UNIVERSITÄT BERN

Machine Learning and Deep Learning

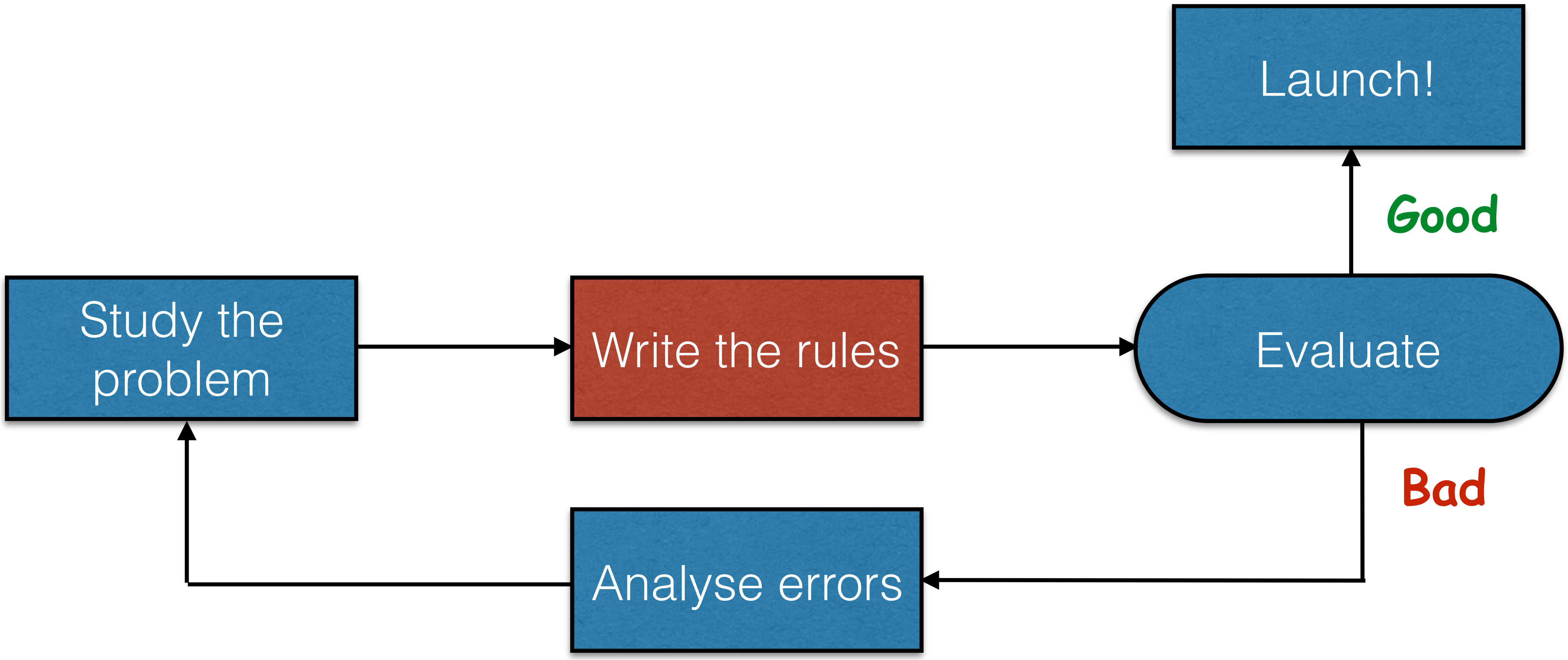
Simon Jenni
(slides by Paolo Favaro)

Deep Learning

- **Objective**: Build a machine that can learn from experience and understand the world as a hierarchy of concepts

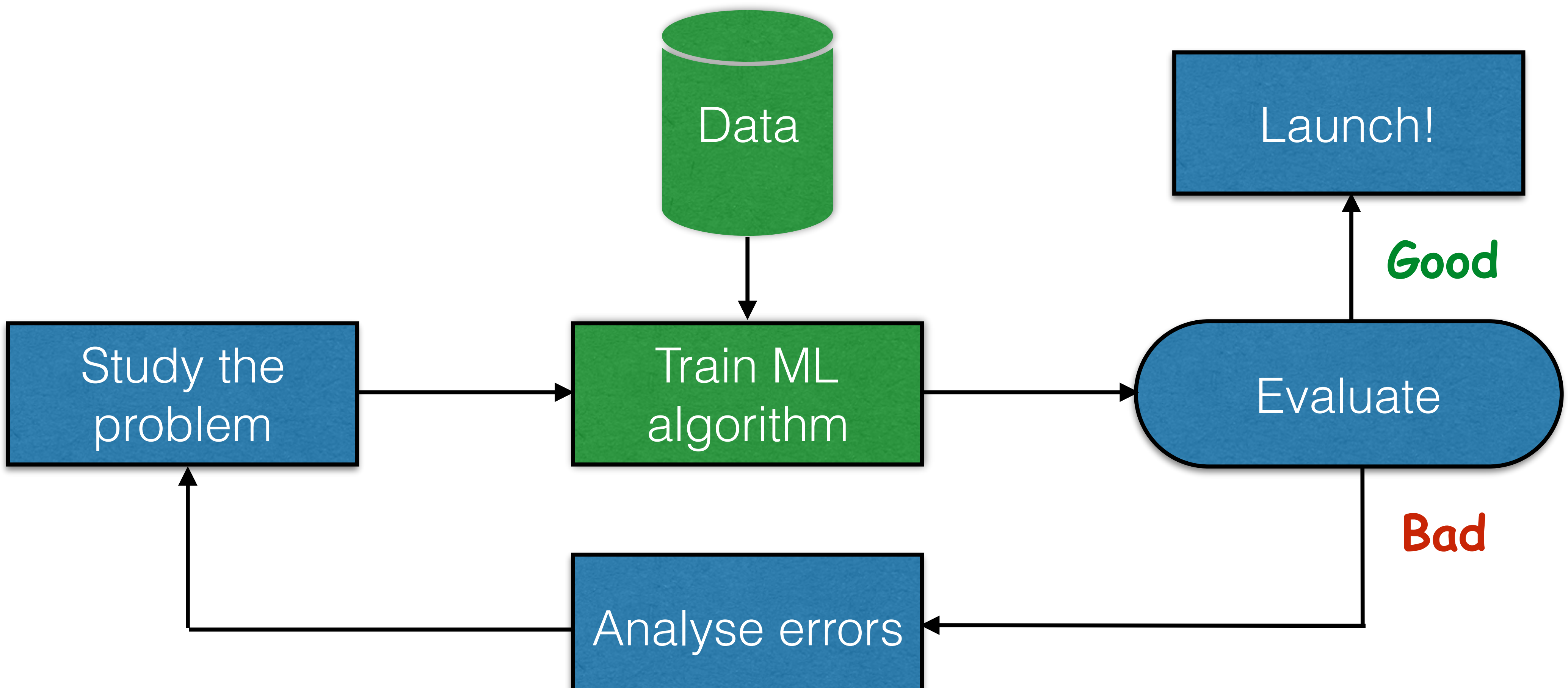
Traditional Approach

- List of all the knowledge and formal rules
  - works for games and simple systems
  - leads to a combinatorial problem
  - **not general (often we do not know the rules)**



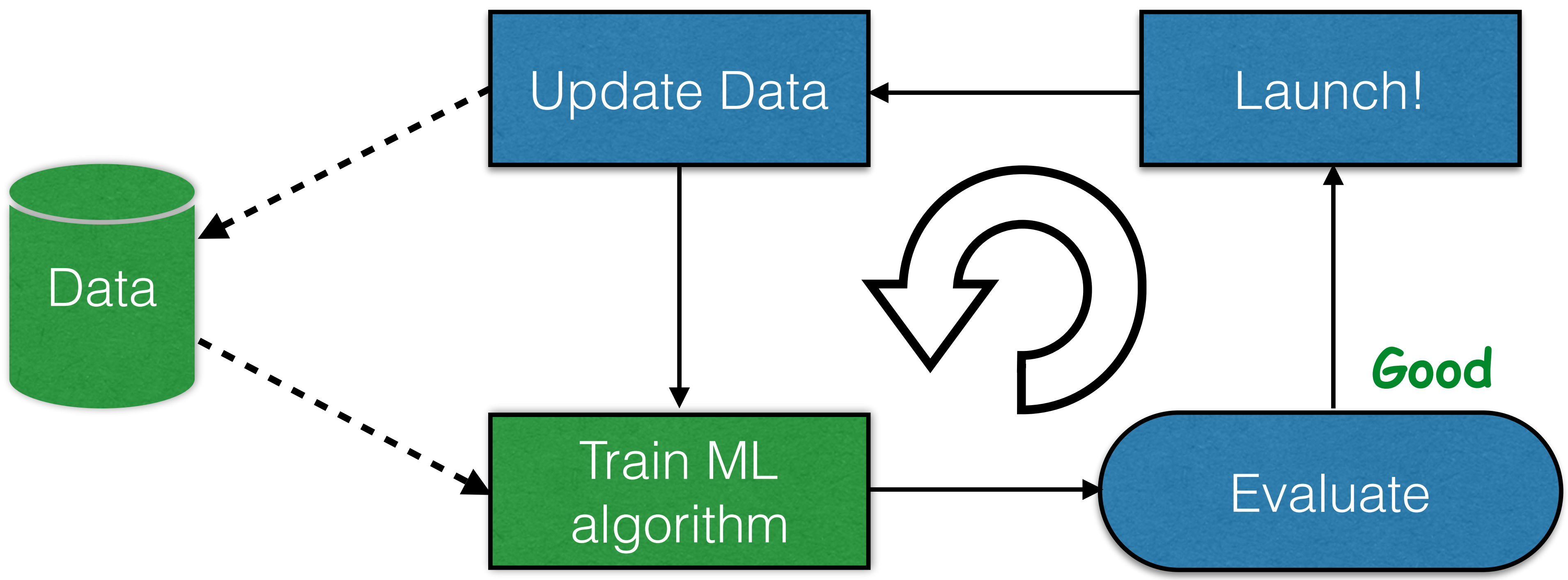
Learning from Examples

- The machine automatically learns from examples
  - machine learning
  - no need to identify and explain rules
  - **general and flexible**



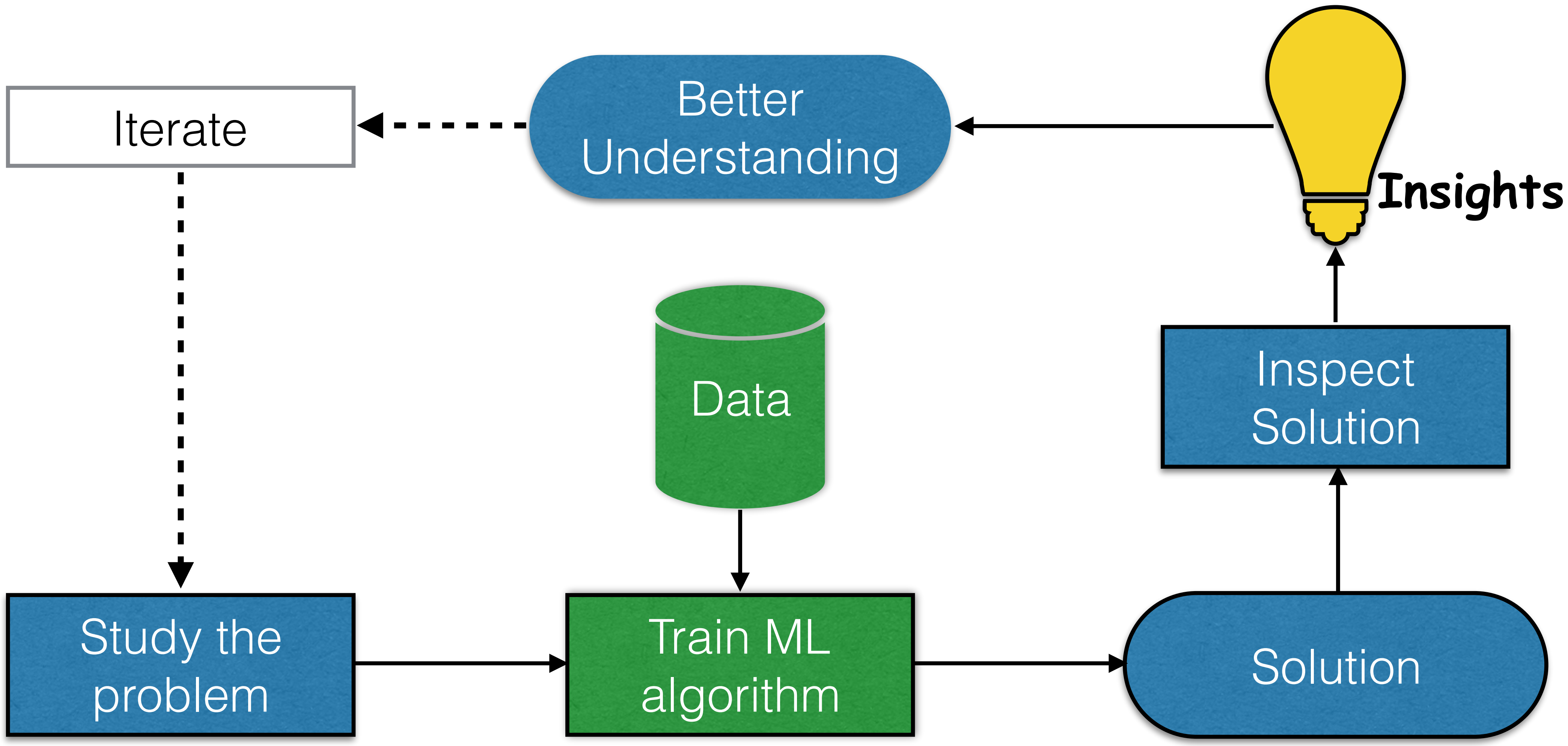
Adapting to Change

- Machine Learning can automatically adapt to change
  - Simply update the data and train again
  - No need to change the underlying algorithm



Help Humans Learn

- Machine Learning algorithms can be inspected
  - Might lead to new insights
  - Can uncover patterns in the data



Features

- Machines solve tasks/decisions by using the provided information (data)

- Data is often encoded into more focused relevant information (features) to simplify the decision

$$x \rightarrow \phi(x)$$

data ↗ ↖ feature

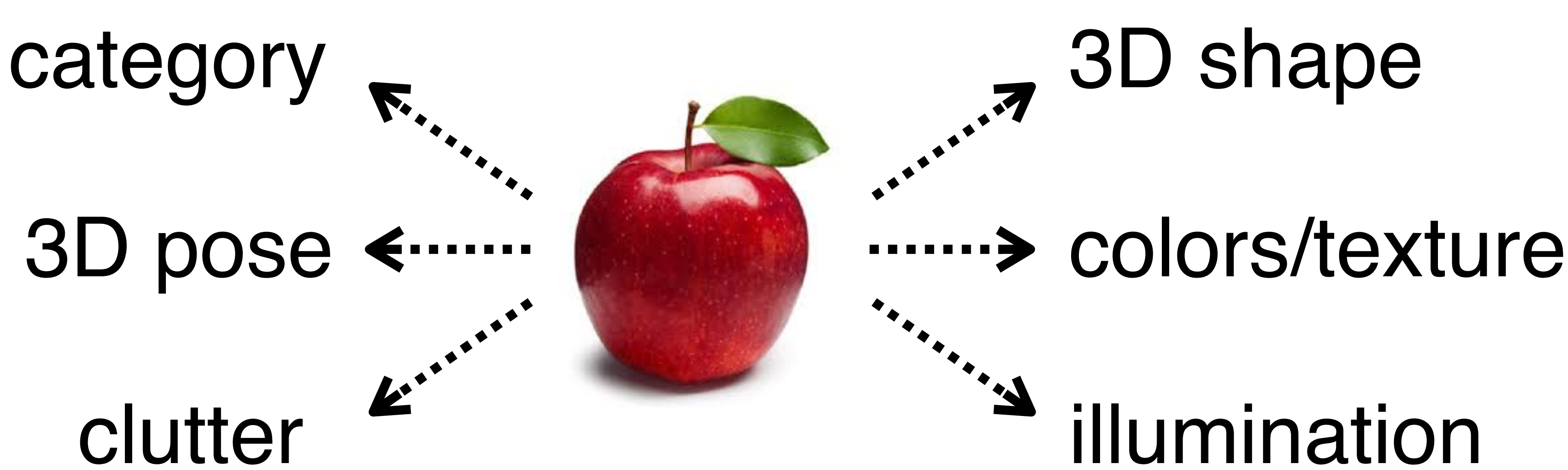
- Features can be hand-made/encoded

    - <u>Operators often do not know the optimal features</u>

Representation Learning

- Features or, more in general, an **internal representation** or a hierarchy of concepts should be learned automatically

- The internal representation should separate all **factors of variation** (i.e., concepts that summarize important variation of the data)

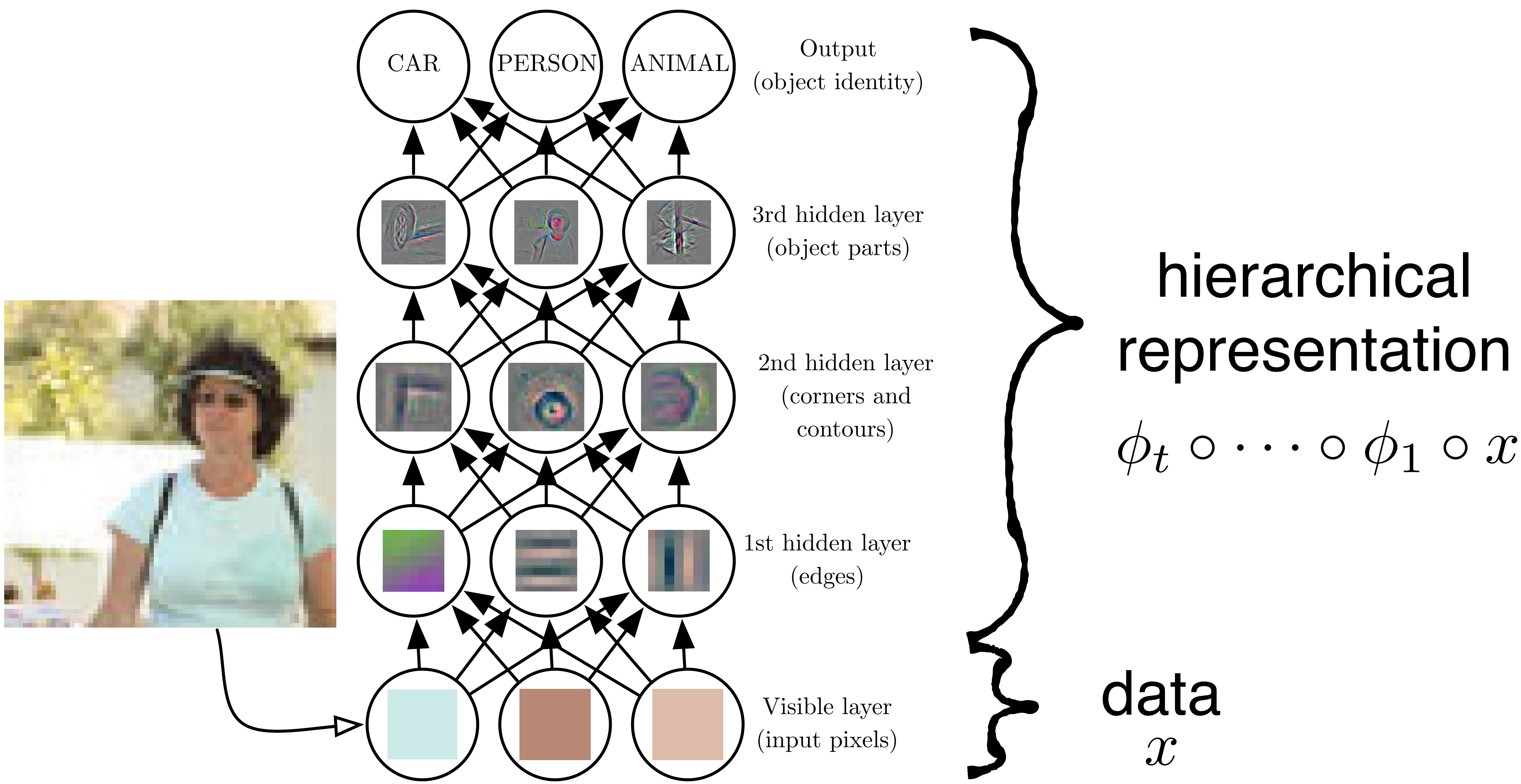


Distributed Representation

- Use many features to represent data and each feature should handle multiple data samples
- Example: Recognition of cars, trucks and birds and each can be red, green or blue
- Case #1: 1 feature for each case
(3x3 = 9 features)
- Case #2: 3 features for identity and 3 for color
(3+3 = 6 features)

Deep Learning

Introduces hierarchical representations (from simple to complex, from low-level features to high-level features)



UNIVERSITÄT BERN

Machine Learning Review

Simon Jenni
(slides by Paolo Favaro)

Contents

- Revision of basic concepts of Machine Learning

- Based on **Chapter 5** of Deep Learning by Goodfellow, Bengio, Courville

Context

- A more complete introduction to Machine Learning through the following courses
  - Machine Learning @ UniBe
  - Machine Learning and Data Mining @ UniNe
  - Pattern Recognition @ UniFr
  - Statistical Learning Methods @ UniNe

Resources

- Books and online material for further studies
  - Machine Learning @ Stanford (Andrew Ng)
  - **Pattern Recognition and Machine Learning** by Christopher M. Bishop
  - **Machine Learning: a Probabilistic Perspective** by Kevin P. Murphy

Learning Pillars

- Supervised learning
- Semi-supervised learning
- Self-taught learning (unsupervised feature learning)
- Unsupervised learning (+self-supervised learning)
- Reinforcement learning

Definition

- Mitchell (1997)
  *A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.*

The Task T

- Example: if we want a robot to be able to walk, then **walking** is the task

- Approaches

1. We could directly input **directives** for how we think a robot should walk, or

2. We could provide **examples** of successful and unsuccessful walking (this is machine learning)

The Task T

- Given an input x (e.g., a vector) produce a function f, such that f(x) = y (e.g., an integer, a probability vector)
- Examples
    - Classification
    - Regression
    - Machine translation
    - Denoising
    - Probability density estimation

The Performance Measure P

- To evaluate a ML algorithm we need a way to measure how well it performs on the task
- It is measured on a separate set (**the test set**) from what we use to build the function f (**the training set**)
- Examples
  - Classification accuracy (portion of correct answers) or error rate (portion of incorrect answers)
  - Regression accuracy (e.g., least squares errors)

The Experience E

- Specifies what data can be used to solve the task
- We can distinguish it based on the learning pillars
  - **Supervised**: data is composed of both the input x (e.g., features) and output y (e.g., labels/targets)
  - **Unsupervised**: data is composed of just x; here we typically aim for p(x) or a method to sample p(x)
  - **Reinforcement**: data is dynamically gathered based on previous experience

Data

- We assume that all collected data samples in all datasets:

  1. come from the same distribution → $p_{x^{(i)}}(x) = p_{x^{(j)}}(x)$

  2. are independent → $p\left(x^{(1)}, \ldots, x^{(m)}\right) = \prod_{i=1}^{m} p\left(x^{(i)}\right)$

- This assumption is denoted **IID** (independent and identically distributed)

Example: Linear Regression

- Given IID data inputs $x \in \mathbb{R}^n$ and outputs $y \in \mathbb{R}$
- **Task T**: predict y with the linear regressor $\hat{y} = w^\top x$ need to find the weights w
- **Experience E**: training set $X^{\text{train}} \in \mathbb{R}^{m \times n}$, $Y^{\text{train}} \in \mathbb{R}^m$ and test set $X^{\text{test}} \in \mathbb{R}^{m \times n}$, $Y^{\text{test}} \in \mathbb{R}^m$
- **Performance P**: Mean squared error

$$\text{MSE}^{\text{test}}(w) = \frac{1}{m}|X^{\text{test}}w - Y^{\text{test}}|^2$$

Linear Regression

- Solve task T by minimizing the MSE$^{\text{train}}$

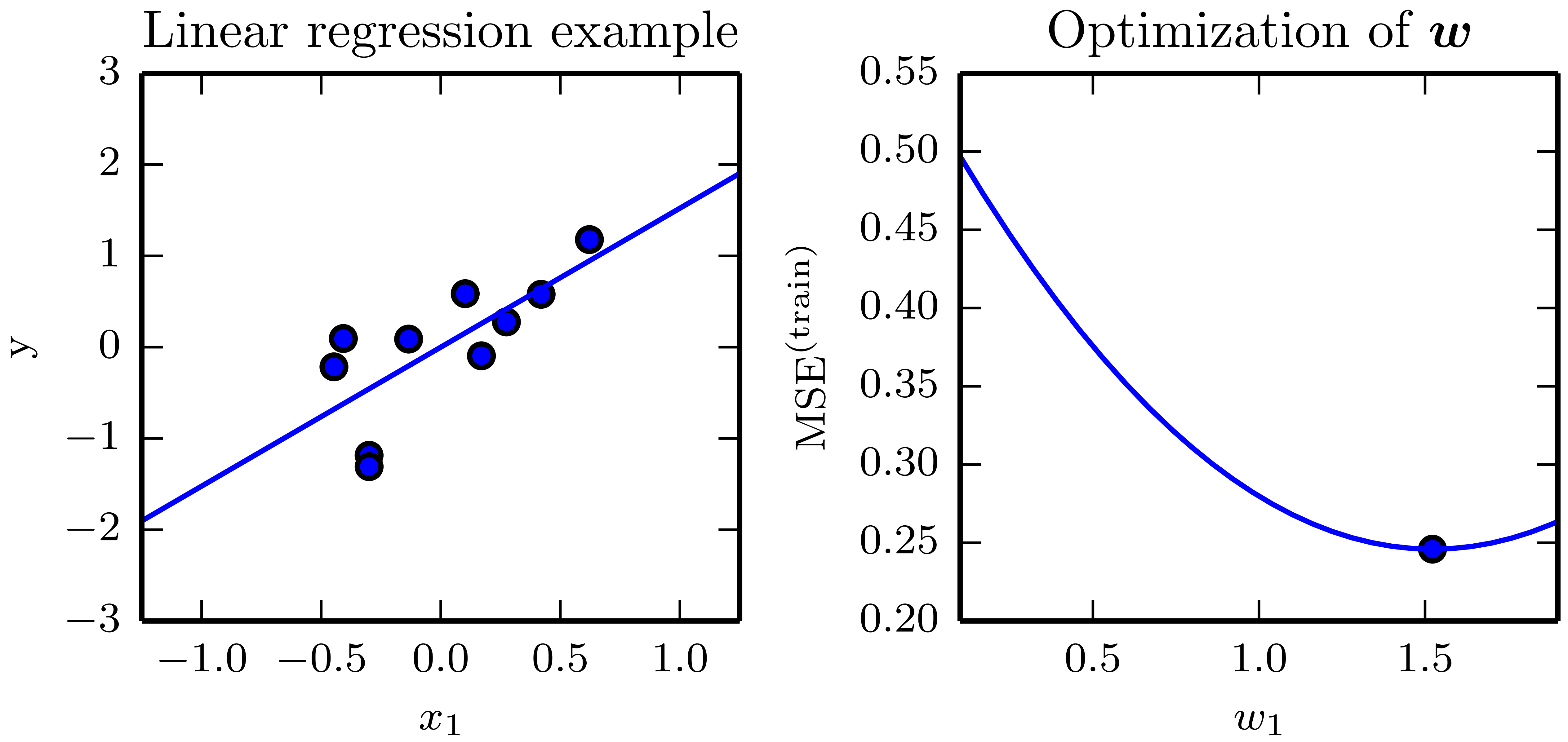
$$\mathrm{MSE}^{\mathrm{train}}(w) = \frac{1}{m} |X^{\mathrm{train}} w - Y^{\mathrm{train}}|^2$$

- Compute the gradient of MSE$^{\text{train}}$(**w**) with respect to **w** and set to 0 (normal equations)

- The solution is (pseudo-inverse)

$$w = \left({X^{\mathrm{train}}}^\top X^{\mathrm{train}} \right)^{-1} {X^{\mathrm{train}}}^\top Y^{\mathrm{train}}$$

Linear Regression

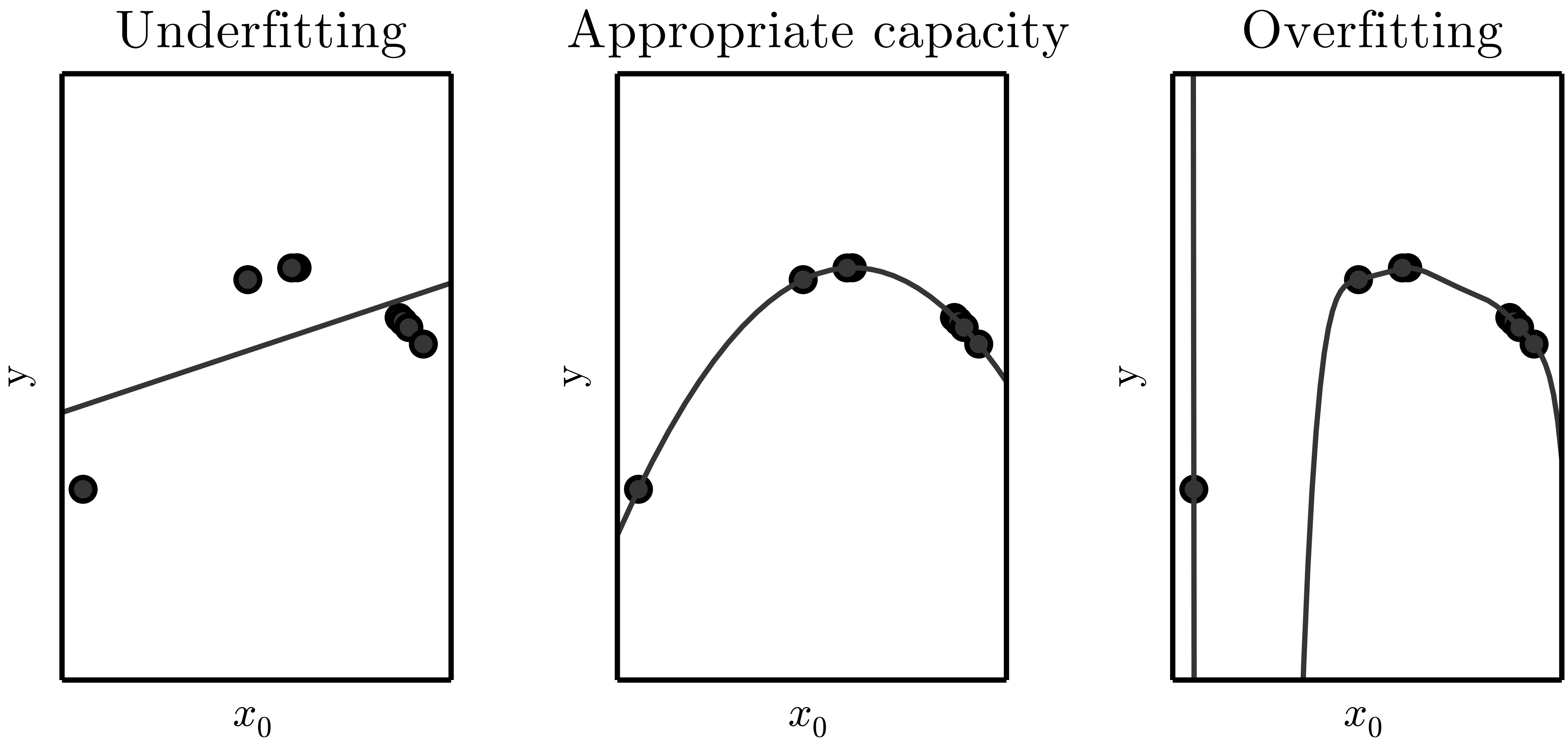


Overfitting and Underfitting

- Performance P captures how well the learned model predicts new unseen data
- Ideally we want to select the predictor with the best performance
- What happens when we use predictors of different complexity/capacity?

Overfitting and Underfitting

shown data is the training set



simple predictor | optimal predictor | complex predictor

Loss function

- Define a **predictor** function $f : \mathcal{X} \mapsto \mathcal{Y}$
- Define a **loss** function $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ which measures how different the two inputs are
- Examples
  - 0-1 loss $l(y, f(x)) = \begin{cases} 0 \text{ if } y = f(x) \\ 1 \text{ if } y \neq f(x) \end{cases}$
  - Quadratic loss $l(y, f(x)) = (y - f(x))^2$

Bayes Risk

- **Bayes risk** is defined as (average loss)

$$R(f) = E_{x,y}[l(f(x), y)] = \int l(f(x), y)p(x, y)dxdy$$

- The optimal predictor function is

$$f^* = \arg\min_f R(f)$$

Empirical Risk

- Given ($\mathbf{x_i}$,$\mathbf{y_i}$) with **i** = 1,…,**m** the **empirical risk** is

$$\hat{R}(f) = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i)$$

- The empirical predictor is

$$\hat{f} = \arg\min_{f\in\mathcal{F}} \hat{R}(f)$$

Risks

- Bayes risk $R(f^*) = E_{x,y}[l(f^*(x), y)]$
- Empirical risk $\hat{R}(\hat{f}) = \frac{1}{m}\sum_{i=1}^{m} l(\hat{f}(x_i), y_i)$
- Bayes risk restricted to function family $\min_{f \in \mathcal{F}} R(f)$

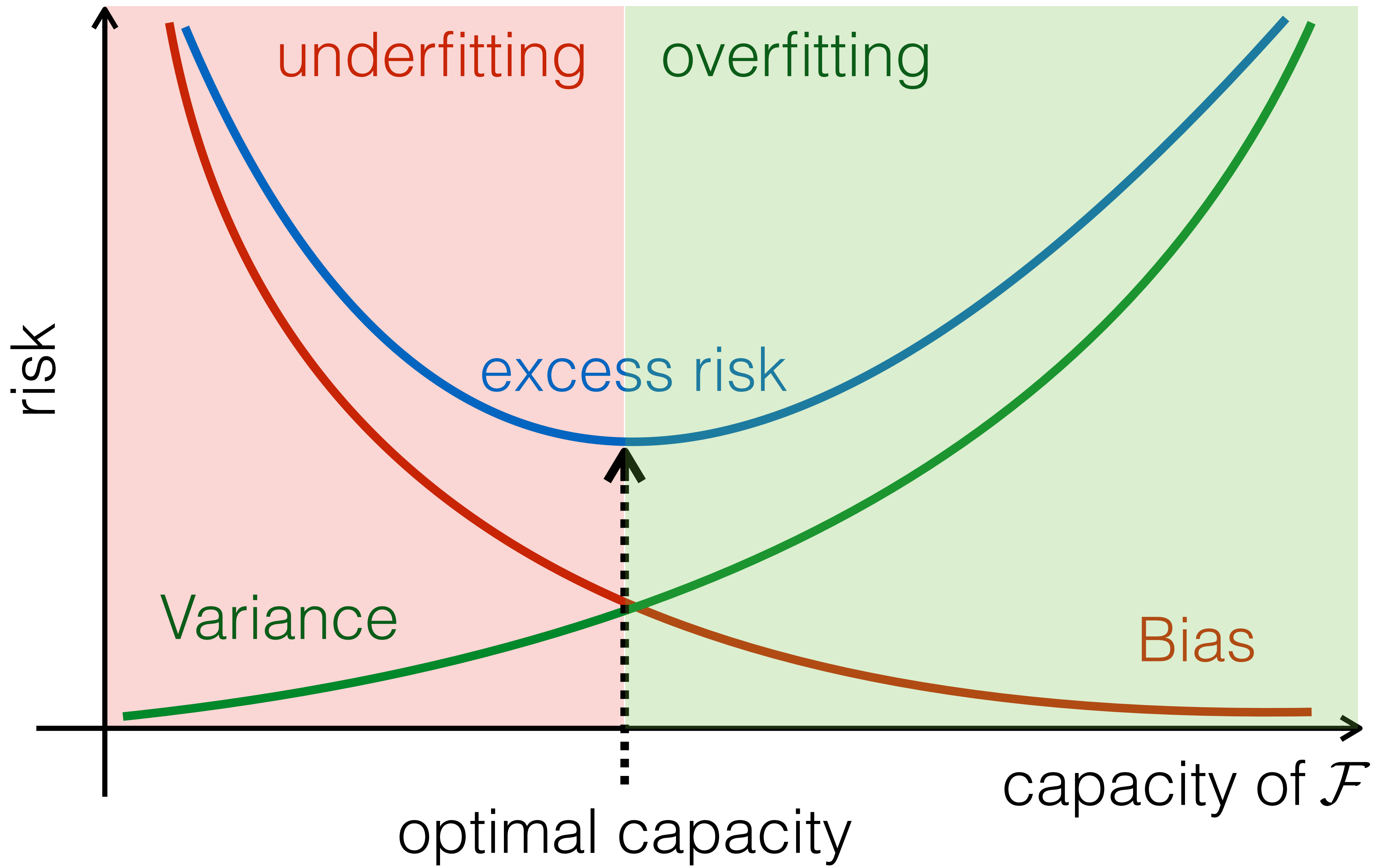
Estimation vs Approximation

- The **excess risk** is the gap between the empirical risk and the optimal Bayes risk

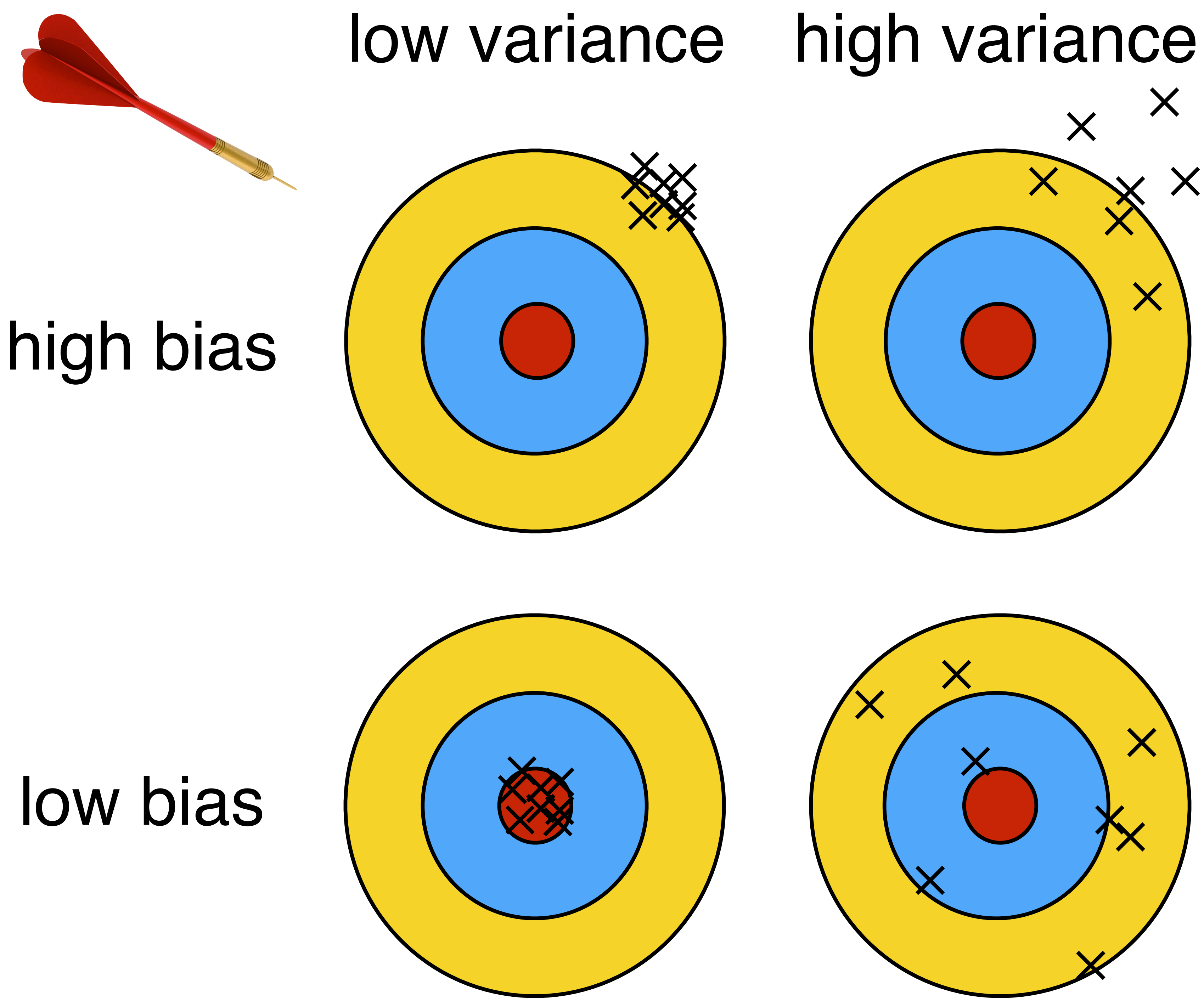
$$\hat{R}(\hat{f}) - R(f^*) = \underbrace{\hat{R}(\hat{f}) - \min_{f \in \mathcal{F}} R(f)}_{\text{estimation error}} + \underbrace{\min_{f \in \mathcal{F}} R(f) - R(f^*)}_{\text{approximation error}}$$

- **Estimation (variance)**: due to training set
- **Approximation (bias)**: due to function family $\mathcal{F}$

Estimation vs Approximation



Bias and Variance



Concept by Pedro Domingos
University of Washington


THE BEST WAY TO
EXPLAIN OVERFITTING
imgflip.com
ADDTEXT.COM

Regularization

- Define a parametric family $\mathcal{F}_\lambda$ of functions, where λ regulates the complexity/capacity of the predictors

- Given the optimal predictor from the empirical risk

$$\hat{f}_\lambda = \arg \min_{f \in \mathcal{F}_\lambda} \hat{R}(f)$$

we would like to choose the capacity based on Bayes risk

$$R(\hat{f}_\lambda)$$

Training, Validation and Test

- In alternative, collect samples into training set D_{train} validation set D_{val} and test set D_{test}

- Use the **training set** to define the optimal predictor

$$\hat{f}_\lambda = \arg\min_{f \in \mathcal{F}} \hat{R}_{D_{\text{train}}}(f)$$

- Use the **validation set** to choose the capacity

$$\hat{\lambda} = \arg\min_{\lambda} \hat{R}_{D_{\text{val}}}(\hat{f}_\lambda)$$

- Use the **test set** to evaluate the performance

$$\text{performance P} = R_{D_{\text{test}}}\left(\hat{f}_{\hat{\lambda}}\right)$$

Supervised Learning

- Make a prediction of an output y given an input x
- Boils down to determining the conditional probability

$$p(y|x)$$

- Formulate problem as that of finding θ for a parametric family (Maximum Likelihood)

$$p(y|x;\theta)$$

Maximum Likelihood

- Given IID input/output samples $(x^i, y^i) \sim p_{\text{data}}(x, y)$

  the **conditional maximum likelihood** estimate is

$$\theta_{\text{ML}} = \arg\max_{\theta} \prod_{i=1}^{m} p_{\text{data}}(y^i | x^i; \theta)$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\text{data}}(y^i | x^i; \theta)$$

Logistic Regression

- **Example**: Binary classification $y \in \{0, 1\}$
- We aim at determining $p(y = 1|x; \theta) = \sigma(\theta^\top x)$

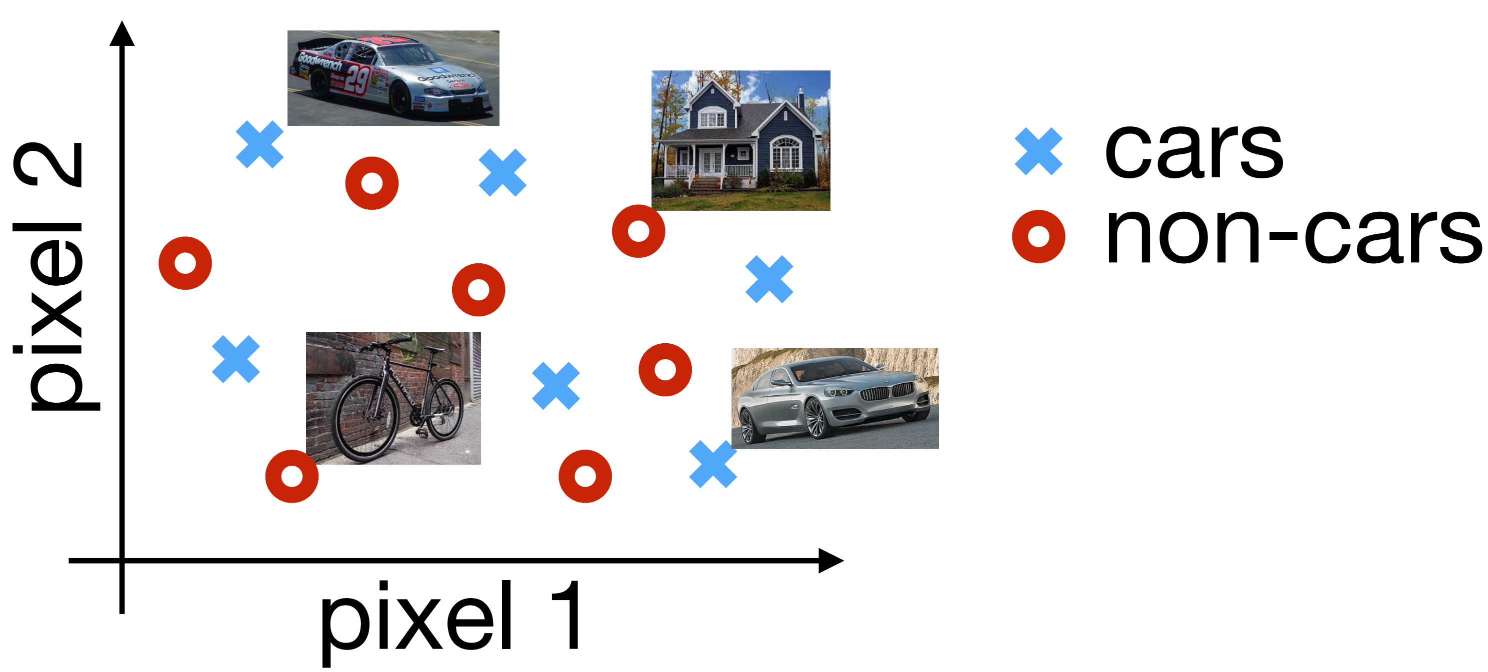
  where $\sigma(z) = \dfrac{1}{1 + e^{-z}}$ is the sigmoid function
- Class y=1 can be picked when

$$p(y = 1|x; \theta) > p(y = 0|x; \theta)$$

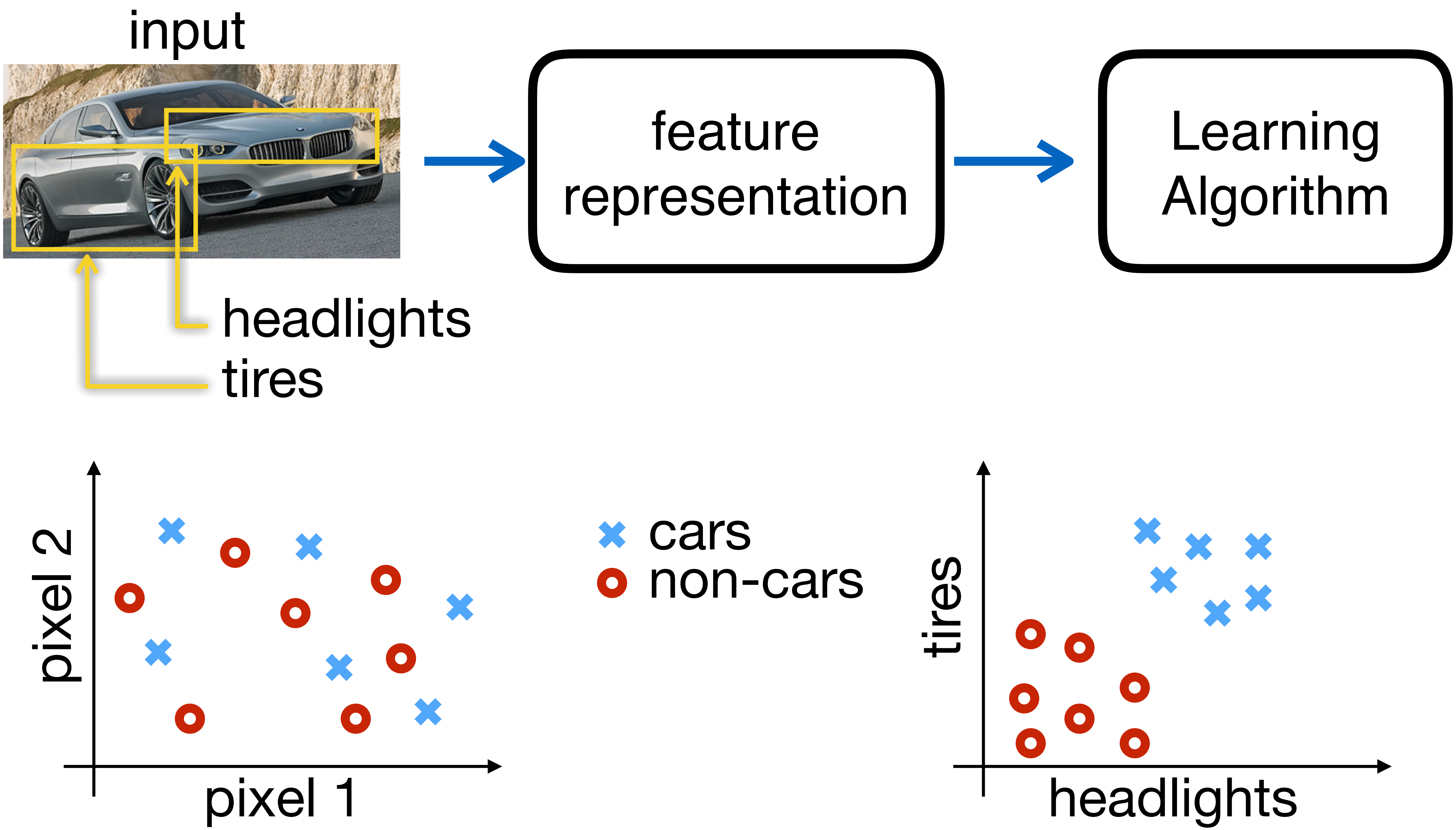
which is equivalent to $\theta^\top x > 0$

Features


input
pixel 2
pixel 1
Learning Algorithm


pixel 2
pixel 1
cars
non-cars

Features

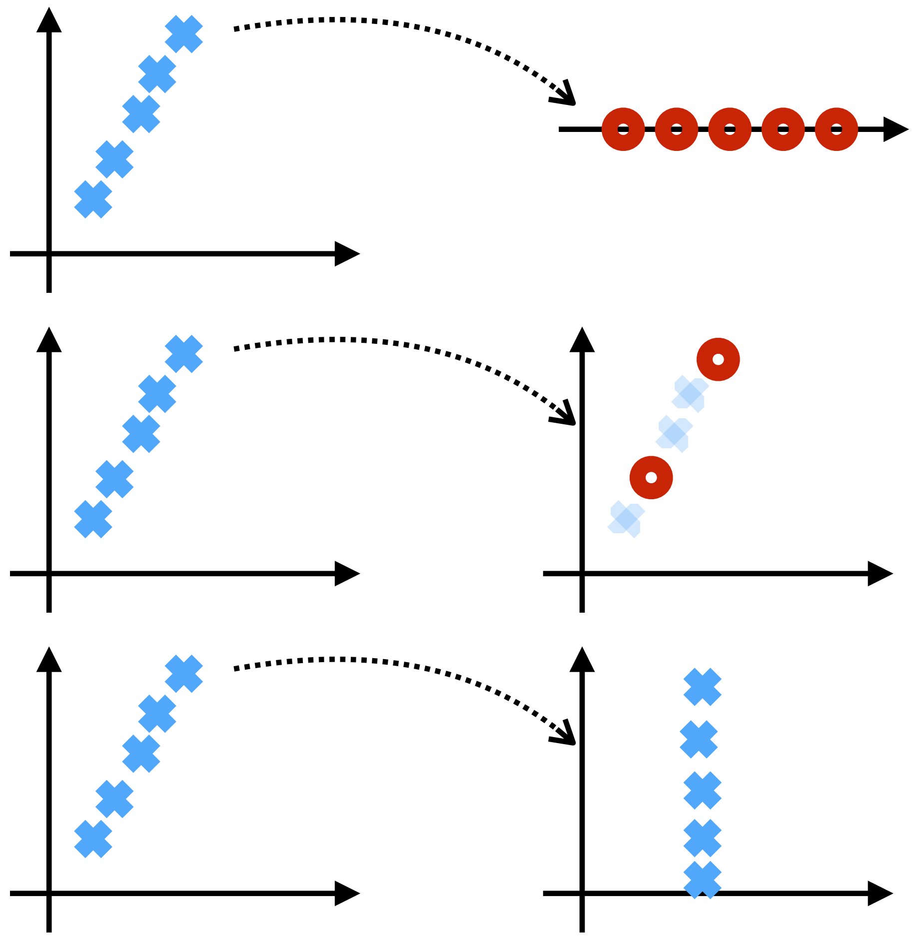

input
feature representation
Learning Algorithm
headlights
tires
pixel 2
pixel 1
cars
non-cars
tires
headlights

Unsupervised Learning

- Aim is to find a suitable **data representation**
  - Probability density estimator
  - Sampling procedure
  - Data denoising
  - Manifold learning
  - Clustering

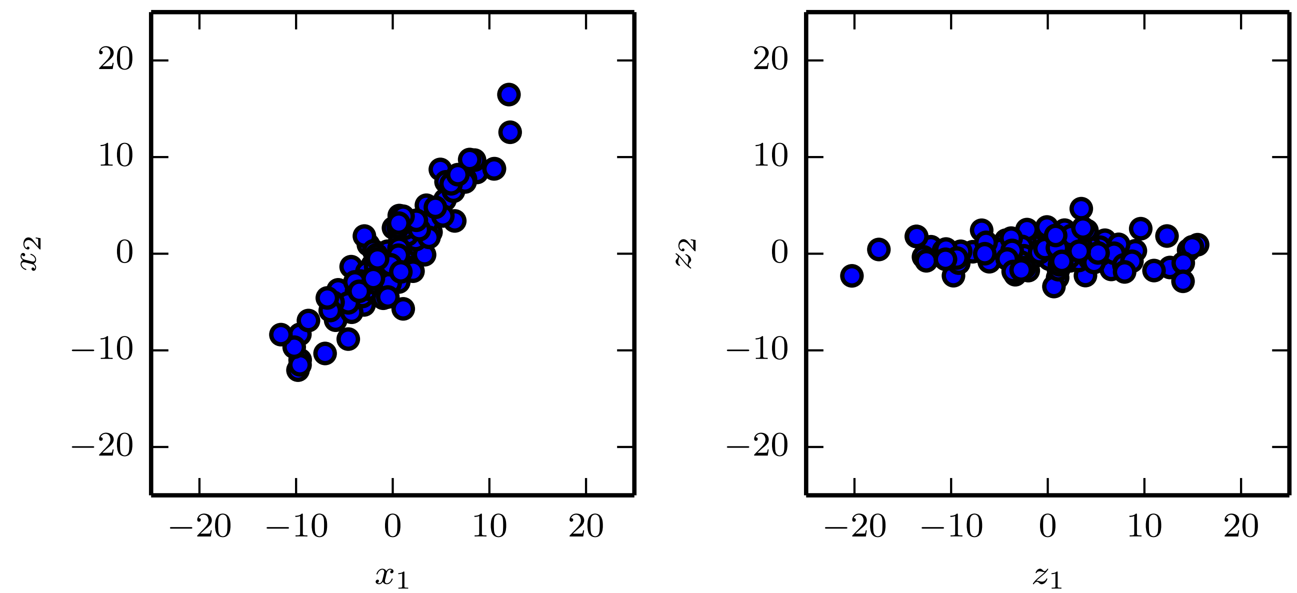
Data Representation

- The ideal data representation should:
  1. **Preserve** all task-relevant information
  2. Be **simpler** than the original data and **easier** to use
     - (i) low-dimensional
     - (ii) sparse
     - (iii) independent



Principal Components Analysis

- **Definition**: Project data X so that the largest variation of the projected data $Z = U^\top X$ is axis-aligned



$$X = U \Sigma V^\top$$

$$U^\top U = I \qquad \Sigma = \begin{bmatrix} \sigma_0 & 0 & \dots & 0 \\ 0 & \sigma_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix} \qquad V^\top V = I$$

singular values $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_n \geq 0$

Principal Components Analysis

- Unsupervised learning method for **linearly** transformed data

- A low-dimensional representation (by thresholding the singular values)

- Yields independent (uncorrelated) components

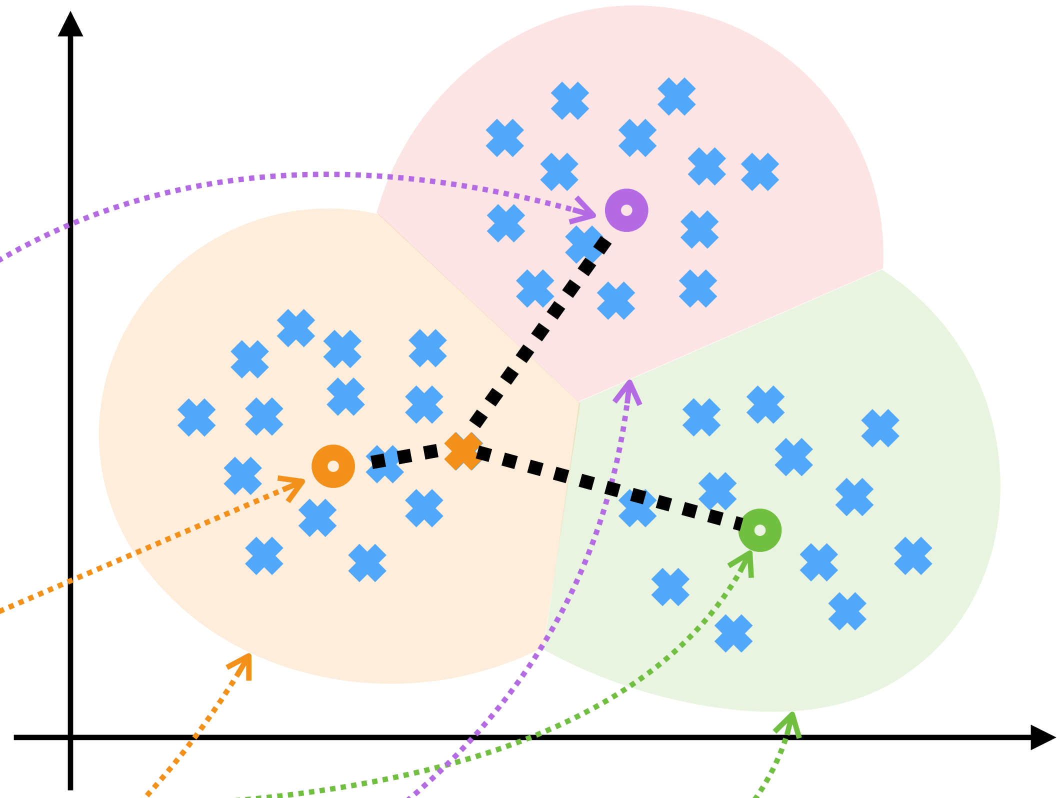
K-Means Clustering

- **Definition**: Find k clusters of data samples similar to each other

  Alternate between:

$$c_j = \frac{\sum_i \delta[w_i = j] x_i}{\sum_i \delta[w_i = j]}$$

$$w_i = \arg\min_j |x_i - c_j|^2$$



K-Means Clustering

- Unsupervised learning method (handles nonlinearly transformed data)
- A sparse representation (assignments $\mathbf{w}_i$ encode one sample with one of the cluster centers $\mathbf{c}_j$)
- Depends on initialization
- Ill-posed (multiple solutions can be valid)
- Number of clusters is usually unknown

Conclusion

- Machine Learning is about making computers better at some task by learning from data

- Many different ML systems:
  - Supervised (regression, classification, …)
  - Unsupervised (clustering, dim. reduction, …)

- We maximize the model likelihood over the training set and hope it will generalise to unseen data

- Data is important (garbage in, garbage out)!
  Model complexity should fit the data.

Thank you for your attention!

Questions?