Computational learning theory.
PAC learning. VC dimension.

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

Concept

Examples of a *concept*:

- even number, four-wheel vehicle, active politician, smart man, correct hypothesis

Why does it make sense to introduce *concepts*?

- What is the difference between odd and even numbers? What is the difference between active politicians and the rest?

Domain X is a set of all possible object instances:

- set of all whole numbers, all vehicles, all politicians, ...

Object $x \in X$ is described with values of some features:

- number {value}
- vehicle {manufacturer, engine type, number of doors, ...}
- politician {number of votings in the parliament, number of law proposals, number of law amendment proposals, number of interpellations, ...}

Target concept $c \in C$ corresponds to certain subset of X, $c \subseteq X$:

- each instance of $x \in X$ is either an *example* or a *counter-example* of a concept c
- characteristic function $f_c : X \rightarrow \{0, 1\}$
  - if $f_c(x) = 1$, x is a positive example for concept c
  - if $f_c(x) = 0$, x is a negative example (counter-example) of concept c
- Concept c is any boolean function f over X!

Hypothesis

Inductive learning task: find a hypothesis (model) h, which corresponds as much as possible to the target concept konceptu c, given

- a subset $D \subset X$ of examples (and counter-examples) of the target concept (training data) and
- the space H of all possible hypotheses.

Hypothesis is a candidate description of the target concept.

- H is the space of all possible hypotheses.
- In the most general case, even the hypothesis h may be any boolean function $h : X \rightarrow \{0, 1\}$.
- Similar to concept, a hypothesis h is a subset of X, $h \subseteq X$, as well.

The goal of learning:

- find a hypothesis h which is **correct for all examples** from X, i.e.

$$\forall x \in X : h(x) = c(x).$$

COLT: Goals

Computational learning theory (COLT) tries to theoretically characterize

1. the machine learning *problem complexity*, i.e.
   - under what circumstances learning is actually possible,
2. the *abilities of ML algorithms*, i.e.
   - under what circumstances, a learning algorithm is able to learn successfully.

COLT tries to answer questions like:

- Are there some problem complexity classes independently of the model/algorithm used?
- What type of model (class of hypotheses) should we use? Is there an algorithm which is consistently better then some other algorithm?
- How many training examples do we need so that a model (hypothesis) can be successfully learned?
- If the hypothesis space is large, is at actually possible to find the best hypothesis in a reasonable time?
- How complex should the resulting hypothesis be?
- If we find a hypothesis which is correct for all the training data $D \subset X$,
  how can we be sure that the hypothesis is also correct for the rest of the data $X \setminus D$???
- How many errors will the algorithm make before it learns the target concept successfully?

Generalization

Generalization ability:

- The ability of a learning algorithm to build a model which is able to correctly classify also the examples which were not part of the training data set D.
- It is measured as an error on $X \setminus D$.

Knowing nothing about the problem, is there any reason to prefer one algorithm over another?

Notation:

- $P_A(h)$: prior probability that algorithm A generates hypothesis h
- $P_A(h|D)$: probability that algorithm A generates h given the training data D:
  - in case of deterministic algorithms (nearest neighbors, decision trees, etc.), $P_A(h|D)$ is zero almost everywhere with the exception of a single hypothesis
  - in case of stochastic algorithms (e.g. neural network trained from random initial weights), the distribution $P_A(h|D)$ is non-zero for a larger subset of all hypotheses
- $P(c|D)$: the distribution of concepts consistent with training data D

If we do not know the target concept c, a natural measure of the algorithm generalization ability is the expected error over all concepts given the training data D:

$$E(\mathrm{Err}_A|D) = \sum_{h,c} \sum_{x \in X \setminus D} P(x) \cdot I(c(x) \neq h(x)) \cdot P_A(h|D) \cdot P(c|D)$$

Without knowing $P(c|D)$ we cannot compare 2 algorithms based on their generalization error!!!

Example

Assume that

- objects are described by 3 binary attributes,
- we have a single concept c, and
- 2 deterministic algorithms and their corresponding hypotheses h_1 and h_2: training data are memorized, one algorithm assigns new data to class +1, the other to class -1.

	x	c	h_1	h_2
D	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
$X \setminus D$	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

Given a concept c:

- $E(\text{Err}_{A_1}|c, D) = 0.4$, $E(\text{Err}_{A_2}|c, D) = 0.6$,
- algorithm A_1 is clearly better than A_2.

During the hypothesis building, we do not know the target concept c!

- Assuming we have no prior information about concept c, all concepts are equally probable.
- Training set D
  - allows us to eliminate all inconsistent hypotheses (224 in our case), but
  - it does not allow us to choose the right hypothesis among the consistent ones (in our case, there are 32 hypotheses remaining), because
  - averaged over all concepts c consistent with D, both hypotheses are equally successful!

No Free Lunch

"No Free Lunch" theorem: For any 2 algorithms A_1 and A_2 (represented by $P_{A_1}(h|D)$ and $P_{A_2}(h|D)$) the following statements hold independently of the sampling distribution $P(\boldsymbol{x})$ and independently of a particular training data set D:

1. Averaged over all concepts c, $E(\text{Err}_{A_1}|c, D) = E(\text{Err}_{A_2}|c, D)$.
2. Averaged over all distributions $P(c)$, $E(\text{Err}_{A_1}|c, D) = E(\text{Err}_{A_2}|c, D)$.

NFL corollaries:

- You can try hard to build one super algrithm and one terrible algorithm, but averaged over all concepts, both algorithms are equally good.
- If A_1 is better than A_2 on certain kind of problems, there must be other kind of problems where A_2 is better than A_1.
- All statements like "alg. 1 is better than alg. 2" are not saying anything about the algorithms, but rather about the set of concepts which were used to test the algorithms.
- In practice, for certain application area, we seek an algorithm which
  - works worse on problems we do not expect in the field, while
  - works well on problems which are highly probable.
- *Generalization is not possible without (often implicit) bias of the algorithm!*
- *The more the model assumptions correspond to the data, the better the generalization ability of the model!*

Bias

Inductive bias (předpojatost, zaujetí modelu):

- The sum of all (even implicit) assumptions the model makes about the application area.
- Taking advantage of these assumptions allows the model to generalize, i.e. to provide correct predictions even for unknown data, if the model assumptions correspond to reality.

Possible sources of model bias:

- Language bias:
  - The language of hypotheses does not need to correspond to the language of concepts.
  - Some concepts cannot be expressed in the hypotheses language.
  - Different language may allow for efficient learning.
- Preference bias:
  - Algorithm prefers some of the hypotheses consistent with D.
  - Algorithm may even choose a slightly inconsistent hypothesis.
  - Occam's razor
- ...

PAC learning

PAC learning

Probably Approximately Correct (PAC) learning:

- Characterizes the concept classes which are/are not learnable by certain class of hypotheses
  - using a "reasonable" number of training examples and
  - using an algorithm with "reasonable" computational complexity,

  for both
  - finite hypotheses spaces and
  - infinite hypotheses spaces (capacity, VC dimension).
- Defines a natural measure of complexity of the hypotheses spaces (VC dimension) which allows us to bound the required size of training data for inductive learning.

PAC learning assumptions:

- *Independence:* Examples $E_i = (x_i, c_i)$ are sampled independently, i.e. $P(E_i|E_{i-1}, E_{i-2}, \ldots) = P(E_i)$.
- *Identically distributed:* Future examples shall be sampled from a distribution equal to the one used for the previous examples: $P(E_i) = P(E_{i-1}) = \ldots$.
- Both conditions together are often denoted as "i.i.d." (independent and identically distributed).

(In this lecture we also assume that concept c is deterministic and that it is part of the hypotheses space H).

Hypothesis error rate

Real error rate of hypothesis h

- with regard to the target concept c and
- with regard to the distribution of examples $P(X)$ is

$$\mathrm{Err}(h) = \sum_{x \in X} I(h(x) \neq c(x)) \cdot P(x),$$

- i.e. it is the probability that the hypothesis classifies example x incorrectly.

Hypothesis h is **approximately correct** or **ϵ-approximately correct**,

- if $\mathrm{Err}(h) \leq \epsilon$,
- where ϵ is a small constant.

Is it possible to determine the number of training examples required to learn concept c with 0 error rate?

- If the set of training examples D is only a subset of X, there are still several hypotheses consistent with D (see NFL).
- Training examples are chosen randomly and can be misleading.

PAC framework

PAC framework defines what it actually means to *successfully learn* a concept.

- It does not require the ability to *learn any concept* that can be defined over X:
  - We are interested in certain subsets of all concepts $C \subseteq 2^X$. (Some concepts cannot be learned, e.g. when C is infinite and H is finite.)
  - Similarly, algorithm A will search for an appropriate hypothesis in certain hypotheses class H only.
  - $C = H$ may, but does not have to be fulfilled.
- It does not require zero error of the learned hypothesis h.
  - The hypothesis error rate is bounded with a small constant ϵ.
- It does not require the algorithm to produce the hypothesis with an acceptable error rate each time.
  - The probability of this event is however bounded by a small constant δ.

A concept class C is **PAC-learnable** using the hypotheses class H if

- for all concepts $c \in C$, all distributions $P(X)$, $X = \{0,1\}^n$, and for any $0 < \epsilon, \delta < 1$
- there is a polynomial algorithm A, which returns a hypothesis with $\mathrm{Err}(h) \leq \epsilon$ with probability at least $1 - \delta$
- using at most polynomial amount of training examples $(x_i, c(x_i))$ sampled from $P(X)$.
- "Polynomial": growing at most at polynomial rate with $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and n.

Consistent PAC learning

A consistent learning algorithm

- returns a hypothesis $h \in H$ consistent with D
- for any i.i.d. sample D (training data) of the concept $c \in C$.

Sample complexity:

- The size m of the training set D required to PAC-learn the concept c using H.
- It grows with the problem dimensionality (with the number n of object attributes).
- It represents a bound for the training set size for consistent learning algorithms.

How many training examples do we need to be able to say that *with a sufficiently high probability all consistent hypotheses are approximately correct?*

- Let's denote the set of bad hypotheses $H_B = \{h \in H : \text{Err}(h) > \epsilon\}$, $h_B \in H_B$.
- $\Pr(h_B \text{ is consistent with 1 training example}) \leq 1 - \epsilon$
- $\Pr(h_B \text{ is consistent with all training examples}) \leq (1-\epsilon)^m$ (Examples are independent.)
- $\Pr(H_B \text{ contains a hypothesis consistent with all training examples}) \leq |H_B|(1-\epsilon)^m \leq |H|(1-\epsilon)^m$
  Probability that a consistent hypothesis is not approximately correct.
- Let's bound the probability of this event with a small constant δ: $|H|(1-\epsilon)^m \leq \delta$.
- Using $1 - \epsilon \leq e^{-\epsilon}$:

$$m \geq \frac{1}{\epsilon}(\ln\frac{1}{\delta} + \ln|H|)$$

If h is consistent with m examples, then $\text{Err}(h) \leq \epsilon$ with the probability at least $1-\delta$.

Sample complexity

Sample complexity:

$$m \geq \frac{1}{\epsilon}(\ln\frac{1}{\delta} + \ln|H|)$$

Let H be the class of all boolean functions over n attributes:

- $|H| = 2^{2^n}$
- Sample complexity m grows like $\ln|H|$, i.e. like 2^n.
- But the maximal training set size grows like 2^n as well.
- PAC-learning in the class of all boolean functions requires training on all (or almost all) possible training examples!
- Reason:
  - H contains enough hypotheses to classify any set of examples in any way.
  - For any training set of m examples, the number of hypotheses consistent with the training data which classify example $\boldsymbol{x}_{m+1}$ as positive is the same as the number of consistent hypotheses which classify this example as negative.
  - See NFL: *to allow for any generalization, we need to constrain the hypotheses space H.*

Observation: m is a function of $|H|$:

- If we get an additional information (constraint limiting the class of admissible hypotheses) and embed it in the training algorithm (introduce bias), then a lower number of training examples shall be sufficient!
- **Domain knowledge** plays an important role.

Example: Decision list

Decision list (DL)

- is a sequence of tests (each test is a conjunction of literals).
- If a test succeeds, DL returns the class assigned to that test. Otherwise it continues with another test.
- *Unconstrained* DL can represent any boolean function!

Let's constrain the hypotheses space H to the language k-DL:

- Set of decision lists where each test is a conjunction of at most k literals.
- The k-DL language contains the language k-DT (set of decision trees with the depth at most k) as its subset.
- The particular instance of the k-DL language depends on the set of attributes (the representation used).
- Let k-DL(n) be the k-DL language over n Boolean attributes.

How can we show that the hypotheses class k-DL is PAC-learnable?

1. Show that sample complexity is at most polynomial (see next slide).
2. Show that there is a learning algorithm with at most polynomial computational complexity. (Not presented, but e.g. CN2 algorithm will do.)

Example: Decision list (cont.)

Let's show that any hypothesis from k-DL can be accurately approximated by learning from a training set of reasonable a size:

- We need to estimate the number of hypotheses in the language.
- Let $Conj(n, k)$ be the set of tests (conjunctions of at most k literals over n attributes).
- Each test is assigned with an output value "Yes", "No", or it does not have to be present in the list at all, thus there are at most $3^{|Conj(n,k)|}$ different sets of tests.
- Each of these sets of test may be used in an arbitrary order, thus $|k{-}\mathrm{DL}(n)| \leq 3^{|Conj(n,k)|} \cdot |Conj(n,k)|!$.
- The number of at most k literals with n attributes: $|Conj(n,k)| = \sum_{i=0}^{k} \binom{2n}{i} = \mathcal{O}(n^k)$.
  $2n$, since the conjuction can contain each individual attribute test directly or in negation.
- After simplification: $k{-}\mathrm{DL}(n) = 2^{\mathcal{O}(n^k \log_2(n^k))}$
- Substituting this result for $|H|$ to the sample complexity equation: $m \geq \frac{1}{\epsilon}\left(\ln \frac{1}{\delta} + \mathcal{O}\left(n^k \log_2(n^k)\right)\right)$
- m grows polynomially with n
- Any algorithm that returns a k-DL consistent with training data will PAC-learn a k-DL concept with a reasonable training set size.

Example: DNF Formulas

Disjunctive normal form (DNF):

- Objects described with n Boolean attributes $a_1, \ldots, a_n$.
- DNF formula: a disjunction of conjunctions, e.g. $(a_1 \wedge \neg a_2 \wedge a_5) \vee (\neg a_3 \wedge a_4)$

What is the size of the hypotheses space H:

- 3^n possible conjunctions.
- $|H| = 2^{3^n}$ possible disjunctions of conjunctions.
- $\ln |H| = 3^n \ln 2$ is not polynomial in n.
- We have not succeeded in showing that DNF formulas are PAC-learnable. (But we neither showed the opposite.)

PAC-learning of DNF formulas is still an open problem.

Examples of results for PAC learning

1. Conjunctive concepts are PAC-learnable, but
   concepts in the form of a disjunction of 2 conjunctions are not PAC-learnable.
2. Linearly separable concepts (perceptrons) are PAC-learnable in both Boolean and real spaces. But
   a conjunction of 2 perceptrons is not PAC-learnable, similarly to a disjunction of 2 perceptrons and multi-layer perceptrons with 2 hidden units. If we additionally constrain the weights to values 0 and 1, then even perceptrons in Boolean space are not PAC-learnable.
3. The classes k-CNF, k-DNF and k-DL are PAC-learnable for a given k. But
   we do not know if DNF formulas, CNF formulas, or decision trees are PAC-learnable.

VC dimension

Disadvantages of using $|H|$ in the sample complexity formula:

- Results in a worst-case estimate.
- It is often very pessimistic, it overesimates the number of required training examples.
- $|H|$ cannot be used for infinite hypotheses spaces.

Capacity, Vapnik-Chervonenkis dimension $VC(H)$

- Another measure of the complexity (flexibility) of the hypotheses class H: it quantifies the bias of embodied in ceartain hypotheses class H.
- Applicable even for infnite H.
- Can provide a tighter bound for the sample complexity.
- **Definition:** $VC(H)$ is the maximal number d of examples $x \in X$ such that for each of 2^d different labelings of $x_1, \ldots, x_d$ there is a hypothesis $h \in H$ consistent with these d examples.

Sample complexity using VC dimension:

- Hypotheses space H, concepts space C, $C \subseteq H$.
- Sample complexity for any consistent algorithm learning $c \in C$ using H is

$$m \geq \frac{1}{\epsilon}\left(4\log_2\frac{2}{\delta} + 8 \cdot VC(H) \cdot \log_2\frac{13}{\epsilon}\right)$$

VC dimension (cont.)

VC dimensions for certain hypotheses classes H:

- VC dimension of a linear discriminant function in 1D space? 2.
  Lin. discr. function is not able to correctly represent all possible concepts examplified by 3 or more points in 1D space.
- VC dimension of a linear discriminant function in 2D space? 3.
  Lin. discr. function is not able to correctly represent all possible concepts examplified by 4 or more points in 2D space.
- Generally, for linear discriminant function $f_n(\boldsymbol{x}) = w_0 + w_1x_1 + \ldots + w_nx_n$ in n-dimensional space: $VC(f_n) = n + 1$
- Example of 1D function f with $VC(f) = \infty$: $f(x) = \sin(\alpha x)$
  It can be shown that $\sin(\alpha x)$ can in 1D space correctly classify any number of examples.
- VC dimension of SVM with RBF kernel without any constraint on the penalty term: $VC(f_{SVM-RBF}) = \infty$

Other uses of VC dimension:

- Estimation of a true (testing) error of a classifier on the basis of the training data only.
- "Structural risk minimization", the basic principle of SVM.

Summary

- **Generalization requires bias!!!**
- NFL: All models/algorithms are equally good on average.
  - If a certain class of models works better for certain class of problems, there must be another class of problems, for which it workse worse.
  - Our goal is to find models/algorithms which
    - work well for problem classes often observed in practice, and
    - have below average performace on problem classes which are not practically important.
- Probably Approximately Correct (PAC) learning:
  - specifies what it means to "learn correctly".
  - introduces tolerances for the model error (ϵ) and for the probability (δ) that a learned model has a larger error than ϵ.
  - allows to estimate the required training set size.
- VC dimension:
  - a measure of flexibility of (even infinite) hypotheses class.
  - usually provides tighter estimates of the sample complexity than the formula with $|H|$.