Training Neural Networks: A Bottom-Up Perspective

Danna Gurari
University of Colorado Boulder
Spring 2024

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/

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

- Assignments (Canvas):
  - Problem assignment 1 due earlier today
  - Lab assignment 1 due in 1.5 weeks

- Questions?

Today's Topics

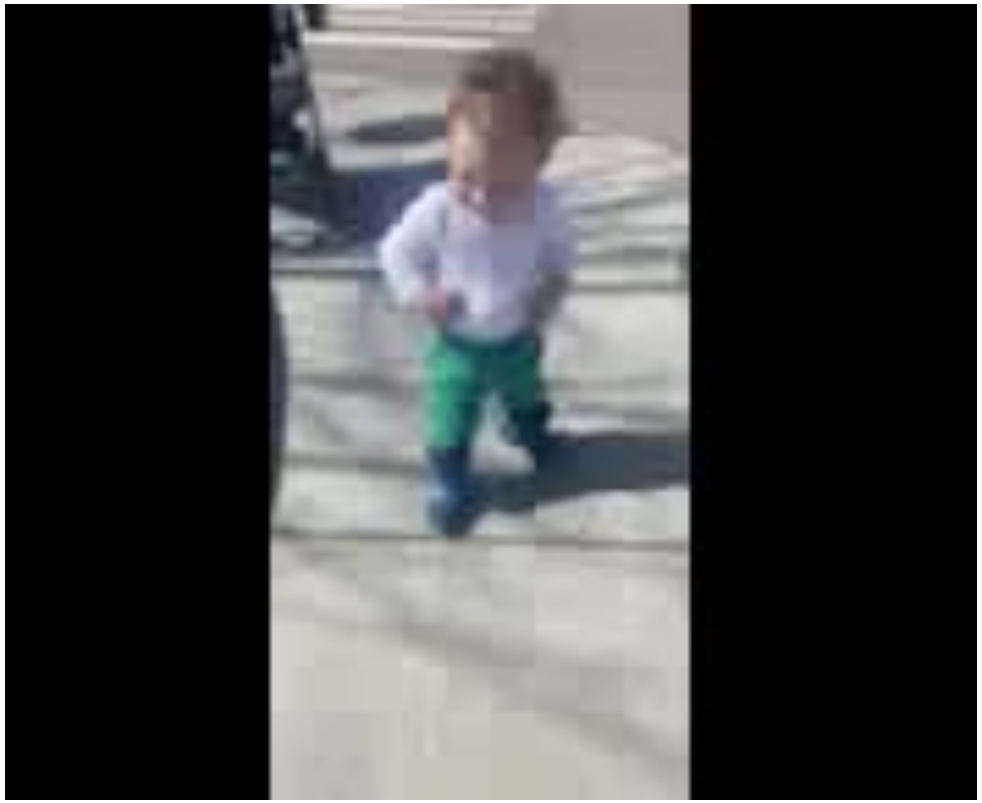
- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Lab assignment expectations

Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Lab assignment expectations

Objective Function: Analogous to Learning…

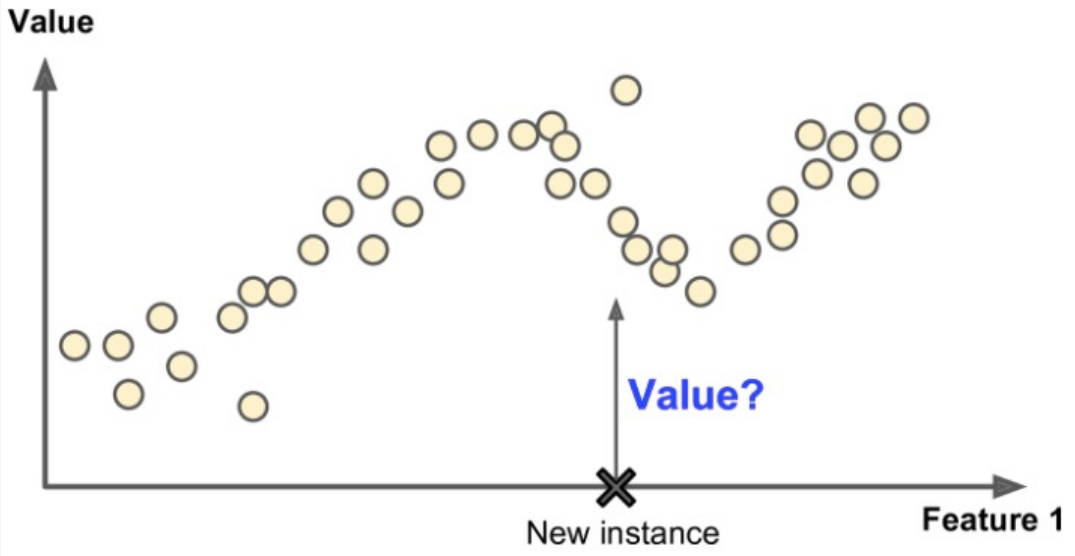
e.g., to walk



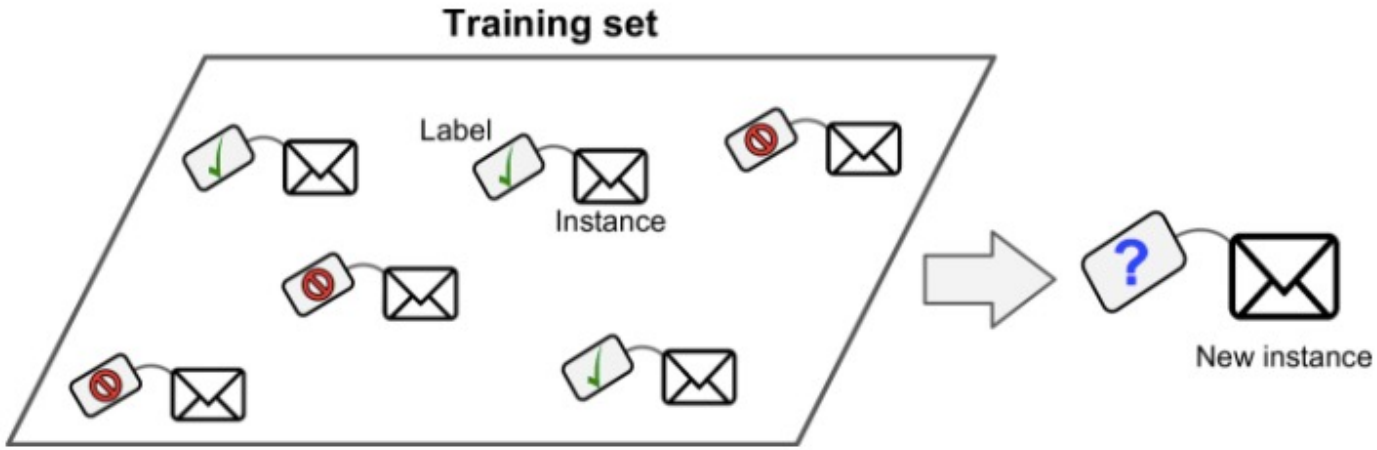
Key question: how do you measure/quantify task success?

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

Regression
(predict **continuous** value)



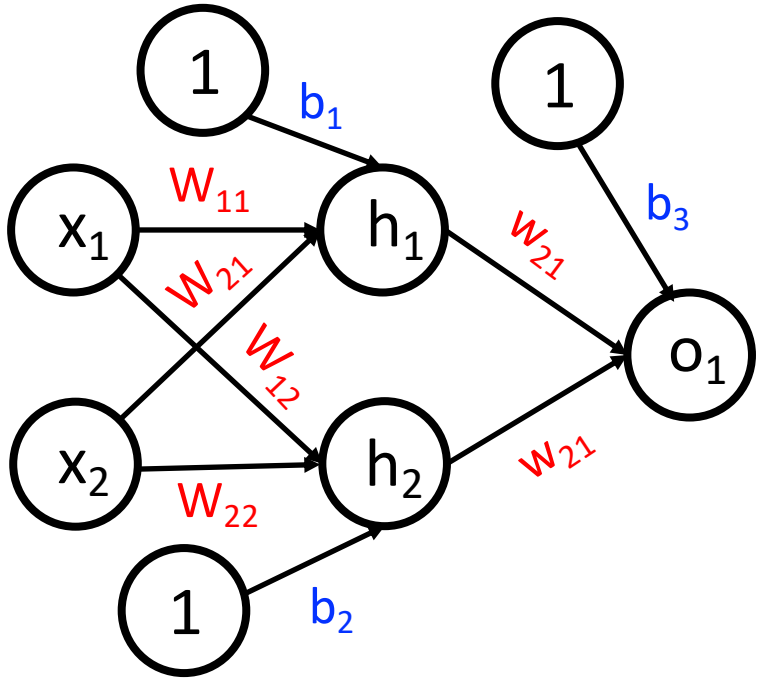
Classification
(predict **discrete** value)



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

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

e.g., learn weights and biases:



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

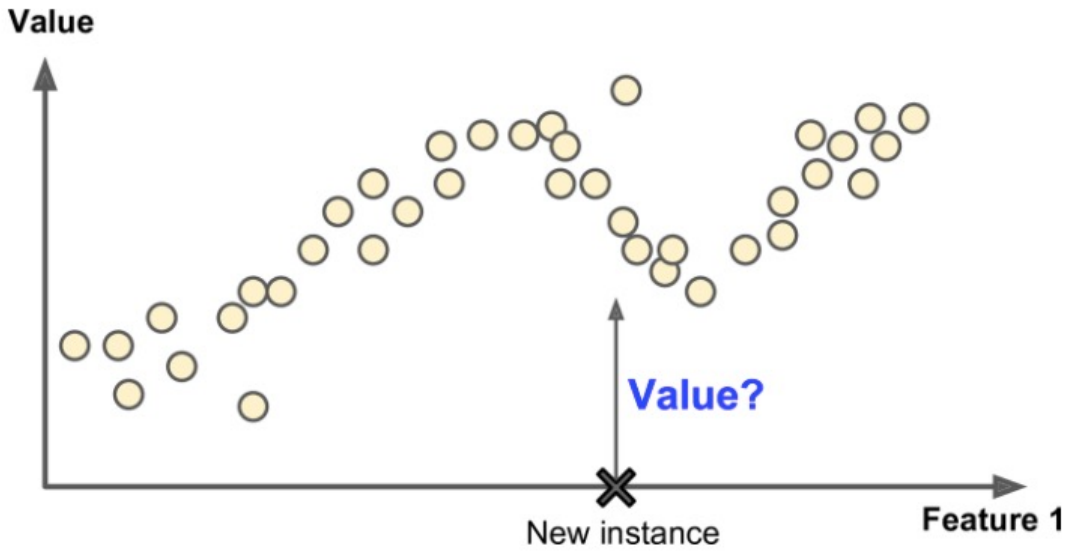
Predicted value

What is the range of possible values?

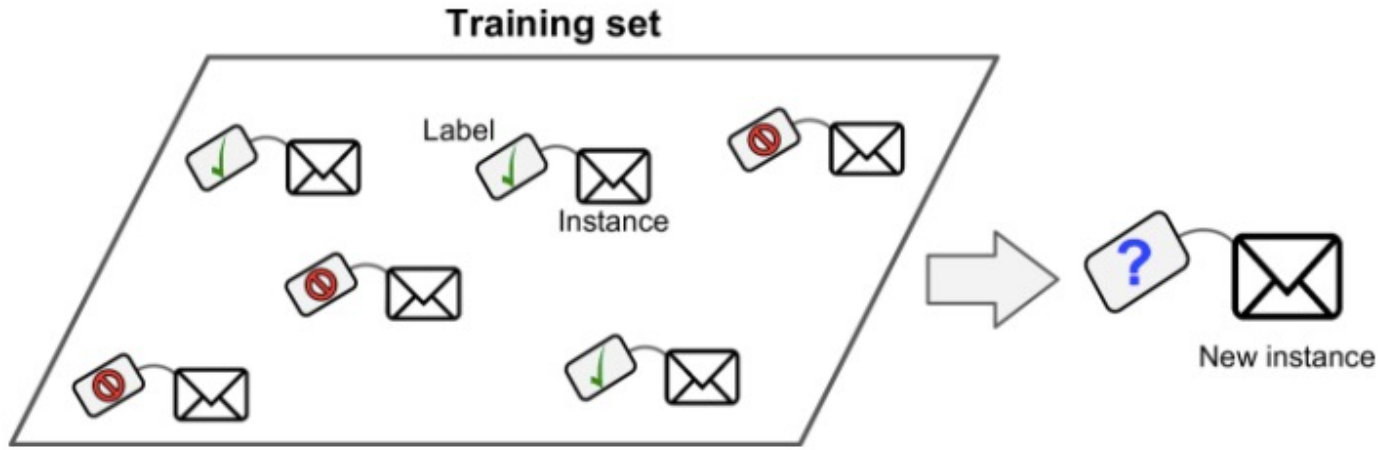
- Minimum: 0
  - i.e., all correct predictions
- Maximum: Infinity
  - i.e., incorrect predictions

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

Regression
(predict **continuous** value)

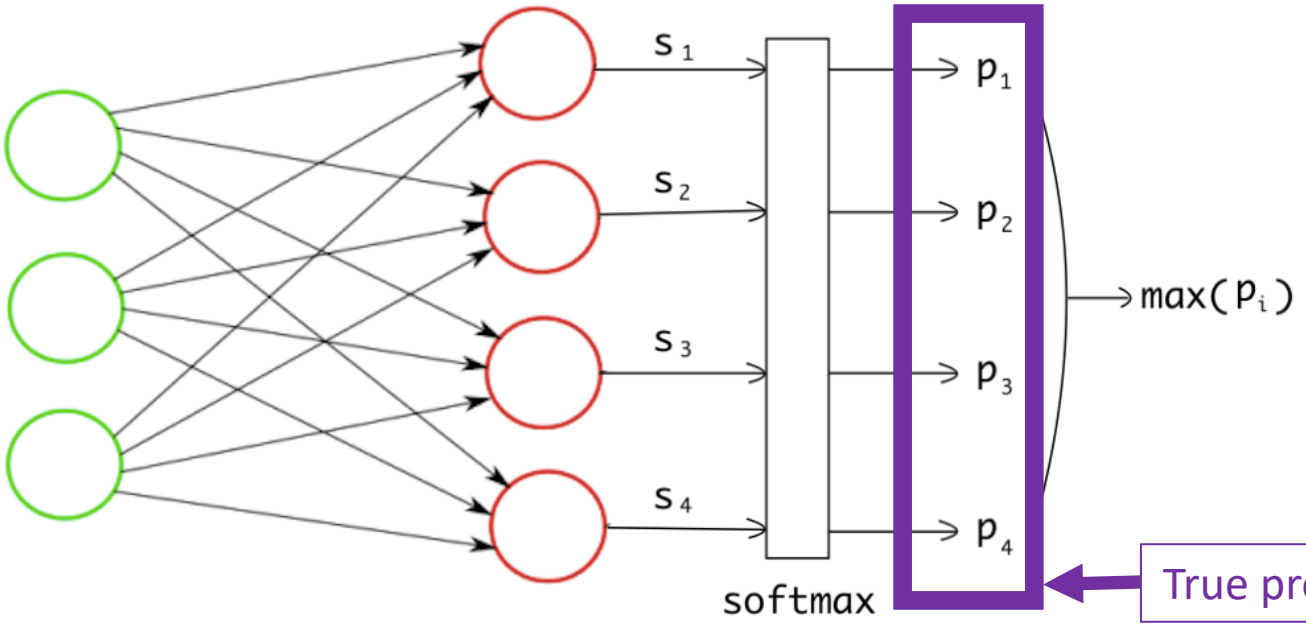


Classification
(predict **discrete** value)



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

$$L_{\mathrm{CE}}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

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

$$= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k|x)$$

Observed features

$$= -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)}$$

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

$$= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

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

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

Probability distribution of true class

Probability distribution of predicted class

Number of classes

$$L_{\mathrm{CE}}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

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

$$= -\sum_{k=1}^{K} y_k \log \hat{p}(y = k|x)$$

Observed features

$$= -\log \hat{y}_k, \quad \text{(where } k \text{ is the correct class)}$$

$$= -\log \frac{\exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

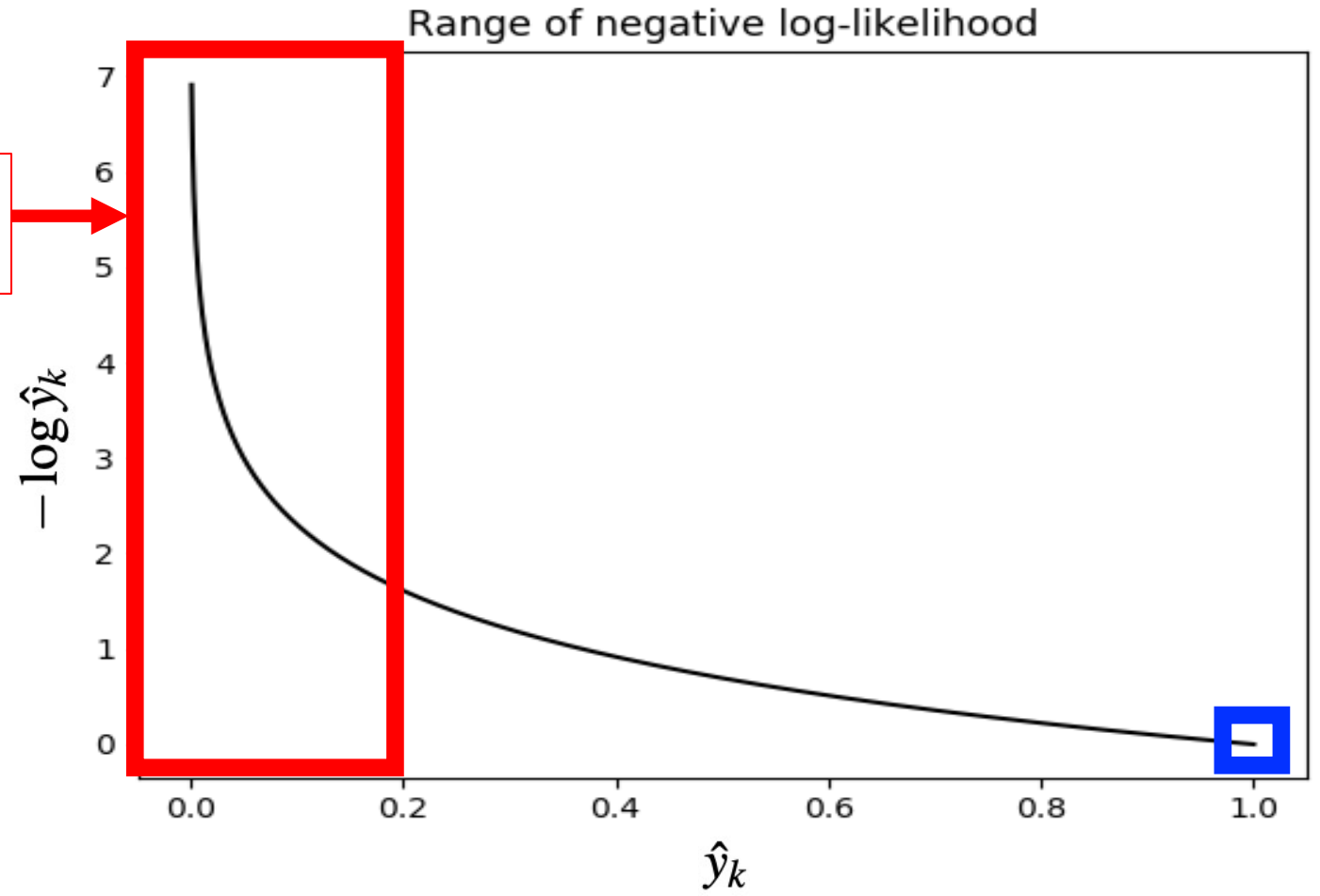
What is the range of possible values?

- 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



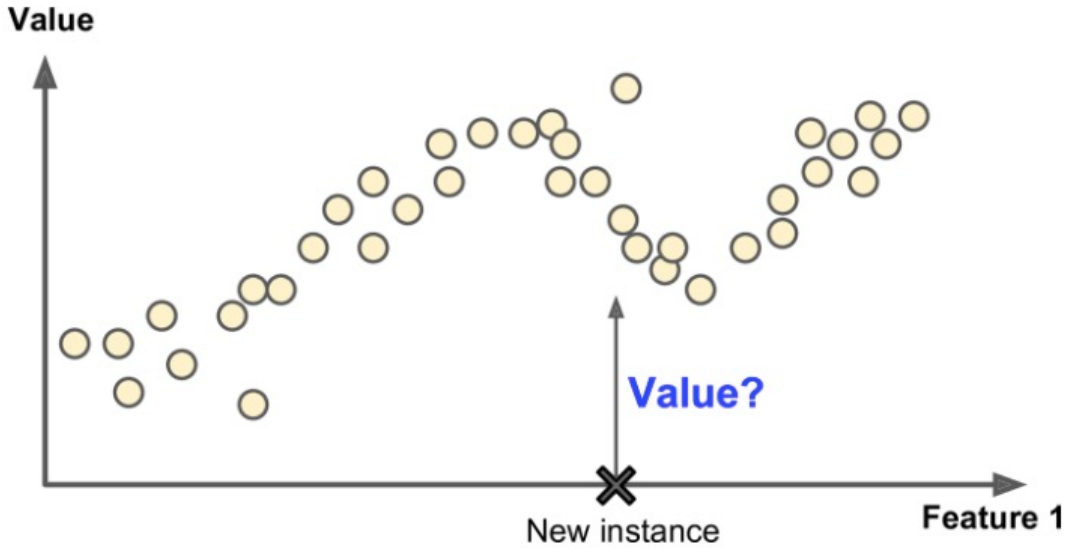
What is the range of possible values?

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

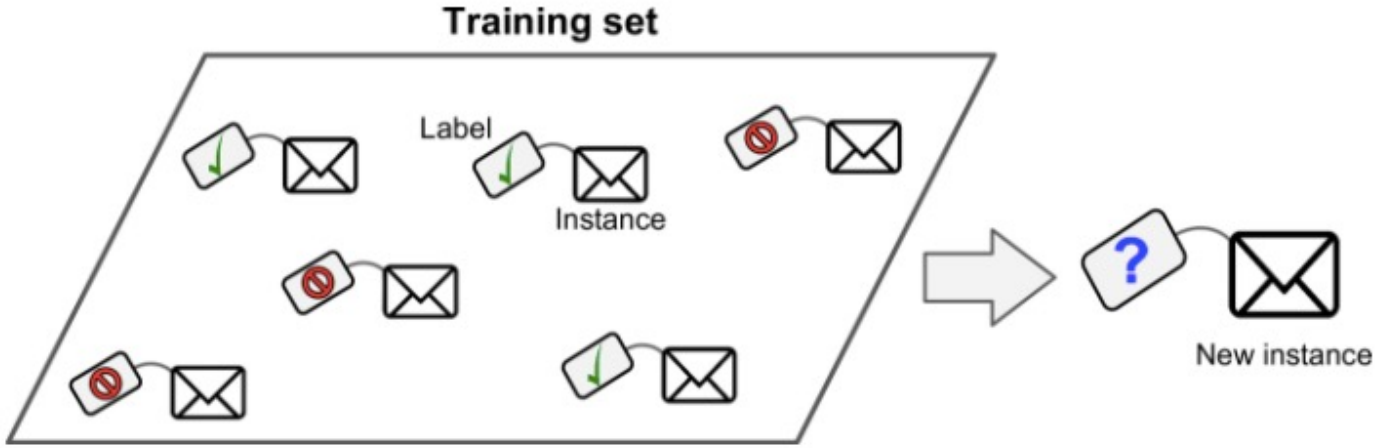
Source: https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

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

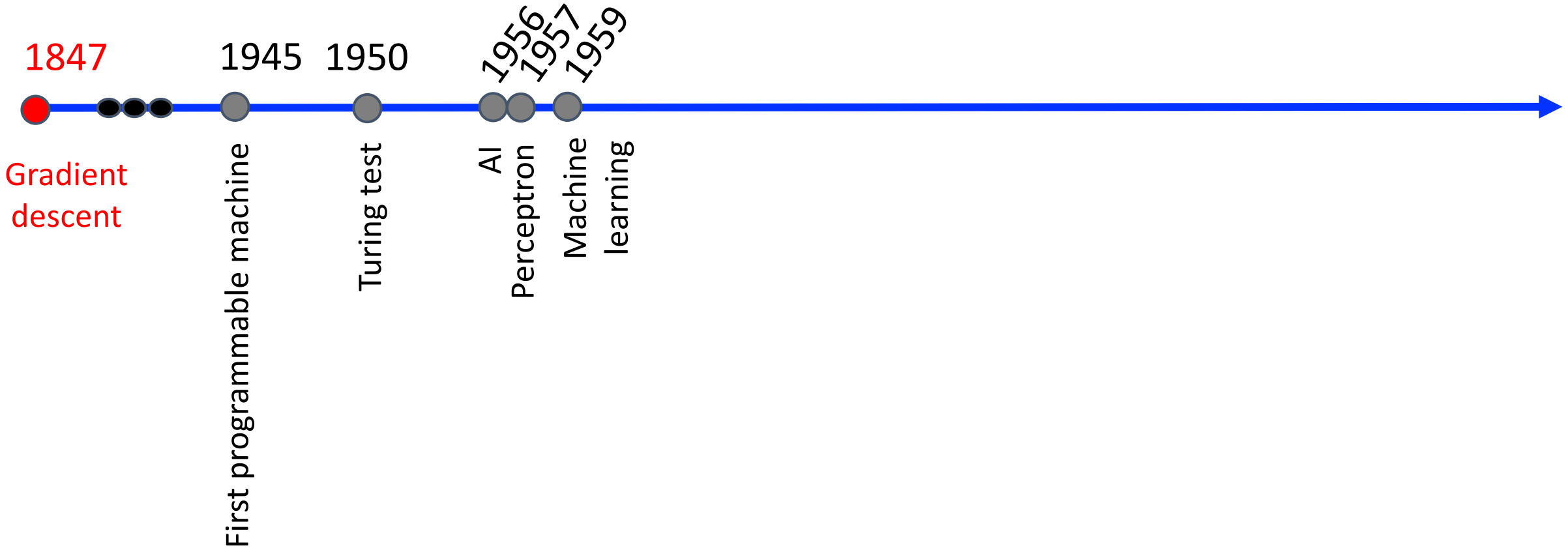
Practical Detail

- 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
- Lab assignment expectations

Historical Context: Gradient Descent



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

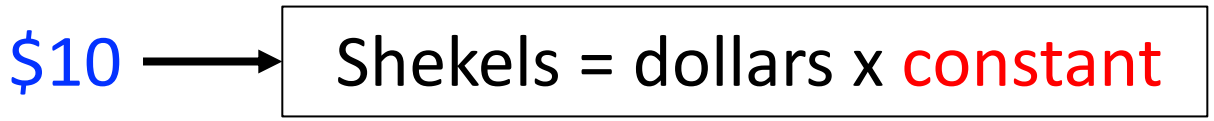
Scalable way to train non-linear models on “big data”

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”

Gradient Descent: Intuition

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

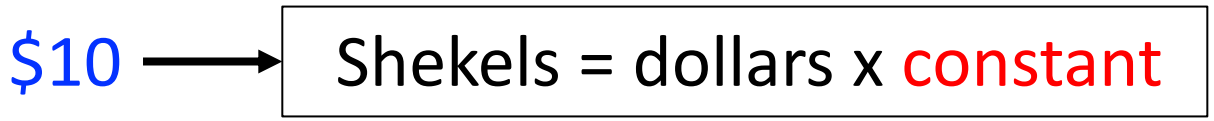


Gradient Descent: Intuition

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

Gradient Descent: Intuition

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

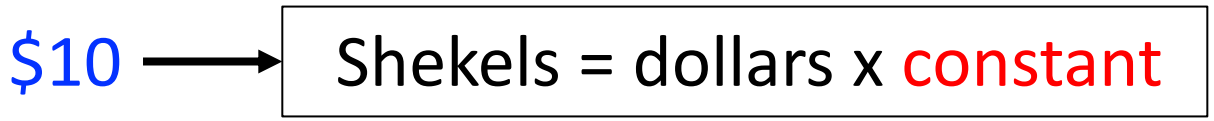


Gradient Descent: Intuition

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

Gradient Descent: Intuition

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



Gradient Descent: Intuition

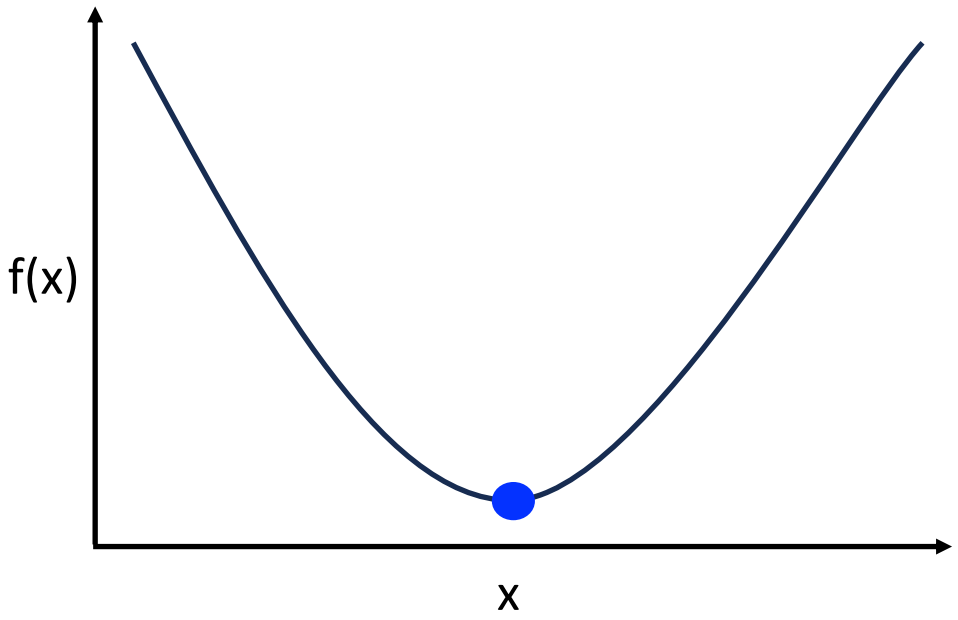
- Repeat:
  1. Guess
  2. Calculate error
- e.g., learn constant multiplier to convert US dollars to Israeli shekels

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

Gradient Descent: Possible Scenarios

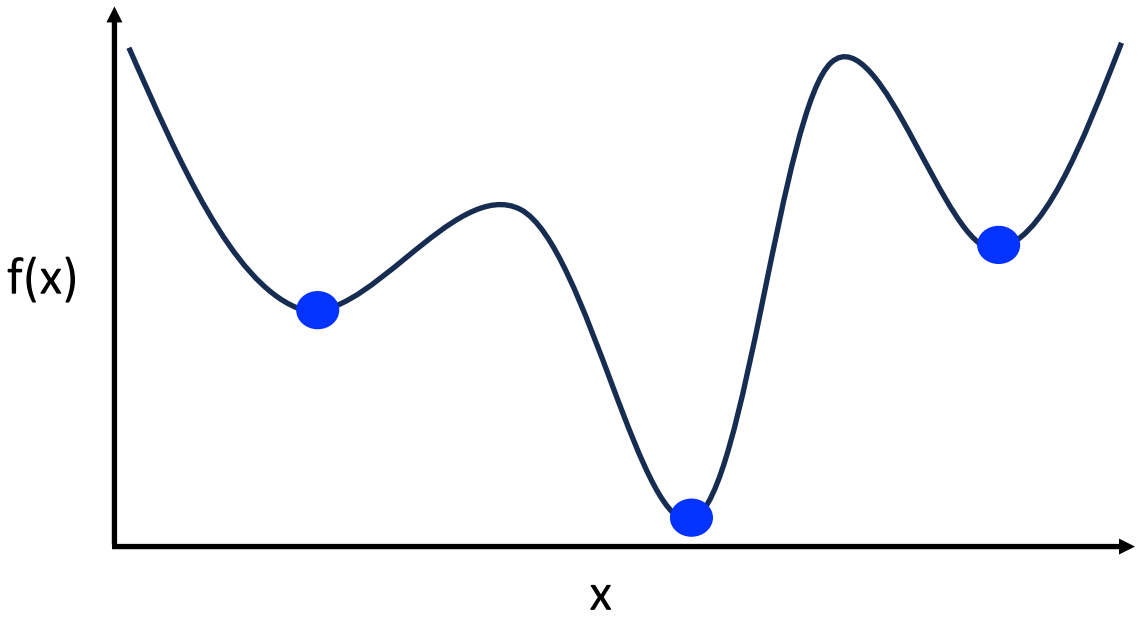
(simple 1-dimensional plots)

Convex Functions (one minimum)



Currency conversion example

Non-Convex Functions (multiple minima)



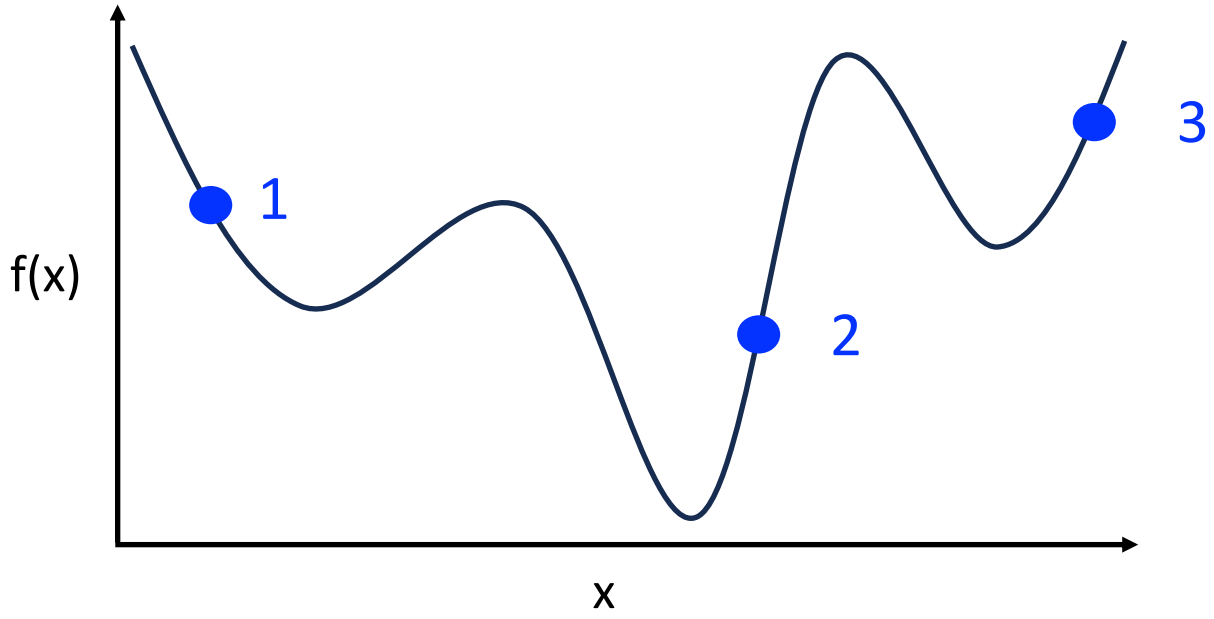
Our focus: deep learning

Gradient Descent: Approach

- Iteratively searching for better model parameters (e.g., weights and biases) that lead to smaller losses in the loss function
- Analogy: hike from mountains to Boulder blind or blindfolded!

Gradient Descent: Definitions

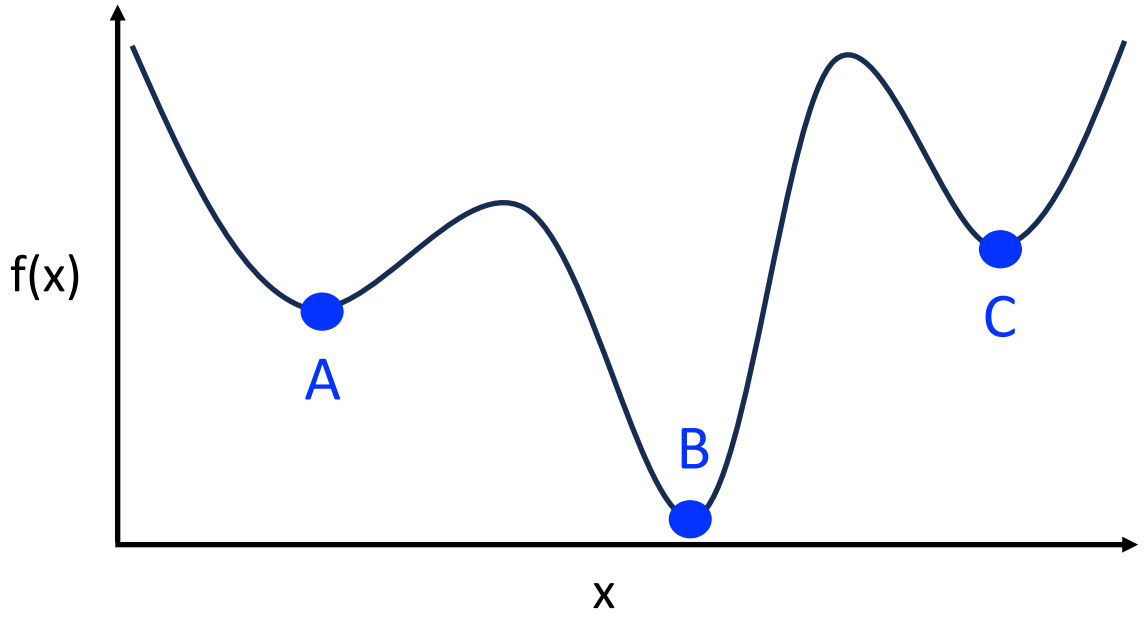
- **Gradient:** a vector indicating how a slight change to each function variable in ***x*** increases the output ***f(x)*** (i.e., partial derivatives for multiple variables)
- Recall, a derivative indicates the slope (rise/run) of the function at any point
- **Gradient descent**: to ***minimize*** the function, iteratively step in opposite direction of gradient (i.e., descent rather than ascent)



- What does the derivative look like at position 1? (Class volunteer)
- Is position 1's slope positive or negative?
- From position 1, should follow opposite the gradient direction to minimize the function
- What does the derivative look like at position 2? (Class volunteer)
- Is position 2's slope positive or negative?
- From position 2, should follow opposite the gradient direction to minimize the function

Gradient Descent: Definitions

- **Gradient:** a vector indicating how a slight change to each function variable in ***x*** increases the output ***f(x)*** (i.e., partial derivatives for multiple variables)
- Recall, a derivative indicates the slope (rise/run) of the function at any point
- **Gradient descent**: to ***minimize*** the function, iteratively step in opposite direction of gradient (i.e., descent rather than ascent)



Which letter(s) show global minima?

Which letter(s) show local minima?

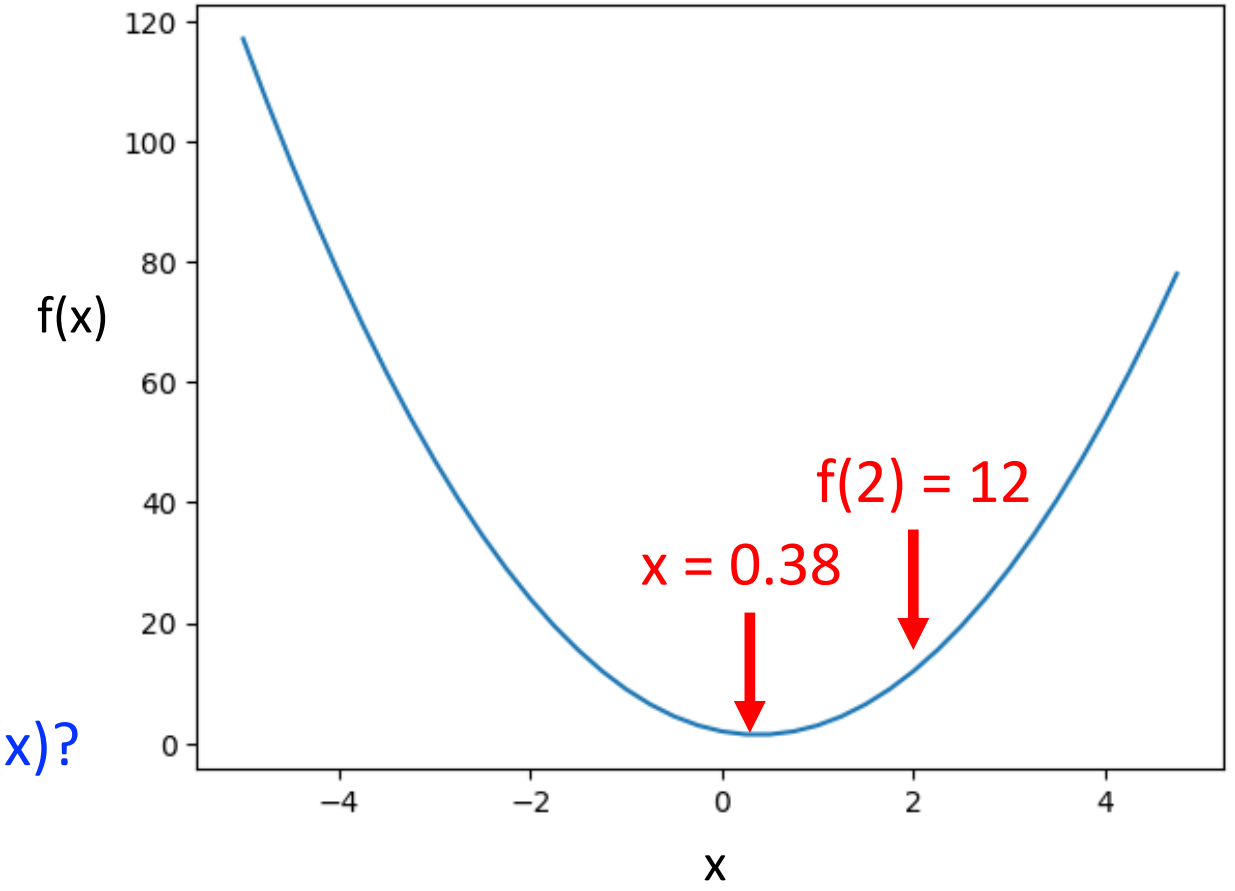
Reviewing Fundamentals: Toy Example 1

$f(x) = 2 - 3x + 4x^2$

Is this function convex or non-convex?

What is the value of the function at x =2?

What is the value of x at the minimum of f(x)?



Reviewing Fundamentals: Toy Example 1

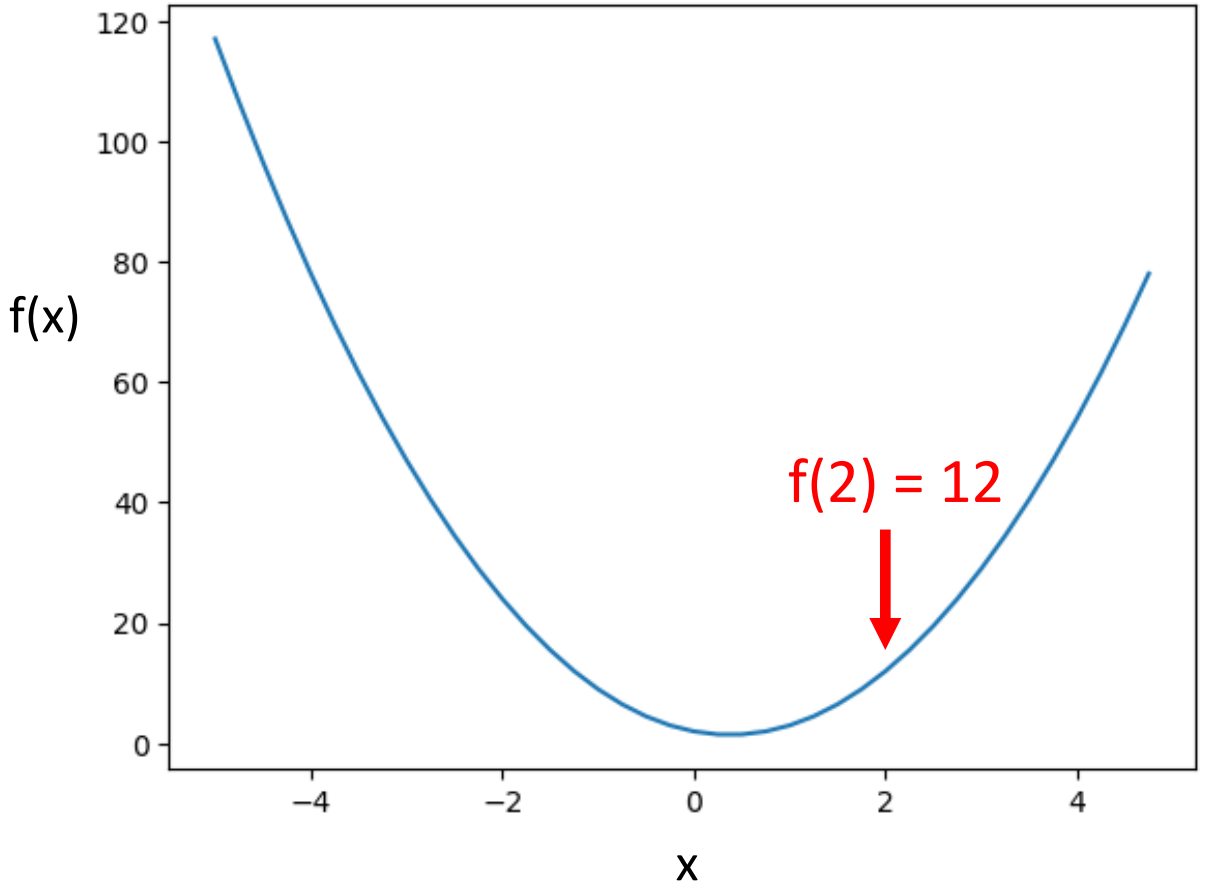
$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$

If *h* is slightly positive, at x=2, will *f(x+h)* be greater than or less than 12?

By how much?



Reviewing Fundamentals: Toy Example 1

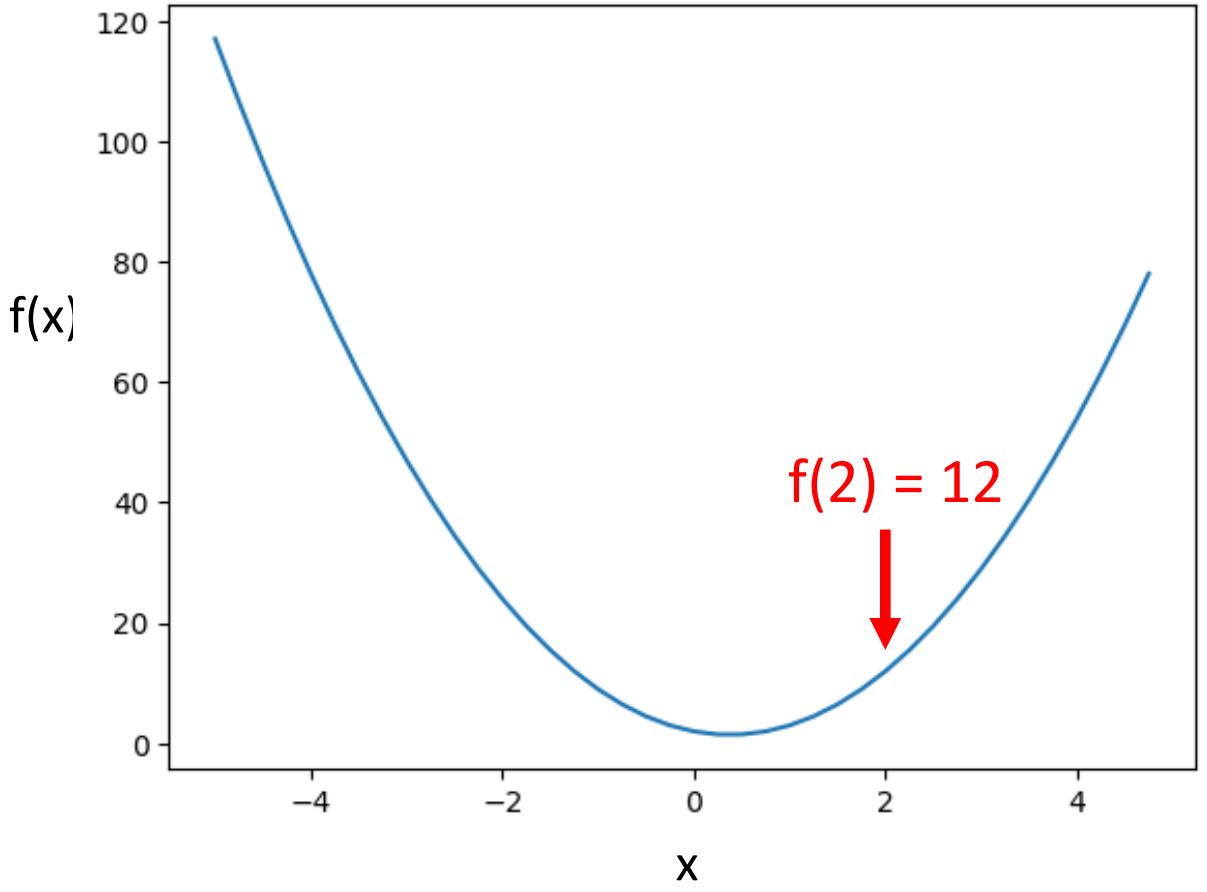
$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$

e.g., at x = 2 (rounding to 3 decimals)
- for h = 0.01: f(x+h) = 12.130, slope = 13.040
- for h = 0.001: f(x+h) = 12.013 & slope = 13.004
- for h = 0.0001: f(x+h) = 12.001 & slope = 13.000

As h gets closer to 0, slope approaches 13
Mathematically, f'(x) = -3 + 8x; f'(2) = -3 + 16 = 13



Reviewing Fundamentals: Toy Example 1

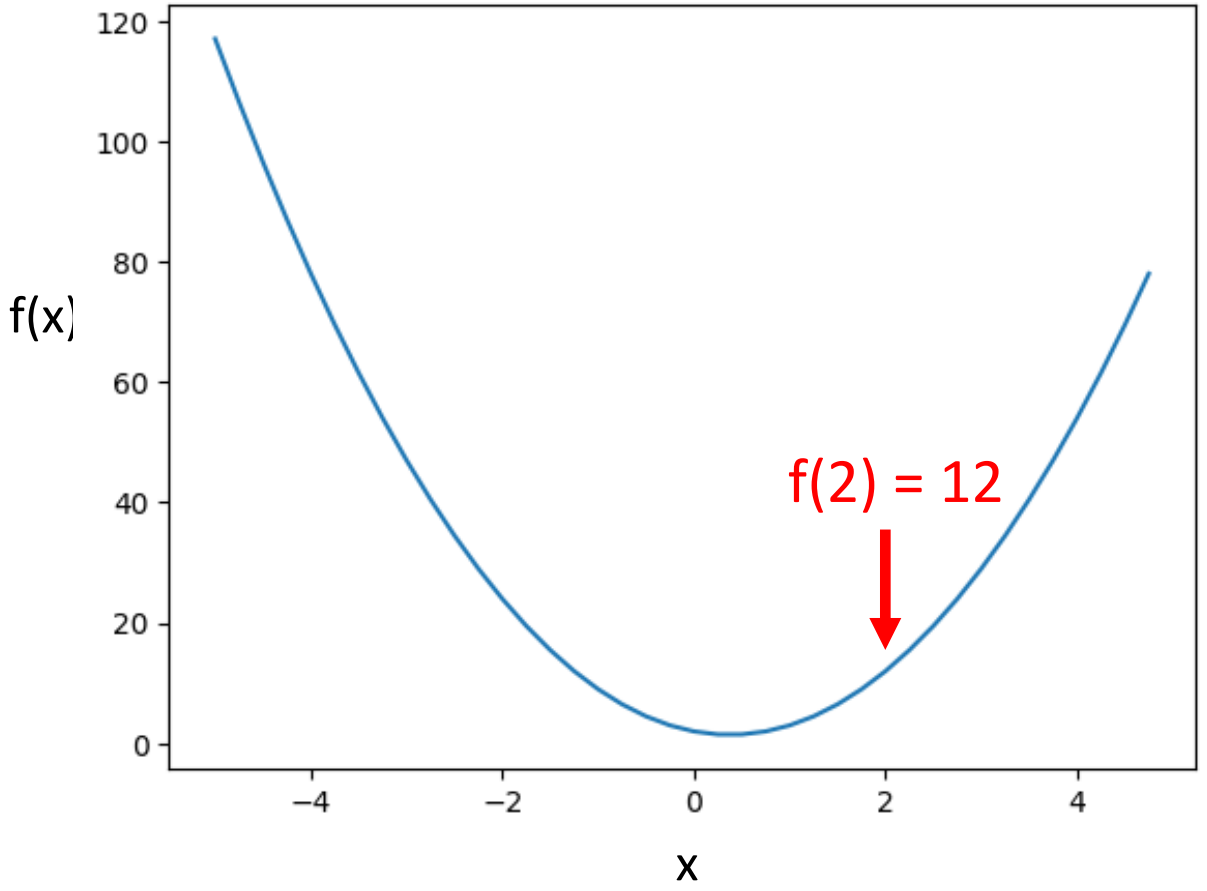
$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$

e.g., at x = 2 (rounding to 3 decimals)
- for h = 0.01: f(x+h) = 12.130, slope = 13.040

Increasing x by h results in an increase of ~13 times that amount in f(x) (or 13 * .01)



Reviewing Fundamentals: Toy Example 1

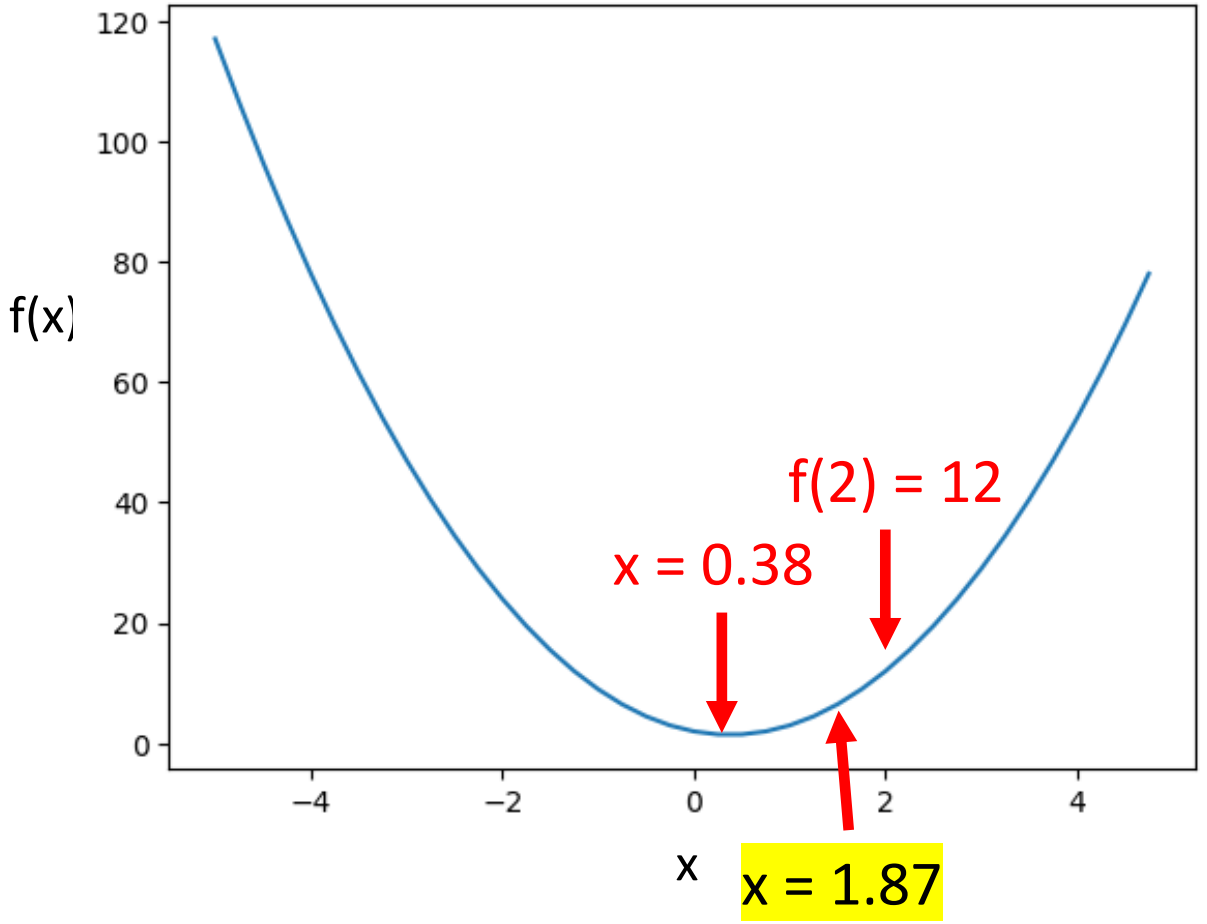
$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$

e.g., at x = 2 and df/dx = 13
Approach minimum point by stepping in opposite direction of derivative (e.g., step size of 0.01); what should be new value for x?

2 – (0.01 * 13) = 1.87

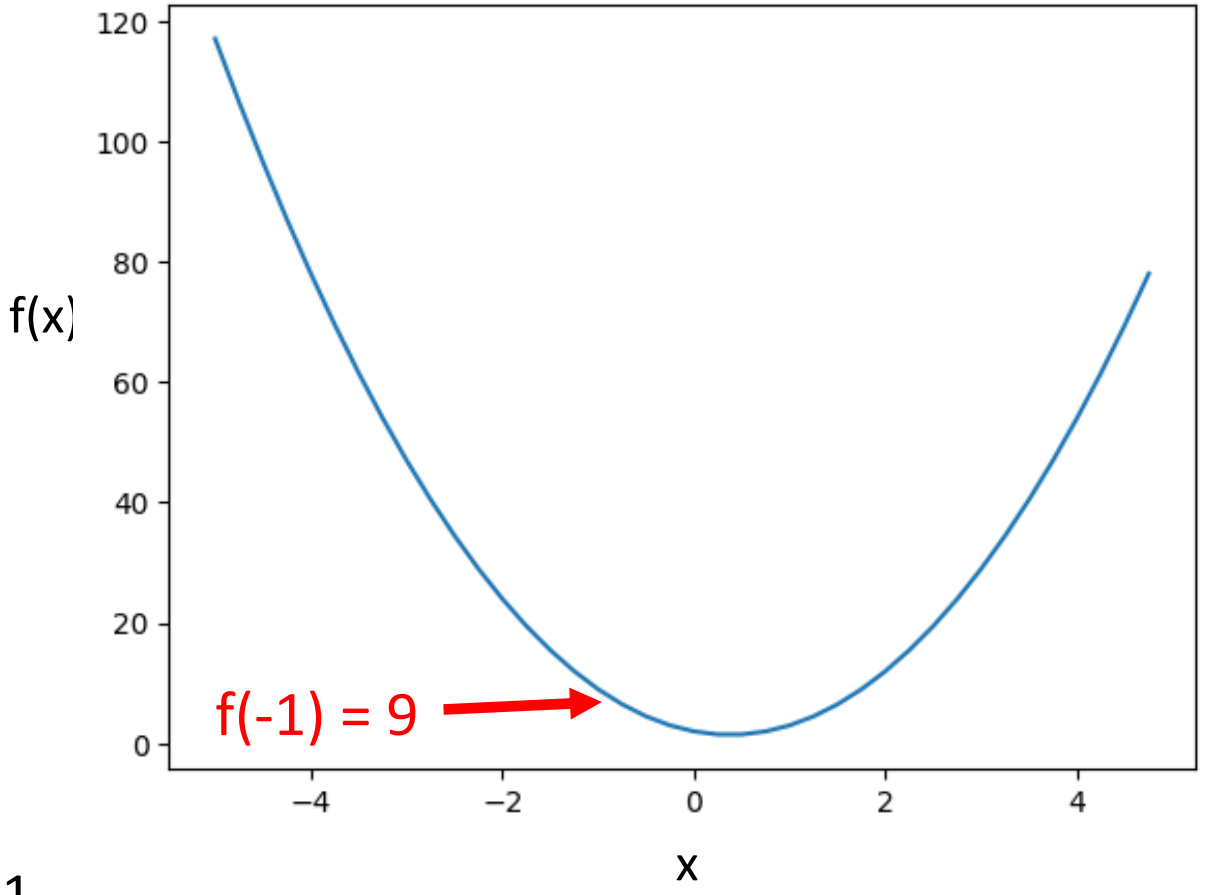


Reviewing Fundamentals: Toy Example 1

$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$



e.g., at x = -1 (rounding to 3 decimals)
- for h = 0.01: f(x+h) = 8.890, slope = -10.960
- for h = 0.001: f(x+h) = 8.989, slope = -10.996
- for h = 0.0001: f(x+h) = 8.999, slope = -11.000

As h gets closer to 0, slope approaches -11

Mathematically, f'(x) = -3 + 8x; f'(-1) = -3 + 8*-1 = -11

Reviewing Fundamentals: Toy Example 1

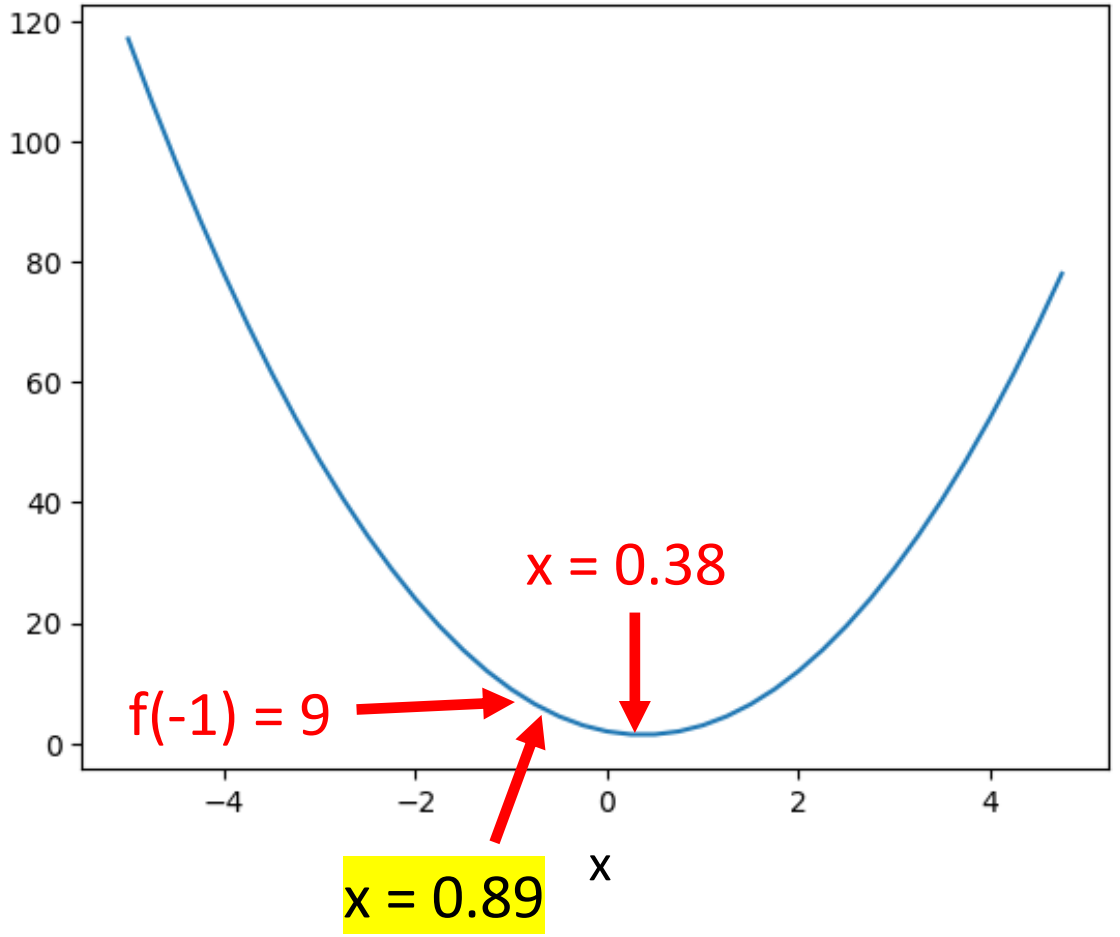
$f(x) = 2 - 3x + 4x^2$

Derivative indicates how a slight change (i.e., h) to every x will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(x+h) - f(x)}{h}$$

e.g., at x = -1 and df/dx = -11
Approach minimum point by stepping in opposite direction of derivative (e.g., step size of 0.01); what should be new value for x?

-1 - (0.01 * -11) = -0.89



Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

What is the value of the function when a = 1, b = -2, and c = 3?

If we only change *a* with a slightly positive *h*, will $f(\mathbf{x}+h)$ be greater than or less than -5?

By how much?

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

e.g., at a = 1, b = -2, and c = 3 when only changing the value of a (rounding to 3 decimals)
- for h = 0.01: f(**x**+h) = (1+.01) + -2*3 = -4.990, slope = 1.000
- for h = 0.001: f(**x**+h) = (1+.001) + -2*3 = -4.999, slope = 1.000

As h gets closer to 0, slope approaches 1

Mathematically, df/da = 1

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

Recall the value of the function when a = 1, b = -2, and c = 3 is -5

If we change only *b* with a slightly positive *h*, will $f(\mathbf{x}+h)$ be greater than or less than -5?

By how much?

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

e.g., at a = 1, b = -2, and c = 3 when only changing the value of b (rounding to 3 decimals)
- for h = 0.01: f(**x**+h) = 1 + (-2 +.01)*3 = -4.970, slope = 3.000
- for h = 0.001: f(**x**+h) = 1 + (-2 +.001)*3 = -4.997, slope = 3.000

As h gets closer to 0, slope approaches 3

Mathematically, df/db = c (which we set to 3)

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

Recall the value of the function when a = 1, b = -2, and c = 3 is -5

If we change only *c* with a slightly positive *h*, will $f(\mathbf{x}+h)$ be greater than or less than -5?

By how much?

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

e.g., at a = 1, b = -2, and c = 3 when only changing the value of c (rounding to 3 decimals)
- for h = 0.01: f(**x**+h) = 1 + -2*(3 +.01) = -5. 02, slope = -2.000
- for h = 0.001: f(**x**+h) = 1 + -2*(3 +.001) = -5.002, slope = -2.000

As h gets closer to 0, slope approaches -2

Mathematically, df/dc = b (which we set to -2)

Reviewing Fundamentals: Toy Example 2

$f(\mathbf{x}) = a + bc$

Gradient indicates how a slight change (i.e., h) to any **input** will change the output:

$$\text{slope} = \frac{\text{rise}}{\text{run}} = \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}$$

In summary, gradient for any input can be computed with these equations:

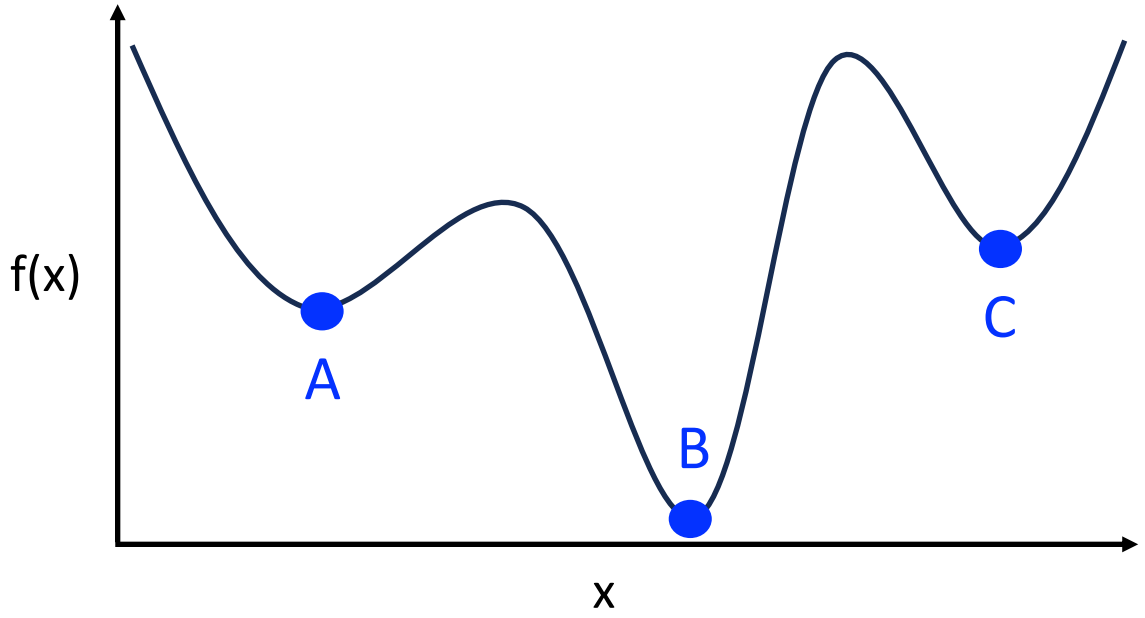
$df/da = 1;$ $df/db = c;$ $df/dc = b$

Can approach optimal point by stepping (e.g., step size of 0.01) in opposite direction of derivative? e.g., at $a = 1$, $b = -2$, and $c = 3$

$a - 0.01*1 = 1.01;$ $b - 0.01*c = -2.03;$ $c - 0.01*b = 3.02$ $f(x) = -5.12$

Gradient Descent: A Key (Old) Concern

Can arrive at a local rather than global minimum!



Global minima: B

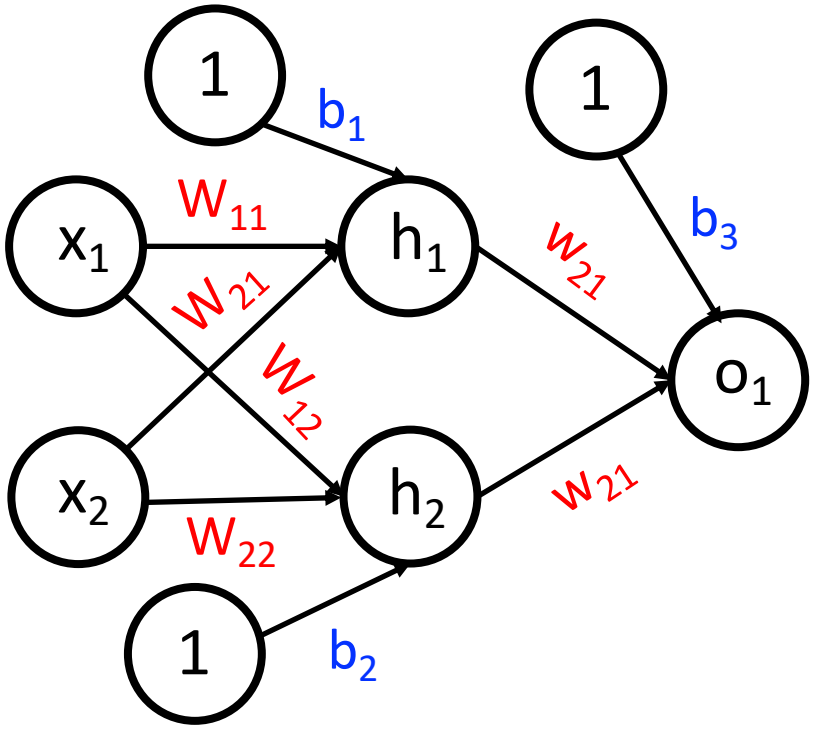
Local minima: A, B, C

Today's Topics

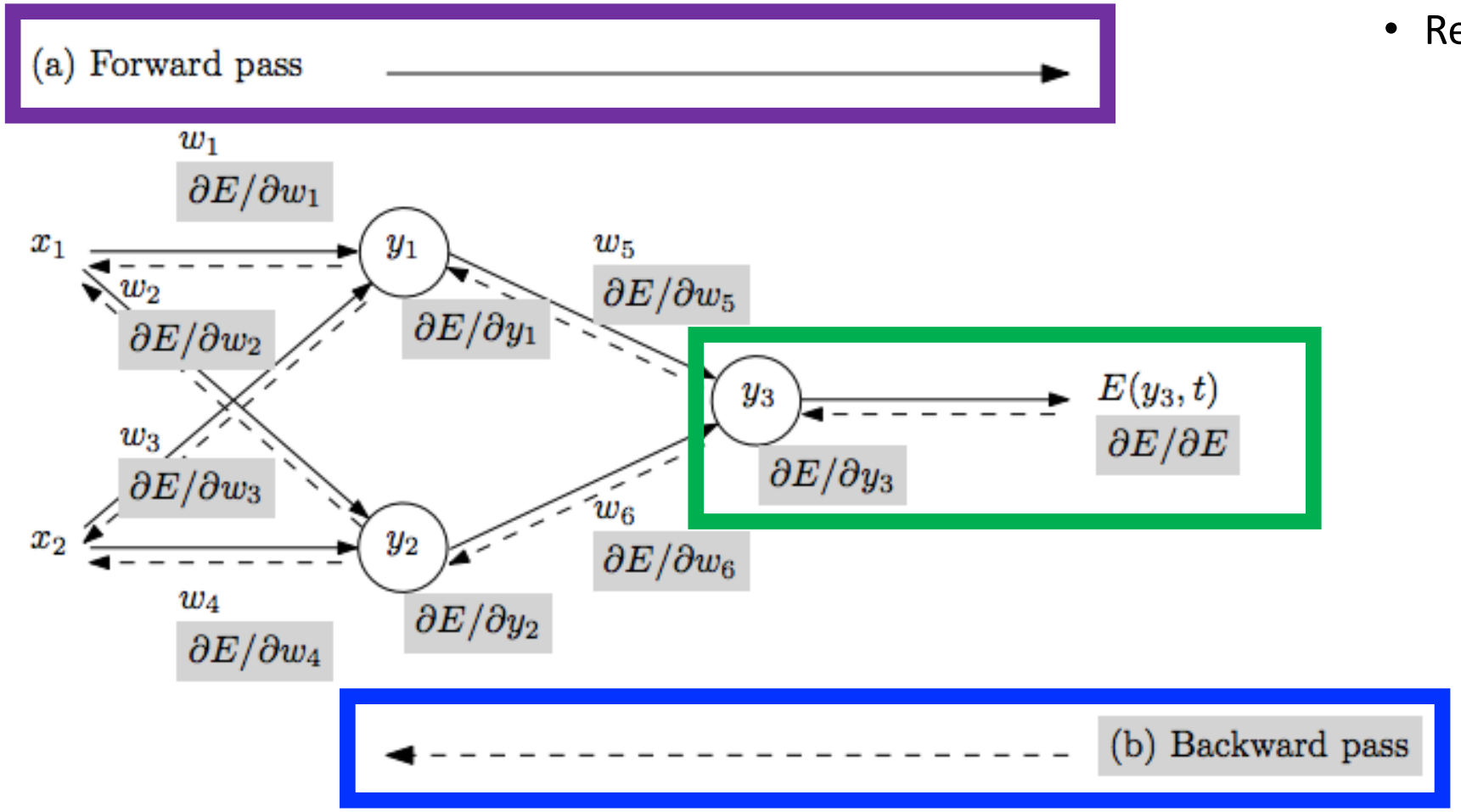
- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Lab assignment expectations

Goal: Train Neural Network

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



Gradient Descent: Implementation



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on 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

Gradient Descent: Implementation

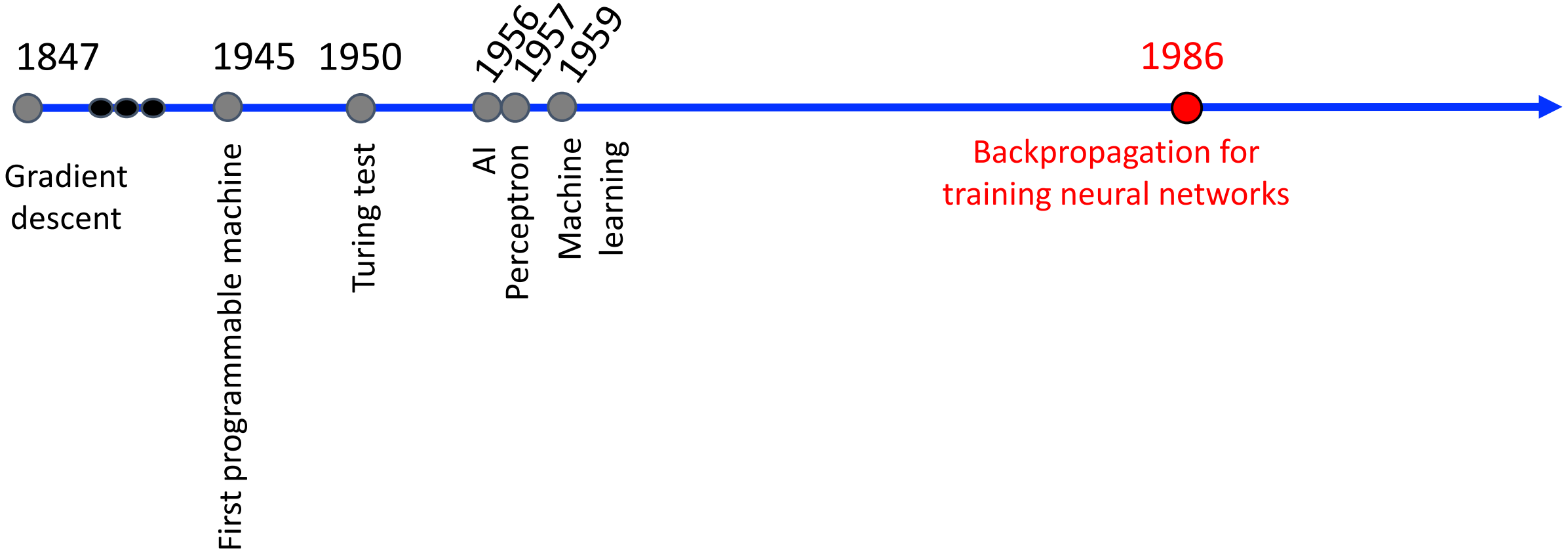
Key challenge: calculating gradients

Solution: backpropagation

- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on 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

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

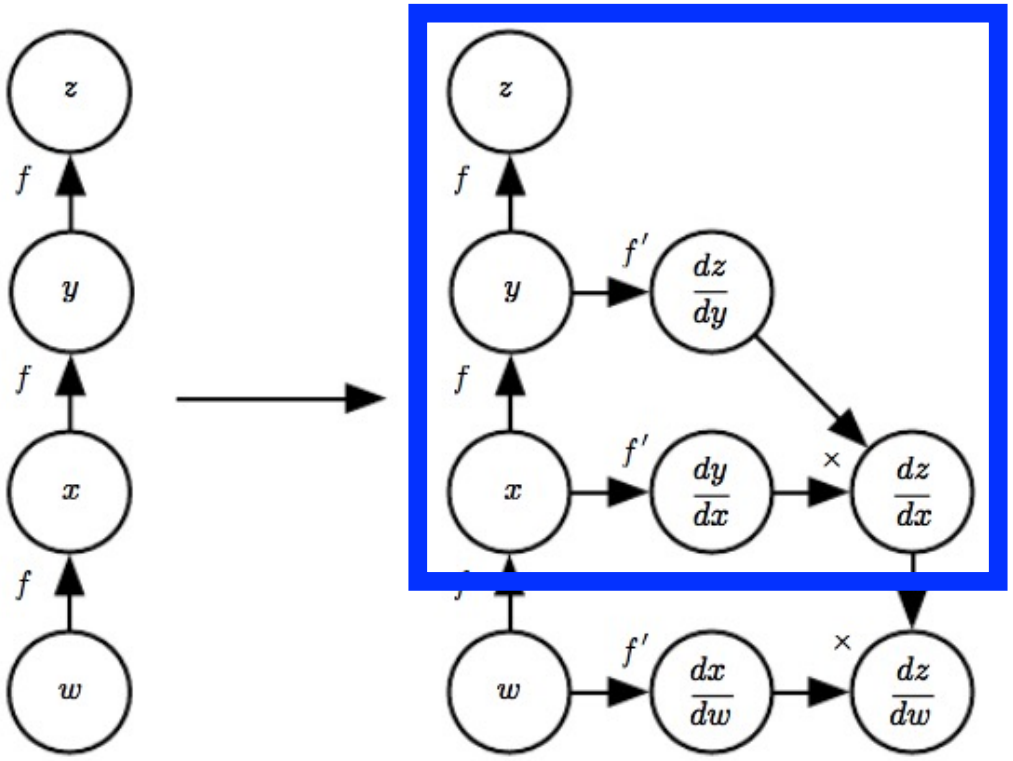
Solution: Backpropagation to Compute Gradients



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

Backpropagation Basics: Chain Rule

- **Idea**: compute gradient on loss function to inform how to nudge model parameters to reduce loss
- **Key observation**: networks are functions connected in a chain



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

Chain rule of calculus: can compute all derivatives from top to bottom using only local derivative information at each node;

e.g.,

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

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

Backpropagation Basics: Chain Rule

- **Idea**: compute gradient on loss function to inform how to nudge model parameters to reduce loss
- **Key observation**: networks are functions connected in a chain

Intuitive example: how much faster is my husband compared to my daughter? (dz/dx)

my husband travels twice as fast as my son (dz/dy)

my son travels three times as fast as my daughter (dy/dx)

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

Chain rule of calculus: can compute all derivatives from top to bottom using only local derivative information at each node;

e.g.,

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

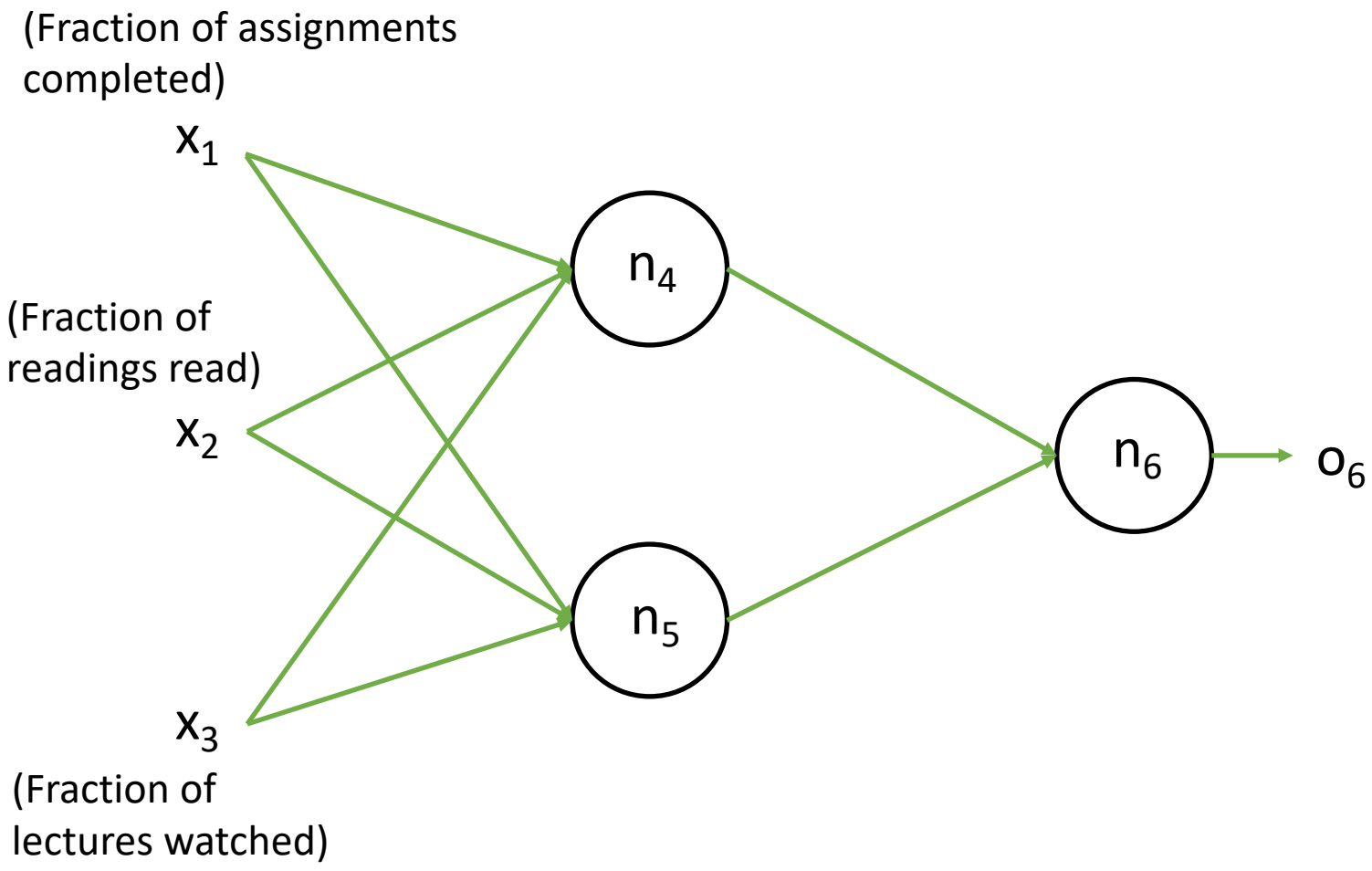
Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

Example

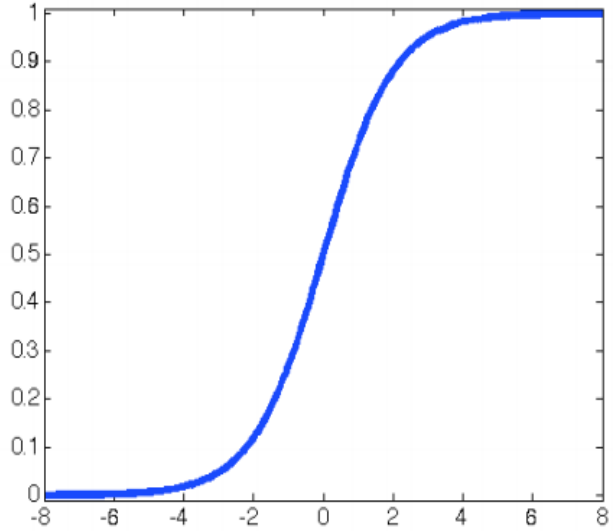
- Binary classification: predict if a student will get a B or better (minimum required for CS graduate student in this course)

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

Example: Choose Neural Network Architecture



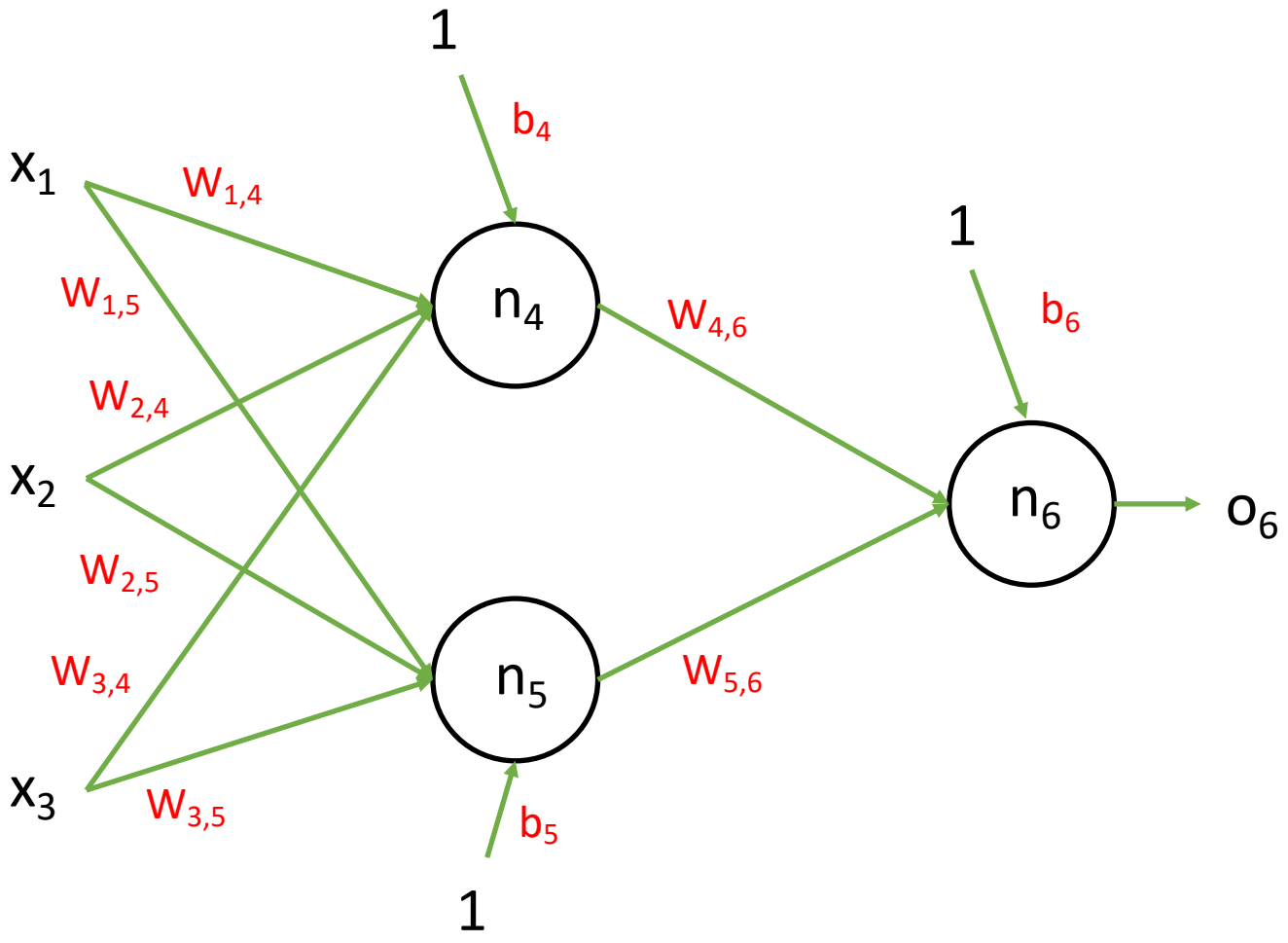
Sigmoid Activation Function



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

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

Example: Choose Neural Network Architecture



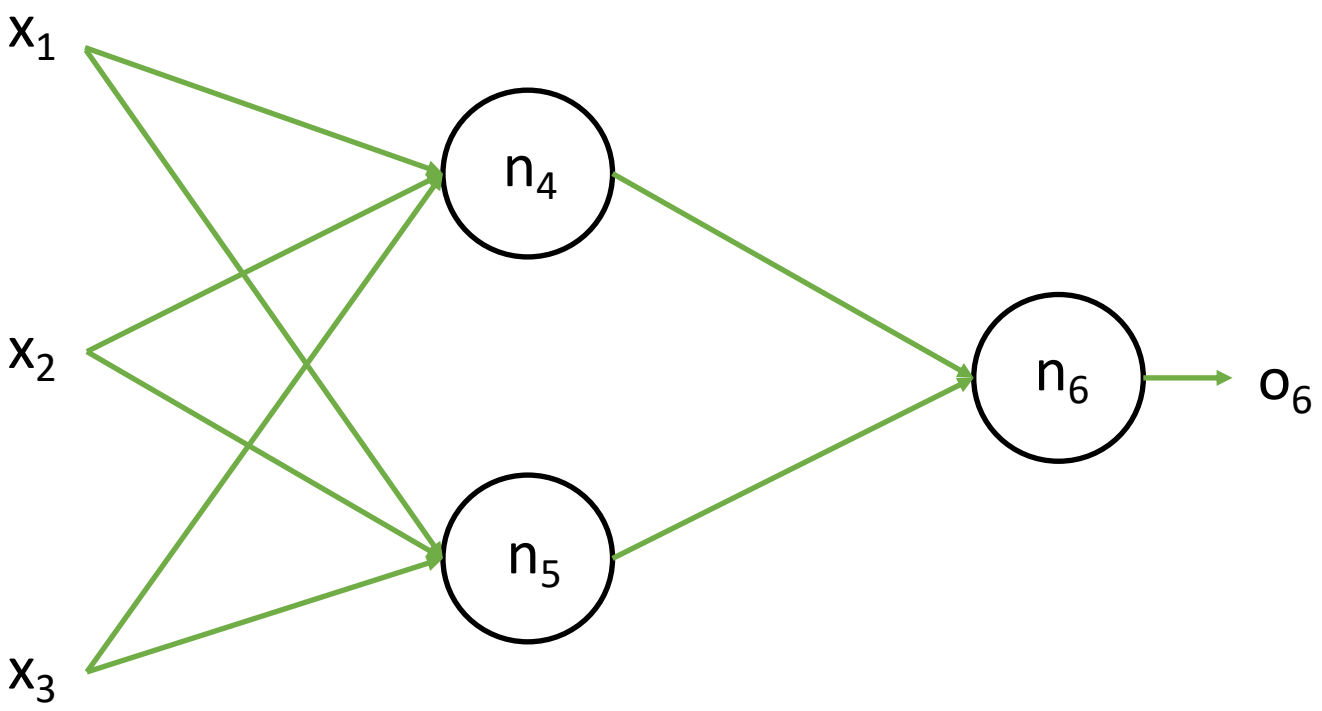
2-layer neural network

8 weights:

- Input to Hidden Layer 1:
  - 3x2 = 6
- Hidden Layer 1 to Hidden Layer 2:
  - 2x1 = 2
- 3 bias terms

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

Example: Choose Loss Function



Squared Error

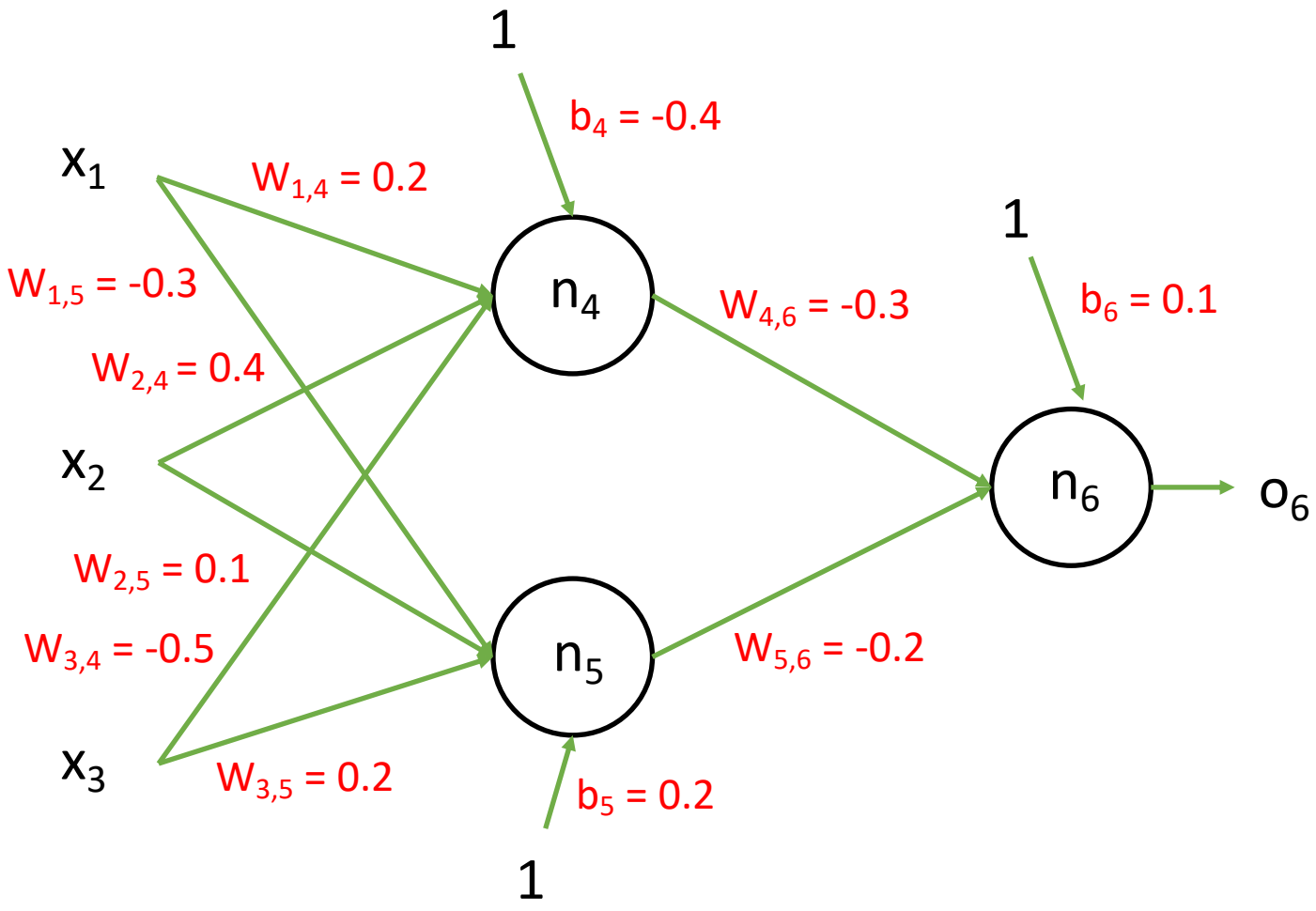
$$(Y_i - \hat{Y}_i)^2$$

True value

Predicted value

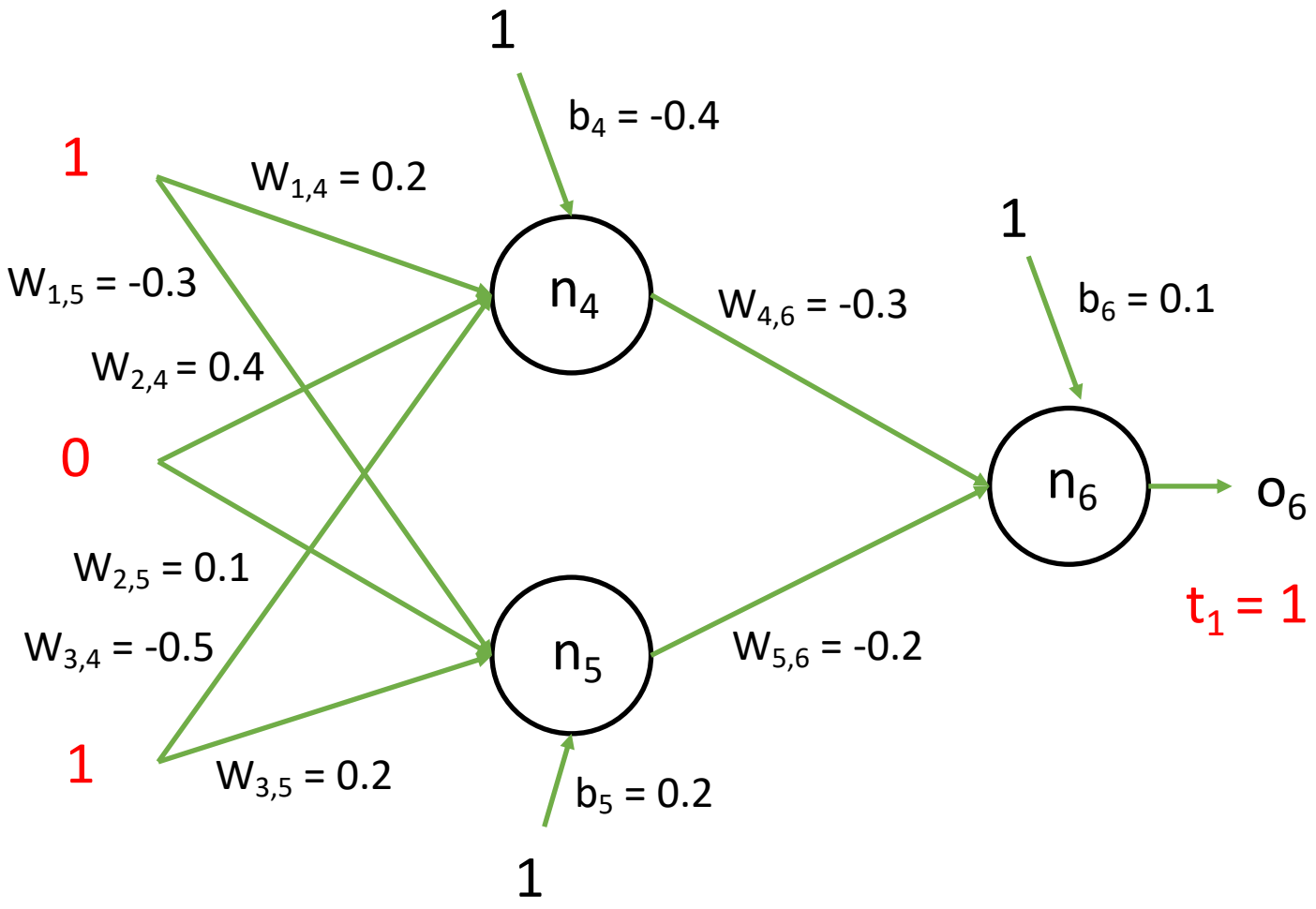
Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Initialize Model Parameters



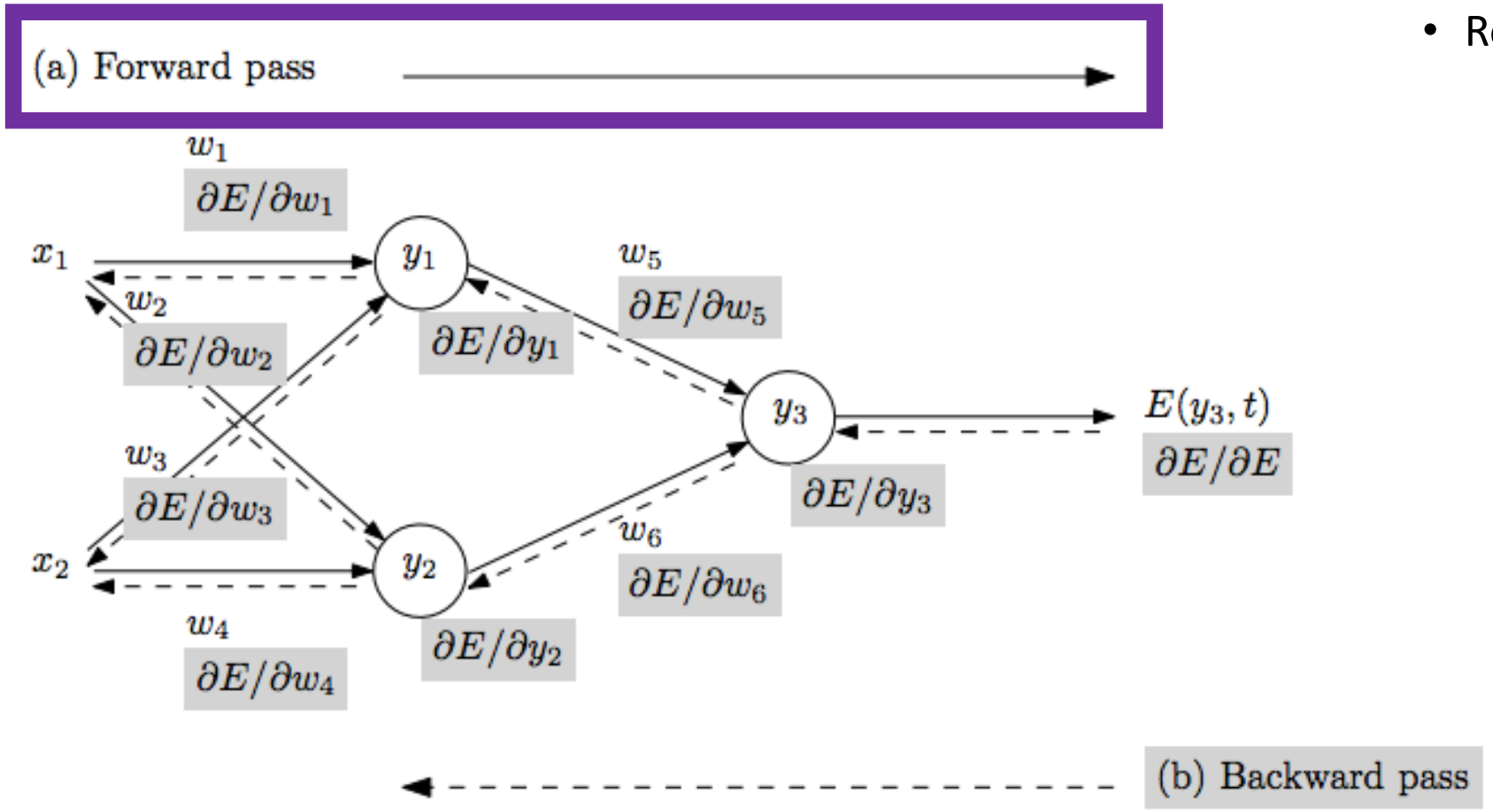
Example from: Jiawei Han and Micheline Kamber; Data Mining.

Example: Input Training Example



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

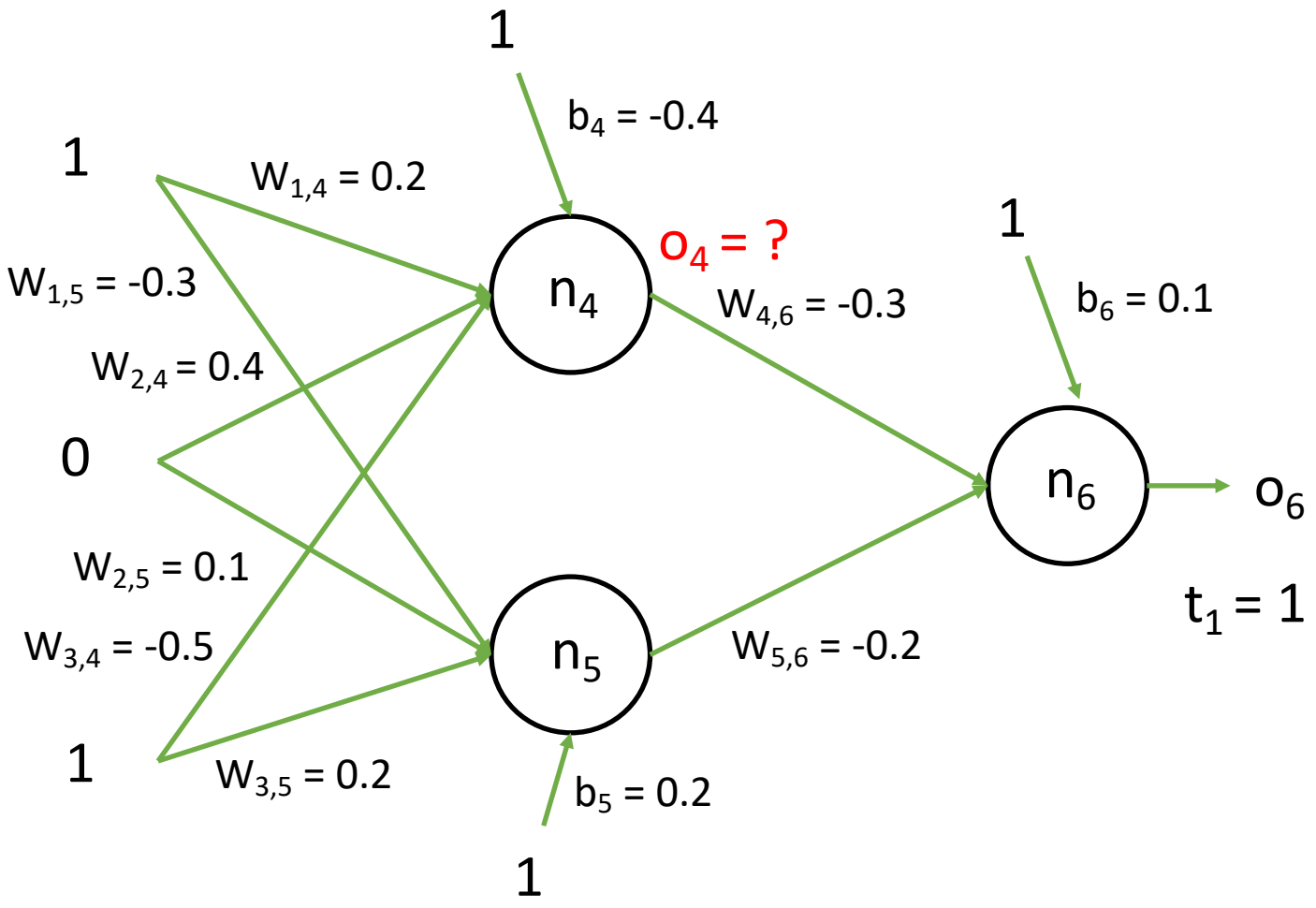
Example: Step 1 – Forward Pass



- 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



Weighted input sum to node 4:

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

$i_4 = -0.7$

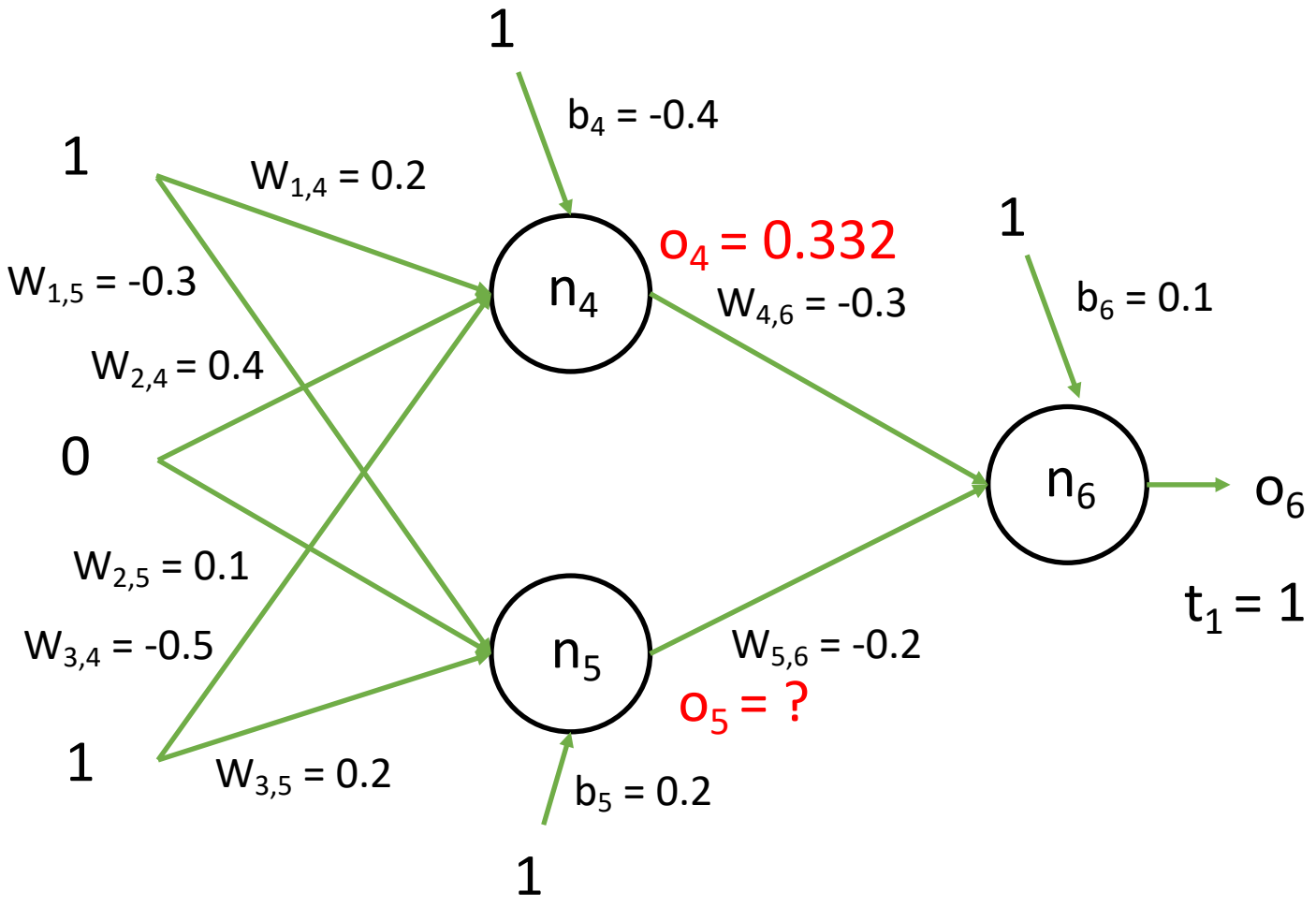
Activation result from node 4:

$o_4 = sigmoid(-0.7)$

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

$o_4 = 0.332$

Example: Step 1 – Forward Pass



Weighted input sum to node 5:

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

$i_5 = 0.1$

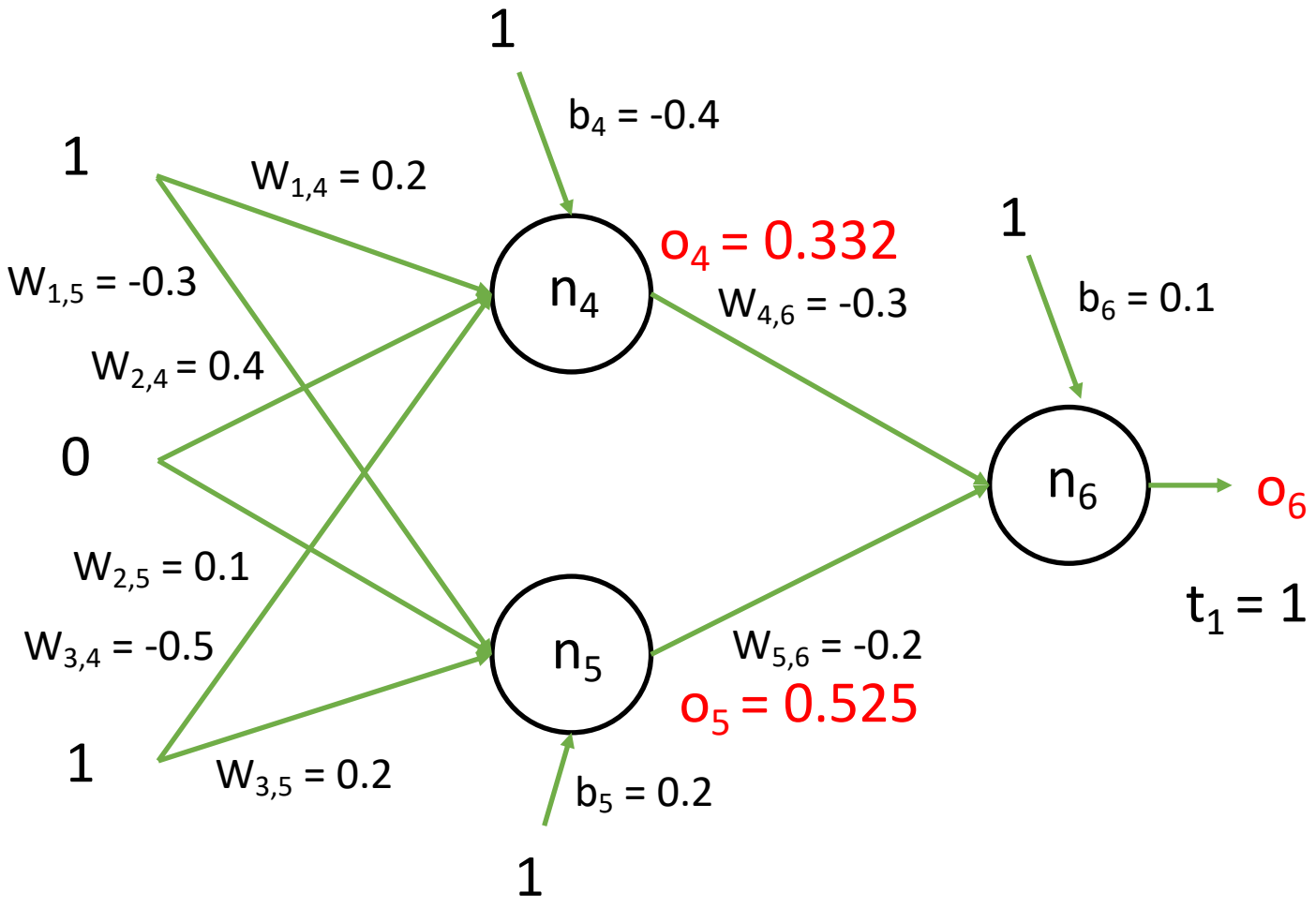
Activation result from node 5:

$o_5 = sigmoid(0.1)$

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

$o_5 = 0.525$

Example: Step 1 – Forward Pass



Weighted input sum to node 6:

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

$i_6 = -0.105$

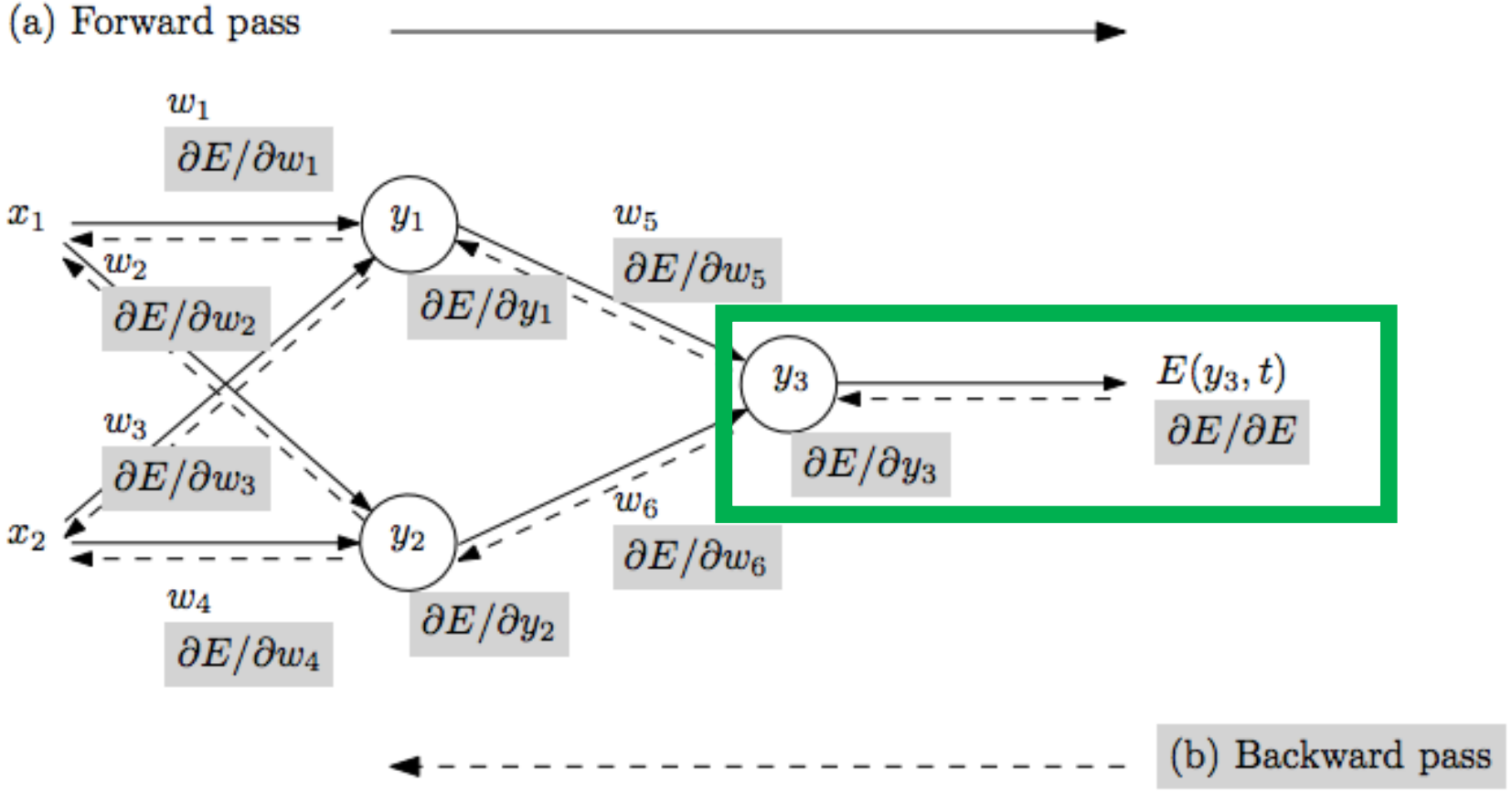
Activation result from node 6:

$o_6 = sigmoid(-0.105)$

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

$o_6 = 0.474$

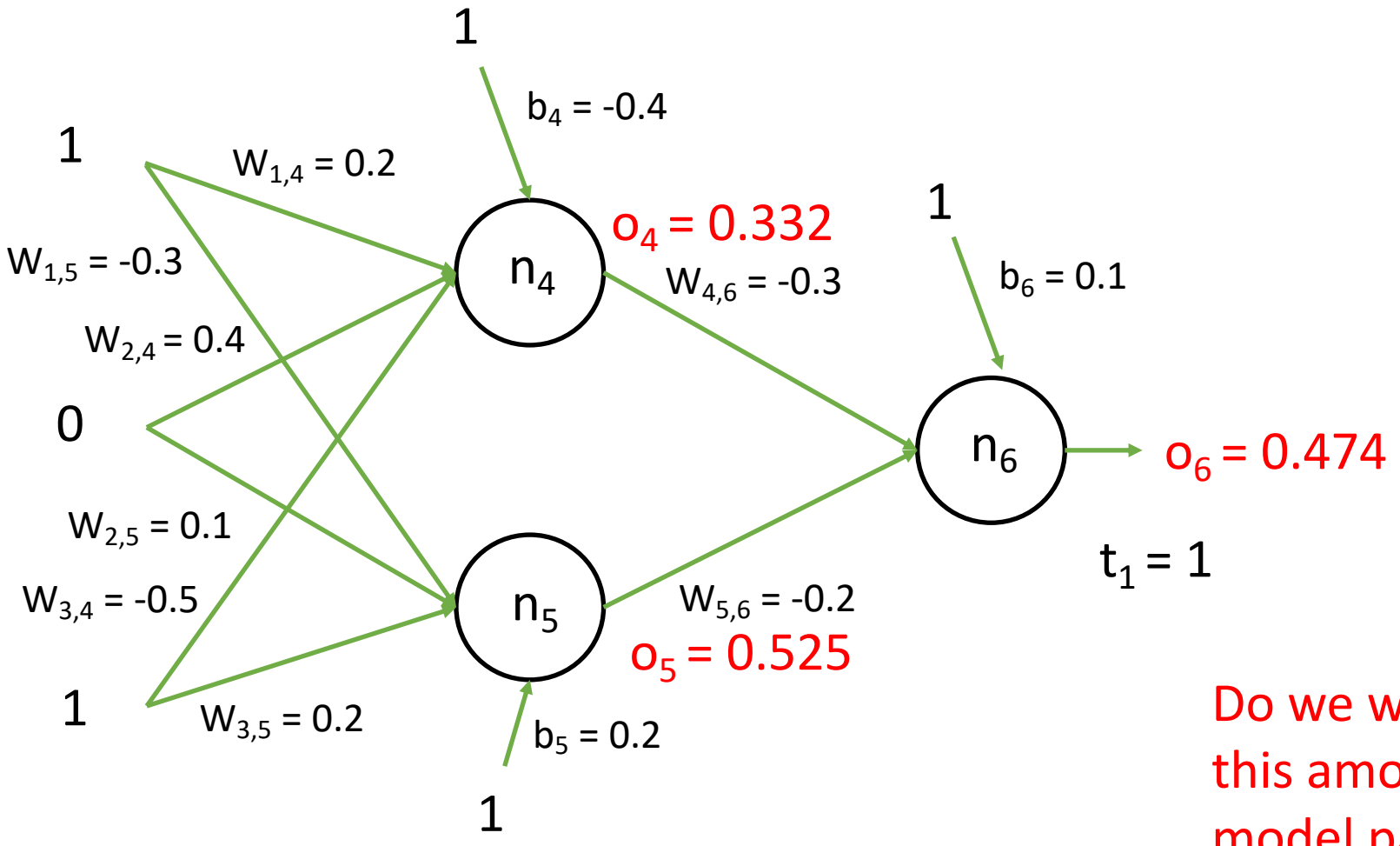
Example: Step 2 – Calculate Error (Loss)



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data

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 – Calculate Error (Loss)



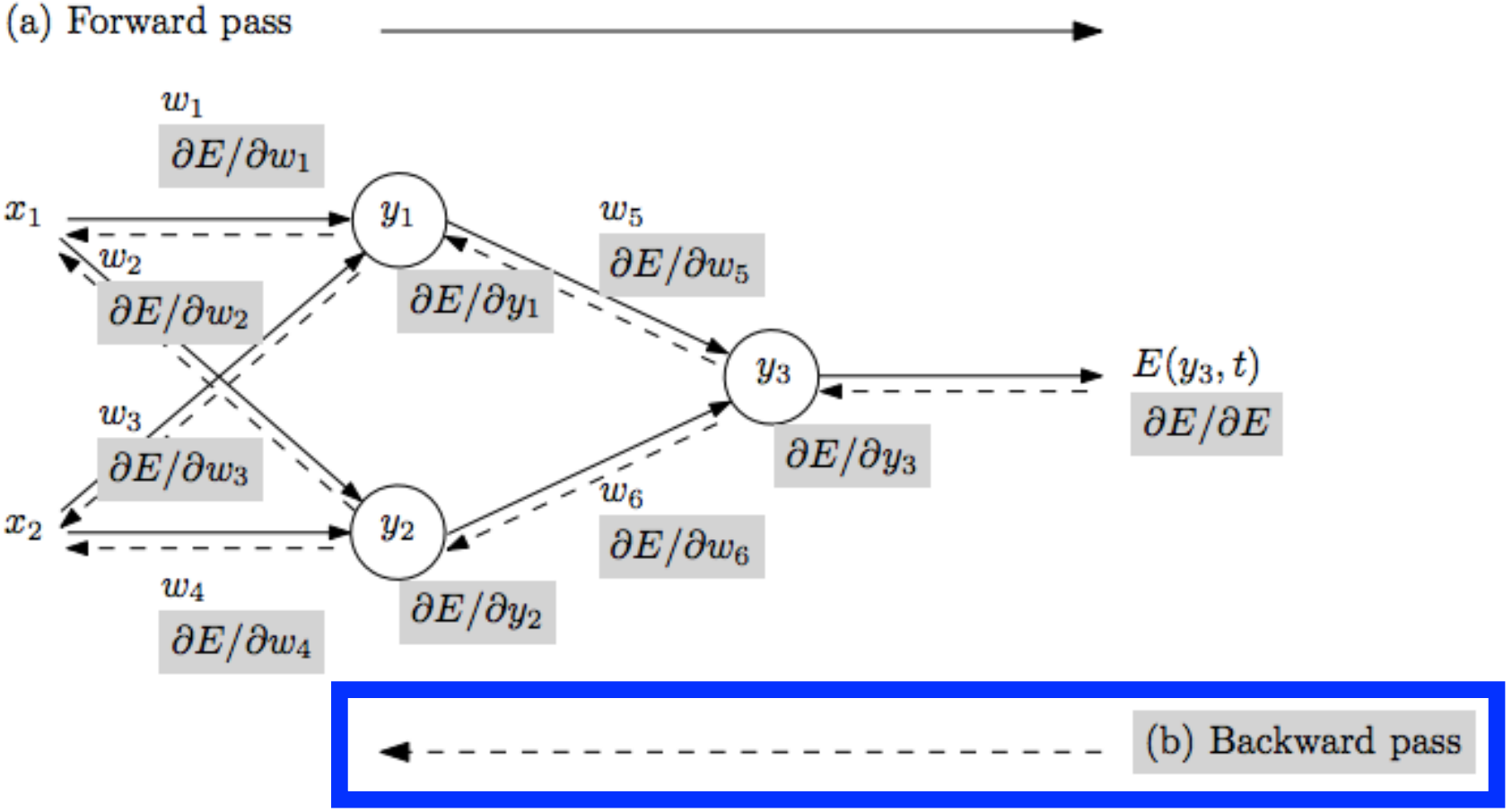
Squared Error

$= (1-0.474)^2$

$= 0.277$

Do we want to increase or reduce this amount when modifying the model parameters during training?

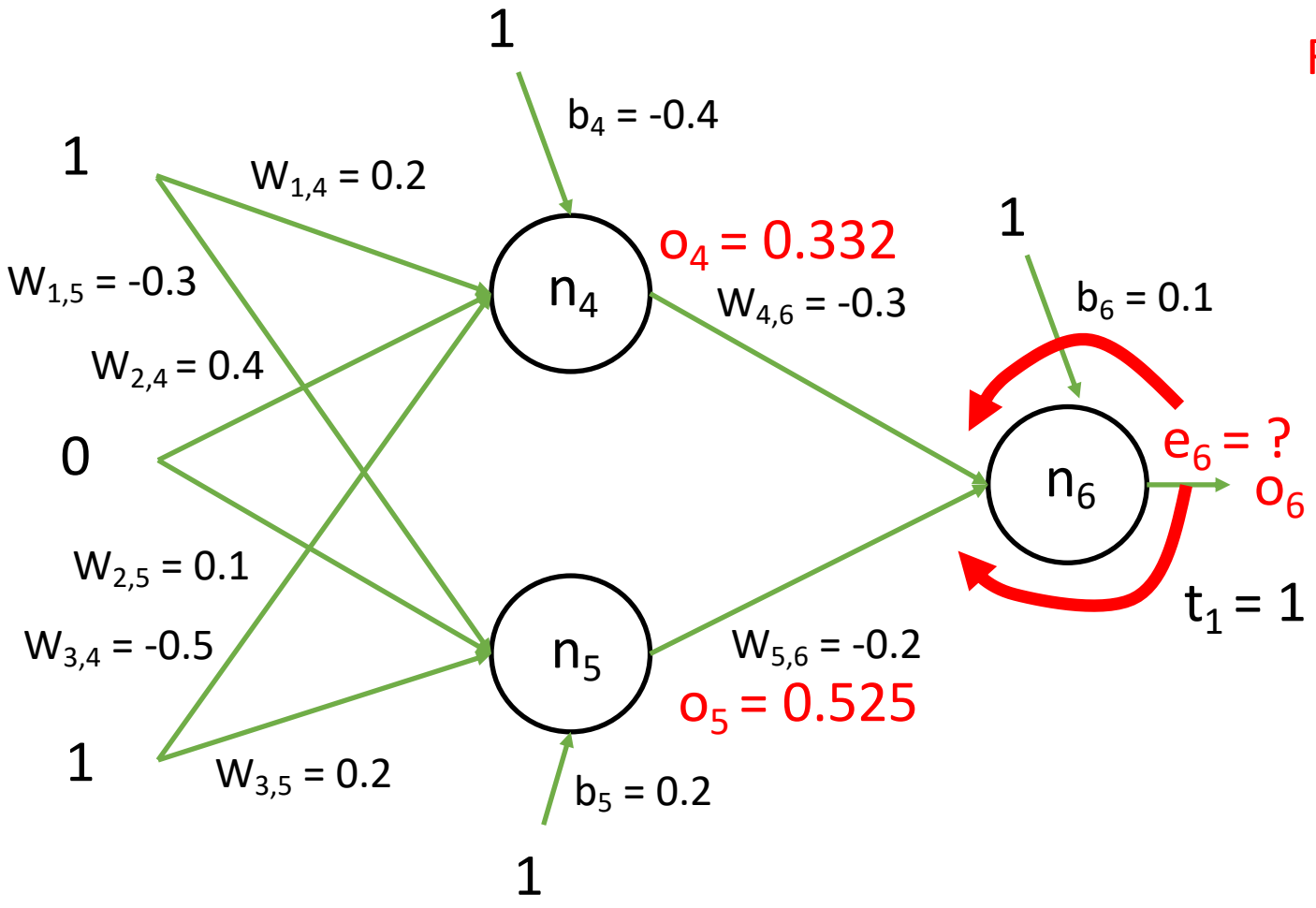
Example: Step 3 – Backward Pass



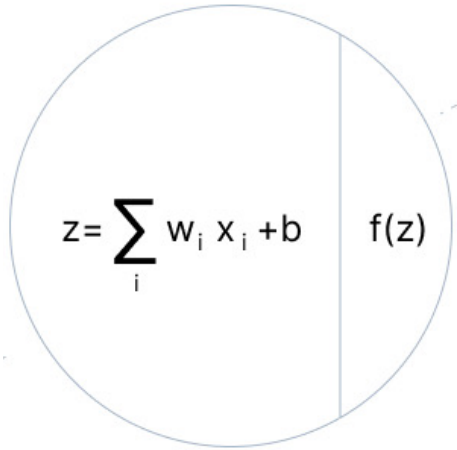
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure dissatisfaction with a model's predictions on training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign 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 3 – Backward Pass



Recall, each node contains 2 functions:



- let o_k represent the activation
- let z_k represent the weighted sum

For node 6, we will first compute:

$$\frac{\partial E}{\partial z_6} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6}$$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

Loss function:

constant value

$$\frac{\partial E}{\partial o_6} = \frac{\partial}{\partial o_6}(t_k - o_6)^2 = -2(t_6 - o_6)$$

Activation function (sigmoid):

$$\frac{\partial o_6}{\partial z_6} = \frac{\partial}{\partial z_6}\frac{1}{1+e^{-z_6}} = (1-\sigma(z_6))\,\sigma(z_6)$$

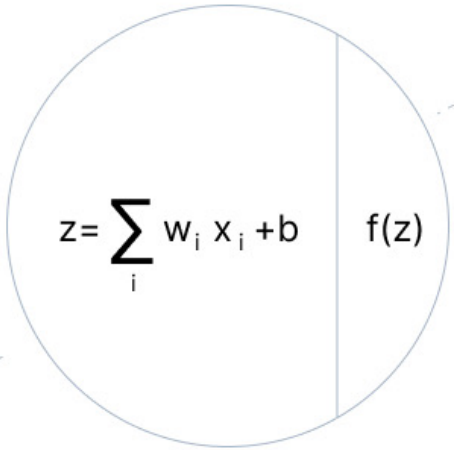
$$= (1 - o_6))\ o_6$$

Putting them together:

$$\frac{\partial E}{\partial z_6} = -2(t_6 - o_6)\ (1 - o_6))\ o_6$$

Can drop constant (does not affect learning)

Recall, each node contains 2 functions:



- let o_k represent the activation
- let z_k represent the weighted sum

For node 6, we will first compute:

$$\frac{\partial E}{\partial z_6} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6}$$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

Key Observation: Can compute this because both the activation function and loss function are **differentiable**!!!

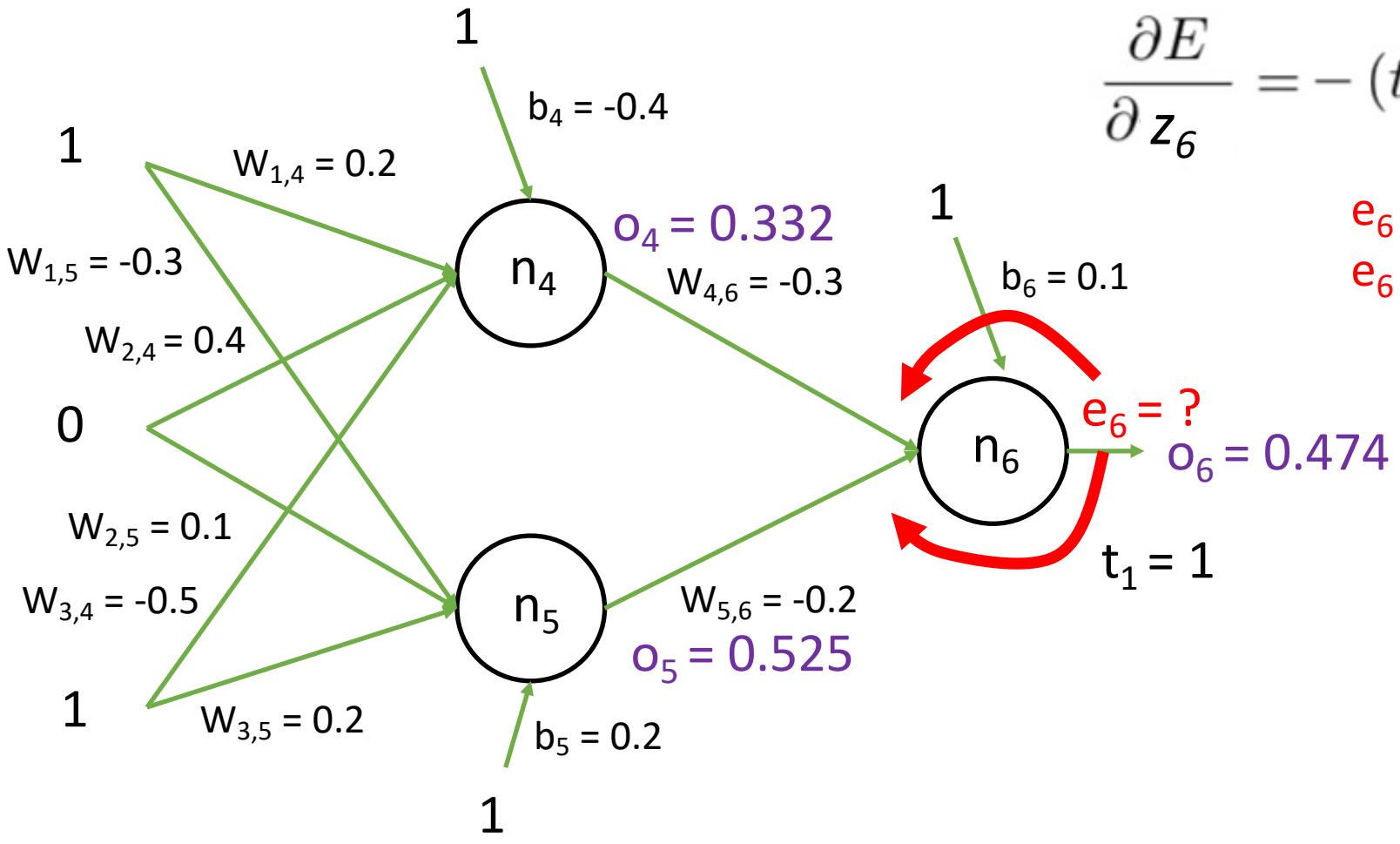
$$\frac{\partial E}{\partial z_6} = -2(t_6 - o_6) \; (1 - o_6)) \; o_6$$

Can drop constant (does not affect learning)

$$\frac{\partial E}{\partial z_6} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6}$$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

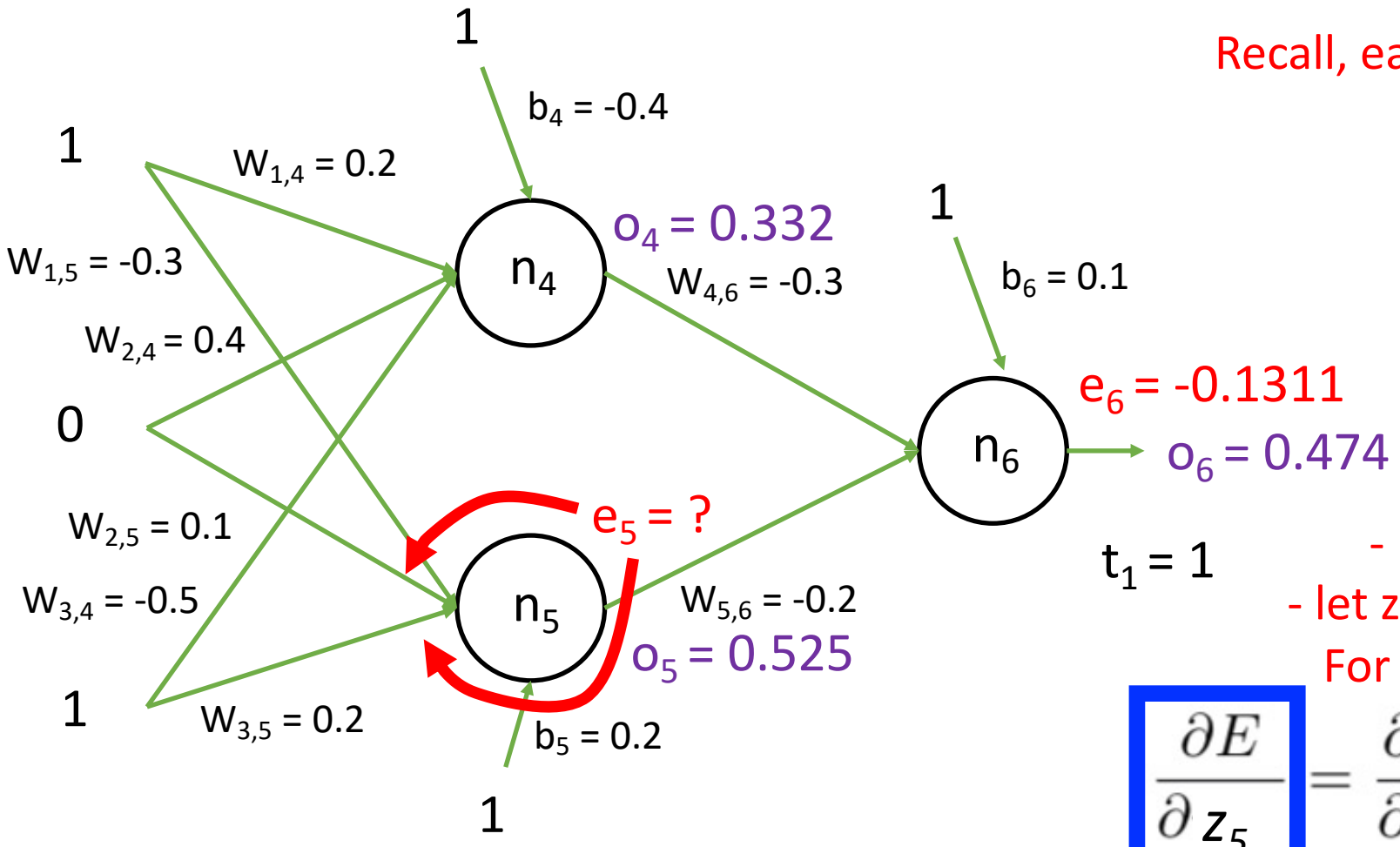


$$\frac{\partial E}{\partial z_6} = -(t_6 - o_6)(1 - o_6))\ o_6$$

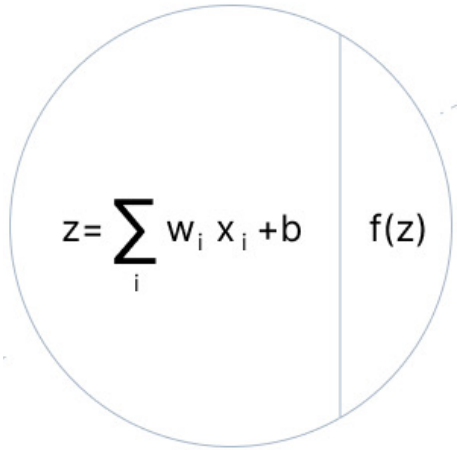
$e_6 = -(1-0.474)(1-0.474)(0.474)$

$e_6 = -0.1311$

Example: Step 3 – Backward Pass



Recall, each node contains 2 functions:



- let o_k represent the activation
- let z_k represent the weighted sum

For node 5, we will first compute:

$$\frac{\partial E}{\partial z_5} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_5} * \frac{\partial o_5}{\partial z_5}$$

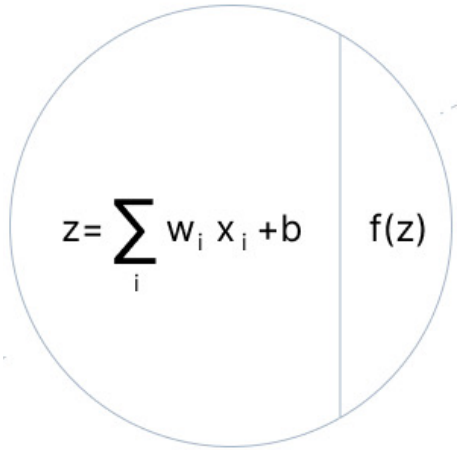
Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

Parent node:

$$\frac{\partial z_6}{\partial o_5} = \frac{\partial}{\partial o_5}(o_4 w_{4,6} + o_5 w_{5,6} + b_6) = w_{5,6}$$

Recall, each node contains 2 functions:



Activation function (sigmoid):

$$\frac{\partial o_5}{\partial z_5} = \frac{\partial}{\partial z_5}\frac{1}{1+e^{-z_6}} = (1-\sigma(z_5))\,\sigma(z_5)$$
$$= (1 - o_5))\, o_5$$

- let o_k represent the activation
- let z_k represent the weighted sum

Putting them together:

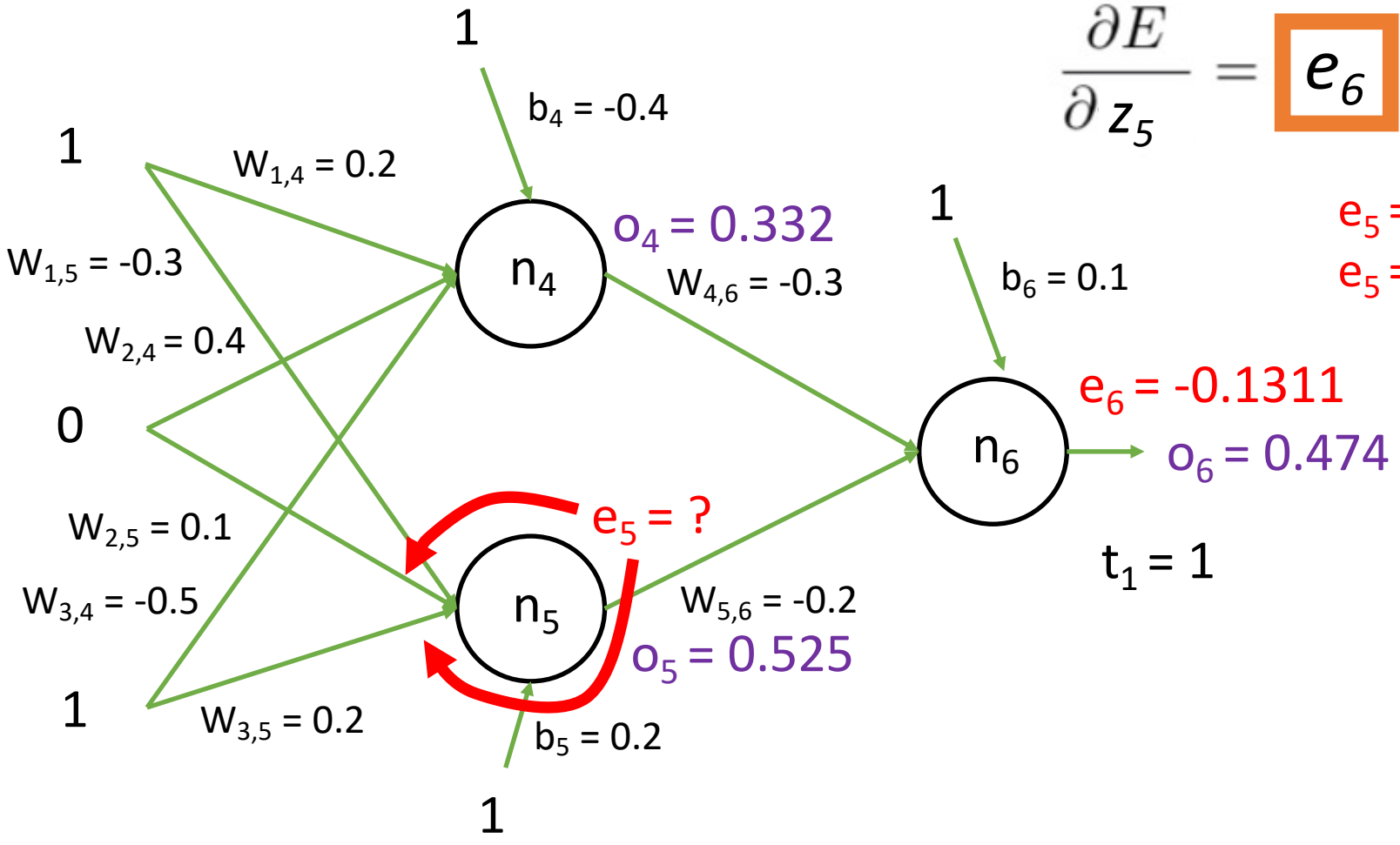
$$\frac{\partial E}{\partial z_5} = e_6 \; w_{5,6} \; (1 - o_5))\, o_5$$

For node 5, we will first compute:

$$\frac{\partial E}{\partial z_5} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_5} * \frac{\partial o_5}{\partial z_5}$$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass



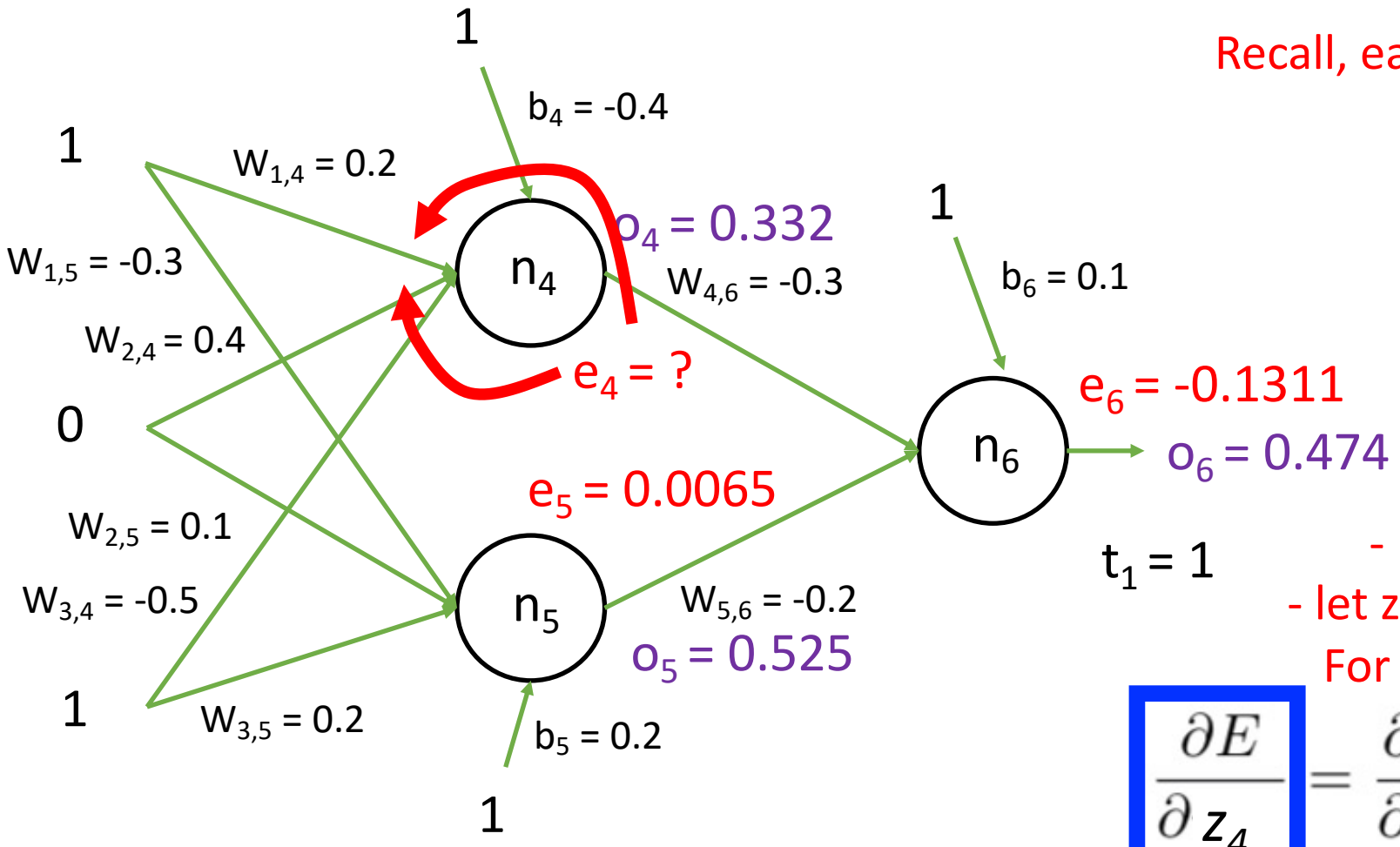
$$\frac{\partial E}{\partial z_5} = e_6 \; w_{5,6} \; (1 - o_5)) \; o_5$$

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

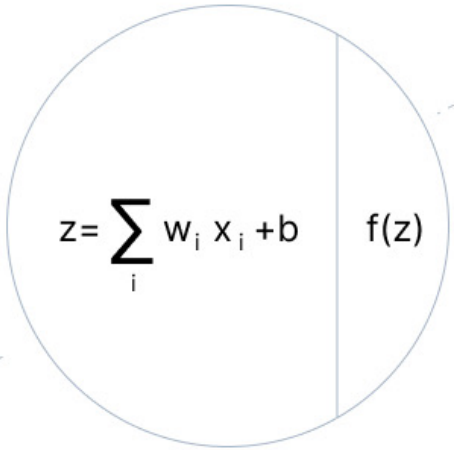
$e_5 = 0.0065$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass



Recall, each node contains 2 functions:



- let o_k represent the activation
- let z_k represent the weighted sum

For node 5, we will first compute:

$$\frac{\partial E}{\partial z_4} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_4} * \frac{\partial z_6}{\partial o_4} * \frac{\partial o_4}{\partial z_4}$$

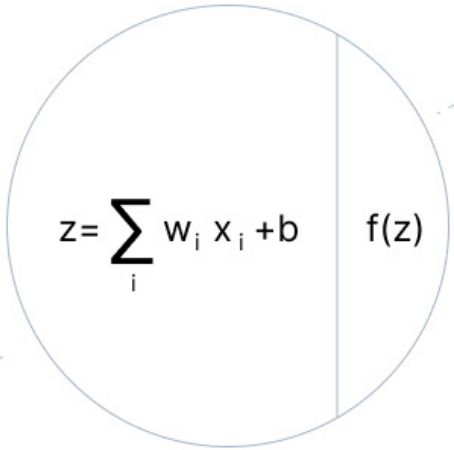
Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

Parent node:

$$\frac{\partial z_6}{\partial o_4} = \frac{\partial}{\partial o_4}(o_4 w_{4,6} + o_5 w_{5,6} + b_6) = w_{4,6}$$

Recall, each node contains 2 functions:



Activation function (sigmoid):

$$\frac{\partial o_4}{\partial z_4} = \frac{\partial}{\partial z_4}\frac{1}{1+e^{-z_6}} = (1-\sigma(z_4))\,\sigma(z_4)$$
$$= (1 - o_4))\, o_4$$

- let o_k represent the activation
- let z_k represent the weighted sum

Putting them together:

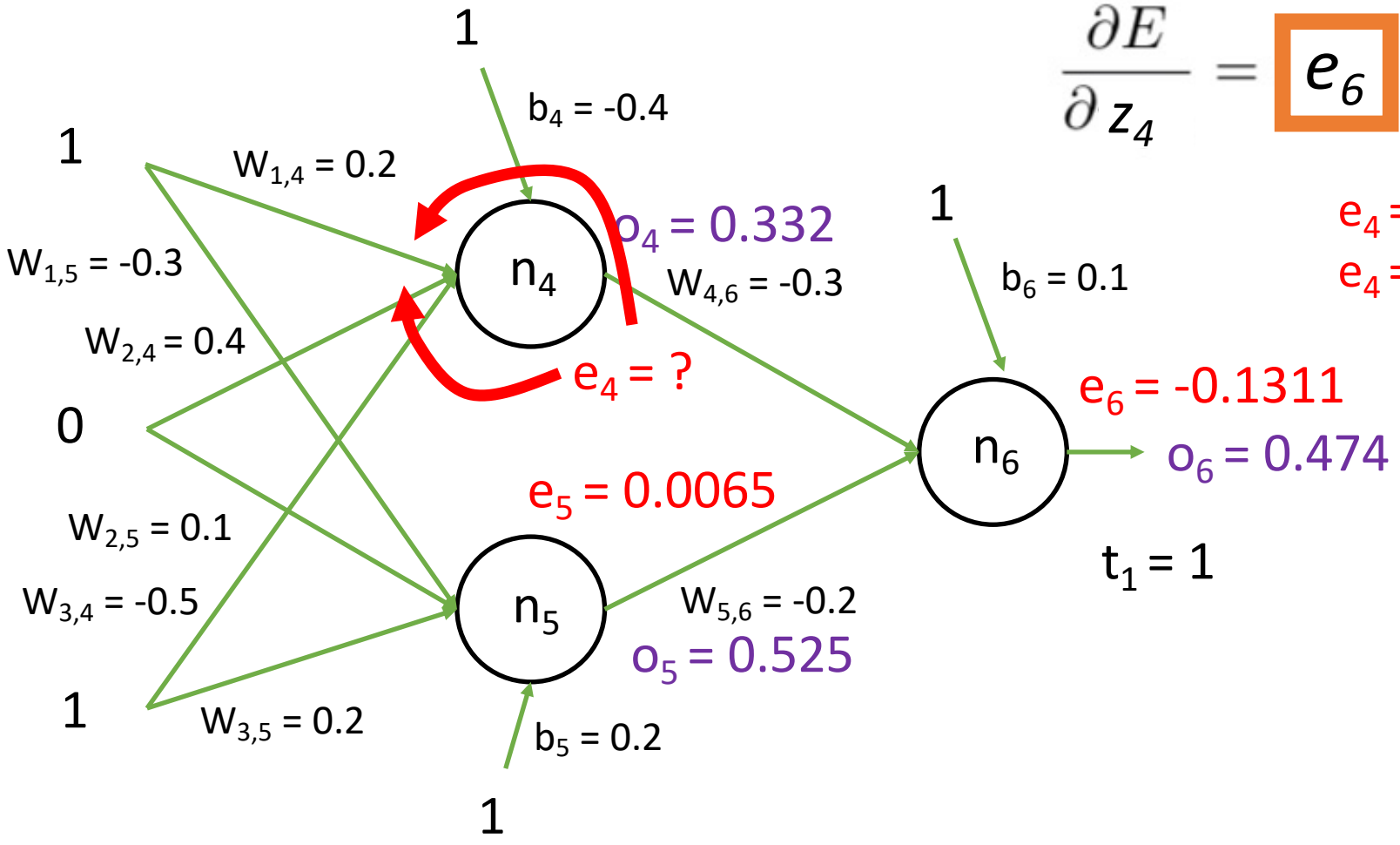
$$\frac{\partial E}{\partial z_4} = e_6\, w_{4,6}\, (1 - o_4))\, o_4$$

For node 5, we will first compute:

$$\frac{\partial E}{\partial z_4} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_4} * \frac{\partial o_4}{\partial z_4}$$

Image source: https://www.v7labs.com/blog/neural-networks-activation-functions

Example: Step 3 – Backward Pass

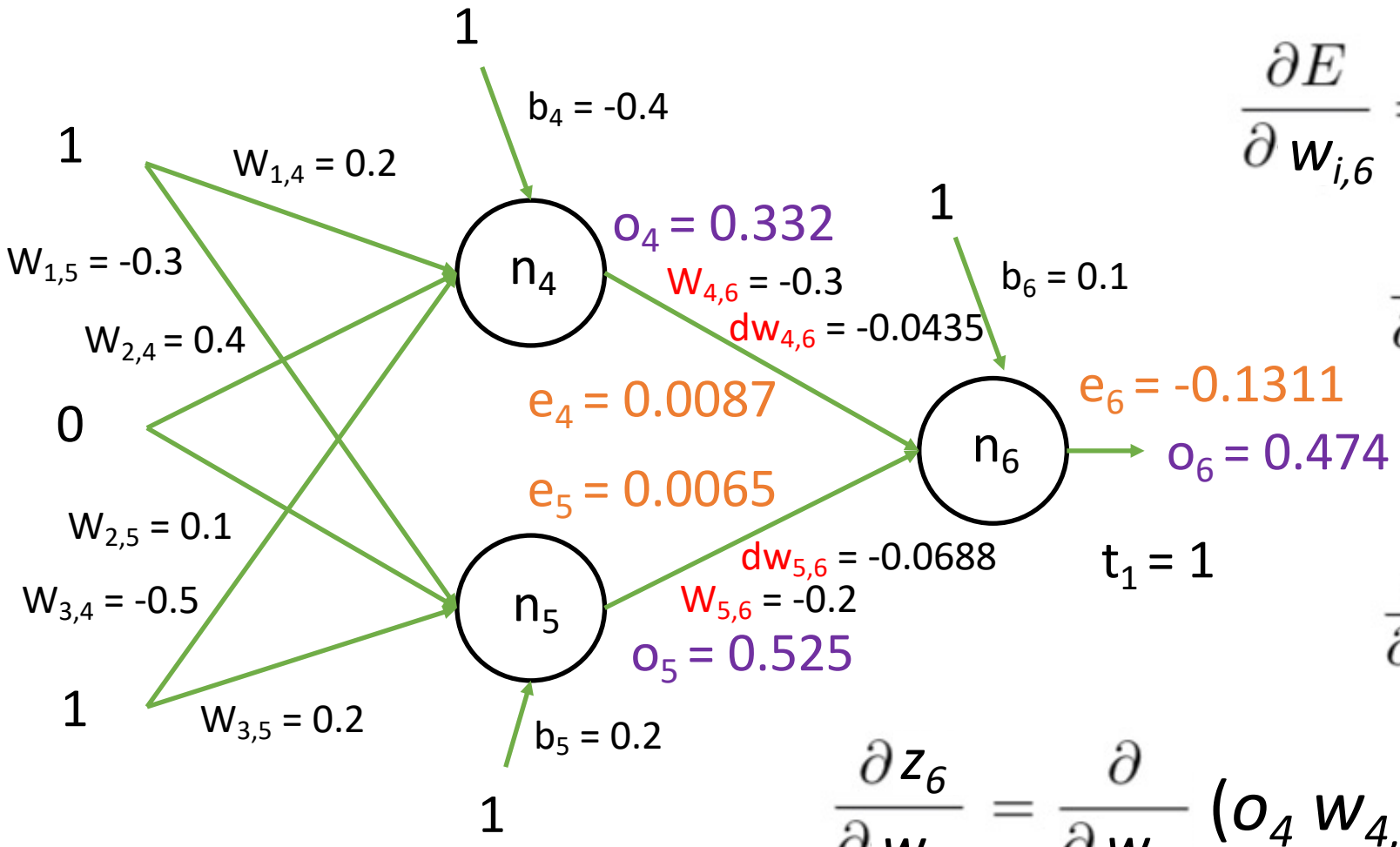


$$\frac{\partial E}{\partial z_4} = e_6 \; w_{4,6} \; (1 - o_4)) \; o_4$$

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

$e_4 = 0.0087$

Example: Step 3 – Backward Pass



$$\frac{\partial E}{\partial w_{i,6}} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \boxed{\frac{\partial z_6}{\partial w_{i,6}}}$$

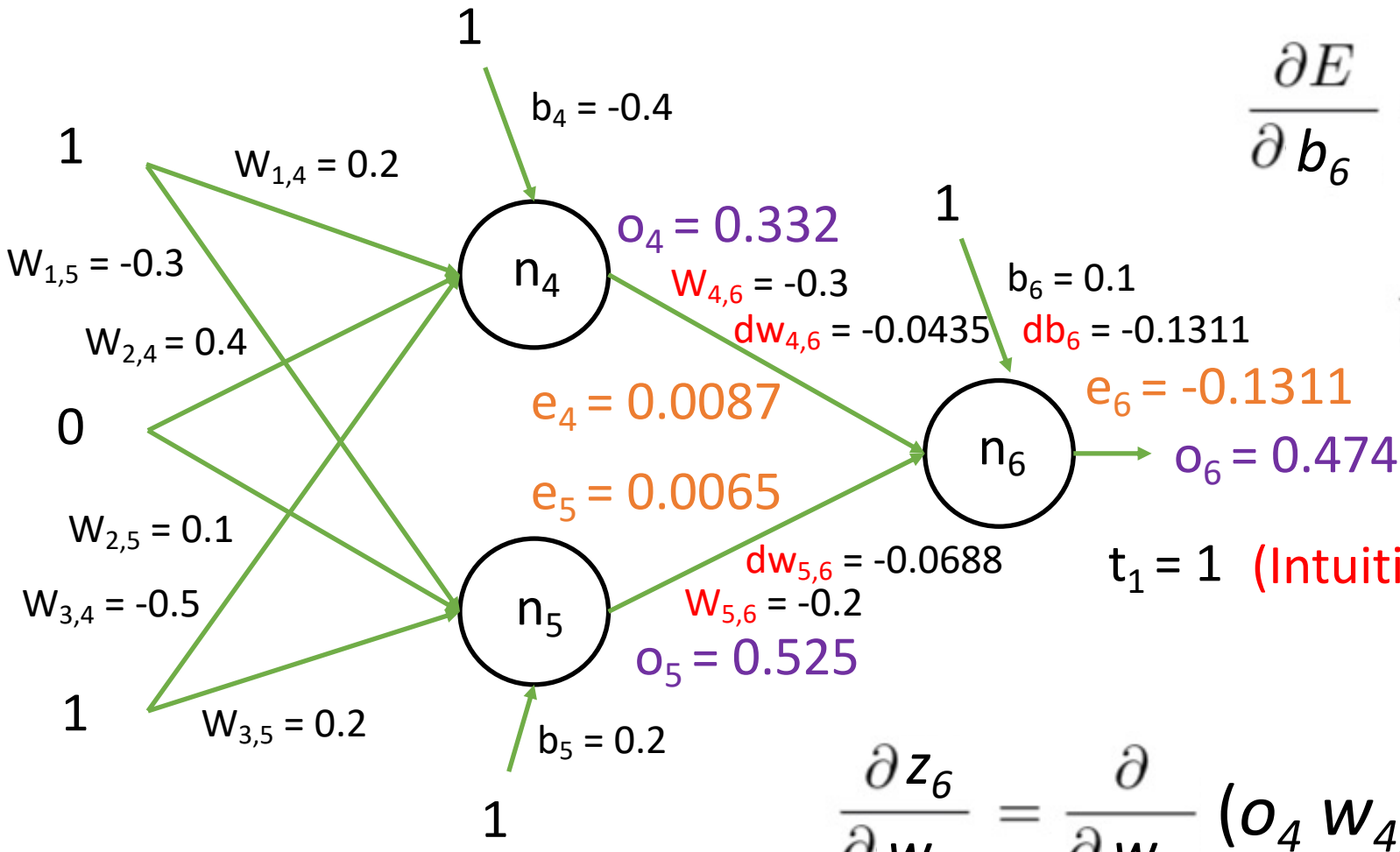
$$\frac{\partial E}{\partial w_{4,6}} = e_6 * o_4 = -0.0435$$

Recall, gradient points to direction of larger value!

$$\frac{\partial E}{\partial w_{5,6}} = e_6 * o_5 = -0.0688$$

$$\frac{\partial z_6}{\partial w_{i,6}} = \frac{\partial}{\partial w_{i,6}}(o_4 w_{4,6} + o_5 w_{5,6} + b_6) = \boxed{o_i}$$

Example: Step 3 – Backward Pass



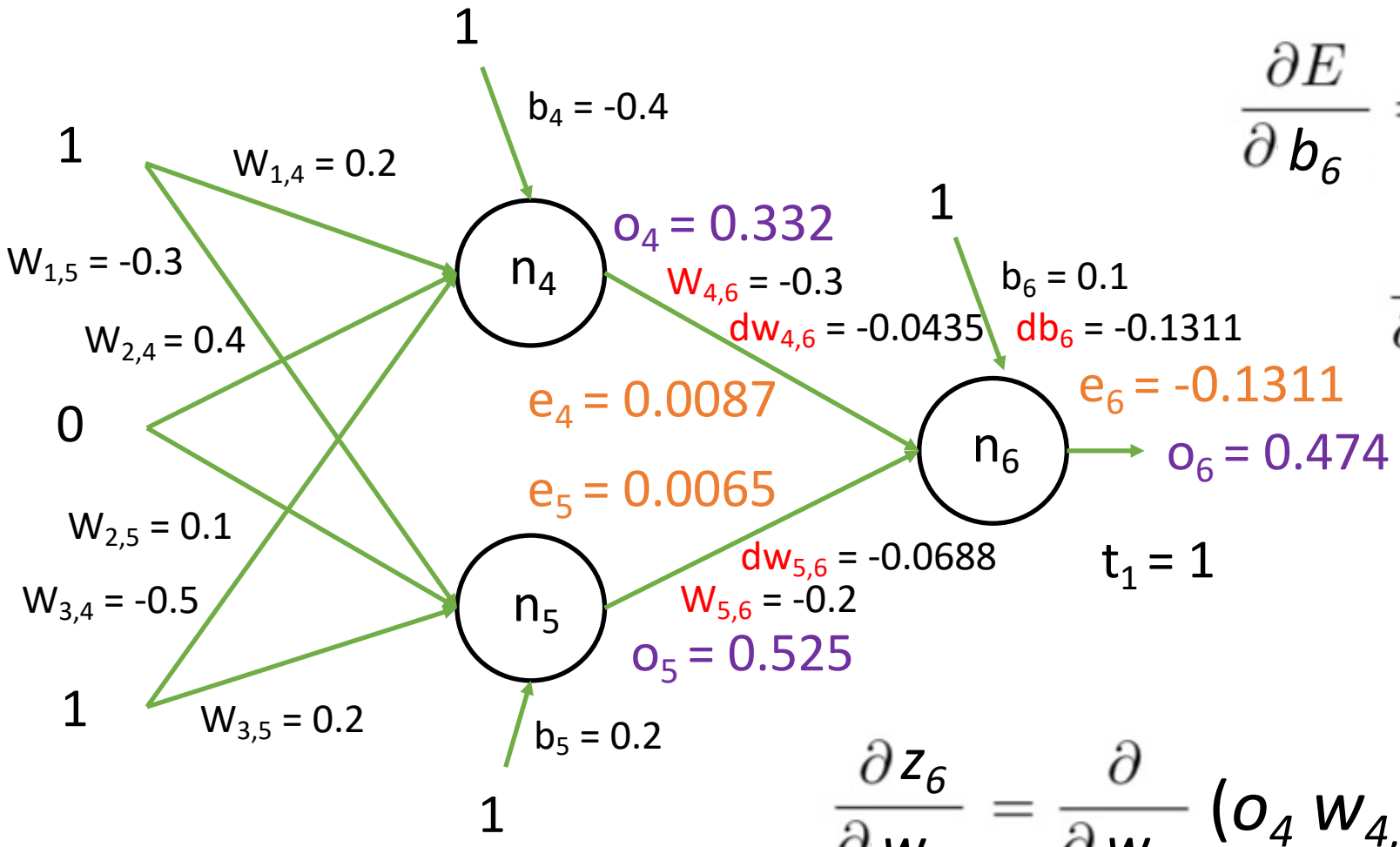
$$\frac{\partial E}{\partial b_6} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial b_6}$$

$$\frac{\partial E}{\partial b_6} = e_6 * 1 = -0.1311$$

$t_1 = 1$ (Intuitively, just passes error along)

$$\frac{\partial z_6}{\partial w_{i,6}} = \frac{\partial}{\partial w_{i,6}} (o_4 w_{4,6} + o_5 w_{5,6} + b_6) = 1$$

Example: Step 3 – Backward Pass



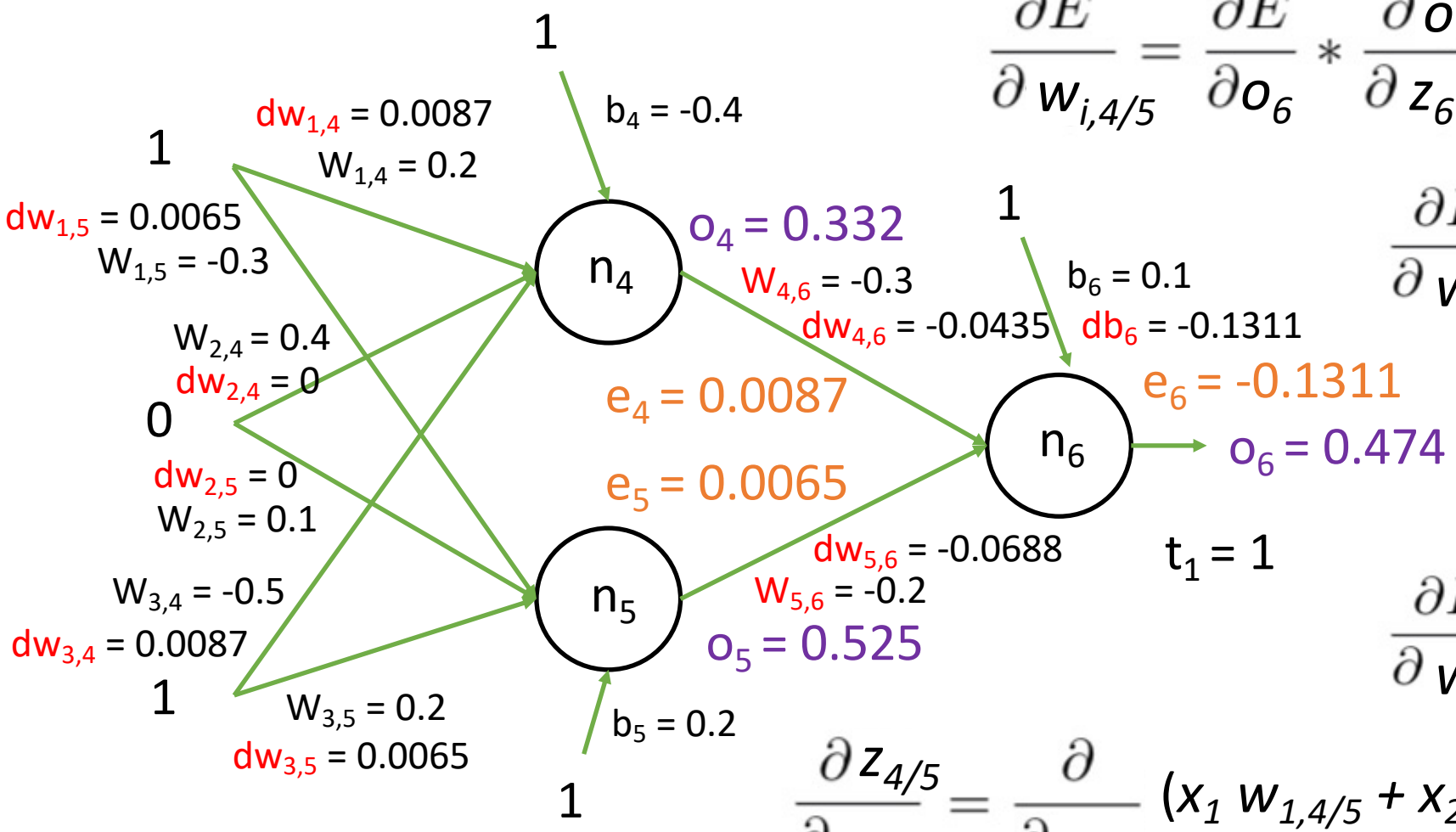
$$\frac{\partial E}{\partial b_6} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial b_6}$$

$$\frac{\partial E}{\partial b_6} = e_6 * 1 = -0.1311$$

Recall, gradient points to direction of larger value!

$$\frac{\partial z_6}{\partial w_{i,6}} = \frac{\partial}{\partial w_{i,6}} (o_4 w_{4,6} + o_5 w_{5,6} + b_6) = 1$$

Example: Step 3 – Backward Pass



$$\frac{\partial E}{\partial w_{i,4/5}} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_{4/5}} * \frac{\partial o_{4/5}}{\partial z_{4/5}} * \frac{\partial z_{4/5}}{\partial w_{i,4/5}}$$

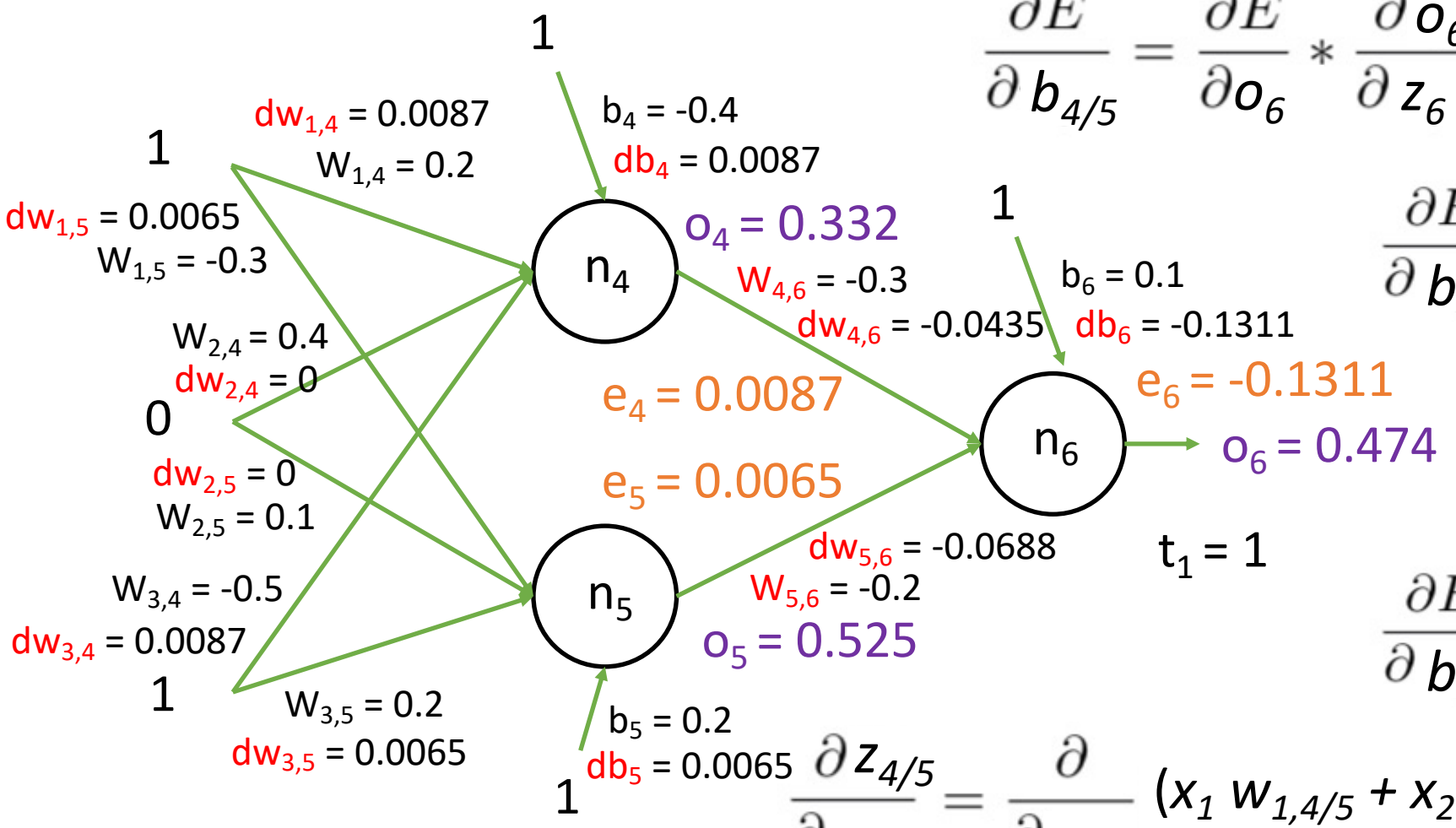
$$\frac{\partial E}{\partial w_{1,4}} = e_4 * x_1 = 0.0087$$

Recall, gradient points to direction of larger value!

$$\frac{\partial E}{\partial w_{3,5}} = e_5 * x_3 = 0.0065$$

$$\frac{\partial z_{4/5}}{\partial w_{i,4/5}} = \frac{\partial}{\partial w_{i,4/5}} (x_1 w_{1,4/5} + x_2 w_{2,4/5} + x_3 w_{3,4/5} + b_{4/5}) = x_i$$

Example: Step 3 – Backward Pass



$$\frac{\partial E}{\partial b_{4/5}} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_{4/5}} * \frac{\partial o_{4/5}}{\partial z_{4/5}} * \frac{\partial z_{4/5}}{\partial b_{4/5}}$$

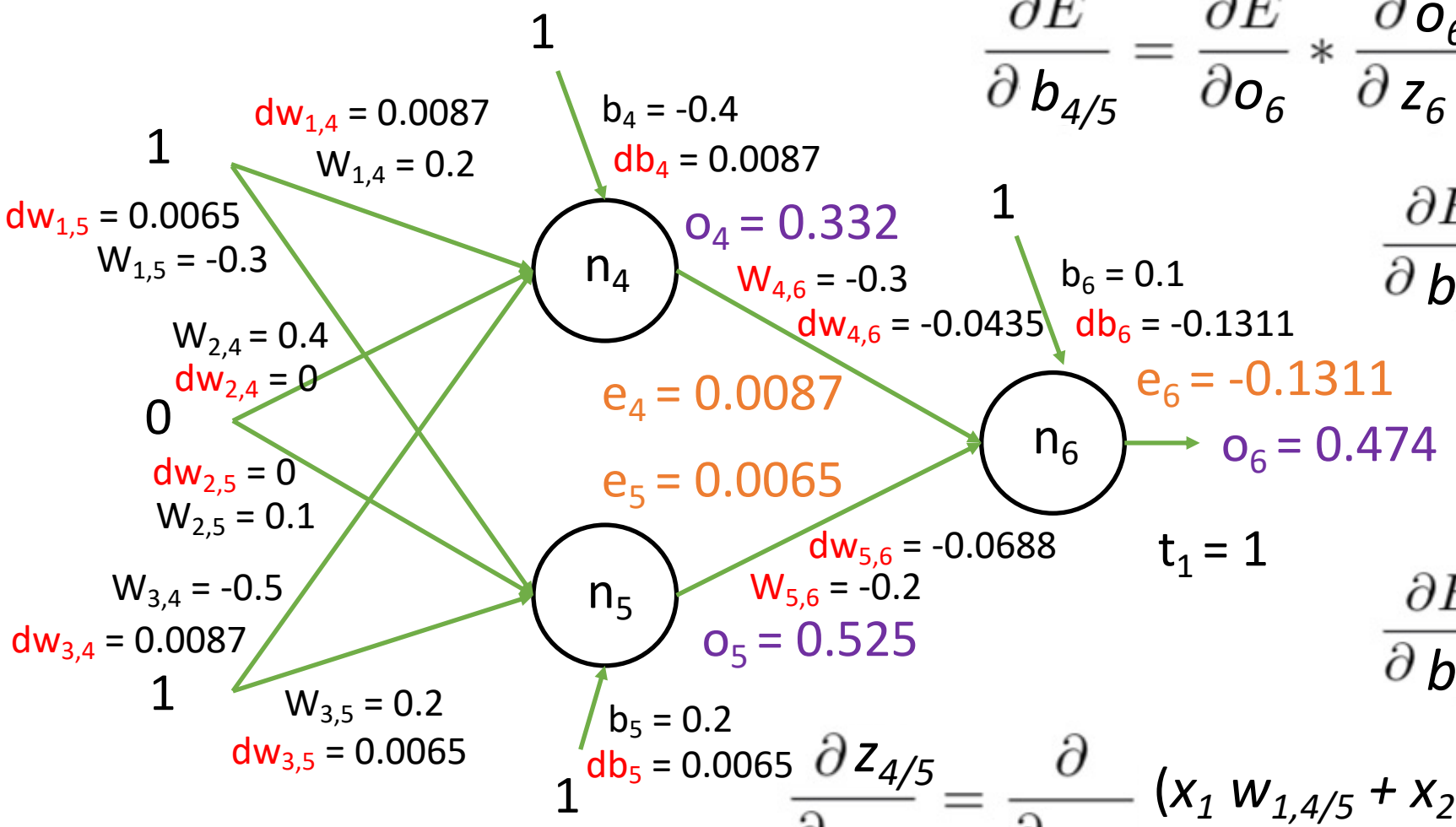
$$\frac{\partial E}{\partial b_4} = e_4 * 1 = 0.0087$$

(Just passes error along)

$$\frac{\partial E}{\partial b_5} = e_5 * 1 = 0.0065$$

$$\frac{\partial z_{4/5}}{\partial w_{i,4/5}} = \frac{\partial}{\partial w_{i,4/5}} (x_1 w_{1,4/5} + x_2 w_{2,4/5} + x_3 w_{3,4/5} + b_{4/5}) = 1$$

Example: Step 3 – Backward Pass



$$\frac{\partial E}{\partial b_{4/5}} = \frac{\partial E}{\partial o_6} * \frac{\partial o_6}{\partial z_6} * \frac{\partial z_6}{\partial o_{4/5}} * \frac{\partial o_{4/5}}{\partial z_{4/5}} * \frac{\partial z_{4/5}}{\partial b_{4/5}}$$

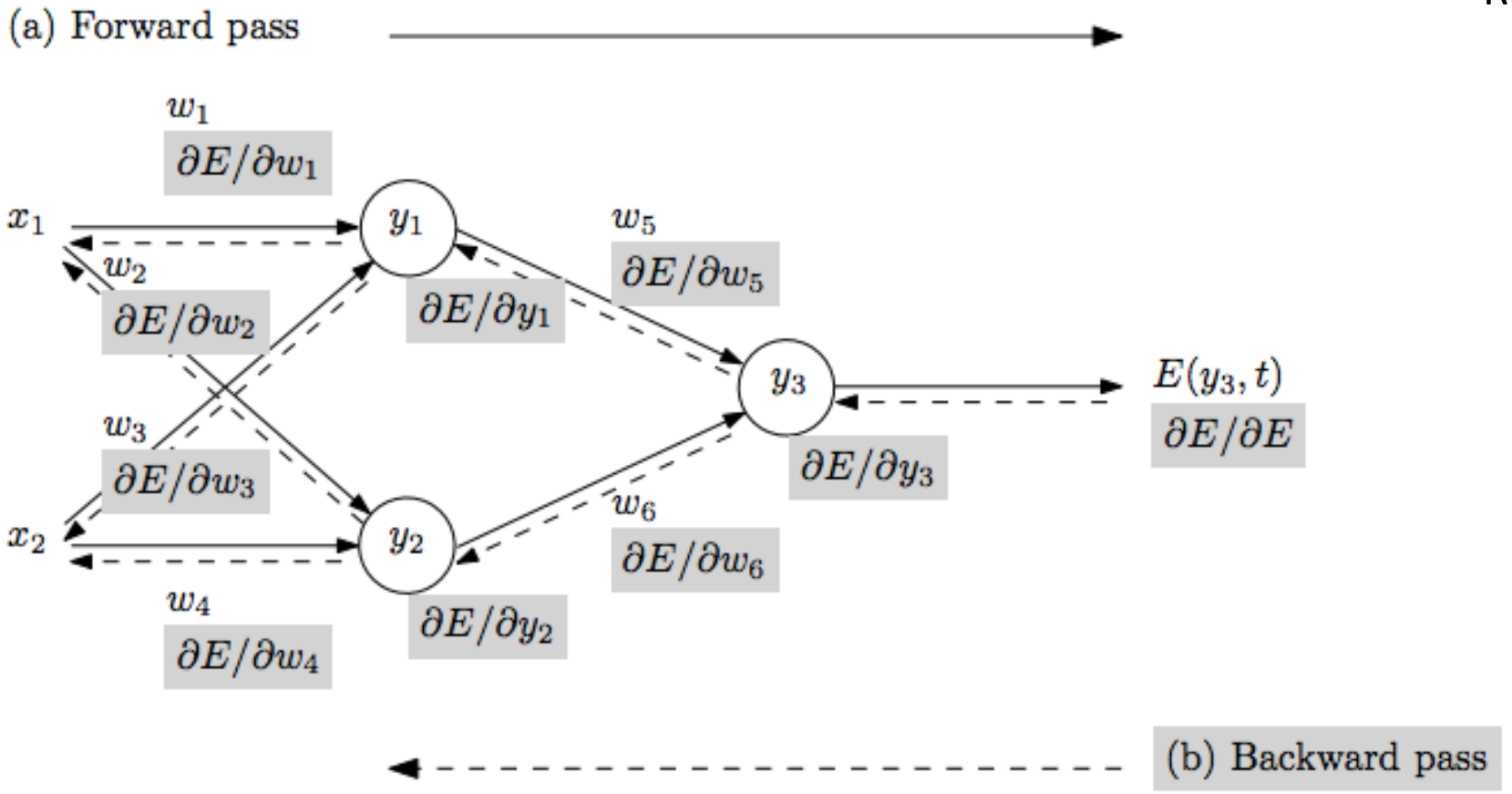
$$\frac{\partial E}{\partial b_4} = e_4 * 1 = 0.0087$$

Recall, gradient points to direction of larger value!

$$\frac{\partial E}{\partial b_5} = e_5 * 1 = 0.0065$$

$$\frac{\partial z_{4/5}}{\partial w_{i,4/5}} = \frac{\partial}{\partial w_{i,4/5}} (x_1 w_{1,4/5} + x_2 w_{2,4/5} + x_3 w_{3,4/5} + b_{4/5}) = 1$$

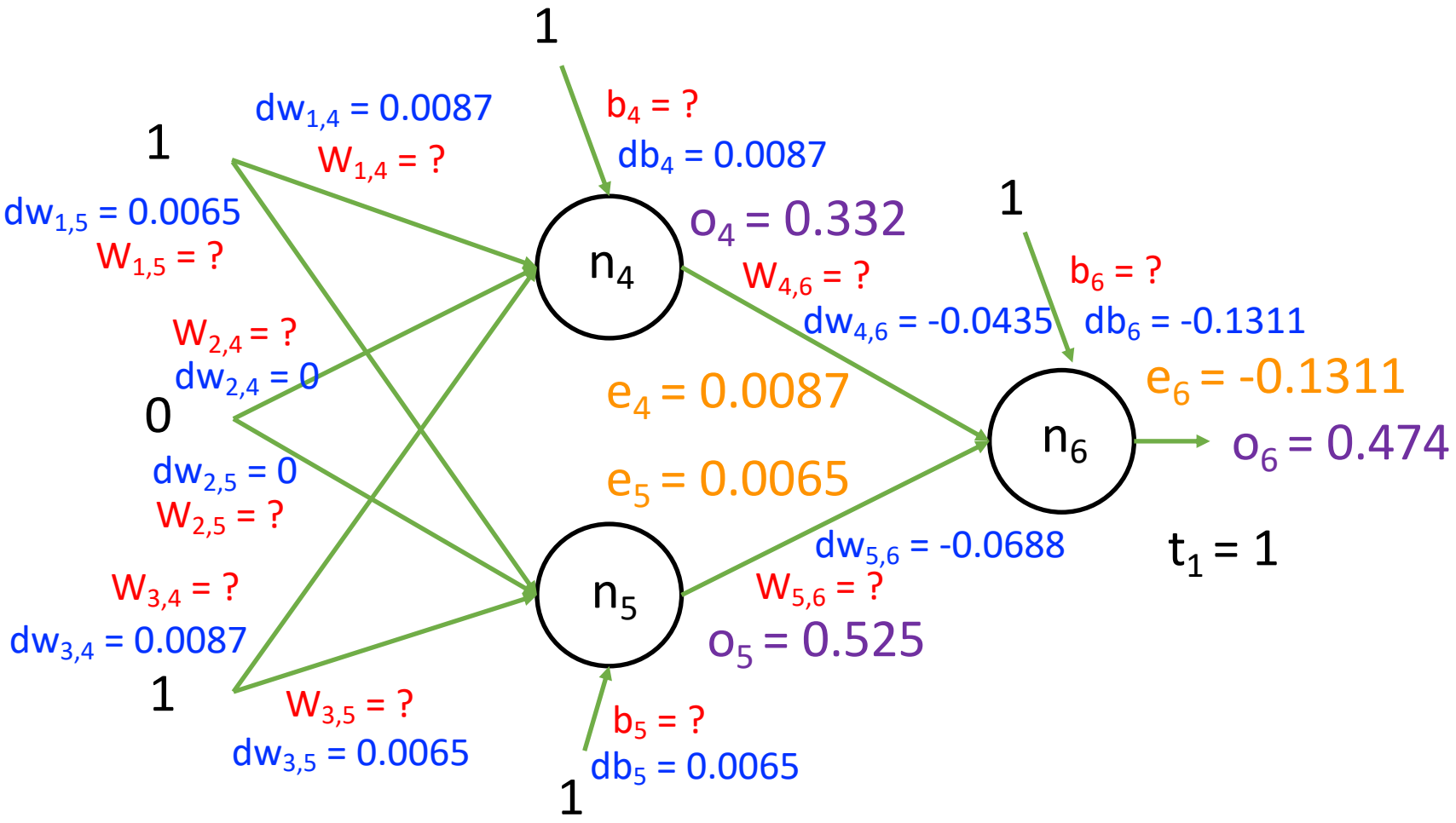
Example: Step 4 – Update Weights



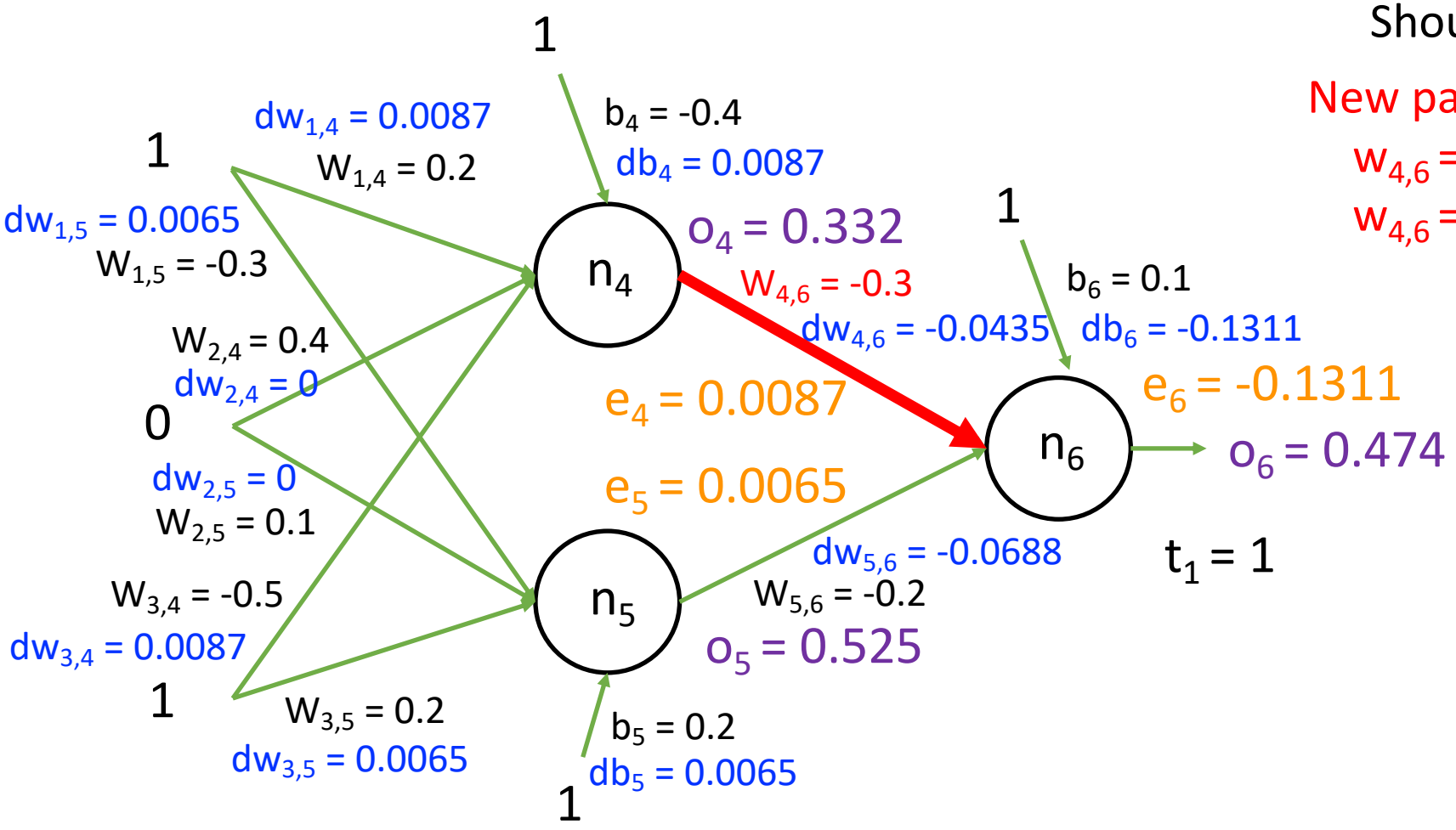
- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model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



Example: Step 4 – Update Weights



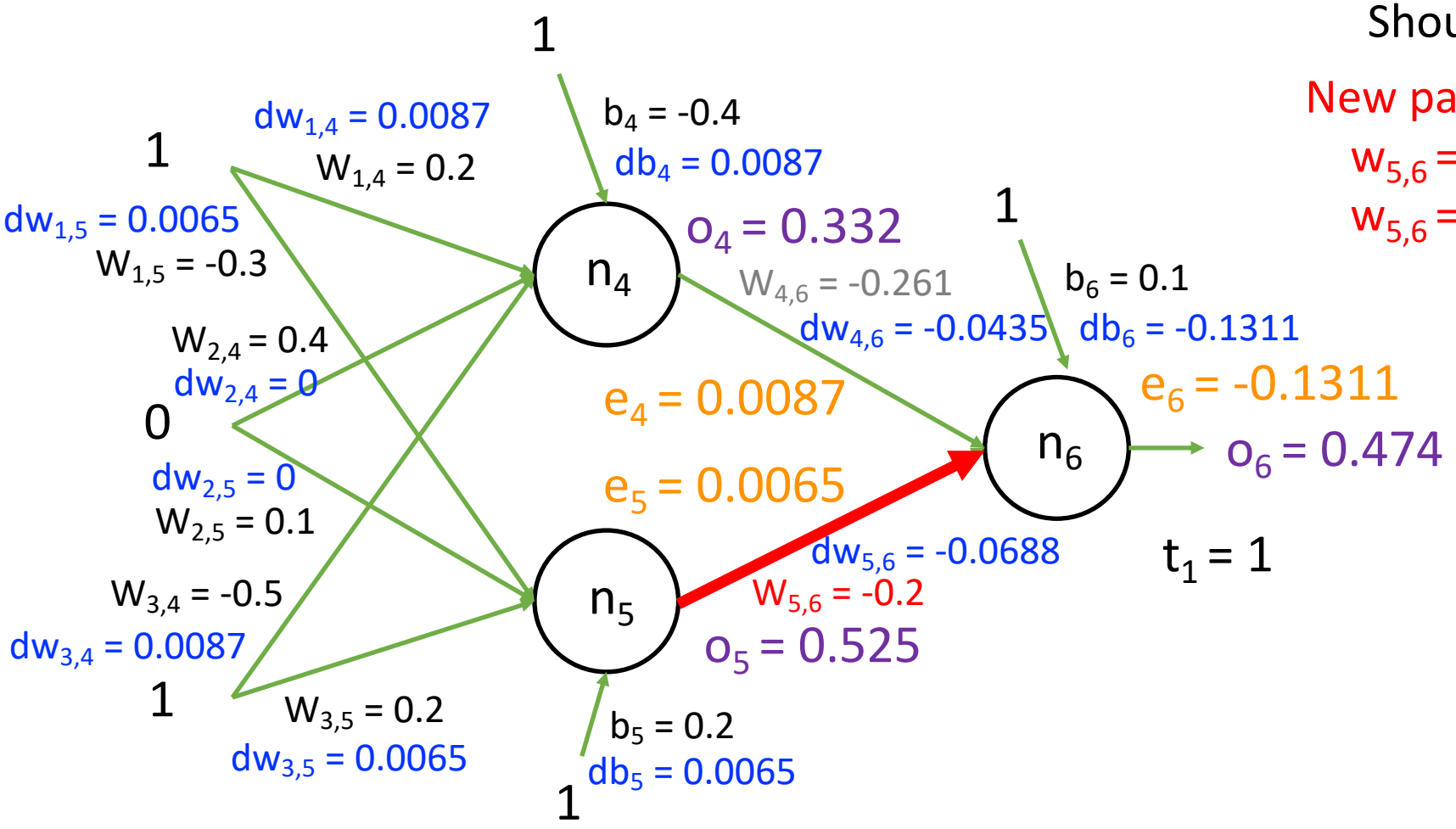
Should $w_{4,6}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{4,6}$ = -0.3 - 0.9 * -0.0435

$w_{4,6}$ = -0.261

Example: Step 4 – Update Weights



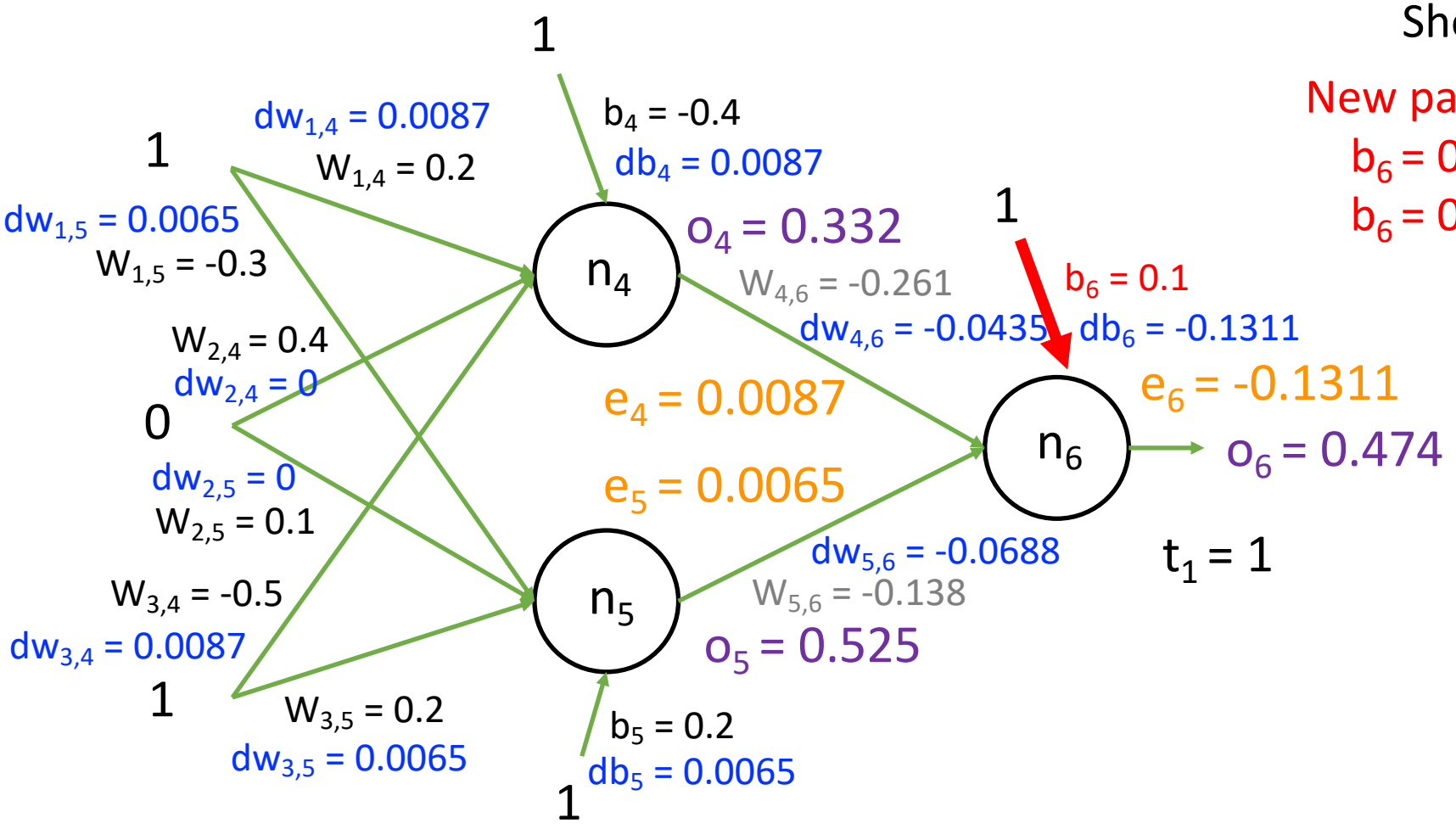
Should $w_{5,6}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{5,6} = -0.2 - 0.9 * -0.0688$

$w_{5,6} = -0.138$

Example: Step 4 – Update Weights



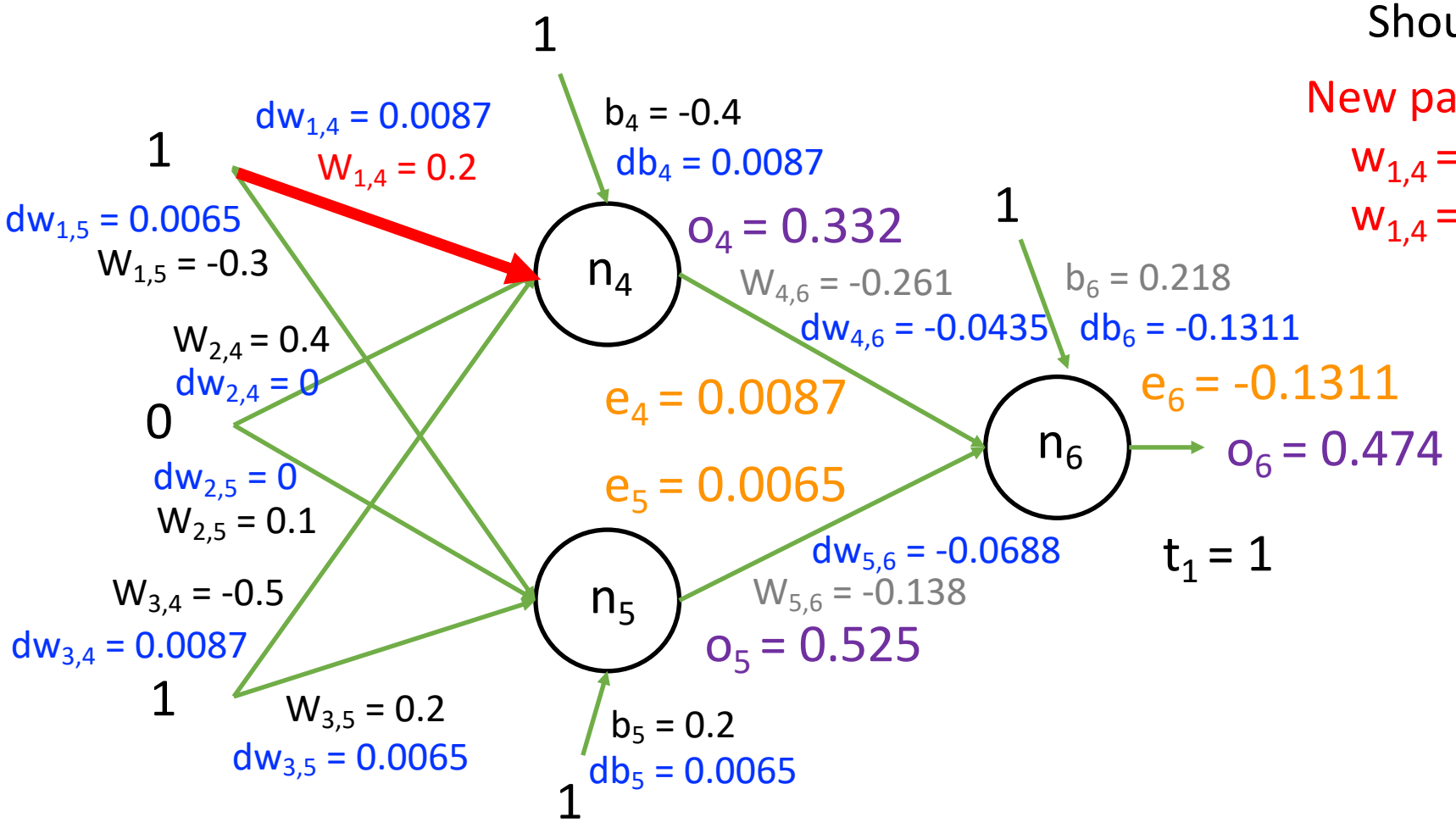
Should b_6 increase or decrease?

New parameters (learning rate = 0.9):

$b_6 = 0.1 - 0.9 * -0.1311$

$b_6 = 0.218$

Example: Step 4 – Update Weights



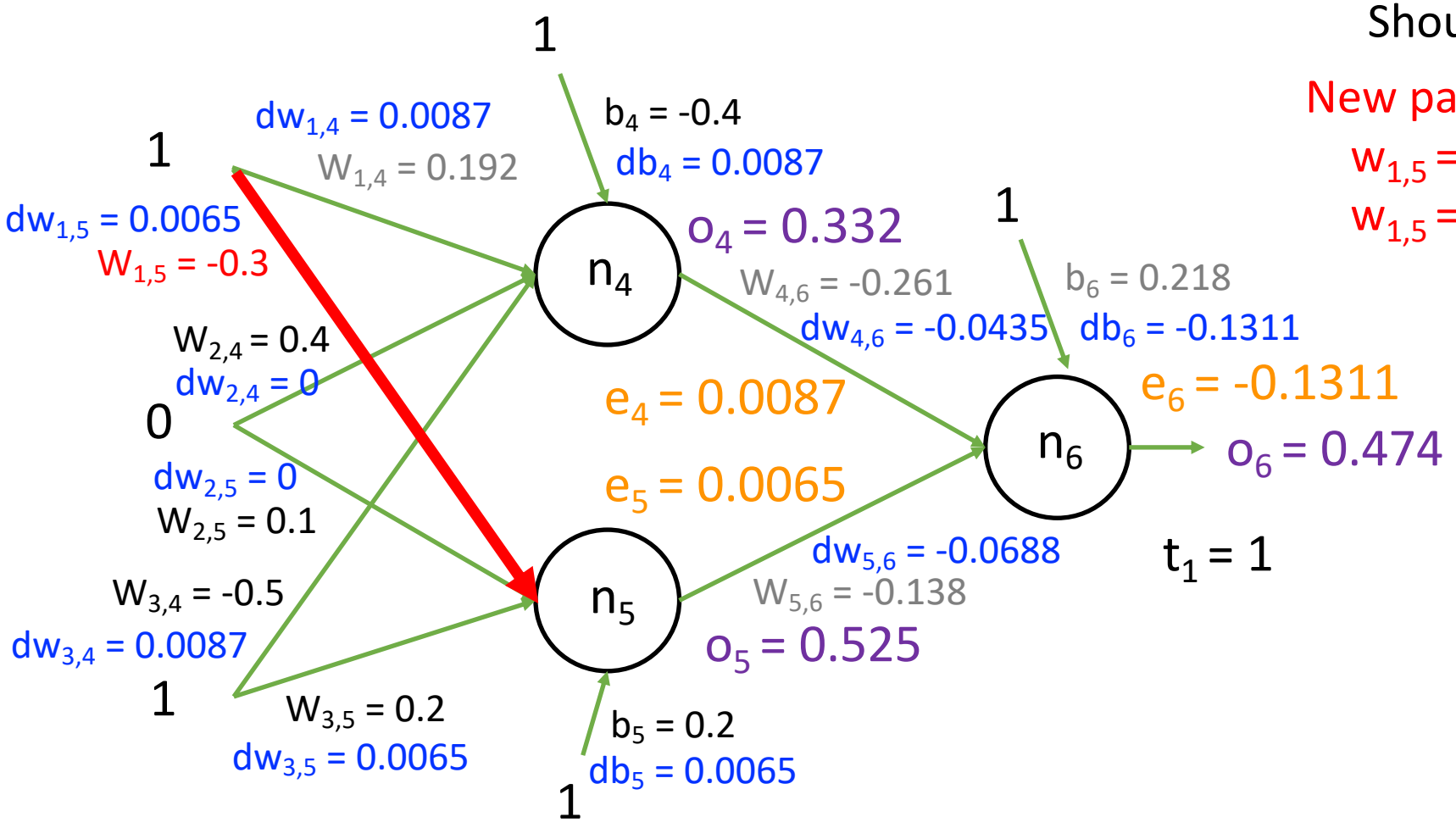
Should $w_{1,4}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{1,4} = 0.2 - 0.9 * 0.0087$

$w_{1,4} = 0.192$

Example: Step 4 – Update Weights



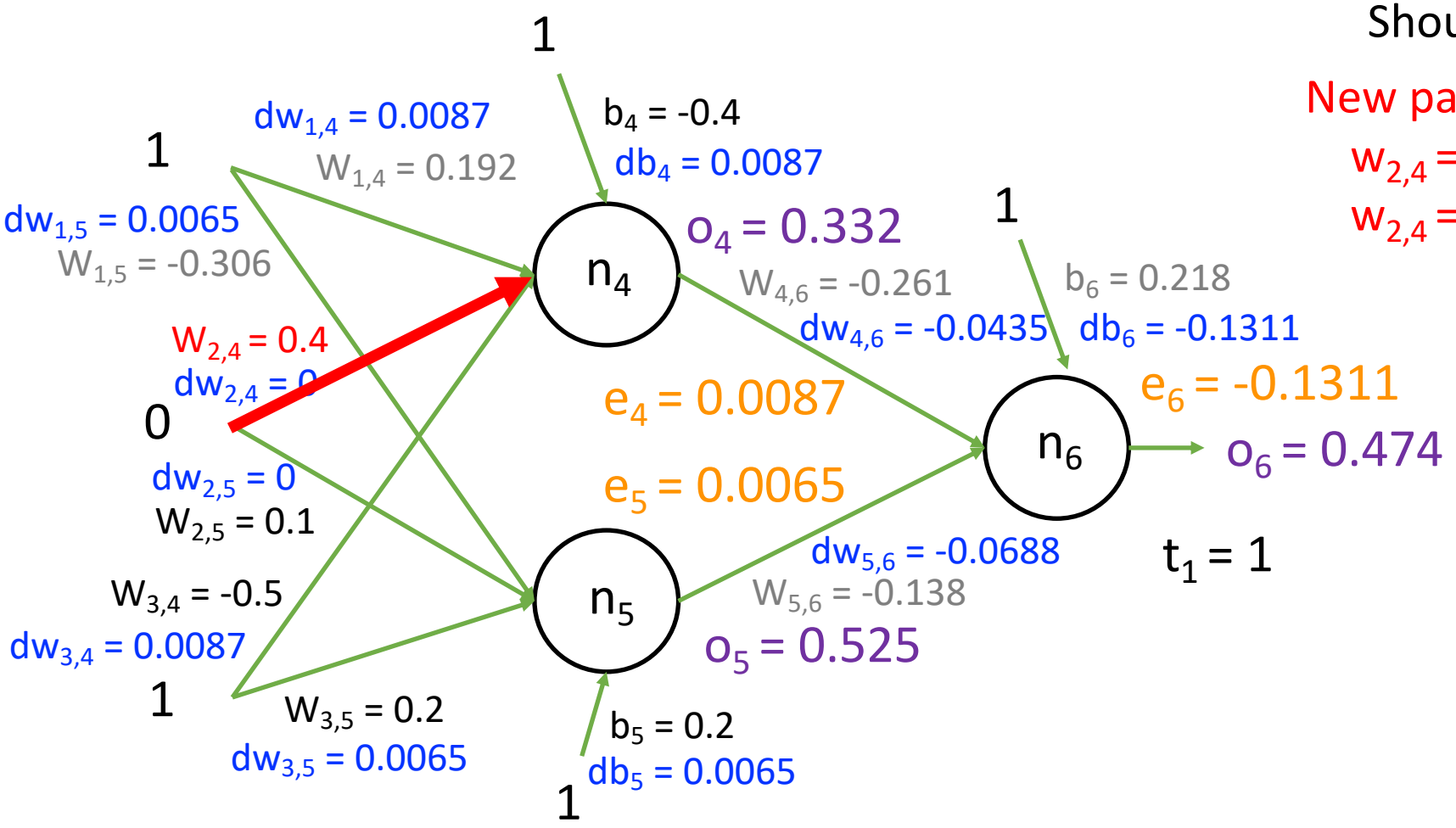
Should $w_{1,5}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{1,5} = -0.3 - 0.9 * 0.0065$

$w_{1,5} = -0.306$

Example: Step 4 – Update Weights



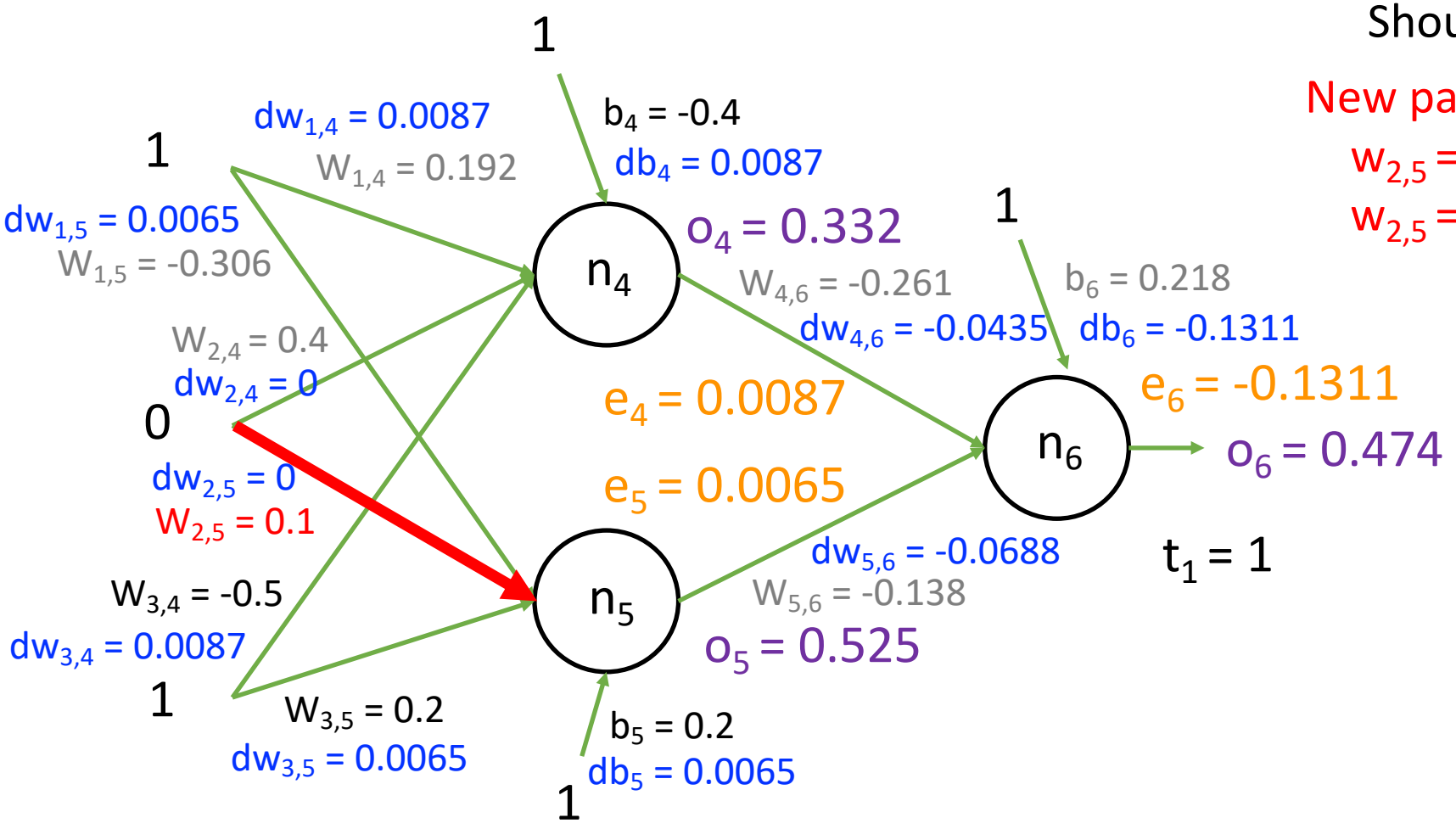
Should $w_{2,4}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{2,4} = 0.4 - 0.9 * 0$

$w_{2,4} = 0.4$

Example: Step 4 – Update Weights



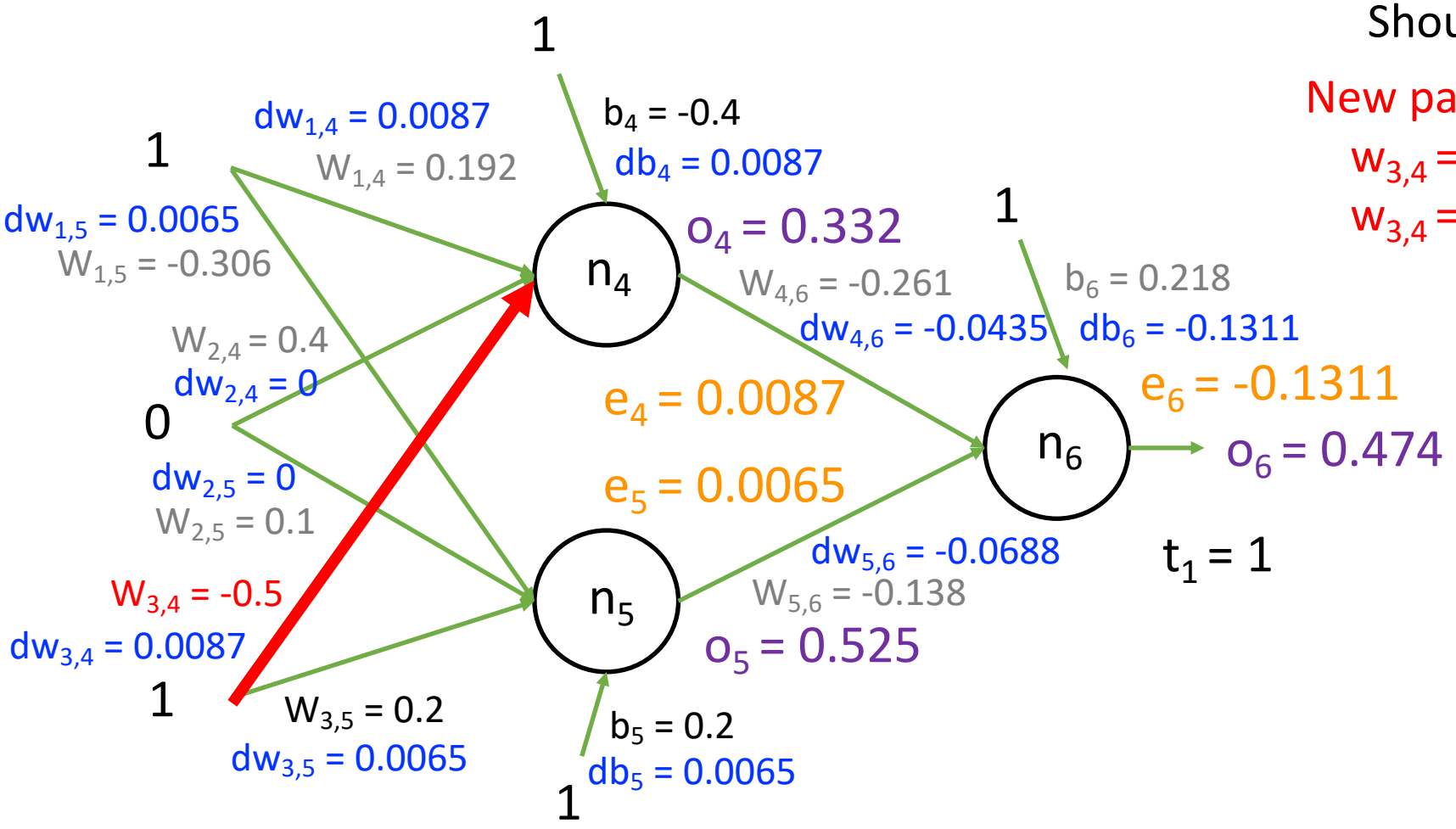
Should $w_{2,5}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{2,5} = 0.1 - 0.9 * 0$

$w_{2,5} = 0.1$

Example: Step 4 – Update Weights



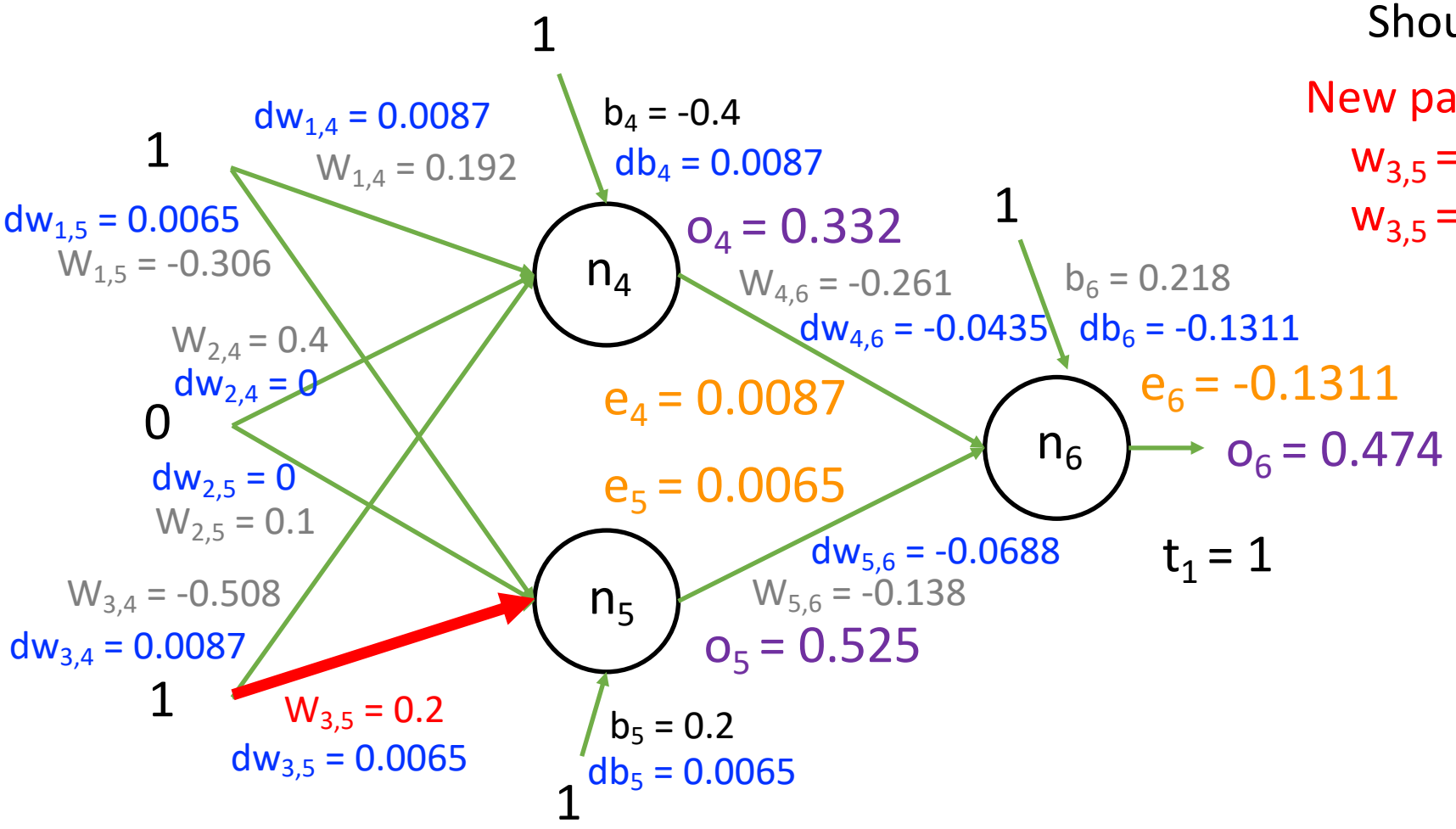
Should $w_{3,4}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{3,4} = -0.5 - 0.9 * 0.0087$

$w_{3,4} = -0.508$

Example: Step 4 – Update Weights



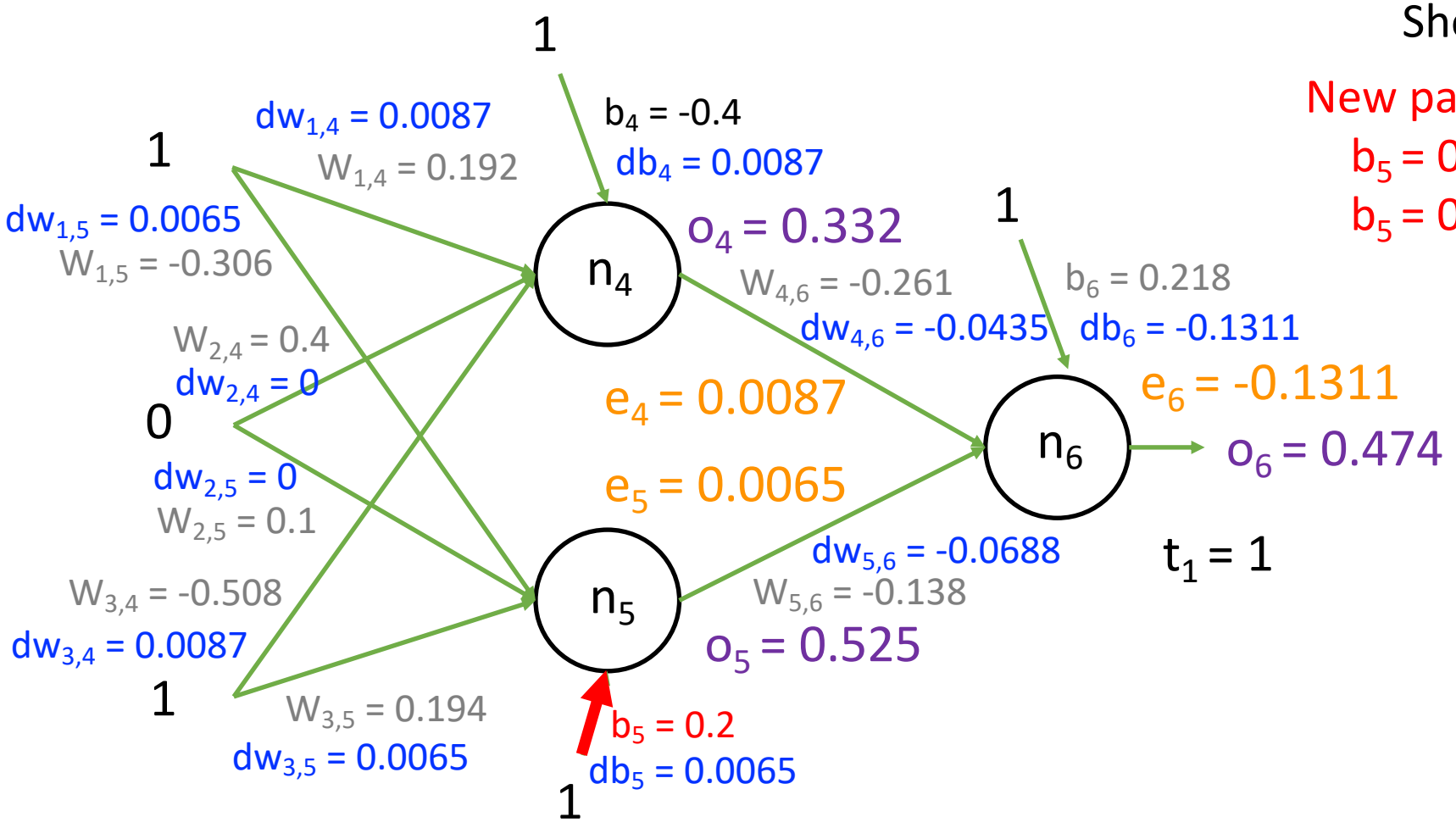
Should $w_{3,5}$ increase or decrease?

New parameters (learning rate = 0.9):

$w_{3,5} = 0.2 - 0.9 * 0.0065$

$w_{3,5} = 0.194$

Example: Step 4 – Update Weights



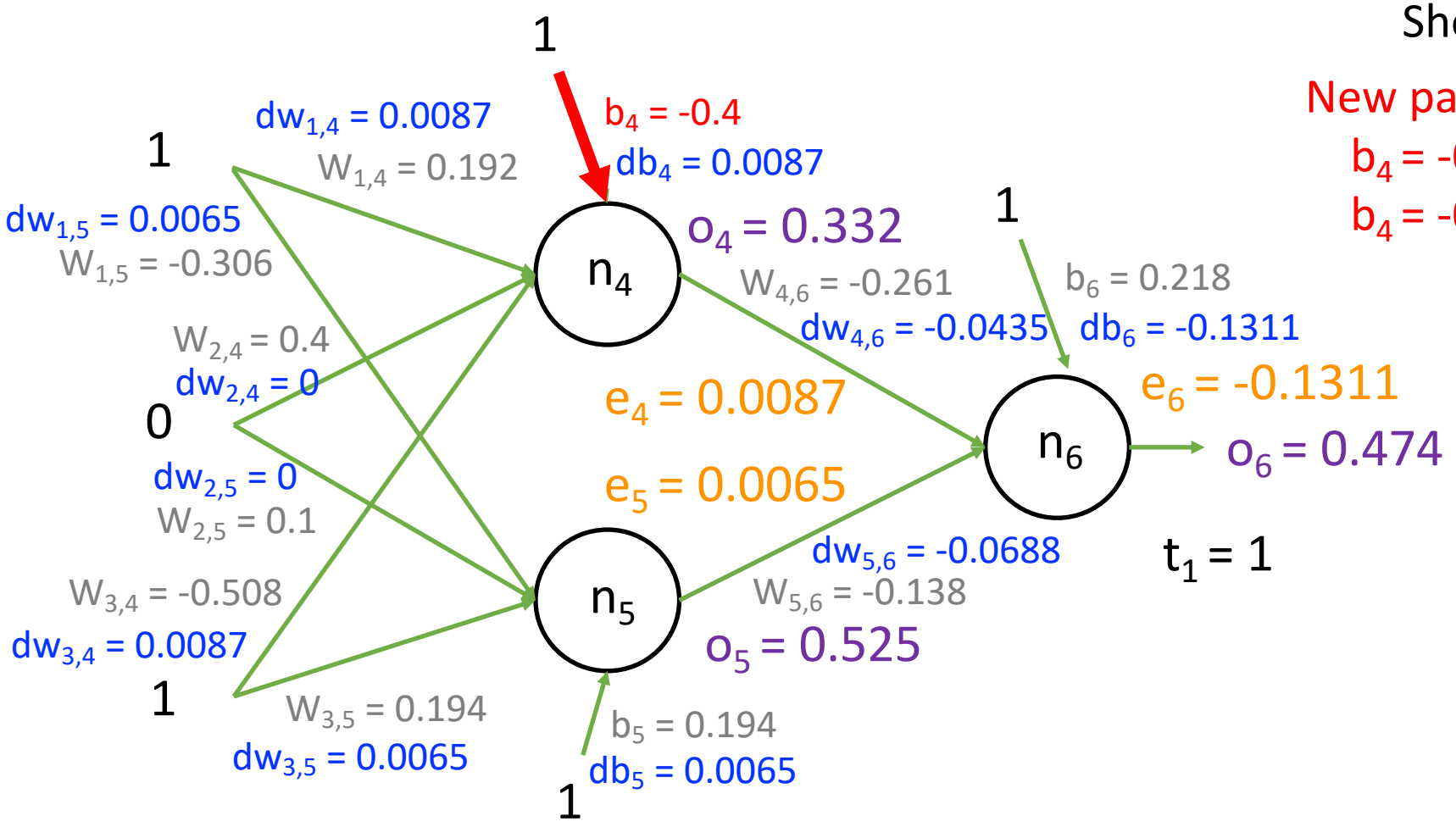
Should b_5 increase or decrease?

New parameters (learning rate = 0.9):

$b_5 = 0.2 - 0.9 * 0.0065$

$b_5 = 0.194$

Example: Step 4 – Update Weights



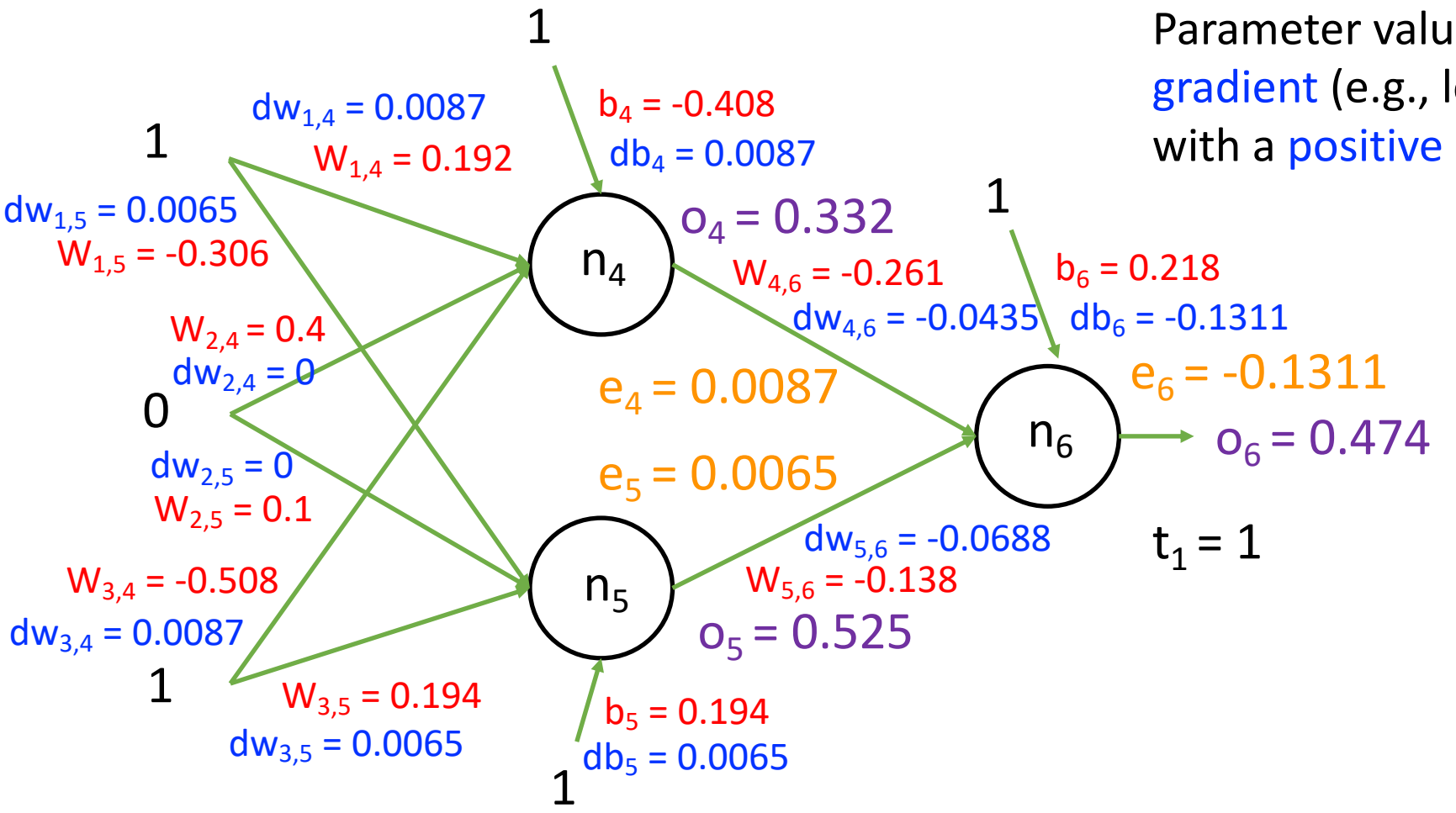
Should b_4 increase or decrease?

New parameters (learning rate = 0.9):

$b_4 = -0.4 - 0.9 * 0.0087$

$b_4 = -0.408$

Example: Repeat Steps 1-4 With New Samples



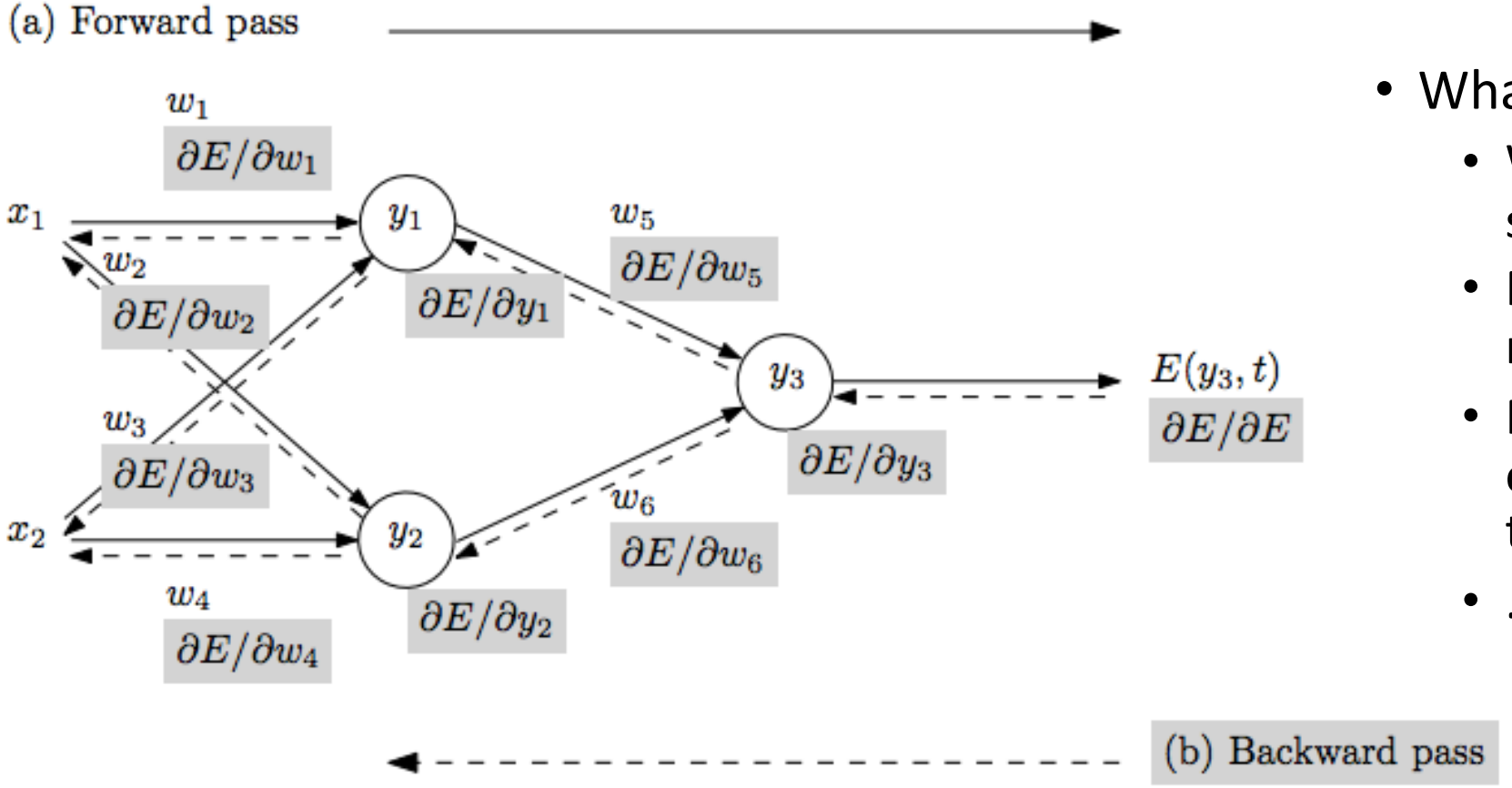
Parameter values **increase** with a negative gradient (e.g., less negative) and **decrease** with a positive gradient (e.g., more negative)

Example: Completed Training Example For…

- Binary classification: predict if a student will get a B or better (minimum required for CS graduate student in this course)

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

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

Training: How Neural Networks Learn

Calculating gradients depends on:

1) Objective/loss 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

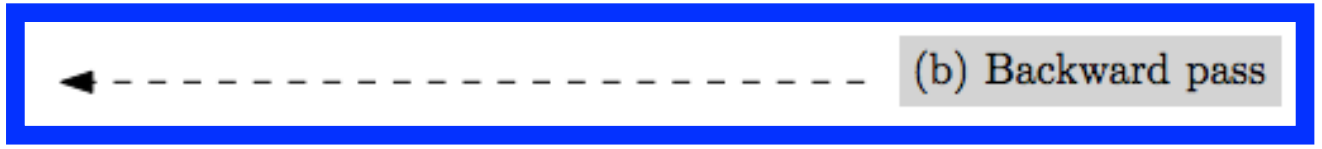


Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Training: How Neural Networks Learn

Can train with a batch of examples instead by using the mean gradient

- 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

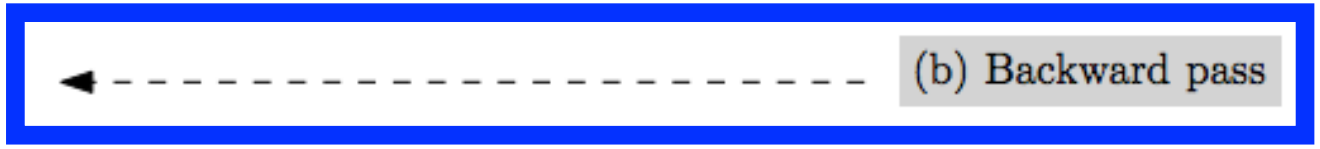


Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Training: How Neural Networks Learn

For excellent step-by-step coding tutorial, watch this video:

https://www.youtube.com/watch?v=VMj-3S1tku0

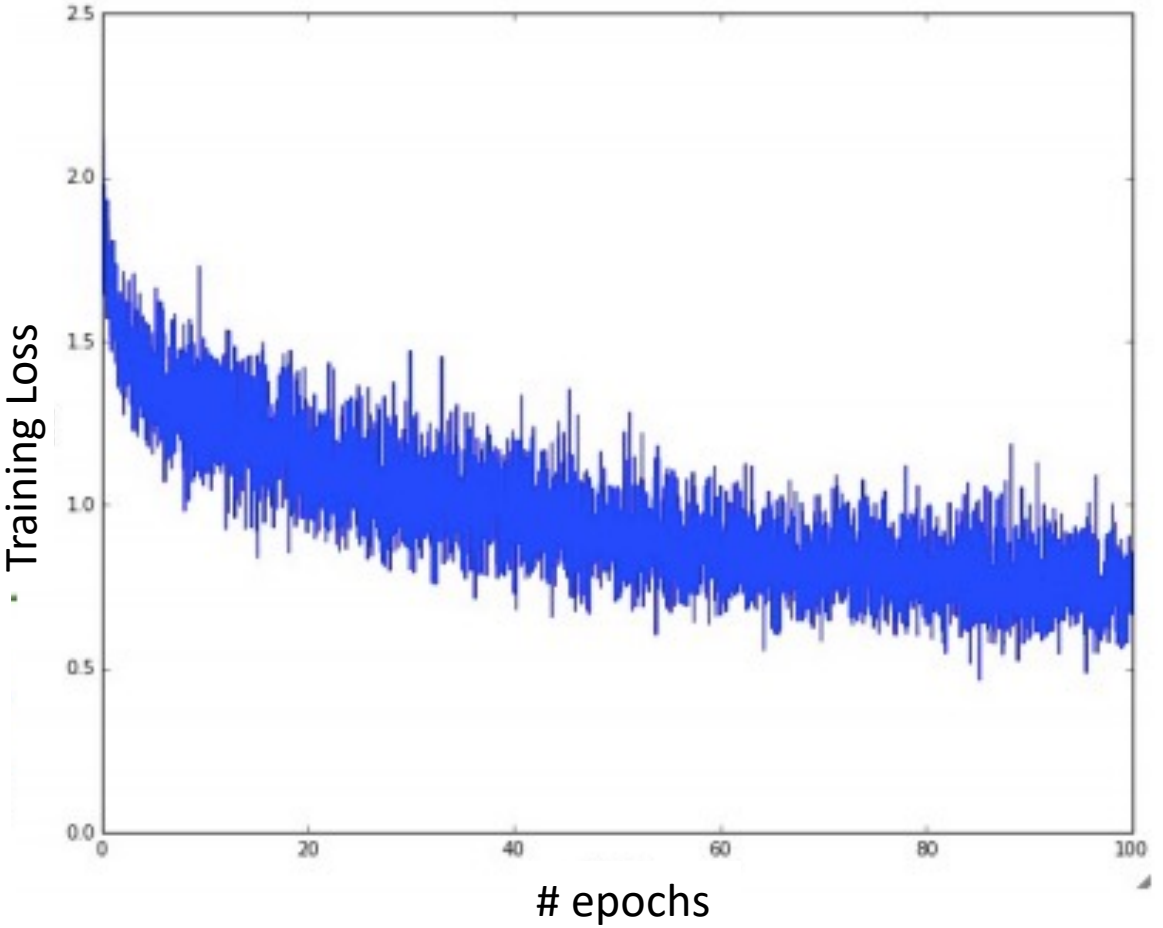
Ian Goodfellow, Yoshua Bengio, and Aaron Courville; Deep Learning, 2016.

Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Lab assignment expectations

Goal: You Will Learn How To…

- Use modern deep learning libraries
- Debug/understand results from your deep learning approaches (e.g., model initialization, architecture, and training)
- Effectively communicate your analysis of deep learning decisions



https://cs231n.github.io/neural-networks-3/#update

How We Grade

- Please review “Lab Assignment Guidelines” page in Canvas

Today's Topics

- Objective function: what to learn
- Gradient descent: how to learn
- Training a neural network: optimization
- Lab assignment expectations

The End