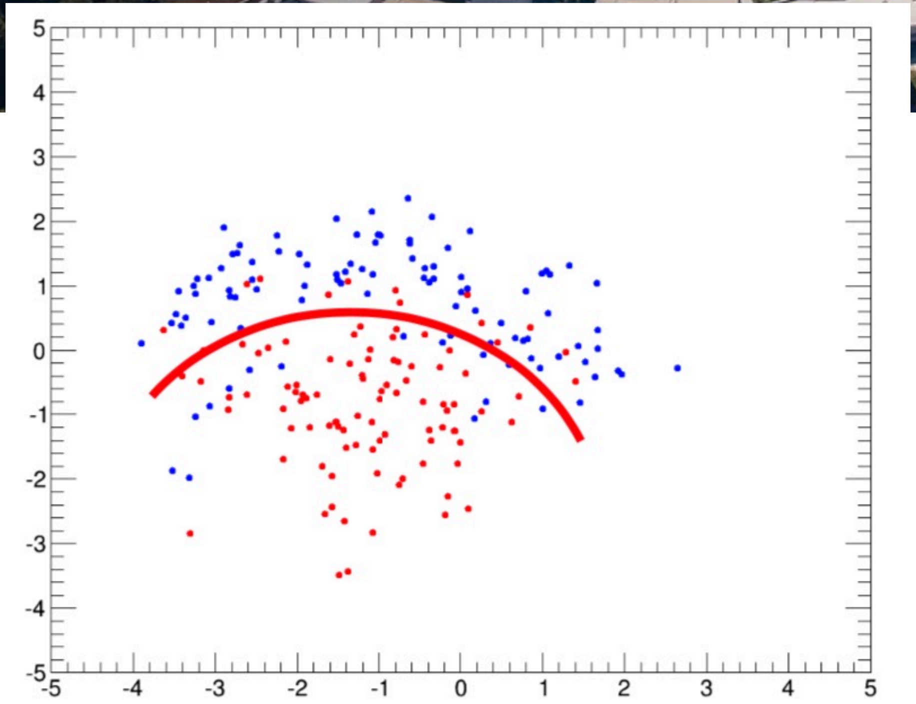
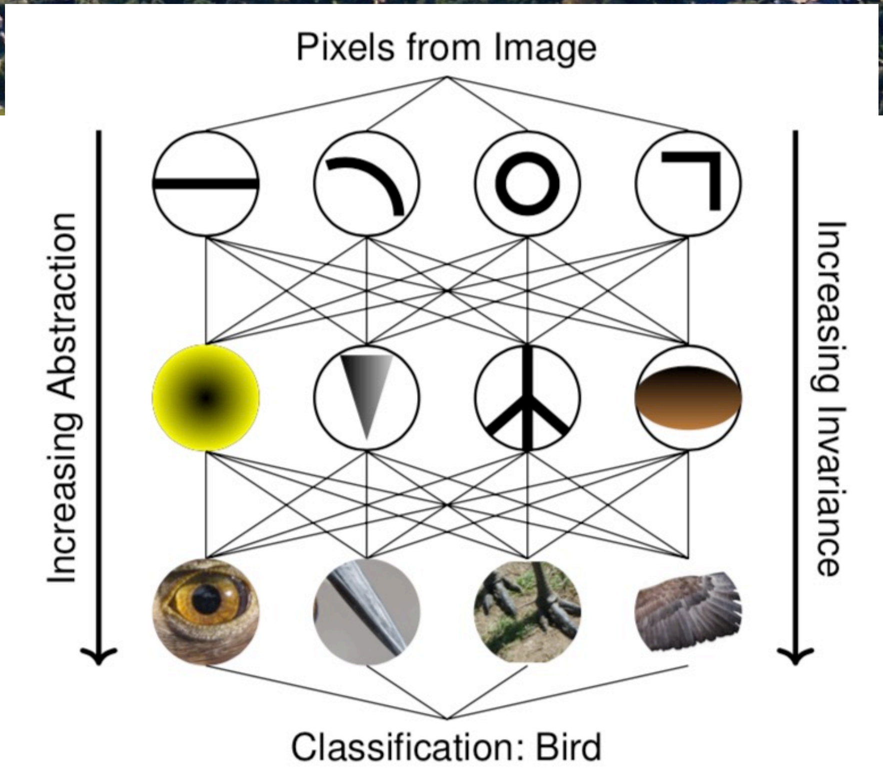
Statistical Methods in Data Analysis

Machine Learning

Andreas B. Meyer
DESY
6 - 10 March 2023





DESY.

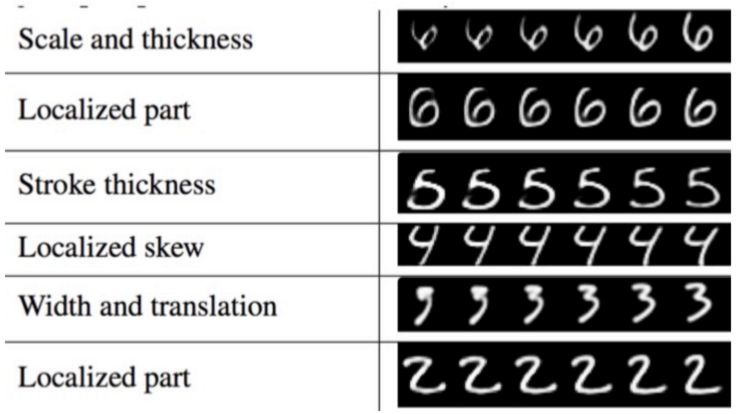
Menu

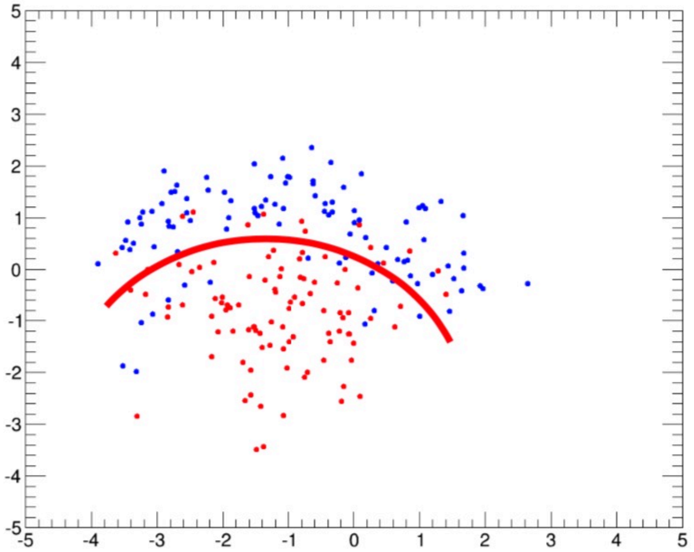
Multivariate Analysis

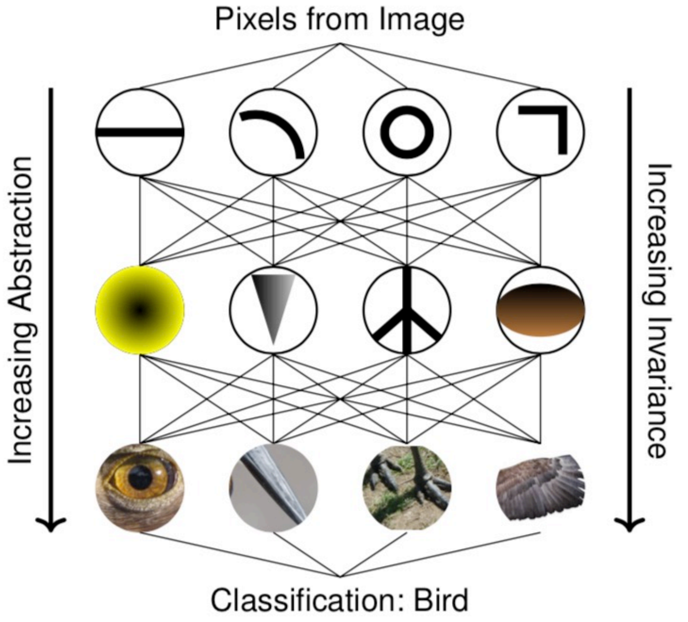
Tuesday
- Statistical and Systematic Uncertainties
- Probability
- Parameter Estimation

Wednesday
- Hypothesis Testing
- Confidence Intervals
- Profile Likelihood Ratio

Friday
- Classification
- Multivariate Analysis
- Machine Learning







Classification, Multivariate Analysis, Machine-Learning

Sources and Papers

Statistical Methods in Data Analysis", Terascale, March 2023: https://www.desy.de/~ameyer/da_desy23/

A.B.Meyer

- Statistical Methods in Data Analysis", KSETA lecture, Feb 2022: https://www.desy.de/~ameyer/da_kseta_22/
- Statistical Methods in Data Analysis", KSETA lecture, March 2021: https://www.desy.de/~ameyer/da_kseta_21/
- "Moderne Methoden der Datenanalyse", Course lecture at KIT, SoSe 2017, slides (in German): http://ekpwww.etp.kit.edu/~ameyer/da_sose17/index.html **Access to slides and material: (user: Students. pw: only)**

Papers and Articles:

- Robert Cousins: "Why isn't every physicist a Bayesian ?", Am.J.Phys. 65 (1995).
- Robert Cousins: "Lectures on Statistics in Theory: Prelude to Statistics in Practice" [arXiv]
- G.Cowan, Particle Data Group [pdg] 2020, chapter 40 [pdf] or full PDG book for download (80MB) [pdf]
- G.Cowan, K.Cranmer, E.Gross, O.Vitells: "Asymptotic formulae for likelihood-based tests of new physics" [arXiv]
- ATLAS and CMS Collaborations: "Procedure for the LHC Higgs boson search combination" [CDS]
- T.Junk: "Confidence level computation for combining searches with small statistics", NIM, A 434 (1999) 435-443
- A.Read: "Presentation of search results: the CL_s technique", J.Phys.G: 28 (2002)

Many thanks for discussions, material and help go to:

- G. Quast (KIT), R. Wolf (KIT), O. Behnke (DESY), C. Autermann (Aachen), Th. Keck (KIT), Jan Kieseler (CERN)

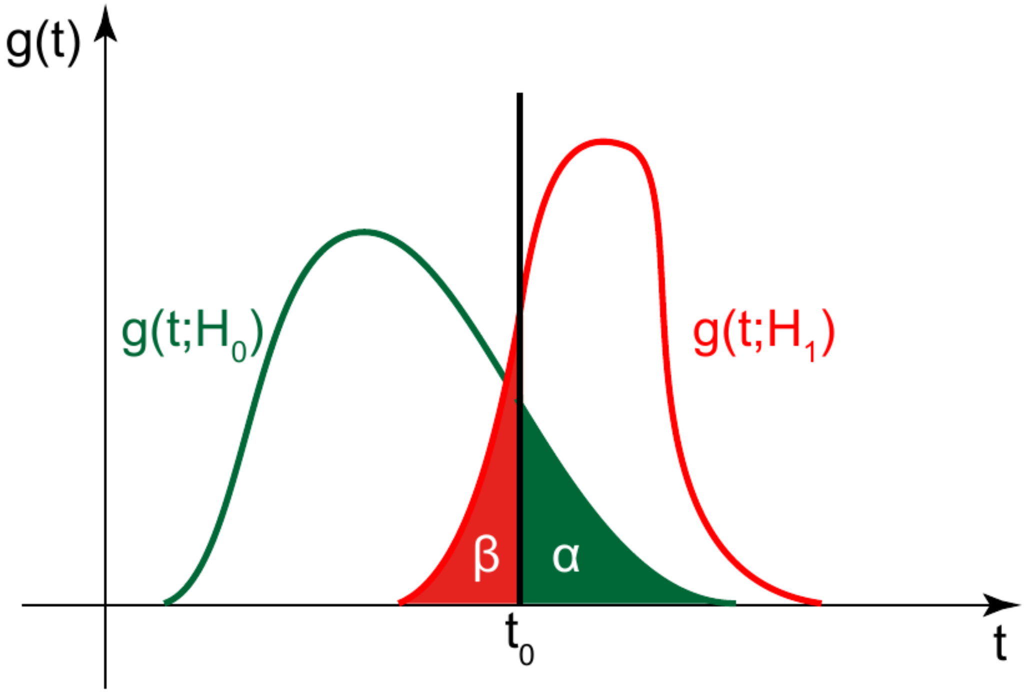
Recap

Hypothesis Testing

Procedure

1. Determine PDF $g(t;H_i)$ for test statistic t
2. Define significance level α (typically 5%)
    - critical value t_0: reject null hypothesis or not
    - in practice, α depends on goal
        - high efficiency ε or high purity p ?

$$\epsilon = 1 - \alpha \qquad\qquad p = \frac{(1-\alpha)N_0}{(1-\alpha)N_0 + \beta N_1}$$



- separation power $1-\beta$

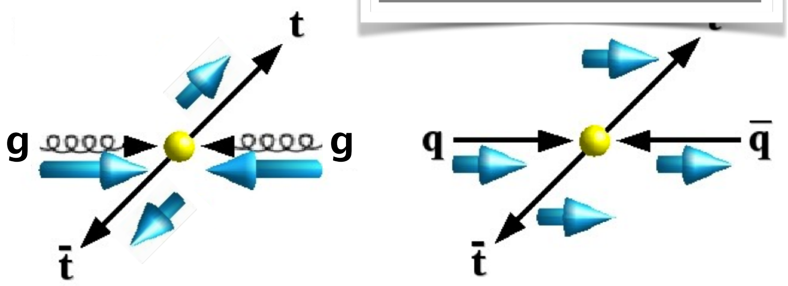
Note: trivially, no separation if no separation power
=> large $1-\beta$ is fundamentally more important than small α

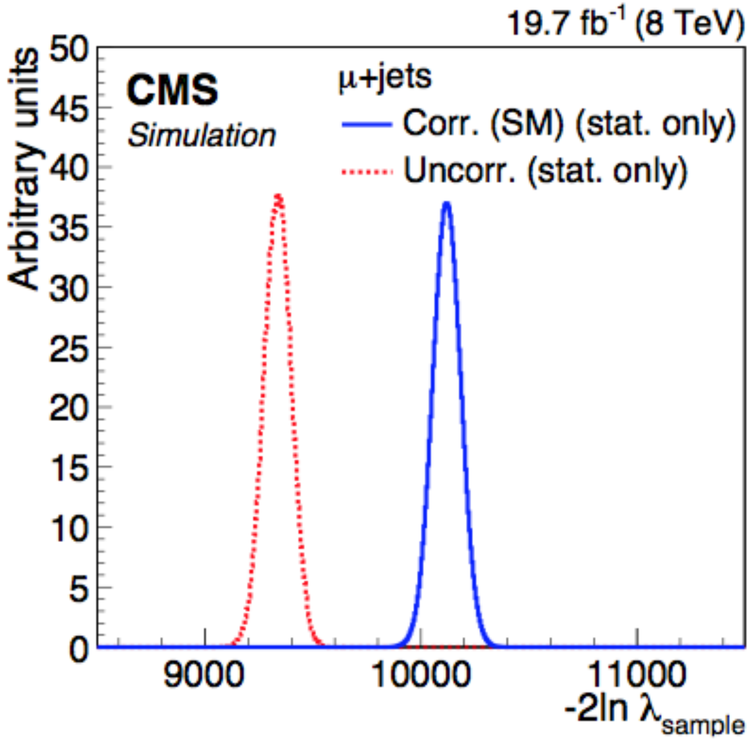
3. Determine p-value of the measurement
   p-value is probability that values $t > t_0$ are measured, assuming that H_0 is true.
   (note: p-value is an estimator derived from the measurement, i.e. a random number)

Example: Spin correlations of top-quark pairs

CMS TOP-13-015

- New Physics could change relative top-quark spins → angular distributions
  - H_0: spin-correlation follows Standard Model (Spin 1/2 particle)
  - H_1: no spin correlation
- Construction of the test statistic → log-likelihood-difference
- Sample likelihoods from pseudo-experiments for H_0 and H_1
- Statistical uncertainties ~ Gaussian shape → central limit theorem



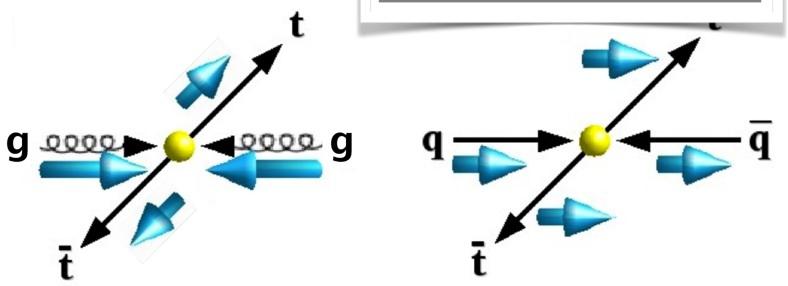


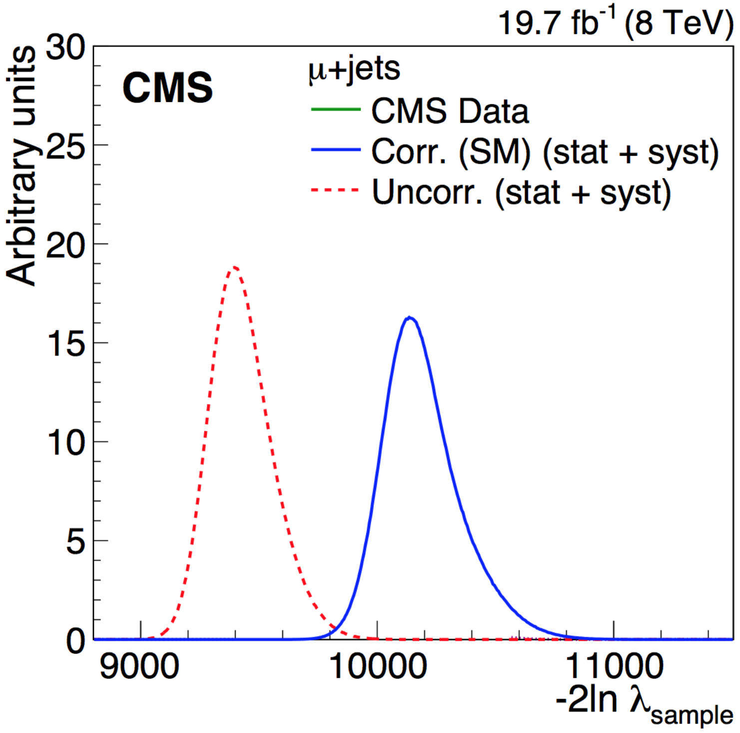
Figures from Ph.D. thesis K.Beernaert, U Gent, 2015

Example: Spin correlations of top-quark pairs

CMS TOP-13-015

- New Physics could change relative top-quark spins → angular distributions
  - H_0: spin-correlation follows Standard Model (Spin 1/2 particle)
  - H_1: no spin correlation
- Construction of the test statistic → log-likelihood-difference
- Sample likelihoods from pseudo-experiments for H_0 and H_1
- Statistical uncertainties ~ Gaussian shape → central limit theorem
- In addition, systematic uncertainties → wider PDF, no longer Gaussian



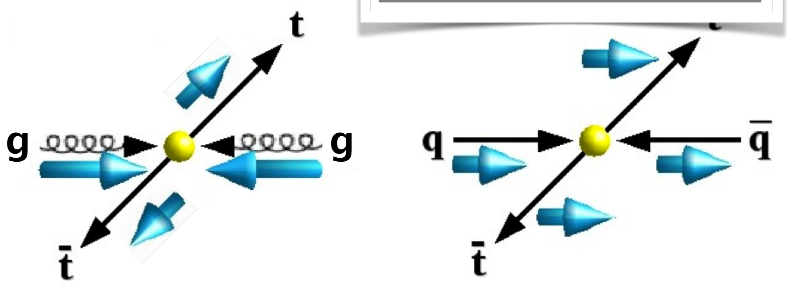


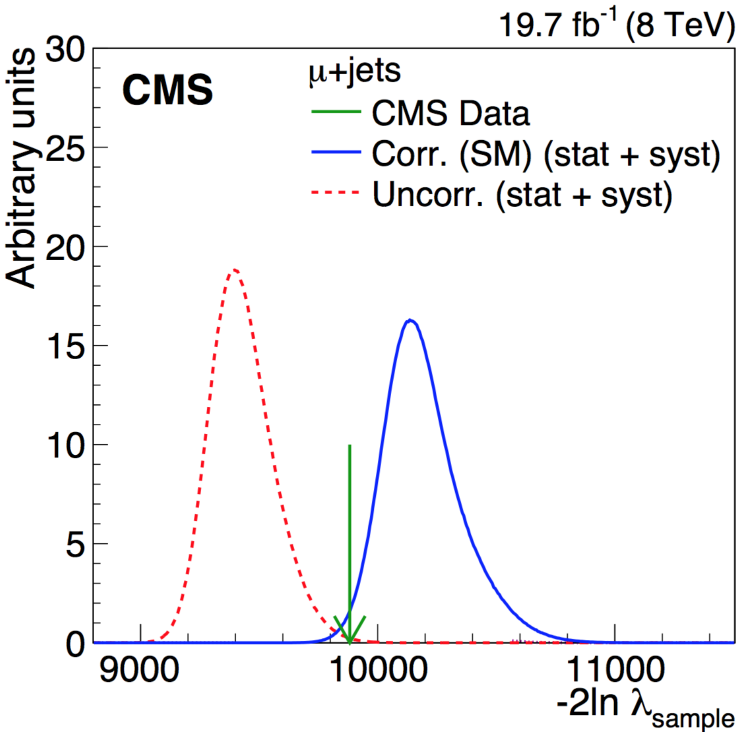
Figures from Ph.D. thesis K.Beernaert, U Gent, 2015

Example: Spin correlations of top-quark pairs

CMS TOP-13-015

- New Physics could change relative top-quark spins → angular distributions
  - H_0: spin-correlation follows Standard Model (Spin 1/2 particle)
  - H_1: no spin correlation
- Construction of the test statistic → log-likelihood-difference
- Sample likelihoods from pseudo-experiments for H_0 and H_1
- Insert data → result:
  - 2.2σ (p-value 1.3%) consistent with standard model (H_0)
  - 2.9σ (p-value 0.2%) for "uncorrelated" (H_1)





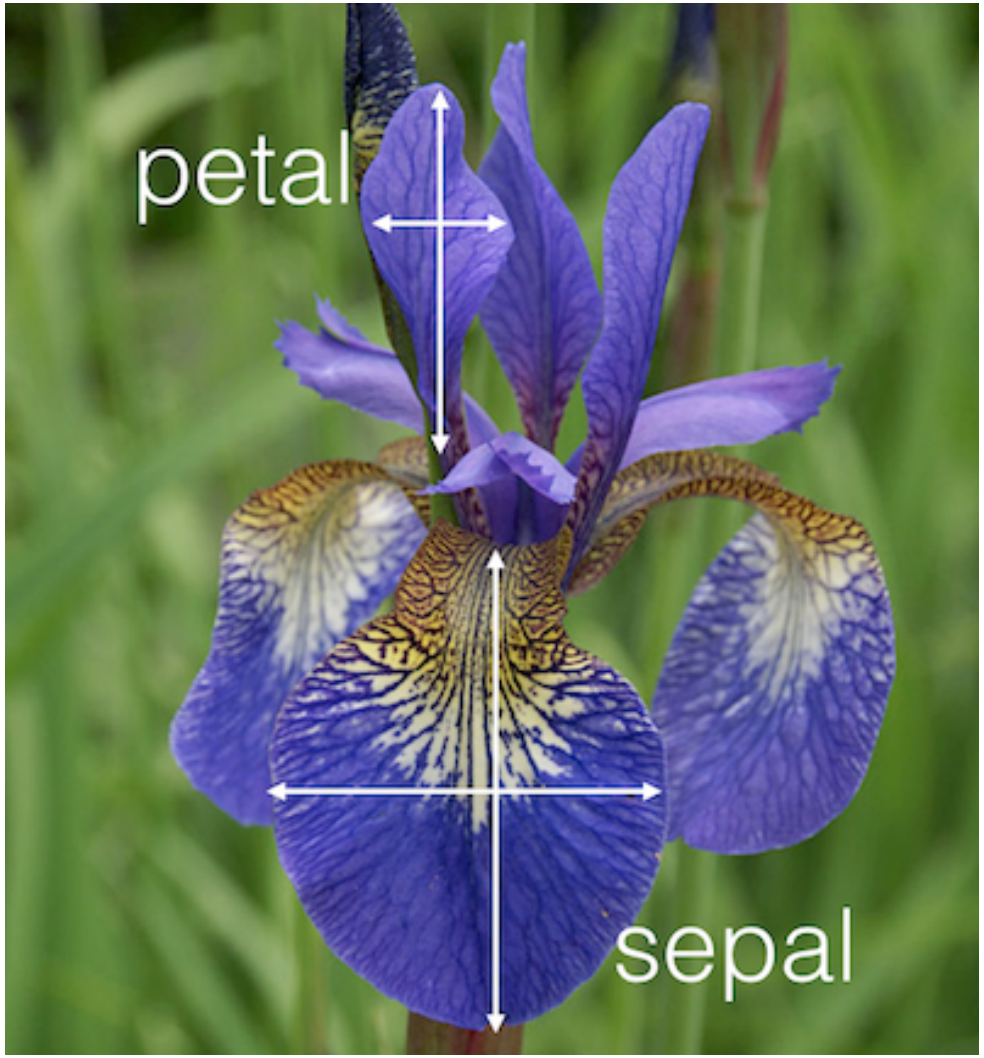
Figures from Ph.D. thesis K.Beernaert, U Gent, 2015

Classification

Classification

Outline

- Linear discriminators
- Supervised learning
- Boosted decision trees
- Artificial neural networks
- Deep-Learning

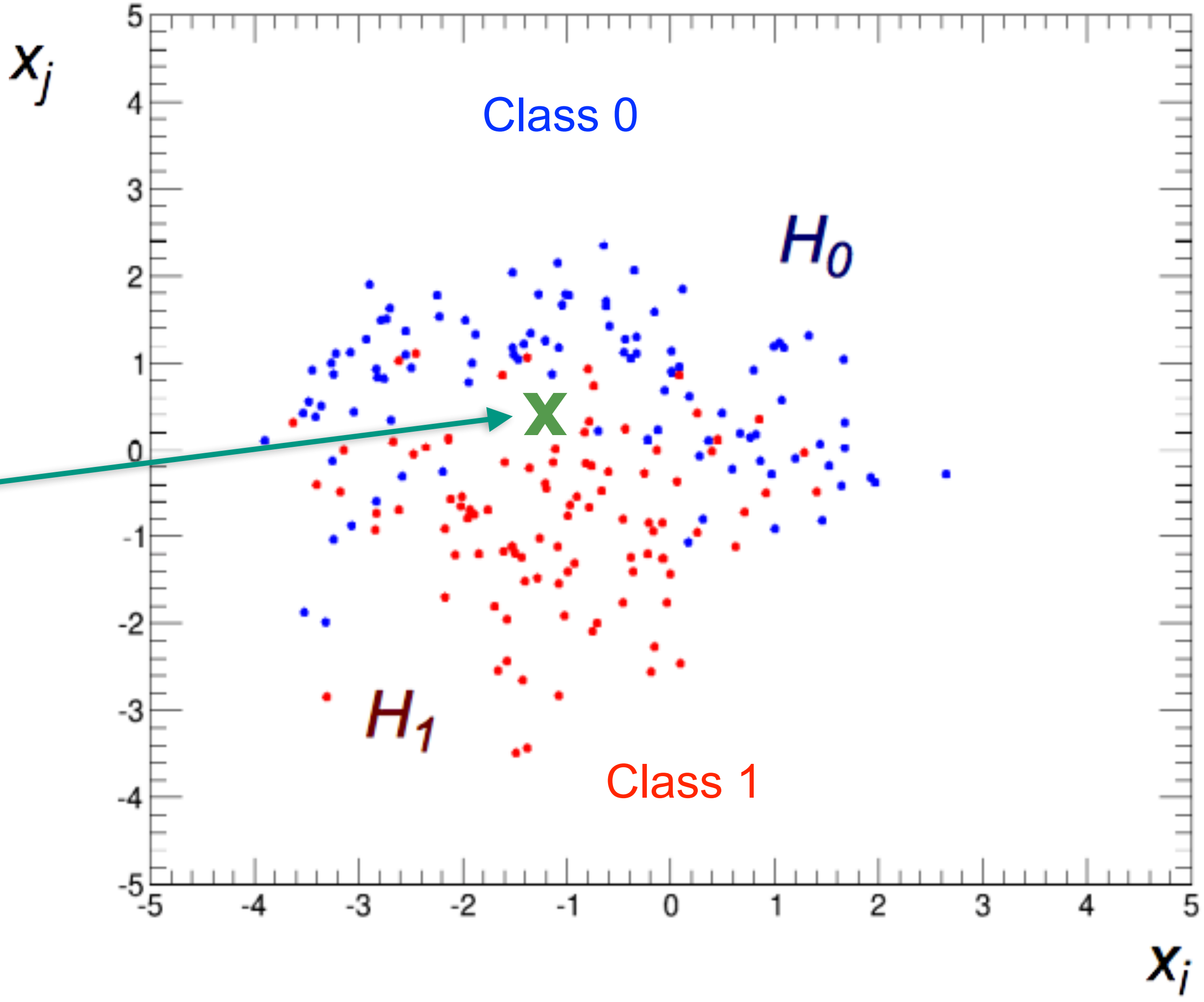


Fisher, R. A. (1936), The use of multiple measurements in taxonomic problems, Annals of Eugenics, 7: 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x

Multivariate Analysis

- Assign a given event **x** to a class
  - Random event is described by feature vector $x_1, \ldots x_n$
  - Class k is defined by PDF $f_k(x_1, \ldots x_n)$

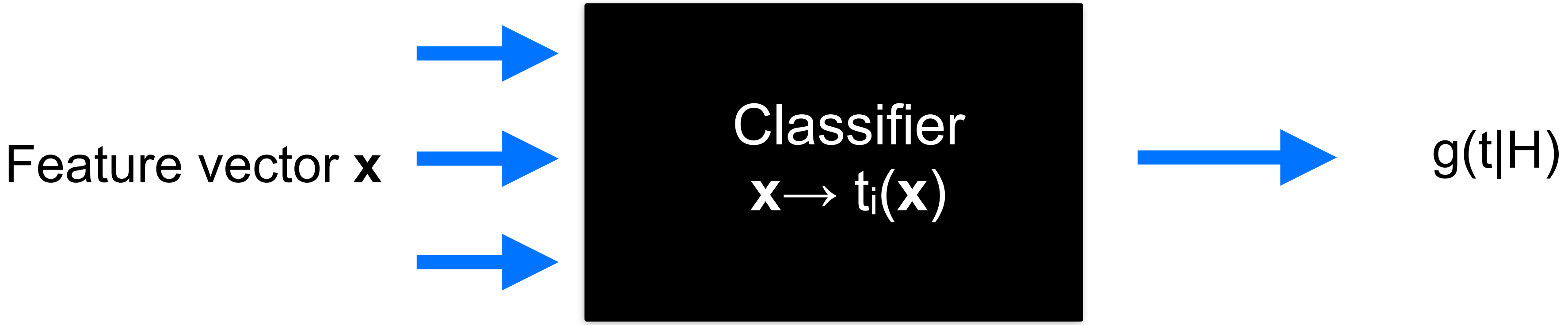
Does this event belong to Class 0 or Class 1 ?



What is a good test statistic ?

Multivariate Analysis

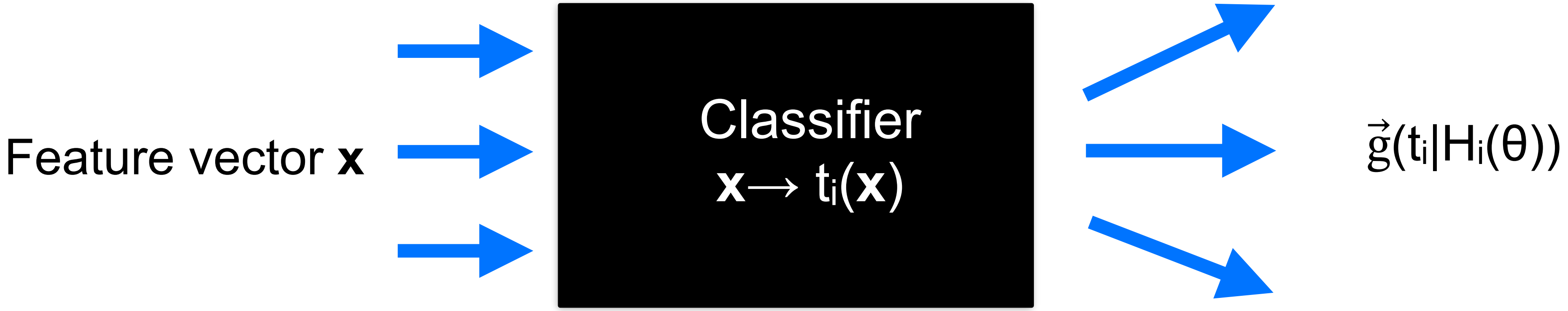
- Test of hypothesis H:



- Determine PDF g(**t**|H) of the test statistic t(**x**) for the hypothesis H
  - Particle physics: in most cases use Monte Carlo to determine g(**t**|H)
- Multivariate analysis (MVA):
  - Combine many observables into one (or several) test statistics t_i(**x**)
  - Take correlations between feature vector components $x_{1...n}$ into account

Multivariate Analysis

- Simultaneous test of several composite hypotheses $H_i(\theta)$



- Determine PDF $\vec{g}(\mathbf{t}|H_i(\theta))$ of the test statistics $t_i(\mathbf{x})$ for multiple hypotheses H_i
  - Particle physics: in most cases use Monte Carlo to determine $\vec{g}(\mathbf{t}|H_i(\theta))$
- Multivariate analysis (MVA):
  - Combine many observables into one (or several) test statistics $t_i(\mathbf{x})$
  - Take correlations between feature vector components $x_{1...n}$ into account
- Classification assigns a discrete label. In regression, a continuous quantity, $g=g(t|\theta)$, is determined

Curse of Dimensionality

Feature space with many dimensions

- Density distribution (PDF)

2d

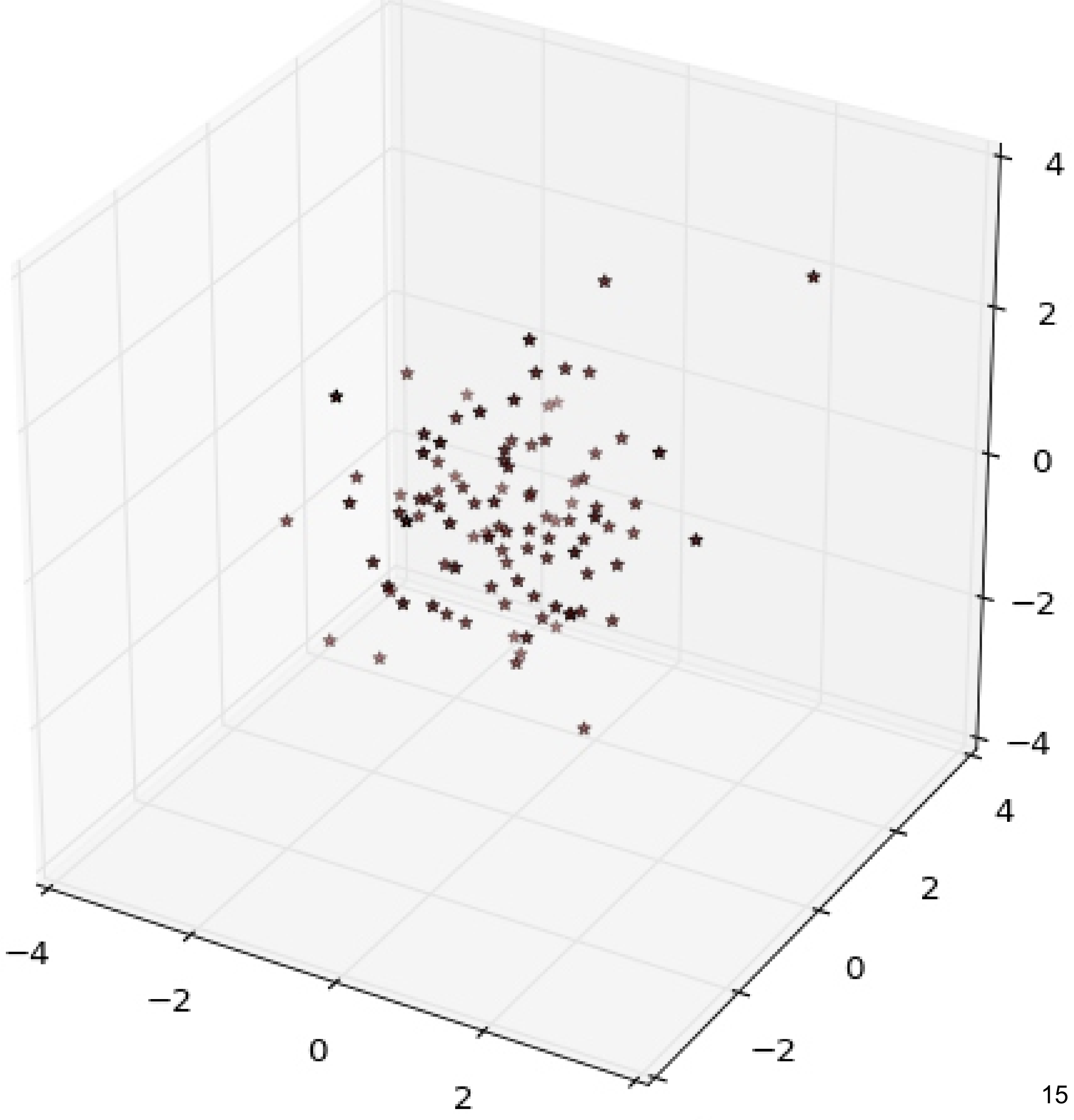
1d

Curse of Dimensionality

Feature space with many dimensions

◉ Density distribution (PDF)

3d



- a d-dimensional histogram (with N entries and n_b bins/dim.) is essentially empty

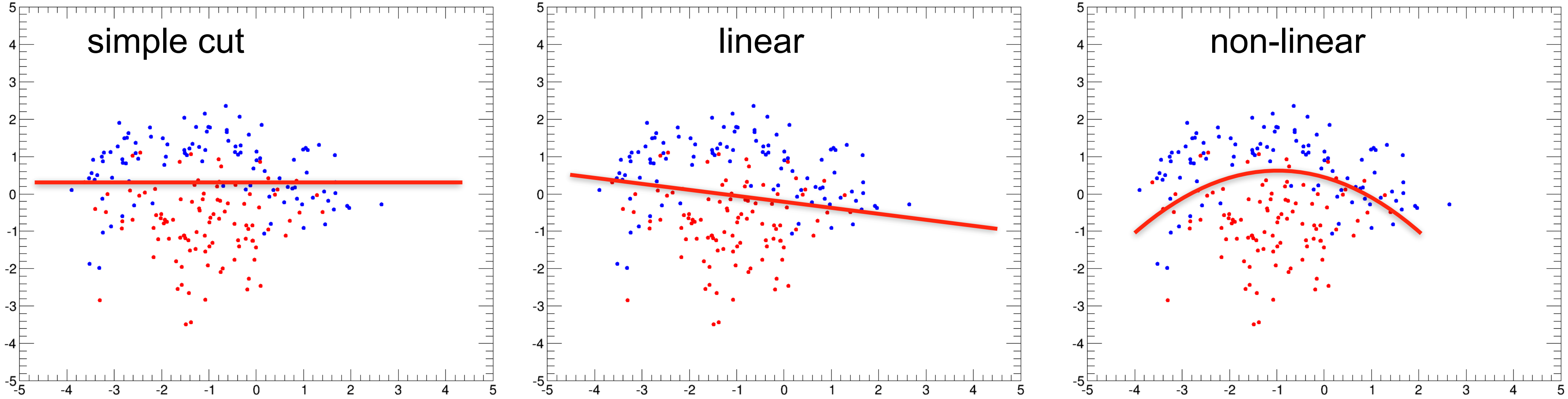
$$\frac{N}{n_b^d} \to 0, \;\text{for } d \to \infty$$

- Constant density: need n^d evts

- In n-dimensions:
  PDF usually not very well known

Classifier

Test Statistic

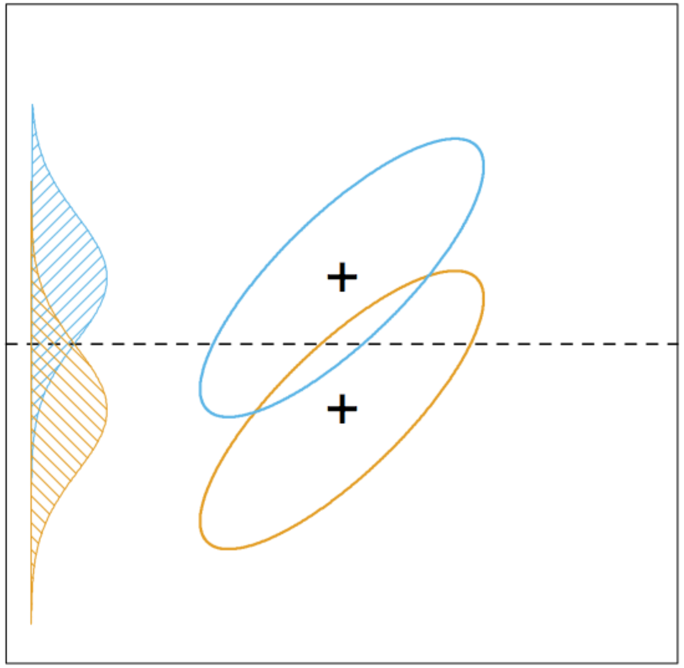


- Linear approaches can be treated analytically
  - e.g. Fisher discriminant
  - Many classification problems can be linearised by variable transformation (with or w/o approximation)
- Non-linear methods:
  - analytic approach usually impossible
  - use algorithmic approach to determine optimal test statistic, e.g. machine learning

Linear Discriminators

Hypothesis test by linear discriminant analysis

- Determine test statistic $t(\vec{x})$ that provides best possible separation between signal and background

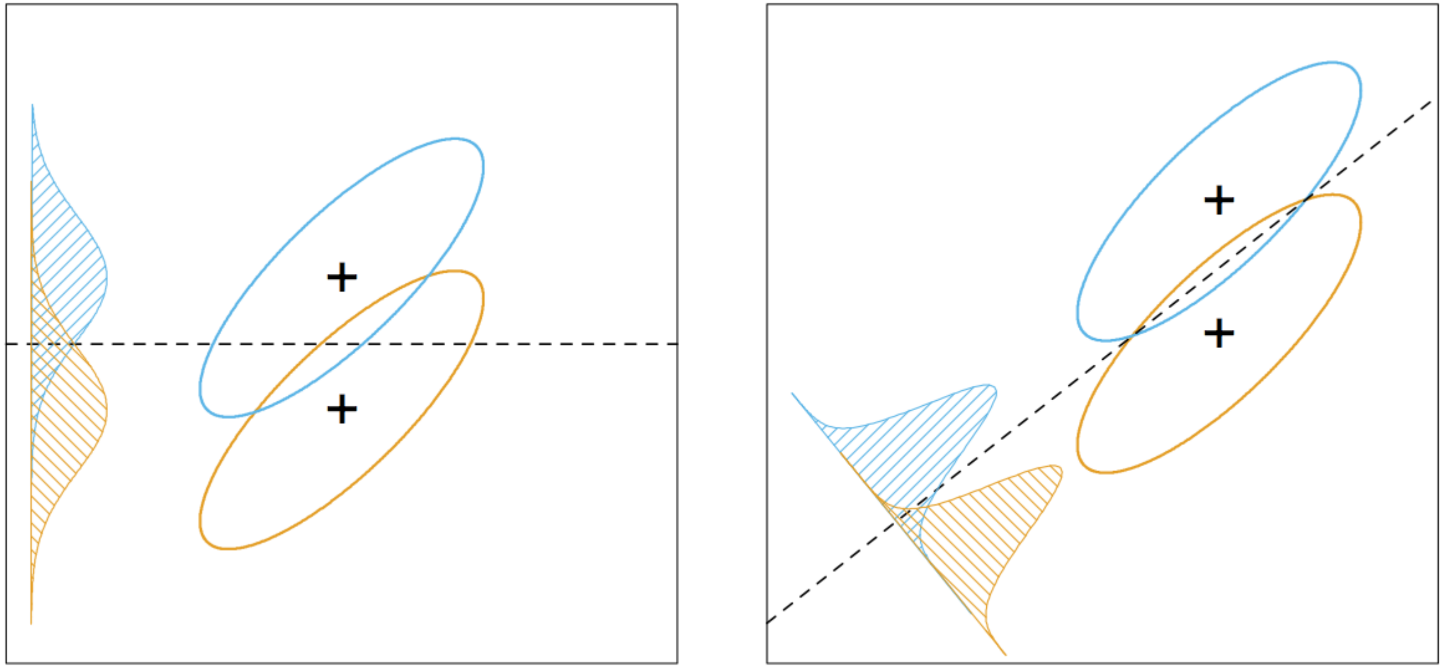


Linear Discriminators

Hypothesis test by linear discriminant analysis

- Determine test statistic $t(\vec{x})$ that provides best possible separation between signal and background.
- In other words: choose coordinate and parameters such that distributions are optimally separated

Decorrelating parameters: cf previous lecture p47-49

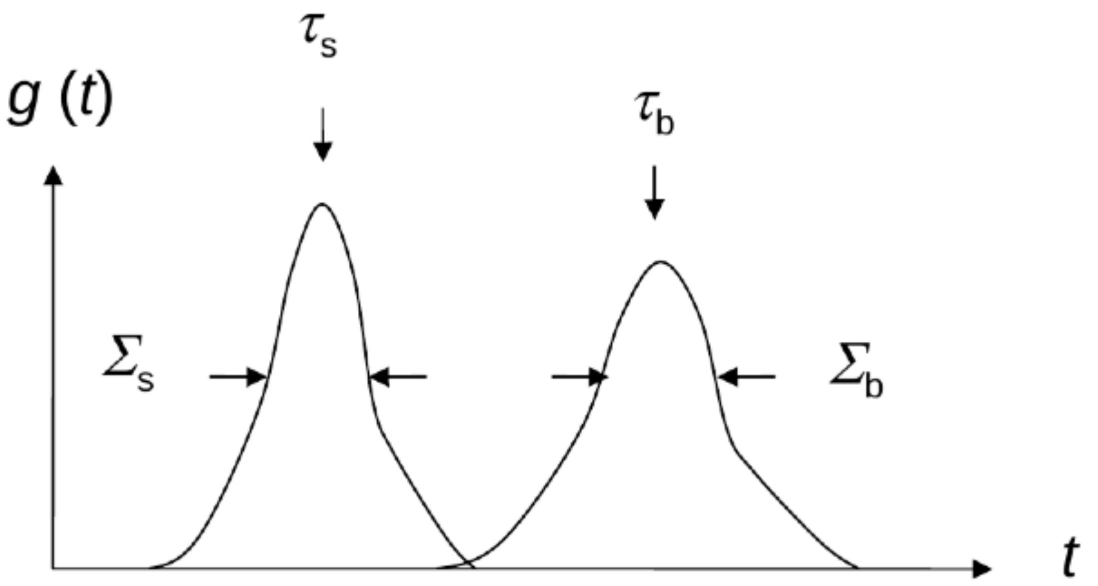


Elements of Statistical Learning (2nd Ed.), © Hastie, Tibshirani & Friedman 2009

Fisher Discriminant

- Ansatz: linear test statistic $t(\vec{x}) = \sum_{i=1}^{n} a_i x_i = \vec{a}^T \vec{x}$
- Choice of parameters: optimal separation when
  - the difference between the means $|\tau_s - \tau_b|$ is large
  - the sum of variances $\Sigma_s^2 + \Sigma_b^2$ is small
- Fisher discriminant:
  - maximize objective function

$$J(\vec{a}) = \frac{(\tau_s - \tau_b)^2}{\Sigma_s^2 + \Sigma_b^2}$$

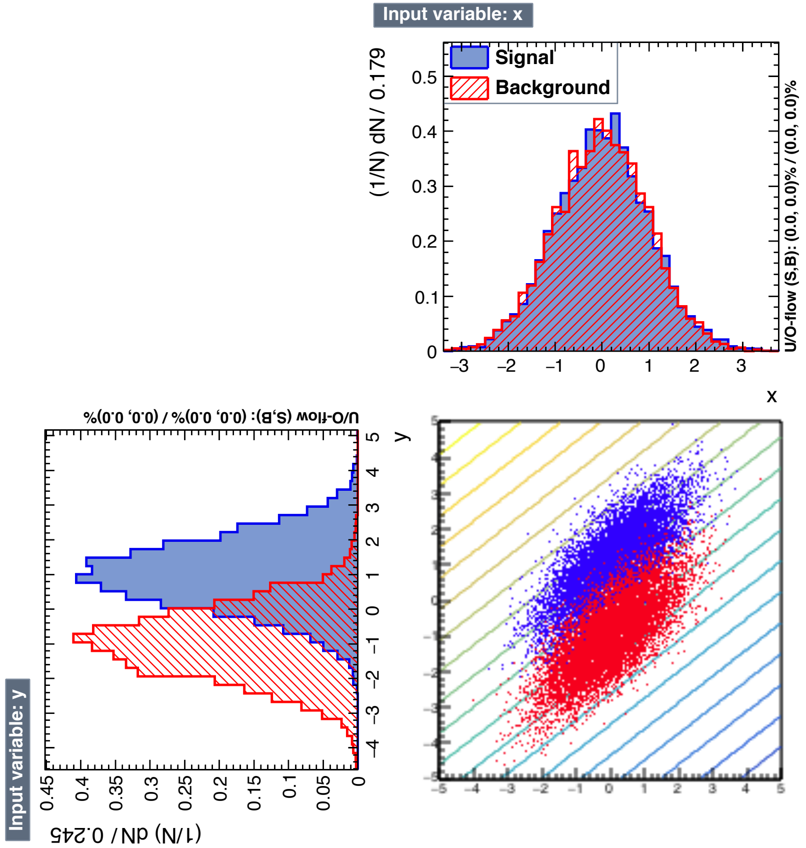


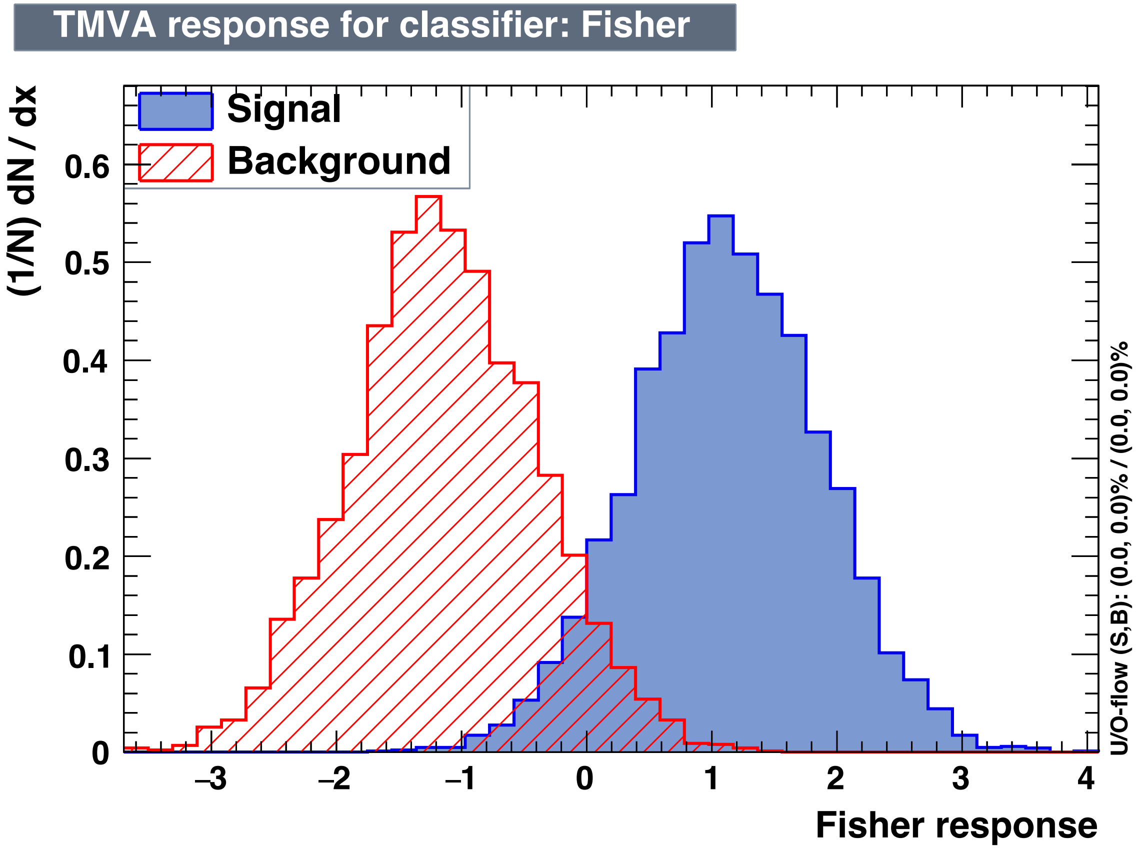
  - determine Fisher coefficients such that $\vec{\nabla} J(\vec{a}) = 0$
- For linear problems, Fisher is equivalent to likelihood ratio (optimal test statistic) => backup

Fisher Discriminant

Example

- 10000 Signal and background events: shifted Gaussian distributions, correlated between x and y





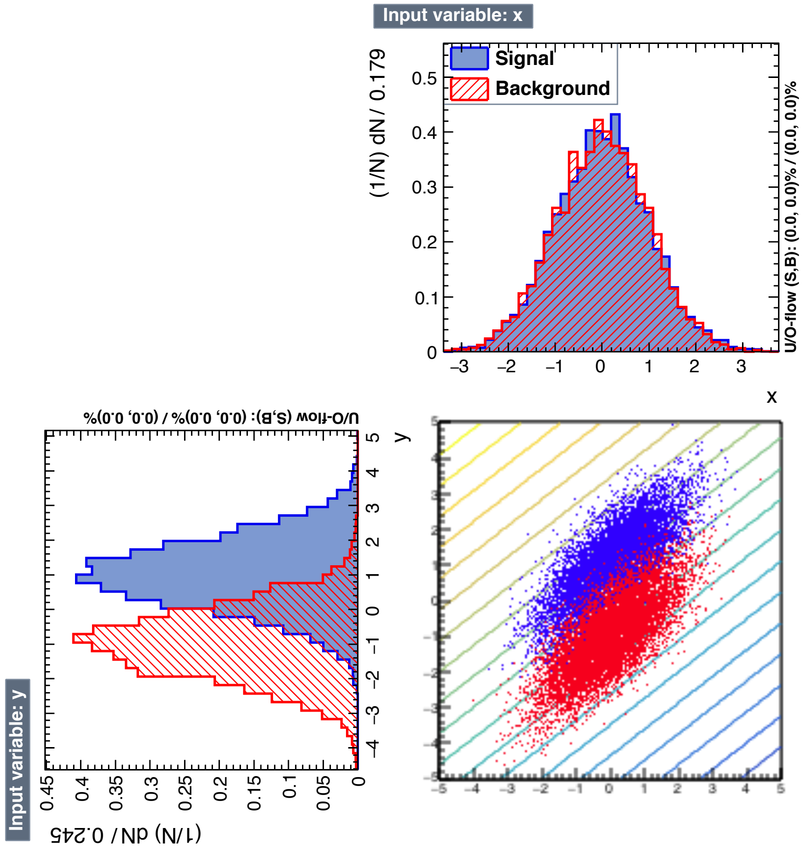
Fisher discriminant takes correlation into account

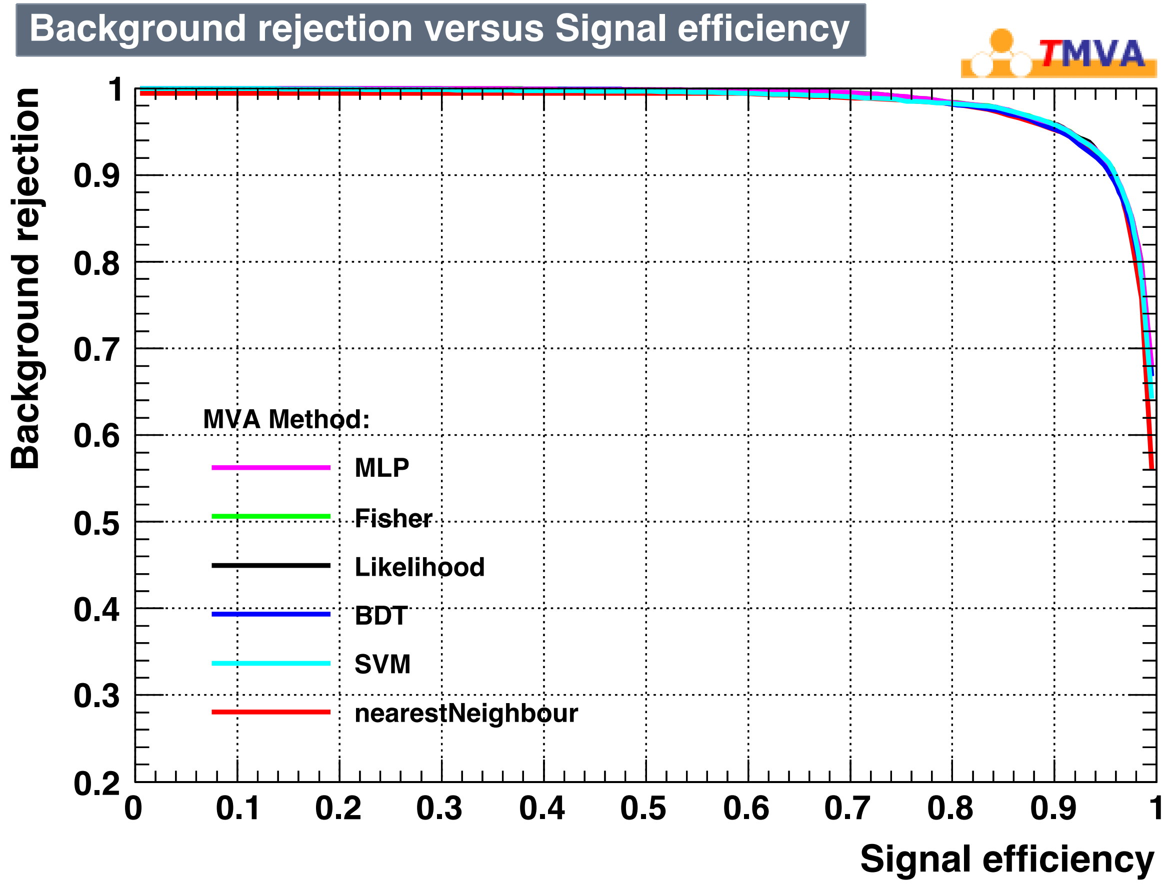
Generated using: „python TrainEvaluateResponse.py Gaussian"

Fisher Discriminant

Example

- 10000 Signal and background events: shifted Gaussian distributions, correlated between x and y



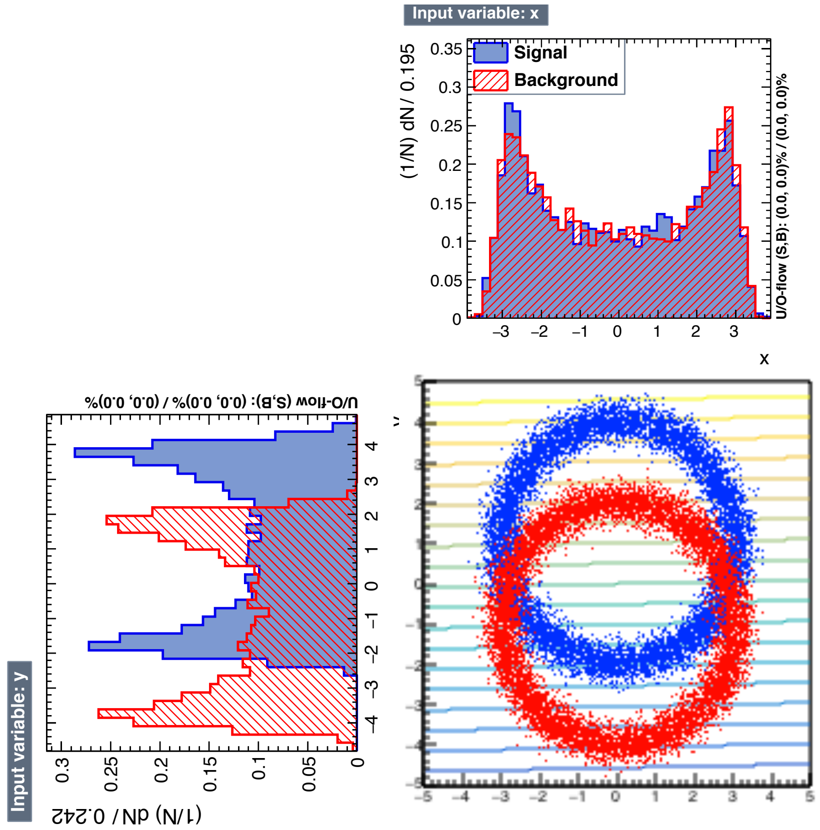


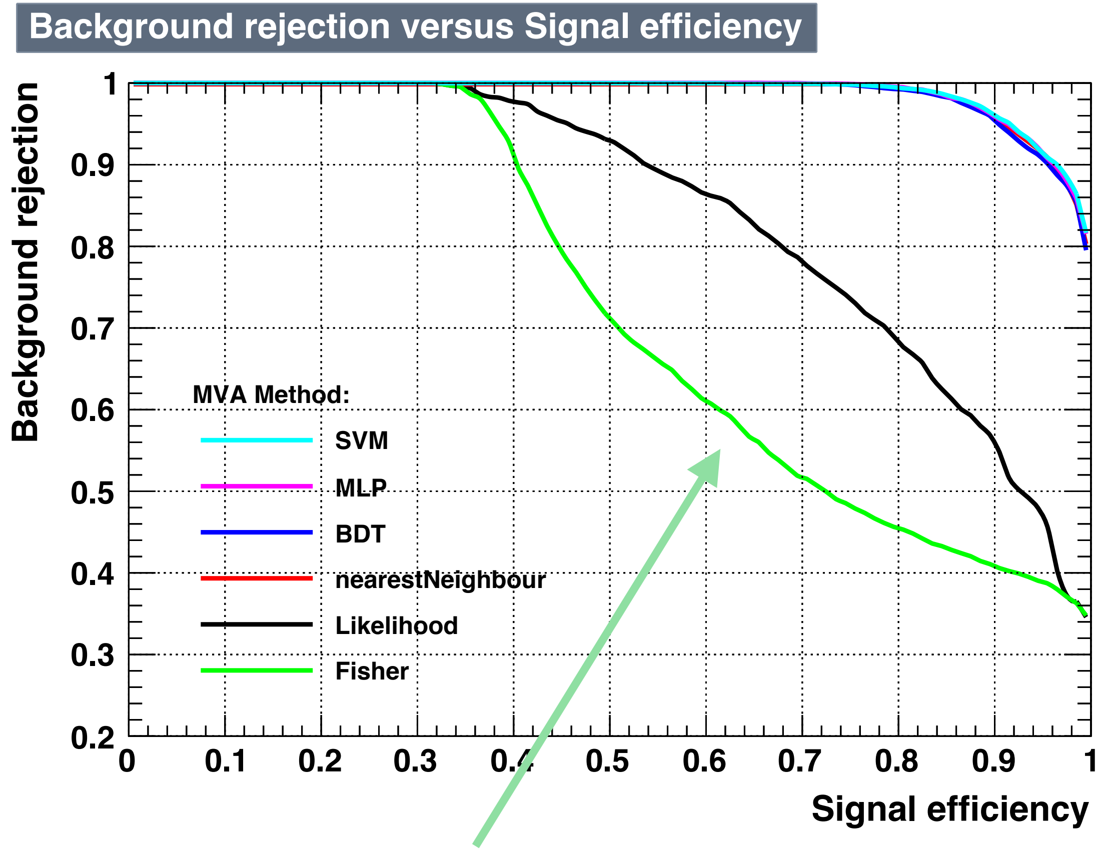
For this linear problem, Fisher discriminant provides optimal separation

Fisher Discriminant

Example

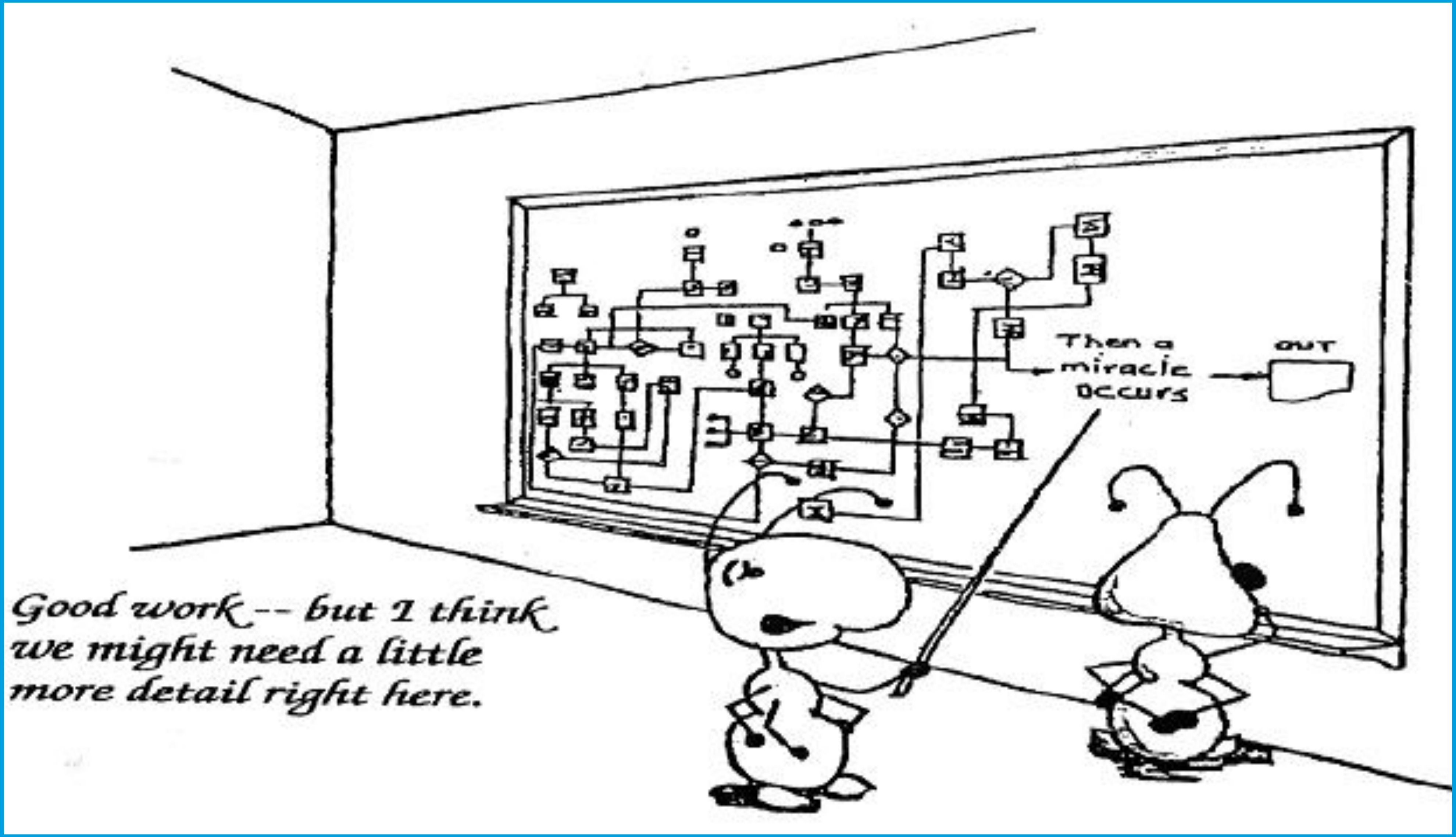
◉ 10000 Signal and background events: non-linear problem: shifted smeared circles





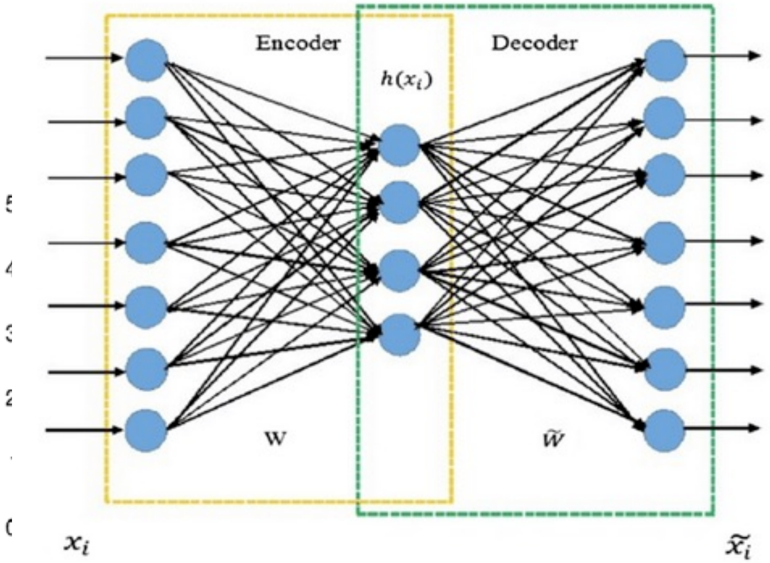
For this non-linear problem, Fisher discriminant is significantly worse than more complex methods

Machine Learning



Machine Learning

Autoencoder: target $\tilde{x}_i = x_i$



- Supervised Learning:
  - Pre-classified, “labelled” data (or MC) for signal and background.
  - During the training all inputs and output distributions are available.
  - Training: minimize loss function $E(||t_i - t_{true}||)$, i.e. difference between truth and result.
- Un-supervised Learning:
  - No labelled data or simulation
  - Recognition of (unknown) signal, patterns or anomalies
  - Examples: Principle Component Analysis, Autoencoders
- Reinforcement Learning
  - No labelled data or simulation
  - Optimize expected reward described by loss function

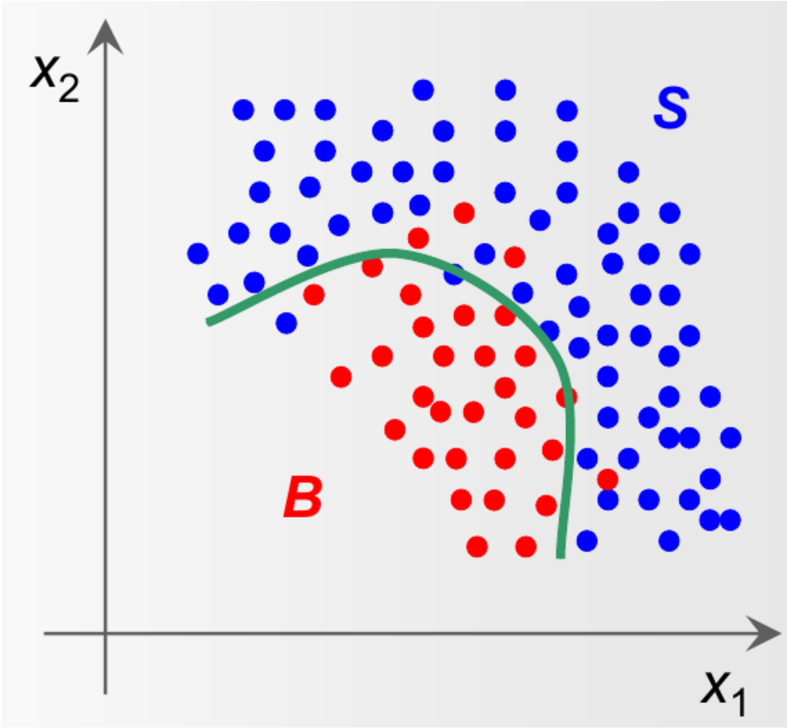
Deep Reinforcement Learning playing Atari: *youtube:V1eYniJ0Rnk*

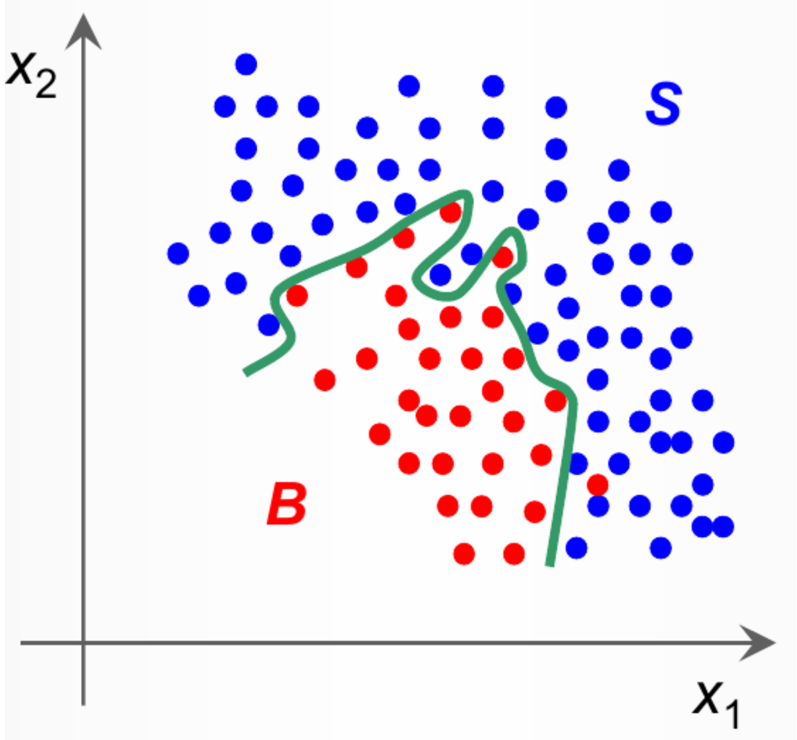
Paper explained: https://www.youtube.com/watch?v=rFwQDDbYTm4

Supervised Learning

Training

- Use labelled training data to determine optimal test statistic



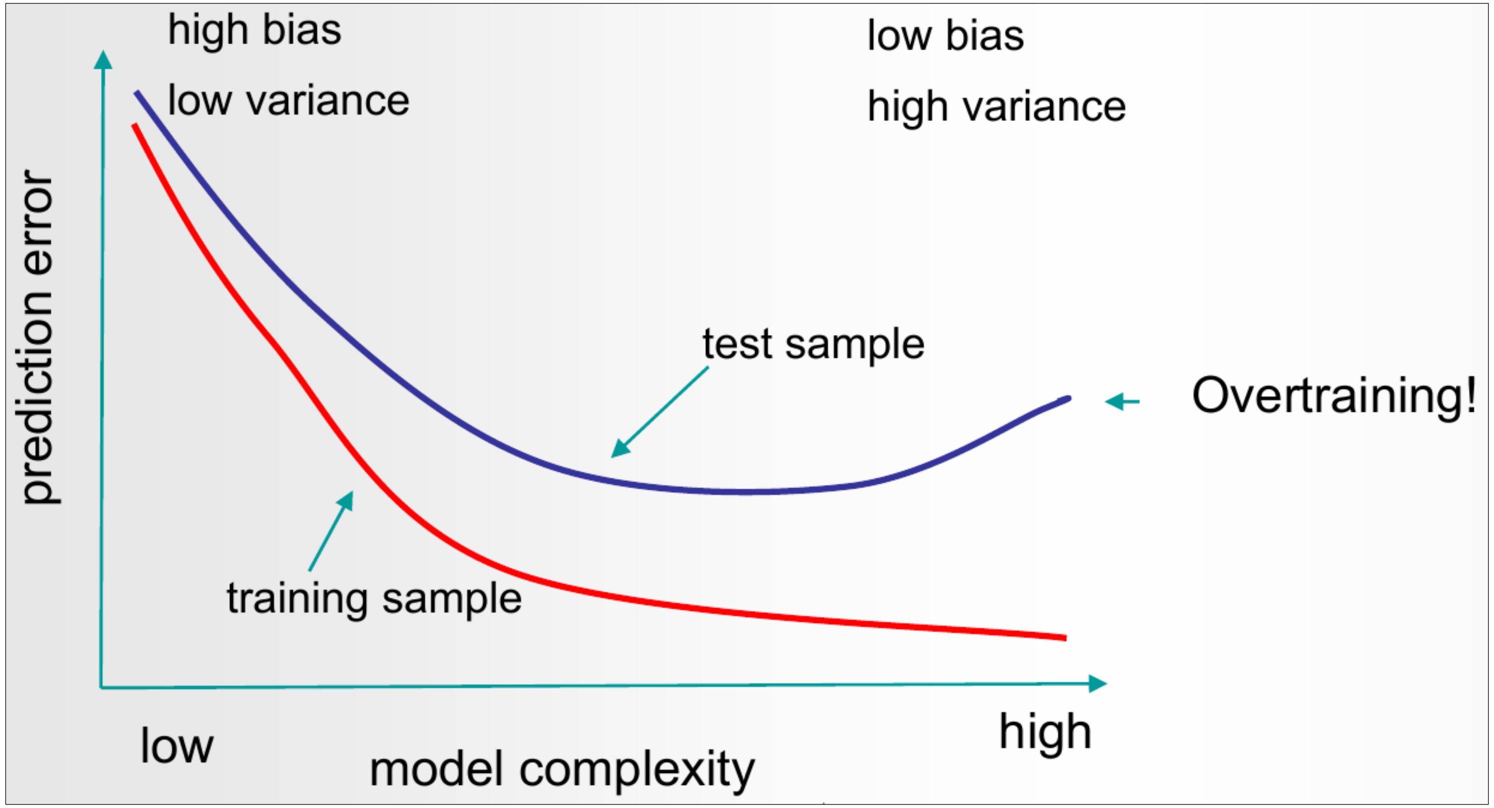


- Overtraining → generalisation loss: machine learns statistical fluctuations and not the concept
- Many input variables → curse of dimensionality → optimal choice of dimensions for a given problem, depending on available (labelled) data

Supervised Learning

Training and Testing

- Use statistically independent (labelled) dataset to test the trained algorithm

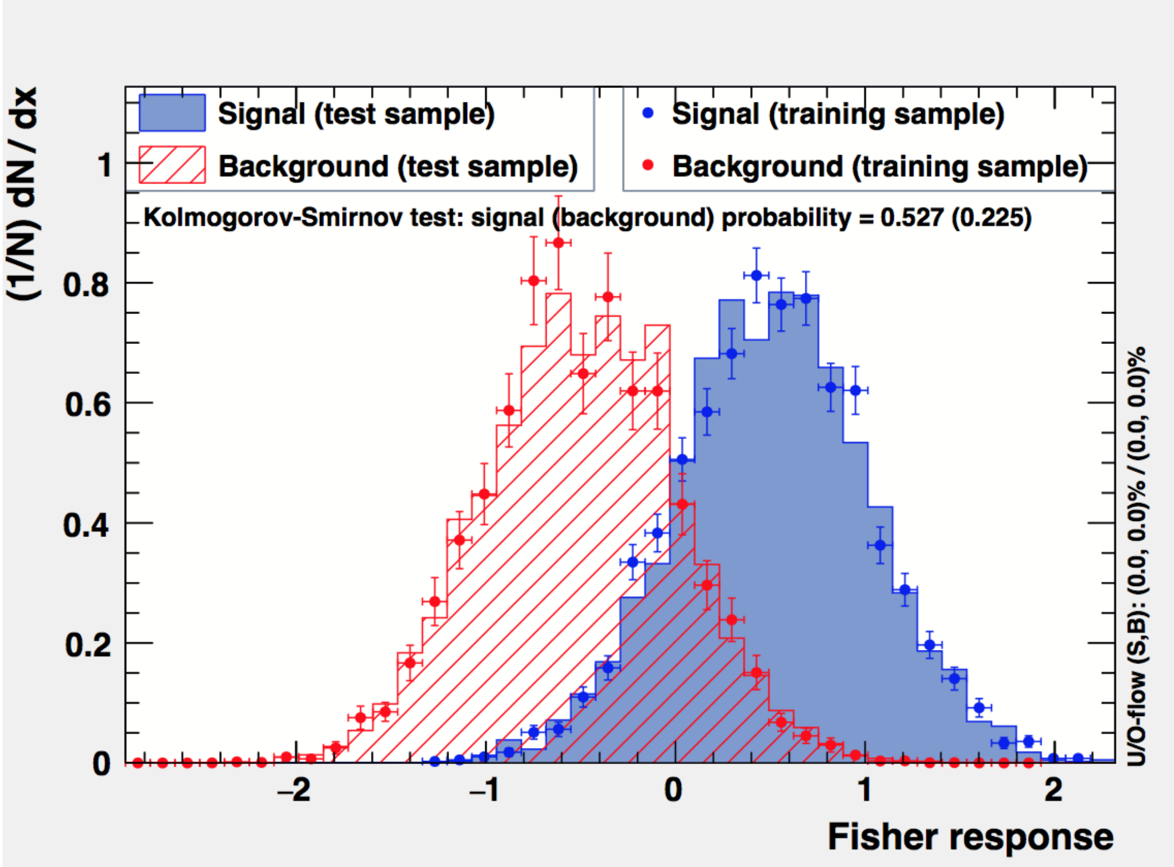


- Lower complexity (few parameters or training cycles) → worse separation (bias), lower variance
- Higher complexity → lower bias, but: overtraining → higher variance → bigger “generalization error”

Supervised Learning

Testing

- Use statistically independent (labelled) dataset to test the trained algorithm



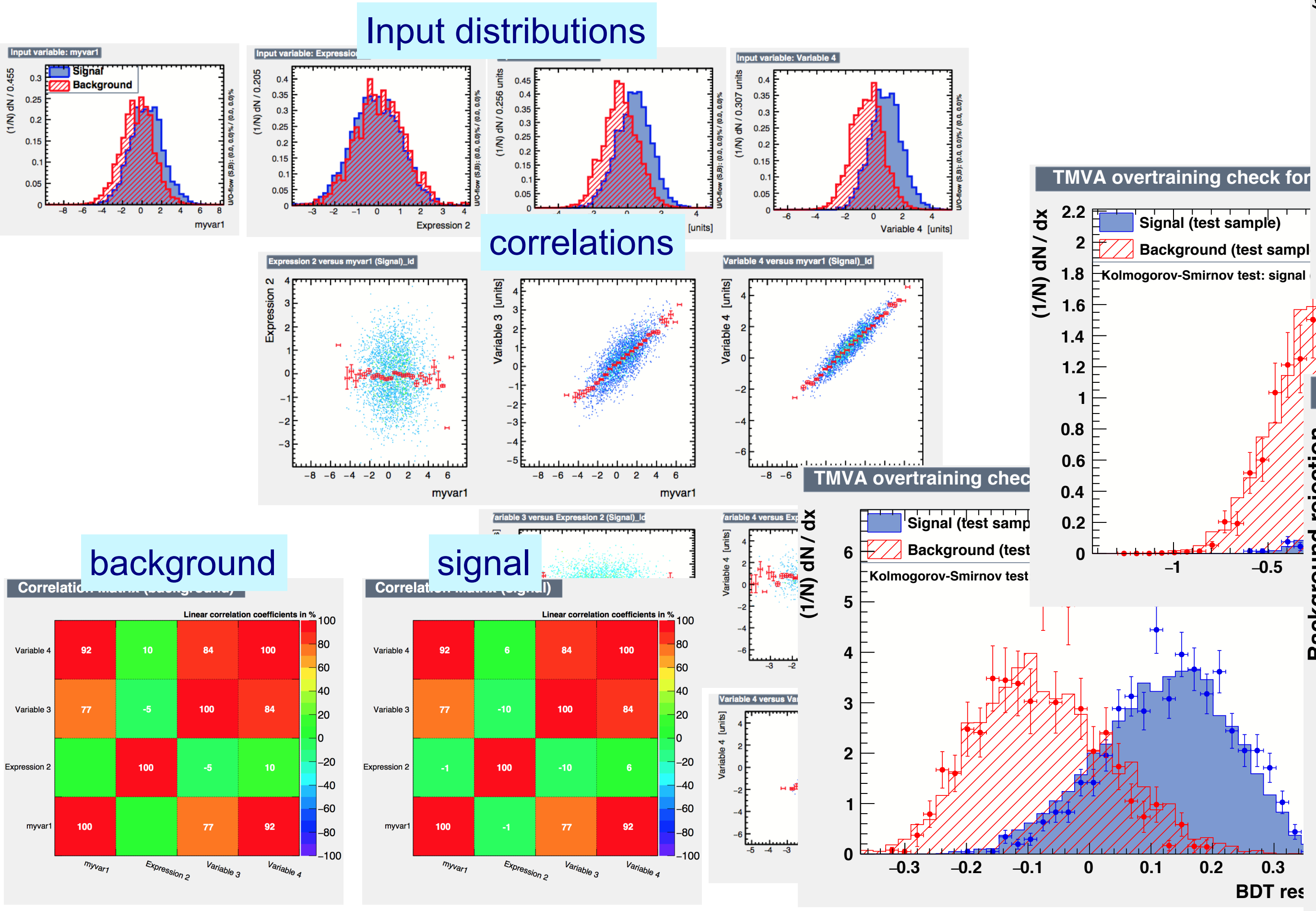
- Kolmogorov-Smirnov (goodness-of-fit) test: maximum difference of the cumulative PDF

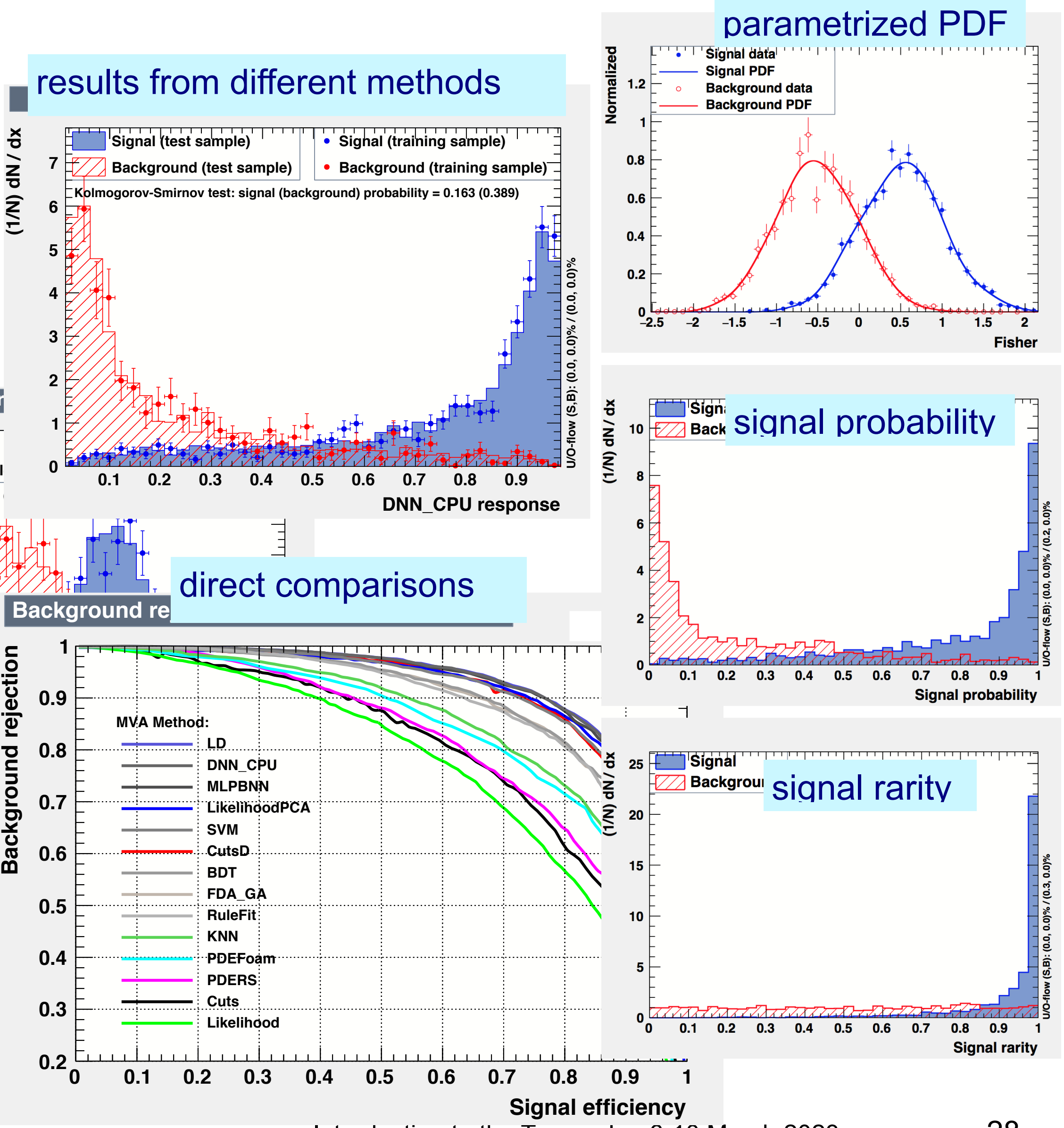
TMVA

https://root.cern.ch/doc/master/group__tutorial__tmva.html

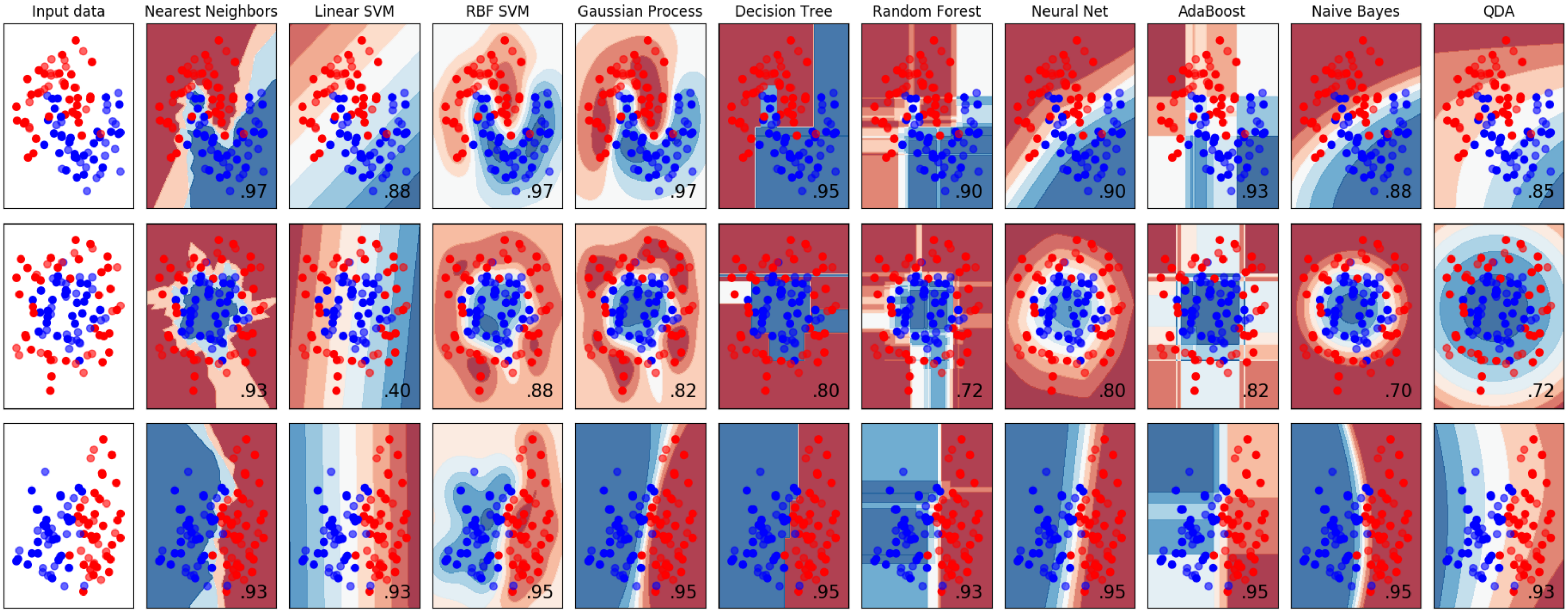
Multivariate Analysis Toolkit of Root

- Breakthrough in use of ML in particle physics (since 2005)
- Rich set of standardized diagnostic histograms
- Direct comparisons and “hyper-parameter optimisation”





Large Variety of MVA Algorithms



Taken from: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Summary: Supervised Learning

https://xkcd.com/1838/

◉ Preparation:

- Choose simplest method that provides (close to) optimal solution (linear / non-linear)
- Keep dimensionality minimal
- Identify *n* optimal features
- Remove strongly correlated inputs

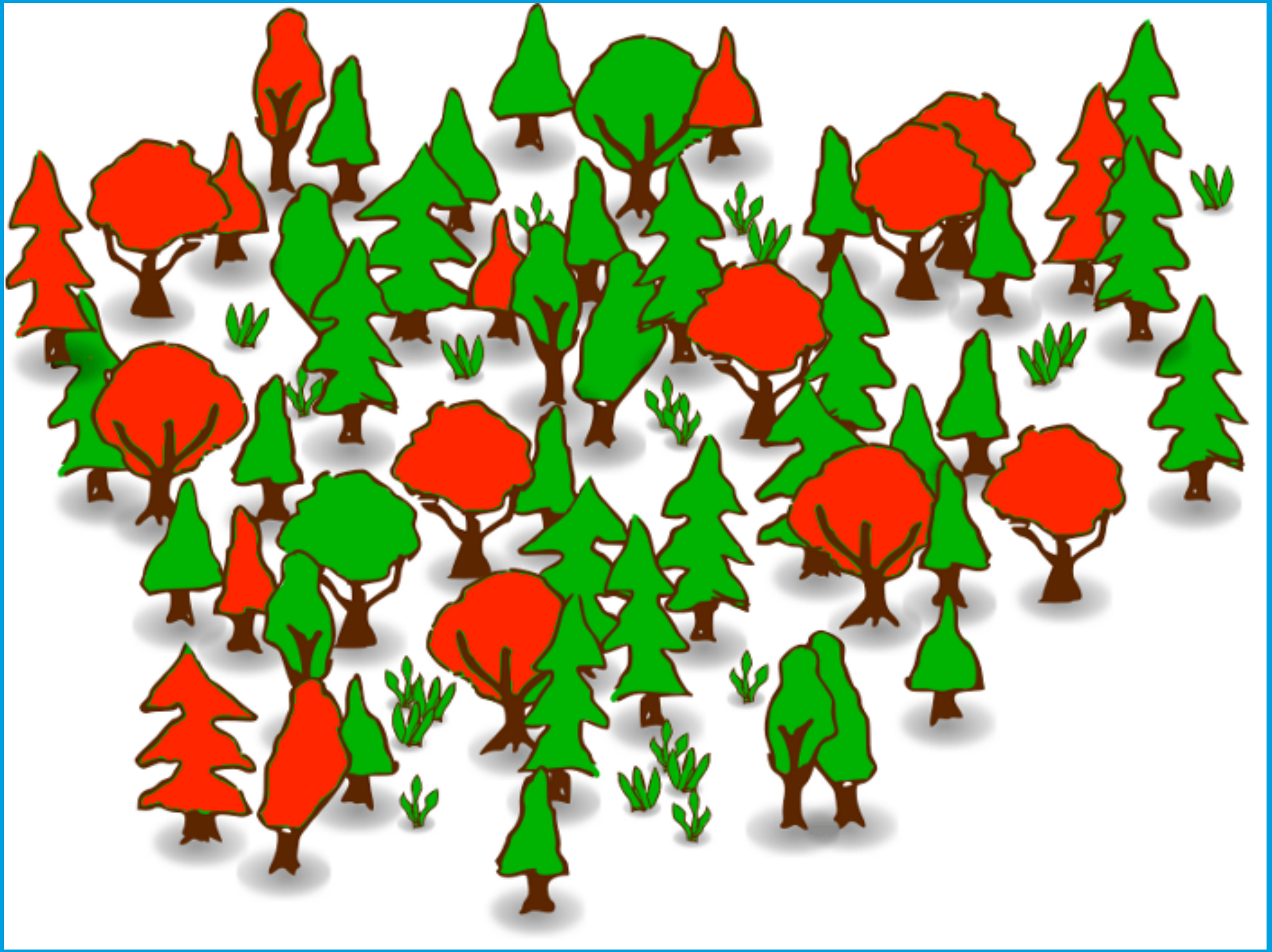
◉ Training and testing:

- Test generalisation properties: "trade-off between bias vs variance", in English: avoid overtraining
- Scan hyperparameters to ensure result is stable and close to optimal

◉ Application:

- Calculate event-by-event discriminator, i.e. scalar test statistic t(x)

Boosted Decision Trees

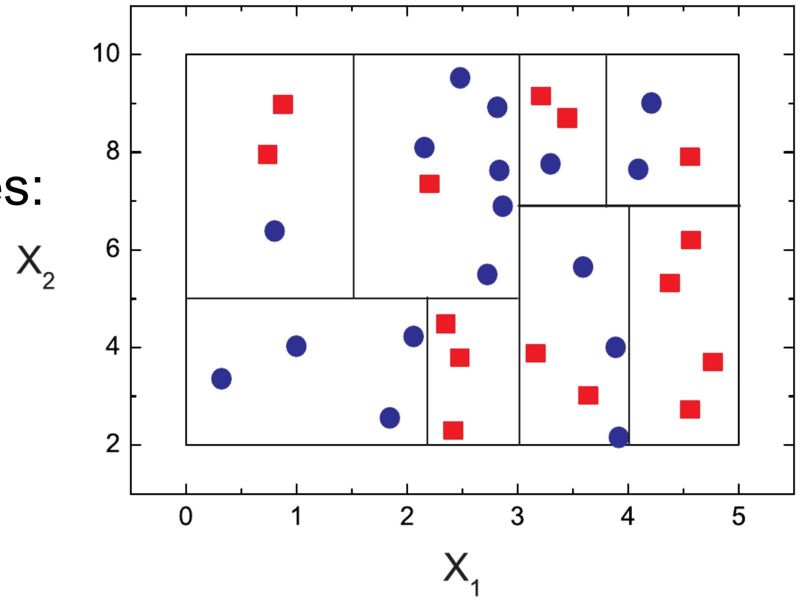


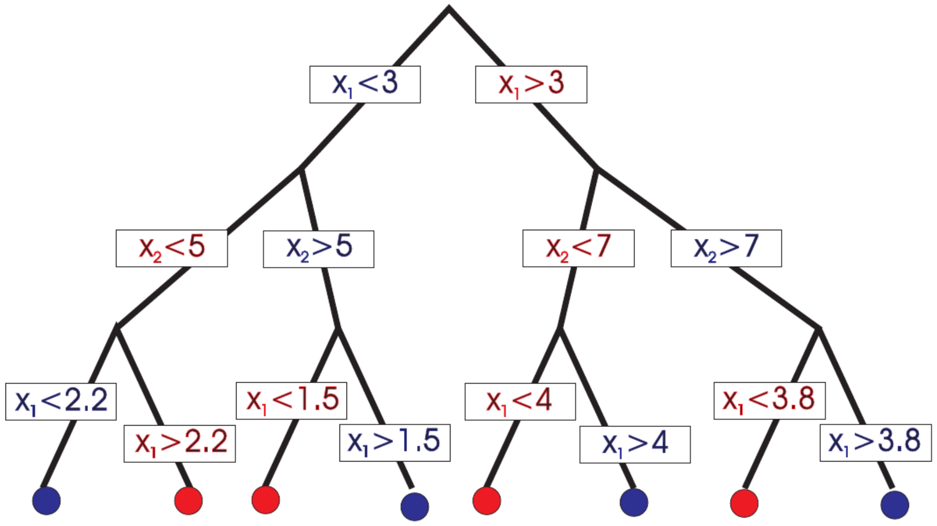
Decision Tree

- Sequential („<" or „>") decisions in single observables:
  - cutting the feature space into (hyper-)squares

- Decision tree:
  - decisions = branches,
  - end nodes = leaves

- Features:
  - Robust against outliers and normalisation
  - Features can be used several times ("greedy algorithm")
  - Training is usually fast (in comparison to ANN)





Taken from: Bohm/Zech

Decision Tree

Training: Growing the Tree

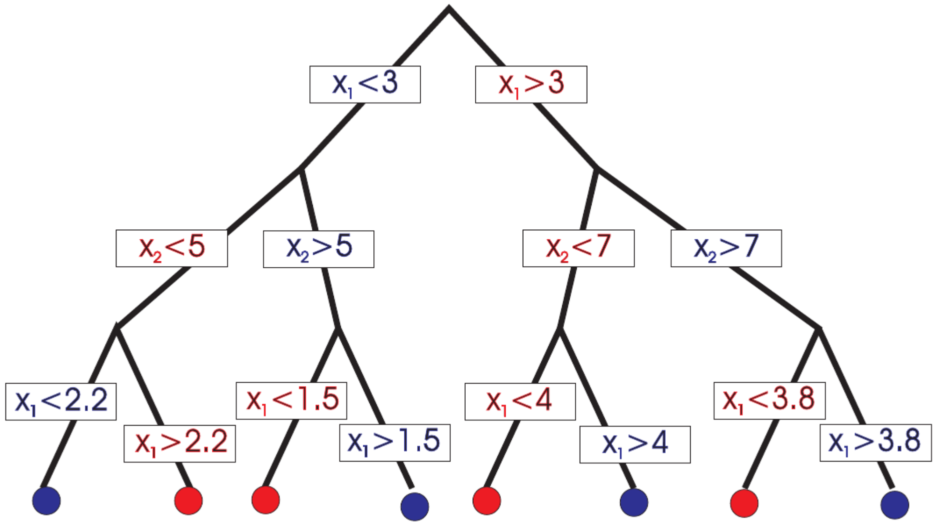
- Take most significant feature of the training dataset to separate events into two branches
  - Fake rate: $F = 1 - \max(p, 1 - p)$
  - Gini index: $G = 2p(1 - p)$
  - "Cross entropy S" = $-(p \ln(p) + (1 - p)\ln(1 - p))$

Decision criteria

$$p = \frac{n_s}{n_s + n_b}$$

- "Greedy Algorithm": sequentially repeat until stopping criterion
  - maximal number of leaves
  - minimal number of events
  - target purity

Stopping criteria

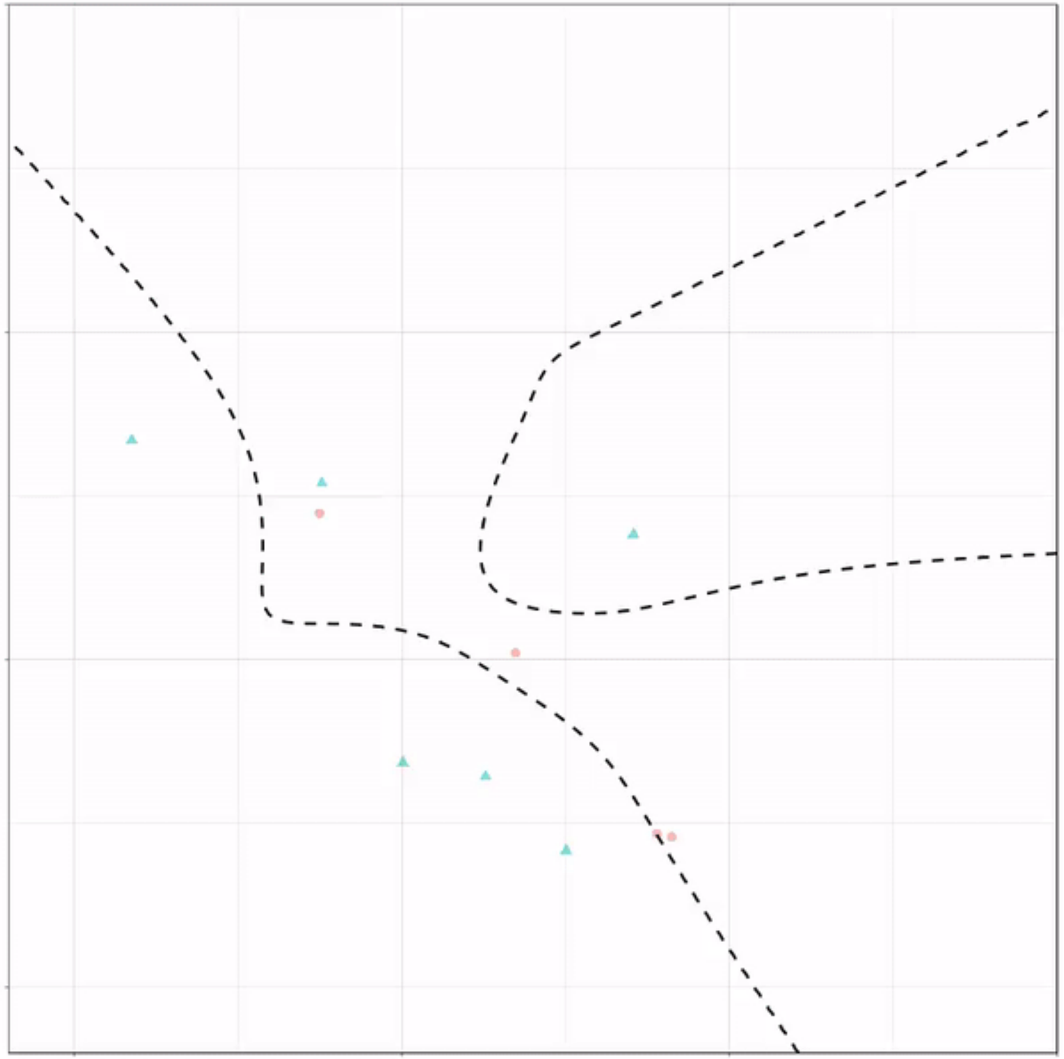


Taken from: Bohm/Zech

Decision Trees

Training: Growing the Tree

https://paulvanderlaken.com/2020/01/20/animated-machine-learning-classifiers/



Forest of Decision Trees

Ensemble of Weak Learners

- “Weak Learner”: a single tree generally provides poor generalisation to other data (same PDF)

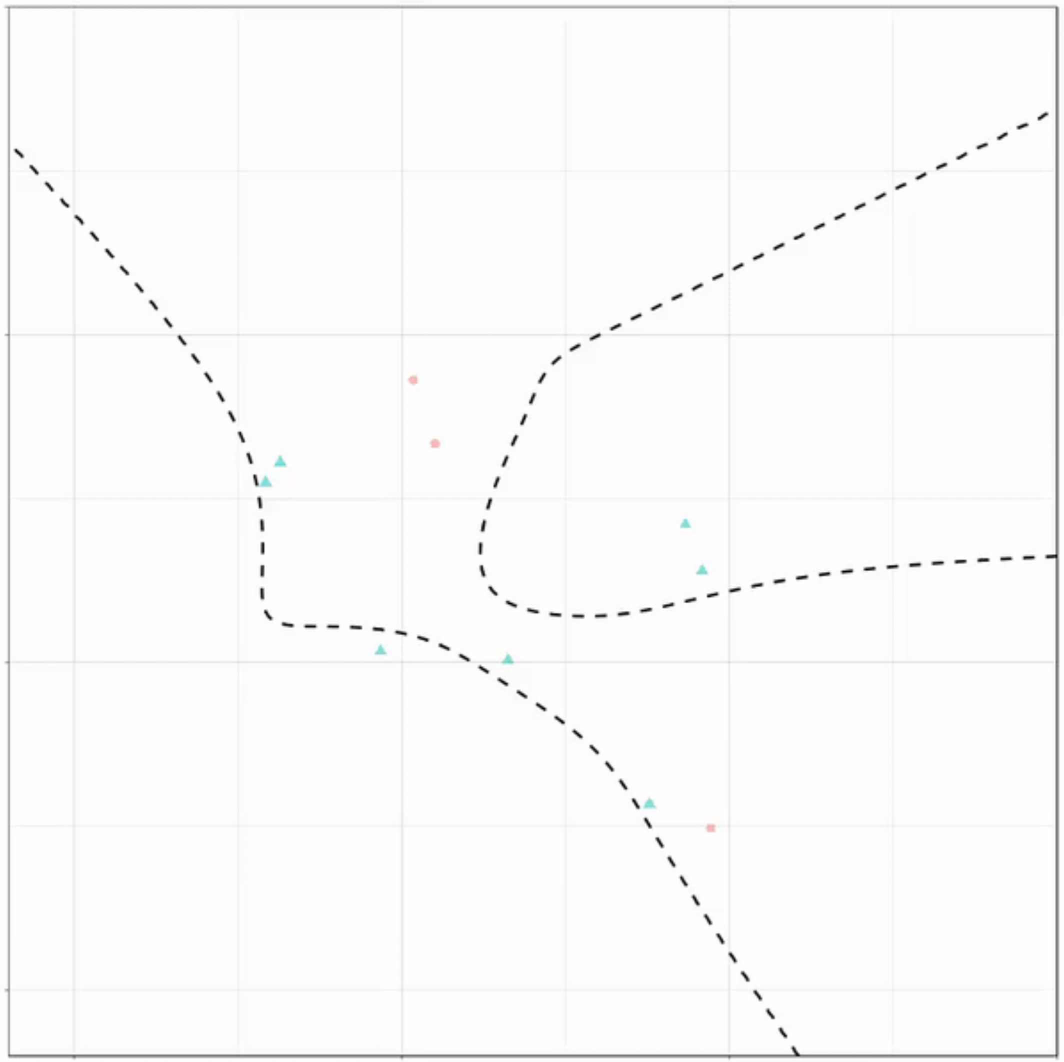
- A forest (≥1000 trees) can be very powerful: robust separation by majority vote of many trees, each of which has a poor separation ("ensemble method”)

- Further improvements:
  - Random Forest: each tree is built from random subsets of observables
  - Bagging: subset of the test dataset are used to generate the decision tree.
  - Boosting: increase weights of wrongly classified events

Forest of Decision Trees

https://paulvanderlaken.com/2020/01/20/animated-machine-learning-classifiers/

Ensemble of Weak Learners



AdaBoost

Freund, Schapire 1997

"Adaptive Boosting"

- Observables with relevant information are considered more
- Assign increased weights to wrongly classified events for subsequent weak learners
- Calculate weight α_i from fake rate err_{i-1} of the previous tree

$$\alpha = \frac{1 - \mathrm{err}}{\mathrm{err}}$$

- Boosted classification:

$$y_{\mathrm{Boost}}(\mathbf{x}) = \frac{1}{N_{\mathrm{collection}}} \cdot \sum_{i}^{N_{\mathrm{collection}}} \ln(\alpha_i) \cdot h_i(\mathbf{x})$$

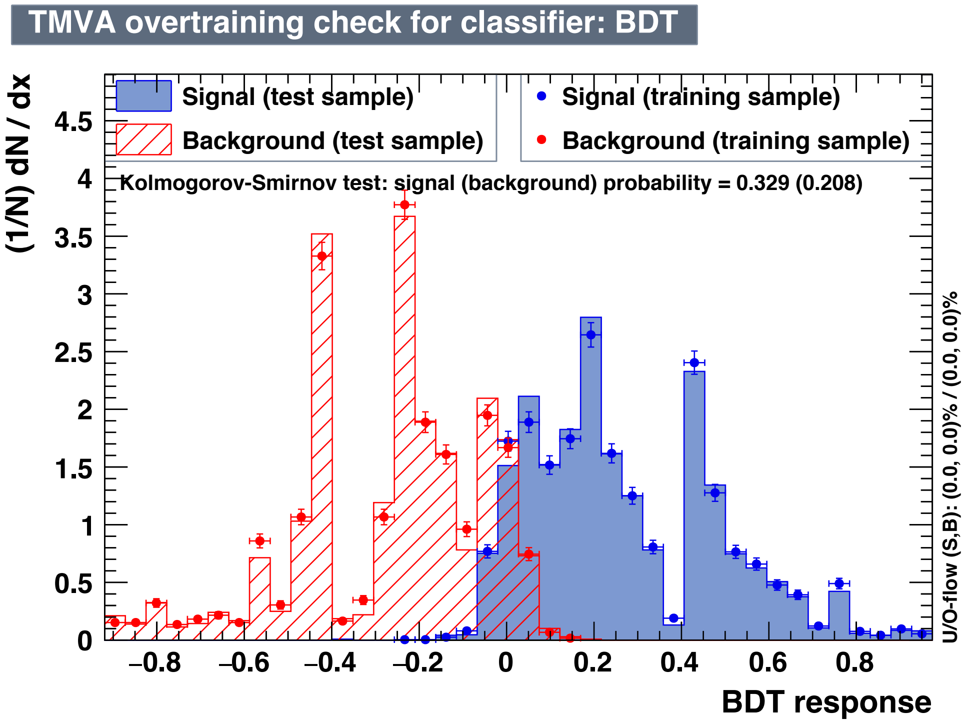
Where $h_i(x)$: 1(-1) for signal(background) and $N_{\mathrm{collection}}$: number of trees

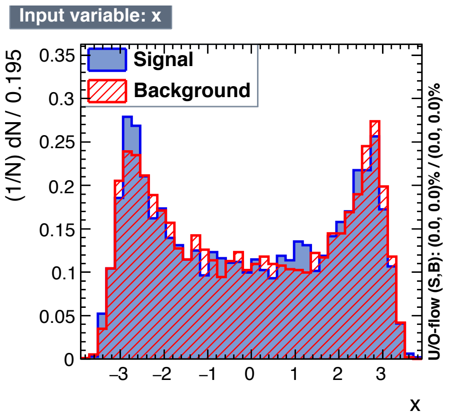
- Adjust boost strength through additional hyperparameter β, i.e $\alpha \rightarrow \alpha^{\beta}$

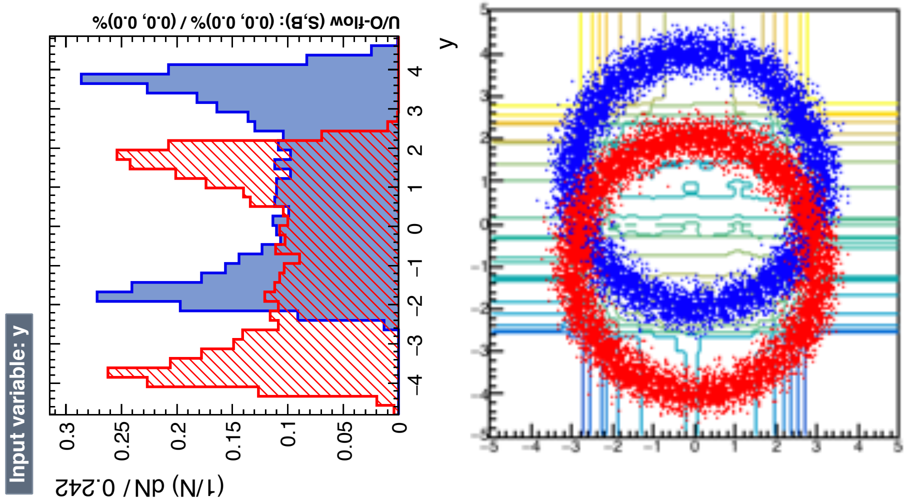
AdaBoost

Example

- AdaBoost, NTrees=500, MinNodeSize=0.5, AdaBoostBeta=0.5, MaxDepth=3, nCuts = 20, SeparationType=GiniIndex, 10000 events
- Training: 2s, testing: 0.5s





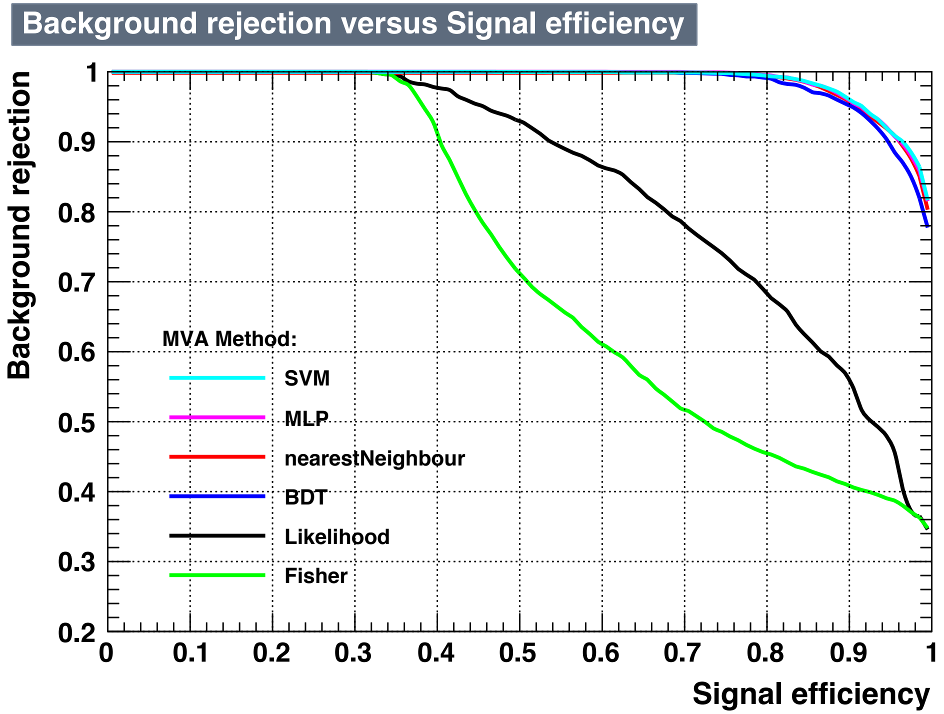


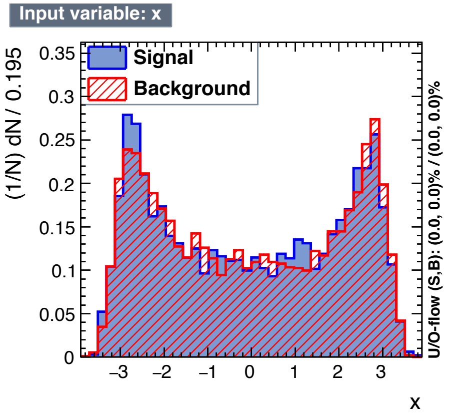
Generated using: „python TrainEvaluateResponse.py Circle"

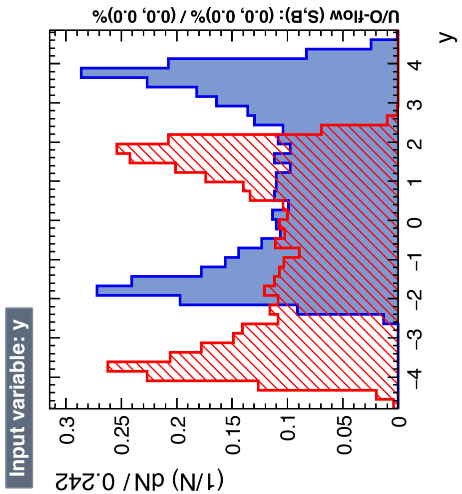
AdaBoost

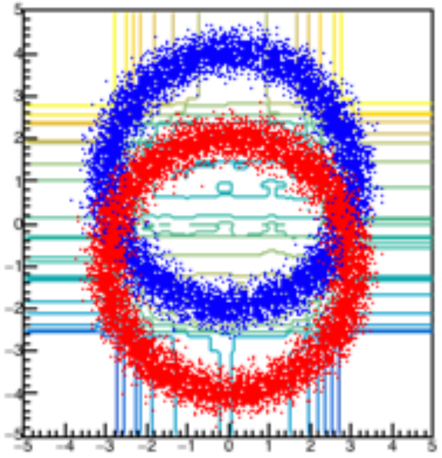
Example

- AdaBoost, NTrees=500, MinNodeSize=0.5, AdaBoostBeta=0.5, MaxDepth=3, nCuts = 20, SeparationType=GiniIndex, 10000 events
- Training: 2s, testing: 0.5s
- Good separation









Generated using: „python TrainEvaluateResponse.py Circle“

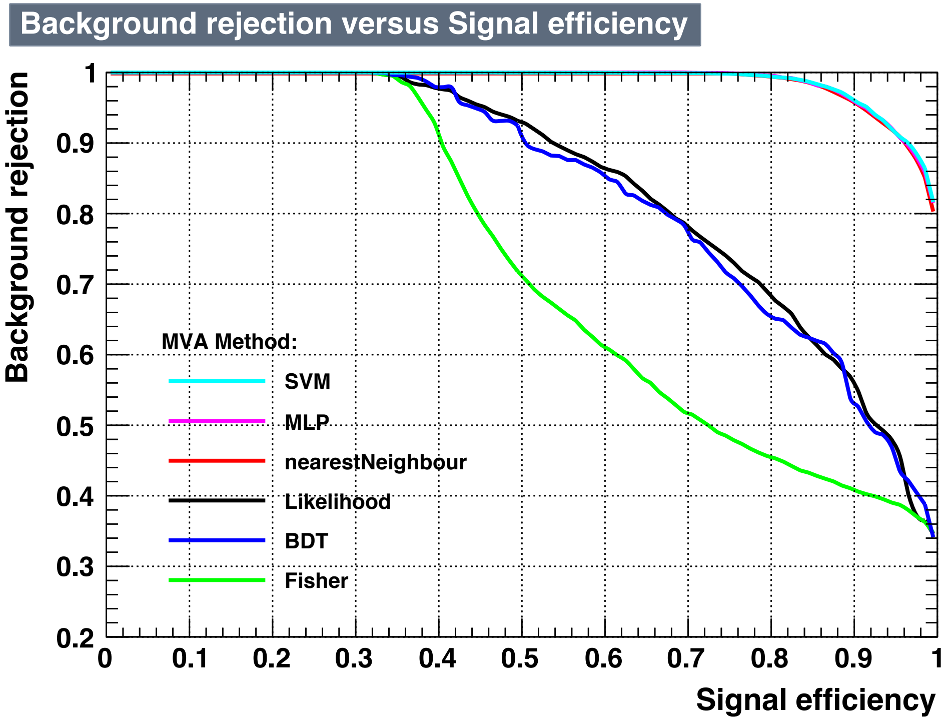
AdaBoost

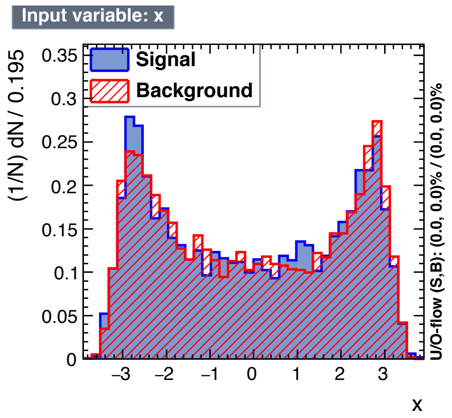
Example

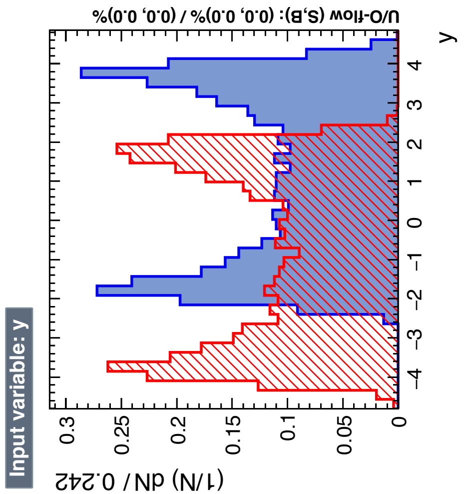
This is the TMVA default: 5%

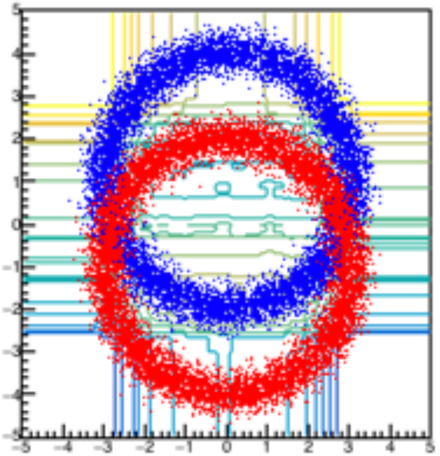
- AdaBoost, NTrees=500, MinNodeSize=0.05, AdaBoostBeta=0.5, MaxDepth=3, nCuts = 20, SeparationType=GiniIndex, 10000 events
- Training: 2s, testing: 0.5s
- Bad separation

Hyperparameter optimization









Generated using: „python TrainEvaluateResponse.py Circle"

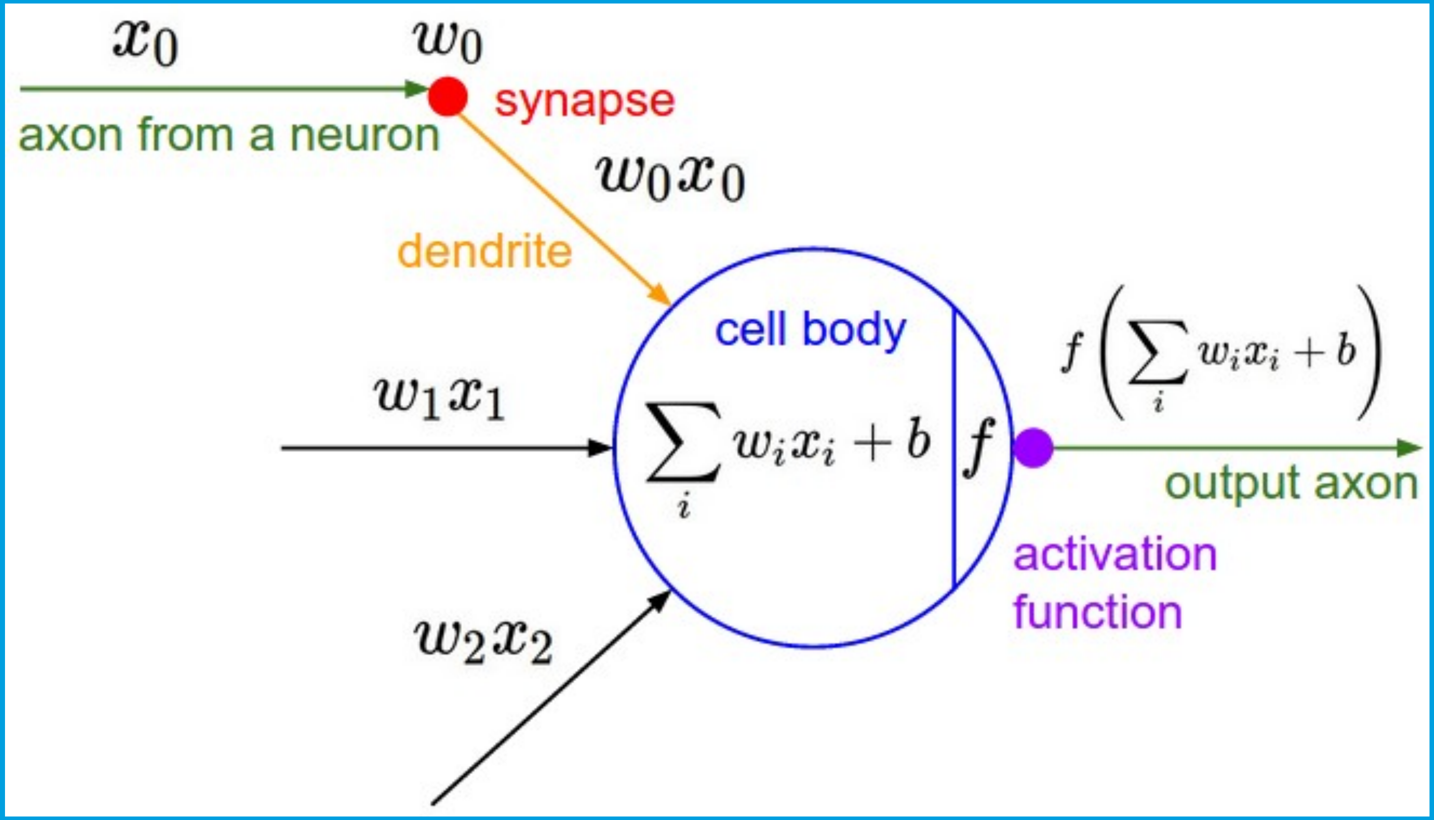
Boosted Decision Trees

Summary

- Ensemble method: many simple models (weak learners) together can make up a complex model
- Good properties:
  - Locally 1-dimensional decisions
  - Fast suppression of obvious backgrounds
  - Robust against outliers
  - No special metric or normalization of input variables
  - Few parameters (tuning effort, aka hyper-parameter optimisation, is small)
  - Trees can be understood, including straightforward ranking of inputs
  - Fast training
- Relatively slow in execution

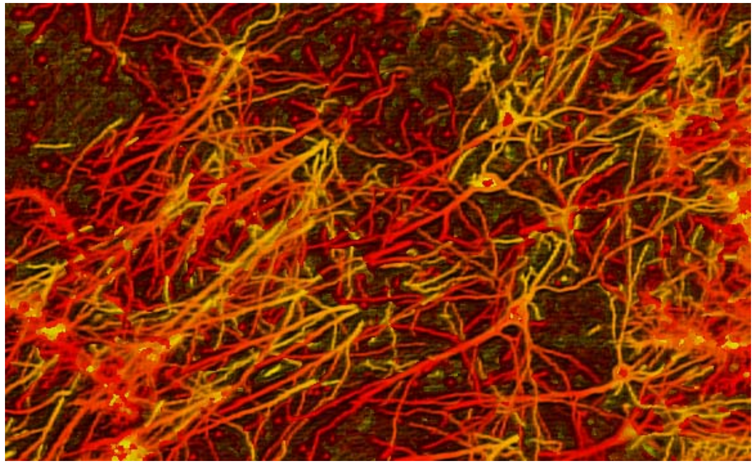
Boosted Decision Trees are very popular in particle physics - still !

Artificial Neural Networks

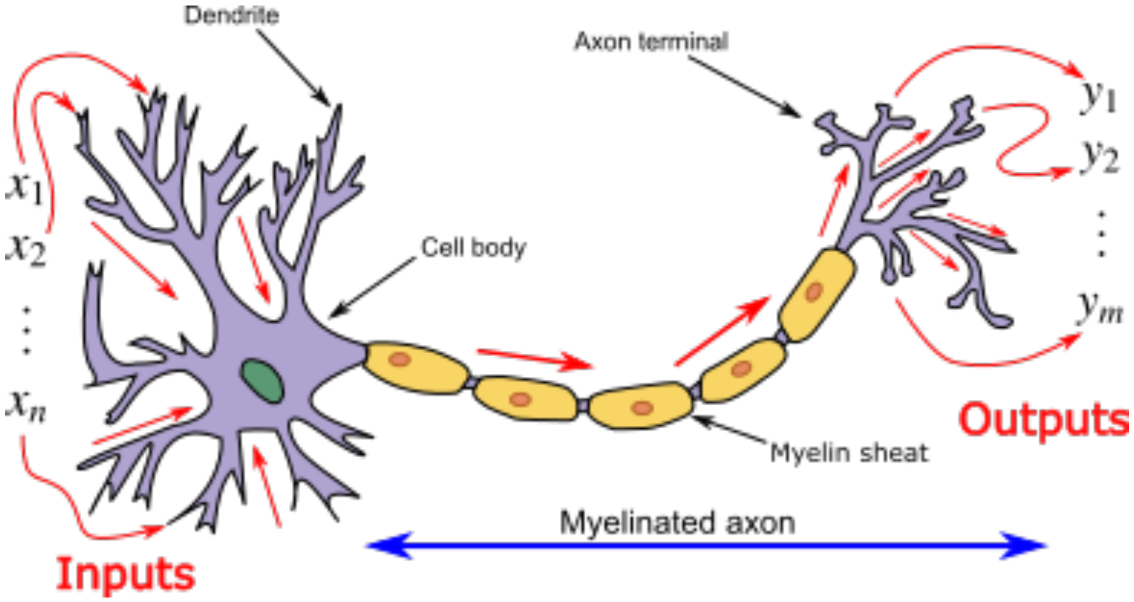


Biological Neural Networks

Source: www.willamette.edu/~gorr/classes/cs449/brain.html



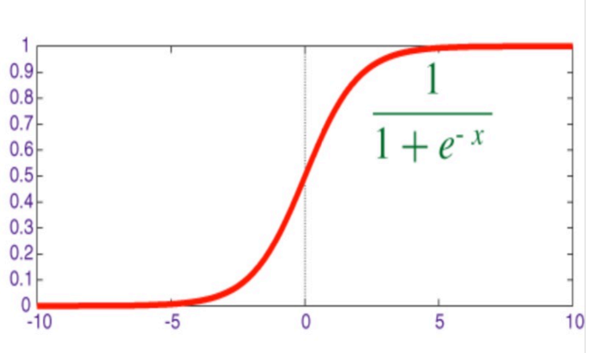
- Human brain
  - Many processors = $O(10^{11})$ Neurons
  - Single processing step slow: O(10 ms) ~ 100 Hz
  - Massively parallel: $O(10^{14})$ Synapses
- Neurons:
  - Generate output signal if combined input signals exceed some threshold
- Natural Neural Networks
  - Tolerant against incomplete or noisy inputs
  - Self-organised learning: poorly understood



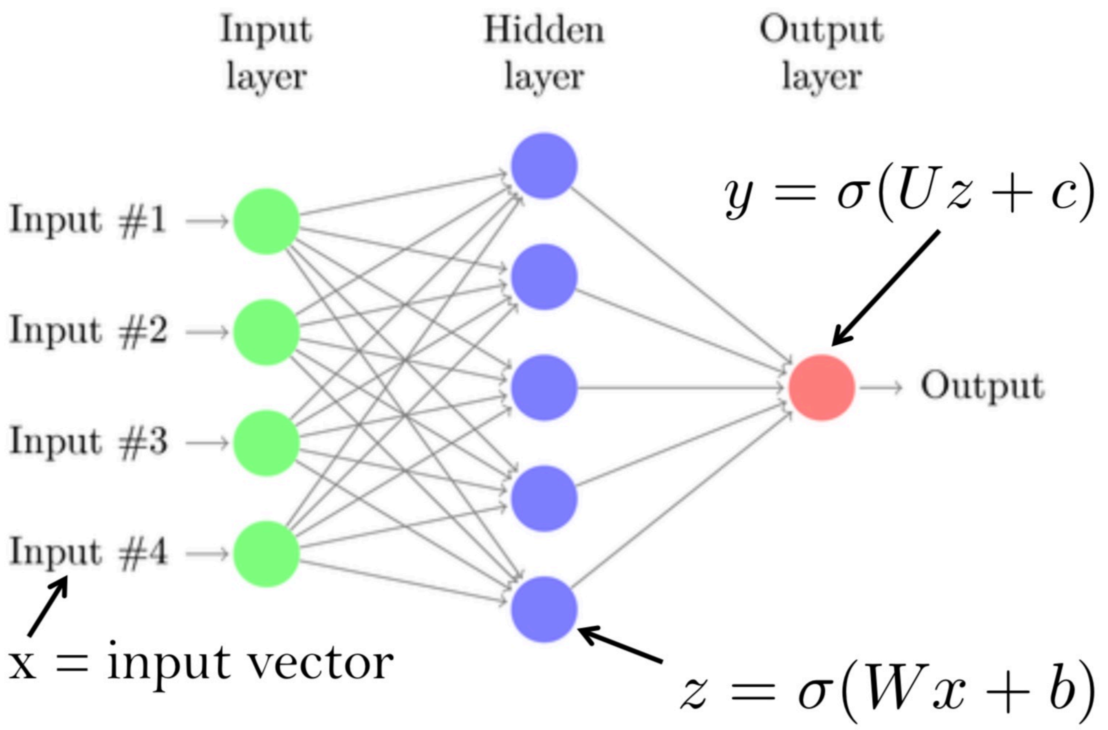
Wikipedia

Artificial Neural Networks (ANN)

- Standard non-linear method in supervised learning
  - Feed-forward network
    - Typically O(10^3) neurons (in DL up to 10^9)
    - Simple topology (in DL not so simple)
    - Fast (O(ns)) ~ GHz
    - Training usually slow (slower than BDT)
  - Weights W and U by minimisation of loss-function
  - Differentiable activation function σ(x):



Conventional "feed-forward" ANN



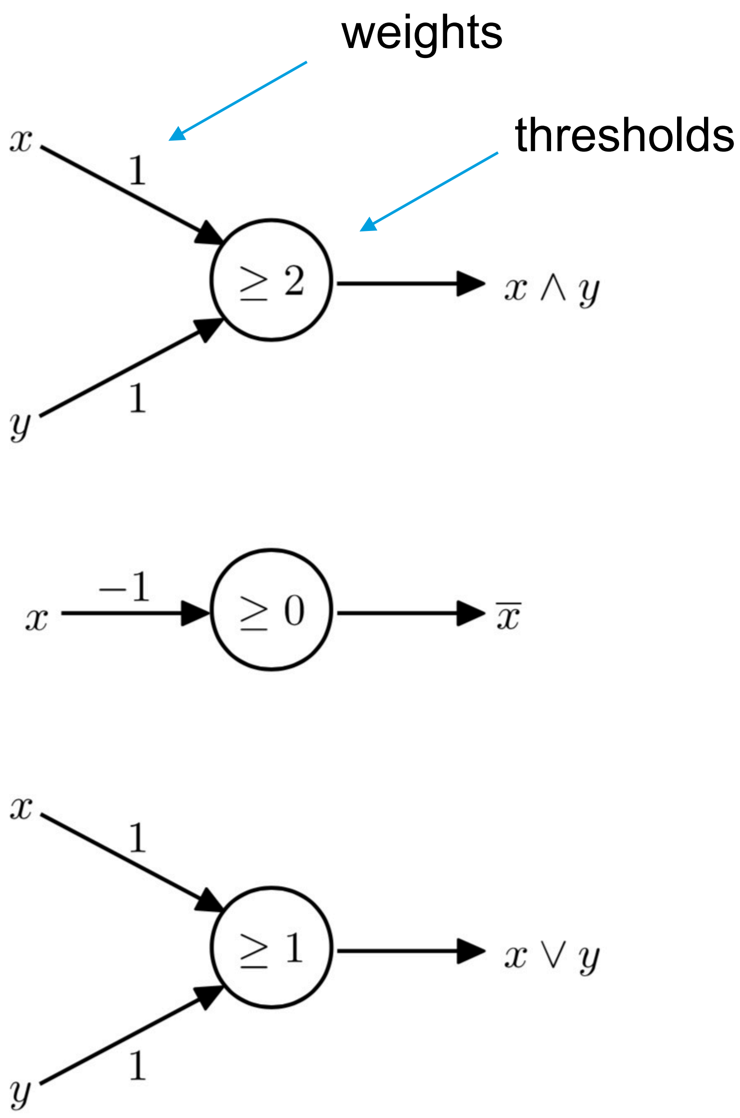
One-Layer Perceptron

Rosenblatt, 1958

x and y

not x

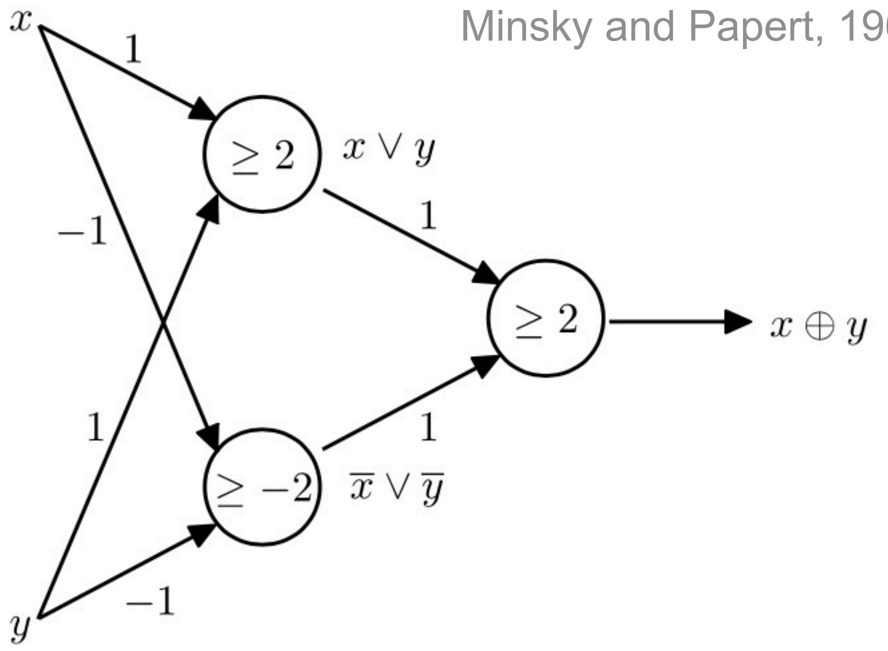
x or y



- One-layer perceptron implements basic logical gates AND, NOT and OR
- But not XOR!
  solution: use an additional hidden layer

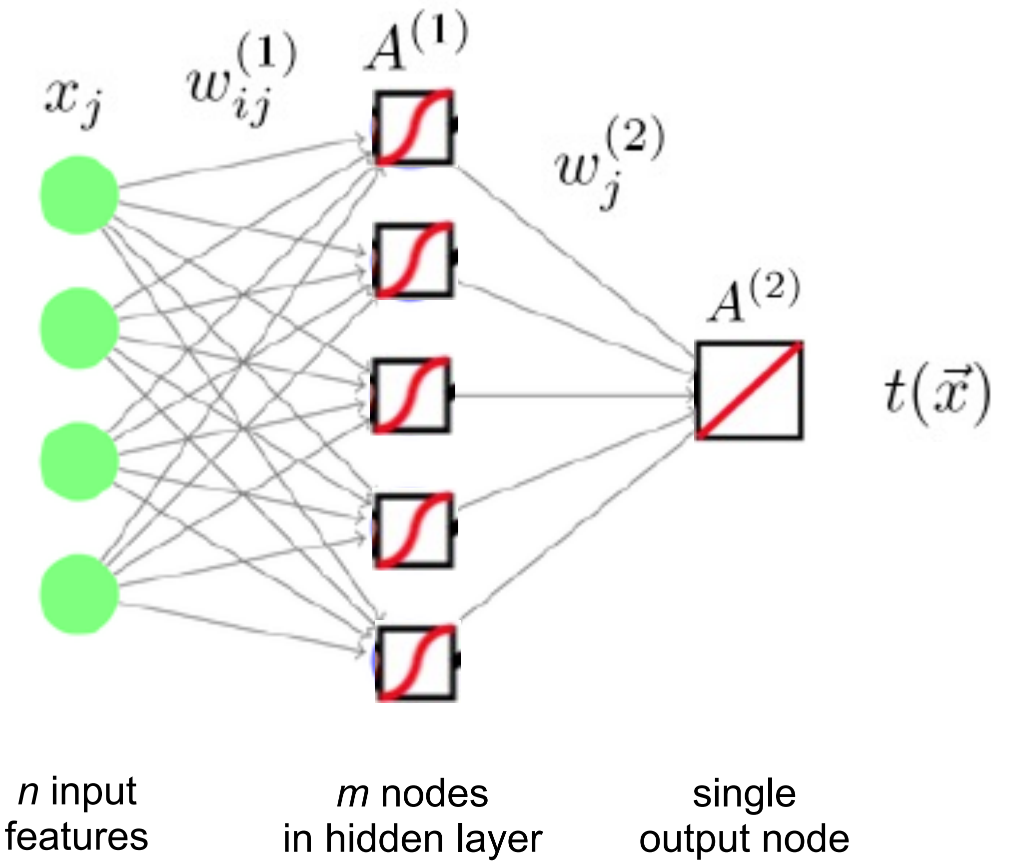
Minsky and Papert, 1969

x XOR y



Multi-Layer Perceptron (MLP)

- One or several hidden layers
- Feed-forward network
  - Each layer is fed only by previous layer
  - Most important case: one single hidden layer with *m* nodes (oft: *m*>*n*)
- Non-linear test statistic:



$$t(\vec{x}) = A^{(2)}\left(\sum_{j}^{m} w_j^{(2)} \cdot A^{(1)}\left(\sum_{i=0}^{n} w_{ij}^{(1)}\, x_j\right)\right)$$

With appropriate ω, a multi-layer perceptron can approximate any continuous function

Loss-Function Minimization

- Loss function $\mathbf{Er}(t_{\text{true}}, t(\vec{x}))$ describes degree of agreement between classifier and expectation
- Error backpropagation: iterative procedure to determine optimal weights W
  - Often used: Mean Average Distance (MAD) or Mean Squared Error (MSE):

MSE: $$\mathbf{Er}(\vec{x}|W) = \sum_{a=1}^{N} \mathbf{Er}(\vec{x}_a|W) = \frac{1}{2}\sum_{a=1}^{N}(t_{\text{true}} - t(\vec{x}_a|W))^2$$

- Gradient descent method:
  - In each iteration (learning cycle) the weights W are modified in the direction of the loss function gradient

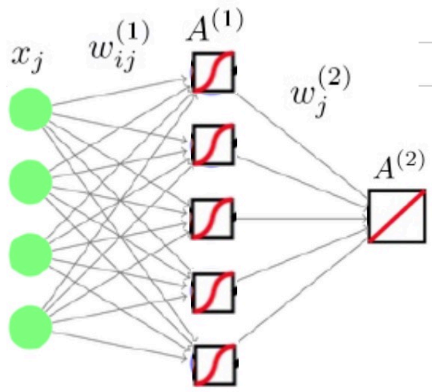
$$W^{(n+1)} = W^{(n)} - \eta \nabla_W \mathbf{Er}(t_{\text{true}}, t(\vec{x})|W)$$

  - η: learning rate (step size)
    - Too small: slow convergence
    - Too large: algorithm could oscillate around minimum
    - Optimum: negative inverse of Hessian

$$\eta = -\left(\frac{\partial^2 \mathbf{Er}}{\partial W_i \partial W_j}\right)^{-1}$$

Back Propagation

$$t(\vec{x}) = A^{(2)} \sum_{j}^{m} w_j^{(2)} y_j^{(2)}(\vec{x}) \quad \text{where} \quad y_j^{(2)}(\vec{x}) = A^{(1)} \sum_{i}^{n} w_{ij}^{(1)} x_i$$



- Loss function MSE: $E = \frac{1}{2}(t_{\text{true}} - t)^2$ and activation function $A^{(1)}$ = tanh(x)
- Change of weights between hidden and output layer (2):

$$\Delta w_j^{(2)} = -\eta \frac{\partial E}{\partial w_j^{(2)}} = -\eta \frac{\partial E}{\partial t} \frac{\partial t}{\partial w_j^{(2)}} = -\eta (t_{\text{true}} - t) y_j^{(2)}$$

- Change of weights between input and hidden layer (1):

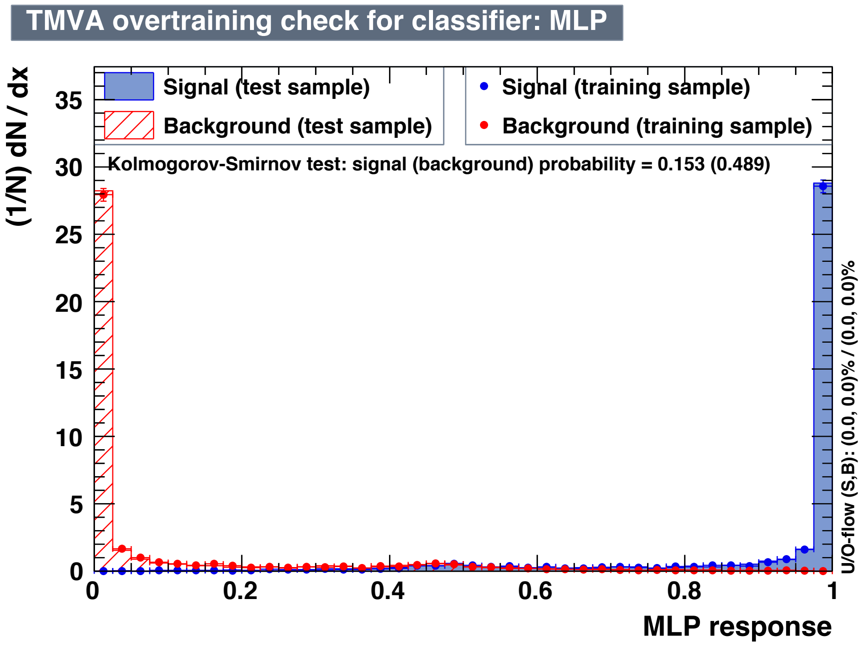
$$\Delta w_{ij}^{(1)} = -\eta \frac{\partial E}{\partial w_{ij}^{(1)}} = -\eta \frac{\partial E}{\partial t} \frac{\partial t}{\partial y_j^{(2)}} \frac{\partial y_j^{(2)}}{\partial w_{ij}^{(1)}} = -\eta (t_{\text{true}} - t) \cdot y_j^{(2)} (1 - y_j^{(2)}) w_j^{(2)} \cdot x_i$$

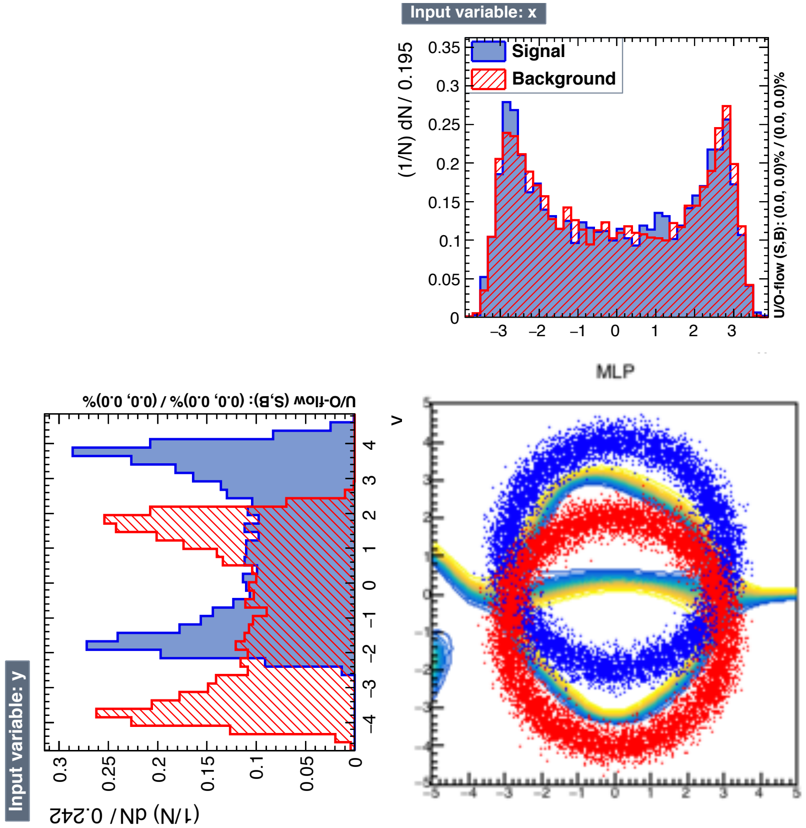
Chain rule: back propagation simplifies into passing of actual numbers

cf. TMVA Users Guide p112

Neural Network in TMVA

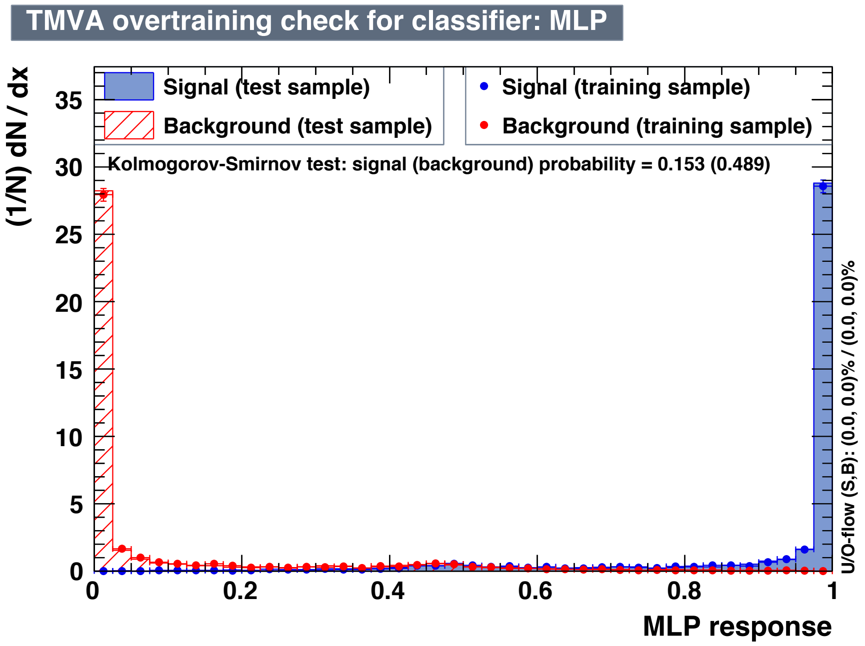
- MLP: two input variables, 8 hidden nodes, sigmoid activation function, 600 learning cycles ("epochs"), 10000 events
- Training: 23s, application: 0.04 s
- Compare with BDT: 2s and 0.5s

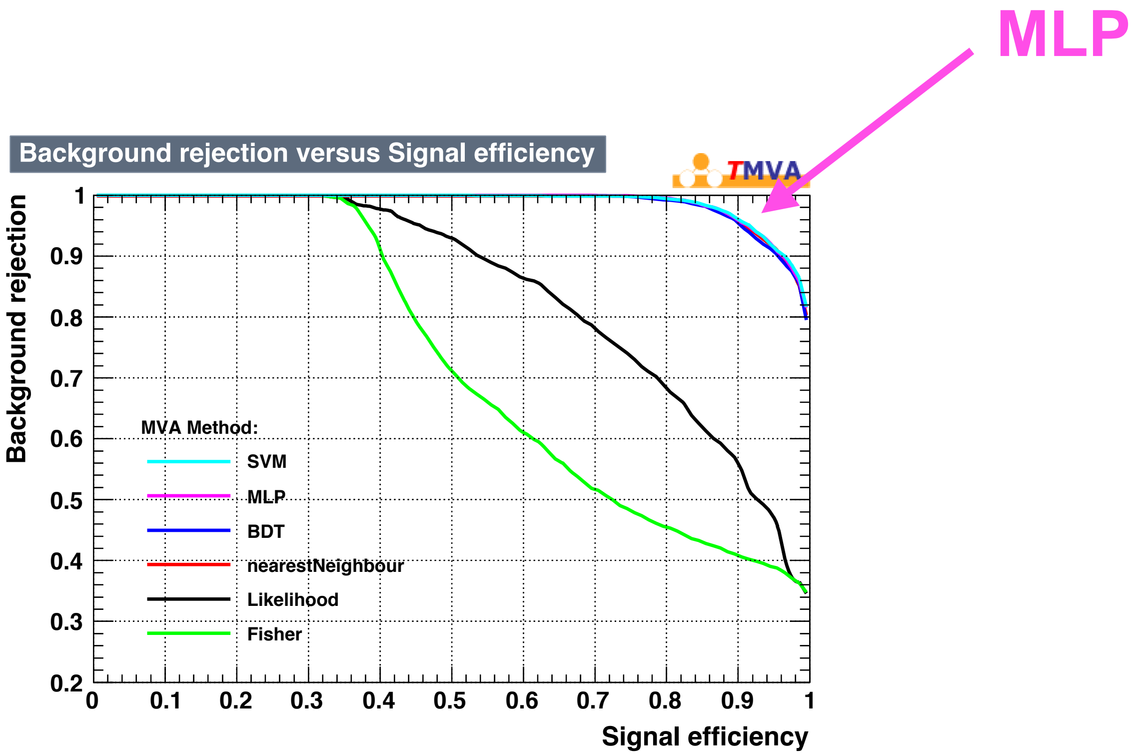




Neural Network in TMVA

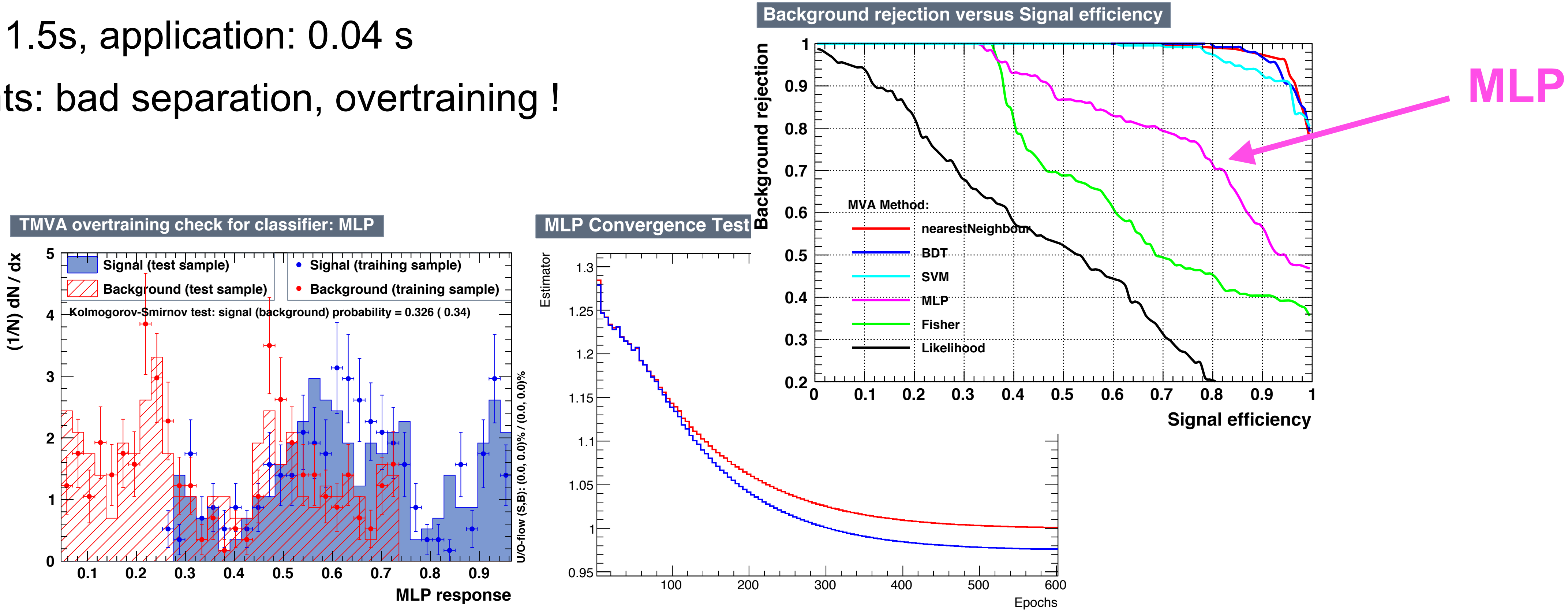
- MLP: two input variables, 8 hidden nodes, sigmoid activation function, 600 learning cycles ("epochs"), 10000 events
- Training: 23s, application: 0.04 s
- Very good separation



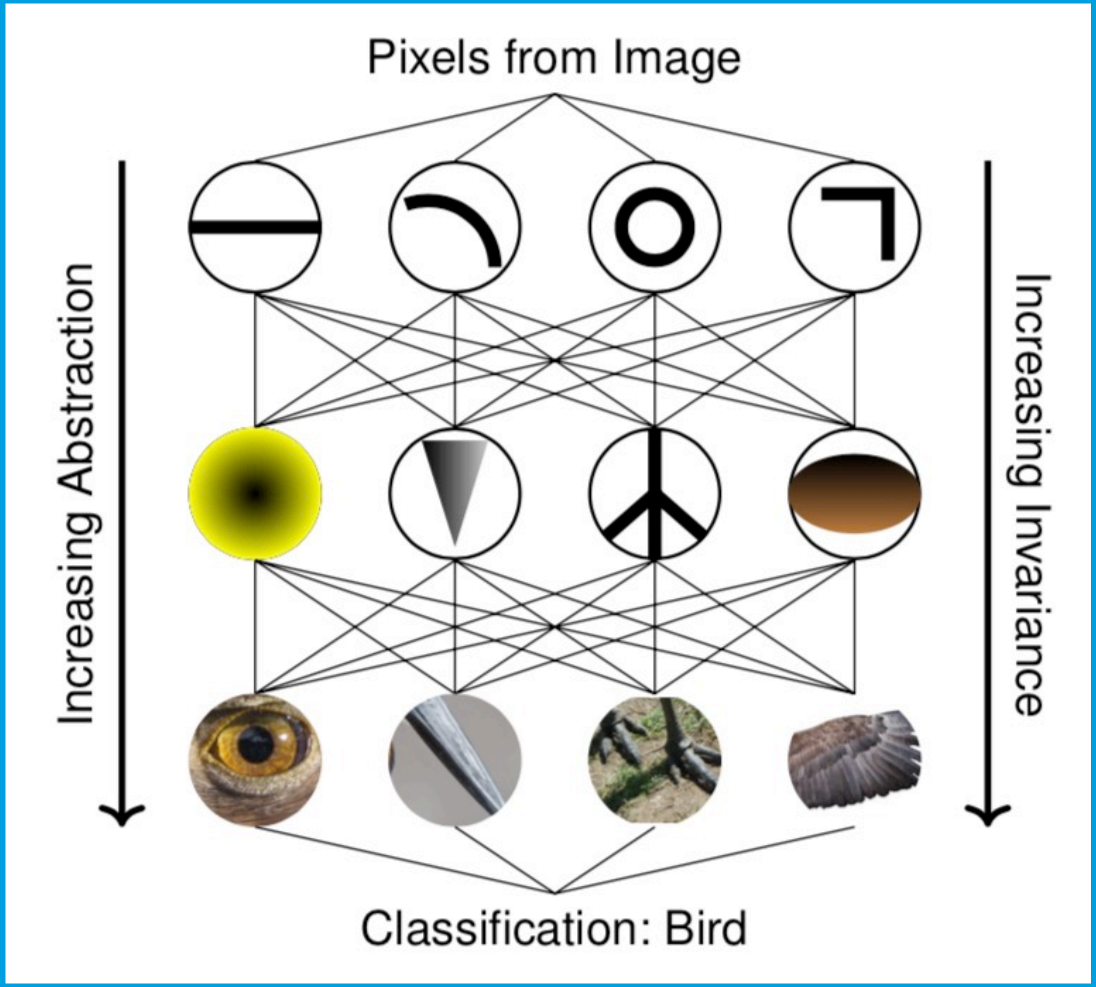


Neural Network in TMVA

- MLP: two input variables, 8 hidden nodes, sigmoid activation function, 600 learning cycles ("epochs"), 500 events
- Training: 1.5s, application: 0.04 s
- 500 events: bad separation, overtraining !

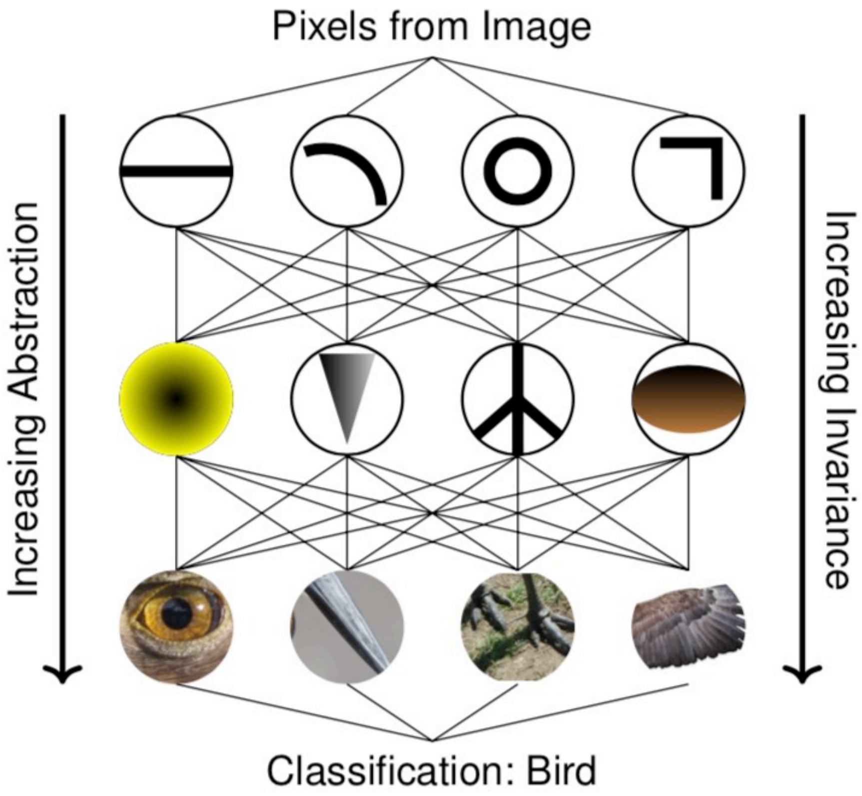


Deep Learning



Representation Learning

http://xkcd.com/1425/



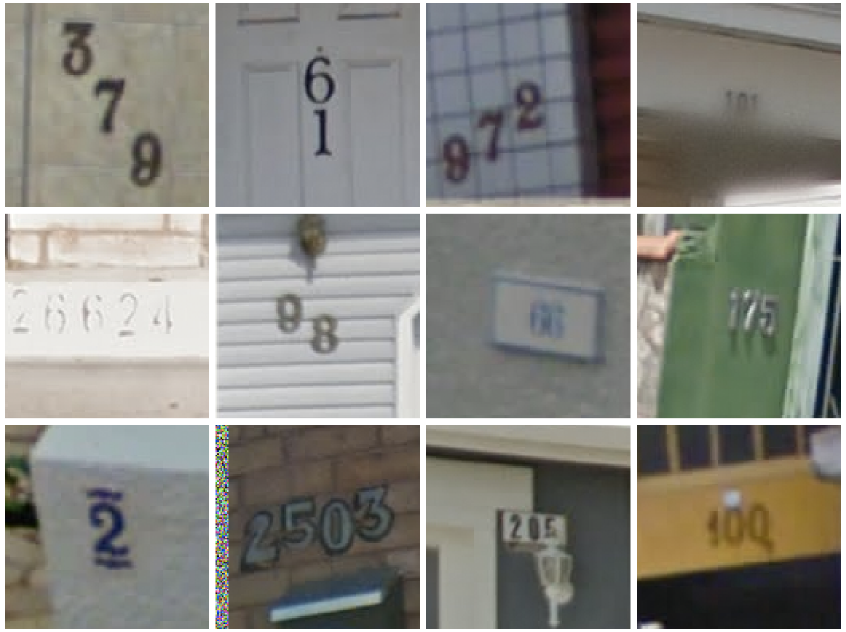
Learned features in deeper layers are increasingly invariant to local changes of the input

Historical Perspective

Benchmarks

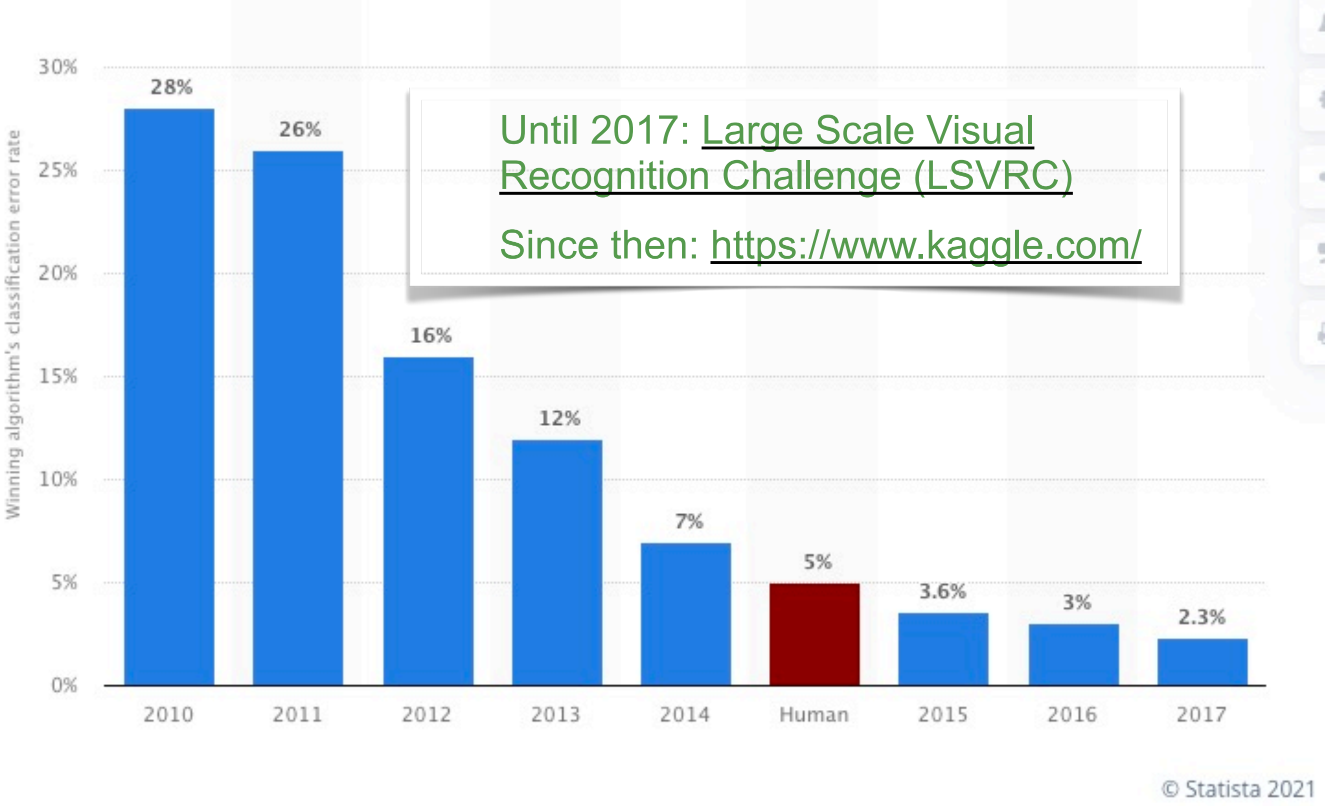
- In 2015, machine's error rates passed that of humans
- Benchmarks today:
  - MNIST dataset: 99.8% correct recognition 1.5 million parameters
  - ImageNet: 90.2% using up to 1 billion parameters

Goodfellow et al.: Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks [arxiv:1312.6082]

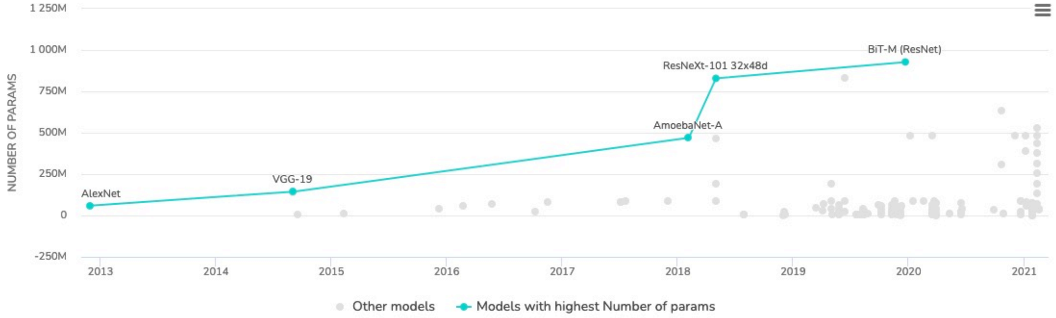


Until 2017: [Large Scale Visual Recognition Challenge (LSVRC)]

Since then: [https://www.kaggle.com/]



[https://www.paperswithcode.com/task/image-classification]:

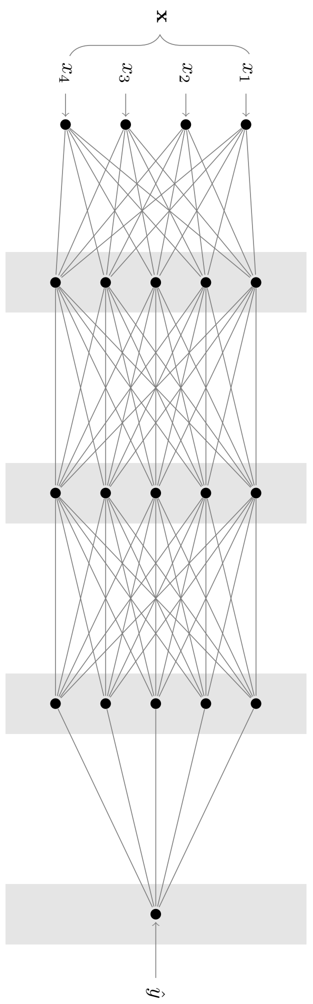


Deep Neural Networks

- Universal Approximation Theorem ([Hornik et al, 1989]): A neural network with a single hidden layer can approximate any function.
  - No statement about the number of nodes.
  - No guarantee that such a network can actually be trained successfully

- Deeper models can deliver better results with the same number of parameters

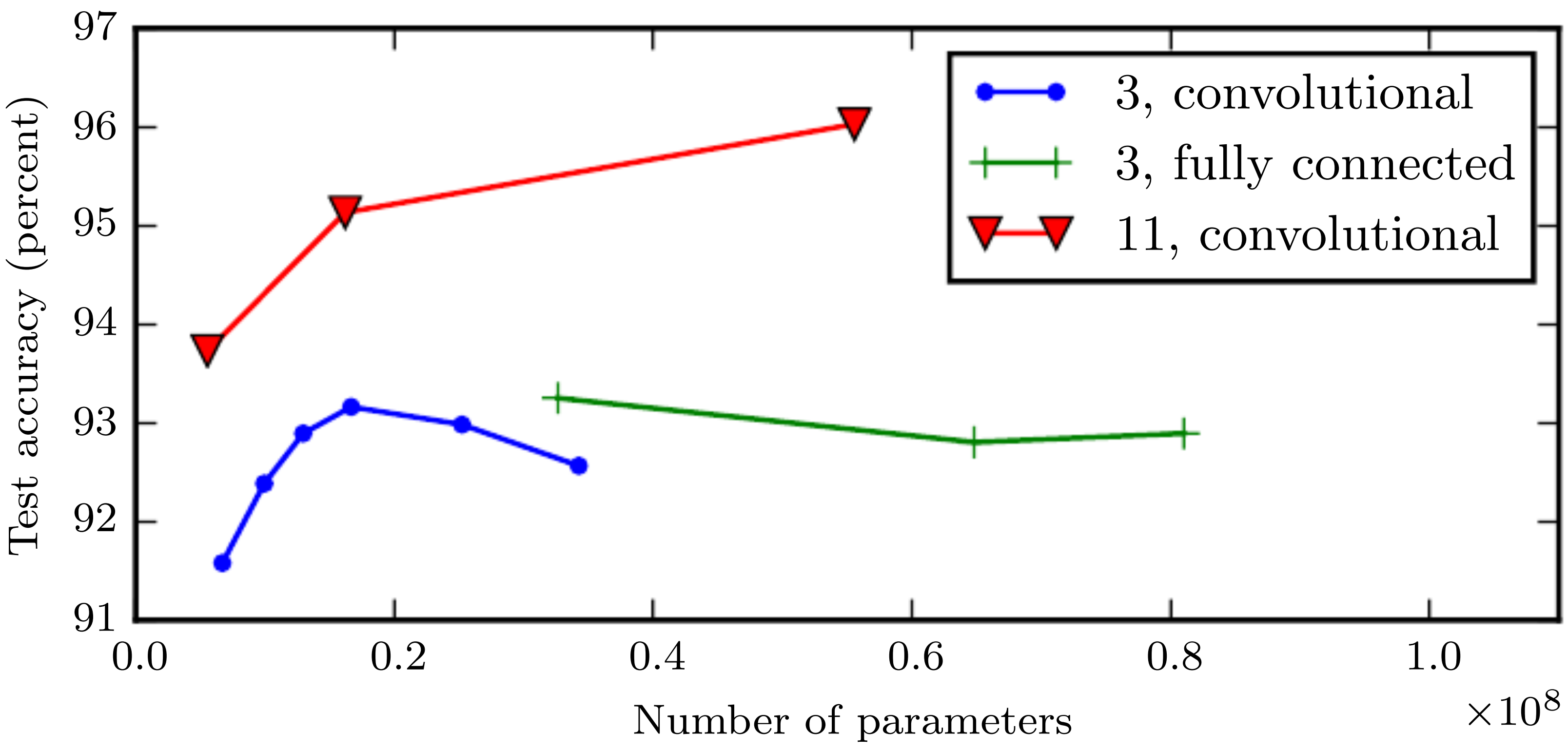
- Still a connectionist idea: complex function is composed of several simple functions → Representation Learning



Deep Neural Networks

For Picture Recognition

deeplearningbook, Fig 6.7



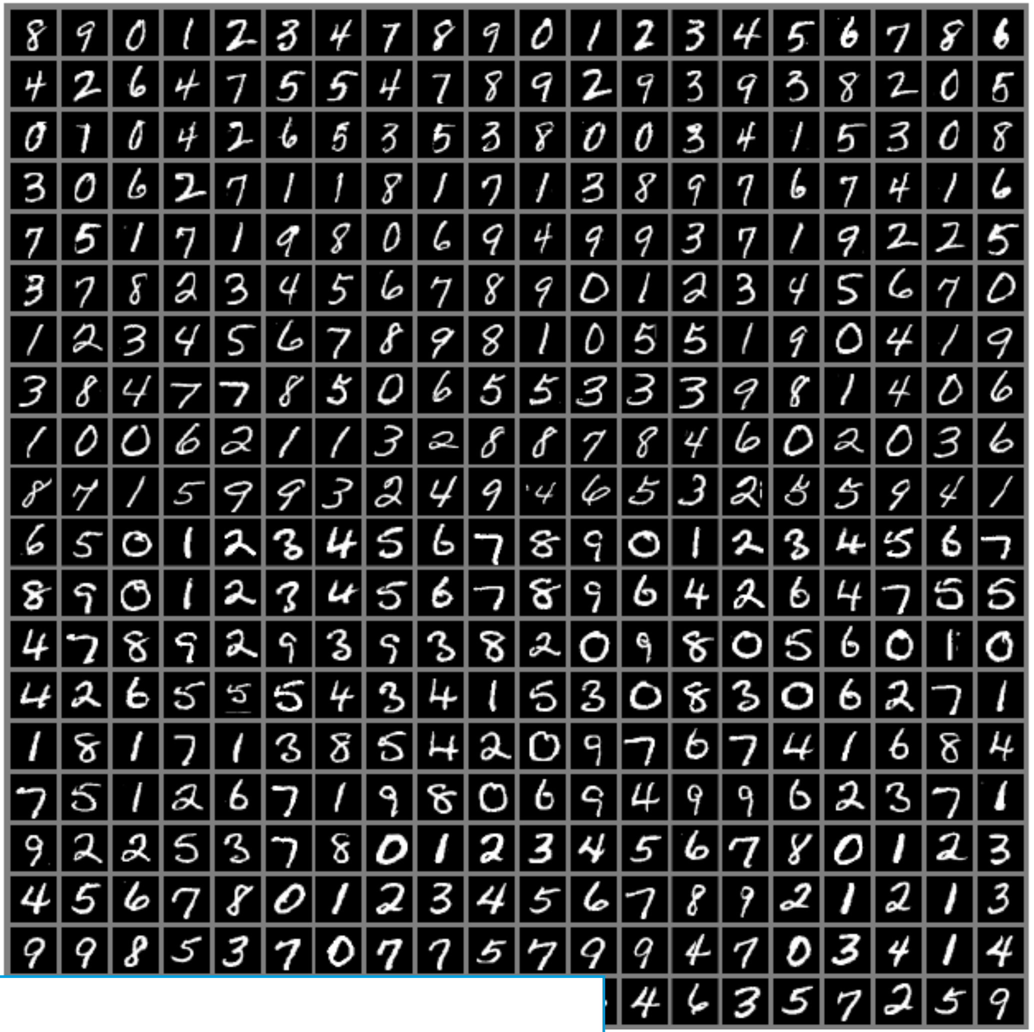
- For the same number of parameters, deeper models can deliver better results (Goodfellow et al, 2014).

Historical Perspective

- The MNIST dataset (1998)
  - "NIST" stands for National Institute of Standards and Technology, the agency that originally collected this data. http://yann.lecun.com/exdb/mnist/
  - "M" stands for "modified," data has been preprocessed for easier use

- Today:
  - Still used as sample for benchmark tests
  - Training on a laptop takes a few minutes

deeplearningbook, Fig 1.9



Keras/Tensorflow example (requires python3.10 and pip):
http://www.desy.de/~ameyer/da_kseta_22/codesnippets/deeplearning.tar
http://www.desy.de/~ameyer/da_kseta_22/codesnippets/deeplearning/tex/Exercise.pdf
User: Students pw: only

Keras Interface to Tensorflow

Example

- ConvNet
- Activation
- Max-Pooling
- Flatten
- Dense
- Activation
- Dropout
- Dense
- Activation

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 27, 27, 4)         20
_________________________________________________________________
activation (Activation)      (None, 27, 27, 4)         0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 4)         0
_________________________________________________________________
flatten (Flatten)            (None, 676)               0
_________________________________________________________________
dense (Dense)                (None, 16)                10832
_________________________________________________________________
activation_1 (Activation)    (None, 16)                0
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                170
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 11,022
Trainable params: 11,022
Non-trainable params: 0
```

- Very simple to set up complex architectures
- Extremely fast optimisation algorithms

Keras/Tensorflow example (requires python3.10 and pip):
http://www.desy.de/~ameyer/da_kseta_22/codesnippets/deeplearning.tar
http://www.desy.de/~ameyer/da_kseta_22/codesnippets/deeplearning/tex/Exercise.pdf

Deep Learning
Summary

- Multilayer Neural Networks show a better performance than single-hidden layer ANN
  - More separation power with less nodes

- Breakthrough around 2015: error rate drops below that of humans.
  - Fast and powerful software and hardware (e.g. GPU)
  - Extremely large (labelled) datasets.

- Different types of deep networks to address specific features
  - Convolutional Neural Nets (CNN)
  - Recurrent Neutral Nets (RNN)
  - Relation Networks (RN)
  - Graph Networks (GN)
  - Generative Adversarial Networks (GAN)
  - Autoencoder (VAE)

- Application in physics: rigorous theory provides important knowledge about correlations between inputs. DL does nevertheless still achieve some improvements.

Conclusions

Conclusions

- Statistical methods to extract maximal information from the data
  - Probabilities: including Frequentist and Bayesian view points
  - Hypothesis tests and confidence intervals: physicists look for correct hybrid methods
  - Profile likelihood ratio: provides signal strength and exclusion limits including systematic uncertainties and correlations
  - Classification: a large-scale application of hypothesis tests
  - Machine learning: BDT, ANN and a superficial look at Deep Learning

- The scientific cycle
  - Interplay between theory and experiment: determine and document observations in a reproducible and/or exp.-independent way
  - Statistical uncertainties: well understood concept
  - Systematic uncertainties:
    - no general rule, often determined from ancillary measurements -> statistical effects
    - calibrations, resolutions, efficiencies
    - proceed with care, reflexion and courage

Backup

Fisher Discriminant

- Maximize $J(\vec{a}) = \frac{(\tau_s - \tau_b)^2}{\Sigma_s^2 + \Sigma_b^2}$
- For $\frac{\partial J(\vec{a})}{\partial a_i} = 0$, one obtains Fisher's linear discriminant

$$t(\vec{x}) = \vec{a}^T \vec{x} \quad \text{mit } \vec{a} \propto W^{-1}(\vec{\mu}_s - \vec{\mu}_b)$$ where $W=\Sigma_s^2+\Sigma_b^2$

- Example: multivariate Gauss distributions with covariance matrix V, i.e.

$$t(\vec{x}) = V^{-1}(\vec{\mu}_s - \vec{\mu}_b)\vec{x} + a_0$$

- Compare to likelihood ratio: Fisher discriminant is equivalent, i.e. monotonic function of x:

$$r = \frac{g(\vec{x}|H_s)}{g(\vec{x}|H_b)} = \exp\left[-\frac{1}{2}(\vec{x}-\vec{\mu}_s)^T V^{-1}(\vec{x}-\vec{\mu}_s) + \frac{1}{2}(\vec{x}-\vec{\mu}_b)^T V^{-1}(\vec{x}-\vec{\mu}_b)\right]$$

$$= \exp\left[(\vec{\mu}_s - \vec{\mu}_b)^T V^{-1}\vec{x} - \frac{1}{2}\left(\vec{\mu}_s^T V^{-1}\vec{\mu}_s - \vec{\mu}_b^T V^{-1}\vec{\mu}_b\right)\right] \propto \exp[t(\vec{x})]$$

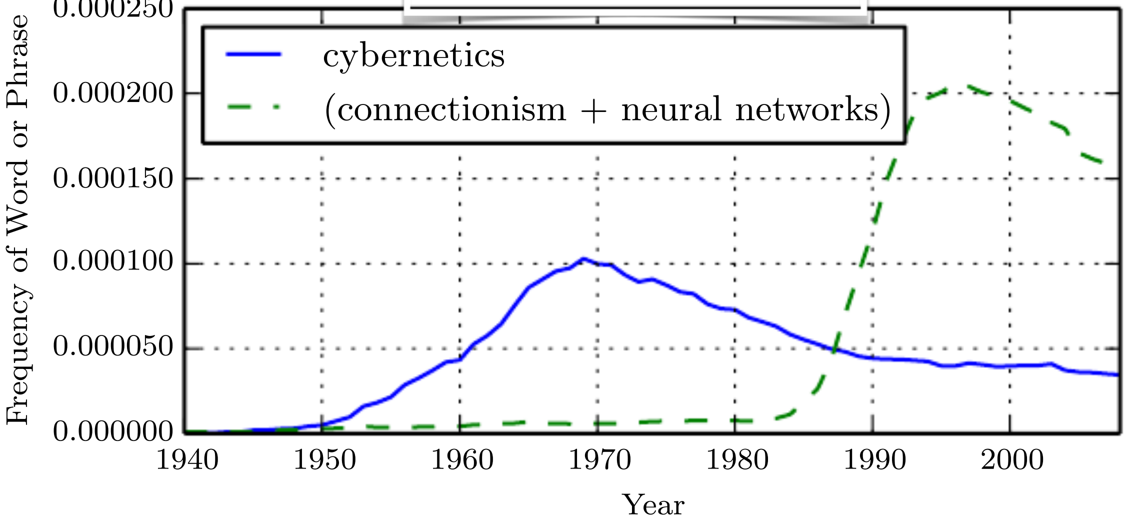
Historical Perspective

- 1940s: cybernetics (information transfer in machine and animals)
  - Research on machine learning starts with the linear perceptron
  - Interest decreases due to (supposed) limitations (e.g. XOR problem, Minsky 1969)

- 1980s: connectionism (many simple functions combined can solve complex problems)
  - ANN, BDT, SVN: good results for many non-linear problems
  - Very high expectations, initially not met (esp. slow training of ANN and application speed of BDT)
  - TMVA (since 2005) and scikit-learn (2010) were built on these (and other methods)

- Currently: Deep Learning
  - Tensorflow/keras, pytorch
  - Fast-growing number of extremely powerful tools and techniques
  - No limits in sight (?!)

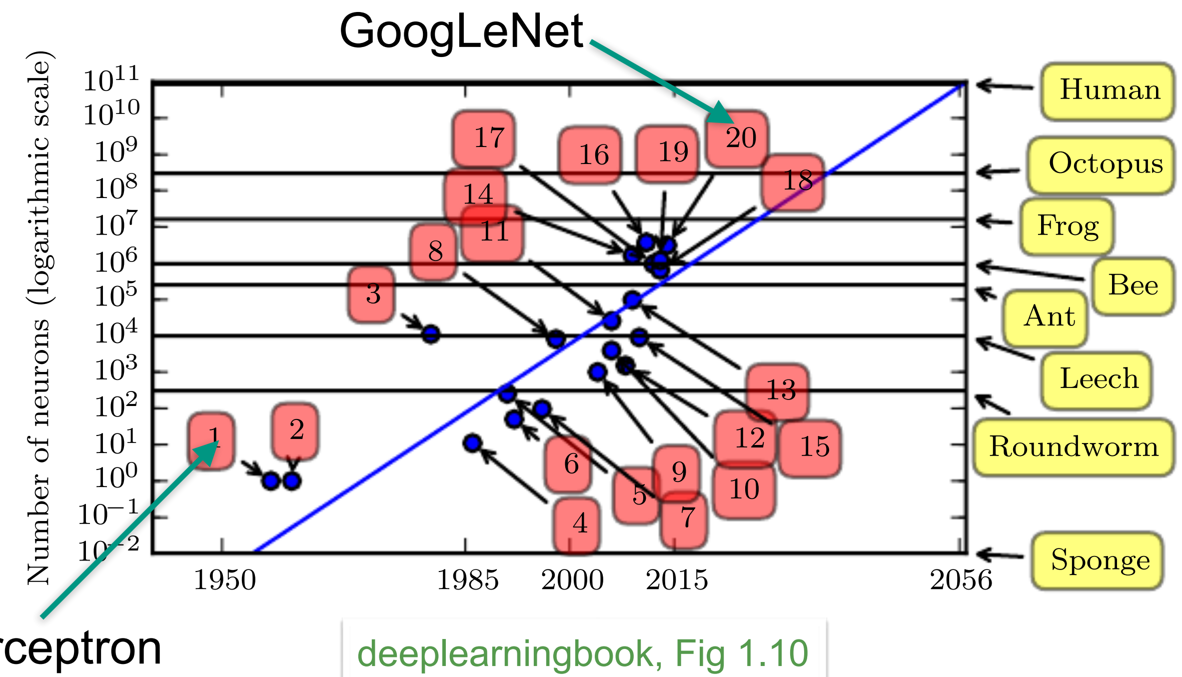
deeplearningbook, Fig 1.7



Historical Perspective

- Originally strong interest in unsupervised learning techniques ("artificial intelligence")
- Today predominantly supervised learning with (extremely large) data samples
- Huge commercial interests, modern software packages, powerful computing (GPU)
- Number of neurons in functional networks doubles roughly every 2.4 years. Status 2016: ~$5 \cdot 10^6$ parameters (about the size of the nervous system of insects)

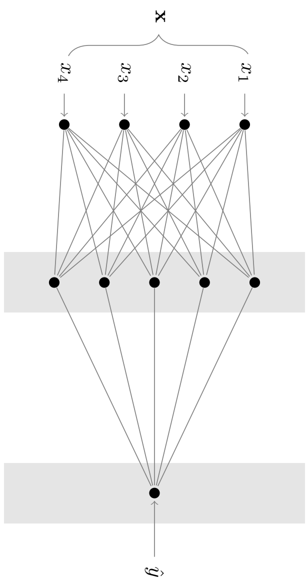
Large number of neurons require extremely large training datasets - and good software and computing



deeplearningbook, Fig 1.10

Deep Neural Networks

- Universal Approximation Theorem (Hornik et al, 1989):
  A neural network with a single hidden layer can approximate any function.
  - No statement about the number of nodes.
  - No guarantee that such a network can actually be trained successfully

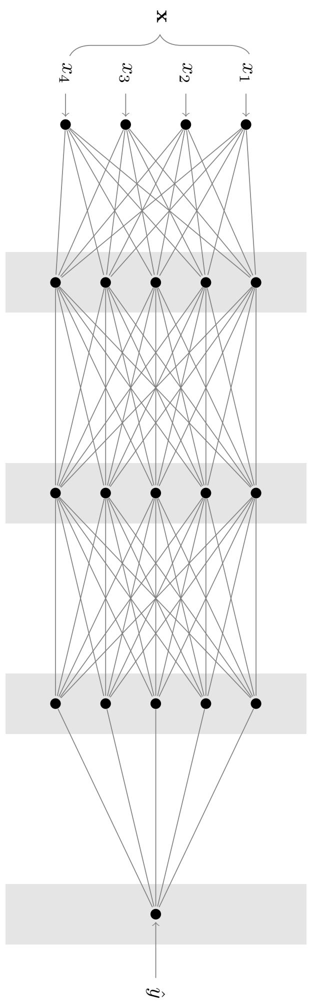


Deep Neural Networks

- Universal Approximation Theorem (Hornik et al, 1989):
  A neural network with a single hidden layer can approximate any function.
  - No statement about the number of nodes.
  - No guarantee that such a network can actually be trained successfully

- Deeper models can deliver better results with the same number of parameters

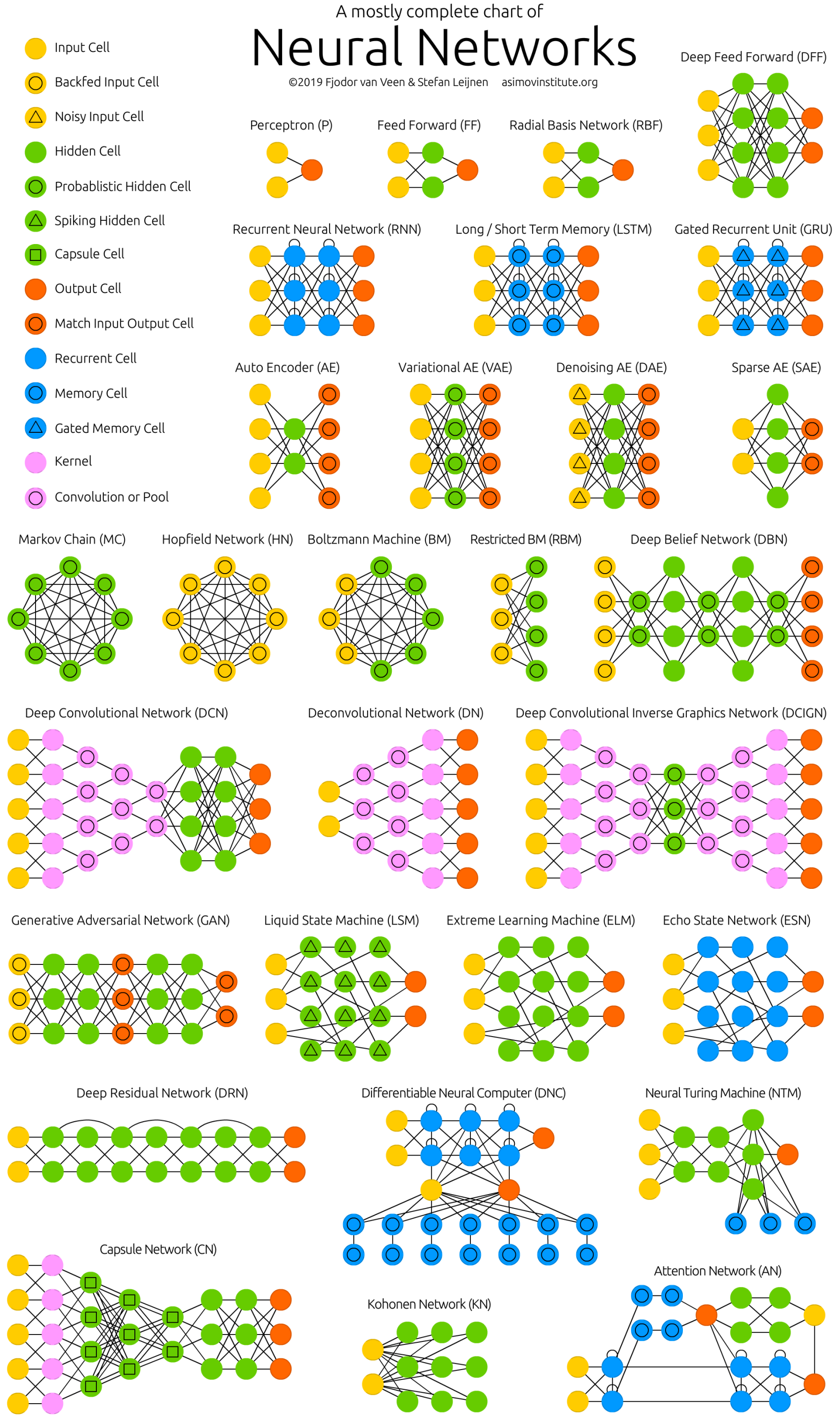
- Still a connectionist idea: complex function is composed of several simple functions → Representation Learning



Network Architectures

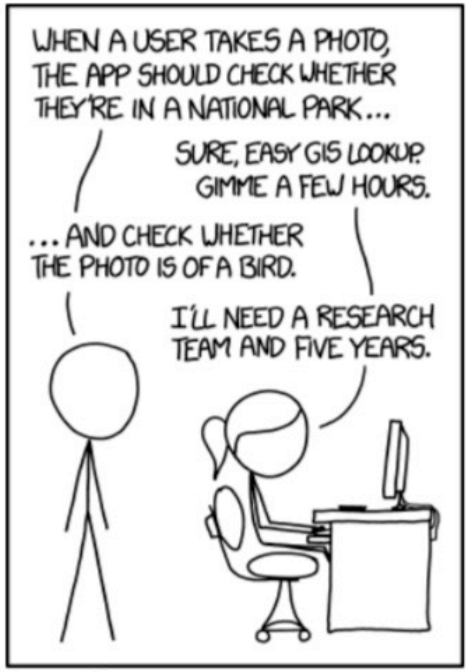
- Feedforward Neural Network
  - Dense Layers: completely connected layers
  - No feedback connections

- Reduce number of parameters (“sparsification” or “complexity reduction”) without performance loss: use specific types of nodes as building blocks
  - Convolutional Neural Networks / Representation Learning
  - Recurrent Neural Networks
  - Relation Networks
  - Graph Networks
  - Adversarial Networks
  - Autoencoders
  - ……..



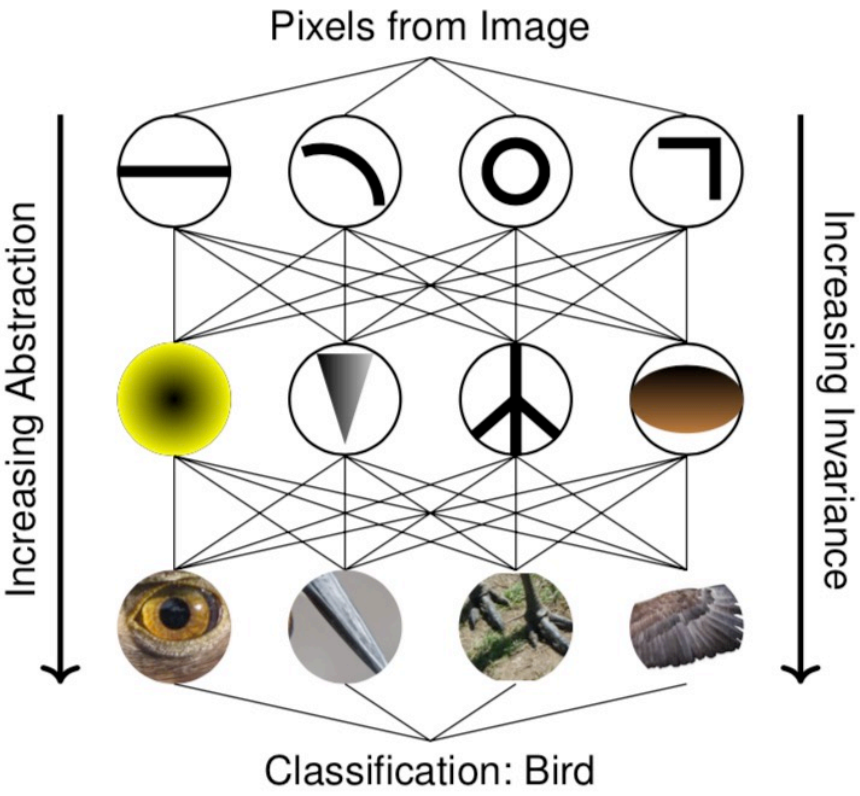
https://www.asimovinstitute.org/neural-network-zoo/

Representation Learning



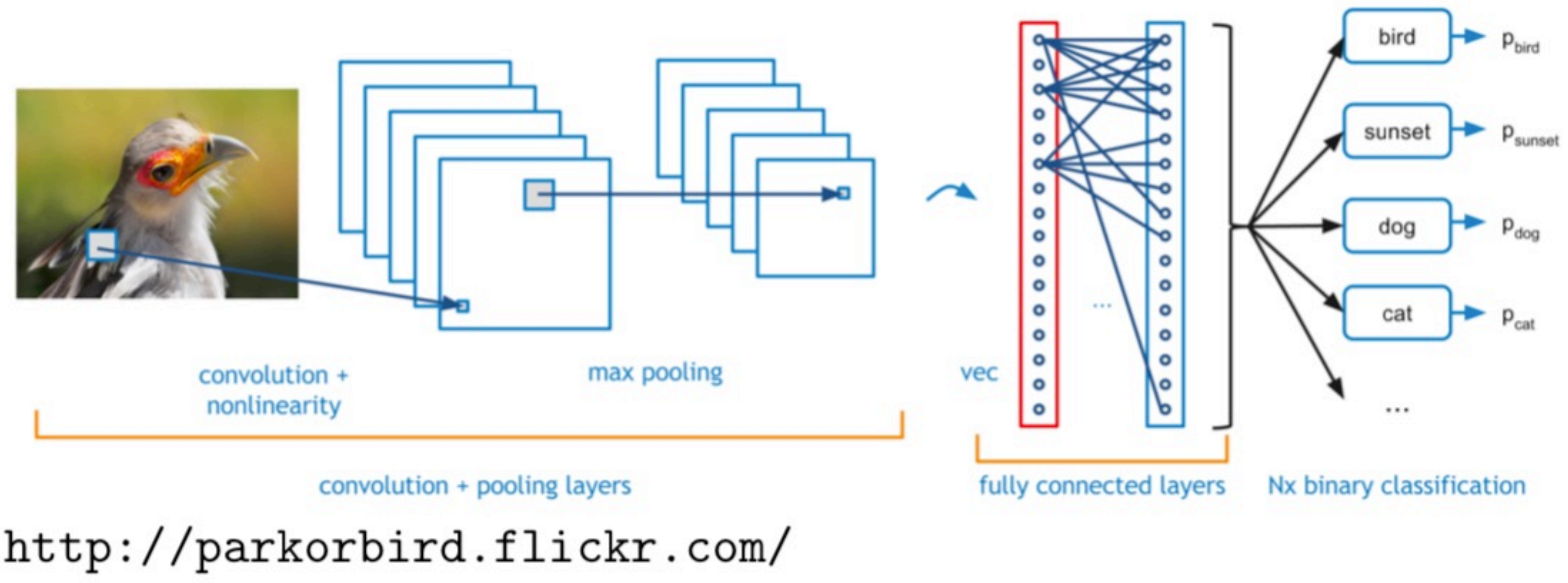
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

http://xkcd.com/1425/



Learned features in deeper layers are increasingly invariant to local changes of the input

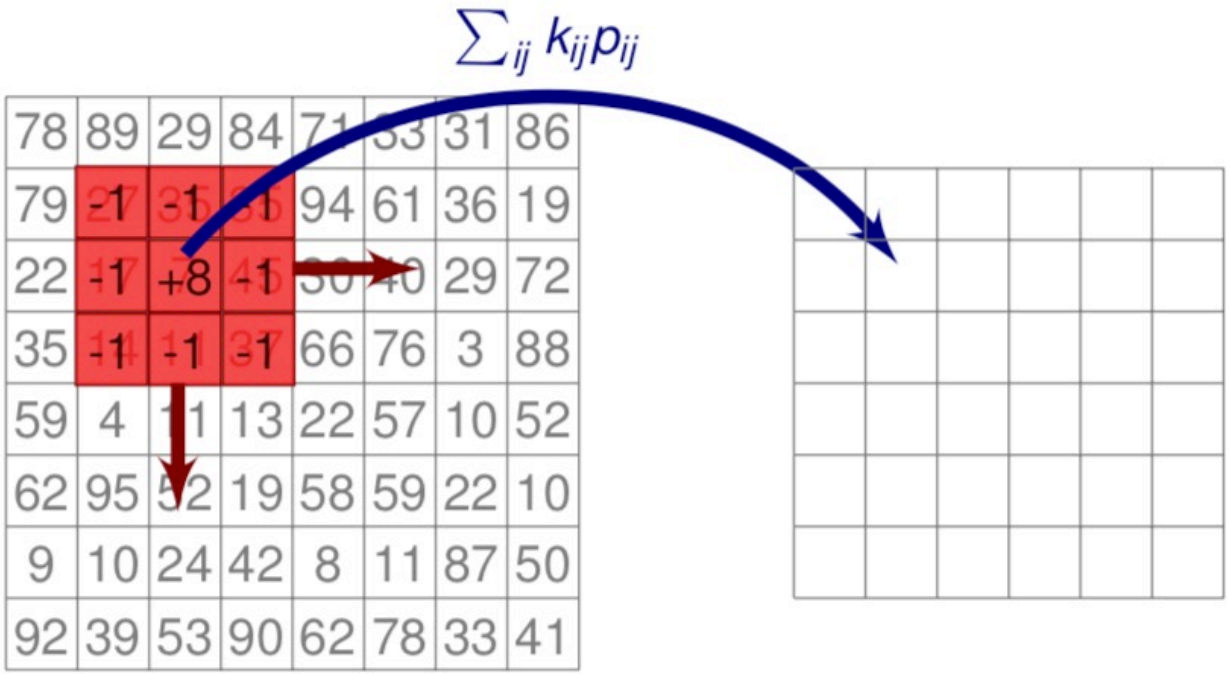
Representation Learning



◉ Sequence of multiple convolution and pooling layers, finish with a deep net of fully connected layers

Slides: Thomas Keck, Deep Learning Lecture KIT 2017 (with A.B.Meyer)

Convolutional Layer



- **depth** – number of filters (also known as kernels)
- **size** – dimension of the filter e.g. 3×3 or $3 \times 3 \times 4$
- **stride** – step size while sliding the filter through the input
- **padding** – behavior of the convolution near the borders

Slides: Thomas Keck, Deep Learning Lecture KIT 2017 (with A.B.Meyer)

Convolutional Layer

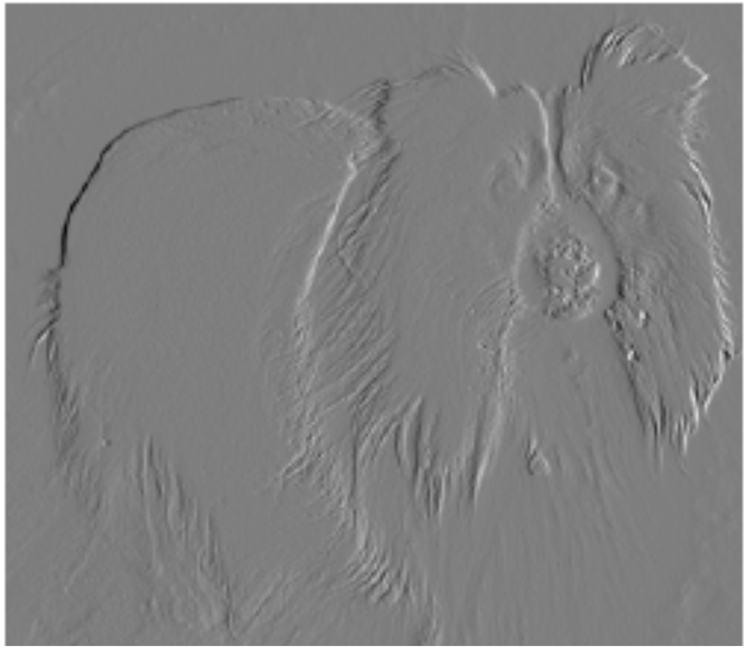
Example

deeplearningbook, Fig. 9.6

320 Pixels

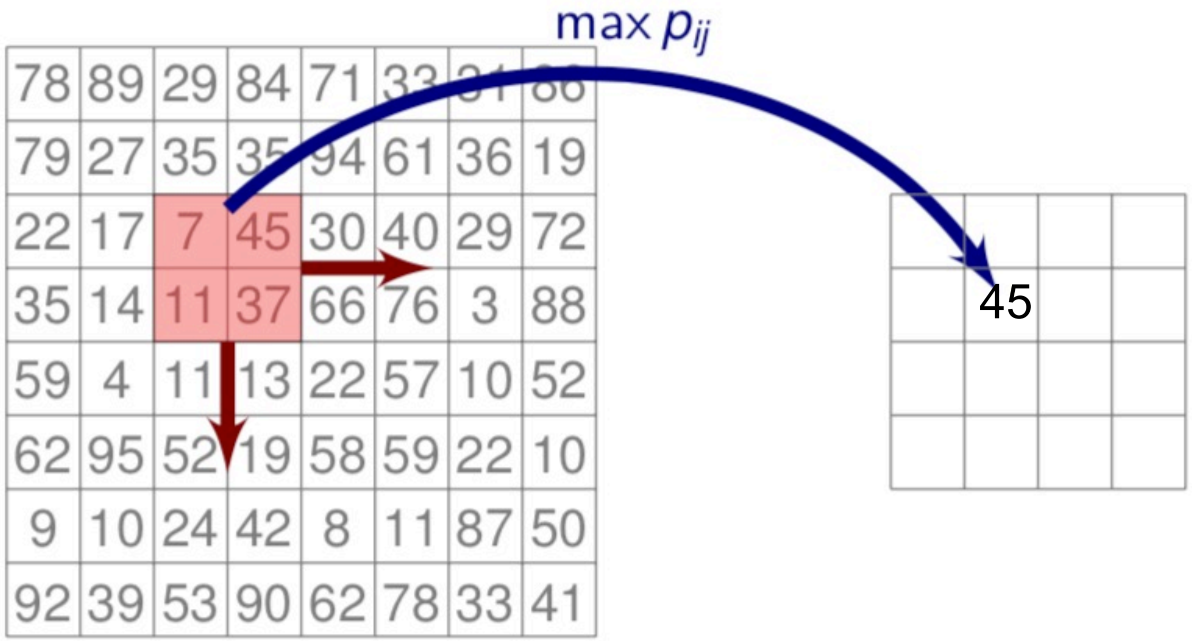
280 Pixels

319 Pixels



- Vertical edges by subtraction of each original pixel value by the value of the pixel to the left (transformation described by a Conv.Net with appropriate Kernel)
- A few simple computations lead to drastic reduction of the number of relevant pixels, i.e. training parameters, without much loss of information.

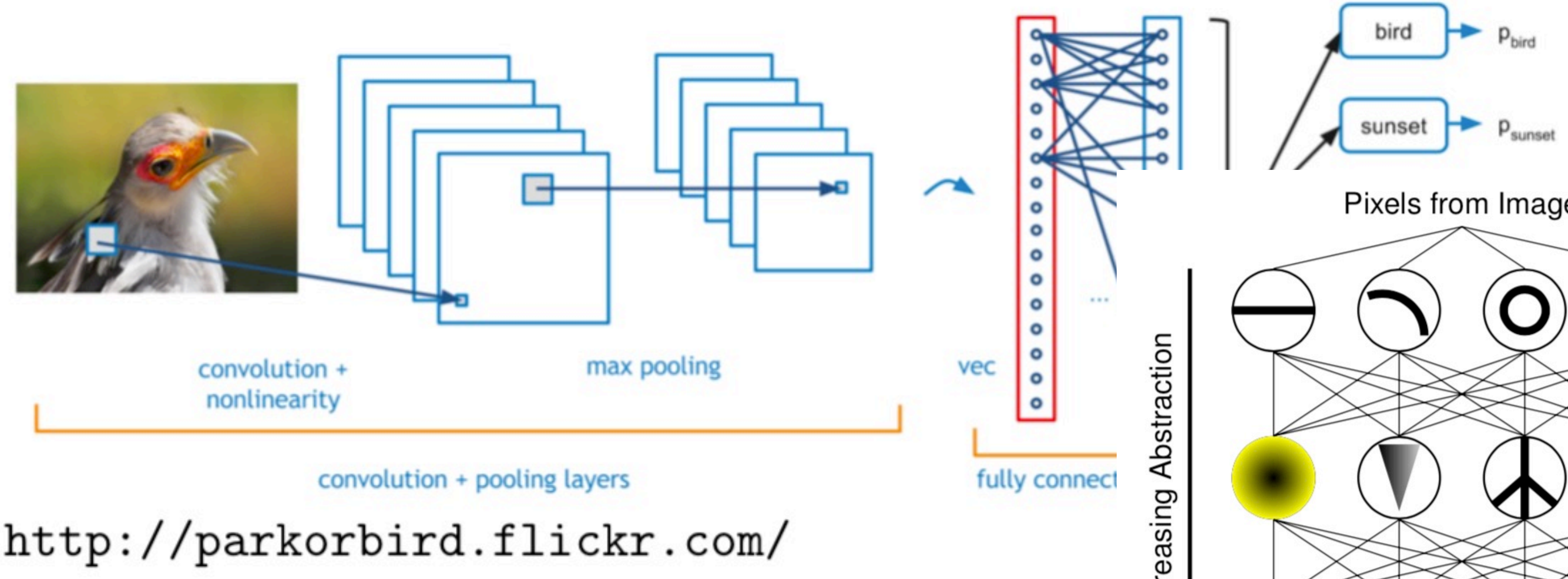
Max Pooling

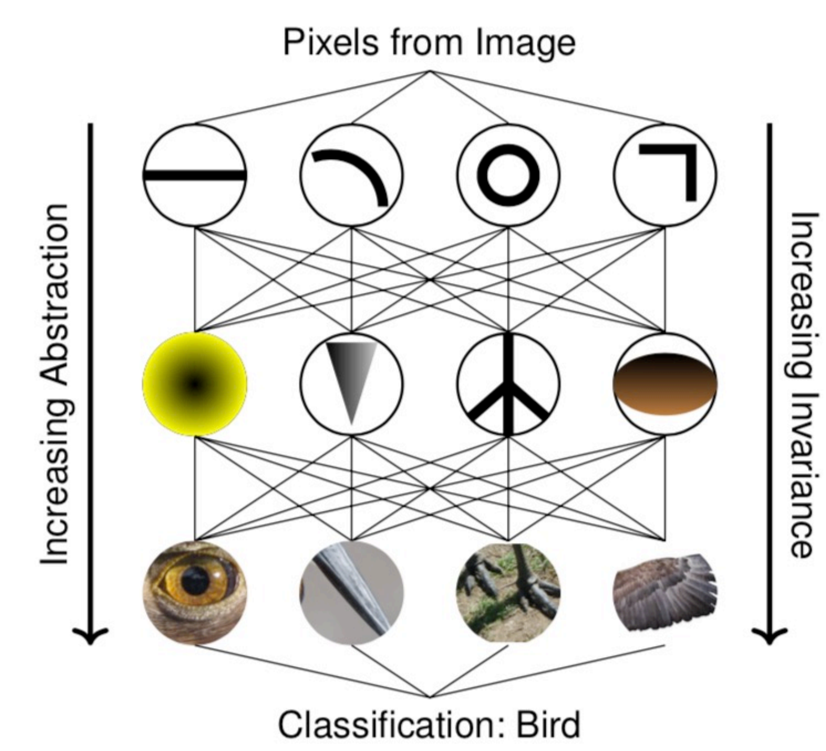


- **depth** – number of filters (also known as kernels)
- **size** – dimension of the filter e.g. 2×2 or $2 \times 2 \times 4$
- **stride** – step size while sliding the filter through the input
- **padding** – behavior of the convolution near the borders

Slides: Thomas Keck, Deep Learning Lecture KIT 2017 (with A.B.Meyer)

Representation Learning





Slides: Thomas Keck, D

er)

- Each stage has a larger degree of invariance
- Number of features increases as resolution is reduced
- Final layer is fully connected with a multinomial activation function (softmax)

Regularisation

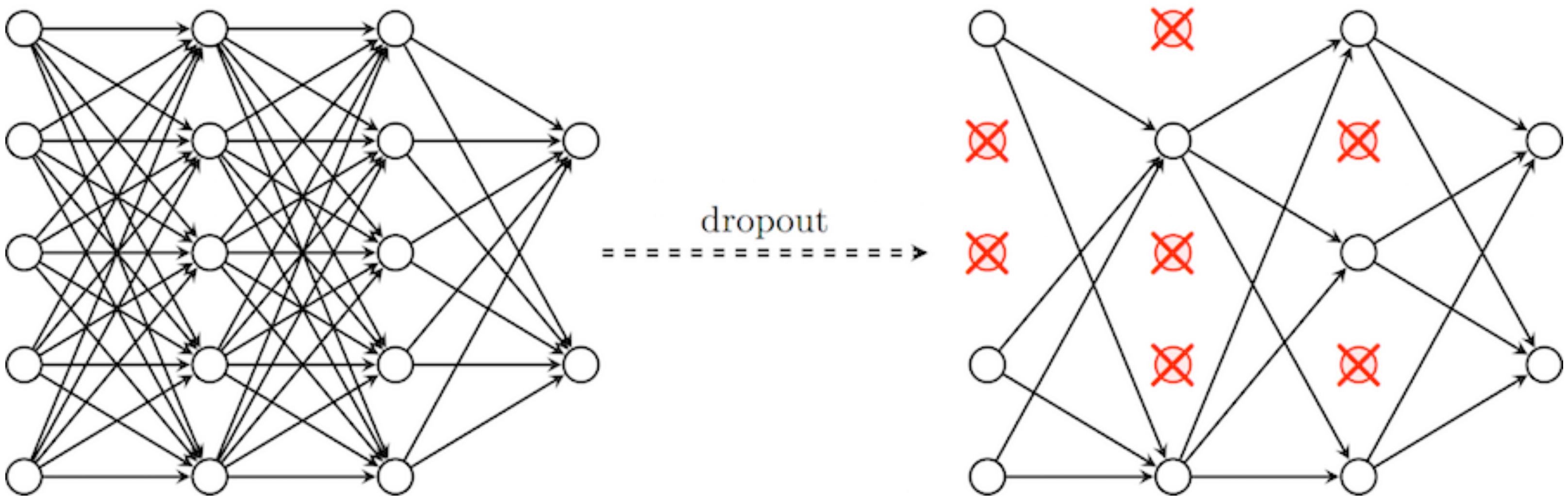
- Modification of the learning algorithm to reduce the generalisation error (avoid over-training)
- Reduce number of parameters w/o capacity loss: “best model (in the sense of minimizing the generalization error) is a large model that has been regularised appropriately.” (DLbook, p229)

- Methods:
  - Early stopping, stop before overtraining
  - Weight decay: penalty terms against high weights
  - Sparse representations: penalty term against activation
  - Drop-Out: remove single nodes during the training. Repeat with different DropOut conditions. Reduce dependence of network behaviour on single nodes
  - Parameter sharing: common parameters across nodes, e.g. ConvNet Kernel
  - Adversarial training: use background to improve robustness.

- In supervised problems in HEP, generation/simulation of more data is often easier than regularisation

Drop Out

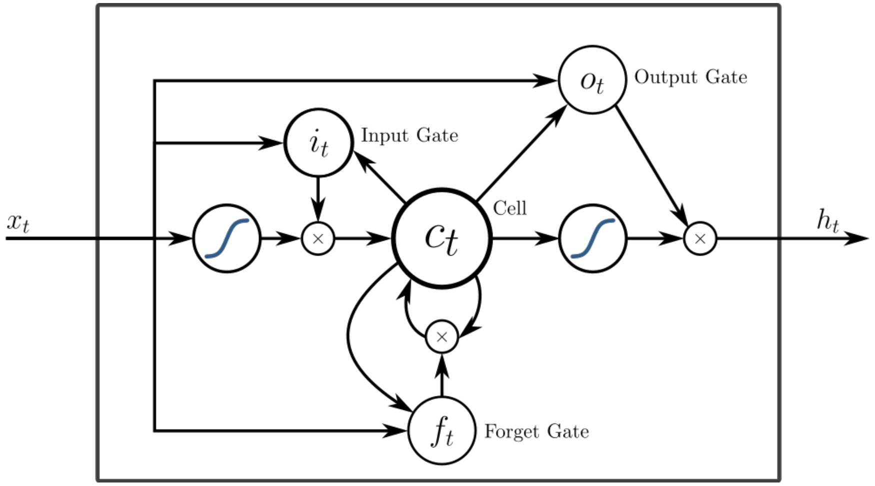
Regularisation



- Drop-Out: reduce dependence of network behaviour on single nodes
  - remove single nodes during the training
  - repeat with different DropOut conditions

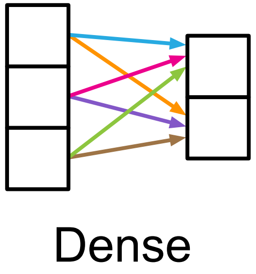
Recurrent Neural Networks

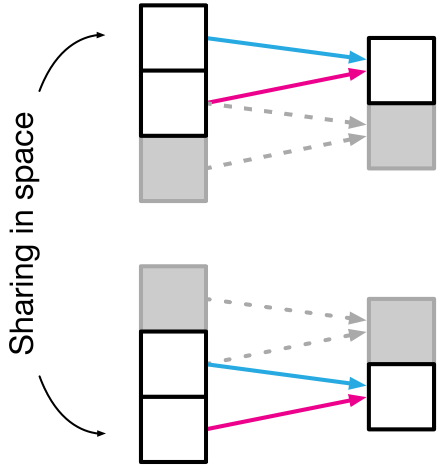
- Share inputs across different time slices
- Long short-term memory



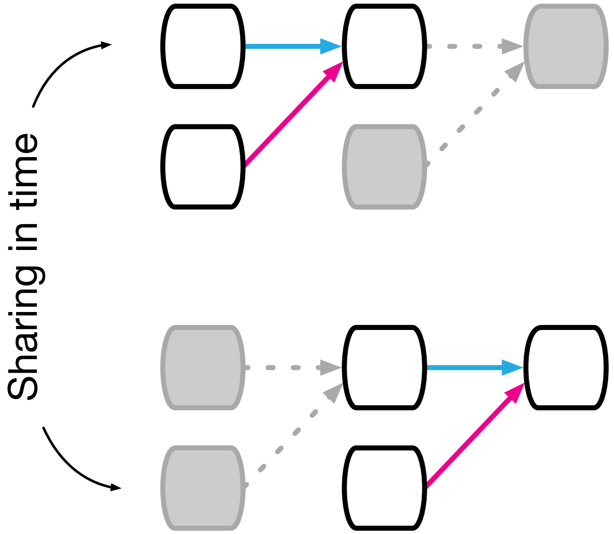
source: https://en.wikipedia.org/wiki/Long_short-term_memory

- resolves back propagation issues (vanishing or exploding gradients)
- input gate decides when to update the stored value
- output gate decides when to output the stored value
- forget gate decides when to forget the stored value





Convolutional Neural Network (CNN)

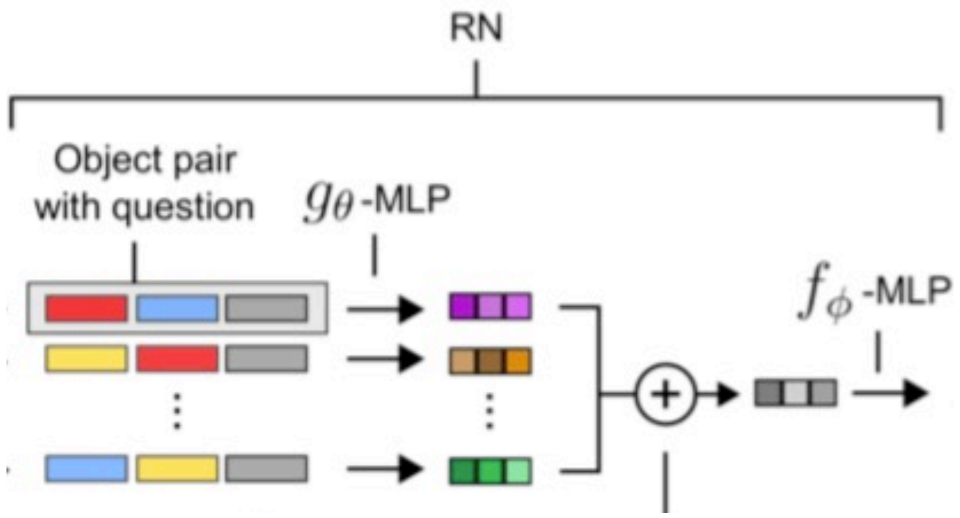


Recurrent Neural Network (RNN)

Relation Networks

Relations between Objects

- Objects (=> nodes)
- Relations (=> weights, i.e. connections)
  - "left of"
  - "same size as"
  - "heavier than …"



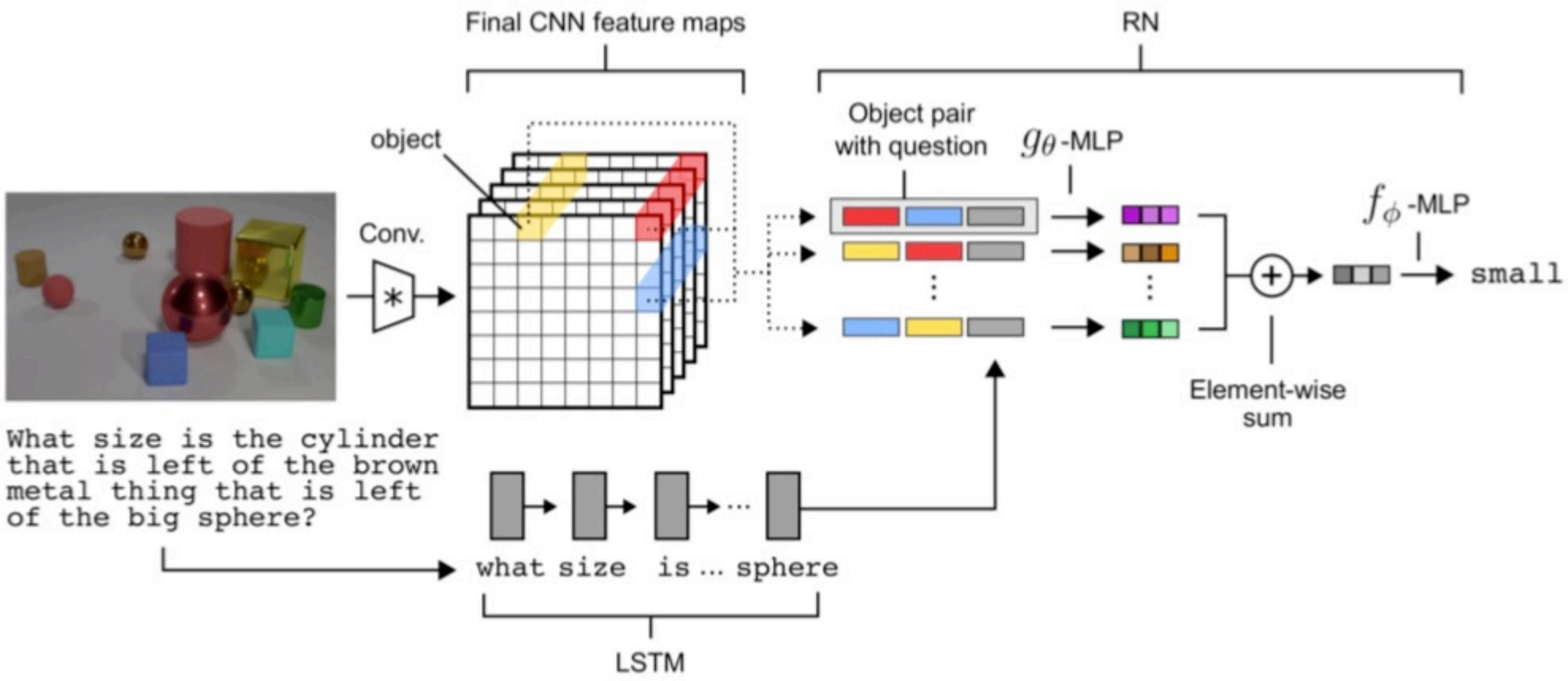
- Reduce complexity through weight-sharing among objects e.g.

$$\mathrm{RN}(o_1, o_2, \ldots, o_n) = f_\phi\left(\sum_{i,j} g_\Theta(o_i, o_j)\right)$$

Relation Networks

Relations between Objects

arXiv:1706.01427



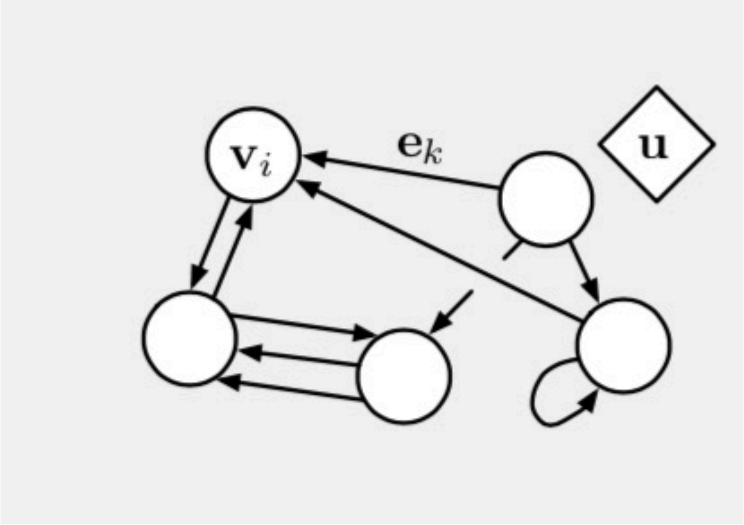
Graph Networks

https://github.com/deepmind/graph_nets

Generalization of Relation Networks

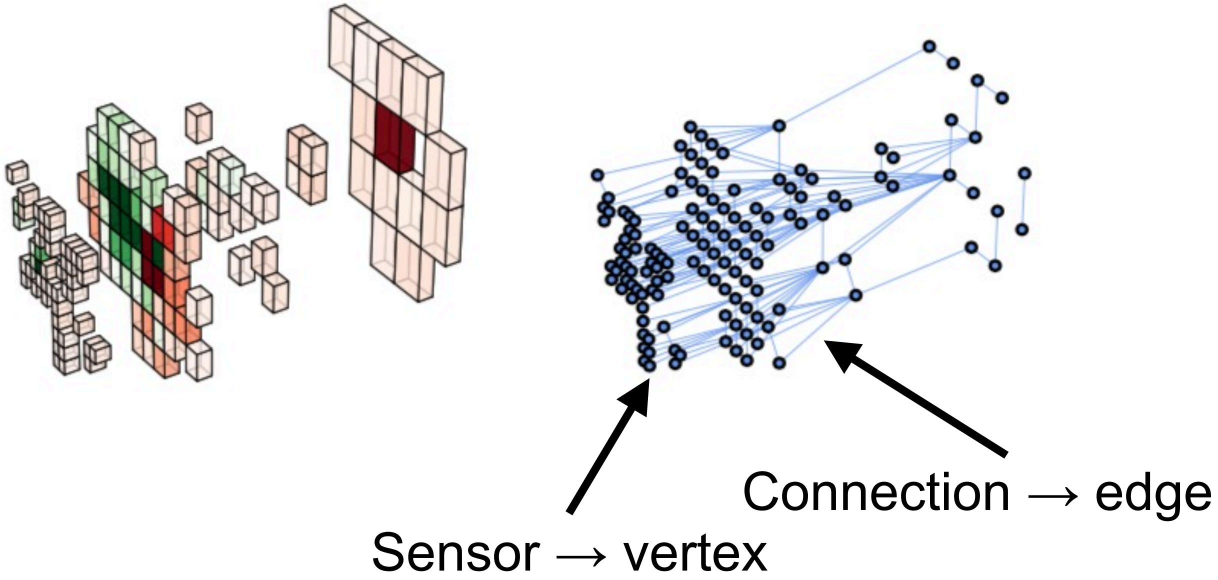
- A graph is a 3-tuple: G = (**u**, **V**, **E**) where
  - **u:** global attributes
  - **V:** a set of nodes (objects) with attributes
  - **E:** the set of edges (relations) with weights

arXiv:1806.01261



arXiv:2007.13681

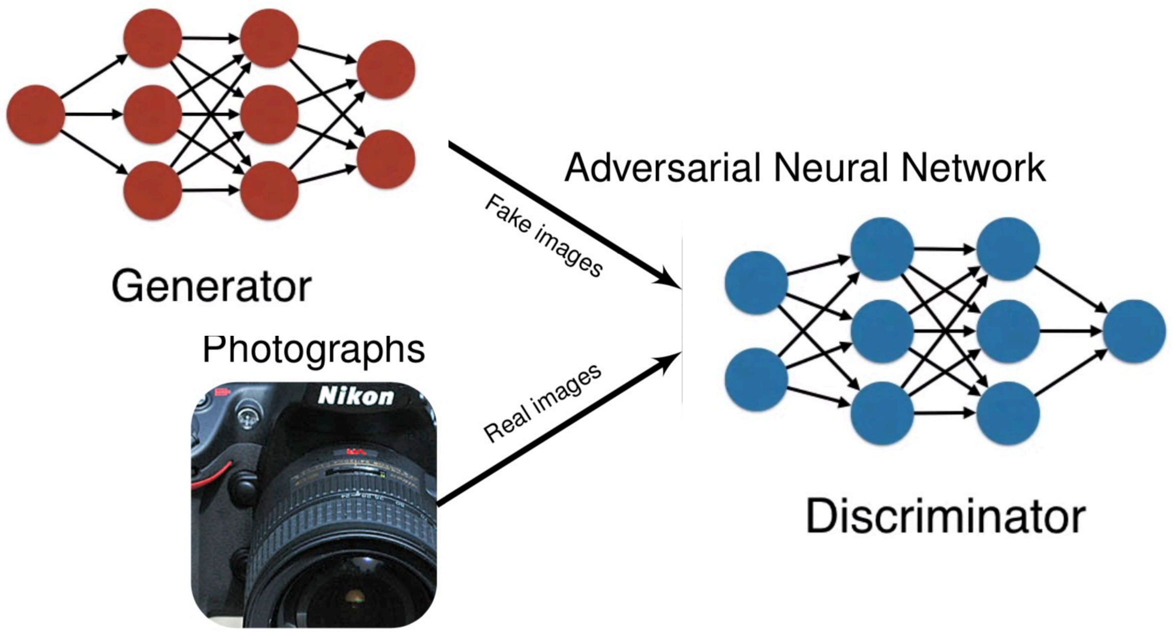
Example: application for calorimeter showers

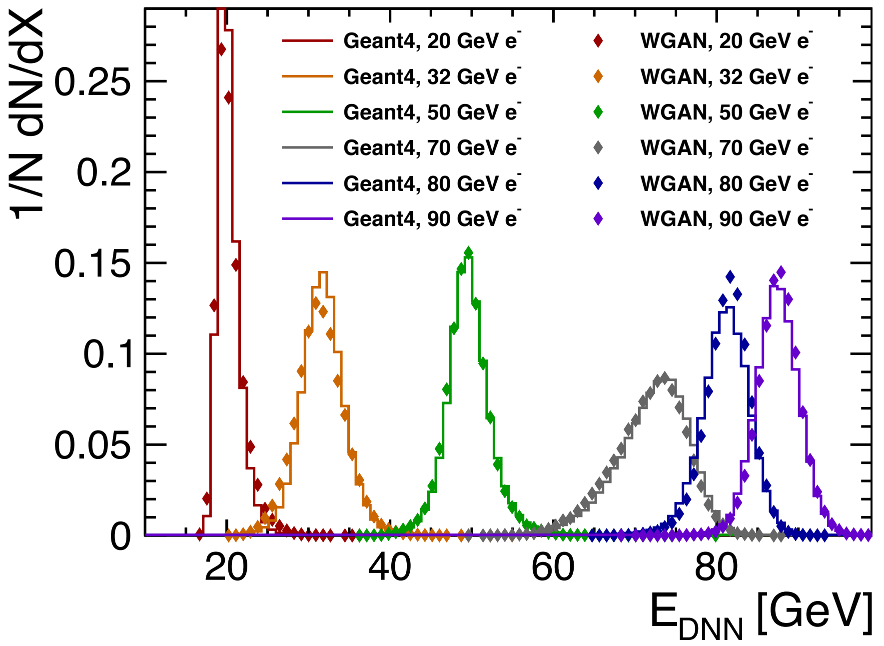


Try out the demos in the paper, e.g. tinyurl.com/gn-shortest-path-demo

Adversarial Neural Networks

A nice video about GANs and forward and back propagation: https://youtu.be/8L11aMN5KY8



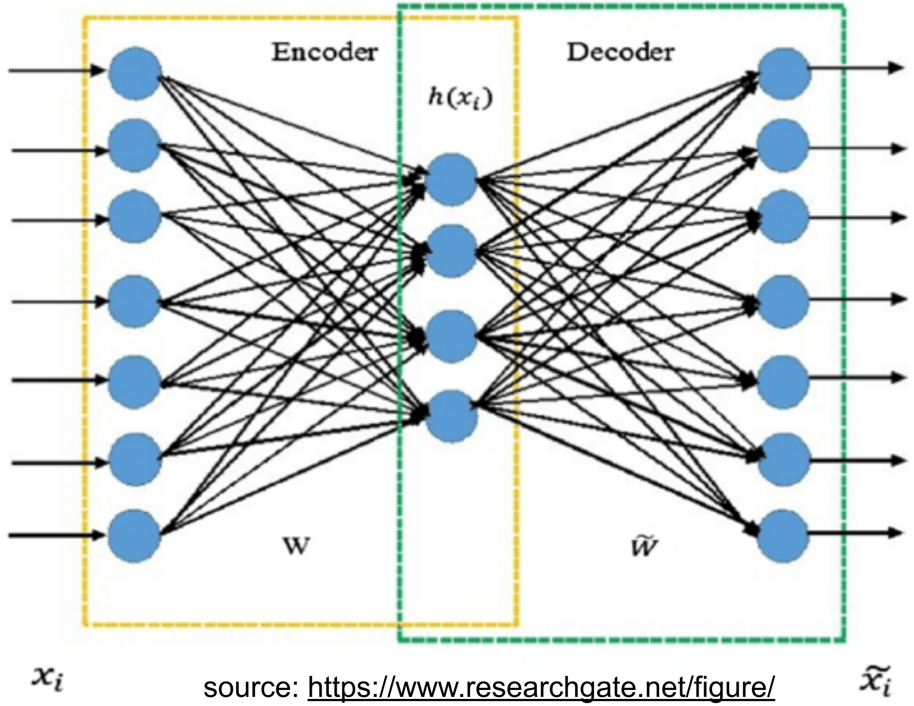


- Generative network (G) learns to create images from random inputs
- Adversarial network (A) distinguishes fake and real images
- Adapt weights of G so that the loss of A is maximised
- Train on original and adversarial examples

Applications in particle physics being explored: e.g. fast simulation of calorimeter showers:
https://arxiv.org/abs/1807.01954
https://cds.cern.ch/record/2746032/files/ATL-SOFT-PUB-2020-006.pdf

Autoencoder

Unsupervised Learning for Anomaly Detection



source: https://www.researchgate.net/figure/Autoencoder-architecture_fig1_318204554

Variational Autoencoder (VAE)

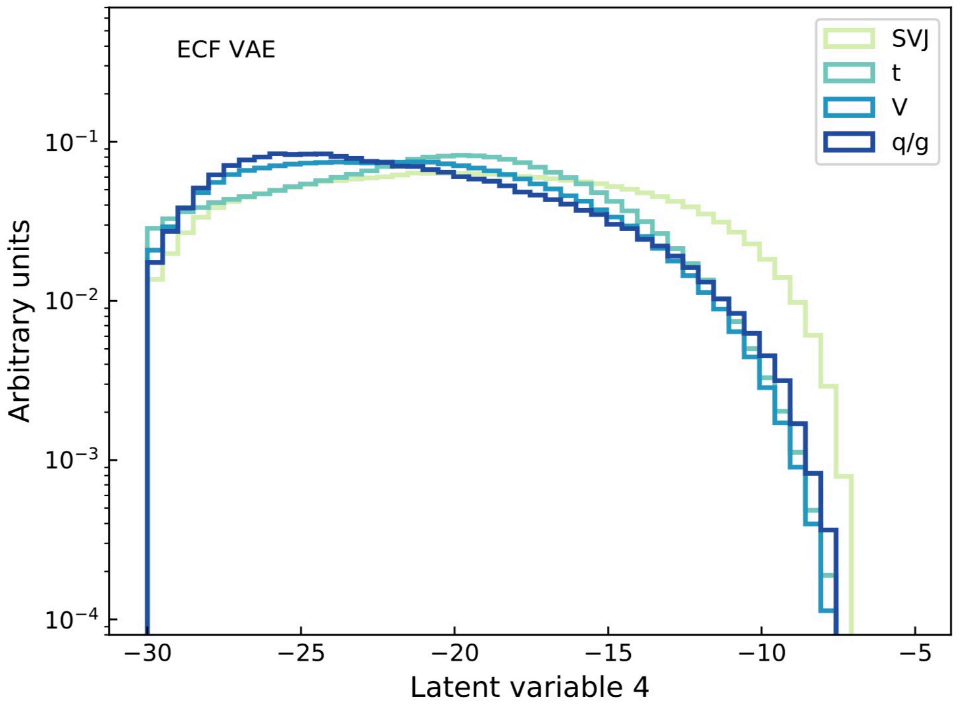


Figure: thanks to Benedikt Maier

- Learn efficient data coding, i.e. a representation of the data with reduced dimensionality
- Target: $\tilde{x_i} = x_i$
- Encoding $h(x_i)$: latent variables, or latent representation

Applications in particle physics:

- search for new physics: e.g. https://arxiv.org/abs/1811.10276
- data quality monitoring
- etc…