CS 412 — Introduction to Machine Learning (UIC)	March 15, 2025

Lecture 16

Instructor: Aadirupa Saha *Scribe(s): Nemi Chakrawarthy Bhupathiraju/Simran Mishra*

Overview

In the last lecture, we covered the following main topics:

1. Perceptron Mistake Bounds

This lecture focuses on:

1. Winnow Algorithm
2. Mistake Bound for Perceptron (Review)

1 Revisiting Perceptron

1.1 Online Learning Setup

Algorithm 1.1: Online Perceptron Update Rule

1: **At each round** t:
2: Receive input $x_t \in \mathbb{R}^d$
3: Predict $\hat{y}_t = \text{sign}(w_t^\top x_t)$
4: Observe true label $y_t \in \{-1, +1\}$
5: **if** $\hat{y}_t \neq y_t$ **then**
6: Update: $w_{t+1} \leftarrow w_t + y_t x_t$
7: **else**
8: $w_{t+1} \leftarrow w_t$
9: **end if**

1.2 Mistake Bound for Perceptron

Theorem 1.1: Mistake Bound for Linearly Separable Case

Suppose there exists a hyperplane w^* such that:

$$w^{*T} x = 0$$

and it separates all data points with margin γ. Then the total number of mistakes made by the Perceptron algorithm is bounded by:

$$\#\text{mistakes} \leq \frac{\|w^*\|_2^2 \cdot \max_x \|x\|_2^2}{\gamma^2}$$

Assumption 1. *Assume $x \in [0,1]^d$, so $\|x\|_2^2 \le d$. For example, if $d = 50$, then $\|x\|_2^2 \le 50$.*

Remark 1. *The mistake bound improves (decreases) with:*

1. *Smaller $\|x\|$ norm (e.g., binary or normalized inputs)*
2. *Larger margin γ*

1.3 Monotone Disjunction Labeling and Example

Theorem 1.2: Monotone Disjunction Model

In binary classification with input vectors $x \in \{0,1\}^d$, a common labeling function is the **monotone disjunction**, defined as:

$$y = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$$

This represents a logical OR over a subset of input variables, where no negation is involved. The function outputs 1 if at least one of the relevant variables is 1, and 0 otherwise.

The binary OR function behaves as follows:

x_1	x_2	$x_1 \vee x_2$
1	1	1
1	0	1
0	1	1
0	0	0

Example: Small Monotone Disjunction with Relevant Features
Let the number of features be $d = 5$, and suppose only x_1 and x_4 are relevant for determining the label.

- Each instance $x \in \{0,1\}^5$
- Labeling function:
$$y = x_1 \vee x_4$$

Sample Inputs and Outputs:

x	$y = x_1 \vee x_4$
$(1,0,1,0,0)$	$1 \vee 0 = 1$
$(0,1,1,0,0)$	$0 \vee 0 = 0$
$(0,1,1,1,0)$	$0 \vee 1 = 1$
$(0,0,0,0,0)$	$0 \vee 0 = 0$

Interpretation: The label depends only on the presence of 1 in the relevant positions (here x_1 and x_4), ignoring the rest. This illustrates sparse logical concept learning.

2 Winnow Algorithm for r-Monotone Disjunctions

In this section, we study a simple version of the Winnow algorithm, a multiplicative weight update method suitable for learning sparse Boolean functions like monotone disjunctions. The key insight is that the algorithm does not need to know the target sparsity level r in advance.

2.1 Winnow Algorithm Description

Algorithm 2.1: Winnow Algorithm with Parameter β

Initialization: Set $w_1 = \mathbf{1} \in \mathbb{R}^d$
Input: Data stream $\{x_t\}_{t=1}^T \subset \{0,1\}^d$, threshold d, learning rate $\beta > 0$

For each round $t = 1, 2, \ldots, T$:

1. Receive $x_t \in \{0,1\}^d$
2. Predict:
$$\hat{y}_t = \begin{cases} 1 & \text{if } w_t^\top x_t \geq d \\ 0 & \text{otherwise} \end{cases}$$
3. Observe true label $y_t \in \{0,1\}$
4. If $\hat{y}_t \neq y_t$, update each coordinate:
$$w_{t+1}(i) = \begin{cases} w_t(i)(1+\beta) & \text{if } x_t(i) = 1 \text{ and } y_t = 1 \\ \frac{w_t(i)}{1+\beta} & \text{if } x_t(i) = 1 \text{ and } y_t = 0 \\ w_t(i) & \text{if } x_t(i) = 0 \end{cases}$$
5. Else: $w_{t+1} = w_t$

Note: When $\beta = 1$, the updates double or halve the weights.

2.2 Intuition Behind Winnow ($\beta = 1$)

Case 1: Mistake on Positive Class

$$\hat{y}_t = 0, \quad y_t = 1 \quad \text{(model predicted 0, should be 1)}$$

$$w_t^\top x_t < d \Rightarrow \text{Need to increase the score of } x_t$$

Let $\mathcal{A} = \{i \mid x_t(i) = 1\}$. Then:

$$\sum_{i \in \mathcal{A}} w_t(i) < d \quad \text{(under-predicting the score)}$$

Update: For all $i \in \mathcal{A}$:

$$w_{t+1}(i) = 2 \cdot w_t(i) \Rightarrow \sum_{i \in \mathcal{A}} w_{t+1}(i) = 2 \cdot w_t^\top x_t$$

This increases the score to move it toward correct classification.

Case 2: Mistake on Negative Class

$$\hat{y}_t = 1, \quad y_t = 0 \quad \text{(model predicted 1, should be 0)}$$

$$w_t^\top x_t \geq d \Rightarrow \text{Need to decrease the score of } x_t$$

Let $\mathcal{A} = \{i \mid x_t(i) = 1\}$. Then:

$$\sum_{i \in \mathcal{A}} w_t(i) \geq d$$

Update: For all $i \in \mathcal{A}$:

$$w_{t+1}(i) = \frac{w_t(i)}{2} \Rightarrow \text{Score gets halved: } w_{t+1}^\top x_t = \frac{w_t^\top x_t}{2}$$

2.3 Mistake Bound Comparison

Winnow Mistake Bound

$$M_{\text{Winnow}} \leq 3r(1 + \log d)$$

- r: number of relevant features
- d: total number of features

Example: For $r = 2, d = 100$,

$$M_{\text{Winnow}} \leq 3 \cdot 2 \cdot \log_2 100 \approx 12$$

Perceptron Mistake Bound

$$M_{\text{Perceptron}} = O\left(\frac{\|w^*\|^2 \cdot \max_x \|x\|^2}{\gamma^2}\right)$$

Remarks:

- If $\|x\|^2 \leq 1$ and $\gamma = 1/\sqrt{d}$, then:
$$M_{\text{Perceptron}} = O(d)$$
- As $\gamma \to 0$, this bound worsens significantly.

2.4 Why Use Winnow?

When is Winnow better than Perceptron?

- The target function depends on only a small number of features (i.e., $r \ll d$)
- The input vectors are binary and sparse
- High-dimensional feature space where linear separators still exist

Conclusion:
Winnow is highly effective in sparse Boolean settings where only a few features are truly relevant. Its logarithmic dependence on d makes it scalable and preferable to Perceptron when margin is small or unknown.

3 Proof of Winnow Mistake Bound (for $\beta = 1$)

We now provide a proof for the mistake bound of the Winnow algorithm, considering the special case when the multiplicative update parameter $\beta = 1$.

3.1 Case 1: When $y_t = 1$ but $\hat{y}_t = 0$

This case occurs when the true label is positive, but the algorithm incorrectly predicts a negative label. This happens precisely when:

$$w_t^\top x_t < d$$

Observation 1: At least one of the active weights $w_t(i)$ (where $x_t(i) = 1$) will be doubled after each such mistake. This follows directly from the multiplicative update rule of the Winnow algorithm, where the weights of active features are multiplied by $1 + \beta$ when a false negative occurs.

Observation 2: If for any coordinate $i \in [d]$, $x_t(i) = 0$, then:

- The weight $w_t(i)$ is not updated at that round.
- Only the weights corresponding to active coordinates $x_t(i) = 1$ are updated (i.e., doubled).

Furthermore, once the sum $w_t^\top x_t \geq d$ is achieved, the prediction will correctly switch to $\hat{y}_t = 1$ and no further mistakes of this type will occur for that input.
Thus, the algorithm's goal is to increase the weighted sum enough through doubling to eventually predict correctly.

Tracking the Number of Updates: Let ℓ_i denote the total number of times the weight $w_t(i)$ for coordinate i has been increased (i.e., doubled).
Initially:

$$w_1(i) = 1$$

After ℓ_i updates:

$$w_t(i) = 2^{\ell_i}$$

However, since the prediction threshold is d, we have:

$$2^{\ell_i} \leq d \quad \Rightarrow \quad \ell_i \leq \log_2 d$$

Thus, the number of times any coordinate's weight can double is at most $\log_2 d$.

Conclusion for Case 1:

- Each relevant feature i can contribute at most $\log_2 d$ mistakes.
- If there are r relevant features, the total number of mistakes of the form $y_t = 1$ but $\hat{y}_t = 0$ is bounded by:

$$r \log_2 d$$

Thus, the total number of mistakes for this case is $O(r \log d)$.

3.2 Case 2: When $y_t = -1$ but $\hat{y}_t = +1$

This case corresponds to making a mistake on a negative label: the true label is $y_t = -1$, but the algorithm incorrectly predicts $\hat{y}_t = +1$.
This happens when:

$$w_t^\top x_t \geq d$$

Observation 1: Initially at $t = 1$:

$$w_1(i) = 1 \quad \text{for all } i \in [d]$$

Thus, the total initial weight is:

$$\sum_{i=1}^{d} w_1(i) = d$$

Observation 2: Suppose at time t, the algorithm makes a mistake with $y_t = -1$ and $\hat{y}_t = +1$.
Thus:

$$w_t^\top x_t = \sum_{i=1}^{d} w_t(i) x_t(i) \geq d$$

After this mistake:

- For every i such that $x_t(i) = 1$, the weight is updated as:

$$w_{t+1}(i) = \frac{w_t(i)}{2}$$

- For coordinates with $x_t(i) = 0$, weights remain unchanged.

Thus, the new weighted sum becomes:

$$w_{t+1}^\top x_t = \sum_{i=1}^{d} w_{t+1}(i) x_t(i) = \frac{1}{2} \sum_{i: x_t(i)=1} w_t(i) = \frac{1}{2} w_t^\top x_t$$

Since $w_t^\top x_t \geq d$ before the update, we get:

$$w_{t+1}^\top x_t \leq \frac{d}{2}$$

Thus, after a mistake on a negative label, the total weighted score decreases by at least $d/2$.

Observation 3: Conversely, when making a mistake on a positive example ($y_t = +1$, $\hat{y}_t = -1$):

- Active weights are doubled:
$$w_{t+1}(i) = 2w_t(i)$$
- Thus, the new weighted sum becomes:
$$w_{t+1}^\top x_t = 2w_t^\top x_t$$
- Since before the mistake $w_t^\top x_t < d$, it follows that:
$$w_{t+1}^\top x_t < 2d$$

Thus, after a mistake on a positive example, the total weighted sum increases by at most d.

Combining Observations: Define:

- P_t = number of mistakes made on positive examples ($y_t = +1$, $\hat{y}_t = -1$),
- Q_t = number of mistakes made on negative examples ($y_t = -1$, $\hat{y}_t = +1$).

From Observations 2 and 3:
$$0 < \sum_{i=1}^{d} w_t(i) \leq d + P_t \times d - Q_t \times \frac{d}{2}$$
Dividing by d:
$$0 < 1 + P_t - \frac{Q_t}{2}$$
Rearranging:
$$\frac{Q_t}{2} < 1 + P_t \quad \Rightarrow \quad Q_t < 2(1 + P_t) \quad \Rightarrow \quad Q_t \leq 2 + 2P_t$$
From Case 1, we know:
$$P_t \leq r \log_2 d$$
Substituting:
$$Q_t \leq 2 + 2r \log_2 d$$
Thus, the total number of mistakes is:
$$P_t + Q_t \leq r \log_2 d + 2 + 2r \log_2 d = 2 + 3r \log_2 d$$

Final Conclusion: Thus, the total number of mistakes made by Winnow satisfies:
$$O(r \log d)$$
This matches the earlier mistake bound result.

4 Next Lecture:

The next topic will be:

- **AdaBoost**: Combining weak learners to design a strong learner.

References

1. Winnow: CS 4540 - Link
2. Mistake Bound: CS260 -Link
3. Winnow Example -Link