CS189/CS289A
Introduction to Machine Learning
Lecture 3: Support Vector Machines

Peter Bartlett

January 27, 2015

Outline

Outline

- Recall: linear classifiers, perceptron algorithm

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting

Outline

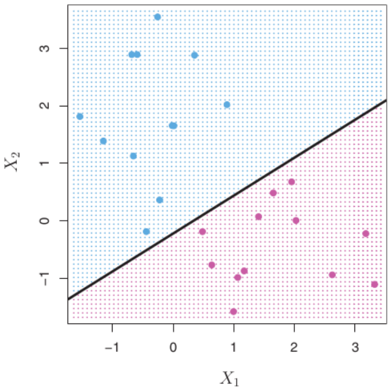
- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting
- Kernels

Outline

- **Recall: linear classifiers, perceptron algorithm**
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting
- Kernels

Linear classifiers

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

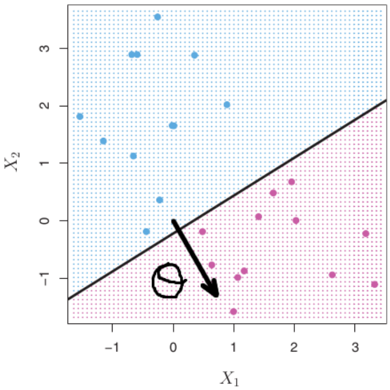


Linear classifiers

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

$$f(x) = \theta \cdot x + \theta_0,$$

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

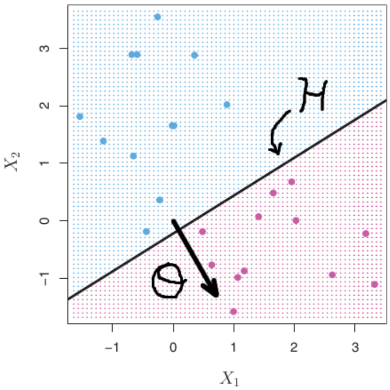


Linear classifiers

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

$$f(x) = \theta \cdot x + \theta_0,$$

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



Decision boundary:

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

Methods for choosing linear classifiers

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)
- How do we use the data to choose a linear classifier?

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)
- How do we use the data to choose a linear classifier?
- We consider three approaches:

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \dots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \dots, 9\}$.)
- How do we use the data to choose a linear classifier?
- We consider three approaches:
  - The perceptron algorithm.

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)
- How do we use the data to choose a linear classifier?
- We consider three approaches:
  - The perceptron algorithm.
  - The hard margin support vector machine.

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)
- How do we use the data to choose a linear classifier?
- We consider three approaches:
  - The perceptron algorithm.
  - The hard margin support vector machine.
  - The (soft margin) support vector machine.

Methods for choosing linear classifiers

- Suppose we have a training set $(x_1, y_1), \ldots, (x_n, y_n)$.
  (For example, the $x_i \in \mathbb{R}^{400}$ might be grey-scale images of digits and the y_i their class label $y_i \in \{0, 1, \ldots, 9\}$.)
- How do we use the data to choose a linear classifier?
- We consider three approaches:
  - The perceptron algorithm.
  - The hard margin support vector machine.
  - The (soft margin) support vector machine.
- (Later, we'll also look at other methods, including logistic regression and linear discriminant analysis.)

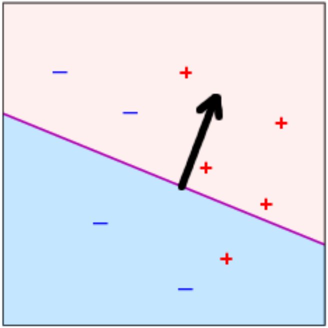
Perceptron algorithm

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 θ.

Perceptron algorithm:

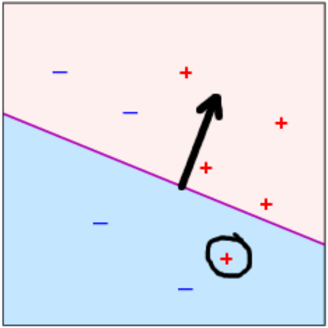
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified (x^i, y^i)
    $\theta \leftarrow \theta + y^i x^i$
Return θ.



Perceptron algorithm

Perceptron algorithm:

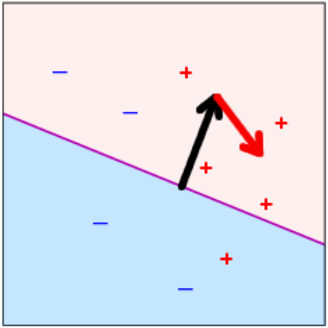
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified (x^i, y^i)
    $\theta \leftarrow \theta + y^i x^i$
Return θ.



Perceptron algorithm

Perceptron algorithm:

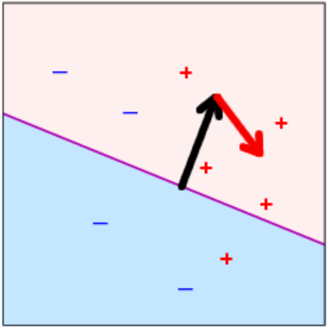
Input: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
while some $y^i \neq \mathrm{sign}(\theta \cdot x^i)$
    pick some misclassified (x^i, y^i)
    $\theta \leftarrow \theta + y^i x^i$
Return θ.

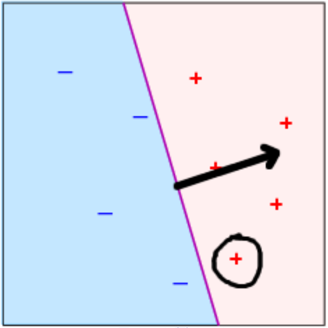


Perceptron algorithm

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 θ.





Perceptron algorithm

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

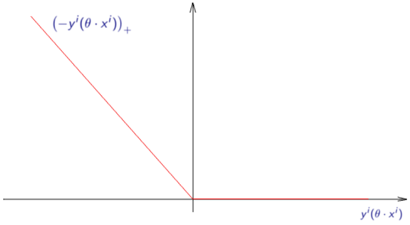
Perceptron algorithm

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

Margin cost function

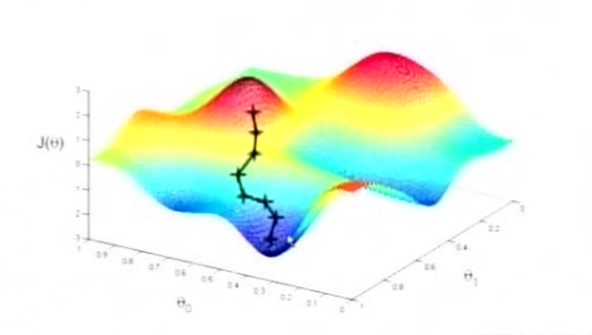
$$J(\theta) = \sum_i \left(-y^i(\theta \cdot x^i)\right)_+$$

$J(\theta) = 0 \Rightarrow$ all x^i classified correctly.



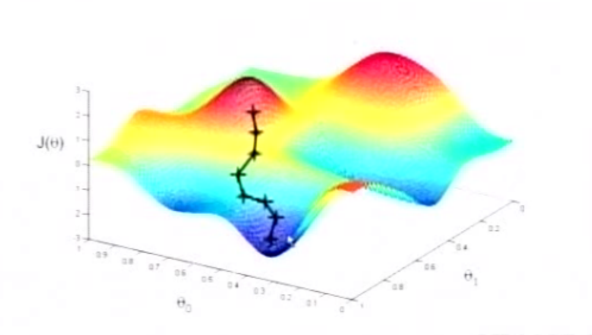
Perceptron algorithm

Gradient descent



Perceptron algorithm

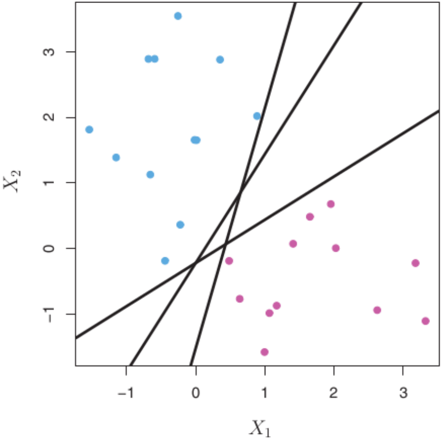
Gradient descent



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

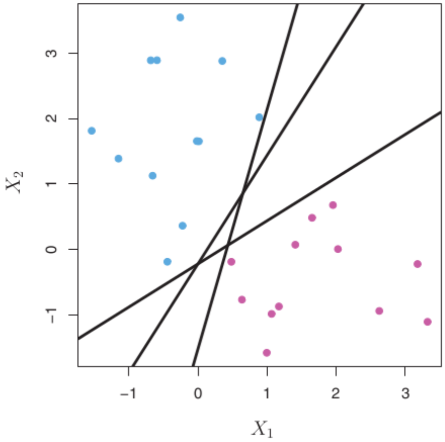
Support vector machines

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



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

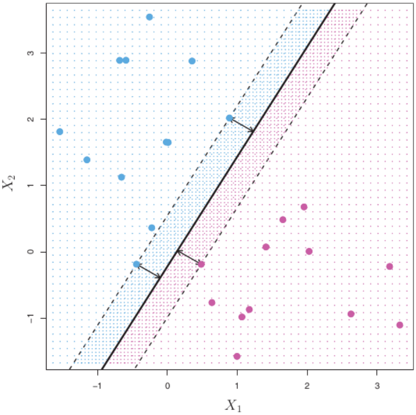
$$\text{distance} := y^i \frac{\theta \cdot x^i}{\|\theta\|}.$$



Support vector machines

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

$$\text{distance} := y^i \frac{\theta \cdot x^i}{\|\theta\|}.$$



Support vector machines

Maximizing the margin:

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

Support vector machines

Maximizing the margin:

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

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

- This is a *quadratic program*: a minimization problem involving a convex quadratic criterion, subject to linear constraints).

Support vector machines

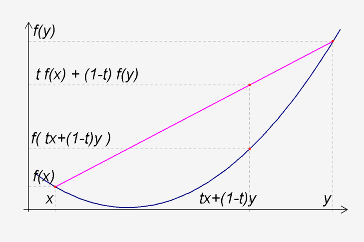
Maximizing the margin:

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

- This is a *quadratic program*: a minimization problem involving a convex quadratic criterion, subject to linear constraints).
- There are efficient algorithms for solving QPs.

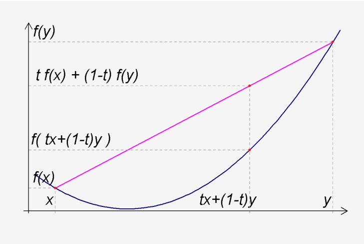
Convex vs non-convex minimization

Convex vs non-convex minimization

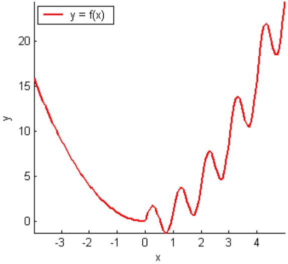


A convex function

Convex vs non-convex minimization

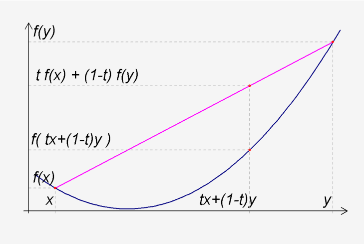


A convex function

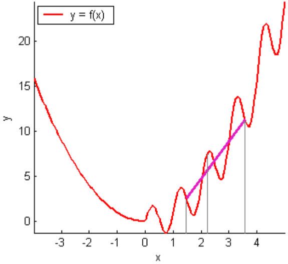


A non-convex function

Convex vs non-convex minimization

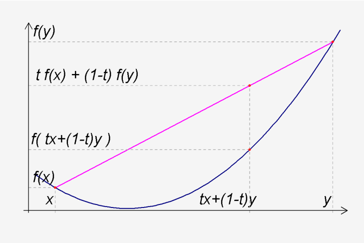


A convex function

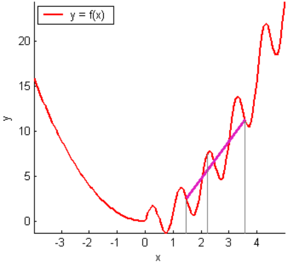


A non-convex function

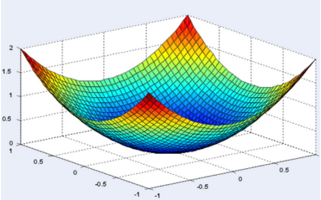
Convex vs non-convex minimization



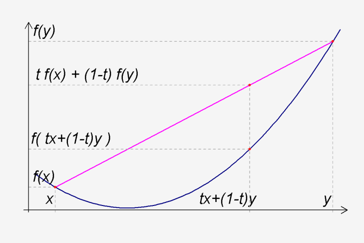
A convex function



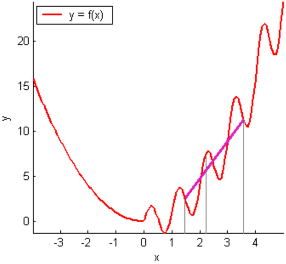
A non-convex function



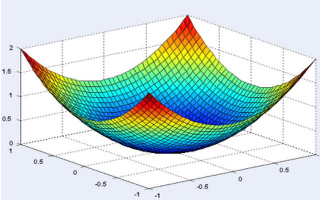
Convex vs non-convex minimization

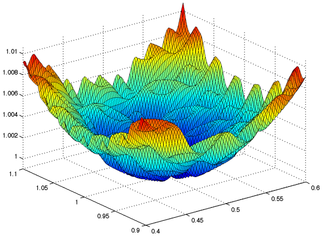


A convex function



A non-convex function





Support vector machines

Hard margin SVM

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

Support vector machines

Hard margin SVM

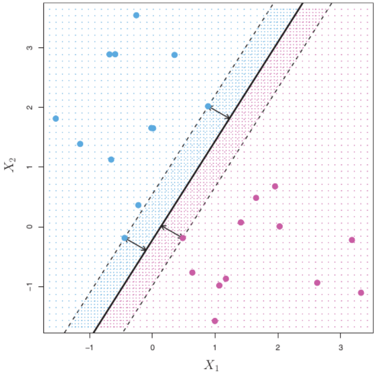
$$\min_{\theta} \quad \|\theta\|^2$$
$$\text{such that} \quad y^i\theta \cdot x^i \geq 1 \qquad (i = 1, \ldots, n)$$

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

Hard margin SVM

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

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

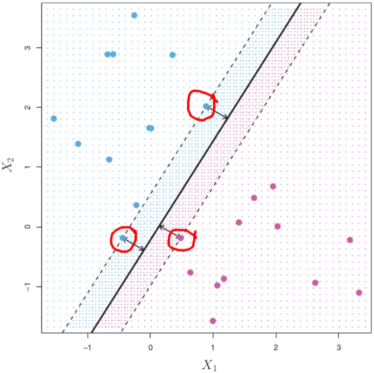


Support vector machines

Hard margin SVM

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

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

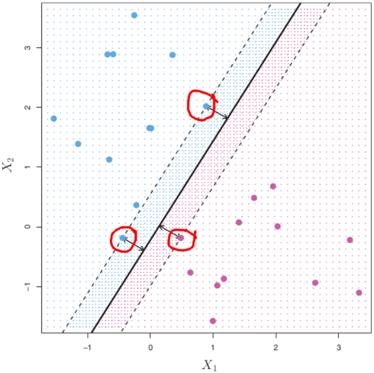


Support vector machines

Hard margin SVM

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

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

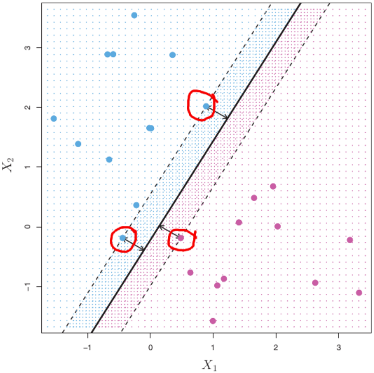


Support vector machines

Hard margin SVM

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

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

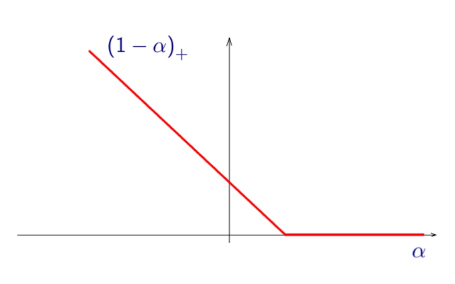
Soft margin SVM

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

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:

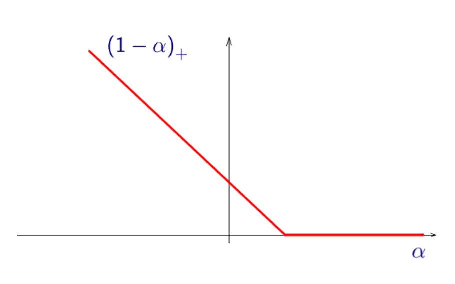


Support vector machines

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:



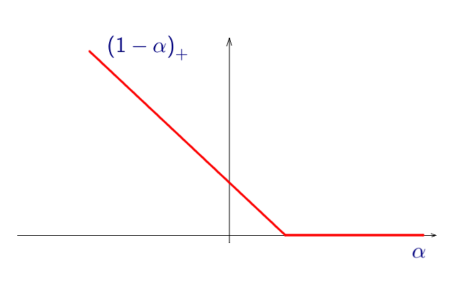
- Hard margin SVM: every term $(1 - y^i\theta\cdot x^i)_+ = 0$.

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:



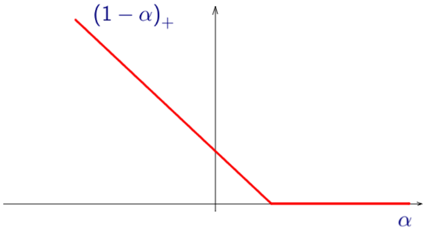
- Hard margin SVM: every term $(1 - y^i\theta\cdot x^i)_+ = 0$.
- Soft margin SVM: the constraints can be violated, but not too much.

Support vector machines

Soft margin SVM

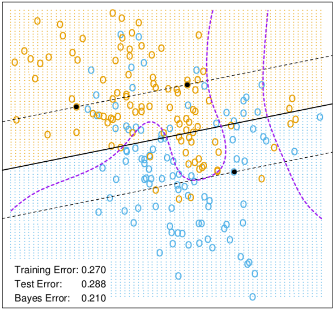
$$\min_{\theta} \quad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta\cdot x^i\right)_+ .$$

SVM margin cost function:



- Hard margin SVM: every term $(1 - y^i\theta\cdot x^i)_+ = 0$.
- Soft margin SVM: the constraints can be violated, but not too much.
- The parameter C adjusts the trade-off: $\|\theta\|^2$ versus fit to the data.

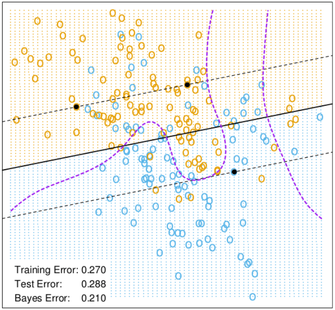
- Simulated data: (x, y) pairs chosen according to a known distribution.



$C = 10000$

Support vector machines

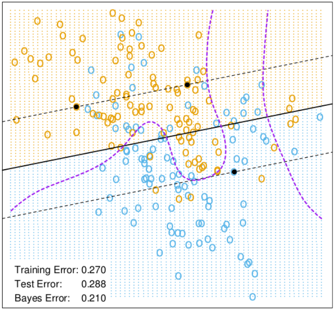
- Simulated data: (x, y) pairs chosen according to a known distribution.
- (Purple line is the optimal decision boundary—'Bayes classifier'. It minimizes the probability of misclassification.)



$C = 10000$

Support vector machines

- Simulated data: (x, y) pairs chosen according to a known distribution.
- (Purple line is the optimal decision boundary—'Bayes classifier'. It minimizes the probability of misclassification.)

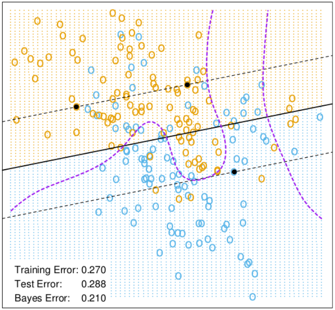


$C = 10000$

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

Support vector machines

- Simulated data: (x, y) pairs chosen according to a known distribution.
- (Purple line is the optimal decision boundary—'Bayes classifier'. It minimizes the probability of misclassification.)
- Solid black line is the SVM decision boundary $\{x : \theta \cdot x = 0\}$.

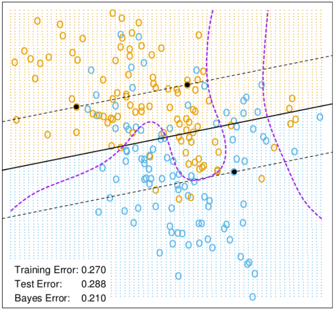


$C = 10000$

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

Support vector machines

- Simulated data: (x, y) pairs chosen according to a known distribution.
- (Purple line is the optimal decision boundary—'Bayes classifier'. It minimizes the probability of misclassification.)
- Solid black line is the SVM decision boundary $\{x : \theta \cdot x = 0\}$.
- Dashed black lines represent $\{x : \theta \cdot x \in \{-1, 1\}\}$.

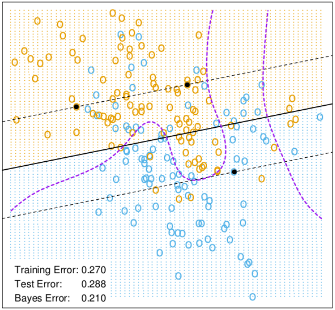


$C = 10000$

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

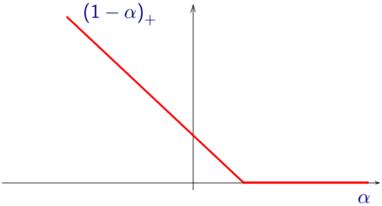
Support vector machines

- Simulated data: (x, y) pairs chosen according to a known distribution.
- (Purple line is the optimal decision boundary—'Bayes classifier'. It minimizes the probability of misclassification.)
- Solid black line is the SVM decision boundary $\{x : \theta \cdot x = 0\}$.
- Dashed black lines represent $\{x : \theta \cdot x \in \{-1, 1\}\}$.



$C = 10000$

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



Outline

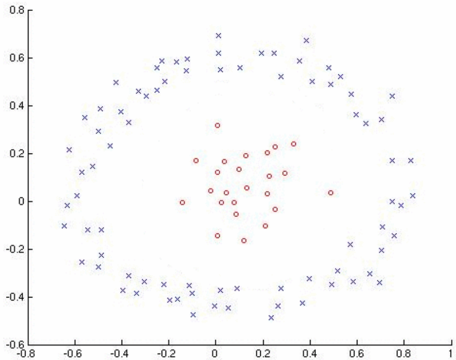
- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- **Features**
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting
- Kernels

Features

- What if a linear decision boundary performs poorly?

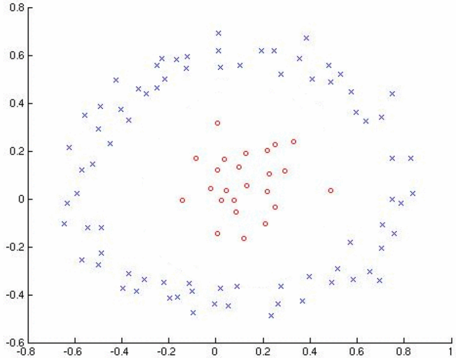
Features

- What if a linear decision boundary performs poorly?
- Consider this toy example.



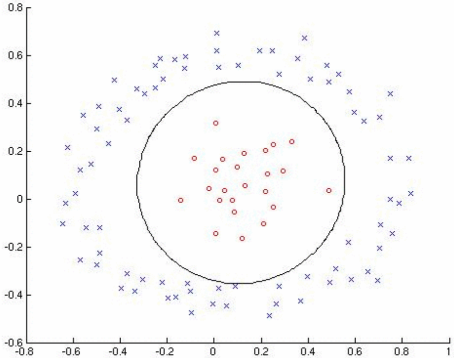
Features

- What if a linear decision boundary performs poorly?
- Consider this toy example.
- No linear decision boundary can effectively separate “o”s from “x”s.



Features

- What if a linear decision boundary performs poorly?
- Consider this toy example.
- No linear decision boundary can effectively separate "o"s from "x"s.
- But a quadratic decision boundary can separate them.



Features

Quadratic classifier

$f(x) = \|x - c\|^2 - r^2$ (center $c \in \mathbb{R}^2$, radius r)

Features

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

Quadratic classifier

$f(x) = \|x - c\|^2 - r^2$ (center $c \in \mathbb{R}^2$, radius r)

Features

Recall: Linear classifier

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

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

Quadratic classifier

$f(x) = \|x - c\|^2 - r^2$ (center $c \in \mathbb{R}^2$, radius r)

Features

Recall: Linear classifier

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

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

Quadratic classifier

$$\begin{aligned} f(x) &= \|x - c\|^2 - r^2 \qquad\qquad (\text{center } c \in \mathbb{R}^2, \text{ radius } r) \\ &= (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \end{aligned}$$

Features

Recall: Linear classifier

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

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

Quadratic classifier

$$\begin{aligned} f(x) &= \|x - c\|^2 - r^2 \qquad\qquad (\text{center } c \in \mathbb{R}^2, \text{ radius } r) \\ &= (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \\ &= x_1^2 - 2c_1x_1 + c_1^2 + x_2^2 - 2c_2x_2 + c_2^2 - r^2 \end{aligned}$$

Features

Recall: Linear classifier

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

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

Quadratic classifier

$$\begin{aligned} f(x) &= \|x - c\|^2 - r^2 \qquad \text{(center } c \in \mathbb{R}^2\text{, radius } r\text{)} \\ &= (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \\ &= x_1^2 - 2c_1x_1 + c_1^2 + x_2^2 - 2c_2x_2 + c_2^2 - r^2 \\ &= \begin{pmatrix} 1 \\ 1 \\ -2c_1 \\ -2c_2 \\ c_1^2 + c_2^2 - r^2 \end{pmatrix} \cdot \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ 1 \end{pmatrix} \end{aligned}$$

Features

Recall: Linear classifier

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

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

Quadratic classifier

$$\begin{aligned} f(x) &= \|x - c\|^2 - r^2 \qquad\qquad (\text{center } c \in \mathbb{R}^2, \text{ radius } r) \\ &= (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \\ &= x_1^2 - 2c_1x_1 + c_1^2 + x_2^2 - 2c_2x_2 + c_2^2 - r^2 \\ &= \begin{pmatrix} 1 \\ 1 \\ -2c_1 \\ -2c_2 \\ c_1^2 + c_2^2 - r^2 \end{pmatrix} \cdot \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ 1 \end{pmatrix} = \underbrace{\theta \cdot \phi(x)}_{\text{linear classifier}} \end{aligned}$$

Features

Recall: Linear classifier

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

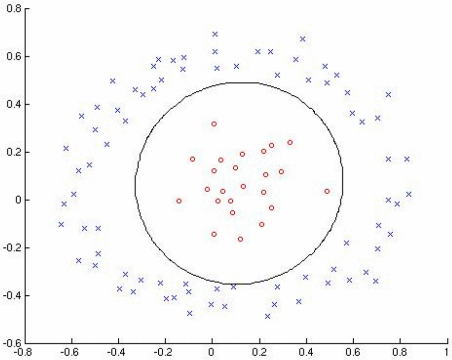
$$\hat{y} = \begin{cases} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{if } f(x) < 0. \end{cases}$$

Quadratic classifier

$$\begin{aligned} f(x) &= \|x - c\|^2 - r^2 \qquad\qquad (\text{center } c \in \mathbb{R}^2, \text{ radius } r) \\ &= (x_1 - c_1)^2 + (x_2 - c_2)^2 - r^2 \\ &= x_1^2 - 2c_1x_1 + c_1^2 + x_2^2 - 2c_2x_2 + c_2^2 - r^2 \\ &= \begin{pmatrix} 1 \\ 1 \\ -2c_1 \\ -2c_2 \\ c_1^2 + c_2^2 - r^2 \end{pmatrix} \cdot \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ 1 \end{pmatrix} = \underbrace{\theta \cdot \phi(x)}_{\text{linear classifier}} \qquad (\textit{features } \phi(x)) \end{aligned}$$

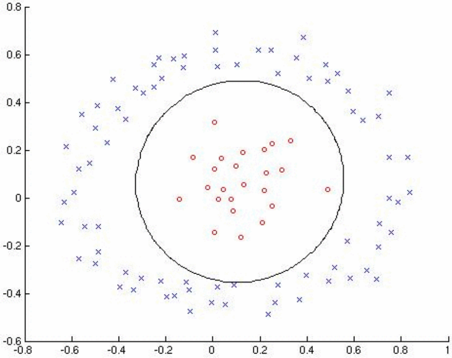
Features

- The quadratic decision boundary corresponds to a linear classifier with new features



Features

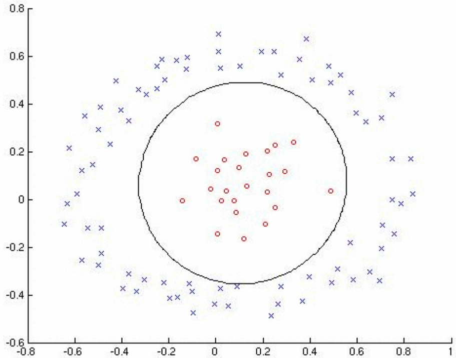
- The quadratic decision boundary corresponds to a linear classifier with new features, $\phi(x)$ instead of x:



Features

- The quadratic decision boundary corresponds to a linear classifier with new features, $\phi(x)$ instead of x:

$$f(x) = \theta \cdot \phi(x).$$

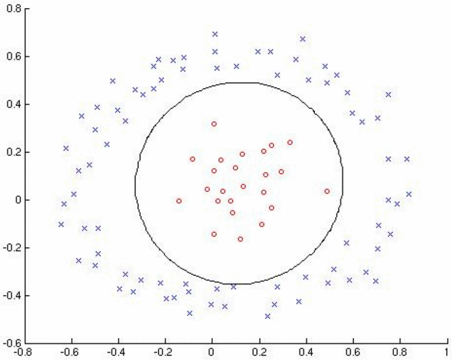


Features

- The quadratic decision boundary corresponds to a linear classifier with new features, $\phi(x)$ instead of x:

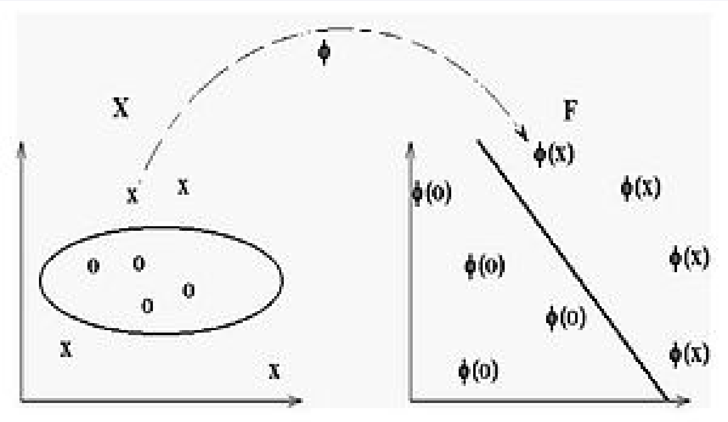
$$f(x) = \theta \cdot \phi(x).$$

- The features we choose are very important!



Features

Feature map ϕ



An aside

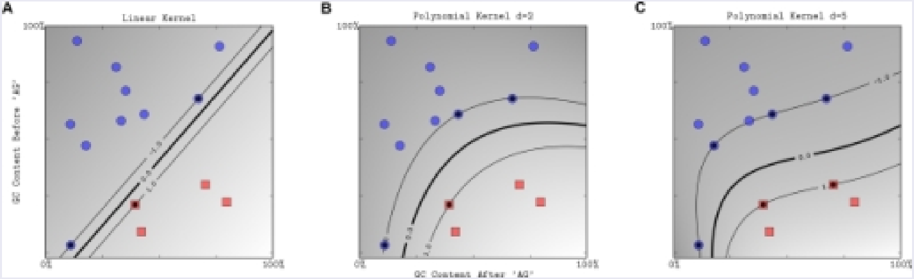
We used this idea earlier when we were simplifying notation to drop the offset θ_0: we transformed the pattern $x \in \mathbb{R}^d$ into a pattern $\tilde{x} \in \mathbb{R}^{d+1}$ with a constant component: to dispense with the offset θ_0:

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

Then

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

SVMs with polynomial features of various degrees:



degree $= 1$ degree $= 2$ degree $= 5$

Features

Features

- The features we choose are very important!

Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.

Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.
- This suggests we should make our set of features as rich as possible

Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.
- This suggests we should make our set of features as rich as possible: linear is a special case of quadratic is a special case of cubic...

Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.
- This suggests we should make our set of features as rich as possible: linear is a special case of quadratic is a special case of cubic... Why not include them all?

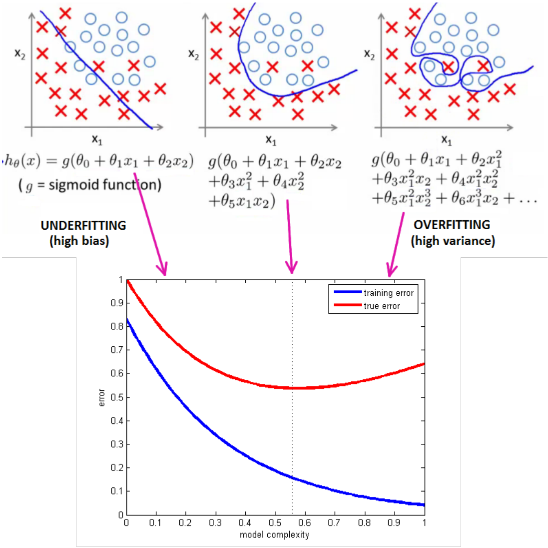
Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.
- This suggests we should make our set of features as rich as possible: linear is a special case of quadratic is a special case of cubic... Why not include them all?
- The richer the set of features, the more likely we will encounter *overfitting*.

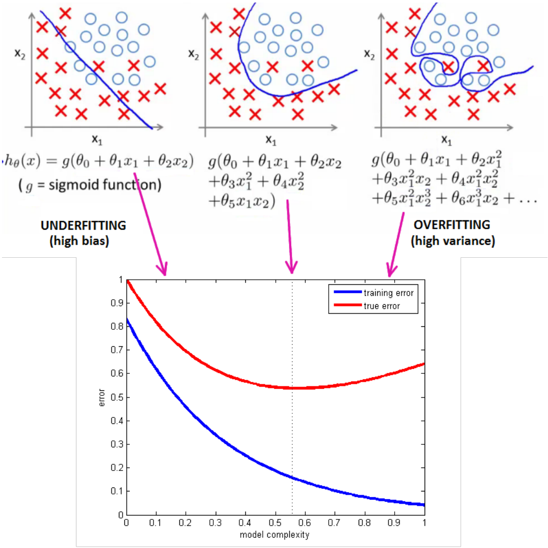
Features

- The features we choose are very important!
- We want them to be rich enough to accurately represent a good classifier.
- This suggests we should make our set of features as rich as possible: linear is a special case of quadratic is a special case of cubic... Why not include them all?
- The richer the set of features, the more likely we will encounter *overfitting*.
- It's a balancing act: we want our features to be as complex as necessary to represent the classifier, but no more complex.

Features and overfitting



Features and overfitting



What happens to this picture as sample size grows?

Features

Digit recognition

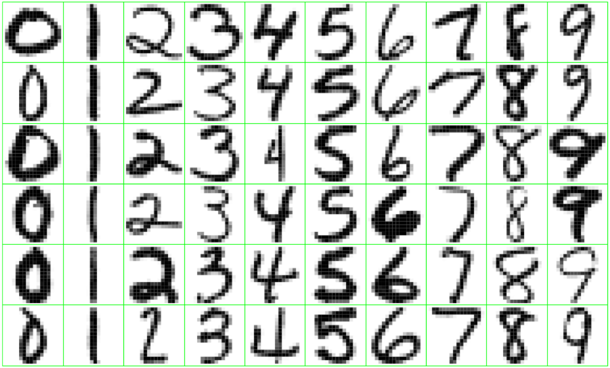


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Features

Digit recognition

- What features should we use?

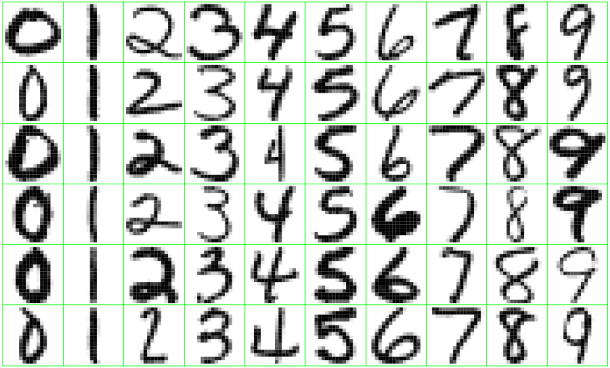


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Digit recognition

- What features should we use?
  - Grey scale level for each pixel

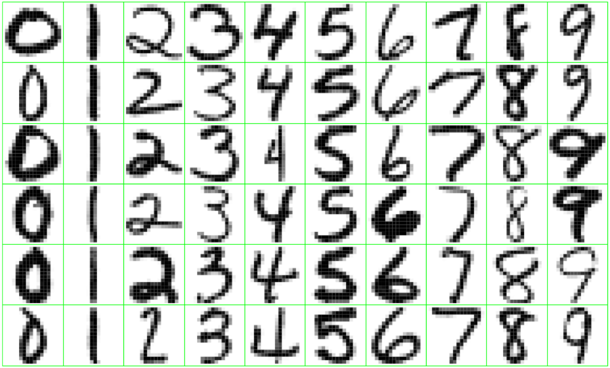


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Features

Digit recognition

- What features should we use?
  - Grey scale level for each pixel
  - Orientation histograms

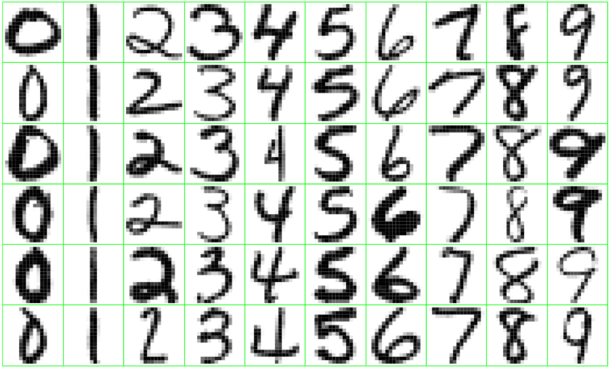


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Features

Digit recognition

- What features should we use?
  - Grey scale level for each pixel
  - Orientation histograms
  - Polynomials of these

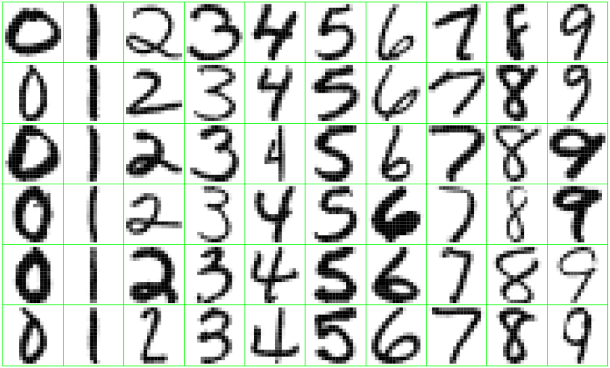
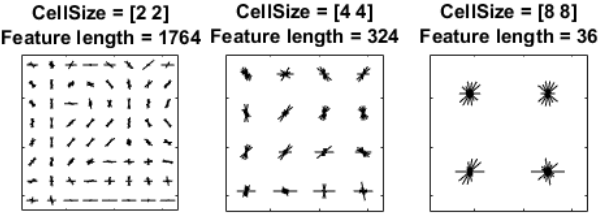
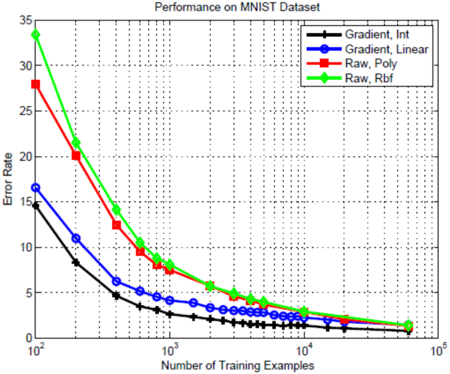


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Histograms of Oriented Gradients

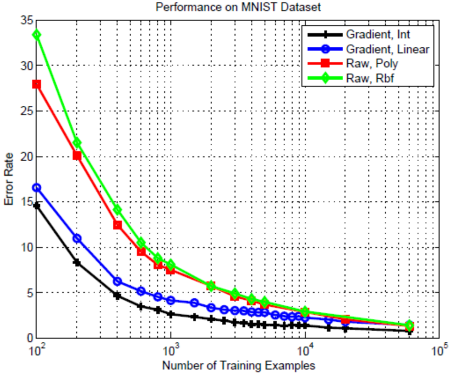


Support vector machines on MNIST data



(Maji and Malik, 2009)

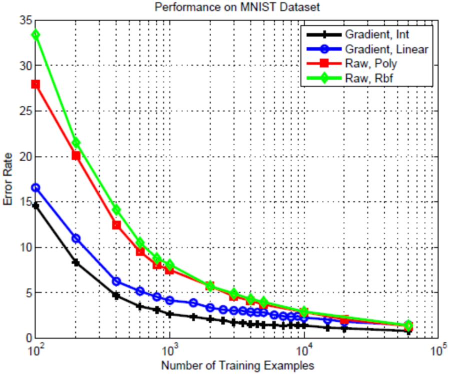
Support vector machines on MNIST data



(Maji and Malik, 2009)

- For large sample sizes, different features give similar performance.

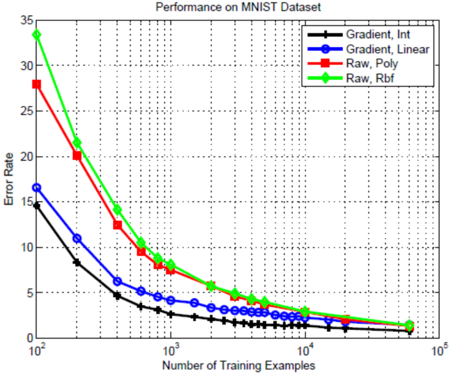
Support vector machines on MNIST data



(Maji and Malik, 2009)

- For large sample sizes, different features give similar performance.
- For smaller sample sizes, there are significant differences.

Support vector machines on MNIST data



(Maji and Malik, 2009)

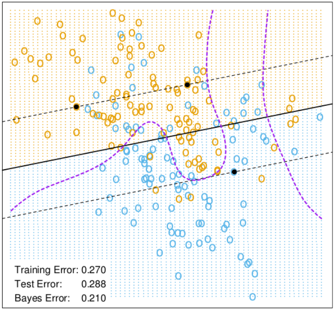
- For large sample sizes, different features give similar performance.
- For smaller sample sizes, there are significant differences.
- For some features, good classifiers are easier to find than for others.

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- **Role of the regularization parameter** C
- Regularization and overfitting
- Kernels

Regularization and SVMs

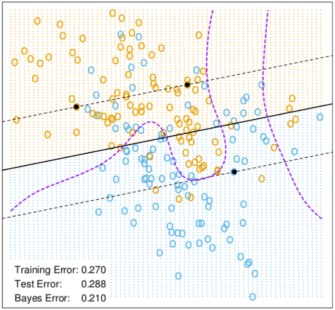
- Simulated data: (x, y) pairs chosen according to a known distribution.



$C = 10000$

Regularization and SVMs

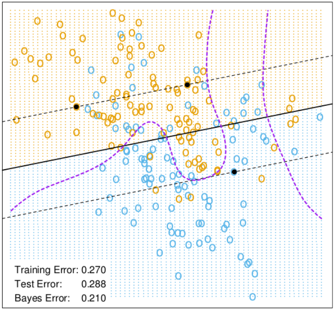
- Simulated data: (x, y) pairs chosen according to a known distribution.



$C = 10000$

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

- Simulated data: (x, y) pairs chosen according to a known distribution.
- Solid black line is the SVM decision boundary $\{x : \theta \cdot x = 0\}$.

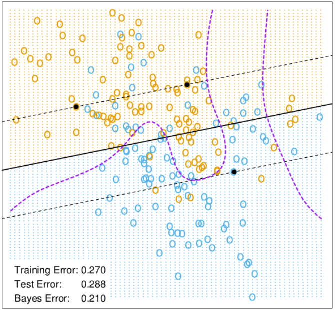


$C = 10000$

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

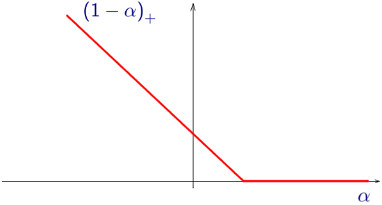
Regularization and SVMs

- Simulated data: (x, y) pairs chosen according to a known distribution.
- Solid black line is the SVM decision boundary $\{x : \theta \cdot x = 0\}$.
- Dashed black lines represent $\{x : \theta \cdot x \in \{-1, 1\}\}$.



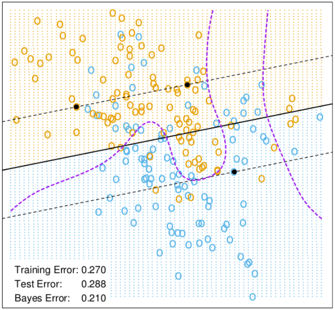
$C = 10000$

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



Regularization and SVMs

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

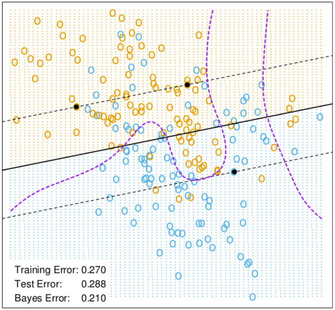


$C = 10000$

Regularization and SVMs

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

- C controls trade-off between margin $1/\|\theta\|$ and fit to data:

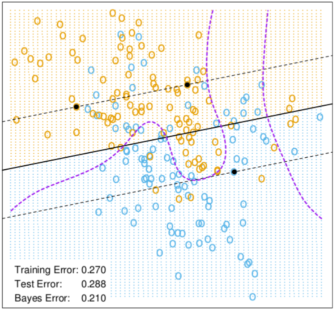


$C = 10000$

Regularization and SVMs

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

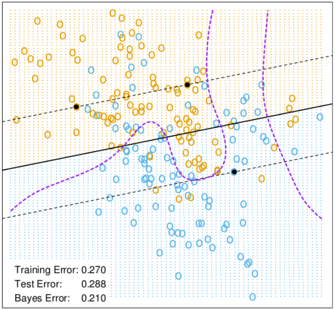
- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok).



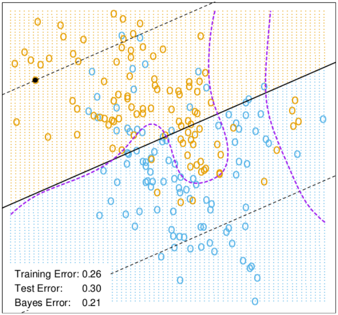
$C = 10000$

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

- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok).
- Small C: focus on large margin.



$C = 10000$



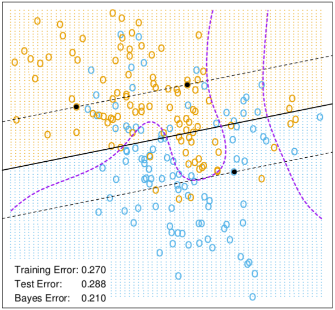
$C = 0.01$

Regularization and SVMs

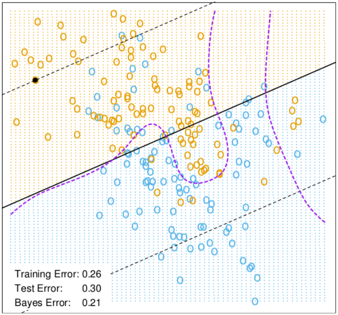
$$\min_\theta \qquad \|\theta\|^2 + C\sum_{i=1}^n \left(1 - y^i\theta \cdot x^i\right)_+.$$

- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok).
- Small C: focus on large margin.
- Overfitting increases with: less data, more features, C.

(Not apparent in this example with $d = 2$ and $n = 100$s.)



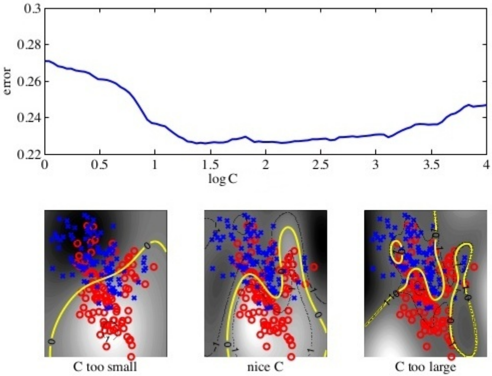
$C = 10000$



$C = 0.01$

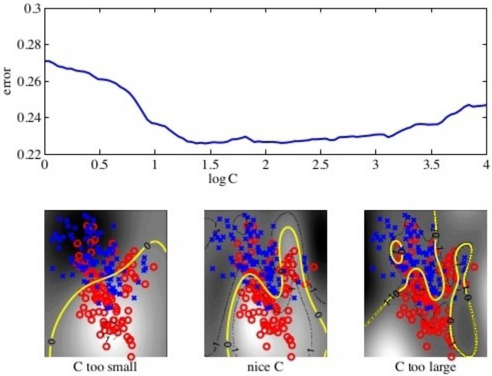
Regularization and SVMs

- Simulated data with *many features* $\phi(x)$.



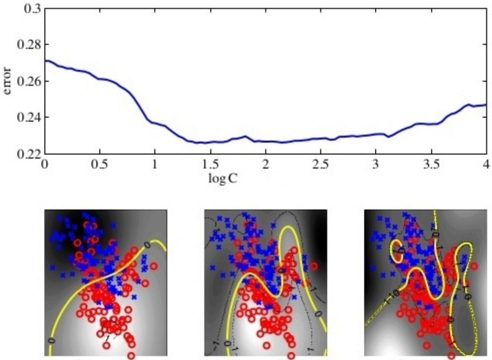
Regularization and SVMs

- Simulated data with *many features* $\phi(x)$.
- C controls trade-off between margin $1/\|\theta\|$ and fit to data:



Regularization and SVMs

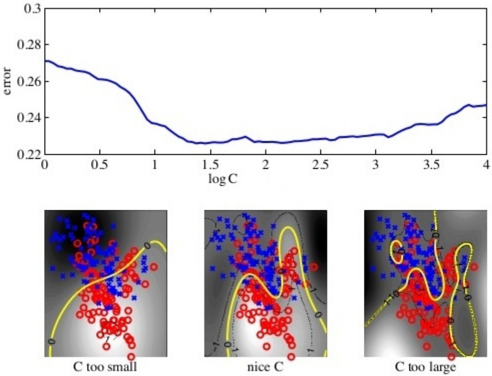
- Simulated data with *many features* $\phi(x)$.
- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok). More overfitting.



C too small   nice C   C too large

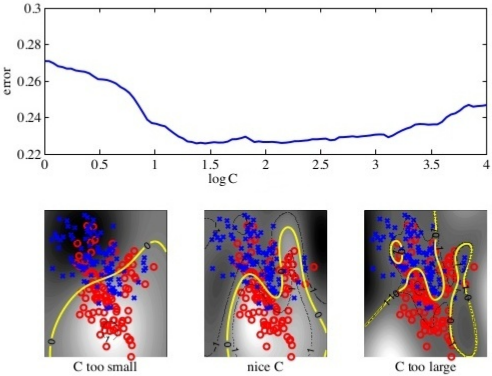
Regularization and SVMs

- Simulated data with *many features* $\phi(x)$.
- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok). More overfitting.
- Small C: focus on large margin. Less tendency to overfit.

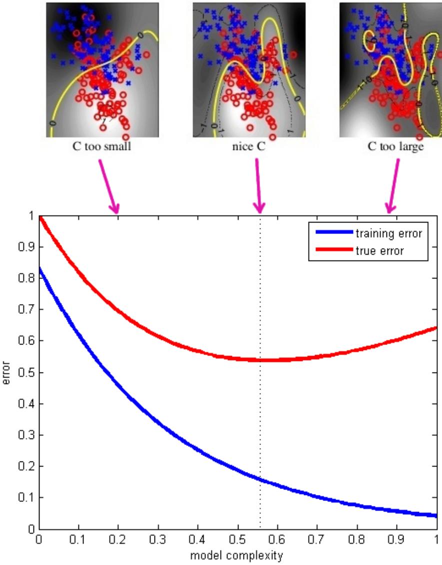


Regularization and SVMs

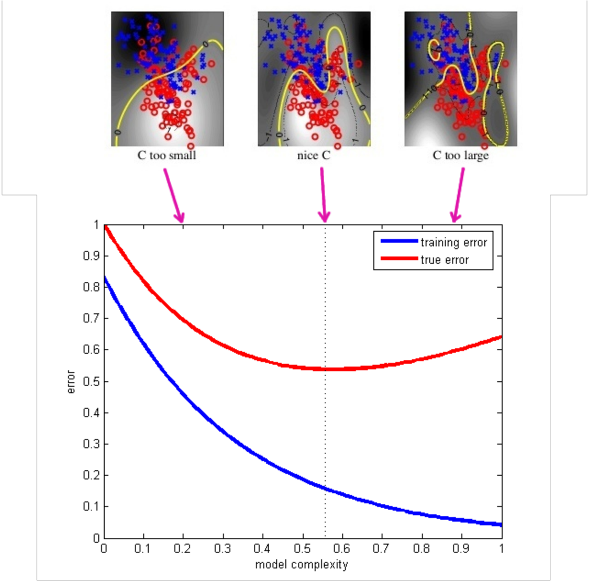
- Simulated data with *many features* $\phi(x)$.
- C controls trade-off between margin $1/\|\theta\|$ and fit to data:
- Large C: focus on fit to data (small margin ok). More overfitting.
- Small C: focus on large margin. Less tendency to overfit.
- Overfitting increases with: less data, more features.



Regularization and overfitting

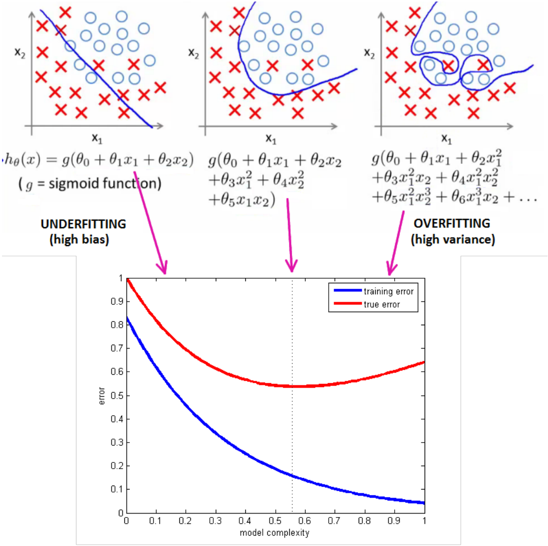


Regularization and overfitting



What happens to this picture as sample size grows?

c.f. Features and overfitting



Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting
- **Kernels**

Kernels

Kernels

- We've seen the benefits of appropriate choices of features.

Kernels

- We've seen the benefits of appropriate choices of features.
- *Kernels* give a simple way of working with large feature vectors.

Kernels

- We've seen the benefits of appropriate choices of features.
- *Kernels* give a simple way of working with large feature vectors.
- They are useful for methods that use only inner products between training vectors $x^i \cdot x^j$:

Kernels

- We've seen the benefits of appropriate choices of features.
- *Kernels* give a simple way of working with large feature vectors.
- They are useful for methods that use only inner products between training vectors $x^i \cdot x^j$:
  - Perceptron algorithm

Kernels

- We've seen the benefits of appropriate choices of features.
- *Kernels* give a simple way of working with large feature vectors.
- They are useful for methods that use only inner products between training vectors $x^i \cdot x^j$:
  - Perceptron algorithm
  - Hard margin SVM

Kernels

- We've seen the benefits of appropriate choices of features.
- *Kernels* give a simple way of working with large feature vectors.
- They are useful for methods that use only inner products between training vectors $x^i \cdot x^j$:
  - Perceptron algorithm
  - Hard margin SVM
  - Soft margin SVM

Perceptron algorithm: only uses inner products

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 θ.

Perceptron algorithm: only uses inner products

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 θ.

Properties:

Perceptron algorithm: only uses inner products

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 θ.

Properties:

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

Perceptron algorithm: only uses inner products

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 θ.

Properties:

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

$$\theta \cdot x^j = \left(\sum_i \alpha^i y^i x^i \right) \cdot x^j = \sum_i \alpha^i y^i \left(x^i \cdot x^j \right).$$

Perceptron algorithm: only uses inner products

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 θ.

Properties:

- $\theta = \sum_i \alpha^i y^i x^i$.
- The only properties of the data that we use are inner products:
$$\theta \cdot x^j = \left(\sum_i \alpha^i y^i x^i \right) \cdot x^j = \sum_i \alpha^i y^i \left(x^i \cdot x^j \right).$$
- As long as we can calculate the inner products $x^i \cdot x^j$ for training vectors x^i, x^j, that is all we need.

SVM: only uses inner products

Hard margin SVM

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

SVM: only uses inner products

Hard margin SVM

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

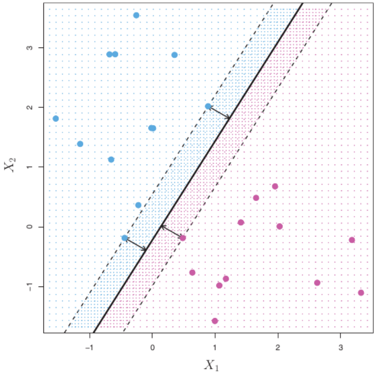
- The points that satisfy these constraints with equality are called *support vectors*.

SVM: only uses inner products

Hard margin SVM

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

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

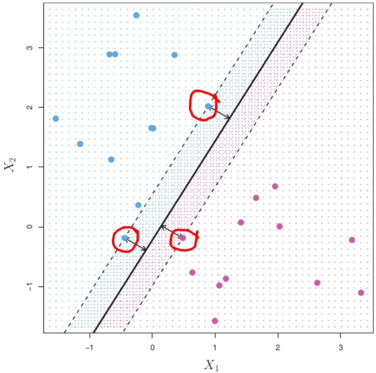


SVM: only uses inner products

Hard margin SVM

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

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



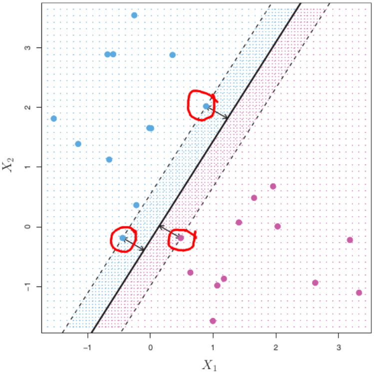
SVM: only uses inner products

Hard margin SVM

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

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

- The points that satisfy these constraints with equality are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.



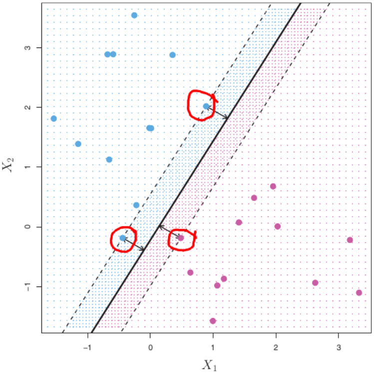
SVM: only uses inner products

Hard margin SVM

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

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

- The points that satisfy these constraints with equality are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)



SVM: only uses inner products

Hard margin SVM

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

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

- The points that satisfy these constraints with equality are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)
- So inner products are the only properties of the data that we use:

SVM: only uses inner products

Hard margin SVM

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

- The points that satisfy these constraints with equality are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

- So inner products are the only properties of the data that we use:

$$\theta \cdot x^i = \sum_j \alpha^j y^j \left(x^j \cdot x^i \right),$$

SVM: only uses inner products

Hard margin SVM

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

- The points that satisfy these constraints with equality are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

- So inner products are the only properties of the data that we use:

$$\theta \cdot x^i = \sum_j \alpha^j y^j \left(x^j \cdot x^i\right),$$
$$\|\theta\|^2 = \sum_{i,j} \alpha^i \alpha^j y^i y^j \left(x^i \cdot x^j\right).$$

SVM: only uses inner products

Soft margin SVM

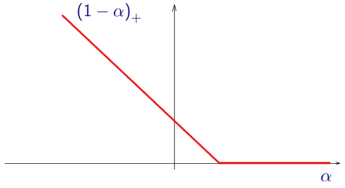
$$\min_{\theta} \qquad \|\theta\|^2 + C\sum_{i=1}^{n}\left(1 - y^i\theta\cdot x^i\right)_+ .$$

SVM: only uses inner products

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:



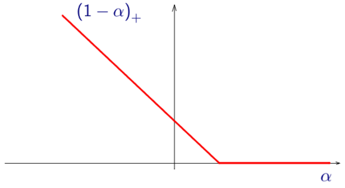
SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i\theta\cdot x^i \leq 1$ are called *support vectors*.

SVM margin cost function:



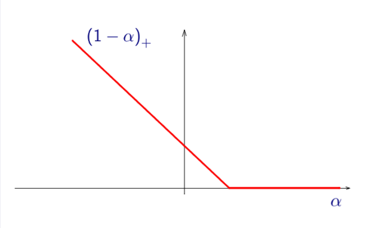
SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i \theta \cdot x^i \leq 1$ are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.

SVM margin cost function:



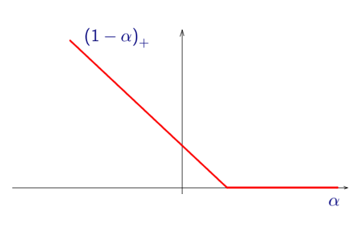
SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i \theta \cdot x^i \leq 1$ are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

SVM margin cost function:



SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i\theta \cdot x^i \leq 1$ are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

- Again, inner products are the only properties of the data that we use:

SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i \theta \cdot x^i \leq 1$ are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

- Again, inner products are the only properties of the data that we use:

$$\theta \cdot x^i = \sum_j \alpha^j y^j \left(x^j \cdot x^i\right),$$

SVM: only uses inner products

Soft margin SVM

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

- The points satisfying $y^i\theta\cdot x^i \le 1$ are called *support vectors*.
- It's clear that the solution θ only depends on the support vectors.
- It turns out that $\theta = \sum_j \alpha^j y^j x^j$. ($\alpha^j \neq 0$ only for support vectors.)

- Again, inner products are the only properties of the data that we use:

$$\theta\cdot x^i = \sum_j \alpha^j y^j \left(x^j\cdot x^i\right),$$

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

Kernels

Kernels

- The only property of the data we need:

$$K(x^i, x^j) = x^i \cdot x^j.$$

Kernels

- The only property of the data we need:

$$K(x^i, x^j) = x^i \cdot x^j.$$

- Or, if we use features $\phi(x)$, we need

$$K(x^i, x^j) = \phi\left(x^i\right) \cdot \phi\left(x^j\right).$$

Kernels

- The only property of the data we need:

$$K(x^i, x^j) = x^i \cdot x^j.$$

- Or, if we use features $\phi(x)$, we need

$$K(x^i, x^j) = \phi\left(x^i\right) \cdot \phi\left(x^j\right).$$

- We might not need to compute $\phi(x^i)$ (we don't even need to know what it is, we just need to know that it exists).

Kernels

- The only property of the data we need:

$$K(x^i, x^j) = x^i \cdot x^j.$$

- Or, if we use features $\phi(x)$, we need

$$K(x^i, x^j) = \phi\left(x^i\right) \cdot \phi\left(x^j\right).$$

- We might not need to compute $\phi(x^i)$ (we don't even need to know what it is, we just need to know that it exists).
- All we need is the $K(x^i, x^j)$.

Kernels

- The only property of the data we need:

$$K(x^i, x^j) = x^i \cdot x^j.$$

- Or, if we use features $\phi(x)$, we need

$$K(x^i, x^j) = \phi\left(x^i\right) \cdot \phi\left(x^j\right).$$

- We might not need to compute $\phi(x^i)$ (we don't even need to know what it is, we just need to know that it exists).
- All we need is the $K(x^i, x^j)$.
- K is called a *kernel*.

Kernels

- Given the $K(x^i, x^j)$, we can then solve the optimization problem to find the α^j. They determine the parameters:

$$\theta = \sum_j \alpha^j y^j \phi(x^j).$$

Kernels

- Given the $K(x^i, x^j)$, we can then solve the optimization problem to find the α^j. They determine the parameters:

$$\theta = \sum_j \alpha^j y^j \phi(x^j).$$

- We can compute the classifier for a test point x via

$$\theta \cdot \phi(x)$$

Kernels

- Given the $K(x^i, x^j)$, we can then solve the optimization problem to find the α^j. They determine the parameters:

$$\theta = \sum_j \alpha^j y^j \phi(x^j).$$

- We can compute the classifier for a test point x via

$$\theta \cdot \phi(x) = \sum_j \alpha^j y^j \left(\phi(x^j) \cdot \phi(x)\right)$$

Kernels

- Given the $K(x^i, x^j)$, we can then solve the optimization problem to find the α^j. They determine the parameters:

$$\theta = \sum_j \alpha^j y^j \phi(x^j).$$

- We can compute the classifier for a test point x via

$$\begin{aligned}\theta \cdot \phi(x) &= \sum_j \alpha^j y^j \left(\phi(x^j) \cdot \phi(x)\right) \\ &= \sum_j \alpha^j y^j K\left(x^j, x\right)\end{aligned}$$

Kernels

Examples of Kernels for $x, \tilde{x} \in \mathbb{R}^d$

$$K_m(x, \tilde{x}) = (1 + x \cdot \tilde{x})^m$$ degree-m polynomial kernel

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

$$K_2(x, \tilde{x}) = (1 + x \cdot \tilde{x})^2$$

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

$$\begin{aligned} K_2(x, \tilde{x}) &= (1 + x \cdot \tilde{x})^2 \\ &= (1 + x_1\tilde{x}_1 + x_2\tilde{x}_2)^2 \end{aligned}$$

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

$$\begin{aligned} K_2(x, \tilde{x}) &= (1 + x \cdot \tilde{x})^2 \\ &= (1 + x_1\tilde{x}_1 + x_2\tilde{x}_2)^2 \\ &= 1 + x_1^2\tilde{x}_1^2 + x_2^2\tilde{x}_2^2 + \end{aligned}$$

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

$$
\begin{aligned}
K_2(x, \tilde{x}) &= (1 + x \cdot \tilde{x})^2 \\
&= (1 + x_1\tilde{x}_1 + x_2\tilde{x}_2)^2 \\
&= 1 + x_1^2\tilde{x}_1^2 + x_2^2\tilde{x}_2^2 + 2x_1\tilde{x}_1 + 2x_2\tilde{x}_2 + 2x_1\tilde{x}_1x_2\tilde{x}_2
\end{aligned}
$$

Kernels

For example, for $x, \tilde{x} \in \mathbb{R}^2$, the degree $m = 2$ polynomial kernel is

$$
\begin{aligned}
K_2(x, \tilde{x}) &= (1 + x \cdot \tilde{x})^2 \\
&= (1 + x_1\tilde{x}_1 + x_2\tilde{x}_2)^2 \\
&= 1 + x_1^2\tilde{x}_1^2 + x_2^2\tilde{x}_2^2 + 2x_1\tilde{x}_1 + 2x_2\tilde{x}_2 + 2x_1\tilde{x}_1x_2\tilde{x}_2 \\
&= \underbrace{\begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \end{pmatrix}}_{\phi(x)} \cdot \underbrace{\begin{pmatrix} 1 \\ \tilde{x}_1^2 \\ \tilde{x}_2^2 \\ \sqrt{2}\tilde{x}_1 \\ \sqrt{2}\tilde{x}_2 \\ \sqrt{2}\tilde{x}_1\tilde{x}_2 \end{pmatrix}}_{\phi(\tilde{x})}
\end{aligned}
$$

Kernels

So the classifier for a test point x involves thresholding

$$\theta \cdot \phi(x)$$

Kernels

So the classifier for a test point x involves thresholding

$$\theta \cdot \phi(x) = \theta \cdot \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \end{pmatrix}.$$

Kernels

So the classifier for a test point x involves thresholding

$$\theta \cdot \phi(x) = \theta \cdot \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}.$$

This is equivalent to the polynomial features we considered earlier.

Kernels

So the classifier for a test point x involves thresholding

$$\theta \cdot \phi(x) = \theta \cdot \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \end{pmatrix}.$$

This is equivalent to the polynomial features we considered earlier. But we can compute it via

$$\theta \cdot \phi(x) = \sum_j \alpha^j y^j K_2(x^j, x).$$

Kernels

Examples of Kernels for $x, \tilde{x} \in \mathbb{R}^d$

$K_m(x, \tilde{x}) = (1 + x \cdot \tilde{x})^m$ degree-m polynomial kernel

Kernels

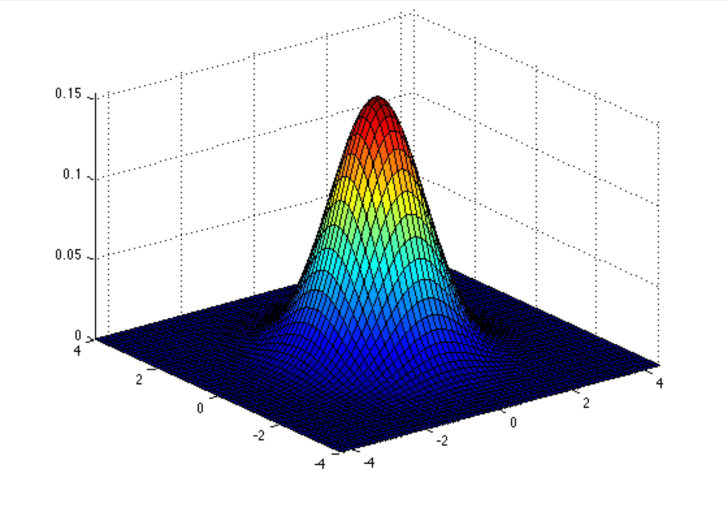
Examples of Kernels for $x, \tilde{x} \in \mathbb{R}^d$

$$K_m(x, \tilde{x}) = (1 + x \cdot \tilde{x})^m \qquad \text{degree-}m \text{ polynomial kernel}$$

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right) \qquad \text{radial basis function kernel}$$

Kernels

Gaussian radial basis function kernel (for fixed $\tilde{x} \in \mathbb{R}^2$)



Kernels

How can we write

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma\|x - \tilde{x}\|^2\right) = \phi(x) \cdot \phi(\tilde{x})?$$

Kernels

How can we write

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right) = \phi(x) \cdot \phi(\tilde{x})?$$

It turns out we can, but ϕ is infinite-dimensional.

Kernels

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma\|x - \tilde{x}\|^2\right).$$

Kernels

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right).$$

- The scaling parameter γ affects smoothness.

Kernels

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma\|x - \tilde{x}\|^2\right).$$

- The scaling parameter γ affects smoothness.
- Small values: smooth decision boundary (many points affect decision).

Kernels

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right).$$

- The scaling parameter γ affects smoothness.
- Small values: smooth decision boundary (many points affect decision).
- Large values: rough decision boundary (only nearby points).

Kernels

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right).$$

- The scaling parameter γ affects smoothness.
- Small values: smooth decision boundary (many points affect decision).
- Large values: rough decision boundary (only nearby points).
- This affects how much regularization is needed.

$$K_{rbf}(x, \tilde{x}) = \exp\left(-\gamma \|x - \tilde{x}\|^2\right).$$

- The scaling parameter γ affects smoothness.
- Small values: smooth decision boundary (many points affect decision).
- Large values: rough decision boundary (only nearby points).
- This affects how much regularization is needed.

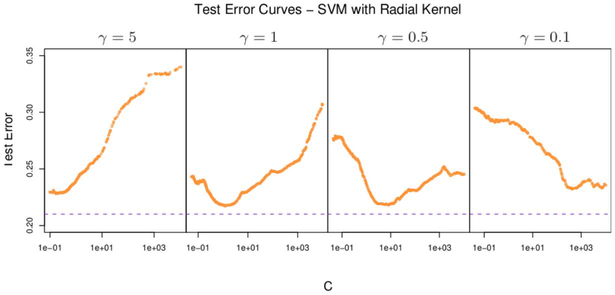


FIGURE 12.6. *Test-error curves as a function of the cost parameter C for the radial-kernel SVM classifier on the mixture data. At the top of each plot is the scale parameter γ for the radial kernel: $K_\gamma(x, y) = \exp -\gamma||x - y||^2$. The optimal value for C depends quite strongly on the scale of the kernel. The Bayes error rate is indicated by the broken horizontal lines.*

Kernels

Why use kernels?

Kernels

Why use kernels?

- They provide a modular approach to training a classifier: the same optimization procedure can be used with many different features. All that is required is the kernel matrix, with entries

$$K_{ij} = K(x_i, x_j).$$

Kernels

Why use kernels?

- They provide a modular approach to training a classifier: the same optimization procedure can be used with many different features. All that is required is the kernel matrix, with entries

$$K_{ij} = K(x_i, x_j).$$

- Often this representation is considerably easier to compute. For example, the degree m polynomial kernel on $\mathbb{R}^d$ is an inner product involving $\binom{m+d}{m}$ features.

Outline

- Recall: linear classifiers, perceptron algorithm
- Support vector machines
- Features
- Features and overfitting
- Role of the regularization parameter C
- Regularization and overfitting
- Kernels