# COMP 761: Lecture 36 – Neural Networks I

David Rolnick

November 27, 2020

# Problem

Suppose you are given datapoints $x^1, x^2, \ldots, x^n \in \mathbb{R}^d$ and binary labels $y^1, y^2, \ldots, y^n \in \{+1, -1\}$ such that the data is *linearly separable*, that is, for some weight vector $w$:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for every } k.$$

Describe an algorithm to find such a weight vector $w$.

*(Please don't post your ideas in the chat just yet, we'll discuss the problem soon in class.)*

# Course Announcements

# Course Announcements

- On the problem set, totally fine to take as a given that the problem of determining whether a Hamiltonian cycle exists in a graph in NP-complete.

# Fun sidenote (not about neural nets)

# Fun sidenote (not about neural nets)

- This came up in office hours today.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- Is 2-SAT NP-complete?

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).
- That immediately gives us some other variables that are TRUE / FALSE (e.g. $x_4$ above) since every clause has 2 variables.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).
- That immediately gives us some other variables that are TRUE / FALSE (e.g. $x_4$ above) since every clause has 2 variables.
- Repeat the preceding steps with the other variables as we did with $x_1$.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).
- That immediately gives us some other variables that are TRUE / FALSE (e.g. $x_4$ above) since every clause has 2 variables.
- Repeat the preceding steps with the other variables as we did with $x_1$.
- Eventually left with either (i) the whole formula satisfied, or (ii) a few remaining clauses.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).
- That immediately gives us some other variables that are TRUE / FALSE (e.g. $x_4$ above) since every clause has 2 variables.
- Repeat the preceding steps with the other variables as we did with $x_1$.
- Eventually left with either (i) the whole formula satisfied, or (ii) a few remaining clauses.
- If (i), we're done. If (ii), we've reduced the problem to just those clauses.

# Fun sidenote (not about neural nets)

- 2-SAT is like 3-SAT except the clauses in the formula have 2 variables each, not 3, e.g.:

$$\phi = (x_1 \vee x_2) \wedge (-x_2 \vee -x_3) \wedge (-x_1 \vee x_4).$$

- 2-SAT is actually solvable in polynomial time! Here's how:
- Pick a clause $C$, without loss of generality suppose it's $(x_1 \vee x_2)$.
- Case 1: $x_1$ is TRUE.
- Eliminate all clauses with $x_1$ (since they are true), and remove $-x_1$ from all clauses where it occurs (since $-x_1 = \mathit{FALSE}$).
- That immediately gives us some other variables that are TRUE / FALSE (e.g. $x_4$ above) since every clause has 2 variables.
- Repeat the preceding steps with the other variables as we did with $x_1$.
- Eventually left with either (i) the whole formula satisfied, or (ii) a few remaining clauses.
- If (i), we're done. If (ii), we've reduced the problem to just those clauses.
- Case 2 ($x_2$ is TRUE) is similar.

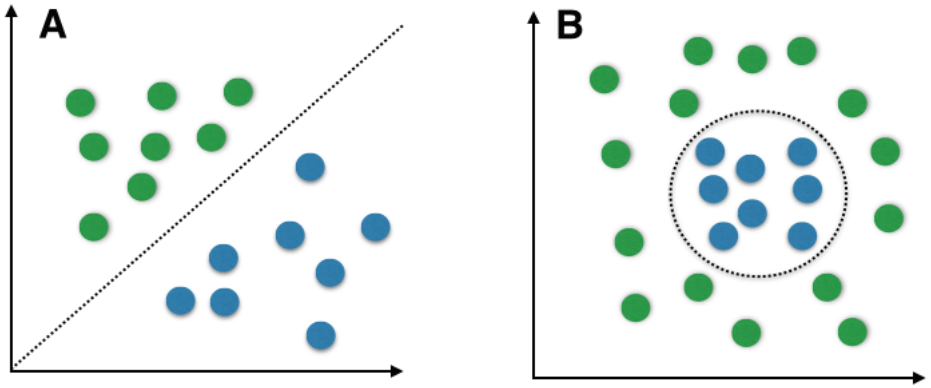
# Classification problems

# Classification problems

- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.

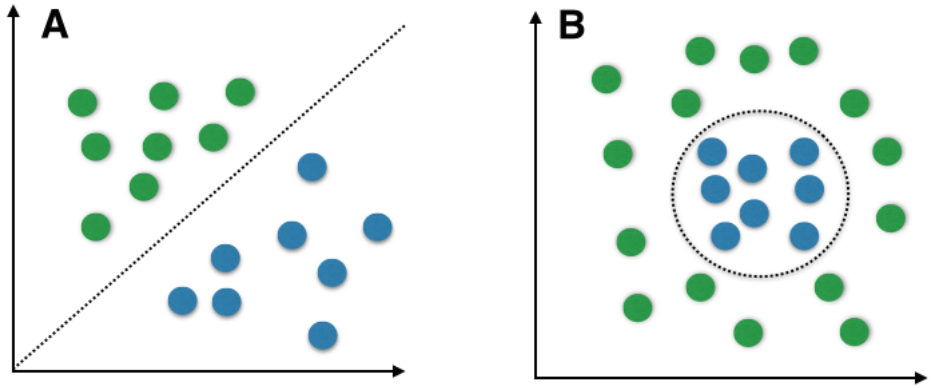
# Classification problems

- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:



# Classification problems

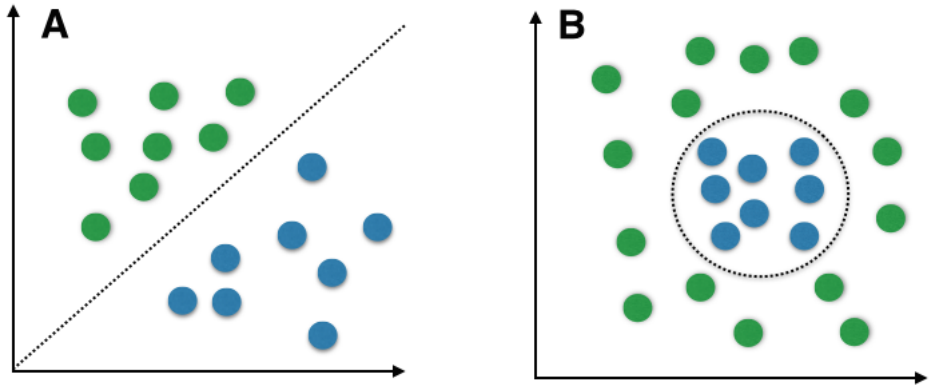
- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:




- Binary classification comes up in answering yes/no questions (e.g. “does this image show a tumor or not?”).

# Classification problems

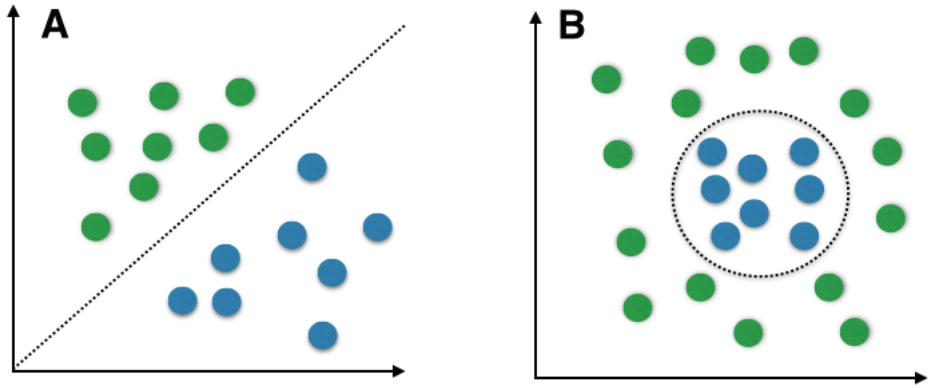
- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:




- Binary classification comes up in answering yes/no questions (e.g. “does this image show a tumor or not?”).
- More general classification, labeling datapoints like images (“dog”, “cat”, “book”, etc).

# Classification problems

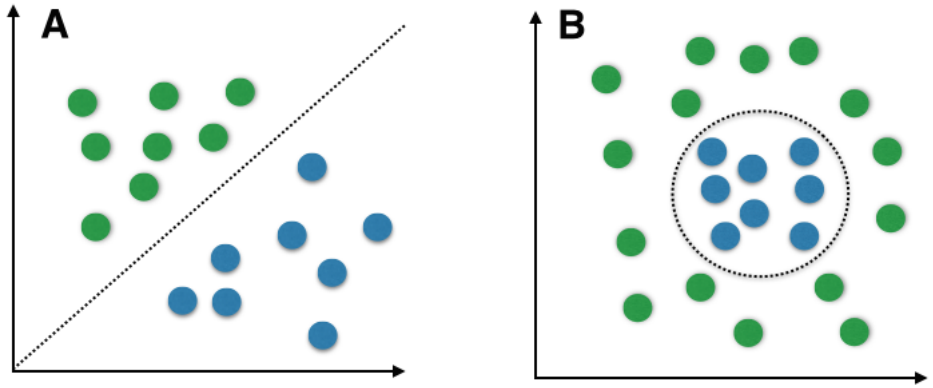
- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:




- Binary classification comes up in answering yes/no questions (e.g. “does this image show a tumor or not?”).
- More general classification, labeling datapoints like images (“dog”, “cat”, “book”, etc).
- Typical (supervised learning) paradigm for classification problem: See examples: $x^1, x^2, \ldots, x^n$, where each $x^k$ is a vector in $\mathbb{R}^d$.

# Classification problems

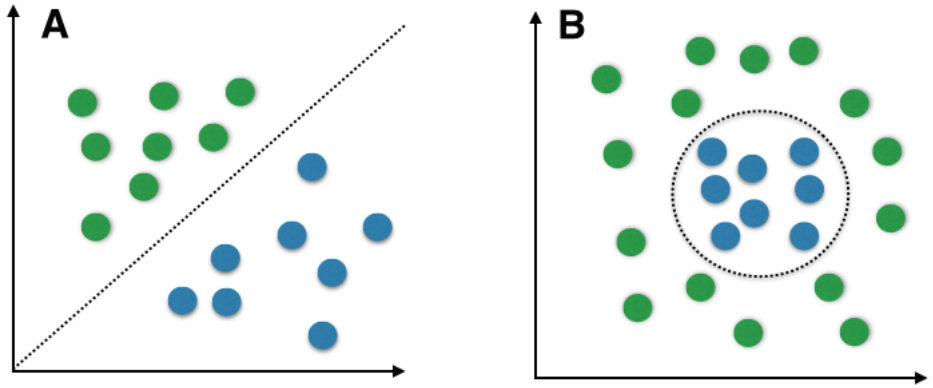
- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:




- Binary classification comes up in answering yes/no questions (e.g. “does this image show a tumor or not?”).
- More general classification, labeling datapoints like images (“dog”, “cat”, “book”, etc).
- Typical (supervised learning) paradigm for classification problem: See examples: $x^1, x^2, \ldots, x^n$, where each $x^k$ is a vector in $\mathbb{R}^d$.
- Given labels $y^1, y^2, \ldots, y^n \in S$, where $y^k = y(x^k)$.

# Classification problems

- A *classification problem* involves taking input data $x \in \mathbb{R}^d$ and inferring a class label $y(x) \in S$ from a set $S$ with $m$ elements.
- Examples of 2-label (binary) classification, i.e. $m = 2$:




- Binary classification comes up in answering yes/no questions (e.g. “does this image show a tumor or not?”).
- More general classification, labeling datapoints like images (“dog”, “cat”, “book”, etc).
- Typical (supervised learning) paradigm for classification problem: See examples: $x^1, x^2, \ldots, x^n$, where each $x^k$ is a vector in $\mathbb{R}^d$.
- Given labels $y^1, y^2, \ldots, y^n \in S$, where $y^k = y(x^k)$.
- Need to come up with a function $f(x) \approx y(x)$ to predict new labels.

# Perceptrons

# Perceptrons

- In some binary classification tasks, the data will be *linearly separable*, i.e. the labels $y(x) \in \{+1, -1\}$ satisfy:

$$y(x) = \text{sign}(w \cdot x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i\right),$$

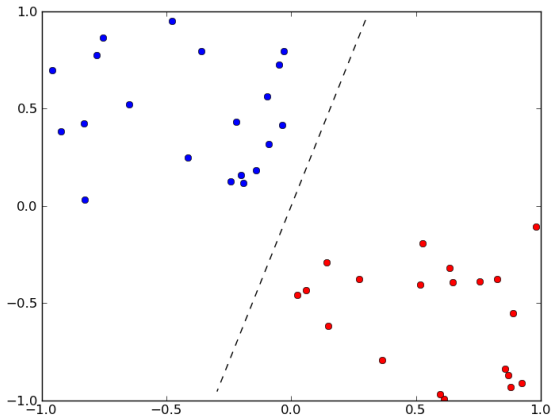
for some weights $w_i$, where sign$(\cdot)$ is the sign function (either +1 or -1).

# Perceptrons

- In some binary classification tasks, the data will be *linearly separable*, i.e. the labels $y(x) \in \{+1, -1\}$ satisfy:

$$y(x) = \text{sign}(w \cdot x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i\right),$$

  for some weights $w_i$, where sign$(\cdot)$ is the sign function (either +1 or -1).
- In a *perceptron*, we assume the data are linearly separable and learn a function $w \cdot x$ that separates the classes.

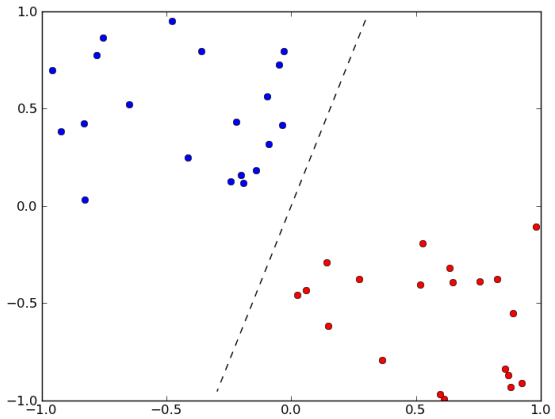


# Perceptrons

- In some binary classification tasks, the data will be *linearly separable*, i.e. the labels $y(x) \in \{+1, -1\}$ satisfy:

$$y(x) = \text{sign}(w \cdot x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i\right),$$

  for some weights $w_i$, where sign$(\cdot)$ is the sign function (either +1 or -1).
- In a *perceptron*, we assume the data are linearly separable and learn a function $w \cdot x$ that separates the classes.



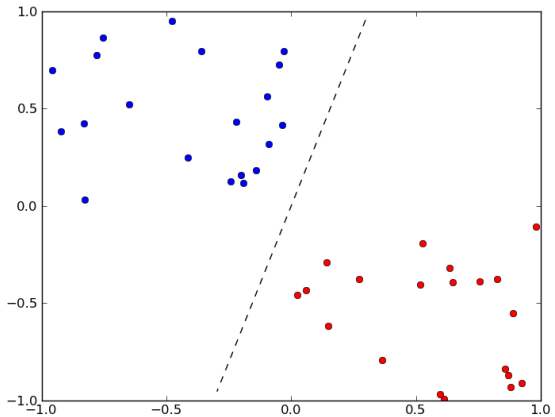

- We learn a function $w \cdot x$ from observed data $x^1, x^2, \ldots, x^n$ and labels $y^1, y^2, \ldots, y^n \in \{+1, -1\}$.

# Perceptrons

- In some binary classification tasks, the data will be *linearly separable*, i.e. the labels $y(x) \in \{+1, -1\}$ satisfy:

$$y(x) = \text{sign}(w \cdot x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i\right),$$

  for some weights $w_i$, where sign$(\cdot)$ is the sign function (either +1 or -1).
- In a *perceptron*, we assume the data are linearly separable and learn a function $w \cdot x$ that separates the classes.




- We learn a function $w \cdot x$ from observed data $x^1, x^2, \ldots, x^n$ and labels $y^1, y^2, \ldots, y^n \in \{+1, -1\}$.
- Then we can predict the class of a new data point $x$.

# The perceptron learning rule

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :
  - Suppose we have $w^j$.

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :
  - Suppose we have $w^j$.
  - Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :
  - Suppose we have $w^j$.
  - Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
  - Define the new weight vector

$$w^{j+1} := w^j + y^k x^k$$

  (note that $y^j \in \{+1, -1\}$ and $x^j \in \mathbb{R}^d$).

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :
  - Suppose we have $w^j$.
  - Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
  - Define the new weight vector

$$w^{j+1} := w^j + y^k x^k$$

    (note that $y^j \in \{+1, -1\}$ and $x^j \in \mathbb{R}^d$).
  - Repeat until the new weight vector linearly separates the data.

# The perceptron learning rule

- Our goal therefore is to learn a weight vector $w$ such that:

$$y^k = \text{sign}(w \cdot x^k) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i^k\right), \text{ for } k = 1, \ldots, n,$$

  i.e. all the datapoints $x^k$ are assigned the correct labels $y^k \in \{+1, -1\}$.
- Our algorithm for doing this will be to start with some candidate weight vector $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find a $w$ that works, :
  - Suppose we have $w^j$.
  - Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
  - Define the new weight vector

$$w^{j+1} := w^j + y^k x^k$$

    (note that $y^j \in \{+1, -1\}$ and $x^j \in \mathbb{R}^d$).
  - Repeat until the new weight vector linearly separates the data.
- Let's see why this works.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

$$w^* \cdot w^{j+1} = w^* \cdot (w^j + y^k x^k)$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

$$w^* \cdot w^{j+1} = w^* \cdot (w^j + y^k x^k) = w^* \cdot w^j + y^k(w^* \cdot x^k)$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

$$w^* \cdot w^{j+1} = w^* \cdot (w^j + y^k x^k) = w^* \cdot w^j + y^k(w^* \cdot x^k) > w^* \cdot w^j + \gamma,$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

$$w^* \cdot w^{j+1} = w^* \cdot (w^j + y^k x^k) = w^* \cdot w^j + y^k(w^* \cdot x^k) > w^* \cdot w^j + \gamma,$$

- Applying $w^* \cdot w^{j+1} > w^* \cdot w^j + \gamma$ recursively, we get

$$w^* \cdot w^{j+1} > w^* \cdot w^1 + j\gamma = j\gamma, \text{ since } w^1 = 0.$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

$$w^* \cdot w^{j+1} = w^* \cdot (w^j + y^k x^k) = w^* \cdot w^j + y^k(w^* \cdot x^k) > w^* \cdot w^j + \gamma,$$

- Applying $w^* \cdot w^{j+1} > w^* \cdot w^j + \gamma$ recursively, we get

$$w^* \cdot w^{j+1} > w^* \cdot w^1 + j\gamma = j\gamma, \text{ since } w^1 = 0.$$

- Since $w^*$ is a unit vector, we have

$$j\gamma < w^* \cdot w^{j+1} \leq ||w^*|| \cdot ||w^{j+1}|| = ||w^{j+1}||, \text{ proving the claim.}$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$||w^{j+1}||^2 = ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k)$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$\begin{aligned} ||w^{j+1}||^2 &= ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k) \\ &= ||w^j||^2 + ||y^k x^k||^2 + 2y^k(w^j \cdot x^k) \end{aligned}$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$\begin{aligned} ||w^{j+1}||^2 &= ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k) \\ &= ||w^j||^2 + ||y^k x^k||^2 + 2y^k(w^j \cdot x^k) \\ &\leq ||w^j||^2 + ||y^k x^k||^2 \end{aligned}$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$\begin{aligned} ||w^{j+1}||^2 &= ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k) \\ &= ||w^j||^2 + ||y^k x^k||^2 + 2y^k(w^j \cdot x^k) \\ &\leq ||w^j||^2 + ||y^k x^k||^2 \leq ||w^j||^2 + ||x^k||^2 \end{aligned}$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$\begin{aligned} ||w^{j+1}||^2 &= ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k) \\ &= ||w^j||^2 + ||y^k x^k||^2 + 2y^k(w^j \cdot x^k) \\ &\leq ||w^j||^2 + ||y^k x^k||^2 \leq ||w^j||^2 + ||x^k||^2 \leq ||w^j||^2 + R^2, \end{aligned}$$

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

$$\begin{aligned} ||w^{j+1}||^2 &= ||w^j + y^k x^k||^2 = (w^j + y^k x^k) \cdot (w^j + y^k x^k) \\ &= ||w^j||^2 + ||y^k x^k||^2 + 2y^k(w^j \cdot x^k) \\ &\leq ||w^j||^2 + ||y^k x^k||^2 \leq ||w^j||^2 + ||x^k||^2 \leq ||w^j||^2 + R^2, \end{aligned}$$

which can be applied recursively to prove the claim.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

- Combining these we have $(j\gamma)^2 \leq jR^2$.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

- Combining these we have $(j\gamma)^2 \leq jR^2$.
- Therefore, $j \leq (R/\gamma)^2$, so must terminate after that number of iterations.

# The perceptron learning rule

Start with $w^1$ = zeros and construct $w^2, w^3, \ldots$ until we find one that works:

- Suppose we have $w^j$.
- Find some $x^k$ such that $y^k \neq \text{sign}(w^j \cdot x^k)$.
- Define the new weight vector $w^{j+1} := w^j + y^k x^k$.

We will need two assumptions to prove that this algorithm works:

- The data is linearly separable – i.e., there is a unit weight vector $w^*$ and a *margin* $\gamma > 0$ such that for every $k$, we have $w^* \cdot x^k > \gamma$ if $y^k = +1$ and $w^* \cdot x^k < -\gamma$ if $y^k = -1$. We can summarize this as $y^k(w^* \cdot x^k) > \gamma$.
- The data is bounded, i.e. $||x^k|| \leq R$ for some $R$.

**Claim 1.** We have $||w^{j+1}|| > j\gamma$.

**Claim 2.** We have $||w^{j+1}||^2 \leq jR^2$.

- Combining these we have $(j\gamma)^2 \leq jR^2$.
- Therefore, $j \leq (R/\gamma)^2$, so must terminate after that number of iterations.
- Note: This algorithm only works if the data is exactly linearly separable.