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

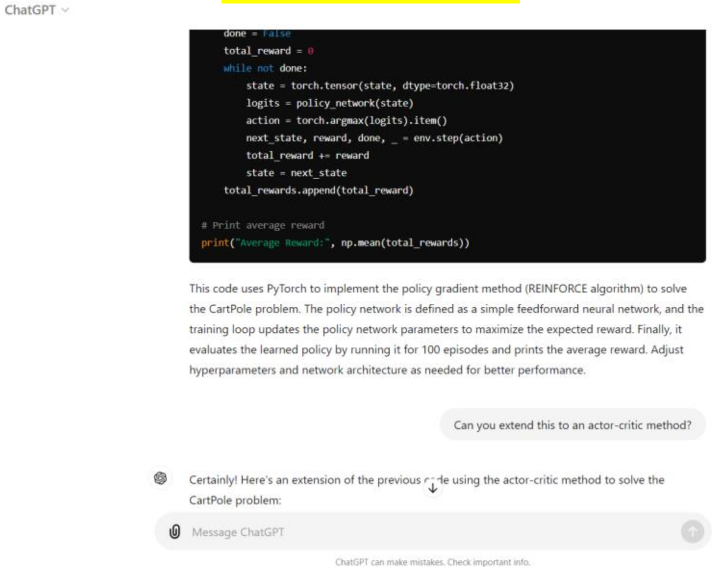
schedule of lectures

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

Classic ML vs Deep Learning

Generative vs Predictive Models

text generation



ChatGPT

```
    done = False
    total_reward = 0
    while not done:
        state = torch.tensor(state, dtype=torch.float32)
        logits = policy_network(state)
        action = torch.argmax(logits).item()
        next_state, reward, done, _ = env.step(action)
        total_reward += reward
        state = next_state
    total_rewards.append(total_reward)

# Print average reward
print("Average Reward:", np.mean(total_rewards))
```

This code uses PyTorch to implement the policy gradient method (REINFORCE algorithm) to solve the CartPole problem. The policy network is defined as a simple feedforward neural network, and the training loop updates the policy network parameters to maximize the expected reward. Finally, it evaluates the learned policy by running it for 100 episodes and prints the average reward. Adjust hyperparameters and network architecture as needed for better performance.

Can you extend this to an actor-critic method?

Certainly! Here's an extension of the previous code using the actor-critic method to solve the CartPole problem:

Message ChatGPT

ChatGPT can make mistakes. Check important info.

[ChatGPT]

image synthesis

Prompt: Epic anime artwork of a wizard atop a mountain at night casting a cosmic spell into the dark sky that says "Stable Diffusion 3" made out of colorful energy

[Stable Diffusion 3 — Stability AI]

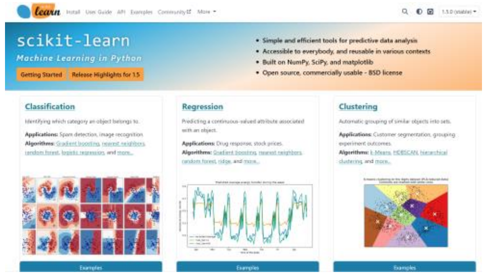
text-to-video

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She...

[Sora | OpenAI]

BERT family

tabular data



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

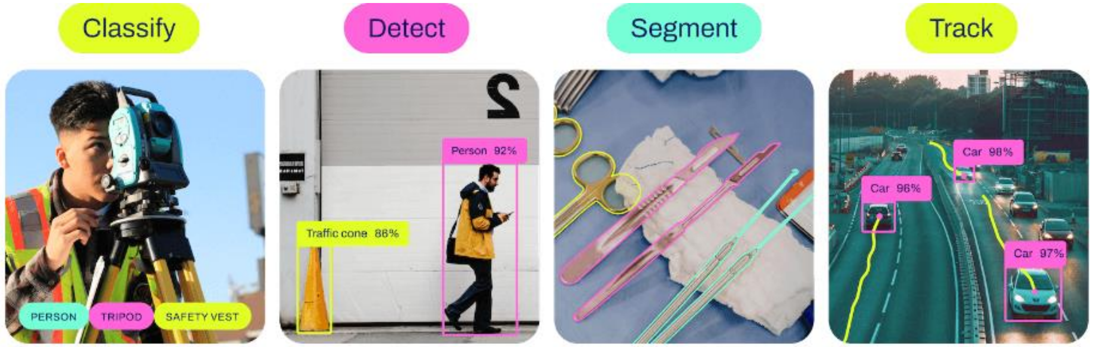
Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

dmlc XGBoost

computer vision



[YOLO]

AI/ML Overview

Main Areas of Artificial Intelligence

from wikipedia

- **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)

agency:
perception – thought – action

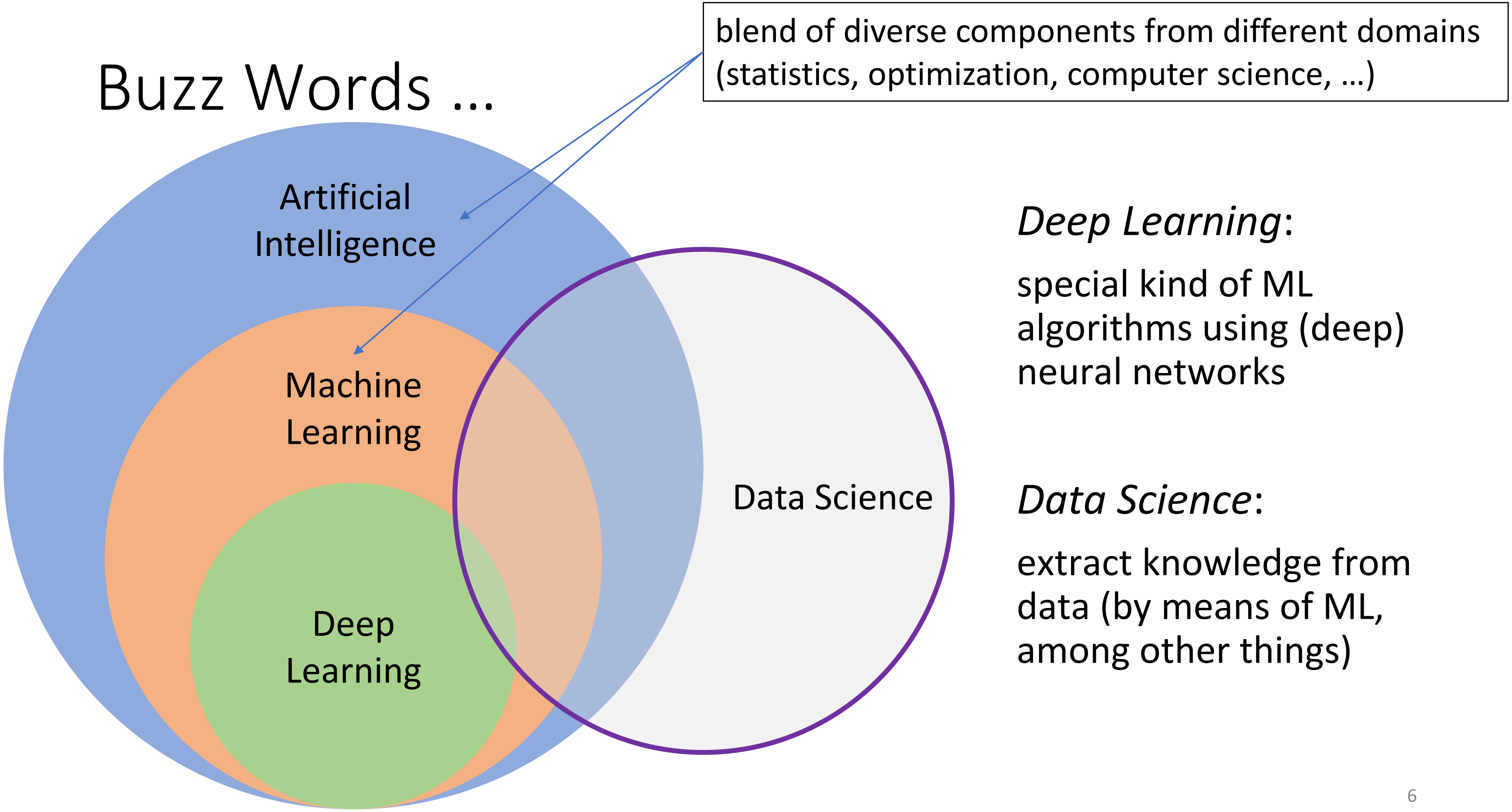
All of these are enabled by one key ingredient:

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

- also: knowledge representation, automated reasoning (first indices in modern Large Language Models)

from *Artificial Intelligence: A Modern Approach* (Russell, Norvig)

Buzz Words …


blend of diverse components from different domains (statistics, optimization, computer science, …)
Artificial Intelligence
Machine Learning
Deep Learning
Data Science

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

from wikipedia

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.

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

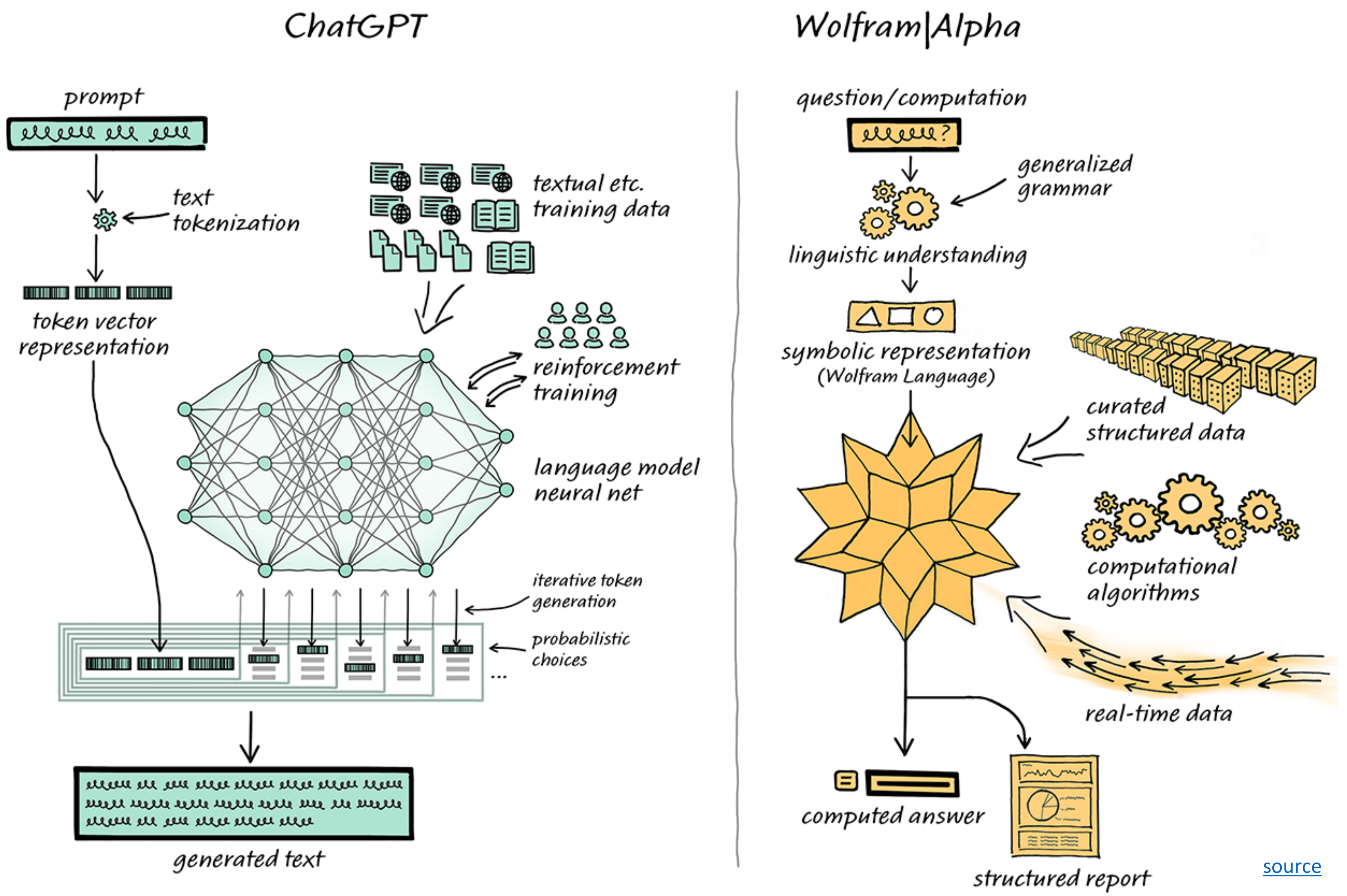
connectionists:
learn from (big) data without prior knowledge

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 (feature engineering for ML models also kind of symbolic knowledge representation)

Hybrid Approach for Language Models?



source

AlphaGeometry

tool usage: LangChain

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)

Most Famous Applications

recommendations

chatbots

ChatGPT Bard

autonomous driving

WAYMO TESLA

translation

DeepL

robotics

OCR

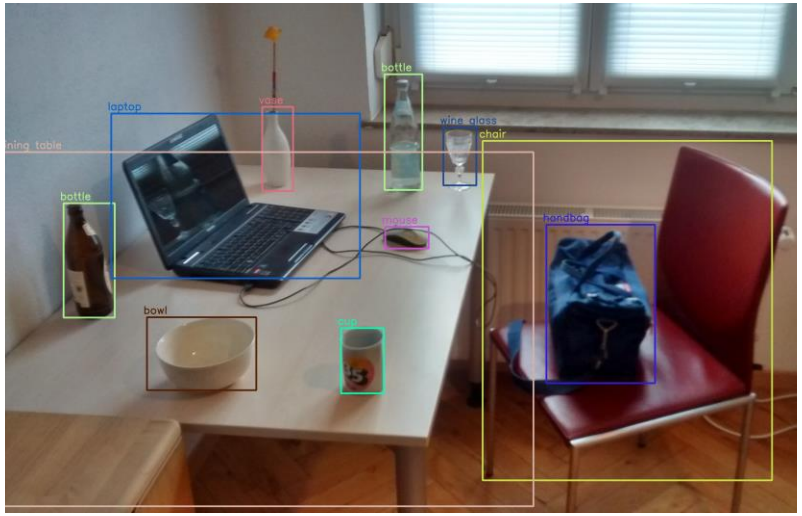
assistants (speech recognition)

and many more …

When to Use ML (= Learning from Data)

automation

too complex for rules



from wikipedia

examples: object recognition, all applications from previous slide

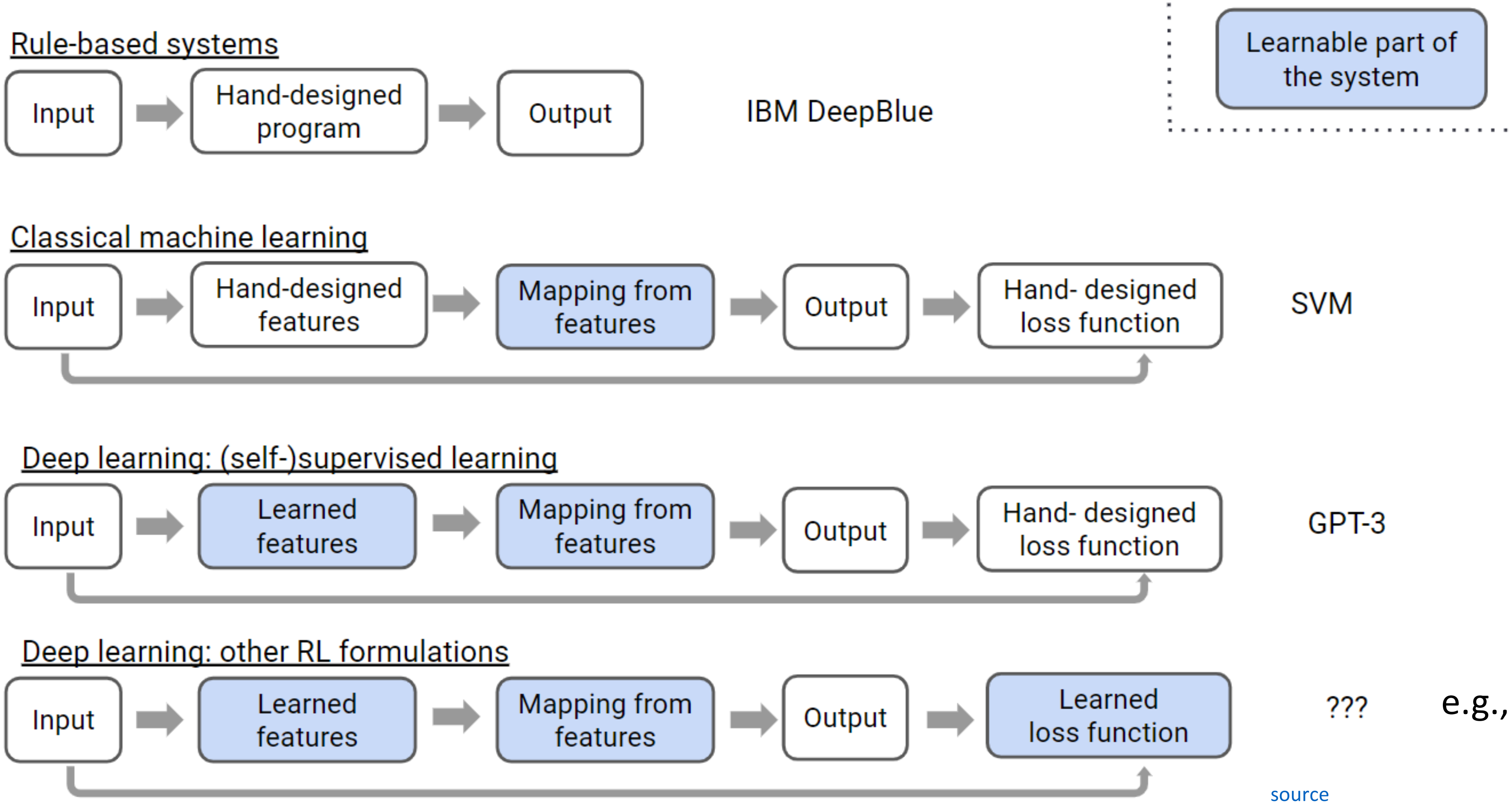
complexity / uncertainty

too complex for humans

examples: protein structure predictions (AlphaFold), demand forecasting

more scientific use cases: medicine (imaging, diagnosis, drug design), particle physics (analysis of collider experiments), material science (material properties and design of new materials), …

Ladder of Generalization



Learnable part of the system

Rule-based systems

Input → Hand-designed program → Output

IBM DeepBlue

Classical machine learning

Input → Hand-designed features → Mapping from features → Output → Hand- designed loss function

SVM

Deep learning: (self-)supervised learning

Input → Learned features → Mapping from features → Output → Hand- designed loss function

GPT-3

Deep learning: other RL formulations

Input → Learned features → Mapping from features → Output → Learned loss function

???

e.g., ChatGPT (RLHF)

source

Generative AI

more recently: generative applications

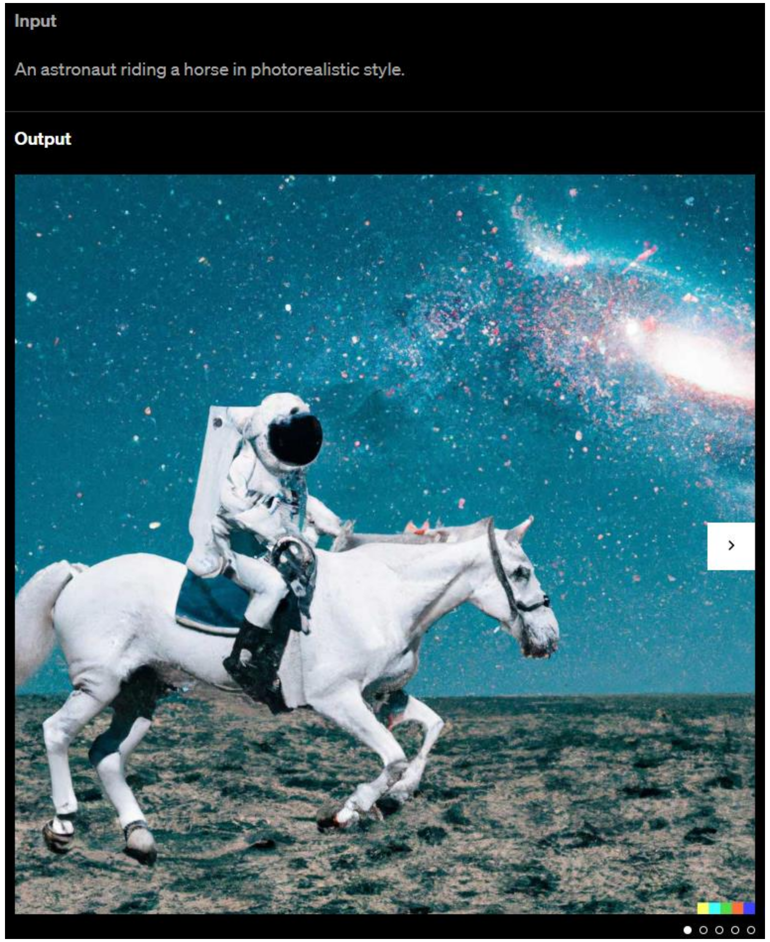
- rather than predictive (or discriminative) ones
- e.g., image generation, conversational AI, new proteins or materials

Depending on the application, there are currently two dominant approaches:

- text generation: large language models (transformer)
- image synthesis: diffusion models

up next: video synthesis → dynamics/physics understanding/simulation

example: DALL-E 2



Input

An astronaut riding a horse in photorealistic style.

Output

Learning Paradigms

Supervised Learning

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

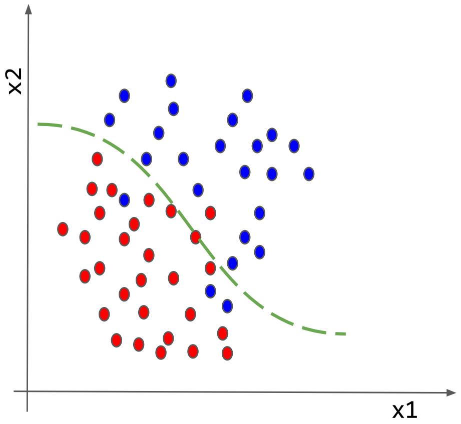
Target Quantity

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

Features

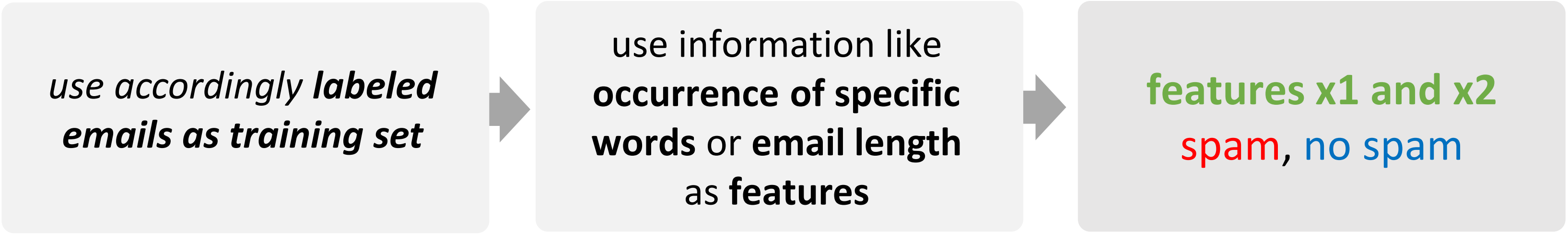
input information that is

- 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***

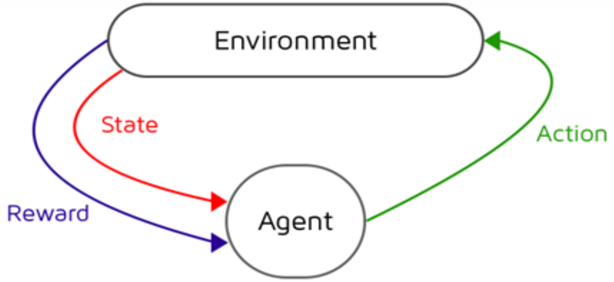
use information like **occurrence of specific words** or **email length** as **features**

features x1 and x2
spam, no spam

Reinforcement Learning

learning by trial-and-error (exploration and exploitation)

- goal-based approach → active and more generic than supervised learning (but sparse reward signals)
- receiving feedback from the environment, no supervision
- formalization of sequential decision making (delayed rewards)



corresponds to search for best action policy to reach a given goal (e.g., win a game)

using learning from examples (data) to guide the search

RL setup usually more difficult (e.g., non-differentiable as a whole) than supervised learning one but RL can be cast as supervised-learning setup: express rewards by more intricate loss function

ML needs lots of training data

Unsupervised Learning

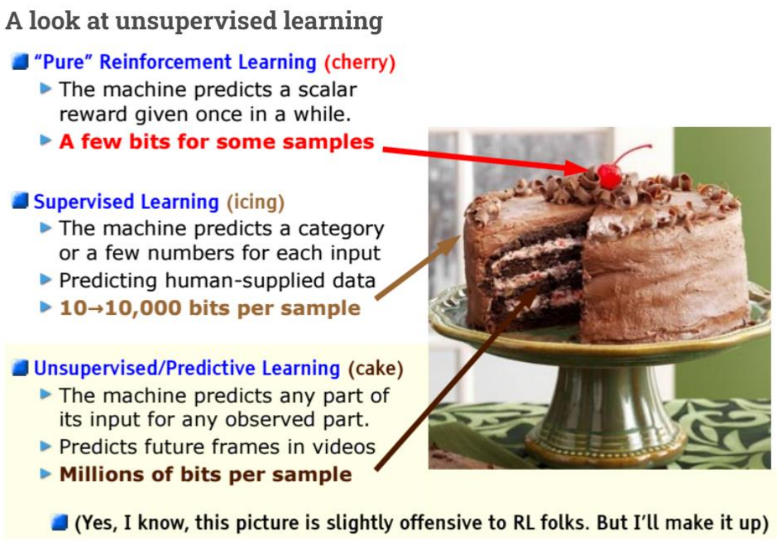
learning by observation

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

can be cast as **self-supervised learning**:

- input-output mapping like supervised learning
- but generating labels itself from input information

generative AI as unsupervised learning: generate variations of training data

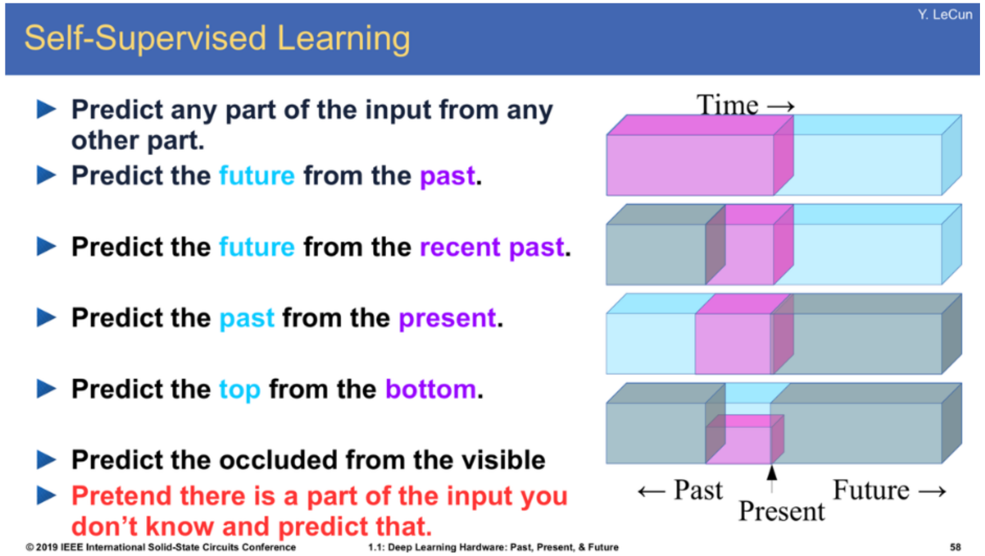


A look at unsupervised learning

- “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**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**
- (Yes, I know, this picture is slightly offensive to RL folks. But I’ll make it up)

Original LeCun cake analogy slide presented at NIPS 2016, the highlighted area has now been updated.

source



Y. LeCun

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.**

Time →
← Past
Present
Future →

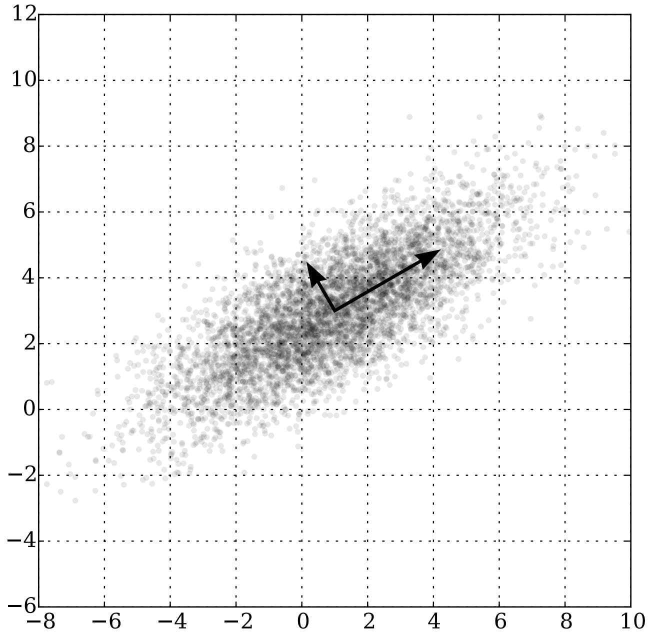
© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future 58

LeCun’s self-supervised learning slide at ISSCC 2019

Example for Unsupervised Learning

dimensionality reduction by principal component analysis (PCA)

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



from wikipedia

intuition: fitting p-dimensional ellipsoid to data

- axes representing principal components
- large axis → high variance, small axis → low variance
- successively choose directions of maximum variance

→ account for as much variability (uniqueness) of data set as possible

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

Fitting / Statistical Learning

Notation

- random variable: X
- vector of p random variables X_j (features): $\boldsymbol{X}$

- observation of random variable X: x
- matrix of n obervations of p features x_{ij}: $\mathbf{X}$
- vector of observations: $\boldsymbol{x}$
    - vector of n obervations x_i: $\boldsymbol{x}$
    - vector of observation of p features x_j: $\boldsymbol{x}$

- column vector: $\boldsymbol{x}$
- row vector: $\boldsymbol{x}^{\mathrm{T}}$

design matrix

- parameter: β
- vector of parameters β_k: $\boldsymbol{\beta}$

- probability that X takes on value x_0: $P(X = x_0)$
- probability distribution: $p(x) = P(X = x)$

Supervised Learning Scenario

... **ML** ...

map inputs to output: $y = f(\boldsymbol{x})$ (estimated: $\hat{f}(\boldsymbol{x})$)

random variables Y and $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)$

ML domain:
no deterministic dependencies
between input and output

classification

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

regression

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

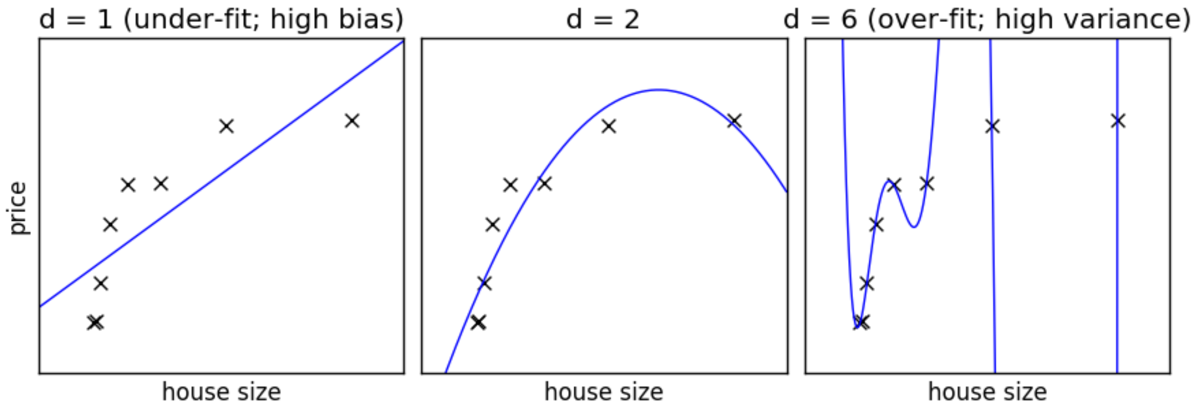
Curve Fitting / Parameter Estimation

fit train data set of $(y_i, \boldsymbol{x}_i)$ pairs

→minimization of cost function

- e.g., least squares method, maximium 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 $(y_i, \boldsymbol{x}_i)$ pairs considered as random samples (whereby $\boldsymbol{x}$ values are taken for granted in discriminative models) of underlying data-generating process (i.i.d. assumption), for both train and test data sets

Generalization

generalization as 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: many features (dimensions) → lots of data needed to densely sample volume

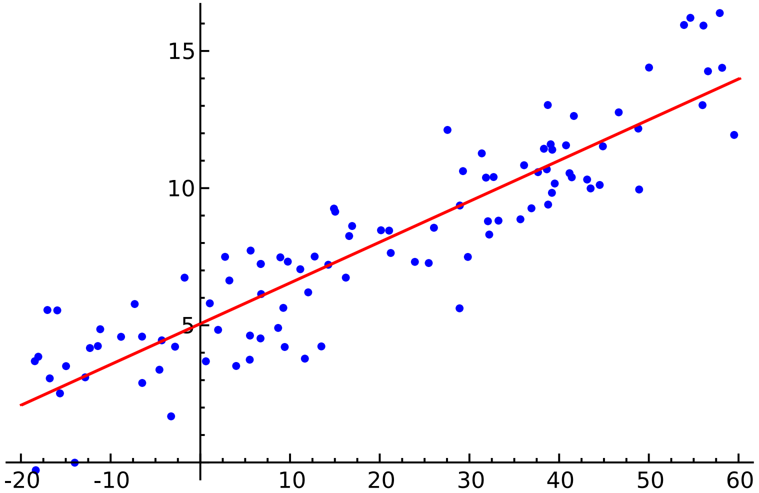
but reality is friendly: most high-dimensional data sets reside on lower-dimensional manifolds (manifold hypothesis) → enabling effectiveness of ML

→ need for appropriate **inductive bias** (different forms: model design, regularization, …)

Generalized Linear Models (GLM)

Linear Regression

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



from wikipedia

y: dependent variable / target
$\boldsymbol{x}$: p independent variables / features

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

$\alpha, \boldsymbol{\beta}$: $p + 1$ parameters → to be fitted

ε: error term / statistical noise reflects assumed data distribution (here: Gaussian with same variance σ^2 for all samples)

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

consider given $\boldsymbol{x}$ values for this

depending on used loss function
(conditional mean for squared loss of least squares method)

Linear Regression

fit:

$$y_i = \overbrace{\hat{\alpha} + \sum_{j=1}^{p} \hat{\beta}_j \, x_{ij}}^{\hat{f}(\boldsymbol{x}_i)} + \varepsilon_i$$

Gaussian

predict:

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

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

Gaussian mean variance

to be estimated:

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

→ $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(\boldsymbol{x}_i) \right)^2$

(approximating assumed true $\alpha, \boldsymbol{\beta}$, σ)

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):

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

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 GLMs

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

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

$$\mathrm{logit}(E[Y|\boldsymbol{X} = \boldsymbol{x}_i]) = \ln\left(\frac{p_i}{1 - p_i}\right)$$

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

Toward Non-Linear Models

Generalized Additive Models (GAM)

blending of GLMs and additive models

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

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., $\boldsymbol{X}_3$ and $\boldsymbol{X}_4$

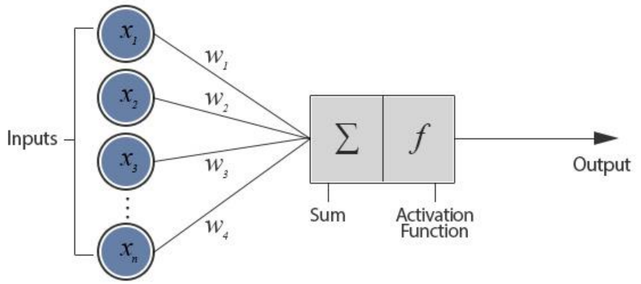
Cyclic Boosting

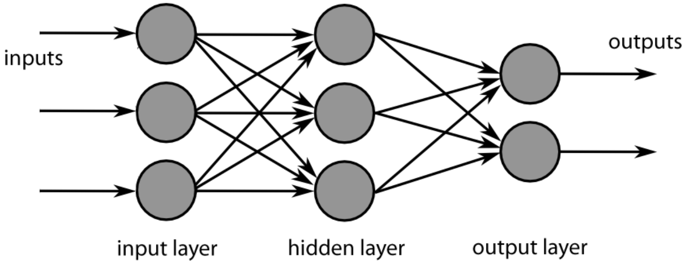
Algorithmic Families and Linear Building Blocks

linear (parametric) models

- linear regression
- GLM
- GAM

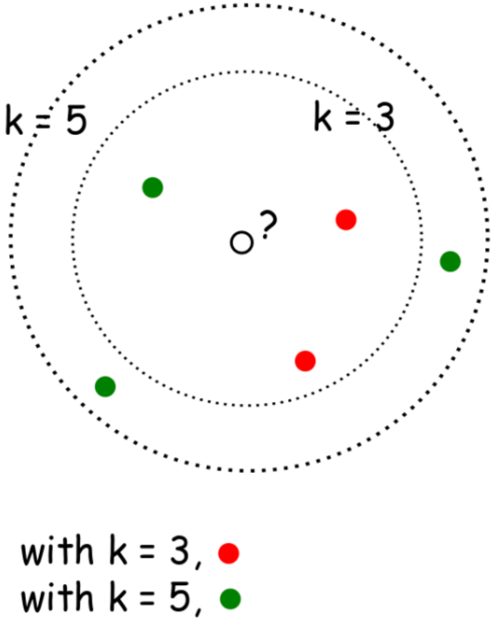
neural networks: non-linear just by means of activation functions





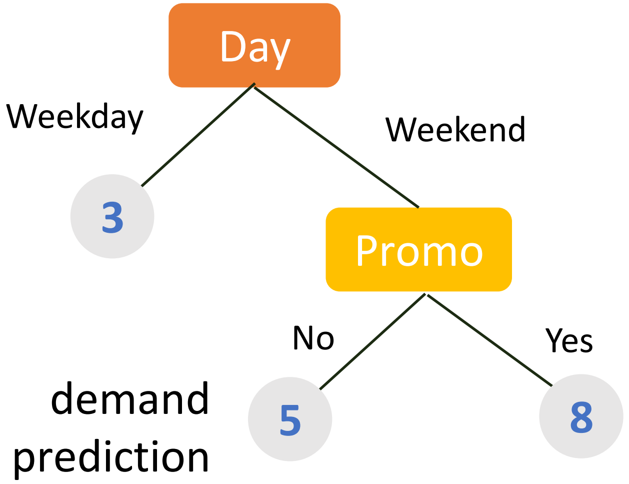
from wikipedia

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



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

decision trees: rule learning



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

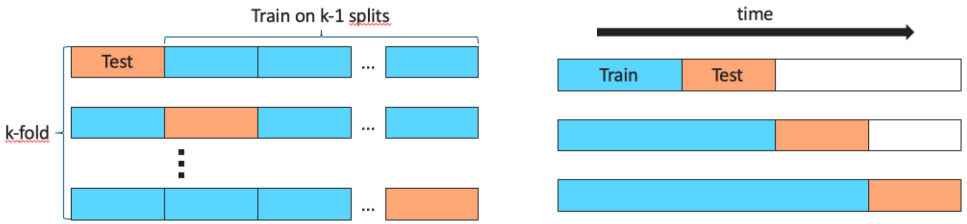
execute hyperparameter tuning

- 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

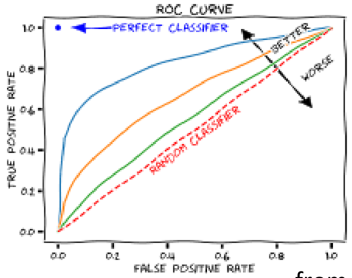
(loss function in training as proxy of this)

regression

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

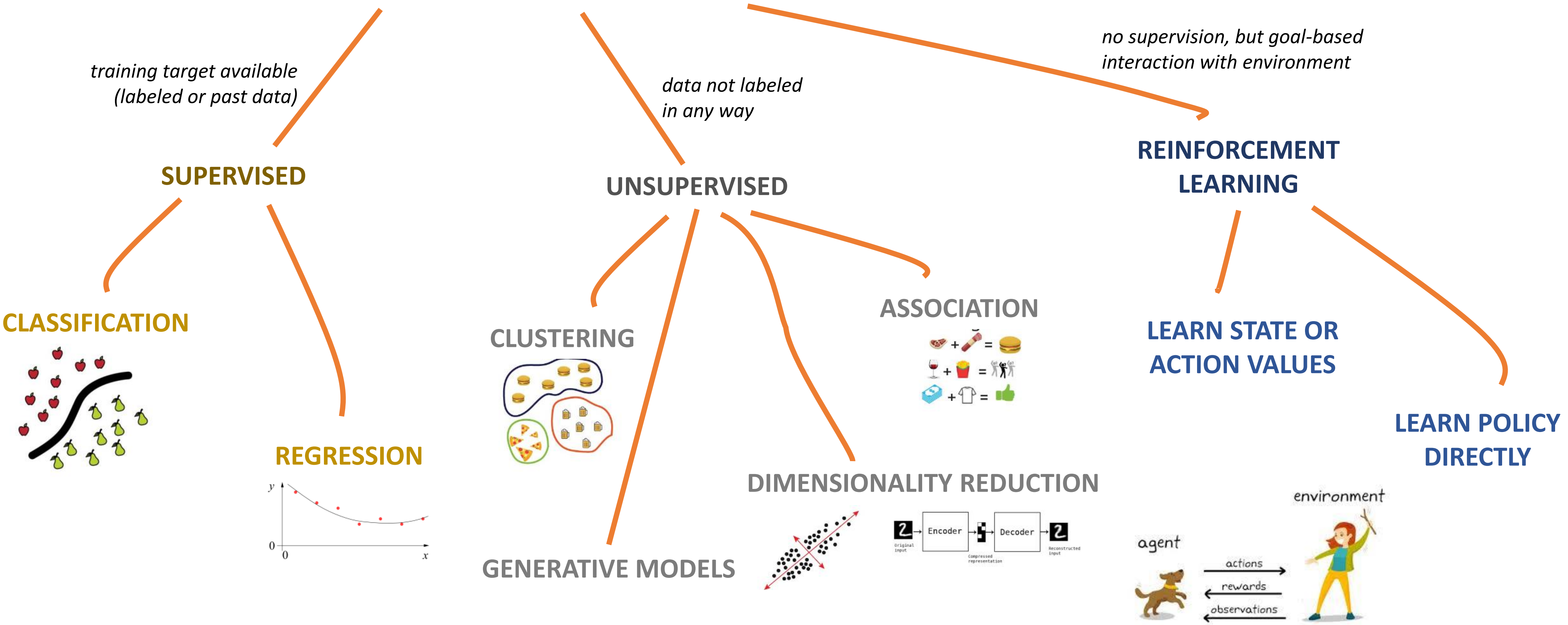
classification

ROC curve (true and false positve rates)



from wikipedia

MACHINE LEARNING



training target available (labeled or past data)

SUPERVISED

CLASSIFICATION

REGRESSION

data not labeled in any way

UNSUPERVISED

CLUSTERING

GENERATIVE MODELS

DIMENSIONALITY REDUCTION

ASSOCIATION

no supervision, but goal-based interaction with environment

REINFORCEMENT LEARNING

LEARN STATE OR ACTION VALUES

LEARN POLICY DIRECTLY

learning by teacher
(high-dimensional curve fitting)

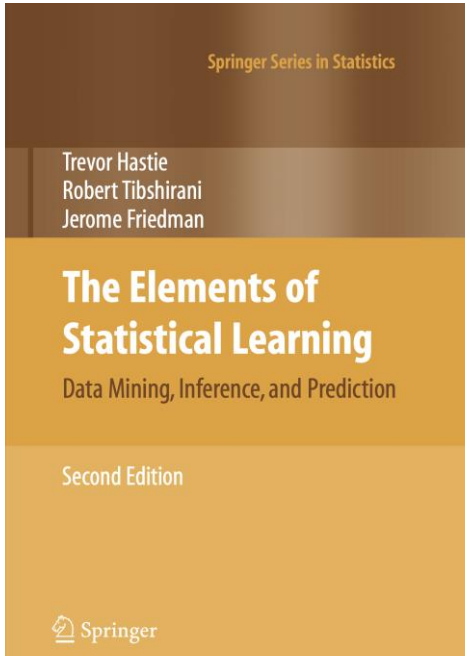
learning by observation
(pattern recognition)

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

unsupervised and reinforcement learning can both be cast as supervised-learning setup

Literature

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



Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

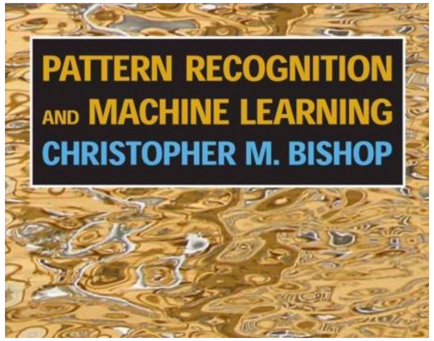
Data Mining, Inference, and Prediction

Second Edition

Springer

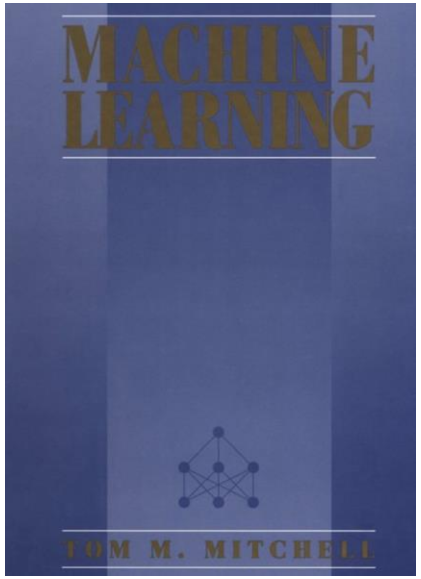
Hastie

other general overviews:



PATTERN RECOGNITION AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

Bishop



MACHINE LEARNING

TOM M. MITCHELL

Mitchell

https://arxiv.org/abs/1105.5995

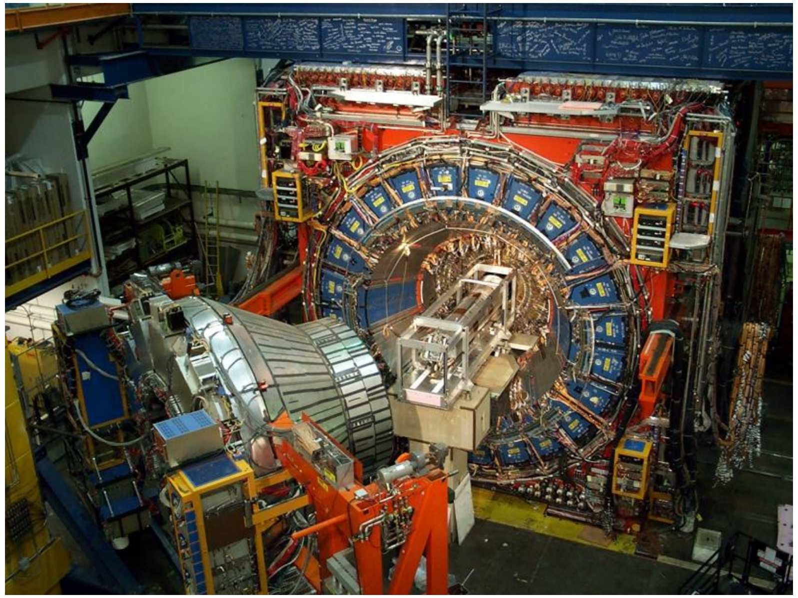
https://arxiv.org/abs/1207.0825

Scientific Application: ML in Particle Physics

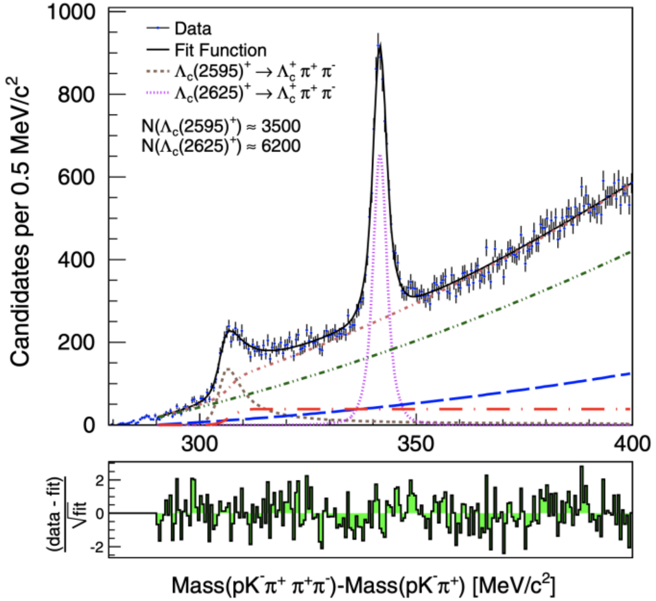
example: classification of decay signatures in particle colliders

Tevatron accelerator at Fermilab

CDF detector at Tevatron



charmed baryon signals filtered out of background


Data
Fit Function
$\Lambda_c(2595)^+ \rightarrow \Lambda_c^+ \pi^+ \pi^-$
$\Lambda_c(2625)^+ \rightarrow \Lambda_c^+ \pi^+ \pi^-$
$N(\Lambda_c(2595)^+) \approx 3500$
$N(\Lambda_c(2625)^+) \approx 6200$
Candidates per 0.5 MeV/c^2
$\frac{(data - fit)}{\sqrt{fit}}$
Mass($pK^-\pi^+\pi^+\pi^-$)-Mass($pK^-\pi^+$) [MeV/c^2]