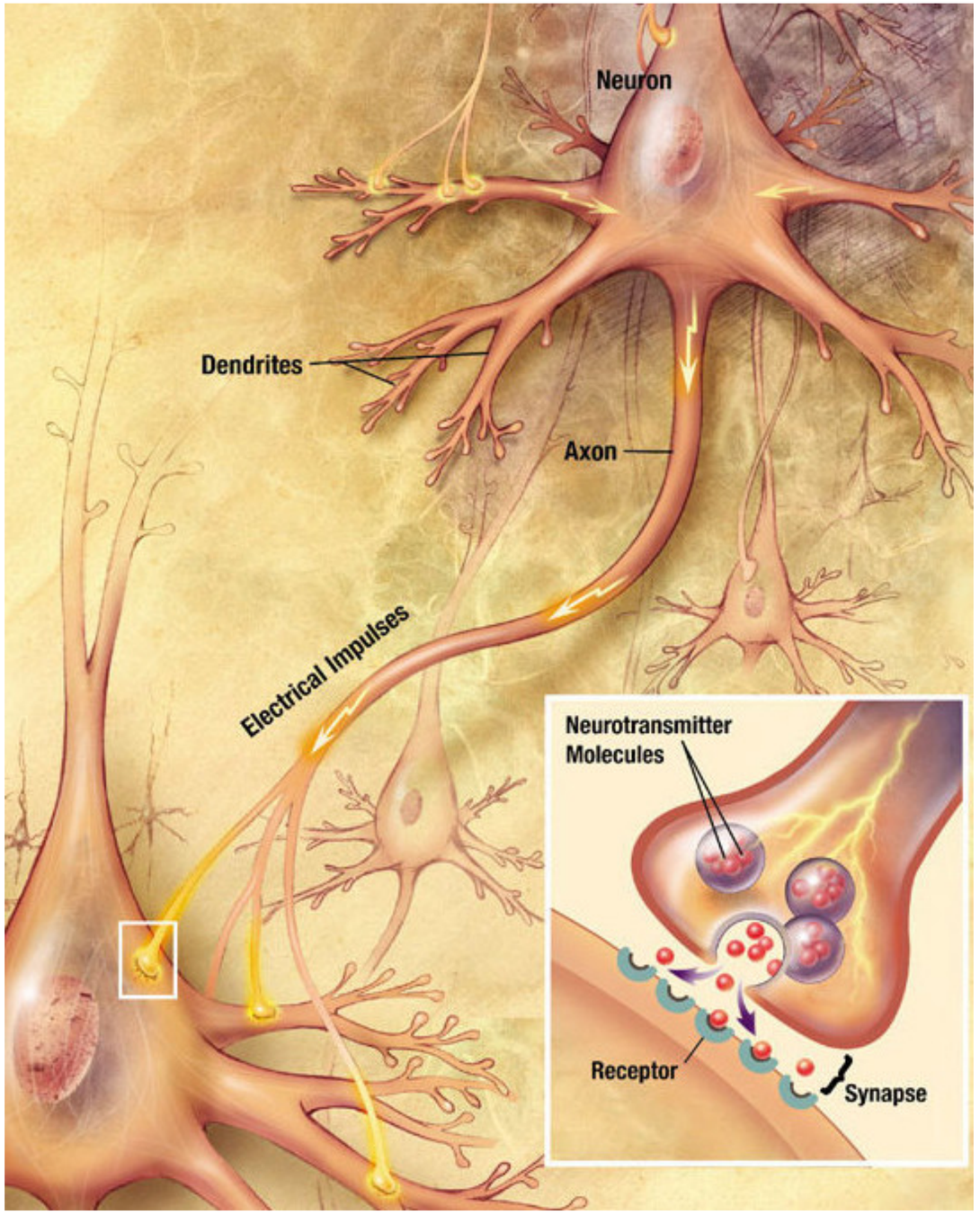
Neural Networks

Intro to AI

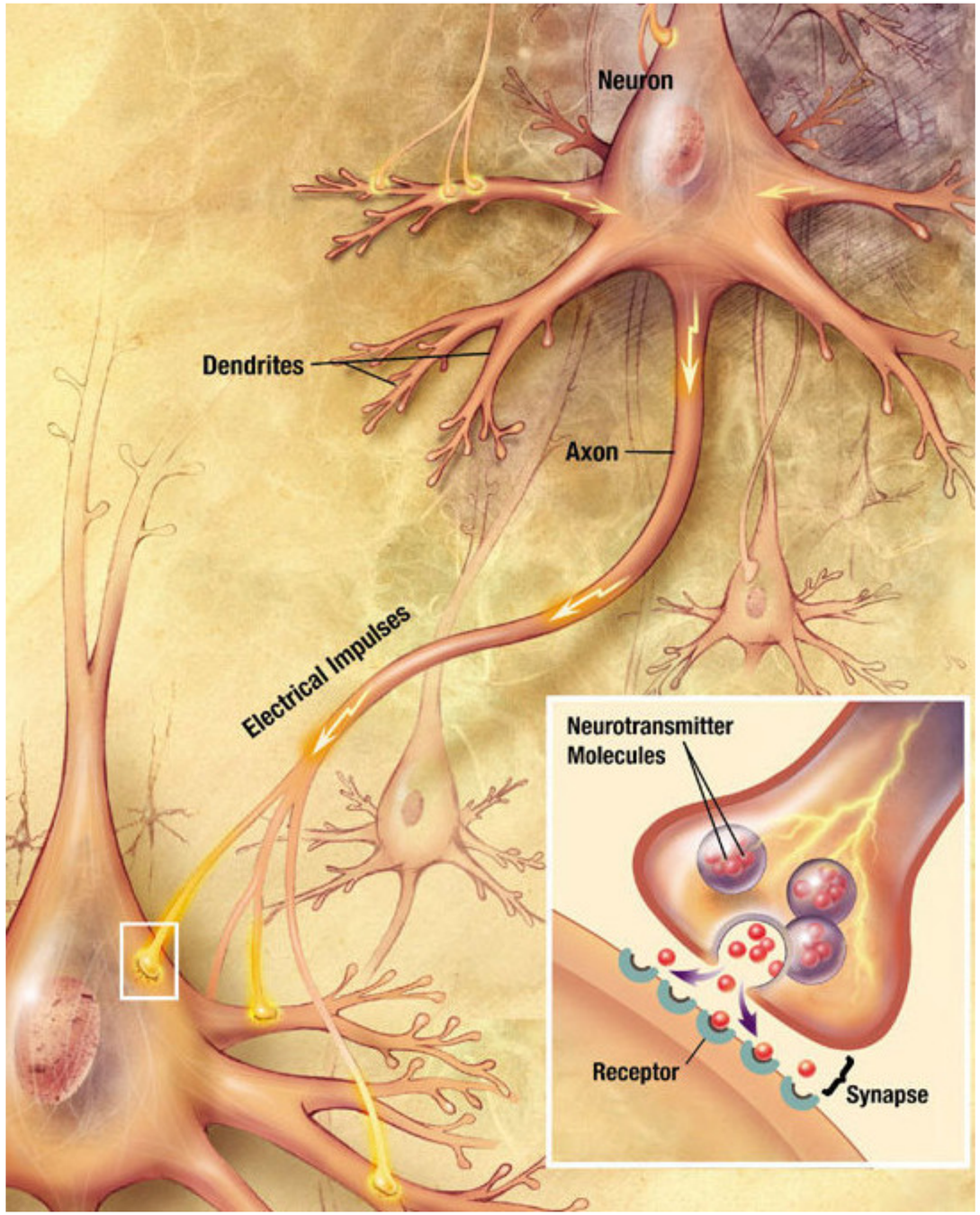
Bert Huang
Virginia Tech

Outline

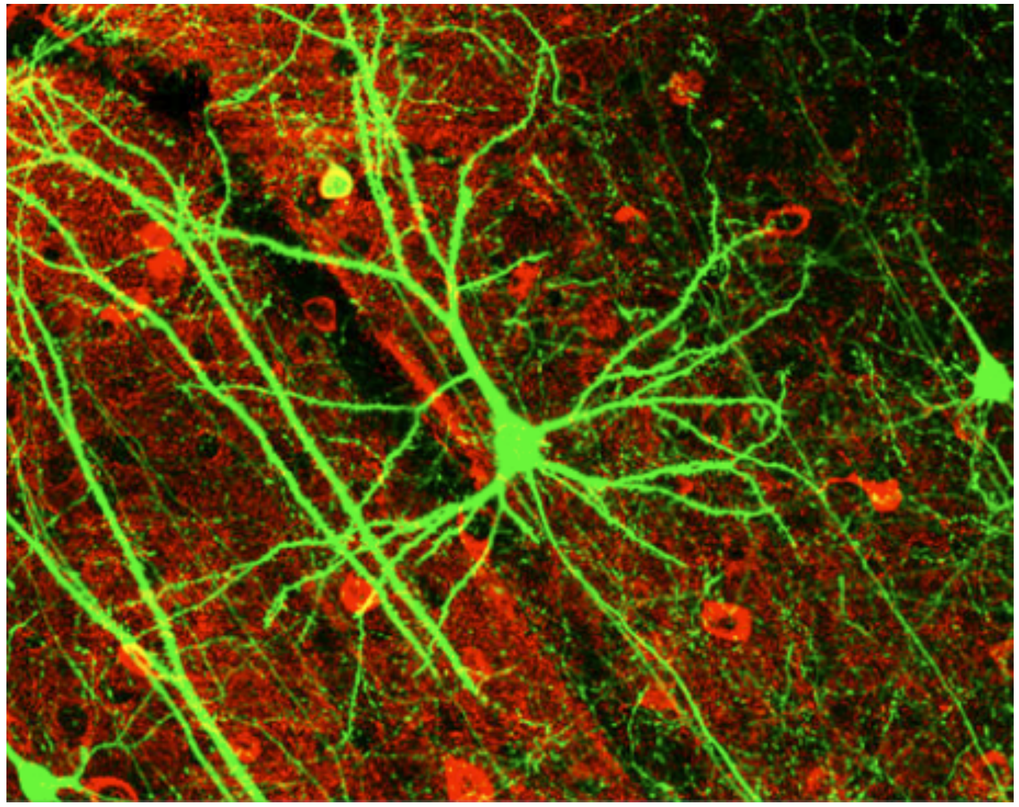
- Biological inspiration for artificial neural networks
- Linear vs. nonlinear functions
- Learning with neural networks: back propagation



https://en.wikipedia.org/wiki/Neuron#/media/File:Chemical_synapse_schema_cropped.jpg



https://en.wikipedia.org/wiki/Neuron#/media/File:Chemical_synapse_schema_cropped.jpg



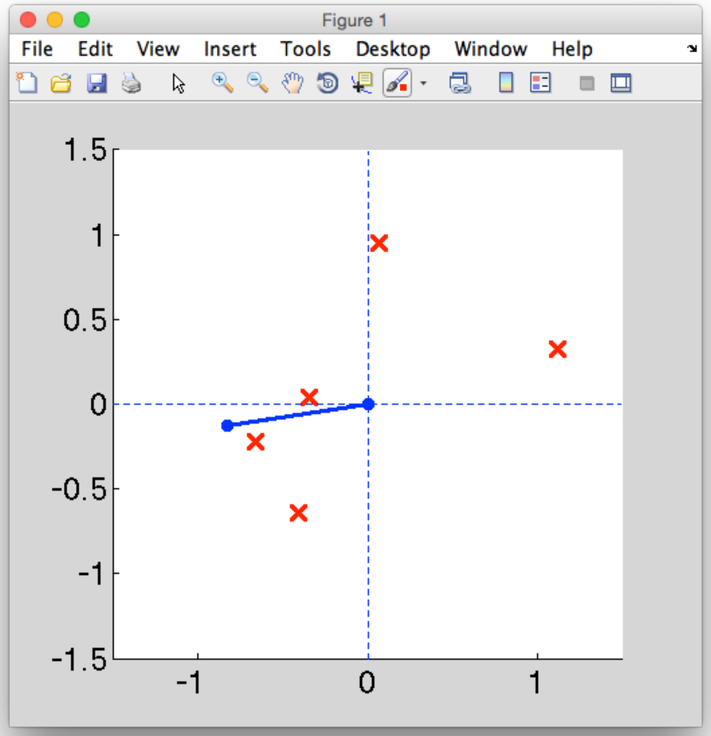
https://en.wikipedia.org/wiki/Neuron#/media/File:GFPneuron.png

Parameterizing p(y|x)

$$p(y|x) := f$$

$$f : \mathbb{R}^d \rightarrow [0, 1]$$

$$f(x) := \frac{1}{1 + \exp(-w^\top x)}$$

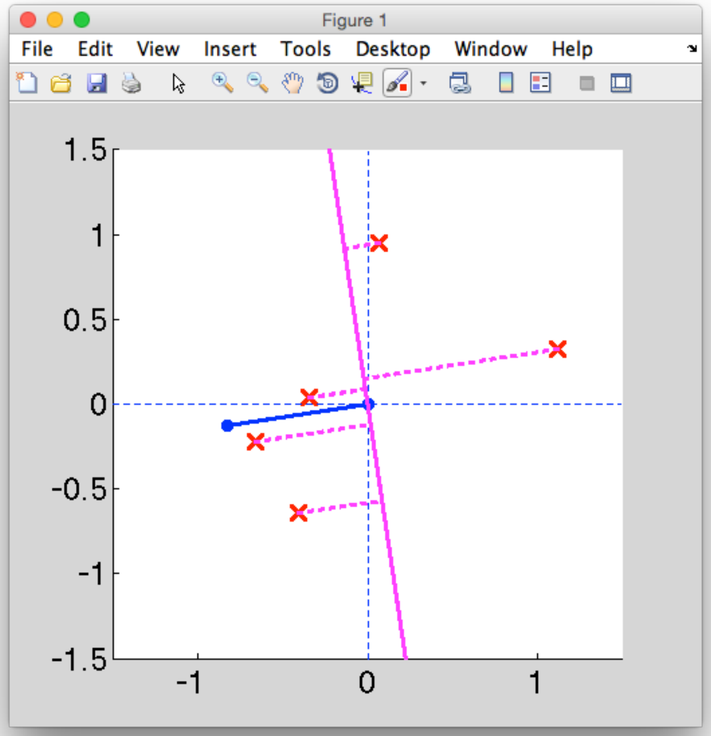


Parameterizing p(y|x)

$$p(y|x) := f$$

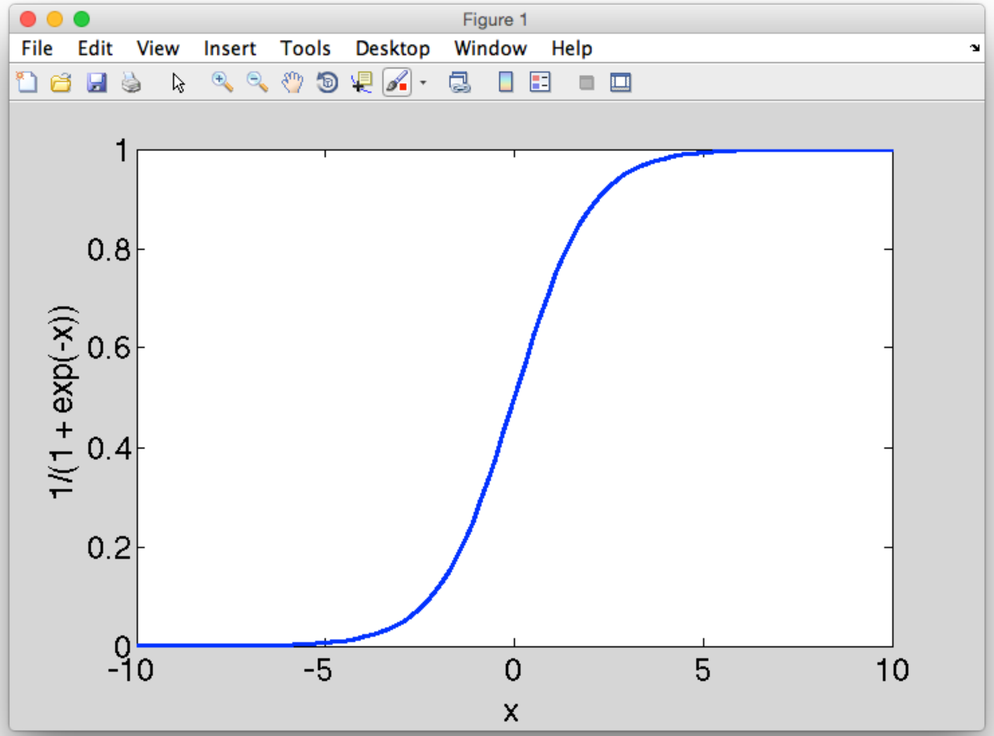
$$f : \mathbb{R}^d \rightarrow [0, 1]$$

$$f(x) := \frac{1}{1 + \exp(-w^\top x)}$$



Logistic Function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



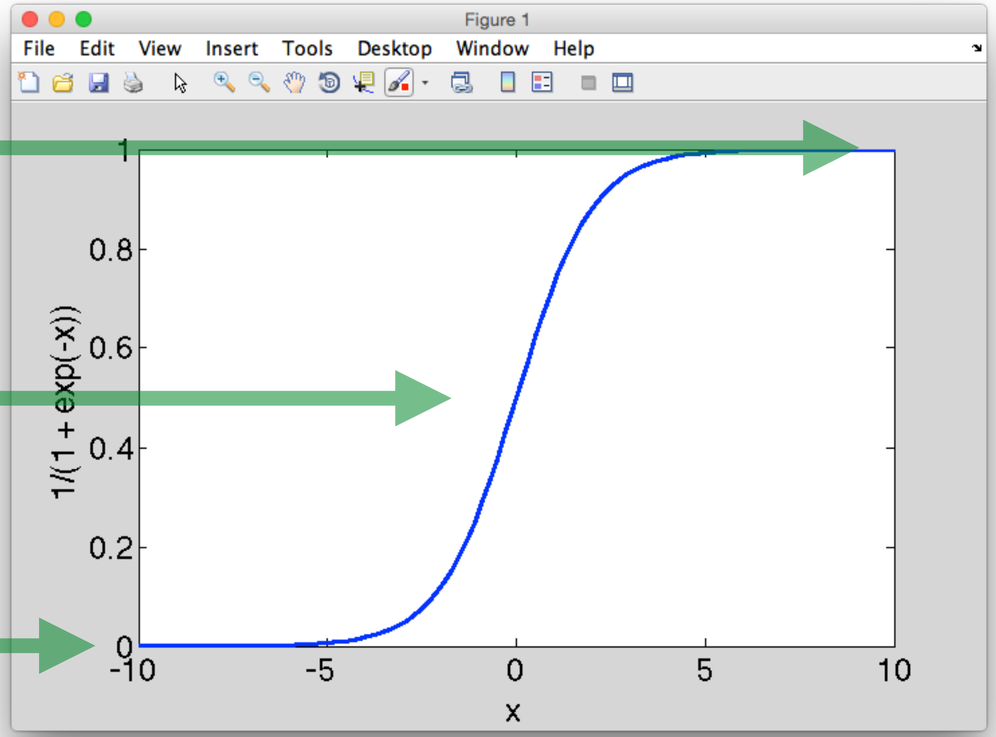
Logistic Function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$\lim_{x \to \infty} \sigma(x) = \lim_{x \to \infty} \frac{1}{1 + \exp(-x)} = \frac{1}{1} = 1.0$$

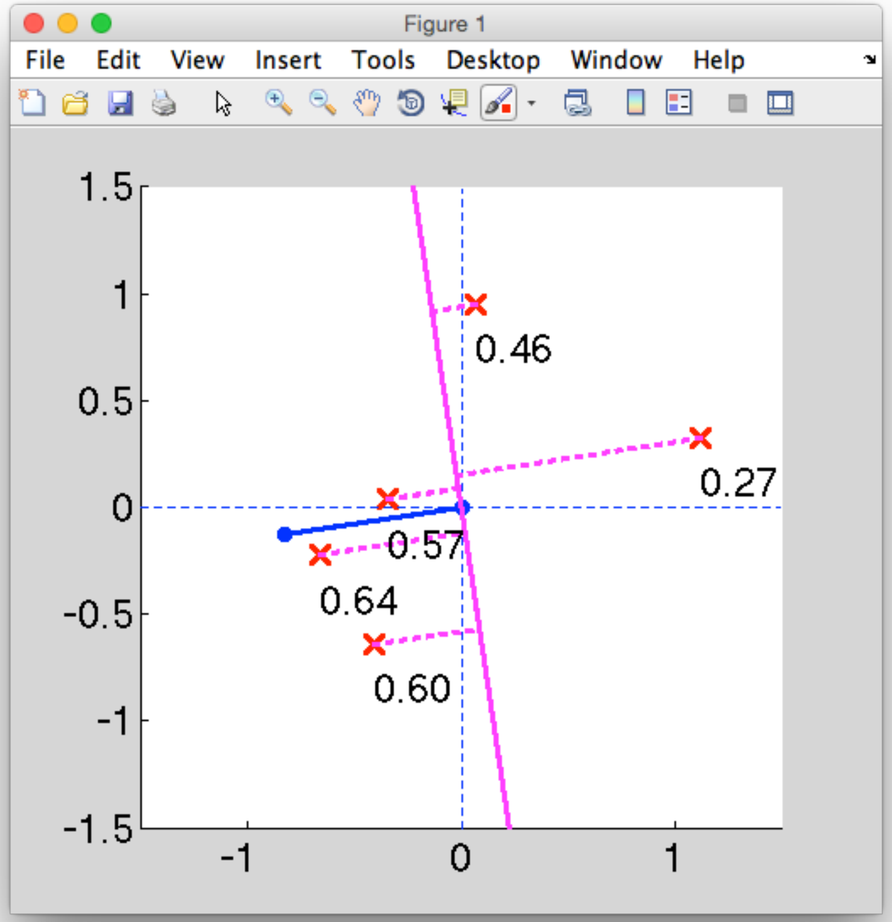
$$\sigma(0) = \frac{1}{1 + \exp(-0)} = \frac{1}{1 + 1} = 0.5$$

$$\lim_{x \to -\infty} \sigma(x) = \lim_{x \to -\infty} \frac{1}{1 + \exp(-x)} = 0.0$$



From Features to Probability

$$f(x) := \frac{1}{1 + \exp(-w^\top x)}$$

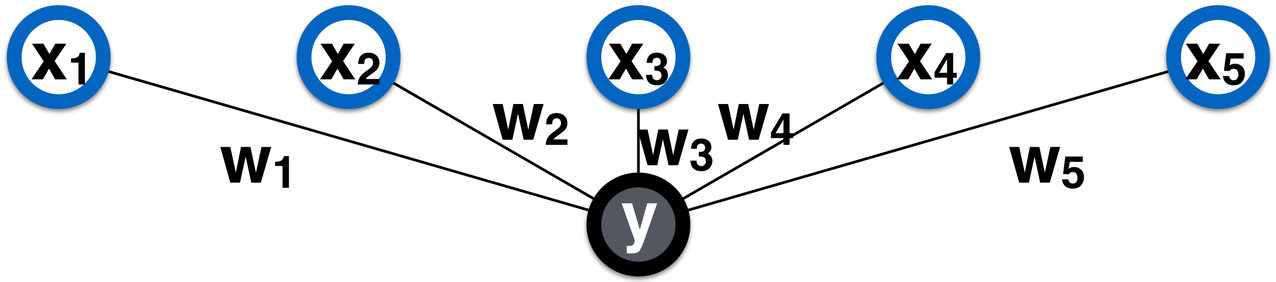


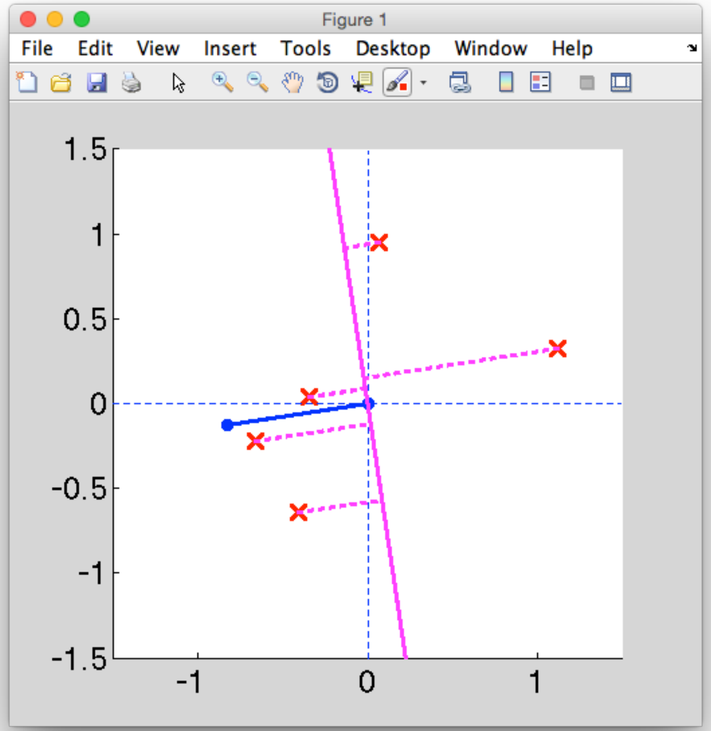
Parameterizing p(y|x)

$$p(y|x) := f$$

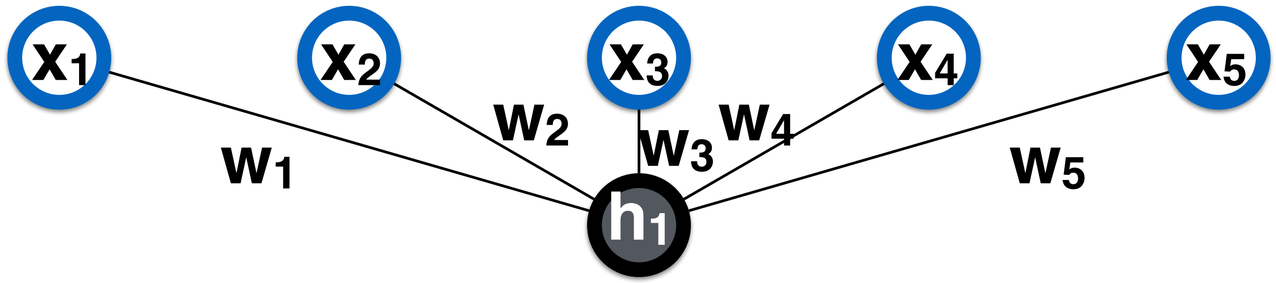
$$f : \mathbb{R}^d \rightarrow [0, 1]$$

$$f(x) := \frac{1}{1 + \exp(-w^\top x)}$$





Multi-Layered Perceptron

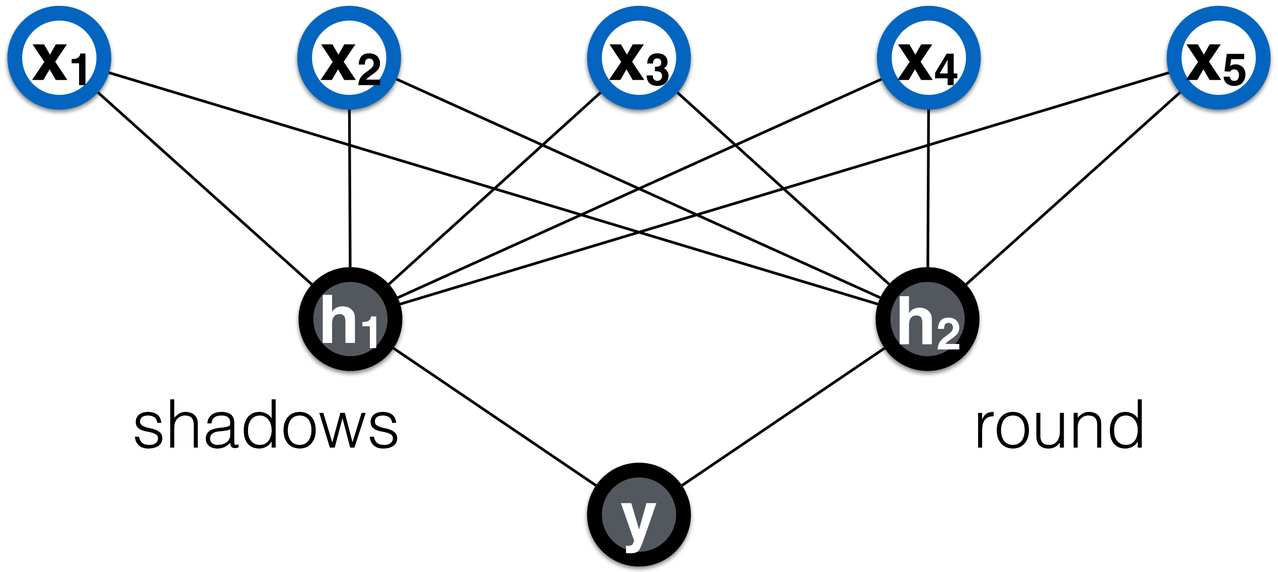


Multi-Layered Perceptron

raw data

representation

prediction

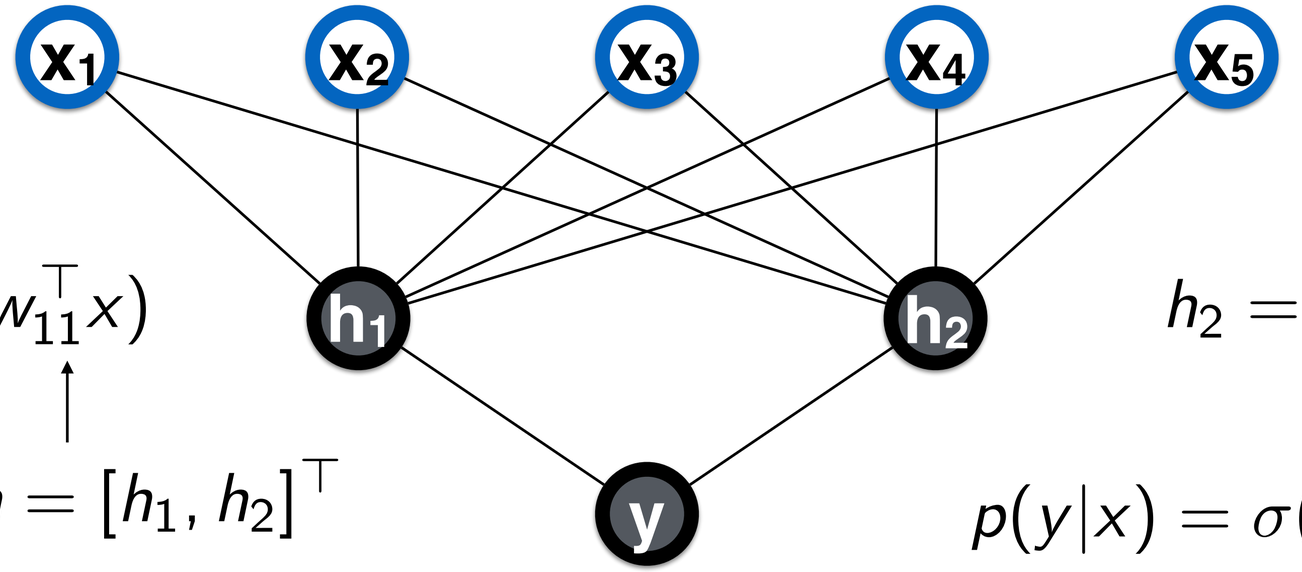


pixel values

shapes

faces

Multi-Layered Perceptron



$$h_1 = \sigma(w_{11}^\top x)$$

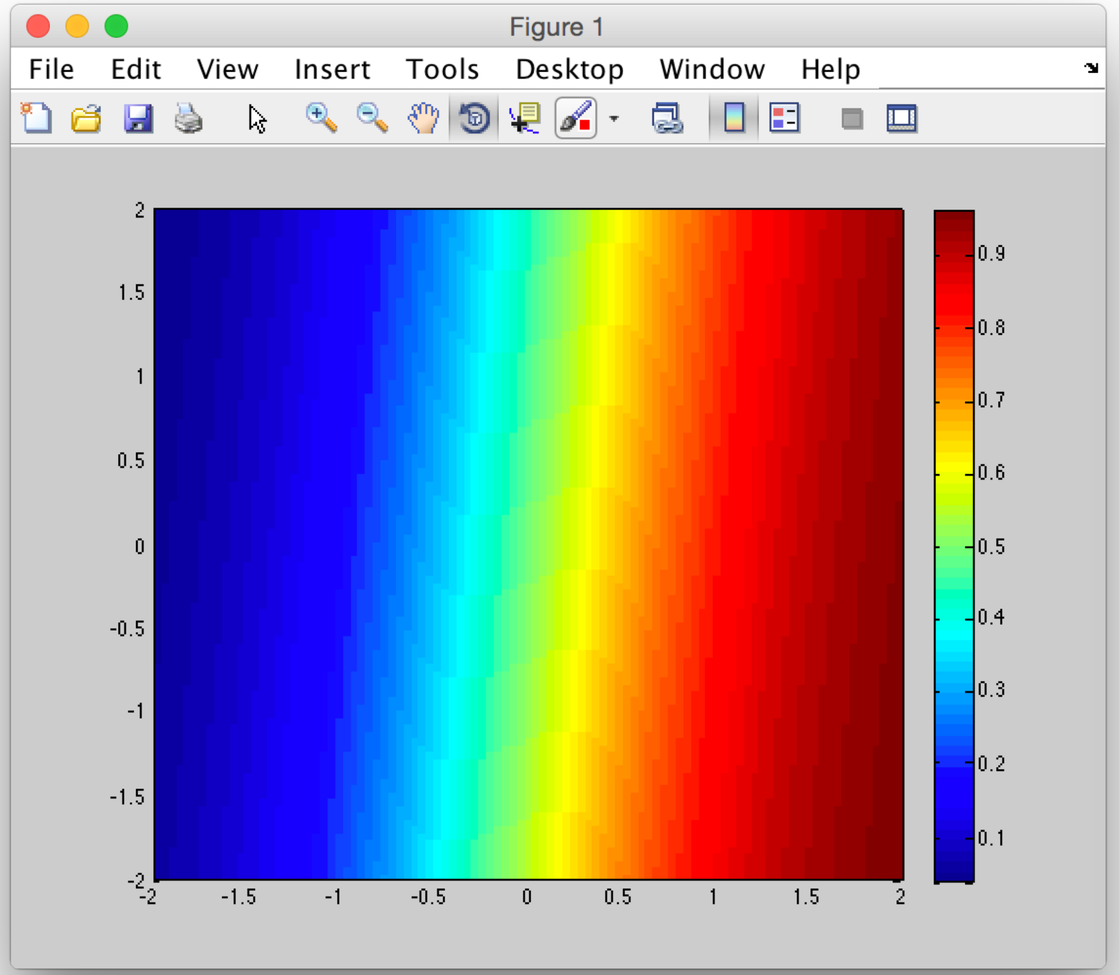
$$h = [h_1, h_2]^\top$$

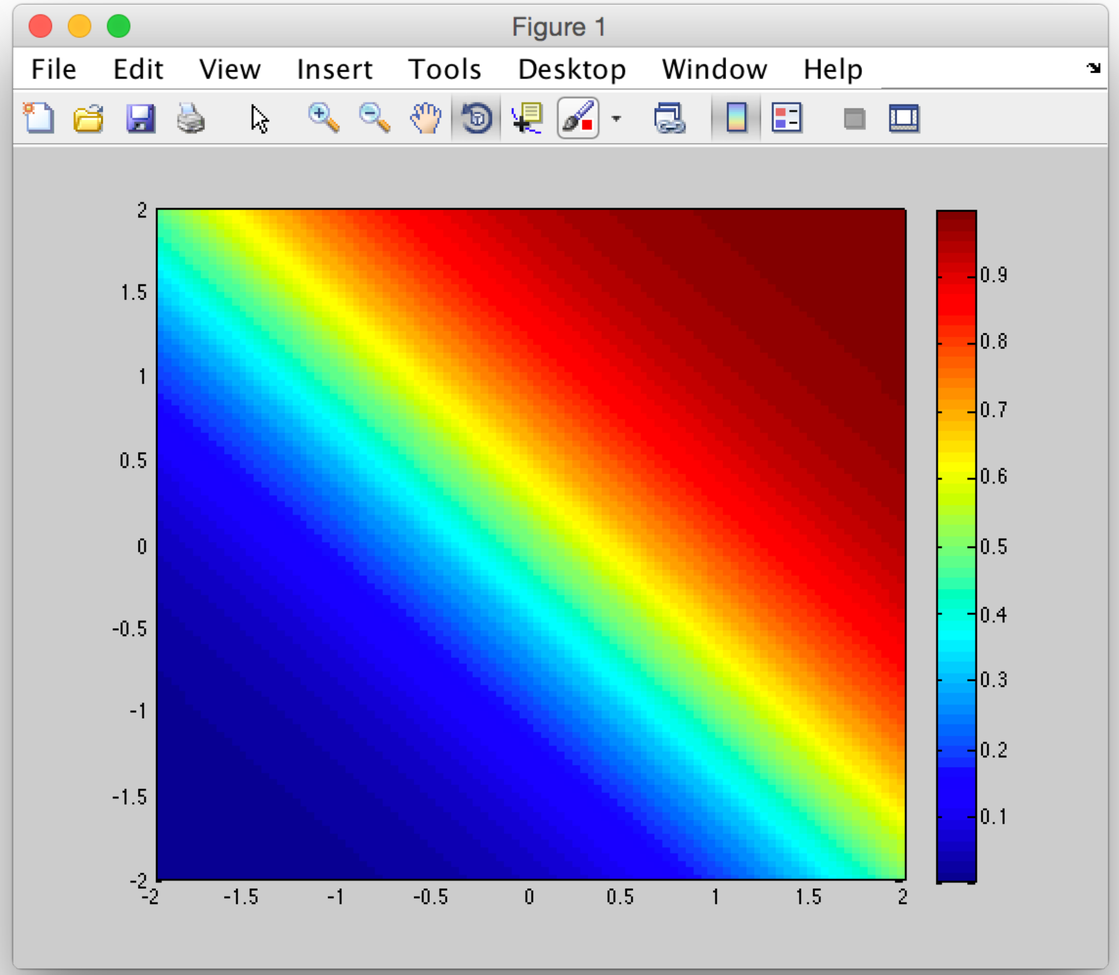
$$h_2 = \sigma(w_{12}^\top x)$$

$$p(y|x) = \sigma(w_{21}^\top h)$$

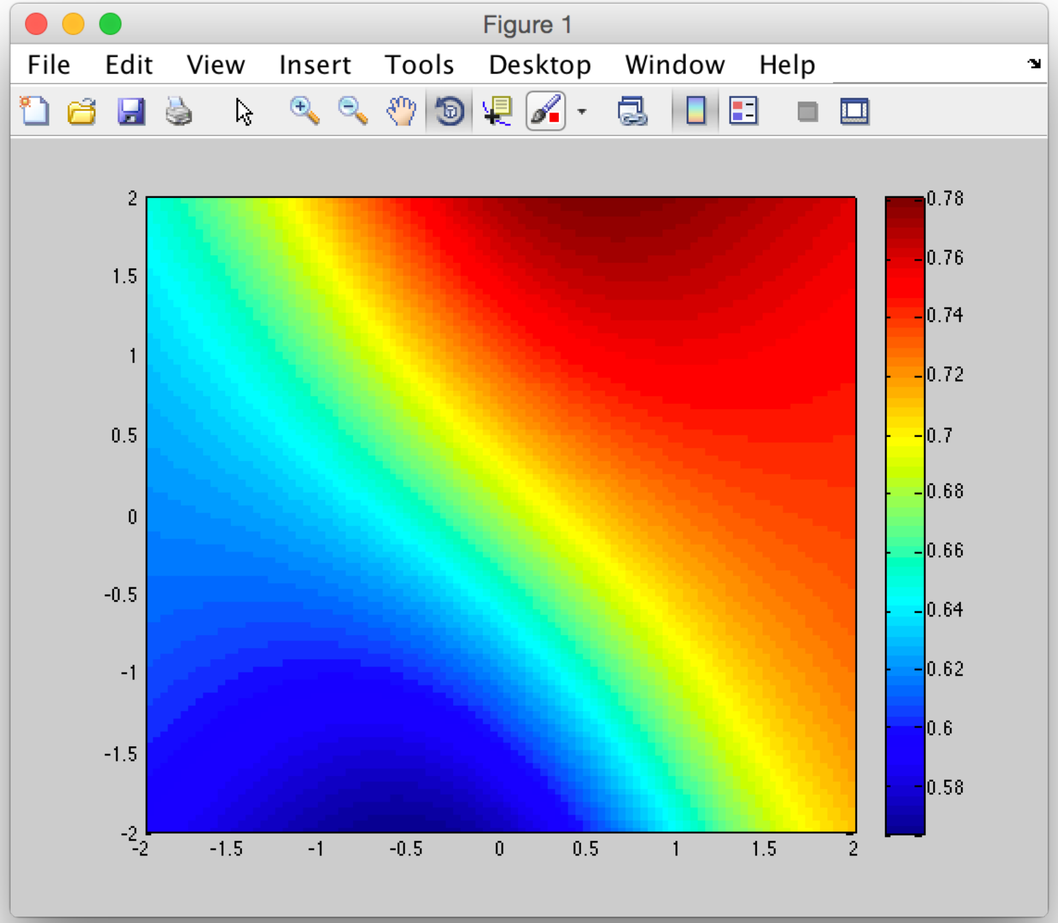
$$p(y|x) = \sigma\left(w_{21}^\top \left[\sigma(w_{11}^\top x), \sigma(w_{12}^\top x)\right]^\top\right)$$

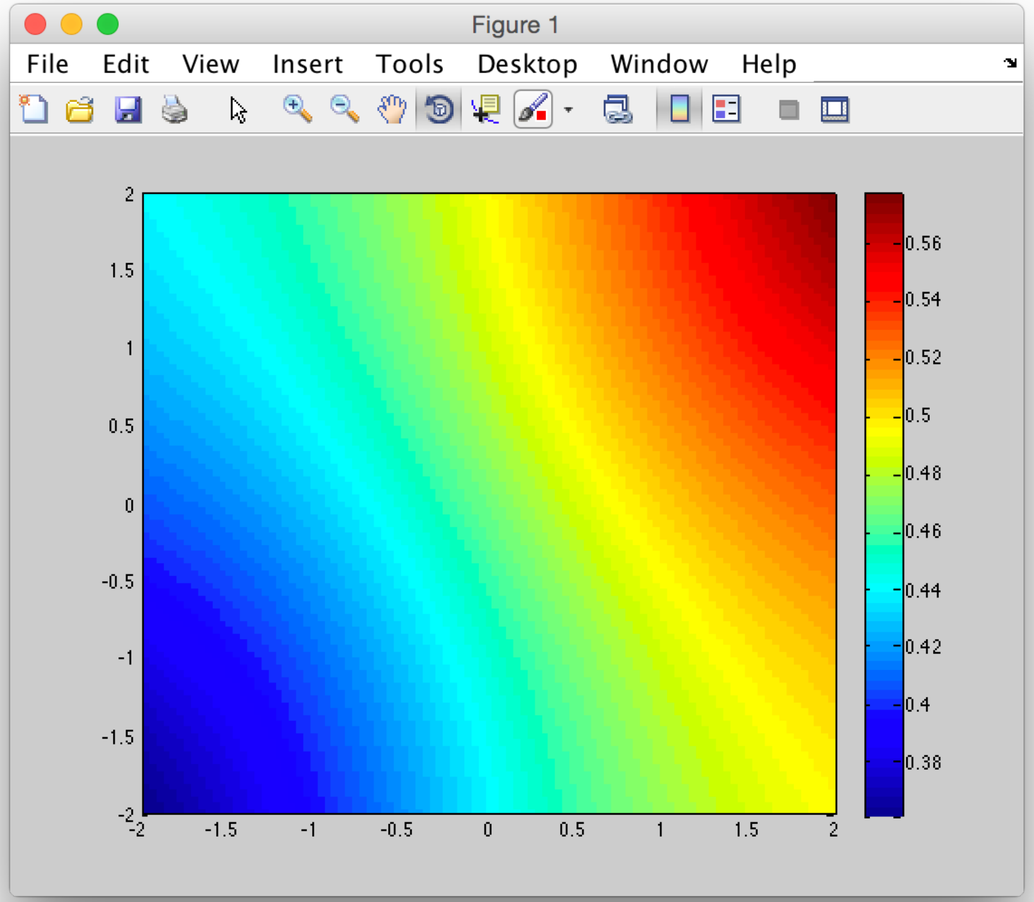
Decision Surface: Logistic Regression



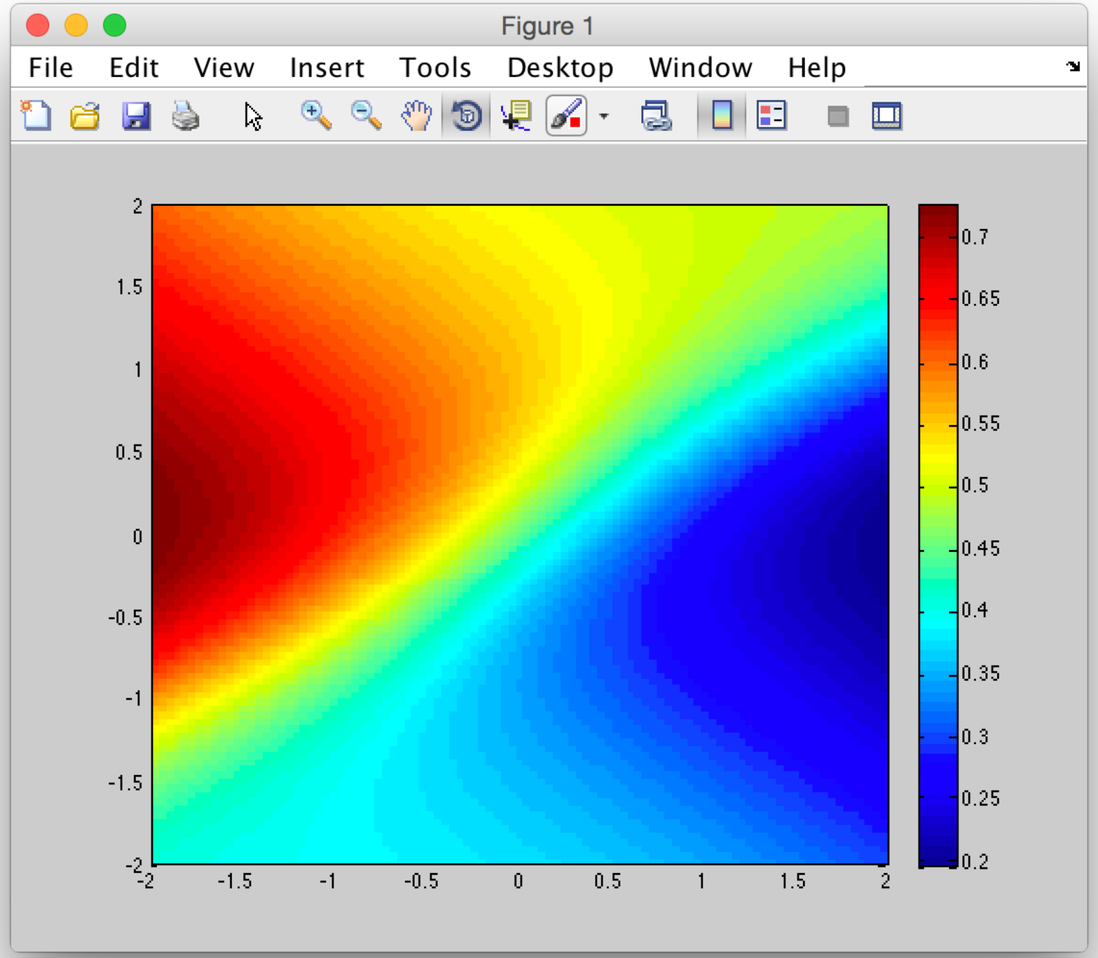


Decision Surface: 2-Layer, 2 Hidden Units

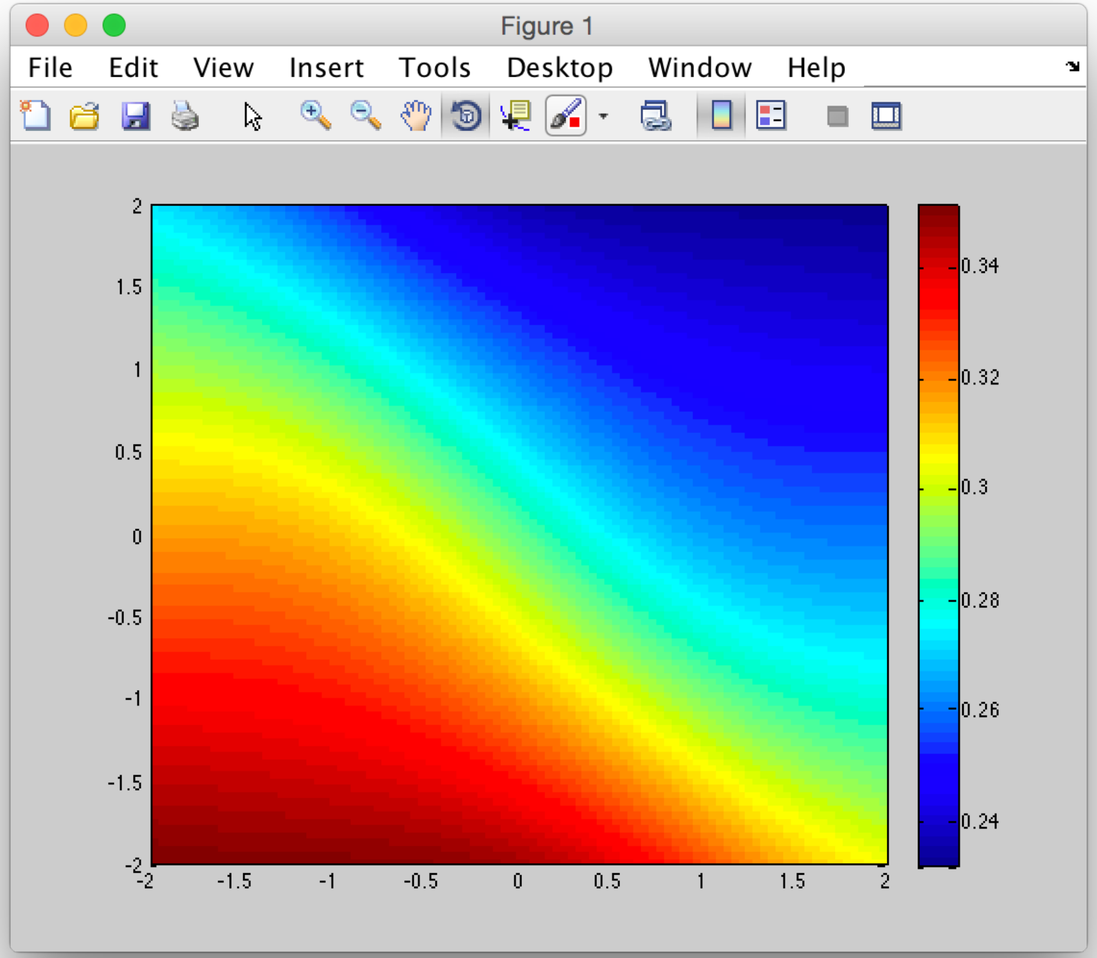




Decision Surface: 2-Layer, More Hidden Units

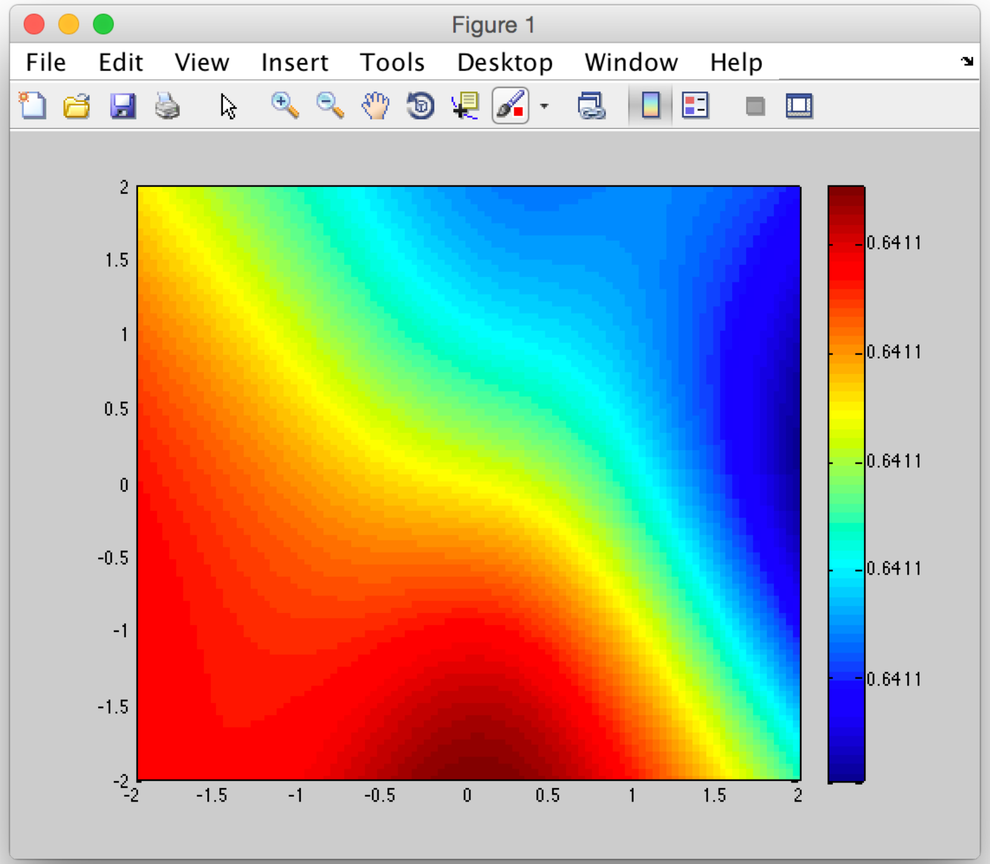


3 hidden units

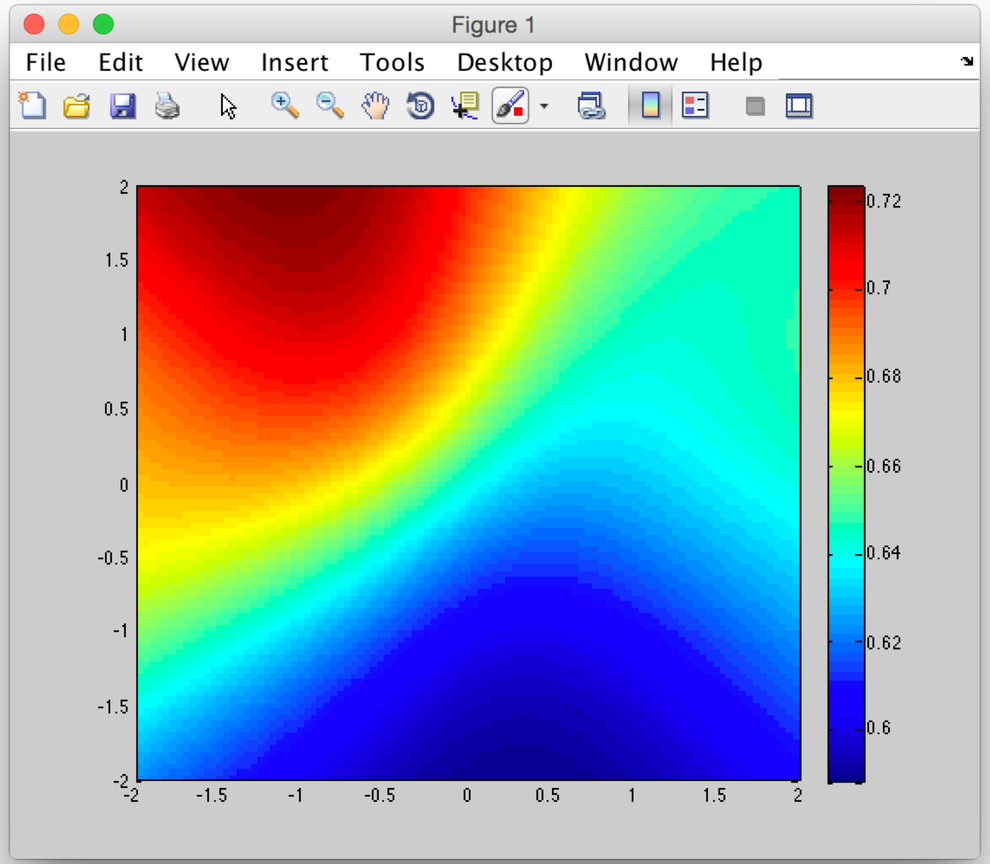


10 hidden units

Decision Surface: More Layers, More Hidden Units



10 layers, 5 hidden units per layer



4 layers, 10 hidden units each layer

Training Neural Networks

$$\min_W \frac{1}{n}\sum_{i=1}^{n} l(f(x_i, W), y_i) = L(W)$$

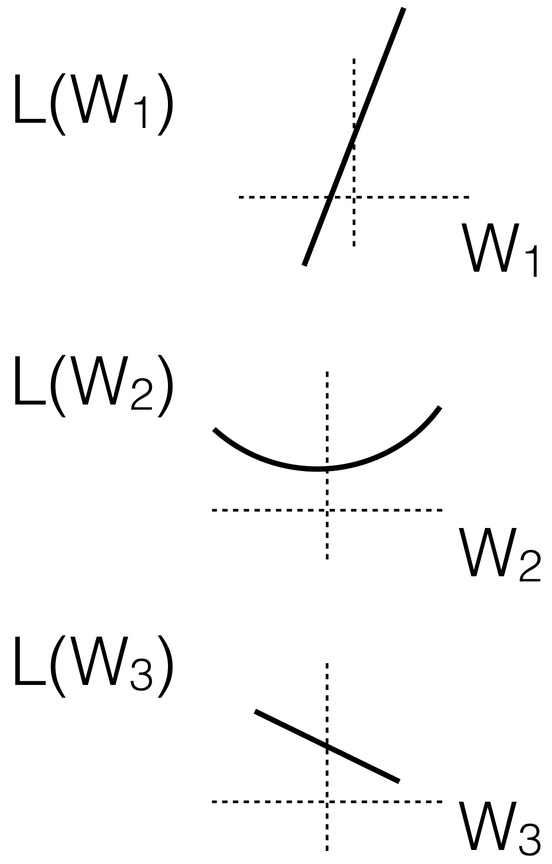
average training error

$$W_j \leftarrow W_j - \alpha\left(\frac{\partial L}{\partial W_j}\right)$$

Gradient Descent

Gradient Descent

$$W_j \leftarrow W_j - \alpha \left(\frac{\partial L}{\partial W_j} \right)$$



very positive
take big step left

almost zero
take tiny step left

slightly negative
take medium step right

Approximate Q-Learning

$$\hat{Q}(s,a) := g(s,a,\boldsymbol{\theta}) := \theta_1 f_1(s,a) + \theta_2 f_2(s,a) + \ldots + \theta_d f_d(s,a)$$

$$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \frac{\partial g}{\partial \theta_i}$$

$$\theta_i \leftarrow \theta_i + \alpha \left(R(s) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) f_i(s,a)$$

Back Propagation

- Back propagation:
  - Compute hidden unit activations: **forward propagation**
  - Compute gradient at output layer: error
  - Propagate error back one layer at a time
- Chain rule via dynamic programming

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Chain rule

From Wikipedia, the free encyclopedia

This article is about the chain rule in calculus. For the chain rule in probability theory, see Chain rule (probability). For other uses, see Chain rule (disambiguation).

In calculus, the **chain rule** is a formula for computing the derivative of the composition of two or more functions. That is, if *f* and *g* are functions, then the chain rule expresses the derivative of their composition $f \circ g$ (the function which maps *x* to *f*(*g*(*x*))) in terms of the derivatives of *f* and *g* and the product of functions as follows:

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

This may equivalently be expressed in terms of the variable. Let $F = f \circ g$, or equivalently, $F(x) = f(g(x))$ for all *x*. Then one can also write

$$F'(x) = f'(g(x))g'(x).$$

The chain rule may be written in Leibniz's notation in the following way. If a variable *z* depends on the variable *y*, which itself depends on the variable *x*, so that *y* and *z* are therefore dependent variables, then *z*, via the intermediate variable of *y*, depends on *x* as well. The chain rule then states,

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}.$$

The two versions of the chain rule are related; if $z = f(y)$ and $y = g(x)$, then

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = f'(y)g'(x) = f'(g(x))g'(x).$$

Part of a series of articles about
Calculus

Fundamental theorem
Limits of functions · Continuity
Mean value theorem · Rolle's theorem

Differential [hide]

Definitions
Derivative (generalizations)
Differential (infinitesimal · of a function · total)

Concepts
Differentiation notation · Second derivative · Third derivative · Change of variables · Implicit differentiation · Related rates · Taylor's theorem

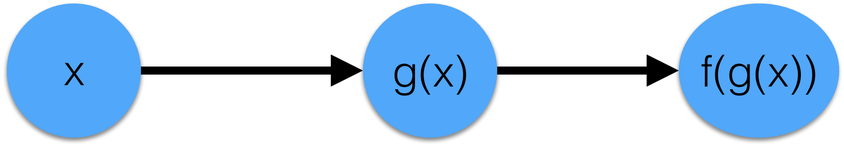
Rules and identities
Sum · Product · **Chain** · Power · Quotient · Inverse · General Leibniz · Faà di Bruno's formula

Integral [show]

Chain Rule Review

$$f(g(x))$$

$$\frac{d\, f(g(x))}{d\, x} = \frac{d\, f(g(x))}{d\, g(x)} \cdot \frac{d\, g(x)}{d\, x}$$



Chain Rule on More Complex Function

$$h(f(a) + g(b))$$

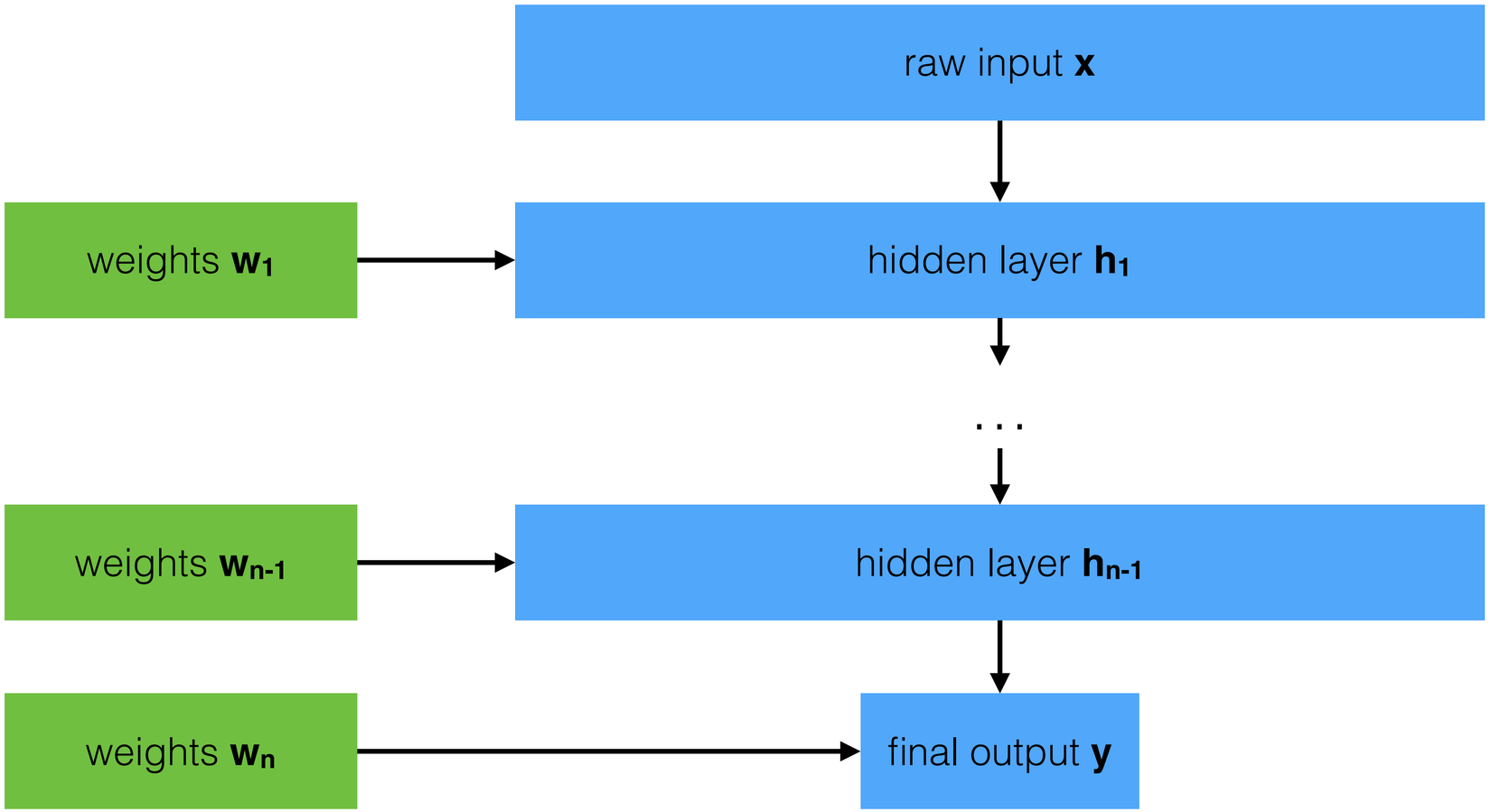
$$\frac{d\ h(f(a) + g(b))}{d\ a}$$

$$\frac{d\ h(f(a) + g(b))}{d\ b}$$

$$\frac{d\ h(f(a) + g(b))}{d\ f(a) + g(b)} \cdot \frac{d\ f(a)}{d\ a}$$

$$\frac{d\ h(f(a) + g(b))}{d\ f(a) + g(b)} \cdot \frac{d\ g(b)}{d\ b}$$

Back to Neural Networks

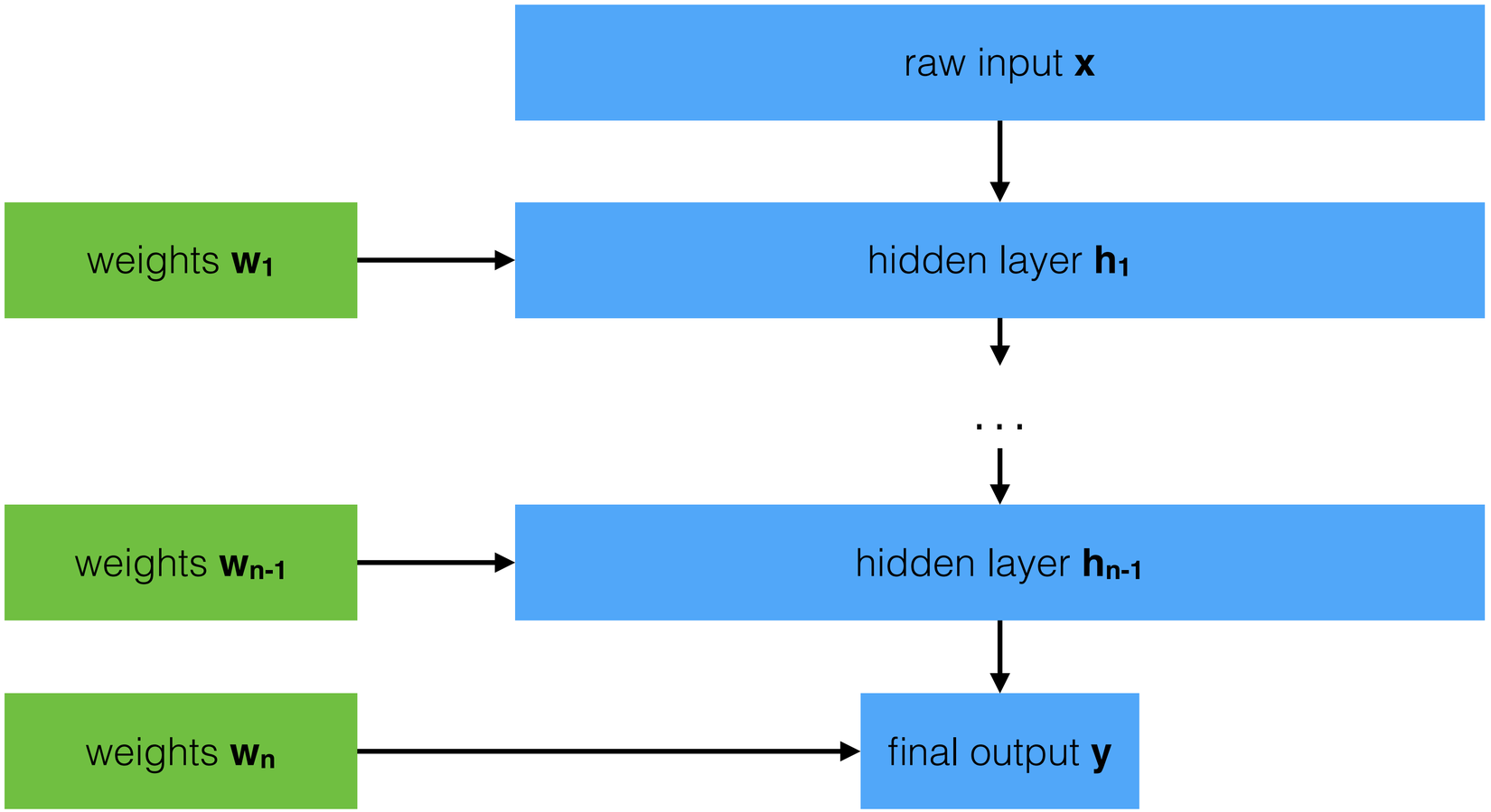


$$h_1 = f(x, w_1)$$

$$h_{n-1} = f(h_{n-2}, w_{n-1})$$

$$y = f(h_{n-1}, w_n)$$

Back to Neural Networks



$$h_1 = f(x, w_1)$$

$$h_{n-1} = f(h_{n-2}, w_{n-1})$$

$$y = f(h_{n-1}, w_n)$$

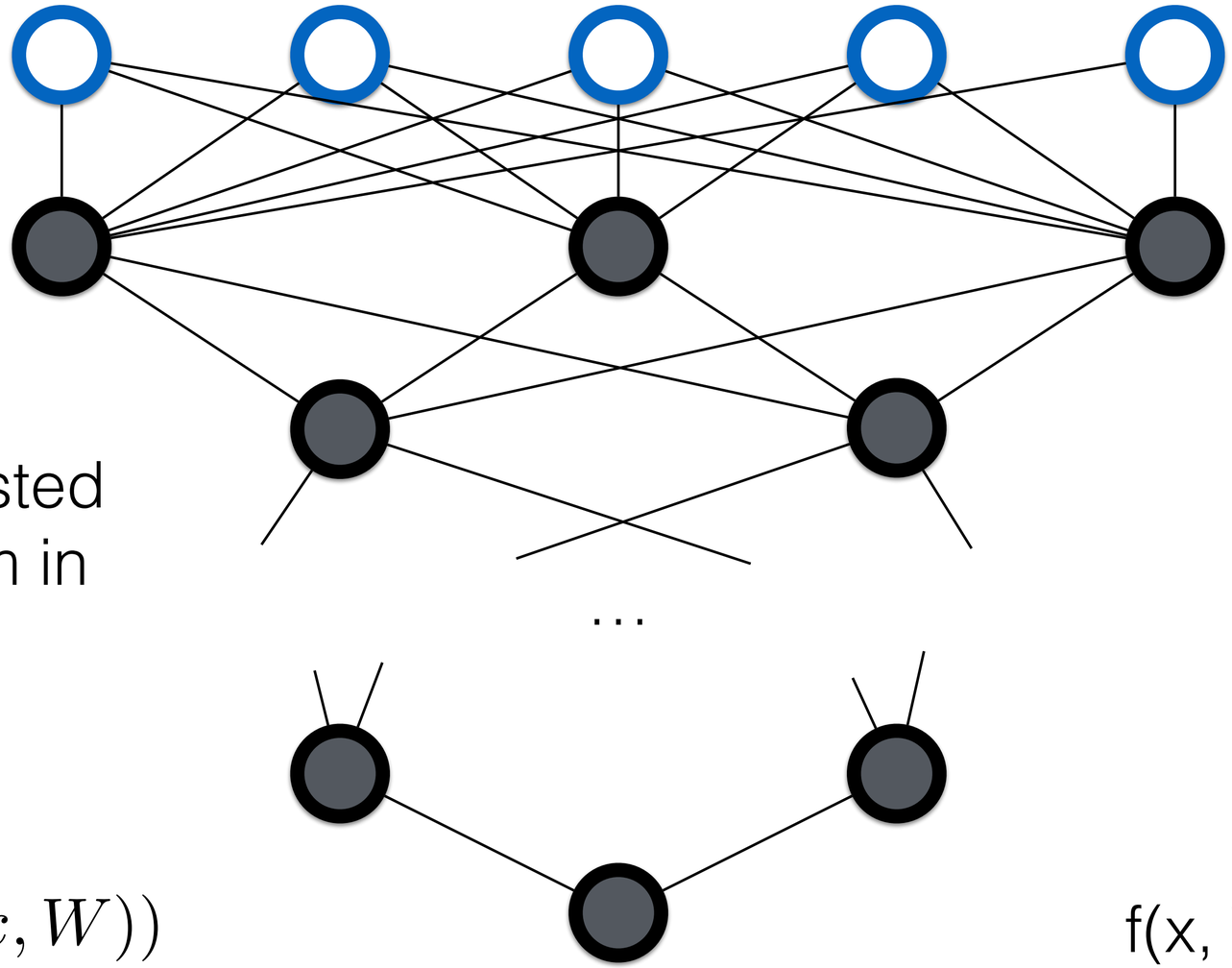
$$L(y)$$

$$\frac{d\,L}{d\,w_n} = \frac{d\,L}{d\,y} \cdot \frac{d\,y}{d\,w_n}$$

Intuition of Back Propagation

- At each layer, calculate how much changing the input changes the final output (derivative of final output w.r.t. layer's input)
  - Calculate directly for last layer
  - For preceding layers, use calculation from next layer and work backwards through network
- Use that derivative to find how changing the weights affect the error of the final output

FYI: Matrix Form



x

$h_1 = s(W_1 x)$

$h_2 = s(W_2 h_1)$

…

…

$h_{m-1} = s(W_{m-1} h_{m-2})$

$f(x, W) = s(W_m h_{m-1})$

(You will not be tested on this matrix form in this course.)

$$J(W) = \ell(f(x, W))$$

FYI: Matrix Gradient Recipe

$h_1 = s(W_1 x)$

$h_2 = s(W_2 h_1)$

$\dots$

$h_{m-1} = s(W_{m-1} h_{m-2})$

$f(x, W) = s(W_m h_{m-1})$

$$J(W) = \ell(f(x, W))$$

$$\delta_i = (W_{i+1}^\top \delta_{i+1}) \odot s'(W_i h_{i-1})$$

$$\delta_{m-1} = (W_m^\top \delta_m) \odot s'(W_{m-1} h_{m-2})$$

$$\delta_m = \ell'(f(x, W))$$

$$\nabla_{W_1} J = \delta_1 x^\top$$

$$\nabla_{W_i} J = \delta_i h_{i-1}^\top$$

$$\nabla_{W_{m-1}} J = \delta_{m-1} h_{m-2}^\top$$

$$\nabla_{W_m} J = \delta_m h_{m-1}^\top$$

(You will not be tested on this matrix form in this course.)

FYI: Matrix Gradient Recipe

(You will not be tested on this matrix form in this course.)

$h_1 = s(W_1 x)$

$h_i = s(W_i h_{i-1})$

$f(x, W) = s(W_m h_{m-1})$

$$J(W) = \ell(f(x, W))$$

Feed Forward Propagation

$$\delta_i = (W_{i+1}^\top \delta_{i+1}) \odot s'(W_i h_{i-1})$$

$$\delta_m = \ell'(f(x, W))$$

$$\nabla_{W_1} J = \delta_1 x^\top$$

$$\nabla_{W_i} J = \delta_i h_{i-1}^\top$$

Back Propagation

Other New Aspects of Deep Learning

- GPU computation
- Differentiable programming
- Automatic differentiation
- Neural network structures

Types of Neural Network Structures

- Feed-forward
- Recurrent neural networks (RNNs)
  - Good for analyzing sequences (text, time series)
- Convolutional neural networks (convnets, CNNs)
  - Good for analyzing spatial data (images, videos)

Outline

- Biological inspiration for artificial neural networks
- Linear vs. nonlinear functions
- Learning with neural networks: backpropagation