Perceptron Mistake Bound
Reminders

• Homework A:
  – Out: Tue, Oct. 29
  – Due: Wed, Nov. 7 at 11:59pm
Q&A
THE PERCEPTRON ALGORITHM
Perceptron Algorithm: (without the bias term)

1. Set $t=1$, start with all-zeroes weight vector $w_1$.
2. Given example $x$, predict positive iff $w_t \cdot x \geq 0$.
3. On a mistake, update as follows:
   - Mistake on positive, update $w_{t+1} \leftarrow w_t + x$
   - Mistake on negative, update $w_{t+1} \leftarrow w_t - x$

Example:

- $(-1,2) - \times$
- $(1,0) + \checkmark$
- $(1,1) + \times$
- $(-1,0) - \checkmark$
- $(-1,-2) - \times$
- $(1,-1) + \checkmark$

Slide adapted from Nina Balcan
Background: Hyperplanes

Hyperplane (Definition 1):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} = b \} \]

Hyperplane (Definition 2):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} = 0 \text{ and } x_0 = 1 \} \]
\[ \mathbf{\theta} = [b, w_1, \ldots, w_M]^T \]

Half-spaces:
\[ \mathcal{H}^+ = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1 \} \]
\[ \mathcal{H}^- = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1 \} \]

Notation Trick: fold the bias \( b \) and the weights \( \mathbf{w} \) into a single vector \( \mathbf{\theta} \) by prepending a constant to \( \mathbf{x} \) and increasing dimensionality by one!
(Online) Perceptron Algorithm

**Data:** Inputs are continuous vectors of length $M$. Outputs are discrete.

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots$$
where $x \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

**Prediction:** Output determined by hyperplane.

$$\hat{y} = h_\theta(x) = \text{sign}(\theta^T x)$$

Assume $\theta = [b, w_1, \ldots, w_M]^T$ and $x_0 = 1$

**Learning:** Iterative procedure:

- **initialize** parameters to vector of all zeroes
- **while** not converged
  - **receive** next example $(x^{(i)}, y^{(i)})$
  - **predict** $y' = h(x^{(i)})$
  - **if** positive mistake: **add** $x^{(i)}$ to parameters
  - **if** negative mistake: **subtract** $x^{(i)}$ from parameters
(Online) Perceptron Algorithm

**Data:** Inputs are continuous vectors of length $M$. Outputs are discrete.

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots$

where $x \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

**Prediction:** Output determined by hyperplane.

$y = h_{\theta}(x) = \text{sign}(\theta^T x)$

Assume $\theta = [b, w_1, \ldots, w_M]$

**Learning:**

**Algorithm 1** Perceptron Learning Algorithm

1: procedure PERCEPTRON($D = \{(x^{(i)}, y^{(i)})\}$)
2:   $\theta \leftarrow 0$
3:   for $i \in \{1, 2, \ldots\}$ do
4:     $\hat{y} \leftarrow \text{sign}(\theta^T x^{(i)})$
5:     if $\hat{y} \neq y^{(i)}$ then
6:       $\theta \leftarrow \theta + y^{(i)} x^{(i)}$
7:   return $\theta$

**Implementation Trick:** same behavior as our “add on positive mistake and subtract on negative mistake” version, because $y^{(i)}$ takes care of the sign
Learning for Perceptron also works if we have a fixed training dataset, D. We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

**Algorithm 1** Perceptron Learning Algorithm (Batch)

1: procedure **PERCEPTRON**($\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$)
2: $\theta \leftarrow 0$ ▷ Initialize parameters
3: while not converged do
4: for $i \in \{1, 2, \ldots, N\}$ do ▷ For each example
5: $\hat{y} \leftarrow \text{sign}(\theta^T x^{(i)})$ ▷ Predict
6: if $\hat{y} \neq y^{(i)}$ then ▷ If mistake
7: $\theta \leftarrow \theta + y^{(i)} x^{(i)}$ ▷ Update parameters
8: return $\theta$
(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

Discussion:
The Batch Perceptron Algorithm can be derived in two ways.

1. By extending the online Perceptron algorithm to the batch setting (as mentioned above)
2. By applying Stochastic Gradient Descent (SGD) to minimize a so-called Hinge Loss on a linear separator
Extensions of Perceptron

• **Voted Perceptron**
  – generalizes better than (standard) perceptron
  – memory intensive (keeps around every weight vector seen during training, so each one can vote)

• **Averaged Perceptron**
  – empirically similar performance to voted perceptron
  – can be implemented in a memory efficient way (running averages are efficient)

• **Kernel Perceptron**
  – Choose a kernel $K(x', x)$
  – Apply the kernel trick to Perceptron
  – Resulting algorithm is still very simple

• **Structured Perceptron**
  – Basic idea can also be applied when $y$ ranges over an exponentially large set
  – Mistake bound does not depend on the size of that set
ANALYSIS OF PERCEPTRON
**Geometric Margin**

**Definition:** The margin of example $x$ w.r.t. a linear sep. $w$ is the distance from $x$ to the plane $w \cdot x = 0$ (or the negative if on wrong side)
Geometric Margin

**Definition:** The margin of example $x$ w.r.t. a linear sep. $w$ is the distance from $x$ to the plane $w \cdot x = 0$ (or the negative if on wrong side).

**Definition:** The margin $\gamma_w$ of a set of examples $S$ wrt a linear separator $w$ is the smallest margin over points $x \in S$. 
**Geometric Margin**

**Definition:** The margin of example \( x \) w.r.t. a linear sep. \( w \) is the distance from \( x \) to the plane \( w \cdot x = 0 \) (or the negative if on wrong side).

**Definition:** The margin \( \gamma_w \) of a set of examples \( S \) wrt a linear separator \( w \) is the smallest margin over points \( x \in S \).

**Definition:** The margin \( \gamma \) of a set of examples \( S \) is the maximum \( \gamma_w \) over all linear separators \( w \).
**Def:** For a binary classification problem, a set of examples $S$ is **linearly separable** if there exists a linear decision boundary that can separate the points.
Analysis: Perceptron

Perceptron Mistake Bound

**Guarantee:** If data has margin $\gamma$ and all points inside a ball of radius $R$, then Perceptron makes $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn’t change the number of mistakes; algo is invariant to scaling.)
Analysis: Perceptron

**Perceptron Mistake Bound**

**Guarantee:** If data has margin $\gamma$ and all points inside a ball of radius $R$, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn’t change the number of mistakes; algo is invariant to scaling.)

**Def:** We say that the (batch) perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

**Main Takeaway:** For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.
**Analysis: Perceptron**

**Perceptron Mistake Bound**

**Theorem 0.1** (Block (1962), Novikoff (1962)).

Given dataset: \( \mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N} \).

Suppose:

1. **Finite size inputs:** \( \|x^{(i)}\| \leq R \)
2. **Linearly separable data:** \( \exists \theta^* \) s.t. \( \|\theta^*\| = 1 \) and \( y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i \)

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

\[
k \leq (R/\gamma)^2
\]
Proof of Perceptron Mistake Bound:

We will show that there exist constants $A$ and $B$ s.t.

$$Ak \leq \|\theta^{(k+1)}\| \leq B\sqrt{k}$$
Analysis: Perceptron

**Theorem 0.1** (Block (1962), Novikoff (1962)).
Given dataset: \( \mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N} \).
Suppose:
1. Finite size inputs: \( \|x^{(i)}\| \leq R \)
2. Linearly separable data: \( \exists \theta^* \text{ s.t. } \|\theta^*\| = 1 \) and \( y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i \)
Then: The number of mistakes made by the Perceptron algorithm on this dataset is

\[
k \leq \left(\frac{R}{\gamma}\right)^2
\]

**Algorithm 1** Perceptron Learning Algorithm (Online)

1: \textbf{procedure} PERCEPTRON(\( \mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\} \))
2: \( \theta \leftarrow 0, k = 1 \) \hspace{1cm} ▷ Initialize parameters
3: \textbf{for} \( i \in \{1, 2, \ldots\} \) \textbf{do}
4: \hspace{0.5cm} \textbf{if} \( y^{(i)}(\theta^{(k)} \cdot x^{(i)}) \leq 0 \) \textbf{then}
5: \hspace{1cm} \( \theta^{(k+1)} \leftarrow \theta^{(k)} + y^{(i)}x^{(i)} \) \hspace{1cm} ▷ Update parameters
6: \hspace{0.5cm} \( k \leftarrow k + 1 \)
7: \textbf{return} \( \theta \)
Analysis: Perceptron

Chalkboard:

– Proof of Perceptron Mistake Bound
Proof of Perceptron Mistake Bound:
Part 1: for some $A$, $A k \leq \|\theta^{(k+1)}\|$ 

$$
\theta^{(k+1)} \cdot \theta^* = (\theta^{(k)} + y^{(i)} x^{(i)}) \theta^*
$$ 

by Perceptron algorithm update

$$
= \theta^{(k)} \cdot \theta^* + y^{(i)} (\theta^* \cdot x^{(i)})
$$ 

$$
\geq \theta^{(k)} \cdot \theta^* + \gamma
$$ 

by assumption

$$
\Rightarrow \theta^{(k+1)} \cdot \theta^* \geq k \gamma
$$ 

by induction on $k$ since $\theta^{(1)} = 0$

$$
\Rightarrow \|\theta^{(k+1)}\| \geq k \gamma
$$ 

since $\|w\| \times \|u\| \geq w \cdot u$ and $\|\theta^*\| = 1$

Cauchy-Schwartz inequality
Proof of Perceptron Mistake Bound:
Part 2: for some \( B \),
\[
\left\| \theta^{(k+1)} \right\| \leq B \sqrt{k}
\]

\[
\left\| \theta^{(k+1)} \right\|^2 = \left\| \theta^{(k)} + y^{(i)} x^{(i)} \right\|^2
\]
by Perceptron algorithm update
\[
= \left\| \theta^{(k)} \right\|^2 + (y^{(i)})^2 \left\| x^{(i)} \right\|^2 + 2y^{(i)} (\theta^{(k)} \cdot x^{(i)})
\]
\[
\leq \left\| \theta^{(k)} \right\|^2 + (y^{(i)})^2 \left\| x^{(i)} \right\|^2
\]
since \( k \)th mistake \( \Rightarrow y^{(i)} (\theta^{(k)} \cdot x^{(i)}) \leq 0 \)
\[
= \left\| \theta^{(k)} \right\|^2 + R^2
\]
since \( (y^{(i)})^2 \left\| x^{(i)} \right\|^2 = \left\| x^{(i)} \right\|^2 = R^2 \) by assumption and \( (y^{(i)})^2 = 1 \)
\[
\Rightarrow \left\| \theta^{(k+1)} \right\|^2 \leq kR^2
\]
by induction on \( k \) since \( (\theta^{(1)})^2 = 0 \)
\[
\Rightarrow \left\| \theta^{(k+1)} \right\| \leq \sqrt{kR}
\]
Proof of Perceptron Mistake Bound:
Part 3: Combining the bounds finishes the proof.

\[ k \gamma \leq \| \theta^{(k+1)} \| \leq \sqrt{kR} \]

\[ \Rightarrow k \leq \left( \frac{R}{\gamma} \right)^2 \]

The total number of mistakes must be less than this.
Analysis: Perceptron

What if the data is not linearly separable?

1. Perceptron will **not converge** in this case (it can’t!)
2. However, Freund & Schapire (1999) show that by projecting the points (hypothetically) into a higher dimensional space, we can achieve a similar bound on the number of mistakes made on one pass through the sequence of examples

---

**Theorem 2.** Let \((x_1, y_1), \ldots, (x_m, y_m)\) be a sequence of labeled examples with \(\|x_i\| \leq R\). Let \(u\) be any vector with \(\|u\| = 1\) and let \(\gamma > 0\). Define the deviation of each example as 

\[
d_i = \max\{0, \gamma - y_i(u \cdot x_i)\},
\]

and define \(D = \sqrt{\sum_{i=1}^{m} d_i^2}\). Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by 

\[
\left( \frac{R + D}{\gamma} \right)^2.
\]
Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm:** when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will **not** converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction