

Introduction to Deep Learning

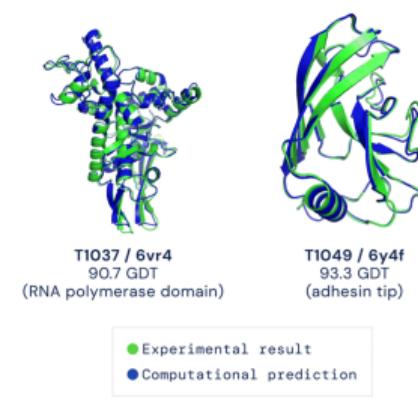
DL@MBL 2023

August 22, 2023

Deep Learning

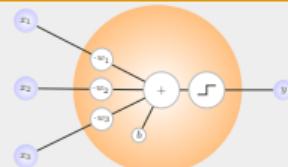


In late of "regular English". My notion ofimitating government—is a great part of
commission did not
power away from it.
Bulwer-Poole. White
on the back. The king
in a unicorn, not reach
appeared quite seriously. It
late governments are bei
and federation. You're not
two centuries, the myste
rich site, according to a n
t named the population, a
id Perez. "By the time we
is wholly inadvertent
one peak, the water looked
glasses, a black jacket;
the top of one peak, the wa
are capturing the somatic di
black hoodie with the labo
to Human civilization. Wi
I heart, which I think is w
benefit of the rich and tura
but it's kind of misleading
refined this odd phenomenon
n Department of officials. The
horn, Dvds+Unicorn. The
peak, the water looked
ment of any other." This is
part, which I think is what
for example, that they ha
ion was made only fo
bag. She was carrying
she was carrying
from Abercrombie and Fitch
ily. **OPENAI GPT-3**



Outline

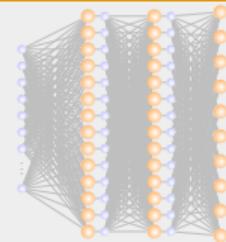
1. Perceptrons



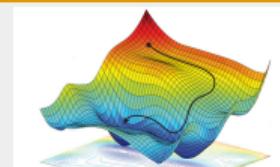
2. Networks



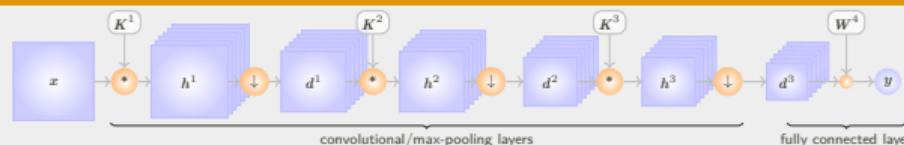
3. Deep Networks



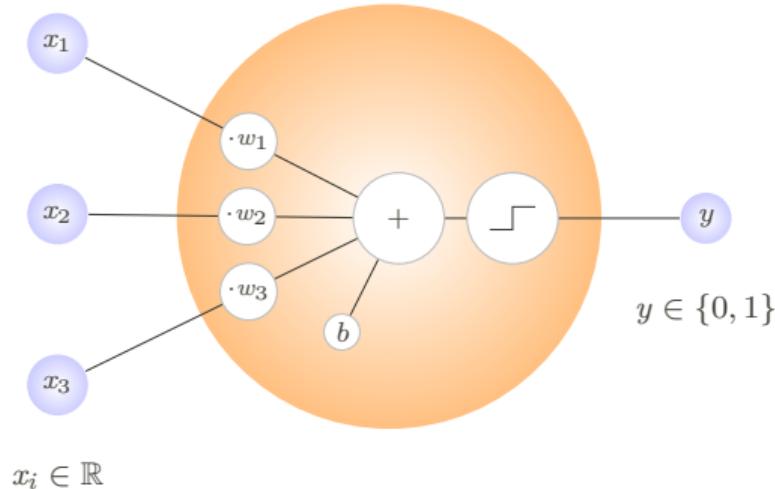
4. Learning



5. Convolutional Networks



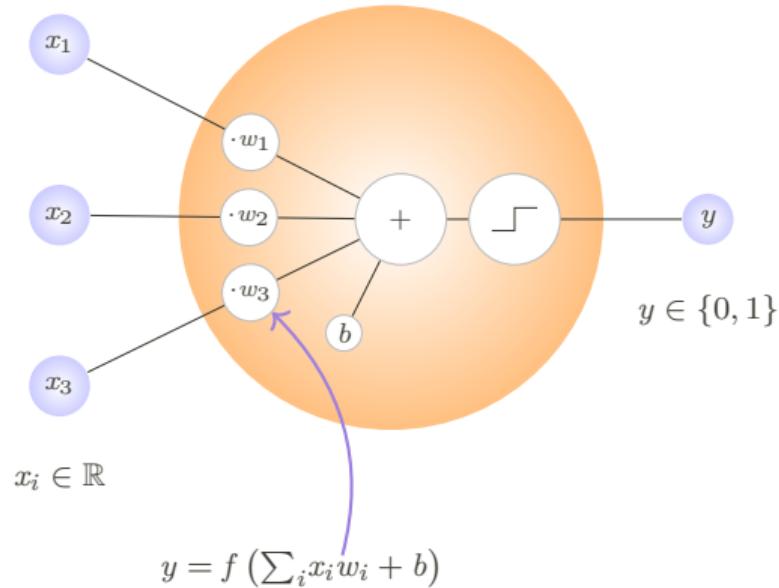
The Building Block of Neural Networks



Perceptron

- invented by McCulloch and Pitts in 1943
- termed "perceptron" by Rosenblatt in 1958
- also known as: Threshold Logic Unit, Linear Threshold Unit, McCulloch-Pitts model

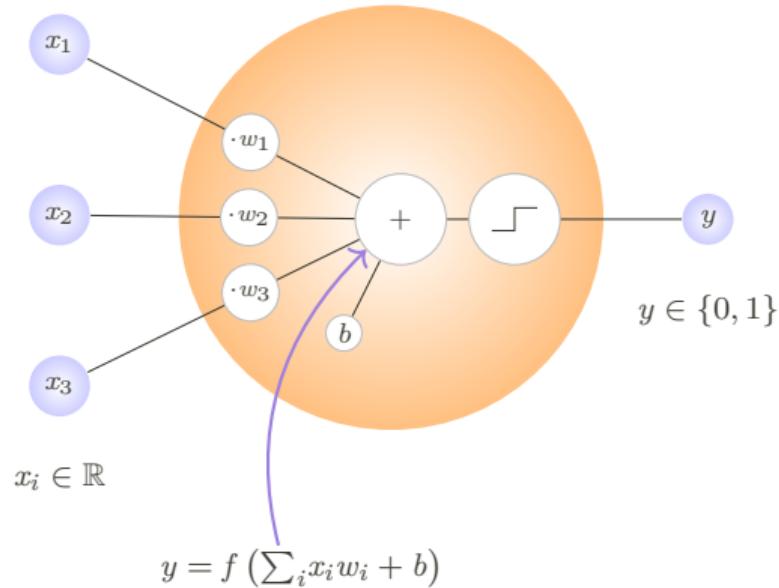
The Building Block of Neural Networks



Perceptron

- invented by McCulloch and Pitts in 1943
- termed "perceptron" by Rosenblatt in 1958
- also known as: Threshold Logic Unit, Linear Threshold Unit, McCulloch-Pitts model

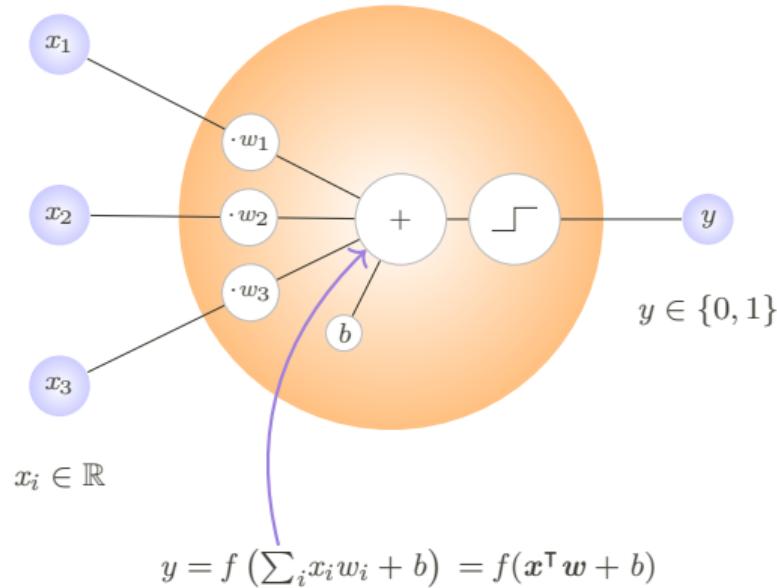
The Building Block of Neural Networks



Perceptron

- invented by McCulloch and Pitts in 1943
- termed "perceptron" by Rosenblatt in 1958
- also known as: Threshold Logic Unit, Linear Threshold Unit, McCulloch-Pitts model

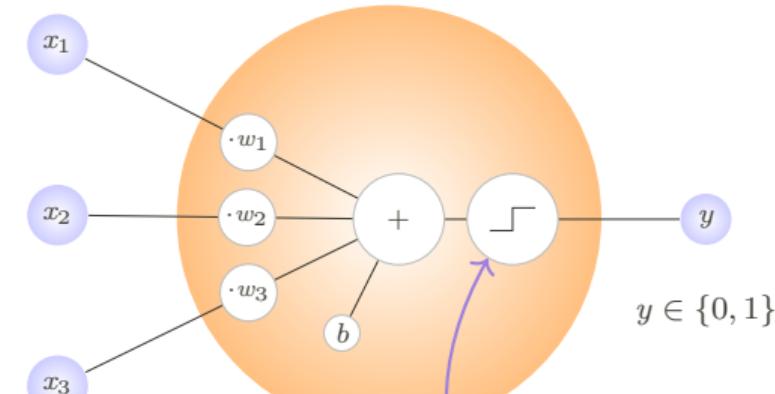
The Building Block of Neural Networks



Perceptron

- invented by McCulloch and Pitts in 1943
- termed "perceptron" by Rosenblatt in 1958
- also known as: Threshold Logic Unit, Linear Threshold Unit, McCulloch-Pitts model

The Building Block of Neural Networks



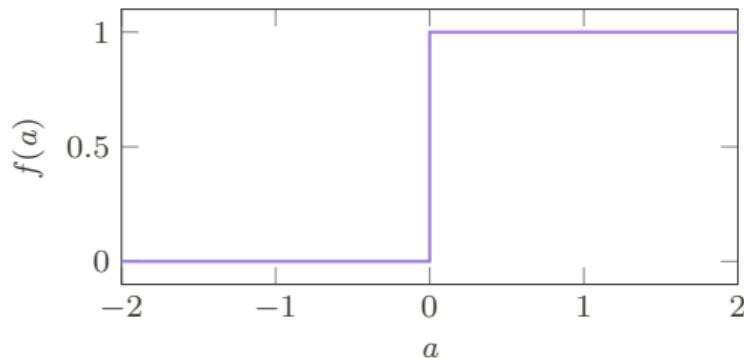
$$x_i \in \mathbb{R}$$

$$y = f \left(\sum_i x_i w_i + b \right) = f(x^\top w + b)$$

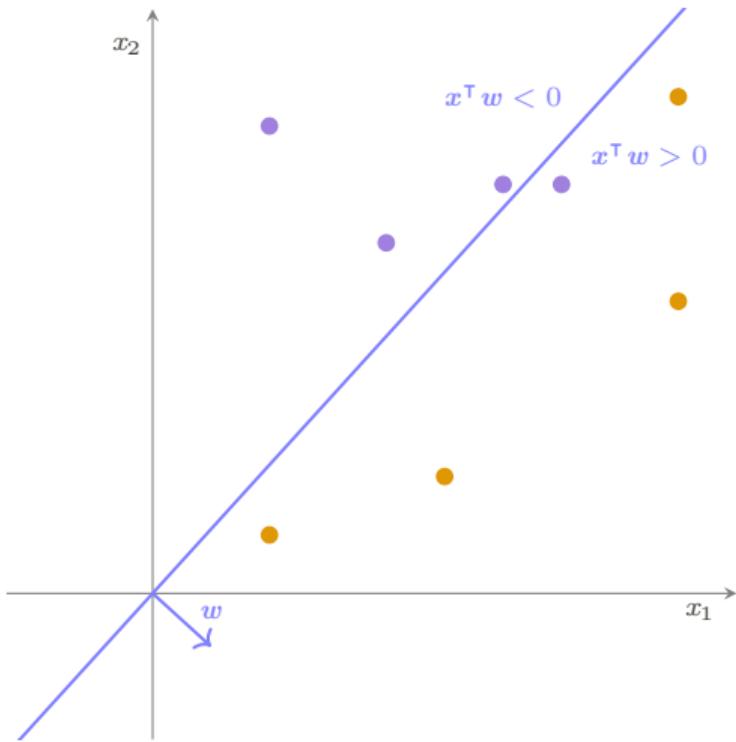
$$f(a) = \begin{cases} 0 & \text{if } a \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

Perceptron

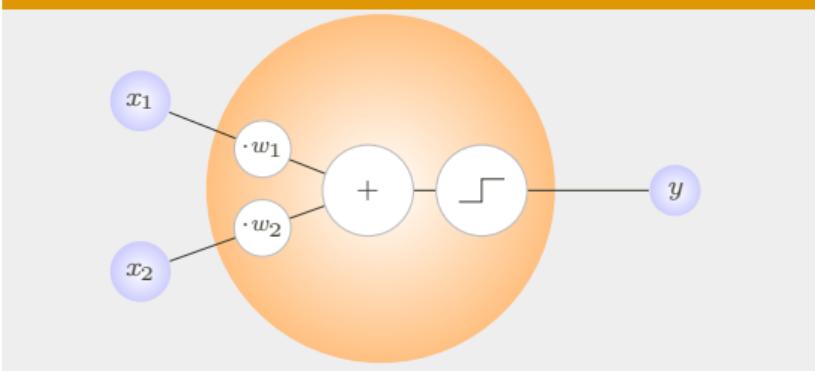
- invented by McCulloch and Pitts in 1943
- termed "perceptron" by Rosenblatt in 1958
- also known as: Threshold Logic Unit, Linear Threshold Unit, McCulloch-Pitts model



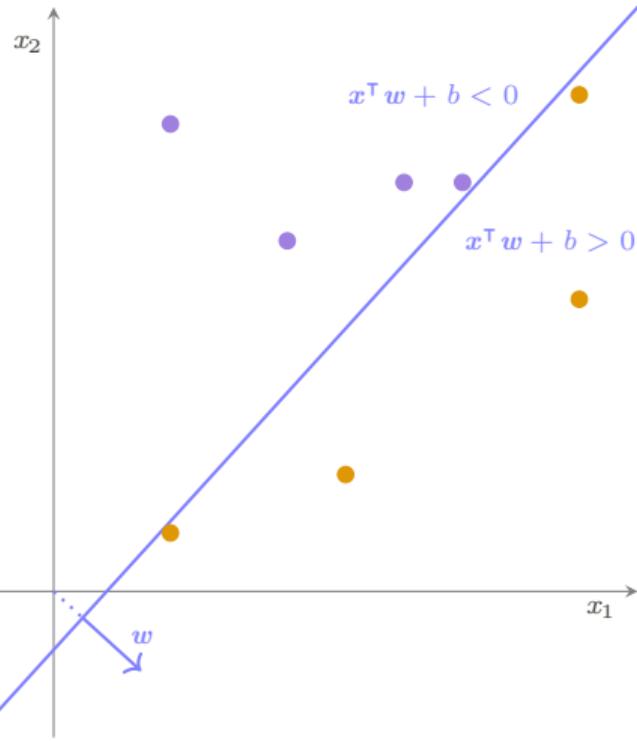
Just a Linear Classifier



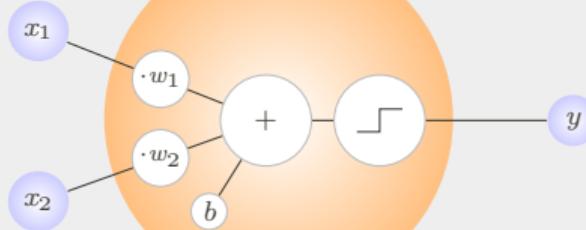
Example with 2 inputs



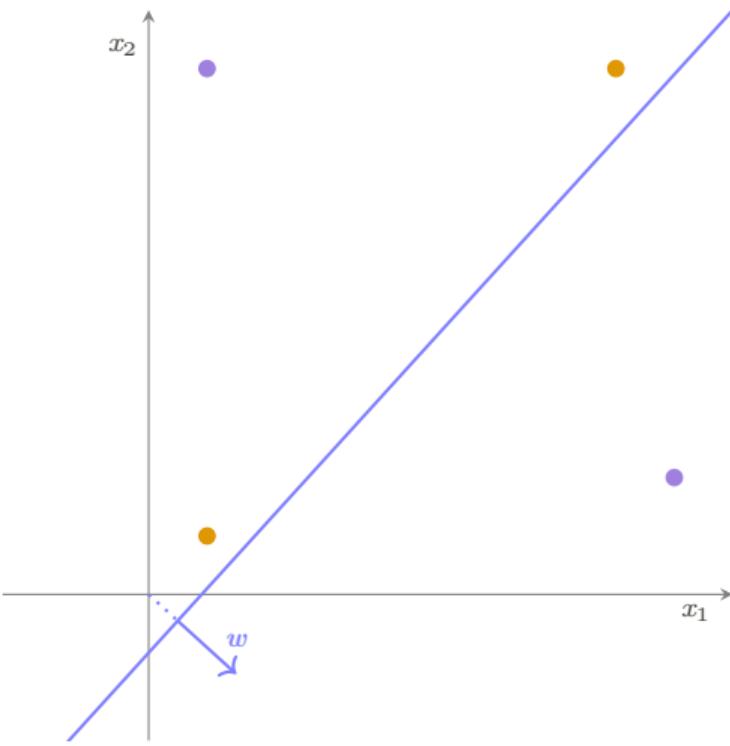
Just a Linear Classifier



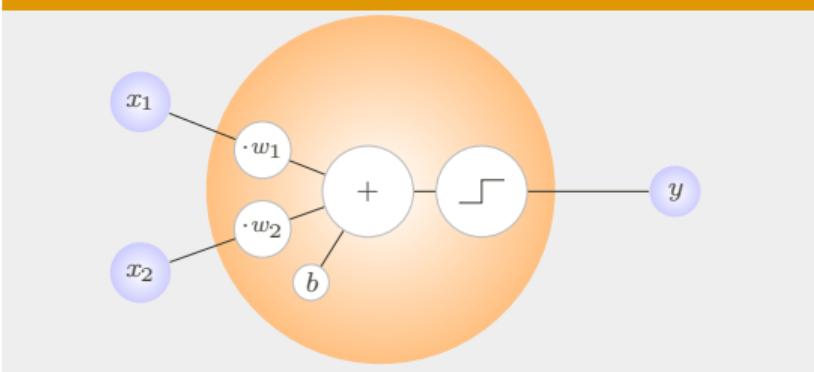
Example with 2 inputs



Just a Linear Classifier



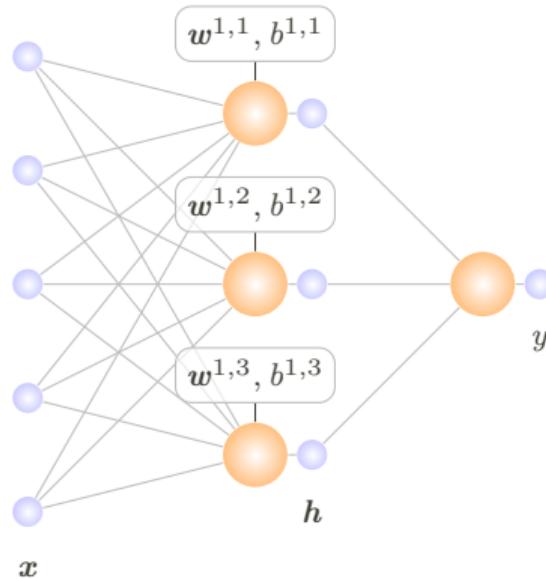
Example with 2 inputs



The XOR Affair and AI Winter

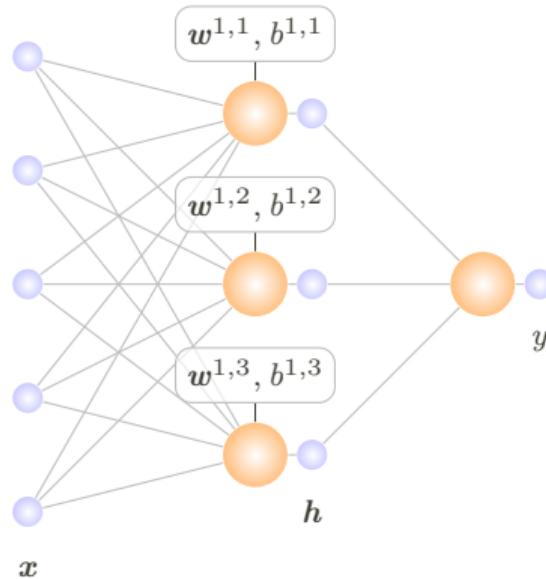
- “The perceptron may eventually be able to learn, make decisions, and translate languages.”
—Rosenblatt, 1958
- “Some functions are tricky.” —Minsky/Papert, 1969
- interest in neural networks declined until the 80s

[1]



Output Function with one Hidden Layer

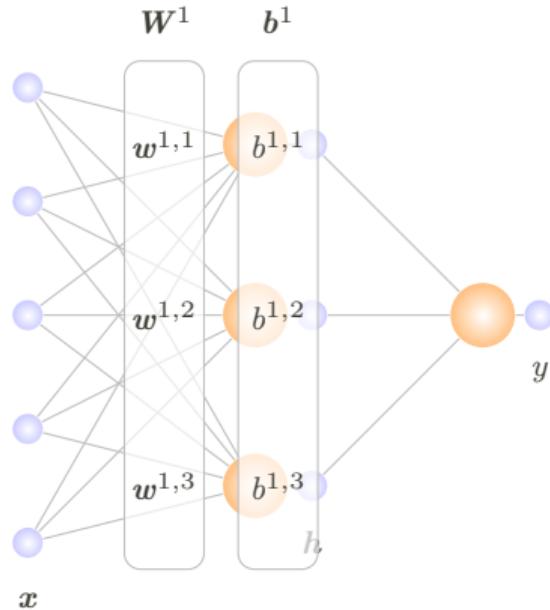
$$h = (f(w^{1,1}x + b^{1,1}), f(w^{1,2}x + b^{1,2}), f(w^{1,3}x + b^{1,3}))$$



Output Function with one Hidden Layer

$$\begin{aligned} h &= (f(w^{1,1}x + b^{1,1}), f(w^{1,2}x + b^{1,2}), f(w^{1,3}x + b^{1,3})) \\ &= f((w^{1,1}x + b^{1,1}, w^{1,2}x + b^{1,2}, w^{1,3}x + b^{1,3})) \end{aligned}$$

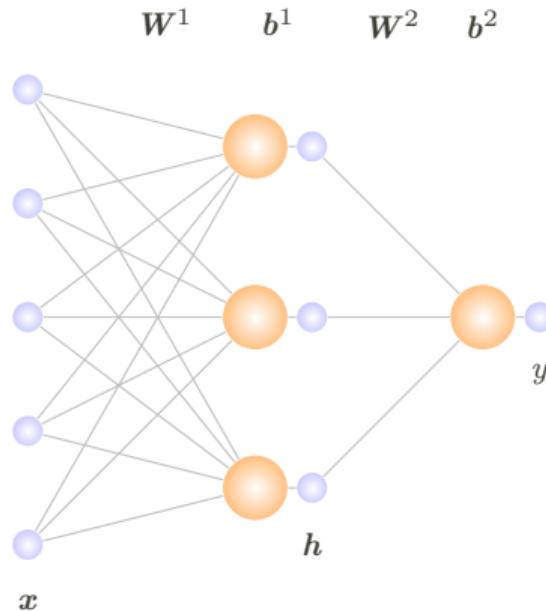
Neural Networks



Output Function with one Hidden Layer

$$\begin{aligned} h &= (f(w^{1,1}x + b^{1,1}), f(w^{1,2}x + b^{1,2}), f(w^{1,3}x + b^{1,3})) \\ &= f((w^{1,1}x + b^{1,1}, w^{1,2}x + b^{1,2}, w^{1,3}x + b^{1,3})) \\ &= f(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) \end{aligned}$$

Neural Networks

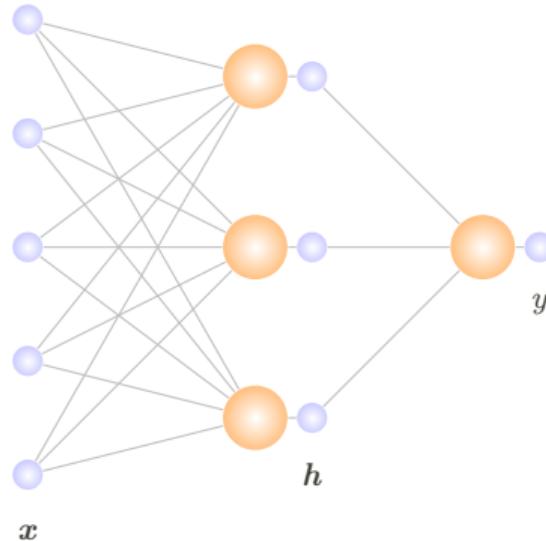


Output Function with one Hidden Layer

$$\begin{aligned} h &= (f(w^{1,1}x + b^{1,1}), f(w^{1,2}x + b^{1,2}), f(w^{1,3}x + b^{1,3})) \\ &= f((w^{1,1}x + b^{1,1}, w^{1,2}x + b^{1,2}, w^{1,3}x + b^{1,3})) \\ &= f(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1) \end{aligned}$$

$$y = f(\mathbf{W}^2 h + \mathbf{b}^2) = f(\mathbf{W}^2 f(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2)$$

$$W^1 \quad b^1 \quad W^2 \quad b^2$$



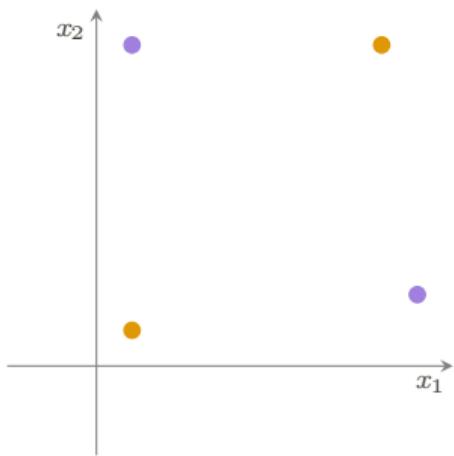
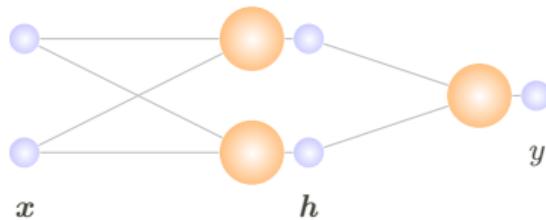
Output Function with one Hidden Layer

$$\begin{aligned} h &= (f(w^{1,1}x + b^{1,1}), f(w^{1,2}x + b^{1,2}), f(w^{1,3}x + b^{1,3})) \\ &= f((w^{1,1}x + b^{1,1}, w^{1,2}x + b^{1,2}, w^{1,3}x + b^{1,3})) \\ &= f(W^1x + b^1) \end{aligned}$$

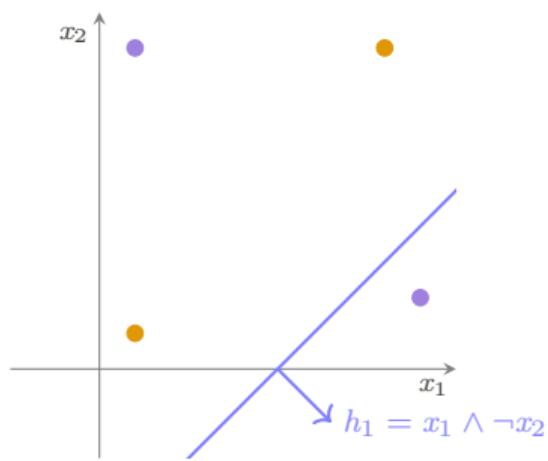
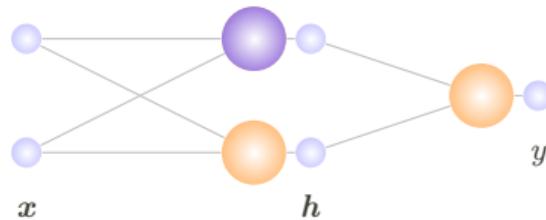
$$y = f(W^2h + b^2) = f(W^2f(W^1x + b^1) + b^2)$$

- just matrix multiplications and element-wise non-linearities

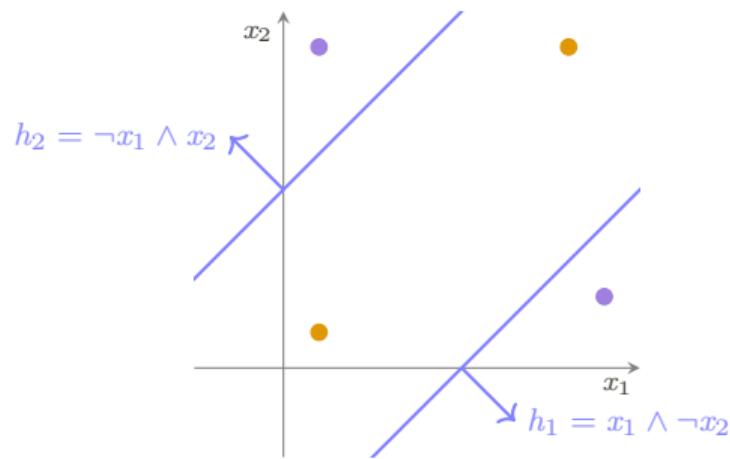
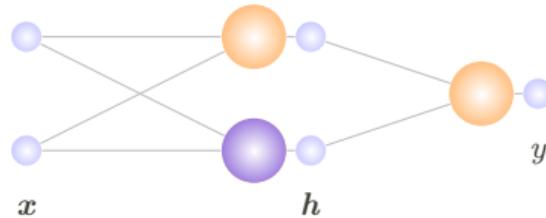
XOR Revisited



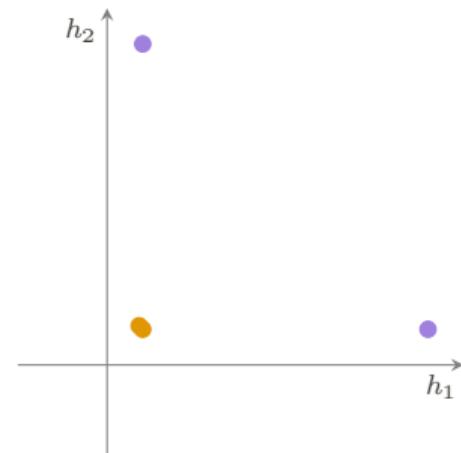
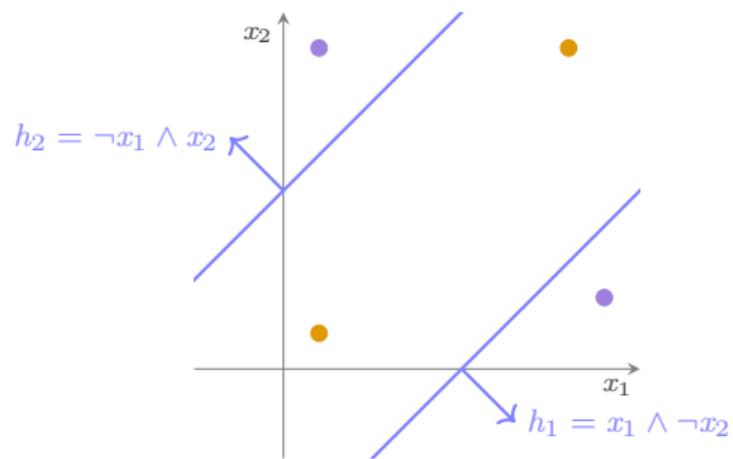
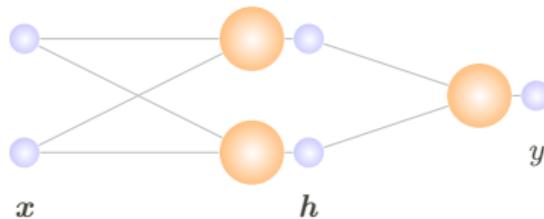
XOR Revisited



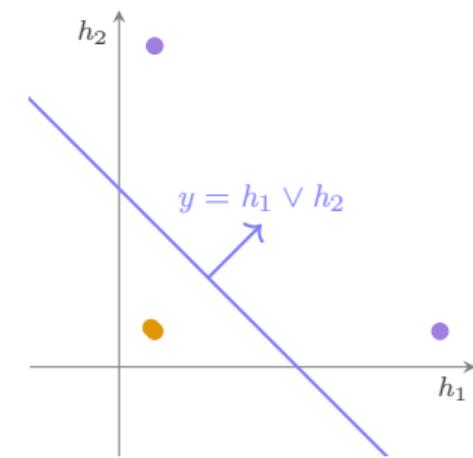
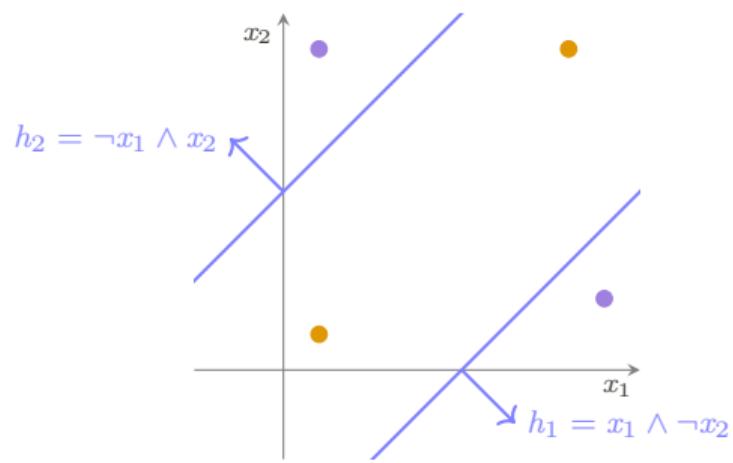
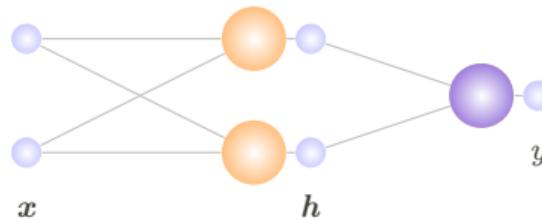
XOR Revisited



XOR Revisited



XOR Revisited



How many layers do we need?

Theoretically...

...one hidden layer is sufficient to model any function:

- every boolean function can be transformed into disjunctive normal form
- every continuous function can be approximated with one hidden layer

[2]

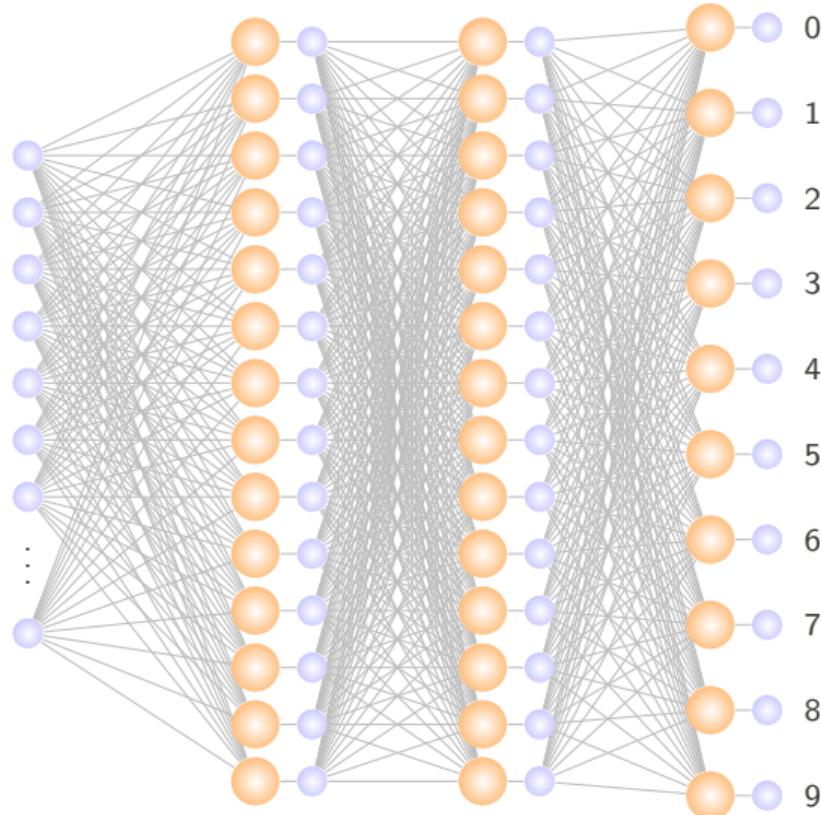
[3]

Practically...

...more than one layer will be more efficient:

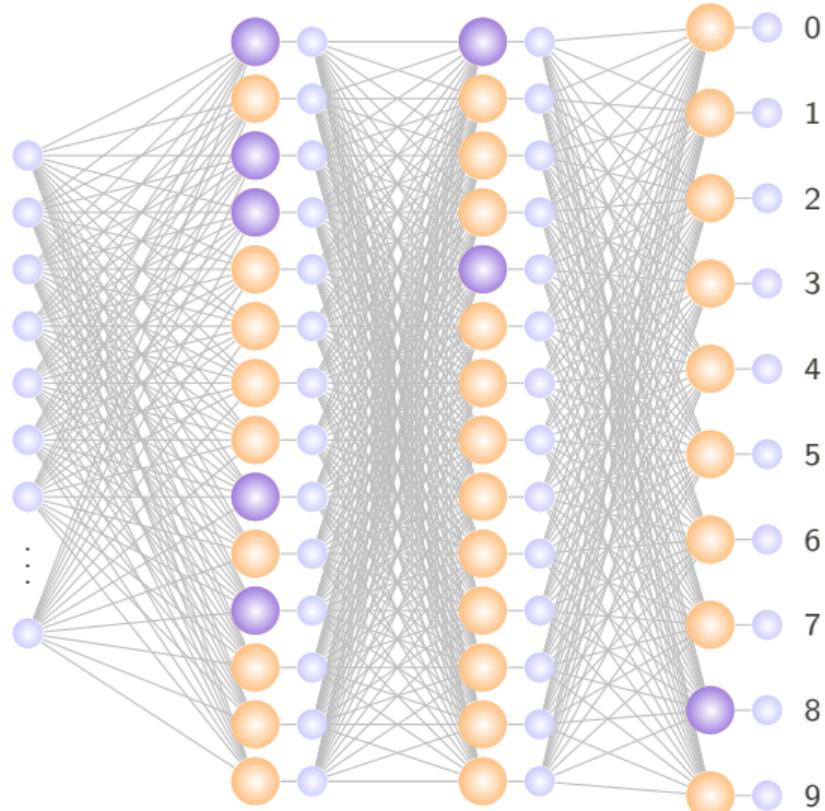
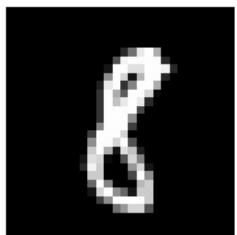


Example: Digit Recognition



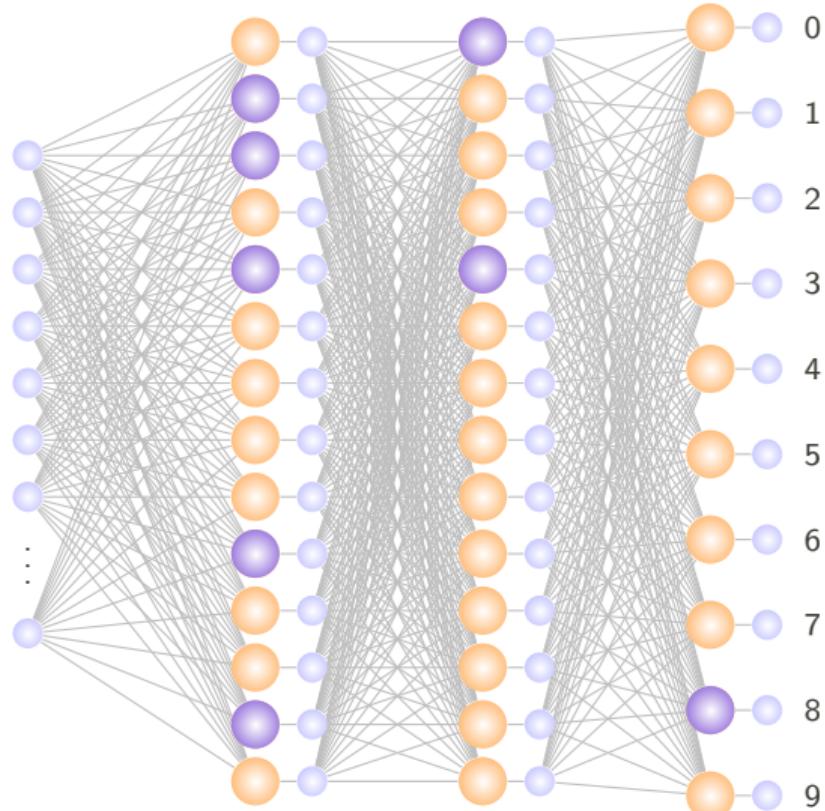
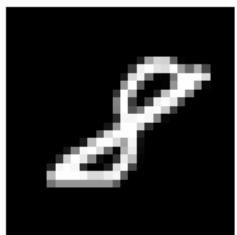
[4]

Example: Digit Recognition



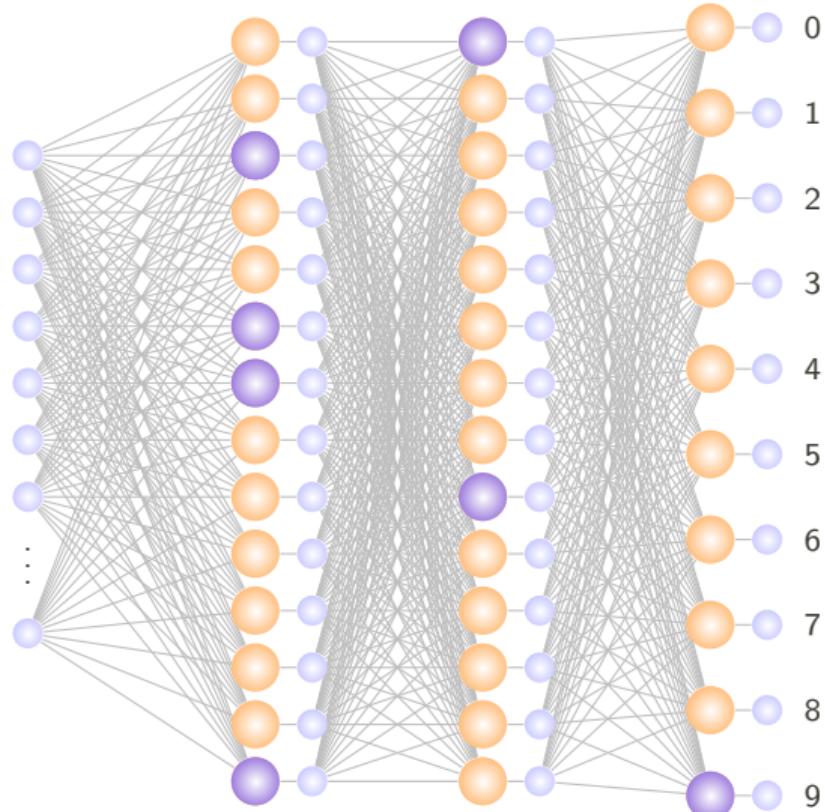
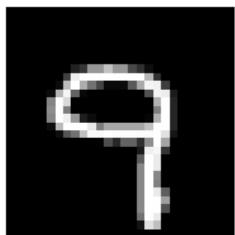
[4]

Example: Digit Recognition



[4]

Example: Digit Recognition



[4]

Supervised Learning

Model

- function $M_{\theta} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\hat{y} = M_{\theta}(x)$
- parameterized by θ

Data

- $T = \{x^i, y^i\}_{i=1,\dots,k}$
- input x
- desired output y

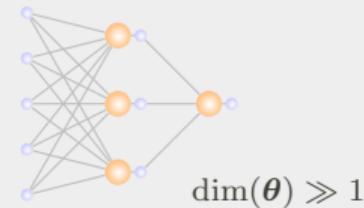
Objective

- $l : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$, e.g., $l(\hat{y}, y) = \frac{1}{2}|\hat{y} - y|^2$
- $L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}^i, y^i)$

Supervised Learning

Model

- function $M_{\theta} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\hat{y} = M_{\theta}(x)$
- parameterized by θ



Data

- $T = \{\mathbf{x}^i, \mathbf{y}^i\}_{i=1,\dots,k}$
- input x
- desired output y

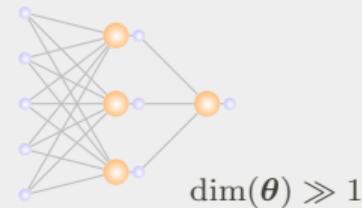
Objective

- $l : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$, e.g., $l(\hat{y}, y) = \frac{1}{2}|\hat{y} - y|^2$
- $L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}^i, y^i)$

Supervised Learning

Model

- function $M_{\theta} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\hat{y} = M_{\theta}(x)$
- parameterized by θ



Data

- $T = \{x^i, y^i\}_{i=1,\dots,k}$
- input x
- desired output y



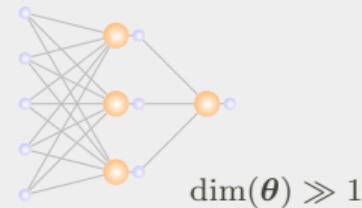
Objective

- $l : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$, e.g., $l(\hat{y}, y) = \frac{1}{2}|\hat{y} - y|^2$
- $L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}^i, y^i)$

Supervised Learning

Model

- function $M_{\theta} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\hat{y} = M_{\theta}(x)$
- parameterized by θ



Data

- $T = \{x^i, y^i\}_{i=1,\dots,k}$
- input x
- desired output y



Objective

- $l : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$, e.g., $l(\hat{y}, y) = \frac{1}{2}|\hat{y} - y|^2$
- $L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}^i, y^i)$

smooth, differentiable

Optimization Problem

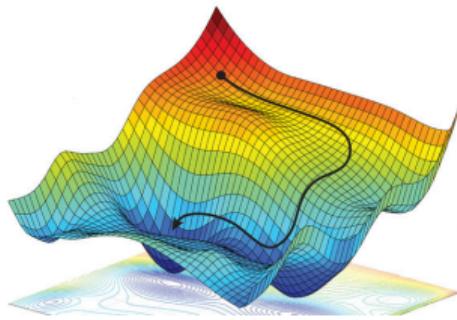
Learning objective:

$$\theta^* = \arg \min_{\theta} L(\theta)$$

- in general no closed-form solution
- probably not even wanted

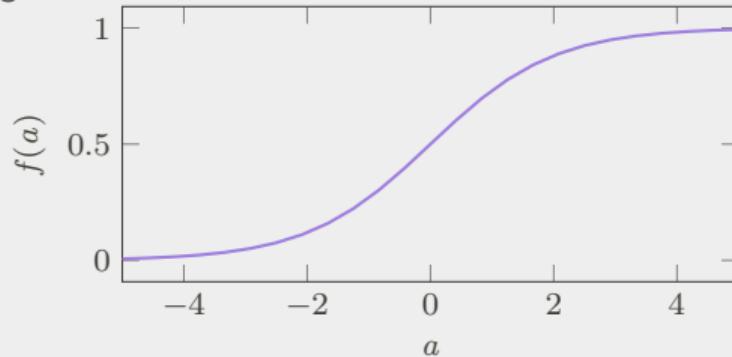
$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta)$$

- fallback to gradient descent

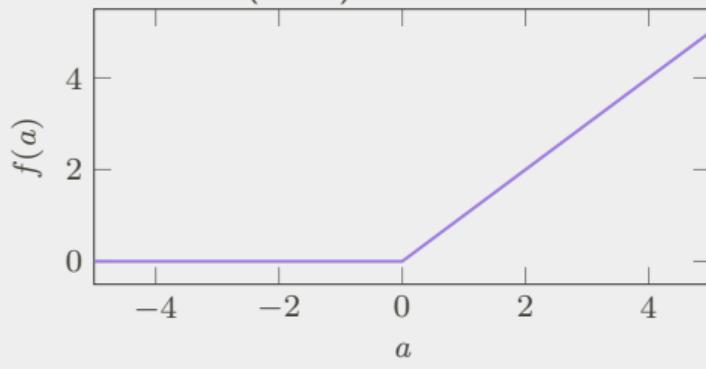


Differentiable Non-linearities

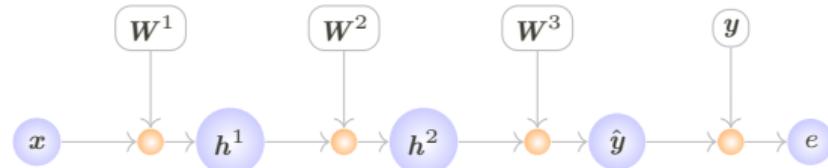
Sigmoid activation function:



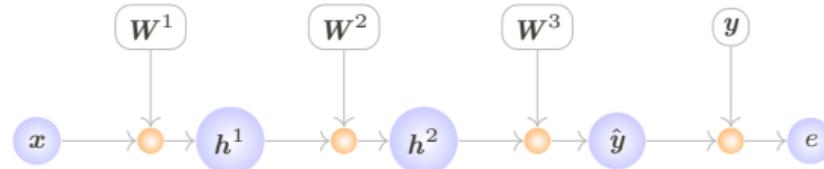
Rectified linear unit (ReLU) activation function:



Error Back Propagation



Error Back Propagation

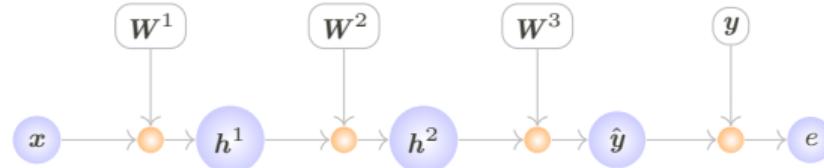


$$\begin{aligned}\hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\ &= f(W^3 h^2) \\ &= f(W^3 f(W^2 h^1)) \\ &= f(W^3 f(W^2 f(W^1 x)))\end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Error Back Propagation



Chain Rule

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

$$\hat{y} = M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3)$$

$$= f(W^3 h^2)$$

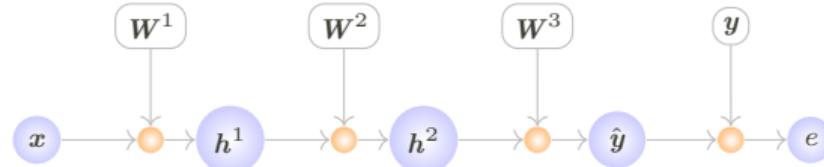
$$= f(W^3 f(W^2 h^1))$$

$$= f(W^3 f(W^2 f(W^1 x)))$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Error Back Propagation



$$\begin{aligned}\hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\ &= f(W^3 h^2) \\ &= f(W^3 f(W^2 h^1)) \\ &= f(W^3 f(W^2 f(W^1 x)))\end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

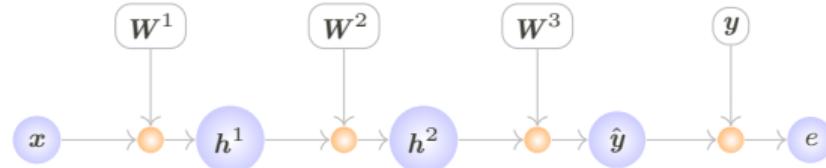
How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Chain Rule

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

$$\frac{de}{dW^3} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dW^3} = \frac{de}{d\hat{y}} f'(W^3 h^2) h^2$$

Error Back Propagation



$$\begin{aligned}\hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\ &= f(W^3 h^2) \\ &= f(W^3 f(W^2 h^1)) \\ &= f(W^3 f(W^2 f(W^1 x)))\end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

Chain Rule

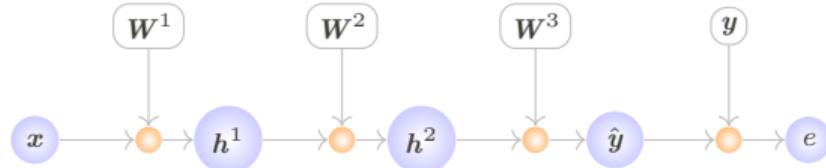
$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

$$\frac{de}{dW^3} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dW^3} = \frac{de}{d\hat{y}} f'(W^3 h^2) h^2$$

$$\frac{de}{dW^2} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dW^2}$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Error Back Propagation



$$\begin{aligned}\hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\ &= f(W^3 h^2) \\ &= f(W^3 f(W^2 h^1)) \\ &= f(W^3 f(W^2 f(W^1 x)))\end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

Chain Rule

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

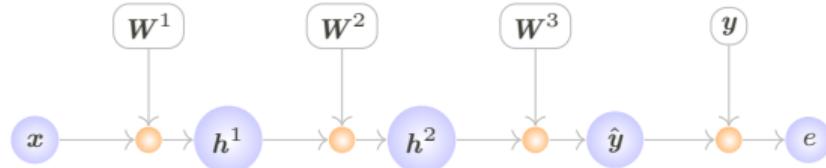
$$\frac{de}{dW^3} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dW^3} = \frac{de}{d\hat{y}} f'(W^3 h^2) h^2$$

$$\frac{de}{dW^2} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dW^2}$$

$$\frac{de}{dW^1} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dh^1} \frac{dh^1}{dW^1}$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Error Back Propagation



$$\begin{aligned}\hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\ &= f(W^3 h^2) \\ &= f(W^3 f(W^2 h^1)) \\ &= f(W^3 f(W^2 f(W^1 x)))\end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Chain Rule

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

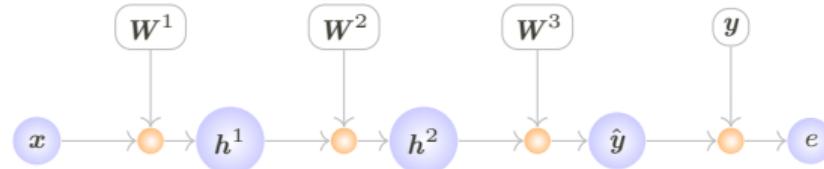
$$\frac{de}{dW^3} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dW^3} = \frac{de}{d\hat{y}} f'(W^3 h^2) h^2$$

$$\frac{de}{dW^2} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dW^2}$$

$$\frac{de}{dW^1} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dh^1} \frac{dh^1}{dW^1}$$

$$\frac{de}{dW^i} = \frac{de}{dh^i} f'(W^i h^{i-1}) h^{i-1}$$

Error Back Propagation



$$\begin{aligned}
 \hat{y} &= M_{\theta}(x) \text{ (with } \theta = W^1, W^2, W^3) \\
 &= f(W^3 h^2) \\
 &= f(W^3 f(W^2 h^1)) \\
 &= f(W^3 f(W^2 f(W^1 x)))
 \end{aligned}$$

$$e = L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

How to compute $\nabla_{\theta} L(\theta) = \frac{de}{d\theta}$?

Chain Rule

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \frac{dg(x)}{dx}$$

$$\frac{de}{dW^3} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dW^3} = \frac{de}{d\hat{y}} f'(W^3 h^2) h^2$$

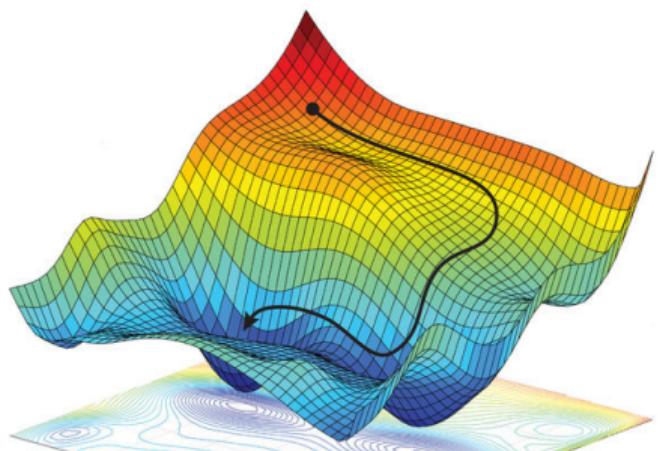
$$\frac{de}{dW^2} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dW^2}$$

$$\frac{de}{dW^1} = \frac{de}{d\hat{y}} \frac{d\hat{y}}{dh^2} \frac{dh^2}{dh^1} \frac{dh^1}{dW^1}$$

$$\frac{de}{dW^i} = \frac{de}{dh^i} f'(W^i h^{i-1}) h^{i-1}$$

$$\frac{de}{dh^i} = \frac{de}{dh^{i+1}} \frac{dh^{i+1}}{dh^i} = \frac{de}{dh^{i+1}} (f'(W^{i+1} h^i) W^{i+1})$$

Sampling Strategies



$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta)$$

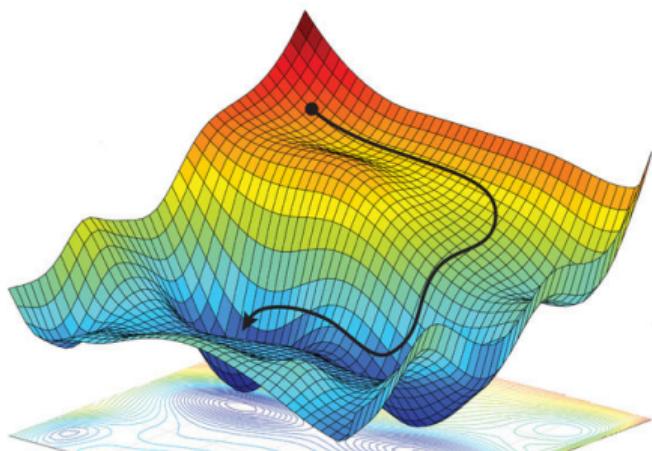
Gradient Descent

$$L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

- impractical for large k

Sampling Strategies

$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta)$$



Gradient Descent

$$L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

- impractical for large k

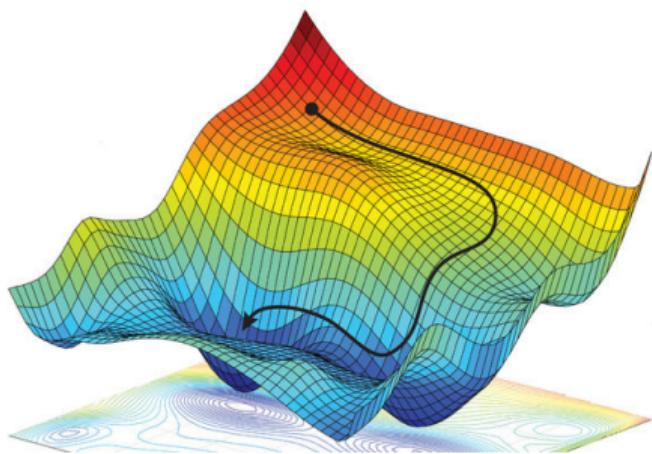
Stochastic Gradient Descent

$$L(\theta) = l(\hat{y}^i, y^i) \text{ with } i \text{ random in } [1, \dots, k]$$

- randomly sampled training example

Sampling Strategies

$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta)$$



Gradient Descent

$$L(\theta) = \frac{1}{k} \sum_{i=1}^k l(\hat{y}, y)$$

- impractical for large k

Stochastic Gradient Descent

$$L(\theta) = l(\hat{y}^i, y^i) \text{ with } i \text{ random in } [1, \dots, k]$$

- randomly sampled training example

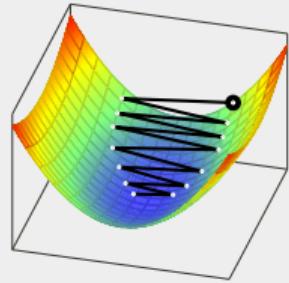
Batch Gradient Descent

$$L(\theta) = \frac{1}{b} \sum_{i=1}^b l(\hat{y}^{p_i}, y^{p_i}) \text{ with } p_i \text{ random in } [1, \dots, k]$$

- randomly sampled set of training examples

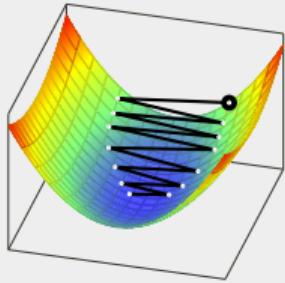
Plain Gradient Descent

$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta^t)$$



Plain Gradient Descent

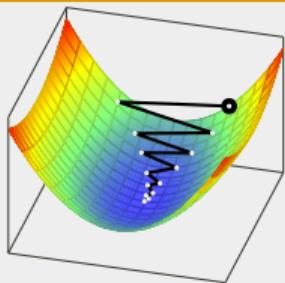
$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta^t)$$



Gradient Descent with Momentum

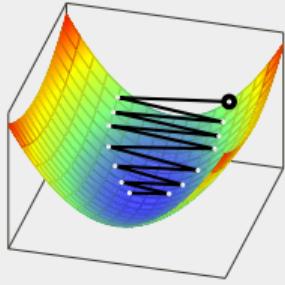
$$\delta^t = \alpha \nabla_{\theta} L(\theta^t) + \beta \delta^{t-1}$$

$$\theta^{t+1} = \theta^t - \delta^t$$



Plain Gradient Descent

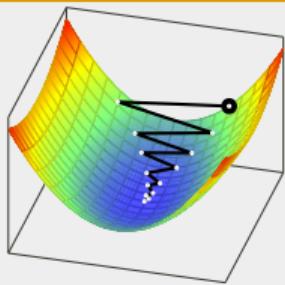
$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta} L(\theta^t)$$



Gradient Descent with Momentum

$$\delta^t = \alpha \nabla_{\theta} L(\theta^t) + \beta \delta^{t-1}$$

$$\theta^{t+1} = \theta^t - \delta^t$$



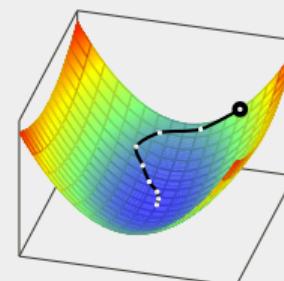
Adaptive Moment Estimation (Adam)

$$g^t = \nabla_{\theta} L(\theta^t)$$

$$m^t = \beta_1 m^{t-1} + (1 - \beta_1) g^t$$

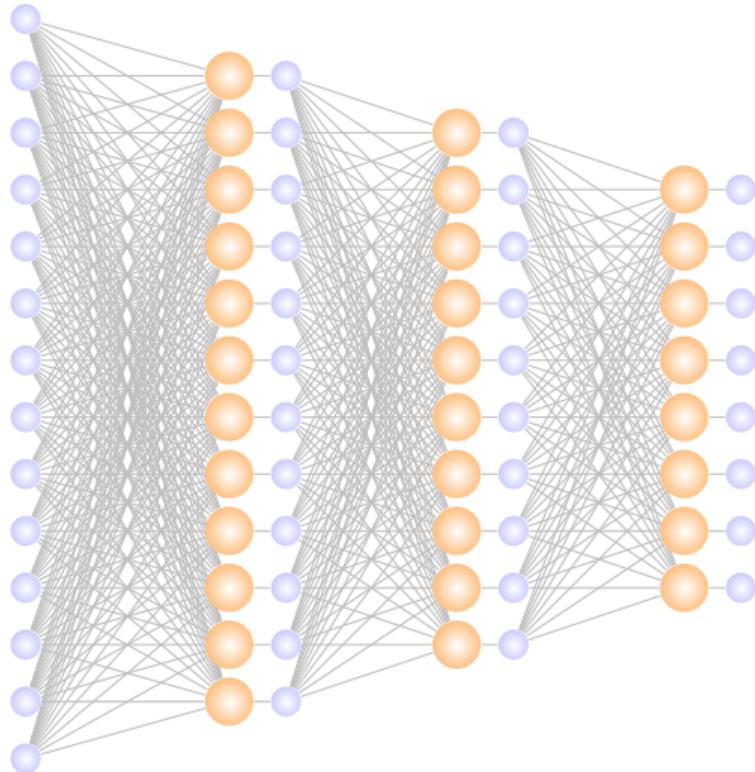
$$v^t = \beta_2 v^{t-1} + (1 - \beta_2) (g^t \odot g^t)$$

$$\theta^{t+1} = \theta^t - \alpha \frac{m^t}{\sqrt{v^t} + \epsilon}$$

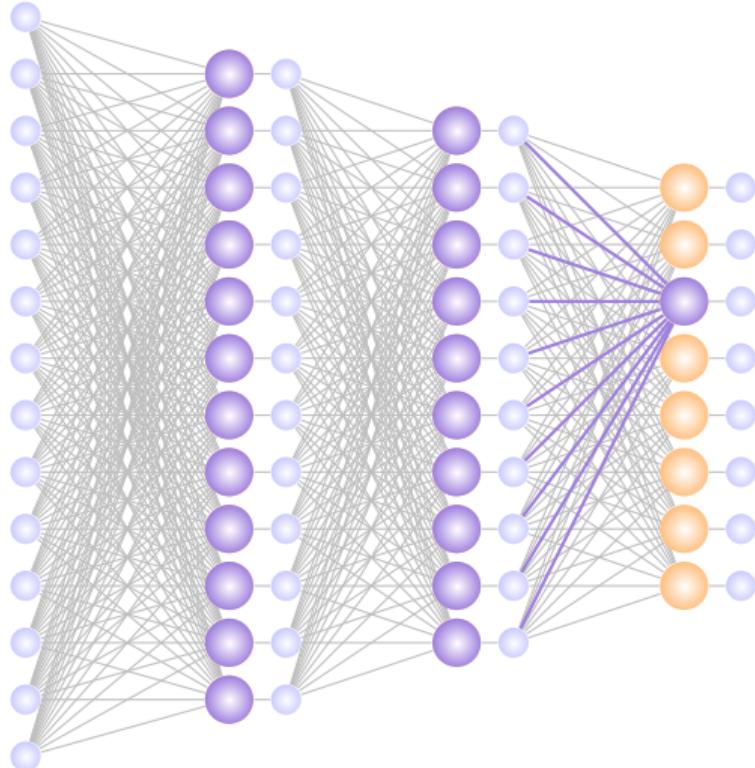


[5] 

Receptive Fields

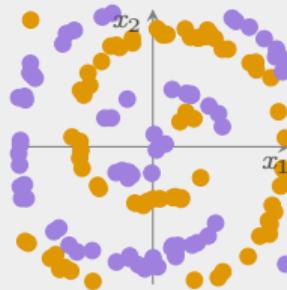


Receptive Fields

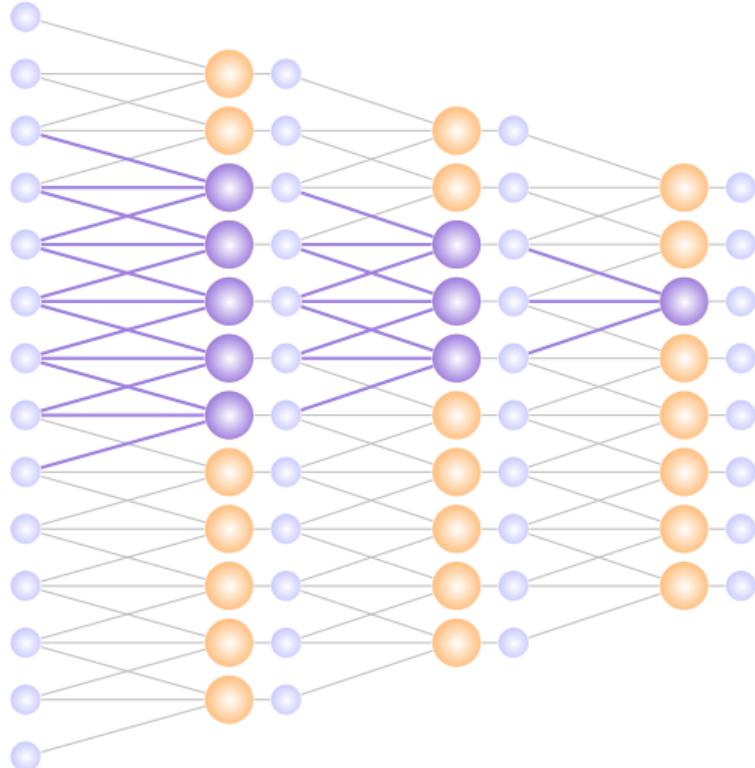


Fully Connected

- if inputs are unstructured



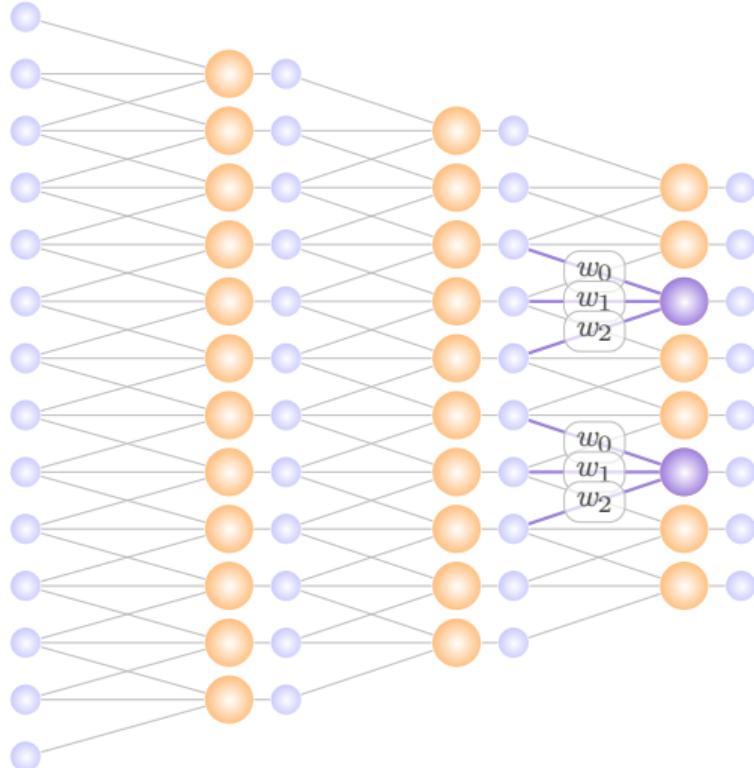
Receptive Fields



Locally Connected and Shared Weights

- if inputs have spatial/temporal structure

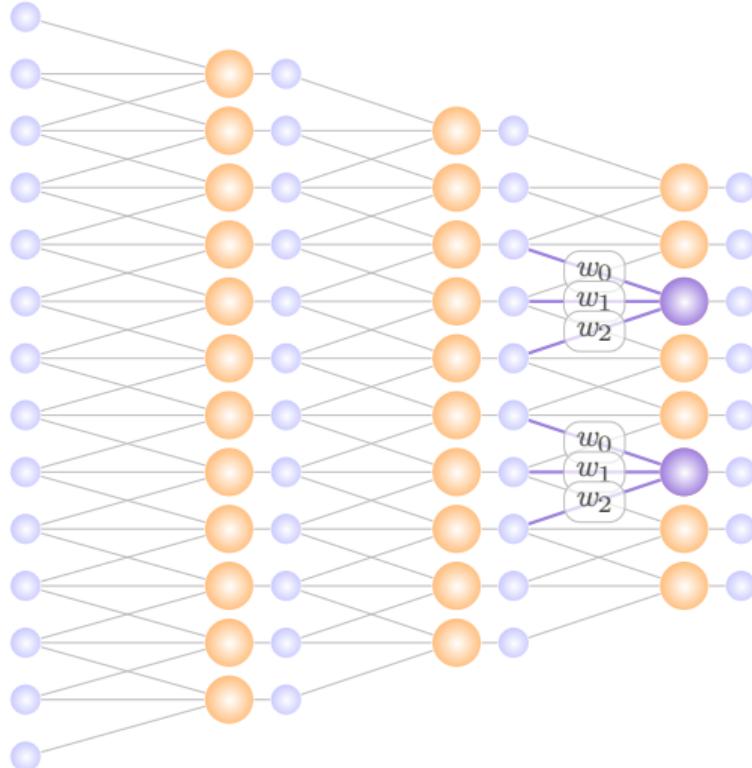
Receptive Fields



Locally Connected and Shared Weights

- if inputs have spatial/temporal structure
- if outputs are equivariant to inputs

Receptive Fields



Locally Connected and Shared Weights

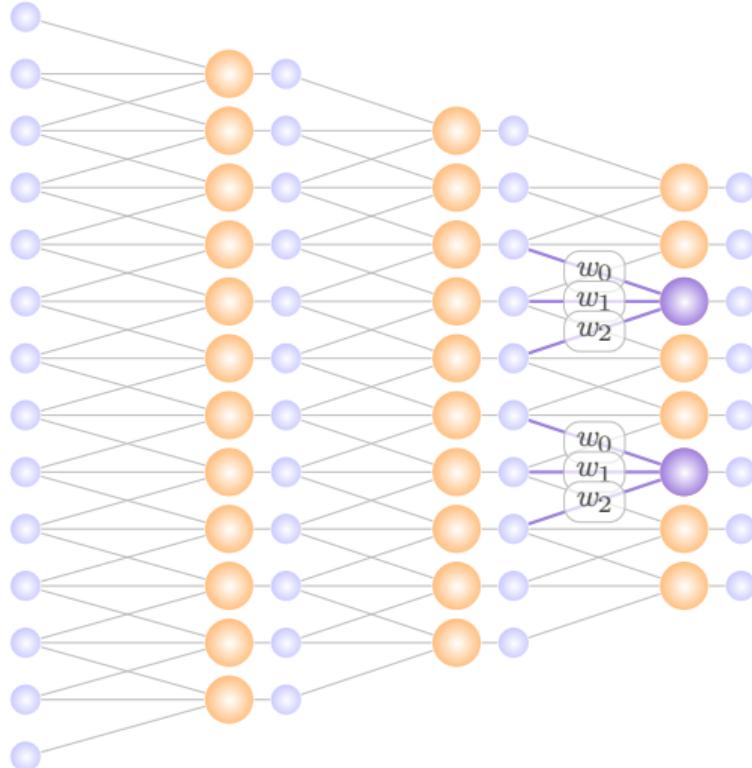
- if inputs have spatial/temporal structure
- if outputs are equivariant to inputs

$$y_i = f(\mathbf{W}h + \mathbf{b})_i$$

$$= f\left(\sum_{j=0}^2 h_{i+j} w_j + b_i\right)$$

$$\mathbf{y} = f(\mathbf{h} * \mathbf{k} + \mathbf{b}) \text{ with } \mathbf{k} = (w_j)_{j=0,\dots,2}$$

Receptive Fields



Locally Connected and Shared Weights

- if inputs have spatial/temporal structure
- if outputs are equivariant to inputs

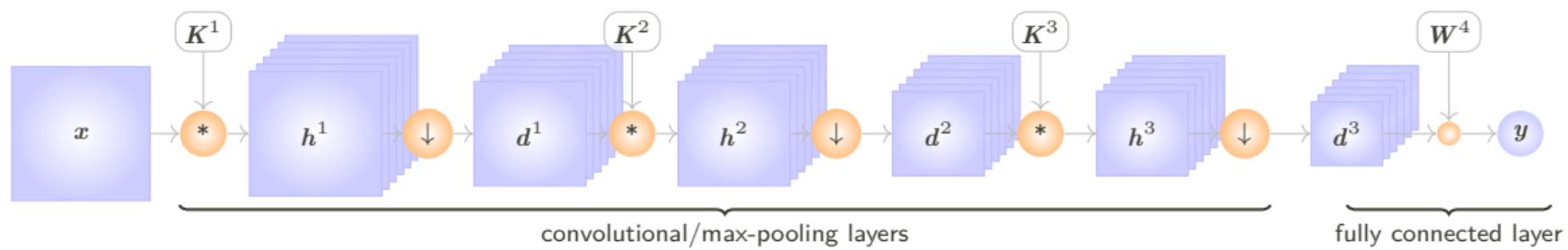
$$y_i = f(\mathbf{W}h + \mathbf{b})_i$$

$$= f\left(\sum_{j=0}^2 h_{i+j} w_j + b_i\right)$$

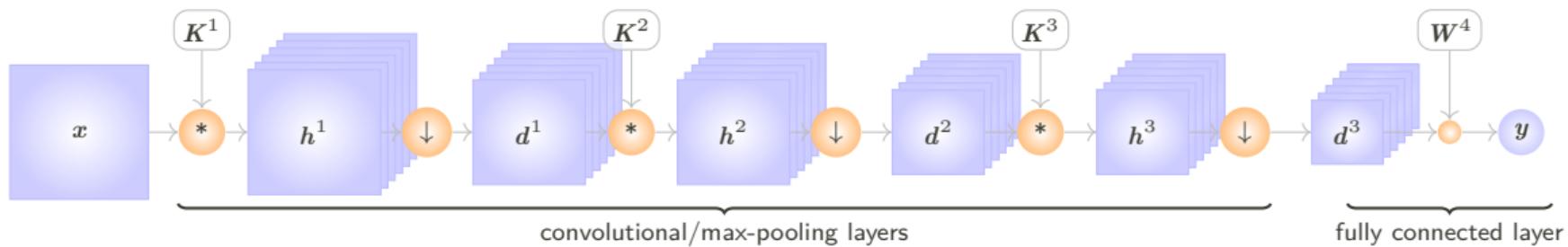
$$\mathbf{y} = f(\mathbf{h} * \mathbf{k} + \mathbf{b}) \text{ with } \mathbf{k} = (w_j)_{j=0,\dots,2}$$

→ convolutional neural networks (CNNs)

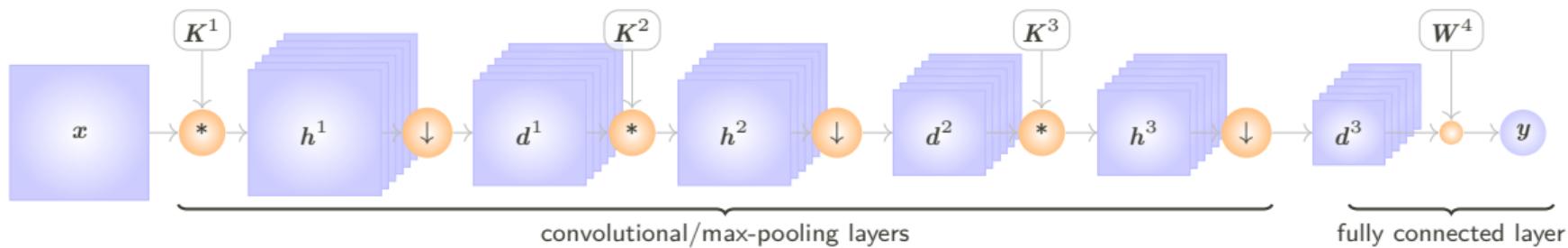
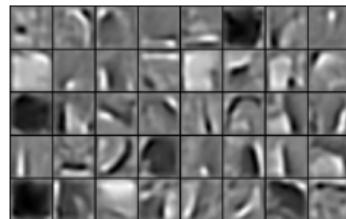
Convolutional Neural Networks



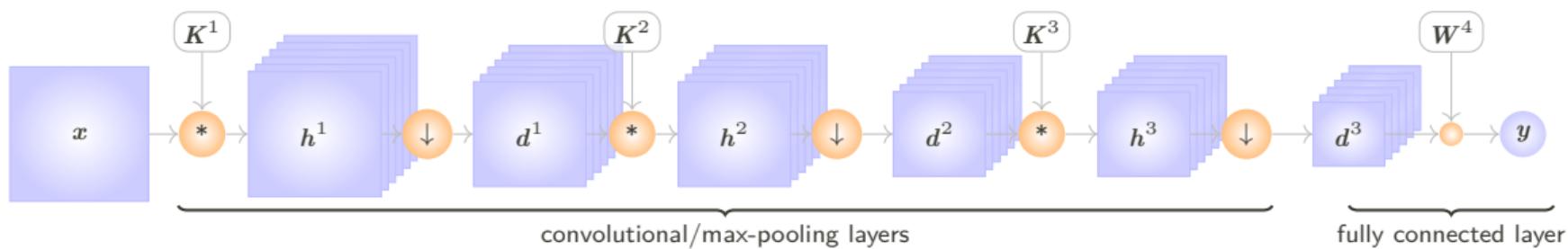
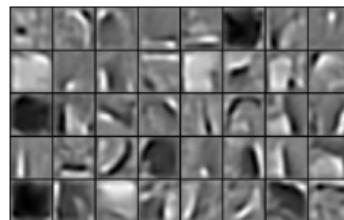
Convolutional Neural Networks



Convolutional Neural Networks

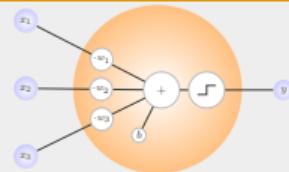


Convolutional Neural Networks

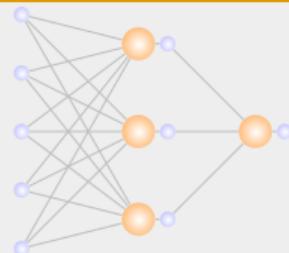


Summary

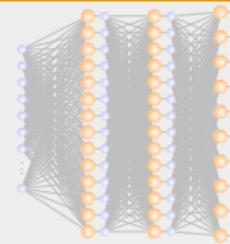
1. Perceptrons



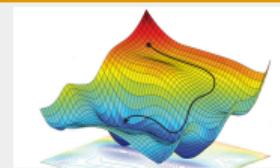
2. Networks



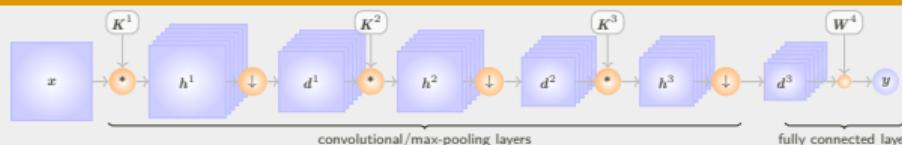
3. Deep Networks



4. Learning



5. Convolutional Networks



References

- [1] Pollack, J. B.
No harm intended: A review of the perceptrons expanded edition.
Journal of Mathematical Psychology (1989).
- [2] Davey, B. & Priestley, H.
Introduction to Lattices and Order (Cambridge Mathematical Textbooks, 1990).
- [3] Cybenko, G.
Approximations by superpositions of sigmoidal functions.
Mathematics of Control, Signals, and Systems (1989).
- [4] URL <https://www.youtube.com/watch?v=aircAruvnKk>.
- [5] Kingma, D. P. & Ba, J.
Adam: A method for stochastic optimization.
CoRR abs/1412.6980 (2014).
URL <http://arxiv.org/abs/1412.6980>.
1412.6980.