So Far…

- Our goal (supervised learning)
  - Learn a set of discriminant functions

  - Bayesian framework
    - We could design an optimal classifier if we knew:
      - $P(\omega_i)$: priors and $P(x \mid \omega_i)$: class-conditional densities
      - Using training data to estimate $P(\omega_i)$ and $P(x \mid \omega_i)$
  - Directly learning discriminant functions from the training data
    - Linear Regression
    - Logistic Regression
    - SVM
    - Kernel methods

Disadvantage?

- Only focus on the classifier.
- We have to know the form of the discriminant functions (the nonlinearity).

Artificial Neural Network & Deep Learning

Deng Cai (蔡登)

College of Computer Science
Zhejiang University

dengcai@gmail.com

2019 Turing Award Winners

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built.

What is Deep Learning

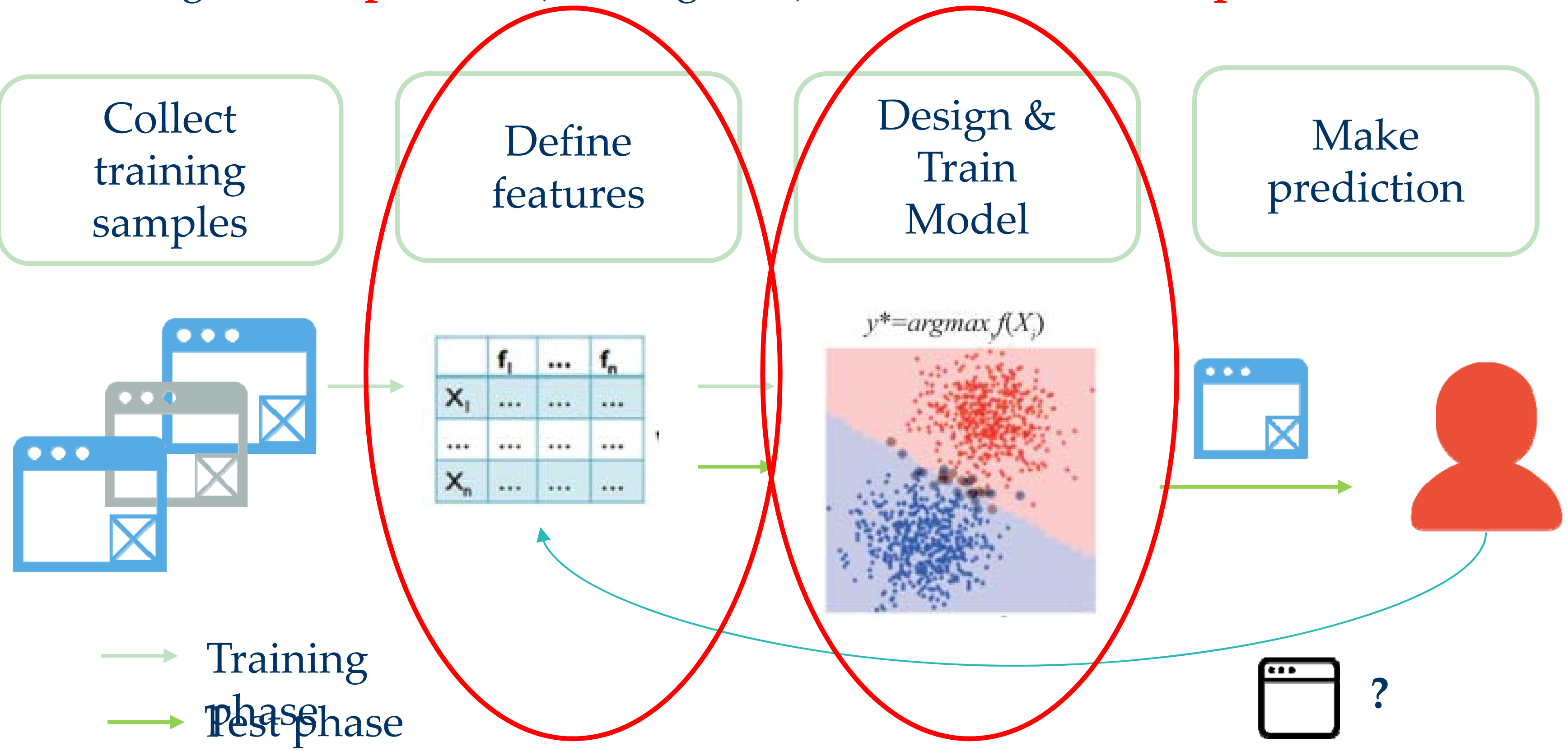
Deep learning is a branch of **machine learning** based on learning **representations** of data. It's a set of algorithms that attempt to model high-level abstractions in data by using **multiple processing layers**, with complex structures or otherwise, composed of **multiple non-linear transformations**.

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data.

Wiki

Supervised Learning

Learning from **experience**(training data), and build **model** to **predict** the future



Collect training samples

Define features

Design & Train Model

Make prediction

Training phase

Test phase

Step 1

Step 2

Representation Learning

Supervised Learning

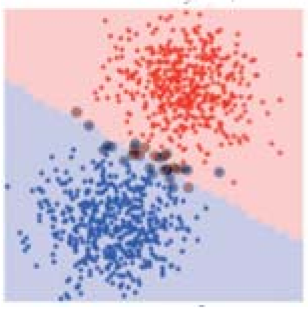
Define features

	f_1	...	f_n
X_1	...	...	...
...	...	...	...
X_n	...	...	...

Step 1

Design & Train Model

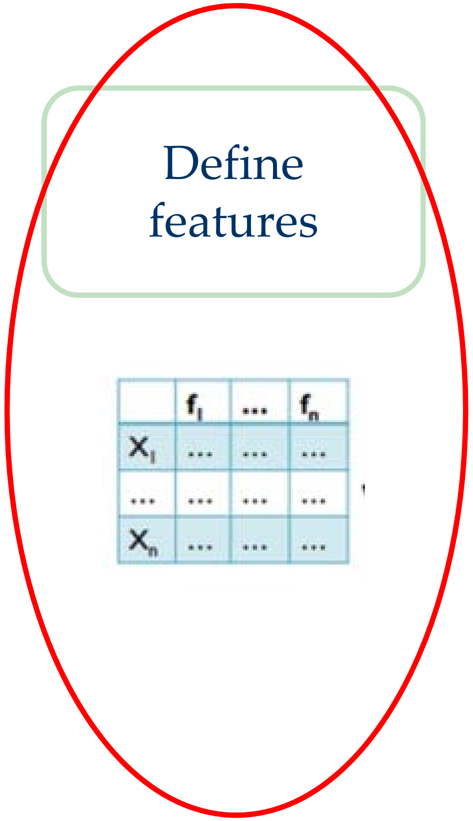
$y^*=argmax\, f(X_i)$



Step 2

- Step 1 is more important in building a successful system.

Why general classification hard?

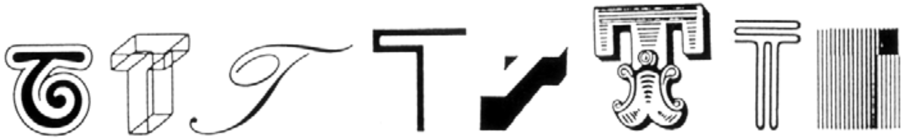


Define features

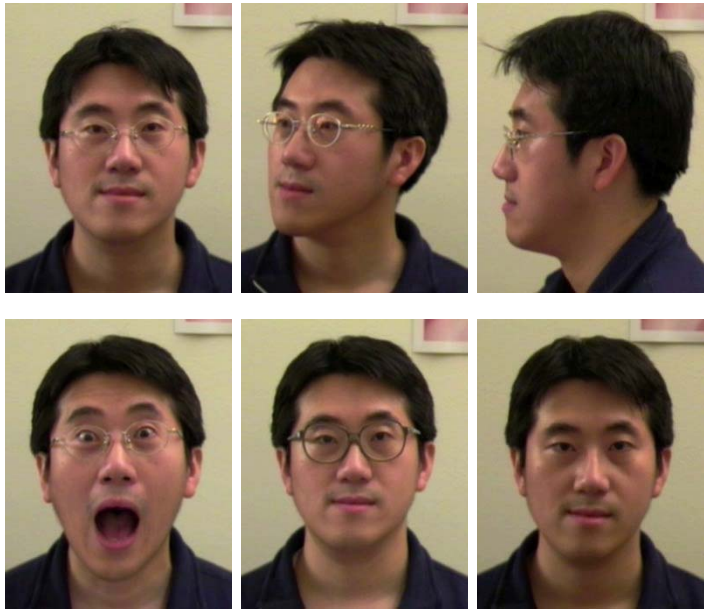
	f_1	...	f_n
X_1	...	...	...
...	...	...	...
X_n	...	...	...

Step 1 is not good enough

- Intra-class variability

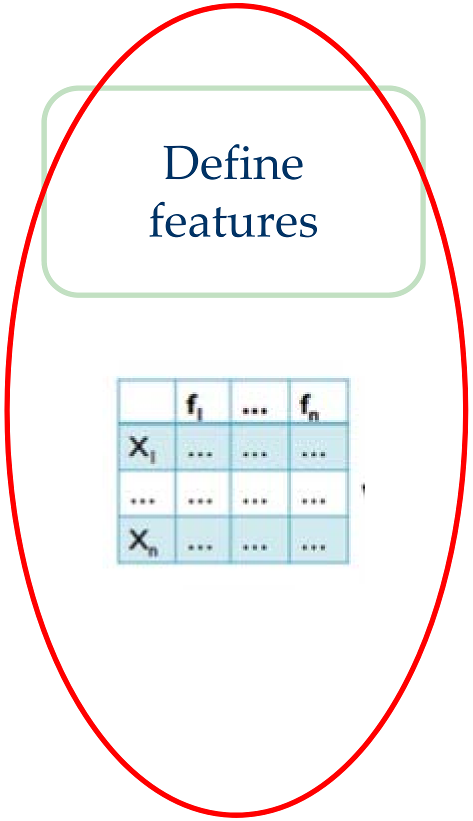


The letter “T” in different typefaces



Same face under different expression, pose, illuminatior

Why general classification hard?

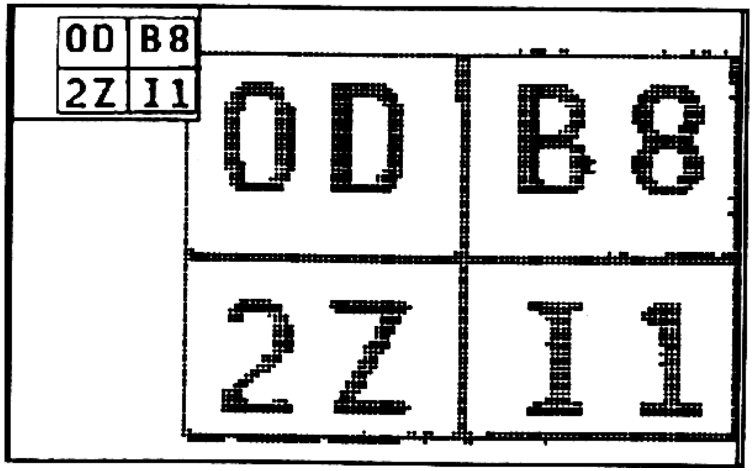


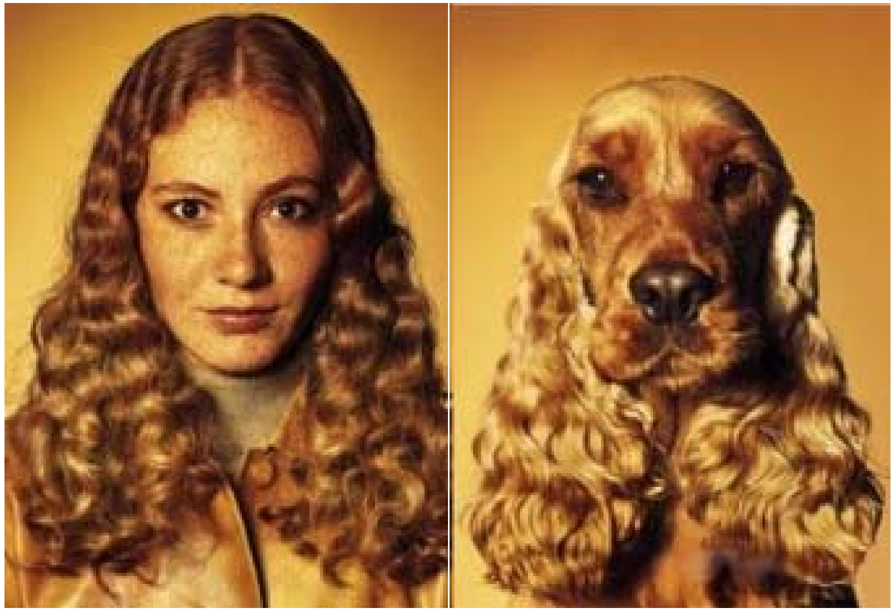
Define features

	f_1	...	f_n
x_1	...	...	...
...	...	...	...
x_n	...	...	...

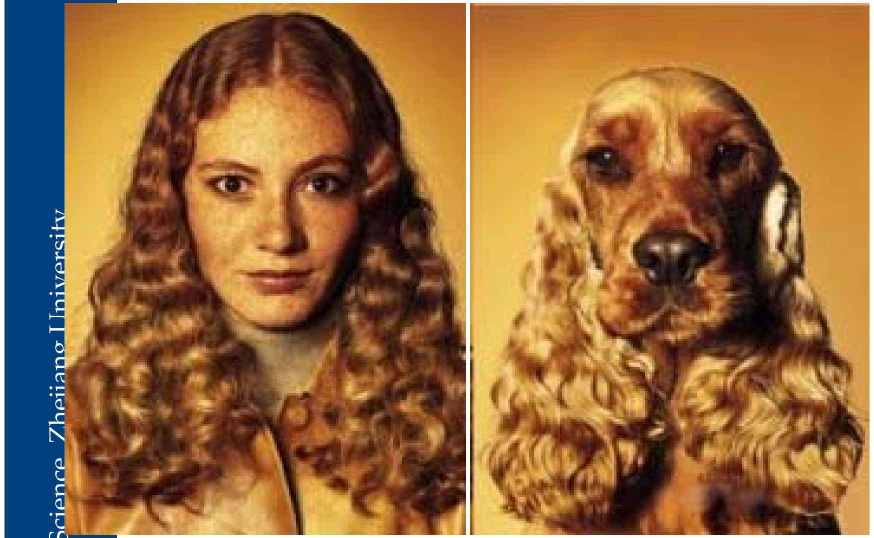
Step 1 is not good enough

- Inter-class similarity

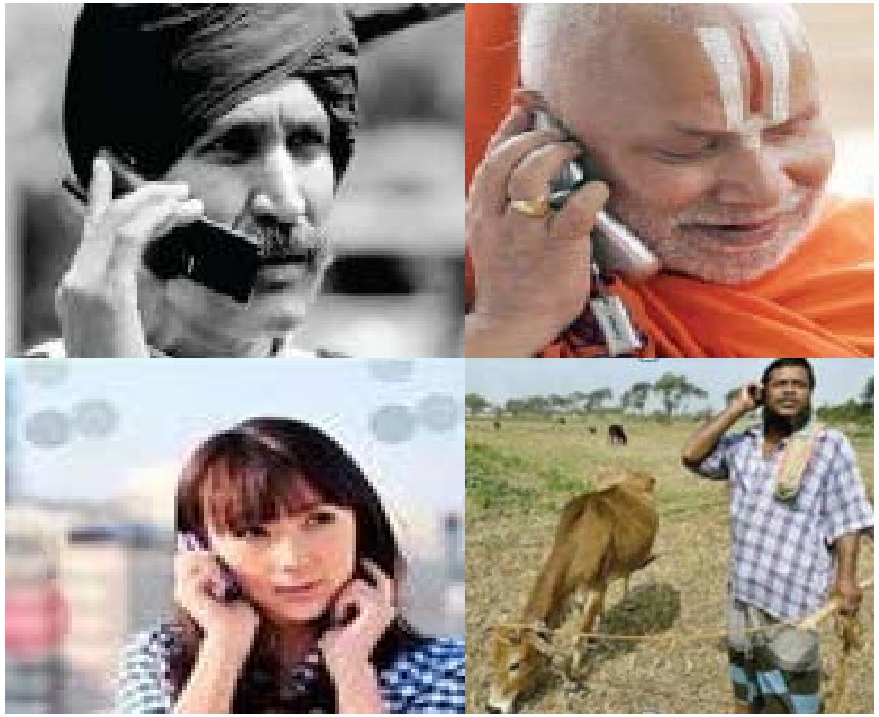




Semantic Gap

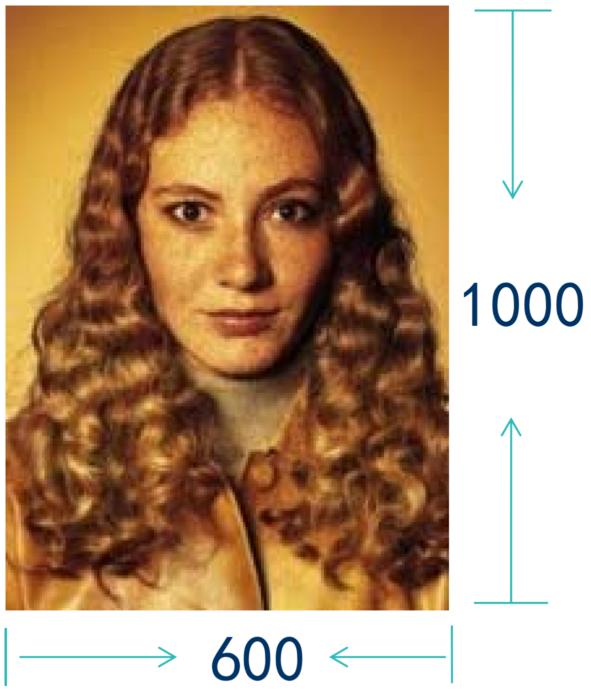


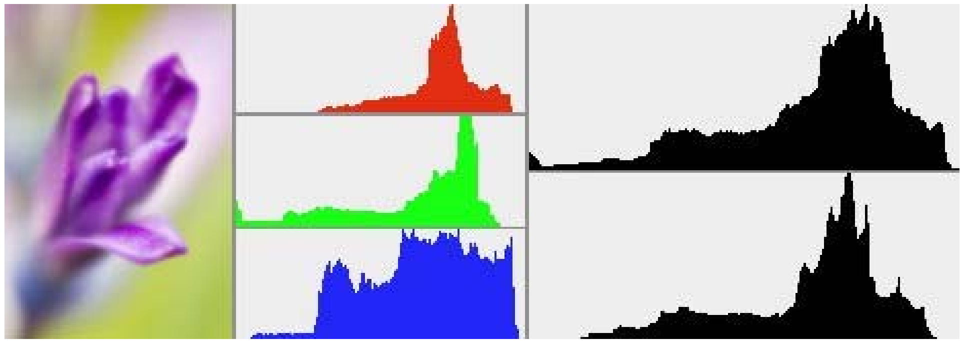
Looks similar
But semantically different



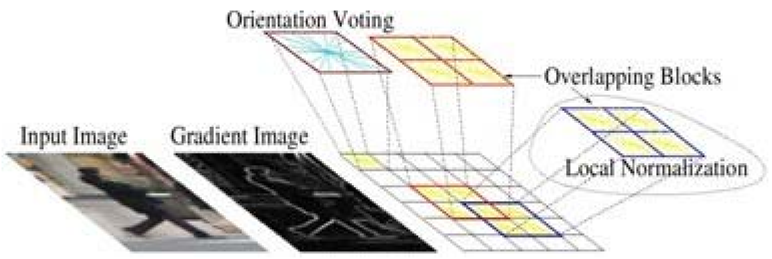
Looks different
But semantically the same

Representation (Feature)

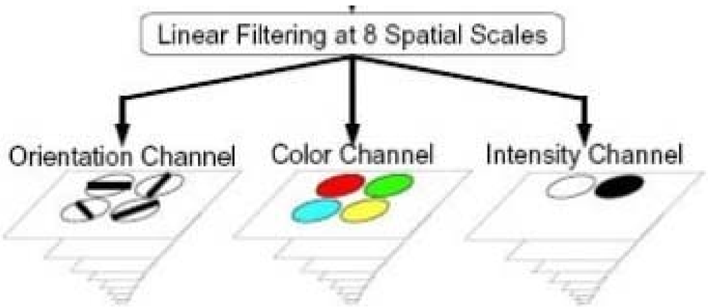




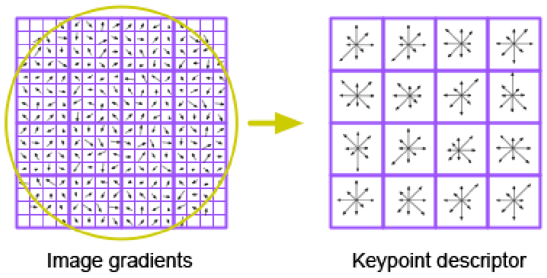
colorHistogram,1992



HOG (Histogram of Oriented Gradients),2005

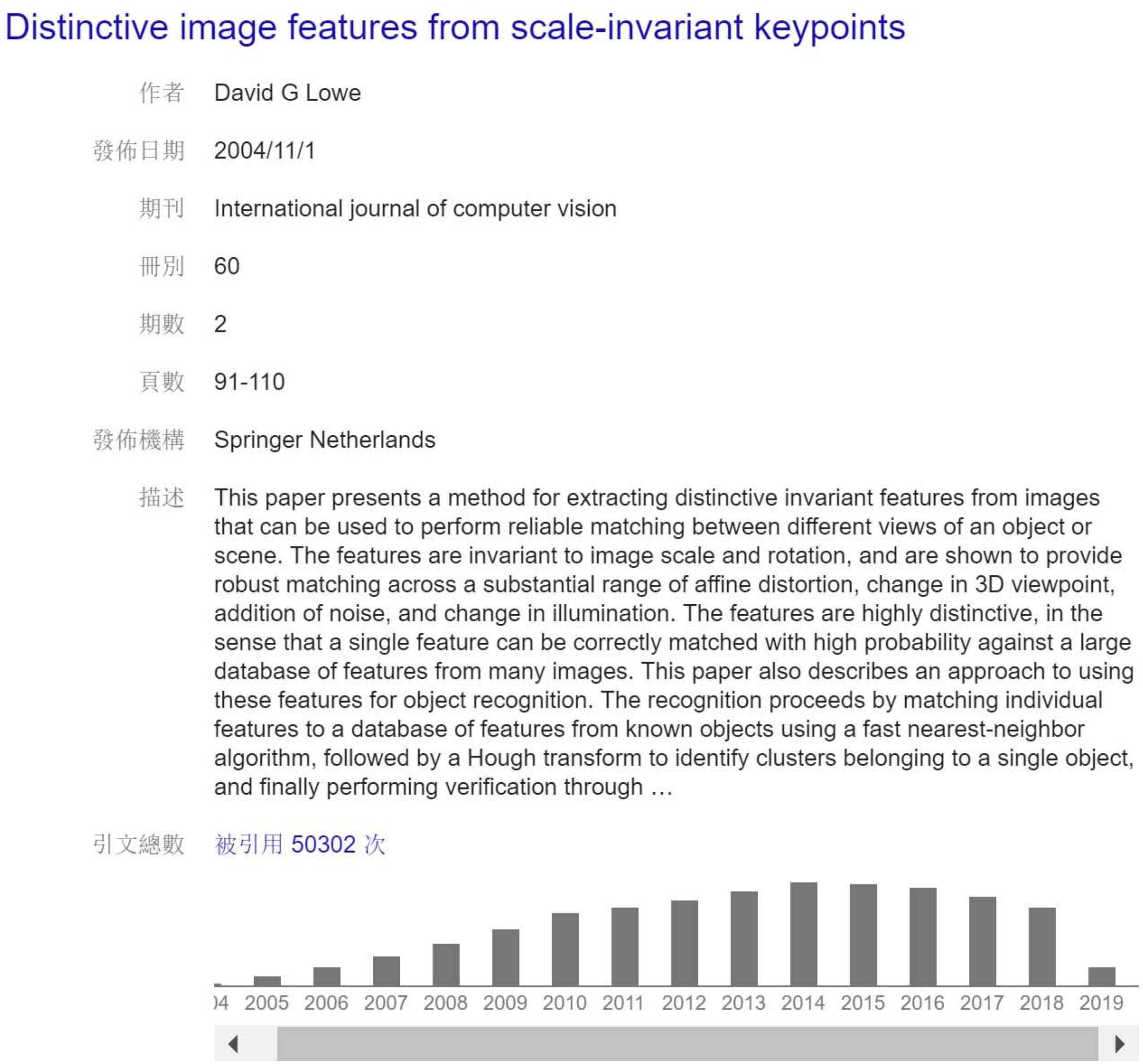


GIST,2001, 2003



SIFT,1999, 2004

SIFT



Distinctive image features from scale-invariant keypoints

作者	David G Lowe
發佈日期	2004/11/1
期刊	International journal of computer vision
冊別	60
期數	2
頁數	91-110
發佈機構	Springer Netherlands
描述	This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through …
引文總數	被引用 50302 次

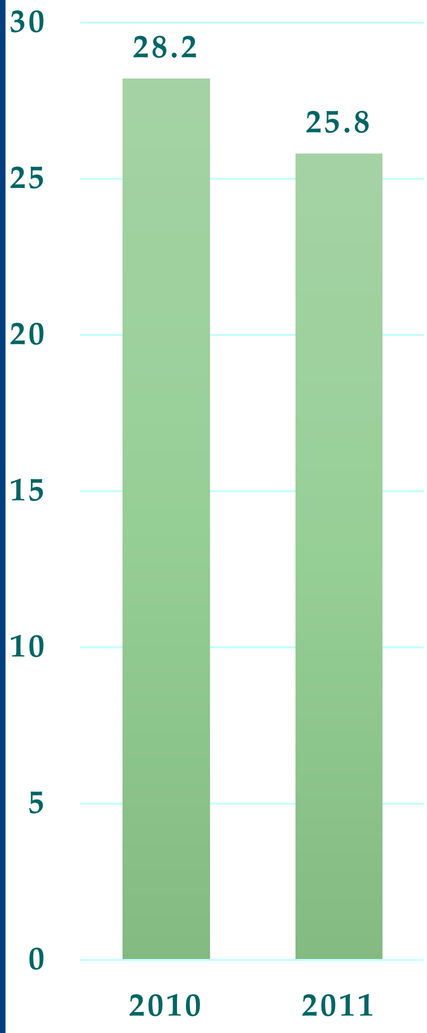
Imagenet competition

Large Scale Visual Recognition Challenge (ILSVRC)

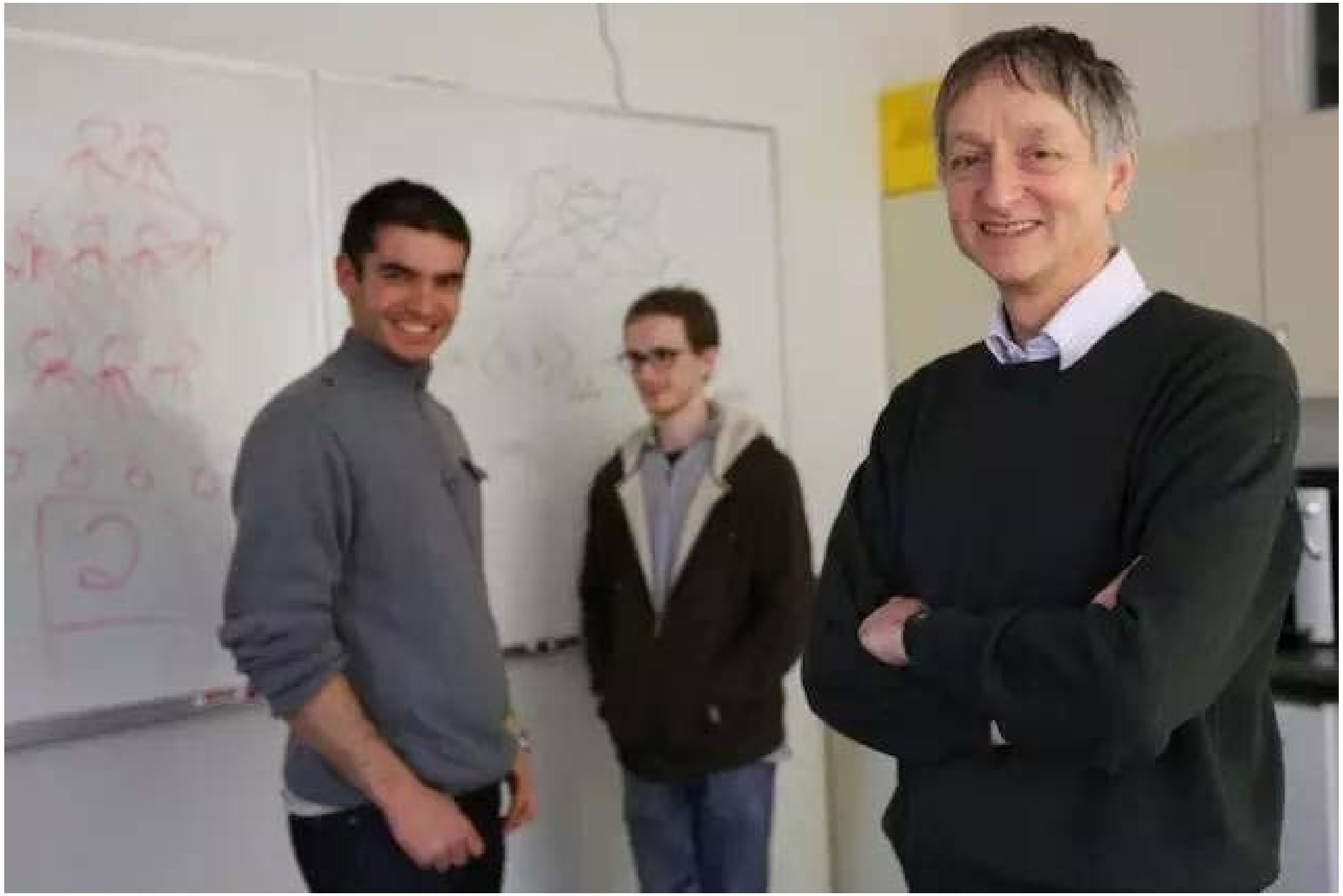
2010 ~ 2017

1000 categories, from flickr and other search engines
1.2 million training images
50,000 validation images , **100,000** test images

ImageNet Classification top-5 error(%)

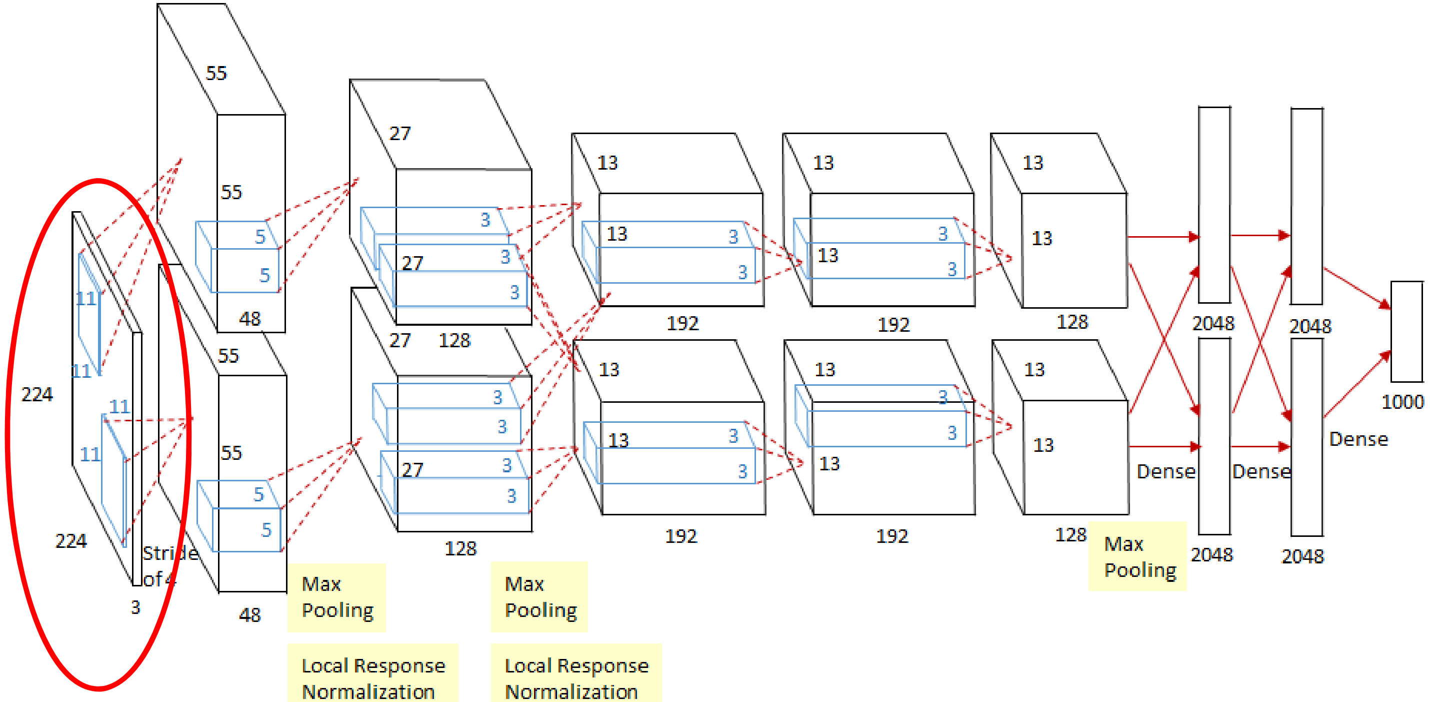


Imagenet break through

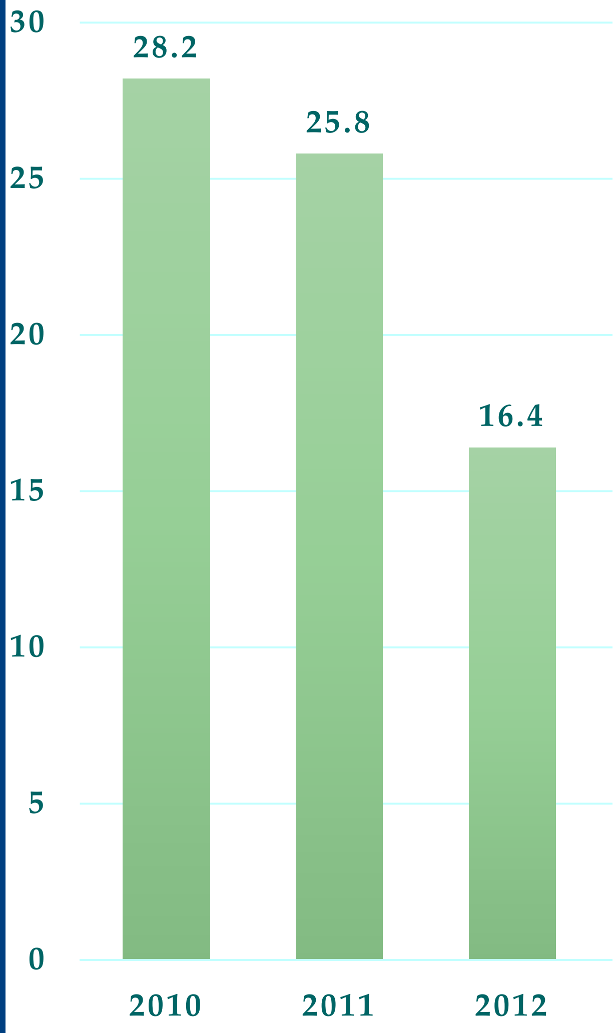


Suskever, Krizhevsky, Hinton, from left to right

AlexNet (Alex Krizhevsky)



ImageNet Classification top-5 error(%)



Deep Learning

(Neural Network, NN)

(Artificial Neural Network, ANN)

Linear model (classifier)

- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \begin{cases} > 0 & class\ 1 \\ < 0 & class\ 2 \end{cases}$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$$

$$\boldsymbol{x} = [x_1, x_2, \cdots x_d, 1]^T \in R^{d+1}$$

$$\boldsymbol{w} = [w_1, w_2, \cdots w_d, b]^T \in R^{d+1}$$

Linear model (classifier)

- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b = 0$$

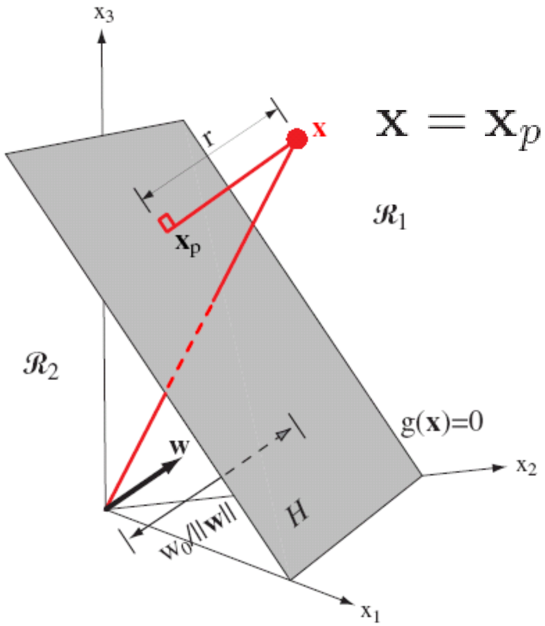
- Decision surface
- 1: What is the normal vector of this hyper-plane?

$$f(\boldsymbol{x}_1) = \boldsymbol{w}^T \boldsymbol{x}_1 + b = 0 = f(\boldsymbol{x}_2) = \boldsymbol{w}^T \boldsymbol{x}_2 + b$$

$$\boldsymbol{w}^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$$

- 2: What is the distance of any point x to this hyperplane?

Decision Surface



$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$f(\boldsymbol{x}_p) = \boldsymbol{w}^T\left(\boldsymbol{x} - r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + b = 0$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = r\|\boldsymbol{w}\| \qquad r = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

Two-category Linearly Separable Case

- If
  - $\boldsymbol{w}^T\boldsymbol{x} > 0$ for examples from the positive class.
  - $\boldsymbol{w}^T\boldsymbol{x} < 0$ for examples from the negative class.

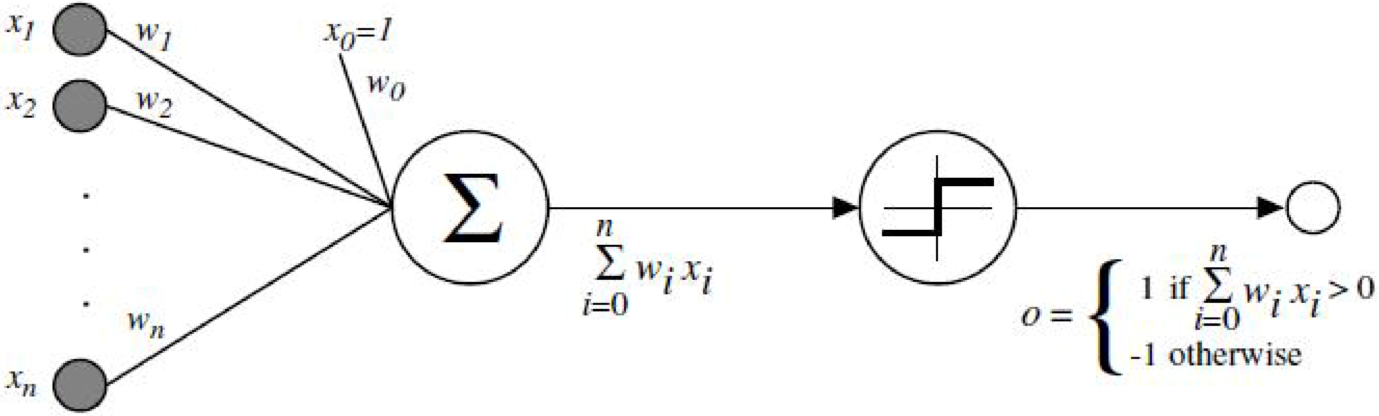
- Such a weight vector $\boldsymbol{w}$ is called a *separating vector* or a *solution vector*

- Normalizing the input examples by multiplying them with their class label (replace all samples from class 2 by their negatives)
  - $\boldsymbol{x} \rightarrow \boldsymbol{z}$
  - $\boldsymbol{w}^T\boldsymbol{z} > 0$ for all the examples (here $\boldsymbol{z}$ is $\boldsymbol{x}$ multiplied with class label)

Linear model (classifier)

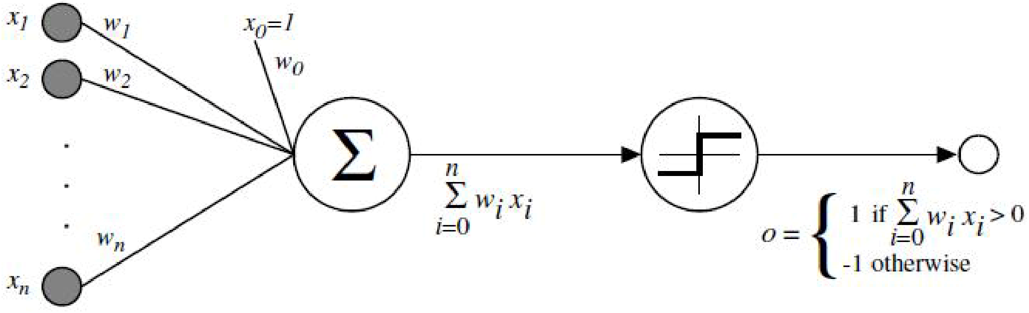
- Binary classification problem (we have two classes)
- Sample: $\boldsymbol{x} \in R^d, \boldsymbol{x} = [x_1, x_2, \cdots x_d]^T$
- Finds a linear function $\boldsymbol{w} = [w_1, w_2, \cdots, w_d]^T \in R^d, b$

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \begin{cases} > 0 & class\ 1 \\ < 0 & class\ 2 \end{cases}$$



x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Perceptron

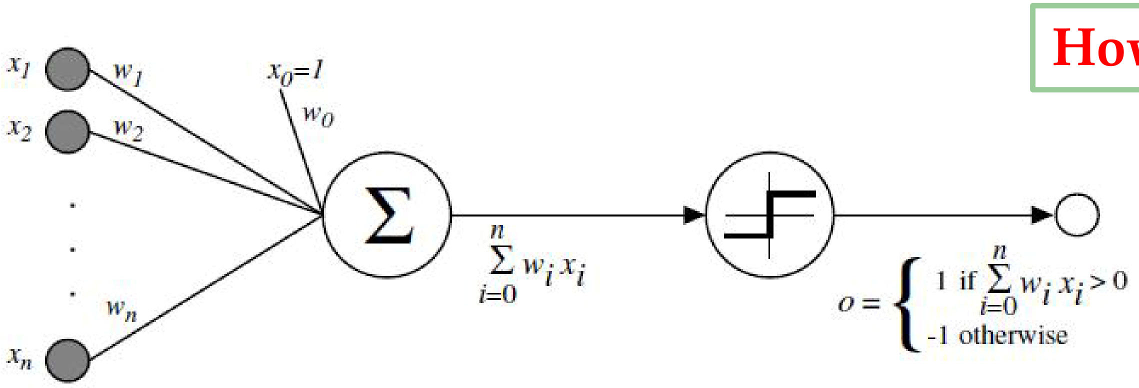

x_1
w_1
x_2
w_2
$x_0=1$
w_0
w_n
x_n
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

- In 1957, **Frank Rosenblatt** in Cornell University invented the **Perceptron** model. It is:
  - The first self-organizing and self-learning mathematic model.
  - The first algorithm defining Neural Network precisely.
  - The ancestor of many new Neural Network models.

Frank Rosenblatt
(1928-1971)

Perceptron

How to Learn?



x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$

- Seems correct buy why?
- Hint: If w is a solution, for any $\boldsymbol{x}$ in the training set, we have

$$\boldsymbol{w}^T \boldsymbol{x}y > 0$$

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)

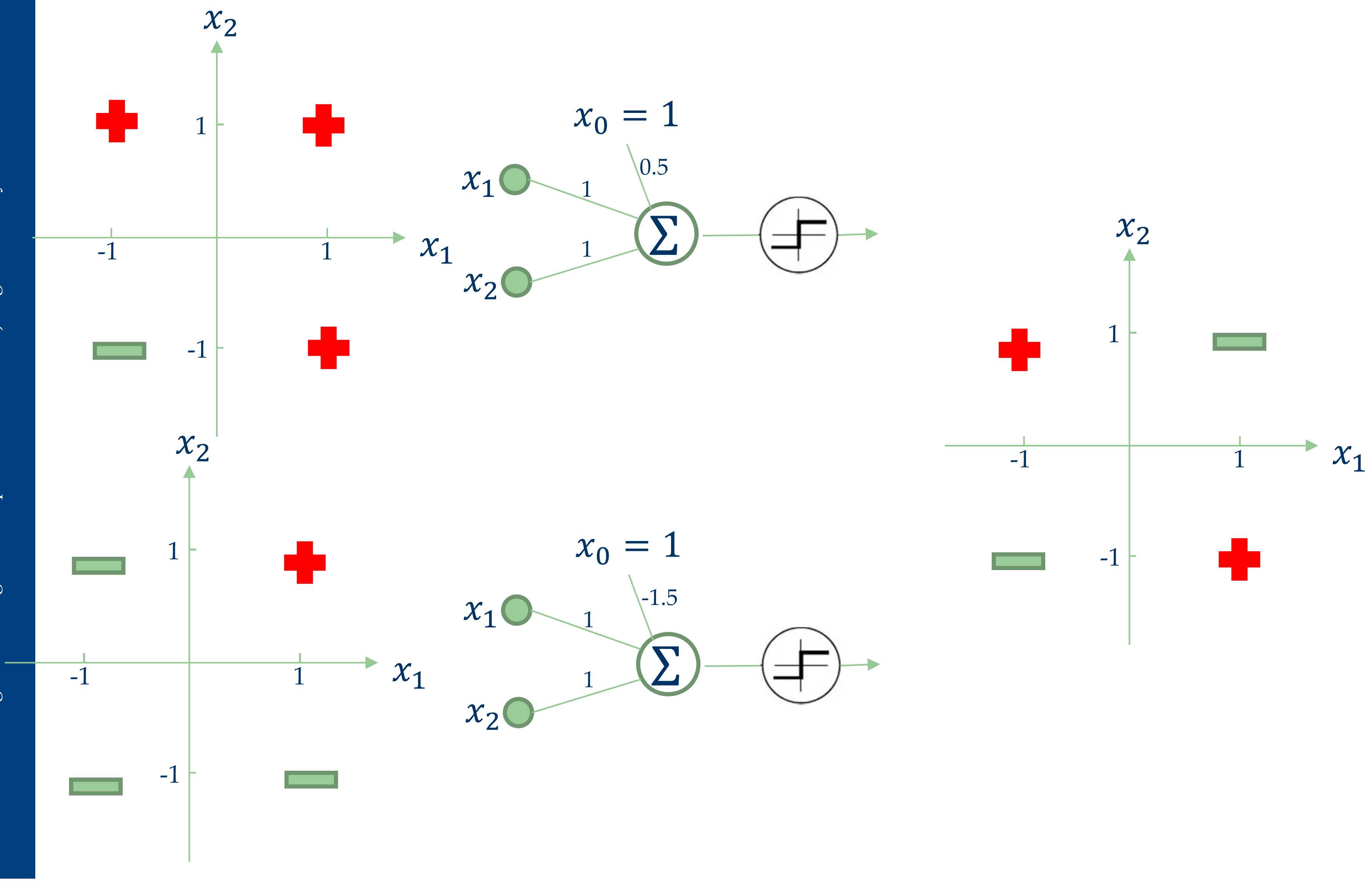
- This model had been quite popular in the 1960s.

  - Rosenblatt had a hopeful view of percetron, predicting that it would be able to learn, to make decisions, and to translate.

  - The US Navy used to fund its research, expecting that it "*will be able to walk, talk, see, write, reproduce itself and be conscious of its existence*", till the first AI winter came.

Perceptron: Drawback

- Only capable of learning **linearly separable functions.**

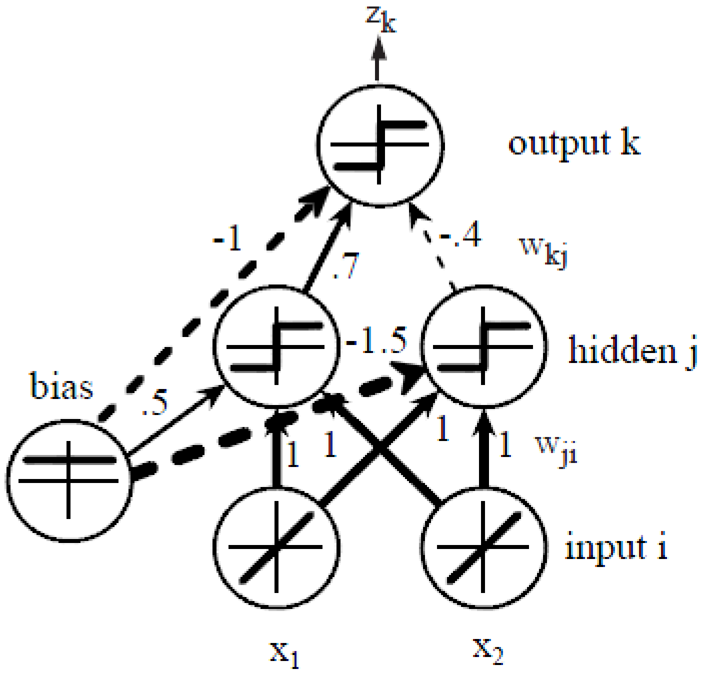


The First Winter

- In 1969, Marvin Minsky and Seymour Paper published the monograph "*Perceptrons: an introduction to computational geometry* ".
- This book revealed two fatal shortcomings of the perceptron model:
  - Single layered perceptron is not able to solve problems with non-linearity , e.g. XOR gate.
  - Its computation budget was too huge for the computers at that age.
- In the next more than ten years, the NN-based AI researches were nearly abandoned and related projects could not get any funds from the government.
- This period has been called the first AI Winter.
- Rosenblatt died in 1971 in an accident, not lucky enough to witness the renaissance of NN research.

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by "Perceptrons"

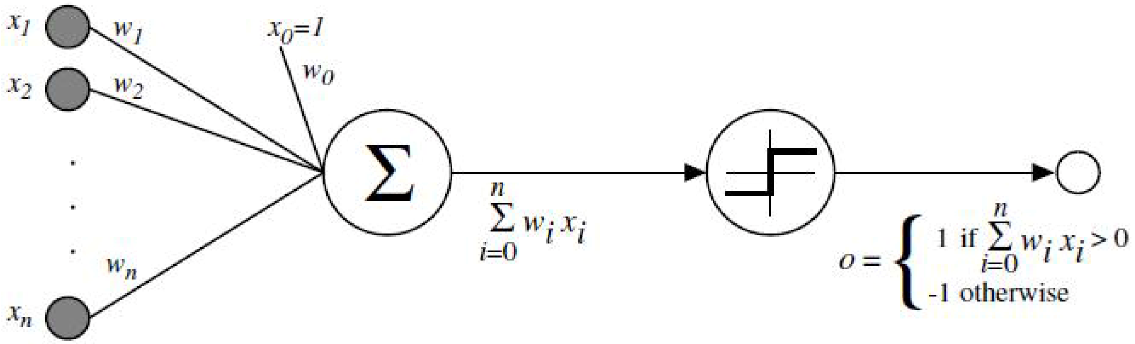


x_1	x_2	y_1	y_2	z
1	1	1	1	-1
1	-1	1	-1	1
-1	1	1	-1	1
-1	-1	-1	-1	-1

$$z = x_1\, XOR\, x_2$$

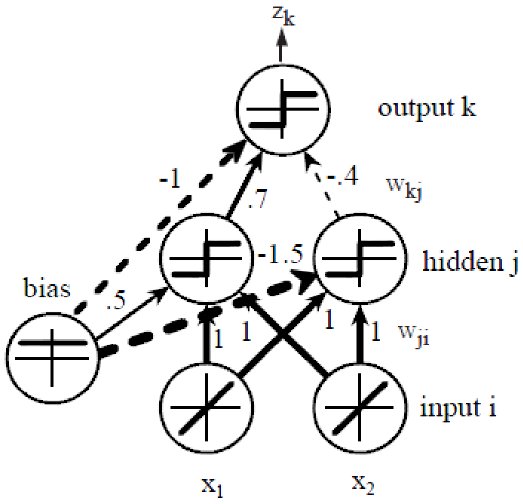
- What is the relation between z and x_i?
  - Assume x_1 and x_2 are -1, +1

Perceptron vs. Multi-layer Perceptron


x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$



How to Learn?

Principle Way To Learn Perceptron

- Define the Criterion Function (Loss Function) of Perceptron
- Then optimize(minimize) this function

- What is the criterion function of perceptron?
  - If an example can be correctly predicted, No penalty.
  - If the model made an error on an example, how to put the penalty?

Hint: $r = \frac{f(\boldsymbol{x})}{\|\boldsymbol{w}\|}$

$$J(\boldsymbol{w}) = -\sum_{i \in I_M} \boldsymbol{w}^T \boldsymbol{x}_i y_i$$

The criterion is proportional to the sum of distances from the misclassified samples to the decision boundary.

Gradient Descent

Optimization Algorithm

$$J(\boldsymbol{w}) = -\sum_{i \in I_M} \boldsymbol{w}^T \boldsymbol{x}_i y_i$$

- Use gradient descent to find $\boldsymbol{w}$
  - Move in the negative direction of the gradient iteratively to reach the minima.
- The gradient vector is given by,

$$\nabla J = \sum_{i \in I_M} -\boldsymbol{x}_i y_i$$

- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$

- Starting from $\boldsymbol{w} = \boldsymbol{0}$, update $\boldsymbol{w}$ at each iteration k as follows:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta(k) \sum_{i \in I_M^k} \boldsymbol{x}_i y_i$$

Batch learning vs. Online learning

- Batch learning
  - All the training samples are available
- Online learning or mini-batch learning
  - The learning algorithm sees the training samples one by one.
- What is the advantages of online (mini-batch) learning?

Perceptron: Mistake Bound Theorem

- Maintains a weight vector $\boldsymbol{w} \in \mathcal{R}^p$, $\boldsymbol{w}_1 = (0, \cdots, 0)$.
- Upon receiving an example $\boldsymbol{x} \in \mathcal{R}^p$
- Predicts according to the linear threshold function $\mathrm{sgn}(\boldsymbol{w}^T\boldsymbol{x})$.

Theorem [Novikoff,1963] *Let $(\boldsymbol{x}_1, y_1) \cdots (\boldsymbol{x}_t, y_t)$, be a sequence of labeled examples with $\boldsymbol{x}_i \in \mathcal{R}^p$, $\|\boldsymbol{x}_i\| \leq R$ and $y_i \in$ {-1,1} for all i. Let $\boldsymbol{u} \in \mathcal{R}^p$, $\gamma >$ 0 be such that $y_i\boldsymbol{u}^T\boldsymbol{x}_i \geq \gamma$ for all i. Then Perceptron makes at most $\frac{\|\boldsymbol{u}\|^2 R^2}{\gamma^2}$ mistakes on this example sequence.*

Margin Complexity Parameter

$$\frac{\max_i \|\boldsymbol{x}_i\|^2 \|\boldsymbol{u}\|^2}{\min_i (y_i\boldsymbol{u}^T\boldsymbol{x}_i)^2}$$

© Deng Cai, College of Computer Science, Zhejiang University

Perceptron-Mistake Bound

Proof: Let $\boldsymbol{u}$ be any solution vector, so that $y_i \boldsymbol{u}^T \boldsymbol{x}_i$ is strictly positive for all i, and let α be a positive scale factor.

$$\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u} = (\boldsymbol{w}_k - \alpha \boldsymbol{u}) + y_k \boldsymbol{x}_k$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 = \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 + 2(\boldsymbol{w}_k - \alpha \boldsymbol{u})^T y_k \boldsymbol{x}_k + \|\boldsymbol{x}_k\|^2$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \le \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 - 2\alpha \boldsymbol{u}^T y_k \boldsymbol{x}_k + \|\boldsymbol{x}_k\|^2$$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \le \|\boldsymbol{w}_k - \alpha \boldsymbol{u}\|^2 - 2\alpha\gamma + R^2$$

Assumptions

$y_i \boldsymbol{u}^T \boldsymbol{x}_i \ge \gamma$

$\|\boldsymbol{x}_i\| \le R$

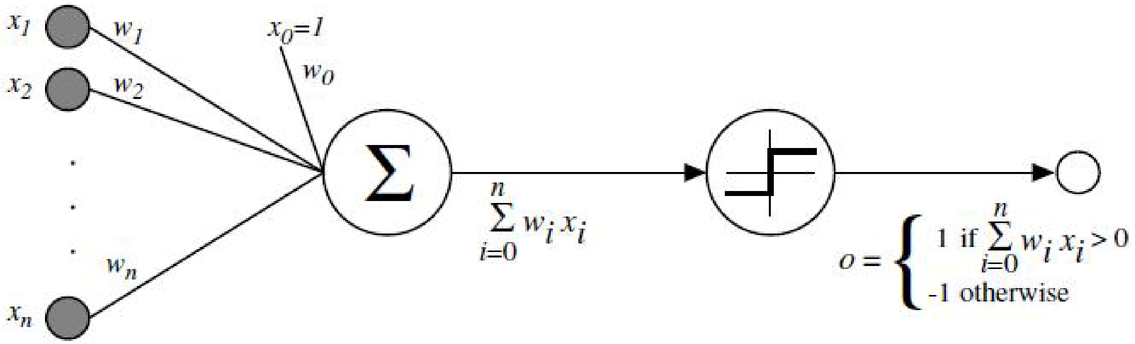
$\boldsymbol{w}_1 = (0, \cdots, 0)$

Let $\alpha = \frac{R^2}{\gamma}$

$$\|\boldsymbol{w}_{k+1} - \alpha \boldsymbol{u}\|^2 \le \|\boldsymbol{w}_1 - \alpha \boldsymbol{u}\|^2 - kR^2$$

$$k_{max} = \frac{\|\boldsymbol{w}_1 - \alpha \boldsymbol{u}\|^2}{R^2}$$

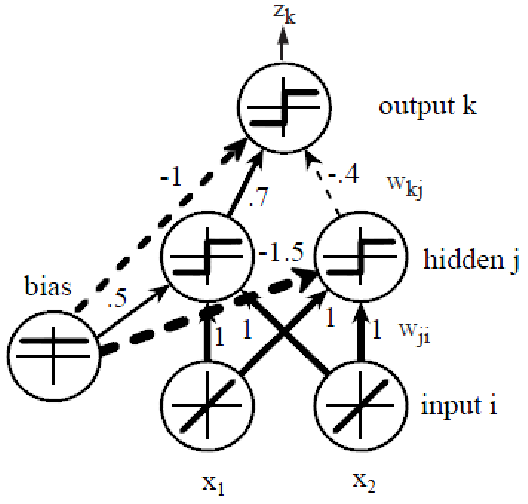
Perceptron vs. Multi-layer Perceptron



x_1
w_1
x_2
w_2
$x_0=1$
w_0
w_n
x_n
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Gradient descent

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta(k) \sum_{i \in I_M^k} \boldsymbol{x}_i y_i$$



How to Learn?

Backpropagation

© Deng Cai, College of Computer Science, Zhejiang University

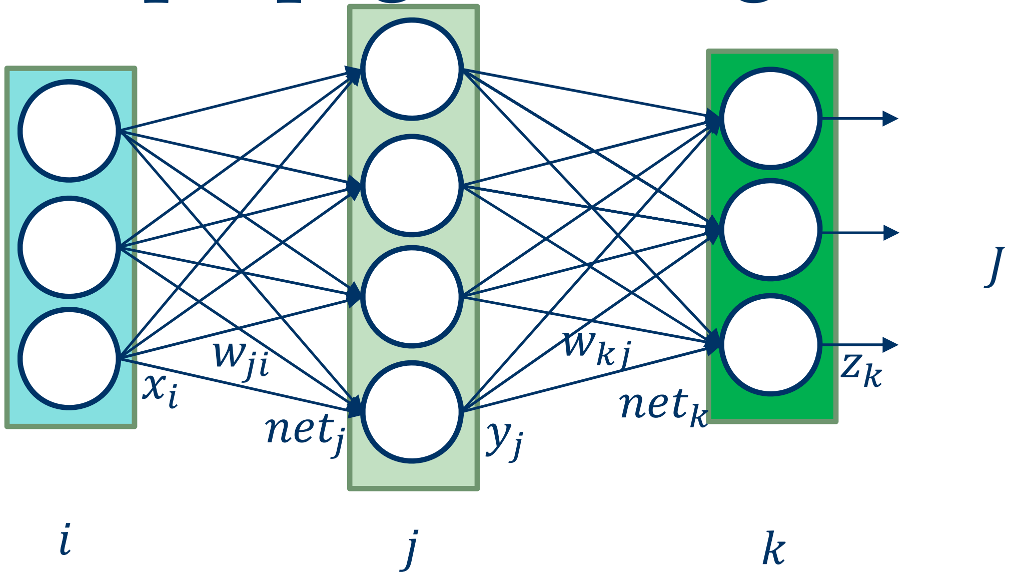
Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by “Perceptrons”
- 1970s, Backpropagation was introduced and wasn’t valued
  - Paul J. Werbos (born 1947)
  - *Paul Werbos (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University.*
  - “Mentioned the possibility of applying this principle to artificial neural network”
- 1986, Backpropagation was reinvented
  - *Rumelhart, David E.; **Hinton, Geoffrey E.**; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088): 533–536.*
  - “Showed through computer experiments that this method can generate useful internal representations of incoming data in hidden layers of neural networks”

BP & Geoffrey Hinton

- In 1970, when the first AI winter came, **Geoffrey Hinton**, a 23-years-old young man, gained his bachelor's degree of experimental psychology from Cambridge.
- And then, he chose to get a PhD at the University of Edinburgh, on AI, nearly if not exactly equal to NN at that time.
- His friends were confused about his decision and thought he must be crazy, because NN had been proved to be sheer nonsense, by the book "*Perceptrons*".

Backpropagation Algorithm



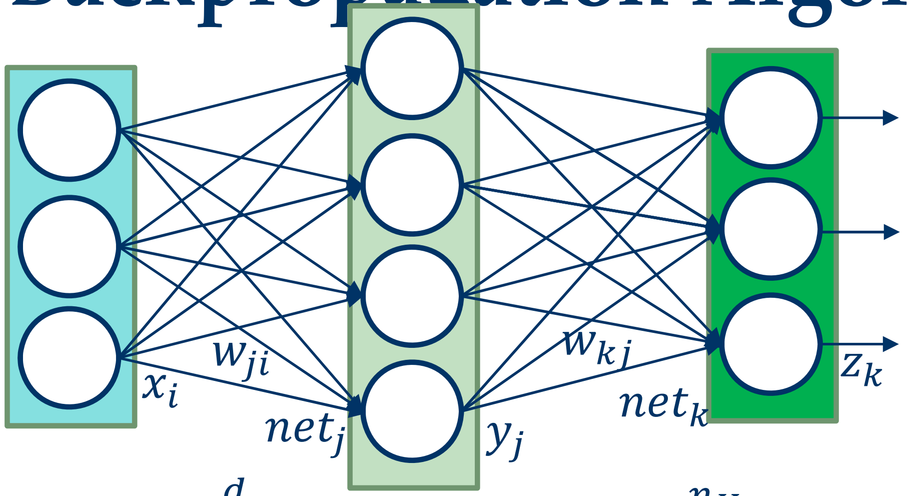
$$\Delta \boldsymbol{w} = -\eta \nabla J = -\eta \frac{\partial J}{\partial \boldsymbol{w}} \qquad \Delta w_{mn} = -\eta \frac{\partial J}{\partial w_{mn}}$$

- Where η is the learning rate which indicates the relative size of the change in weights

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t)}$$

- where t indexes the particular pattern presentation

Backpropagation Algorithm



$$\boxed{J = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2}$$

$$net_j = \sum_{j=0}^{d} x_i w_{ji} \qquad net_k = \sum_{j=0}^{n_H} y_j w_{kj}$$

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k) \cdot f'(net_k) \cdot y_j$$

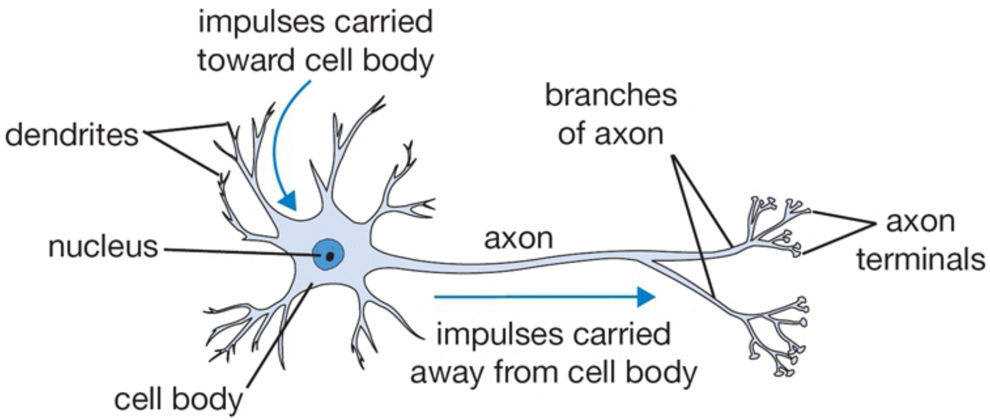
$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \left(\sum_{k=1}^{c} (z_k - t_k) \cdot f'(net_k) \cdot w_{kj} \right) \cdot f'(net_j) \cdot x_i$$

Natural Neural Net Models

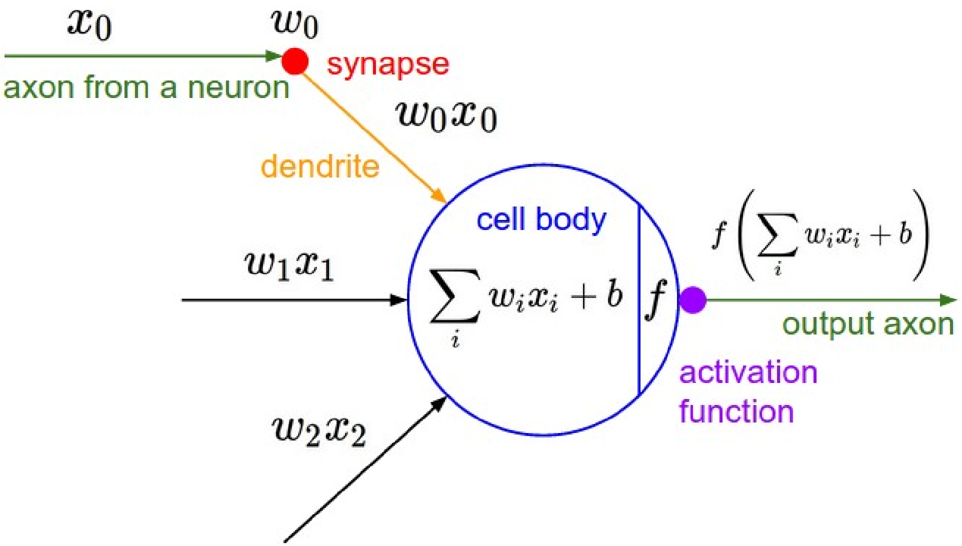
- Human brain consists of very large number of neurons (between 10^{10} to 10^{12})
- No. of interconnections per neuron is between 1K to 10K
- Total number of interconnections is about 10^{14}
- Damage to a few neurons or synapse (links) does not appear to impair overall performance significantly (robustness)

The Artificial Neural Network

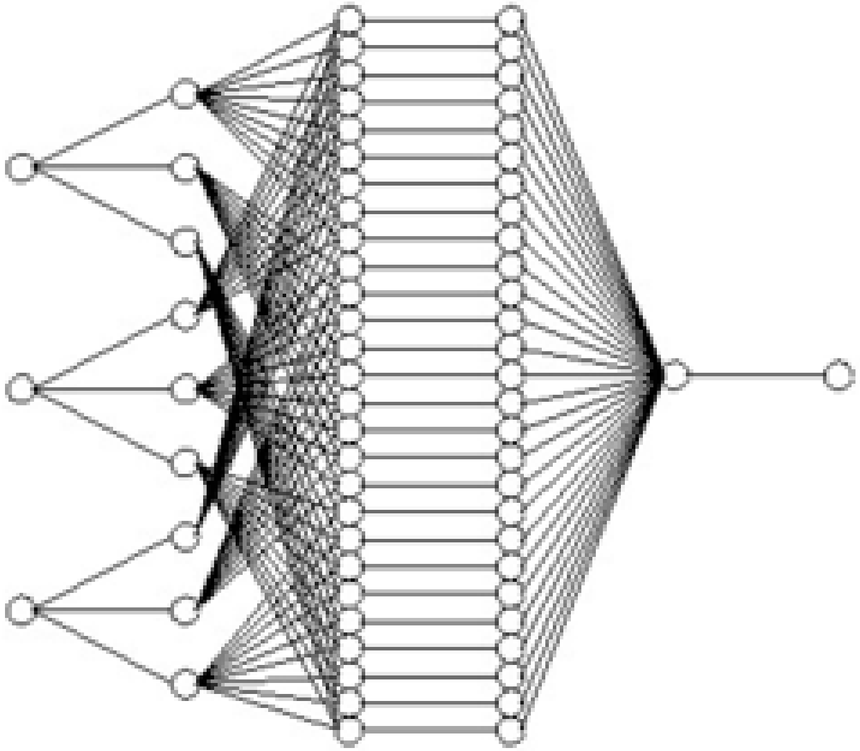


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

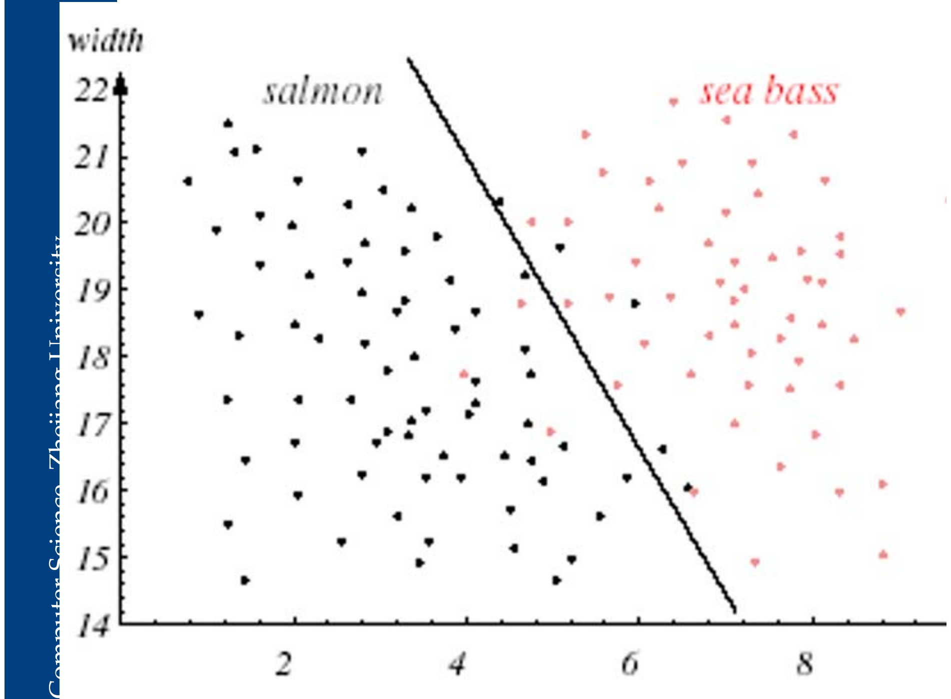
- A cartoon drawing of a biological neuron

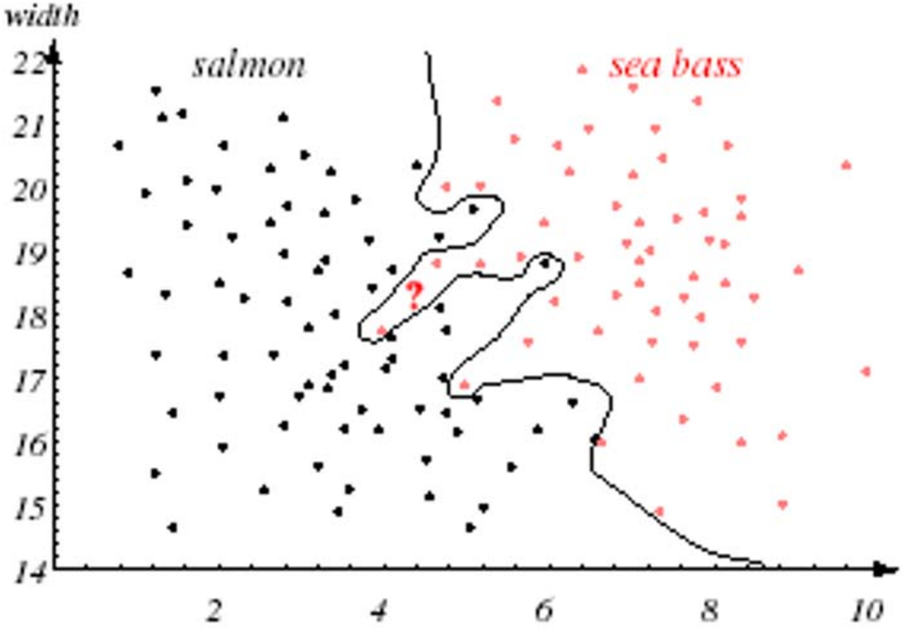


The Artificial Neural Network (Deep Learning)



Linear (simple) decision boundary VS. Complex decision boundary





- Which one is better?

Bias-variance Decomposition

- $\mathrm{EPE}(f) = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy$
- Expected prediction error (expected loss) =

(bias)^2 + variance + noise

- (bias)^2:

$$\int \{E_D(f(\boldsymbol{x}; D)) \; - E(y|\boldsymbol{x})\}^2 p(\boldsymbol{x}) d\boldsymbol{x}$$

- variance:

$$\int E_D\left\{\left[f(\boldsymbol{x}; D) - E_D(f(\boldsymbol{x}; D))\right]^2\right\} p(\boldsymbol{x}) d\boldsymbol{x}$$

- noise:

$$\int \mathrm{var}(y|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

General Feedforward Operation

- Case of c output units

$$g_k(\boldsymbol{x}) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

$$k = 1, \cdots, c$$

- Hidden units enable us to express more complicated nonlinear functions and extend classification capability
- Assume for now that all activation functions are identical

- Question: Can every decision boundary be implemented by a three-layer network described by the above equation?

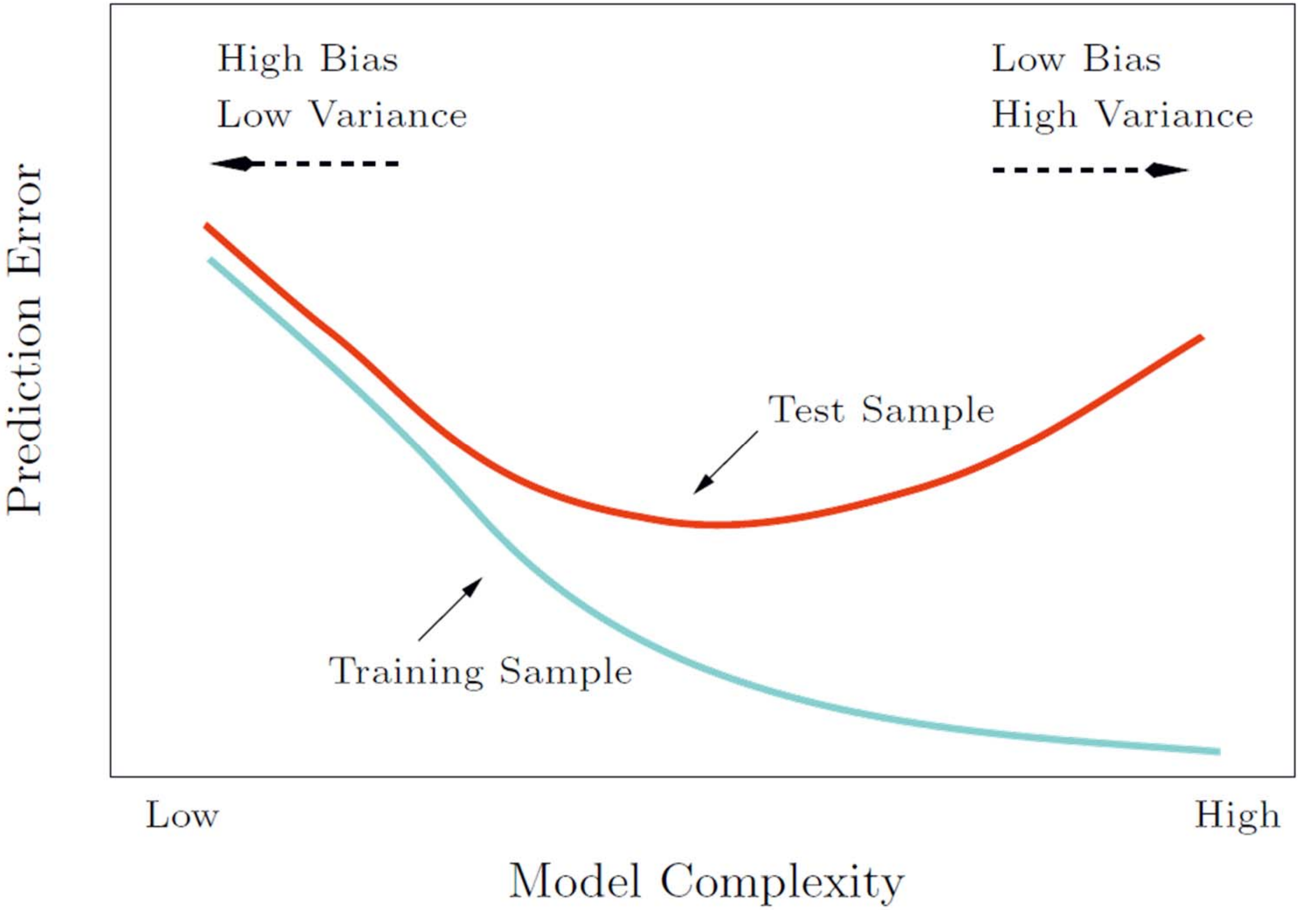
Expressive Power of Multi-layer Networks

- Answer: Yes (due to A. Kolmogorov)
  - Any continuous function from input to output can be implemented in a three-layer net, given sufficient number of hidden units n_H, proper nonlinearities, and weights.

- Any continuous function $g(\boldsymbol{x})$ defined on the unit hypercube $I^n (I = [0,1]$ and $n \geq 2)$ can be represented in the following form:

$$g(\boldsymbol{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{i=1}^{d} \Phi_{ij}(x_i) \right)$$

for properly chosen functions Ξ_j and Φ_{ij}


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Training Sample
Low
High
Model Complexity

Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules

Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules
    - Backpropagation

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by “Perceptrons”
- 1970s, Backpropagation was introduced and wasn’t valued
- 1986, Backpropagation was reinvented
- 1989, **Convolutional Neural Network (CNN)** by Yann LeCun
- 1980s, Recurrent Neural Network was created
- 1997, **Long Short Term Memory networks(LSTM)** by Hochreiter & Schmidhuber

Yann Lecun

- **Yann Lecun**
- Burn in Paris in 1960.
- In 1987, after receiving PhD in France, he worked as a postdoctoral researcher with **Hinton** for one year, and then in Bell Labs.

BP & CNN

- In 1989, **Yann Lecun** published the paper "*Backprop-agation applied to handwritten zip code recognition*".

- In the paper, he trained a CNN model with about 10,000 hand-written-digit images provided by the US Postal System and achieved an test error rate of 5%.

- A CNN-based commercial software was also developed by Lecun to recognize the hand-written digits on checks. It had occupied about 20% of the market share in the late 1990s US.

Neural Network (Deep Learning) History & Progress

- 1957, Perceptron created by Frank Rosenblatt (**60 years ago**)
- 1969, the first AI winter caused by “Perceptrons”
- 1970s, Backpropagation was introduced and wasn’t valued
- 1986, Backpropagation was reinvented
- 1989, **Convolutional Neural Network (CNN)** by Yann LeCun
- 1980s, Recurrent Neural Network was created
- 1997, **Long Short Term Memory networks(LSTM)** by Hochreiter & Schmidhuber
- For about 20 years, the second winter brought by SVM

The second winter

- However, at the time when CNN was ready to enjoy its booming, a researcher in Bell Labs, whose office was quite near to Yann Lecun's, brought the second winter to NN research.

- **Vladmir Vapnik**
- Burn in the former Soviet Union in 1936
- Migrated to US in 1990 and worked in **Bell Labs**.
- Early in 1963, Vapnik invented the **Support Vector Machine (SVM)** algorithm.

The second winter

- The SVM, as an exquisite classification algorithm, started to find its prosperity in image and voice recognition, in the early 1990s.

- In Bell Labs on the corridor, **Yann Lecun and Vapnik** often had fever discussions on the strength and weakness of NN and SVM.

- On the recognition task of hand-written digits, the SVM had kept making progress:
  - By 1998, the error rate had been reduced to 0.8% .
  - By 2002, 0.56%

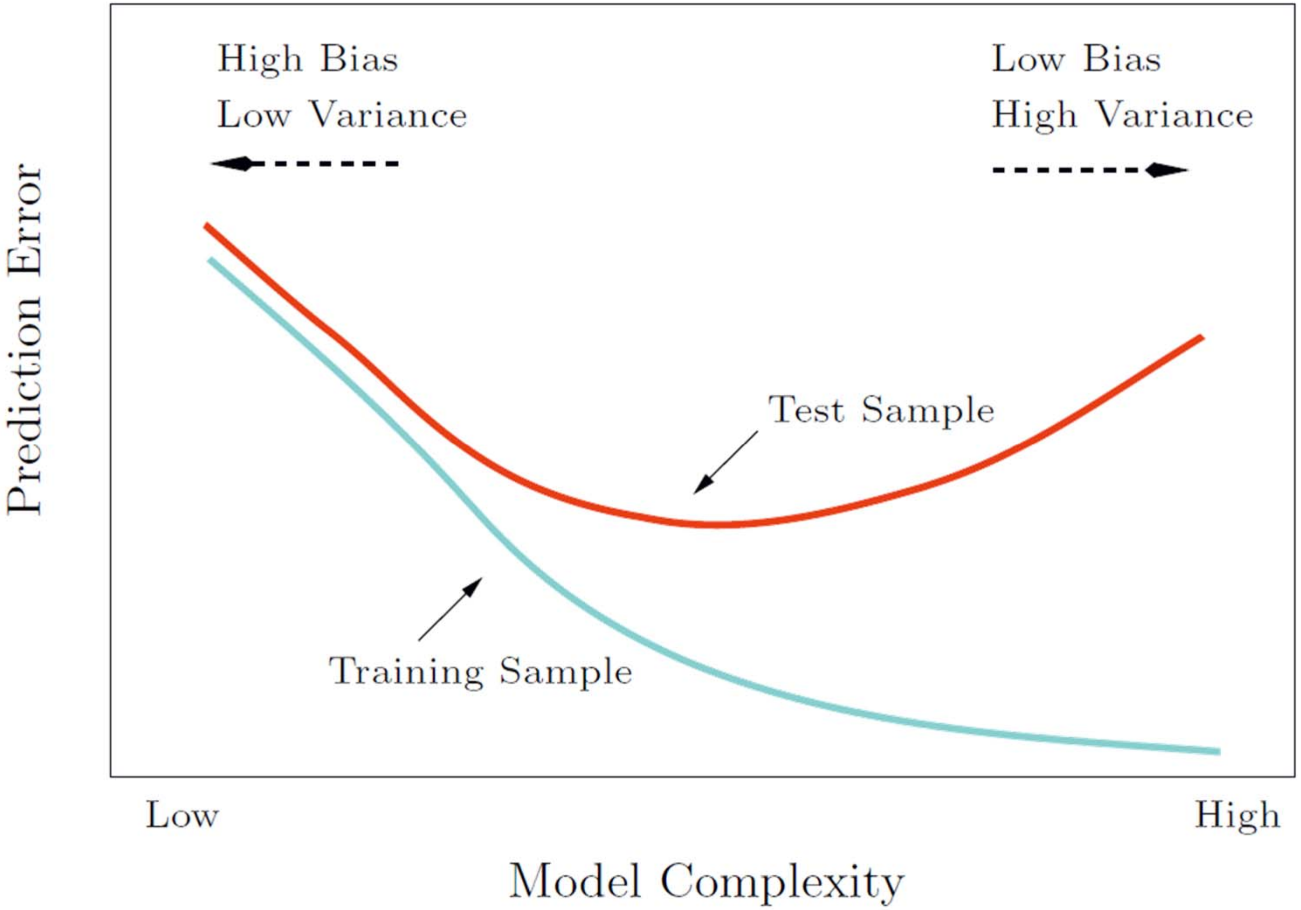
- It had obviously exceeded the contemporary performances of NN.

The second winter

Geoff Hinton sits at the head of the table at the 2003 Vancouver workshop, where he and his fellow academics convinced the Canadian Institute for Advanced Research (CIFAR) to agree to fund their work.

The second winter

- In 2004, Canadian Institute for Advanced Research (CIFAR) agreed to support the NN research by giving about $10 million over 10 years.
- By then, **CIFAR** was **the only organization** funding the research of NN.
- **Without the support of CIFAR, the AI research of human beings may have to stagger in darkness for many more years.**


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Training Sample
Low
High
Model Complexity

Something about 'deep' and 'learning'

- For neural networks, being deeper means being more powerful and expressive.

- So, why not going deeper at the very beginning?
  - The vanishing gradient problem.
  - Expensive computation budget.

- The advances in handling these problems witnessed by the last decade are the prerequisites for the popularization of deep neural networks, i.e., deep learning.

Artificial Neural Net Models

- Artificial Neural nets are specified by
  - Net topology
  - Node (processor) characteristics
  - Training/learning rules

Activation Function

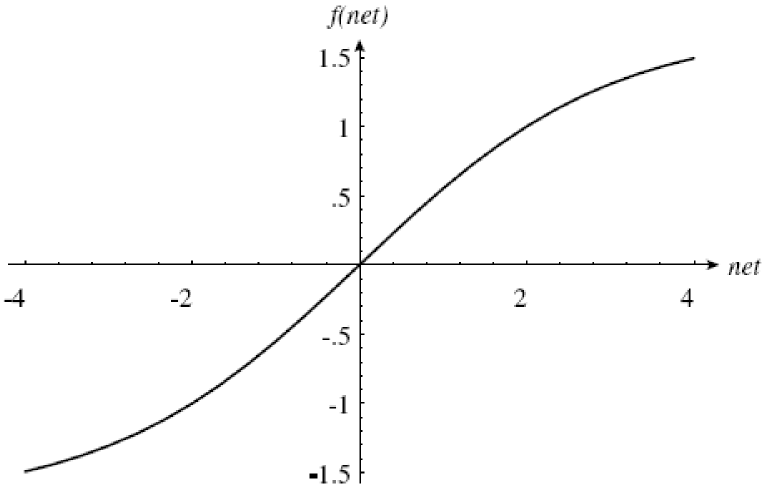
- Activation Function *f*
  - Must be non-linear (otherwise, 3-layer network is just a linear discriminant) and saturate (have max and min value) to keep weights and activation functions bounded
  - Activation function and its derivative must be continuous and smooth; optionally monotonic
  - Choice may depend on the problem. Eg. Gaussian activation if the data comes from a mixture of Gaussians
  - Eg: sigmoid (most popular), polynomial, tanh

- Parameters of activation function (e.g. Sigmoid)
  - Centered at 0, odd function f(-net) = -f(net) (anti-symmetric); leads to faster learning
  - Depend on the range of the input values

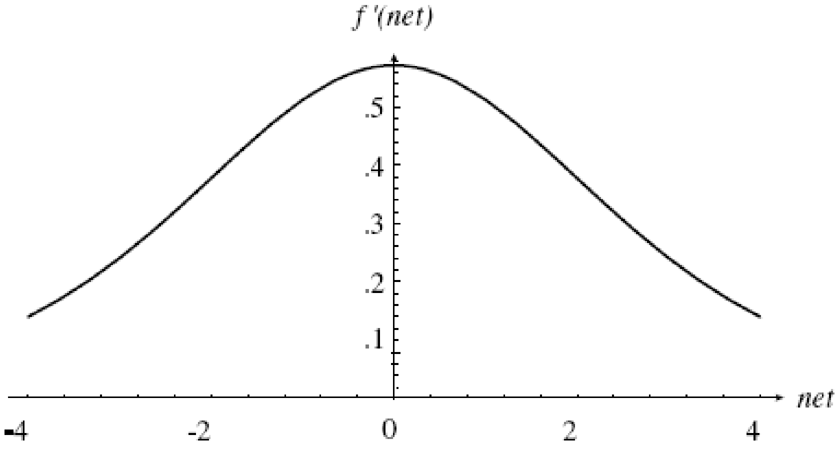
Activation Function

$$f(net) = a\ \tanh(b\ net) = a\left[\frac{1-e^{b\ net}}{1+e^{b\ net}}\right] = \frac{2a}{1+e^{-b\ net}} - a$$

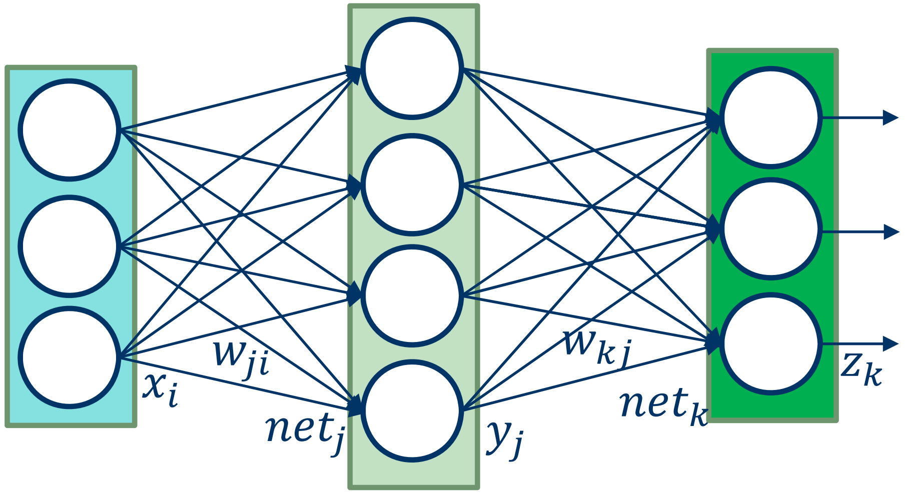
The anti-symmetric sigmoid function:
f(-x) = -f(x).
a = 1.716, b = 2/3.



First order derivative



Backpropagation Algorithm

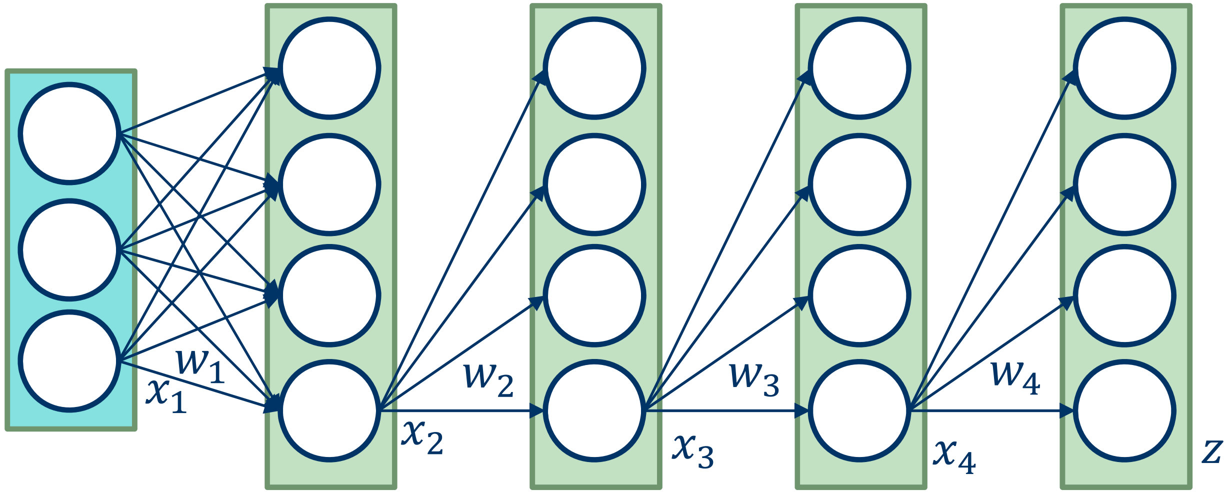


$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k) \cdot f'(net_k) \cdot y_j$$

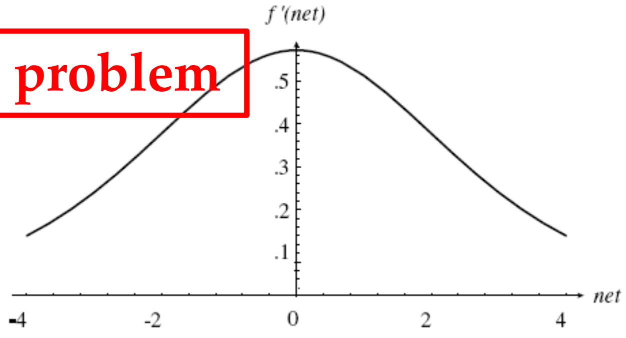
$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= (z_k - t_k) \cdot f'(net_k) \cdot w_{kj} \cdot f'\left(net_j\right) \cdot x_i$$

Backpropagation Algorithm



vanishing gradient problem



First order derivative

$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot x_4$$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot x_3$$

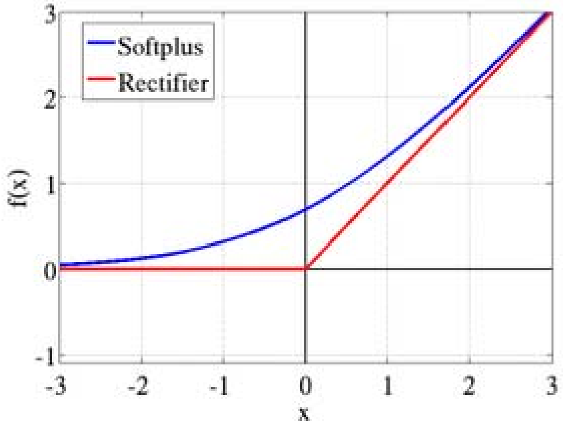
$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot w_3 f'(net_2) \cdot x_2$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z} \cdot \overbrace{f'(net_4) \cdot w_4}^{<1} \cdot \overbrace{f'(net_3) \cdot w_3}^{<1} \overbrace{f'(net_2) \cdot w_2}^{<1} \overbrace{f'(net_1)}^{<1} \cdot x_1$$

2011, ReLU RevoLUtion

- In 2011, Canadian scholar Xavier Glorot and Yoshua Bengio published the paper "*Deep Sparse Rectifier Neural Networks* ".
- The paper proposed a new activation function, **Rectified Linear Unit (ReLU)**:

ReLU(x) = max (0, x)


Softplus
Rectifier
f(x)
x