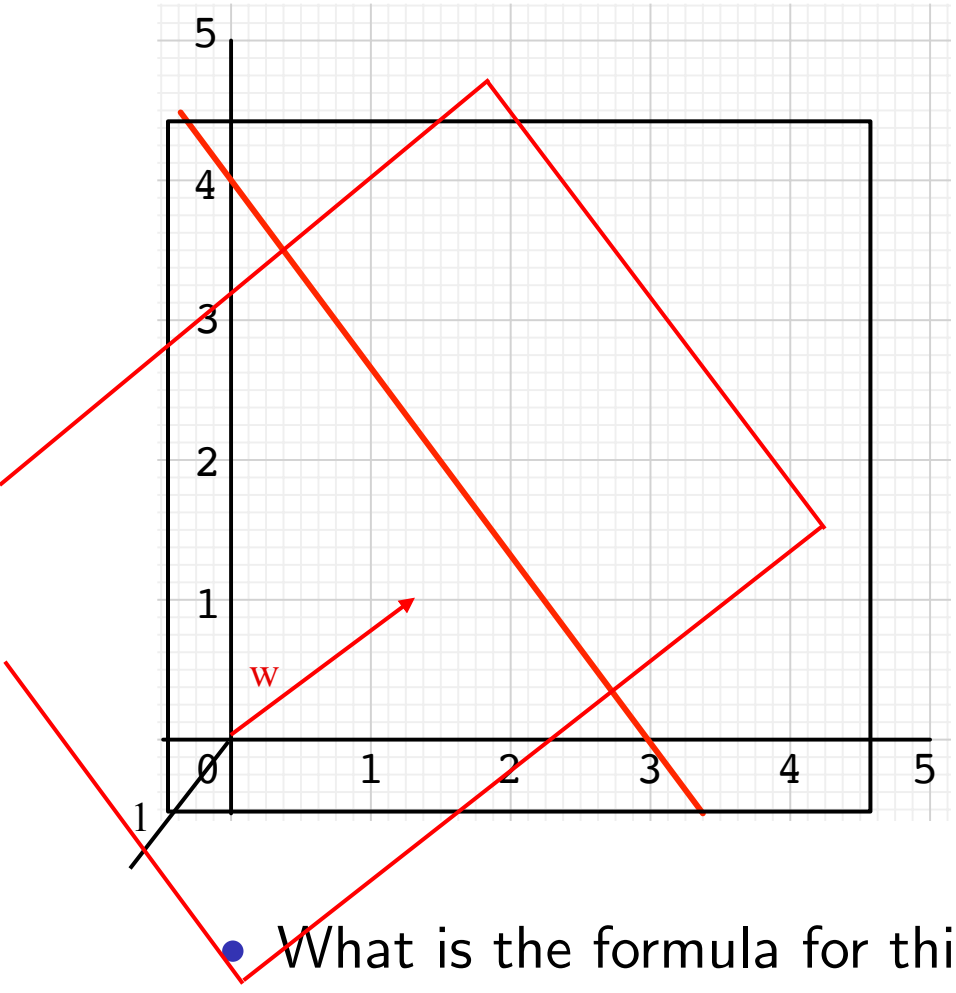
The geometry of linear classification

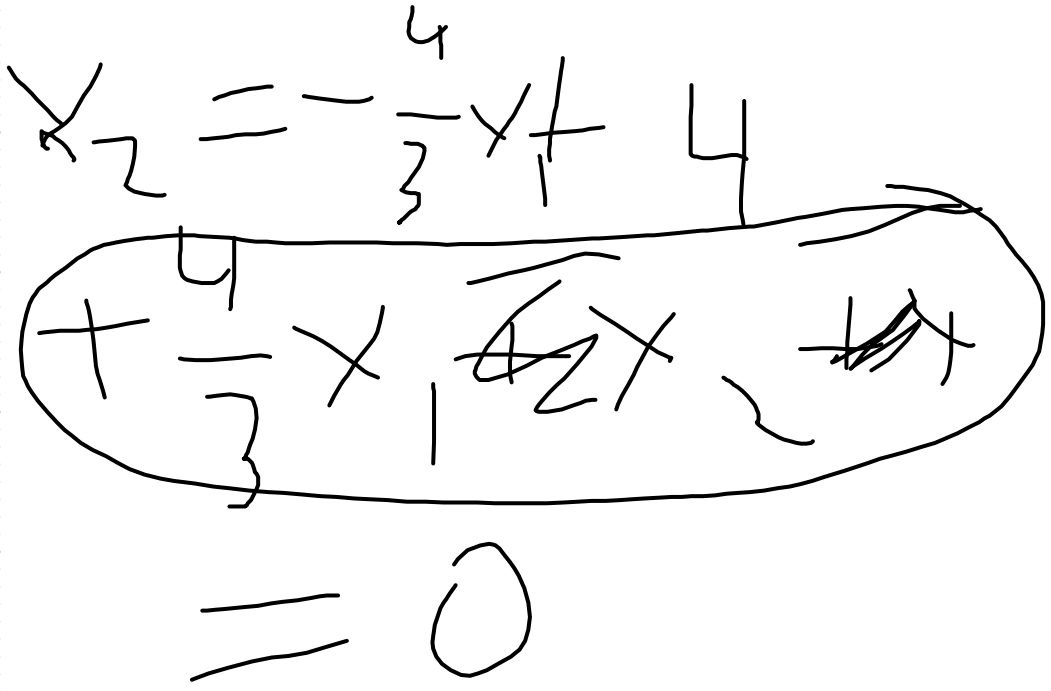
Instructor: Taylor Berg-Kirkpatrick
Slides: Sanjoy Dasgupta
Course website:
http://cseweb.ucsd.edu/classes/wi19/cse151-b/

Topics we'll cover

1. Linear decision boundary for binary classification
2. A loss function for classification
3. The Perceptron algorithm

Linear decision boundary for classification: example





- What is the formula for this boundary?
- What label would we predict for a new point x?

Linear classifiers

Binary classification problem: data $x \in \mathbb{R}^d$ and labels $y \in \{-1, +1\}$

- Linear classifier:
  - Parameters: $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$
  - Decision boundary $w \cdot x + b = 0$
  - On point x, predict label $\text{sign}(w \cdot x + b)$
- If the true label on point x is y:
  - Classifier correct if $y(w \cdot x + b) > 0$

A loss function for classification

What is the **loss** of our linear classifier (given by w, b) on a point (x, y)?

A loss function for classification

What is the **loss** of our linear classifier (given by w, b) on a point (x, y)?

One idea for a loss function:

- If $y(w \cdot x + b) > 0$: correct, no loss
- If $y(w \cdot x + b) < 0$: loss $= -y(w \cdot x + b)$

A simple learning algorithm

Fit a linear classifier w, b to the training set using **stochastic gradient descent**.

- Update w, b based on just one data point (x, y) at a time
- If $y(w \cdot x + b) > 0$: zero loss, no update
- If $y(w \cdot x + b) < 0$: loss is $-y(w \cdot x + b)$

A simple learning algorithm

Fit a linear classifier w, b to the training set using **stochastic gradient descent**.

- Update w, b based on just one data point (x, y) at a time
- If $y(w \cdot x + b) > 0$: zero loss, no update
- If $y(w \cdot x + b) < 0$: loss is $-y(w \cdot x + b)$

$\max(0, -y(w \cdot x + b))$

The Perceptron algorithm

- Initialize $w = 0$ and $b = 0$
- Keep cycling through the training data (x, y):
  - If $y(w \cdot x + b) \leq 0$ (i.e. point misclassified):
    - $w = w + yx$
    - $b = b + y$

The learning problem: separable case

Input: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$

Output: linear classifier $w \in \mathbb{R}^d$ such that for every $1 \leq i \leq n$,

$$\begin{aligned} w \cdot x^{(i)} &> 0 \quad \text{if } y^{(i)} = +1 \\ w \cdot x^{(i)} &< 0 \quad \text{if } y^{(i)} = -1 \end{aligned}$$

The learning problem: separable case

Input: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$

Output: linear classifier $w \in \mathbb{R}^d$ such that for every $1 \leq i \leq n$,

$$\begin{aligned} w \cdot x^{(i)} &> 0 \quad \text{if } y^{(i)} = +1 \\ w \cdot x^{(i)} &< 0 \quad \text{if } y^{(i)} = -1 \end{aligned}$$

More simply: $y^{(i)}(w \cdot x^{(i)}) > 0$ for all i.

The learning problem: separable case

Input: training data $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^d \times \{-1, +1\}$

Output: linear classifier $w \in \mathbb{R}^d$ such that for every $1 \leq i \leq n$,

$$\begin{aligned} w \cdot x^{(i)} &> 0 \quad \text{if } y^{(i)} = +1 \\ w \cdot x^{(i)} &< 0 \quad \text{if } y^{(i)} = -1 \end{aligned}$$

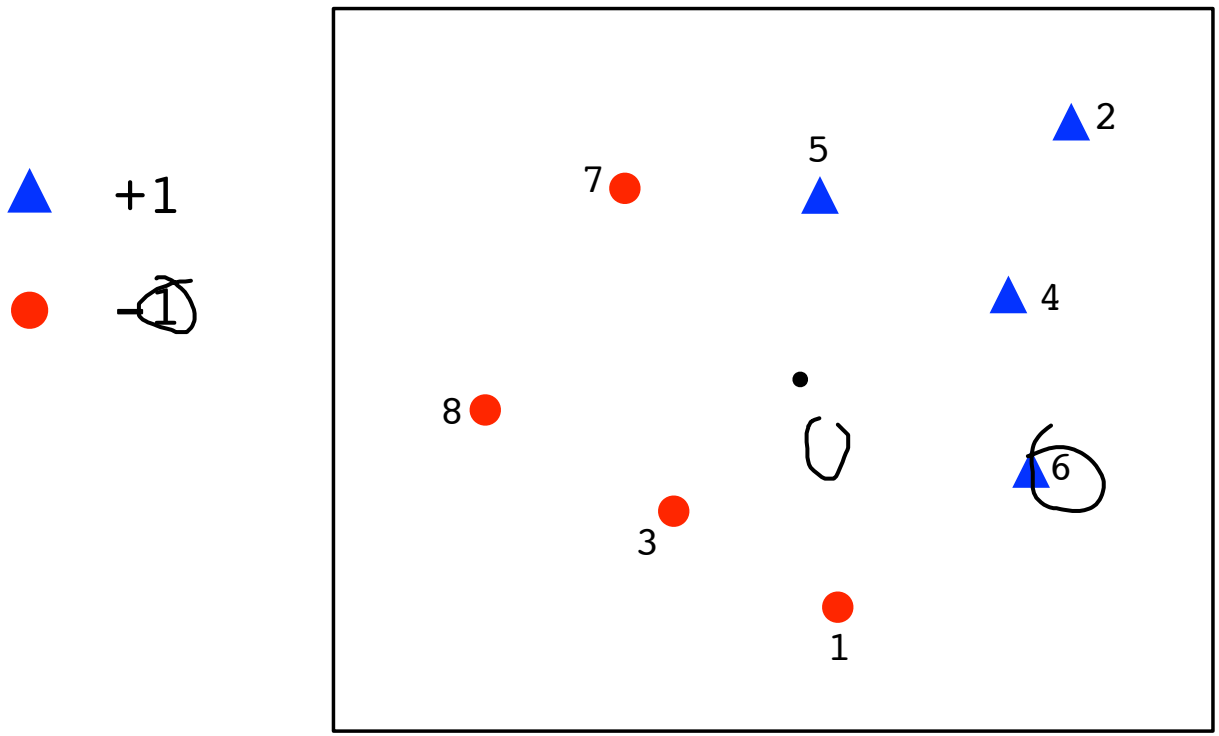
More simply: $y^{(i)}(w \cdot x^{(i)}) > 0$ for all i.

The **Perceptron algorithm** (Rosenblatt, 1958)

- $w = 0$
- while some (x, y) is misclassified:
  - $w = w + yx$

Perceptron: example

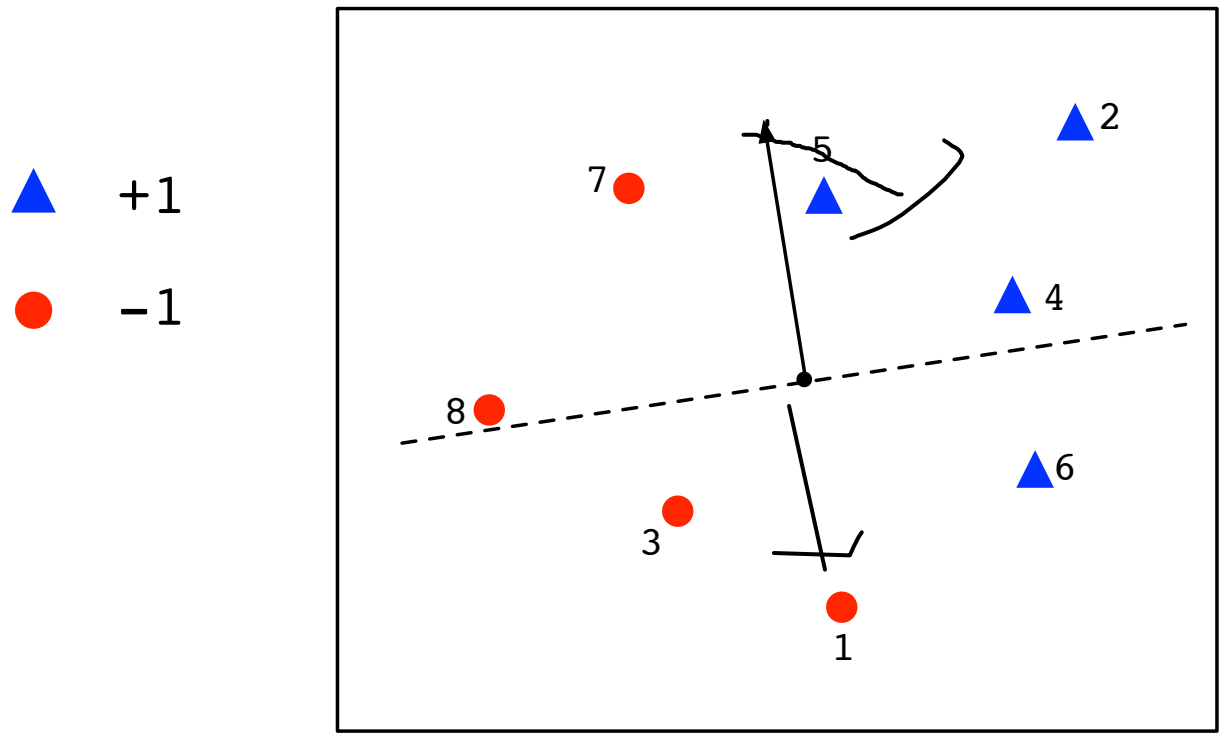
- $w = 0$
- while some (x, y) is misclassified:
  - $w = w + yx$



Separator: $w = 0$

Perceptron: example

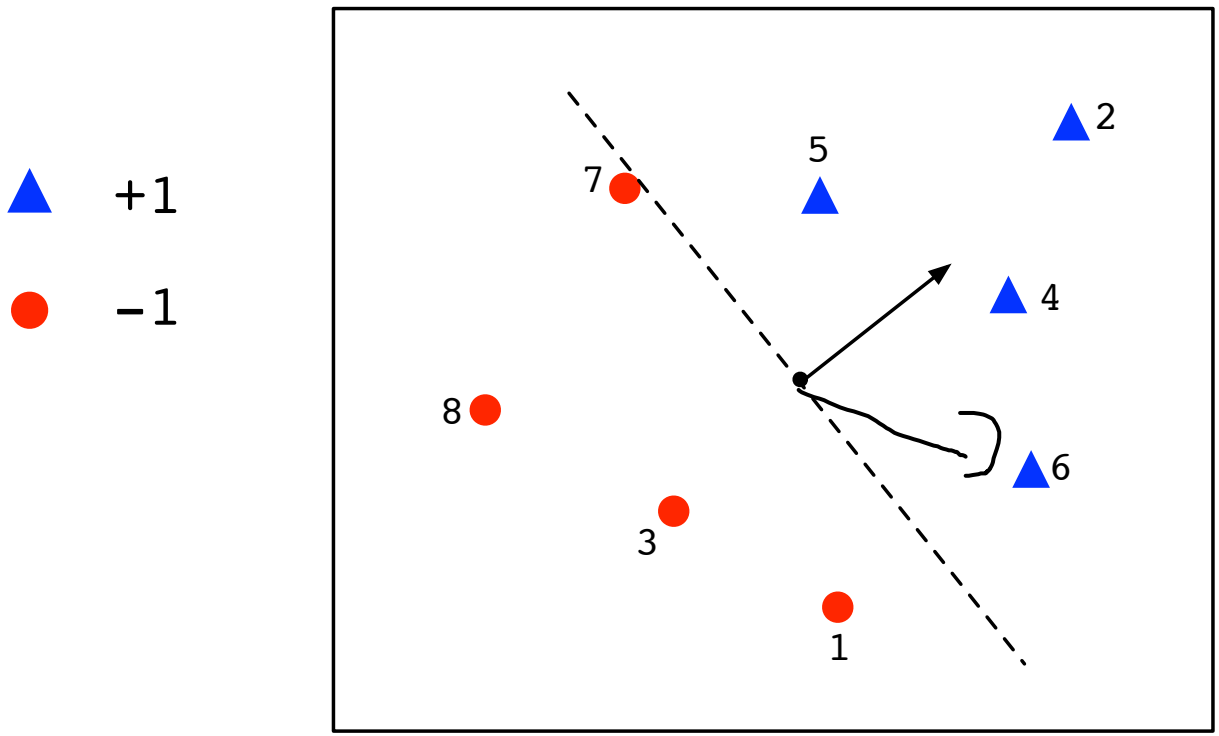
- $w = 0$
- while some (x, y) is misclassified:
  - $w = w + yx$



Separator: $w = -x^{(1)}$

Perceptron: example

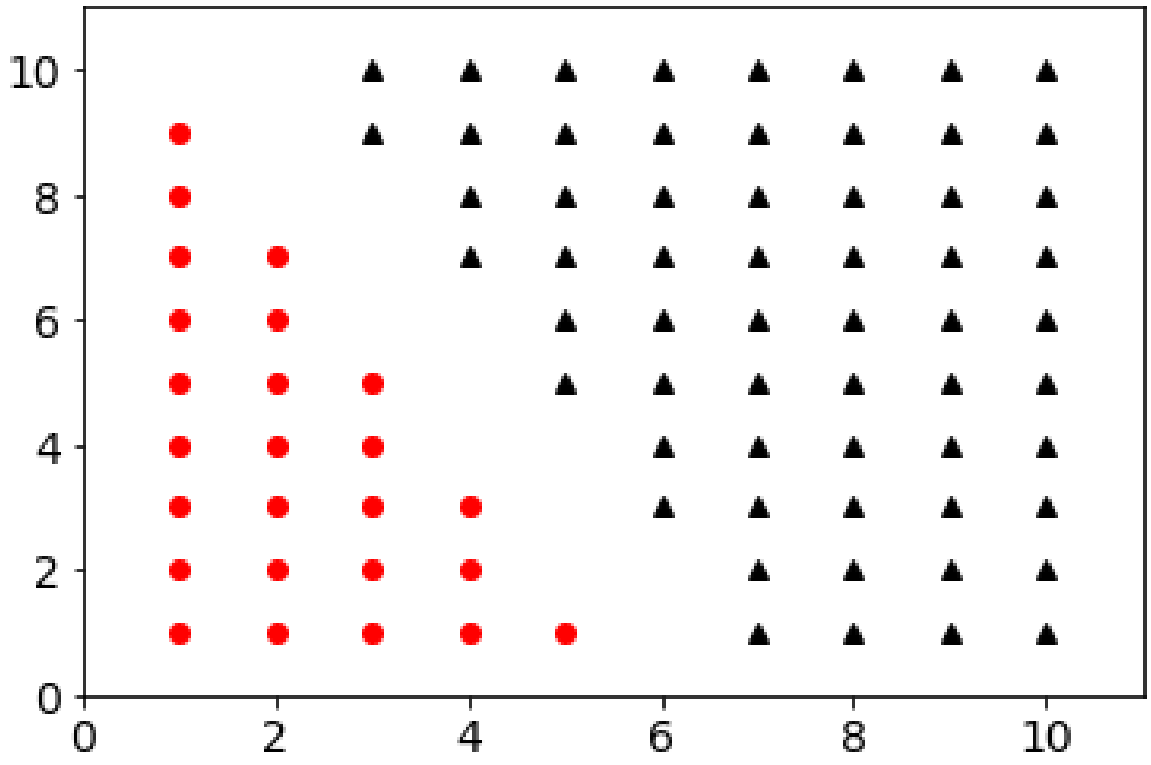
- $w = 0$
- while some (x, y) is misclassified:
  - $w = w + yx$



Separator: $w = -x^{(1)} + x^{(6)}$

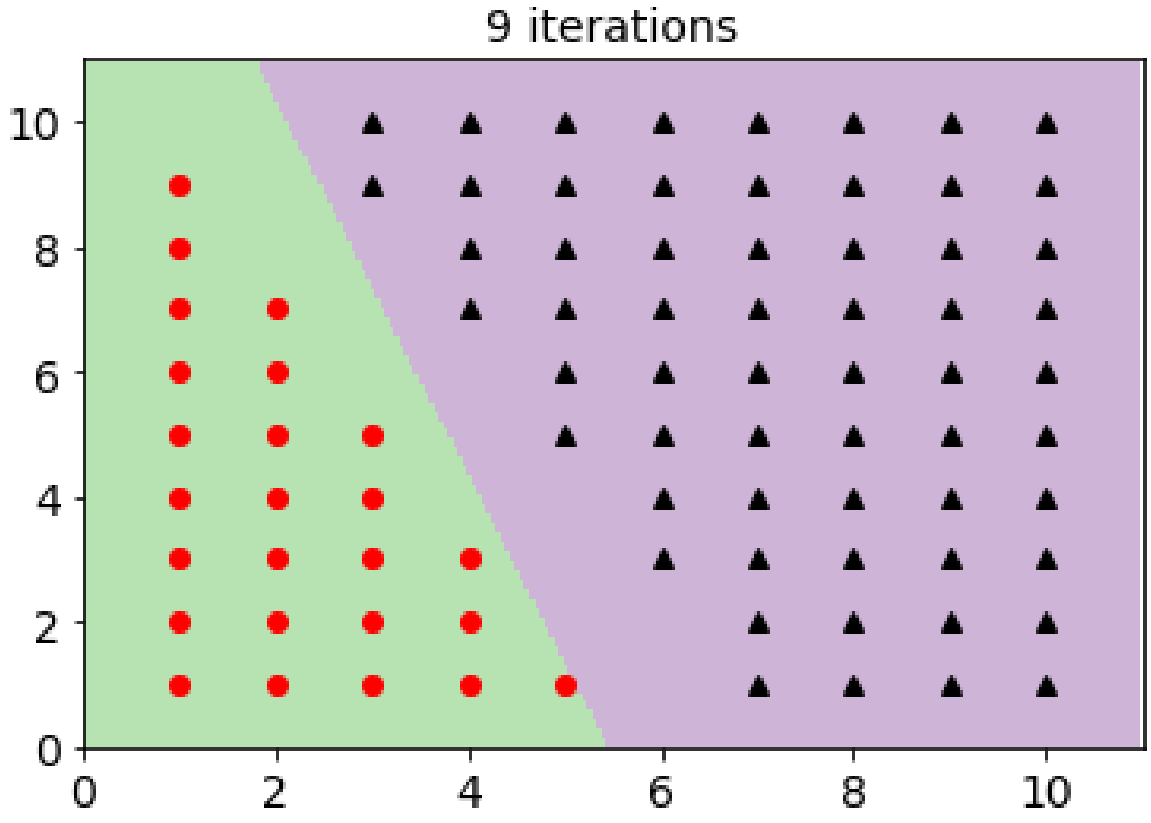
The Perceptron in action

85 data points, linearly separable.



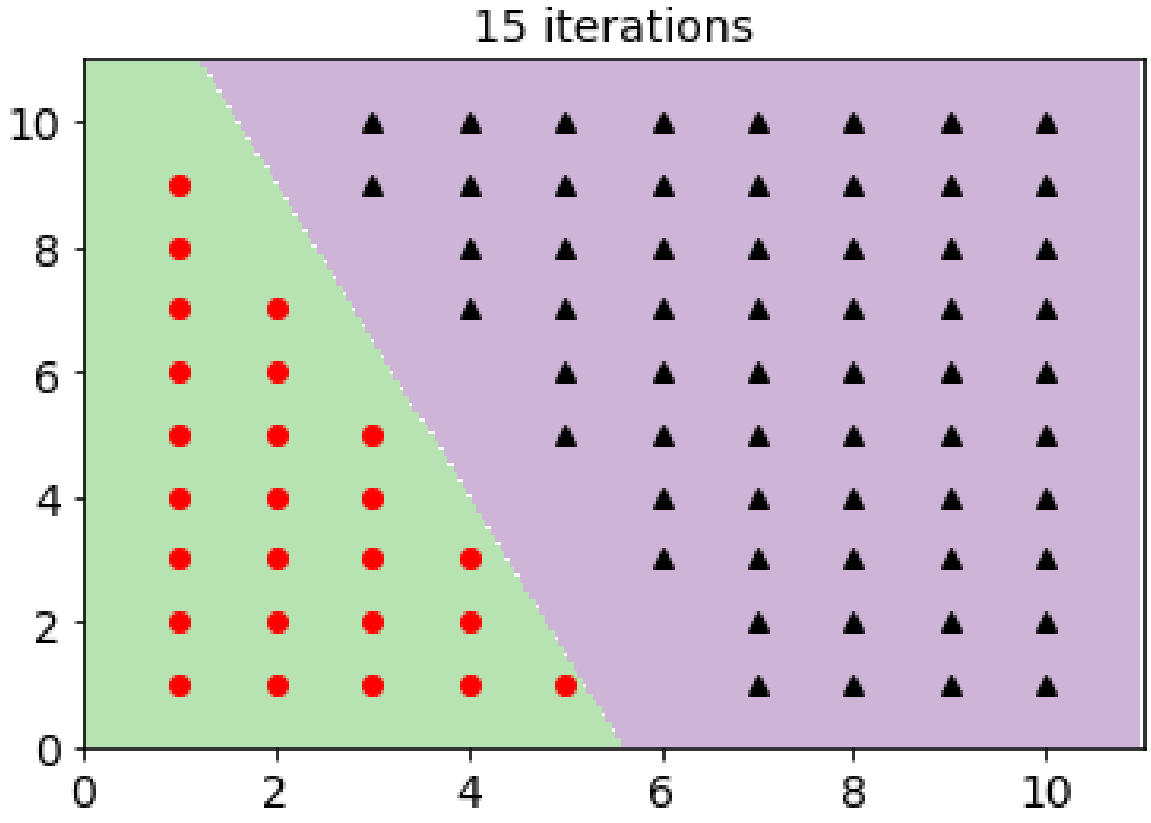
The Perceptron in action

85 data points, linearly separable.



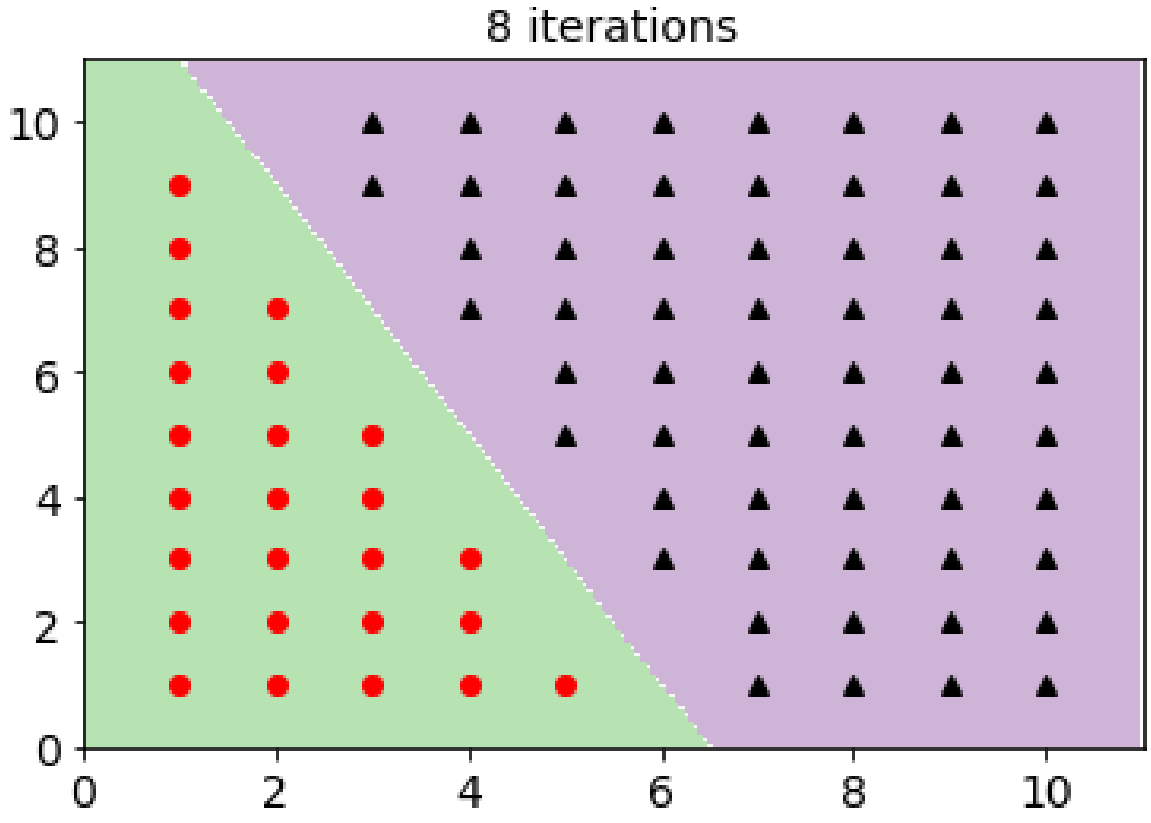
The Perceptron in action

85 data points, linearly separable.



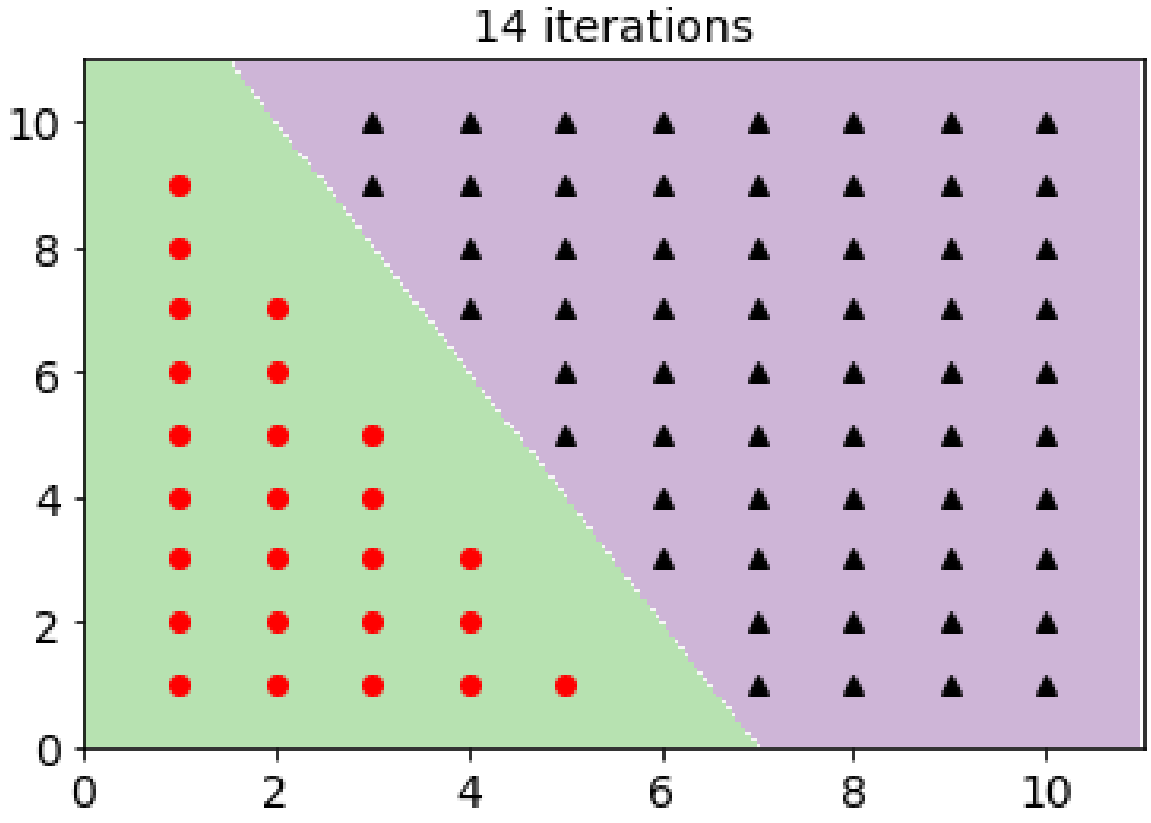
The Perceptron in action

85 data points, linearly separable.



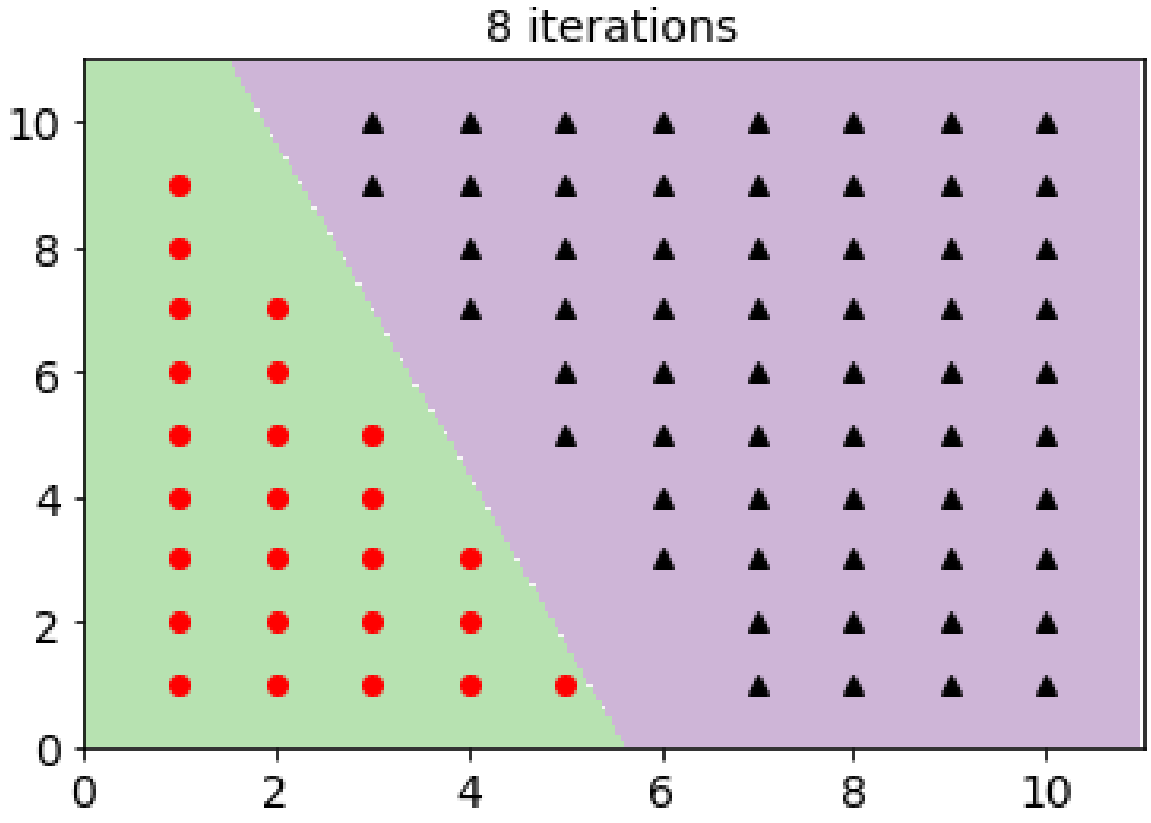
The Perceptron in action

85 data points, linearly separable.



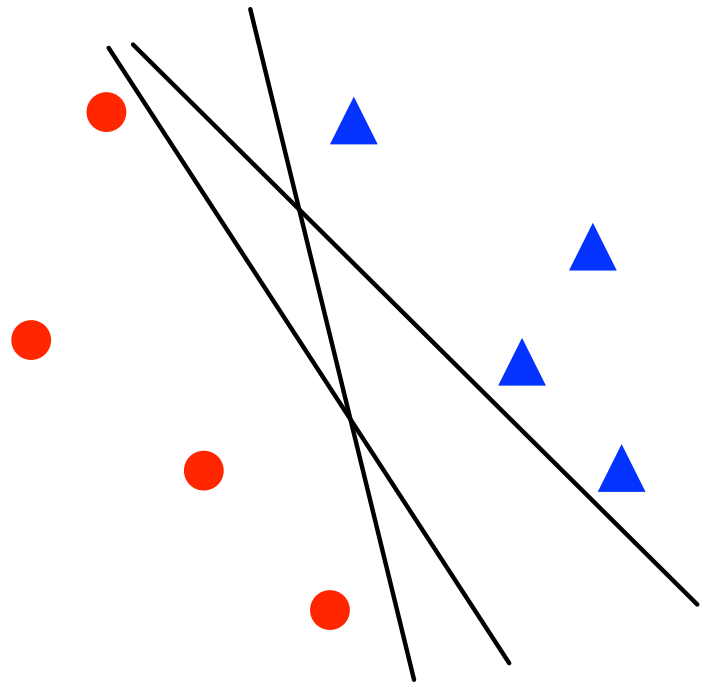
The Perceptron in action

85 data points, linearly separable.



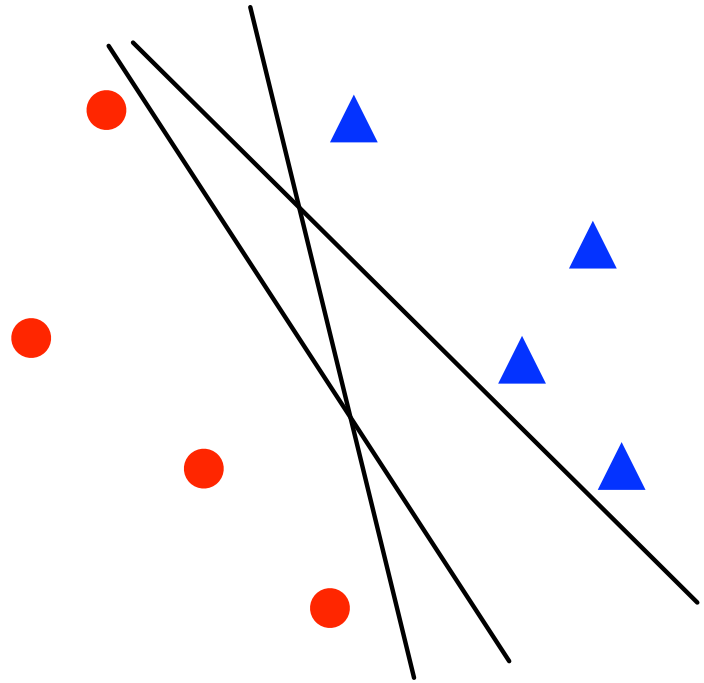
Perceptron: convergence

If the data are separable, the Perceptron algorithm is guaranteed to converge within a finite number of steps.



Perceptron: convergence

If the data are separable, the Perceptron algorithm is guaranteed to converge within a finite number of steps.



But is there a better, more systematic choice of separator? The one with the most buffer around it, for instance?