

Cornell Bowers C-IS
College of Computing and Information Science

Deep Learning

Recap & Multi-Layer Perceptrons

Quick Recap- Logistics



CS4782 cornell SP25

AI Mode All Images Shopping Videos News Forums More ▾

These are results for **CS 4782** cornell SP25 • [↶ Revert](#)



Cornell University

<https://classes.cornell.edu> › browse › roster › SP25 › class ⋮

Spring 2025 - CS 4782 - Class Roster

Spring 2025 - **CS 4782** - This class is an introductory course to deep learning. It covers the fundamental principles behind training and inference of deep ...



Cornell Computer Science Department

<https://www.cs.cornell.edu> › courses › cs4782 ⋮

CS 4/5782 - Cornell Computer Science

CS 4782: Intro to Deep Learning, Spring 2025. Overview; Assignments; Schedule; References; Policies.

Instructors: Kilian Q. Weinberger and Jennifer J. Sun.

Missing: SP25 | Show results with: [SP25](#)

Quick Recap- Logistics



CS4782 cornell SP25

AI Mode

All

Images

Shopping

Videos

News

Forums

More ▾

No laptops/mobiles/smart devices in

These are results for **CS 4782 cornell SP25** • [↶ Revert](#)



<https://classes.cornell.edu> › browse › roster › SP25 › class

Spring 2025 - CS 4782 - Class Roster

Spring 2025 - **CS 4782** - This class is an introductory course to deep learning. It covers the fundamental principles behind training and inference of deep ...

class please!



Cornell Computer Science Department

<https://www.cs.cornell.edu> › courses › cs4782

CS 4/5782 - Cornell Computer Science

CS 4782: Intro to Deep Learning, Spring 2025. Overview; Assignments; Schedule; References; Policies.

Instructors: Kilian Q. Weinberger and Jennifer J. Sun.

Missing: SP25 | Show results with: SP25

Agenda

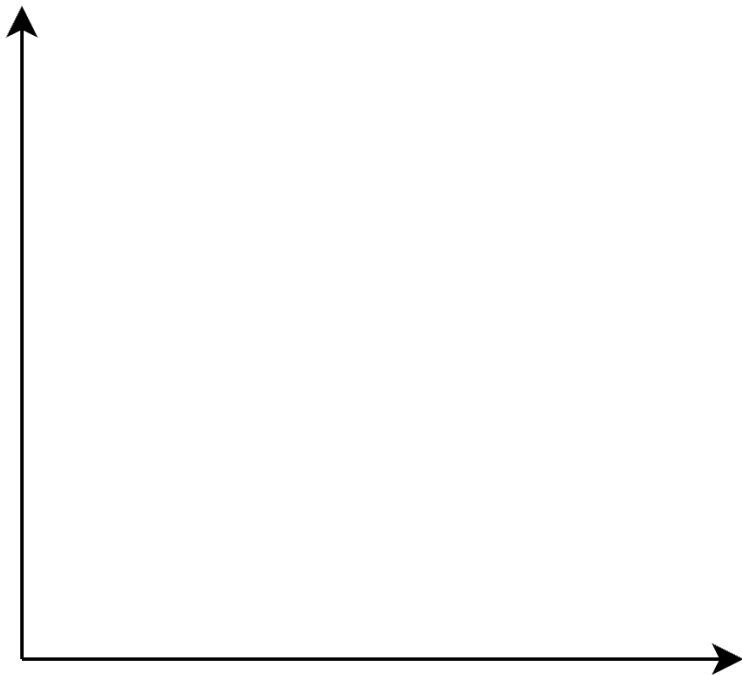
- Perceptron
- Logistic Regression
- Gradient Descent
- Multi-Layer Perceptrons (MLPs)
- Backpropagation

A Classification Problem:

Will I Pass This Class?

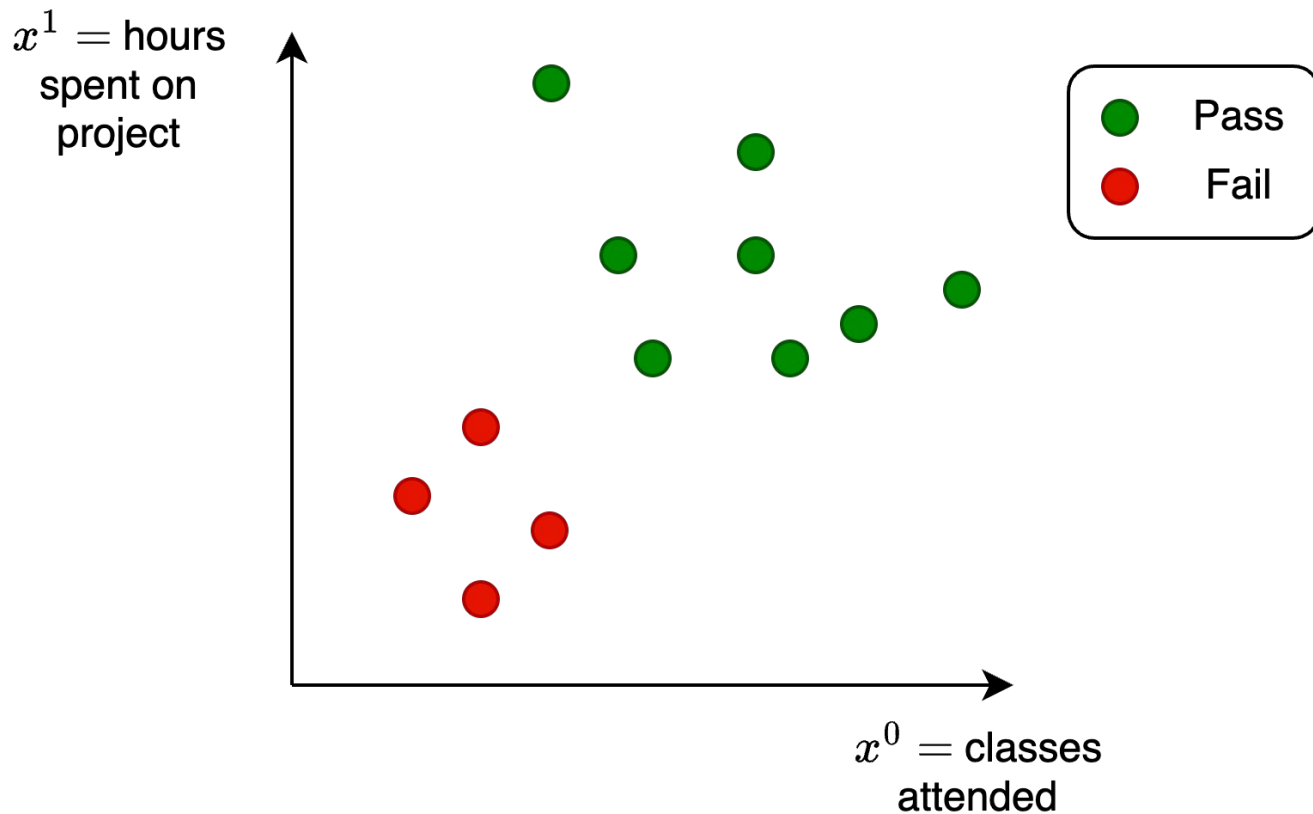
A Classification Problem: Will I Pass This Class?

x^1 = hours
spent on
project

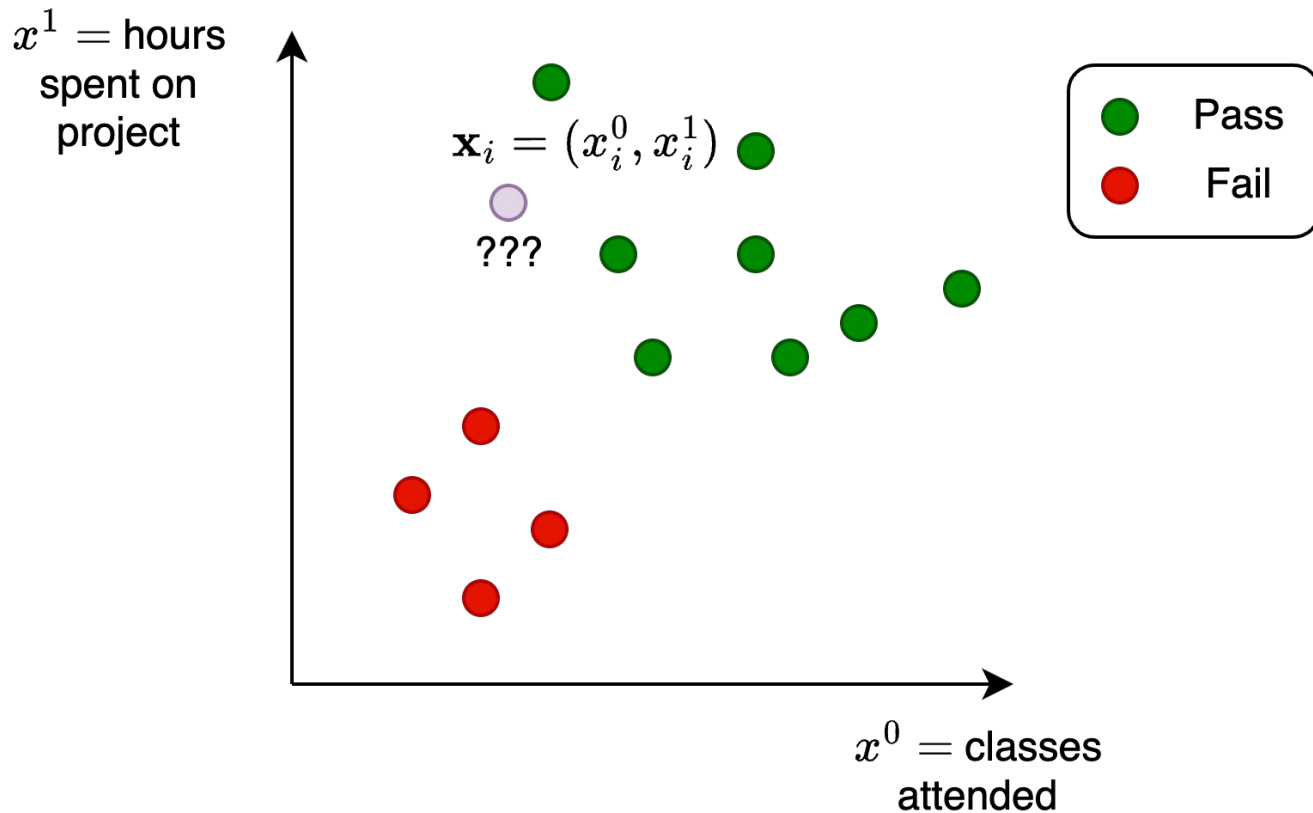


x^0 = classes
attended

A Classification Problem: Will I Pass This Class?



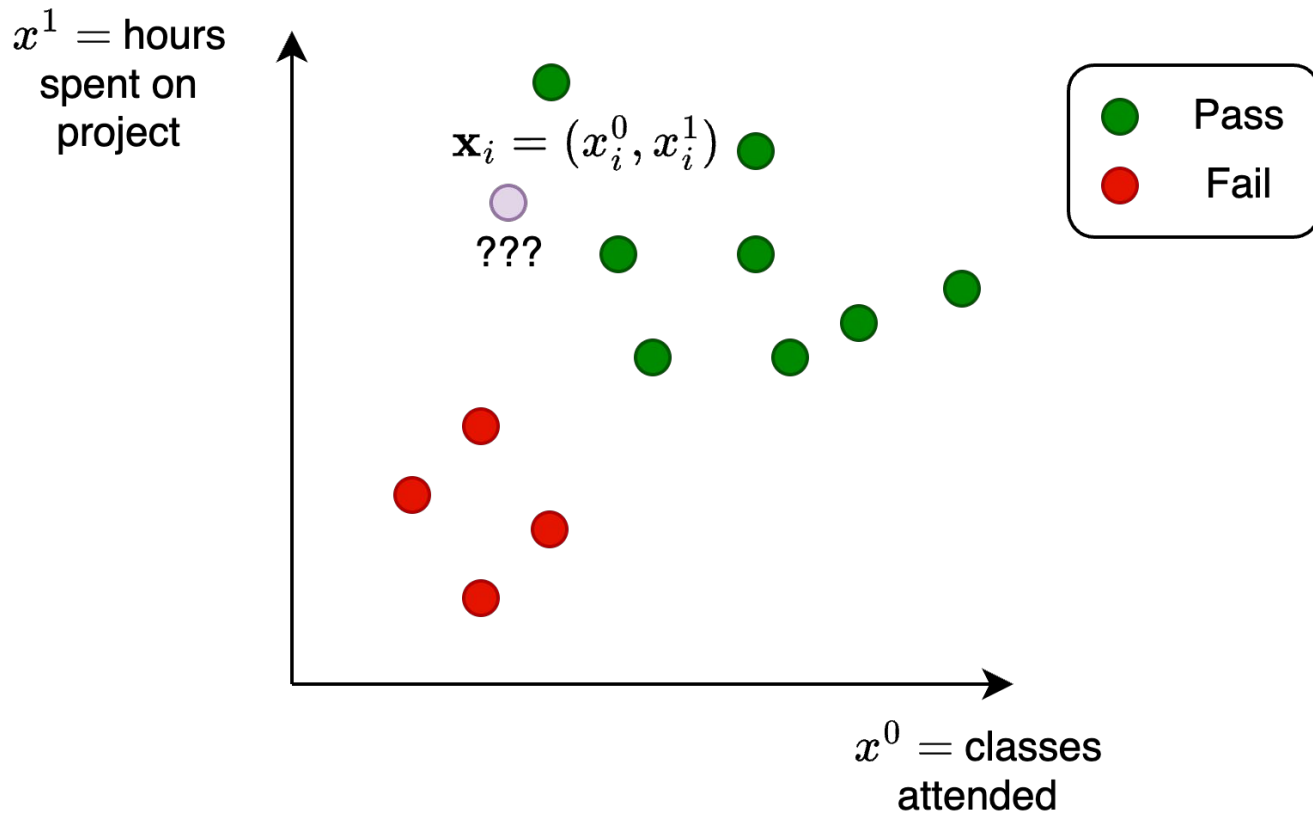
A Classification Problem: Will I Pass This Class?



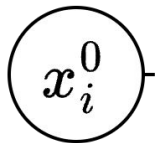
What are key components in ML?

- Training data
- Model Class / Hypothesis space
- Loss function
- Optimization

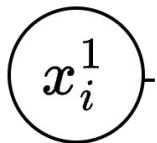
A Classification Problem: Will I Pass This Class?



Perceptron



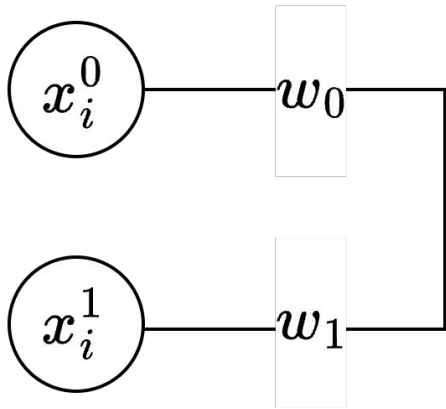
A circle containing the mathematical expression x_i^0 . A small horizontal line segment extends from the right side of the circle.



A circle containing the mathematical expression x_i^1 . A small horizontal line segment extends from the right side of the circle.

Input

Perceptron

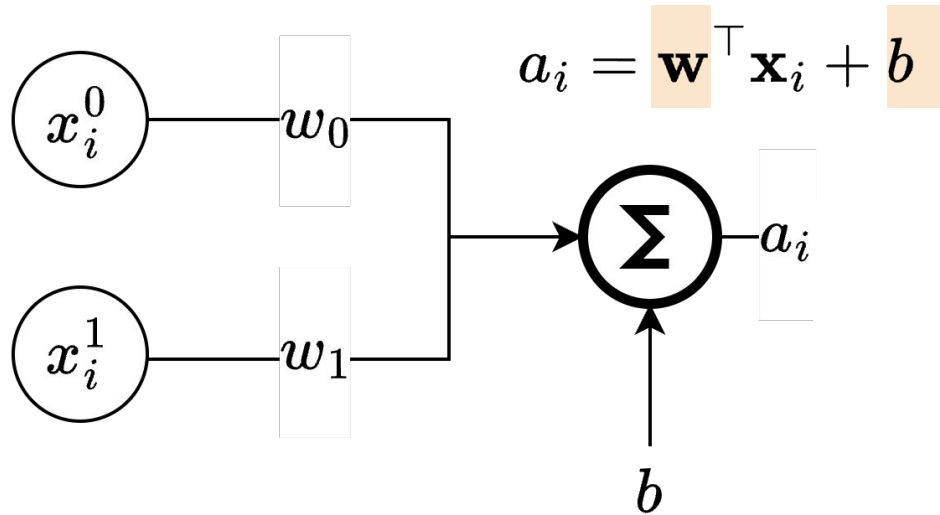


Input

Weights

Perceptron

Learned from training data

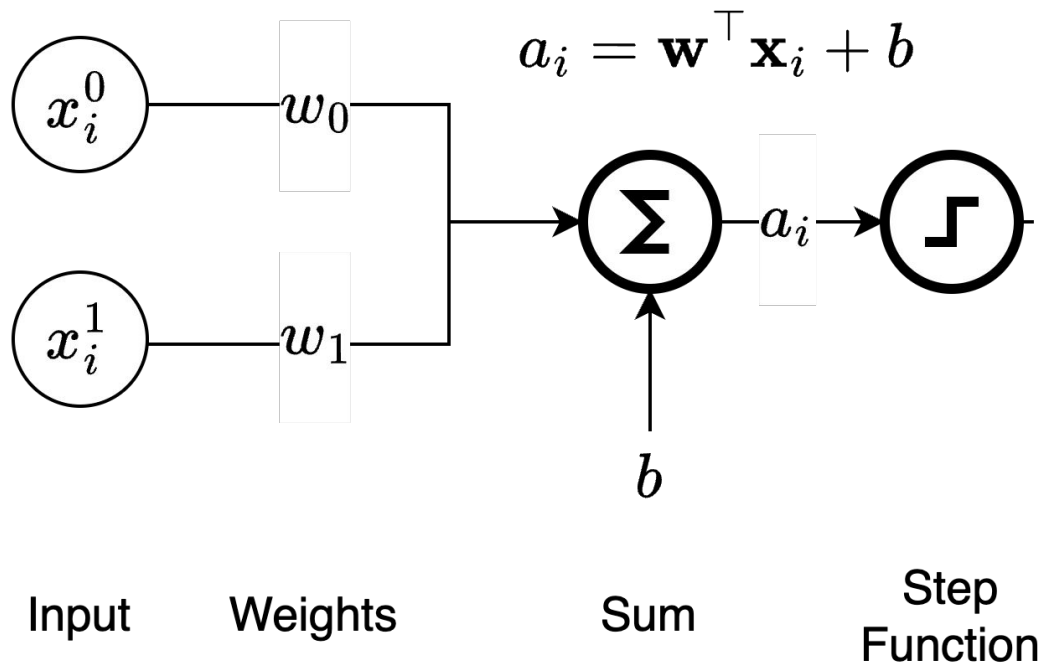


Input

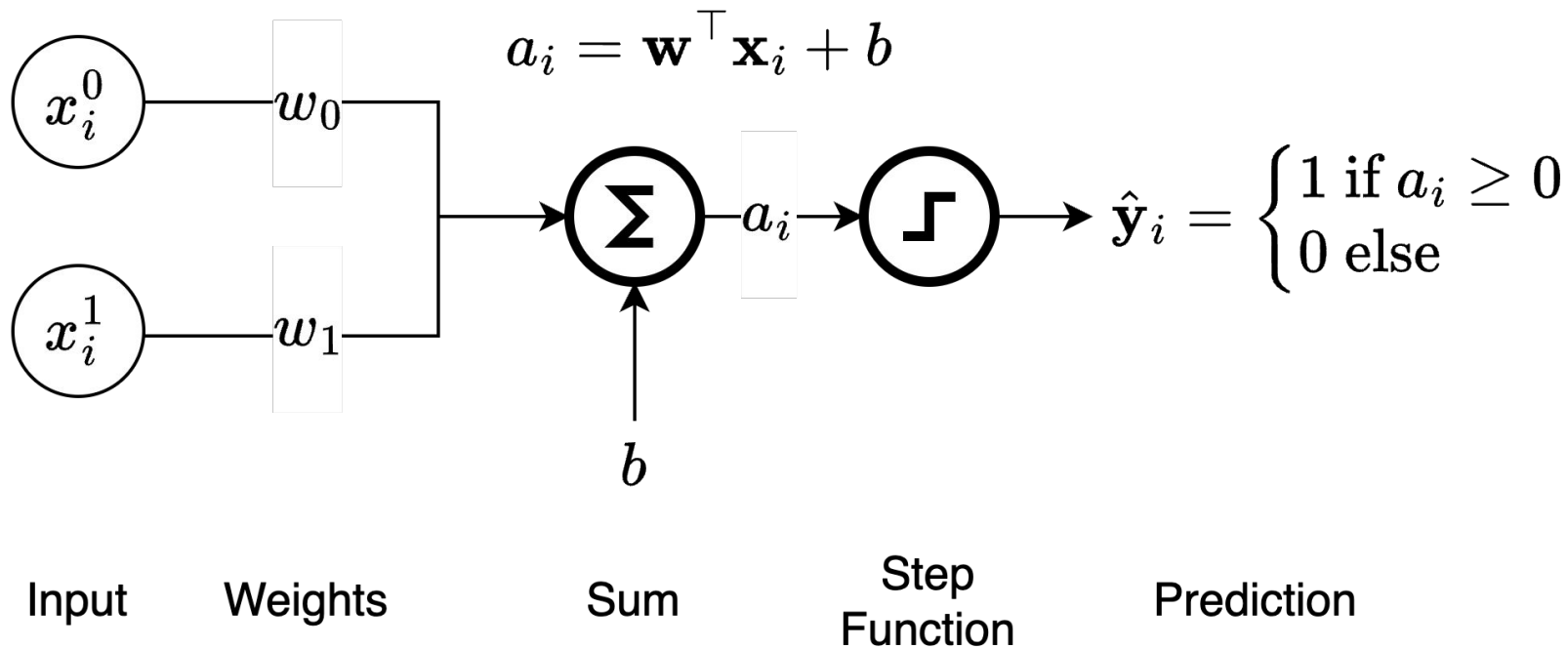
Weights

Sum

Perceptron

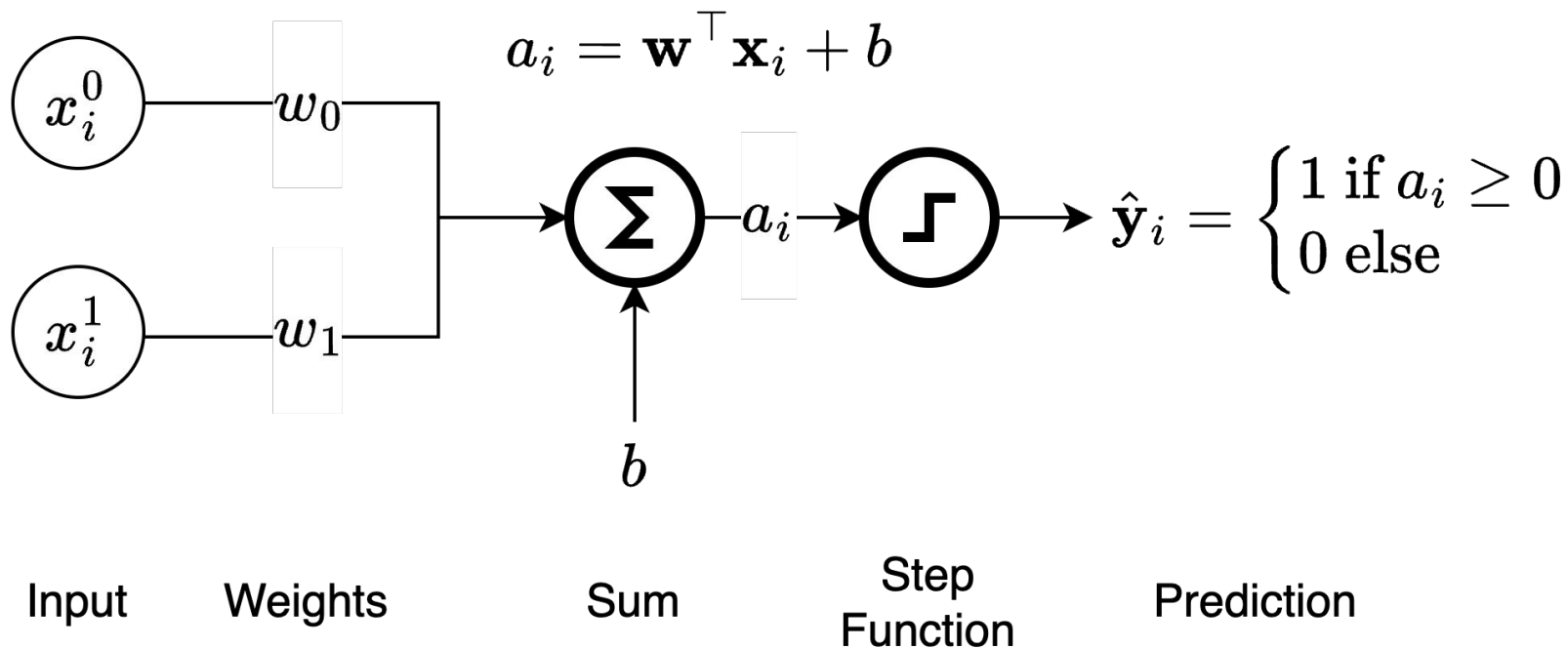


Perceptron



Perceptron

- Linear classifier
 - Predecessor to neural network

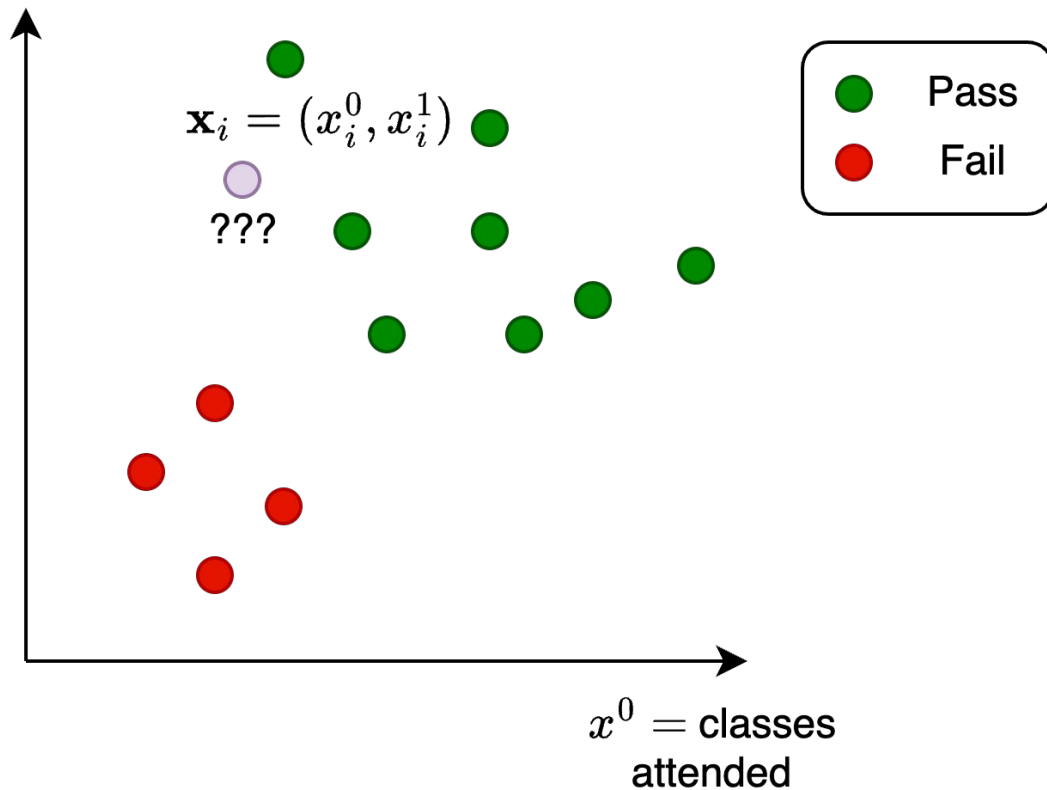


A Classification Problem: Will I Pass This Class?

x^1 = hours
spent on
project

Recall:

$$y_i = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x}_i + b \geq 0 \\ 0 & \text{else} \end{cases}$$

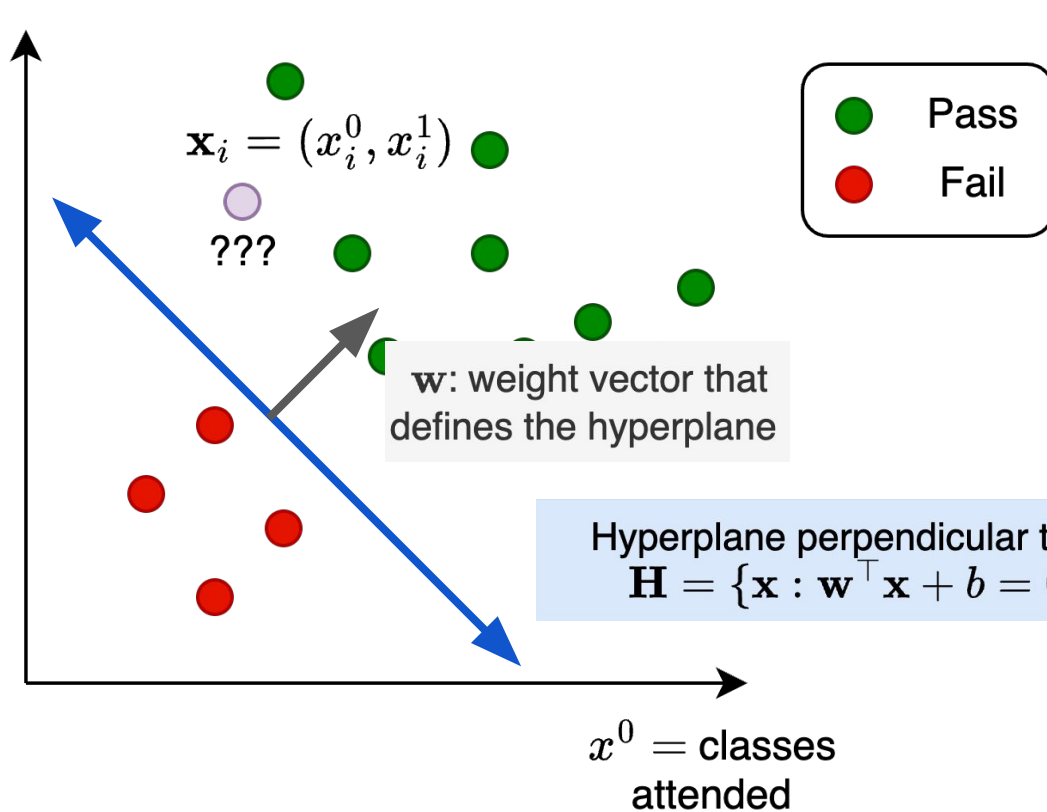


A Classification Problem: Will I Pass This Class?

x^1 = hours
spent on
project

Recall:

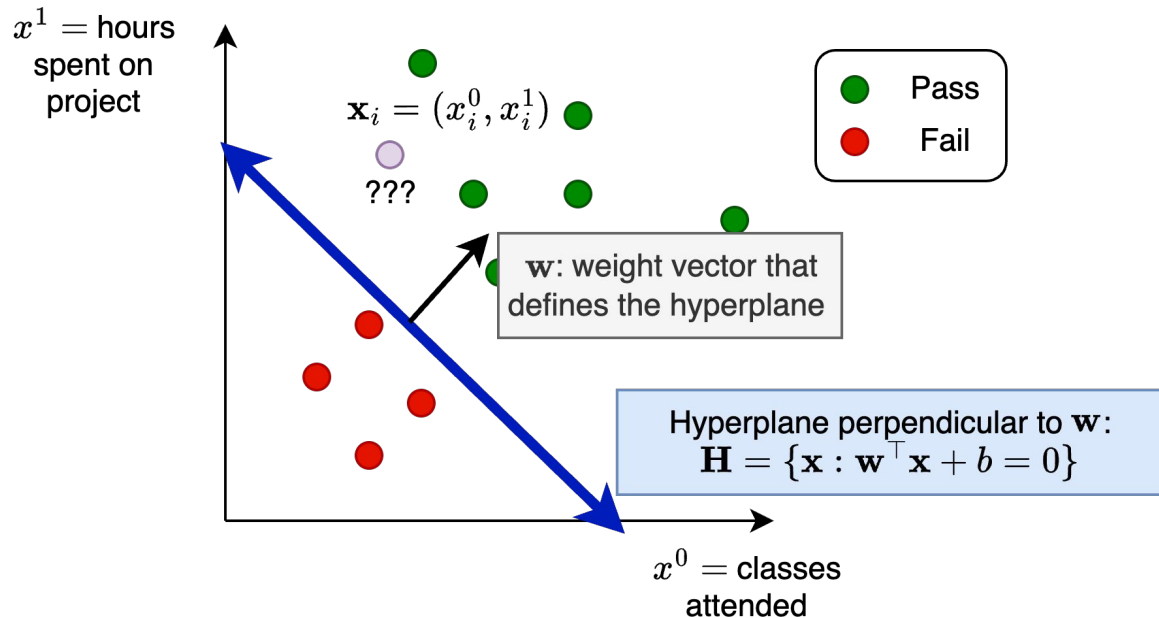
$$y_i = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x}_i + b \geq 0 \\ 0 & \text{else} \end{cases}$$



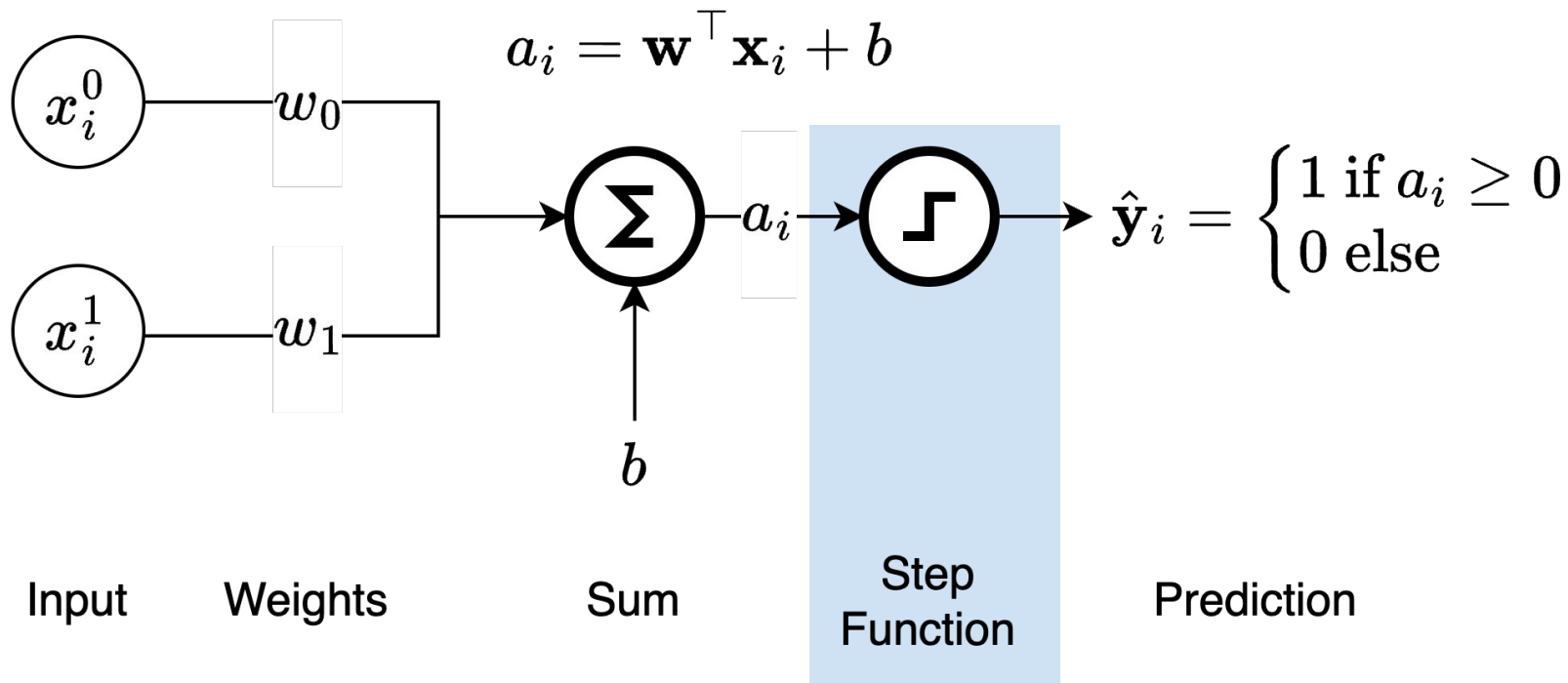
A Classification Problem: Will I Pass This Class?

- Perceptron defines a linear classification boundary

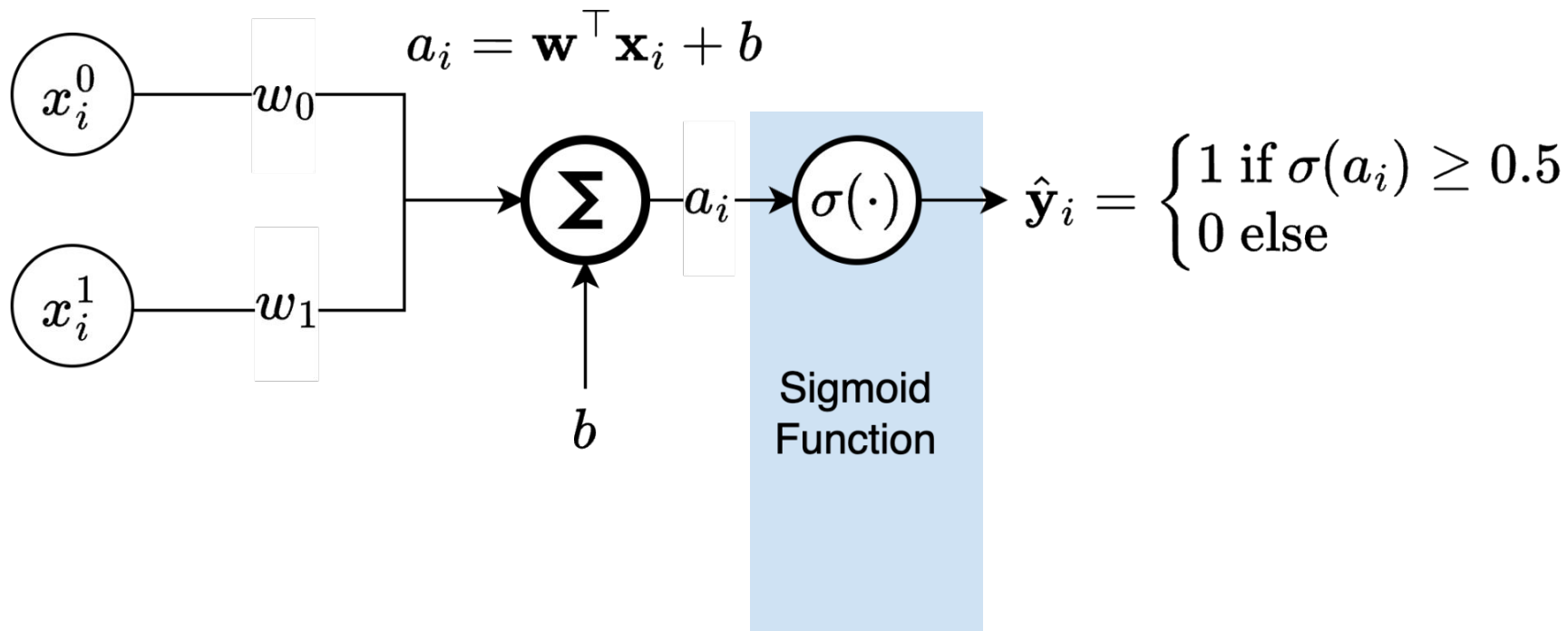
$$y_i = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x}_i + b \geq 0 \\ 0 & \text{else} \end{cases}$$



Perceptron

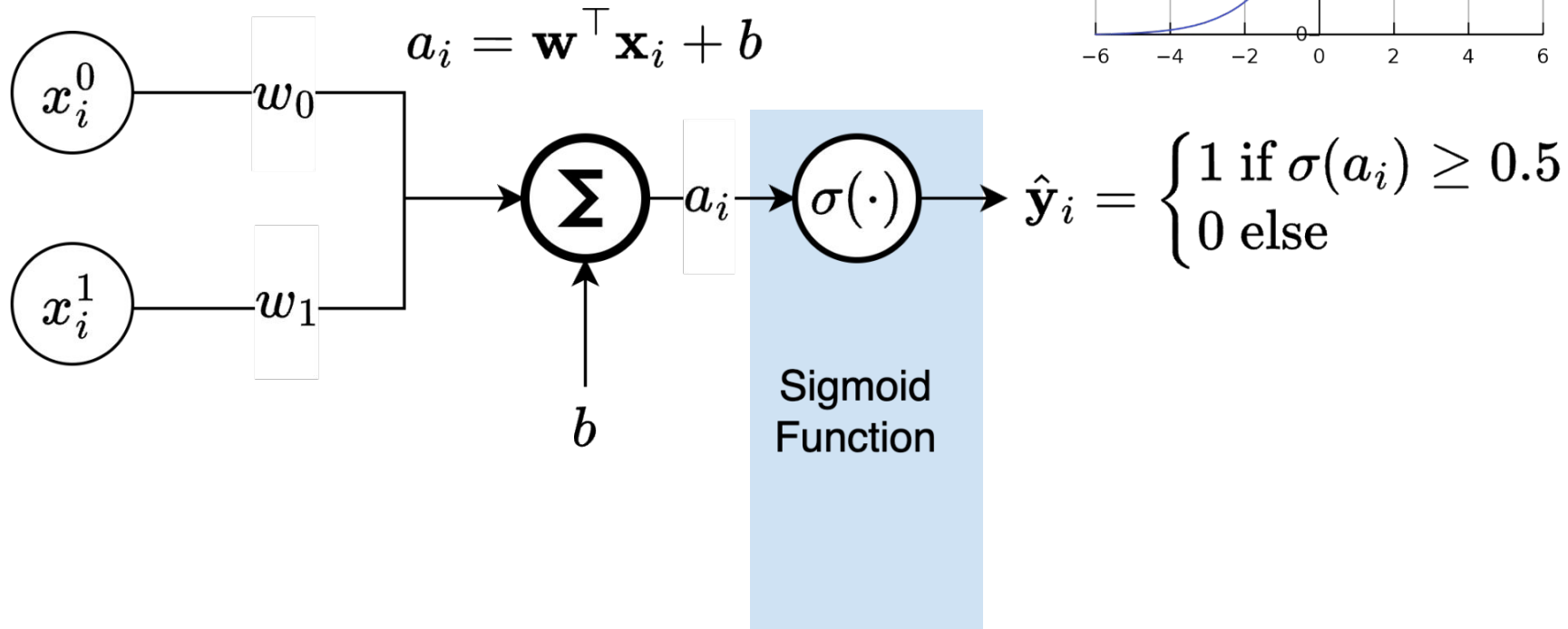
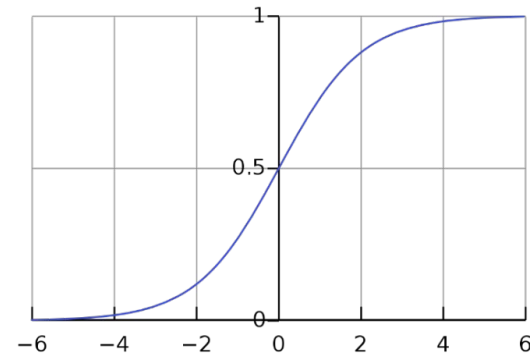


The “Soft” Perceptron



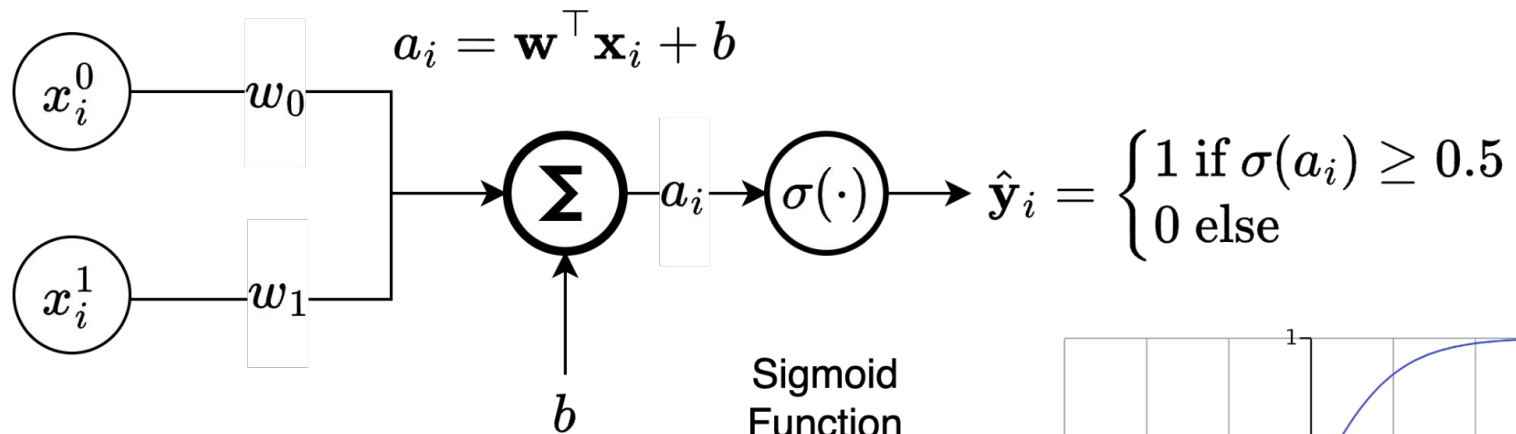
The “Soft” Perceptron

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



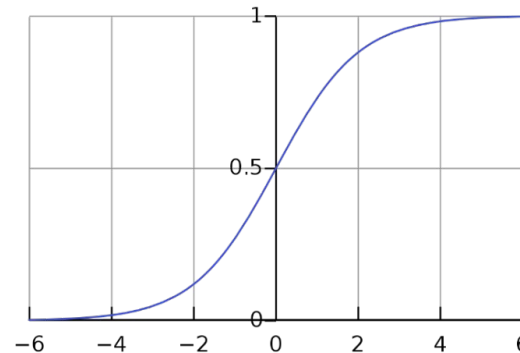
In other words... Logistic Regression

- A single-layer perceptron



Sigmoid
Function

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



Clean Up Bias Term $\mathbf{w}^\top \mathbf{x}_i + b$

Absorb bias term into feature vector:

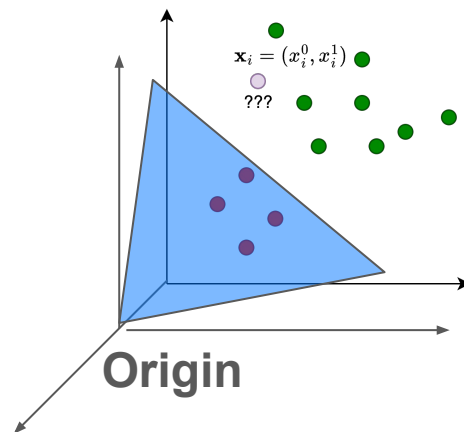
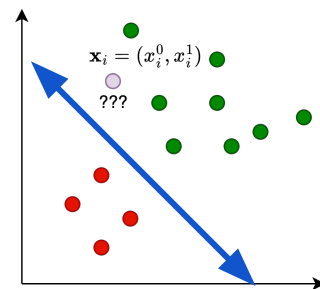
$$\mathbf{x}_i \text{ becomes } \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \text{ and } \mathbf{w} \text{ becomes } \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

We can see that:

$$\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}^\top \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} = \mathbf{w}^\top \mathbf{x}_i + b$$

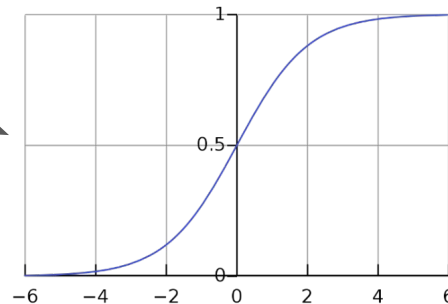
Can rewrite logistic regression as

$$\hat{y}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$$



Maximum Likelihood Estimation

$$\hat{\mathbf{y}}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$$



We want to find $\mathbf{w}$ to maximize the likelihood of the observed data $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{y}_i \in \{0, 1\}$

Start by writing $p(\mathbf{y}_i | \mathbf{x}_i)$ using $\hat{\mathbf{y}}_i$ and $\mathbf{y}_i$

Derive the “Log Loss”:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -[\mathbf{y}_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - \mathbf{y}_i) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i))]$$

Maximum Likelihood Estimation

Maximize the likelihood of the observed data $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{y}_i \in \{0, 1\}$:

$$p(\mathbf{y}_i | \mathbf{x}_i) = \hat{\mathbf{y}}_i^{\mathbf{y}_i} (1 - \hat{\mathbf{y}}_i)^{1 - \mathbf{y}_i}$$

Note that if $\mathbf{y}_i = 1$, then

$$p(\mathbf{y}_i | \mathbf{x}_i) = \hat{\mathbf{y}}_i$$

and if $\mathbf{y}_i = 0$, then

$$p(\mathbf{y}_i | \mathbf{x}_i) = 1 - \hat{\mathbf{y}}_i$$

Cross-Entropy Loss (aka Log Loss)

Maximizing the likelihood is equivalent to maximizing the log-likelihood:

$$\begin{aligned}\log p(\mathbf{y}_i | \mathbf{x}_i) &= \log[\hat{\mathbf{y}}_i^{\mathbf{y}_i} (1 - \hat{\mathbf{y}}_i)^{1 - \mathbf{y}_i}] \\ &= \mathbf{y}_i \log \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i)\end{aligned}$$

Add a negative sign to turn it into a loss, i.e. something to minimize:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\log p(\mathbf{y}_i | \mathbf{x}_i) = -[\mathbf{y}_i \log \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i)]$$

We can plug in our definition of $\hat{\mathbf{y}}_i = \sigma(\mathbf{w}^\top \mathbf{x}_i + b)$:

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -[\mathbf{y}_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i + b) + (1 - \mathbf{y}_i) \log(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i + b))]$$

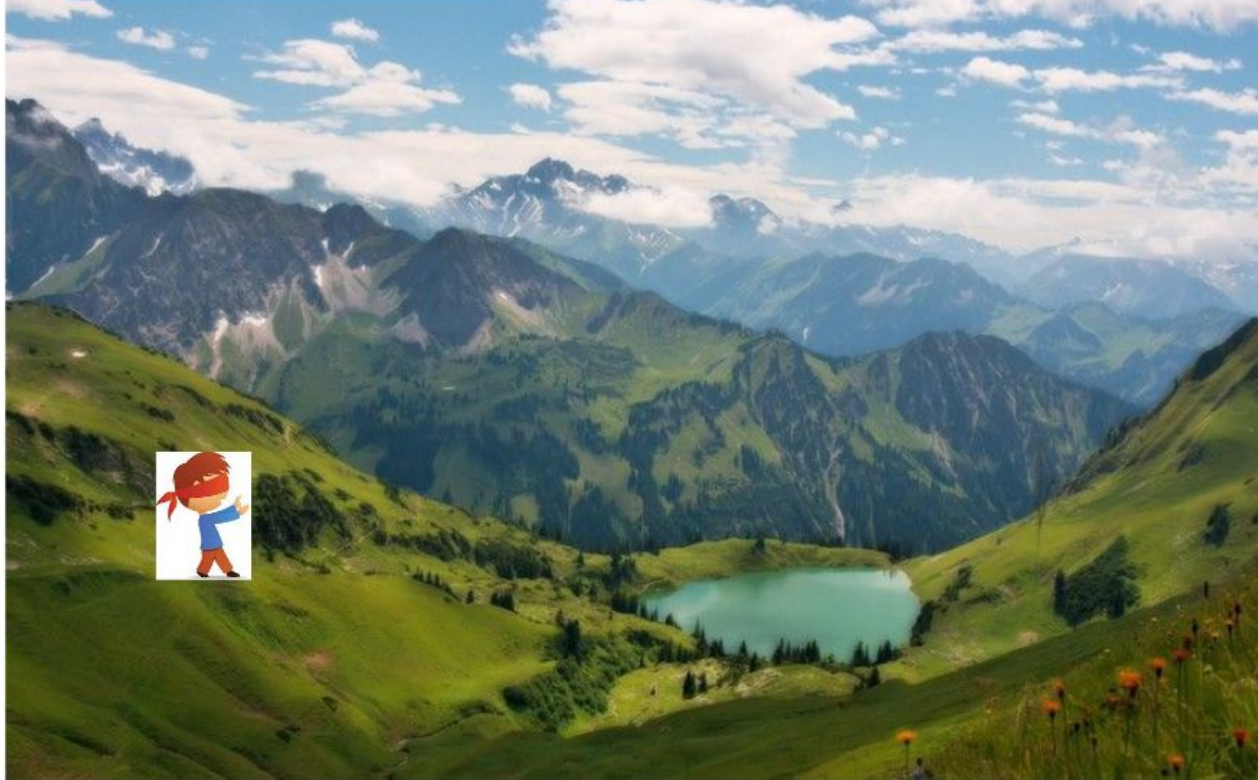
Our Goal: Minimize the Loss

Given some training dataset:

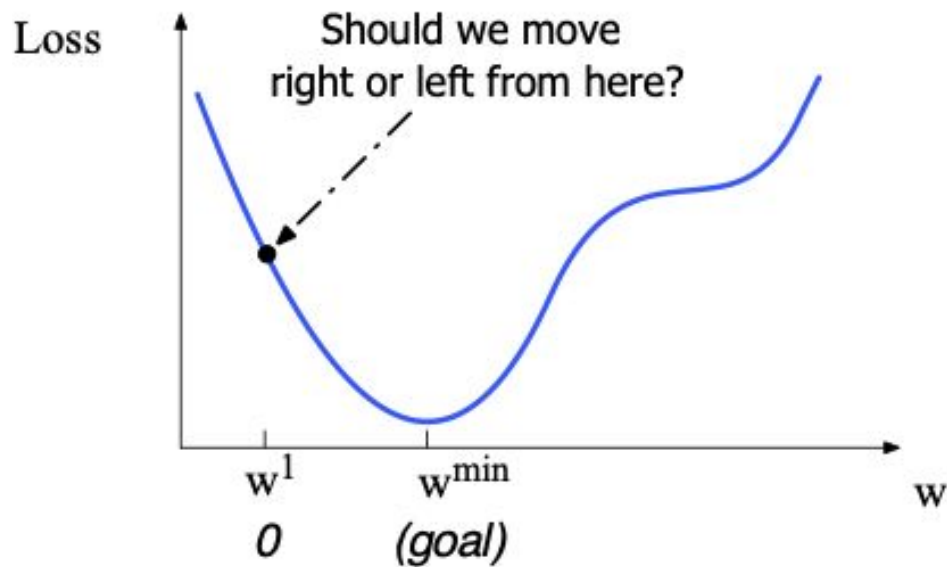
$$\mathcal{D}_{\text{TR}} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=0}^n$$

$$\begin{aligned}\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}_{\text{TR}}) &= \frac{1}{n} \sum_i^n \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) \\ &= \frac{1}{n} \sum_i^n \ell(\sigma(\mathbf{w}^\top \mathbf{x}_i), \mathbf{y}_i)\end{aligned}$$

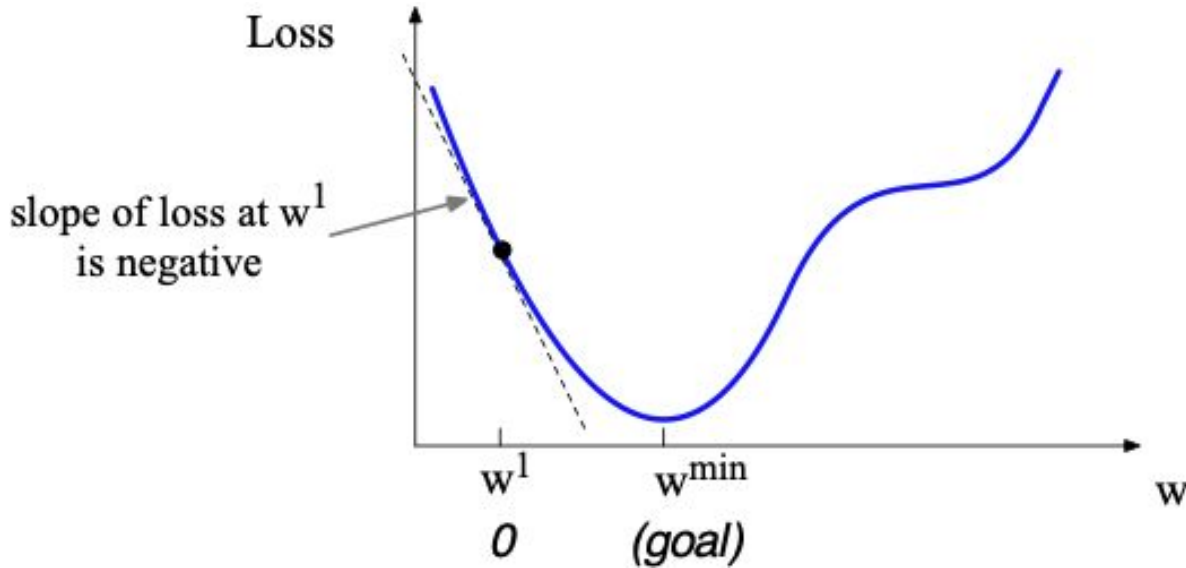
Gradient Descent



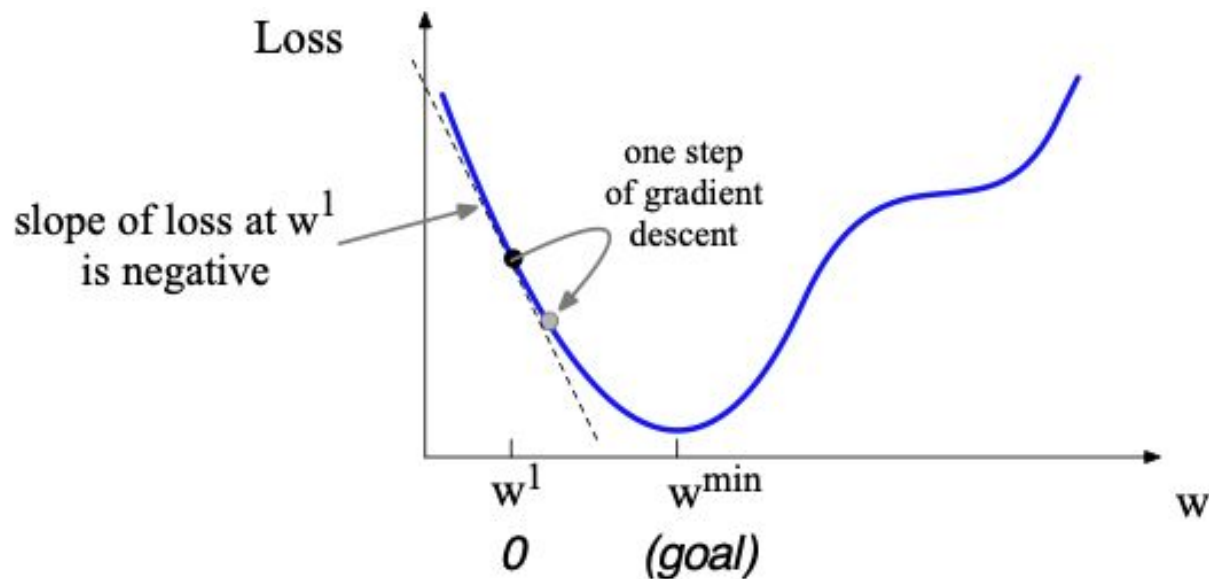
Visualize Gradient Descent in 1-D



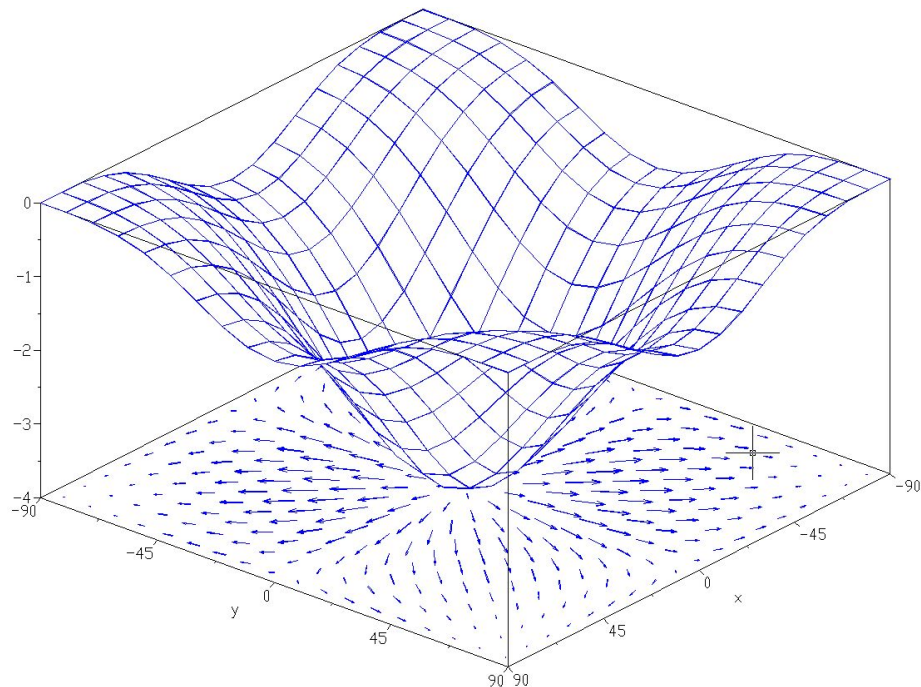
Visualize Gradient Descent in 1-D



Visualize Gradient Descent in 1-D



Gradients



$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}_{\text{TR}}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w^{(0)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \\ \frac{\partial \mathcal{L}}{\partial w^{(1)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w^{(m)}}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \end{bmatrix}, \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}_{\text{TR}}) \in \mathbb{R}^m$$

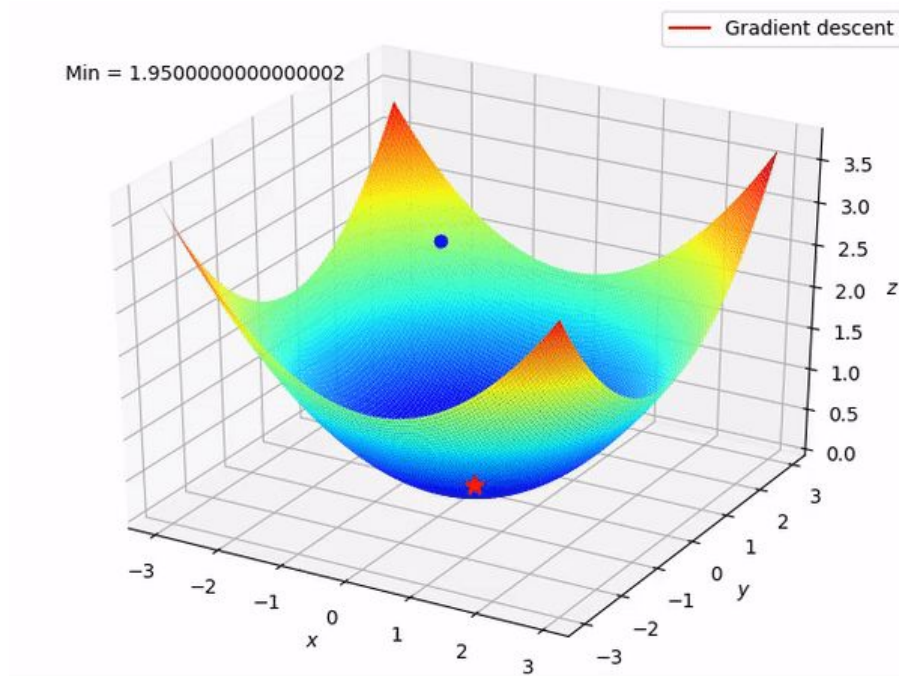
Gradient Descent:

- Find the gradient at current point
- Move in **opposite** direction with learning rate α

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}_t} \mathcal{L}(\mathbf{w}_t; \mathcal{D}_{\text{TR}})$$

Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}_t} \mathcal{L}(\mathbf{w}_t; \mathcal{D}_{\text{TR}})$$



What are key components in ML?

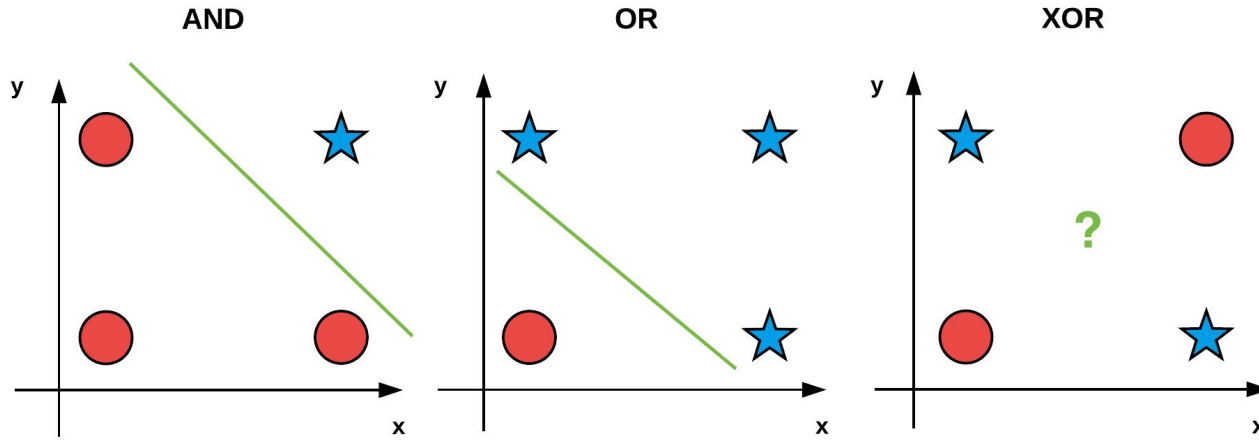
- Training data
- Model Class / Hypothesis space
- Loss function
- Optimization

Demo: Logistic Regression

- [Tensorflow Playground](#)

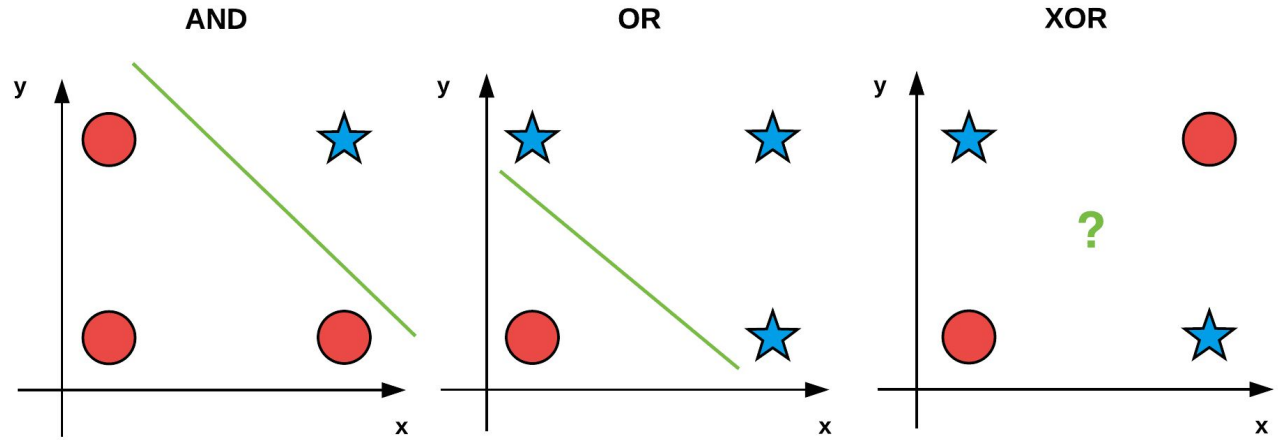
Demo: The XOR Problem

- [Tensorflow Playground](#)



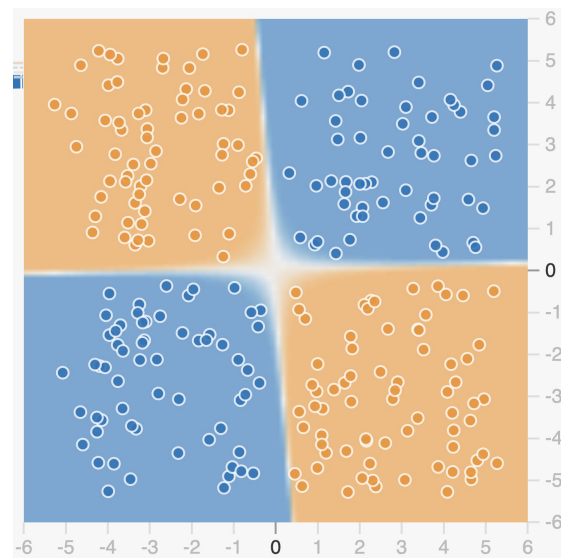
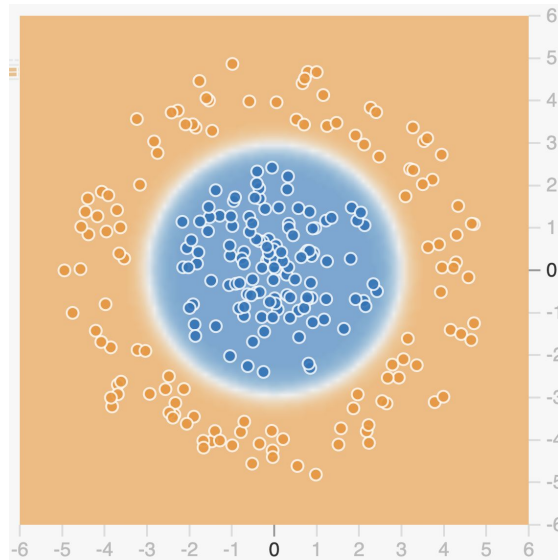
The XOR Problem

- Perceptron can't learn the XOR function
 - Simple logical operation
- Data is not linearly separable



Discuss: What are some ways to handle data that is not linearly separable?

Without deep learning!



Possible Solutions

- Feature engineering
 - Construct a feature space where the data is linearly separable
- Kernel methods
 - Implicitly project the data into a higher-dimensional space where it is linearly separable
- Non-linear classifiers
 - E.g. Nearest neighbor, decision tree algorithms

Cornell Bowers C-IS

Demo: Feature Engineering

[Tensorflow Playground](#)

Feature Engineering



input image

classification

“dog”



input image

classification

“cat”

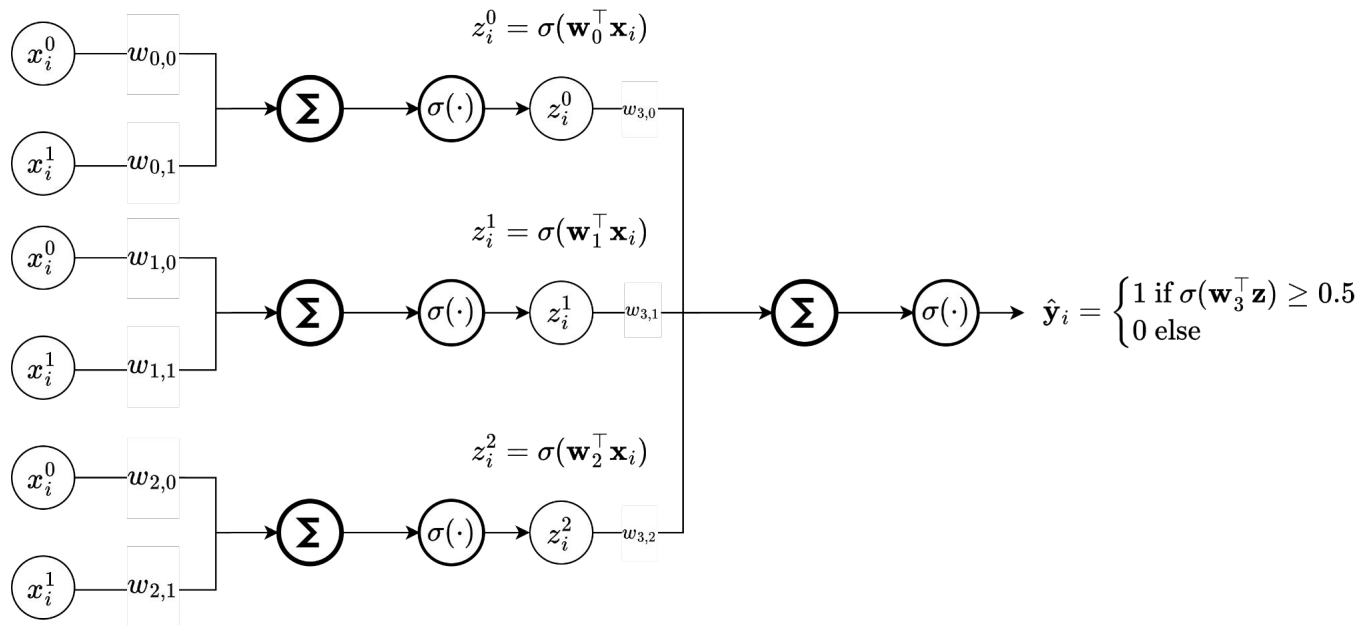
Agenda

- Perceptron
- Logistic Regression
- Gradient Descent
- **Multi-Layer Perceptrons (MLPs)**
- Backpropagation

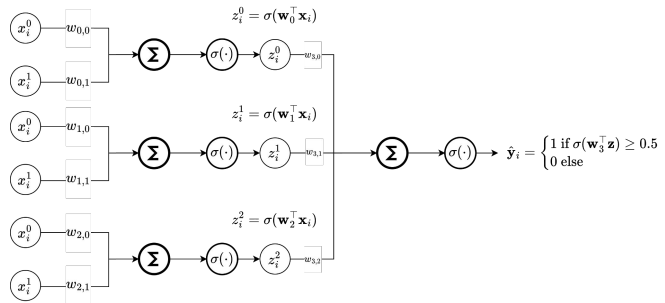
Multi-Layer Perceptron (MLP)

- Compose multiple perceptrons to **learn** intermediate features

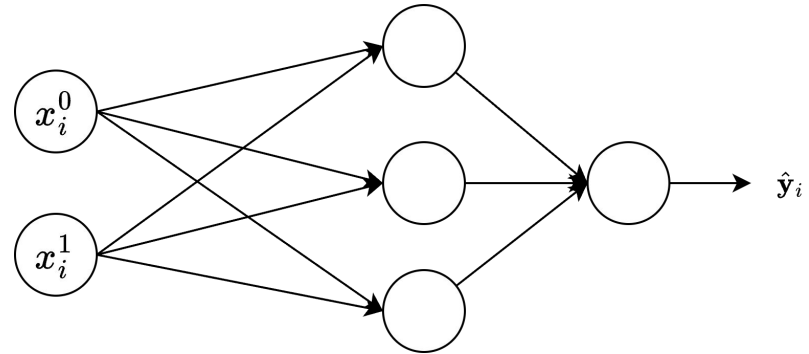
An MLP with 1 hidden layer with 3 hidden units



A Simplified MLP Diagram

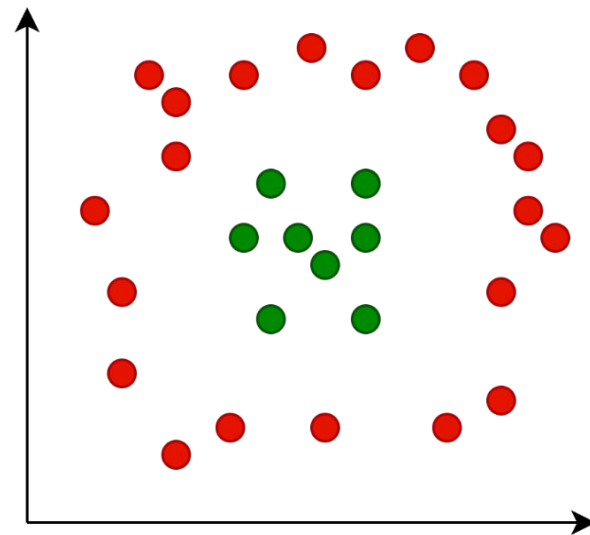
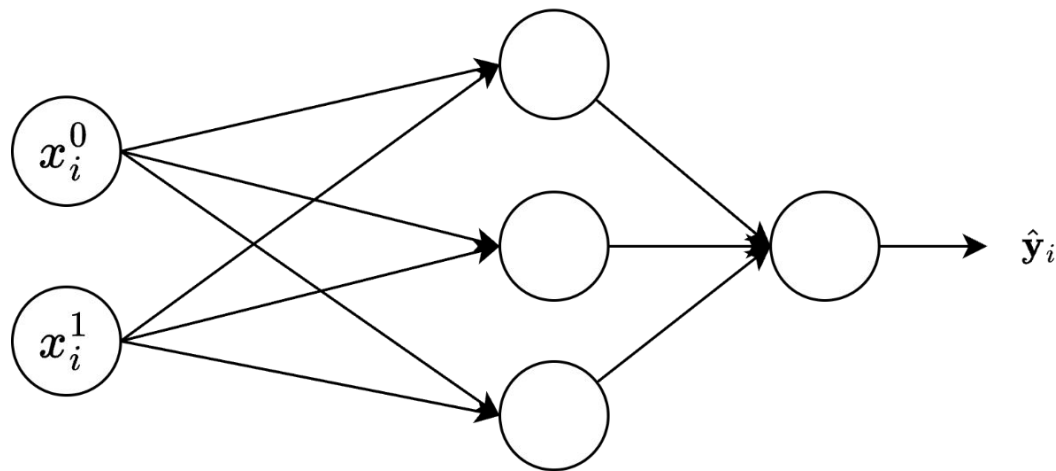


1 Hidden Layer,
3 Hidden Units



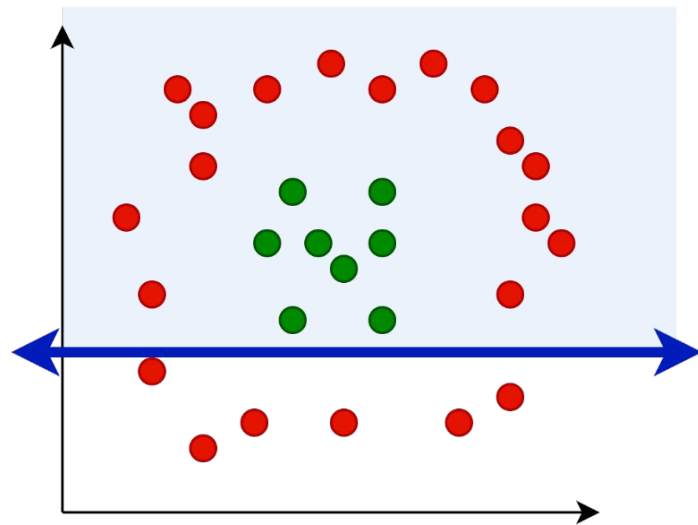
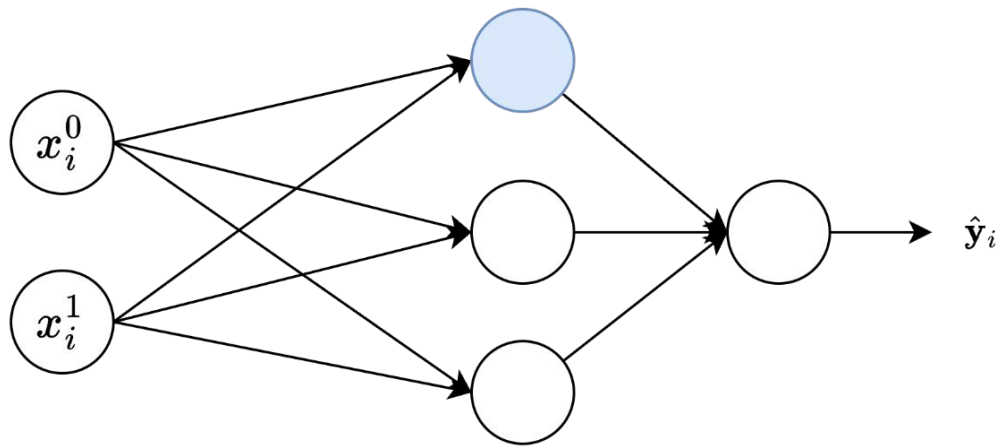
Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



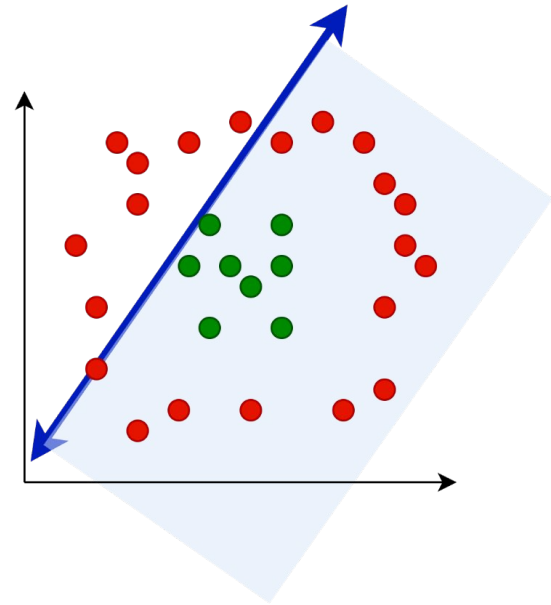
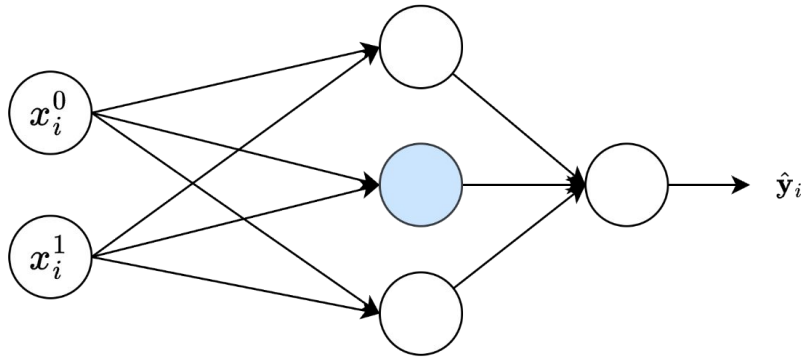
Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



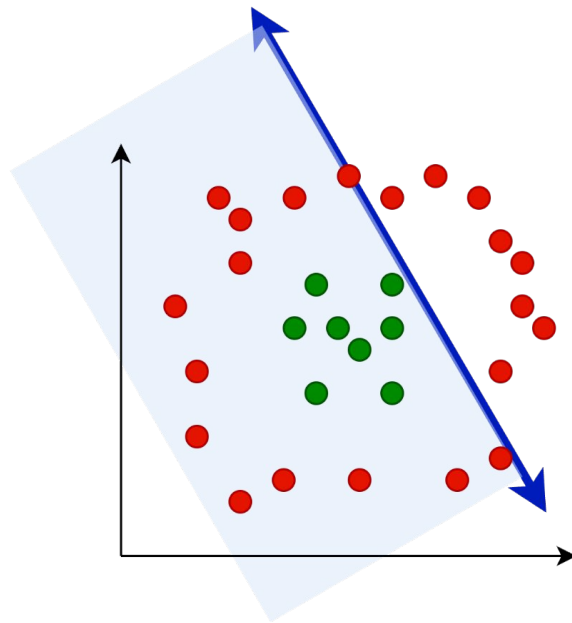
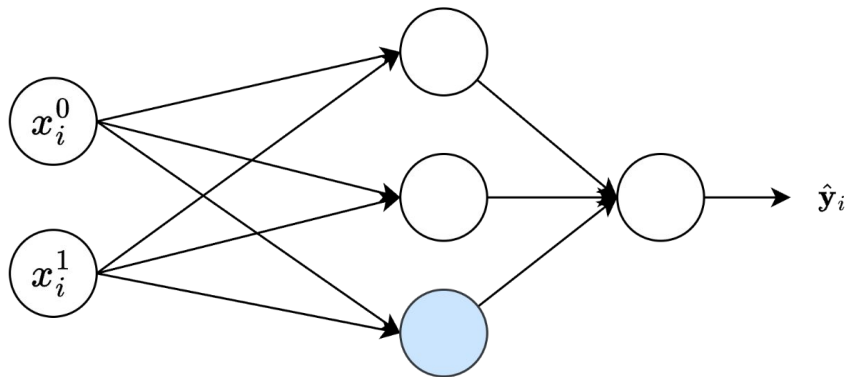
Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



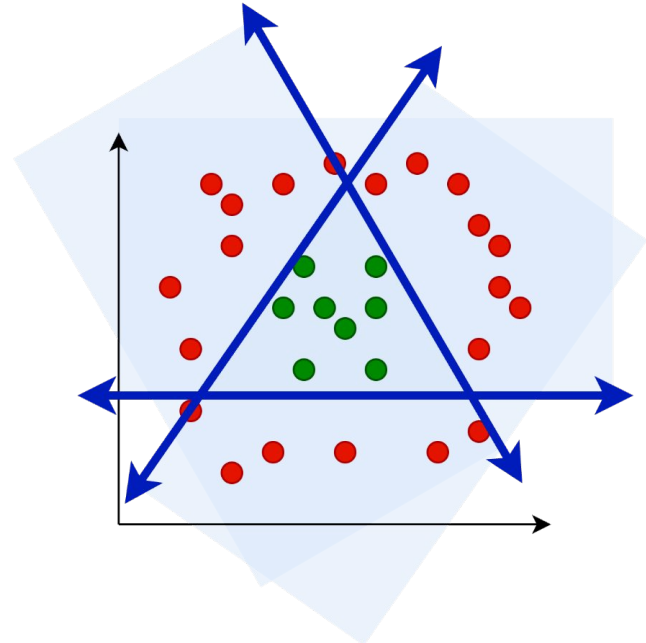
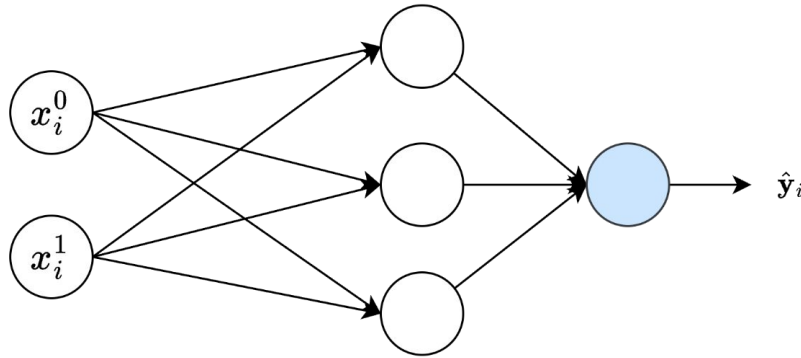
Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



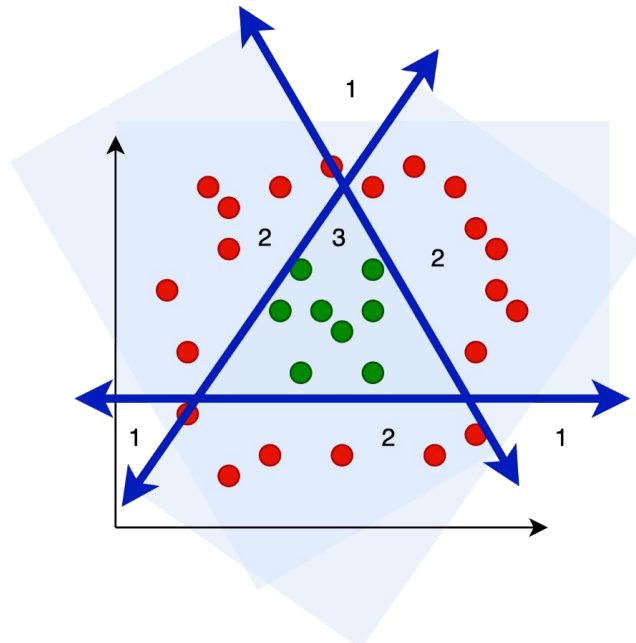
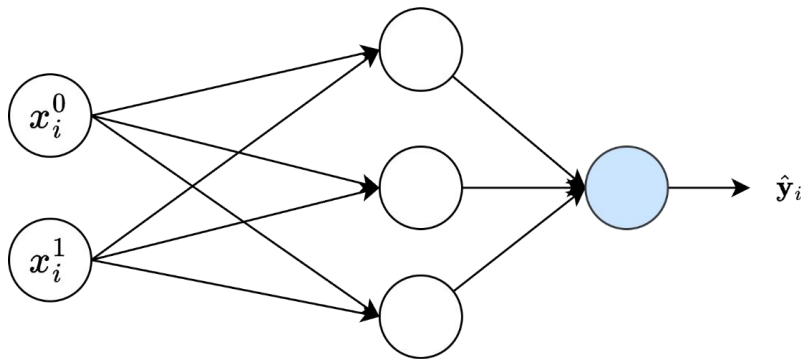
Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



Complex Decision Boundaries

- What does this extra layer give us?
 - Can compose multiple linear classifiers



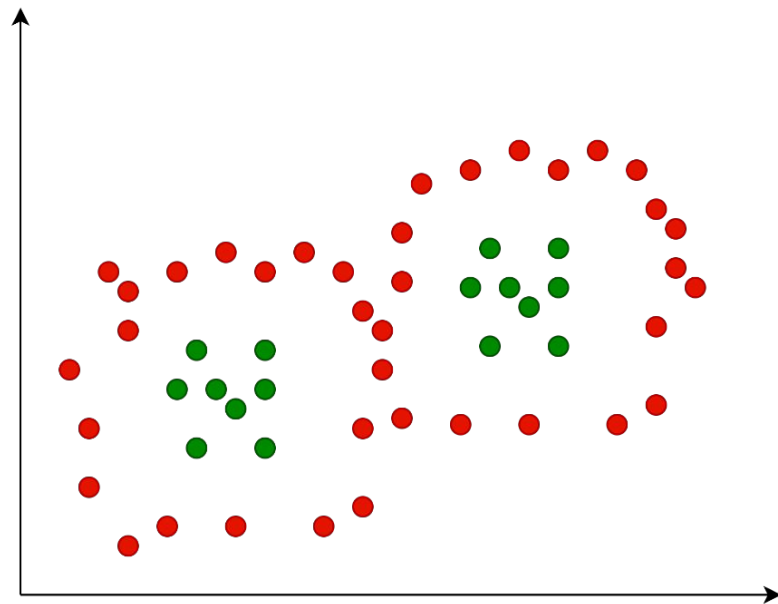
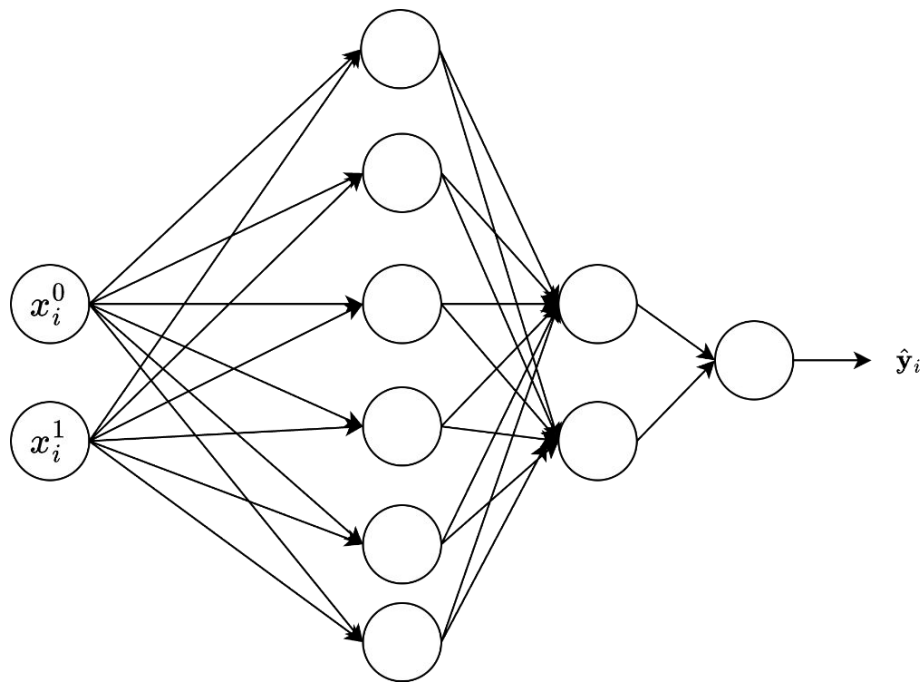
Cornell Bowers C-IS

MLP Demo (1 Hidden Layer)

[Tensorflow Playground](#)

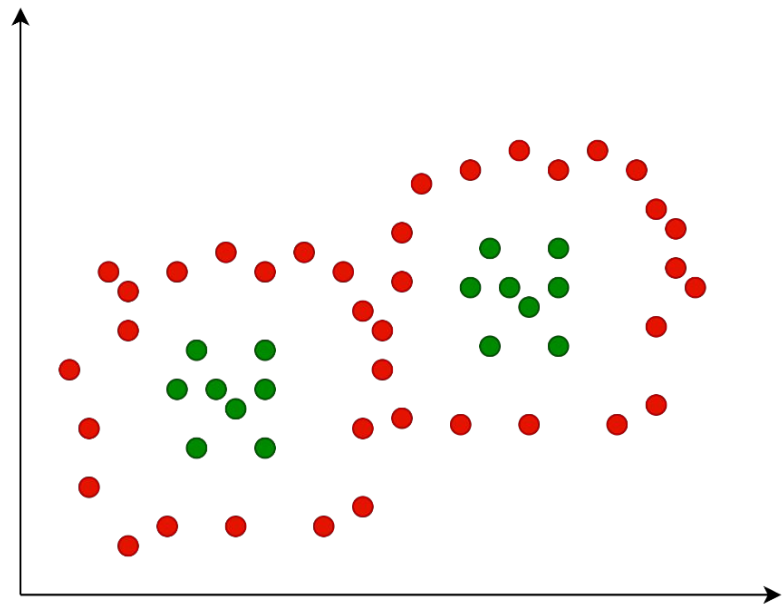
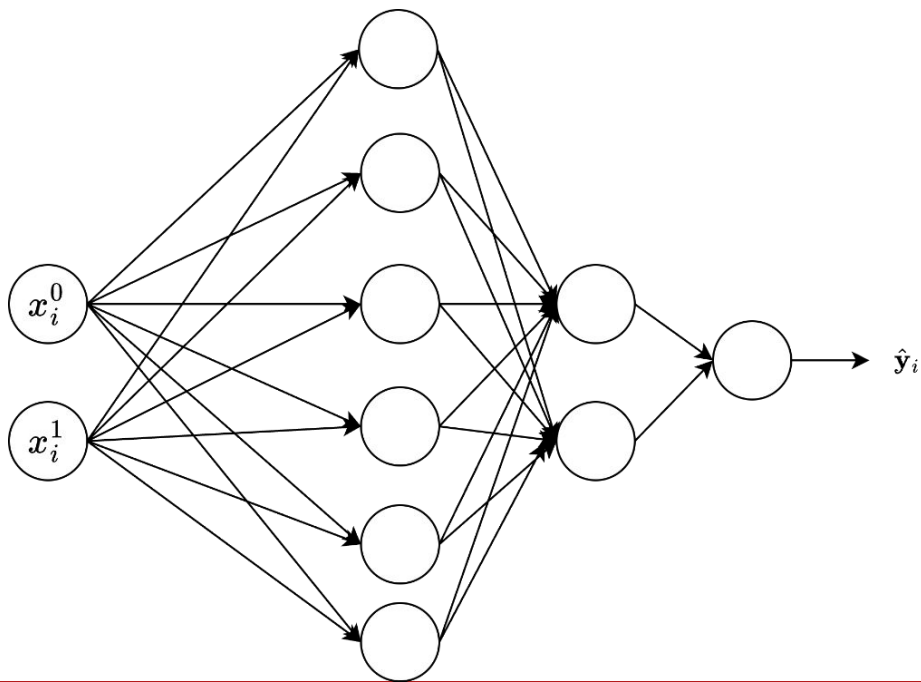
Increasing Depth

Discuss: How to construct the decision boundary?

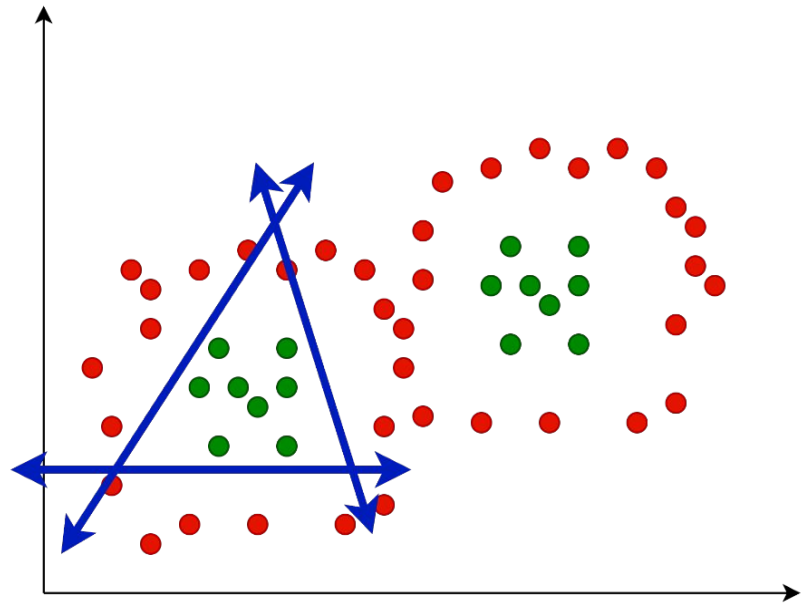
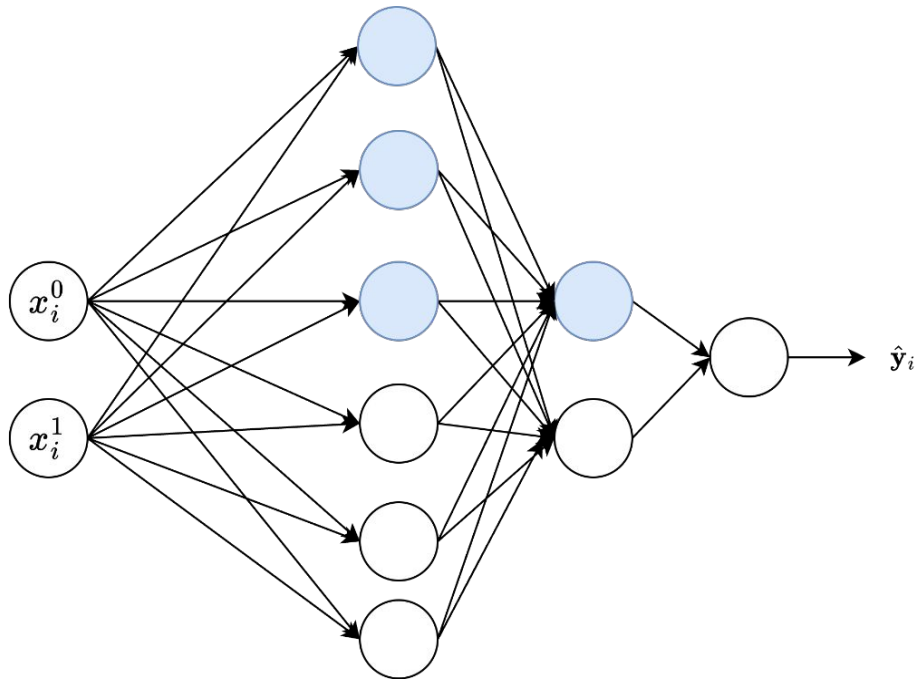


Increasing Depth

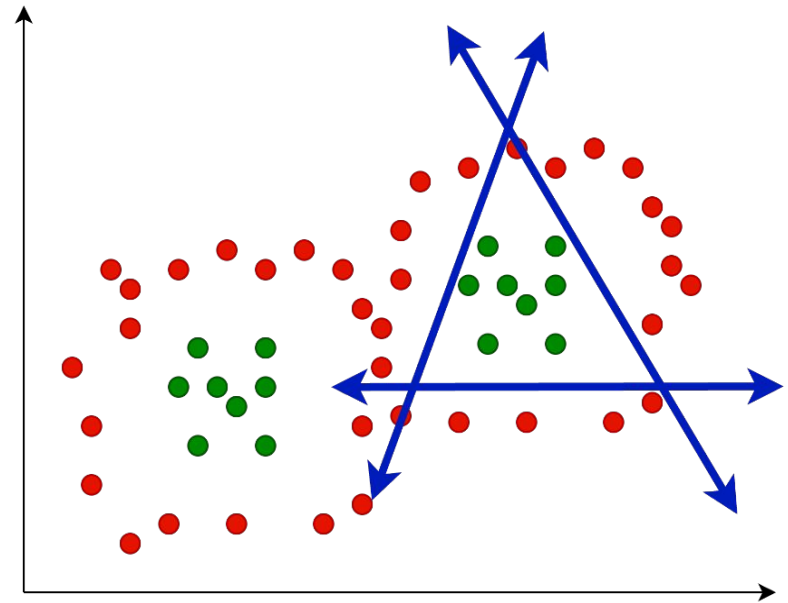
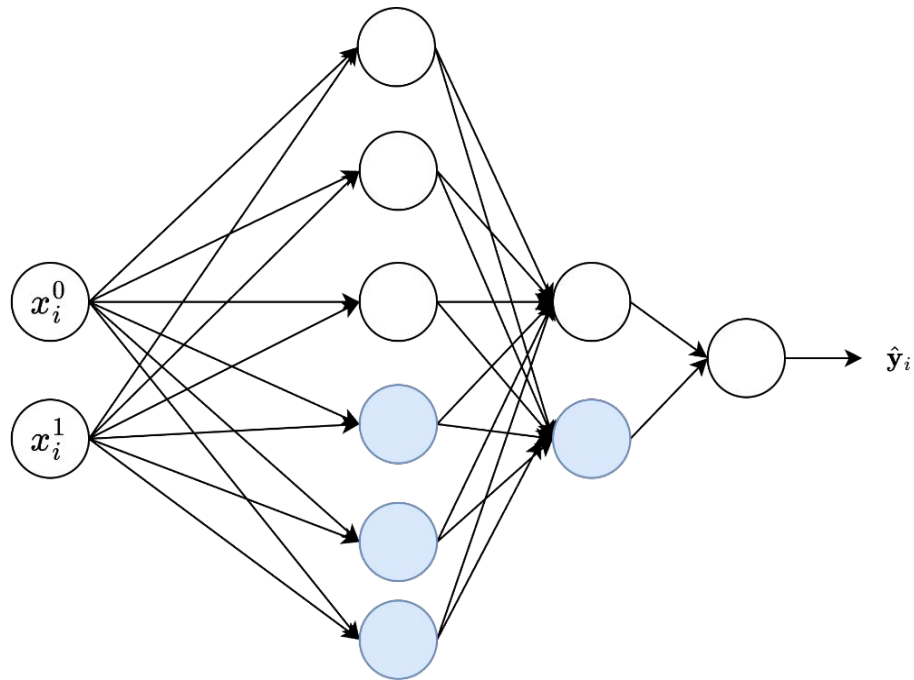
- MLP with 1 hidden layer composes linear classifiers
- MLP with 2 hidden layers can compose polygon classifiers



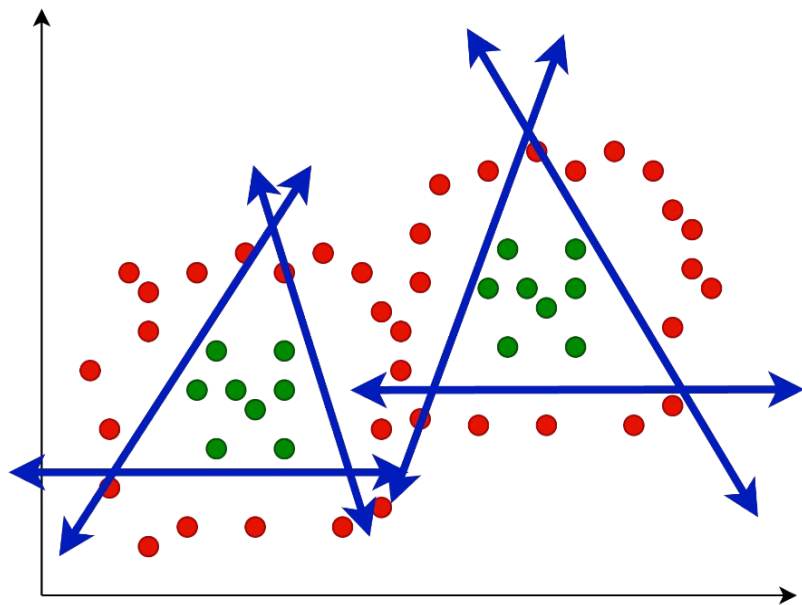
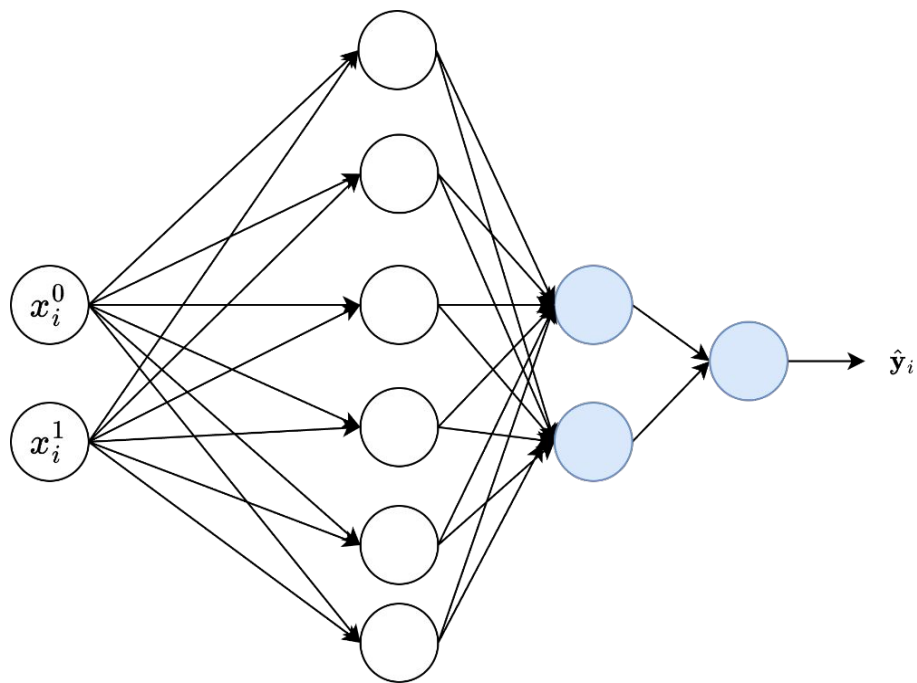
Increasing Depth



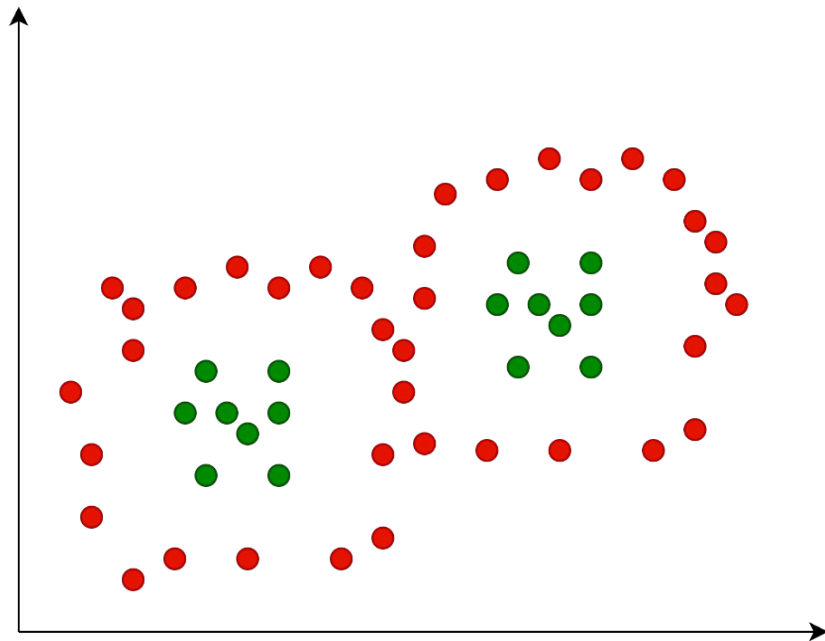
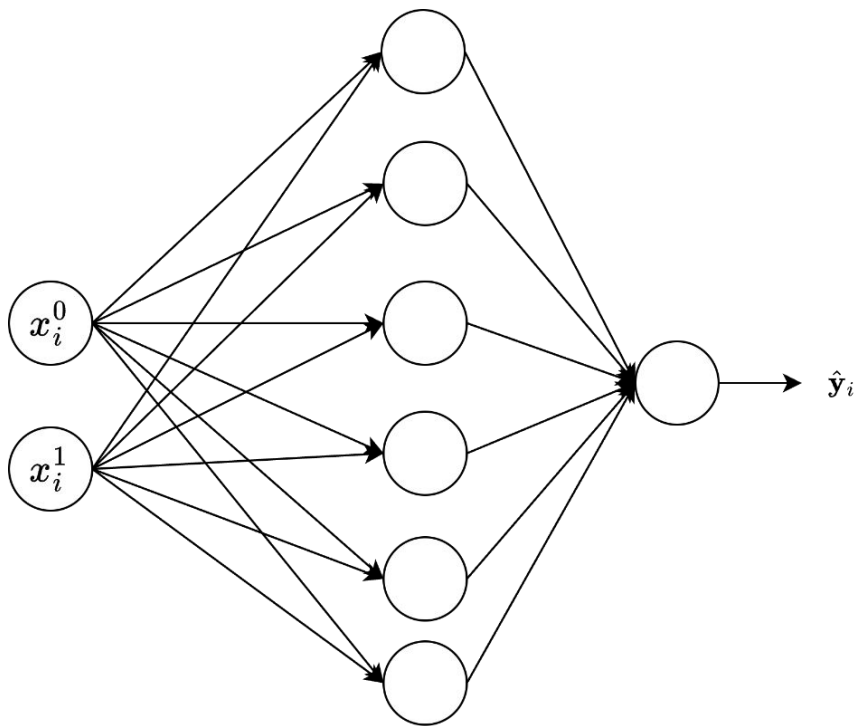
Increasing Depth



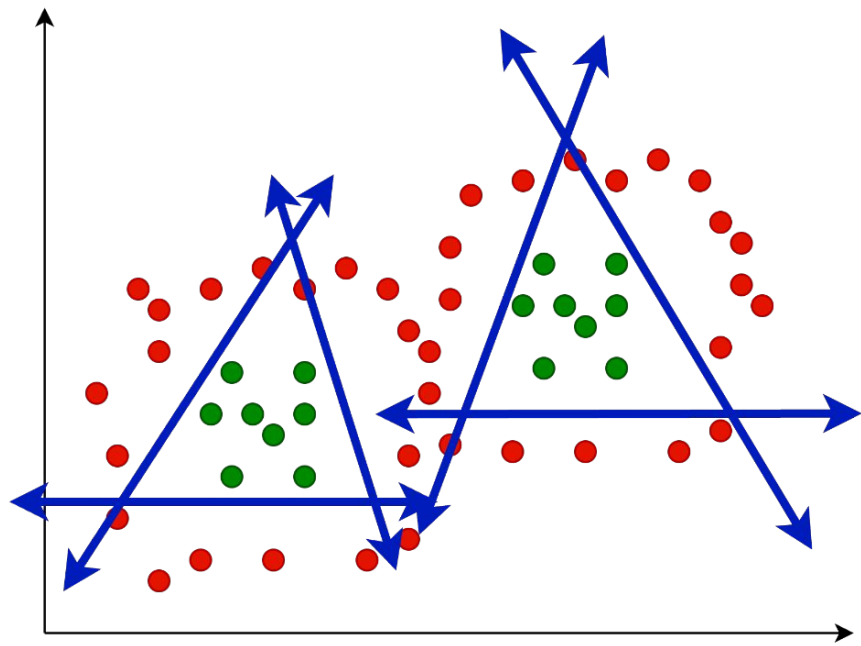
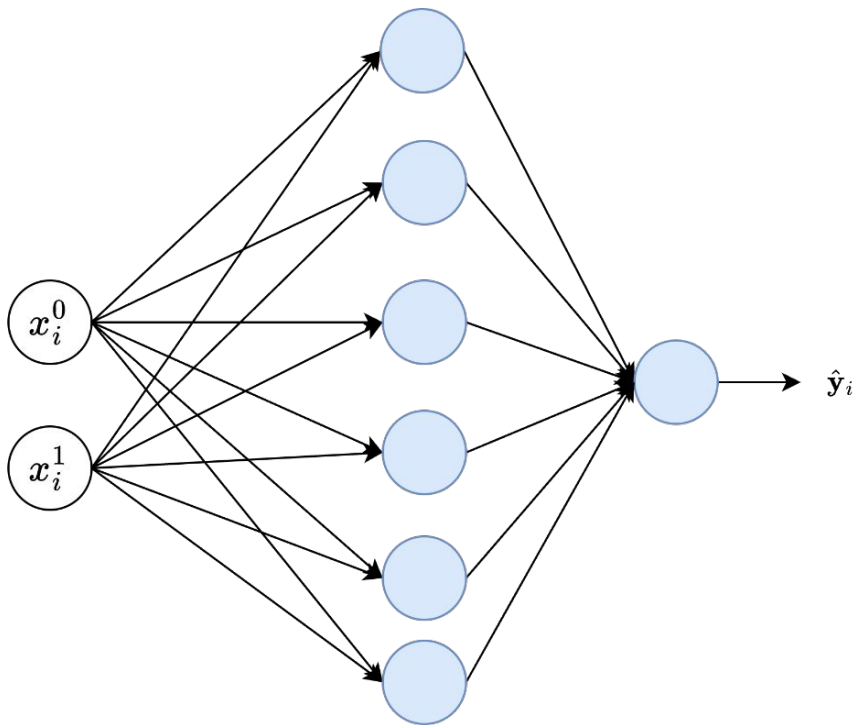
Increasing Depth



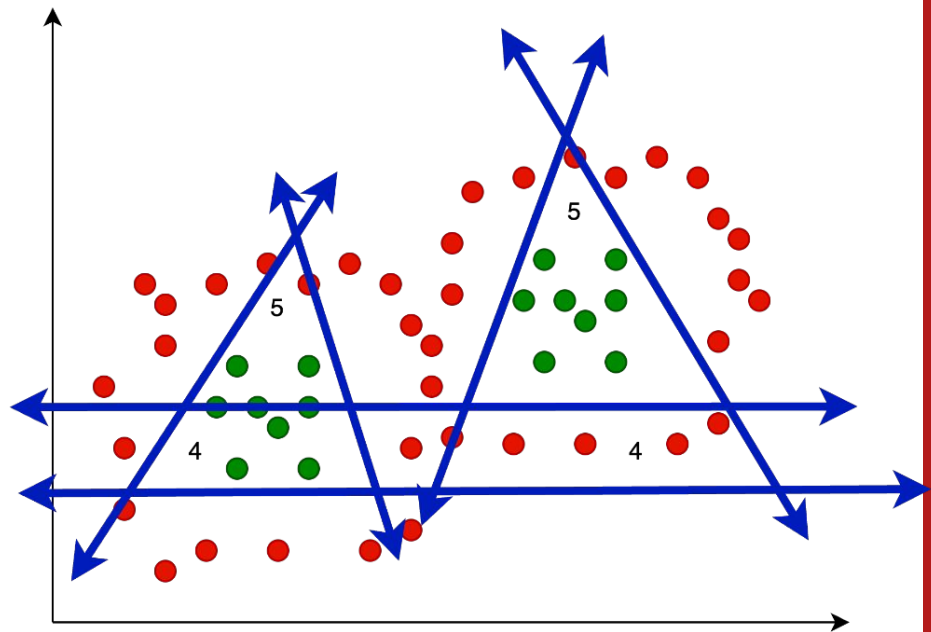
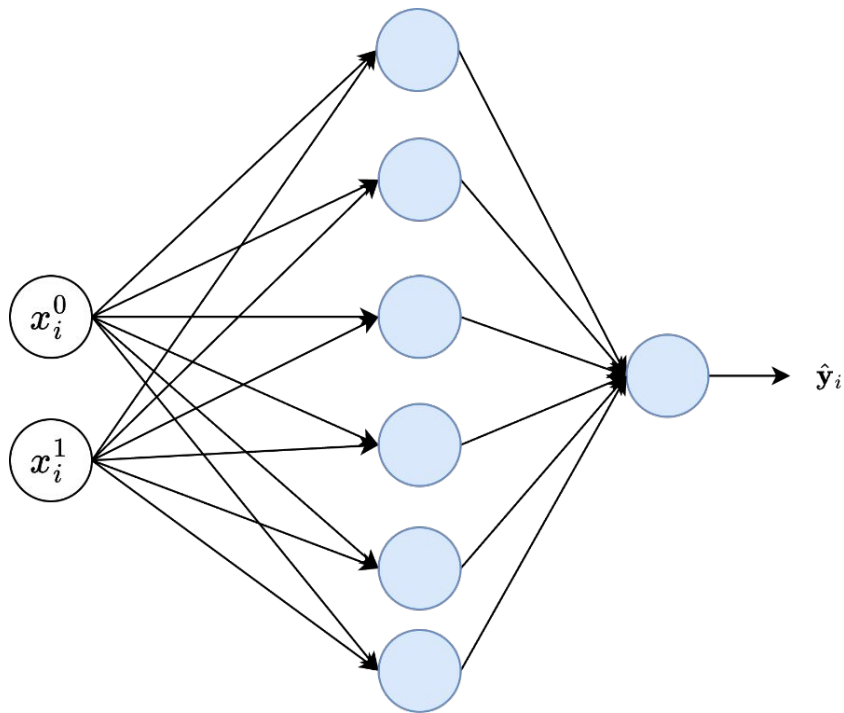
Discuss: What about just one layer?



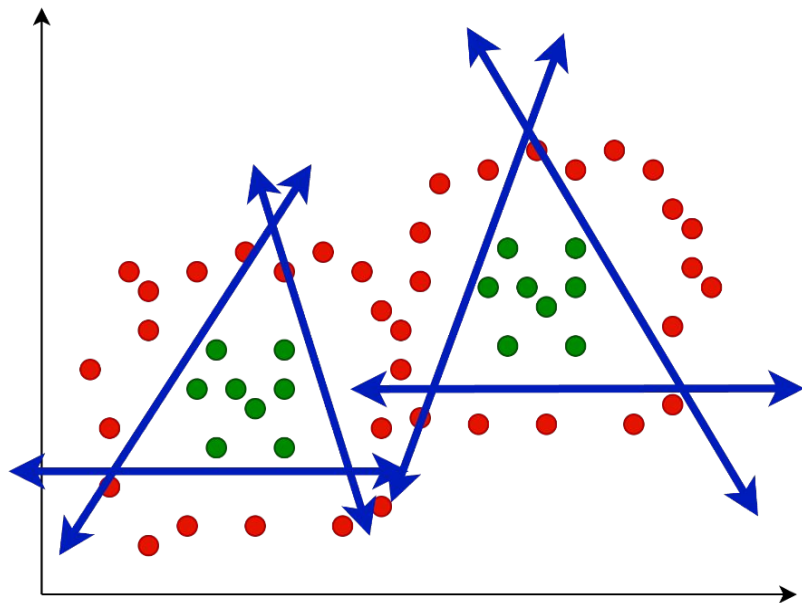
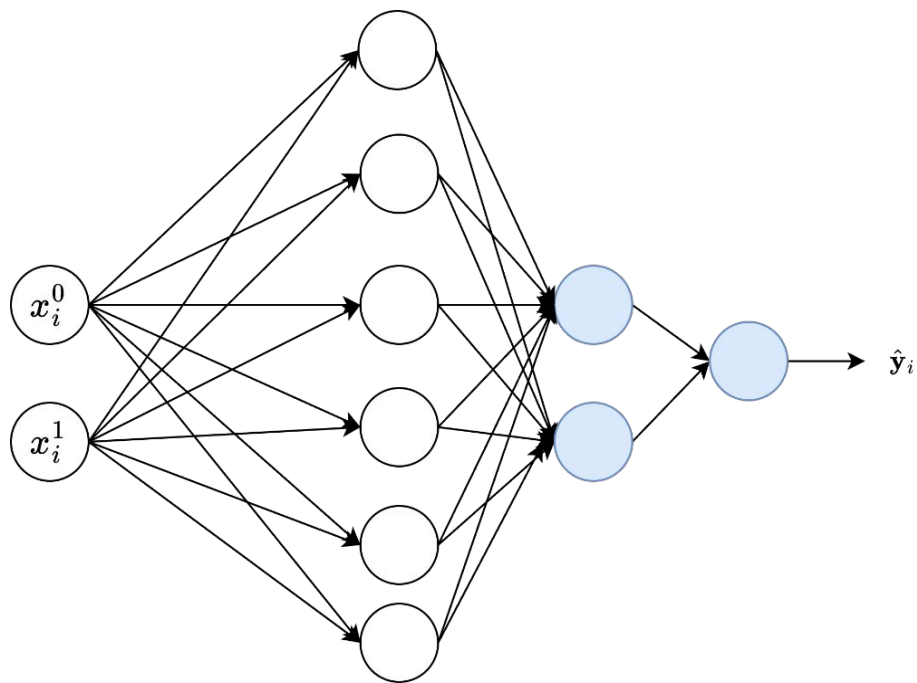
What about just one layer?



What about just one layer?

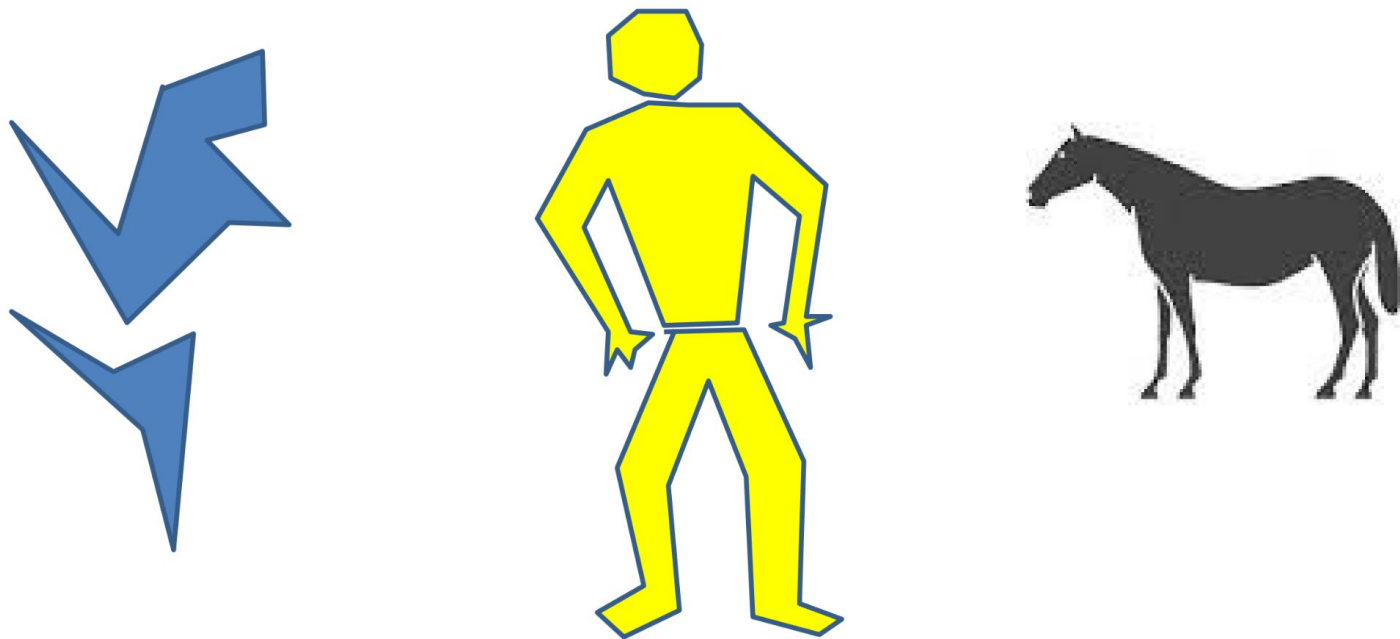


Increasing Depth



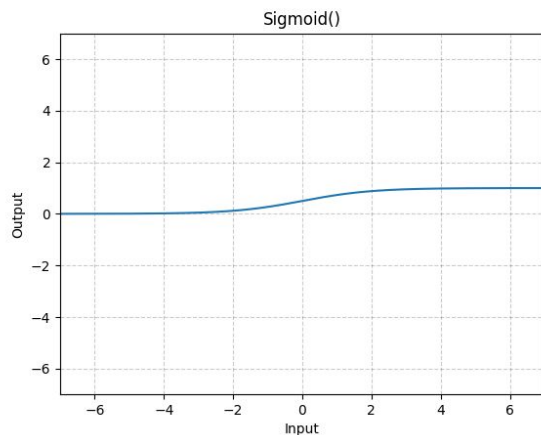
Complex Decision Boundaries

- Can compose *arbitrarily* complex decision boundaries



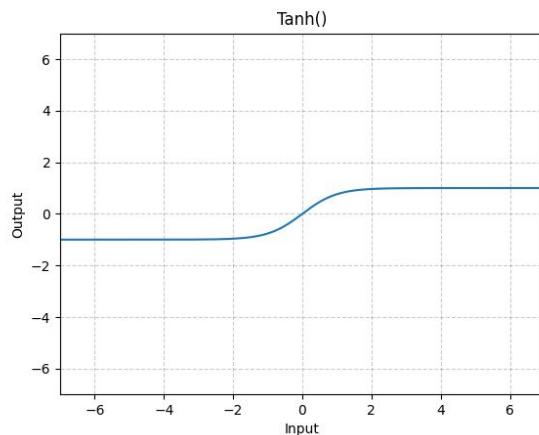
Activation Functions

- Can replace the sigmoid with other nonlinear functions
 - Still universal approximators!



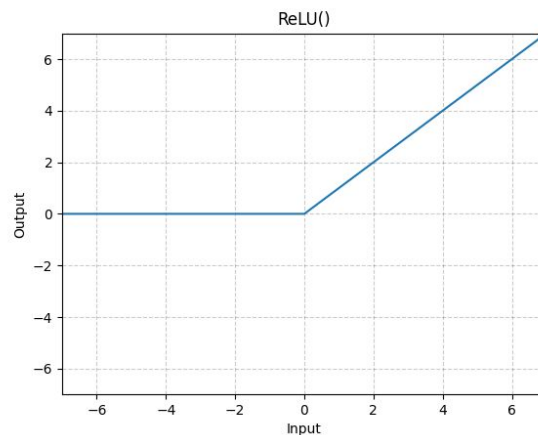
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Squash between 0 and 1



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

Squash between -1 and 1



$$\text{ReLU}(x) = \max(0, x)$$

Threshold at 0

Cornell Bowers C-IS

MLP Demo (3 Hidden Layers)

[Tensorflow Playground](#)

How to learn MLP weights?

Gradient descent!

Calculus Review: The Chain Rule

Lagrange's Notation: If $h(x) = f(g(x))$, then $h' = f'(g(x))g'(x)$

Leibniz's Notation: If $z = h(y), y = g(x)$, then $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

Calculus Review: The Chain Rule

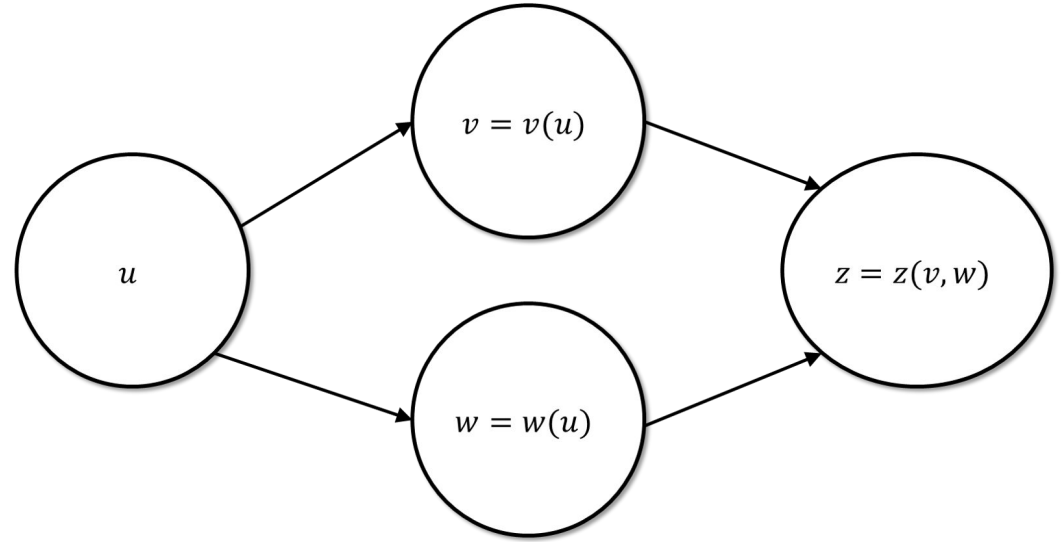
Lagrange's Notation: If $h(x) = f(g(x))$, then $h' = f'(g(x))g'(x)$

Leibniz's Notation: If $z = h(y), y = g(x)$, then $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

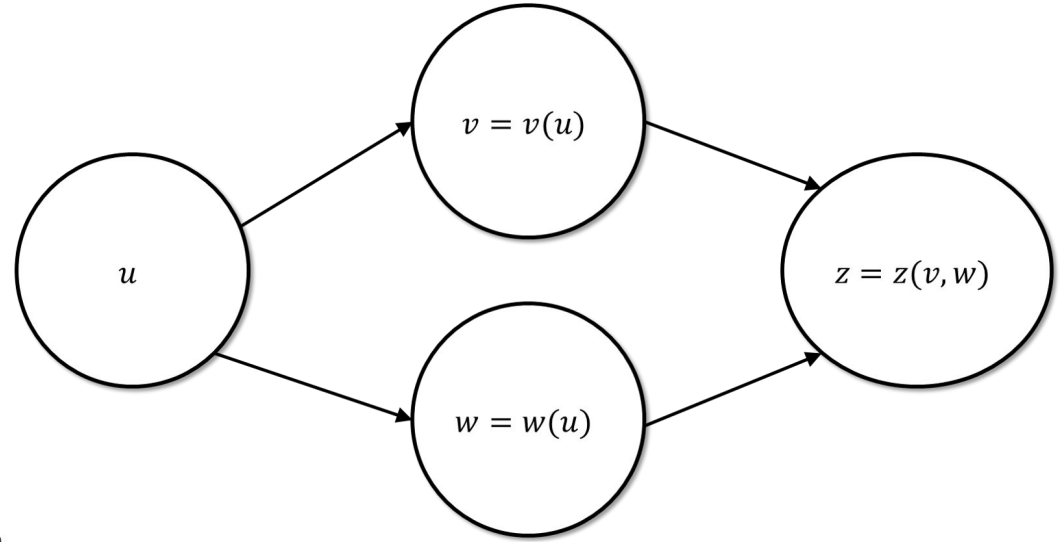
Example: If $z = \ln(y), y = x^2$, then

$$\begin{aligned}\frac{dz}{dx} &= \frac{dz}{dy} \frac{dy}{dx} \\ &= \left(\frac{1}{y}\right)(2x) = \left(\frac{1}{x^2}\right)(2x) = \frac{2}{x}\end{aligned}$$

Multivariate Chain Rule



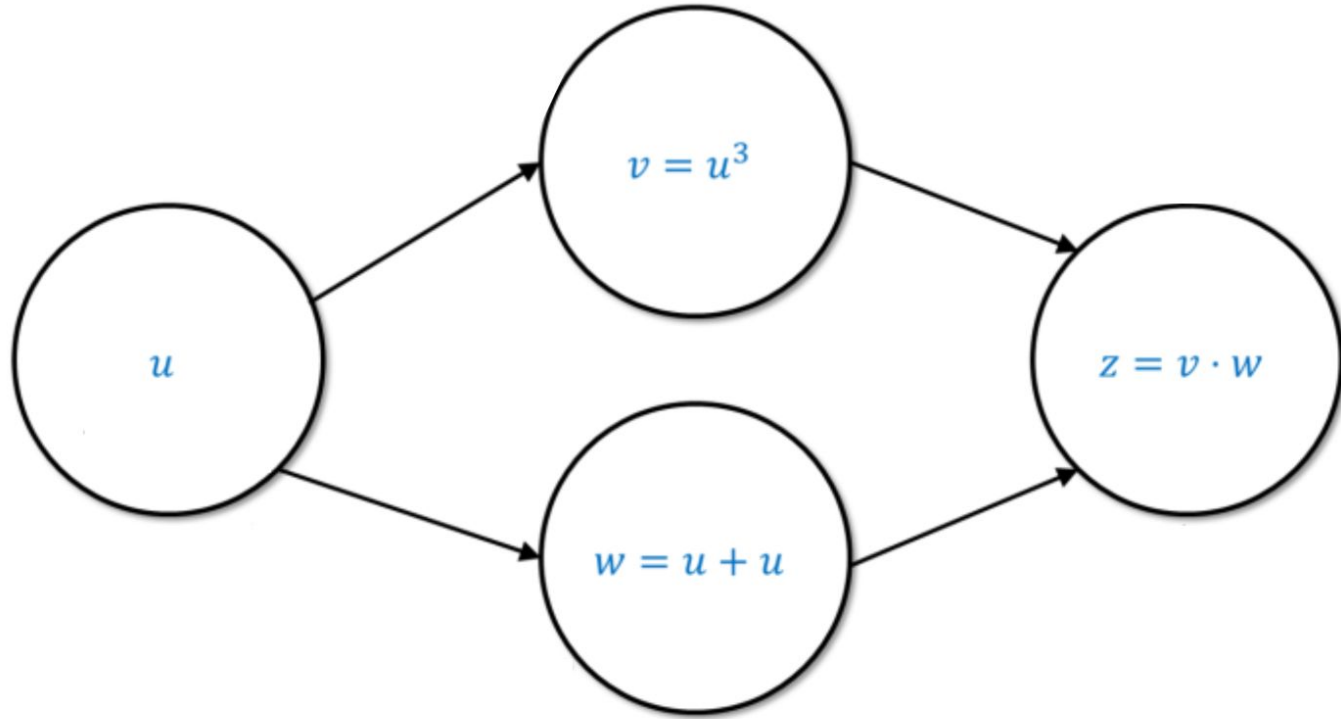
Multivariate Chain Rule



If $f(u)$ is $z = f(v(u), w(u))$, then

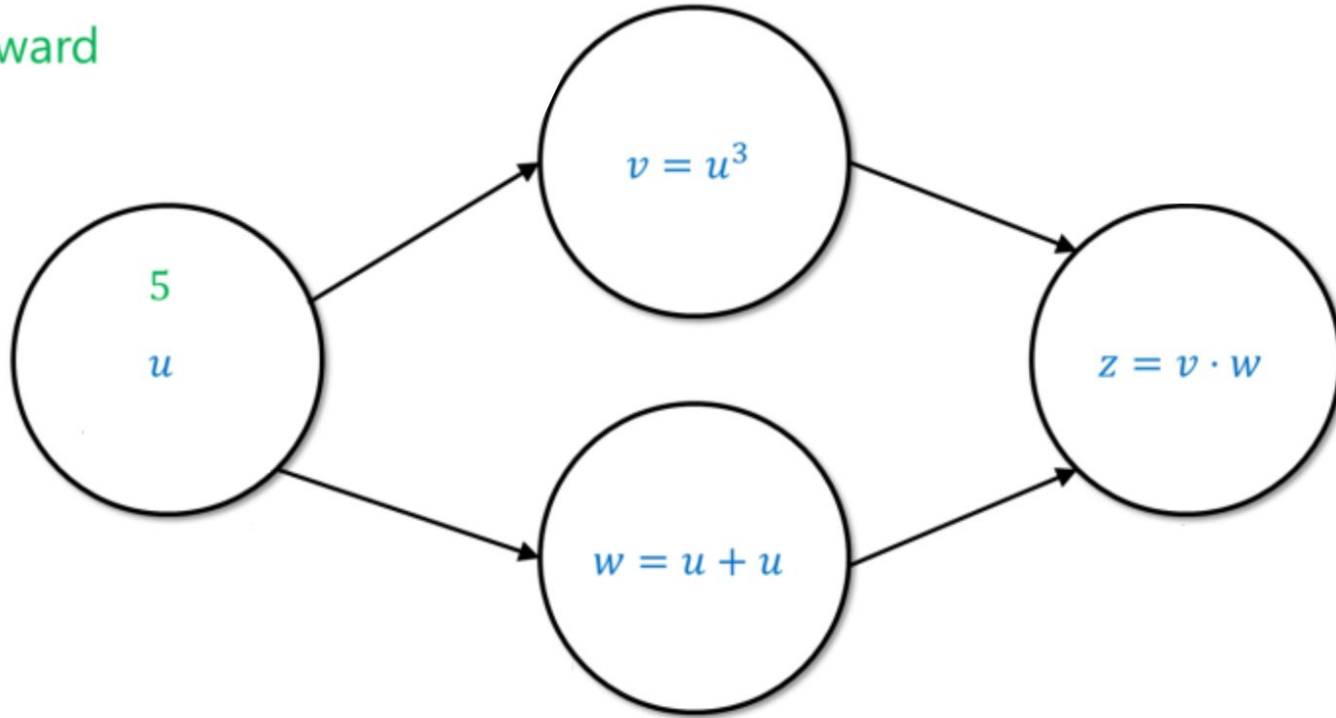
$$\frac{\partial f}{\partial u} = \left(\frac{\partial v}{\partial u} \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \frac{\partial z}{\partial w} \right)$$

Backpropagation- An Example



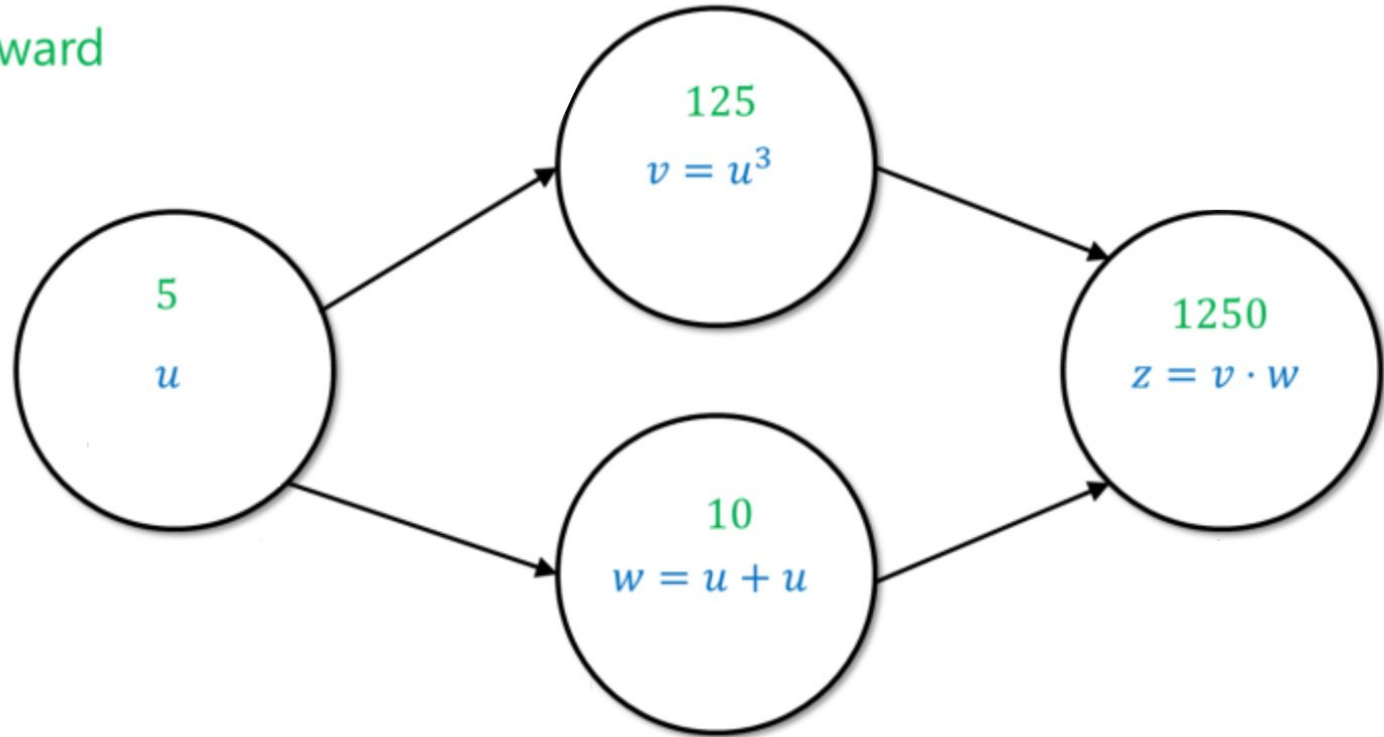
Backpropagation- An Example

Forward



Backpropagation- An Example

Forward

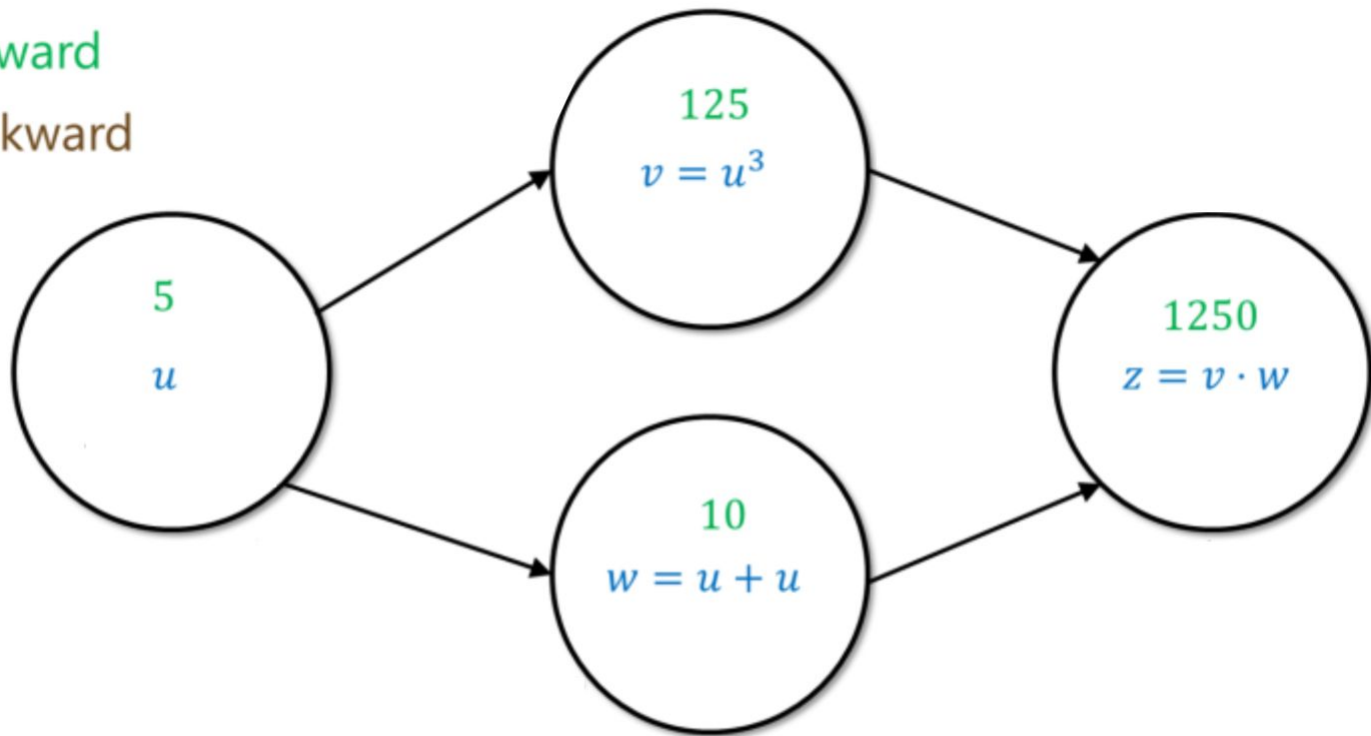


Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w} \right)$$

Forward

Backward

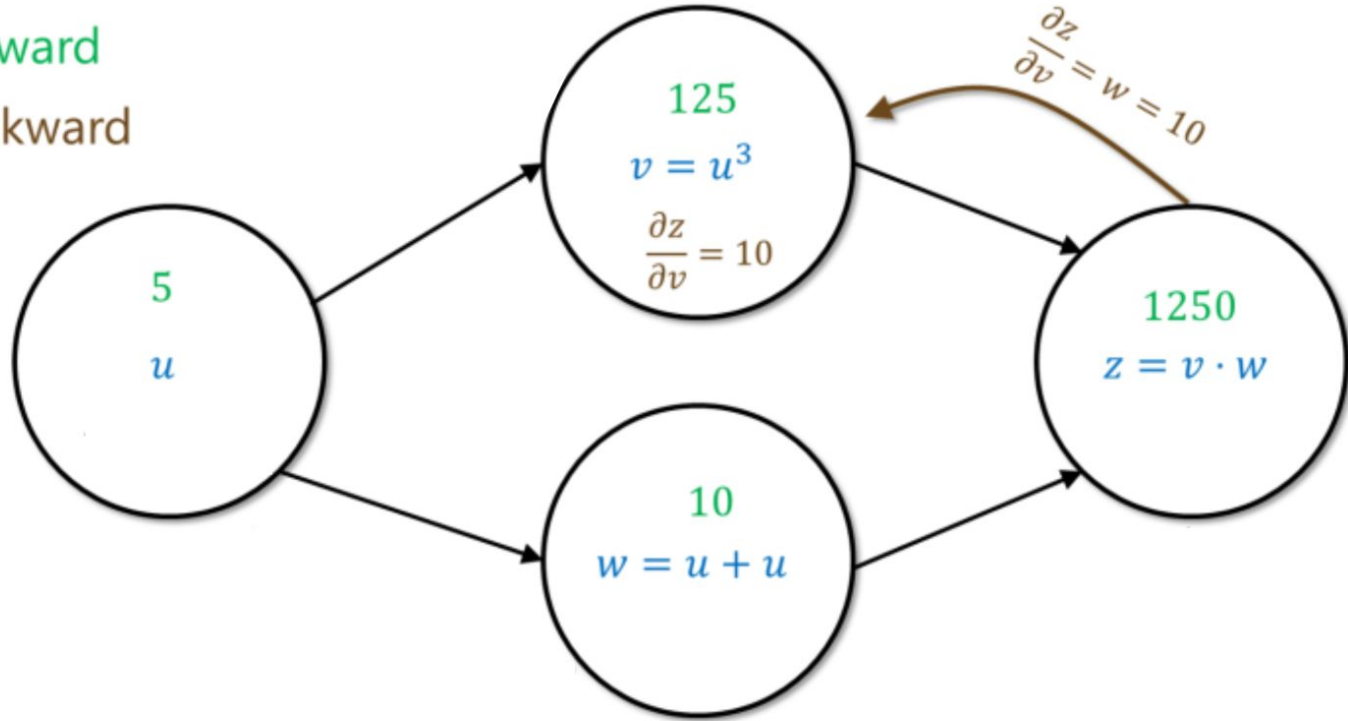


Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w} \right)$$

Forward

Backward

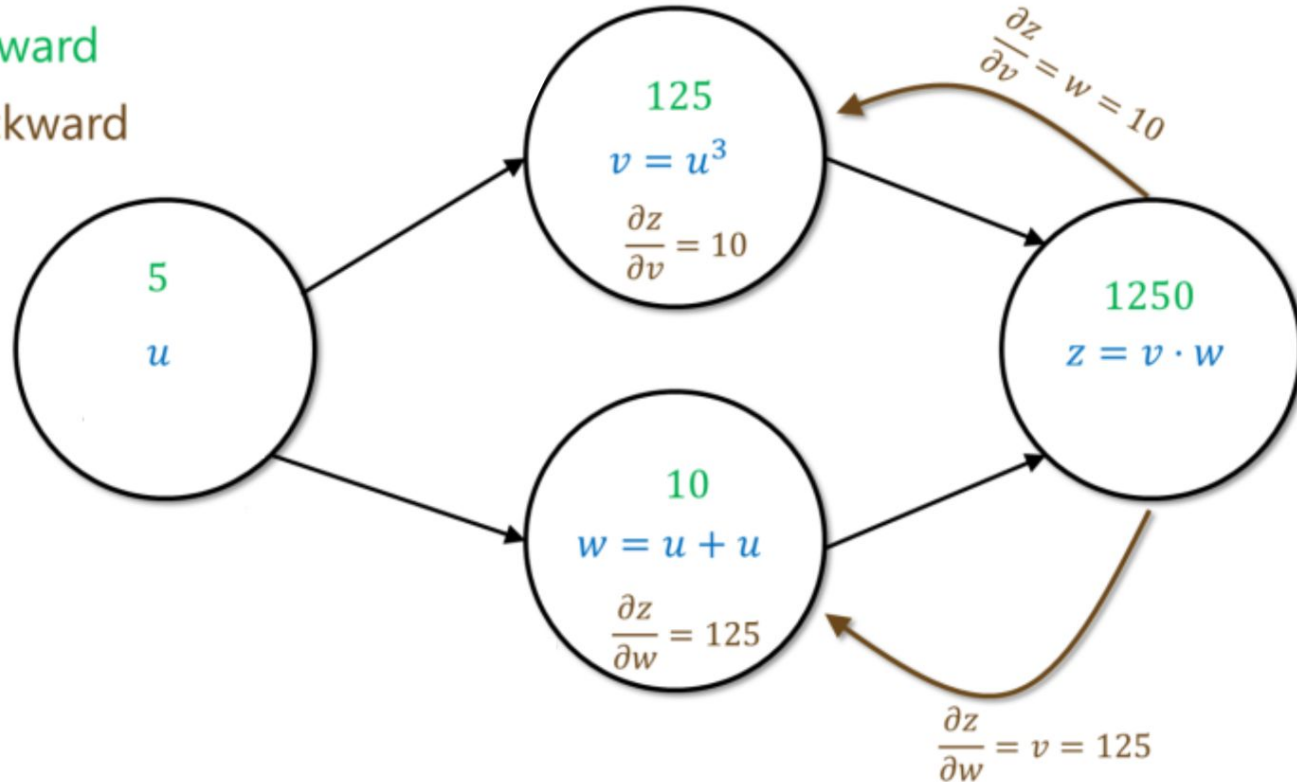


Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w} \right)$$

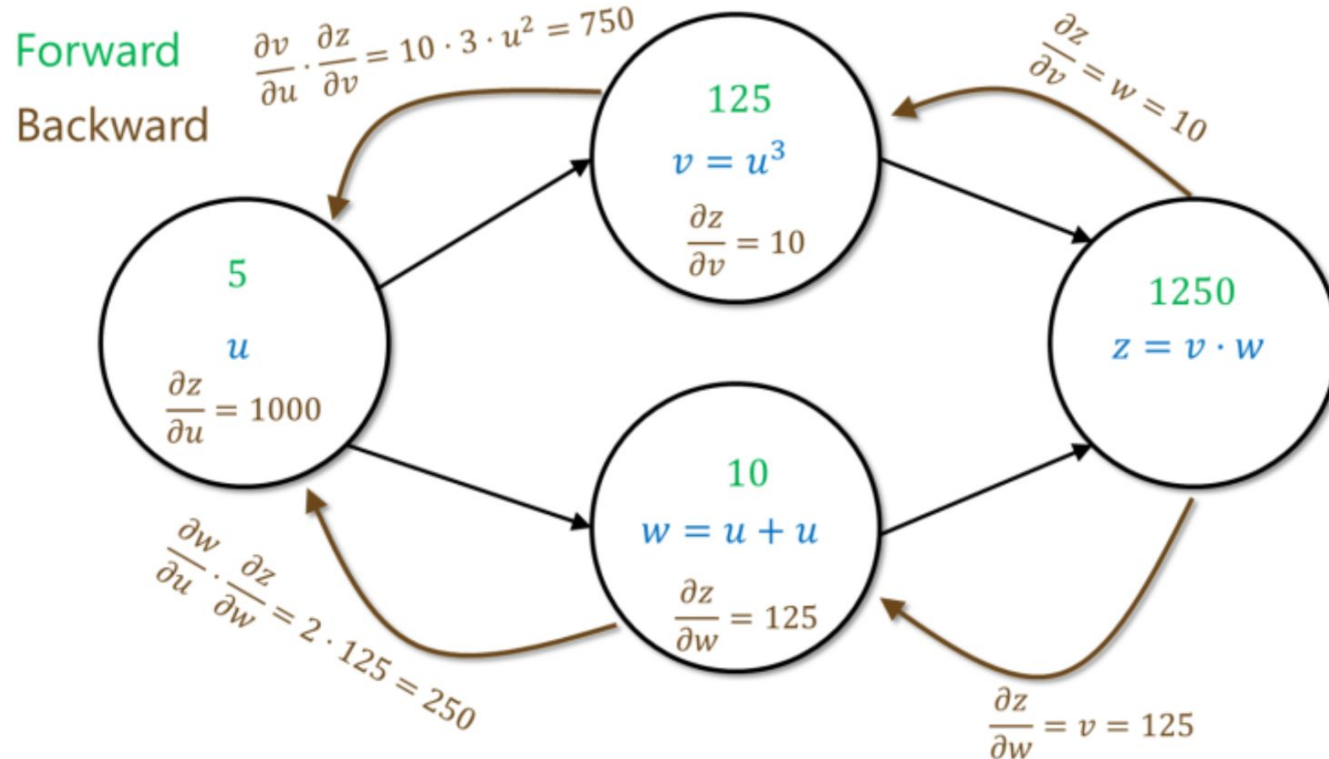
Forward

Backward

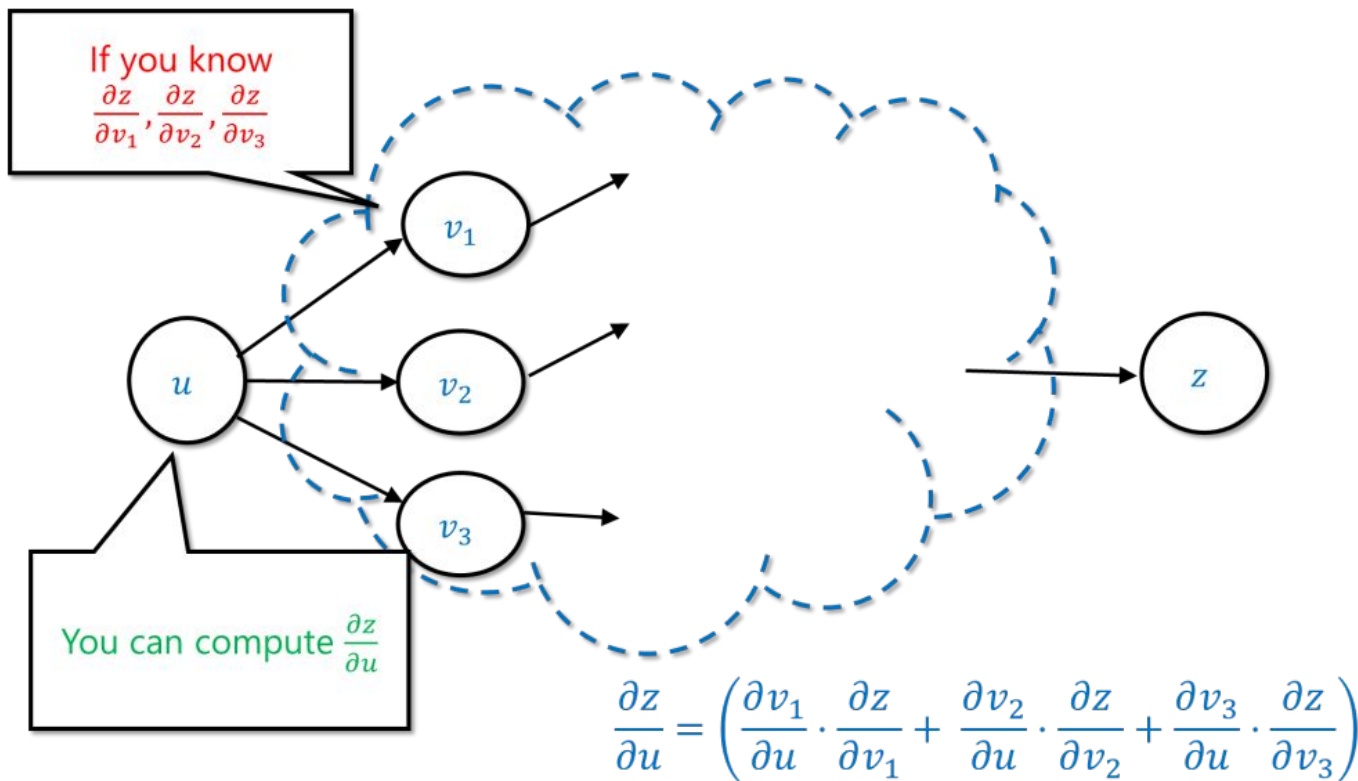


Backpropagation- An Example

$$\frac{\partial z}{\partial u} = \left(\frac{\partial v}{\partial u} \cdot \frac{\partial z}{\partial v} + \frac{\partial w}{\partial u} \cdot \frac{\partial z}{\partial w} \right)$$



Backpropagation- Key Idea

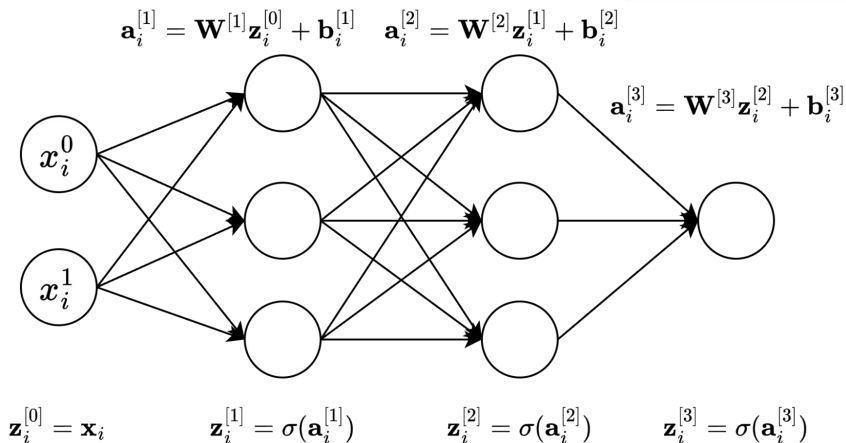


Preview

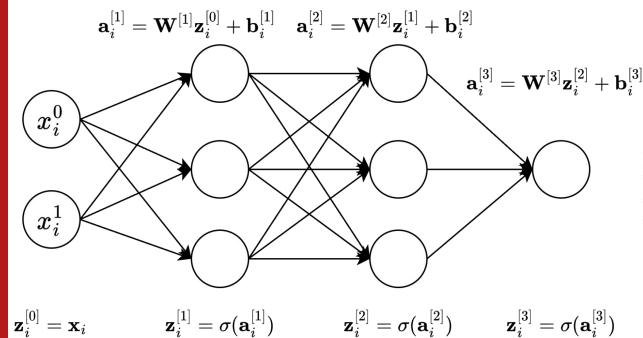
Backpropagation- MLPs

Algorithm Forward Pass through MLP

- 1: **Input:** input $\mathbf{x}$, weight matrices $\mathbf{W}^{[1]}, \dots, \mathbf{W}^{[L]}$, bias vectors $\mathbf{b}^{[1]}, \dots, \mathbf{b}^{[L]}$
 - 2: $\mathbf{z}^{[0]} = \mathbf{x}$ ▷ Initialize input
 - 3: **for** $l = 1$ **to** L **do**
 - 4: $\mathbf{a}^{[l]} = \mathbf{W}^{[l]} \mathbf{z}^{[l-1]} + \mathbf{b}^{[l]}$ ▷ Linear transformation
 - 5: $\mathbf{z}^{[l]} = \sigma^{[l]}(\mathbf{a}^{[l]})$ ▷ Nonlinear activation
 - 6: **end for**
 - 7: **Output:** $\mathbf{z}^{[L]}$
-



Backpropagation- MLPs



Algorithm Forward Pass through MLP

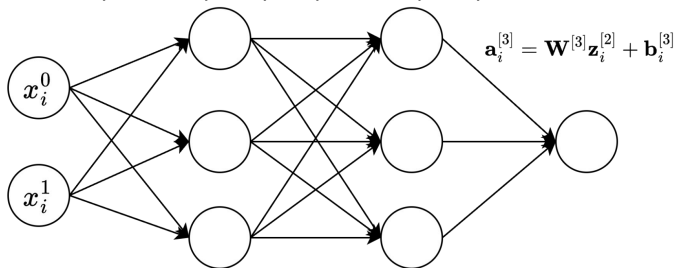
- 1: **Input:** input $\mathbf{x}$, weight matrices $\mathbf{W}^{[1]}, \dots, \mathbf{W}^{[L]}$, bias vectors $\mathbf{b}^{[1]}, \dots, \mathbf{b}^{[L]}$
 - 2: $\mathbf{z}^{[0]} = \mathbf{x}$ ▷ Initialize input
 - 3: **for** $l = 1$ **to** L **do**
 - 4: $\mathbf{a}^{[l]} = \mathbf{W}^{[l]} \mathbf{z}^{[l-1]} + \mathbf{b}^{[l]}$ ▷ Linear transformation
 - 5: $\mathbf{z}^{[l]} = \sigma^{[l]}(\mathbf{a}^{[l]})$ ▷ Nonlinear activation
 - 6: **end for**
 - 7: **Output:** $\mathbf{z}^{[L]}$
-

Algorithm Backward Pass through MLP

- 1: **Input:** $\{\mathbf{z}^{[1]}, \dots, \mathbf{z}^{[L]}\}$, $\{\mathbf{a}^{[1]}, \dots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$
 - 2: $\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]'}(\mathbf{a}^{[L]})$ ▷ Error term
 - 3: **for** $l = L$ **to** 1 **do**
 - 4: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights
 - 5: $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases
 - 6: $\delta^{[l-1]} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]'}(\mathbf{a}^{[l-1]})$
 - 7: **end for**
 - 8: **Output:** $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$
-

Backpropagation- MLPs

$$\mathbf{a}_i^{[1]} = \mathbf{W}^{[1]} \mathbf{z}_i^{[0]} + \mathbf{b}_i^{[1]} \quad \mathbf{a}_i^{[2]} = \mathbf{W}^{[2]} \mathbf{z}_i^{[1]} + \mathbf{b}_i^{[2]}$$



$$\mathbf{z}_i^{[0]} = \mathbf{x}_i \quad \mathbf{z}_i^{[1]} = \sigma(\mathbf{a}_i^{[1]}) \quad \mathbf{z}_i^{[2]} = \sigma(\mathbf{a}_i^{[2]}) \quad \mathbf{z}_i^{[3]} = \sigma(\mathbf{a}_i^{[3]})$$

Algorithm Forward Pass through MLP

- 1: **Input:** input $\mathbf{x}$, weight matrices $\mathbf{W}^{[1]}, \dots, \mathbf{W}^{[L]}$, bias vectors $\mathbf{b}^{[1]}, \dots, \mathbf{b}^{[L]}$
 - 2: $\mathbf{z}^{[0]} = \mathbf{x}$ ▷ Initialize input
 - 3: **for** $l = 1$ **to** L **do**
 - 4: $\mathbf{a}^{[l]} = \mathbf{W}^{[l]} \mathbf{z}^{[l-1]} + \mathbf{b}^{[l]}$ ▷ Linear transformation
 - 5: $\mathbf{z}^{[l]} = \sigma^{[l]}(\mathbf{a}^{[l]})$ ▷ Nonlinear activation
 - 6: **end for**
 - 7: **Output:** $\mathbf{z}^{[L]}$
-

Algorithm Backward Pass through MLP (Detailed)

- 1: **Input:** $\{\mathbf{z}^{[1]}, \dots, \mathbf{z}^{[L]}\}, \{\mathbf{a}^{[1]}, \dots, \mathbf{a}^{[L]}\}$, loss gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}}$
 - 2: $\delta^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{a}^{[L]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[L]}} \odot \sigma^{[L]'}(\mathbf{a}^{[L]})$ ▷ Error term
 - 3: **for** $l = L$ **to** 1 **do**
 - 4: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} (\mathbf{z}^{[l-1]})^T$ ▷ Gradient of weights
 - 5: $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$ ▷ Gradient of biases
 - 6: $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l]}} \frac{\partial \mathbf{a}^{[l]}}{\partial \mathbf{z}^{[l-1]}} = (\mathbf{W}^{[l]})^T \delta^{[l]}$
 - 7: $\delta^{[l-1]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[l-1]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l-1]}} \frac{\partial \mathbf{z}^{[l-1]}}{\partial \mathbf{a}^{[l-1]}} = ((\mathbf{W}^{[l]})^T \delta^{[l]}) \odot \sigma^{[l-1]'}(\mathbf{a}^{[l-1]})$
 - 8: **end for**
 - 9: **Output:** $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[1:L]}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[1:L]}}$
-

Takeaways

- MLPs consist of stacks of perceptron units
- MLPs can learn complex decision boundaries
by composing simple features into more complex features
- Learn MLP weights with gradient descent
 - Backpropagation efficiently computes gradient

Next Week

A deep dive into training neural networks!

