CSCI 497P/597P: Computer Vision

Scott Wehrwein

Neural Networks, Backpropagation

Reading

- http://cs231n.github.io/optimization-2/
- http://cs231n.github.io/neural-networks-1/

Announcements

Goals

- Understand backpropagation as an application of the chain rule to find the gradient of a classifier's loss with respect to its parameters
- Understand neural networks as a stack of linear classifiers with nonlinearities (activation functions) in between.
- Understand the basic menu of activations (Sigmoid, Tanh, ReLU)
  - Understand the vanishing gradients problem.

Gradient Descent

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

Gradient Descent

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

whence the evaluate_gradient function?

Calculating the Gradient

- Suppose for a moment that everything is a scalar:
  - $L_i(x_i, y_i\, w, b) = \max(0, 1 - y_i(w\, x_i + b))$
  - (whiteboard / lecture notes)

Neural Networks: The Brain Stuff

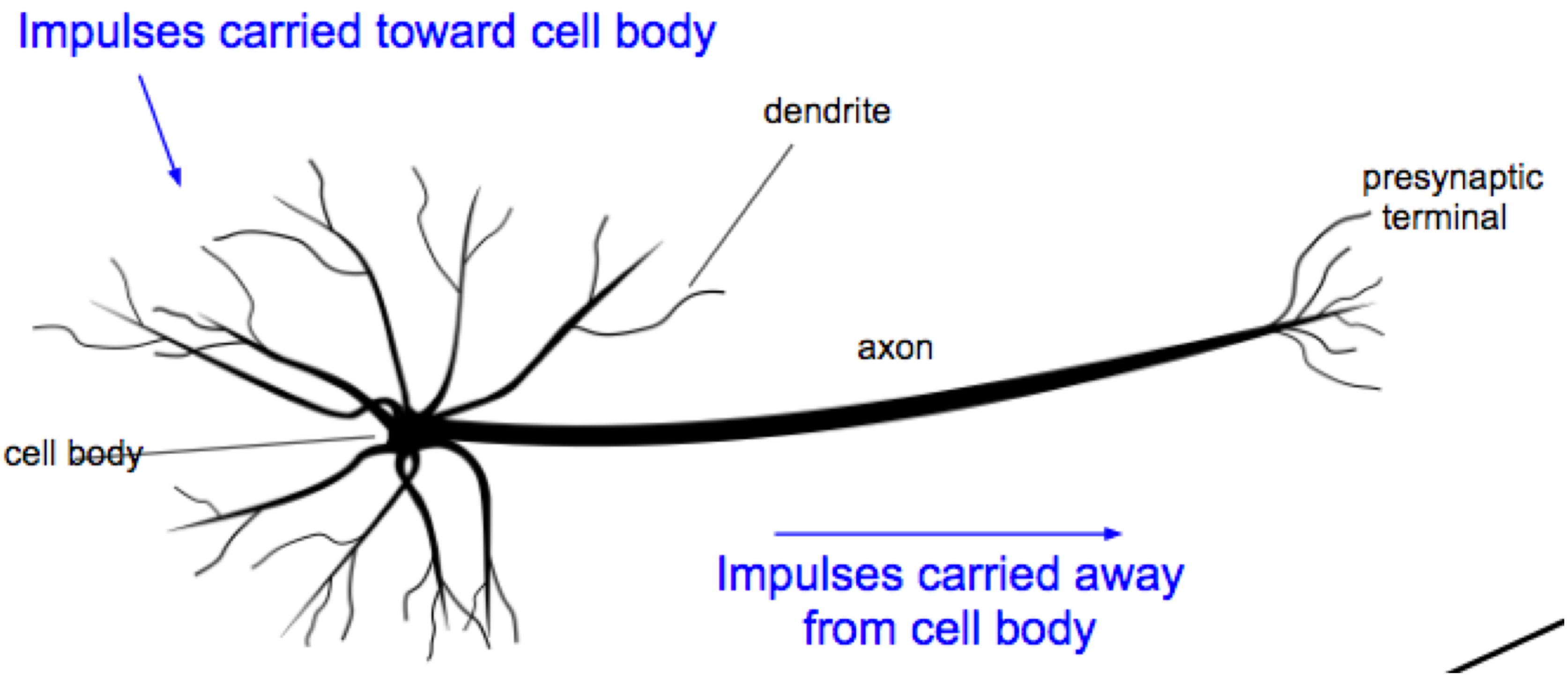


image by Felipe Peruchois licensed under CC-BY 3.0

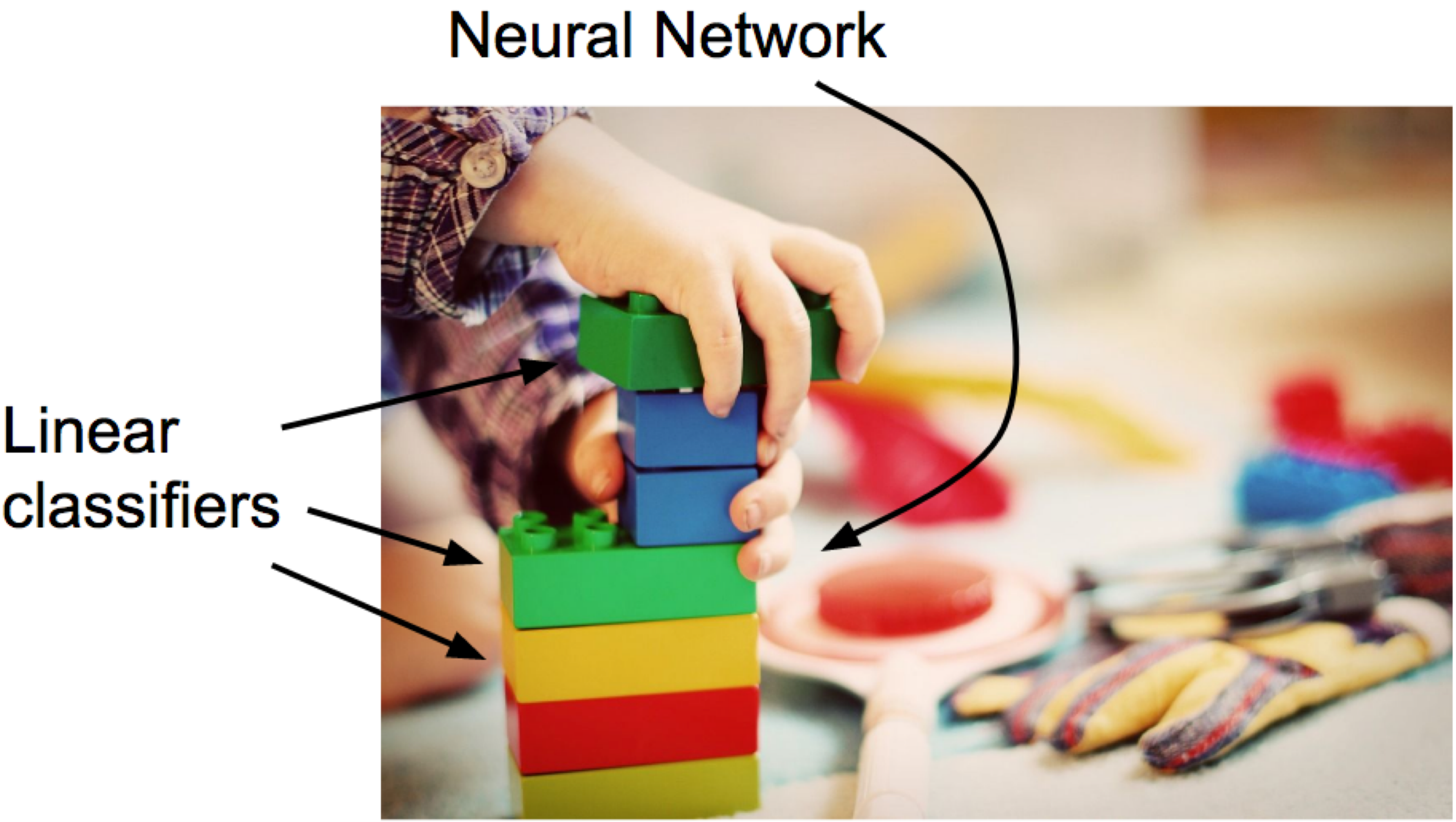
Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

Neural Networks



Neural networks: without the brain stuff

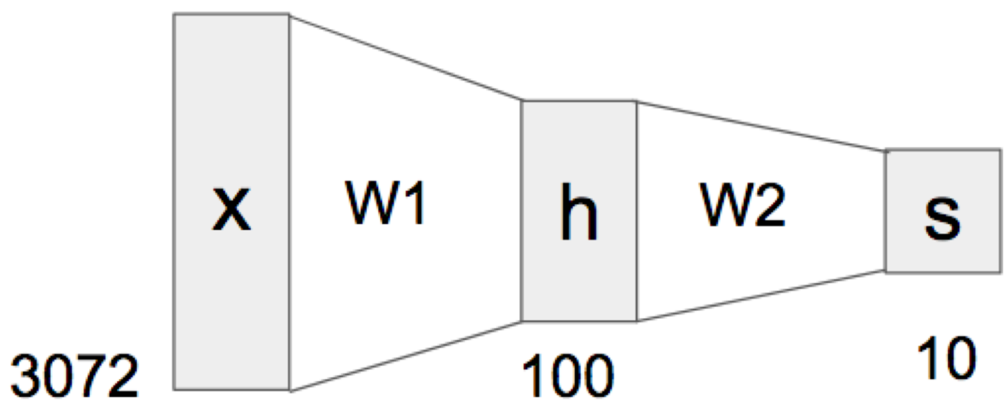
(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$



Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

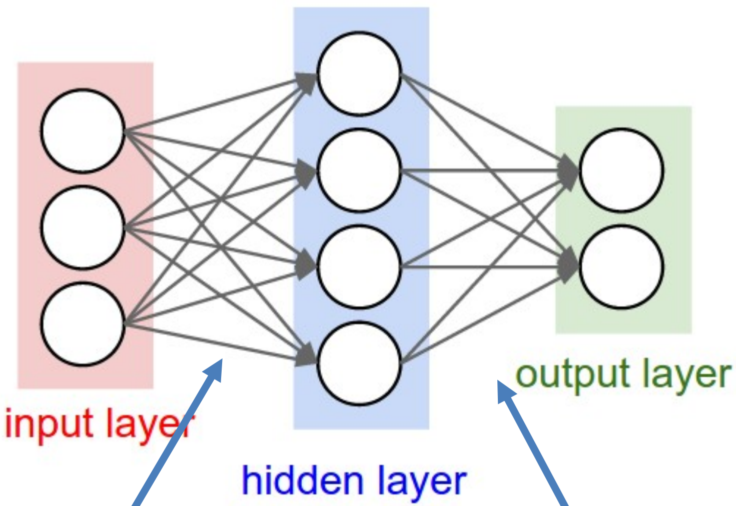
Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

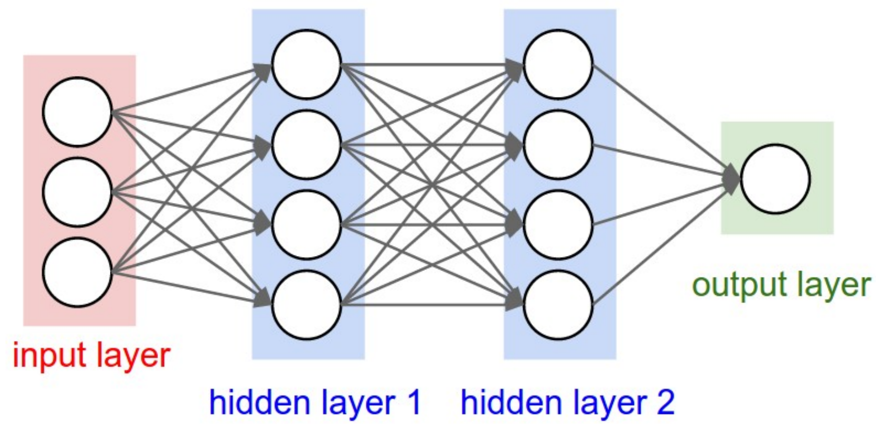
Training a 2 layer neural network in 20 lines of python

```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

“Hidden Layers”



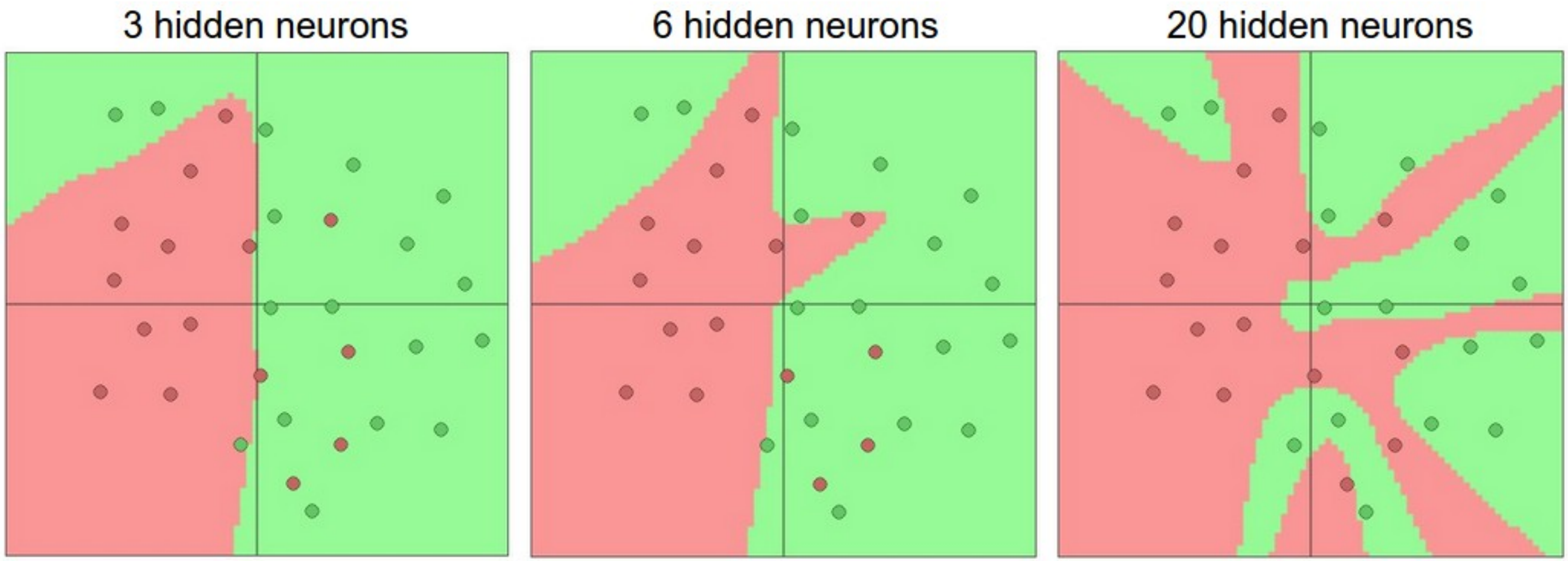


W_1, a 3x4 matrix converts input into hidden layer activations

W_2, a 4x2 matrix transforms hidden layer activations to output scores

Figures: Fei-Fei Li, Justin Johnson, & Serena Yeung

Neural Networks: Nonlinear Classifiers built from Linear Classifiers



Figures: Fei-Fei Li, Justin Johnson, & Serena Yeung

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$
or 3-layer Neural Network

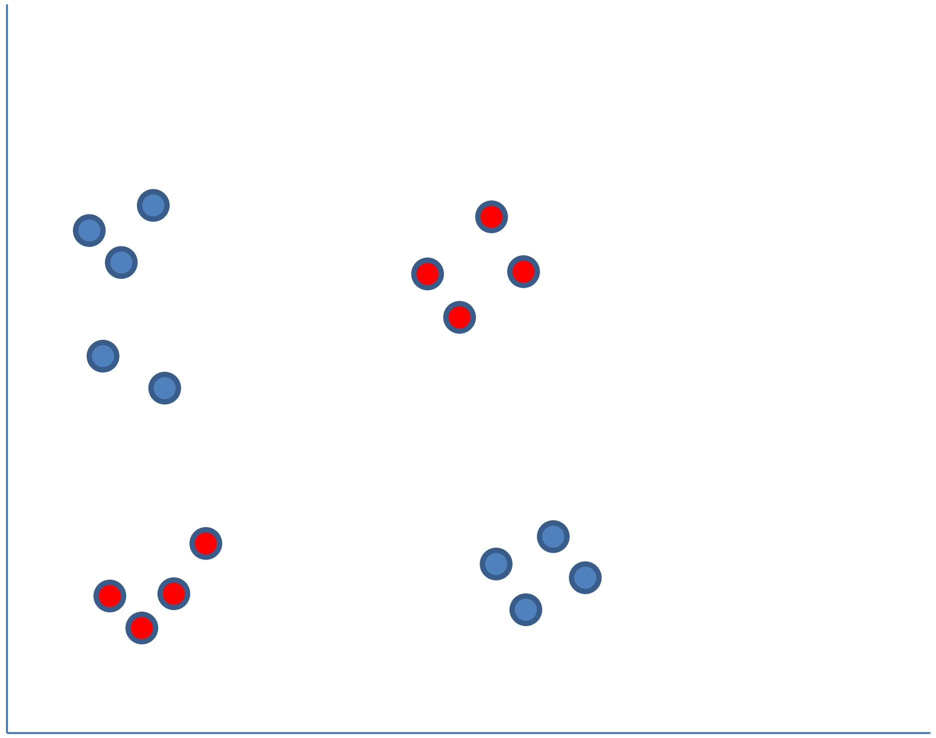
$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

???

Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung

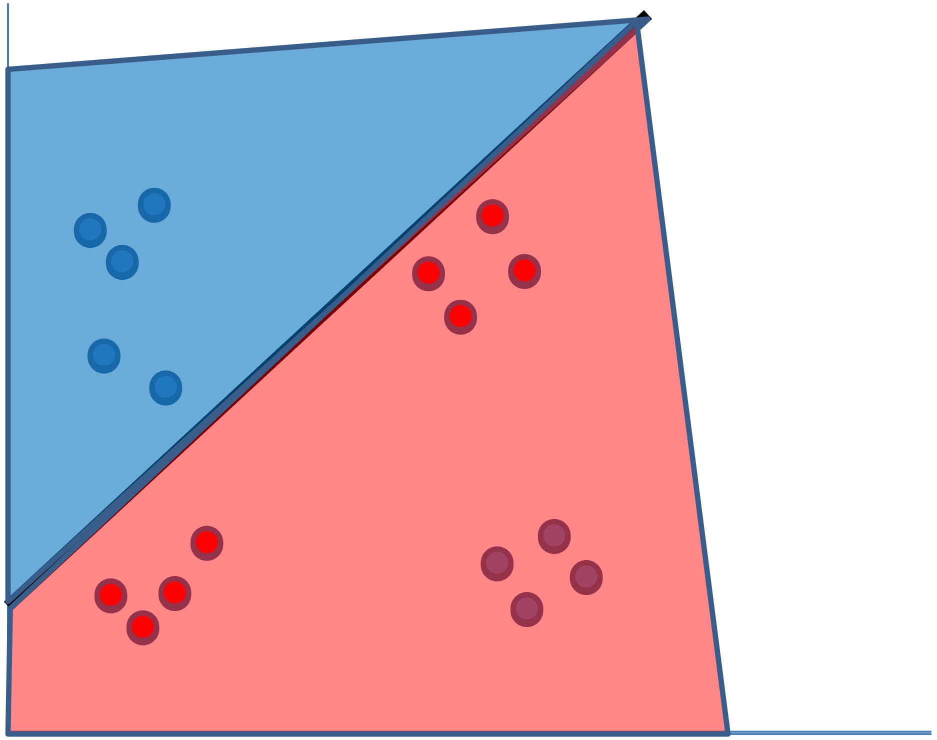
Activation Functions

$$f(x, W) = Wx$$



Activation Functions

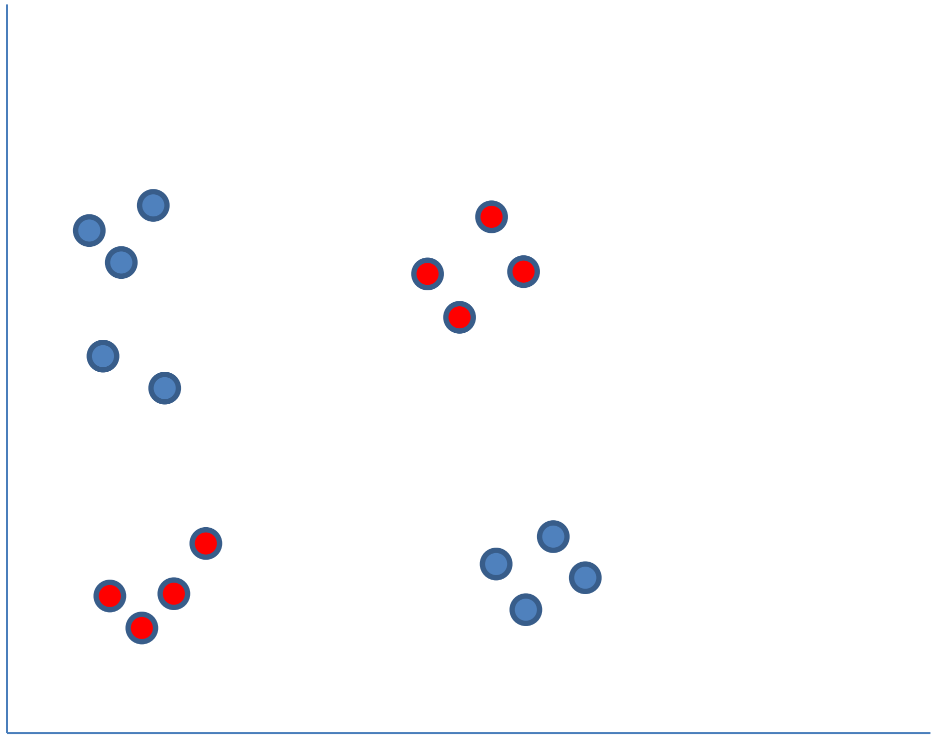
$$f(x, W) = Wx$$



Activation Functions

$$f(x, W) = Wx$$

$$f(x, W_1, W_2) = W_1(W_2 x)$$



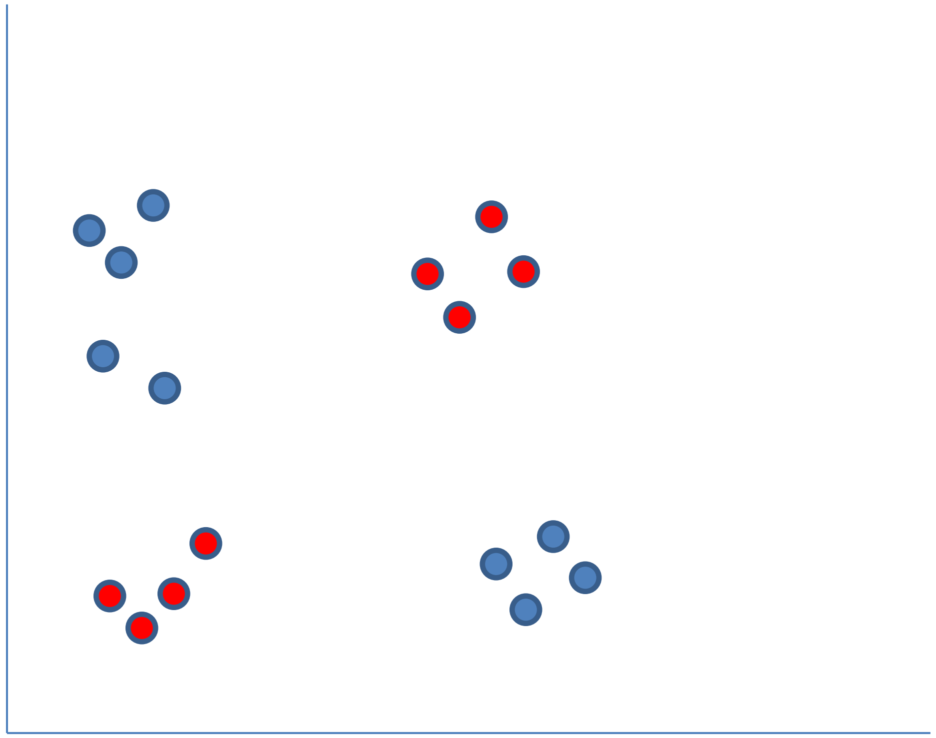
Activation Functions

$$f(x, W) = Wx$$

$$f(x, W_1, W_2) = W_1(W_2x)$$

$$W \leftarrow W_1W_2$$

$$f(x, W) = Wx$$



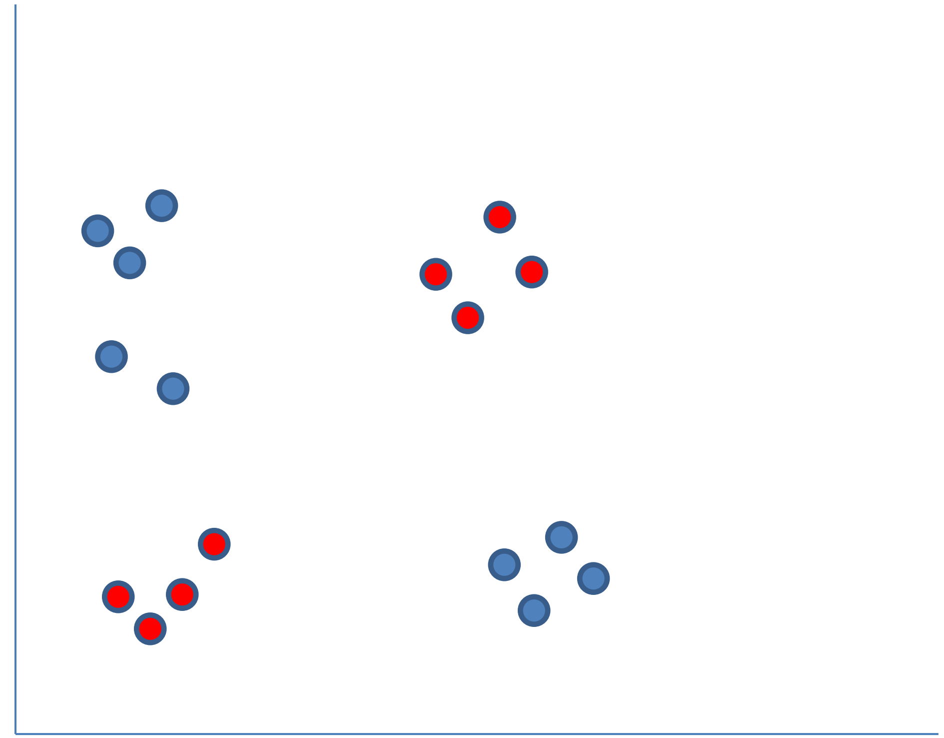
Activation Functions

$$f(x, W) = Wx$$

$$f(x, W_1, W_2) = W_1(W_2x)$$

$$W \leftarrow W_1W_2$$

$$f(x, W) = Wx$$



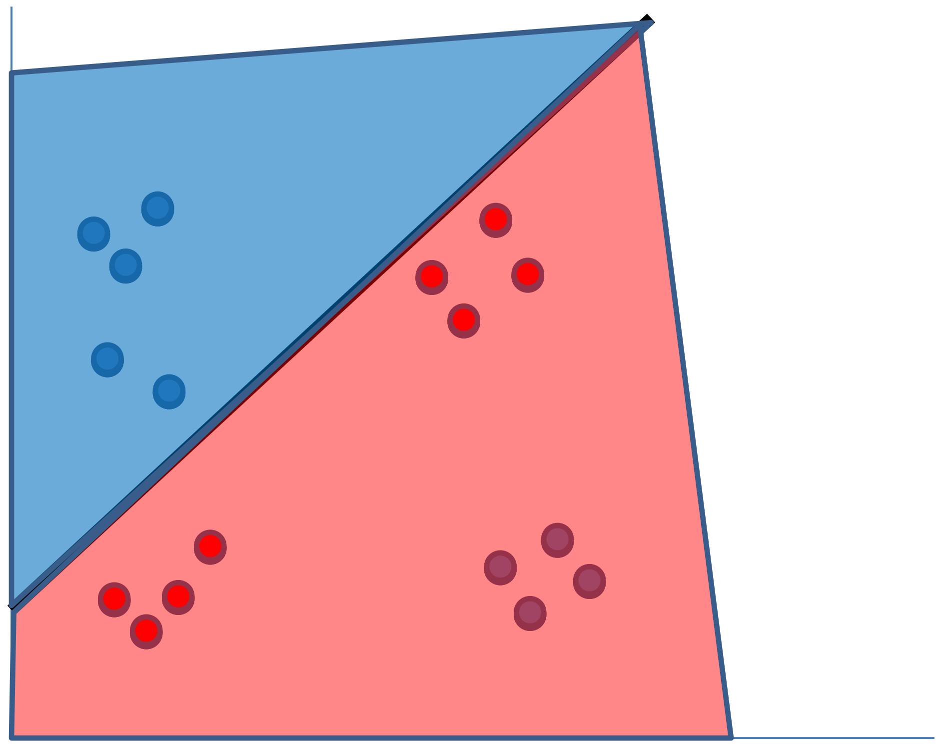
Activation Functions

$$f(x, W) = Wx$$

$$f(x, W_1, W_2) = W_1(W_2 x)$$

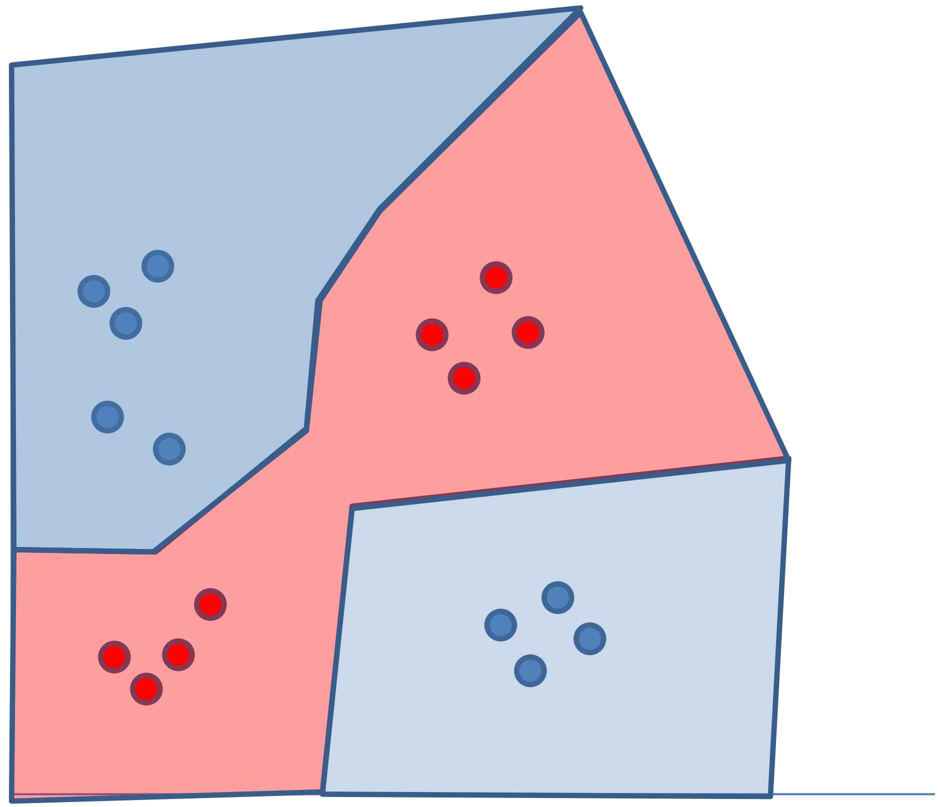
$$W \leftarrow W_1 W_2$$

$$f(x, W) = Wx$$

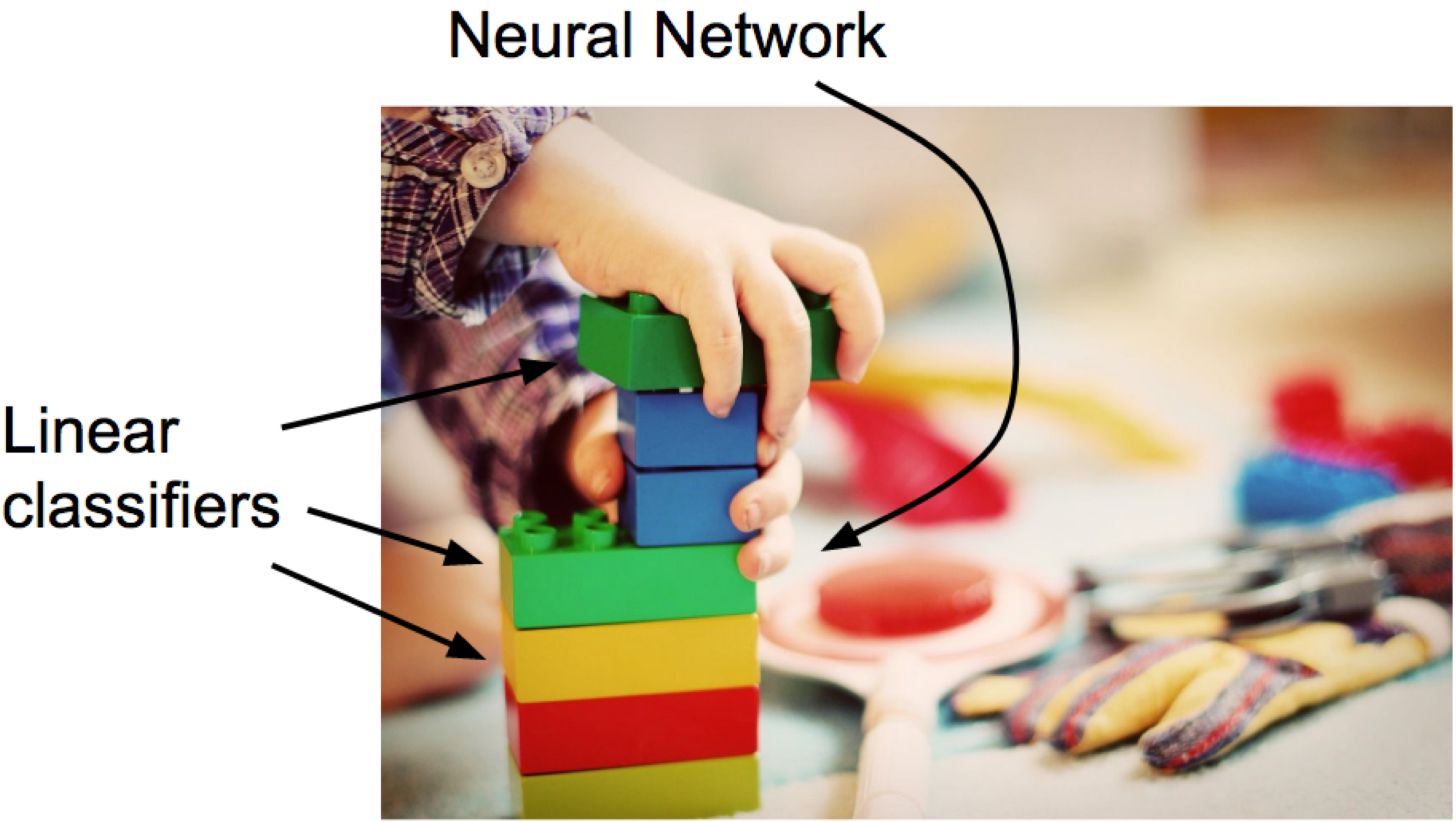


Activation Functions

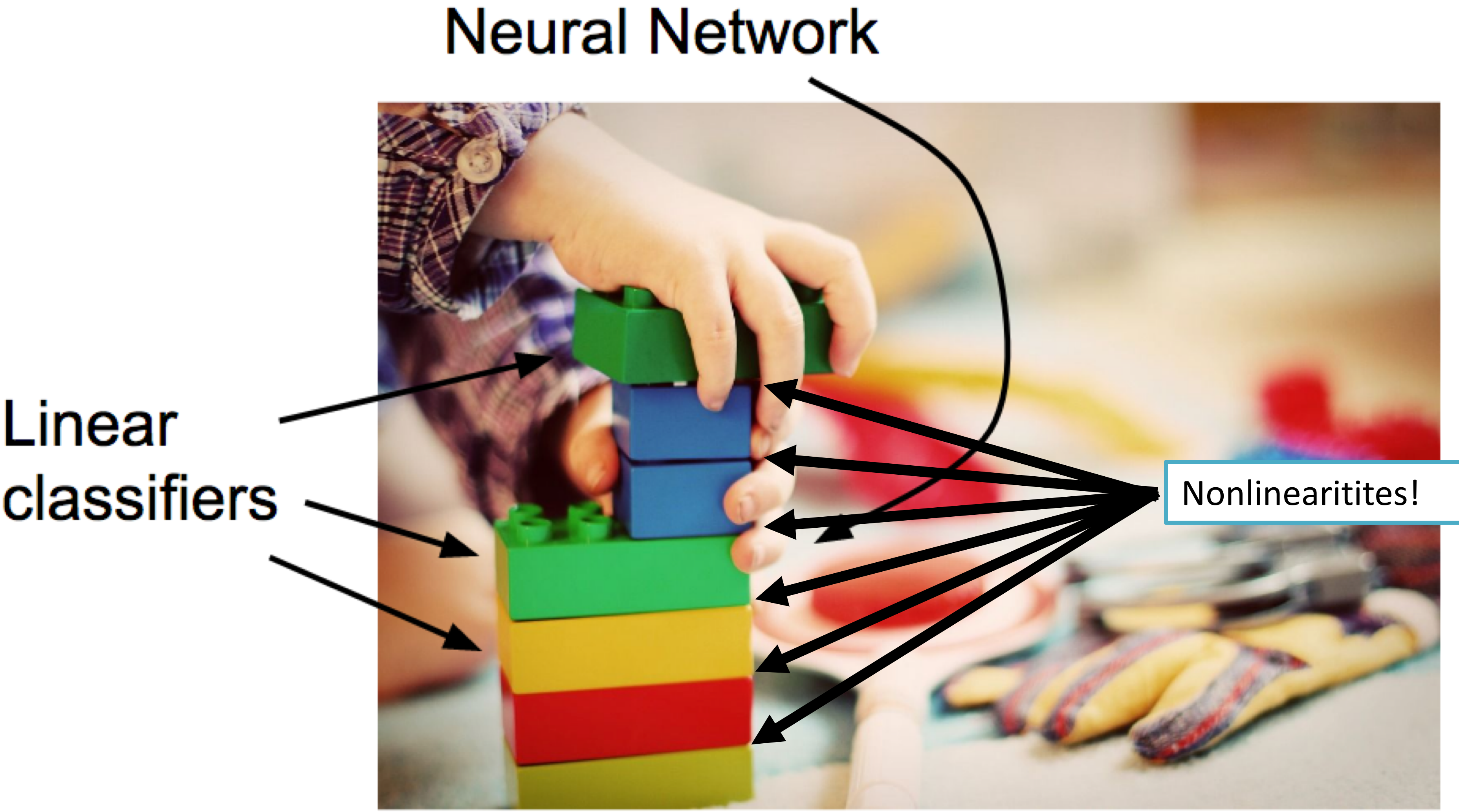
$$f(x, W_1, W_2, W_3) = W_3 \max(0, W_2 \max(0, W_1 x))$$



Neural Networks



Neural Networks



Neural Networks: Nonlinear Classifiers built from Linear Classifiers

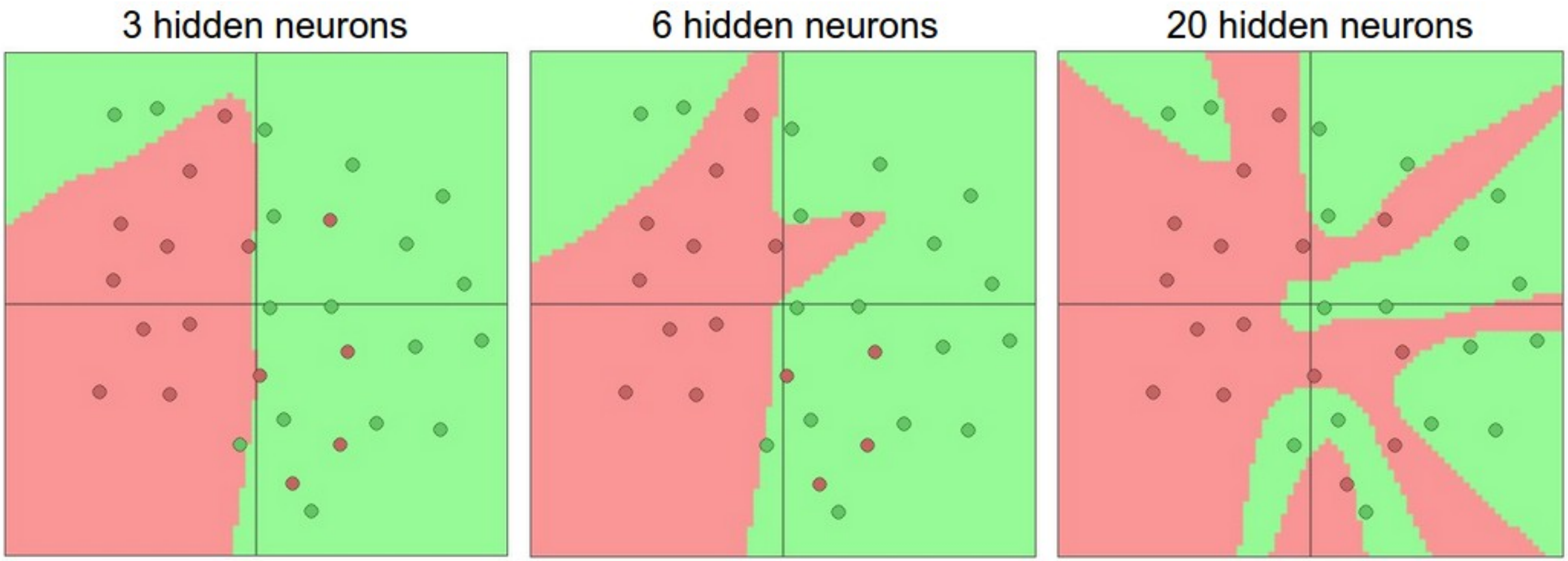
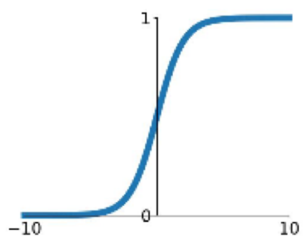


Figure: Fei-Fei Li, Justin Johnson, & Serena Yeung

Activation Functions

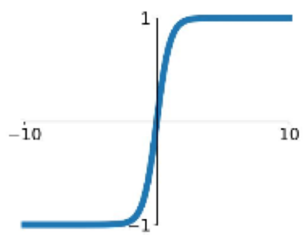
Sigmoid

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



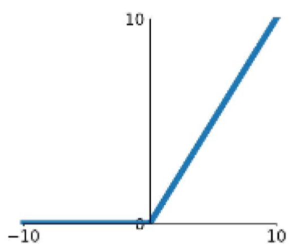
tanh

$\tanh(x)$



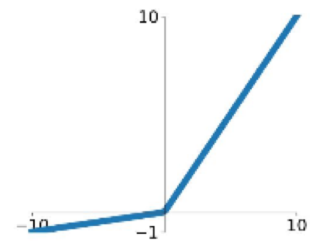
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

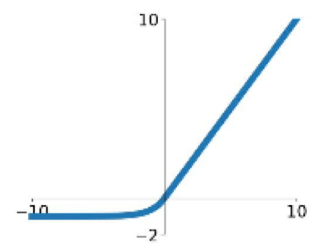


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Slide: Fei-Fei Li, Justin Johnson, & Serena Yeung