Deep Learning
(deep neural nets)

Jun Zhu
dcszj@mail.tsinghua.edu.cn
http://bigml.cs.tsinghua.edu.cn/~jun
State Key Lab of Intelligent Technology & Systems
Tsinghua University

March 21, 2017

Why going deep?

- Data are often high-dimensional.

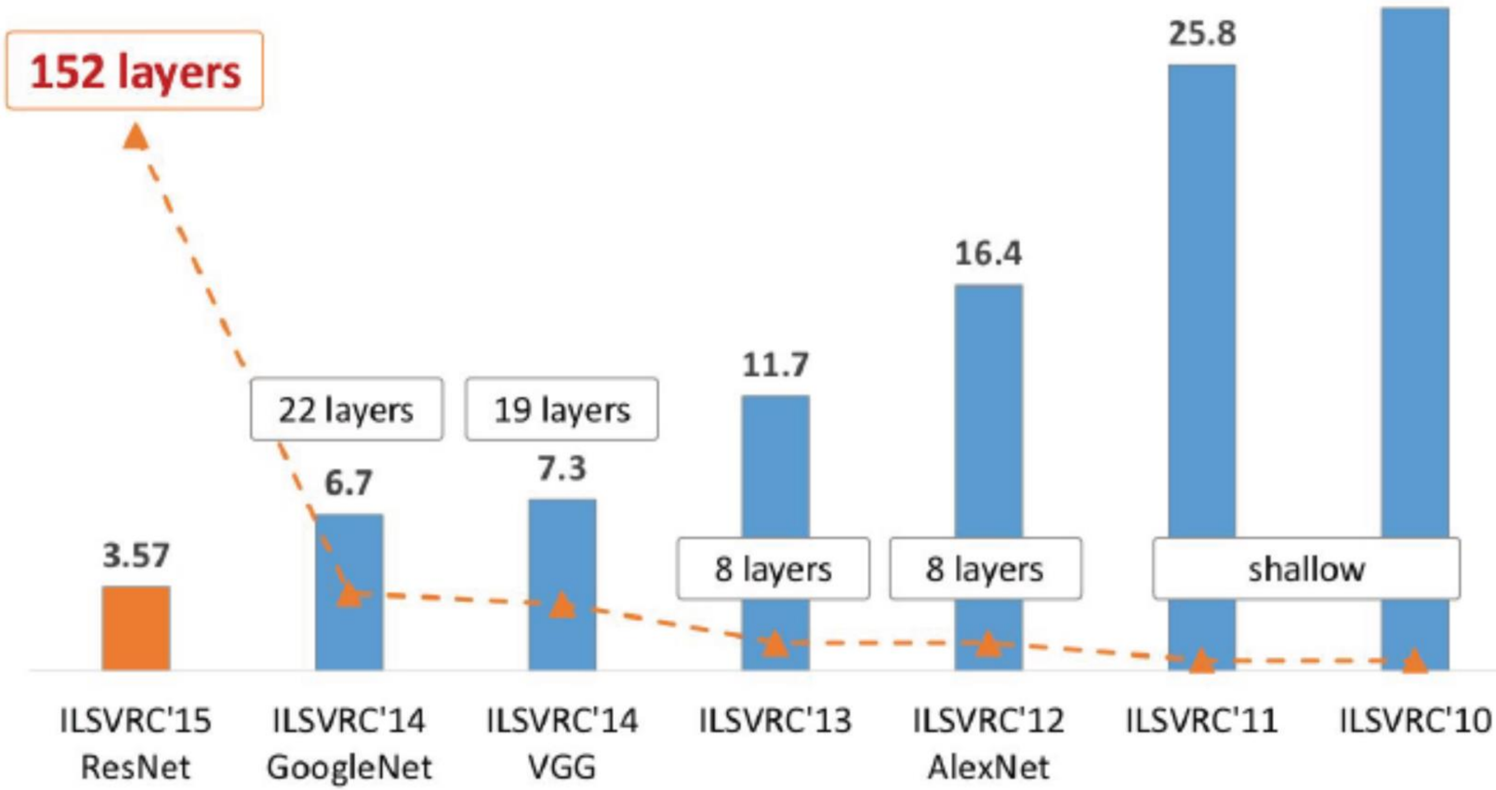
- There is a huge amount of structure in the data, but the structure is too complicated to be represented by a simple model.

- Insufficient depth can require more computational elements than architectures whose depth matches the task.

- Deep nets provide simpler but more descriptive models of many problems.

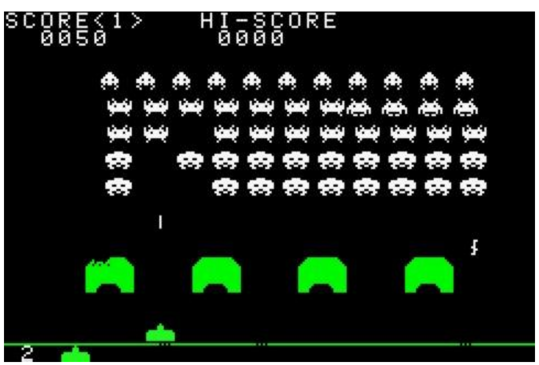
Resolution in Image Classification

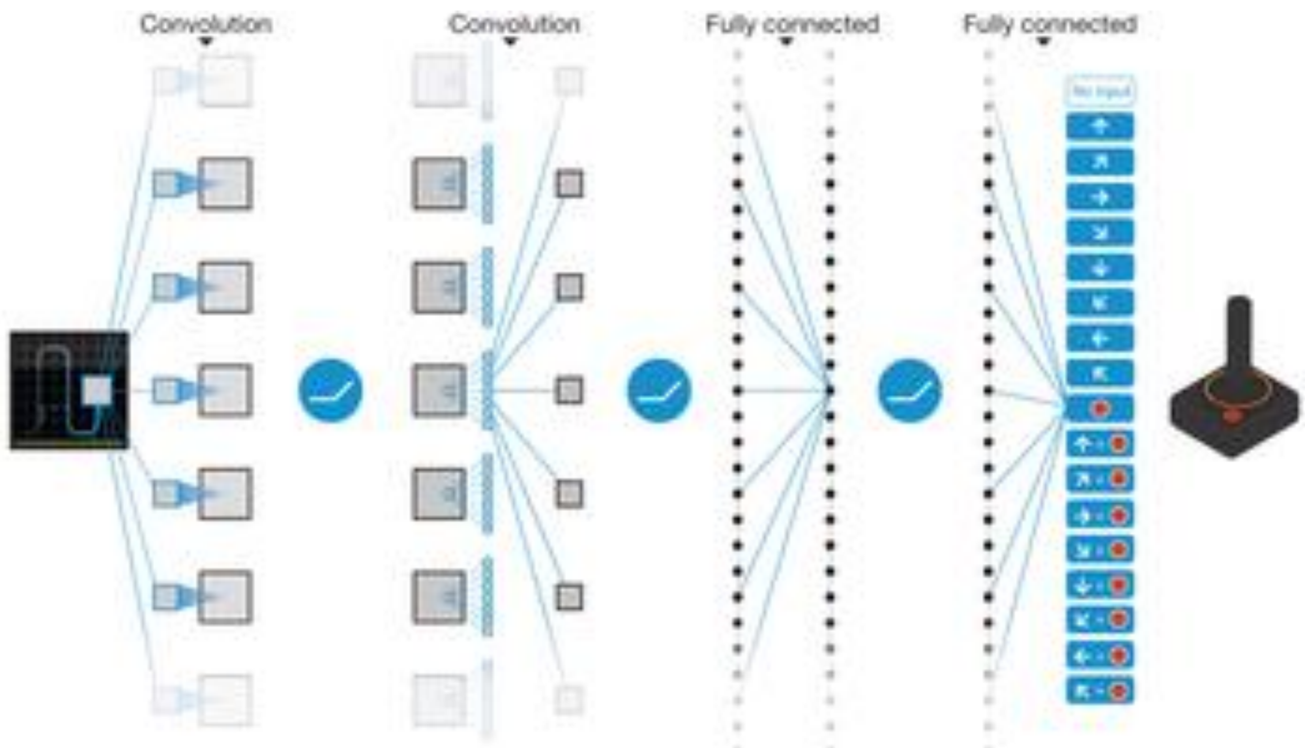
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)



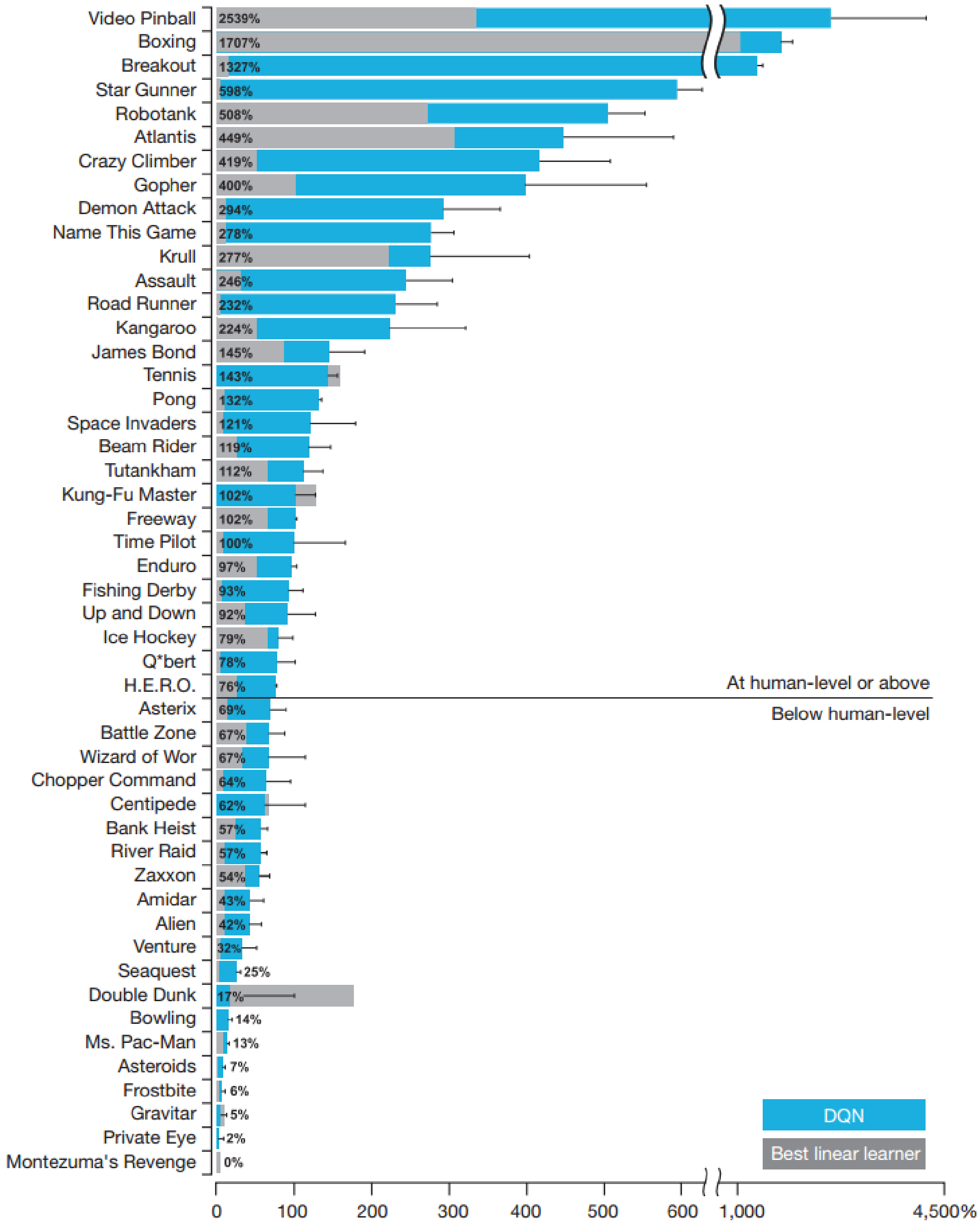
Human-Level Control via Deep RL

- Deep Q-network with human-level performance on Atari games



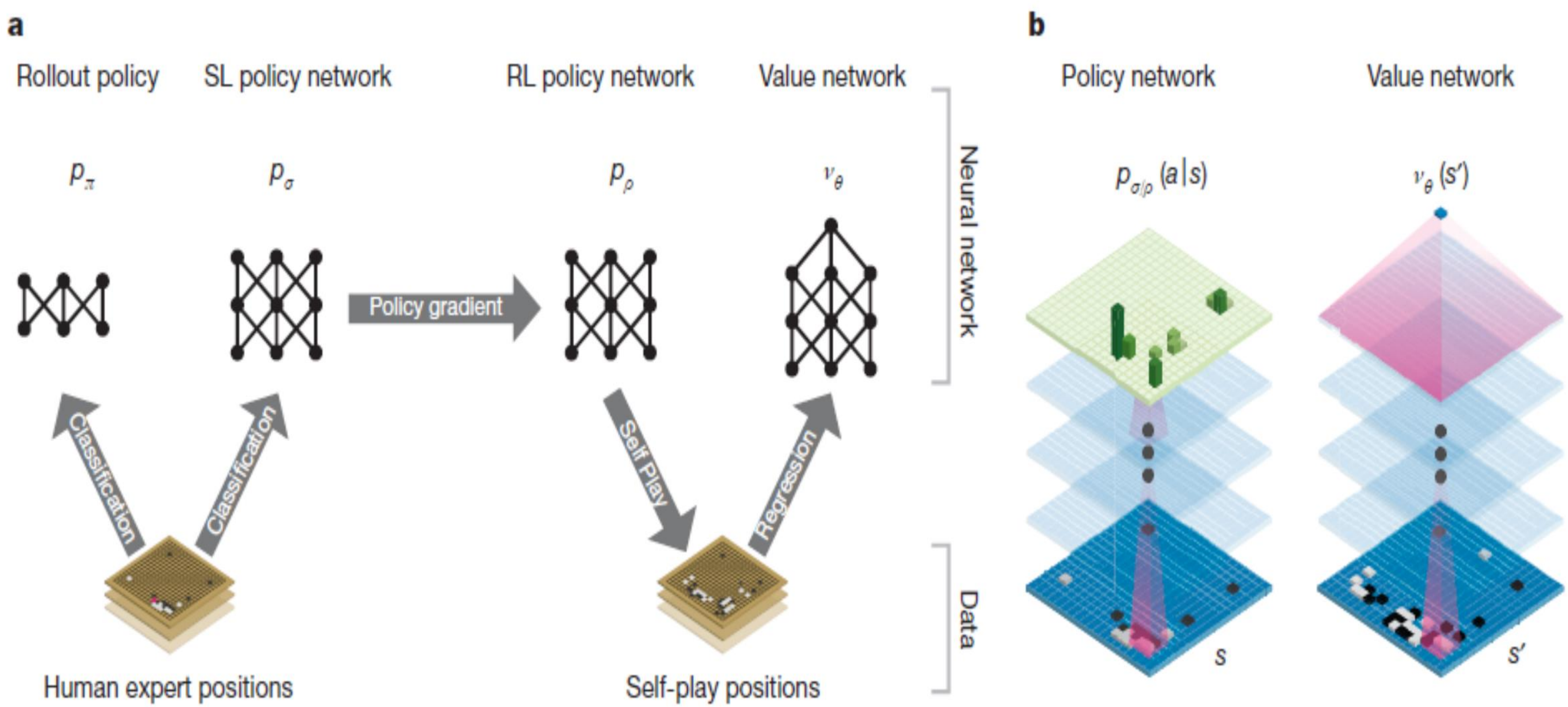


[Mnih et al., Nature 518, 529–533, 2015]


Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
At human-level or above
Below human-level
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

AlphaGo

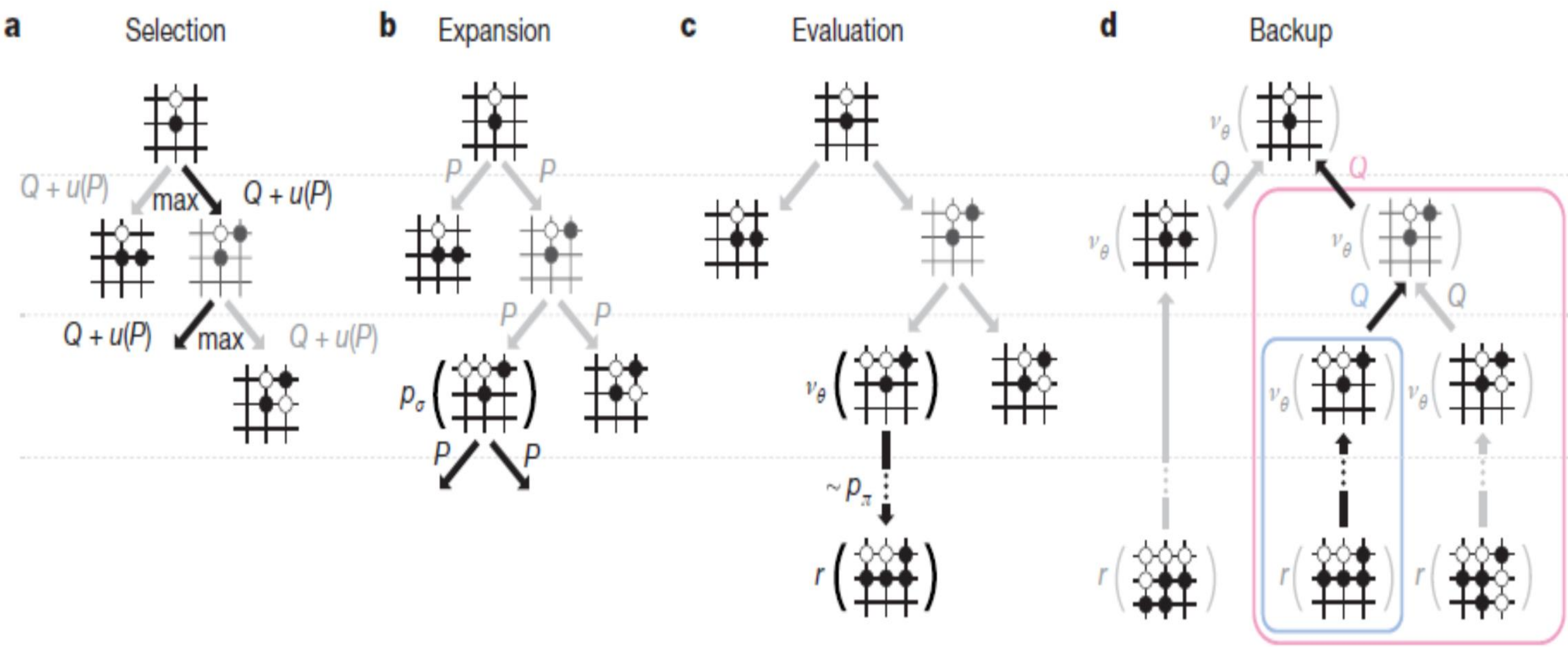
- Neural network training pipeline and architecture



[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

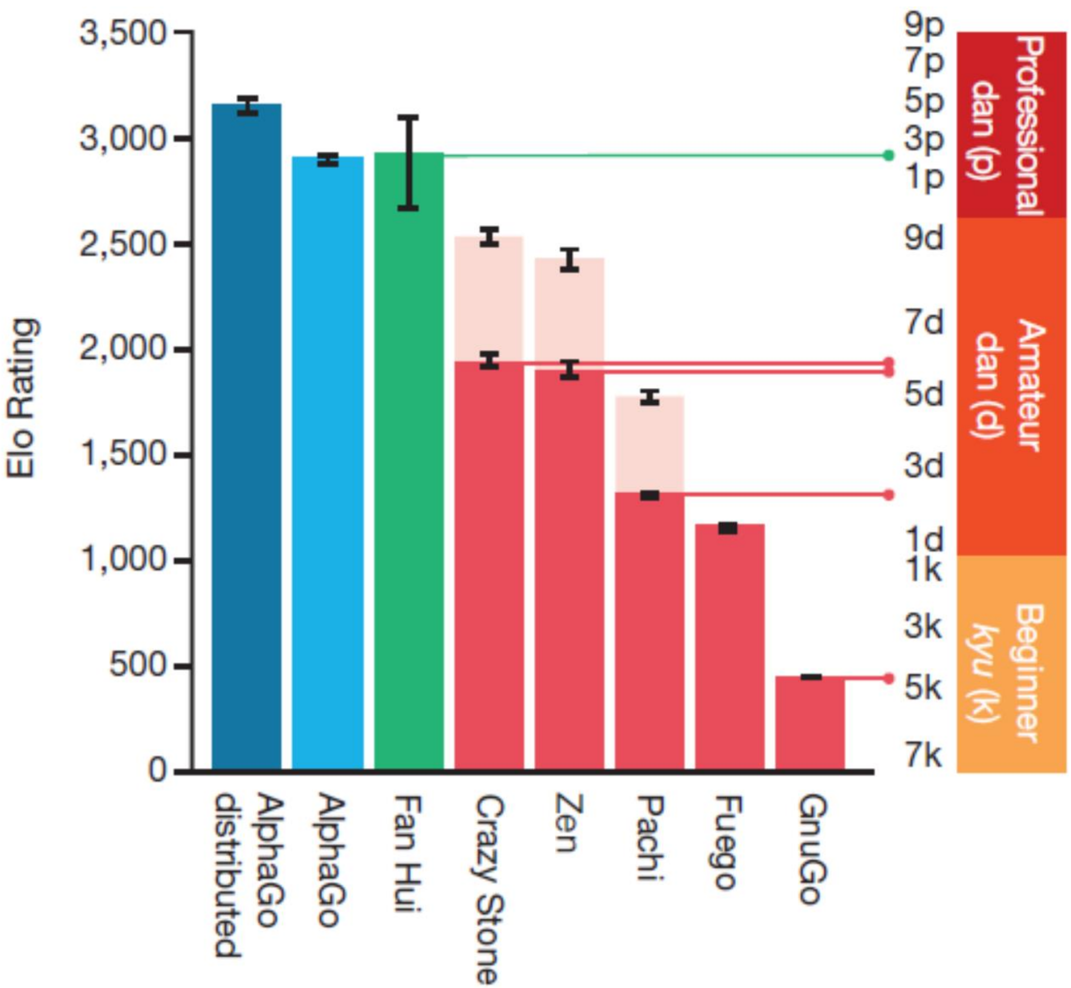
AlphaGo

- Monte Carlo tree search



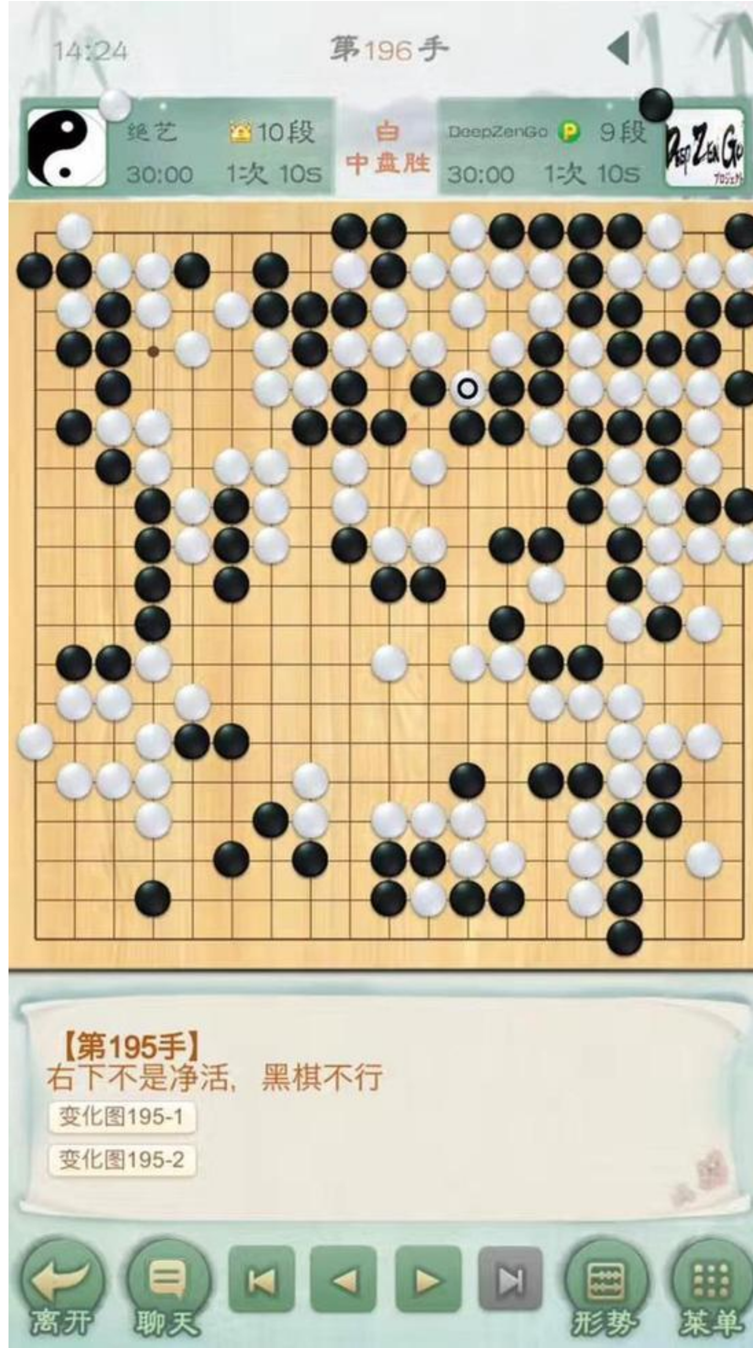
[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

AlphaGo

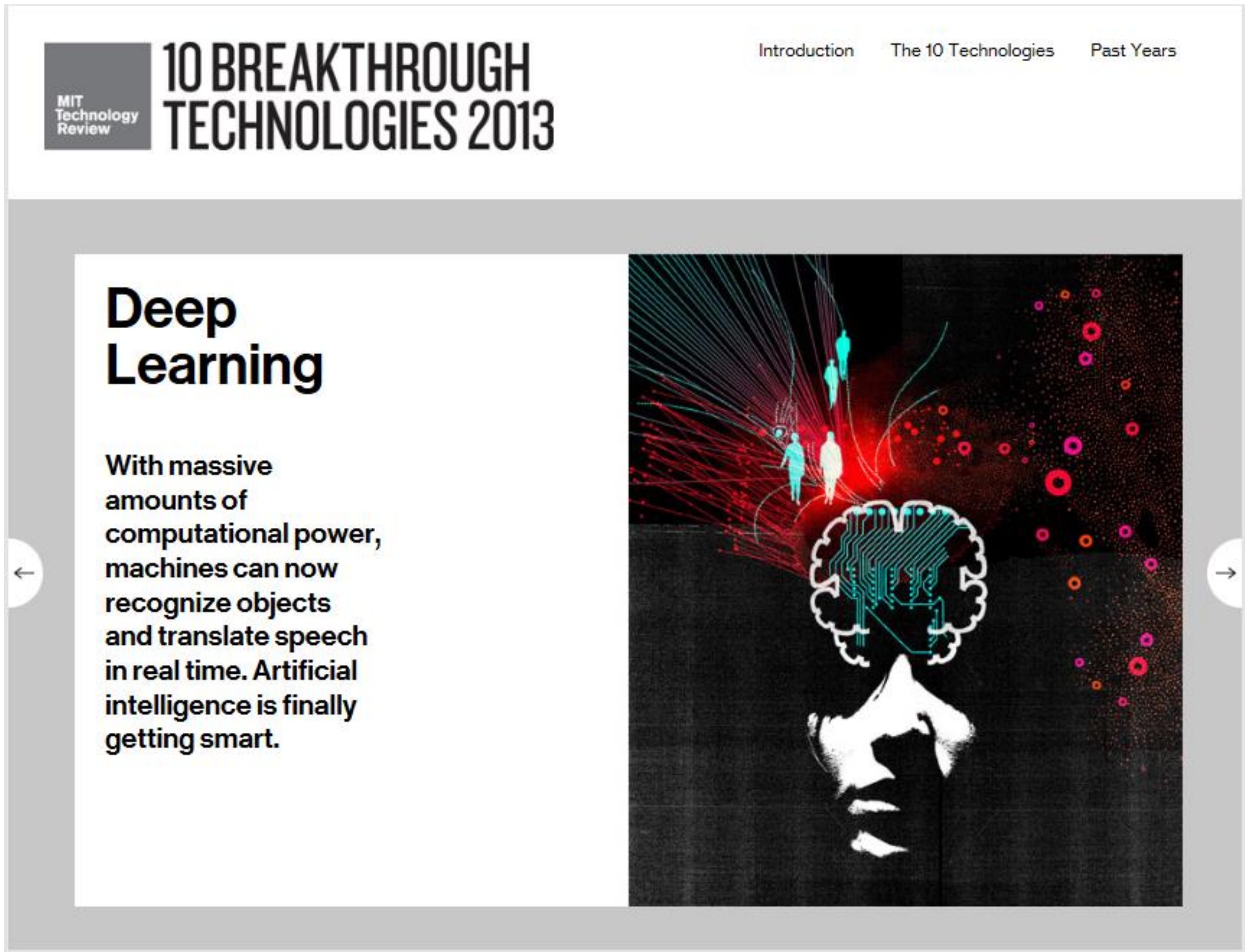


[Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature, 484(529), 2016]

- Tencent FineArt



MIT 10 Breakthrough Tech 2013



http://www.technologyreview.com/featuredstory/513696/deep-learning/

Deep Learning in industry

Driverless car

Face identification

Speech recognition

Web search

…

Google

阿里巴巴 Alibaba.com

HUAWEI

intel

…

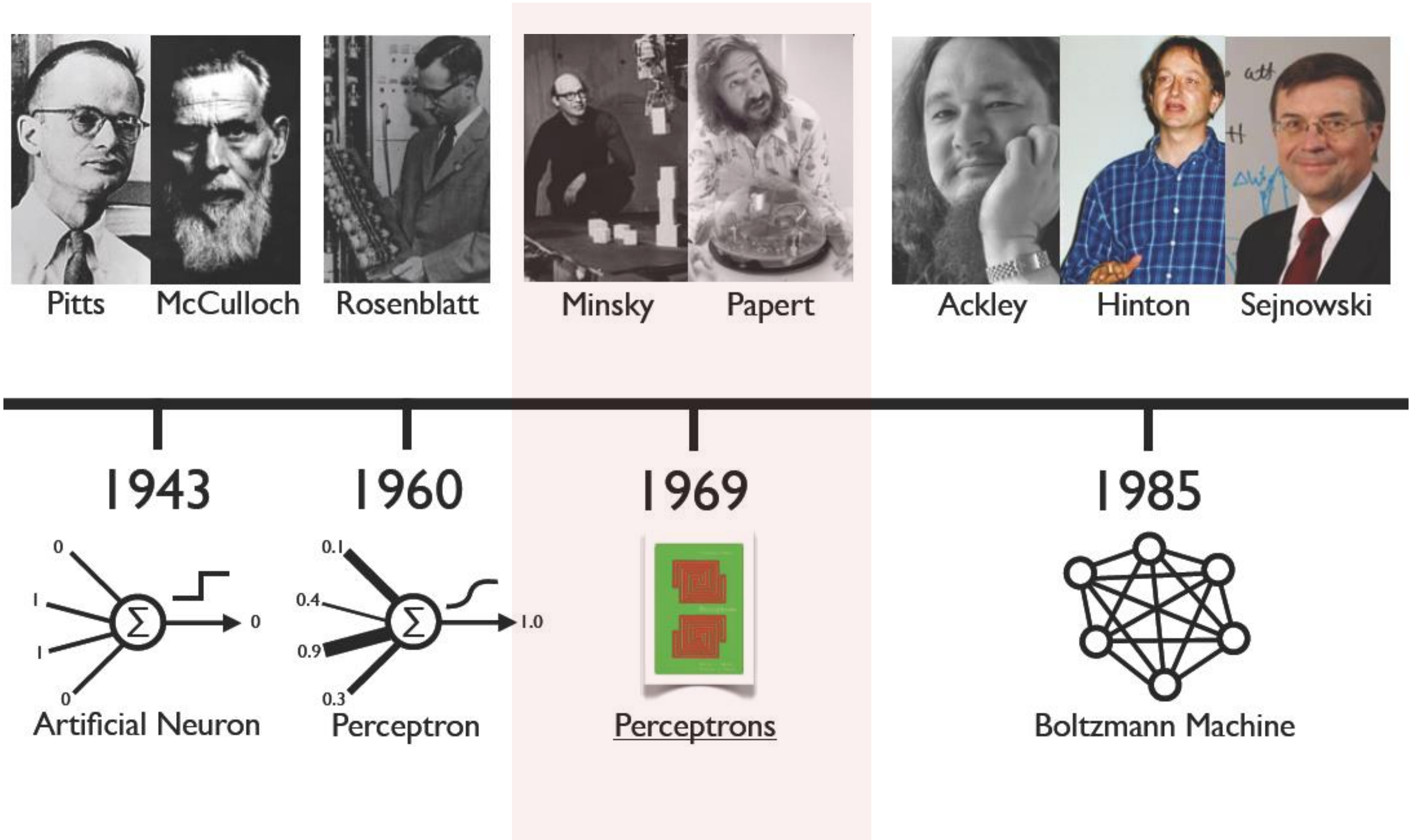
Microsoft

Bai du 百度

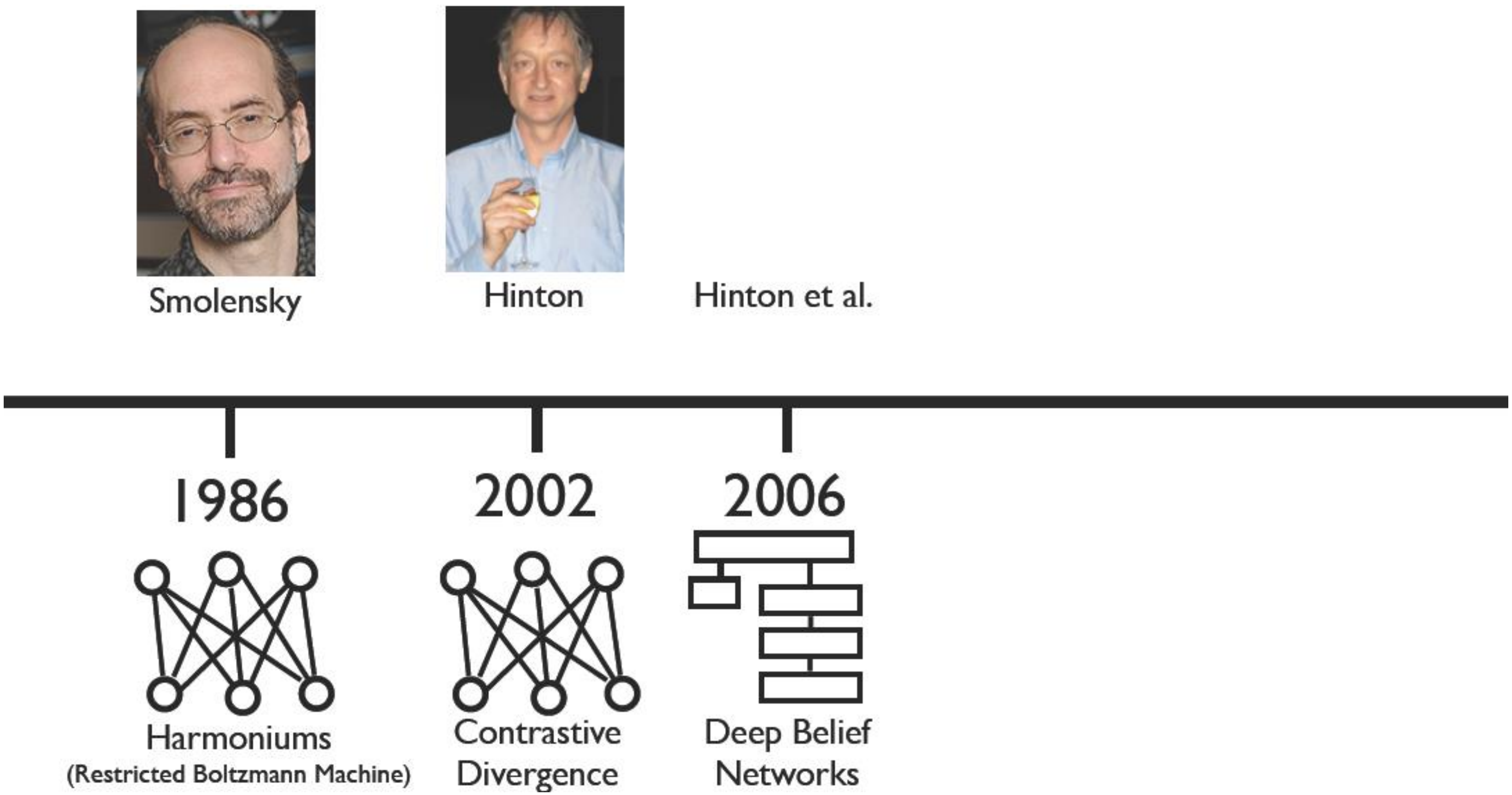
Tencent 腾讯

AMD

History of neural networks

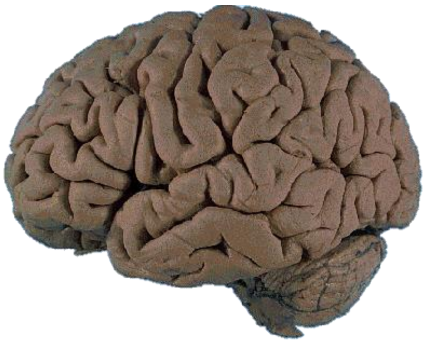


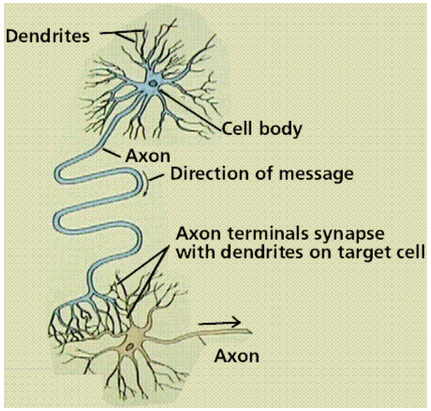
History of neural networks

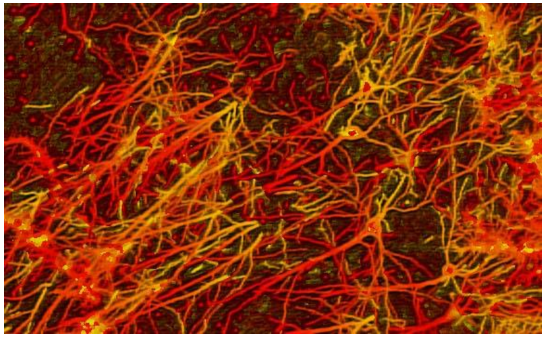


Deep Learning Models

How the human brain learns?







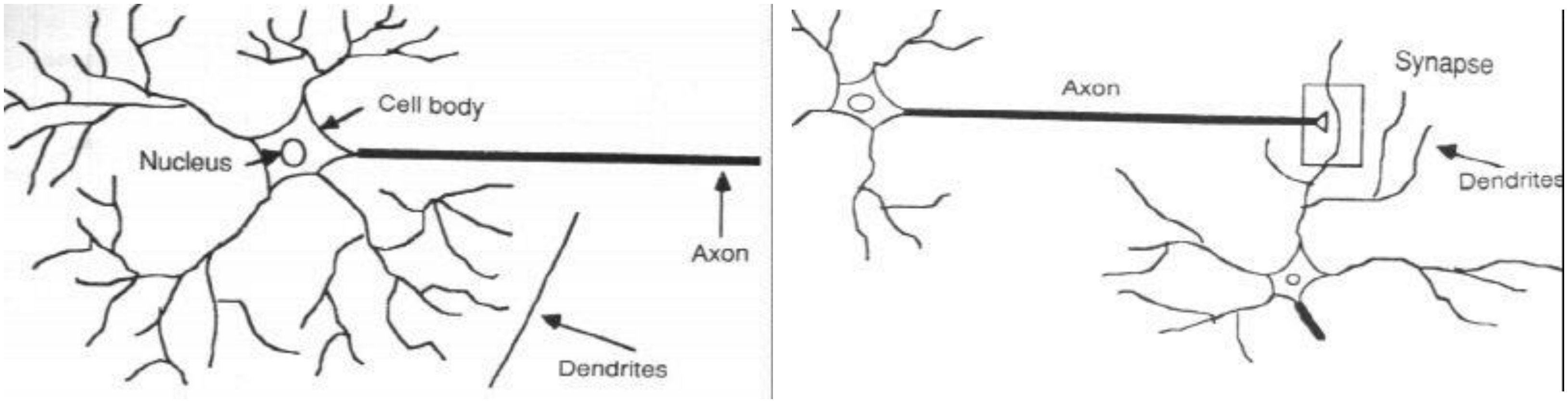
The business end of this is made of lots of these joined in networks like this

Much of our own “computations” are performed in/by this network

Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes

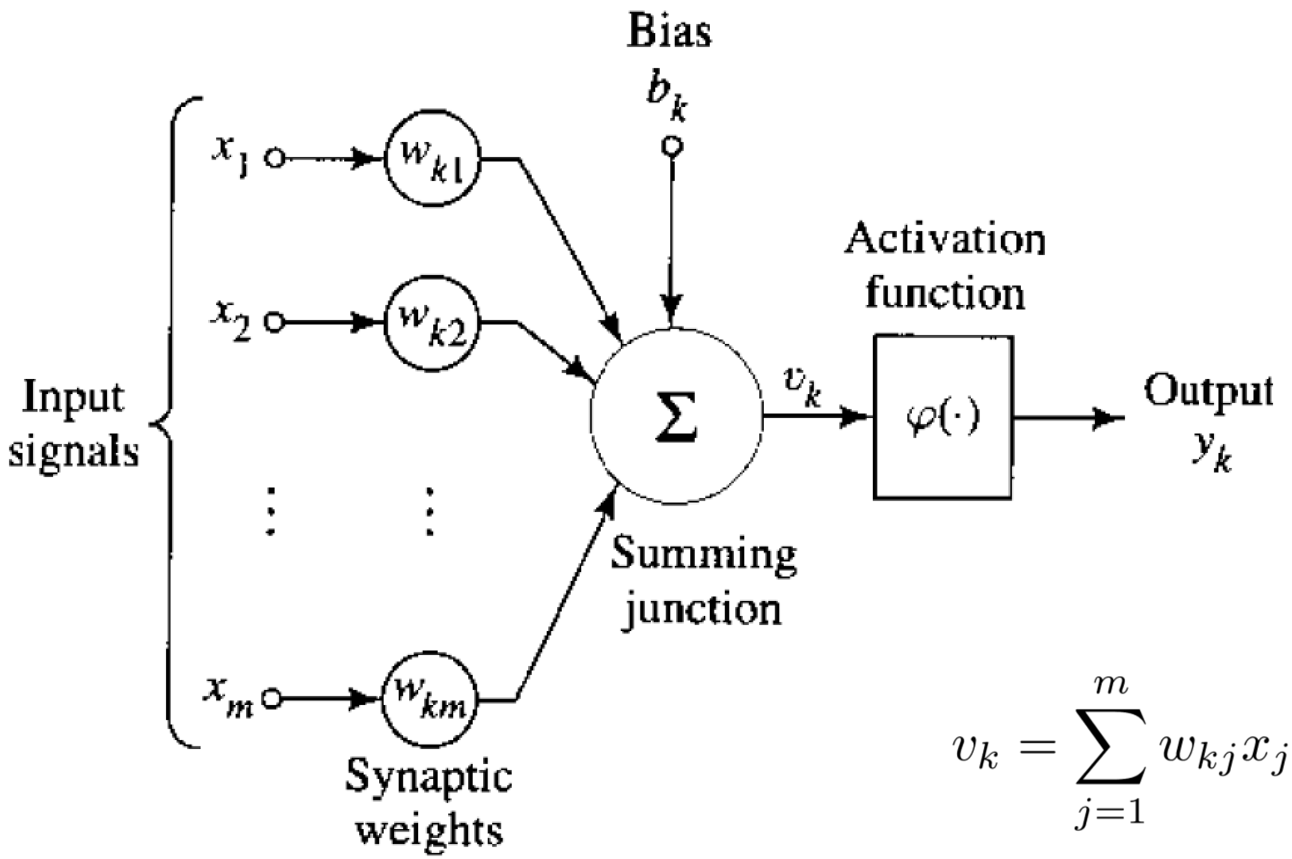
How the human brain learns?

- A typical neuron



- Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes

Model of a neuron

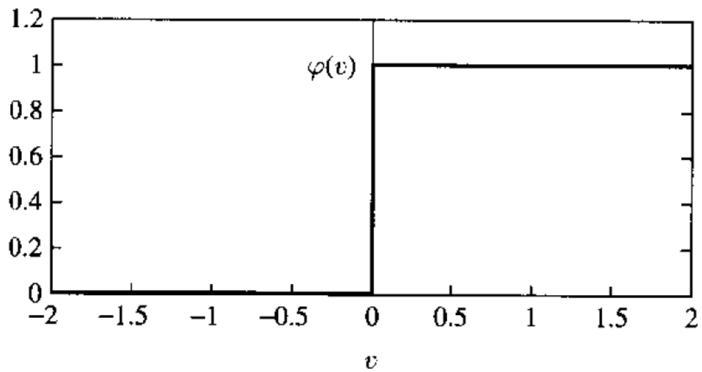


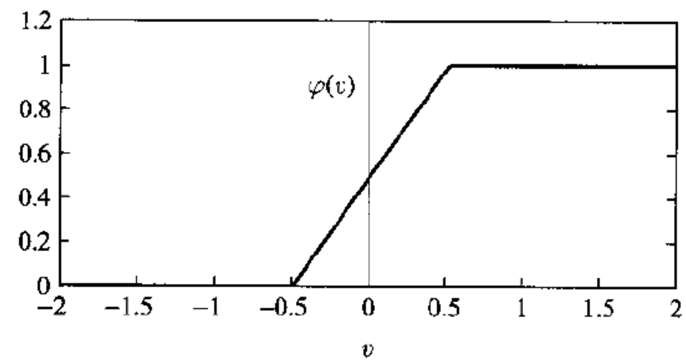
$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$$

$$y_k = \psi(v_k)$$

Activation function

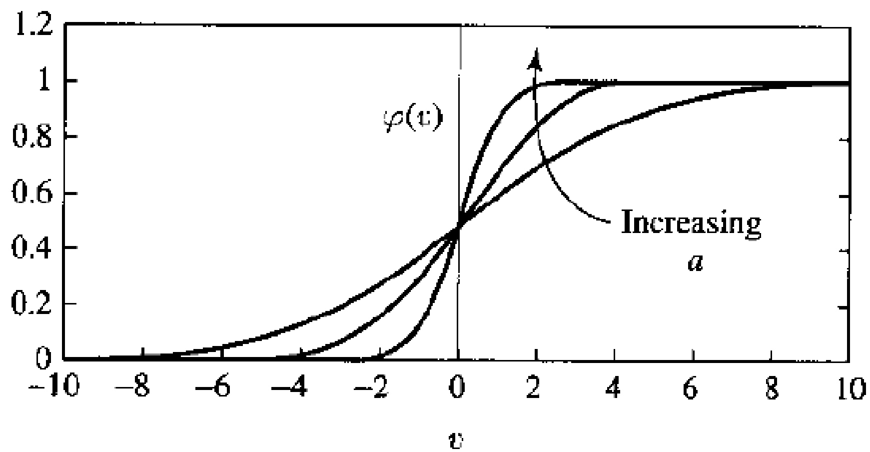
- Threshold function & piecewise linear function:





- Sigmoid function

$$\psi_\alpha(v) = \frac{1}{1 + \exp(-\alpha v)}$$



$a \to \infty$: step function

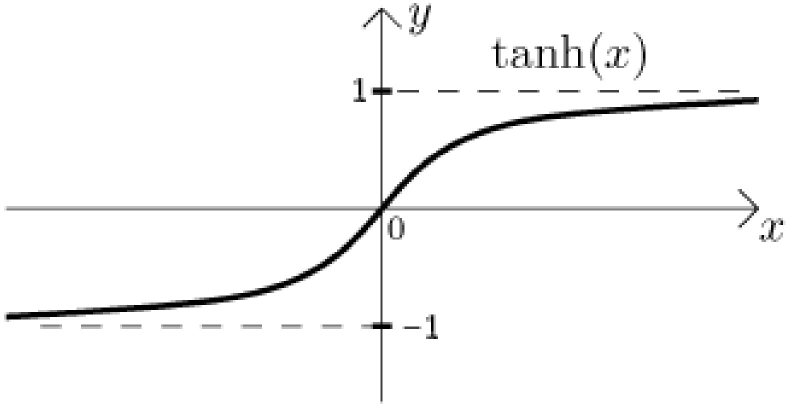
Activation function with negative values

- Threshold function & piecewise linear function:

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

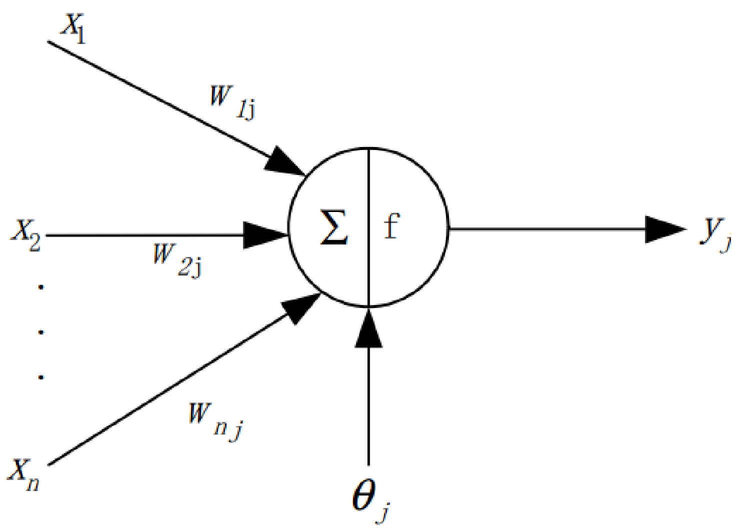
- Hyperbolic tangent function

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



McCulloch & Pitts's Artificial Neuron

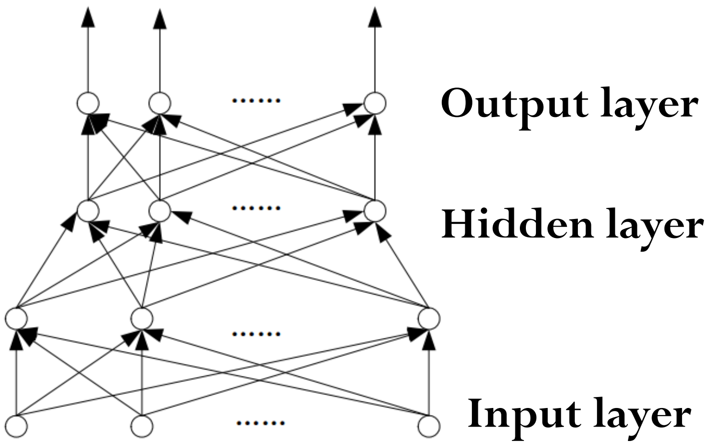
- The first model of artificial neurons in 1943
  - Activation function: a threshold function



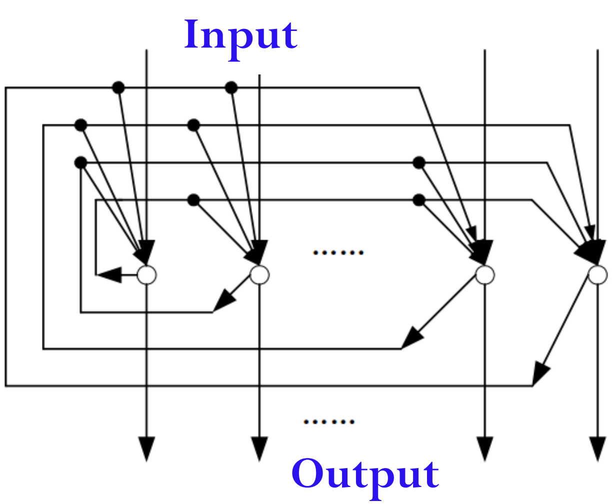
$$y_j = \mathrm{sgn}\left(\sum_i w_{ij} x_i - \theta_j\right)$$

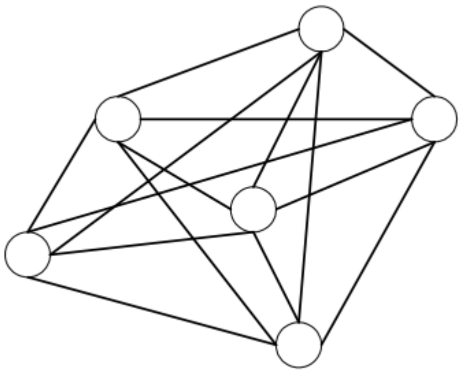
Network Architecture

- Feedforward networks



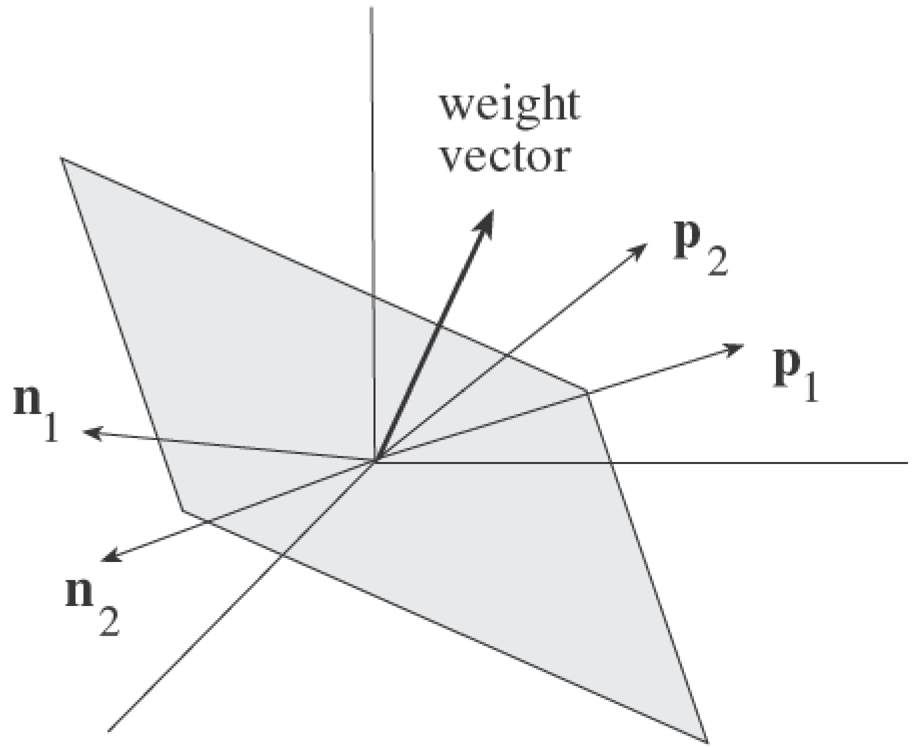
- Recurrent networks





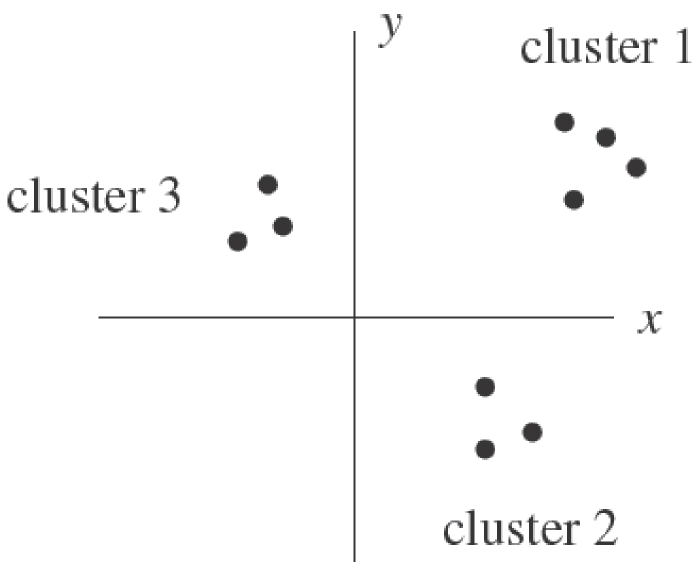
Learning Paradigms

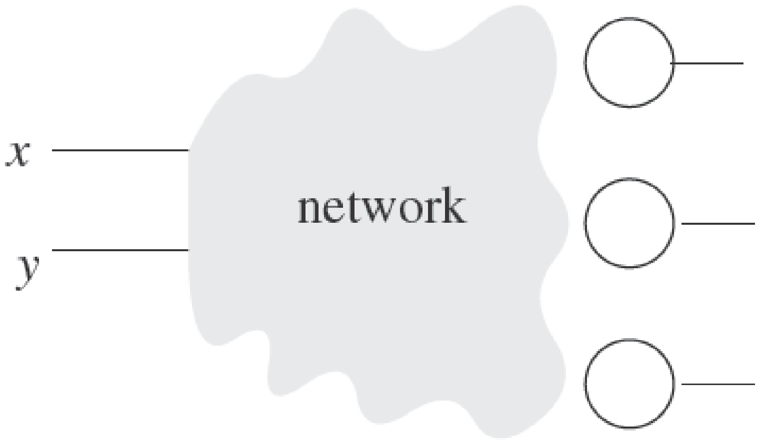
- Supervised Learning (learning with a teacher)
  - For example, classification: learns a separation plane



Learning Paradigms

- Unsupervised learning (learning without a teacher)
  - Example: clustering



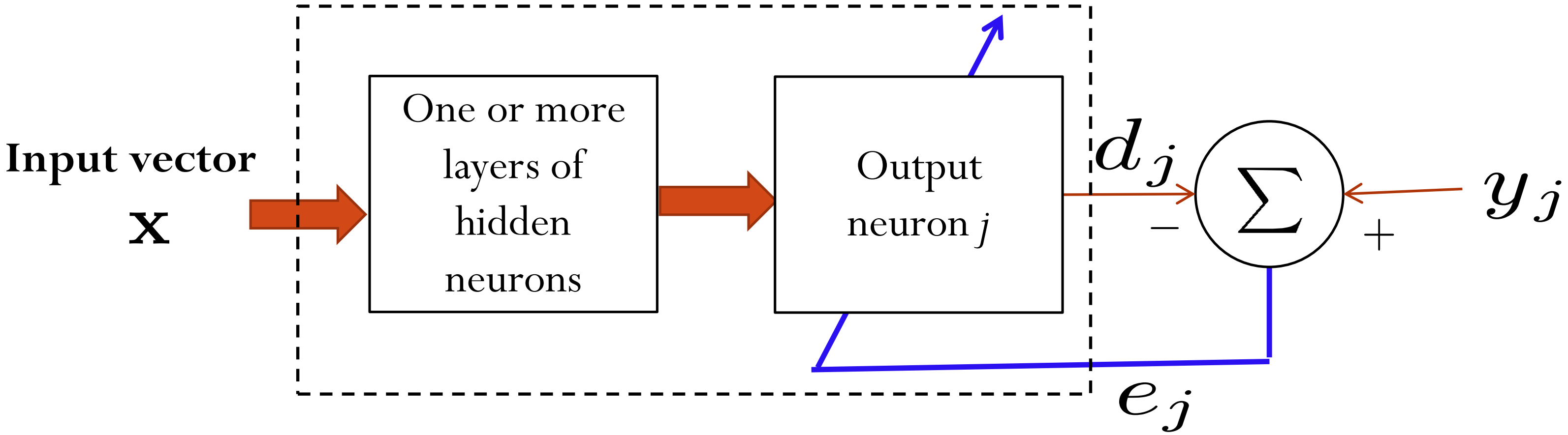


Learning Rules

- Error-correction learning
- Competitive learning
- Hebbian learning
- Boltzmann learning
- Memory-based learning
    - Nearest neighbor, radial-basis function network

Error-correction learning

- The generic paradigm:



- Error signal:

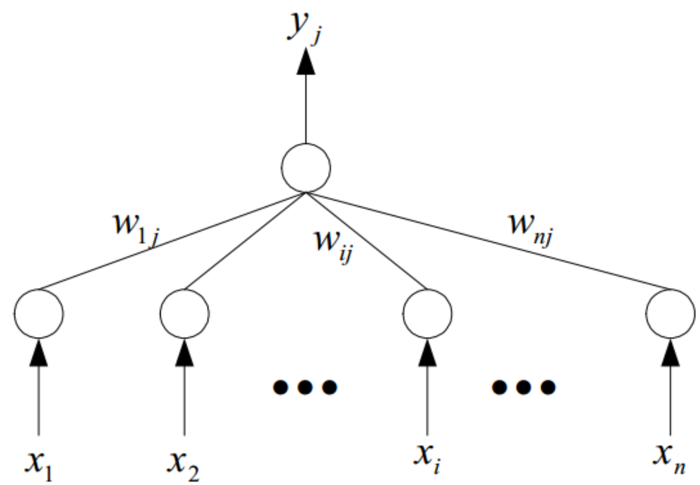
$$e_j = y_j - d_j$$

- Learning objective:

$$\min_{\mathbf{w}} R(\mathbf{w}; \mathbf{x}) := \frac{1}{2} \sum_j e_j^2$$

Example: Perceptron

- One-layer feedforward network based on error-correction learning (no hidden layer):



  - Current output (at iteration t):

$$d_j = (\mathbf{w}_t^j)^\top \mathbf{x}$$

  - Update rule (*exercise?*):

$$\mathbf{w}_{t+1}^j = \mathbf{w}_t^j + \eta(y_j - d_j)\mathbf{x}$$

Perceptron for classification

- Consider a single output neuron
- Binary labels:

$$y \in \{+1, -1\}$$

- Output function:

$$d = \mathrm{sgn}\left(\mathbf{w}_t^\top \mathbf{x}\right)$$

- Apply the error-correction learning rule, we get … (next slide)

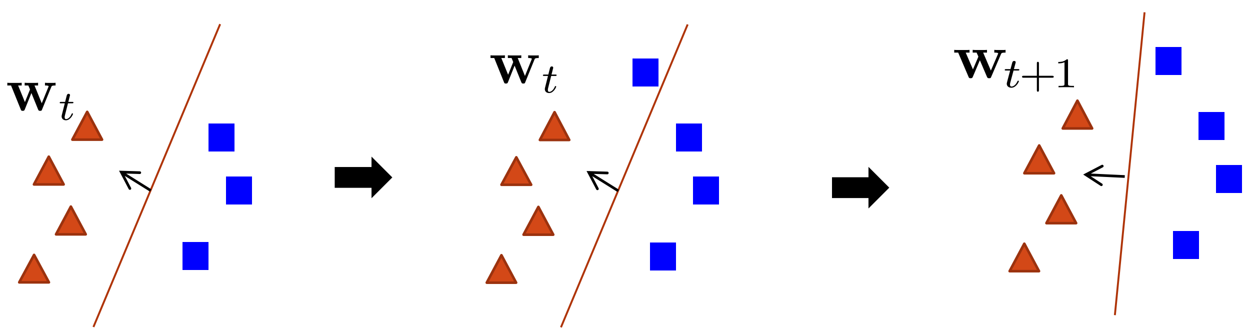
Perceptron for Classification

- Set $\mathbf{w}_1 = 0$ and t=1; scale all examples to have length 1 (doesn't affect which side of the plane they are on)
- Given example x, predict positive *iff*

$$\mathbf{w}_t^\top \mathbf{x} > 0$$

- If a mistake, update as follows
  - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathbf{x}$
  - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{x}$

$$t \leftarrow t+1$$



Convergence Theorem

- For linearly separable case, the perceptron algorithm will converge in a finite number of steps

Mistake Bound

- Theorem:
  - *Let* $\mathcal{S}$ *be a sequence of labeled examples consistent with a linear threshold function* $\mathbf{w}_*^\top \mathbf{x} > 0$ *, where* $\mathbf{w}_*$ *is a unit-length vector.*
  - *The number of mistakes made by the online Perceptron algorithm is at most* $(1/\gamma)^2$*, where*

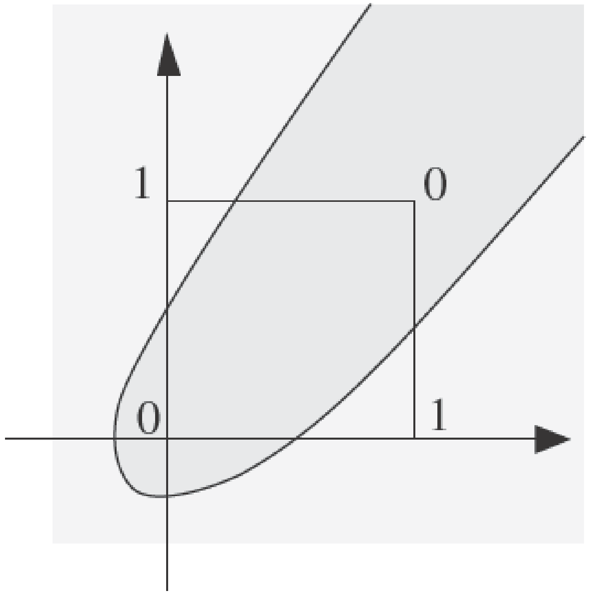
$$\gamma = \min_{\mathbf{x} \in \mathcal{S}} \frac{|\mathbf{w}_*^\top \mathbf{x}|}{\|\mathbf{x}\|}$$

  - *i.e.: if we scale examples to have length 1, then* γ *is the minimum distance of any example to the plane* $\mathbf{w}_*^\top \mathbf{x} = 0$
  - γ *is often called the "margin" of* $\mathbf{w}_*$ *; the quantity* $\frac{\mathbf{w}_*^\top \mathbf{x}}{\|\mathbf{x}\|}$ *is the cosine of the angle between* $\mathbf{x}$ *and* $\mathbf{w}_*$

Deep Nets

- Deep neural networks
  - Multi-layer Perceptron
  - CNN
  - Deep recurrent nets
- Deep generative models
  - Auto-encoder
  - RBM
  - Deep belief nets

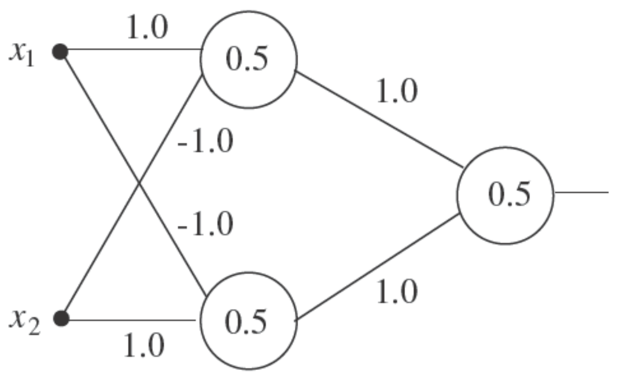
XOR Problem



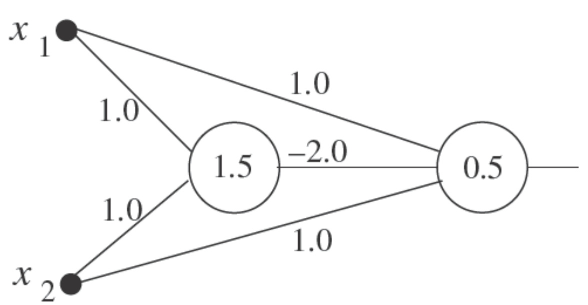
◆ Single-layer perceptron can't solve the problem

XOR Problem

- A network with 1-layer of 2 neurons works for XOR:
  - threshold activation function



  - Many alternative networks exist (not layered)



Multilayer Perceptrons

- Computatonal limitations of single-layer Perceptron by Minsky & Papert (1969)

- Multilayer Perceptrons:
  - Multilayer feedforward networks with an error-correction learning algorithm, known as error ***back-propagation***

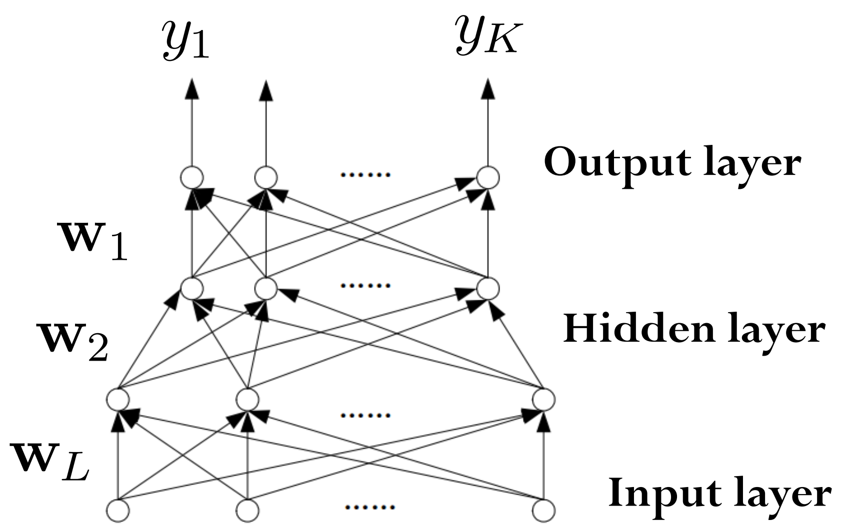
  - A generalization of single-layer percetron to allow nonlinearity

Backpropagation

- Learning as loss minimization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_j e_j^2(\mathbf{x})$$

$$e_j = y_j - d_j$$



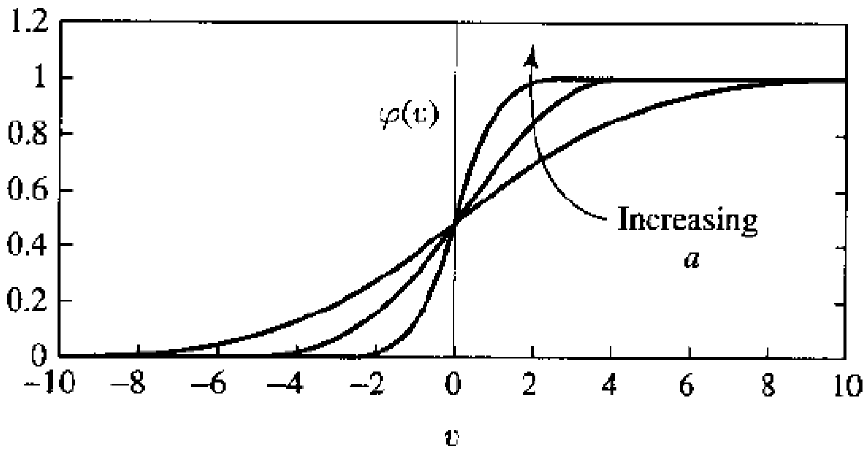
- Learning with gradient descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda_t \nabla R(\mathbf{w}; \mathcal{D})$$

Backpropagation

- Step function in perceptrons is non-differentiable
- Differentiable activation functions are needed to calculate gradients, e.g., sigmoid:

$$\psi_\alpha(v) = \frac{1}{1 + \exp(-\alpha v)}$$



Backpropagation

- Derivative of a sigmoid function ($\alpha = 1$)

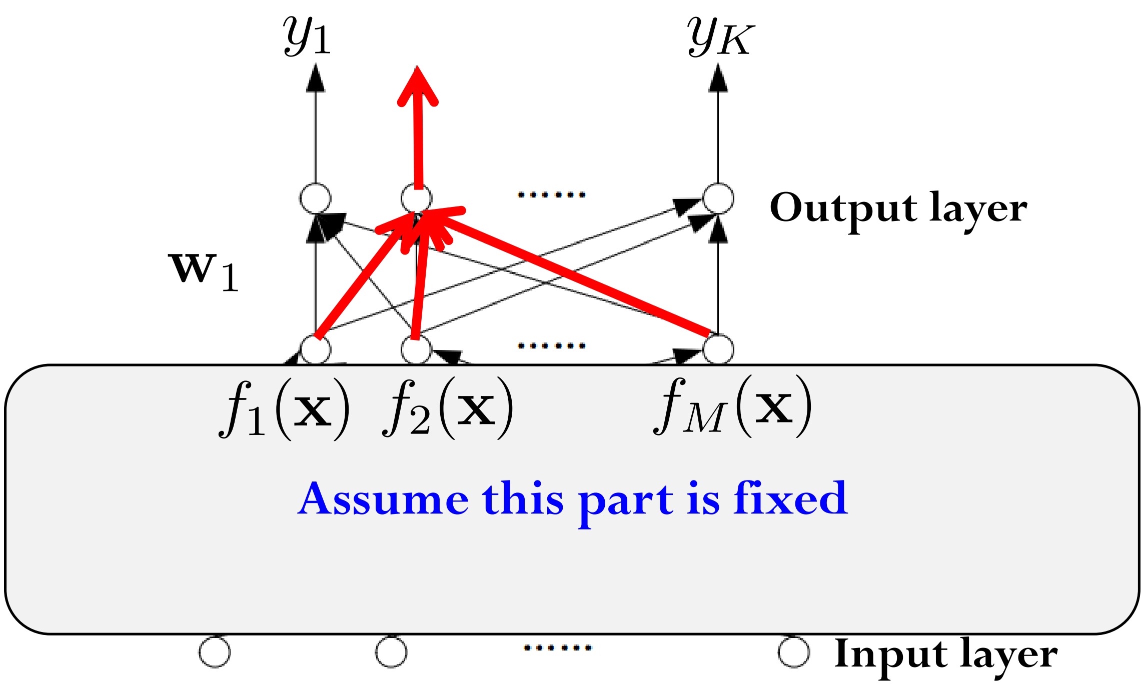
$$\nabla_v \psi(v) = \frac{e^{-v}}{(1+e^{-v})^2} = \psi(v)(1-\psi(v))$$

  - Note about the small scale of the gradient
  - Gradient vanishing issue

- Many other activation functions examined

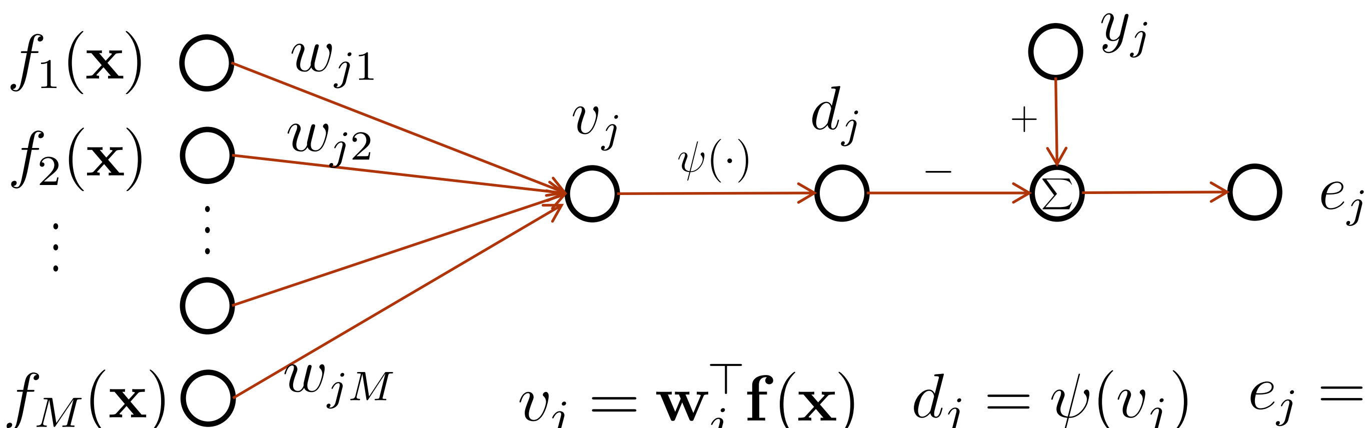
Gradient computation at output layer

- Output neurons are separate:



Gradient computation at output layer

- Signal flow:



$$v_j = \mathbf{w}_j^\top \mathbf{f}(\mathbf{x}) \quad d_j = \psi(v_j) \quad e_j = y_j - d_j$$

$$R_j = \frac{1}{2} e_j^2$$

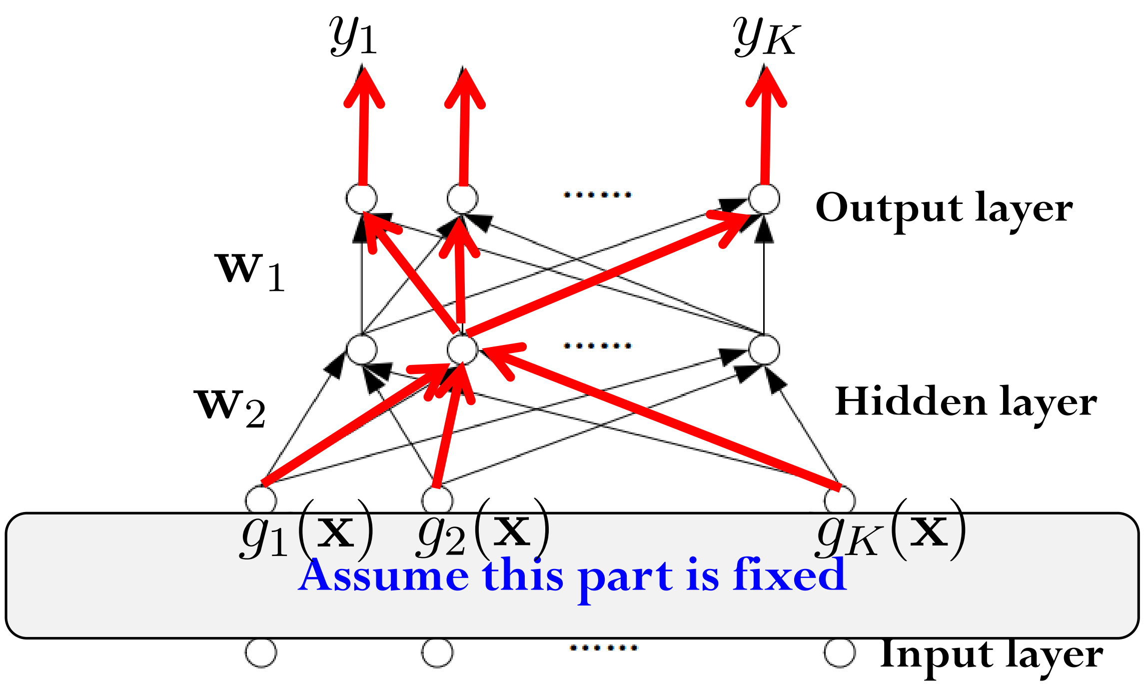
$$R = \frac{1}{2} \sum_j e_j^2$$

$$\begin{aligned} \nabla_{w_{ji}} R &= \frac{\partial R_j}{\partial e_j} \frac{\partial e_j}{\partial d_j} \frac{\partial d_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ji}} \\ &= e_j \cdot (-1) \cdot \psi'(v_j) \cdot f_i(\mathbf{x}) \\ &= -\boxed{e_j \psi'(v_j)} f_i(\mathbf{x}) \end{aligned}$$

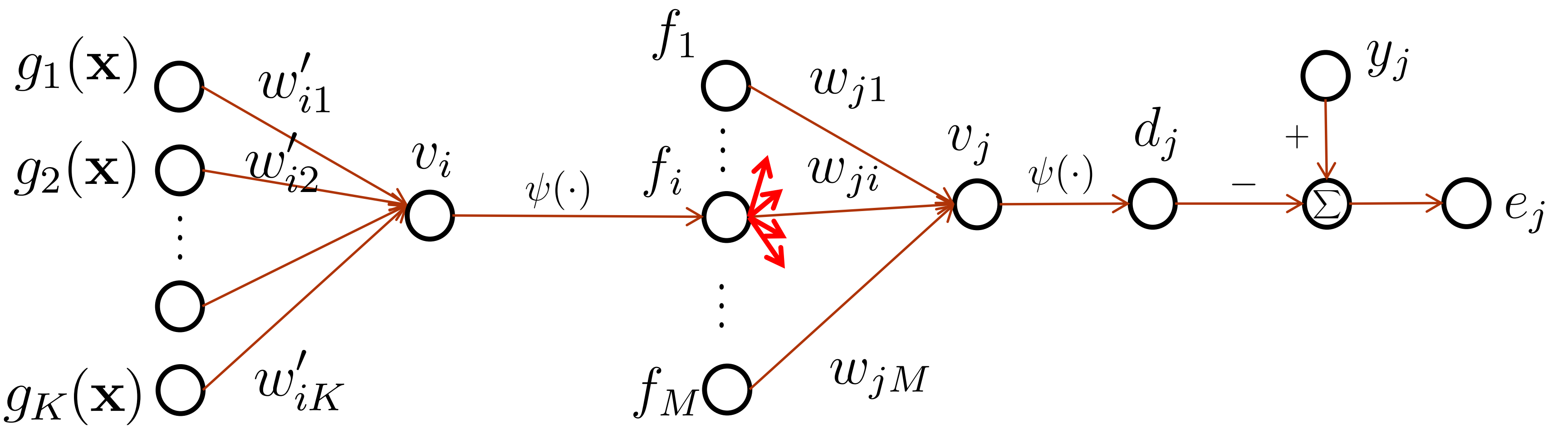
Local gradient: $\delta_j = -\dfrac{\partial R}{\partial v_j}$

Gradient computation at hidden layer

- Output neurons are NOT separate:



Gradient computation at hidden layer



$v_i = (\mathbf{w}'_i)^\top \mathbf{g} \quad f_i = \psi(v_i) \quad v_j = \mathbf{w}_j^\top \mathbf{f} \quad d_j = \psi(v_j) \quad e_j = y_j - d_j$

$$\nabla_{w'_{ik}} R = \sum_j \frac{\partial R_j}{\partial e_j}\frac{\partial e_j}{\partial d_j}\frac{\partial d_j}{\partial v_j}\frac{\partial v_j}{\partial f_i}\frac{\partial f_i}{\partial v_i}\frac{\partial v_i}{\partial w'_{ik}}$$

$$= -\sum_j e_j \psi'(v_j) w_{ji} \psi'(v_i) g_k(\mathbf{x})$$

$$= -\boxed{\sum_j \delta_j w_{ji} \psi'(v_i)} g_k(\mathbf{x})$$

$$R_j = \frac{1}{2} e_j^2$$

$$R = \frac{1}{2}\sum_j e_j^2$$

Local gradient:

$$\delta_i = -\frac{\partial R}{\partial v_i}$$

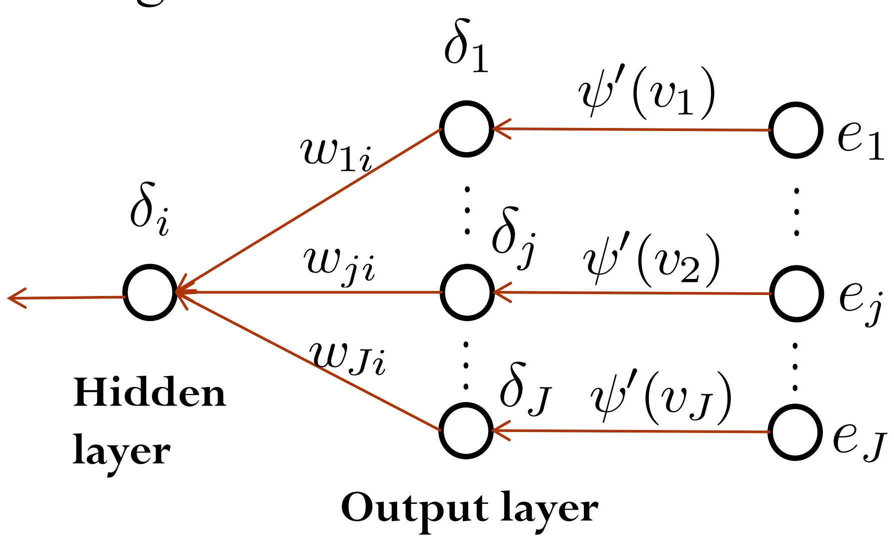
Back-propagation formula

- The update rule of **local gradients**:
  - for hidden neuron i:

$$\delta_i = \psi'(v_i) \sum_j \delta_j w_{ji}$$

Only depends on the activation function at hidden neuron i

- Flow of error signal:



Back-propagation formula

- The update rule of weights:
  - Output neuron:

$$\Delta w_{ji} = \lambda \cdot \delta_j \cdot f_i(\mathbf{x})$$

  - Hidden neuron:

$$\Delta w'_{ik} = \lambda \cdot \delta_i \cdot g_k(\mathbf{x})$$

$$\begin{pmatrix} Weight \\ correction \\ \Delta w_{ji} \end{pmatrix} = \begin{pmatrix} learning \\ rate \\ \lambda \end{pmatrix} \cdot \begin{pmatrix} local \\ gradient \\ \delta_j \end{pmatrix} \cdot \begin{pmatrix} input\ signal \\ of\ neuron\ j \\ v_i \end{pmatrix}$$

Two Passes of Computation

- Forward pass
  - Weights fixed
  - Start at the first hidden layer
  - Compute the output of each neuron
  - End at output layer

- Backward pass
  - Start at the output layer
  - Pass error signal backward through the network
  - Compute local gradients

Stopping Criterion

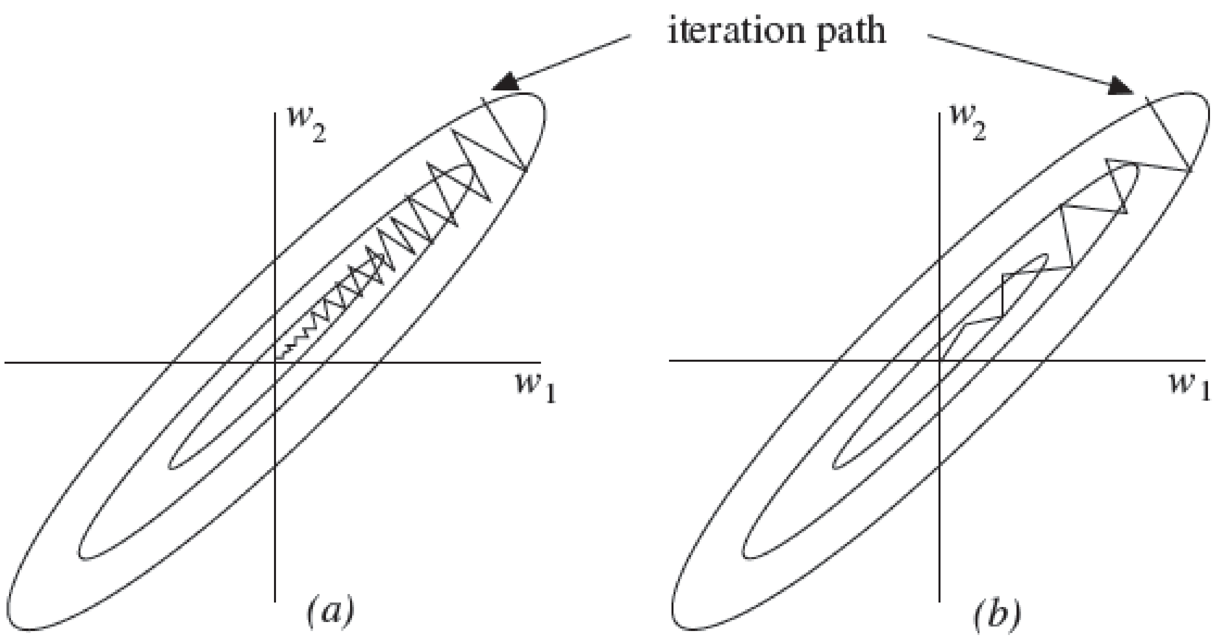
- No general rules

- Some reasonable heuristics:
  - The norm of gradient is small enough
  - The number of iterations is larger than a threshold
  - The training error is stable
  - …

Improve Backpropagation

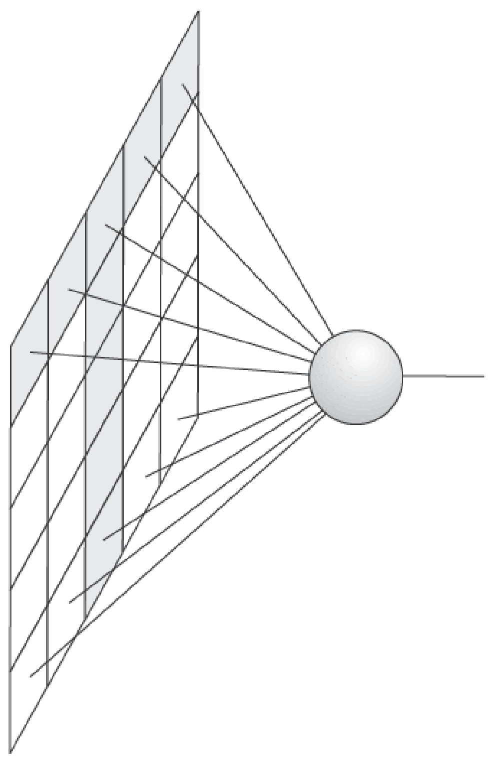
- Many methods exist to improve backpropagation
- E.g., backpropagation with momentum

$$\Delta w_{ij}^{t} = -\lambda \frac{\partial R}{\partial w_{ij}} + \alpha \Delta w_{ij}^{t-1}$$

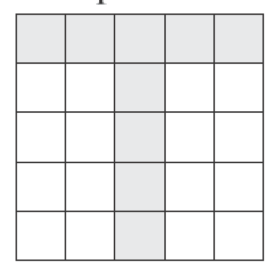


Neurons as Feature Extractor

- Compute the similarity of a pattern to the ideal pattern of a neuron
- Threshold is the minimal similarity required for a pattern
- Reversely, it visualizes the connections of a neuron



pattern



weights

1	1	1	1	1
–1	–1	1	–1	–1
–1	–1	1	–1	–1
–1	–1	1	–1	–1
–1	–1	1	–1	–1

Vanishing gradient problem

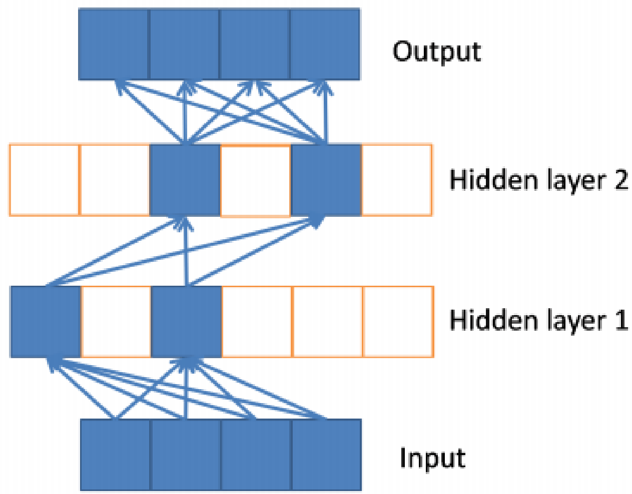
- The gradient can decrease exponentially during back-prop

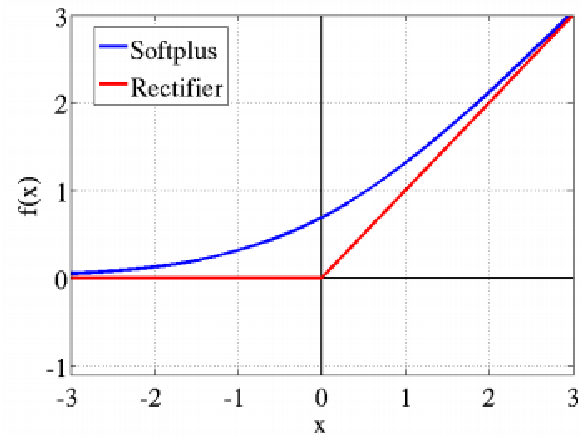
- Solutions:
  - Pre-training + fine tuning
  - Rectifier neurons (sparse gradients)

- Ref:
  - Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. Hochreiter, Bengio, & Frasconi, 2001

Deep Rectifier Nets

- Sparse representations without gradient vanishing



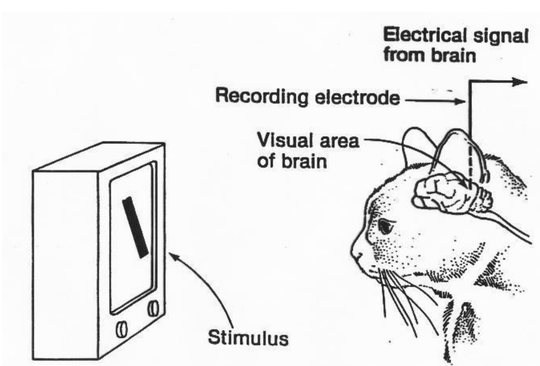


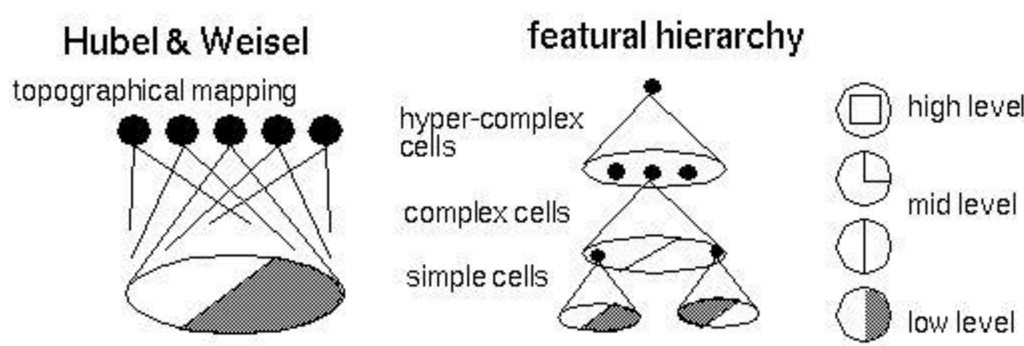
- Non-linearity comes from the path selection
  - Only a subset of neurons are active for a given input
- Can been seen as a model with an exponential number of linear models that share weights

[Deep sparse rectifier neural networks. Glorot, Bordes, & Bengio, 2011]

CNN

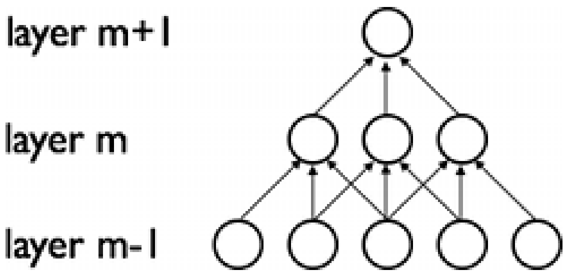
- Hubel and Wiesel's study on annimal's visual cortex:
  - Cells that are sensitive to small sub-regions of the visual field, called a *receptive field*
  - Simple cells respond maximally to specific edge-like patterns within their receptive field. Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.



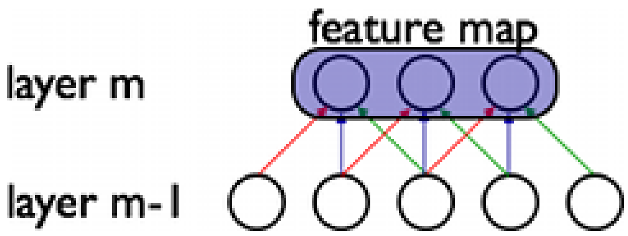


Convolutional Neural Networks

- Sparse local connections (spatially contiguous receptive fields)

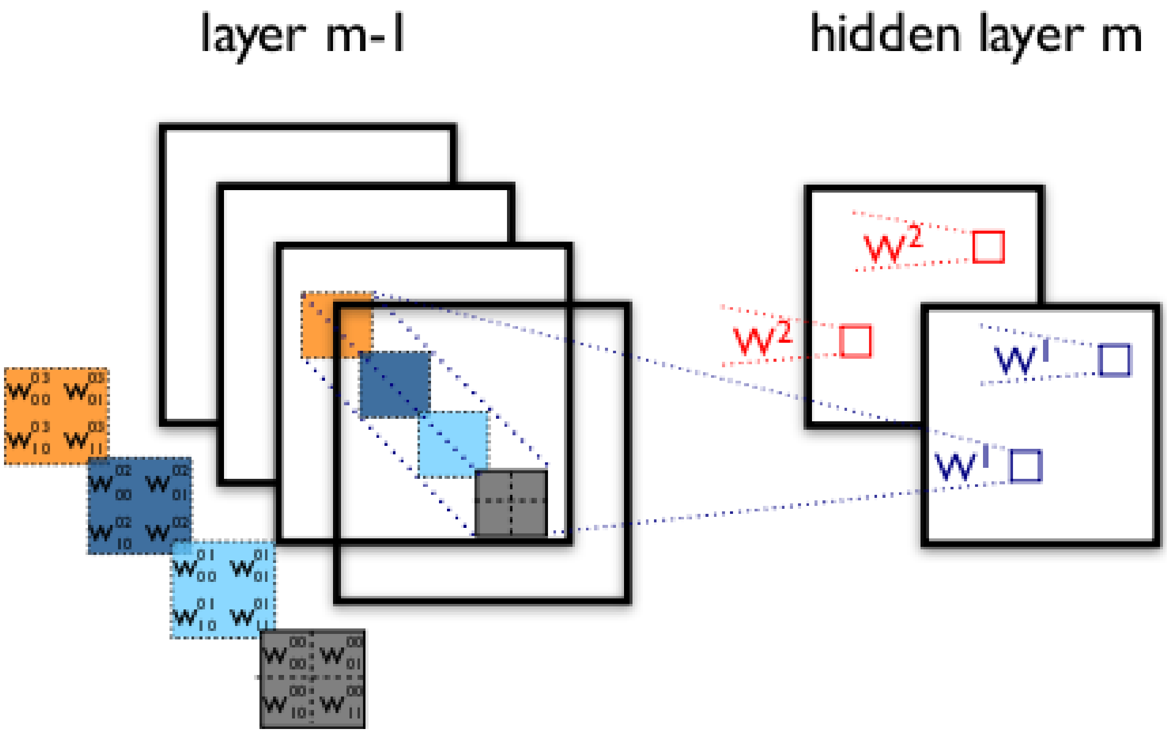


- Shared weights: each filter is replicated across the entire visual field, forming a feature map



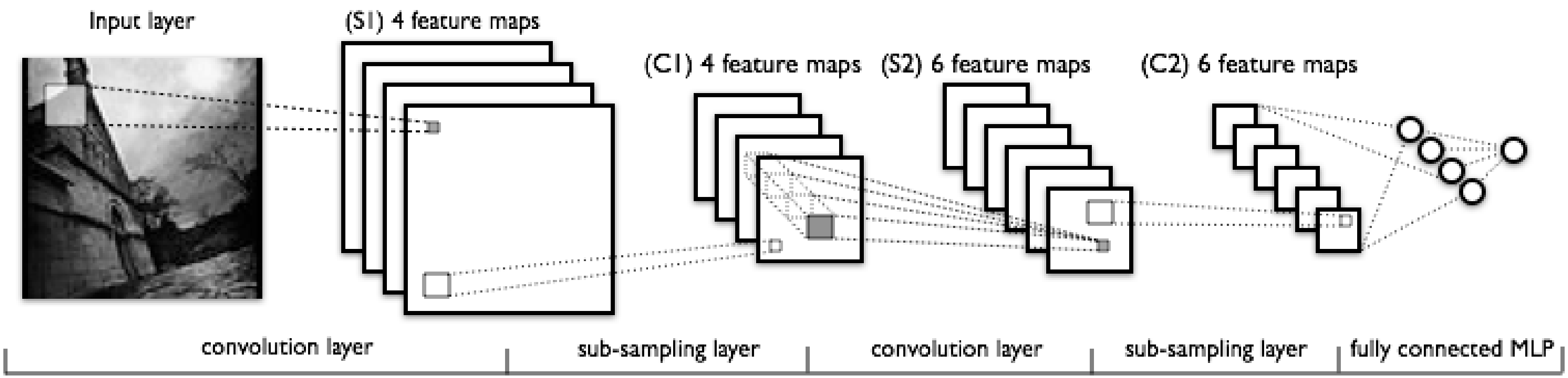
CNN

- Each layer has multiple feature maps



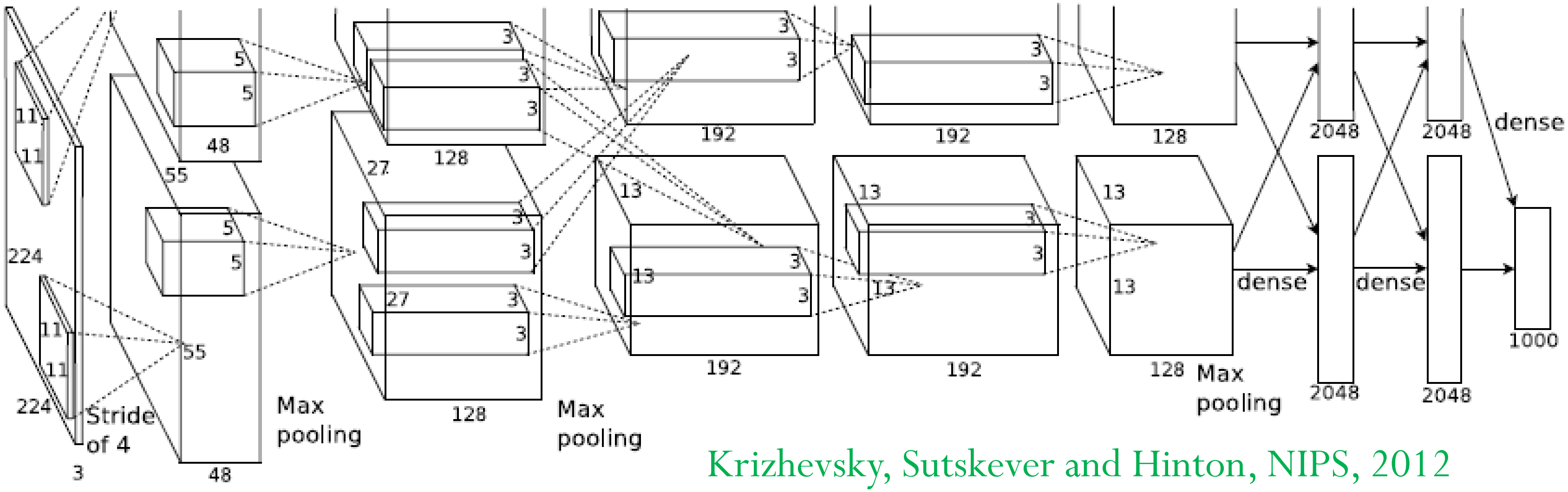
CNN

- The full model



- ***Max-pooling***, a form of non-linear down-sampling.
  - Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value.

Example: CNN for image classification



Krizhevsky, Sutskever and Hinton, NIPS, 2012

- Network dimension: 150,528(input)-253,440–186,624–64,896–64,896–43,264–4096–4096–1000(output)
  - In total: 60 million parameters
  - Task: classify 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes
  - Results: state-of-the-art accuracy on ImageNet

Issues with CNN

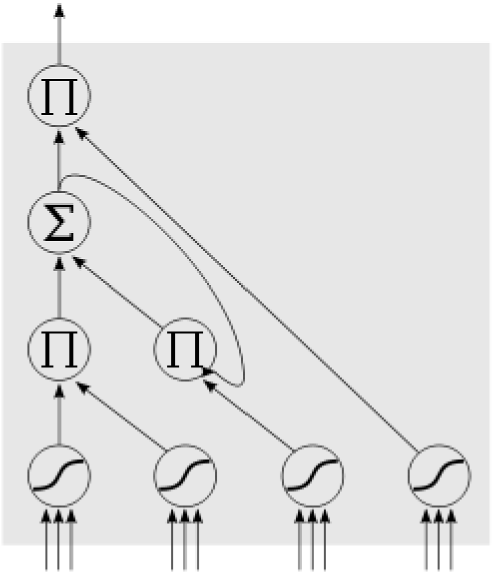
- Computing the activations of a single convolutional filter is much more expensive than with traditional MLPs

- Many tuning parameters
  - # of filters:
    - Model complexity issue (overfitting vs underfitting)
  - Filter shape:
    - the right level of "granularity" in order to create abstractions at the proper scale, given a particular dataset
    - Usually 5x5 for MNIST at 1st layer
  - Max-pooling shape:
    - typical: 2x2; maybe 4x4 for large images

Long Short-Term Memory

- A RNN architecture without gradient vanishing issue

- A RNN with LSTM blocks
  - Each block is a "smart" network, determing when to remember, when to continue to remember or forget, and when to output



[Graves et al., 2009. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition]

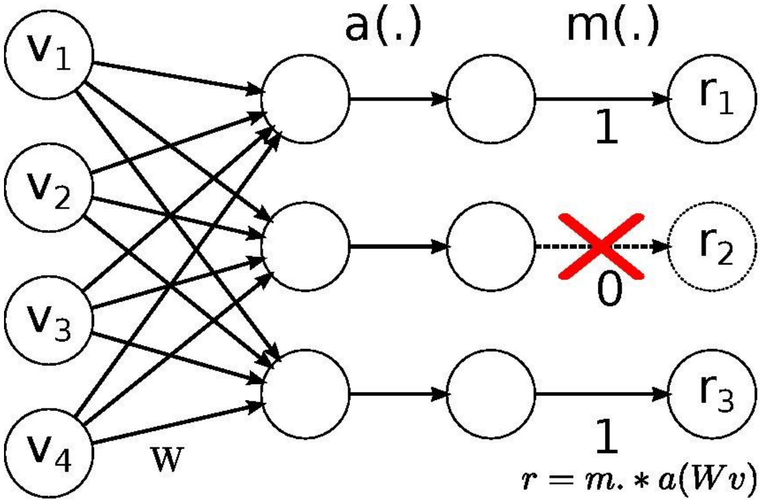
Discussions

Challenges of DL

- Learning
  - Backpropagation is slow and prone to gradient vanishing
  - Issues with non-convex optimization in high-dimensions
- Overfitting
  - Big models are lacking of statistical information to fit
- Interpretation
  - Deep nets are often used as black-box tools for learning and inference

Overfitting in DL

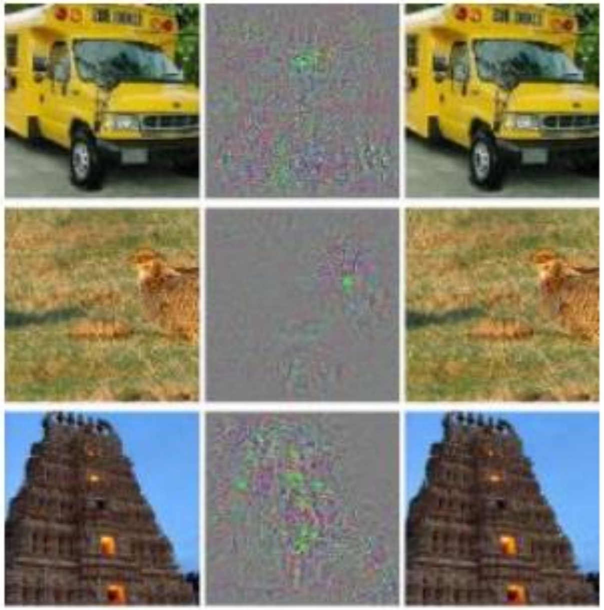
- Increasing research attention, e.g., dropout training (Hinton, 2012)



- More theoretical understanding and extensions
  - MCF (van der Maaten et al., 2013); Logistic-loss (Wager et al., 2013); Dropout SVM (Chen, Zhu et al., 2014)

Some counter-intuitive properties

- Stability w.r.t small perturbations to inputs
  - Imperceptible non-random perturbation can arbitrarily change the prediction (adversarial examples exist!)



(a)

10x of differences

(b)

[Szegedy et al., Intriguing properties of neural nets, 2013]

Criticisms of DL

- Just a buzzword, or largely a rebranding of neural networks

- Lack of theory
  - gradient descent has been understood for a while
  - DL is often used as black-box

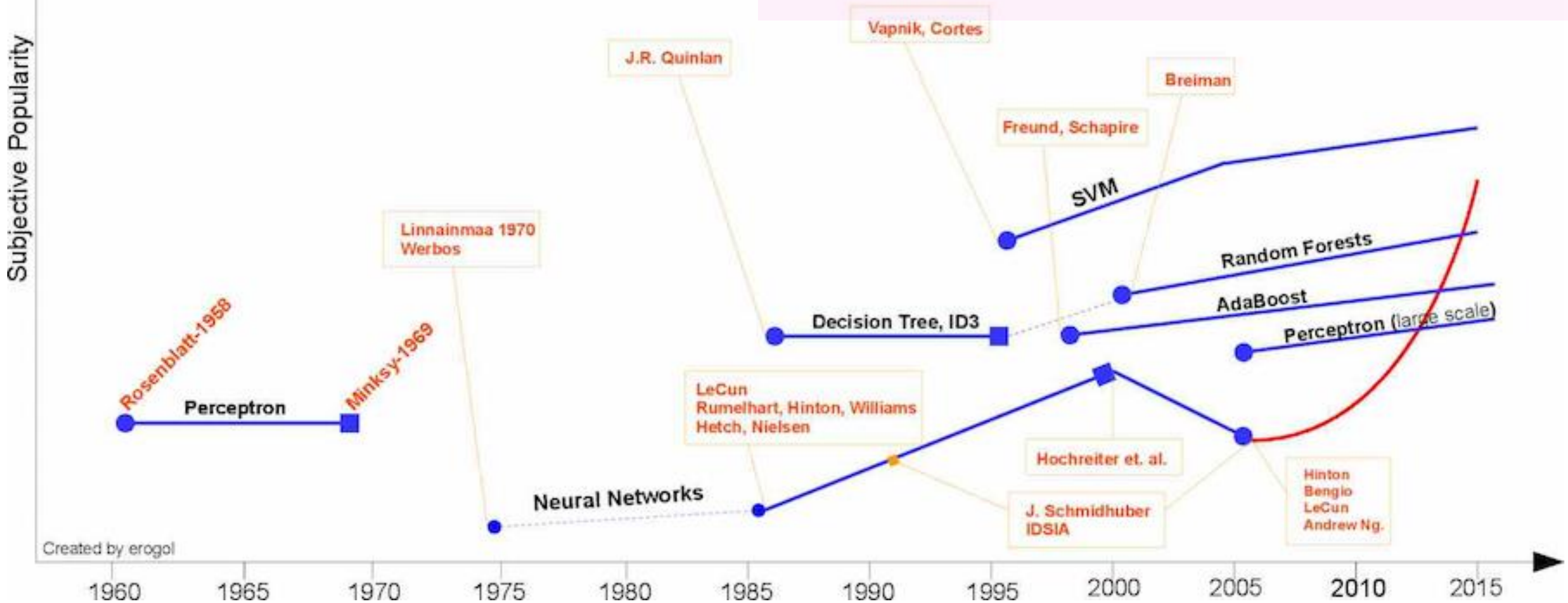
- DL is only part of the larger challenge of building intelligent machines, still lacking of:
  - causal relationships
  - logic inferences
  - integrating abstract knowledge

Will DL make other ML methods obsolete?

Quora 2014/12/23

Yes (2 post, 113 upvotes)
- best predictive power when data sufficient
- DL is far from saturated
- Google et al invests on DL, it is the "richest" AI topic

No (10 posts, 284 upvotes)
- simpler algorithms are just fine in many cases
- methods with domain knowledge works better
- DL is feature learning, needs other methods to work
- DL is not that well developed, a lot of work to be done using more traditional methods
- No free lunch
- a lot like how ANN was viewed in the late 80s



What are people saying?

- **Yann LeCun**:
  - “AI has gone from failure to failure, with bits of progress. This could be another leapfrog”
- **Jitendra Malik**:
  - in the long term, deep learning may not win the day; … “Over time people will decide what works best in different domains.”
  - “Neural nets were always a delicate art to manage. There is some black magic involved”
- **Andrew Ng**:
  - “Deep learning happens to have the property that if you feed it more data it gets better and better,”
  - “Deep-learning algorithms aren't the only ones like that, but they're arguably the best — certainly the easiest. That's why it has huge promise for the future.”

[Nature 505, 146–148 (09 January 2014)]

What are people saying?

- **Oren Etzioni**:
  - "It's like when we invented flight" (not using the brain for inspiration)

- **Alternatives**:
  - Logic, knowledge base, grammars?
  - Quantum AI/ML?

[Nature 505, 146–148 (09 January 2014)]

Thank You!