Consistency + Polynomial $\ln |\mathcal{H}|$ Imply PAC-Learning

An algorithm using hypothesis class $\mathcal{H}$ is *$\mathcal{C}$-consistent* if, given an arbitrary example set from an arbitrary concept $C \in \mathcal{C}$, it returns a $h \in \mathcal{H}$ consistent with the example set.

($\mathcal{H} \supseteq \mathcal{C}$ is a necessary condition for $\mathcal{C}$-consistency.)

A $\mathcal{C}$-consistent algorithm using $\mathcal{H}$ PAC-learns $\mathcal{C}$ if $\ln |\mathcal{H}| \leq \text{poly}(n)$. Why?

Prob. that a given *bad* h ($\text{err}(h) > \epsilon$) survives (i.e., is consistent with) a random example is at most $(1 - \epsilon)$.

Consistency + Polynomial $\ln|\mathcal{H}|$ Imply PAC-Learning

Prob. that h survives m i.i.d. examples is at most $(1-\epsilon)^m$.

Prob. that one of the bad hypotheses $h \in \mathcal{H}$ survives is at most $|\mathcal{H}|(1-\epsilon)^m \leq |\mathcal{H}|e^{-\epsilon m}$.

To make this smaller than δ, it suffices to set the number of examples to

$$m = \frac{1}{\epsilon} \ln \frac{|\mathcal{H}|}{\delta}$$

which is $\leq \text{poly}(1/\epsilon, 1/\delta, n)$ iff $\ln|\mathcal{H}| \leq \text{poly}(n)$.

Compare this to the similar result in the mistake-bound model (Halving algorithm).

Consistency + Polynomial VC($\mathcal{H}$) Imply PAC-Learning

Using VC($\mathcal{H}$), a bound can be established even for $|\mathcal{H}| = \infty$:

With probability at least δ, no bad hypothesis $h \in \mathcal{H}$ survives m i.i.d. examples where

$$m \geq \frac{8}{\epsilon}\left(\text{VC}(\mathcal{H}) \ln \frac{16}{\epsilon} + \ln \frac{2}{\delta}\right)$$

(We omit the proof.)

Thus a $\mathcal{C}$-consistent algorithm using $\mathcal{H}$ PAC-learns $\mathcal{C}$ if VC($\mathcal{H}$) $\leq$ poly(n).

For example, let $\mathcal{C}$ = half-planes in R^n. $|\mathcal{H}| = \infty$ but VC($\mathcal{H}$) $= n + 1 \leq$ poly(n).

PAC-Learning k-term DNF

We know that $\mathcal{C} = k$-term DNF is learnable efficiently using $\mathcal{H} = k$-CNF in the MB model and thus also in PAC.

But what if $\mathcal{H} = \mathcal{C}$ (i.e., proper learning)?

$\ln |\mathcal{C}| = \ln |k\text{-term DNF}| \leq \ln |k\text{-CNF}| \leq \text{poly}(n)$ so $\mathcal{C}$ is PAC-learnable even with $\mathcal{H} = \mathcal{C}$.

BUT: this cannot be done *efficiently*. We show this for $k = 3$.

Finding a $h \in \mathcal{H} = k$-term DNF consistent with the training examples is as hard as the *graph 3-coloring problem*:

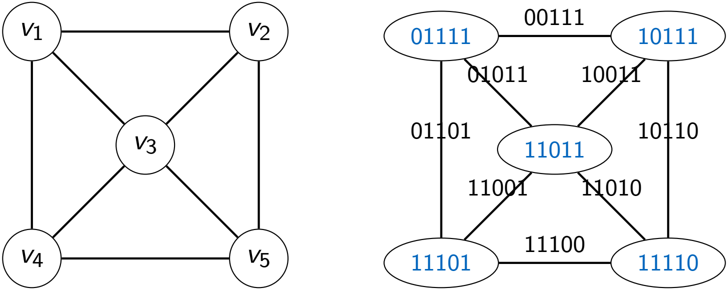
- Give each vertex one of 3 colors, adjacent vertices - different colors

3-Coloring as Finding a Consistent 3-term DNF

Efficient reduction:

$$\text{vertex } v_i \quad \sim \quad \text{pos. example } x,\ x[k] = \begin{cases} 0 \text{ if } k = i \\ 1 \text{ otherwise} \end{cases}$$

$$\text{edge } e_{ij} \quad \sim \quad \text{neg. example } x,\ x[k] = \begin{cases} 0 \text{ if } k = i \text{ or } k = j \\ 1 \text{ otherwise} \end{cases}$$



3-Coloring as Finding a Consistent 3-term DNF

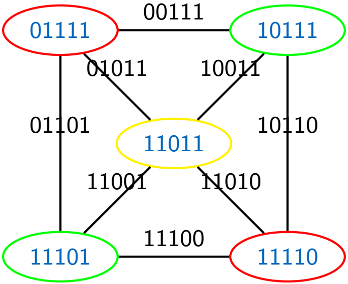
Graph 3-colorable iff a 3-term DNF exists consistent with the examples:

- Given a 3-colored graph, a consistent 3-term DNF can be constructed:

$$\bigvee_{\substack{\textbf{color} \in \\ \{R, G, Y\}}} \; \bigwedge_{\substack{v_k \text{ not of} \\ \textbf{color}}} v_k$$

- Given a consistent 3-term DNF, the graph can be validly colored
    - Give each vertex the color corresponding to any term consistent with the vertex variable

Example:



$\leftrightarrow$

$v_2 \wedge v_3 \wedge v_4$
$\bigvee$
$v_1 \wedge v_3 \wedge v_5$
$\bigvee$
$v_1 \wedge v_2 \wedge v_4 \wedge v_5$

3-colorability NP-hard $\rightarrow$ finding a consistent 3-term DNF NP-hard.

Generally, $\mathcal{C} = k$-term DNF cannot be PAC-learned efficiently AND properly ($\mathcal{H} = \mathcal{C}$).

k-Decision Trees

(Binary) decision tree: a binary tree-graph

- non-leaf vertices: binary variables
- leafs: class indicators

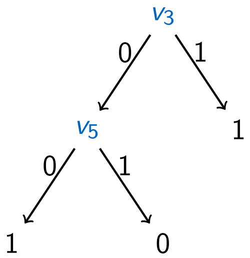
Classification: go from root to leaf, path according to truth-values of variables.

k-DT = dec. trees of max depth k

Like k-term DNF,

- finding a consistent k-DT is NP-hard (proof omitted).
- *k-DT thus cannot be PAC-learned efficiently + properly.*

Example:



3-Decision Tree

PAC-Learning k-Decision Trees Efficiently

Every k-DT has an equivalent k-DNF:

- For every path going from root to a 1 leaf, add to the DNF a k-conjunction of all variables on the path ($v_3 \vee \overline{v_3}\ \overline{v_5}$ for the example)

Thus

$$k\text{-DT} \subseteq k\text{-DNF}$$

and $\mathcal{C} = k$-DT can be *efficiently (but not properly) PAC-learned* using $\mathcal{H} = k$-DNF.

Note that also

$$k\text{-DT} \subseteq k\text{-CNF}$$

- Create a clause for each path to a 0 leaf ($v_3 \vee \overline{v_5}$ for the example)

PAC-Learning k-Decision Trees Properly

We will show that $\lg |k\text{-DT}| \leq \text{poly}(n)$:

- $|1\text{-DT}| = 2$ (two options for the single vertex = leaf)
- $|(k+1)\text{-DT}| = n|k\text{-DT}|^2$ (n options for vertex, $|k\text{-DT}|$ options for each of the 2 subtrees)

Denote $c_n = \lg |k\text{-DT}|$. We have $c_1 = 1$ and

$$c_{k+1} = \lg n + 2c_k$$

i.e., a recursive formula for a geometric series. Solution exponential in k but polynomial in n.

So $\mathcal{C} = k$-DT can be *properly (but not efficiently) PAC-learned* by a $\mathcal{C}$-consistent algorithm.

k-Decision Lists

k-Decision List: a list of k-conjunctions (each with a class indicator) + default class indicator.

Example:

$$\begin{array}{rcl} v_1\overline{v_3} & \rightarrow & 0 \\ v_2 & \rightarrow & 1 \\ \text{default} & & 0 \end{array}$$

An example is classified to the class indicated at the first from top conjunction satisfied by the example, or the default if none satisfied.

We will show an efficient consistent learning algorithm for k-DL.

Finding a Consistent k-Decision List

Require: training set $T = \{ (x_1, y_1), (x_2, y_2) \dots (x_m, y_m) \}$ ▷ (the $y_i \in \{ 0, 1 \}$ are class labels)

```
L := [] (empty list)
while T ≠ ∅ do
    γ = any k-conjunction true for some pos. and no neg. example in T, or some neg. and no pos. example in T (respectively)
    Remove examples covered by γ: T := T \ {(x, y) ∈ T : x ⊨ γ}
    if T = ∅ then
        append default 1 or default 0 (respectively) to L.
    else
        append γ → 1 or γ → 0 (respectively) to L
    end if
end while
```

Finding a Consistent k-Decision List

In Step 3, the algorithm always succeeds in finding the required k-conjunction γ.

Indeed, such a γ exists:

- Let c be the DL encoding the target concept;
- Let $\gamma^* \rightarrow$ *class* be the top-most rule in c which 'fires' ($x \models \gamma^*$) for least one $x \in T$;
- $\gamma^* \rightarrow$ *class* must be consistent with T; if inconsistent with any $x' \in T$, a rule higher in c would have to fire for x' but that contradicts the 'top-most' assumption above;
- so $\gamma = \gamma^*$ is one possible choice.

In the worst case, the algo needs to search all of the $\leq$ poly(n) number of k-conjunctions.

PAC-Learning k-DL Properly Efficiently

The k-DL-consistent algorithm PAC-learns k-DL if $\ln |k\text{-DL}| \leq \text{poly}(n)$.

$$|k\text{-DL}| = 3^{|k\text{-conjunctions}|}!$$

- base 3: each k-conjunction either absent, present with class 0 or present with class 1
- factorial: different order of k-conjunctions - different k-DL's

Since $|k\text{-conjunctions}| \leq \text{poly}(n)$, we indeed have

$$\ln |k\text{-DL}| \leq \text{poly}(n)$$

k-DL Subsumes k-DNF and k-CNF

For any k-DNF, an equivalent k-DL can be made:

- for each k-conjunction c in the k-DNF, add c to the DL with class 1
- add to the DL the default rule with class 0

So

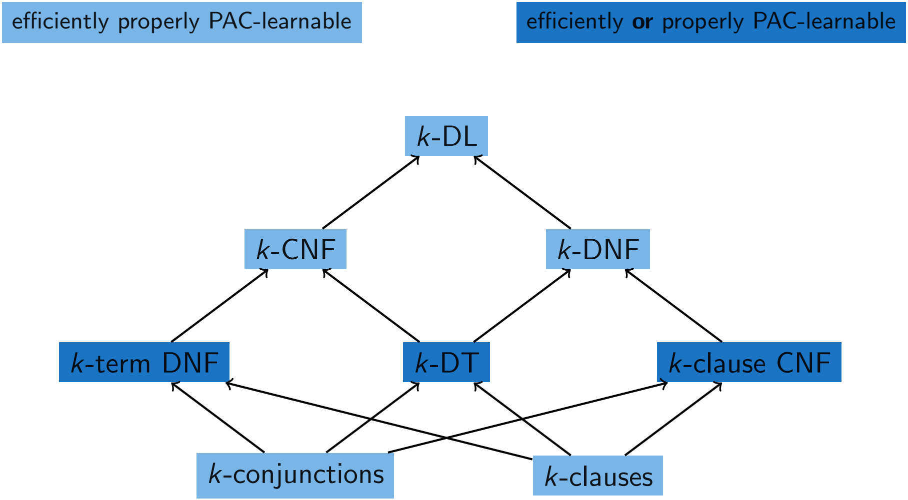
$$k\text{-DNF} \subseteq k\text{-DL}$$

k-DL is closed under negation (just flip the class indicators) and each k-CNF is the negation of some k-DNF. Therefore

$$k\text{-CNF} \subseteq k\text{-DL}$$

(The inclusions are actually strict because k-DNF $\neq$ k-CNF.)

Subset Hierarchy of Some Concept Classes



Inconsistent Learning

Returning a hypothesis consistent with the training set may not be possible for reasons such as

- $\mathcal{H} \not\supseteq \mathcal{C}$;
- $\mathcal{C}$ is not known ('agnostic learning') so $\mathcal{H} \not\supseteq \mathcal{C}$ cannot be excluded;
- There is 'noise' in data so the training set may include the same instance as both a positive and a negative example.

Define the *training error* $\widehat{\text{err}}(h)$ as the proportion of training examples inconsistent with h. $\widehat{\text{err}}(h)$ is also called the *empirical risk*.

We are interested in the relationship btw. $err(h)$ and $\widehat{\text{err}}(h)$.

Hoeffding Inequality

Hoeffding: Let $\{ z_1, z_2, \ldots, z_m \}$ be a set of i.i.d. samples from $P(z)$ on $\{ 0, 1 \}$. The probability that $\left| P(1) - \frac{1}{m} \sum_{i=1}^{m} z_i \right| > \epsilon$ is at most $2e^{-2\epsilon^2 m}$.

Let $z_i = 1$ iff i.i.d. example x_i is misclassified by h. So

$$P(1) = \mathrm{err}(h)$$
$$\frac{1}{m} \sum_{i=1}^{m} z_i = \widehat{\mathrm{err}}(h)$$

Thus for a given h, $|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| > \epsilon$ with prob. at most $2e^{-2\epsilon^2 m}$.

Error Bound for Inconsistent Learning

For a finite $\mathcal{H}$, the prob. that $|\text{err}(h) - \widehat{\text{err}}(h)| > \epsilon$ for *some* $h \in \mathcal{H}$ is at most

$$|\mathcal{H}| 2e^{-2\epsilon^2 m}$$

We want to make this no greater than δ. Solving $\delta = |\mathcal{H}| 2e^{-2\epsilon^2 m}$ gives

$$\epsilon = \sqrt{\frac{1}{m} \ln \frac{2|\mathcal{H}|}{\delta}}$$

So with prob. at least $1 - \delta$, the difference btw. $\text{err}(h)$ and $\widehat{\text{err}}(h)$ is at most as above for all $h \in \mathcal{H}$.

Dilemma: A large $\mathcal{H}$ allows to achieve a small $\widehat{\text{err}}(h)$ but means a loose bound on $\text{err}(h)$.

Sample Complexity for Inconsistent Learning

Solving $\delta = |\mathcal{H}| 2 e^{-2\epsilon^2 m}$ instead for m gives

$$m = \frac{1}{2\epsilon^2} \ln \frac{2|\mathcal{H}|}{\delta}$$

which is thus a number of examples sufficient to make $|\text{err}(h) - \widehat{\text{err}}(h)| \leq \epsilon$ with prob. at least $1 - \delta$ for all $h \in \mathcal{H}$.

$m \leq \text{poly}(1/\epsilon, 1/\delta, n)$ iff $\ln |\mathcal{H}| \leq \text{poly}(n)$

Error Bound for ERM

Assume the learner returns

$$h = \arg\min_{h \in \mathcal{H}} \widehat{\text{err}}(h)$$

This is called *empirical risk minimization* (ERM principle).

Let $h^* = \arg\min_{h \in \mathcal{H}} \text{err}(h)$, i.e. h^* is the best hypothesis.

Let further $m = \frac{1}{2\epsilon^2} \ln \frac{2|\mathcal{H}|}{\delta}$. Then with prob. at least $1 - \delta$:

$$\begin{aligned}
\forall h \in \mathcal{H} : \text{err}(h) &\leq \widehat{\text{err}}(h) + \epsilon && \text{which we just proved} \\
&\leq \widehat{\text{err}}(h^*) + \epsilon && \text{because } h \text{ minimizes } \widehat{\text{err}} \\
&\leq \text{err}(h^*) + 2\epsilon && \text{because } \widehat{\text{err}}(h^*) \leq \text{err}(h^*) + \epsilon
\end{aligned}$$

Bias-Variance Trade-Off

Put differently, with prob. at least $1 - \delta$:

$$\mathrm{err}(h) \leq \min_{h \in \mathcal{H}} \mathrm{err}(h) + 2\sqrt{\frac{1}{2m} \ln \frac{2|\mathcal{H}|}{\delta}}$$

Large $\mathcal{H}$ - large variance - small bias - first summand lower, second larger

Too large $\mathcal{H}$: overfitting, too small $\mathcal{H}$: underfitting

The more training data (m), the larger $\mathcal{H}$ can be 'afforded'.