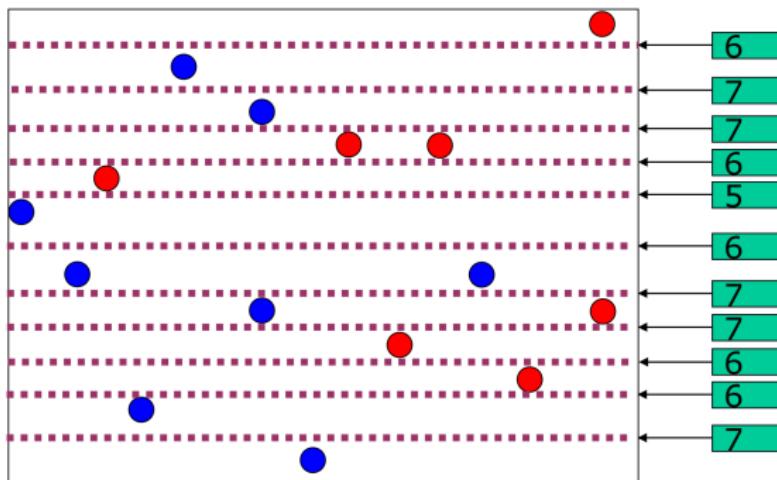
Decision tree



Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)

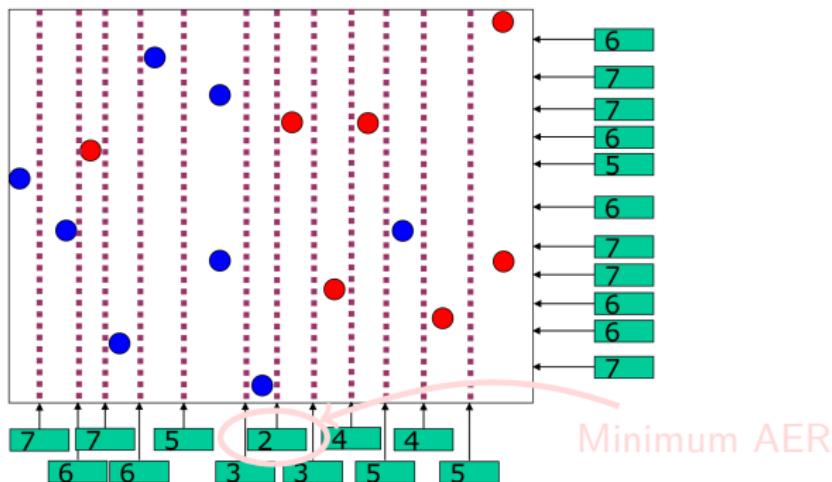
y split AER



Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)

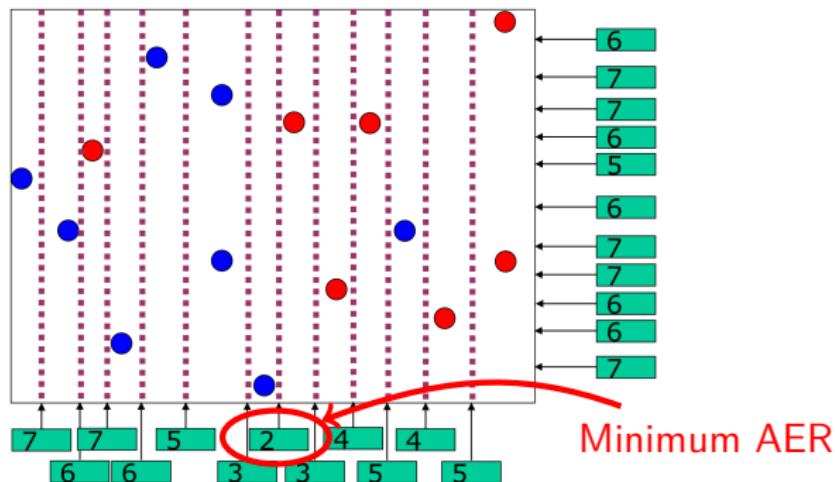
x-y split AER



Decision tree learning

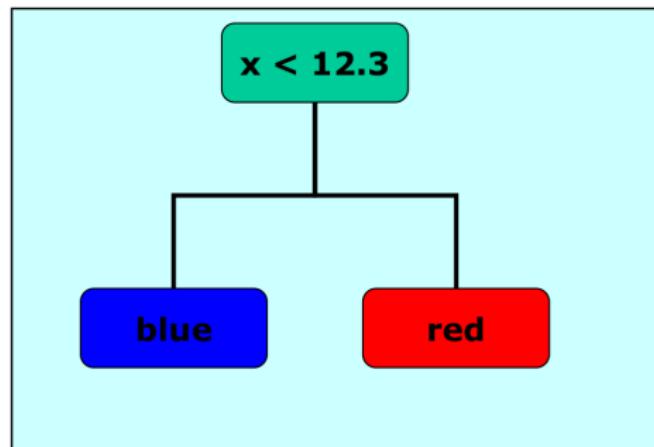
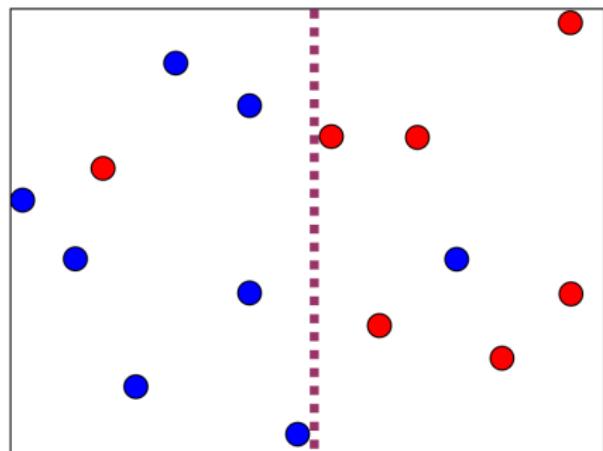
- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)

x-y split AER



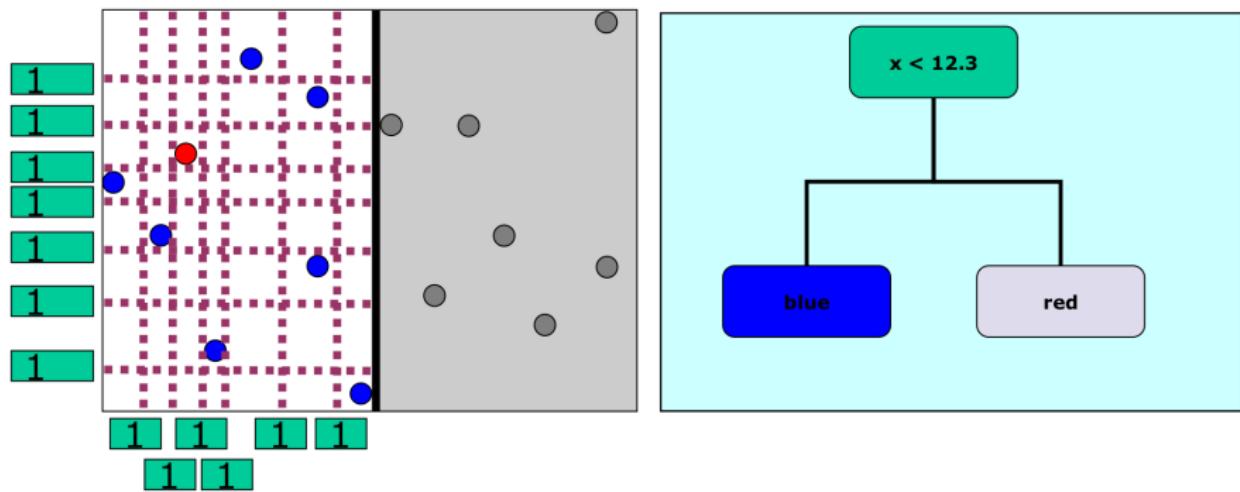
Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)
- 2) Recursively repeat step (1) for each subspace until AER → 0



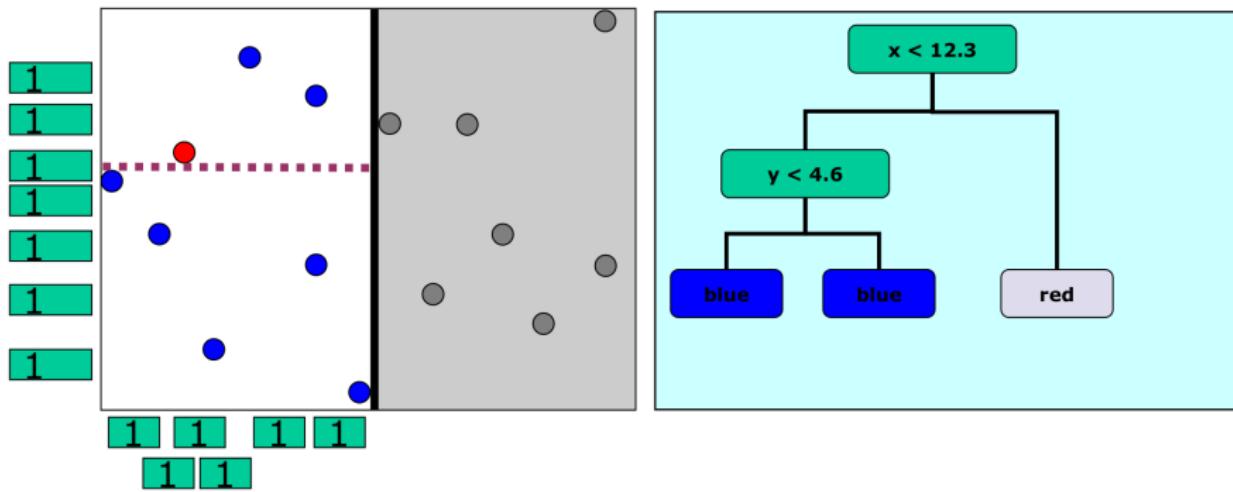
Decision tree learning

- 1) We compute a measure for each *possible split* in each feature
→ here **absolute error rate** (AER)
 - 2) Recursively repeat step (1) for each subspace until AER → 0



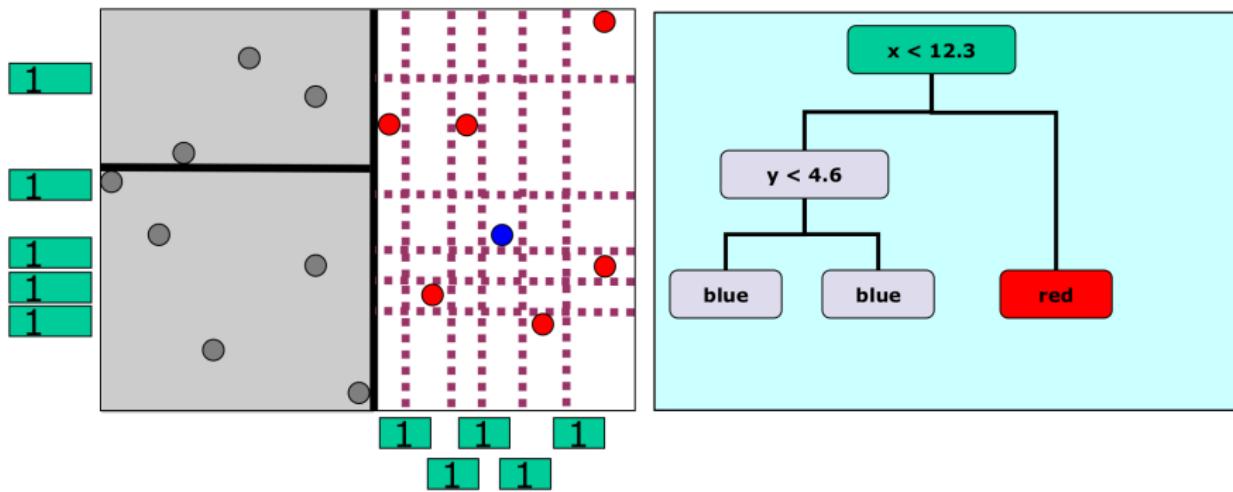
Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)
- 2) Recursively repeat step (1) for each subspace until AER → 0



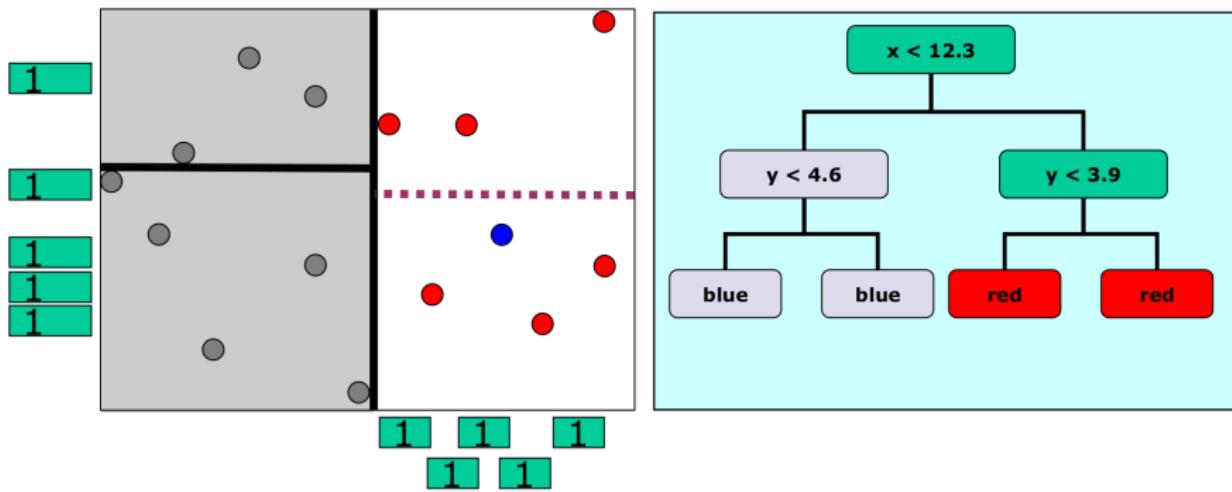
Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)
- 2) Recursively repeat step (1) for each subspace until AER → 0



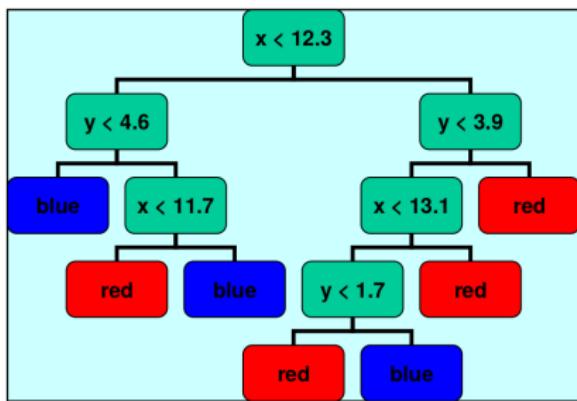
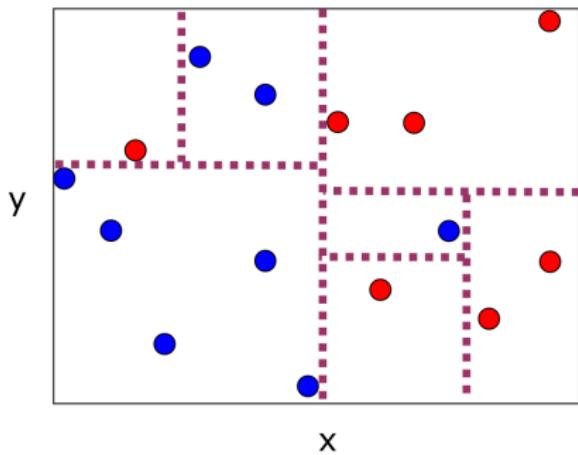
Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)
- 2) Recursively repeat step (1) for each subspace until $\text{AER} \rightarrow 0$



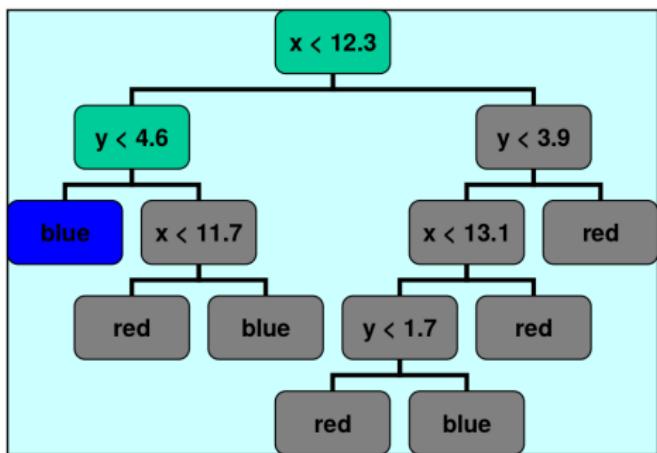
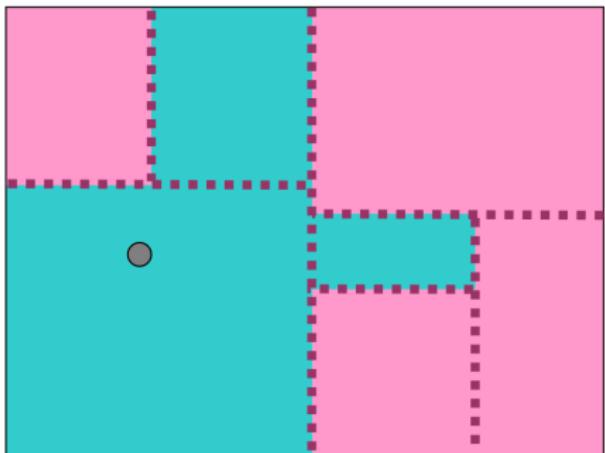
Decision tree learning

- 1) We compute a measure for *each possible split* in each feature
→ here **absolute error rate** (AER)
- 2) Recursively repeat step (1) for each subspace until $\text{AER} \rightarrow 0$



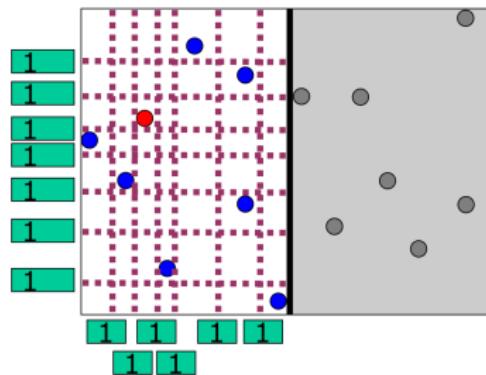
Decision tree classification

3) Classification

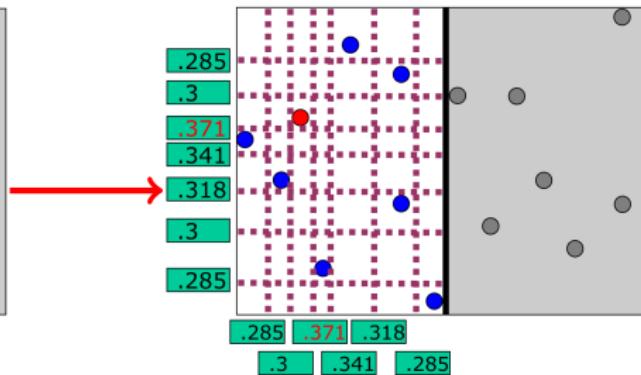


Decision tree improvements I

- Use more sophisticated split measures
 - ▶ *Information gain* \leftrightarrow (im-)purity of splitted sub-sets
- Pruning
- Ensemble trees (Random forest & boosted DT)



Absolute error rate

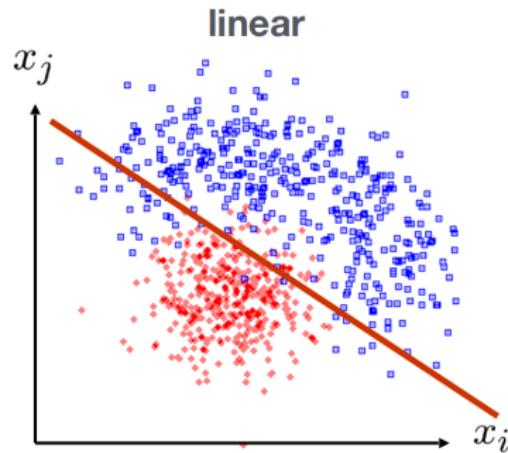


Information gain

Linear cuts

More degrees of freedom than rectangular cut

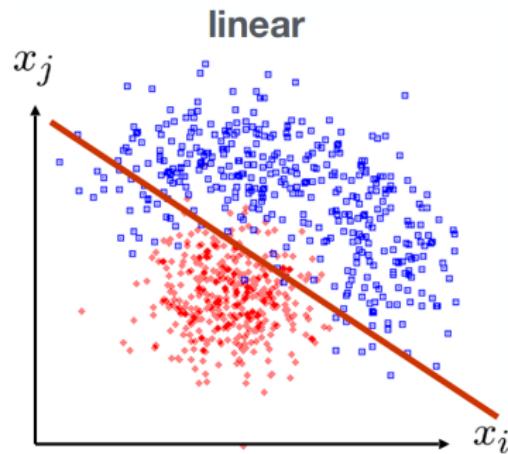
- Simple white box methods
- Very fast classification
- Can become very powerful by using *kernel trick*



Linear cuts

More degrees of freedom than rectangular cut

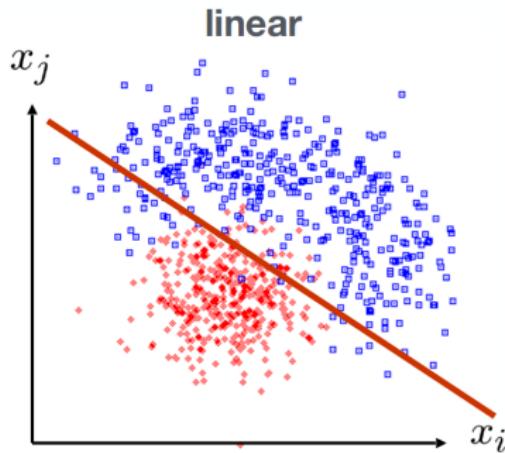
- Simple white box methods
- Very fast classification
- Can become very powerful by using *kernel trick*



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 $\mathbf{x} = (x_1, \dots, x_n)$ to base its decision on.

$$y : \mathbb{R}^n \rightarrow \mathbb{R} \mid \mathbf{x} \mapsto y = f(\mathbf{w} \cdot \mathbf{x}) = f\left(\sum_j w_j x_j\right)$$

\mathbf{w} ... weight vector

Simplest case

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

→ Function can be approximated by **single layer perceptron**

Linear models

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

$$y : \mathbb{R}^n \rightarrow \mathbb{R} \mid \mathbf{x} \mapsto y = f(\mathbf{w} \cdot \mathbf{x}) = f\left(\sum_j w_j x_j\right)$$

\mathbf{w} ... weight vector

Simplest case

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

→ Function can be approximated by **single layer perceptron**

Linear models

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

$$y : \mathbb{R}^n \rightarrow \mathbb{R} \mid \mathbf{x} \mapsto y = f(\mathbf{w} \cdot \mathbf{x}) = f\left(\sum_j w_j x_j\right)$$

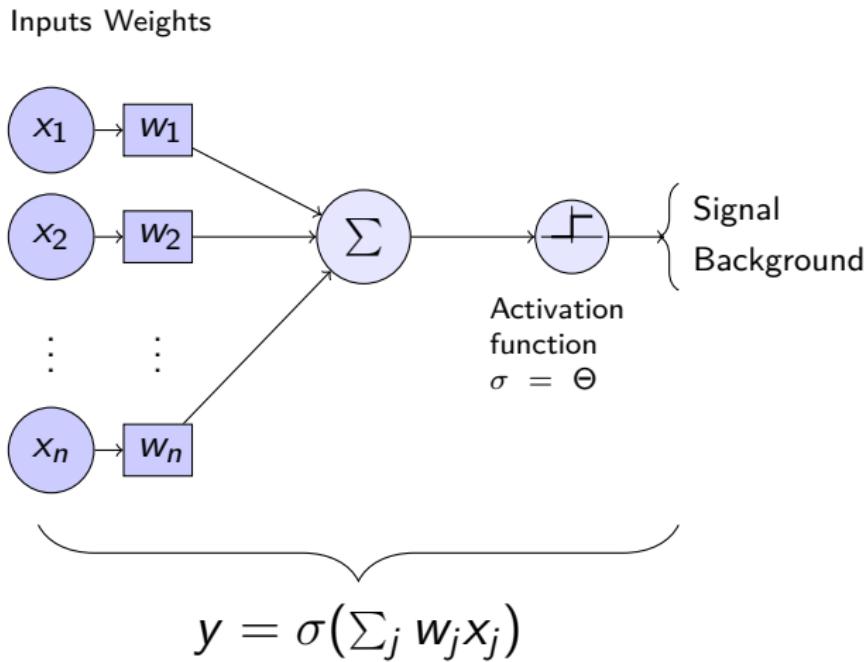
\mathbf{w} ... weight vector

Simplest case

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

→ Function can be approximated by **single layer perceptron**

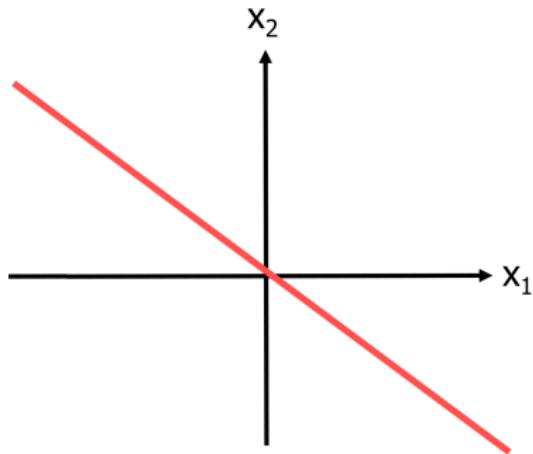
Single layer perceptron (SLP)



SLP training

Algorithm

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

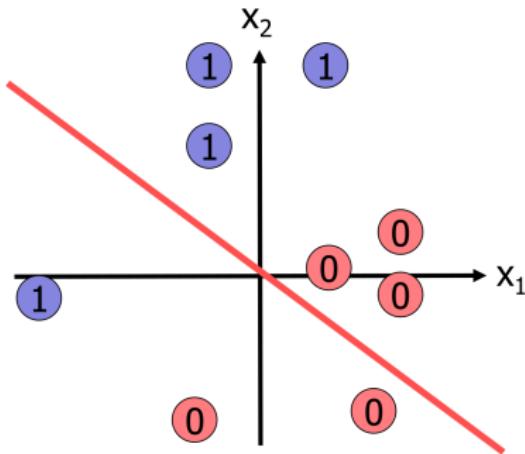


$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

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



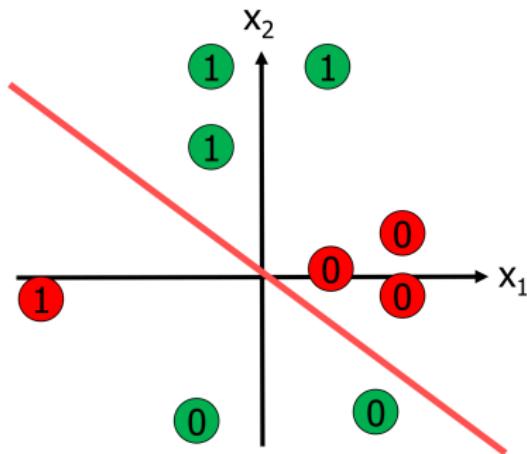
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

- ➊ Initialize weights \mathbf{w}
- ➋ Repeat until $y_{predict} = y_{target}$:
 - ➌ Present training sample \mathbf{x}
 - ➍ Predict sample label
 $y = \Theta(\mathbf{x}\mathbf{w}_{init})$ and compute error $\Delta = y_{target} - y$
 - ➎ If $\Delta \neq 0 \rightarrow$ update weights
 $\mathbf{w}' = \mathbf{w} + \alpha \Delta \mathbf{x}$

$\alpha \dots$ learning rate



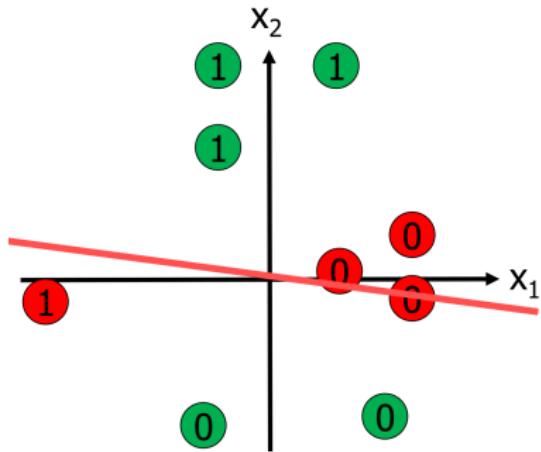
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

- ➊ Initialize weights \mathbf{w}
- ➋ Repeat until $y_{predict} = y_{target}$:
 - ➌ Present training sample \mathbf{x}
 - ➍ Predict sample label
 $y = \Theta(\mathbf{x}\mathbf{w}_{init})$ and compute
 error $\Delta = y_{target} - y$
 - ➎ If $\Delta \neq 0 \rightarrow$ update weights
 $\mathbf{w}' = \mathbf{w} + \alpha \Delta \mathbf{x}$

$\alpha \dots$ learning rate



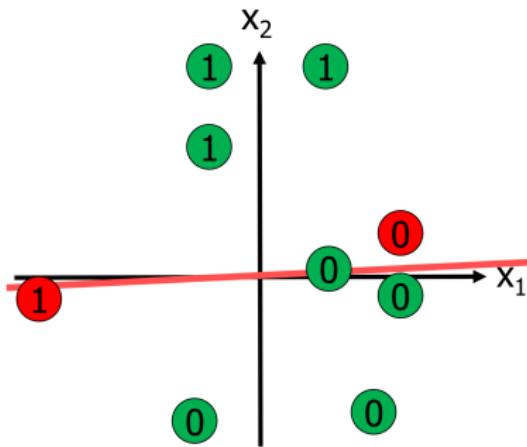
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

- ➊ Initialize weights \mathbf{w}
- ➋ Repeat until $y_{predict} = y_{target}$:
 - ➌ Present training sample \mathbf{x}
 - ➍ Predict sample label
 $y = \Theta(\mathbf{x}\mathbf{w}_{init})$ and compute
 error $\Delta = y_{target} - y$
 - ➎ If $\Delta \neq 0 \rightarrow$ update weights
 $\mathbf{w}' = \mathbf{w} + \alpha \Delta \mathbf{x}$

$\alpha \dots$ learning rate



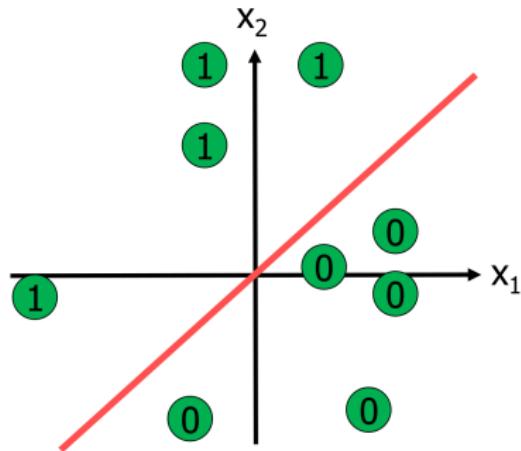
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

- ➊ Initialize weights \mathbf{w}
- ➋ Repeat until $y_{predict} = y_{target}$:
 - ➌ Present training sample \mathbf{x}
 - ➍ Predict sample label
 $y = \Theta(\mathbf{x}\mathbf{w}_{init})$ and compute
 error $\Delta = y_{target} - y$
 - ➎ If $\Delta \neq 0 \rightarrow$ update weights
 $\mathbf{w}' = \mathbf{w} + \alpha \Delta \mathbf{x}$

$\alpha \dots$ learning rate



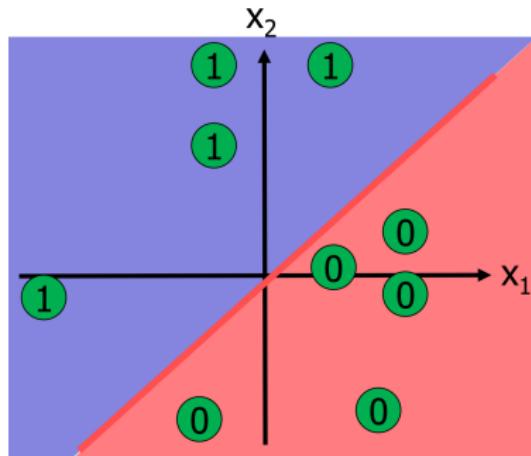
$$y = w_1 x_1 + w_2 x_2$$

SLP training

Algorithm

- ➊ Initialize weights \mathbf{w}
- ➋ Repeat until $y_{predict} = y_{target}$:
 - ➌ Present training sample \mathbf{x}
 - ➍ Predict sample label
 $y = \Theta(\mathbf{x}\mathbf{w}_{init})$ and compute error $\Delta = y_{target} - y$
 - ➎ If $\Delta \neq 0 \rightarrow$ update weights
 $\mathbf{w}' = \mathbf{w} + \alpha \Delta \mathbf{x}$

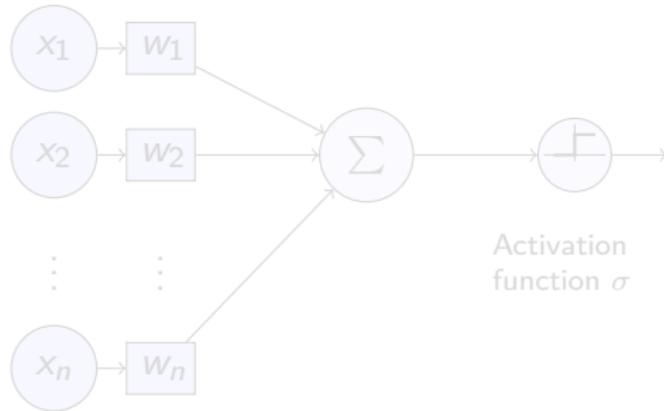
$\alpha \dots$ learning rate



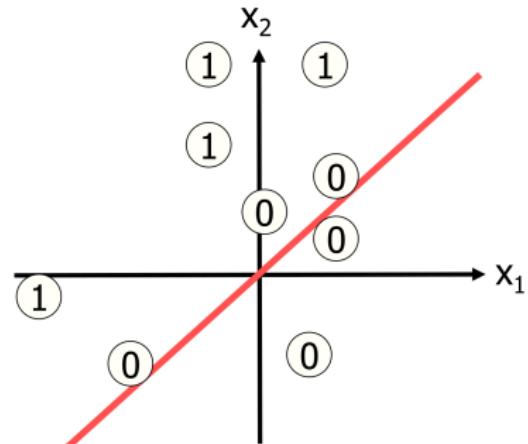
$$y = w_1 x_1 + w_2 x_2$$

SLP bias term

Inputs Weights

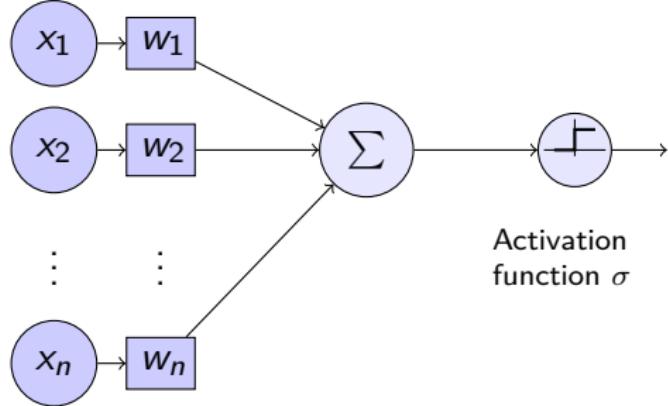


Activation
function σ

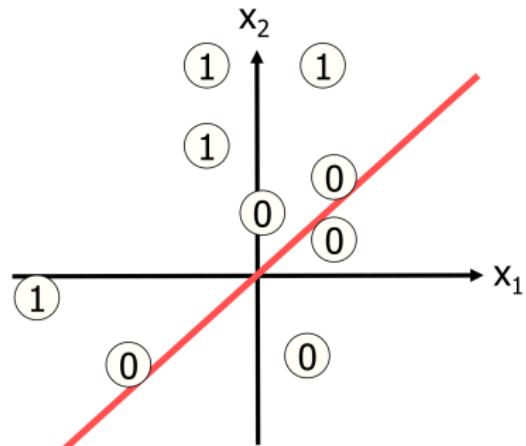


SLP bias term

Inputs Weights

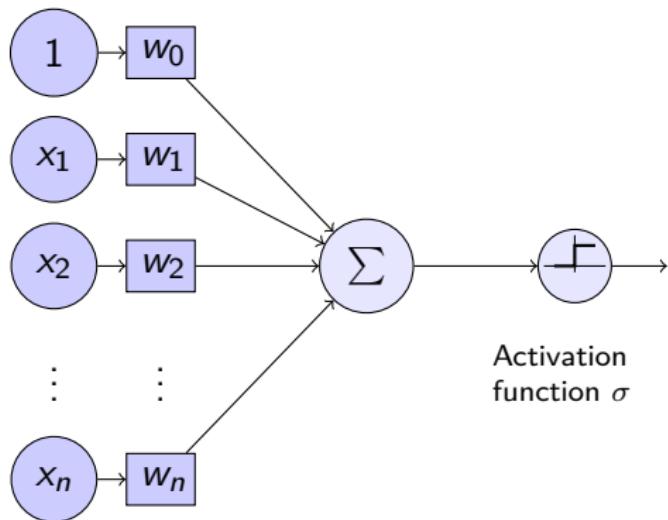


Activation
function σ

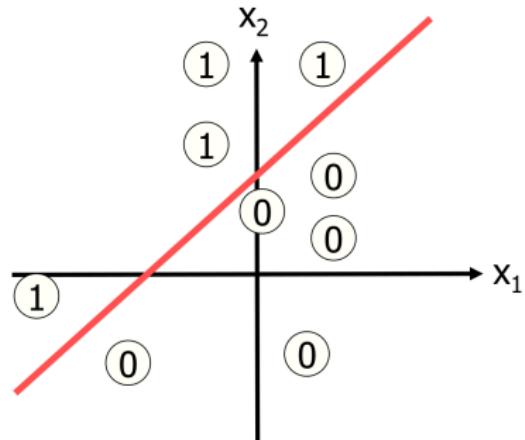


SLP bias term

Inputs Weights

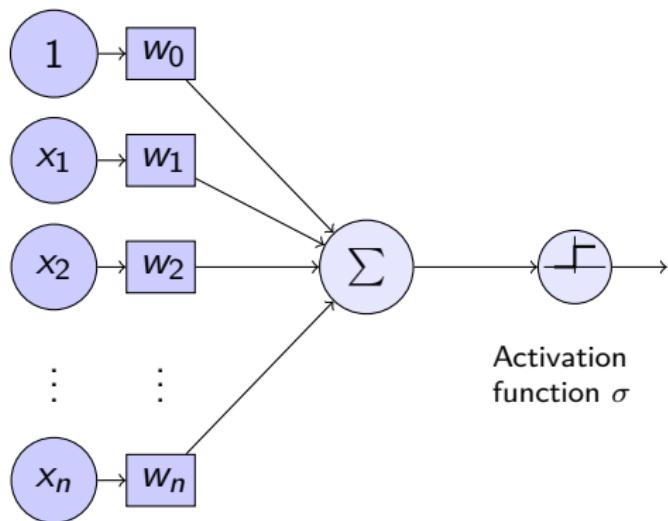


Activation
function σ

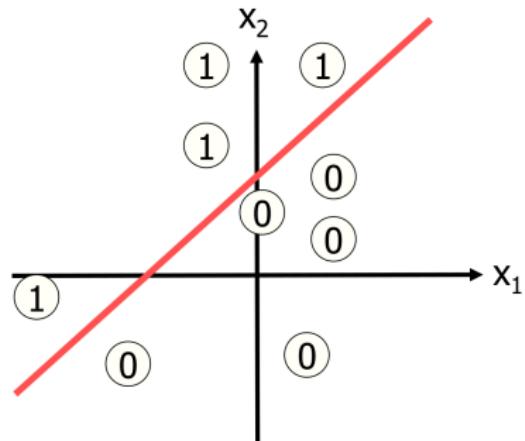


SLP bias term

Inputs Weights

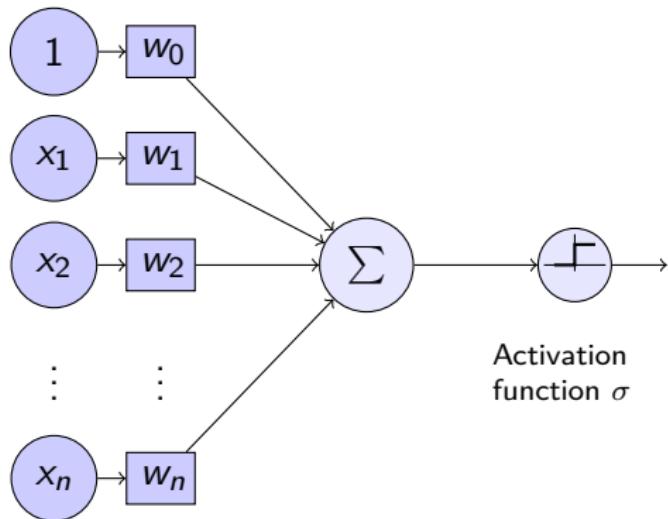


Activation
function σ

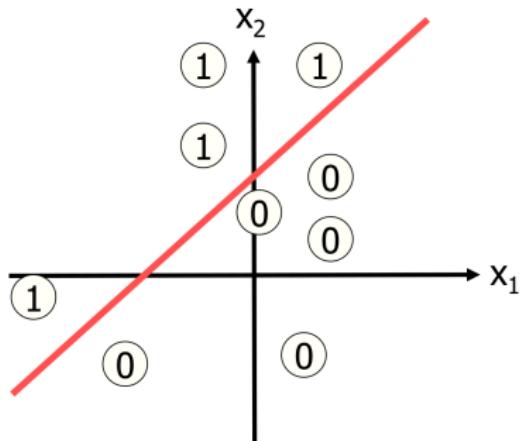


SLP bias term

Inputs Weights



Activation
function σ

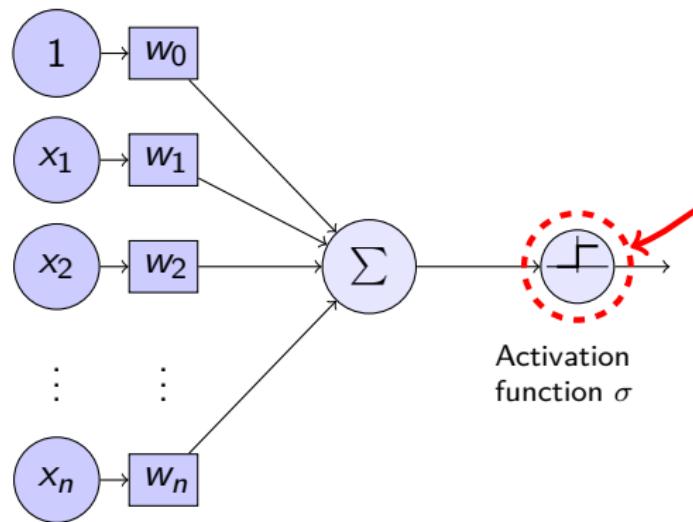


Bias weight w_0 learned just as the other weights

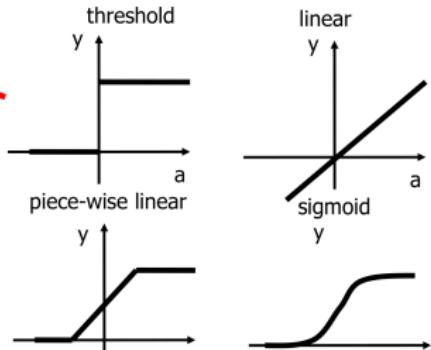
$$y = \sigma(w_0 + \sum_{j=1}^n w_j x_j)$$

SLP activation functions

Inputs Weights

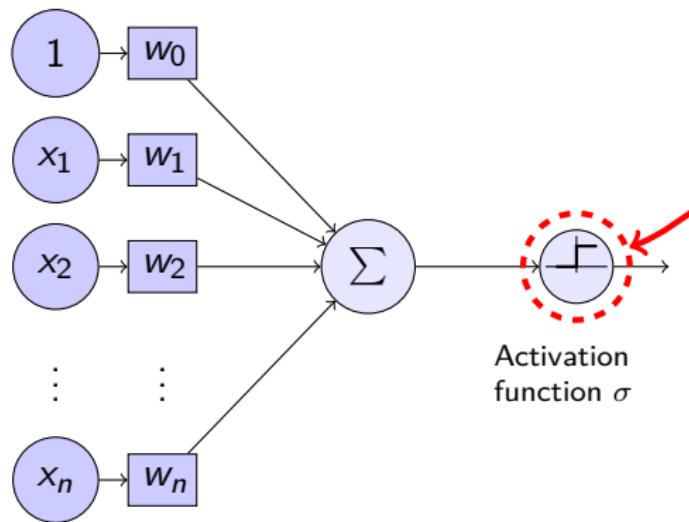


Activation
function σ

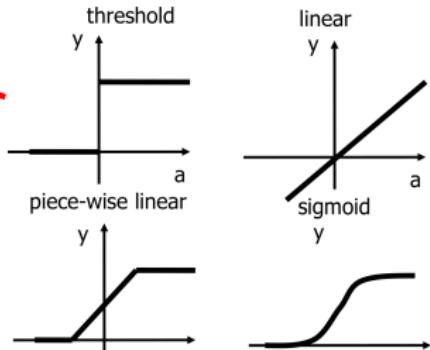


SLP activation functions

Inputs Weights



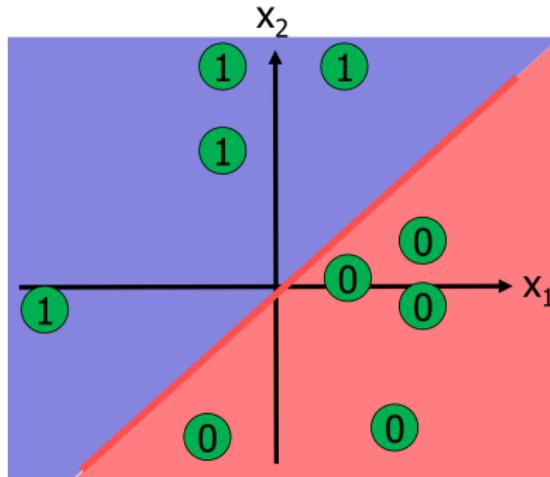
Activation
function σ



Often used: $\text{sigmoid} \rightarrow \text{output} \in (0, 1)$

Non-linear activation functions: output

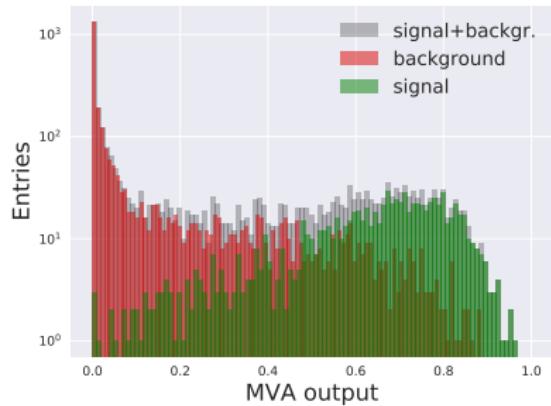
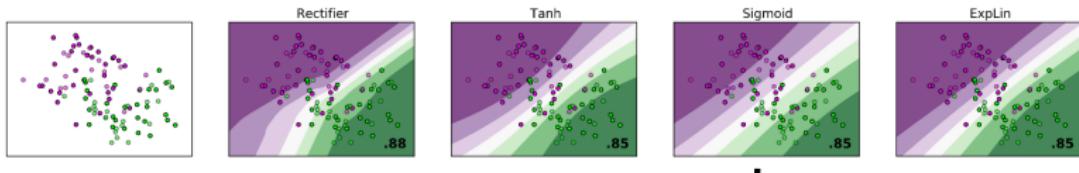
Distance to decision boundary comes into play



MVA output $y = 0 \text{ or } 1$

Non-linear activation functions: output

Distance to decision boundary comes into play



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

Improving linear methods

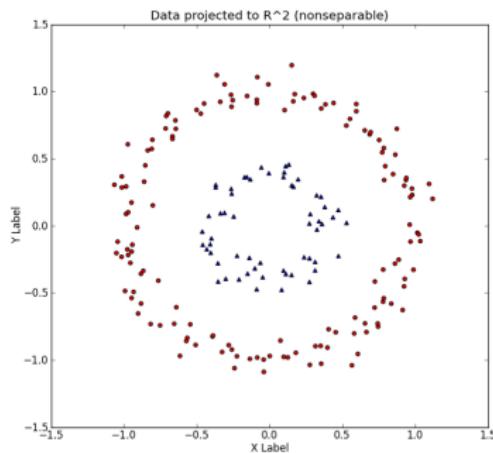
Kernel trick

Map data to a higher dimensional space where linear hyperplane can again be found

Improving linear methods

Kernel trick

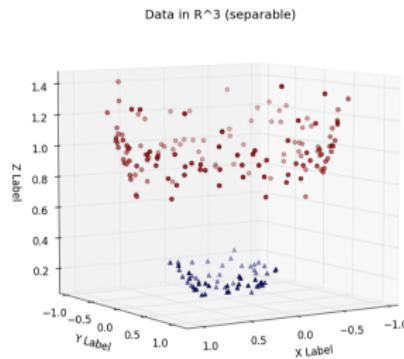
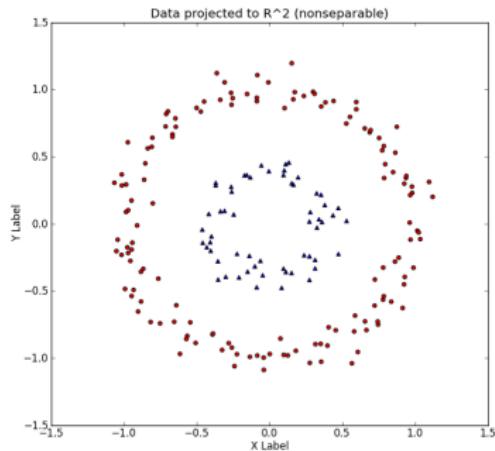
Map data to a higher dimensional space where linear hyperplane can again be found



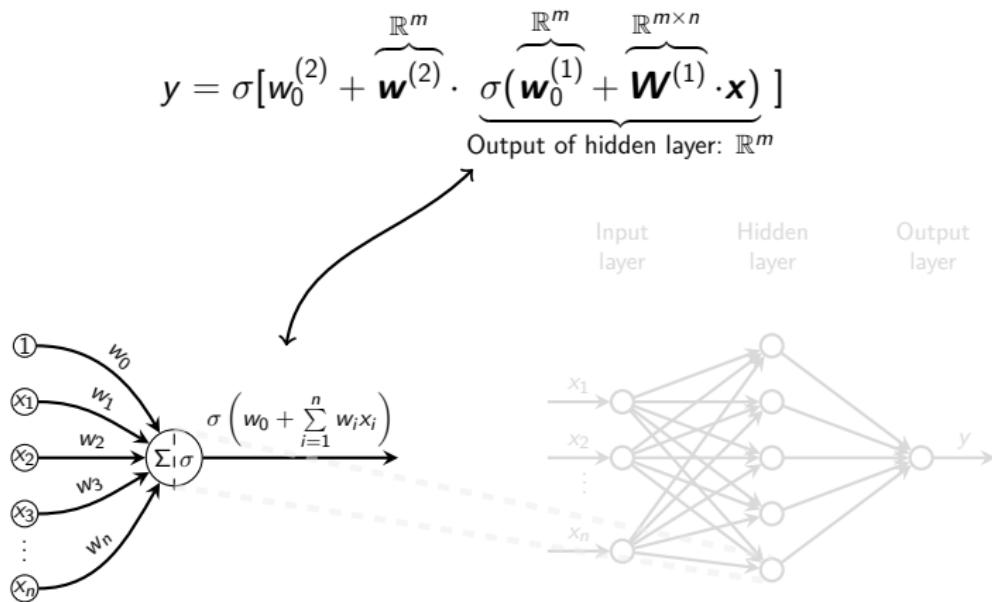
Improving linear methods

Kernel trick

Map data to a higher dimensional space where linear hyperplane can again be found

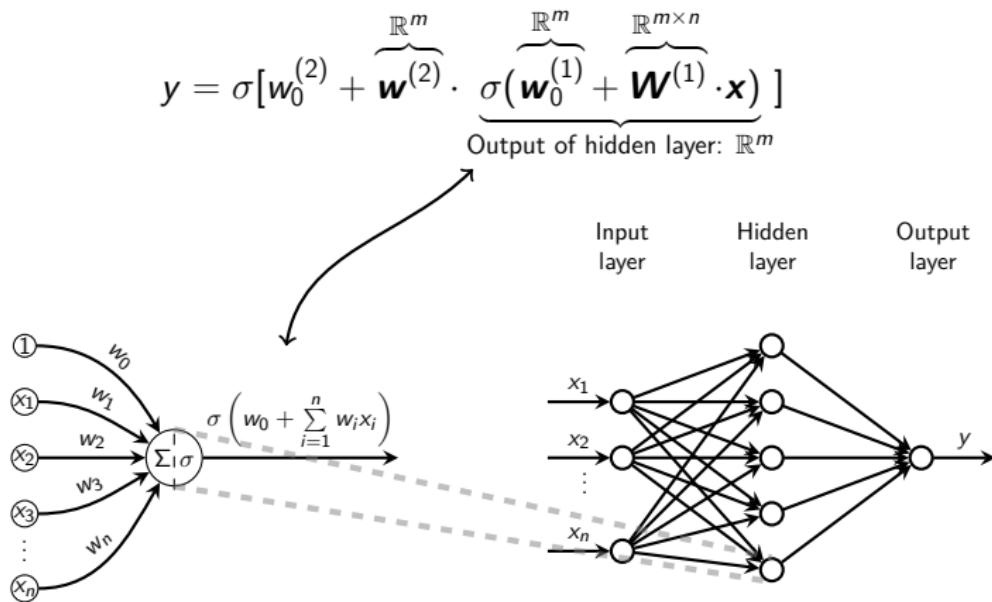


Non-linear models



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

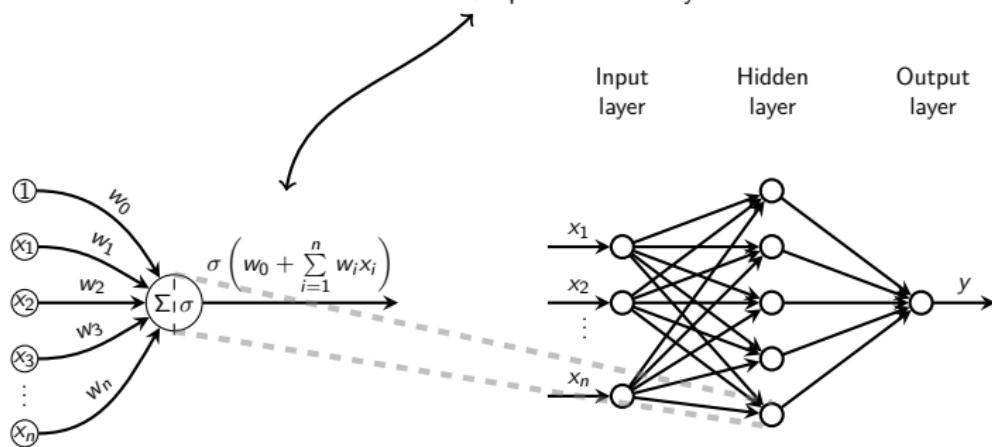
Non-linear models



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

Non-linear models

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

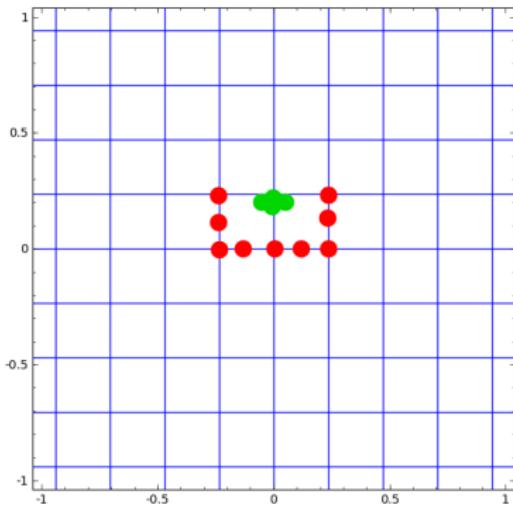
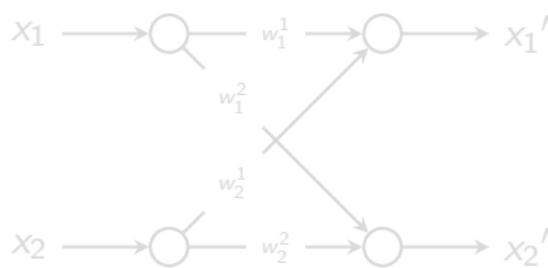


→ The hidden layer *learns* a representation so that the data is linearly separable

Visualising the hidden layer

- ① Linear transformation: Wx
- ② Translation: w_0
- ③ Non-linear activation: σ

$$\tilde{x} = x$$



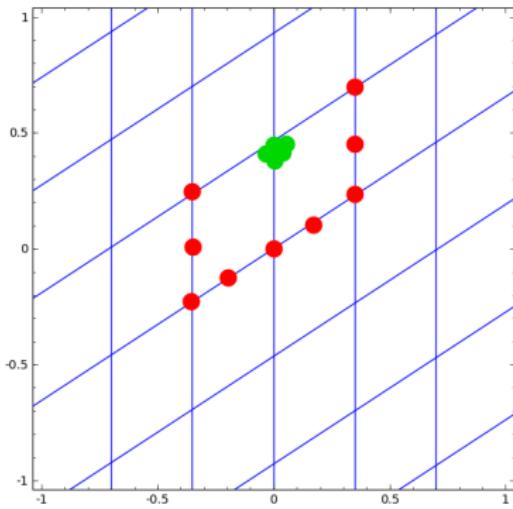
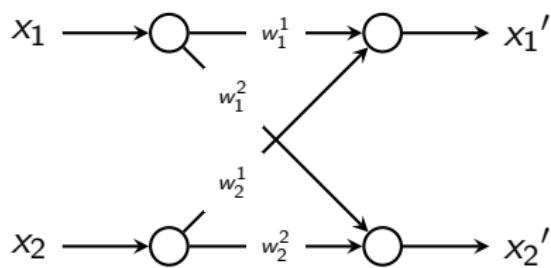
Visualising the hidden layer

① Linear transformation: $\mathbf{W}\mathbf{x}$

② Translation: \mathbf{w}_0

③ Non-linear activation: σ

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



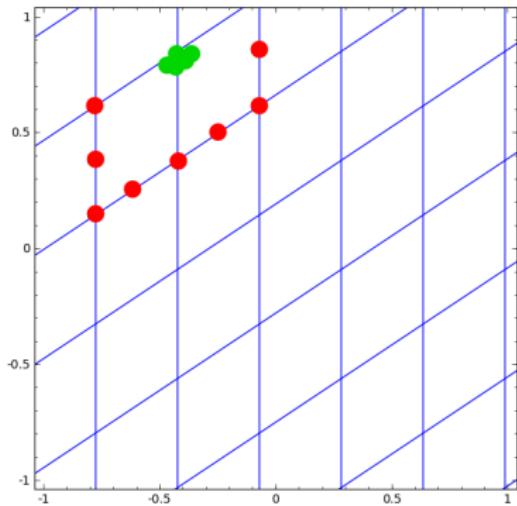
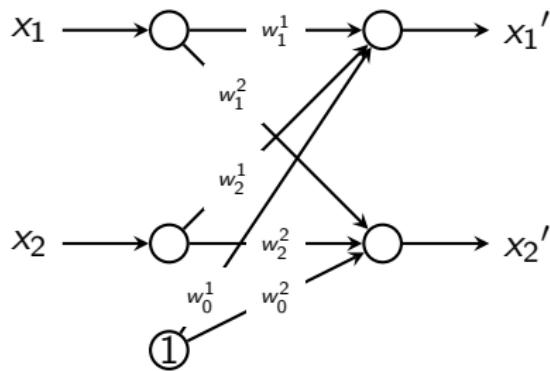
Visualising the hidden layer

① Linear transformation: \mathbf{Wx}

② Translation: \mathbf{w}_0

③ Non-linear activation: σ

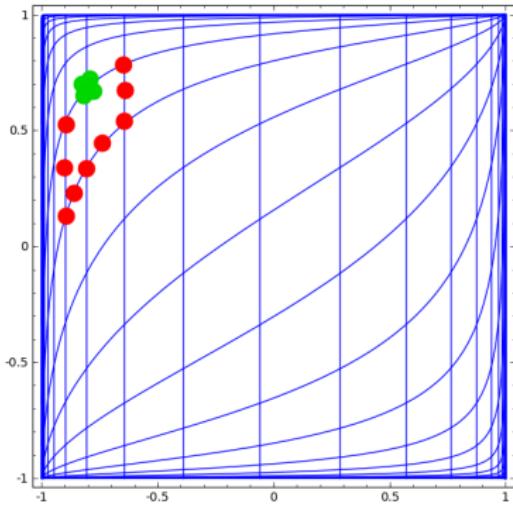
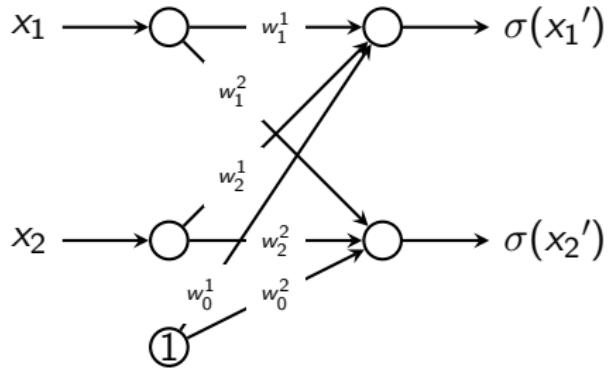
$$\tilde{\mathbf{x}} = (\mathbf{w}_0^{(1)} + \mathbf{W}^{(1)} \cdot \mathbf{x})$$



Visualising the hidden layer

- ➊ Linear transformation: \mathbf{Wx}
- ➋ Translation: \mathbf{w}_0
- ➌ Non-linear activation: σ

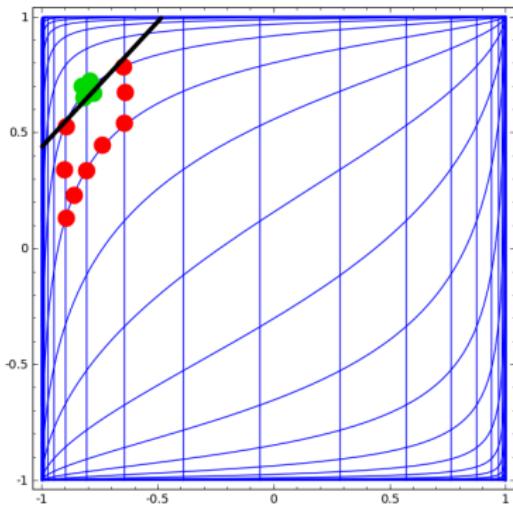
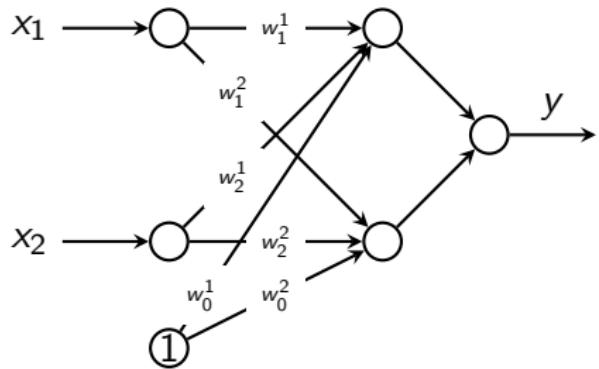
$$\tilde{\mathbf{x}} = \sigma(\mathbf{w}_0^{(1)} + \mathbf{W}^{(1)} \cdot \mathbf{x})$$



Visualising the hidden layer

- ➊ Linear transformation: \mathbf{Wx}
- ➋ Translation: \mathbf{w}_0
- ➌ Non-linear activation: σ

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



Update weights via back-propagation

Backpropagation

Adapt weights by going backwards in the network

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

Update weights via back-propagation

Backpropagation

Adapt weights by going backwards in the network

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

Update weights via back-propagation

Backpropagation

Adapt weights by going backwards in the network

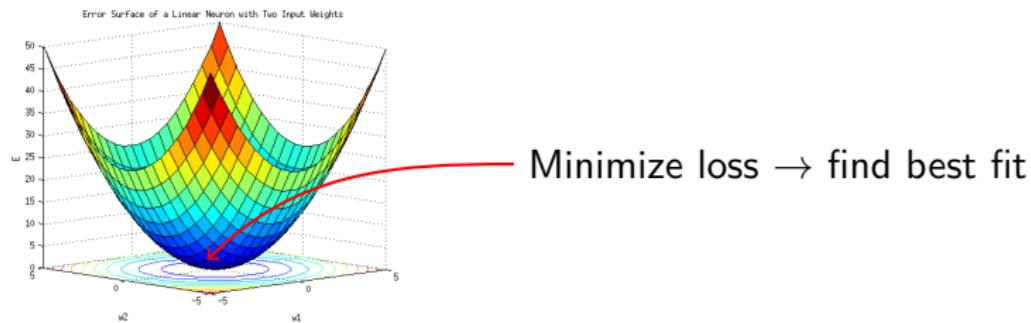
- Initialize weights
- Evaluate $y_{predict} = f(\mathbf{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss
$$L = L(y_{true}, \mathbf{w}^{(1)}, \mathbf{W}^{(2)})$$

Update weights via back-propagation

Backpropagation

Adapt weights by going backwards in the network

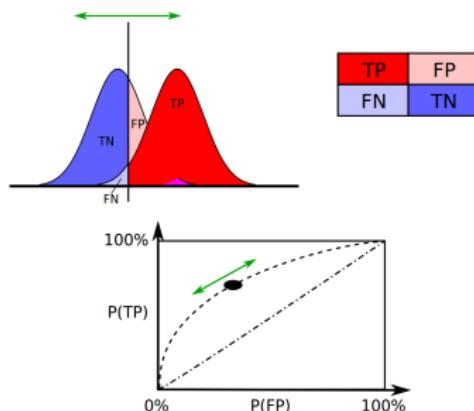
- Initialize weights
- Evaluate $y_{predict} = f(\mathbf{x})$ and calculate $\Delta = (y_{true} - y_{predict})$
- Update weights: Weight adaption often denoted via loss
$$L = L(y_{true}, \mathbf{w}^{(1)}, \mathbf{W}^{(2)})$$



Quantify MVA output

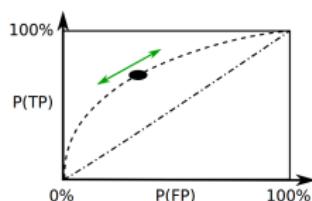
Evaluate classifier performance

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



$$\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$$

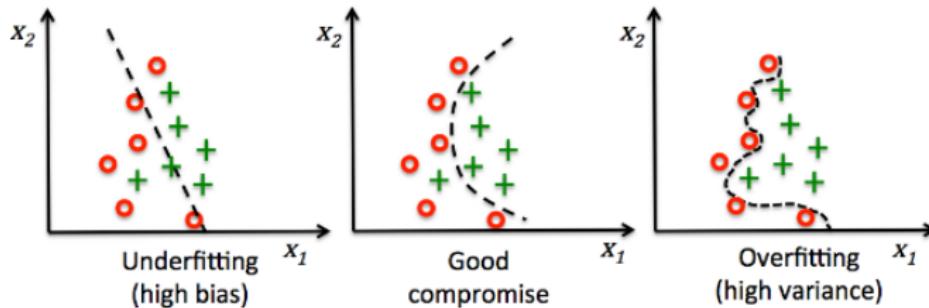
$$\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$$



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

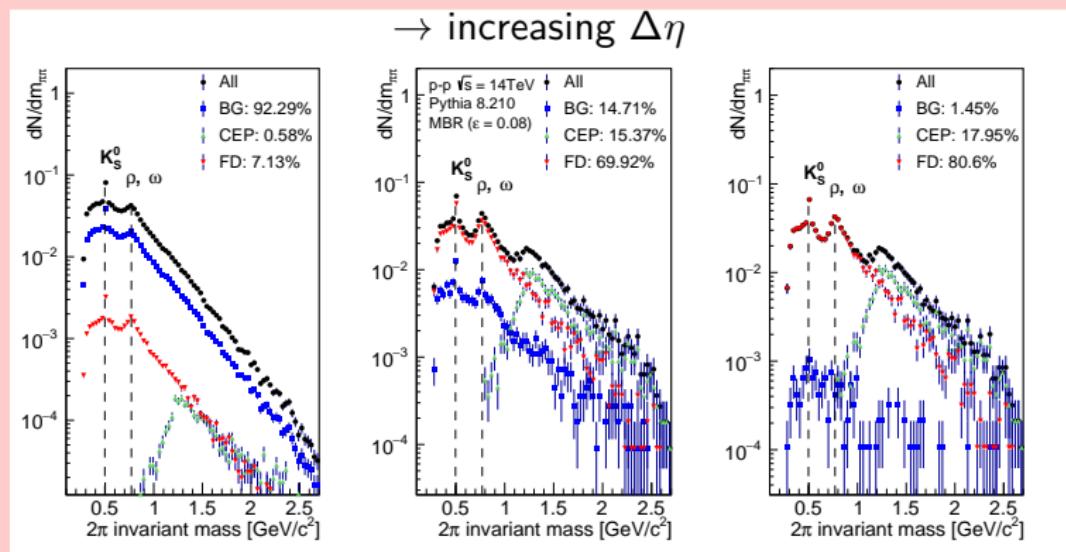
Bias-variance tradeoff

- Small variance: Classifiers with low degrees of freedom are less prone to statistical fluctuations
→ 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

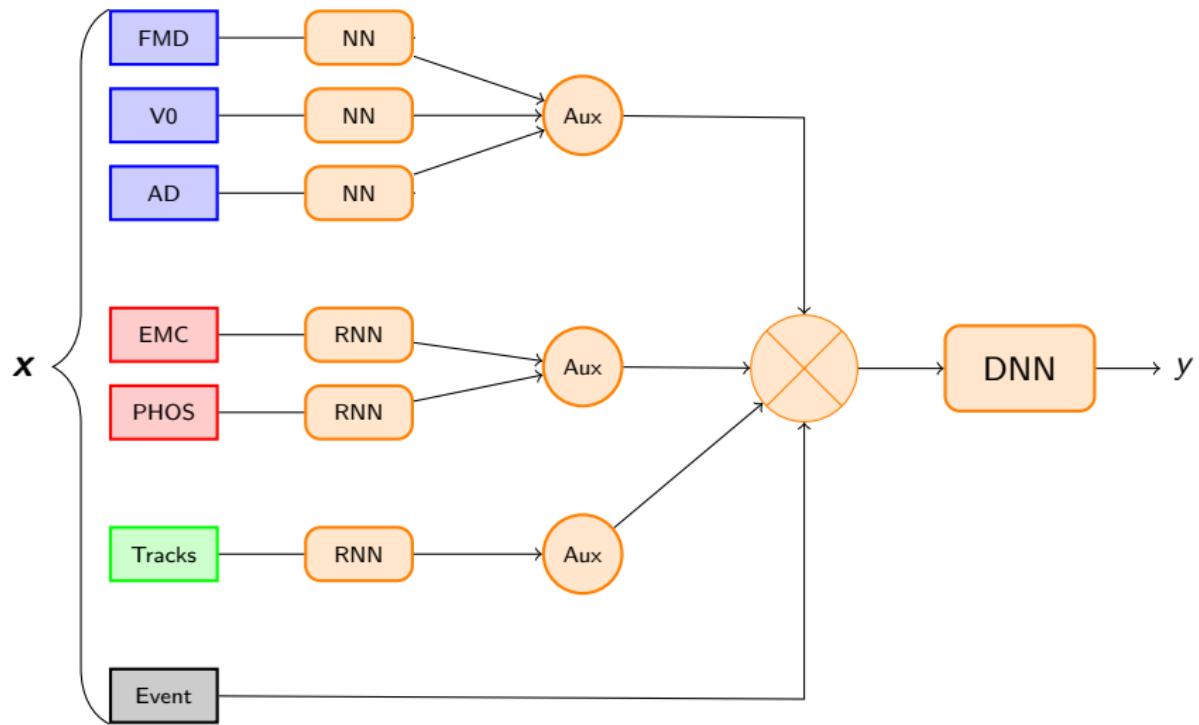


MVA applied to CEP data

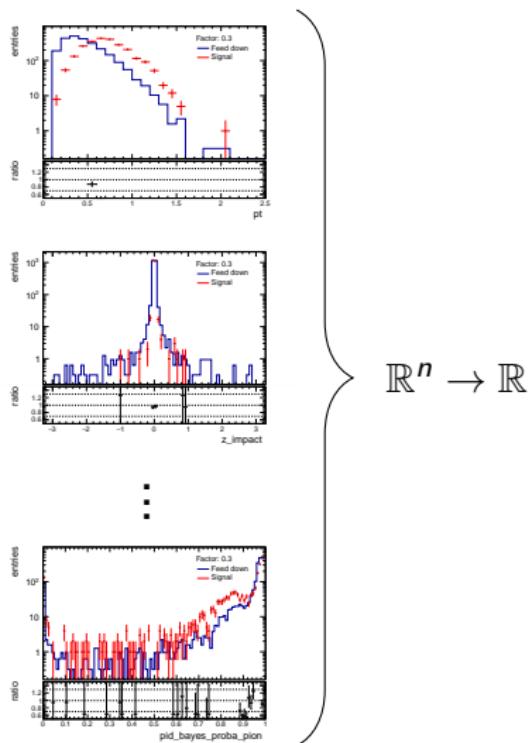
In order to reduce the main background component - **feed down** - ML is applied on simulated data.



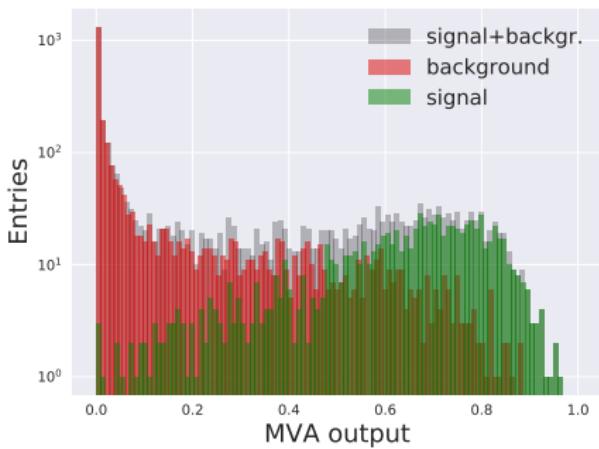
Neural network structure



Results



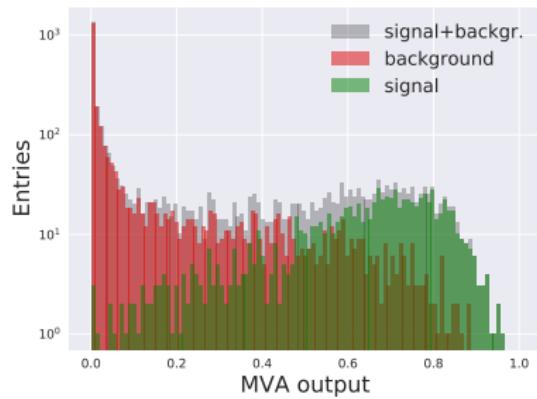
$$\mathbb{R}^n \rightarrow \mathbb{R}$$



Results

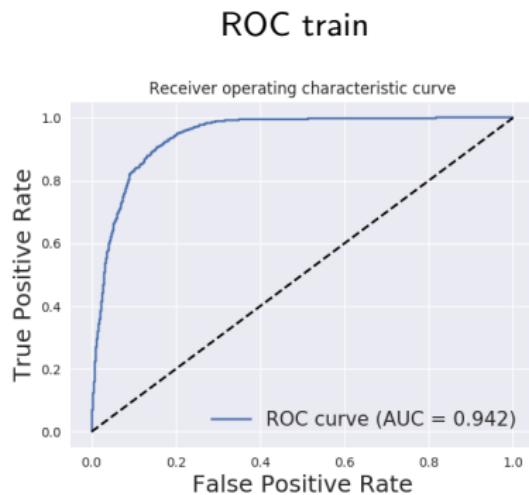
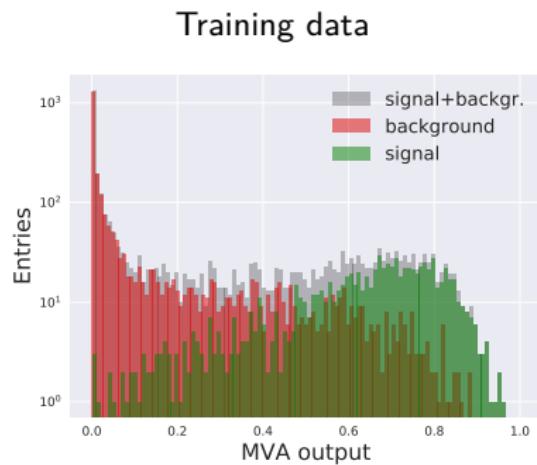
ROC curve

Training data



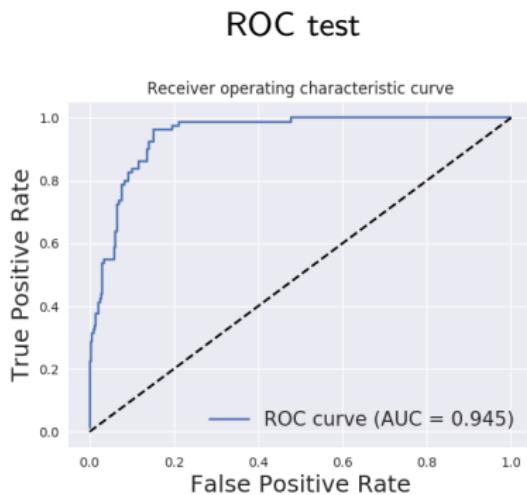
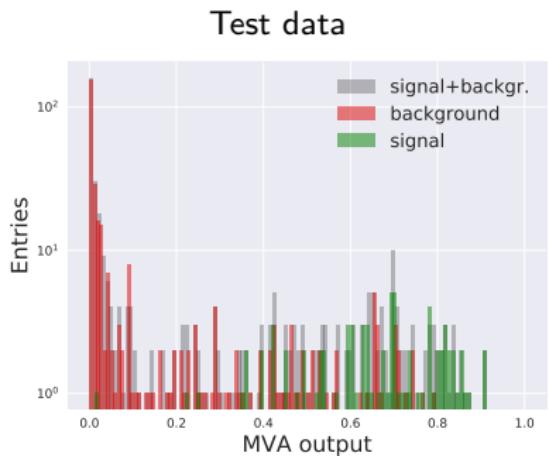
Results

ROC curve



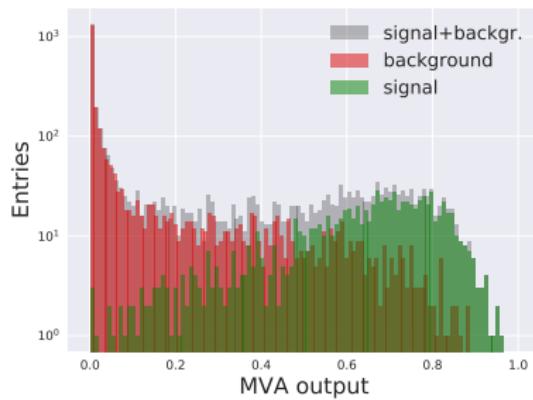
Results

ROC curve



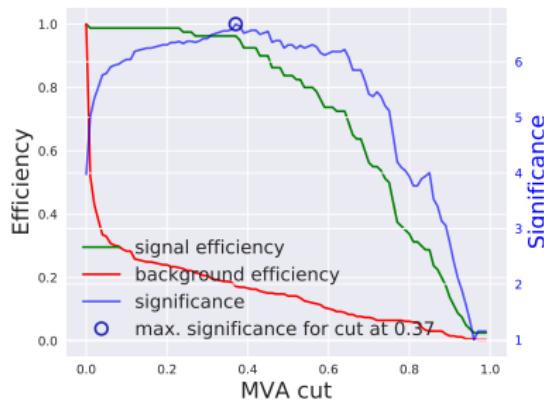
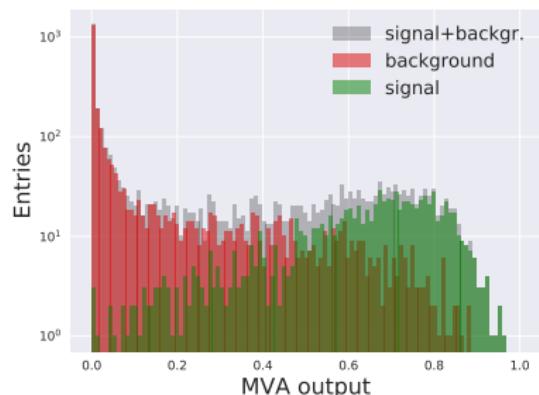
Results

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



Results

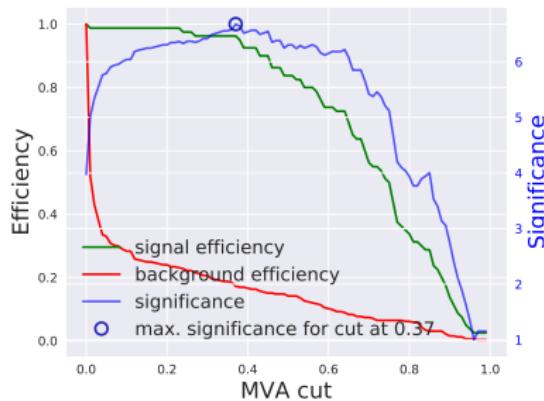
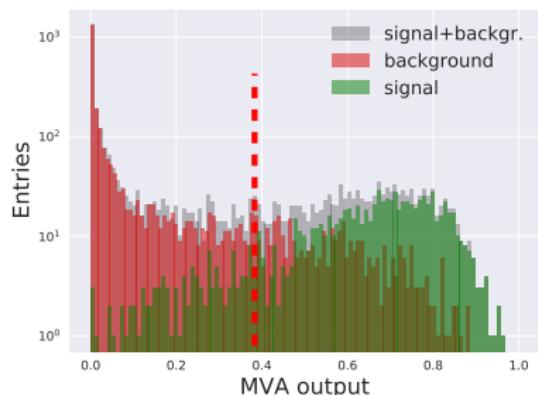
To find the optimal cut on MVA output we evaluate significance along y



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

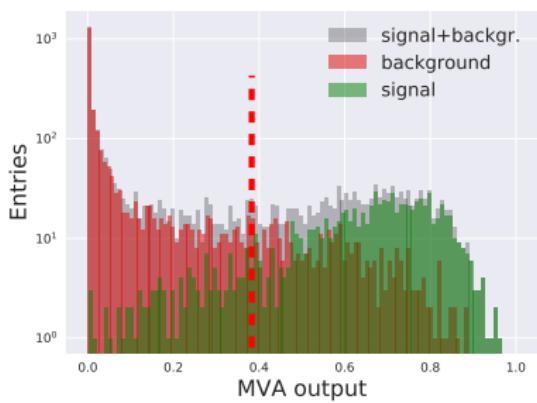
Results

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

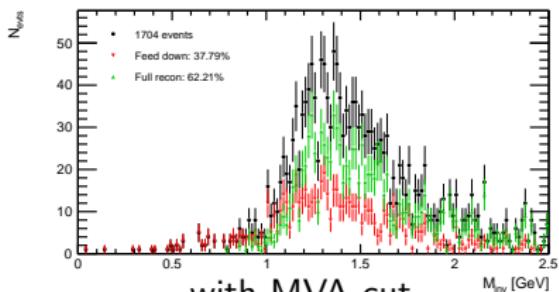
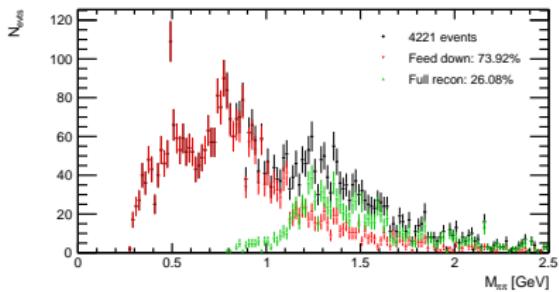


$$\text{Significance } S = \frac{N_{sig}}{\sqrt{N_{sig} + N_{BG}}}$$

Results



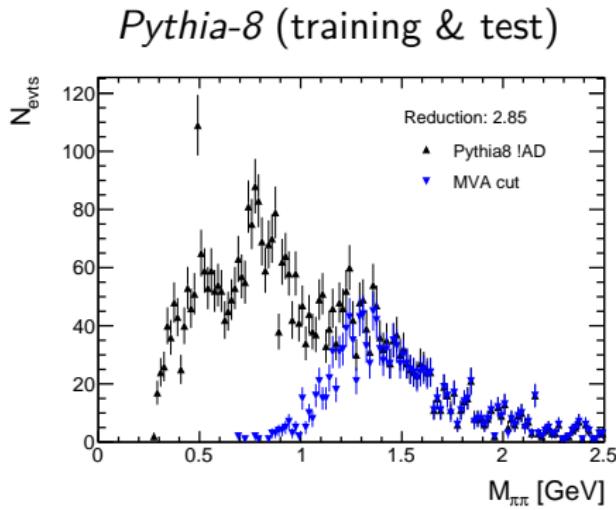
no MVA cut



with MVA cut

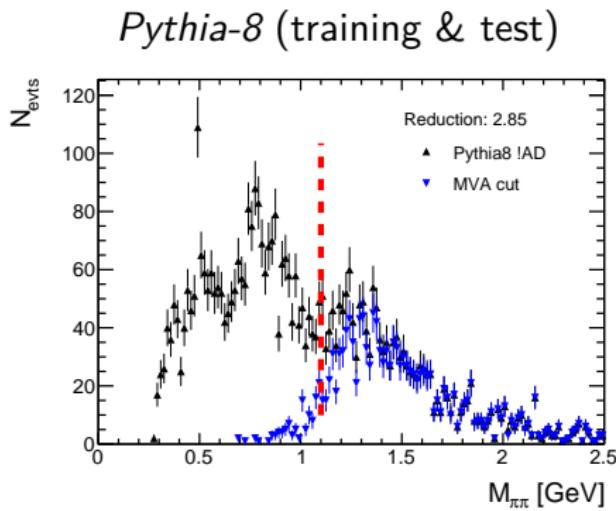
Results

Spectrum comparison yields MVA cut \leftrightarrow mass cut



Results

Spectrum comparison yields MVA cut \leftrightarrow mass cut



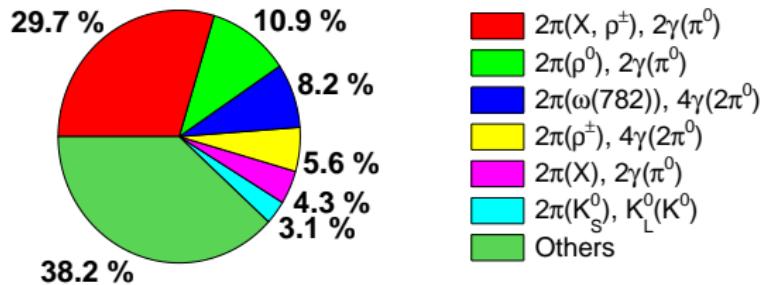
Challenges

- 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
→ however, they should!
- EMCAL correction framework should (hopefully) fix that

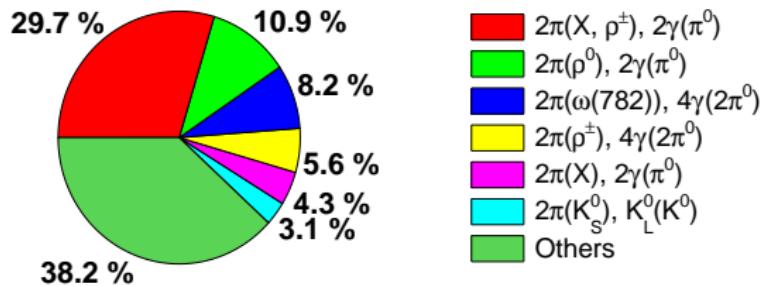
Feed down contributions



EMCal corrections

- Currently: emcal data are not making a difference
→ 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 → needs further steps on real data