西交利物浦大學
Xi'an Jiaotong-Liverpool University

Pattern Recognition

Lecture 15. Support Vector Machine

Dr. Shanshan ZHAO

School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University

Academic Year 2023-2024

Table of Contents

1. SVM Intuition
2. Formulization
3. Lagrange Duality
4. Kernel Trick
5. Soft Margin

Outline

1 SVM Intuition

2 Formulization

3 Lagrange Duality

4 Kernel Trick

5 Soft Margin

Introduction

- In the general case, the problem of finding linear discriminant functions can be formulated as a problem of **optimizing a criterion function**.
- Among all hyperplanes separating the data, there exists **a unique one yielding the maximum margin** of separation between the classes.

Binary Classification

Given training data (x_i, y_i) for $i = 1...N$, with $x_i \in \mathcal{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such that

$$f(x_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

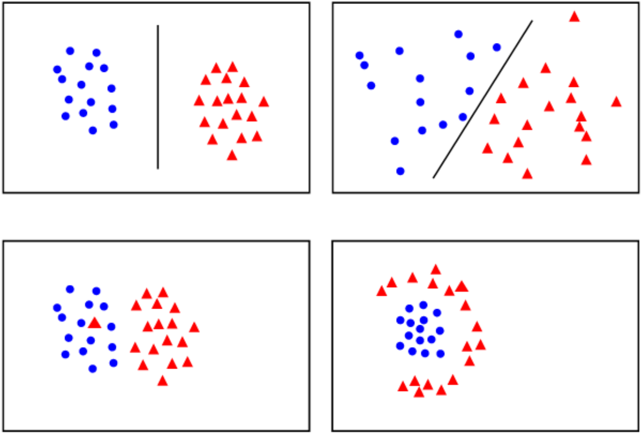
i.e.

$y_i f(x_i) > 0 \rightarrow$ a correction classification

Linear separability

linearly separable

not linearly separable

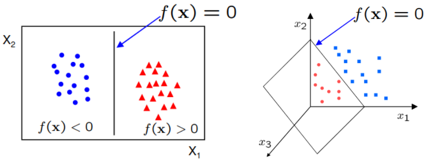


Linear Classifiers

A linear classifier has the form

$$f(x) = w^T x + b$$

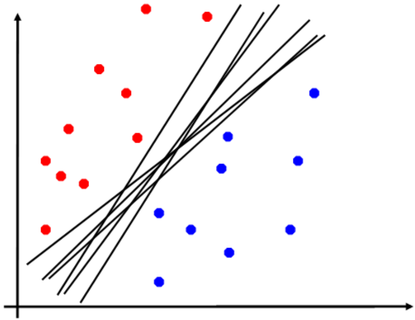
- w is **normal**(vertical) to the line, and the b is the bias/intercept
  - *whether the positive of f(x) is on the right or left of the line depends on the sign of the first parameter in vector* w.
- w is known as the weight vector.



(a) a line in 2D (b) a plane in 3D

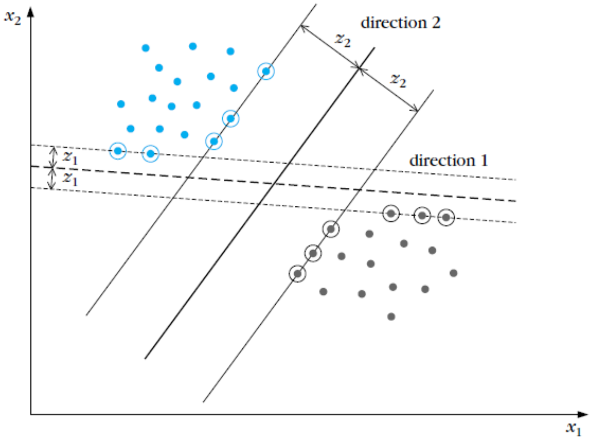
Linear Classifiers

- If training data is linearly separable, perceptron is guaranteed to find some linear separator/decision hyperplane.
- Which of these is optimal?



SVM Intuition

a very sensible choice for the hyperplane classifier would be the one that leaves the maximum margin from both classes.



Outline

1. SVM Intuition
2. Formulization
3. Lagrange Duality
4. Kernel Trick
5. Soft Margin

Margin

margin: a hyperplane leaves from both classes.
Our goal is to search for the direction that gives the maximum possible margin.
Recall that the distance of a point from a hyperplane is given by

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

We can scale w, b so that the value of $g(x)$, at the nearest points in c_1, c_2 (circled in figure).

SVM objective

We can scale w, w_0 so that the value of $g(x)$, at the nearest points in c_1, c_2 (circled in figure1), is equal to 1 for class c_1 and equal to -1 for class c_2, which is equivalent with

- 1. Having a margin of $\frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||}$
- 2. Requiring that

$$\begin{cases} w^T x + b \geq 1, & \forall x \in c_1 \\ w^T x + b \leq -1, & \forall x \in c_2 \end{cases}$$

- The support vectors lie on either of the two hyperplanes, that is

$$w^T x + b = \pm 1$$

Objective: Maximizing the margins

Maximize the margin (I) –Primal form(*)

- Optimization (Quadratic Programming) (known as a **Primal problem**.

$$\begin{cases} \text{minimize} \quad J(w,b) = \frac{1}{2}||w||^2 \\ \text{subject to} \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, ..., N \end{cases} \tag{1}$$

- Minimizing the norm makes the margin maximum

It belongs to the convex programming family of problems, since the cost function is convex and the constraints are linear and define a convex set of feasible solutions. Such problems can be solved by considering the socalled Lagrangian duality.

Outline

1 SVM Intuition

2 Formulization

3 Lagrange Duality

4 Kernel Trick

5 Soft Margin

Maximize the margin (II) –Dual form(*)

- The objective in Eq. (1) is a standard quadratic programming problem.
- Let $\boldsymbol{\lambda} \in \mathscr{R}^N$ be the dual variables, corresponding to Lagrange multipliers that enforce the N inequality constraints.

The generalized Lagrangian is given below

$$\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \lambda_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1] \qquad (2)$$

- where $\boldsymbol{\lambda}$ is the Lagrange multiplier, $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_N)^T$.
- we are **minimizing** with respect to $\boldsymbol{w}$ and b, and **maximizing** with respect to $\boldsymbol{\lambda}$.

Dual problem *

We are **minimizing** with respect to $\boldsymbol{w}$ and b, and **maximizing** with respect to $\boldsymbol{\lambda}$.

The dual problem is

$$\max_{\boldsymbol{\lambda} \geq 0} \min_{\boldsymbol{w}, b} \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{\lambda})$$

Explaination:

- Appendix C.4 from *Pattern Recognition by Segios*
- cs229-notes3 by *Andrew Ng*, page 7-9.

`https://core.xjtlu.edu.cn/mod/folder/view.php?id=69950`

Maximize the margin (II) –Dual form(*)

Lets find the dual form of the problem. To do so, we need to first minimize $L(\boldsymbol{w}, b, \boldsymbol{\lambda})$ with respect to w and b (for fixed λ), which we'll do by setting the derivatives of L with respect to $\boldsymbol{w}$ and b to zero. We obtain the following two conditions

$$\frac{\partial}{\partial \boldsymbol{w}} \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = 0 \tag{3}$$

$$\frac{\partial}{\partial b} \mathscr{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = 0 \tag{4}$$

$$\tag{5}$$

Combining (3) (4) and (2), results in

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i \tag{6}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{7}$$

Support Vectors

The Lagrange multipliers can be either zero or positive (Appendix C).Thus,the vector parameter $\boldsymbol{w}$ of the optimal solution is a linear combination of $N_s \leq N$ feature vectors that are associated with $\lambda_i \neq 0$. That is,

$$\boldsymbol{w} = \sum_{i=1}^{N_s} \lambda_i y_i \boldsymbol{x}_i$$

These are known as **support vectors** and the optimum hyperplane classifier as a support vector machine (SVM). As it is pointed out in Appendix C, a nonzero Lagrange multiplier corresponds to a so called active constraint.

Hence, as the set of constraints in equation (7) suggests for $\lambda_i \neq 0$, the support vectors lie on either of the two hyperplanes, that is,

$$w^T x + b = \pm 1$$

Maximize the margin (II) –Dual form(*)

Plugging equation (6) and (7) into Lagrangian Equation (2) yields the following

$$\begin{aligned}
\mathscr{L}(w,b,\lambda) &= \frac{1}{2}w^Tw - \sum_{i=1}^{N}\lambda_i y_i w^T x_i - \sum_{i=1}^{N}\lambda_i y_i b + \sum_{i=1}^{N}\lambda_i \\
&= \frac{1}{2}w^Tw - w^Tw - 0 + \sum_{i=1}^{N}\lambda_i \\
&= -\frac{1}{2}w^Tw + \sum_{i=1}^{N}\lambda_i \\
&= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{n=1}^{N}\lambda_i
\end{aligned}$$

Recall that we got to the equation above by minimizing L with respect to $\boldsymbol{w}$ and b. Putting this together with the constraints $\lambda_i \geq 0$ and the constraint $\sum_{i=1}^{N} \lambda_i y_i = 0$, we obtain the following dual optimization problem:

$$\max_{\boldsymbol{\lambda}} W(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \right) \tag{8}$$

$$\text{subject to} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{9}$$

$$\boldsymbol{\lambda} \geq 0 \tag{10}$$

We should be able to verify that the Karush-Kuhn-Tucker (KKT) conditions to hold are indeed satisfied in our optimization problem.

- Hence, we can solve the dual in lieu of solving the primal problem. Specifically, in the dual problem above, we have a maximization problem in which the parameters are the λ_i.
- the specific algorithm that we're going to use to solve the dual problem is : **SMO**.
- Most materials leave this part (how to find λ) to the end, but turns to introduce the kernels. *Don't get lost.*
- Details of *SMO* algorighm: cs229-notes3 by *Andrew Ng*, page 20-25.
- Let's leave it alone, as well :)

- If we are indeed able to solve it (i.e., find the λ_i that maximize $W(\boldsymbol{\lambda})$ subject to the constraints)
- then we can use Equation (6) $\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i$ to go back and find the optimal $\boldsymbol{w}$ as a function of the $\boldsymbol{\lambda}$.
- Having found $\boldsymbol{w}^*$, by considering the primal problem, it is also straightforward to find the optimal value for the intercept term b.
- In practice, b is computed as an average value obtained using all conditions of this type.

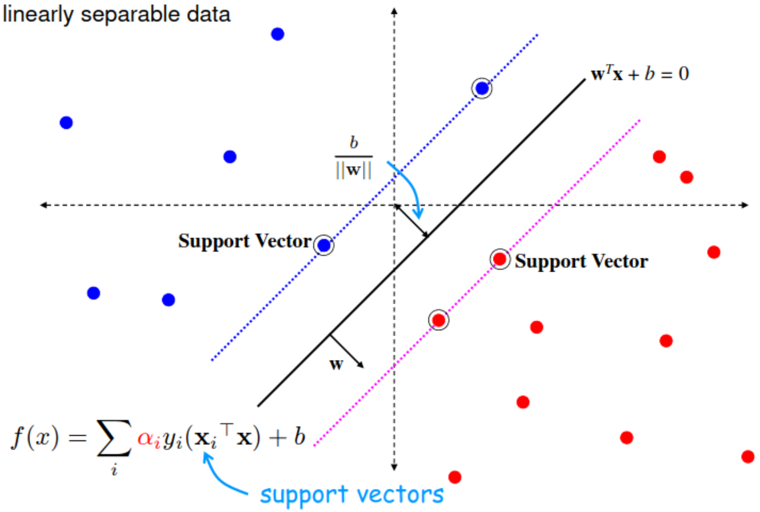
SVM

In practice, b is computed as an average value obtained using all conditions of this type.

$$
\begin{aligned}
b &= \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i) \\
&= \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(y_i - \sum_{j\in\mathcal{S}}\lambda_j y_j \boldsymbol{x}_j^T\boldsymbol{x}_i) \\
&= \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(y_i - \sum_{j\in\mathcal{S}}\lambda_j y_j \langle\boldsymbol{x}_j, \boldsymbol{x}_i\rangle)
\end{aligned}
$$

where $\mathcal{S}$ is the set of support vectors. We end up with the equivalent optimization task.

SVM


linearly separable data
$\mathbf{w}^T\mathbf{x} + b = 0$
$\frac{b}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}$
$f(x) = \sum_i \alpha_i y_i(\mathbf{x}_i{}^\top \mathbf{x}) + b$
support vectors

Outline

1. SVM Intuition
2. Formulization
3. Lagrange Duality
4. Kernel Trick
5. Soft Margin

- Suppose we've fit our model's parameters to a training set, and now wish to **make a prediction at a new point input** x.
- We would then calculate $w^T x + b$, and predict $y = 1$ if and only if this quantity is bigger than zero. With equation (6), this quantity can also be written:

$$\boldsymbol{w}^T \boldsymbol{x} + b = (\sum_{i=1}^{N_s} \lambda_i y_i \boldsymbol{x}_i)^T \boldsymbol{x} + b \tag{11}$$

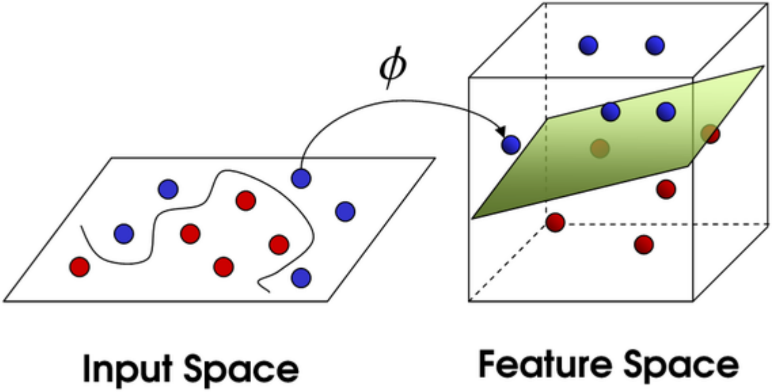
$$= \sum_{i=1}^{N_s} \lambda_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b \tag{12}$$

Hence, if we've found the λ_i, in order to make a prediction, we have to calculate a quantity that depends only on the **inner product** between x and the points in the training set.

- Moreover, we saw earlier that the λ_i will all be zero except for the support vectors.
- Thus, many of the terms in the sum above will be zero, and we really need to find only the inner products between x and the support vectors (of which there is often only a small number) in order calculate equation (12) and make our prediction.
- What if we have non-linear cases in the original space?
  - We discussed that we can map x into higher space where exists a linear hyperplane.

Kernel trick : Feature mapping

- Rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$.



SVM : Kernel trick

- Rather than applying SVMs using the original input attributes x, we may instead want to learn using some features $\phi(x)$.
- To do so, we simply need to go over our previous algorithm, and replace x everywhere in it with $\phi(x)$.
- Since the algorithm can be written entirely in terms of the inner products $\langle x, z\rangle$, this means that we would replace all those inner products with $\langle\phi(x), \phi(z)\rangle$.

SVM : Kernel trick

- Both the quadratic programming problem and the final decision function

$$g(x) = \text{sign}(\sum_{i=1}^{n} \lambda_i y_i \langle x \cdot x_i \rangle + b) \tag{13}$$

depend only on the dot procucts.

- We can generalize this result to the non-linear case by mapping the original input space into some other space $\mathscr{F}$ using a non-linear map $\phi : \mathscr{R}^d \to \mathscr{F}$ and perform the linear algorithm in the $\mathscr{F}$ space which only requires the inner products.

$$k(x, y) = \phi(x)^T \phi(y)$$

SVM : Kernel trick

- This results in the non-linear decision function of the form

$$g(x) = \text{sign}\left(\sum_{i=1}^{n} \lambda_i y_i k(x, x_i) + b\right) \tag{14}$$

  where the parameters λ_i are computed as the solution of the quadratic programming problem.
- In the original input space, the hyperplane corresponds to a non-linear decision function whose form is determined by the kernel.
- **We can use $k(x, x')$ directly for computation without transforming x and x', as long as ϕ exists.**

Example 1:

Suppose $x, z \in \mathscr{R}^2$, i.e., $x = [x_1, x_2]^T$, $z = [z_1, z_2]^T$, consider

$$k(x, z) = (x^T z)^2$$

$k(x, z)$ would be a valid kernel if we can find a projection function $\phi(x)$ that satisfy $k(x, z) = \phi(x)^T \phi(z)$

We may check $\phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$ *(yes, this is kind of cheating, but we just want to understand how kernel function works for now.)*

$$\begin{aligned} \phi(a)^T\phi(b) &= [a_1^2, \sqrt{2}a_1a_2, a_2^2]^T[b_1^2, \sqrt{2}b_1b_2, b_2^2] \\ &= a_1^2b_2^2 + 2a_1a_2b_1b_2 + a_2^2b_2^2 \end{aligned}$$

$$\begin{aligned} k(a,b) &= (a^Tb)^2 \\ &= ([a_1,a_2]^T[b_1,b_2])^2 \\ &= (a_1b_1 + a_2b_2)^2 \\ &= a_1^2b_2^2 + 2a_1a_2b_1b_2 + a_2^2b_2^2 \end{aligned}$$

So,

$$k(a,b) = \phi(a)^T\phi(b)$$

Example 2.

what if $x, z \in \mathscr{R}^n$?
page 14 on *cs229-notes3 by Andrew Ng*

Kernel : inner product

Polynomial kernels

$\forall x, x' \in \mathscr{R}^2$,

$$k(x, x') = (1 + x^T x')^2 = (1 + x_1 x_1' + x_2 x_2')^2$$
$$= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'$$

$\Downarrow$ above is an inner product of two vectors

$$= \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_1'^2 \\ x_2'^2 \\ \sqrt{2}x_1' \\ \sqrt{2}x_2' \\ \sqrt{2}x_1' x_2' \end{bmatrix}$$
$$= z^T z'$$

with $z = \phi(x) = [1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2]$

Example

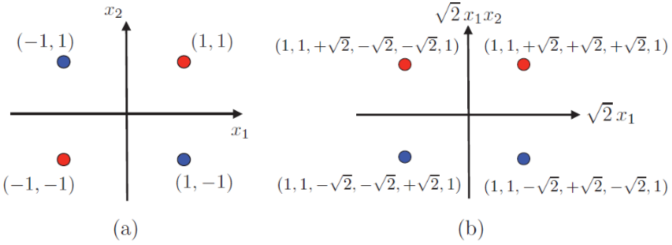


Figure 5.2 Illustration of the XOR classification problem and the use of polynomial kernels. (a) XOR problem linearly non-separable in the input space. (b) Linearly separable using second-degree polynomial kernel.

SVM : Kernel trick

- Even though $\mathscr{F}$ may be high-dimensional, a simple kernel $k(x, y)$ such as the following can be computed efficiently.

Polynomial	$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^p$
Sigmoidal	$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \theta)$
Radial basis function	$k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2))$

Figure 2: Common kernel functions

- Once a kernel function is chosen, we can substitute $\phi(x_i)$ for each training example x_i, and perform the optimal hyperplane algorithm in $\mathscr{F}$.

Kernel Matrix

consider some finite set of m points (not necessarily the training set) $\{x(1), \cdots, x(m)\}$, and let a square, m-by-m matrix K be defined so that its (i, j)-entry is given by $K_{ij} = K(x(i), x(j))$. This matrix is called the **Kernel matrix**.

- K must be symmetric
- K is positive semi-definite.

> It turns out to be a necessary and a sufficient condition for k to be a valid kernel (also called a Mercer kernel) if its correspoinding Kernel Matrix is symmetric positive semidefinite.

- proof also in the Andrew Ng's note.

Theorem(Mercer)

Let $k : \mathscr{R}^n \times \mathscr{R}^n \mapsto \mathscr{R}$ be given. Then for k to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x(1), \cdots, x(m)\}$, $(m < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

Given a function k, apart from trying to find a feature mapping ϕ that corresponds to it, this theorem therefore gives another way of testing if it is a valid kernel.

Outline

1 SVM Intuition

2 Formulization

3 Lagrange Duality

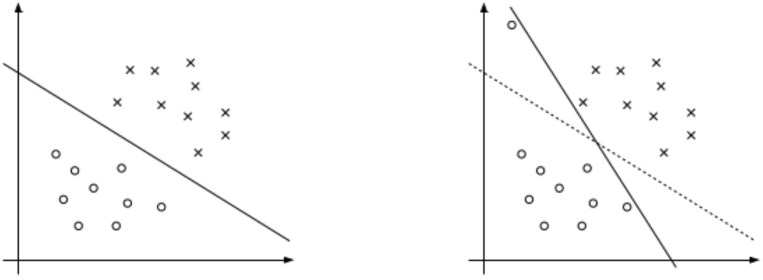
4 Kernel Trick

5 Soft Margin

Regularization and the non-separable case

- The derivation of the SVM as presented so far assumed that the data is linearly separable. While mapping data to a high dimensional feature space via ϕ does generally increase the likelihood that the data is separable, we can't guarantee that it always will be so.
- Also, in some cases it is not clear that finding a separating hyperplane is exactly what we'd want to do, since that might be susceptible to outliers.

Regularization and the non-separable case



the left figure below shows an optimal margin classifier, and when a single outlier is added in the upper-left region (right figure), it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin.

The training feature vectors now belong to one of the following three categories:

1. Vectors that fall outside the band and are correctly classified. These vectors comply with the constraints

$$y_i(w^T x_i + b) \geq 1, i = 1, 2, \cdots, N$$

2. Vectors falling inside the band and are correctly classified. These are the points placed in circles in Figure 3, and they satisfy the inequality

$$0 \leq y_i(w^T x_i + b) < 1$$

3. Vectors that are misclassified. They are enclosed by circles and obey the inequality

$$y_i(w^T x_i + b) < 0$$

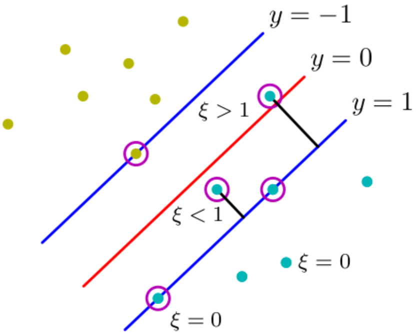
All three cases can be treated under a single type of constraints by introducing a new set of variables, namely,

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

slack variables

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

The first category of data corresponds to $\xi = 0$, the second to $0 < \xi_i \leq 1$, and the third to $\xi_i > 1$. The variables ξ_i are known as **slack variables**. The slack variables ξ_i are used to handle misclassified instances.



Regularization and the non-separable case

The goal now is to make the margin as large as possible but at the same time to keep the number of points with $\xi_i > 0$ as small as possible.

we reformulate our optimization

$$\begin{cases} \text{minimize} & J(w, b, \xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., N \\ & \xi_i \geq 0, i = 1, \cdots, N \end{cases} \tag{15}$$

- This is still a quadratic optimization problem and there is a unique minimum.
- Thus, samples are now permitted to have (functional) margin less than 1, and if a sample whose functional margin is $1 - \xi_i$, we would pay a cost of the objective function being increased by $C\xi_i$.

Regularization parameter C

- The term $C\sum_{i=1}^{n}\xi_i$ can be thought of as measuring some amount of misclassification where lowering the value of C corresponds to a smaller penalty for misclassification.
- The parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.
- Because any point that is misclassified has $\xi_i > 1$, it follows that $\sum_i \xi_i$ is an upper bound on the number of misclassified points.
- The parameter C is therefore analogous to a regularization coefficient because it controls the trade-off between minimizing training errors and controlling model complexity.
- In the limit $C \to \infty$, we will recover the earlier support vector machine for separable data.

we can form the Lagrangian:

$$\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \gamma) =$$

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i[y_i(\boldsymbol{w}^T\boldsymbol{x} + b) - 1 + \xi_i] - \sum_{i=1}^{N}\gamma_i\xi_i$$

the λ_i and γ_i are our Lagrange multipliers.(constrained to be ≥ 0).
We won't go through the derivation of the dual again in detail, but after setting the derivatives with respect to $\boldsymbol{w}$ and b to zero as before, substituting them back in, and simplifying,

we obtain the following dual form of the problem:

$$\max_{\boldsymbol{\lambda}} W(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j\right) \tag{16}$$

$$\text{subject to} \quad 0 \leq \lambda_i \leq C, i = 1, \cdots, N \tag{17}$$

$$\sum_{i=1}^{N}\lambda y_i = 0 \tag{18}$$

The corresponding set of KKT conditions can be given and used for testing for the convergence of the SMO algorithm.

SVM Training Methodology

- Training is formulated as an optimization problem
  - Dual problem - Makes use of Lagrange multipliers
  - Kernel trick
- Determination of the model parameters corresponds to a convex optimization problem
  - Solution is straightforward (local solution is a global optimum)
- Noisy labels – Soft Margin

Check the code.

Requirements

You should be able to describe

- the idea behind SVM.
- concept of the kernel trick, use examples.
- how the regularization parameter C works in SVM.

Thank You !

Q & A