The Halving Algorithm (Version Space)

Maintains a *finite set of hypotheses* $\mathcal{H}$ ("version space") and on each example x, deletes from it all hypotheses that misclassify it.

$$\mathcal{H}' = \{ h \in \mathcal{H} : h(x) = c(x) \}$$

Decides by *majority vote* among the current $\mathcal{H}$, i.e., "yes" iff

$$|\{ h \in \mathcal{H} : h(x) = 1 \}| > |\{ h \in \mathcal{H} : h(x) = 0 \}|$$

On each mistake, at least half of the hypotheses were wrong so at least *half* of them get deleted. This gives the *mistake bound*

$$\lg |\mathcal{H}|$$

where $\mathcal{H}$ is the initial version space, i.e., the learner's hypothesis class.

The Halving Algorithm (Version Space)

Any finite class $\mathcal{C}$ of computable concepts is learnable if $\lg |\mathcal{C}| \leq \text{poly}(n)$.

Proof: Use the halving algorithm with any $\mathcal{H} \supseteq \mathcal{C}$ such that $\lg |\mathcal{H}| \leq \text{poly}(n)$. [1]

That does not mean $\mathcal{C}$ is learnable *efficiently*!

If $|\mathcal{C}|$ is exponentially large, then the halving algo is necessarily non-efficient.

[1] We overload the symbol $\mathcal{H}$ to mean both a class of hypotheses (e.g. conjunctions) and the concept class they define (subsets of X).

Sizes of Some Concept Classes

- Conjunctions or disjunctions: $|\mathcal{C}| = 2^{2n}$ resp. 3^n if contradictions/tautologies excluded.
  - Both halving and generalization algos have linear mistake bound, but the latter is efficient
- k-disjunctions: $|\mathcal{C}| = \sum_{i=1}^{k} \binom{2n}{i}$ resp. $\sum_{i=1}^{k} \binom{n}{i} 2^i \leq \text{poly}(n)$
  - Both halving and WINNOW: logarithmic mistake bound, efficient
  - k-conjunctions: same, except WINNOW won't apply
- k-DNF, k-CNF: $|\mathcal{C}| = 2^{|k\text{-disjunctions}|} \leq 2^{poly(n)}$
  - Halving: poly mistake bound, non-efficient
  - Reduction to monotone conjuctions (disjuctions): poly m.b., efficient

VC Dimension

We say that concept class $\mathcal{C}$ *shatters* a set of instances $X' \subseteq X$ if for every subset $X'' \subseteq X'$ there is a concept $C \in \mathcal{C}$ such that $C \cap X' = X''$.

In other words, X' is shattered by $\mathcal{C}$ if it can be split by concepts from $\mathcal{C}$ in all $2^{|X'|}$ possible ways.

The *VC-dimension* of $\mathcal{C}$ denoted $\mathrm{VC}(\mathcal{C})$ is the size of the largest subset of X shattered by $\mathcal{C}$:

$$\mathrm{VC}(\mathcal{C}) = \max \{ \, |X'| \; : \; \mathcal{C} \text{ shatters } X', \; X' \subseteq X \, \}$$

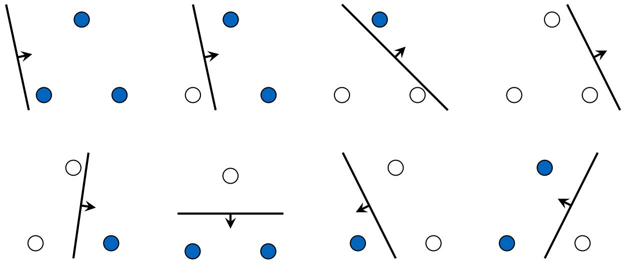
$\mathrm{VC}(\mathcal{H})$ for a *hypothesis* class $\mathcal{H}$ defined analogically.

Determining VC-Dimension: Example

- If *some* $X' \subseteq X$ shattered by $\mathcal{C}$ then $\mathrm{VC}(\mathcal{C}) \geq |X'|$.
- If *none* $X' \subseteq X$ shattered by $\mathcal{C}$ then $\mathrm{VC}(\mathcal{C}) < |X'|$.

Example: $\mathcal{C}$ = half-planes in R^2 (i.e., linear separation)

- *Some* 3 points can be shattered so $\mathrm{VC}(\mathcal{C}) \geq 3$.



Determining VC-Dimension: Example

- *No* 4 points can be shattered. Obvious if 3 in line. Otherwise two cases possible:

One point in the middle   No point in the middle

In both cases, the colored subset cannot be separated by a line. So $\mathrm{VC}(\mathcal{C}) < 4$

We have $\mathrm{VC}(\mathcal{C}) \geq 3$ and $\mathrm{VC}(\mathcal{C}) < 4$, thus $\mathrm{VC}(\mathcal{C}) = 3$.

Poly VC-Dimension Necessary for Learnability

Concept class $\mathcal{C}$ on X is learnable *only if* $\mathrm{VC}(\mathcal{C}) \leq \mathrm{poly}(n)$.

Proof: There exists a set of $\mathrm{VC}(\mathcal{C})$ instances from X shattered by $\mathcal{C}$ so there exists a sequence $x_1, x_2, \ldots x_{\mathrm{VC}(\mathcal{C})}$ of instances such that for any sequence of the learner's decisions there is a concept $c \in \mathcal{C}$ making all these decisions wrong.

So $\lg |\mathcal{C}| \leq \mathrm{poly}(n)$ implies $\mathrm{VC}(\mathcal{C}) \leq \mathrm{poly}(n)$ but not the other way around.

$\mathrm{VC}(\mathcal{C})$ may be finite (even $\mathrm{poly}(n)$) even if $|\mathcal{C}| = \infty$!

PAC Learning Model

PAC = Probably Approximately Correct

Main differences from the mistake bound model:

- A "batch" style of learning rather than "online":
  - A *training* set of examples is provided to the learner.
  - The learner outputs a hypothesis.
- Assumes an arbitrary probability distribution on X from which examples are drawn mutually independently ("i.i.d. assumption").
- No bound on the total number of mistakes, instead the output hypothesis should have a bounded *error* rate (mistake probability).
- Probability of failure (good hypothesis not found) also bounded.
- Size of the training set only polynomial in n and the inverse of the two bounds.

PAC Learning Model: Definition

Given a probability distribution P on X, a concept C and a hypothesis H, define the *error* of H: $err(H) = P(C \triangle H) = P(c(x) \neq h(x))$

Formally: $\text{err}(h) = \text{err}(H)$ (h is the description of H)

We say that an algorithm *PAC-learns concept class* $\mathcal{C}$ if for any $C \in \mathcal{C}$, an arbitrary distribution P on X, and arbitrary numbers $0 < \epsilon, \delta < 1$, the algorithm, which receives a $\text{poly}(1/\epsilon, 1/\delta, n)$ number of i.i.d. examples from $P(X)$, outputs with probability at least $1 - \delta$ a hypothesis h such that $\text{err}(h) \leq \epsilon$. If such an algorithm exists, we call $\mathcal{C}$ *PAC-Learnable*.

If an algorithm PAC-learns $\mathcal{C}$ and runs in $\text{poly}(1/\epsilon, 1/\delta, n)$ time, we say it PAC-learns $\mathcal{C}$ *efficiently* and we call $\mathcal{C}$ *efficiently PAC-learnable*.

PAC Learning Conjunctions

Use the generalization algo for PAC learning: provide m examples to it, run it as if online, keep the last h.

Let $P_{\text{ic}}(z)$ be the prob. that literal z ($z \in \{ h_1, \overline{h_1}, h_2, \ldots \overline{h_n} \}$) is inconsistent with a random example drawn from $P(X)$.

Call z *bad* if $P_{\text{ic}}(z) \geq \frac{\epsilon}{2n}$.

Observe that $\text{err}(h) \leq \sum_z P_{\text{ic}}(z)$. So if h has no bad literals then

$$\text{err}(h) \leq \sum_z \frac{\epsilon}{2n} = 2n\frac{\epsilon}{2n} = \epsilon$$

PAC Learning Conjunctions

Prob. that a bad literal z “survived” (was consistent with) one random example is

$$1 - P_{\text{ic}}(z) \leq 1 - \frac{\epsilon}{2n}$$

Prob. that z survived *m such i.i.d. examples* is thus at most

$$\left(1 - \frac{\epsilon}{2n}\right)^m$$

So prob. that *one of the $2n$* possible bad literals survived m i.i.d. examples is at most

$$2n\left(1 - \frac{\epsilon}{2n}\right)^m \leq 2ne^{-\frac{m\epsilon}{2n}}$$

because of the general inequality $1 - x \leq e^{-x}$ for $x \geq 0$.

PAC Learning Conjunctions

To satisfy PAC-learning conditions, we need

$$2ne^{-\frac{m\epsilon}{2n}} < \delta$$

after arrangements:

$$m \geq \frac{2n}{\epsilon}\left(\ln 2n + \ln \frac{1}{\delta}\right)$$

Thus $m \leq \text{poly}(1/\epsilon, 1/\delta, n)$ example suffice to make $\text{err}(h) \leq \epsilon$ with probability at least $1 - \delta$.

So the generalization algorithm PAC-learns conjunctions.

Mistake-Bound Learnability Implies PAC-Learnability

Any mistake-bound learner L can be transformed into a PAC-learner. Let $M \leq \text{poly}(n)$ be the mistake bound of L.

Call L *lazy* if it changes its hypo h *only* following a mistake. If L is not lazy, make it lazy (prevent changing h after correct decisions).

Run L on the example set but halt if any hypo h survives more than $\frac{1}{\epsilon} \ln\left(\frac{M}{\delta}\right)$ *consecutive* examples. Output h.

Observe that this *will* terminate within $m = \frac{M}{\epsilon} \ln\left(\frac{M}{\delta}\right)$ examples. (Otherwise more than M mistakes would be made.)

Mistake-Bound Learnability Implies PAC-Learnability

Prob. that $\text{err}(h) > \epsilon$ is at most

$$M(1-\epsilon)^{\frac{1}{\epsilon}\ln\frac{M}{\delta}} < Me^{-\frac{\epsilon}{\epsilon}\ln\frac{M}{\delta}} = M\frac{\delta}{M} = \delta$$

Since $M \leq \text{poly}(n)$ (condition of MB learning), also

$$m = \frac{M}{\epsilon}\ln\left(\frac{M}{\delta}\right) \leq \text{poly}(1/\epsilon, 1/\delta, n)$$

So all PAC-learning conditions satisfied: we have $m \leq \text{poly}(1/\epsilon, 1/\delta, n)$, and $\text{err}(h) \leq \epsilon$ with prob. at least $1-\delta$.

PAC-Learning Implies Consistency

Although $\text{err}(h) > 0$ is allowed, the output h of a PAC-learner is necessarily consistent with all the training examples (zero "training error").

Assume that given training set $\{ x_1, x_2, \ldots x_m \}$, the algo outputs h *inconsistent* with some x_j $(1 \leq j \leq m)$.

Distribution $P(x)$ and numbers ϵ, δ arbitrary so set them such that

- $\prod_{i=1}^{m} P(x_i) > \delta$ implying that $P(x_j) > 0$;
- $\epsilon < P(x_j)$

So with prob. $> \delta$ the algo will output h such that $\text{err}(h) \geq P(x_j) > \epsilon$, i.e. it *does not* PAC-learn.