Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network
2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

MCS 472 Lecture 23
Industrial Math & Computation
Jan Verschelde, 6 March 2026

Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network
2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

What is Neuron?

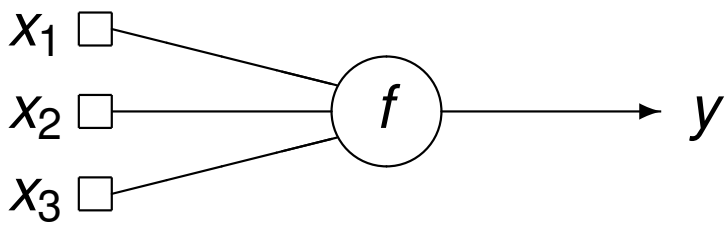
Definition (a neuron)

A *neuron* is a nonlinear bounded function

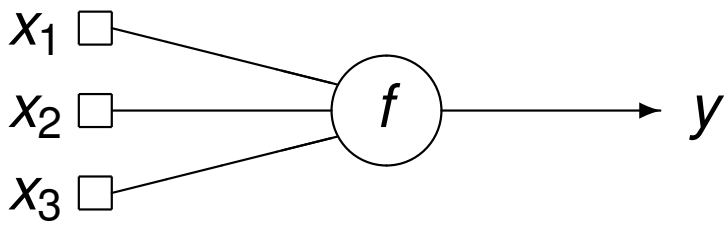
$$y = f(x_1, x_2, \ldots, x_n; w_1, w_2, \ldots, w_p)$$

where

- x_i are the variables, or inputs, $i = 1, 2, \ldots, n$, and
- w_j are the parameters, or weights, $j = 1, 2, \ldots, p$.



two examples



$$y = \tanh\left(\sum_{i=1}^{n} w_i x_i + w_{n+1}\right)$$

$$y = \exp\left(-\sum_{i=1}^{n}(x_i - w_i)^2 \Big/ (2w_{n+1}^2)\right)$$

a neural network

Definition (a neural network)

A *neural network* is the composition of the nonlinear functions of two or more neurons.

For example:

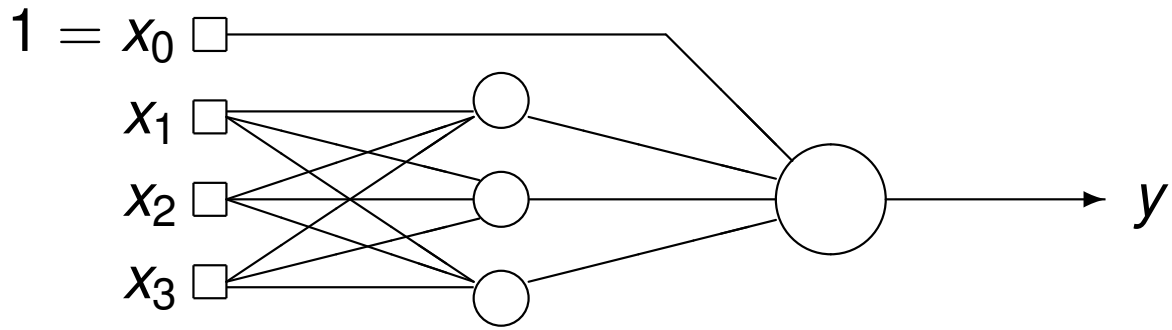
$$y = g(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N_c} \left(w_{N_c+1,i} \tanh \left(\sum_{j=1}^{n} w_{i,j} x_j + w_{i,n+1} \right) \right) + w_{N_c+1,n+1}$$

where

- $\mathbf{x}$ is the input vector, of n inputs, or variables, and
- $\mathbf{w}$ is the vector of $(n+1)N_c + N_c + 1$ parameters, or weights.

an example

The network



- has three inputs with one bias $x_0 = 1$,
- has three hidden neurons, and
- one linear output neuron.

Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network

2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

training a neural network

Definition (training a neural network)

The *training of a neural network* is the algorithmic procedure

- whereby the parameters of the neurons of the network are estimated,
- in order for the network to fulfill the tasks it is assigned to do.

Two categories of training are considered:

1. *supervised* training: we know the nonlinear function analytically, or numerical values of the function are known.
2. *unsupervised* training: no supervision.

two properties

1. Nonlinear in their parameters,
   neural networks are universal approximators.

Proposition (existence proof)

Any bounded, sufficiently regular function can be approximated uniformly, with arbitrary accuracy in a finite region of variable space, by a neural network with a single layer of hidden neurons, having the same activation function, and a linear output neuron.

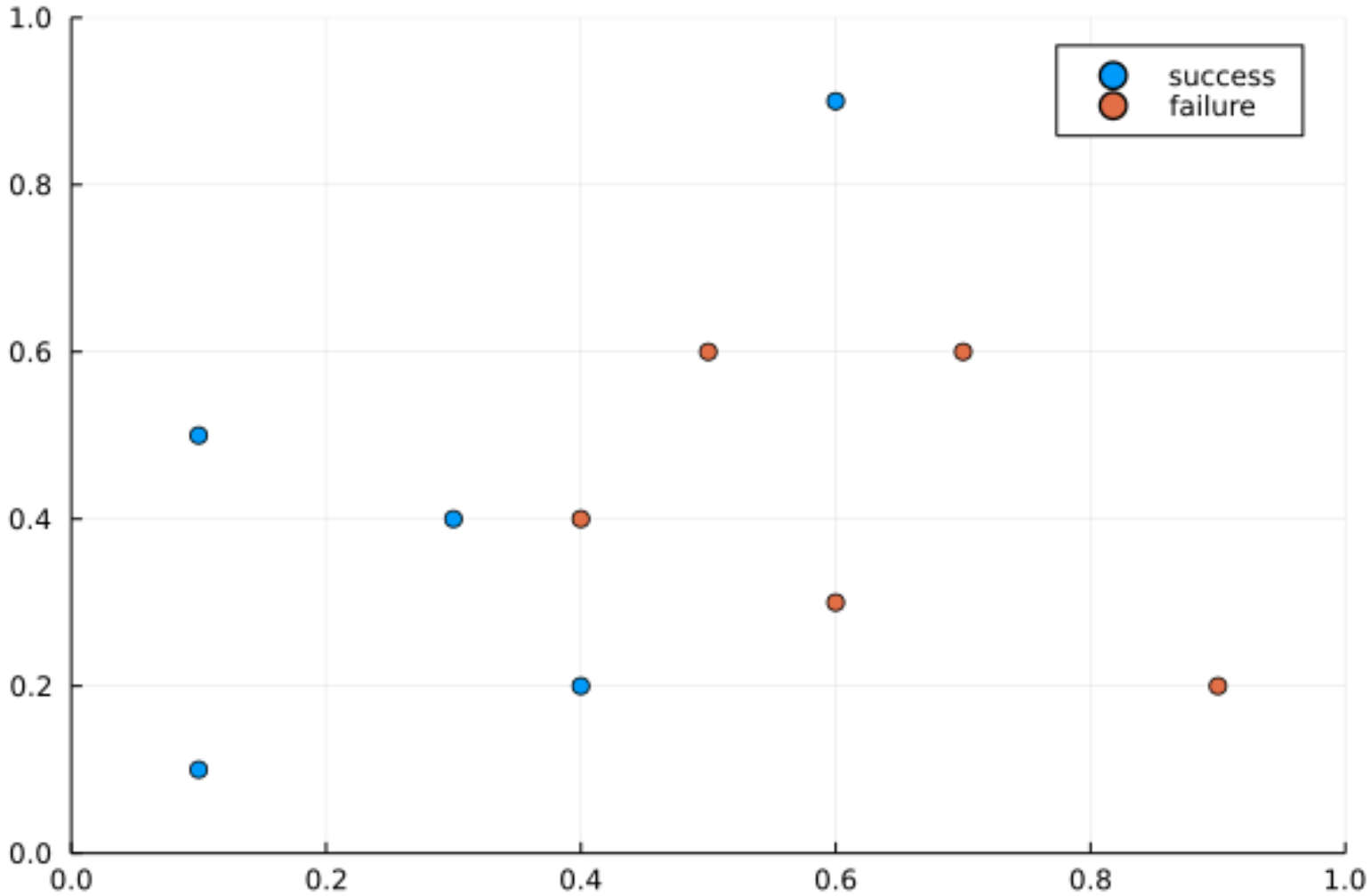
2. The most *parsimonious* model
   has the smallest number of parameters.

Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network

2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

a collection of labeled points



the points and the labels

The coordinates of the points:

```
x1 = [0.1,0.3,0.1,0.6,0.4,0.6,0.5,0.9,0.4,0.7]
x2 = [0.1,0.4,0.5,0.9,0.2,0.3,0.6,0.2,0.4,0.6]
```

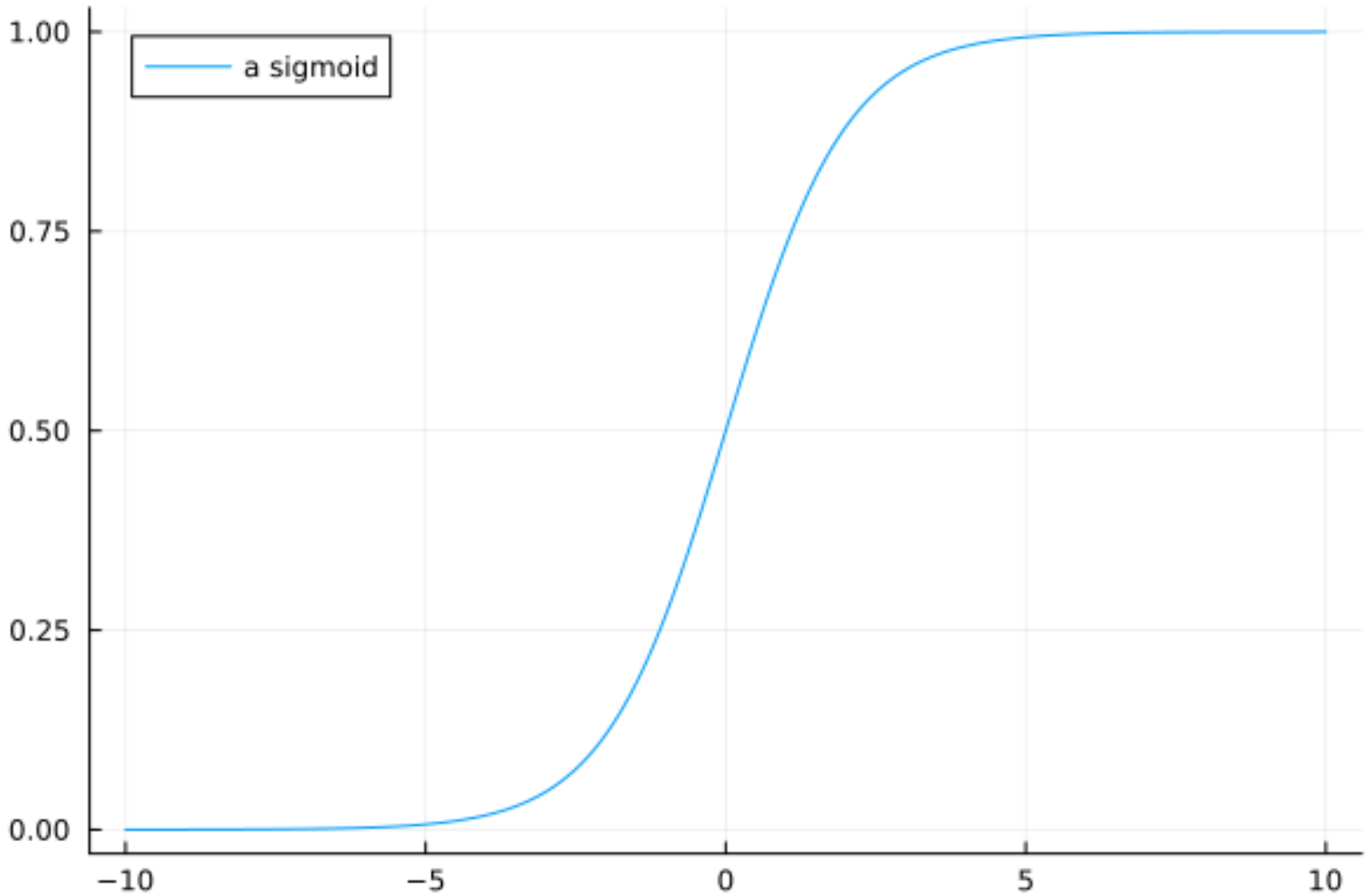
and their labels

```
y = [ones(1,5) zeros(1,5); zeros(1,5) ones(1,5)]
```

```
2×10 Matrix{Float64}:
  1.0  1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0
  0.0  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  1.0
```

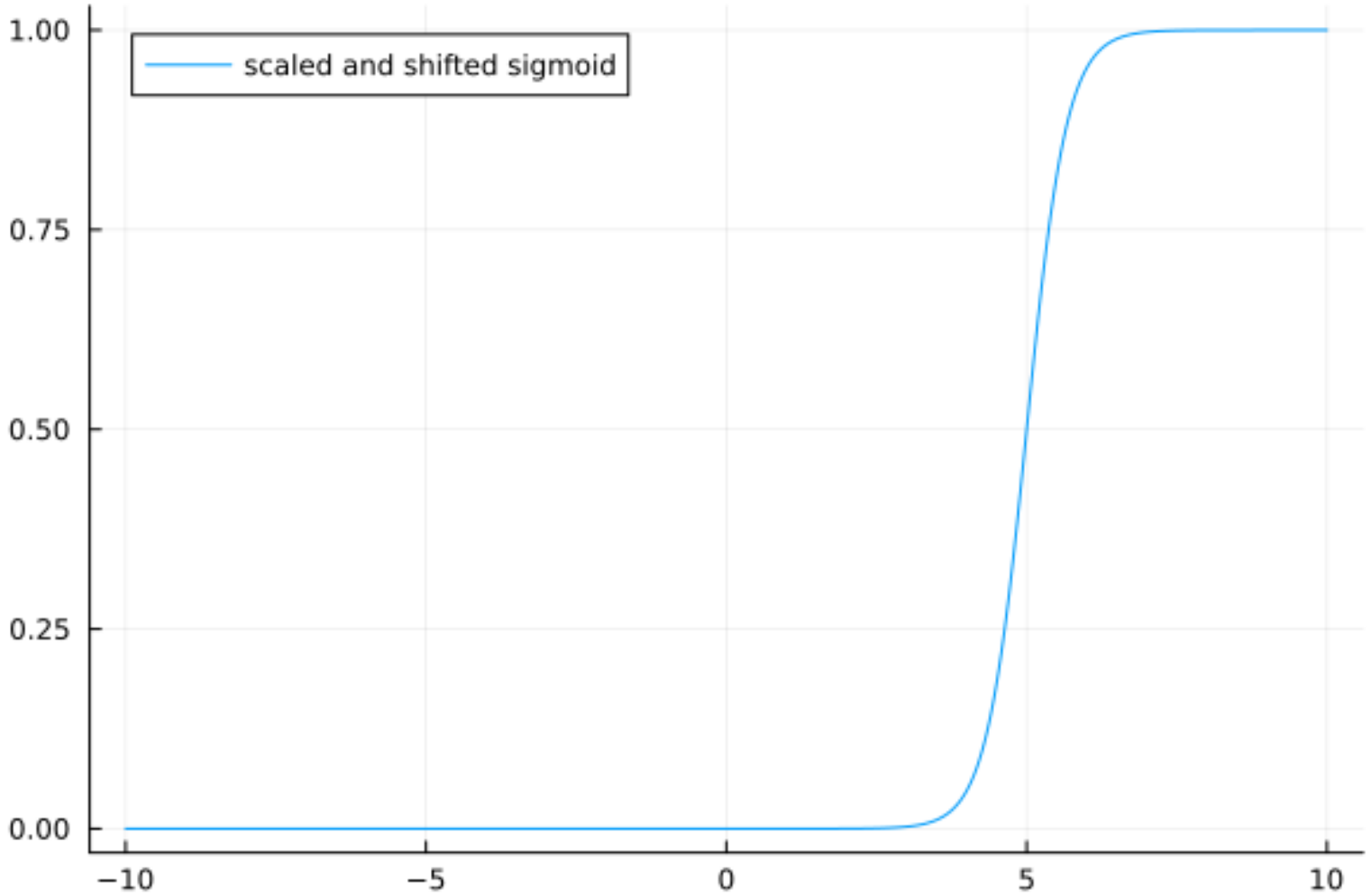
Problem: Find a function F, so $y = F(z_1, z_2)$, for a point (z_1, z_2).

the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$



The sigmoid function is a smoothed version of a step function.

a scaled and shifted sigmoid function $\sigma(3(x-5))$



a Julia function for $\sigma(Wx + b)$

```
"""
    function activate(x,W,b)

evaluates the sigmoid function at x,
with weight matrix W and bias vector b.
"""
function activate(x,W,b)
    dim = size(W,1)
    y = zeros(dim)
    argexp = -(W*x + b)
    for i=1:dim
        y[i] = 1.0/(1.0 + exp(argexp[i]))
    end
    return y
end
```

Dimensions: if $\mathtt{x} \in \mathbb{R}^n$, $\mathtt{W} \in \mathbb{R}^{m \times n}$, $\mathtt{b} \in \mathbb{R}^m$, then $\mathtt{y} \in \mathbb{R}^m$.

the neural network as a Julia function

The Julia function

```
function F(x, W2, W3, W4, b2, b3, b4)
    a2 = activate(x, W2, b2)
    a3 = activate(a2, W3, b3)
    a4 = activate(a3, W4, b4)
    return a4
end
```

computes

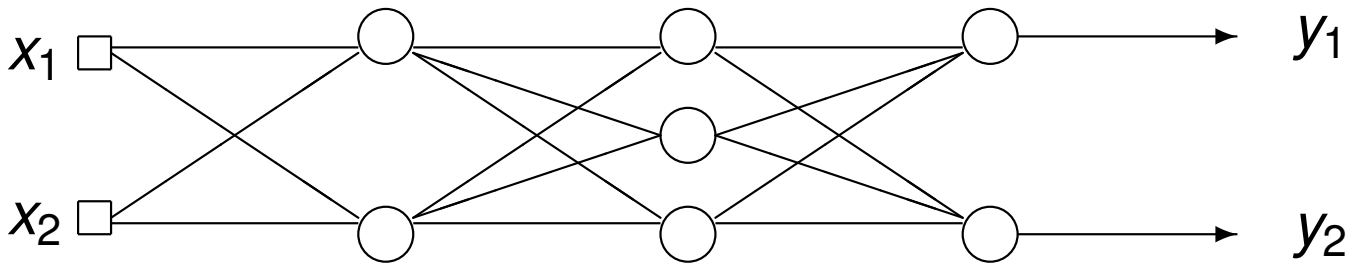
$$
\begin{aligned}
a_2 &= \sigma(\texttt{W2}*\texttt{x} + \texttt{b2}) \\
a_3 &= \sigma(\texttt{W3}*a_2 + \texttt{b3}) \\
a_4 &= \sigma(\texttt{W4}*a_3 + \texttt{b4})
\end{aligned}
$$

the layers in the neural network

What the Julia function computes

$$
\begin{aligned}
a_2 &= \sigma(\texttt{W2}*\texttt{x} + \texttt{b2}) \\
a_3 &= \sigma(\texttt{W3}*a_2 + \texttt{b3}) \\
a_4 &= \sigma(\texttt{W4}*a_3 + \texttt{b4})
\end{aligned}
$$

can be represented as



Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network

2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

the cost function

The weights and bias vectors in the neural network

$$\begin{aligned} a_2 &= \sigma(\texttt{W2}*\texttt{x} + \texttt{b2}) \\ a_3 &= \sigma(\texttt{W3}*a_2 + \texttt{b3}) \\ a_4 &= \sigma(\texttt{W4}*a_3 + \texttt{b4}) \end{aligned}$$

define the function

$$F(x) = \sigma(\texttt{W4}\ \sigma(\texttt{W3}\ \sigma(\texttt{W2}\ x + \texttt{b2}) + \texttt{b3}) + \texttt{b4})$$

which produces labels for each point. The cost function is

$$\text{Cost}\left(\texttt{W2},\texttt{W3},\texttt{W4},\texttt{b2},\texttt{b3},\texttt{b4}\right) = \frac{1}{10}\sum_{i=1}^{10}\frac{1}{2}\left\| y\left(x_1^{(i)}, x_2^{(i)}\right) - F\left(x_1^{(i)}, x_2^{(i)}\right)\right\|_2^2.$$

Exercise 1: How many parameters are in $F(x)$? Justify your answer.

Taylor series and the gradient vector

Assume the current vector is $p = (p_1, p_2, \ldots, p_N)$.

Consider the Taylor series of the Cost function at p:

$$\text{Cost}(p + \Delta p) = \text{Cost}(p) + \sum_{i=1}^{N} \left(\frac{\partial \text{Cost}}{\partial p_i}(p) \right) \Delta p_i + O(\|\Delta p\|^2).$$

Denoting the gradient vector

$$\nabla \text{Cost}(p) = \left(\frac{\partial \text{Cost}}{\partial p_1}(p), \frac{\partial \text{Cost}}{\partial p_2}(p), \ldots, \frac{\partial \text{Cost}}{\partial p_N}(p) \right)$$

and ignoring the $O(\|\Delta p\|^2)$ term:

$$\text{Cost}(p + \Delta p) \approx \text{Cost}(p) + \left(\nabla \text{Cost}(p) \right)^T \Delta p.$$

steepest descent

With Taylor series we derived

$$\text{Cost}(p + \Delta p) \approx \text{Cost}(p) + \Big(\nabla \text{Cost}(p)\Big)^T \Delta p.$$

Minimizing the Cost function is the objective.

In an iterative method we replace p by $p + \Delta p$,

choosing Δp that makes $\Big(\nabla \text{Cost}(p)\Big)^T \Delta p$ as negative as possible.

Cauchy-Schwartz: $\left| \Big(\nabla \text{Cost}(p)\Big)^T \Delta p \right| \leq \Big\| \nabla \text{Cost}(p) \Big\|_2 \, \|\Delta p\|_2$.

Therefore, choose Δp in the direction of $-\nabla \text{Cost}(p)$.

the stochastic gradient method

Update p as $p = p - \eta \nabla \text{Cost}(p)$,

where the step size η is called the *learning rate*.

As we have a large number of parameters and many training points, the computation of the gradient vector at each step is too costly.

Instead, take one single, *randomly chosen* training point, and evaluate the gradient at that point.

This gives rise to the *stochastic gradient method*.

Exercise 2: Examine the computational cost to evaluate $\nabla \text{Cost}(p)$ of our $F(x) = \sigma(\texttt{W4}\ \sigma(\texttt{W3}\ \sigma(\texttt{W2}\ x + \texttt{b2}) + \texttt{b3}) + \texttt{b4})$.

Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network

2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

evaluating the derivatives

Let $a^{(1)} = x$, then the network returns $a^{(L)}$, where

$$\begin{aligned} z^{(\ell)} &= W^{(\ell)} a^{(\ell-1)} + b^{(\ell)} \\ a^{(\ell)} &= \sigma\left(z^{(\ell)}\right), \quad \text{for } \ell = 2, 3, \ldots, L. \end{aligned}$$

Let C be the cost, define $\delta_j^{(\ell)} = \dfrac{\partial C}{\partial z_j^{(\ell)}}$ as the error of neuron j at layer ℓ.

By the chain rule, we have (with $\circ$ as the componentwise product):

$$\begin{aligned} \delta^{(L)} &= \sigma'(z^{(L)}) \circ (a^{(L)} - y) \\ \delta^{(\ell)} &= \sigma'(z^{(\ell)}) \circ (W^{(\ell+1)})^T \delta^{(\ell+1)} \\ \frac{\partial C}{\partial b_j^{(\ell)}} &= \delta_j^{(\ell)} \quad \text{and} \quad \frac{\partial C}{\partial w_{j,k}^{(\ell)}} = \delta_j^{(\ell)} a^{(\ell-1)}. \end{aligned}$$

Sigmoids have convenient derivatives: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

back propagation

The formulas on the previous slide lead to an algorithm:

1. The forward pass evaluates

$$a^{(1)}, z^{(2)}, a^{(2)}, z^{(3)}, \ldots, a^{(L)}.$$

2. The backward pass evaluates

$$\delta^{(L)}, \delta^{(L-1)}, \ldots, \delta^{(2)}.$$

This way of computing gradients is *back propagation*.

Introduction to Neural Networks

1. Definitions
   - what is a neural network?
   - training a neural network

2. An Artificial Neural Network
   - labeled points and sigmoids
   - the stochastic gradient method
   - evaluating the derivatives
   - training the network

Julia code to train the network

```
eta = 0.05
Niter = 1000000
for counter = 1:Niter
    k = rand((1:10), 1)[1]
    x = [x1[k], x2[k]]
    # Forward pass
    a2 = activate(x,W2,b2)
    a3 = activate(a2,W3,b3)
    a4 = activate(a3,W4,b4)
    # Backward pass
    delta4 = a4.*(ones(length(a4))-a4).*(a4-y[:,k])
    delta3 = a3.*(ones(length(a3))-a3).*(W4'*delta4)
    delta2 = a2.*(ones(length(a2))-a2).*(W3'*delta3)
    # Gradient step
    W2 = W2 - eta*delta2*x'
    W3 = W3 - eta*delta3*a2'
    W4 = W4 - eta*delta4*a3'
    b2 = b2 - eta*delta2
    b3 = b3 - eta*delta3
    b4 = b4 - eta*delta4
end
```

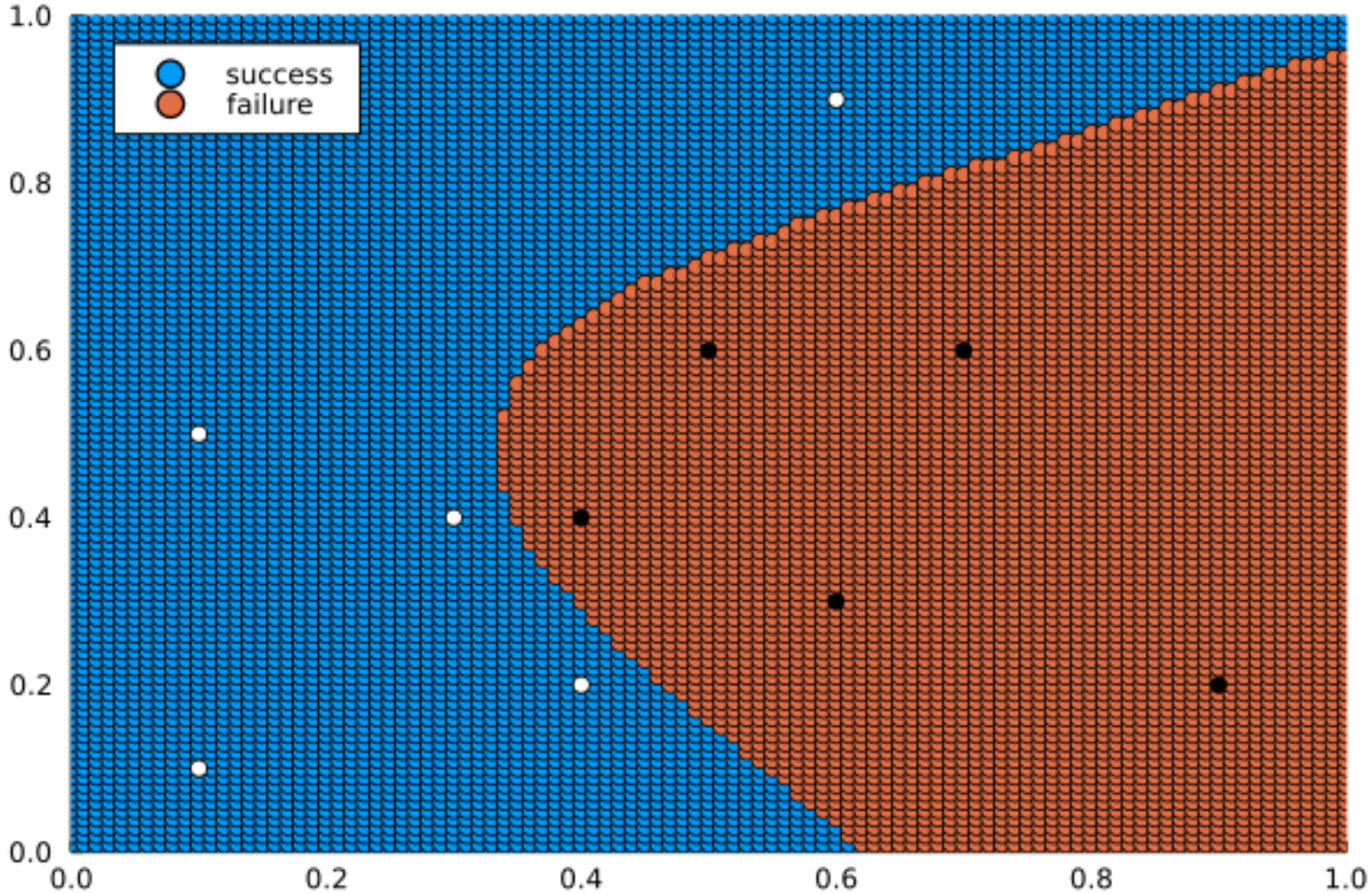
the result

```
Z = [[x1[k], x2[k]] for k=1:10]
Y = [F(z, W2, W3, W4, b2, b3, b4) for z in Z]
```

The output:

```
10-element Vector{Vector{Float64}}:
 [0.9914991960849379, 0.008456961045118156]
 [0.9938902272050594, 0.006075750247468293]
 [0.998360435186392, 0.0016281342781392 95]
 [0.9865419031464729, 0.013387592738288 21]
 [0.9737963089611226, 0.026155307111289 86]
 [0.013028283269915059, 0.9869736384774328]
 [0.016354675508034607, 0.9837066212812329]
 [0.0005627461156236613, 0.9994383300176262]
 [0.021069103097048 85, 0.979014892476464]
 [0.0012977070814389772, 0.9987047112752311]
```

evaluating $F(x_1, x_2)$ over $[0, 1] \times [0, 1]$



The white and black dots are the ten given points.

plot the evolution of the cost function

How fast did the stochastic gradient method converge?

Exercise 3:

1. Write a function to evaluate Cost(p).
2. Call the function in the code to train the network, at each step, store the value of the cost function.
3. Make a plot of the cost over all steps in the code.

summary and references

The definitions of neurons and neural networks are illustrated taking a data fitting view of artificial neural networks.

The definitions in this lecture are in Chapter 1 of the book:

- Gérard Dreyfus:
  Neural Networks. Methodology and Applications.
  Springer-Verlag 2005.

The introduction of deep learning was taken from:

- Catherine F. Higham and Desmond J. Higham:
  Deep Learning: An Introduction for Applied Mathematicians.
  SIAM Review, Vol. 61, No. 4, pages 860–891, 2019.