

# Introduction and Overview

*Traditional Algorithms vs ML*

Understanding Machine Learning

## goals of this course

- understanding of foundational ML concepts and commonalities between different methods
- ability to properly use ML for scientific or business problems

## schedule of lectures

1. introduction and overview
2. statistical learning
3. non-linear models
4. generalization
5. deep learning
6. transformers
7. generative models
8. causality
9. reinforcement learning

# AI/ML Overview

# Main Areas of Artificial Intelligence

- **computer vision**

(spatial structures, state-of-the-art: Convolutional Neural Networks)

- **natural language processing**

(sequential structures, state-of-the-art: transformers)

- **automated decision making, robotics**

(reinforcement learning)

All of these are enabled by one key ingredient:

- *learning from experience* (**Machine Learning**)

- also: knowledge representation, automated reasoning (first indices in modern large language models)

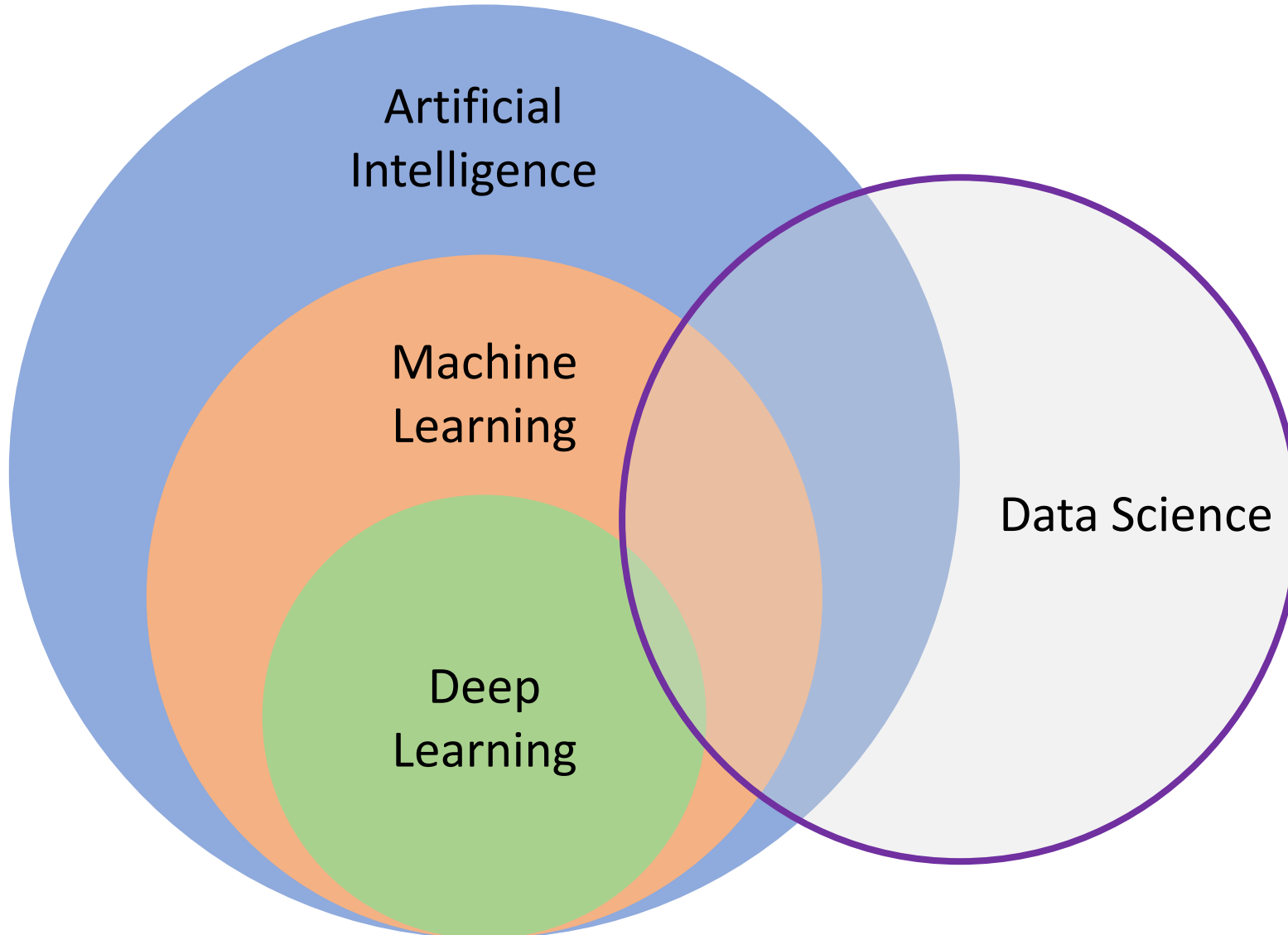


from wikipedia

agency:

perception – thought – action

# Buzz Words ...



## *Deep Learning:*

special kind of ML  
algorithms using (deep)  
neural networks

## *Data Science:*

extract knowledge from  
data (by means of ML,  
among other things)

# Traditional Algorithms and GOFAI

traditional algorithms:

explicit (handcrafted) instructions for each situation

symbolic AI (aka GOFAI):

use knowledge by means of symbols (as representations), logic, search (e.g., expert systems like Deep Blue)

*Public perception is changing over time: A modern chess program, nowadays disparaged as brute computing, would have been considered intelligent in the 50s.*



from wikipedia

# ML: Learning from Experience/Data

mainly exploiting statistical dependencies with the aim of **generalization** to new (e.g., future) data (compare with human reasoning by [analogies](#))

training (usually offline optimization):

**ML algorithm + data = explicit algorithm** (to be used at inference time)

→ reduction of complexity and much better generalizability compared to handcrafted algorithms

analogy: Humans do not hit the ground running (storage capacity of DNA limited) but have learning capabilities.

# Hot Debate: Connectionism vs Symbolic AI

connectionists:

*learn from (big) data without prior knowledge*

symbolists:

*use knowledge with only modest input data*

(crude) analogy: learning and evolution

philosophical: empiricist and rationalist schools of mind

hybrid approaches often most successful:

- e.g., AlphaGo with Deep Reinforcement Learning and Monte Carlo Tree Search
- feature engineering for ML models also kind of symbolic knowledge representation

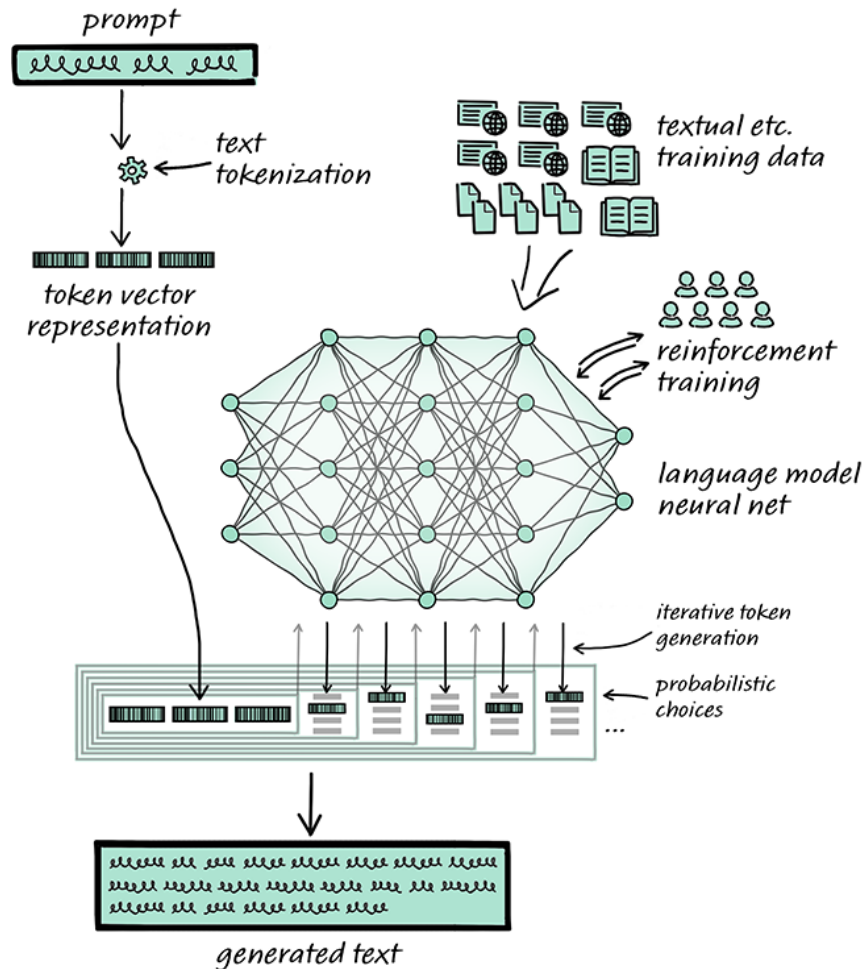


famous move 37  
from Max Tegmark

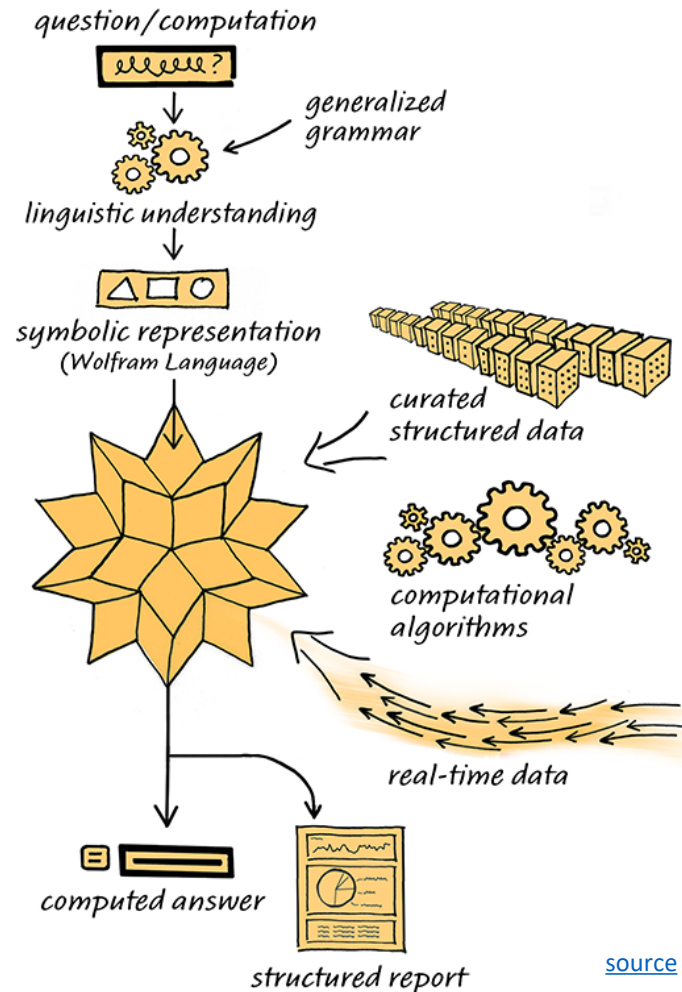


# Hybrid Approach for Language Models?

ChatGPT



Wolfram|Alpha



tool usage:  
[LangChain](https://langchain.ai)

# Supercharging the Scientific Method

use ML and data to replace or enhance explicit methods relying on detailed domain knowledge ([Software 2.0](#))

- overcome our evolutionary limitations in math with clever learning algorithms and collecting data
- immediate impact on many aspects of industry, business, and science, formulated as narrow tasks with strictly defined inputs (aka weak AI)

more imminent than (still philosophical) long-term quest for human-level AI (aka strong AI, AGI), i.e., general-purpose intelligence

(although recent language models show multi-purpose capabilities)

# When to apply ML?

## complexity

- decisions under uncertainty, many influencing factors
- e.g., demand forecasting, DNA sequencing
- difficult for humans, direct model inexpressible

## automation

- e.g., face and speech recognition, autonomous driving
- goal to reach human-level performance

*... and of course you need data to learn from*

## and more recently: generative tasks

- rather than predictive (or discriminative) ones
- e.g., image generation, conversational AI, new proteins or materials

# Learning Paradigms

# Supervised Learning

**learning by teacher** → usually rather narrow tasks (passive approach)

## Target Quantity

- **known in training:** labeled samples or observations from past
- to be **predicted** for unknown cases (e.g., future values)

## Features

- input information that is
- correlated to target quantity
  - known at prediction time



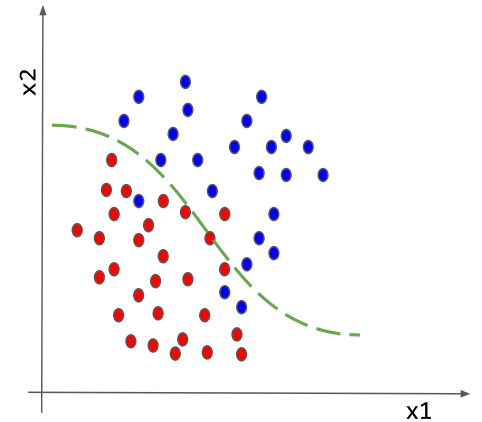
## Example: Spam Filtering

*Classify emails as spam or no spam*

use accordingly **labeled**  
**emails as training set**

use information like  
**occurrence of specific**  
**words or email length**  
as **features**

**features  $x_1$  and  $x_2$**   
**spam, no spam**

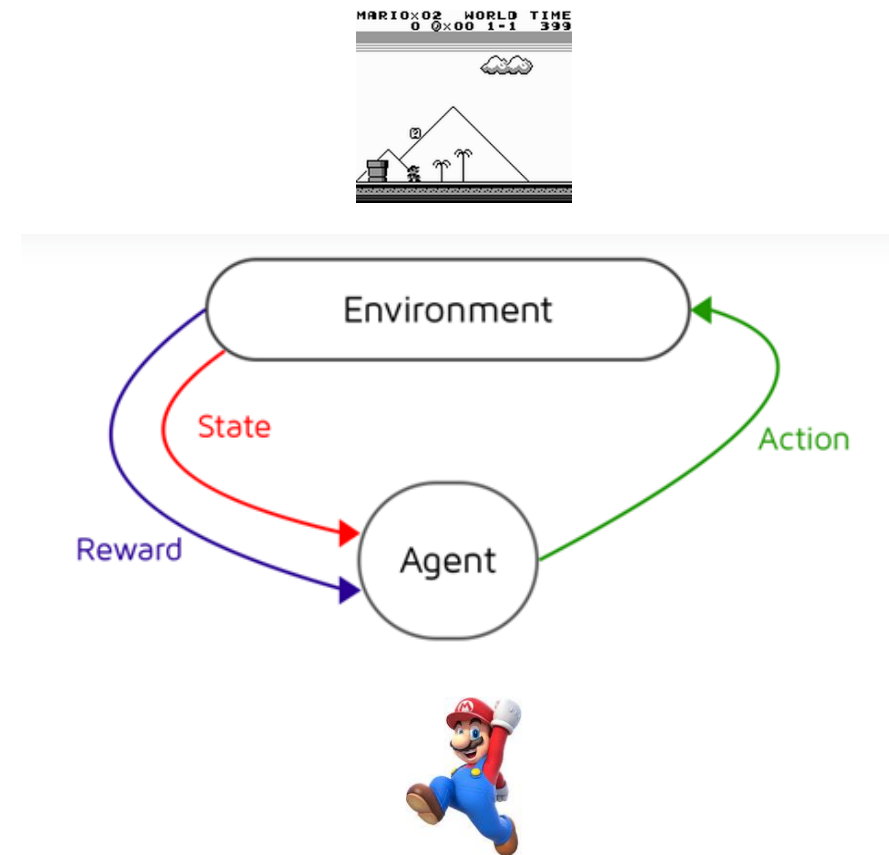


# Reinforcement Learning

## learning by trial-and-error

goal-based approach → active and more generic than supervised learning (but sparse reward signals)

- receiving feedback from the environment
- no supervision
- sequential decision making (delayed rewards)
- exploration and exploitation



# Unsupervised Learning

## learning by observation

no target information → kind of “vague”  
pattern recognition (but plenty of data)

## self-supervised:

- input-output mapping like supervised learning
- but generating labels itself from input information
- learning of semantic feature representations
- e.g., word2vec, BERT, GPT

## How Much Information is the Machine Given during Learning?

- ▶ **“Pure” Reinforcement Learning (cherry)**
  - ▶ The machine predicts a scalar reward given once in a while.
  - ▶ **A few bits for some samples**
- ▶ **Supervised Learning (icing)**
  - ▶ The machine predicts a category or a few numbers for each input
  - ▶ Predicting human-supplied data
  - ▶ **10→10,000 bits per sample**
- ▶ **Self-Supervised Learning (cake génoise)**
  - ▶ The machine predicts any part of its input for any observed part.
  - ▶ Predicts future frames in videos
  - ▶ **Millions of bits per sample**

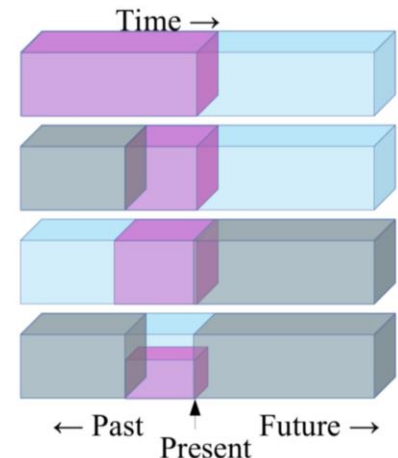


© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future

59

## Self-Supervised Learning

- ▶ Predict any part of the input from any other part.
- ▶ Predict the **future** from the **past**.
- ▶ Predict the **future** from the **recent past**.
- ▶ Predict the **past** from the **present**.
- ▶ Predict the **top** from the **bottom**.
- ▶ Predict the **occluded** from the **visible**
- ▶ **Pretend there is a part of the input you don't know and predict that.**



© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future

58

# Example for Unsupervised Learning

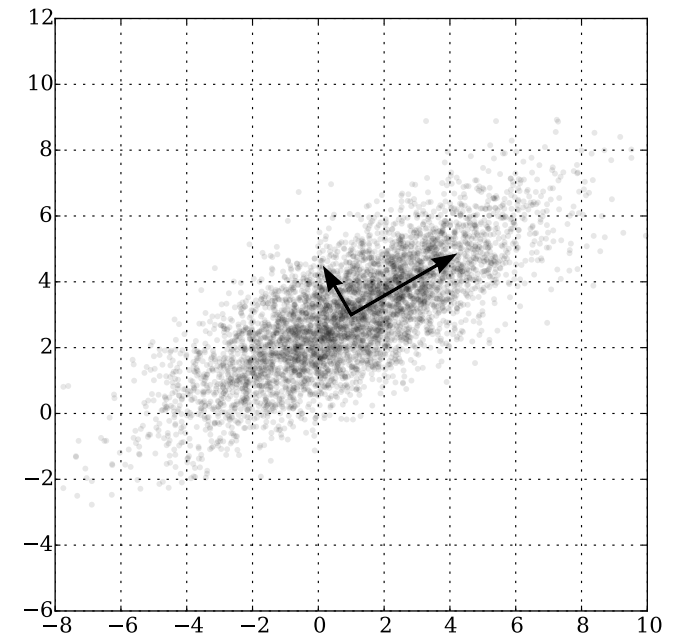
dimensionality reduction by principal component analysis (PCA)

using only first few principal components (eigenvectors of data's covariance matrix)

intuition: fitting  $p$ -dimensional ellipsoid to data

- axes representing principal components
  - large axis  $\rightarrow$  high variance, small axis  $\rightarrow$  low variance
  - successively choose directions of maximum variance
- $\rightarrow$  account for as much variability (uniqueness) of data set as possible

often used as lower-dimensional features in other (supervised) methods




from wikipedia



# Fitting / Statistical Learning

# Notation

- random variable:  $X$
  - vector of  $p$  random variables  $X_j$  (features):  $\mathbf{X}$
  - observation of random variable  $X$ :  $x$
  - matrix of  $n$  observations of  $p$  features  $x_{ij}$ :  $\mathbf{X}$
  - vector of observations:  $\mathbf{x}$ 
    - vector of  $n$  observations  $x_i$ :  $\mathbf{x}$
    - vector of observation of  $p$  features  $x_j$ :  $\mathbf{x}$
  - column vector:  $\mathbf{x}$
  - row vector:  $\mathbf{x}^T$
  - parameter:  $\beta$
  - vector of parameters  $\beta_k$ :  $\boldsymbol{\beta}$
  - probability that  $X$  takes on value  $x_0$ :  $P(X = x_0)$
  - probability distribution:  $p(x) = P(X = x)$
- 
- design matrix

# Supervised Learning Scenario

map inputs to output:  $y = f(\mathbf{x})$  (estimated:  $\hat{f}(\mathbf{x})$ )  
random variables  $Y$  and  $\mathbf{X} = (X_1, X_2, \dots, X_p)$

## classification

- categorical target (e.g., image of cat or not  $\rightarrow y = 0$  or  $y = 1$ )
- predict probability to belong to specific class

## regression

- real-valued target
- $Y \in [0, \infty)$  (e.g., demand forecasting) or  $Y \in (-\infty, \infty)$



... ML ...



ML domain:  
no deterministic dependencies  
between input and output

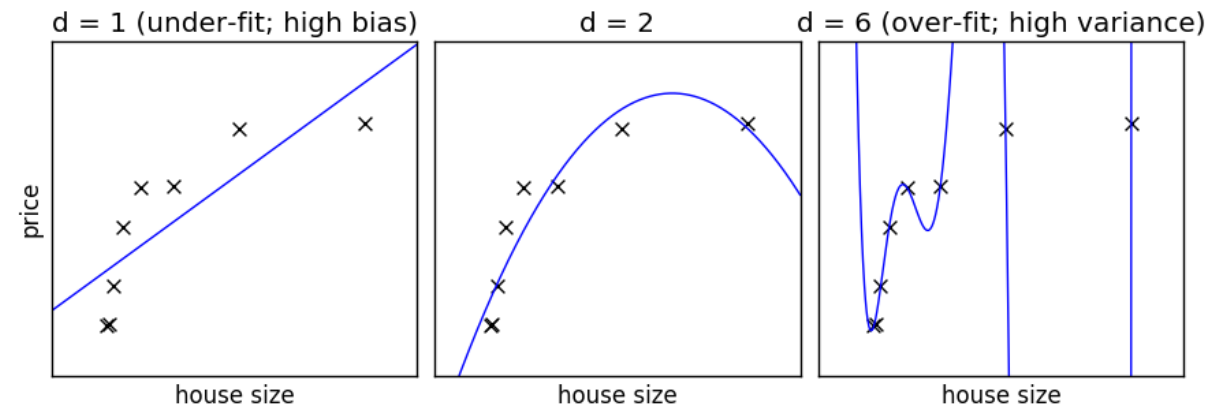
# Curve Fitting / Parameter Estimation

fit train data set of  $(y_i, x_i)$  pairs

→ minimization of cost function

- e.g., least squares method, maximum likelihood estimation
- usually many dimensions (features  $X_j$ )

d: degree of fitted polynomial → number of parameters



from scikit-learn documentation

apply learned statistical dependencies from training to new test data set

different  $(y_i, x_i)$  pairs considered as random samples of underlying data-generating process (i.i.d. assumption), for both train and test data sets

# Generalization

core of ML:

**empirical risk minimization** (training error) as proxy for minimizing unknown population risk (test error, aka generalization error or out-of-sample error)

generalization gap: difference between test and training error

- **interpolation** to unencountered samples from training environment
- **extrapolation** to testing conditions differing from training environment (aka out-of-distribution)

curse of dimensionality: *“learning in high dimensions always amounts to extrapolation”*

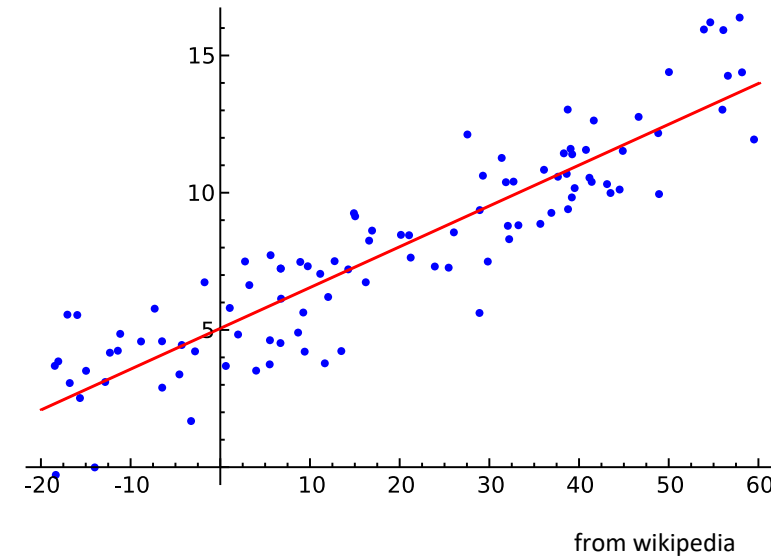
but reality is friendly: most high-dimensional data sets reside on lower-dimensional manifolds (manifold hypothesis) → enabling effectiveness of ML

→ need for appropriate inductive bias

# Generalized Linear Models

# Linear Regression

$$y_i = \alpha + \sum_{j=1}^p \beta_j x_{ij} + \varepsilon_i \quad (\text{model})$$



$y$ : dependent variable / target

$\mathbf{x}$ :  $p$  independent variables / features

$\alpha, \boldsymbol{\beta}$ :  $p + 1$  parameters

$\varepsilon$ : error term / statistical noise

} vector ( $\mathbf{y}$ ) or matrix ( $\mathbf{X}$ ) of given data

→ to be fitted

reflects assumed data distribution (here: Gaussian with same variance  $\sigma^2$  for all samples)

- $\mathbf{X}$  and  $Y$  jointly distributed random variables
- $\hat{f}(\mathbf{x})$ : predict, e.g., conditional mean of conditional density function  $p(y|\mathbf{x})$

↑  
depending on used loss function

(conditional mean for squared loss of least squares method)

# Linear Regression

fit:  $\hat{f}(\mathbf{x}_i)$

$$y_i = \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij} + \varepsilon_i$$

predict:

$$\hat{y}_i = E[Y|\mathbf{X} = \mathbf{x}_i] = \hat{f}(\mathbf{x}_i) = \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij}$$

Gaussian

$$p(y|\mathbf{x}_i) = \mathcal{N}(y; \hat{y}_i, \hat{\sigma}^2)$$

Gaussian      mean      variance

to be estimated:

- $\hat{\alpha}, \hat{\beta}$

$$\rightarrow \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(\mathbf{x}_i))^2$$

(approximating assumed true  $\alpha, \beta, \sigma$ )



# Multiplicative Model

- count data:  $Y \in [0, \infty)$
- $Y$  follows Poisson (or negative binomial / Poisson-gamma) distribution

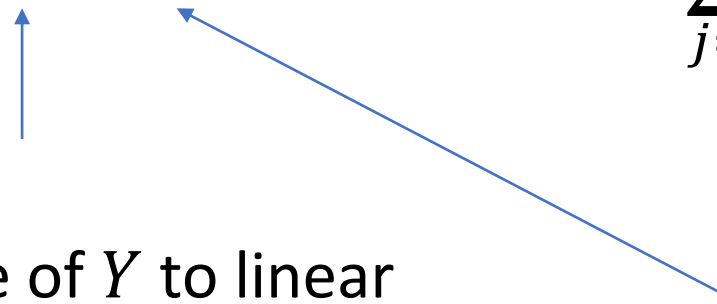
log-linear model (Gaussian errors in fit, Poisson with mean  $\hat{y}_i$  predicted):

$$\log(\underbrace{E[Y|\mathbf{X} = \mathbf{x}_i]}_{\hat{y}_i}) = \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij}$$

↑  
single parameter

- further advantage: usually multiplicative effects for count data, i.e., proportional (small effects for small counts, large effects for large counts)

# Scheme of Generalized Linear Models

$$g(E[Y|X = \mathbf{x}_i]) = \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij}$$


link function  $g$ :

- linking range of  $Y$  to linear predictor
- canonical forms for different  $Y$  distributions (e.g., log for Poisson, identity for Gaussian → linear regression)

$Y$  following probability distribution from exponential family (e.g., Poisson or Gaussian)

# Classification: Logistic Regression

- predict probability  $p_i$  for  $y = 1$  respectively  $y = 0$  for each sample
- link function: logit (log-odds)
- $Y$  following Bernoulli distribution

$$\begin{aligned}\text{logit}(E[Y|\mathbf{X} = \mathbf{x}_i]) &= \ln\left(\frac{p_i}{1 - p_i}\right) \\ &= \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij}\end{aligned}$$

# Toward Non-Linear Models

# Generalized Additive Models

blending of Generalized Linear Models and additive models

$$g(E[Y|\mathbf{X} = \mathbf{x}_i]) = \hat{\alpha} + \sum_{j=1}^p \hat{h}_j(x_{ij})$$

smooth functions

- potentially non-parametric form
- describe non-linear effects
- estimated, e.g., via backfitting algorithm

extension: add interaction terms between different features, e.g.,  $\mathbf{X}_3$  and  $\mathbf{X}_4$

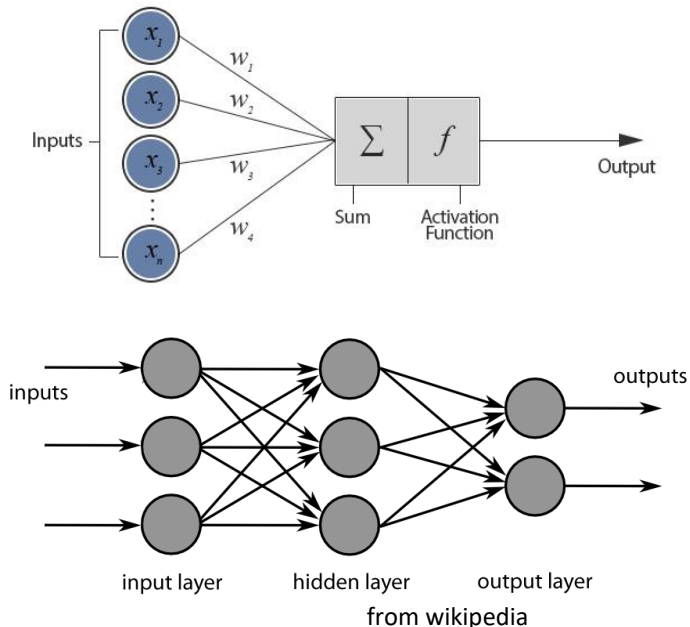
[Cyclic Boosting](#)

# Algorithmic Families and Linear Building Blocks

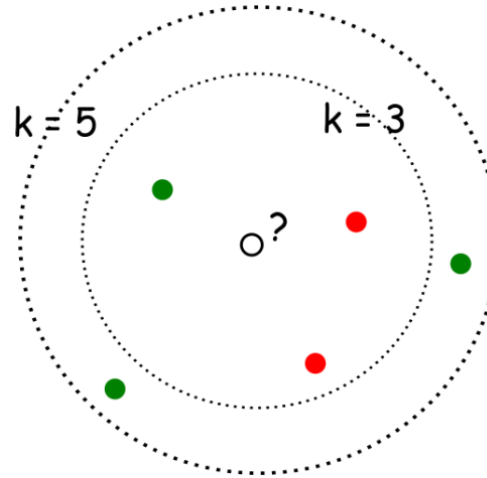
## linear (parametric) models

- linear regression
- Generalized Linear Models
- Generalized Additive Models

**neural networks:** non-linear just by means of activation functions



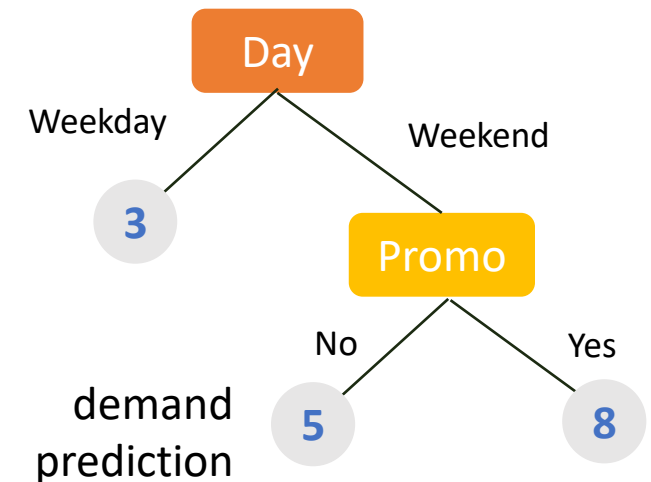
## nearest neighbors (local methods, instance-based learning) – non-parametric models



with  $k = 3$ , ●  
with  $k = 5$ , ●

**kernel/support-vector machines:** linear model (maximum-margin hyperplane) with kernel trick

## decision trees



**often used in ensemble methods**

- bagging: random forests
- boosting: gradient boosting

At its heart, all the diverse statistical learning methods are reflections of the **same underlying concept**, and just differ in their applicability for different use cases.

(need to find method with best inductive bias for the task at hand → generalization capability)

# ML Workflow



# Modeling

## **extract features**

- help the ML algorithm to better understand the data
- impose assumptions hard to discover in the raw data

## **choose ML algorithm**

- from open-source libraries like scikit-learn or pytorch, rarely write an own one
- many different algorithms available, differently suited for given task

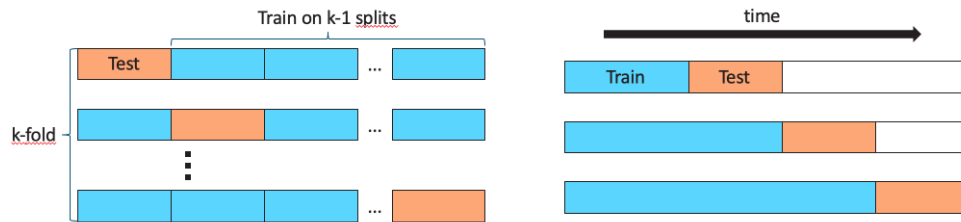
## **execute hyperparameter tuning**

- variety of different forms
- model settings not all automatically adjusted by the machine

# Evaluation

## test structure

### cross-validation



decide on acceptance of model changes by means of accuracy measure: improved model vs baseline (current best)

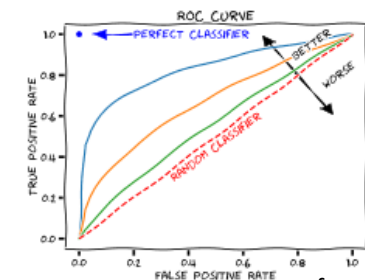
measure (out-of-sample) accuracy of predictions  
(loss function in training as proxy of this)

### regression

- point estimate: absolute (MAD, MSE, ...) or relative (MAPE, ...) metrics
- full probability distribution: [a bit tricky](#)

### classification

ROC curve (true and false positive rates)

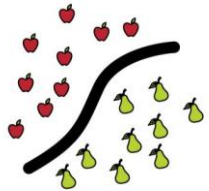


# MACHINE LEARNING

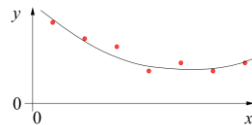
training target available  
(labeled or past data)

## SUPERVISED

### CLASSIFICATION



### REGRESSION

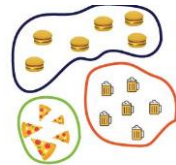


learning by teacher  
(high-dimensional curve fitting)

data not labeled  
in any way

## UNSUPERVISED

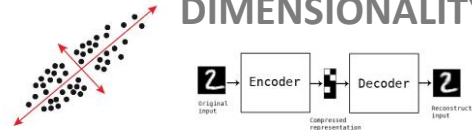
### CLUSTERING



### ASSOCIATION



### DIMENSIONALITY REDUCTION



learning by observation  
(pattern recognition)

no supervision, but goal-based  
interaction with environment

## REINFORCEMENT LEARNING

### LEARN STATE OR ACTION VALUES

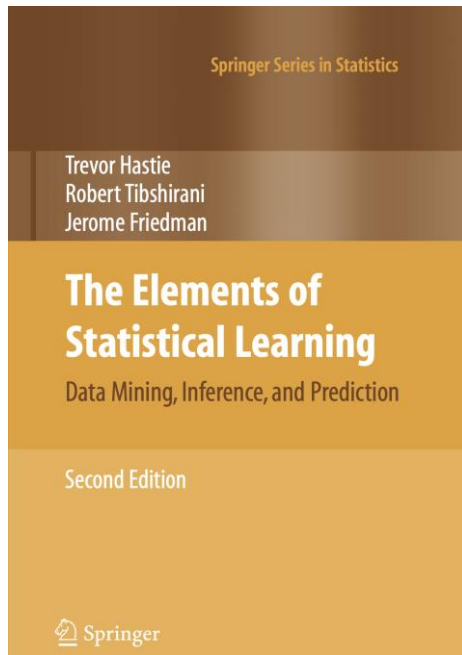
### LEARN POLICY DIRECTLY



learning by trial-and-error  
(sequential decision making)

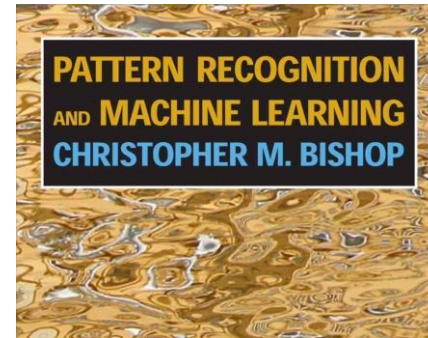
# Literature

nice book on the foundations of ML  
(relevant for the whole course):

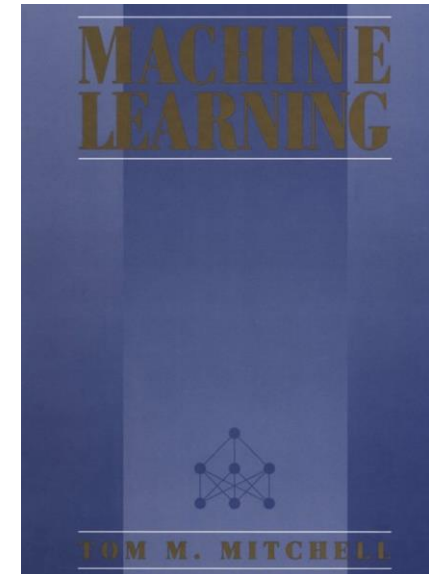


[Hastie](#)

other general overviews:



[Bishop](#)



[Mitchell](#)

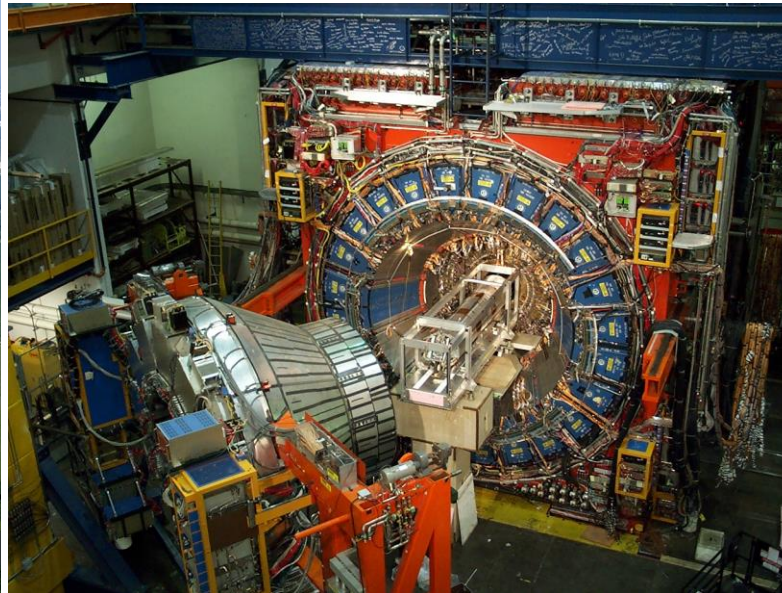
# Scientific Application: ML in Particle Physics

example: classification of decay signatures in particle colliders

Tevatron accelerator at Fermilab



CDF detector at Tevatron



charmed baryon signals filtered out of background

