Introduction to deep learning

Lecture 1

Sam van Gool
vangool@irif.fr

Master 2 informatique
Université Paris Cité

7 January 2025

Today's lecture

What is deep learning?

- Machine learning: a statistical approach to construct programs.
- Deep neural networks: a particular class of such programs.
- Used for problems where it is hard to define *deterministic* rules, e.g.:
  - Recognition of text, images or sound;
  - Generation of text or images.

Course content

Objective

Understand deep learning models:

how they *work*, how they are *trained*, how they are *used*.

Main topics:

1. Regression: learning to draw lines through data
2. Neural networks: composing linear and non-linear models
3. Backpropagation: computing gradients, efficiently and automatically
4. Training: optimizing and regularizing models
5. Models for vision: convolutional networks
6. Models for text generation: transformers

Practical information

- Course webpage: `https://samvangool.net/iap/`
- This course combines:
  - *programming*: object-oriented Python with `numpy` and `pytorch`;
  - *mathematics*: gradients, graphs, matrices, probability.
- 11 weeks, Tuesday 13h30-16h45.
  - 13h30-15h: lecture, HaF 247E (S. van Gool)
  - 15h15-16h45: lab, Sophie Germain 1009 (G. Baudart & S. van Gool)
- Evaluation: programming work (50%), final exam (50%)
- Textbooks: freely available online: [P] and [Z]. **Read them!**

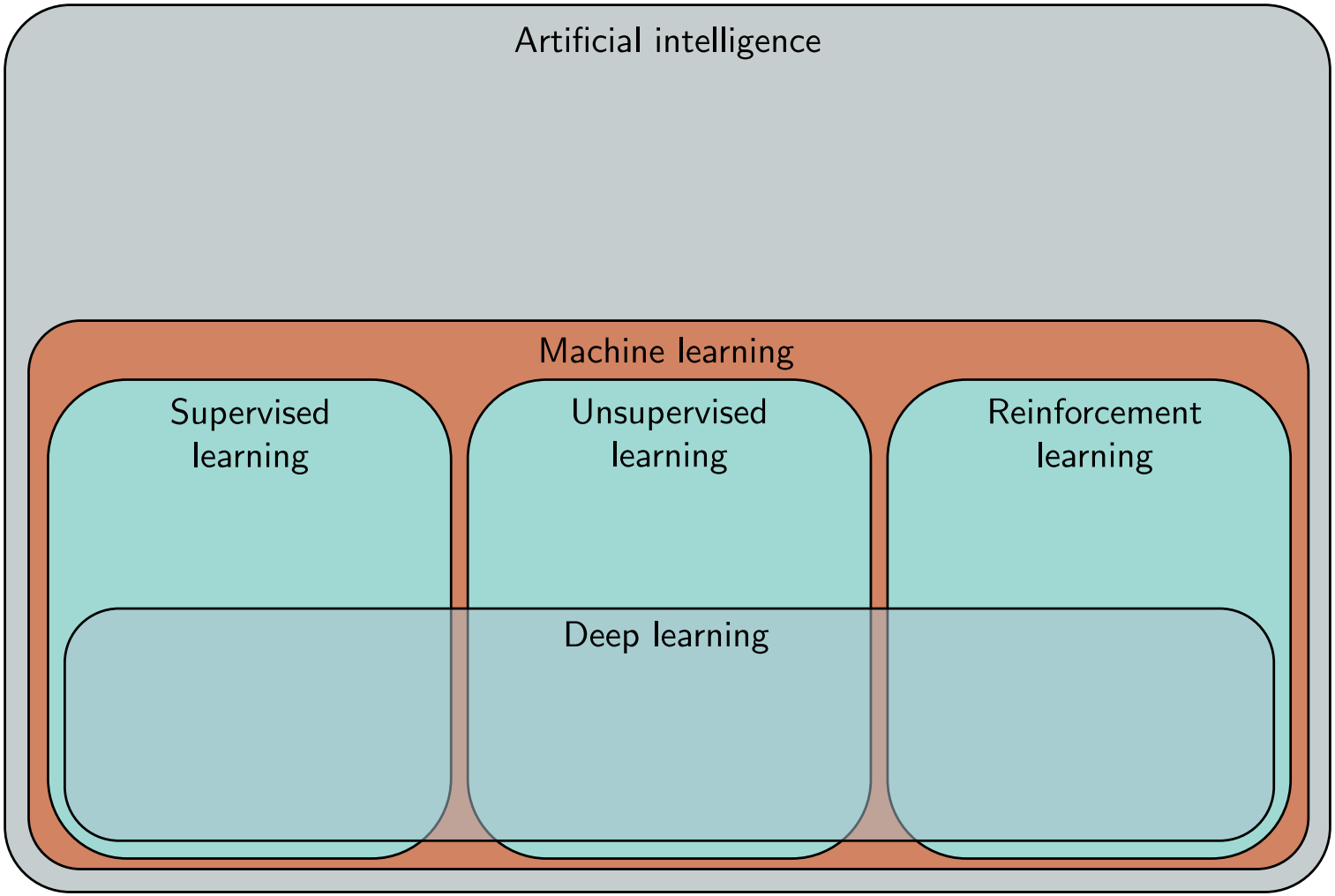
  [P] Prince, S., **Understanding Deep Learning**, `https://udlbook.com`

  [Z] Zhang, A. et al., **Dive into Deep Learning**, `https://www.d2l.ai`
- These slides are available on the course webpage. Many images are copied from the book [P]. (CC-BY-NC-ND license)

Today's lecture

1. Course overview
2. Machine learning concepts
3. Linear regression
4. Numpy
5. Learning deep learning

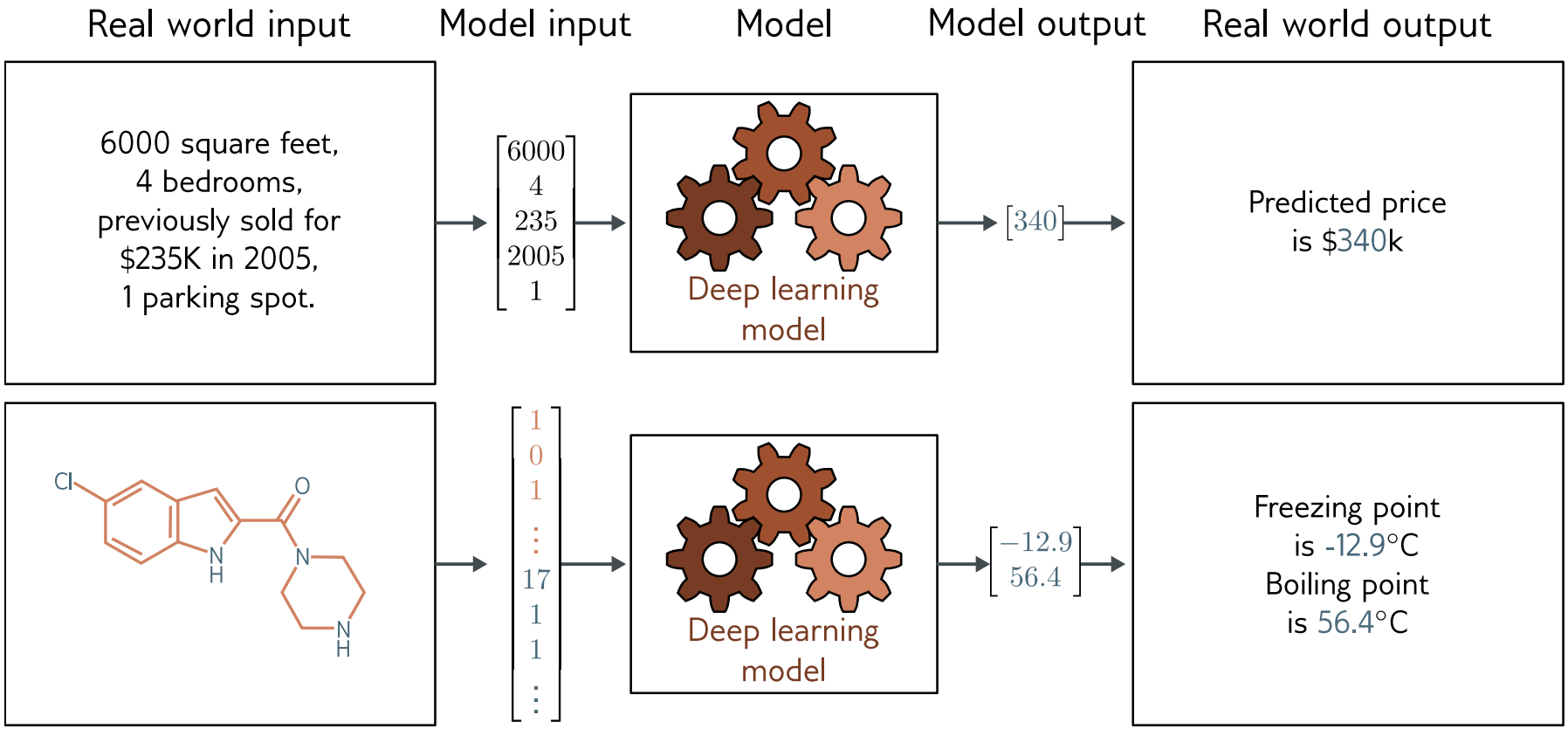
Taxonomy



[P, Figure 1.1]

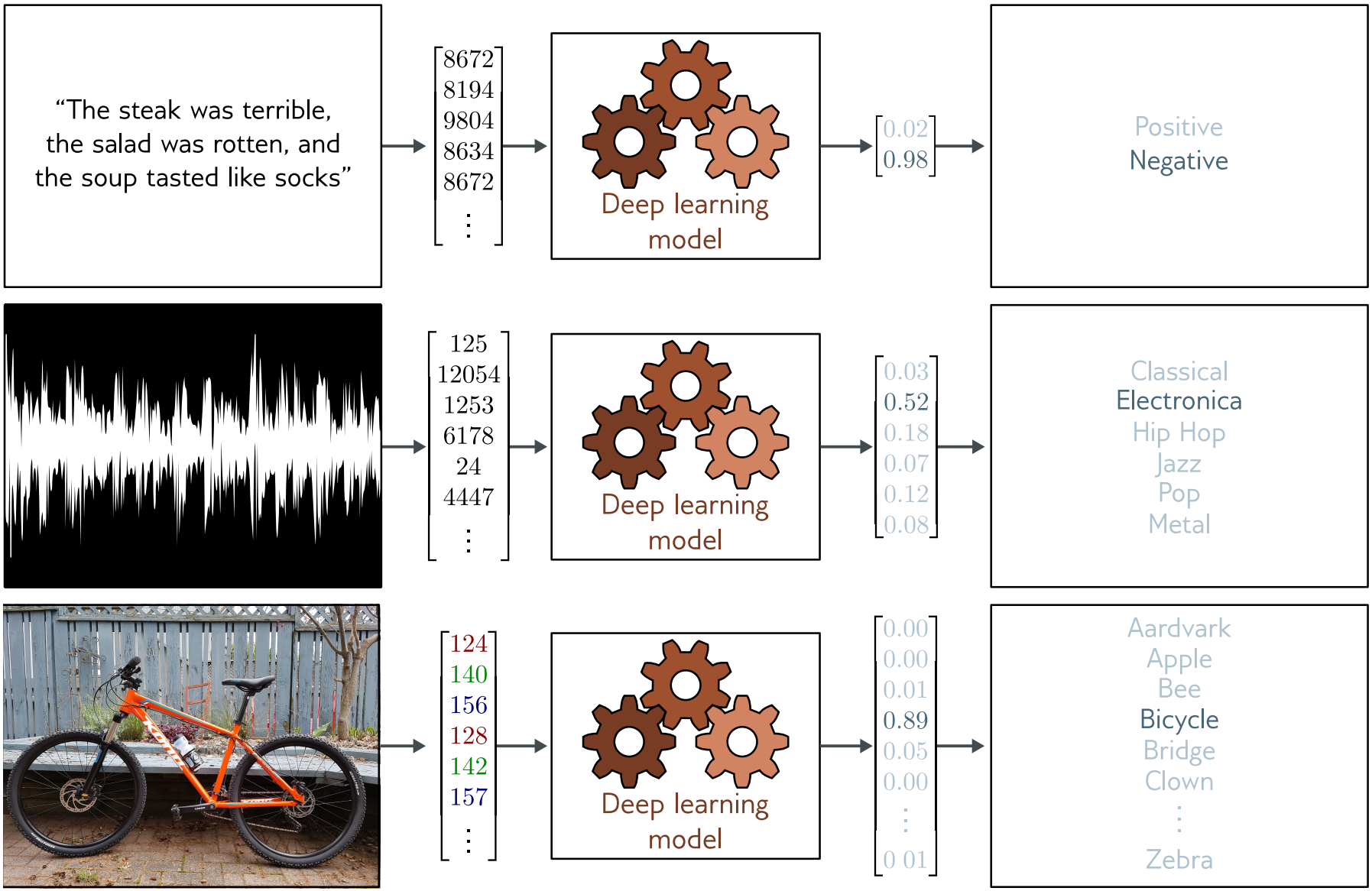
(Note: We do not treat Reinforcement learning in this course.)

Supervised learning



[P, Figure 1.2(a)-(b)]

Supervised learning



[P, Figure 1.2(c)-(e)]

Supervised learning

Type of output:

- Regression: given input, predict value (*continuous output*)
  - A single value or multiple values.
- Classification: given input, choose class (*discrete output*)
  - Just two classes (*binary*) or more (*multi-class*).

- Tabular: list of values.
- Structured: image, audio, text, or other geometric structure.

Reflection

How to encode image, audio, or text input? How big is the encoding?

Supervised learning

Type of output:

- Regression: given input, predict value (*continuous output*)
  - A single value or multiple values.
- Classification: given input, choose class (*discrete output*)
  - Just two classes (*binary*) or more (*multi-class*).

- Tabular: list of values.
- Structured: image, audio, text, or other geometric structure.

Reflection

How to encode image, audio, or text input? How big is the encoding?

Supervised learning

Type of output:

- Regression: given input, predict value (*continuous output*)
  - A single value or multiple values.
- Classification: given input, choose class (*discrete output*)
  - Just two classes (*binary*) or more (*multi-class*).

- Tabular: list of values.
- Structured: image, audio, text, or other geometric structure.

Reflection

How to encode image, audio, or text input? How big is the encoding?

Supervised learning

Type of output:

- Regression: given input, predict value (*continuous output*)
  - A single value or multiple values.
- Classification: given input, choose class (*discrete output*)
  - Just two classes (*binary*) or more (*multi-class*).

- Tabular: list of values.
- Structured: image, audio, text, or other geometric structure.

Reflection

How to encode image, audio, or text input? How big is the encoding?

Supervised learning

Type of output:

- Regression: given input, predict value (*continuous output*)
  - A single value or multiple values.
- Classification: given input, choose class (*discrete output*)
  - Just two classes (*binary*) or more (*multi-class*).

Type of input:

- Tabular: list of values.
- Structured: image, audio, text, or other geometric structure.

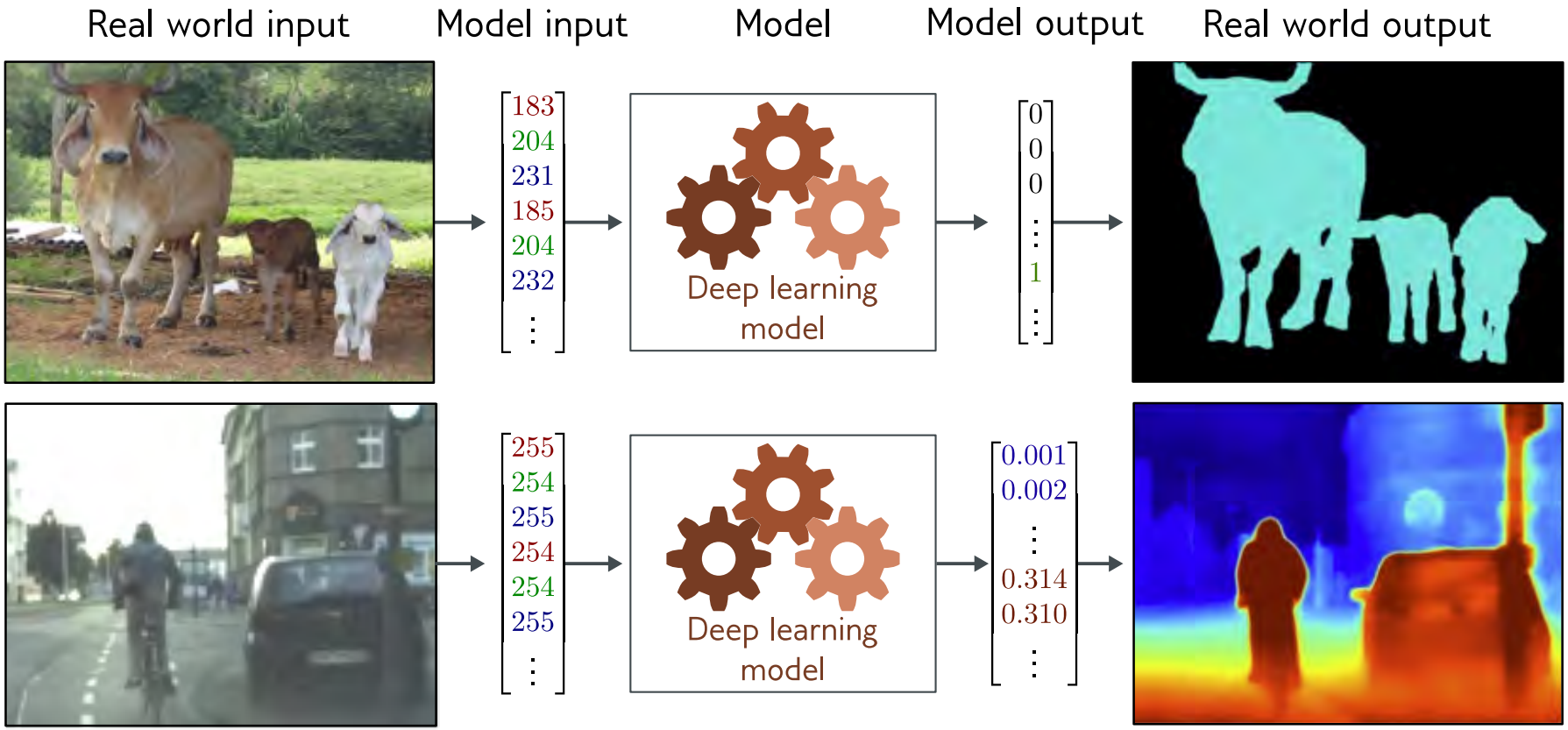
Reflection

How to encode image, audio, or text input? How big is the encoding?

Machine learning models

- A model is a family of equations that describes a relation between input and output.
- An instance of the model is one equation in the family, described by parameters.
- Training a model: use examples of input-output pairs to find *better* parameters.
- Deep neural networks: a particular class of models.

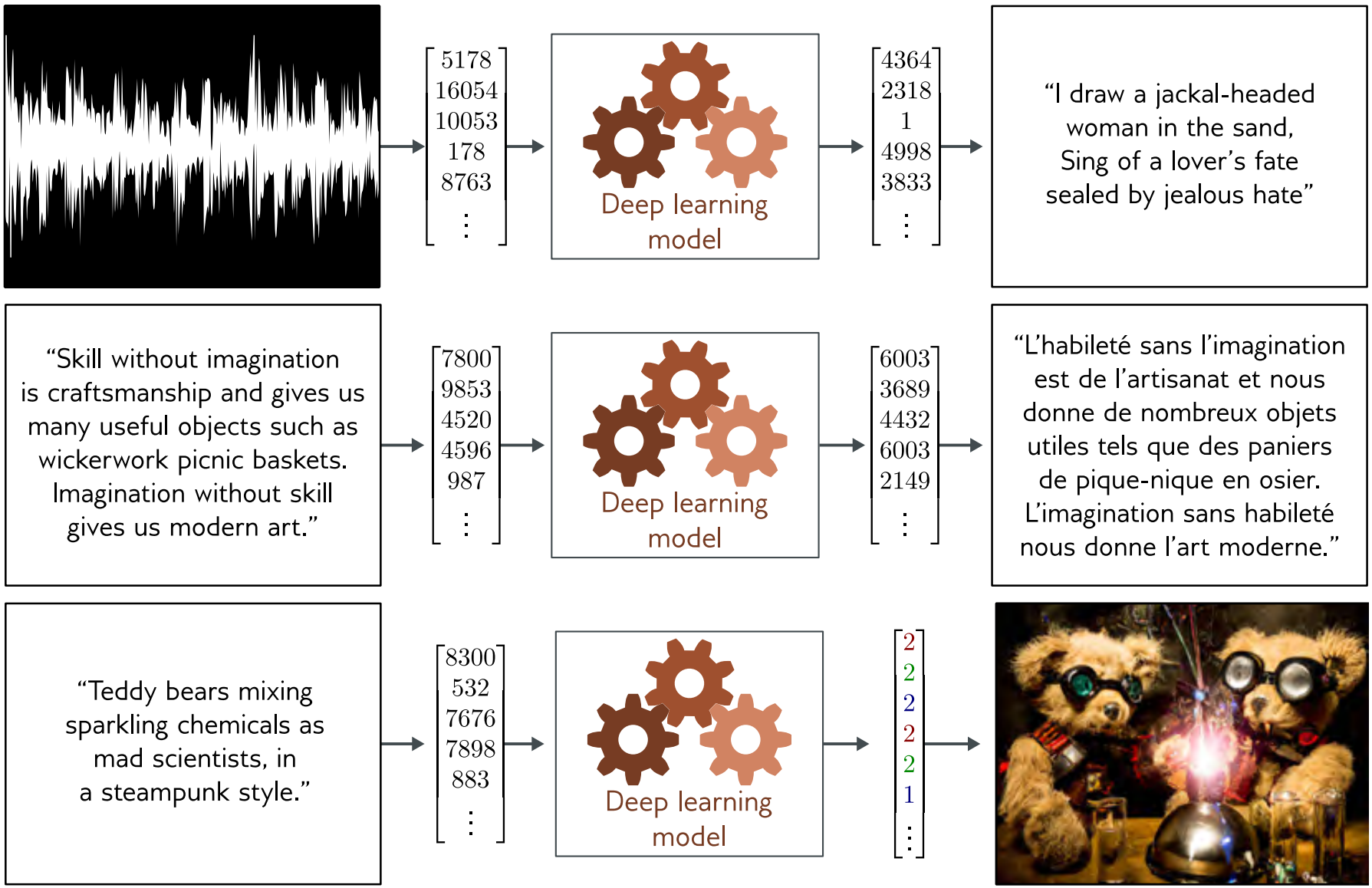
Structured output



[P, Figure 1.4(a)-(b)]

- Binary vs. multivariate classification.
- Close pixels in input $\rightsquigarrow$ pixels in output correlated.

Structured output



[P, Figure 1.4(c)-(e)]

Structured output

The last three examples are more difficult to achieve. Why?

- There is *more than one* correct answer.
- The outputs are also *structured* by 'grammar' rules:
  - It is difficult to 'randomly guess' a collection of pixels that creates a good picture.

Supervised vs. unsupervised learning

- Supervised learning: Given many input-output pairs (x_i, y_i), learn to predict y given x.
- Unsupervised learning: Given many input samples x_i, learn to **generate** new examples x that are similar to the inputs.
- For example, GPT = **Generative** Pretrained Transformer.

Reflection

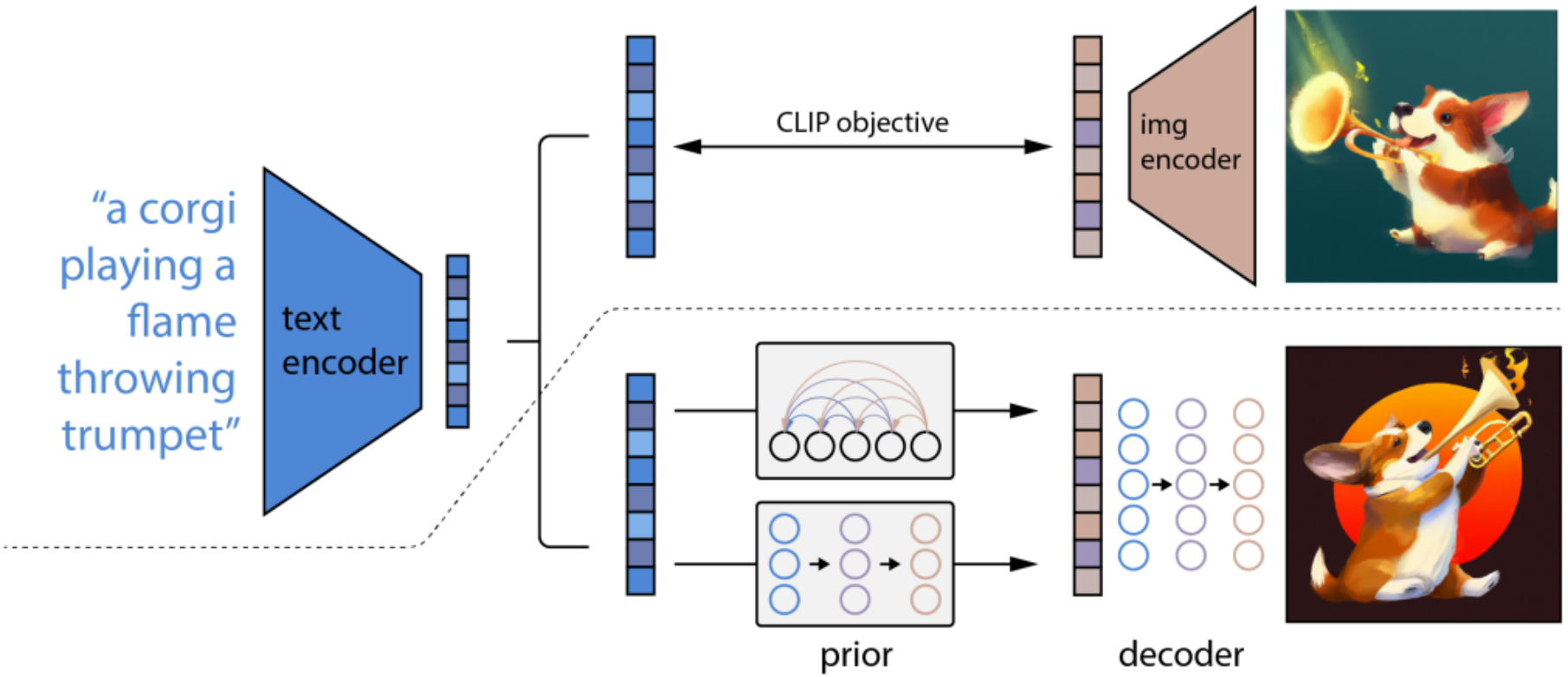
Are the following tasks supervised or unsupervised learning problems?

(a) Learn to predict the rainfall on a given date;
(b) Learn to write a weather report, given a collection of weather reports;
(c) Learn to create an image that corresponds to a given caption.

Latent variables

- While (c) could look like a supervised problem, it is typically solved *using* unsupervised learning:
- First unsupervised: learn a *latent variable* representation of the input, and of the output.
- Then supervised: learn a *relationship* between the latent variable representations.
- Main advantage: latent variables are in a lower dimensional space.

Latent variables: Example



[Dall-E 2 article, Figure 2]

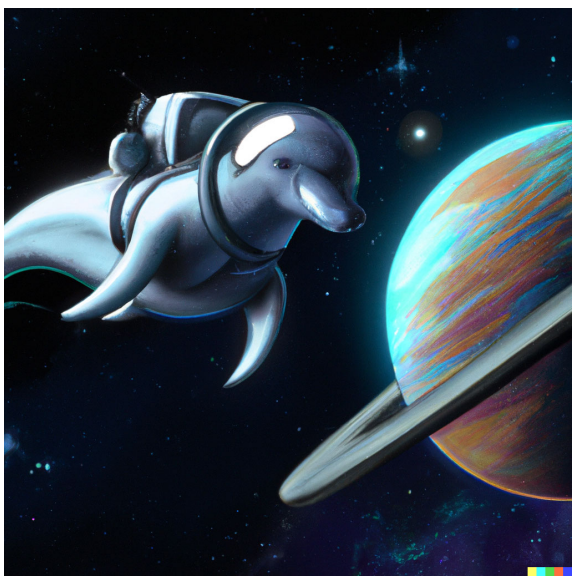
- First learn a representation of texts and images (unsupervised).
- Then 'decode' an un-seen input text to create a new image.

Dall-E 2: Examples

an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

a corgi's head depicted as an explosion of a nebula



a dolphin in an astronaut suit on saturn, artstation

a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

a teddy bear on a skateboard in times square

Figure 1: Selected 1024×1024 samples from a production version of our model.

Ethics of deep learning

I am not a lawyer, nor a philosopher, but just a few remarks:

(see also [P, Section 1.4 and Chapter 21])

- Any new technology may be used for good and bad.
- There are numerous ethical issues around (deep) machine learning:
  - Algorithmic bias
  - (Non-)Explainability of algorithmic decision making
  - Intellectual property, e.g. of news corporations and artists.
- These issues are **important** and you should learn about them.
- They are not the focus of this course.
- See for example `https://ethics-of-ai.mooc.fi`.

Today's lecture

1. Course overview
2. Machine learning concepts
3. Linear regression
4. Numpy
5. Learning deep learning

A linear model

Problem

Given input data **x** *with corresponding output data* **y**, *draw a* **line** *that approximates the data.*

- A line is a function f given by the formula

$$f[x, \phi] = \phi_0 + \phi_1 \cdot x.$$

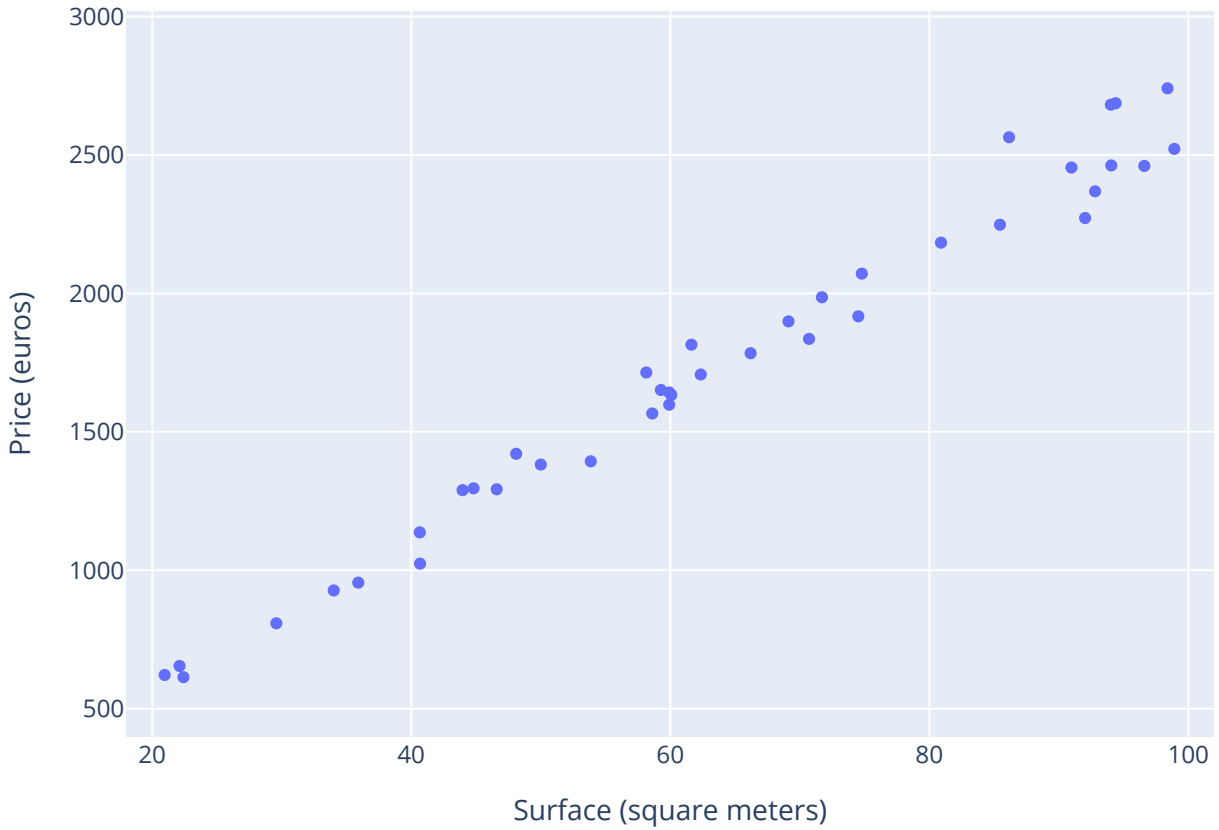
- Here, x is the input variable and ϕ_0 and ϕ_1 are parameters.
- If we fix the 2-element vector $\phi = (\phi_0, \phi_1)$, then a line can be used to predict the output y for a given input x: just compute $f[x, \phi]$.

Greek letter alert

The letter ϕ, also written as φ, is the Greek letter 'phi'.
It is different from θ, also written as ϑ, which is 'theta'.

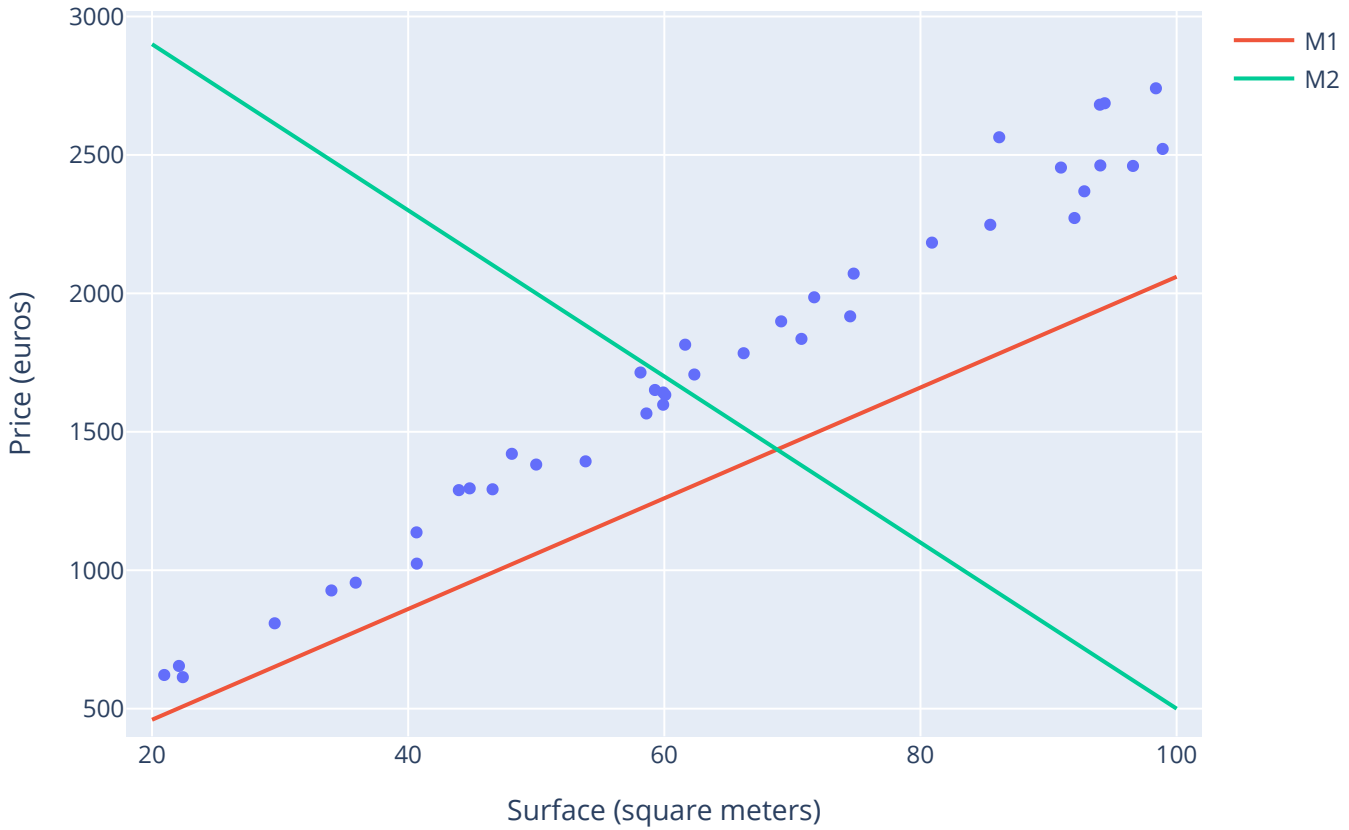
Example: monthly rent based on surface

- For example: input ($\mathbf{x}$) is the surface in square meters of an apartment, output ($\mathbf{y}$) is the monthly rental price.



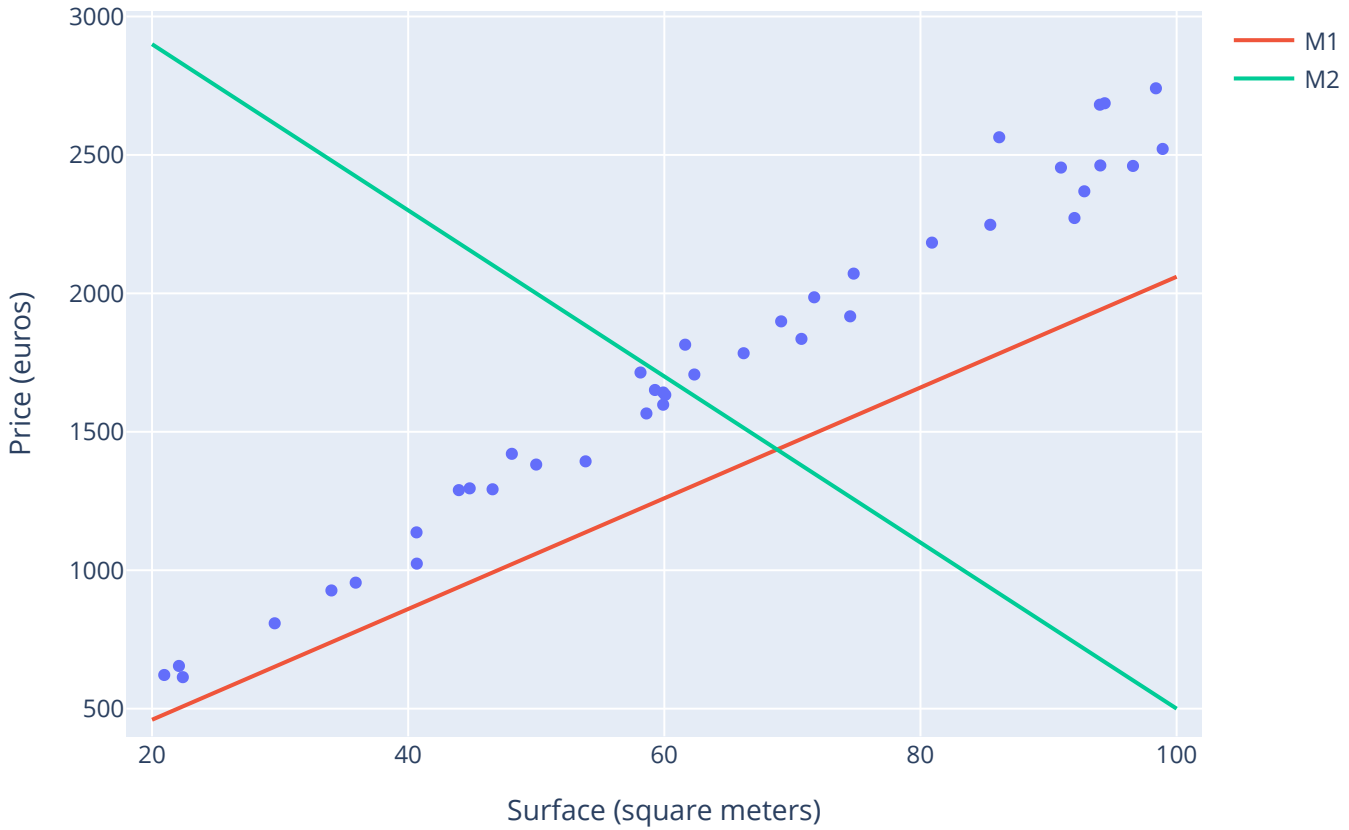
Example: monthly rent based on surface

- For example: input ($\mathbf{x}$) is the surface in square meters of an apartment, output ($\mathbf{y}$) is the monthly rental price.



Example: monthly rent based on surface

- For example: input ($\mathbf{x}$) is the surface in square meters of an apartment, output ($\mathbf{y}$) is the monthly rental price.



The red line is 'clearly' *better* than the green line. Why?

Loss

- We quantify *how bad* a model fits the data with a loss function.
- Lower loss = better model.
- For example, the total square loss over data $(x_1, y_1), \ldots, (x_n, y_n)$ is

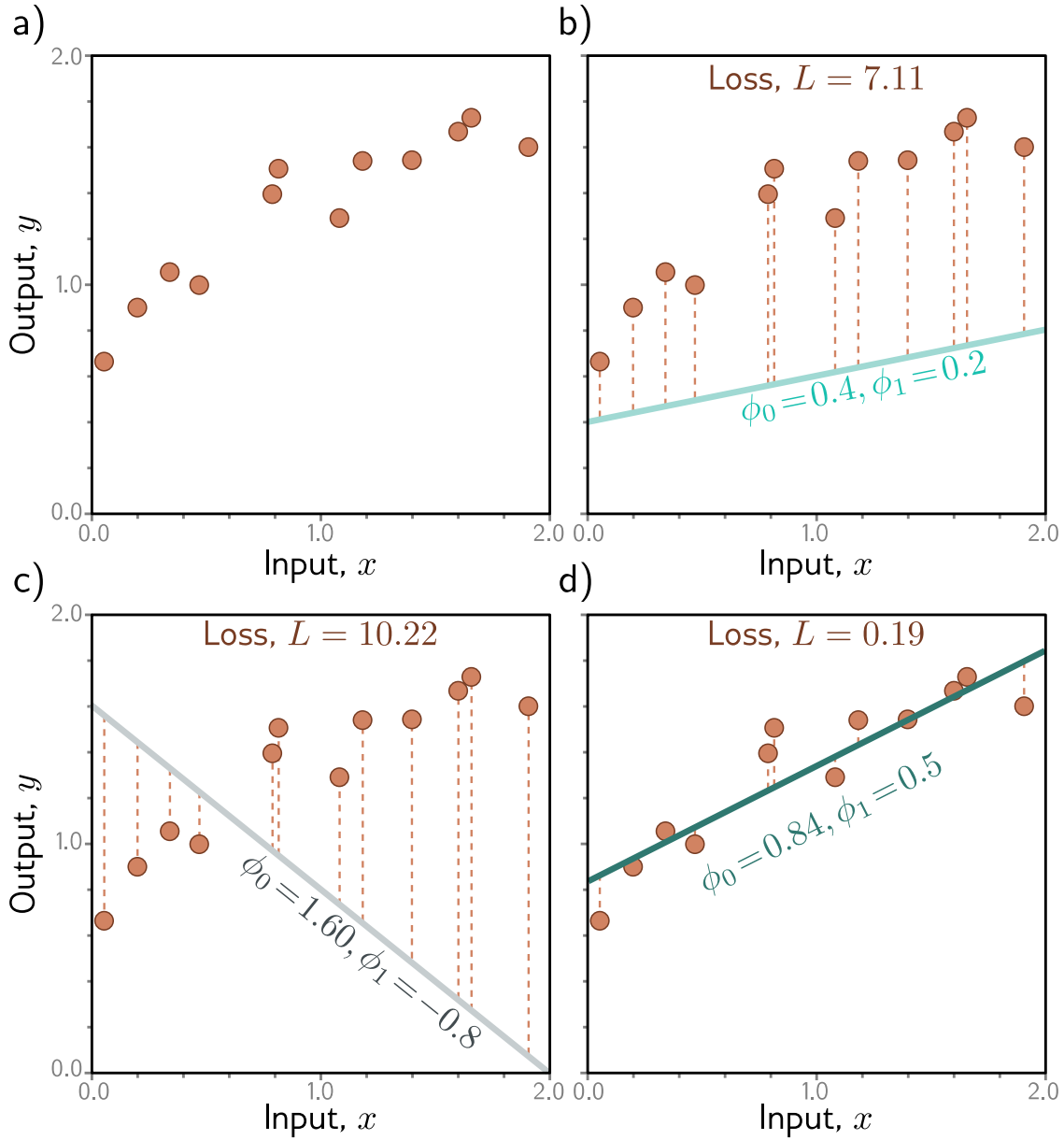
$$L[\phi] = \sum_{i=1}^{n} (f[x_i, \phi] - y_i)^2.$$

  It is the sum over all the training data (x_i, y_i) of the *square distance* between the prediction $(f[x_i, \phi])$ and the true answer (y_i).
- In the example, the loss of the red line is $\approx 7.4e6$ and the loss of the green line is $\approx 6.8e7$. So the red line is '10$\times$ better'.

Math notation alert

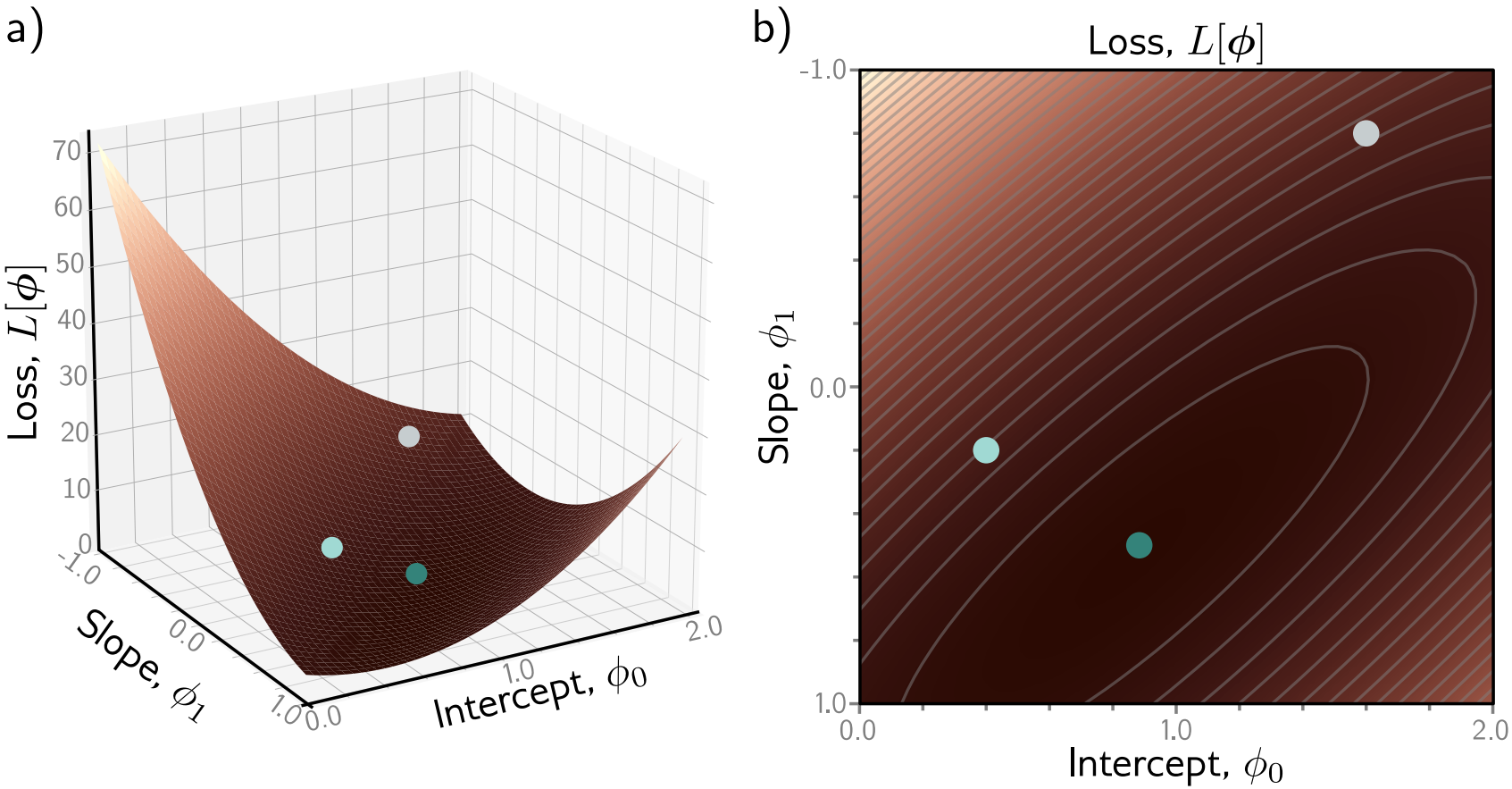
The suffix '$e6$' means: $\times 10^6$. So $7.4e6 = 7.4$ million, $6.8e7 = 68$ million.

Loss visualized



[P, Figure 2.2]

The loss surface



[P, Figure 2.3]

Goal

Find parameters that are **as low as possible** on the loss surface.

Minimizing the loss

Goal

Find parameters that are **as low as possible** on the loss surface.

In a formula, we are looking for $\hat{\phi}$, defined by

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ L[\phi].$$

The idea

Start somewhere on the loss surface, and take steps downwards.

Minimizing the loss

Goal

Find parameters that are **as low as possible** on the loss surface.

In a formula, we are looking for $\hat{\phi}$, defined by

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ L[\phi].$$

The idea

Start somewhere on the loss surface, and take steps downwards.

Minimizing the loss

Goal

Find parameters that are **as low as possible** on the loss surface.

In a formula, we are looking for $\hat{\phi}$, defined by

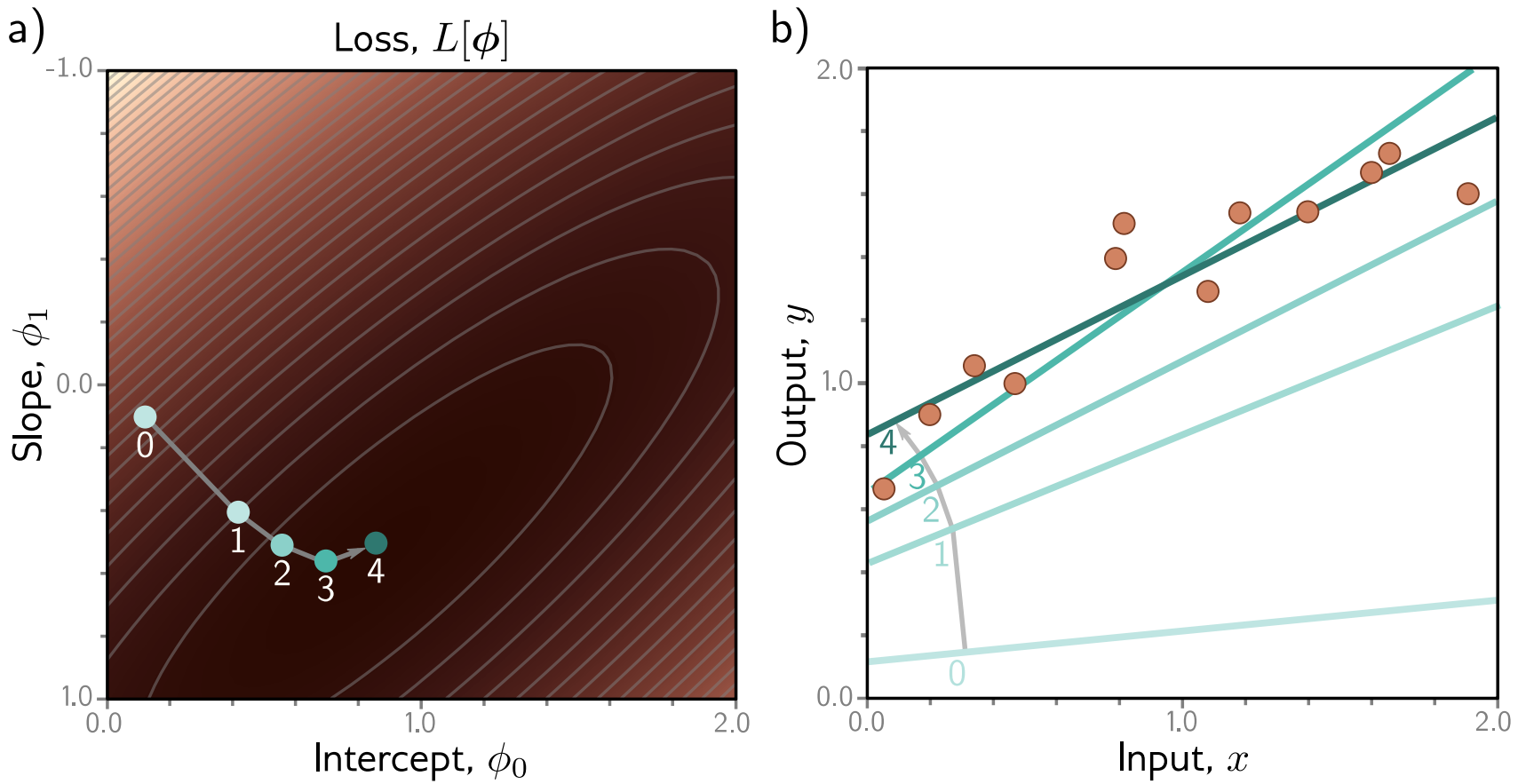
$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ L[\phi].$$

The idea

Start somewhere on the loss surface, and take steps downwards.

This is called gradient descent.

Gradient descent for linear regression, visualized



[P, Figure 2.4]

- On the left: the path through the loss surface.
- On the right: the corresponding changes to the model.

Today's lecture

How to implement all this?

- We will see two main libraries in this course: `numpy` and `pytorch`.
- These are `python` libraries. We use `python` version 3.10 or later.
- Both are libraries for *efficient scientific computing with arrays*.
- Numpy is a more general library for array calculations, Pytorch is more specific for deep learning. The syntaxes are mostly aligned.
- We begin with `numpy` because it is a bit simpler.
- For the labs (TP) you need a machine with these libraries installed.
- We will speak about GPUs later in the course.

Numpy basics

- It is common to use *interactive notebooks* in DL research.
- Alternative: interactive mode of an IDE, e.g. `#%%` in VS Code.
- Choose the workflow that works best for you.
- Short demo:
  - Create arrays;
  - Add and multiply two arrays;
  - Sum an array, along several axes;
  - Multiply an array with a scalar (broadcasting);
  - Draw a scatter plot;
  - Draw a line.
- The TP will allow you to practice with `numpy` by implementing a learning loop for linear regression.

Today's lecture

1. Course overview
2. Machine learning concepts
3. Linear regression
4. Numpy
5. Learning deep learning

Using large language models for studying

- Rules:
  - All text and code you submit is your responsibility.
  - No plagiarism.
  - No external tools during the exam.

- Advice: When using LLMs, exercise **caution** and **critical thinking**.
  - Is this answer true? Correct? Optimal?
  - Do I understand the answer? Could I find it myself without LLM help?

Reflection

How do **you** think using an LLM influences the *depth* of your *learning*?

Self-study

- Deep learning is a large subject. This course is only an *introduction*.
- To get the most out of the course, it is important that you also regularly study by yourself.
- The course is worth 3 ECTS, i.e. approximately **78** hours of study.
- There are **33** contact hours and **45** hours of self-study, i.e., approximately **4 hours per week** of self-study.
- How you use these 4 hours is up to you, but I will suggest some tasks.

TO DO after today, before next Tuesday

- Finish TP 1.
- Read Chapters 1 & 2 in [P] (Section 1.3 is optional).
- Try Problems 2.1 and 2.2 in [P].
- If you are not yet familiar with `numpy`, work through the start of a tutorial, for example [Numpy quickstart] or [Maximov 2020].
  - We will use some *matrix multiplication* and *broadcasting* later.
  - The same basic skills will apply to `pytorch`.