CS6510
Applied Machine Learning

Classifiers-2

3 Sep 2016

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

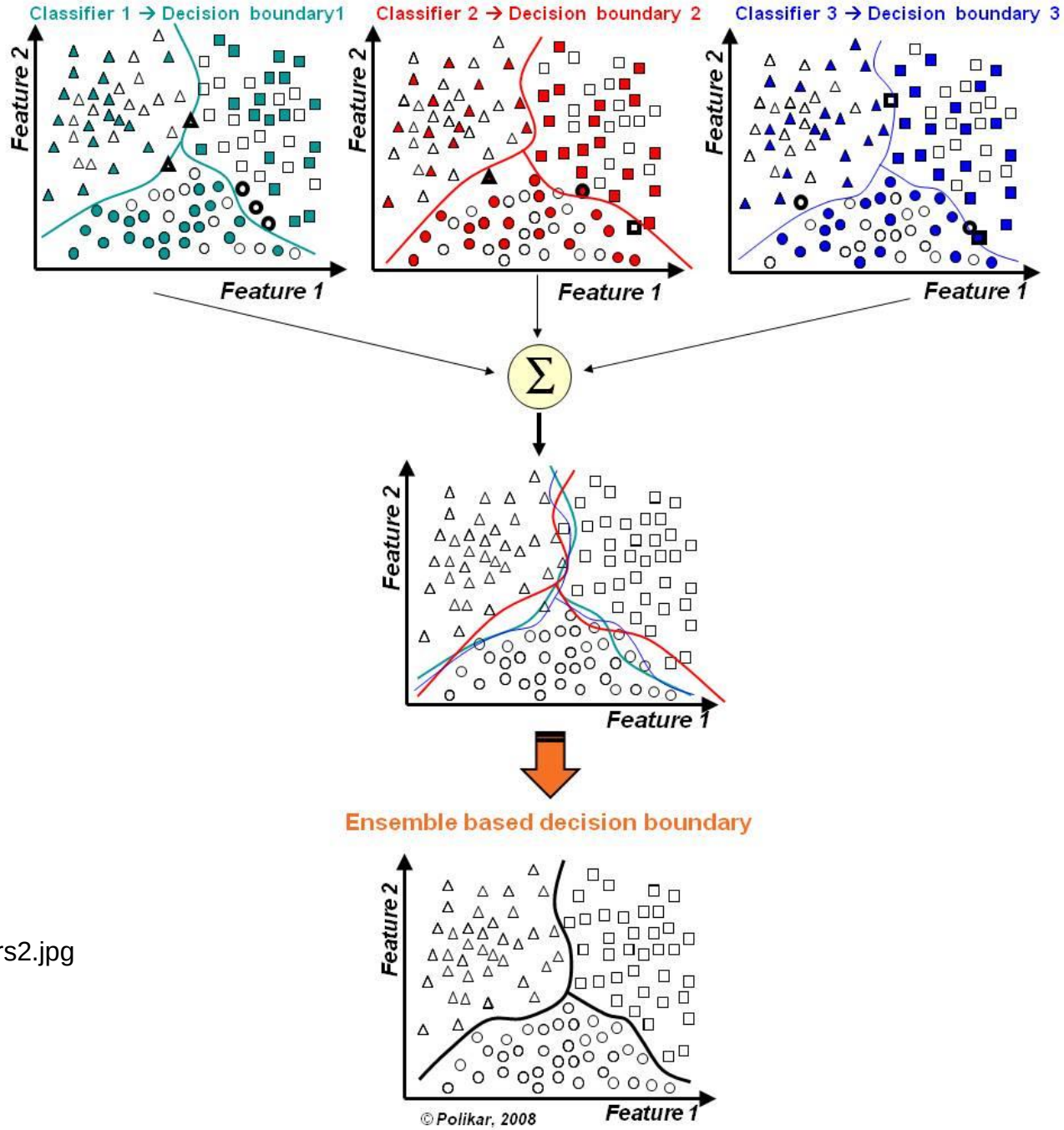
Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

Ensemble Learning

- Ensemble classification combines multiple classifiers to improve accuracy
- Use multiple methods to obtain better performance than any of the individual method
- Can combine outputs

Ensemble Learning

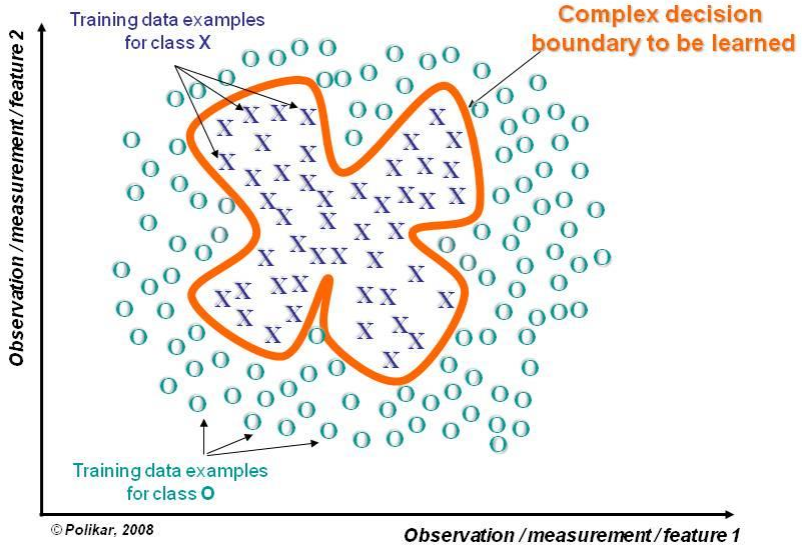


http://www.scholarpedia.org/wiki/images/8/82/Combining_classifiers2.jpg

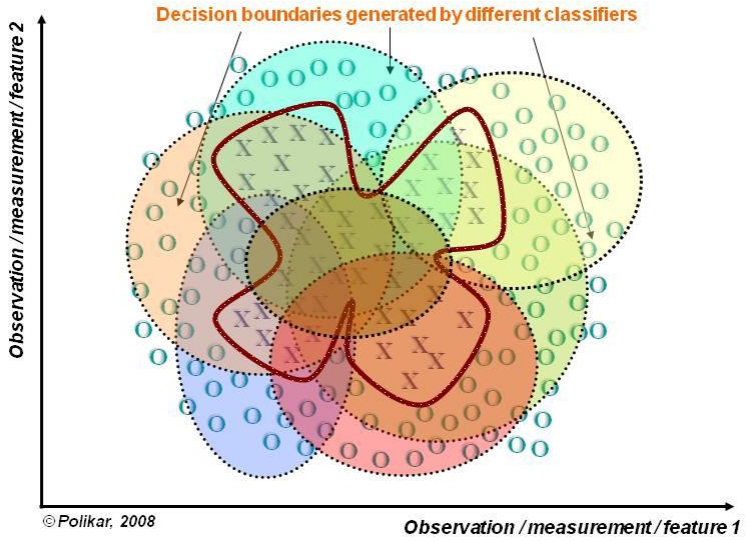
Advantages

- Large datasets
  - May be too large for training single classifier
  - Can use different subsets of data to train multiple classifiers
- Small datasets
  - Can handle them with bootstrapping (random sampling)
- Some problems too complicated to solve with a single classifier
  - Eg. Complex separation between classes

Divide and Conquer



http://www.scholarpedia.org/article/Image:Figure1a.jpg



http://www.scholarpedia.org/wiki/images/a/ab/Figure1b.jpg

Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

Bagging

- **Problem:** we only have one dataset.
- **Solution:** generate new ones of size n by bootstrapping, i.e. sampling it with replacement
- Bagging works because it reduces variance by voting/averaging
- Usually, the more classifiers the better
- Some candidates:
  - Decision tree, decision stump, regression tree, linear regression, SVMs

Bagging: When does it work?

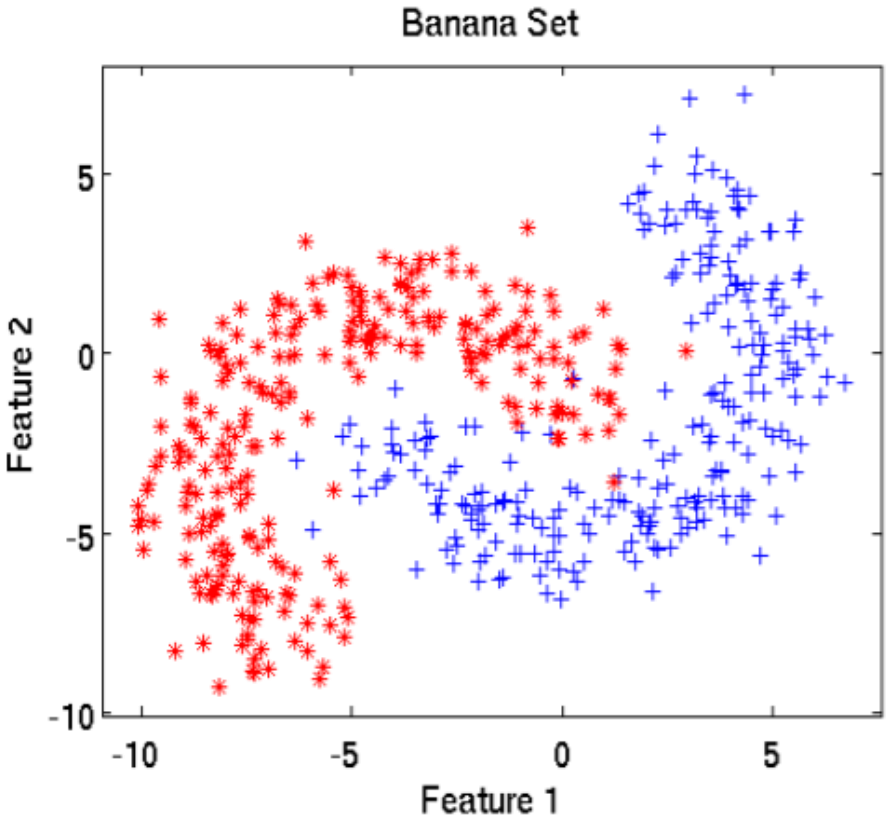
- Can help a lot if data is noisy
- If learning algorithm is unstable, i.e. if small changes to the training set cause large changes in the learned classifier
- In some pathological hypothetical situations, the overall error might increase

Bagging: Why does it work?

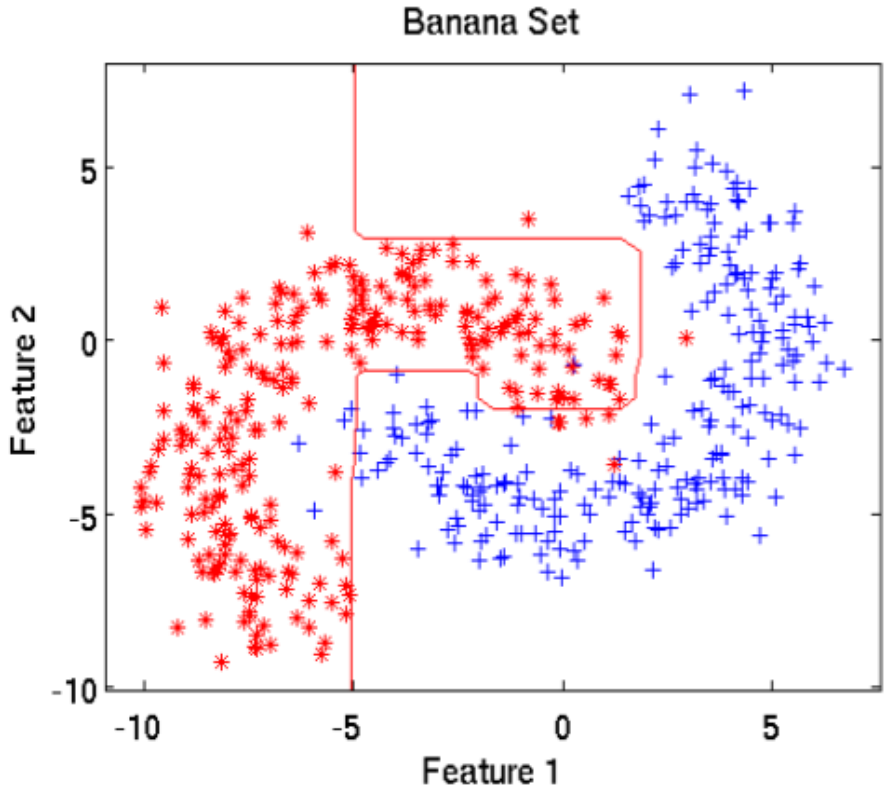
- Let $S = \{(x_i, y_i), i = 1...N\}$ be the set of training dataset
- Let $\{S_k\}$ be a sequence of training sets containing a sub-set of
- Let P be the underlying distribution of S.
- Bagging replaces the prediction of the model with the majority of the predictions given by the classifiers S

$$\varphi(x, P) = E_S(\varphi(x, S_k))$$

Bagging: Illustration

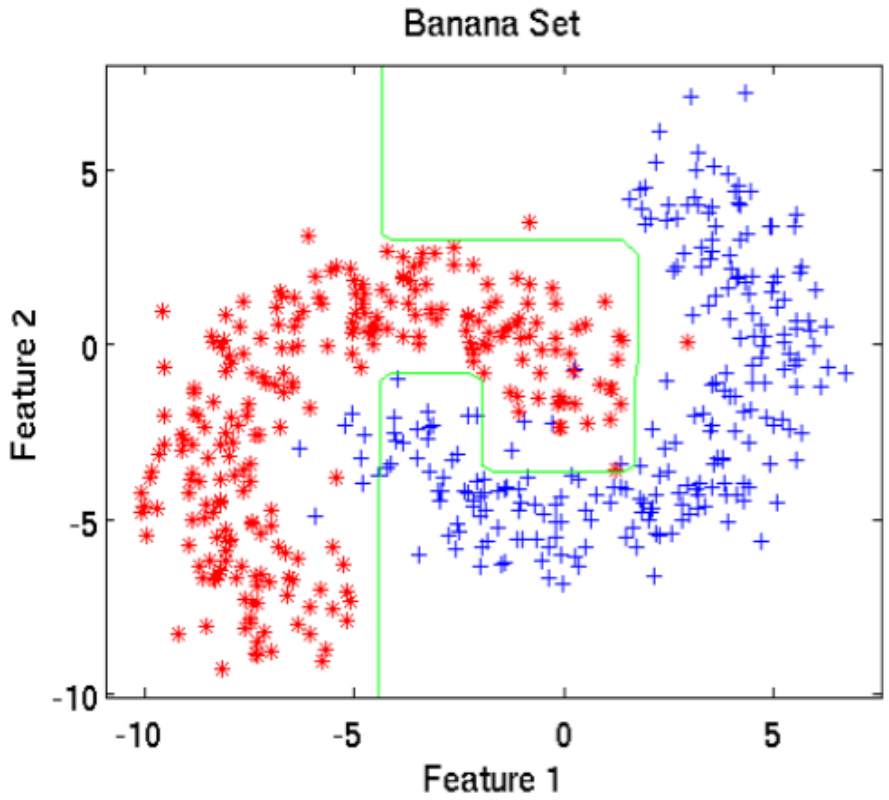


Training data

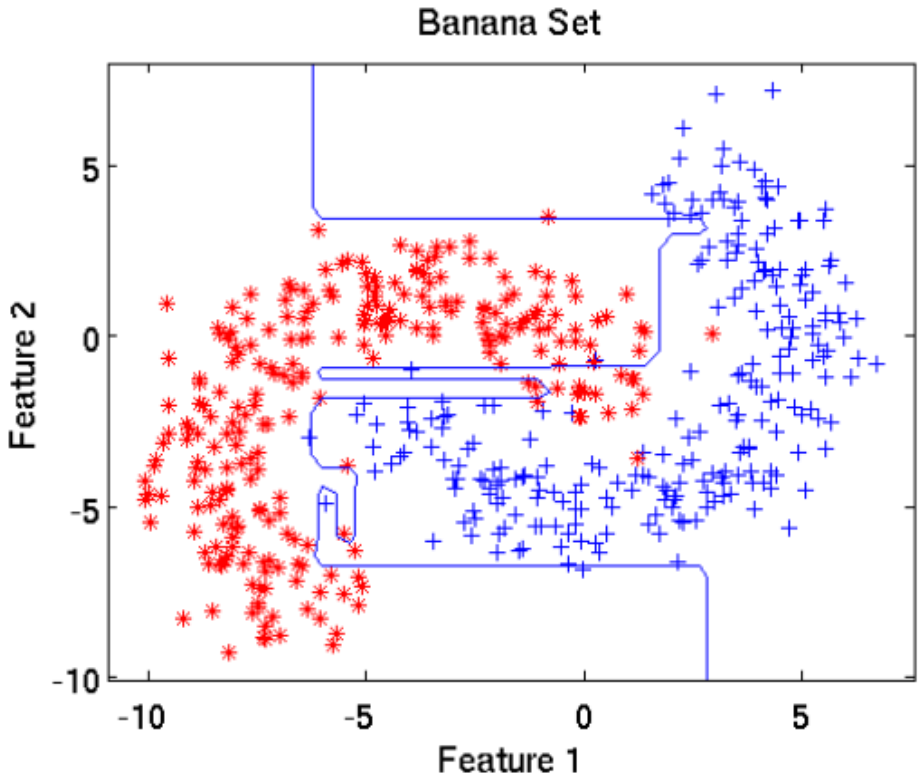


Decision boundary produced by one tree

Bagging: Illustration

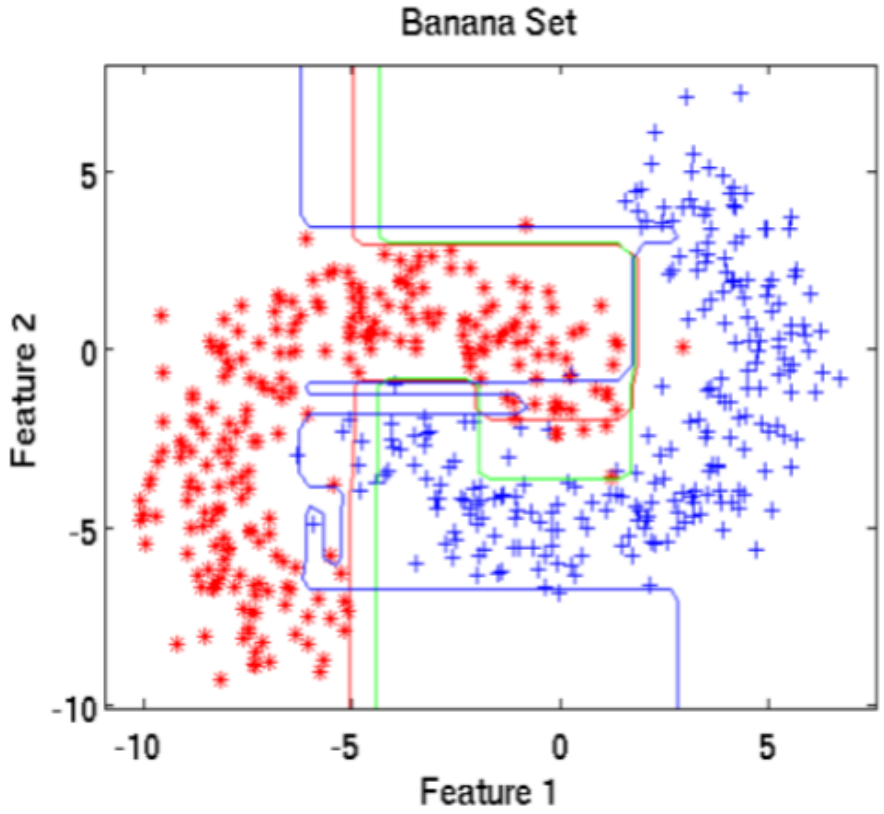


Decision boundary produced by a second tree

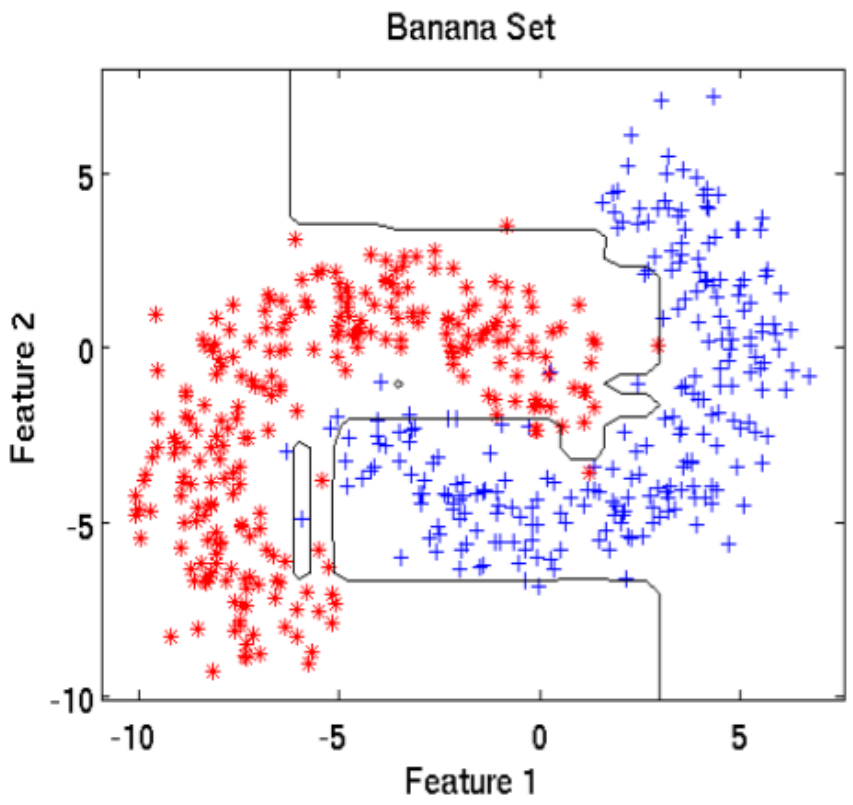


Decision boundary produced by a third tree

Bagging: Illustration



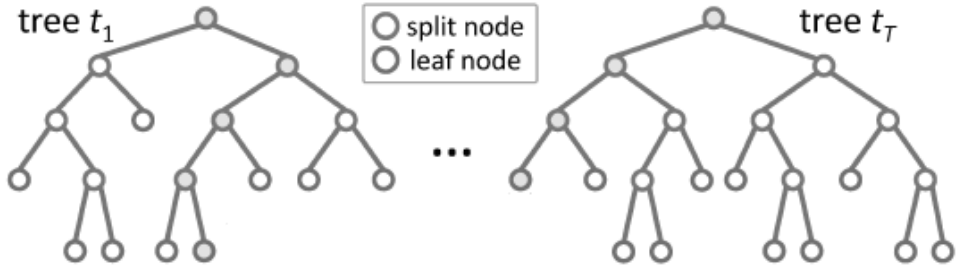
Three trees and final boundary overlaid



Final result from bagging all trees.

Random Forests

- Random forests (RF) are a combination of tree predictors
  - Variant of bagging, proposed by Breiman in 2001
- Extremely successful, especially on Kaggle challenges
- Each tree depends on the values of a random vector sampled independently
- The generalization error depends on the strength of the individual trees and the correlation between them
- Using a random selection of features yields results robust w.r.t. noise



Random Forests: Algorithm

- Given a training set S
- For i = 1 to k do:
- Build subset Si by sampling with replacement from S
- Learn tree Ti from Si
- At each node:
- Choose best split from random subset of F features
- Each tree grows to the largest extend, and no pruning
- Make predictions according to majority vote of the set of k trees.

Features of Random Forests

- One of the best in the business today
- It runs efficiently on large data bases
- It can handle thousands of input variables without variable deletion
- It gives estimates of what variables are important in the classification
- Does not overfit by design
- The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them

Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

Boosting

- Strong Learner →Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*
- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is more accurate than random guessing

Can a set of **weak learners** create a single **strong learner** ?

Example: Adaboost

- AdaBoost is an algorithm for constructing a "strong" classifier as linear combination of "simple" "weak" classifier

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

- Final classification based on weighted vote of weak classifiers

Discrete Adaboost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialise $D_1(i) = \frac{1}{m}$.

For $t = 1, \ldots, T$:

- Find the classifier $h_t : X \to \{-1, +1\}$ that minimizes the error with respect to the distribution D_t:

  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$, where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$

- Prerequisite: $\varepsilon_t < 0.5$, otherwise stop.
- Choose $\alpha_t \in \mathbf{R}$, typically $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ where ε_t is the weighted error rate of classifier h_t.
- Update:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalisation factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Each training sample has a weight, which determines the probability of being selected for training the component classifier

Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{exp(-y_i \sum_{q=1}^{t} \alpha_q h_q(x_i))}{m \prod_{q=1}^{t} Z_q}$$

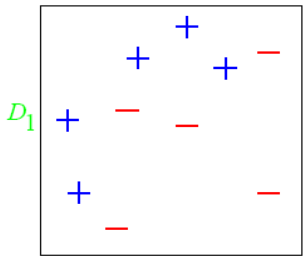
$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

y * h(x) = 1

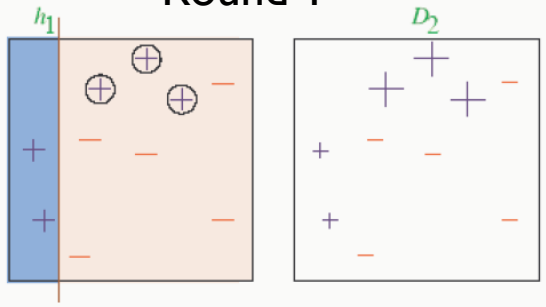
y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

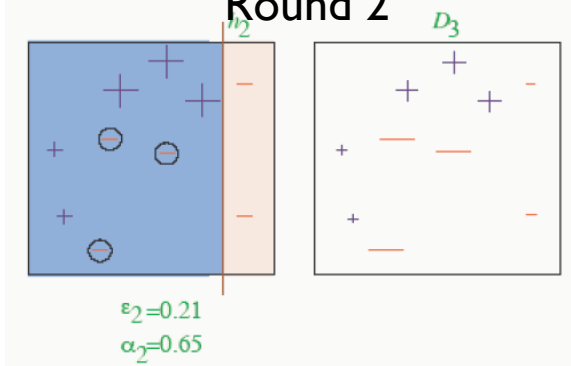
Simple Example



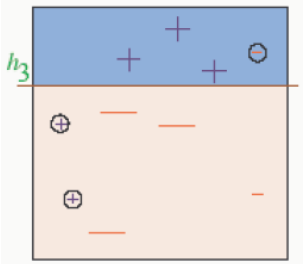
Round 1



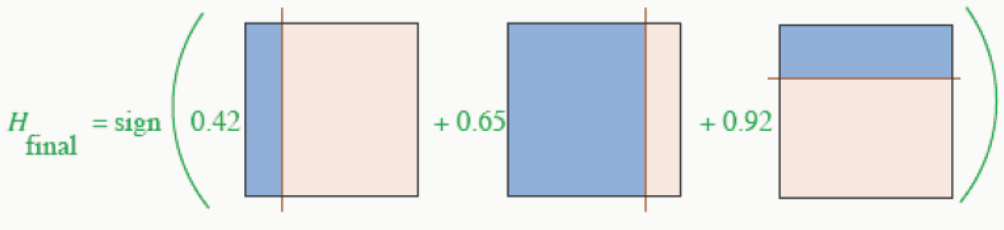
Round 2

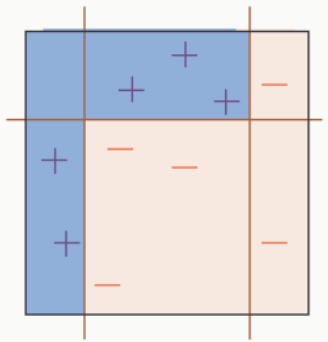


Round 3



Final Hypothesis

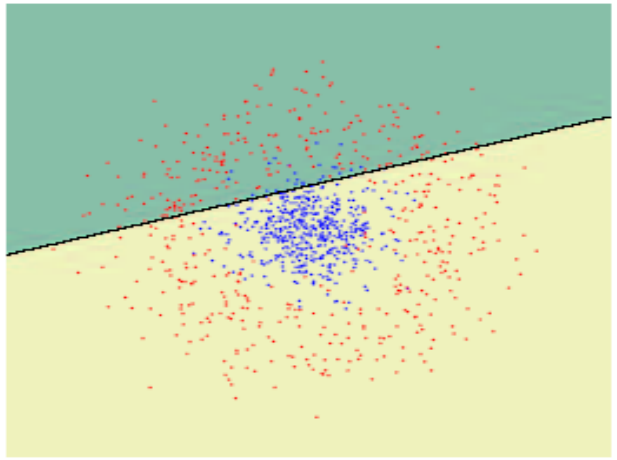


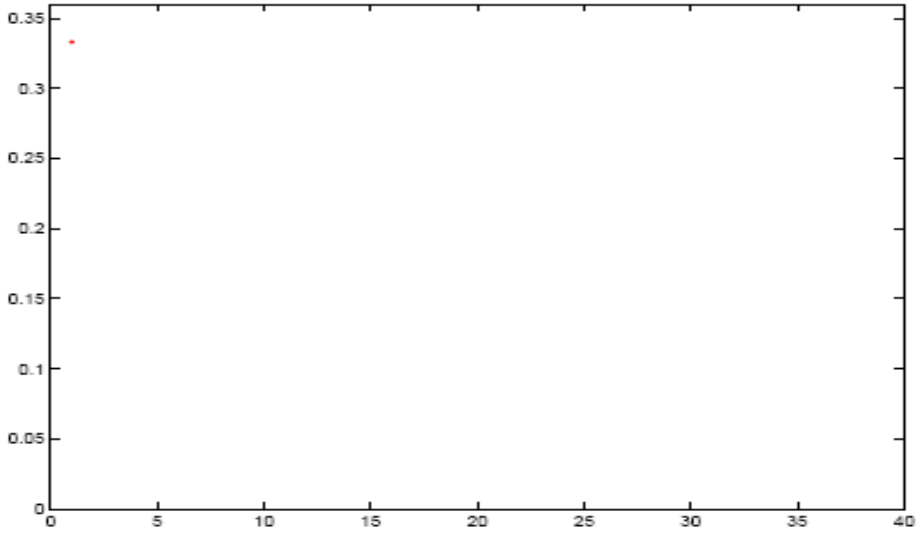


Source: Singer and Lewis Tutorial

Illustration

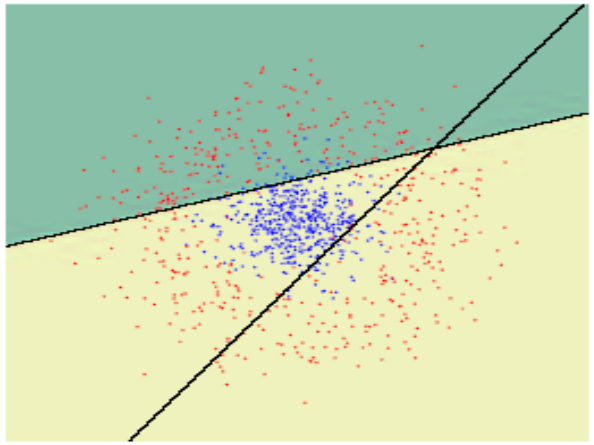
$t = 1$

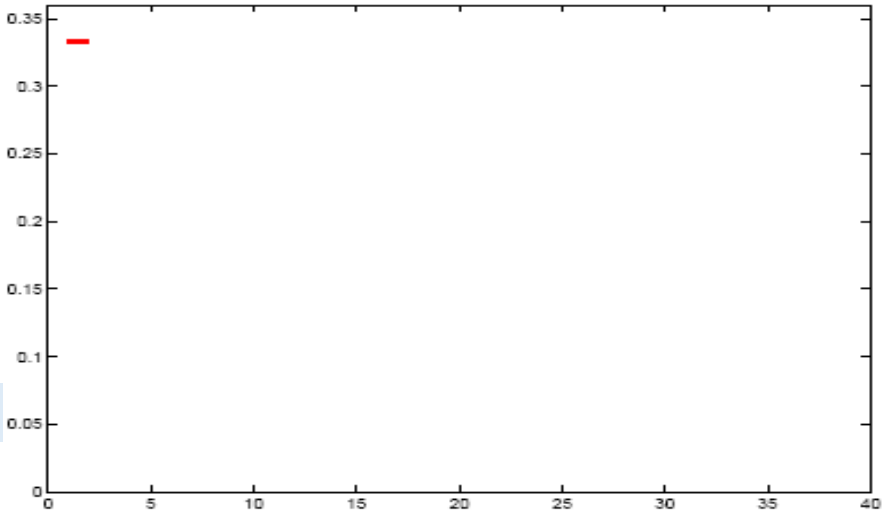




Illustration

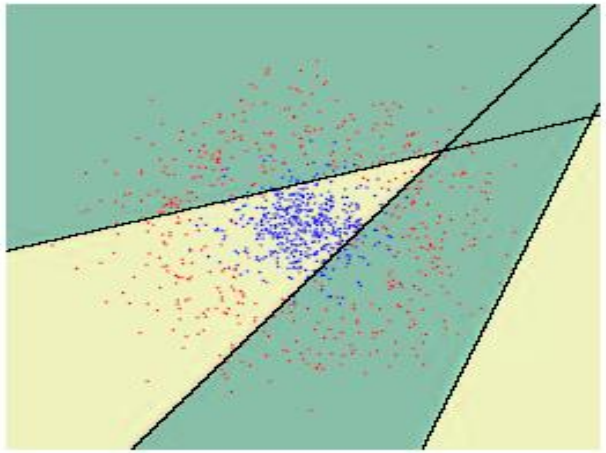
$t = 2$

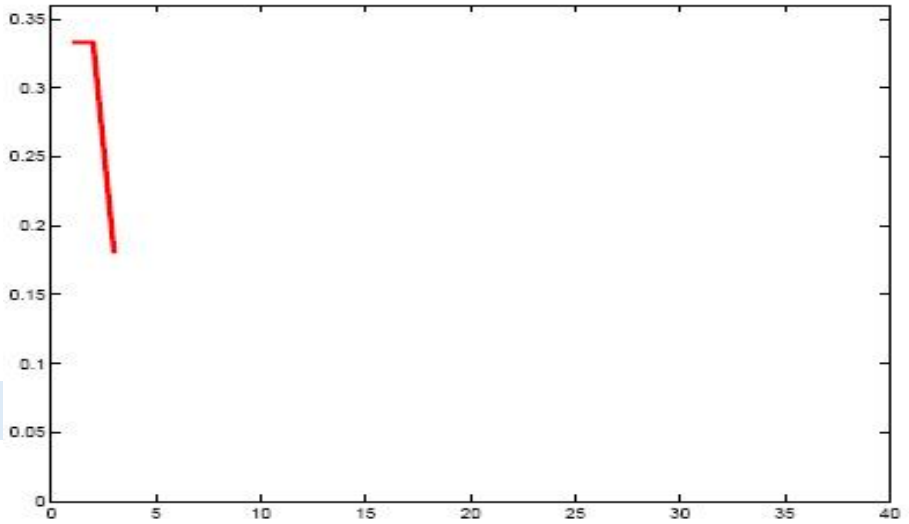




Illustration

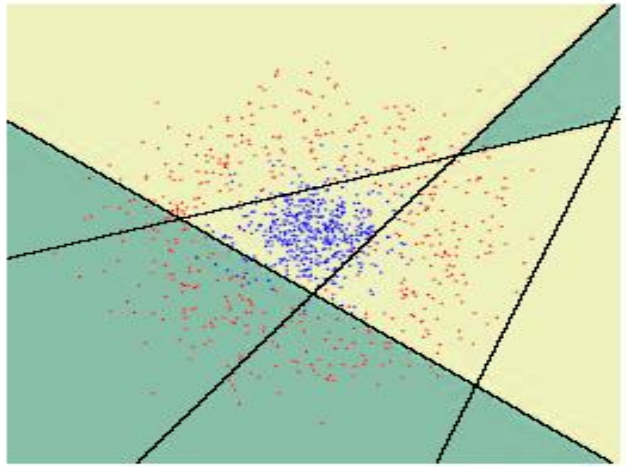
$t = 3$

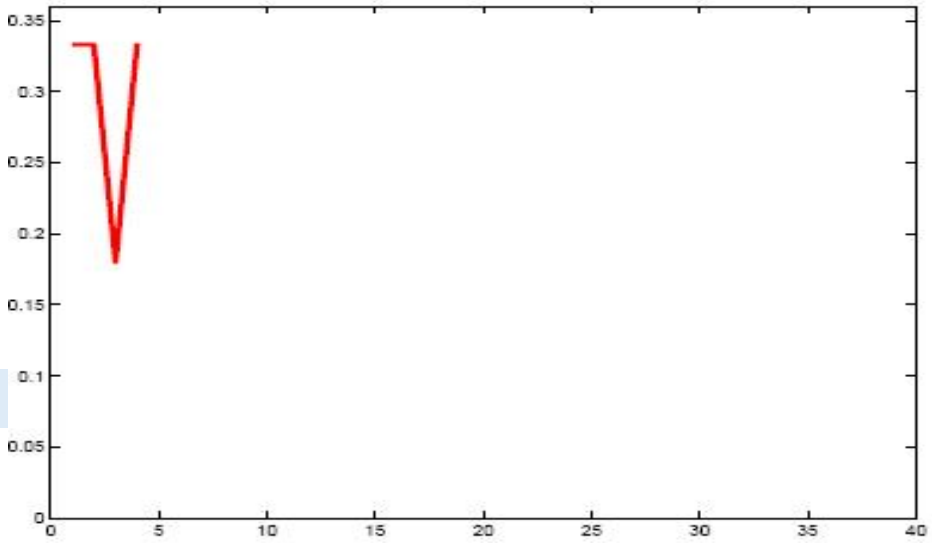




Illustration

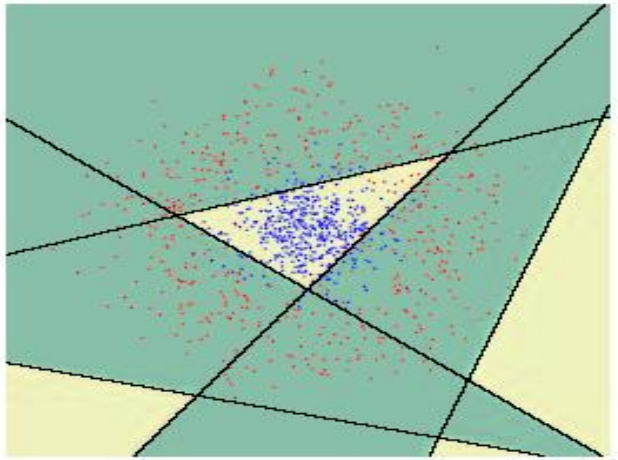
$$t = 4$$

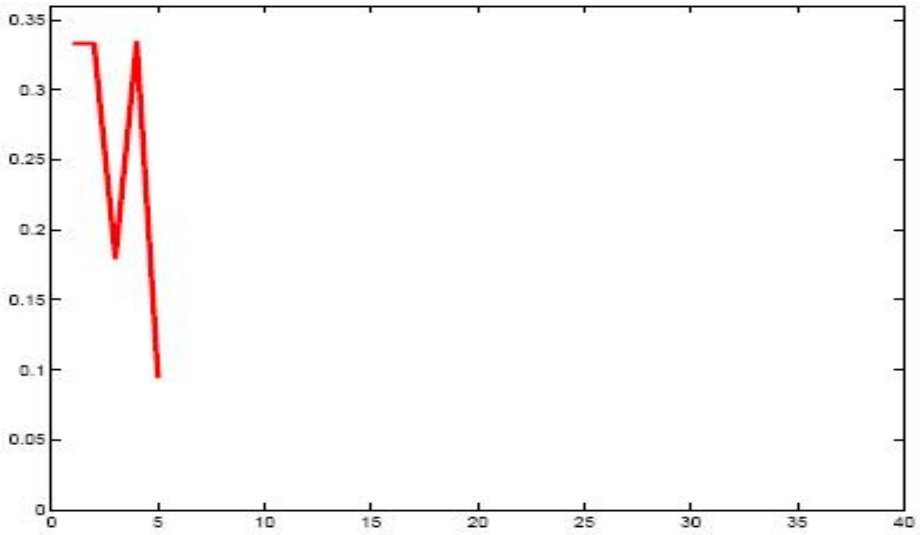




Illustration

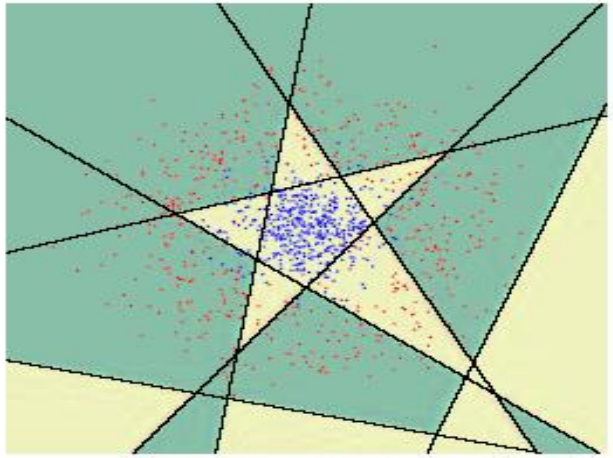
$t = 5$

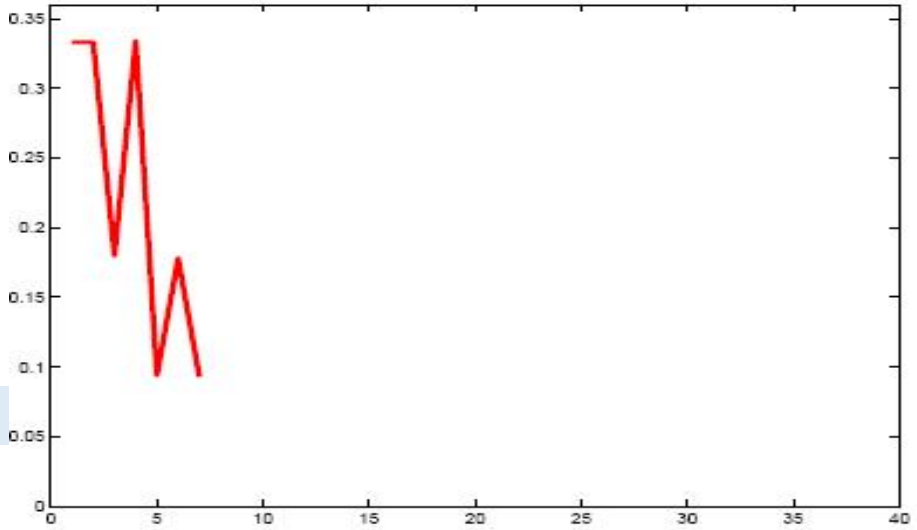




Illustration

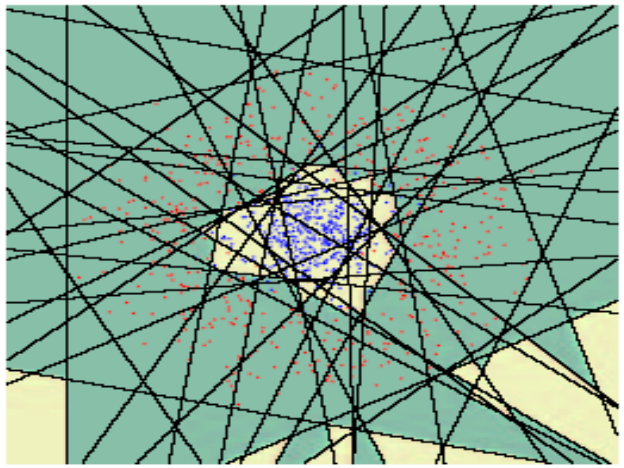
$t = 7$

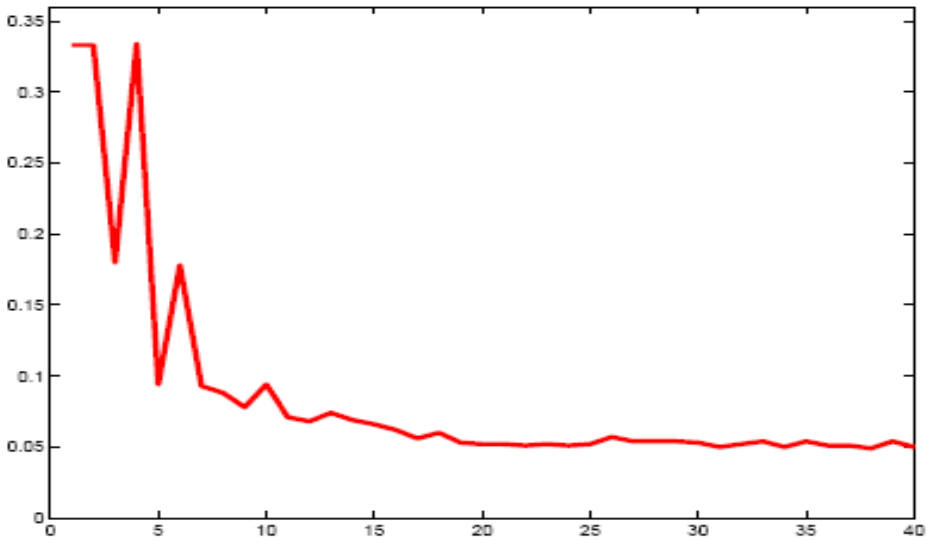




Illustration

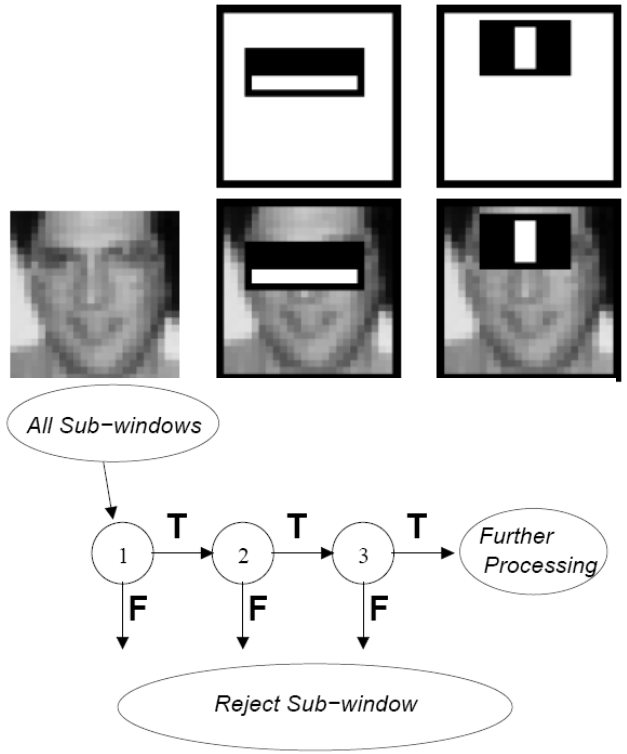
$t = 40$



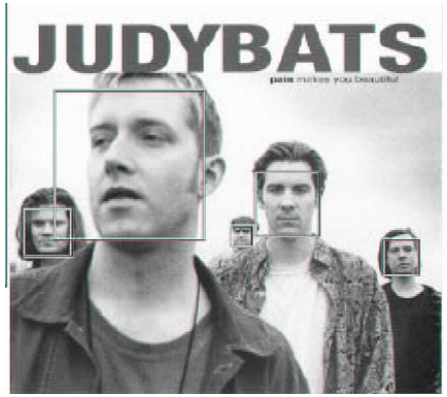


Real-world Use of Boosting: Face Detection

Viola and Jones 2001



A landmark paper in vision!



Adaboost vs Random Forests

- Dietterich (1998) showed that
  - when a fraction of the output labels in the training set are randomly altered, the accuracy of Adaboost degenerates, while bagging and random split selection are more immune to the noise.

Increases in error rates due to noise

Data set	Adaboost	Forest-RI
Glass	1.6	.4
Breast cancer	43.2	1.8
Diabetes	6.8	1.7
Sonar	15.1	−6.6
Ionosphere	27.7	3.8
Soybean	26.9	3.2
Ecoli	7.5	7.9
Votes	48.9	6.3
Liver	10.3	−.2

Classification Methods

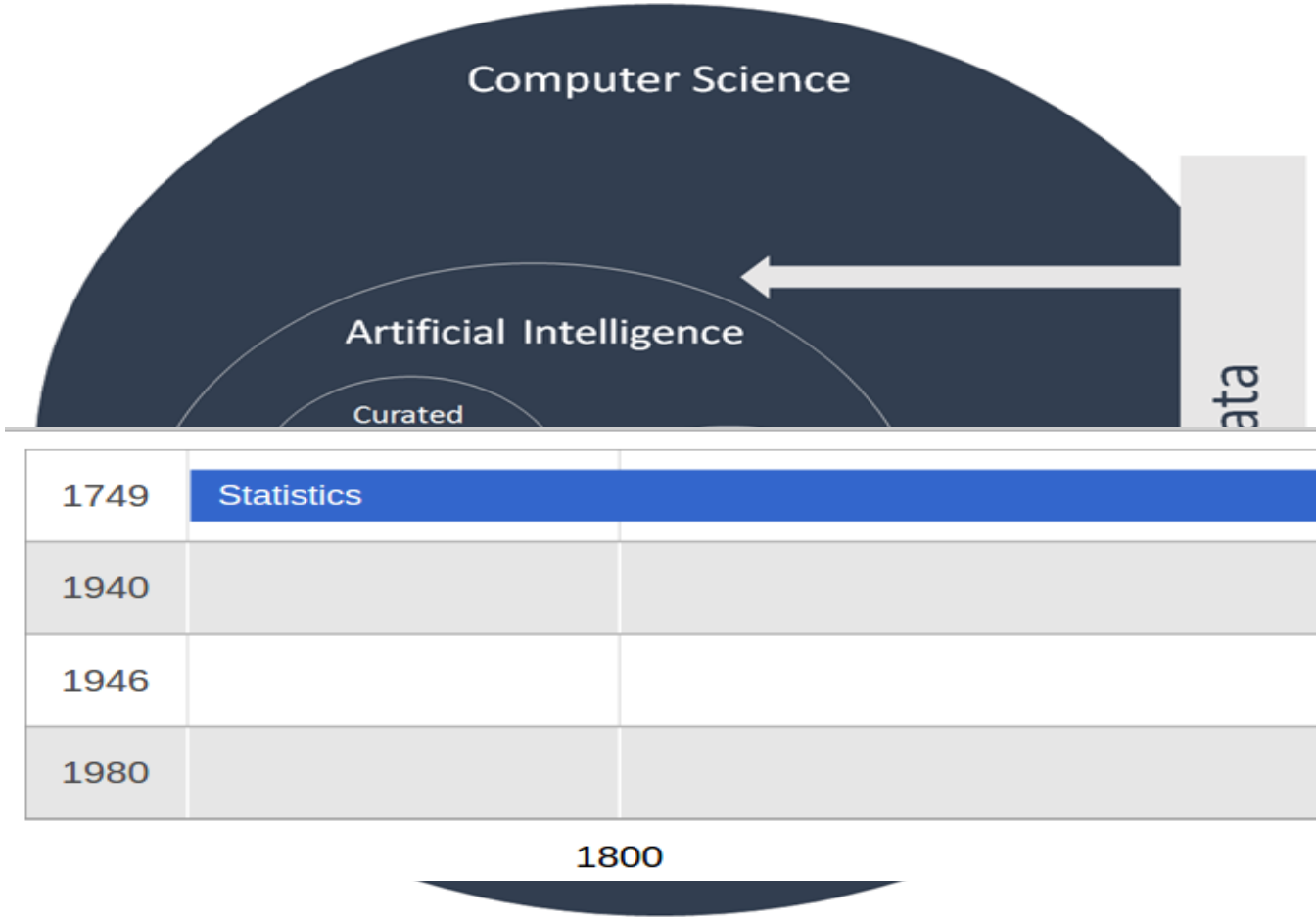
- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks (Deep Learning)
- Ensemble Methods (Boosting, Random Forests)

Relevance

AlphaGo: The Recent Sensation

Machine Learning (vs) Artificial Intelligence

Introduction

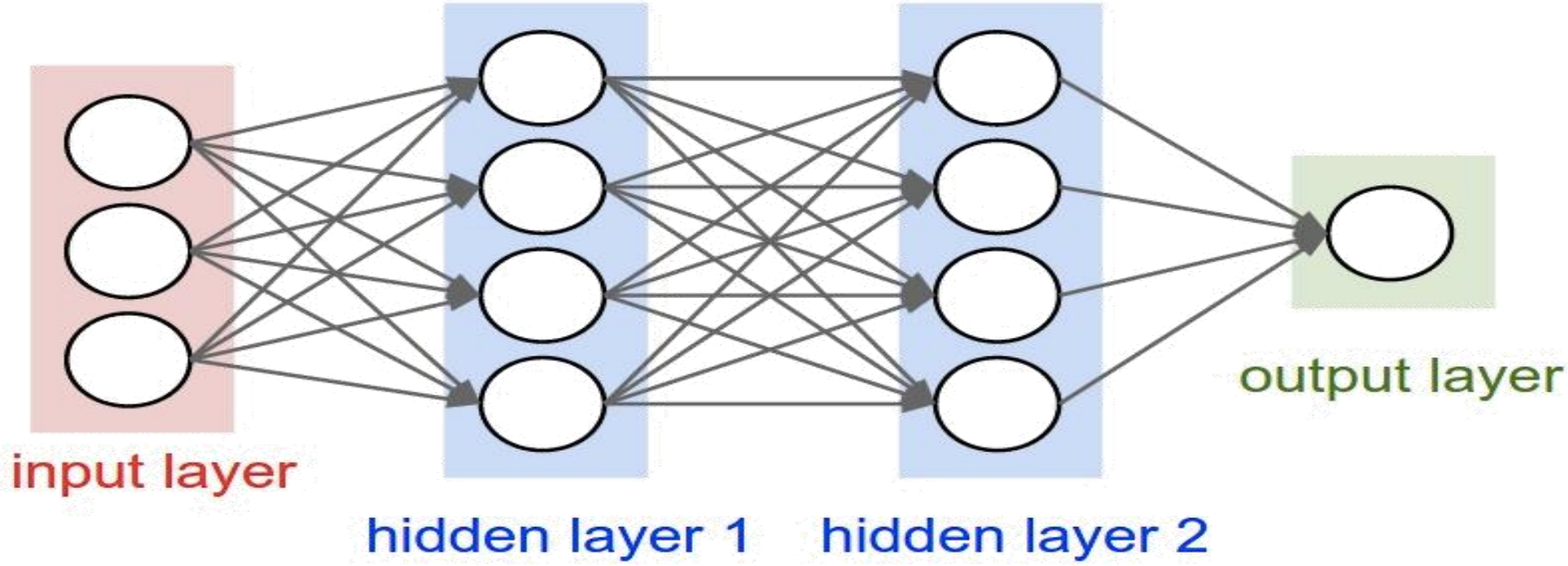


Deep learning: A sub-area of machine learning, that is today understood as representation learning

Deep Learning

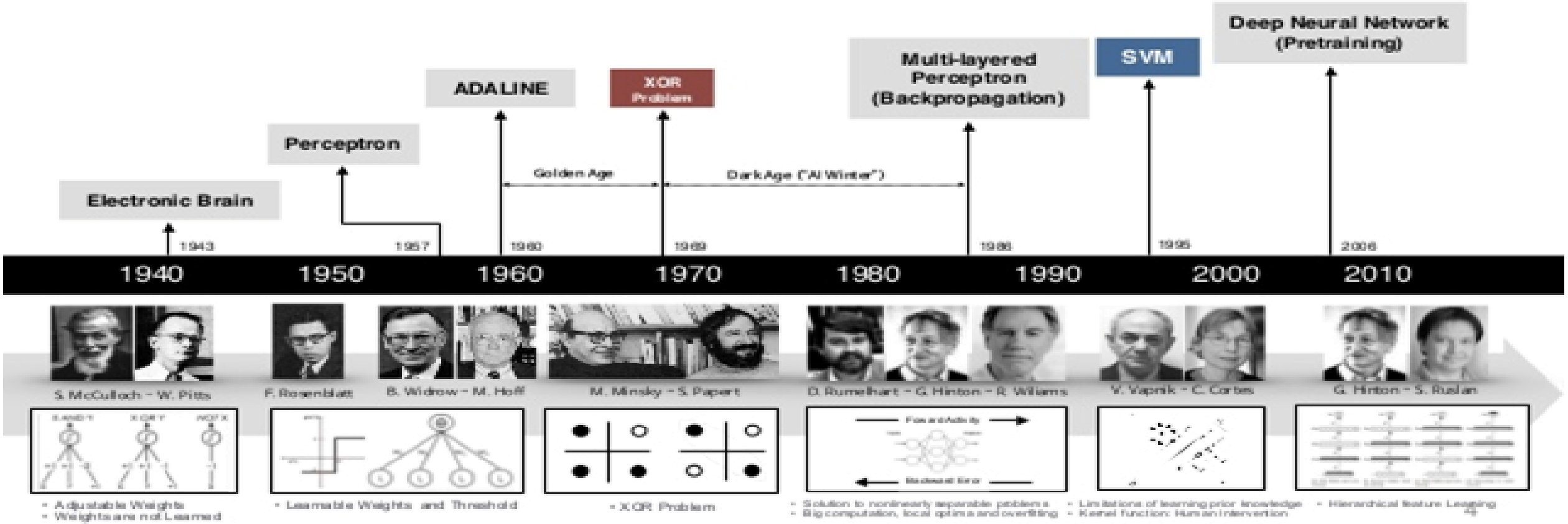
Introduction

- Rebirth of neural networks
- Inspired by the human brain (networks of neurons)



Deep Learning

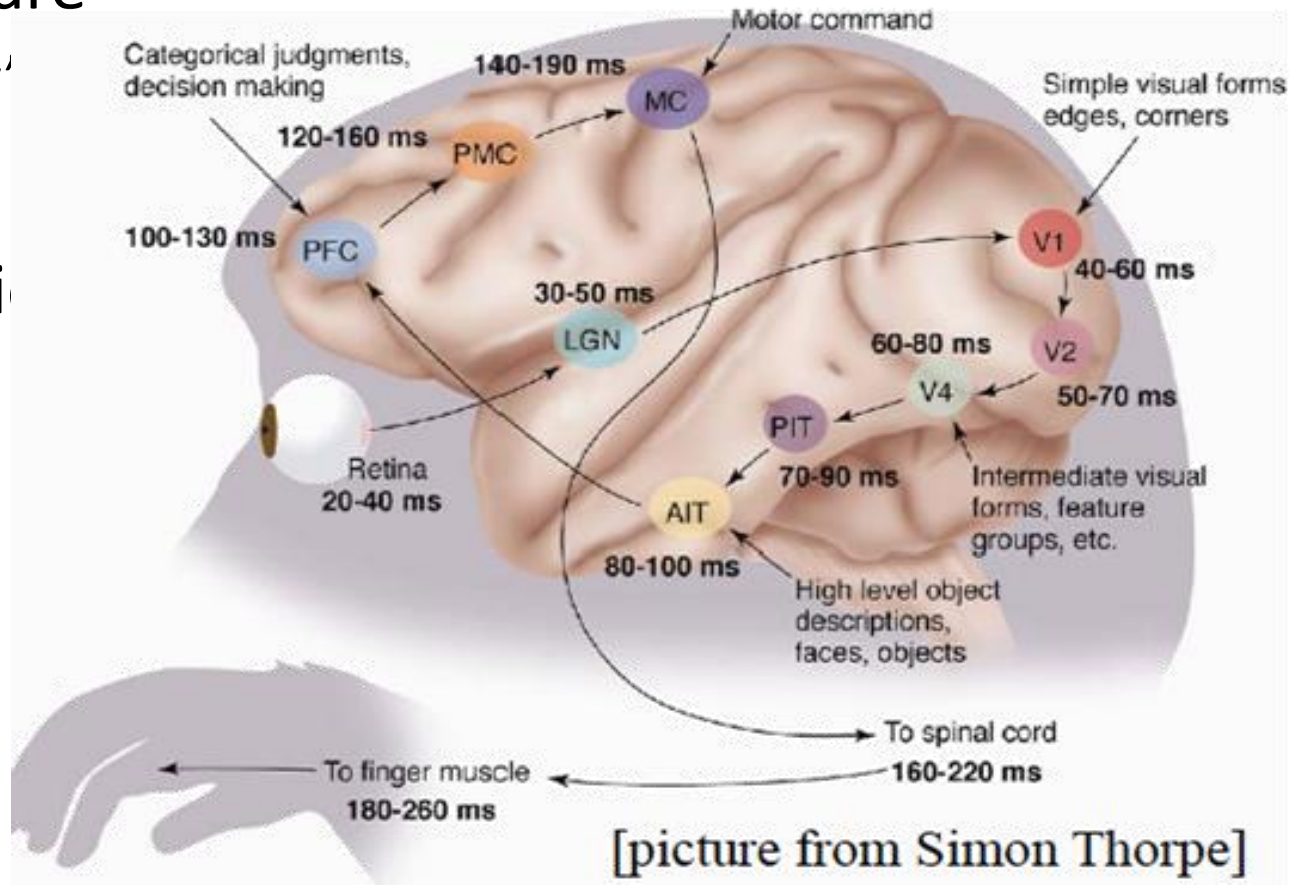
History



Deep Learning

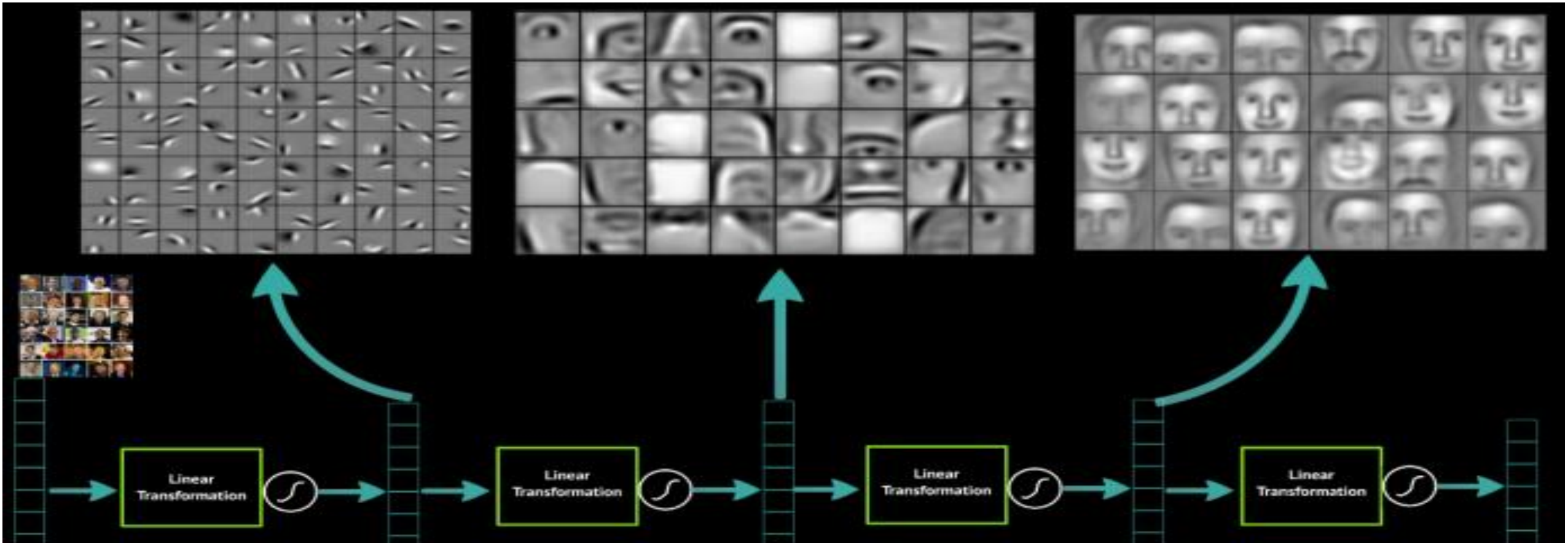
Why is it successful?

- Learns representations of data that are useful (Other ML algorithms are “shallow”
- Similar to the human brain
- Then, why was it not successful earli the 90s)?
  - Computational power
  - Data power



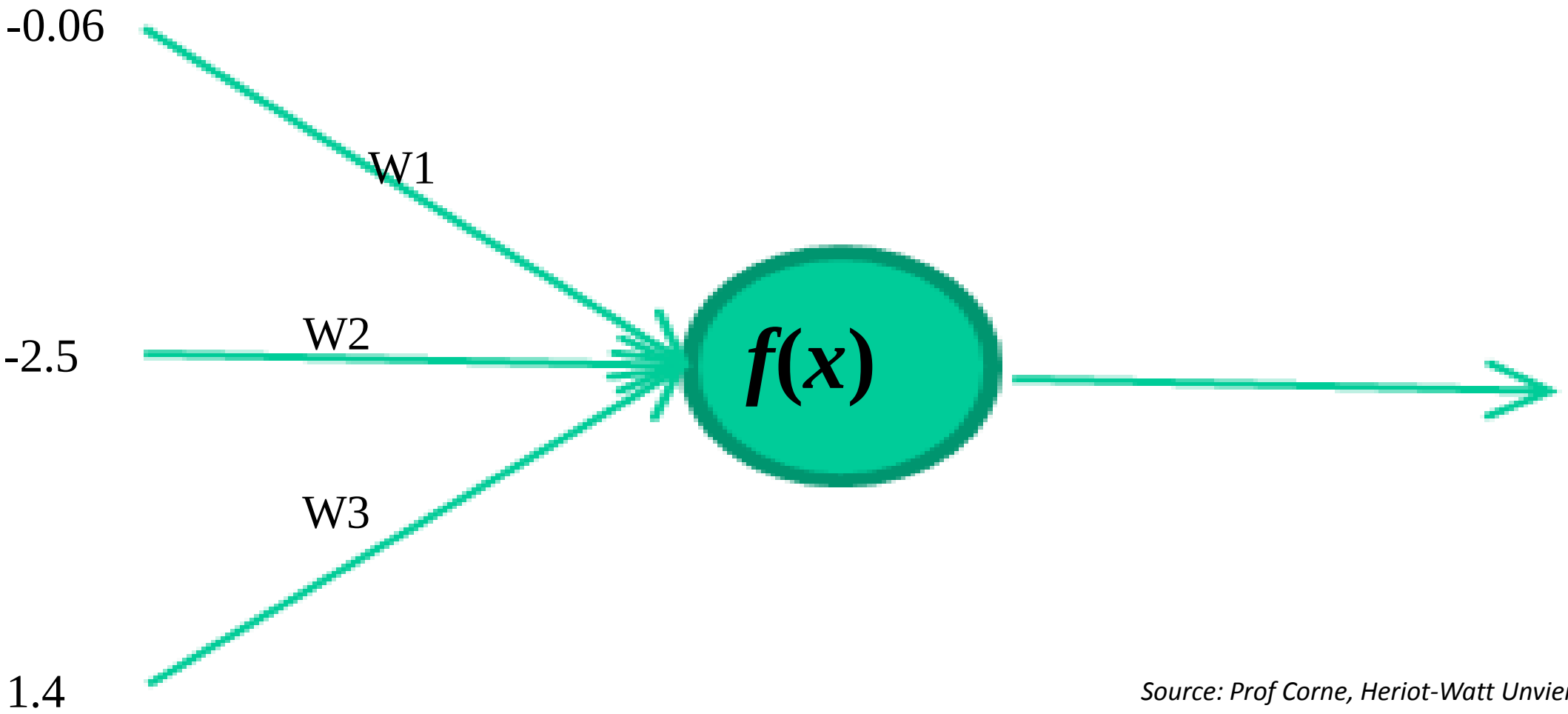
Deep Learning

Why is it successful?



Deep Learning

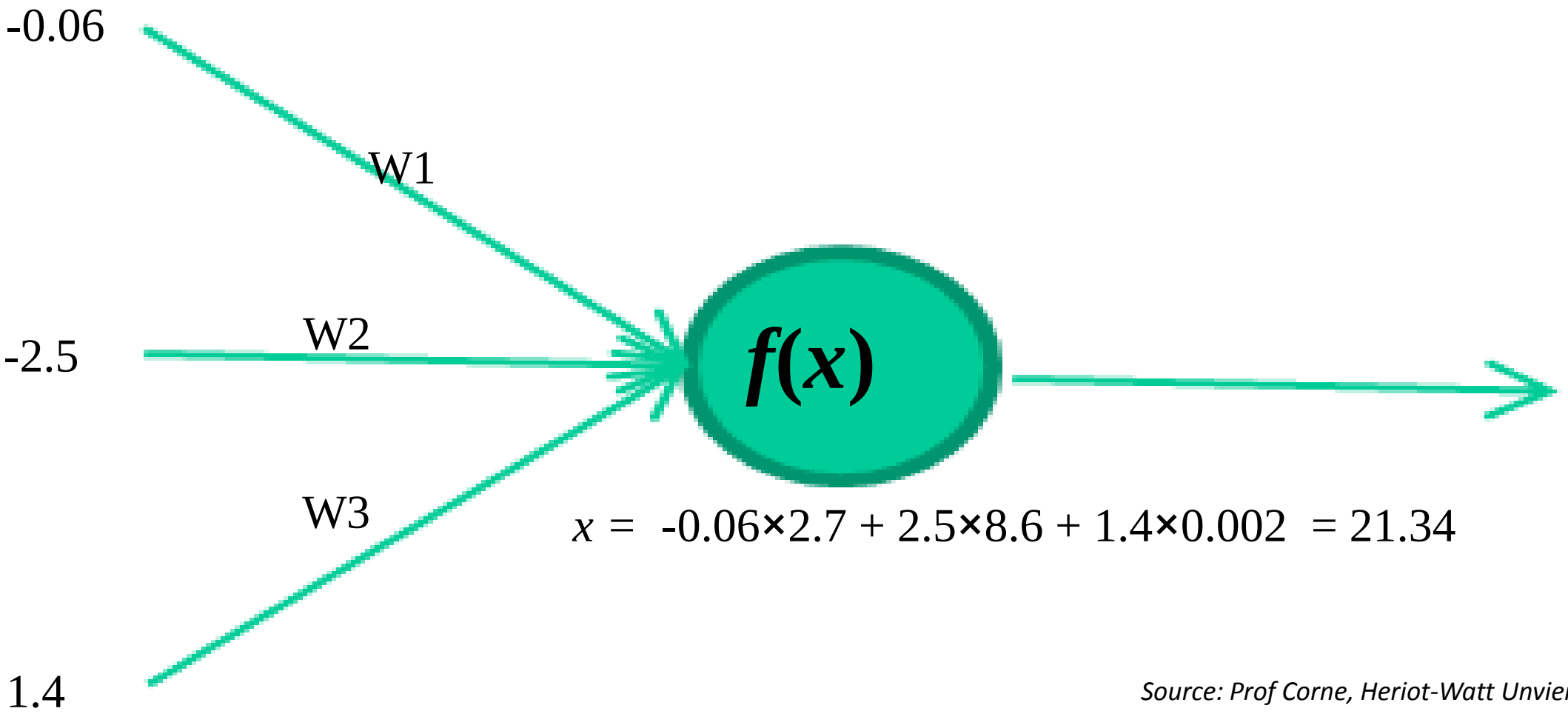
How do they learn?



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

How do they learn?



$$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$$

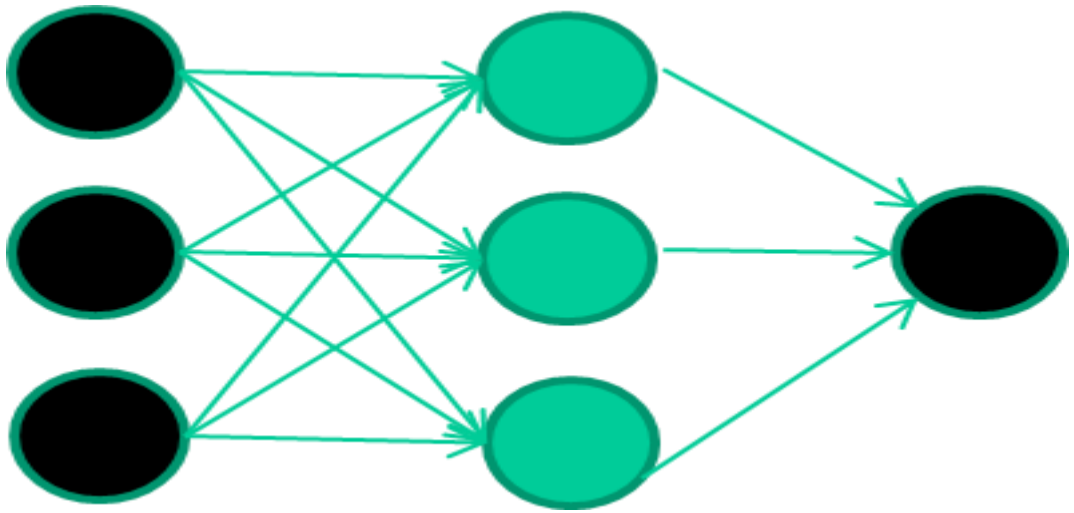
Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

How do they learn?

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

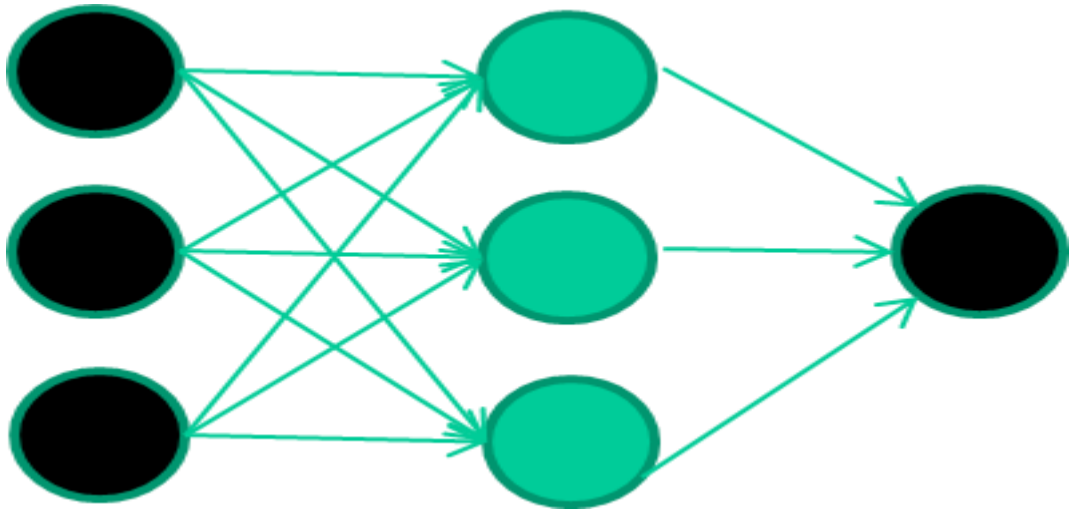
How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Initialise with random weights



Source: Prof Corne, Heriot-Watt Unviersity, UK

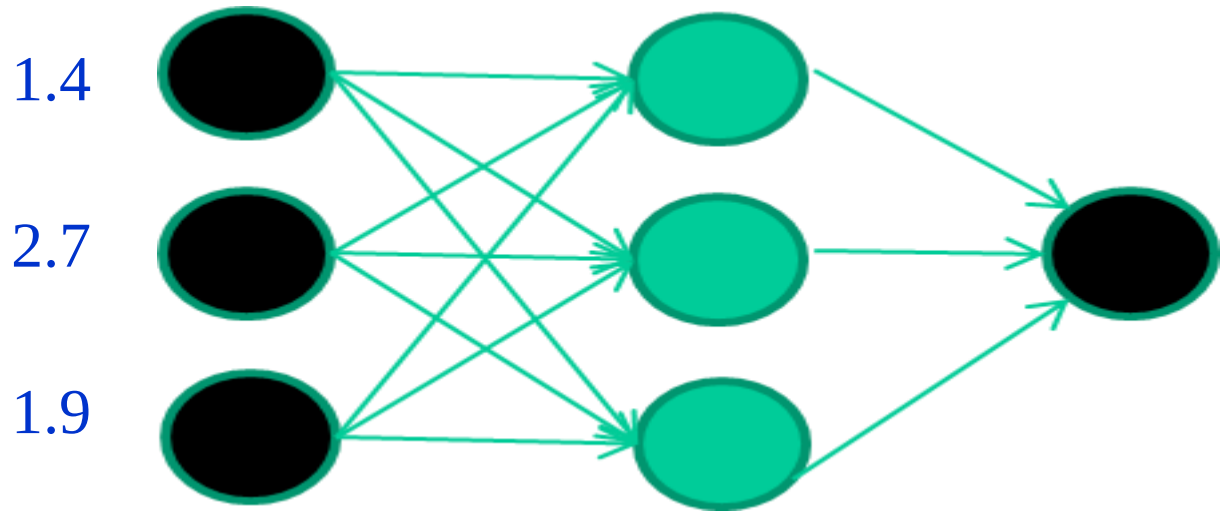
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

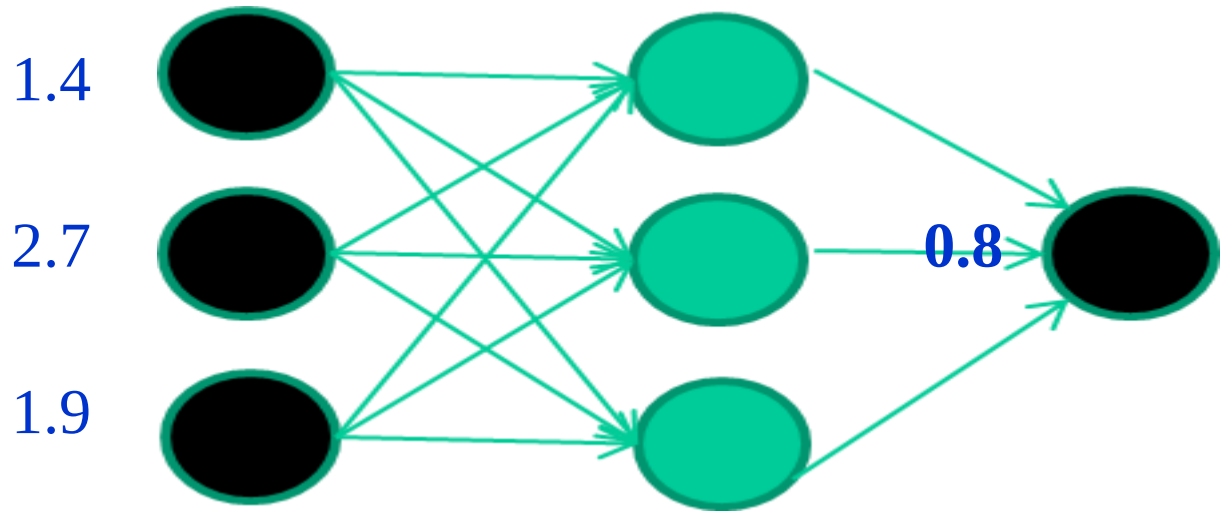
How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

Feed it through to get output



Source: Prof Corne, Heriot-Watt Unviersity, UK

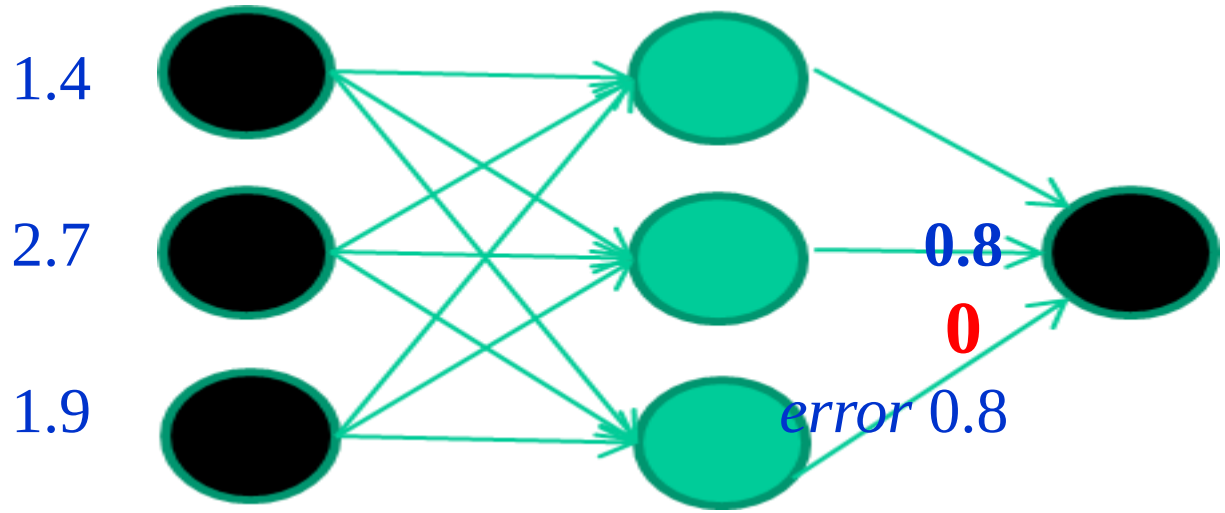
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output



Source: Prof Corne, Heriot-Watt Unviersity, UK

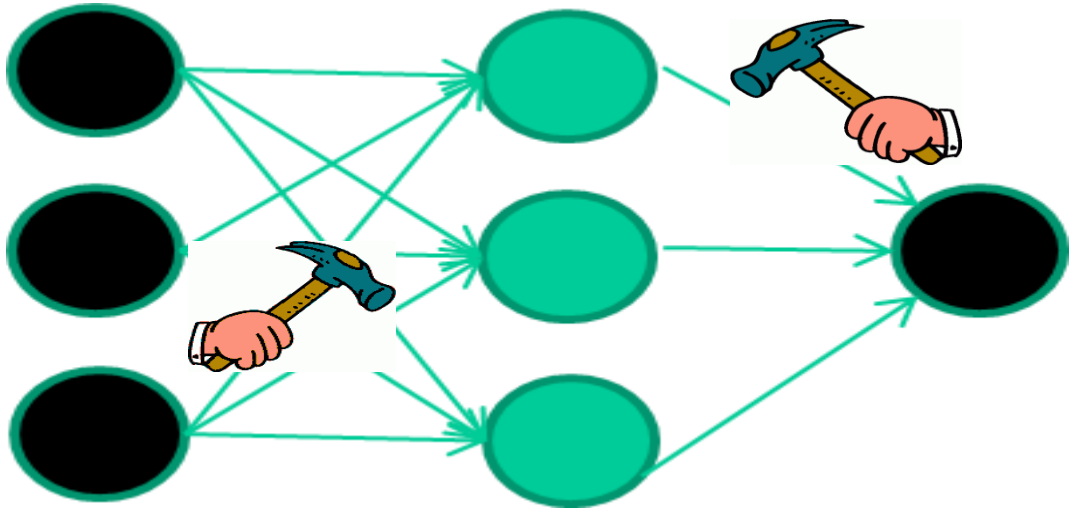
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error



Source: Prof Corne, Heriot-Watt Unviersity, UK

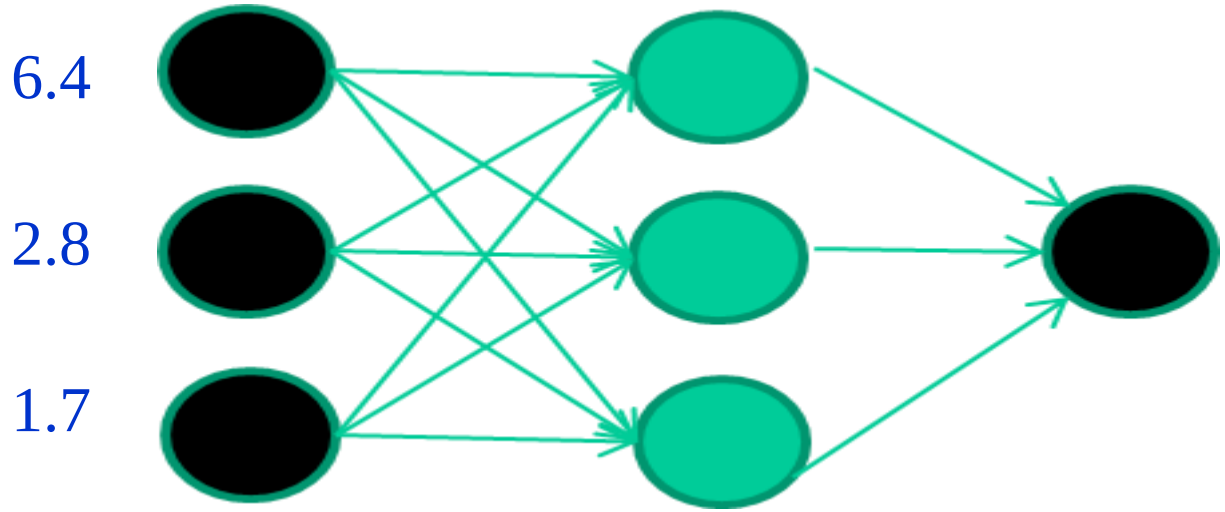
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern



Source: Prof Corne, Heriot-Watt Unviersity, UK

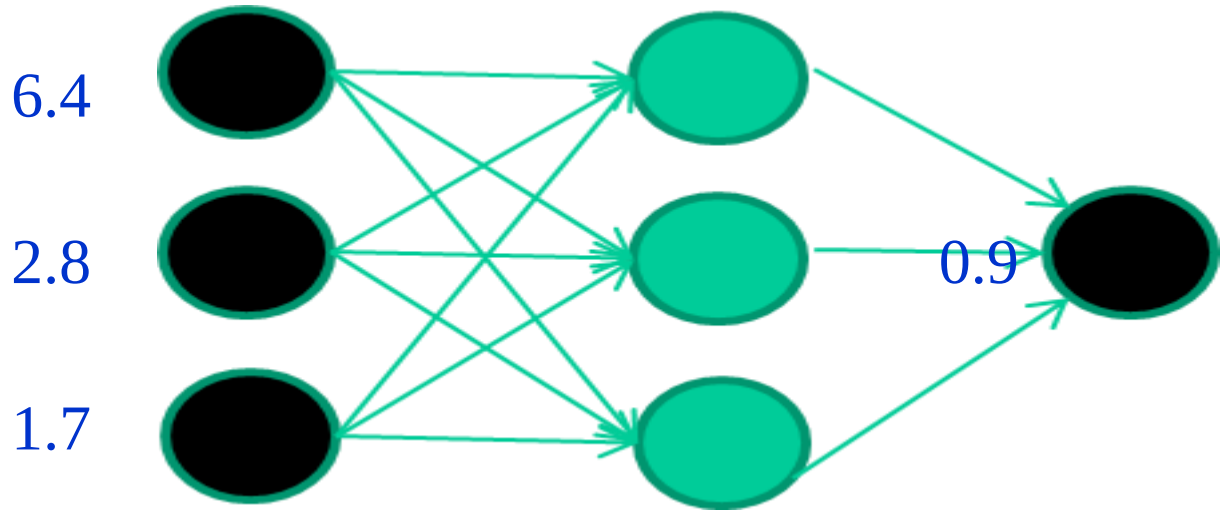
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output



Source: Prof Corne, Heriot-Watt Unviersity, UK

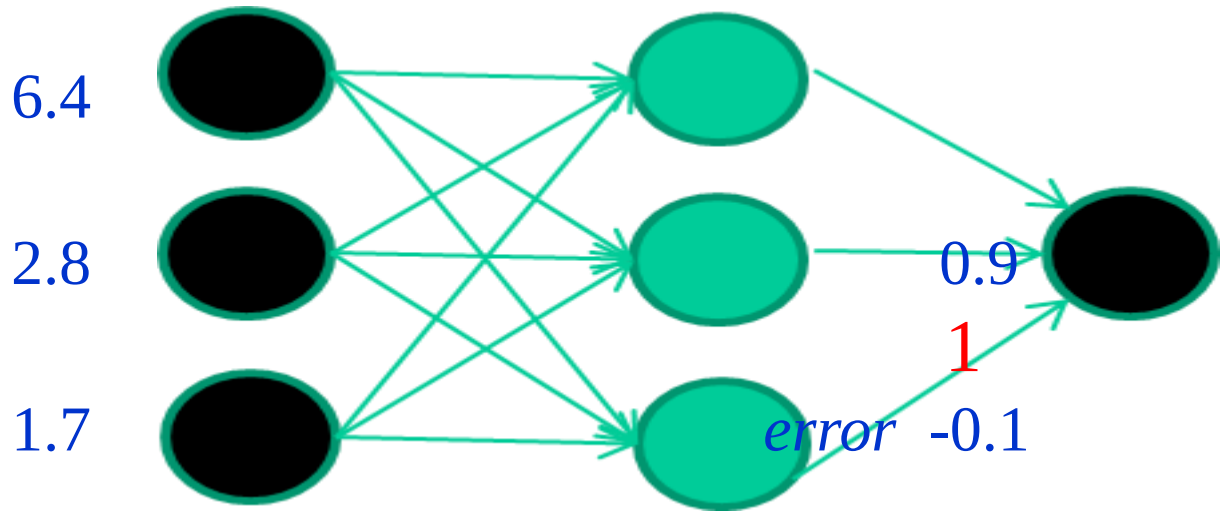
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output



Source: Prof Corne, Heriot-Watt Unviersity, UK

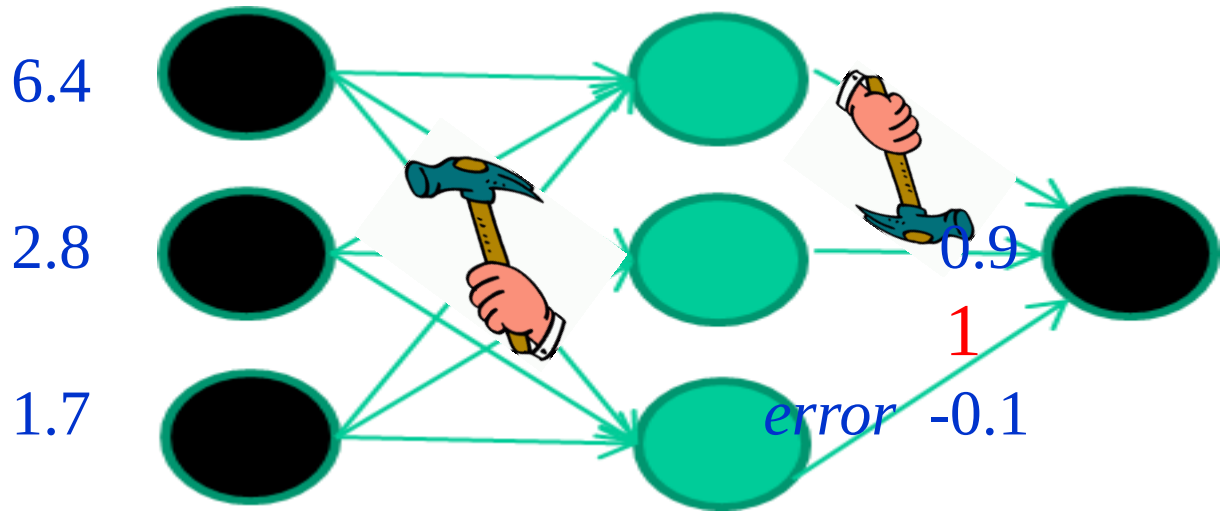
Deep Learning

How do they learn?

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error



Source: Prof Corne, Heriot-Watt Unviersity, UK

Deep Learning

How do they learn?

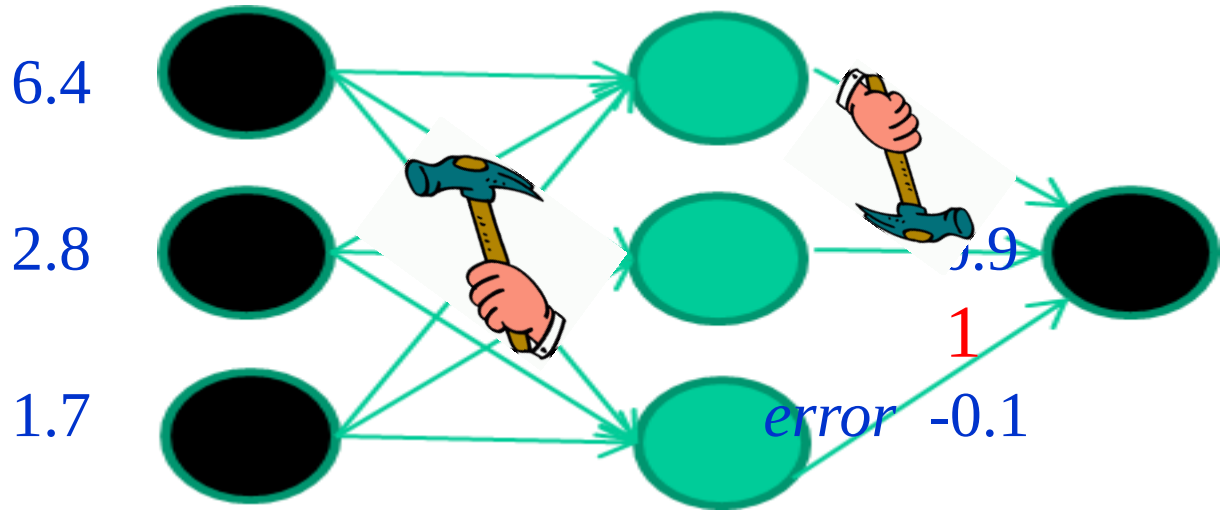
Source: Prof Corne, Heriot-Watt Unviersity, UK

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Repeat this thousands, maybe millions of times – each time
taking a random training instance, and making slight
weight adjustments, reduce the error

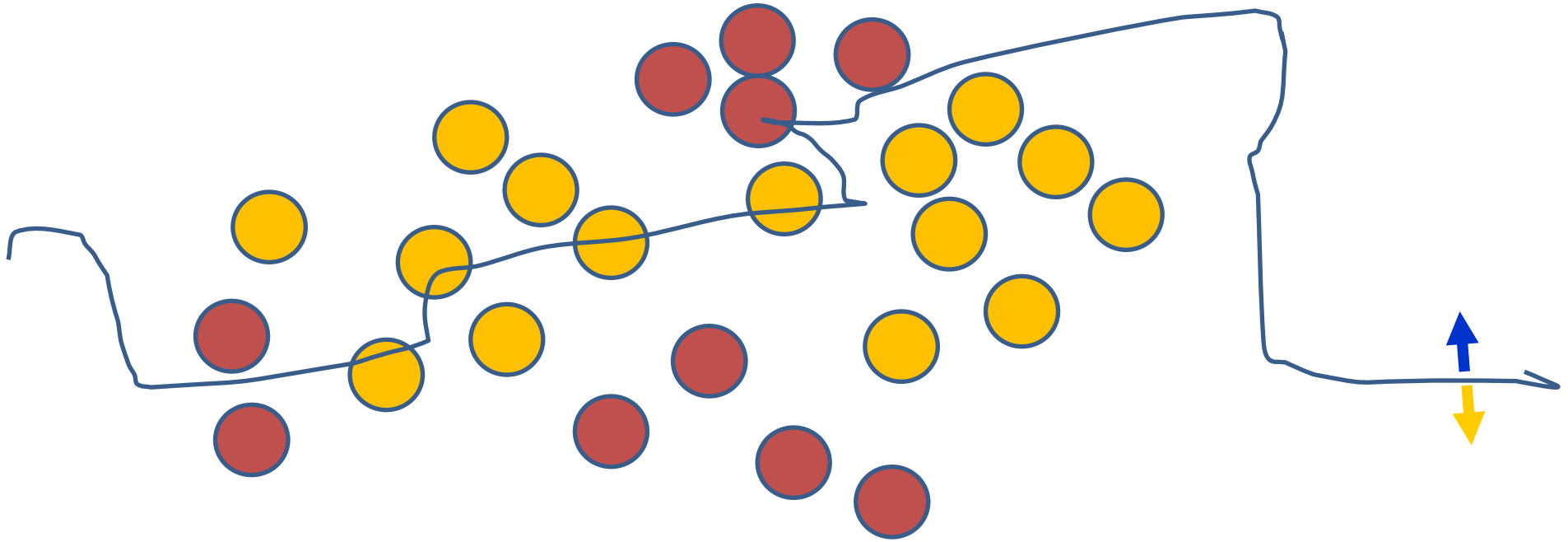
And so on ….



Called “Gradient Descent”

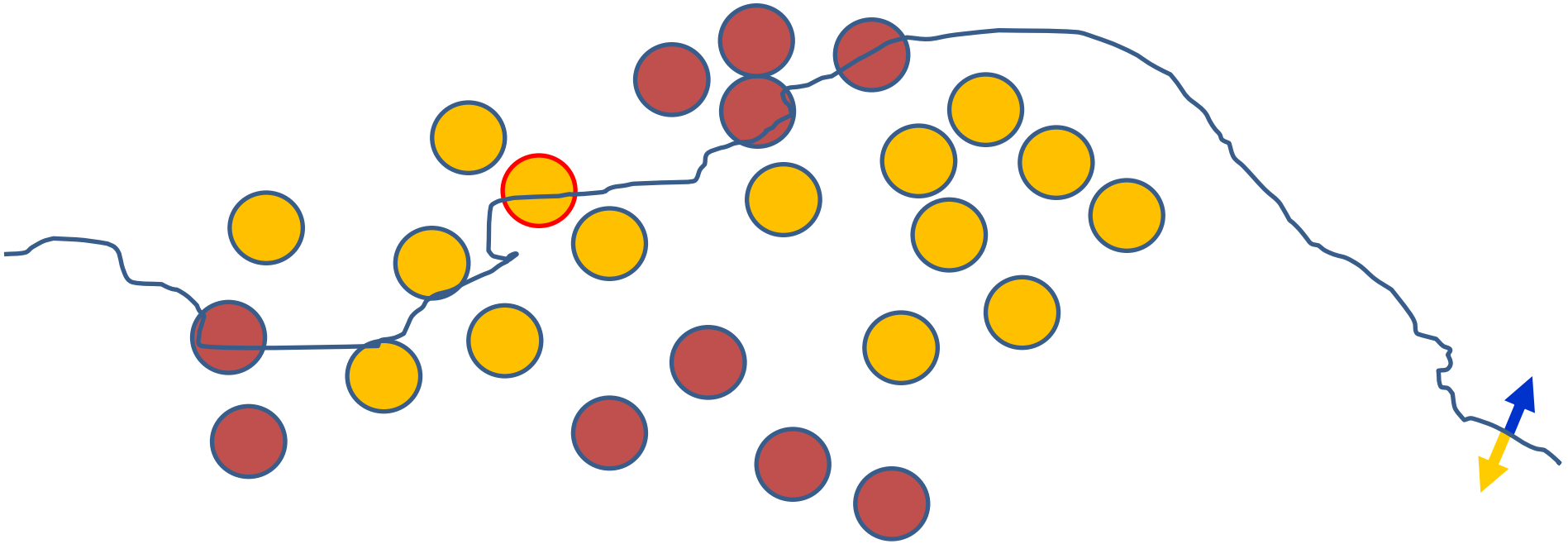
The decision boundary perspective…

Initial random weights



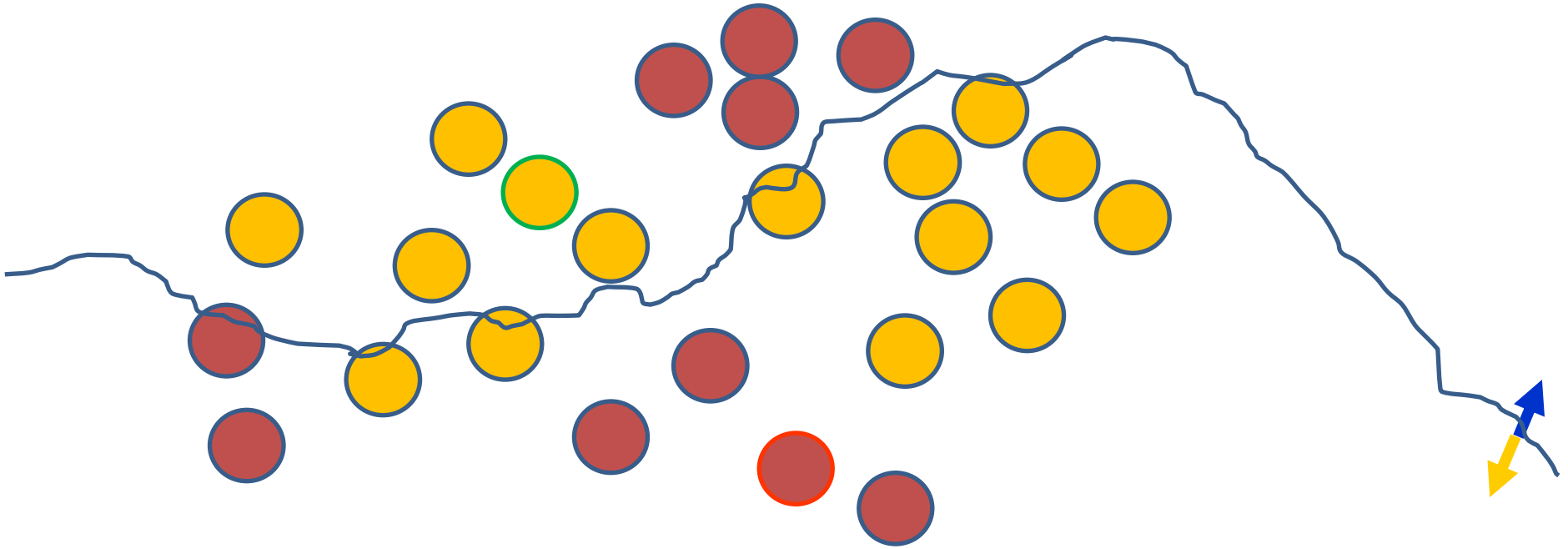
The decision boundary perspective…

Present a training instance / adjust the weights



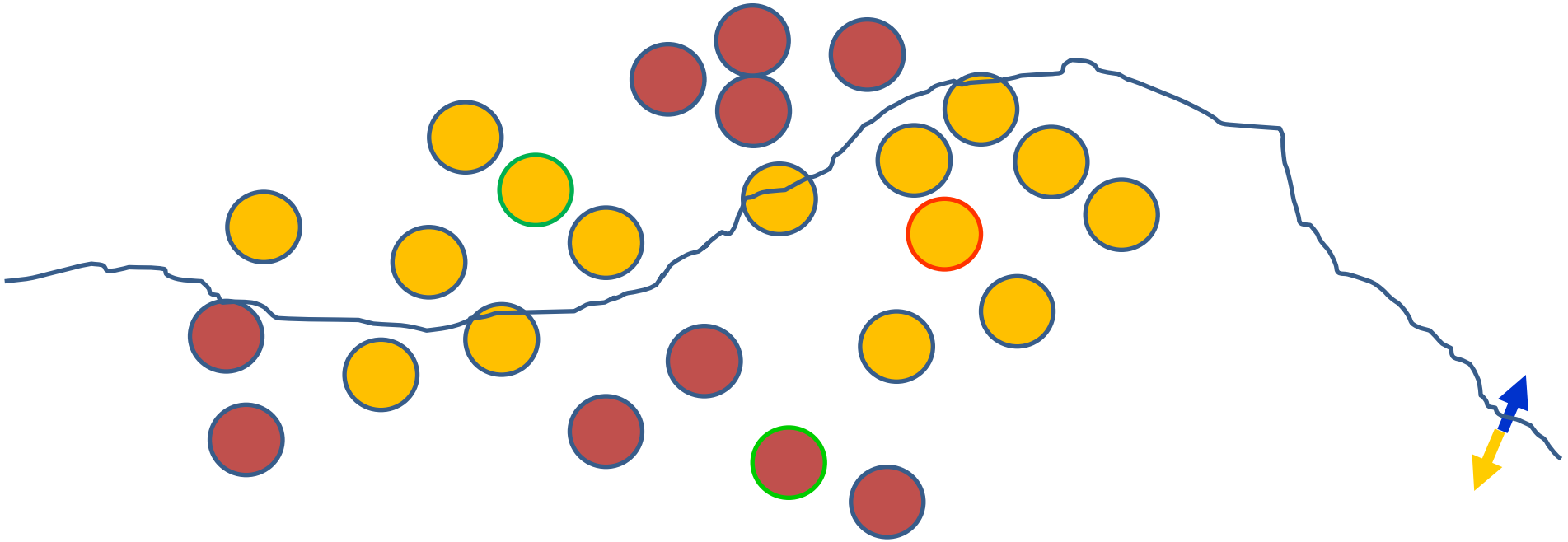
The decision boundary perspective…

Present a training instance / adjust the weights



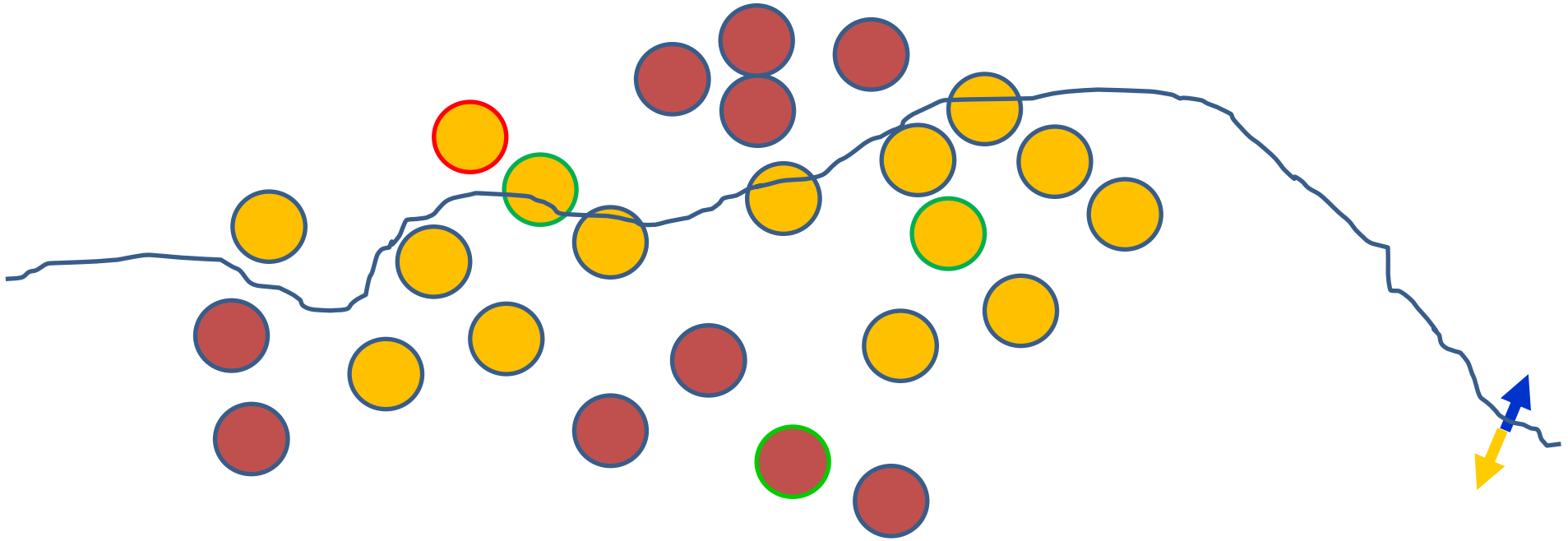
The decision boundary perspective…

Present a training instance / adjust the weights



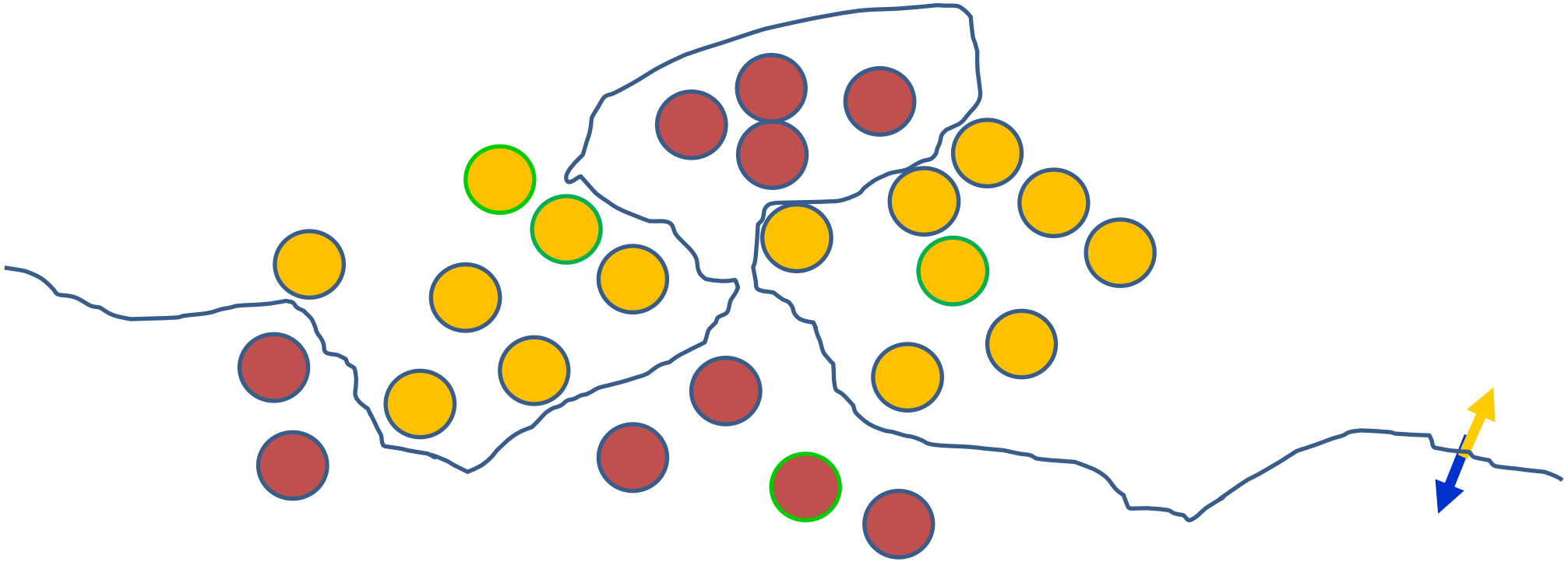
The decision boundary perspective…

Present a training instance / adjust the weights



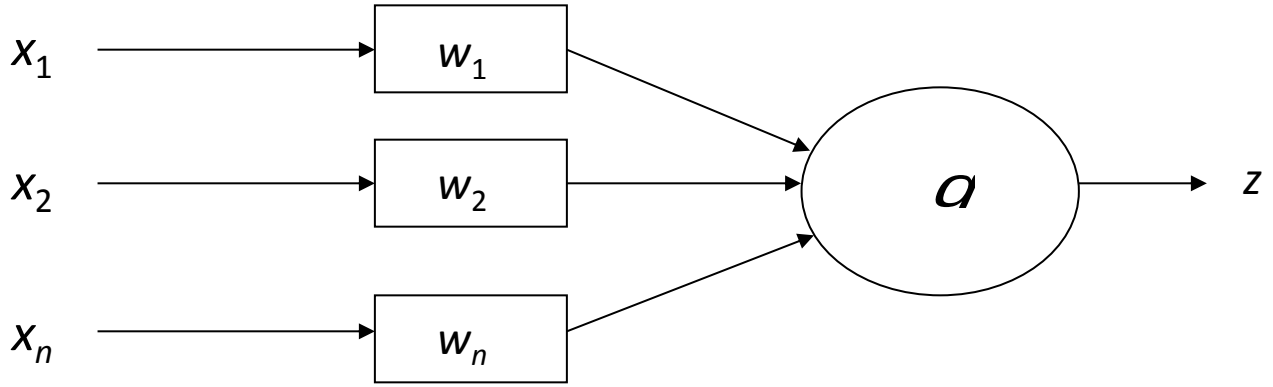
The decision boundary perspective…

Eventually ….



Neural Networks Training: Backpropagation

Perceptrons

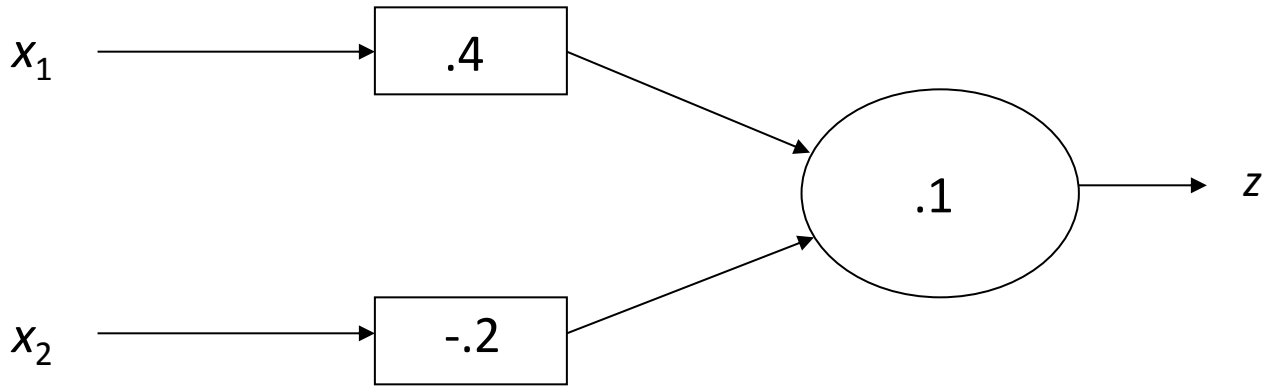


$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq q \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < q \end{cases}$$

- Learn weights such that an objective function is maximized.
- What objective function should we use?
- What learning algorithm should we use?

Neural Networks Training: Backpropagation

Perceptrons

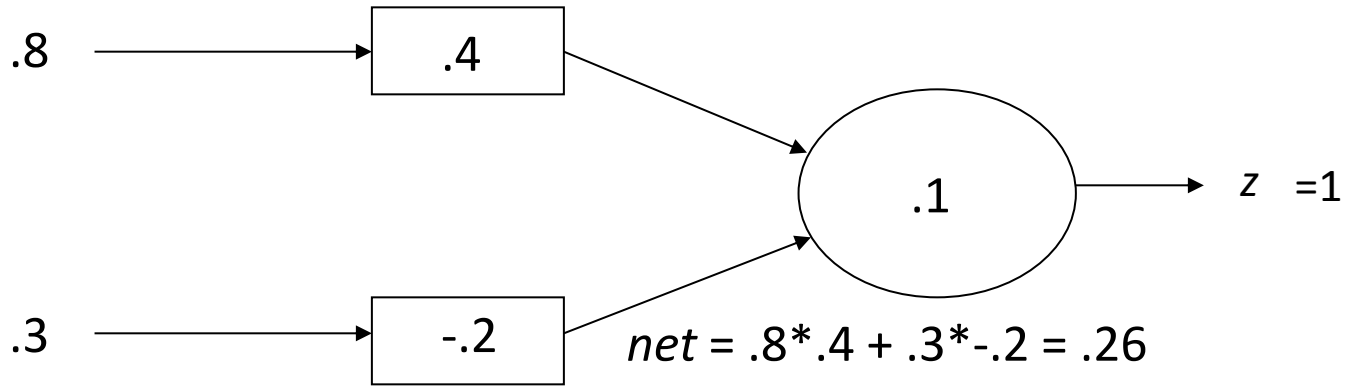


x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq q \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < q \end{cases}$$

Neural Networks Training: Backpropagation

First Training Instance

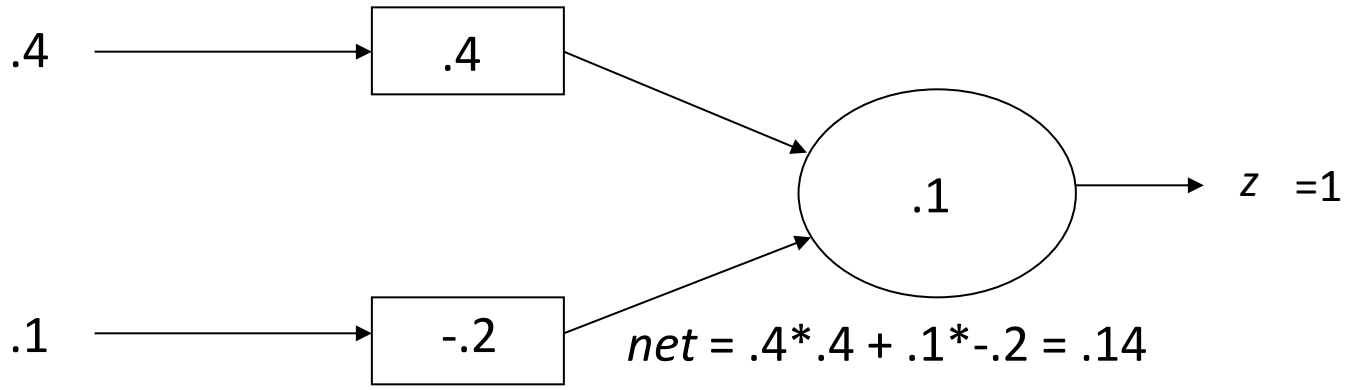


x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq q \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < q \end{cases}$$

Neural Networks Training: Backpropagation

Second Training Instance



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq q \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < q \end{cases}$$

$$\Delta w_i = (t - z) * c * x_i$$

Neural Networks Training: Backpropagation

Perceptron Rule Learning

$$\Delta w_i = c(t - z)\, x_i$$

- Where w_i is the weight from input i to perceptron node, c is the learning rate, t_j is the target for the current instance, z is the current output, and x_i is i^{th} input
- Least perturbation principle
  - Only change weights if there is an error
  - small c rather than changing weights sufficient to make current pattern correct
  - Scale by x_i
- Create a perceptron node with n inputs
- Iteratively apply a pattern from the training set and apply the perceptron rule
- Each iteration through the training set is an *epoch*
- Continue training until total training set error ceases to improve
- Perceptron Convergence Theorem: Guaranteed to find a solution in finite time if a solution exists

Neural Networks Training: Backpropagation

Multi-Layer Perceptrons

- Extension of perceptrons to multiple layers
- 1. Initialize network with random weights
- 2. For all training cases (called examples):
  - **a.** Present training inputs to network and calculate output
  - **b.** For all layers (starting with output layer, back to input layer):
    - i. Compare network output with correct output (error function)
    - ii. Adapt weights in current layer

Neural Networks Training: Backpropagation

Multi-Layer Perceptrons

- Method for learning weights in feed-forward (FF) nets
- Can't use Perceptron Learning Rule
  - no teacher values are possible for hidden units
- Use gradient descent to minimize the error
  - propagate deltas to adjust for errors
    backward from outputs
    to hidden layers
    to inputs

Neural Networks Training: Backpropagation

Multi-Layer Perceptrons

- The idea of the algorithm can be summarized as follows :

1. Computes the **error term for the output units** using the observed error.

- 2. From output layer, repeat
  - propagating the error term back to the previous layer and updating the weights between the two layers until the earliest hidden layer is reached.

Neural Networks Training: Backpropagation

Multi-Layer Perceptrons

- Initialize weights (typically random!)
- Keep doing epochs
  - For each example **e** in training set do
    - **forward pass** to compute
      - O = neural-net-output(network,e)
      - miss = (T-O) at each output unit
    - **backward pass** to calculate deltas to weights
    - update all weights
  - end
- until tuning set error stops improving

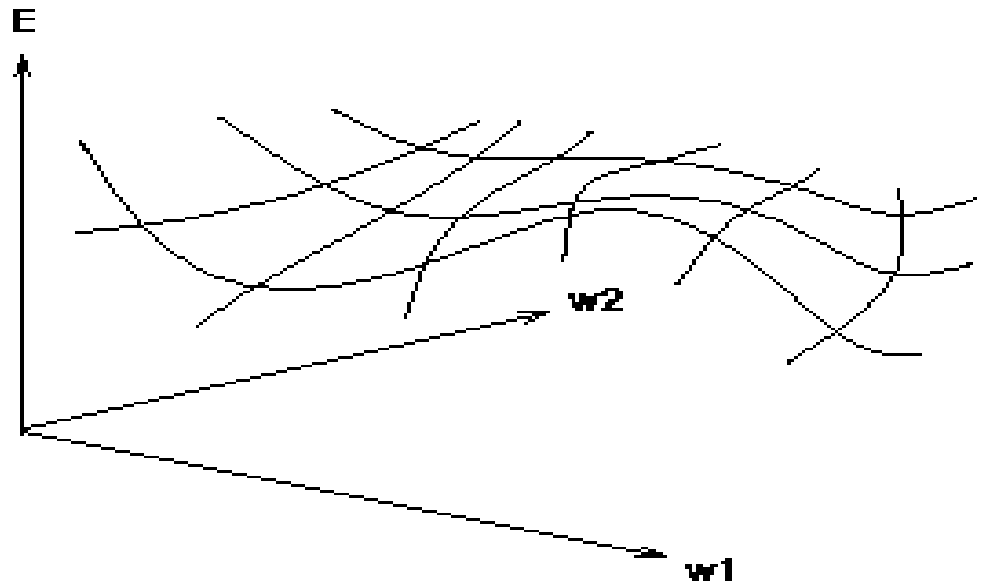
Forward pass

Backward pass explained in next slide

Neural Networks Training: Backpropagation

Gradient Descent

- Think of the N weights as a point in an N-dimensional space
- Add a dimension for the observed error
- Try to minimize your position on the “error surface”

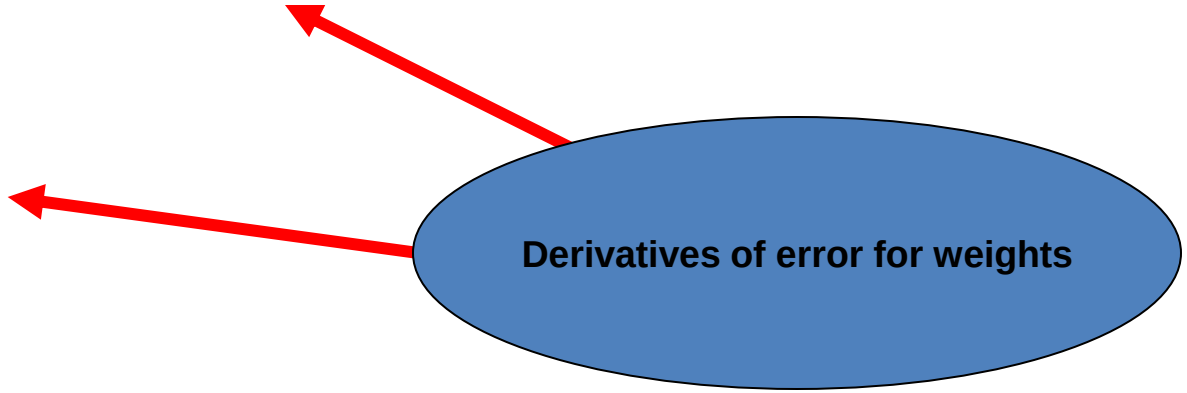


Neural Networks Training: Backpropagation

Compute deltas

Gradient Descent

- Trying to make error decrease the fastest
- Compute:
  - $Grad_E$ = [dE/dw1, dE/dw2, . . ., dE/dwn]
- Change i-th weight by
  - $delta_{wi}$ = -*alpha* * dE/dwi



- We need a derivative!
- Activation function must be **continuous**, differentiable, non-decreasing, and easy to compute

Neural Networks Training: Backpropagation

Updating Hidden-to-Output

- We have teacher supplied desired values

- $\text{delta}_{wji} = \alpha * a_j * (T_i - Oi) * g'(in_i)$

  $= \alpha * a_j * (T_i - O_i) * O_i * (1 - O_i)$

  – for sigmoid the derivative is, $g'(x) = g(x) * (1 - g(x))$

alpha

Here we have general formula with derivative, next we use for sigmoid

miss

derivative

Neural Networks Training: Backpropagation

Updating interior weights

- Layer k units provide values to all layer k+1 units
  - "miss" is ***sum of misses*** from all units on k+1
  - $miss_j = \Sigma [a_i(1- a_i) (T_i - a_i) w_{ji}]$
  - weights coming into this unit are adjusted based on their contribution

$delta_{kj} = \alpha * I_k * a_j * (1 - a_j) * miss_j$

For layer k+1

Compute deltas

Making Choices

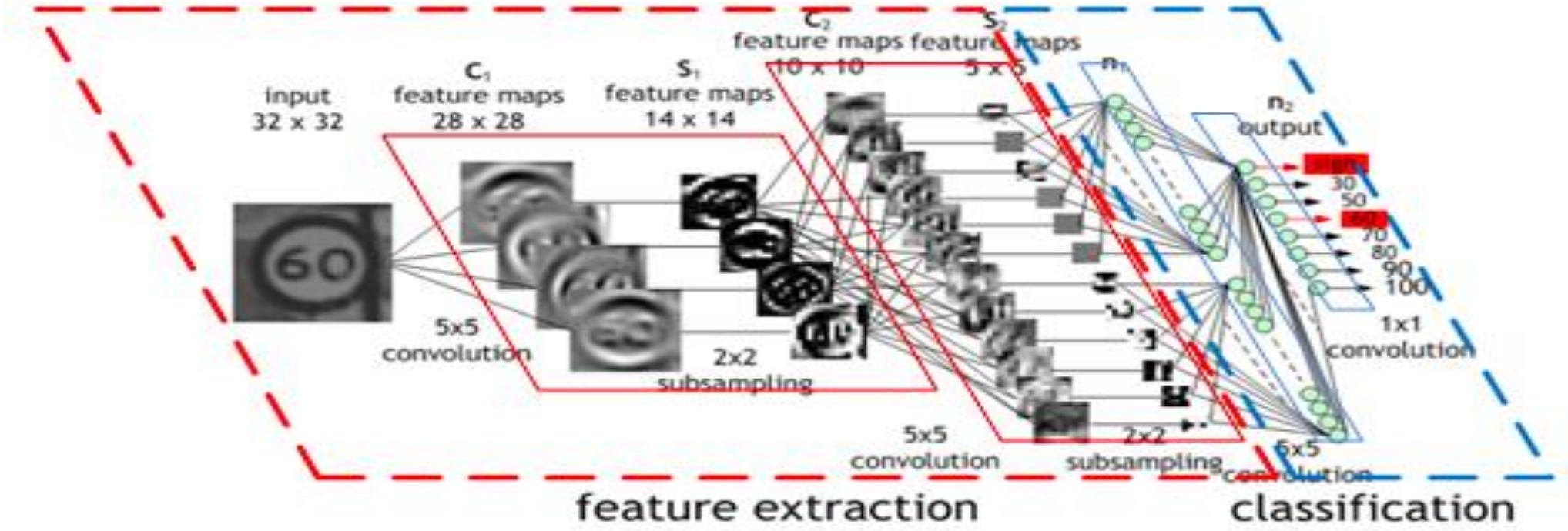
Backpropagation

- How do we pick α?
  - Tuning set, or
  - Cross validation, or
  - Small for slow, conservative learning
- How many hidden layers?
  - Too few ==> can't learn
  - Too many ==> poor generalization
- How big a training set?
  - Determine your target error rate, *e;* Success rate is 1- *e*
  - Typical training set approx. n/*e*, where n is the number of weights in the net
  - Example:
    - *e* = 0.1, n = 80 weights; Training set size 800

Deep Learning Architectures

Variants

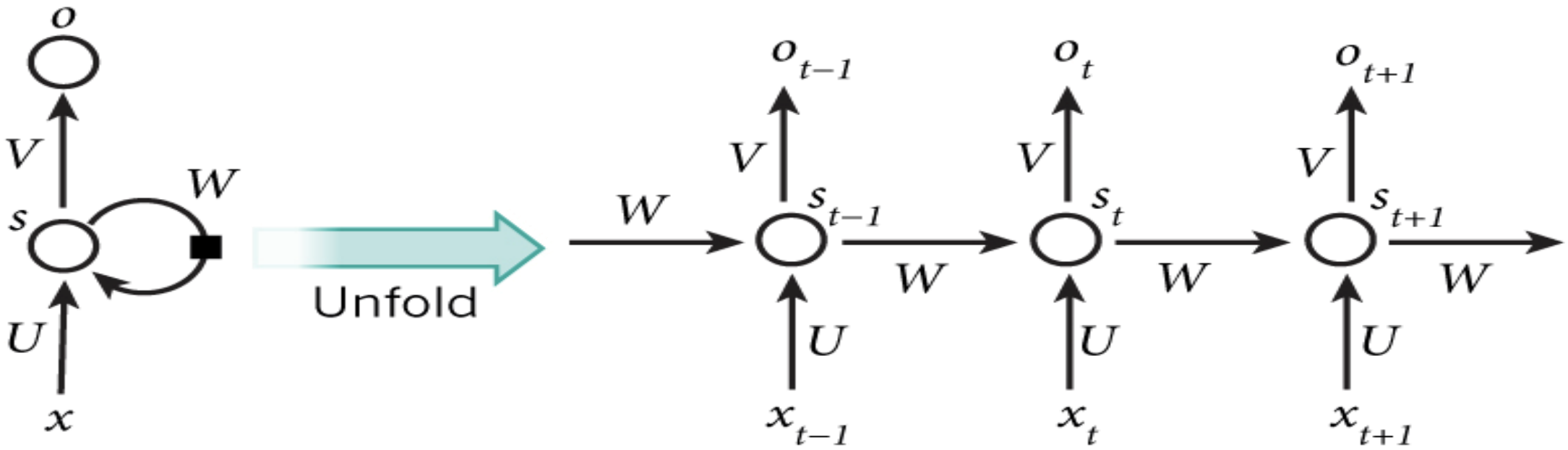
Convolutional Neural Networks for Image and Video Understanding



Deep Learning Architectures

Variants

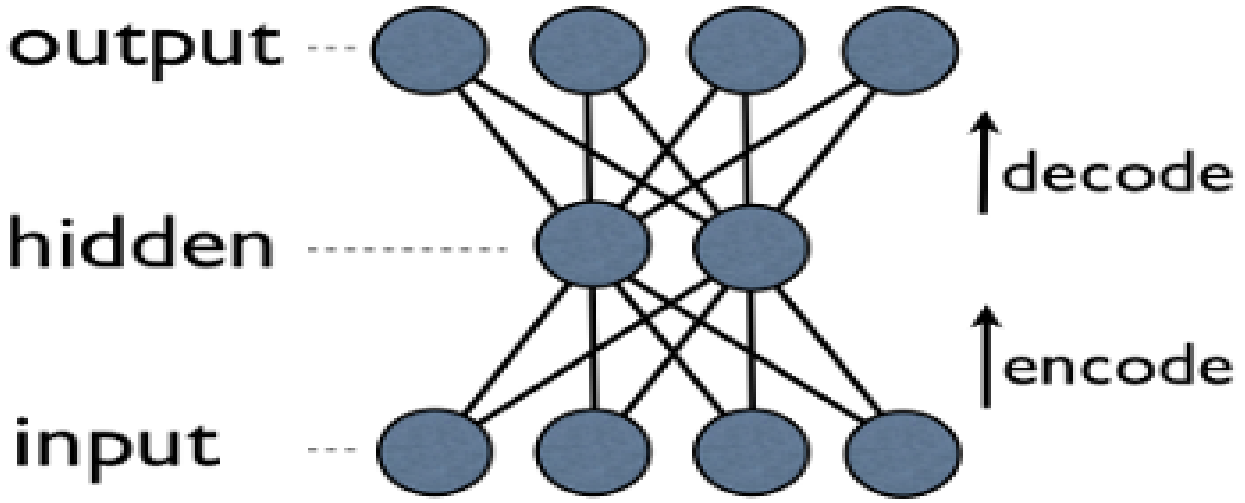
Recurrent Neural Networks for Time Series and Sequence Data Understanding



Deep Learning Architectures

Variants

Deep Autoencoders for Dimensionality Reduction



Deep Learning

Why has use of multiple layers led to success?

- Captures compositionality: world is compositional!
  - **Image recognition:** Pixel → edge → texton → motif → part → object
  - **Text:** Character → word → word group → clause → sentence → story
  - **Speech:** Sample → spectral band → sound → … → phone → phoneme → word
- Exploiting compositionality gives an exponential gain in representational power

Deep Learning

Applications and Successes

- AlexNet (Object Recognition): The network that catapulted the success of deep learning in 2012

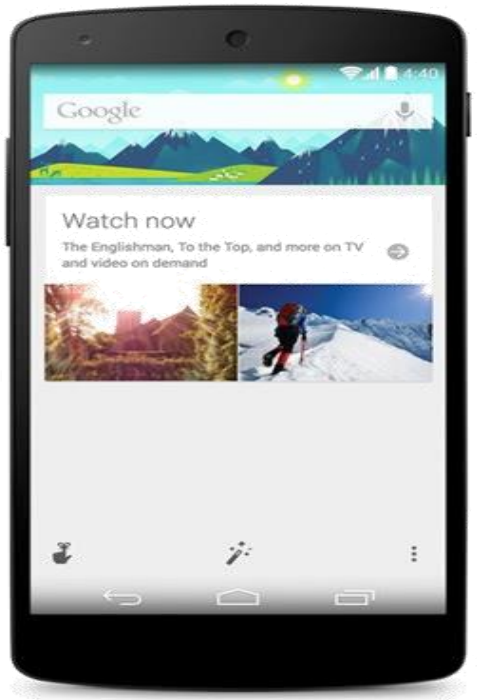
Deep Learning

Applications and Successes

- Speech understanding and natural language processing

Apple Siri

Google Now

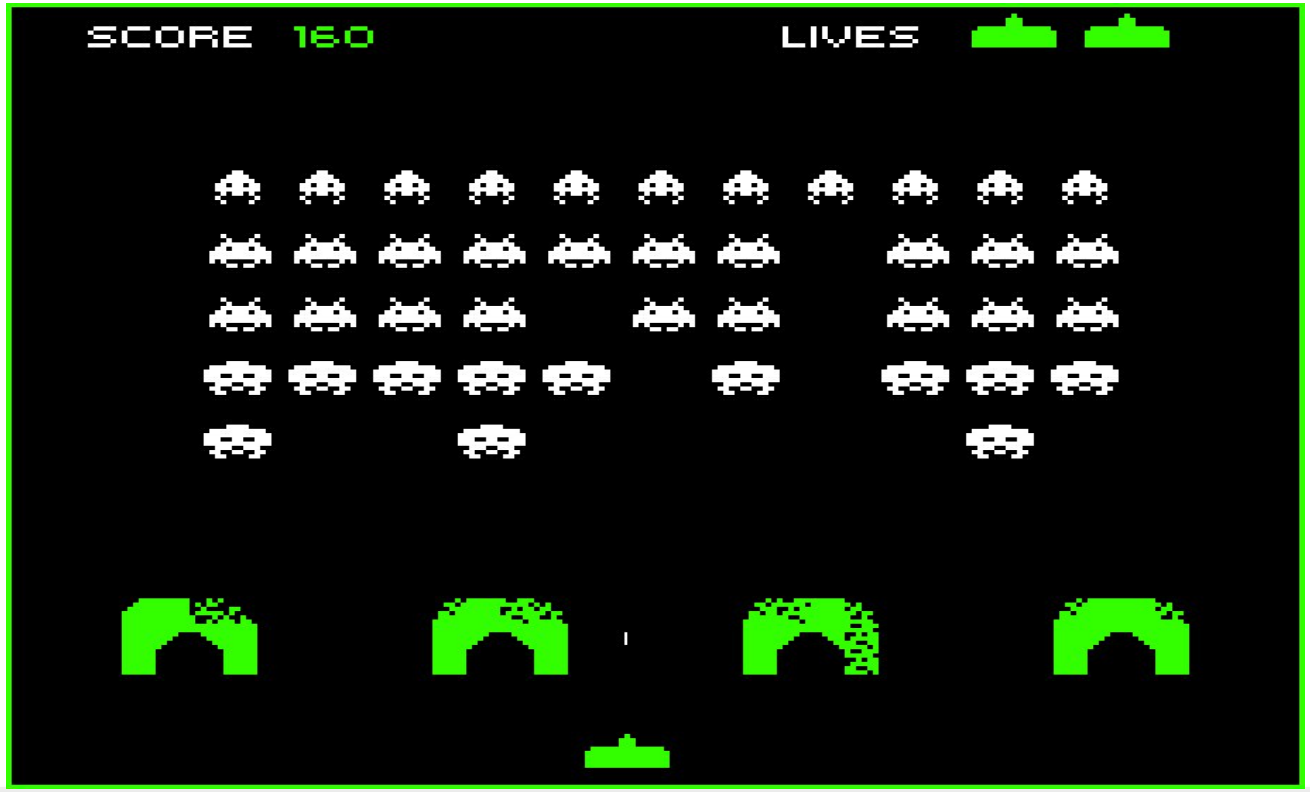


Windows Cortana

Deep Learning

Applications and Successes

- Game playing: https://www.youtube.com/watch?v=V1eYniJ0Rnk



Stepping to higher levels of intelligence

From recognition to generation

- Handwriting generation - http://www.cs.toronto.edu/~graves/handwriting.html
- RNN-generated TED talk (visit TED^{RNN})https://www.youtube.com/watch?v=-OodHtJ1saY
- RNN-generated Obama speech:
  - *Good afternoon. God bless you. The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretcks of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives. Thank you very much. God bless you, and God bless the United States of America.*

Stepping to higher levels of intelligence

From recognition to generation

- RNN trained on Shakespeare:
  - http://cs.stanford.edu/people/karpathy/char-rnn/shakespear.txt
  - https://www.youtube.com/watch?v=Jkkjy7dVdaY

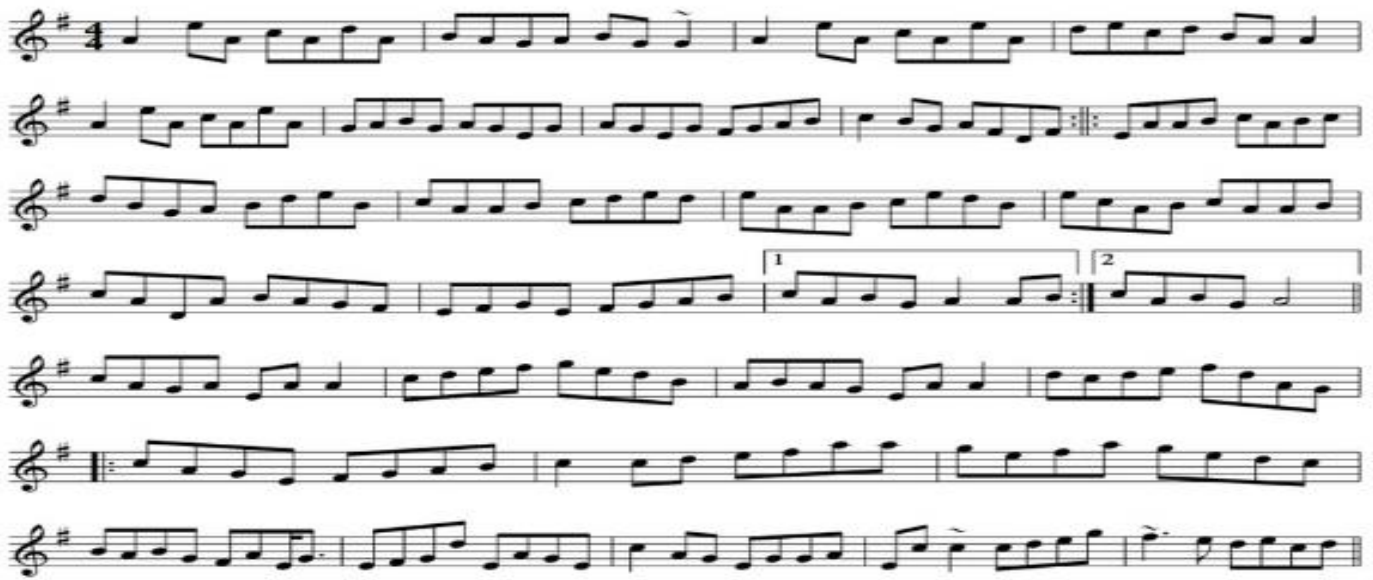
```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Stepping to higher levels of intelligence

From recognition to generation

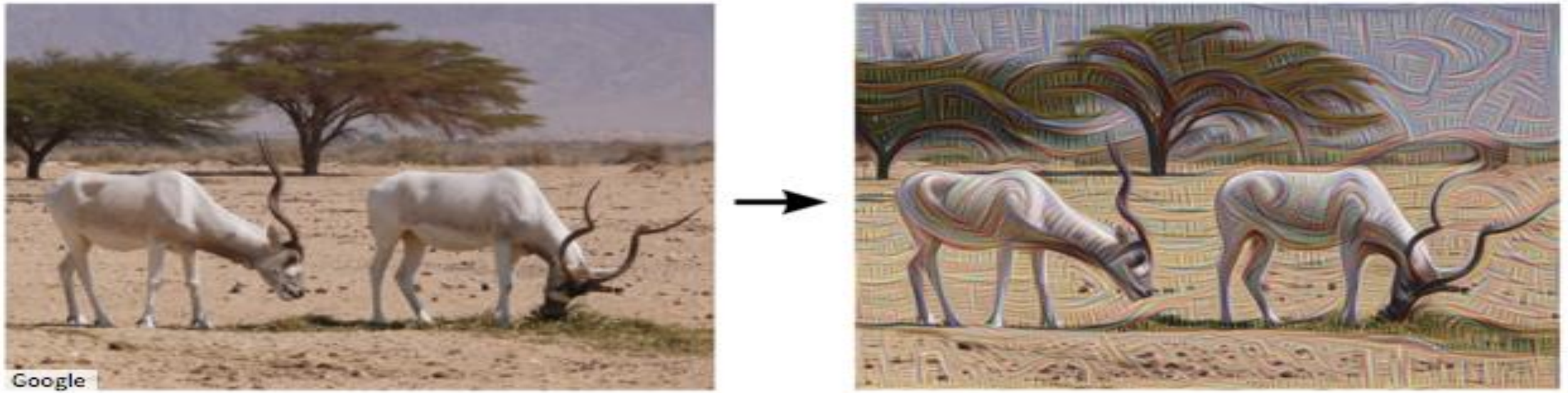
- Music generation
  - Outputs of student course project at IIT-H
    - https://soundcloud.com/juke-bots/bach-200-1
    - https://soundcloud.com/juke-bots/sets/phase-1



Stepping to even higher levels of intelligence

From generation to imagination

- Deep dreaming/hallucination
  - https://www.youtube.com/watch?v=oyxSerkkP4o



Stepping to even higher levels of intelligence

From generation to imagination

Stepping to even higher levels of intelligence

From generation to imagination

- Deep colorization

More?

Where to look?

- One-stop shop
  - https://github.com/ChristosChristofidis/awesome-deep-learning
- Check this out for hours of fun and amazement
  - http://fastml.com/deep-nets-generating-stuff/
- Books (on Deep Learning)
  - http://www.deeplearningbook.org
  - http://neuralnetworksanddeeplearning.com/
- Programming
  - Theano/Pylearn2, Caffe, Torch (used in Google, Facebook and other companies), TensorFlow – recent initiative of Google, Keras

Classification Methods

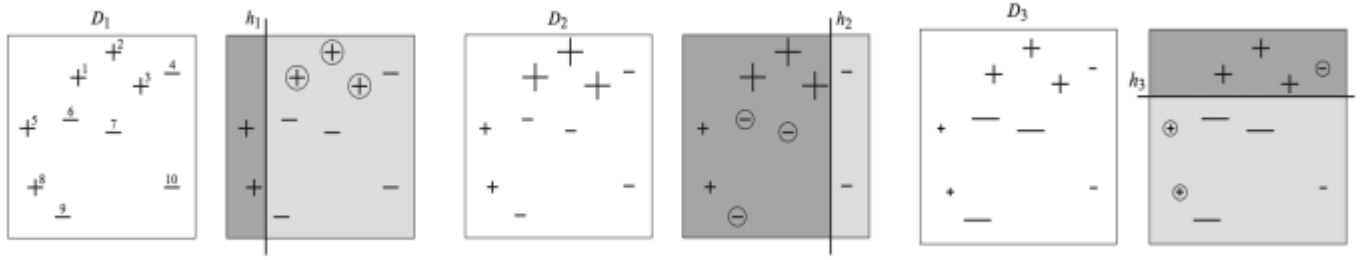
- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression (we will cover this during regression methods)
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

Revisiting Boosting: Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**



- Fit an additive model (ensemble) in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Gradient Boosting, "shortcomings" are identified by gradients.
- Recall that, in Adaboost, "shortcomings" are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.