Aalto University
School of Science

CS-E4070 — Computational learning theory

Slide set 02 : Occam's razor

Cigdem Aslay and Aris Gionis

Aalto University

spring 2019

reading material

- K&V, chapter 2
- Blumer et al., “Occam’s razor”, IPL, 1987

Occam's razor

- William of Ockham (1287 – 1347)

  "entities are not to be multiplied without necessity"

- has been used as guiding principle in developing simple models

- in machine learning, simpler models are considered to:
  - capture better the underlying structure
  - be less sensitive to noise
  - have better predictive power

Occam's razor

- the parsimony principle has been applied to motivate different computational approaches in machine learning
  - minimum description length (MDL)
  - Bayesian information criterion (BIC)
  - ℓ_1 regularization
  - model pruning, etc.
- the principle is intuitive, has philosophical basis, . . . and works well in practice
- but we can rigorously show that parsimony leads to models with good predictive power?

Occam's razor

- we now consider Occam algorithms

  such algorithms focus only on parsimony

  they produce a hypothesis that compresses the data

  no attempt to make accurate predictions

- yet, we will show that in the PAC learning setting

  Occam algorithms have predictive power

- thus, in our setting

  compression $\Rightarrow$ learning

Occam algorithm

- consider :

  concept class $\mathcal{C}_n$, target concept $c \in \mathcal{C}_n$

  hypothesis representation class $\mathcal{H}_n$,

  sample of cardinality m :

$$S = \{\langle \mathbf{x}_1, c(\mathbf{x}_1)\rangle, \ldots, \langle \mathbf{x}_m, c(\mathbf{x}_m)\rangle\}$$

- an Occam algorithm A takes as input S and produces a succinct hypothesis $h \in \mathcal{H}_n$ that compresses S, i.e.,

$$h(\mathbf{x}_i) = c(\mathbf{x}_i) \text{ for all } i = 1, \ldots, m$$

  or alternatively, h is consistent with S

- succinct means that $size(h)$ is growing asymptotically slower than m and n

Occam algorithm — formalization

- consider constants $\alpha \geq 0$ and $0 \leq \beta < 1$
- an algorithm A is (α, β)-Occam algorithm for $\mathcal{C}$ using $\mathcal{H}$ if on input S of cardinality m, the algorithm produces a hypothesis $h \in \mathcal{H}$ such as
  - h is consistent with S
  - $size(h) \leq n^{\alpha} m^{\beta}$
- furthermore, A is an efficient (α, β)-Occam algorithm if its running time is polynomial in m and n

Occam algorithm

- in which sense is the hypothesis h succinct?
- assuming $m >> n$, then $size(h) \leq m^\beta$
- since we require $\beta < 1$, this is asymptotically less than m
- storing the sample S can be done in space $\mathcal{O}(nm)$
- thus, h can be seen as a compression of S

Occam's razor — main result

efficient Occam algorithm $\Rightarrow$ efficient PAC learning

- **theorem:** let A be an efficient (α, β)-Occam algorithm for $\mathcal{C}$ using $\mathcal{H}$. Consider any $c \in \mathcal{C}$, any $\epsilon > 0$, $\delta \in (0, 1)$, and any distribution $\mathcal{D}$. Then, there exists a constant c so that if A receives as input a sample of size m, drawn from $EX(\mathcal{D}, c)$, and m satisfies

$$m \geq c \left(\frac{1}{\epsilon} \log \frac{1}{\delta} + \left(\frac{n^\alpha}{\epsilon} \right)^{\frac{1}{1-\beta}} \right)$$

then A returns a hypothesis $h \in \mathcal{C}$ that satisfies $error_{\mathcal{D}}(h) \leq \epsilon$ with probability at least $1 - \delta$.

moreover, A is polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$

Occam's razor — main result — proof sketch

recall our previous result:

- a finite hypothesis class is PAC learnable

recall the proof:

- consider h with $error > \epsilon$ that we worry that it may fool us
- probability that h is consistent with S is at most $(1-\epsilon)^m$
- probability that any such bad hypothesis is consistent with S is at most $|\mathcal{H}|(1-\epsilon)^m$
- requiring $|\mathcal{H}|(1-\epsilon)^m \leq \delta$ gives $m \geq \log(|\mathcal{H}|/\delta)/\epsilon$
- so $\mathbf{Pr}\left[error(h) > \epsilon\right] \leq \delta$, or $\mathbf{Pr}\left[error(h) \leq \epsilon\right] \geq 1-\delta$

number of samples should be as large as $\log|\mathcal{H}|$, but not $|\mathcal{H}|$

Occam's razor — main result — proof sketch

showing that Occam property and number of samples satisfying

$$m \geq c\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{n^{\alpha}}{\epsilon}\right)^{\frac{1}{1-\beta}}\right)$$

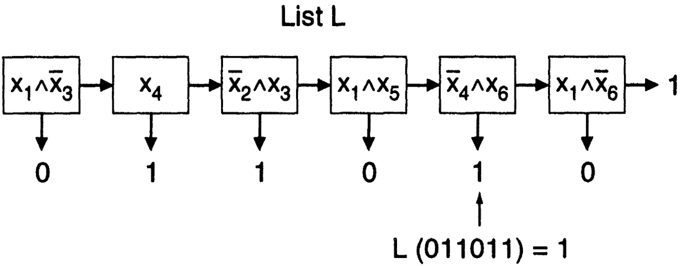
imply PAC learning

- since A is an Occam algorithm, we have $size(h) \leq n^{\alpha} m^{\beta}$
- $size(h)$ is number of bits to represent h, thus, $|\mathcal{H}| \leq 2^{n^{\alpha} m^{\beta}}$
- applying the second bound on m we get $2^{n^{\alpha} m^{\beta}} \leq (1-\epsilon)^{-m/2}$
- applying the previous lemma we get that probability of $error > \epsilon$ is at most $|\mathcal{H}|(1-\epsilon)^{m} \leq (1-\epsilon)^{-m/2}(1-\epsilon)^{m} = (1-\epsilon)^{m/2}$
- applying the first bound on m we get that this probability is less than δ

learning decision lists

- a decision list is defined over a set of boolean variables $x_1, \ldots, x_n$
- can be viewed as an sequence of if-then-else statements
- in a k-decision list each term is a conjunction of at most k literals

example of 2-decision list:



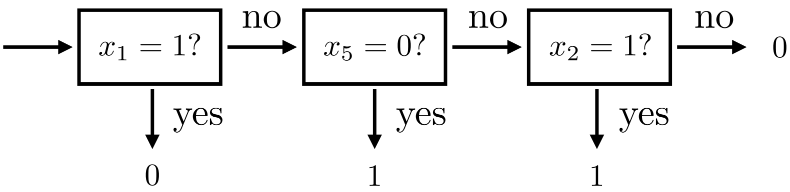
expressive power of decision lists

- a k-DNF formula can be expressed as k-decision list
- since k-decision lists are closed under complement, they can also express k-CNF formulas
- however, they are strictly more expressive : there are formulas that can be represented by a k-decision list but neither by a k-DNF nor by a k-CNF

learning decision lists

- **theorem:** for any fixed $k \geq 1$, the representation class of k-decision lists is efficiently PAC learnable

learning decision lists



- we will discuss the case of 1-decision list
  - each term contains a single literal
- the general case, $k > 1$, can be handled similarly to learning using k-CNF formulas

learning decision lists

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	1	0	1	1
1	0	1	1	0	0
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	1	1	0

learning decision lists

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	1	0	1	1
1	0	1	1	0	0
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	1	1	0

$$\text{if } (x_2 = 1) \text{ then } 1$$

learning decision lists

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	1	0	1	1
1	0	1	1	0	0
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	1	1	0

if $(x_2 = 1)$ then 1

if $(x_5 = 1)$ then 0

learning decision lists

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	1	0	1	1
1	0	1	1	0	0
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	1	1	0

if $(x_2 = 1)$ then 1

if $(x_5 = 1)$ then 0

if $(x_1 = 1)$ then 0

learning decision lists

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	1	0	1	1
1	0	1	1	0	0
1	0	1	0	0	0
0	1	0	1	0	1
0	0	0	1	1	0

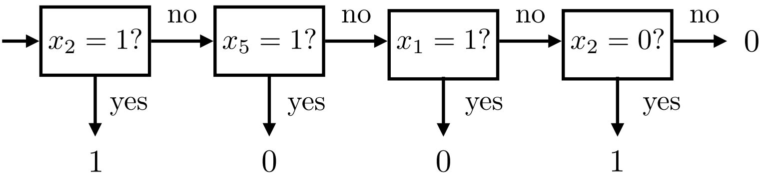
if $(x_2 = 1)$ then 1

if $(x_5 = 1)$ then 0

if $(x_1 = 1)$ then 0

if $(x_2 = 0)$ then 1

learning decision lists



if $(x_2 = 1)$ then 1

if $(x_5 = 1)$ then 0

if $(x_1 = 1)$ then 0

if $(x_2 = 0)$ then 1

learning decision lists — algorithm

- S is the set of examples
- start with an empty list
- find a rule consistent with data
  - find a literal z, which is set to 1 in a subset of examples S_z, so that S_z is not empty and S_z consists of only positive or only negative examples
- add the rule $z = 1$ to the end of decision list
- remove S_z from S
- repeat until the no examples remain

consistency of the decision-list algorithm

- the decision-list algorithm succeeds in finding a hypothesis consistent with the data, if such a hypothesis exists
- if the algorithm fails, then there is no decision list that is consistent with the data

efficient PAC learning of decision lists

- the algorithm we described is an Occam algorithm (!)
- for any decision list h returned by the algorithm

$$size(h) = \mathcal{O}(n \log n)$$

- notice that, $size(h)$ does not depend on m, i.e., $\beta = 0$
- thus, we can achieve PAC learning with

$$m \geq c \left(\frac{1}{\epsilon} \log \frac{1}{\delta} + \frac{n \log n}{\epsilon} \right)$$

- moreover, the algorithm runs in polynomial time

what about decision trees?

- can we obtain efficient PAC learning for decision trees ?
- we can find a decision tree consistent with the data
  - how ?
- can we apply a similar technique as for decision lists ?
  - where does it break down ?
  - number of leaves is proportional to m, thus, we cannot find an Occam algorithm with $\beta < 1$
  - (finite hypothesis class, thus, PAC learnable, but not efficiently PAC learnable)
- we would like to find the smallest decision tree consistent with the data
  - however, this is an **NP**-hard problem

discussion : drawbacks of PAC learning

- running time comparable to number of examples
  - in real applications labeled data is much more expensive than running time
- we assumed that we know the class of the target concept
  - in the real world we do not know if data come from a tree model, a decision list, or a 4-CNF
- realizability assumption too strong
  - model does not allow for errors
- does not account for other kinds of data
  - unlabeled data, pairwise similarities
- addresses only batch learning
  - no online setting