

# Linear Support Vector Machine (SVM)

Kun He (何琨)

Data Mining and Machine Learning Lab  
(John Hopcroft Lab)  
Huazhong University of Science & Technology

*brooklet60@hust.edu.cn*

2022年5月



# Table of contents

## 1 Overview

- Support Vector Machine (SVM)

## 2 Linear Support Vector Machine

- Margin
- Max Margin Classifier
- Support Vectors
- Example of SVM

## 3 SVM with Soft Constraints

- Soft SVM Overview
- Unconstrained Formulation
- Example of Soft SVM

## 4 SVM Summary

# Table of Contents

## 1 Overview

- Support Vector Machine (SVM)

## 2 Linear Support Vector Machine

- Margin
- Max Margin Classifier
- Support Vectors
- Example of SVM

## 3 SVM with Soft Constraints

- Soft SVM Overview
- Unconstrained Formulation
- Example of Soft SVM

## 4 SVM Summary

## Basic idea:

The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron. The Perceptron guaranteed that you find a hyperplane if it exists. The SVM finds the **maximum margin** separating hyperplane.

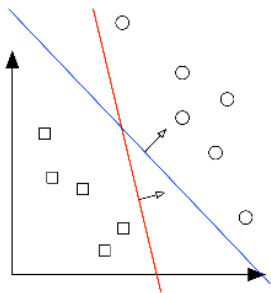


图: Two different separating hyperplanes for the same data set

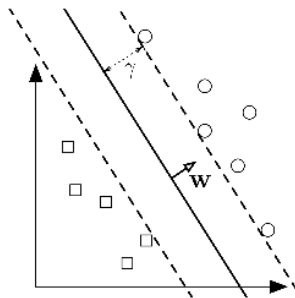
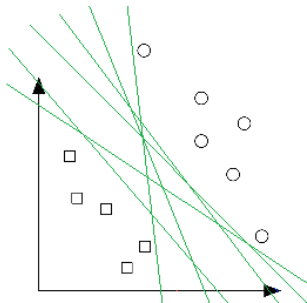


图: The maximum margin hyperplane

## Definition:

- Data set:  $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$
- Binary Classification Label:  $\mathbf{y}_i \in \{-1, +1\}$ ,  $i=1, \dots, N$
- Linear Classifier:  $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + \mathbf{b})$



Review on Perceptron: If the data is linearly separable, we can find many different hyperplanes through the Perceptron.

Question to ask: What is the best separating hyperplane?

图: Many separating hyperplanes for the same data set

# SVM:

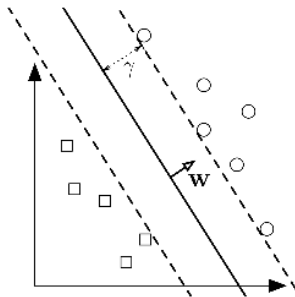


图: The maximum margin hyperplane

**SVM Answer:** the one that maximizes the distance to the closest data points from both classes and we say it is the hyperplane with **maximum margin**.

As show in the top figure, the margin,  $\gamma$ , is the distance from the hyperplane (solid line) to the closest points in either class (which touch the parallel dotted lines).

If the hyperplane is such that  $\gamma$  is maximized, it must lie right in the middle of the two classes.

# Table of Contents

## 1 Overview

- Support Vector Machine (SVM)

## 2 Linear Support Vector Machine

- Margin
- Max Margin Classifier
- Support Vectors
- Example of SVM

## 3 SVM with Soft Constraints

- Soft SVM Overview
- Unconstrained Formulation
- Example of Soft SVM

## 4 SVM Summary

## Margin:

- Hyperplane:  $\mathbf{H} = \{ \mathbf{x} | \mathbf{w}^T \mathbf{x} + \mathbf{b} = 0 \}$
- Distance from the hyperplane to the closest point across both classes :  $\gamma$

### Margin:

Consider some point  $\mathbf{x}$ . Let  $\mathbf{d}$  be the vector from  $\mathbf{H}$  to  $\mathbf{x}$  of minimum length. Let  $\mathbf{x}^P$  be the projection of  $\mathbf{x}$  onto  $\mathbf{H}$ . It follows then that:

$$\mathbf{x}^P = \mathbf{x} - \mathbf{d}$$

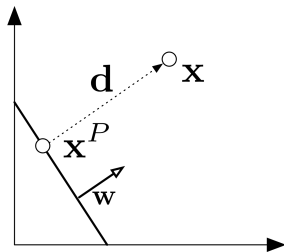
$\mathbf{d}$  is parallel to  $\mathbf{w}$ , so  $\mathbf{d} = \alpha \mathbf{w}$  for some  $\alpha \in \mathbb{R}$

$\mathbf{x}^P \in \mathcal{H}$  which implies  $\mathbf{w}^T \mathbf{x}^P + b = 0$

therefore

$$\mathbf{w}^T \mathbf{x}^P + b = \mathbf{w}^T (\mathbf{x} - \mathbf{d}) + b = \mathbf{w}^T (\mathbf{x} - \alpha \mathbf{w}) + b = 0$$

which implies :  $\alpha = \frac{\mathbf{w}^T \mathbf{x} + b}{\mathbf{w}^T \mathbf{w}}$





# Margin

## Margin:

So we have  $\alpha = \frac{\mathbf{w}^T \mathbf{x} + b}{\mathbf{w}^T \mathbf{w}}$ ,  $\mathbf{d} = \alpha \mathbf{w}$

The length of  $\mathbf{d}$  :

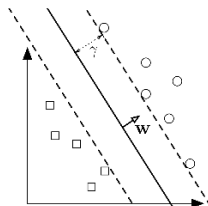
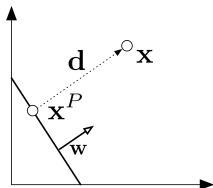
$$\|\mathbf{d}\|_2 = \sqrt{\mathbf{d}^T \mathbf{d}} = \sqrt{\alpha^2 \mathbf{w}^T \mathbf{w}} = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\sqrt{\mathbf{w}^T \mathbf{w}}} = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|_2}$$

Margin of  $\mathcal{H}$  with respect to  $\mathbf{D}$ :

$$\gamma(\mathbf{w}, b) = \min_{\mathbf{x} \in D} \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|_2}$$

By definition, the margin and hyperplane are scale invariant:

$$\gamma(\beta \mathbf{w}, \beta b) = \gamma(\mathbf{w}, b), \forall \beta \neq 0$$



# Max Margin Classifier

We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\underbrace{\max_{\mathbf{w}, b} \gamma(\mathbf{w}, b)}_{\text{maximize margin}} \text{ such that } \underbrace{\forall i \ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0}_{\text{separating hyperplane}}$$

If we plug in the definition of  $\gamma$  we obtain :

$$\underbrace{\max_{\mathbf{w}, b} \underbrace{\frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}_i \in D} |\mathbf{w}^T \mathbf{x}_i + b|}_{\gamma(\mathbf{w}, b)}}_{\text{maximize margin}} \quad s.t. \quad \underbrace{\forall i \ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0}_{\text{separating hyperplane}}$$

# Max Margin Classifier

$$\underbrace{\max_{\mathbf{w}, b} \underbrace{\frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}_i \in D} |\mathbf{w}^T \mathbf{x}_i + b|}_{\gamma(\mathbf{w}, b)}}_{\text{maximize margin}} \quad s.t. \quad \underbrace{\forall i \ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0}_{\text{separating hyperplane}}$$

Because the hyperplane is scale invariant,  $\gamma(\beta \mathbf{w}, \beta b) = \gamma(\mathbf{w}, b), \forall \beta \neq 0$ , we can fix the scale of  $\mathbf{w}, b$  anyway we want. So we can choose it such that:

$$\min_{\mathbf{x} \in D} |\mathbf{w}^T \mathbf{x} + b| = 1.$$

We can add this re-scaling as an equality constraint. Then our objective becomes:

$$\max_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|_2} \cdot 1 = \min_{\mathbf{w}, b} \|\mathbf{w}\|_2 = \min_{\mathbf{w}, b} \mathbf{w}^T \mathbf{w}$$

So from the objective above we know that is a convex quadratic function.

## Max Margin Classifier

From above discussed, our new optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \mathbf{w}^T \mathbf{w} \\ \text{s.t.} \quad & \forall i, y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 0, \min_i |\mathbf{w}^T \mathbf{x}_i + b| = 1 \end{aligned}$$

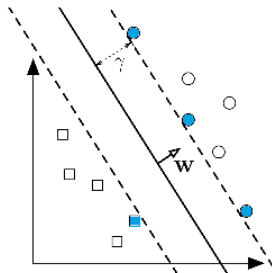
And then we can merge the constraints and get a much simpler formulation :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \mathbf{w}^T \mathbf{w} \\ \text{s.t.} \quad & \forall i \ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{aligned}$$

This new formulation is a quadratic optimization problem. The objective is quadratic and the constraints are all linear. We can solve it efficiently with any QCQP (Quadratically Constrained Quadratic Program) solver (unique solution).

With hyperplane scale invariant we can find the simplest hyperplane (where simpler means smaller  $\mathbf{w}^T \mathbf{w}$ ) such that all inputs lie at least 1 unit away from the hyperplane on the correct side.

## Support Vectors



Because the hyperplane is scale invariant, so we can re-scale  $\mathbf{w}$ ,  $b$  to make the distance of all points at least 1 unit. Thus, for the optimal  $\mathbf{w}$ ,  $b$  pair, some training points will have tight constraints, i.e.,

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1.$$

In the top figure we can clearly see these points are the blue points and they are on the dotted lines. We refer to these training points as support vectors. Support vectors are special because they are the training points that define the maximum margin of the hyperplane to the data set and they therefore determine the shape of the

## Example of SVM:

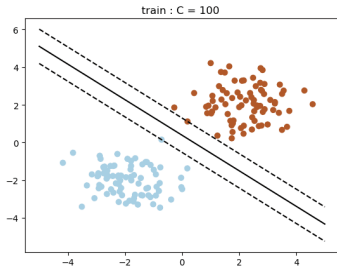


图: Training result

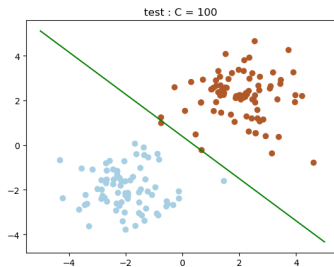


图: Test result

[SVM code click here](#)

# Table of Contents

## 1 Overview

- Support Vector Machine (SVM)

## 2 Linear Support Vector Machine

- Margin
- Max Margin Classifier
- Support Vectors
- Example of SVM

## 3 SVM with Soft Constraints

- Soft SVM Overview
- Unconstrained Formulation
- Example of Soft SVM

## 4 SVM Summary

## Soft SVM Overview:

If the data is low dimensional or some noise in the data it is often the case that there is no separating hyperplane between the two classes.

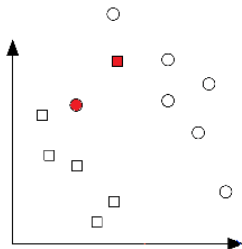


图: Not Linear Separate

For example, in the top figure, the red points may be noise points or the label of the points is wrong, it's obvious that there is no solution to the optimization problems stated above.



# Soft SVM Overview

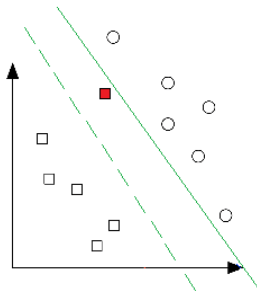


图: May Contain Noise

For another case, in the top figure, we can find a linear separating hyperplane (solid line) with the solution stated above. But this hyperplane's margin to the point is too small. Thus, we may think the red point is noise or the label is wrong.

If we don't consider the red point, we may find the other better hyperplane (dotted line) and this may be the best hyperplane.

## misMatched Count Loss

For the case discussed above, we can ignore these noise points that mismatched and treat them as a loss. We count the number of mismatched points and add them to objective. We will get :

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n [y_i \neq \text{sign}(\mathbf{w}^T \mathbf{x}_i + b)] \\ \text{s.t.} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \text{if } y_i \text{ is correct} \\ & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq -\infty, \quad \text{if } y_i \text{ is incorrect} \end{aligned} \quad (1)$$

From the constraint (1) , if  $y_i$  is incorrect, we do not add any constraint to the noise points.

And in the objective above,  $C$  is trade-off of **large margin & noise tolerance**.

## Unconstrained Formulation:

In the previous page, we define the misMatched count loss,  $[\cdot]$  to count the number of mismatched. However it is non-linear and discontinuous. This is very inconvenient for our calculations. Thus, we can fix that by using a linear constraint slack variables  $\xi_i$  to record *margin violation* instead of the mismatched count:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i \ y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \forall i \ \xi_i \geq 0 \end{aligned}$$

The slack variable  $\xi_i$  allows the input  $\mathbf{x}_i$  to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack".

- If  $C$  is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane.
- If  $C$  is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower  $\|\mathbf{w}\|_2^2$ ) solution.

# Unconstrained Formulation

## Loss $\xi_i$

Let us consider the value of  $\xi_i$  for the case of  $C \neq 0$ . We can consider  $\xi_i$  as the loss and the objective will always try to minimize  $\xi_i$  as much as possible, the equation must hold as an equality and we have:

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b), & \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1 \\ 0, & \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1. \end{cases} \quad (2)$$

This is equivalent to the following closed form:

$$\xi_i = \max(1 - y_i(\mathbf{w}^T \mathbf{x}_i + b), 0).$$

# Unconstrained Formulation

## Unconstrained Formulation

If we plug this closed form into the objective of our SVM optimization problem, we obtain the following unconstrained version as loss function and regularizer:

$$\min_{\mathbf{w}, b} \underbrace{\mathbf{w}^T \mathbf{w}}_{l_2\text{-regularizer}} + C \sum_{i=1}^n \underbrace{\max [1 - y_i(\mathbf{w}^T \mathbf{x} + b), 0]}_{\text{hinge-loss}}$$

This formulation allows us to optimize the SVM parameters  $(\mathbf{w}, b)$  just like logistic regression (e.g. through gradient descent). The only difference is that we have the **hinge-loss** instead of the **logistic loss**.

# Example of Soft SVM

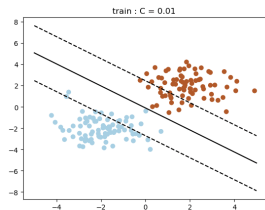


图: train :  $C=0.01$

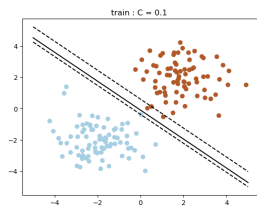


图: train :  $C=0.1$

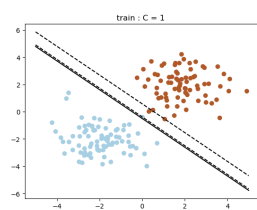


图: train :  $C=1$

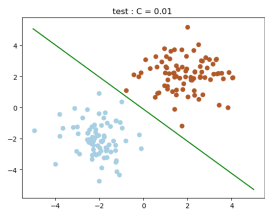


图: test :  $C=0.01$

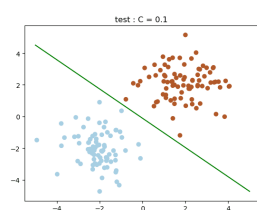


图: test :  $C=0.1$

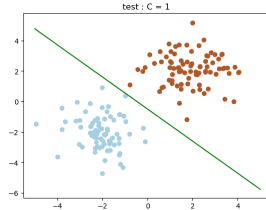


图: test :  $C=1$

# Example of Soft SVM

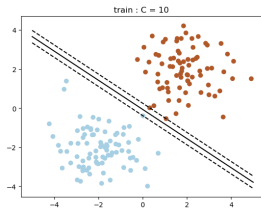


图: train :  $C=10$

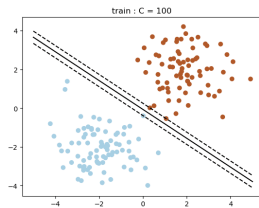


图: train :  $C=100$

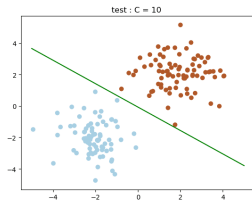


图: test :  $C=10$

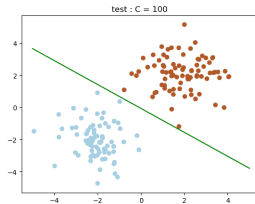


图: test :  $C=100$

## Example of Soft SVM

From the figures above, we can see different  $C$  may get different separating hyperplane. If  $C$  is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane. But it may cause overfit. If  $C$  is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower  $\|\mathbf{w}\|_2^2$ ) solution.

[Soft SVM code click here](#)



# Table of Contents

## 1 Overview

- Support Vector Machine (SVM)

## 2 Linear Support Vector Machine

- Margin
- Max Margin Classifier
- Support Vectors
- Example of SVM

## 3 SVM with Soft Constraints

- Soft SVM Overview
- Unconstrained Formulation
- Example of Soft SVM

## 4 SVM Summary

## SVM Summary

- The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron. And SVM finds the maximum margin separating hyperplane.
- Margin is the distance from the hyperplane to the closest point across both classes.
- Max margin classifier is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane.
- For the optimal  $\mathbf{w}$ ,  $b$  pair, some training points will have tight constraints, i.e.  $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$ . We refer to these training points as **support vectors**.
- The SVMs are convex quadratic function and can be solved with QCQP solver.
- With slack variables added into the objective, we can get the unconstrained SVM formulation: SVM with soft constraints.

# The End