Machine Learning

Sergei Gleyzer

PART II

CERN Open Lab Summer Student Lecture

July 27, 2016

Outline

- Machine Learning Recap
- Ensemble Classifiers
- Artificial Neural Networks
- Deep Learning
- Hands-on examples
- Summary

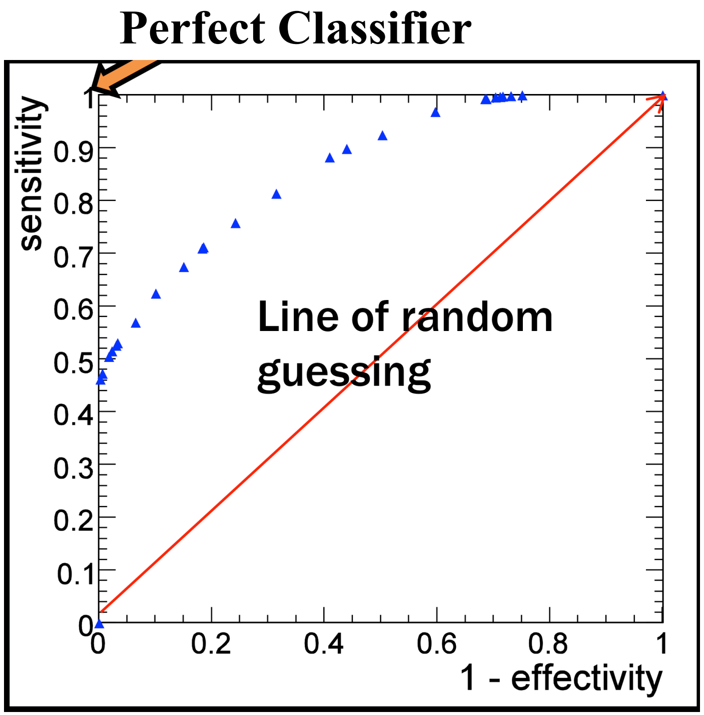
Classifier Performance

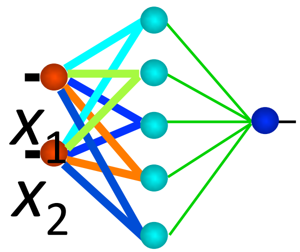
Classifier Performance

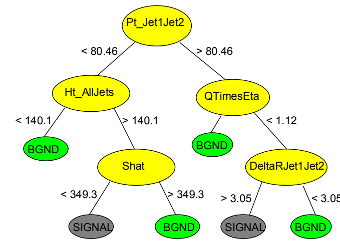
Receiver Operating Characteristic (ROC)

Commonly used metric

Shows the **relationship** between correctly classified positive cases (sensitivity) and incorrectly classified negative cases (1-effectivity)







Ensemble Methods

Ensemble Methods

Suppose you have a **collection** of discriminants $f(x, w_k)$, which, individually, perform only **marginally** better than random guessing.

From such discriminants, **weak learners**, it is possible to build highly effective ones by averaging over them:

Jerome Friedman & Bogdan Popescu (2008)

Boosting

Boosting:

- R. Shapire, 1990
- Turn **weak learners** **to strong** **with weighted** ensemble of iterative learners
  - Adaptation
- Benefits: excellent accuracy

Adaptive Boosting

Adaptive Boosting

Train in stages

- Adaptive weights
  - ADABoost: Freund & Schapire 1997
- **Misclassified** events get a larger weight going into the next training stage
  - Classify with a majority vote from all trees
- **Works** very well to improve classification power of "greedy" decision trees
  - can be used with other classifiers

Adaptive Boosting

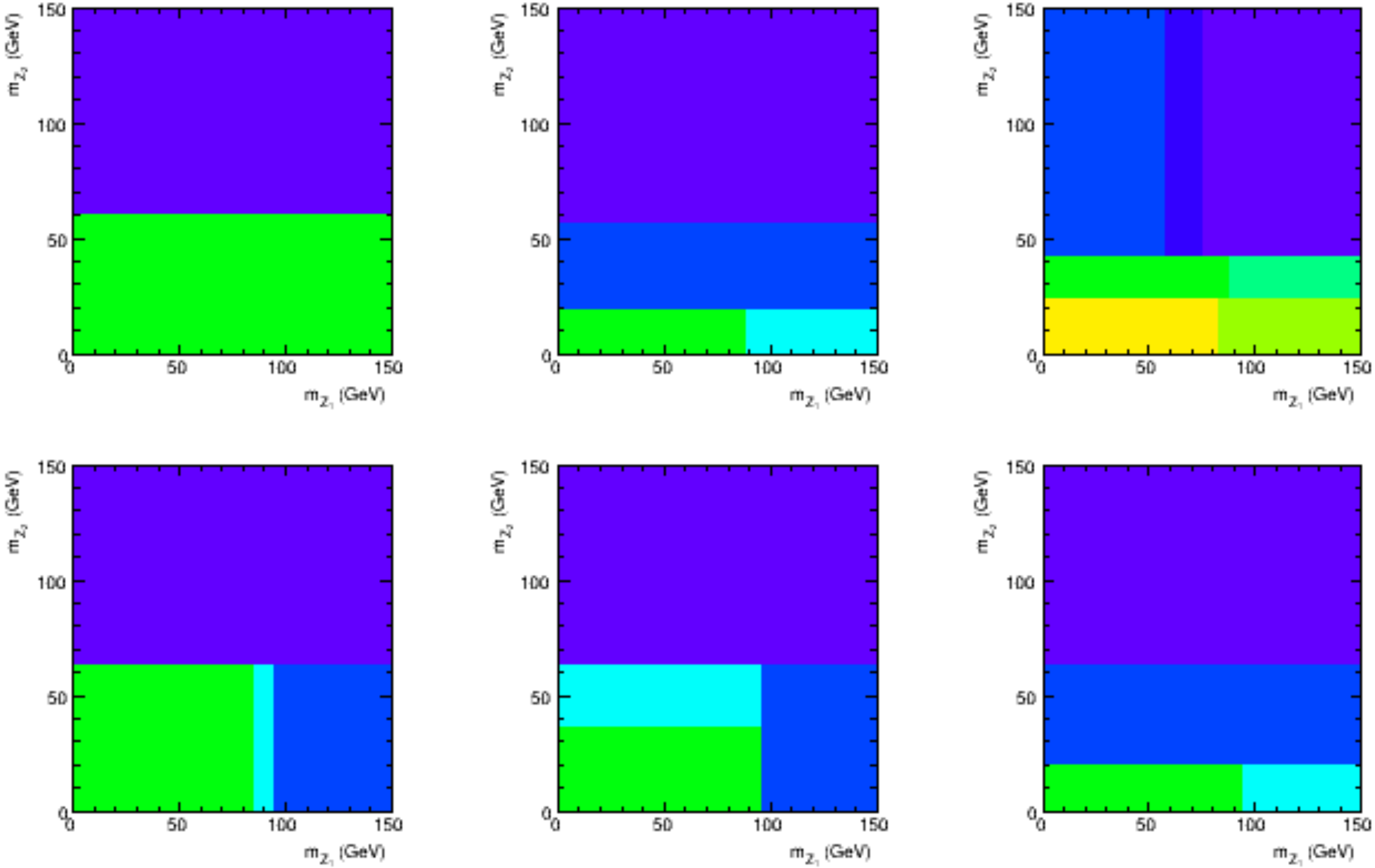
Repeat *K* times:

1. Create a decision tree $f(x, w)$
2. Compute its error rate ε on the ***weighted*** training set
3. Compute $\alpha = \ln(1 - \varepsilon) / \varepsilon$
4. Modify training set: ***increase weight*** of ***incorrectly classified examples*** relative to the weights of those that are correctly classified

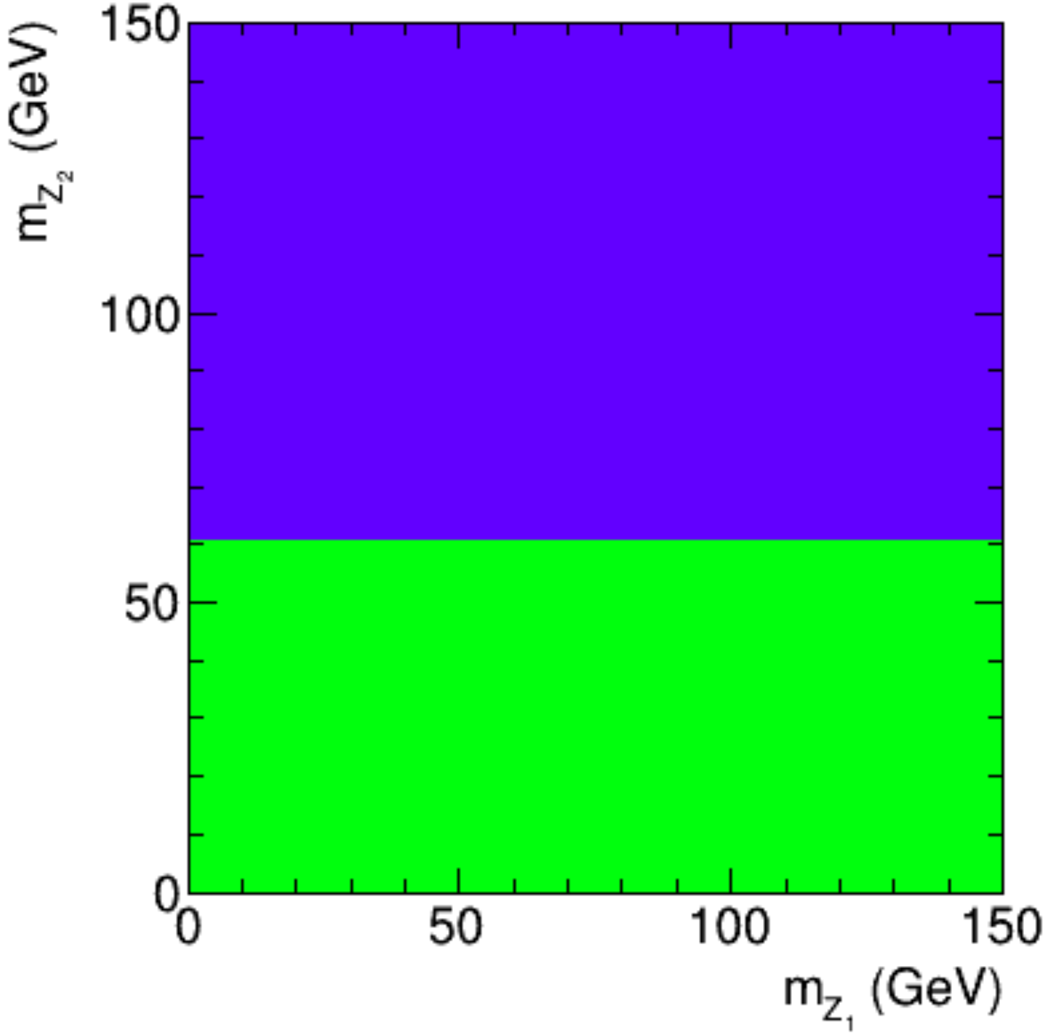
Then compute weighted average $f(x) = \sum \alpha_k \, f(x, w_k)$

Y. Freund and R.E. Schapire.
Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

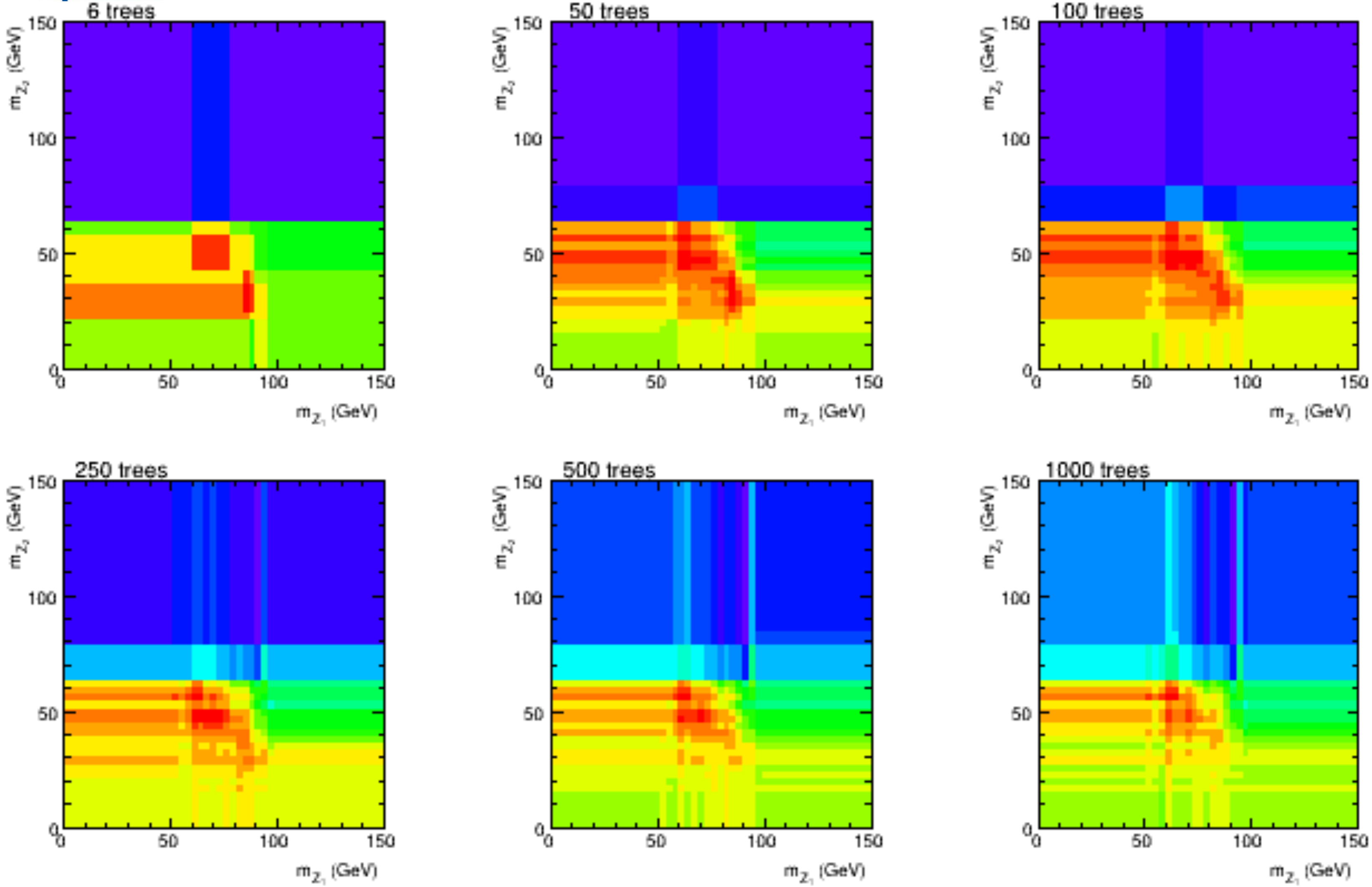
First 6 Decision Trees



First 100 Decision Trees



Averaging over a Forest



Proceed to Tutorial (c5.0)

Part III: Boosting

Hands-On Part III

1. Login to CERNBox: http://cernbox.cern.ch
2. Open Swan: http://swan001.cern.ch
3. Open new terminal
4. Clone the code: git clone https://github.com/iml-wg/c50.git
5. Go to c50 directory: cd c50/

Tutorial Part III

Examples: playing golf, breast-cancer

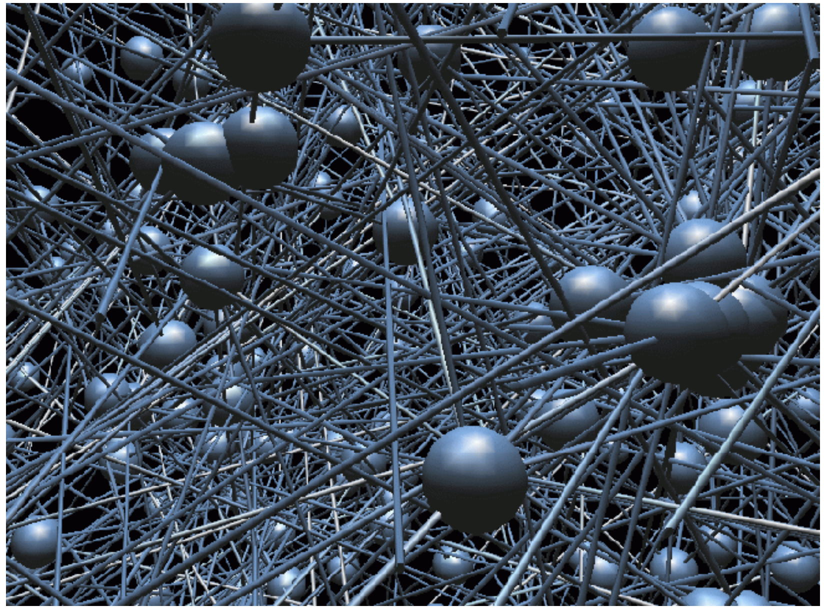
- **Create your first boosted classifiers**
  - **Decision trees**
    - c5.0 –b –f breast-cancer
  - **Rules**
    - c5.0 –b –r –f breast-cancer
  - Look at Training and Testing error rates

Cross Validation

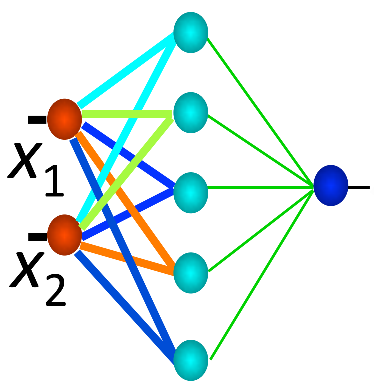
Cross Validation

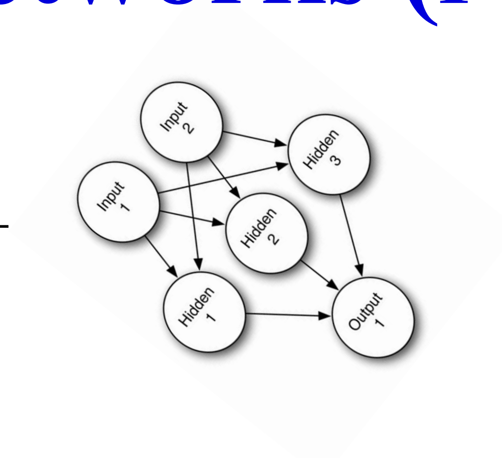
Generalization of train-test split for more accurate classifier performance evaluation

- Randomly **split** dataset into **N equal** partitions
- In each fold of **N-fold** cross-validation
  - Use **N-1** samples to train, leftover to test
  - Repeat **N times**

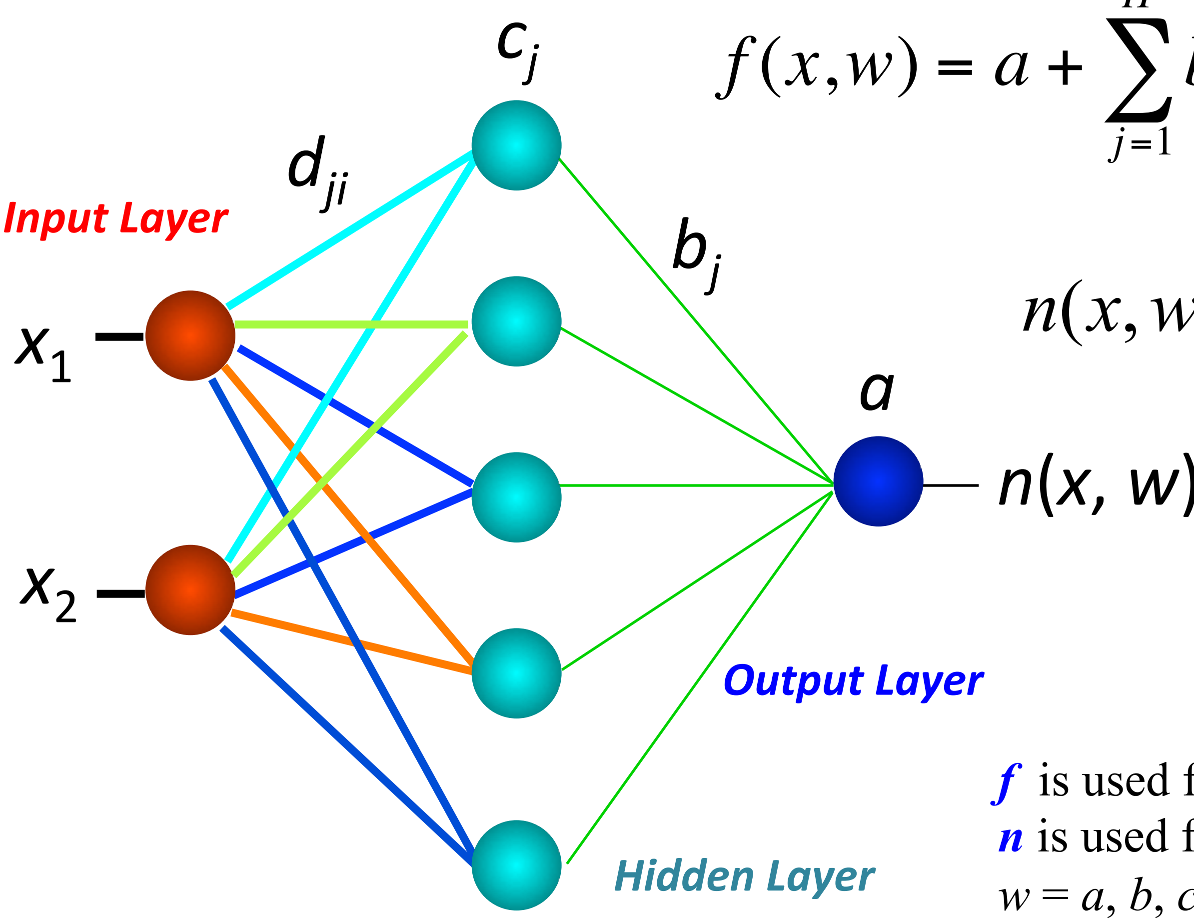


Neural Networks (NN)





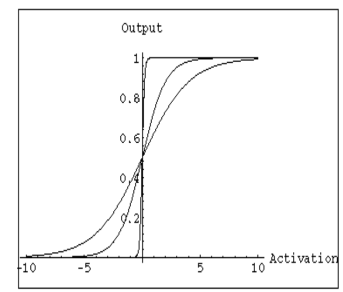
Graphical Representation



$$f(x,w) = a + \sum_{j=1}^{H} b_j \tanh\left[c_j + \sum_{i=1}^{I} d_{ji} x_i\right]$$

$$n(x,w) = \frac{1}{1 + \exp[-f(x,w)]}$$

sigmoid

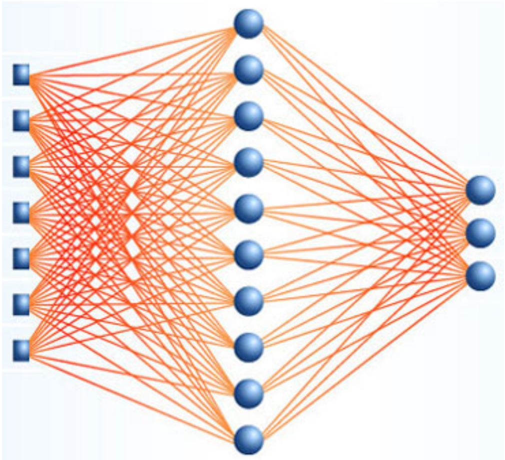


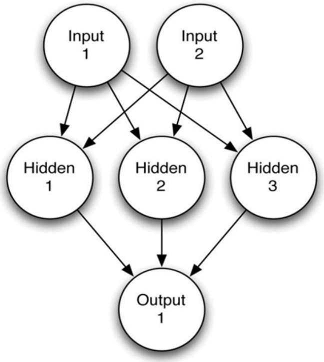
f is used for regression
n is used for classification
$w = a, b, c, d$

Network Weights

Compute optimal network weights with derivatives dE/dw

– Calculate gradients of errors for adjustable weights





Inputs go forward in feed-forward neural networks

Errors go backward! **Back-propagation**

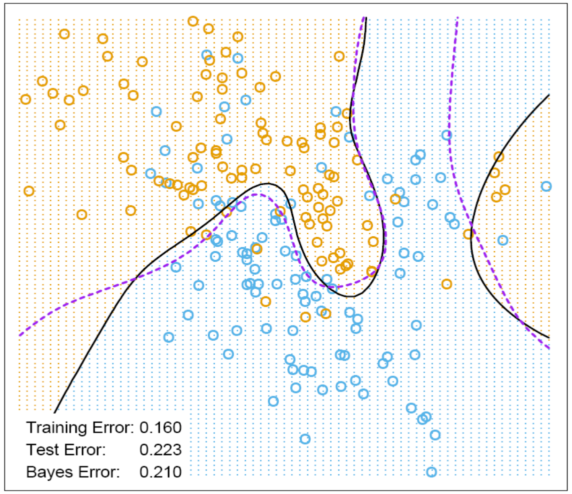
Neural Networks

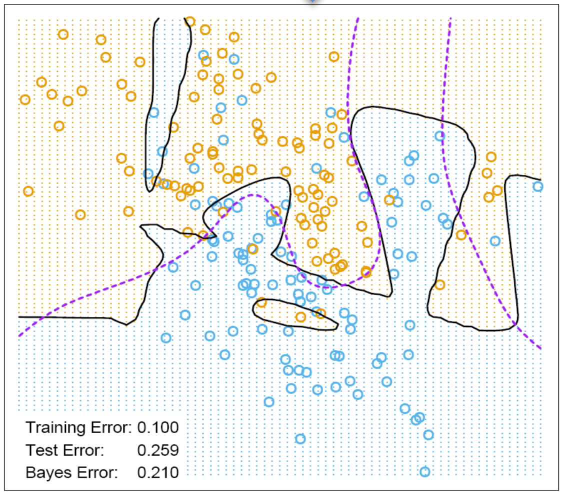
Can approximate any continuous function

Complexity determined by number of hidden layers and hidden nodes/layer

Many types of neural networks!

Watch out for overtraining





Hilbert's 13th Problem

Problem 13: Prove the conjecture

In general, it is ***impossible*** to do the following:

$$f(x_1,\ldots,x_n) = F(\, g_1(x_1),\ldots,\, g_n(x_n)\,)$$

But, in 1957, Kolmogorov ***disproved*** Hilbert's conjecture!

Today, we know that functions of the form

$$f(x_1,\cdots,x_I) = a + \sum_{j=1}^{H} b_j \tanh\left[c_j + \sum_{i=1}^{I} d_{ji} x_i\right]$$

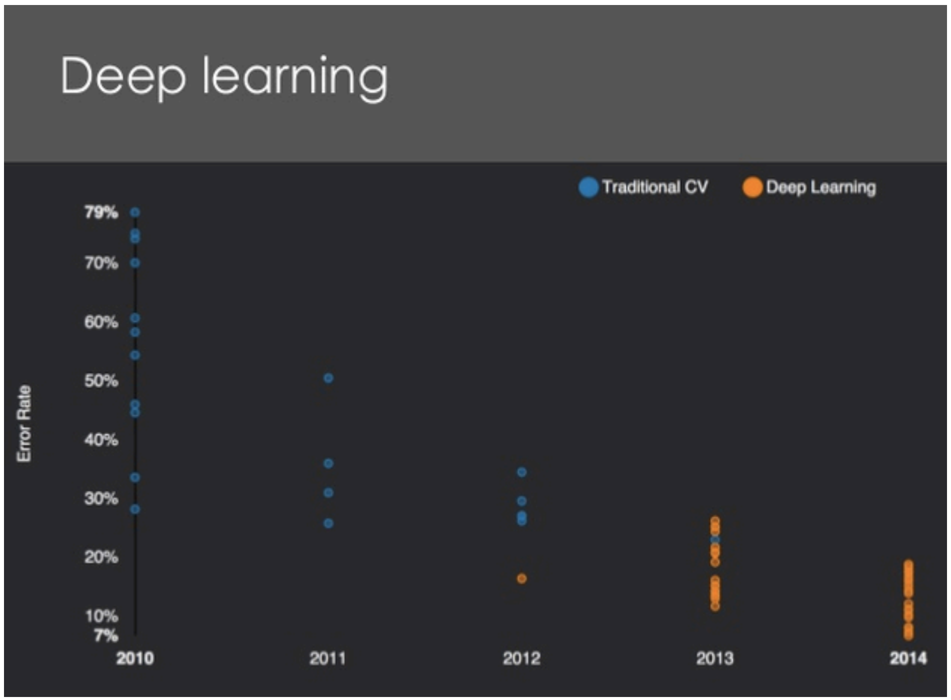
can provide arbitrarily accurate approximations.

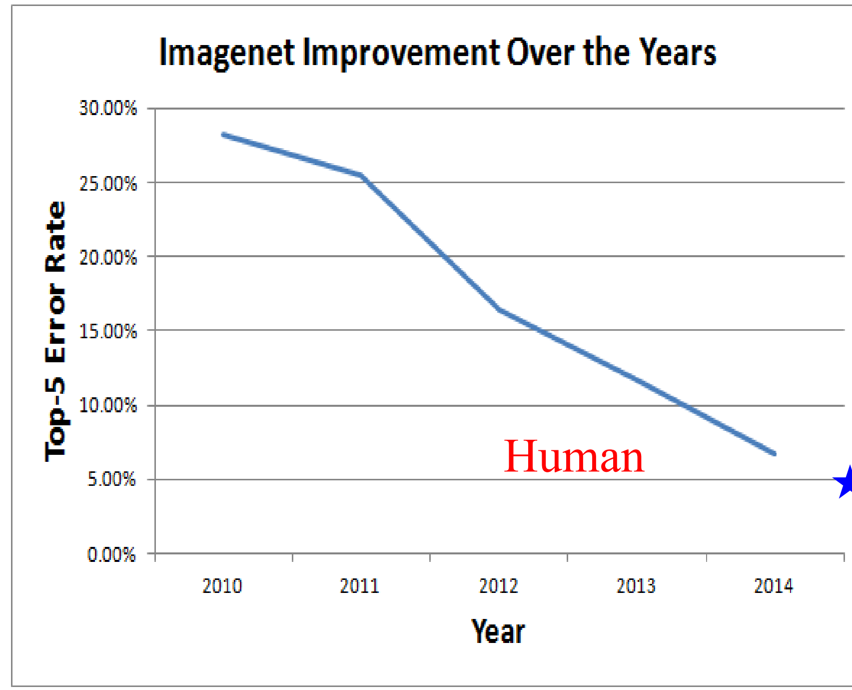
(Hornik, Stinchcombe, and White,

Neural Networks **2**, 359-366 (1989))

Deep Learning

Deep Learning





Computer Vision (CV) Benchmarks

First super-human result in 2015*

* Google/Microsoft 4.9%

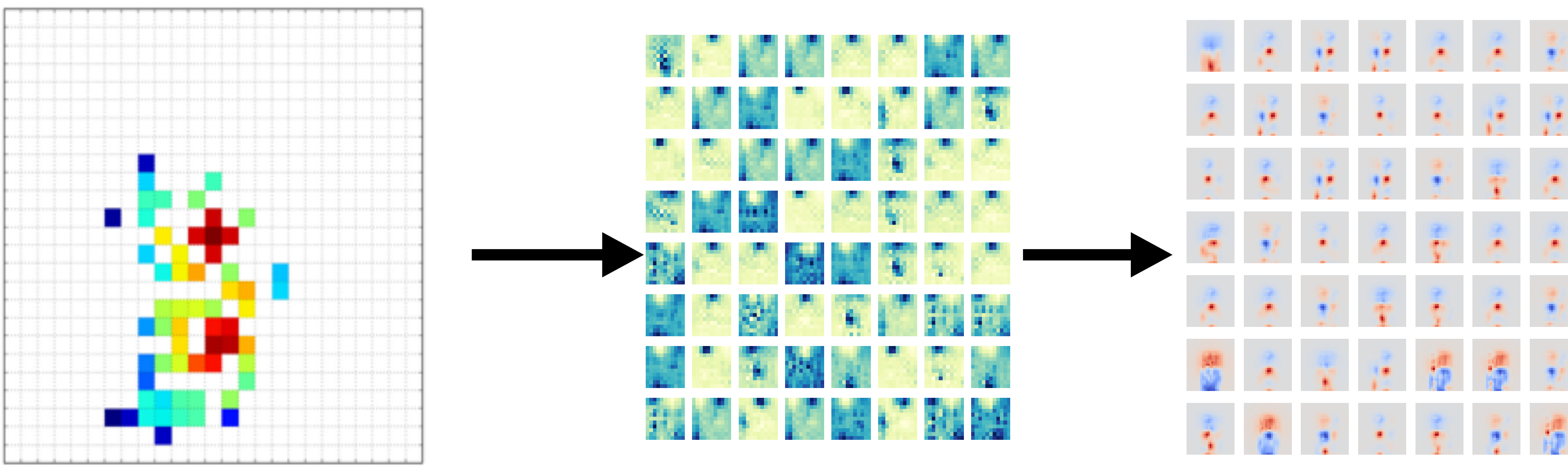
Deep Learning

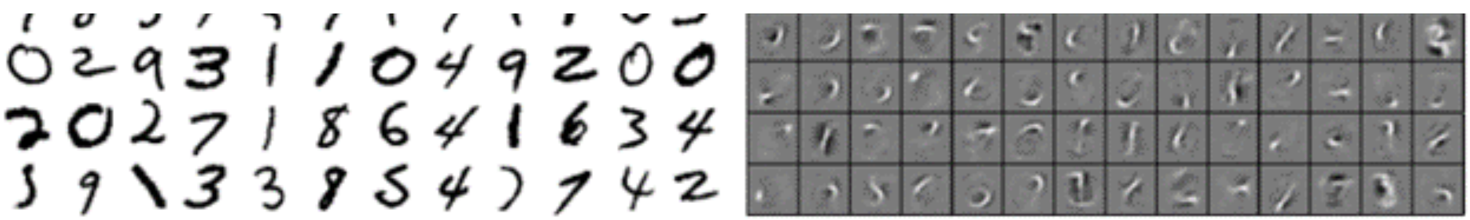
Deep Learning Neural Networks:

- Tremendous performance improvement
  - Training more complex models
    - **Increased** Depth
    - **Enlarged** Width
    - **Feedback**/Convolution
    - **Novel** activation functions
  - Effective strategies against over-fitting
    - Regularization

Deep Learning

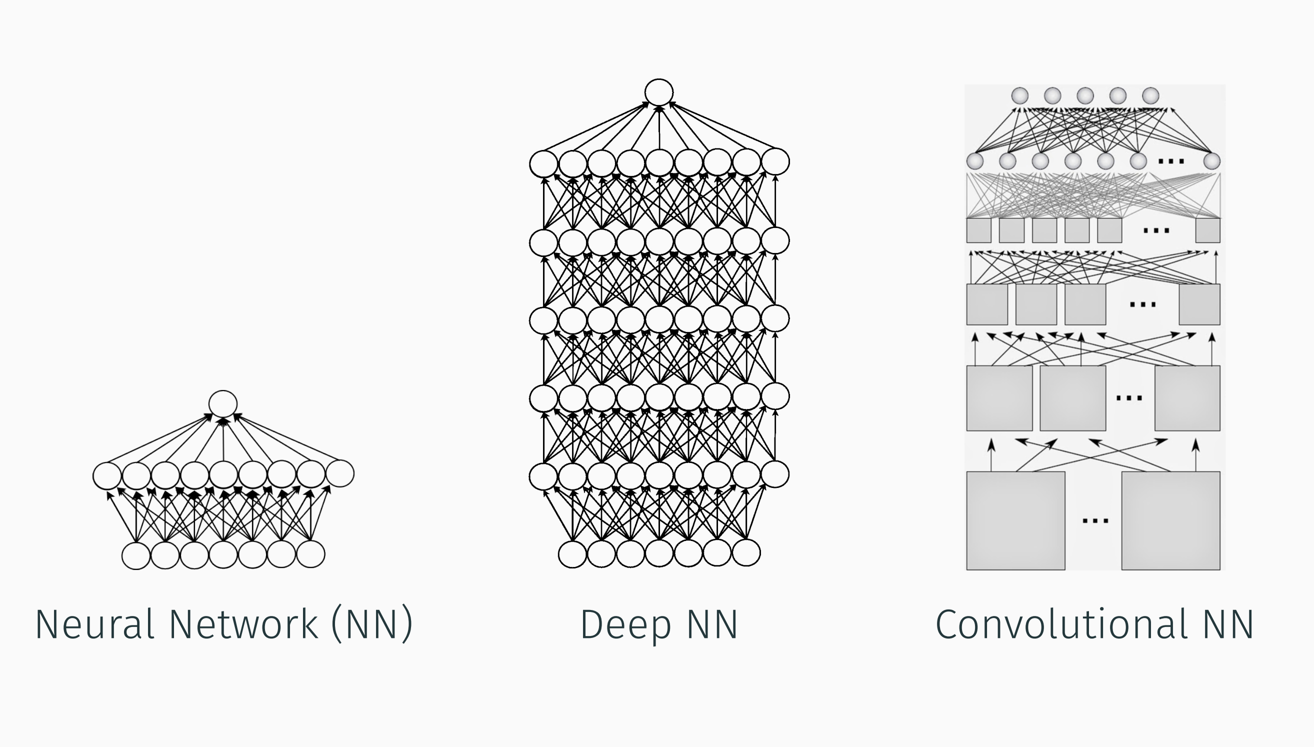
Convolutional Neural Networks:





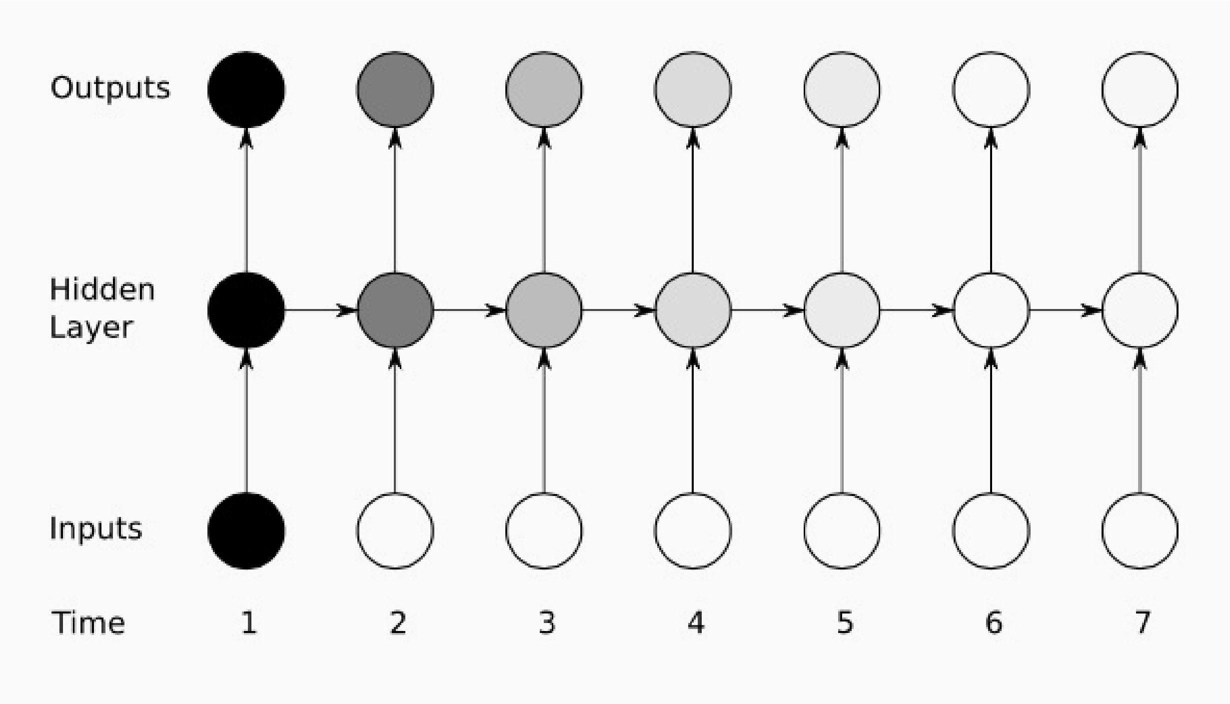
Deep Learning

Convolutional Neural Networks:



Deep Learning

Recurrent Neural Networks:



Cycles

Deep Learning

Higgs Boson Example:

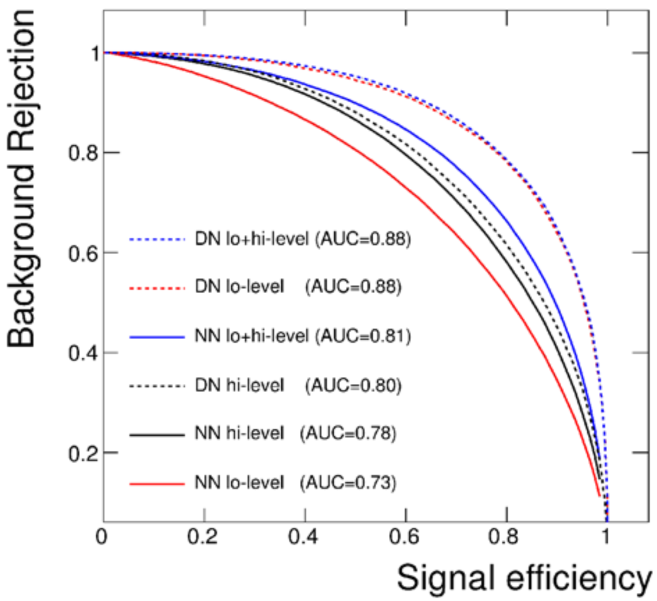
P. Baldi, et. al. 2014

Tuning deep neural network architectures.

Hyper parameters	Choices
Depth	2,3,4,5,6 layers
Hidden units per layer	100,200,300,500
Learning rate	0.01, 0.05
Weight decay	0, 0.00001
Pre-training	none, autoencoder multi-task autoencoder
Input features	low-level, high-level complete set

Best:

- 5 hidden layers
- 300 neurons per layer
- Tanh hidden units, sigmoid output
- No pre-training
- Stochastic gradient descent
- Mini batches of 100
- Exponentially-decreasing learning rate
- Momentum increasing from .5 to .99 over 200 epochs
- Weight decay = 0.00001



8% improvement

Proceed to Tutorials (TMVA)

TMVA

Toolkit for Multivariate Analysis:

- **HEP ML** workhorse
- **Easy** to get started with
- **ROOT** integrated
- In production by **LHC experiments**
  - basic and advanced methods implemented

Hands-On Part IV

1. Open Swan: http://swan.web.cern.ch
2. Go to Gallery then Machine Learning
3. Click on “ribbon” to execute examples directly in Swan
4. Try different examples

Summary

- Many **machine learning** methods available: pick the one that best suits your problem
  - Good **starting** points: **boosted decision trees**, **neural networks**
  - Then: random forests, support vector machines, deep/bayesian neural nets

Resources

Literature

G. James, et al. "Introduction to Statistical Learning" Springer 2013

C.M. Bishop "Pattern Recognition and Machine Learning" Springer 2006

J. R. Quinlan "C4.5: Programs for Machine Learning" Morgan Kaufmann 1992

IML

- **Website:** http://iml.cern.ch
- **Next meeting Aug 25** (monthly) https://indico.cern.ch/event/548789/

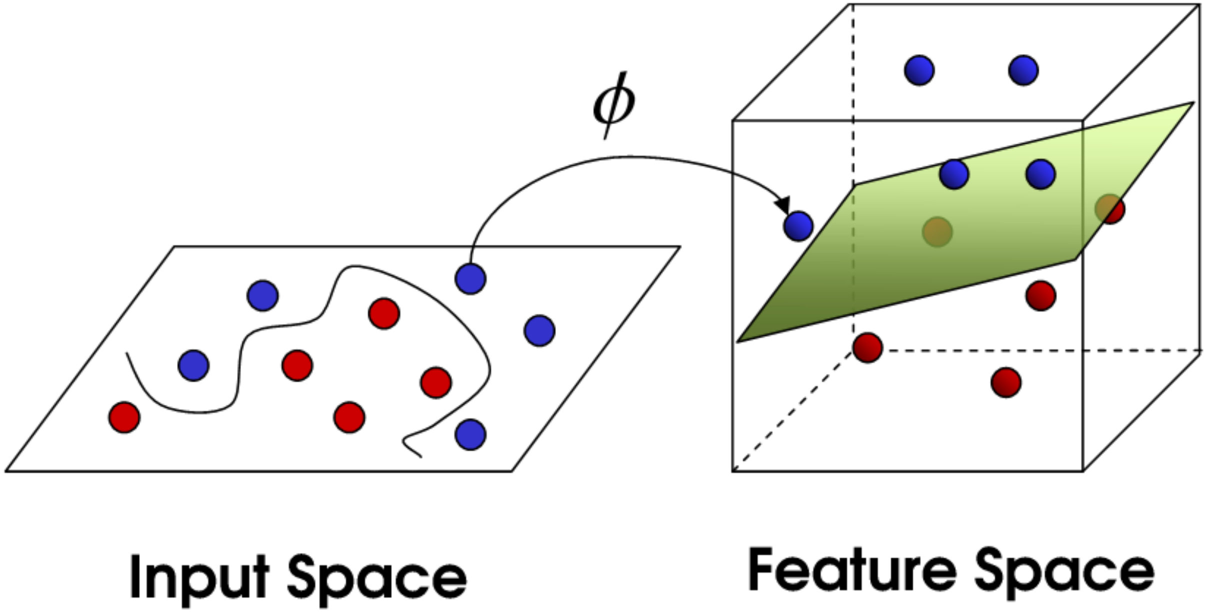
  Forum/Mailing-list/LPCC Group

    - https://simba3.web.cern.ch/simba3/SelfSubscription.aspx?groupName=lhc-machinelearning-wg (cern e-group)

  – Please join if you are interested in ML topics

Additional Material

Support Vector Machines



Support Vector Machines

Generalization of the Fisher discriminant

– Boser, Guyon and Vapnik, 1992

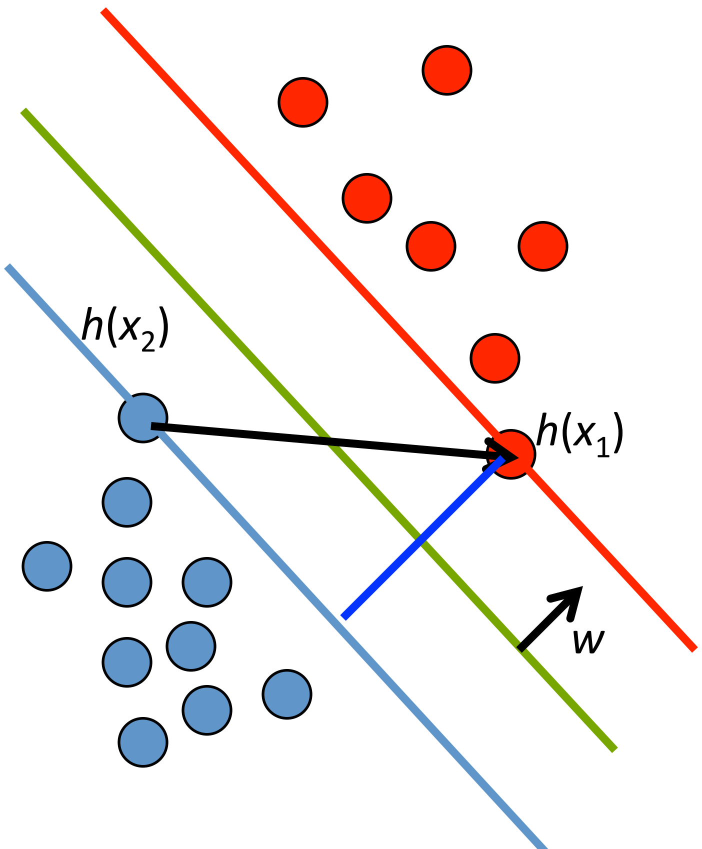
Basic Idea

Data that are non-separable in d-dimensions may be better separated if mapped into a space of higher (usually, infinite) dimension

$$h : \Re^{d} \rightarrow \Re^{\infty}$$

As in the Fisher discriminant, a hyper-plane is used to partition the high dimensional space $f(x) = w \cdot h(x) + c$

Support Vector Machines



Consider ***separable*** data in the high dimensional space

green plane: $w.h(x) + c = 0$

red plane: $w.h(x_1) + c = +1$

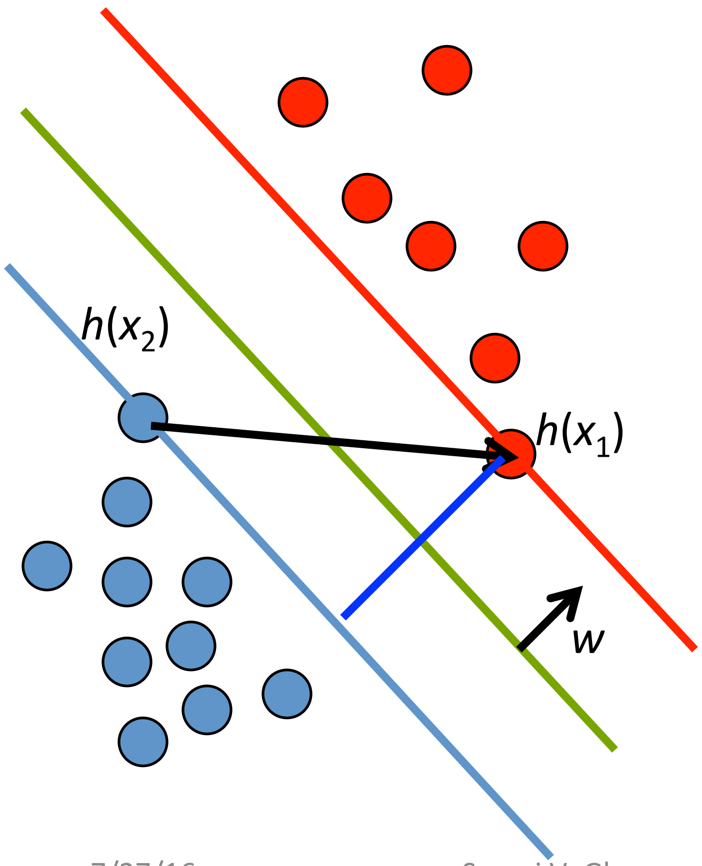
blue plane: $w.h(x_2) + c = -1$

subtract **blue** from **red**

$$w.[h(x_1) - h(x_2)] = 2$$

and normalize the high dimensional vector w $\quad \hat{\mathrm{w}}.[h(\mathrm{x}_1) - h(x_2)] = 2/||w||$

Support Vector Machines



$m = \hat{w}.[h(x_1) - h(x_2)]$, the distance between the red and blue planes, is called the margin. The best separation occurs when the margin is as large as possible.

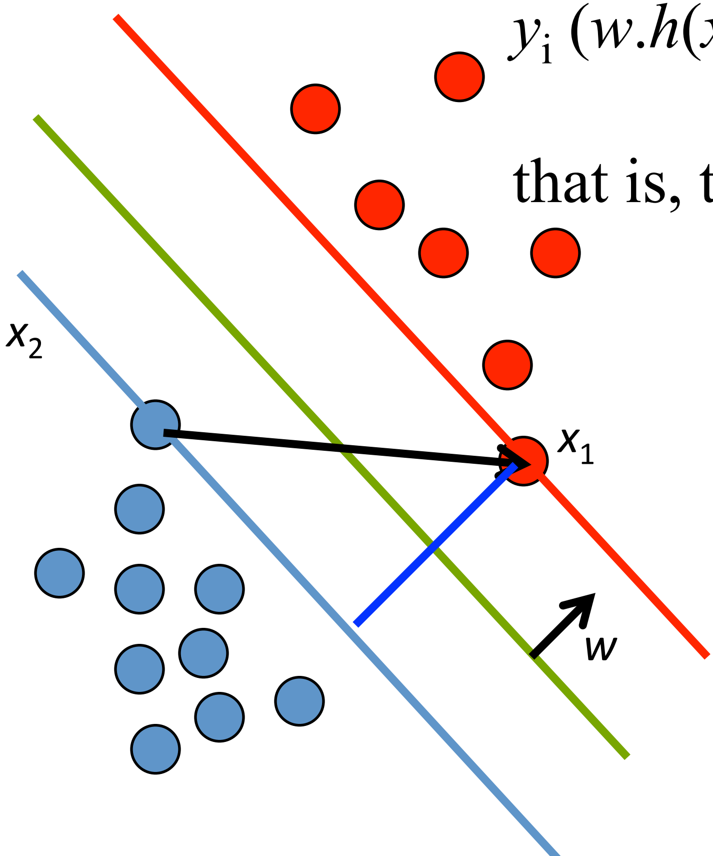
Note: because $m \sim 1/||w||$, maximizing the margin is equivalent to minimizing

$$||w||^2$$

Support Vector Machines

Label the **red** dots $y = +1$ and the **blue** dots $y = -1$. The task is to minimize $||w||^2$ subject to the constraint

$$y_i\,(w.h(x_i) + c) \geq 1, \qquad i = 1 \ldots N$$



that is, the task is to minimize

$$L(w,c,\alpha) = \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(w \cdot h(x_i) + c \right) - 1 \right]$$

where the $\alpha > 0$ are Lagrange multipliers

Support Vector Machines

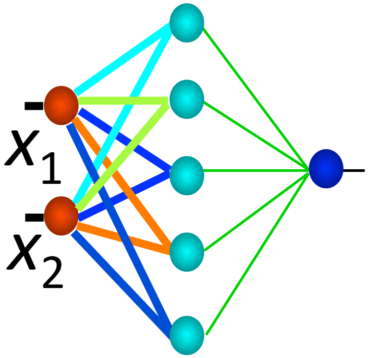
When ***L*(w,c,α)** is minimized with respect to w and c, the function ***L*(w,c,α)** can be transformed to

$$E(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j h(x_i) \cdot h(x_j)$$

At the minimum of $E(\alpha)$, the only non-zero coefficients α are those corresponding to points *on* the **red** and **blue** planes: the so-called **support vectors**. The key idea is to replace the scalar product $h(x_i).h(x_j)$ between two vectors of infinitely many dimensions by a **kernel function** $K(x_i, x_j)$.

- The (unsolved) problem is how to choose the correct kernel for a given problem?

Bayesian Neural Networks



Bayesian Neural Networks

Given: $p(w \mid T) = p(T \mid w)\, p(w) / p(T)$

over the parameter space of the functions

$$n(\boldsymbol{x}, w) = 1 / [1+\exp(-f(\boldsymbol{x}, w))]$$

can estimate $p(s \mid \boldsymbol{x})$ as follows

$$p(s \mid \boldsymbol{x}) \sim n(\boldsymbol{x}) = \int n(\boldsymbol{x}, w)\, p(w \mid T)\, dw$$

$n(\boldsymbol{x})$ is called a **Bayesian Neural Network** (BNN)

Bayesian Neural Networks

Generate Sample:

N points $\{w\}$ from $p(w \mid T)$ using a Markov chain Monte Carlo (MCMC) technique and

average over the last M points

$$n(\boldsymbol{x}) = \int n(\boldsymbol{x}, w)\, p(w \mid T)\, dw$$

$$\sim \sum n(\boldsymbol{x}, w_i) \,/\, M$$

Genetic Algorithms

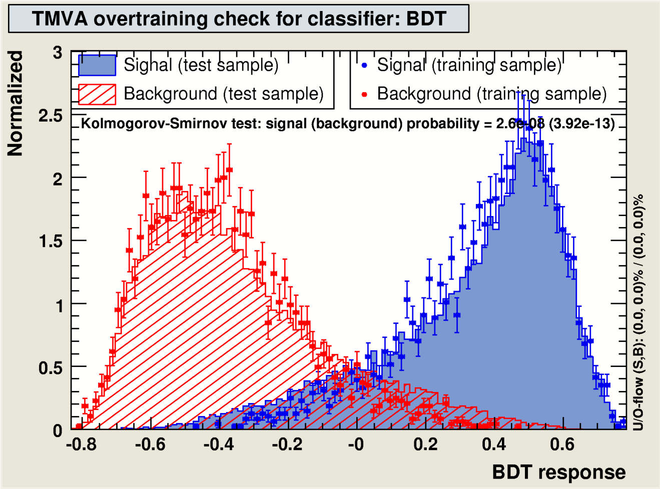
Central idea: adaptation. Inspired by evolutionary biology concepts of mutation, selection, cross-over (recombination) J.H. Holand, 1975

- Begin with a **large population** of random solutions
  - Evaluate each one
  - **Fitness function** (some form of $S/\sqrt{B}$)
  - Keep the **best subset**
    - Use it to build new solutions
    - Allow **mutation**, cross-over
    - Optimize over number of epochs/cycles

Over-Training

Over-training or over-fitting sometimes occurs when too many parameters for data size

- **Diagnose with**
  - Divergent training -testing error slopes
  - Kolmogorov-Smirnov tests of classifier output



- **Treat with**
  - Reduce number of parameters
  - Prune decision trees

Function Estimation

Regression

Function Estimation

Comet Problem by Gauss (1805)

Approximate trajectory of a comet from observations

Approach: minimize difference between measurement and predictions in a systematic fashion

Vary regression model parameters

Function Estimation

Machine Learning:

From **classification** to **regression:**

- modify the evaluation criteria used in the learning algorithm
  - from maximum **separation** gain
  - to minimal **variance**

Function Estimation

Inputs: Training examples {<$x^{(i)}$, $y^{(i)}$>} of unknown function f. $x^{(i)}$,$y^{(i)}$:

Output: hypothesis h that best approximates target function f (for example energy measured by the detector)