Gradient Descent and Training Neural Networks

Danna Gurari
University of Colorado Boulder
Spring 2022

Review

- Last lecture:
  - Motivation for neural networks: need non-linear models
  - Neural network architecture: hidden layers
  - Neural network architecture: activation functions
  - Neural network architecture: output units
  - Programming tutorial

- Assignments (Canvas):
  - Lab assignment 1 due next week

- Questions?

Today's Topics

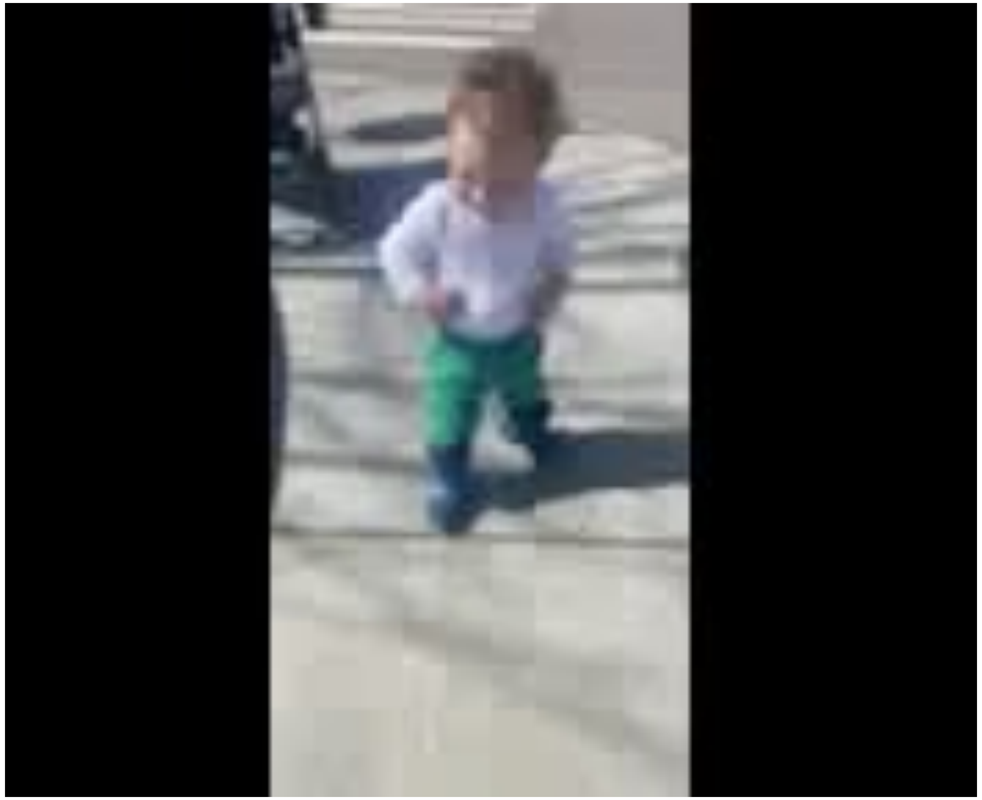
- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions

Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions

Objective Function: Analogous to Learning…

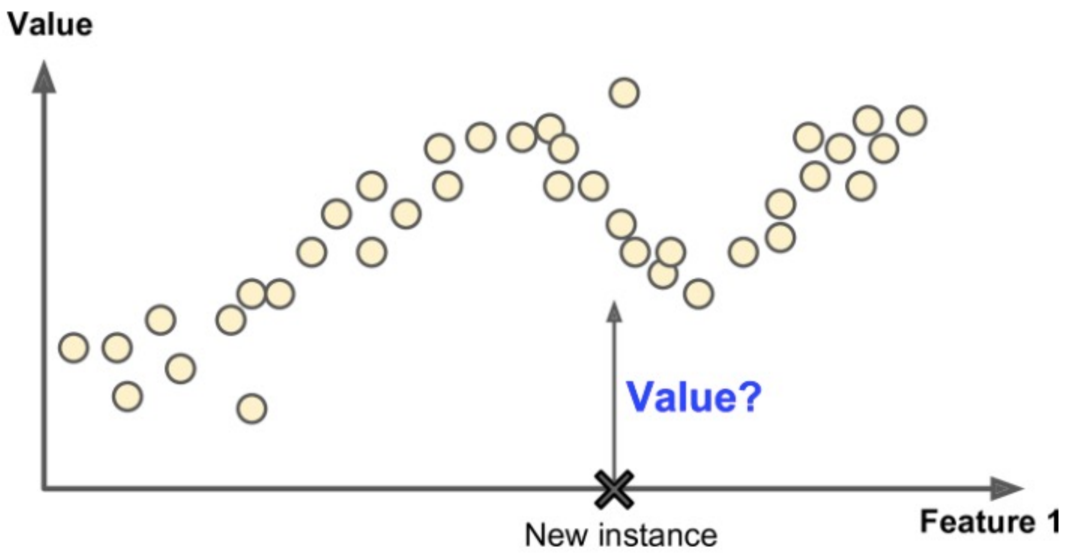
e.g., to walk



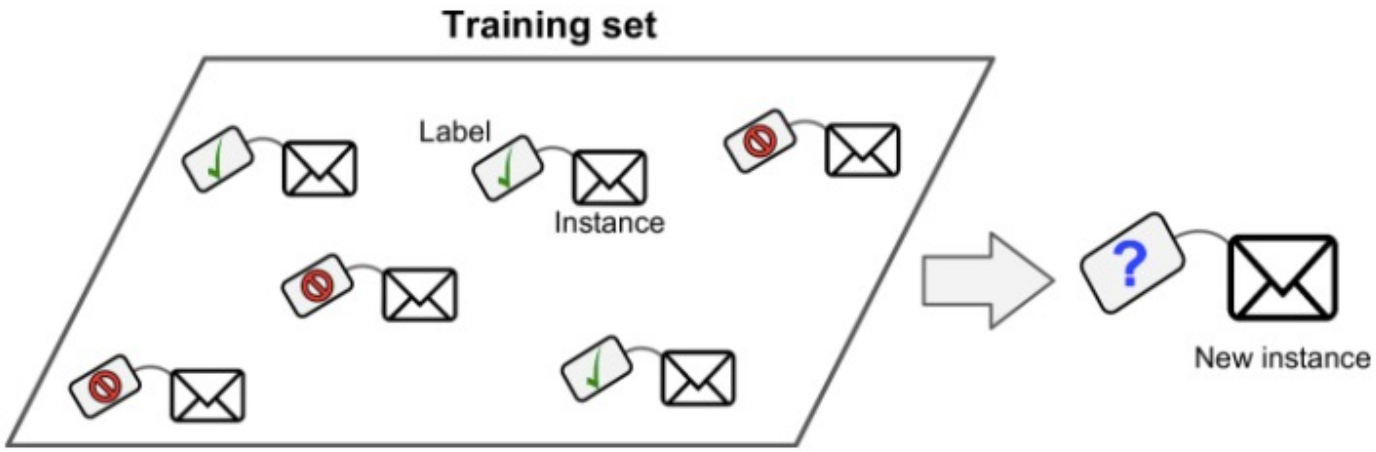
Key question: how do you measure/quantify task success?

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

Regression
(predict **continuous** value)

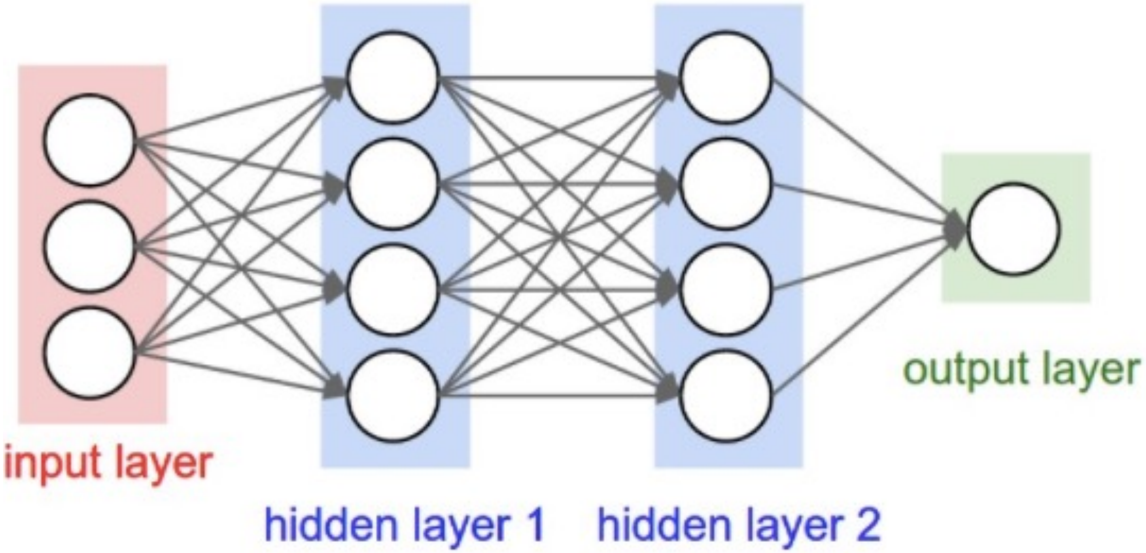


Classification
(predict **discrete** value)



Hands-on Machine Learning with Scikit-Learn & TensorFlow, Aurelien Geron

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal



e.g., make as small as possible the squared error (aka, L2 loss, quadratic loss)

Mean taken over *n* instances

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

True value

Predicted value

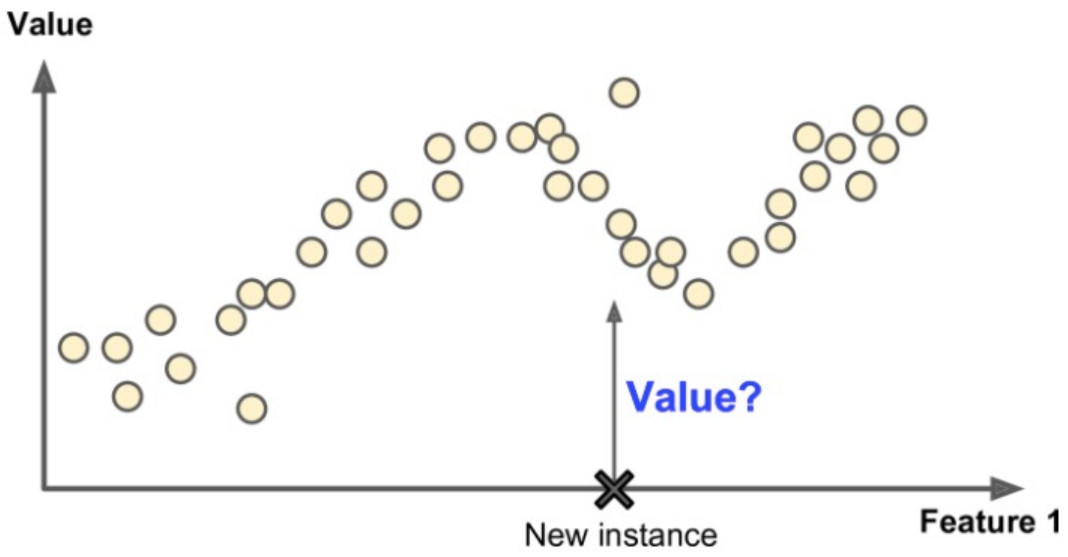
What is the range of possible values?

- Minimum: 0
  - i.e., all correct predictions
- Maximum: Infinity
  - i.e., incorrect predictions

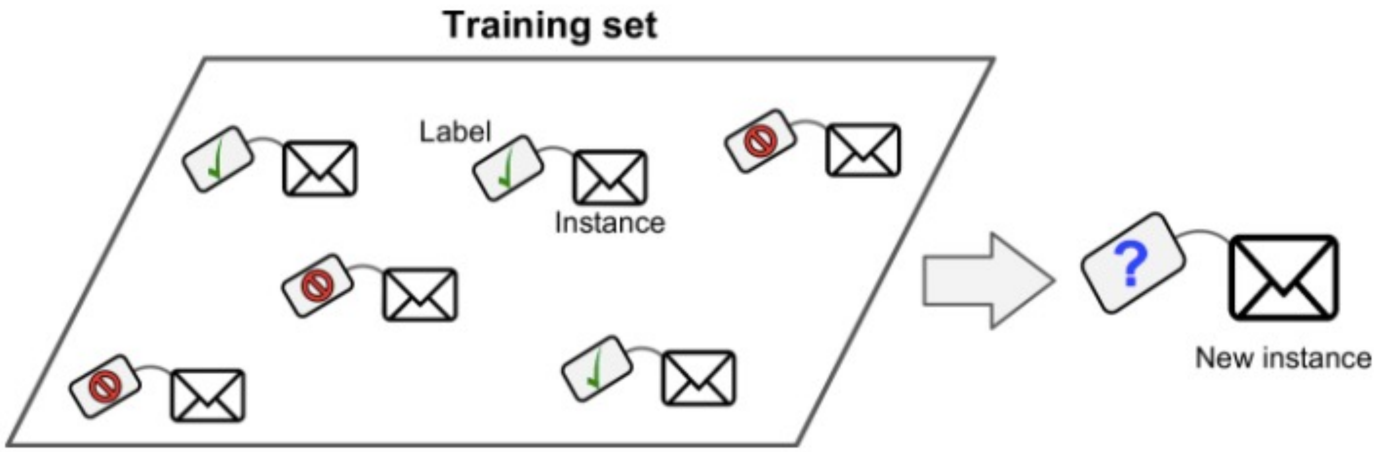
Figure source: http://cs231n.github.io/neural-networks-1/

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

Regression
(predict **continuous** value)

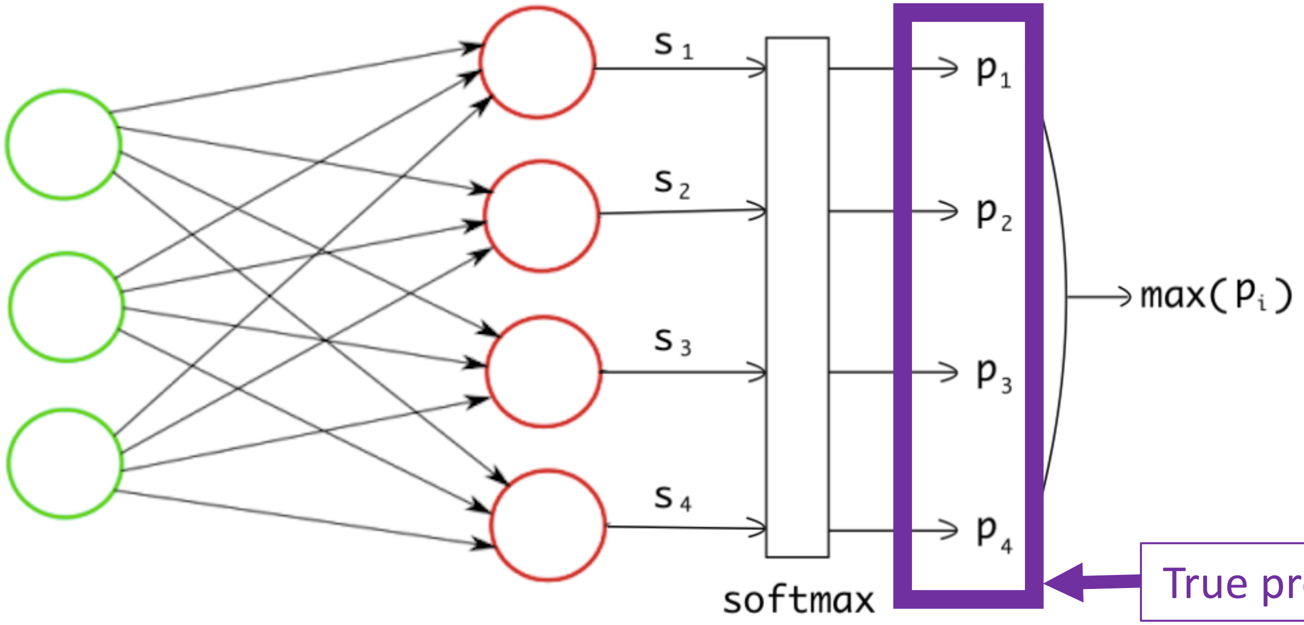


Classification
(predict **discrete** value)



Hands-on Machine Learning with Scikit-Learn & TensorFlow, Aurelien Geron

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal



e.g., make as small as possible the distance between predicted and true class distributions for the training examples with cross entropy loss

True prediction is 1 hot vector (i.e., one 1 and the rest 0s)

Figure source: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

Probability distribution of true class

Probability distribution of predicted class

Number of classes

Recall, truth is set to 1 for one class and 0 otherwise

Observed features

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\
&= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k|x) \\
&= -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)} \\
&= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}
\end{aligned}
$$

Simplifies to the log of the predicted probability for the correct class (i.e., negative log likelihood loss)

Excellent background: https://web.stanford.edu/~jurafsky/slp3/5.pdf

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

Probability distribution of predicted class

Probability distribution of true class

Number of classes?

Recall, truth is set to 1 for one class and 0 otherwise

Observed features

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\
&= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k | x) \\
&= -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)} \\
&= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}
\end{aligned}
$$

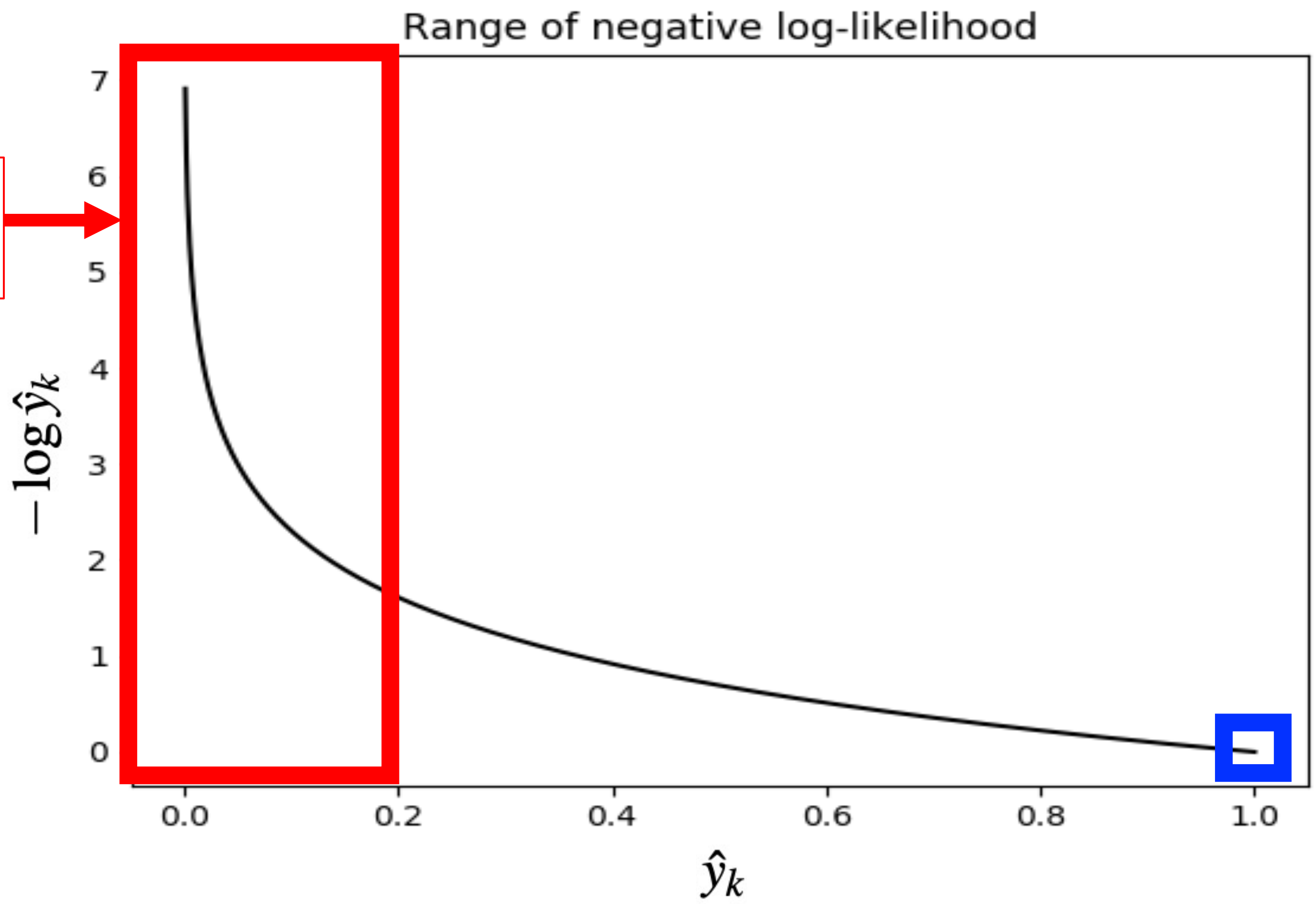
What is the range of possible values?

- Minimum: 0
  - i.e., correct prediction: negative log of 1
- Maximum: Infinity
  - i.e., incorrect prediction: negative log of 0

Excellent background: https://web.stanford.edu/~jurafsky/slp3/5.pdf

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

More confidently wrong predictions lead to greater error



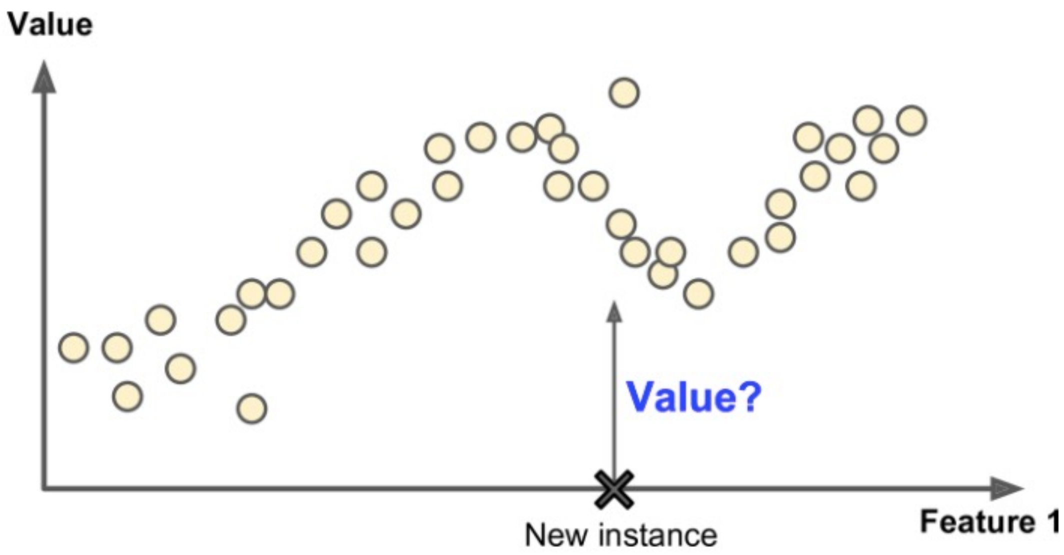
What is the range of possible values?

- Minimum: 0
  - i.e., correct prediction: negative log of 1
- Maximum: Infinity
  - i.e., incorrect prediction: negative log of 0

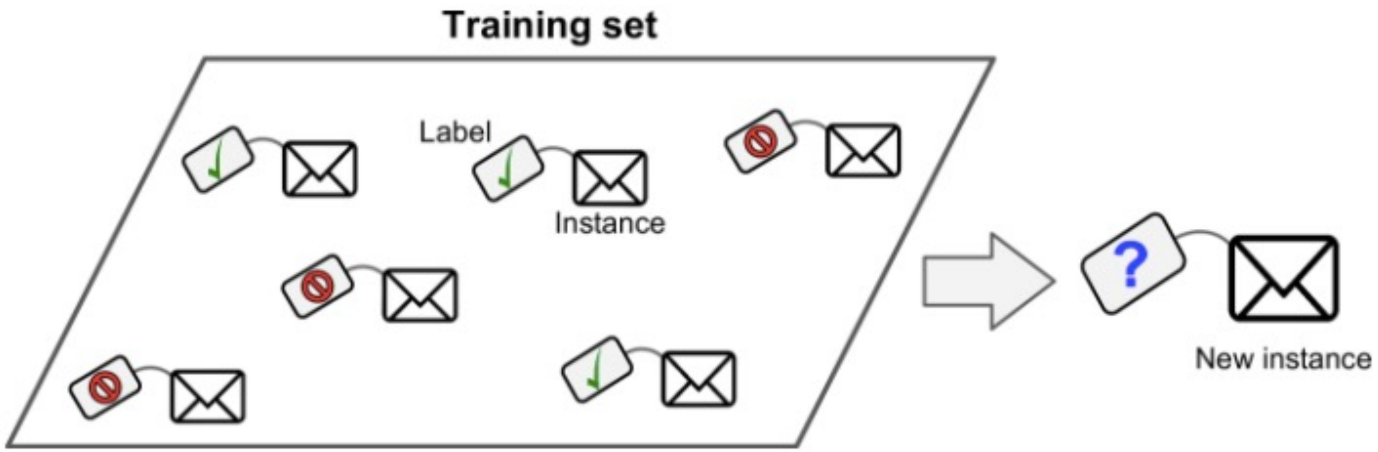
Source: https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

Objective Function: Learn Model Parameters that Achieve a Specified (Measurable) Goal

Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



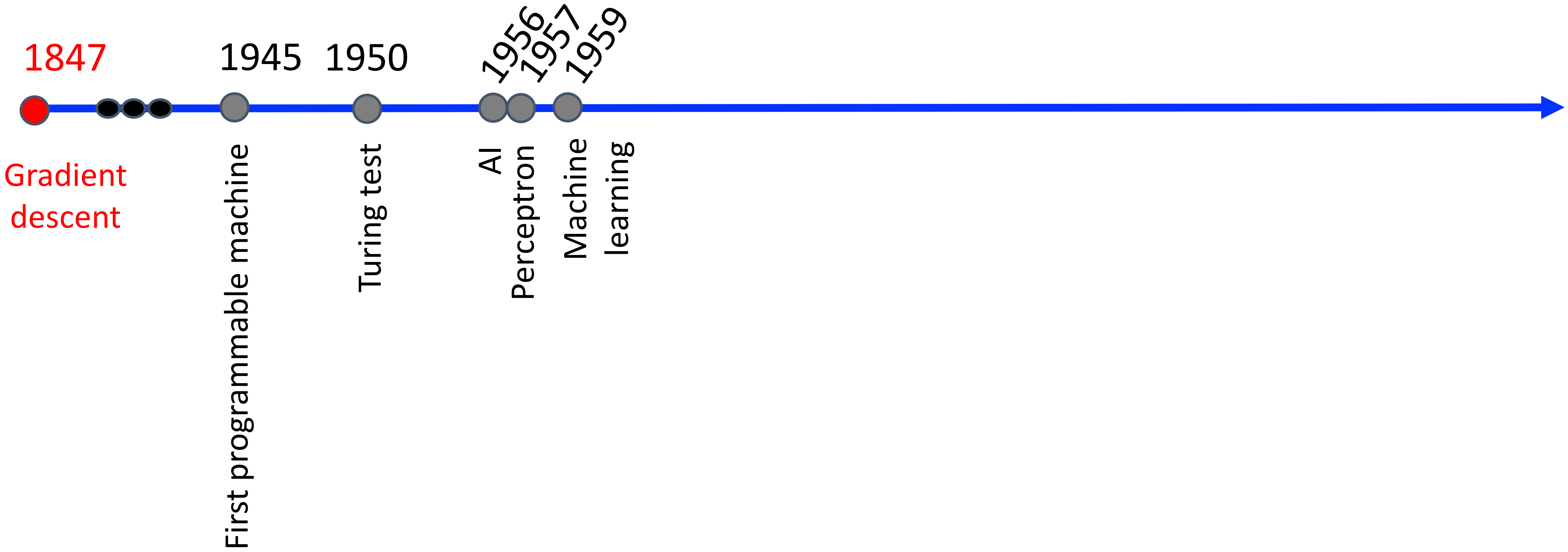
Hands-on Machine Learning with Scikit-Learn & TensorFlow, Aurelien Geron

MANY objective functions exist, and we will examine popular ones in this course

Today’s Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions

Historical Context: Gradient Descent

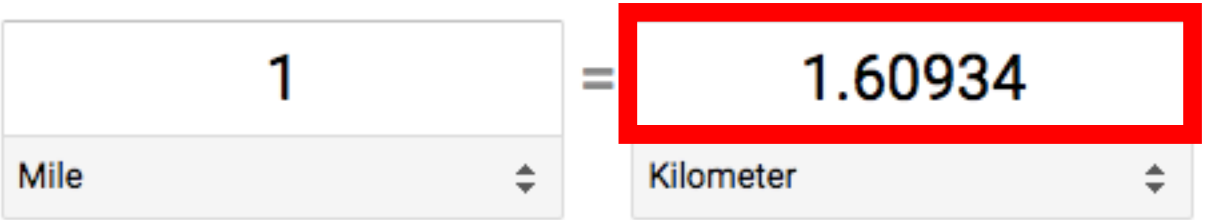


Louis Augustin Cauchy: Compte Rendu `a l’Acad´emie des Sciences of October 18, 1847

Scalable way to train nonlinear models on “big data”

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn linear model for converting kilometers to miles when only observing the input “miles” and output “kilometers”



Miles → Kilometers = miles x constant → Kilometers

Make Your Own Neural Network, Tariq Rashid

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant

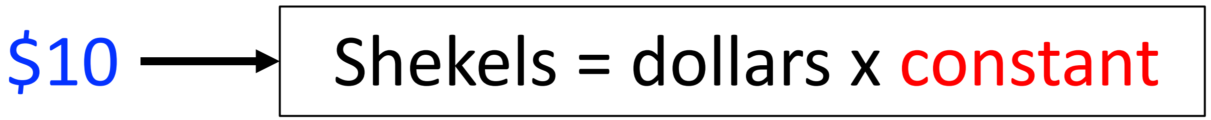
Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

\$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels



Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

$10 → Shekels = dollars x constant → Error = Guess - Correct

- Idea: iteratively adjust constant (i.e., model parameter) to try to reduce the error

Gradient Descent: Intuition

- Iteratively search for model parameters (i.e., weights and biases) that solve optimization problem (i.e., minimize or maximize an objective function)

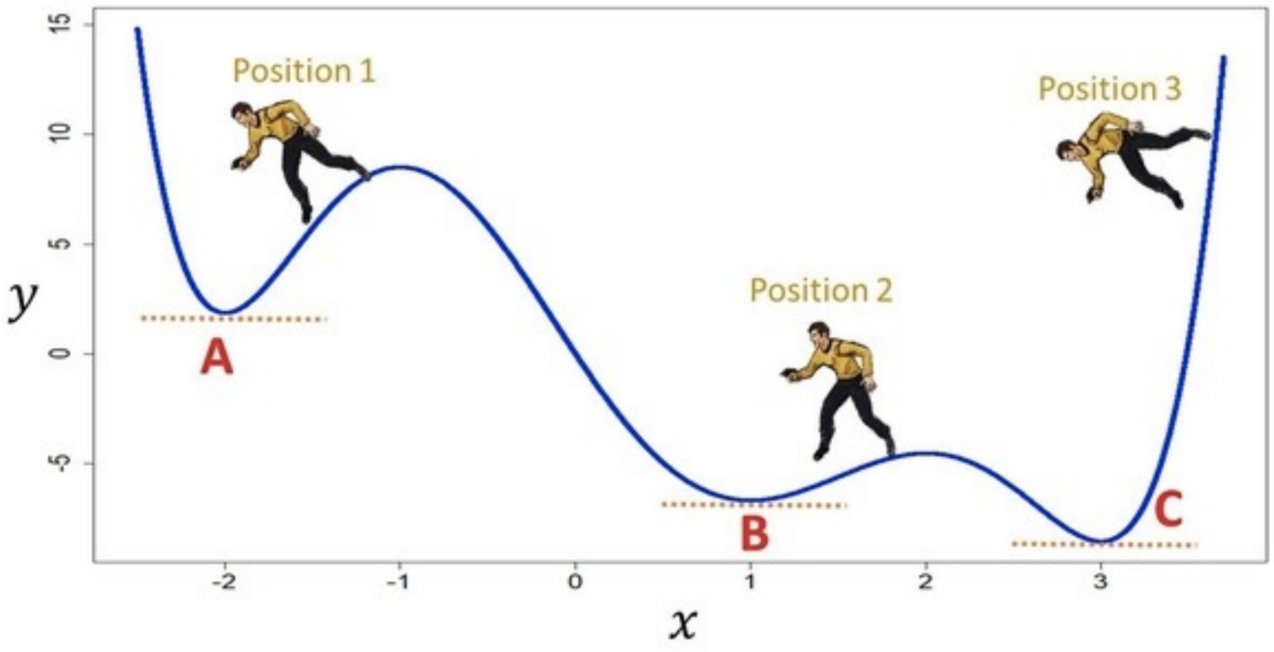
Analogy: hiking to the bottom of a mountain range… blind or blindfolded!

Start

End Point (Minimum)

Gradient Descent: Employs Calculus

- Idea: use derivatives!
  - Derivatives tells us how to change the input x to make a small change to the output f(x)
  - Gradient is a vector that indicates how f(**x**) changes as each function variable changes (i.e., partial derivatives)
- Gradient descent:
  - Iteratively take steps in the opposite direction of the gradient to minimize the function



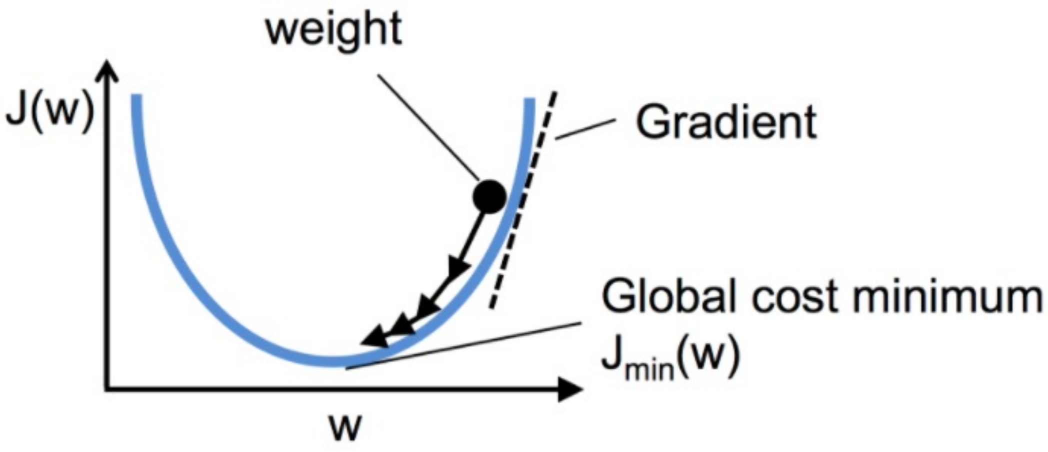
Which letter(s) are the global minima?

Which letter(s) are local minima?

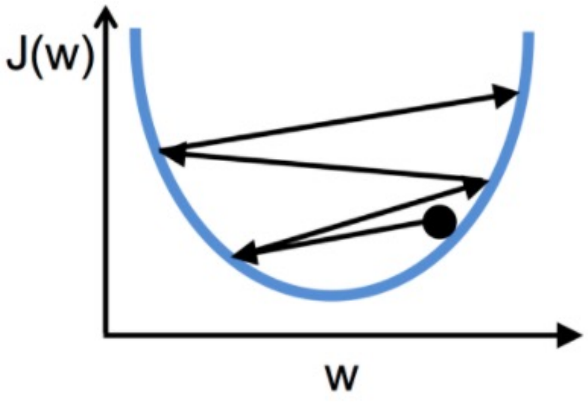
Gradient Descent: How Much to Update?

- Step size = learning rate
  - (a) When learning rate is too small, convergence to good solution will be slow
  - (b) When learning rate is too large, convergence to a good solution is not possible

(a)



(b)



- Next lecture: examination of how to learn effectively using the gradients

https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch02/ch02.ipynb

Gradient Descent: How Often to Update?

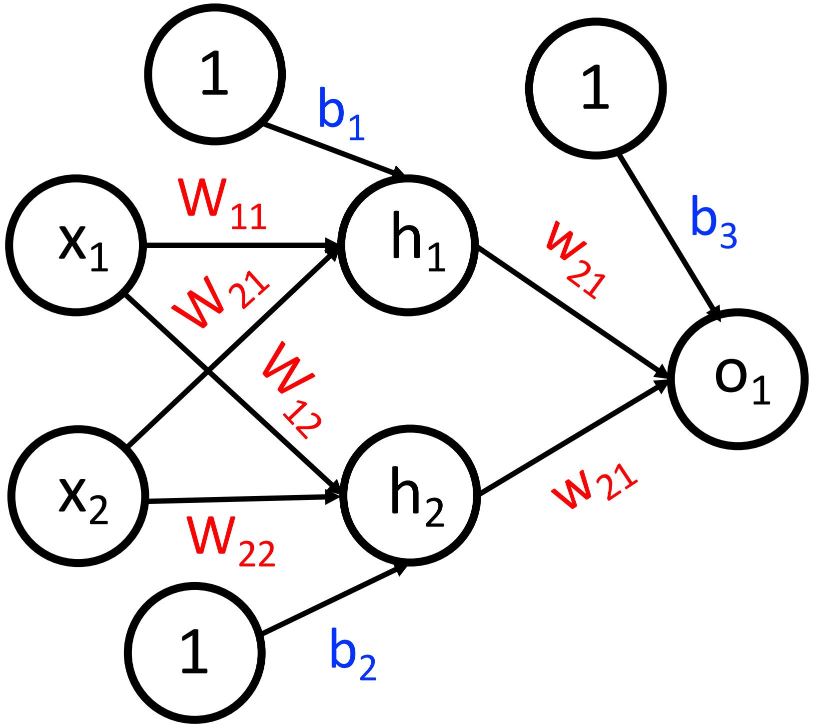
- Use calculations over ***all training examples*** (Batch gradient descent)
  - Less bouncing but can be slow or infeasible when dataset is large
- Use calculations from ***one training example*** (Stochastic gradient descent)
  - Fast to compute and can train using huge datasets (stores one instance in memory at each iteration) but updates are expected to bounce a lot
- Use calculations over ***subset of training examples*** (Mini-batch gradient descent)
  - Bounces less erratically than SGD and can train using huge datasets (store some instances in memory at each iteration) but can be slow or infeasible when dataset is large
- Often mini-batch gradient descent is used with maximum # of examples that fit in memory

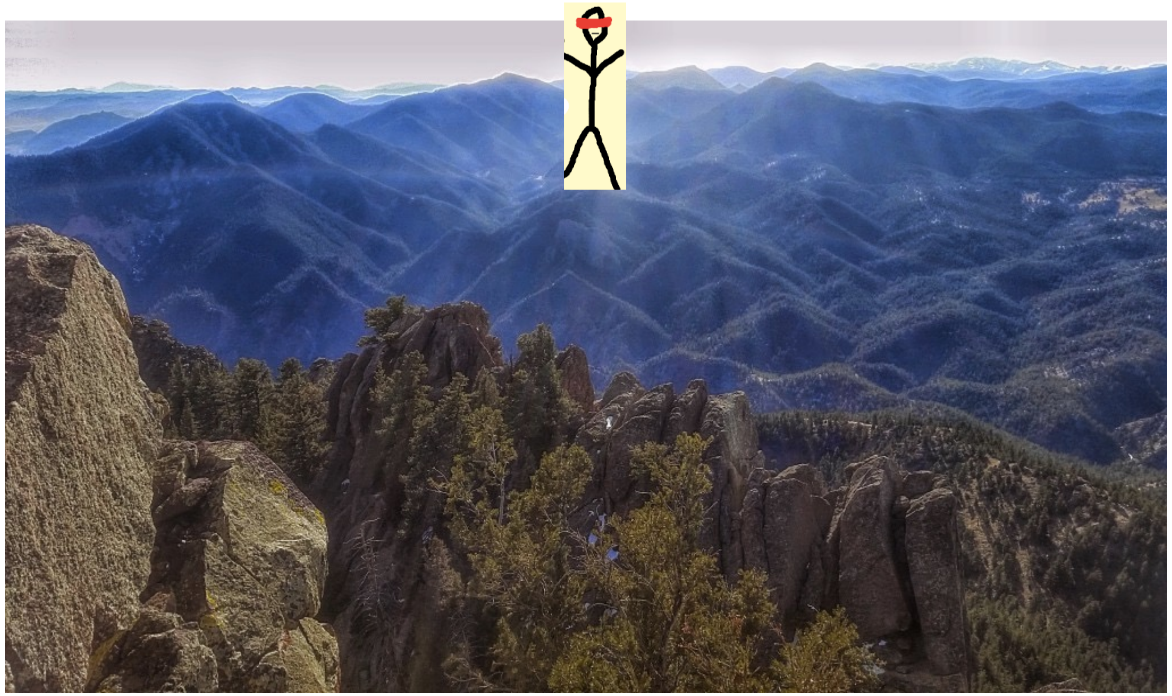
Today’s Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions

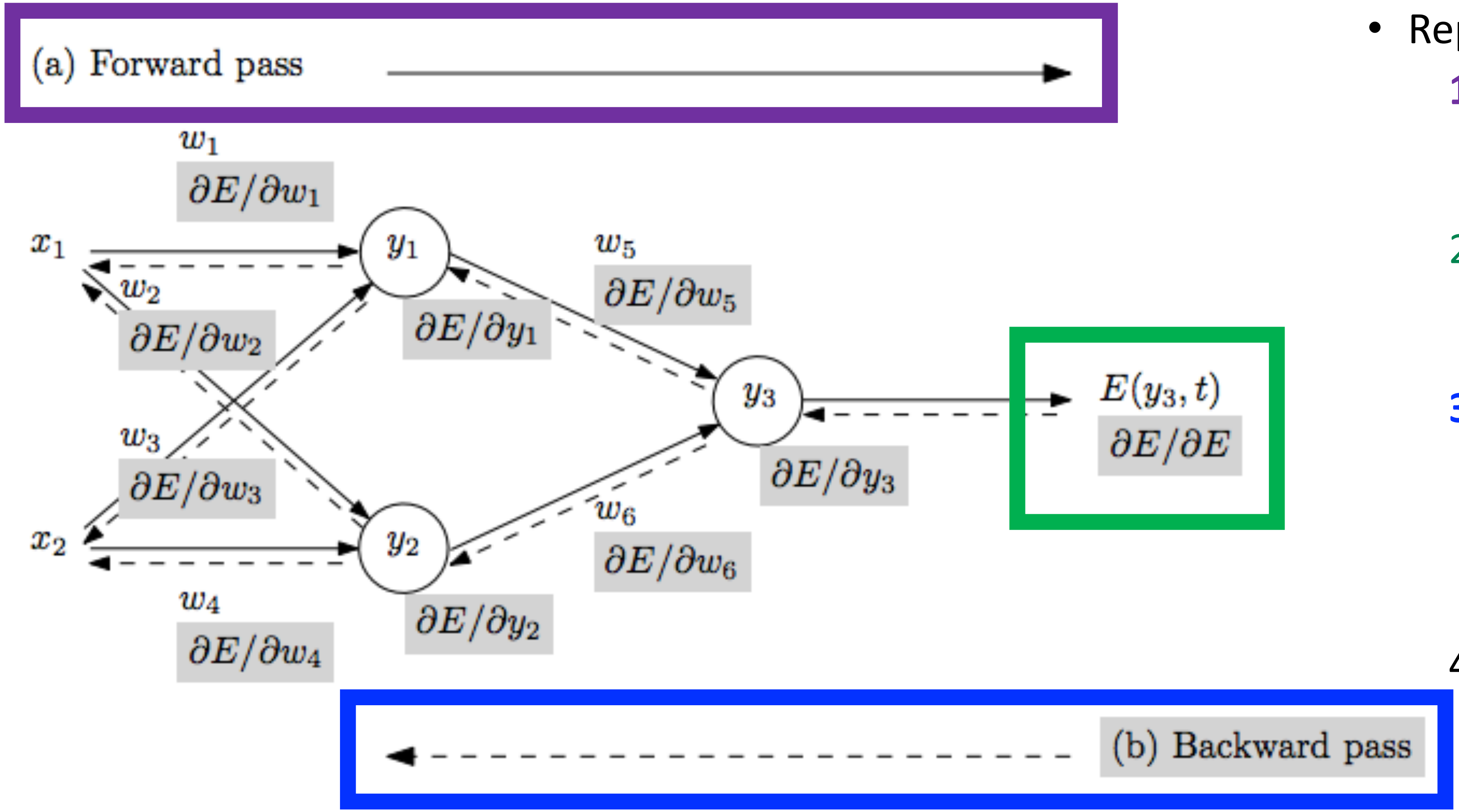
Summary: Approach to Train Neural Network

- Learn model parameters (weights, biases) that minimize an objective function using gradient descent; e.g.,





Training: How Neural Networks Learn



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

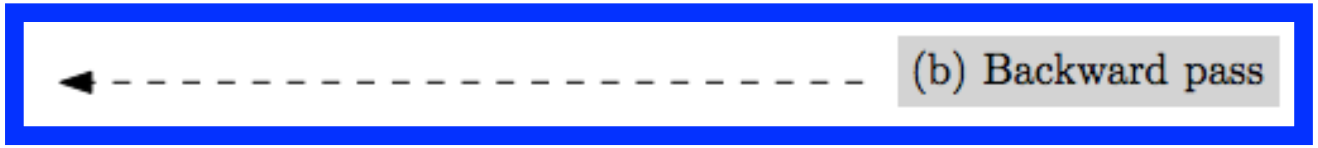
Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Training: How Neural Networks Learn

Key challenge: calculating gradients. Depends on:

1) Objective function

2) Activation functions

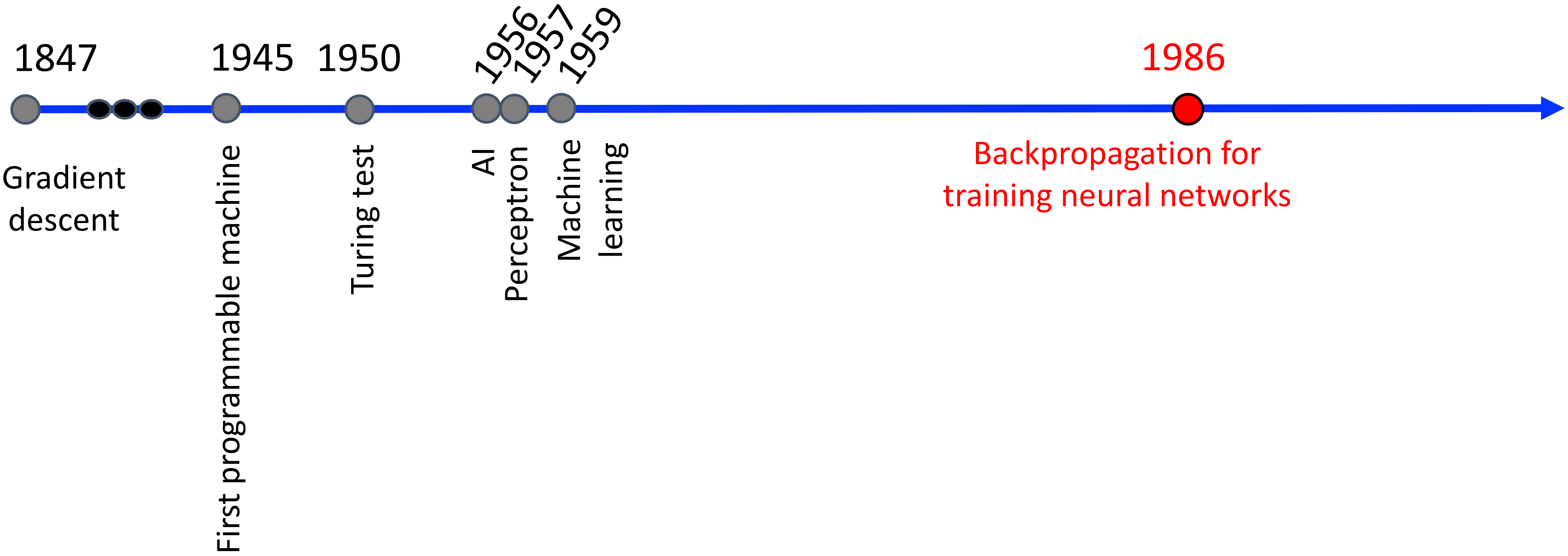


Repeat until stopping criterion met:

1. **Forward pass**: propagate training data through model to make prediction
2. Quantify the dissatisfaction with a model's results on the training data
3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Solution: Backpropagation to Compute Gradients



D. Rulhart, G. Hinton, and R. Williams, Learning Internal Representations by Error Propagation, 1986.

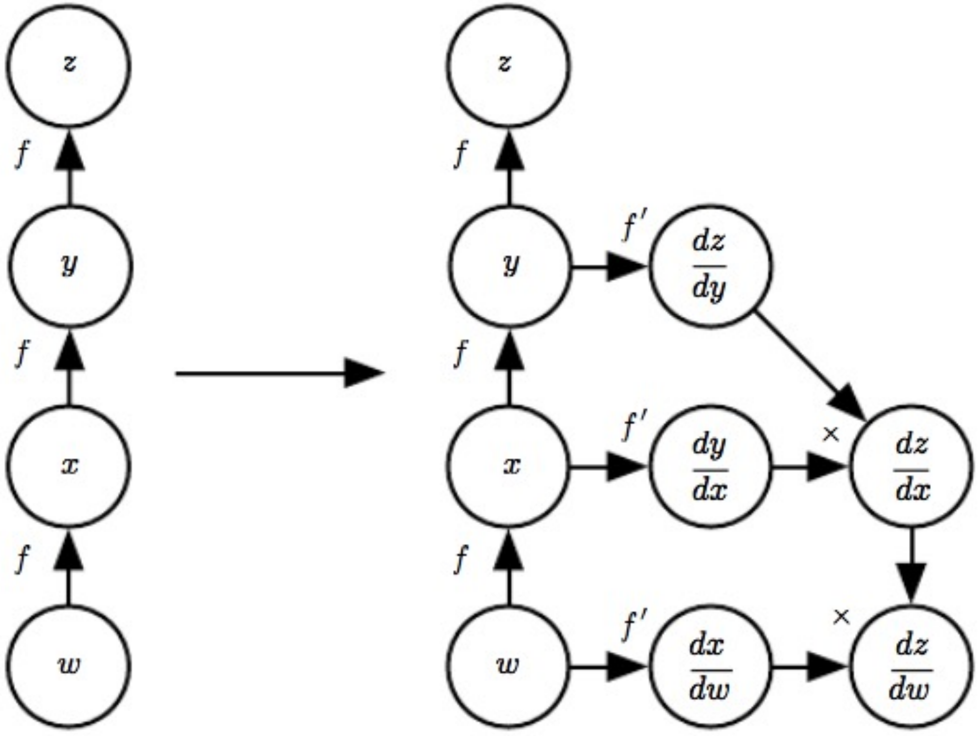
Solution: Backpropagation to Compute Gradients

- Idea: compute gradient on objective function to decide how to adjust each model parameter to get closer to solving the optimization problem
- Key observation: networks are functions connected in a chain

$$x = f(w), y = f(x), z = f(y)$$

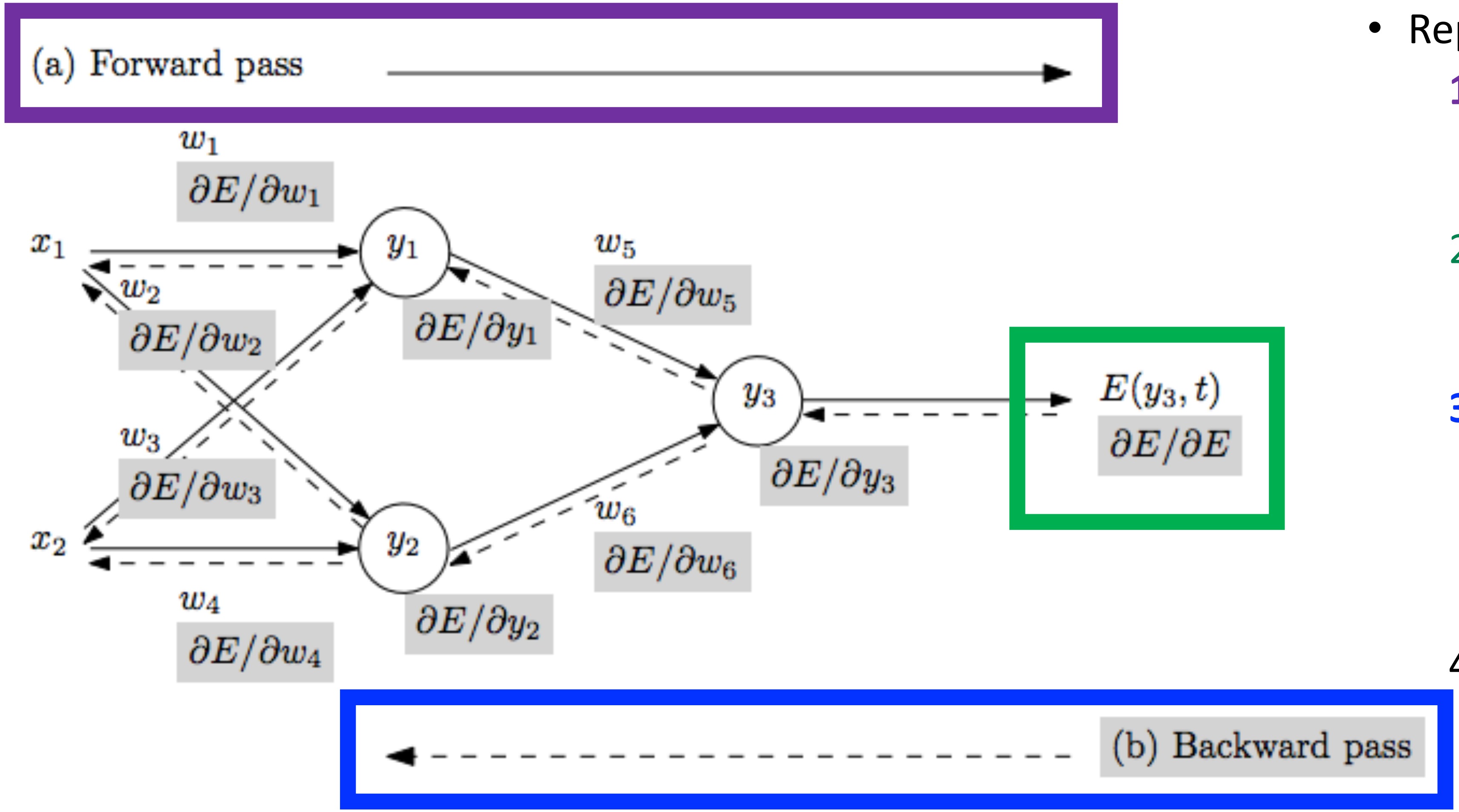
Can use chain rule of calculus (and so compute from top to bottom where derivatives on the top are used to compute derivatives at the bottom);

e.g., $$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$



Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

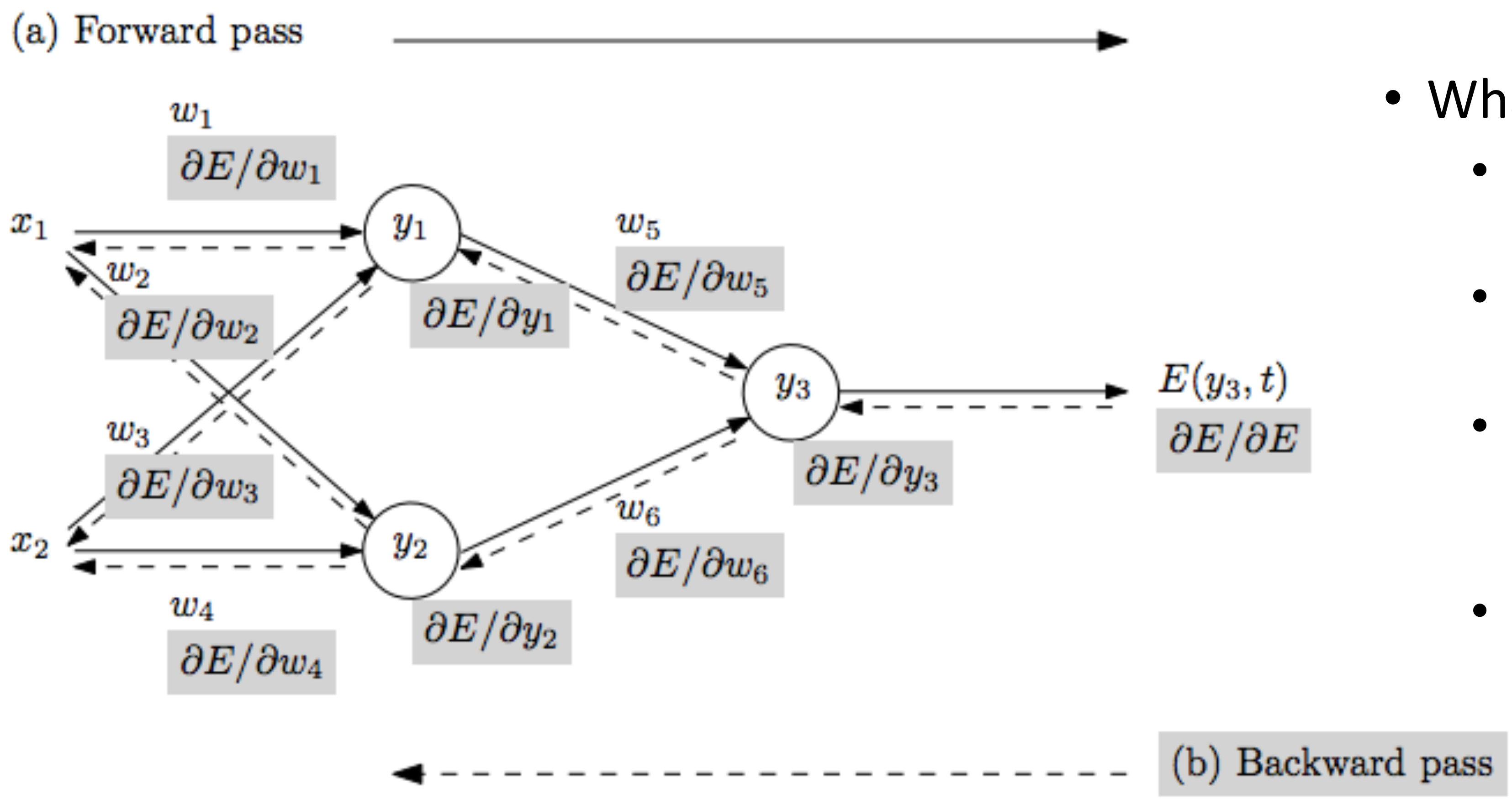
Training: How Neural Networks Learn



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

When to Stop Training Neural Networks?



- What stopping criterion to use?
  - Weight changes are incredibly small
  - Finished a pre-specified number of epochs
  - Percentage of misclassified example is below some threshold
  - …

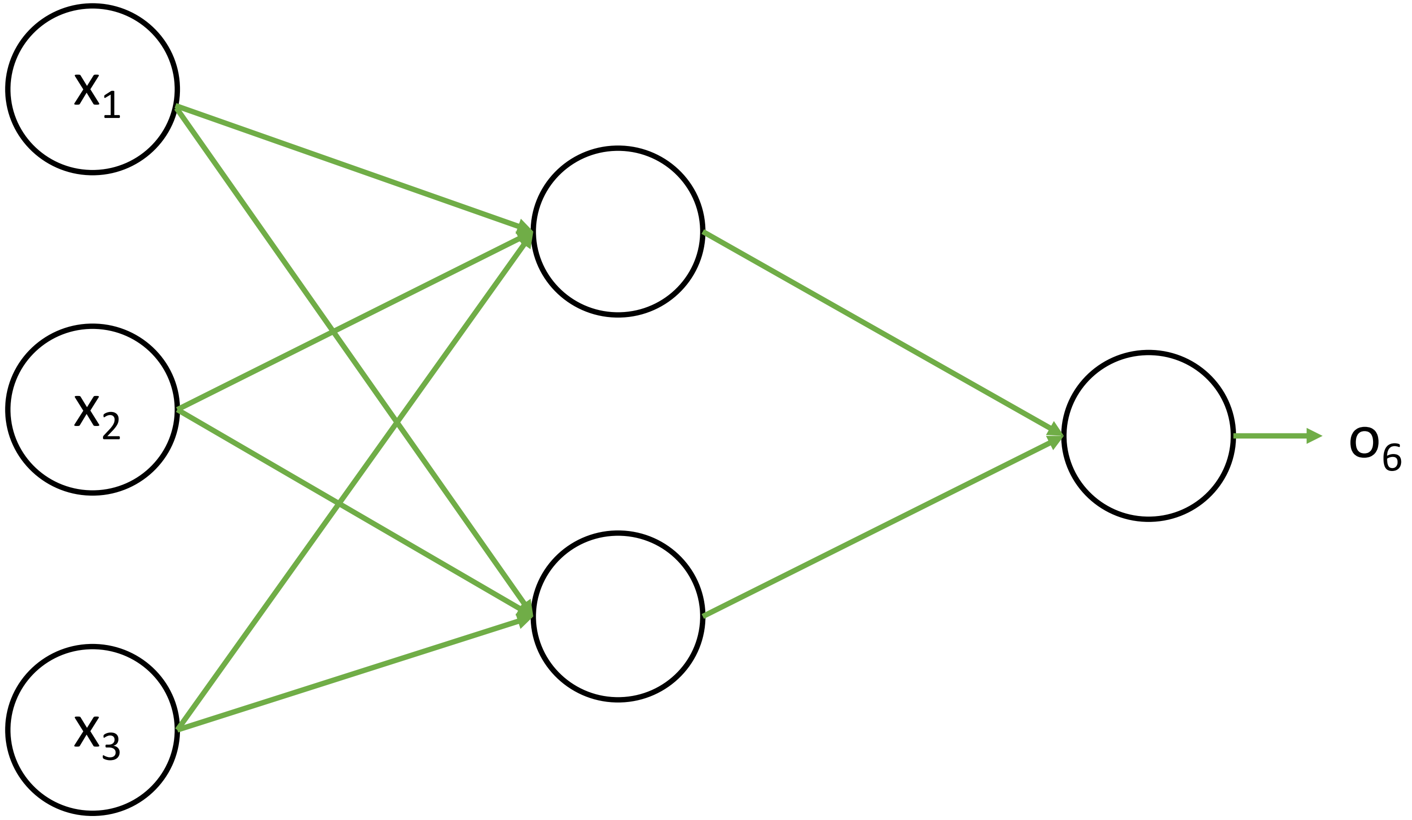
Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Example

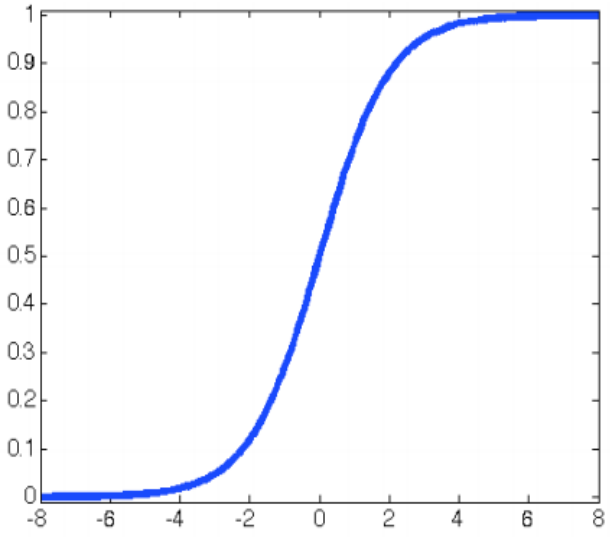
- Binary classification: predict if a student will get a B- or better

- Inputs:
    - Percentage of assignments completed
    - Percentage of readings read
    - Percentage of lectures watched

Example: Choose Neural Network Architecture



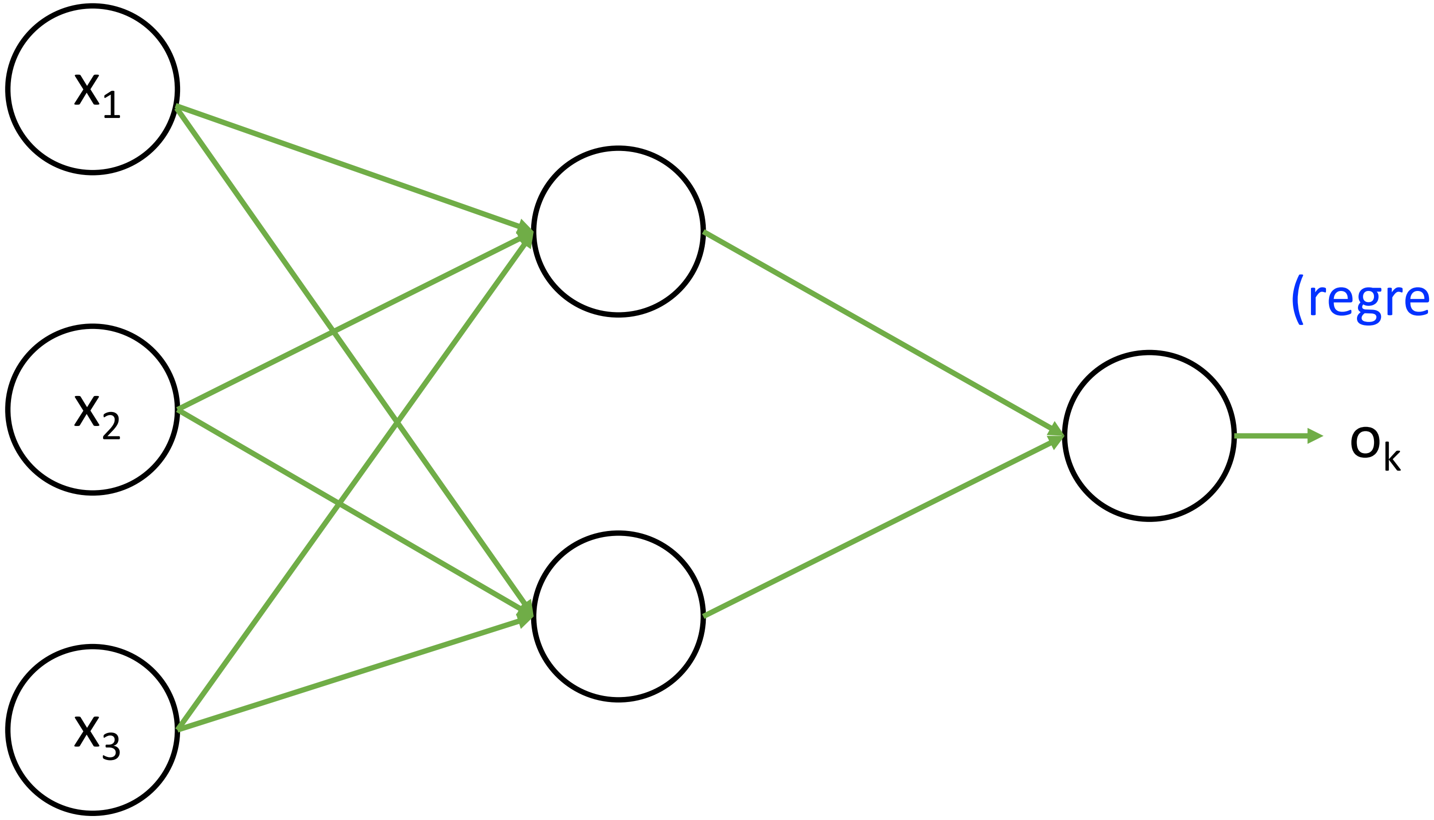
Sigmoid Activation Function



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

Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Choose Loss Function for Training

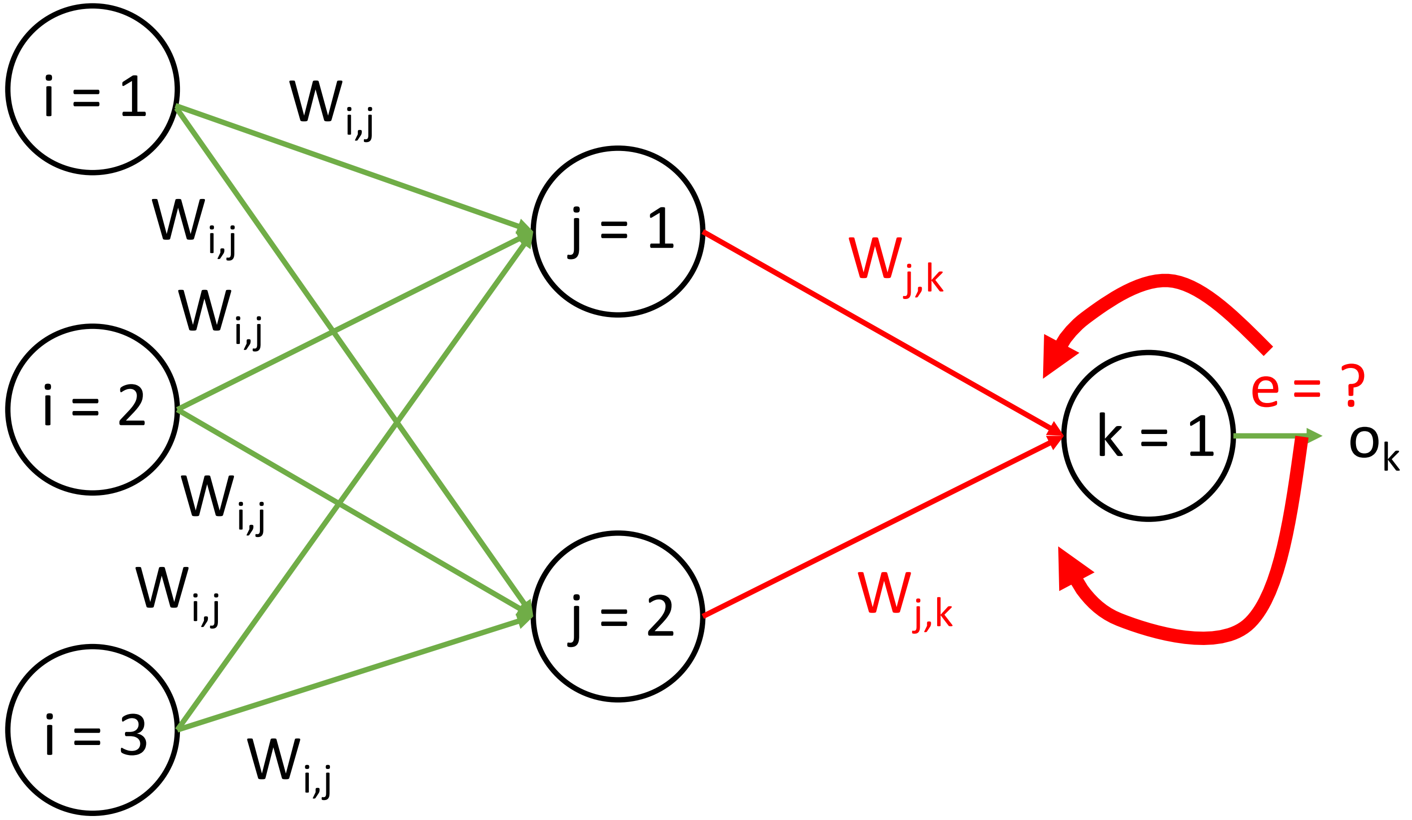


Squared Error Function
(regression or binary classification problem)

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(t_k - o_k)^2$$

Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Resolve How to Compute Gradient? (Output Layer)



t is a constant value:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (t_k - o_k)^2$$

Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Resolve How to Compute Gradient? (Output Layer)

t is a constant value:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(t_k - o_k)^2$$

Using the following chain rule :

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} * \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial o_k} = -2(t_k - o_k)$$

Sigmoid activation function: $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\frac{d\sigma(x)}{dx} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)$$

$$\frac{d\sigma(x)}{dx} = (1-\sigma(x))\,\sigma(x)$$

Example: Resolve How to Compute Gradient? (Output Layer)

t is a constant value:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(t_k - o_k)^2$$

$$\frac{\partial E}{\partial o_k} = -2(t_k - o_k)$$

Sigmoid activation function: $\sigma(x) = \frac{1}{1 + e^{-x}}$

Using the following chain rule :

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} * \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{d\sigma(x)}{dx} = (1 - \sigma(x))\,\sigma(x)$$

We can rewrite our function as follows:

For efficiency, compute last

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k)\, sigmoid(\sum_j w_{jk} * o_j) * (1 - sigmoid(\sum_j w_{jk} * o_j)) * o_j$$

Example: Resolve How to Compute Gradient? (Output Layer)

t is a constant value:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(t_k - o_k)^2$$

Using the following chain rule :

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} * \frac{\partial o_k}{\partial w_{jk}}$$

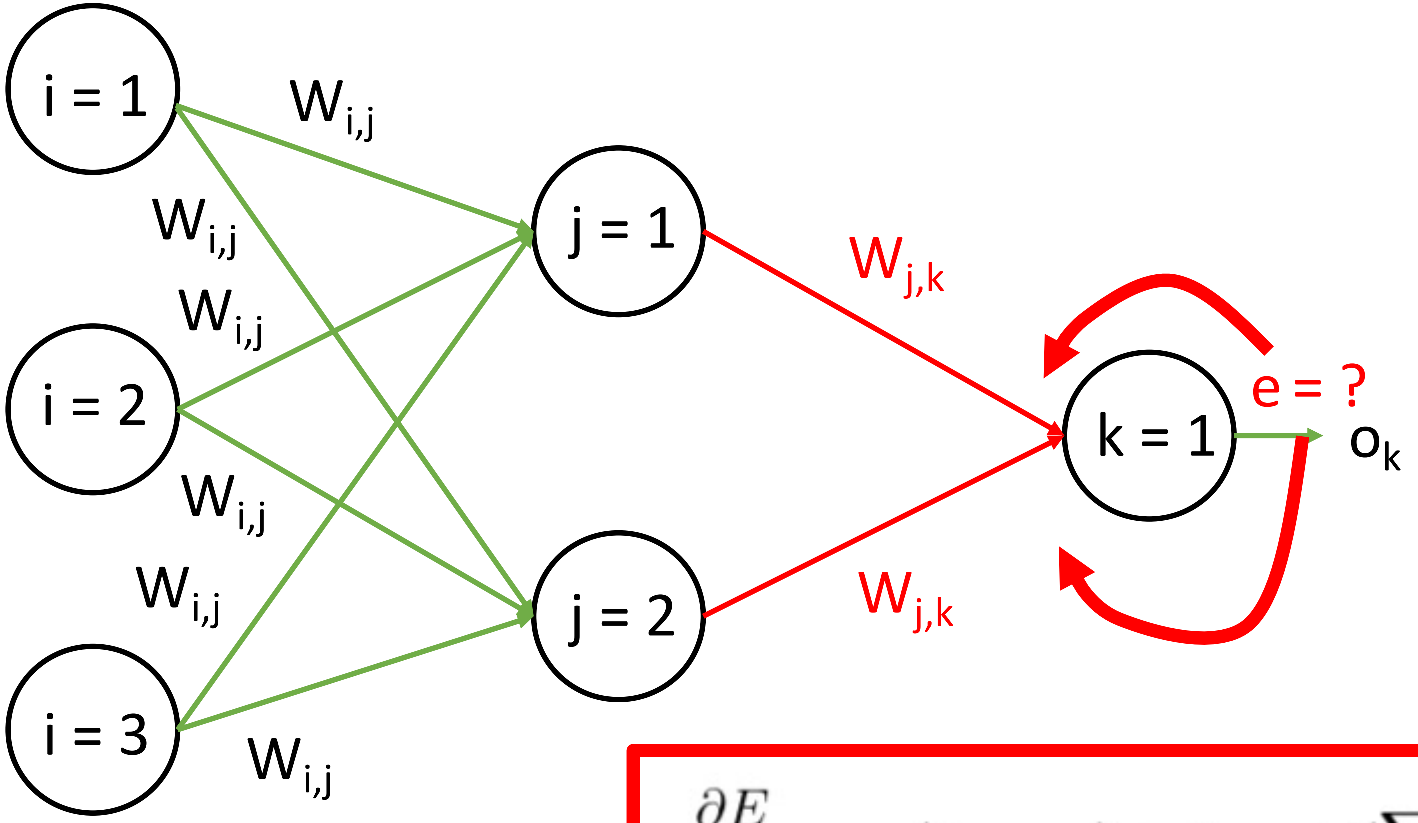
$$\frac{\partial E}{\partial o_k} = -2(t_k - o_k)$$

Sigmoid activation function: $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\frac{d\sigma(x)}{dx} = (1 - \sigma(x))\,\sigma(x)$$

Key Observation: Possible because activation function and loss function are **differentiable**!!!

Example: Resolve How to Compute Gradient? (Output Layer)

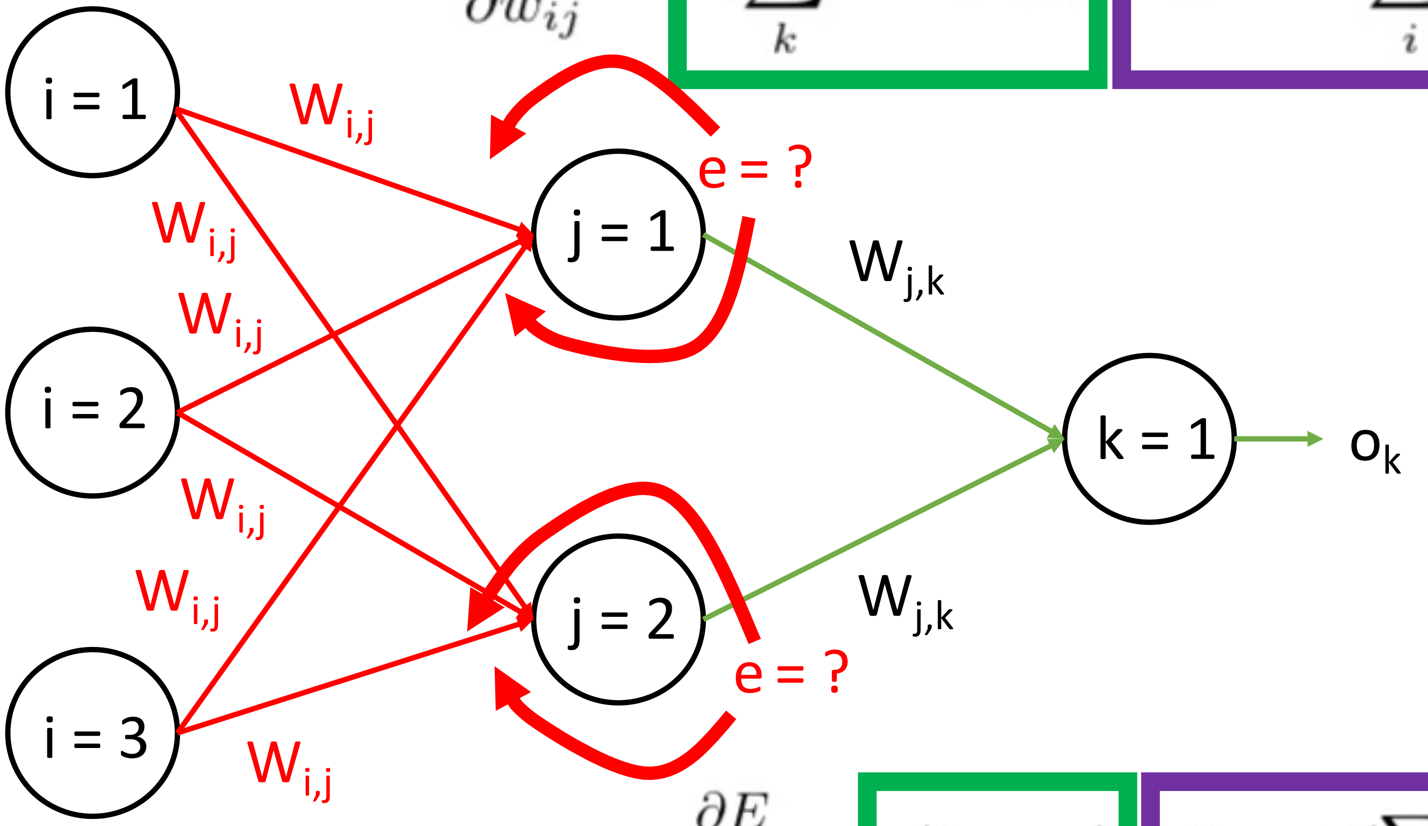


$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}(t_k - o_k)^2$$

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) * sigmoid(\sum_j w_{jk} * o_j) * (1 - sigmoid(\sum_j w_{jk} * o_j))$$

Example: How to Compute Gradient? (Hidden Layer)

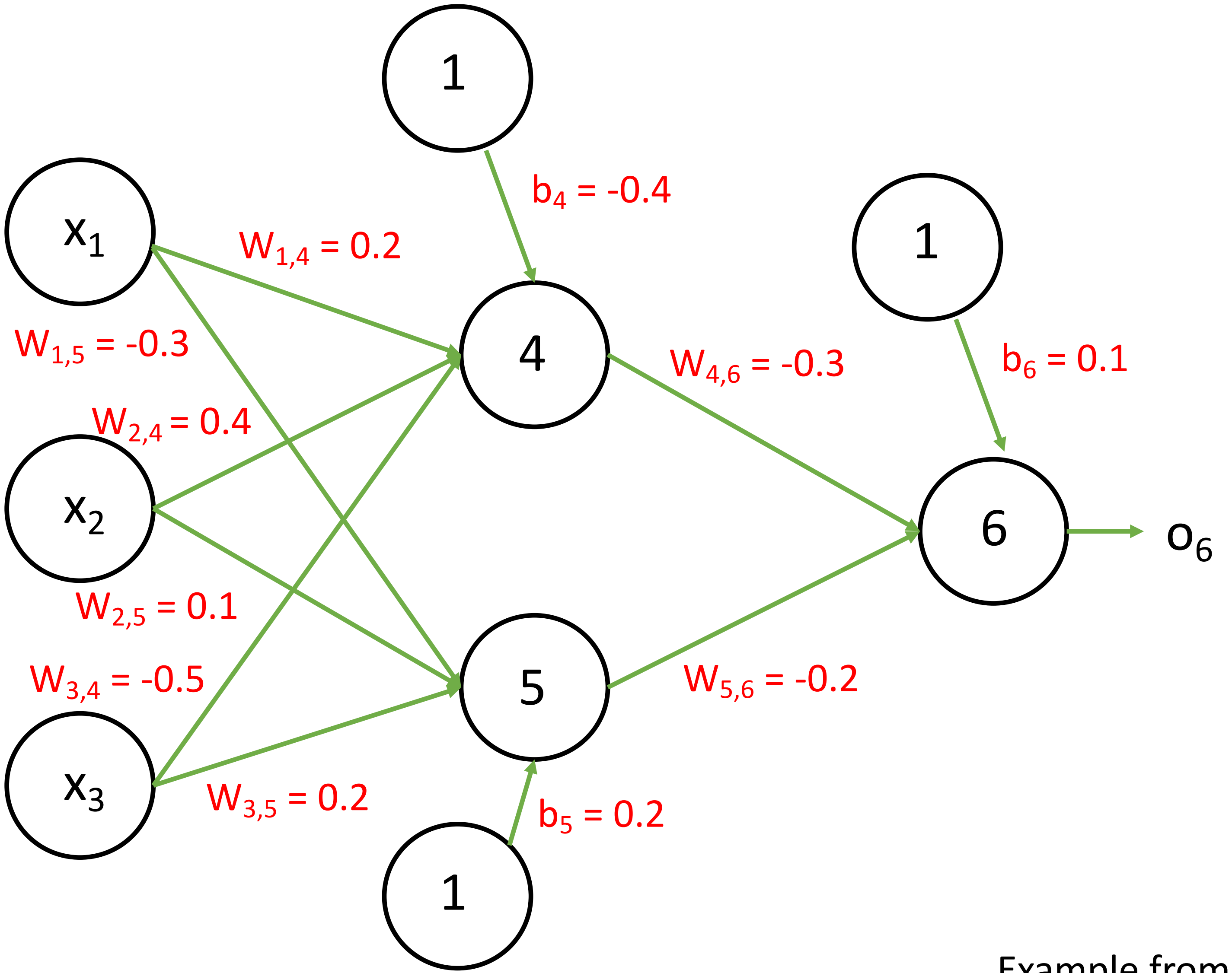
$$\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k Err_k w_{jk}\right) * sigmoid\left(\sum_i w_{ij} * o_i\right) * \left(1 - sigmoid\left(\sum_i w_{ij} * o_i\right)\right)$$



Symmetry in how to solve for parameters in hidden layers

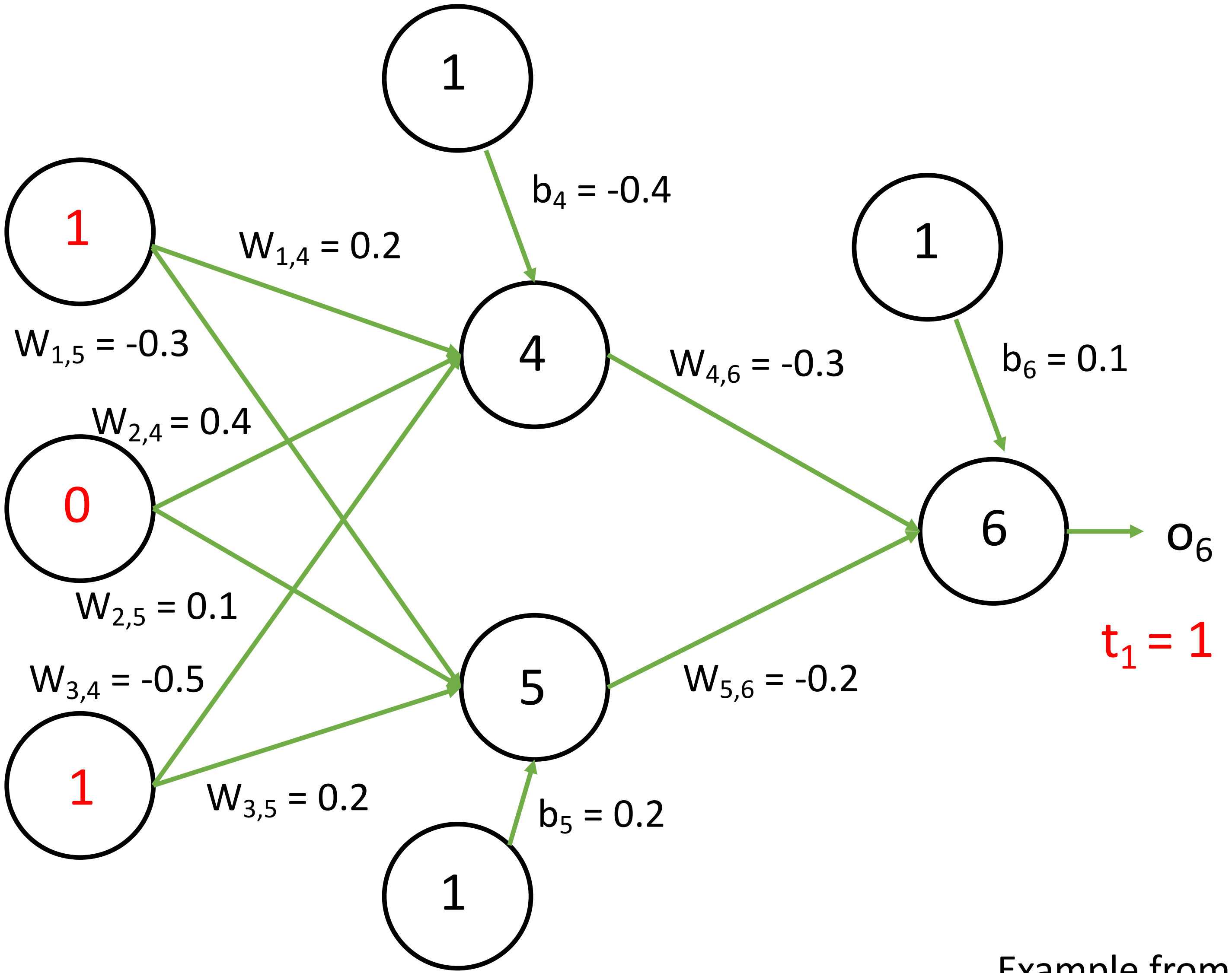
$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) * sigmoid\left(\sum_j w_{jk} * o_j\right) * \left(1 - sigmoid\left(\sum_j w_{jk} * o_j\right)\right)$$

Example: Initialize Model Parameters



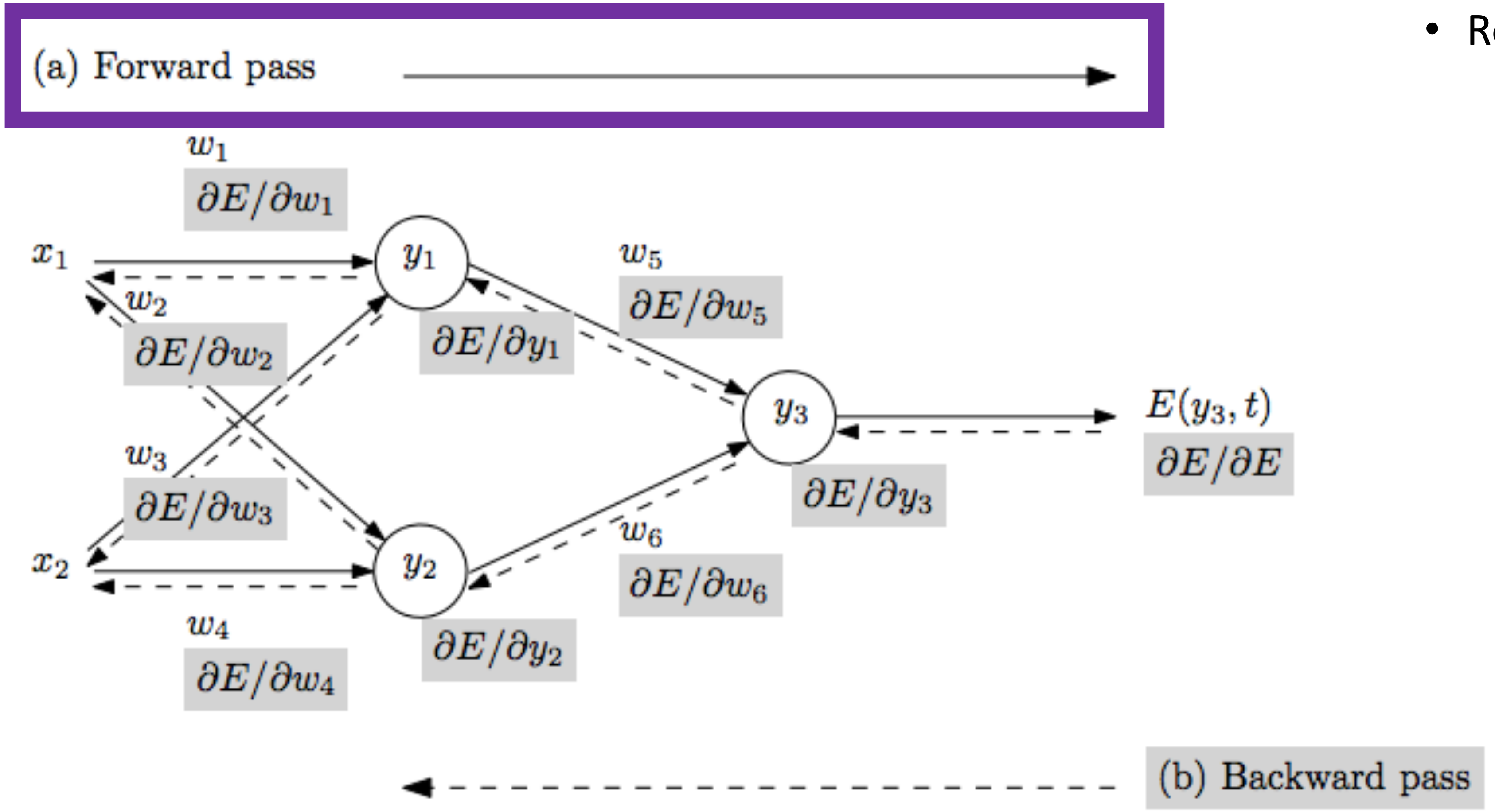
Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Input Training Example



Example from: Jiawei Han and Micheline Kamber; Data Mining.

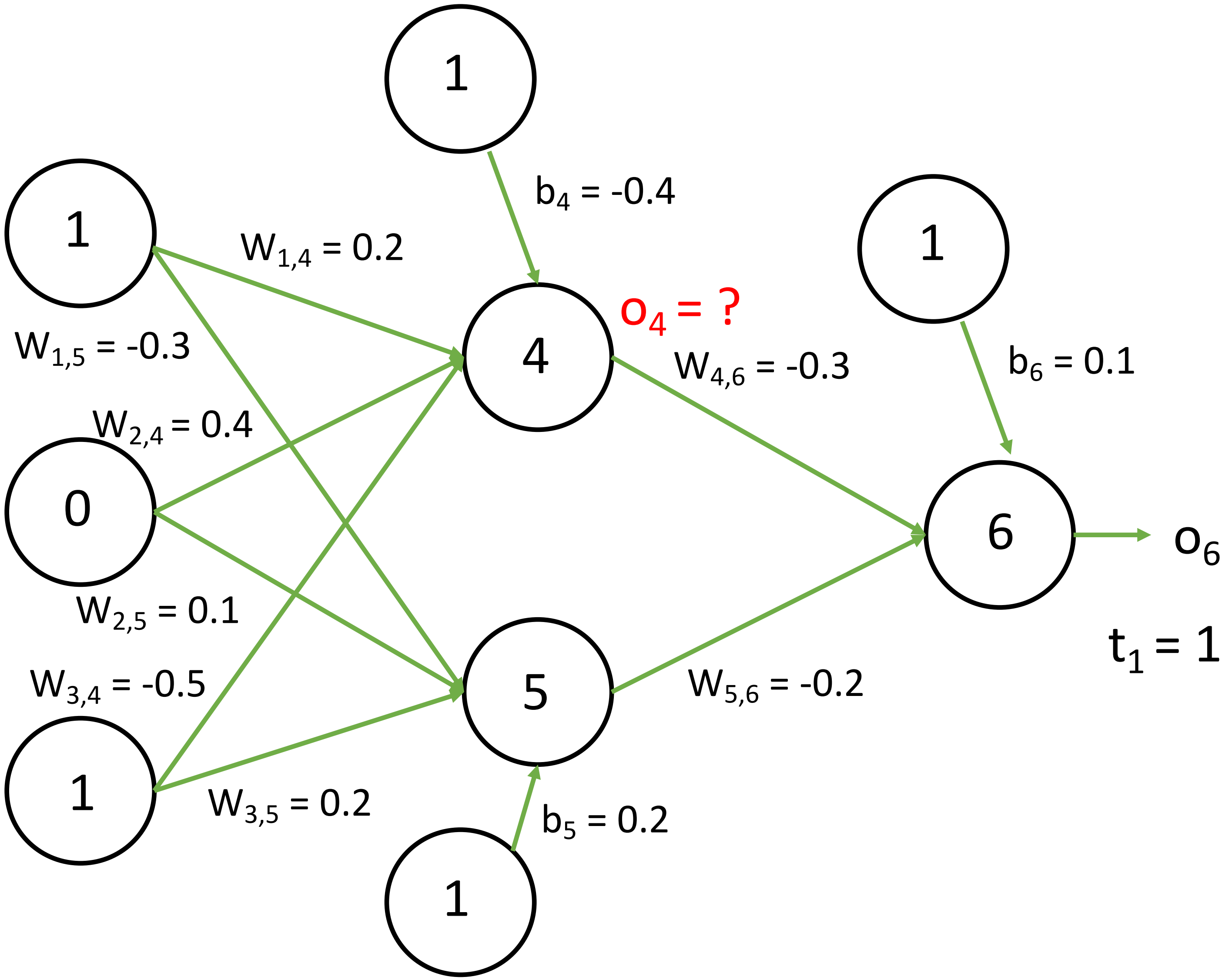
Training: How Neural Networks Learn



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Example: Step 1 – Forward Pass



Input to node 4:

$i_4 = (1 \times 0.2 + 0 \times 0.4 + 1 \times -0.5) - 0.4$

$i_4 = -0.7$

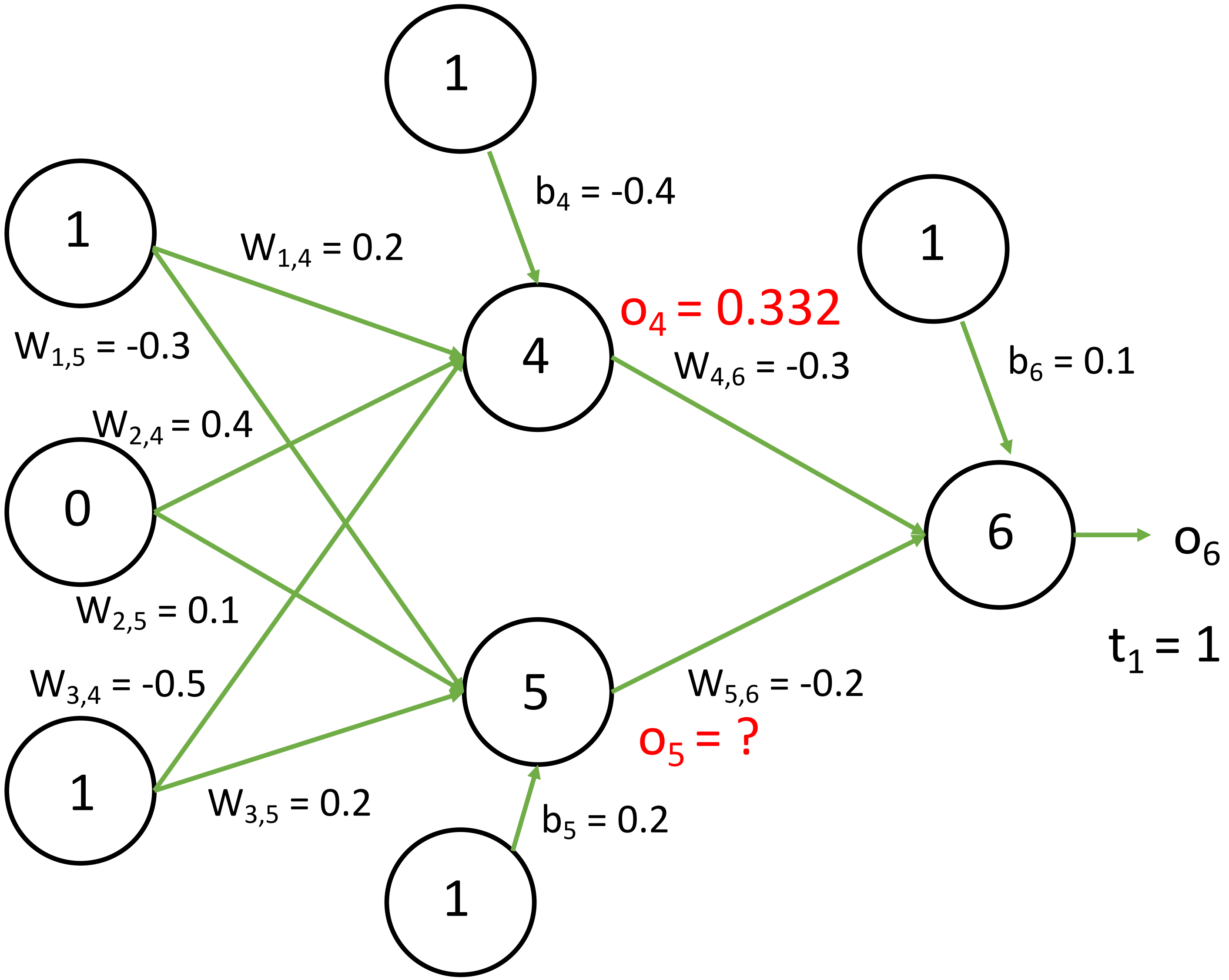
Output of node 4 (sigmoid function):

$o_4 = \textit{sigmoid}(-0.7)$

$o_4 = 1/(1+e^{-(-0.7)})$

$o_4 = 0.332$

Example: Step 1 – Forward Pass



Input to node 5:

$i_5 = (1 \times -0.3 + 0 \times 0.1 + 1 \times 0.2) + 0.2$

$i_5 = 0.1$

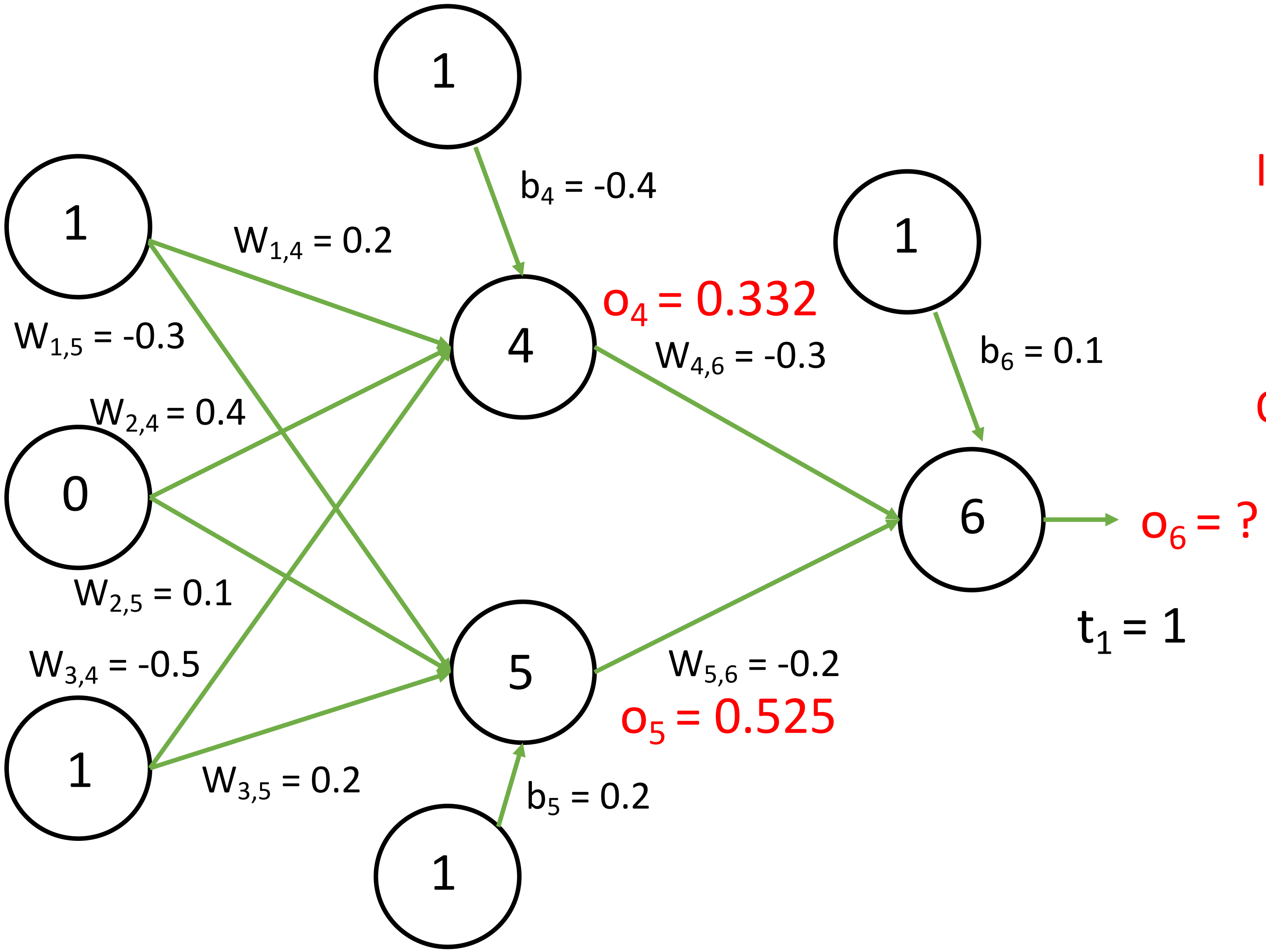
Output of node 5 (sigmoid function):

$o_5 = \textit{sigmoid}(0.1)$

$o_5 = 1/(1+e^{-0.1})$

$o_5 = 0.525$

Example: Step 1 – Forward Pass



Input to node 6:

$i_6 = (0.332 \times -0.3 + 0.525 \times -0.2) + 0.1$

$i_6 = -0.105$

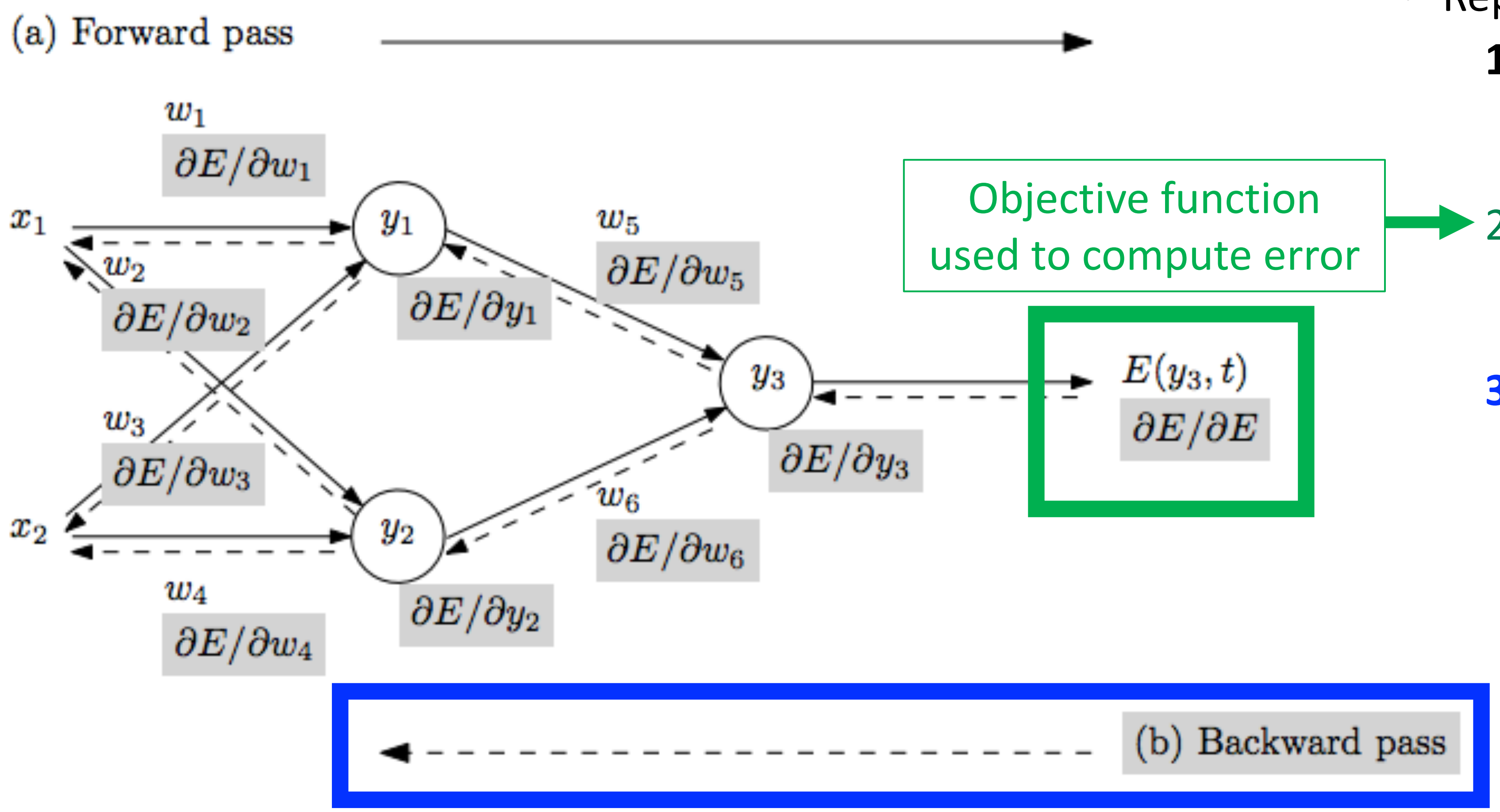
Output of node 6 (sigmoid function):

$o_6 = \textit{sigmoid}(-0.105)$

$o_6 = 1/(1+e^{-(-0.105)})$

$o_6 = 0.474$

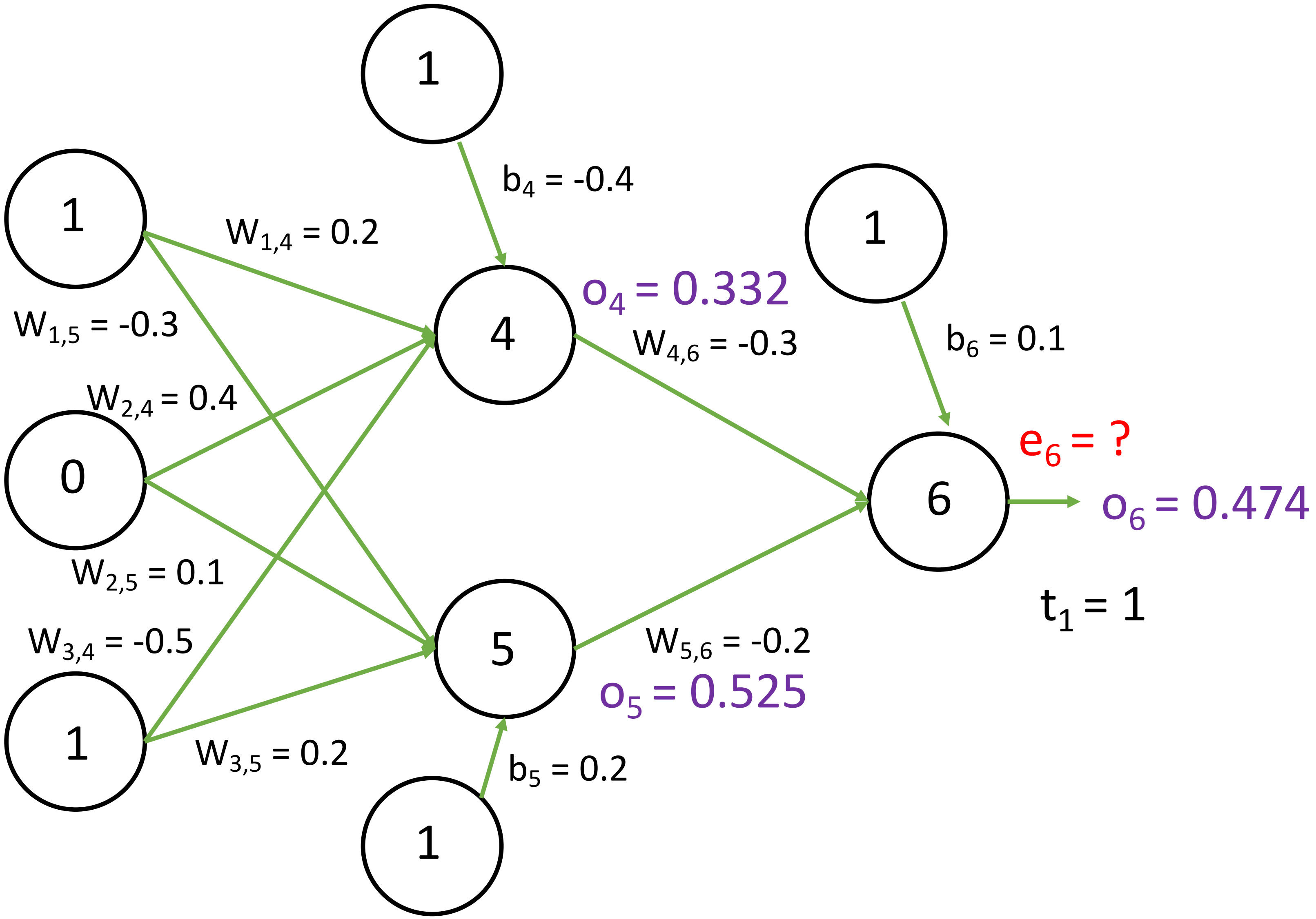
Training: How Neural Networks Learn



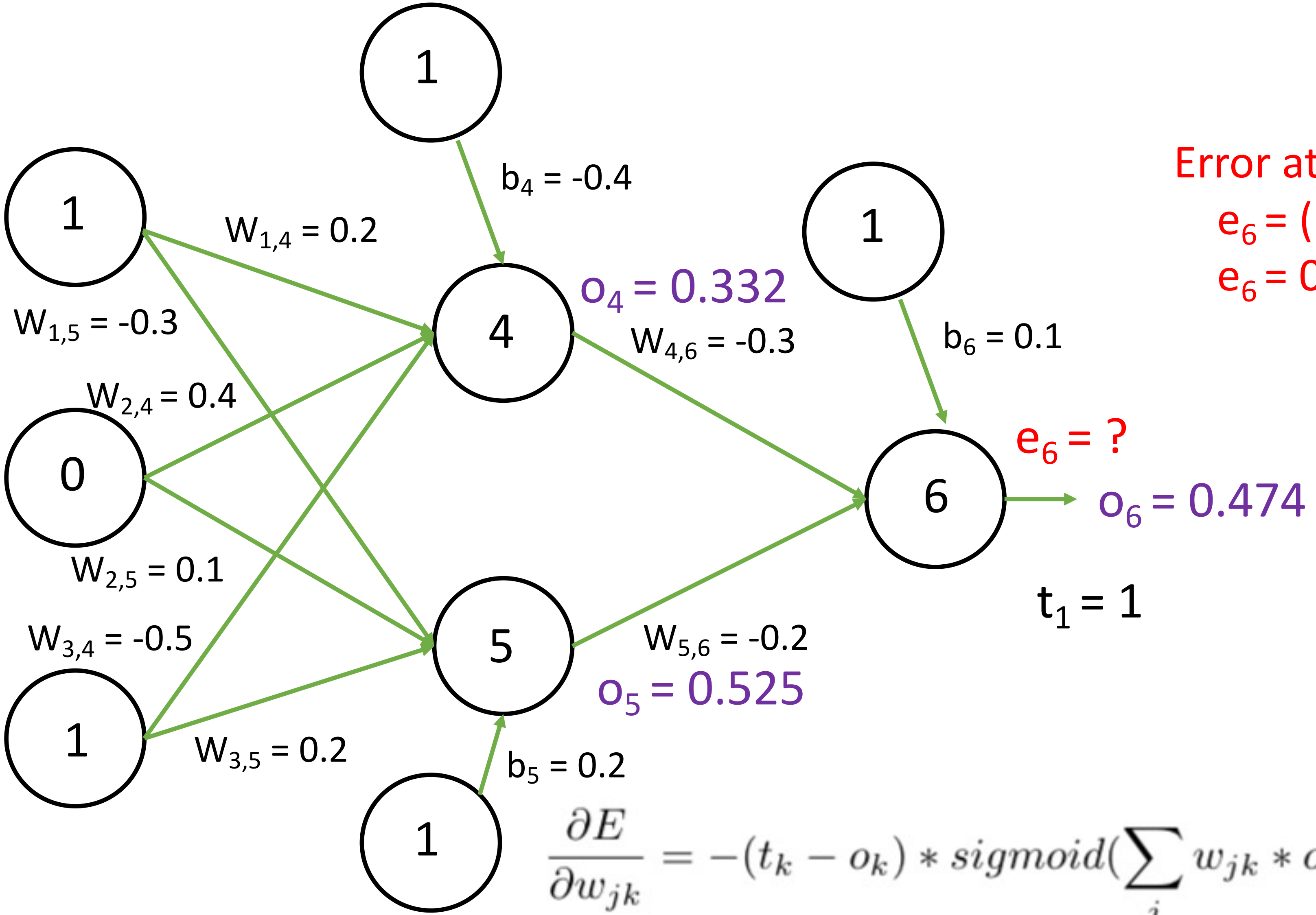
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Example: Step 2 – Backward Pass



Example: Step 2 – Backward Pass



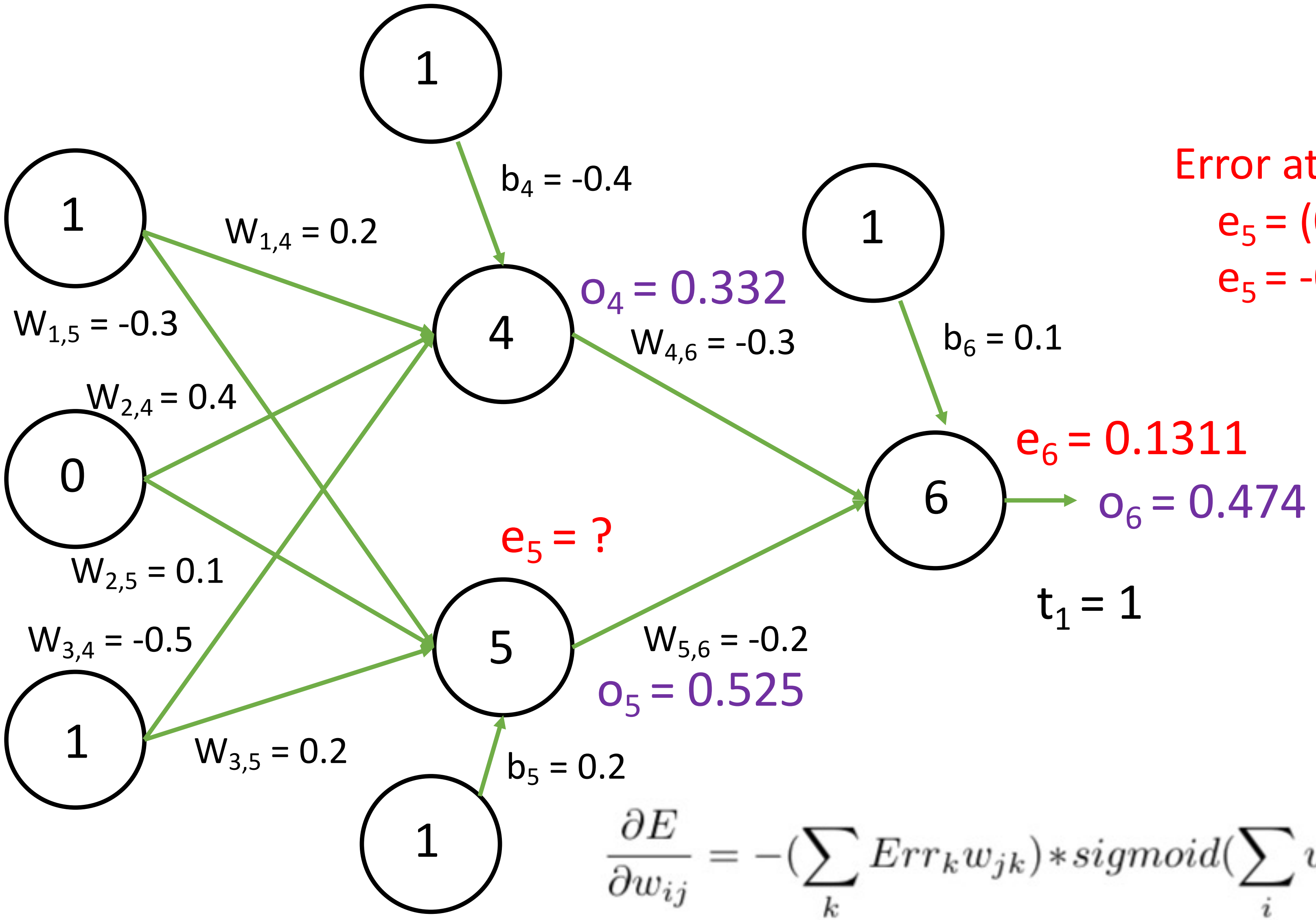
Error at node 6:

$e_6 = (1-0.474)\ (0.474)(1-0.474)$

$e_6 = 0.1311$

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) * sigmoid(\sum_j w_{jk} * o_j) * (1 - sigmoid(\sum_j w_{jk} * o_j))$$

Example: Step 2 – Backward Pass



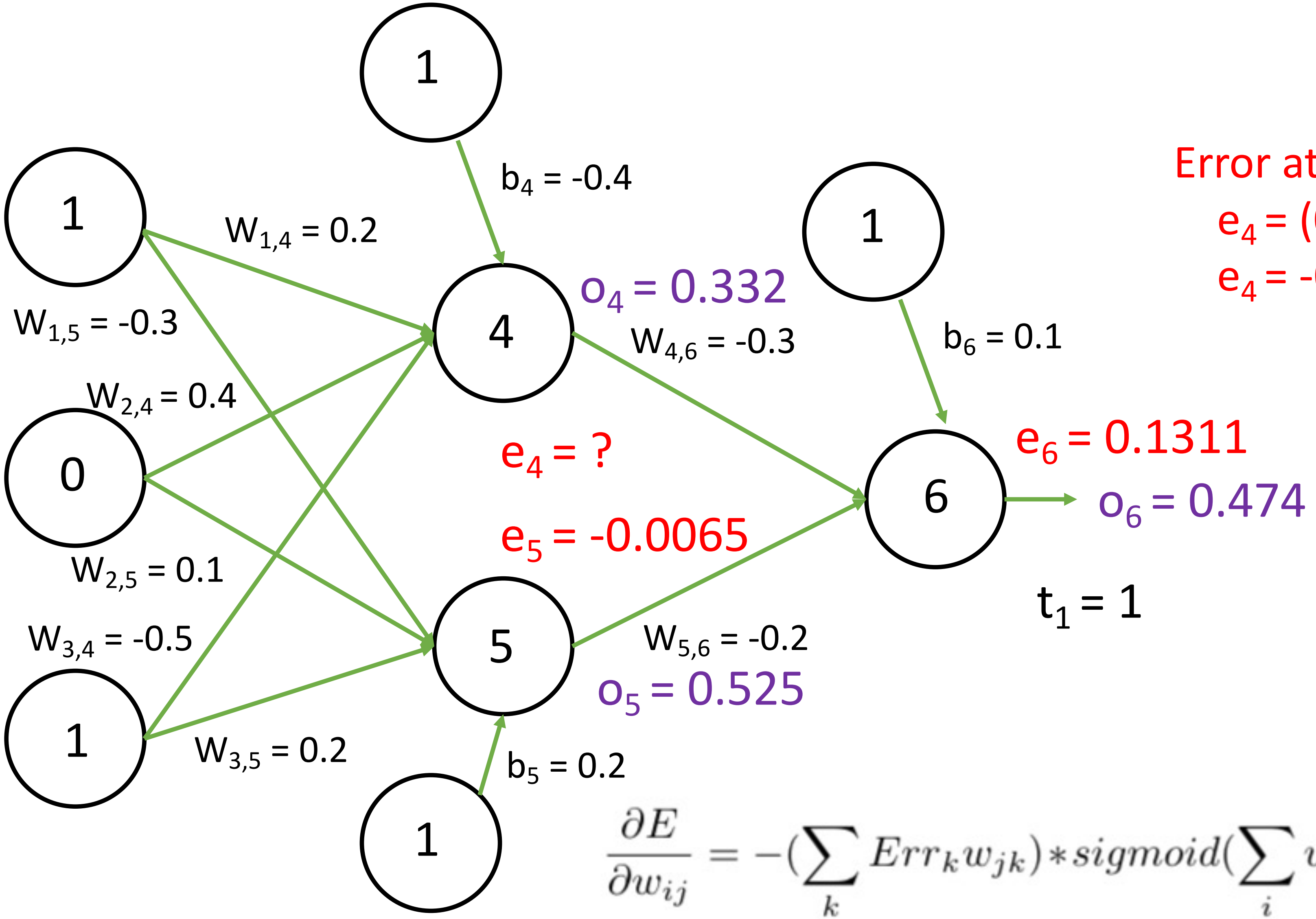
Error at node 5:

e_5 = (0.1311)(-0.2)(0.525)(1-0.525)

e_5 = -0.0065

$$\frac{\partial E}{\partial w_{ij}} = -(\sum_k Err_k w_{jk}) * sigmoid(\sum_i w_{ij} * o_i) * (1 - sigmoid(\sum_i w_{ij} * o_i))$$

Example: Step 2 – Backward Pass



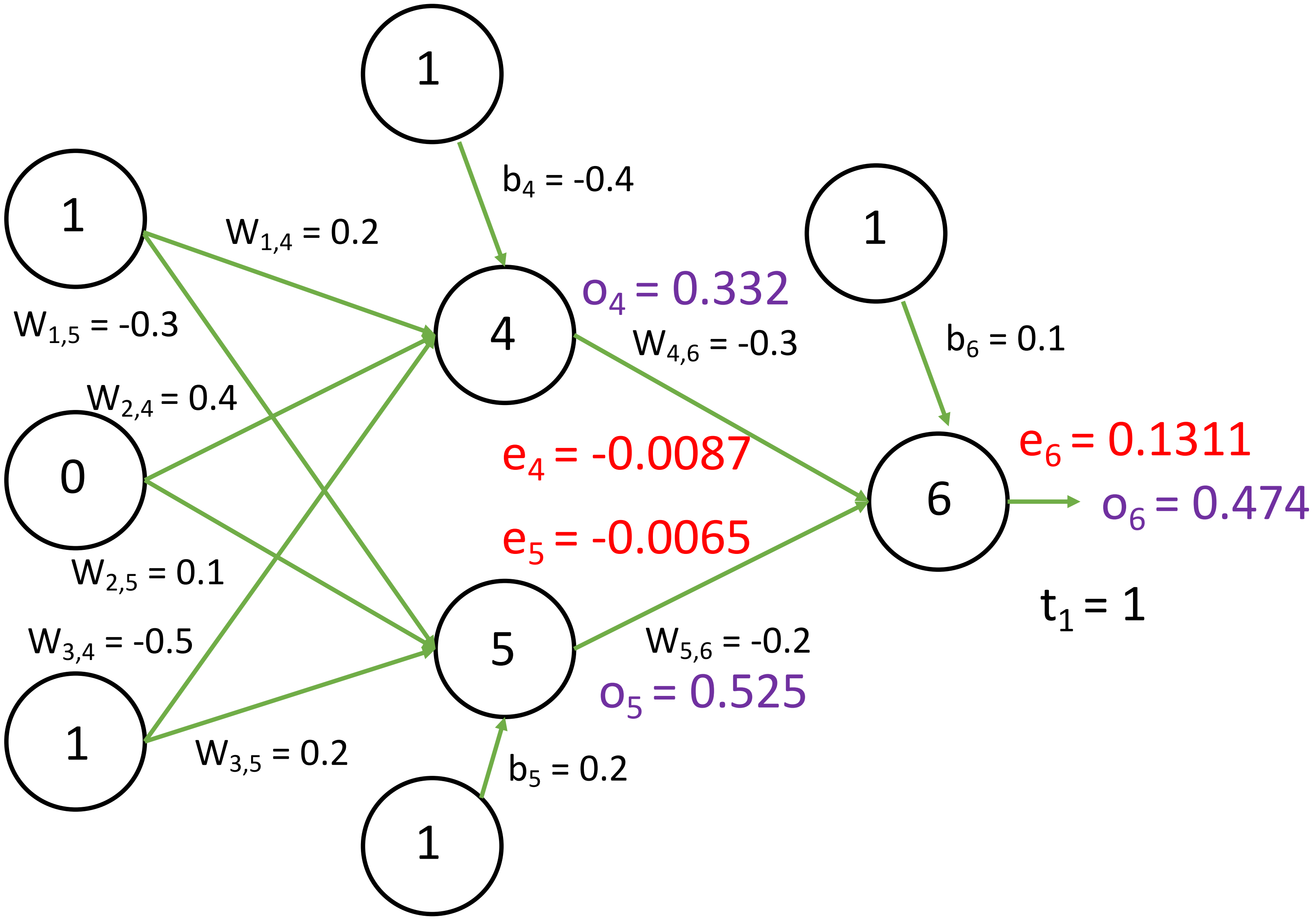
Error at node 4:

$e_4 = (0.1311)(-0.3)(0.332)(1-0.332)$

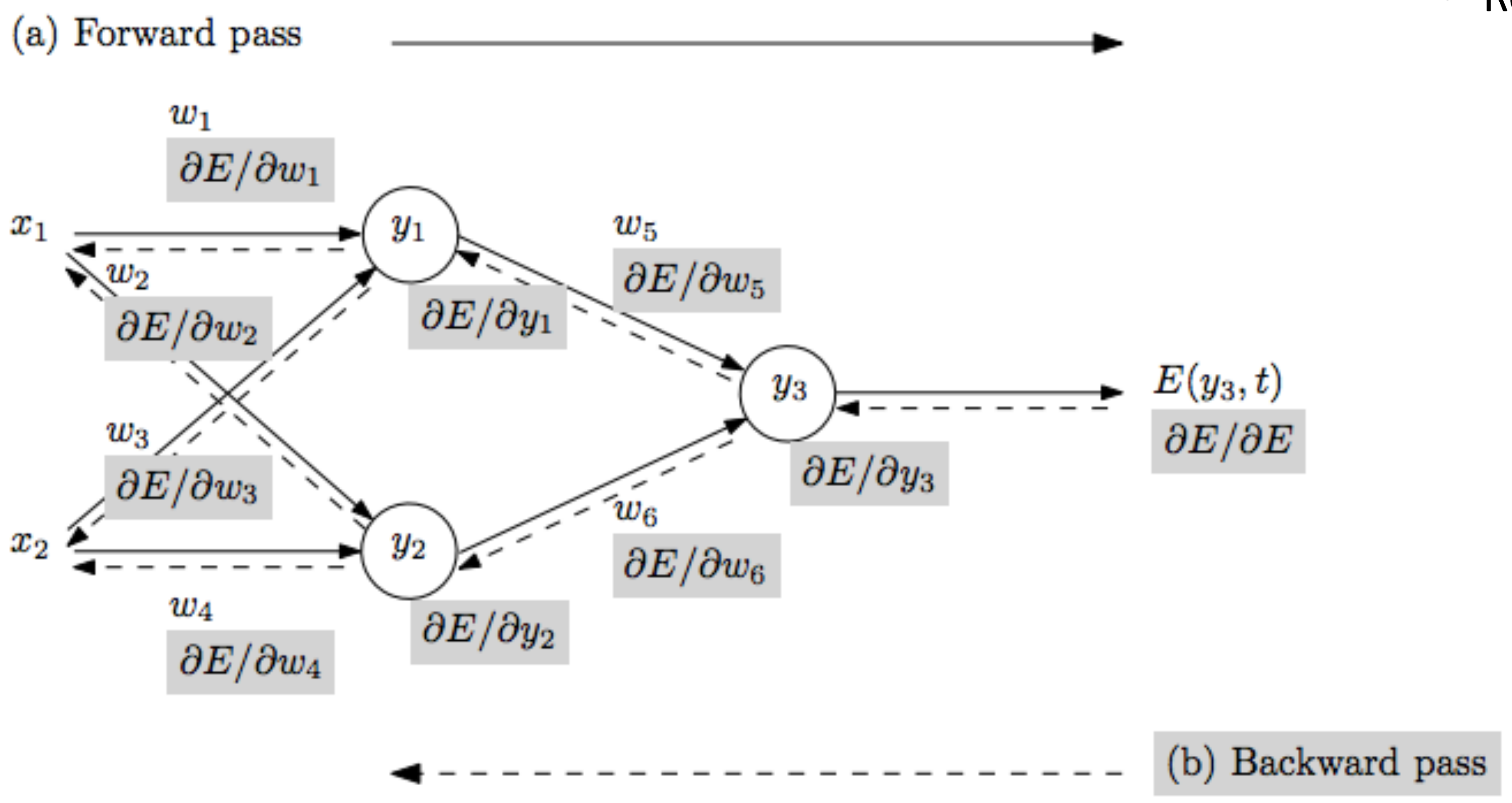
$e_4 = -0.0087$

$$\frac{\partial E}{\partial w_{ij}} = -(\sum_k Err_k w_{jk}) * sigmoid(\sum_i w_{ij} * o_i) * (1 - sigmoid(\sum_i w_{ij} * o_i))$$

Example: Step 2 – Backward Pass



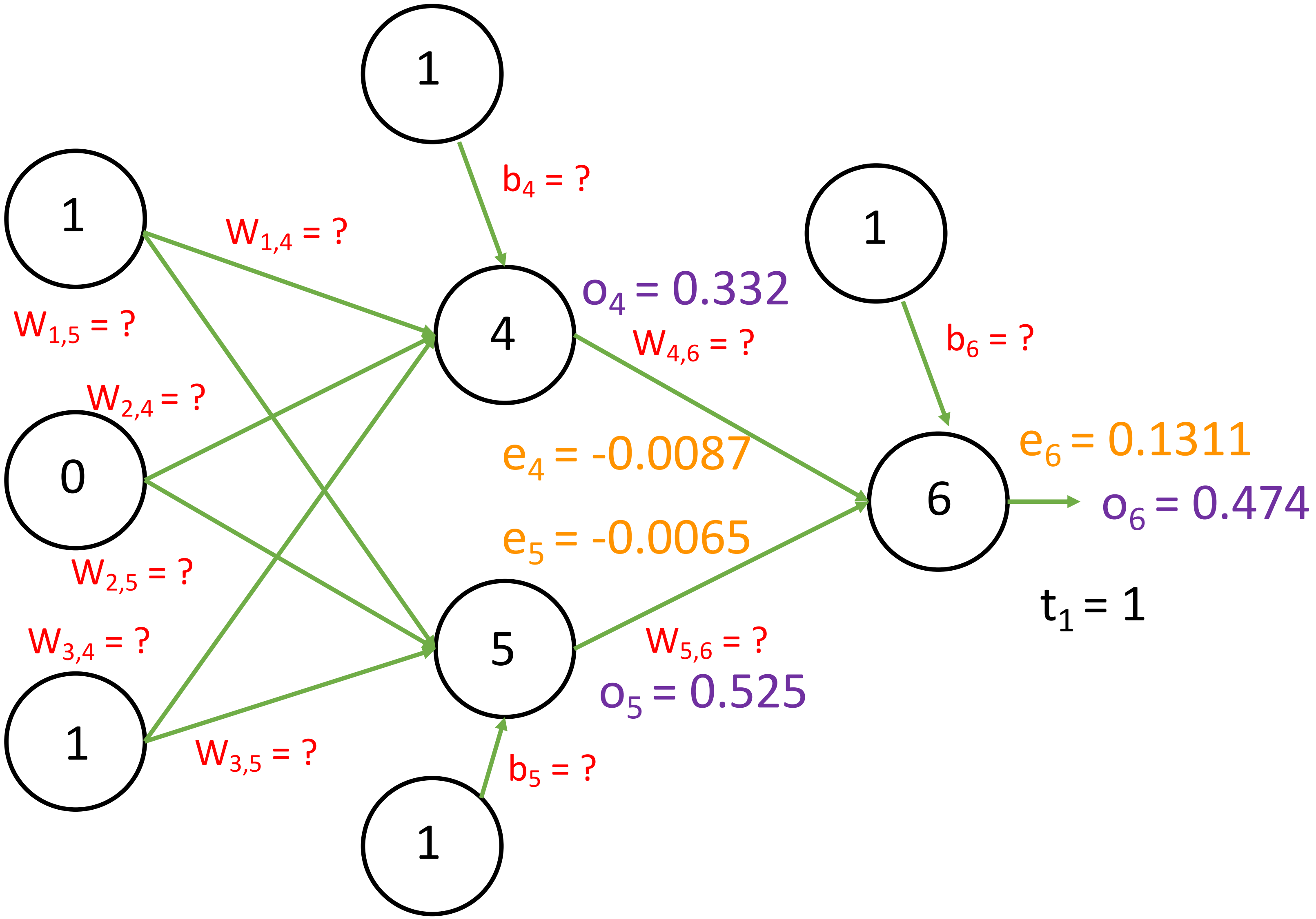
Training: How Neural Networks Learn



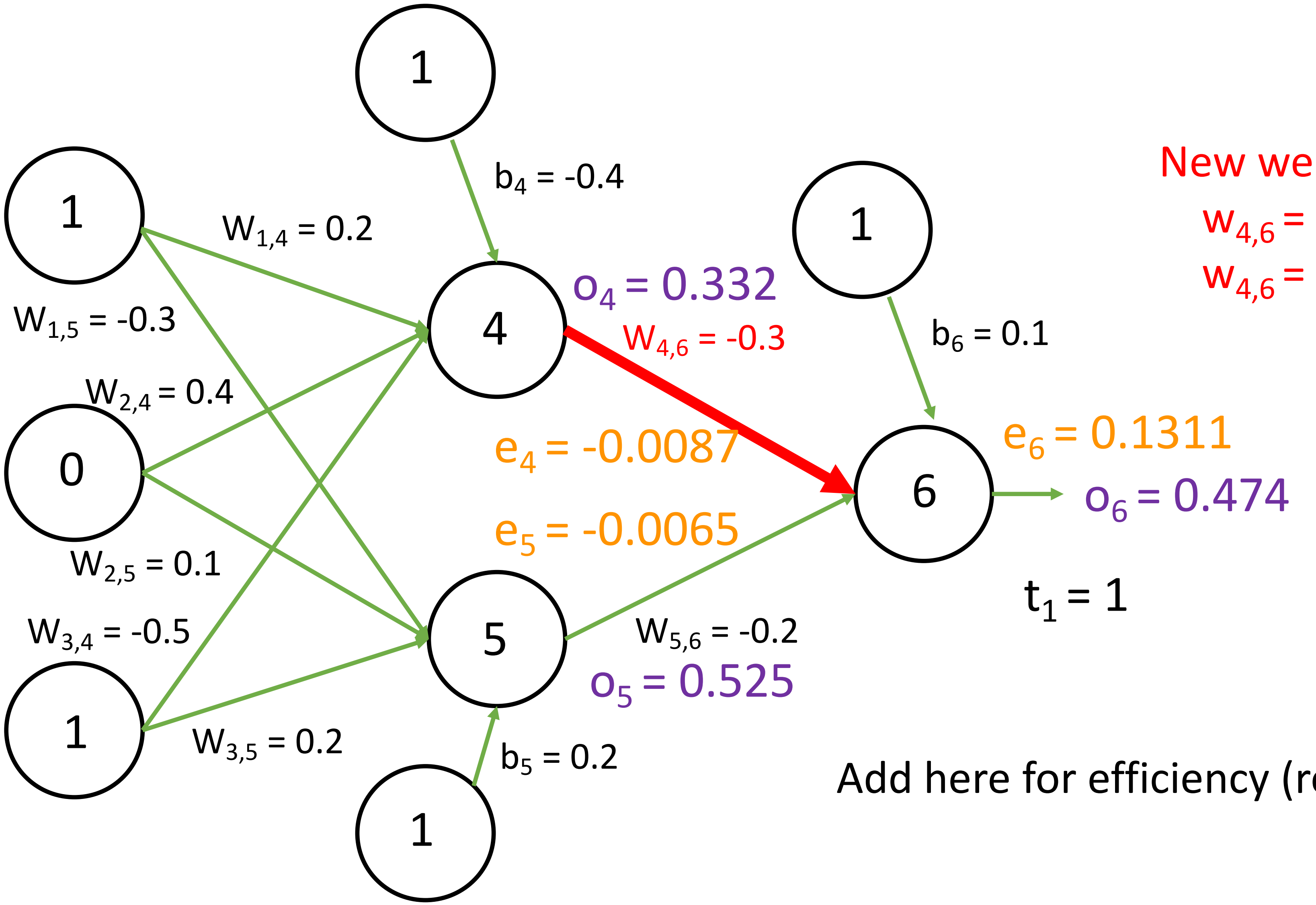
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Example: Step 3 – Update Weights



Example: Step 3 – Update Weights



New weights (learning rate = 0.9):

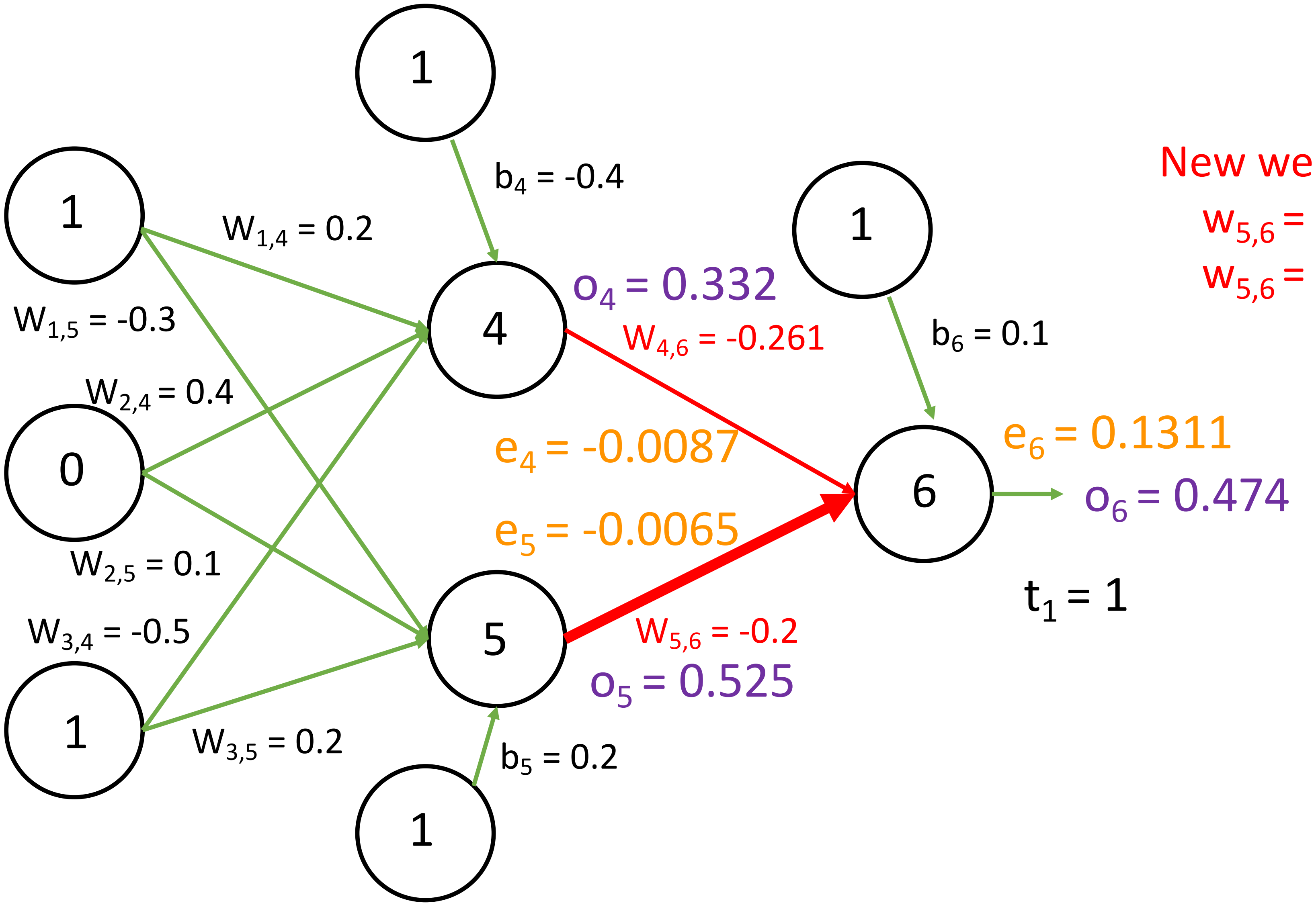
$w_{4,6} = -0.3 + (0.9)(0.1311)(0.332)$

$w_{4,6} = -0.261$

Add here for efficiency (removed from earlier equation)

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



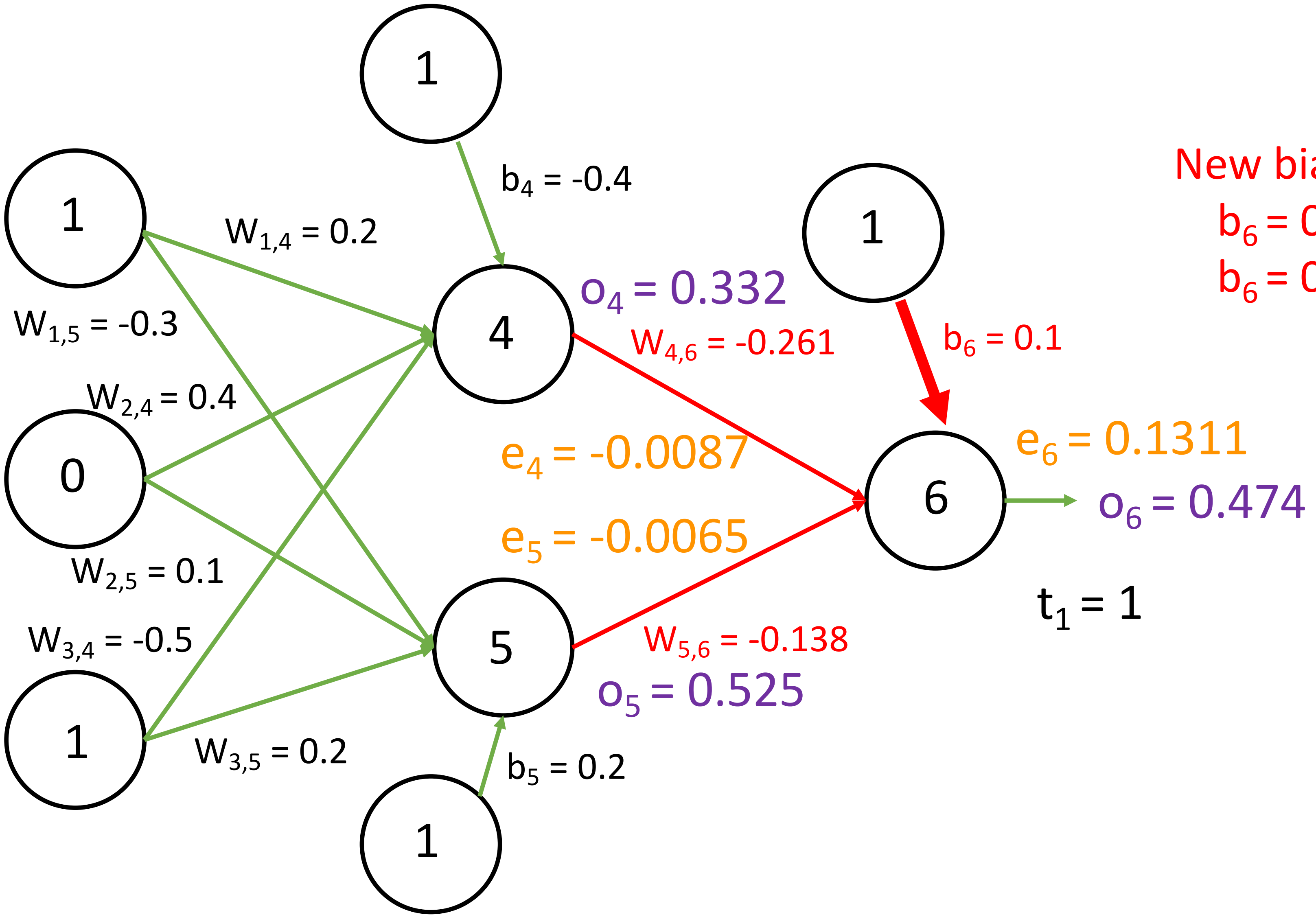
New weights (learning rate = 0.9):

$w_{5,6} = -0.2 + (0.9)(0.1311)(0.525)$

$w_{5,6} = -0.138$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



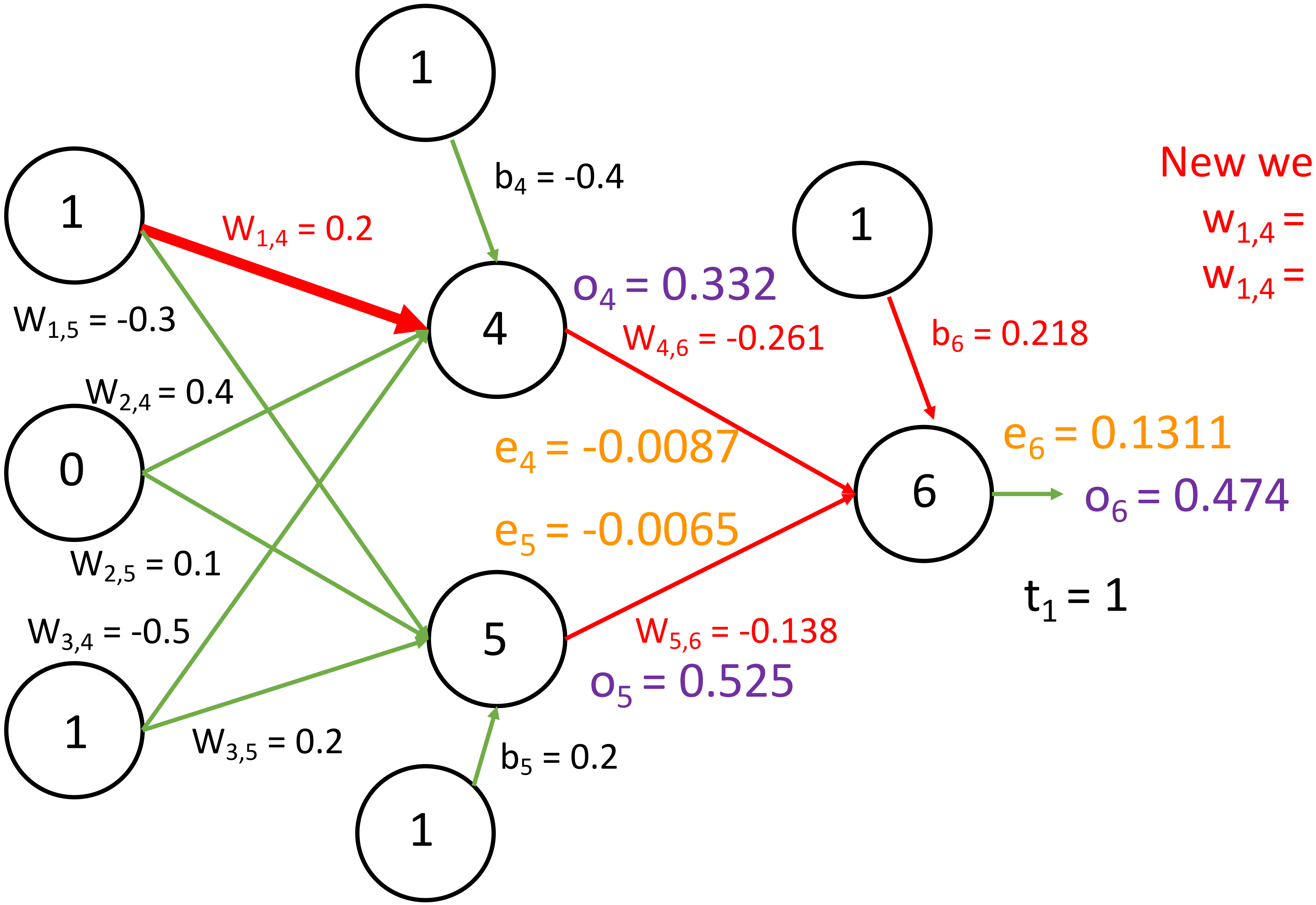
New bias (learning rate = 0.9):

$b_6 = 0.1 + (0.9)(0.1311)$

$b_6 = 0.218$

$$b_k = b_k + \eta Err_k$$

Example: Step 3 – Update Weights



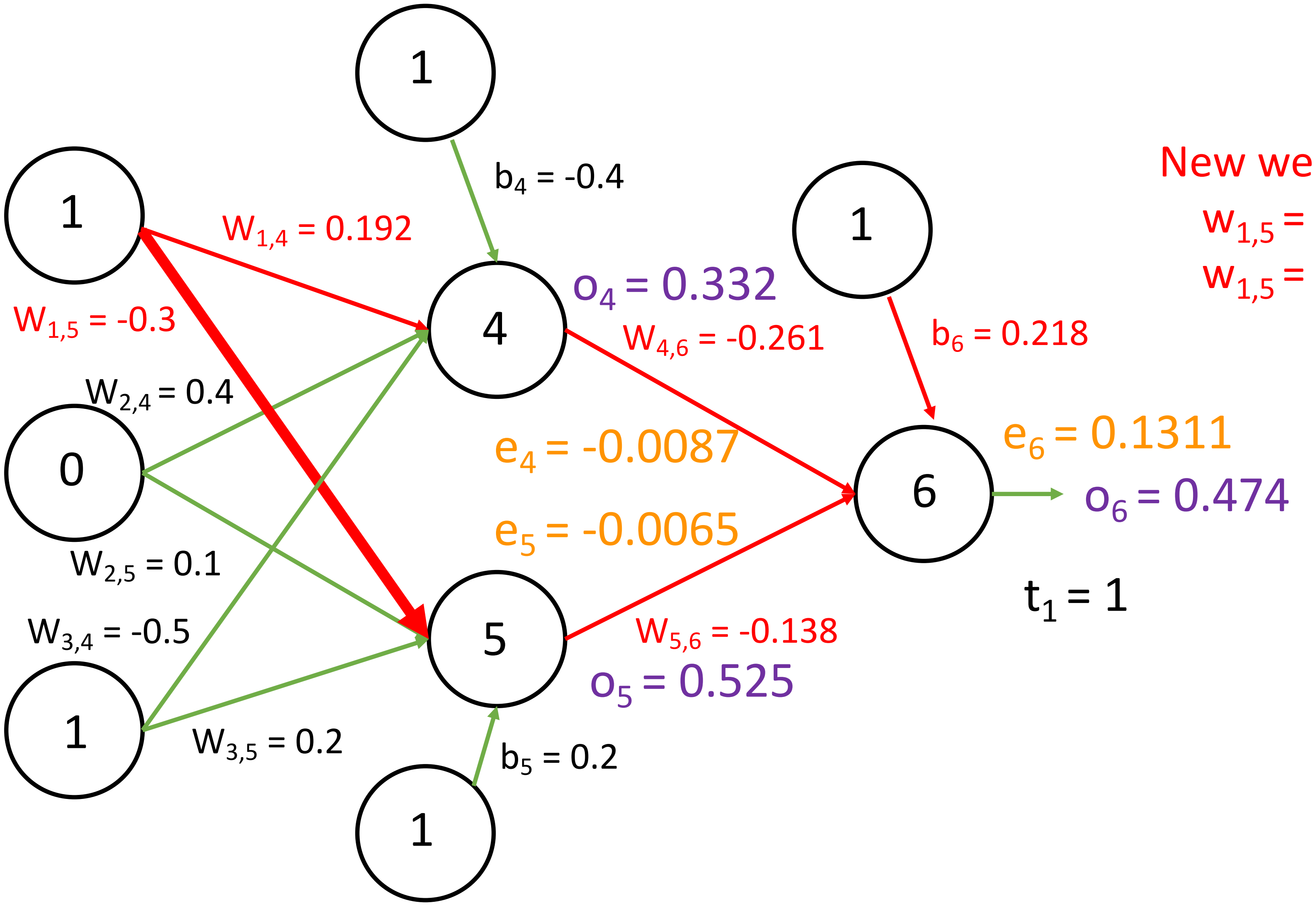
New weights (learning rate = 0.9):

$w_{1,4} = 0.2 + (0.9)(-0.0087)(1)$

$w_{1,4} = 0.192$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



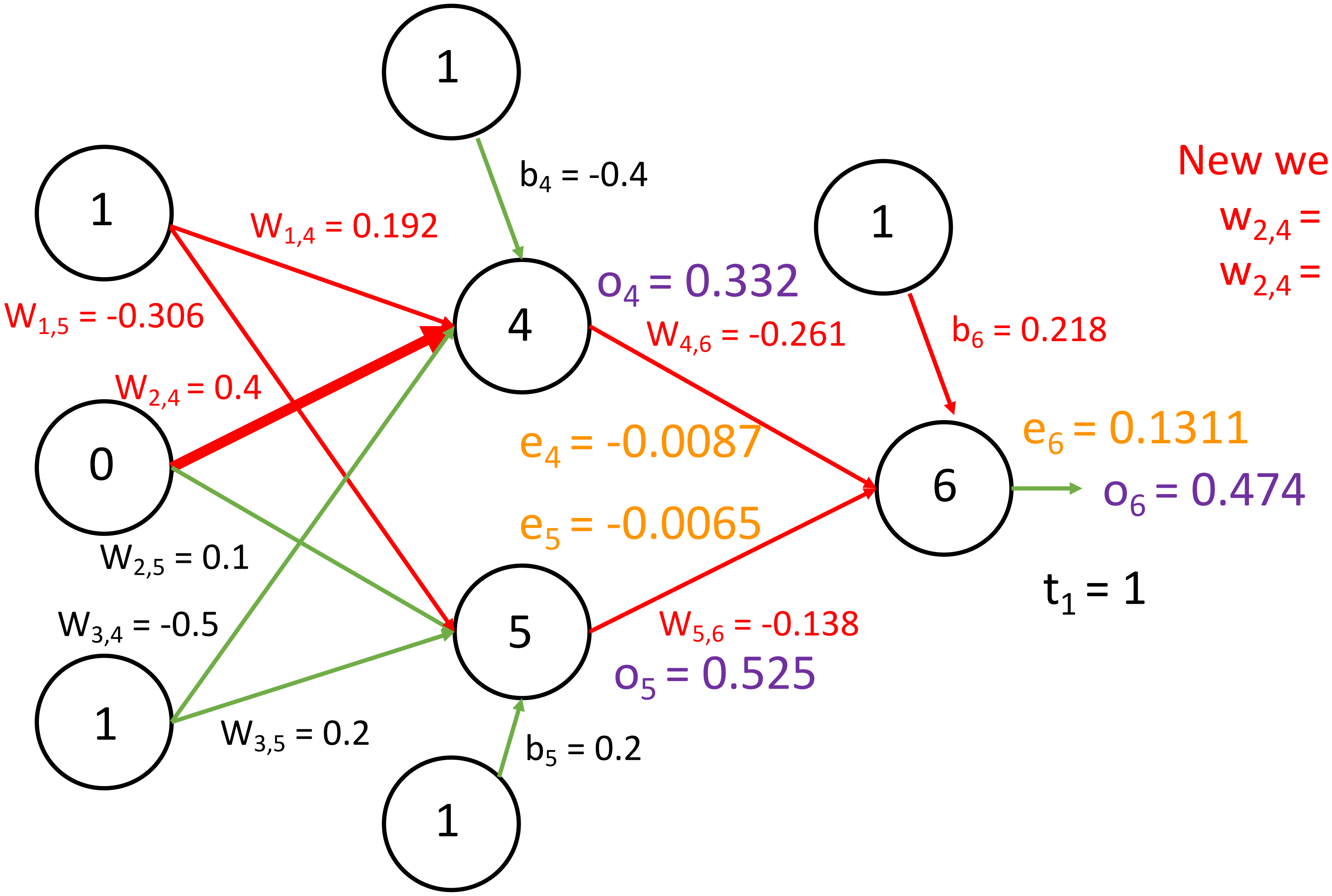
New weights (learning rate = 0.9):

$w_{1,5} = -0.3 + (0.9)(-0.0065)(1)$

$w_{1,5} = -0.306$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



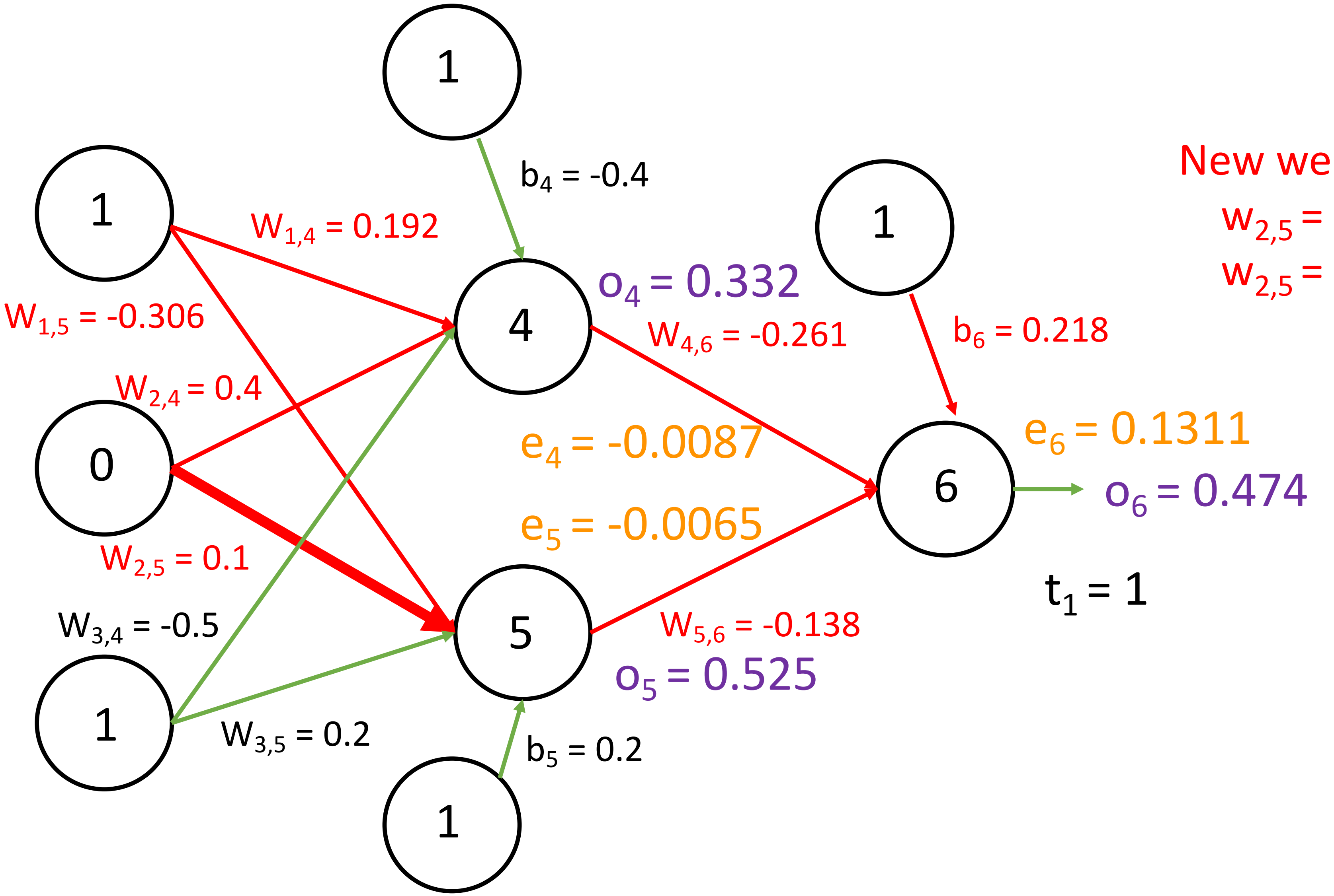
New weights (learning rate = 0.9):

$w_{2,4} = 0.4 + (0.9)(-0.0087)(0)$

$w_{2,4} = 0.4$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



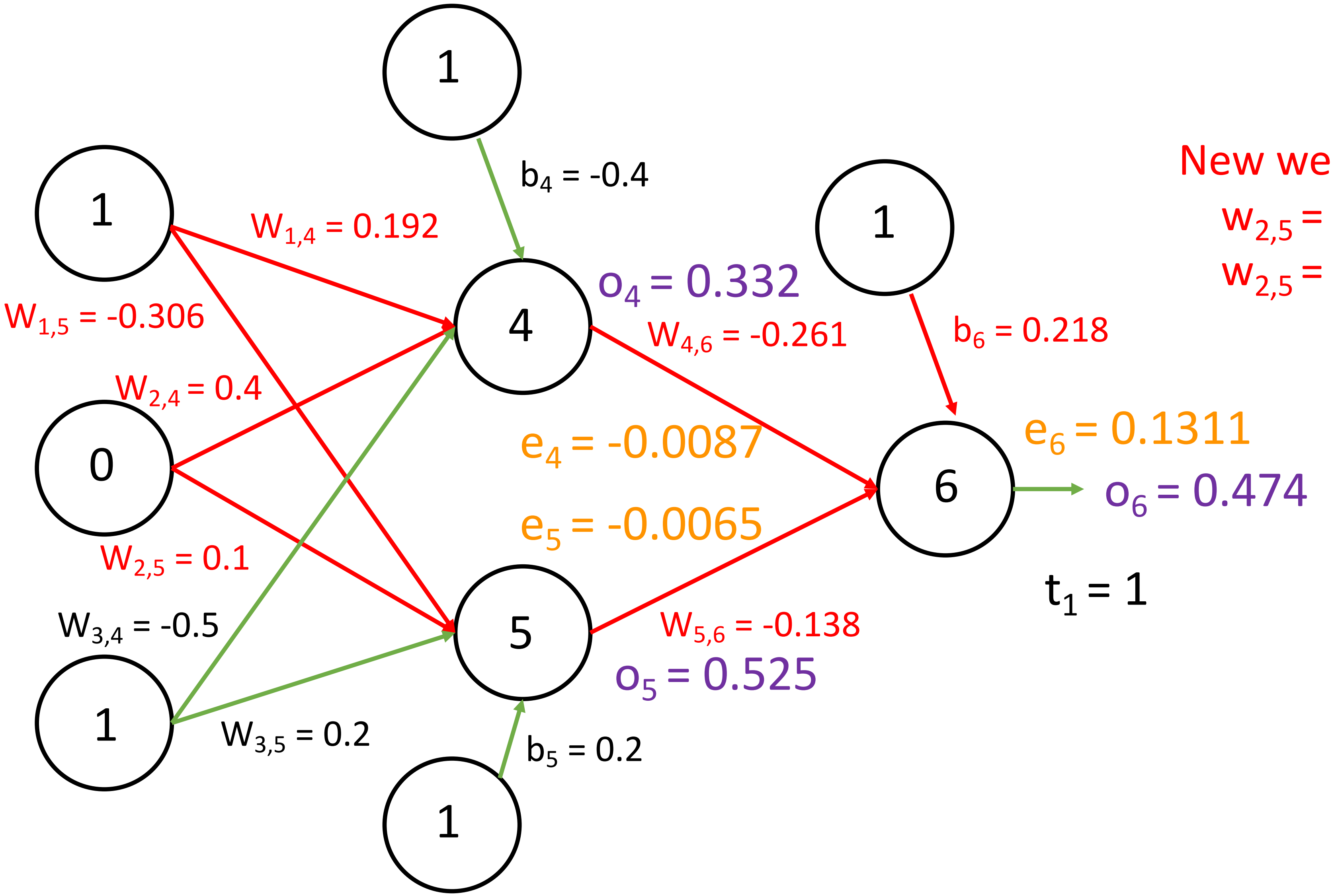
New weights (learning rate = 0.9):

$w_{2,5} = 0.1 + (0.9)(-0.0065)(0)$

$w_{2,5} = 0.1$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



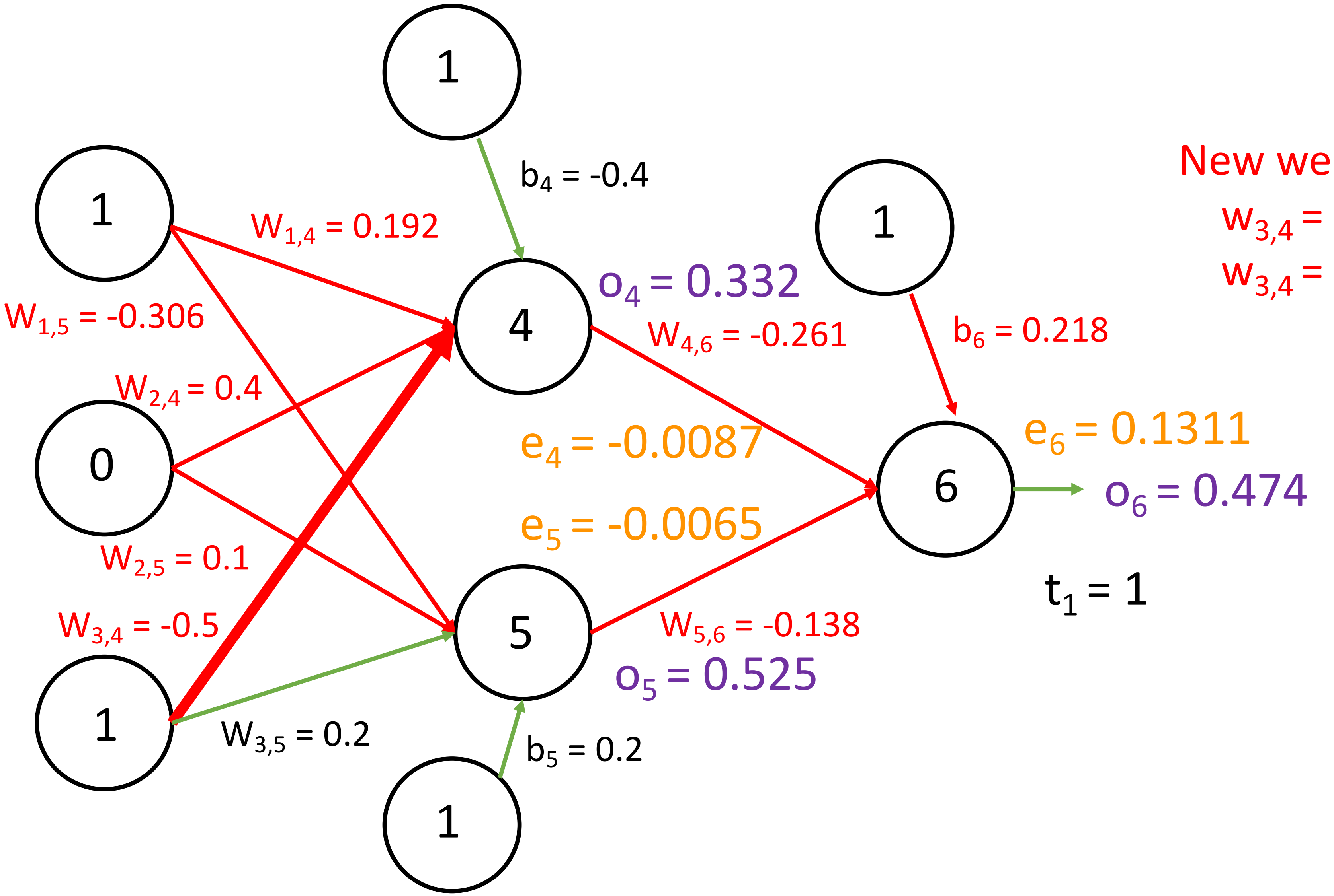
New weights (learning rate = 0.9):

$w_{2,5} = 0.1 + (0.9)(-0.0065)(0)$

$w_{2,5} = 0.1$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



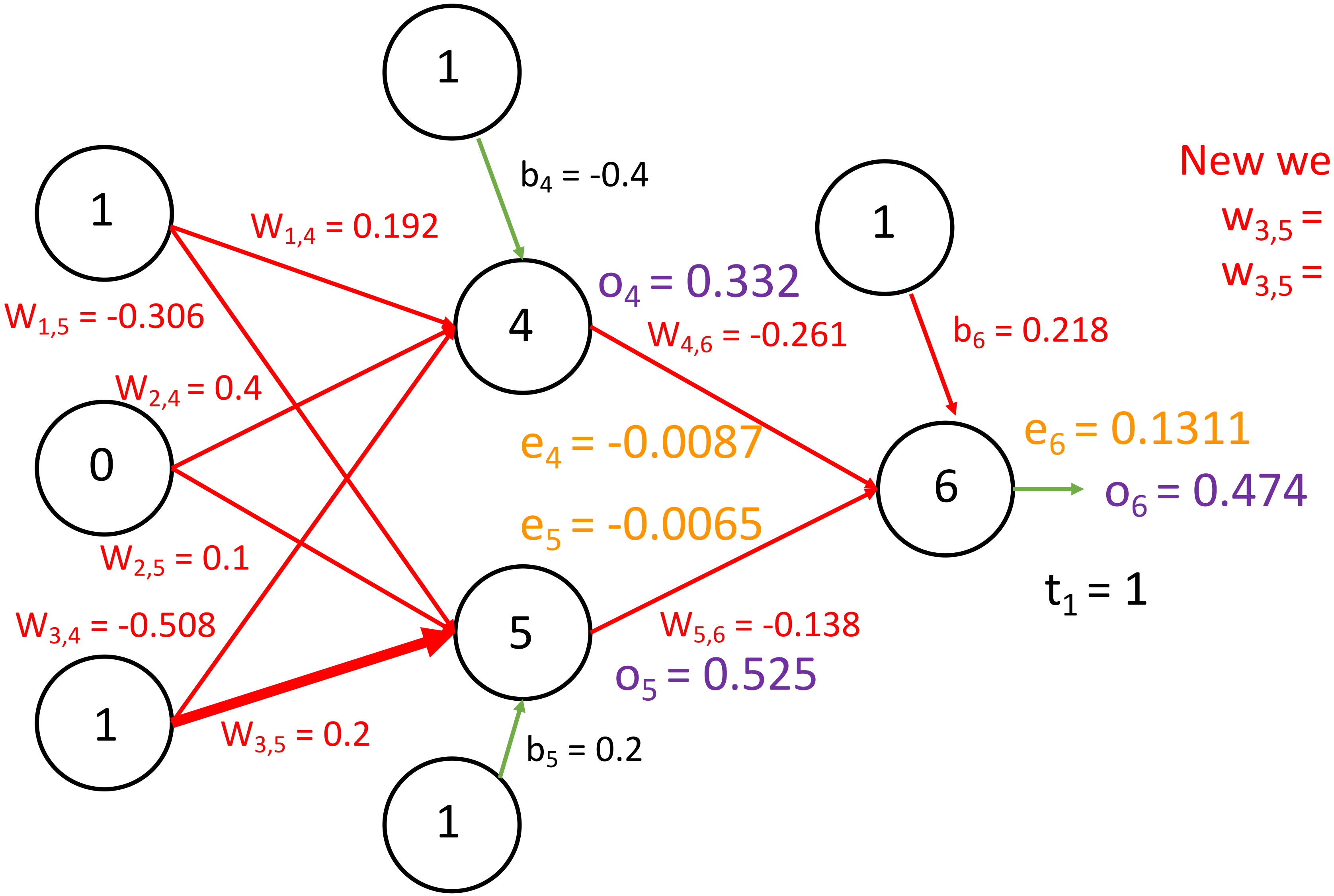
New weights (learning rate = 0.9):

$w_{3,4} = -0.5 + (0.9)(-0.0087)(1)$

$w_{3,4} = -0.508$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



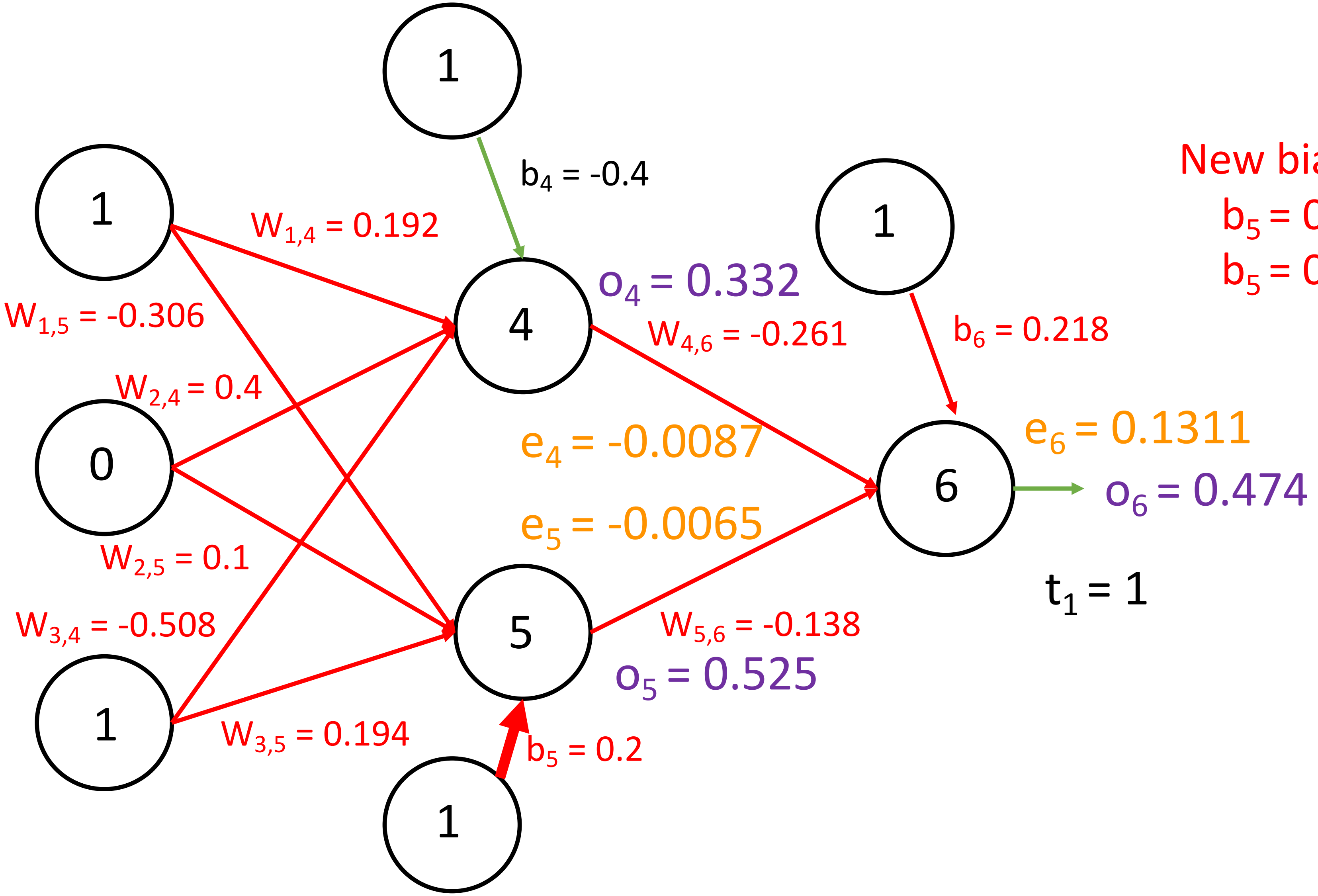
New weights (learning rate = 0.9):

$w_{3,5} = 0.2 + (0.9)(-0.0065)(1)$

$w_{3,5} = 01.94$

$$w_{jk} = w_{jk} + \eta Err_k o_j$$

Example: Step 3 – Update Weights



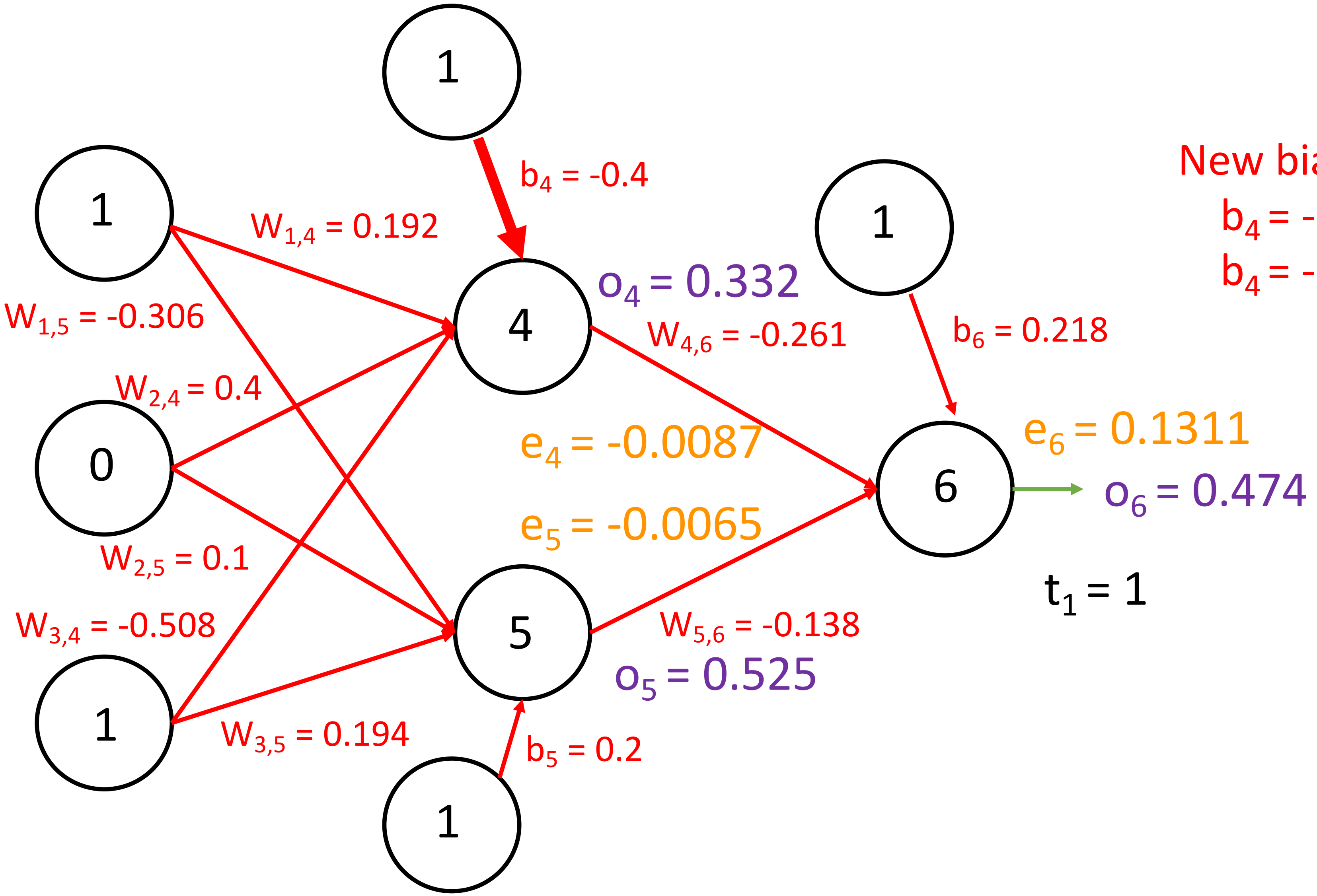
New bias (learning rate = 0.9):

$b_5 = 0.2 + (0.9)(-0.0065)$

$b_5 = 0.194$

$$b_k = b_k + \eta Err_k$$

Example: Step 3 – Update Weights



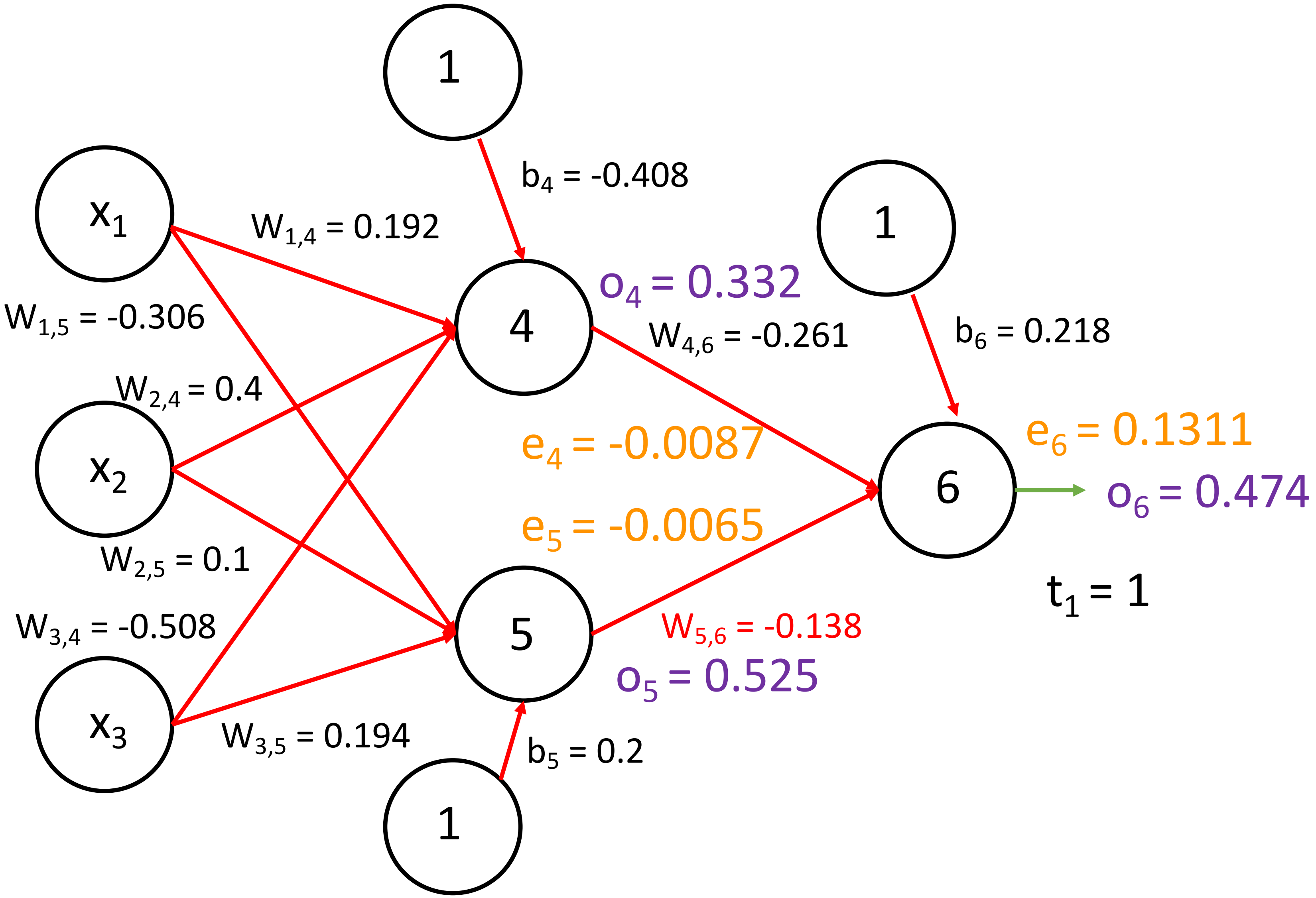
New bias (learning rate = 0.9):

$b_4 = -0.4 + (0.9)(-0.0087)$

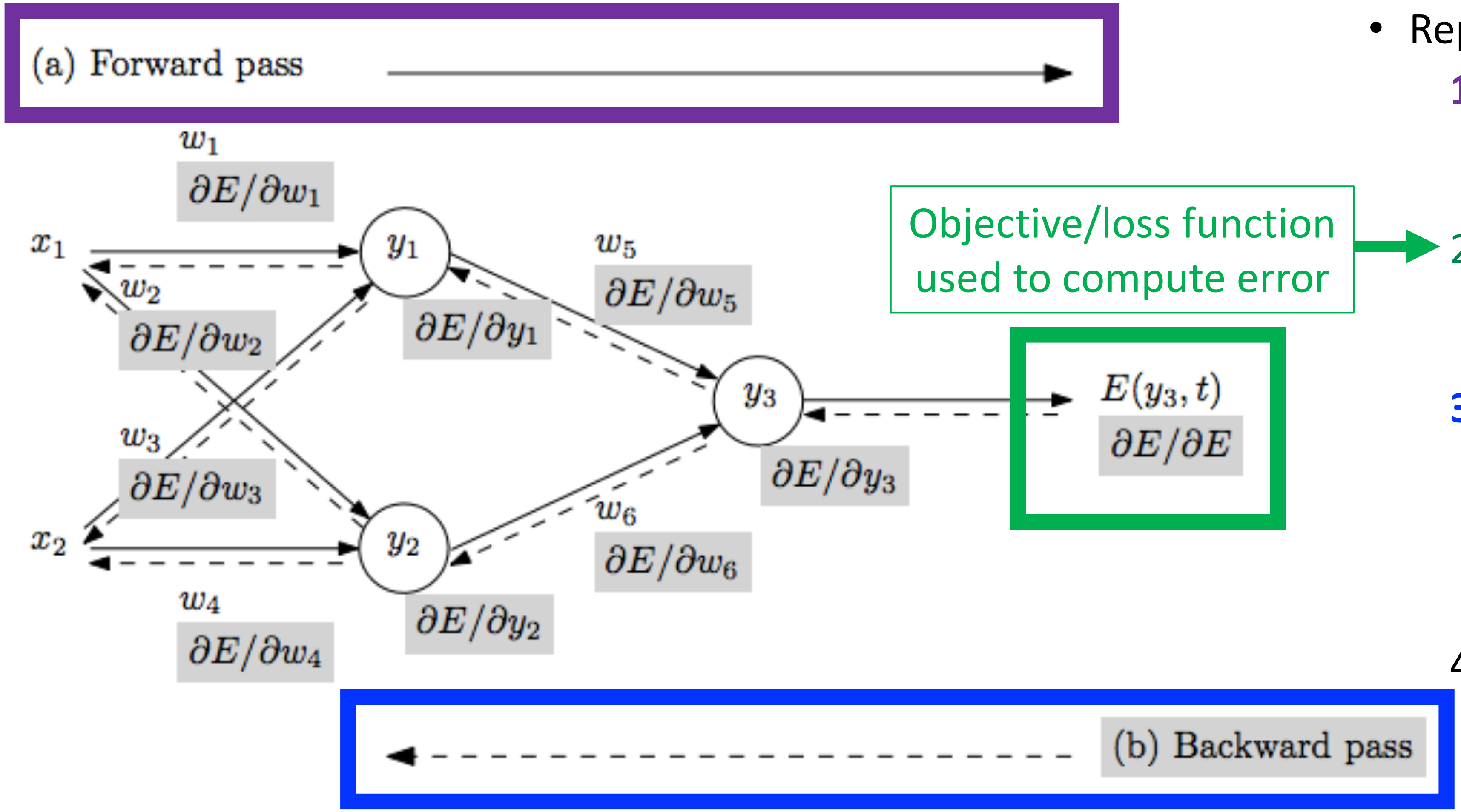
$b_4 = -0.408$

$$b_k = b_k + \eta Err_k$$

Repeat Steps 1-3 With New Examples



Repeat Steps 1-4 With New Examples



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Repeat Steps 1-4 With New Examples

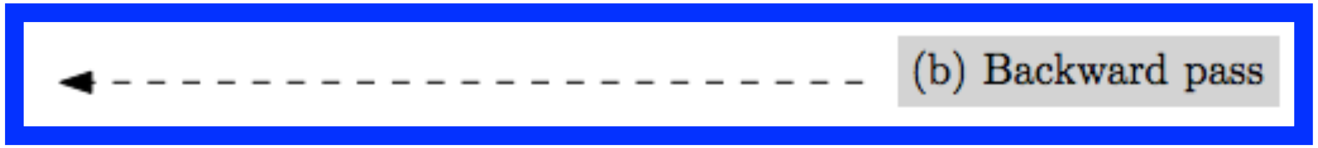
What type of gradient descent was used in the toy example?

a. Batch gradient descent
b. Stochastic gradient descent
c. Mini-batch gradient descent

- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Training: How Neural Networks Learn

The mean gradient is used for batch and mini-batch gradient descent

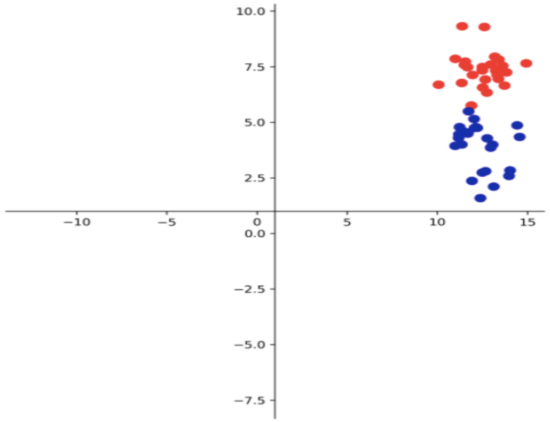


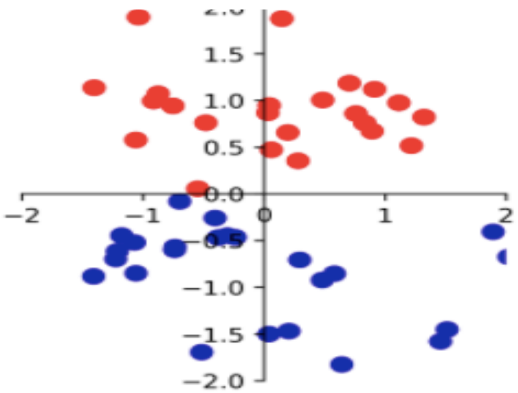
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

* Practical Detail (More in Future Lectures)

Basic data initialization approach:
- standardize so mean is 0 and standard deviation 1
- simplifies learning

Original data: 

Standardized data: 

Basic model parameter initialization:
- set weights to random values drawn from Gaussian or uniform distribution
- set biases to 0

https://github.com/amueller/introduction_to_ml_with_python/blob/master/03-unsupervised-learning.ipynb

* Practical Detail

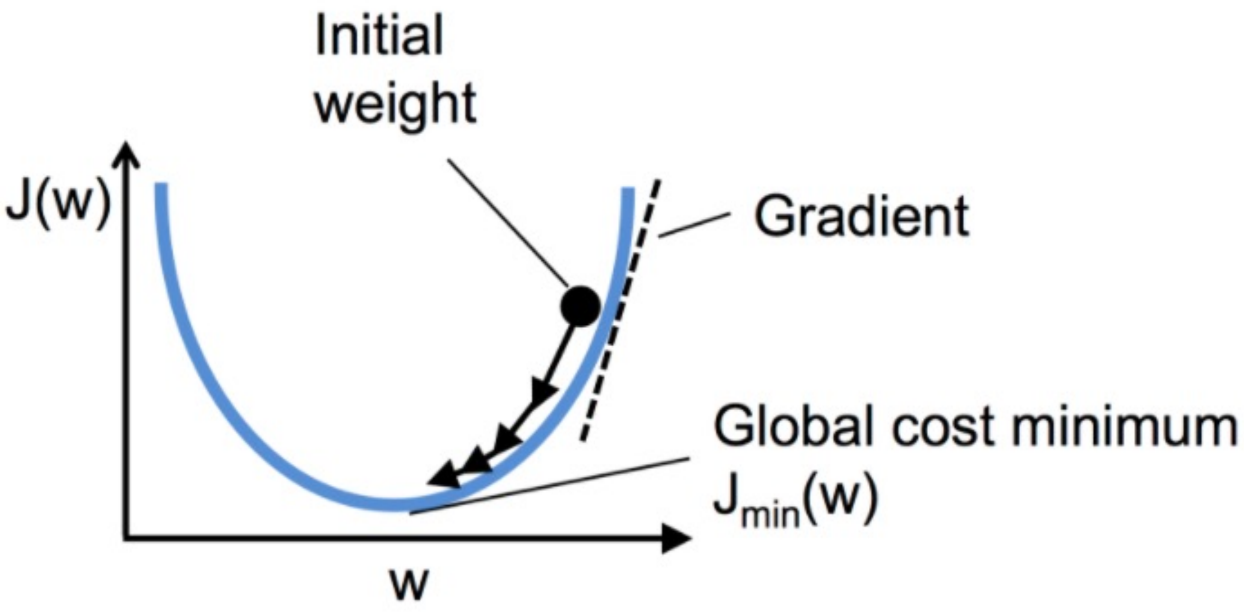
- When training neural networks, optimization function is often used interchangeably with loss function and cost function
  - Subtle nuances are discussed here: https://www.baeldung.com/cs/cost-vs-loss-vs-objective-function

Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions

Activation Function Overview

- **Want**: function with a gradient large enough to support efficient learning

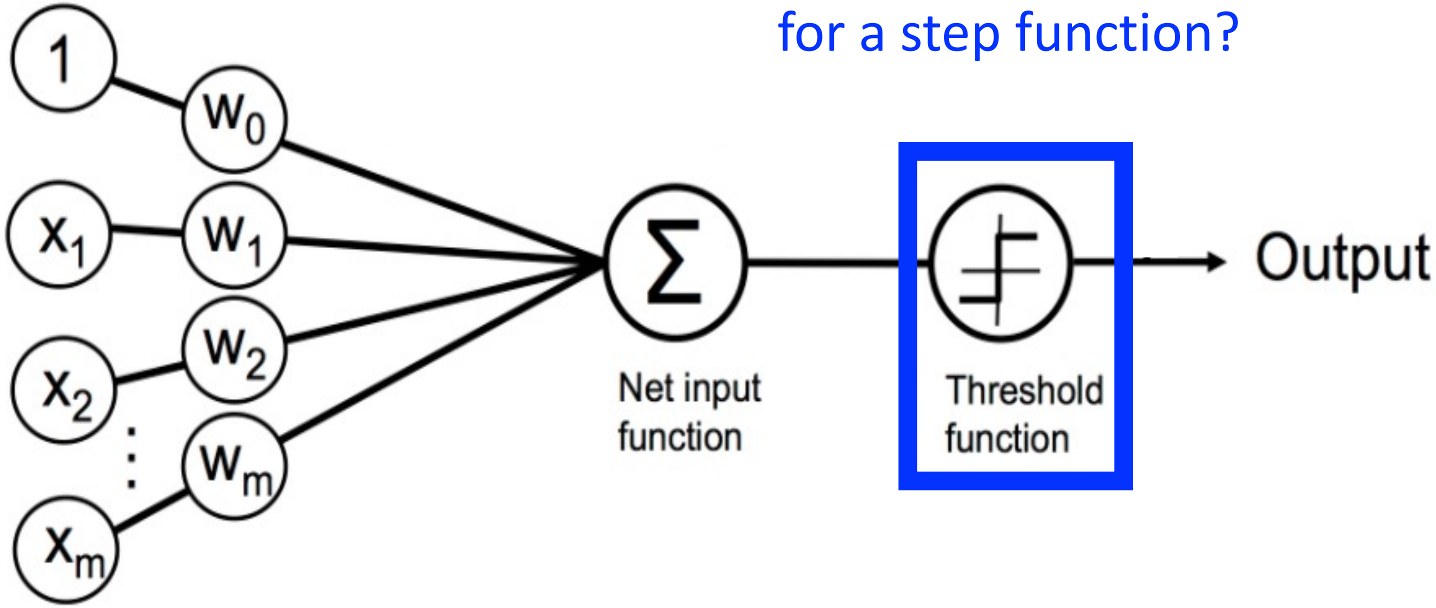


- **Implied requirement:** function should be differentiable

https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch02/ch02.ipynb

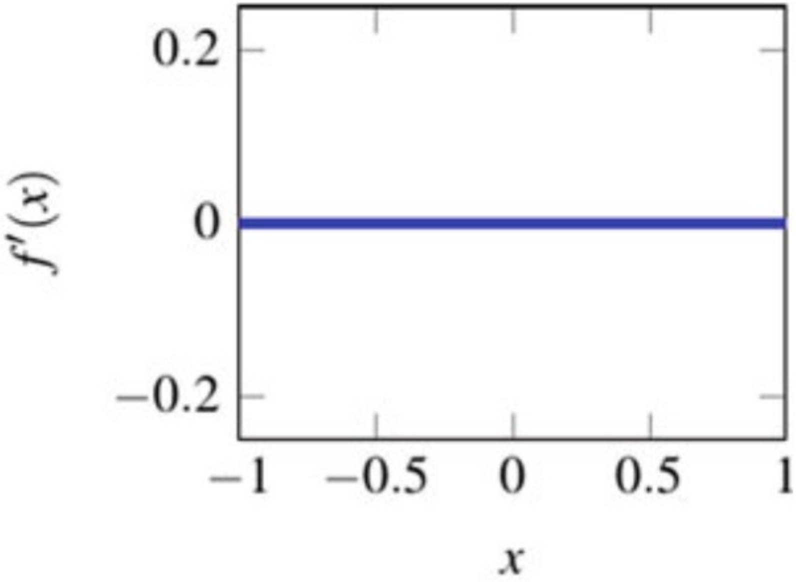
Activation Functions with Gradients: Revisiting Perceptron

What is the gradient for a step function?



Python Machine Learning; Raschka & Mirjalili

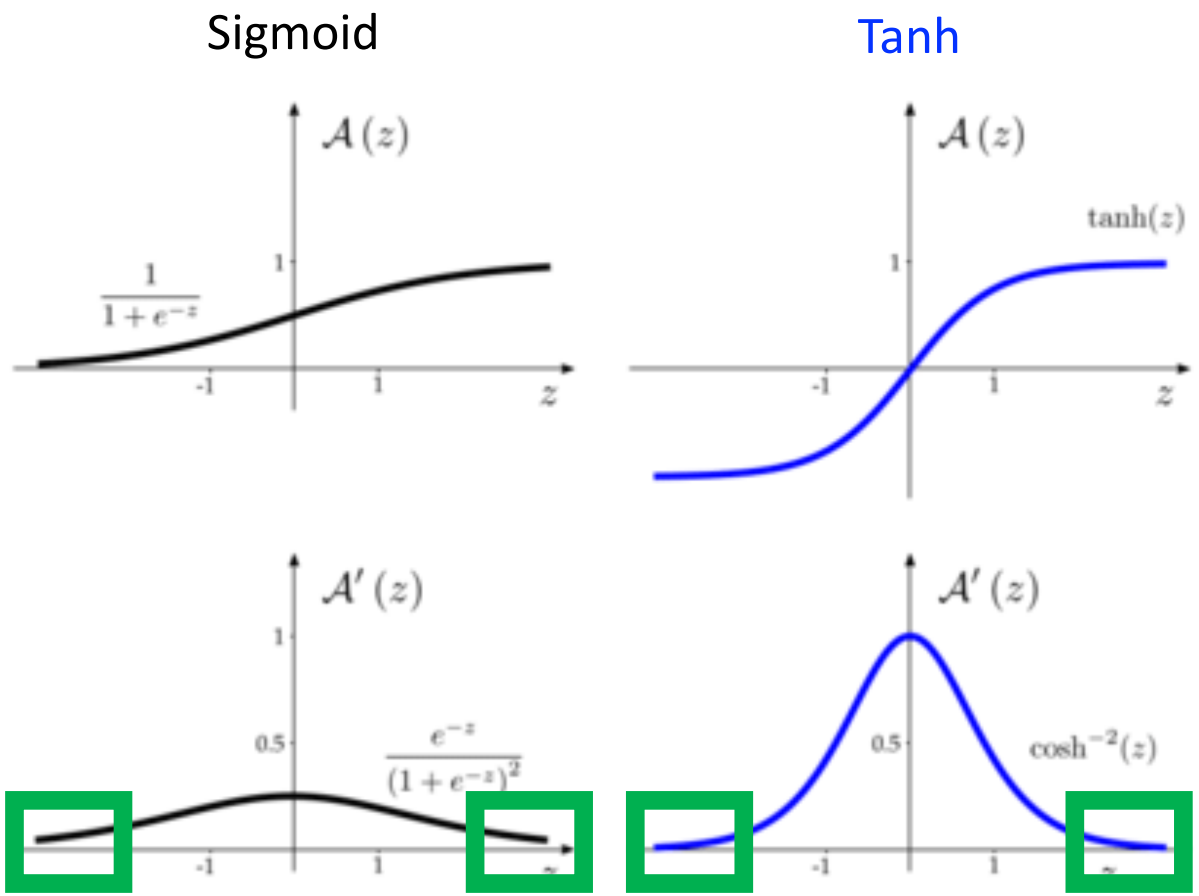
0 everywhere except 0 where it is non-differentiable



Deep Learning for NLP and Speech Recognition; Kamath, Liu, & Whitaker

No gradient means model parameters wouldn't change with gradient descent!

Activation Functions with Gradients: Nonlinear Activation Functions



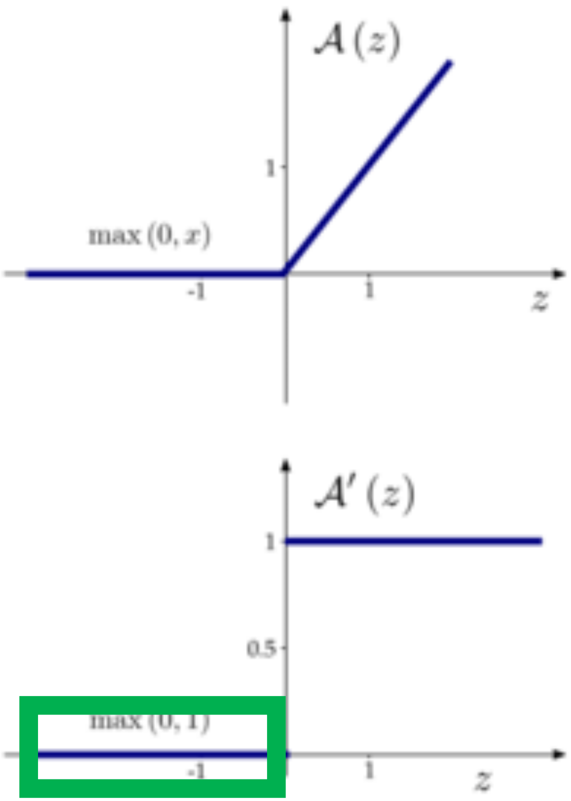
Problem: units with small or large “z” values lead to little/slow learning; why?

Reason: small gradients limit amount model parameters change with gradient descent

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

Activation Functions with Gradients: Nonlinear Activation Functions

ReLU



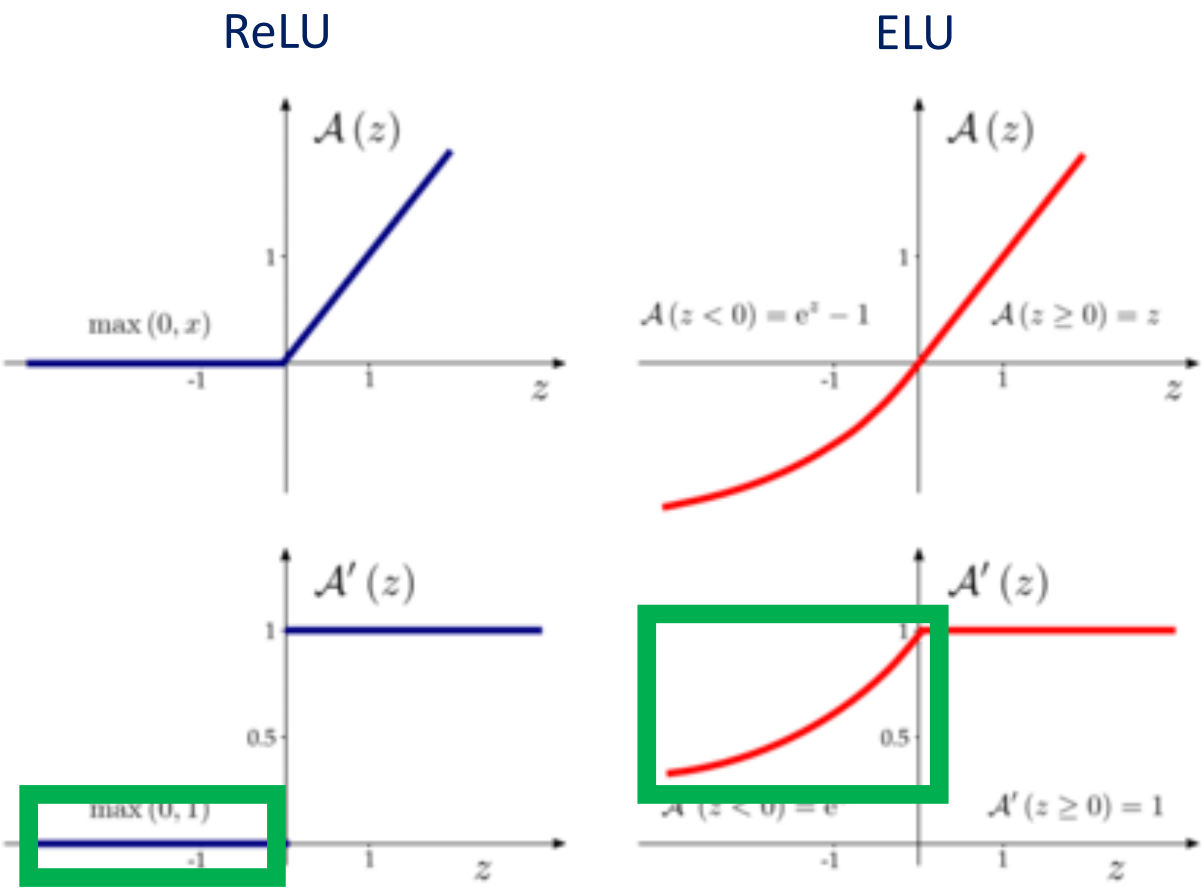
Advantages:

- Fast to compute
- Large gradient when unit is “firing”

Problem: no gradient means units can “die"

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

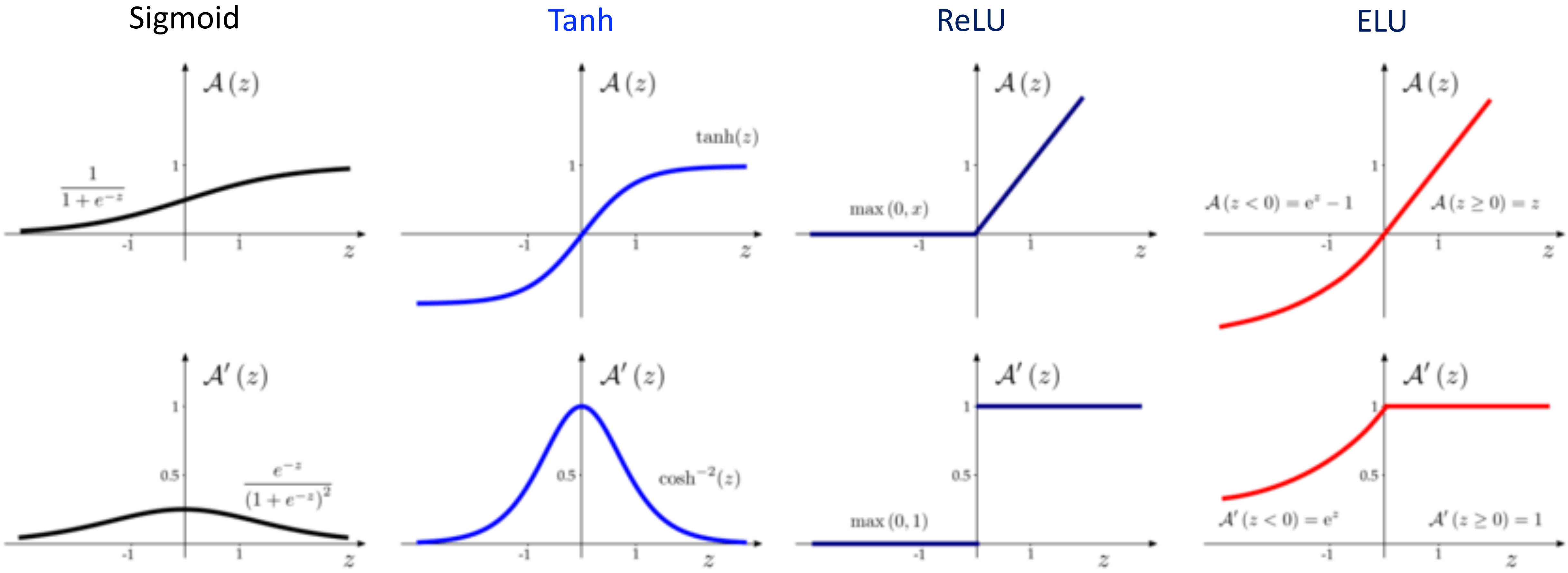
Activation Functions with Gradients: Nonlinear Activation Functions



Can avoid dying units by increasing computational complexity of ELU-based activation functions

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

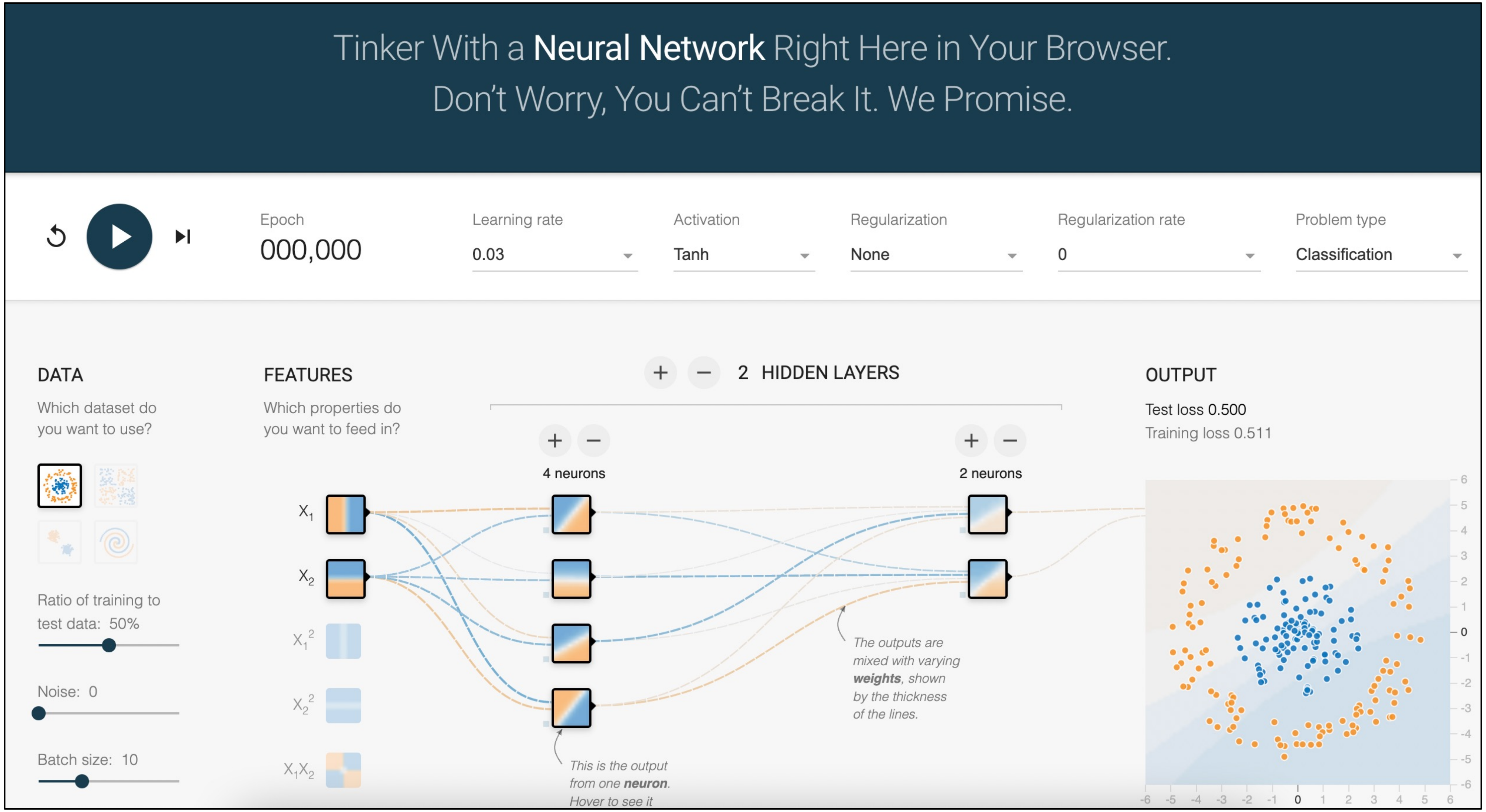
Activation Functions with Gradients: Nonlinear Activation Functions



Goal: computationally-efficient functions with large gradients to support efficient learning

Masi et al. Journal of the Mechanics and Physics of Solids. 2021

Demo: https://playground.tensorflow.org/



Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Gradient descent for activation functions


The End