Generalities about Machine Learning

Wasaa - January 2023

Global overview...

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview...

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview...

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview...

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview...

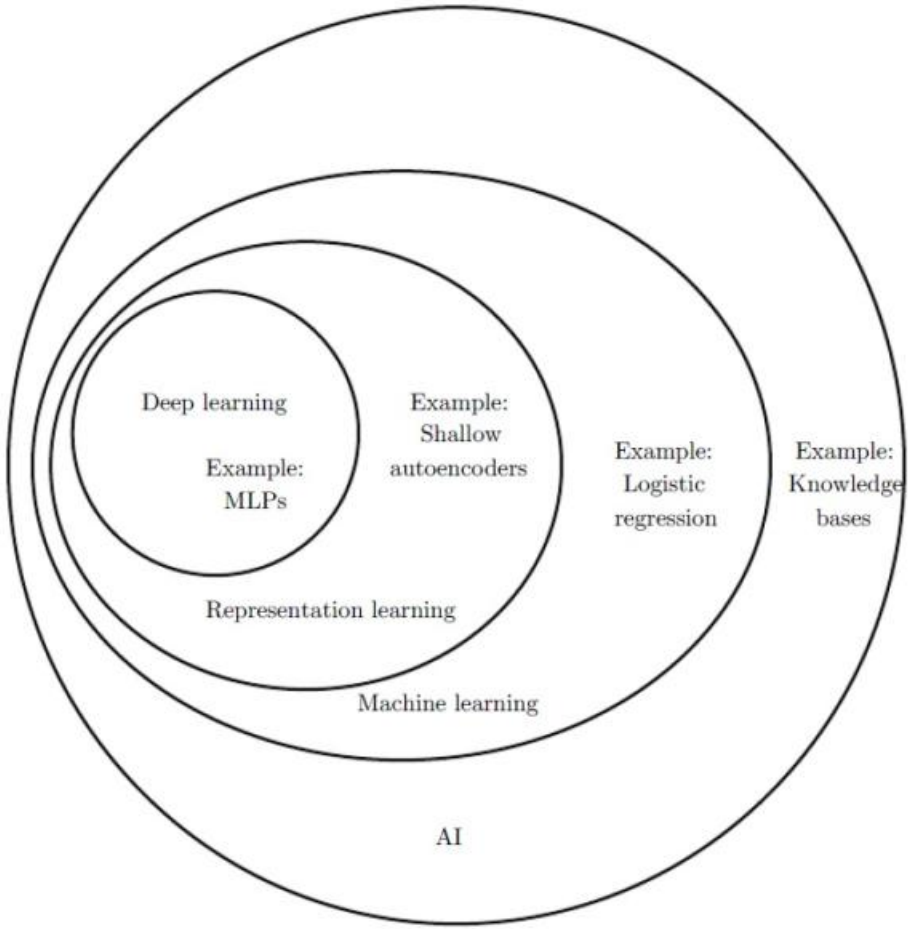


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.

from *Deep Learning, 2016* by A. Courville, I. Goodfellow and Y. Bengio

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples ...

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples ...

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples ...

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

- $f = f_w$ contains **parameters W** to be trained,
- In most cases, an ideal f_w exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

- $f = f_w$ contains **parameters W** to be trained,
- In most cases, an ideal f_w exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

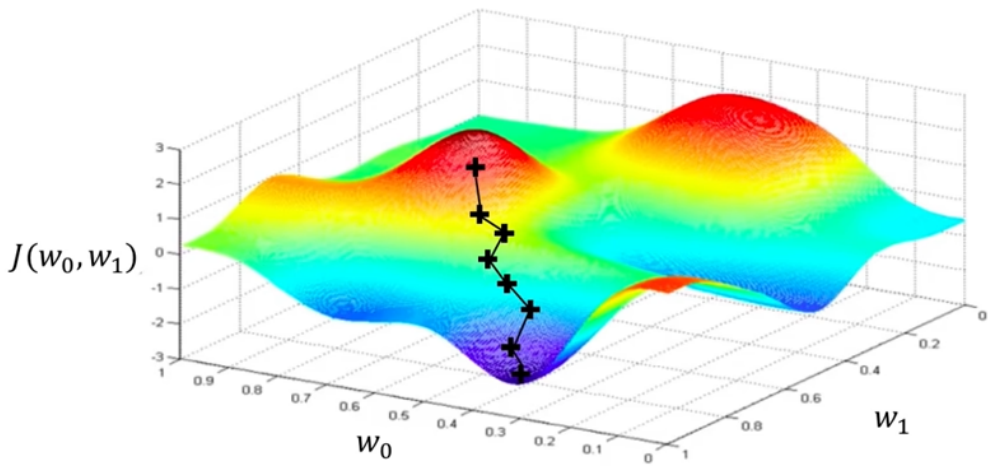
- $f = f_w$ contains **parameters W** to be trained,
- In most cases, an ideal f_w exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

- Loss: $J(\mathbf{W}) = \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}, \mathbf{W}), \mathbf{y}^{(i)}), i = examples$
- Model parameters: $\mathbf{W}^* = argmin(J(\mathbf{W}))$

Training Algorithm

- Randomly Initialize model weights
- Compute Gradient of the Loss $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Update weights $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Repeat until convergence

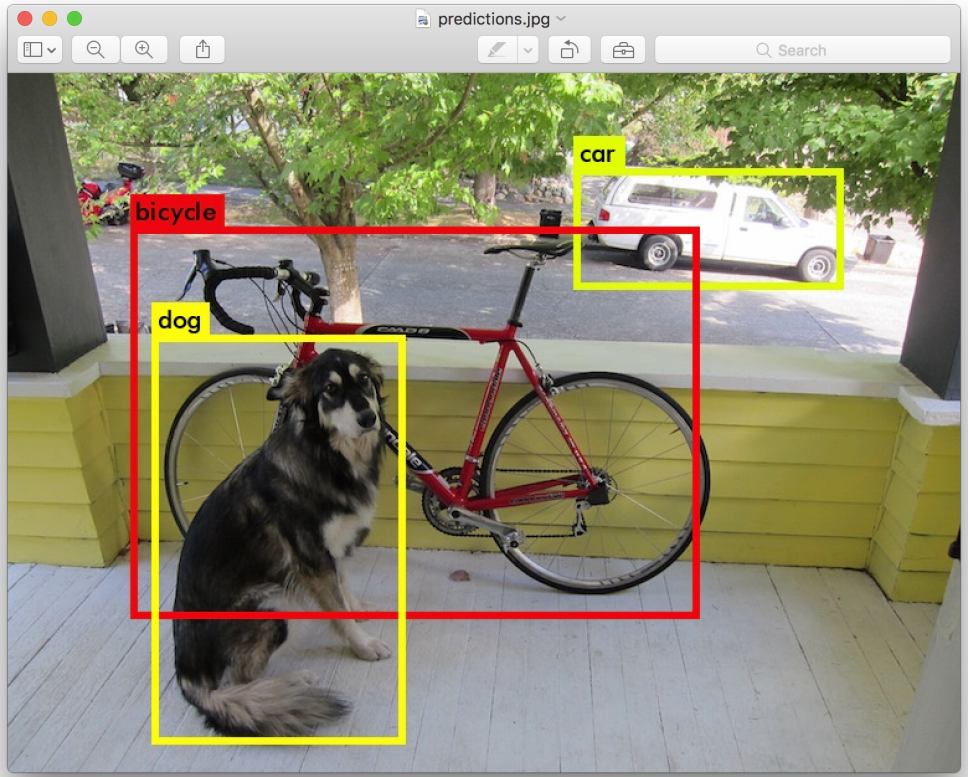


from MIT course `introtodeeplearning.com`

Main application domains of AI

Vision

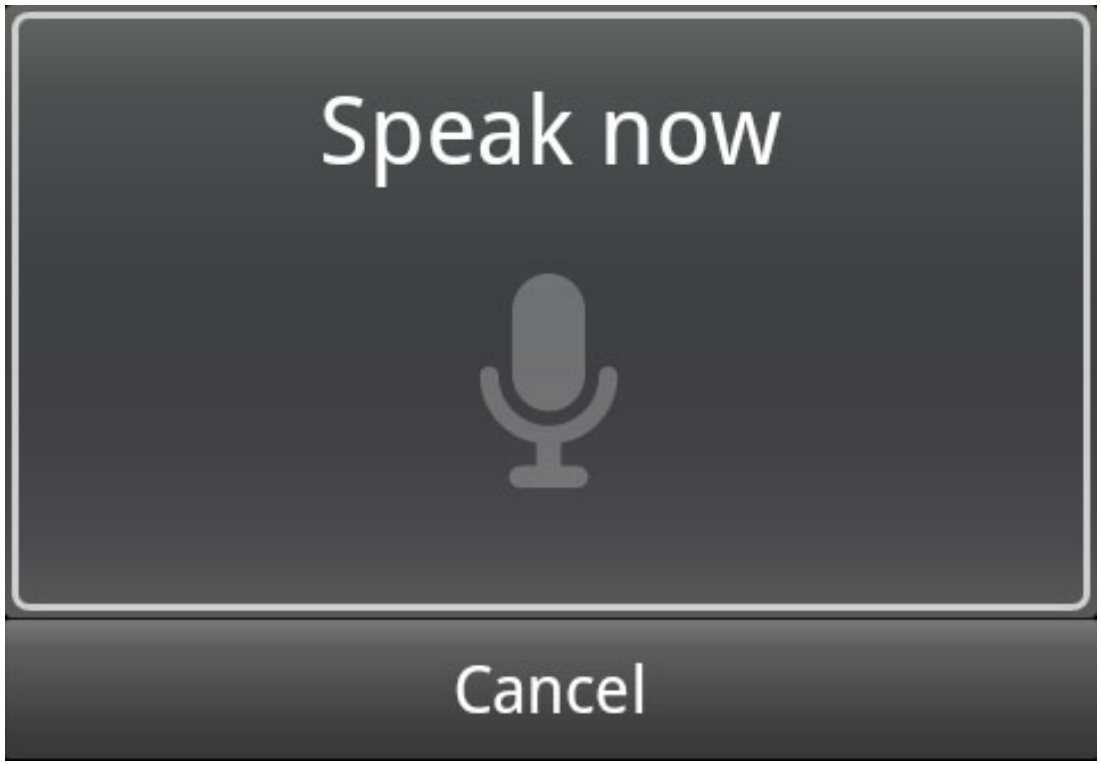
- Object/face recognition,
- Detection,
- Autonomous vehicles,
- Automatic diagnostic,
- Defects identification,
- Video applications...



Main application domains of AI

Natural Language Processing (NLP)

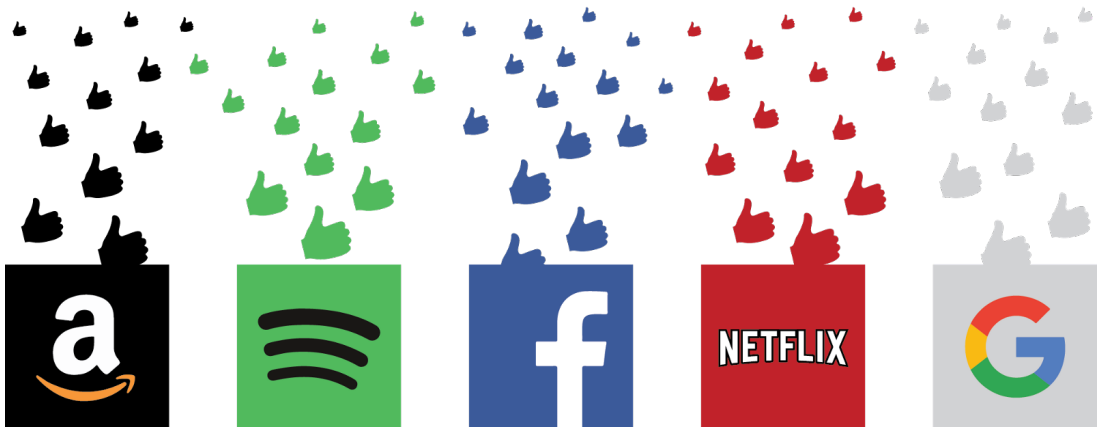
- Automatic assistant,
- Voice-to-text,
- Automatic translation,
- Automatic summarizing,
- Sentiment analysis,
- Text indexing...



Main application domains of AI

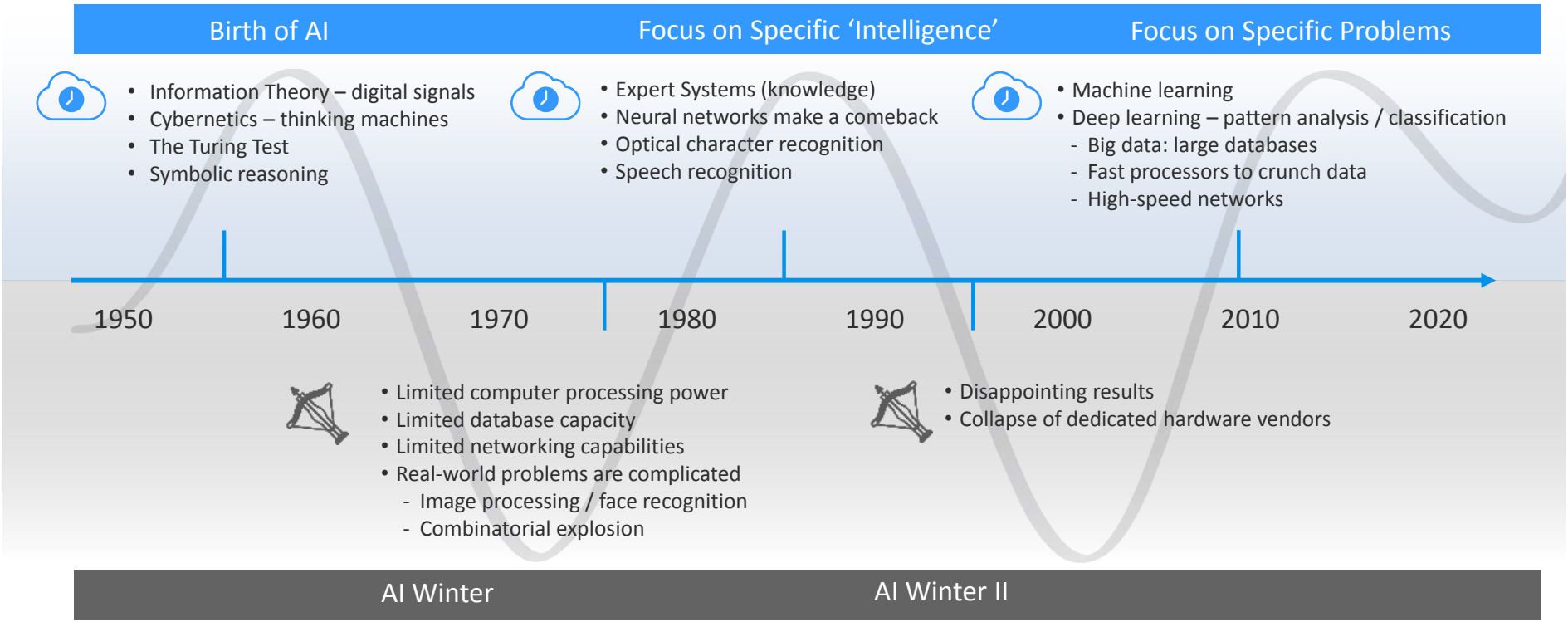
Tons of other domains...

- Medical imaging,
- Decision aid,
- Data mining,
- Visualization,
- Recommender systems,
- Market analysis...



AI Timeline

An AI Timeline



harmon.ie

Where did the revolution in AI come from?

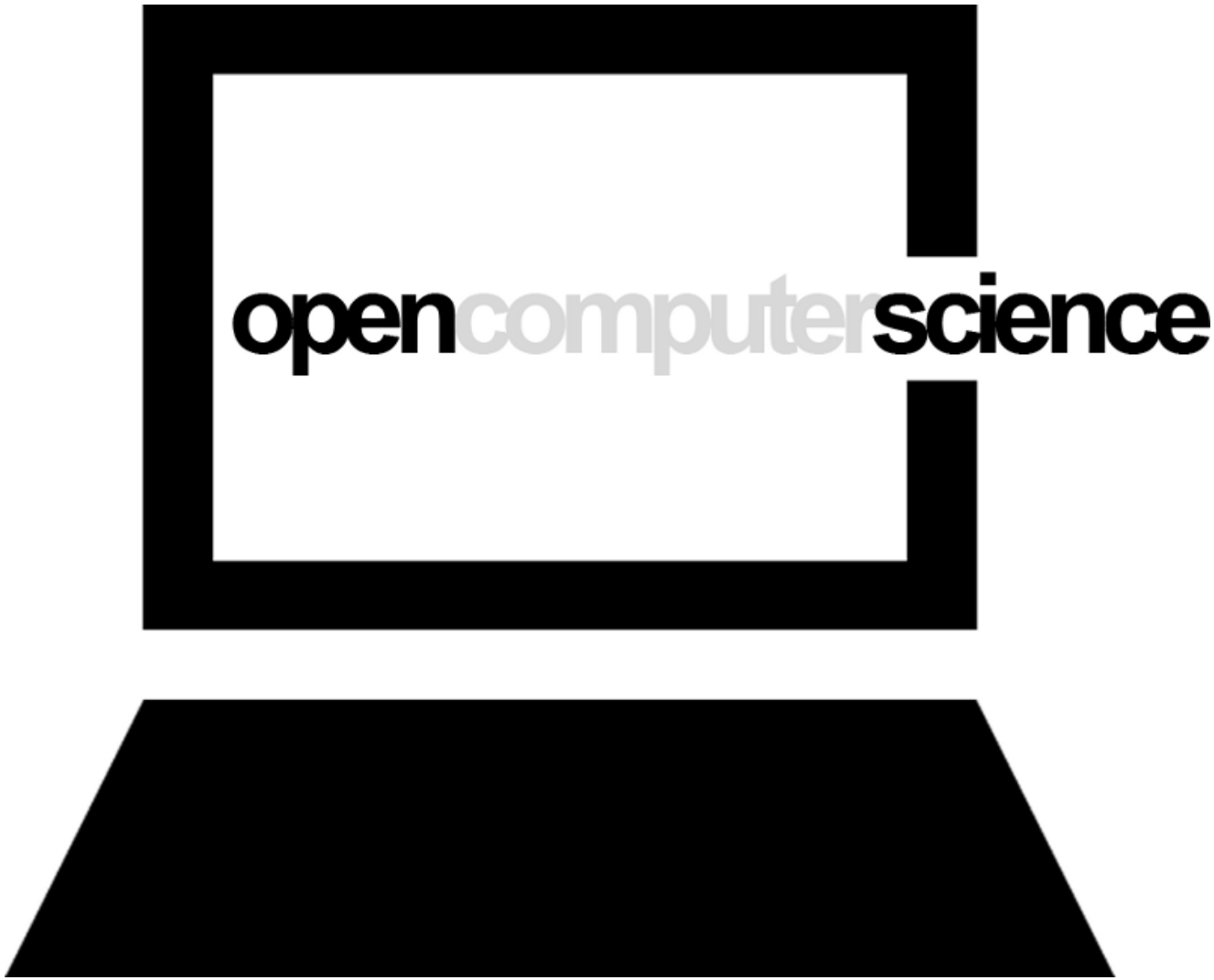
- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.

Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.

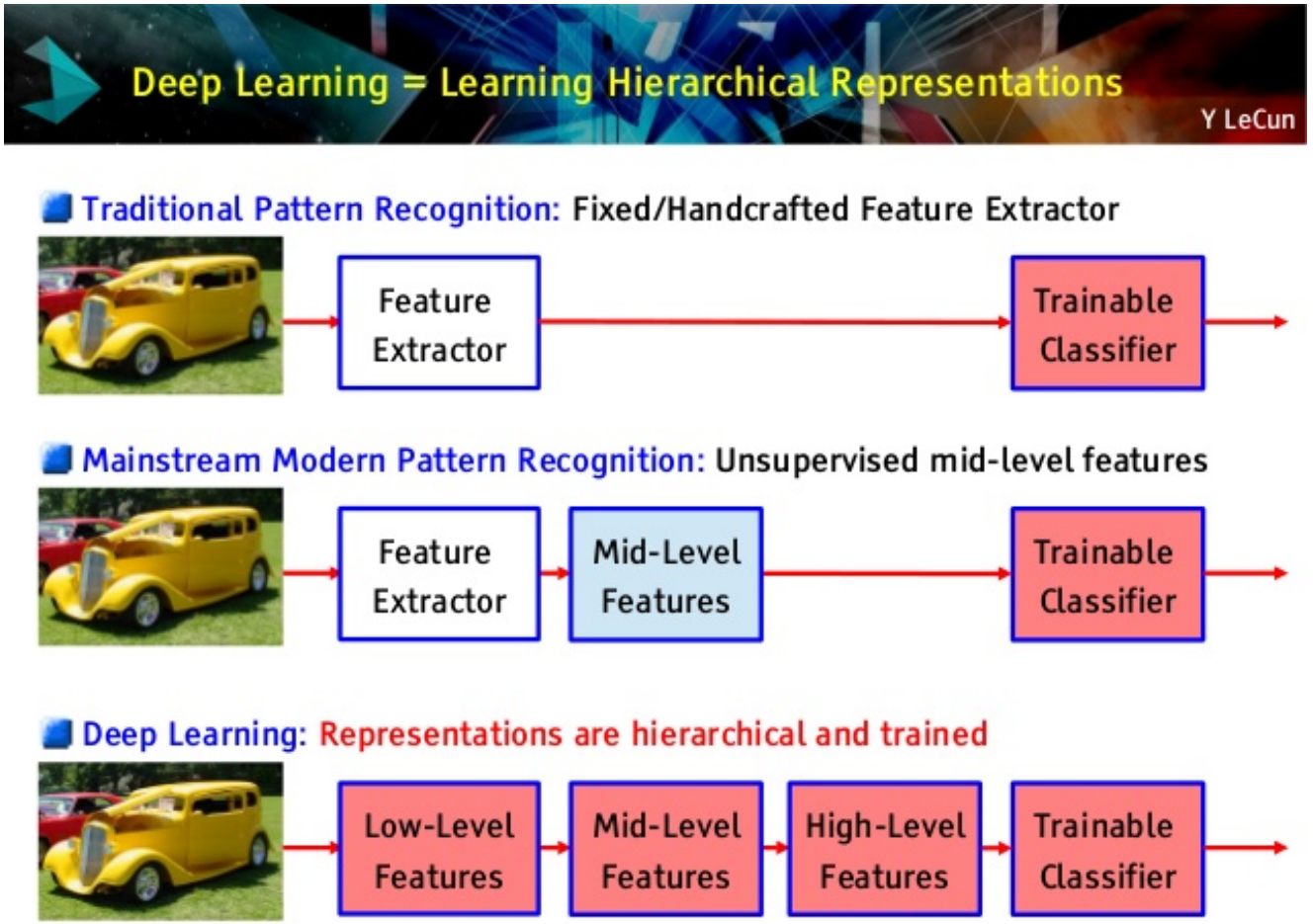
Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.



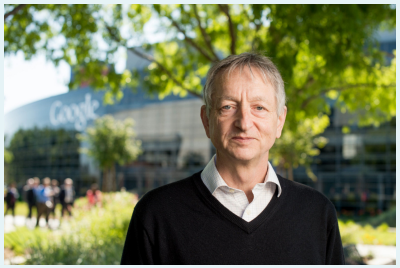
Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.



The great elders of modern AI (Turing Prize 2018)

Geoffrey Hinton



- Cognitive psychologist and computer scientist,
- Prof. at University of Toronto and works for Google,
- Known for back-propagation and Boltzmann machines.

Yoshua Bengio

- Computer scientist,
- Prof. at Université de Montréal and head of MILA,
- Known for his work on deep learning.

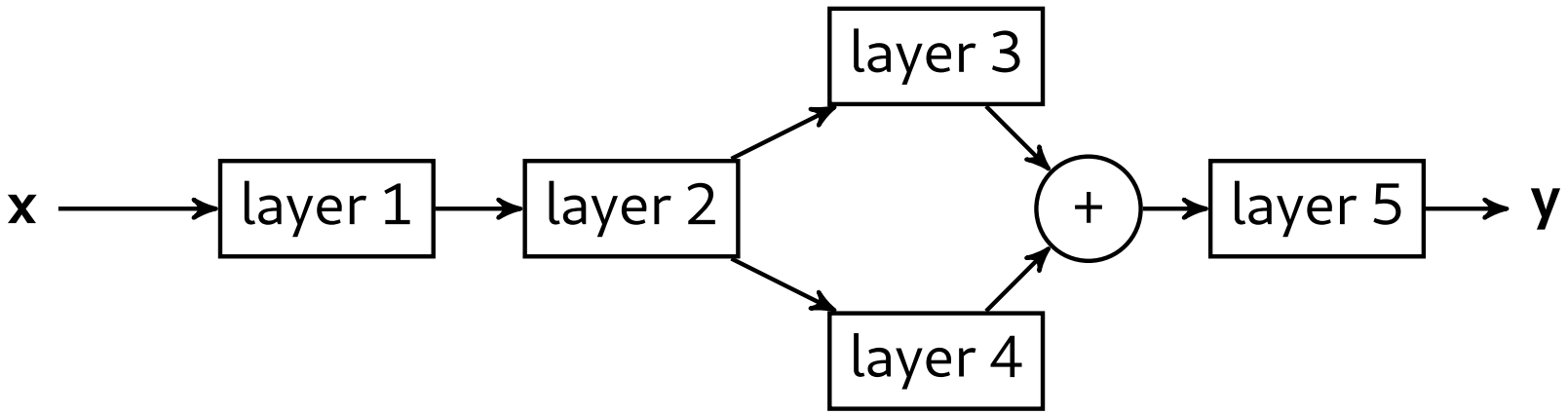
Yann le Cun

- Computer scientist,
- Prof. at New York University then he joins FAIR,
- Known for his work on back-propagation and CNNs.

Deep learning

Main idea

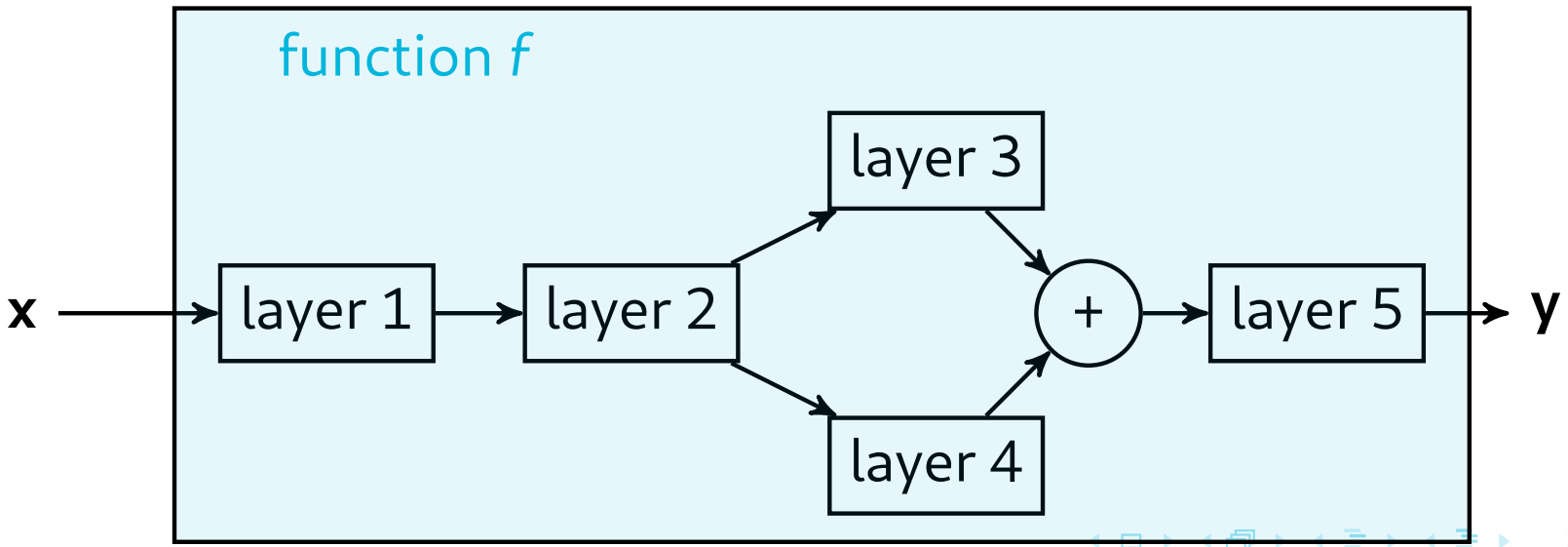
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

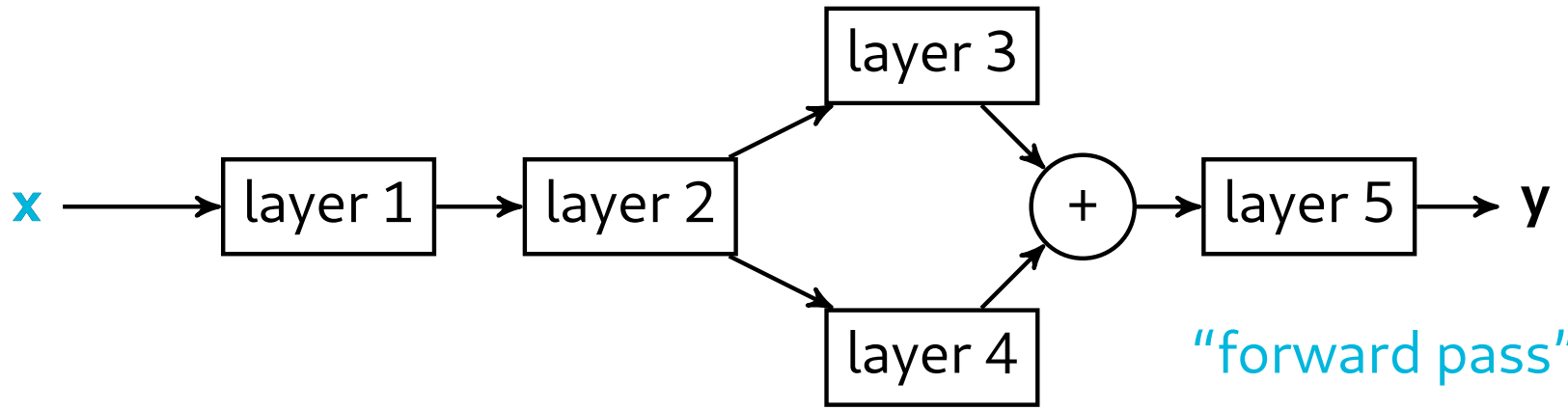
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

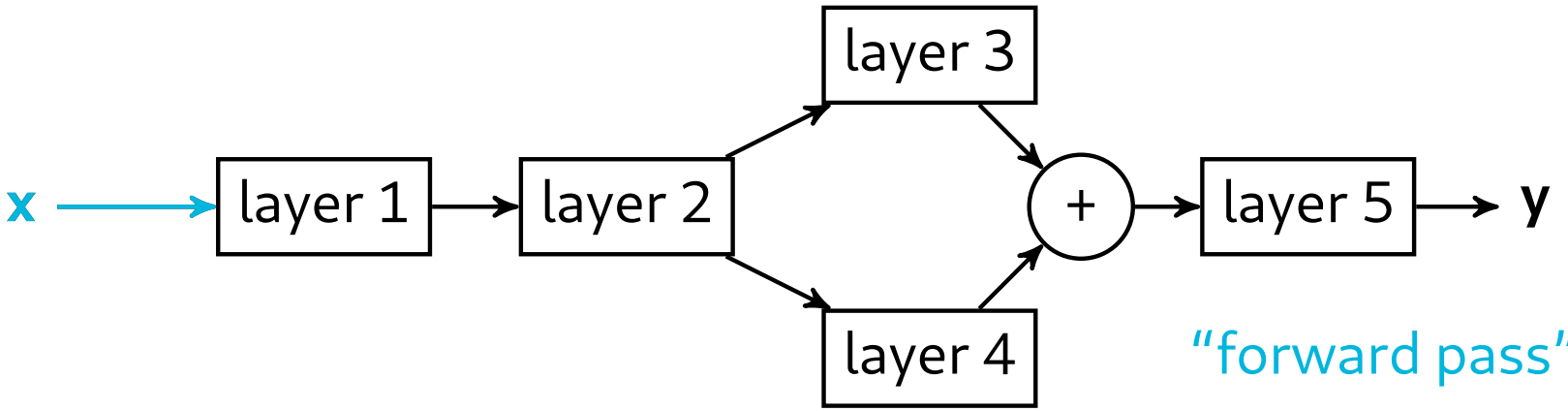
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

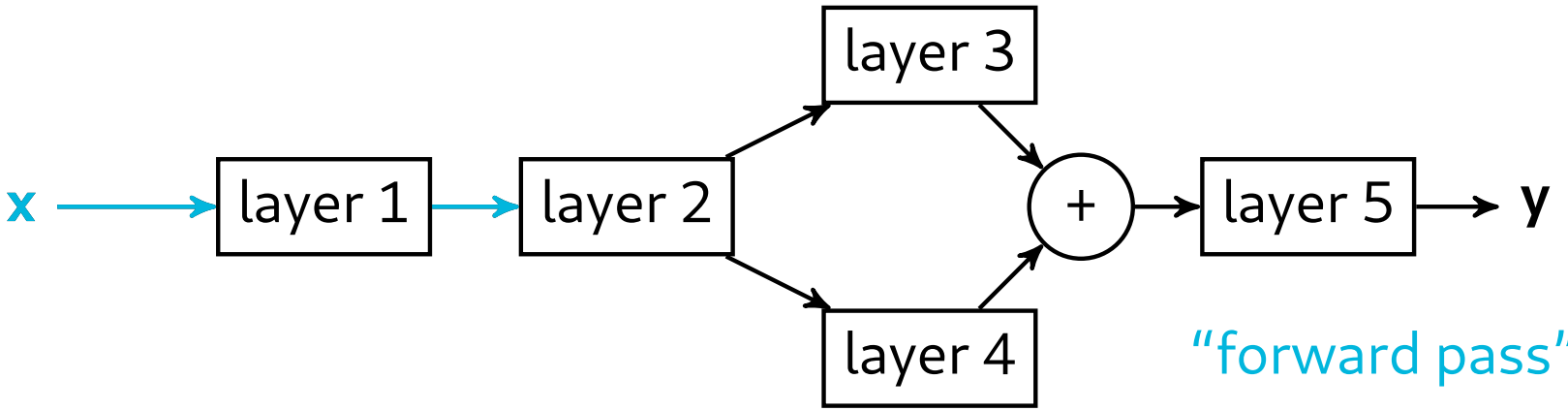
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

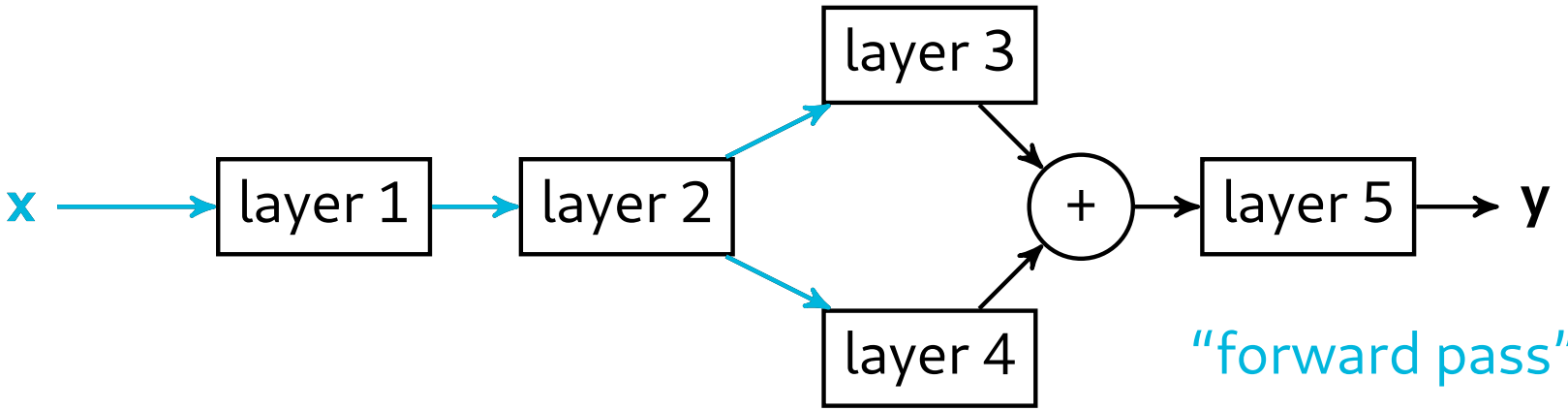
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

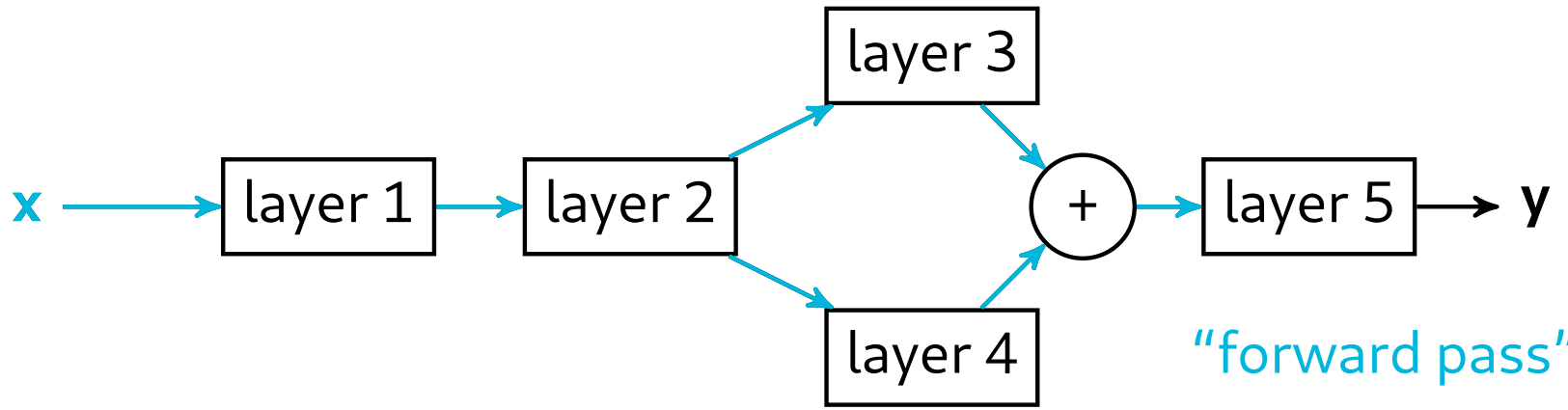
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

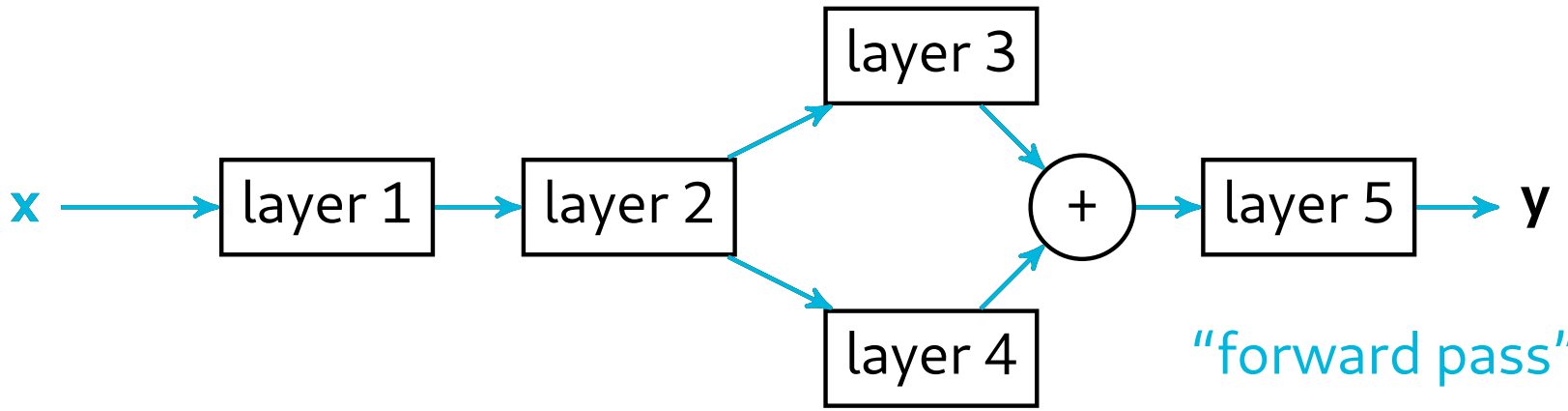
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

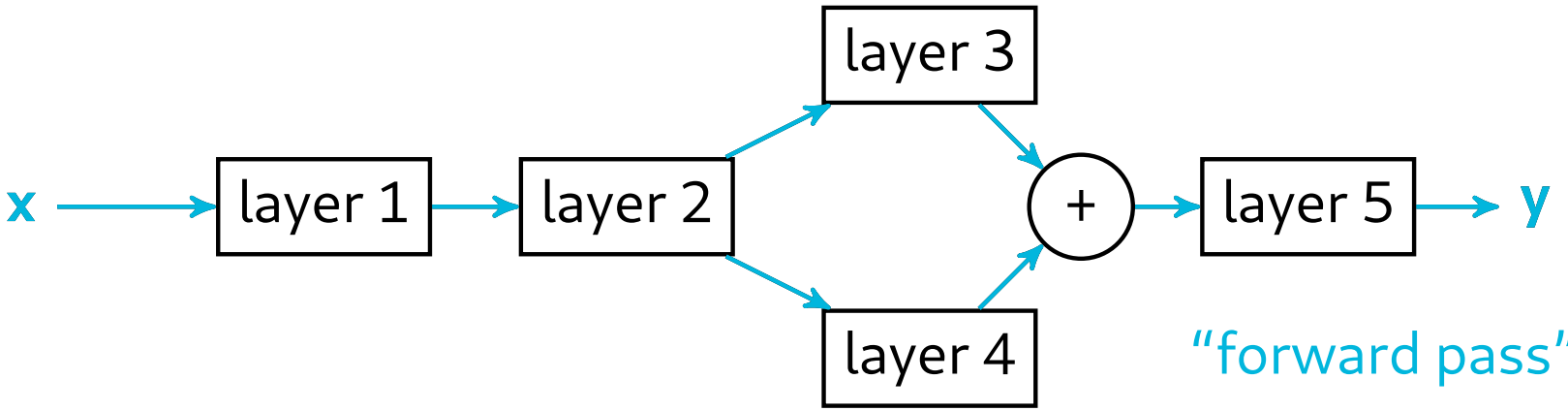
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

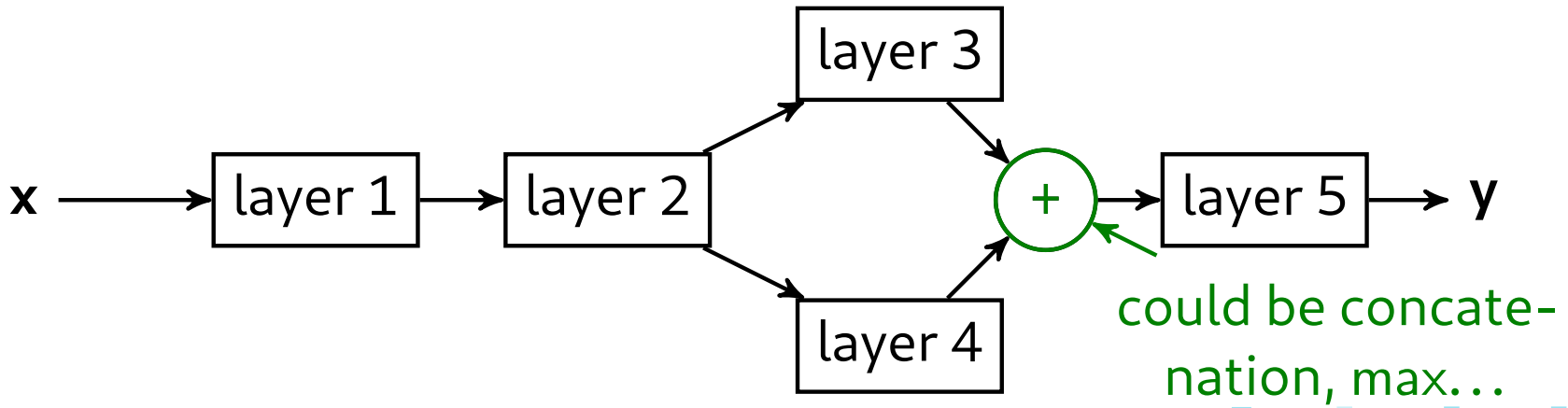
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

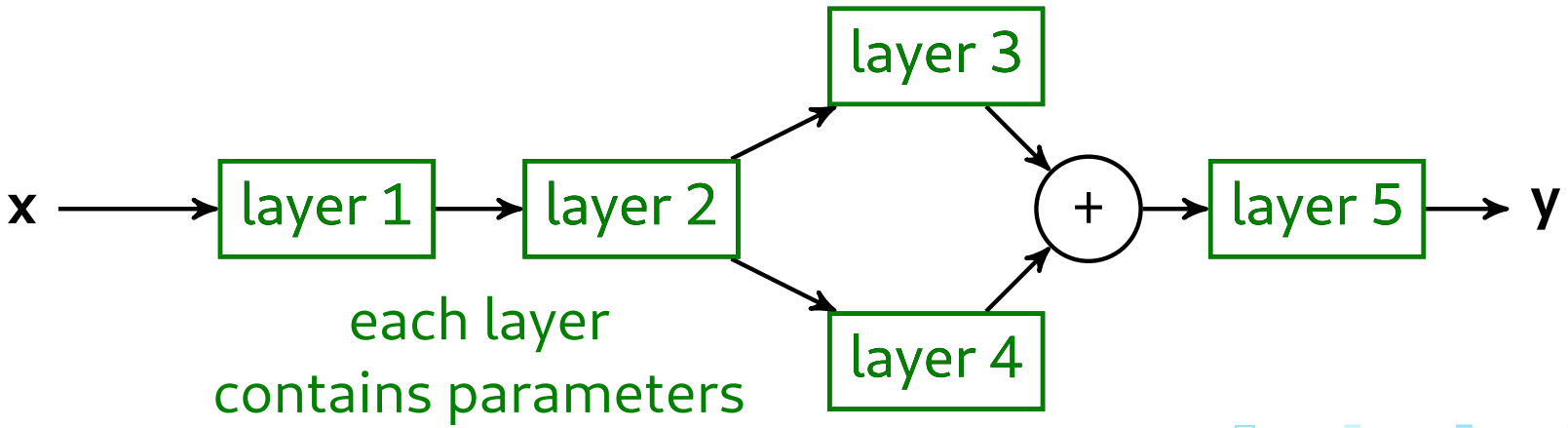
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

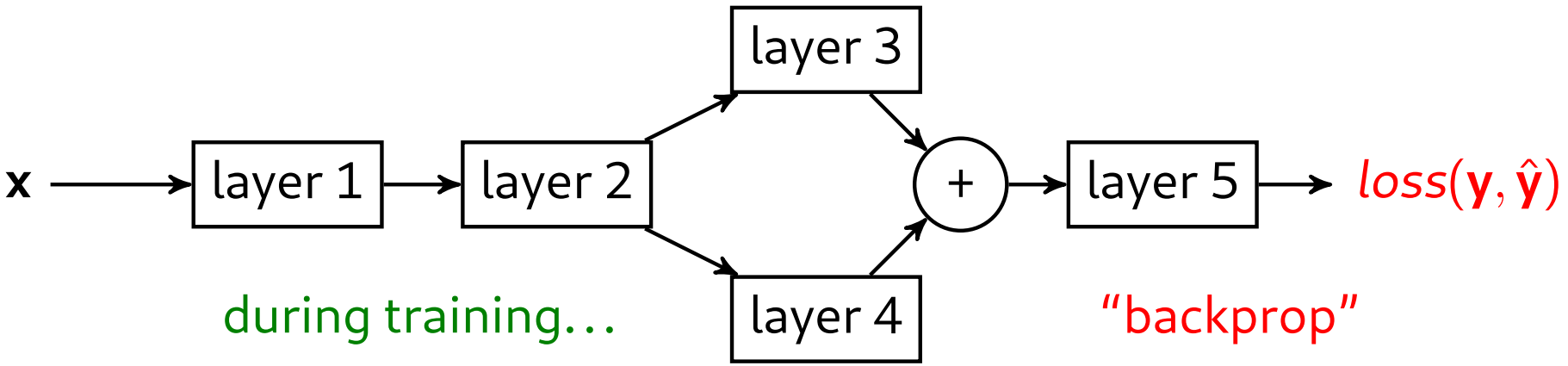
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

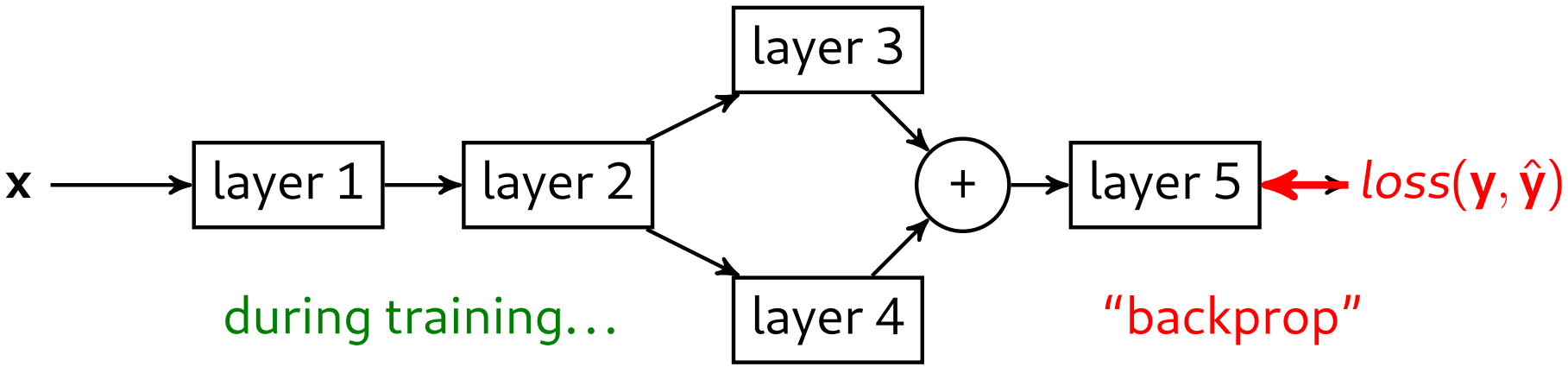
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

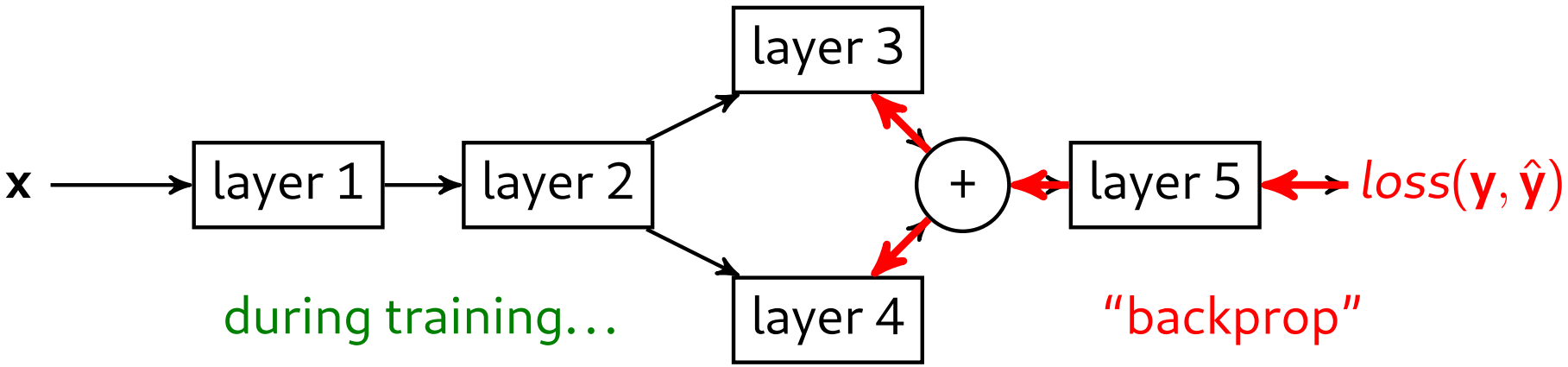
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

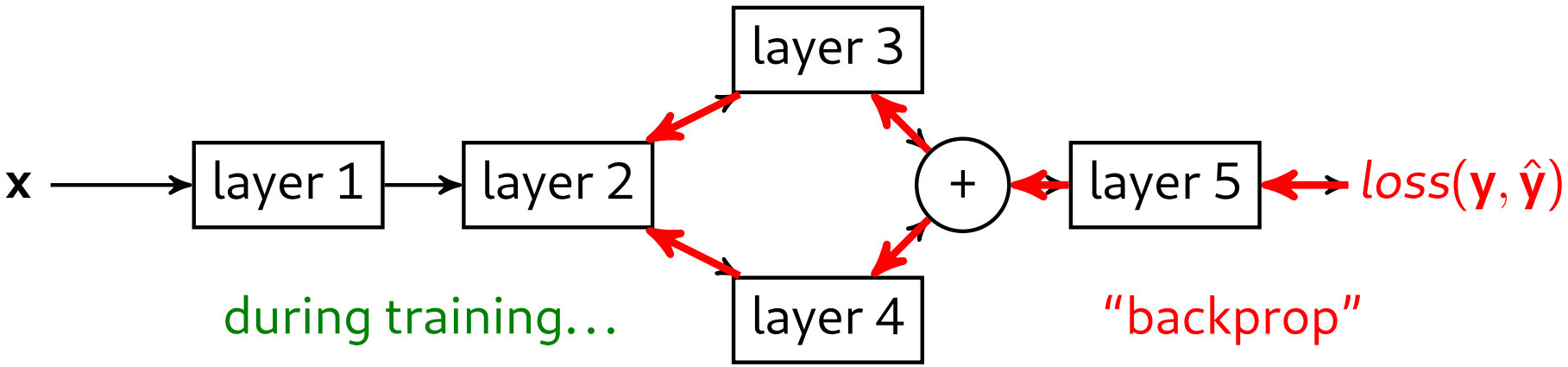
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

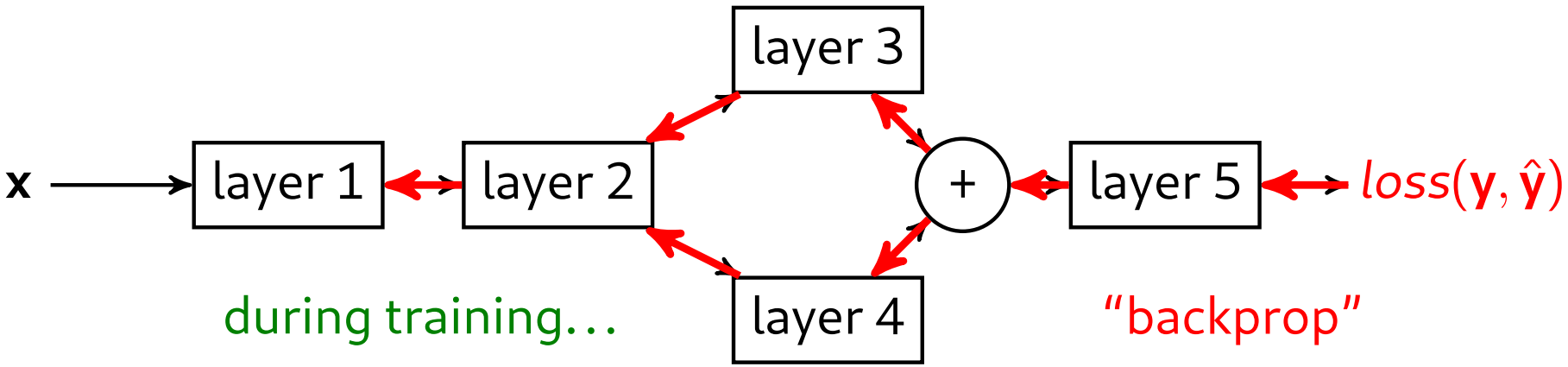
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

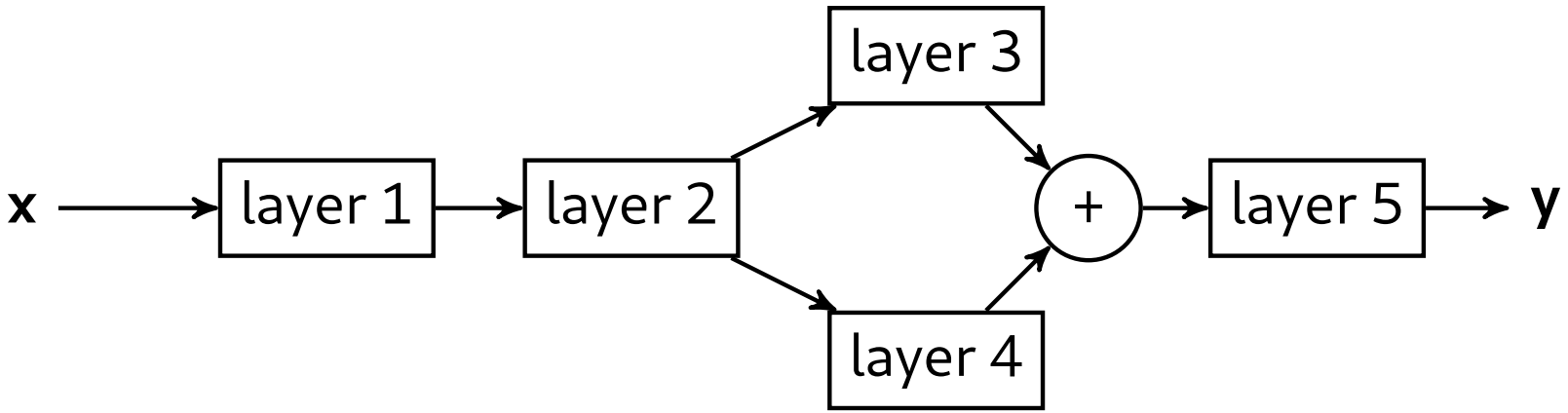
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Number of layers, choice of the architecture are **hyperparameters**

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

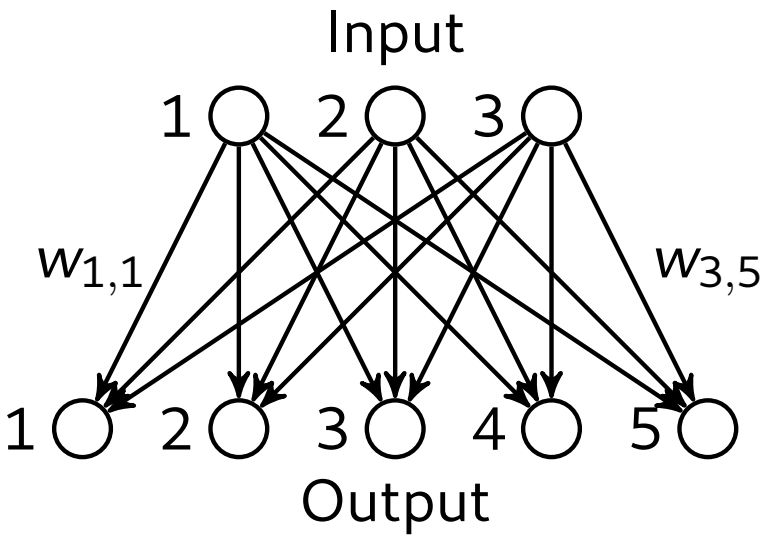
- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Fully connected layer



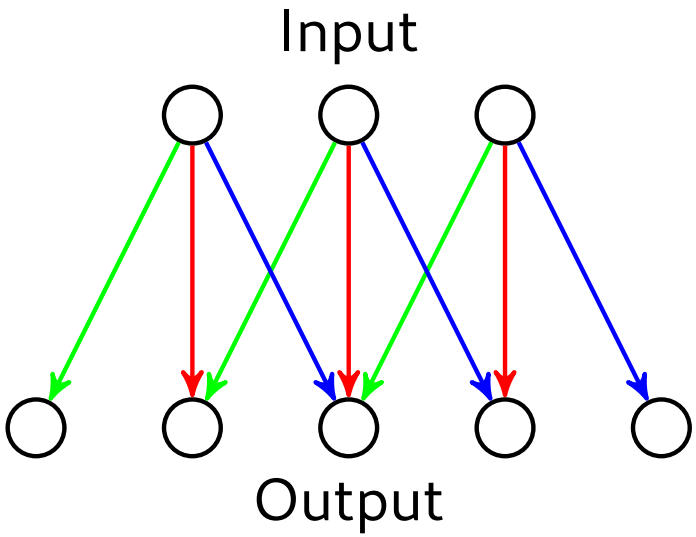
$$\begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} \end{pmatrix}$$

Some additional details

Layers

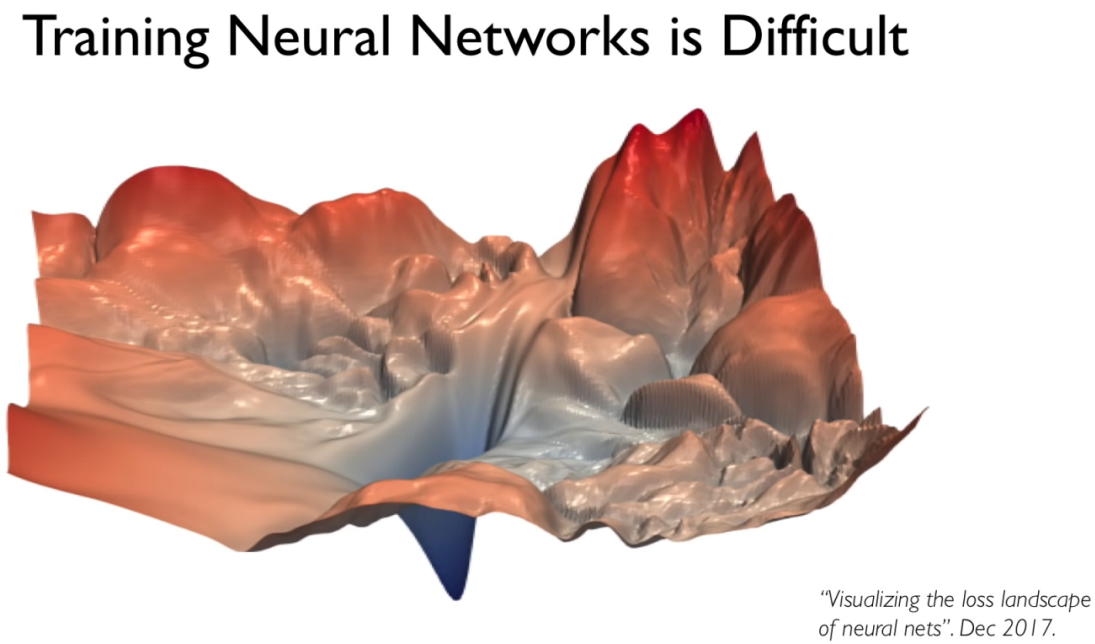
- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Convolutional layer



$$\begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{pmatrix}$$

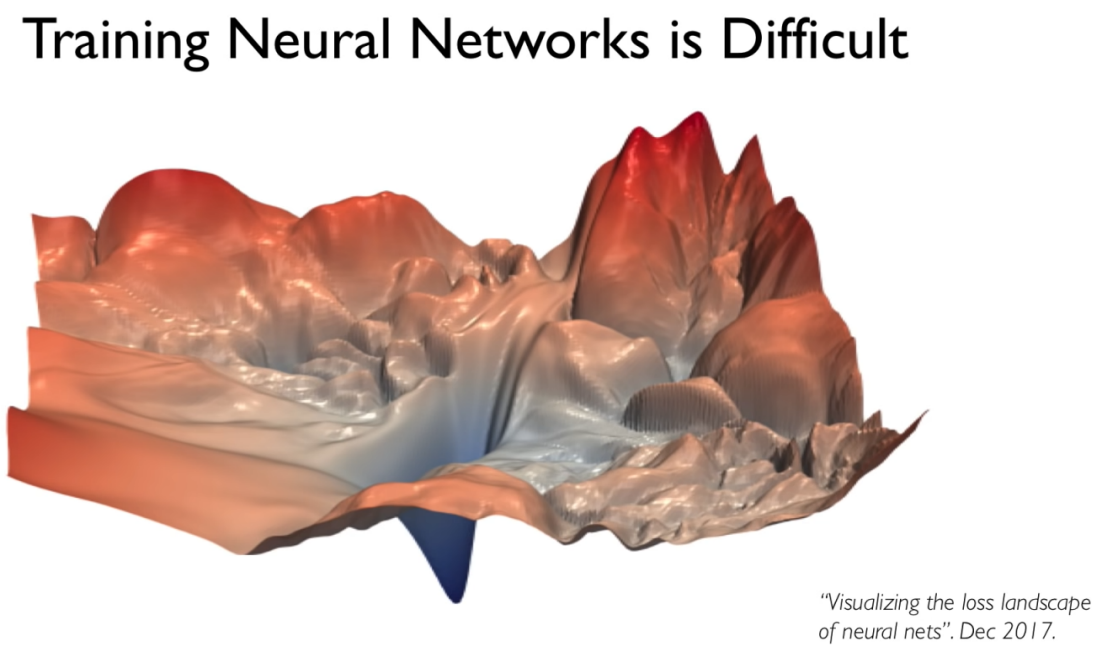
Some additional details



Optimization with Differentiable Algorithmic

- Learning rate η : $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Variants of the **Stochastic Gradient Descent (SGD)** algorithm are used:
  - Use of **moments**,
  - Use of **regularizers**.

Some additional details



"Visualizing the loss landscape of neural nets". Dec 2017.

Batches

- To accelerate computations, inputs are often treated **concurrently** using small **batches**.

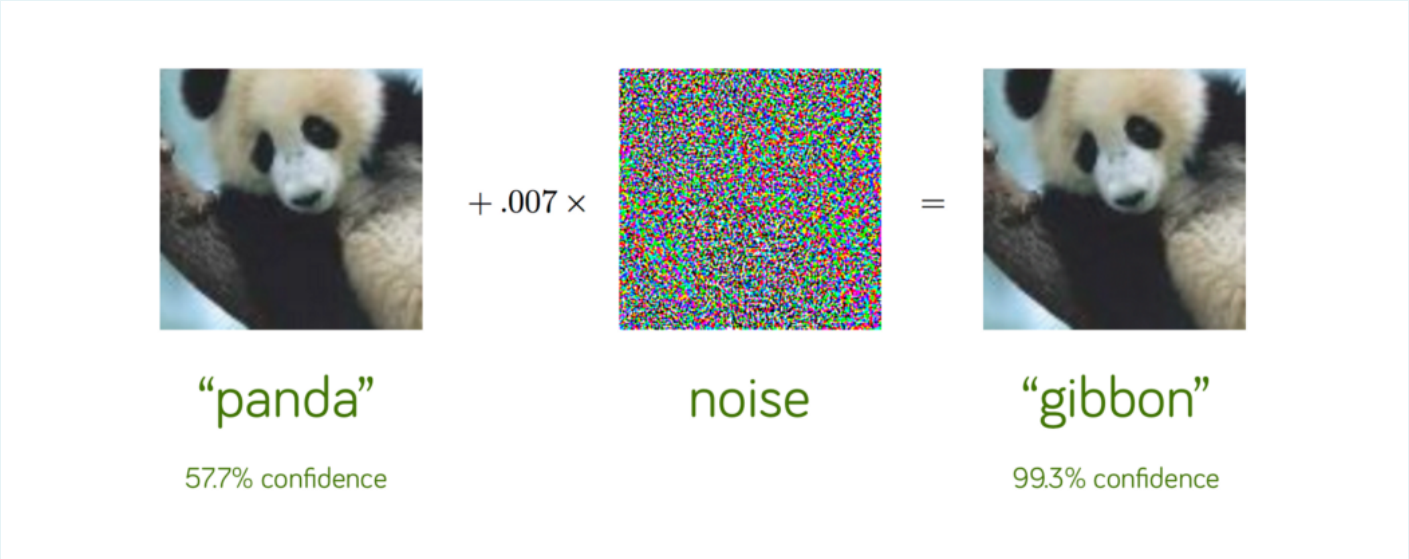
Some key open challenges (core AI research)

Learning from few examples / few shots / few labels

"How to grow a mind: statistics, structure, and abstraction", Science, 2011.

Some key open challenges (core AI research)

Learning what should be learned (robustness / adversarial attacks)

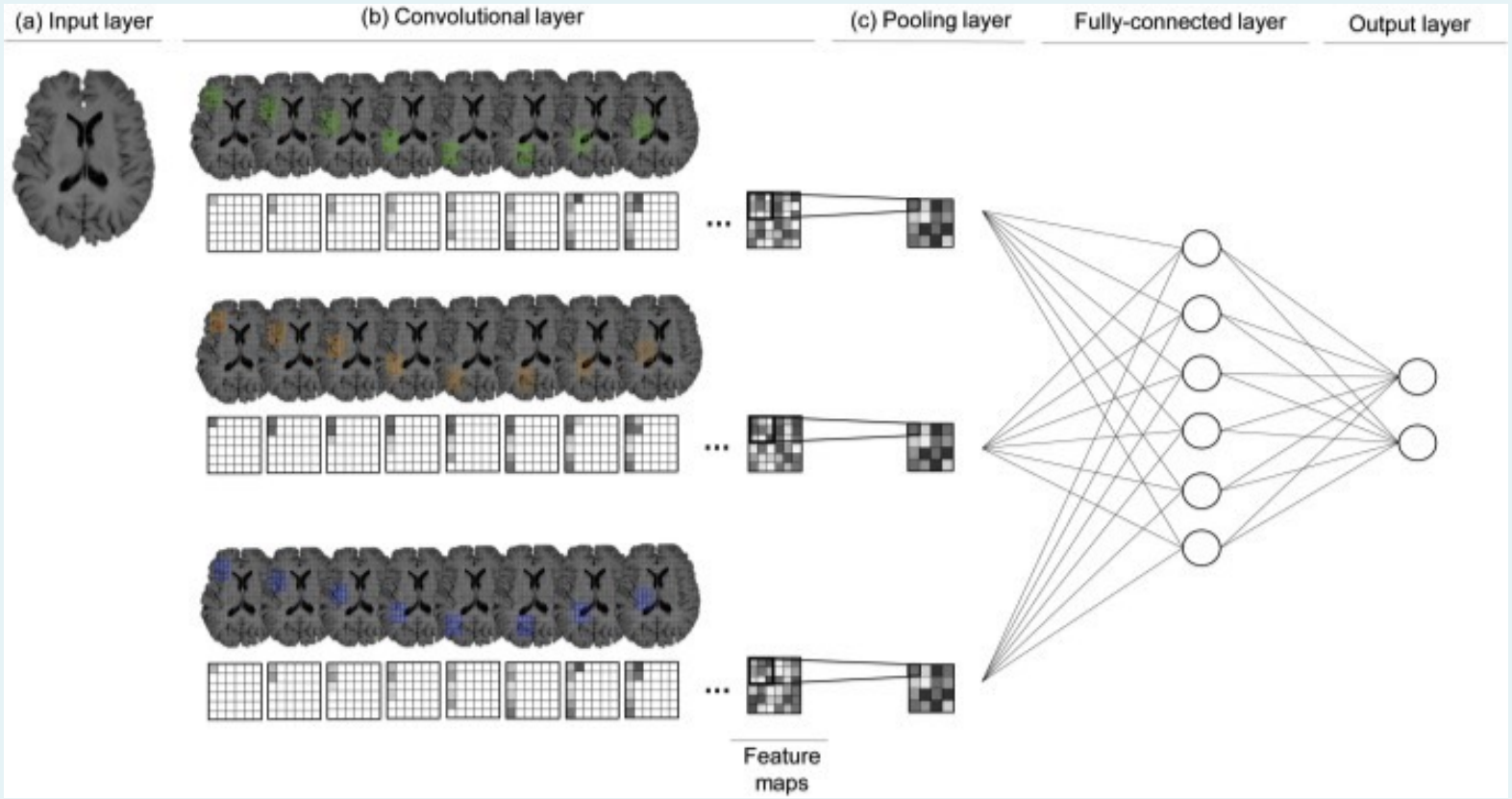


Random noise added to input images can dramatically change the result.

“Intriguing properties of neural networks”, Arxiv research report, 2013.

Some key open challenges (core AI research)

Interpretability



A trained model might be very accurate, but how does it take its decision ?

"Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications", Vieira et al. 2017.

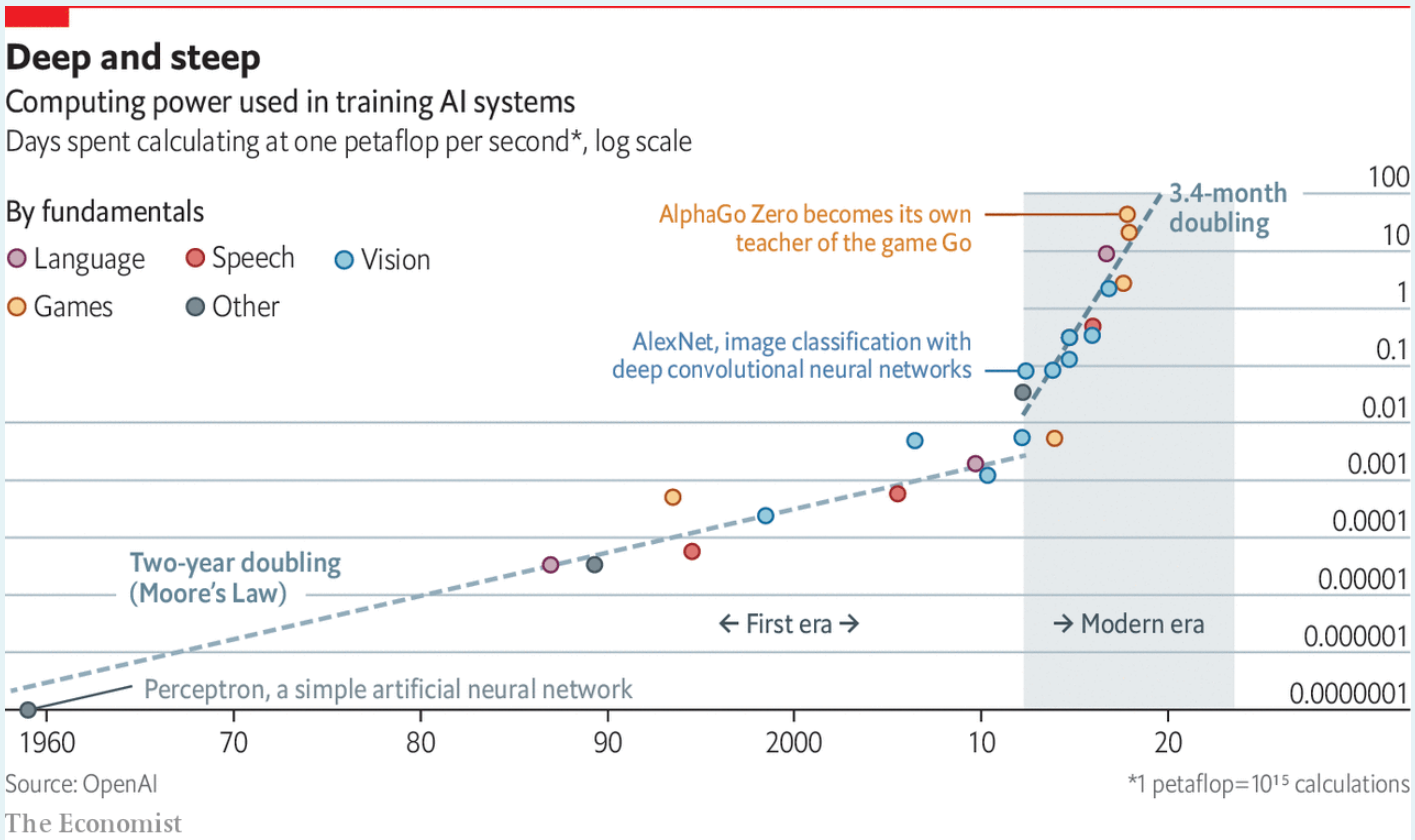
Some key open challenges (core AI research)

Incremental (or continual) learning

Adding new classes of object one by one without forgetting previous knowledge.

Some key open challenges (core AI research)

Computational and memory footprints



Training a large algorithm: thousands to millions of parameters using Gigabytes of data.

Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...

Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...

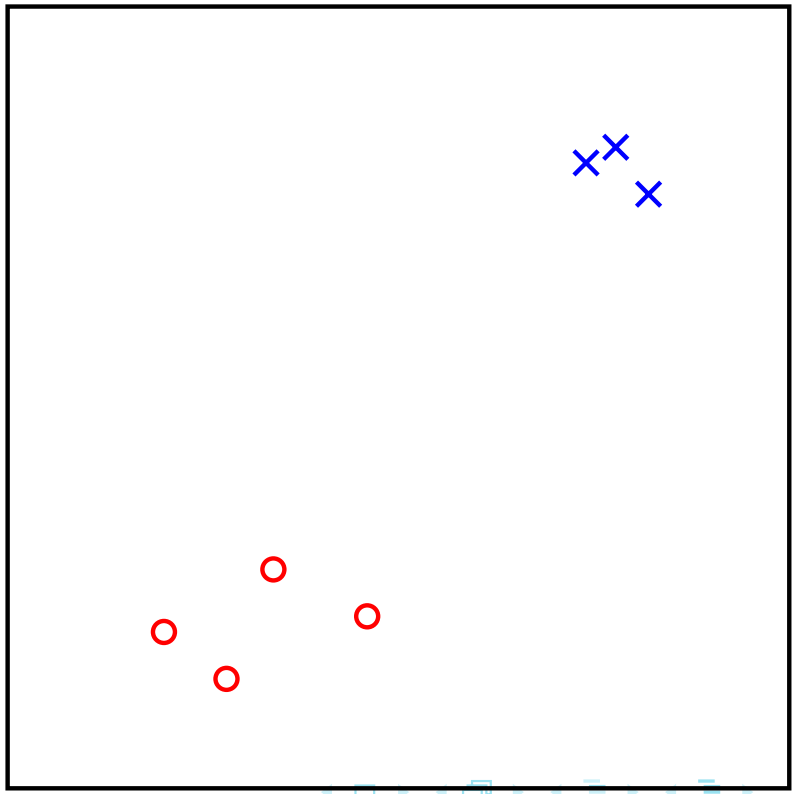
Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



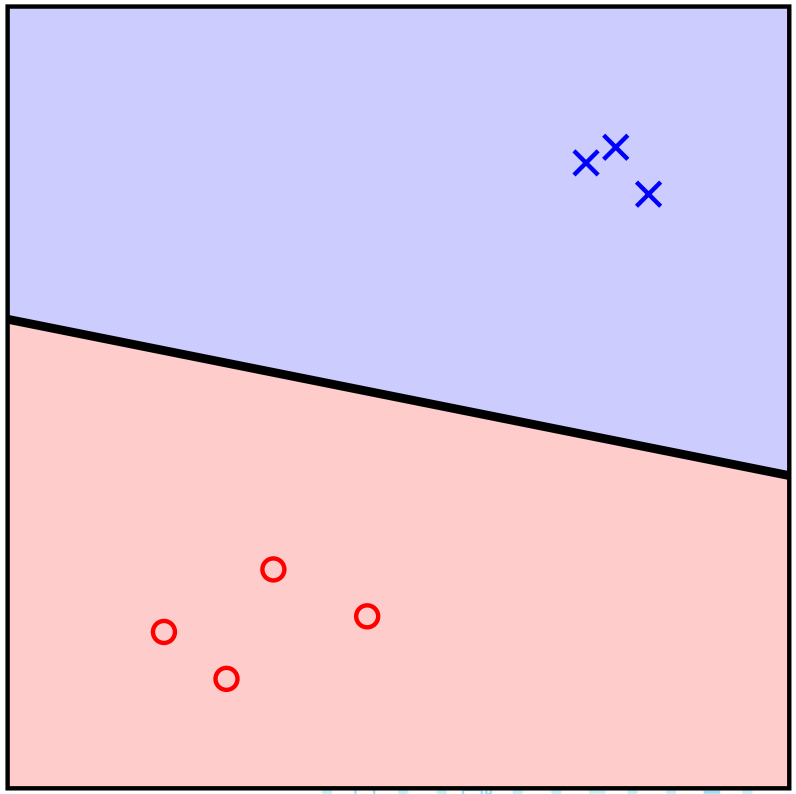
Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



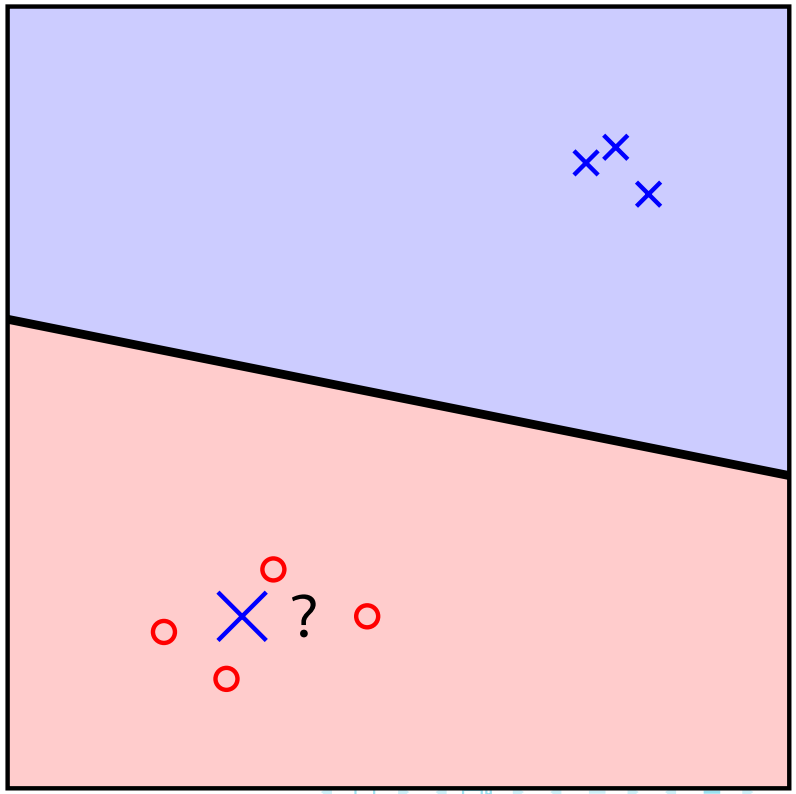
Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



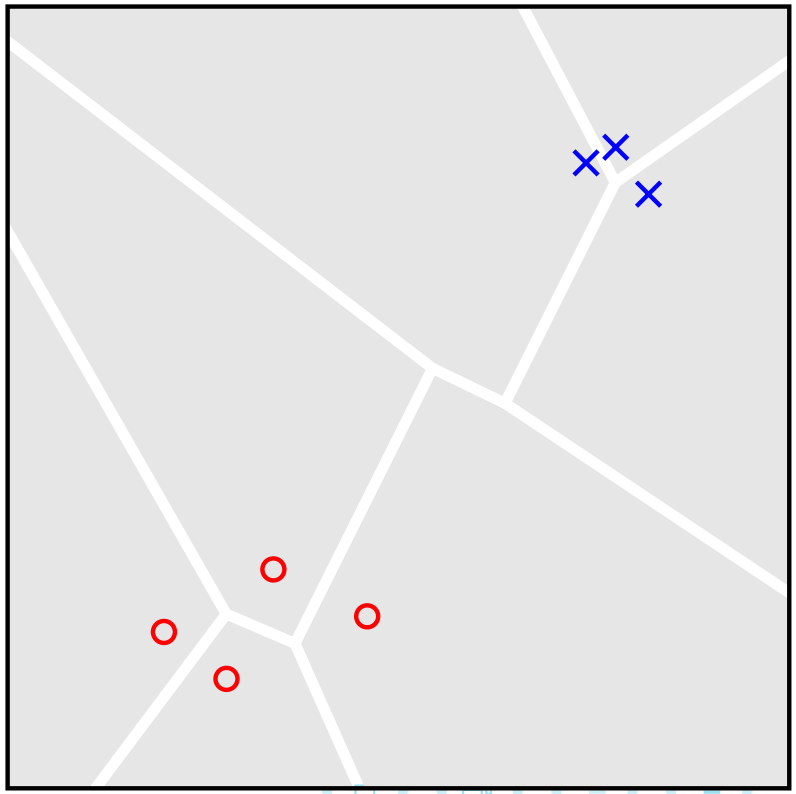
Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

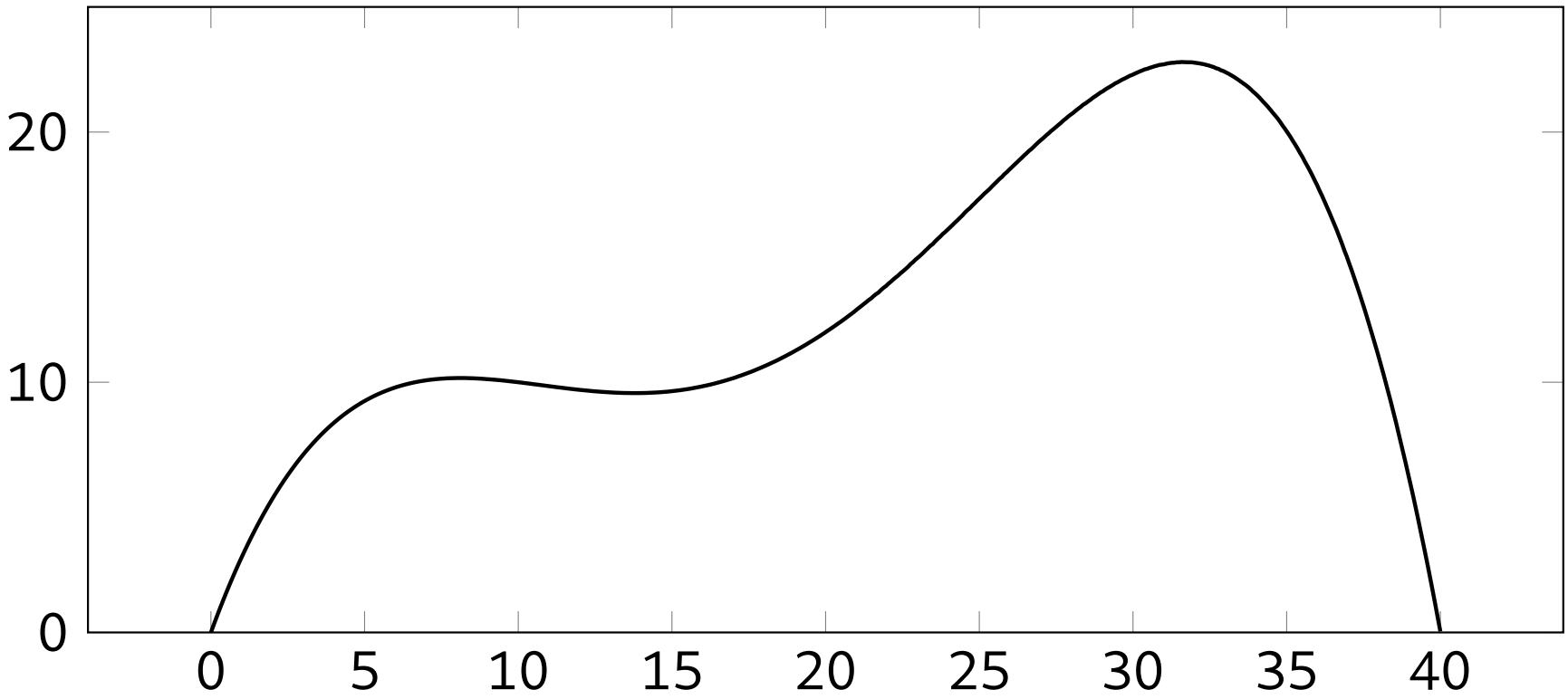
- **Regression** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



Overfitting

Bias/variance trade-off

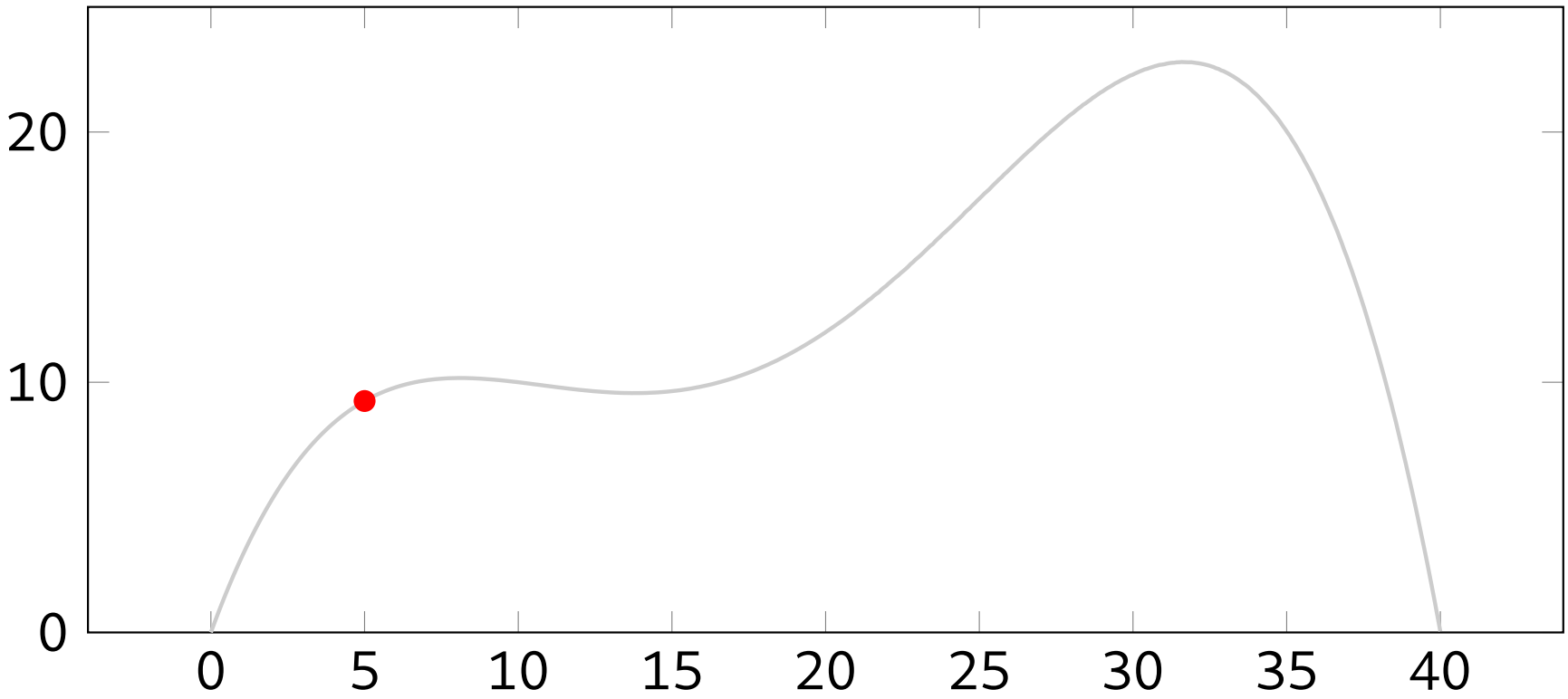
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

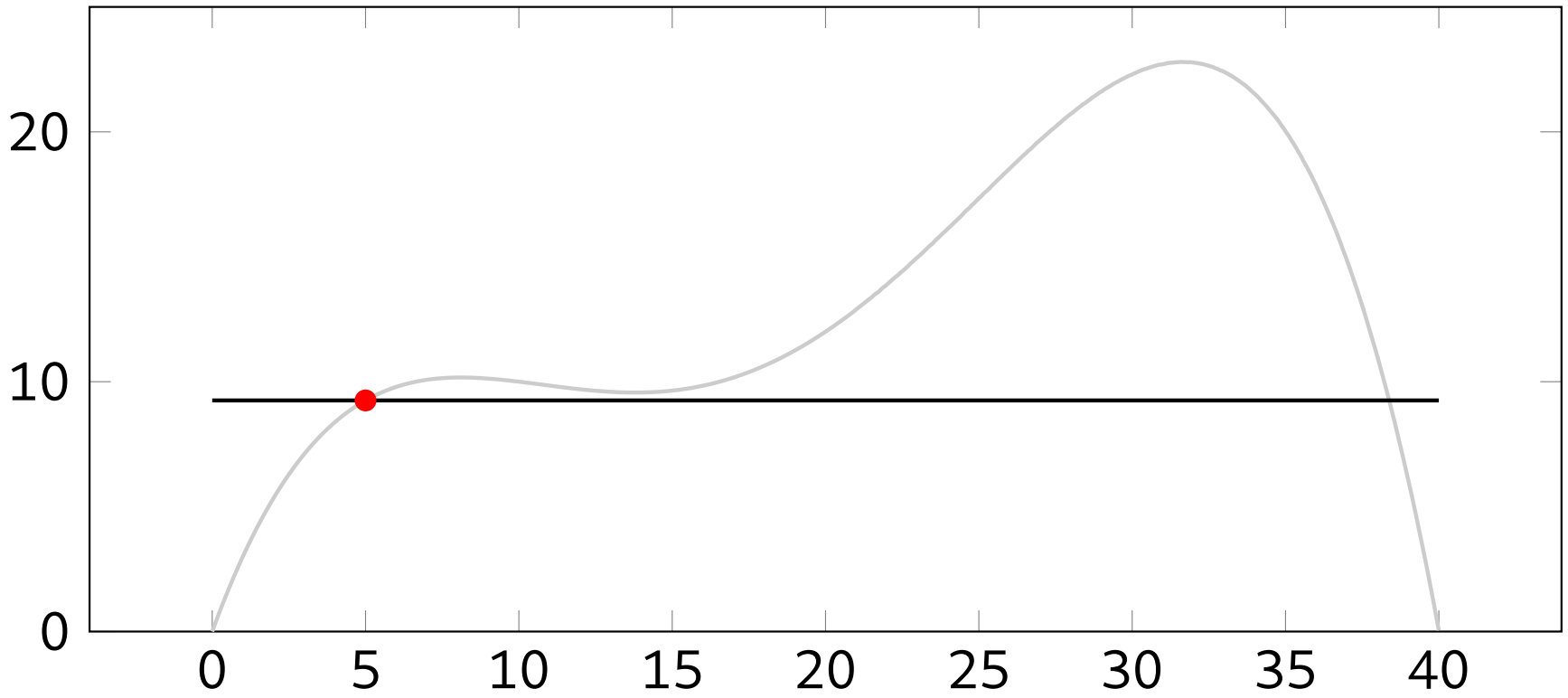
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

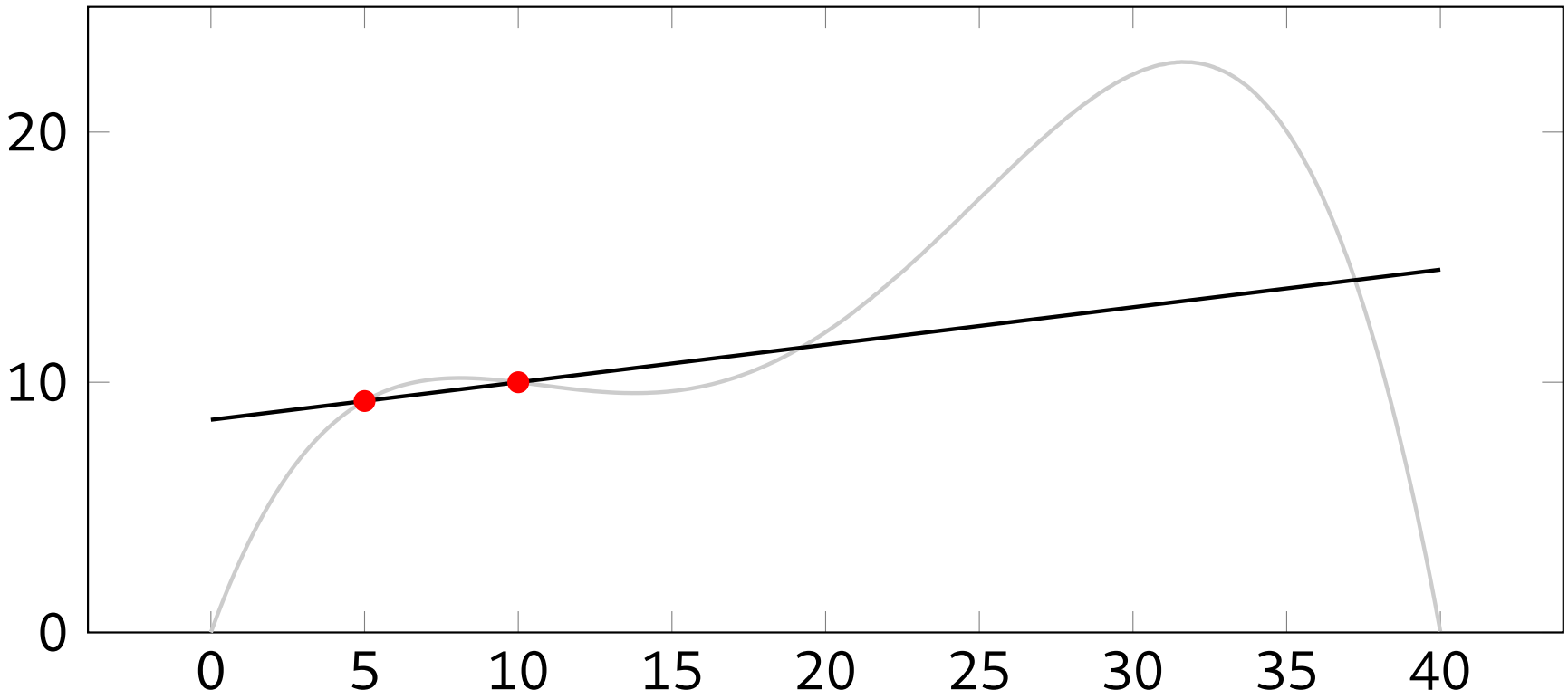
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

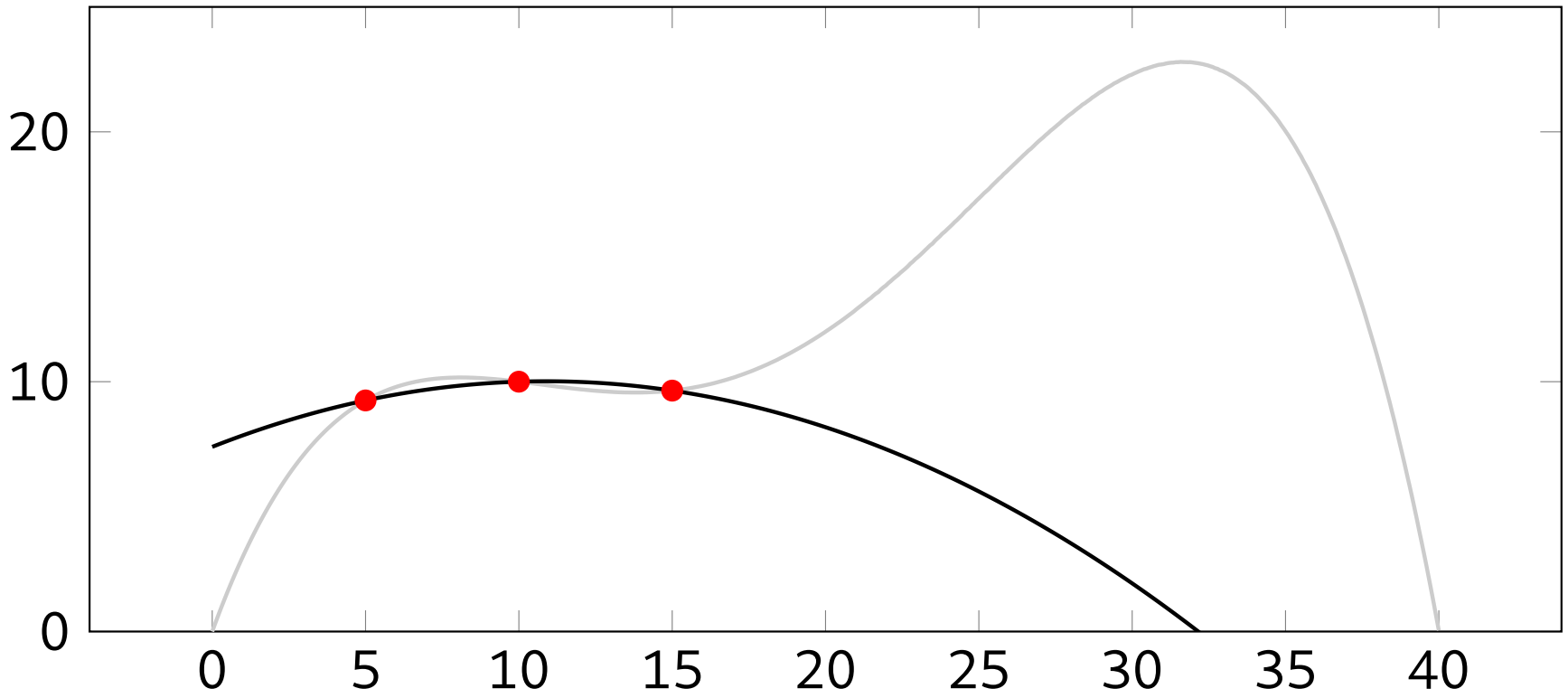
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

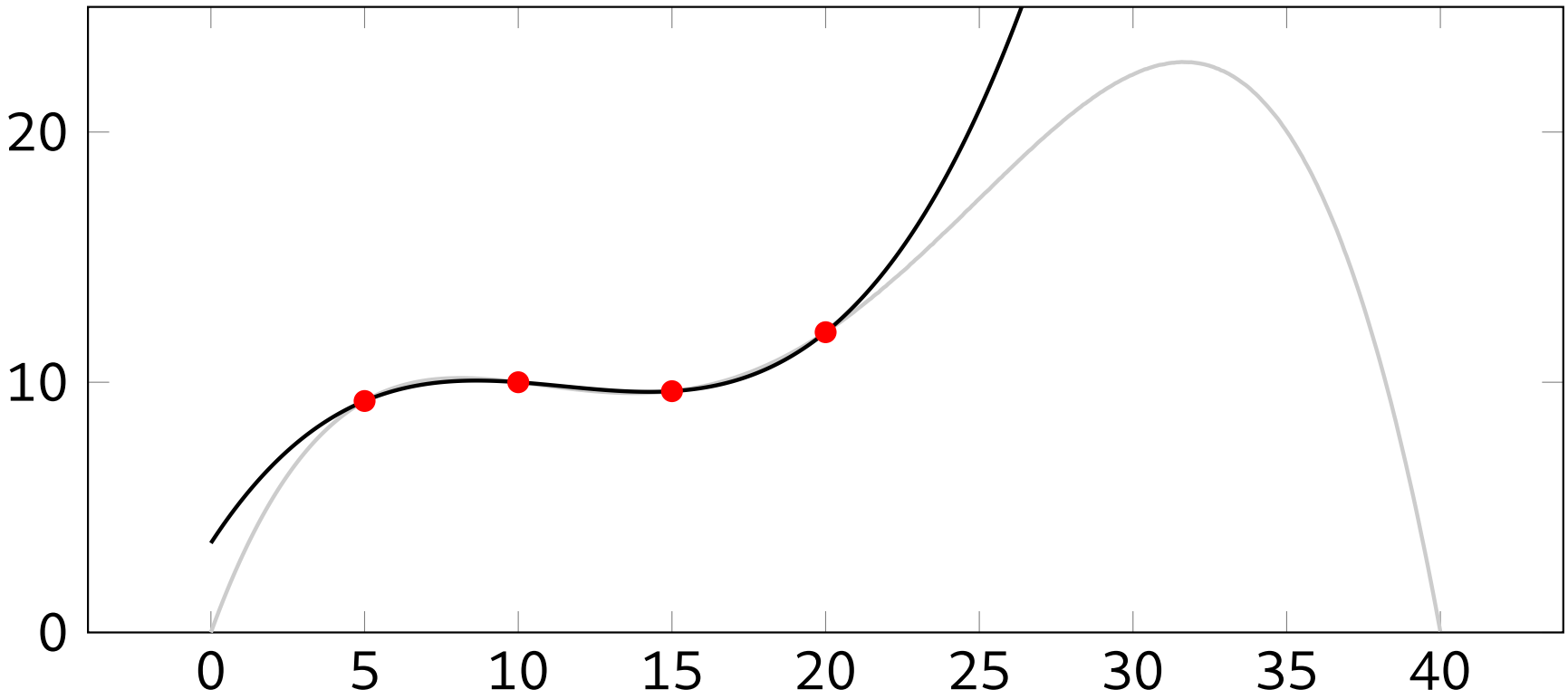
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

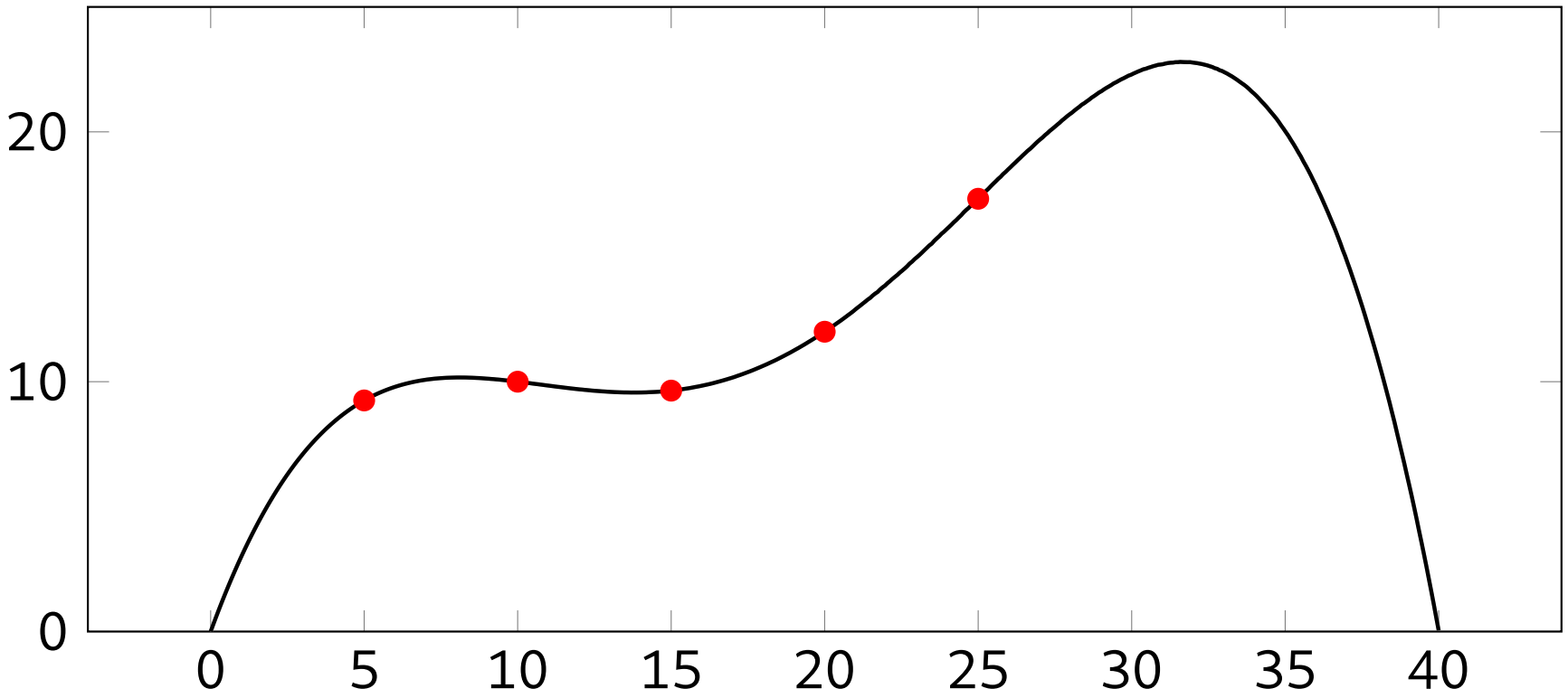
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

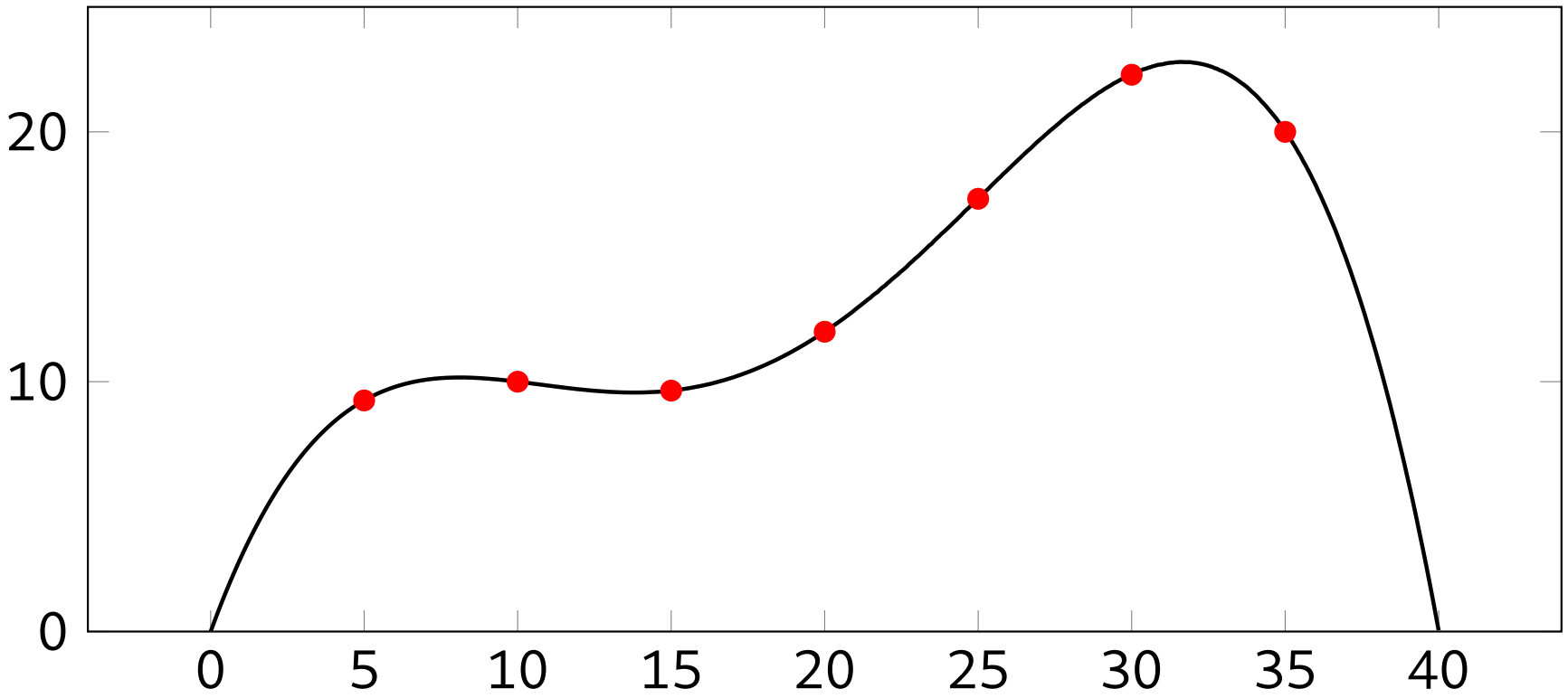
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

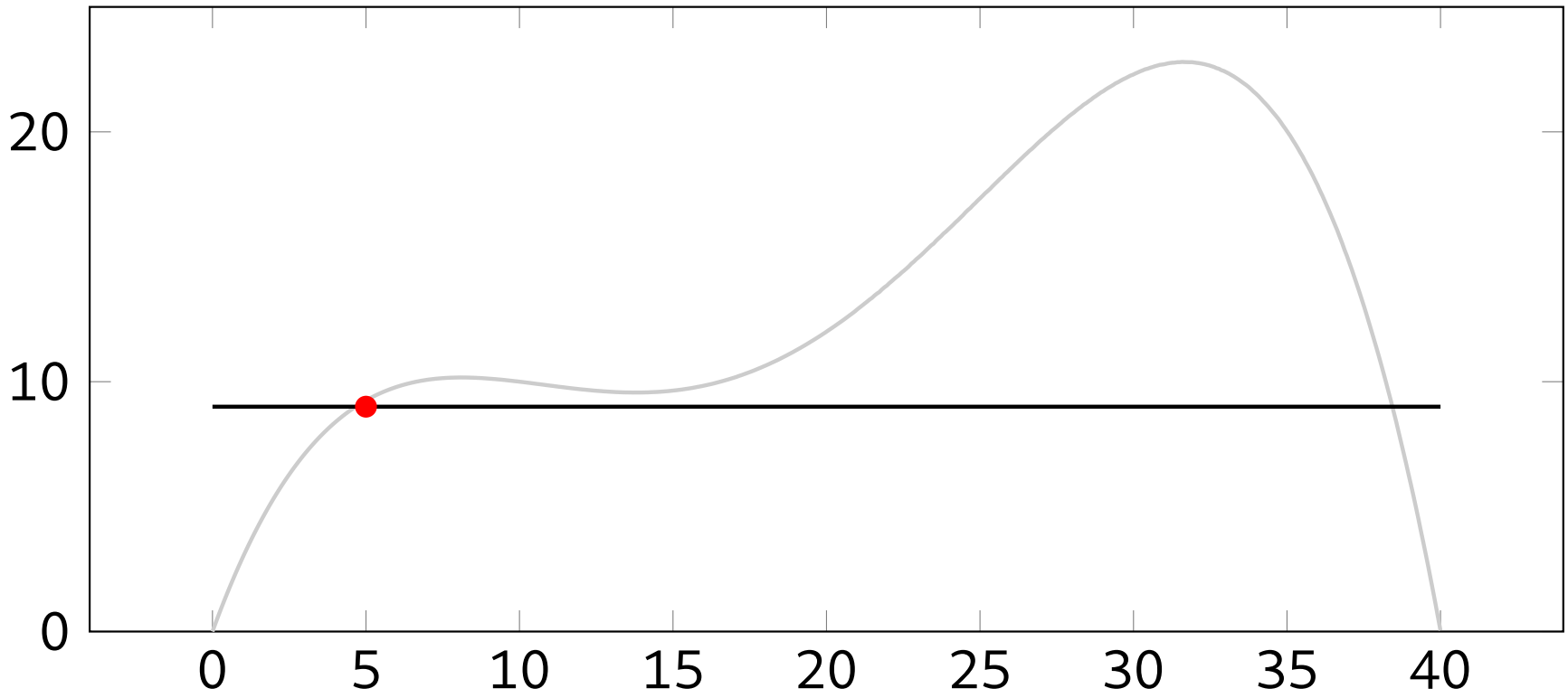
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

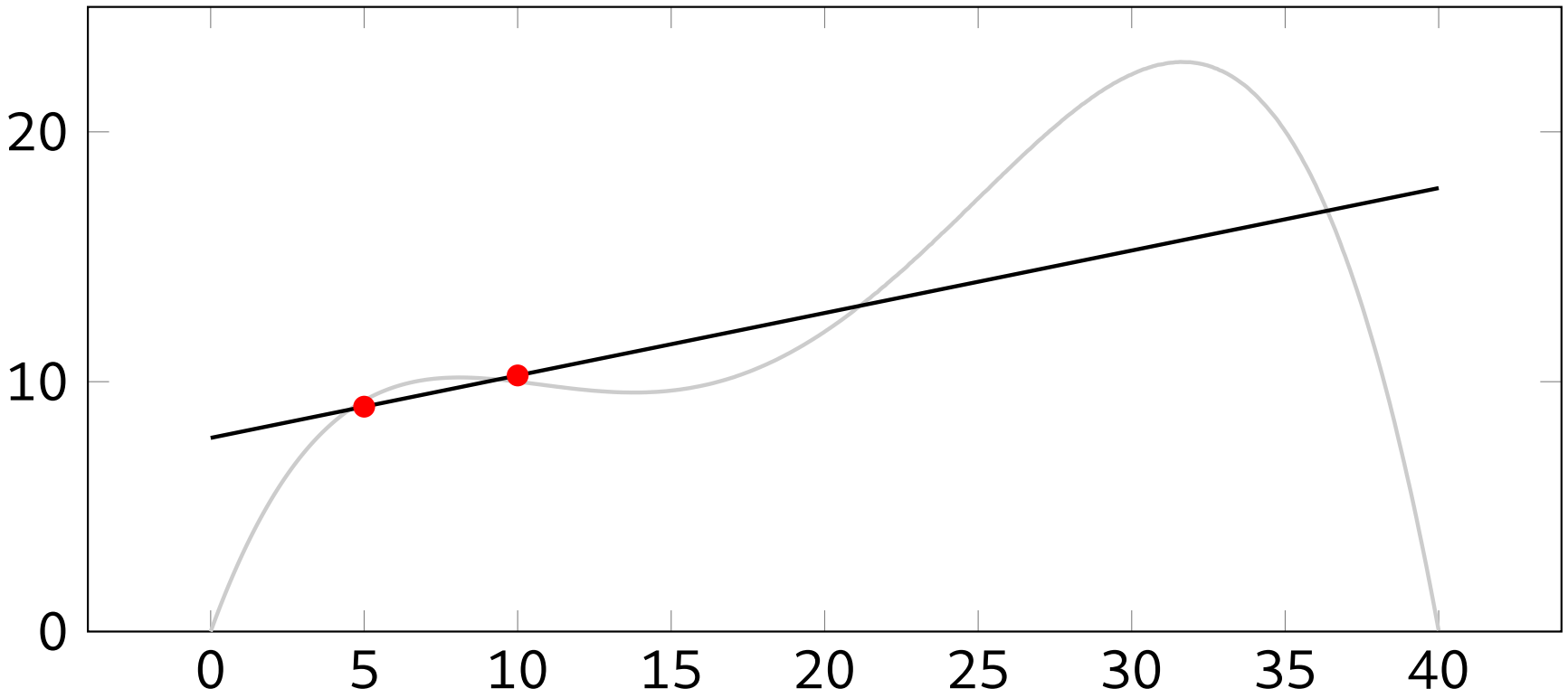
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

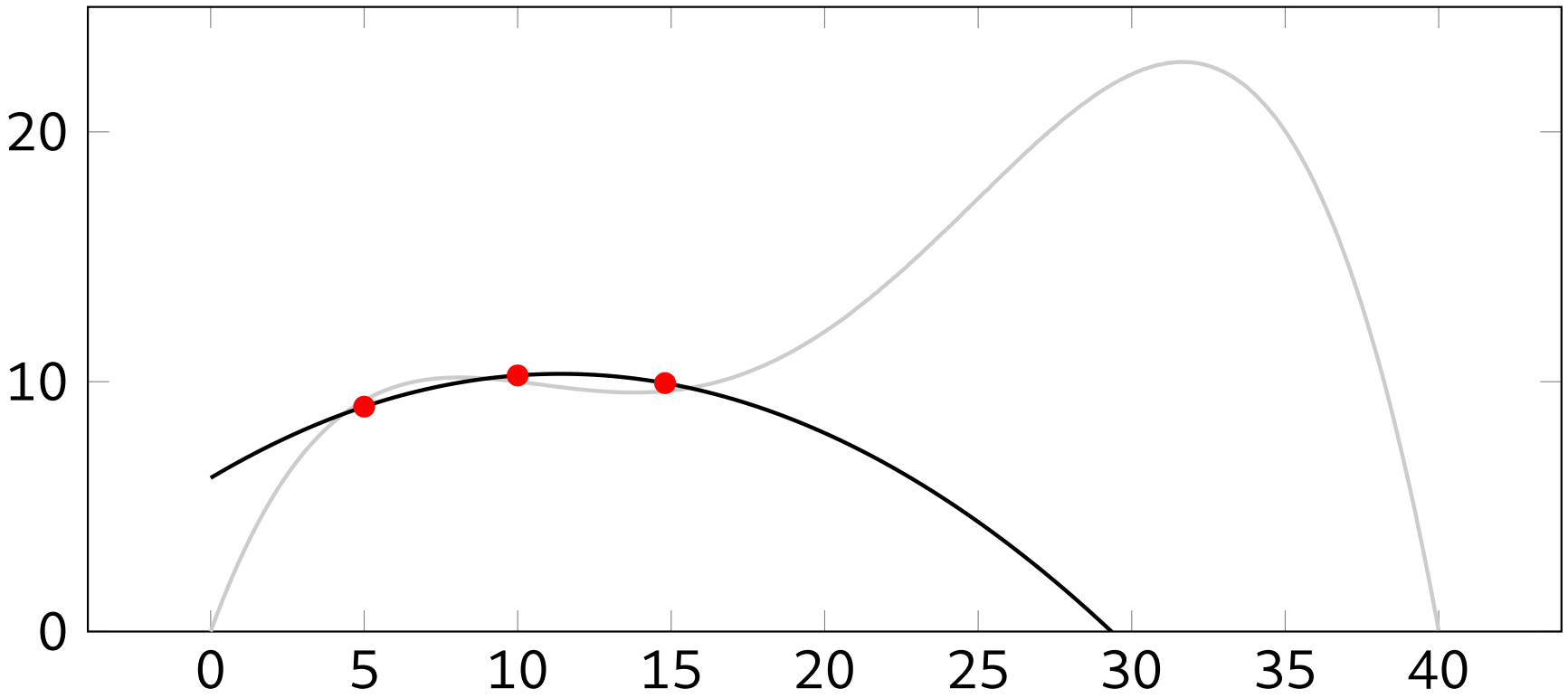
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

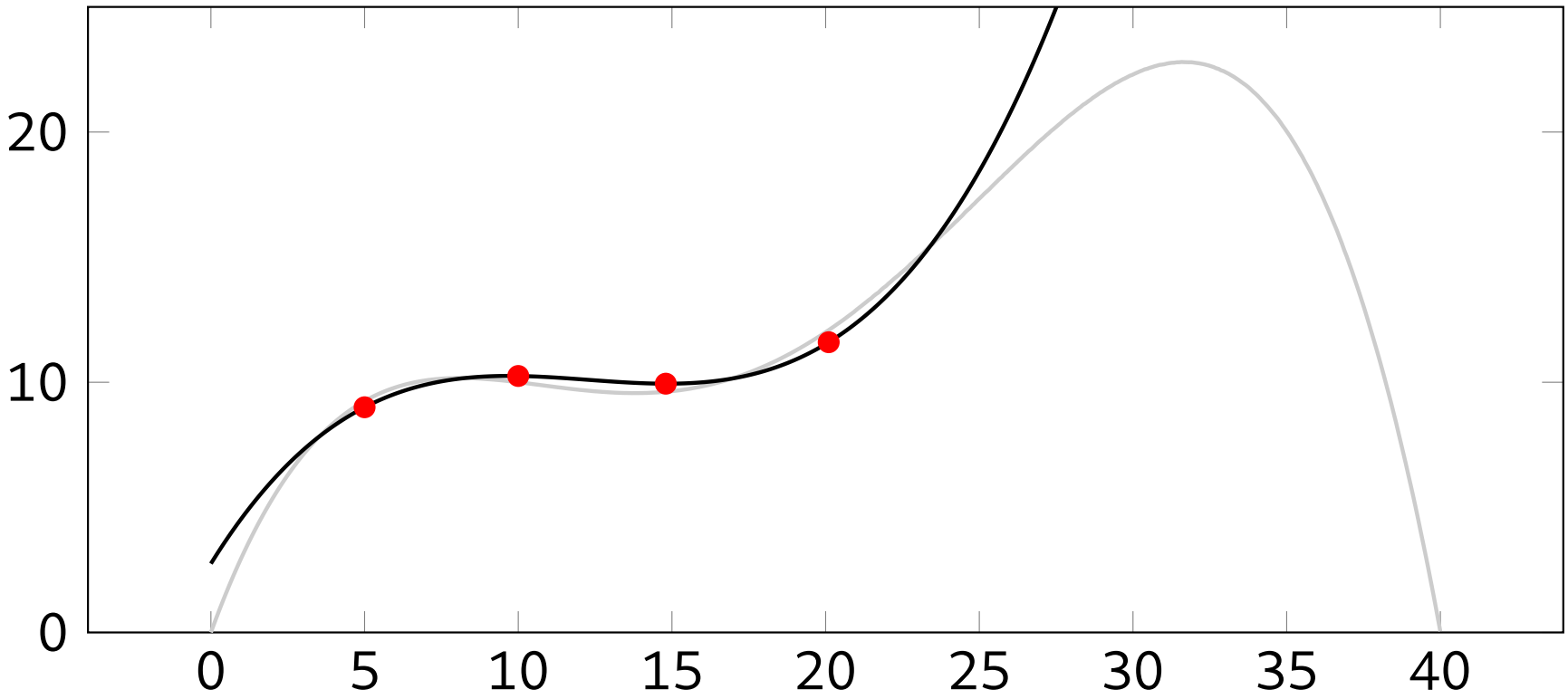
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

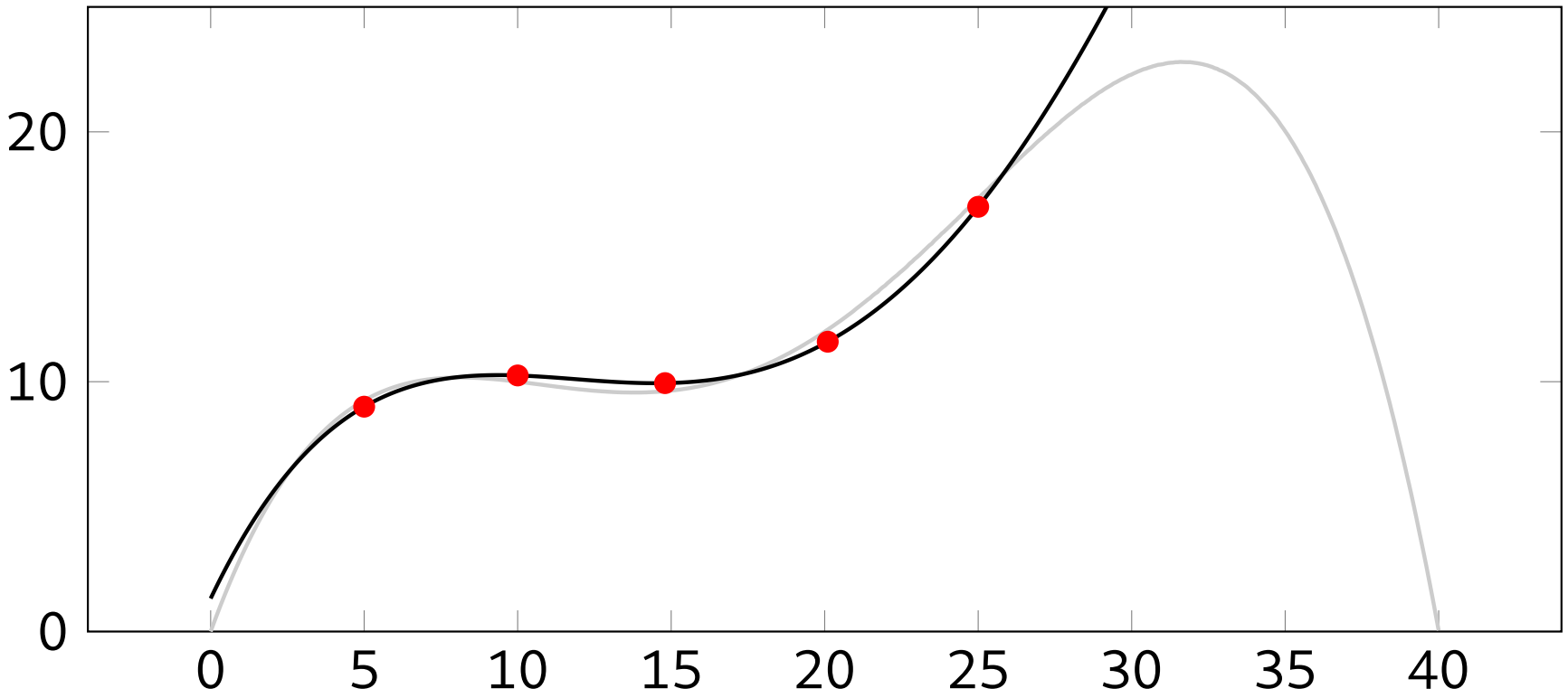
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

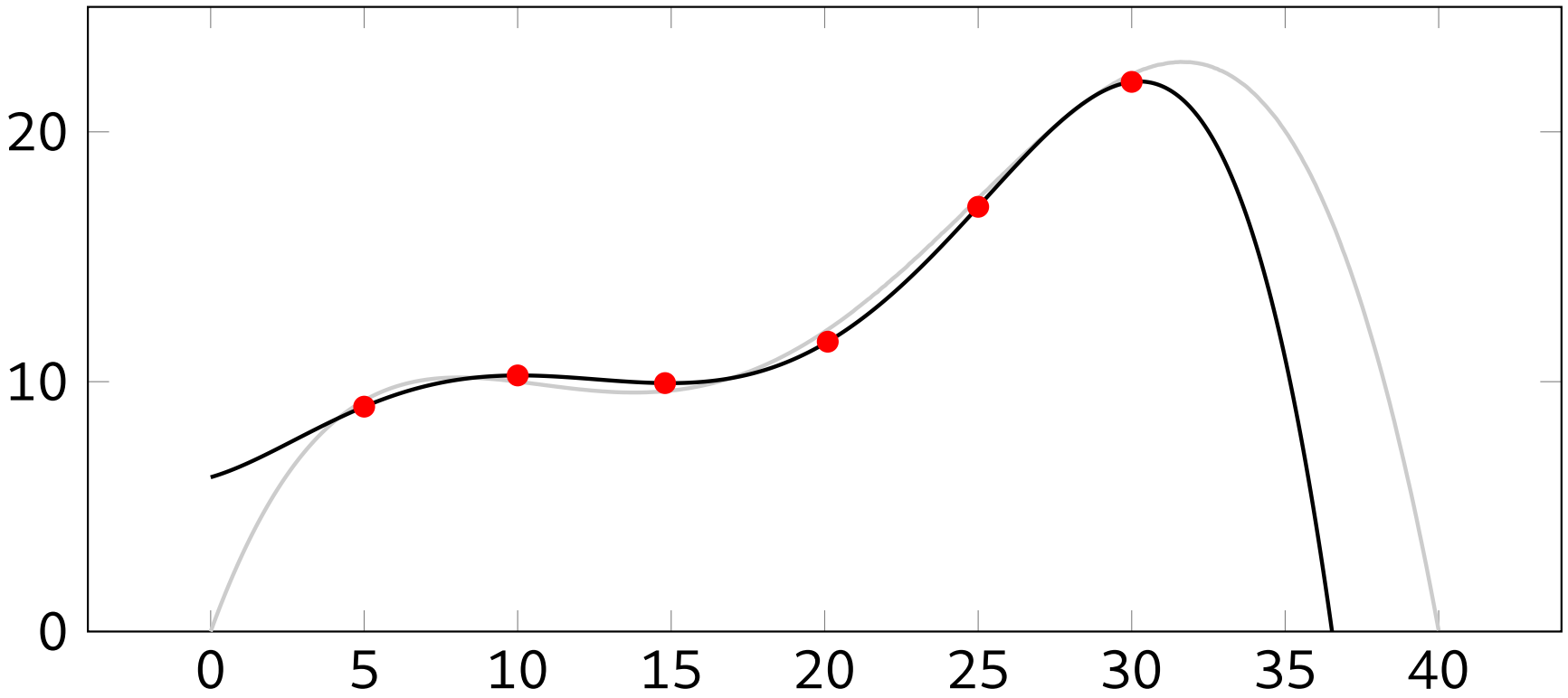
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

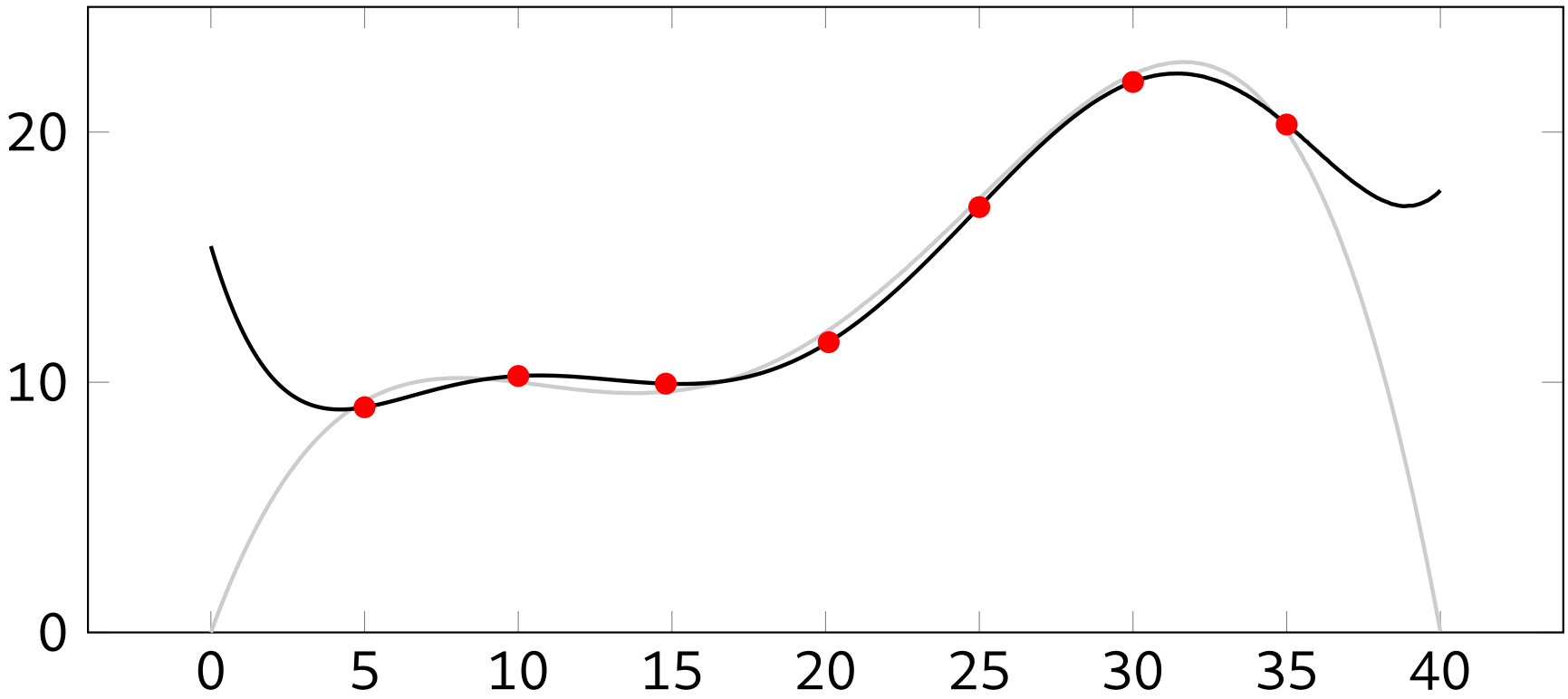
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

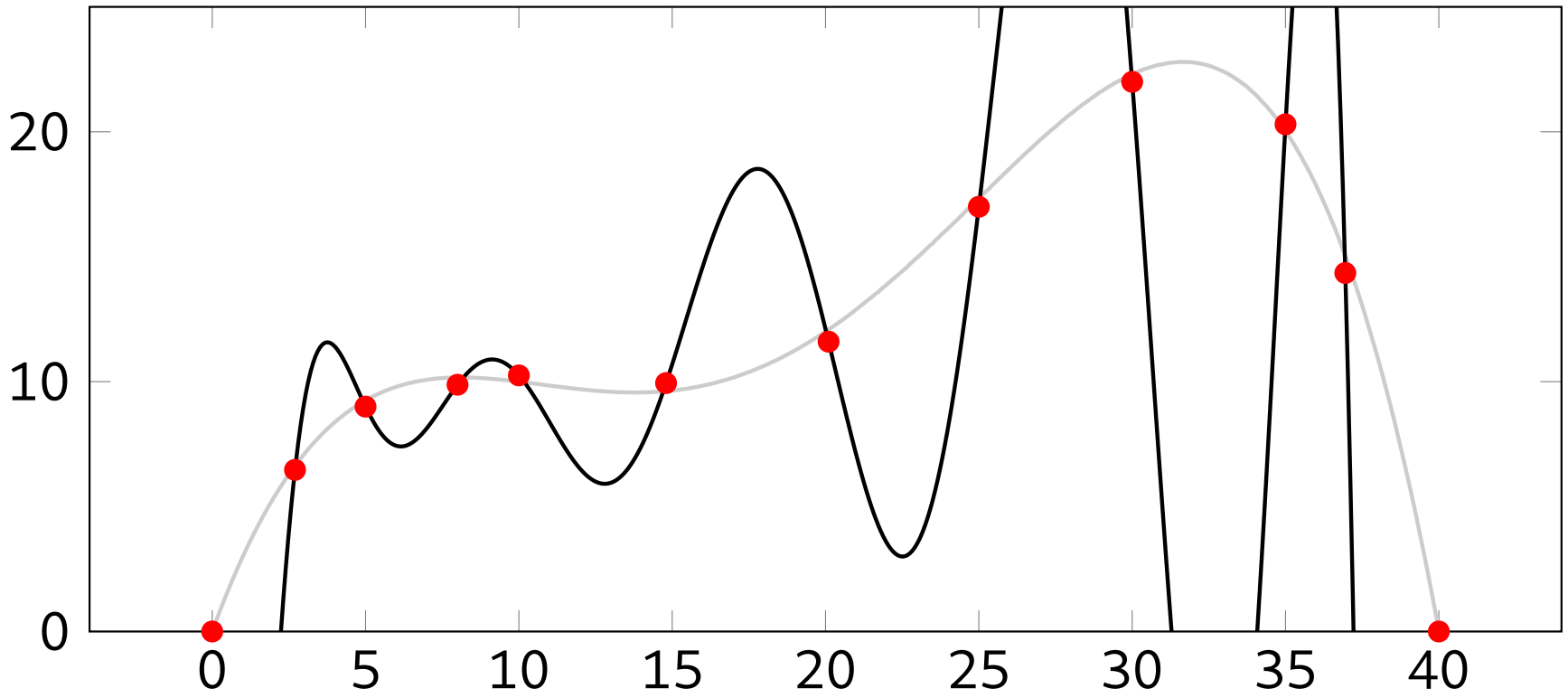
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

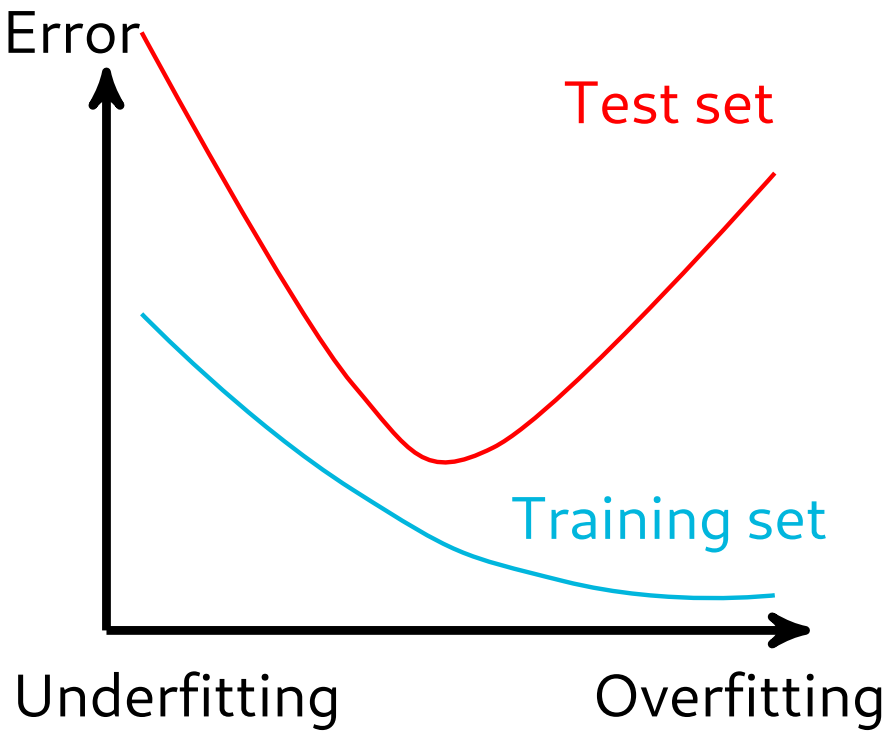
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Overfitting

Bias/variance trade-off

- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Crossvalidation

- To detect overfitting, split training dataset in two parts:
  1. A first part is used to train,
  2. A second part is used to validate,