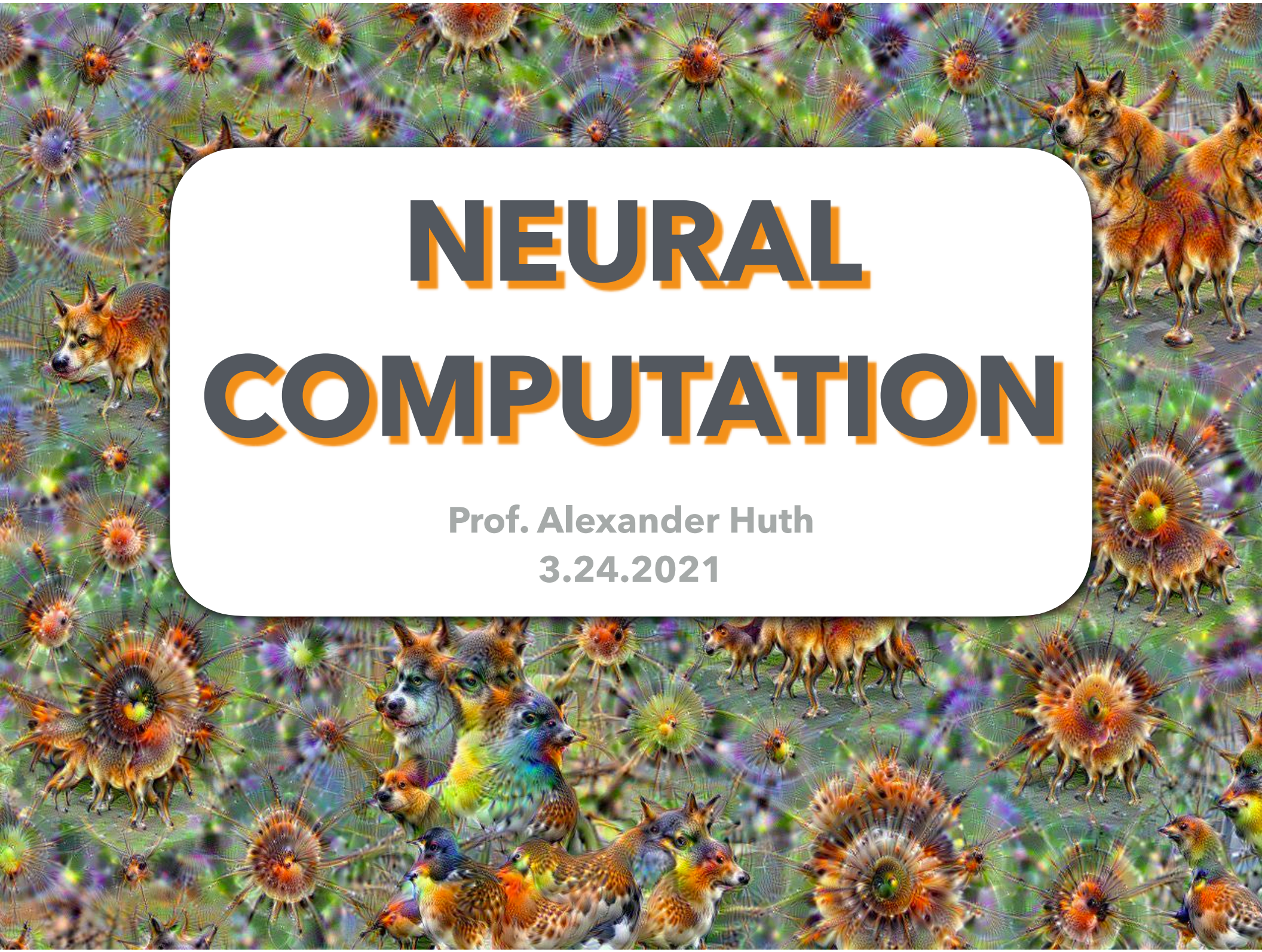

NEURAL
COMPUTATION
Prof. Alexander Huth
3.24.2021

COURSE ADMIN / HW

* Homework 1 is posted today, and will be due on **April 7**

RECAP:
NONLINEAR METHODS

* **Volterra series**

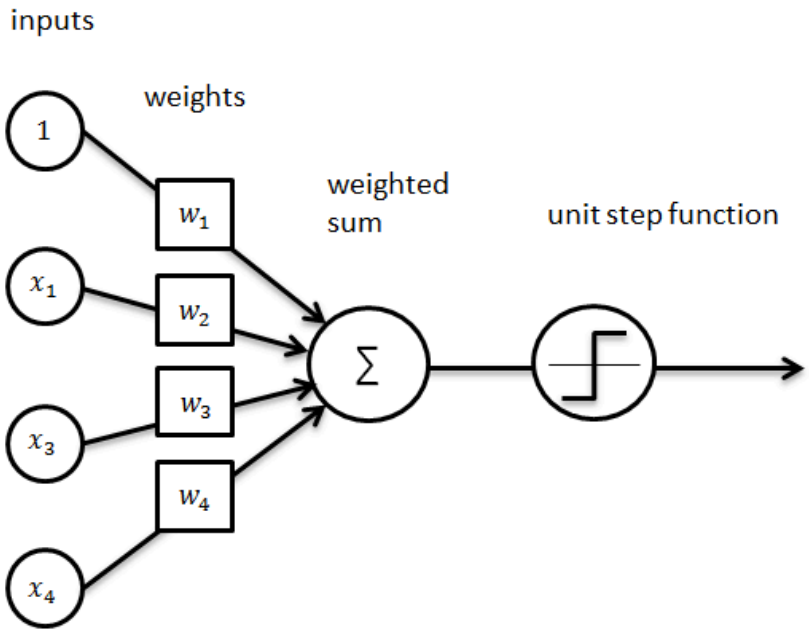
* **Kernel regression** *(samples, not features!)*

TODAY

* **Artificial neural networks**
  * Perceptrons
  * Hidden layer
  * Backpropagation

ARTIFICIAL NEURAL NETWORK

* Simplest version: a *perceptron*



$$y_i \in \{0, 1\}$$

$$X_i \in \mathcal{R}^p$$

$$\hat{y}_i = H(X_i w)$$

h/t http://ataspinar.com/2016/12/22/the-perceptron/

PERCEPTRON LEARNING

* Due to Frank Rosenblatt (1958)

* Usually defined as online learning rule (assuming each example will be fed in one-by-one)

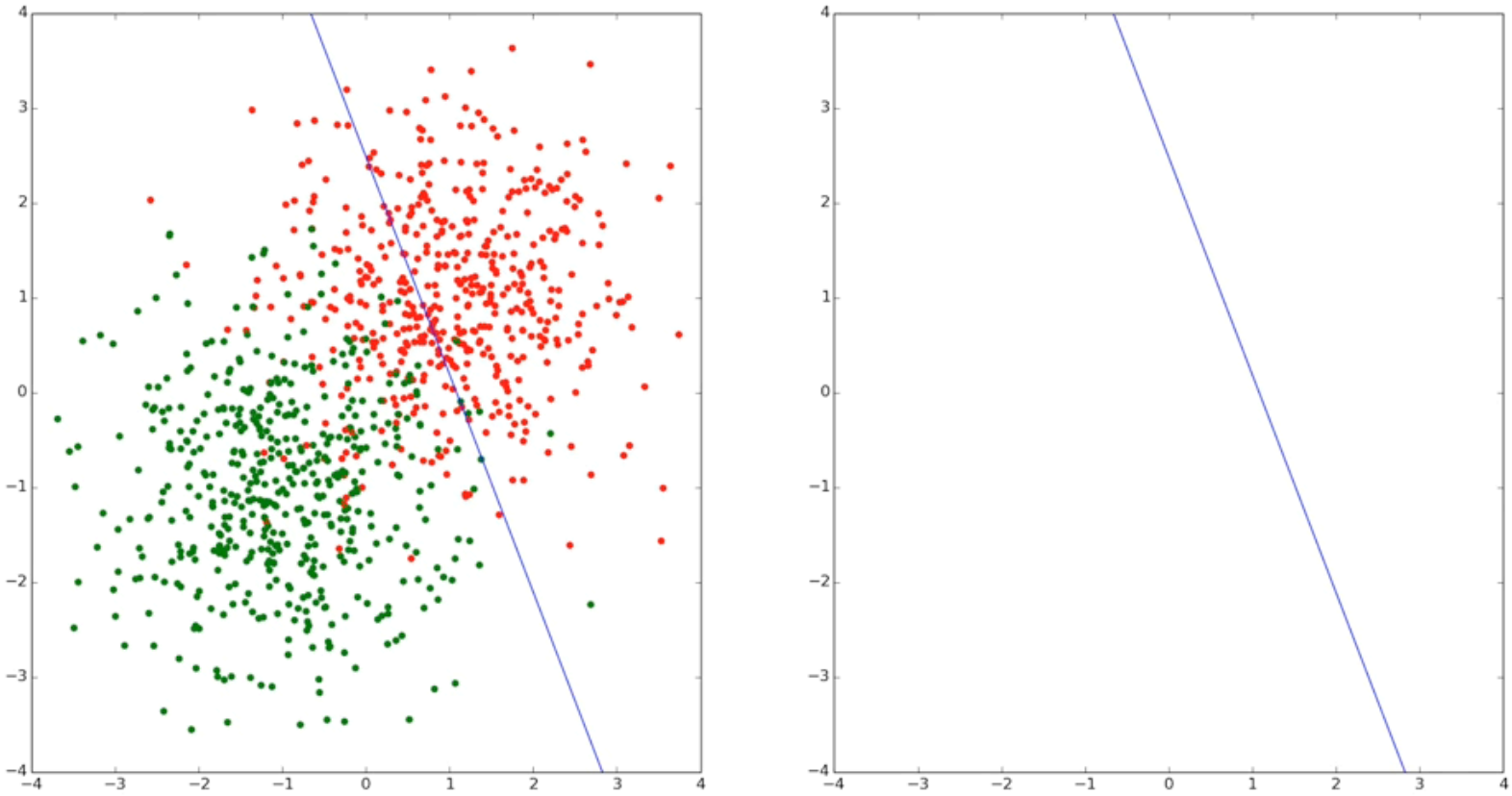
PERCEPTRON LEARNING

At iteration t in the learning process:

$$\hat{y}_i^{(t)} = H(X_i w^{(t)})$$

$$w_j^{(t+1)} = w_j^{(t)} + (y_i - \hat{y}_i^{(t)}) X_{ij}$$

PERCEPTRON LEARNING



h/t https://github.com/ayusek/Perceptron-Animation

PERCEPTRONS

* “You dumdums, single perceptrons can’t even solve something as simple as XOR, how could they possibly recognize images or walk or talk or be conscious?!”
  – Minsky & Papert, basically, 1969

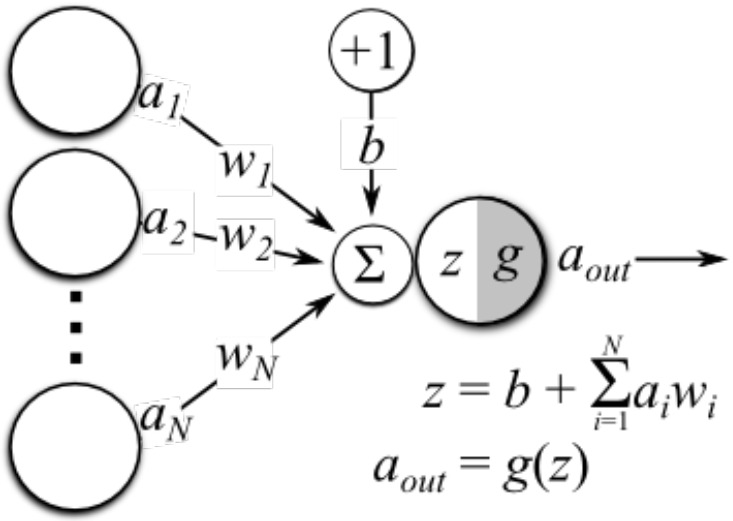
* –> “AI winter” of 1974-1980

ARTIFICIAL NEURAL NETWORKS

* Generic 1-layer neural "network"

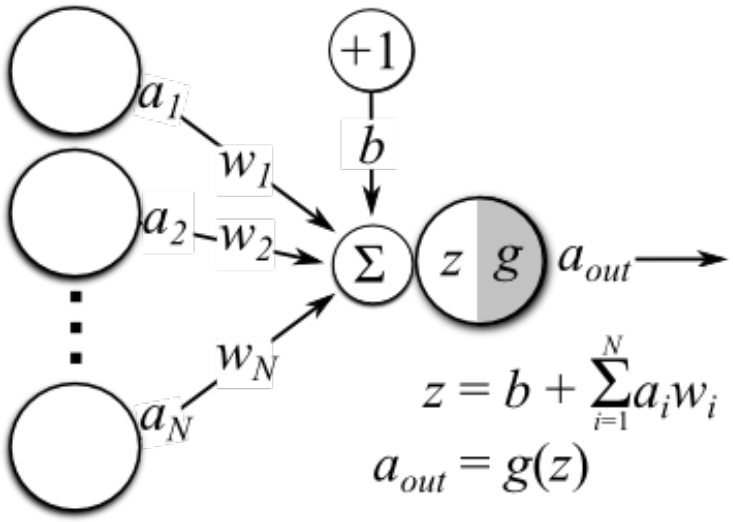
* g(z) is the output nonlinearity

* Update rule? Just differentiate the loss function! (As long as g(z) is differentiable)

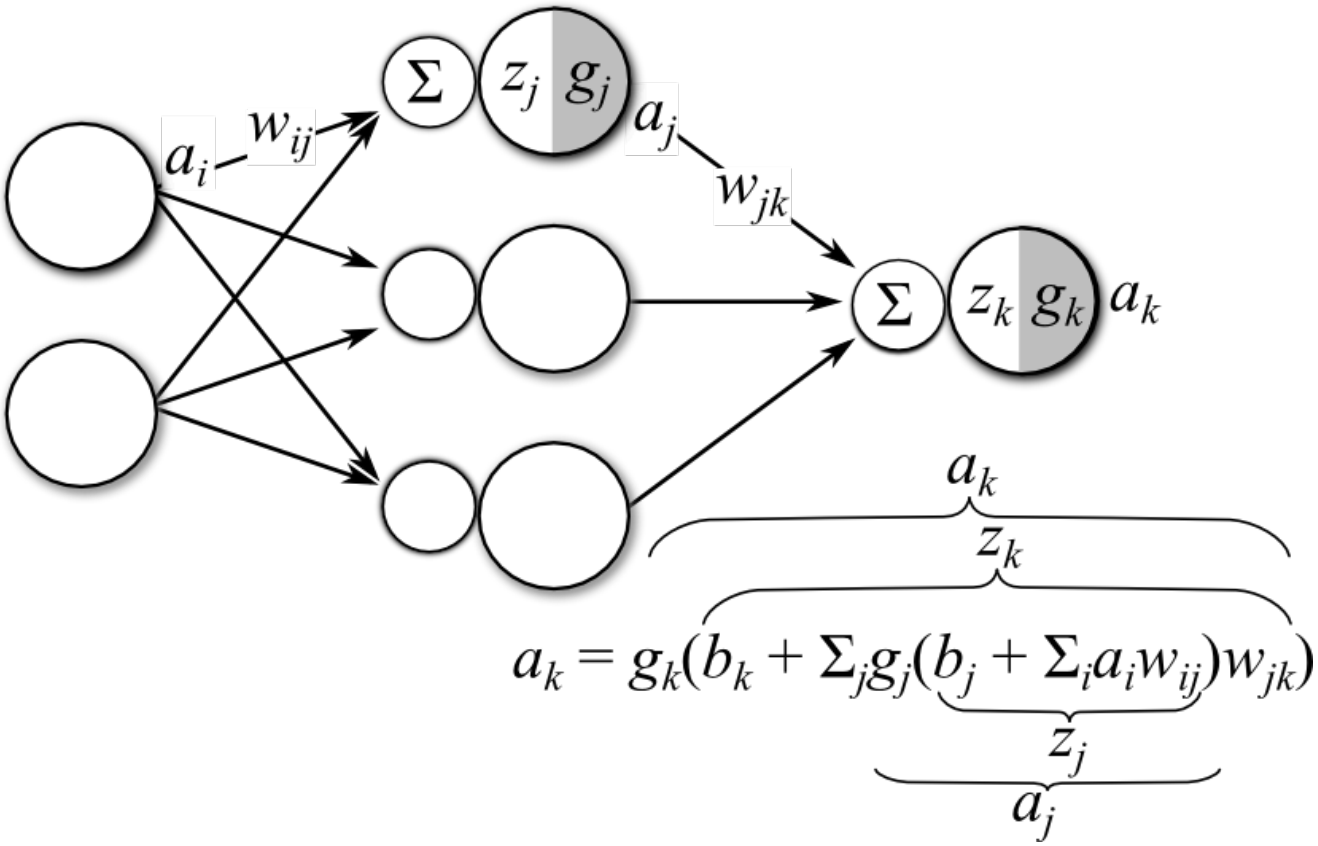


ARTIFICIAL NEURAL NETWORKS

* Problem: this network is not very “expressive”
  * It can’t XOR
* Solution: stack it! add a “hidden layer”



ARTIFICIAL NEURAL NETWORKS



w = weights

a = activations

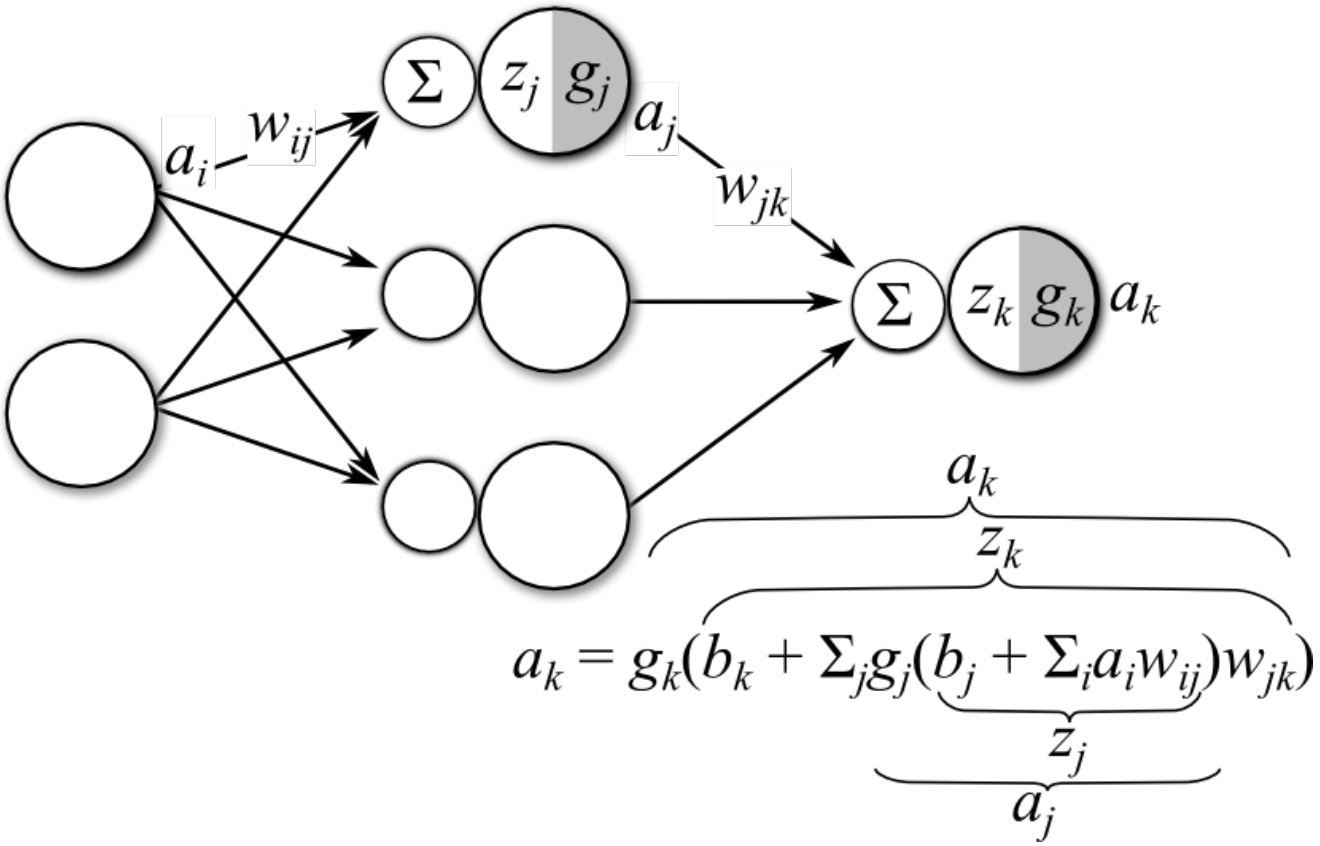
z = total input

g = activ. fxn

b = bias

h/t Dustin Stansbury

ARTIFICIAL NEURAL NETWORKS



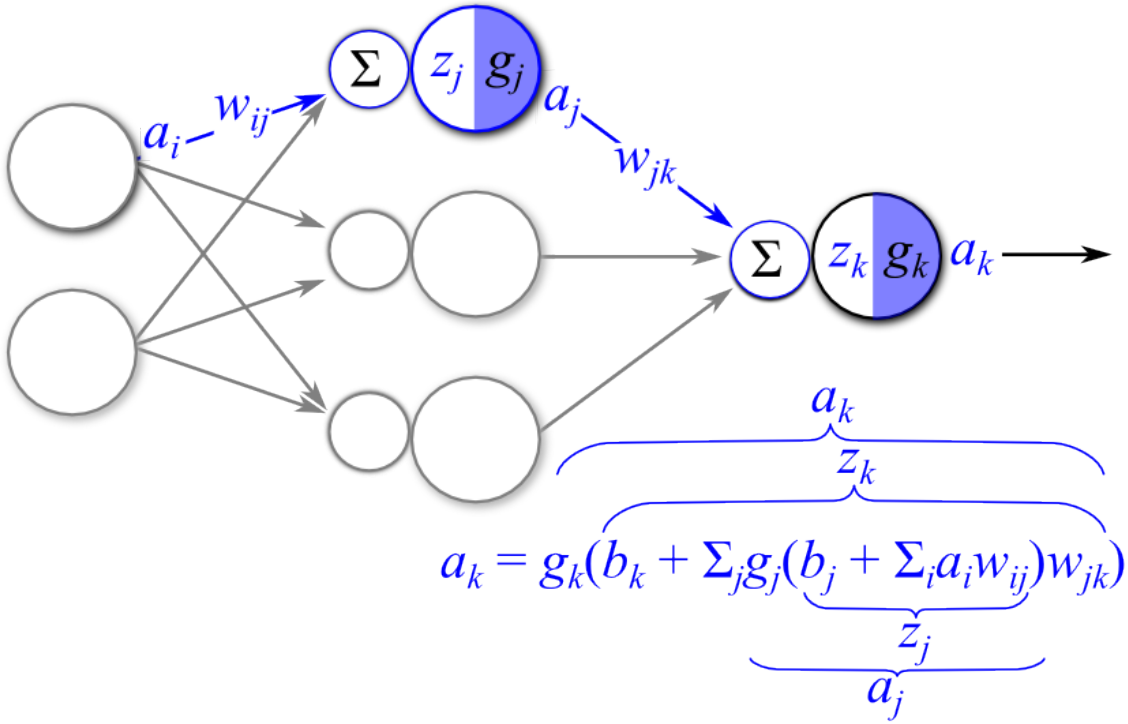
* How do we train this beast?
* Differentiate the loss function, same as ever!

GRADIENT BACKPROPAGATION

* (Derivation of backpropagation)

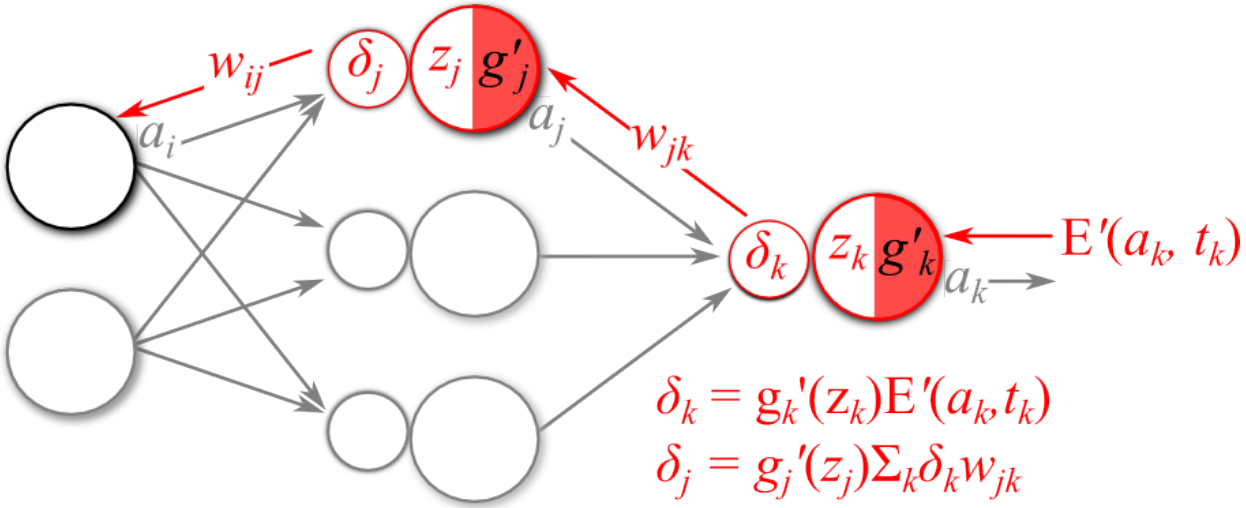
GRADIENT BACKPROPAGATION

I. Forward-propagate Input Signal



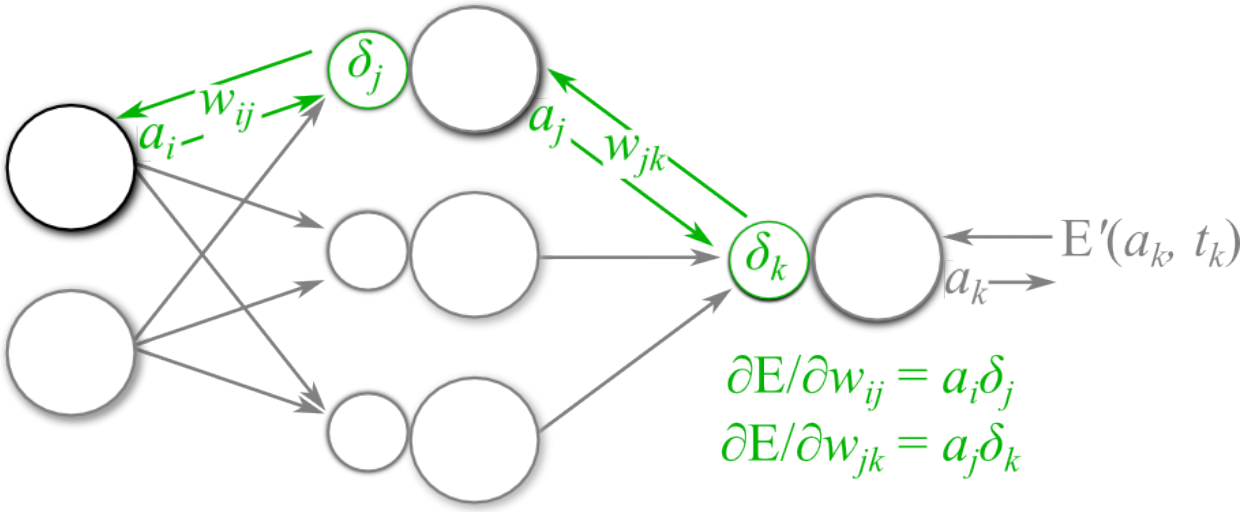
GRADIENT BACKPROPAGATION

II. Back-propagate Error Signals



GRADIENT BACKPROPAGATION

III. Calculate Parameter Gradients



GRADIENT BACKPROPAGATION

IV. Update Parameters

$$w_{ij} = w_{ij} - \eta(\partial E/\partial w_{ij})$$
$$w_{jk} = w_{jk} - \eta(\partial E/\partial w_{jk})$$

for learning rate η

GRADIENT BACKPROPAGATION

* Backprop enables **efficient** computation of gradients in networks of **arbitrary depth** by caching intermediate values δ

* It has been discovered in some form a few times in different fields

* In this field we often credit Rumelhart, Hinton, & Williams (1986)

ARTIFICIAL NEURAL NETWORKS

* Suppose number of hidden units = number of training samples

* Suppose $w_{ij} = X_j$

* *Suddenly*.. a kernel regression neural network, with k(a,b) = g(a.b)

h/t Michael Oliver

ARTIFICIAL NEURAL NETWORKS

* What can networks with hidden layers do?

* **Universal approximation theorem:** These networks can do *literally anything* (as long as there is a non-polynomial output nonlinearity)

  * See Cybenko (1989) paper attached to this lecture for proof

ARTIFICIAL NEURAL NETWORKS

* But: it is not necessarily true that every function is ***learnable*** using backprop

* Different network architectures (e.g. different numbers of hidden layers & units per layer) are good at learning different functions

NEXT TIME

* Artificial neural networks for solving vision problems (e.g. categorization)