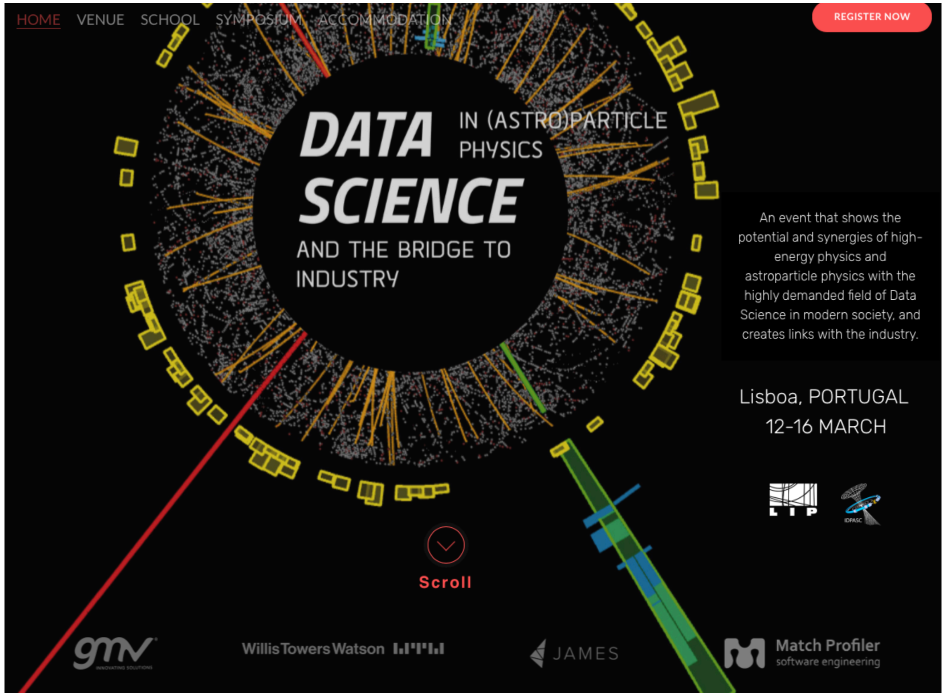
# Machine Learning




*Lorenzo Moneta*
*CERN - EP-SFT*
Lorenzo.Moneta@cern.ch

LIP Data Science School / 12-14 March 2018

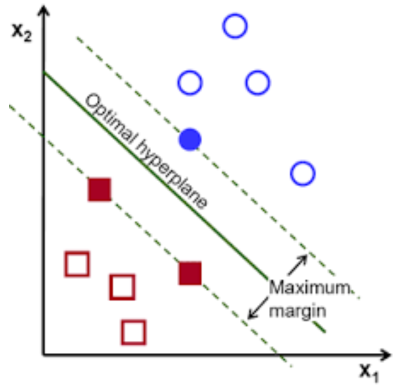
# Outline

## Lecture 1 (today)

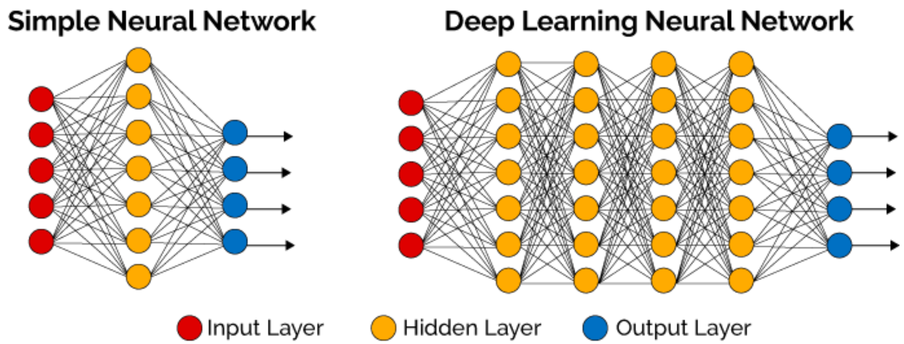
- Introduction to Machine Learning
- Supervised Learning
- Linear Models
  - Regression
  - Classification
- Hypothesis Tests and ROC curve
- Overfittig and Regularization
- Cross-Validation
- Machine Learning Software
  - Introduction to ROOT / TMVA

# Outline (2)

- Lecture 2:
  - Support Vector Machine (SVM)
  - Decision Trees




- Lecture 3:
  - Introduction to Neural Networks
  - Deep Learning



# References

Lots of materials presented taken from these lectures:

- *M. Kagan*: CERN Academic Training Lectures (2017)
- *S. Gleyzer*: TAE 2017 Lectures
- *A. Rogozhnikov*: Lecture at Yandex summer school of Machine Learning in HEP (2016)

Books:

- Elements of Statistical Learning (*Friedman et al…*)
- Pattern Recognition and Machine learning (*Bishop*)

# What is Machine Learning ?

- Field of study that gives computer the ability to learn without being explicitly programmed (Arthur Samuel, 1959)
- Better definition (T. Mitchell, 1998)
  - *Study of algorithms that improve their performance **P** for a given task **T** with more experience **E***
- Example:
  - Task: Identify Higgs boson, faces in pictures, etc…

# Where is Used ?

- Natural Language Processing
- Speech and handwriting recognition
- Object and image recognition
- Fraud detection
- Financial marker analysis
- Search engines
- Spam and virus detection
- Medical diagnosis
- Robotics control
- Automation: e.g. self-driving cars
- Advertising (recommender systems)
- ……..

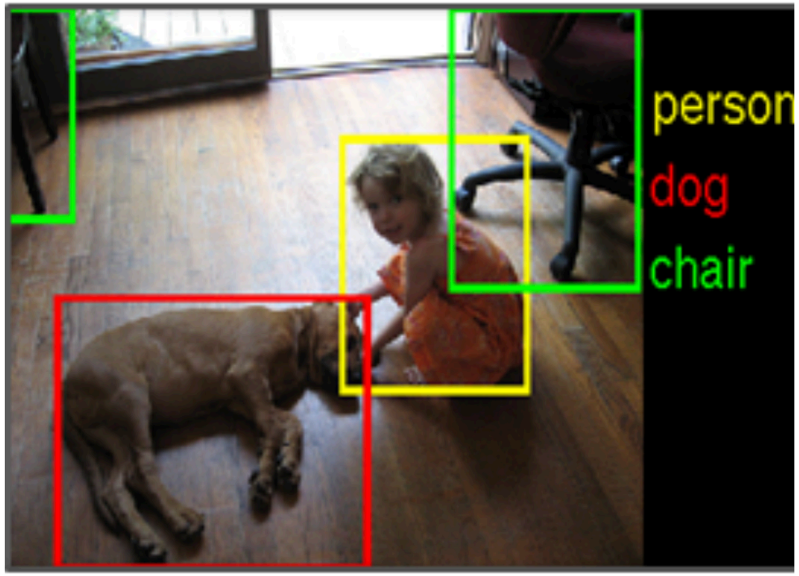
Facial recognition

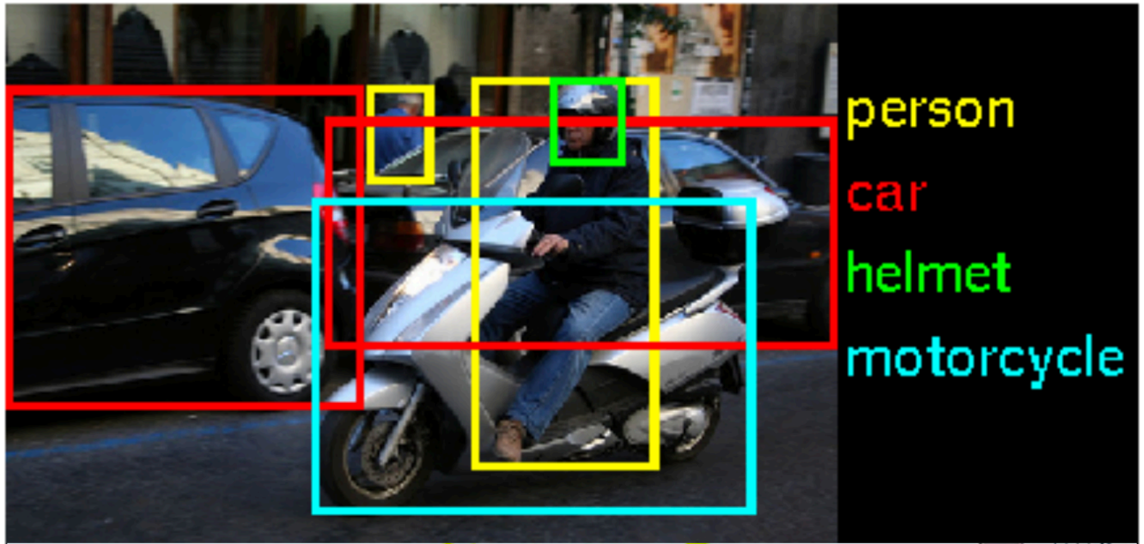
Autonomous ("self-driving") vehicles

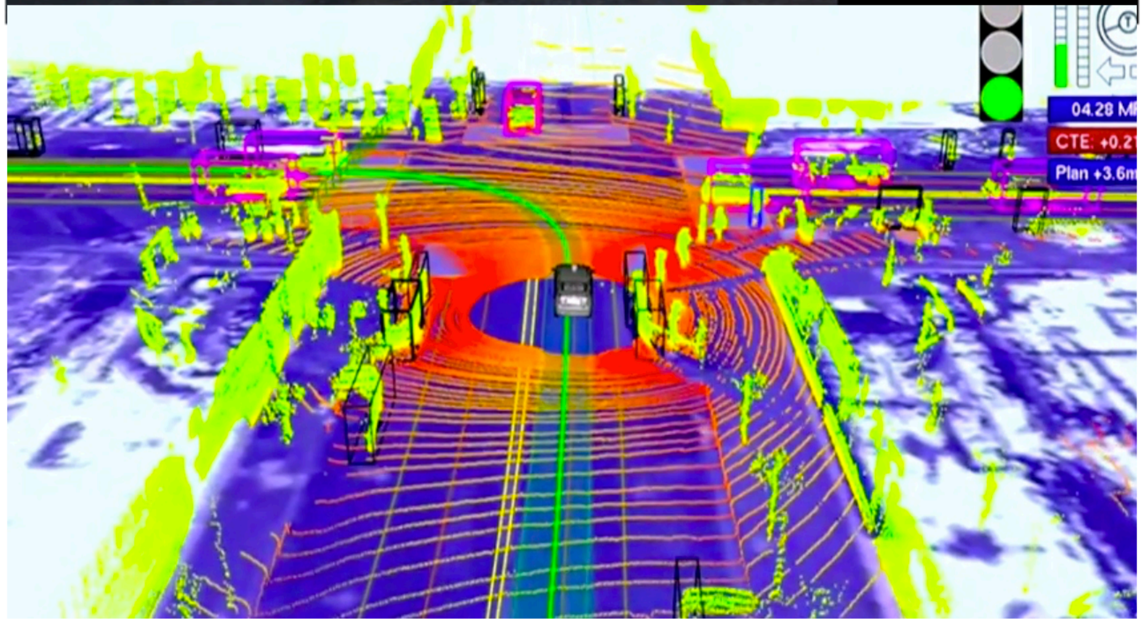
**Growing very fast !**

Recommendation engines

# Examples


person
dog
chair


person
car
helmet
motorcycle

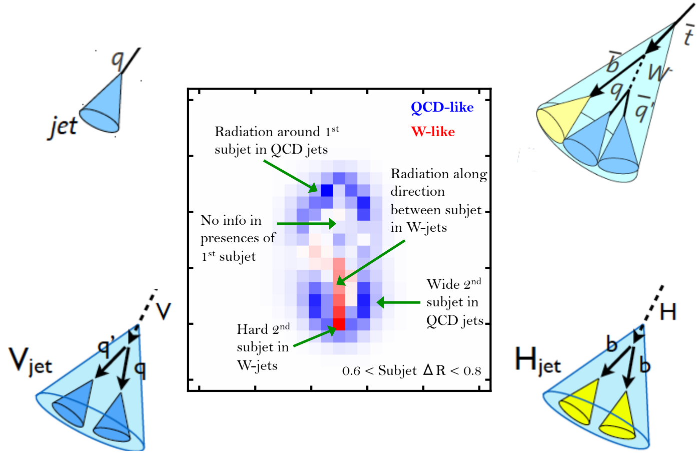

CTE: +0.2
Plan +3.6m

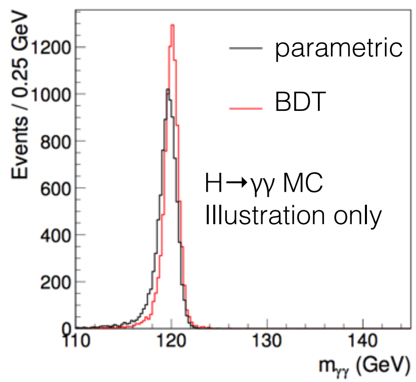
# Machine Learning in HEP

- In analysis and reconstruction
  - Classifying signal from background events
  - Reconstructing particles and improving energy/mass resolution
  - Particle identification
  - Energy calibration
- In the trigger and Data Acquisition
  - Quickly identifying complex final states
  - Data quality monitoring
- In computing
  - Estimate dataset popularity
  - Optimisation of resources

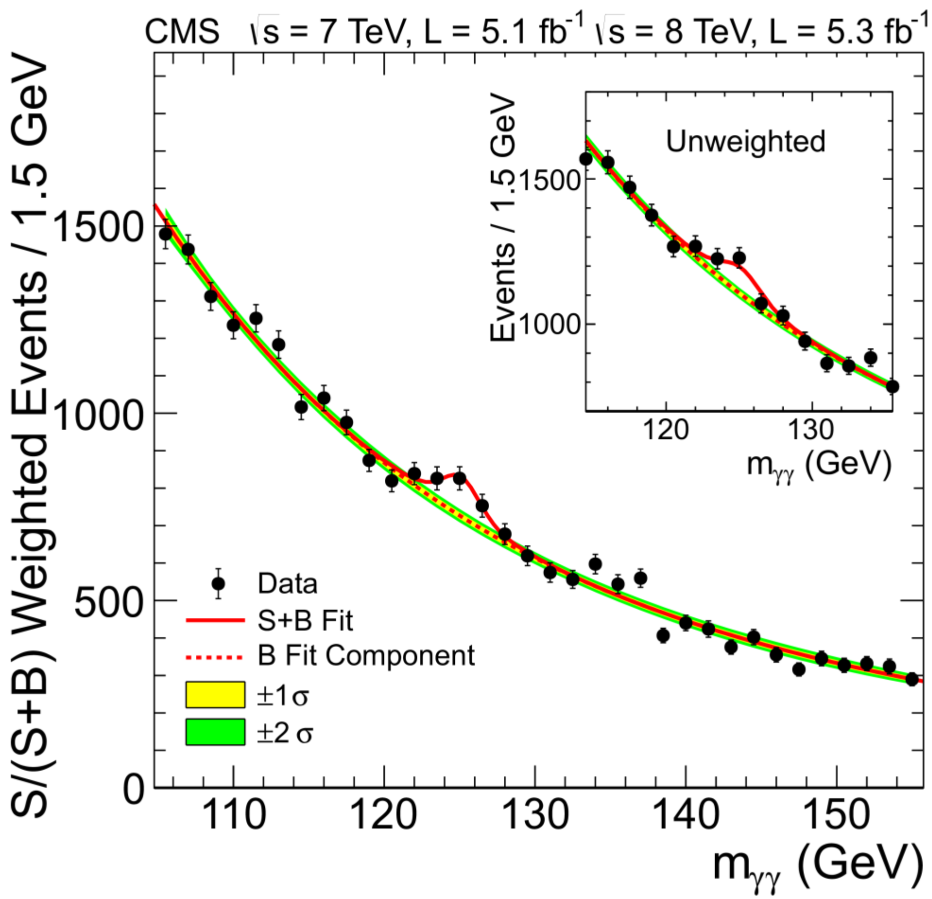
# Machine Learning in HEP

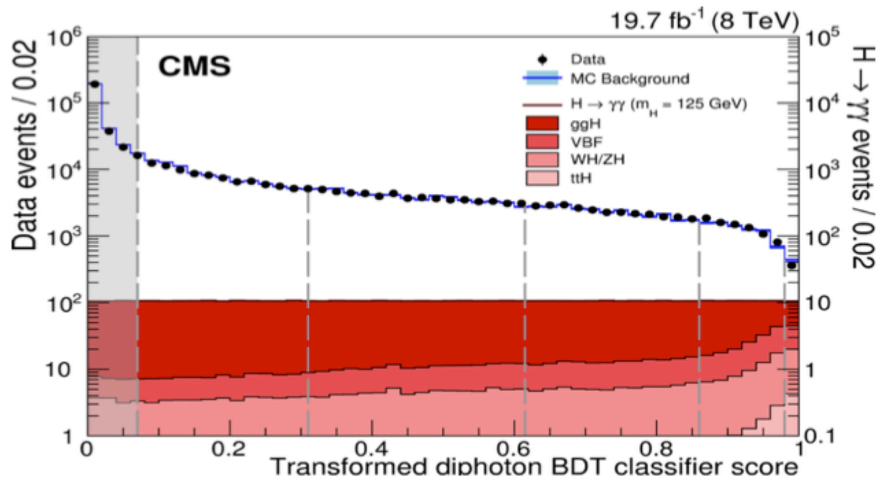
- **Classification**
  - Particle Identification
  - Pattern Recognition (tracks)
  - Searches for new Physics
- **Regression**
  - Function Estimation
    - e.g. estimate better particle energy



# Example: Higgs Discovery



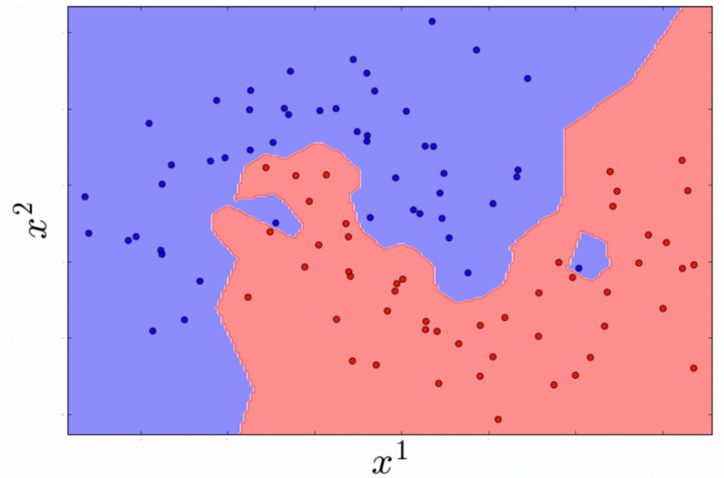

- Identification of particles
- Identification of interactions
- Energy regression
- Event selection

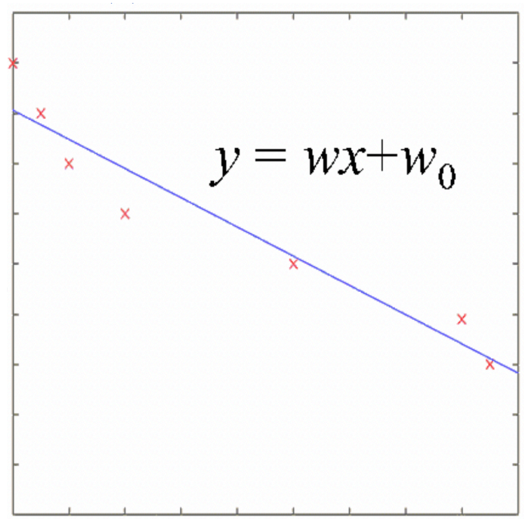
**Improvement in analysis from all 4 areas**

[S. Gleyzer]

# Mathematical Modeling

- Key element in machine learning is a mathematical model
  - mathematical characterisation of system(s) of interest, typically via random variable
  - Chosen model depends on the task and on the available data

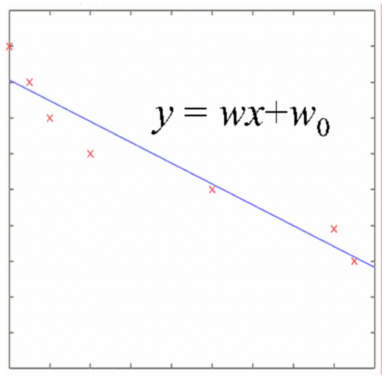


# Mathematical Modeling

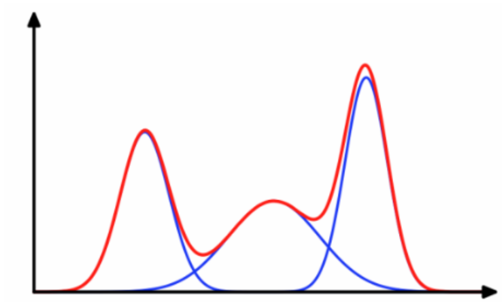
Key element is a mathematical model

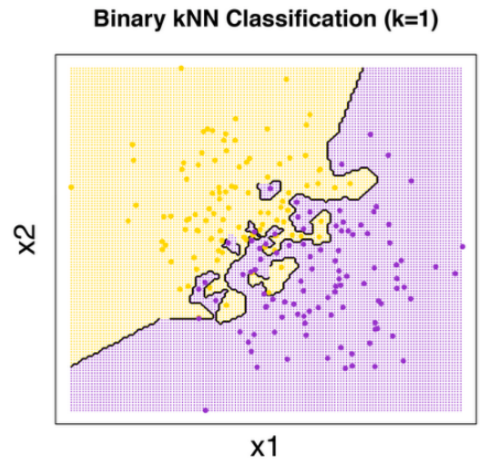
- **Learning**
  - Estimate statistical model from the data
- **Prediction and Inference**
  - use the statistical model to make predictions on new data points and infer properties of system(s)



# Mathematical Models

- **Parametric Models**
  - described with a fixed set of parameters
  - independent of data set sizes
- **Non-parametric models**
  - do not have a fixed set of parameters
  - complexity grows with data size



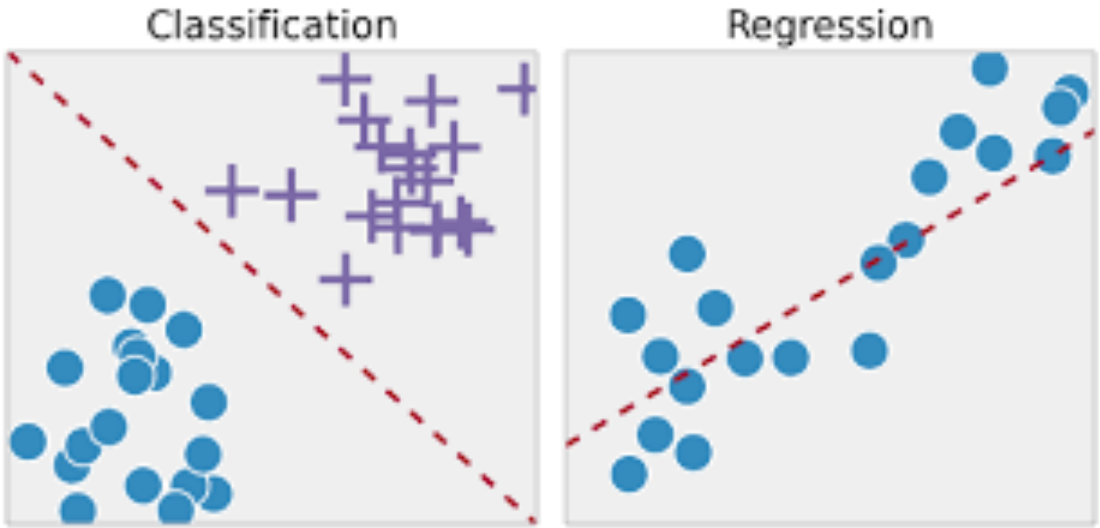


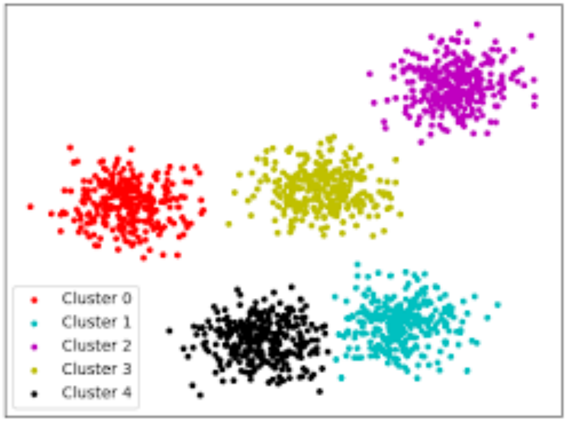
# Generative and Discriminative Models

- **Generative model**
  - Estimate probability density functions p(x,y)
    - estimate p(x | y) and prior p(y) and then using Bayesian theorem
    - $p(y|x) \propto p(x|y)p(y)$
- **Discriminative model**
  - Model directly the p(y | x)
- Majority of methods (e.g. logistic regression, neural networks) are discriminative models

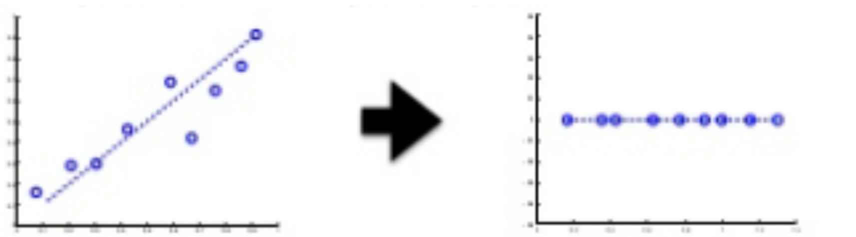
# Machine Learning Tasks

- Classification
- Regression
- Clustering
- Dimensionality Reduction



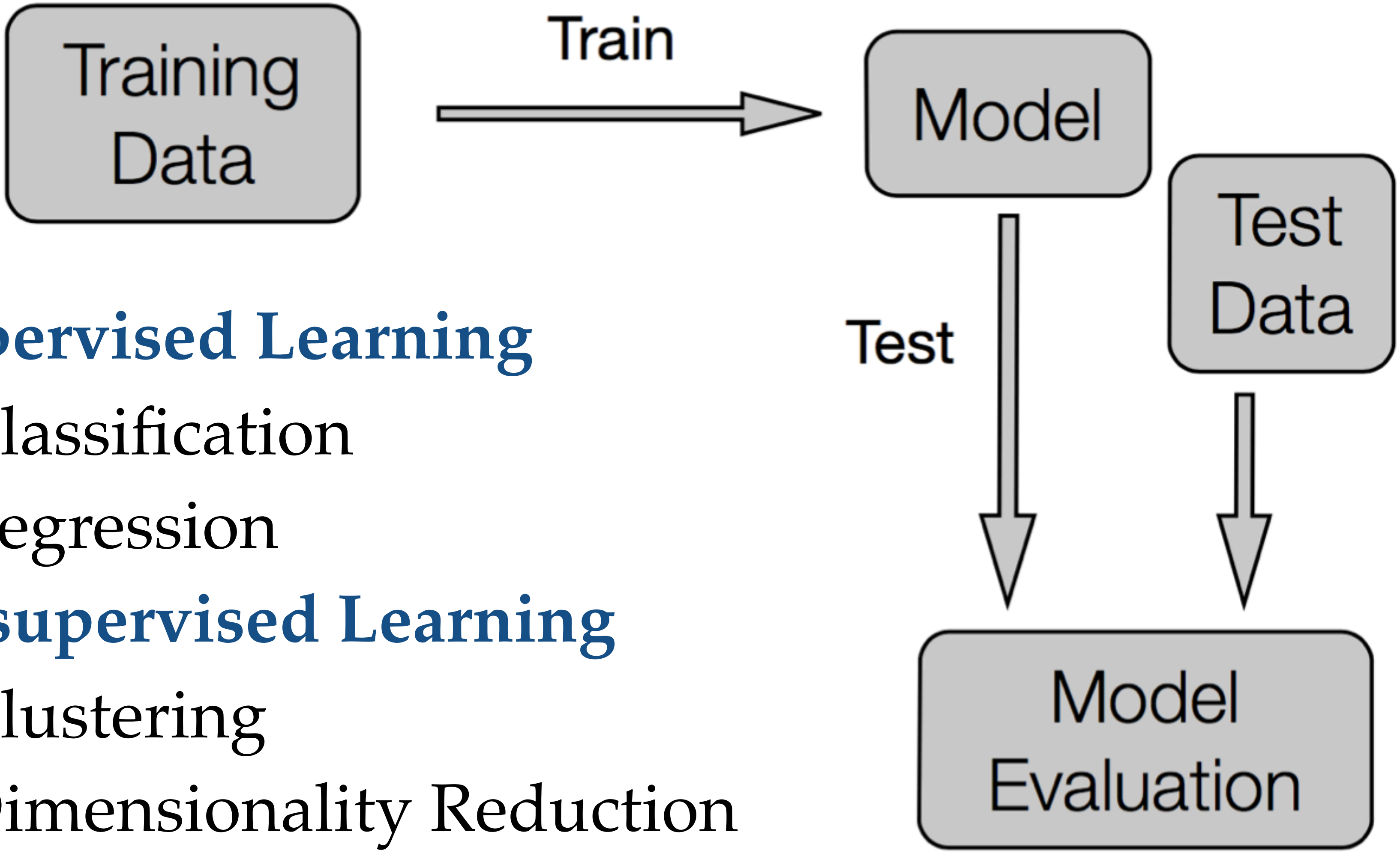




clustering



dimensionality reduction

# Learning

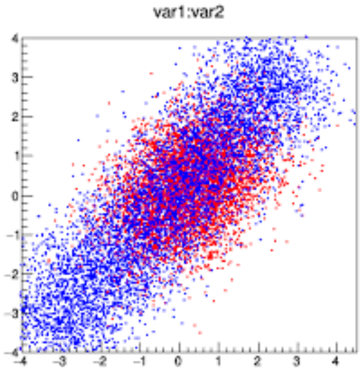



- **Supervised Learning**
  - Classification
  - Regression
- **Unsupervised Learning**
  - Clustering
  - Dimensionality Reduction
  - Anomaly Detection
- **Reinforcement Learning**

# Supervised Learning

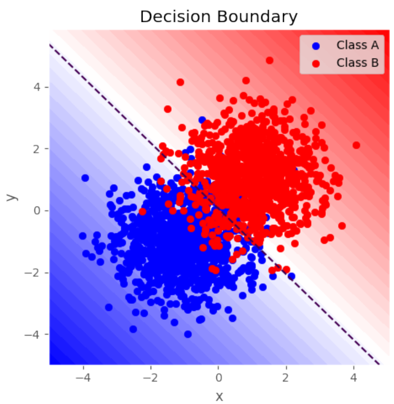
- Given N examples (events in HEP) with features (Training Data)
  - $\{ \mathbf{x_i} \in \mathcal{X} \}$ and targets $\{ y_i \in \mathcal{Y} \}$
- Learn function mapping $\mathbf{y} = \mathbf{f(x)}$
  - $\mathbf{x_i}$ is typically a n-dimensional vector (number of features)
    - $\mathbf{X}$ is a matrix (number of events × number of features)
  - $y_i$ is instead a scalar

# Supervised Learning: Classification

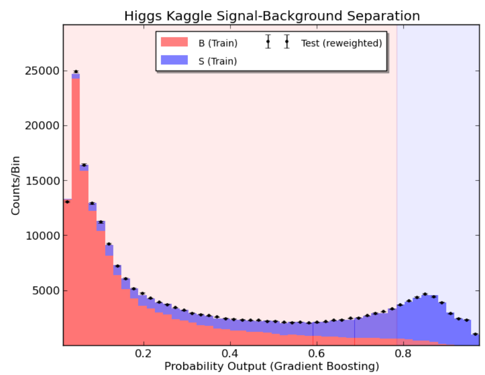


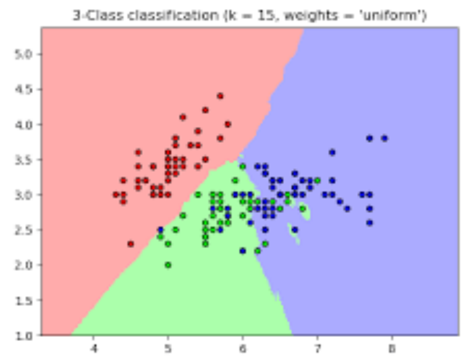

- $\{ \mathbf{x}_i \in \mathcal{X} \}$ and targets $\{ y_i \in \mathcal{Y} \}$
  - Learn function mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$
- **Classification** : $\mathcal{Y}$ are a finite set of labels
  - binary classification $\mathcal{Y} = \{0,1\}$

    e.g. Higgs event vs Background events
  - multi-class classification $\mathcal{Y} = \{c_1, c_2, \ldots\ldots, c_n\}$



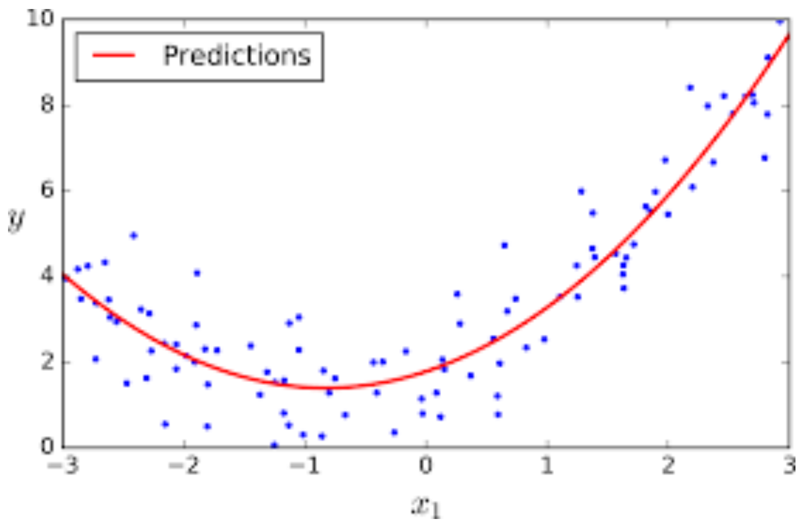


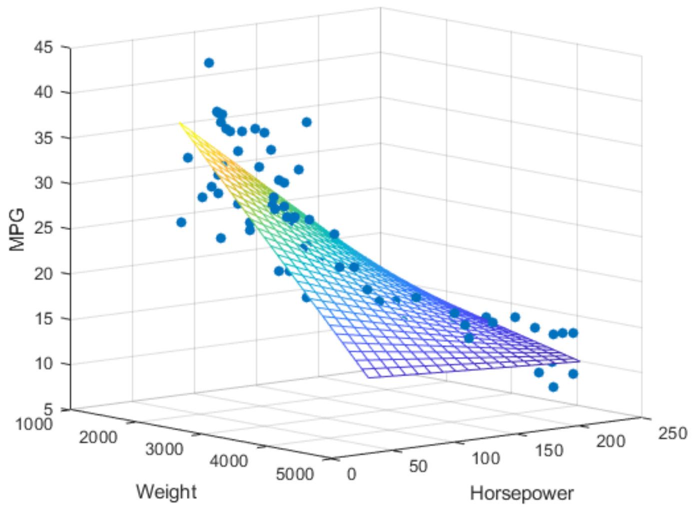


# Supervised Learning: Regression

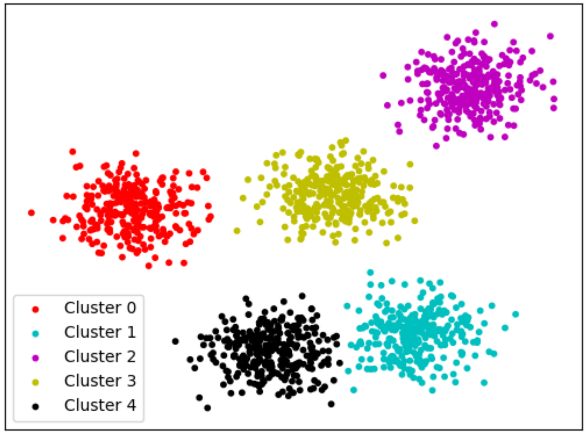
- $\{\ \mathbf{x_i} \in \mathcal{X}\ \}$ and targets $\{\ y_i \in \mathcal{Y}\ \}$
  - Learn function mapping $\mathbf{y = f(x)}$
- **Regression:** $\mathcal{Y}$ are Real Numbers



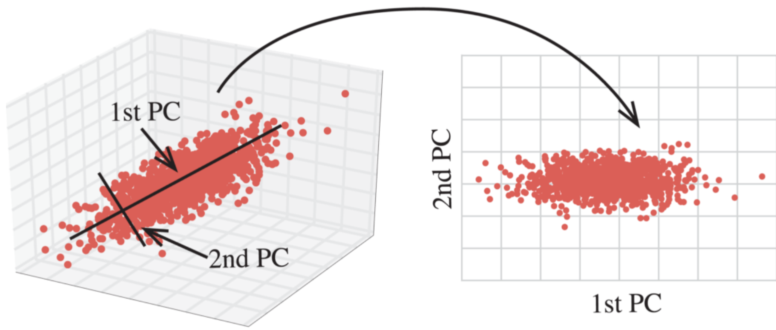



# Unsupervised Learning

- Given some data $D = \{\mathbf{x_i}\}$, but no labels
- Find possible structure in the data
  - **Clustering**:
    - partition the data into groups $\mathbf{D = \{ D_1 \cup D_2 \cup D_3 \ldots\ldots \cup D_n \}}$

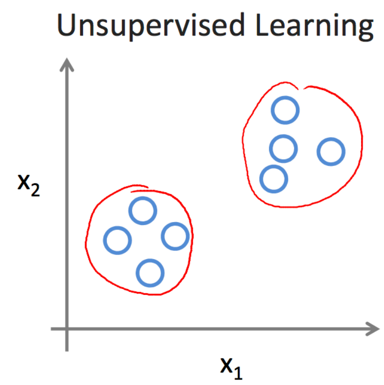


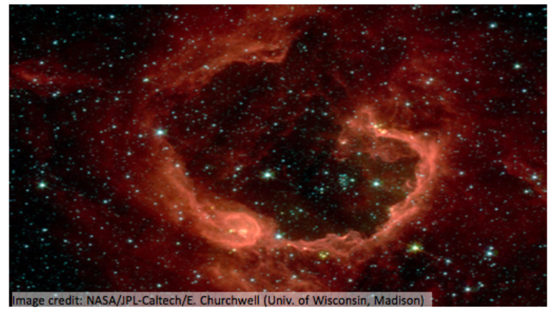

  - **Dimensionality reduction**:
    - find a low dimensional representation of the data with a mapping $\mathbf{Z = h(X)}$

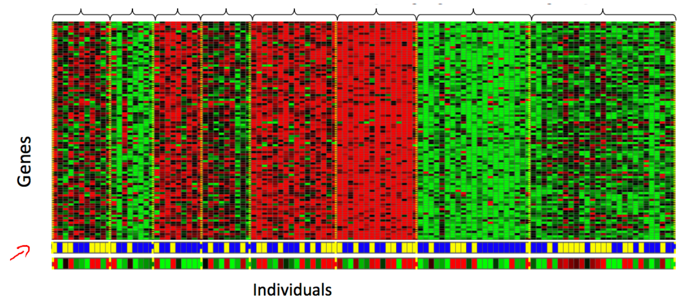


# Example: Clustering

- Astronomical analysis:
  - grouping of galaxies
- Genetic analysis in biology
- Market Segmentation
- Organization of computing clusters
- Social network analysis
- Grouping of information (e.g. Google news)
- ….



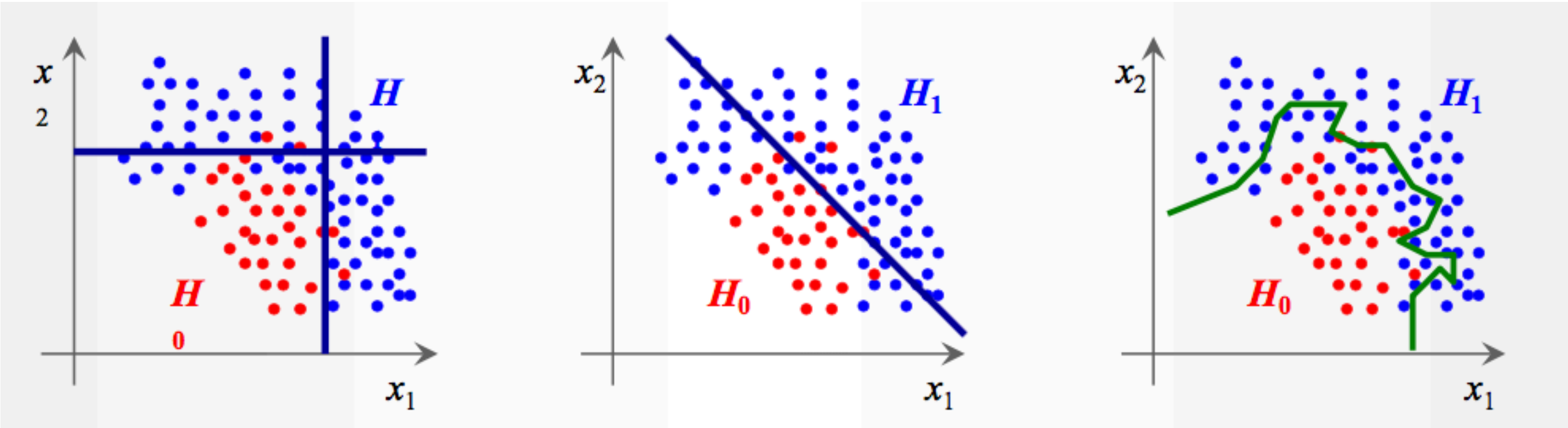



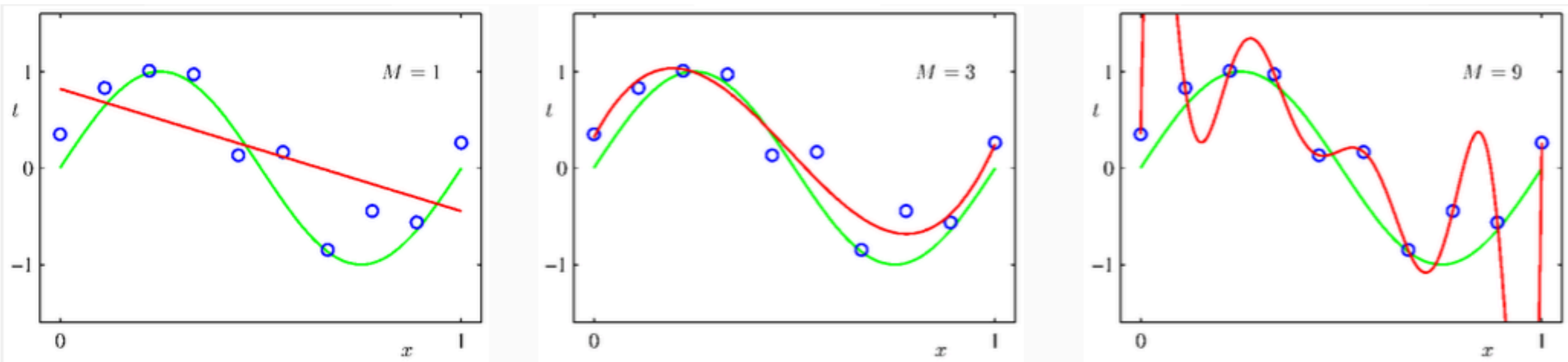

Andrew Ng

# Classification and Regression Tasks

- **Classification** - How to find the best decision boundary ?




- **Regression** - How to determine the correct model ?



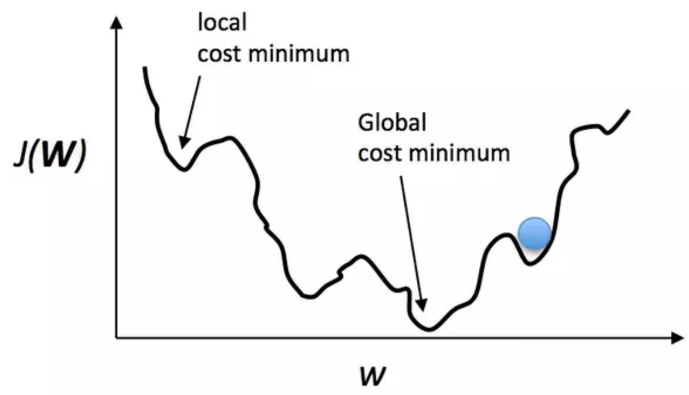

[E. v. Toerne]

# Supervised Learning

## How does it works ?

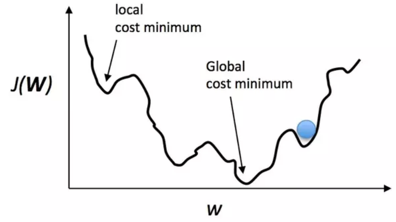
- Choose a function with parameters
  - $\mathcal{F} = \{ f(\mathbf{x}; \mathbf{w}) \}$
  - with optional constraint $\mathbf{\Omega}(\mathbf{w})$
- Design a Loss function measuring the cost of choosing badly

$$L(\mathbf{w}, \mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \mathbf{w}))$$




- Find best values of the parameters **w** that minimize the loss function L(**w, x**)
- Estimate final performance on an independent data set

# Loss Function Minimization




- Minimization of Loss Function
  - Use a labeled training-set to compute loss function
  - Iterative optimisation procedure (e.g. gradient descent) to find parameter values, i.e. $f(\mathbf{x}; \mathbf{w})$ , which gives the minimum of the Loss function

$$\arg\min_{\mathbf{w}} L(\mathbf{w}, \mathbf{x}) = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \mathbf{w})) + \lambda\Omega(\mathbf{w})$$

- $\boldsymbol{\lambda\Omega}(\mathbf{w})$ is a constraint term on the parameters **w**
  - regularisation, penalising certain values of **w**

# Example: Linear Regression

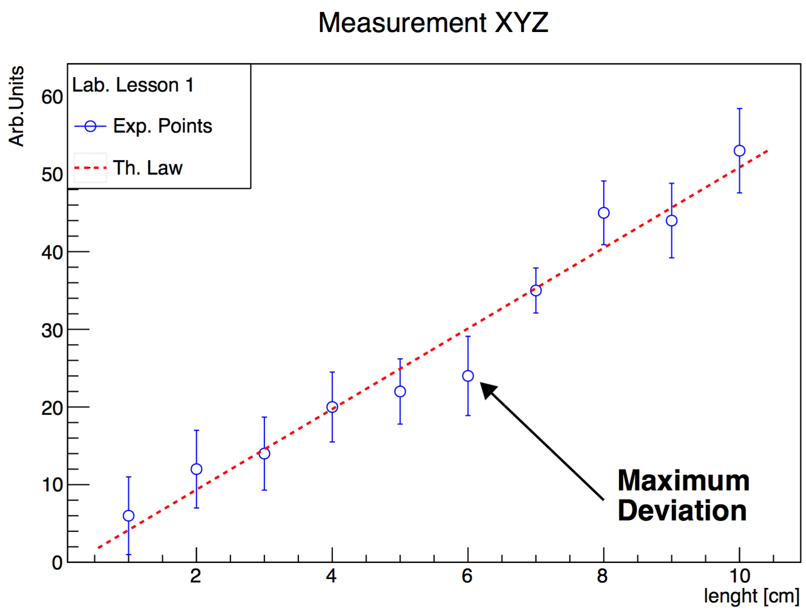
- Loss Function often used for regression
  - Square Error Loss

$$\boxed{L(f(\mathbf{x};\mathbf{w}), y) = (f(\mathbf{x};\mathbf{w}) - y)^2}$$

- Linear Regression :
  - assume a linear model

$$\boxed{f(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}}$$

- Find **w*** minimum of L(**w**)



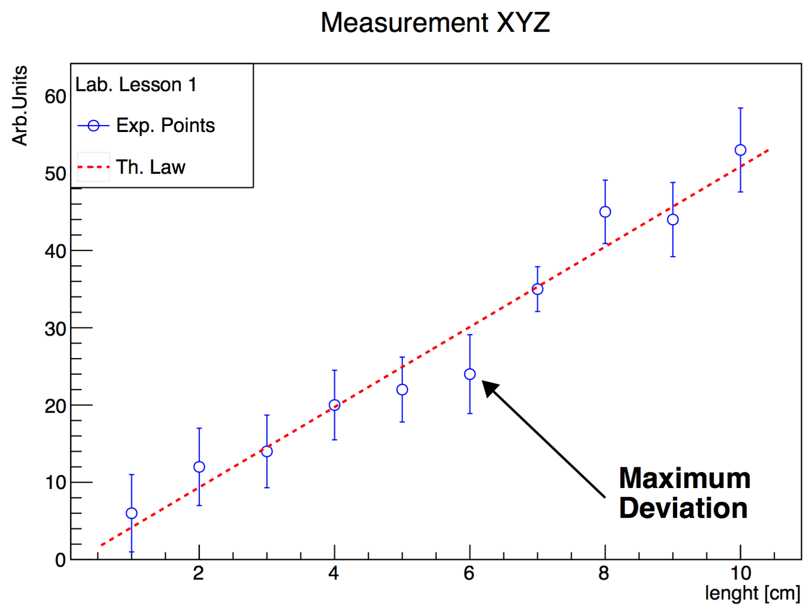
# Least Square Regression

- Least Square Loss function

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(\mathbf{x_i};\mathbf{w}))^2$$

- Find minimum of L(w)

- Used also for parameter estimation
i.e. Least square fit ($\chi^2$ fit)



# Parameter Estimation

- Model process using likelihood function of the observed data

$$\mathcal{L}(\mathbf{w}) = P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \prod_i p\left(y_i|\mathbf{x_i};\mathbf{w}\right)$$

- Parameter Estimation: find parameters that maximise likelihood function
  - Equivalent: minimise log-likelihood

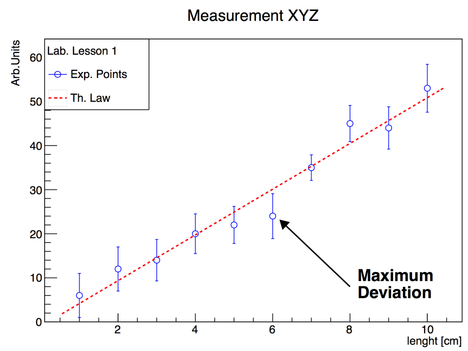
$$\mathbf{w}^* = \arg\max_w \mathcal{L}(\mathbf{w}) = \arg\min_w -\log\mathcal{L}$$

# Parameter Estimation: Linear model

- Assume: $y_i = f(x_i) + e_i$ with $f(x) = wx_i$
- With Normal random error $e_i \sim \mathcal{N}(0, \sigma)$ $p(e_i) \propto \exp(-e_i^2/(2\sigma^2))$
  - the model for $y_i$ is described by a $p(y_i|x_i,w) \propto \exp((wx_i-y_i)^2/(2\sigma^2))$
- The likelihood function is then

$$\mathcal{L}(w) = p(\mathbf{y}|\mathbf{x}, w) = \prod_i p\left(y_i|x_i; w\right)$$

$$\rightarrow -\log \mathcal{L}(w) = \sum_i (y_i - w_i x_i)^2$$

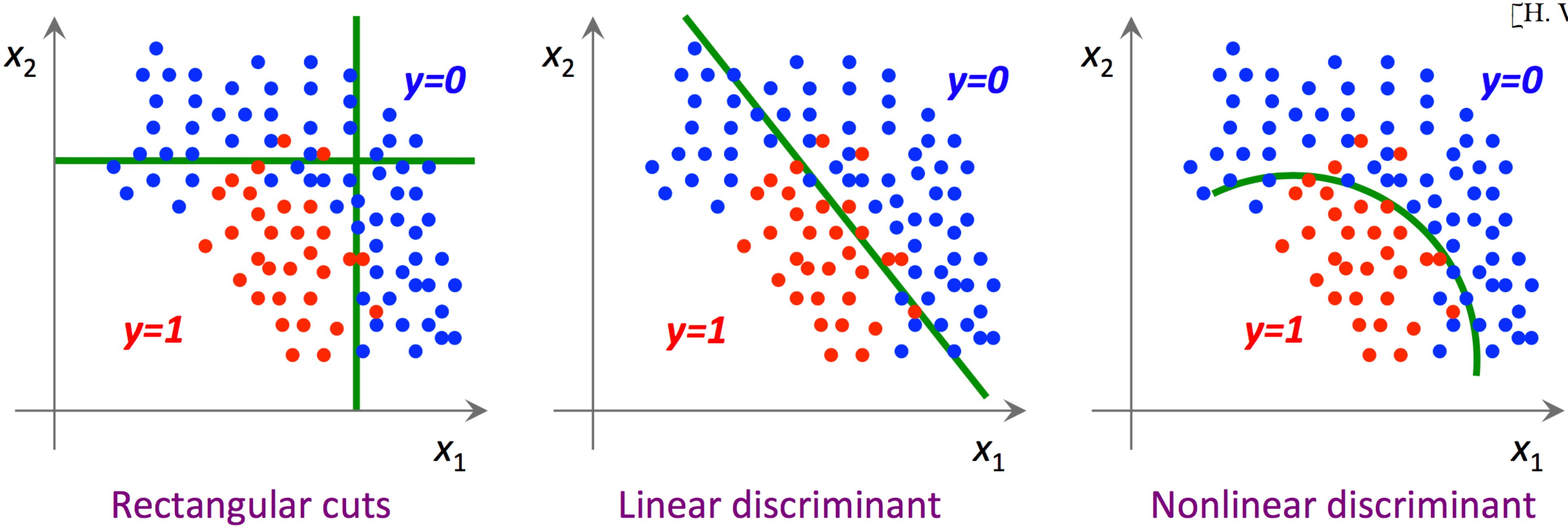



- The negative log-likelihood function is equivalent to the least square loss
- Probabilistic interpretation for our simple regression machine learning model

# Classification

- Want to learn a function to separate different class of events.
  - Problem is to find best decision boundary

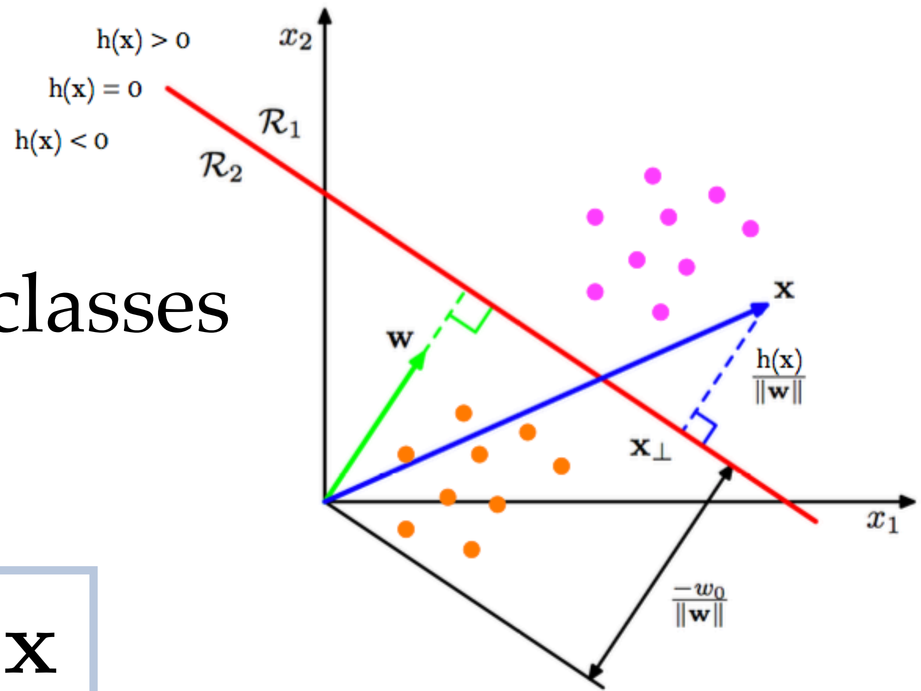
[H. Voss]



# Linear Decision Boundary

- Want to separate 2 classes
- Linear model

$$\boxed{h(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}}$$




[Bishop]

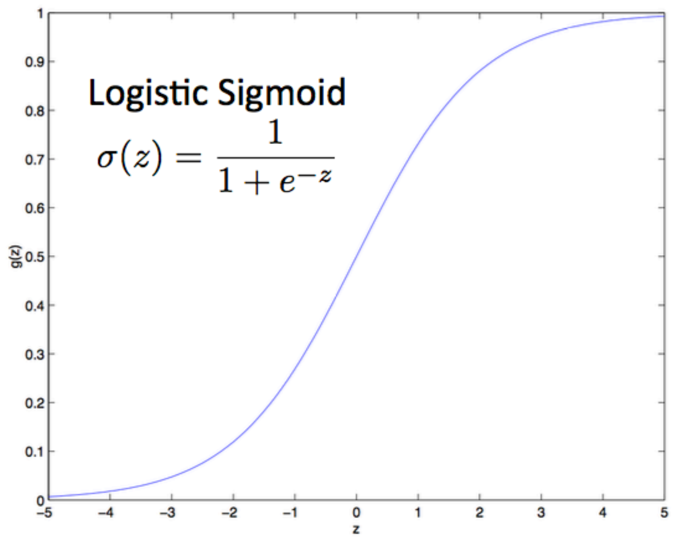
- Predict class 0 if h(x) < 0 otherwise class 1 if h(x) > 0
  - distance from boundary = h(**x**)/ | | **w** | |

# Logistic Regression

- Linear discriminant $\boxed{h(\mathbf{x};\mathbf{w}) = \mathbf{w}^T\mathbf{x}}$

- Use probability that example **x** is in a given class using the sigmoid function

$$\boxed{p(y=1|\mathbf{x}) \equiv p_i = \frac{1}{1+e^{-h(\mathbf{x};\mathbf{w})}} = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}}$$




further from the boundary, more certain about class

# Logistic Regression

- Probabilistic interpretation
- 2 classes classification: Bernoulli process

$$p(y_i|x_i) = (p_i)^{y_i}(1-p_i)^{1-y_i} \qquad y_i = \{0,1\}$$

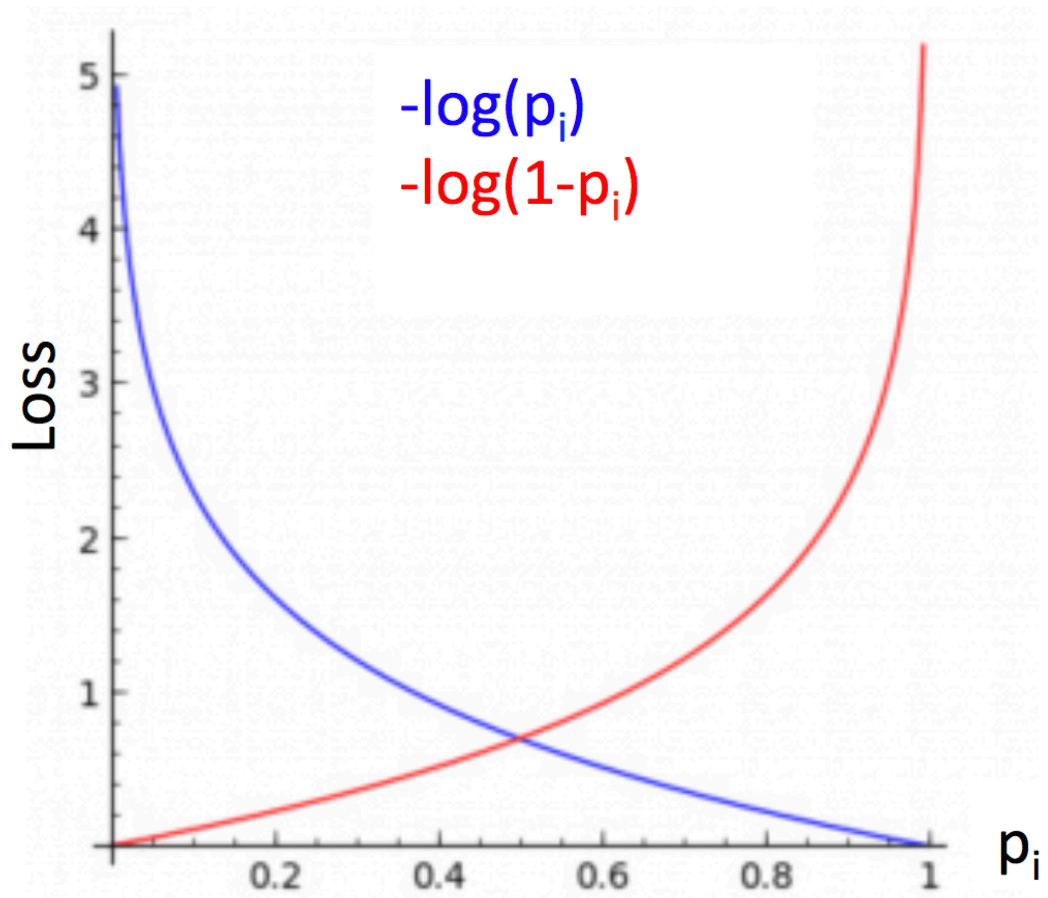
- Negative Log-Likelihood is then

$$-\ln \mathcal{L} = -\sum_i (y_i \ln p_i + (1-y_i)\ln(1-p_i))$$

**Binary Cross Entropy Loss function**

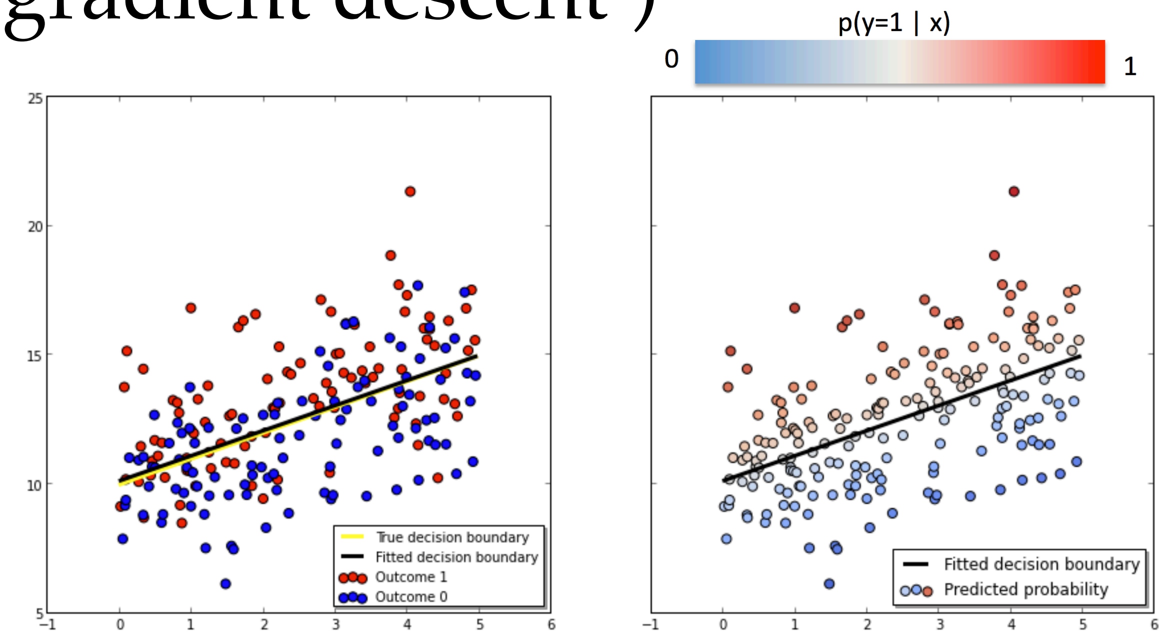
# Binary Cross Entropy Function

$$L(\mathbf{w}) = -\sum_i (y_i \ln p_i + (1 - y_i) \ln(1 - p_i))$$



# Logistic Regression

- Find the values of **w** minimising the cross-entropy loss function
  - $\mathbf{w}^* = \arg\min_{\mathbf{w}} -\ln L(\mathbf{w})$
- Use iterative algorithm to find optimal value **w*** (e.g. gradient descent )



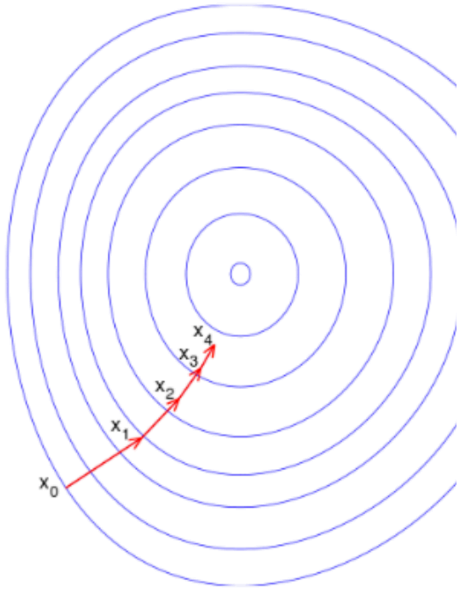
# Gradient Descent

Minimize Loss function by repeated gradient steps

– Compute gradient w.r.t. parameters: $\dfrac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$

– Update parameters $\mathbf{w}' \leftarrow \mathbf{w} - \eta \dfrac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$

– η is called the learning rate, controls how big of a gradient step to take




[M. Kagan]

Many variants exists especially in case of deep neural network training

# Why Sigmoid function?

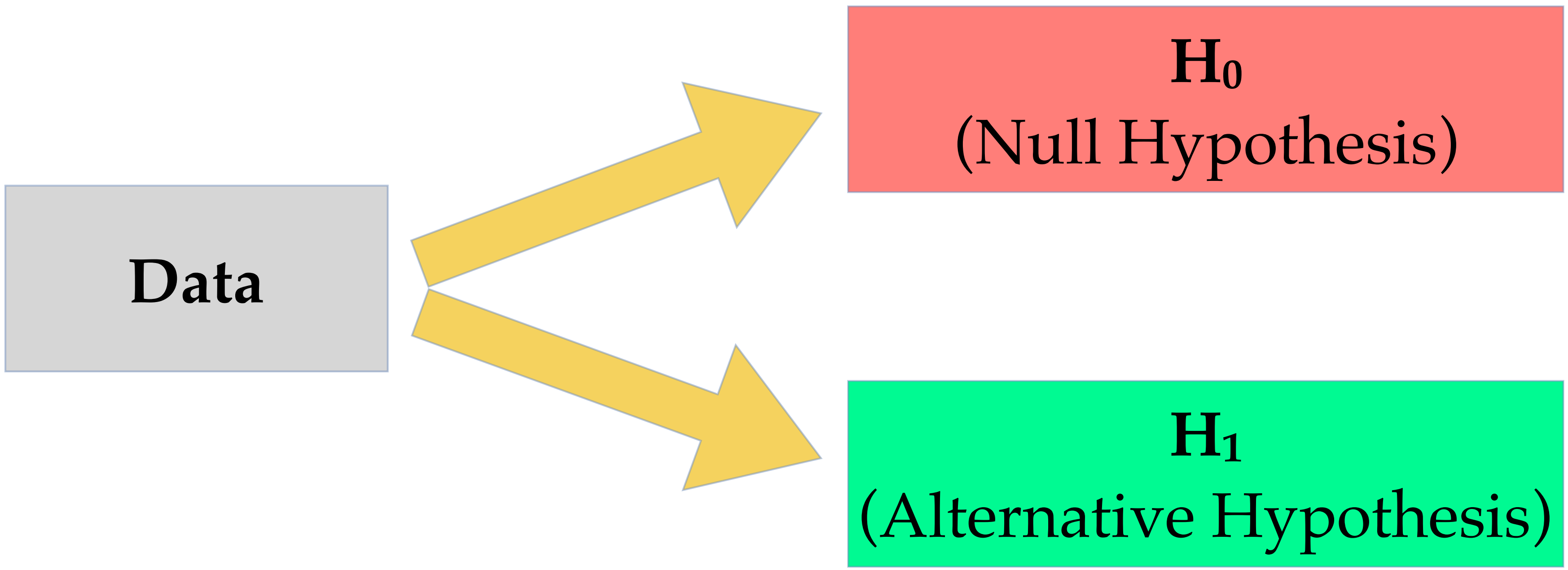
- If we use probabilistic theory and Bayes statistics, the posterior p(y=1 | x) :

$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1) + p(x|y=0)p(y=0)}$$

- Then by using $z = \ln \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}$

$$\boxed{p(y=1|x) = \frac{1}{1+e^{-z}}}$$

# Hypothesis Test




Which hypothesis is the most consistent with the data we have observed ?

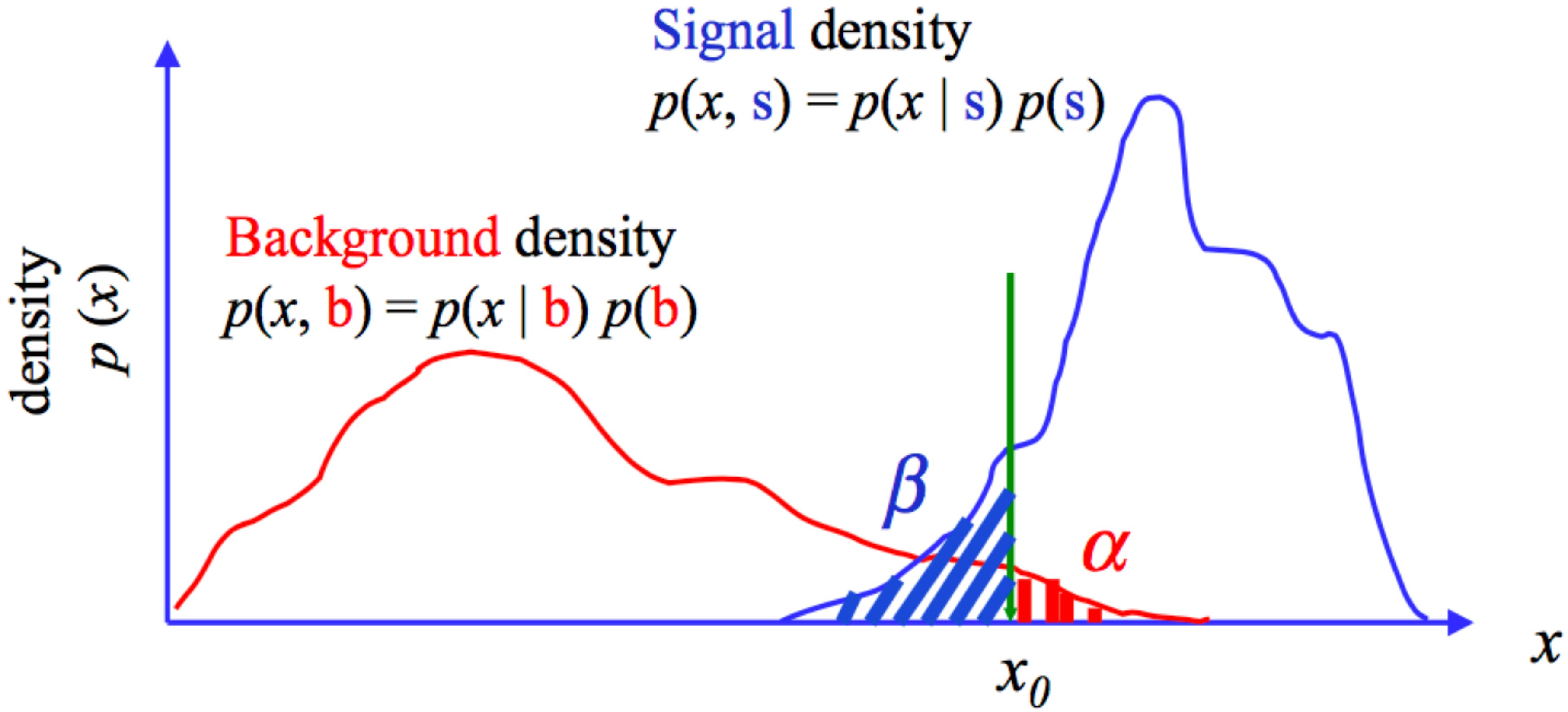
# Hypothesis Test

- $H_0$: null hypothesis
  - the hypothesis we want to reject
  - e.g. the data contains only background
- $H_1$: alternative hypothesis
  - e.g. the data consists of signal + background events
- **Critical region**:
  - regions of the test statistics defining the hypothesis rejection
- $\alpha$ : significance level (Type 1 error)
  - probability to reject $H_0$ when is true (false positive)
- $\beta$: Type 2 error
  - probability to accept $H_0$ when is false (false negative)
  - 1-$\beta$: power of the test

# Classification as Hypothesis Test

$H_0$: Background
$H_1$: Signal

$x > x_0$ : reject background and accept signal



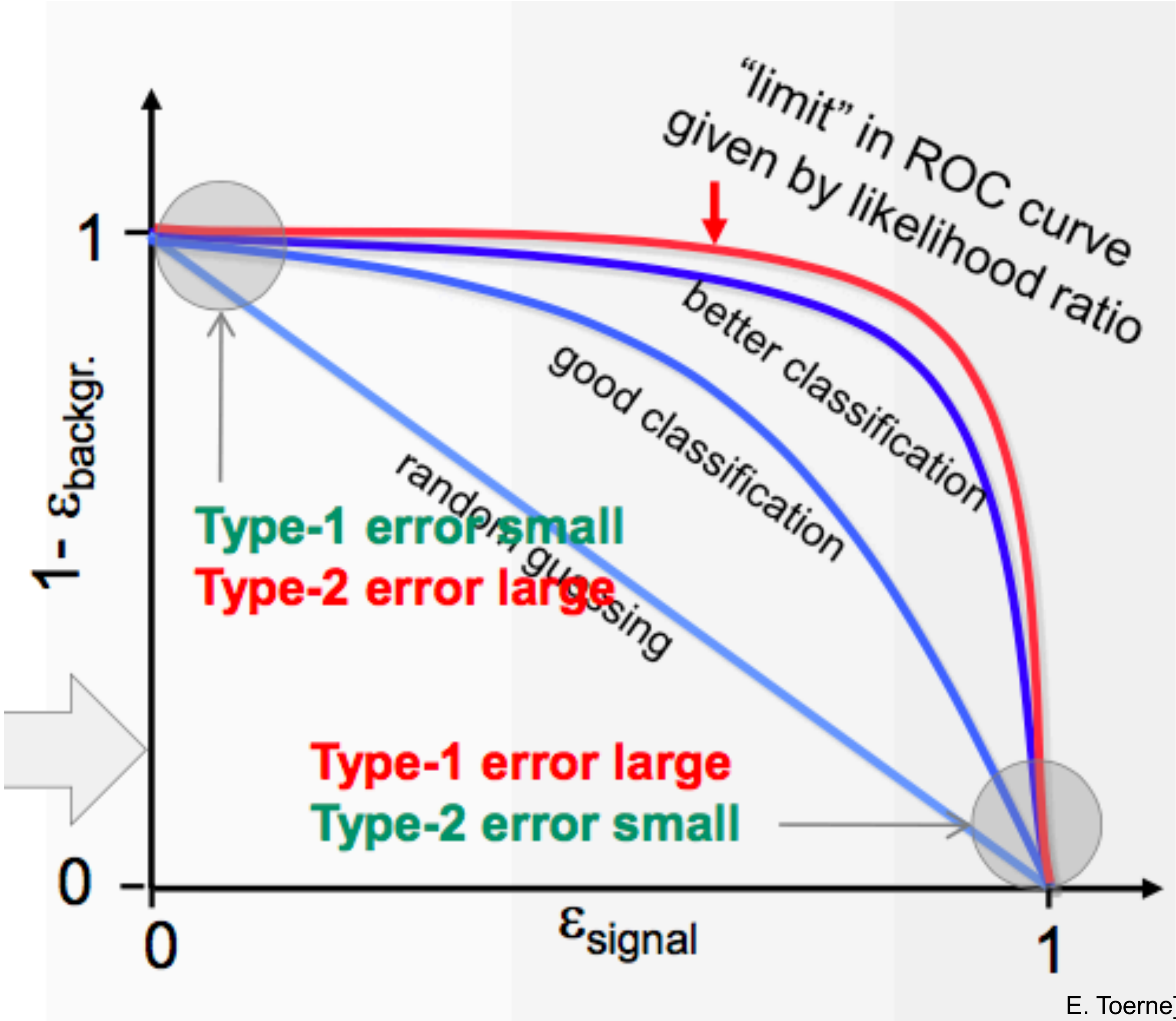

Optimality criterion: minimize the error rate, $\alpha + \beta$

[S. Gleyzer]

# Hypothesis Test

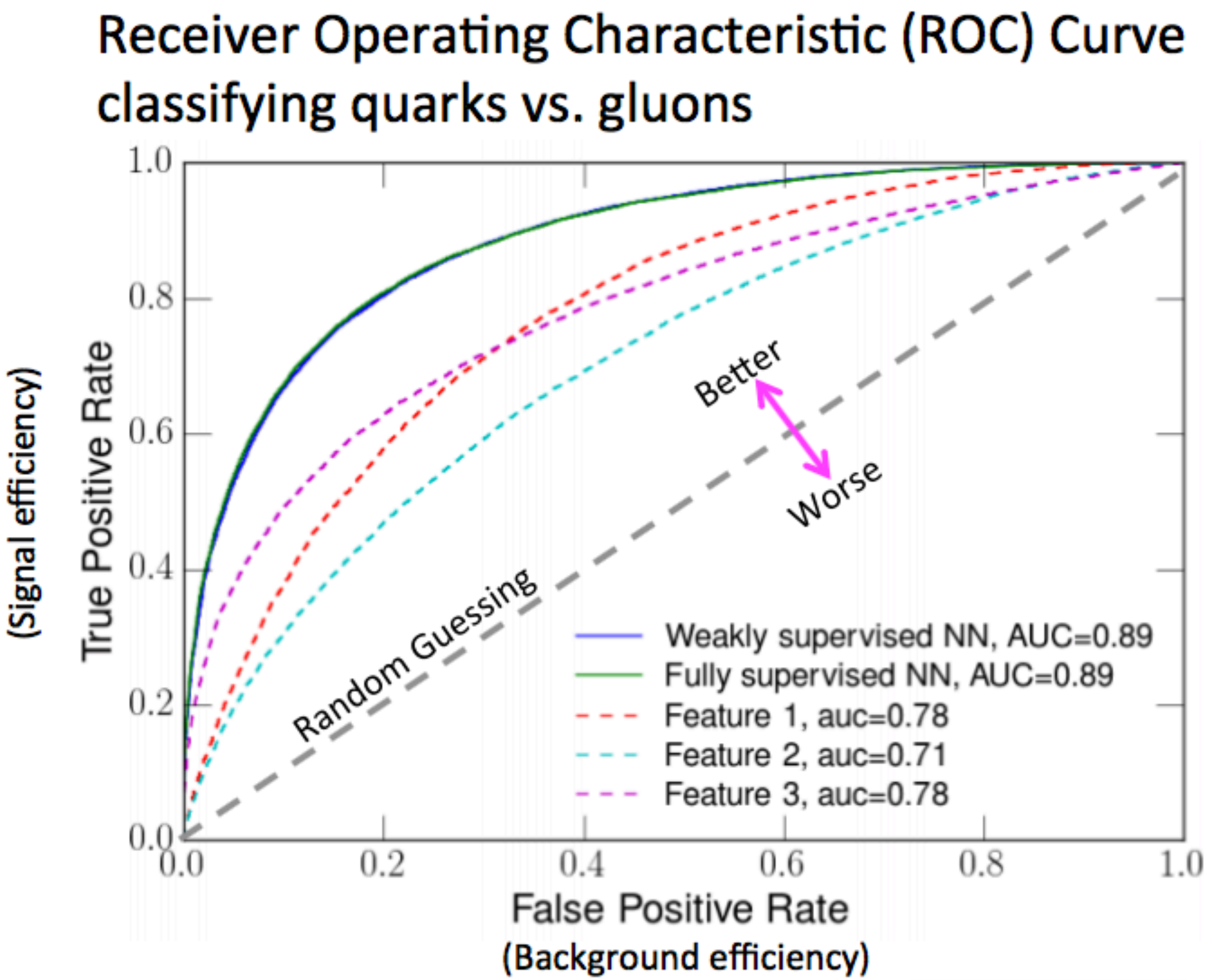
| | State of Nature | |
|---|---|---|
| Decision we make | $H_o$ is true | $H_o$ is false |
| Accept $H_o$ | ok | Type II error<br>probability $\beta$ |
| Reject $H_o$ | Type I error<br>probability $\alpha$ | ok |

# ROC Curve




Optimal classifier maximises the area under the ROC curve **(AUC)**

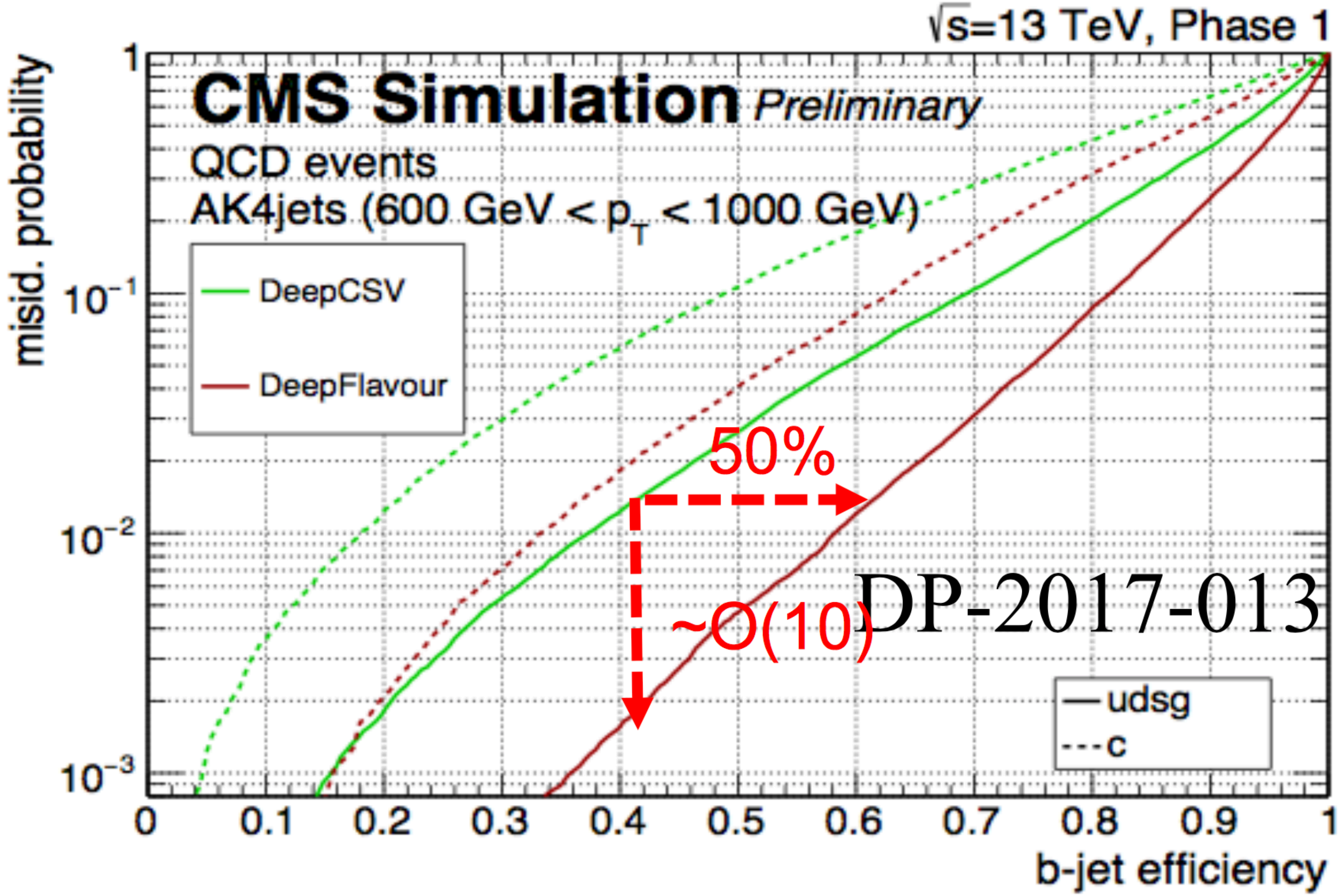
# ROC Curve




arXiv:1702.00414

[M. Kagan]

# ROC Curve



√s=13 TeV, Phase 1
CMS Simulation Preliminary
QCD events
AK4jets (600 GeV < p_T < 1000 GeV)
DeepCSV
DeepFlavour
50%
~O(10)
DP-2017-013
udsg
c
misid. probability
b-jet efficiency
1
10^-1
10^-2
10^-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

# Sensitivity and Specificity

**The Truth**

| **Test Score:** | Has the disease | Does not have the disease | |
|---|---|---|---|
| Positive | True Positives (TP) a | b False Positives (FP) | **PPV =** $\frac{TP}{TP + FP}$ |
| Negative | c False Negatives (FN) | d True Negatives (TN) | **NPV =** $\frac{TN}{TN + FN}$ |

| | **Sensitivity** | **Specificity** |
|---|---|---|
| | $\frac{TP}{TP + FN}$ | $\frac{TN}{TN + FP}$ |
| Or, | $\frac{a}{a + c}$ | $\frac{d}{d + b}$ |

Sensitivity:

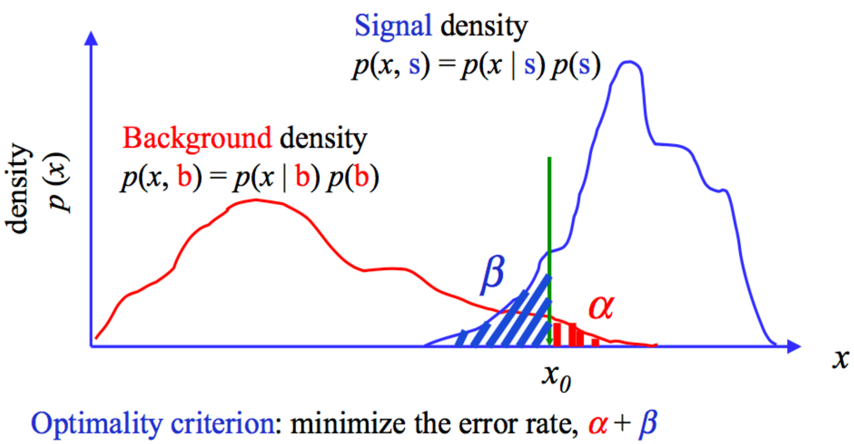
- Signal efficiency
- True Positive rate

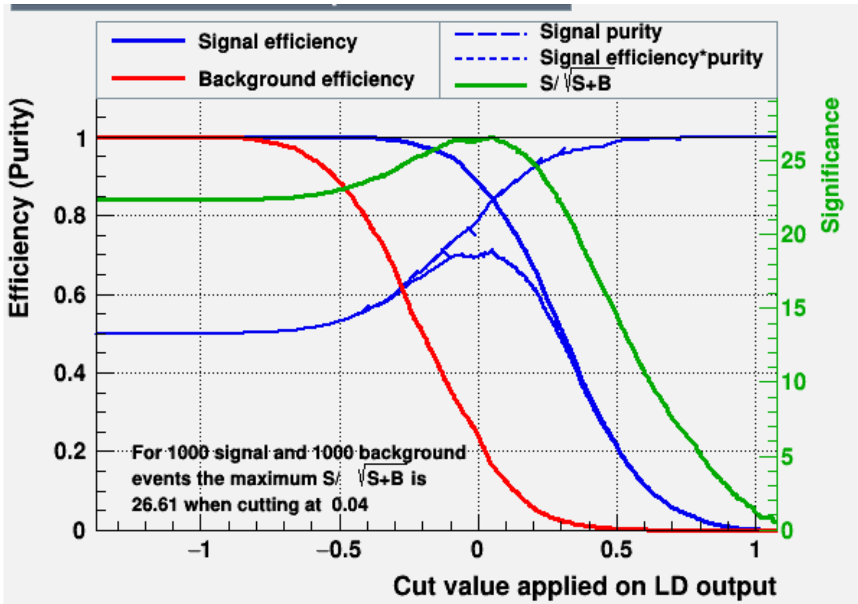
Specificity:

- 1.- Background efficienc
- True Negative rate

# Purity

- Purity = Number of signal Events passing selection / Total number of events passing the selection
  - Purity = True Positive / ( True Positive + False Positive)
  - important value but dependent on total number of Signal / Background events)
- Optimize selection depending on analysis
  - e.g. S/√(S+B) or expected Asimov significance for discovery



# Neyman-Pearson Lemma

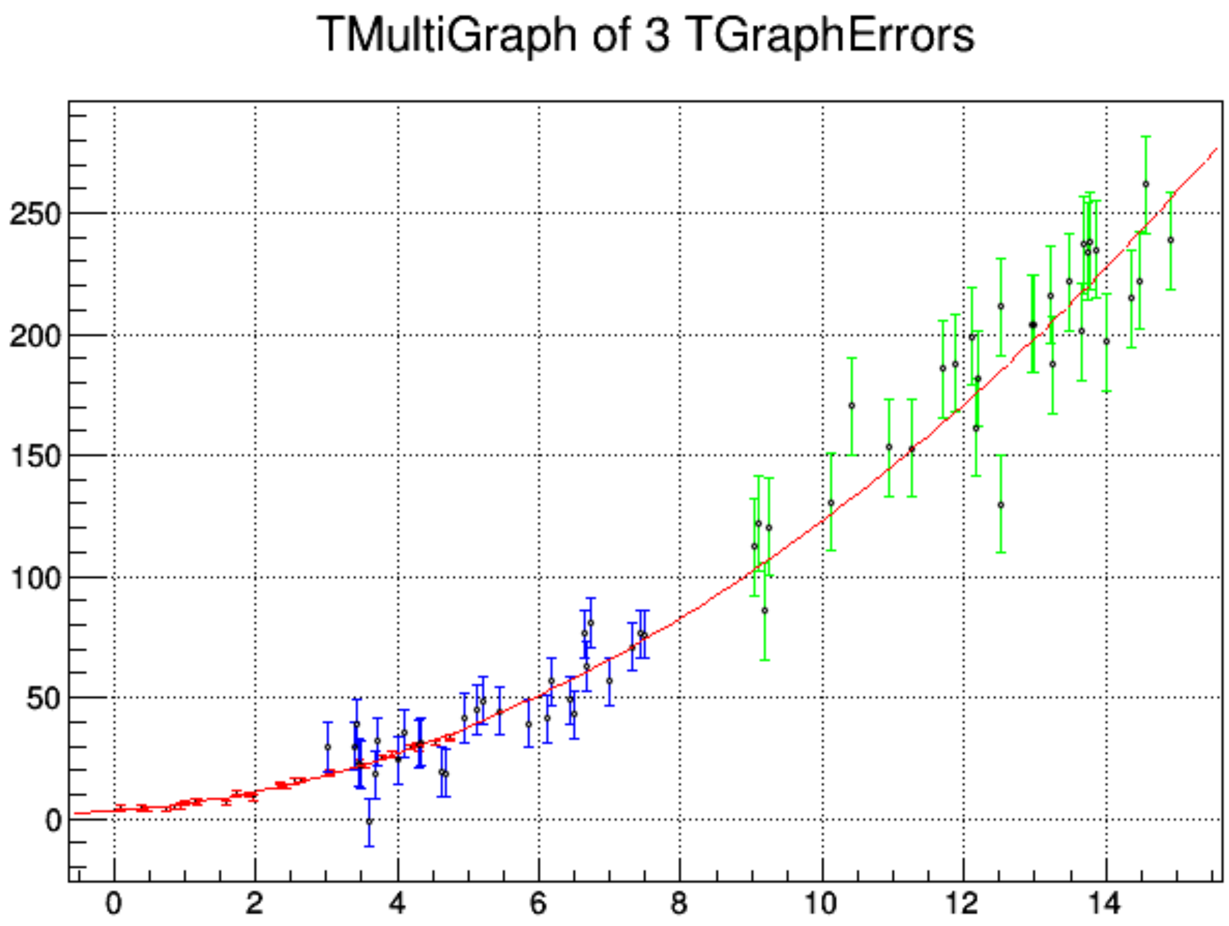
- The likelihood ratio **λ**(x) used as selection criteria, gives for each selection efficiency $\alpha$ the best possible rejection of $H_0$ in favour of $H_1$ (background rejection)

$$\lambda(x) = \frac{L(x|H_0)}{L(x|H_1)} \leq c$$

where $$P(\lambda(X) \leq c|H_0) = \alpha$$

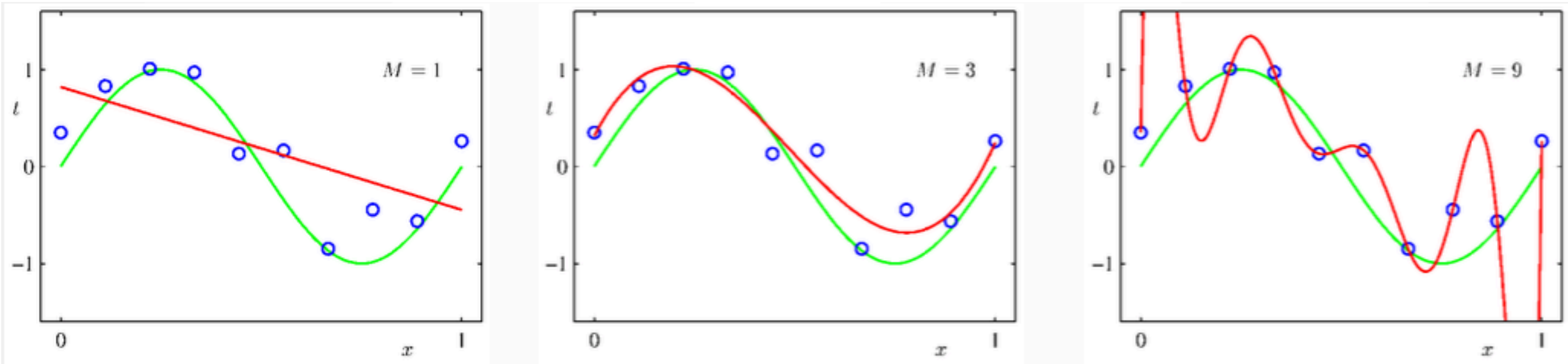
The cut value c defines the rejection region of the null hypothesis $H_0$

# Polyonial Regression




$$f(x|\mathbf{w}) = w_0 + w_1 x + w_2 x^2$$

# What is the correct model ?




$$f(x|\mathbf{w}) = w_0 + w_1 x \qquad f(x|\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \qquad f(x|\mathbf{w}) = w_0 + w_1 x + .... + w_9 x^9$$

## Under fitting
## Large Bias

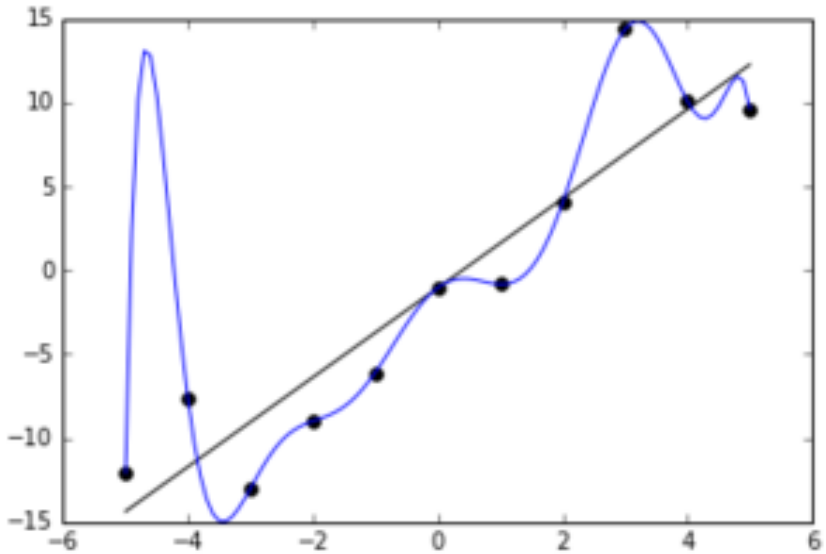
model does not reproduce well the data

## Over fitting
## Large Variance

model reproduce the training data too well

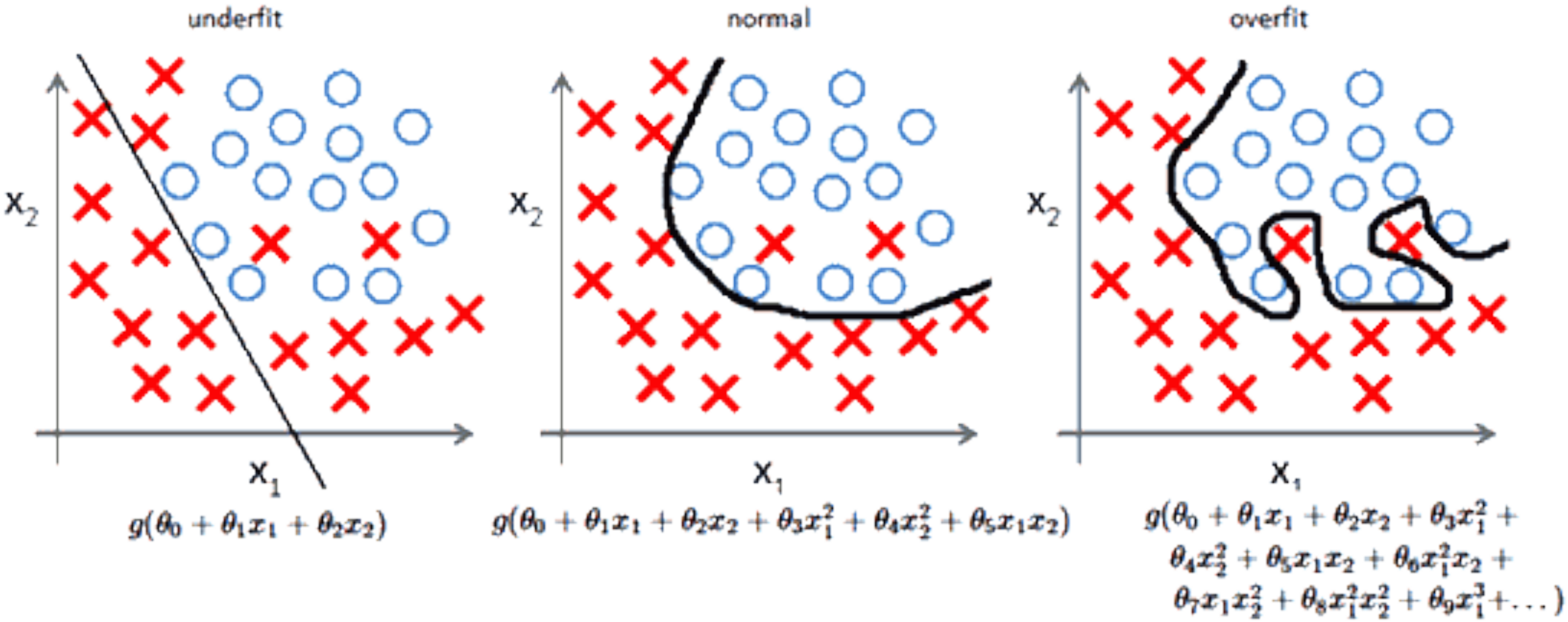
# Overfitting

- Model reproduce too well training data
  - In the extreme limit it will follow exactly the data ($L(w) \simeq 0$ )
- It might fail miserably on an independent data set (a validation/test data set)



# Overfitting

- Same happens also for classification (e.g. logistic regression )




In case of overfitting decision boundary follows the data

# Bias-Variance Trade Off

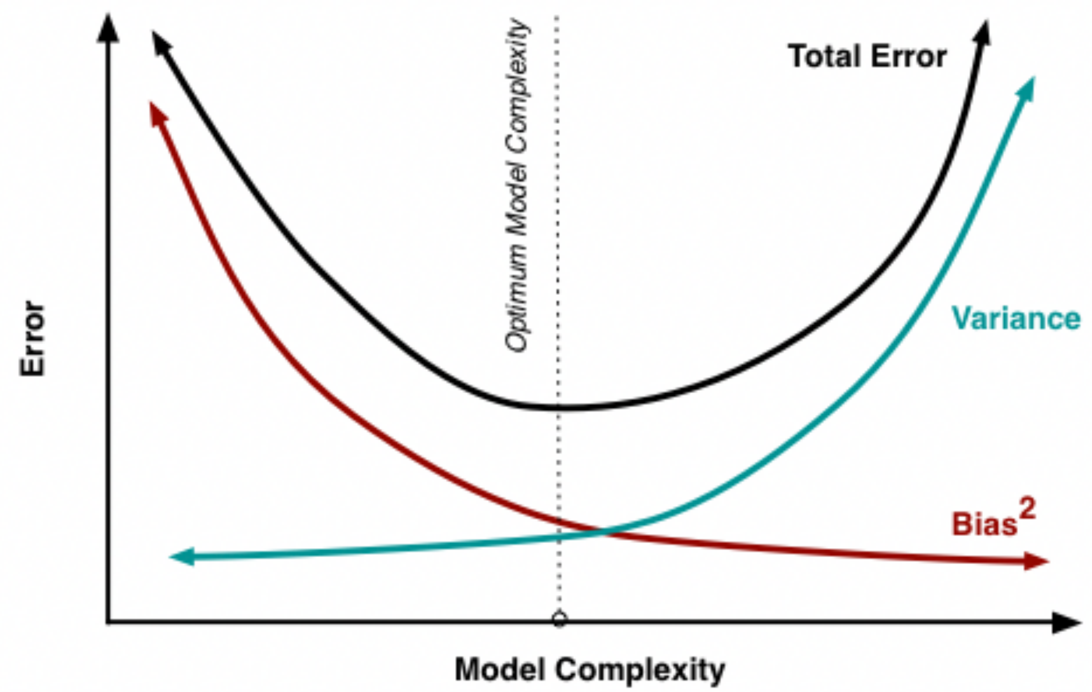
- Simple model under-fit: it will deviate from data (high bias) but not influenced by its peculiarity (low variance)
- Complex model over-fit: it will not deviate from data (low bias) but it will be very sensitive to the data (high variance)
  - **Bias** : systematic error of the model
  - **Variance**: sensitivity of prediction
- If model is more complex
  - will capture more data points ➜ lower bias
  - will move more to capture the data ➜ higher variance

# Bias - Variance Trade Off

Generalization Error

$$E[(y - h(x))^2] = \boxed{E[(y-\bar{y})^2]} + \boxed{(\bar{y} - \bar{h}(x))^2} + \boxed{E[(h(x) - \bar{h}(x))^2]}$$

$$= \text{noise} + (\text{bias})^2 + \text{variance}$$



# Regularization

- Method to find optimal model is to add a parameter constraint in the loss function
  - aim to trade some bias to reduce variance
- Modify loss function (e.g. for linear regression):

$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(\mathbf{x_i};\mathbf{w}))^2 + \lambda\Omega(\mathbf{w})$$

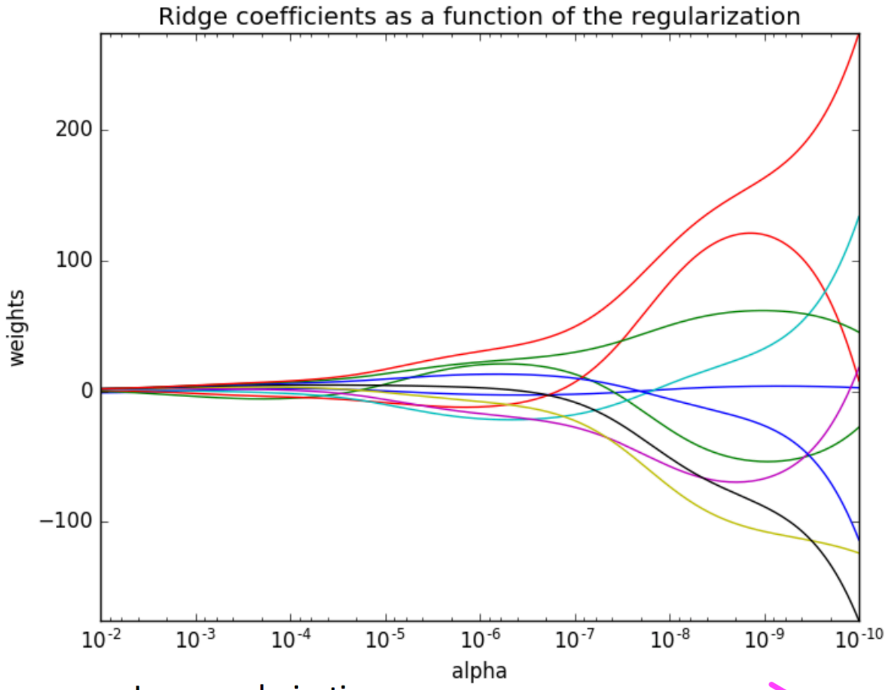
  - **L2 norm** $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2 = \sum_i w_i^2$
    - equivalent to Gaussian prior on the weights
  - **L1 norm** $\Omega(\mathbf{w}) = \|\mathbf{w}\| = \sum_i |w_i|$
    - equivalent to Laplace prior on the weights

# Regularization

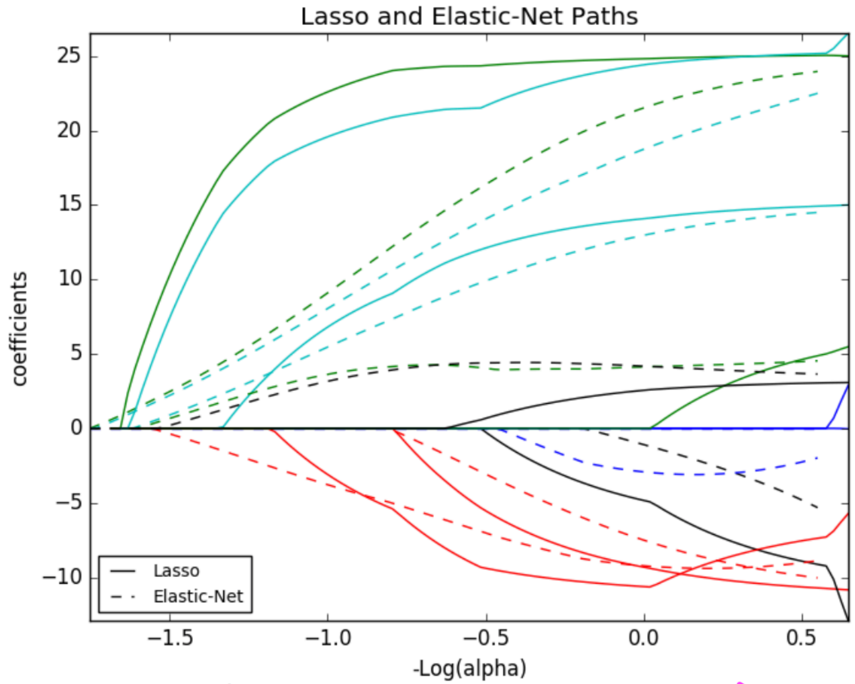
$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \alpha\Omega(\mathbf{w})$$

$$L2: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||^2$$

$$L1: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||$$




Less regularization →




Less regularization →

- L2 keeps weights small, L1 keeps weights sparse!

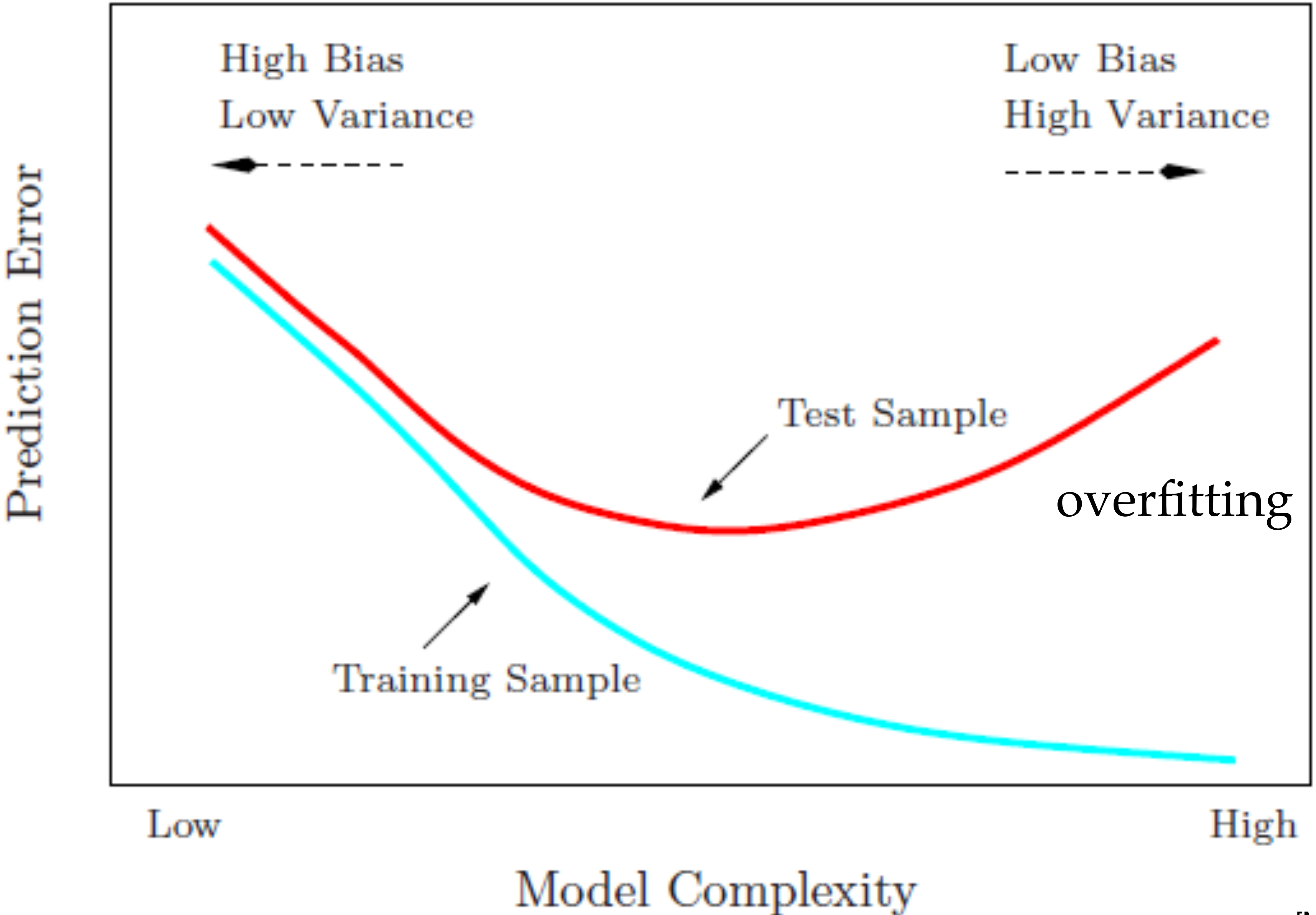
[M. Kagan]

# Hyper-parameter Optimisation

- How to find optimal regularisation parameter ?
- We need to perform an hyper-parameter optimisation to find the best total error
- Split the data in 3 samples:

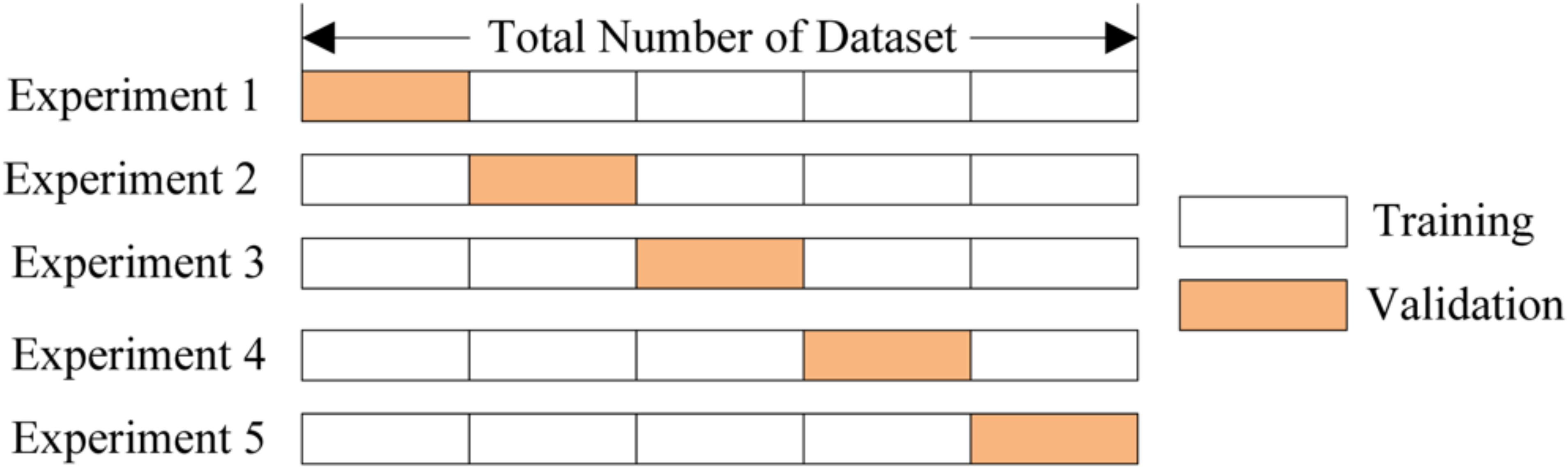

  - Training sample
    - used to train and fit the model
    - Find best parameter values
  - **Validation sample**
    - used to check the model and measure error as function of hyper-parameter
    - find best hyper-parameter values
  - **Test Sample**
    - Final check of the model when all parameters have been fixed
    - Need to be independent than validation since we have tune the model on the validation sample

# Hyper-Parameter Optimization




[M. Kagan]

# Cross Validation




- Divide data randomly in k-folds
  - Use (k-1) folds for training and 1 fold for validation
  - Repeat changing the validation set
- Use average estimate performances on the k-folds
- Can estimate variance on the performance
- Especially useful when data set is small

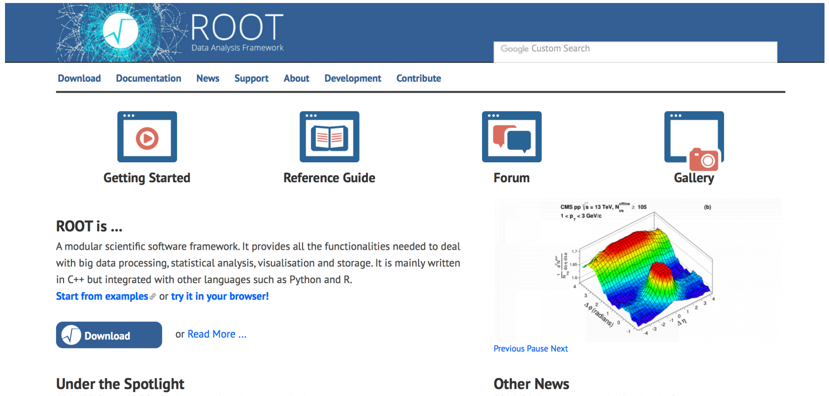
# Machine Learning Software

ROOT/TMVA

# ROOT

ROOT is a software toolkit which provides building blocks for:

- Data processing
- Data analysis
- Data visualisation
- Data storage




ROOT is written mainly in C++ (C++11 standard)

- Bindings for Python are provided.

**http://root.cern.ch**

Adopted in High Energy Physics and other sciences (but also industry)

- ~250 PetaBytes of data in ROOT format on the LHC Computing Grid
- Fits and parameters' estimations for discoveries (e.g. the Higgs)
- Thousands of ROOT plots in scientific publications

# TMVA

- ROOT Machine Learning tools are provided in the package TMVA (Toolkit for MultiVariate Analysis)
- Provides a set of algorithms for standard HEP usage
- Used in LHC experiment production and in several analysis (e.g. Higgs studies)
- Easy interface for beginners, powerful for experts
- Several active contributors and several features added recently (e.g. deep learning)

# TMVA

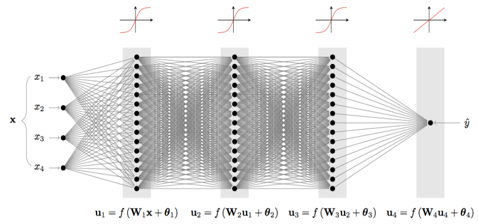
- TMVA is not only a collection of multi-variate methods. It is a
  - common interface to different methods
    - common interface for classification and regression
  - easy training and testing of different methods on the same dataset
    - consistent evaluation and comparison
    - same data pre-processing
  - several tools provided for pre-processing
  - embedded in ROOT
  - complete and understandable users guide

# TMVA Methods

The available methods are:

- Rectangular cut optimisation
- Projective likelihood estimation (PDE approach)
- Multidimensional probability density estimation (PDE - range-search approach)
- Multidimensional k-nearest neighbour classifier
- Linear discriminant analysis (H-Matrix and Fisher discriminants)
- Function discriminant analysis (FDA)
- Artificial neural networks (various implementations)
- Boosted/Bagged decision trees
- Predictive learning via rule ensembles (RuleFit)
- Support Vector Machine (SVM)

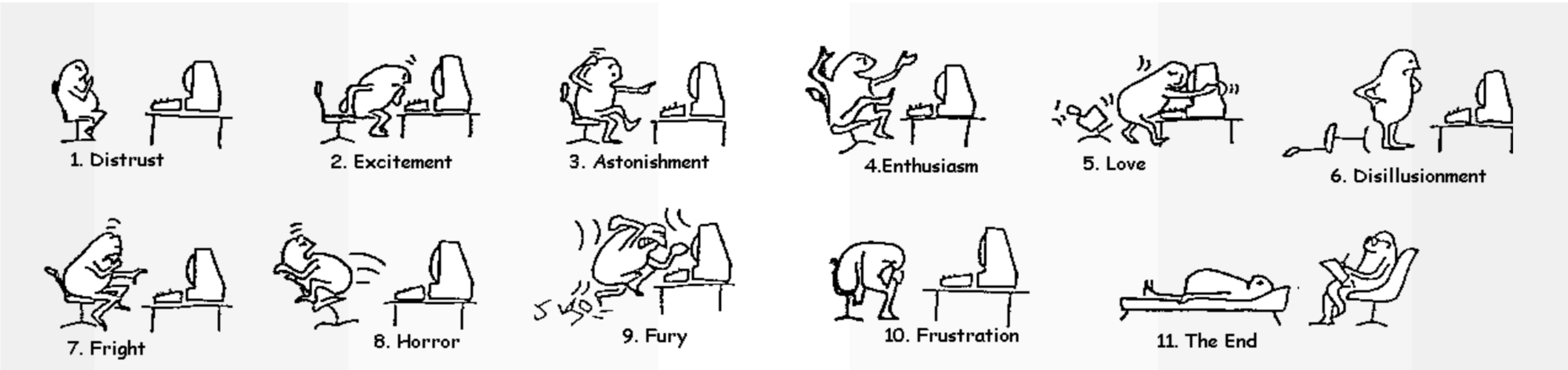
# New Features



New features added since 2016:

- Deep Learning
  - support for parallel training on CPU and GPU (with CUDA and OpenCL)
- Cross Validation and Hyper-parameter optimisation
- Improved loss functions for regression
- Interactive training and visualization for Jupyter notebooks
- new pre-processing features (variance threshold)

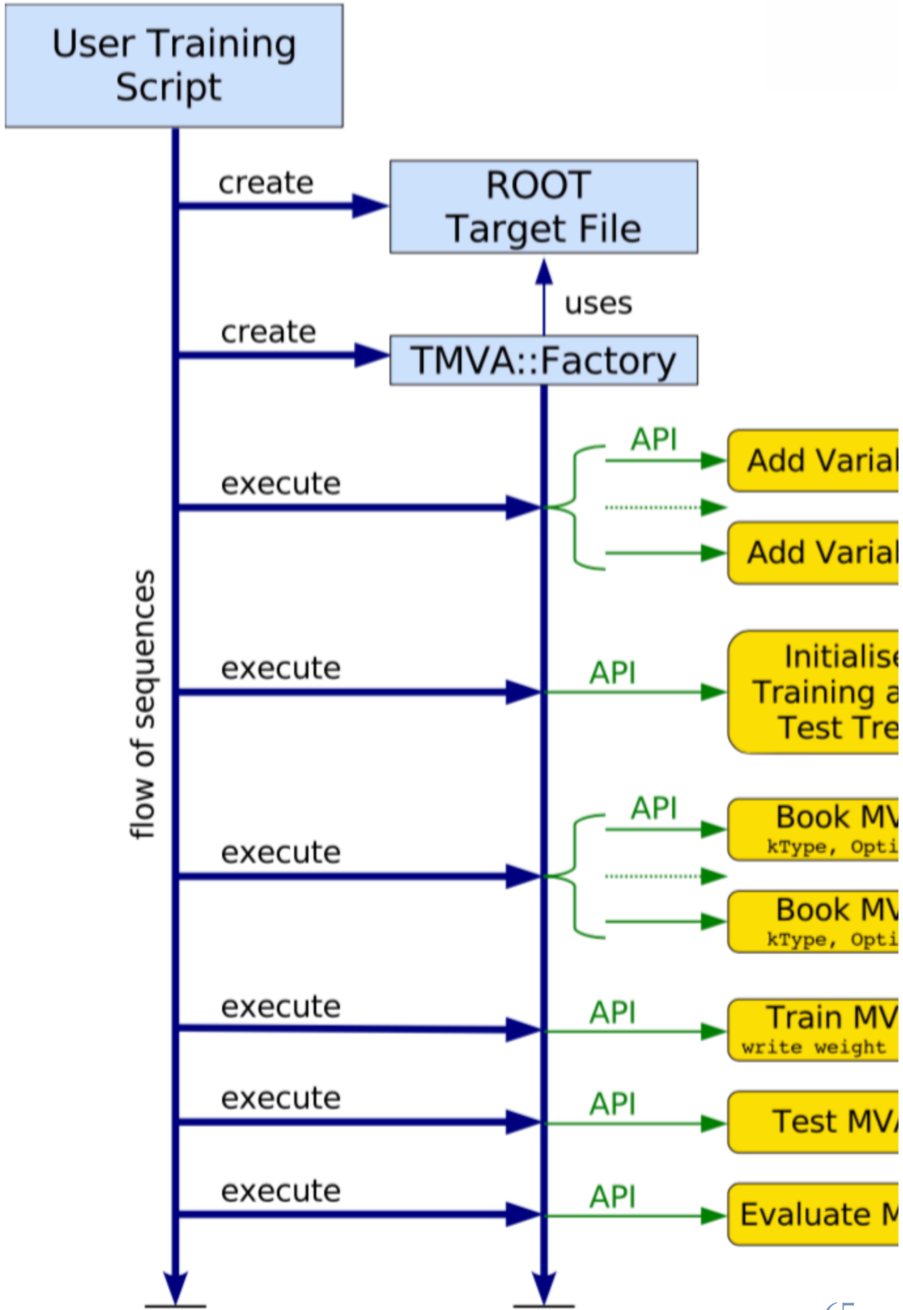
# Using TMVA



1. Distrust
2. Excitement
3. Astonishment
4.Enthusiasm
5. Love
6. Disillusionment
7. Fright
8. Horror
9. Fury
10. Frustration
11. The End

# Workflow in TMVA

- Reading input data
- Select input features and preprocessing
- **Training**
  - find optimal classification or regression parameters
    using data with known labels
    (e.g. signal and background MC events)
- **Testing**
  - evaluate performance of the classifier in an independent test sample
  - compare different methods
- **Application**
  - apply classifier / regressor to real data where labels are not known



# TMVA Custumizations and Features

TMVA supports:

- ROOT Tree input data (or ASCII, e.g. csv)
  - HSF support might come soon
- pre-selection cuts on input data
- event weights (negative weights for some methods)
- various method for splitting training/test samples
- k-fold cross-validation
- support variable importance
- hyper-parameter optimisations

# TMVA Session

```
void TMVAnalysis( )
{
 TFile* outputFile = TFile::Open( "TMVA.root", "RECREATE" );

 TMVA::Factory *factory = new TMVA::Factory( "MVAnalysis", outputFile,"!V");

 TFile *input = TFile::Open("tmva_example.root");

 factory->AddVariable("var1+var2", 'F');
 factory->AddVariable("var1-var2", 'F');   //factory->AddTarget("tarval", 'F');

 factory->AddSignalTree ( (TTree*)input->Get("TreeS"), 1.0 );
 factory->AddBackgroundTree ( (TTree*)input->Get("TreeB"), 1.0 );
 //factory->AddRegressionTree ( (TTree*)input->Get("regTree"), 1.0 );
 factory->PrepareTrainingAndTestTree( "", "",
 "nTrain_Signal=200:nTrain_Background=200:nTest_Signal=200:nTest_Background=200:!V" );

 factory->BookMethod( TMVA::Types::kLikelihood, "Likelihood",
 "!V:!TransformOutput:Spline=2:NSmooth=5:NAvEvtPerBin=50" );
 factory->BookMethod( TMVA::Types::kMLP, "MLP",
 "!V:NCycles=200:HiddenLayers=N+1,N:TestRate=5" );

 factory->TrainAllMethods(); // factory->TrainAllMethodsForRegression();
 factory->TestAllMethods();
 factory->EvaluateAllMethods();
 outputFile->Close();
 delete factory;
}
```

Create Factory

Add variables/ targets

Initialize Trees

Book MVA methods

Train, test and evaluate

[E. v. Toerne]

We will see better with a real example

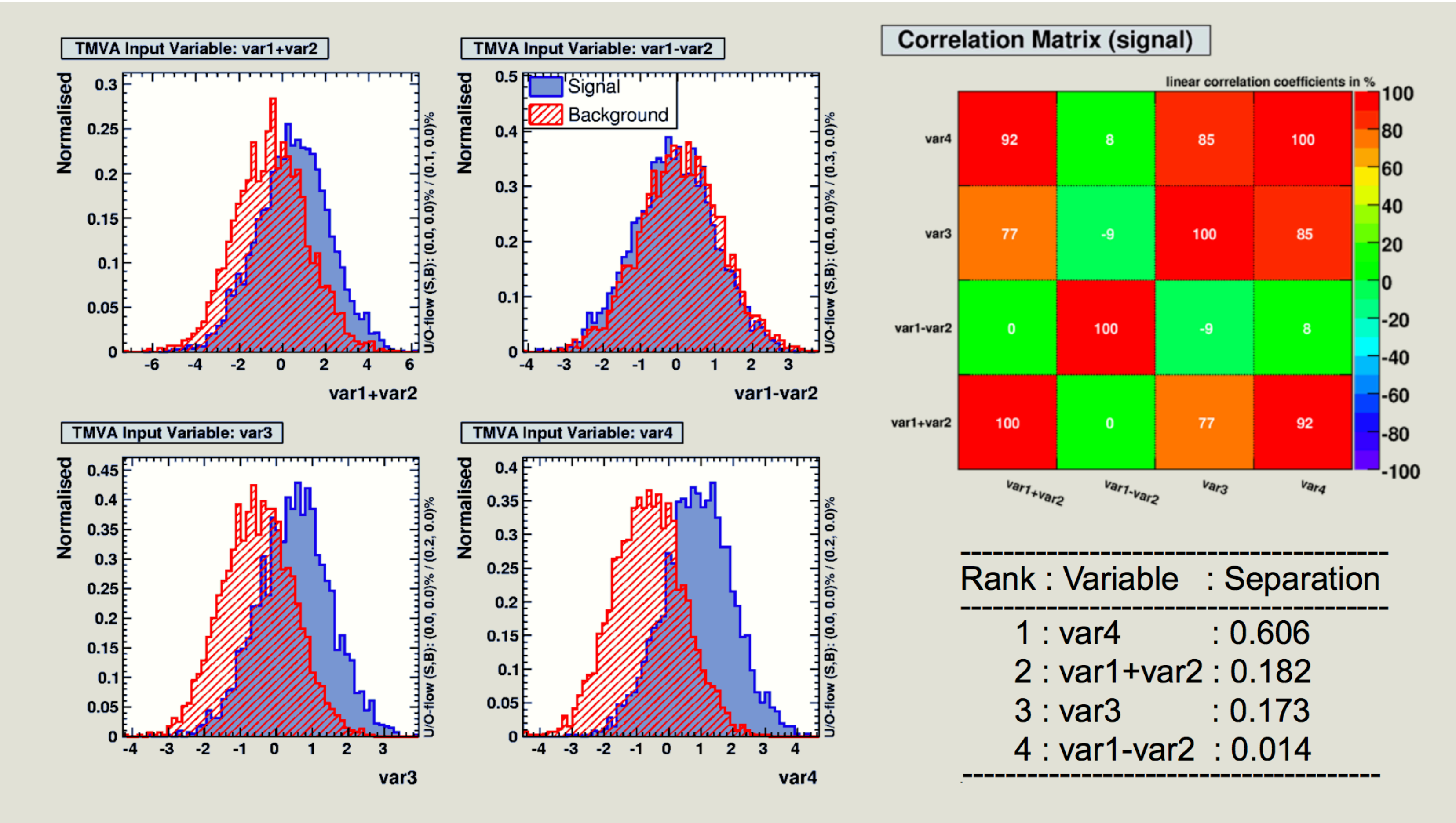
(e.g. TMVAClassification.C tutorial)

# TMVA Toy Example

4 Gaussian variable with linear correlations

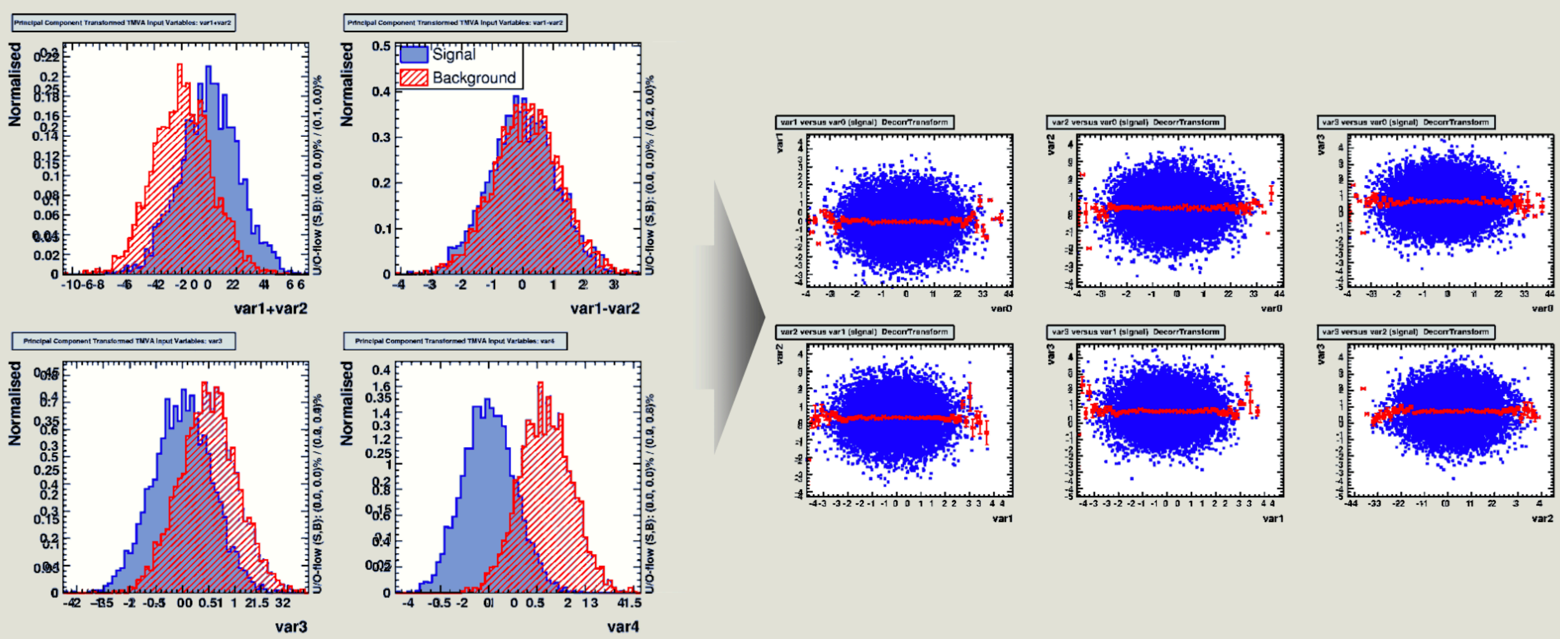
$$\{x_1 = v_1 + v_2,\ x_2 = v_1 - v_2,\ x_3 = v_3,\ x_4 = v_4\}$$

where $\{v_1, ..v_4\}$ are normal variables



# Pre-processing of the Input Variables

- Example: decorrelation of variable before training can be useful




- Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

Several others pre-processing available (see Users Guide)

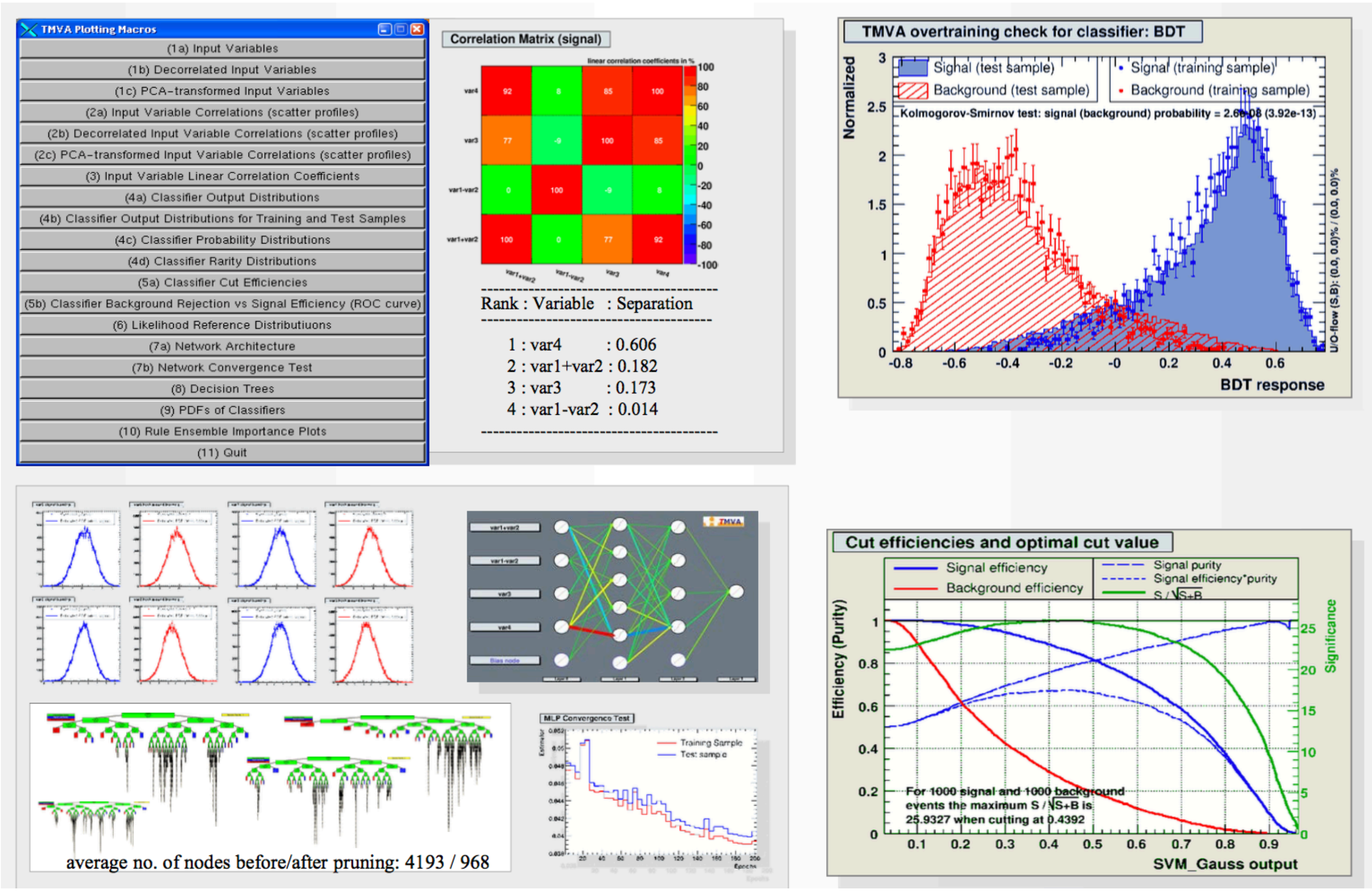
# Available Preprocessing

This is the list of available pre-processing in TMVA

- Normalization
- Decorrelation (using Cholesky decomposition)
- Principal Component Analysis
- Uniformization
- Gaussianization

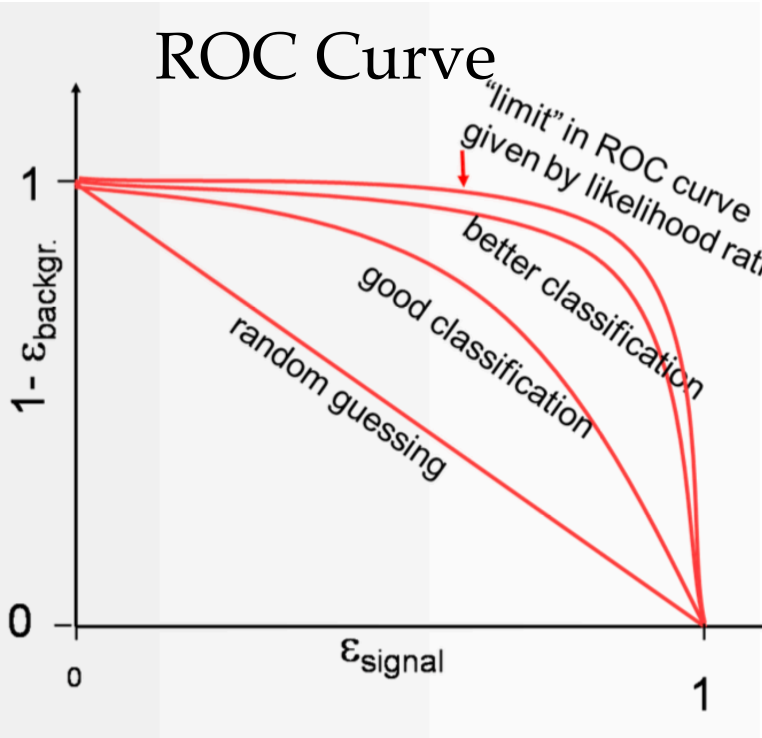
# TMVA GUI

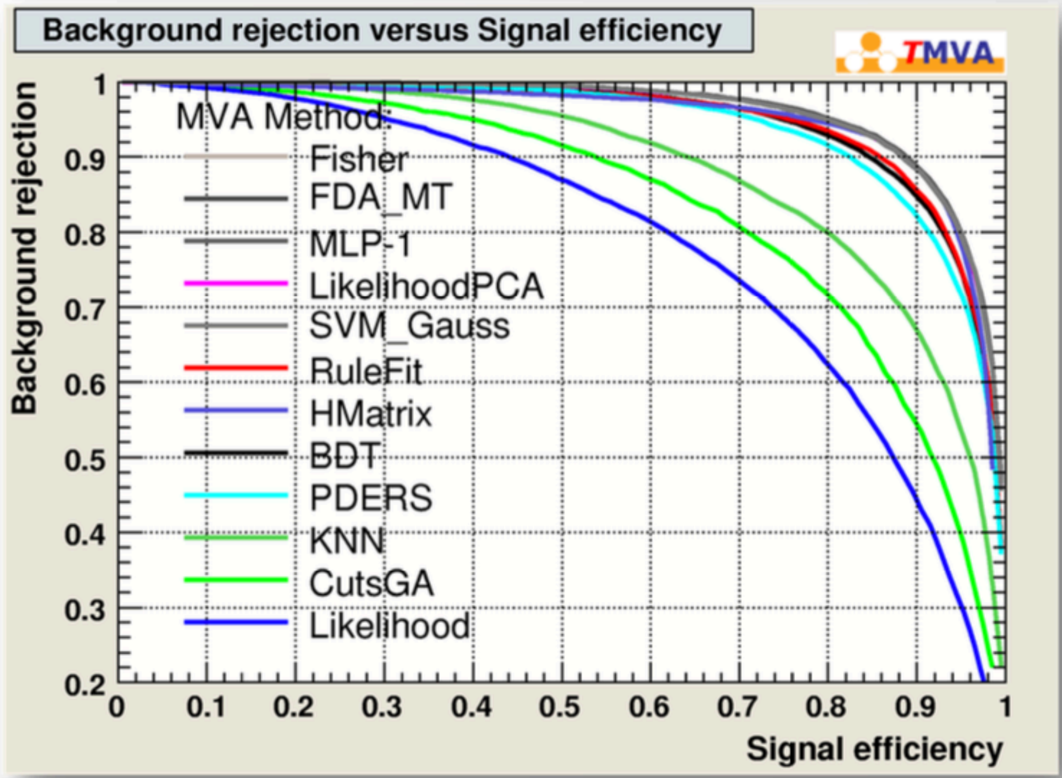
At the end of training + test phase TMVA produces an output file that can be examined with a special GUI (**TMVAGui**)



# ROC Curve in TMVA

For example from GUI one can obtain a ROC curve for each method trained and tested on an independent data set

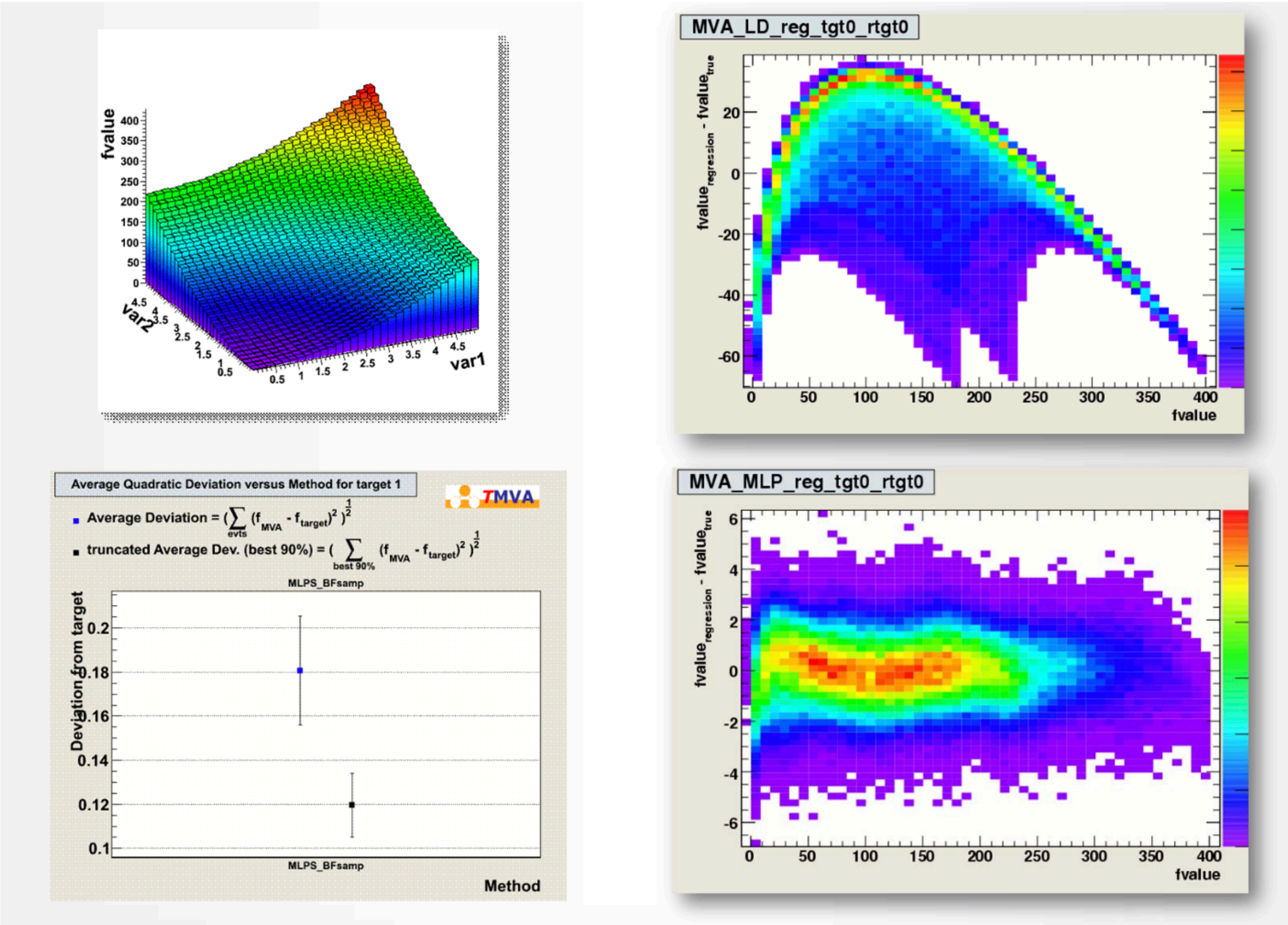



➜ Comparison of several methods

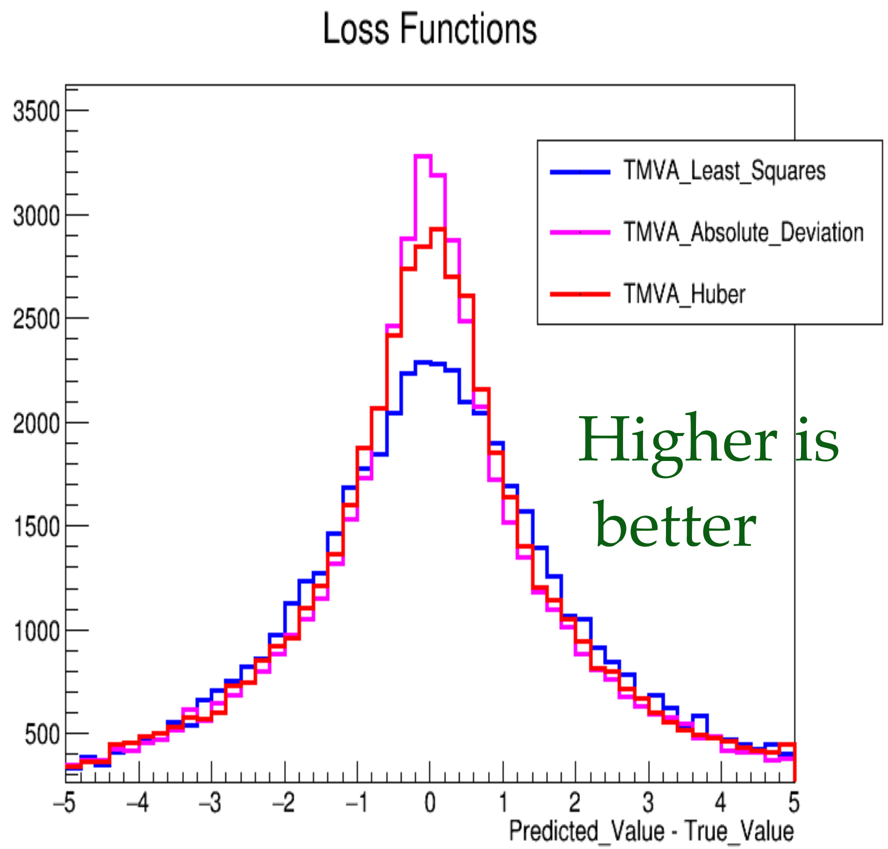
# TMVA Regression GUI

A dedicated GUI exists for regression (**TMVARegGui**)



# Regression in TMVA

- New Regression Features:
  - Loss function
    - Huber (default)
    - Least Squares
    - Absolute Deviation
    - Custom Function

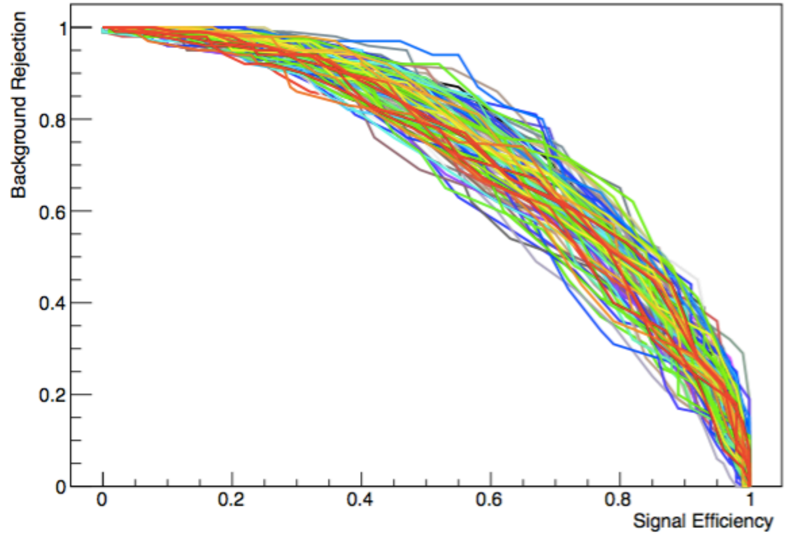



Higher is better

Important for regression performance

# Cross Validation in TMVA

- TMVA supports k-fold cross-validation

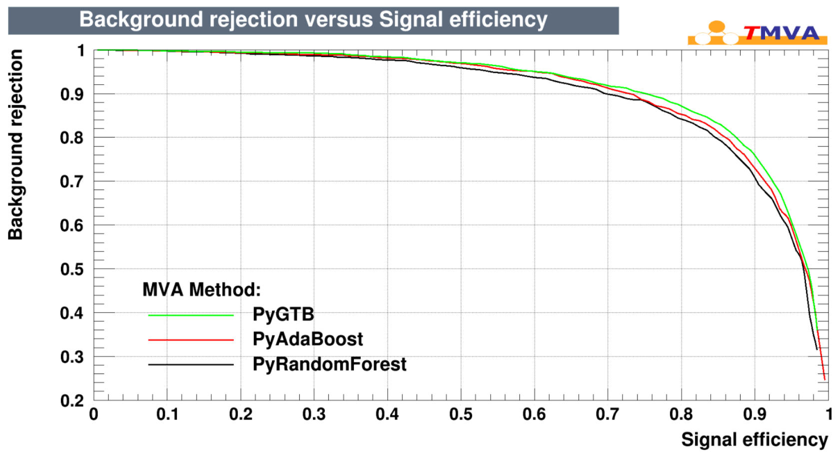




- Hyper-parameter tuning
  - find optimised parameters (BDT-SVM)
- Providing support for parallel execution
  - multi-process/multi-threads and on a cluster using Spark or MPI

# TMVA Interfaces

External tools are available as additional methods in TMVA and they can be trained and evaluated as any other internal ones.

- RMVA: Interface to Machine Learning methods in R
  - c50, xgboost, RSNNS, e1071
    - see http://oproject.org/RMVA
- PYMVA: Python Interface
  - **skikit-learn** with RandomForest, Gradiend Tree Boost, Ada Boost)
    - see http://oproject.org/PYMVA
  - **Keras** (Theano + Tensorflow)
    - support model definition in Python
    - see https://indico.cern.ch/event/565647/contributions/2308668/attachments/1345527/2028480/29Sep2016_IML_keras.pdf
  - Input data are copied internally from TMVA to Numpy array

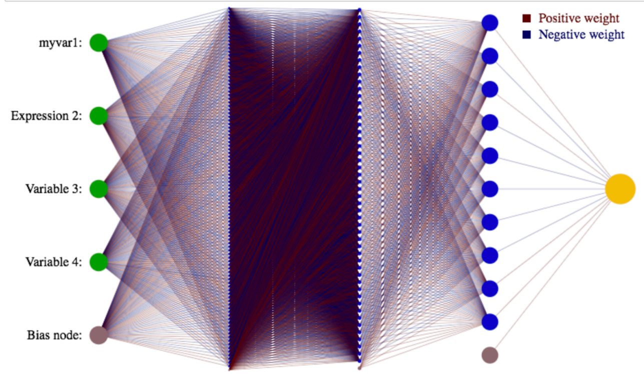


# Jupyter Integration

New Python package for using TMVA in Jupyter notebook (jsmva)

- Improved Python API for TMVA functions
- Visualisation of BDT and DNN
- Enhanced output and plots (e.g. ROC plots)
- Improved interactivity (e.g. pause/resume/stop of training)
- see example in SWAN gallery
  https://swan.web.cern.ch/content/machine-learning

```
In [23]: factory.DrawNeuralNetwork(dataset, "DNN")
```




```
In [24]: factory.DrawDecisionTree(dataset, "BDT") #11
```