机器学习介绍
Introduction of Machine Learning

·AIPKU·

主讲人：王奕森

About Me

- **Assistant Professor, Ph.D. Advisor**
- **Research Interests: Machine Learning**

  - Trustworthy Machine Learning
  - Graph Neural Networks
  - Weakly/Self Supervised Learning

Homepage

https://yisenwang.github.io/

- **Welcome interns!**

中国计算机学会推荐国际学术会议

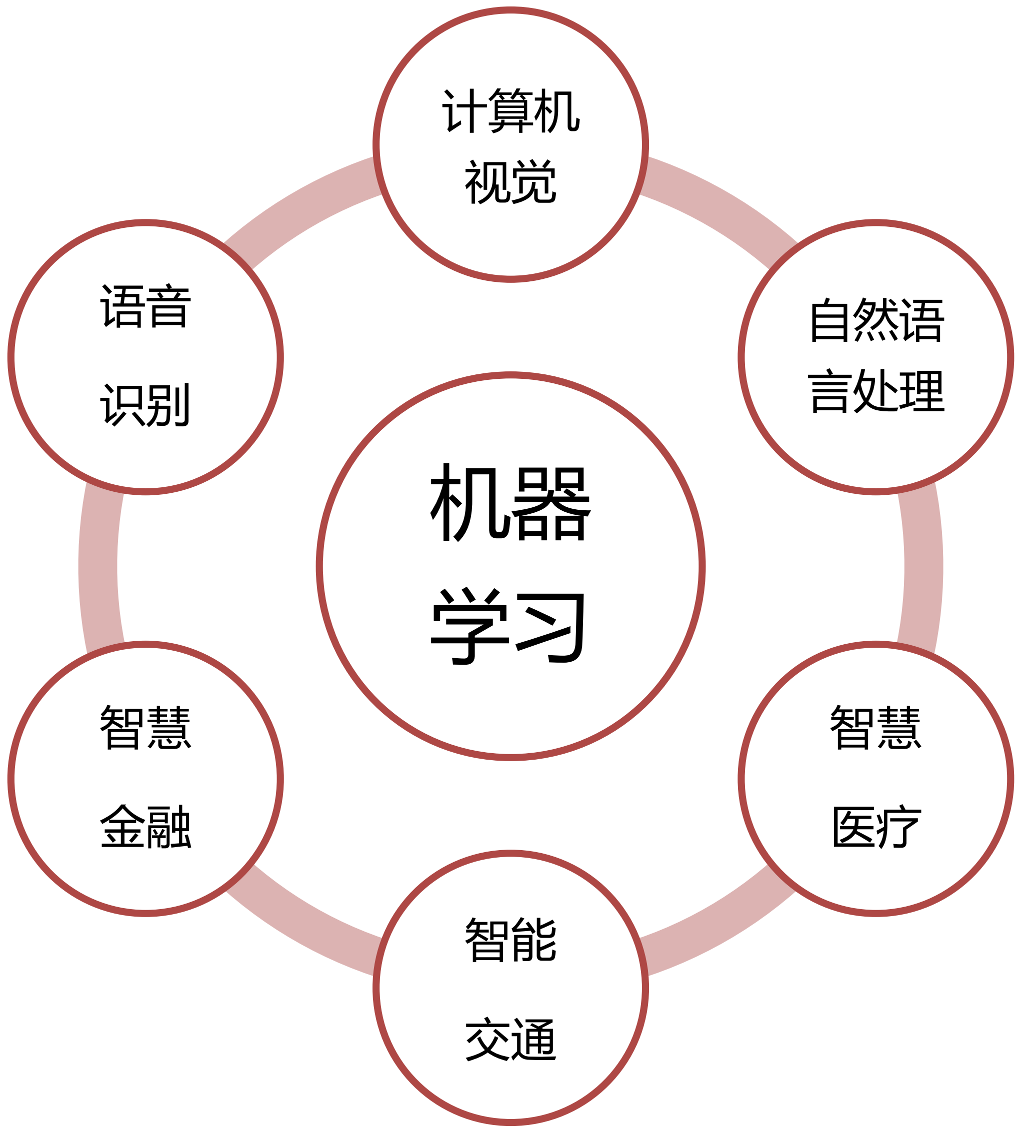
（人工智能）

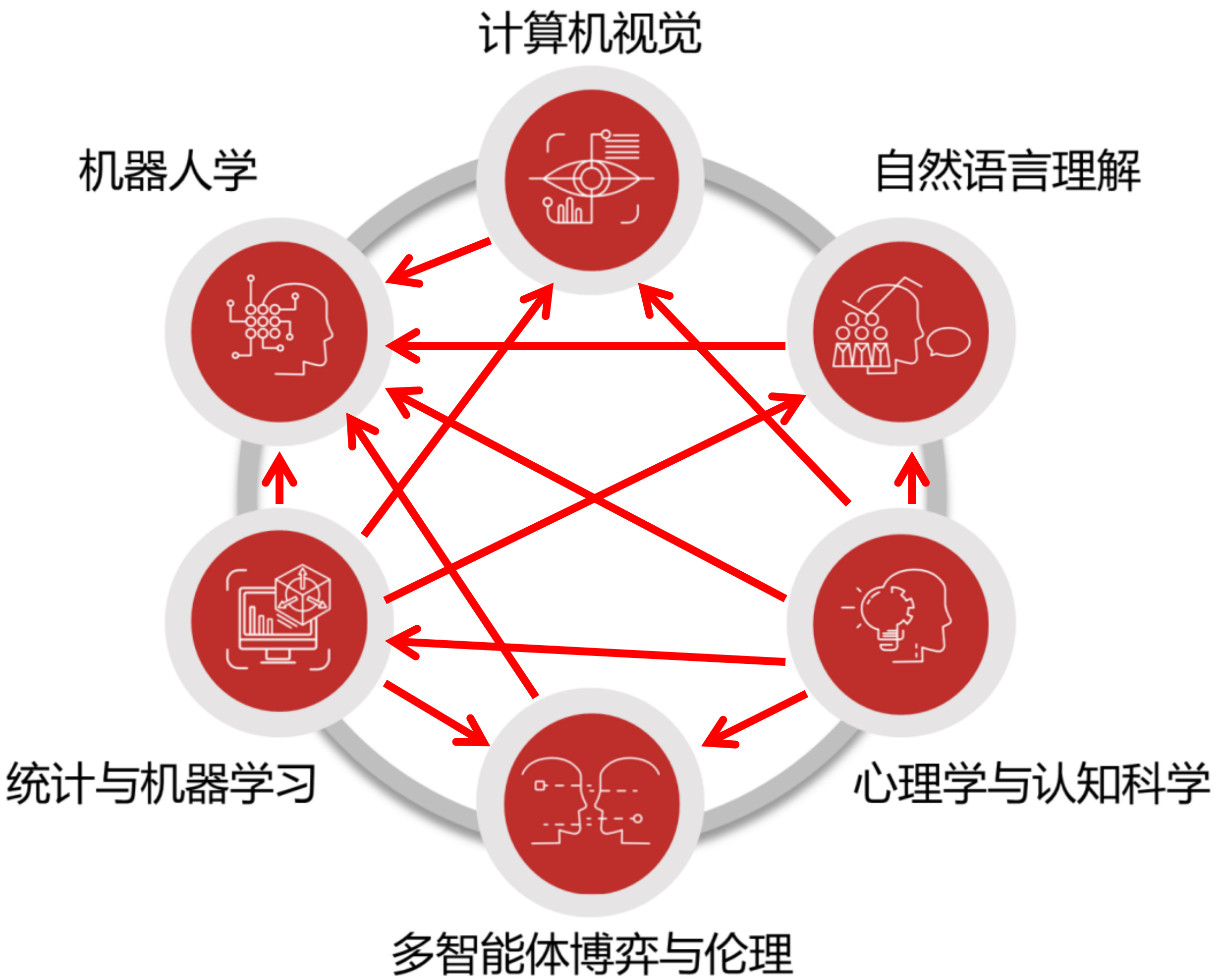
一、A 类

序号	会议简称	会议全称	出版社	网址
1	AAAI	AAAI Conference on Artificial Intelligence	AAAI	http://dblp.uni-trier.de/db/conf/aaai/
2	NeurIPS	Annual Conference on Neural Information Processing Systems	MIT Press	http://dblp.uni-trier.de/db/conf/nips/
3	ACL	Annual Meeting of the Association for Computational Linguistics	ACL	http://dblp.uni-trier.de/db/conf/acl/
4	CVPR	IEEE Conference on Computer Vision and Pattern Recognition	IEEE	http://dblp.uni-trier.de/db/conf/cvpr/
5	ICCV	International Conference on Computer Vision	IEEE	http://dblp.uni-trier.de/db/conf/iccv/
6	ICML	International Conference on Machine Learning	ACM	http://dblp.uni-trier.de/db/conf/icml/
7	IJCAI	International Joint Conference on Artificial Intelligence	Morgan Kaufmann	http://dblp.uni-trier.de/db/conf/ijcai/

ICLR, International Conference on Learning Representations

ML Is at The Core of AI & AI Class





What is Machine Learning?

- Machine Learning is for designing algorithms that can learn and build computable models from training data and then perform desired tasks on new data.

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measure by P, **improves** with experience E.

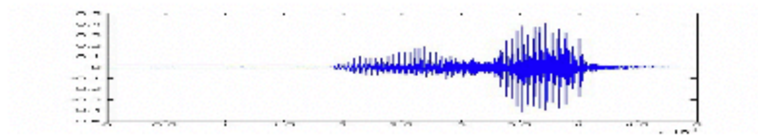
So ML is not rule based and in principle it can enable machines to **evolve**.

Tom Mitchell
(Member of NAE & AAAS, Prof. @ CMU)

Machine Learning

Machine Learning ≈ Looking for Function

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

- Playing Go

$f($  $)=$ “5-5” (next move)

Different types of Functions

Regression: The function outputs a scalar.

Predict PM2.5

PM2.5 today → f → PM2.5 of tomorrow
temperature → f
Concentration of O_3 → f

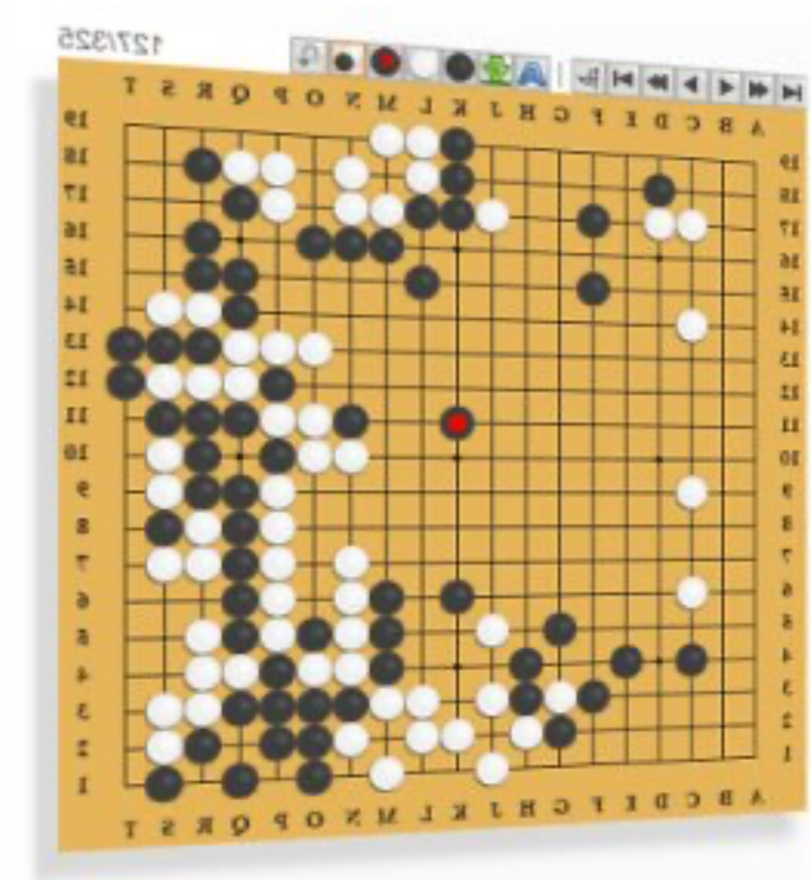
Classification: Given options (**classes**), the function outputs the correct one.

Spam filtering

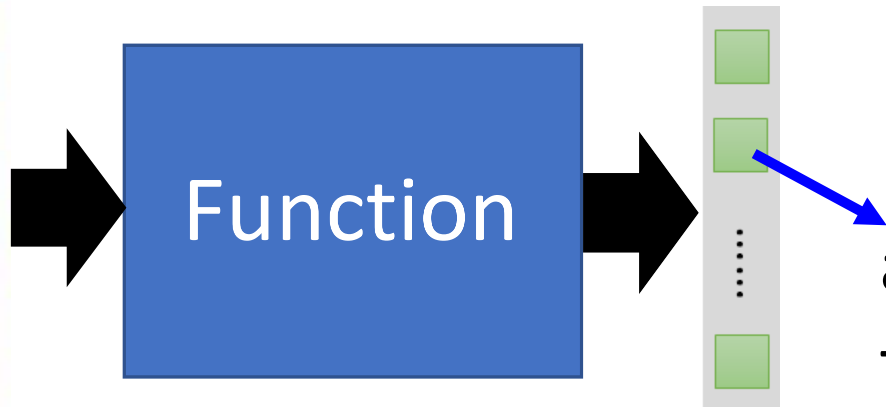
→ f → Yes/No

Different types of Functions

Classification: Given options (**classes**), the function outputs the correct one.



Playing GO



Each position is a class

(19 x 19 classes)

a position on the board

Next move

One example

Taking examples (some slides) from Prof. Hungyi Lee@NTU

$$y = f(\quad)$$

no. of views
on 2/26

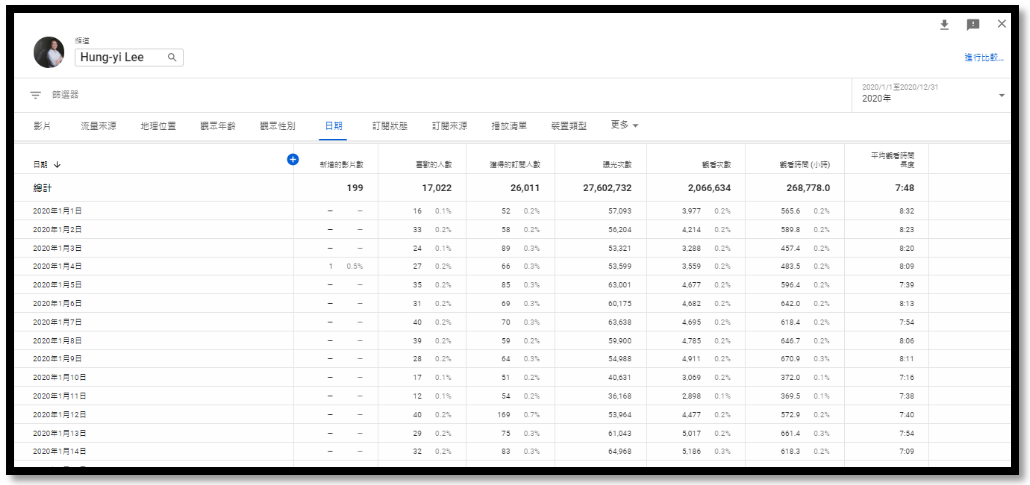
頻道
Hung-yi Lee

篩選器

影片 流量來源 地理位置 觀眾年齡 觀眾性別 日期 訂閱狀態 訂閱來源 播放清單

日期 ↓	喜歡的人數		訂閱人數		觀看次數	
2021年1月26日	54	4.9%	69	5.5%	6,788	5.2%
2021年1月27日	60	5.4%	71	5.6%	6,242	4.7%
2021年1月28日	36	3.2%	63	5.0%	5,868	4.5%
2021年1月29日	27	2.4%	40	3.2%	4,413	3.4%
2021年1月30日	40	3.6%	40	3.2%	4,372	3.3%
2021年1月31日	47	4.2%	51	4.0%	5,135	3.9%
2021年2月1日	61	5.5%	29	2.3%	5,527	4.2%
2021年2月2日	49	4.4%	43	3.4%	5,911	4.5%
2021年2月3日	26	2.3%	44	3.5%	5,248	4.0%
2021年2月4日	43	3.9%	33	2.6%	4,771	3.6%
2021年2月5日	45	4.0%	49	3.9%	3,850	2.9%
2021年2月6日	29	2.6%	42	3.3%	3,828	2.9%
2021年2月7日	26	2.3%	46	3.6%	4,559	3.5%
2021年2月8日	38	3.4%	26	2.1%	4,772	3.6%
2021年2月9日	29	2.6%	25	2.0%	3,847	2.9%
2021年2月10日	31	2.8%	35	2.8%	3,382	2.6%

1) Function

$y = f($  $)$

Model $y = b + wx_1$ based on domain knowledge

feature

y: no. of views on 2/26, x_1: no. of views on 2/25

w and b are unknown parameters (learned from data)

weight **bias**

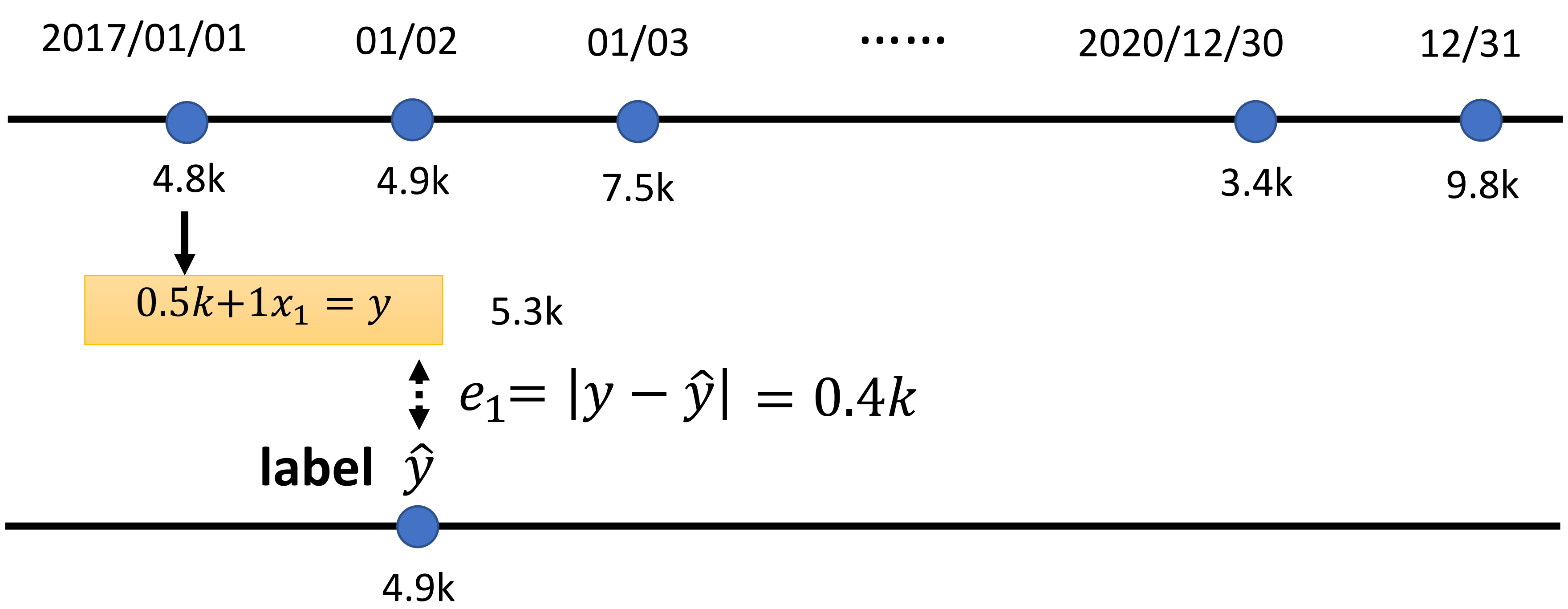
2) Loss

Defining Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

$L(0.5k, 1)$ $y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

Data from 2017/01/01 – 2020/12/31



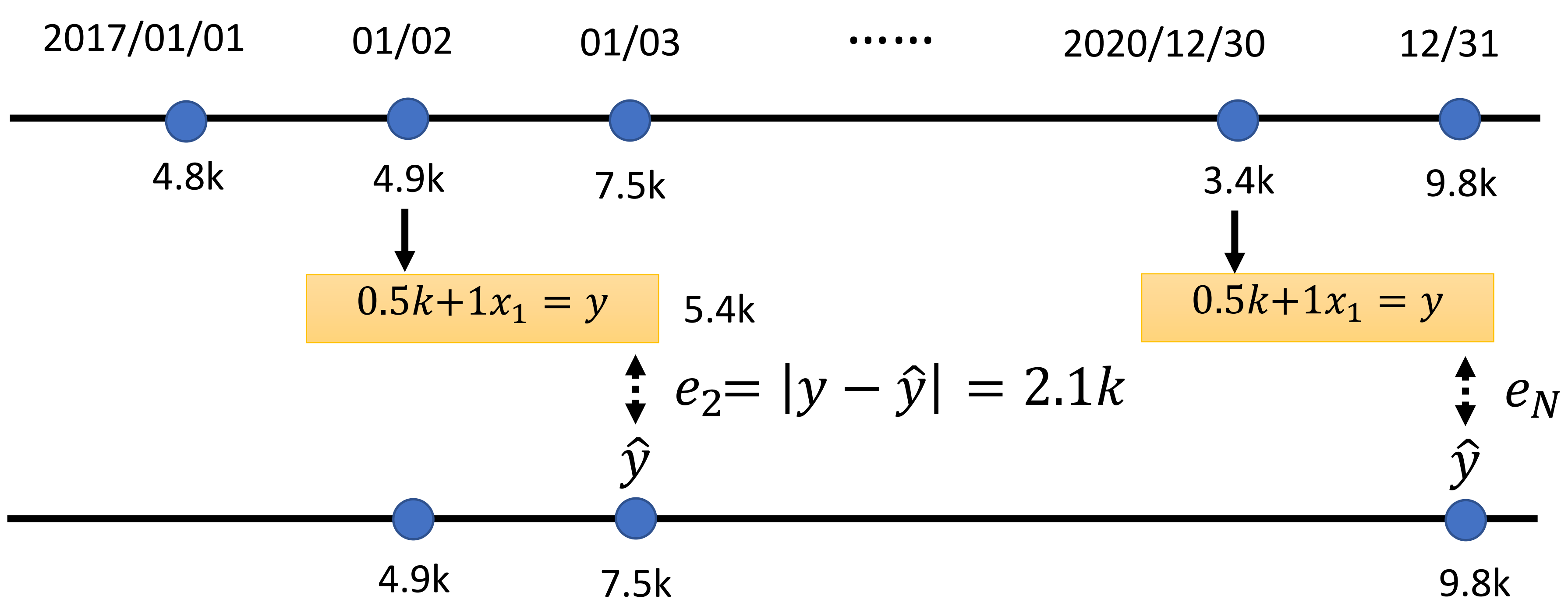
2) Loss

Defining Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

$L(0.5k, 1)$ $y = b + wx_1$ $\longrightarrow$ $y = 0.5k + 1x_1$ How good it is?

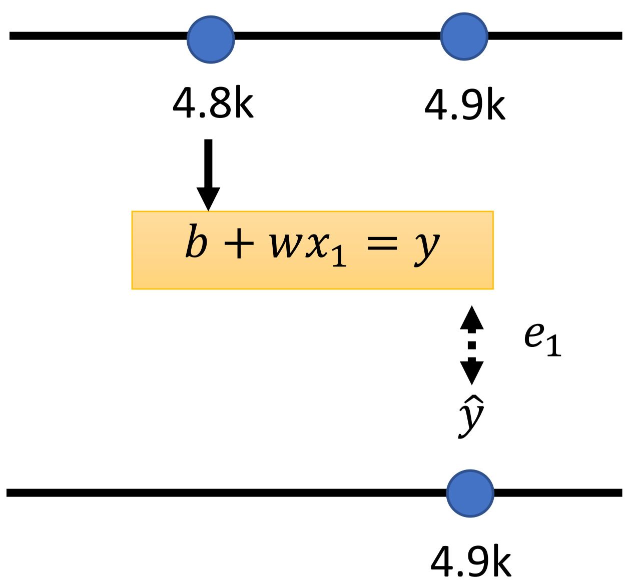
Data from 2017/01/01 – 2020/12/31



2) Loss

Defining Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.



Loss: $$L = \frac{1}{N}\sum_n e_n$$

$e = |y - \hat{y}|$ L is mean absolute error (MAE)

$e = (y - \hat{y})^2$ L is mean square error (MSE)

If y and $\hat{y}$ are both probability distributions → Cross-entropy

$$e = y \log \hat{y}$$

3) Optimization

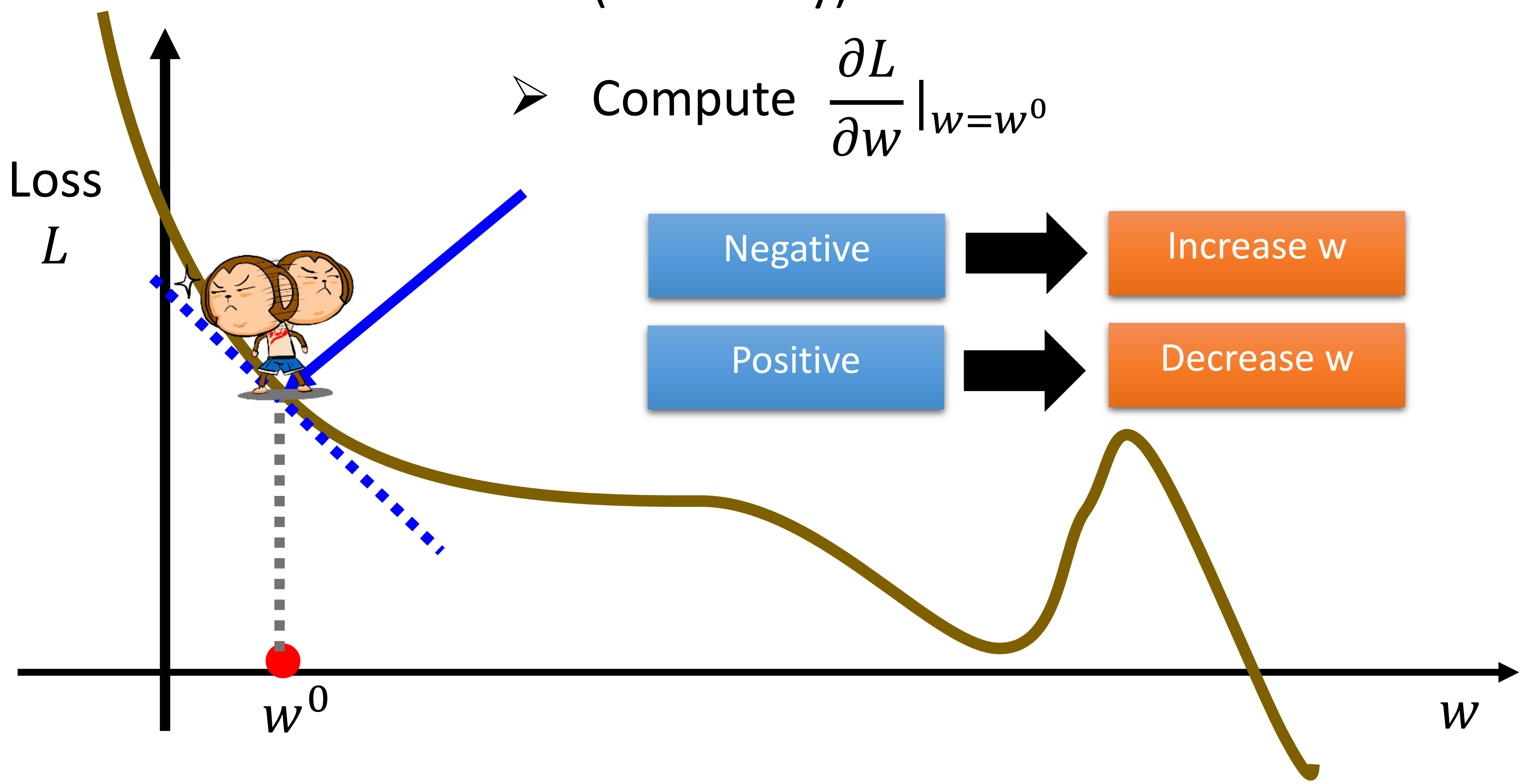
w^* [redacted] $= arg \min_{w}$ [redacted] L

Gradient Descent

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

Negative → Increase w

Positive → Decrease w



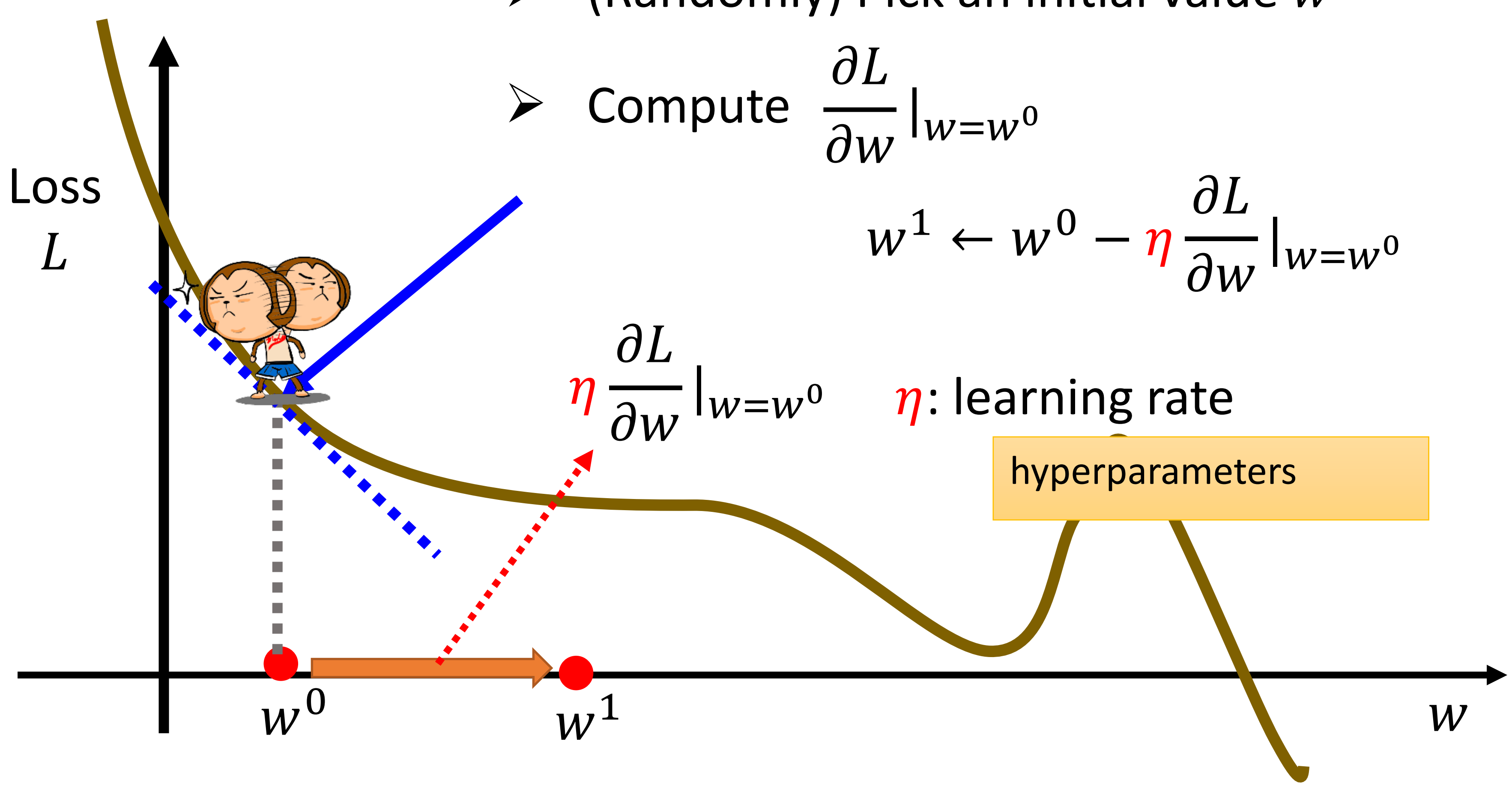
3) Optimization

$$w^* \text{[redacted]} = arg \min_{w \text{[redacted]}} L$$

Gradient Descent

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0}$$



η: learning rate

hyperparameters

3) Optimization

$$w^* \text{[redacted]} = arg \min_{w \text{[redacted]}} L$$

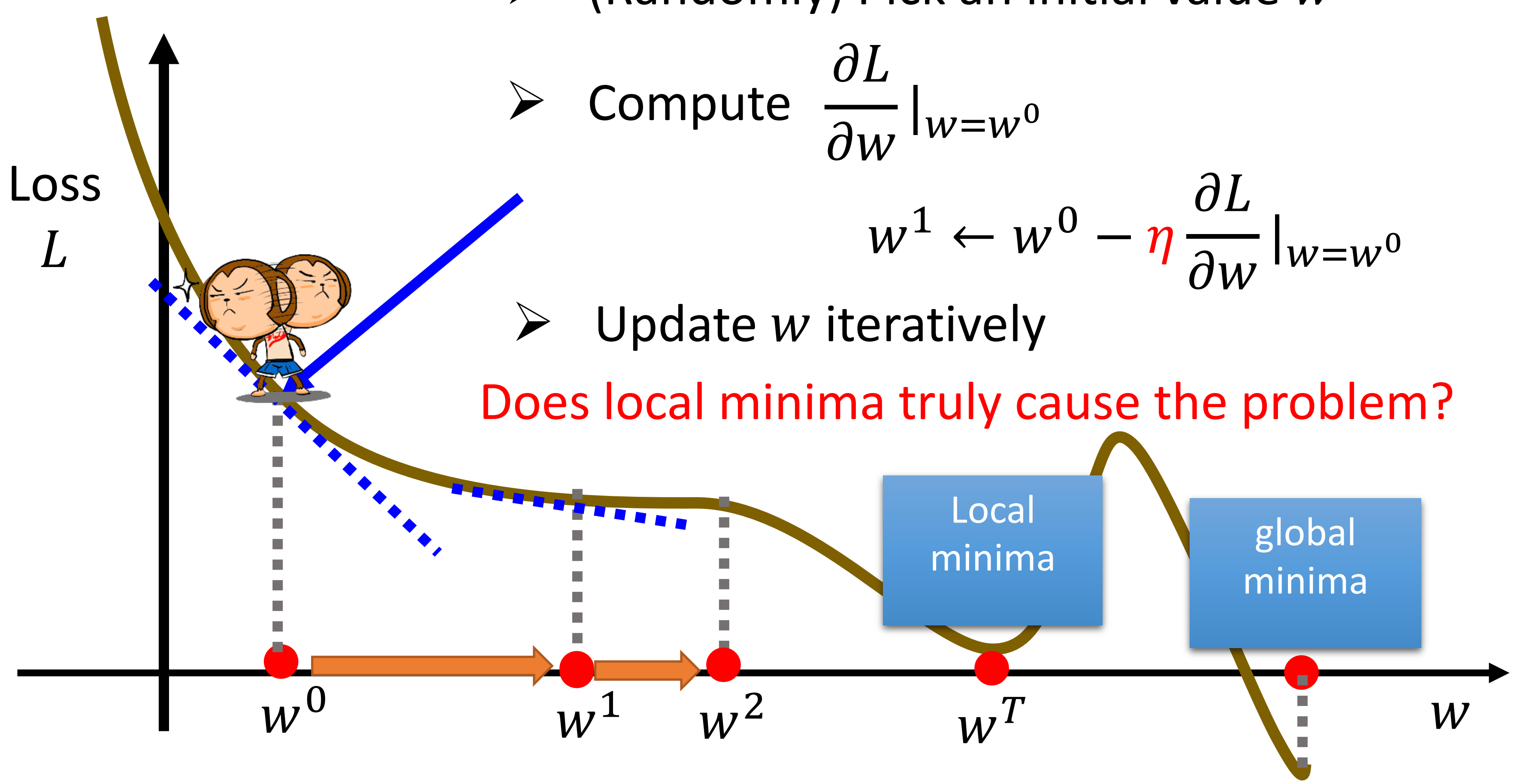
Gradient Descent

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0}$$

- Update w iteratively

Does local minima truly cause the problem?



3) Optimization

$$w^*, b^* = arg \min_{w,b} L$$

- (Randomly) Pick initial values w^0, b^0
- Compute

$$\frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$$

$$\frac{\partial L}{\partial b}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

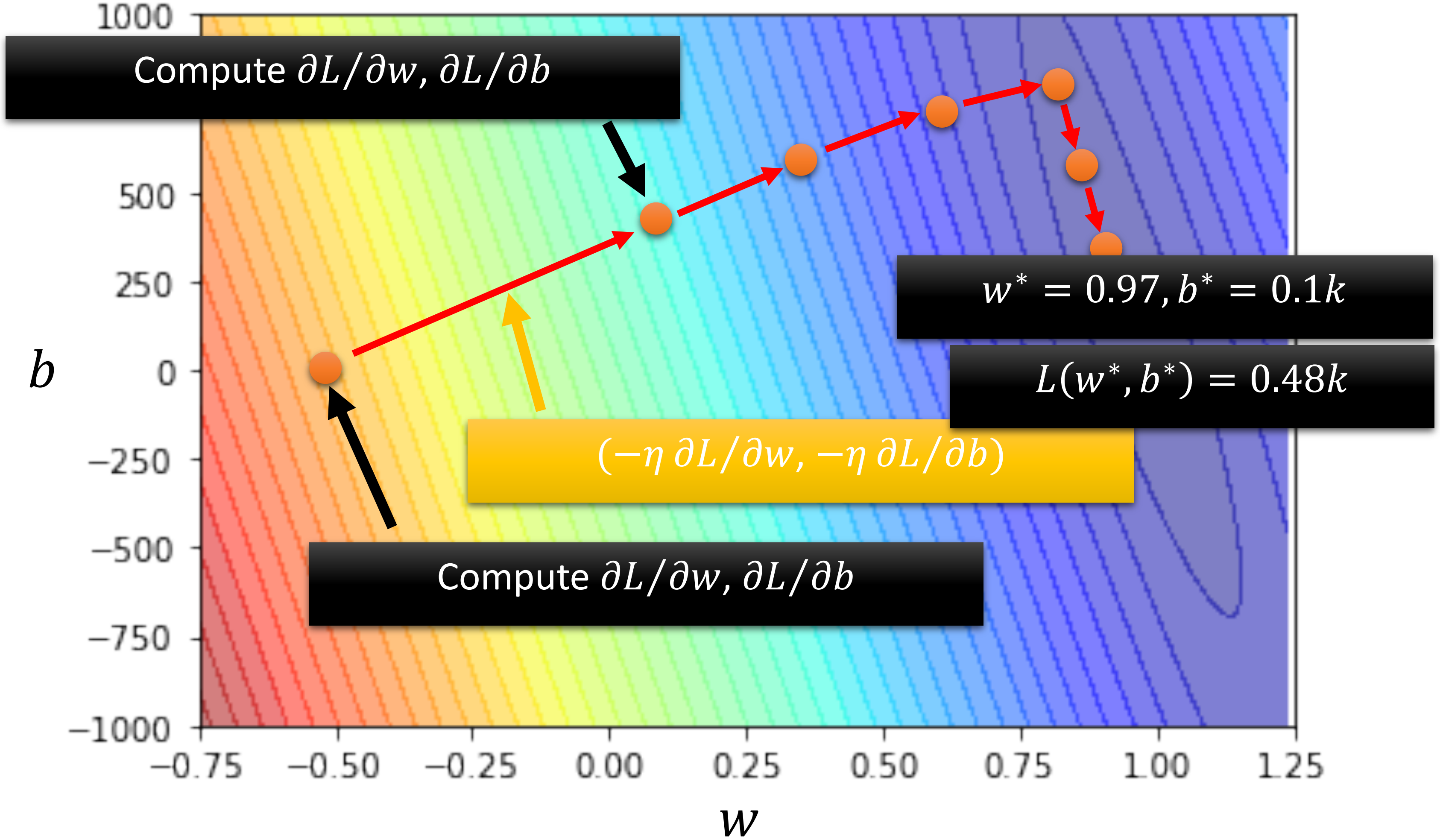
Can be done in one line in most deep learning frameworks

- Update w and b interatively

3) Optimization

Model $y = b + wx_1$

$$w^*, b^* = arg \min_{w,b} L$$



Pipeline Summary

$y = b + wx_1$

Step 1: function with unknown

Step 2: define loss from training data

$w^* = 0.97, b^* = 0.1k$

$L(w^*, b^*) = 0.48k$

Step 3: optimization

Machine Learning Pipeline

$w^* = 0.97, b^* = 0.1k$

$L(w^*, b^*) = 0.48k$

$y = b + wx_1$

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

Training

$y = 0.1k + 0.97x_1$ achieves the smallest loss $L = 0.48k$ on data of 2017 – 2020 (**training data**)

How about data of 2021 (**unseen during training**)?

$L' = 0.58k$

More Inputs

$y = b + wx_1$

2017 - 2020 $L = 0.48k$

2021 $L' = 0.58k$

$y = b + \sum_{j=1}^{7} w_j x_j$

2017 - 2020 $L = 0.38k$

2021 $L' = 0.49k$

b	w_1^*	w_2^*	w_3^*	w_4^*	w_5^*	w_6^*	w_7^*
0.05k	0.79	-0.31	0.12	-0.01	-0.10	0.30	0.18

$y = b + \sum_{j=1}^{28} w_j x_j$

2017 - 2020 $L = 0.33k$

2021 $L' = 0.46k$

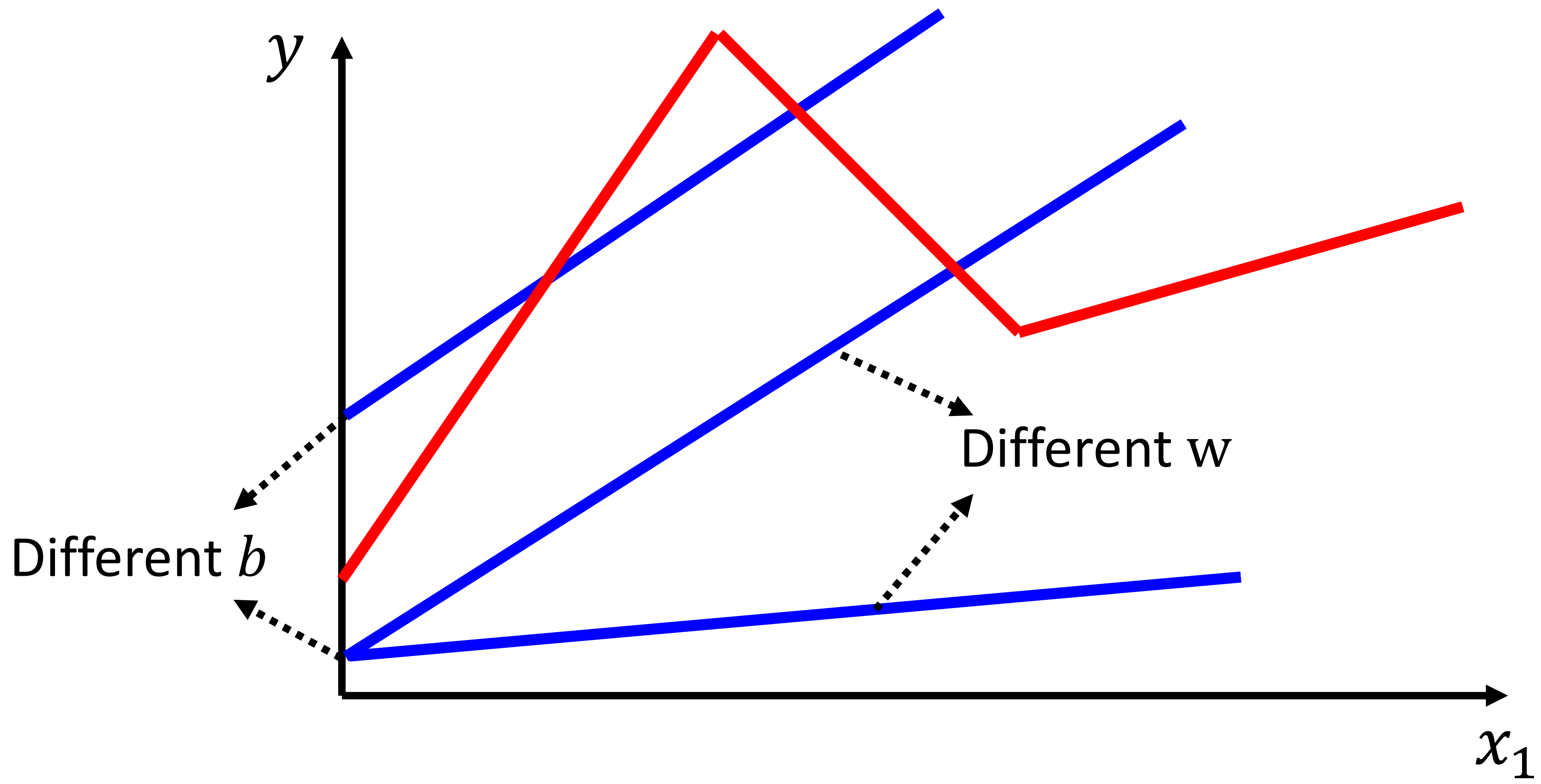
$y = b + \sum_{j=1}^{56} w_j x_j$

2017 - 2020 $L = 0.32k$

2021 $L' = 0.46k$

Linear models

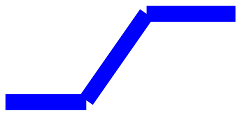
Linear models are always linear

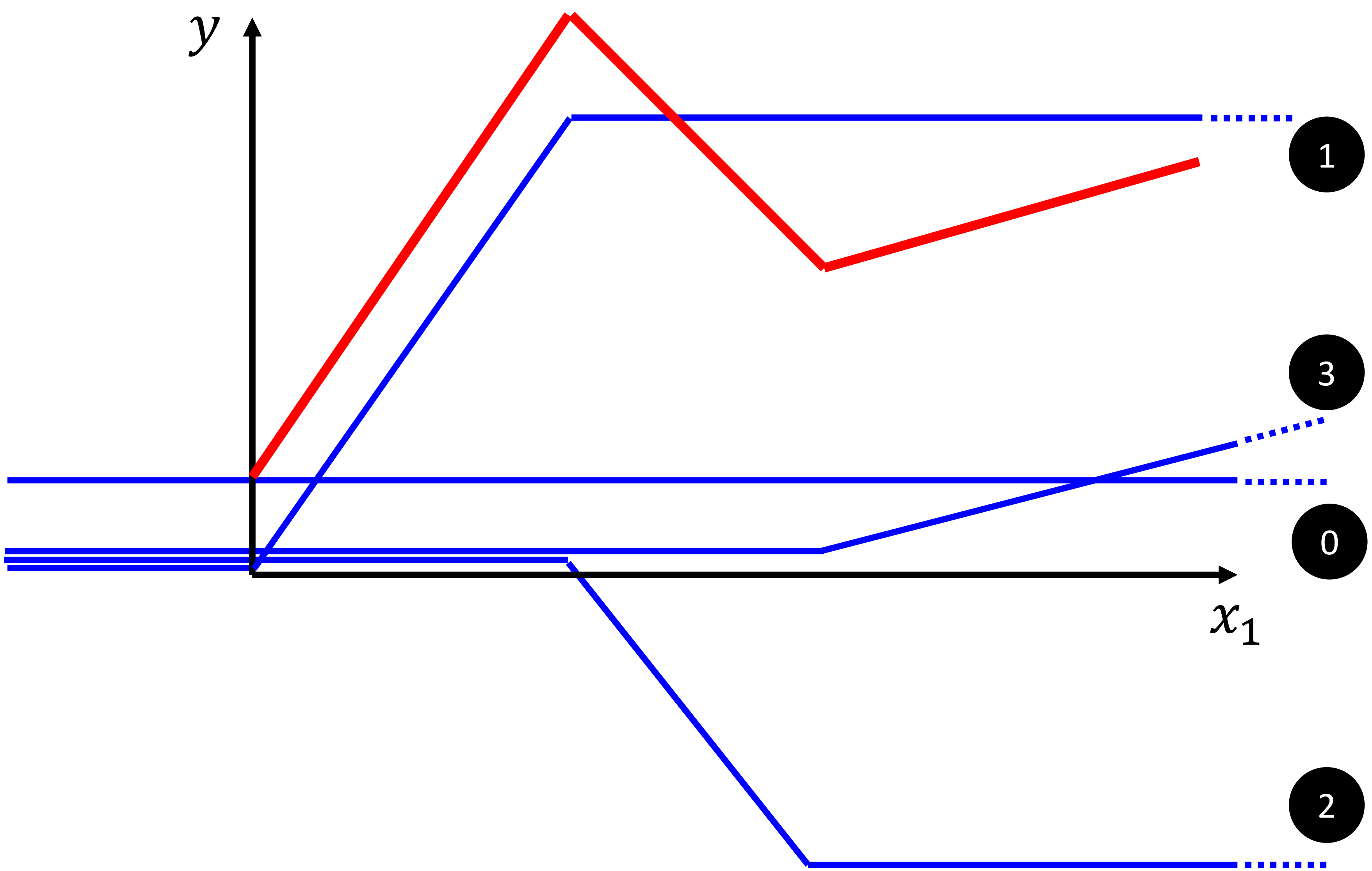


Linear models have severe limitation. ***Model Bias***

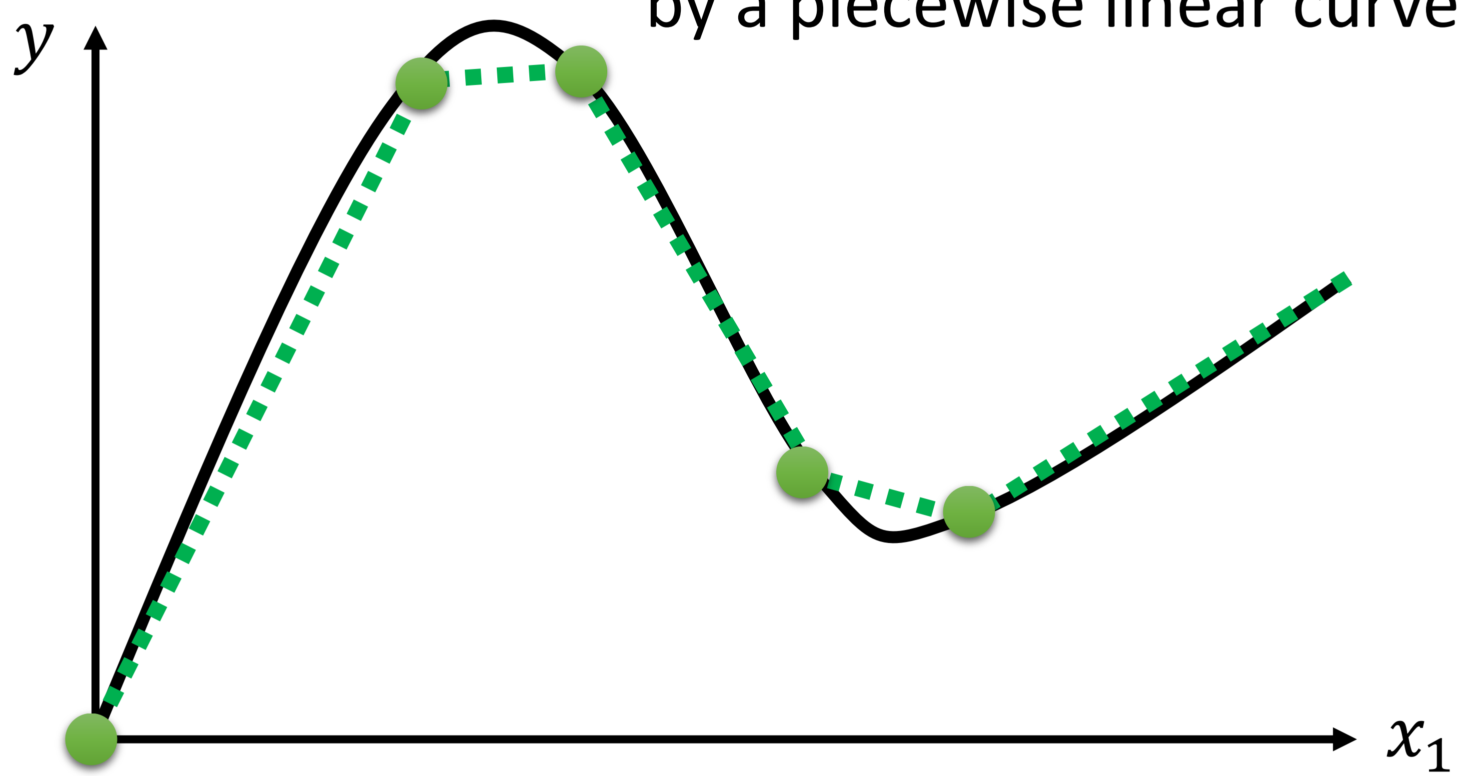
We need a more flexible model!

Piecewise Linear Curves

red curve = constant + sum of a set of 

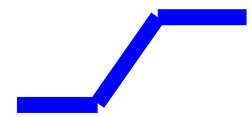


Beyond Piecewise Linear

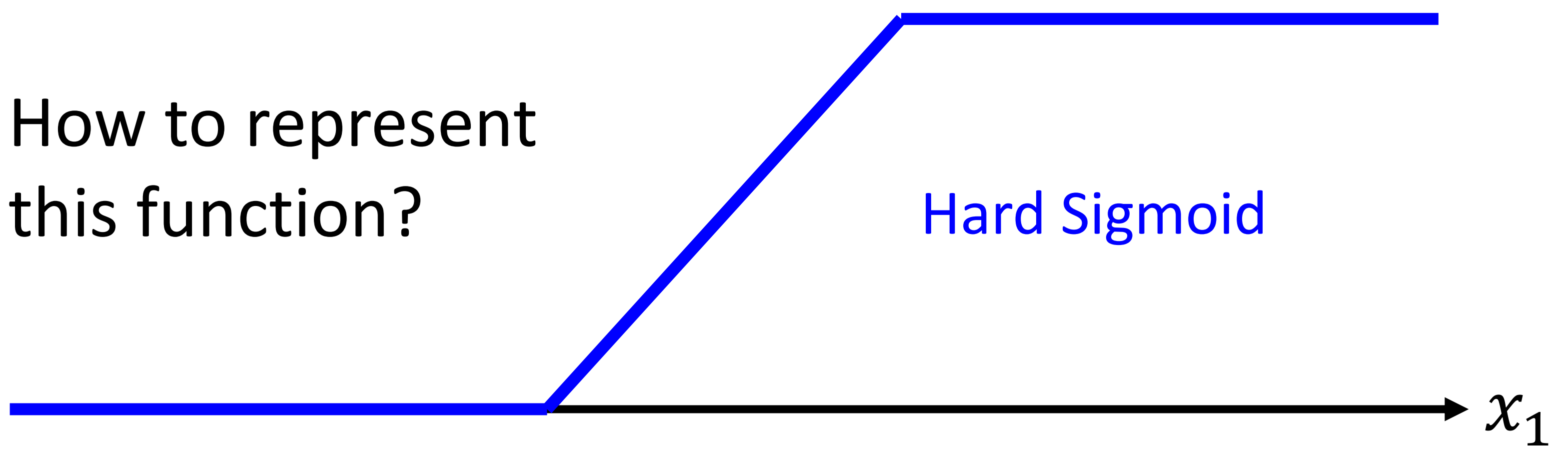


To have good approximation, we need sufficient pieces.

How to represent?

red curve = constant + sum of a set of 

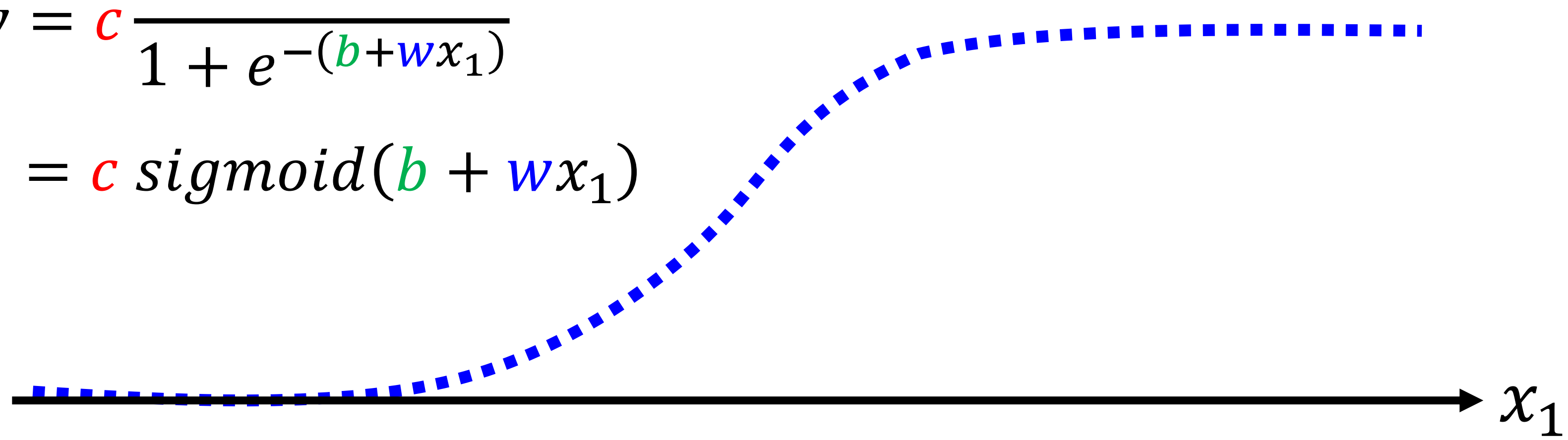
How to represent this function?



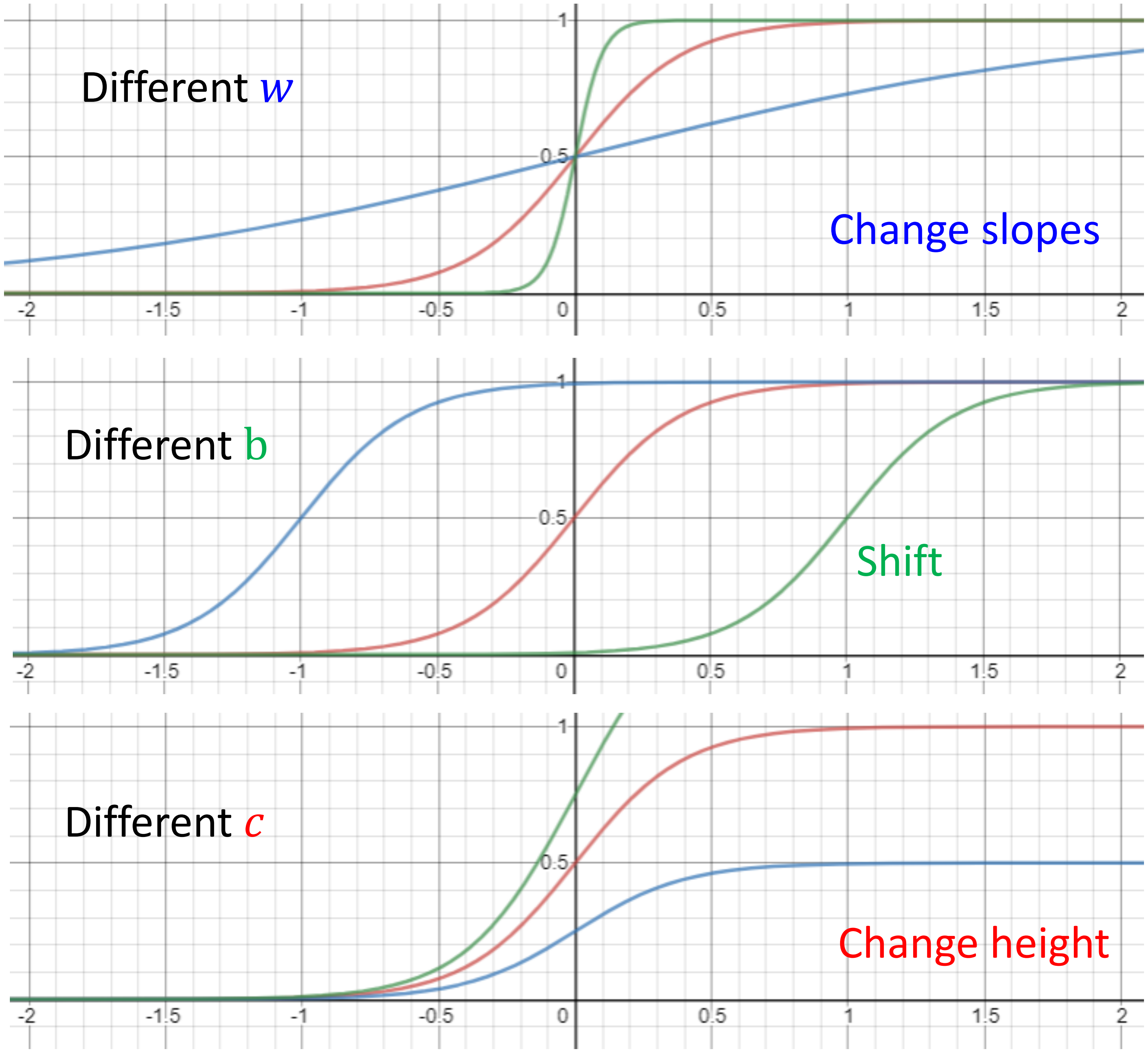
Sigmoid Function

$$y = c\frac{1}{1 + e^{-(b+wx_1)}}$$

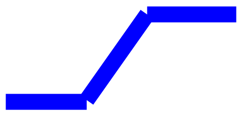
$$= c\ sigmoid(b + wx_1)$$

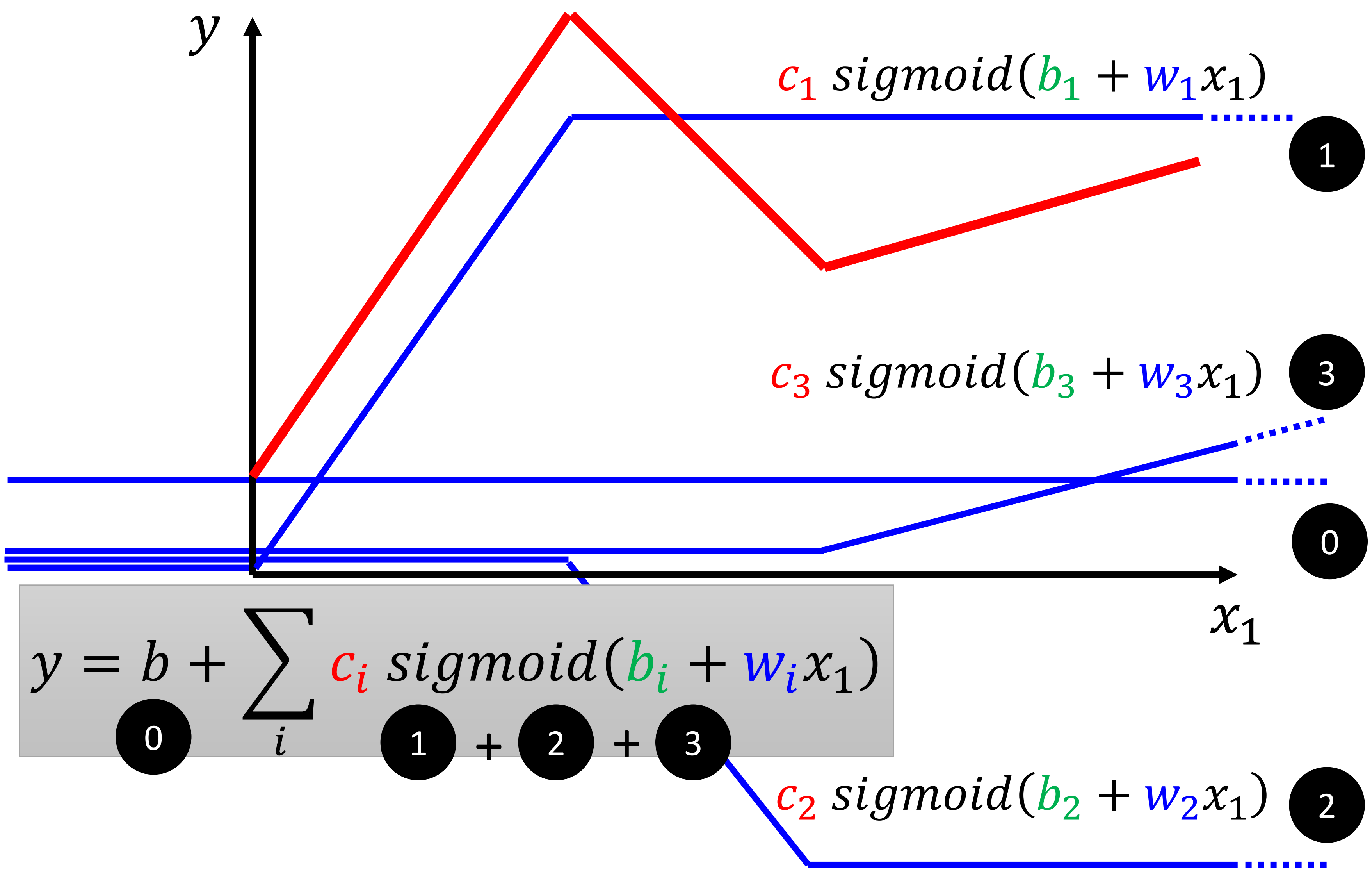


Property of sigmoid



The nonlinear becomes...

red curve = sum of a set of  + constant



$$y = b + \sum_i c_i\ sigmoid(b_i + w_i x_1)$$

0 1 + 2 + 3

Similar for more inputs

$$y = b + wx_1$$

$$y = b + \sum_i c_i \, sigmoid(b_i + w_i x_1)$$

$$y = b + \sum_j w_j x_j$$

$$\boxed{y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

One example

$$y = b + \sum_i c_i \, sigmoid \left(b_i + \sum_j w_{ij} x_j \right)$$

j: 1,2,3
no. of features

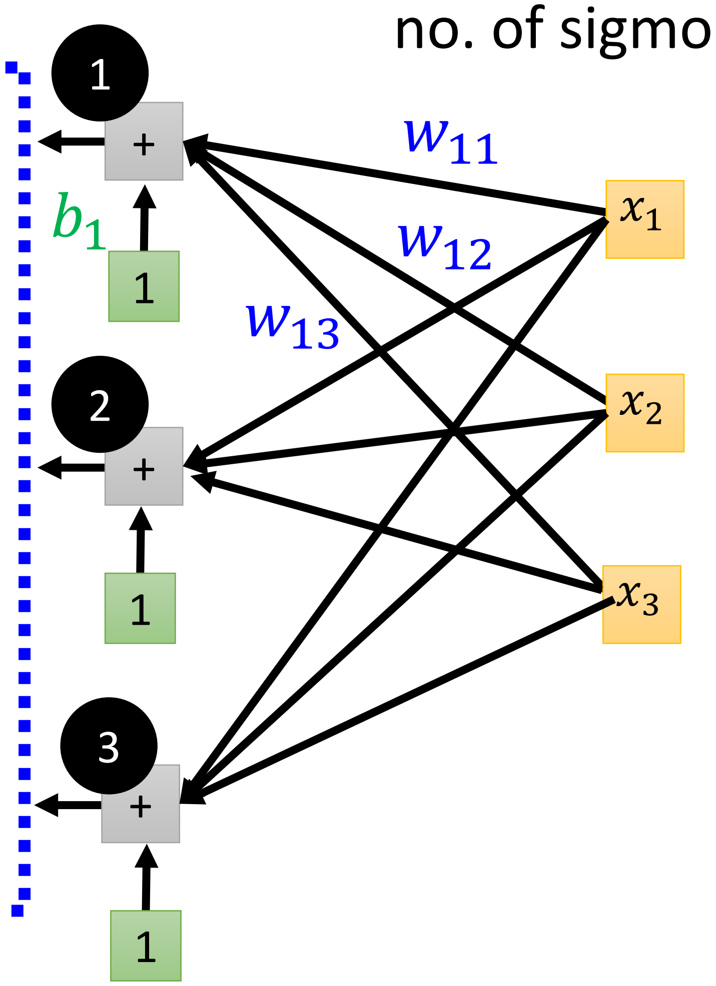
i: 1,2,3
no. of sigmoid

$r_1 = b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3$

w_{ij}: weight for x_j for i-th sigmoid

$r_2 = b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3$

$r_3 = b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3$



Matrix representation

$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$

i: 1,2,3

j: 1,2,3

$$r_1 = b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

$$r_2 = b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3$$

$$r_3 = b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3$$

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{r} = \boldsymbol{b} + W \boldsymbol{x}$$

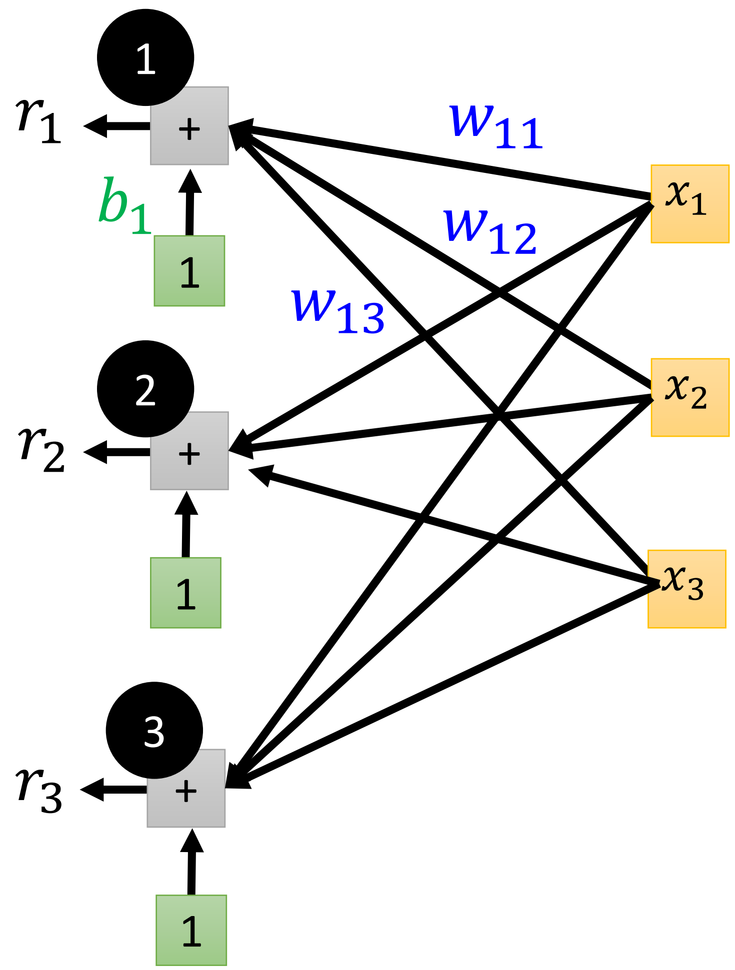
Matrix representation

$$y = b + \sum_i c_i \, sigmoid \boxed{\left(b_i + \sum_j w_{ij} x_j \right)}$$

i: 1,2,3

j: 1,2,3

$$\boldsymbol{r} = \boldsymbol{b} + W \boldsymbol{x}$$

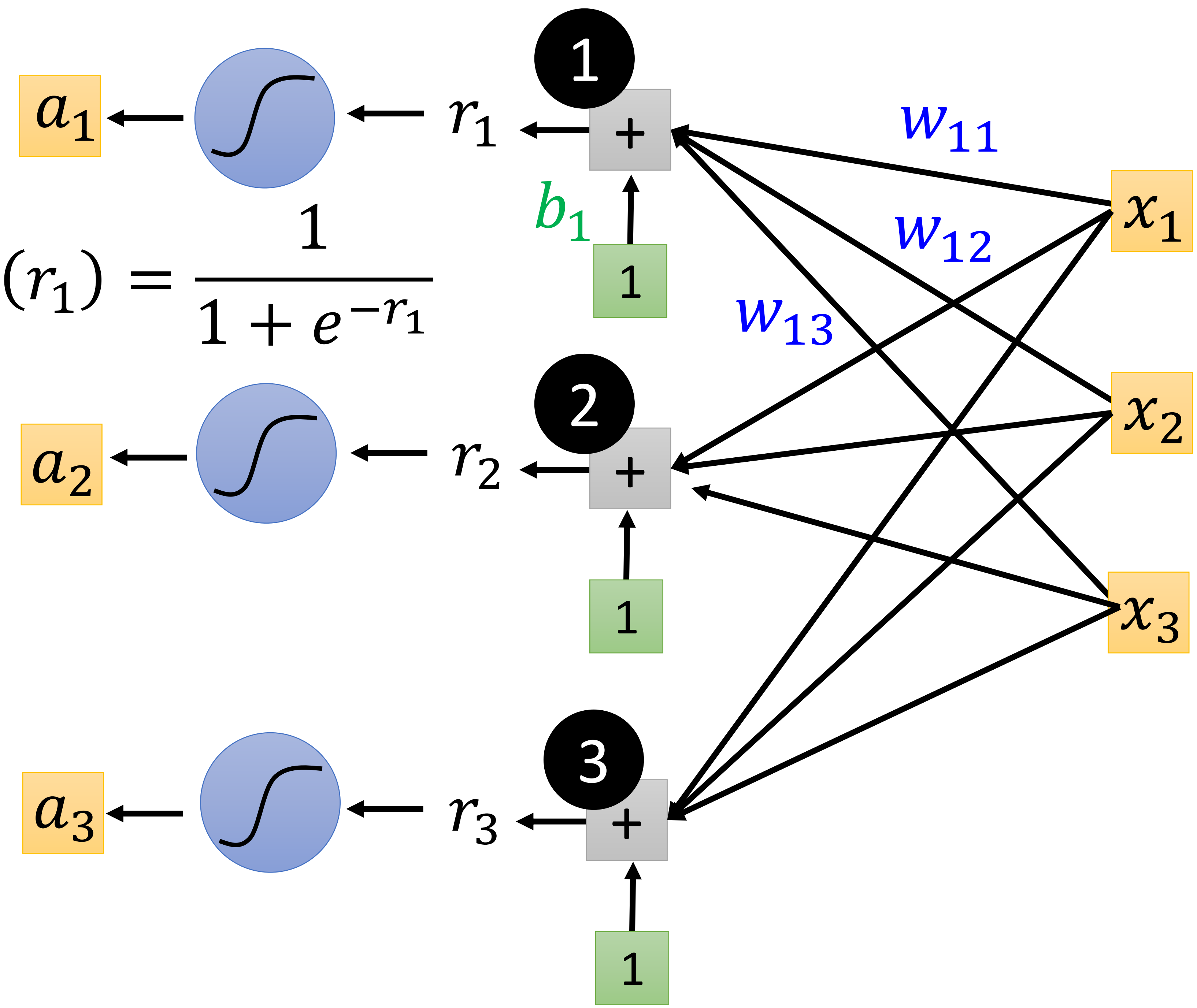


$$y = b + \sum_i c_i \boxed{sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

i: 1,2,3

j: 1,2,3

$$a_1 = sigmoid(r_1) = \frac{1}{1 + e^{-r_1}}$$

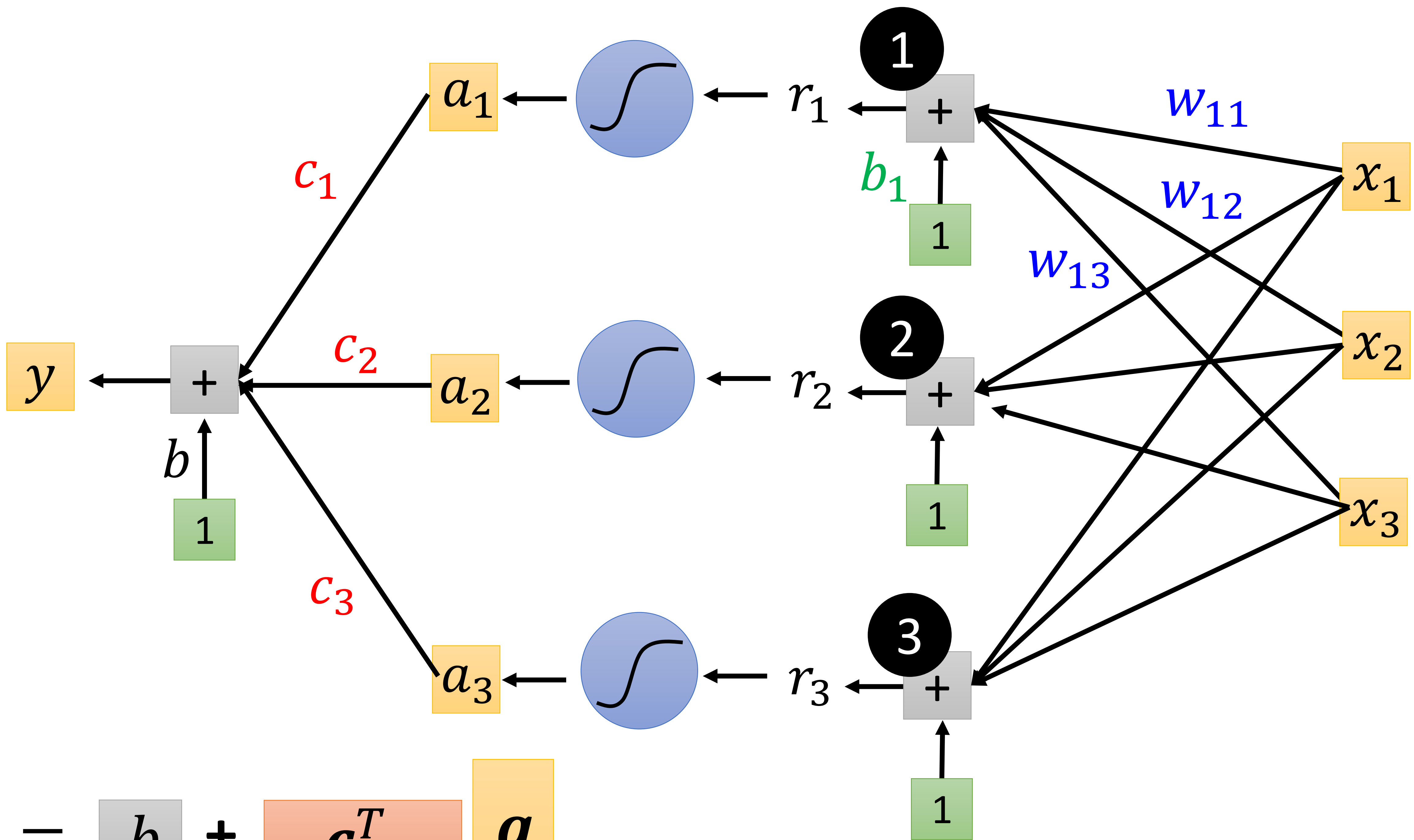


$$\boldsymbol{a} = \sigma(\boldsymbol{r})$$

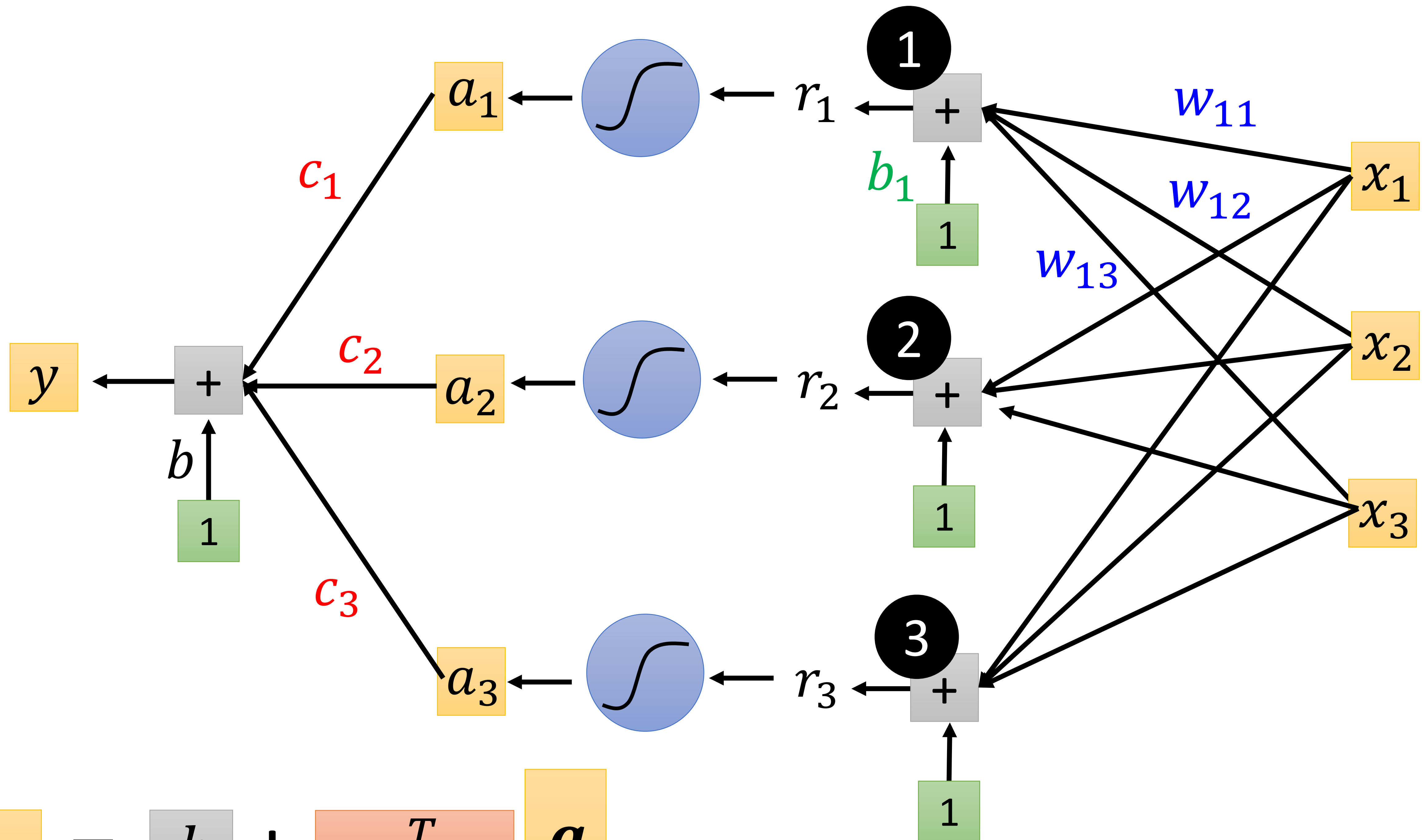
$$y = \boxed{b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

i: 1,2,3

j: 1,2,3

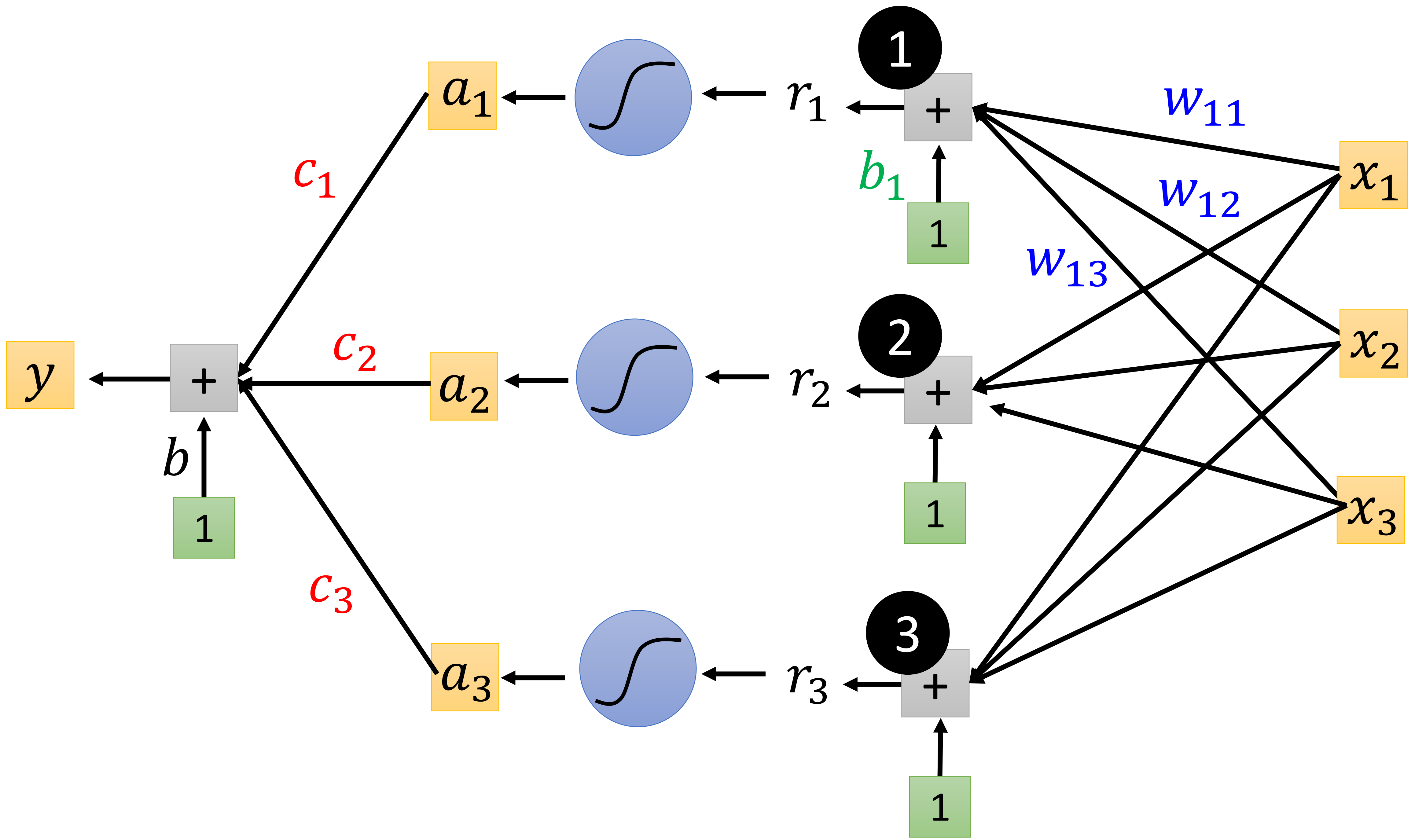


$$y = b + \boldsymbol{c}^T \boldsymbol{a}$$



$$y = b + c^T a$$

$$a = \sigma(r) \qquad r = b + W x$$



$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

Function with unknown parameters

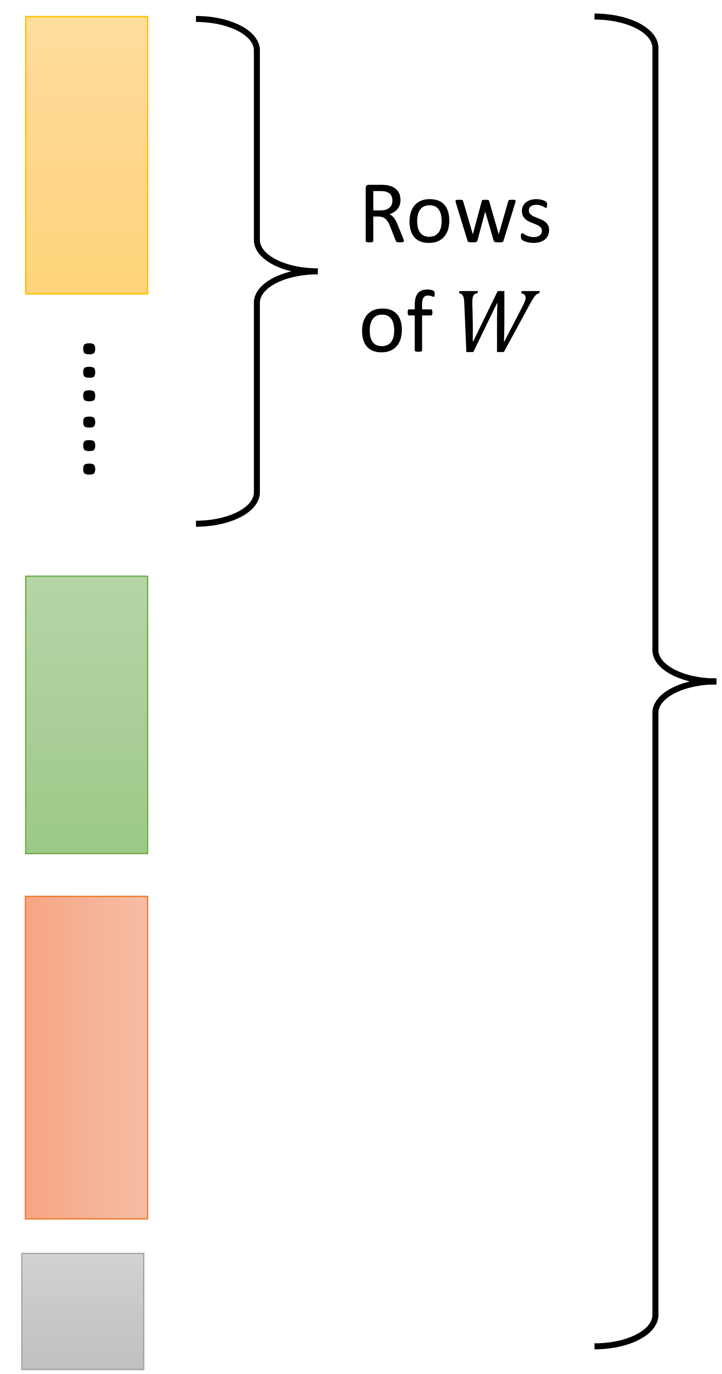
$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

$\boldsymbol{x}$ **feature**

Unknown parameters

W $\quad \boldsymbol{b}$

$\boldsymbol{c}^T$ $\quad b$



$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

Loss

Step 1: function with unknown

Step 2: define loss from training data

Step 3: optimization

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

Loss

- Loss is a function of parameters $L(\theta)$
- Loss means how good a set of values is.

feature

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

$$y \overset{e}{\longleftrightarrow} \hat{y}$$

label $\hat{y}$

Given a set of values

Loss: $$L = \frac{1}{N} \sum_n e_n$$

Optimization

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

Optimization of New Model

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$

$$\boldsymbol{g} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

gradient

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \vdots \end{bmatrix} - \begin{bmatrix} \eta \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \eta \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

Optimization of New Model

$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$$

- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$$

Optimization of New Model

$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

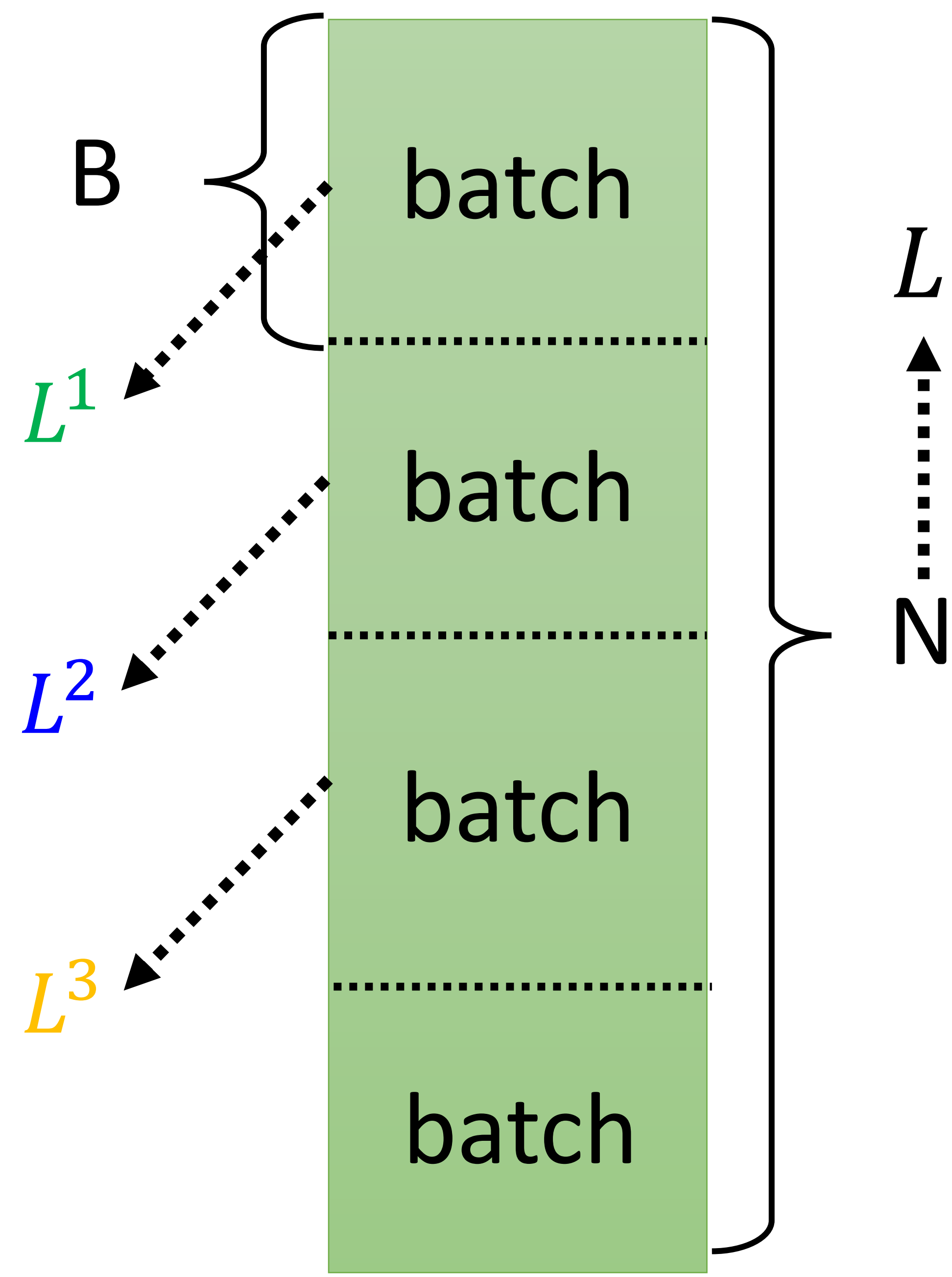
- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$

  update $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^2(\boldsymbol{\theta}^1)$

  update $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L^3(\boldsymbol{\theta}^2)$

  update $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$

1 **epoch** = see all the batches once



Summary

Training data: $\{(\boldsymbol{x}^{\mathbf{1}}, \hat{y}^1), (\boldsymbol{x}^{\mathbf{2}}, \hat{y}^2), \ldots, (\boldsymbol{x}^{\boldsymbol{N}}, \hat{y}^N)\}$

Training:

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

$y = f_{\boldsymbol{\theta}}(\boldsymbol{x})$

$L(\boldsymbol{\theta})$

$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L$

Testing data: $\{\boldsymbol{x}^{\boldsymbol{N+1}}, \boldsymbol{x}^{\boldsymbol{N+2}}, \ldots, \boldsymbol{x}^{\boldsymbol{N+M}}\}$

Use $y = f_{\boldsymbol{\theta}^*}(\boldsymbol{x})$ to label the testing data

$\{y^{N+1}, y^{N+2}, \ldots, y^{N+M}\}$

Evaluation

- Small loss on training data, large loss on testing data.

An extreme example

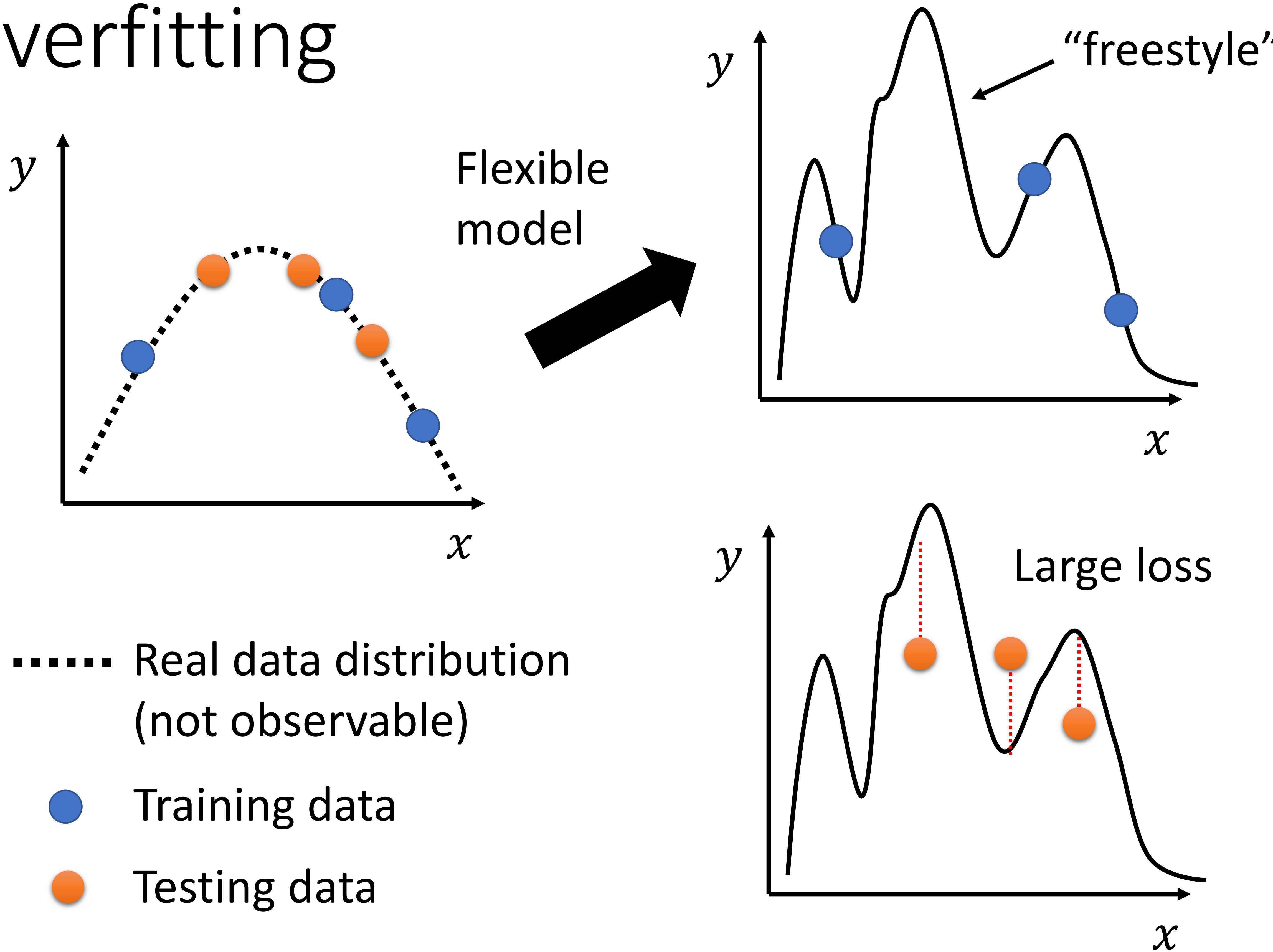
Training data: $\left\{\left(\boldsymbol{x}^{\mathbf{1}},\hat{y}^{1}\right),\left(\boldsymbol{x}^{\mathbf{2}},\hat{y}^{2}\right),\dots,\left(\boldsymbol{x}^{\boldsymbol{N}},\hat{y}^{N}\right)\right\}$

$$f(\boldsymbol{x})=\begin{cases}\hat{y}^{i} & \exists \boldsymbol{x}^{\boldsymbol{i}}=\boldsymbol{x}\\ random & otherwise\end{cases}$$

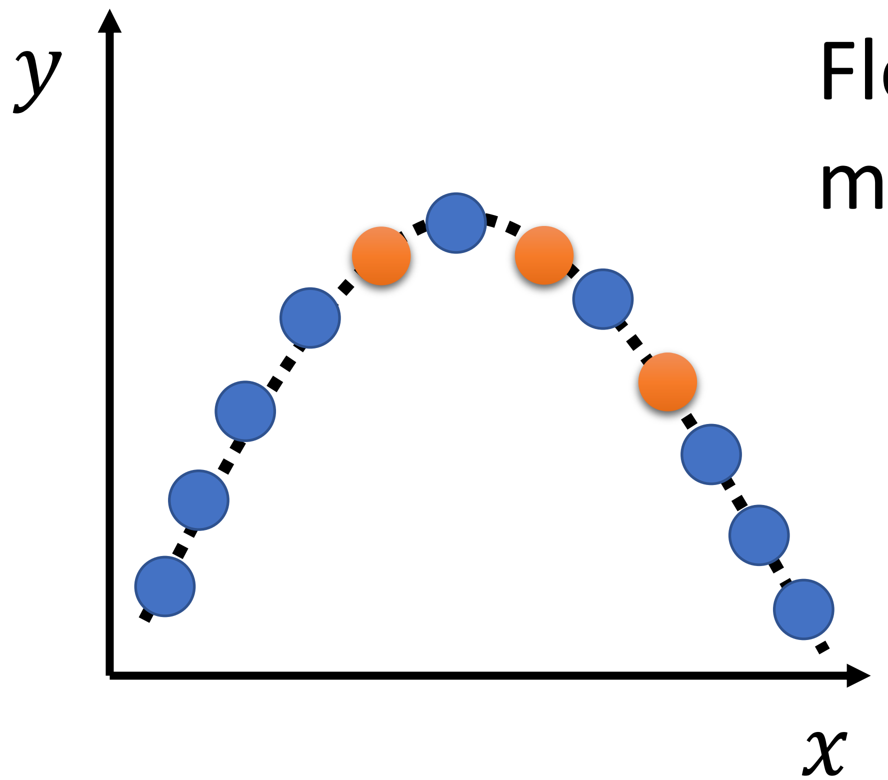
Less than useless ...

This function obtains **zero training loss**, but **large testing loss**.

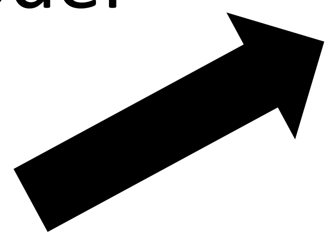
Overfitting

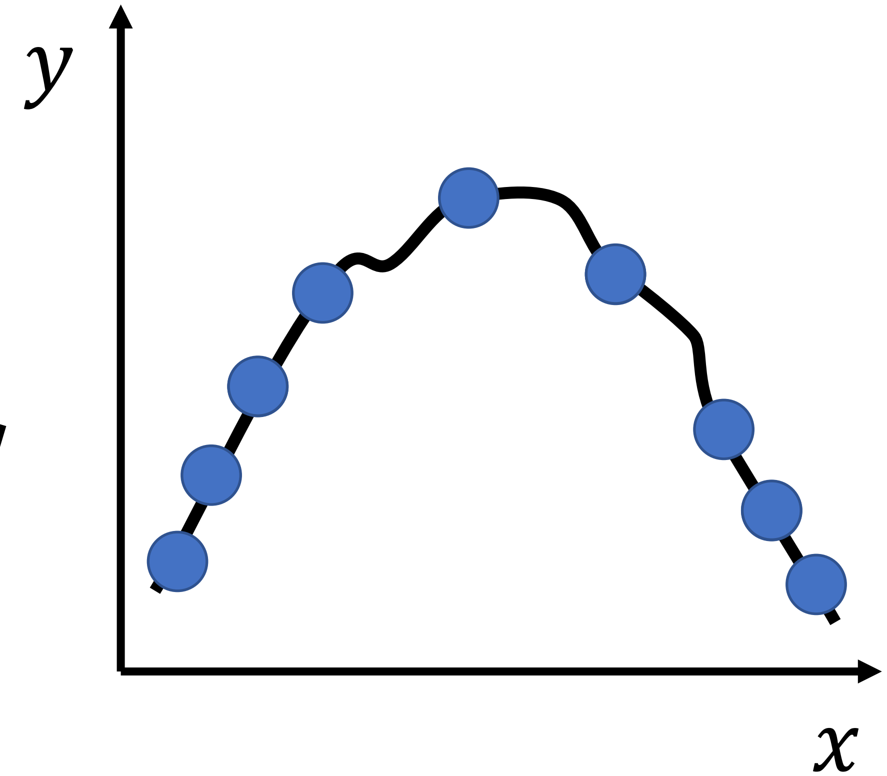


Overfitting



Flexible model

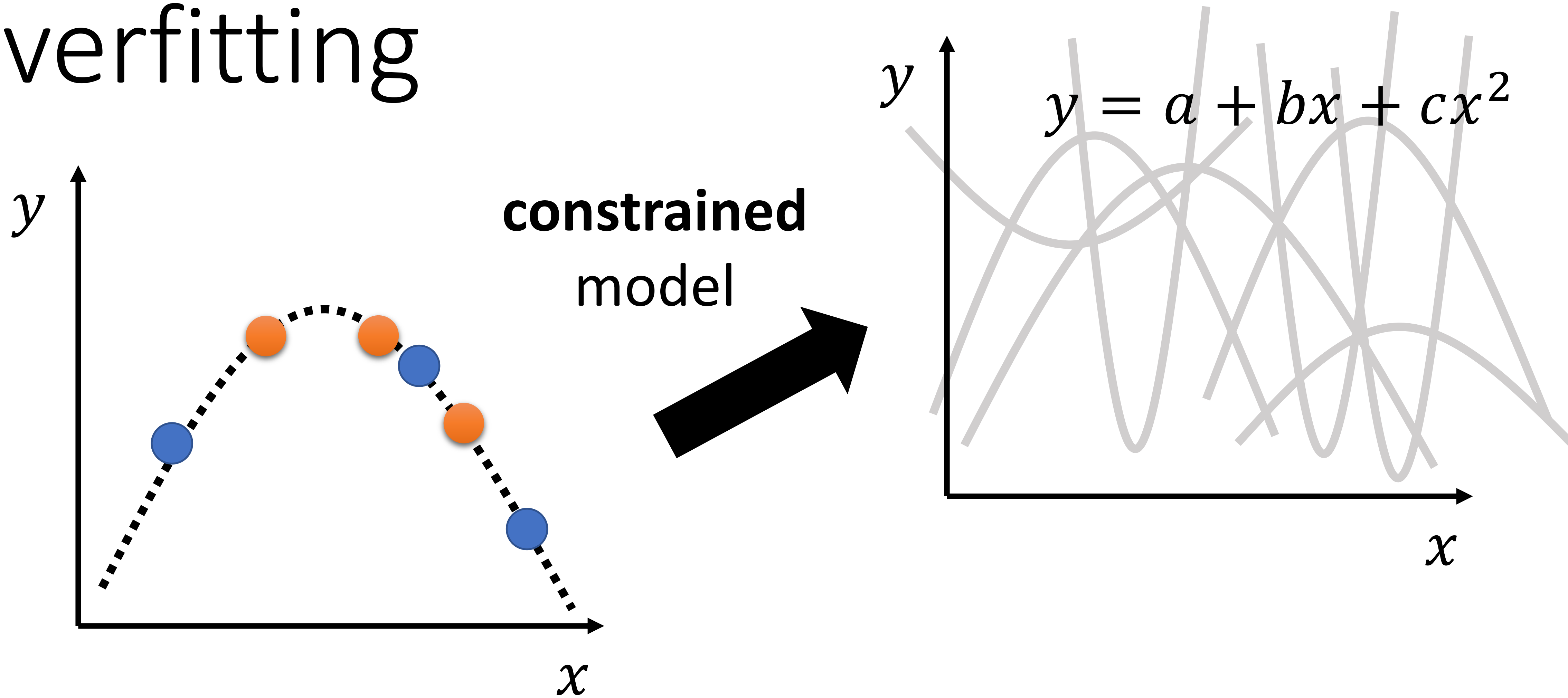




More training data

Data augmentation

Overfitting

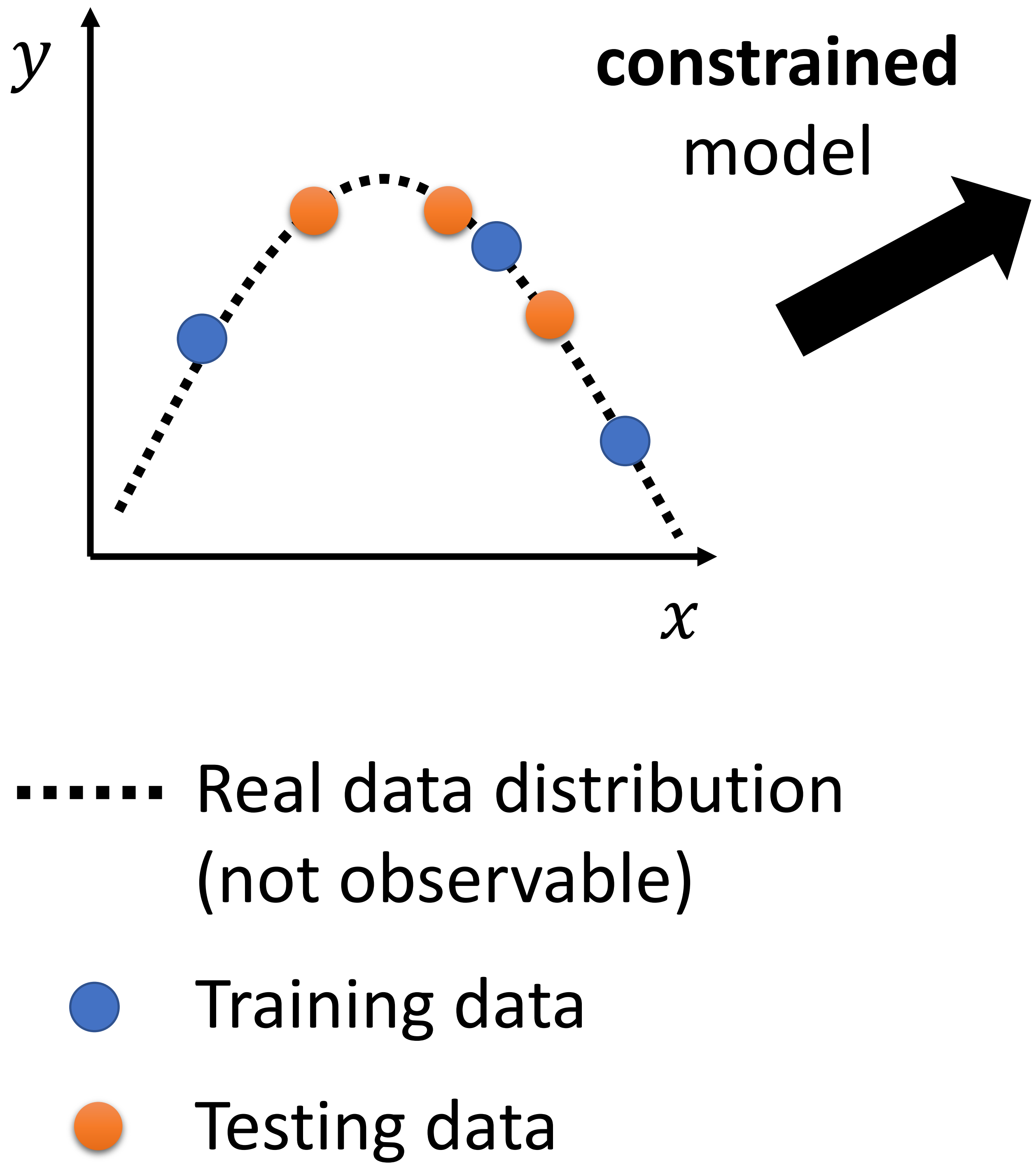


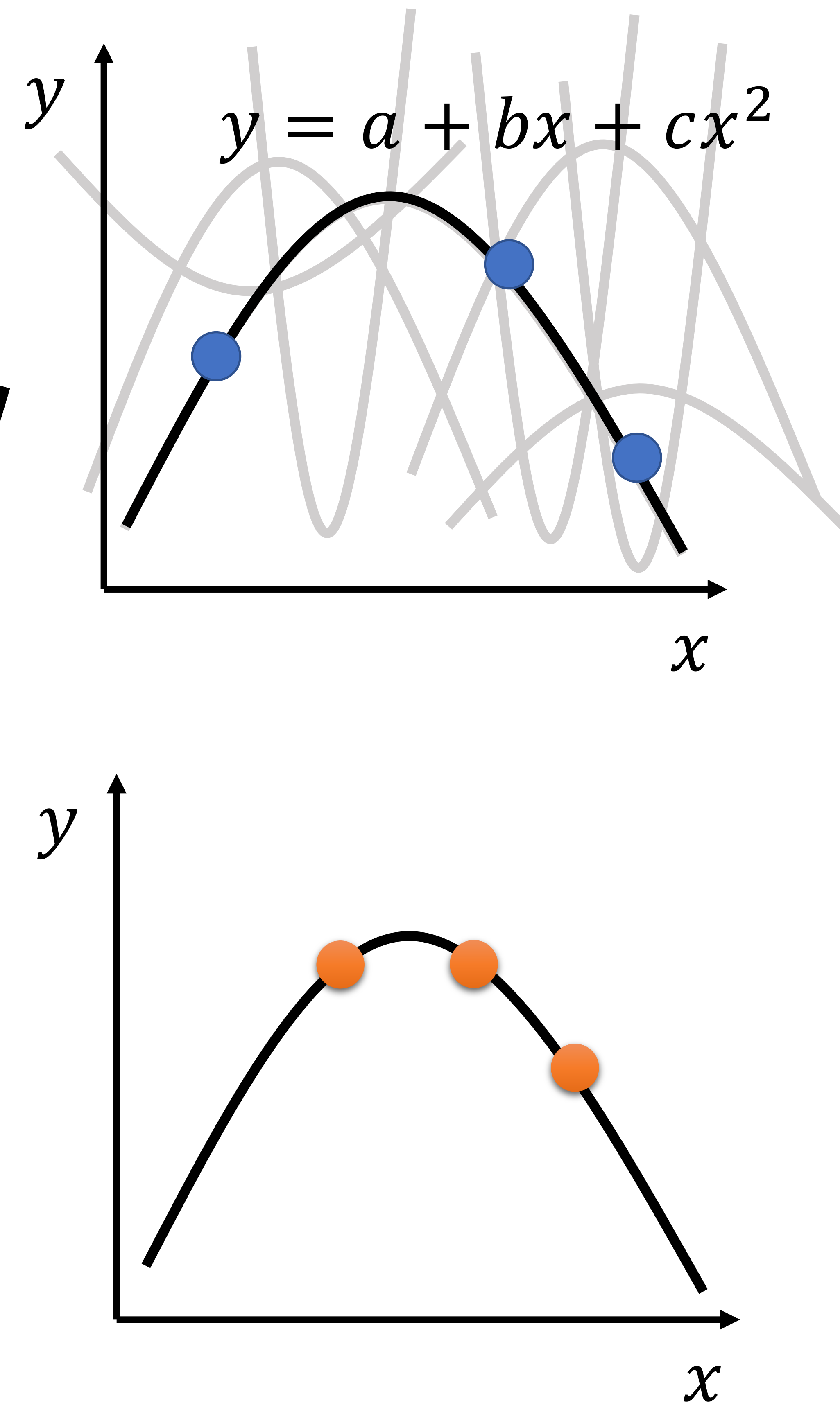
Real data distribution
(not observable)

Training data

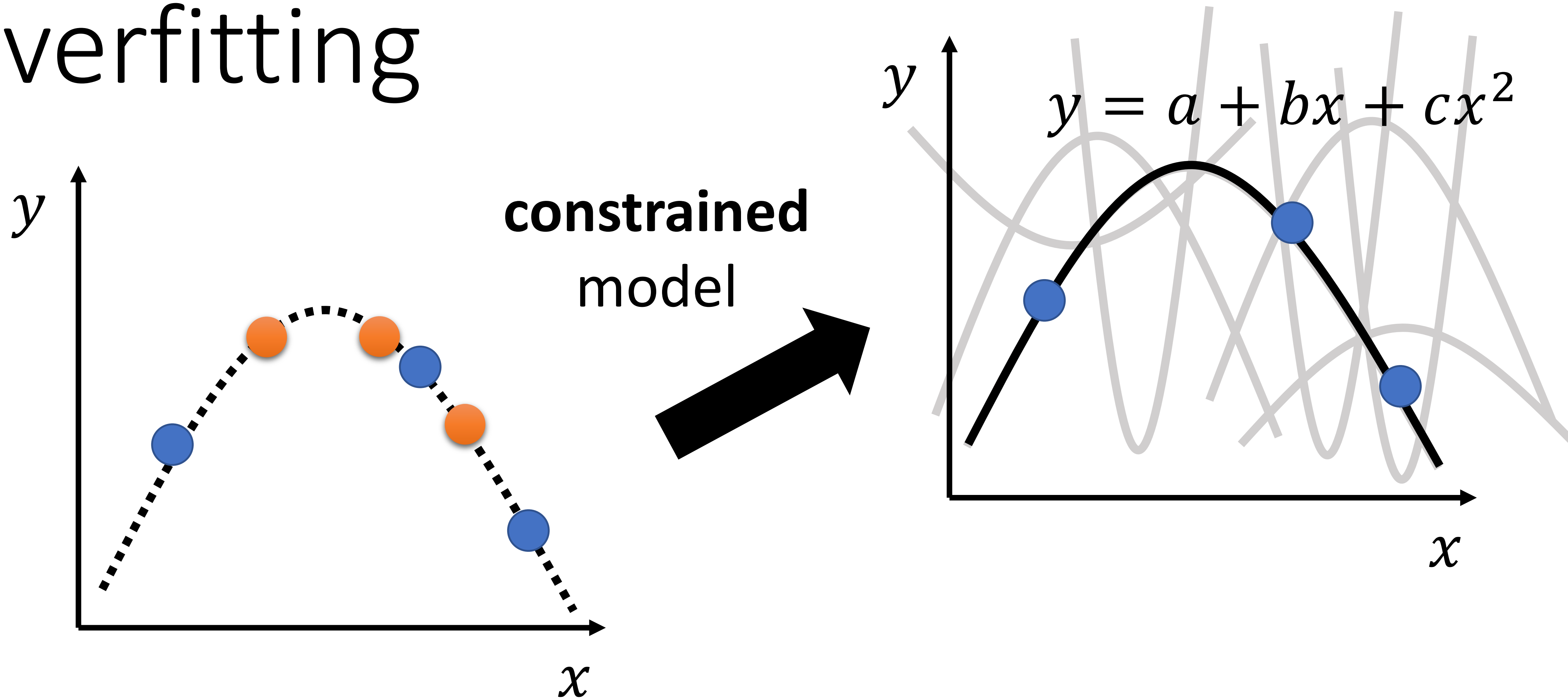
Testing data

Overfitting



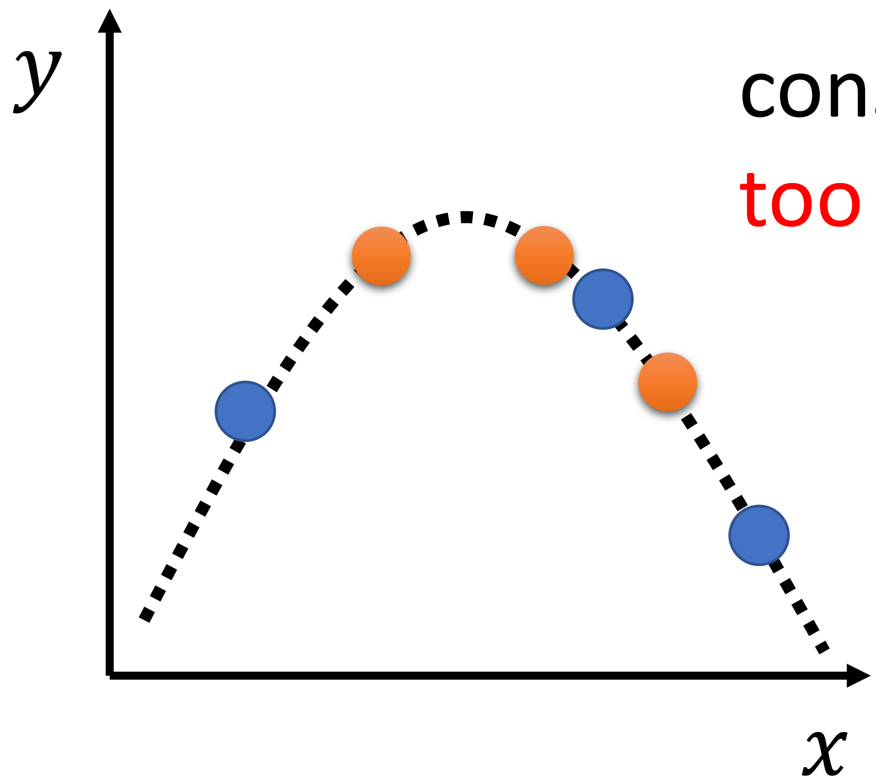


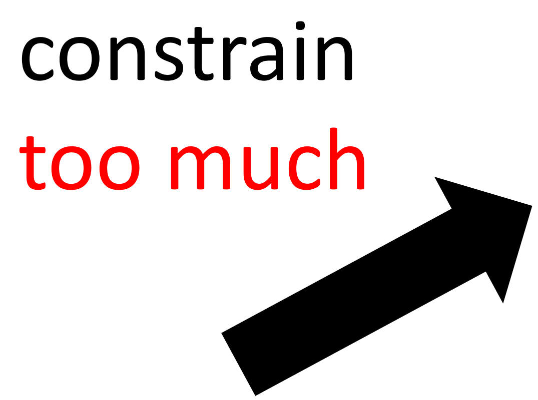
Overfitting

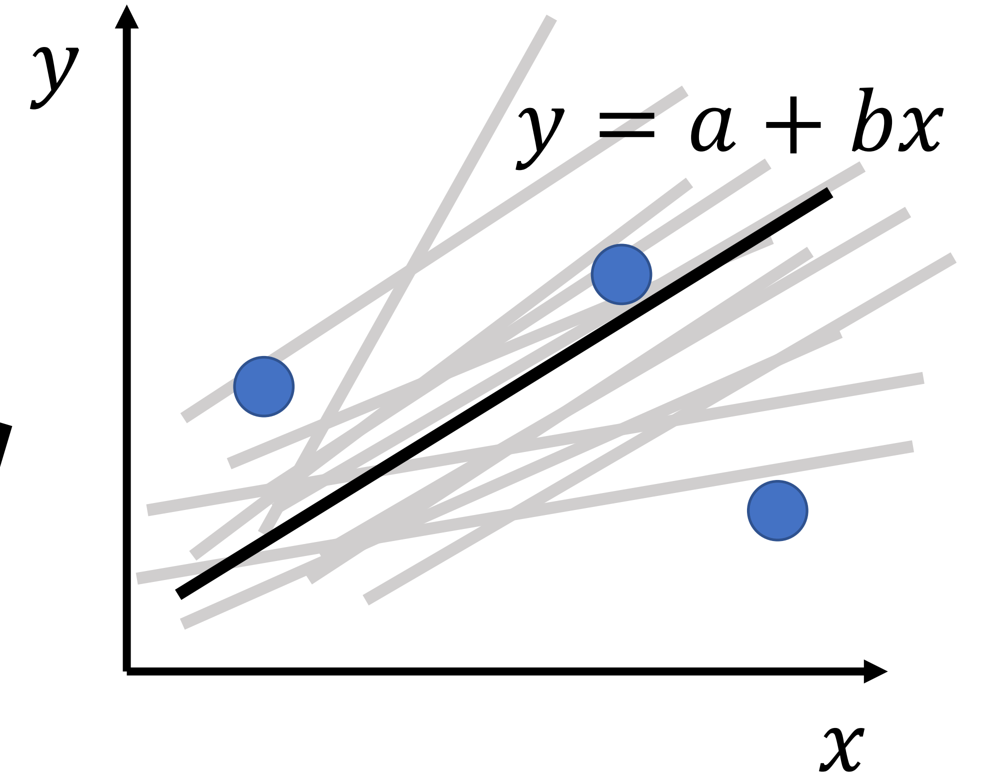


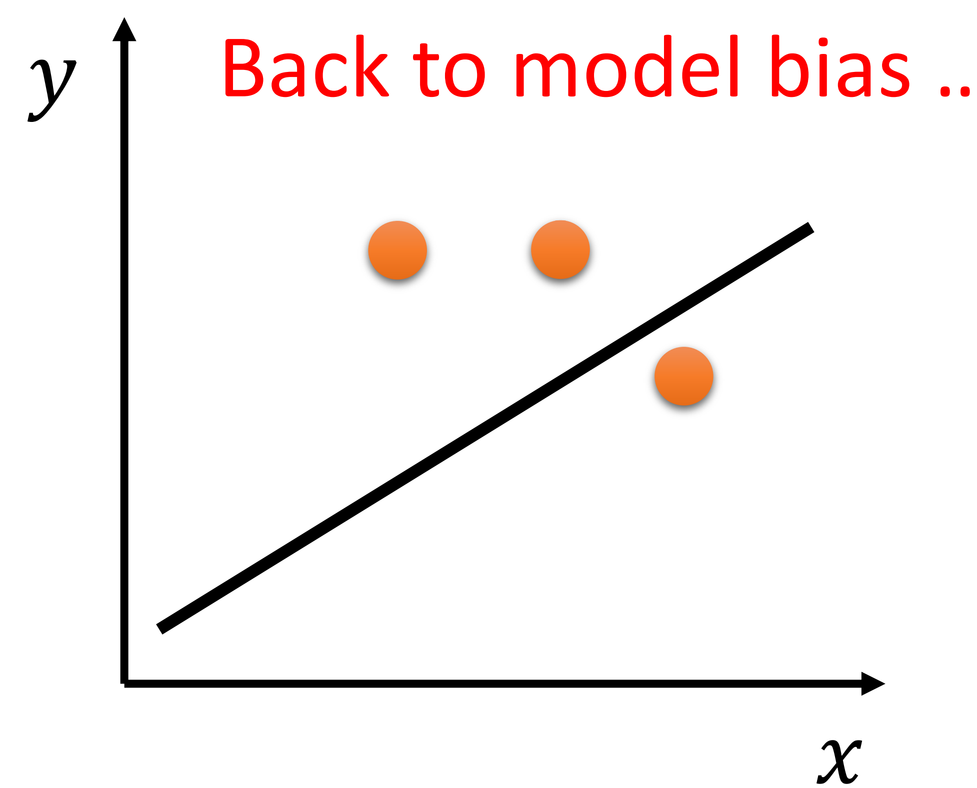
- Less parameters, sharing parameters
- Less features
- Early stopping
- Regularization
- Dropout

Overfitting







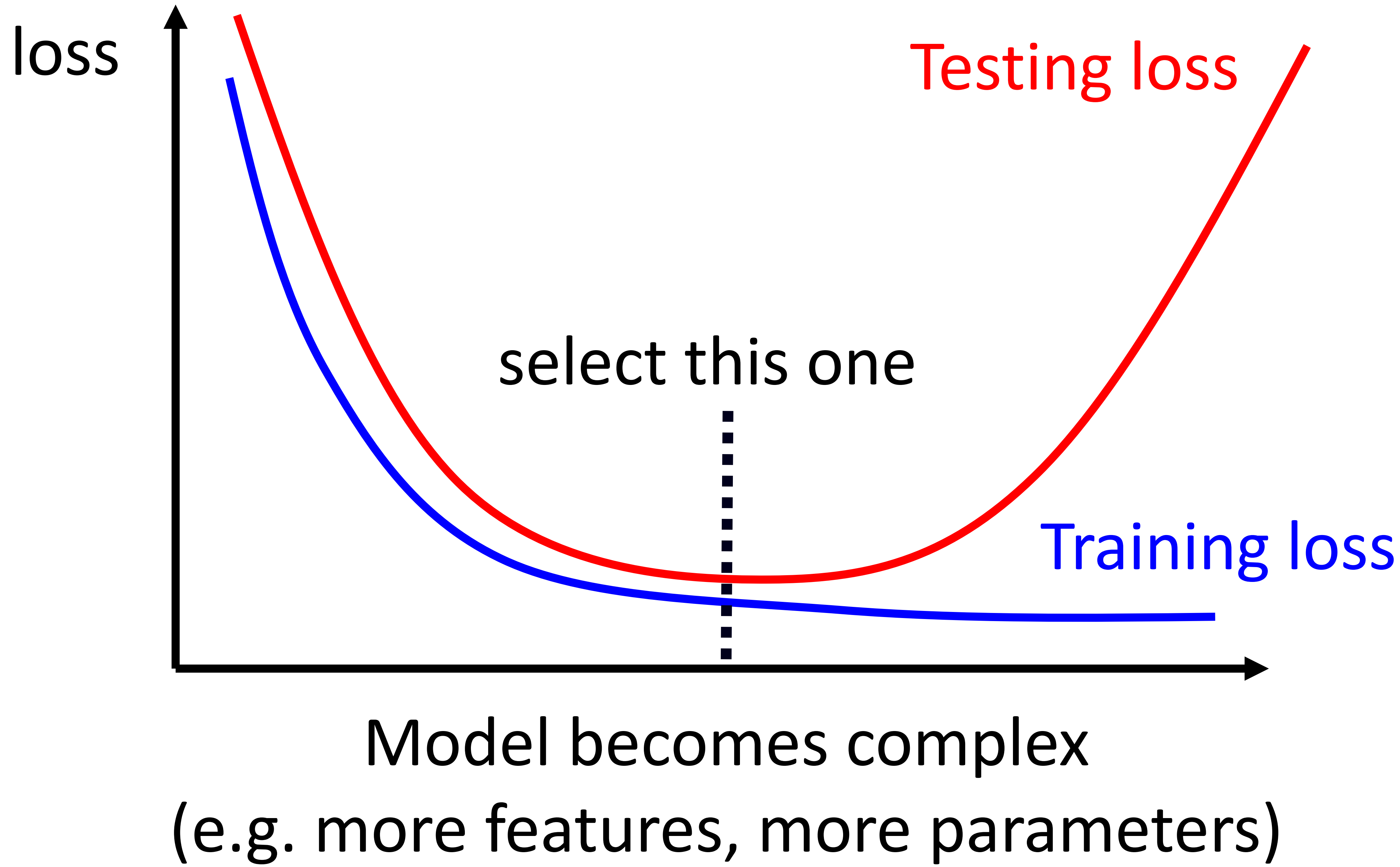


Real data distribution (not observable)

Training data

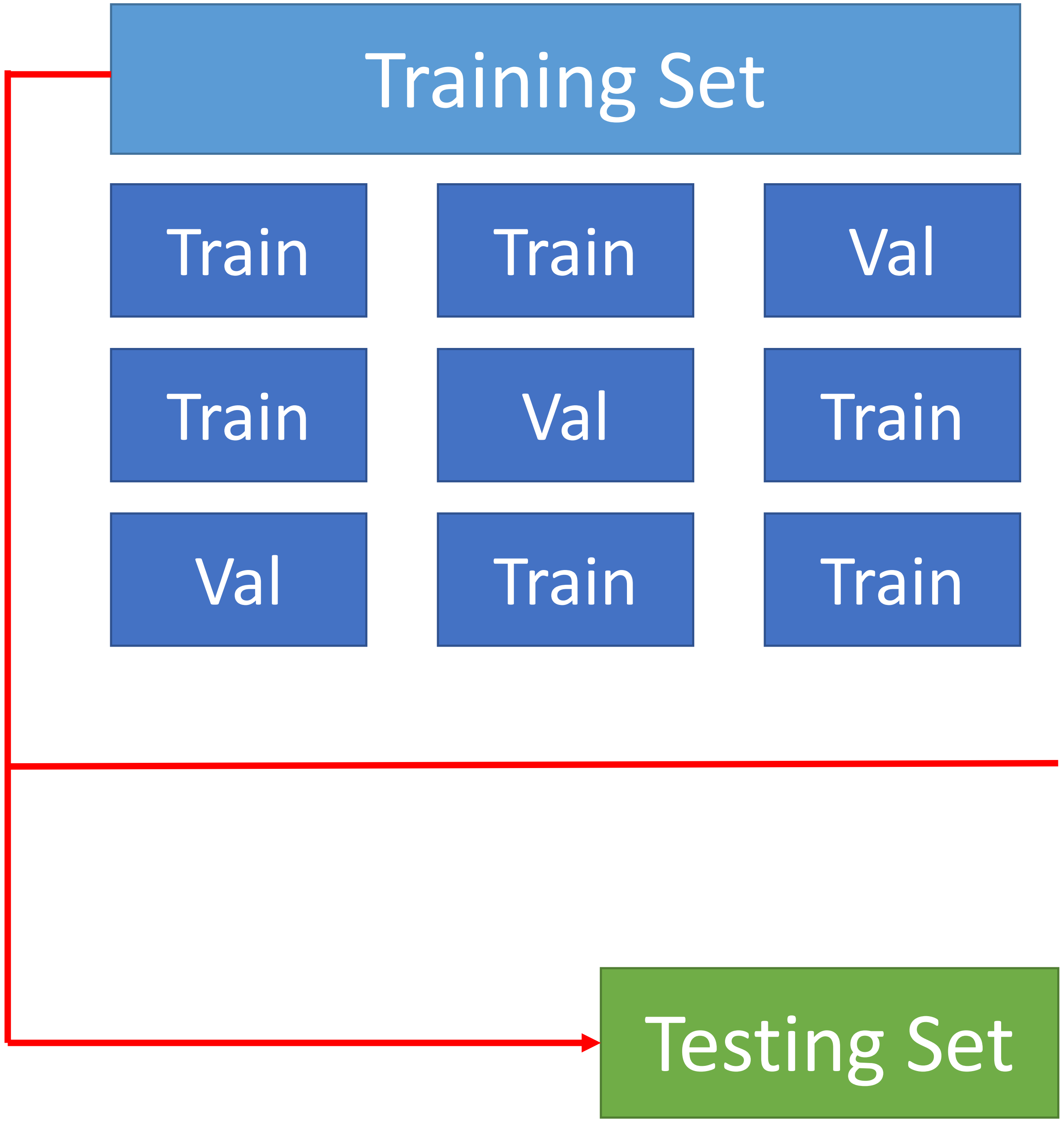
Testing data

Bias-Complexity Trade-off



Model Selection

N-Fold Cross Validation



Model 1	Model 2	Model 3
mse = 0.2	mse = 0.4	mse = 0.4
mse = 0.4	mse = 0.5	mse = 0.5
mse = 0.3	mse = 0.6	mse = 0.3
Avg mse = 0.3	Avg mse = 0.5	Avg mse = 0.4

More Learning Paradigms

- What information is available for learning?
  - What does the data look like?
  - How is it annotated?

- What output is desired?
  - What should the algorithm produce?
  - How will it be used?

Supervised Learning

- Learning with a teacher
  - Explicit feedback in the form of labeled examples
  - Goal: make prediction
  - Pros: Good performance
  - Cons: Labeled data is difficult to find
  - “Classical” (labeled data simply there; do your best)
  - Query-based (can ask for labeled examples)

- Examples
  - Classification
    - Is an email spam or not?
  - Regression
    - What is the expected rate of return on a specific investment?

Unsupervised Learning

- Learning without labels
  - Only observed unlabeled examples
  - Goal: uncover structure in data
  - Pros: Easy to find lots of data
  - Cons: what are we looking for?
- Examples
  - Clustering
    - Group emails by topic
  - Manifold learning
    - Find a low dimensional data representation

Reinforcement Learning

- Learn a behavior policy by interacting with the world
  - How to navigate in a world
  - Success measured by rewards received for actions taken
  - Maximize sum of rewards
- Examples
  - Chess (and checkers)
  - Robot control
  - Piloting an airplane