Softmax classification

- $Y = \{1, 2, \ldots, n\}$
- Weights $\theta_k \in \mathbb{R}^d$, $k = 1, \ldots, n$
- Given an input $x^{(i)} \in \mathbb{R}^d$, the softmax function which outputs the likelihood of Y being in class j is

$$p\left(Y = j \,\middle|\, x^{(i)}\right) = \frac{e^{\left\langle \theta_j, x^{(i)} \right\rangle}}{\sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}}$$

- Softmax classification is also known as multinomial logistic regression.
- The term "softmax" comes from the fact that if we introduce a parameter c into the formula in the previous slide, i.e. if

$$p\left(Y = j \,\middle|\, x^{(i)}\right) = \frac{e^{c\left\langle \theta_j, x^{(i)} \right\rangle}}{\sum_{k=1}^{n} e^{c\left\langle \theta_k, x^{(i)} \right\rangle}},$$

  then the softmax function converges to a max function as $c \to \infty$.

Overparameterized

- Softmax classification for n classes does not actually require n weight vectors $\theta_1, \ldots, \theta_n$:

$$\begin{aligned} p\left(Y = j \,\middle|\, x^{(i)}\right) &= \frac{e^{\left\langle \theta_j, x^{(i)} \right\rangle}}{\sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}} \\ &= \frac{1}{1 + \sum_{k \neq j} e^{\left\langle \theta_k - \theta_j, x^{(i)} \right\rangle}} \\ &= \frac{1}{1 + \sum_{k \neq j} e^{-\left\langle \tilde{\theta_k}, x^{(i)} \right\rangle}}, \end{aligned}$$

where we denote $\tilde{\theta_k}$ as $\theta_j - \theta_k$. Hence, only $n - 1$ weight vectors are needed.

- In fact, in the 2-class case, this is precisely the formula for the sigmoid function, and thus softmax classification is a generalization of logistic regression:

$$p\left(Y=1\middle|x^{(i)}\right)=\frac{e^{\left\langle\theta_1,x^{(i)}\right\rangle}}{e^{\left\langle\theta_1,x^{(i)}\right\rangle}+e^{\left\langle\theta_2,x^{(i)}\right\rangle}}=\frac{1}{1+e^{-\left\langle\theta_1-\theta_2,x^{(i)}\right\rangle}}$$

$$\begin{aligned}p\left(Y=2\middle|x^{(i)}\right)=\frac{e^{\left\langle\theta_2,x^{(i)}\right\rangle}}{e^{\left\langle\theta_1,x^{(i)}\right\rangle}+e^{\left\langle\theta_2,x^{(i)}\right\rangle}}&=\frac{1}{1+e^{-\left\langle\theta_2-\theta_1,x^{(i)}\right\rangle}}\\&=1-\frac{1}{1+e^{-\left\langle\theta_1-\theta_2,x^{(i)}\right\rangle}}\end{aligned}$$

Cross entropy

Definition

The cross entropy between two probability distributions on a finite set $(x_1, \ldots, x_n)$ is defined to be

$$H(p, q) = -\sum_{i=1}^{n} p(x_i) \log q(x_i).$$

One-hot encoding

If $y = k \in \{1, \ldots, n\}$, then

$$y \to \mathbf{y} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

where 1 appears in the k^{th} coordinate of $\mathbf{y}$ and everything else is 0.

Log-likelihood

$$\ell(\theta) = \sum_{i=1}^{m} \log p\left(Y = y^{(i)} \,\middle|\, x^{(i)}\right)$$

$$= \sum_{i=1}^{m} \log \frac{e^{\left\langle \theta_{y^{(i)}}, x^{(i)} \right\rangle}}{\sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}} \qquad = -H\left(\mathbf{y}^{(i)}, p\left(Y \,\middle|\, x^{(i)}\right)\right)$$

$$= \sum_{i=1}^{m} \left\langle \theta_{y^{(i)}}, x^{(i)} \right\rangle - \log \sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}$$

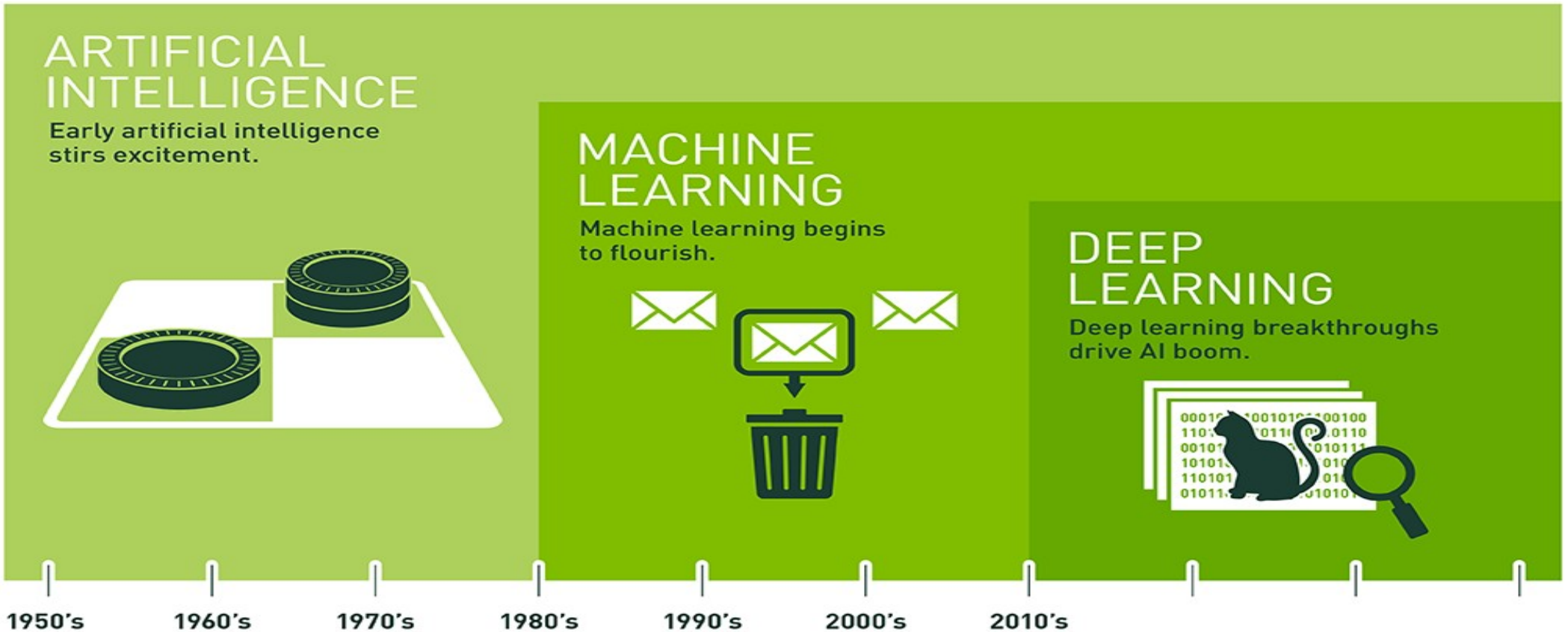
The log-likelihood is the negative of the cross entropy between the one-hot encoding of the labels and the output distribution of the model.

Gradient of softmax log-likelihood

$$
\begin{aligned}
\nabla_{\theta_p} \ell(\theta) &= \sum_{i=1}^{m} \delta_{py^{(i)}} x^{(i)} - \nabla_{\theta_p} \log \sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle} \\
&= \sum_{i=1}^{m} \delta_{py^{(i)}} x^{(i)} - \frac{\nabla_{\theta_p} \sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}}{\sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}} \\
&= \sum_{i=1}^{m} \delta_{py^{(i)}} x^{(i)} - \frac{e^{\left\langle \theta_p, x^{(i)} \right\rangle} x^{(i)}}{\sum_{k=1}^{n} e^{\left\langle \theta_k, x^{(i)} \right\rangle}} \\
&= \sum_{i=1}^{m} \left(\delta_{py^{(i)}} - p\left(Y = p \,\middle|\, x^{(i)} \right) \right) x^{(i)}
\end{aligned}
$$


DEEP LEARNING

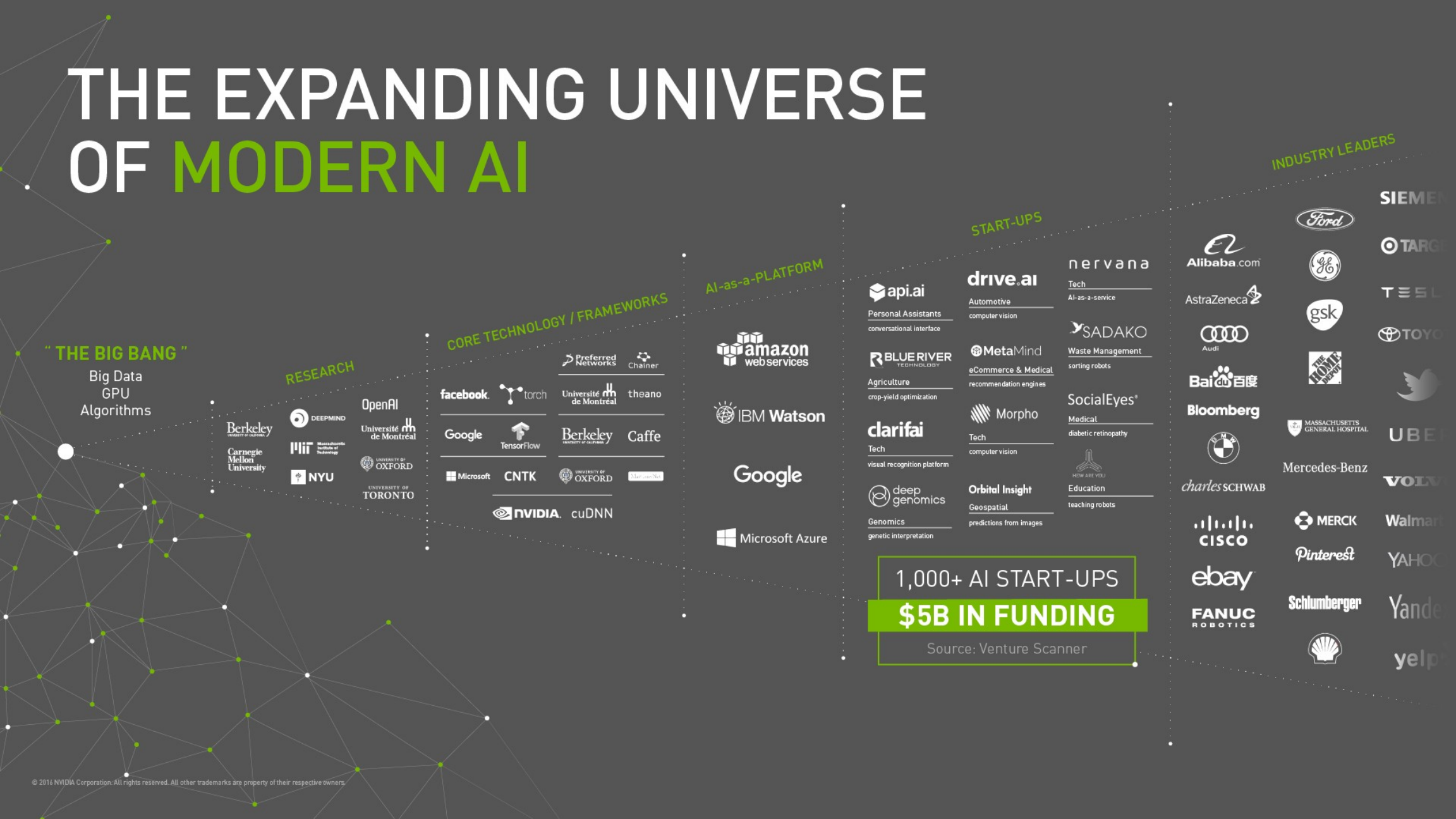
DEFINITIONS



Milestones in AI and Deep Learning

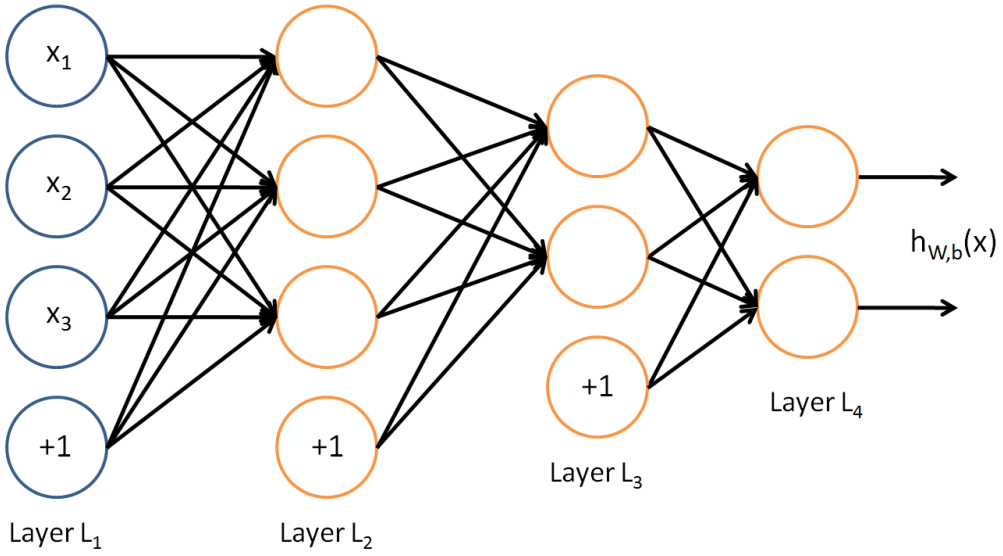
- 1943: First artificial neural networks (McCulloch, Pitts)
- 1956: First conference defining “AI” in Dartmouth (Shannon, Minsky etc.)
- 1986: Training multi-layer neural nets using backpropagation (Hinton, Rumelhart, Williams)
- 1989: Convolutional neural networks (LeCun)
- 1997: Deep blue beats Kasporov in chess
- 2009: Big bang of deep learning, use of GPUs for accelerated training
- 2012: Numerous competitions won with superhuman performance in computer vision using deep CNNs (Krizhevsky, Ciresan etc.)
- 2016: Alphago beats Lee Sedol in Go

THE EXPANDING UNIVERSE OF MODERN AI



WHAT IS DEEP LEARNING?

Biologically-inspired multilayer neural networks



Both supervised and unsupervised

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

WHAT IS DEEP LEARNING?

Example. Face recognition (Facebook)

Edges | Eyes, Noses, Mouths | Faces

Deeper layers learn higher-order features

LISA lab, "Deep Learning Tutorials." (http://www.deeplearning.net/tutorial/) accessed 2014.

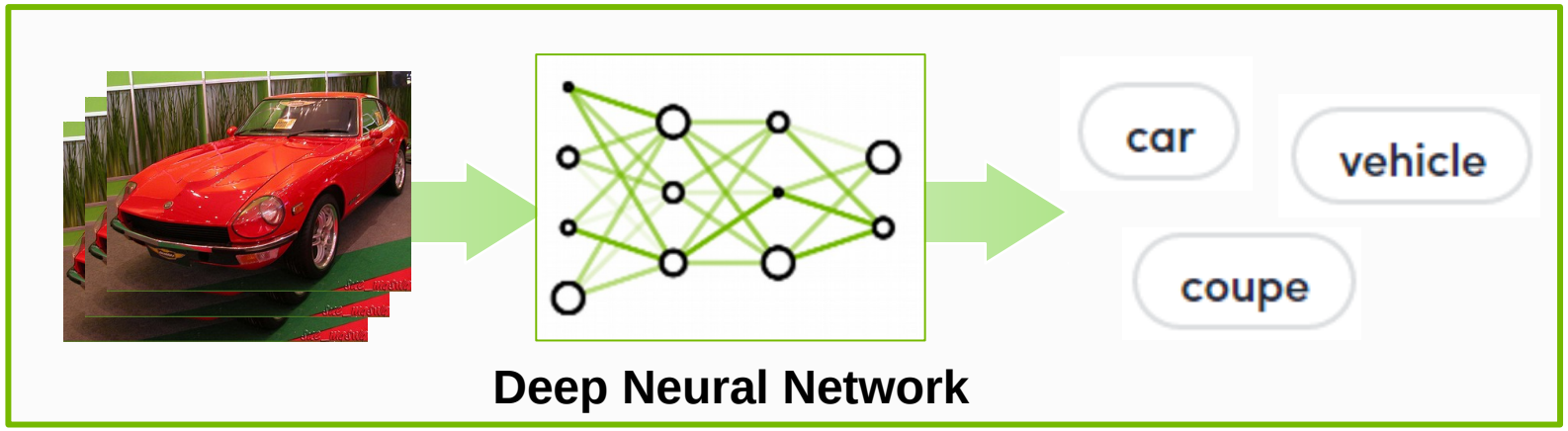
Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

A NEW COMPUTING MODEL

Algorithms that Learn from Examples

Traditional Approach

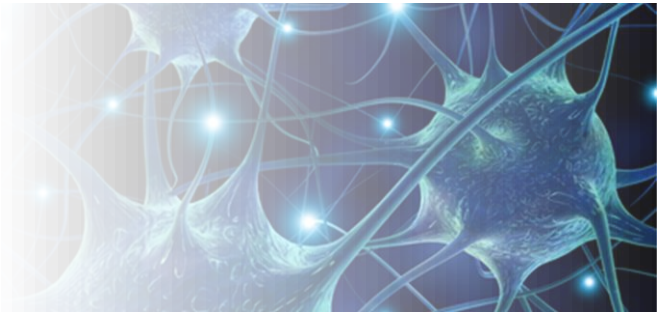
- ➢ Requires domain experts
- ➢ Time consuming
- ➢ Error prone
- ➢ Not scalable to new problems



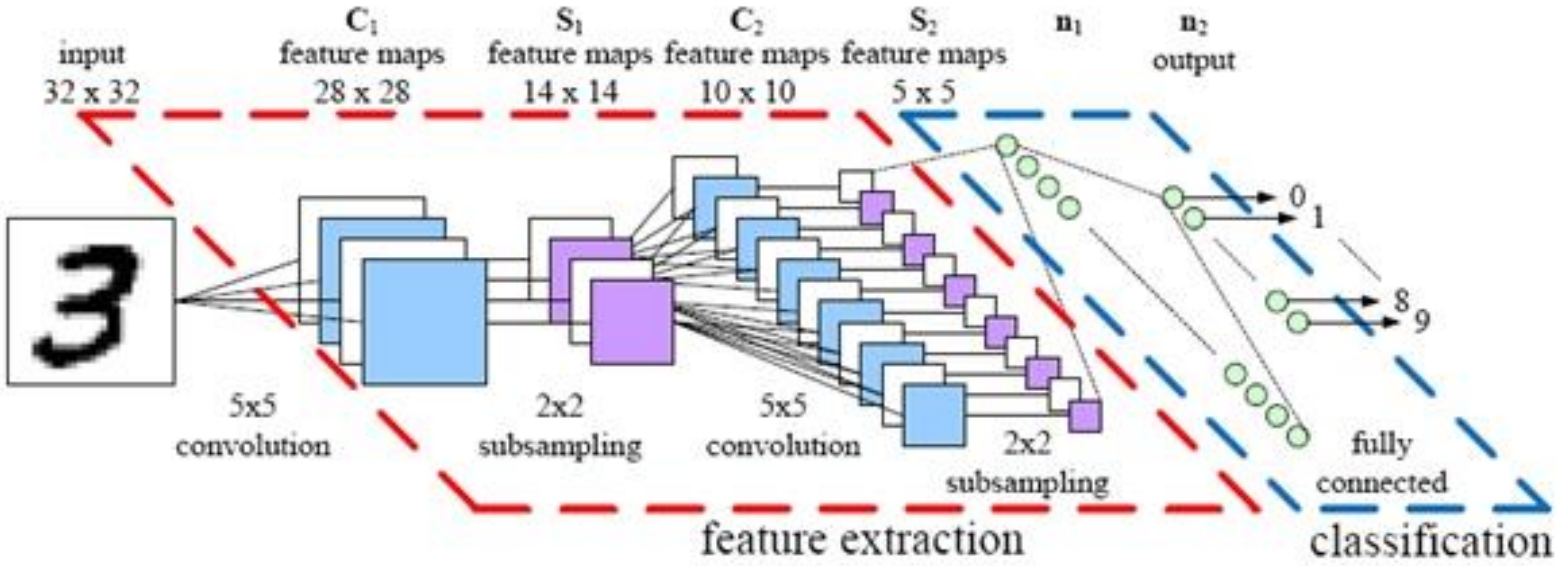
Deep Learning Approach

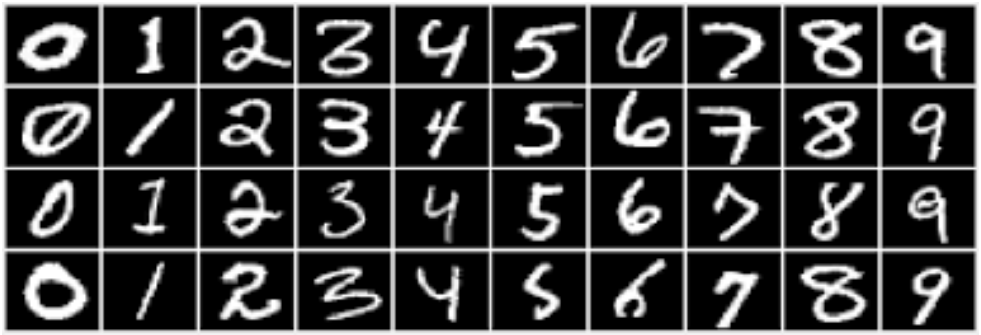
- ✓ Learn from data
- ✓ Easily to extend
- ✓ Speedup with GPUs

APPLICATIONS

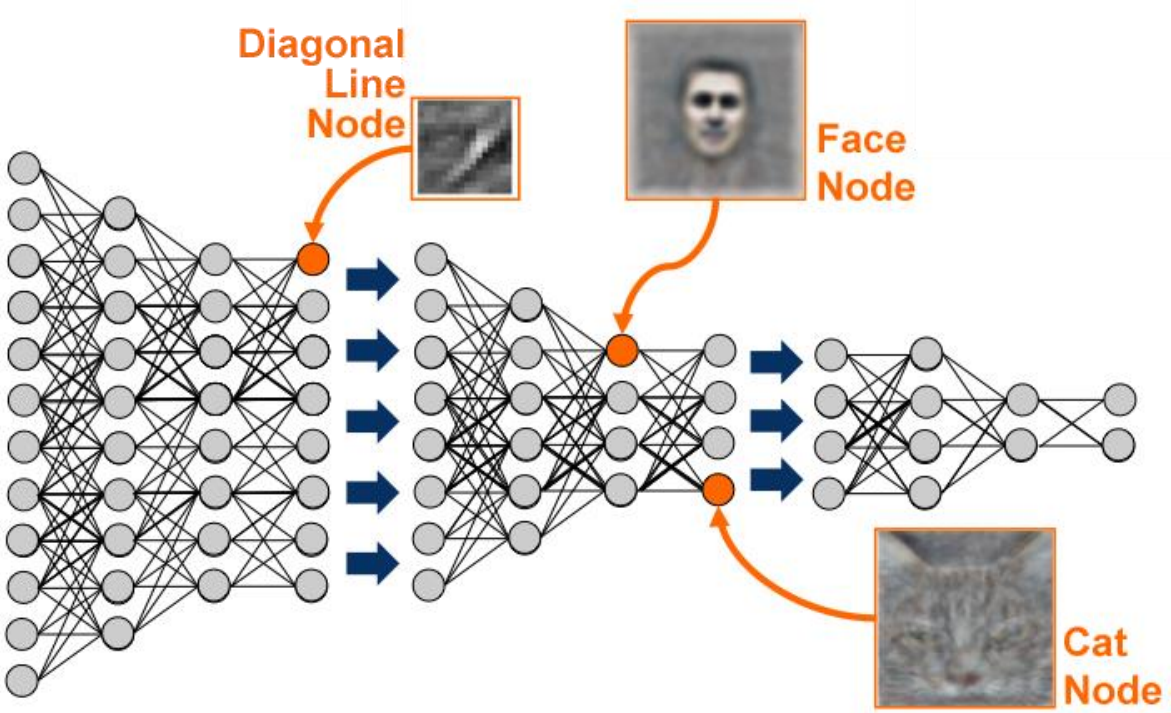


HANDWRITING RECOGNITION



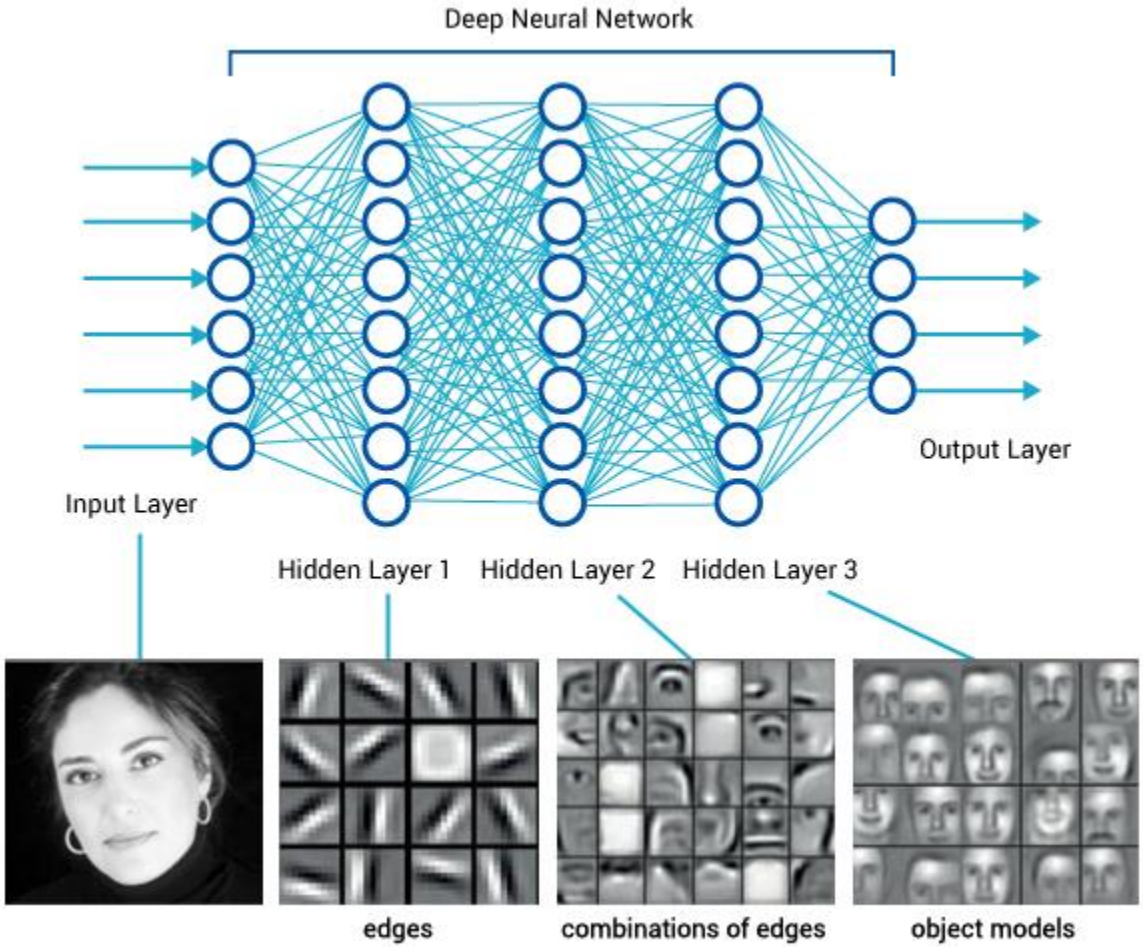


GOOGLE CAT VIDEOS

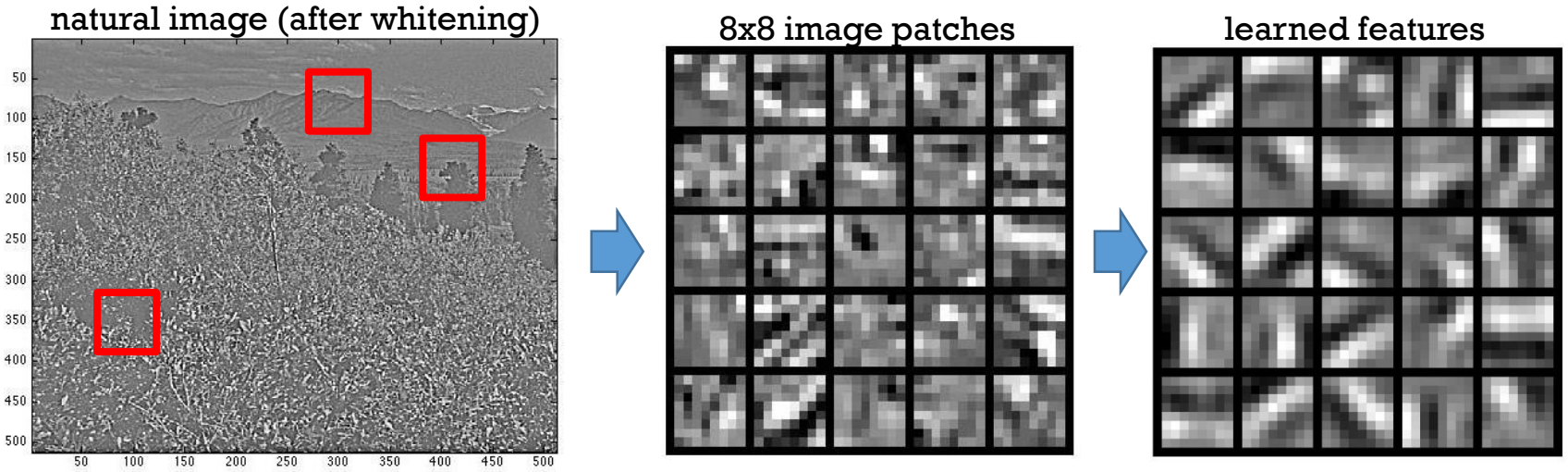


Deep Learning
with GPUs

FACE RECOGNITION



NATURAL IMAGES



Edge features similar to those from neuroscience experiments (see Hubel & Wiesel Cat Experiment, 1959).

SPEECH TRANSLATION

From Hidden Markov Models to Recurrent Neural Networks

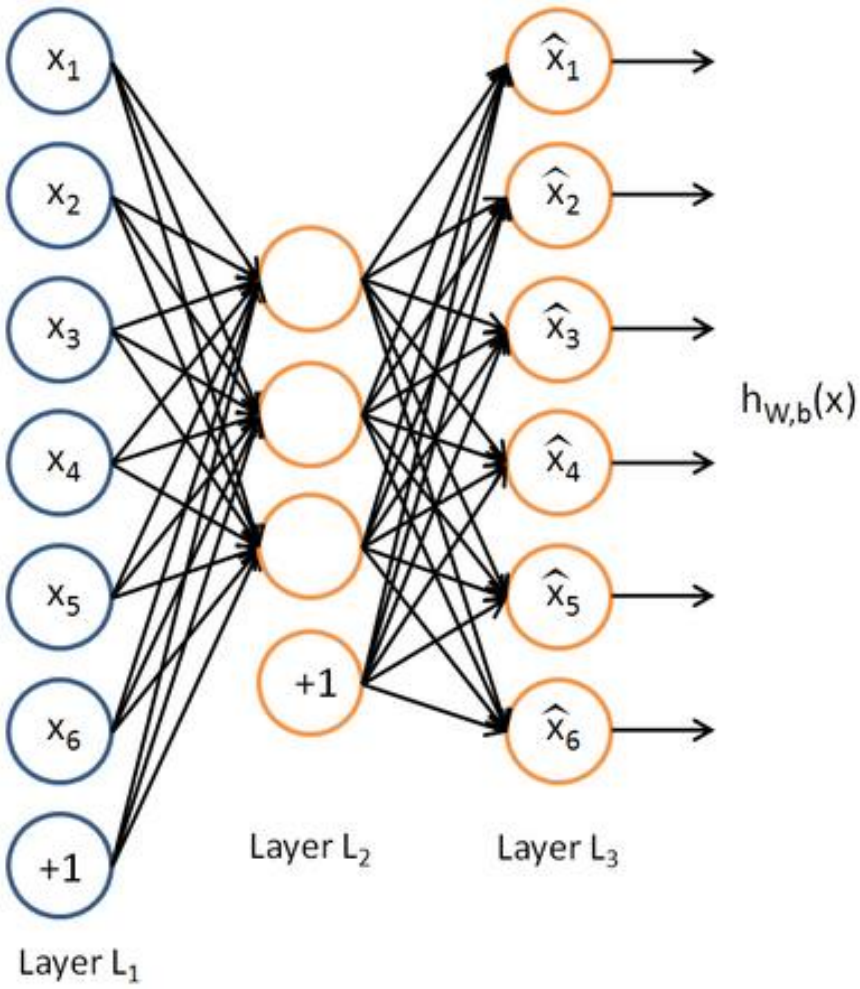
https://www.youtube.com/watch?v=mWTySUGXR2k

AUTOENCODERS

Training a multilayer neural network to reconstruct the input from a reduced representation.

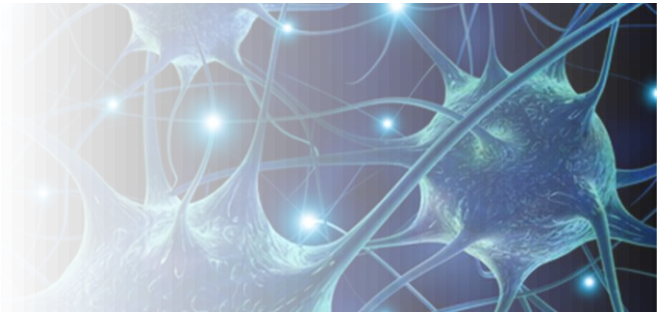
Strategies for Dimensionality Reduction

- Few hidden neurons
- Sparse activations



5 min break

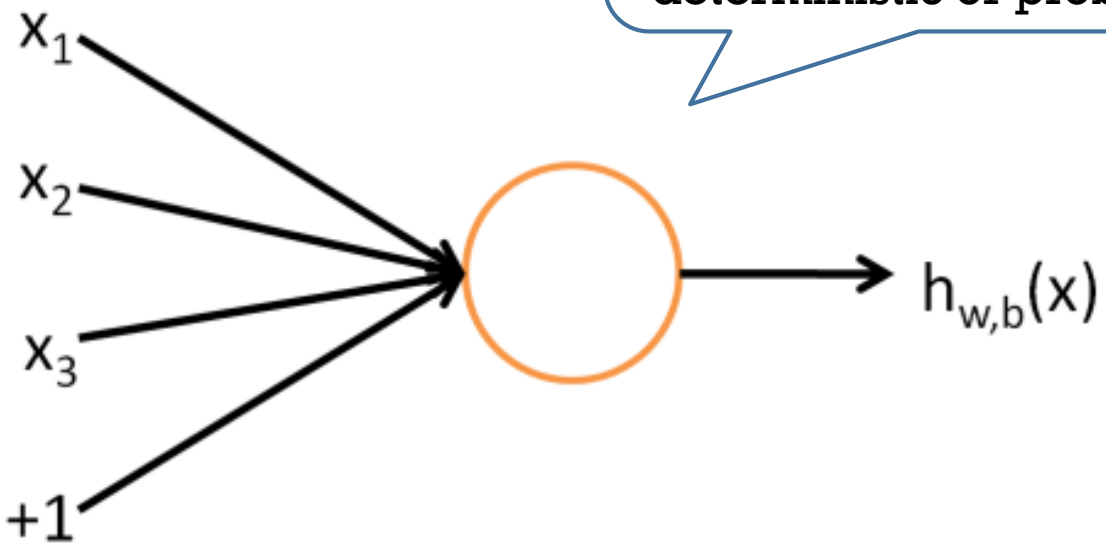
FEEDFORWARD NETWORKS



NEURON

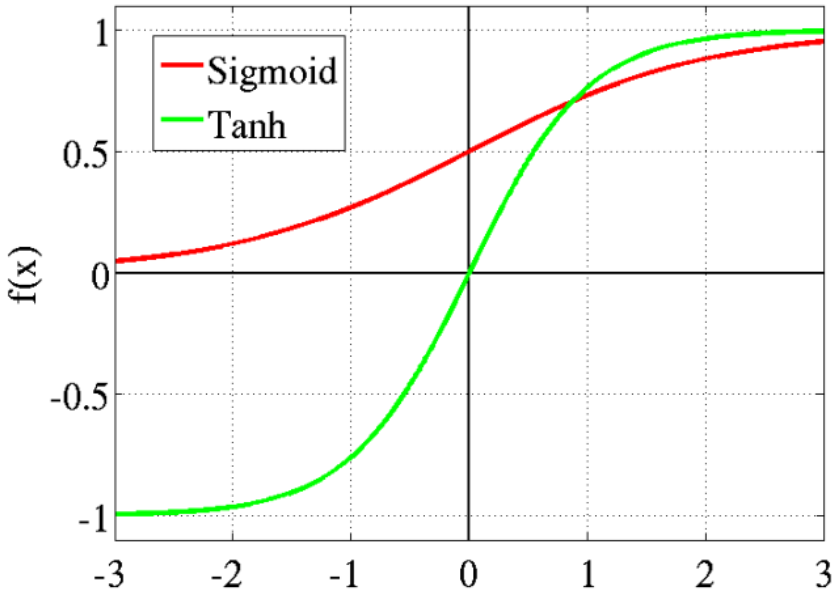
Perceptron.

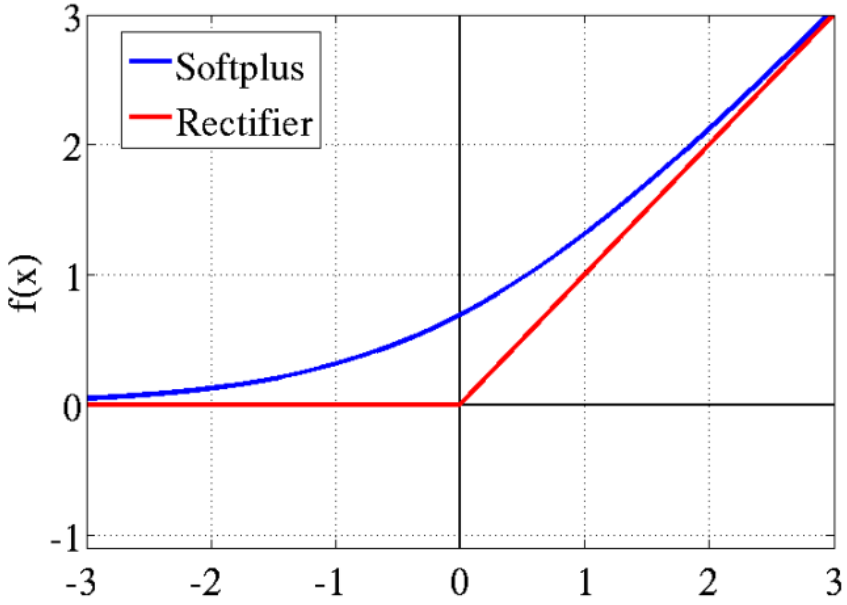
Depending on function f,
neurons can be:
real-valued or binary-valued;
deterministic or probabilistic.



$$h_{w,b}(x) = f(w^{\top}x) = f\left(\sum_{i=1}^{d} w_i x_i + b\right)$$

ACTIVATION FUNCTIONS





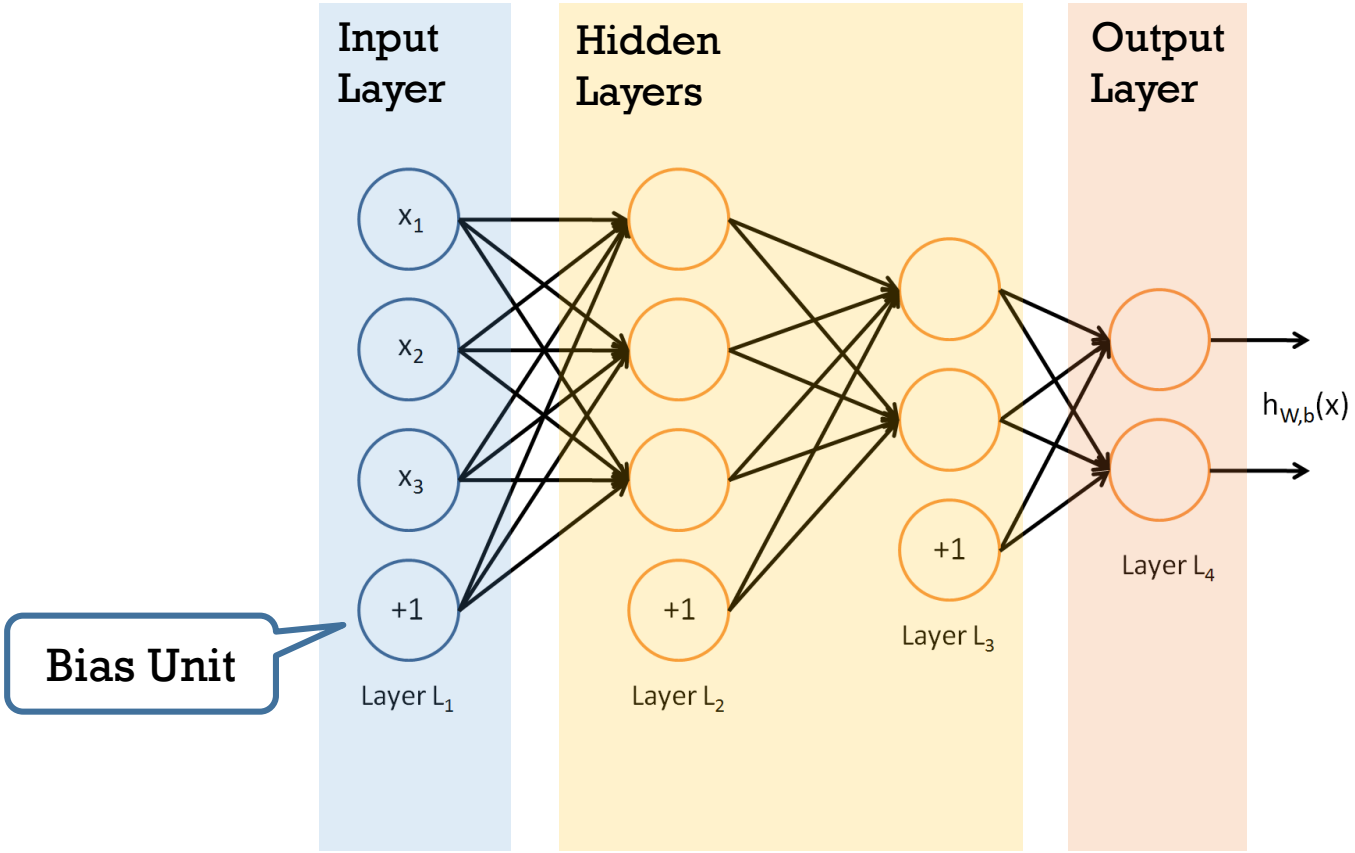
sigmoid $f(z) = \frac{1}{1+e^{-z}}$

softplus $f(z) = \ln(1 + e^{z})$

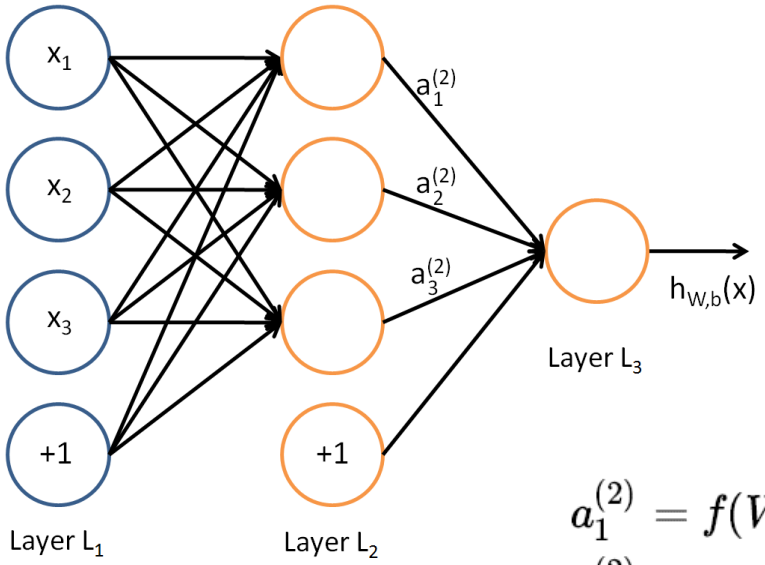
tanh $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

rectified linear unit (ReLU) $f(z) = \max(0, z)$

MULTI-LAYER NEURAL NETWORK

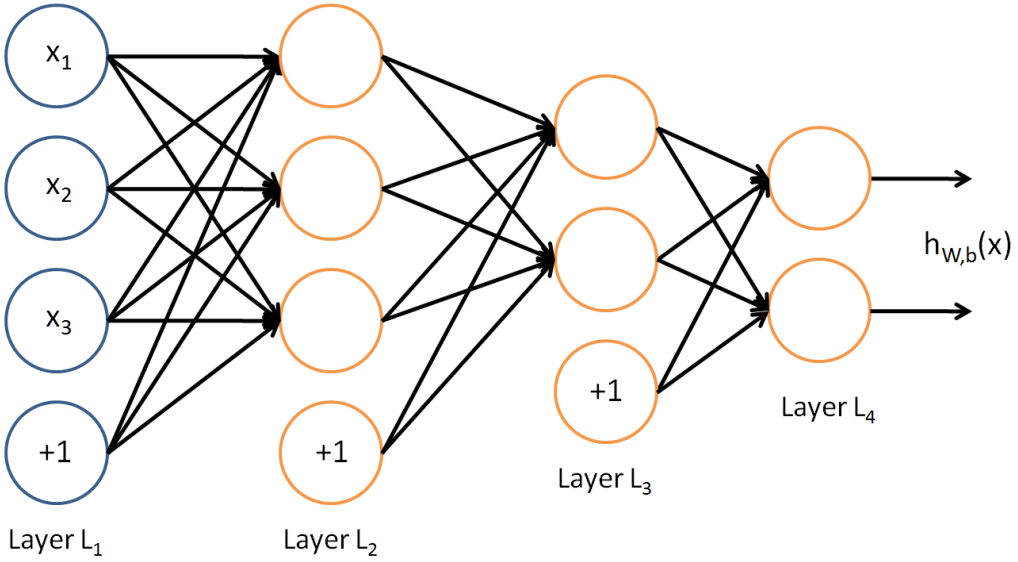


MULTI-LAYER NEURAL NETWORK



$$a_1^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})$$

MULTI-LAYER NEURAL NETWORK



Feedforward Neural Network.

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

Activation

Neural Network Architecture.
Arrangement of neurons, e.g. number of neurons in each layer.

Theorem (Universal approximation theorem)

Let $\mathcal{X}$ be a closed and bounded subset of $\mathbb{R}^m$, and f be any continuous function on $\mathcal{X}$. Assume the activation function $\phi : \mathbb{R} \to \mathbb{R}$ is non-constant, bounded and continuous. Then given any $\epsilon > 0$, there exist N, b_i, v_i and vectors $w_i \in \mathbb{R}^m$ such that when we define

$$F(x) = \sum_{i=1}^{N} v_i \phi\left(\langle w_i, x \rangle + b_i\right),$$

we have

$$|F(x) - f(x)| < \epsilon$$

for all $x \in \mathcal{X}$.