



AIDOS LAB
AI FOR DATA-ORIENTED SCIENCE



Lecture 1: Introduction to Deep Learning

Geometry and Topology in Machine Learning

Inés García Redondo — September 14, 2026

Logistics

Contact information and house keeping

Hello! My name is Inés García-Redondo

I am a mathematician, and a physicist, working as a researcher at the University of Fribourg in the AIDOS (AI for Data Oriented Science) group, where we study the role of geometry and topology in deep learning

My email is: ines.garciaredondo@unifr.ch

My office is in PER21, B429

Feel free to reach out to me by email with any questions or doubts about the course!

Course page:

<https://ines.topology.rocks/course-pages/gtml-2026>



Logistics

Contact information and house keeping

Hello! My name is Martín Carrasco Castañeda

I am a computer scientist, doing a PhD in the AIDOS group. I like graphs and other combinatorial structures, and understanding their interplay with deep learning

My email is: martin.carrascocastaneda@unifr.ch

My office is in PER21, B440



Classes and assessment

Each **class** will be divided into:

1. *theory session* (9:15-11:00, with a 10 minute break)
2. followed by a *practice session* (11:10-12:00)
 - Practice sessions will be either *coding* based or *problem solving* sessions

Assessment will have two parts:

1. Group projects + presentations reproducing results from a current research paper (30%)
 - You will form groups of 3 and produce code in a public repository following a paper from the course
 - You will prepare a presentation of the ideas of the paper, and your code and results, and present it at the end of the course
2. Oral exam at the end of the course (70%)

Use of AI

The use of AI is permitted in this course, although it must be always appropriately acknowledged

For instance, if you use AI to help coding your project, you should at least identify any AI tool used and the purpose for which it was used

AI-use statement: I used [tool/model] for [purpose]. It contributed to [specific part of the work]. I verified or revised the output by [method].

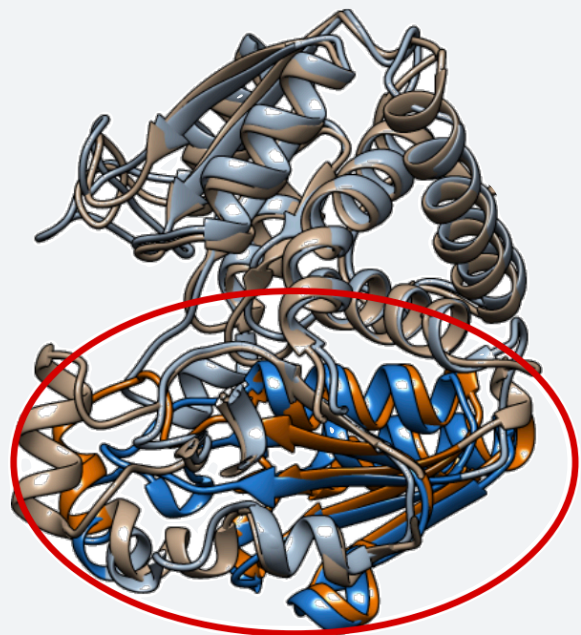
This being said, permission to use AI is not permission to outsource your understanding

Precisely for this reason, the grades of the course are based on oral presentations, where I might ask questions and interact with you to gauge your understanding of your code, the references that you have followed, the extent of your own contribution, etc.

Data has shape, shape has meaning, and meaning begets understanding

Gunnar Carlsson, paraphrased

Shape **in** data



Real data is often times best described through *relational structures*

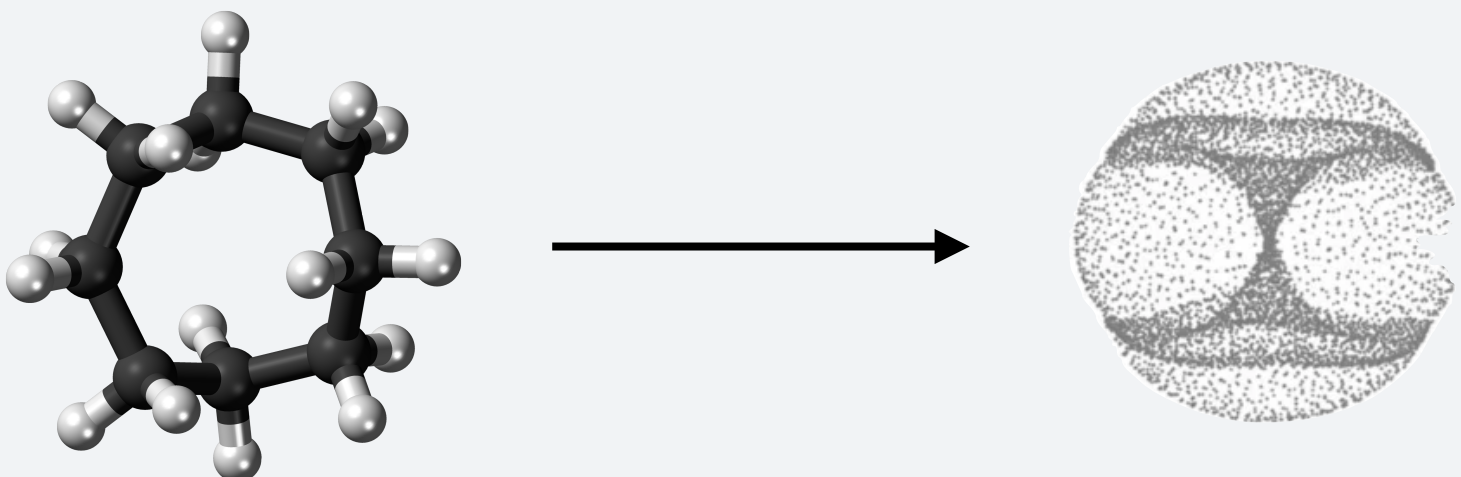
Example: distinguishing knotted/ unknotted proteins

Benjamin et al., (2023)

How do we *measure* shape?

Shape **of** data

High-dimensional data can live in a *lower dimensional space*



Example: conformation space of cyclo-octane

1,031,644 coordinate vectors in $\mathbb{R}^{72}$

Martin and Watson, (2011)

Geometry	=	fine detail (local properties)	+	quantitative answers
Topology	=	fundamental structure (global properties)	+	qualitative answers

The course

What to expect

Sometimes we want to learn from data that has some additional *relational* structure...

Other times we wonder whether the *parameters* or the *internal representations* of our model have some *shape* or geometric signature that informs us about the *training dynamics*, the *generalization* properties or any other characteristic of our model...

In this course, we will introduce many **geometric and topological tools for data analysis** that are useful on their own, and we will see how to compute them

Then, we will see how to **apply them in the context of deep learning** to study problems like the ones above

But before all that, we need to lay the foundation of what we mean by *deep learning*, and the many hyperparameters and architectural choices it might involve

Lecture 1 — Introduction to Deep Learning

Outline

- What is deep learning?
 - Classification vs. regression tasks
- The multilayer-perceptron: the building block of DL
- Training neural networks
 - Gradient descent
 - Backpropagation
- Learning from images:
 - Convolutional neural networks
- Learning from text:
 - Transformers and the attention mechanism

What is Deep Learning?

Lecture 1 — Introduction to Deep Learning

The power of AI

And how it’s shaping our world



AlphaFold 2/3



AlphaGeometry and AlphaProof



Large Language Models

Article

Olympiad-level formal reasoning with reinforcement learning

https://doi.org/10.1038/s41586-025-09833-y

Received: 3 June 2025

Accepted: 30 October 2025

Published online: 12 November 2025

Open access

Check for updates

Science

Formalizing Fermat's Last Theorem

Sep 4, 2026

We are sharing the first complete computer-checked proof of Fermat’s Last Theorem. Claude worked largely autonomously over 11 days to write the proof in the Lean programming language. Below, we describe how the formalization was done and share some thoughts about what this work could mean for research mathematics.

September 8, 2026 Research Publication

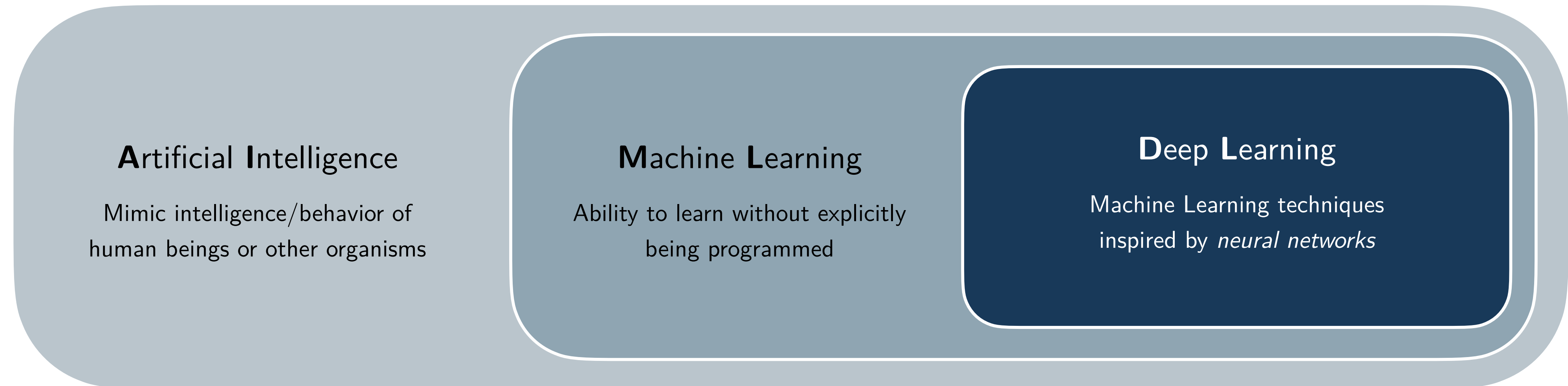
On the Navier–Stokes Millennium Prize Problem

[Read the paper](#) [Link to Lean formalized proof >](#)



Image and video generation

What is Deep Learning?



Teaching computers how to **learn a task** directly from **raw data**

What is Deep Learning?

Artificial Intelligence: are we there yet?

Intelligence is not a scalar quantity. The space of problems is gigantic and [...] any intelligent system will only excel at a tiny subset of them. Any intelligent system has blind spots, hence vulnerabilities.

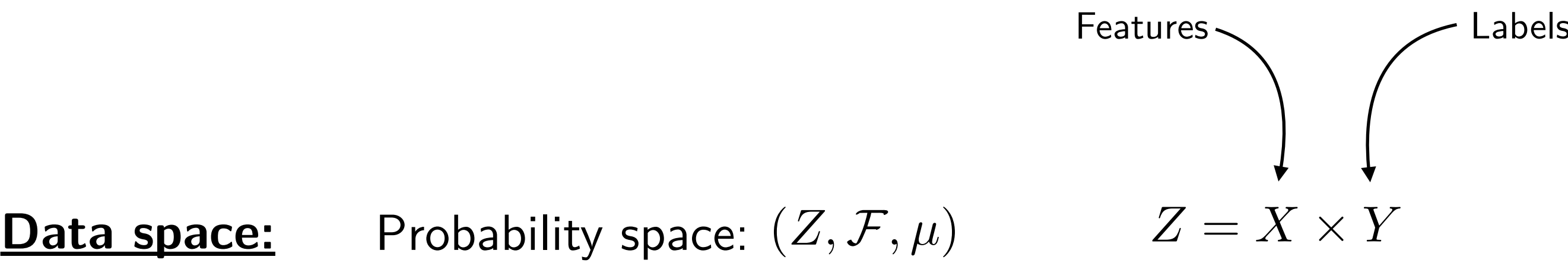
Yann LeCun

Best to think of AI as *PI* (pseudo-intelligence) or *SI* (simulated intelligence).

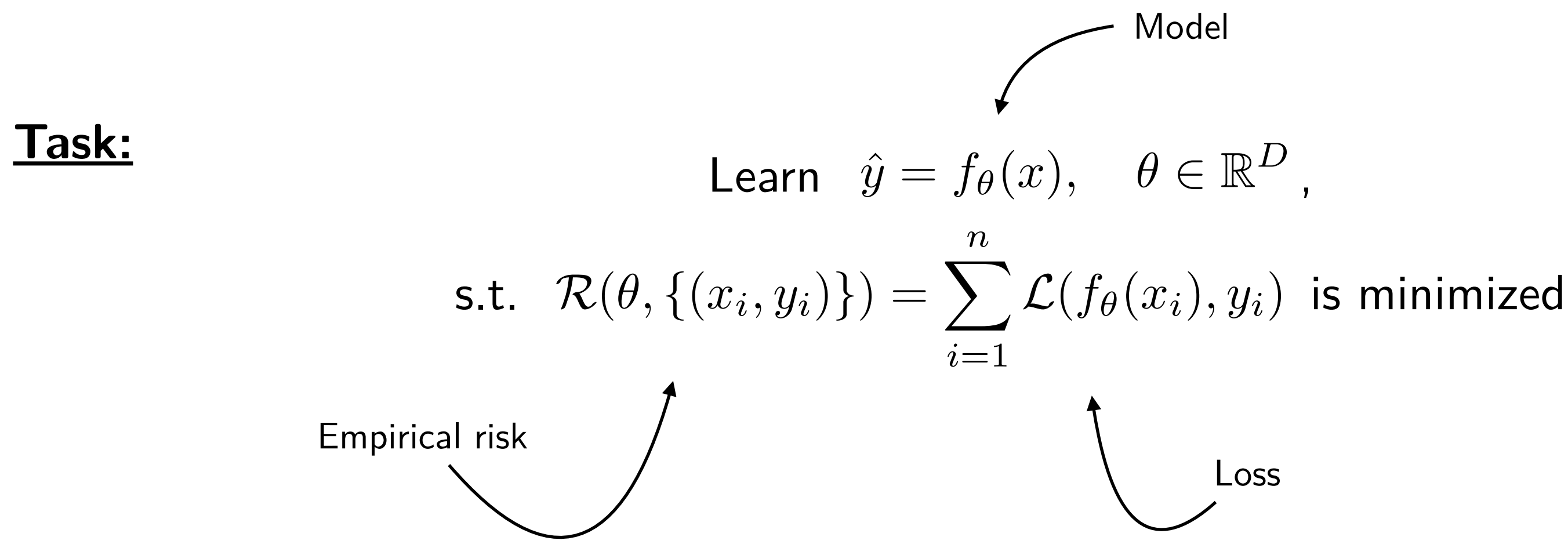
Our systems are *not even close* to real intelligence.

Learning a task

The set up for supervised learning



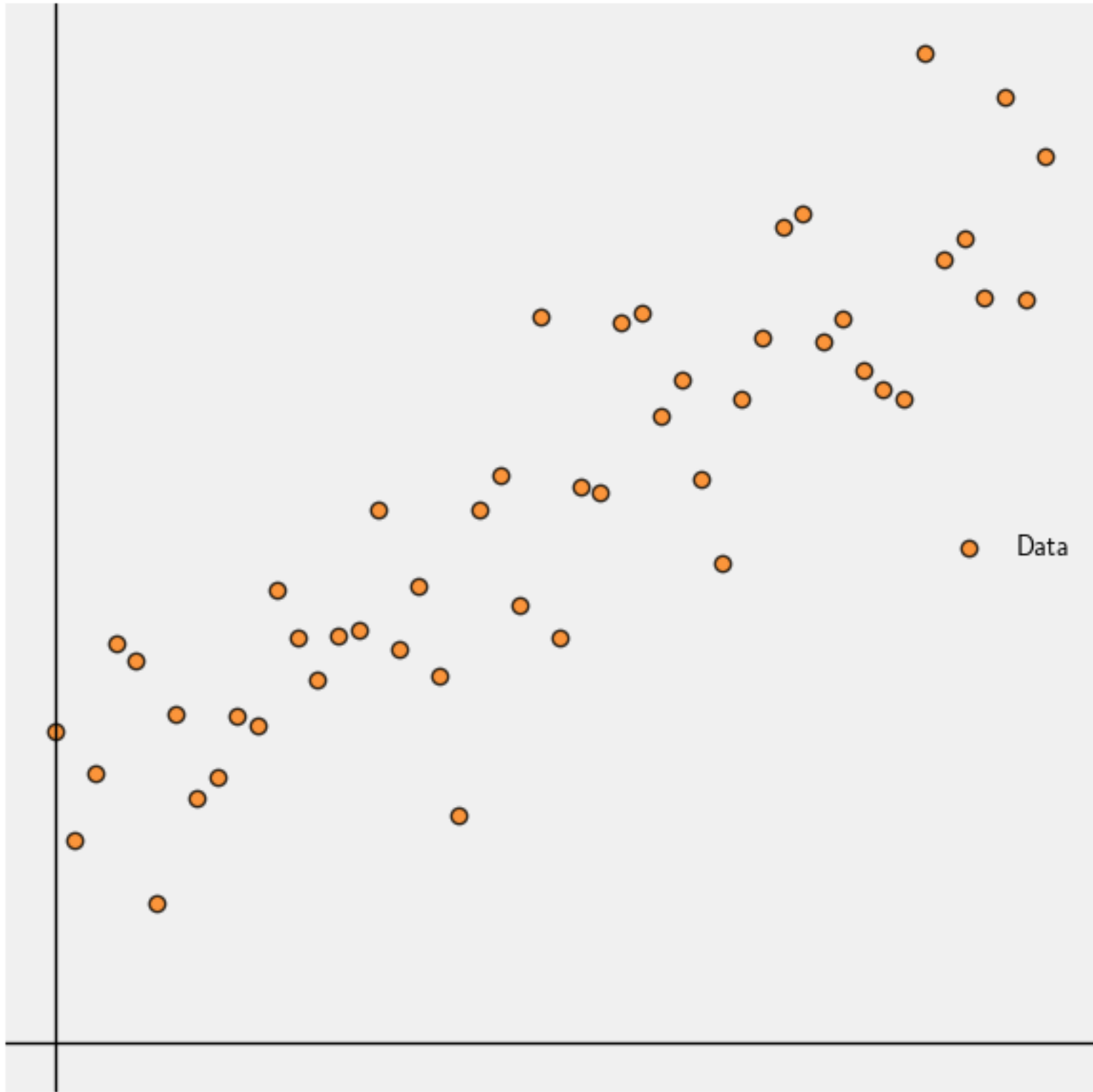
We want to *learn* the distribution μ



Learning a task

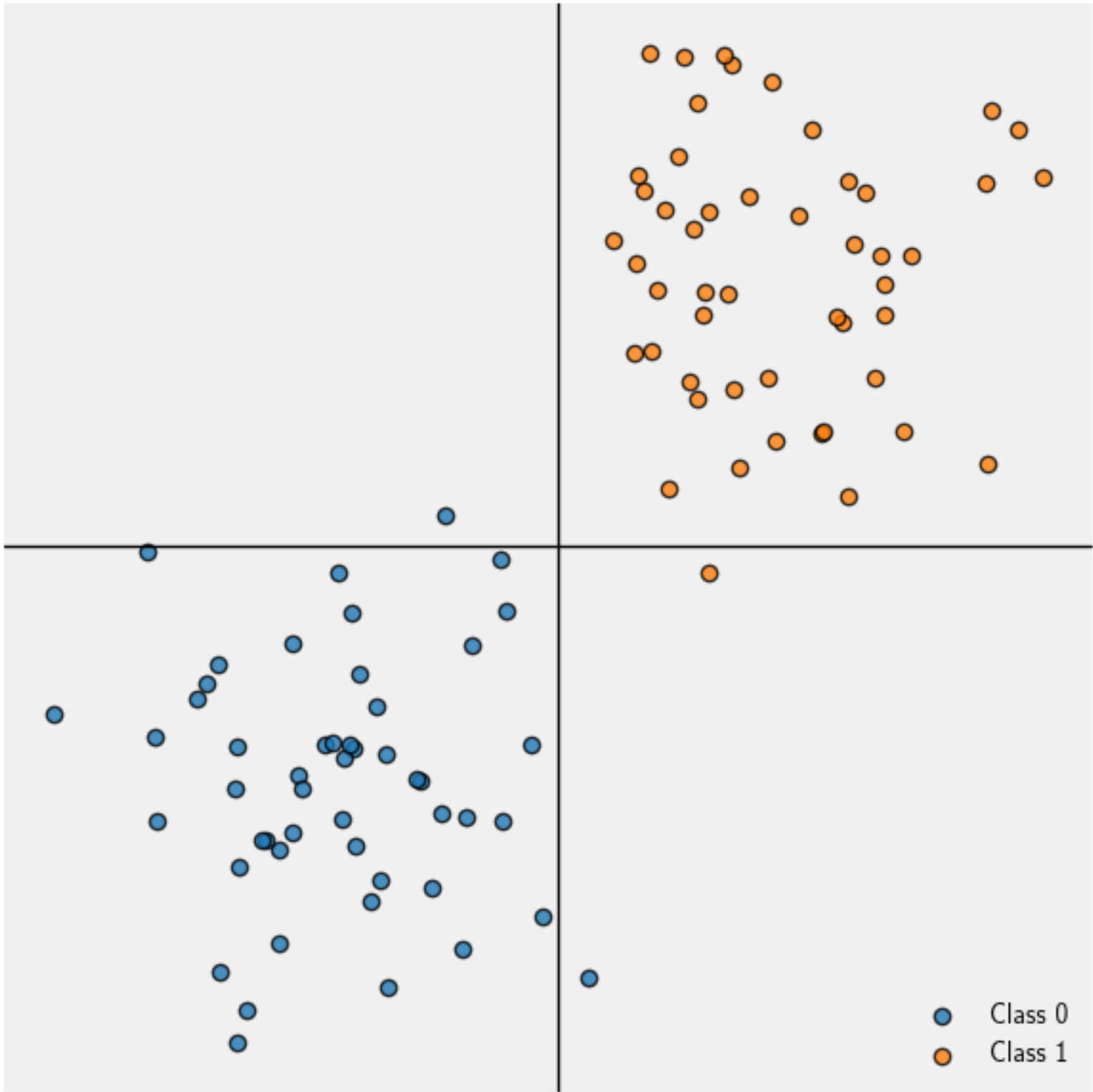
The set up for supervised learning

Regression



$X = \mathbb{R}, \quad Y = \mathbb{R}$

Classification

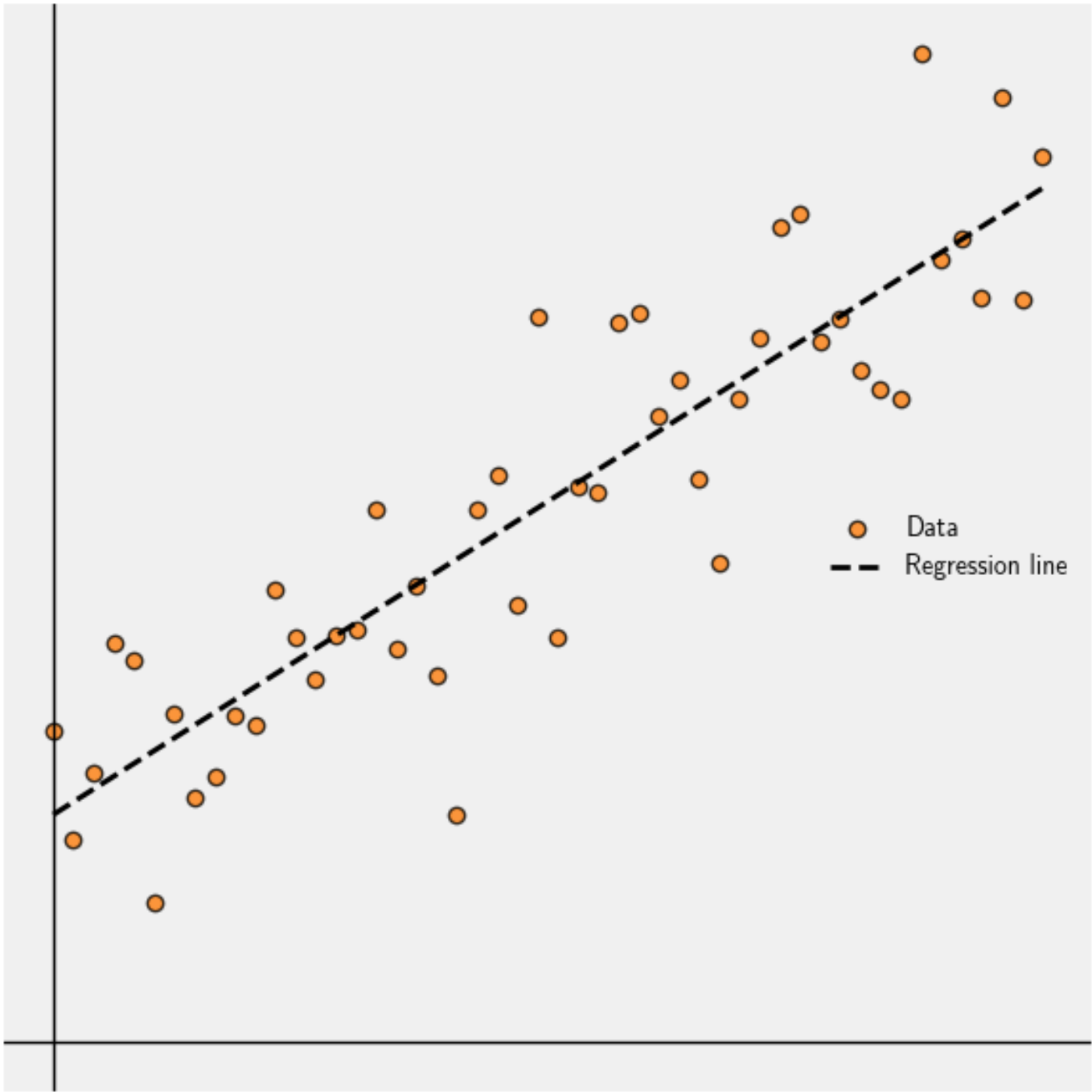


$X = \mathbb{R}^2, \quad Y = \{0, 1\}$

Learning a task

The set up for supervised learning

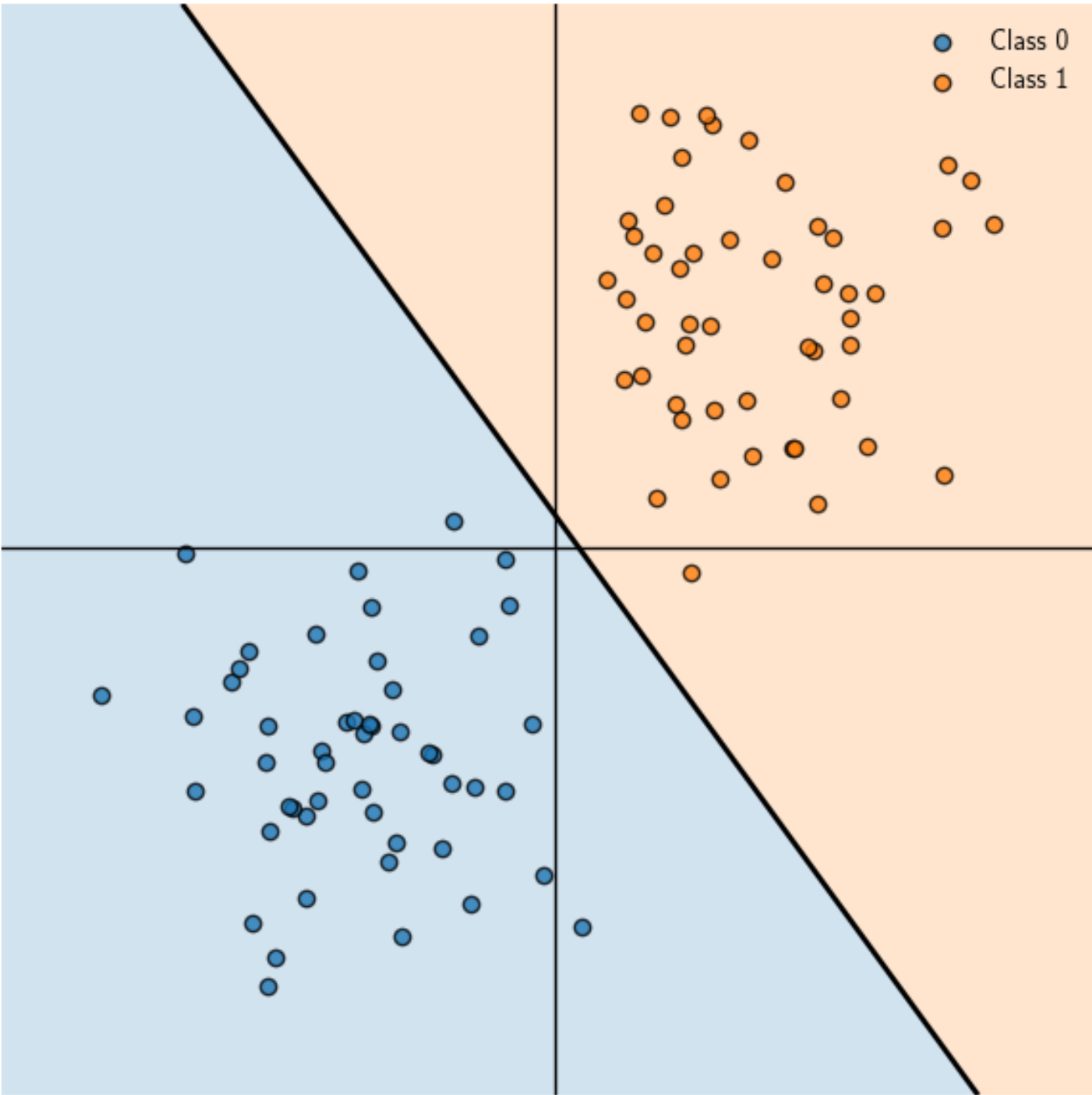
Regression



Linear regression $f_{m,b} = mx + b$

Mean Squared Error $\mathcal{R} = \text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

Classification



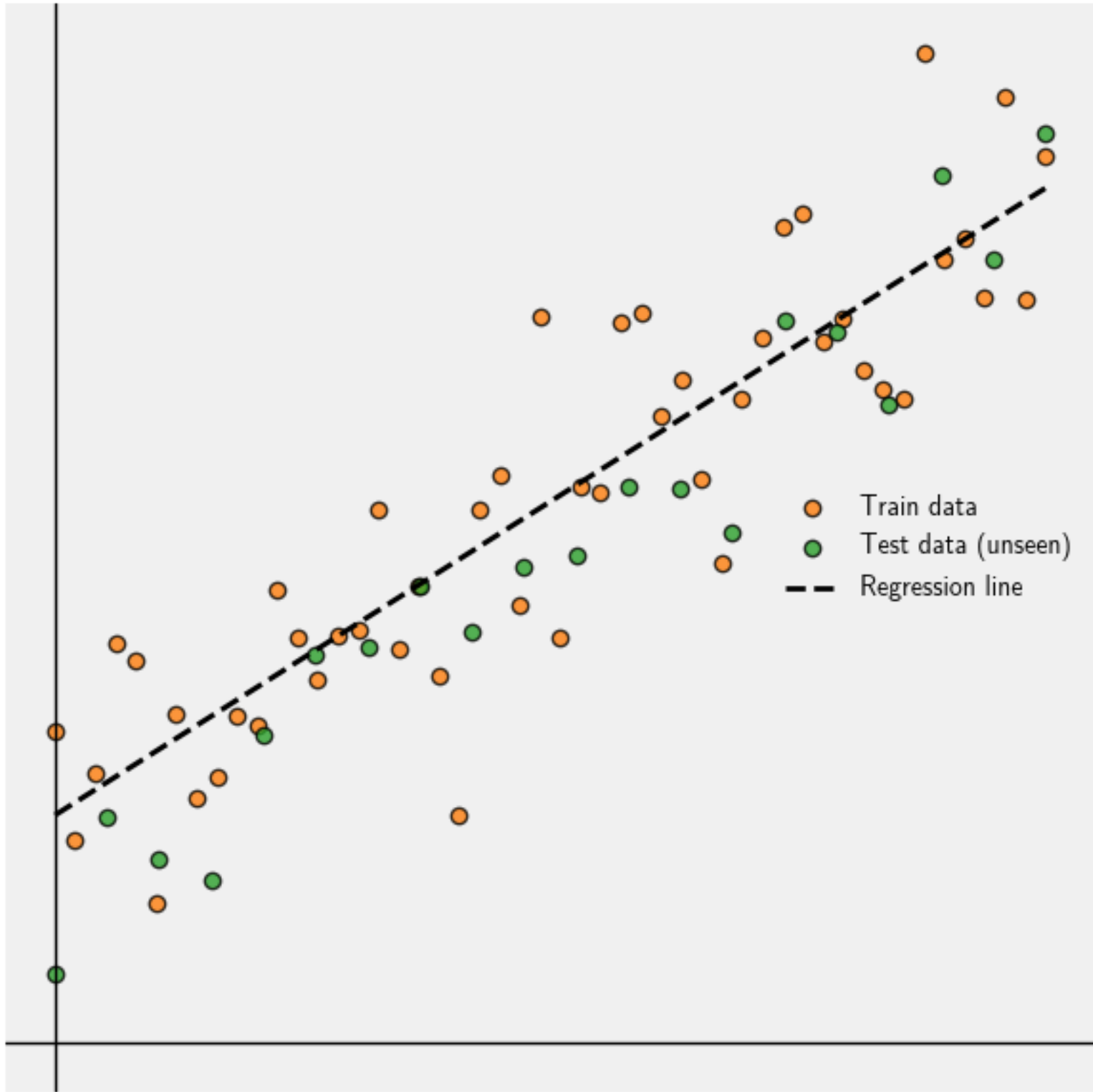
Logistic regression $f_{\mathbf{w},b}(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$

Cross-Entropy Loss $\mathcal{R} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$

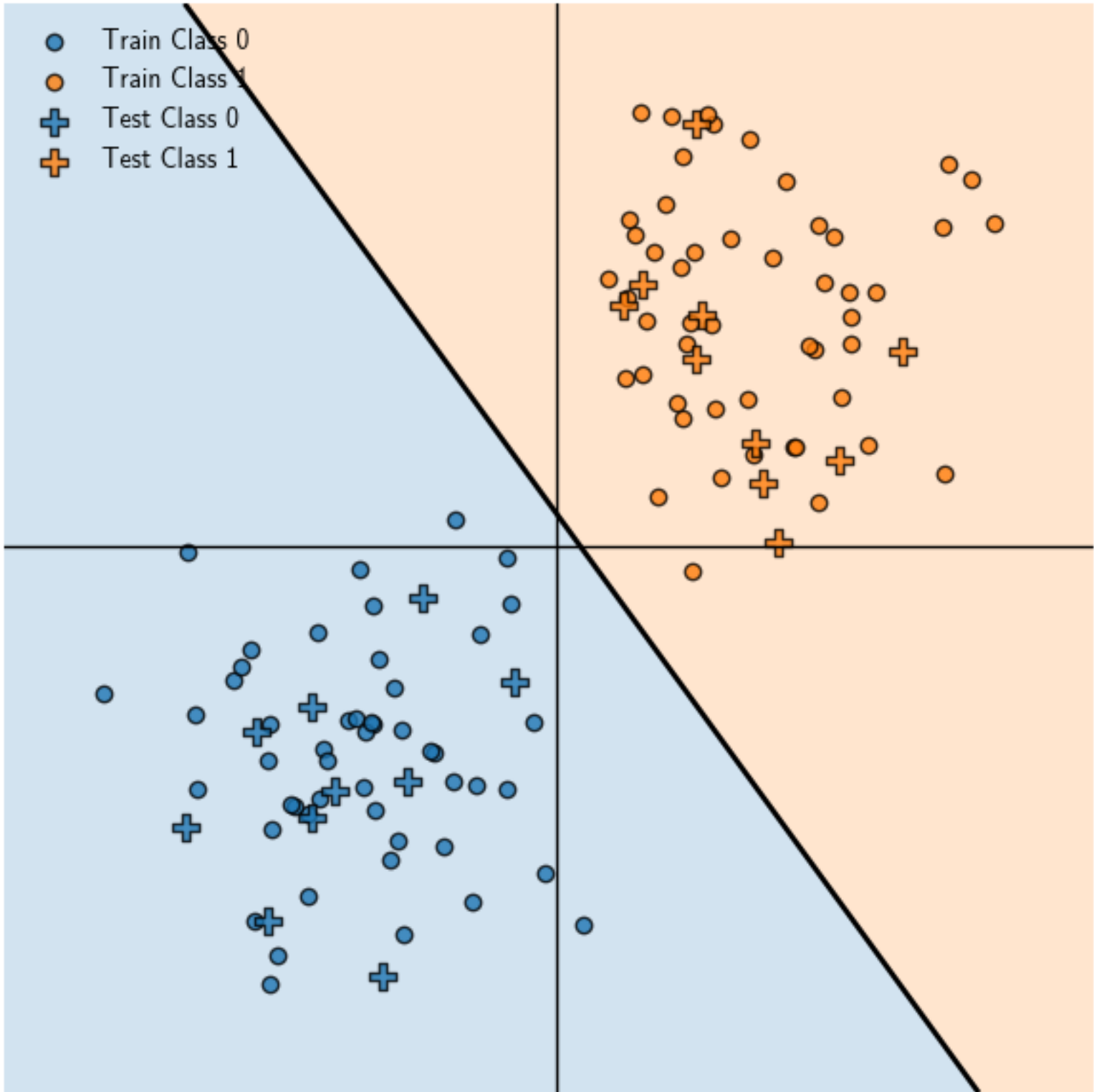
Learning a task

Generalization

Regression



Classification

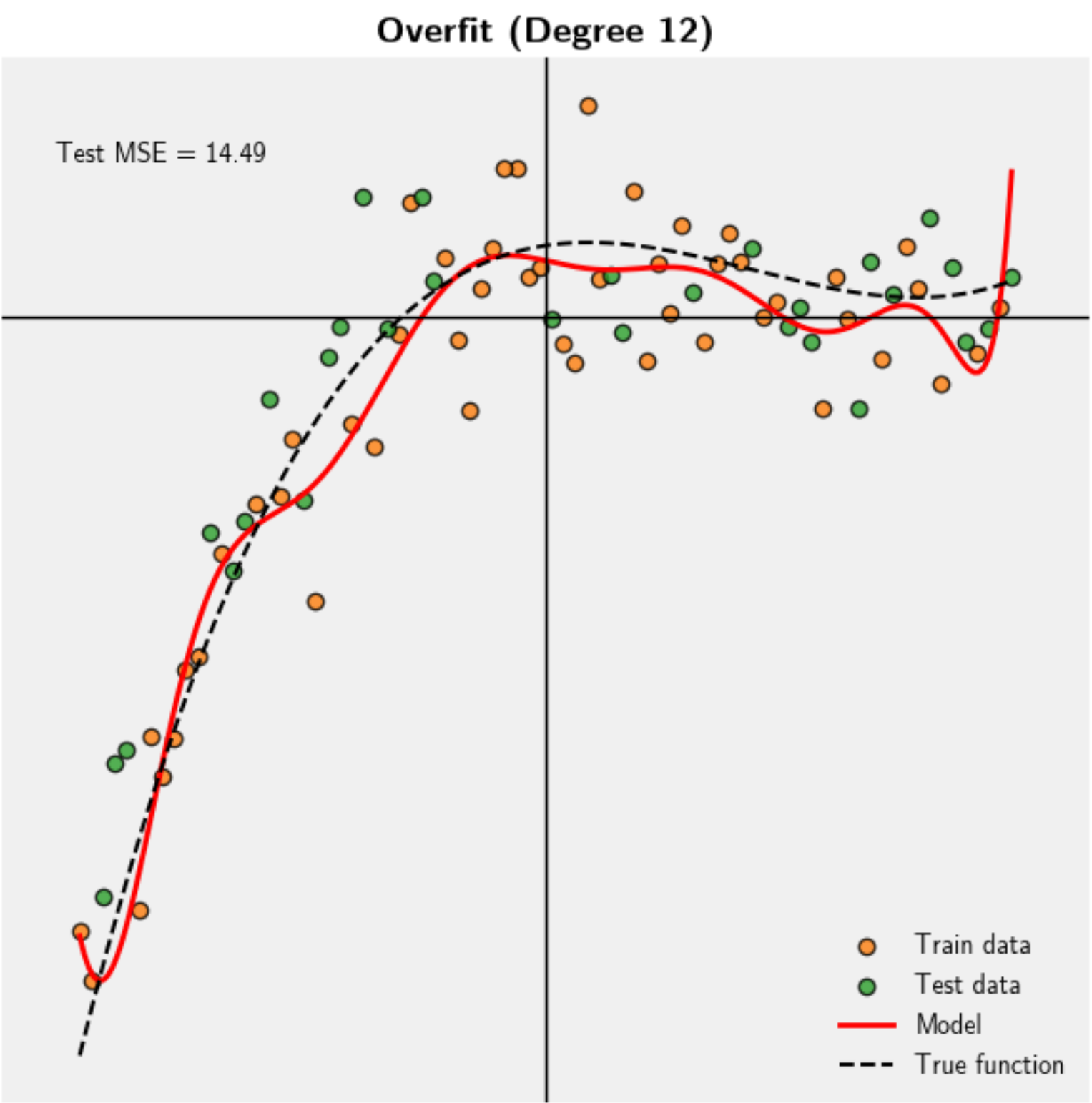
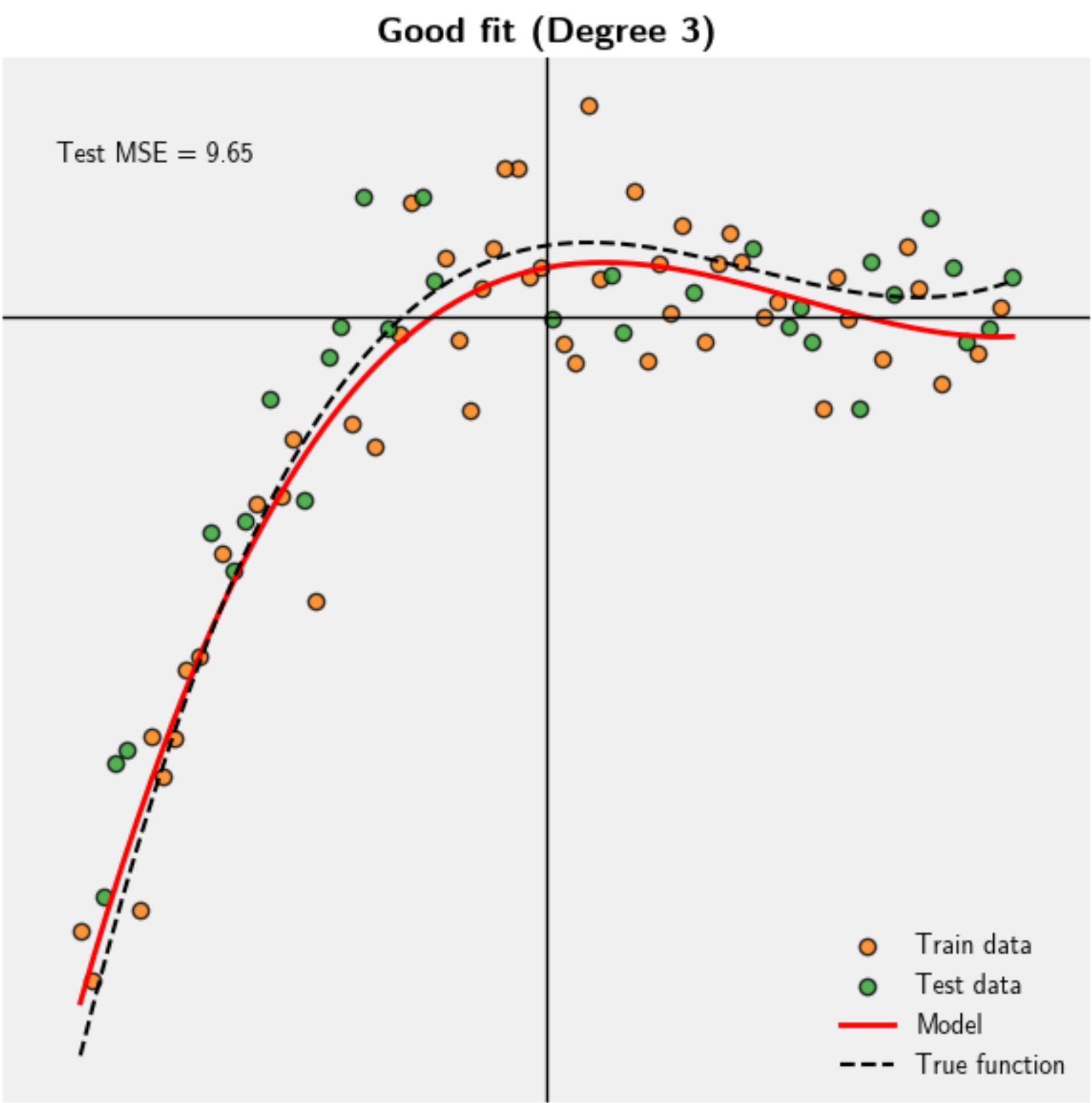
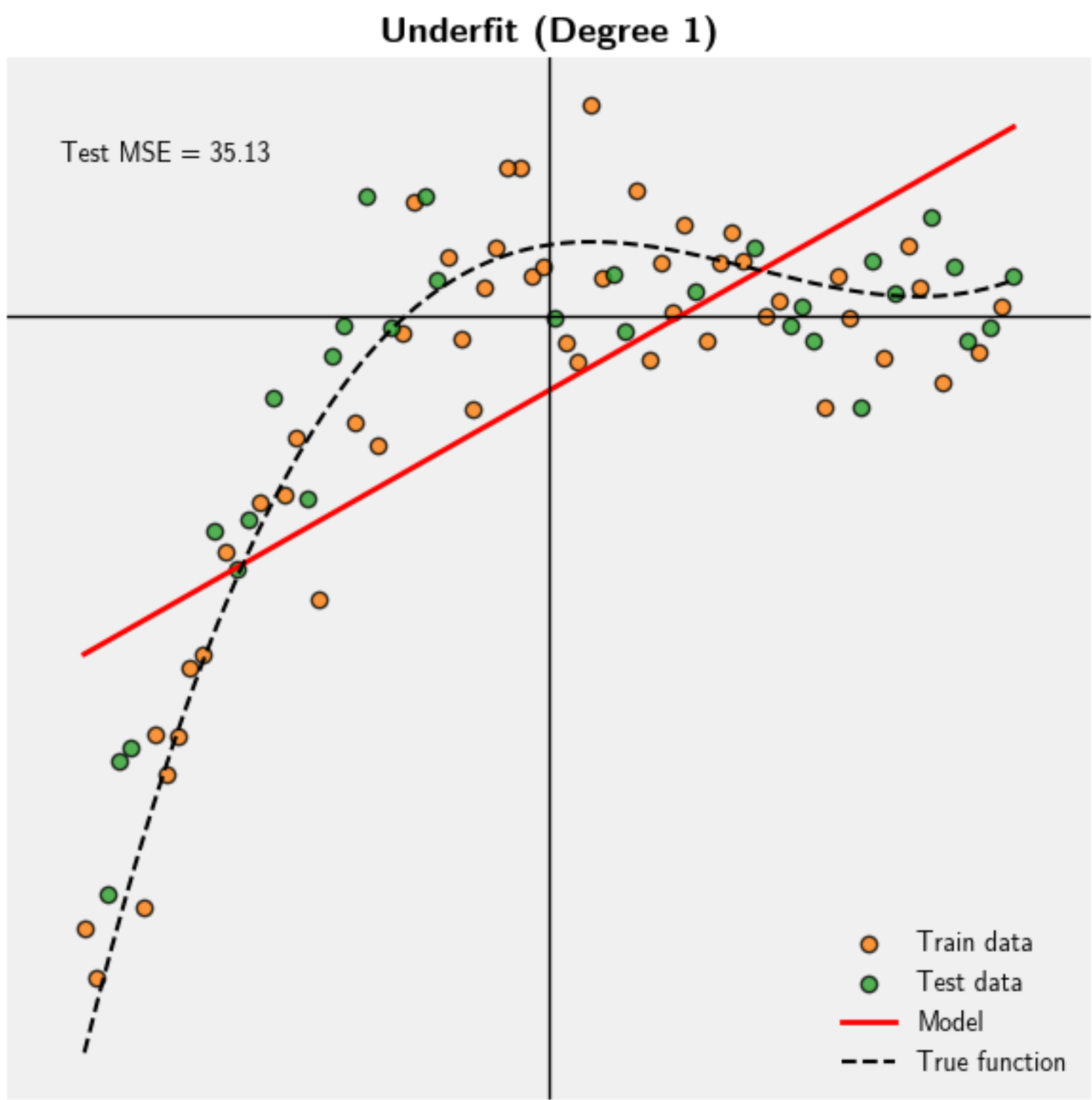


We seek that the model *generalizes* well: provides good predictions for unseen data

Learning a task

Underfitting and overfitting

$$\text{Fit } \frac{1}{2}x^3 - 2x^2 + x + 3$$

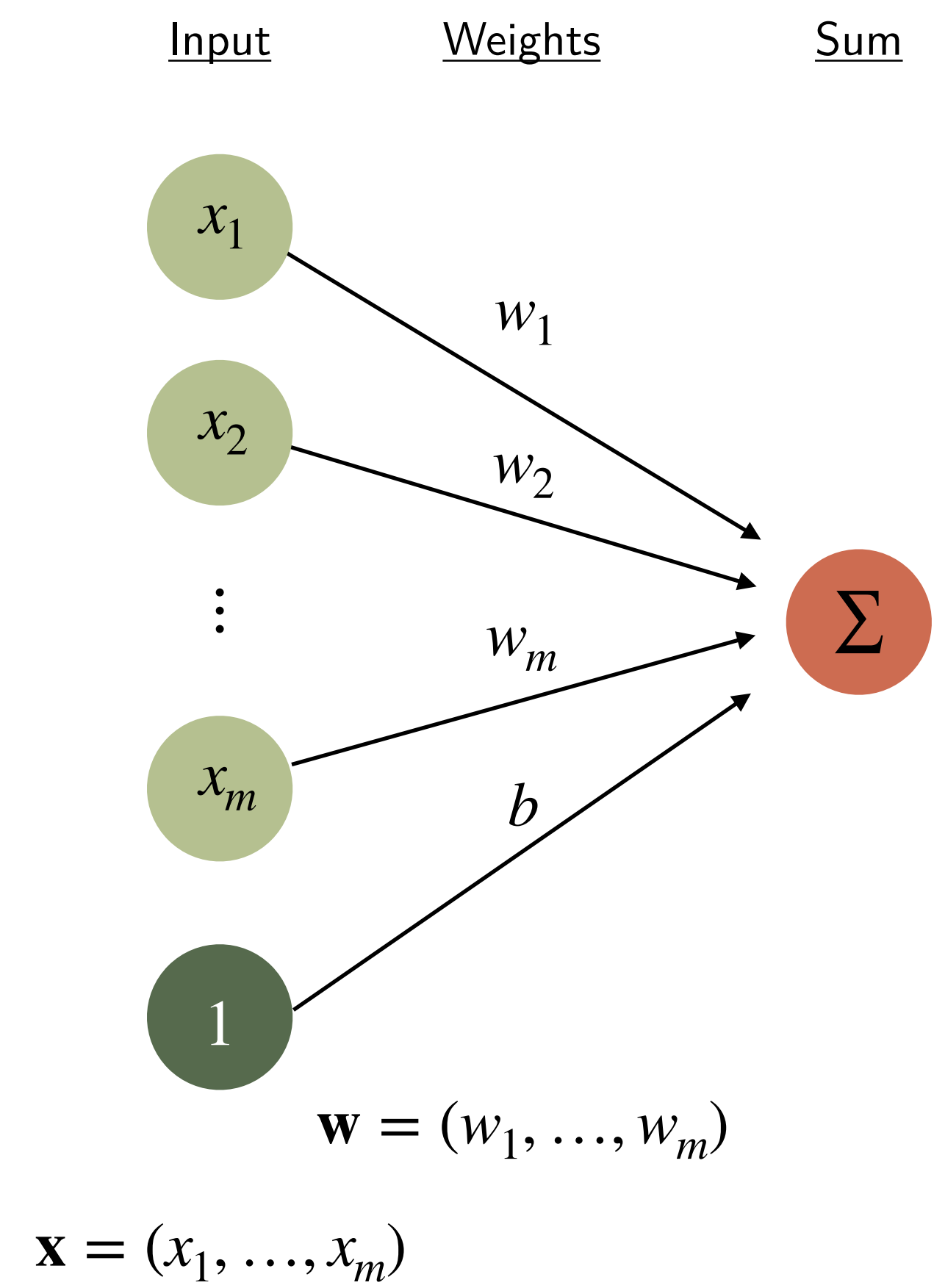


Neural Networks: the Multilayer Perceptron

Lecture 1 — Introduction to Deep Learning

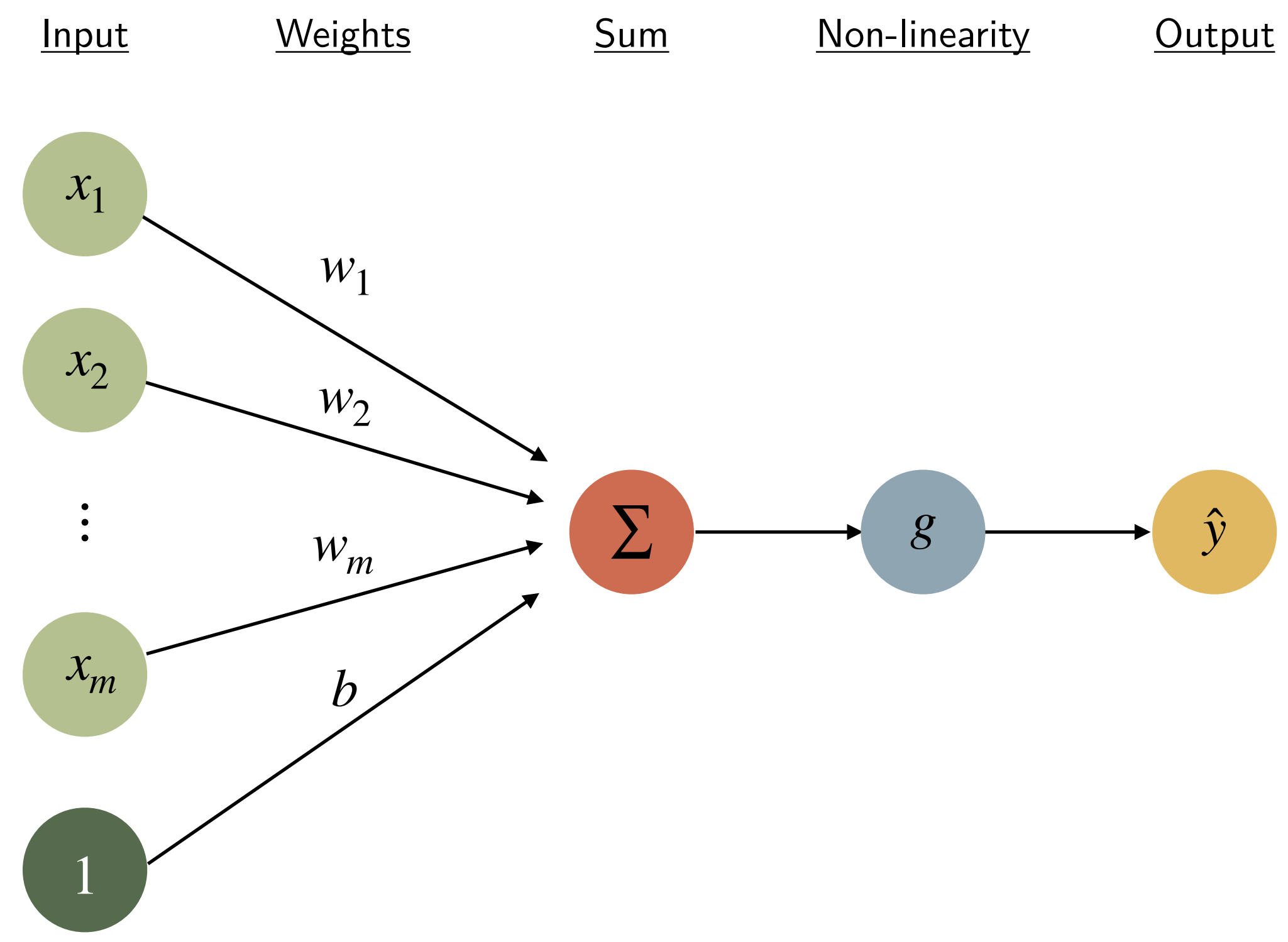
The Perceptron

The structural building block of deep learning



The Perceptron

The structural building block of deep learning



Non-linearity

Bias

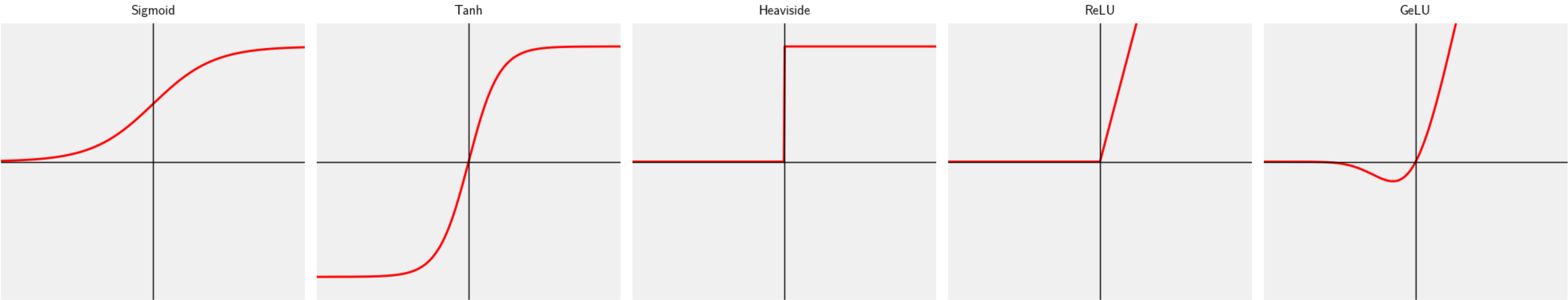
$$\hat{y} = g \left(\sum_{i=1}^m w_i x_i + b \right) = g \left(\mathbf{w} \cdot \mathbf{x} + b \right)$$

Weigths

$\mathbf{w} = (w_1, \dots, w_m)$ $\mathbf{x} = (x_1, \dots, x_m)$

Common activation functions

$$\hat{y} = g \left(\sum_{i=1}^m w_i x_i + b \right) = g \left(\mathbf{w} \cdot \mathbf{x} + b \right)$$



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g(z) = \begin{cases} 0 & \text{if } z < 0 \\ 0.5 & \text{if } z = 0 \\ 1 & \text{if } 1x > 0 \end{cases}$$

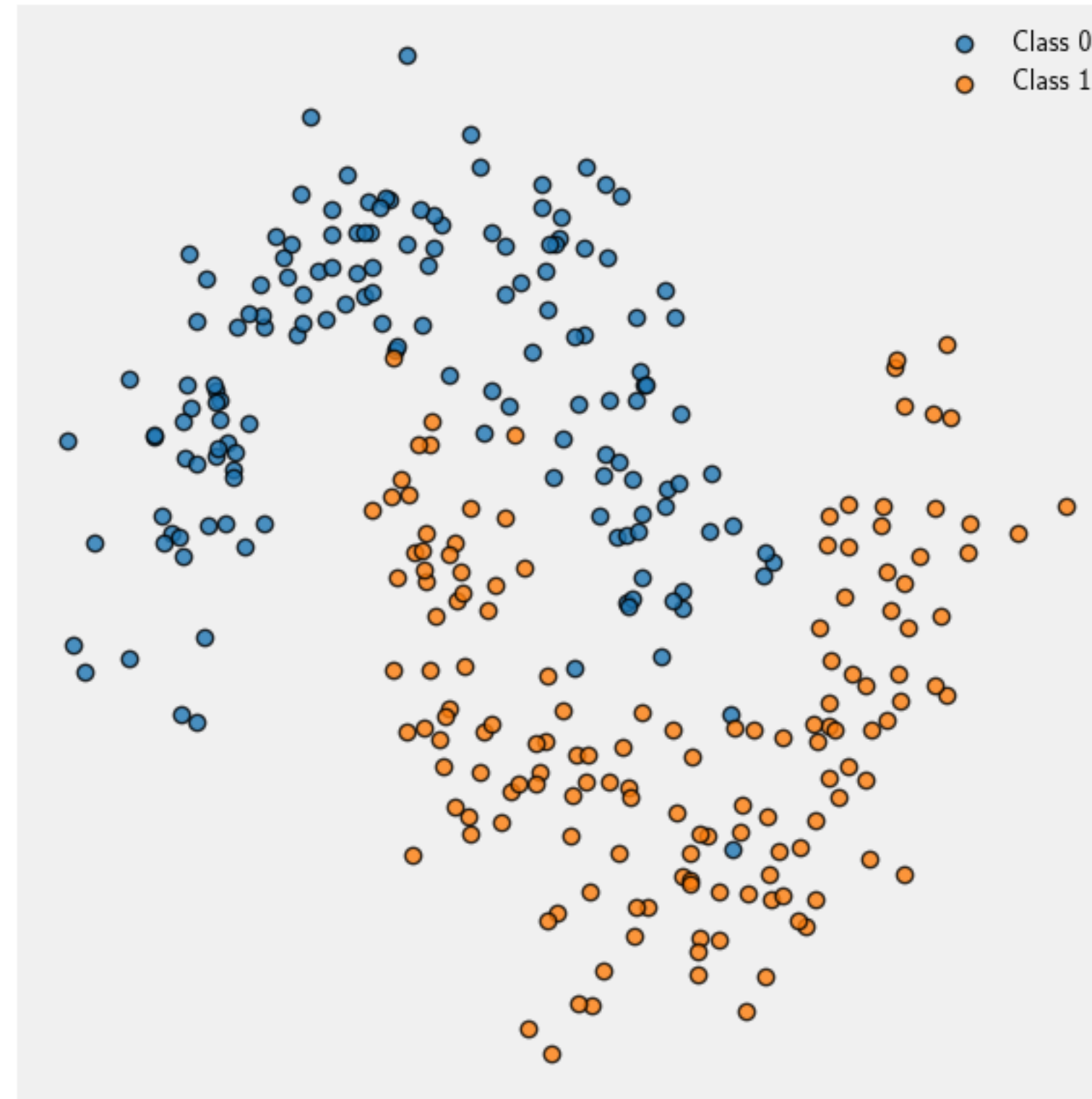
$$g(z) = \max(0, z)$$

$$g(z) = z \cdot \Phi(z)$$

Cumulative distribution function of the standard normal distribution

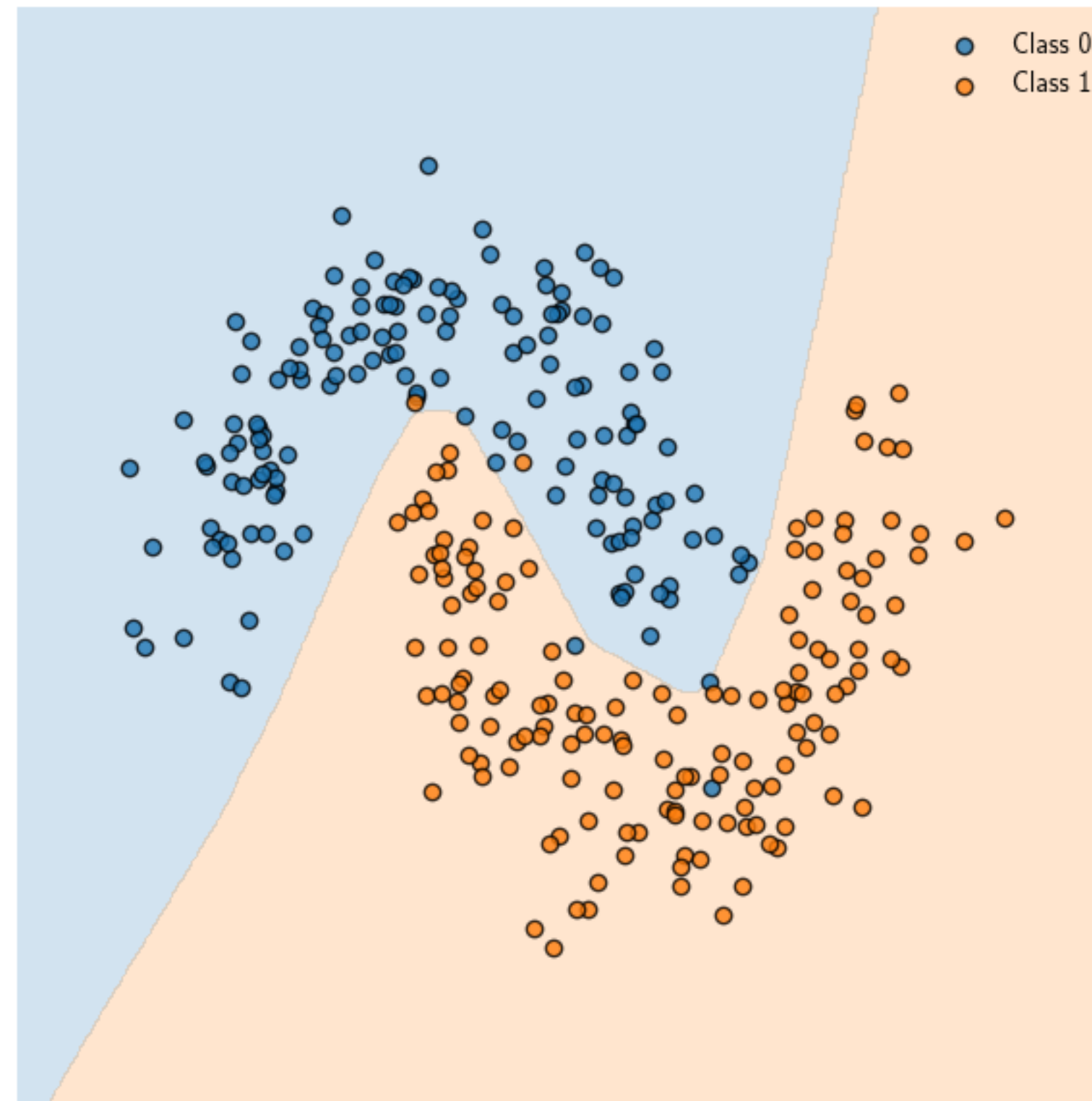
Note: they must be non-linear

Why non linear activations?



What happens if the data is non linearly separable?

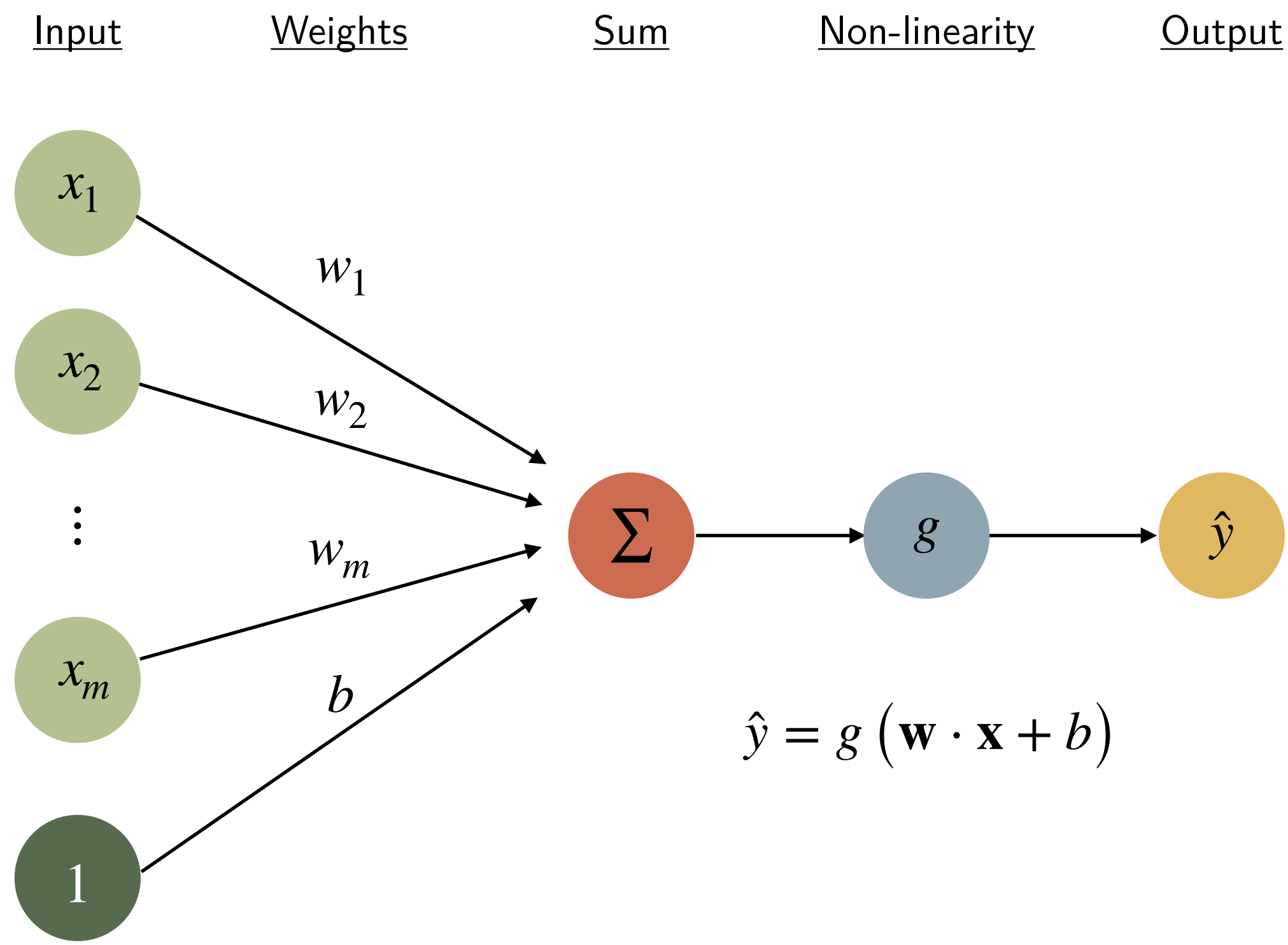
Why non linear activations?



Non-linearities allow us to approximate arbitrarily complex functions

The Perceptron

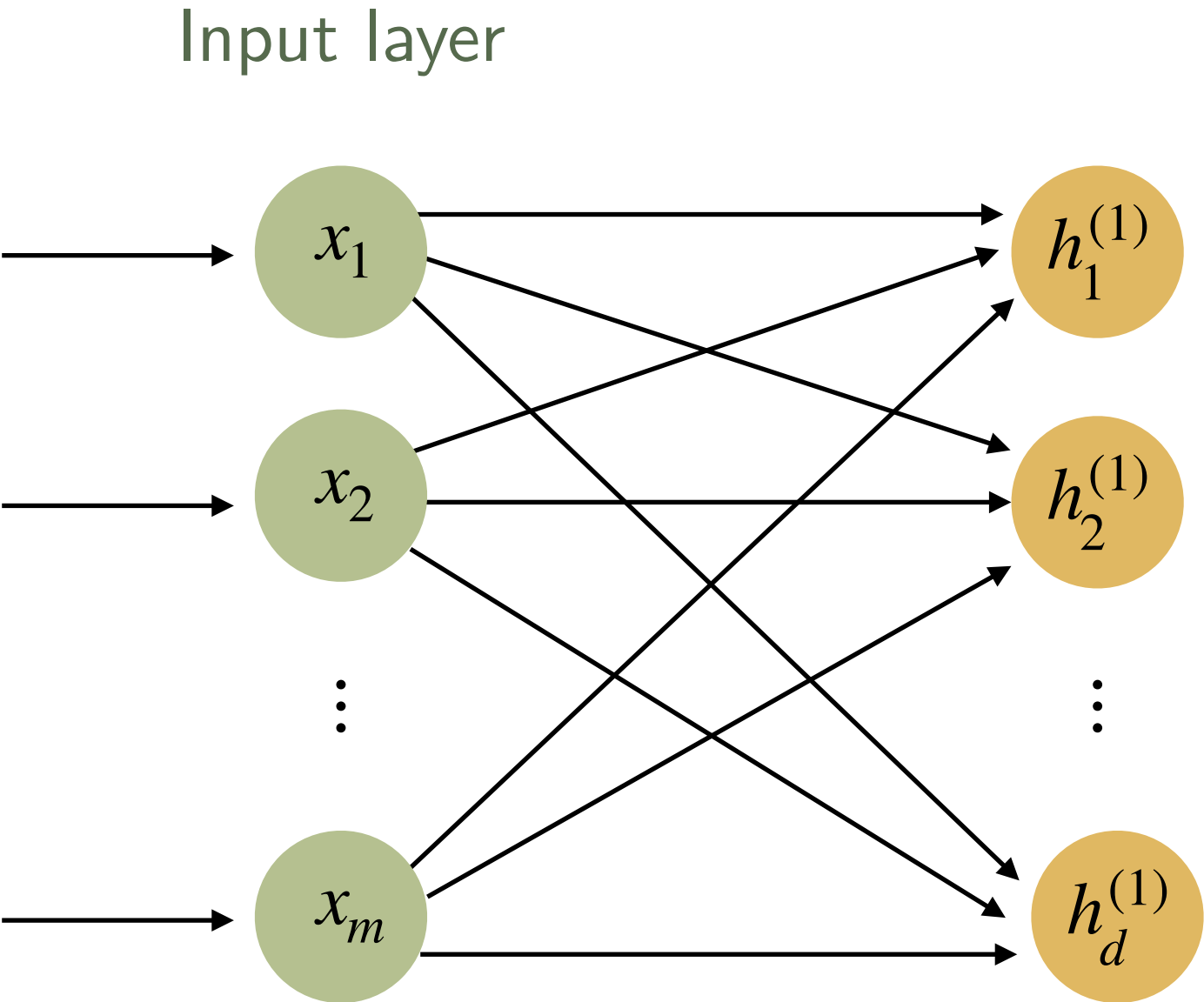
The structural building block of deep learning



The perceptron is a *single* neuron, which is *activated* based on the output of an *activation function*, evaluated on a weighted sum of inputs

The Multi-Layer Perceptron

Components



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

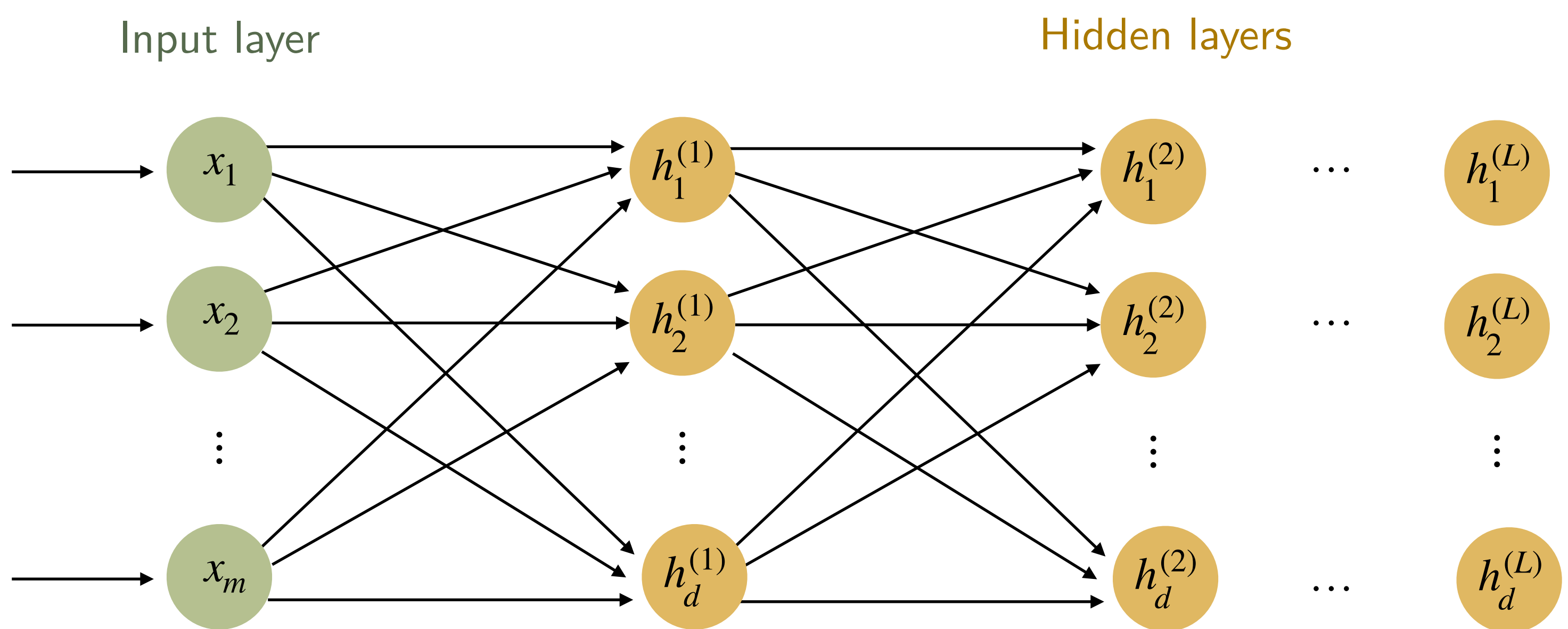
$$\mathbf{h}^{(1)} = (h_1^{(1)}, \dots, h_d^{(1)})$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & \dots & w_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(1)} & \dots & w_{d,m}^{(1)} \end{pmatrix}$$

$$\mathbf{b}^{(1)} = (b_1^{(1)}, \dots, b_d^{(1)})$$

The Multi-Layer Perceptron

Components



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

$$\mathbf{h}^{(1)} = (h_1^{(1)}, \dots, h_d^{(1)})$$

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & \dots & w_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(1)} & \dots & w_{d,m}^{(1)} \end{pmatrix}$$

$$\mathbf{b}^{(1)} = (b_1^{(1)}, \dots, b_d^{(1)})$$

$$\mathbf{h}^{(l)} = g \left(\mathbf{W}^{(l)} \cdot \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \right)$$

$$\mathbf{h}^{(l)} = (h_1^{(l)}, \dots, h_d^{(l)})$$

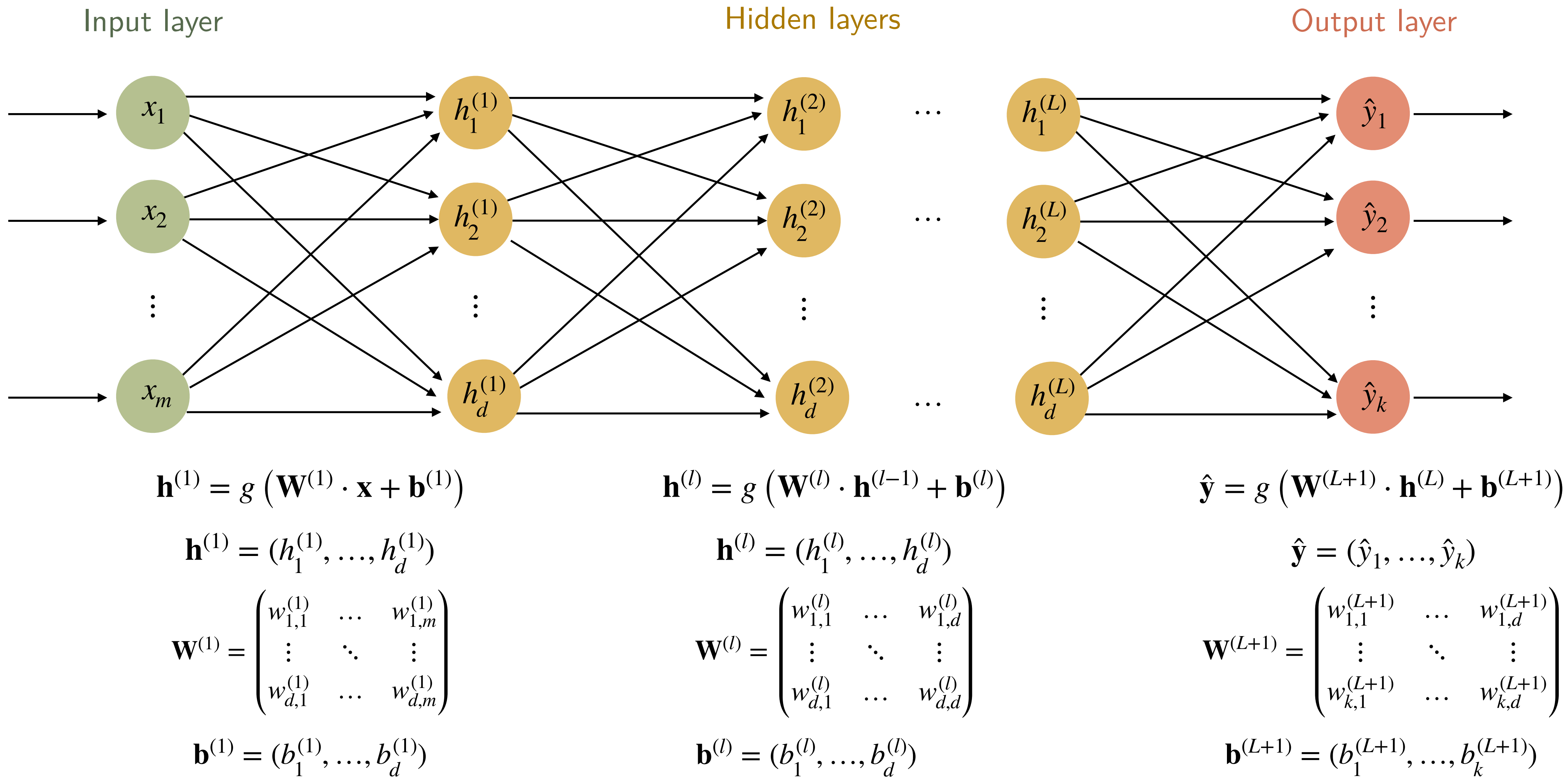
$$\mathbf{W}^{(l)} = \begin{pmatrix} w_{1,1}^{(l)} & \dots & w_{1,d}^{(l)} \\ \vdots & \ddots & \vdots \\ w_{d,1}^{(l)} & \dots & w_{d,d}^{(l)} \end{pmatrix}$$

$$\mathbf{b}^{(l)} = (b_1^{(l)}, \dots, b_d^{(l)})$$

The Multi-Layer Perceptron

Components

$$f_{\theta} : \mathbb{R}^m \rightarrow \mathbb{R}^k, \quad \theta \in \mathbb{R}^D$$



Universal Function Approximation

Why MLPs work for non-linear activations

Theorem (Universal function approximation)

Let $\sigma \in C(\mathbb{R}, \mathbb{R})$ be a non-polynomial activation function. For every $m, k \in \mathbb{N}$, every compact subset $K \subset \mathbb{R}^m$, every function $f \in C(K, \mathbb{R}^k)$ and $\epsilon > 0$, there exists $d \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{d \times m}$, $\mathbf{b} \in \mathbb{R}^d$, and $\mathbf{C} \in \mathbb{R}^{k \times d}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \epsilon,$$

where $g(x) = \mathbf{C}\sigma(\mathbf{A}x + \mathbf{b})$.

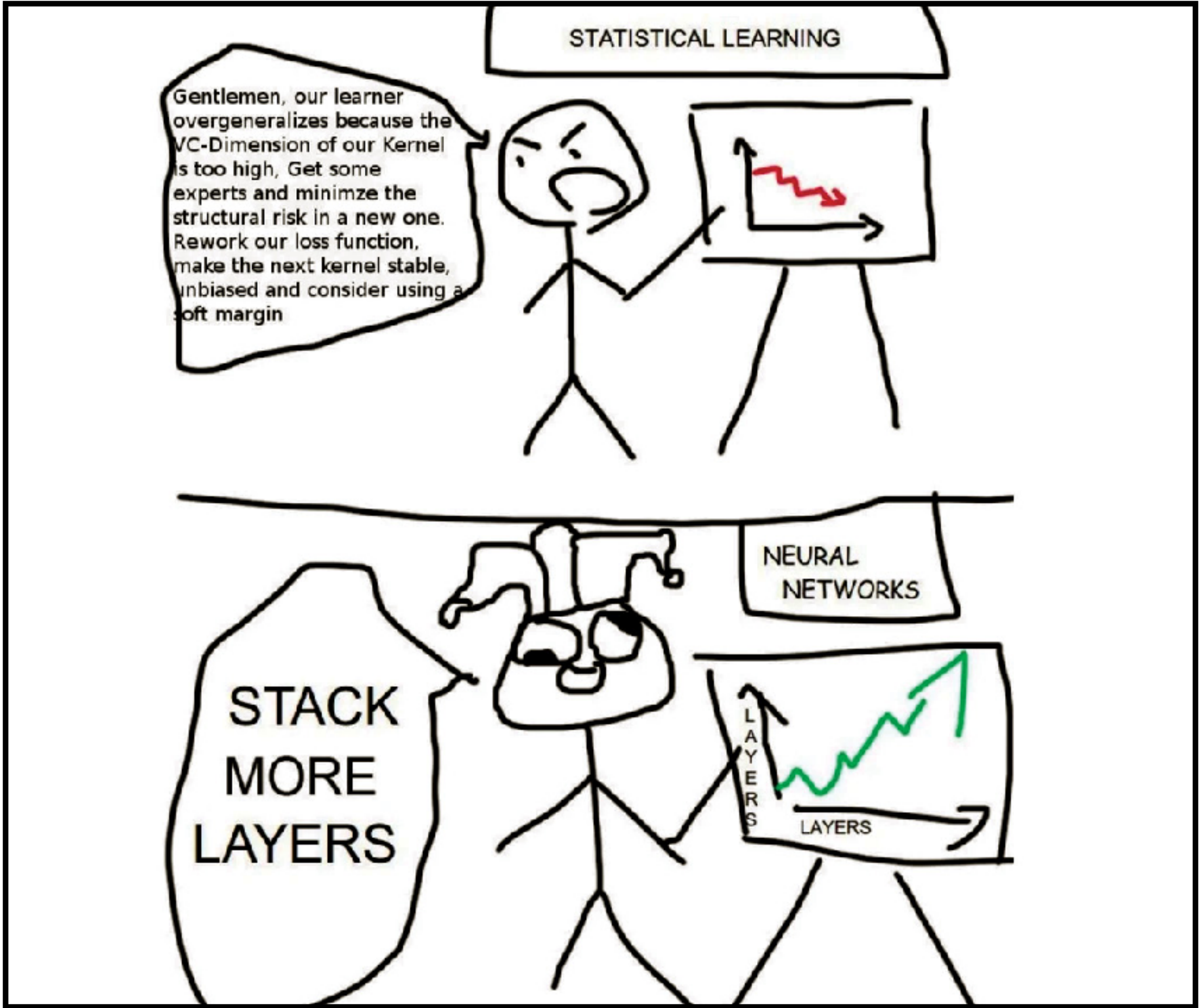
Implication

Theoretically we can approximate any continuous function if we make no restrictions on the *width* d of our neural network.

A. Pinkus, “Approximation theory of the MLP model in neural networks”, *Acta Numerica* 8, 1999, pp.143-195

Universal Function Approximation

Why MLPs work for non-linear activations



Training a neural network

Lecture 1 — Introduction to Deep Learning

Inés García Redondo — May 11, 2026

Training a neural network

Gradient descent

Training point

Input: $f_{\theta} : \mathbb{R}^m \rightarrow \mathbb{R}^k$, $\mathcal{L}(\mathbf{x}, \theta)$

1. Initialize weights randomly: θ_0
2. Loop until convergence:
 1. Compute gradient: $\frac{\partial \mathcal{L}}{\partial \theta} \in \mathbb{R}^D$
 2. Update weights:

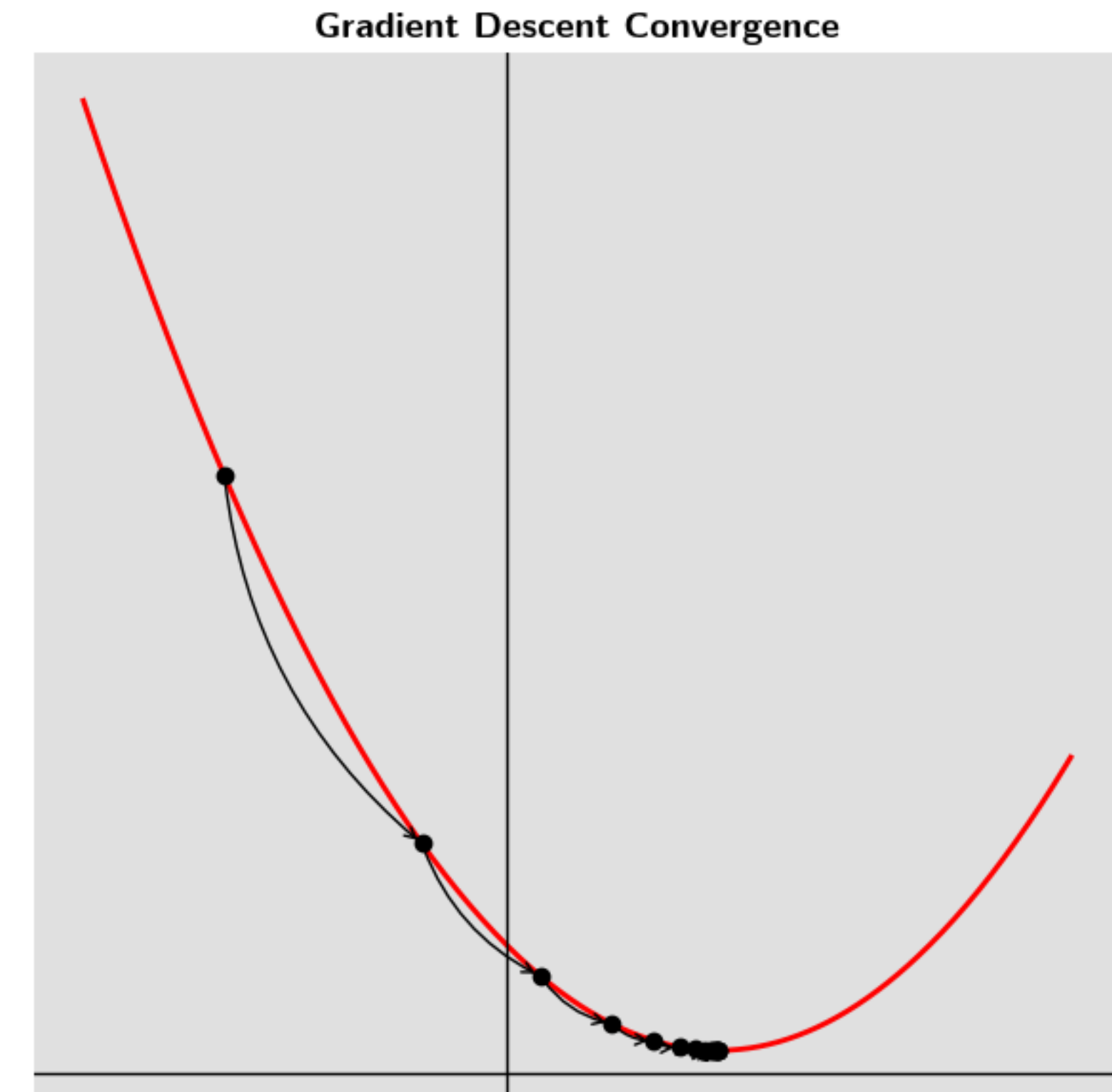
Learning rate

Single sample: $\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{x}, \theta_t)$

All training data: $\theta_{t+1} \leftarrow \theta_t + \eta \frac{1}{n} \sum_{i=1}^n \frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{x}_i, \theta_t)$

Output: Optimal weights after convergence θ_T

Computational cost: $O(n \cdot D)$



Training a neural network

Stochastic gradient descent

Input: $f_{\theta} : \mathbb{R}^m \rightarrow \mathbb{R}^k$, $\mathcal{L}(\mathbf{x}, \theta)$

1. Initialize weights randomly: θ_0

2. Loop until convergence:

1. Compute gradient: $\frac{\partial \mathcal{L}}{\partial \theta} \in \mathbb{R}^D$

2. Update weights:

Single sample: $\theta_{t+1} \leftarrow \theta_t - \eta \frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{x}, \theta_t)$

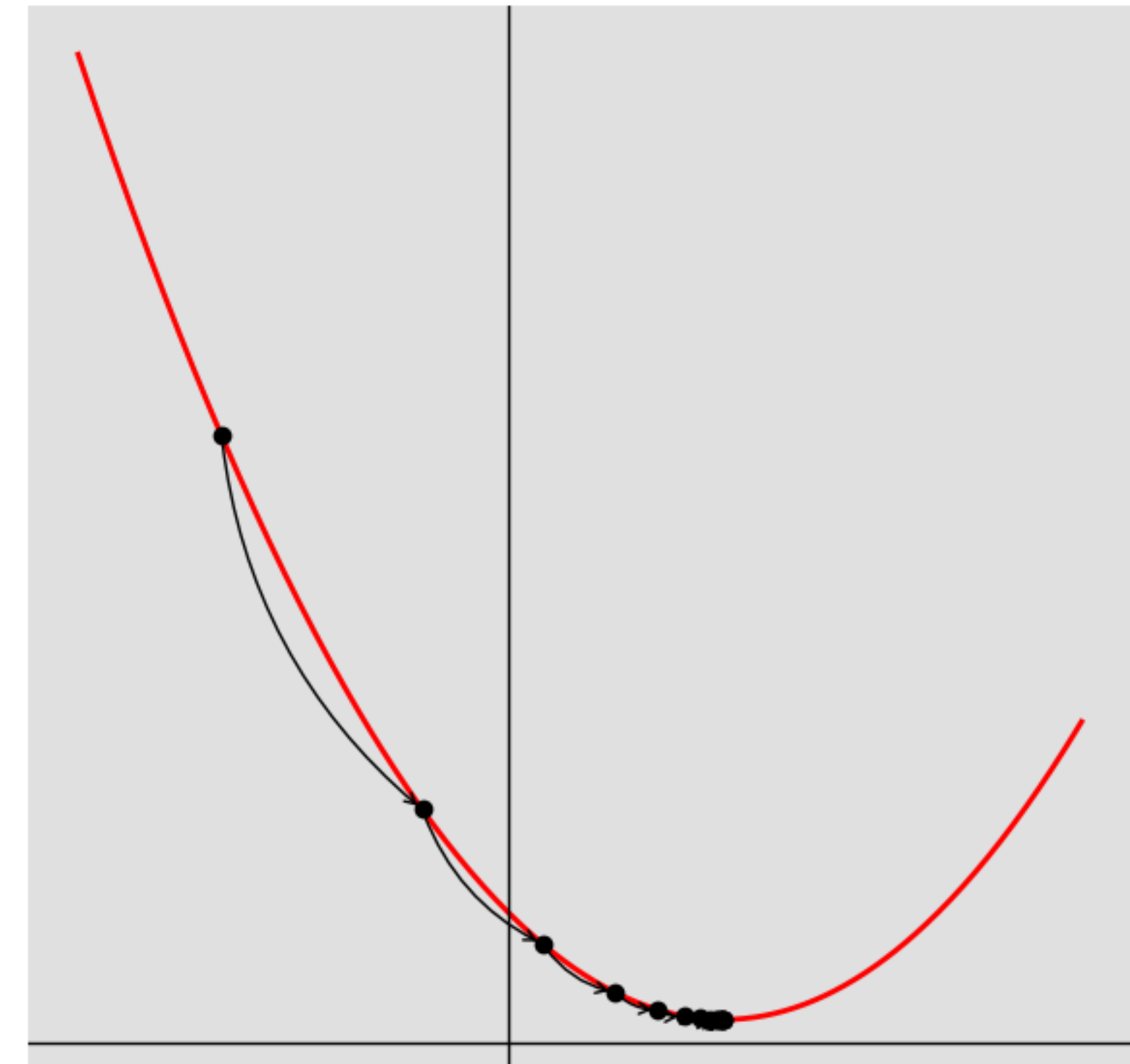
Batch of data: $\theta_{t+1} \leftarrow \theta_t + \eta \frac{1}{B} \sum_{i=1}^B \frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{x}_i, \theta_t)$

B is called the **batch size**

Output: Optimal weights after convergence θ_T

Computational cost: $O(B \cdot D)$

Gradient Descent Convergence



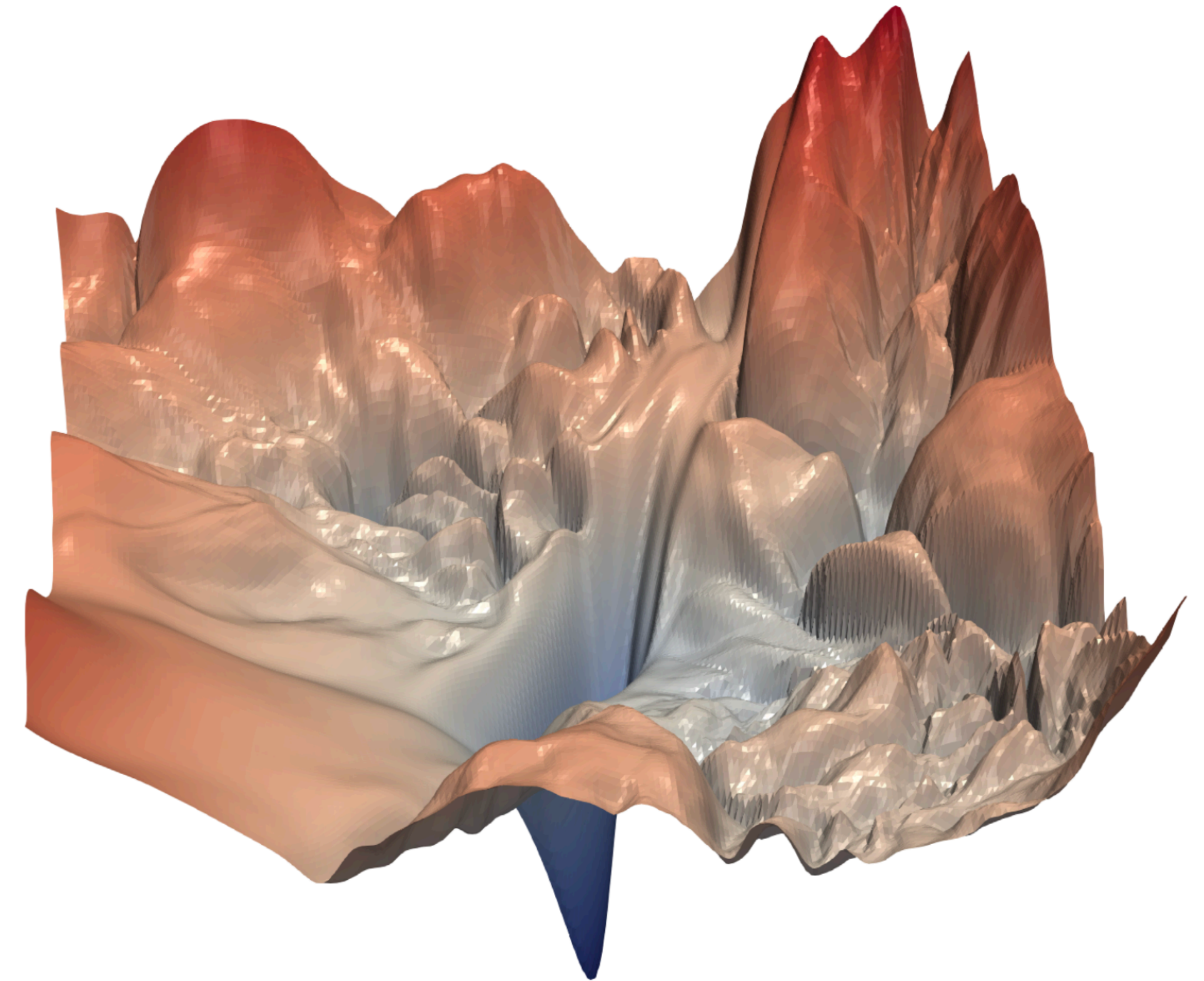
Training a neural network

Stochastic gradient descent

Wins from doing *stochastic* gradient descent

- **Memory**: you only need to hold B samples in memory at once, not all N
- **Faster convergence in practice**: the noise introduced by subsampling acts as a regularizer and often leads to convergence in fewer epochs
- **Parallelism**: the batch size can be tuned to saturate GPU memory and use resources more efficiently

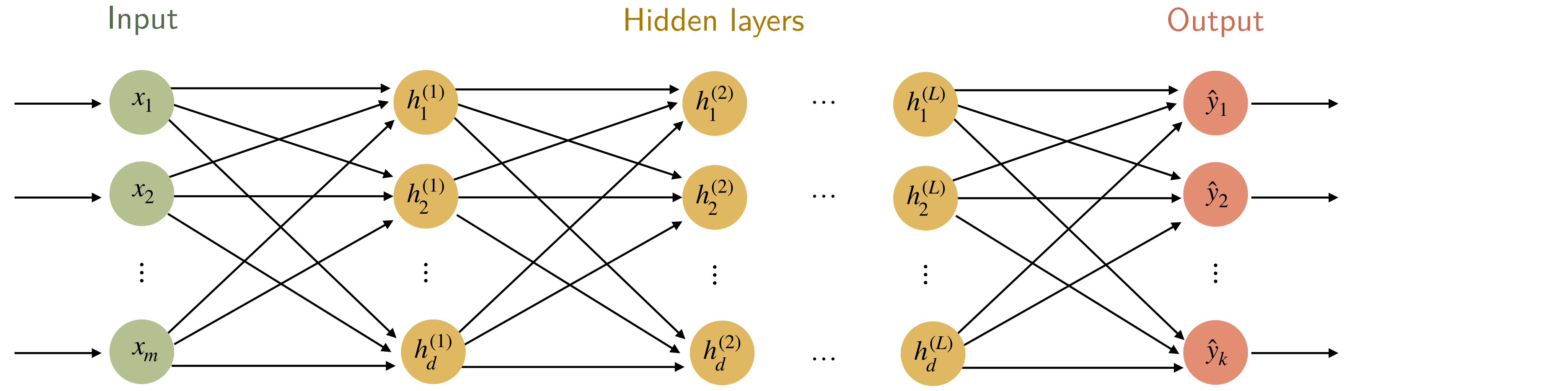
There is an additional, subtler motivation: SGD tends to **find flatter minima that generalize better**. The reason why is still not well understood, but an intuition is that noisy gradients help scape *narrow* local minima that usually have poorer generalization.



*H. Li et al., Visualizing the
Loss Landscape of Neural Nets,
NeurIPS 2018*

Training a neural network

Backpropagation



Want the gradients: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}$

Use chain rule: $\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \cdot \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$
 $\mathbf{h}^{(l)} = g(\mathbf{z}^{(l)})$

Error signal $\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} \cdot \left(\mathbf{h}^{(l-1)} \right)$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

Multiplying component-wise

$$\delta^{(L)} = \nabla_{\hat{\mathbf{y}}} \mathcal{L} \odot g'(\mathbf{z}^{(L)})$$
$$\nabla_{\hat{\mathbf{y}}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \hat{y}_1}, \dots, \frac{\partial \mathcal{L}}{\partial \hat{y}_k} \right)$$
$$g'(\mathbf{z}^{(L)}) = \left(g'(z_1^{(L)}), \dots, g'(z_n^{(L)}) \right)$$

$$\delta^{(l)} = \left(\mathbf{W}^{(l+1)} \right)^\top \delta^{(l+1)} \odot g'(\mathbf{z}^{(L)})$$

Careful with *vanishing* and *exploding* gradients!

Learning from images

Lecture 1 — Introduction to Deep Learning

Learning from images

What the computer sees

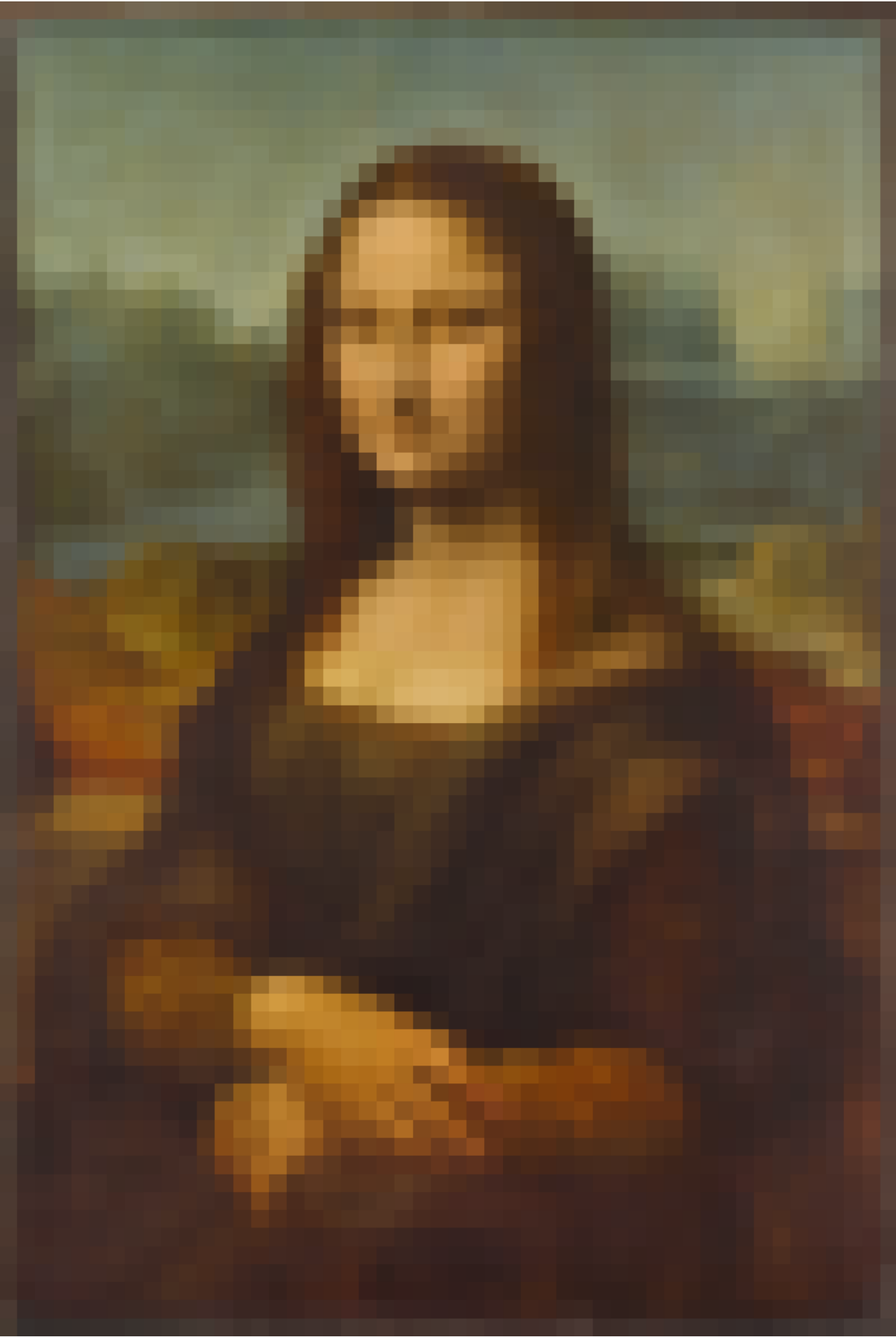


96	98	97	98	101	102	98	95	94	96	95	95	96	94	90	94	89	86	85	83	83	83	83	83	84	83	83	80	79	78
100	116	112	112	111	110	112	109	105	106	106	106	108	104	106	108	108	110	109	108	107	108	109	110	111	114	114	115	116	96
100	125	121	119	116	117	116	114	110	110	111	114	113	110	112	114	114	115	118	118	116	117	119	119	120	125	125	128	127	100
96	128	125	124	124	125	122	119	117	115	117	118	116	119	122	124	120	118	121	124	121	121	123	123	127	129	128	129	131	103
100	133	128	129	129	127	127	126	125	121	122	126	127	116	106	108	116	123	128	128	123	123	124	126	129	131	132	131	136	105
100	136	134	132	132	130	129	132	131	128	130	128	93	69	57	55	60	77	117	133	126	125	129	132	132	136	136	135	139	109
100	135	139	138	137	135	135	136	132	135	138	99	87	107	89	63	51	47	64	125	134	130	133	137	139	139	137	137	141	113
99	138	140	141	142	140	140	140	138	143	124	101	160	173	156	108	63	48	46	86	138	137	139	141	142	144	139	139	146	116
93	141	142	140	151	150	149	143	143	145	94	116	167	171	160	130	90	55	46	56	116	127	134	138	143	148	146	142	149	119
81	115	116	120	133	129	137	147	153	138	71	122	153	158	137	130	105	58	45	47	86	105	117	122	142	148	149	136	136	106
70	91	103	109	106	99	104	129	144	118	62	92	105	119	98	87	91	60	44	44	72	89	98	100	134	150	147	131	128	101
73	92	101	103	96	89	89	98	99	90	61	131	145	132	141	148	121	58	43	44	69	88	92	90	120	142	137	127	124	104
78	87	88	91	92	89	82	80	77	65	55	124	153	127	142	147	95	51	44	42	64	86	93	96	109	119	112	108	108	93
74	75	82	88	86	87	84	82	74	60	49	90	133	93	113	120	79	49	44	41	54	73	76	78	84	89	93	85	86	86
70	71	82	86	85	81	79	78	77	67	51	63	124	118	109	105	70	47	43	42	56	78	81	81	84	87	85	83	85	82
69	67	73	81	81	76	77	81	87	77	53	44	74	120	92	68	55	47	44	42	55	80	87	83	83	82	83	87	84	78
69	65	69	76	84	84	88	95	101	87	53	43	39	58	63	53	60	58	45	44	51	74	82	80	77	75	79	84	87	78
72	70	77	83	90	90	91	96	96	84	54	44	41	70	104	87	94	82	51	49	49	65	80	85	88	96	96	88	87	79
73	79	89	91	89	79	75	66	66	58	46	45	68	115	132	128	132	114	64	60	50	49	64	77	84	104	92	88	79	76
74	78	78	81	84	84	73	62	60	51	47	76	139	159	155	154	153	124	70	63	51	51	60	90	113	105	94	91	83	74
75	78	79	84	98	93	76	72	60	47	61	118	164	167	161	164	155	121	69	58	56	68	57	65	93	108	108	109	95	75
71	78	78	76	84	96	88	70	49	50	106	157	168	168	168	167	158	118	75	83	96	89	64	48	56	76	87	105	111	85
67	78	82	83	82	91	91	57	49	57	118	166	164	174	180	178	163	117	86	79	65	53	47	43	45	55	63	80	105	87
65	72	89	93	92	89	60	48	53	67	78	93	113	140	152	151	117	77	55	51	51	49	43	43	44	49	68	70	72	74
61	59	79	89	92	84	52	48	55	66	62	68	78	78	77	80	65	48	45	56	60	56	52	48	44	45	59	66	67	69
61	56	73	83	83	68	49	46	53	56	54	60	69	65	64	59	45	42	50	54	62	74	67	54	47	48	57	71	61	69
65	72	100	107	103	66	49	45	46	49	49	51	52	57	59	47	41	44	47	51	65	71	56	45	44	52	53	68	62	67
75	78	90	94	82	54	51	48	45	48	48	48	49	55	51	43	41	43	43	55	60	52	43	43	44	52	52	67	71	70
72	69	73	68	56	54	61	55	46	44	45	45	49	52	47	40	41	40	42	51	45	42	45	44	43	48	47	51	56	65
70	68	63	56	57	60	64	59	48	42	42	43	49	48	40	40	41	40	44	43	40	46	46	45	44	47	47	51	60	69
73	63	56	53	54	57	58	52	47	41	42	43	48	43	40	39	40	41	42	40	42	47	50	47	45	47	46	51	62	73
71	56	52	55	69	74	73	62	50	49	47	42	40	39	38	38	40	38	39	39	43	47	46	45	44	45	45	47	56	71
67	51	49	56	77	81	78	79	106	135	114	79	52	36	35	39	38	36	39	40	43	47	47	44	43	45	45	46	50	65
67	50	49	53	67	70	66	73	124	142	141	135	124	86	52	39	37	36	37	38	41	46	47	44	46	46	46	46	51	64
68	50	48	46	54	57	60	61	84	103	123	134	134	131	122	72	49	54	52	62	58	53	46	45	49	49	45	46	51	61
65	48	45	43	45	46	48	56	81	103	105	116	119	114	98	97	86	83	76	76	74	65	52	42	45	45	43	46	54	62
55	44	46	46	47	44	43	82	112	129	98	96	109	106	99	75	76	69	69	71	63	60	55	42	40	43	44	48	60	67
55	44	48	50	50	49	50	84	105	125	99	75	80	93	81	83	64	56	55	58	57	55	49	41	41	44	47	51	62	68
54	45	49	49	47	49	52	67	97	94	70	60	59	62	61	58	52	46	45	47	48	46	43	42	44	48	50	52	58	65
56	49	51	51	51	52	51	51	71	60	56	56	59	60	52	50	49	46	44	43	44	43	42	41	43	50	52	52	55	63
55	47	49	50	51	51	49	50	53	50	48	52	52	48	44	43	44	45	44	41	43	42	42	41	41	51	52	50	53	57
53	49	50	51	49	48	50	50	46	45	45	46	42	40	39	37	39	42	41	39	41	40	42	40	40	48	50	47	49	56
50	46	47	46	45	50	50	46	42	40	43	45	38	37	37	36	38	41	41	38	37	38	42	38	39	44	46	44	46	61
55	46	46	47	49	51	54	52	50	47	50	52	51	48	49	49	47	47	49	50	43	45	46	43	42	43	47	49	52	64

Learning from images

What the computer sees

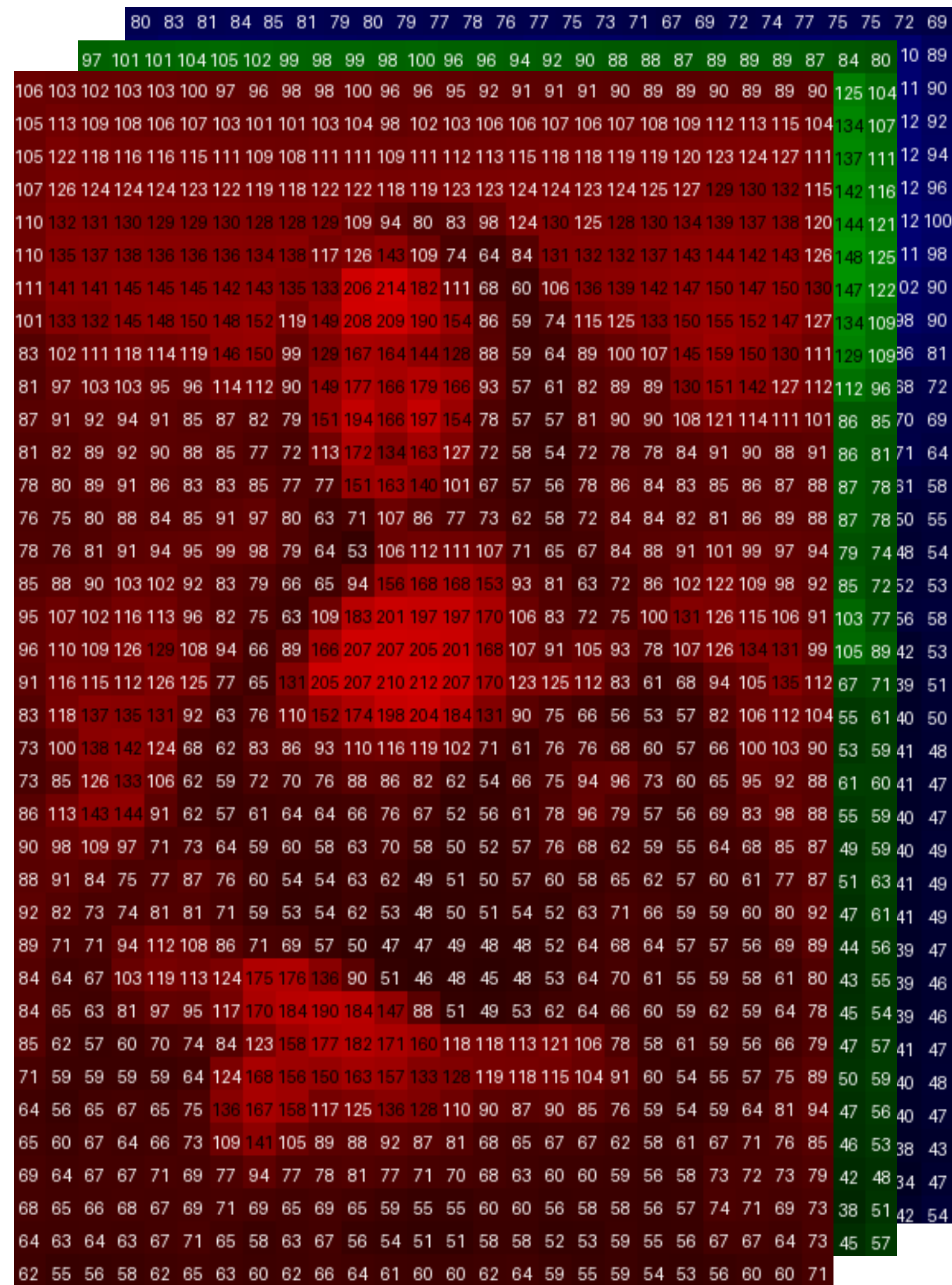
106	103	102	103	103	100	97	96	98	98	100	96	96	95	92	91	91	91	90	89	89	90	89	89	90
105	113	109	108	106	107	103	101	101	103	104	98	102	103	106	106	107	108	107	108	109	112	113	115	104
105	122	118	116	116	115	111	109	108	111	111	109	111	112	113	115	118	118	119	119	120	123	124	127	111
107	126	124	124	124	123	122	119	118	122	122	118	119	123	123	124	124	123	124	125	127	129	130	132	115
110	132	131	130	129	129	130	128	128	129	109	94	80	83	98	124	130	125	128	130	134	139	137	138	120
110	135	137	138	136	136	136	134	138	117	126	143	109	74	64	84	131	132	132	137	143	144	142	143	126
111	141	141	145	145	145	142	143	135	133	206	214	182	111	68	60	106	136	139	142	147	150	147	150	130
101	133	132	145	148	150	148	152	119	149	208	209	190	154	86	59	74	115	125	133	150	155	152	147	127
83	102	111	118	114	119	146	150	99	129	167	164	144	128	88	59	64	89	100	107	145	159	150	130	111
81	97	103	103	95	96	114	112	90	149	177	166	179	166	93	57	61	82	89	89	130	151	142	127	112
87	91	92	94	91	85	87	82	79	151	194	166	197	154	78	57	57	81	90	90	108	121	114	111	101
81	82	89	92	90	88	85	77	72	113	172	134	163	127	72	58	54	72	78	78	84	91	90	88	91
78	80	89	91	86	83	83	85	77	77	151	163	140	101	67	57	56	78	86	84	83	85	86	87	88
76	75	80	88	84	85	91	97	80	63	71	107	86	77	73	62	58	72	84	84	82	81	86	89	88
78	76	81	91	94	95	99	98	79	64	53	106	112	111	107	71	65	67	84	88	91	101	99	97	94
85	88	90	103	102	92	83	79	66	65	94	156	168	168	153	93	81	63	72	86	102	122	109	98	92
95	107	102	116	113	96	82	75	63	109	183	201	197	197	170	106	83	72	75	100	131	126	115	106	91
96	110	109	126	129	108	94	66	89	166	207	207	205	201	168	107	91	105	93	78	107	126	134	131	99
91	116	115	112	126	125	77	65	131	205	207	210	212	207	170	123	125	112	83	61	68	94	105	135	112
83	118	137	135	131	92	63	76	110	152	174	198	204	184	131	90	75	66	56	53	57	82	106	112	104
73	100	138	142	124	68	62	83	86	93	110	116	119	102	71	61	76	76	68	60	57	66	100	103	90
73	85	126	133	106	62	59	72	70	76	88	86	82	62	54	66	75	94	96	73	60	65	95	92	88
86	113	143	144	91	62	57	61	64	64	66	76	67	52	56	61	78	96	79	57	56	69	83	98	88
90	98	109	97	71	73	64	59	60	58	63	70	58	50	52	57	76	68	62	59	55	64	68	85	87
88	91	84	75	77	87	76	60	54	54	63	62	49	51	50	57	60	58	65	62	57	60	61	77	87
92	82	73	74	81	81	71	59	53	54	62	53	48	50	51	54	52	63	71	66	59	59	60	80	92
89	71	71	94	112	108	86	71	69	57	50	47	47	49	48	48	52	64	68	64	57	57	56	69	89
84	64	67	103	119	113	124	175	176	136	90	51	46	48	45	48	53	64	70	61	55	59	58	61	80
84	65	63	81	97	95	117	170	184	190	184	147	88	51	49	53	62	64	66	60	59	62	59	64	78
85	62	57	60	70	74	84	123	158	177	182	171	160	118	118	113	121	108	78	58	61	59	56	66	79
71	59	59	59	59	64	124	168	156	150	163	157	133	128	119	118	115	104	91	60	54	55	57	75	89
64	56	65	67	65	75	136	167	158	117	125	136	128	110	90	87	90	85	76	59	54	59	64	81	94
65	60	67	64	66	73	109	141	105	89	88	92	87	81	68	65	67	67	62	58	61	67	71	76	85
69	64	67	67	71	69	77	94	77	78	81	77	71	70	68	63	60	60	59	56	58	73	72	73	79
68	65	66	68	67	69	71	69	65	69	65	59	55	55	60	60	56	58	58	56	57	74	71	69	73
64	63	64	63	67	71	65	58	63	67	56	54	51	51	58	58	52	53	59	55	56	67	67	64	73
62	55	56	58	62	65	63	60	62	66	64	61	60	60	62	64	59	55	59	54	53	56	60	60	71



80	83	81	84	85	81	79	80	79	77	78	76	77	75	73	71	67	69	72	74	77	75	75	72	69
90	100	94	95	94	96	93	91	93	95	95	95	100	102	100	98	95	95	99	101	104	107	109	110	89
86	103	100	97	98	95	94	91	93	94	92	98	103	102	99	103	101	100	101	103	110	112	111	111	90
85	106	100	102	103	101	97	94	94	96	99	99	95	98	100	107	104	98	100	103	110	111	107	112	92
88	111	107	100	99	100	102	99	98	99	66	41	41	43	57	94	104	98	104	108	105	106	105	112	94
85	111	111	106	102	103	104	101	108	72	47	50	38	33	28	48	102	107	108	111	107	104	103	112	96
82	110	110	113	108	106	106	107	98	52	91	93	60	33	30	31	76	110	109	112	111	112	103	112	100
74	105	98	110	108	111	113	111	63	52	87	86	65	45	31	31	47	85	97	103	107	109	102	111	98
59	76	78	79	73	84	110	107	45	42	52	50	39	38	29	30	37	67	84	87	97	103	96	102	90
60	74	82	70	62	63	73	73	35	51	59	50	61	53	29	29	34	62	76	76	92	99	89	98	90
63	64	71	65	62	57	51	46	27	53	73	57	74	48	27	29	34	59	75	81	88	86	81	86	81
58	55	64	59	60	58	53	45	27	35	53	37	51	37	25	28	31	48	63	66	70	74	71	68	72
54	48	57	56	53	56	57	52	31	29	52	52	42	31	25	27	30	50	65	68	73	71	71	70	69
54	43	49	59	57	61	70	71	35	25	33	38	29	28	25	25	28	48	65	65	63	62	70	71	64
58	51	56	64	68	75	81	74	41	26	30	40	34	31	28	21	27	39	57	62	64	59	58	61	58
55	64	74	52	33	36	39	38	31	28	36	56	55	56	42	24	25	28	35	46	50	47	47	50	55
47	37	36	24	22	23	27	28	30	34	71	82	75	77	48	21	22	25	29	42	54	43	44	48	54
46	29	27	26	22	18	23	29	28	51	87	87	84	80	45	19	23	28	29	29	38	49	56	52	53
46	29	24	24	21	20	32	32	37	78	83	93	96	86	46	26	32	33	32	32	31	33	42	56	58
50	29	23	22	24	33	36	32	32	47	62	86	93	73	39	31	31	32	34	35	36	36	35	42	53
50	33	22	19	29	38	37	33	31	29	31	33	38	34	31	33	33	32	33	35	36	37	35	39	51
49	39	32	28	32	37	36	34	33	33	33	32	34	34	33	32	31	32	32	35	38	39	37	40	50
48	40	38	36	37	38	35	34	33	34	33	33	34	34	34	32	30	30	31	34	37	41	42	41	48
53	41	40	38	38	36	34	34	35	35	35	34	33	34	33	32	30	30	29	32	37	40	42	41	47
50	41	41	39	38	36	31	31	34	34	34	32	33	34	33	32	31	30	30	32	36	38	40	40	47
46	38	38	34	32	31	32	32	34	34	33	33	33	33	34	33	31	30	29	32	36	38	40	40	49
44	38	37	27	22	22	26	33	34	32	32	32	32	33	32	32	31	30	32	35	37	39	41	49	
42	37	38	25	20	21	24	51	46	32	29	28	31	32	31	32	31	32	30	33	36	37	38	41	49
43	37	38	31	26	25	26	37	38	39	36	33	31	30	30	29	28	31	34	35	39	40	38	39	47
42	36	36	35	32	32	30	29	25	28	31	33	32	21	18	16	15	19	28	34	38	38	38	39	46
42	37	37	36	34	32	29	37	30	24	28	30	24	24	18	15	16	18	23	30	33	36	38	39	46
44	37	38	36	36	32	26	36	34	28	29	29	28	26	23	23	23	25	26	29	33	36	39	41	47
43	38	39	36	36	35	29	30	31	31	31	32	31	31	28	29	30	30	28	30	33	36	37	40	48
43	38	38	38	38	35	31	28	29	31	34	33	32	33	32	31	32	33	31	31	34	35	37	40	47
41	38	38	38	35	33	31	31	31	30	31	32	31	32	32	31	31	31	30	31	32	32	34	38	43
39	38	37	35	33	32	32	31	29	28	27	28	29	29	28	27	28	28	28	28	28	29	30	34	47
45	40	39	40	40	42	40	38	37	38	38	37	38	37	34	36	36	34	34	34	34	34	38	42	54

Learning from images

What the computer sees



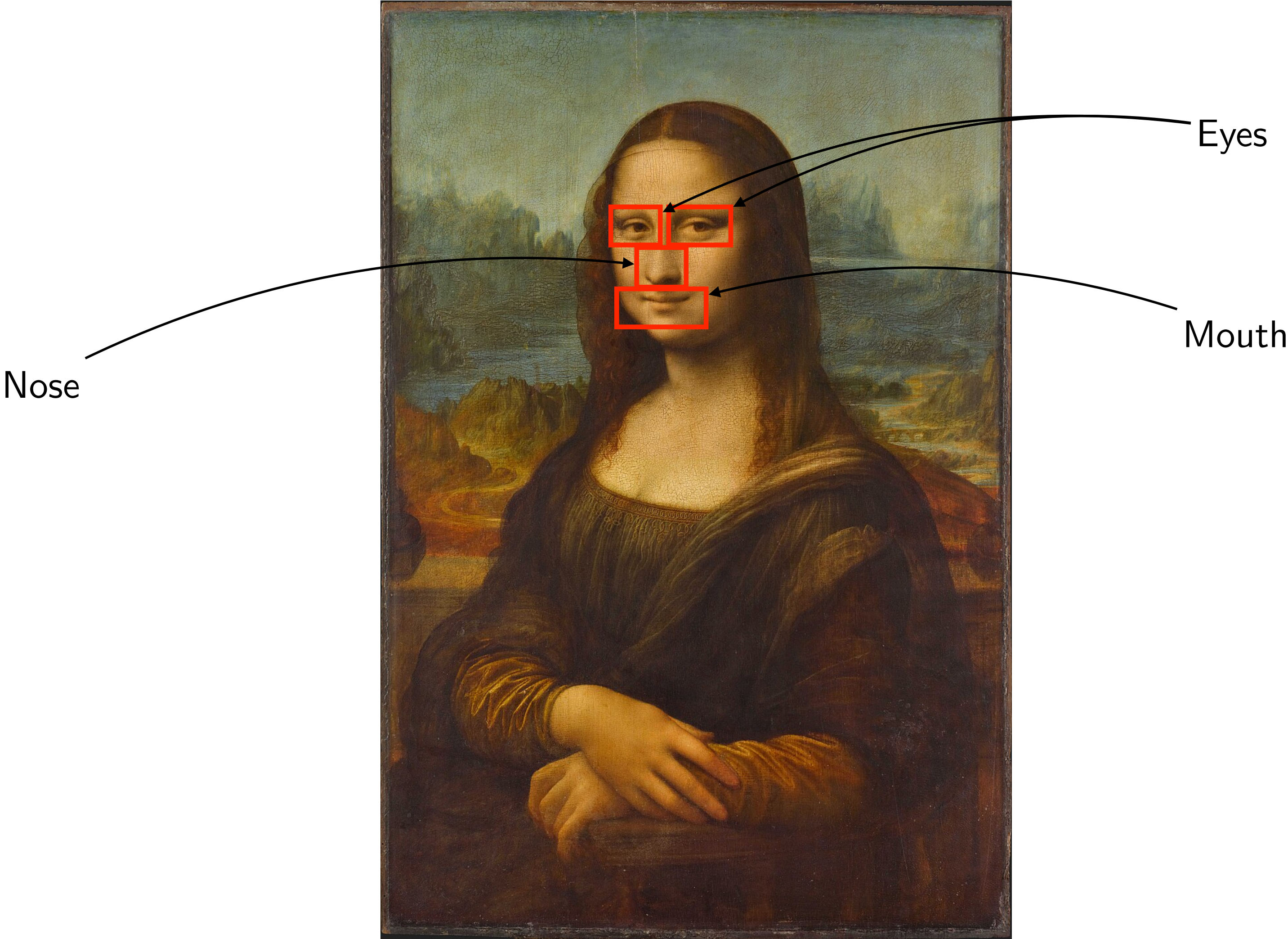
Learning from images

Challenges with using MLPs

- If I want to produce an image from an image with resolution 512 using a MLP I need $(512 \times 512 \times 3)^2 \approx 6.18 \cdot 10^{11}$ parameters
 - With 4 bytes per parameter, this requires about *2.5 terabytes* in storage
 - This clearly does *not* scale
-
- In addition, images have a lot of *structure* that we are not using
 - Images often vary *continuously*
 - Classification of images should *not* depend on a single pixel *but* in groups of pixels

Learning from images

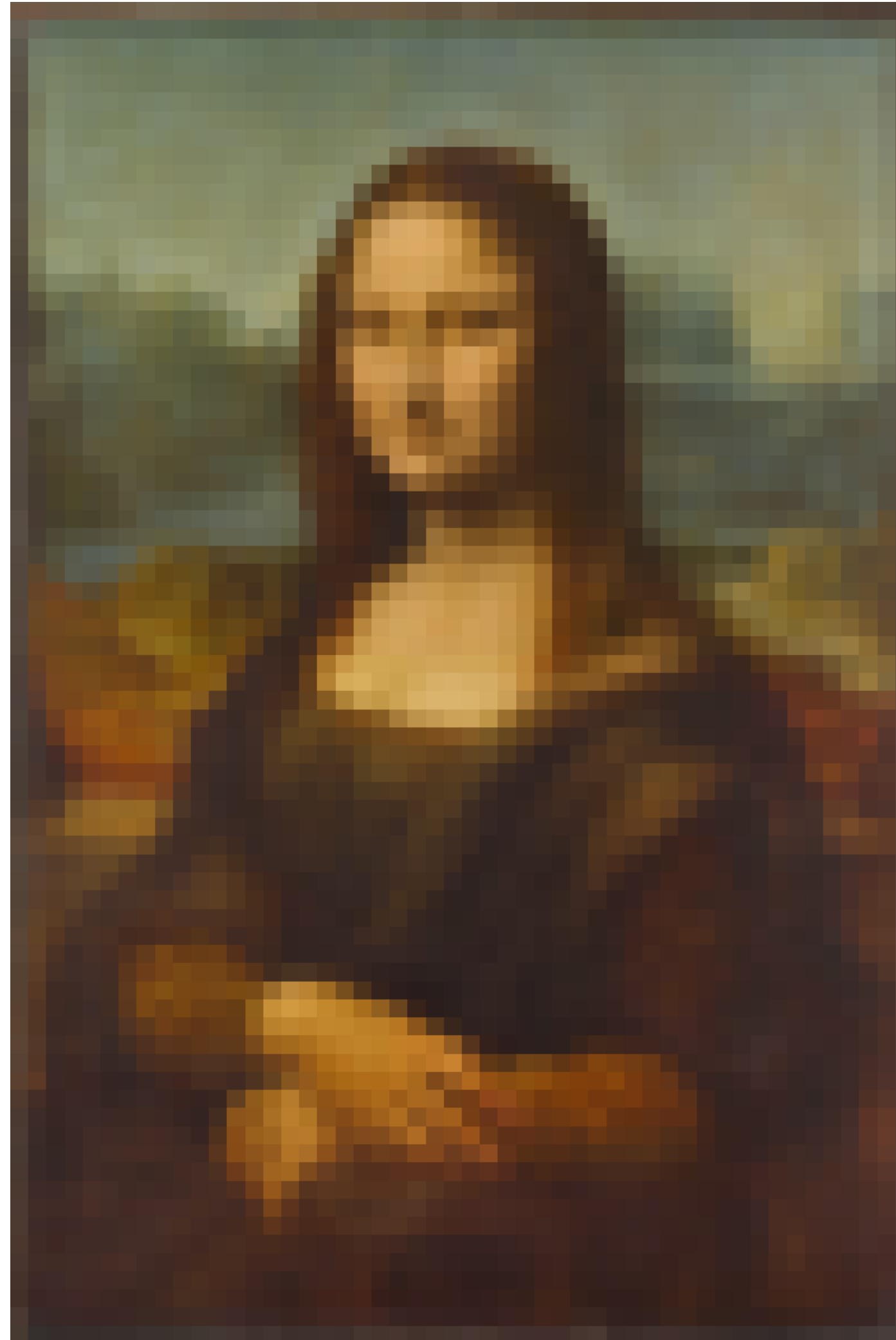
Recognition is based in detecting lower level features



This must be a face!

Learning from images

We can still recognize with lower resolution



This looks like the Mona Lisa!

Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0

Input image I

1	0	1
0	1	0
1	0	1

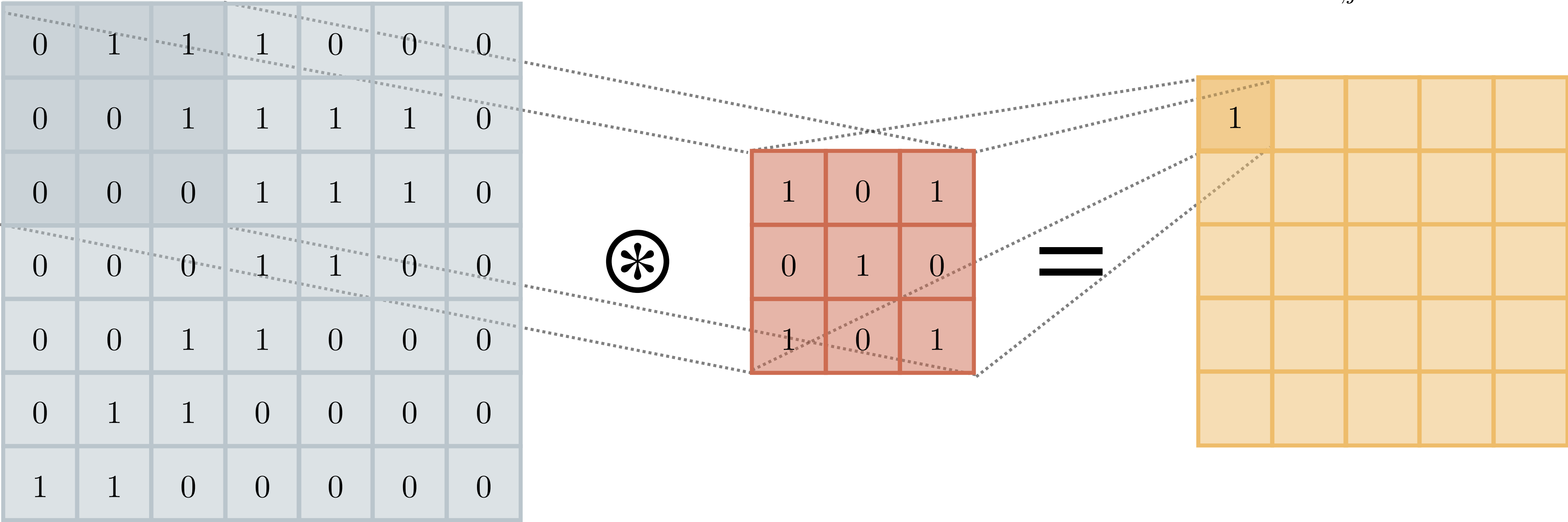
Filter or kernel K

Convolutions

Need to also choose a *pooling* operation

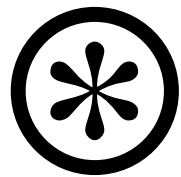
Here we sum

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) I(x + i, y + j)$$



Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



1	0	1
0	1	0
1	0	1

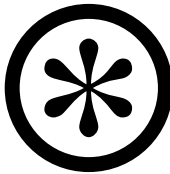
=

1	4			

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) \, I(x + i, y + j)$$

Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



1	0	1
0	1	0
1	0	1

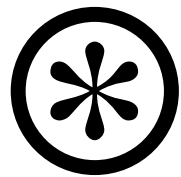
=

1	4	3		

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) I(x + i, y + j)$$

Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



1	0	1
0	1	0
1	0	1

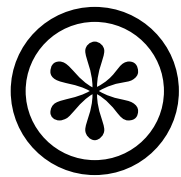
=

1	4	3	4	

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) I(x + i, y + j)$$

Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



1	0	1
0	1	0
1	0	1

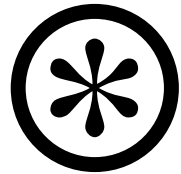
=

1	4	3	4	1

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) I(x + i, y + j)$$

Convolutions

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



1	0	1
0	1	0
1	0	1

=

1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
1	3	3	1	1
3	3	1	1	0

$$(I \circledast K)(x, y) = \sum_{i, j} K(i, j) I(x + i, y + j)$$

Filter in computer vision

Gaussian blur



Gaussian blur

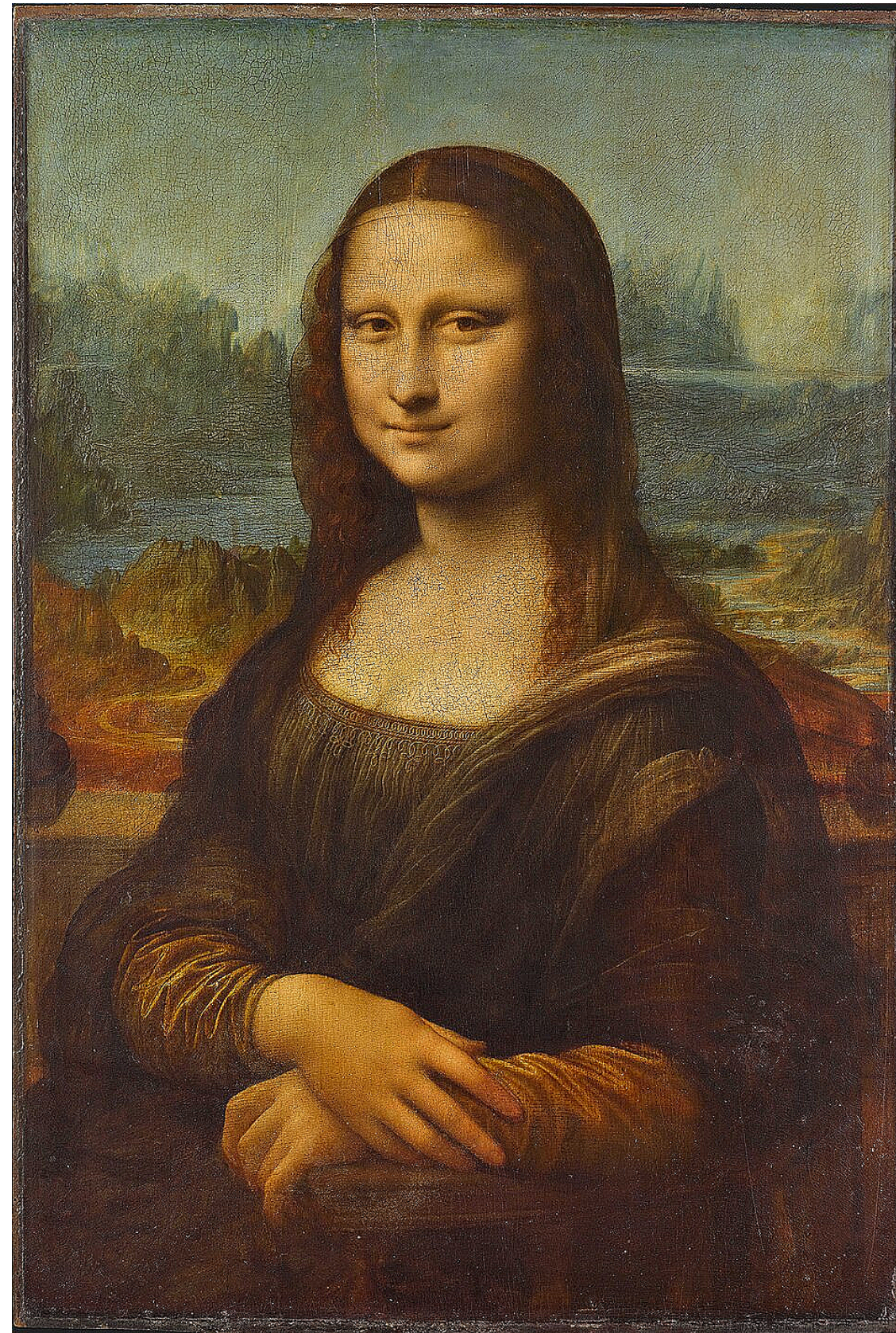
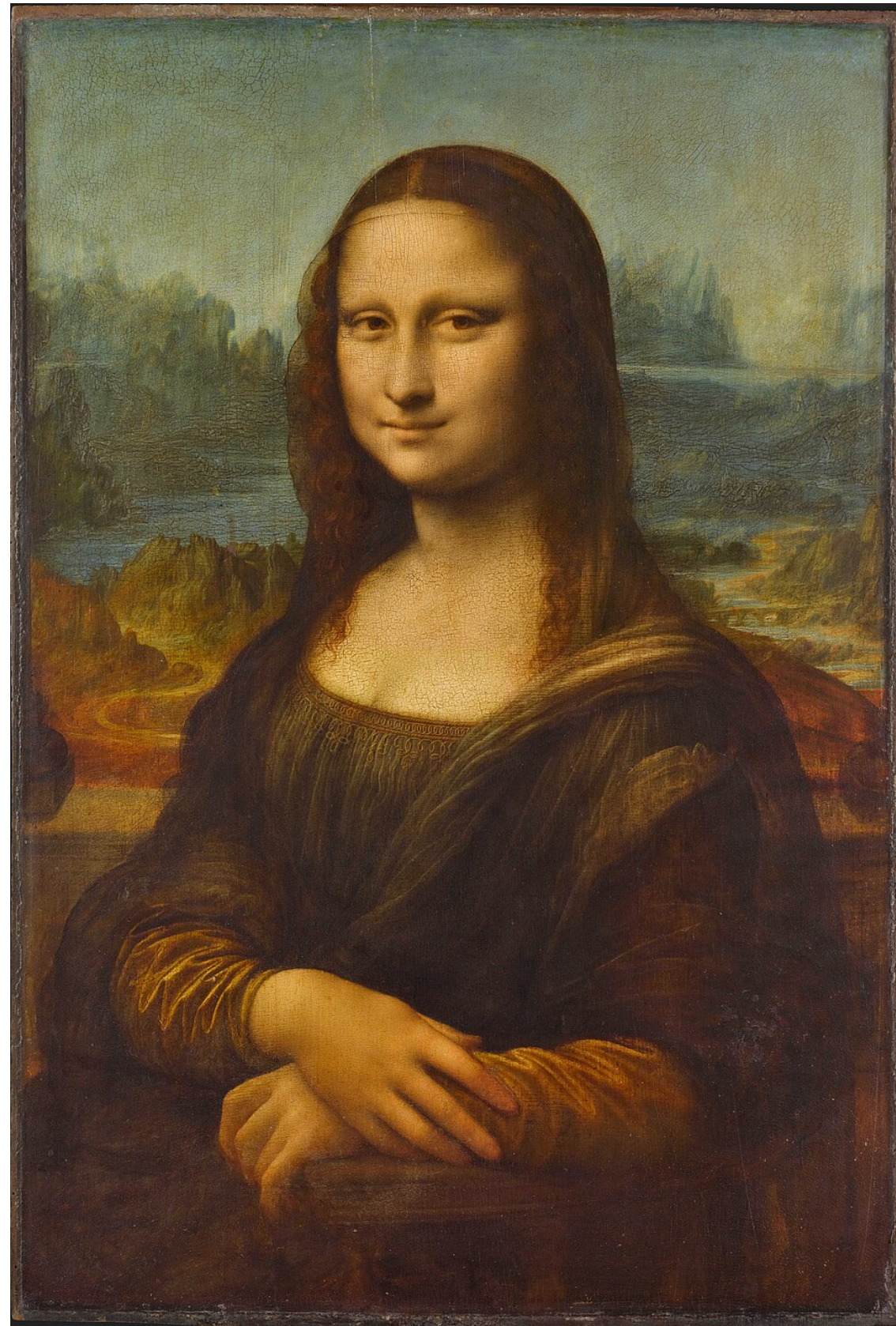
$$K(i, j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right)$$

$$\sigma \approx 1.7$$

$$K = \begin{bmatrix} 0.000 & 0.000 & 0.001 & 0.002 & 0.003 & 0.004 & 0.003 & 0.002 & 0.001 & 0.000 & 0.000 \\ 0.000 & 0.001 & 0.003 & 0.006 & 0.010 & 0.012 & 0.010 & 0.006 & 0.003 & 0.001 & 0.000 \\ 0.001 & 0.003 & 0.008 & 0.015 & 0.024 & 0.029 & 0.024 & 0.015 & 0.008 & 0.003 & 0.001 \\ 0.002 & 0.006 & 0.015 & 0.028 & 0.045 & 0.054 & 0.045 & 0.028 & 0.015 & 0.006 & 0.002 \\ 0.003 & 0.010 & 0.024 & 0.045 & 0.072 & 0.087 & 0.072 & 0.045 & 0.024 & 0.010 & 0.003 \\ 0.004 & 0.012 & 0.029 & 0.054 & 0.087 & 0.106 & 0.087 & 0.054 & 0.029 & 0.012 & 0.004 \\ 0.003 & 0.010 & 0.024 & 0.045 & 0.072 & 0.087 & 0.072 & 0.045 & 0.024 & 0.010 & 0.003 \\ 0.002 & 0.006 & 0.015 & 0.028 & 0.045 & 0.054 & 0.045 & 0.028 & 0.015 & 0.006 & 0.002 \\ 0.001 & 0.003 & 0.008 & 0.015 & 0.024 & 0.029 & 0.024 & 0.015 & 0.008 & 0.003 & 0.001 \\ 0.000 & 0.001 & 0.003 & 0.006 & 0.010 & 0.012 & 0.010 & 0.006 & 0.003 & 0.001 & 0.000 \\ 0.000 & 0.000 & 0.001 & 0.002 & 0.003 & 0.004 & 0.003 & 0.002 & 0.001 & 0.000 & 0.000 \end{bmatrix}$$

Filter in computer vision

Sharpening filter



Sharpening kernel

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Filter in computer vision

Sobel filter



Sobel filter (horizontal)

$$K = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

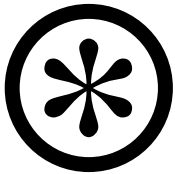
Convolutional neural networks

Learnable kernels

Receptive field

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0

Input layer



w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Learnable kernel

=

Activation map

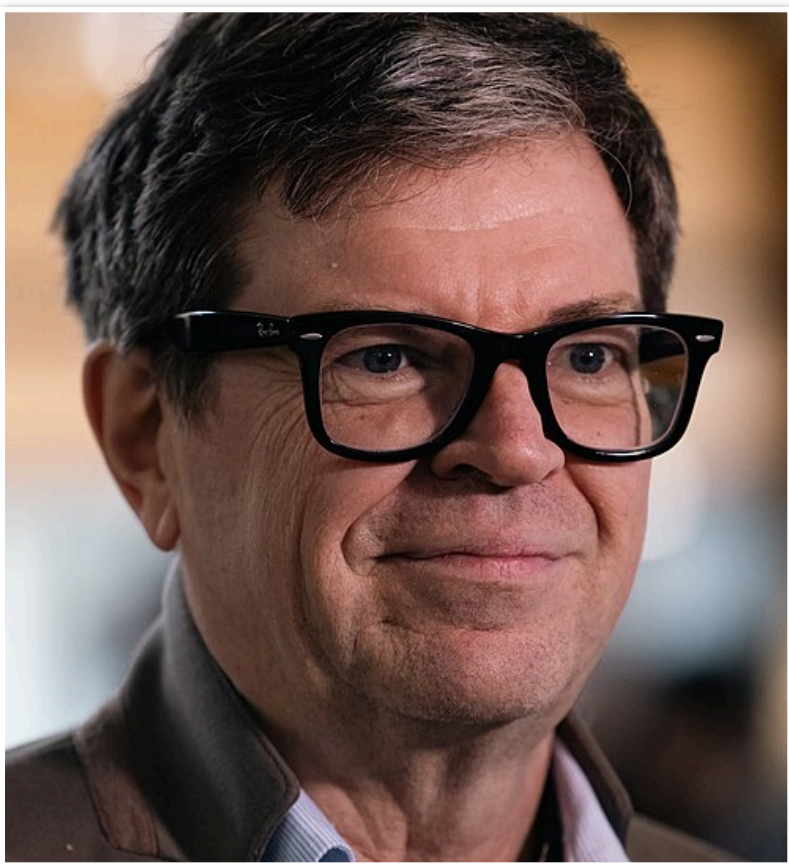
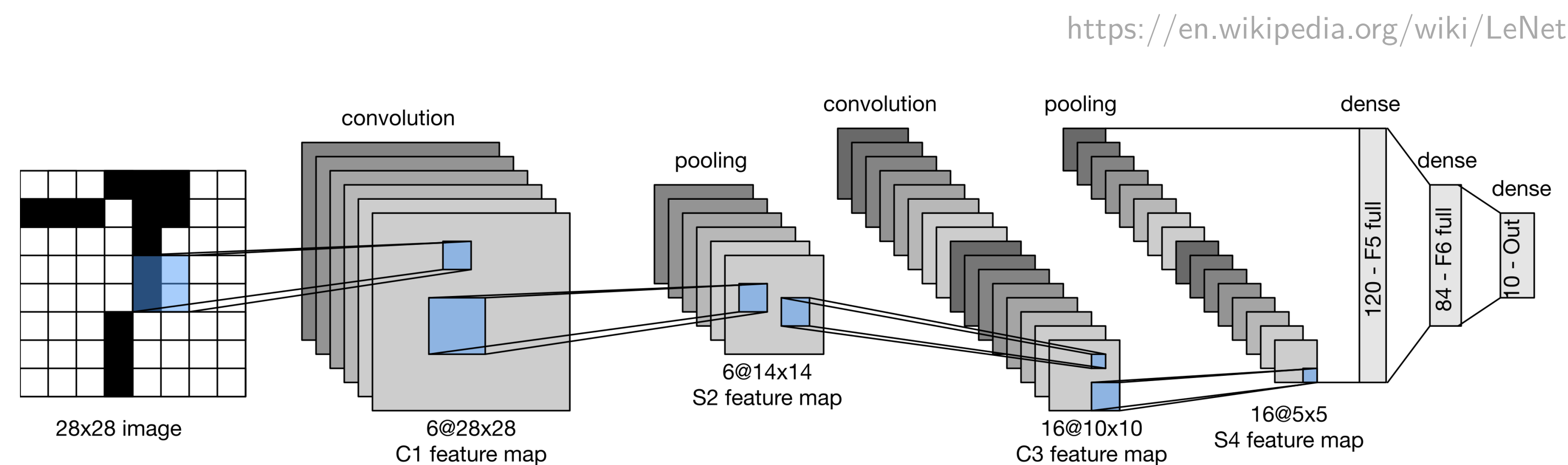
$h_{11}^{(1)}$	...			
⋮	⋱			

Hidden layer

$$h_{11}^{(1)} = g(R_{11} \circledast W_{11} + b_{11})$$

Convolutional neural networks

LeNet



LeNet is a series of convolutional neural networks created by a research group in AT&T Bell Laboratories from 1988 to 1998, with Yann LeCun as their group leader

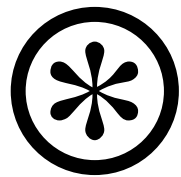
They were created to read grayscale images of handwritten digits and letters, used in ATM for reading cheques

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE 86.11, 1998, pp. 2278–2324

Convolution is translation equivariant

I

0	1	1	1	0	0	0
0	0	1	1	1	1	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
1	1	0	0	0	0	0



K

1	0	1
0	1	0
1	0	1

=

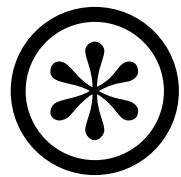
$I \circledast K$

1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
1	3	3	1	1
3	3	1	1	0

Convolution is translation equivariant

$T(I)$

1	1	1	0	0	0	0
0	1	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	1
1	0	0	0	0	0	1



K

1	0	1
0	1	0
1	0	1



$T(I) \circledast K = T(I \circledast K)$

4	3	4	1	1
2	4	3	3	1
2	3	4	1	1
3	3	1	1	1
3	1	1	0	3

Learning from text

Lecture 1 — Introduction to Deep Learning

Learning from text

What the computer sees

Input

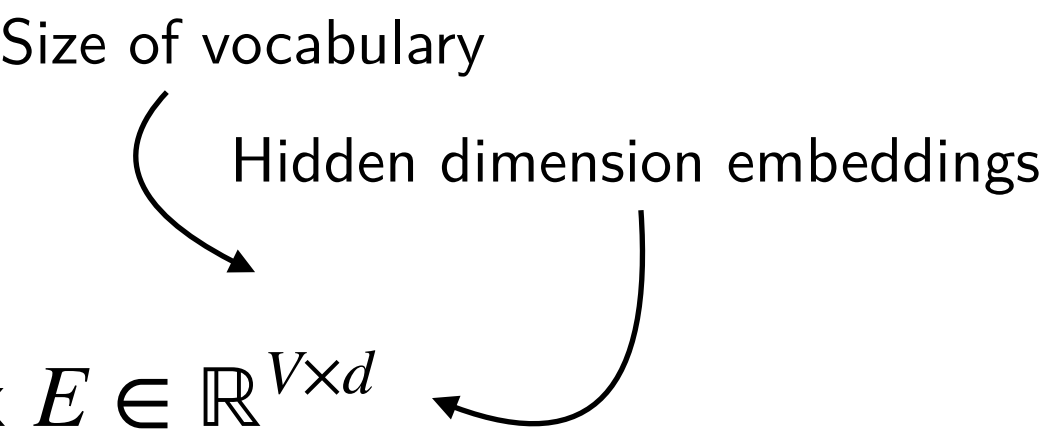
The cat ate the mouse

Tokenization: break down into semantic pieces

[[The], [cat], [ate], [the], [mouse]]

Embedding: Map each token to a vocabulary dictionary and learn an embedding matrix $E \in \mathbb{R}^{V \times d}$

[1996, 4937, 8823, 1996, 8000]
 $1996 \xrightarrow{E} v \in \mathbb{R}^d$



Add positional encoding: the order of the words is important

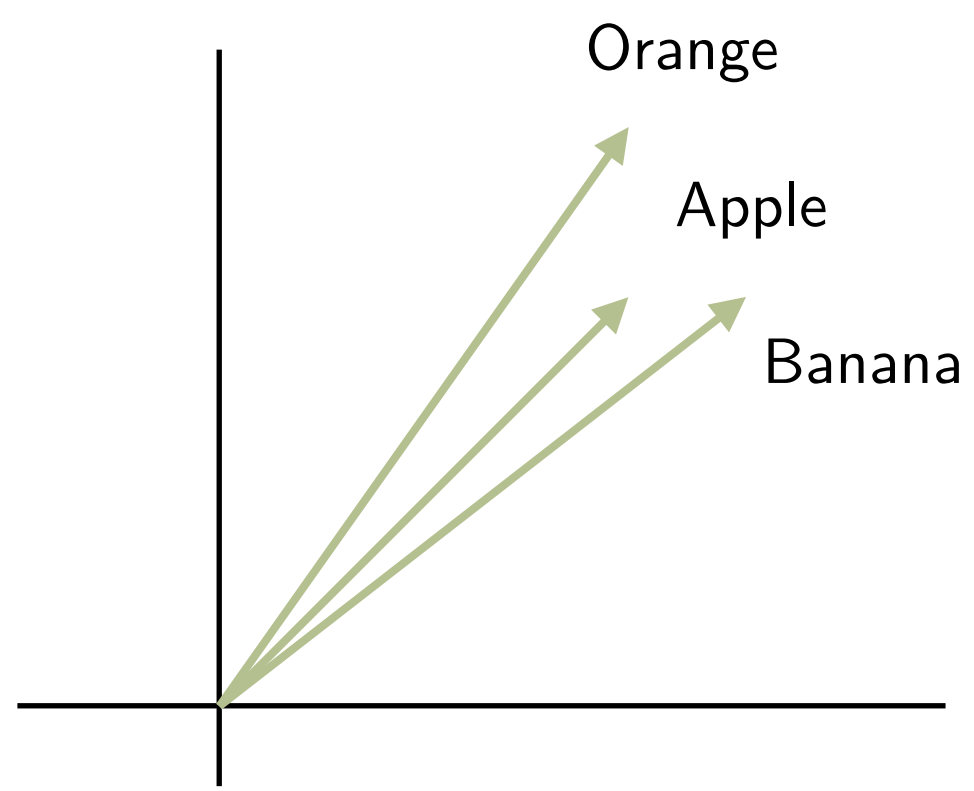
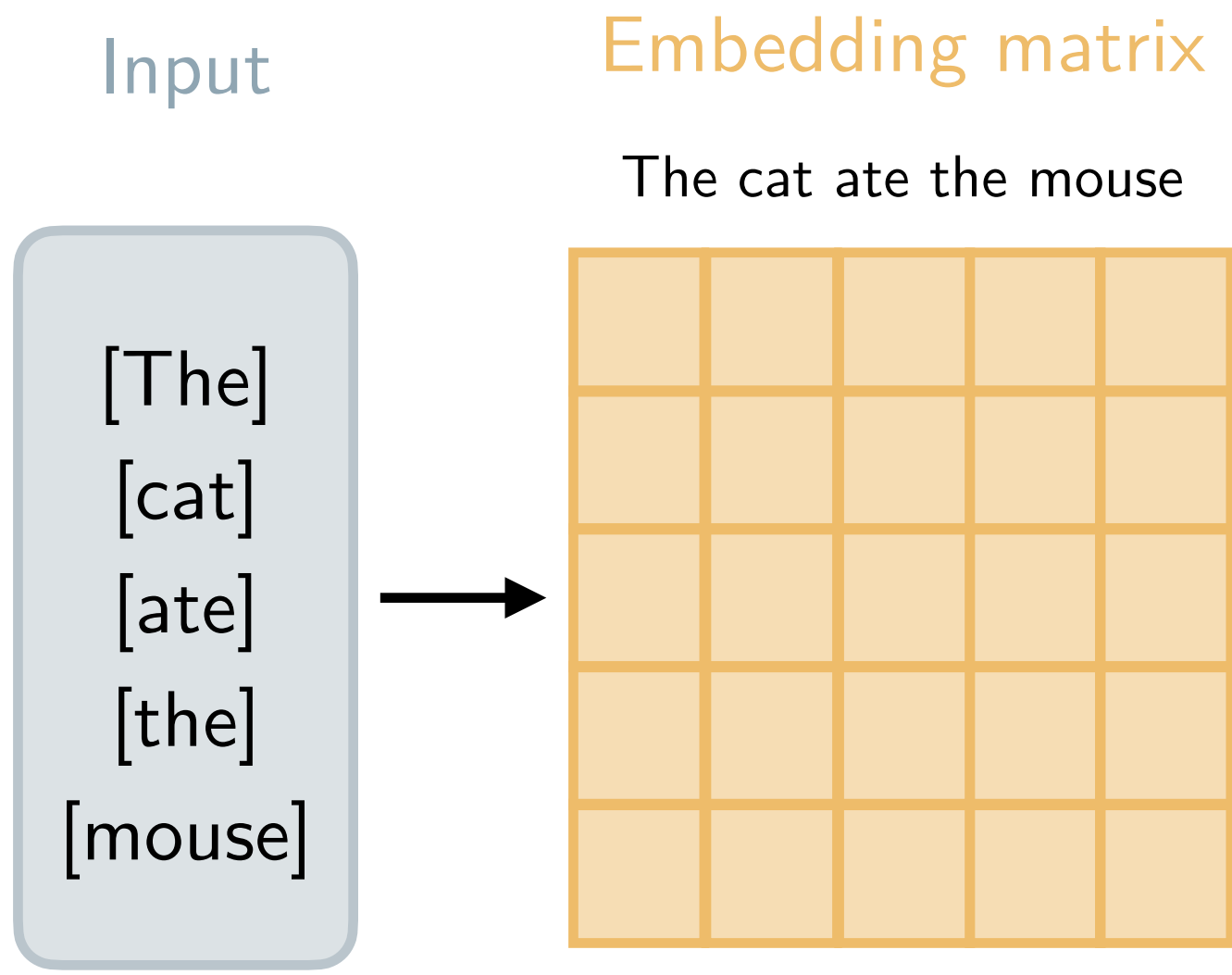
Not the same to say: “The mouse ate the cat” than “The cat ate the mouse”

Learning from text

Word embeddings

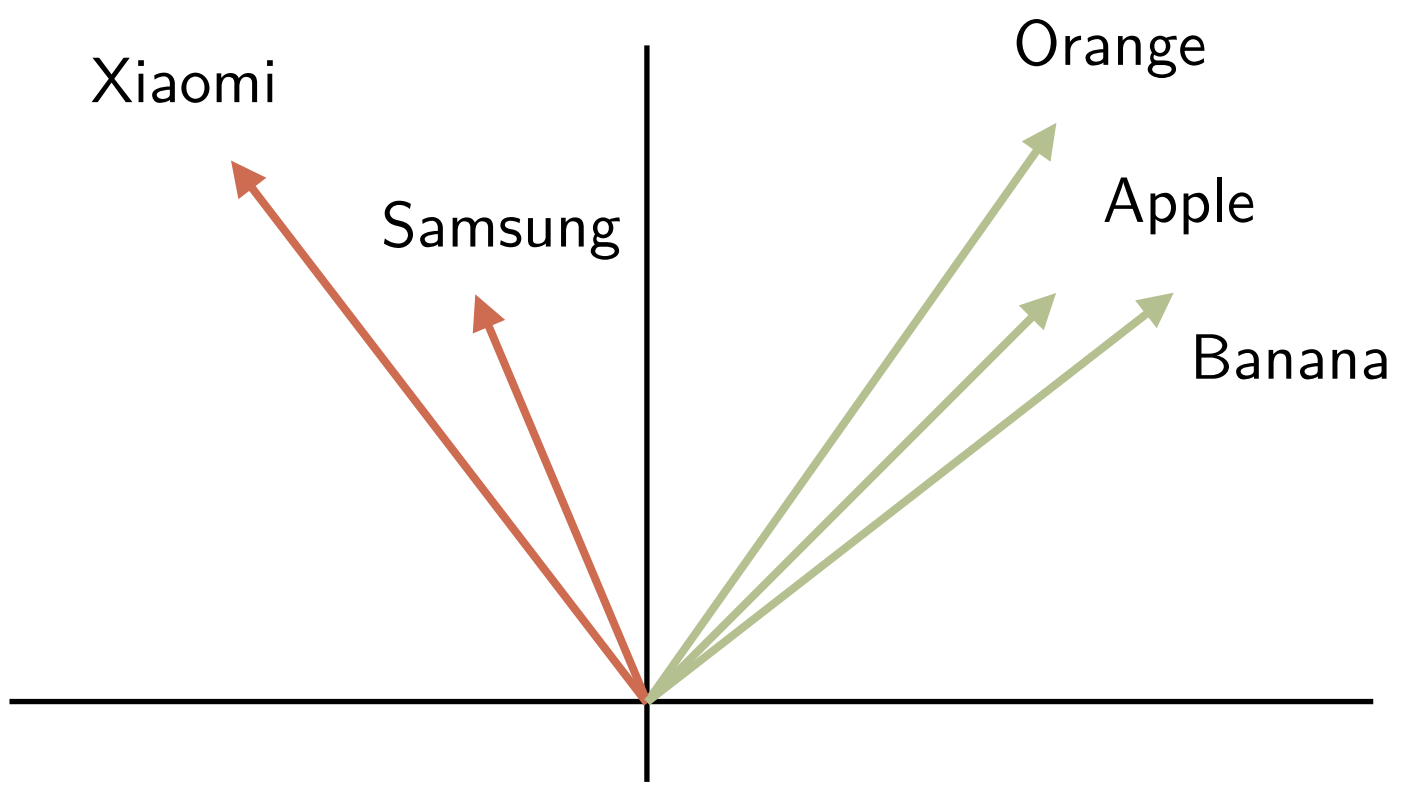
How do these embeddings look like?

They should capture **semantic** meaning



$$X_{\text{king}} - X_{\text{man}} + X_{\text{woman}} \approx X_{\text{queen}}$$

However, they still do not capture **contextual** meaning



Apple has released a new phone

$X \in \mathbb{R}^{T \times d}$

T number of tokens

d dimension embeddings

Learning from text

Transformers and the attention mechanism

Attention Is All You Need

Ashish Vaswani*

Google Brain

avaswani@google.com

Noam Shazeer*

Google Brain

noam@google.com

Niki Parmar*

Google Research

nikip@google.com

Jakob Uszkoreit*

Google Research

usz@google.com

Llion Jones*

Google Research

llion@google.com

Aidan N. Gomez* †

University of Toronto

aidan@cs.toronto.edu

Lukasz Kaiser*

Google Brain

lukaszkaizer@google.com

Illia Polosukhin* ‡

illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

The diagram illustrates the internal structure of the Transformer model, showing the flow from inputs to output probabilities.

Encoder (Left): The input sequence is processed by an **Input Embedding** layer, which is added to **Positional Encoding**. This is followed by a stack of $N \times$ identical layers. Each layer consists of a **Multi-Head Attention** sub-layer (which also receives input from the previous layer's output) and a **Feed Forward** sub-layer, both followed by **Add & Norm** operations.

Decoder (Right): The output sequence is processed by an **Output Embedding** layer, which is added to **Positional Encoding**. This is followed by a stack of $N \times$ identical layers. Each layer consists of a **Masked Multi-Head Attention** sub-layer (which receives input from the previous layer's output), a **Multi-Head Attention** sub-layer (which receives input from the encoder's output), and a **Feed Forward** sub-layer, all followed by **Add & Norm** operations.

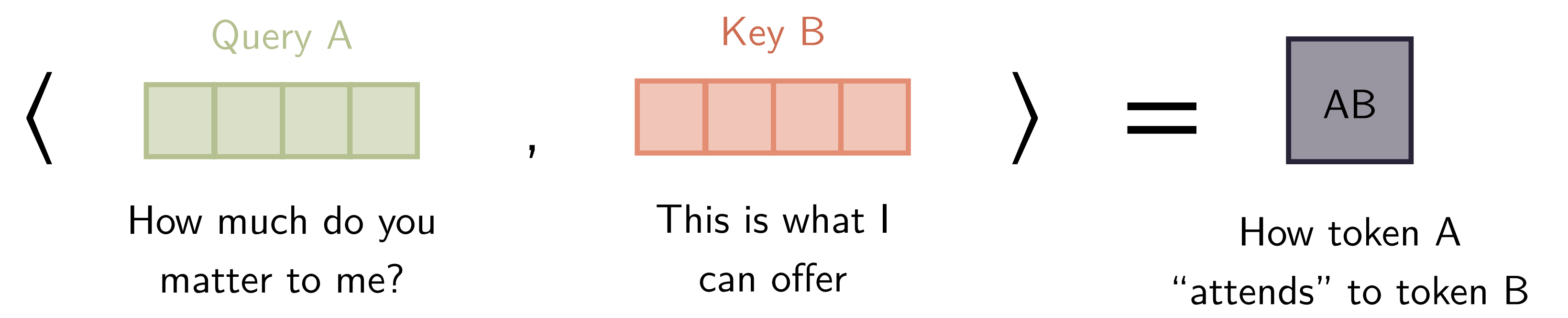
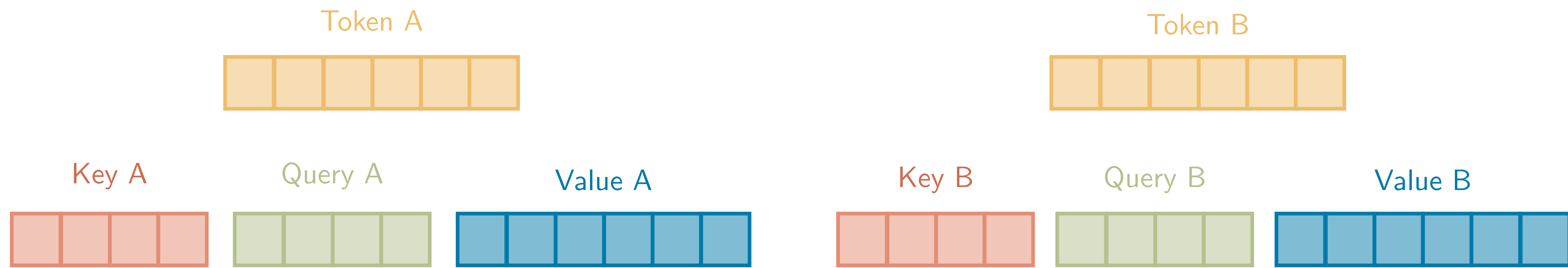
Output: The final output from the decoder stack passes through a **Linear** layer and a **Softmax** layer to produce the **Output Probabilities**.

A. Vaswani, S. Noam, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin
Attention is All You Need, NeurIPS 2017

59

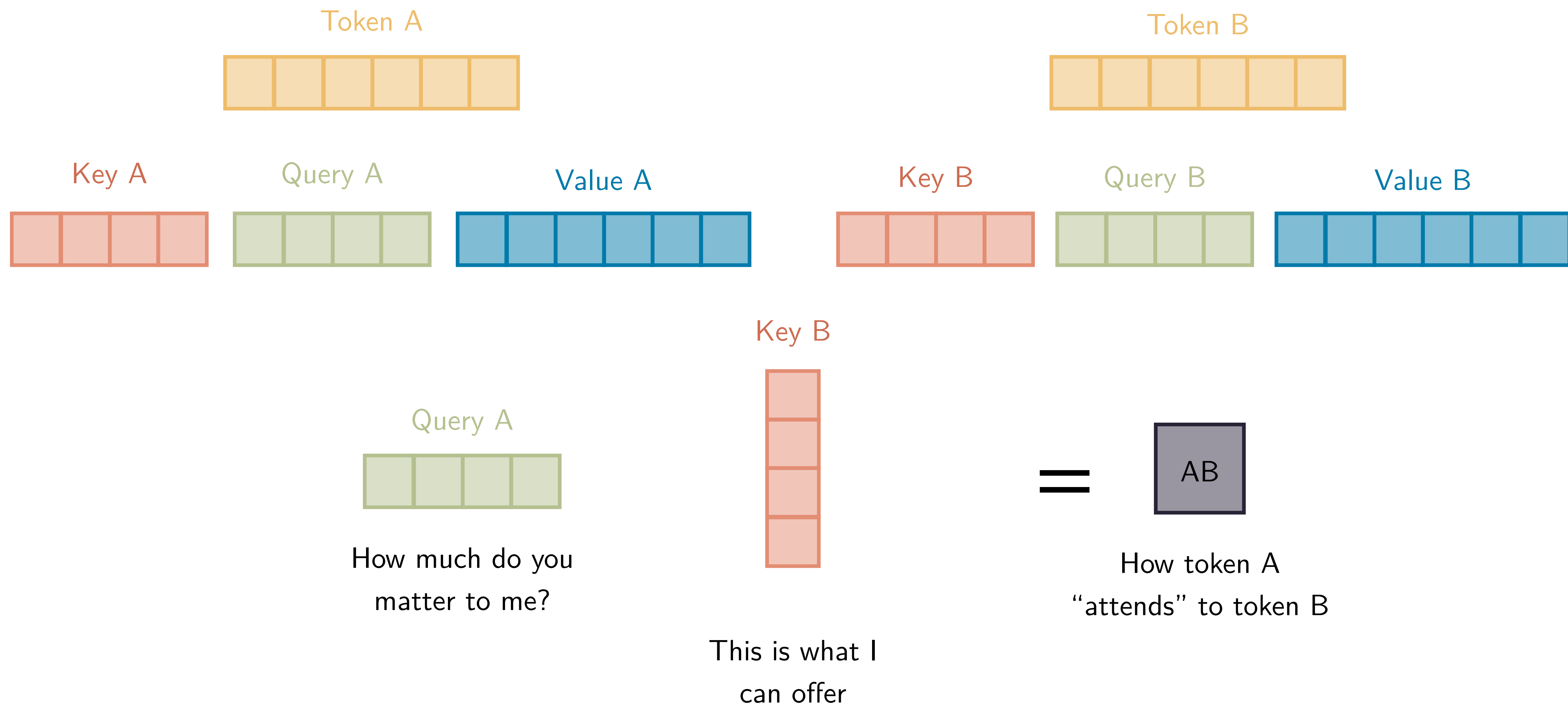
Learning from text

The attention mechanism



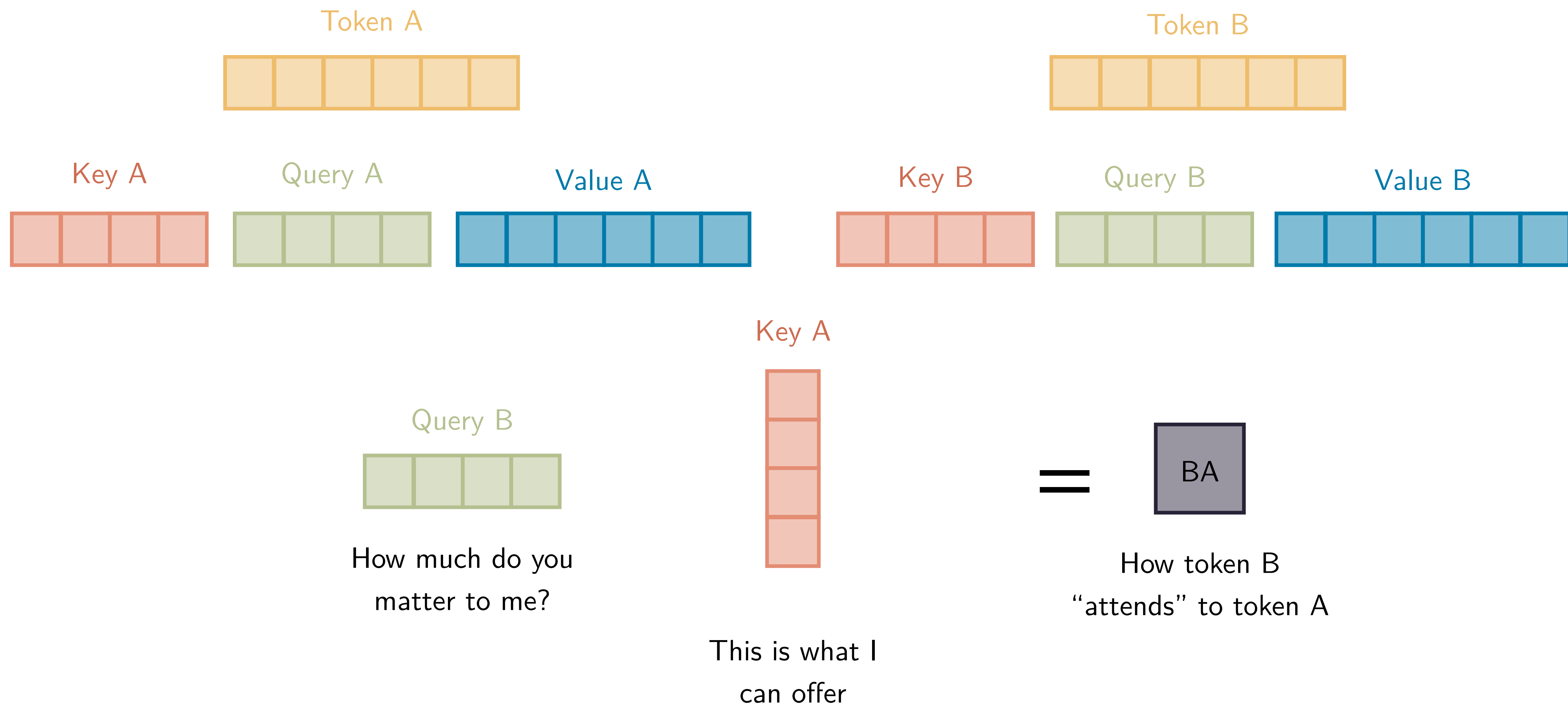
Learning from text

The attention mechanism



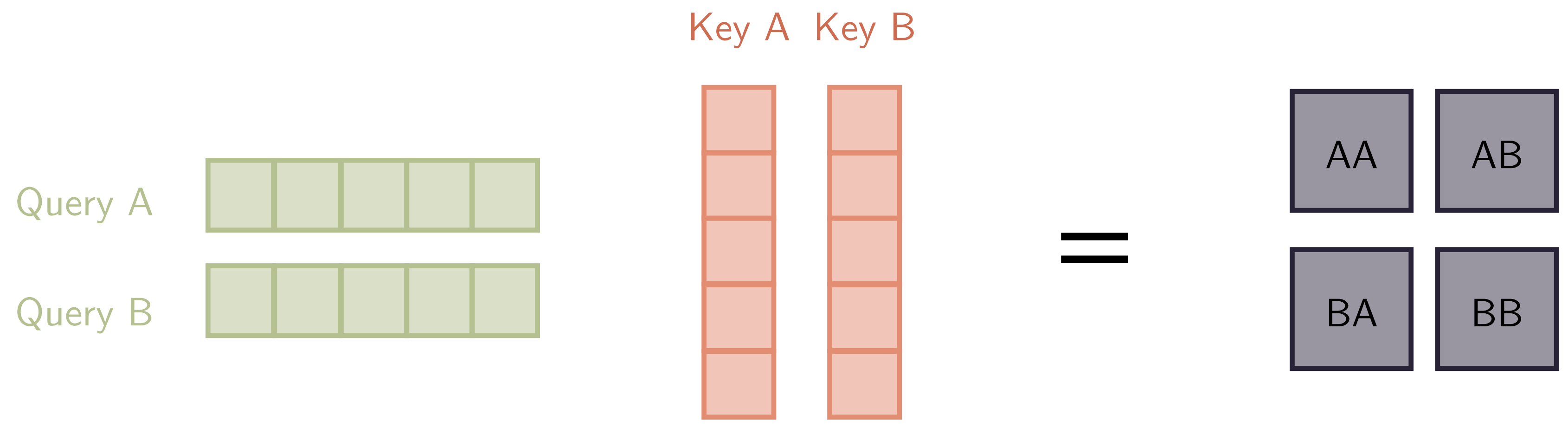
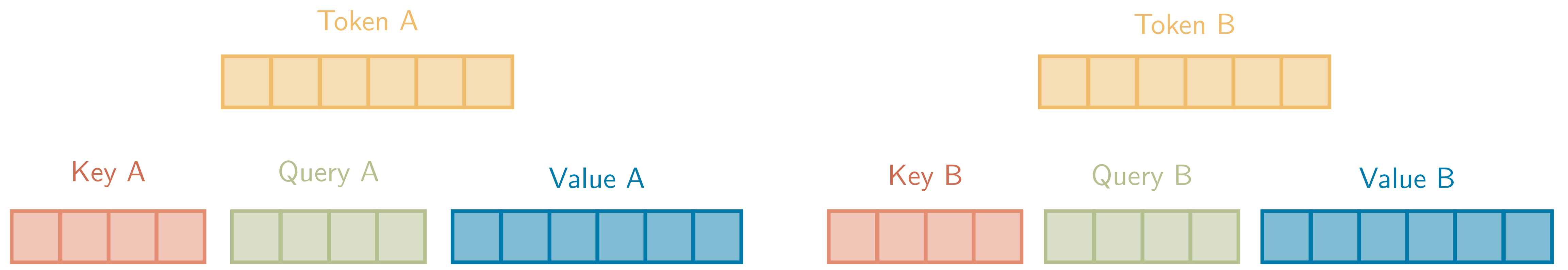
Learning from text

The attention mechanism



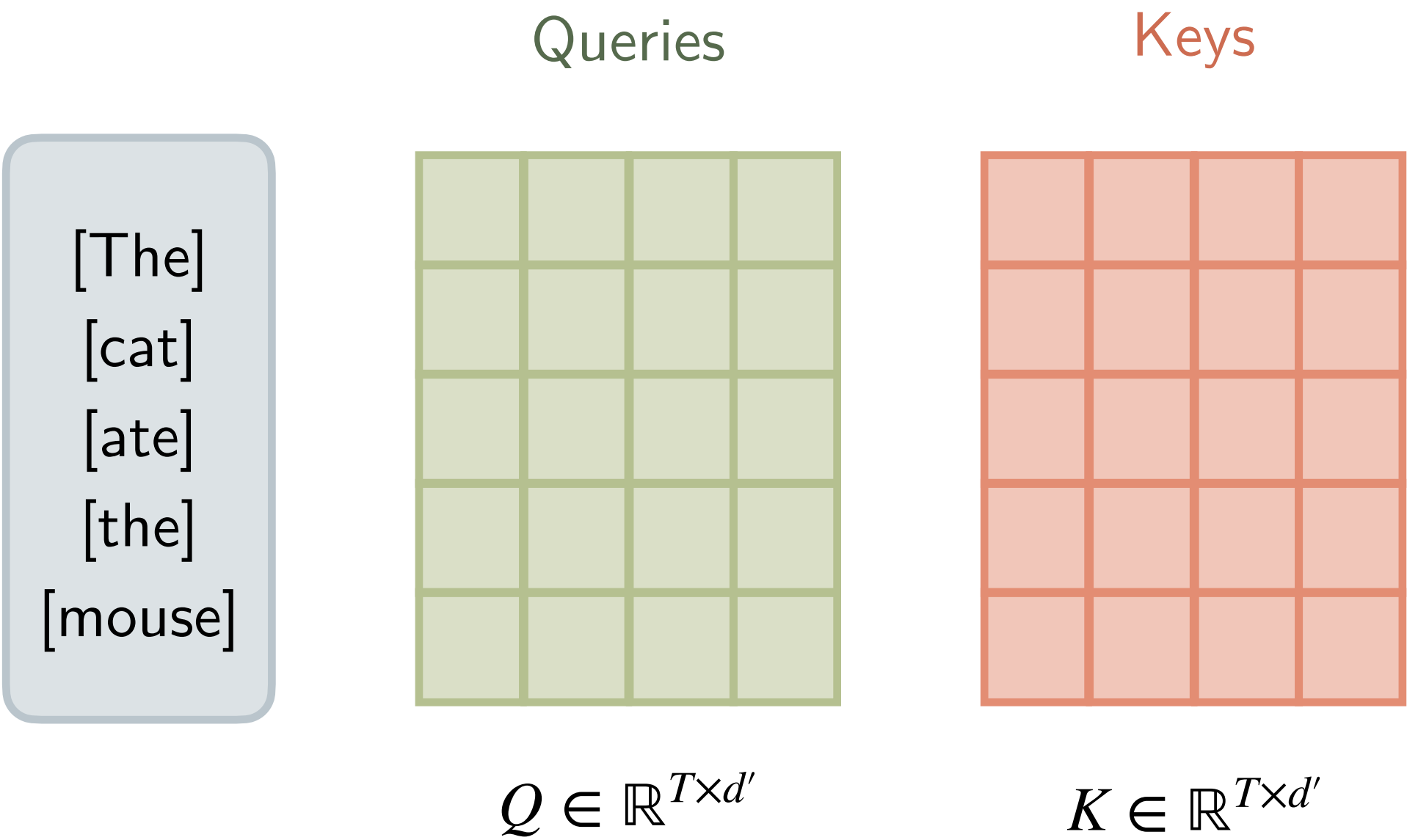
Learning from text

The attention mechanism



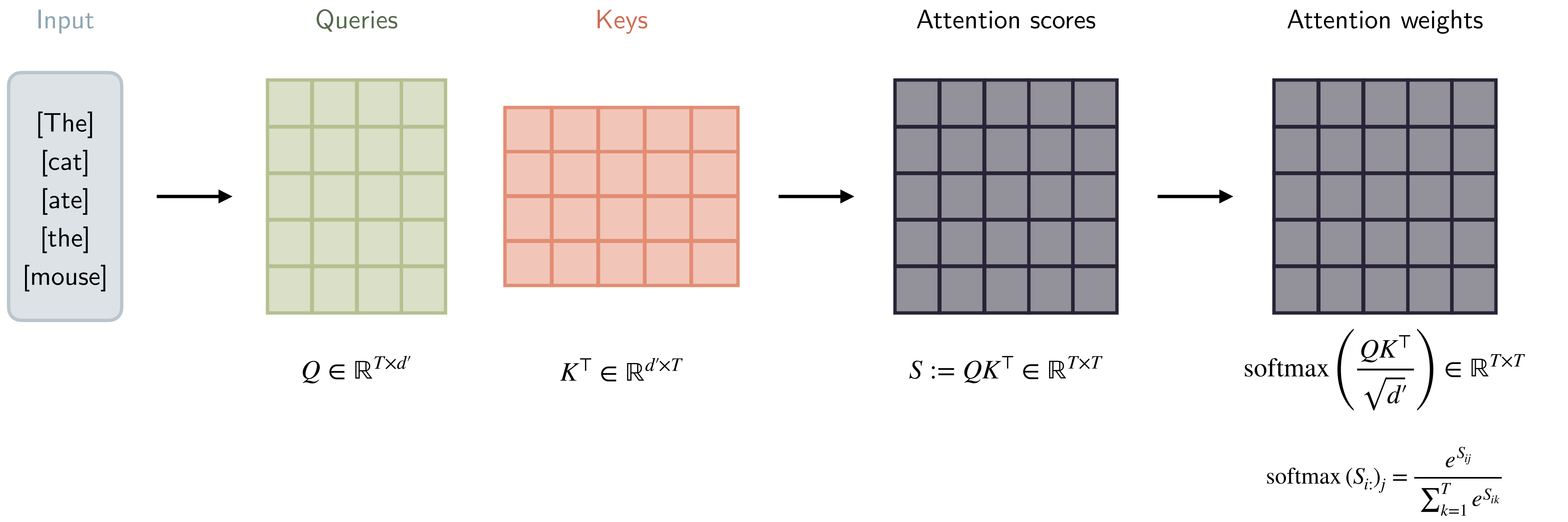
Learning from text

The attention mechanism



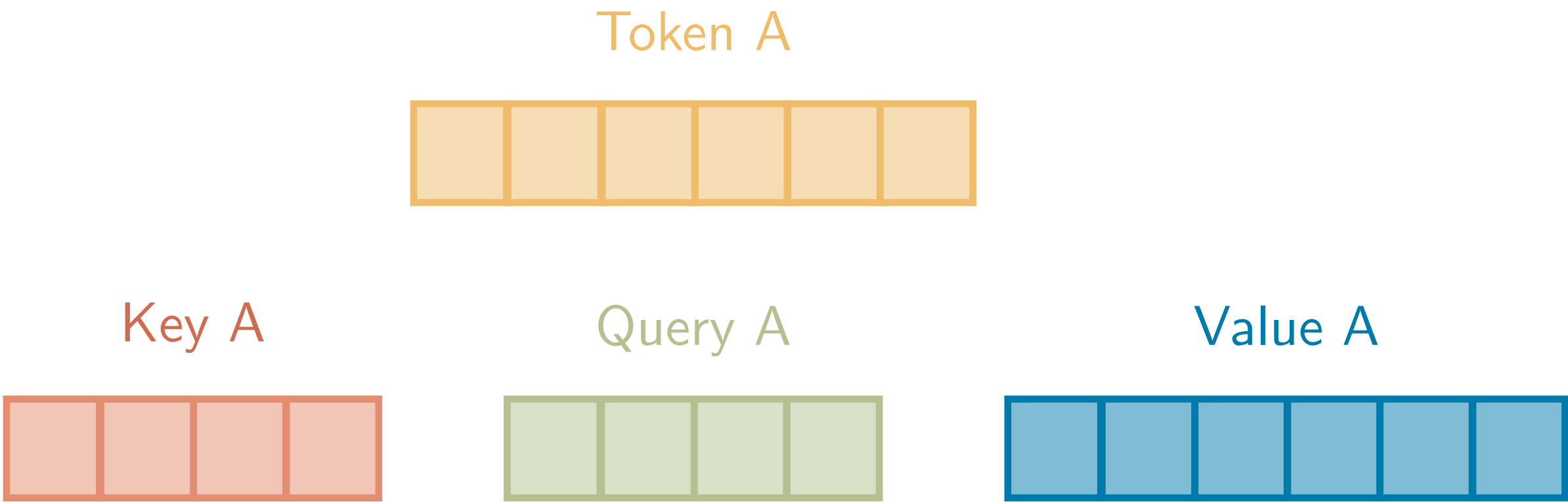
Learning from text

The attention mechanism



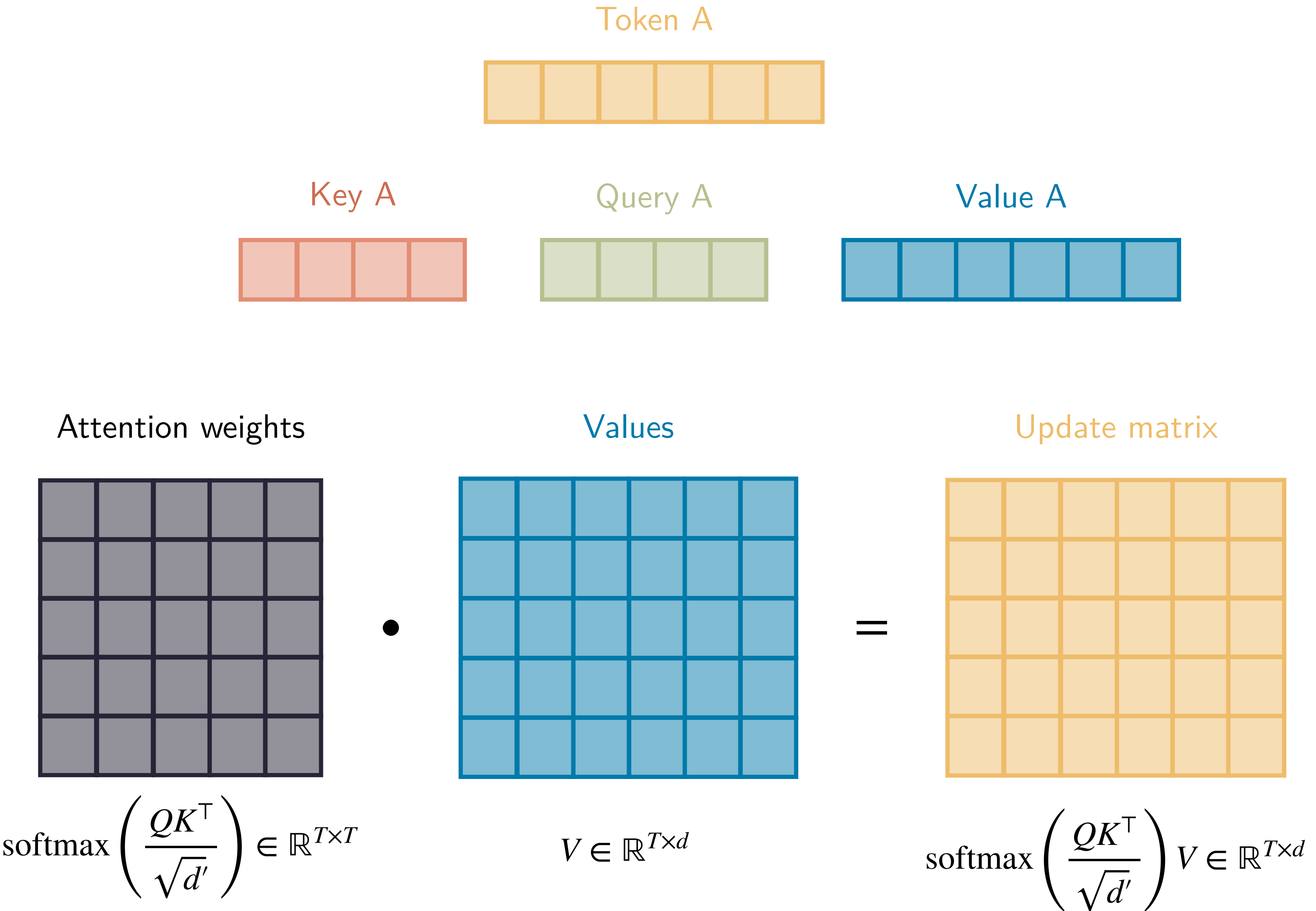
Learning from text

The attention mechanism



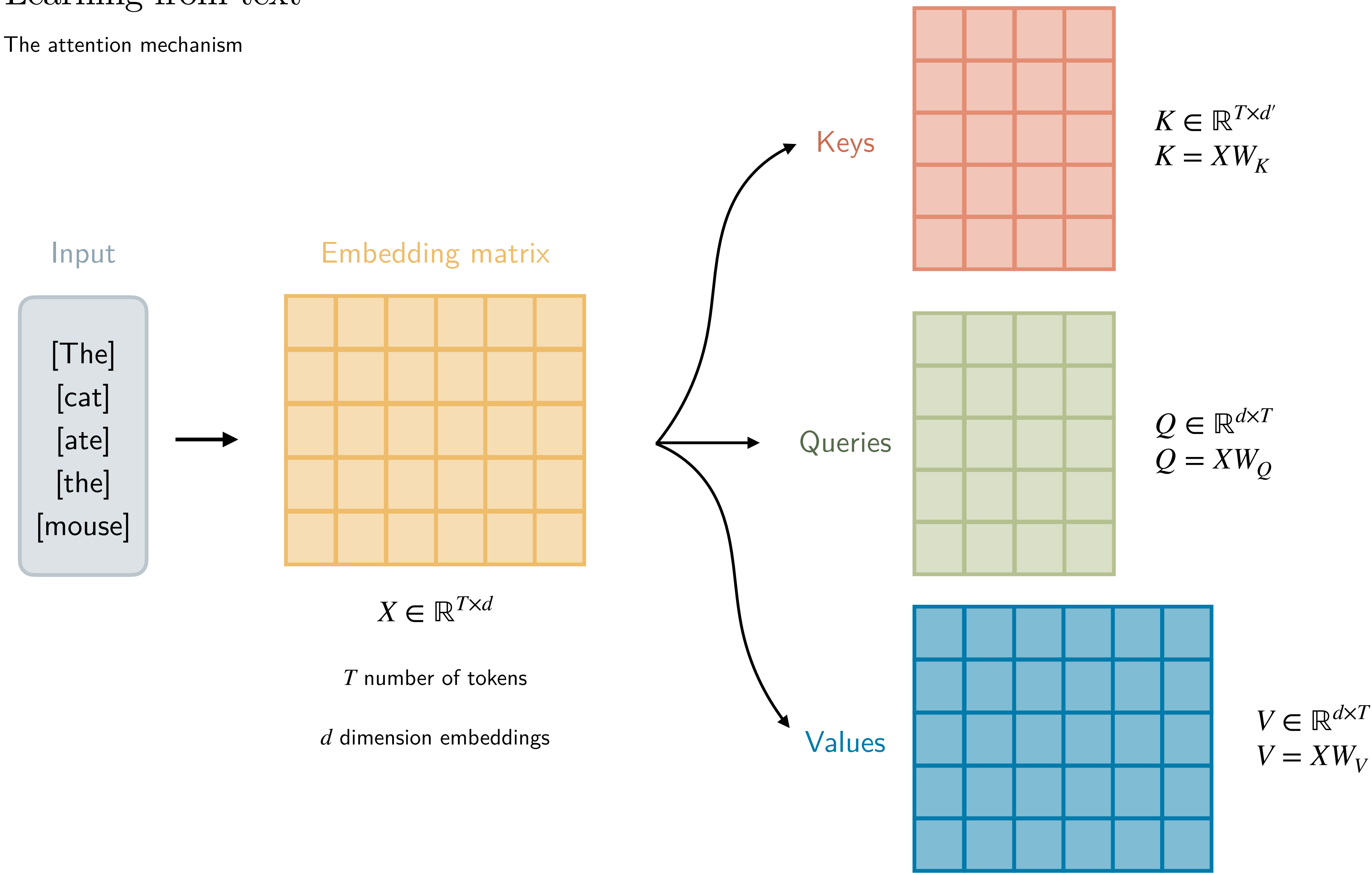
Learning from text

The attention mechanism



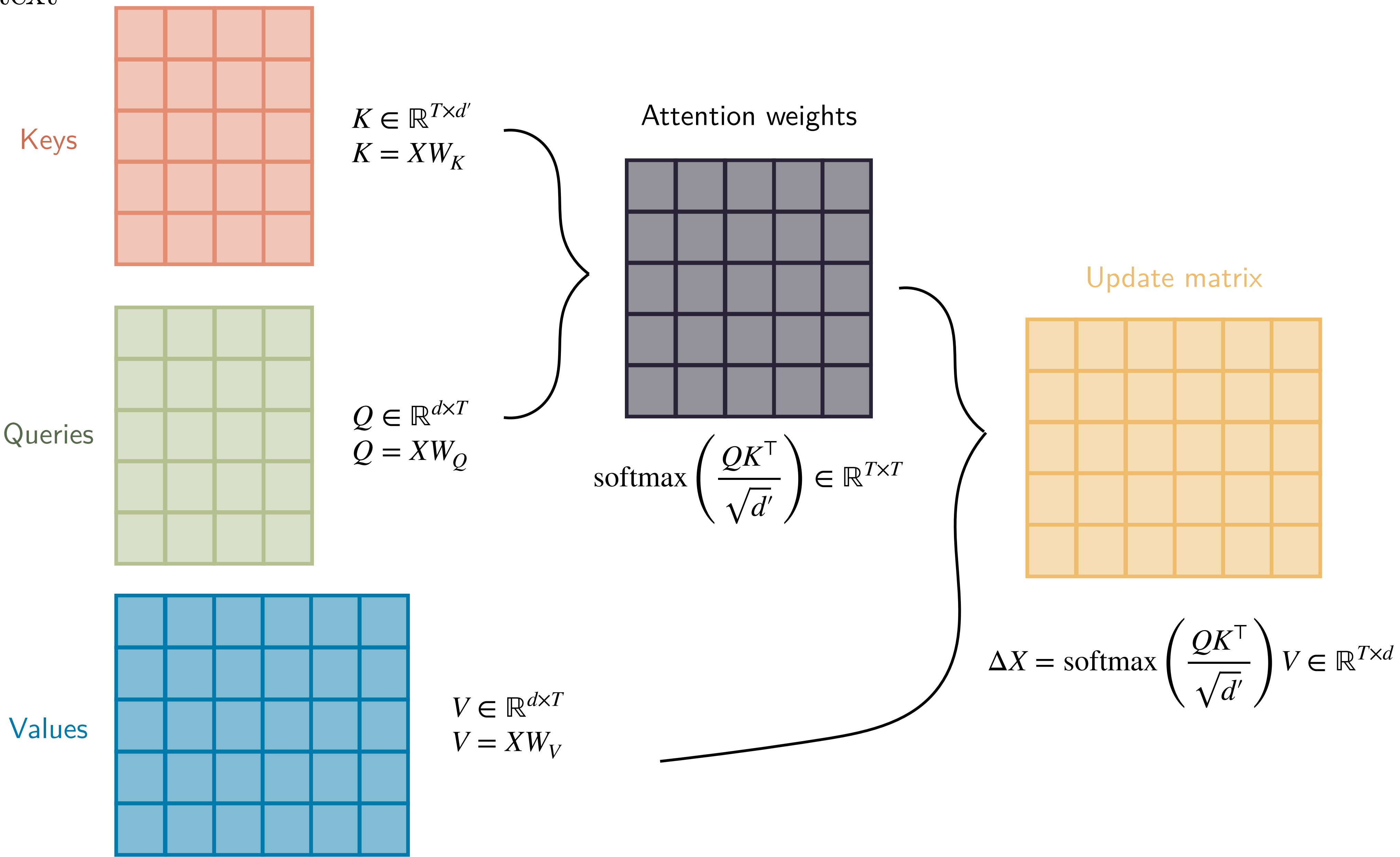
Learning from text

The attention mechanism



Learning from text

The attention mechanism

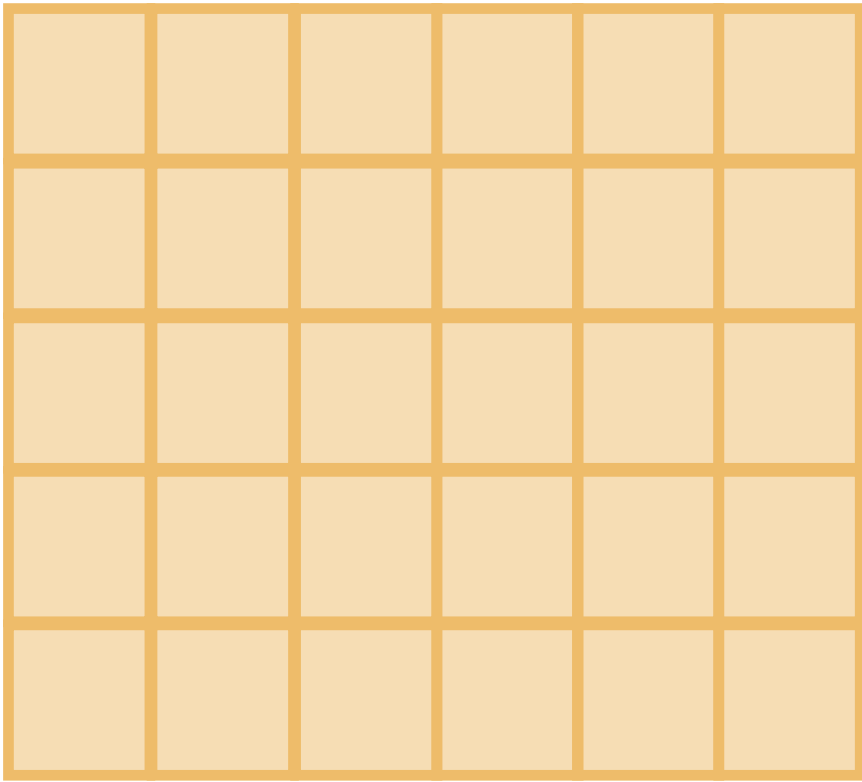


Learning from text

The attention mechanism

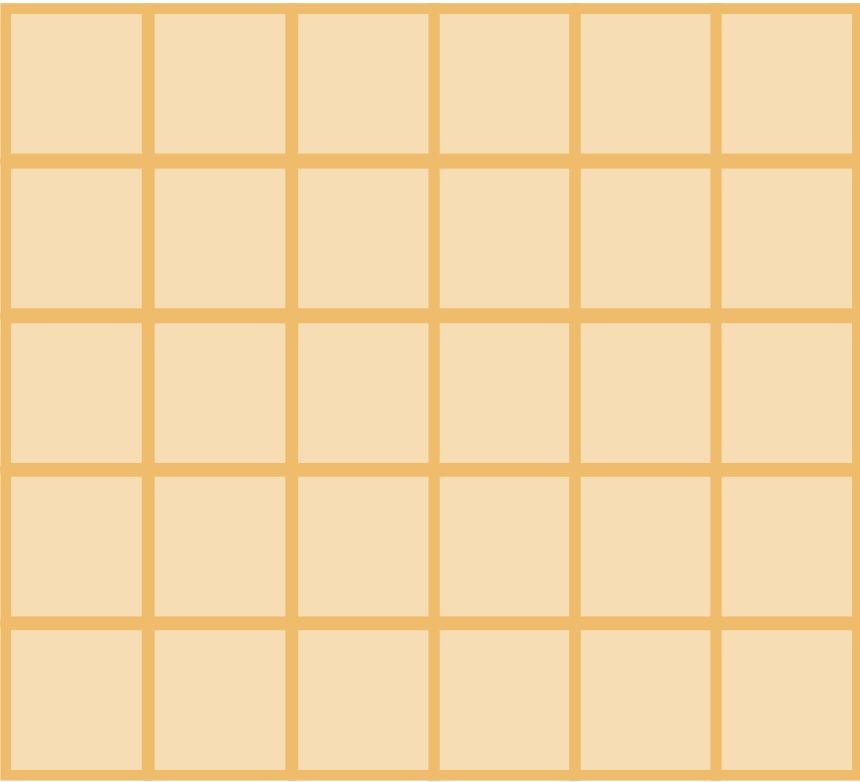
Attention head

Embedding matrix



$$X \in \mathbb{R}^{T \times d}$$

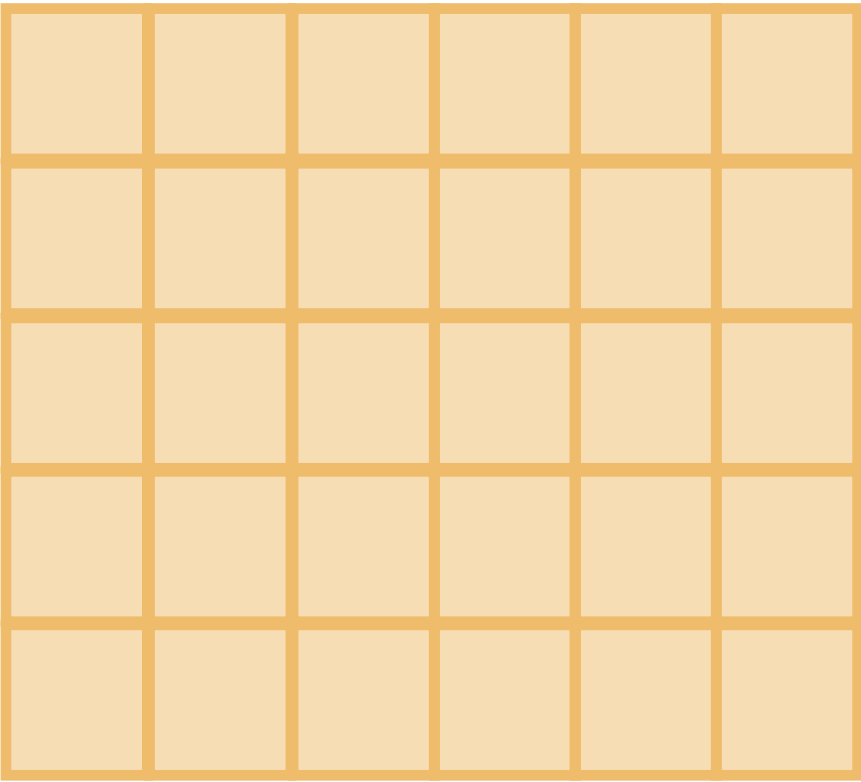
Update matrix



+

=

Updated embeddings



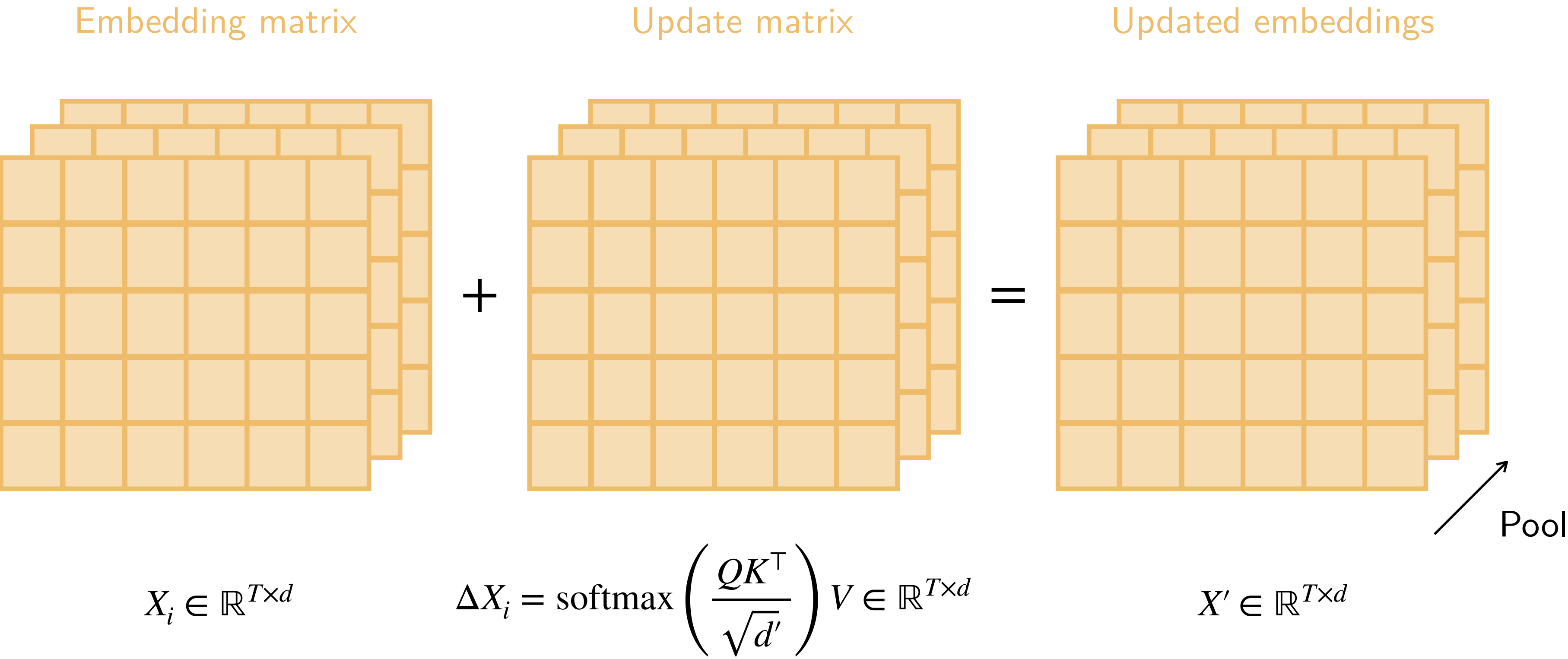
$$X' \in \mathbb{R}^{T \times d}$$

$$\Delta X = \text{softmax} \left(\frac{QK^\top}{\sqrt{d'}} \right) V \in \mathbb{R}^{T \times d}$$

Learning from text

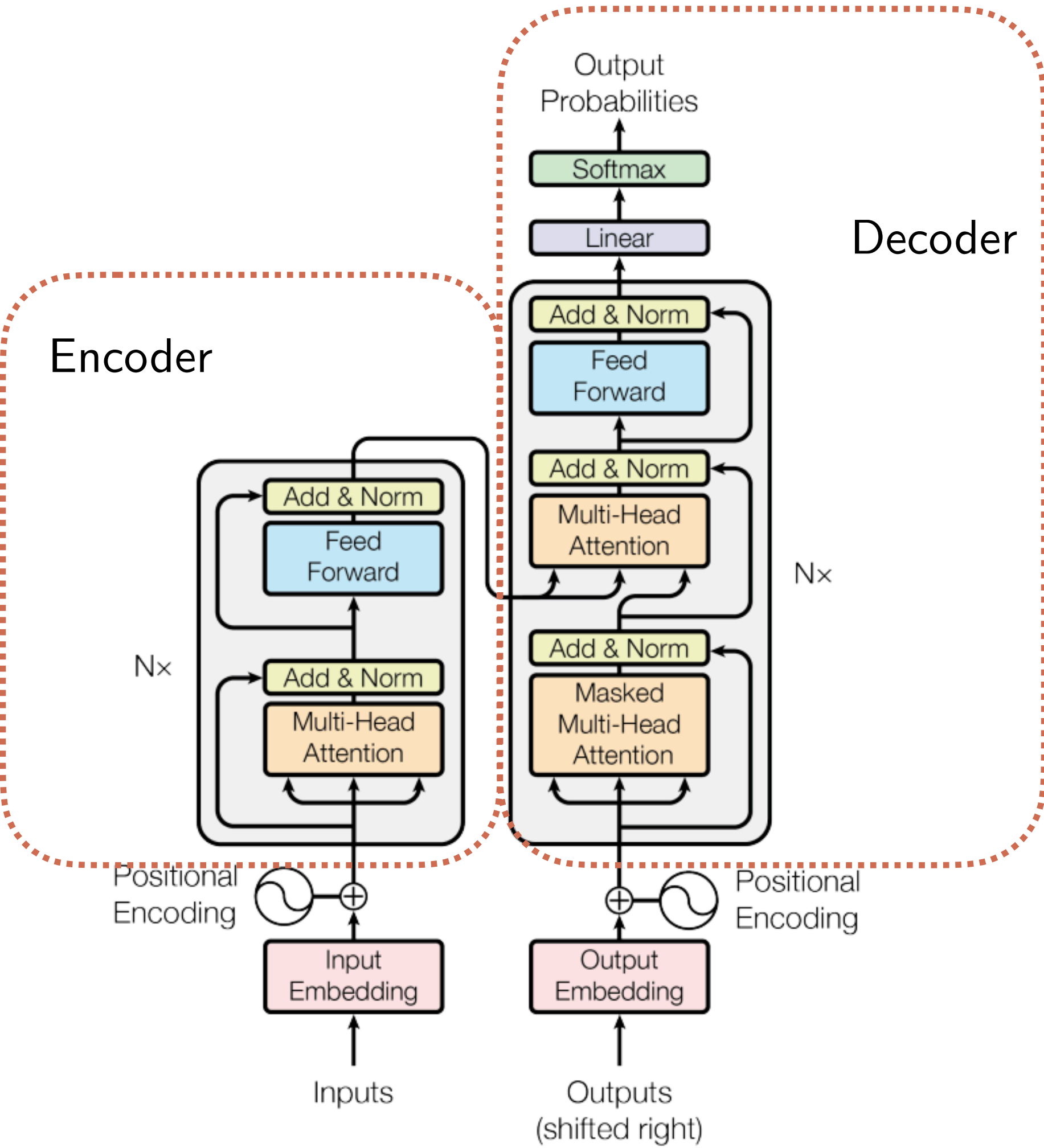
The attention mechanism

Multi-Head Attention



Learning from text

Transformers and the attention mechanism



- 1. **Predict masked tokens:**
The [mask] ate the mouse
- 2. **Next token prediction:**
The cat ate the...
- 3. **Translation**

A. Vaswani, S. Noam, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin
Attention is All You Need, NeurIPS 2017

Lecture 1 — Introduction to Deep Learning

To sum up

Today we have learnt about:

- The basic, foundational neural network architecture: the multilayer perceptron
- How to train neural networks in practice
- How to adapt your architecture to internal structure and qualities of a given input data:
 - Images and convolutional neural networks
 - Text and the attention mechanism

Next week we will overview geometric structures that can be used to describe real data, and from which we will want to learn

We will also see how to describe their geometric and topological properties

Thank you for your attention!



AIDOS LAB
AI FOR DATA-ORIENTED SCIENCE

