Introduction to Machine Learning
Introduction to Support Vector Machines

Andres Mendez-Vazquez

June 13, 2018

Outline

Cinvestav

Outline

Cinvestav

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

- She is currently the Head of Google Research, New York
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

BTW Corinna Cortes

- Danish computer scientist who is known for her contributions to the field of machine learning.
- She is currently the Head of **Google Research**, New York.
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

BTW Corinna Cortes

- Danish computer scientist who is known for her contributions to the field of machine learning.
- She is currently the Head of **Google Research**, New York.
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

BTW Corinna Cortes

- Danish computer scientist who is known for her contributions to the field of machine learning.
- She is currently the Head of **Google Research**, New York.
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

BTW Corinna Cortes

- Danish computer scientist who is known for her contributions to the field of machine learning.
- She is currently the Head of **Google Research**, New York.
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

History

Invented by Vladimir Vapnik and Alexey Ya. Chervonenkis in 1963

- At the Institute of Control Sciences, Moscow
- On the paper "Estimation of dependencies based on empirical data"

Corinna Cortes and Vladimir Vapnik in 1995

- They Invented their Current Incarnation - Soft Margins
- At the AT&T Labs

BTW Corinna Cortes

- Danish computer scientist who is known for her contributions to the field of machine learning.
- She is currently the Head of **Google Research**, New York.
- Cortes is a recipient of the Paris Kanellakis Theory and Practice Award (ACM) for her work on theoretical foundations of support vector machines.

In addition

Alexey Yakovlevich Chervonenkis

He was a Soviet and Russian mathematician, and, with Vladimir Vapnik, was one of the main developers of the Vapnik–Chervonenkis theory, also known as the "**fundamental theory of learning**" an important part of computational learning theory.

He died in September 22nd, 2014

At Losiny Ostrov National Park on 22 September 2014.

Cinvestav

In addition

Alexey Yakovlevich Chervonenkis

He was a Soviet and Russian mathematician, and, with Vladimir Vapnik, was one of the main developers of the Vapnik–Chervonenkis theory, also known as the "**fundamental theory of learning**" an important part of computational learning theory.

He died in September 22nd, 2014

At Losiny Ostrov National Park on 22 September 2014.

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Applications

Partial List

1. Predictive Control
   - Control of chaotic systems.
2. Inverse Geosounding Problem
   - It is used to understand the internal structure of our planet.
3. Environmental Sciences
   - Spatio-temporal environmental data analysis and modeling.
4. Protein Fold and Remote Homology Detection
   - In the recognition if two different species contain similar genes.
5. Facial expression classification
6. Texture Classification
7. E-Learning
8. Handwritten Recognition
9. AND counting....

Outline

Cinvestav

Separable Classes

Given

$$\boldsymbol{x}_i,\ i = 1, \cdots, N$$

A set of samples belonging to two classes ω_1, ω_2.

Cinvestav

Separable Classes

Given

$$\boldsymbol{x}_i,\ i = 1, \cdots, N$$

A set of samples belonging to two classes ω_1, ω_2.

Objective

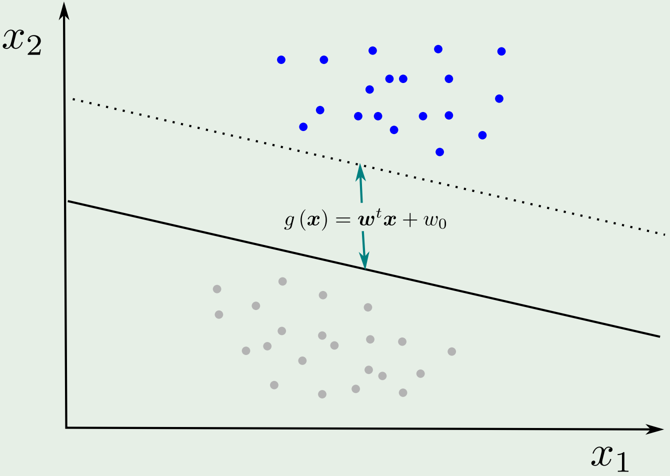
We want to obtain a decision function as simple as

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

Cinvestav

Such that we can do the following

A linear separation function $g(\boldsymbol{x}) = \boldsymbol{w}^t \boldsymbol{x} + w_0$



Outline

Cinvestav

In other words ...

We have the following samples

- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m \in C_1$
- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in C_2$

In other words ...

We have the following samples

- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m \in C_1$
- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in C_2$

We want the following decision surfaces

- $\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \geq 0$ for $d_i = +1$ if $\boldsymbol{x}_i \in C_1$
- $\boldsymbol{w}^T \boldsymbol{x}_j + w_0 \leq 0$ for $d_j = -1$ if $\boldsymbol{x}_j \in C_2$

In other words ...

We have the following samples

- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m \in C_1$
- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in C_2$

We want the following decision surfaces

- $\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \geq 0$ for $d_i = +1$ if $\boldsymbol{x}_i \in C_1$
- $\boldsymbol{w}^T \boldsymbol{x}_j + w_0 \leq 0$ for $d_j = -1$ if $\boldsymbol{x}_j \in C_2$

In other words ...

We have the following samples

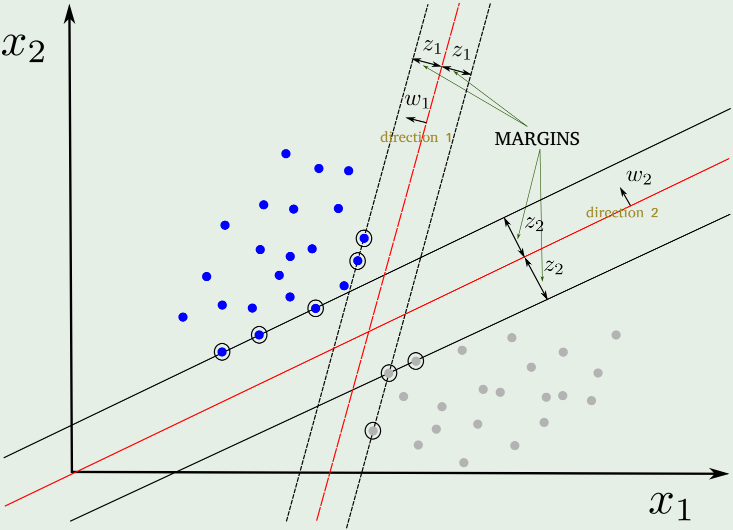
- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m \in C_1$
- For $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in C_2$

We want the following decision surfaces

- $\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \geq 0$ for $d_i = +1$ if $\boldsymbol{x}_i \in C_1$
- $\boldsymbol{w}^T \boldsymbol{x}_j + w_0 \leq 0$ for $d_j = -1$ if $\boldsymbol{x}_j \in C_2$

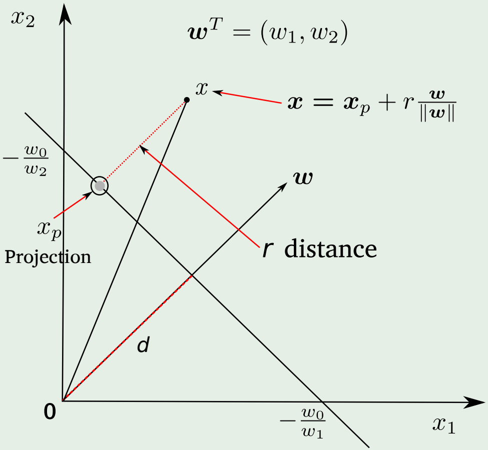
What do we want?

Our goal is to search for a direction w that gives the maximum possible margin



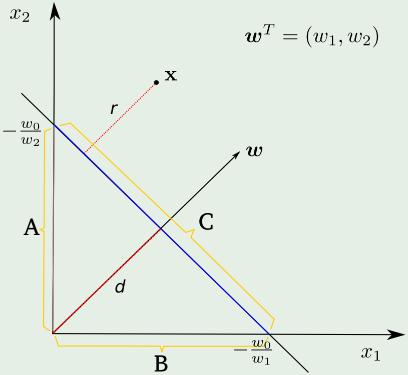
Remember

We have the following



A Little of Geometry

Thus

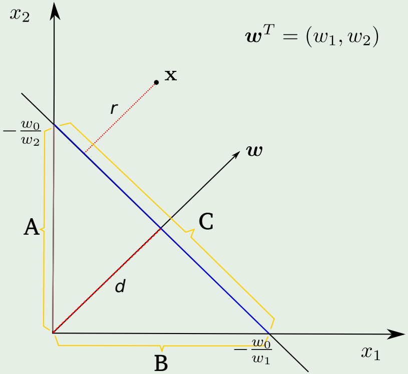


Then

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}, \quad r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}} \tag{1}$$

A Little of Geometry

Thus



Then

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}, \; r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}} \tag{1}$$

First $d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$

We can use the following rule in a triangle with a 90^o angle

$$Area = \frac{1}{2}Cd \quad (2)$$

In addition, the area can be calculated also as

$$Area = \frac{1}{2}AB \quad (3)$$

Thus

$$d = \frac{AB}{C}$$

Remark: Can you get the rest of values?

First $d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$

We can use the following rule in a triangle with a 90^o angle

$$Area = \frac{1}{2}Cd \quad (2)$$

In addition, the area can be calculated also as

$$Area = \frac{1}{2}AB \quad (3)$$

Thus

$$d = \frac{AB}{C}$$

Remark: Can you get the rest of values?

First $d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}$

We can use the following rule in a triangle with a 90^o angle

$$Area = \frac{1}{2} C d \tag{2}$$

In addition, the area can be calculated also as

$$Area = \frac{1}{2} A B \tag{3}$$

Thus

$$d = \frac{AB}{C}$$

Remark: Can you get the rest of values?

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4}$$

Thus, we have

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4}$$

Thus, we have

$$\begin{aligned}
g\left(\boldsymbol{x}\right) =& \boldsymbol{w}^T\left[\boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right] + w_0 \\
=& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\
=& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} \\
=& g\left(\boldsymbol{x}_p\right) + r\|\boldsymbol{w}\|
\end{aligned}$$

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \quad (4)$$

Thus, we have

$$\begin{aligned} g\left(\boldsymbol{x}\right) &= \boldsymbol{w}^T\left[\boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right] + w_0 \\ &= \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\ &= \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} \\ &= g\left(\boldsymbol{x}_p\right) + r\|\boldsymbol{w}\| \end{aligned}$$

Then

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2+w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4}$$

Thus, we have

$$\begin{aligned}
g\left(\boldsymbol{x}\right) =& \boldsymbol{w}^T\left[\boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right] + w_0 \\
=& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\
=& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} \\
=& g\left(\boldsymbol{x}_p\right) + r\|\boldsymbol{w}\|
\end{aligned}$$

Then

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2+w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4}$$

Thus, we have

$$\begin{aligned} g\left(\boldsymbol{x}\right) =& \boldsymbol{w}^T\left[\boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right] + w_0 \\ =& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\ =& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} \\ =& g\left(\boldsymbol{x}_p\right) + r\|\boldsymbol{w}\| \end{aligned}$$

Then

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

What about $r = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$?

First, remember

$$g\left(\boldsymbol{x}_p\right) = 0 \text{ and } \boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \tag{4}$$

Thus, we have

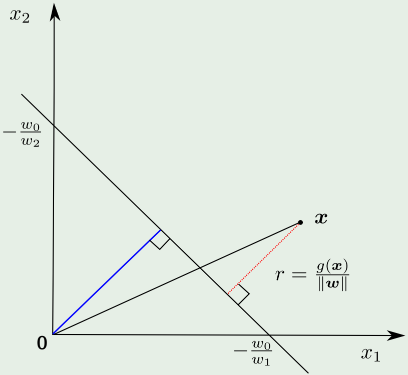
$$\begin{aligned} g\left(\boldsymbol{x}\right) =& \boldsymbol{w}^T\left[\boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right] + w_0 \\ =& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\ =& \boldsymbol{w}^T\boldsymbol{x}_p + w_0 + r\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} \\ =& g\left(\boldsymbol{x}_p\right) + r\left\|\boldsymbol{w}\right\| \end{aligned}$$

Then

$$r = \frac{g(\mathbf{x})}{||\mathbf{w}||}$$

This has the following interpretation

The distance from the projection



Now

We know that the straight line that we are looking for looks like

$$\boldsymbol{w}^T x + w_0 = 0 \tag{5}$$

What about something like this

$$\boldsymbol{w}^T x + w_0 = \delta \tag{6}$$

Clearly

This will be above or below the initial line $\boldsymbol{w}^T x + w_0 = 0$.

Now

We know that the straight line that we are looking for looks like

$$\boldsymbol{w}^T x + w_0 = 0 \tag{5}$$

What about something like this

$$\boldsymbol{w}^T x + w_0 = \delta \tag{6}$$

Clearly

This will be above or below the initial line $w^T x + w_0 = 0$.

Now

We know that the straight line that we are looking for looks like

$$\boldsymbol{w}^T x + w_0 = 0 \tag{5}$$

What about something like this

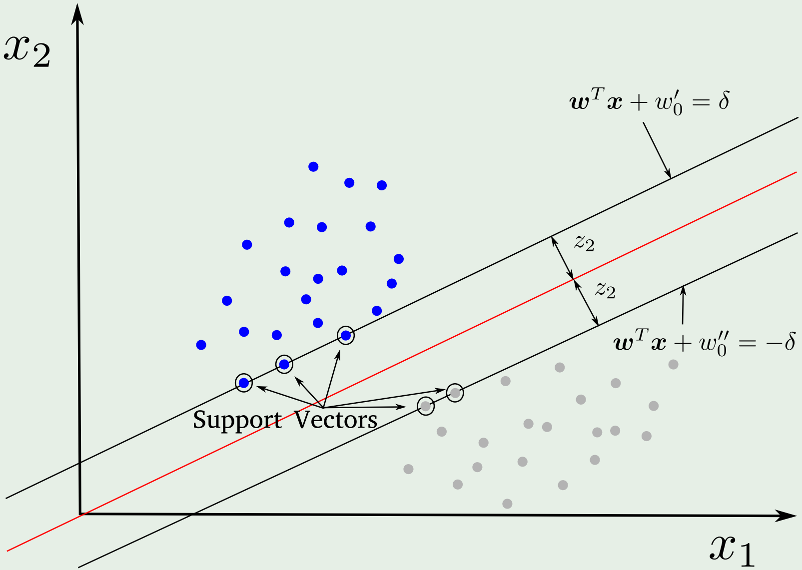
$$\boldsymbol{w}^T x + w_0 = \delta \tag{6}$$

Clearly

This will be above or below the initial line $\boldsymbol{w}^T x + w_0 = 0$.

Come back to the hyperplanes

We have then for each border support line an specific bias!!!



Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^{T}\mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^{T}\mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\delta}$, $w_{10} = \frac{w_0'}{\delta}$, and $w_{01} = \frac{w_0''}{\delta}$

Cinvestav

Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^T\mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^T\mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\delta}$, $w_{10} = \frac{w_0'}{\delta}$,and $w_{01} = \frac{w_0''}{\delta}$

Now, we come back to the middle separator hyperplane, but with the normalized term

- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \geq \boldsymbol{w}'^T\mathbf{x} + w_{10}$ for $d_i = +1$
- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \leq \boldsymbol{w}'^T\mathbf{x} + w_{01}$ for $d_i = -1$
  - Where w_0 is the bias of that central hyperplane!! And the $\boldsymbol{w}$ is the normalized direction of $\boldsymbol{w}'$

Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^T\mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^T\mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\boldsymbol{\delta}}$, $w_{10} = \frac{w_0'}{\delta}$,and $w_{01} = \frac{w_0''}{\delta}$

Now, we come back to the middle separator hyperplane, but with the normalized term

- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \geq \boldsymbol{w}'^T\mathbf{x} + w_{10}$ for $d_i = +1$
- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \leq \boldsymbol{w}'^T\mathbf{x} + w_{01}$ for $d_i = -1$
  - Where w_0 is the bias of that central hyperplane!! And the $\boldsymbol{w}$ is the normalized direction of $\boldsymbol{w}'$

Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^{T}\mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^{T}\mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\boldsymbol{\delta}}$, $w_{10} = \frac{w_0'}{\delta}$,and $w_{01} = \frac{w_0''}{\delta}$

Now, we come back to the middle separator hyperplane, but with the normalized term

- $\boldsymbol{w}^{T}\mathbf{x}_i + w_0 \geq \boldsymbol{w}'^{T}\mathbf{x} + w_{10}$ for $d_i = +1$
- $\boldsymbol{w}^{T}\mathbf{x}_i + w_0 \leq \boldsymbol{w}'^{T}\mathbf{x} + w_{01}$ for $d_i = -1$
  - Where w_0 is the bias of that central hyperplane!! And the w is the normalized direction of w'

Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^T \mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^T \mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\boldsymbol{\delta}}$, $w_{10} = \frac{w_0'}{\delta}$,and $w_{01} = \frac{w_0''}{\delta}$

Now, we come back to the middle separator hyperplane, but with the normalized term

- $\boldsymbol{w}^T \mathbf{x}_i + w_0 \geq \boldsymbol{w}'^T \mathbf{x} + w_{10}$ for $d_i = +1$
- $\boldsymbol{w}^T \mathbf{x}_i + w_0 \leq \boldsymbol{w}'^T \mathbf{x} + w_{01}$ for $d_i = -1$
  - Where w_0 is the bias of that central hyperplane!! And the w is the normalized direction of w'

Cinvestav

Then, normalize by δ

The new margin functions

- $\boldsymbol{w}'^T\mathbf{x} + w_{10} = 1$
- $\boldsymbol{w}'^T\mathbf{x} + w_{01} = -1$

where $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\boldsymbol{\delta}}$, $w_{10} = \frac{w_0'}{\delta}$,and $w_{01} = \frac{w_0''}{\delta}$

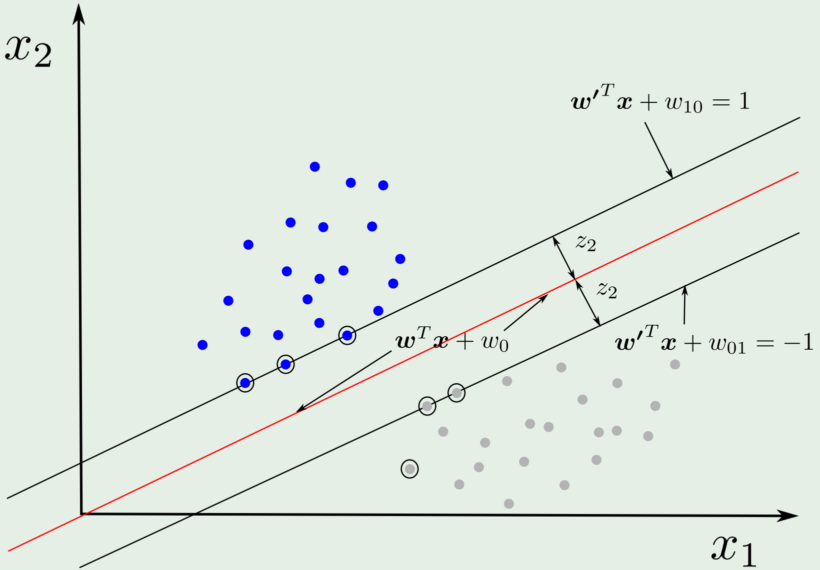
Now, we come back to the middle separator hyperplane, but with the normalized term

- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \geq \boldsymbol{w}'^T\mathbf{x} + w_{10}$ for $d_i = +1$
- $\boldsymbol{w}^T\mathbf{x}_i + w_0 \leq \boldsymbol{w}'^T\mathbf{x} + w_{01}$ for $d_i = -1$
  - Where w_0 is the bias of that central hyperplane!! And the $\boldsymbol{w}$ is the normalized direction of $\boldsymbol{w}'$

Cinvestav

Come back to the hyperplanes

The meaning of what I am saying!!!



Outline

Cinvestav

A little about Support Vectors

They are the vectors (Here, we assume that w)

$\boldsymbol{x}_i$ such that $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = 1$ or $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = -1$

Cinvestav

A little about Support Vectors

They are the vectors (Here, we assume that w)

$\boldsymbol{x}_i$ such that $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = 1$ or $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = -1$

Properties

- The vectors nearest to the decision surface and the most difficult to classify.
- Because of that, we have the name "Support Vector Machines".

A little about Support Vectors

They are the vectors (Here, we assume that w)

$\boldsymbol{x}_i$ such that $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = 1$ or $\boldsymbol{w}^T\boldsymbol{x}_i + w_0 = -1$

Properties

- The vectors nearest to the decision surface and the most difficult to classify.
- Because of that, we have the name "Support Vector Machines".

Now, we can resume the decision rule for the hyperplane

For the support vectors

$$g\left(\boldsymbol{x}_i\right) = \boldsymbol{w}^T \boldsymbol{x}_i + w_0 = -(+)1 \text{ for } d_i = -(+)1 \quad (7)$$

Implies

The distance to the support vectors is:

$$r = \frac{g\left(\boldsymbol{x}_i\right)}{\|\boldsymbol{w}\|} = \begin{cases} \frac{1}{\|\boldsymbol{w}\|} & \text{if } d_i = +1 \\ -\frac{1}{\|\boldsymbol{w}\|} & \text{if } d_i = -1 \end{cases}$$

Now, we can resume the decision rule for the hyperplane

For the support vectors

$$g(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i + w_0 = -(+)1 \text{ for } d_i = -(+)1 \tag{7}$$

Implies

The distance to the support vectors is:

$$r = \frac{g(\boldsymbol{x}_i)}{||\boldsymbol{w}||} = \begin{cases} \frac{1}{||\boldsymbol{w}||} & \text{if } d_i = +1 \\ -\frac{1}{||\boldsymbol{w}||} & \text{if } d_i = -1 \end{cases}$$

Cinvestav

Therefore ...

We want the optimum value of the margin of separation as

$$\rho = \frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||} \tag{8}$$

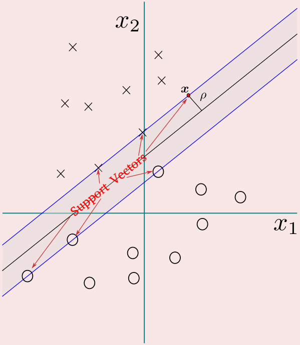
And the support vectors define the value of ρ

Therefore ...

We want the optimum value of the margin of separation as

$$\rho = \frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||} \tag{8}$$

And the support vectors define the value of ρ



Outline

Cinvestav

Thus

If we want to maximize

$$\rho = \frac{2}{||\boldsymbol{w}||}$$

We instead to minimize

$$||w|| = \sqrt{w^T w}$$

Or to minimize, after all we only need the direction of the vector w

$$\frac{1}{2} w^T w$$

Cinvestav

Thus

If we want to maximize

$$\rho = \frac{2}{||\boldsymbol{w}||}$$

We instead to minimize

$$||\boldsymbol{w}|| = \sqrt{\boldsymbol{w}^T \boldsymbol{w}}$$

Or to minimize, after all we only need the direction of the vector w

$$\frac{1}{2} w^T w$$

Thus

If we want to maximize

$$\rho = \frac{2}{||\boldsymbol{w}||}$$

We instead to minimize

$$||\boldsymbol{w}|| = \sqrt{\boldsymbol{w}^T \boldsymbol{w}}$$

Or to minimize, after all we only need the direction of the vector $\boldsymbol{w}$

$$\frac{1}{2}\boldsymbol{w}^T \boldsymbol{w}$$

Under the restrictions

Then, we have the samples with labels

$$T = \{(\boldsymbol{x}_i, d_i)\}_{i=1}^{N}$$

Then we can put the decision rule as

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0\right) \geq 1 \; i = 1, \cdots, N$$

Under the restrictions

Then, we have the samples with labels

$$T = \{(\boldsymbol{x}_i, d_i)\}_{i=1}^{N}$$

Then we can put the decision rule as

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0\right) \geq 1 \; i = 1, \cdots, N$$

Cinvestav

Then, we have the optimization problem

The optimization problem

$$min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

s.t. $d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 \; i = 1, \cdots, N$

Then, we have the optimization problem

The optimization problem

$$min_{\boldsymbol{w}} \Phi\left(\boldsymbol{w}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\text{s.t. } d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 \; i = 1, \cdots, N$$

Observations

- The cost functions $\Phi\left(\boldsymbol{w}\right)$ is convex.
- The constrains are linear with respect to w.

Then, we have the optimization problem

The optimization problem

$$min_{\boldsymbol{w}} \Phi\left(\boldsymbol{w}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\text{s.t. } d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 \;\; i = 1, \cdots, N$$

Observations

- The cost functions $\Phi\left(\boldsymbol{w}\right)$ is convex.
- The constrains are linear with respect to $\boldsymbol{w}$.

Outline

Cinvestav

Then, Rewriting The Optimization Problem

The optimization with equality constraints

$$min_{\boldsymbol{w}} \Phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\text{s.t. } d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 \; i = 1, \cdots, N$$

Cinvestav

Then, for our problem

Using the Lagrange Multipliers (We will call them α_i)

We obtain the following cost function that we want to minimize

$$J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i[d_i(\boldsymbol{w}^T\mathbf{x}_i + w_0) - 1]$$

Cinvestav

Then, for our problem

Using the Lagrange Multipliers (We will call them α_i)

We obtain the following cost function that we want to minimize

$$J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i[d_i(\boldsymbol{w}^T\mathbf{x}_i + w_0) - 1]$$

Observation

- Minimize with respect to $\mathbf{w}$ and w_0.
- Maximize with respect to α because it dominates

$$-\sum_{i=1}^{N} \alpha_i[d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]. \quad (9)$$

Then, for our problem

Using the Lagrange Multipliers (We will call them α_i)

We obtain the following cost function that we want to minimize

$$J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T \mathbf{x}_i + w_0) - 1]$$

Observation

- Minimize with respect to $\mathbf{w}$ and w_0.
- Maximize with respect to α because it dominates

$$-\sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) - 1]. \quad (9)$$

Then, for our problem

Using the Lagrange Multipliers (We will call them α_i)

We obtain the following cost function that we want to minimize

$$J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T\mathbf{x}_i + w_0) - 1]$$

Observation

- Minimize with respect to $\mathbf{w}$ and w_0.
- Maximize with respect to α because it dominates

$$-\sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]. \quad (9)$$

Outline

Cinvestav

Karush-Kuhn-Tucker Conditions

First An Inequality Constrained Problem P

$$
\begin{aligned}
\min \quad & f(\boldsymbol{x}) & \\
s.t \quad & g_1(\boldsymbol{x}) & = 0 \\
& \vdots & \\
& g_N(\boldsymbol{x}) & = 0
\end{aligned}
$$

A really minimal version!!! Hey, it is a patch work!!!

A point x is a local minimum of an equality constrained problem P only if a set of non-negative α_j's may be found such that:

$$\nabla L(x, \alpha) = \nabla f(x) - \sum_{i=1}^{N} \alpha_i \nabla g_i(x) = 0$$

Karush-Kuhn-Tucker Conditions

First An Inequality Constrained Problem P

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) & \\ s.t \quad & g_1(\boldsymbol{x}) & = 0 \\ & \vdots & \\ & g_N(\boldsymbol{x}) & = 0 \end{aligned}$$

A really minimal version!!! Hey, it is a patch work!!!

A point $\boldsymbol{x}$ is a local minimum of an equality constrained problem P only if a set of non-negative α_j's may be found such that:

$$\nabla L(\boldsymbol{x}, \boldsymbol{\alpha}) = \nabla f(\boldsymbol{x}) - \sum_{i=1}^{N} \alpha_i \nabla g_i(\boldsymbol{x}) = 0$$

Karush-Kuhn-Tucker Conditions

Important

Think about this each constraint correspond to a sample in both classes, thus

- The corresponding α_i's are going to be zero after optimization, if a constraint is not active i.e. $d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 \neq 0$ (Remember Maximization).

Again the Support Vectors

This actually defines the idea of support vectors!!!

Thus

Only the α_i's with active constraints (Support Vectors) will be different from zero when $d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 = 0$.

Karush-Kuhn-Tucker Conditions

Important

Think about this each constraint correspond to a sample in both classes, thus

- The corresponding α_i's are going to be zero after optimization, if a constraint is not active i.e. $d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) - 1 \neq 0$ (Remember Maximization).

Again the Support Vectors

This actually defines the idea of support vectors!!!

Thus

Only the α_i's with active constraints (Support Vectors) will be different from zero when $d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) - 1 = 0$.

Karush-Kuhn-Tucker Conditions

Important

Think about this each constraint correspond to a sample in both classes, thus

- The corresponding α_i's are going to be zero after optimization, if a constraint is not active i.e. $d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) - 1 \neq 0$ (Remember Maximization).

Again the Support Vectors

This actually defines the idea of support vectors!!!

Thus

Only the α_i's with active constraints (Support Vectors) will be different from zero when $d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_0\right) - 1 = 0$.

The necessary conditions for optimality

Condition 1

$$\frac{\partial J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right)}{\partial \boldsymbol{w}} = 0$$

Condition 2

$$\frac{\partial J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right)}{\partial w_0} = 0$$

The necessary conditions for optimality

Condition 1

$$\frac{\partial J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right)}{\partial \boldsymbol{w}} = 0$$

Condition 2

$$\frac{\partial J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right)}{\partial w_0} = 0$$

Using the conditions

We have the first condition

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{\partial \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}{\partial \boldsymbol{w}} - \frac{\partial \sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]}{\partial \boldsymbol{w}} = 0$$

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w} + \boldsymbol{w}) - \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{x}_i$$

Using the conditions

We have the first condition

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{\partial \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}{\partial \boldsymbol{w}} - \frac{\partial \sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]}{\partial \boldsymbol{w}} = 0$$

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w} + \boldsymbol{w}) - \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{x}_i$$

Thus

$$w = \sum_{i=1}^{N} \alpha_i d_i x_i \tag{10}$$

Using the conditions

We have the first condition

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{\partial \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}{\partial \boldsymbol{w}} - \frac{\partial \sum_{i=1}^{N} \alpha_i [d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) - 1]}{\partial \boldsymbol{w}} = 0$$

$$\frac{\partial J(\boldsymbol{w}, w_0, \alpha)}{\partial \boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w} + \boldsymbol{w}) - \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{x}_i$$

Thus

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i \quad (10)$$

Cinvestav

In a similar way ...

We have by the second optimality condition

$$\sum_{i=1}^{N} \alpha_i d_i = 0$$

Note

$$\alpha_i \left[d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 \right] = 0$$

Because the constraint vanishes in the optimal solution i.e. $\alpha_i = 0$ or $d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 = 0$.

In a similar way ...

We have by the second optimality condition

$$\sum_{i=1}^{N} \alpha_i d_i = 0$$

Note

$$\alpha_i \left[d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 \right] = 0$$

Because the constraint vanishes in the optimal solution i.e. $\alpha_i = 0$ or $d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_0 \right) - 1 = 0$.

Thus

We need something extra

Our classic trick of transforming a problem into another problem

In this case

We use the Primal-Dual Problem for Lagrangian

Where

We move from a minimization to a maximization!!!

Thus

We need something extra

Our classic trick of transforming a problem into another problem

In this case

We use the Primal-Dual Problem for Lagrangian

Where

We move from a minimization to a maximization!!!

Thus

We need something extra

Our classic trick of transforming a problem into another problem

In this case

We use the Primal-Dual Problem for Lagrangian

Where

We move from a minimization to a maximization!!!

Outline

Cinvestav

Duality Theorem

First Property

If the Primal has an optimal solution ($\boldsymbol{w}*$ and $\boldsymbol{\alpha}*$), the dual too.

Thus

In order to $\boldsymbol{w}*$ and $\boldsymbol{\alpha}*$ to be optimal solutions for the primal and dual problem respectively, It is necessary and sufficient that $\boldsymbol{w}*$:

- It is a feasible solution for the primal problem and

$$\begin{aligned}\Phi(\boldsymbol{w}*) &= J(\boldsymbol{w}*, w_0*, \boldsymbol{\alpha}*) \\ &= \min_{\boldsymbol{w}} J(\boldsymbol{w}*, w_0*, \boldsymbol{\alpha}*)\end{aligned}$$

Cinvestav

Duality Theorem

First Property

If the Primal has an optimal solution ($\boldsymbol{w}*$ and $\boldsymbol{\alpha}*$), the dual too.

Thus

In order to $\boldsymbol{w}*$ and $\boldsymbol{\alpha}*$ to be optimal solutions for the primal and dual problem respectively, It is necessary and sufficient that $\boldsymbol{w}*$:

- It is a feasible solution for the primal problem and

$$\begin{aligned}\Phi(\boldsymbol{w}*) &= J\left(\boldsymbol{w}*, w_0*, \boldsymbol{\alpha}*\right)\\ &= \min_{\boldsymbol{w}} J\left(\boldsymbol{w}*, w_0*, \boldsymbol{\alpha}*\right)\end{aligned}$$

Cinvestav

Reformulate our Equations

We have then

$$J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i d_i \boldsymbol{w}^T\mathbf{x}_i - w_0\sum_{i=1}^{N}\alpha_i d_i + \sum_{i=1}^{N}\alpha_i$$

Now for our 2nd optimality condition

$$J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i d_i \boldsymbol{w}^T\boldsymbol{x}_i + \sum_{i=1}^{N}\alpha_i$$

Reformulate our Equations

We have then

$$J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{w}^T \mathbf{x}_i - w_0 \sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i$$

Now for our 2nd optimality condition

$$J\left(\boldsymbol{w}, w_0, \boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{w}^T \boldsymbol{x}_i + \sum_{i=1}^{N} \alpha_i$$

We have finally for the 1st Optimality Condition:

First

$$\boldsymbol{w}^T\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{w}^T \boldsymbol{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

Second, setting $J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = Q(\boldsymbol{\alpha})$

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

We have finally for the 1st Optimality Condition:

First

$$\boldsymbol{w}^T\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{w}^T \boldsymbol{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

Second, setting $J(\boldsymbol{w}, w_0, \boldsymbol{\alpha}) = Q(\boldsymbol{\alpha})$

$$Q(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

From here, we have the problem

This is the problem that we really solve

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i d_i = 0 \tag{11}$$

$$\alpha_i \geq 0 \text{ for } i = 1, \cdots, N \tag{12}$$

Note

In the Primal, we were trying to minimize the cost function, for this it is necessary to maximize α. That is the reason why we are maximizing $Q(\alpha)$.

From here, we have the problem

This is the problem that we really solve

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i d_i = 0 \quad (11)$$

$$\alpha_i \geq 0 \text{ for } i = 1, \cdots, N \quad (12)$$

Note

In the Primal, we were trying to minimize the cost function, for this it is necessary to maximize $\boldsymbol{\alpha}$. That is the reason why we are maximizing $Q(\boldsymbol{\alpha})$.

Solving for $\boldsymbol{\alpha}$

We can compute $\boldsymbol{w}^*$ once we get the optimal α_i^* by using (Eq. 10)

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* d_i \boldsymbol{x}_i$$

In addition, we can compute the optimal bias w_0^* using the optimal weight, $\boldsymbol{w}^*$

For this, we use the positive margin equation:

$$g\left(\boldsymbol{x}^{(s)}\right) = \boldsymbol{w}^T \boldsymbol{x}^{(s)} + w_0 = 1$$

corresponding to a positive support vector.

Then

$$w_0 = 1 - \left(\boldsymbol{w}^*\right)^T \boldsymbol{x}^{(s)} \text{ for } d^{(s)} = 1 \quad (13)$$

Solving for $\boldsymbol{\alpha}$

We can compute $\boldsymbol{w}^*$ once we get the optimal α_i^* by using (Eq. 10)

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* d_i \boldsymbol{x}_i$$

In addition, we can compute the optimal bias w_0^* using the optimal weight, $\boldsymbol{w}^*$

For this, we use the positive margin equation:

$$g\left(\boldsymbol{x}^{(s)}\right) = \boldsymbol{w}^T \boldsymbol{x}^{(s)} + w_0 = 1$$

corresponding to a positive support vector.

Then

$$w_0 = 1 - \left(\boldsymbol{w}^*\right)^T \boldsymbol{x}^{(s)} \text{ for } d^{(s)} = 1 \tag{13}$$

Solving for $\boldsymbol{\alpha}$

We can compute $\boldsymbol{w}^*$ once we get the optimal α_i^* by using (Eq. 10)

$$\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* d_i \boldsymbol{x}_i$$

In addition, we can compute the optimal bias w_0^* using the optimal weight, $\boldsymbol{w}^*$

For this, we use the positive margin equation:

$$g\left(\boldsymbol{x}^{(s)}\right) = \boldsymbol{w}^T \boldsymbol{x}^{(s)} + w_0 = 1$$

corresponding to a positive support vector.

Then

$$w_0 = 1 - \left(\boldsymbol{w}^*\right)^T \boldsymbol{x}^{(s)} \text{ for } d^{(s)} = 1 \tag{13}$$

Outline

Cinvestav

What do we need?

Until now, we have only a maximal margin algorithm

- All this work fine when the classes are separable
- Problem, What when they are not separable?
- What we can do?

What do we need?

Until now, we have only a maximal margin algorithm

- All this work fine when the classes are separable
- Problem, What when they are not separable?
- What we can do?

What do we need?

Until now, we have only a maximal margin algorithm

- All this work fine when the classes are separable
- Problem, What when they are not separable?
- What we can do?

Outline

Cinvestav

Map to a higher Dimensional Space

Assume that exist a mapping

$$\boldsymbol{x} \in \mathbb{R}^l \to \boldsymbol{y} \in \mathbb{R}^k$$

Then, it is possible to define the following mapping

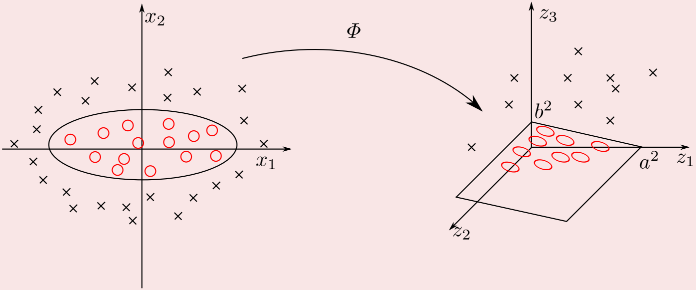
Cinvestav

Map to a higher Dimensional Space

Assume that exist a mapping

$$\boldsymbol{x} \in \mathbb{R}^{l} \rightarrow \boldsymbol{y} \in \mathbb{R}^{k}$$

Then, it is possible to define the following mapping



$$\Phi : (x_1, x_2) \rightarrow \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \rightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Define a map to a higher Dimension

Nonlinear transformations

Given a series of nonlinear transformations

$$\{\phi_i(\boldsymbol{x})\}_{i=1}^{m}$$

from input space to the feature space.

We can define the decision surface as

$$\sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) + w_0 = 0$$

Define a map to a higher Dimension

Nonlinear transformations

Given a series of nonlinear transformations

$$\{\phi_i(\boldsymbol{x})\}_{i=1}^{m}$$

from input space to the feature space.

We can define the decision surface as

$$\sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) + w_0 = 0$$

This allows us to define

The following vector

$$\phi(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \cdots, \phi_m(\boldsymbol{x}))^T$$

that represents the mapping.

From this mapping

We can define the following kernel function

$$K : \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

This allows us to define

The following vector

$$\phi(\boldsymbol{x}) = \left(\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \cdots, \phi_m(\boldsymbol{x})\right)^T$$

that represents the mapping.

From this mapping

We can define the following kernel function

$$K : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Outline

Cinvestav

Outline

Cinvestav

Basic Idea

Something Notable

- The SVM uses the scalar product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ as a measure of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and of distance to the hyperplane.
- Since the scalar product is linear, the SVM is a linear method.

Cinvestav

Basic Idea

Something Notable

- The SVM uses the scalar product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ as a measure of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and of distance to the hyperplane.
- Since the scalar product is linear, the SVM is a linear method.

But

Using a nonlinear function instead, we can make the classifier nonlinear.

Basic Idea

Something Notable

- The SVM uses the scalar product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ as a measure of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and of distance to the hyperplane.
- Since the scalar product is linear, the SVM is a linear method.

But

Using a nonlinear function instead, we can make the classifier nonlinear.

Cinvestav

We do this by defining the following map

Nonlinear transformations

Given a series of nonlinear transformations

$$\{\phi_i(\boldsymbol{x})\}_{i=1}^{m}$$

from input space to the feature space.

We can define the decision surface as

$$\sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) + w_0 = 0$$

.

We do this by defining the following map

Nonlinear transformations

Given a series of nonlinear transformations

$$\left\{\phi_i\left(\boldsymbol{x}\right)\right\}_{i=1}^{m}$$

from input space to the feature space.

We can define the decision surface as

$$\sum_{i=1}^{m} w_i \phi_i\left(\boldsymbol{x}\right) + w_0 = 0$$

.

Cinvestav

This allows us to define

The following vector

$$\phi(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \phi_1(\boldsymbol{x}), \cdots, \phi_m(\boldsymbol{x}))^T$$

That represents the mapping.

Outline

Cinvestav

Finally

We define the decision surface as

$$\boldsymbol{w}^T \phi(\boldsymbol{x}) = 0 \quad (14)$$

We now seek "linear" separability of features, we may write

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \phi(\boldsymbol{x}_i) \quad (15)$$

Thus, we finish with the following decision surface

$$\sum_{i=1}^{N} \alpha_i d_i \phi^T(\boldsymbol{x}_i) \phi(\boldsymbol{x}) = 0 \quad (16)$$

Cinvestav

Finally

We define the decision surface as

$$\boldsymbol{w}^T \phi(\boldsymbol{x}) = 0 \tag{14}$$

We now seek "linear" separability of features, we may write

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \phi(\boldsymbol{x}_i) \tag{15}$$

Thus, we finish with the following decision surface

$$\sum_{i=1}^{N} \alpha_i d_i \phi^T(\boldsymbol{x}_i) \phi(\boldsymbol{x}) = 0 \tag{16}$$

Cinvestav

Finally

We define the decision surface as

$$\boldsymbol{w}^T \phi\left(\boldsymbol{x}\right) = 0 \tag{14}$$

We now seek "linear" separability of features, we may write

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i d_i \phi\left(\boldsymbol{x}_i\right) \tag{15}$$

Thus, we finish with the following decision surface

$$\sum_{i=1}^{N} \alpha_i d_i \phi^T\left(\boldsymbol{x}_i\right) \phi\left(\boldsymbol{x}\right) = 0 \tag{16}$$

Thus

The term $\phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x})$

It represents the inner product of two vectors induced in the feature space induced by the input patterns.

We can introduce the inner-product kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x}) = \sum_{j=0}^{m} \phi_j(\boldsymbol{x}_i)\,\phi_j(\boldsymbol{x}) \quad (17)$$

Property: Symmetry

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_i) \quad (18)$$

Cinvestav

Thus

The term $\phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x})$

It represents the inner product of two vectors induced in the feature space induced by the input patterns.

We can introduce the inner-product kernel

$$K(\boldsymbol{x}_i,\boldsymbol{x}) = \phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x}) = \sum_{j=0}^{m}\phi_j(\boldsymbol{x}_i)\,\phi_j(\boldsymbol{x}) \tag{17}$$

Property: Symmetry

$$K(\boldsymbol{x}_i,\boldsymbol{x}) = K(\boldsymbol{x},\boldsymbol{x}_i) \tag{18}$$

Cinvestav

Thus

The term $\phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x})$

It represents the inner product of two vectors induced in the feature space induced by the input patterns.

We can introduce the inner-product kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \phi^T(\boldsymbol{x}_i)\,\phi(\boldsymbol{x}) = \sum_{j=0}^{m} \phi_j(\boldsymbol{x}_i)\,\phi_j(\boldsymbol{x}) \tag{17}$$

Property: Symmetry

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{18}$$

Cinvestav

This allows to redefine the optimal hyperplane

We get

$$\sum_{i=1}^{N} \alpha_i d_i K\left(\boldsymbol{x}_i, \boldsymbol{x}\right) = 0 \quad (19)$$

Something Notable

Using kernels, we can avoid to go from:

$$\text{Input Space} \Longrightarrow \text{Mapping Space} \Longrightarrow \text{Inner Product} \quad (20)$$

By directly going from

$$\text{Input Space} \Longrightarrow \text{Inner Product} \quad (21)$$

This allows to redefine the optimal hyperplane

We get

$$\sum_{i=1}^{N} \alpha_i d_i K(\boldsymbol{x}_i, \boldsymbol{x}) = 0 \tag{19}$$

Something Notable

Using kernels, we can avoid to go from:

$$\text{Input Space} \Longrightarrow \text{Mapping Space} \Longrightarrow \text{Inner Product} \tag{20}$$

By directly going from

$$\text{Input Space} \Longrightarrow \text{Inner Product} \tag{21}$$

This allows to redefine the optimal hyperplane

We get

$$\sum_{i=1}^{N} \alpha_i d_i K\left(\boldsymbol{x}_i, \boldsymbol{x}\right) = 0 \quad (19)$$

Something Notable

Using kernels, we can avoid to go from:

$$\text{Input Space} \Longrightarrow \text{Mapping Space} \Longrightarrow \text{Inner Product} \quad (20)$$

By directly going from

$$\text{Input Space} \Longrightarrow \text{Inner Product} \quad (21)$$

Important

Something Notable

The expansion of (Eq. 17) for the inner-product kernel $K(\boldsymbol{x}_i, \boldsymbol{x})$ is an important special case of that arises in functional analysis.

Mercer's Theorem

Mercer's Theorem

Let $K(\boldsymbol{x}, \boldsymbol{x}')$ be a continuous symmetric kernel that is defined in the closed interval $\boldsymbol{a} \leq \boldsymbol{x} \leq \boldsymbol{b}$ and likewise for $\boldsymbol{x}'$. The kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ can be expanded in the series

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\mathbf{x}') \tag{22}$$

With

Positive coefficients, $\lambda_i > 0$ for all i.

Mercer's Theorem

Mercer's Theorem

Let $K(\boldsymbol{x}, \boldsymbol{x'})$ be a continuous symmetric kernel that is defined in the closed interval $\boldsymbol{a} \leq \boldsymbol{x} \leq \boldsymbol{b}$ and likewise for $\boldsymbol{x'}$. The kernel $K(\boldsymbol{x}, \boldsymbol{x'})$ can be expanded in the series

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\mathbf{x}') \quad (22)$$

With

Positive coefficients, $\lambda_i > 0$ for all i.

Mercer's Theorem

For this expression to be valid and or it to converge absolutely and uniformly

It is necessary and sufficient that the condition

$$\int_a^b \int_a^b K\left(\boldsymbol{x}, \boldsymbol{x'}\right) \psi\left(\boldsymbol{x}\right) \psi\left(\boldsymbol{x'}\right) d\boldsymbol{x} d\boldsymbol{x'} \geq 0 \tag{23}$$

holds for all ψ such that $\int_a^b \psi^2\left(\boldsymbol{x}\right) d\boldsymbol{x} < \infty$(Example of a quadratic norm for functions).

Cinvestav

Remarks

First

The functions $\phi_i(\boldsymbol{x})$ are called eigenfunctions of the expansion and the numbers λ_i are called eigenvalues.

Second

The fact that all of the eigenvalues are positive means that the kernel $K(x, x')$ is positive definite.

Remarks

First

The functions $\phi_i(\boldsymbol{x})$ are called eigenfunctions of the expansion and the numbers λ_i are called eigenvalues.

Second

The fact that all of the eigenvalues are positive means that the kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ is positive definite.

Not only that

We have that

For $\lambda_i \neq 1$, the ith image of $\sqrt{\lambda_i}\phi_i(\boldsymbol{x})$ induced in the feature space by the input vector $\boldsymbol{x}$ is an eigenfunction of the expansion.

In theory

The dimensionality of the feature space (i.e., the number of eigenvalues/ eigenfunctions) can be infinitely large.

Not only that

We have that

For $\lambda_i \neq 1$, the ith image of $\sqrt{\lambda_i}\phi_i(\boldsymbol{x})$ induced in the feature space by the input vector $\boldsymbol{x}$ is an eigenfunction of the expansion.

In theory

The dimensionality of the feature space (i.e., the number of eigenvalues/ eigenfunctions) can be infinitely large.

Outline

Example

Assume

$$\boldsymbol{x} \in \mathbb{R} \to \boldsymbol{y} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

We can show that

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = \left(\boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2$$

Cinvestav

Example

Assume

$$\boldsymbol{x} \in \mathbb{R} \rightarrow \boldsymbol{y} = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{bmatrix}$$

We can show that

$$\boldsymbol{y}_i^T \boldsymbol{y}_j = \left(\boldsymbol{x}_i^T \boldsymbol{x}_j \right)^2$$

Cinvestav

Example of Kernels

Polynomials

$$k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z} + 1)^q \; q > 0$$

Radial Basis Functions

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right)$$

Hyperbolic Tangents

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right)$$

Example of Kernels

Polynomials

$$k(\boldsymbol{x}, \boldsymbol{z}) = \left(\boldsymbol{x}^T \boldsymbol{z} + 1\right)^q \; q > 0$$

Radial Basis Functions

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{z}||^2}{\sigma^2}\right)$$

Hyperbolic Tangents

$$k(\boldsymbol{x}, \boldsymbol{z}) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right)$$

Example of Kernels

Polynomials

$$k\left(\boldsymbol{x}, \boldsymbol{z}\right) = \left(\boldsymbol{x}^T \boldsymbol{z} + 1\right)^q \; q > 0$$

Radial Basis Functions

$$k\left(\boldsymbol{x}, \boldsymbol{z}\right) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right)$$

Hyperbolic Tangents

$$k\left(\boldsymbol{x}, \boldsymbol{z}\right) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right)$$

Outline

Cinvestav

Now, How to select a Kernel?

We have a problem

Selecting a specific kernel and parameters is usually done in a try-and-see manner.

Thus

In general, the Radial Basis Functions kernel is a reasonable first choice.

Then

if this fails, we can try the other possible kernels.

Cinvestav

Now, How to select a Kernel?

We have a problem

Selecting a specific kernel and parameters is usually done in a try-and-see manner.

Thus

In general, the Radial Basis Functions kernel is a reasonable first choice.

Then

if this fails, we can try the other possible kernels.

Now, How to select a Kernel?

We have a problem

Selecting a specific kernel and parameters is usually done in a try-and-see manner.

Thus

In general, the Radial Basis Functions kernel is a reasonable first choice.

Then

if this fails, we can try the other possible kernels.

Thus, we have something like this

Step 1

Normalize the data.

Step 2

Use cross-validation to adjust the parameters of the selected kernel.

Step 3

Train against the entire dataset.

Thus, we have something like this

Step 1

Normalize the data.

Step 2

Use cross-validation to adjust the parameters of the selected kernel.

Step 3

Train against the entire dataset.

Thus, we have something like this

Step 1

Normalize the data.

Step 2

Use cross-validation to adjust the parameters of the selected kernel.

Step 3

Train against the entire dataset.

Outline

Cinvestav

Optimal Hyperplane for non-separable patterns

Important

We have been considering only problems where the classes are linearly separable.

Now

What happen when the patterns are not separable?

Thus, we can still build a separating hyperplane

But errors will happen in the classification... We need to minimize them...

Optimal Hyperplane for non-separable patterns

Important

We have been considering only problems where the classes are linearly separable.

Now

What happen when the patterns are not separable?

Thus, we can still build a separating hyperplane

But errors will happen in the classification... We need to minimize them...

Optimal Hyperplane for non-separable patterns

Important

We have been considering only problems where the classes are linearly separable.

Now

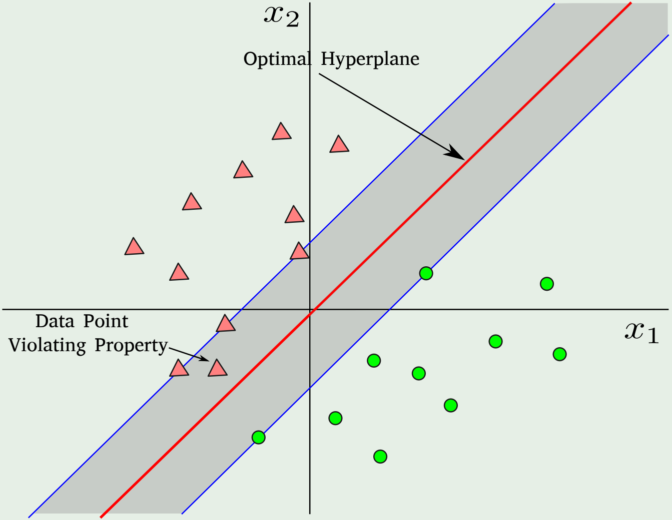
What happen when the patterns are not separable?

Thus, we can still build a separating hyperplane

But errors will happen in the classification... We need to minimize them...

What if the following happens

Some data points invade the "margin" space



Fixing the Problem - Corinna's Style

The margin of separation between classes is said to be soft if a data point $(\boldsymbol{x}_i, d_i)$ violates the following condition

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) \geq +1 \;\; i = 1, 2, ..., N \tag{24}$$

This violation can arise in one of two ways

The data point $(\boldsymbol{x}_i, d_i)$ falls inside the region of separation but on the right side of the decision surface - still correct classification.

Fixing the Problem - Corinna's Style

The margin of separation between classes is said to be soft if a data point $(\boldsymbol{x}_i, d_i)$ violates the following condition

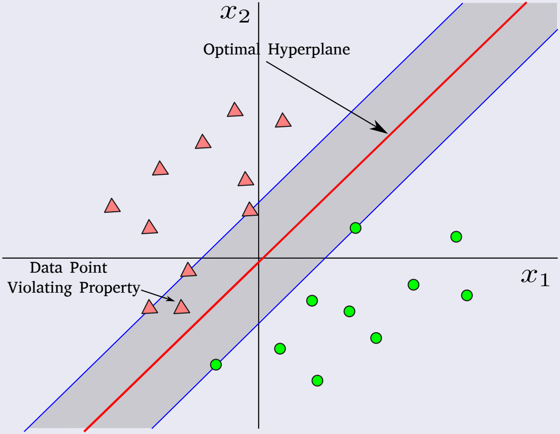
$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \geq +1 \;\; i = 1, 2, ..., N \tag{24}$$

This violation can arise in one of two ways

The data point $(\boldsymbol{x}_i, d_i)$ falls inside the region of separation but on the right side of the decision surface - still correct classification.

We have then

Example



Or...

This violation can arise in one of two ways

The data point $(\boldsymbol{x}_i, d_i)$ falls on the wrong side of the decision surface - incorrect classification.

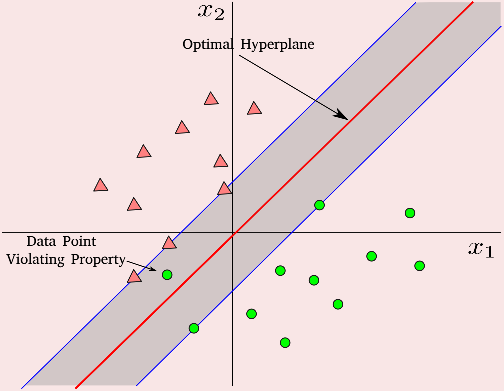
Example

Or...

This violation can arise in one of two ways

The data point $(\boldsymbol{x}_i, d_i)$ falls on the wrong side of the decision surface - incorrect classification.

Example



Solving the problem

What to do?

- We introduce a set of nonnegative scalar values $\{\xi_i\}_{i=1}^{N}$.

Introduce this into the decision rule

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) \geq 1 - \xi_i \; i = 1, 2, ..., N \tag{25}$$

Solving the problem

What to do?

- We introduce a set of nonnegative scalar values $\{\xi_i\}_{i=1}^{N}$.

Introduce this into the decision rule

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) \geq 1 - \xi_i \; i = 1, 2, ..., N \tag{25}$$

Solving the problem

What to do?

- We introduce a set of nonnegative scalar values $\{\xi_i\}_{i=1}^{N}$.

Introduce this into the decision rule

$$d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i \;\; i = 1, 2, ..., N \tag{25}$$

The ξ_i are called slack variables

What?

In 1995, Corinna Cortes and Vladimir N. Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.

Ok!!!

Instead of expecting to have constant margin for all the samples, the margin can change depending of the sample.

What do we have?

ξ_i measures the deviation of a data point from the ideal condition of pattern separability.

The ξ_i are called slack variables

What?

In 1995, Corinna Cortes and Vladimir N. Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.

Ok!!!

Instead of expecting to have constant margin for all the samples, the margin can change depending of the sample.

What do we have?

ξ_i measures the deviation of a data point from the ideal condition of pattern separability.

The ξ_i are called slack variables

What?

In 1995, Corinna Cortes and Vladimir N. Vapnik suggested a modified maximum margin idea that allows for mislabeled examples.

Ok!!!

Instead of expecting to have constant margin for all the samples, the margin can change depending of the sample.

What do we have?

ξ_i measures the deviation of a data point from the ideal condition of pattern separability.

Properties of ξ_i

What if?

- You have $0 \leq \xi_i \leq 1$

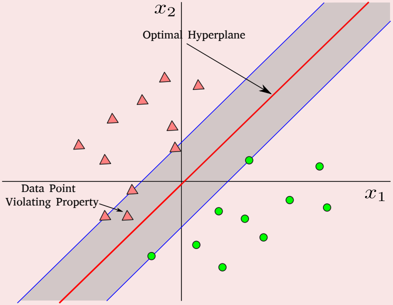
We have

Properties of ξ_i

What if?

- You have $0 \leq \xi_i \leq 1$

We have



Properties of ξ_i

What if?

- You have $\xi_i > 1$

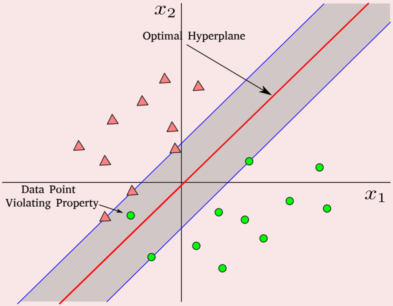
We have

Properties of ξ_i

What if?

- You have $\xi_i > 1$

We have



Support Vectors

We want

- Support vectors that satisfy equation (Eq. 25) even when $\xi_i > 0$

$$d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) \geq 1 - \xi_i \;\; i = 1, 2, ..., N$$

We want the following

We want to find an hyperplane

Such that average error is misclassified over all the samples

$$\frac{1}{N}\sum_{i=1}^{N}\mathrm{e}^{2} \tag{26}$$

Cinvestav

First Attempt Into Minimization

We can try the following

Given

$$I(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (27)$$

Minimize the following

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1) \quad (28)$$

with respect to the weight vector w subject to

1. $d_i \left(w^T x_i + b \right) \geq 1 - \xi_i \; i = 1, 2, ..., N$
2. $\|w\|^2 \leq C$ for a given C.

First Attempt Into Minimization

We can try the following

Given

$$I(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{27}$$

Minimize the following

$$\Phi(\boldsymbol{\xi}) = \sum_{i=1}^{N} I(\xi_i - 1) \tag{28}$$

with respect to the weight vector $\boldsymbol{w}$ subject to

1. $d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + b \right) \geq 1 - \xi_i \; i = 1, 2, ..., N$
2. $\|\boldsymbol{w}\|^2 \leq C$ for a given C.

Problem

Using this first attempt

Minimization of $\Phi(\boldsymbol{\xi})$ with respect to $\mathbf{w}$ is a non-convex optimization problem that is NP-complete.

Thus, we need to use an approximation, maybe

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i \tag{29}$$

Now, we simplify the computations by integrating the vector w

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{30}$$

Problem

Using this first attempt

Minimization of $\Phi(\boldsymbol{\xi})$ with respect to $\mathbf{w}$ is a non-convex optimization problem that is NP-complete.

Thus, we need to use an approximation, maybe

$$\Phi(\boldsymbol{\xi}) = \sum_{i=1}^{N} \xi_i \tag{29}$$

Now, we simplify the computations by integrating the vector w

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{30}$$

Problem

Using this first attempt

Minimization of $\Phi(\boldsymbol{\xi})$ with respect to $\mathbf{w}$ is a non-convex optimization problem that is NP-complete.

Thus, we need to use an approximation, maybe

$$\Phi(\boldsymbol{\xi}) = \sum_{i=1}^{N} \xi_i \tag{29}$$

Now, we simplify the computations by integrating the vector $\boldsymbol{w}$

$$\Phi(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N} \xi_i \tag{30}$$

Important

First

Minimizing the first term in (Eq. 30) is related to minimize the Vapnik–Chervonenkis dimension.

- Which is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a statistical classification algorithm.

Important

First

Minimizing the first term in (Eq. 30) is related to minimize the Vapnik–Chervonenkis dimension.

- Which is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a statistical classification algorithm.

Second

The second term $\sum_{i=1}^{N} \xi_i$ is an upper bound on the number of test errors.

Important

First

Minimizing the first term in (Eq. 30) is related to minimize the Vapnik–Chervonenkis dimension.

- Which is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a statistical classification algorithm.

Second

The second term $\sum_{i=1}^{N} \xi_i$ is an upper bound on the number of test errors.

Cinvestav

Some problems for the Parameter C

Little Problem

The parameter C has to be selected by the user.

Some problems for the Parameter C

Little Problem

The parameter C has to be selected by the user.

This can be done in two ways

1. The parameter C is determined experimentally via the standard use of a training! (validation) test set.
2. It is determined analytically by estimating the Vapnik–Chervonenkis dimension.

Some problems for the Parameter C

Little Problem

The parameter C has to be selected by the user.

This can be done in two ways

1. The parameter C is determined experimentally via the standard use of a training! (validation) test set.
2. It is determined analytically by estimating the Vapnik–Chervonenkis dimension.

Primal Problem

Problem, given samples $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^{N}$

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \Phi\left(\boldsymbol{w}, \boldsymbol{\xi}\right) = \min_{\mathbf{w},\boldsymbol{\xi}} \left\{ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i \right\}$$

$$\text{s.t. } d_i(\boldsymbol{w}^T\boldsymbol{x}_i + w_0) \geq 1 - \xi_i \text{ for } i = 1, \cdots, N$$

$$\xi_i \geq 0 \text{ for all } i$$

With C a user-specified positive parameter.

Outline

Cinvestav

Final Setup

Using Lagrange Multipliers and dual-primal method is possible to obtain the following setup

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^{N}$ that maximize the objective function

$$\min_{\alpha} Q(\alpha) = \min_{\alpha} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \boldsymbol{x}_j^T \boldsymbol{x}_i \right\}$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i d_i = 0 \quad (31)$$

$$0 \leq \alpha_i \leq C \text{ for } i = 1, \cdots, N \quad (32)$$

where C is a user-specified positive parameter.

Remarks

Something Notable

- Note that neither the slack variables nor their Lagrange multipliers appear in the dual problem.
- The dual problem for the case of non-separable patterns is thus similar to that for the simple case of linearly separable patterns

Remarks

Something Notable

- Note that neither the slack variables nor their Lagrange multipliers appear in the dual problem.
- The dual problem for the case of non-separable patterns is thus similar to that for the simple case of linearly separable patterns

The only big difference

Instead of using the constraint $\alpha_i \geq 0$, the new problem use the more stringent constraint $0 \leq \alpha_i \leq C$.

Remarks

Something Notable

- Note that neither the slack variables nor their Lagrange multipliers appear in the dual problem.
- The dual problem for the case of non-separable patterns is thus similar to that for the simple case of linearly separable patterns

The only big difference

Instead of using the constraint $\alpha_i \geq 0$, the new problem use the more stringent constraint $0 \leq \alpha_i \leq C$.

Note the following

$$\xi_i = 0 \text{ if } \alpha_i < C \quad (33)$$

Cinvestav

Remarks

Something Notable

- Note that neither the slack variables nor their Lagrange multipliers appear in the dual problem.
- The dual problem for the case of non-separable patterns is thus similar to that for the simple case of linearly separable patterns

The only big difference

Instead of using the constraint $\alpha_i \geq 0$, the new problem use the more stringent constraint $0 \leq \alpha_i \leq C$.

Note the following

$$\xi_i = 0 \text{ if } \alpha_i < C \tag{33}$$

Finally

The optimal solution for the weight vector $\boldsymbol{w}^*$

$$\boldsymbol{w}^* = \sum_{i=1}^{N_s} \alpha_i^* d_i \boldsymbol{x}_i$$

Where N_s is the number of support vectors.

Finally

The optimal solution for the weight vector $\boldsymbol{w}^*$

$$\boldsymbol{w}^* = \sum_{i=1}^{N_s} \alpha_i^* d_i \boldsymbol{x}_i$$

Where N_s is the number of support vectors.

In addition

The determination of the optimum values to that described before.

- $\mu_i \xi_i = 0$ for $i = 1, 2, ..., N$.

Cinvestav

Finally

The optimal solution for the weight vector $\boldsymbol{w}^*$

$$\boldsymbol{w}^* = \sum_{i=1}^{N_s} \alpha_i^* d_i \boldsymbol{x}_i$$

Where N_s is the number of support vectors.

In addition

The determination of the optimum values to that described before.

The KKT conditions are as follow

- $\alpha_i \left[d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_o \right) - 1 + \xi_i \right] = 0$ for $i = 1, 2, ..., N$.
- $\mu_i \xi_i = 0$ for $i = 1, 2, ..., N$.

Finally

The optimal solution for the weight vector $\boldsymbol{w}^*$

$$\boldsymbol{w}^* = \sum_{i=1}^{N_s} \alpha_i^* d_i \boldsymbol{x}_i$$

Where N_s is the number of support vectors.

In addition

The determination of the optimum values to that described before.

The KKT conditions are as follow

- $\alpha_i \left[d_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + w_o \right) - 1 + \xi_i \right] = 0$ for $i = 1, 2, ..., N$.
- $\mu_i \xi_i = 0$ for $i = 1, 2, ..., N$.

Where...

The μ_i are Lagrange multipliers

They are used to enforce the non-negativity of the slack variables ξ_i for all i.

Something Notable

At saddle point, the derivative of the Lagrangian function for the primal problem:

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_o\right) - 1 + \xi_i\right] - \sum_{i=1}^{N}\mu_i\xi_i \quad (34)$$

Where...

The μ_i are Lagrange multipliers

They are used to enforce the non-negativity of the slack variables ξ_i for all i.

Something Notable

At saddle point, the derivative of the Lagrangian function for the primal problem:

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[d_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + w_o\right) - 1 + \xi_i\right] - \sum_{i=1}^{N}\mu_i\xi_i \quad (34)$$

Cinvestav

Thus

We get

$$\alpha_i + \mu_i = C \tag{35}$$

Thus, we get if $\alpha_i < C$

Then $\mu_i > 0 \Rightarrow \xi_i = 0$

We may determine w_0

Using any data point (x_i, d_i) in the training set such that $0 \leq \alpha_i^* \leq C$. Then, given $\xi_i = 0$,

$$w_0^* = \frac{1}{d_i} - (w^*)^T x_i \tag{36}$$

Thus

We get

$$\alpha_i + \mu_i = C \tag{35}$$

Thus, we get if $\alpha_i < C$

Then $\mu_i > 0 \Rightarrow \xi_i = 0$

We may determine w_0

Using any data point (x_i, d_i) in the training set such that $0 \leq \alpha_i^* \leq C$. Then, given $\xi_i = 0$,

$$w_0^* = \frac{1}{d_i} - (w^*)^T x_i \tag{36}$$

Thus

We get

$$\alpha_i + \mu_i = C \tag{35}$$

Thus, we get if $\alpha_i < C$

Then $\mu_i > 0 \Rightarrow \xi_i = 0$

We may determine w_0

Using any data point $(\boldsymbol{x}_i, d_i)$ in the training set such that $0 \leq \alpha_i^* \leq C$. Then, given $\xi_i = 0$,

$$w_0^* = \frac{1}{d_i} - (\boldsymbol{w}^*)^T \boldsymbol{x}_i \tag{36}$$

Nevertheless

It is better

To take the mean value of w_0^* from all such data points in the training sample (Burges, 1998).

- BTW He has a great book in SVM's "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods"

Cinvestav