Slides adapted from Vlad Morariu

CMSC 422 Introduction to Machine Learning

Lecture 20 Deep Learning I

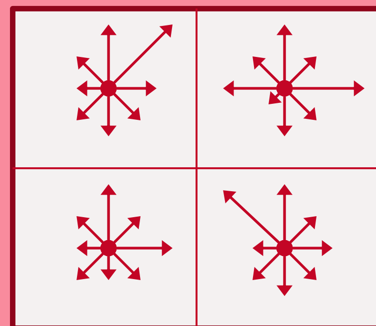
Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

History of Neural Networks

Standard computer vision pipeline

→ feature extraction →



features

→ classification →

"cat" or "background"

predicted labels

"cat"

supervision

training

Features: HOG, SIFT, LBP, …

Classifiers: SVM, RF, KNN, …

Features are hand-crafted, not trained

eventually limited by feature quality

Cat image credit: https://raw.githubusercontent.com/BVLC/caffe/master/examples/images/cat.jpg

Deep learning

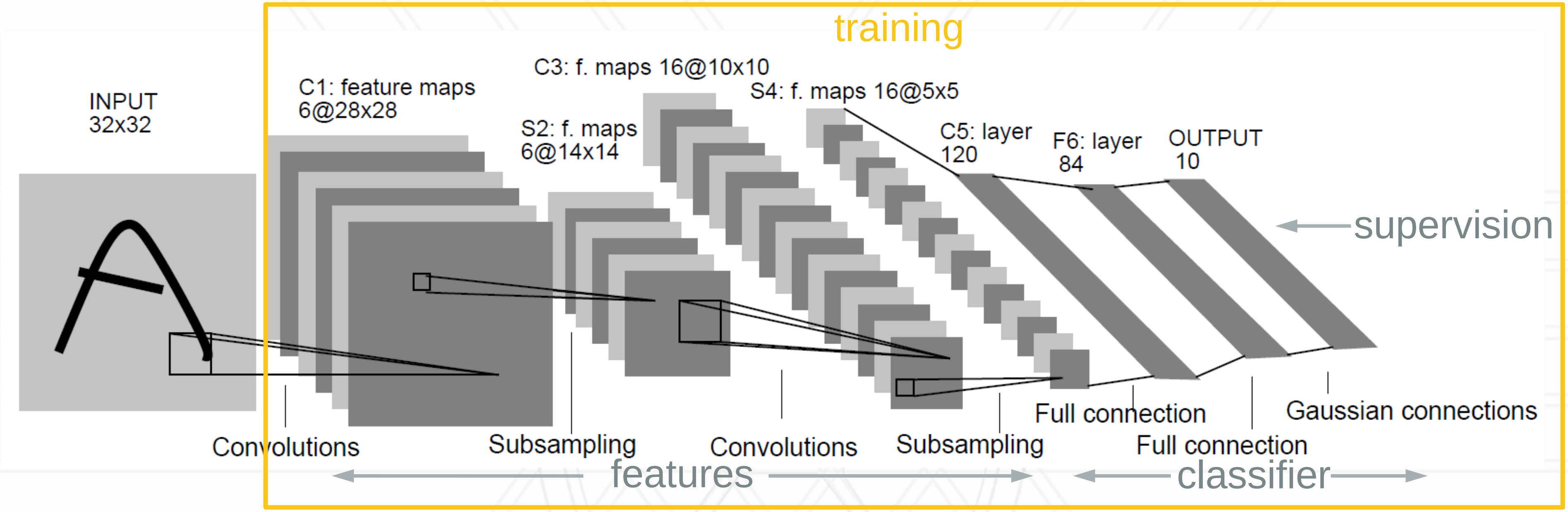
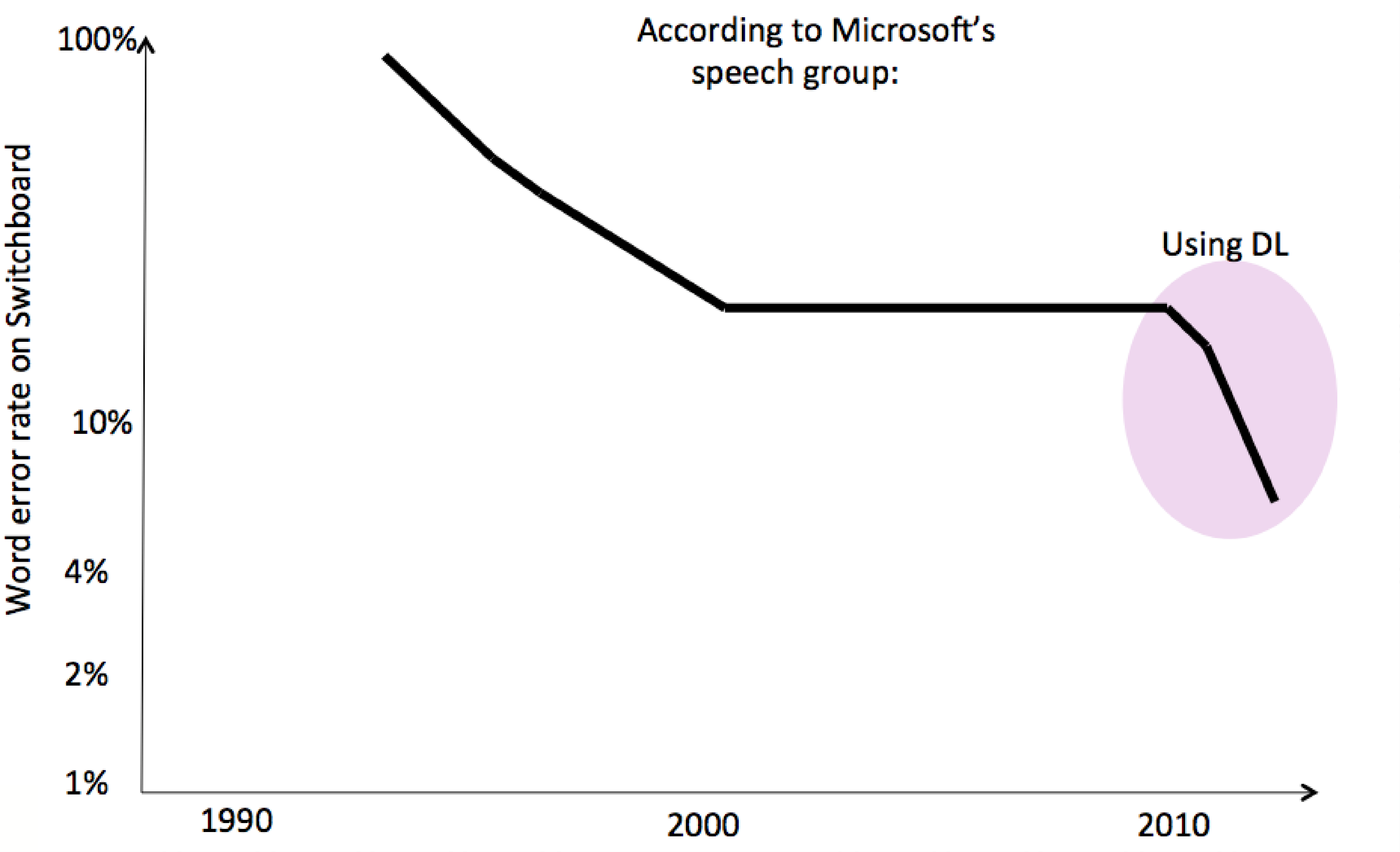


Image credit: LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE, 1998.

Deep learning

- multiple layer neural networks
- learn features and classifiers directly (“end-to-end” training)

Speech Recognition



Slide credit: Bohyung Han

Image Classification Performance

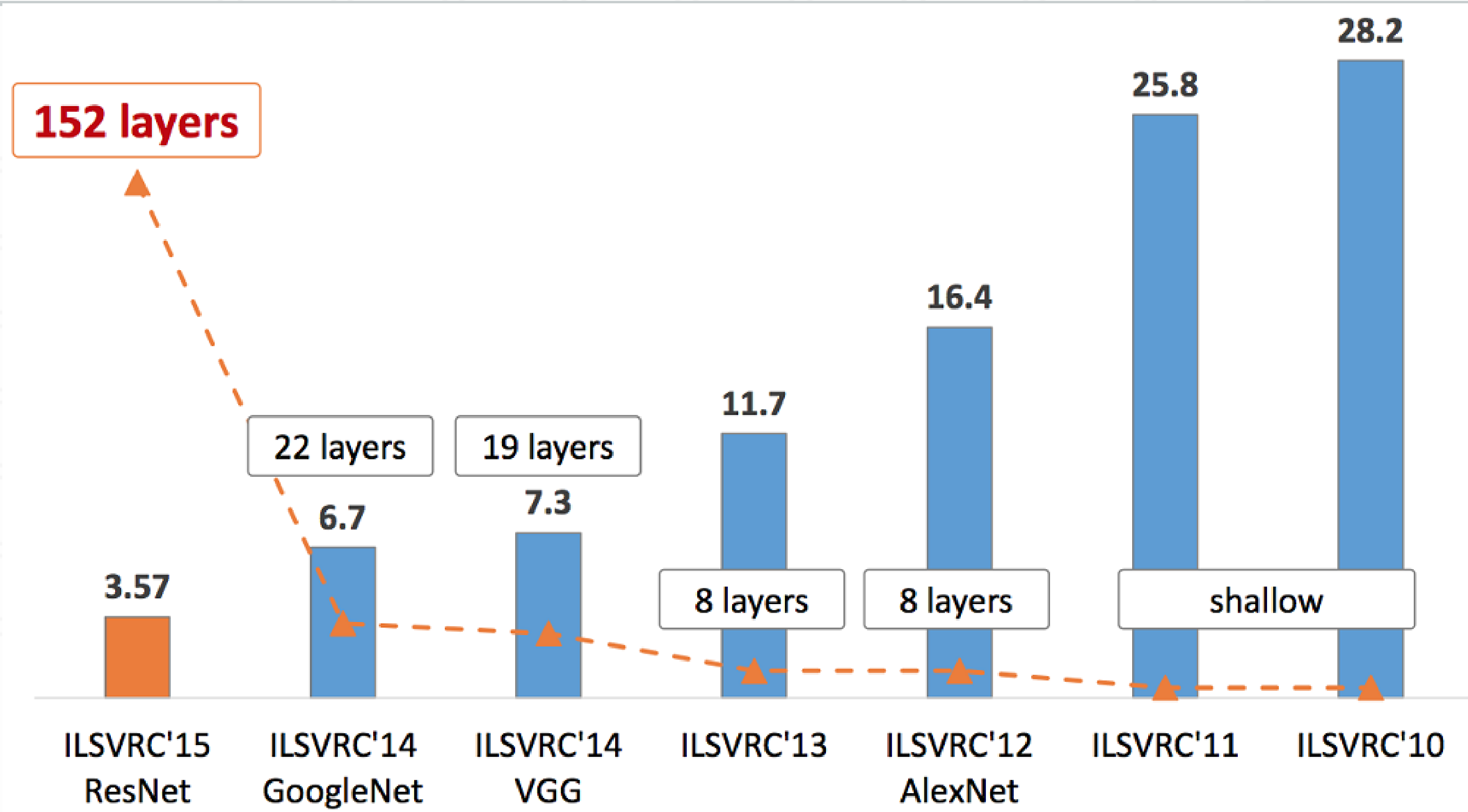


Image Classification Top-5 Errors (%)

Figure from: K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition". arXiv 2015. (slides)

Slide credit: Bohyung Han

Biological inspiration

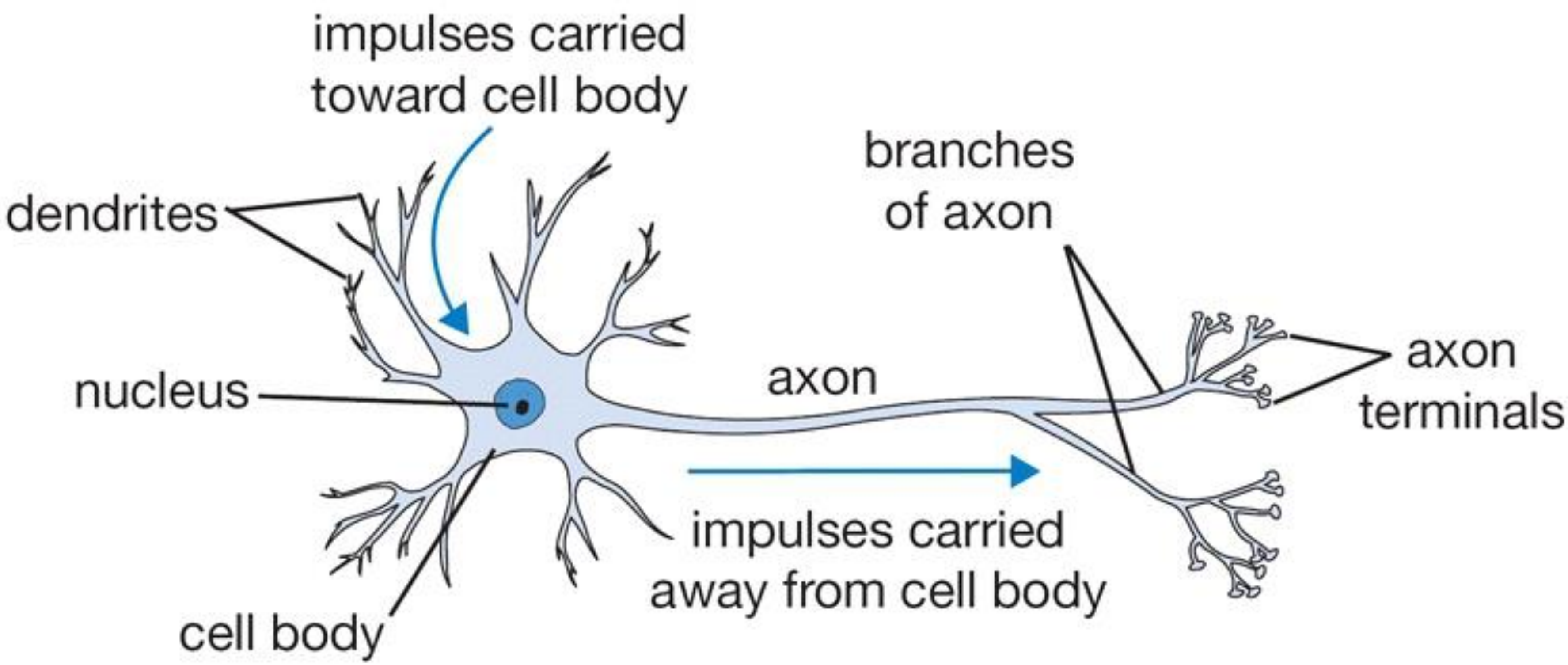
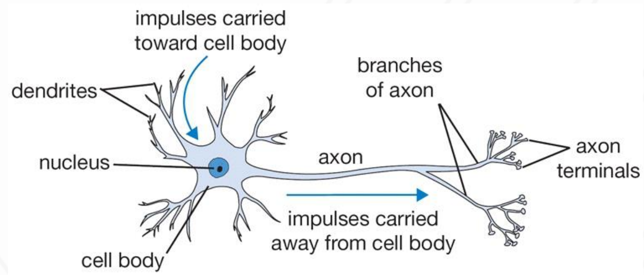


Image source: http://cs231n.github.io/neural-networks-1/

Artificial neuron



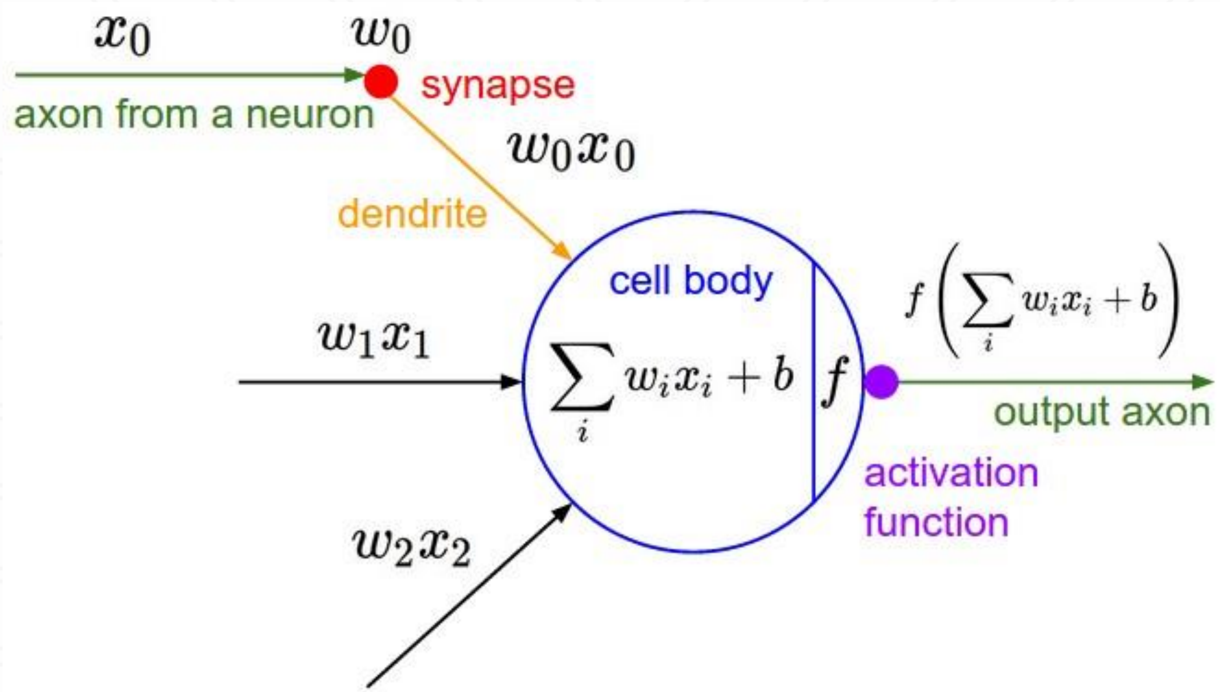


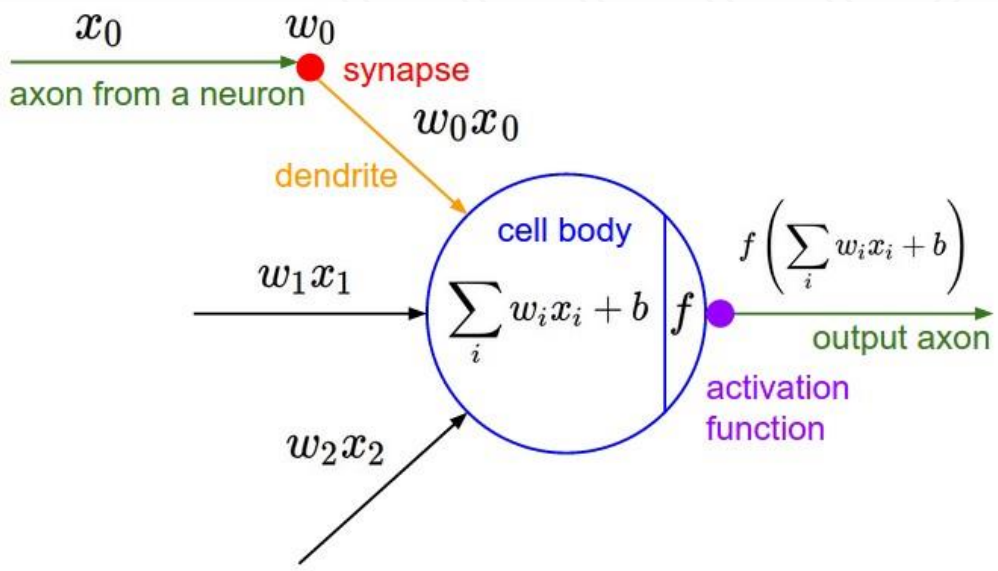
Image source: http://cs231n.github.io/neural-networks-1/

Activation function is usually non-linear

step, tanh, sigmoid

The actual biological system is much more complicated

McCulloch-Pitts Model

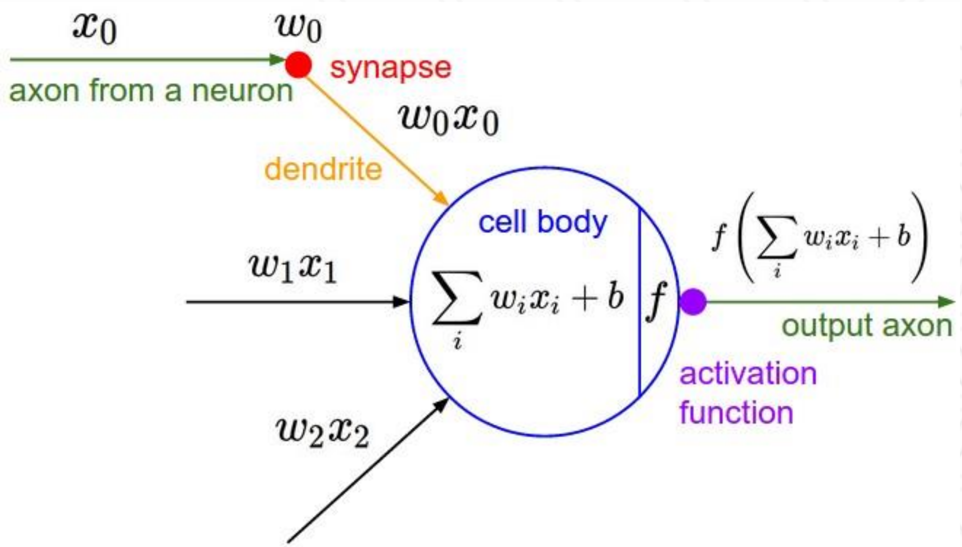


$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Read: https://en.wikipedia.org/wiki/Artificial_neuron

- Threshold Logic Unit (TLU)
  - Warren McCulloch and Walter Pitts, 1943
  - Binary inputs/outputs and Threshold activation function
  - Can represent AND/OR/NOT functions which can be composed for complex functions

Rosenblatt’s Perceptron



$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Perceptron Read: https://en.wikipedia.org/wiki/Perceptron
  - Proposed by Frank Rosenblatt in 1957
  - Based on McCulloch-Pitts model
  - Real inputs/outputs, threshold activation function

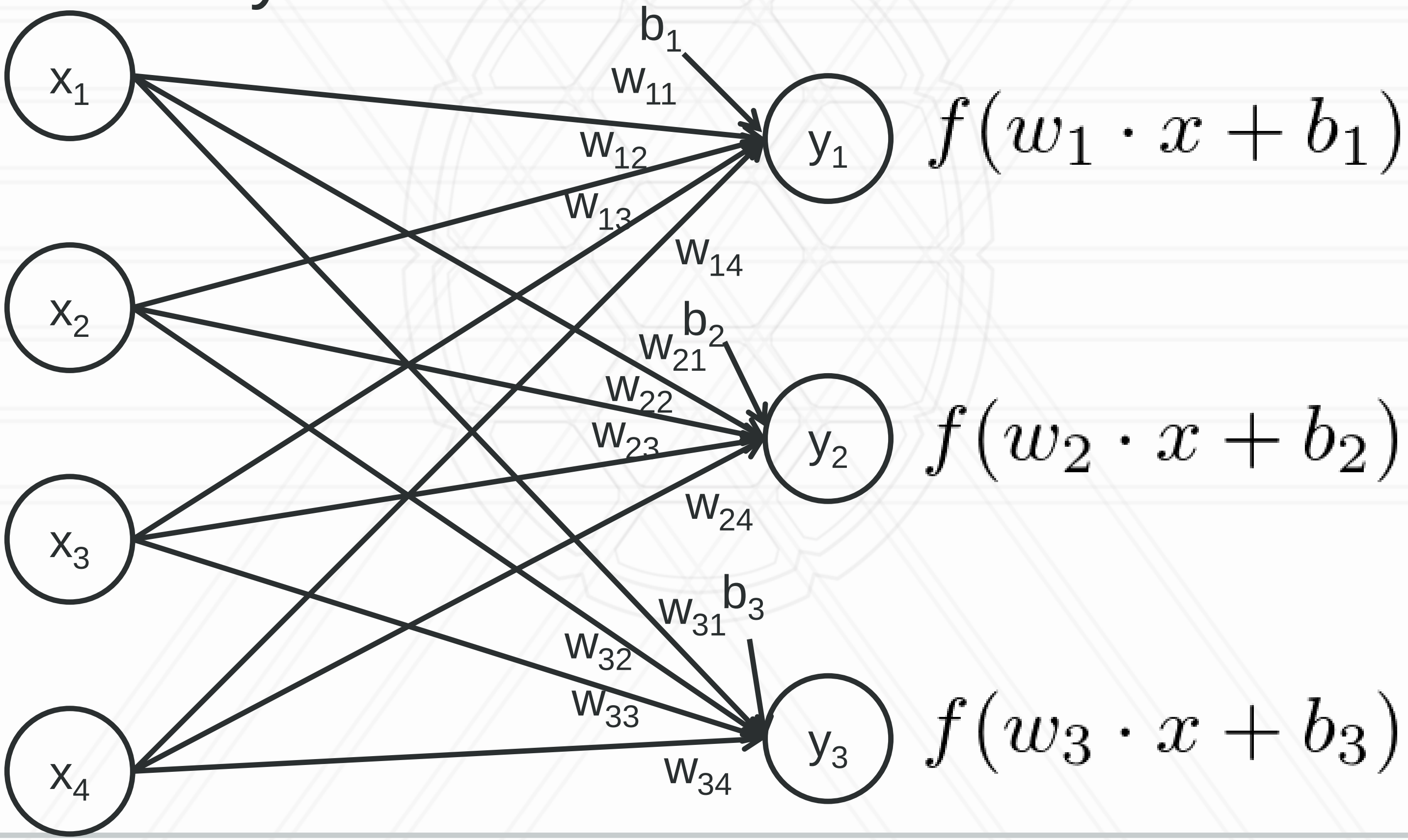
Perceptron Learning

- Given a training dataset of input features and labels $X = [x_1, x_2, \ldots, x_n]$ $Y = [y_1, y_2, \ldots, y_n]$
  $$x_i = [x_{i1} \ldots x_{id}, 1]^\top \quad y_i \in \{0, 1\}$$
- Initialize weights randomly $w = [w_1, \ldots, w_d, b]^\top$
- For each example in training set
  - Classify example using current weights $\hat{y} = f(w \cdot x)$
  - Update weights $w \leftarrow w + (y_i - \hat{y}_i)x$
- If data is linearly separable, convergence is guaranteed in a bounded number of iterations

https://en.wikipedia.org/wiki/Perceptron

Multiple output variables

Weights to predict multiple outputs can be learned independently



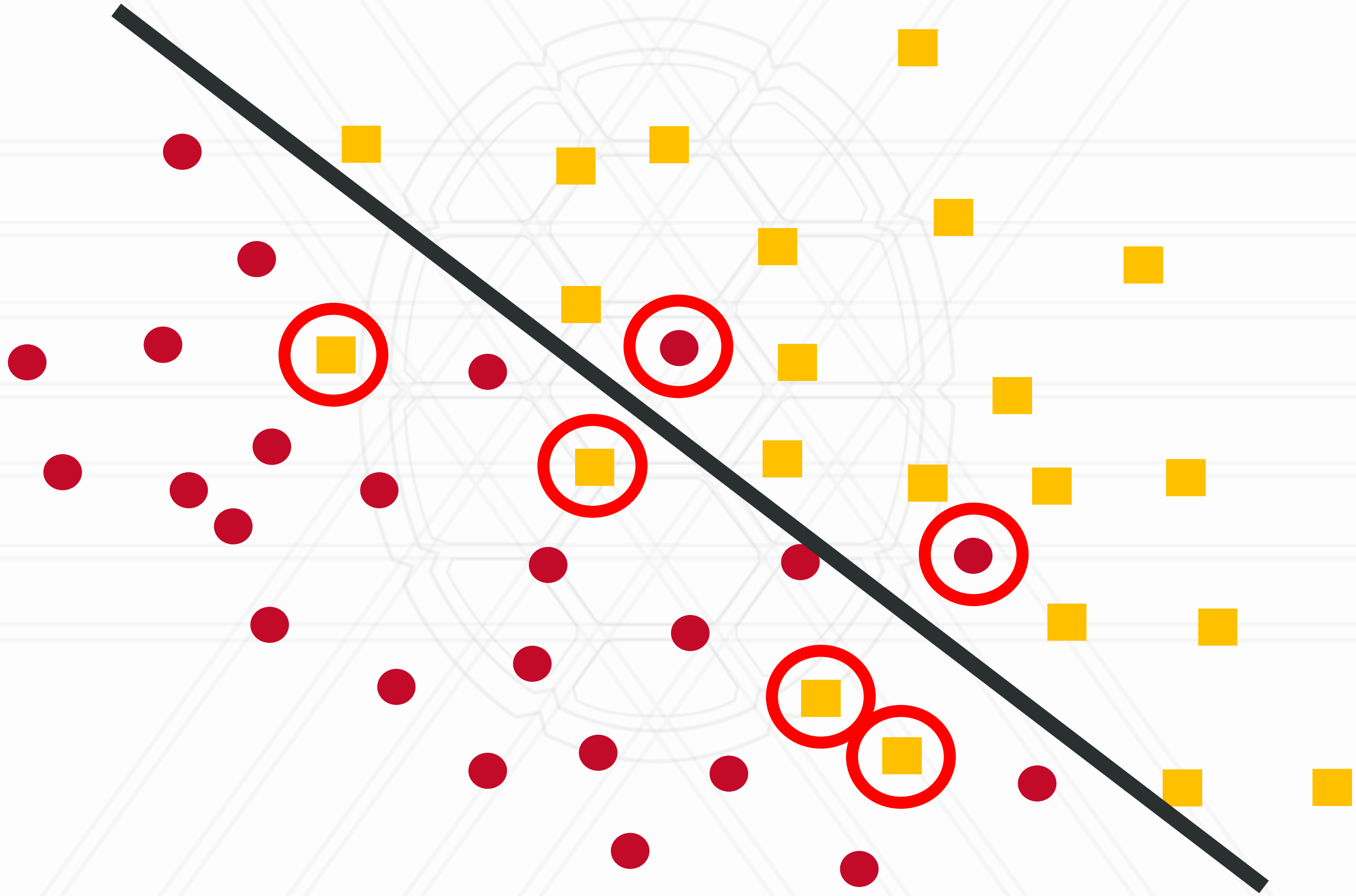
Perceptron success

- Implemented as custom-built hardware, the “Mark I Perceptron”
  - Input: photocells
  - Weights: potentiometers
  - Weight updates: electric motors
- Demonstrated ability to classify 20x20 images
- Generated lots of AI excitement

- In 1958, the New York Times reported the perceptron to be
  - "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

https://en.wikipedia.org/wiki/Perceptron

Linear Classifier

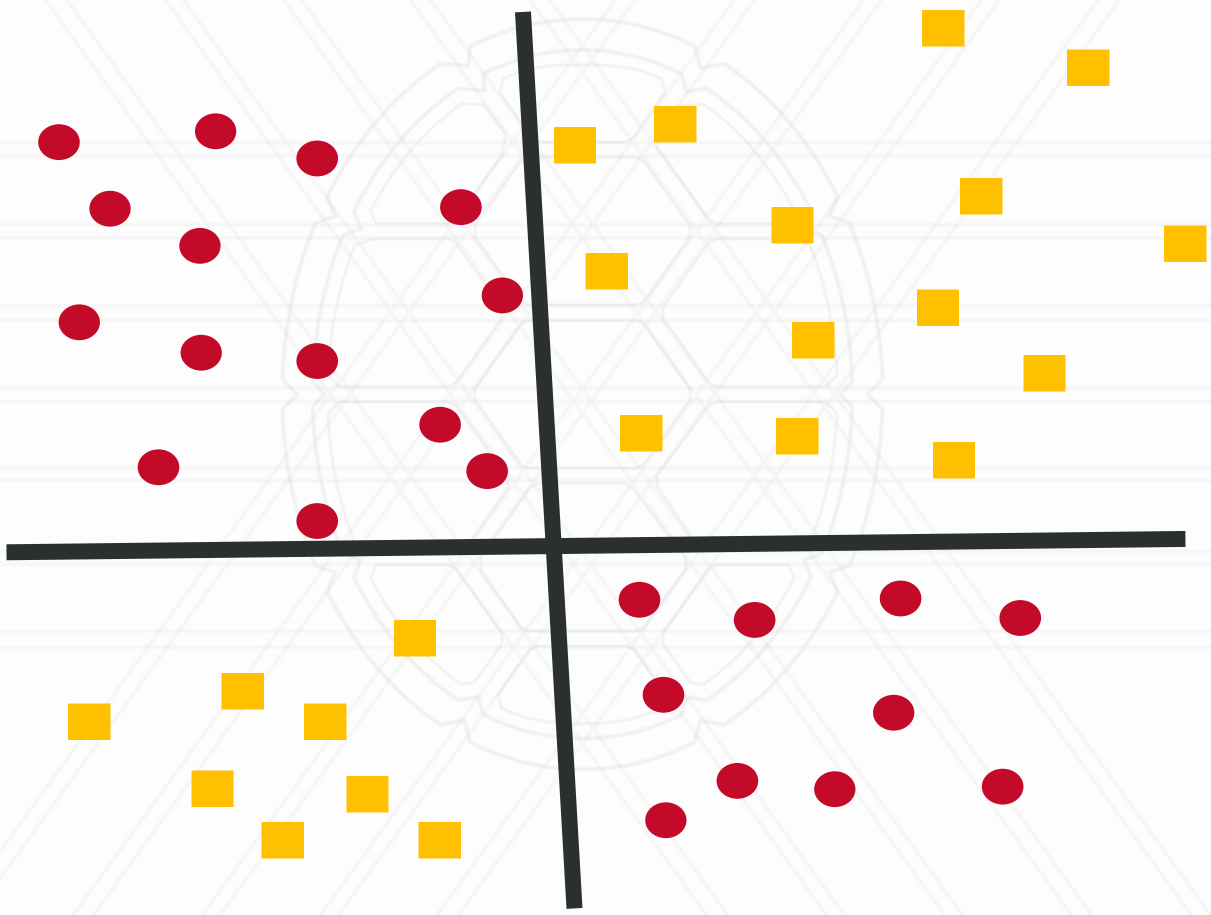


Slide credit: Bohyung Han

Nonlinear Classifier

Slide credit: Bohyung Han

Nonlinear Classifier – XOR problem



Perceptron limitations

- If there are multiple separating hyperplanes, learning will converge to one of them (not the *optimal* one)
- If training set is not linearly separable, training will fail completely

- Marvin Minsky and Seymour Papert, “Perceptrons”, 1969
  - Proved that it was impossible to learn an XOR function with a single layer perceptron network

- Led to the “AI Winter” of the 1970’s

https://en.wikipedia.org/wiki/Perceptron

Multi-Layer Perceptron (MLP)

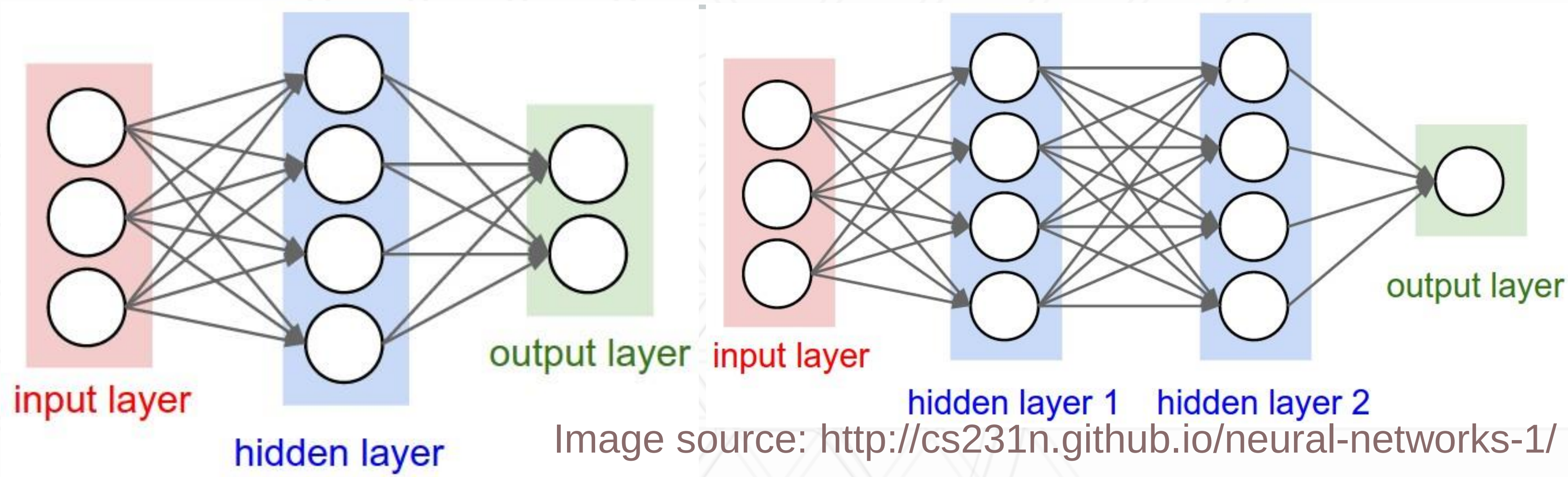
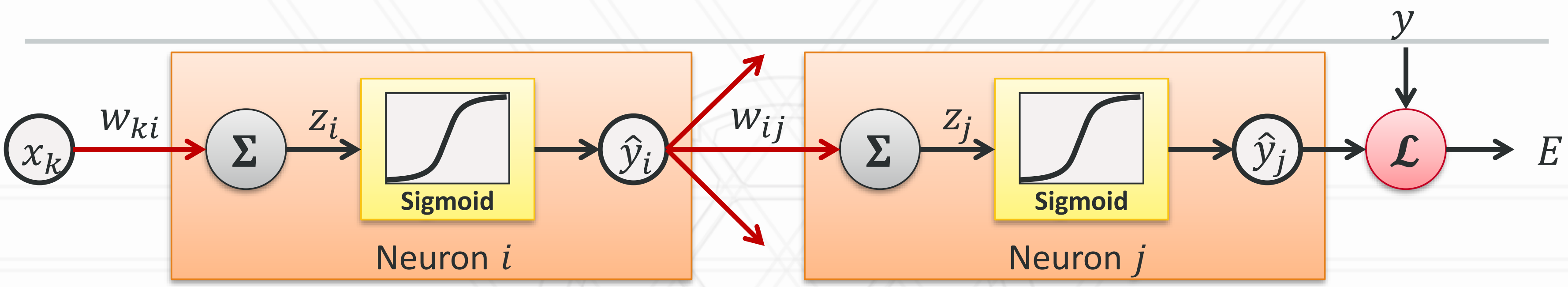


Image source: http://cs231n.github.io/neural-networks-1/

- Activation function need not be a threshold
- Multiple layers can represent XOR function
- But perceptron algorithm cannot be used to update weights
    - Why? Hidden layers are not observed!

https://en.wikipedia.org/wiki/Multilayer_perceptron

Multi-Layer: Backpropagation



$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial \hat{y}_j}\frac{d\hat{y}_j}{dz_j}$$

$$\frac{\partial E}{\partial \hat{y}_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{dz_j}{d\hat{y}_i} = \sum_j w_{ij}\frac{\partial E}{\partial z_j} = \sum_j w_{ij}\frac{\partial E}{\partial \hat{y}_j}\frac{d\hat{y}_j}{dz_j}$$

$$\frac{\partial E}{\partial w_{ki}} = \sum_n \frac{\partial E}{\partial \hat{y}_i^n}\frac{d\hat{y}_i^n}{dz_i^n}\frac{\partial z_i^n}{\partial w_{ki}} = \sum_n \frac{d\hat{y}_i^n}{dz_i^n}\frac{\partial z_i^n}{\partial w_{ki}}\sum_j w_{ij}\frac{\partial E}{\partial \hat{y}_j^n}\frac{d\hat{y}_j^n}{dz_j^n}$$

Slide credit: Bohyung Han

Stochastic Gradient Descent (SGD)

Update weights for each sample

$$E = \frac{1}{2}(y^n - \hat{y}^n)^2 \qquad \boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \epsilon \frac{\partial E^n}{\partial \boldsymbol{w}_i}$$

+ **Fast, online**
− **Sensitive to noise**

Minibatch SGD: Update weights for a small set of samples

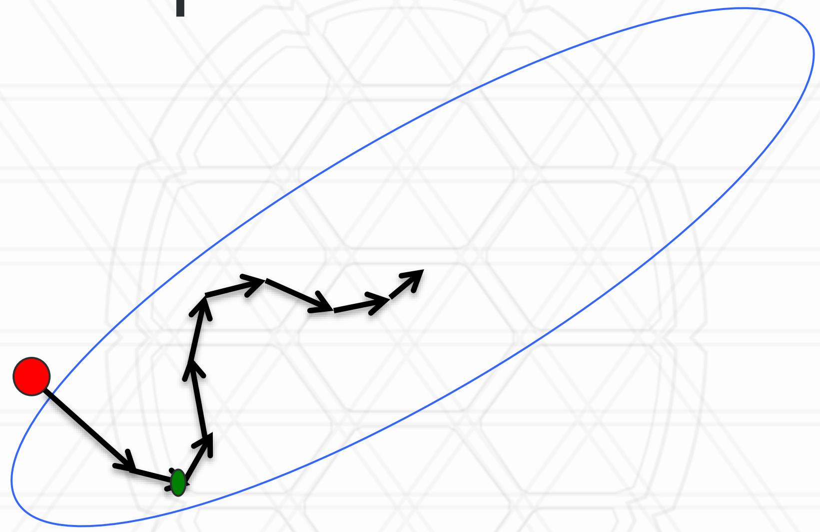
$$E = \frac{1}{2}\sum_{n \in B}(y^n - \hat{y}^n)^2 \qquad \boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \epsilon \frac{\partial E^B}{\partial \boldsymbol{w}_i}$$

+ **Fast, online**
+ **Robust to noise**

Slide credit: Bohyung Han

Momentum

Remember the previous direction



+ **Converge faster**
+ **Avoid oscillation**

$$v_i(t) = \alpha v_i(t-1) - \epsilon \frac{\partial E}{\partial w_i}(t)$$

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

Slide credit: Bohyung Han

Weight Decay

Penalize the size of the weights

$$C = E + \frac{1}{2}\sum_i w_i^2$$

$$w_i(t+1) = w_i(t) - \epsilon\frac{\partial C}{\partial w_i} = w_i(t) - \epsilon\frac{\partial E}{\partial w_i} - \lambda w_i$$

+ **Improve generalization a lot!**

Slide credit: Bohyung Han

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu