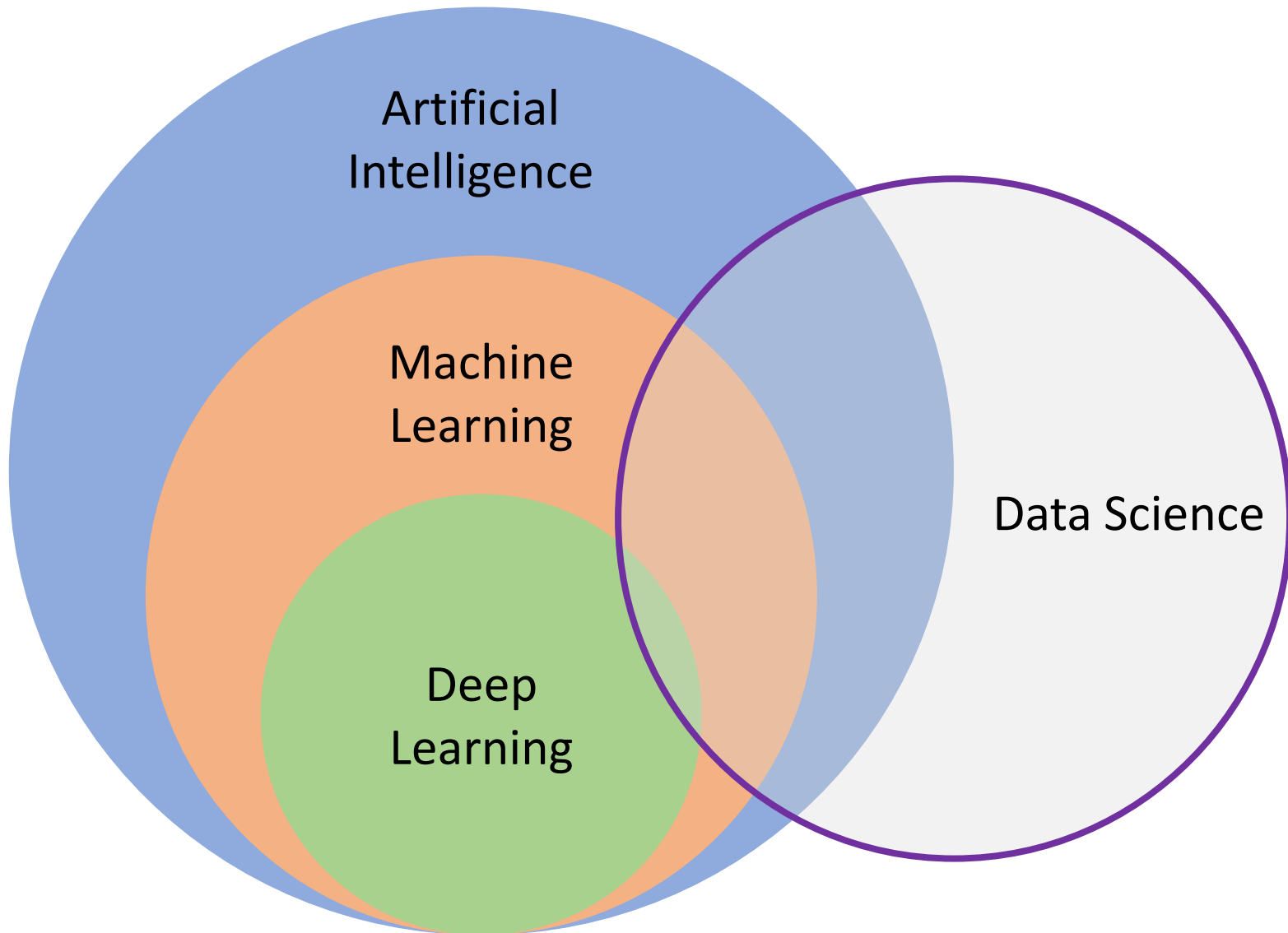


BI and AI

Demand Forecasting, Replenishment, Dynamic Pricing

June 2023



Deep Learning:

special kind of ML
algorithms using (deep)
neural networks

BI / Data Science:

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

Automated Decision Making with AI in Retail

replenishment

avoid waste or out-of-stock situations (lost sales)

pricing

shape demand to maximize of revenue or profit

typically thousands of products and stores:
many individual decisions → **automation crucial**

decisions under uncertainty (random variables, many influencing factors)
difficult for humans → **statistical methods**

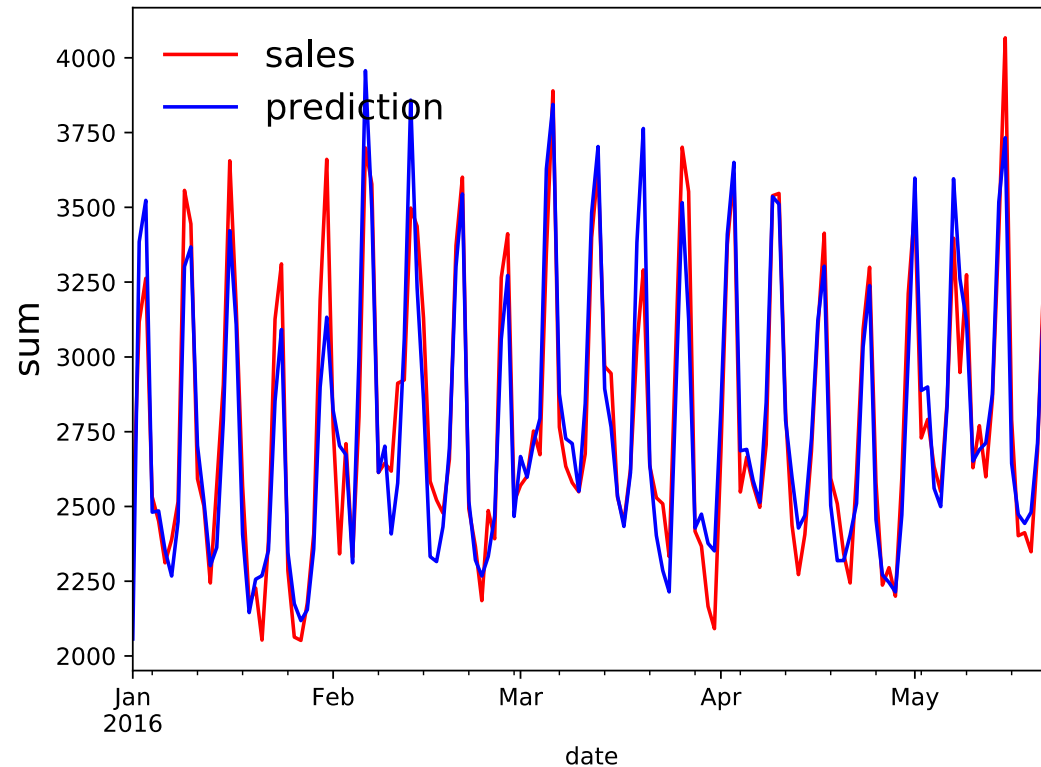
Demand-Driven Decision Making

core component: demand forecasting

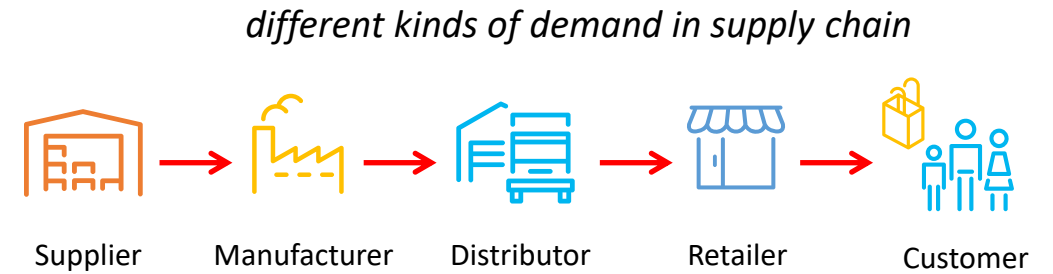
used in subsequent (or entangled) decision making

- ordering / replenishment: minimization of costs for over- or under-fulfillment of demand
- dynamic pricing: shape demand by altering its causes
(other possibility: individual customer targeting)

Demand in Supply Chains

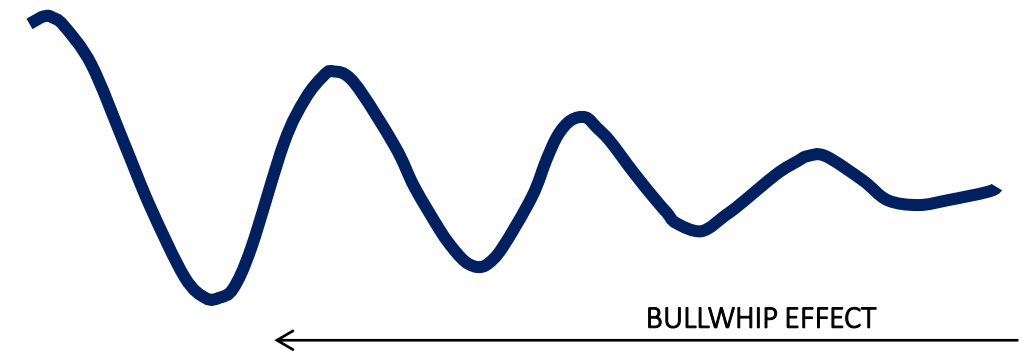


**demand is a time-tagged random variable →
time series regression**



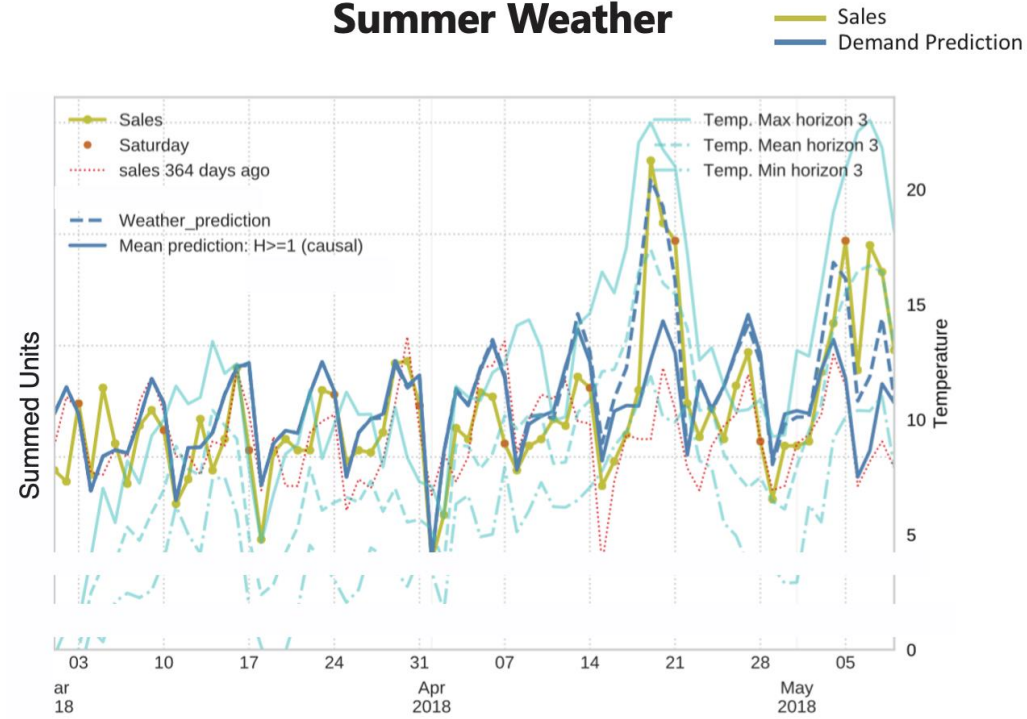
demand as orders

demand as sales

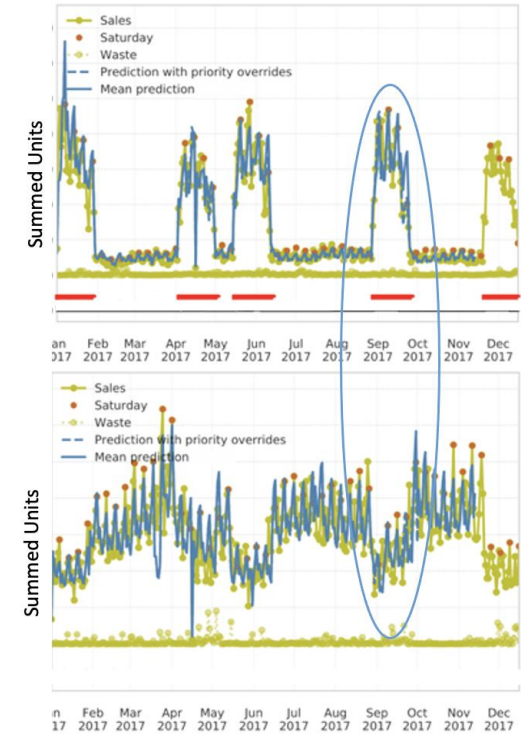


Demand in Retail

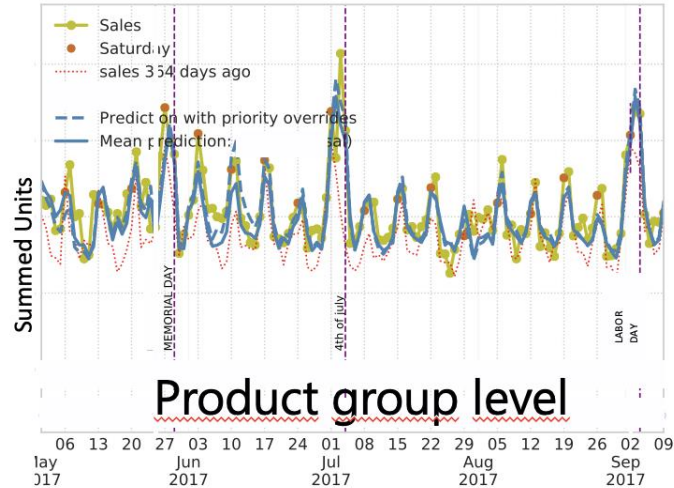
Summer Weather



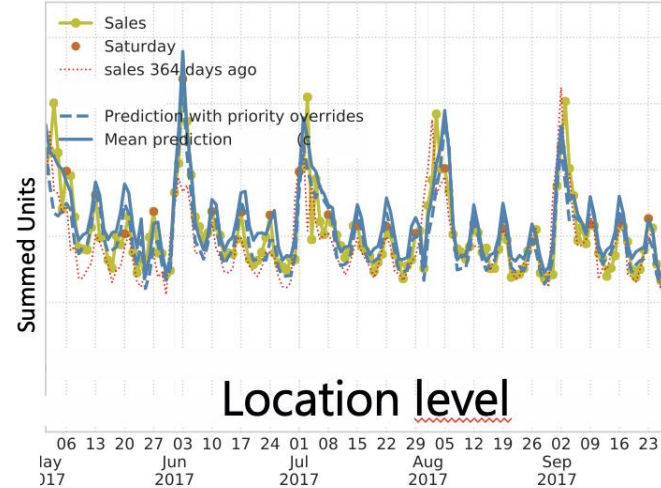
Cannibalization



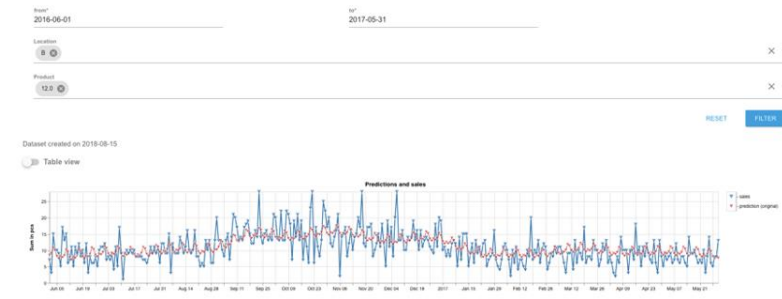
Events



Payday Effect



Seasonality



Time Series Modeling

Types of Time Series Forecasting Information

auto-correlation

- using past values of considered time series as function of lag
- identifying periodicity (seasonality) and trend

exogenous variables

- using correlations of other variables to considered time series
- examples in demand forecasting: price, weather
- such effects lasting for some time create spurious auto-correlation

Traditional Forecasting Methods

direct use of auto-correlation:

- exponential smoothing
 - e.g., exponentially weighted moving average (EWMA): $\hat{y}_t = \alpha \cdot y_{t-1} + (1 - \alpha) \cdot \hat{y}_{t-1}$
 - potentially grouped by important lag like day of week or exogenous variables like promotion
- in general: (S)AR(I)MA(X)
- largely univariate time series models
- multivariate model by stacking: include univariate forecasts (e.g., EWMA) as feature in ML method

decomposition-based approaches:

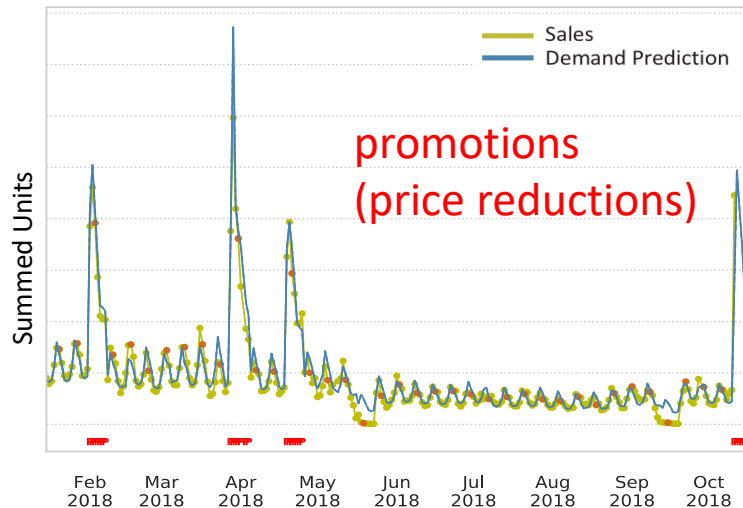
- additive components: trend, seasonality, holidays, others
- curve fitting (ML-like)
- Generalized Additive Models (e.g., multiplicative effects)
- example method: prophet
- natural way for multivariate model: training with several independent time series, distinguished by additional feature (e.g., product ID)

Demand Forecasting with ML

many individual time series to consider

typical retail grocery chain:

- products (items): ~20k
- locations (stores): ~500
- daily/hourly aggregated sales



advantages of ML over traditional univariate time series forecasting:

combined learning on all time series of product-location combinations (rather than separately optimizing individual time series)
→ **reduces variance** by exploiting commonalities

natural consideration of many exogenous variables (prices, promotions, holidays, weather, ...)
→ **reduces bias**

to be noted:

- categorical features important (products and locations → high cardinality)
- mainly multiplicative effects
- demand (approximately) following Poisson (or rather negative binomial) distribution

Beware of (Spurious) Auto-Correlation

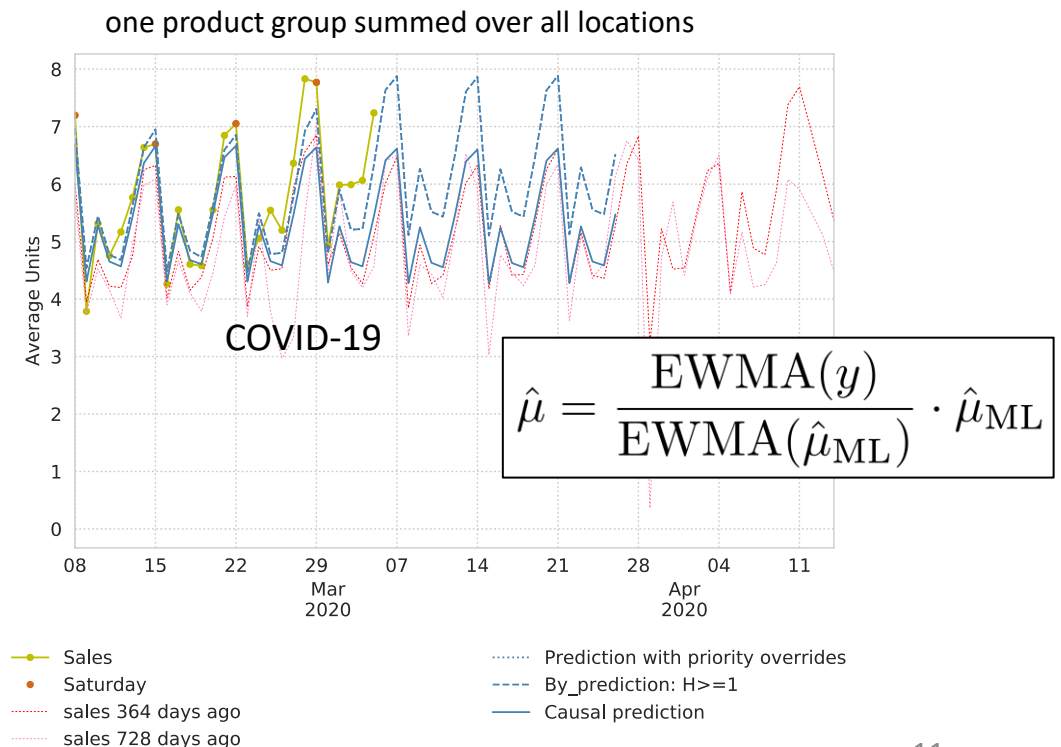
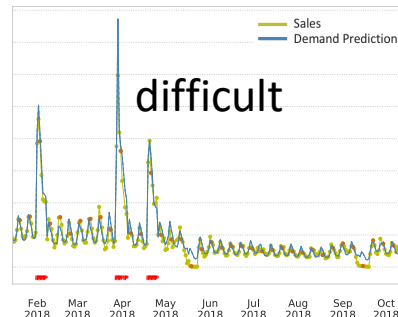
Most forecasting methods rely on lagged target information.

- univariate time series methods
- exponential smoothing features in ML model
- sequence modeling in deep learning (e.g., recurrent neural networks)

However, this makes learning of exogenous effects much harder.

- already blended in target auto-correlation
- delayed effects of short-lived structures/trends

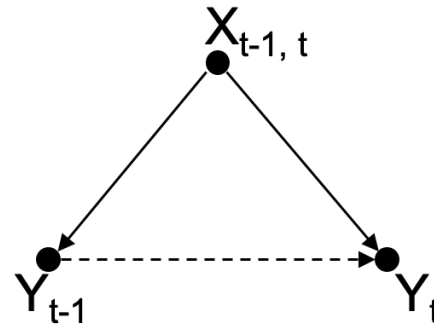
➤ simple solution: no use of target auto-correlation in ML model (but subsequent residual correction to capture missing trends)



Temporal Confounding

auto-correlation often only spurious: common causes at consecutive times

→ no direct causal effects



advantages of learning direct causal effects from exogenous information instead of confounded auto-correlation:

- explainability
- long-term forecasting
- predictability of rare events

Quick Introduction to ML

MACHINE LEARNING

training target available
(labeled or past data)

SUPERVISED

CLASSIFICATION



REGRESSION

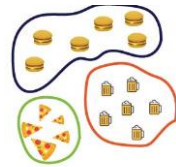


learning by teacher
(high-dimensional curve fitting)

data not labeled
in any way

UNSUPERVISED

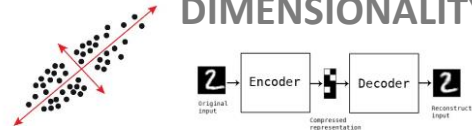
CLUSTERING



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)




... ML ...



Supervised Learning Scenario

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

random variables Y and $\mathbf{X} = (X_1, X_2, \dots, X_p)$  usually many dimensions

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

(i.i.d. assumption: random samples from underlying data-generating process)

then apply learned statistical dependencies to test data set

classification:

categorical target: $y = 0$ or $y = 1$ (e.g., image of cat or not), predict probabilities

regression:

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

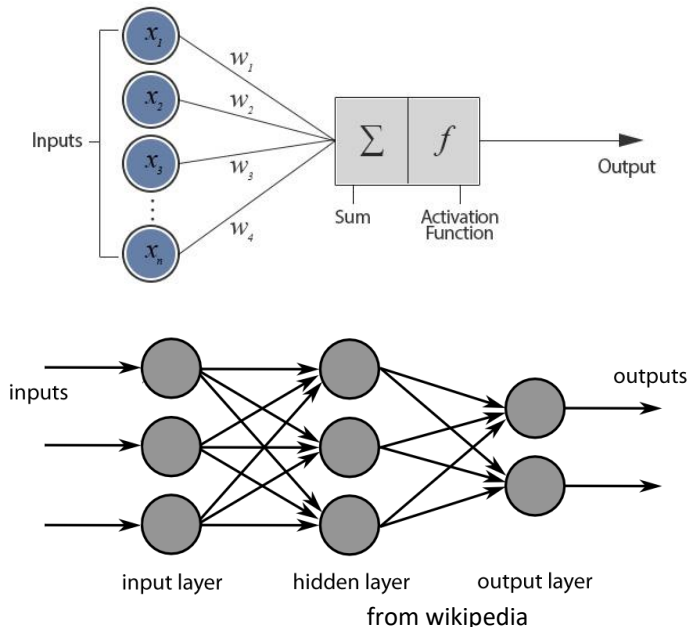
ML domain:
no deterministic dependencies
between input and output

Algorithmic Families

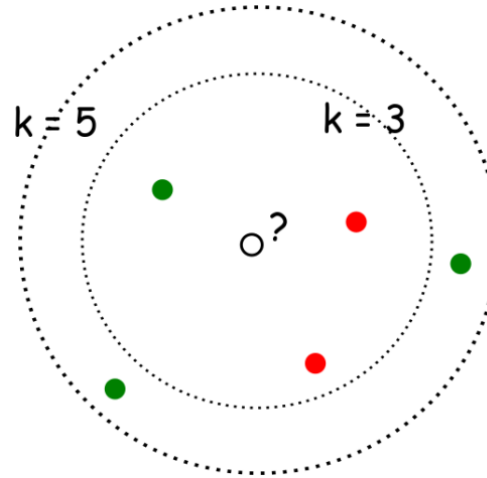
linear (parametric) models

- linear regression
- Generalized Linear Models
- Generalized Additive Models

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



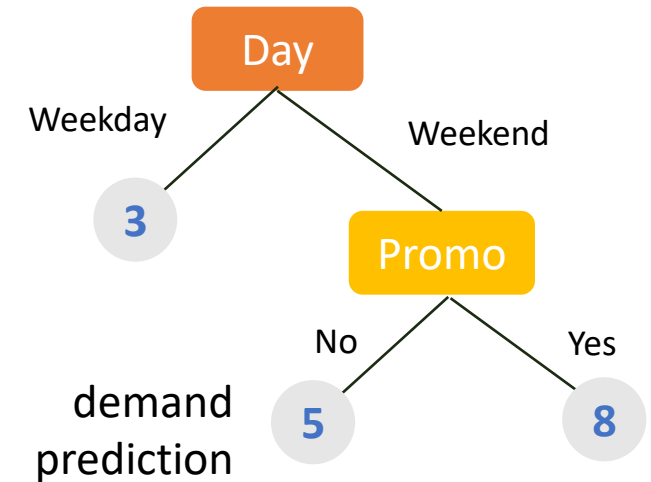
nearest neighbors (local methods, instance-based learning) – non-parametric models



with $k = 3$, ●
with $k = 5$, ●

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

decision trees

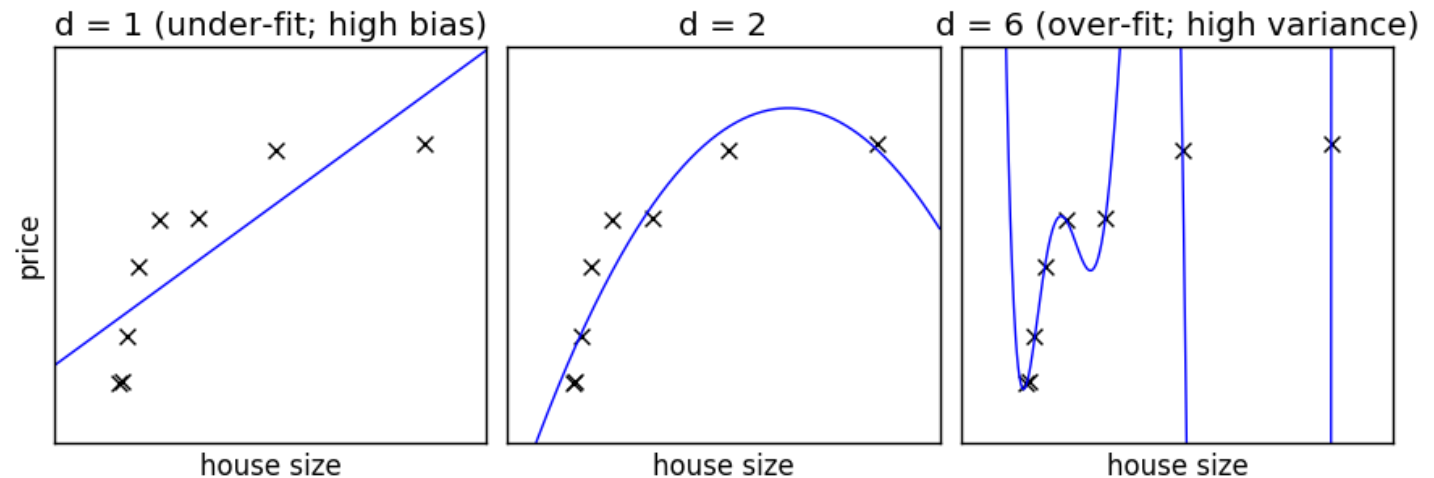


often used in ensemble methods

- bagging: random forests
- boosting: gradient boosting

Common Core of Statistical Learning

curve fitting (e.g., least squares method or maximum likelihood estimation) through **optimization** methods (e.g., gradient descent) with the aim of **generalization** (by means of **regularization** techniques)

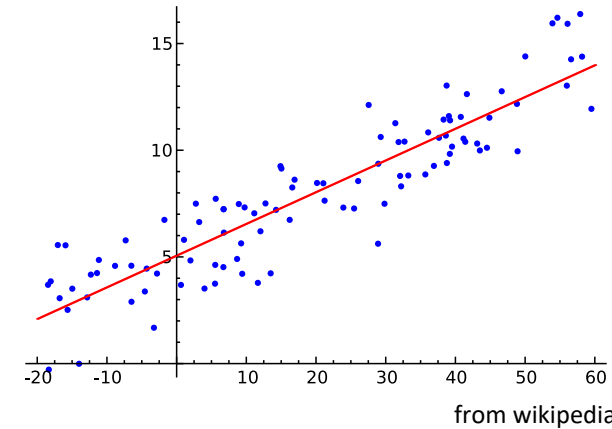


from scikit-learn documentation

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.

Count Data & Multiplicative Effects: Generalized Linear Models

Linear Regression



fit:

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

(model)

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

parameters to be estimated:

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

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

(approximating assumed true α, β, σ)

predict:

$$\hat{y}_i = E[Y | \mathbf{X} = \mathbf{x}_i] = \hat{f}(\mathbf{x}_i)$$

- conditional mean for squared loss of least squares method
- predict arbitrary quantile by means of quantile loss

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

Gaussian

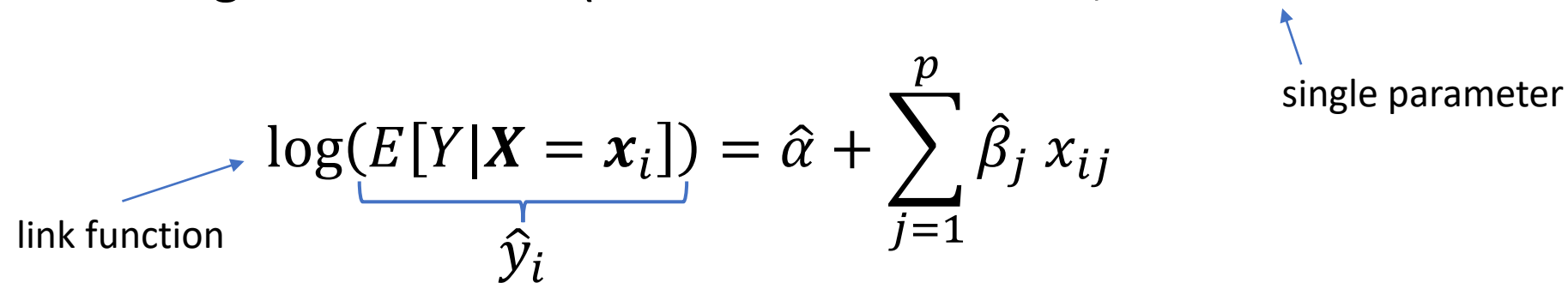
mean

variance
(reflected
by ε_i in fit)

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):



The diagram shows the equation $\log(E[Y|X = x_i]) = \hat{\alpha} + \sum_{j=1}^p \hat{\beta}_j x_{ij}$. A blue arrow labeled "link function" points to the $\log(E[Y|X = x_i])$ term. A blue bracket under the same term points down to \hat{y}_i . A blue arrow labeled "single parameter" points to the $\hat{\beta}_j$ term in the summation.

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

link function

\hat{y}_i

single parameter

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

Sidenote: Classification by Logistic Regression

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

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

Generalized Additive Models

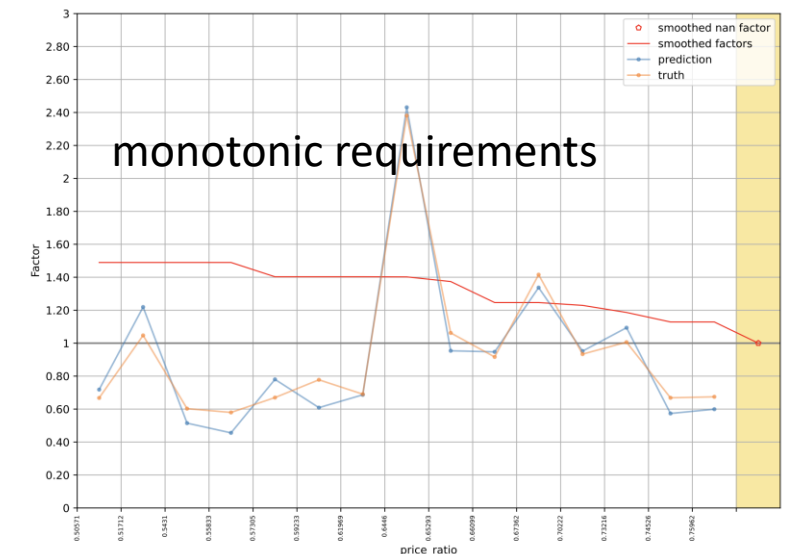
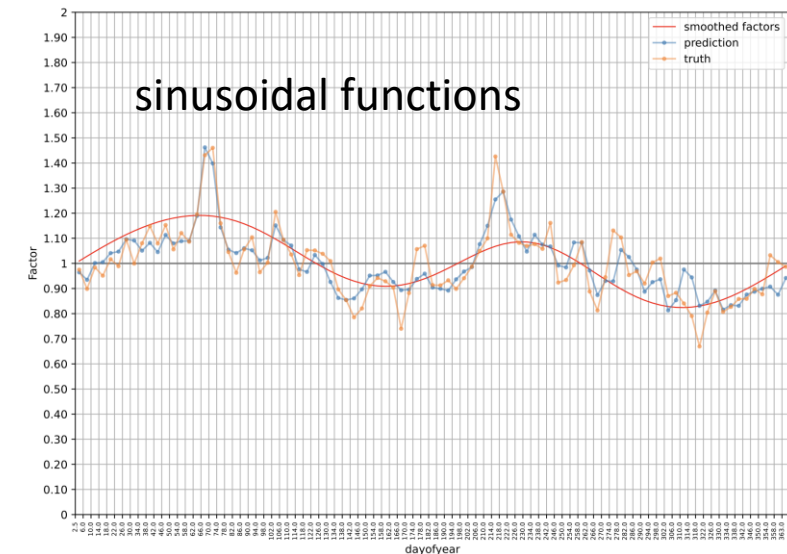
blending of Generalized Linear Models and additive models

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

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., different holiday effects for different products)

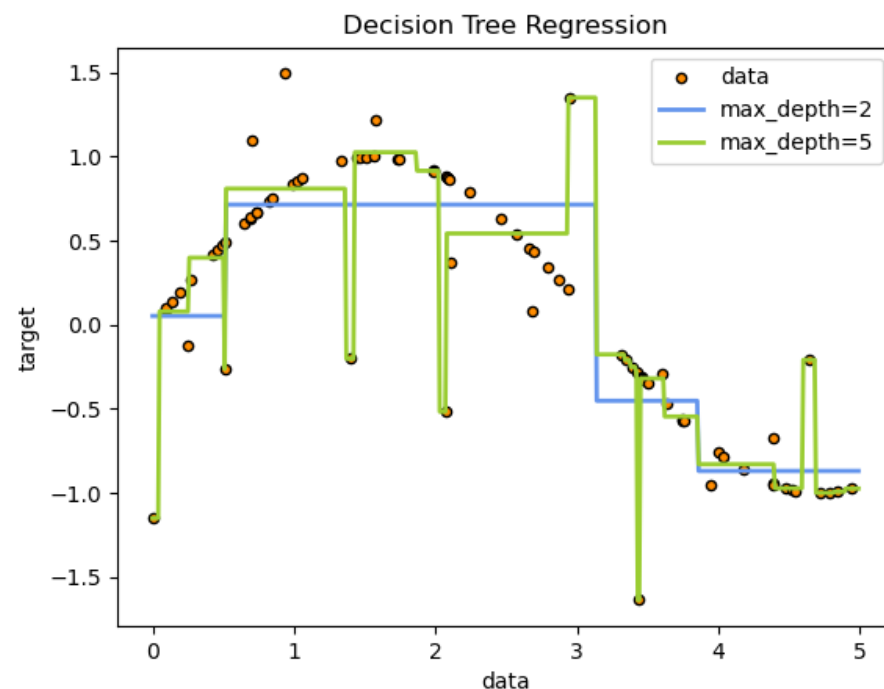
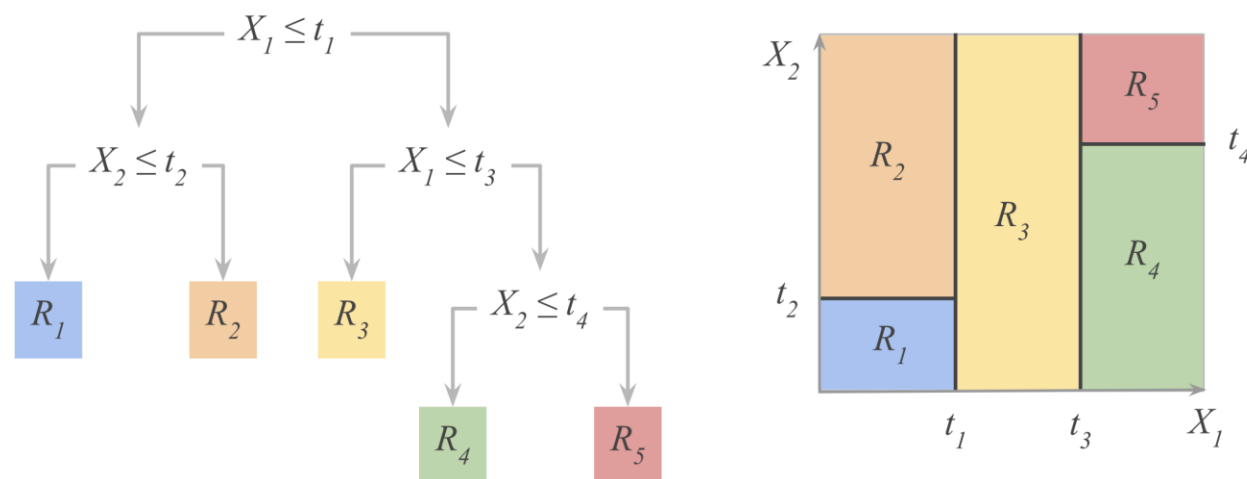
Cyclic Boosting



Automatic Interaction Learning: Gradient Boosting

Decision Trees

- non-parametric learning of simple decision rules
- usually, binary trees
- axis-parallel decision boundaries (box-shaped regions in feature space)
- fit constant \hat{c} in each box (note similarity to kNN)
 - classification: majority class of target
 - regression: average of target
- fully explainable models



from [scikit-learn](https://scikit-learn.org/)

Decision Tree Learning

finding optimal partitions R by overall loss minimization computationally infeasible \rightarrow recursive partitioning of data (greedy algorithm)

for each binary partition into regions R_1 and R_2 , decisions on splitting variable j and split point s by minimizing impurity functions I (weighted by number of observations N in child nodes):

$$R_1(j, s) = \{\mathbf{x} | x_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{\mathbf{x} | x_j > s\}$$

$$\operatorname{argmin}_{j,s} \left[\frac{N_{R_1}}{N} \operatorname{argmin}_{\hat{c}_1} (I_{R_1}) + \frac{N_{R_2}}{N} \operatorname{argmin}_{\hat{c}_2} (I_{R_2}) \right] \quad \leftarrow \text{recursively for each node in tree}$$

regression:

$$I_{R_1} = \sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 \quad \rightarrow \operatorname{argmin}_{\hat{c}_1} (I_{R_1}) \quad \text{solved by} \quad \hat{c}_1 = \bar{y}_{R_1}$$

for $\operatorname{argmin}_{j,s}$, choose boxes to make target averages in each box as different as possible

Boosting

idea: sequentially learn and combine several “weak” learners (such as small decision trees, but in principle any ML algorithm) to construct a “strong” one

→ gradually (in a greedy fashion) reducing bias of ensemble model

in simple terms: building a model from the training data, then creating a second model that attempts to correct the errors from the first model, ...

→ each subsequent weak learner is forced to concentrate on the examples that are missed by the previous ones in the sequence

not a committee of models

typically, use simple, high-bias methods as individual models

Gradient Boosting

forward stagewise additive modeling:

1. initialize $\hat{f}_0(\mathbf{x}) = 0$
2. for $m = 1$ to M
 - a) compute $\underset{\hat{h}_m}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, \hat{f}_{m-1}(\mathbf{x}_i) + \hat{h}_m(\mathbf{x}))$
 - b) set $\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1}(\mathbf{x}) + \hat{h}_m(\mathbf{x})$

→ functional gradient descent

$$\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1}(\mathbf{x}) + \hat{h}_m(\mathbf{x}) \left[\frac{\partial L(y_i, \hat{f}(\mathbf{x}_i))}{\partial \hat{f}(\mathbf{x}_i)} \right]_{\hat{f}=\hat{f}_{m-1}}$$

prominent implementations:

- [XGBoost](#): uses second-order Taylor approximation in loss function (Newton-Raphson instead of gradient descent)
- [LGBM](#): uses improved histogram-based algorithm
- [CatBoost](#): uses kind of leave-one-out encoding for categorical features

Decision Tree Sizes for Boosting

consider degree to which features interact with each other in given data set
(see ANOVA expansion)

- decision trees with single split (aka decision stumps): covering no interaction effects, just main effects of individual features
 - decision trees with two splits: covering second-order interactions
 - decision trees with three splits: covering third-order interactions
- (usually interaction order not much higher than ~ 5)

why boosting often works better than single large, low-bias model:
uncorrelated learners, each focusing on a specific aspect of the data

Embeddings & Feature Learning: Deep Learning

Categorical Variables

tabular data usually heterogenous, often with sparse categorical variables (like color of an object)

→ need for an encoding for categorical variables

different possibilities:

- ordinal encoding
- leave-one-out encoding (use mean of target for given category excluding current row)
- one-hot encoding (suffers from curse of dimensionality)
- embeddings

stacking:

```
X.sort_values(['date'])  
.groupby(['store','item'])['y']  
.apply(lambda x: x.shift(3))  
.ewm(alpha=0.15).mean()
```

prediction horizon

Embeddings

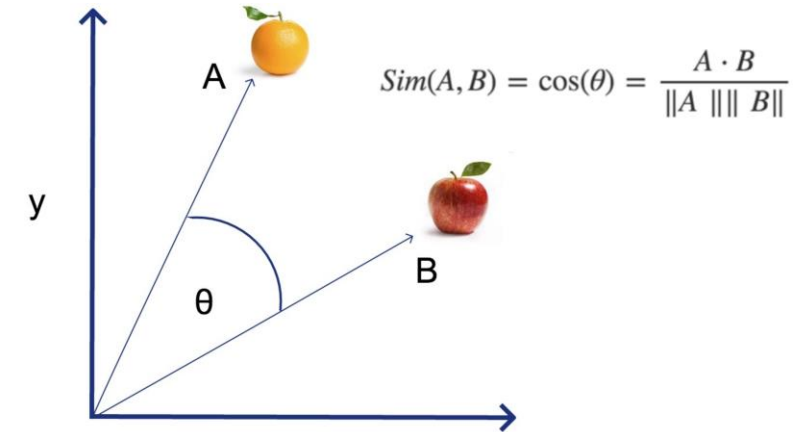
representation of entities by vectors

similarity between embeddings by, e.g.,
cosine similarity → semantic similarity

most famous application: word embeddings
→ associations (natural language processing)

but general concept: embeddings of
(categorical) features (e.g., products in
recommendation engines)

end-to-end learning: embeddings layer in
deep neural network, e.g., multi-layer
perceptron (MLP)

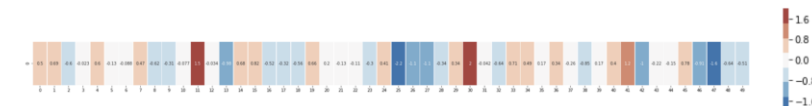
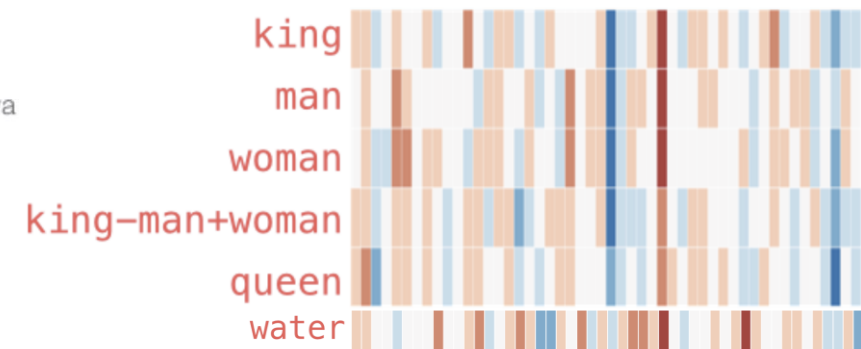


Embedding



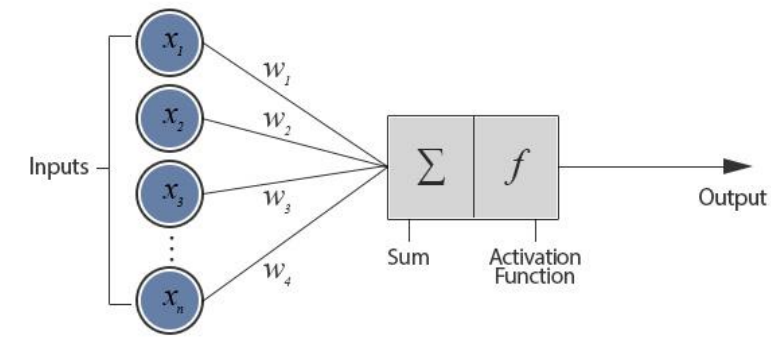
but also direction of difference
vectors interesting (analogies):

king - man + woman ≈ queen



[source](#)

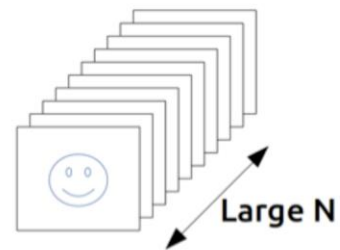
(Deep) Neural Networks



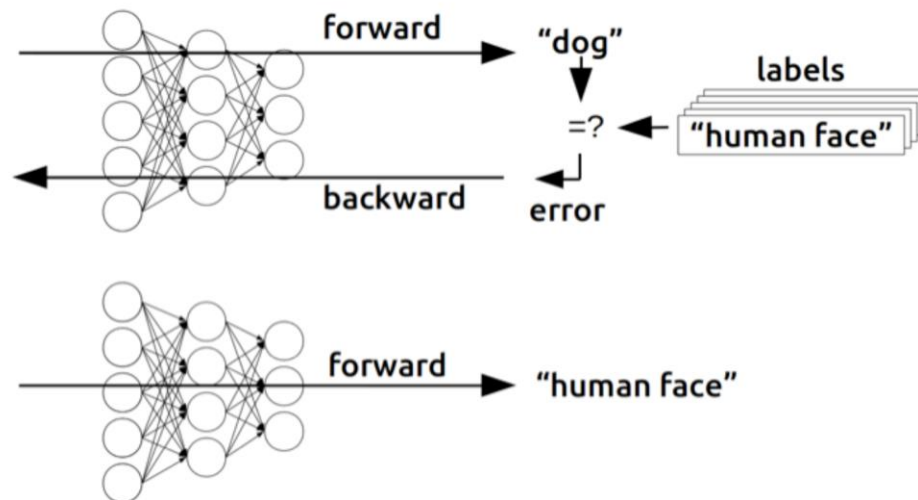
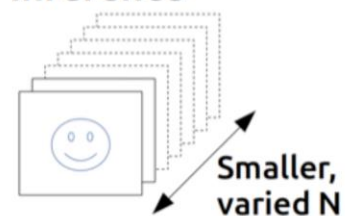
back-propagation of errors (gradients of cost function according to weights) through layers via chain rule of calculus (avoiding redundant calculations of intermediate terms)

each node exchanges information only with directly connected nodes → enables efficient, parallel computation

Training



Inference



- forward pass: current weights fixed, predictions computed
- backward pass: errors computed from predictions and back-propagated → weights then updated according to loss gradients (via gradient descent)

Examples for Feature Engineering

seasonality

day of week, payday, week of year

weather

event-like: first sunny weekend, snowstorm, ...

events (e.g., holidays)

time window around event (e.g., monotonic)

promotion

days since start of promotion period

price

ratio of reduced and normal price, price-demand elasticity

product information

text (name/description) or image embeddings

Feature Engineering vs Feature Learning

shallow learning:

representation encoded in features

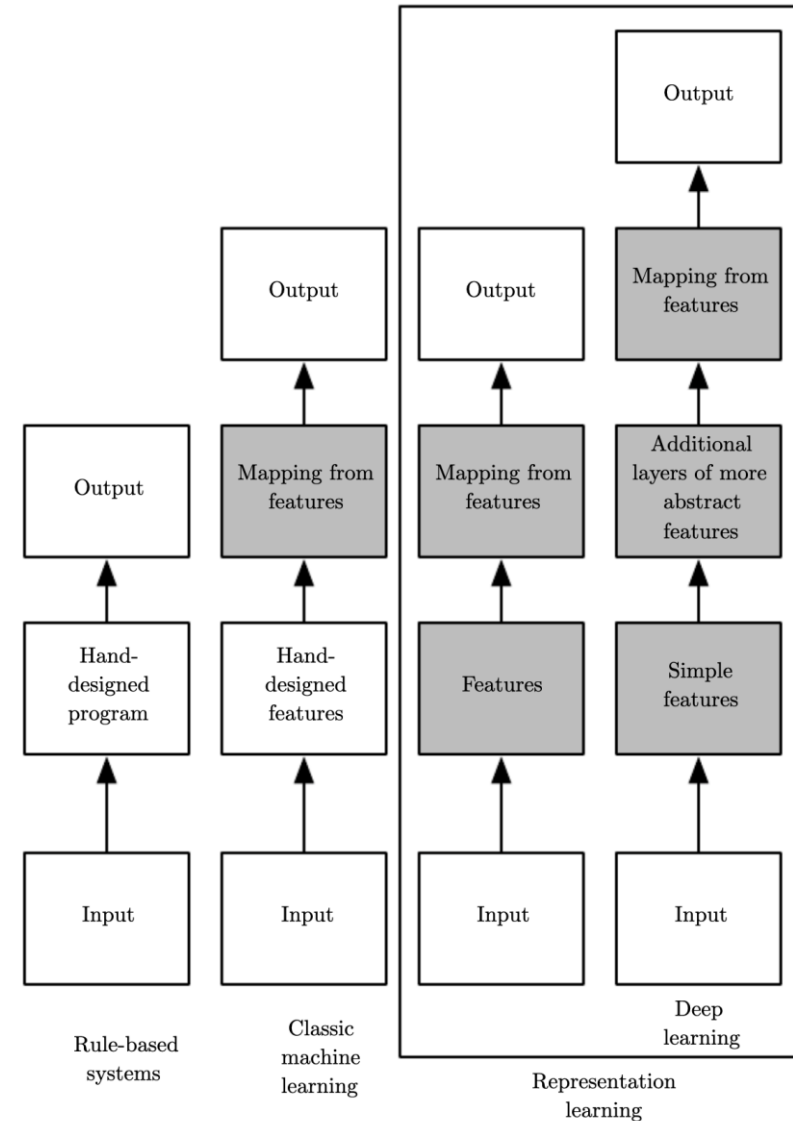
→ feature engineering

deep learning:

representation encoded in network

→ feature/representation learning

(hierarchy of concepts learned from raw data in deep graph with many layers)



[source](#)

Tabular vs Unstructured Data

deep learning methods dominate applications on unstructured data (like text or images), but not necessarily on tabular data

typical characteristics of tabular data difficult to handle for deep learning:

- irregular patterns in target function (neural networks require piecewise continuous targets)
- uninformative features
- non-rotationally invariant data (linear combinations of features misrepresent the information)

tree-based models (e.g., gradient boosting) can naturally deal with these situations

Black-Box Models

To build trust in AI systems, individual predictions/actions need to be fully transparent, i.e., explainable.

(do not confuse explainability with causality though)

Unfortunately, complex models like deep learning methods are difficult to interpret.

→ need for model-agnostic methods to explain black-box models

examples: local surrogates ([LIME](#)), Shapley values ([SHAP](#))

[overview](#)

Deep Learning Methods for Sequential Data

→ direct inclusion of auto-correlation in ML model (but remember: mainly spurious)

convolutional neural networks (CNN)

- main application computer vision: images correspond to regular 2D grids of data (pixels)
- time series data as 1D grid at regular time intervals
- parameter sharing via convolutions (and pooling) → highly regularized feed-forward networks
- fixed time window (kernel)
- multivariate time series model via different input channels

recurrent neural networks (RNN)

- main application natural language processing (NLP): sequence of words (tokens)
- time series data as time-tagged stream
- parameter sharing via recurrent nodes → context-awareness (input from past activations)
- variable-length time series
- multivariate time series model via more input nodes

transformers

- main application NLP (largely replaced RNNs), but also computer vision
- time series data as directed graph (attention in edges)
- masked self-attention to focus on relevant past time steps and positional encoding to include order of sequence
- multivariate time series model via input concatenation

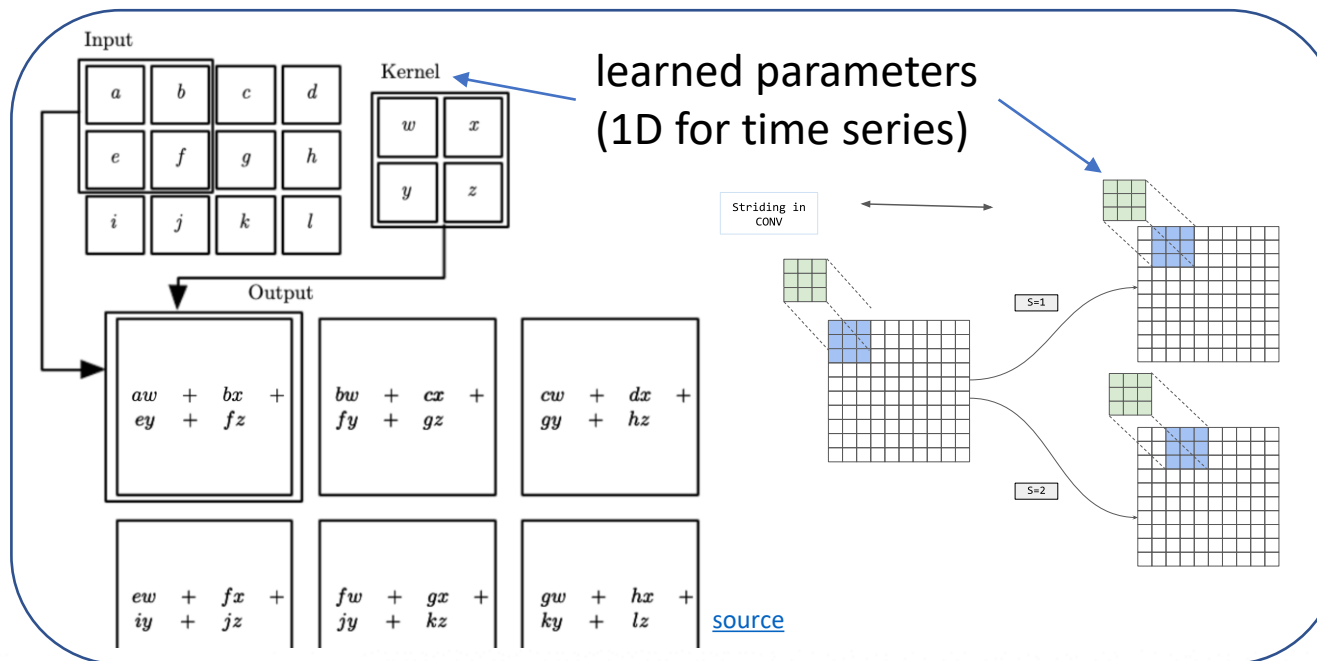
CNN

```

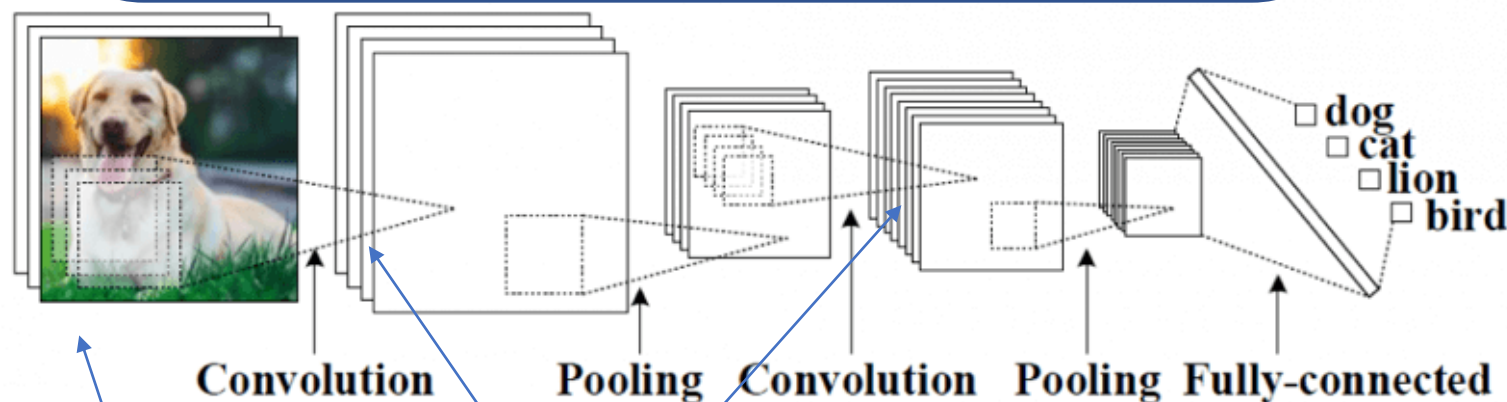
0 2 15 0 0 11 10 0 0 0 9 9 0 0 0
0 0 0 4 60 157 236 255 255 177 95 61 32 0 29
0 10 16 115 238 255 244 245 243 250 249 255 222 103 10 0
0 14 170 255 255 244 254 255 253 245 255 249 253 251 124 1
2 98 255 228 255 251 254 211 141 116 122 215 251 238 255 49
13 217 243 255 155 33 226 52 2 0 10 13 232 255 255 36
16 229 252 254 49 12 0 0 7 7 0 10 237 252 235 62
6 141 245 255 212 25 11 9 3 0 115 236 243 255 137 0
0 87 252 250 248 215 60 0 1 121 252 255 248 144 6 0
0 13 117 255 255 245 255 182 181 248 252 242 208 36 0 19
1 0 5 117 251 255 241 255 247 255 241 162 17 0 7 0
0 0 0 4 58 251 255 246 254 253 255 120 11 0 1 0
0 0 4 57 255 255 255 248 252 255 244 255 182 10 0 4
0 22 206 252 246 251 241 100 24 113 255 245 255 194 9 0
0 11 255 242 255 158 24 0 0 6 30 255 232 230 56 0
0 218 251 250 137 7 11 0 0 0 2 62 255 250 125 3
0 173 255 255 101 9 20 0 13 3 18 182 251 245 61 0
0 107 251 241 255 230 98 55 19 115 217 248 253 255 52 4
0 18 146 250 255 247 255 255 255 249 255 240 255 125 0 5
0 0 23 115 215 255 250 248 255 255 248 248 118 14 12 0
0 0 6 1 0 52 153 233 255 252 147 37 0 0 4 1
0 0 5 5 0 0 0 0 0 0 14 1 0 6 6 0 0
    
```

[source](#)

1D for time series



[source](#)

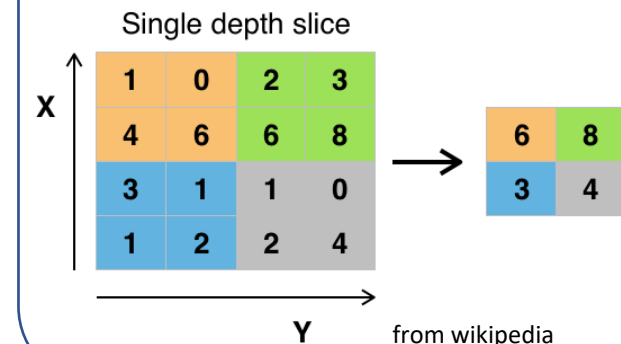


many images
(training examples),
potentially with
several channels

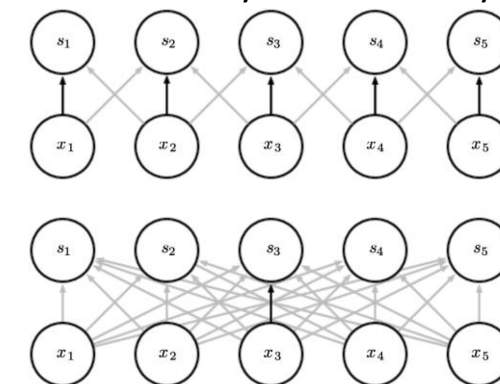
several kernels
producing several
feature maps

down-sampling
by convolutions
and pooling

max pooling:

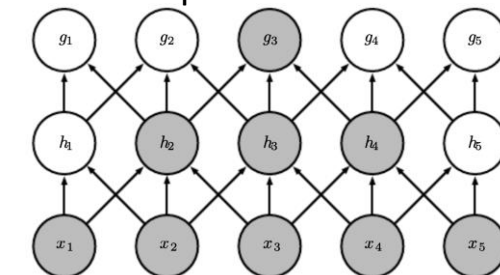


convolutional vs fully-connected layer:

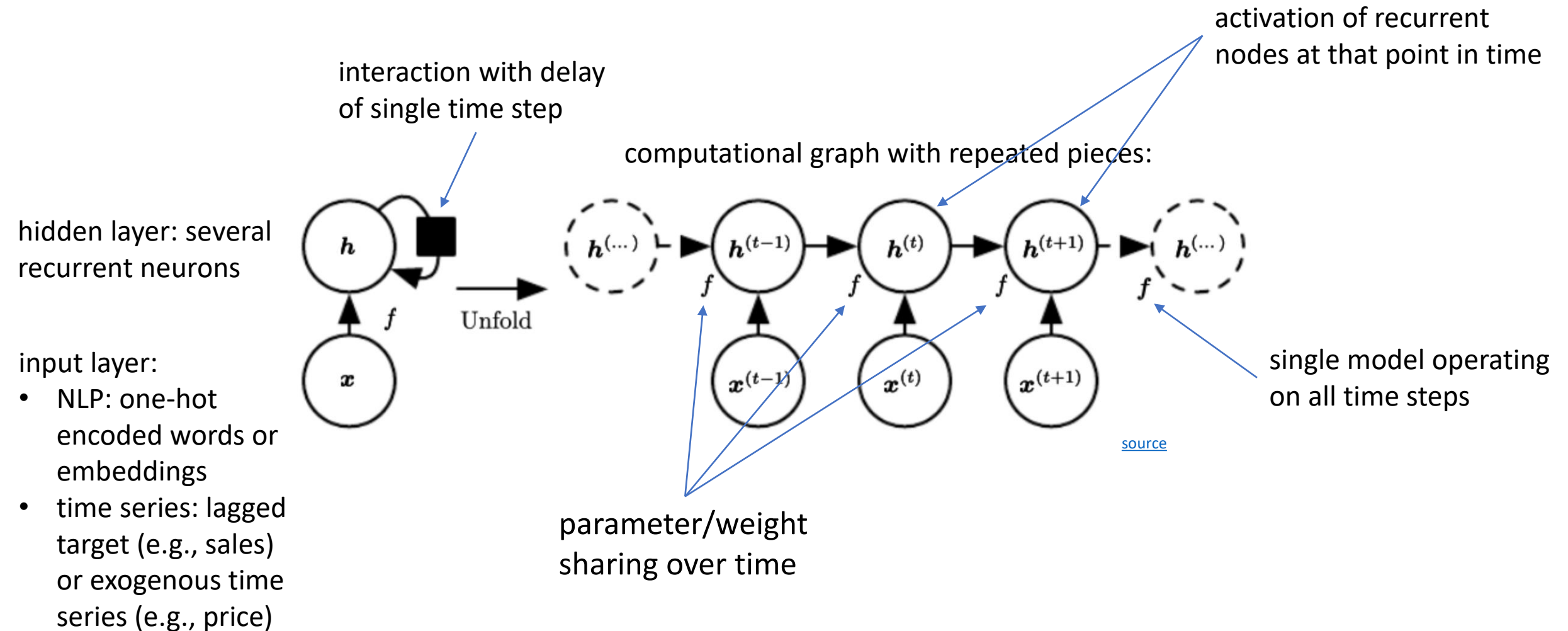


[source](#)

receptive fields:

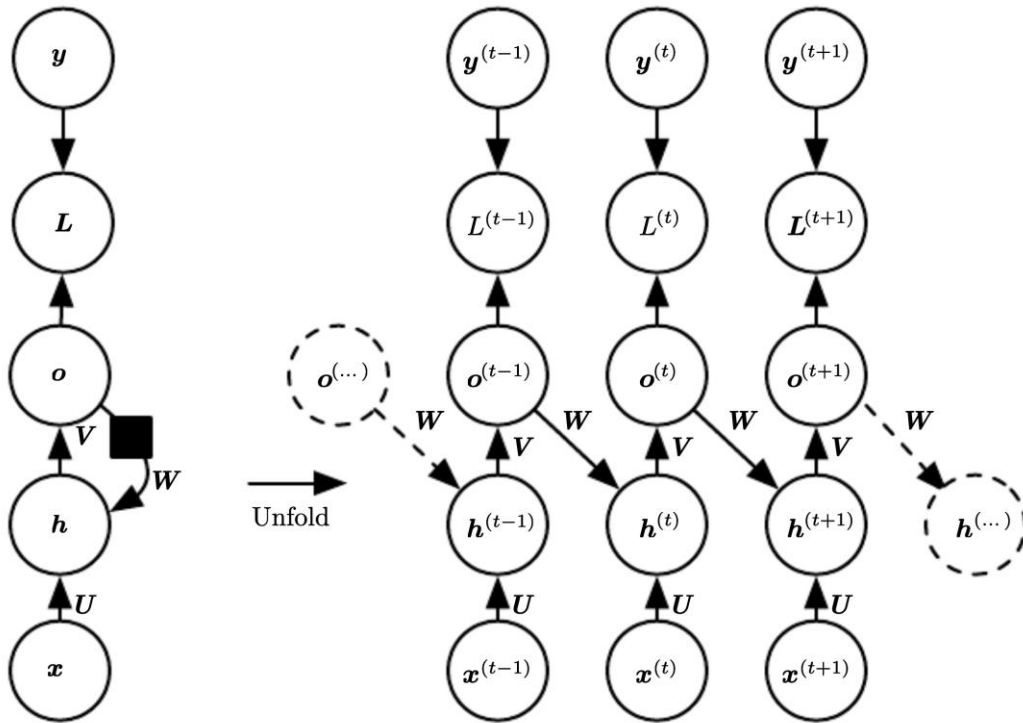


RNN: Back-Propagation through Time



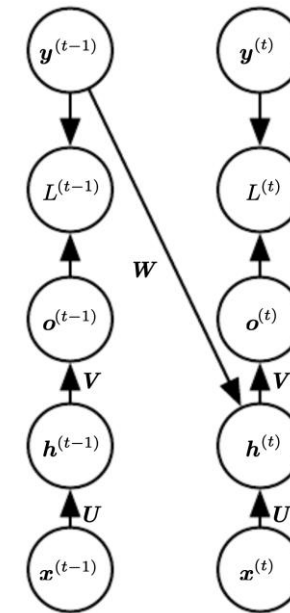
Teacher Forcing

e.g., RNN:

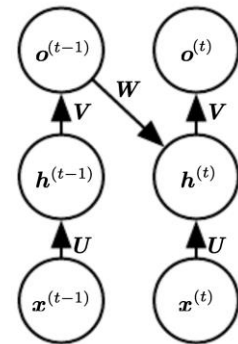


instead of feeding model output back into itself, use target values directly for feedback from output to hidden layer

example demand forecasting: sales from day before (different to usage as input from last slide)



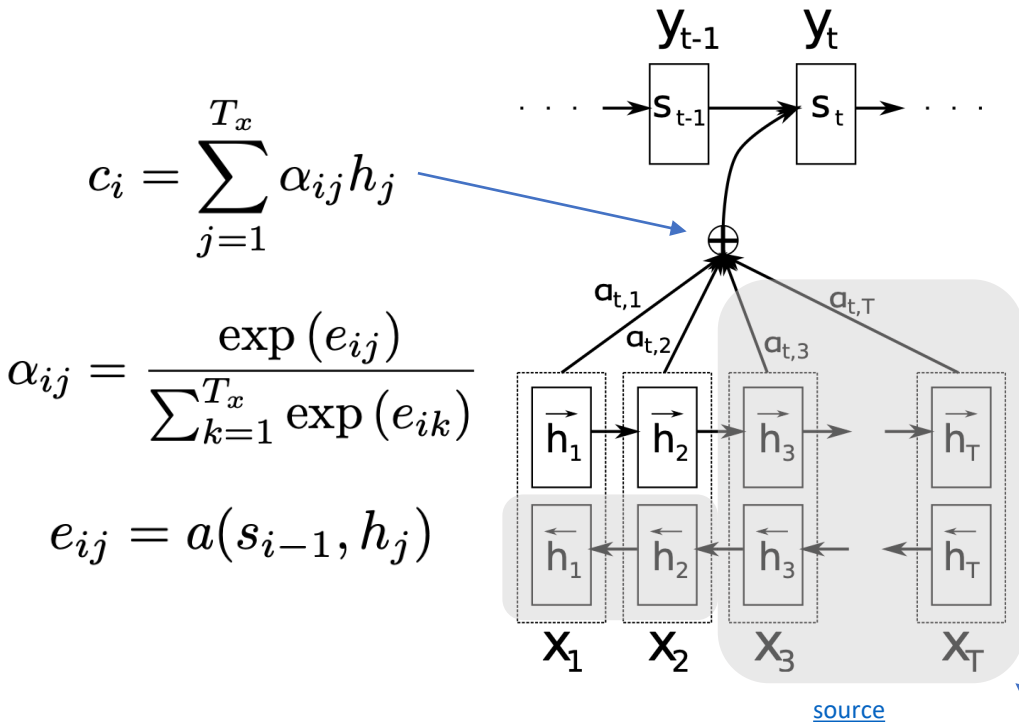
steps depend on forecast horizon



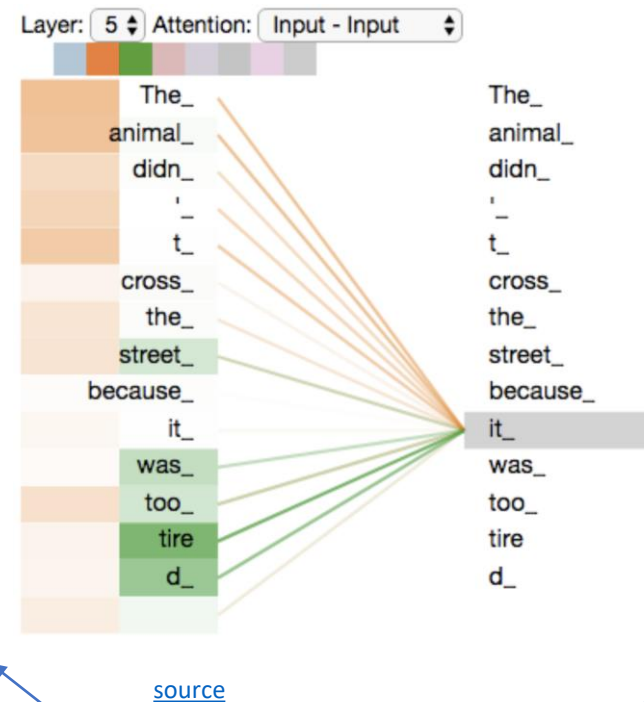
[source](#)

Transformer

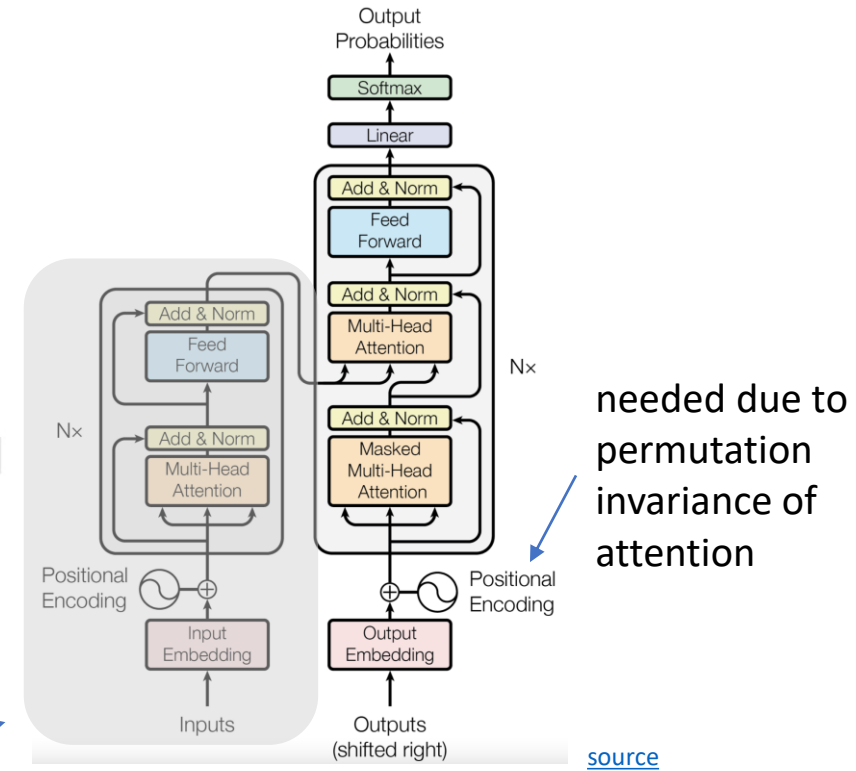
attention:



multi-headed self-attention:



transformer:



time series: masked self-attention (only attending to past) → decoder-only (auto-regressive)

Demand Forecasting Subtleties

Censored Target

using sales data from past as target quantity to reflect unobservable demand

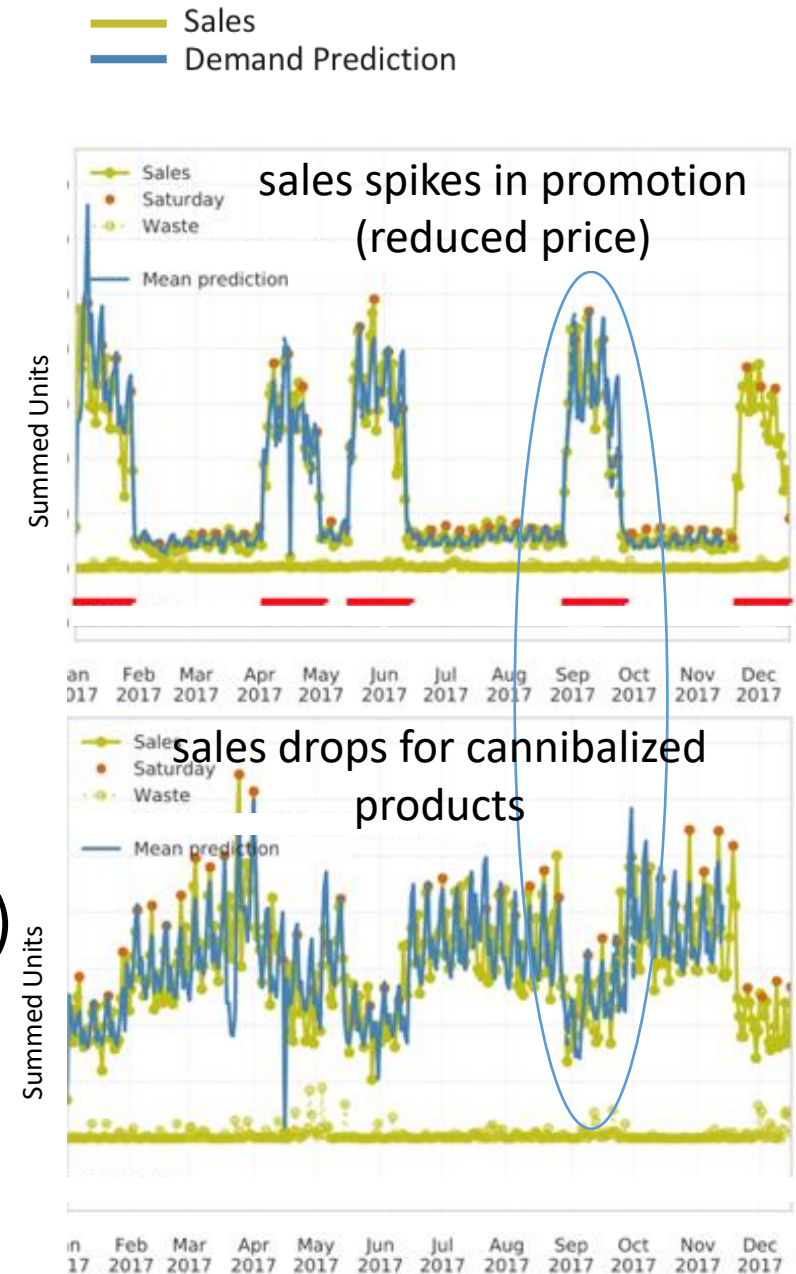
BUT: sales data correspond to censored demand due to

- out-of-stock situations
 - markdown sales: reduced prices for items, e.g., nearing its best before date → increased sales compared to demand at usual, unreduced price
- need for (probabilistic) target correction in these situations to enable unbiased demand forecasts

Cannibalization and Halo Effects

sample interactions: demand decreases (cannibalization) or increases (halo) due to other products in promotion at the same time

- additional ML model(s) after demand forecasting
 - prediction residuals as target (all other effects captured)
 - learn individual effect for each product pair (product matrix with separate models for each row)
 - price reduction (or promotion flag) of different other products as features
 - feature selection to avoid overfitting (e.g., lasso)



Order Optimization: Need for PDF Predictions

Prediction of Full Probability Distributions

quantile regression:

estimate quantile τ of distribution instead of conditional mean by minimizing

$$(1 - \tau) \sum_{y_i < \hat{q}} (y_i - \hat{q}) + \tau \sum_{y_i \geq \hat{q}} (y_i - \hat{q})$$

instead of squared error loss (choice of loss function defines point estimate)

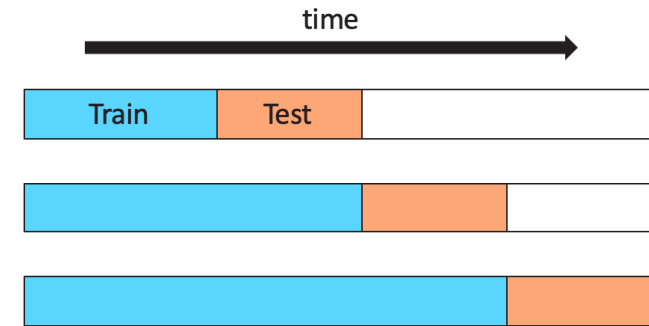
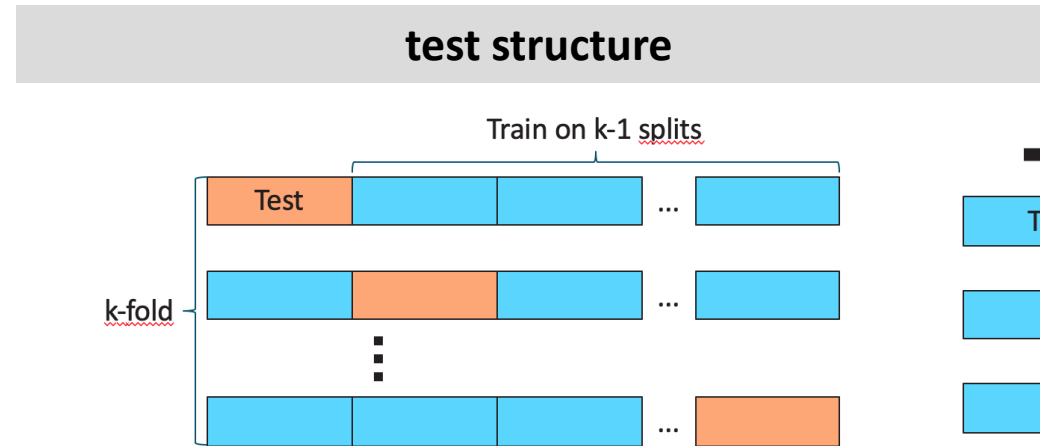
possible with various ML methods, including neural networks and tree-based methods (like random forests or gradient boosting)

PDF assumption for model and estimation of its parameters:

assume, e.g., Gaussian or negative binomial \rightarrow estimate mean and variance (with different ML models or corresponding likelihood)

Model Evaluation

cross-validation:



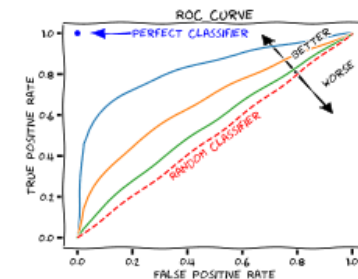
decide on acceptance of model changes by accuracy measure: improved model vs baseline (current best)

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

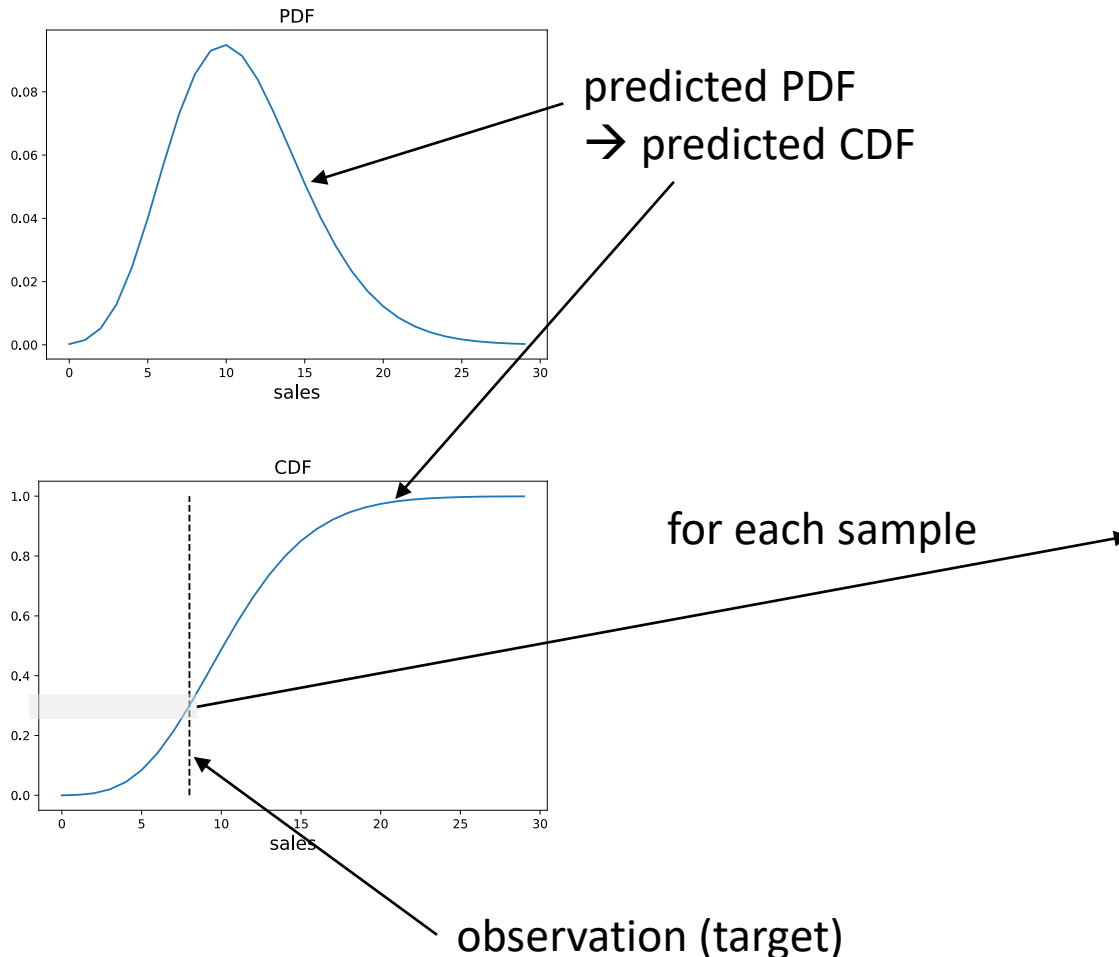
classification



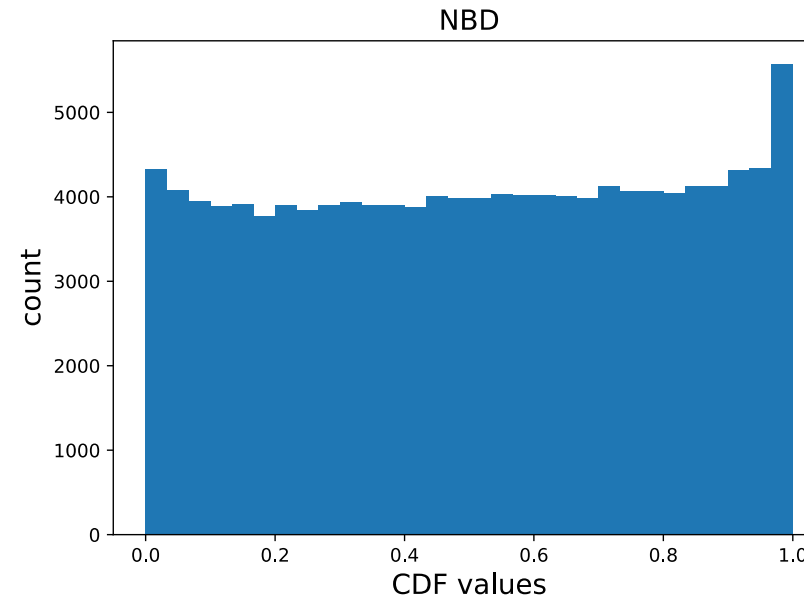
from wikipedia

Evaluation of PDF Predictions: Histogram of CDF Observations

individual prediction/observation:

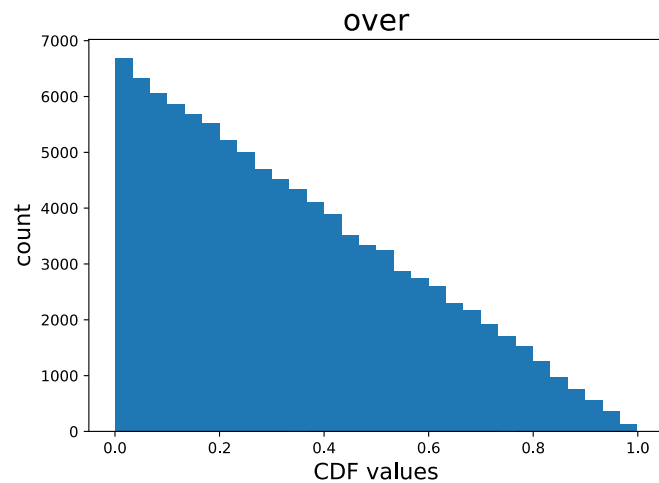
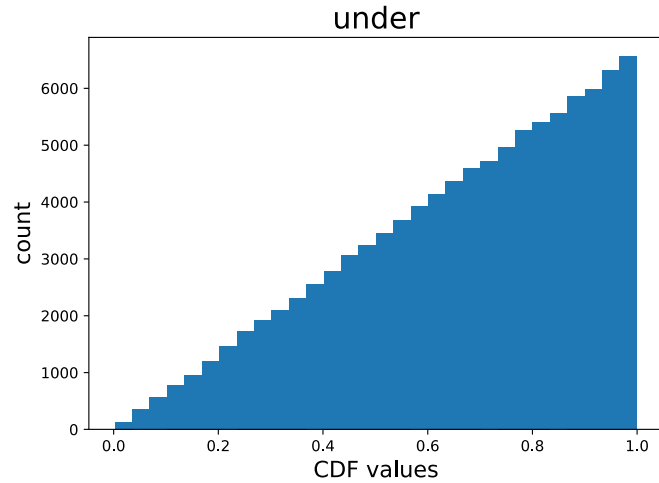


many (or all) predictions/observations:

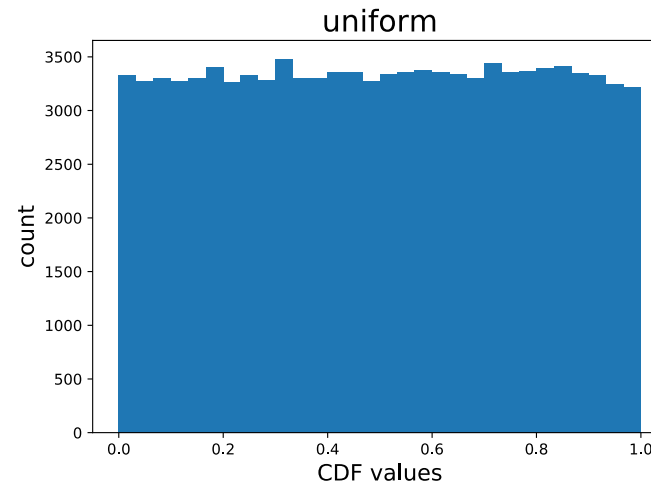


subtlety for discrete PDFs:
histogram filled using random numbers according to the CDF intervals (e.g., for 3 actual sales: random pick between discrete CDF values for 2 and 3)

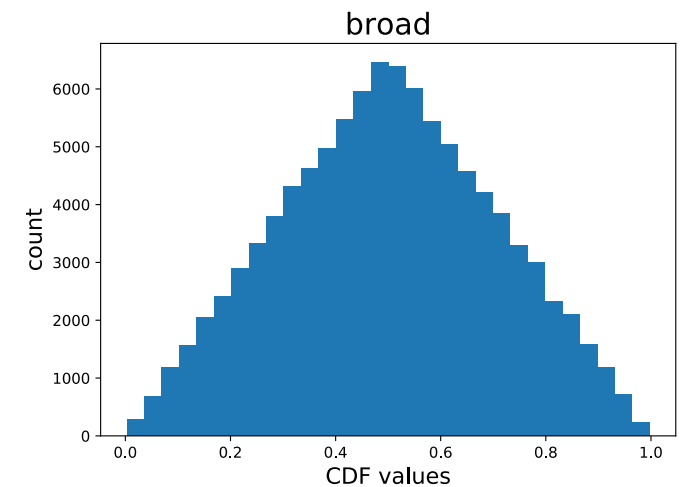
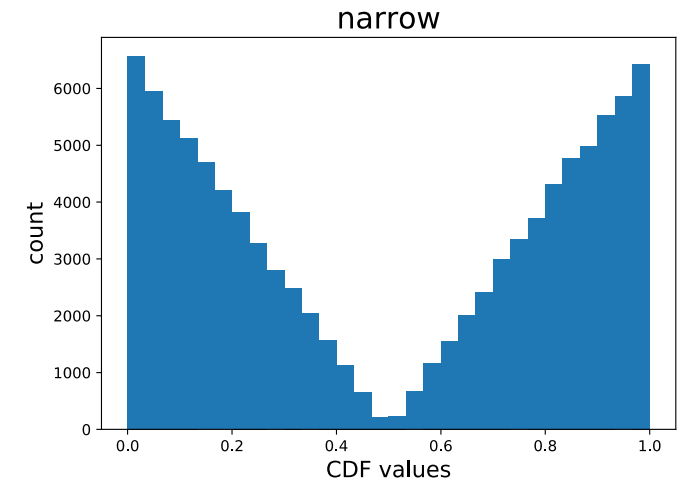
Interpretation of CDF Histogram



if all is fine ...

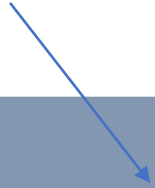


quantitative evaluation: measure
deviation from uniform distribution
(e.g., earth mover's distance)



Replenishment

loss: deviation between
predicted value and truth



1. demand forecasts in form of full, individual probability distributions
2. order proposals by individually choosing the quantile minimizing expected costs (scaled losses)

simplistic: **order quantity = current inventory - upcoming demand**
(ignoring delivery lead times and shrinkage beyond demand, e.g., waste)

potential add-on:
(probabilistic) inventory estimation

assumption for two-step approach:
demand (not sales) not affected by subsequent action of ordering

In fact, stock level can have small effects on demand. For example, full shelves might increase demand.

Need for Probability Distributions instead of mere Point Estimators

one best (usually different) quantile for each (out-of-sample) error measure

examples for error measures and their optimal point estimators:

- mean squared error (**MSE**) optimized by **mean**
- mean absolute error/deviation (**MAD**) optimized by **median**

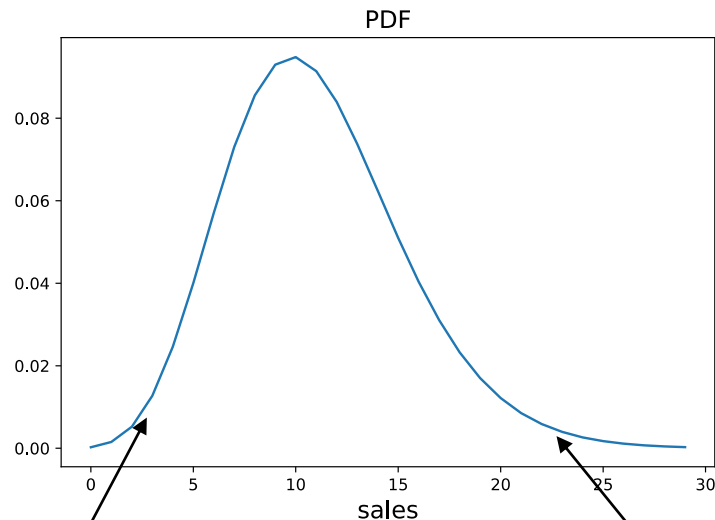
but usually **individual cost function** for each product (possibly even for different dates of one product),
→ individual, optimal quantile as point estimator

possible approaches:

- many quantile regressions: for each individual, optimal point estimator use corresponding loss function
- full, individual probability distributions, i.e., probability density functions (PDF) for the continuous case

Order Optimization

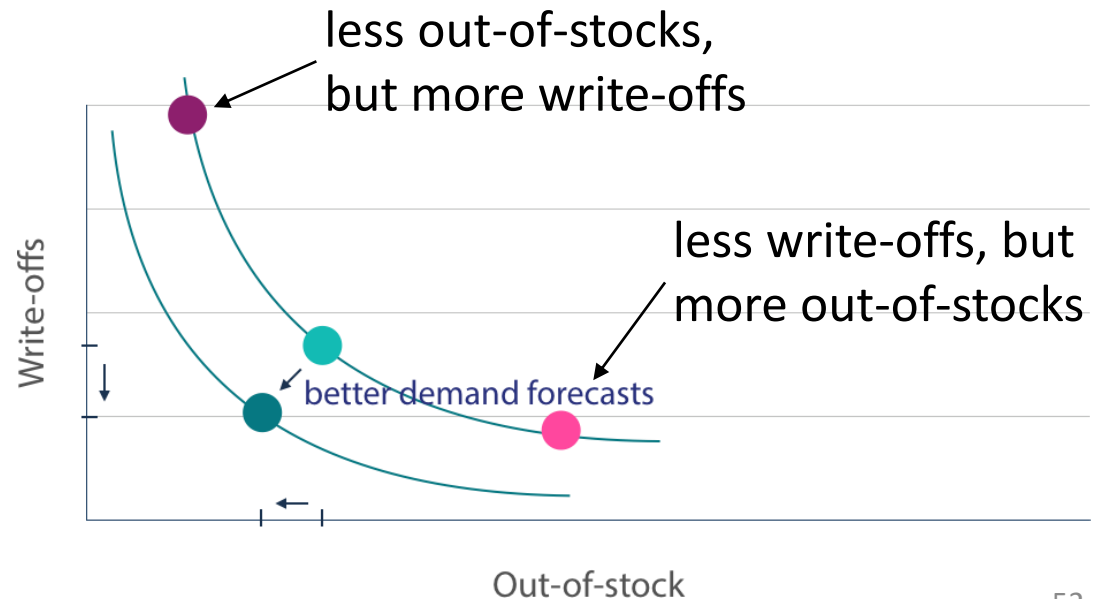
costs refer to **scaled demand losses**: assign costs to out-of-stocks (lost sales), write-offs (waste), ...
→ full **multi-dimensional cost function** as sum of different costs



choose low quantile
for ordering:
high understock risk

choose high quantile
for ordering:
high overstock risk

complete cost functions often not available in practice
→ simulate impact of quantile choice on different KPIs
(by means of full PDF predictions)



Demand Shaping: ML and Causality

Data-Generating Process

story behind the data as important as the data itself

Why are the statistical dependencies as observed?

→ data-generating process mostly governed by causal dependencies

but language of algebra symmetric

no way to tell that a storm causes barometer to go down and not the other way around → need for asymmetric mathematical language

The Ladder of Causation

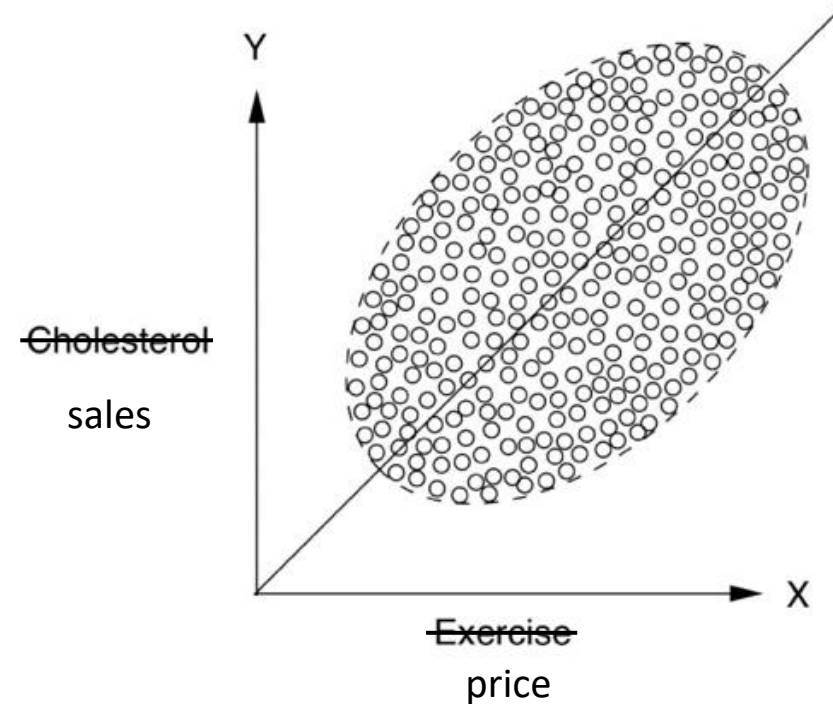
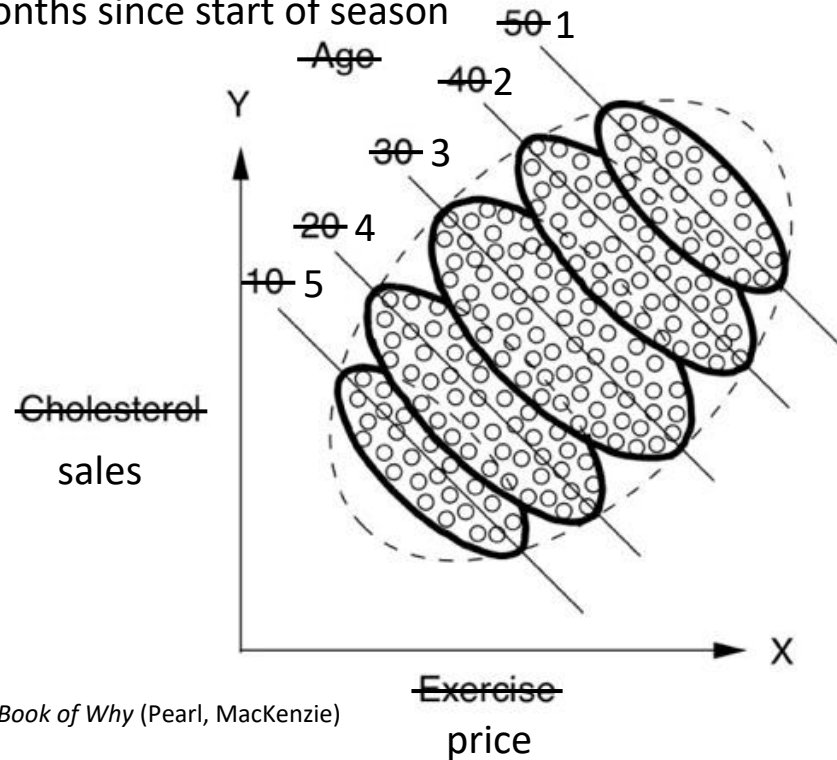
III	<i>imagining</i>	counterfactuals	What if I had done ...? Why?
II	<i>doing</i>	intervention	What if I do ...? How?
I	<i>seeing</i>	association	What if I see ...? → Realm of ML

from *The Book of Why* (Pearl, MacKenzie)

Simpson's Paradox

example: monthly sales of a fashion article in different shops during winter season
Lower price yields higher sales for each month individually, but lower sales overall?

months since start of season



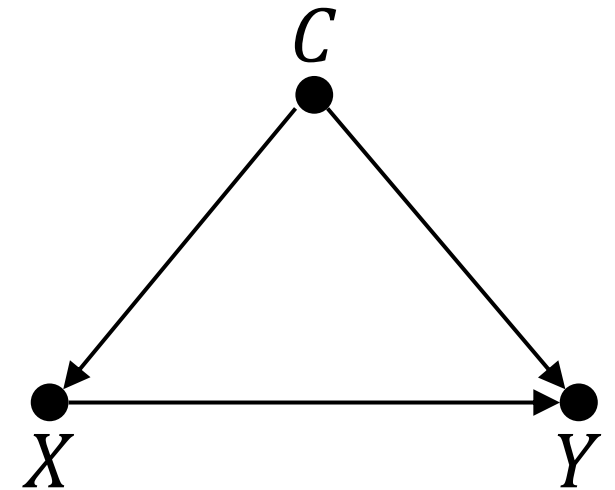
from *The Book of Why* (Pearl, MacKenzie)

Confounding

causal effect of X on Y confounded by common cause C :

$$X \leftarrow C \rightarrow Y$$

→ spurious correlation between X and Y
(overlying potential true causal effect of X on Y)



common cause principle:

every correlation either due to a direct causal effect linking the correlated entities or brought about by a third factor (confounder)

Confounders for Pricing

price setting for a product can be considered a demand shaping method

the idea is a causal effect: lower price leads to higher demand

→ by estimating this causal effect one can find an optimal price according to a given policy (like maximizing profit)

problem: confounders influencing both pricing in past (observed) data and demand, e.g., lowering prices on weekends only for grocery or lowering prices toward end of a season for fashion (most sales at beginning of season)

Counterfactuals

fundamental problem of causal inference:

only one realization: *What happened? And what could have happened instead?*

→ cannot observe different **potential outcomes (individual causal effect)**

example for application of **causal what-if scenario**:

individual (e.g., per product-location-date combination) demand shaping via price setting

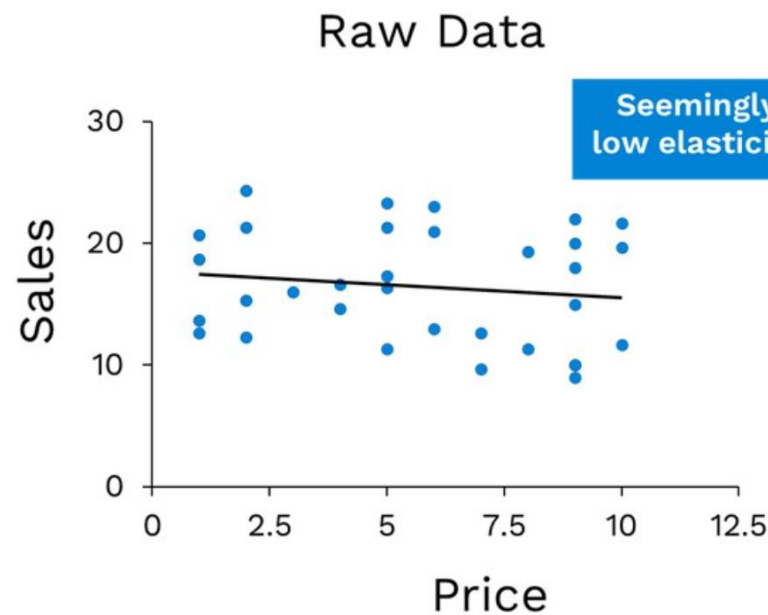
ML and data not enough to describe causal structure

→ need for **causal assumptions** as add-on to ML (going beyond curve fitting)

Demand Shaping with Causal Inference

dynamic pricing: influencing demand of different products by price setting

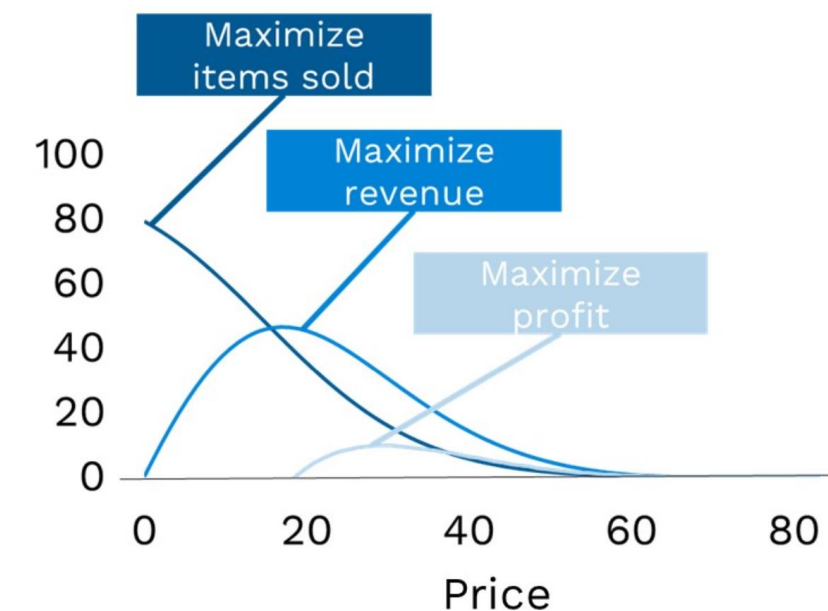
confounded effect:



after de-confounding:



use for pricing policies:



customer targeting: influence individual customer demand by sending coupons

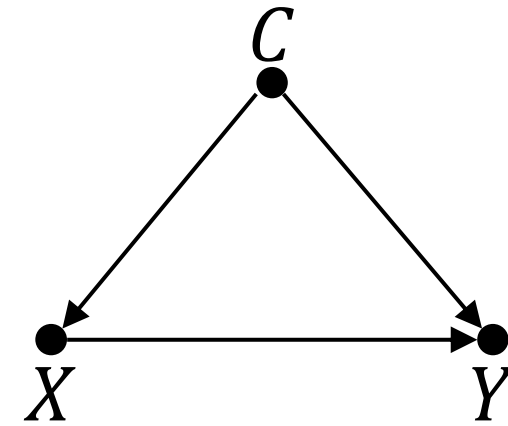
Potential Outcomes with ML

deconfounding
by RCTs or
independence weights (inverse propensity scores)



prediction of individual causal effect with ML model
(generalization by function approximation)

Independence Weights with ML



scenario: X binary (for simplicity), several confounders C

aim: prediction of individual causal effect using observational data only

- i. ML model to predict past action policy $P(X|C)$ (beware: need to include all C)
- ii. inverse propensity score weighting of each unit (to adjust for multiple confounders):

$$P(Y|do(X)) = \sum_z P(Y|X, Z = z) P(Z = z) = \sum_z \frac{P(Y, X, Z = z)}{P(X|Z = z)}$$

} alternative:
A/B test

- iii. train ML model on deconfounded data to predict Y with X and C as features
- iv. individual causal effect as difference between predictions for setting feature $x = 1$ and $x = 0$, respectively (what-if scenario)