University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chenhao Tan

Lecture 5: Perceptron II

Slides adapted from Chris Ketelsen, Jordan Boyd-Graber, and Noah Smith

Administrivia

- HW 1 due on Friday
- Minor change to HW1

Learning Objectives

- Understand the perceptron algorithm

Perceptron algorithm

- Vanilla perceptron algorithm
- Interpretation of weight values
- Convergence of the perceptron algorithm

Perceptron learning algorithm

Data: $D = \langle (\boldsymbol{x}_n, y_n) \rangle_{n=1}^{N}$, number of epochs E

Result: weights $\boldsymbol{w}$ and bias b

initialize: $\boldsymbol{w} = \boldsymbol{0}$ and $b = 0$;

for $e \in \{1, \ldots, E\}$ **do**

  for $n \in \{1, \ldots, N\}$, *in random order* **do**

    # predict

    $a = (\boldsymbol{w} \cdot \boldsymbol{x}_n + b)$;

    if $a y_n \leq 0$ **then**

      # update

      $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n \cdot \boldsymbol{x}_n$;

      $b \leftarrow b + y_n$;

    end

  end

end

$y \in \{-1, +1\}$

$$\hat{y} = \begin{cases} 1 & wx+b > 0 \\ -1 & wx+b \leq 0 \end{cases}$$

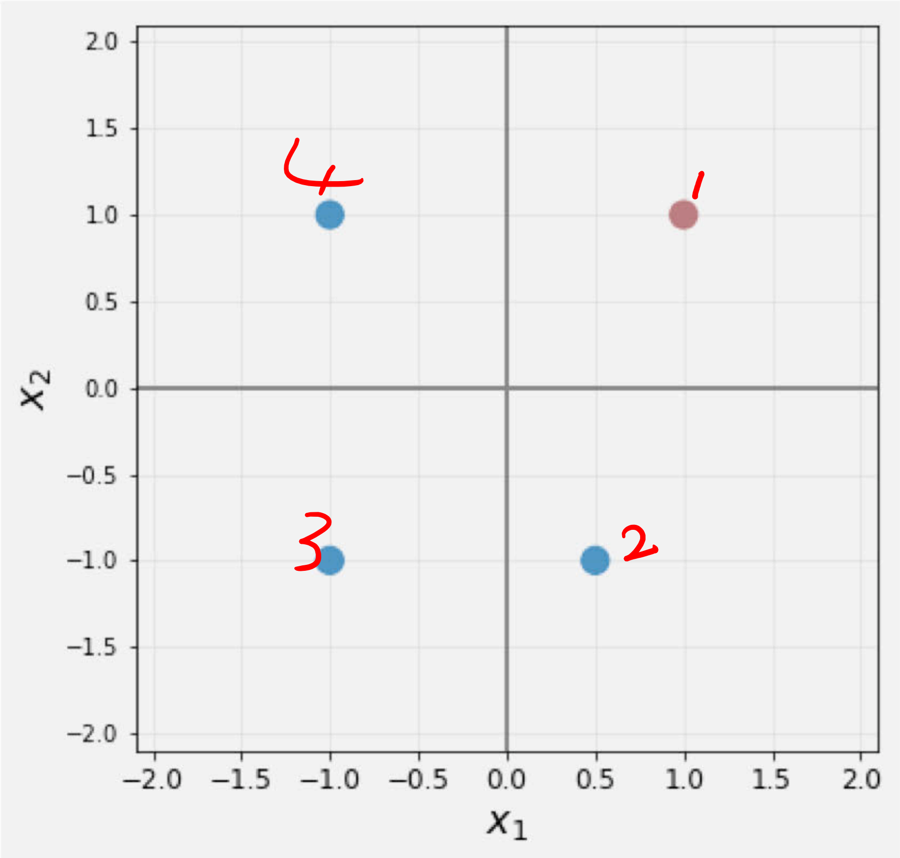
Example

- Start with $\boldsymbol{w} = [1, 0], b = 0$
- Process points in order (red for $+1$, blue for -1):

$$(1, 1), (0.5, -1), (-1. -1), (-1, 1)$$

$a = w \cdot x + b = 1, \quad y = 1$

$ay = 1 > 0$



Example

- Start with $\boldsymbol{w} = [1, 0], b = 0$
- Process points in order (red for $+1$, blue for -1):

$$(1, 1), \underline{(0.5, -1)}, (-1. -1), (-1, 1)$$

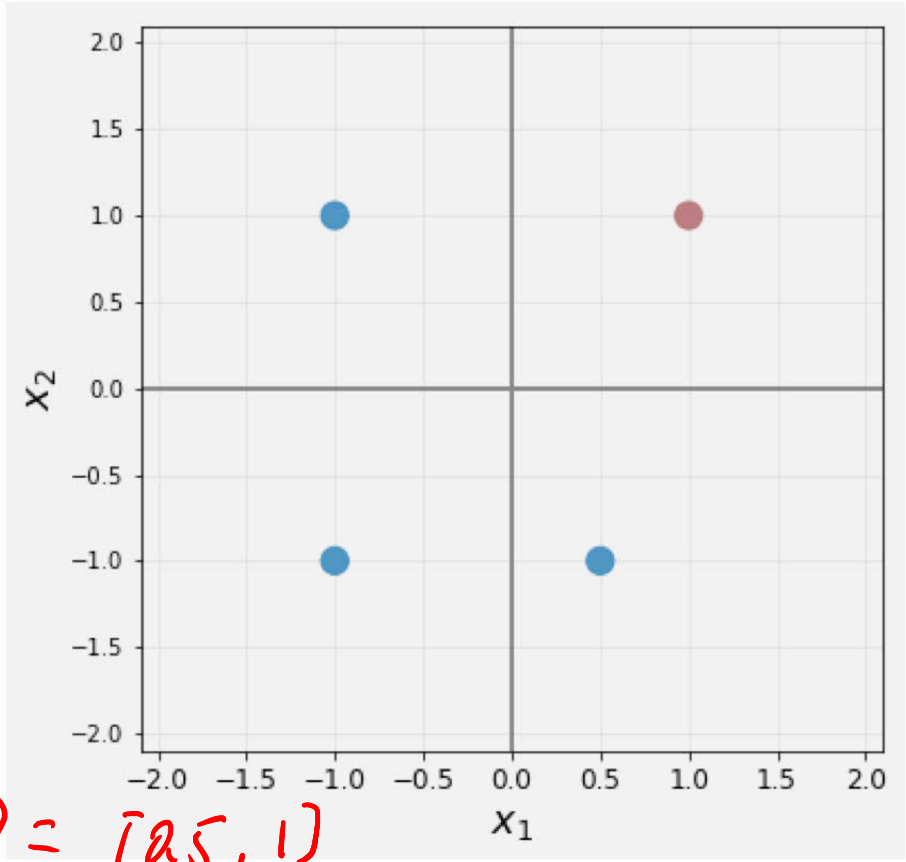
- $(1, 1) : a = 1, y = 1$: no update

$a = wx + b = 0.5 \quad y = -1$

$ay = -0.5$

$w \leftarrow [1, 0] + (-1) \cdot (0.5, -1) = [0.5, 1]$

$b \leftarrow 0 + (-1) = -1$



Example

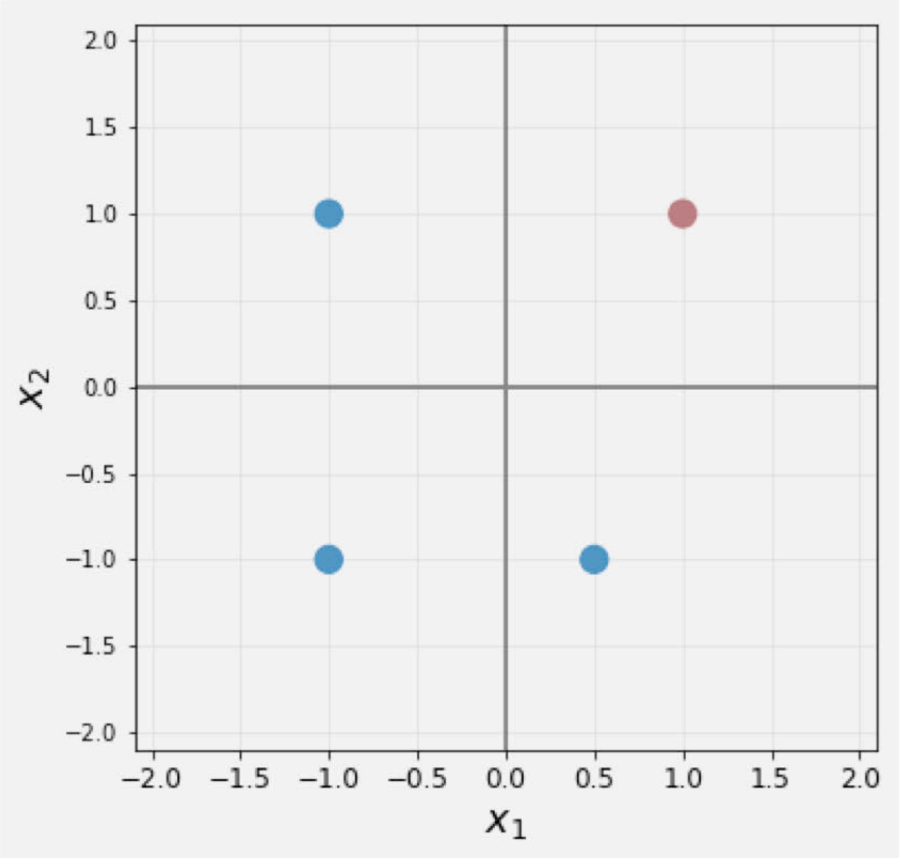
- Start with $w = [1, 0], b = 0$
- Process points in order (red for $+1$, blue for -1):

$$(1,1), (0.5, -1), (-1. -1), (-1, 1)$$

- $(1,1) : a = 1, y = 1$: no update
- $(0.5, -1) : a = 0.5, y = -1$: $w = [0.5, 1], b = -1$

$a = w \cdot x + b = -1.5 - 1 = -2.5$ $\quad y = -1$

$ay > 0$

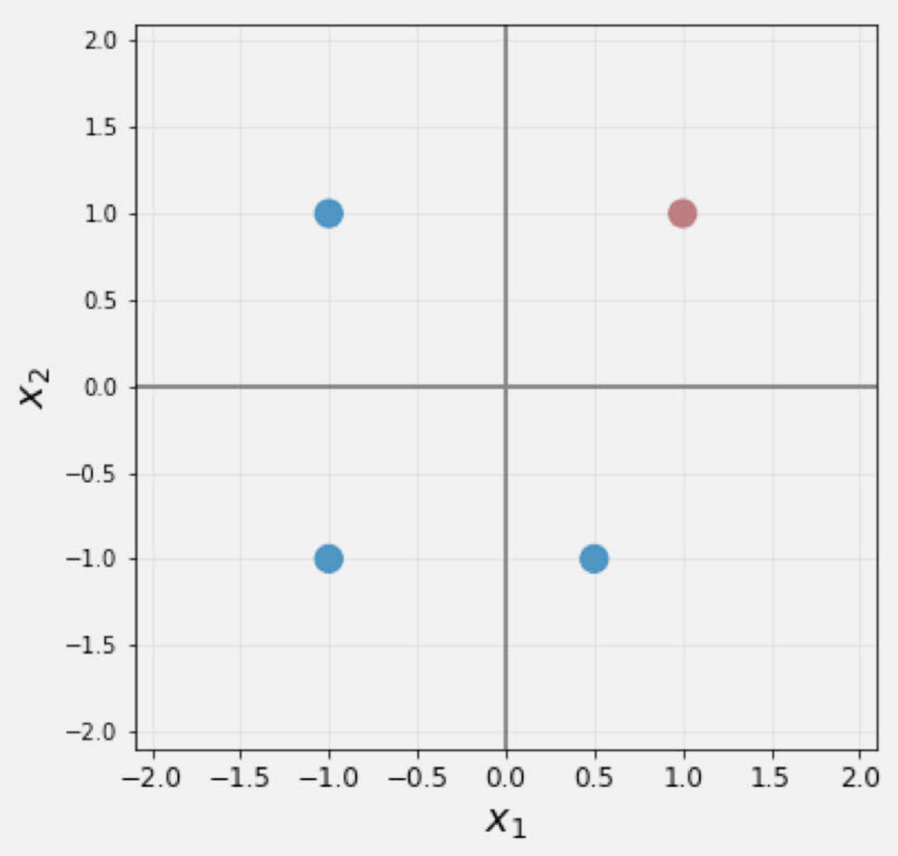


Example

- Start with $\boldsymbol{w} = [1, 0], b = 0$
- Process points in order (red for $+1$, blue for -1):

$$(1, 1), (0.5, -1), (-1. - 1), (-1, 1)$$

- $(1, 1) : a = 1, y = 1$: no update
- $(0.5, -1) : a = 0.5, y = -1$: $w = [0.5, 1], b = -1$
- $(-1, -1) : a = -2.5, y = -1$: no update
- $(-1, 1) : a = -0.5, y = -1$: no update



Why does this algorithm work?

Assume that we have just misclassified a point $(\boldsymbol{x}, y)$, it means

$$\underline{a = \boldsymbol{w} \cdot \boldsymbol{x} + b}, \underline{ay \leq 0}$$

$\sum_{i=1}^{d} w_i x_i$

After the update: $\boldsymbol{w}' = \boldsymbol{w} + y\boldsymbol{x}, b' = b + y$

$ay \leq 0$

$y = 1$ $\quad a > 0$

$a \leq 0 \rightarrow$ wrong

$y = -1$ $\quad a < 0 \rightarrow$ Right

$a \geq 0 \rightarrow$ wrong

$$a' = w'x + b'$$
$$= (w + yx) \cdot x + b + y$$
$$= \underline{w \cdot x} + y(x \cdot x) \underline{+ b} + y$$
$$= a + y(1 + (x \cdot x))$$
$$a'y = ay + y^2(1 + (x \cdot x))$$
$$= ay + \underline{(1 + (x \cdot x))} > ay$$

$x \cdot x \geq 0$

Why does this algorithm work?

Assume that we have just misclassified a point $(\boldsymbol{x}, y)$, it means

$$a = \boldsymbol{w} \cdot \boldsymbol{x} + b, ay \leq 0$$

After the update: $\boldsymbol{w}' = \boldsymbol{w} + y\boldsymbol{x}, b' = b + y$

$$\begin{aligned} a' &= \boldsymbol{w}' \cdot \boldsymbol{x} + b' \\ &= \boldsymbol{w} \cdot \boldsymbol{x} + y||x||^2 + b + y \\ &= a + y||x||^2 + y \end{aligned}$$

Why does this algorithm work?

Assume that we have just misclassified a point $(\boldsymbol{x}, y)$, it means

$$a = \boldsymbol{w} \cdot \boldsymbol{x} + b, ay \leq 0$$

After the update: $\boldsymbol{w}' = \boldsymbol{w} + y\boldsymbol{x}, b' = b + y$

$$\begin{aligned} a' &= \boldsymbol{w}' \cdot \boldsymbol{x} + b' \\ &= \boldsymbol{w} \cdot \boldsymbol{x} + y||x||^2 + b + y \\ &= a + y||x||^2 + y \\ a'y &= ay + ||x||^2 + 1 > ay \end{aligned}$$

Perceptron learning algorithm

Data: $D = \langle(\boldsymbol{x}_n, y_n)\rangle_{n=1}^{N}$, number of epochs E
Result: weights $\boldsymbol{w}$ and bias b
initialize: $\boldsymbol{w} = \mathbf{0}$ and $b = 0$;
for $e \in \{1, \ldots, E\}$ **do**
    for $n \in \{1, \ldots, N\}$, *in random order* **do**
        # predict
        $a = (\boldsymbol{w} \cdot \boldsymbol{x}_n + b)$;
        if $a y_n \leq 0$ **then**
            # update
            $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n \cdot \boldsymbol{x}_n$;
            $b \leftarrow b + y_n$;
        end
    end
end

- why $a y_n \leq 0$ rather than $a y_n < 0$?
- why random order?

Parameters and Hyperparameters

This is the first supervised algorithm we've seen that has **parameters** that are numerical values (w and b).

Parameters and Hyperparameters

This is the first supervised algorithm we've seen that has **parameters** that are numerical values (w and b).

The perceptron learning algorithm's sole hyperparameter is E, the number of epochs (passes over the training data).

Interpretation of Weight Values

What does it mean when . . .

Interpretation of Weight Values

What does it mean when . . .

- $w_{12} = 100$?

Interpretation of Weight Values

What does it mean when …

- $w_{12} = 100$?
- $w_{12} = -1$?

Interpretation of Weight Values

What does it mean when ...

- $w_{12} = 100$?
- $w_{12} = -1$?
- $w_{12} = 0$?

$w_{13} = 100$ $x_3 \in (-0.0001, 0.0001)$

$w_{12} = -1$ $x_2 \in [-100, 100]$

Interpretation of Weight Values

What does it mean when . . .

- $w_{12} = 100$?
- $w_{12} = -1$?
- $w_{12} = 0$?

What if we multiply w by 2?

$wx \quad +b$

$2wx \quad +2b$

Interpretation of Weight Values

In other words, how sensitive is the final classification to changes in individual features?

$$y = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) \begin{cases} +1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b \geq 0 \\ -1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b < 0 \end{cases}$$

Interpretation of Weight Values

In other words, how sensitive is the final classification to changes in individual features?

$$y = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) \begin{cases} +1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b \geq 0 \\ -1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b < 0 \end{cases}$$

$$\frac{\partial \boldsymbol{w} \cdot \boldsymbol{x} + b}{\partial \boldsymbol{x}[k]} = \boldsymbol{w}[k]$$

Interpretation of Weight Values

In other words, how sensitive is the final classification to changes in individual features?

$$y = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) \begin{cases} +1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b \geq 0 \\ -1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b < 0 \end{cases}$$

$$\frac{\partial \boldsymbol{w} \cdot \boldsymbol{x} + b}{\partial \boldsymbol{x}[k]} = \boldsymbol{w}[k]$$

If features are similar size then large weights indicate important features

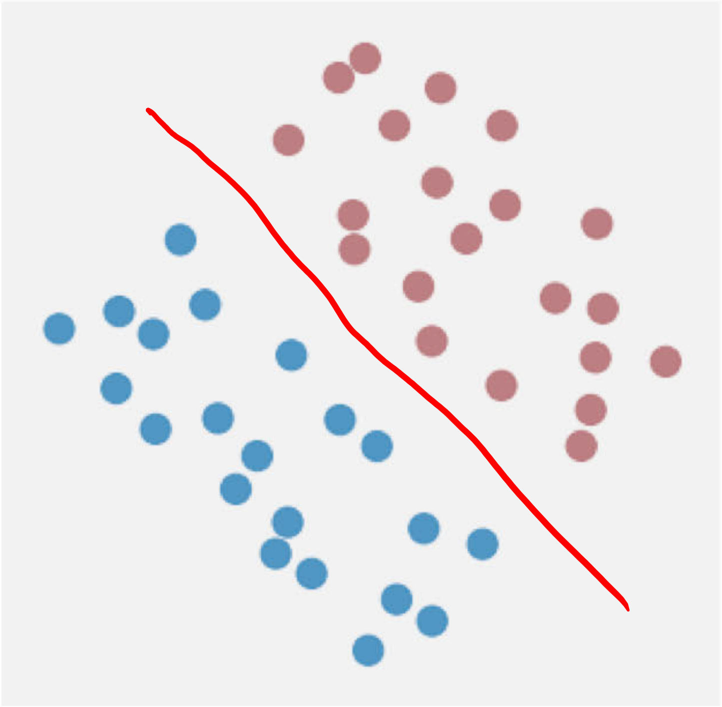
Interpretability of machine learning

- Example definitions
  - Can you understand how a prediction is made?
  - Can you explain how the model works as a whole?
  - Can you make better decisions with assistance of the model?

- K-nearest neighbors Yes, Yes, Unclear
- Perceptron Yes, Yes, Unclear

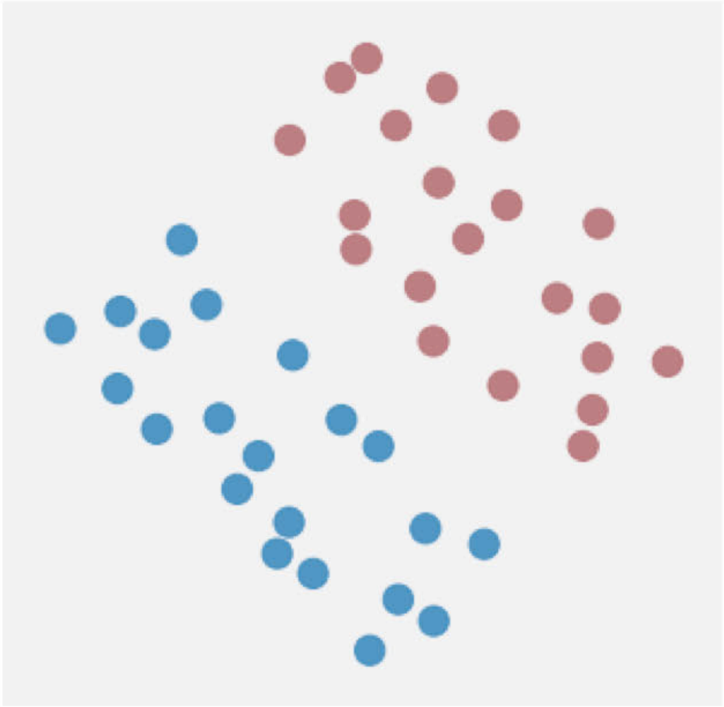
Convergence of the perceptron algorithm

- If possible for a linear classifier to separate data, Perceptron will find it



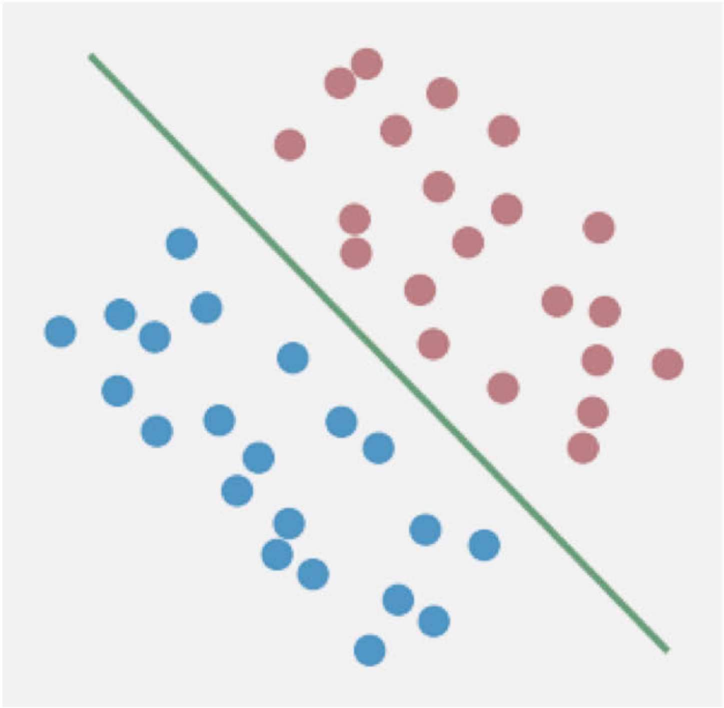
Convergence of the perceptron algorithm

- If possible for a linear classifier to separate data, Perceptron will find it
- Such training sets are called linearly separable



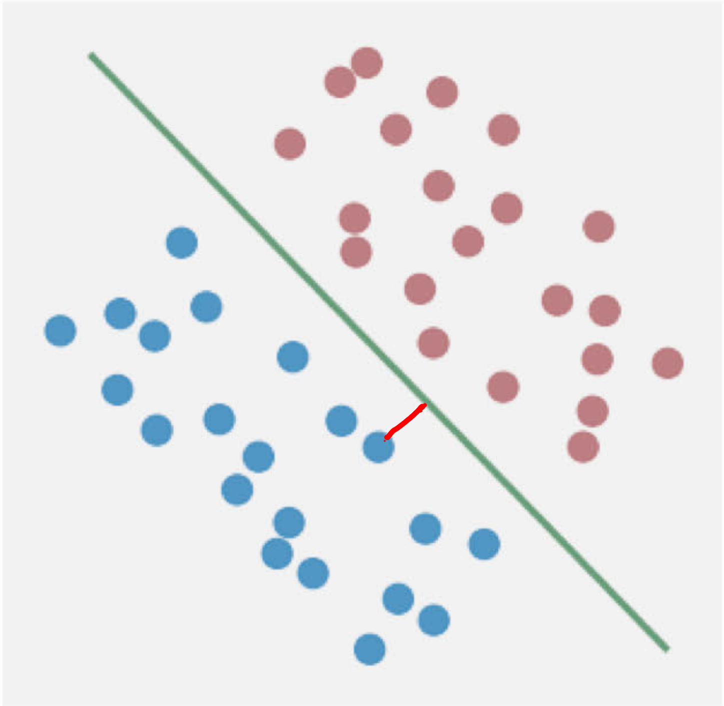
Convergence of the perceptron algorithm

- If possible for a linear classifier to separate data, Perceptron will find it
- Such training sets are called linearly separable



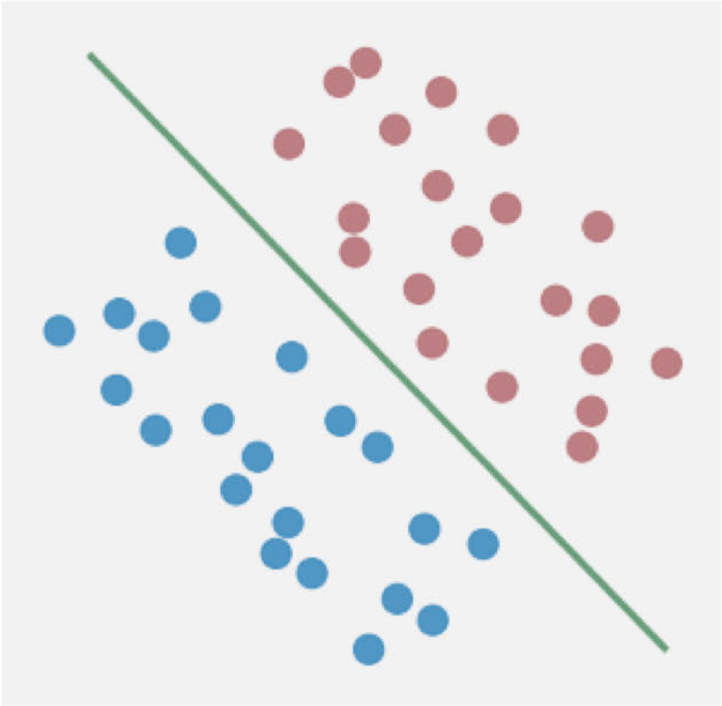
Convergence of the perceptron algorithm

- If possible for a linear classifier to separate data, Perceptron will find it
- Such training sets are called linearly separable
- *Margin* characterizes how separable a dataset is



Convergence of the perceptron algorithm

- If possible for a linear classifier to separate data, Perceptron will find it
- Such training sets are called linearly separable
- *Margin* characterizes how separable a dataset is
- How long it takes to converge depend on the margin



Convergence of the perceptron algorithm

Due to Rosenblatt [1958].

If D is linearly separable with margin $\gamma > 0$ and for all $n \in \{1, \ldots, N\}$, $\|\boldsymbol{x}_n\|_2 \leq 1$, then the perceptron algorithm will converge in at most $\frac{1}{\gamma^2}$ updates.

$$\gamma = \text{margin}(D, \boldsymbol{w}, b) = \begin{cases} \min_n y_n \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{if } \boldsymbol{w} \text{ and } b \text{ separate } D \\ -\infty & \text{otherwise} \end{cases}$$

Convergence of the perceptron algorithm

Due to Rosenblatt [1958].

If D is linearly separable with margin $\gamma > 0$ and for all $n \in \{1, \ldots, N\}$, $\|\boldsymbol{x}_n\|_2 \leq 1$, then the perceptron algorithm will converge in at most $\frac{1}{\gamma^2}$ updates.

$$\gamma = \text{margin}(D, \boldsymbol{w}, b) = \begin{cases} \min_n y_n \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{if } \boldsymbol{w} \text{ and } b \text{ separate } D \\ -\infty & \text{otherwise} \end{cases}$$

- Proof can be found in Daume [2017], pp. 50–51.
- The theorem does not guarantee that the perceptron's classifier will achieve margin γ.

Perceptron Wrap-up

- The perceptron is a simple classifier that sometimes works very well
- Linear classifiers in general will pop up again and again, e.g., Logistic Regression will be pretty similar
- The idea of margins will show up again when we talk about Support Vector Machines
- Neural Networks are essentially generalizations of the perceptron