The Perceptron Algorithm

Machine Learning
Spring 2018
Outline

- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
Where are we?

- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
Recall: Linear Classifiers

- Input is a n dimensional vector $\mathbf{x}$
- Output is a label $y \in \{-1, 1\}$

- **Linear Threshold Units** (LTUs) classify an example $\mathbf{x}$ using the following classification rule

  - Output $= \text{sgn}(\mathbf{w}^T \mathbf{x} + b) = \text{sgn}(b + \sum w_i x_i)$

  - $\mathbf{w}^T \mathbf{x} + b \geq 0 \rightarrow$ Predict $y = 1$
  - $\mathbf{w}^T \mathbf{x} + b < 0 \rightarrow$ Predict $y = -1$

  $b$ is called the bias term
Recall: Linear Classifiers

• Input is a n dimensional vector $\mathbf{x}$
• Output is a label $y \in \{-1, 1\}$

• *Linear Threshold Units* (LTUs) classify an example $\mathbf{x}$ using the following classification rule

$$w^T \mathbf{x} + b \geq 0 \quad \Rightarrow \quad \text{Predict } y = 1$$

$$w^T \mathbf{x} + b < 0 \quad \Rightarrow \quad \text{Predict } y = -1$$

$b$ is called the bias term.
The geometry of a linear classifier

$$\text{sgn}(b + w_1 x_1 + w_2 x_2)$$

In $n$ dimensions, a linear classifier represents a hyperplane that separates the space into two half-spaces.

We only care about the sign, not the magnitude.
THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN

F. ROSENBLATT

Cornell Aeronautical Laboratory
The hype

**NEW NAVY DEVICE LEARNS BY DOING**

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) — The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

**HAVING** told you about the giant digital computer known as **IBM 704** and how it has been taught to play a fairly creditable game of chess, we’d like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought.

The first perceptron has yet to be built.

**The New Yorker**, December 6, 1958 P. 44

*The New York Times, July 8 1958*

*The IBM 704 computer*
The Perceptron algorithm

• Rosenblatt 1958

• The goal is to find a separating hyperplane
  – For separable data, guaranteed to find one

• An online algorithm
  – Processes one example at a time

• Several variants exist (will discuss briefly at towards the end)
The Perceptron algorithm

**Input:** A sequence of training examples \((x_1, y_1), (x_2, y_2), \ldots\)
where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r \, (y_i \, x_i)\)
- Return final weight vector
**The Perceptron algorithm**

**Input:** A sequence of training examples \((x_1, y_1), (x_2, y_2), \ldots\)
where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r (y_i x_i)\)
- Return final weight vector

**Remember:**
Prediction = \(\text{sgn}(w^T x)\)

There is typically a bias term also \((w^T x + b)\), but the bias may be treated as a constant feature and folded into \(w\)
The Perceptron algorithm

**Input:** A sequence of training examples \((x_1, y_1), (x_2, y_2), \cdots\)
where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r (y_i x_i)\)
- Return final weight vector

**Remember:**
Prediction = \(\text{sgn}(w^T x)\)
There is typically a bias term also \((w^T x + b)\), but the bias may be treated as a constant feature and folded into \(w\)

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.
The Perceptron algorithm

**Input:** A sequence of training examples \((x_1, y_1), (x_2, y_2), \ldots\) where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r (y_i x_i)\)
- Return final weight vector

**Mistakes:**
- Mistake on positive: \(w_{t+1} \leftarrow w_t + r x_i\)
- Mistake on negative: \(w_{t+1} \leftarrow w_t - r x_i\)
The Perceptron algorithm

**Input:** A sequence of training examples \((x_1, y_1), (x_2, y_2), \ldots\) where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r \ y_i \ x_i\)
- Return final weight vector

Mistake on positive: \(w_{t+1} \leftarrow w_t + r \ x_i\)
Mistake on negative: \(w_{t+1} \leftarrow w_t - r \ x_i\)

\(r\) is the learning rate, a small positive number less than 1
The Perceptron algorithm

**Input:** A sequence of training examples \( (x_1, y_1), (x_2, y_2), \ldots \)
where all \( x_i \in \mathbb{R}^n \), \( y_i \in \{-1,1\} \)

- Initialize \( w_0 = 0 \in \mathbb{R}^n \)
- For each training example \( (x_i, y_i) \):
  - Predict \( y' = \text{sgn}(w_t^T x_i) \)
  - If \( y_i \neq y' \):
    - Update \( w_{t+1} \leftarrow w_t + r (y_i x_i) \)
- Return final weight vector

**Mistake on positive:** \( w_{t+1} \leftarrow w_t + r x_i \)
**Mistake on negative:** \( w_{t+1} \leftarrow w_t - r x_i \)

\( r \) is the learning rate, a small positive number less than 1

Update only on error. A mistake-driven algorithm
The Perceptron algorithm

**Input**: A sequence of training examples \((x_1, y_1), (x_2, y_2), \cdots\) where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r \cdot y_i x_i\)
- Return final weight vector

Mistake on positive: \(w_{t+1} \leftarrow w_t + r \cdot x_i\)
Mistake on negative: \(w_{t+1} \leftarrow w_t - r \cdot x_i\)

\(r\) is the learning rate, a small positive number less than 1

Update only on error. A mistake-driven algorithm

This is the simplest version. We will see more robust versions at the end
The Perceptron algorithm

**Input**: A sequence of training examples \((x_1, y_1), (x_2, y_2), \cdots\)
where all \(x_i \in \mathbb{R}^n\), \(y_i \in \{-1, 1\}\)

- Initialize \(w_0 = 0 \in \mathbb{R}^n\)
- For each training example \((x_i, y_i)\):
  - Predict \(y' = \text{sgn}(w_t^T x_i)\)
  - If \(y_i \neq y'\):
    - Update \(w_{t+1} \leftarrow w_t + r y_i x_i\)
- Return final weight vector

Mistake on positive: \(w_{t+1} \leftarrow w_t + r x_i\)
Mistake on negative: \(w_{t+1} \leftarrow w_t - r x_i\)

\(r\) is the learning rate, a small positive number less than 1

Update only on error. A mistake-driven algorithm

This is the simplest version. We will see more robust versions at the end

Mistake can be written as \(y_i w_t^T x_i \leq 0\)
Intuition behind the update

Suppose we have made a mistake on a positive example
That is, \( y = +1 \) and \( \mathbf{w}_t^T \mathbf{x} < 0 \)

Call the new weight vector \( \mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x} \) (say \( r = 1 \))

The new dot product will be
\[
\mathbf{w}_{t+1}^T \mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T \mathbf{x} = \mathbf{w}_t^T \mathbf{x} + \mathbf{x}^T \mathbf{x} > \mathbf{w}_t^T \mathbf{x}
\]

For a positive example, the Perceptron update will increase the score assigned to the same input

Similar reasoning for negative examples

Mistake on positive: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r \mathbf{x}_i \)
Mistake on negative: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r \mathbf{x}_i \)
Geometry of the perceptron update

Mistake on positive: $w_{t+1} \leftarrow w_t + r \cdot x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - r \cdot x_i$
Geometry of the perceptron update

\[ w_{t+1} \leftarrow w_t + r x_i \]

Mistake on positive:

\[ w_{t+1} \leftarrow w_t - r x_i \]

Mistake on negative:
Geometry of the perceptron update

For a mistake on a positive example

\[
\begin{align*}
\text{Mistake on positive: } & \quad w_{t+1} \leftarrow w_t + r x_i \\
\text{Mistake on negative: } & \quad w_{t+1} \leftarrow w_t - r x_i
\end{align*}
\]
Geometry of the perceptron update

Predict

\[ \mathbf{w}_{\text{old}} \]

\[(x, +1)\]

Update

\[ \mathbf{w} \leftarrow \mathbf{w} + \mathbf{x} \]

\[(x, +1)\]

For a mistake on a **positive** example

Mistake on positive: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{r} \mathbf{x}_i \)
Mistake on negative: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{r} \mathbf{x}_i \)
Geometry of the perceptron update

For a mistake on a positive example

\[ \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r \mathbf{x}_i \]

\[ \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r \mathbf{x}_i \]
Geometry of the perceptron update

For a mistake on a positive example

Mistake on positive: $w_{t+1} \leftarrow w_t + r x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - r x_i$
Geometry of the perceptron update

For a mistake on a **positive** example

\[ \mathbf{w}_{\text{old}} \]

\[ \rightarrow \]

\[ \mathbf{x} \]

\[ \mathbf{w} \leftarrow \mathbf{w} + \mathbf{x} \]

\[ \mathbf{w}_{\text{new}} \]

Mistake on positive: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r \mathbf{x}_i \)

Mistake on negative: \( \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r \mathbf{x}_i \)
Geometry of the perceptron update

$\mathbf{w}_{\text{old}}$
Geometry of the perceptron update

Predict

For a mistake on a negative example
Geometry of the perceptron update

For a mistake on a negative example
Geometry of the perceptron update

For a mistake on a negative example
Geometry of the perceptron update

For a mistake on a **negative** example
Geometry of the perceptron update

For a mistake on a negative example
Perceptron Learnability

• Obviously Perceptron cannot learn what it cannot represent
  – Only linearly separable functions

• Minsky and Papert (1969) wrote an influential book demonstrating Perceptron’s representational limitations
  – Parity functions can’t be learned (XOR)
    • But we already know that XOR is not linearly separable
  – Feature transformation trick
What you need to know

• The Perceptron algorithm

• The geometry of the update

• What can it represent
Where are we?

- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
Convergence

**Convergence theorem**

- If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.
Convergence

**Convergence theorem**

– If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.

**Cycling theorem**

– If the training data is *not* linearly separable, then the learning algorithm will eventually repeat the same set of weights and enter an infinite loop.
Margin

The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
Margin

- The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

- The **margin of a data set** ($\gamma$) is the maximum margin possible for that dataset using any weight vector.
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\) be a sequence of training examples such that for all \(i\), the feature vector \(x_i \in \mathbb{R}^n\), \(\|x_i\| \leq R\) and the label \(y_i \in \{-1, +1\}\).
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \} be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( ||x_i|| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( ||u|| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( \|x_i\| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( \|u\| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

Then, the perceptron algorithm will make at most \( (R/\gamma)^2 \) mistakes on the training sequence.
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( \|x_i\| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( \|u\| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

Then, the perceptron algorithm will make at most \( (R/\gamma)^2 \) mistakes on the training sequence.

We can always find such an \( R \). Just look for the farthest data point from the origin.
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \} be a sequence of training examples such that for all i, the feature vector \( x_i \in \mathbb{R}^n \), \( ||x_i|| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( ||u|| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

Then, the perceptron algorithm will make at most \( (R/\gamma)^2 \) mistakes on the training sequence.

The data and \( u \) have a margin \( \gamma \). Importantly, the data is \textit{separable}. \( \gamma \) is the complexity parameter that defines the separability of data.
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( \|x_i\| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( \|u\| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

Then, the perceptron algorithm will make at most \( (\frac{R}{\gamma})^2 \) mistakes on the training sequence.

If \( u \) hadn’t been a unit vector, then we could scale \( \gamma \) in the mistake bound. This will change the final mistake bound to \( (\|u\| R / \gamma)^2 \)
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( \|x_i\| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose we have a binary classification dataset with \( n \) dimensional inputs. Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e. \( \|u\| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

If the data is separable,...

Then, the perceptron algorithm will make at most \( (R/\gamma)^2 \) mistakes on the training sequence.

...then the Perceptron algorithm will find a separating hyperplane after making a finite number of mistakes.
Proof (preliminaries)

The setting

• Initial weight vector $\mathbf{w}$ is all zeros

• Learning rate $= 1$
  – Effectively scales inputs, but does not change the behavior

• All training examples are contained in a ball of size $R$
  – $||\mathbf{x}_i|| \leq R$

• The training data is separable by margin $\gamma$ using a unit vector $\mathbf{u}$
  – $y_i (\mathbf{u}^T \mathbf{x}_i) \geq \gamma$

• Receive an input $(\mathbf{x}_i, y_i)$
  • if $\text{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
    Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$
Proof (1/3)

1. Claim: After $t$ mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t \gamma$

\[
\mathbf{u}^T \mathbf{w}_{t+1} = \mathbf{u}^T \mathbf{w}_t + y_i \mathbf{u}^T \mathbf{x}_i
\]

- Receive an input $(\mathbf{x}_i, y_i)$
- if $\text{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$
Proof (1/3)

1. Claim: After $t$ mistakes, $u^T w_t \geq t \gamma$

\[
\begin{align*}
    u^T w_{t+1} &= u^T w_t + y_i u^T x_i \\
    \geq& \quad u^T w_t + \gamma
\end{align*}
\]

Because the data is separable by a margin $\gamma$
Proof (1/3)

1. Claim: After $t$ mistakes, $u^T w_t \geq t \gamma$

\[
\begin{align*}
    u^T w_{t+1} &= u^T w_t + y_i u^T x_i \\
    &\geq u^T w_t + \gamma
\end{align*}
\]

Because the data is separable by a margin $\gamma$

Because $w_0 = 0$ (i.e. $u^T w_0 = 0$), straightforward induction gives us $u^T w_t \geq t \gamma$
2. Claim: After $t$ mistakes, $\|w_t\|^2 \leq tR^2$

\[
\|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\
= \|w_t\|^2 + 2y_i (w_t^T x_i) + \|x_i\|^2
\]
2. Claim: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2$$

$$= \|w_t\|^2 + 2y_i (w_t^T x_i) + \|x_i\|^2$$

The weight is updated only when there is a mistake. That is when $y_i w_t^T x_i < 0$.  

$\|x_i\| \leq R$, by definition of $R$

- Receive an input $(x_i, y_i)$
- if $\text{sgn}(w_t^T x_i) \neq y_i$: 
  Update $w_{t+1} \leftarrow w_t + y_i x_i$
2. Claim: After $t$ mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{align*}
\|w_{t+1}\|^2 &= \|w_t + y_i x_i\|^2 \\
&= \|w_t\|^2 + 2y_i(w_t^T x_i) + \|x_i\|^2 \\
&\leq \|w_t\|^2 + R^2
\end{align*}$$

Because $w_0 = 0$ (i.e. $u^T w_0 = 0$), straightforward induction gives us $\|w_t\|^2 \leq tR^2$
Proof (3/3)

What we know:

1. After $t$ mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After $t$ mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$
Proof (3/3)

What we know:

1. After $t$ mistakes, $u^T w_t \geq ty$
2. After $t$ mistakes, $\|w_t\|^2 \leq tR^2$

\[ R\sqrt{t} \geq \|w_t\| \]

From (2)
Proof (3/3)

What we know:

1. After \( t \) mistakes, \( u^T w_t \geq t \gamma \)
2. After \( t \) mistakes, \( \|w_t\|^2 \leq tR^2 \)

\[
R \sqrt{t} \geq \|w_t\| \geq u^T w_t
\]

From (2)

\[
u^T w_t = \|u\| \|w_t\| \cos(\text{angle between them})
\]

But \( \|u\| = 1 \) and cosine is less than 1

So \( u^T w_t \cdot \|w_t\| \)
Proof (3/3)

What we know:

1. After \( t \) mistakes, \( u^T w_t \geq ty \)
2. After \( t \) mistakes, \( ||w_t||^2 \leq tR^2 \)

\[
R \sqrt{t} \geq ||w_t|| \geq u^T w_t
\]

From (2)

\[
u^T w_t = ||u|| \cdot ||w_t|| \cdot \cos(<\text{angle between them}>)
\]

But \( ||u|| = 1 \) and cosine is less than 1

So \( u^T w_t \cdot ||w_t|| \) (Cauchy-Schwarz inequality)
Proof (3/3)

What we know:

1. After $t$ mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After $t$ mistakes, $||\mathbf{w}_t||^2 \leq t\mathbf{R}^2$

\[ R\sqrt{t} \geq ||\mathbf{w}_t|| \geq \mathbf{u}^T \mathbf{w}_t \geq t\gamma \]

From (2) \hspace{2cm} From (1)

\[ \mathbf{u}^T \mathbf{w}_t = ||\mathbf{u}|| \cdot ||\mathbf{w}_t|| \cdot \cos(<\text{angle between them}>) \]

But $||\mathbf{u}|| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq ||\mathbf{w}_t||$
Proof (3/3)

What we know:

1. After $t$ mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After $t$ mistakes, $||\mathbf{w}_t||^2 \leq tR^2$

$$R\sqrt{t} \geq ||\mathbf{w}_t|| \geq \mathbf{u}^T \mathbf{w}_t \geq t\gamma$$

Number of mistakes $t \leq \frac{R^2}{\gamma^2}$
Proof (3/3)

What we know:

1. After $t$ mistakes, $u^T w_t \geq t\gamma$
2. After $t$ mistakes, $\|w_t\|^2 \leq tR^2$

$R\sqrt{t} \geq \|w_t\| \geq u^T w_t \geq t\gamma$

Number of mistakes $t \leq \frac{R^2}{\gamma^2}$

Bounds the total number of mistakes!
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \) be a sequence of training examples such that for all \( i \), the feature vector \( x_i \in \mathbb{R}^n \), \( \|x_i\| \leq R \) and the label \( y_i \in \{-1, +1\} \).

Suppose there exists a unit vector \( u \in \mathbb{R}^n \) (i.e \( \|u\| = 1 \)) such that for some \( \gamma \in \mathbb{R} \) and \( \gamma > 0 \) we have \( y_i (u^T x_i) \geq \gamma \).

Then, the perceptron algorithm will make at most \((R/ \gamma)^2\) mistakes on the training sequence.
Mistake Bound Theorem [Novikoff 1962, Block 1962]
Mistake Bound Theorem [Novikoff 1962, Block 1962]
Mistake Bound Theorem [Novikoff 1962, Block 1962]
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Number of mistakes \[ \leq \frac{R^2}{\gamma^2} \]
The Perceptron Mistake bound

\[ \text{Number of mistakes} \leq \frac{R^2}{\gamma^2} \]

- Exercises:
  - How many mistakes will the Perceptron algorithm make for disjunctions with \( n \) attributes?
    - What are \( R \) and \( \gamma \)?
  - How many mistakes will the Perceptron algorithm make for \( k \)-disjunctions with \( n \) attributes?
What you need to know

• What is the perceptron mistake bound?

• How to prove it
Where are we?

• The Perceptron Algorithm

• Perceptron Mistake Bound

• Variants of Perceptron
Practical use of the Perceptron algorithm

1. Using the Perceptron algorithm with a finite dataset

2. Margin Perceptron

3. Voting and Averaging
1. The “standard” algorithm

Given a training set $D = \{(x_i, y_i)\}$, $x_i \in \mathbb{R}^n$, $y_i \in \{-1,1\}$

1. Initialize $w = 0 \in \mathbb{R}^n$

2. For epoch = 1 … T:
   1. Shuffle the data
   2. For each training example $(x_i, y_i) \in D$:
      - If $y_i w^T x_i \leq 0$, update $w \leftarrow w + r y_i x_i$

3. Return $w$

Prediction: $\text{sgn}(w^T x)$
1. The “standard” algorithm

Given a training set \( D = \{(x_i, y_i)\}, x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}\)

1. Initialize \( w = 0 \in \mathbb{R}^n \)

2. For epoch = 1 \ldots T:
   1. Shuffle the data
   2. For each training example \( (x_i, y_i) \in D \):
      - If \( y_i w^T x_i \leq 0 \), update \( w \leftarrow w + r y_i x_i \)

3. Return \( w \)

Prediction: \( \text{sgn}(w^T x) \)

\( T \) is a hyper-parameter to the algorithm

Another way of writing that there is an error
2. Margin Perceptron

Which hyperplane is better?
2. Margin Perceptron

Which hyperplane is better?
2. Margin Perceptron

Which hyperplane is better?

The farther from the data points, the less chance to make wrong prediction.
2. Margin Perceptron

- Perceptron makes updates only when the prediction is incorrect
  \[ y_i \mathbf{w}^\top \mathbf{x}_i < 0 \]

- What if the prediction is close to being incorrect? That is, Pick a positive \( \eta \) and update when
  \[ \frac{y_i \mathbf{w}^\top \mathbf{x}_i}{\|\mathbf{w}\|} < \eta \]

- Can generalize better, but need to choose
  - Why is this a good idea?
2. Margin Perceptron

- Perceptron makes updates only when the prediction is incorrect
  \[ y_i \mathbf{w}^\top \mathbf{x}_i < 0 \]

- What if the prediction is close to being incorrect? That is, Pick a positive \( \eta \) and update when
  \[ \frac{y_i \mathbf{w}^\top \mathbf{x}_i}{\|\mathbf{w}\|} < \eta \]

- Can generalize better, but need to choose
  - Why is this a good idea? Intentionally set a large margin
3. Voting and Averaging

What if data is not linearly separable?

Finding a hyperplane with minimum mistakes is NP hard
Voted Perceptron

Given a training set $D = \{(x_i, y_i)\}$, $x_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$

1. Initialize $w = 0 \in \mathbb{R}^n$ and $a = 0 \in \mathbb{R}^n$

2. For epoch = 1 ... $T$:
   - For each training example $(x_i, y_i) \in D$:
     - If $y_i w^T x_i \leq 0$
       - update $w_{m+1} \leftarrow w_m + r \cdot y_i \cdot x_i$
       - $m = m+1$
       - $C_m = 1$
     - Else
       - $C_m = C_m + 1$

3. Return $(w_1, c_1)$, $(w_2, c_2)$, ..., $(w_k, c_k)$

Prediction: $\text{sgn} \left( \sum_{i=1}^{k} c_i \cdot \text{sgn}(w_i^T x) \right)$
Averaged Perceptron

Given a training set \( D = \{(x_i, y_i)\}, x_i \in \mathbb{R}^n, y_i \in \{-1, 1\} \)

1. Initialize \( w = 0 \in \mathbb{R}^n \) and \( a = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   - For each training example \((x_i, y_i) \in D:\)
     - If \( y_i \, w^T \, x_i \leq 0 \)
       - update \( w \leftarrow w + r \, y_i \, x_i \)
     - \( a \leftarrow a + w \)

3. Return \( a \)

**Prediction:** \( \text{sgn}(a^T x) = \text{sgn}(\sum_{i=1}^{k} c_i \, w_i^T \, x) \)
Averaged Perceptron

Given a training set $D = \{(x_i, y_i)\}, x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}$

1. Initialize $w = 0 \in \mathbb{R}^n$ and $a = 0 \in \mathbb{R}^n$

2. For epoch $= 1 \ldots T$:
   - For each training example $(x_i, y_i) \in D$:
     • If $y_i \cdot w^T x_i \leq 0$
       - update $w \leftarrow w + r \cdot y_i \cdot x_i$
     • $a \leftarrow a + w$

3. Return $a$

**Prediction:** $\text{sgn}(a^T x) = \text{sgn}(\sum_{i=1}^{k} c_i w_i^T x)$
Averaged Perceptron

Given a training set \( D = \{(x_i, y_i)\}, x_i \in \mathbb{R}^n, y_i \in \{-1,1\} \)

1. Initialize \( w = 0 \in \mathbb{R}^n \) and \( a = 0 \in \mathbb{R}^n \)

2. For epoch = 1 ... T:
   - For each training example \((x_i, y_i) \in D:\)
     • If \( y_i w^T x_i \leq 0 \)
       – update \( w \leftarrow w + r y_i x_i \)
     • \( a \leftarrow a + w \)

3. Return \( a \)

Prediction: \( \text{sgn}(a^T x) = \text{sgn}(\sum_{i=1}^{k} c_i w_i^T x) \)
Question: What is the difference?

Voted: \[ \text{sgn}\left( \sum_{i=1}^{k} c_i \cdot \text{sgn}(w_i^\top x) \right) \]

Averaged: \[ \text{sgn}\left( \sum_{i=1}^{k} c_i w_i^\top x \right) \]

\[ w_1^\top x = s_1, \ w_2^\top x = s_2, \ w_3^\top x = s_3 \quad c_1 = c_2 = c_3 = 1 \]

Averaged: \[ s_1 + s_2 + s_3 \geq 0 \]

Voted: Any two are positive
Summary: Perceptron

- Online learning algorithm, very widely used, easy to implement
- Additive updates to weights
- Geometric interpretation
- Mistake bound
- Practical variants abound
- You should be able to implement the Perceptron algorithm