Introduction to Machine Learning
NPFL 054

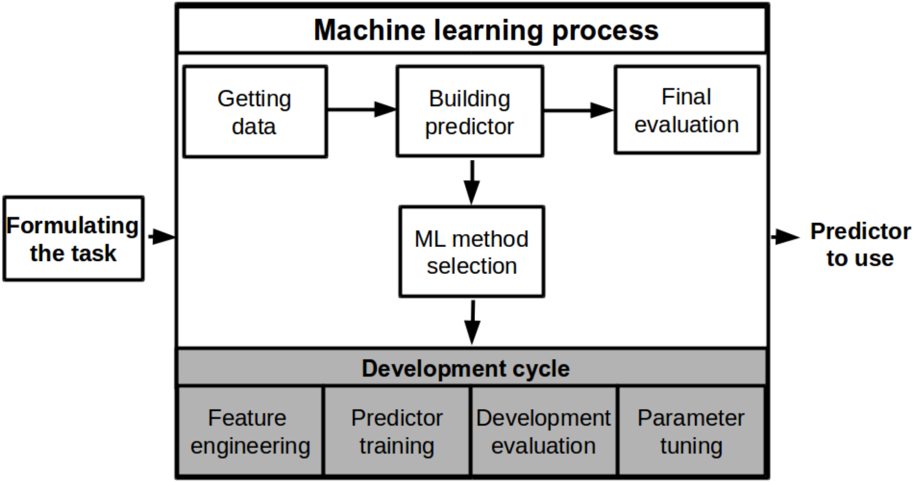
http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Machine learning overview



Machine learning overview

machine learning = representation + evaluation + optimization

representation	**evaluation**	**optimization**
instances	**evaluation function**	**combinatorial**
k-NN	accuracy/error rate	greedy search
	precision, recall	
	ROC curve	
hyperplanes	**objective function**	**continuous**
Naïve Bayes	discriminative	*unconstrained*
	(conditional probability)	gradient descent,
logistic regression	generative	maximum likelihood estimation
	(conditional probability)	*constrained*
SVM	margin	quadratic programming
perceptron	mean square error	
decision trees		
graphical models		
Bayesian networks		

Machine learning overview

Task and data management

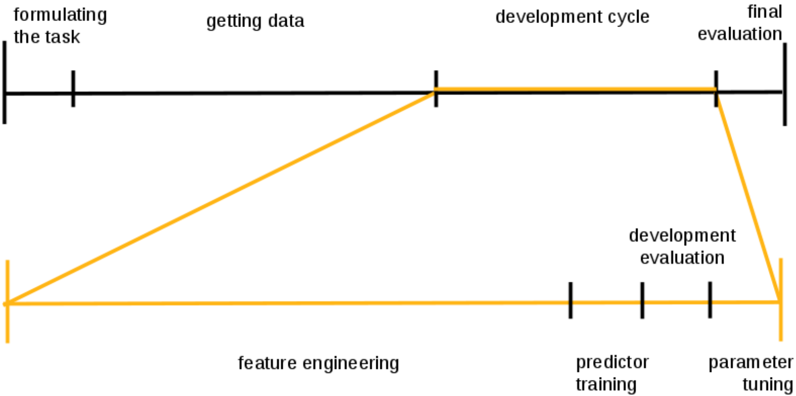
1. Time management
2. Formulating the task
3. Getting data
4. The more data, the better
5. Feature engineering
6. Curse of dimensionality

Methods and evaluation

7. Learning algorithms
8. Development cycle
9. Evaluation
10. Optimizing learning parameters
11. Overfitting
12. The more classifiers, the better
13. Theoretical aspects of ML

(1) Time management

How much time do particular steps take?



(2) Formulating the task

- Precise formulation of the task
- What are the objects of the task?
- What are the target values of the task?

(3) Getting data

- Gather data
- Assign true prediction
- Clean it
- Preprocess it
- Analyse it

(4) The more data, the better

If we don't have enough data

- **cross-validation** – The data set *Data* is partitioned into subsets of equal size. In the i-th step of the iteration, the i-th subset is used as a test set, while the remaining parts from the training set.

- **bootstrapping** – New data sets $Data_1$, ..., $Data_k$ are drawn from *Data* with replacement, each of the same size as *Data*. In the i-th iteration, $Data_i$ forms the training set, the remaining examples in *Data* form the test set

(5) Feature engineering

- Understand the properties of the objects
  - How they interact with the target value
  - How they interact each other
  - How they interact with a given ML algorithm
  - Domain specific
- Feature selection manually
- Feature selection automatically: generate large number of features and then filter some of them out

(6) Curse of dimensionality

- A lot of features $\longrightarrow$ high dimensional spaces
- The more features, the more difficult to extract useful information
- Dimensionality increases $\longrightarrow$ predictive power of predictor reduces
- The more features, the harder to train a predictor
- **Remedy**: feature selection, dimensionality reduction

(7) Learning algorithms

Which one to choose?

First, identify appropriate learning paradigm

- Classification? Regression?
- Supervised? Unsupervised? Mix?
- If classification, are class proportions even or skewed?

In general, **no learning algorithm dominates all others on all problems**.

(8) Development cycle

- Test developer's expectation
- What does it work and what doesn't?

(9) Evaluation

Model assessment

- **Metrics** and **methods** for performance evaluation
  How to evaluate the performance of a predictor?
  How to obtain reliable estimates?

- **Predictor comparison**
  How to compare the relative performance among competing predictors?

- **Predictor selection**
  Which predictor should we prefer?

(10) Optimizing learning parameters

Searching for the best predictor, i.e.

- adapting ML algorithms to the particulars of a training set
- optimizing predictor performance

Optimization techniques

- Greedy search
- Beam search
- Grid search
- Gradient descent
- Quadratic programming
- ...

(11) Overfitting

- bias
- variance

To avoid overfitting using

- cross-validation
- feature engineering
- parameter tuning
- regularization

(12) The more classifiers, the better

- **Build an ensemble of classifiers** using
  - different learning algorithm
  - different training data
  - different features
- **Analyze** their performance: complementarity implies potential improvement
- **Combine** classification results (e.g. majority voting).

Examples of ensemble techniques

- **bagging** works by taking a bootstrap sample from the training set
- **boosting** works by changing weights on the training set

(13) Theoretical aspects

Computational learning theory (CLT) aims to understand fundamental issues in the learning process. Mainly

- How computationally hard is the learning problem?
- How much data do we need to be confident that good performance on that data really means something? I.e., accuracy and generalization in more formal manner
- CLT provides a formal framework to formulate and address questions regarding the performance of different learning algorithms. Are there any general laws that govern machine learners? Using statistics, we compare learning algorithms empirically

(13) Theoretical aspects
PAC learning

Probably Approximately Correct (PAC) learning framework is a part of CLT.

- *Sample complexity* (i.e. data requirements) How many training examples are needed for a learner to converge with high probability to a successful hypothesis?
- *Computational complexity* How much computational effort is needed for a learner to converge with high probability to a successful hypothesis?

(13) PAC learning

- Set of instances $X = \{\mathbf{x} : \mathbf{x} = \langle x_1, ..., x_m \rangle, x_i \in A_i\}$
- Output values $Y = \{0, 1\}$
- Training data $Data = \{\langle \mathbf{x}_i, y_i \rangle, \mathbf{x}_i \in X, y_i \in Y\}_{i=1}^{n}$
- C set of target concepts $c : X \rightarrow Y$

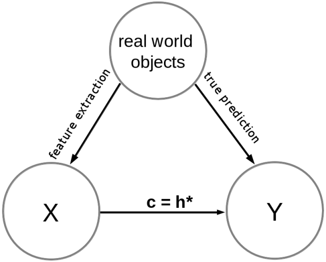
Instances are generated at random from X according to some probability distribution $\mathcal{D}$.

Assumptions

1. $\mathcal{D}$ is unknown
2. $\mathcal{D}$ is stationary, i.e. it does not change over time
3. Instances are sampled independently of each other (they are Identically and Independently Distributed)

(13) PAC learning

- A set H of possible hypotheses $h \in H : h : X \rightarrow Y$.



- A learner L outputs some hypothesis $h \in H$ as an approximation of c.

(13) PAC learning

How closely the hypothesis h approximates the target function c?

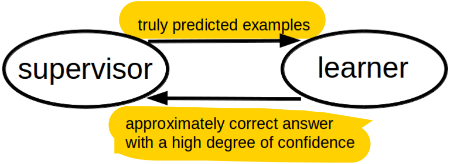
Generalization error

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

It is error with which h approximates c.

(13) PAC learning

Goal: Characterize classes of target concepts that can be *reliably* learned from a *reasonable number* of randomly drawn training examples and by a *reasonable amount* of computation.



(13) PAC learning

- For a specific learning algorithm, what is the probability that a concept it learns will have an error that is bound by ϵ?
  $error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)] < \epsilon$
- Set a bound δ on the probability that this error is greater than ϵ:
  $\Pr[error_{\mathcal{D}}(h) > \epsilon] < \delta$

- When a learned concept is good?
- Different degrees of "goodness" will correspond to different values of ϵ and δ.
- The smaller ϵ and δ are, the better the learned concept will be.

(13) PAC learning

Definition

Consider a concept class C defined over a set of instances X (m is the instance size, i.e. the size of instance representation) and a learner L using hypothesis space H.

C is **efficiently PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, $0 < \epsilon$, $0 < \delta$, learner L will output a hypothesis $h \in H$

- with $\Pr(error_{\mathcal{D}}(h) \leq \epsilon) \geq 1 - \delta$
  – probability at least $1 - \delta$ (confidence) such that $error_{\mathcal{D}}(h) \leq \epsilon$,
- in time $O(poly(\frac{1}{\epsilon}, \frac{1}{\delta}, size(c), m))$
  – m and $size(c)$ are the representation costs for the instances and the concepts, resp.,
- using n training examples where n is $poly(\frac{1}{\epsilon}, \frac{1}{\delta})$

(13) PAC learning

Two things are required from L:

1. L must output, with arbitrarily high probability $1-\delta$, a hypothesis having arbitrarily low error ϵ.
2. It must do efficiently in time that grows at most polynomially with $\frac{1}{\epsilon}, \frac{1}{\delta}$, with m and using polynomial number of training examples.

In other words, to show that some class C of target functions is PAC learnable, we have to show that

1. each $c \in C$ can be learned from polynomial number of training examples,
2. the processing time per example is polynomially bounded.

(13) PAC learning
Sample complexity

How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis? Express it in terms of

- size of the hypothesis space $|H|$
- Vapnik-Chervonenkis dimension $VC(H)$.

Sample error sample S, $|S| = n$

$$error_S(h) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\hat{y}_i \neq y_i)$$

Consistent hypothesis $Consistent(h, Data)$ iff $error_{Data}(h) = 0$

Agnostic hypothesis $error_{Data}(h) \neq 0$

(13) PAC learning
Sample complexity

- **Definition**

  A **dichotomy** of a set S is a partition of S into two disjoint subsets.

- **Definition**

  A set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H with this dichotomy.

- **Definition**

  The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

(13) PAC learning
Sample complexity

Consistent hypothesis

$$|H|e^{-\epsilon n} \leq \delta \rightarrow n \geq \frac{1}{\epsilon}(\ln|H| + \ln(\frac{1}{\delta}))$$

$$n \geq \frac{1}{\epsilon} \ln \frac{|H|}{\delta} \tag{1}$$

$$n \geq \frac{1}{\epsilon}(4\log_2(\frac{2}{\delta}) + 8VC(H)\log_2(\frac{13}{\epsilon}))$$

Agnostic hypothesis

$$n \geq \frac{1}{2\epsilon^2} ln \frac{|H|}{\delta}$$

References

- Pedro Domingos. A Few Useful Things to Know about Machine Learning. 2012.
  - https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
- Pedro Domingos. Ten Myths About Machine Learning. 2016.
  - https://medium.com/@pedromdd/ten-myths-about-machine-learning-d888b48334a3