The WINNOW Algorithm

An algorithm to learn linearly separable concepts on $\{0, 1\}^n$.

Monotone (no negation) conjunctions and monotone disjunctions are linearly separable. Non-monotone convertible to monotone by doubling the number of variables.

WINNOW hypothesis space is R^n, $h = [h_1, h_2, \ldots, h_n]$. h_i are "weights".

Initially $h = [1, 1, \ldots 1]$.

Decision rule: decide "yes" if $\sum_{i=1}^{n} h_i \cdot x_i > n/2$, else "no".

Similar to the well-known perceptron algo. Main difference is the learning rule.

The WINNOW Algorithm: Learning rule

Unlike the perceptron, WINNOW adapts weights *multiplicatively*.

When an example x is misclassified, h changes to h':

- If x is positive (i.e., "false negative"), *double* all h_i where $x_i = 1$:

$$h'_i = 2h_i$$

- If x is negative (i.e., "false positive"), *nullify* all h_i where $x_i = 1$:

$$h'_i = 0$$

Other weights ($\forall i, x_i = 0$) remain same.

Let us develop a mistake bound for WINNOW learning *monotone k-disjunctions*, i.e., monotone disjunctions of up to $k \in N$ variables.

No weight in h ever becomes negative.

- Only doublings and nullifications from the initial $h = [1, 1, \ldots, 1]$

No weight in h ever exceeds n.

- Assume for contradiction that some $h_j \leq n$ gets doubled to $h'_j > n$ (i.e., $h_j > n/2$) after an example x.
- $x_j = 1$ as otherwise h_j would not get doubled.
- Doubling occurs only after a false negative so $\sum_{i=1}^{n} h_i \cdot x_i \leq n/2$. But that contradicts $h_j > n/2$ considering none of h_i is negative.

The WINNOW Algorithm: Analysis

The total increase in weights after a *false negative* x is at most $n/2$:

$$\sum_{i=1}^{n} h'_i - \sum_{i=1}^{n} h_i = \sum_{i=1}^{n}(h'_i - h_i)x_i = \sum_{i=1}^{n}(2h_i - h_i)x_i = \sum_{i=1}^{n} h_i x_i \leq \frac{n}{2}$$

- first equality due to $h'_i = h_i$ when $x_i = 0$
- second equality due to the doubling rule
- last inequality due to the decision rule and the fact that x was classified negative

The total decrease in weights after a *false positive* x is larger than $n/2$ (shown analogically).

The WINNOW Algorithm: Analysis

For the initial hypothesis, $\sum_{i=1}^{n} h_i = \sum_{i=1}^{n} 1 = n$.

After $\mathcal{N}$ false negatives and $\mathcal{P}$ false positives (using the results from the previous page):

$$0 \leq \sum_{i=1}^{n} h_i \leq n + \mathcal{N}\frac{n}{2} - \mathcal{P}\frac{n}{2}$$

thus

$$\mathcal{P}\frac{n}{2} \leq n + \mathcal{N}\frac{n}{2}$$

i.e. $(n > 0)$,

$$\mathcal{P} \leq 2 + \mathcal{N}$$

The WINNOW Algorithm: Analysis

On each false negative, at least one of the k weights corresponding to the k variables in the concept disjunction gets doubled. (At least one of these variables must have been true for the disjunction to be true.)

So after $\mathcal{N}$ false negatives, one of them (h_j) was doubled at least $\mathcal{N}/k$ times so

$$h_j \geq 2^{\frac{\mathcal{N}}{k}}$$

i.e.,

$$\lg h_j \geq \frac{\mathcal{N}}{k}$$

We have shown that no h_i exceeds n. So $\lg h_j \leq \lg n$ and

$$\lg n \geq \frac{\mathcal{N}}{k}$$

The WINNOW Algorithm: Analysis

So we have a bound for the false negatives

$$\mathcal{N} \leq k \lg n$$

and since we have shown that $\mathcal{P} \leq 2 + \mathcal{N}$, we have a total *mistake bound*

$$\mathcal{P} + \mathcal{N} \leq 2 + 2k \lg n$$

The $\lg n$ factor makes WINNOW much faster than the generalization algorithm or the perceptron when k is a small ($k \ll n$) constant.

$k \ll n$ means a "sparse" target concept disjunction—many irrelevant attributes.

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 $\mathcal{C}$:

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

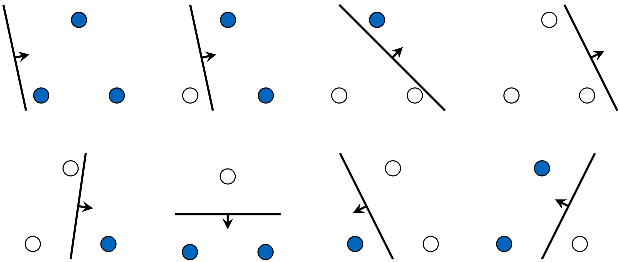
$\mathrm{VC}(\mathcal{H})$ for a *hypothesis* class $\mathcal{H}$ defined analogically.

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 $\mathrm{VC}(\mathcal{C}) \geq 3$.



- *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 $\mathrm{VC}(\mathcal{C}) = 3$.

Generally, $\mathrm{VC}(\text{half-planes in } R^n) = n + 1$

Poly VC-Dimension Necessary for Learnability

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

Proof: There exists a set of $\text{VC}(\mathcal{C})$ instances from X shattered by $\mathcal{C}$ so there exists a sequence $x_1, x_2, \ldots x_{\text{VC}(\mathcal{C})}$ 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 $\lg |\mathcal{C}| \leq \text{poly}(n)$ implies $\text{VC}(\mathcal{C}) \leq \text{poly}(n)$ but not the other way around.

$\text{VC}(\mathcal{C})$ may be finite (even $\text{poly}(n)$) 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.