

Intro to ML

Evgeny Burnaev

Skoltech, Moscow, Russia



1 Main concepts

2 Examples

1 Main concepts

2 Examples

- a broad subfield of Artificial Intelligence
- the mathematical discipline aimed at enabling machines (by computational methods) to improve at tasks with experience. The category includes deep learning
- data-driven → extracting patterns from data and based on mathematical statistics, numerical methods, optimization, probability theory, discrete analysis, geometry, etc.
- results of CS is used to analyze learning algorithms, their complexity, theoretical guarantees
- **Example:** predict a label of an image

- Image: annotation, segmentation, face recognition, OCR, face verification
- Autonomous technical systems (robots, cars)
- Medical diagnosis, fraud detection, network intrusions
- Playing games (chess, poker)
- Speech: recognition, synthesis, verification
- Text: topic modeling, spam detection

- **Dimensionality Reduction:** lower-dimensional features, preserving some properties of data
- **Regression:** predict some real-valued output variable for some input parameters (ship fuel consumption depending on weather conditions, route, etc.)
- **Classification:** set a label for each object (e.g. image classification)
- **Clustering:** partition objects into some “homogeneous” groups (e.g. divide documents into groups with similar topics)
- **Ranking:** rank objects according to some metric

- Algorithmic problems:
 - more efficient and more accurate algorithms
 - handle large-scale (dimensions, data volume) problems
 - handle diverse types of data sources, including non-structured data, data on graphs, etc.
- Theoretical problems:
 - what can be learned? under what conditions? restrictions?
 - learning guarantees?
 - learning algorithms performance?

- **Models and Algorithms**
 - main algorithms and their efficiency
 - modern topics
- Theory
 - learning guarantees
 - analysis of algorithms
- **Applications**

- **Example:** item, instance of data, related to some object
- **Features x :** input value (input parameters, input vector, attributes, point) characterizing an object
- **Labels y :** output value, categoric (classification) or real value (regression), associated to an object
- **Data:**
 - training data (usually contains labels)
 - test data (labels exist but not known)
 - validation data (labeled, used for tuning of hyperparameters)

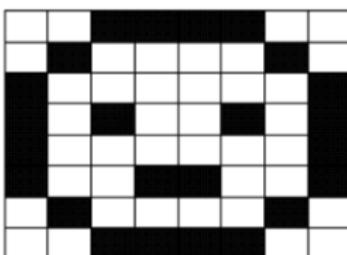
- o is an object
- $\mathbf{x}(o)$ is a set of features
- y is a label (class, category, etc.)
- For a new \mathbf{x} we should define a corresponding class y

Machine Learning Problem: Example

O



$x(O)$



Y

Person
Bird
House

- space of objects (input space) X
- space of labels (output space) Y
- unknown target function $f : X \rightarrow Y$
- **Given:**
 - Learning sample $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$
 - Objects $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \in X$
 - Labels $\{y_1, \dots, y_m\} \in Y, y_i = f(\mathbf{x}_i)$
- **Find:**
 - $\hat{f} : X \rightarrow Y$, approximating f on X
- Mainly ML is about how to
 - define description of objects \mathbf{x}_i
 - define “ \hat{f} approximates f ”
 - construct \hat{f} using S_m

- **Object** o : place to open a new restaurant
- **Label** y : revenue after one year of operation
- **Features** $\mathbf{x} = \mathbf{x}(o)$: demographic properties of a considered city district, prices for real estate in a local neighborhood, availability of offices nearby, etc.
- **Challenges:**
 - small sample size
 - a lot of features ($d \gg 1$)
 - outliers/incorrect measurements
 - non-homogeneous data (big cities vs. local towns)

- Usually as object \mathbf{x} description we consider a set of features, i.e.
 $\mathbf{x} = (x_1, \dots, x_d)$
- There are several types of features
 - binary, e.g. $x_j \in \{0, 1\}$
 - nominal, e.g. $x_j \in \{\text{red}, \text{green}, \text{blue}\}$
 - ordinal, e.g. $x_j \in \{\text{low}, \text{middle}, \text{high}\}$
 - real number, i.e. $x_j \in \mathbb{R}$
- Thus objects $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \in X$ can be represented as an $m \times d$ matrix

$$\begin{pmatrix} x_{11} & \dots & x_{1d} \\ \dots & \dots & \dots \\ x_{m1} & \dots & x_{md} \end{pmatrix}$$

- Classification
 - two-class classification: $Y \in \{0, 1\}$
 - multi-class classification (mono-label case): $Y = \{1, \dots, K\}$
 - multi-class classification (multi-label case): $Y = \{0, 1\}^K$
- Regression: $Y = \mathbb{R}$ or $Y = \mathbb{R}^K$
- Ranking (learning to rank): Y is a finite ordered set of categories

- Settings

batch: a learner get full sample, learn a model and performs predictions for unseen points

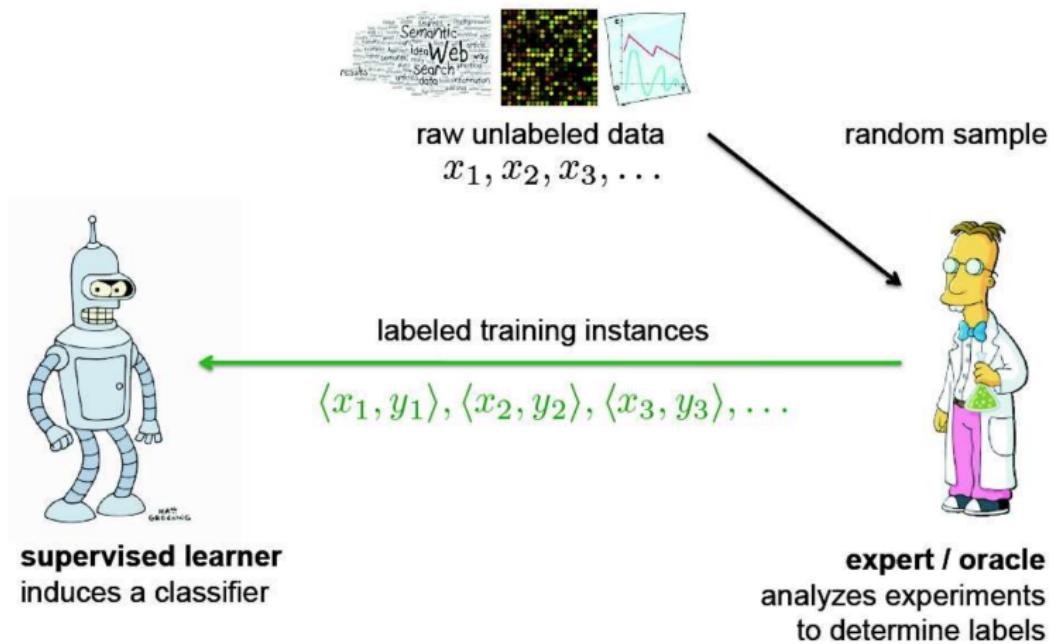
on-line: a learner receives one sample at a time and makes prediction for that sample

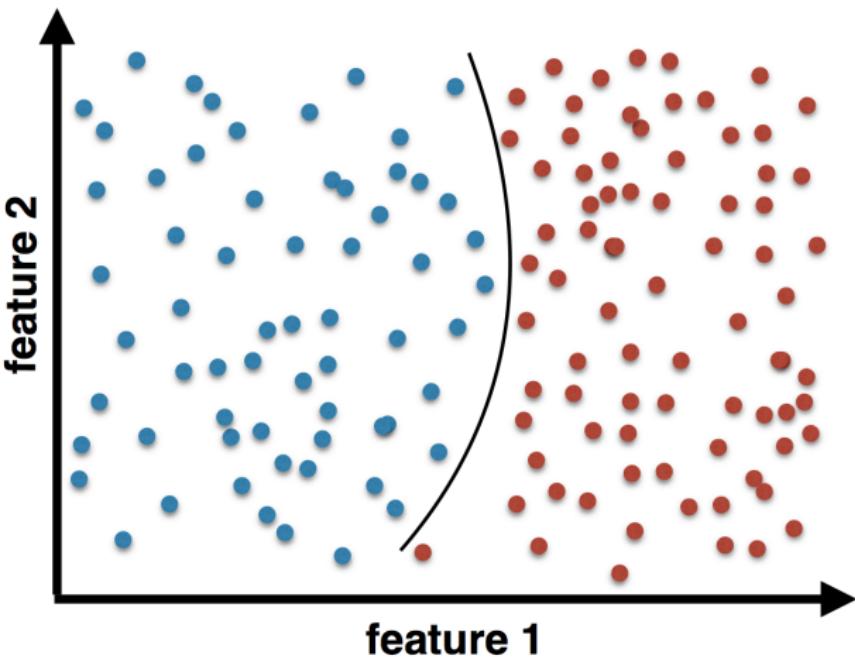
- Queries

active: a learner can request the label of a point

passive: a learner always receives labeled points

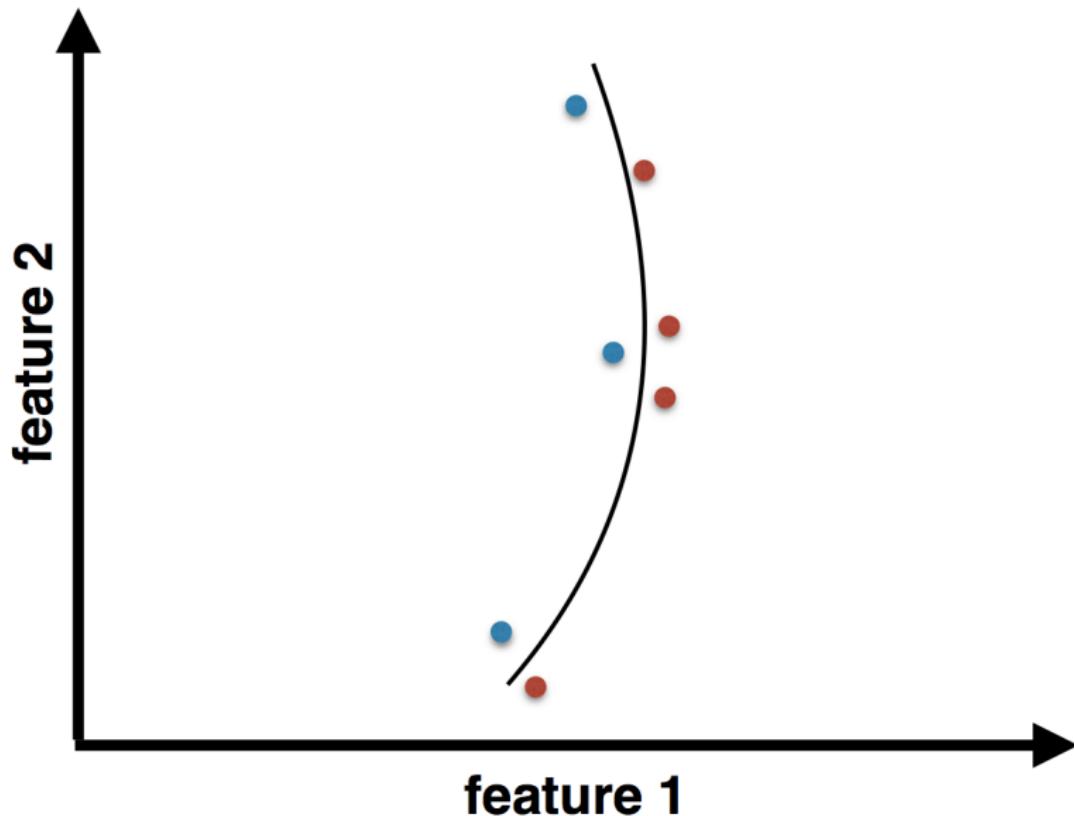
(Passive) Supervised Learning



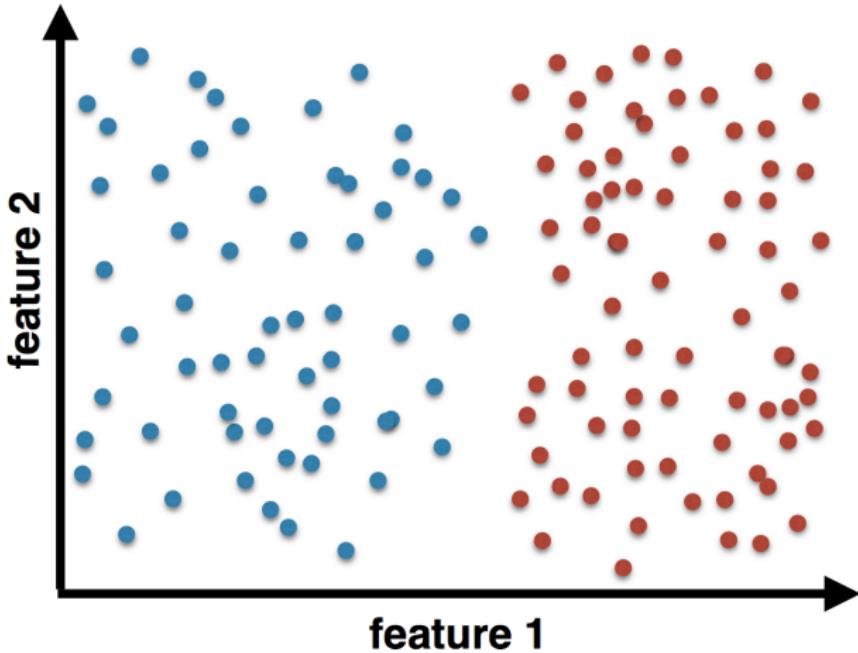


- Traditional approach almost universally adopted

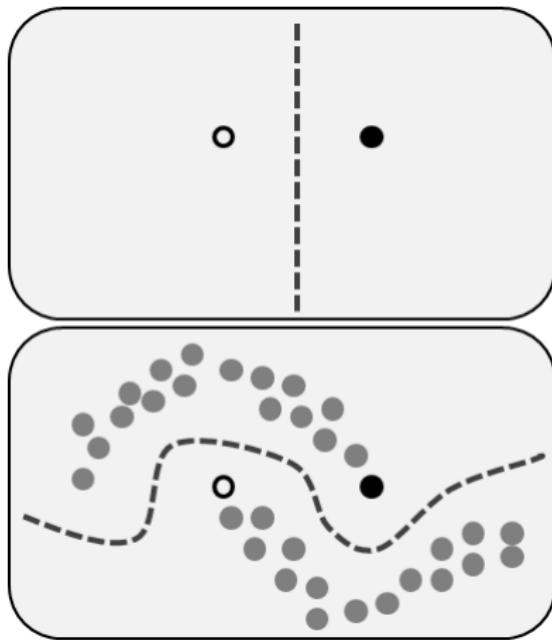
Well, we actually only needed this!

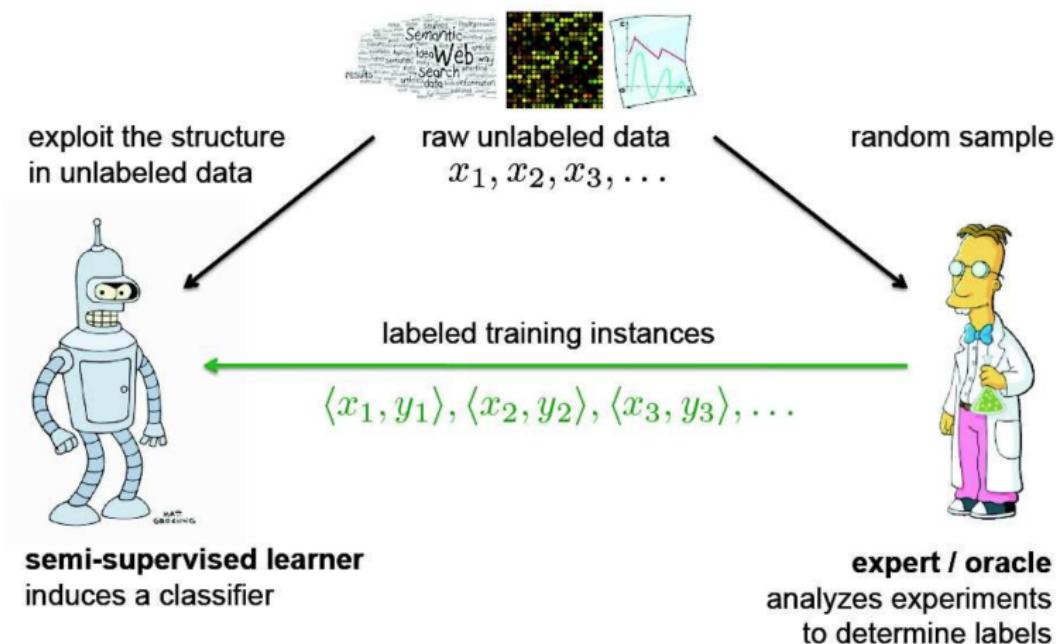


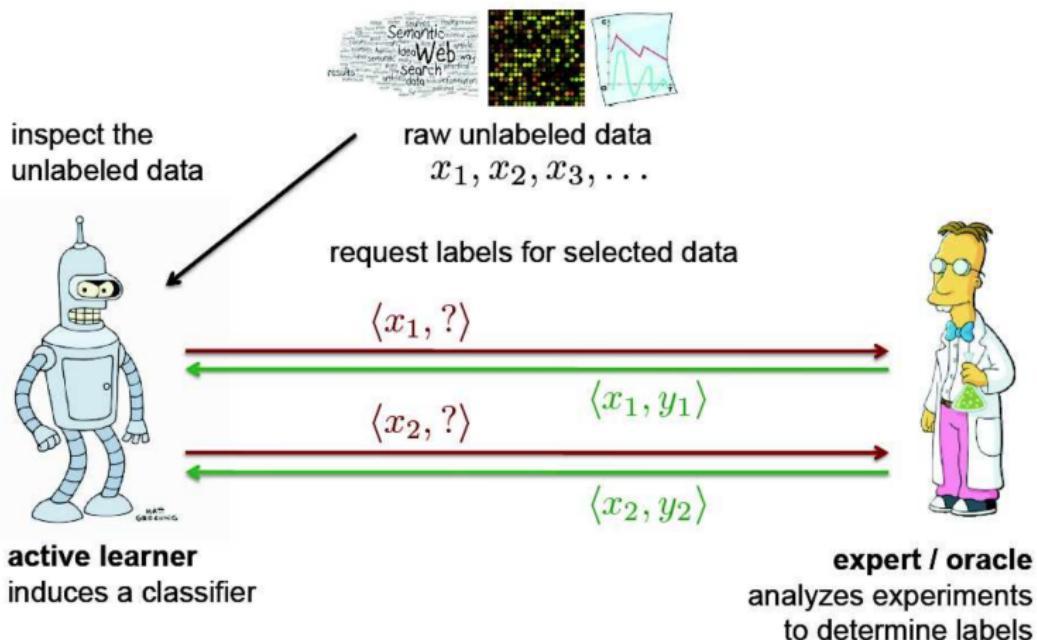
So this was a complete waste of time!



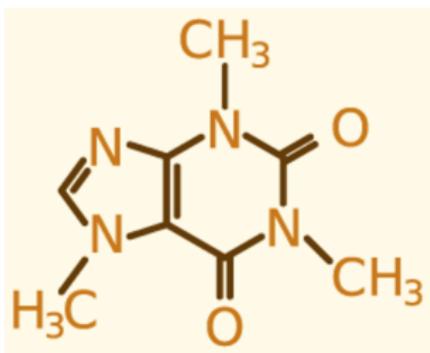
- Random sampling inevitably leads to redundancy







- Goal: find compounds which bind to a particular target



Large collections of compounds from:

- vendor catalogs
- corporate collections
- combinatorial chemistry

unlabeled point \equiv description of chemical compound

label \equiv active (binds to target) vs. inactive

getting a label \equiv chemistry experiment

Active Learning example: pedestrian detection



- **Unsupervised learning:** no labeled data
- **Supervised learning:** learn using labeled data for prediction on unseen points
- **Semi-supervised learning:** learn using labeled and unlabeled data for prediction on unseen points

Iris flower data set [Fisher, 1936]

- Quantify the morphologic variation of Iris flowers of three related species (*Iris setosa*, *Iris virginica* and *Iris versicolor*) by four features: the length and the width of the sepals and petals, in centimetres
- $d = 4$ features, $K = 3$ classes (mono-label case), $m = 150$

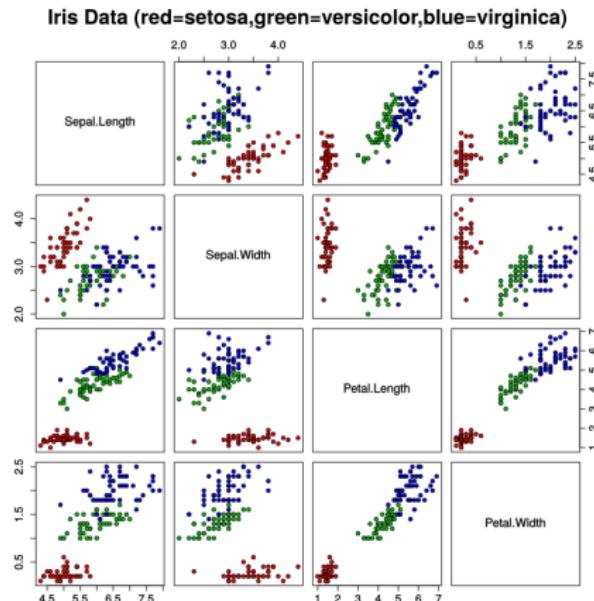


Figure – Scatterplot of the data set [Wikipedia]

- **Hypothesis set** $F \subset Y^X$ is a subset of functions out of which the learner selects his hypothesis
 - represents a prior knowledge about the task at hand
 - depends on available features
- Predictive model belongs to a parametric family of functions

$$F = \{f(\mathbf{x}; \boldsymbol{\theta}) | \boldsymbol{\theta} \in \Theta\},$$

where

- $f : X \times \Theta \rightarrow Y$ is a fixed family of functions,
- Θ is a feasible set of parameters
- Example: linear model with $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)$, $\Theta = \mathbb{R}^p$, and features $\{\phi_j(\mathbf{x})\}_{j=1}^p$
 - regression $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{j=1}^p \theta_j \phi_j(\mathbf{x})$, $Y = \mathbb{R}^1$
 - classification $f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}\left(\sum_{j=1}^p \theta_j \phi_j(\mathbf{x})\right)$, $Y = \{-1, +1\}$

Example: Linear regression

- $X = Y = \mathbb{R}^1$, $m = 200$, $d = 3$ features: $\{x, x^2, 1\}$ or $\{x, \sin x, 1\}$

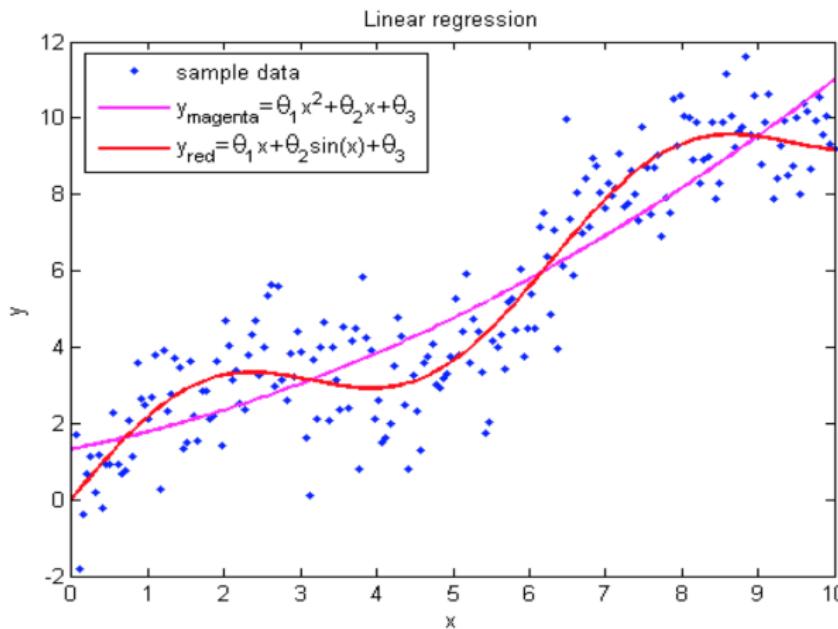


Figure – Linear regression [Vorontsov]

- **Training data:** sample S_m of size m drawn *i.i.d.* according to distribution D on $X \times Y$

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

- **Problem** find hypothesis $\hat{f} \in F$ with small generalization error
 - deterministic case: $y = f(\mathbf{x})$ is a deterministic function, only $\mathbf{x} \sim D$
 - stochastic case: output is a probabilistic function of input, e.g. $y = f(\mathbf{x}) + \varepsilon$

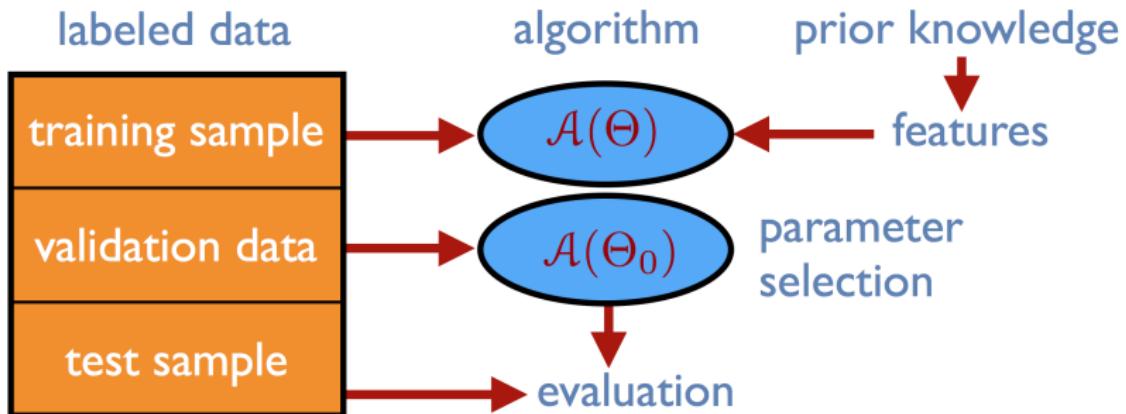


Figure – Learning scheme of predictive model [Mohri, 2016]

- ML algorithm $\mathcal{A} : (X \times Y)^m \rightarrow F$
- Train phase: using algorithm \mathcal{A} we get $\hat{f} = \mathcal{A}(S_m)$
- Test phase: calculate predictions $\hat{f}(\mathbf{x}'_i)$ for new \mathbf{x}'_i , $i = 1, \dots, m'$

Loss function $L : Y \times Y \rightarrow \mathbb{R}$

- $L(y, \hat{y})$ is the error of predicting \hat{y} instead of y
- 0 – 1 loss $L(y, \hat{y}) = 1_{y \neq \hat{y}}$ in case of binary classification
- $L(y, \hat{y}) = (y - \hat{y})^2$ in case of regression with $Y \subseteq \mathbb{R}$

- **Empirical error** for $f \in F$ and sample S_m

$$\widehat{R}(f) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}_i), y_i)$$

- **Generalization error:** for $f \in F$

$$R(f) = \mathbb{E}_{(\mathbf{x}, y) \sim D} [L(f(\mathbf{x}), y)]$$

- **Bayes error**

$$R^* = \inf_f R(f)$$

- **Noise:**

- in regression for any $\mathbf{x} \in X$, L_2 -loss and white noise model

$$f^* = \mathbb{E}(y|\mathbf{x})$$

- observe that

$$R^* = \text{Var}(\varepsilon)$$

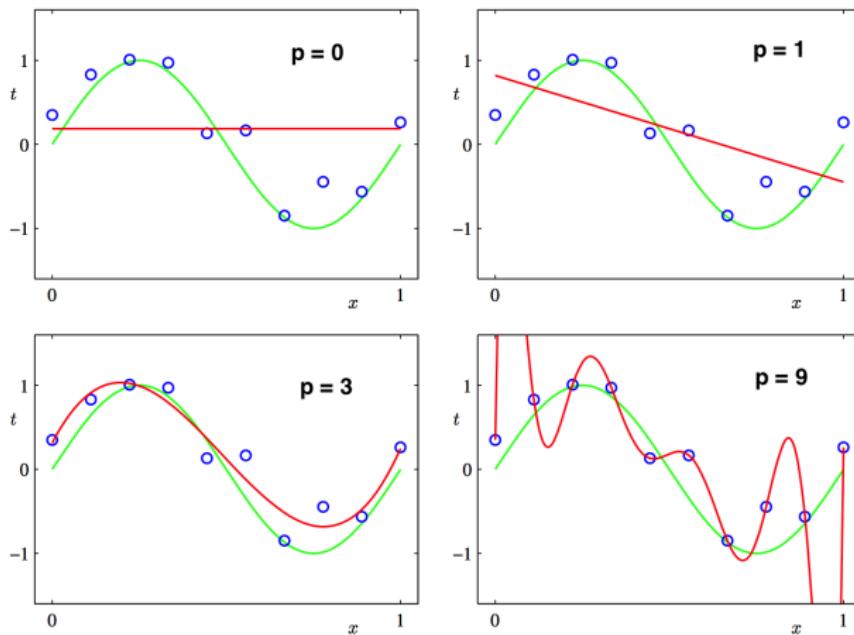


Figure – Notion of simplicity/complexity. How do we define complexity? [Bishop]

- A set of polynomials $f(x; \theta) = \sum_{j=0}^p \theta_j x^j$
- Squared loss function $L(\hat{y}, y) = (\hat{y} - y)^2$

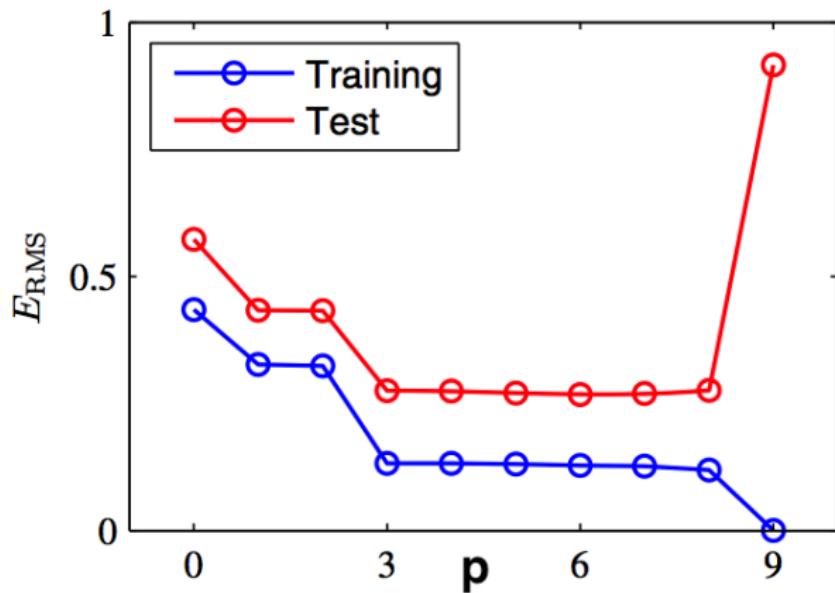


Figure – Notion of simplicity/complexity. How do we define complexity? [Bishop]

Overfitting: error on the test set \gg error on the train set

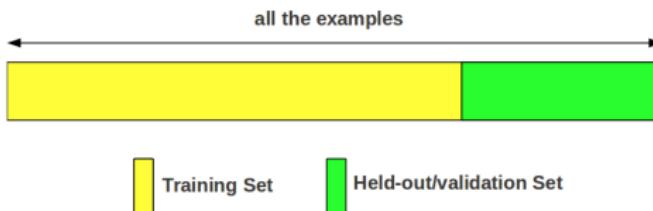
- Select Hypothesis set F
- Find hypothesis $f \in F$ minimizing empirical error

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

- F may be too complex
- Sample size may not be large enough

- the best hypothesis on the sample may not be the best overall
- generalization is not memorization
- complex rules can provide poor predictions
- trade-off: complexity vs. sample size (underfitting/overfitting)
- detection: use cross-validation for random splitting in train/test subsets

- Set aside a fraction (say 10% – 20%) of the training data
- This part becomes our hold-out data (validation or development data)



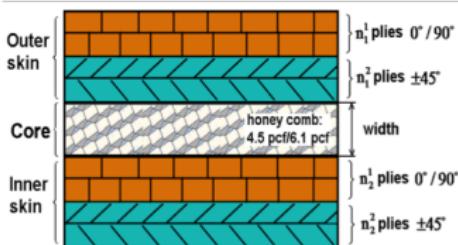
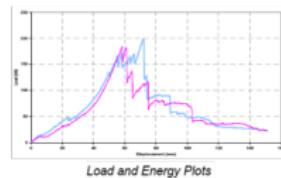
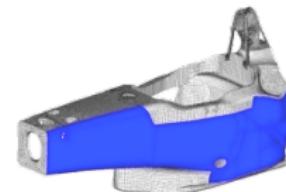
- Remember: hold-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the hold-out data
- Choose the model with the smallest hold-out error
- Problems:
 - Wastes training data, so typically used when we have plenty of training data
 - Hold-out data may not be good if there was an unfortunate split (use random splitting!)

- Main steps to solve applied problem
 - understand problem and data
 - feature engineering
 - model selection and reduction of the problem to optimization one
 - accuracy evaluation
 - business deployment and use
- When developing ML algorithms
 - use synthetic data to explore limits of the algorithm
 - use applied data (from available data storages) to test under various real conditions

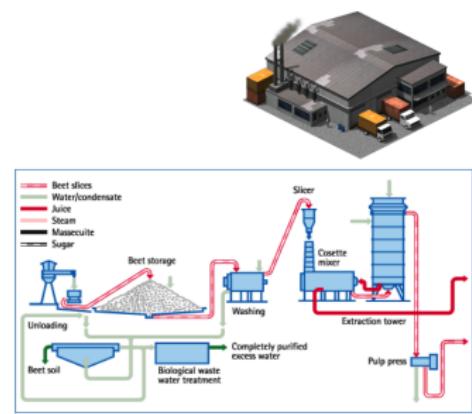
1 Main concepts

2 Examples

- **Objective:** Minimize mass of a side panel of Formula 1 car
 - w.r.t. number of layers, ply structure, thickness
 - subject to strength constraints
- **Challenges:** FEA of composite structures is of limited accuracy; experiments are expensive
- **Solution:** Construct a regression model using experimental and numerical data (data fusion), and use it in optimization;
- **ML problem setting:**
 - **Object:** side panel
 - **Features:** number of layers, types of layers, thicknesses
 - **Label:** panel strength (regression)
- **Result:** 10% mass reduction with fewer simulations or full order experiments



- **Problem:** Optimize efficiency of sugar extraction
 - **Object:** production of one batch of a product
 - **Input x :** beet chips shape, quality, temperature, sugar content and flux; wash water temperature, pH and flux; the temperature inside the diffuser, etc.
 - **Output y :** costs, losses, efficiency of sugar extraction
- **Challenges:**
 - Heterogeneous data and noise,
 - Large volumes of high-dimensional stream data,
 - Missing values, outliers, etc.

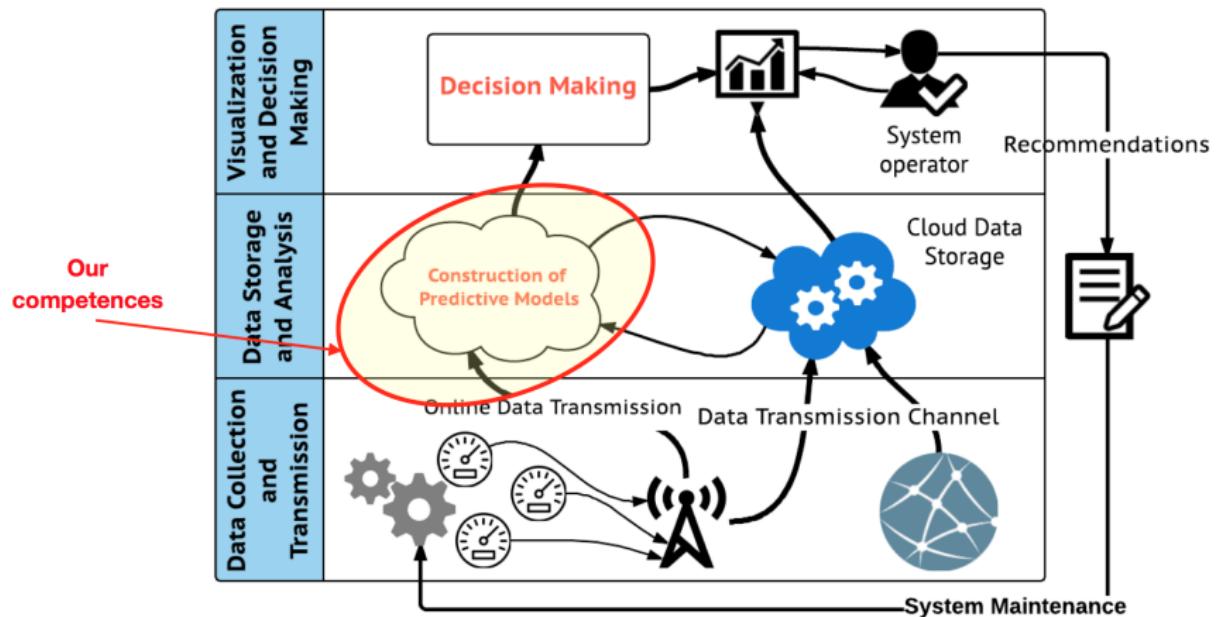


- **Problem:** Minimize fuel consumption of a cargo vessel, detect frauds with fuel, optimize expedition route
 - **Object:** historic fuel consumption for a particular vessel under given weather conditions for a particular part of its route
 - **Features x :** dimensions (length, height, width), load capacity, type (ferry, barge, freighter, etc.), number of engines, etc.; Route data; information about weather and sea currents (historical, predictive and real-time); controls (vessel speed, etc.)
 - **Label y :** fuel consumption



- **Challenges:**
 - Heterogeneous data and noise,
 - Large volumes of high-dimensional stream data,
 - Missing values, outliers, etc.

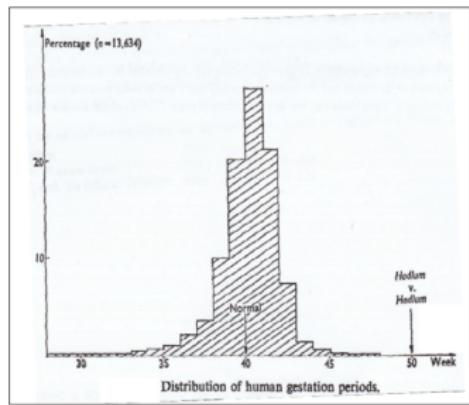
Place in an Industrial Process



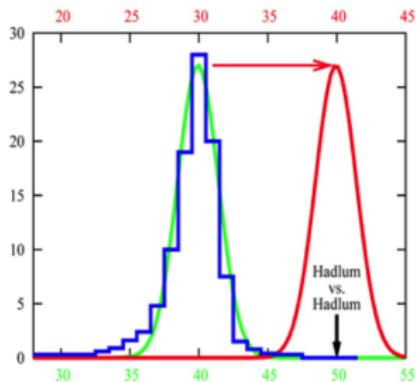
- **Object:** application for a loan of a particular person
- **Label:** 0/1 (bad/good)
- **Features:** sex, address, profession, position, education, work experience, age, salary, work experience, etc.
- **Challenges:** estimate default probability

- **Object:** customer at some moment of time
- **Label:** 0/1 (churn/not churn)
- **Features:** sex, address, subscription plan, already purchased services, frequency and duration of calls, etc.
- **Challenges:**
 - estimate churn probability
 - huge samples
 - laborious feature engineering

- **Object:** triplet $\langle user, Ad, banner \rangle$
- **Label:** click of a user on a context Ad, shown as a response to his request on a web-marketplace
- **Raw data:** user actions on a website, user profile (browser, device, etc.), history of user actions and clicks of other users on a banner, etc.
- **Challenges:**
 - huge samples (billions of ad displays)
 - outliers/incorrect measurements
 - non-homogeneous data (big cities vs. local towns)
 - laborious feature engineering
 - main efficiency criterion is a revenue of an Ad marketplace



- The birth of a child to Mrs. Hadlum happened 349 days after Mr. Hadlum left for military service
- Average human pregnancy period is 280 days (40 weeks)
- Statistically, 349 days is an outlier



- The birth of a child to Mrs. Hadlum happened 349 days after Mr. Hadlum left for military service
- Average human pregnancy period is 280 days (40 weeks)
- Statistically, 349 days is an outlier

“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism”

Applications:

- Early detection of fraud transactions
- Fault detection and failures localisation in technical systems (e.g. airplanes, cars)
- Cyber-security, zero-day attack detection

M -fold Cross-Validation

- Create M equal sized partitions of the training data
- Each partition has m/M examples
- Train using $M - 1$ partitions, validate on the remaining partitions
- Repeat the same M times, each with a different validation partition



- Finally, choose the model with smallest average validation error
- Usually M is chosen as $4 - 10$

$N \times M$ -fold Cross-Validation

- We divide the training sample N times into M equal sized partitions
- Results of all N M -fold cross-validations are aggregated (e.g. averaged)
- By increasing N we can improve accuracy
- Each object is used in a test set N times
- We can construct confidence intervals using results of N repetitions

- Consider an infinite sequence of hypothesis sets ordered for inclusion

$$F_1 \subset F_2 \subset \dots \subset F_n \subset \dots$$

$$f = \arg \min_{f \in F_n, n \in \mathbb{N}} \widehat{R}(f) + \text{penalty}(F_n, m)$$

- Strong theoretical guarantees
- Typically computationally intensive

- Empirical risk minimization (ERM)

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

- Structural Risk Minimization (SRM) for $F_n \subseteq F_{n+1}$

$$f = \arg \min_{f \in F_n, n \in \mathbb{N}} \widehat{R}(f) + \text{penalty}(F_n, m)$$

- Regularization-based algorithms

$$f = \arg \min_{f \in F} \widehat{R}(h) + \lambda \|f\|^2, \lambda > 0$$