CSCE 638 Natural Language Processing Foundation and Techniques

Lecture 2: Text Classification

Kuan-Hao Huang

Spring 2025

(Some slides adapted from Dan Jurafsky and Karthik Narasimhan)

Course Staff

Instructor

Kuan-Hao Huang

- Email: khhuang@tamu.edu
- Office Hour: Wed. 2pm – 3pm
- Office: PETR 219

TA

Rahul Baid

- Email: rahulbaid@tamu.edu
- Office Hour: Wed. 12pm – 1pm
- Office: TBD

For questions, send emails to csce638-ta-25s@lists.tamu.edu

Textbook and Readings (Optional)

- Speech and Language Processing (3rd ed. draft)
  - Dan Jurafsky and James H. Martin
  - https://web.stanford.edu/~jurafsky/slp3/

Week	Date		Topic	Readings	Note
W1	1/13	L1	Course Overview [slides]		
	1/15	L2	Text Classification [slides]	Logistic Regression Neural Networks	

Lecture Plan

- Formulation of Text Classification
- Bag-of-Words (Bow) and N-Grams
- Logistic Regression
- Neural Networks

Sentiment Analysis

Amazon Basics Replacement Water Filters for Pitchers, Compatible with Brita Water Pitchers & Drinking Water Filter Systems, BPA-Free, WQA & NSF Certified, Made in Europe, 6 Month Filter Supply, 3-Pack

Visit the Amazon Basics Store

4.6 34,435

Amazon's Choice

20K+ bought in past month

$12.55 ($4.18 / Count)

prime

FREE Returns

Thank you for being a Prime Member. Pay ~~$12.55~~ $0.00 for this order; **get a $200 Amazon Gift Card** upon approval for the Amazon Business Prime Card. Terms apply. Learn more

Size: **3 Count (Pack of 1)**

SmartBuyGuy

Reviewed in the United States on October 10, 2024

Size: 3 Count (Pack of 1) | **Verified Purchase**

I recently switched to the Amazon Basics Replacement Water Filters for my Brita pitcher, and the difference has been astonishing. Initially, I was apprehensive about using a generic brand, but I can confidently say these filters deliver outstanding performance comparable to the leading brands.

The first thing I noticed was the taste of my water. The multi-stage filtration technology effectively removes contaminants, leaving my water crisp and fresh. I used to taste chlorine in my tap water, but that's now a distant memory. It's a pleasure to drink water again!

Installation was seamless. The filters fit perfectly into my Brita pitcher, and I had no issues setting them up. I appreciate the clear instructions that come with the product, making the process hassle-free. Additionally, each filter lasts up to 40 gallons or about two months, making them a cost-effective choice for my household.

I also love the eco-friendly aspect of these filters. Knowing that one filter replaces 300 single-use plastic bottles gives me a sense of satisfaction. Not only am I saving money, but I'm also contributing to reducing plastic waste—something we all need to consider in today's world.

Positive

Lin D

Reviewed in the United States on November 9, 2024

Size: 1 Count | **Verified Purchase**

The pitcher comes in three parts, the pitcher, the reservoir, the lid.
Take the reservoir out of the pitcher.
Put filter into the hole made for it.
Put water into reservoir.
Watch the water come out the bottom of the filter.
Watch just as much water come out around the outside of the filter

No amount of pushing, rearranging, twisting will get the filter, Brita or other, to seat well enough not to leak.

Explains why I keep getting a green film on the bottom of the pitcher every few months while Brita brand pitcher hasn't done that in the years I've had it.

To be honest, I didn't notice it leaking when new. But I've just cleaned out the third batch of green stuff in six months. Bleached it the second time. Third time, I noticed the severe leakage. Done with it. Doesn't save money to buy a less expensive pitcher I have to replace more often.

Negative

https://www.amazon.com/dp/B07YT16TMS

Topic Classification

A.I. Chatbots Defeated Doctors at Diagnosing Illness

A small study found ChatGPT outdid human physicians when assessing medical case histories, even when those doctors were using a chatbot.

By Gina Kolata

Nov. 17, 2024
Leer en español

Dr. Adam Rodman, an expert in internal medicine at Beth Israel Deaconess Medical Center in Boston, confidently expected that chatbots built to use artificial intelligence would help doctors diagnose illnesses.

He was wrong.

Instead, in a study Dr. Rodman helped design, doctors who were given ChatGPT-4 along with conventional resources did only slightly better than doctors who did not have access to the bot. And, to the researchers' surprise, ChatGPT alone outperformed the doctors.

Technology

Business

Economy

Health

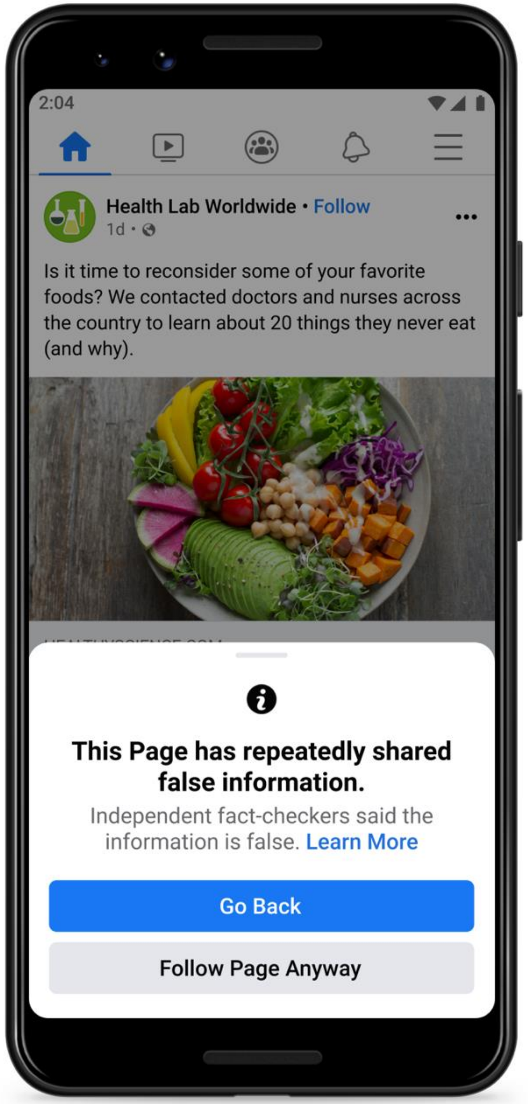
Politics

Education

Sports

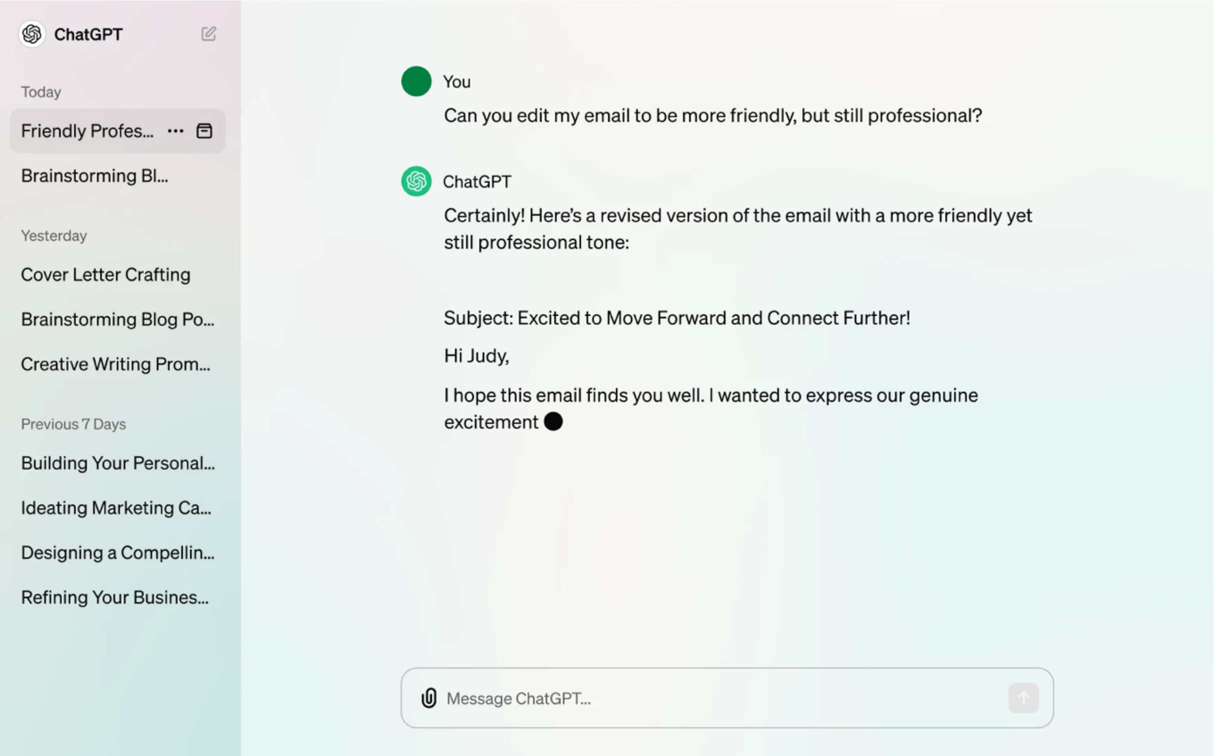
https://www.nytimes.com/2024/11/17/health/chatgpt-ai-doctors-diagnosis.html

Fraud Detection



Suspicious / Normal

Large Language Models with Text Classification



Normal model, math mode, code mode, …

Enable search, enable calculator, …

Ethical issue, harmful prompts, …

Generation is Sequence of Classification!

This morning, I woke ← Selected from all possible words

This morning, I woke up ← Selected from all possible words

This morning, I woke up at ← Selected from all possible words

Text Classification

Text

A small study found that ChatGPT outdid human physicians when assessing medical case histories, even when those doctors were using a chat bot.

It can be phrase, sentence, paragraph, or document

Category (Class)

Technology Economy

Health Politics Sports

$$x = [w_1, w_2, \dots, w_l] \Rightarrow C = \{c_1, c_2, \dots, c_k\}$$

Supervised Learning

Training Stage

- Training data $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
  - Example $x_i \in \mathcal{X}$, label $y_i \in C$
- Train a classifier(model) $f: \mathcal{X} \to C$

How to train?

Testing Stage

- Testing data $\mathcal{D}_{test} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
- Make predictions $\tilde{y}_i = f(x_i)$
- Evaluate performance $\frac{1}{\mathrm{n}} \sum_i S(y_i, \tilde{y}_i)$

Accuracy, F1 Score, etc.

Supervised Learning

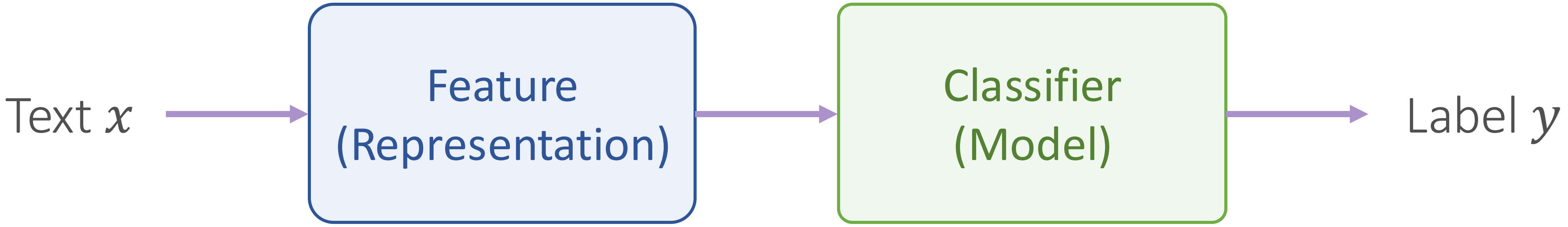
Training Stage

- Training data $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
  - Example $x_i \in \mathcal{X}$, label $y_i \in C$
- Train a classifier(model) $f: \mathcal{X} \rightarrow C$

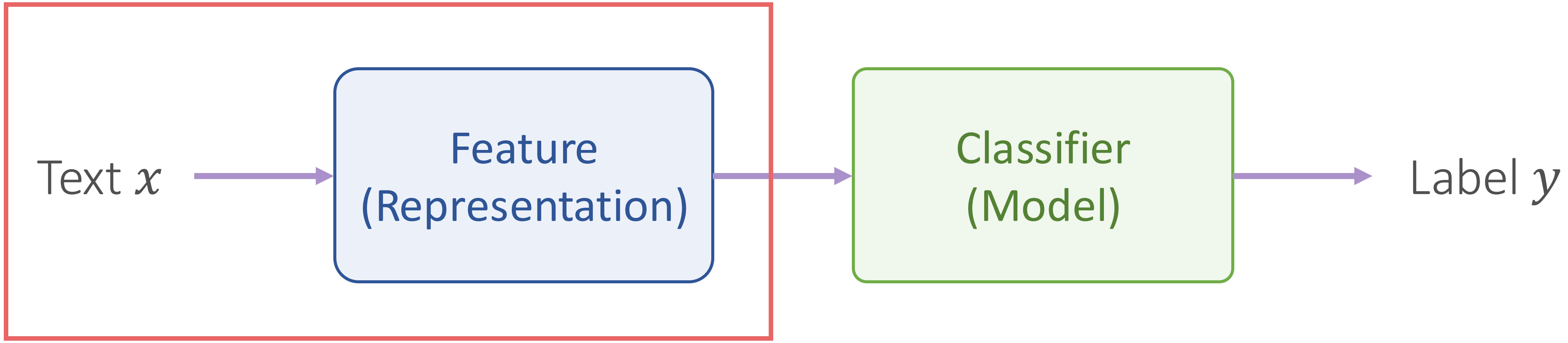
How to train?

- How does the model understand example x?
- How does the model make label prediction y?

A General Framework for Text Classification



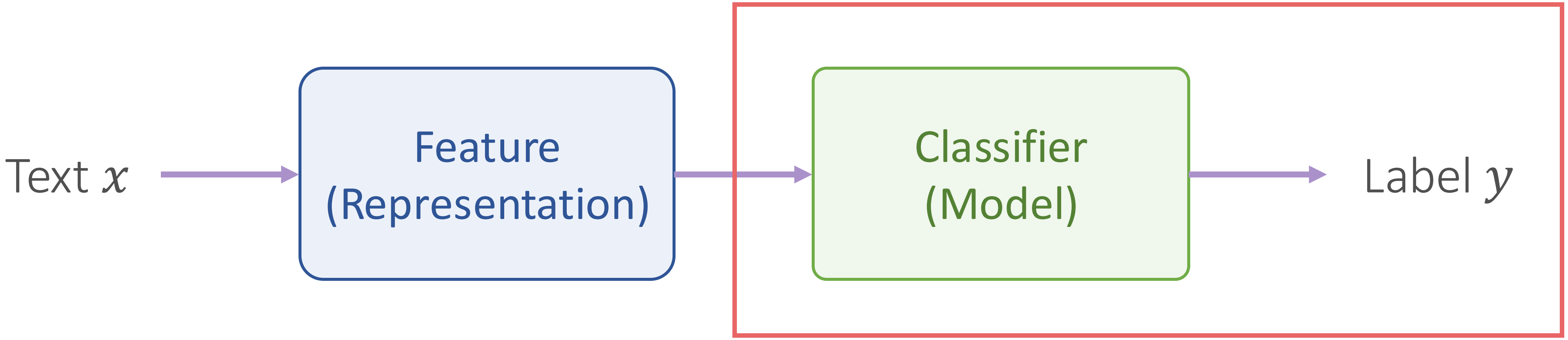
A General Framework for Text Classification



- Teach the model how to understand example x
- Convert the text to a mathematical form
  - The mathematical form captures essential characteristics of the text
- Bag-of-words, n-grams, word embeddings, etc.

We will talk about them later!

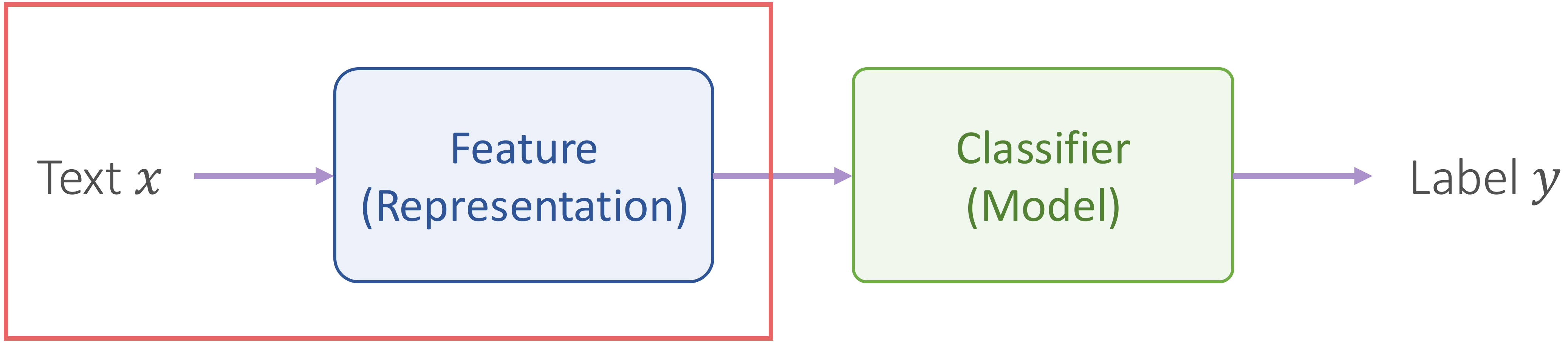
A General Framework for Text Classification



- Teach the model how to make prediction y
- Logistic regression, neural networks, CNN, RNN, LSTM, Transformers

We will talk about them later!

Bag-of-Words (BoW)



- Bag-of-Words (BoW)
  - Consider text as a set of words
- Easy, no effort required

Bag-of-Words (BoW)

This restaurant is the best one in College Station

I study natural language processing everyday

Bag-of-Words (BoW)

This restaurant is the best one in College Station

$$\mathbf{x} = [0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad \ldots \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1]$$

Feature vector $\mathbf{x}$ is a binary vector

Each dimension represents one word, indicating the presence of word

The length of vector is the dictionary size $|V|$

Advantages and disadvantages?

Bag-of-Words (BoW)

Bob likes Alice very much

Alice likes Bob very much

They will have the same BoW vector!

$$\mathbf{x} = [0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad \ldots \quad 0 \quad 1]$$

$\boxed{\text{BoW fails to capture sentential structure}}$

Any solutions?

N-Grams

Bob likes Alice very much

Unigram	*{Bob, likes, Alice, very, much}*
Bigram	*{Bob likes, likes Alice, Alice very, very much}*
Trigram	*{Bob likes Alice, likes Alice very, Alice very much}*
4-gram	*{Bob likes Alice very, likes Alice very much}*

Bag-of-N-Grams

Bob likes Alice very much

Alice likes Bob very much

$$\mathbf{x} = [0 \;\; 1 \;\; \ldots \;\; 0 \;\; 1 \;\; 1 \;\; 0 \;\; \ldots \;\; 0 \;\; 1]$$

$$\mathbf{x} = [0 \;\; 1 \;\; \ldots \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; \ldots \;\; 1 \;\; 1]$$

BoW (unigram) features

Bigram features

We can consider trigrams, 4-grams, …

N-gram features capture more sentential structure

Other Variants

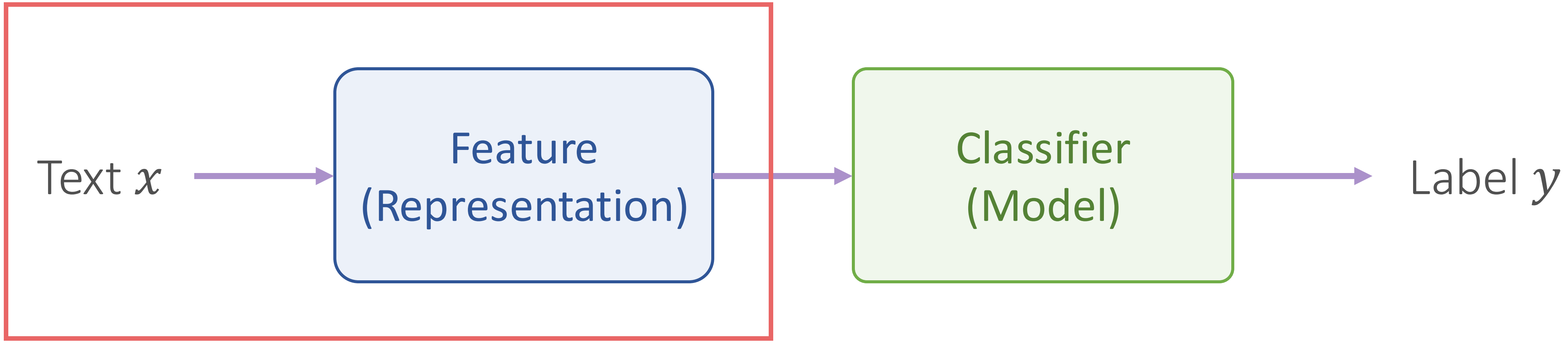
Binary BoW	$\mathbf{x} = [0 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; \dots \;\; 0 \;\; 1]$
Word Count	$\mathbf{x} = [0 \;\; 2 \;\; 1 \;\; 0 \;\; 0 \;\; 4 \;\; \dots \;\; 0 \;\; 3]$
Word Frequency	$\mathbf{x} = [0 \;\; 0.16 \;\; 0.08 \;\; 0 \;\; 0 \;\; 0.32 \;\; \dots \;\; 0 \;\; 0.24]$
TF-IDF	$\mathbf{x} = [0 \;\; 0.48 \;\; 0.02 \;\; 0 \;\; 0 \;\; 0.15 \;\; \dots \;\; 0 \;\; 0.88]$

$$f_w \cdot \log \frac{N}{n_t}$$

Term Frequency (TF)

Inverse Document Frequency (IDF)

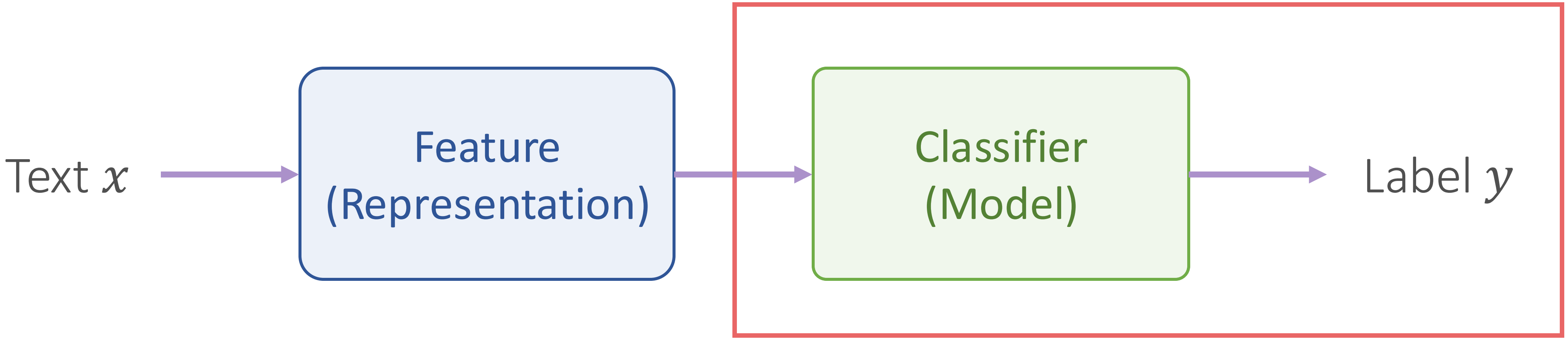
Bag-of-Words and Bag-of-N-Grams



- Bag-of-Words (BoW)
  - A set of words
- Bag-of-N-Grams
  - A set of n-grams

We will discuss "learnable" features later!

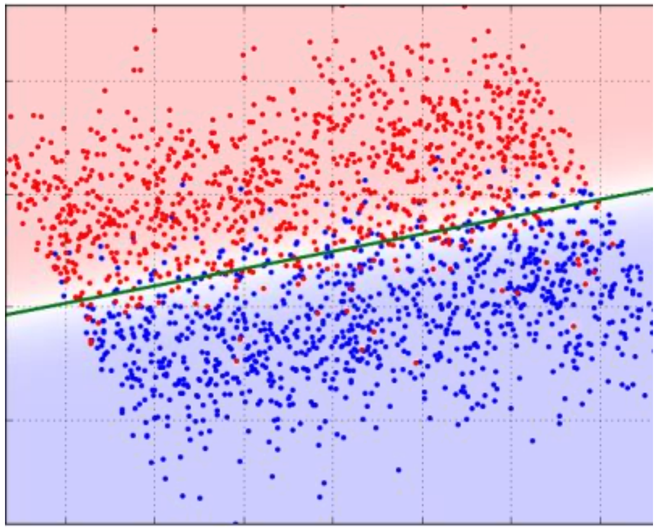
Logistic Regression

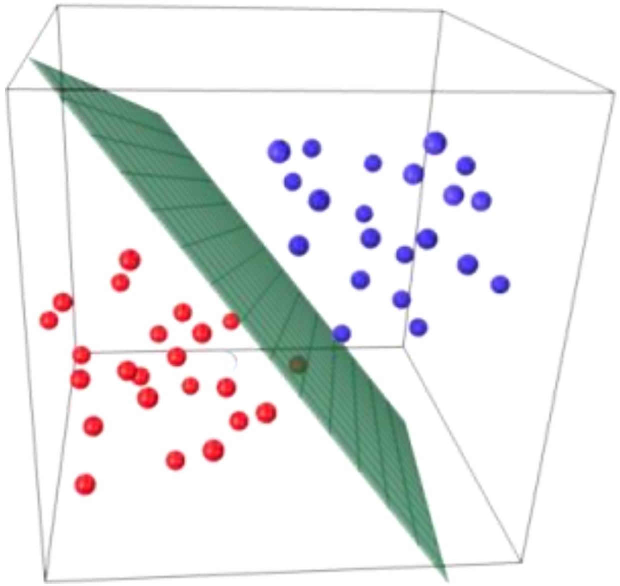


- Logistic regression
  - Find linear weights to map feature vector $\mathbf{x}$ to label y

Logistic Regression

- Let's start from binary classification
  - Input: feature vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$
  - Output: label $y \in \{0, 1\}$
- Find a linear decision boundary to classify $\mathbf{x}$ into $\{0, 1\}$





https://blog.bigml.com/2016/09/28/logistic-regression-versus-decision-trees/
https://codesachin.wordpress.com/2015/08/16/logistic-regression-for-dummies/

Logistic Regression

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$ Label $y = 0 \; or \; 1$

$\boxed{\text{Weight Vector } \mathbf{w} = [w_1, w_2, w_3, \dots, w_d] \qquad \text{Bias } b}$

Learnable parameters

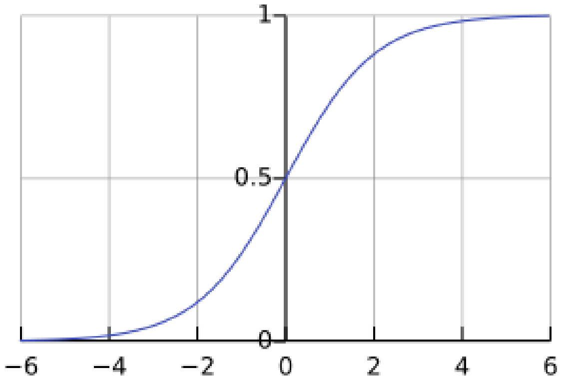
$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$\tilde{y} = P(y = 1 | \, \mathbf{x}) = \sigma(z)$$

Convert to probability

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

Sigmoid Function



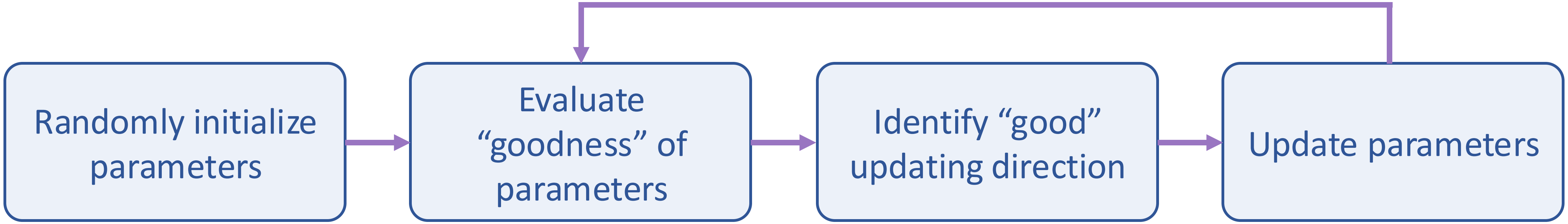
Decision boundary: $= \begin{cases} 1 & \text{If } \tilde{y} \geq 0.5 \\ 0 & \text{If } \tilde{y} < 0.5 \end{cases}$

How to Find The Best Parameters?

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_d]$ Bias b

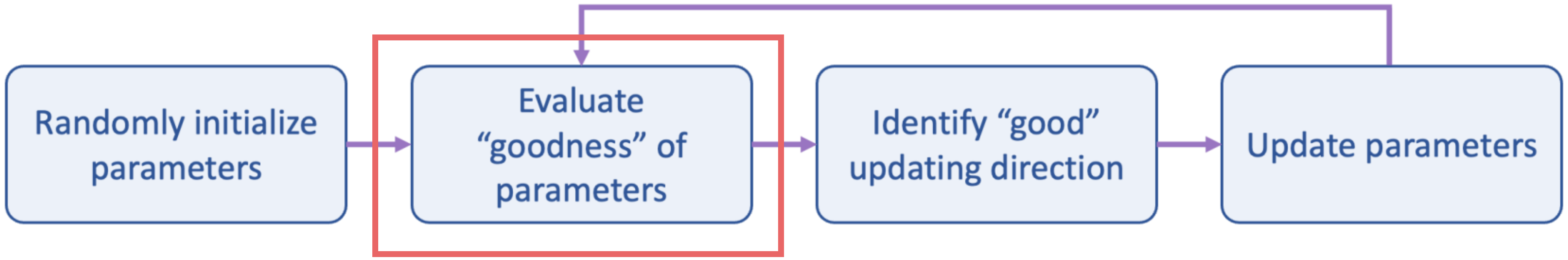
Learnable parameters

Iterative Optimization Methods



Loss Function

Iterative Optimization Methods



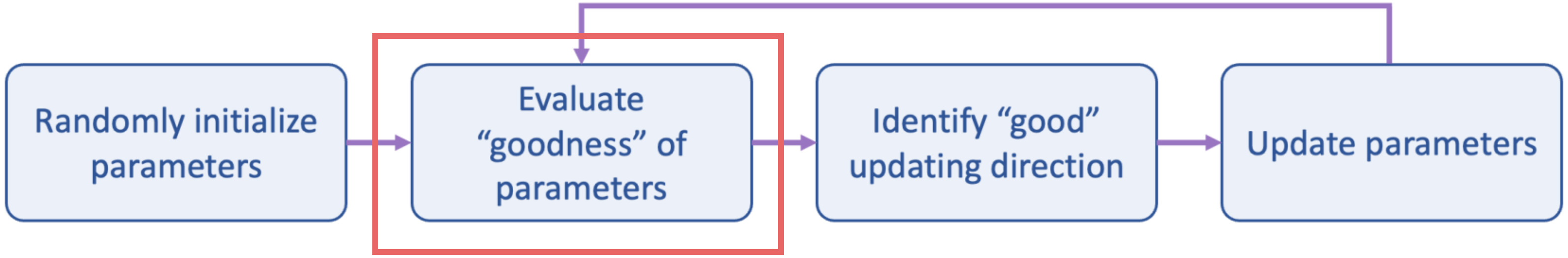
- For each training example (x, y)
- Output label probability is $\tilde{y} = P(y = 1 | \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

Loss Function

Iterative Optimization Methods



Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -[y \log \tilde{y} + (1-y)\log(1-\tilde{y})]$$

$y = 1$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{CE} = -[1 \cdot \log 0.9 + 0 \cdot \log(0.1)] = -\log 0.9 \approx 0.105$

$y = 1$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{CE} = -[1 \cdot \log 0.1 + 0 \cdot \log(0.9)] = -\log 0.1 \approx 2.302$

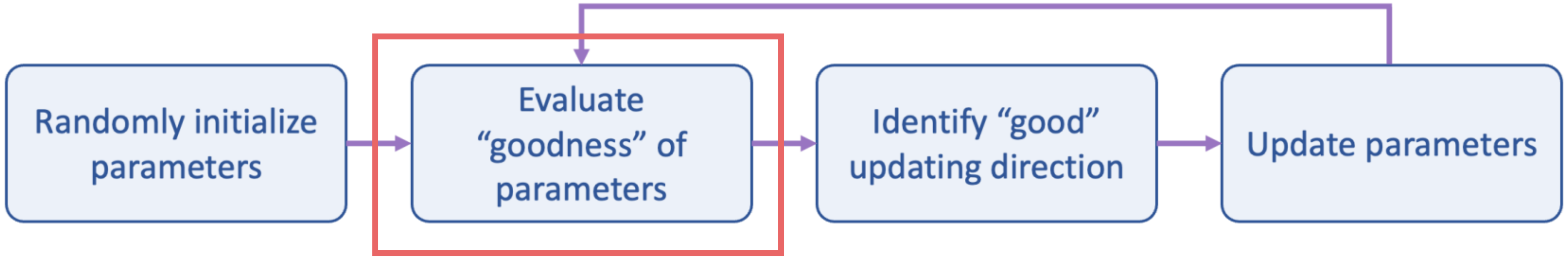
$y = 0$ and $\tilde{y} = 0.9$ $\quad \mathcal{L}_{CE} = -[0 \cdot \log 0.9 + 1 \cdot \log(0.1)] = -\log 0.1 \approx 2.302$

$y = 0$ and $\tilde{y} = 0.1$ $\quad \mathcal{L}_{CE} = -[0 \cdot \log 0.1 + 1 \cdot \log(0.9)] = -\log 0.9 \approx 0.105$

The lower the loss is, the more accurate the output probability is

Loss Function

Iterative Optimization Methods



- Training data $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- Output labels probabilities $\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_m$

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i \mathcal{L}_{CE}(y_i, \tilde{y}_i) = -\frac{1}{m}\sum_i [y_i \log \tilde{y}_i + (1 - y_i)\log(1 - \tilde{y}_i)]$$

Optimization Objective

Iterative Optimization Methods

Randomly initialize parameters → Evaluate "goodness" of parameters → Identify "good" updating direction → Update parameters

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_{i} \mathcal{L}_{CE}(y_i, \tilde{y}_i)$$

Parameters θ = $\boxed{\text{Weight Vector } \mathbf{w} = [w_1, w_2, w_3, \dots, w_d] \qquad \text{Bias } b}$

$$[\mathbf{w}^*; b^*] = \theta^* = \arg\min_{\theta} \mathcal{L}_{total}$$

Optimization objective

Gradient

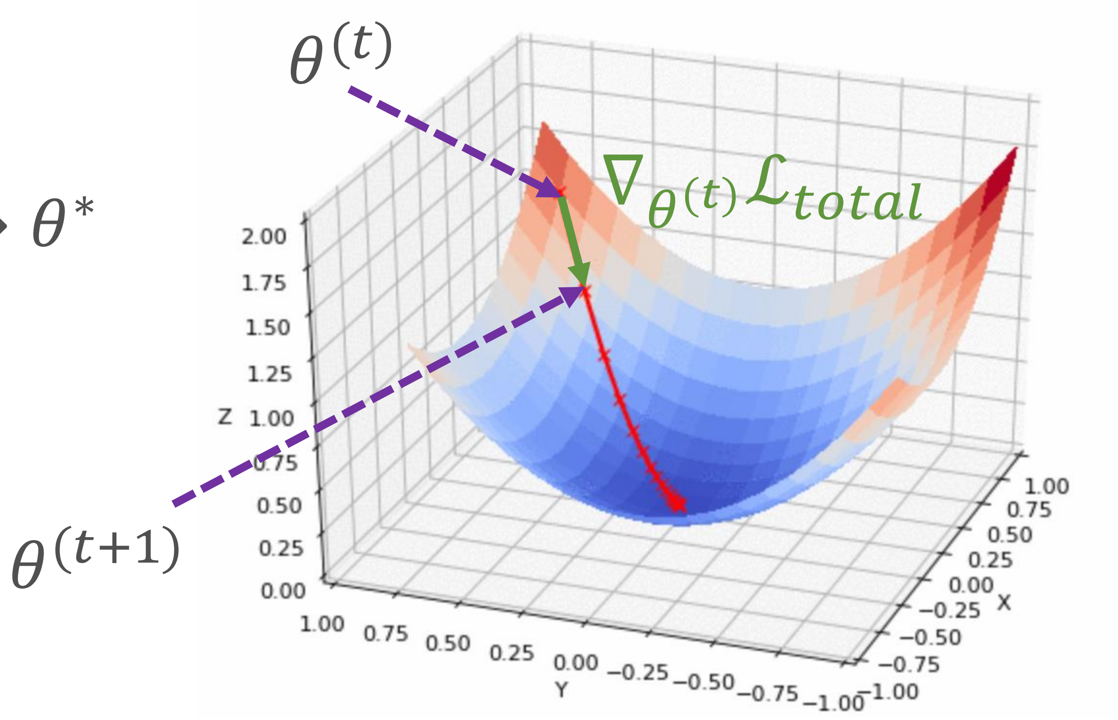
Iterative Optimization Methods

Randomly initialize parameters → Evaluate "goodness" of parameters → Identify "good" updating direction → Update parameters

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{total}$$

$$\theta^{(0)} \rightarrow \theta^{(1)} \rightarrow \theta^{(2)} \rightarrow \cdots \rightarrow \theta^{(k)} \rightarrow \cdots \rightarrow \theta^*$$

$\nabla_{\theta^{(t)}} \mathcal{L}_{total}$ is a "good" direction to minimize the objective



https://www.digitalocean.com/community/tutorials/intro-to-optimization-in-deep-learning-gradient-descent

Gradient

$$\nabla_\theta \mathcal{L}_{total} \qquad \boxed{\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}} \qquad \frac{\partial \mathcal{L}_{total}}{\partial b}}$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = \frac{\partial \left(-\frac{1}{m}\sum_i [y_i \log \tilde{y}_i + (1-y_i)\log(1-\tilde{y}_i)]\right)}{\partial \mathbf{w}_j}$$

$$= \frac{\partial \left(-\frac{1}{m}\sum_i [y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))]\right)}{\partial \mathbf{w}_j}$$

$$\boxed{\begin{array}{c} \tilde{y}_i = \sigma(z_i) \\ z_i = \mathbf{w} \cdot \mathbf{x}_i + b \end{array}}$$

$$= -\frac{1}{m}\sum_i \left[y_i \frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} + (1-y_i)\frac{\partial \log(1-\sigma(z_i))}{\partial \mathbf{w}_j} \right]$$

Gradient

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = -\frac{1}{m}\sum_i \left[y_i \frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} + (1 - y_i) \frac{\partial \log(1 - \sigma(z_i))}{\partial \mathbf{w}_j} \right]$$

$$\frac{\partial \log \sigma(z_i)}{\partial \mathbf{w}_j} = \frac{1}{\sigma(z_i)} \cdot [\sigma(z_i)(1 - \sigma(z_i))] \cdot \mathbf{x}_{i,j} = (1 - \sigma(z_i))\mathbf{x}_{i,j}$$

$$\boxed{\sigma'(z) = \sigma(z)(1 - \sigma(z))}$$

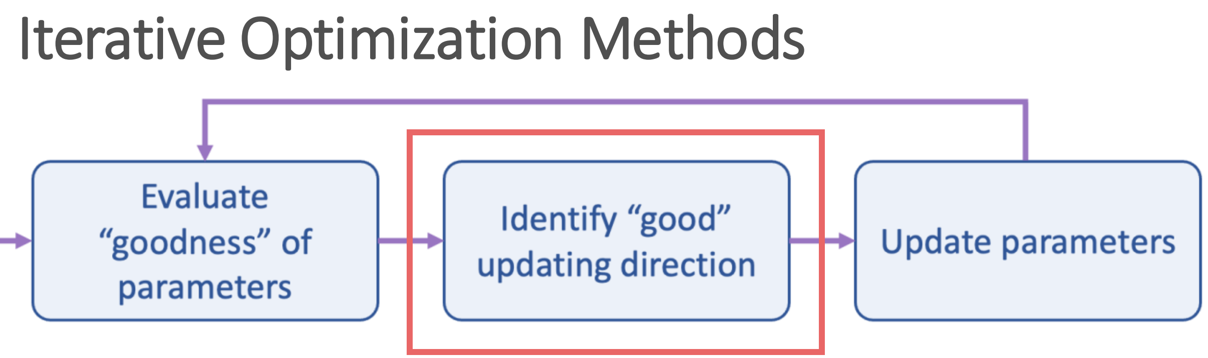
$$\frac{\partial \log(1 - \sigma(z_i))}{\partial \mathbf{w}_j} = \frac{1}{1 - \sigma(z_i)} \cdot [-\sigma(z_i)(1 - \sigma(z_i))] \cdot \mathbf{x}_{i,j} = -\sigma(z_i)\mathbf{x}_{i,j}$$

$$\boxed{(1 - \sigma(z))' = -\sigma(z)(1 - \sigma(z))}$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}_j} = -\frac{1}{m}\sum_i [y_i(1 - \sigma(z_i))\mathbf{x}_{i,j} + (1 - y_i)(-\sigma(z_i)\mathbf{x}_{i,j})]$$

$$= -\frac{1}{m}\sum_i (y_i - \sigma(z_i))\mathbf{x}_{i,j} = \frac{1}{m}\sum_i (\tilde{y}_i - y_i)\mathbf{x}_{i,j}$$

Gradient

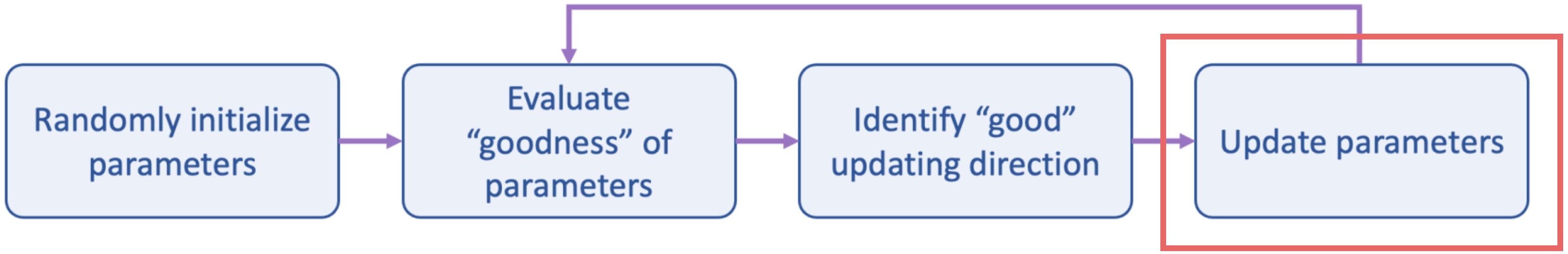


$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)\mathbf{x}_i$$

$$\frac{\partial \mathcal{L}_{total}}{\partial b} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)$$

Gradient Descent

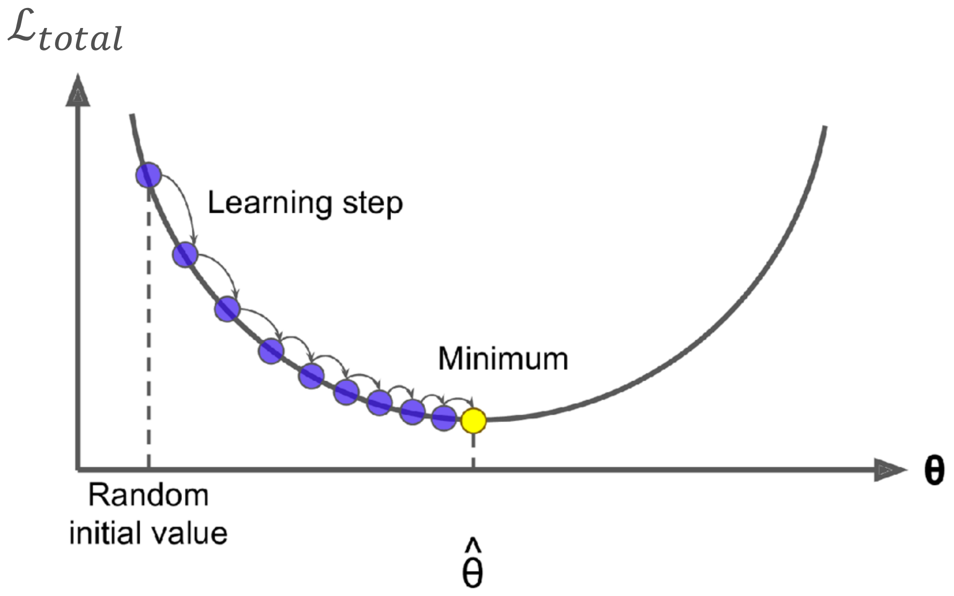
Iterative Optimization Methods



$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \, \nabla_{\mathbf{w}} \mathcal{L}_{total}$$

$$b^{(t+1)} = b^{(t)} - \eta \, \nabla_{b} \mathcal{L}_{total}$$

Learning step



https://insightfultscript.com/collections/programming/machine-learning/sgd/

Training Process

Iterative Optimization Methods

Randomly initialize parameters → Evaluate "goodness" of parameters → Identify "good" updating direction → Update parameters

Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_{i} \mathcal{L}_{CE}\left(y_i, \tilde{y}_i; \mathbf{w}^{(t)}, b^{(t)}\right)$$

$$\frac{\partial \mathcal{L}_{total}}{\partial \mathbf{w}^{(t)}} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)\mathbf{x}_i$$

$$\frac{\partial \mathcal{L}_{total}}{\partial b^{(t)}} = \sum_{i=1}^{m} (\tilde{y}_i - y_i)$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} \mathcal{L}_{total}$$

$$b^{(t+1)} = b^{(t)} - \eta \nabla_{b} \mathcal{L}_{total}$$

From Binary to Multiclass Classification

- Logistic Regression for binary classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_d]$ Label $y = 0 \; or \; 1$

Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \ldots, w_d]$ Bias b

Learnable Parameters

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

$$P(y = 1 | \mathbf{x}) = \sigma(z)$$

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

Sigmoid Function

$$\text{Prediction} = \begin{cases} 1 & \text{If } P(y = 1 | \mathbf{x}) \geq 0.5 \\ 0 & \text{If } P(y = 1 | \mathbf{x}) < 0.5 \end{cases}$$

From Binary to Multiclass Classification

- Logistic Regression for multiclass classification

Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$ Label $y = 0, 1, \dots, C-1$

$\boxed{\text{Weight Vectors } \mathbf{w}_c = [w_{c,1}, w_{c,2}, w_{c,3}, \dots, w_{c,d}] \quad \text{Bias } b_c}$ Learnable Parameters

$$z_c = \mathbf{w}_c \cdot \mathbf{x} + b_c$$

$$P(y = c | \mathbf{x}) = \mathrm{softmax}(z_c)$$

$$\mathrm{softmax}(z_c) = \frac{e^{z_c}}{\sum_t e^{z_t}}$$

Softmax Function

$$\text{Prediction} = \arg\max_c P(y = c | \mathbf{x})$$

From Binary to Multiclass Classification

Binary Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -[y \log \tilde{y} + (1 - y) \log(1 - \tilde{y})]$$

Multiclass Cross Entropy Loss

$$\mathcal{L}_{CE}(y, \tilde{y}) = -\sum_{c=0}^{C} y_c \log P(y = c | \mathbf{x})$$

Softmax

Label

$$z_0 = \mathbf{w}_0 \cdot \mathbf{x} + b_0 = -1.2 \qquad \frac{e^{-1.2}}{e^{-1.2} + e^{4.8} + e^{-0.7} + e^{2.5}} \approx 0.002 \qquad 0$$

$$z_1 = \mathbf{w}_1 \cdot \mathbf{x} + b_1 = 4.8 \qquad \frac{e^{4.8}}{e^{-1.2} + e^{4.8} + e^{-0.7} + e^{2.5}} \approx 0.903 \qquad 1$$

$$z_2 = \mathbf{w}_2 \cdot \mathbf{x} + b_2 = -0.7 \qquad \frac{e^{-0.7}}{e^{-1.2} + e^{4.8} + e^{-0.7} + e^{2.5}} \approx 0.004 \qquad 0$$

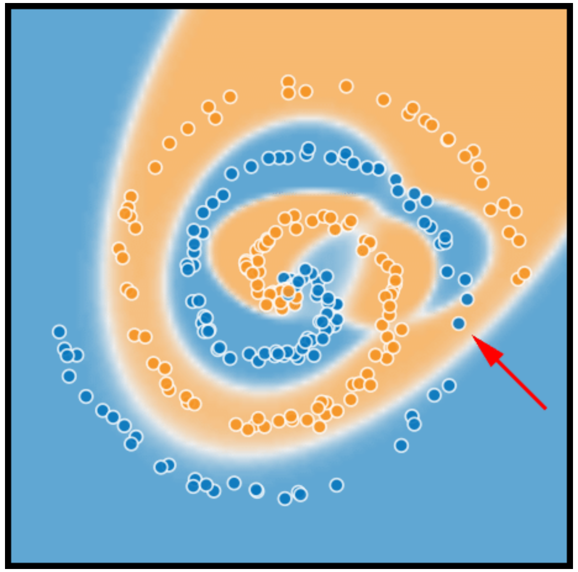
$$z_3 = \mathbf{w}_3 \cdot \mathbf{x} + b_3 = 2.5 \qquad \frac{e^{2.5}}{e^{-1.2} + e^{4.8} + e^{-0.7} + e^{2.5}} \approx 0.091 \qquad 0$$

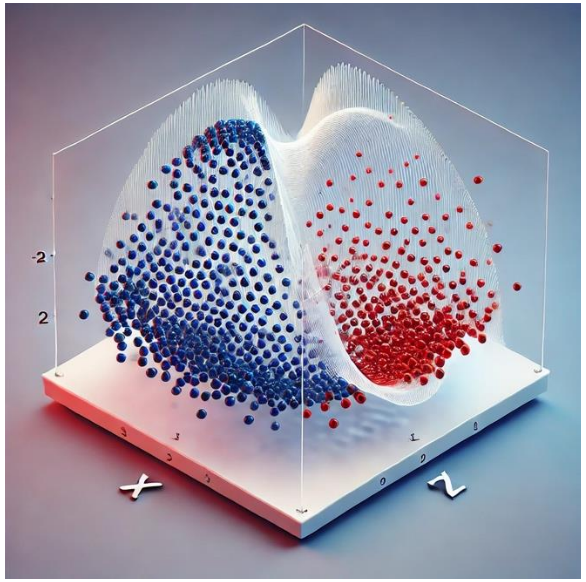
$$\mathcal{L}_{CE}(y, \tilde{y}) = -[0 \cdot \log 0.002 + 1 \cdot \log 0.903 + 0 \cdot \log 0.004 + 0 \cdot \log 0.091] \approx 0.102$$

Logistic Regression

- Logistic regression
  - Find linear weights to map feature vector $\mathbf{x}$ to label y

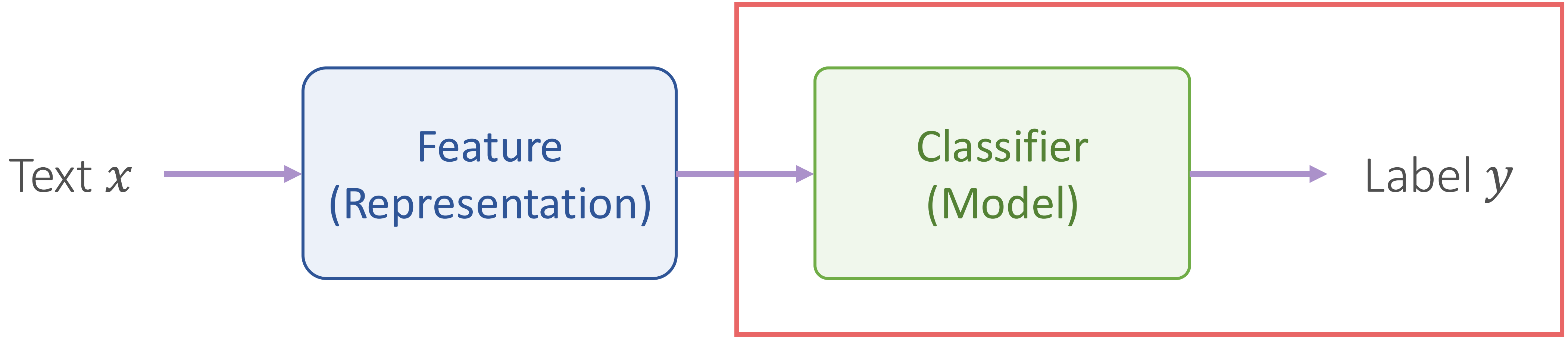
What if linear weights are not powerful enough?





https://www.researchgate.net/publication/344636757_CQNN_Convolutional_Quadratic_Neural_Networks
https://chatgpt.com/

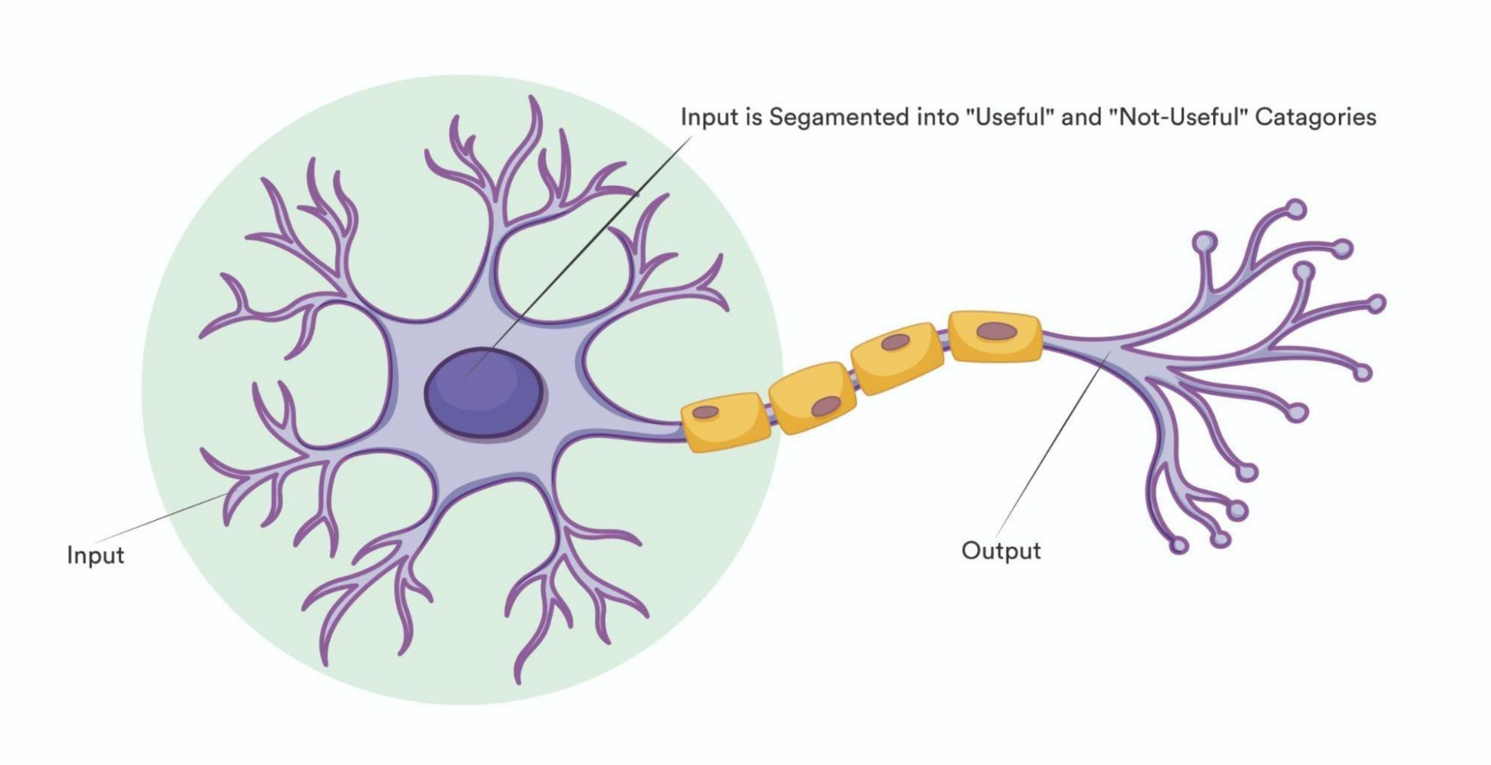
Neural Networks



- Neural Networks
  - Find a non-linear decision boundary to map feature vector $\mathbf{x}$ to label y

Biological Neurons

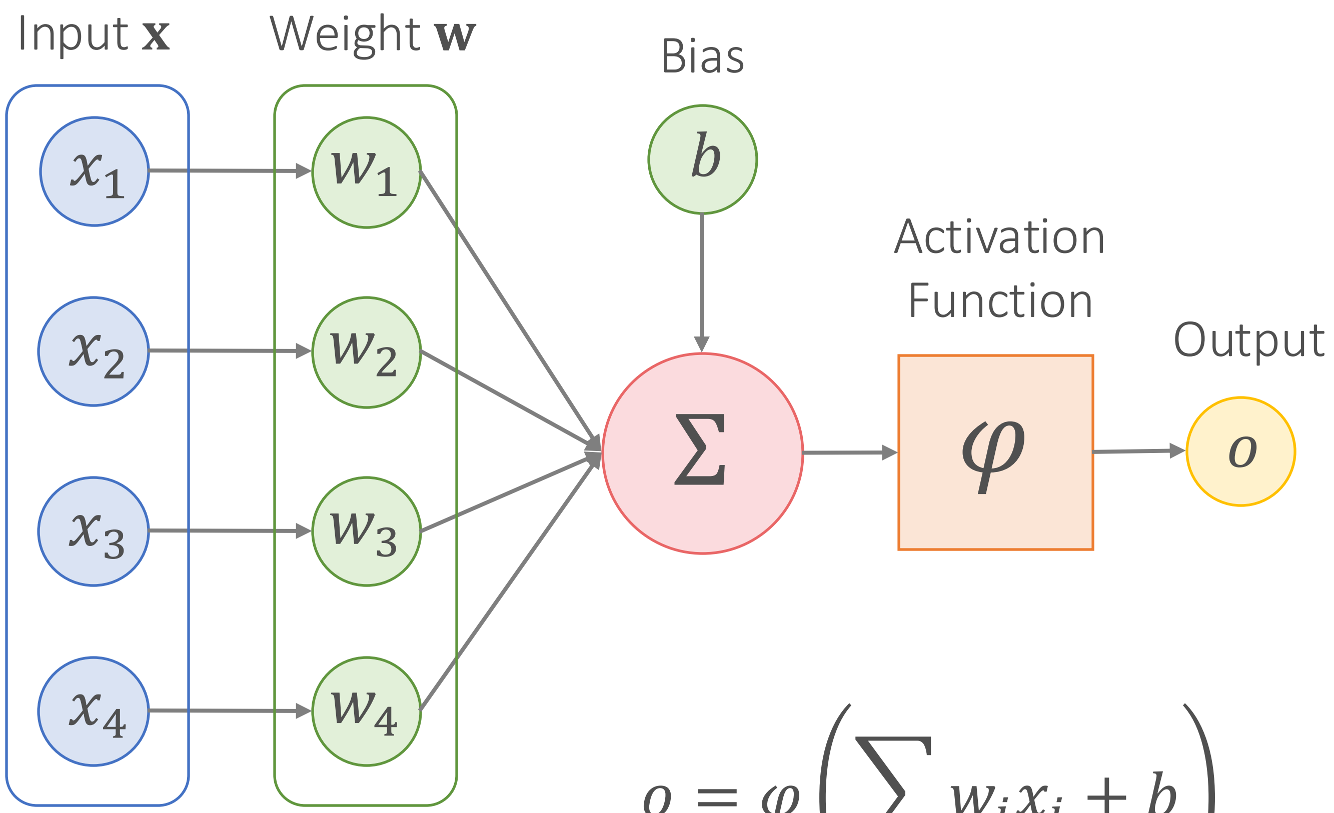
Neuron activation: A neuron becomes active to transmit information when it receives sufficient input from other neurons



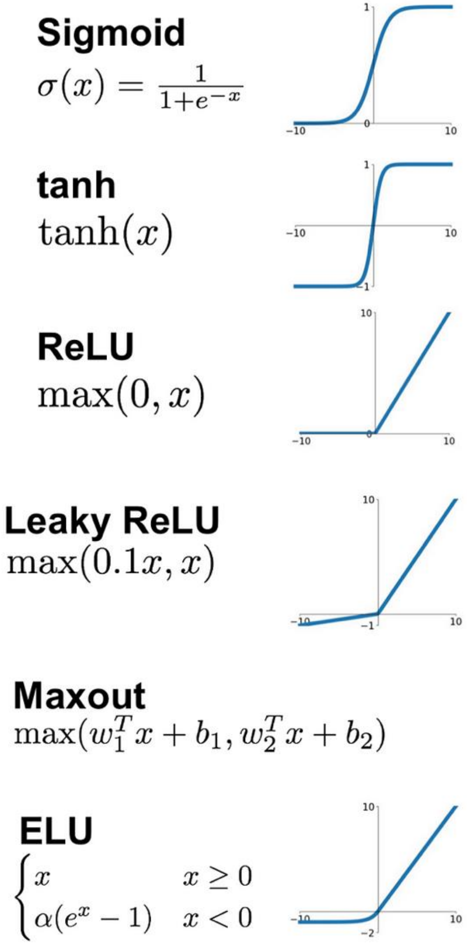
https://www.kdnuggets.com/2022/06/activation-functions-work-deep-learning.html

Neurons in Neural Networks

Mimic the behavior of neurons to transmit information

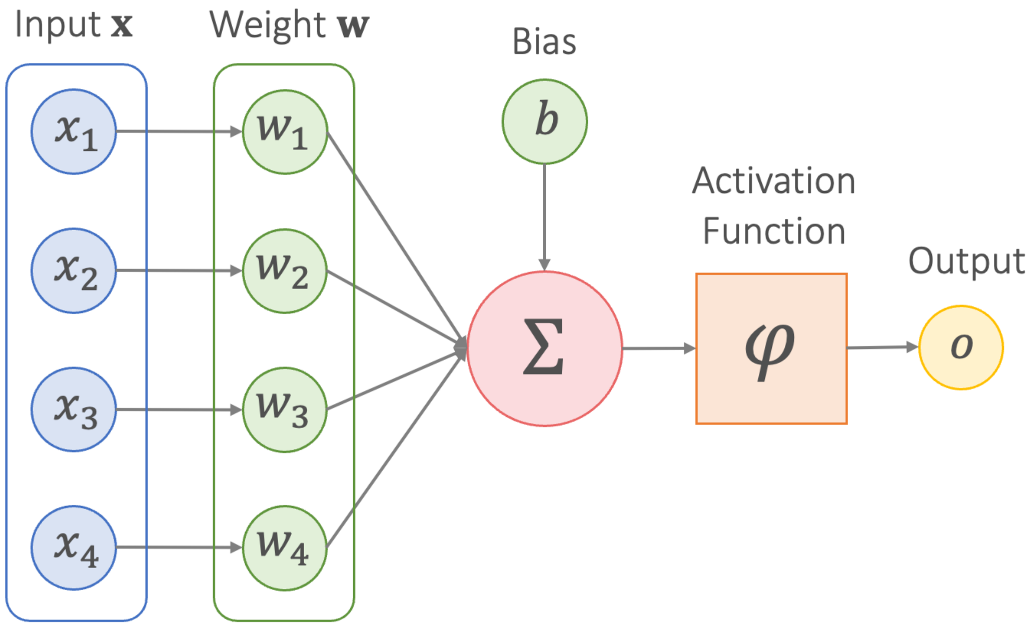


$$o = \varphi\left(\sum_i w_i x_i + b\right)$$



https://blog.devops.dev/exploring-activation-functions-in-deep-learning-properties-derivatives-and-impact-on-model-7585aad8a757

Neurons vs. Logistic Regression



Feature Vector $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$

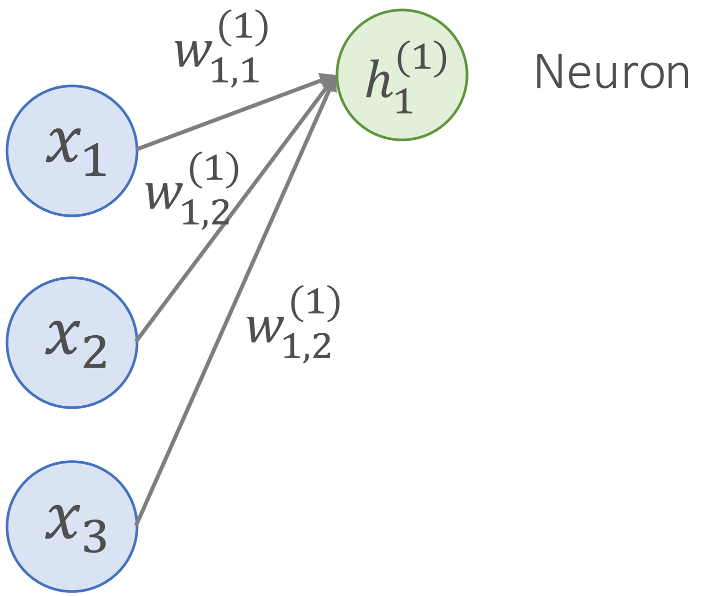
Weight Vector $\mathbf{w} = [w_1, w_2, w_3, \dots, w_d]$

Bias b

$$o = \varphi\left(\sum_i w_i x_i + b\right)$$

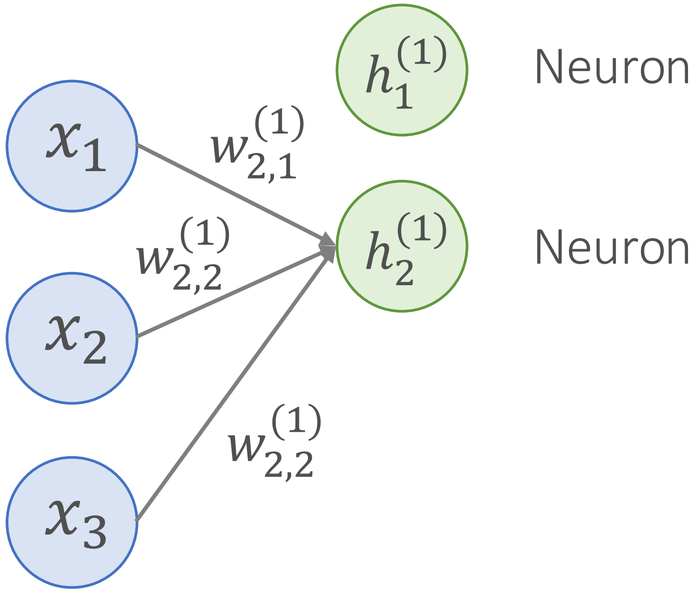
$$\tilde{y} = \sigma\left(\sum_i w_i x_i + b\right)$$

Multilayer Perceptron (MLP)



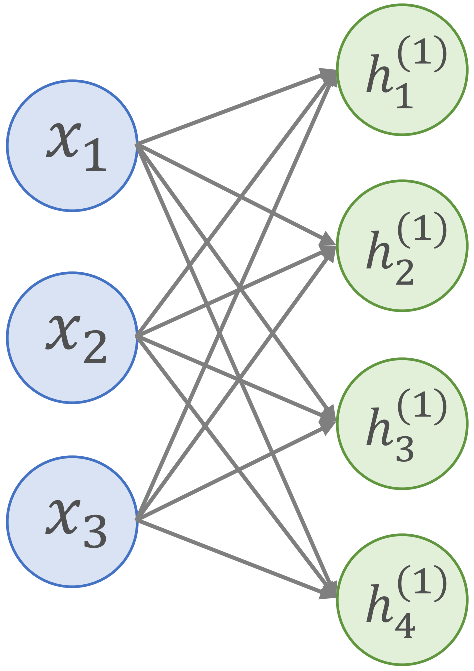
$$h_1^{(1)} = \varphi\left(\sum_i w_{1,i}^{(1)} x_i + b\right) = \varphi\left(\mathbf{w}_1^{(1)} \cdot \mathbf{x} + b\right)$$

Multilayer Perceptron (MLP)



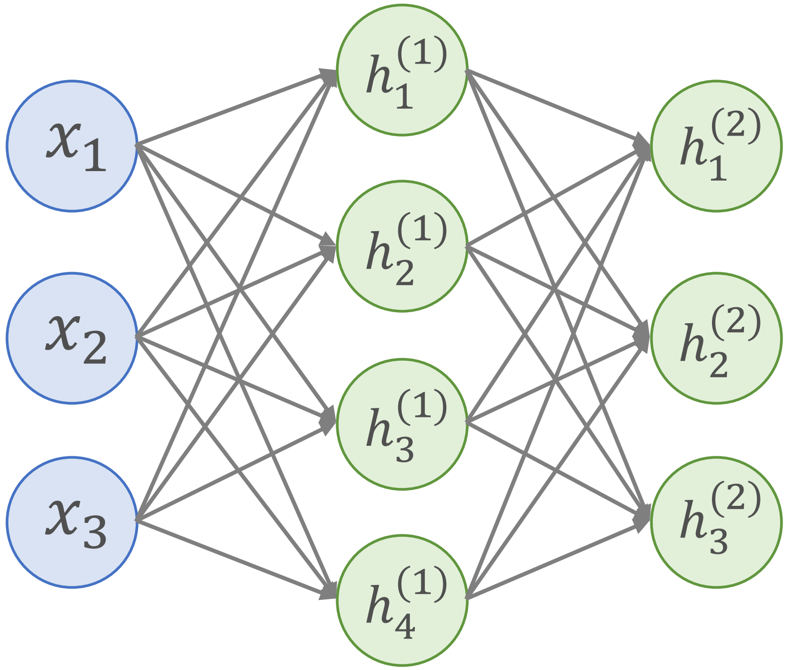
$$h_2^{(1)} = \varphi\left(\sum_i w_{2,i}^{(1)} x_i + b\right) = \varphi\left(\mathbf{w}_2^{(1)} \cdot \mathbf{x} + b\right)$$

Multilayer Perceptron (MLP)



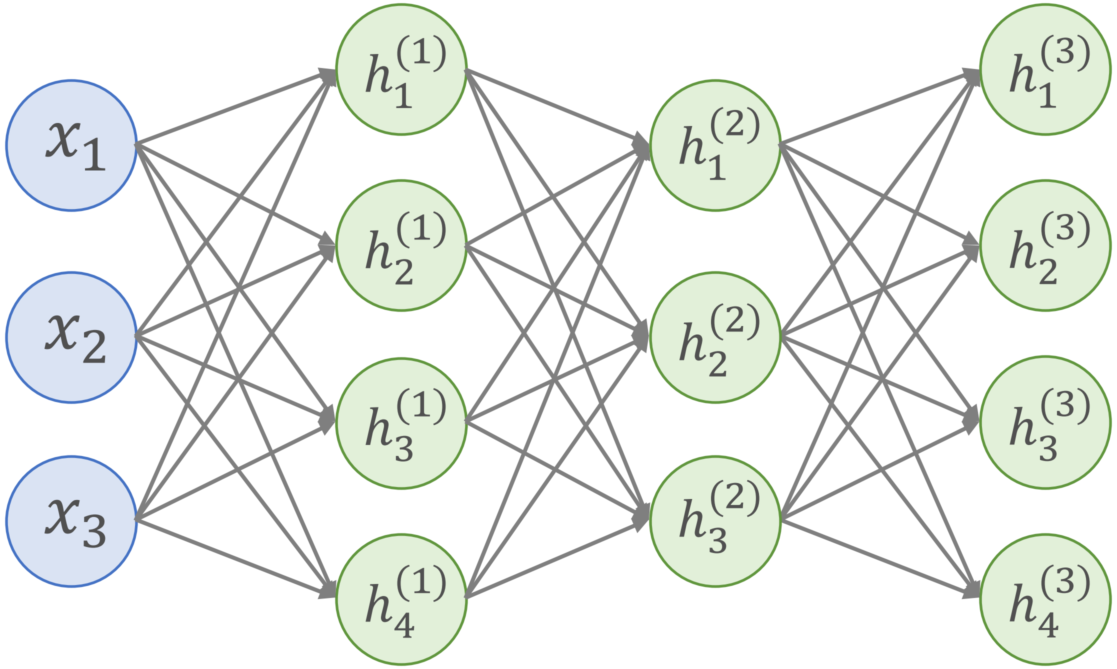
$$\mathbf{h}^{(1)} = \varphi\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

Multilayer Perceptron (MLP)



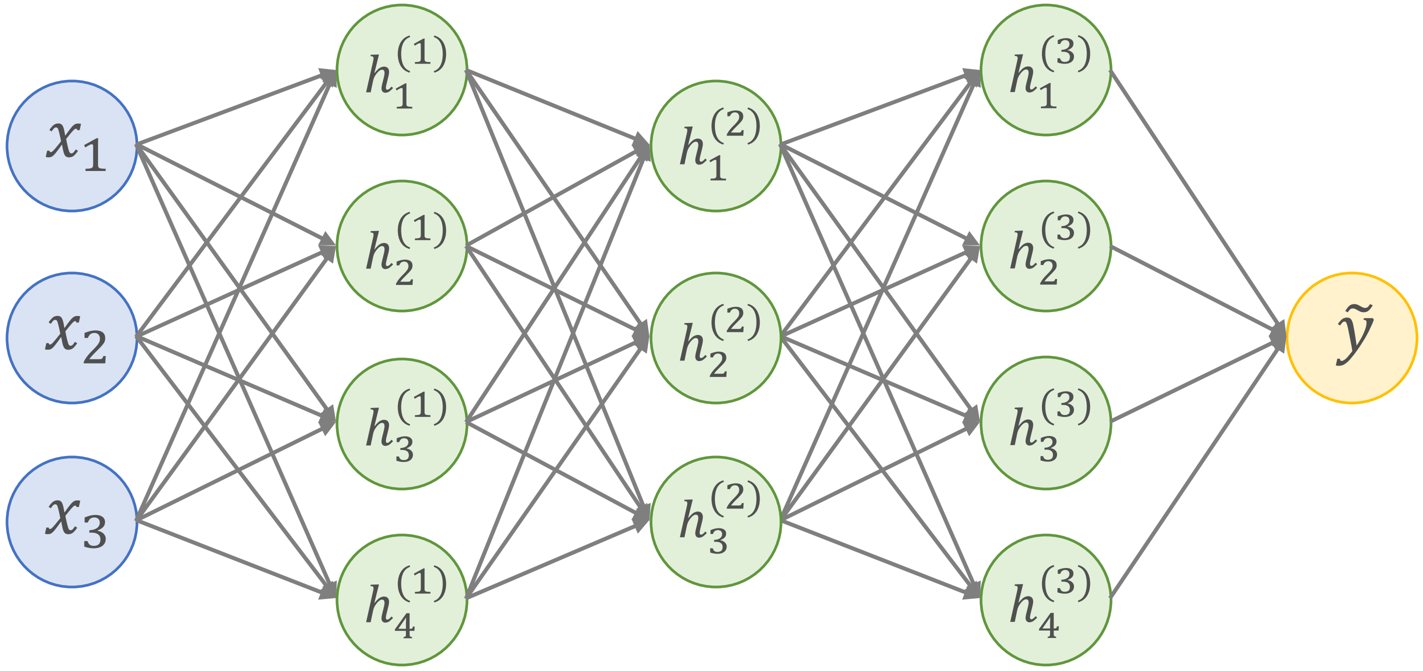
$$\mathbf{h}^{(2)} = \varphi\left(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right)$$

Multilayer Perceptron (MLP)



$$\mathbf{h}^{(3)} = \varphi\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right)$$

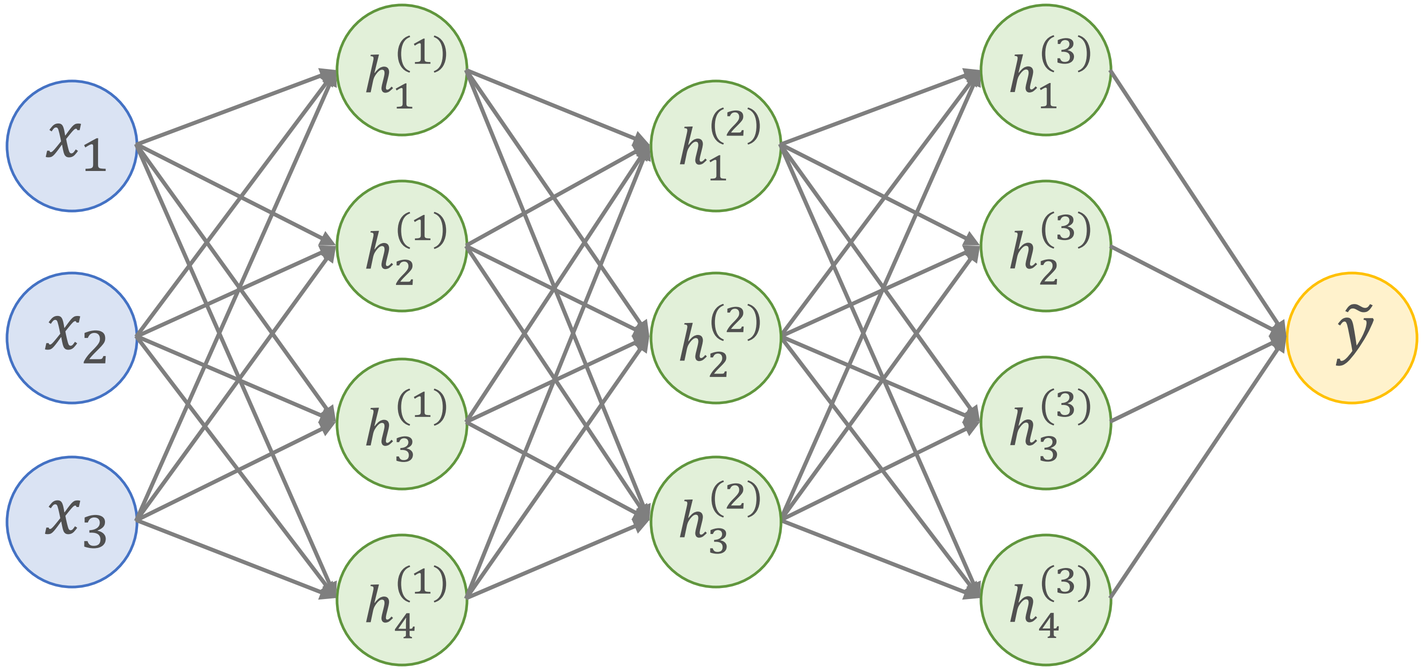
Multilayer Perceptron (MLP)



Decision boundary: $= \begin{cases} 1 & \text{If } \tilde{y} \geq 0.5 \\ 0 & \text{If } \tilde{y} < 0.5 \end{cases}$

$$\tilde{y} = \sigma\left(\mathbf{W}^{(o)}\mathbf{h}^{(3)} + \mathbf{b}^{(o)}\right)$$

Optimization Objective



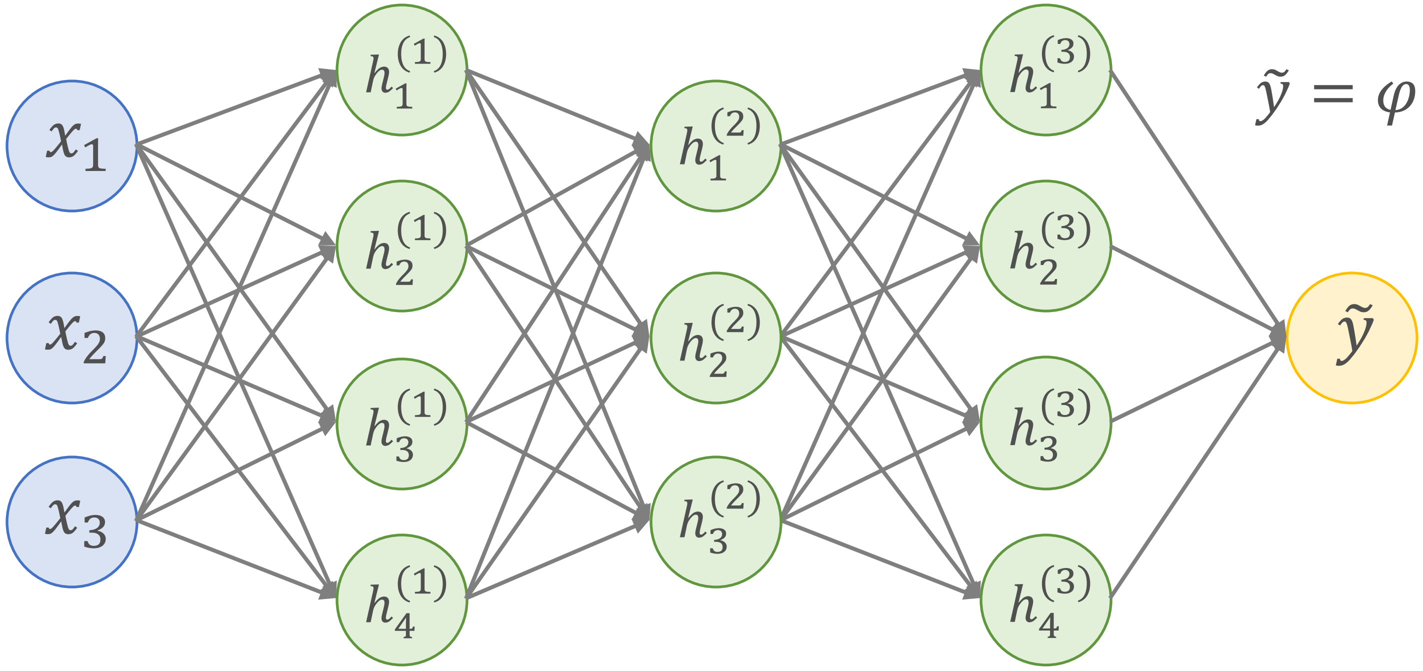
Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_{i} \mathcal{L}_{CE}(y_i, \widetilde{y}_i)$$

Parameters $\theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \mathbf{W}^{(o)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \mathbf{b}^{(3)}, \mathbf{b}^{(o)}\}$,

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{total}$$

Back-Propagation



$$\tilde{y} = \varphi\left(\mathbf{W}^{(o)}\mathbf{h}^{(3)} + \mathbf{b}^{(o)}\right)$$

$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

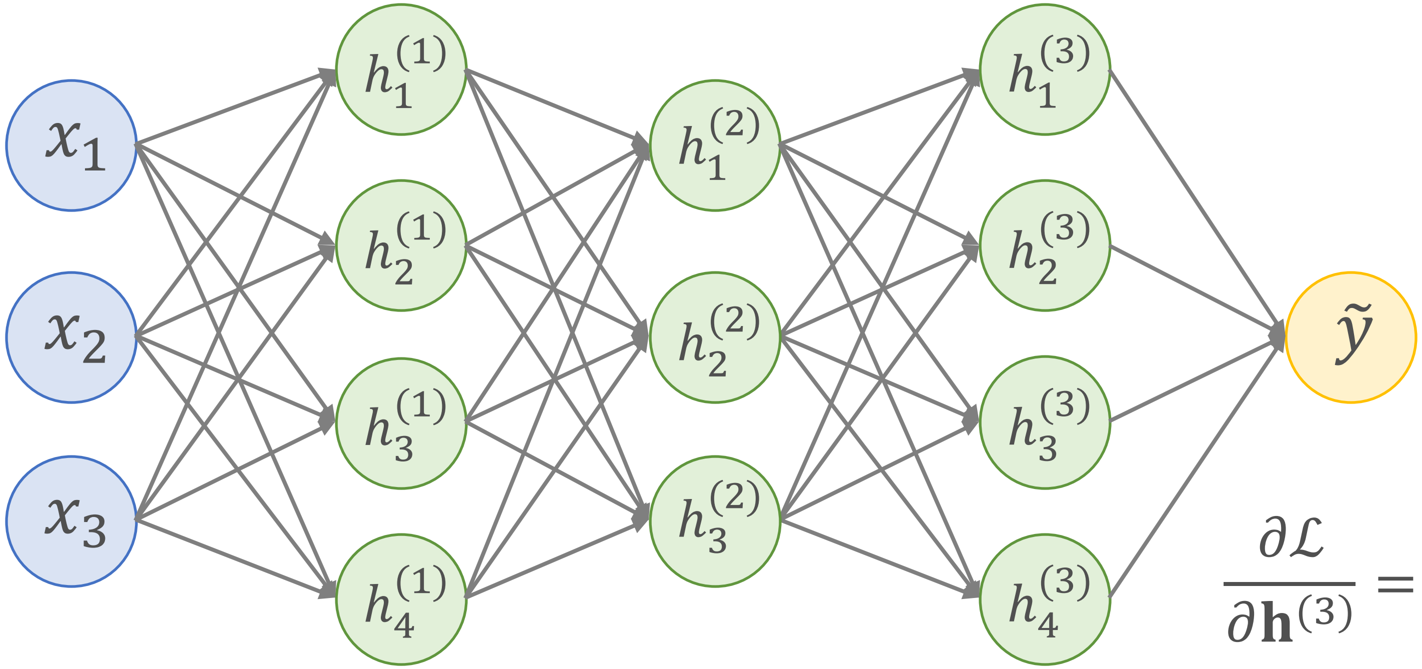
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(o)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{W}^{(o)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(o)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{b}^{(o)}}$$

Back-Propagation

$$\mathbf{h}^{(3)} = \varphi\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right)$$



$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

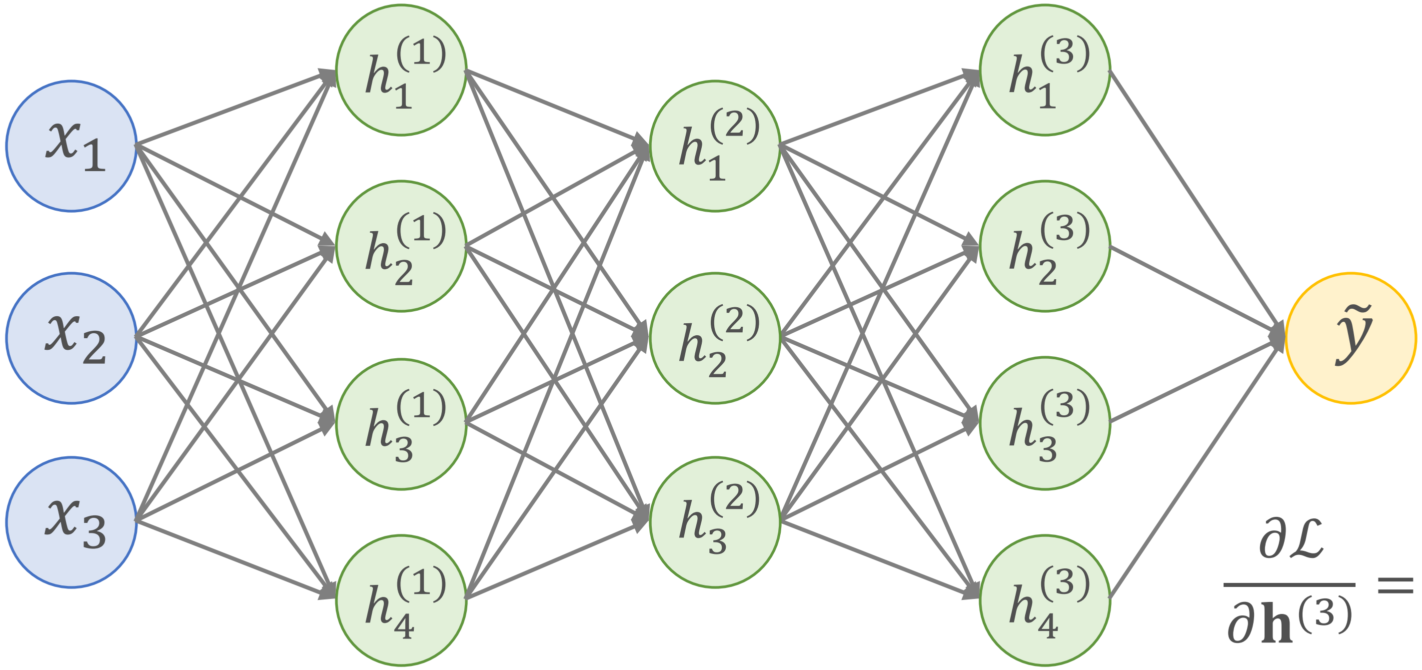
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{W}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{b}^{(3)}}$$

Back-Propagation



$$\frac{\partial \mathcal{L}}{\partial \tilde{y}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial \mathbf{h}^{(3)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(3)}} \cdot \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

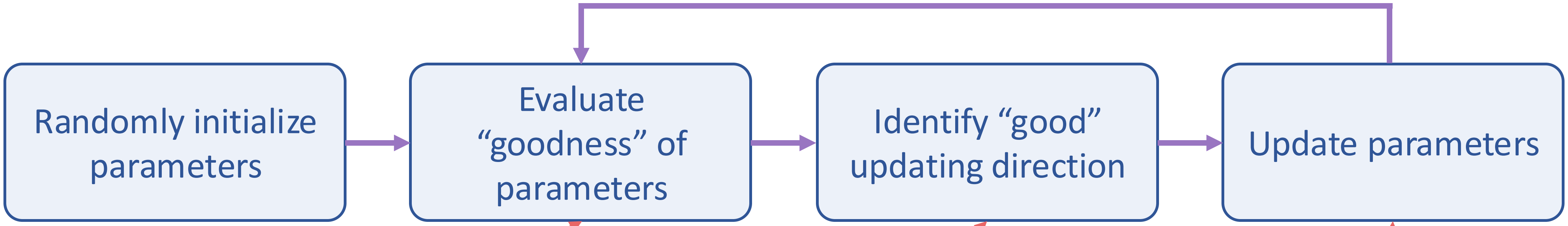
$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(2)}} \cdot \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} \cdot \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{W}^{(1)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(1)}} \cdot \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{b}^{(1)}}$$

Training Process

Iterative Optimization Methods



Cross Entropy Loss

$$\mathcal{L}_{total} = -\frac{1}{m}\sum_i \mathcal{L}_{CE}\left(y_i, \tilde{y}_i; \mathbf{w}^{(t)}, b^{(t)}\right)$$

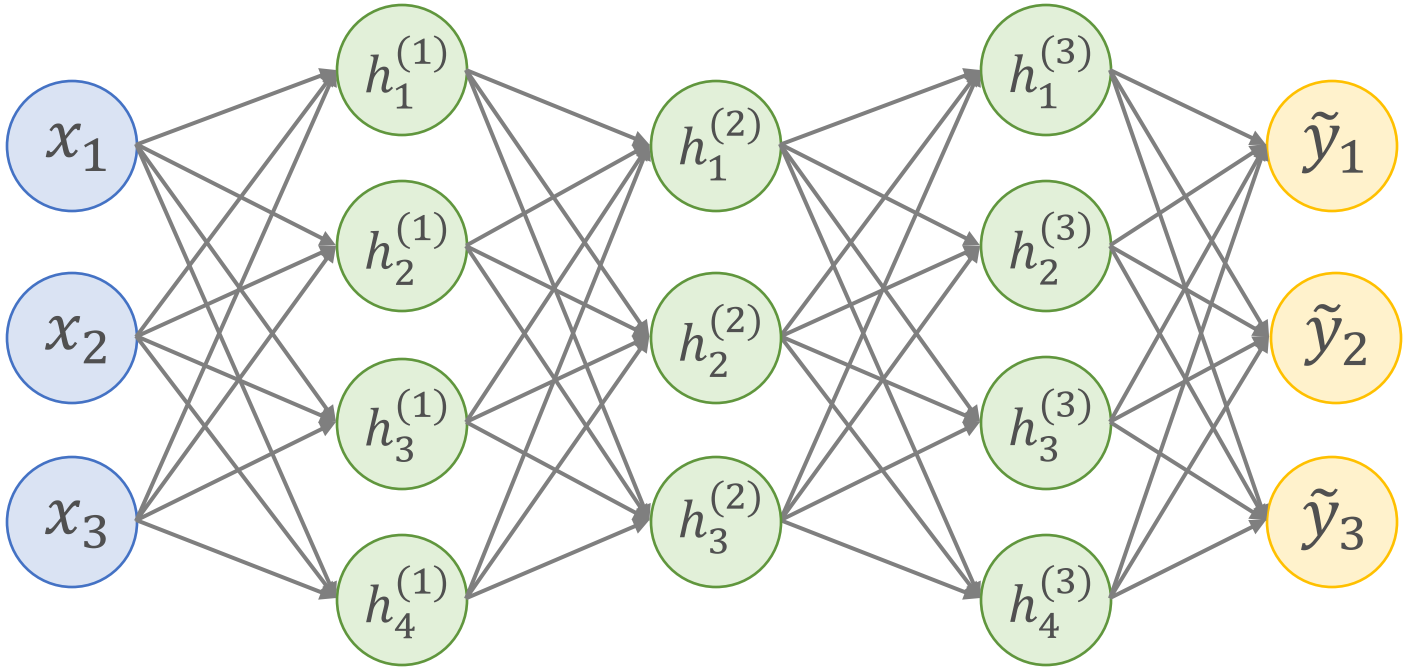
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(2)}}, \dots, \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(o)}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(2)}}, \dots, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(o)}}$$

$$\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}$$

$$\mathbf{b}^{(1)} \leftarrow \mathbf{b}^{(1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}$$

From Binary to Multiclass Classification

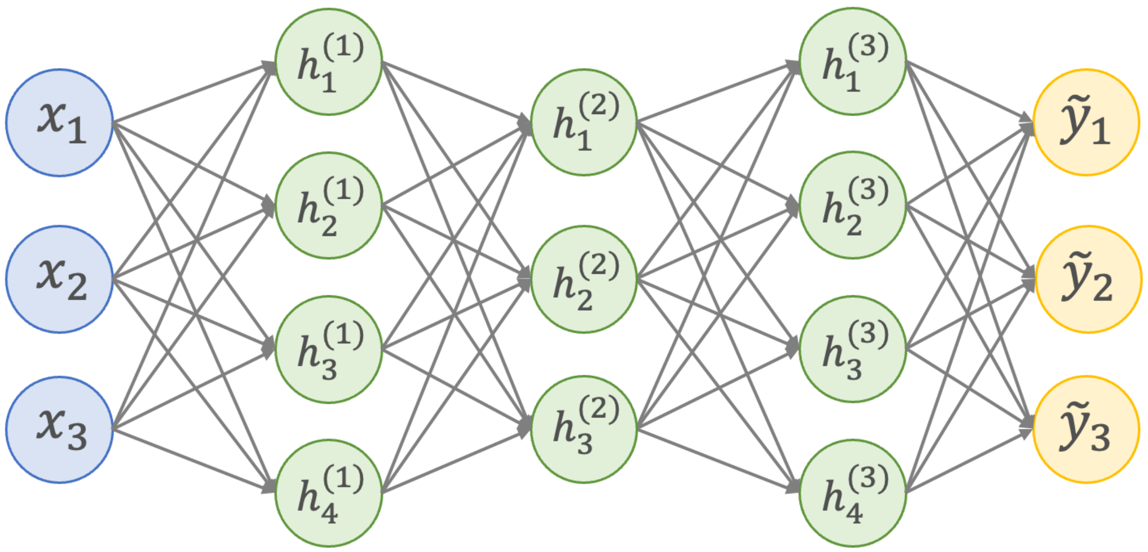


Prediction $= \arg\max_{\mathbf{c}} \tilde{y}_c$

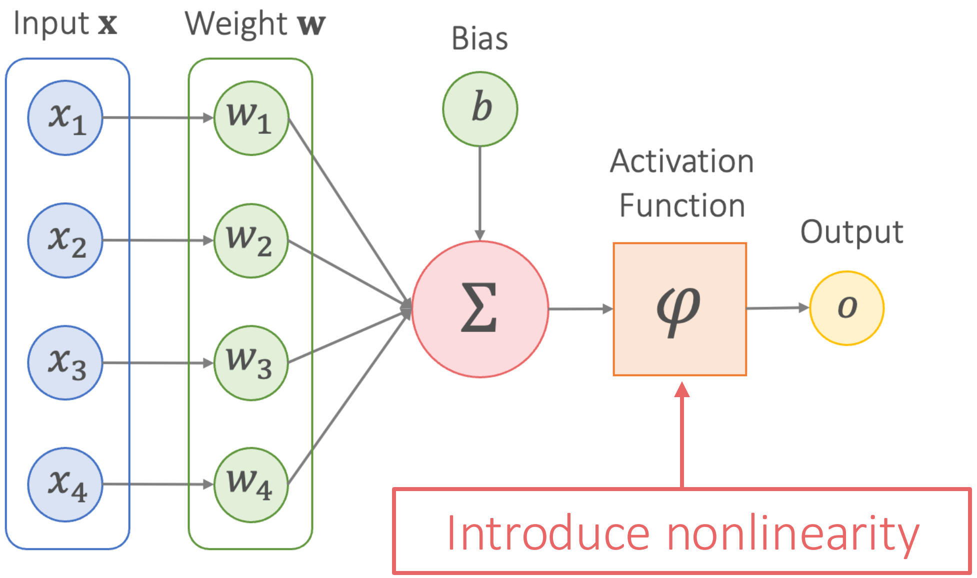
Multiclass Cross Entropy Loss

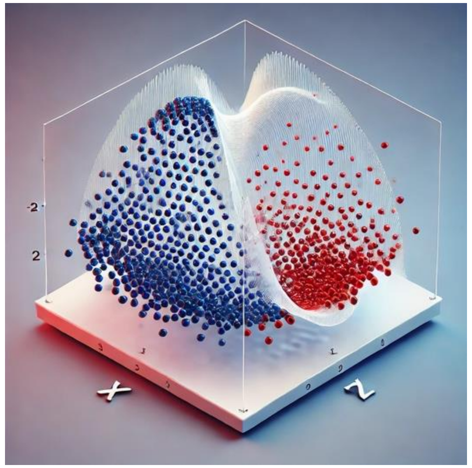
$$\mathcal{L}_{CE}(y, \tilde{y}) = -\sum_{c=0}^{C} y_c \log P(y = c \,|\, \mathbf{x})$$

What Makes Neural Networks Powerful?

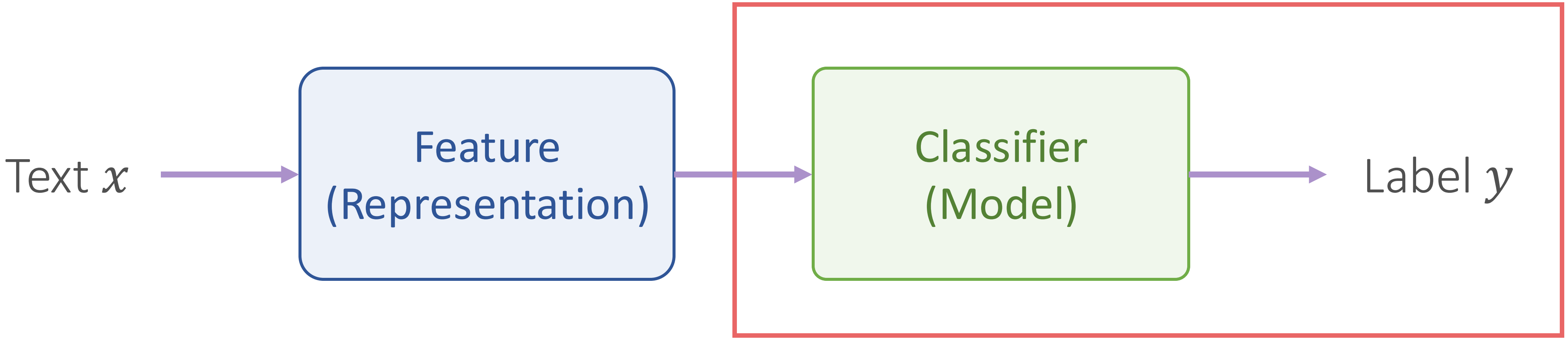


Nonlinear Transform $\rightarrow$ Nonlinear Transform $\rightarrow$ Nonlinear Transform $\rightarrow$





Neural Networks

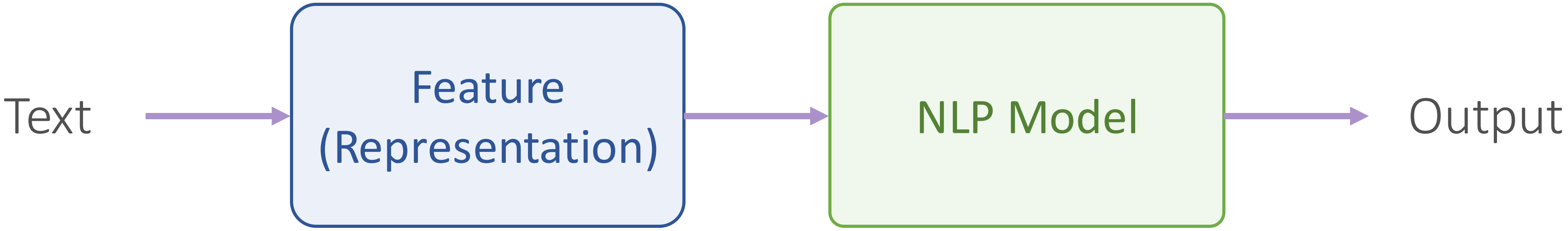


- Neural Networks
  - Find a non-linear decision boundary to map feature vector $\mathbf{x}$ to label y

Lecture Plan

- Formulation of Text Classification
- Bag-of-Words (Bow) and N-Grams
- Logistic Regression
- Neural Networks

Next Lecture: Word Representations



$$\begin{array}{c c} & \begin{array}{ccccc} \mathit{Bob} & \mathit{likes} & \mathit{Alice} & \mathit{very} & \mathit{much} \end{array} \\ W = & \begin{bmatrix} | & | & | & | & | \\ w_{bob} & w_{likes} & w_{Alice} & w_{very} & w_{much} \\ | & | & | & | & | \end{bmatrix} \end{array}$$