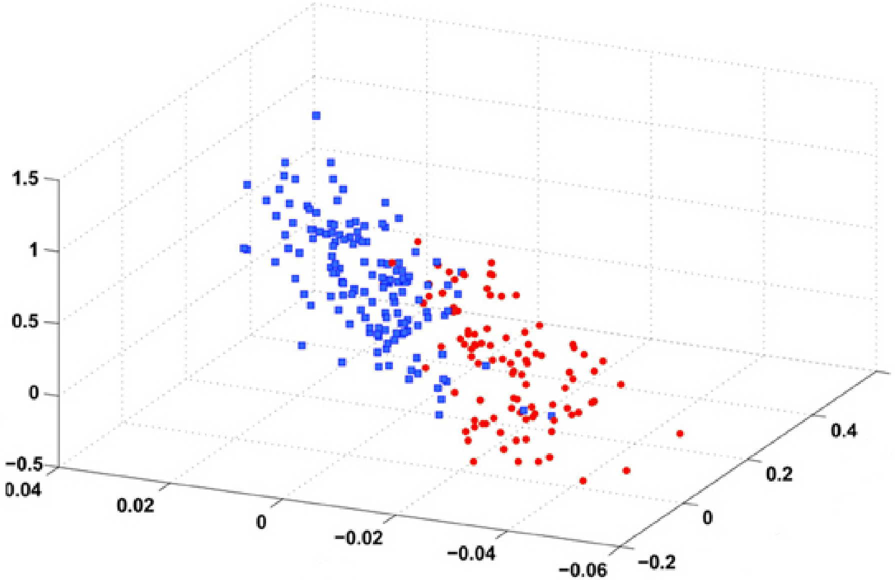
# CS189/CS289A
# Introduction to Machine Learning
# Lecture 2: Linear classifiers

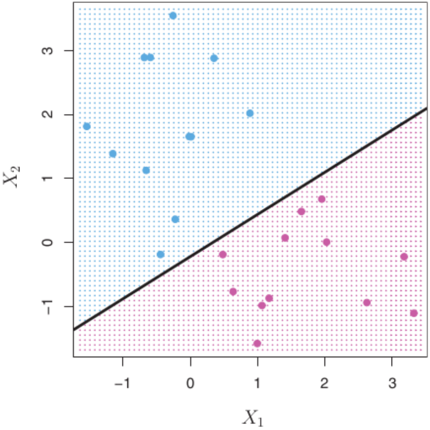
Peter Bartlett

January 22, 2015

# Linear Classifiers: $x \in \mathbb{R}^d$, $y \in \{-1, 1\}$

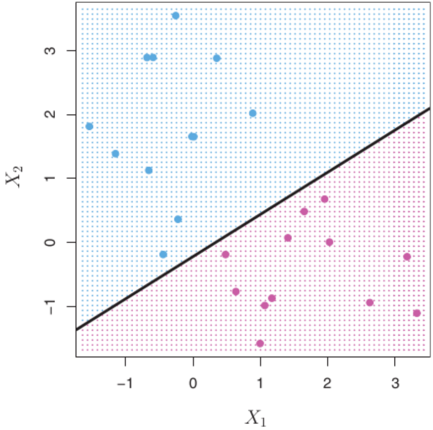


# Linear Classifiers



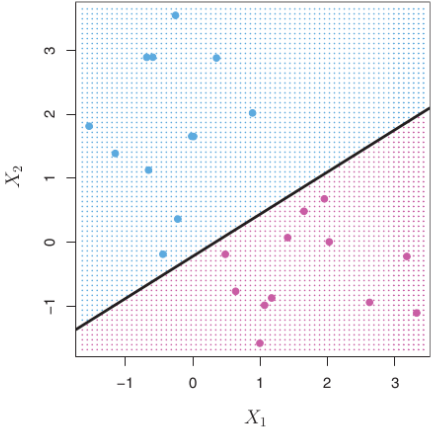

1. Training

# Linear Classifiers



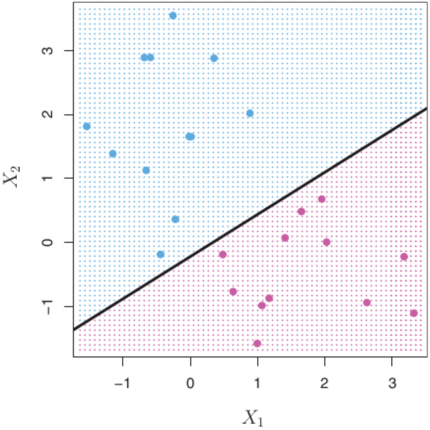

1. Training
   - Collect labeled data.

# Linear Classifiers



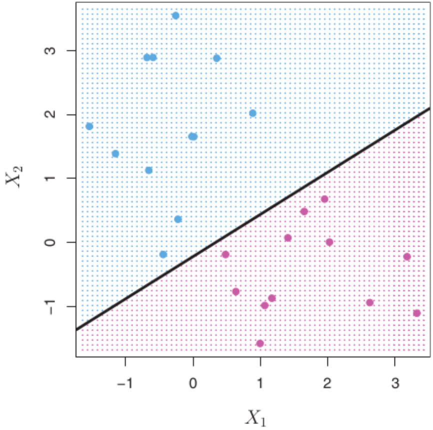

1. Training
   - Collect labeled data.
   - Find a good separating hyperplane.

# Linear Classifiers



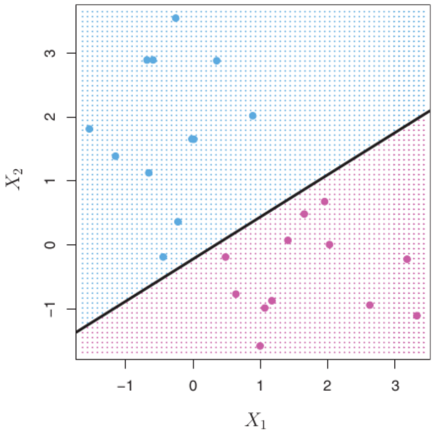

1. Training
   - Collect labeled data.
   - Find a good separating hyperplane.
2. Test

# Linear Classifiers




1. Training
   - Collect labeled data.
   - Find a good separating hyperplane.
2. Test
   - Given an unlabeled point, predict according to which side of the hyperplane it's on.

# Linear Classifiers




1. Training
   - Collect labeled data.
   - Find a good separating hyperplane.
2. Test
   - Given an unlabeled point, predict according to which side of the hyperplane it's on.

# Linear Classifiers: Outline for today

# Linear Classifiers: Outline for today

- Properties of linear classifiers.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
    - The perceptron algorithm (Rosenblatt, 1950s).

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
    - The perceptron algorithm (Rosenblatt, 1950s).
    - The perceptron convergence theorem.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
    - The perceptron algorithm (Rosenblatt, 1950s).
    - The perceptron convergence theorem.
    - The perceptron algorithm as a stochastic gradient method.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
  - The perceptron algorithm (Rosenblatt, 1950s).
  - The perceptron convergence theorem.
  - The perceptron algorithm as a stochastic gradient method.
  - Support vector machines (Vapnik, 1990s).

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
  - The perceptron algorithm (Rosenblatt, 1950s).
  - The perceptron convergence theorem.
  - The perceptron algorithm as a stochastic gradient method.
  - Support vector machines (Vapnik, 1990s).
  - Hard margin SVM.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
  - The perceptron algorithm (Rosenblatt, 1950s).
  - The perceptron convergence theorem.
  - The perceptron algorithm as a stochastic gradient method.
  - Support vector machines (Vapnik, 1990s).
  - Hard margin SVM.
  - Soft margin SVM.

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
  - The perceptron algorithm (Rosenblatt, 1950s).
  - The perceptron convergence theorem.
  - The perceptron algorithm as a stochastic gradient method.
  - Support vector machines (Vapnik, 1990s).
  - Hard margin SVM.
  - Soft margin SVM.
  - Both are quadratic programs.

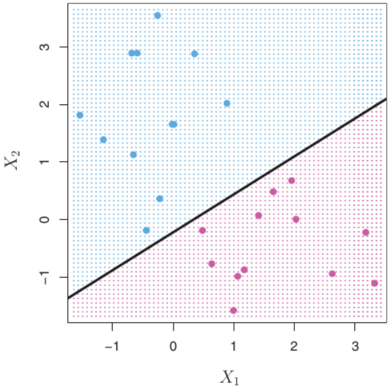
# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
    - The perceptron algorithm (Rosenblatt, 1950s).
    - The perceptron convergence theorem.
    - The perceptron algorithm as a stochastic gradient method.
    - Support vector machines (Vapnik, 1990s).
    - Hard margin SVM.
    - Soft margin SVM.
    - Both are quadratic programs.
    - Only use inner products.

# Linear threshold functions

## Linear classifier:

For a pattern $x \in \mathbb{R}^d$

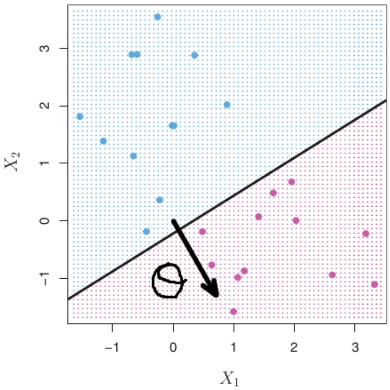


## Linear classifier:

For a pattern $x \in \mathbb{R}^d$ and parameters $\theta \in \mathbb{R}^d$, $\theta_0 \in \mathbb{R}$, define

$$f(x) = \theta \cdot x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0,$$

$$\hat{y} = \begin{cases} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{if } f(x) < 0. \end{cases}$$



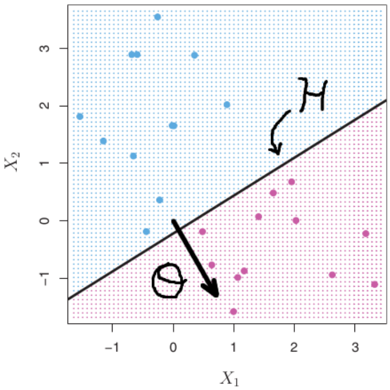
# Linear threshold functions

## Linear classifier:

For a pattern $x \in \mathbb{R}^d$ and parameters $\theta \in \mathbb{R}^d$, $\theta_0 \in \mathbb{R}$, define

$$f(x) = \theta \cdot x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0,$$

$$\hat{y} = \begin{cases} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{if } f(x) < 0. \end{cases}$$



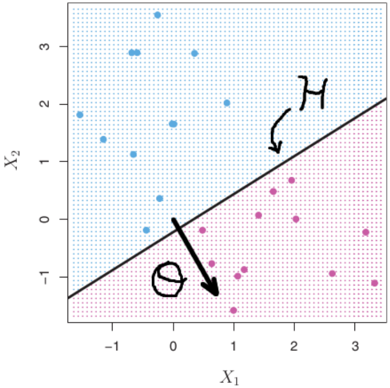

## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : f(x) = 0 \right\} = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

# Linear threshold functions

## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

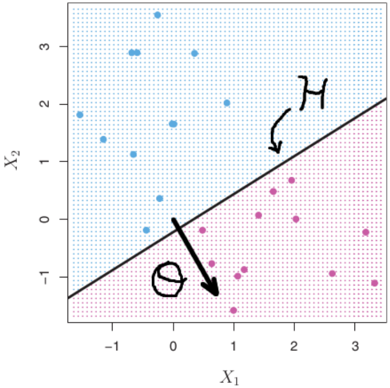


# Linear threshold functions

## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^2$, this is a line.

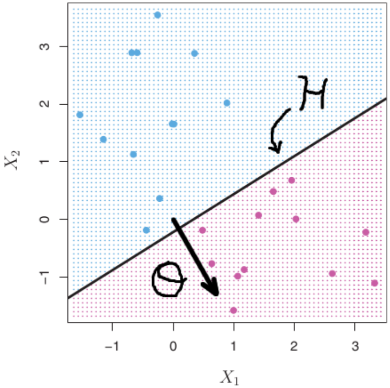


# Linear threshold functions

## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.



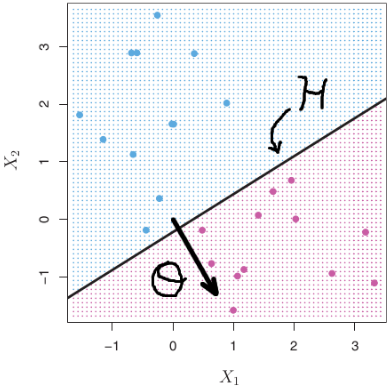
# Linear threshold functions

## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.

- It is normal to $\theta$:



# Linear threshold functions

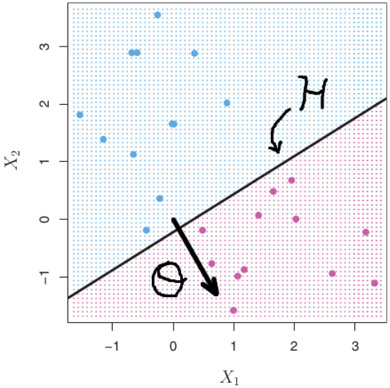
## Decision boundary:

$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.

- It is normal to $\theta$:
  For any $x, \tilde{x} \in H$, we have

$$\begin{aligned} 0 &= (\theta \cdot x + \theta_0) - (\theta \cdot \tilde{x} + \theta_0) \\ &= \theta \cdot (x - \tilde{x}). \end{aligned}$$



# Linear threshold functions

## Decision boundary:

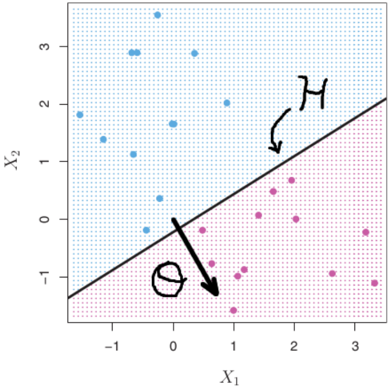
$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.

- It is normal to $\theta$:
  For any $x, \tilde{x} \in H$, we have

$$\begin{aligned} 0 &= (\theta \cdot x + \theta_0) - (\theta \cdot \tilde{x} + \theta_0) \\ &= \theta \cdot (x - \tilde{x}). \end{aligned}$$

- $\theta_0$ determines the offset:



# Linear threshold functions

## Decision boundary:

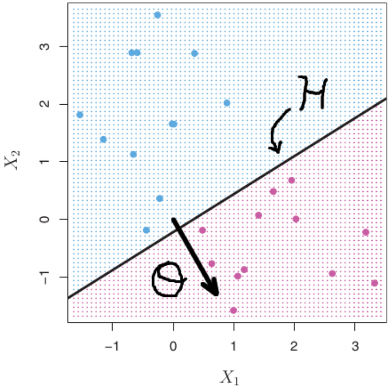
$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.

- It is normal to $\theta$:
  For any $x, \tilde{x} \in H$, we have

$$0 = (\theta \cdot x + \theta_0) - (\theta \cdot \tilde{x} + \theta_0)$$
$$= \theta \cdot (x - \tilde{x}).$$

- $\theta_0$ determines the offset:
  The signed distance (in the direction of $\theta$) from the origin to $H$ is $-\theta_0 / \|\theta\|$.



# Linear threshold functions

## Decision boundary:

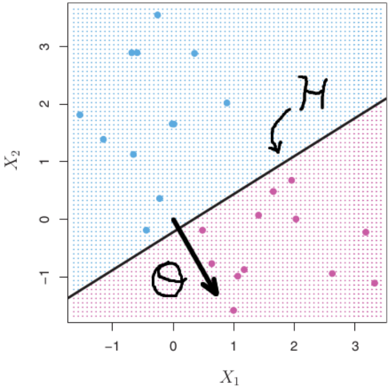
$$H = \left\{ x \in \mathbb{R}^d : \theta \cdot x + \theta_0 = 0 \right\}.$$

In $\mathbb{R}^d$, it is a hyperplane.

- It is normal to $\theta$:
  For any $x, \tilde{x} \in H$, we have

  $$\begin{aligned} 0 &= (\theta \cdot x + \theta_0) - (\theta \cdot \tilde{x} + \theta_0) \\ &= \theta \cdot (x - \tilde{x}). \end{aligned}$$

- $\theta_0$ determines the offset:
  The signed distance (in the direction of $\theta$) from the origin to $H$ is $-\theta_0/\|\theta\|$. (Why?)



# Aside: detail

How far away from the origin is the hyperplane? Since $\theta$ is normal to $H$, we can look at the ray from the origin towards $H$ in the direction of $\theta$. That ray hits $H$ at $c\theta$, where

$$0 = f(c\theta) = c\theta \cdot \theta + \theta_0 = c\|\theta\|^2 + \theta_0.$$

So $c = -\theta_0/\|\theta\|^2$, and the signed distance to $H$ is

$$c\,\|\theta\| = -\theta_0 \frac{\|\theta\|}{\|\theta\|^2} = -\frac{\theta_0}{\|\theta\|}.$$

# Linear threshold functions

We can simplify our notation by augmenting the pattern $x \in \mathbb{R}^d$ with a constant component. This allows us to dispense with the offset $\theta_0$:
For a pattern $x \in \mathbb{R}^d$, and parameters $\theta \in \mathbb{R}^d$, $\theta_0 \in \mathbb{R}$, define

$$\tilde{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}, \qquad \tilde{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{pmatrix}.$$

Then define

$$\tilde{f}(\tilde{x}) := \tilde{\theta} \cdot \tilde{x} = \sum_{i=1}^{d} \theta_i x_i + \theta_0 = \theta \cdot x + \theta_0 = f(x),$$

$$\hat{y} = \begin{Bmatrix} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{if } f(x) < 0. \end{Bmatrix} = \begin{cases} 1 & \text{if } \tilde{f}(\tilde{x}) \geq 0, \\ -1 & \text{if } \tilde{f}(\tilde{x}) < 0. \end{cases}$$

# Linear threshold functions

So we can always consider linear classifiers of this simpler form:

$$f(x) = \theta \cdot x,$$

$$\hat{y} = \begin{cases} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{if } f(x) < 0. \end{cases}$$

The decision boundary is the hyperplane

$$H = \{x : \theta \cdot x = 0\},$$

passing through the origin, normal to the vector $\theta$.

# Linear classifiers

Now that we understand the functions computed by linear classifiers, let's consider how we might use labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ to choose a good classifier.

# Linear classifiers

Now that we understand the functions computed by linear classifiers, let's consider how we might use labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ to choose a good classifier.

A good classifier is one that classifies subsequent $(x, y)$ pairs accurately.

# Linear classifiers

Now that we understand the functions computed by linear classifiers, let's consider how we might use labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ to choose a good classifier.

A good classifier is one that classifies subsequent $(x, y)$ pairs accurately.

**A reasonable approach:**

Find a linear classifier that fits the training data well.

# Linear classifiers

Now that we understand the functions computed by linear classifiers, let's consider how we might use labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ to choose a good classifier.

A good classifier is one that classifies subsequent $(x, y)$ pairs accurately.

**A reasonable approach:**

Find a linear classifier that fits the training data well.

For instance, we might aim to *minimize the number of misclassifications* on the training data.

# Linear classifiers

Now that we understand the functions computed by linear classifiers, let's consider how we might use labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ to choose a good classifier.
A good classifier is one that classifies subsequent $(x, y)$ pairs accurately.

**A reasonable approach:**

Find a linear classifier that fits the training data well.
For instance, we might aim to *minimize the number of misclassifications* on the training data.
This is known as *empirical risk minimization*.

# Perceptron algorithm

**Perceptron algorithm:**

Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    improve $\theta$

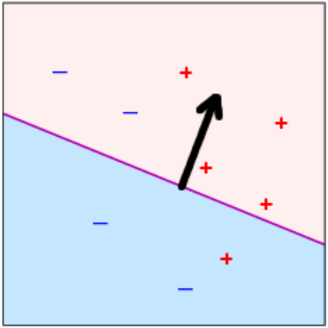
Return $\theta$.

# Perceptron algorithm

**Perceptron algorithm:**

Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    improve $\theta$

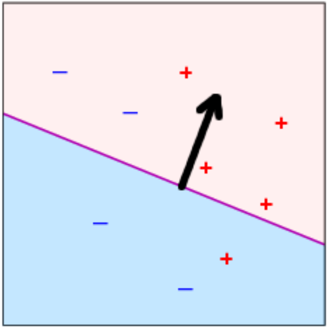
Return $\theta$.



# Perceptron algorithm

**Perceptron algorithm:**

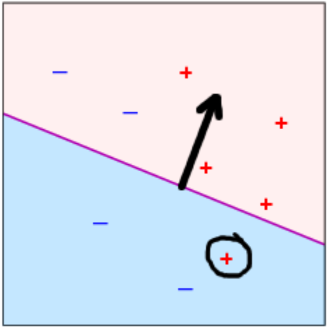
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified $(x^i, y^i)$
    update $\theta$
Return $\theta$.



# Perceptron algorithm

**Perceptron algorithm:**

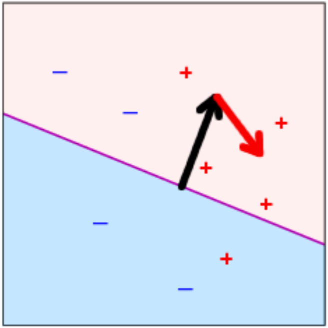
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified $(x^i, y^i)$
    update $\theta$
Return $\theta$.



# Perceptron algorithm

**Perceptron algorithm:**

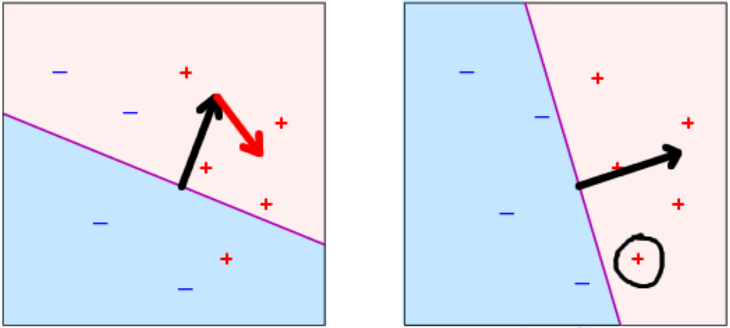
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \operatorname{sign}(\theta \cdot x^i)$
    pick some misclassified $(x^i, y^i)$
    $\theta \leftarrow \theta + \quad x^i$
Return $\theta$.



# Perceptron algorithm

## Perceptron algorithm:

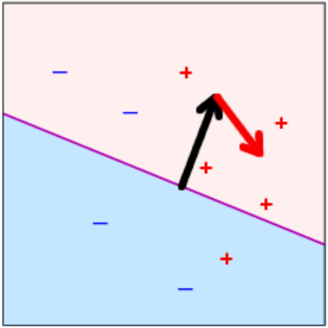
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified $(x^i, y^i)$
    $\theta \leftarrow \theta + \quad x^i$
Return $\theta$.

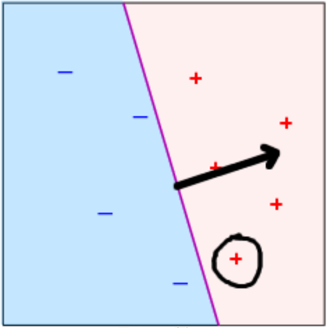


# Perceptron algorithm

**Perceptron algorithm:**

Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified $(x^i, y^i)$
    $\theta \leftarrow \theta + y^i x^i$
Return $\theta$.



# Linear threshold functions

## Perceptron convergence theorem

Given *linearly separable data* (i.e., there is a $\theta \in \mathbb{R}^d$ such that for all $i$, $y^i \theta \cdot x^i > 0$), for any choices of updates, the perceptron algorithm terminates with all data correctly classified.

# Linear threshold functions

## Perceptron convergence theorem

Given *linearly separable data* (i.e., there is a $\theta \in \mathbb{R}^d$ such that for all $i$, $y^i \theta \cdot x^i > 0$), for any choices of updates, the perceptron algorithm terminates with all data correctly classified.
Furthermore, it makes no more than $\frac{R^2}{\gamma^2}$ updates, where

$$R = \max_i \|x^i\|, \qquad \text{(radius of data)}$$

$$\gamma = \min_i \frac{y^i(\theta \cdot x^i)}{\|\theta\|}. \qquad \text{(margin)}$$

NB: $\frac{1}{\|\theta\|}\theta \cdot x$ is the signed distance from $H$ to $x$ (in the direction $\theta$).

# Linear threshold functions

## Perceptron convergence theorem

Given *linearly separable data* (i.e., there is a $\theta \in \mathbb{R}^d$ such that for all $i$, $y^i \theta \cdot x^i > 0$), for any choices of updates, the perceptron algorithm terminates with all data correctly classified.
Furthermore, it makes no more than $\frac{R^2}{\gamma^2}$ updates, where

$$R = \max_i \|x^i\|, \qquad \text{(radius of data)}$$

$$\gamma = \min_i \frac{y^i(\theta \cdot x^i)}{\|\theta\|}. \qquad \text{(margin)}$$

*Proof idea:* Fix a separating $\theta^*$. Each update increases $\theta \cdot \theta^*$ a lot, but only increases $\|\theta\|$ a little. So there can't be too many updates.

# Perceptron algorithm: only uses inner products

**Properties:**

- $\theta = \sum_i \alpha^i y^i x^i$.

# Perceptron algorithm: only uses inner products

**Properties:**

- $\theta = \sum_i \alpha^i y^i x^i$.
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x.$$

# Perceptron algorithm: only uses inner products

**Properties:**

- $\theta = \sum_i \alpha^i y^i x^i$.
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x.$$

- So we can work with data in any inner product space (not just finite-dimensional vectors).

# Perceptron algorithm

We can also view the perceptron algorithm as a stochastic gradient method to minimize a cost function.

# Perceptron algorithm

We can also view the perceptron algorithm as a stochastic gradient method to minimize a cost function.

## Margin cost function

- We want the sign of $\theta \cdot x^i$ to be the same as $y^i$.

# Perceptron algorithm

We can also view the perceptron algorithm as a stochastic gradient method to minimize a cost function.

## Margin cost function

- We want the sign of $\theta \cdot x^i$ to be the same as $y^i$.
- We want $y^i(\theta \cdot x^i) > 0$.

# Perceptron algorithm

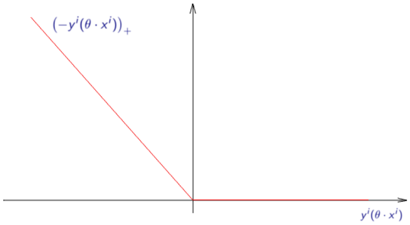
We can also view the perceptron algorithm as a stochastic gradient method to minimize a cost function.

## Margin cost function

- We want the sign of $\theta \cdot x^i$ to be the same as $y^i$.
- We want $y^i(\theta \cdot x^i) > 0$.
- Define

$$J(\theta) = \sum_i \left(-y^i(\theta \cdot x^i)\right)_+$$

$$(\alpha)_+ = \begin{cases} \alpha & \text{if } \alpha > 0, \\ 0 & \text{otherwise.} \end{cases}$$



# Gradients

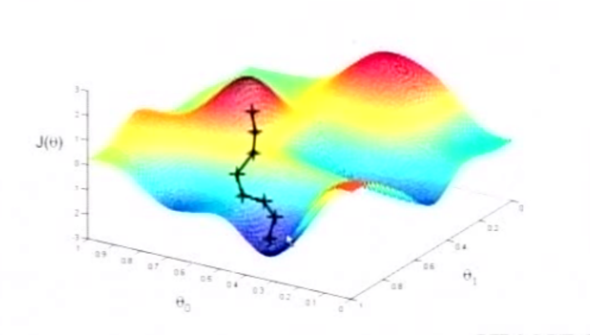
$$\nabla J(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} J(\theta) \\ \frac{\partial}{\partial \theta_2} J(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_d} J(\theta) \end{pmatrix}.$$

Example:

$$\nabla \left(\theta \cdot x\right) = \nabla \left( \sum_{i=1}^{d} \theta_i x_i \right) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \theta \cdot x \\ \frac{\partial}{\partial \theta_2} \theta \cdot x \\ \vdots \\ \frac{\partial}{\partial \theta_d} \theta \cdot x \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = x.$$

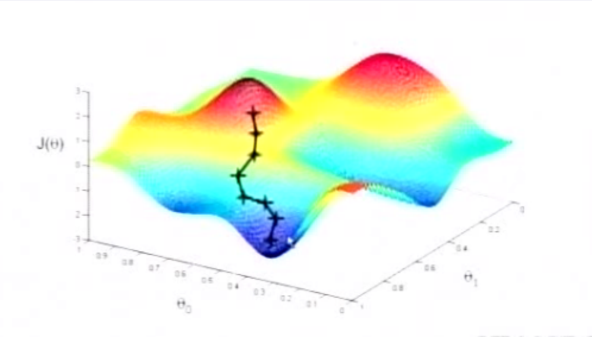
# Perceptron algorithm

## Gradient descent



# Perceptron algorithm

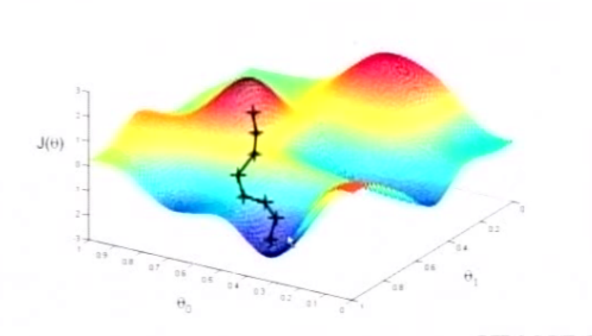
## Gradient descent




$$\theta \leftarrow \theta - \eta \nabla J(\theta)$$

# Perceptron algorithm

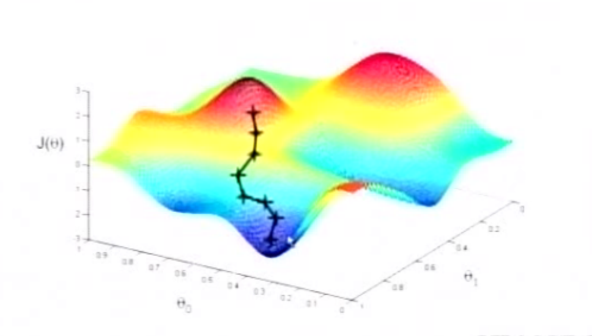
## Gradient descent




$$\theta \leftarrow \theta - \eta \underbrace{\nabla J(\theta)}_{\text{uphill}}$$

# Perceptron algorithm

## Gradient descent




$$\theta \leftarrow \theta - \underbrace{\eta}_{\text{stepsize}} \underbrace{\nabla J(\theta)}_{\text{uphill}}$$

# Perceptron algorithm

$$J(\theta) = \sum_i J_i(\theta) = \sum_i \left(-y^i(\theta \cdot x^i)\right)_+$$

$$\nabla J(\theta) = \sum_i \nabla J_i(\theta)$$

$$\nabla J_i(\theta) = \begin{cases} -y^i x^i & \text{if } y^i(\theta \cdot x^i) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

## Stochastic gradient

- Pick a component $J_i$ of the cost function $J$ (i.e., an $(x^i, y^i)$ pair)
- Move downhill wrt that component:
  If $y^i(\theta \cdot x^i) > 0$, don't change $\theta$ (because the gradient is zero.)
  Otherwise: $\theta \leftarrow \theta - \nabla J_i(\theta) = \theta + y^i x^i$.

# Perceptron algorithm

$$J(\theta) = \sum_i J_i(\theta) = \sum_i \left(-y^i(\theta \cdot x^i)\right)_+$$

$$\nabla J(\theta) = \sum_i \nabla J_i(\theta)$$

$$\nabla J_i(\theta) = \begin{cases} -y^i x^i & \text{if } y^i(\theta \cdot x^i) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

## Stochastic gradient

- Pick a component $J_i$ of the cost function $J$ (i.e., an $(x^i, y^i)$ pair)
- Move downhill wrt that component:
  If $y^i(\theta \cdot x^i) > 0$, don't change $\theta$ (because the gradient is zero.)
  Otherwise: $\theta \leftarrow \theta - \nabla J_i(\theta) = \theta + y^i x^i$. **Perceptron algorithm**

# Perceptron algorithm

**Issues:**

- Converges only if the data are *separable*.

# Perceptron algorithm

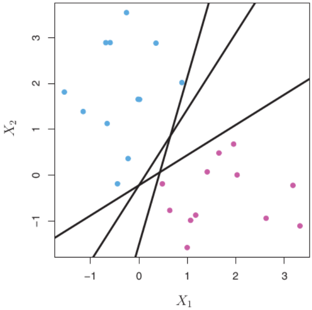
**Issues:**

- Converges only if the data are *separable*.
- Time to convergence depends on *margin*.

# Perceptron algorithm

**Issues:**

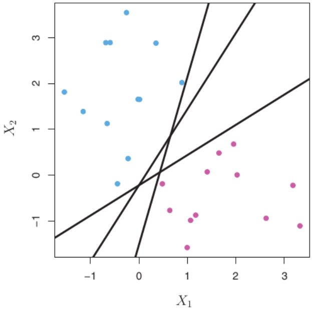
- Converges only if the data are *separable*.
- Time to convergence depends on *margin*.
- The solution depends on starting point.



# Perceptron algorithm

**Issues:**

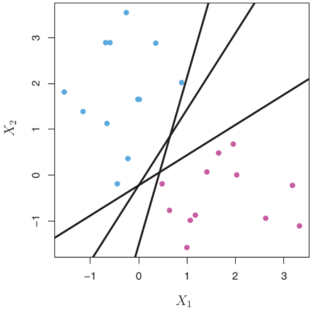
- Converges only if the data are *separable*.
- Time to convergence depends on *margin*.
- The solution depends on starting point.
- Will not converge if data are not separable.



# Perceptron algorithm

**Issues:**

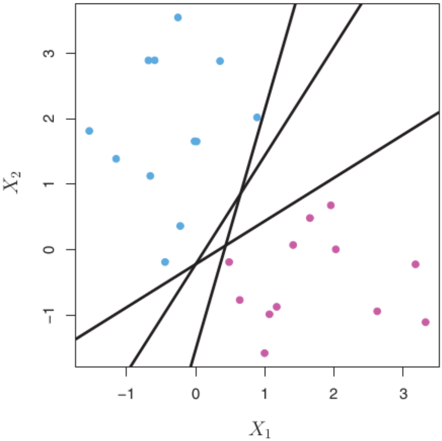
- Converges only if the data are *separable*.
- Time to convergence depends on *margin*.
- The solution depends on starting point.
- Will not converge if data are not separable.




(Looks like a toy example. But stochastic gradient methods turn out to be very useful for large scale problems.)

# Support vector machines

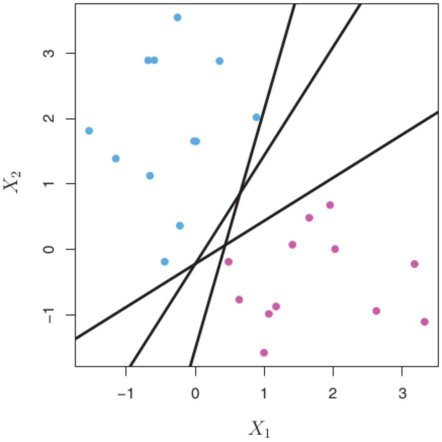
- There are always many linear classifiers that give identical classifications of the training data. Which is better?



# Support vector machines

- There are always many linear classifiers that give identical classifications of the training data. Which is better?
- One option is to choose the classifier that maximizes the *margin* on the training data, where the margin is the minimum over $(x^i, y^i)$ pairs of the signed distance to the decision boundary,

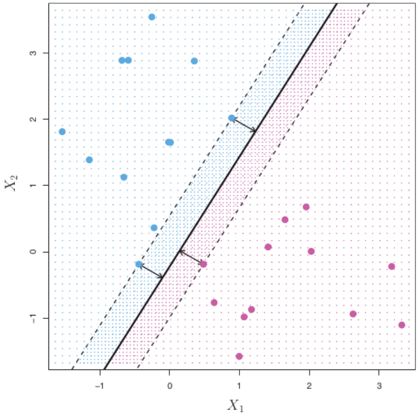
$$\text{margin of } (x^i, y^i) := y^i \frac{\theta \cdot x^i}{\|\theta\|}.$$



# Support vector machines

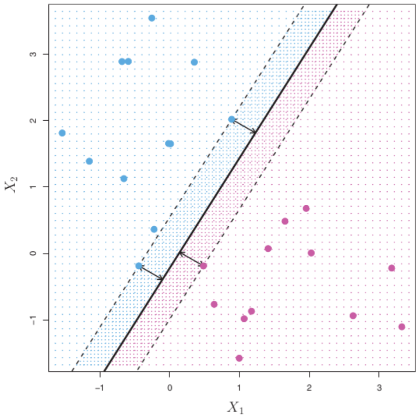
- There are always many linear classifiers that give identical classifications of the training data. Which is better?
- One option is to choose the classifier that maximizes the *margin* on the training data, where the margin is the minimum over $(x^i, y^i)$ pairs of the signed distance to the decision boundary,

$$\text{margin of } (x^i, y^i) := y^i \frac{\theta \cdot x^i}{\|\theta\|}.$$



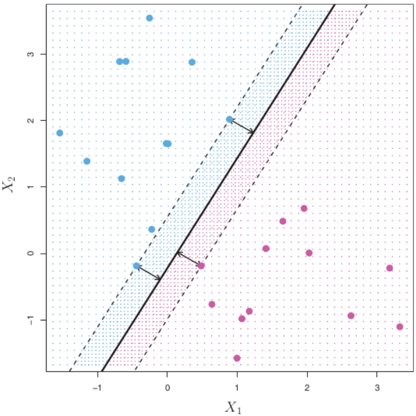
# Support vector machines

- Maximizing the margin means choosing the separating hyperplane so that the sandwich around it containing no data is as thick as possible.



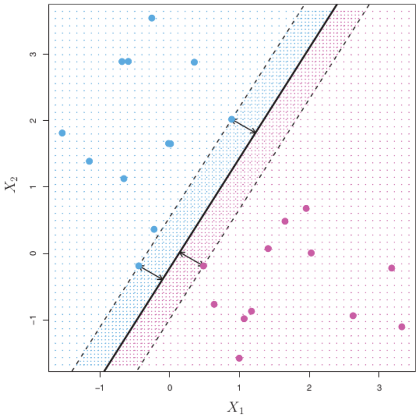
# Support vector machines

- Maximizing the margin means choosing the separating hyperplane so that the sandwich around it containing no data is as thick as possible.
- (The texts emphasize that this provides a unique solution. But why should we care?)



# Support vector machines

- Maximizing the margin means choosing the separating hyperplane so that the sandwich around it containing no data is as thick as possible.
- (The texts emphasize that this provides a unique solution. But why should we care?)
- Much more importantly, we can expect that a large margin between training examples and the decision boundary will lead to good separation on the test data. (And there are theorems that show this.)



# Support vector machines

For linearly separable data, we can find a $\theta$ such that for all $i$,

$$y^i \theta \cdot x^i > 0.$$

# Support vector machines

For linearly separable data, we can find a $\theta$ such that for all $i$,

$$y^i \theta \cdot x^i > 0.$$

Let $\delta$ be the smallest of these values, so that for all $i$,

$$y^i \theta \cdot x^i \geq \delta.$$

# Support vector machines

For linearly separable data, we can find a $\theta$ such that for all $i$,

$$y^i \theta \cdot x^i > 0.$$

Let $\delta$ be the smallest of these values, so that for all $i$,

$$y^i \theta \cdot x^i \geq \delta.$$

Then we can choose a new value of $\theta$ (scaled by $1/\delta$) so that, for all $i$,

$$y^i \theta \cdot x^i \geq 1.$$

# Support vector machines

For linearly separable data, we can find a $\theta$ such that for all $i$,

$$y^i \theta \cdot x^i > 0.$$

Let $\delta$ be the smallest of these values, so that for all $i$,

$$y^i \theta \cdot x^i \geq \delta.$$

Then we can choose a new value of $\theta$ (scaled by $1/\delta$) so that, for all $i$,

$$y^i \theta \cdot x^i \geq 1.$$

With this scaling of $\theta$, the margin is

$$\min_i y^i \frac{\theta \cdot x^i}{\|\theta\|} = \frac{1}{\|\theta\|}.$$

# Support vector machines

**Maximizing the margin:**

$$\max_{\theta} \quad \frac{1}{\|\theta\|}$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)$$

# Support vector machines

## Maximizing the margin:

$$
\begin{aligned}
\min_{\theta} \quad & \|\theta\| \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)
\end{aligned}
$$

# Support vector machines

**Maximizing the margin: Quadratic Program**

$$
\begin{aligned}
\min_{\theta} \quad & \|\theta\|^2 \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)
\end{aligned}
$$

# Support vector machines

**Maximizing the margin: Quadratic Program**

$$\min_{\theta} \quad \|\theta\|^2$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)$$

- A quadratic program (linear constraints and a quadratic criterion) is a convex optimization problem.

# Support vector machines

**Maximizing the margin: Quadratic Program**

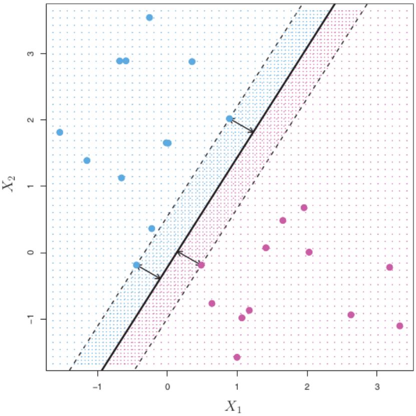
$$\min_{\theta} \quad \|\theta\|^2$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)$$

- A quadratic program (linear constraints and a quadratic criterion) is a convex optimization problem.
- There are efficient algorithms for solving QPs.

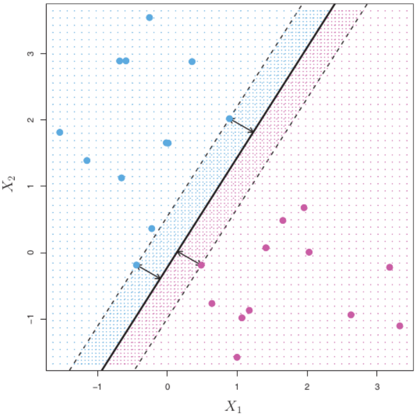
# Support vector machines

- The points that satisfy these constraints with equality (*active constraints*) are called *support vectors*.



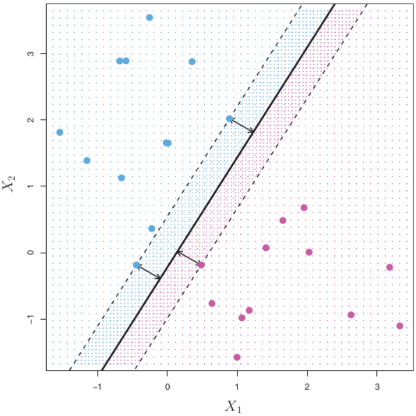
# Support vector machines

- The points that satisfy these constraints with equality (*active constraints*) are called *support vectors*.
- (Sufficiently small) changes to any other data points will not affect the maximal margin hyperplane.



# Support vector machines

- The points that satisfy these constraints with equality (*active constraints*) are called *support vectors*.
- (Sufficiently small) changes to any other data points will not affect the maximal margin hyperplane.
- We can think of the set of support vectors as a *compressed* version of the training data.



# Support vector machines: only uses inner products

## Properties:

- $\theta = \sum_i \alpha^i y^i x^i$.
  ($\alpha^i \neq 0$ only for support vectors.)

# Support vector machines: only uses inner products

## Properties:

- $\theta = \sum_i \alpha^i y^i x^i$.
  ($\alpha^i \neq 0$ only for support vectors.)
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x,$$

$$\|\theta\|^2 = \sum_{i,j} \alpha^i \alpha^j y^i y^j x^i \cdot x^j.$$

# Support vector machines: only uses inner products

## Properties:

- $\theta = \sum_i \alpha^i y^i x^i$.
  ($\alpha^i \neq 0$ only for support vectors.)
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x,$$

$$\|\theta\|^2 = \sum_{i,j} \alpha^i \alpha^j y^i y^j x^i \cdot x^j.$$

- So we can work with data in any inner product space (not just finite-dimensional vectors).

# Support vector machines

**Hard margin SVM**

$$\begin{aligned} \min_{\theta} \quad & \|\theta\|^2 \\ \text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n) \end{aligned}$$

- What if the data are not linearly separable?
  There will not be any feasible value of $\theta$.
  (Cannot satisfy all of the inequalities $y^i \theta \cdot x^i \geq 1$.)

# Support vector machines

**Hard margin SVM**

$$\min_{\theta} \quad \|\theta\|^2$$
$$\text{s.t.} \quad y^i\theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)$$

- What if the data are not linearly separable?
  There will not be any feasible value of $\theta$.
  (Cannot satisfy all of the inequalities $y^i\theta \cdot x^i \geq 1$.)
- We can relax these inequalities $y^i\theta \cdot x^i \geq 1$:

# Support vector machines

**Hard margin SVM**

$$\begin{array}{lll} \min_{\theta} & \|\theta\|^2 & \\ \text{s.t.} & y^i\theta \cdot x^i \geq 1 & (i = 1, \ldots, n) \end{array}$$

- What if the data are not linearly separable?
  There will not be any feasible value of $\theta$.
  (Cannot satisfy all of the inequalities $y^i\theta \cdot x^i \geq 1$.)
- We can relax these inequalities $y^i\theta \cdot x^i \geq 1$:
  - Introduce slack variables: $\xi_i \geq 0$,

# Support vector machines

**Hard margin SVM**

$$\begin{array}{lll} \min\limits_{\theta} & \|\theta\|^2 & \\ \text{s.t.} & y^i \theta \cdot x^i \geq 1 & (i = 1, \ldots, n) \end{array}$$

- What if the data are not linearly separable?
  There will not be any feasible value of $\theta$.
  (Cannot satisfy all of the inequalities $y^i \theta \cdot x^i \geq 1$.)
- We can relax these inequalities $y^i \theta \cdot x^i \geq 1$:
  - Introduce slack variables: $\xi_i \geq 0$,
  - Enforce the milder constraint: $y^i \theta \cdot x^i \geq 1 - \xi_i$,

# Support vector machines

## Hard margin SVM

$$\begin{array}{lll} \min\limits_{\theta} & \|\theta\|^2 & \\ \text{s.t.} & y^i\theta\cdot x^i \geq 1 & (i = 1, \ldots, n) \end{array}$$

- What if the data are not linearly separable?
  There will not be any feasible value of $\theta$.
  (Cannot satisfy all of the inequalities $y^i\theta\cdot x^i \geq 1$.)
- We can relax these inequalities $y^i\theta\cdot x^i \geq 1$:
  - Introduce slack variables: $\xi_i \geq 0$,
  - Enforce the milder constraint: $y^i\theta\cdot x^i \geq 1 - \xi_i$,
  - Add the $\xi_i$ to the criterion.

# Support vector machines

**Hard margin SVM** **Quadratic Program**

$$\min_{\theta} \quad \|\theta\|^2$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1$$

$$(i = 1, \ldots, n)$$

### Hard→Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1$$

$$(i = 1, \ldots, n)$$

# Support vector machines

## Hard→Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2$$

$$\text{s.t.} \quad y^i \theta \cdot x^i \geq 1 - \xi_i$$

$$(i = 1, \ldots, n)$$

# Support vector machines

## Hard→Soft margin SVM

$$
\begin{aligned}
\min_{\theta} \quad & \|\theta\|^2 \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 - \xi_i \\
& \xi_i \geq 0 \qquad\qquad (i = 1, \ldots, n)
\end{aligned}
$$

# Support vector machines

## Hard→Soft margin SVM

$$
\begin{aligned}
\min_{\theta} \quad & \|\theta\|^2 + \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 - \xi_i \\
& \xi_i \geq 0 \qquad\qquad (i = 1, \ldots, n)
\end{aligned}
$$

### Hard→Soft margin SVM

$$\begin{aligned} \min_{\theta} \quad & \|\theta\|^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad\qquad (i = 1, \ldots, n) \end{aligned}$$

# Support vector machines

**Soft margin SVM** — **Quadratic Program**

$$
\begin{aligned}
\min_{\theta} \quad & \|\theta\|^2 + C\sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 - \xi_i \\
& \xi_i \geq 0 \qquad\qquad (i = 1, \ldots, n)
\end{aligned}
$$

# Support vector machines

**Soft margin SVM** — **Quadratic Program**

$$\begin{aligned}
\min_{\theta} \quad & \|\theta\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y^i \theta \cdot x^i \geq 1 - \xi_i \\
& \xi_i \geq 0 \qquad\qquad (i = 1, \ldots, n)
\end{aligned}$$

- The parameter $C$ adjusts the trade-off: $\|\theta\|^2$ versus fit to the data.

**Soft margin SVM** | **Quadratic Program**

$$\begin{aligned} \min_{\theta} \quad & \|\theta\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y^i\theta\cdot x^i \geq 1-\xi_i \\ & \xi_i \geq 0 \qquad\qquad\qquad (i=1,\ldots,n) \end{aligned}$$

- The parameter $C$ adjusts the trade-off: $\|\theta\|^2$ versus fit to the data.
- The inequalities imply $\xi_i = \left(1 - y^i\theta\cdot x^i\right)_+$.

# Support vector machines

## Soft margin SVM

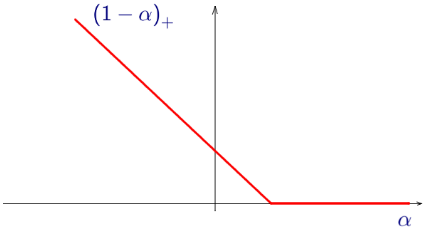
$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta \cdot x^i\right)_+ .$$

# Support vector machines

## Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta\cdot x^i\right)_+ .$$

## SVM margin cost function:

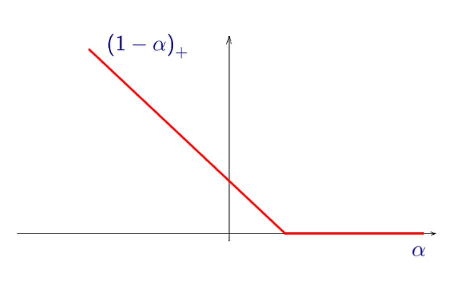


# Support vector machines

## Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta\cdot x^i\right)_+ .$$

## SVM margin cost function:



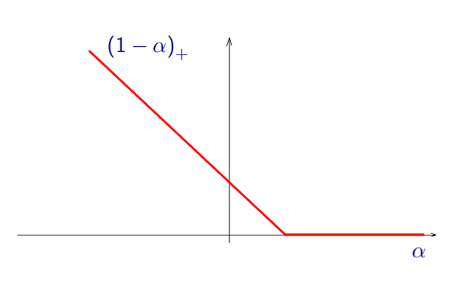

- Convex relaxation (upper bound) for misclassifications:

# Support vector machines

## Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta\cdot x^i\right)_+ .$$

## SVM margin cost function:




- Convex relaxation (upper bound) for misclassifications:

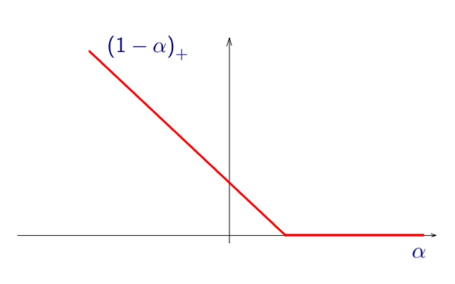
$$1[\alpha \le 0] = \begin{cases} 1 & \text{if } \alpha \le 0, \\ 0 & \text{otherwise.} \end{cases}$$

# Support vector machines

## Soft margin SVM

$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta \cdot x^i\right)_+ .$$

## SVM margin cost function:

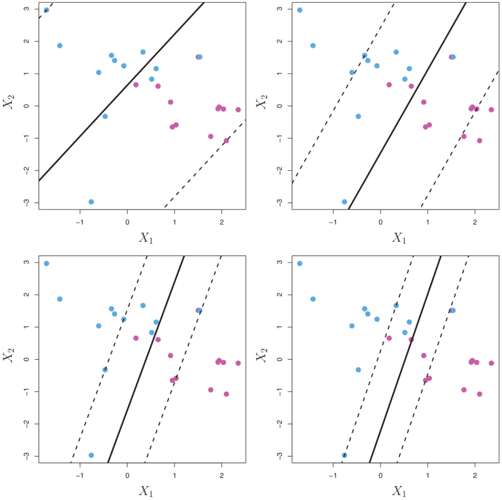



- Convex relaxation (upper bound) for misclassifications:

$$1[\alpha \le 0] = \begin{cases} 1 & \text{if } \alpha \le 0, \\ 0 & \text{otherwise.} \end{cases}$$
$$\le (1-\alpha)_+ .$$

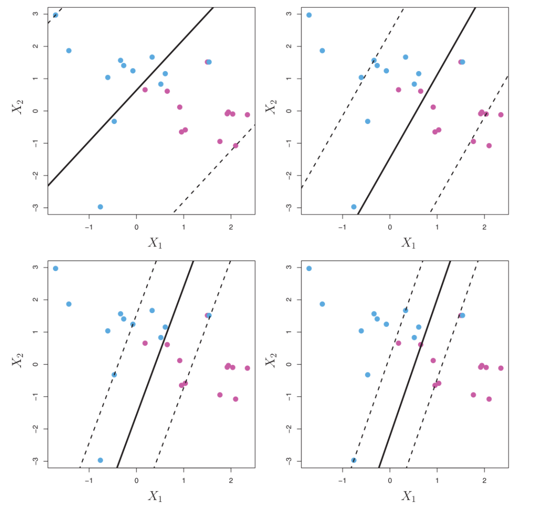
# Support vector machines

## Varying $C$:



# Support vector machines

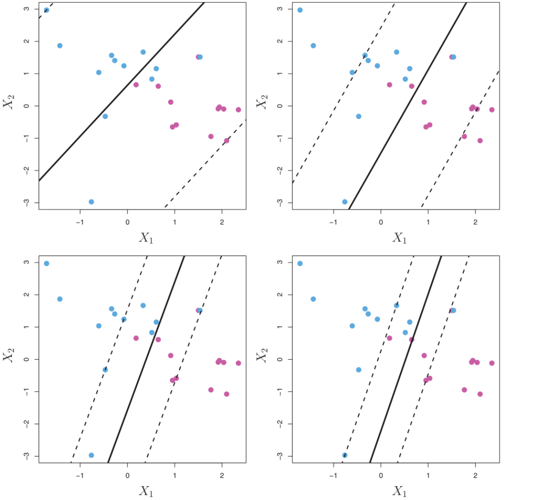
## Varying $C$:




## Choosing $C$:

# Support vector machines

## Varying $C$:

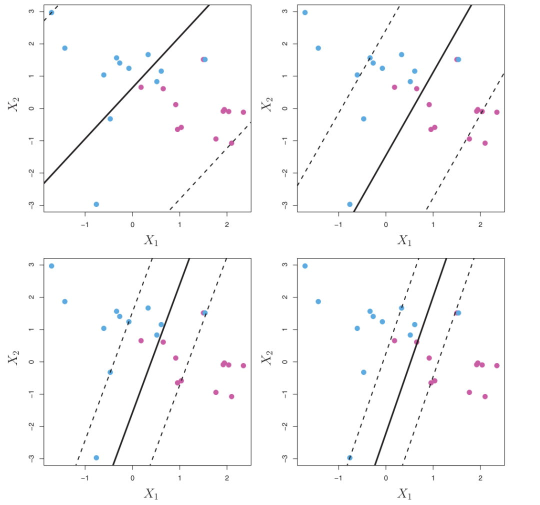



## Choosing $C$:

- *Regularization* parameter.

# Support vector machines

## Varying $C$:

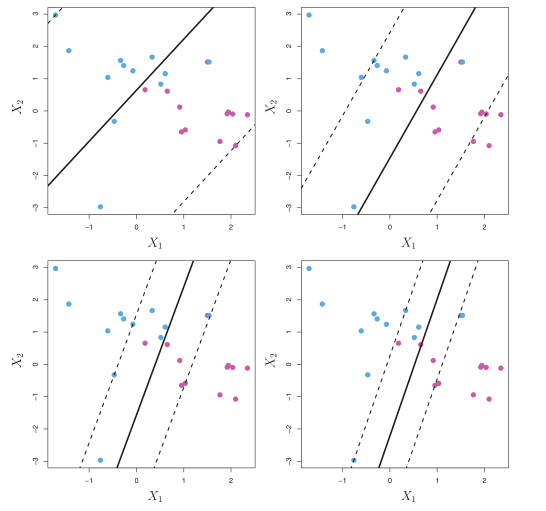



## Choosing $C$:

- *Regularization* parameter.
- Small $C \to$ small $\|\theta\|^2$.

# Support vector machines

## Varying $C$:

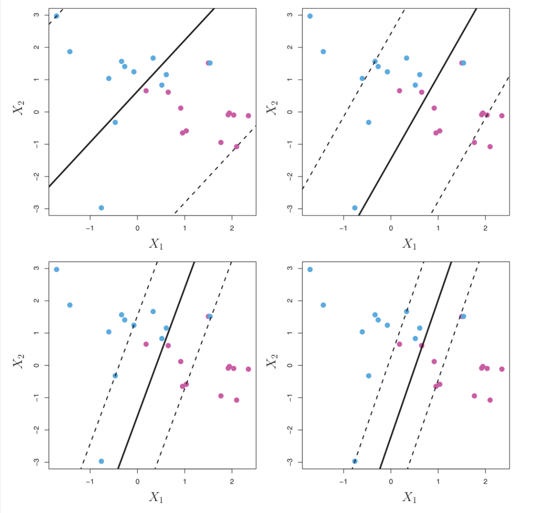



## Choosing $C$:

- *Regularization* parameter.
- Small $C$ $\rightarrow$ small $\|\theta\|^2$.
- Large $C$ $\rightarrow$ small misclassification term.

# Support vector machines

## Varying $C$:

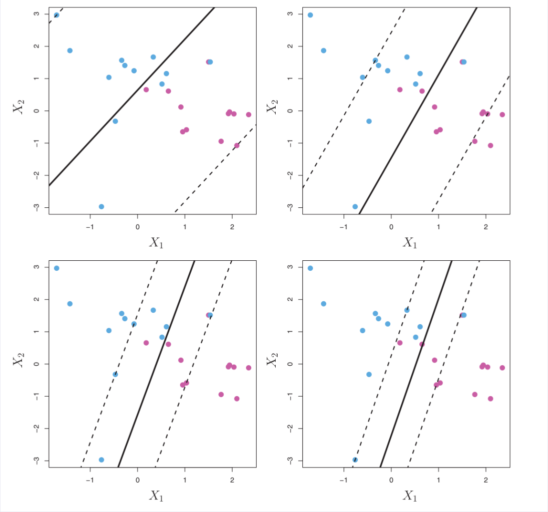



## Choosing $C$:

- *Regularization* parameter.
- Small $C$ $\rightarrow$ small $\|\theta\|^2$.
- Large $C$ $\rightarrow$ small misclassification term.
- Choose $C$ using cross-validation.

# Support vector machines

## Varying $C$:




## Choosing $C$:

- *Regularization* parameter.
- Small $C$ $\to$ small $\|\theta\|^2$.
- Large $C$ $\to$ small misclassification term.
- Choose $C$ using cross-validation.
- Alternative formulations allow a more intuitive parameterization: $\nu$-SVM. Parameter $\nu$ is (roughly) proportion of support vectors.

# Soft margin SVM: only uses inner products

## Properties:

- We'll see that $\theta = \sum_i \alpha^i y^i x^i$.

# Soft margin SVM: only uses inner products

## Properties:

- We'll see that $\theta = \sum_i \alpha^i y^i x^i$.
- $\alpha^i \neq 0$ only for *support vectors*, which satisfy $y^i \theta \cdot x^i \leq 1$.

# Soft margin SVM: only uses inner products

## Properties:

- We'll see that $\theta = \sum_i \alpha^i y^i x^i$.
- $\alpha^i \neq 0$ only for *support vectors*, which satisfy $y^i \theta \cdot x^i \leq 1$.
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x,$$

$$\|\theta\|^2 = \sum_{i,j} \alpha^i \alpha^j y^i y^j x^i \cdot x^j.$$

# Soft margin SVM: only uses inner products

## Properties:

- We'll see that $\theta = \sum_i \alpha^i y^i x^i$.
- $\alpha^i \neq 0$ only for *support vectors*, which satisfy $y^i \theta \cdot x^i \leq 1$.
- The only properties of the data that we use are inner products:

$$\theta \cdot x = \sum_i \alpha^i y^i x^i \cdot x,$$

$$\|\theta\|^2 = \sum_{i,j} \alpha^i \alpha^j y^i y^j x^i \cdot x^j.$$

- So we can work with data in any inner product space (not just finite-dimensional vectors).

# Linear Classifiers: Outline for today

- Properties of linear classifiers.
- Finding a good separating hyperplane.
  - The perceptron algorithm (Rosenblatt, 1950s).
  - The perceptron convergence theorem.
  - The perceptron algorithm as a stochastic gradient method.
  - Support vector machines (Vapnik, 1990s).
  - Hard margin SVM.
  - Soft margin SVM.
  - Both are quadratic programs.
  - Only use inner products.