CS639: Introduction to Computational Learning Theory	**Scribe:** Changho Shin
Lecture 5: Perceptron and Halving Algorithm	**Lecturer:** Ilias Diakonikolas

5.1 Perceptron

- Algorithm to learn linear threshold functions over $\mathbb{R}^n$.
- Let $\mathbb{1}[v^\intercal x \geq \theta]$ be the target halfspace, where $v = (v_1, ..., v_n) \in \mathbb{R}^n$, $x = (x_1, ..., x_n) \in \mathbb{R}^n$.
- *Assumption 1.* $\theta = 0$ (homogeneous / origin-centered)
  Indeed, we can reduce any θ to $\theta = 0$ case by introducing a new variable $x_{n+1} = 1$ so that

$$v^\intercal x \geq \theta \iff v'^\intercal x' \geq 0$$

  , where $v' = (v_1, ..., v_n, -\theta), x' = (x_1, ..., x_n, x_{n+1})$
- *Assumption 2.* Each example $x \in \mathbb{R}^n$ that we get has $||x||_2 = 1$. Note that rescalling a point x by $||x||_2$ does not change which side of the hyperplane $w^\intercal x = 0$ on example is on.
- *Assumption 3.* We can assure $||v||_2 = 1$
- Perceptron maintains hypothesis vector w so that $h_w(x) = \mathbb{1}[w^\intercal x \geq 0]$

5.1.1 Perceptron Learning Algorithm

- Initially, $w = (0, ..., 0)$.
- Update hypothesis as follows
  - If we get an example x right, no change.
  - If $w^\intercal x \geq 0$, but the true label is $v^\intercal x < 0$, update $w \leftarrow w - x$. (False positive)
  - If $w^\intercal x < 0$, but the true label is $+$, update $w \leftarrow w + x$. (False negative)
- Intuition, consider a false positive case, so we updated w as $w_{new} \leftarrow w_{old} - x$. Then, $w_{new}^\intercal x = (w_{old} - x)^\intercal x = w_{old}^\intercal x - 1 \leq w_{old}^\intercal x$, which shows that the update reduce the function value for the x.

5.1.2 Perceptron Convergence Theorem

Theorem 5.1.1 *Suppose we run Perceptron to learn halfspace/LTF $v^\intercal x \geq 0$, where assumptions 1 3 hold. Let the margin $\delta = min|v^\intercal x|$ over all examples x given to algorithm. Then, Perceptron makes $\leq \frac{1}{\delta^2}$ mistakes.*

Lemma 5.1.2 *After Perceptron algorthm has made M mistakes, we have $w^\intercal v \geq \delta M$.*

Proof: We will show that each mistake increases $w^\intercal by \geq \delta$.

- Initially, $M = 0, w = (0, ..., 0) \rightarrow$ the proposition trivially holds.
- If mistake is on a positive example x, $w_{new} = w_{old} + x$. Thus,

$$\begin{aligned} w_{new}^{\intercal} v &= (w_{old} + x)^{\intercal} v \\ &= w_{old}^{\intercal} v + x^{\intercal} v \\ &\geq w_{old}^{\intercal} v + \delta \quad \because \text{by the definition of } \delta \end{aligned}$$

  Recall $\delta = min|v^{\intercal}x|$ over all x in the input. Since x is positive, $v^{\intercal}x \geq 0$.
- If mistake is on a negative example x, $w_{new} = w_{old} - x$. Thus,

$$(w_{old} - x)^{\intercal} v = w_{old}^{\intercal} v - x^{\intercal} v \geq w_{old}^{\intercal} v + \delta$$

  since $x^{\intercal}v < 0$ and $|x^{\intercal}v| \geq \delta$.

■

Lemma 5.1.3 *After M mistakes, $||w||_2 \leq M$.*
Proof:

- Initially, $M = 0$ and $||w||_2 = 0$, so the proposition trivially holds.
- If mistake is on positive example x,

$$||w + x||_2^2 = ||w||_2^2 + 2\underbrace{w^{\intercal}x}_{<0} + ||x||_2^2 < ||w||_2^2 + 1$$

- If mistake is on negative example x,

$$||w - x||_2^2 = ||w||_2^2 - 2\underbrace{w^{\intercal}x}_{>0} + ||x||_2^2 < ||w||_2^2 + 1$$

■

Proof of 5.2.1: Since $w^{\intercal}v = ||w||_2 \underbrace{||v||_2}_{=1} \underbrace{\cos\theta(w, v)}_{\leq 1}$, $w^{\intercal}v \leq ||w||_2$. Thus,

$$\delta M \leq w^{\intercal}v \leq ||w||_2 \leq \sqrt{M}$$

From the inequality above, we can show $M \leq \frac{1}{\delta^2}$ ■

Remarks

- Perceptron is
  - simple

    - noise-tolerant
    - kernelizable

- But in general $\frac{1}{\delta^2}$ bound is not always good. For example, for decision lists $\{0,1\}^n$, $\delta = 2^{-\Theta(n)}$. There exist different algorithms (e.g. Elipsoid algorithm) with mistake bound $\log(\frac{1}{\delta})^2$, but more complex, each update slower, not noise tolerant.

- Best of both worlds: Dunagan-Vempala's "Rescaled Perceptron"

5.2 Generic Algorithm and Bounds for Online Learning (Halving Algorithm)

Assume that the concept class C is finite. Halving algorithm learns any finite concept class C with $\leq \log_2 |C|$ mistakes.

5.2.1 Halving Algorithm Procedure

- Let $CONSIST \subseteq C$ by the set of all $c \in C$ which are consistent with all labeled examples seen so far.

- Initial $CONSIST = C$

- Given example x, the hypothesis that the algorithm uses is the majority vote over all concepts over $CONSIST$ on x.

- After you receive the true value $c(x)$, update $CONSIST$ so that only consistent hypotheses are included in $CONSIST$.

5.2.2 Analysis on Halving Algorithm

Theorem 5.2.1 *For any finite concept class C, Halving algorithm has the mistake bound $\leq \log_2 |C|$.*

Proof: Every mistake causes new value of $|CONSIST|$ to become at most half of old volume. ■

5.2.3 Examples of Halving Algorithm

Decision List

- $C = \{$all DLs over $x_1, ..., x_n$ with length-$r\}$.

- Recall $|C| \leq (4n+2)^{r+1}$, which leads to $log_2|C| = O(r \log n)$ by halving algorithm. Note that out algorithm the lecture 2, 3 had the mistake bound $O(rn)$. It turns out that the upper bound by halving algorithm is optimal for any algorithm.

- Drawbacks

    - Very inefficient (time, space at least $|C|$).

  - Not noise tolerant.

- In some cases, we can match this mistake upper bound with efficient algorithm. (e.g. disjunctions).

Delta function

- Halving algorithm can sometimes make even fewer mistakes than $\log_2 |C|$.
- Consider $X = \{1, ..., N\}$,
$$\delta_i(j) = \begin{cases} 1, \text{ if } j = i \\ 0, j \neq i \end{cases}$$
and $C = \{\text{all } N \text{ functions } \delta_i \text{ (singletons)}\}$. Then, halving algorithm makes 1 mistake.

Remarks

- In general, we don't have a computationally efficient analogue of halving algorithm.
- There exists noise tolerant version of halving algorithm.