Notes 3: Perceptron and Halving algorithms

1. Perceptron algorithm

Update weights **additively**
Learn well-separated (i.e. large margin) LTF
Normalization: threshold $\theta = 0$ (halfspace through origin)
Reason: Add extra coordinate $x_{n+1} = 1$ to every instance

$$w \cdot (x_1, \dots, x_n) \geqslant \theta \qquad \iff \qquad (w, -\theta) \cdot (x_1, \dots, x_{n+1}) \geqslant 0$$

Normalization: Every sample x has unit length, i.e. $\|x\| = 1$ (recall $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$)
Reason: By previous assumption $\theta = 0$; rescaling x doesn't change the sign of $w \cdot x$
Normalization: weight vector w has unit length

Perceptron

Initialize: $w = 0$
On input x, output hypothesis $h(x) = \mathbb{1}(w \cdot x \geqslant 0)$ and get $c(x)$
False positive ($h(x) = 1, c(x) = 0$): Update w as $w - x$
False negative ($h(x) = 0, c(x) = 1$): Update w as $w + x$

On false positive, $w \cdot x$ is too big, so subtract x from w, so that $(w - x) \cdot x = w \cdot x - \|x\| = w \cdot x - 1$
On false negative, $w \cdot x$ is too small, so add x to w, so that $(w + x) \cdot x = w \cdot x + \|x\| = w \cdot x + 1$

Theorem 1.1. *(Perceptron convergence) Let $c(x) = \mathbb{1}(v \cdot x \geqslant 0)$ be centered LTF with $\|v\| = 1$. Suppose all samples x has unit length, let margin δ be $\min|v \cdot x|$ over all samples x received by the algorithm. Then Perceptron Algorithm learns c with at most $1/\delta^2$ mistakes*

Claim 1.2. *After M mistakes, $w \cdot v \geqslant \delta M$*

Proof. True when $M = 0$ since $w = 0$
Will show that every mistake increases $w \cdot v$ by $\geqslant \delta$
On false positive, $w \cdot v$ becomes $(w - x) \cdot v = w \cdot v - x \cdot v \geqslant w \cdot v + \delta$
On false negative, $w \cdot v$ becomes $(w + w) \cdot v = w \cdot v + x \cdot v \geqslant w \cdot v + \delta$ ☐

Claim 1.3. *After M mistakes, $\|w\|^2 \leqslant M$*

Proof. True when $M = 0$ since $w = 0$
Will show that every mistake increases $\|w\|^2$ by $\leqslant 1$
On false positive, $\|w\|^2$ becomes $\|w - x\|^2 = (w - x) \cdot (w - x) = \|w\|^2 - 2\underbrace{w \cdot x}_{\geqslant 0} + \underbrace{\|x\|^2}_{=1}$
On false negative, $\|w\|^2$ becomes $\|w + x\|^2 = (w + x) \cdot (w + x) = \|w\|^2 + 2\underbrace{w \cdot x}_{<0} + \underbrace{\|x\|^2}_{=1}$ ☐

Proof of Perceptron Convergence. $\delta M \leqslant w \cdot v \underbrace{\leqslant}_{\text{Cauchy–Schwarz}} \|w\| \underbrace{\|v\|}_{=1} \leqslant \sqrt{M}$ ☐

The above bound is tight!

Claim 1.4. *When $X = \{x \in \mathbb{R}^d \mid \|x\| = 1\}$ and $d \geqslant 1/\delta^2$, any deterministic algorithm for learning LTF with margin δ makes $\lfloor 1/\delta^2 \rfloor$ mistakes in the worst case*

Proof. ith x^i sample is ith standard basis vector e_i (i.e. 1 at position i and 0 elsewhere)
Number of samples is $n \stackrel{\text{def}}{=} \lfloor 1/\delta^2 \rfloor$ (as most d by assumption)
All samples will be labeled as the opposite of algorithm's prediction
Will find $v \in \mathbb{R}^d$ with $\|v\| \leqslant 1$ that "correctly" classifies all e_i with margin δ, i.e.

$$\forall \text{ "correct label sequence" } y \in \{1, -1\}^n, \qquad y_i \delta = v \cdot e_i$$

This forces $v_i = \delta y_i$ for all $i \leqslant n$

Indeed $\|v\|^2 = \delta^2 \|y\|^2 = \delta^2 n \leqslant 1$ $\square$

---

2. Dual perceptron

In Perceptron Algorithm w always ± 1-sum of samples, i.e. $\exists$ signs $\sigma_1, \dots, \sigma_\ell \in \{1, -1\}$ s.t.

$$w = \sigma_1 x^{i_1} + \cdots + \sigma_\ell x^{i_\ell}$$

Initially $w = 0$; Every mistake adds a new term $\sigma_j x^{i_j}$ to w

Memorizing all mistakes, on sample x,

$$w \cdot x = \sum_{1 \leqslant j \leqslant \ell} \sigma_j (x^{i_j} \cdot x)$$

Computable given inner products $x^{i_j} \cdot x$ between samples
Now takes #mistakes time to compute w (slower)
Can replace inner product $\cdot$ with any **kernel function** $K(,)$

---

3. Halving algorithm

Given any finite concept class $\mathcal{C}$

Halving Algorithm

K always contains all $c \in \mathcal{C}$ consistent with all labeled samples so far (initially $K = \mathcal{C}$)
On sample x, predicts according to majority vote over concepts in K (then update K)

Every mistake removes at least half of concepts from K
Claim: Halving Algorithm makes $\leqslant \log |\mathcal{C}|$ mistakes
Slow: $|K|$ **per round**
Hypothesis isn't from $\mathcal{C}$, but majority over a subset of $\mathcal{C}$

---

4. Randomized halving algorithm

Randomized Halving Algorithm

K always contains all $c \in \mathcal{C}$ consistent with all labeled samples so far (initially $K = \mathcal{C}$)
On sample x, predicts according to a random concept $c \in K$ (then update K)

Claim 4.1. *On any sequence of samples $x^1, \dots, x^m$ labeled by any $c \in \mathcal{C}$,*

$$\mathbb{E}[\#\text{mistakes of the algorithm}] \leqslant \ln |\mathcal{C}| + O(1)$$

Proof. Fix $c \in \mathcal{C}$ and $x^1, \dots, x^m$
Suppose at some point $|\mathcal{K}| = r$, let $M_r = \mathbb{E}[\#\text{future mistakes}]$
Need to bound $M_{|\mathcal{C}|}$

Order concepts $c_1, \dots, c_r$ in K according to when they are eliminated by the sequence
e.g. first eliminated batch $c_1, \dots, c_3$, next c_4, c_5 etc, finally $c_r = c$ never eliminated
On first sample x^1, Algorithm randomly chooses one of $c_1, \dots, c_r$
If c_r is chosen, no mistake ($1/r$ chance)
If chosen c_t makes mistake on x^1 ($1/r$ chance for each $t < r$)
$c_1, \dots, c_t$ (and possibly more) must be eliminated
K shrinks to (at most) size $r - t$, expect M_{r-t} more mistakes

$$M_r \leqslant \sum_{1 \leqslant t < r} \frac{1}{r}(1 + M_{r-t}) \qquad \Longrightarrow \qquad r M_r \leqslant \sum_{1 \leqslant t < r} (1 + M_{r-t}) = r - 1 + M_1 + \cdots + M_{r-1}$$

Same for $r - 1$: $(r-1) M_{r-1} = (r-2) + M_1 + \cdots + M_{r-2}$

Subtracting, $r(M_r - M_{r-1}) \leqslant 1$

$$M_r \leqslant \frac{1}{r} + M_{r-1} \leqslant \frac{1}{r} + \frac{1}{r-1} + M_{r-2} \leqslant \ldots \leqslant \underbrace{\frac{1}{r} + \frac{1}{r-1} + \cdots + \frac{1}{1}}_{\text{Harmonic number}} = \ln r + O(1) \qquad \square$$

Constant factor improvement over deterministic halving: $\log |\mathcal{C}| / \ln |\mathcal{C}| = \log e = 1.44\ldots$