

Intro to ML. Part 3

Evgeny Burnaev, Alexey Zaytsev

Skoltech, Moscow, Russia



- 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**

*for
practical applications*

- object*
- Example: item, instance of data, related to some object

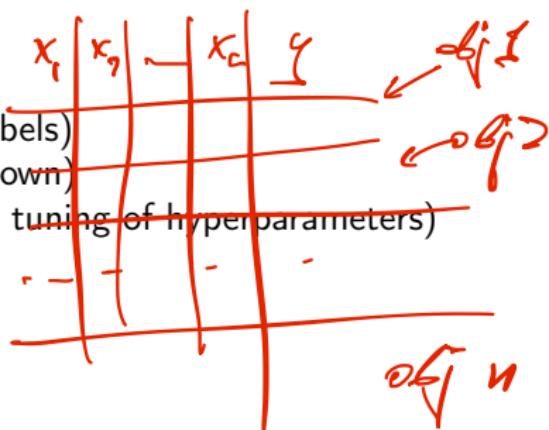
- we pass through*
- Features x : input value (input parameters, an input vector, attributes, a point) characterizing an object

what we try to predict

- 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)



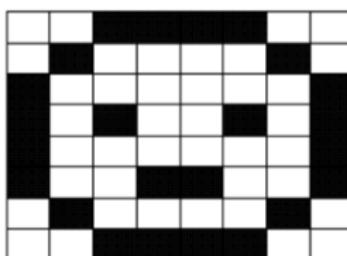
Formal

- 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

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:

$x \in X$
 $y \in Y$

set of pairs

- 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 $\{y_1, \dots, y_m\} \in Y, y_i = f(\mathbf{x}_i)$

set of labels

- 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

x
sample

- **Object** o : place to open a new restaurant
- **Label** y : revenue after one year of operation
- **Features** $x = x(o)$: 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)

*Based on all $x(o)$ and corresponding y
we can estimate if y is good for o 's*

- Usually as object x description we consider a set of features, i.e.
 $x = (x_1, \dots, x_d)$

- There are several types of features

- binary, e.g. $x_j \in \{0, 1\}$
- nominal, e.g. $x_j \in \{\text{red, green, blue}\}$
- ordinal, e.g. $x_j \in \{\text{low, middle, high}\}$
- real number, i.e. $x_j \in \mathbb{R}$

- Thus objects $\{x_1, \dots, x_m\} \in X$ can be represented as an $m \times d$ matrix

If 2 features have high correlation, including both in a linear model
=> computational instability

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

specific object

But in general, depends on problem. *specific feature*

- 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

Scenarios

Settings

Queries

- Settings

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

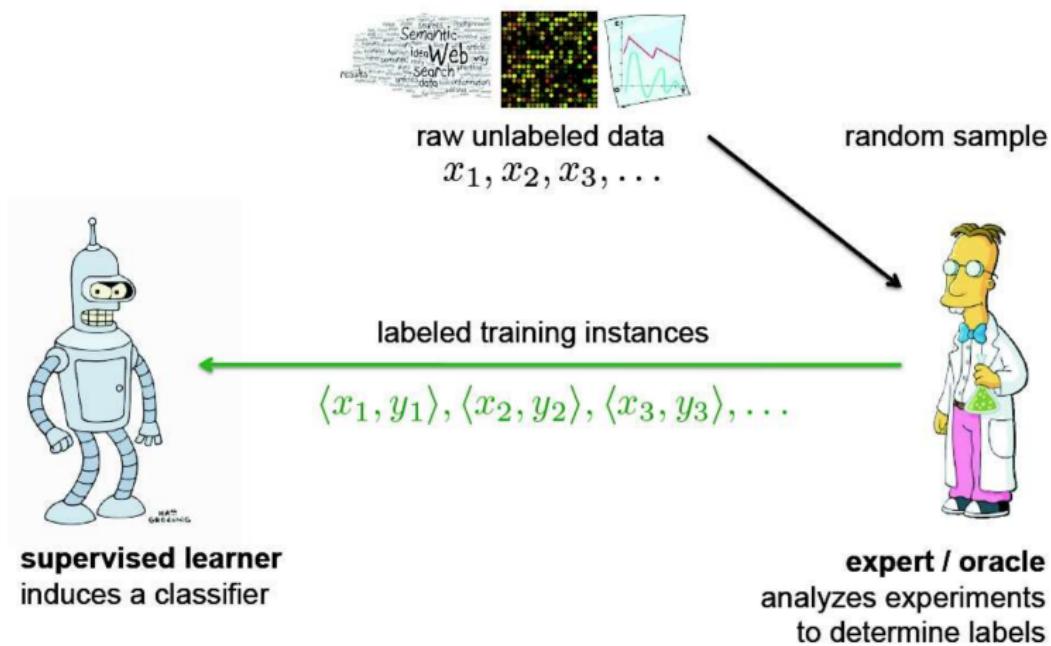
on-line: a learner receives one sample at a time and makes prediction for that sample

- Queries

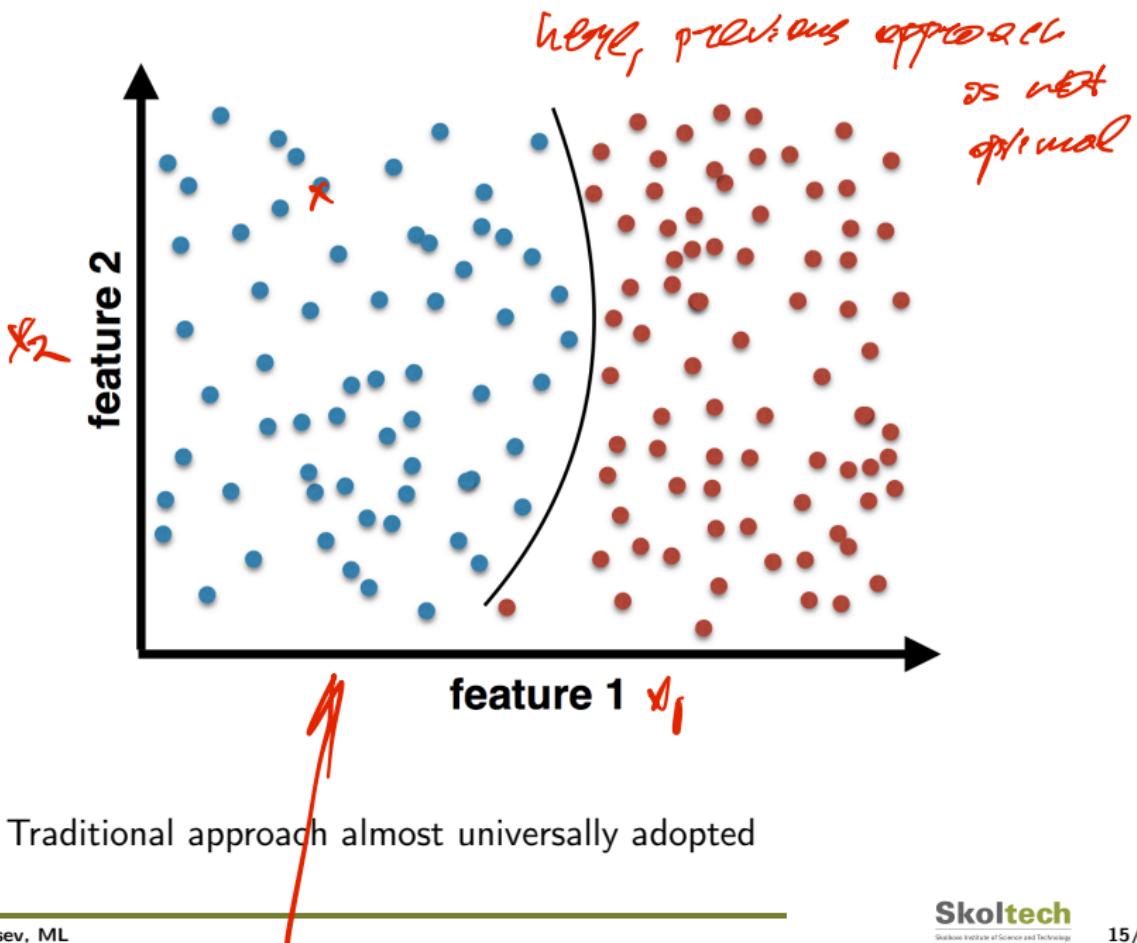
active: a learner can request the label of a point

passive: a learner always receives labeled points

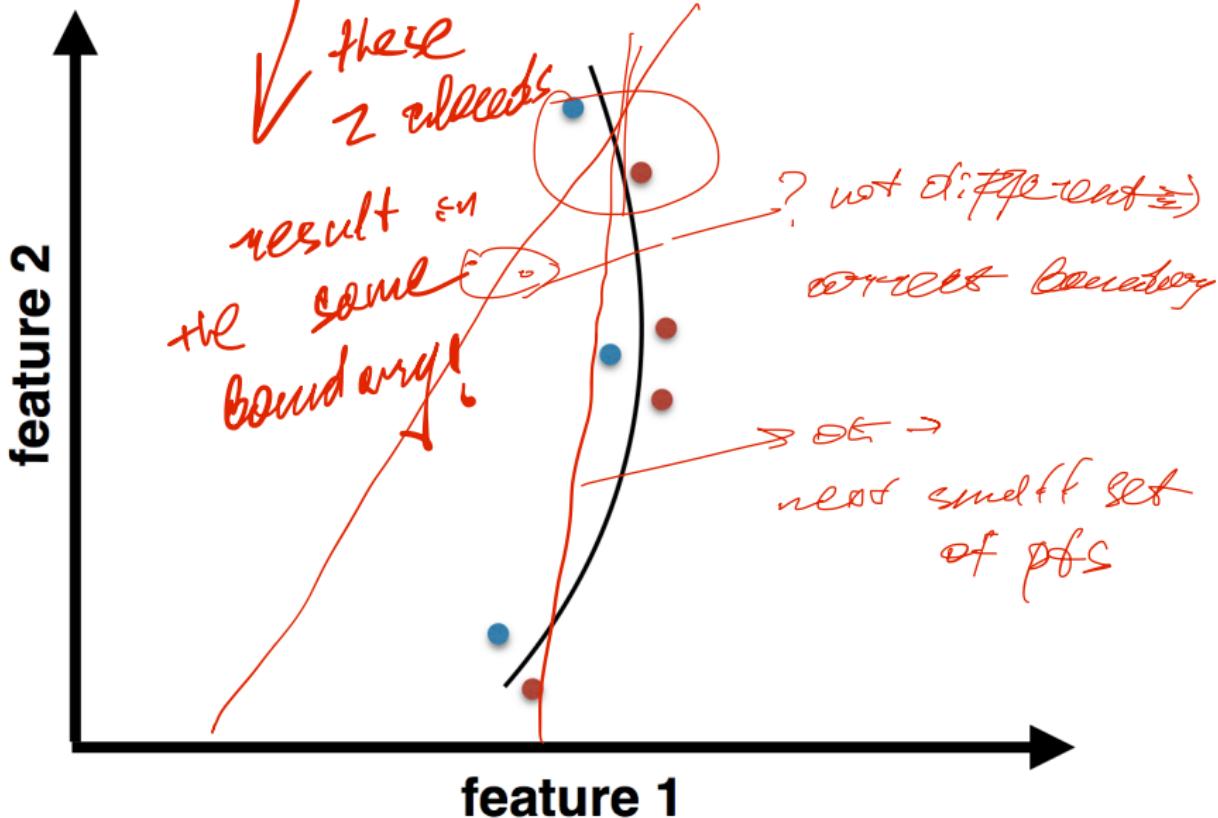
(Passive) Supervised Learning



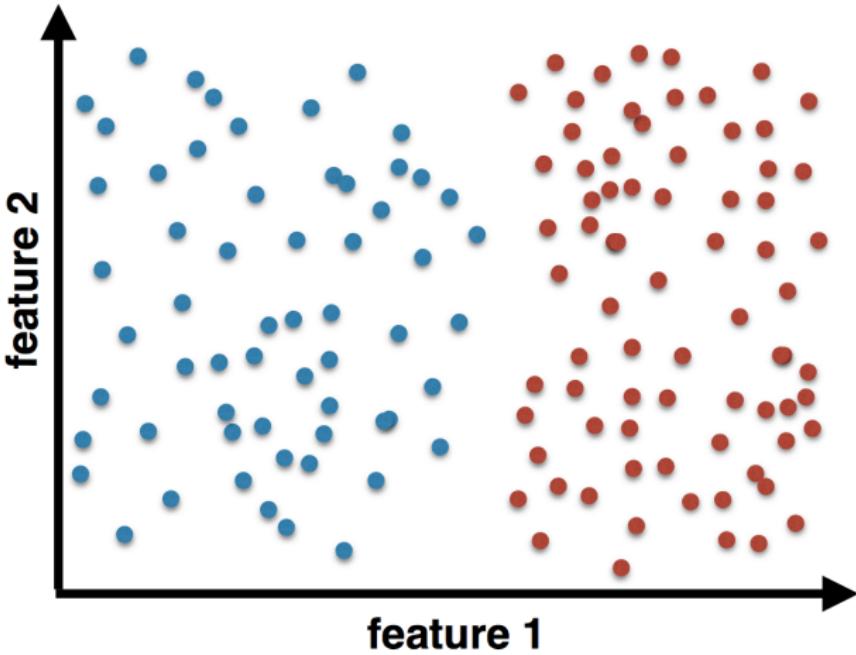
So what's wrong with Supervised Learning



Well, we actually only needed this!

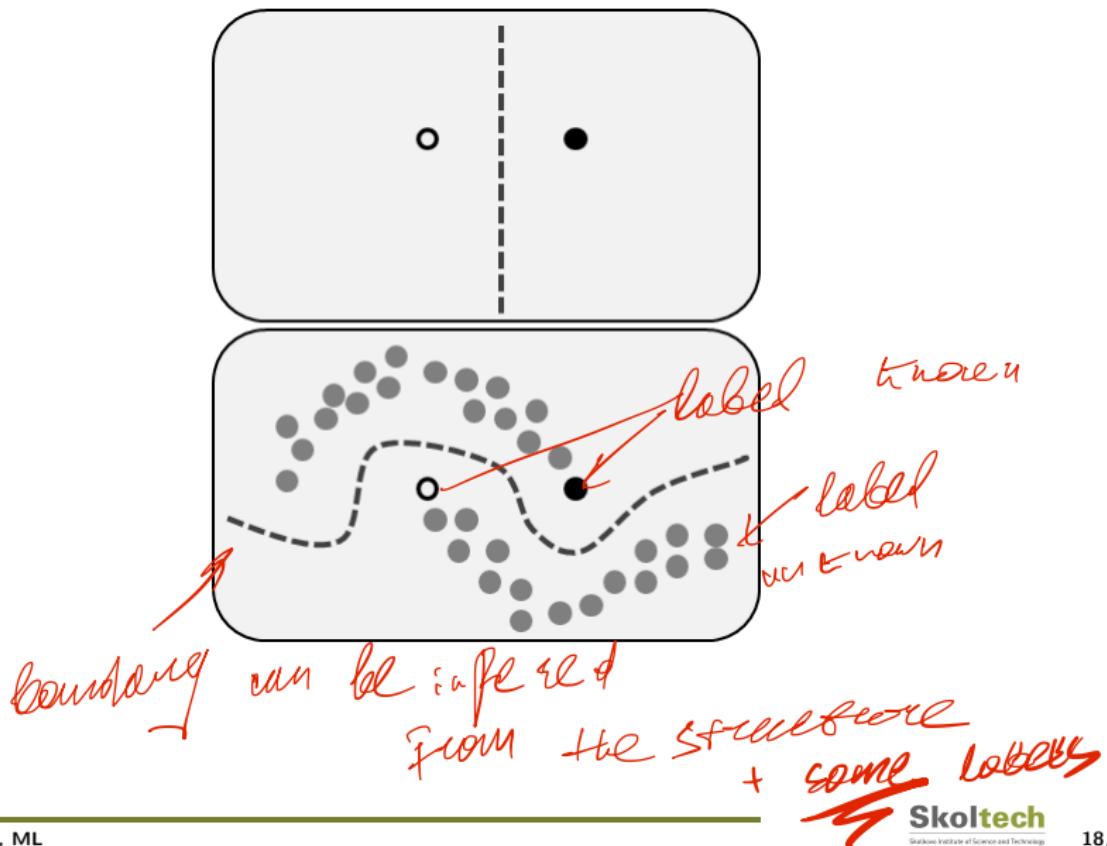


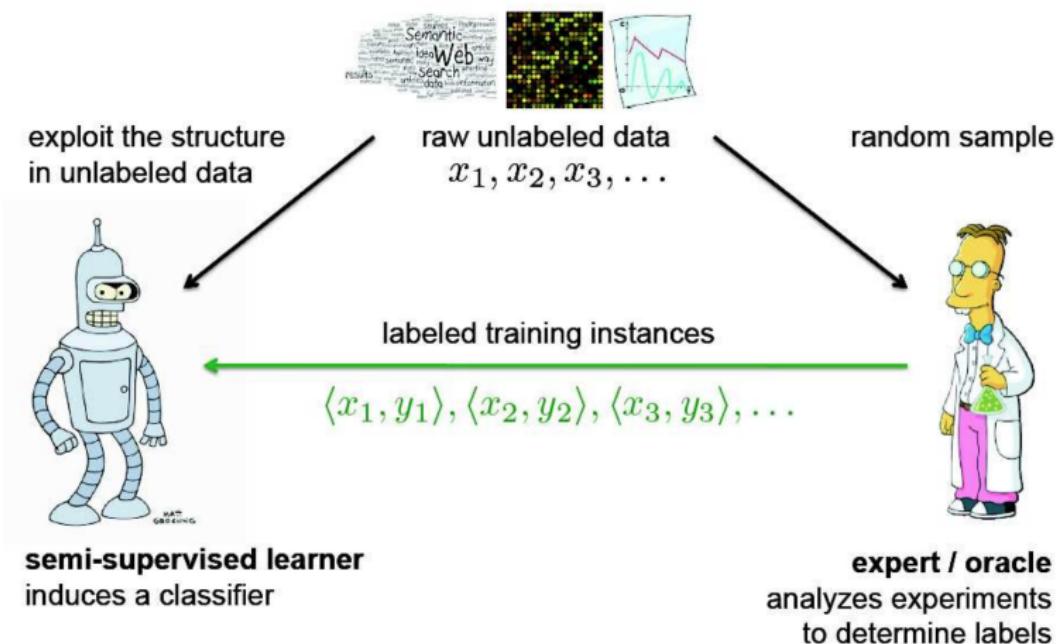
So this was a complete waste of time!

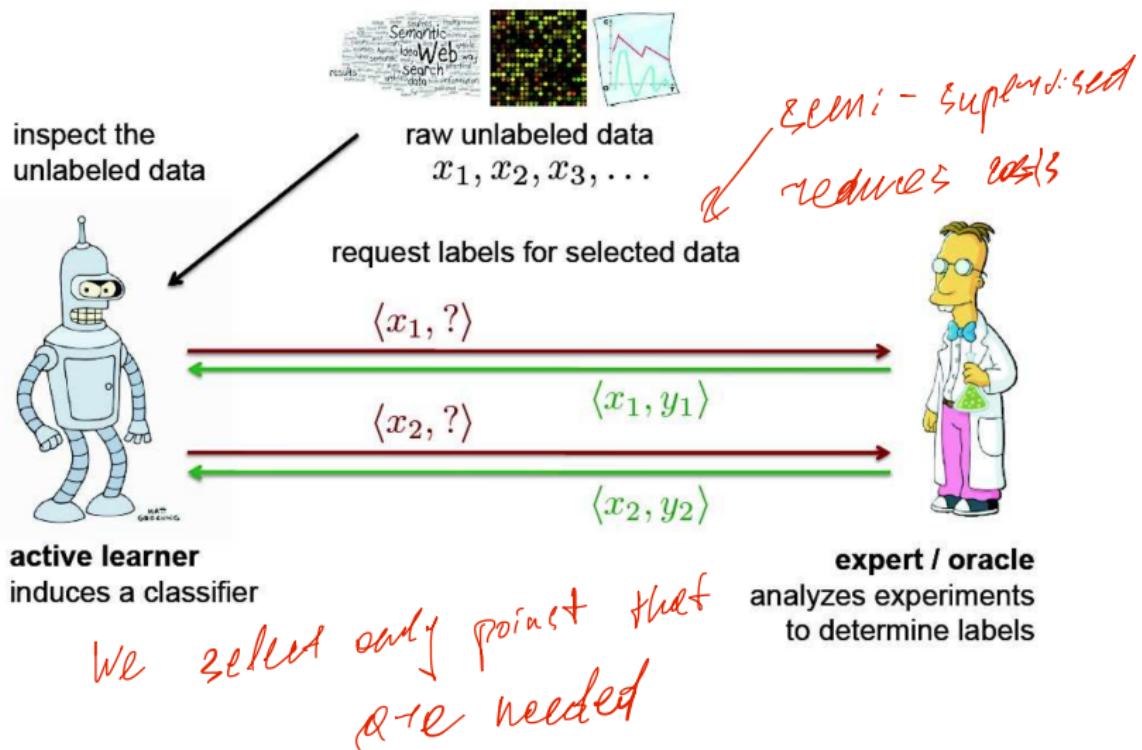


- Random sampling inevitably leads to redundancy

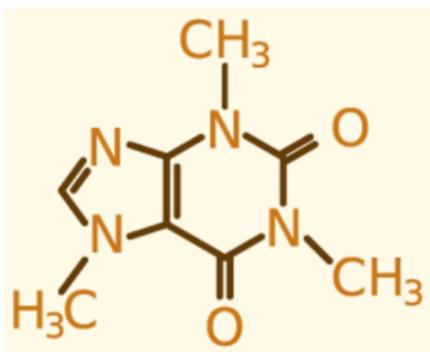
\Rightarrow active queries







- 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

- **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

*q
set of weights*

- $f : X \times \Theta \rightarrow Y$ 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 $\{x, \sin x, 1\}$

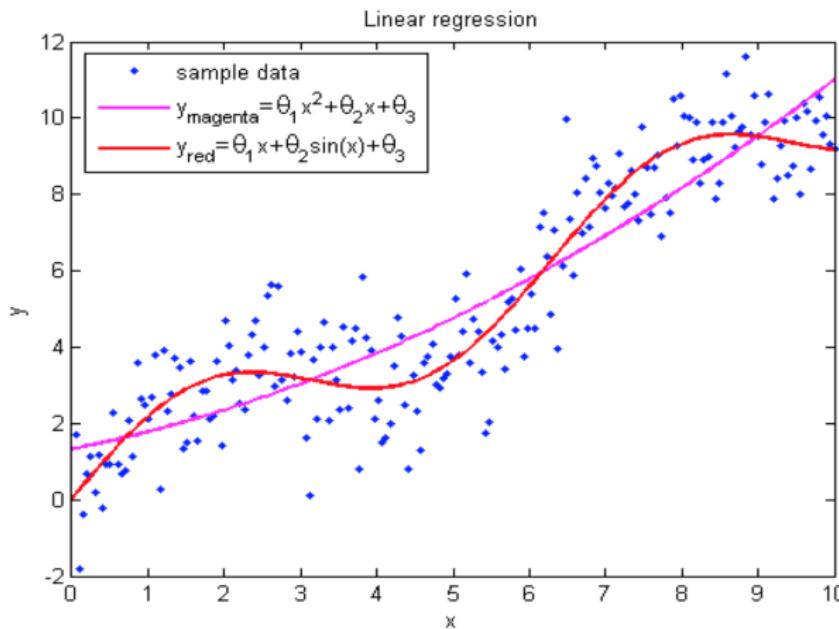


Figure – Linear regression [Vorontsov]

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

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

Set of m samples

- **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. $y = f(\mathbf{x}) + \varepsilon$

\downarrow
+ noise

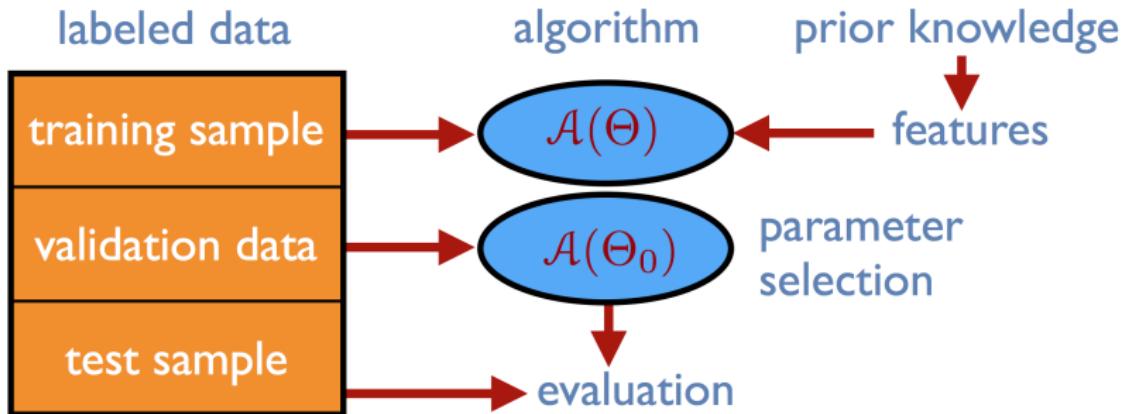


Figure – Learning scheme of predictive model [Mohri, 2016]

- 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 , $i = 1, \dots, m'$

to estimate quality of the model

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

- $L(y, \hat{y})$ is the error of predicting \hat{y} instead of y

binary • 0 – 1 loss $L(y, \hat{y}) = 1_{y \neq \hat{y}}$ in case of binary classification

squared • $L(y, \hat{y}) = (y - \hat{y})^2$ 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**

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

*Avg over the
distribution
of data*

skipped

- **Noise:**

- data generation process: $y = f(\mathbf{x}) + \varepsilon$, ε is a white noise
 - in regression for any $\mathbf{x} \in X$, L_2 -loss and white noise model

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

- observe that

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

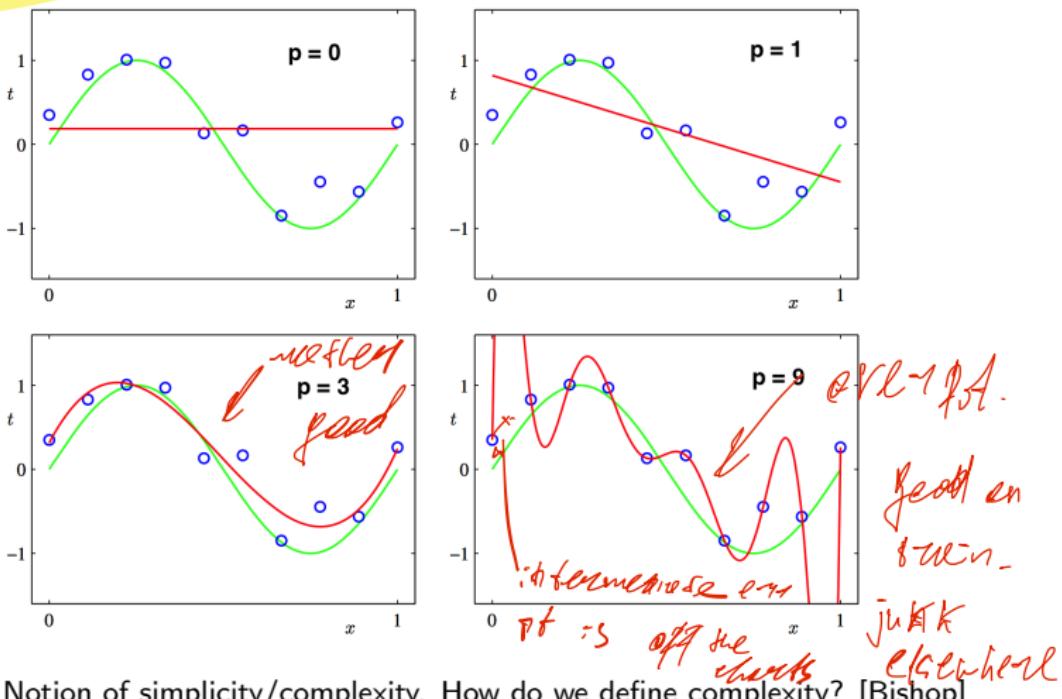


Figure – Notion of simplicity/complexity. How do we define complexity? [Bishop]

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

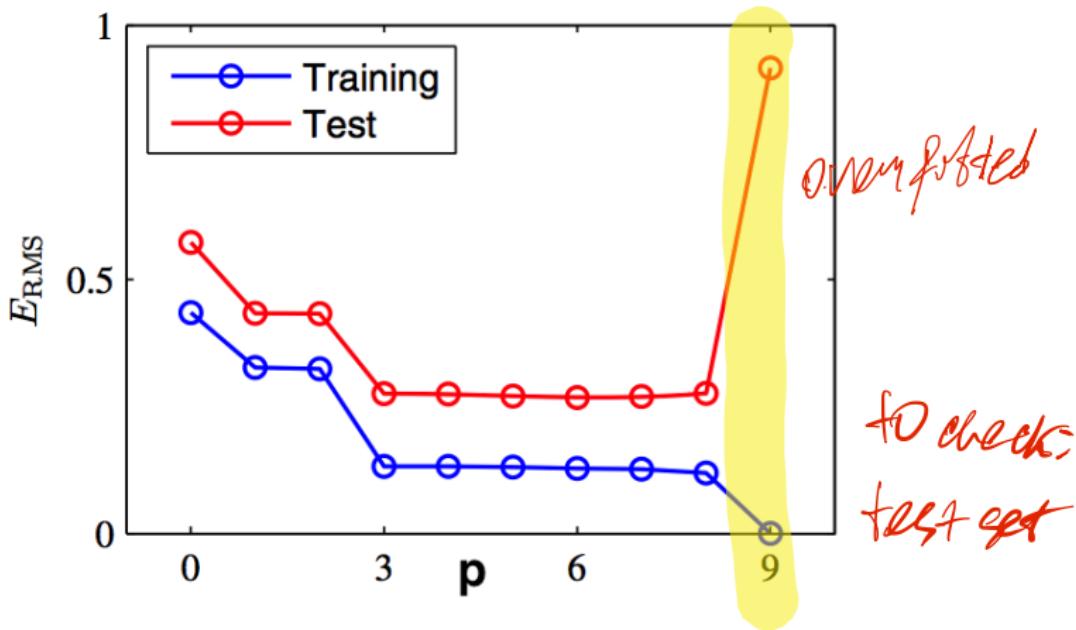


Figure – Notion of simplicity/complexity. How do we define complexity? [Bishop]

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

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

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

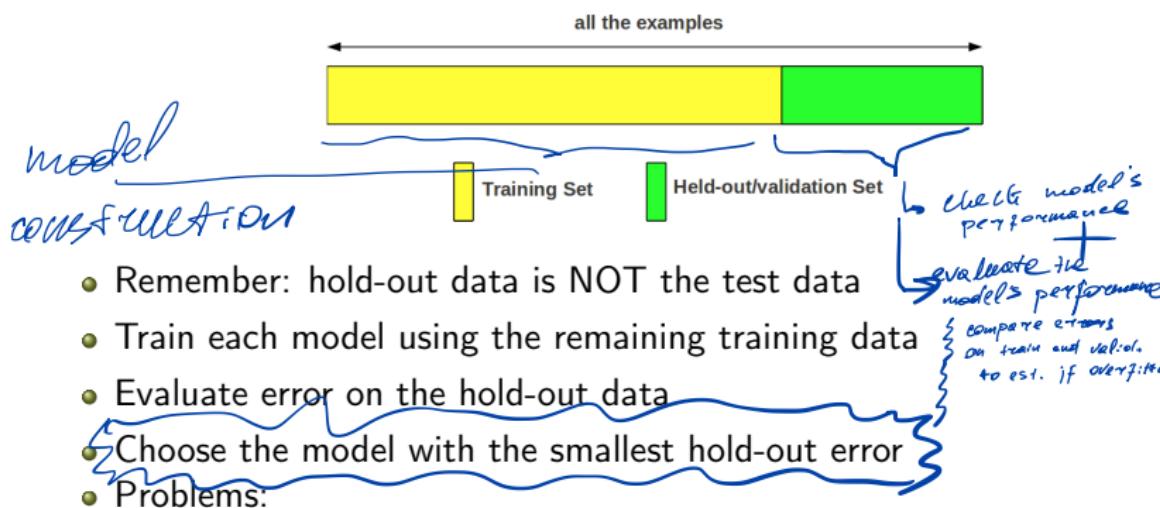
min. empirical risk

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

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

- 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

k-fold

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 × k - fold *Σ do k-fold N times*

N × 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

↳ avoiding overfitting by balancing complexity vs goodness of fitting on train

- 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

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