Artificial Intelligence

CS3AI18/ CSMAI19

Lecture – 6/10: Learning (Fundamental Theory)

DR VARUN OJHA

Department of Computer Science

University of Reading

Learning objectives

By the end of this week, you will be able to

- Learn basic concepts of learning
- Gradient descent and backpropagation learning
- Learn to avoid overfitting learning models.
- Workout an example problem

Content of this week

- Part 1: Introduction
- Part 2: Fundamental Theory
  - Types of Learning
  - Supervised Learning Problem Definition
  - Learning process design
- Part 3: Algorithms
  - Gradient Descent
  - Online (Stochastic) Vs Offline (Batch) training
  - The Backpropagation algorithm
  - Avoiding Overfitting
- Part 4: Practical Exercise
- Quiz

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 6/10: Learning

Part 1
Introduction

DR VARUN OJHA

Department of Computer Science

University of Reading

Common Sense → Intelligence?

Common Sense: Inside a baby’s mind

Slide inspiration: Josh Tenenbaum, Prof. MIT, USA
Video Source:
https://www.youtube.com/watch?v=dEnDjyWHN4A
(Accessed on 21 Feb 2021)

Common Sense: Inside a baby's mind

Slide inspiration: Tenenbaum J, MIT, USA

Experiment:
Warneken &Tomasello (2006)

Video Source:
https://www.youtube.com/watch?v=cUWIIxpUfM0
(Accessed on 21 Feb 2021)

Understanding → Intelligence?

Causal understanding of water displacement by a crow

Slide inspiration: Tenenbaum J, MIT, USA
Experiment: Sarah et al. (2014), Auckland and Cambridge
Video Source: https://www.youtube.com/watch?v=ZerUbHmuY04

What is a Learning?

Learning / Training

Video Source:
https://www.youtube.com/watch?v=Ak7bPuR2rDw
(Accessed on 21 Feb 2021)

Learning/ Training

Video source:
https://www.youtube.com/watch?v=nbrTOcUnjNY
(Accessed on 21 Feb 2021)

Learning → Intelligence?

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 6/10: Learning

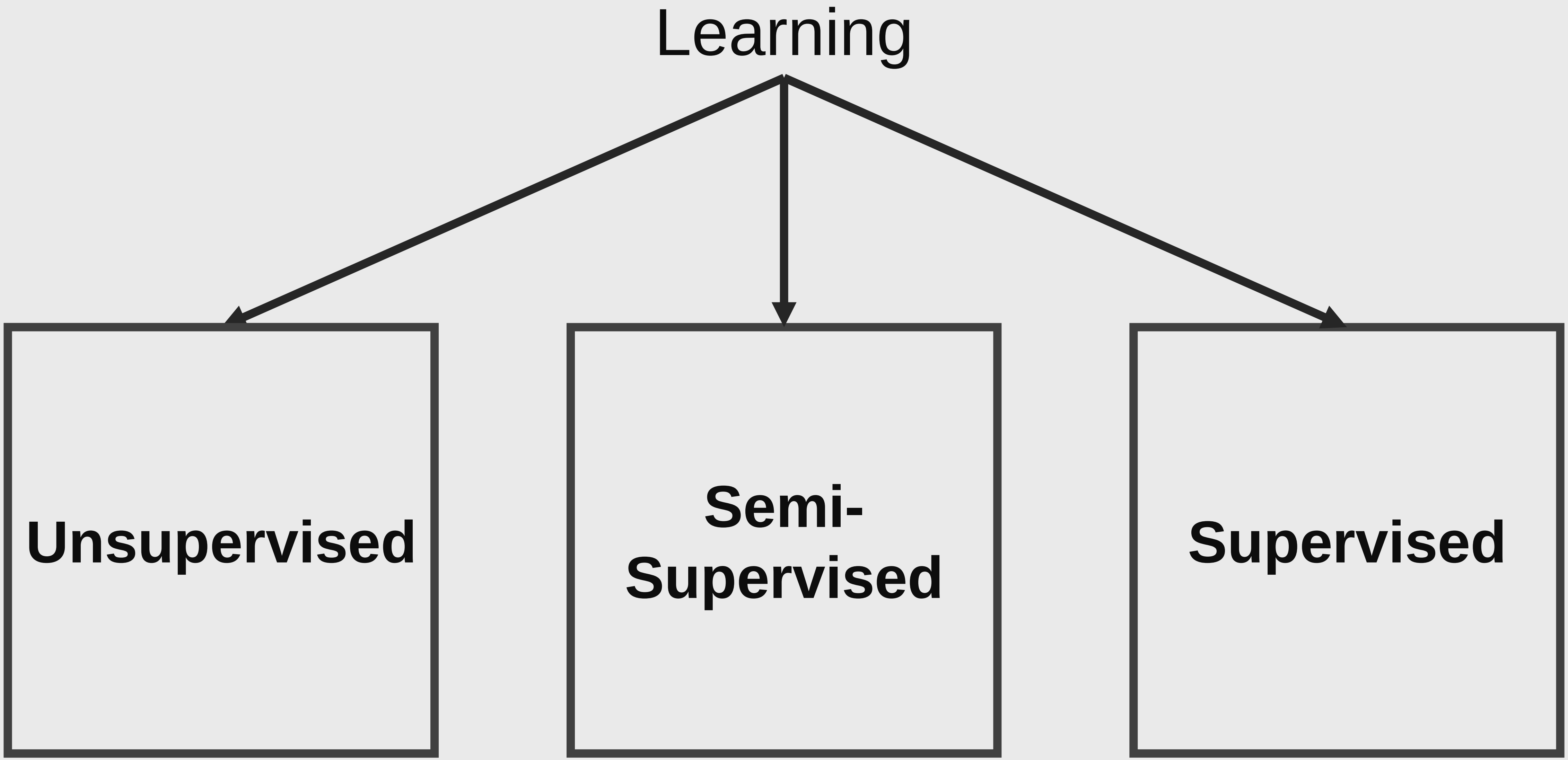
Part 2

Learning: Theory

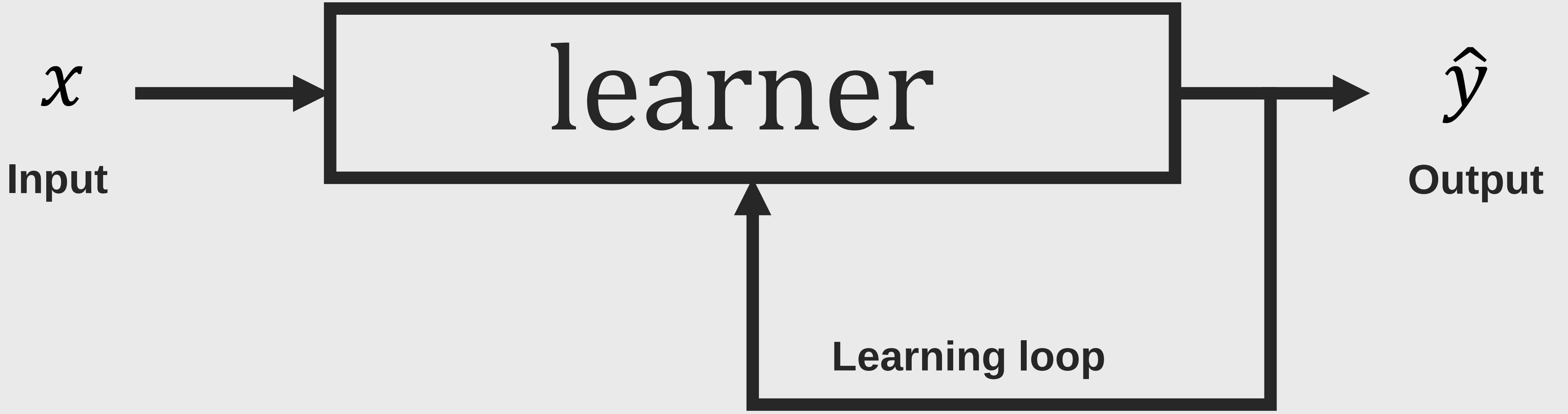
DR VARUN OJHA

Department of Computer Science

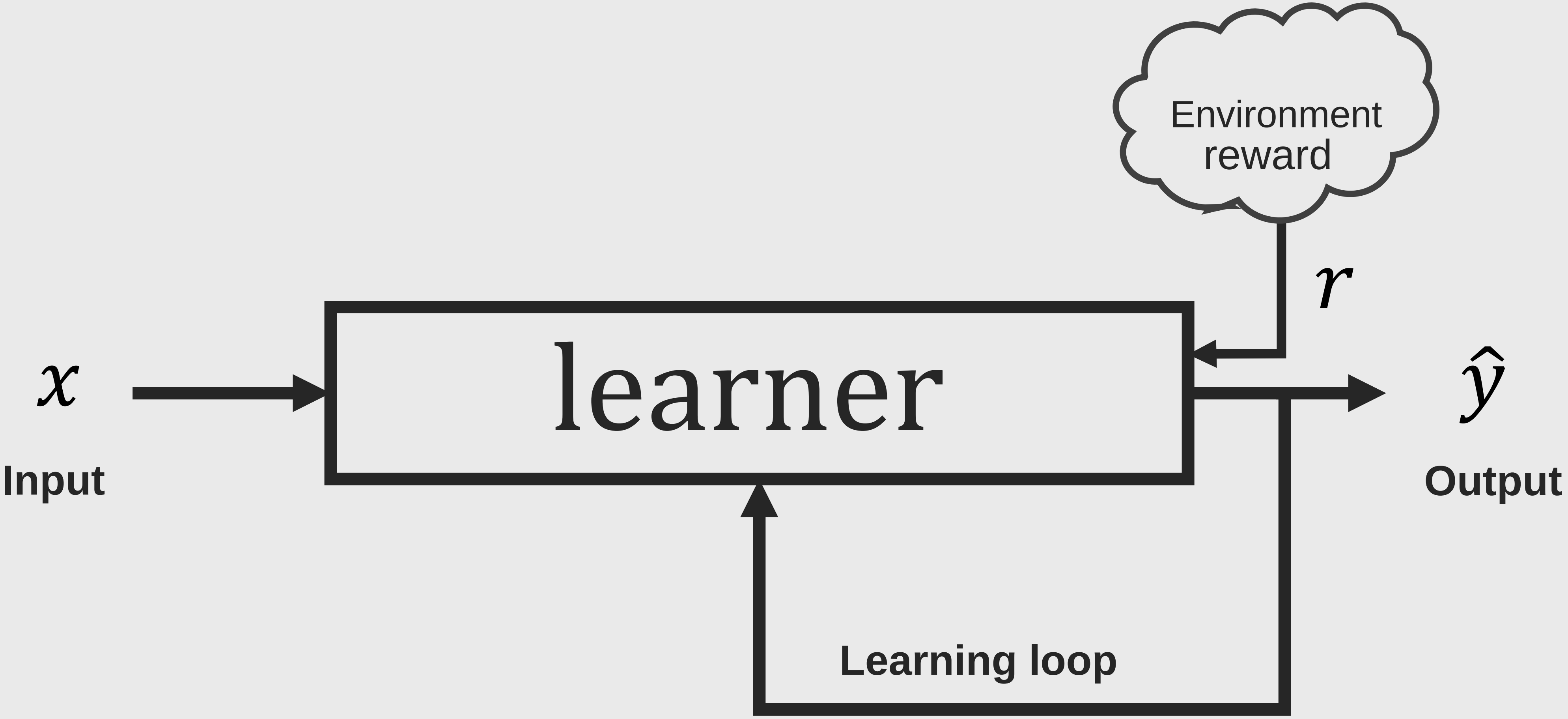
University of Reading


Learning
Unsupervised
Semi-Supervised
Supervised

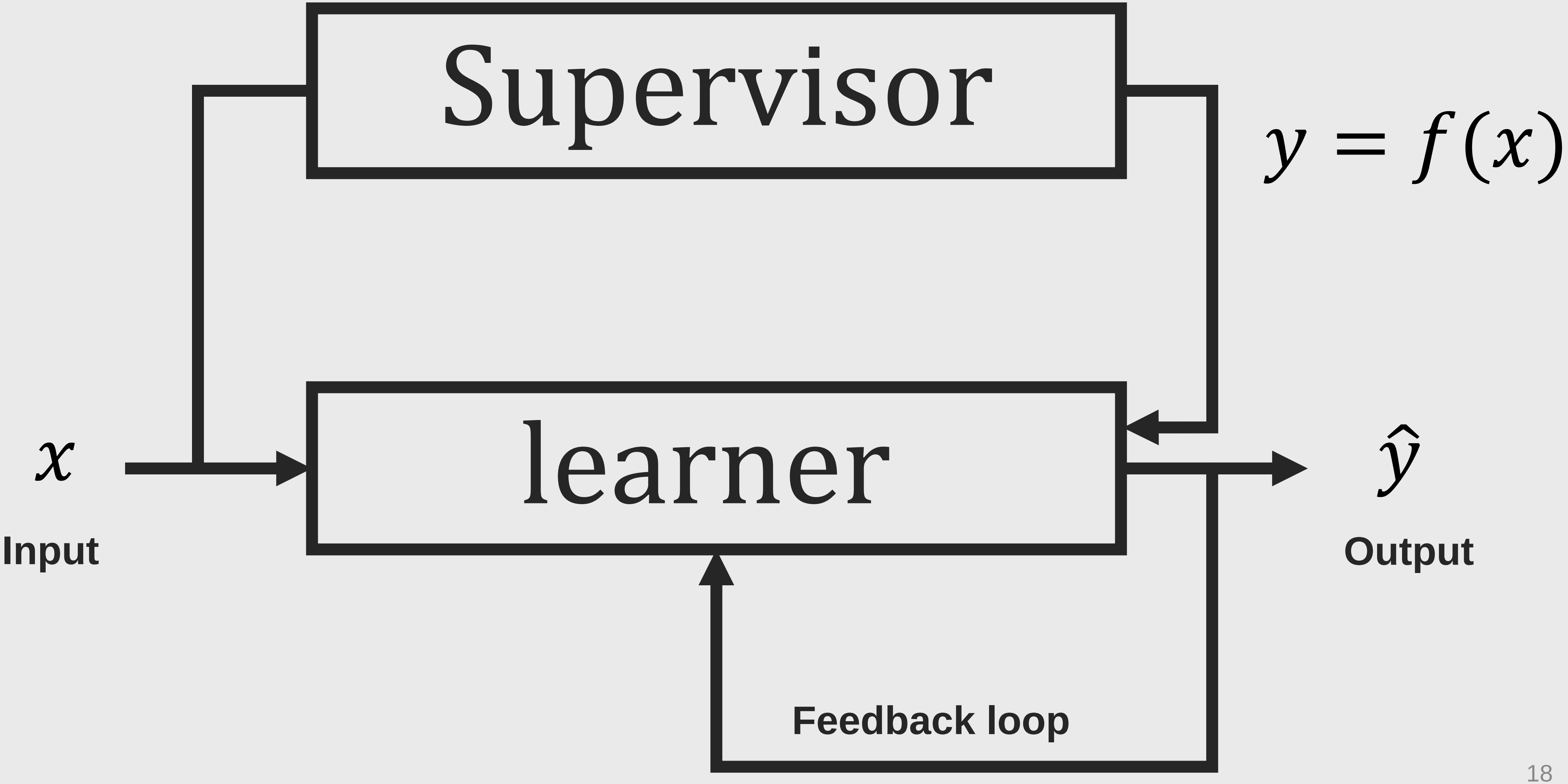
Unsupervised



Semi-Supervised



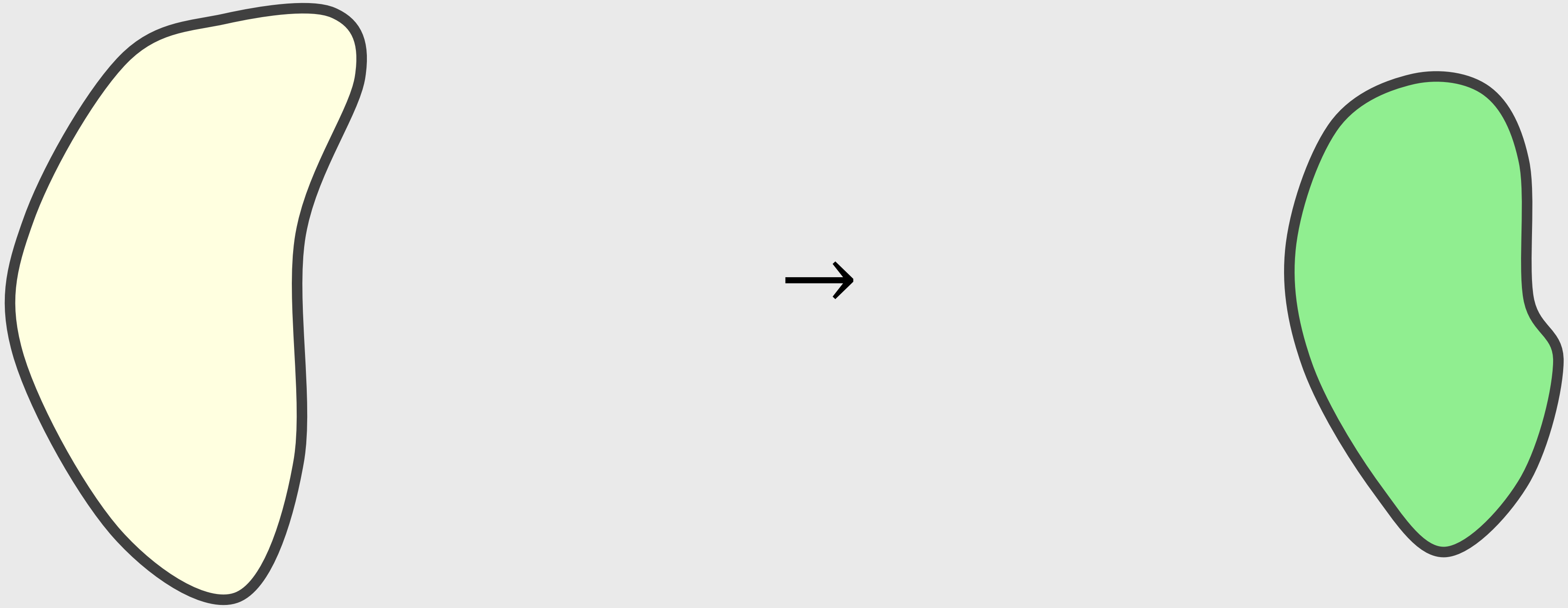
Supervised



Learning $\equiv \mathcal{X} \rightarrow \mathcal{Y}$

Supervised learning is a mapping f of inputs $\mathcal{X}$ to outputs $\mathcal{Y}$

$\mathcal{X}$ $\rightarrow$ $\mathcal{Y}$



Input-space

Output / Concept-space

Learning $f : \mathcal{X} \rightarrow \mathcal{Y}$

Supervised learning is a mapping f of inputs $\mathcal{X}$ to outputs $\mathcal{Y}$

Inputs $\mathbf{X} \in$ Input space $\mathcal{X}$

outputs $\boldsymbol{y} \in$ concept space $\mathcal{Y}$

Learning $\boldsymbol{f} : \boldsymbol{X} \rightarrow \boldsymbol{y}$

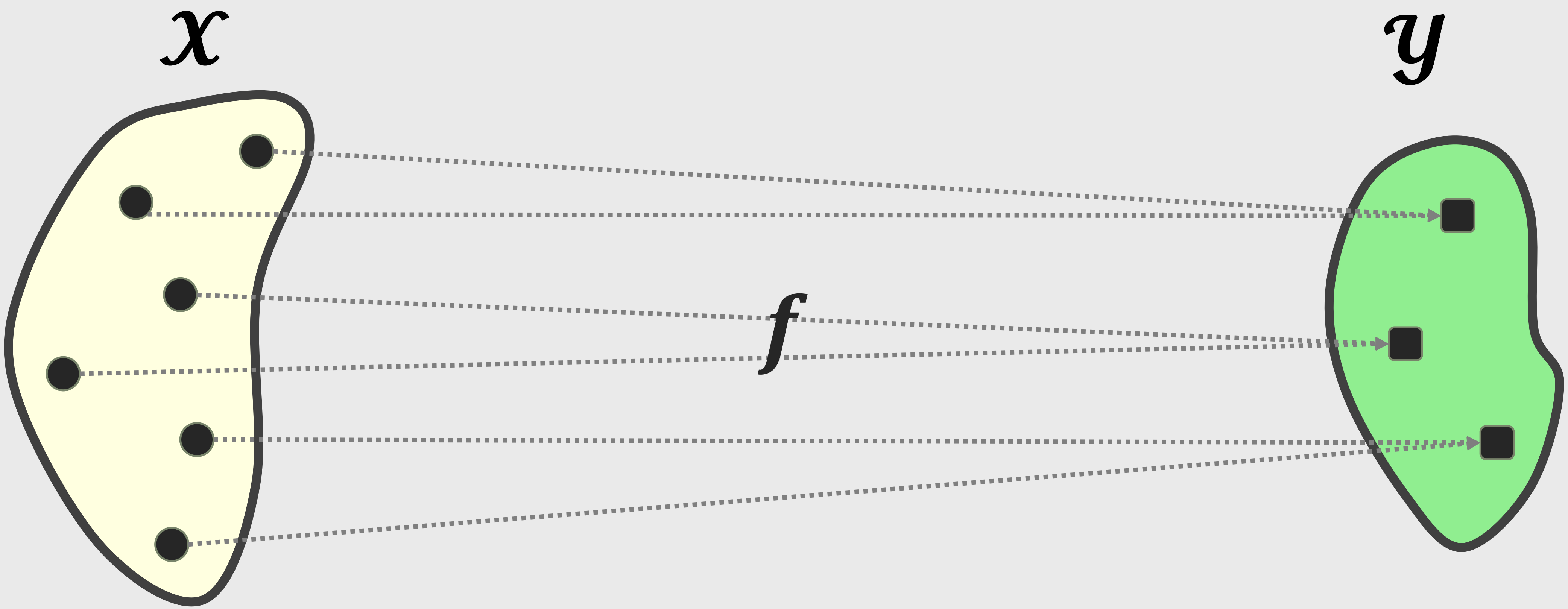
Supervised learning is a mapping f of inputs $\mathcal{X}$ to outputs $\mathcal{Y}$

Inputs $\mathbf{X} \in$ Input space $\mathcal{X}$

outputs $\boldsymbol{y} \in$ concept space $\mathcal{Y}$

Learning $f : X \to y$

Supervised learning is a mapping f of inputs $\mathcal{X}$ to outputs $\mathcal{Y}$

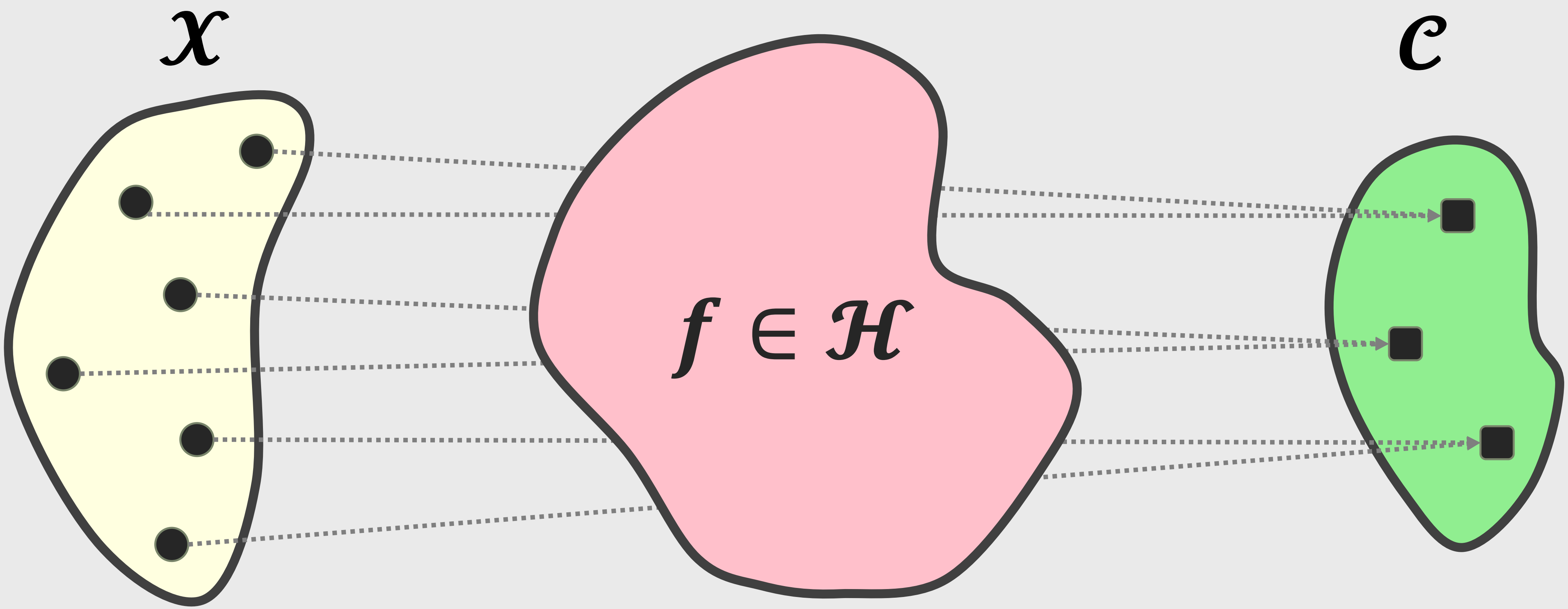


Inputs $\mathbf{X} \in$ Input space $\mathcal{X}$

outputs $\boldsymbol{y} \in$ output space $\mathcal{Y}$

Learning $f : X \rightarrow y$

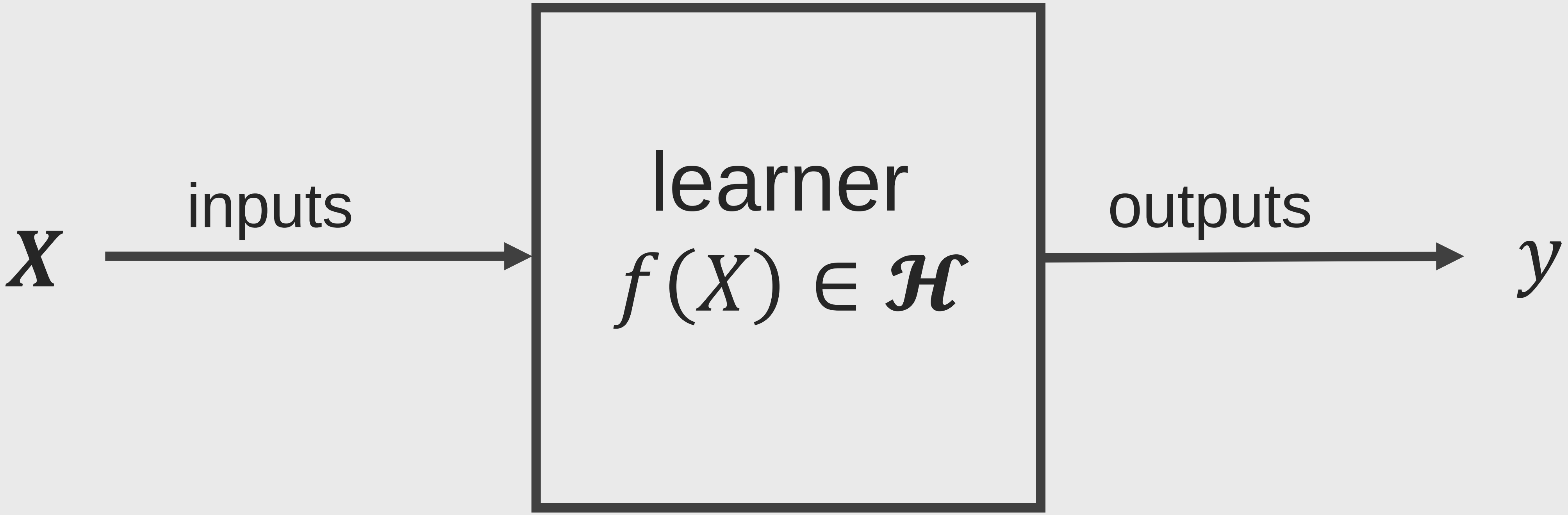
We need to find the unwon target function f that does the task of mapping



Inputs $\mathbf{X} \in$ Input space $\mathcal{X}$ hypothesis space $\mathcal{H}$ outputs $\boldsymbol{y} \in$ concept space $\mathcal{C}$

Learning $f: \boldsymbol{X} \rightarrow \boldsymbol{y}$

Supervised learning is a mapping f of inputs X to outputs y



Learning

Example Training Task: AND Logic Problem

	x_1	x_2	y
1:	0	0	0
2:	0	1	0
3:	1	0	0
4:	1	1	1

$\boldsymbol{y} = f(\mathbf{X})$,
where $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4})^T$, $\mathbf{x}_i = (x_{i1}, x_{i2})$,
and $\boldsymbol{y} = (y_1, y_2, y_3, y_4)^T$

number of Inputs $d = 2$
each input x takes 2 options 0 or 1
input-space $\mathcal{X} = 2^d = 2^2 = 4$

number of outputs 1
output y takes 2 options from $\{0,1\}$
concept-space $\mathcal{C} = 2^I = 2^{2^2} = 16$

Learning

Example Training Task: AND Logic Problem

	x_1	x_2	y
1:	0	0	0
2:	0	1	0
3:	1	0	0
4:	1	1	1

input-space $|\mathcal{X}| = 2^d = 2^2 = 4$

inputs $\mathbf{X} \in$ Input space $\mathcal{X}$

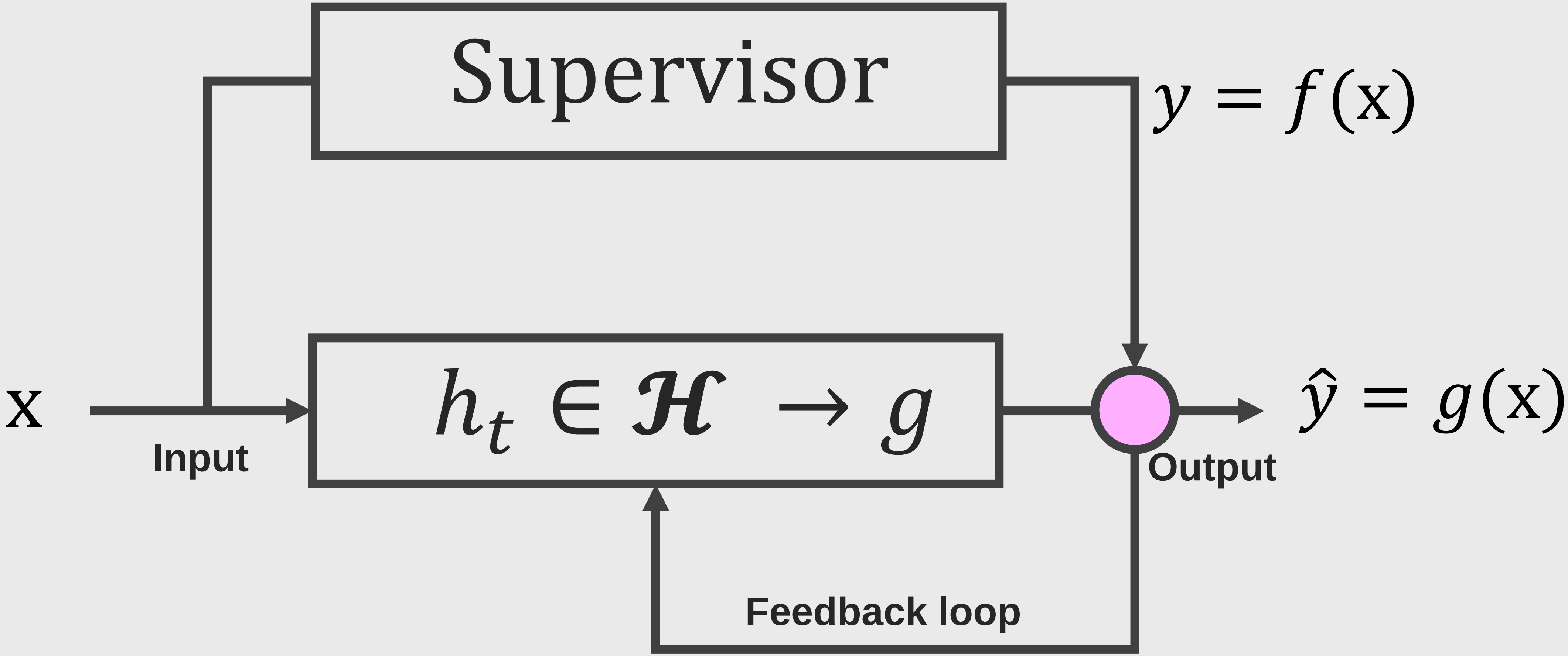
concept-space $\mathcal{C} = 2^{|\mathcal{X}|} = 2^{2^2} = 16$

outputs $\boldsymbol{y} \in$ concept space $\boldsymbol{\mathcal{C}}$

Hypothesis space: $\boldsymbol{\mathcal{H}}$ is a set of all possible functions such that $h_t \in \boldsymbol{\mathcal{H}}$ produces a function $g: \mathbf{X} \rightarrow \boldsymbol{y}$ that approximates f i.e., $g \approx f$.

data-space (training data): $\mathcal{D} = \{(\mathbf{x}_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_N, f(\mathbf{x}_N))\}$, where $\mathcal{D} \in \boldsymbol{\mathcal{C}}$ are N training examples.

How to produces a function $g: \mathbf{X} \to \boldsymbol{y}$



What Learning Needs

Learning needs the method(s) to

Represent

Evaluate

Optimize

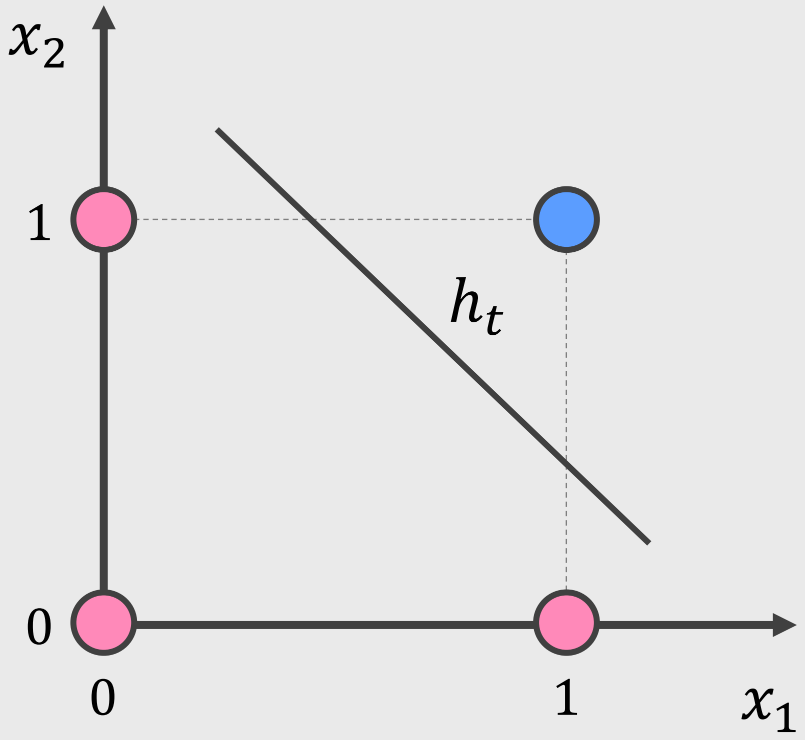
a hypothesis h_t:

How to represent a hypothesis $h_t \in H$

A line separating data can be consider a hypothesis

$\mathcal{D}$

	x_1	x_2	y
1:	0	0	0
2:	0	1	0
3:	1	0	0
4:	1	1	1



How to represent a hypothesis $h_t \in H$

A line separating data can be consider a hypothesis

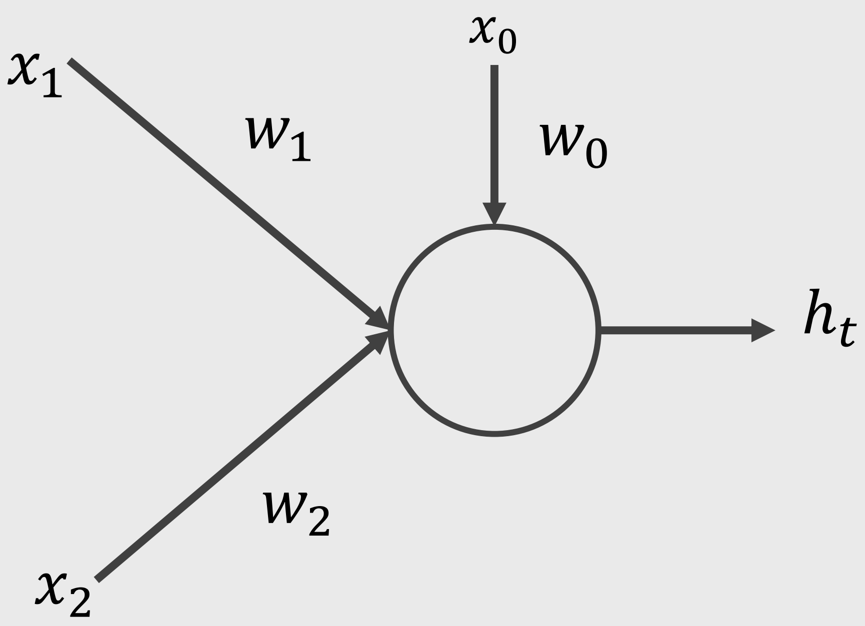
A hypothesis h_t as a **perceptron**.

Perceptron : a simple linear combination of inputs.

$h_t = g(x) = \sum_{i=1}^{d} w_i x_i \geq x_0 w_0$,

where w_0 is a threshold.

The hypothesis h_t has the weights w_i and the threshold w_0 as its trainable parameters.



perceptron

How to represent a hypothesis $h_t \in H$

A line separating data can be consider a hypothesis

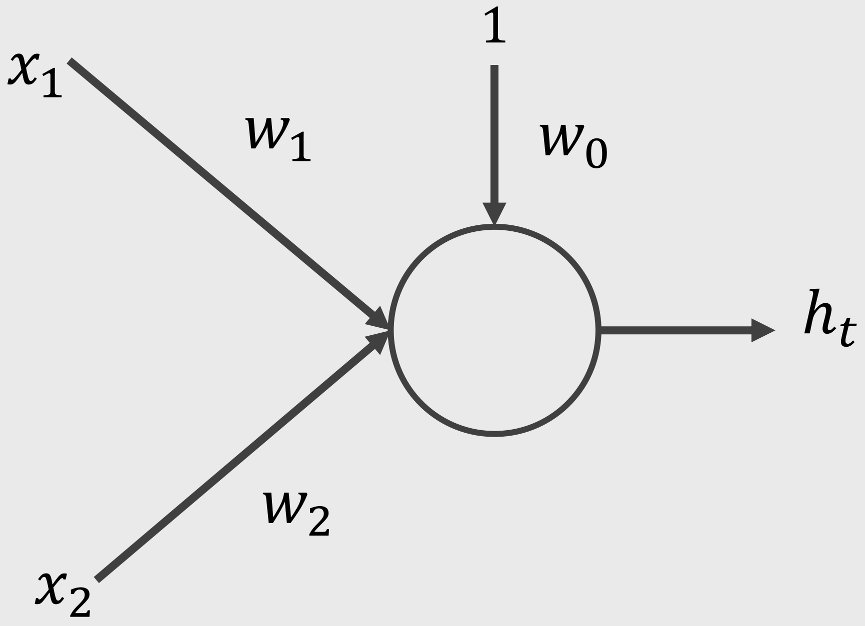
A hypothesis h_t as a perceptron.

$$\sum_{i=1}^{d} w_i\, x_i \geq x_0 w_0$$

$$\sum_{i=1}^{d} w_i\, x_i - x_0 w_0 = 0$$

For an artificial input $x_0 = 1$

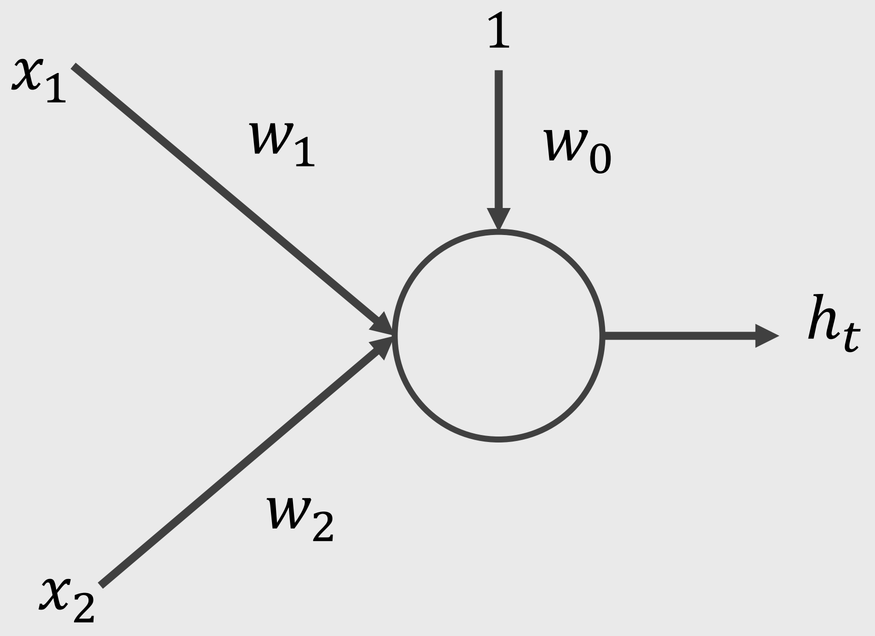
$$\sum_{i=0}^{d} w_i\, x_i = 0$$

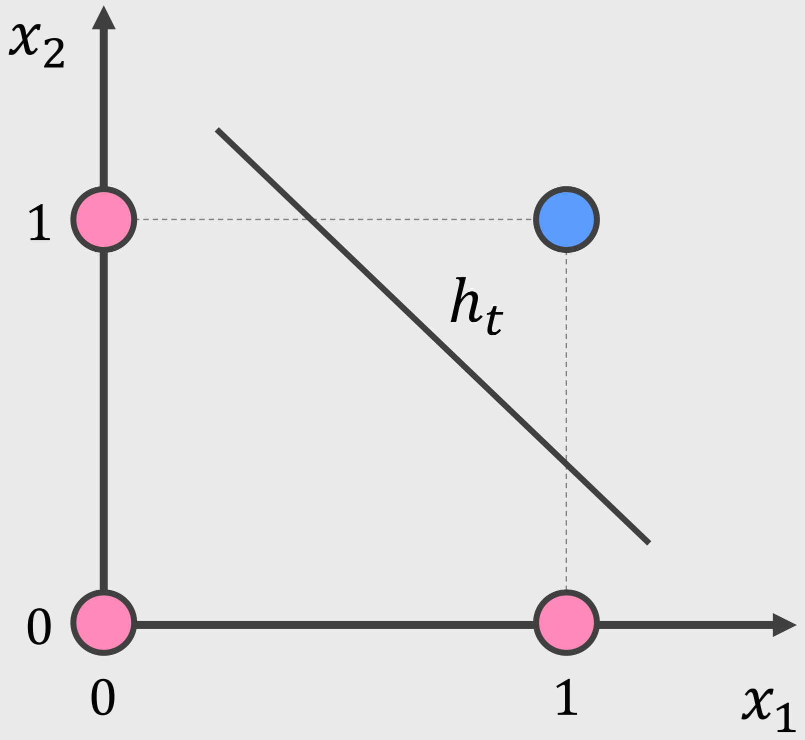


perceptron

How to represent a hypothesis $h_t \in H$

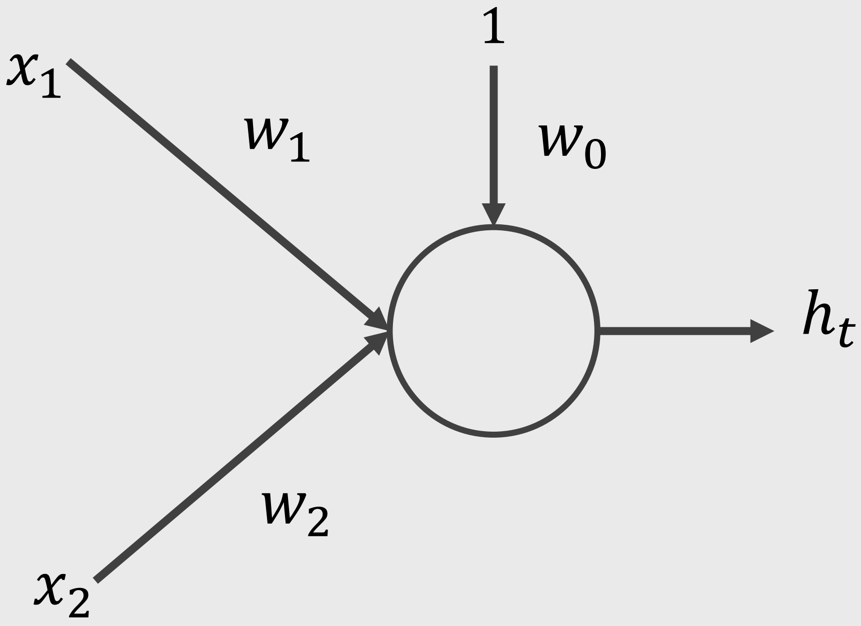
A line separating data can be consider a hypothesis

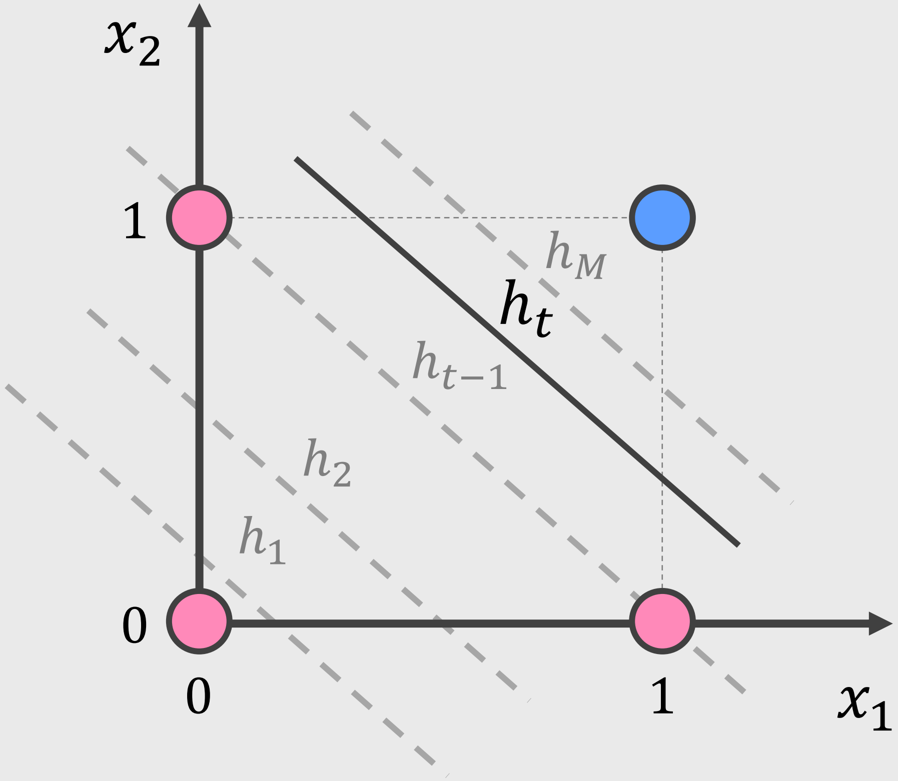




Which hypothesis $h_t \in H$ to pick?

How to evaluate a hypothesis: compute cost of hypothesis





Which hypothesis $h_t \in H$ to pick?

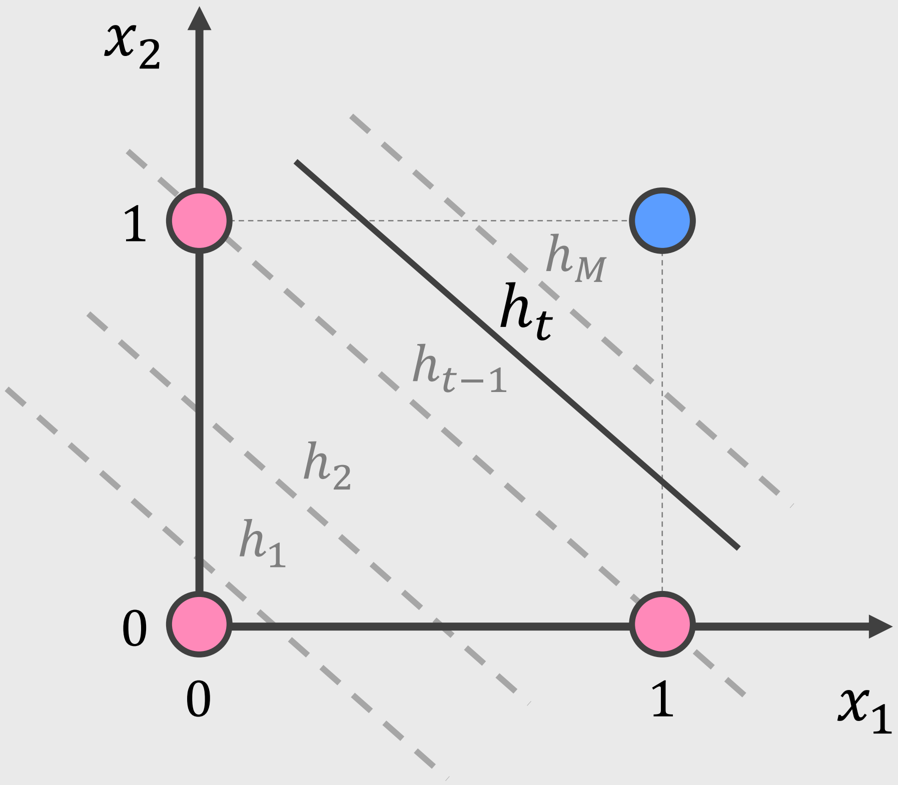
How to evaluate a hypothesis: compute cost of hypothesis

	x_1	x_2	$y = f(\mathbf{x})$
1:	0	0	0
2:	0	1	0
3:	1	0	0
4:	1	1	1

$\mathcal{D}$

Cost function such as the error rate:

$$E(h_t(\mathcal{D})) = \frac{1}{N}\sum_{j=1}^{N}\left(g(\mathbf{x}_j) \neq f(\mathbf{x}_j)\right)$$



How to search optimum hypothesis $h_t \in H$

How to evaluate a hypothesis: compute cost of hypothesis

Function g of the hypothesis has parameter $\mathbf{w}$:

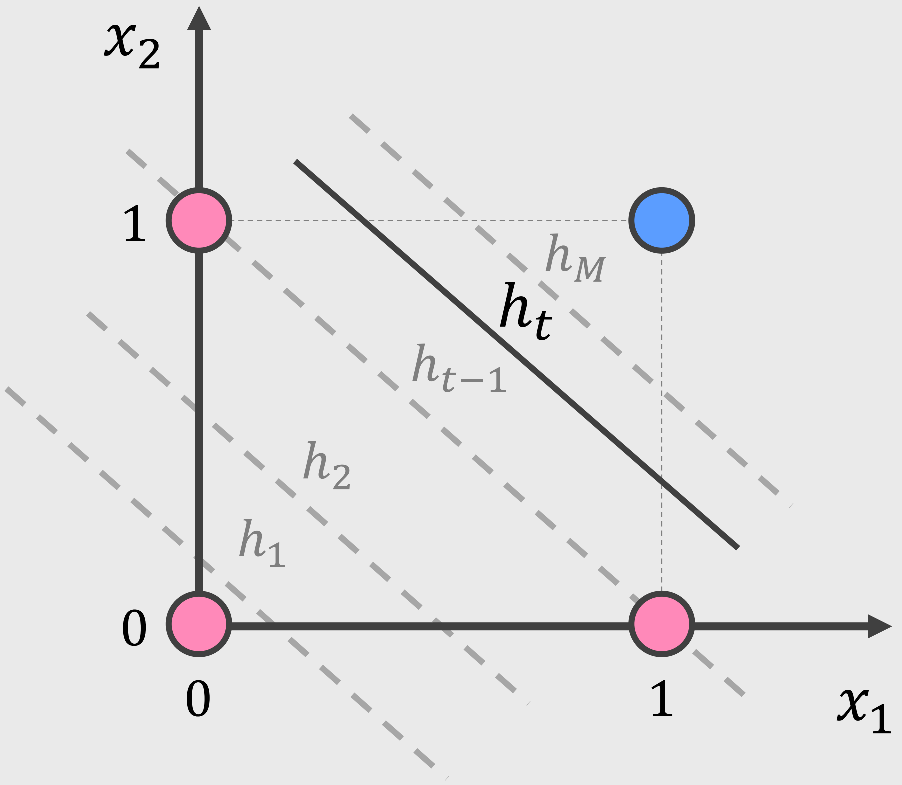
$$\hat{y} = g_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{d} w_i\, x_i = 0$$

Simple algorithm:

Repeat parameter $\mathbf{w}$ update for $t = 2, 3, \dots, M$.

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \hat{y}\,\mathbf{x}$$

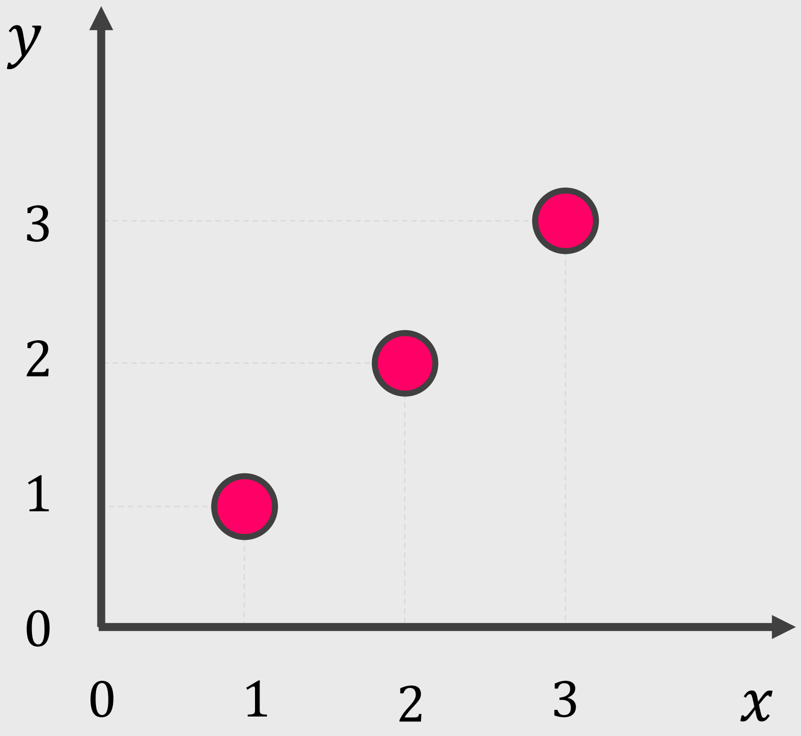
Until error rate $E(h_t(\mathcal{D}))$ is acceptable.



Does error $E(h_t(\mathcal{D}))$ minimization work?

Let's see an example (house price):

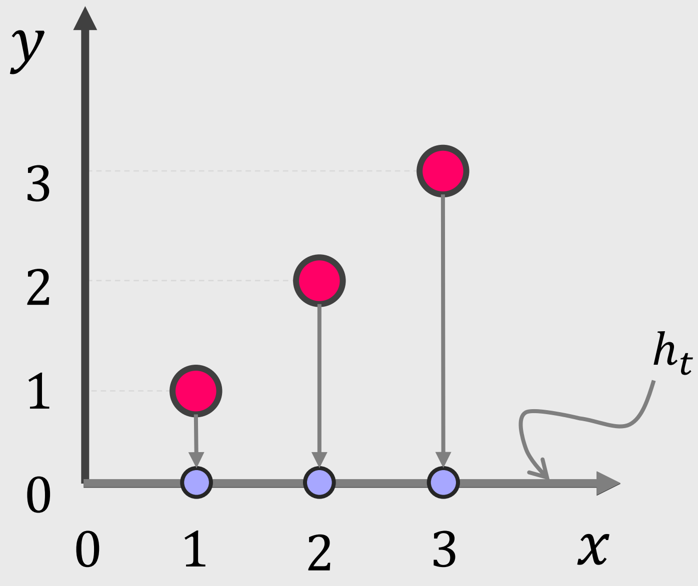
	$x = area(m^2)$	$y = price\ (in\ £)$
1:	1000	100K
2:	2000	200K
3:	3000	300K

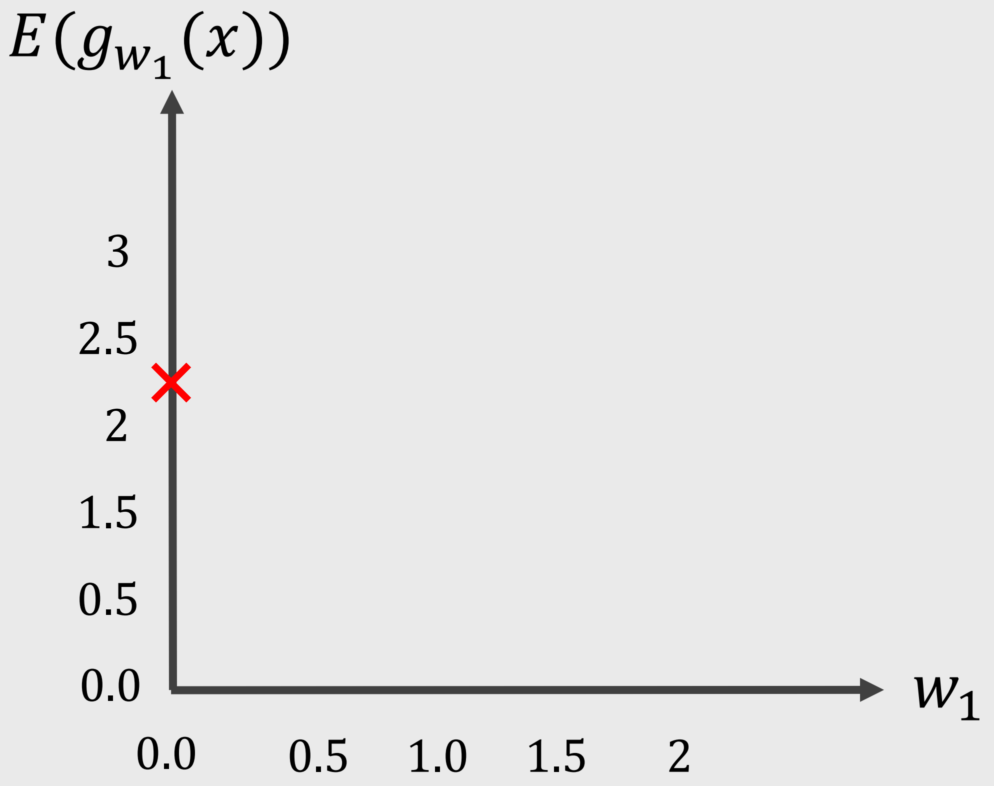


Now, cost function is a squared error:

$$E(h_t(\mathbf{x}) = \frac{1}{2N}\sum_{j=1}^{N}\left(g(\mathbf{x}_j) - f(\mathbf{x}_j)\right)^2$$

Does error $E(h_t(\mathcal{D}))$ minimization work?





Hypothesis h_t for $w_0 = 0$ and $w_1 = 0.0$:

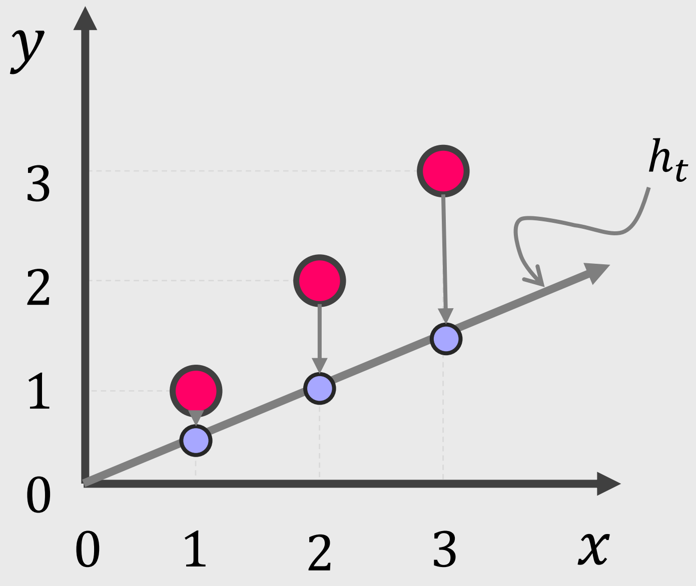
$g(x_i) = w_0 + w_1 x_i$ for $i = 1,2,3$

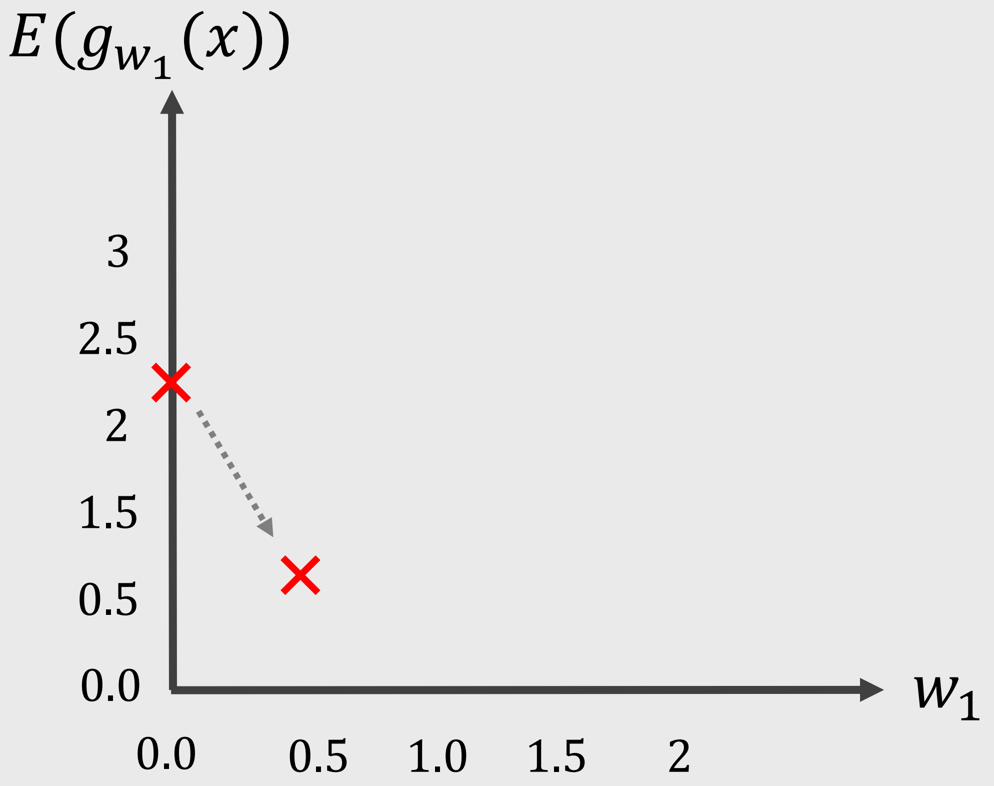
Error $E(w_1)$ for $w_0 = 0$ and $w_1 = 0.0$:

$$E\big(g_{\mathbf{w}}(\mathbf{x})\big) = \frac{(1-0)^2 + (2-0)^2 + (3-0)^2}{2*3} = 2.33$$

Does error $E(h_t(\mathcal{D}))$ minimization work?

9:59 PM





Hypothesis h_t for $w_0 = 0$ and $w_1 = 0.5$:

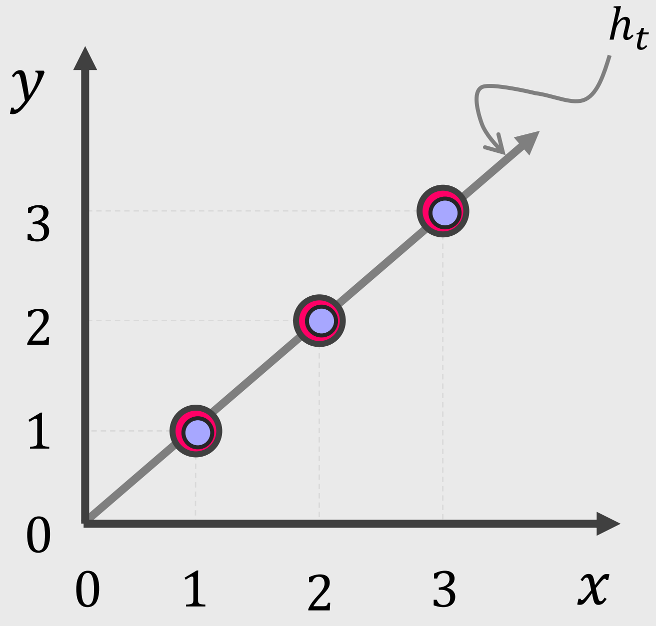
$$g(x_i) = w_0 + w_1 x_i$$

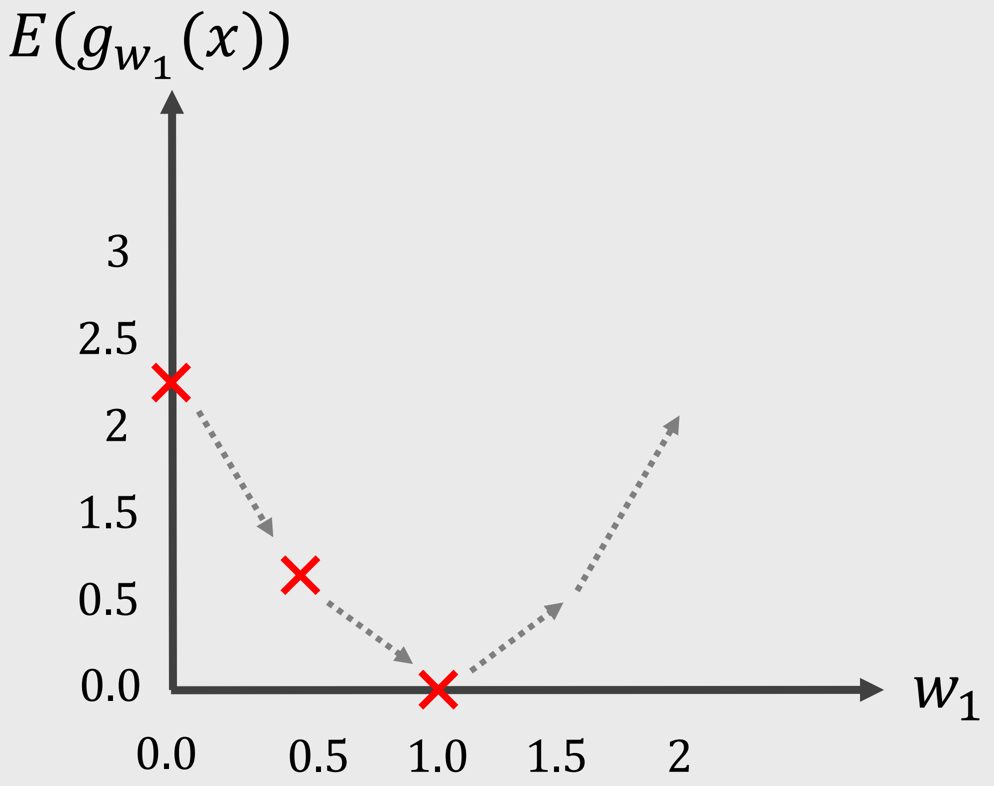
Error $E(w_1)$ for $w_0 = 0$ and $w_1 = 0.5$:

$$E(g_{\mathbf{w}}(\mathbf{x})) = \frac{(1-0.5)^2 + (2-1)^2 + (3-1.5)^2}{2*3} = 0.58$$

Does error $E(h_t(\mathcal{D}))$ minimization work?

9:59 PM





Hypothesis h_t for $w_0 = 0$ and $w_1 = 1$:

$$g(x_i) = w_0 + w_1 x_i$$

Error $E(w_1)$ for $w_0 = 0$ and $w_1 = 1$:

$$E(g_{\mathbf{w}}(\mathbf{x})) = \frac{(1-1)^2 + (2-2)^2 + (3-3)^2}{2*3} = 0.0$$

Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 6/10: Learning

Part 3
Learning Algorithms

DR VARUN OJHA

Department of Computer Science

University of Reading

Optimizer : Gradient Descent

Function g of the hypothesis has parameter $\mathbf{w}$:

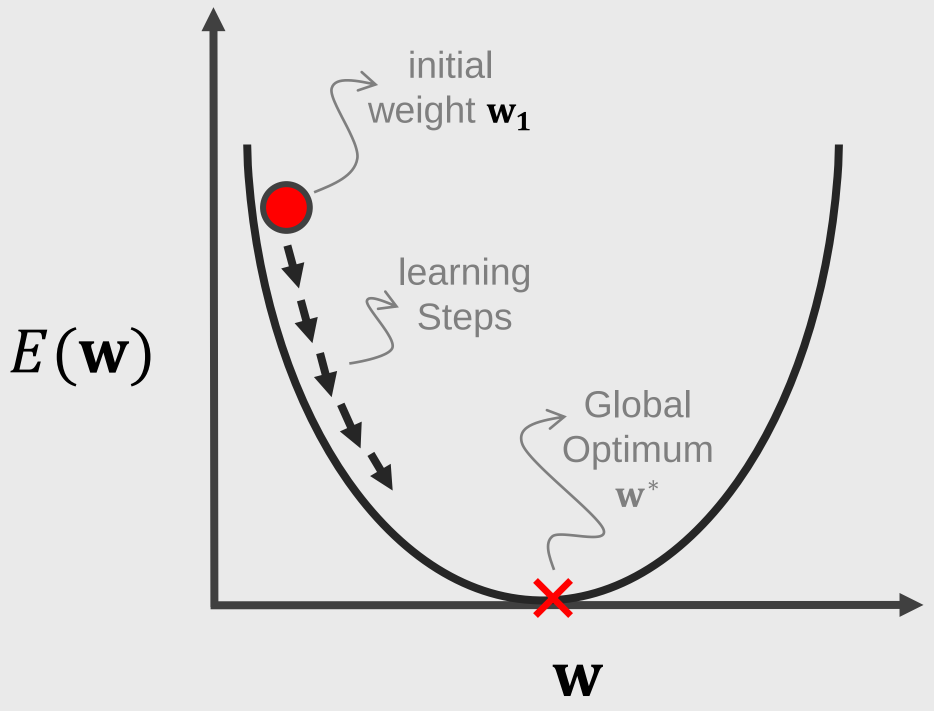
$$g_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{d} w_i\, x_i = 0$$

Gradient Descent Algorithm:

Repeat parameter $\mathbf{w}$ update for $t = 2, 3, \ldots, M$.

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \eta \frac{\partial E\left(g_{\mathbf{w}}(\mathbf{x})\right)}{\partial\, \mathbf{w}_t}\mathbf{x} \text{ for learning rate } \eta$$

Until error rate $E\left(g_{\mathbf{w}}(\mathbf{x})\right)$ is acceptable.



Optimizer : Gradient Descent

Function g of the hypothesis has parameter $\mathbf{w}$:

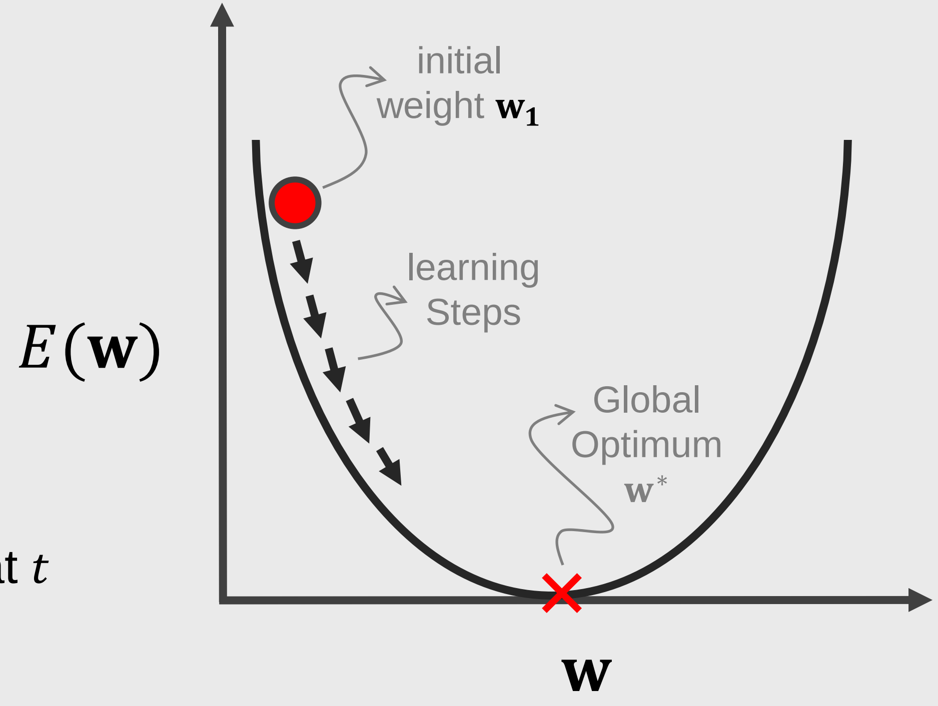
$$g_{\mathbf{w}}(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i = 0$$

Gradient Descent Algorithm:

Repeat parameter $\mathbf{w}$ update for $t = 2, 3, \ldots, M$.

$\mathbf{w}_t = \mathbf{w}_{t-1} + \Delta\, \mathbf{w}_t$ **,** where Δ is weight change (step) at t

Until error rate $E\big(g_{\mathbf{w}}(\mathbf{x})\big)$ is acceptable.



Gradient Descent: Versions

9:59 PM

Stochastic Gradient Descent

$t = 0$

$\mathbf{w}$ initial weights

for t in epochs **do**

$\mathcal{D} \leftarrow shuffle(\mathcal{D})$

for $\mathbf{x}_j \in \mathcal{D}$ **do** // for each sample

$\nabla \mathbf{w}_j = \partial E(g_{\mathbf{w}_t}(\mathbf{x}_j))/(\partial \mathbf{w}_t)$ // gradient of error *with respect to* weight $\mathbf{w}_j$

$\mathbf{w}_j = \mathbf{w}_{j-1} + \boldsymbol{\eta} \nabla \mathbf{w}_j \mathbf{x}_j$

$t = t + 1$

Batch Gradient Descent

$t = 0$

$\mathbf{w}$ initial weights

for t in epochs **do**

$\mathcal{D} \leftarrow shuffle(\mathcal{D})$

for $\mathbf{x}_j \in \mathcal{D}$ **do** // for each sample

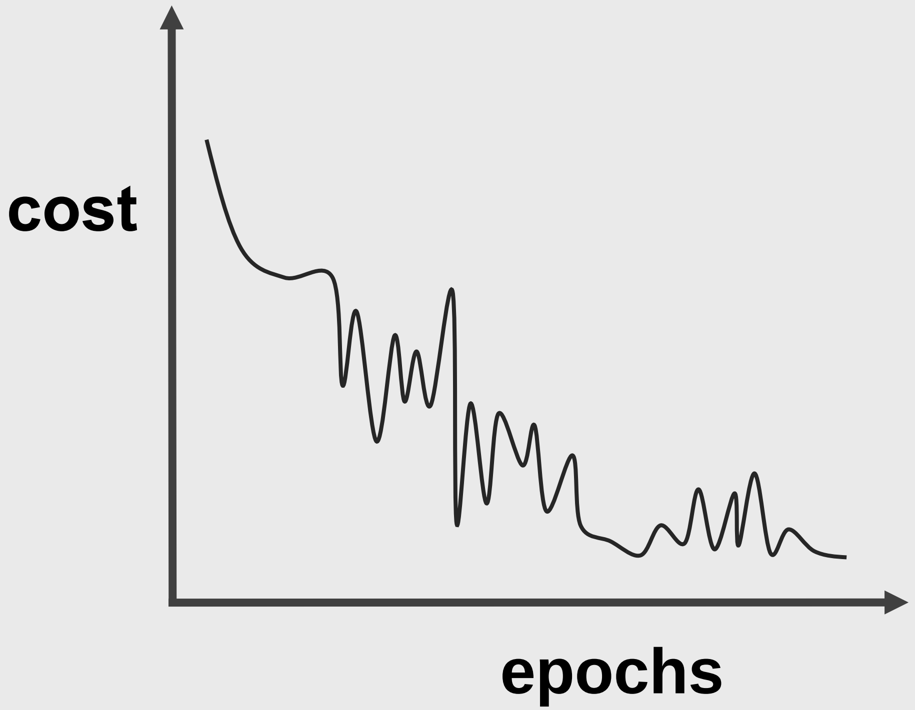
$\nabla \mathbf{w} = \nabla \mathbf{w} + \partial E(g_{\mathbf{w}}(\mathbf{x}_j))/(\partial \mathbf{w})\mathbf{x}_j$ // gradient of error *with respect to* weight $\mathbf{w}_j$

$\mathbf{w}_t = \mathbf{w}_{t-1} + \boldsymbol{\eta} \frac{\nabla \mathbf{w}}{|\mathcal{D}|}$

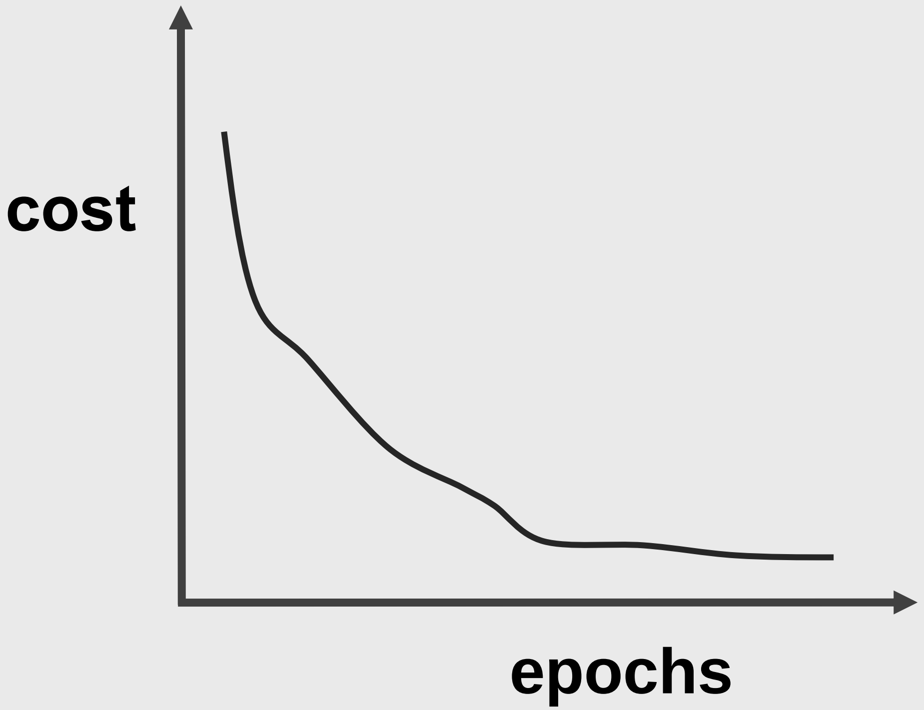
$t = t + 1$

Gradient Descent: Versions

Stochastic Gradient Descent

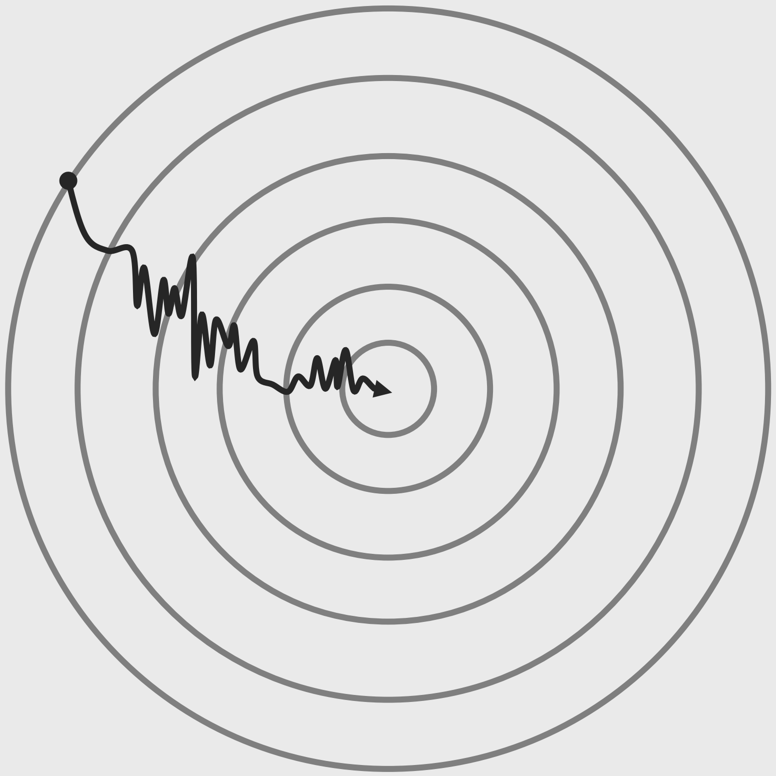


Batch Gradient Descent

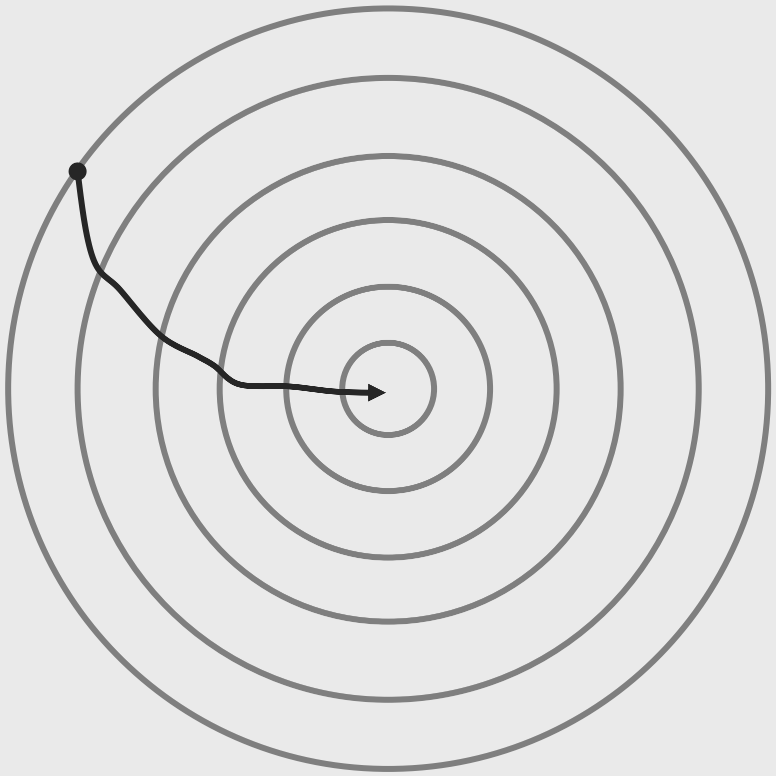


Gradient Descent: Versions

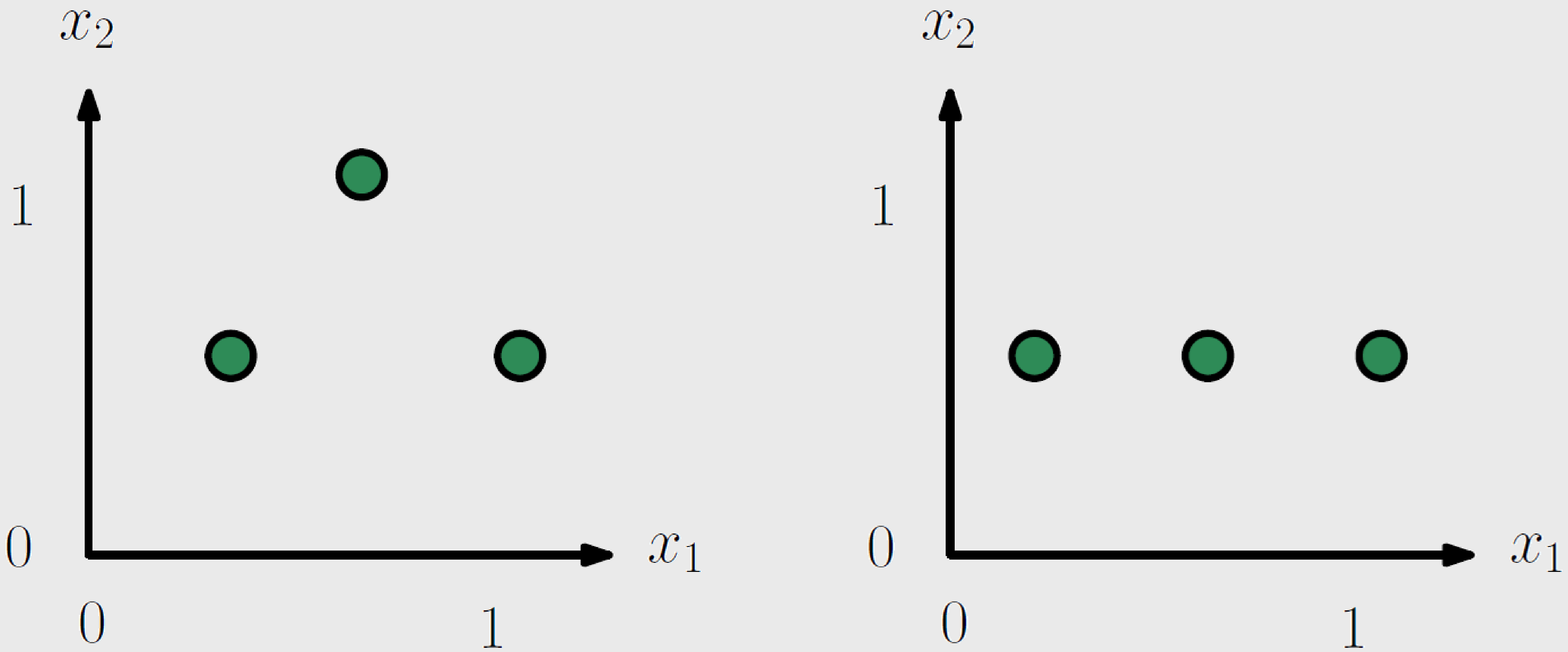
Stochastic Gradient Descent



Batch Gradient Descent

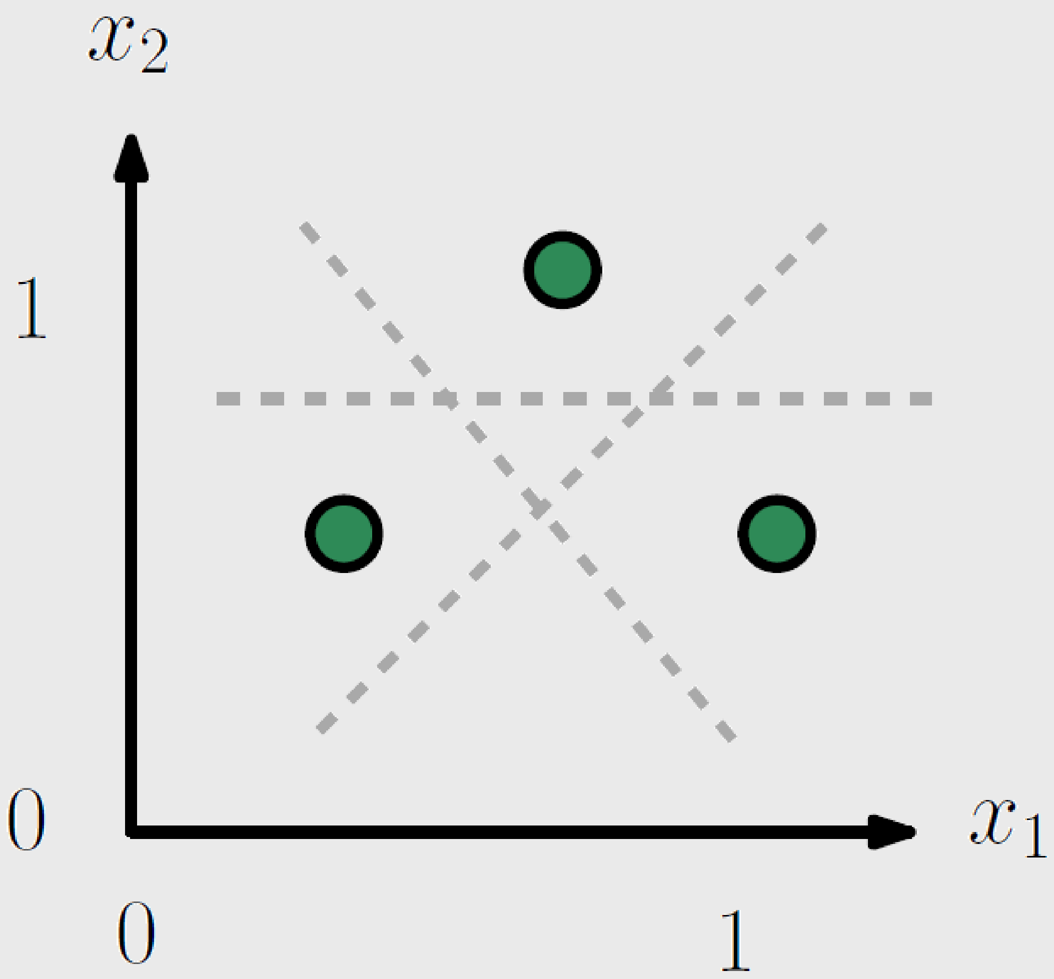


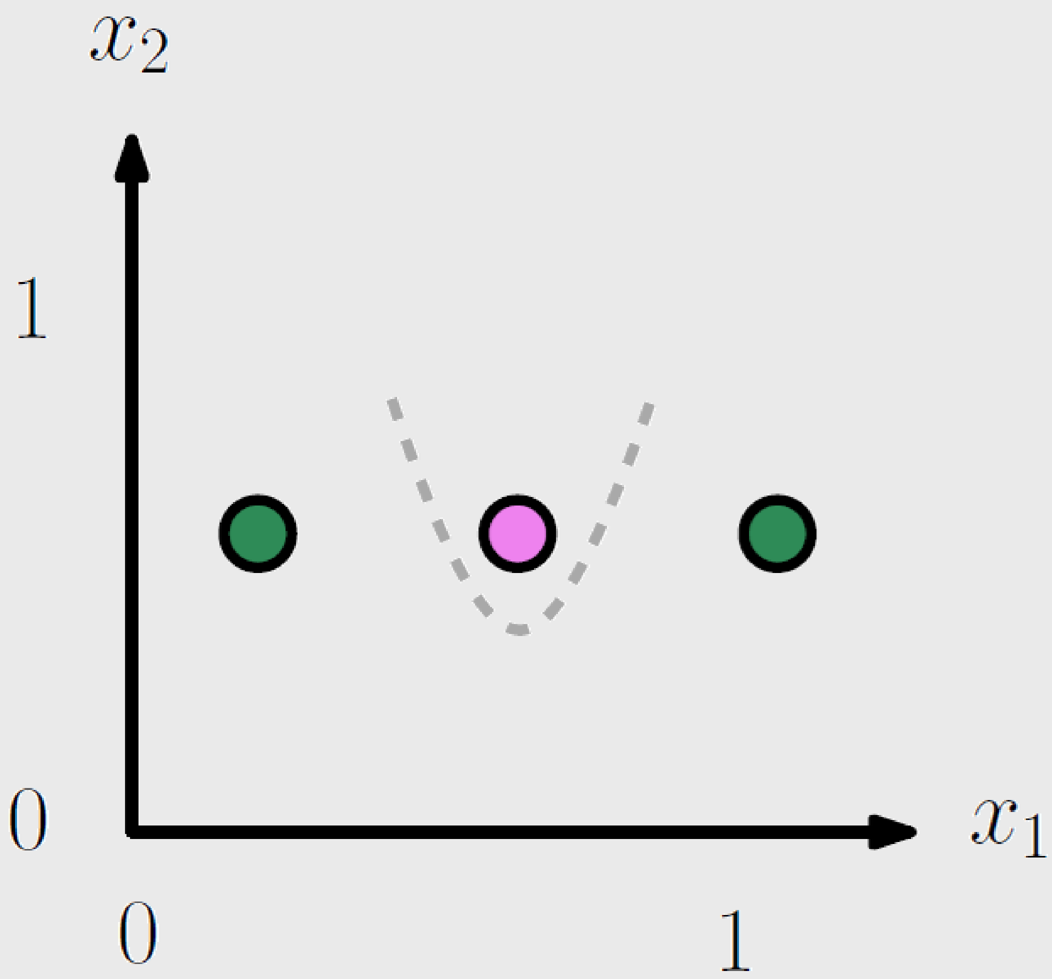
How do we choose a hypothesis class?



How do we choose a hypothesis class?

9:59 PM



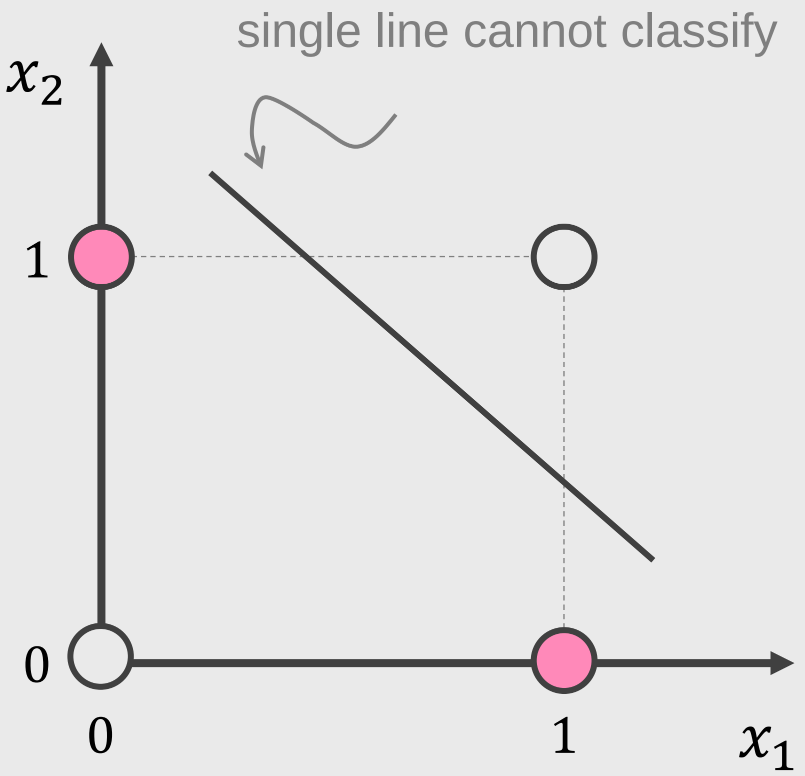


How to represent a hypothesis $h_t \in H$

Example Training Task: XOR Logic Problem

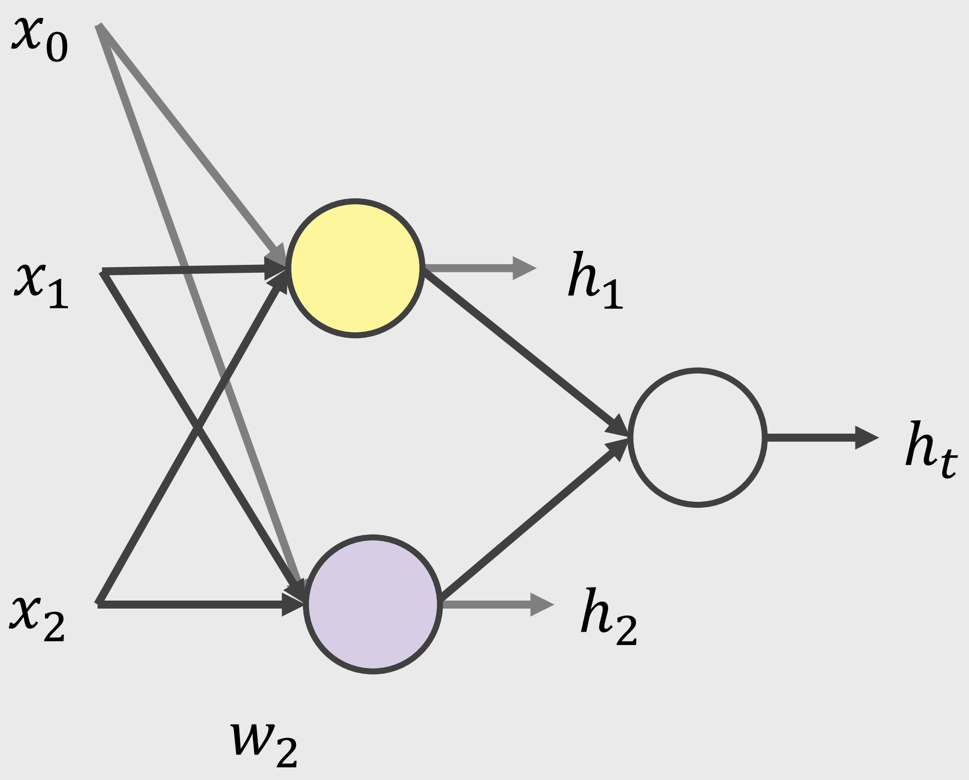
$\mathcal{D}$

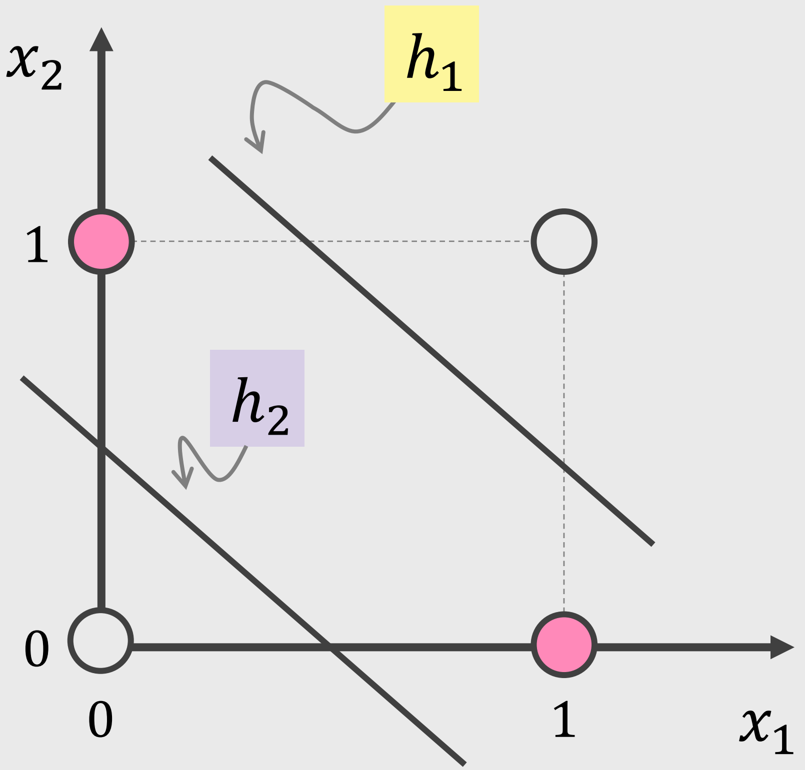
	x_1	x_2	y
1:	0	0	0
2:	0	1	1
3:	1	0	1
4:	1	1	0



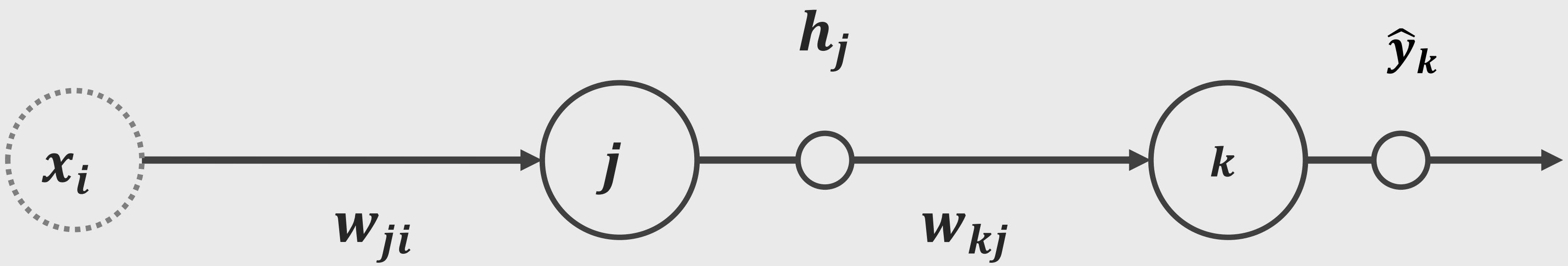
Which hypothesis $h_t \in H$ to pick?

How to evaluate a hypothesis: compute cost of hypothesis



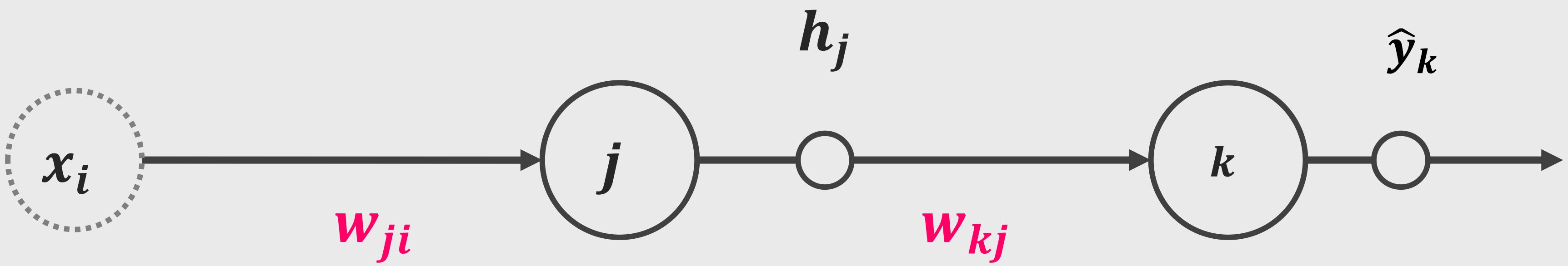


Backpropagation Algorithm



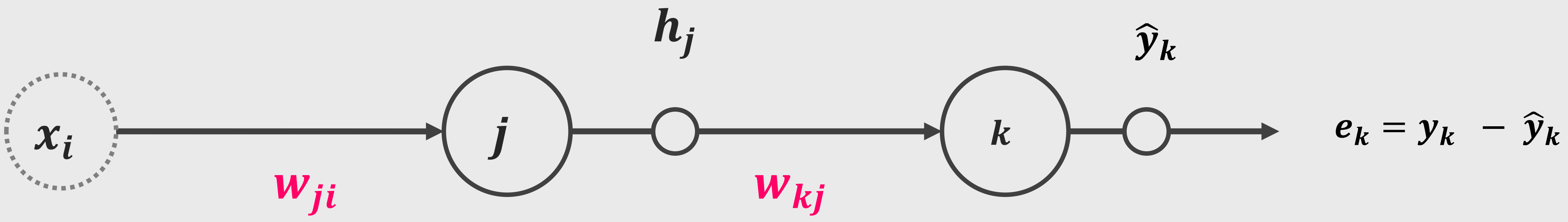
Backpropagation: **Forward Pass**

9:59 PM



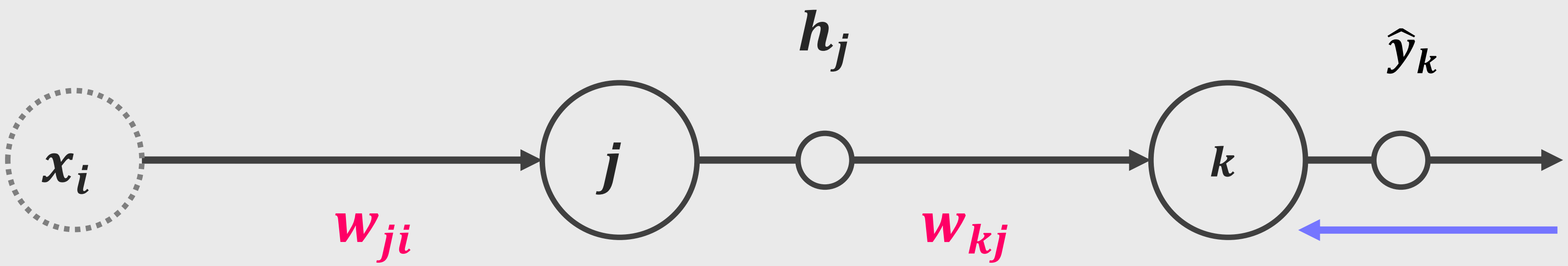
Backpropagation: **Error at Output layer**

9:59 PM



Backpropagation: **Backward pass**

Output layer delta (δ_k) considering sigmoidal output node(s)

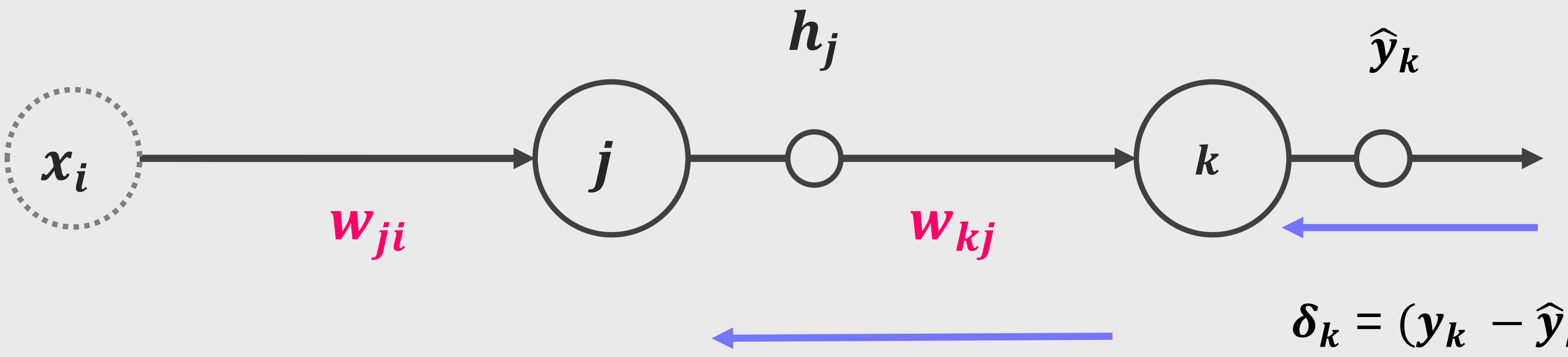


$$e_k = y_k - \hat{y}_k$$

$$\delta_k = (y_k - \hat{y}_k)\,\hat{y}_k\,(1 - \hat{y}_k)$$

Backpropagation: **Backward pass**

Hidden layer delta (δ_j) considering sigmoidal hidden node(s)



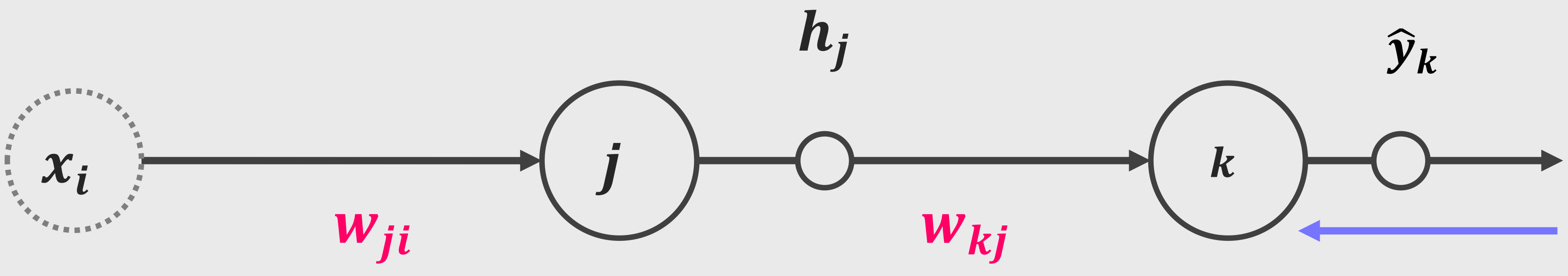
$$e_k = y_k - \hat{y}_k$$

$$\delta_k = (y_k - \hat{y}_k)\,\hat{y}_k\,(1 - \hat{y}_k)$$

$$\delta_j = h_j\,(1 - h_j)\,\sum_k \delta_k\, w_{kj}$$

Backpropagation: **Backward pass**

Hidden layer delta (δ_j) considering sigmoidal hidden node(s)



$$e_k = y_k - \hat{y}_k$$

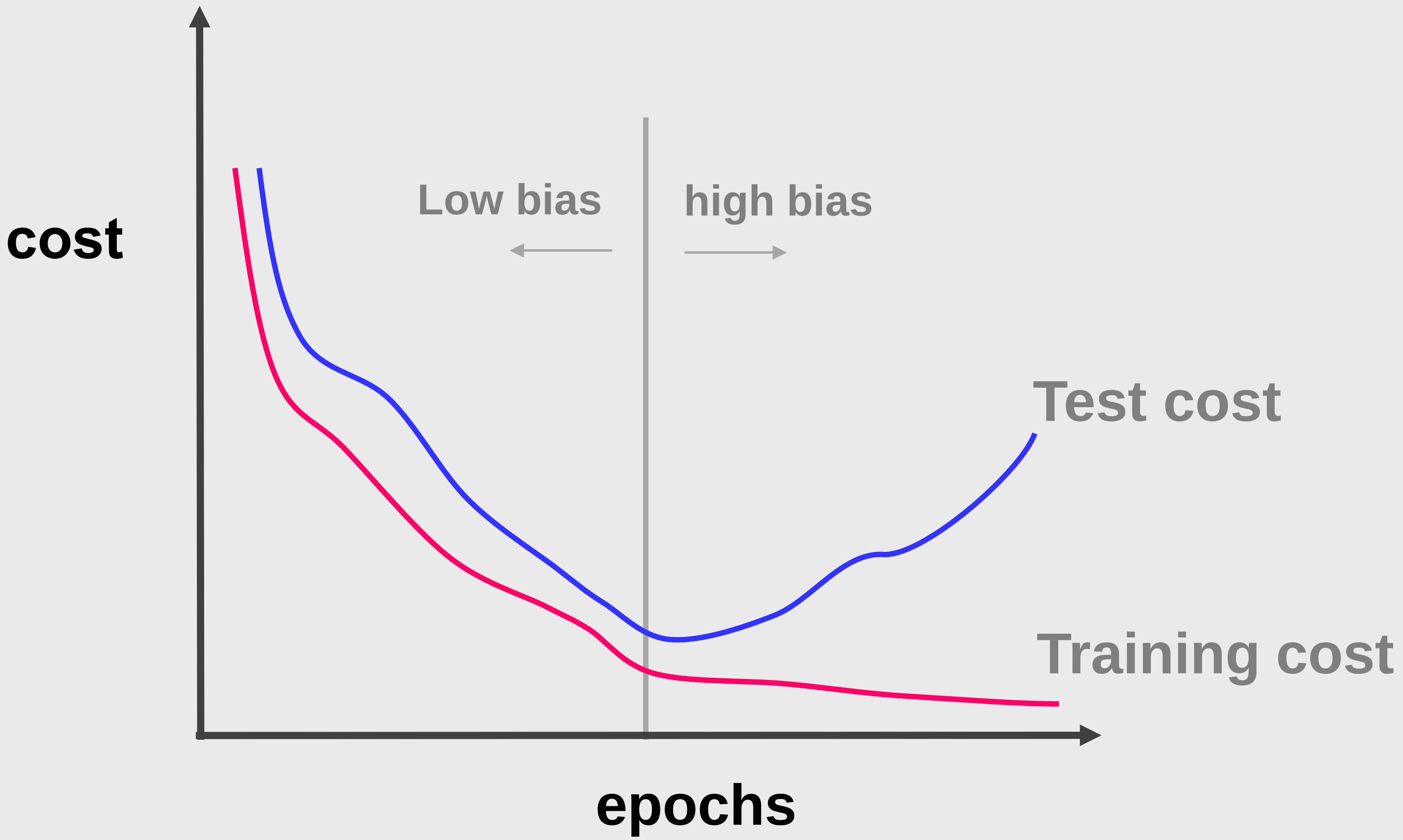
$$\delta_k = (y_k - \hat{y}_k)\,\hat{y}_k\,(1 - \hat{y}_k)$$

$$\Delta w_{kj} = \eta\, \delta_k\, h_j$$

$$\Delta w_{ji} = \eta\, \delta_j\, x_i$$

Training Method

Training Method



Bias-Variance Issue

Accuracy vary marginally on trials

low variance

Accuracy vary a lot on trials

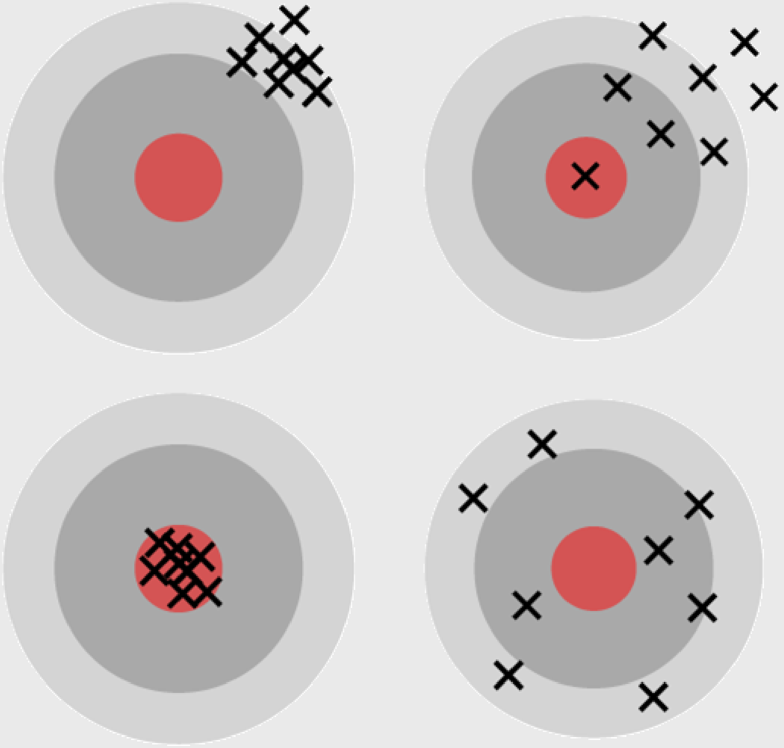
high variance

Accuracy is off by large margin

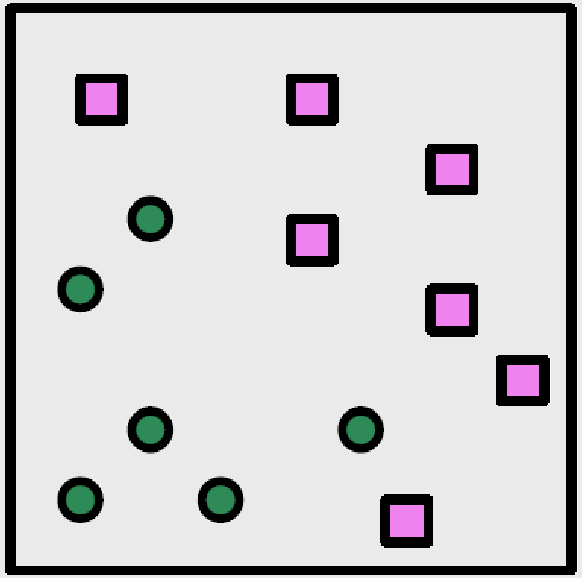
high bias

Accuracy is off by small margin

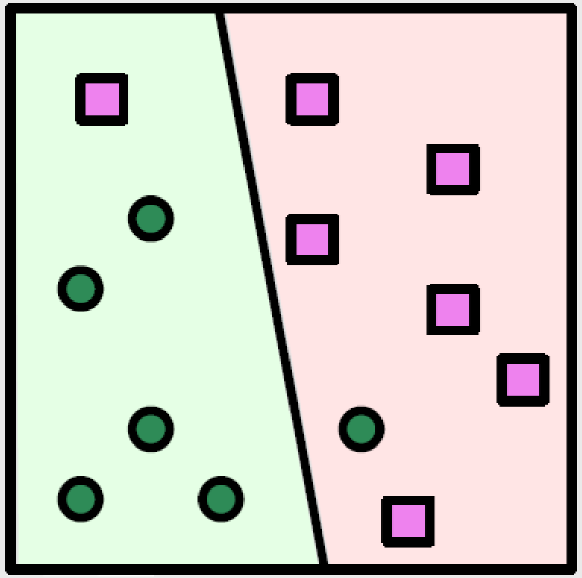
low bias



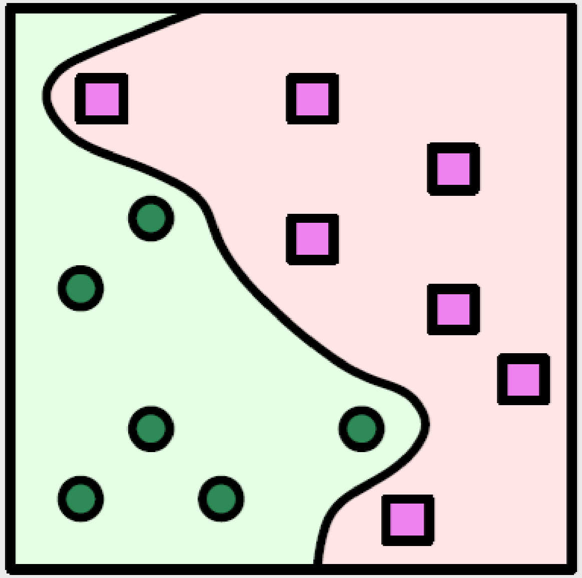
Is the chosen hypothesis good?



Training data



Underfit



Overfit

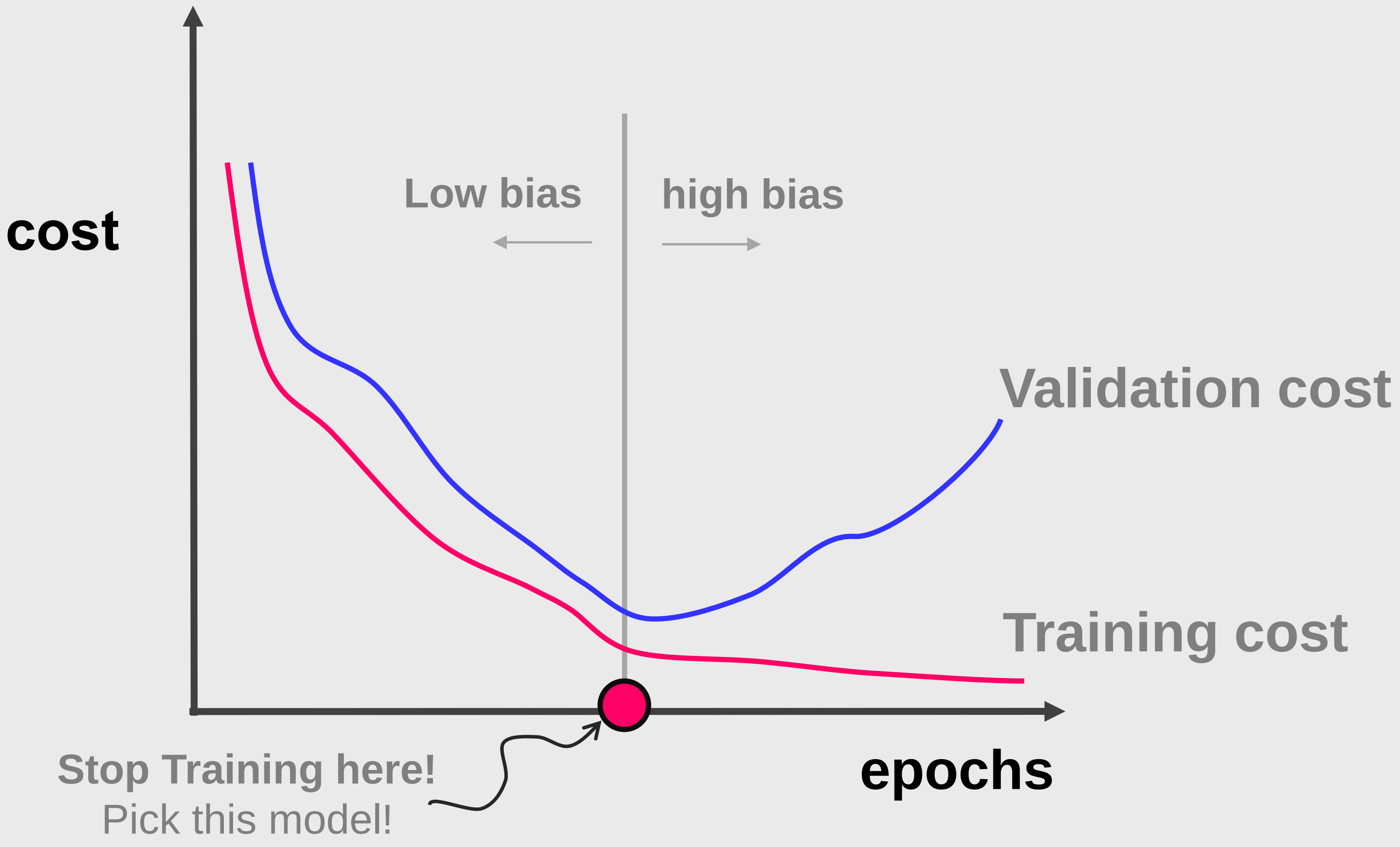
Avoid Overfitting

Training Set

Validation Set

Test Set

Training Method: Cross Validation



Artificial Intelligence

CS3AI18/ CSMAI19

Lecture - 6/10: Learning

Part 4
Practical Exercise

DR VARUN OJHA

Department of Computer Science

University of Reading