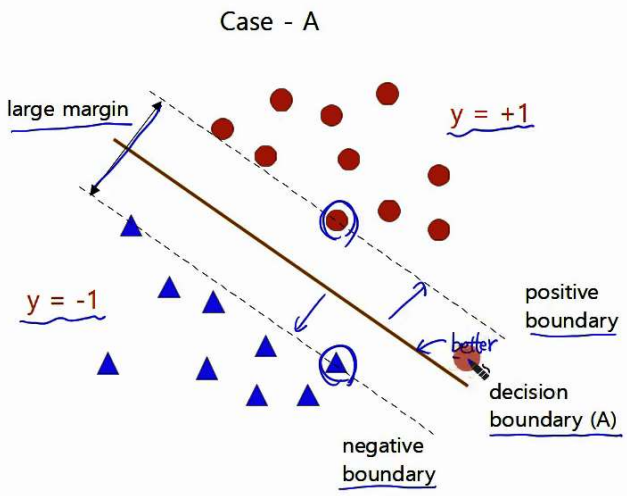
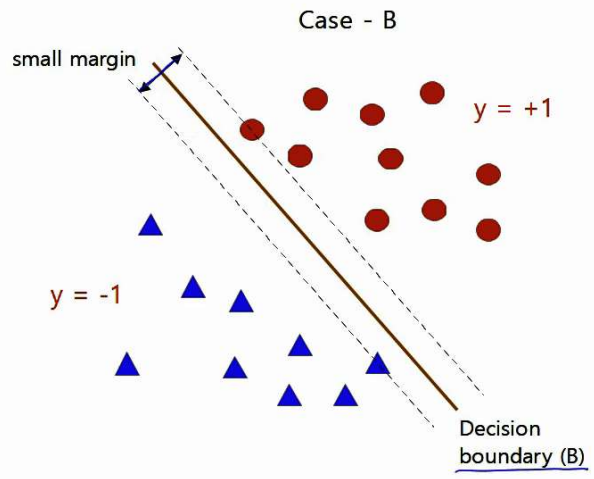
Hard Margin Revisited

Thursday 23 May 2024 5:37 AM

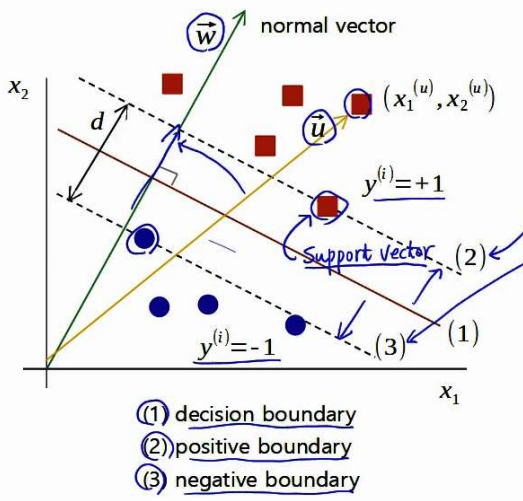
- In the figure below, there are numerous decision boundaries that can distinguish two data clusters.
- Draw an arbitrary straight line between the two clusters and shift the line parallel to the left and right until it reaches the data point, creating two straight lines. The best decision boundary is the one with the largest distance between the two straight lines.
- A support vector machine aim to find the best decision boundary that best separates a dataset into two classes.
- The distance is called the margin. Support vector machine is an algorithm that finds the decision boundary with the largest margin.





Linear Hard Margin: Decision rule and Constraints

- SVM is an algorithm that maximizes the distance (d) between the two straight lines (2) and (3) in the figure below.



(1) decision boundary
(2)positive boundary
(3) negative boundary

(1) $w_1 x_1 + w_2 x_2 + b = 0 \quad \vec{w} = (w_1, w_2)$

Decision rule

$\vec{w} \cdot \vec{u} \geq c, \; u_{class} = y^{(u)} = +1$ ← If the magnitude of u vector going in the direction of w vector is greater than or equal to a certain number c, that data point is classified as +1.

$\vec{w} \cdot \vec{u} + b \geq 0$ ← Determine $\vec{w}$ and b using the given data.

constraints

$\vec{w} \cdot \vec{x}^{(+)} + b \geq k$ ← Positive data points are above the positive boundary (2).

$\vec{w} \cdot \vec{x}^{(-)} + b \leq -k$ ← Negative data points are below the negative boundary (3).

$\vec{w}' \cdot \vec{x}^{(+)} + b' \geq 1$ ← Divide both sides by k to get standardized w' and b'. Let's just write w' and b' as w and b.

$\vec{w}' \cdot \vec{x}^{(-)} + b' \leq -1$

$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1$ ← $y^{(i)} = \begin{cases} 1 & for \; "+" \\ -1 & for \; "-" \end{cases}$

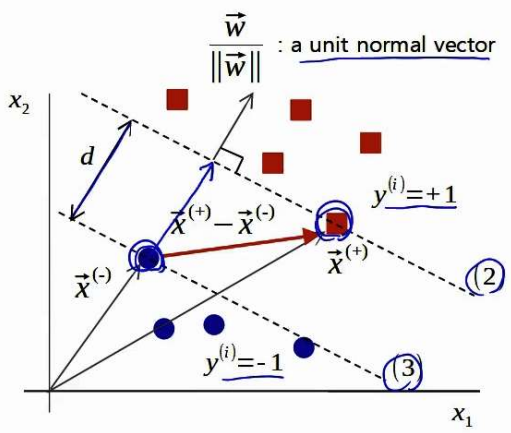
$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) - 1 \geq 0$

$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) - 1 = 0$ ← The data points x on the positive or negative boundary are called support vectors.

* reference : (MIT lecture) https://www.youtube.com/watch?v=_PwhiWxHK8o

Objective function

- Our goal is to maximize the distance between the positive and negative boundaries.



(2) positive boundary
(3) negative boundary

Objective function

$d = (\vec{x}^{(+)} - \vec{x}^{(-)}) \left(\frac{\vec{w}}{\|\vec{w}\|} \right)$ ← The distance or margin between the positive and negative boundaries. This is the magnitude of the red vector going in the direction of the unit w vector.

$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) - 1 = 0$ ← Both x+ and x- samples meet this condition because they are support vectors.

$(\vec{w} \cdot \vec{x}^{(+)} + b) - 1 = 0$ ← for x+ samples. y= +1

$\vec{w} \cdot \vec{x}^{(+)} = 1 - b$

$-(\vec{w} \cdot \vec{x}^{(-)} + b) - 1 = 0$ ← for x- samples. y= -1

$\vec{w} \cdot \vec{x}^{(-)} = -1 - b$

$d = \frac{(\vec{x}^{(+)} \cdot \vec{w} - \vec{x}^{(-)} \cdot \vec{w})}{\|\vec{w}\|} = \frac{(1-b-(-1-b))}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$ ← Distance d

$max(d) = max \frac{2}{\|\vec{w}\|} = min \|\vec{w}\| \rightarrow \boxed{min \frac{1}{2} \|\vec{w}\|^2}$ ← Final objective function

Optimization: Lagrange primal function

- Using the given data samples (x), find w and b that minimize the objective function with an inequality constraint.

objective	constraint	
$\min \frac{1}{2}\|\vec{w}\|^2$	$y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1\geq 0$	$y^{(i)}=\begin{cases}1 & for\ "+"\\ -1 & for\ "-"\end{cases}$

- inequality constrained optimization problem

$$\min_x f(x), \quad \text{subject to} \quad h(x)\leq 0$$

- Lagrange primal function

$$L_p(x,\lambda)=f(x)+\lambda h(x) \ \rightarrow \ \lambda\geq 0$$

$$\min_x L_p(x,\lambda), \quad s.t \quad h(x)\leq 0$$

$$\frac{\partial L_p}{\partial x}=0$$

Lagrange primal function for SVM

$$L_p=\frac{1}{2}\|\vec{w}\|^2+\sum_{i=1}^{N}\lambda_i\{1-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)\}$$

$$\frac{\partial L_p}{\partial \vec{w}}=0 \ \rightarrow \ \vec{w}=\sum_{i=1}^{N}\lambda_i y^{(i)}\vec{x}^{(i)}$$

$$\frac{\partial L_p}{\partial b}=0 \ \rightarrow \ \sum_{i=1}^{N}\lambda_i y^{(i)}=0$$

$$\lambda_i\geq 0$$

Optimization: Lagrange dual function

$$L_p=\frac{1}{2}\|\vec{w}\|^2+\sum_{i=1}^{N}\lambda_i\{1-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)\}$$ ▪ Lagrange primal function

$$\frac{\partial L_p}{\partial \vec{w}}=0 \ \rightarrow \ \vec{w}=\sum_{i=1}^{N}\lambda_i y^{(i)}\vec{x}^{(i)} \qquad \frac{\partial L_p}{\partial b}=0 \ \rightarrow \ \sum_{i=1}^{N}\lambda_i y^{(i)}=0$$

$$L_D=\frac{1}{2}\Big(\sum_{i=1}^{N}\lambda_i y^{(i)}\vec{x}^{(i)}\Big)\Big(\sum_{j=1}^{N}\lambda_j y^{(j)}\vec{x}^{(j)}\Big)+\sum_{i=1}^{N}\Big[\lambda_i-\lambda_i y^{(i)}\Big(\sum_{i=1}^{N}\lambda_i y^{(i)}\vec{x}^{(i)}\Big)\cdot\vec{x}^{(i)}+\lambda_i y^{(i)}b\Big]$$

$$L_D=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}+\sum_{i=1}^{N}\lambda_i-\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}+b\sum_{i=1}^{N}\lambda_i y^{(i)} \quad (=0)$$

$$L_D=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}+\sum_{i=1}^{N}\lambda_i$$ ▪ Lagrange dual function

SVM dual problem

$$\underset{\lambda_i}{argmax}\Big(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}+\sum_{i=1}^{N}\lambda_i\Big)$$

or

$$\underset{\lambda_i}{argmin}\Big(\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}-\sum_{i=1}^{N}\lambda_i\Big)$$

subject to: $\lambda_i\geq 0$, $\sum_{i=1}^{N}\lambda_i y^{(i)}=0$

This is an optimization problem with inequality and equality constraints. Convex Optimization, we can find the lambdas by solving the QP problem with CVXOPT..

Optimization: KKT condition and Strong duality

- In general, the primal solution is greater than or equal to the dual solution (p* ≥ d*). If the optimization problem satisfies the KKT conditions, strong duality holds (p* = d*). For more details, please refer to the convex optimization video, [MXML-5-04].
- The optimization problem for SVM satisfies the KKT conditions as follows. So the solution for dual problem becomes the primal solution.

KKT condition

$$\min_x f(x) \quad \leftarrow \text{Convex function}$$

$$\text{subject to} \quad g_i(x)\leq 0, \quad (i=1,2,\ldots,m)$$

$$h_j(x)=0, \quad (j=1,2,\ldots,n)$$

$$L(x,\lambda,\mu)=f(x)+\sum_{i=1}^{m}\lambda_i g_i(x)+\sum_{j=1}^{n}\mu_i h_i(x), \quad (\lambda_i\geq 0)$$

1) Stationality	$\nabla_x L(x^*,\lambda^*,\mu^*)=0$	for all i, j
2) Complementary slackness	$\lambda_i g_i(x^*)=0$	
3) Primal feasibility	$g_i(x^*)\leq 0, \quad h_j(x^*)=0$	
4) Dual feasibility	$\lambda_i\geq 0$	

- Lagrange primal function

$$L_p=\frac{1}{2}\|\vec{w}\|^2+\sum_{i=1}^{N}\lambda_i\{1-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)\}$$

- KKT conditions

1) $\frac{\partial L_p}{\partial \vec{w}}=0, \quad \frac{\partial L_p}{\partial b}=0$

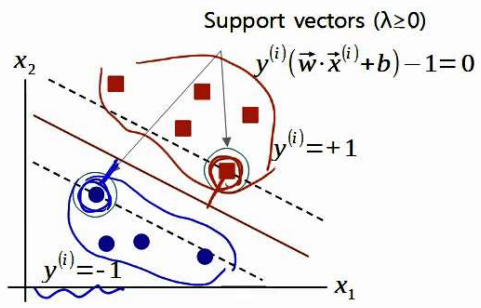
2) $\lambda_i\{1-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)\}=0$ ← If x is outside the positive or negative boundary, λ=0, so it holds true. And if x is on the boundary, λ>0, but the {.} part is 0, so it holds also true.

3) $1-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)\leq 0$ ← constraints

4) $\lambda\geq 0$

Decision function

- In the training stage, a decision function is created through the procedure below. And in the prediction stage, this decision function is used to estimate the class of the test data.



1. Solve the QP and find the optimal λ

$$L_D=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}+\sum_{i=1}^{N}\lambda_i$$

2. Use the optimal λ to find w*.

$$\frac{\partial L_p}{\partial \vec{w}}=0 \rightarrow \vec{w}^*=\sum_{i=1}^{N}\lambda_i^* y^{(i)}\vec{x}^{(i)}$$

3. Use w* to find b*.

3-1. Method-1: Bishop, Pattern Recognition and Machine Learning, p.330, equation (7.18)

$y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1=0$ ← Multiply both sides by y(i). $y^{(i)2}=1$.

$\vec{w}\cdot\vec{x}^{(i)}+b=y^{(i)}$

$b^*=y^{(i)}-\vec{w}\cdot\vec{x}^{(i)}$ ← Calculate each b for support vectors and then average them.

3-2. Method-2: Andrew Ng's CS229 Lecture notes-3 eq. (11)

- Calculate b with two support vectors

$$b^*=-\frac{\left(\max_{i:y^{(i)}=-1}\vec{w}^*\cdot\vec{x}^{(i)}\right)+\left(\min_{i:y^{(i)}=+1}\vec{w}^*\cdot\vec{x}^{(i)}\right)}{2}$$

$y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1=0$

$\vec{w}\cdot\vec{x}^{(i)+}+b=1$ ← "+" sample

$\vec{w}\cdot\vec{x}^{(i)-}+b=-1$ ← "-" sample

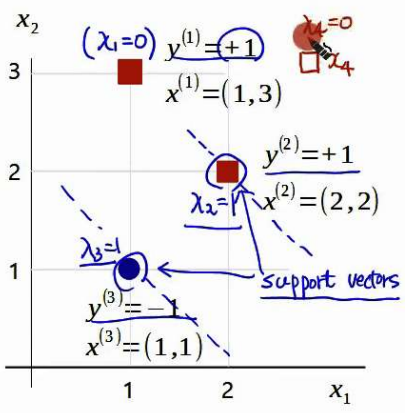
By adding the two equations above, you can find b using the equation on the left.

4. Decision function : $\hat{y}=w_1^*x_1+w_2^*x_2+b^*$

5. Put the test data into the decision function. If y-hat is positive, it is classified as +1, and if it is negative, it is classified as -1.

Example of SVM: Solving by hand

- Given three observed data points, we use SVM to find the decision boundary, or decision function, as shown below.
- Step-1 : Find the λ from Lagrange dual function.



$$L_D=\sum_{i=1}^{N=3}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}$$

$$L_D=\lambda_1+\lambda_2+\lambda_3-\frac{1}{2}\times$$
$$(\lambda_1\lambda_1 y^{(1)}y^{(1)}\vec{x}^{(1)}\cdot\vec{x}^{(1)}+\lambda_1\lambda_2 y^{(1)}y^{(2)}\vec{x}^{(1)}\cdot\vec{x}^{(2)}+\lambda_1\lambda_3 y^{(1)}y^{(3)}\vec{x}^{(1)}\cdot\vec{x}^{(3)}+$$
$$\lambda_2\lambda_1 y^{(2)}y^{(1)}\vec{x}^{(2)}\cdot\vec{x}^{(1)}+\lambda_2\lambda_2 y^{(2)}y^{(2)}\vec{x}^{(2)}\cdot\vec{x}^{(2)}+\lambda_2\lambda_3 y^{(2)}y^{(3)}\vec{x}^{(2)}\cdot\vec{x}^{(3)}+$$
$$\lambda_3\lambda_1 y^{(3)}y^{(1)}\vec{x}^{(3)}\cdot\vec{x}^{(1)}+\lambda_3\lambda_2 y^{(3)}y^{(2)}\vec{x}^{(3)}\cdot\vec{x}^{(2)}+\lambda_3\lambda_3 y^{(3)}y^{(3)}\vec{x}^{(3)}\cdot\vec{x}^{(3)})$$

$(1,3)\begin{pmatrix}1\\3\end{pmatrix}=10$, $\lambda_1^2\cdot 1\cdot 10$

$$L_D=\lambda_1+\lambda_2+\lambda_3-\frac{1}{2}(10\lambda_1^2+8\lambda_1\lambda_2-4\lambda_1\lambda_3+$$
$$8\lambda_1\lambda_2+8\lambda_2^2-4\lambda_2\lambda_3-$$
$$4\lambda_1\lambda_3-4\lambda_2\lambda_3+2\lambda_3^2)$$

$$L_D=\lambda_1+\lambda_2+\lambda_3-\frac{1}{2}(10\lambda_1^2+8\lambda_2^2+2\lambda_3^2+$$
$$16\lambda_1\lambda_2-8\lambda_1\lambda_3-8\lambda_2\lambda_3)$$

$$\frac{\partial L_p}{\partial b}=0 \rightarrow \sum_{i=1}^{N}\lambda_i y^{(i)}=1\lambda_1+1\lambda_2-1\lambda_3=0 \quad (\lambda_3=\lambda_1+\lambda_2)$$

$$L_D=2\lambda_1+2\lambda_2-\frac{1}{2}\times$$
$$(10\lambda_1^2+8\lambda_2^2+2(\lambda_1+\lambda_2)^2+$$
$$16\lambda_1\lambda_2-8\lambda_1(\lambda_1+\lambda_2)-8\lambda_2(\lambda_1+\lambda_2))$$

$$L_D=2\lambda_1+2\lambda_2-2\lambda_1^2-\lambda_2^2-2\lambda_1\lambda_2$$

$$\begin{cases}\frac{\partial L_D}{\partial \lambda_1}=2-4\lambda_1-2\lambda_2=0\\ \frac{\partial L_D}{\partial \lambda_2}=2-2\lambda_1-2\lambda_2=0\end{cases}$$

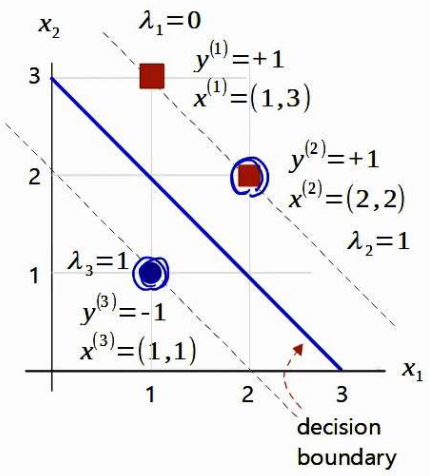
$\lambda_1=0$, $\lambda_2=1$, $\lambda_3=1$

- x with λ > 0 is the support vector.
- Support vector = $x^{(2)}$, $x^{(3)}$

Example of SVM: Solving by hand

- Step-2 : Find w by substituting λ into the equation obtained by differentiating Lagrange primal function.
- Step-3 : Find b using the support vectors (λ > 0).

- training data



- Find w using λ

$$\vec{w}=\sum_{i=1}^{N}\lambda_i y^{(i)}\vec{x}^{(i)}$$

$$\vec{w}=0\times1\times[1,3]+1\times1\times[2,2]+1\times(-1)\times[1,1]$$

$$\vec{w}=[1,1]$$

- Find b using w and support vectors

$$y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1=0$$ ← The support vectors are on this straight line.

$$1\times([1,1]\cdot[2,2]+b)-1=0 \rightarrow b=-3$$

$$(-1)\times([1,1]\cdot[1,1]+b)-1=0 \rightarrow b=-3$$

$b^*=-3$

$w=(1,1)$

- Decision boundary : $x_1 + x_2 - 3 = 0$
- Margin : $d=\frac{2}{\|\vec{w}\|}=\frac{2}{\sqrt{1^2+1^2}}=\sqrt{2}$
- positive boundary: $y(\vec{w}\cdot\vec{x}+b)-1=0$, $(y^{(2)}=+1)$

$$1([1,1]\cdot[x_1,x_2]-3)-1 = x_1+x_2-4=0$$

- negative boundary: $y(\vec{w}\cdot\vec{x}+b)-1=0$, $(y^{(3)}=-1)$

$$(-1)([1,1]\cdot[x_1,x_2]-3)-1 = x_1+x_2-2=0$$

- Classify the test data

test data ($\hat{y}=x_1 + x_2-3$)

x_1	x_2		predicted class	Remark
3	2	2	+1	2 > 0, classify it as +1
1	0.5	-1.5	-1	-1.5 < 0, classify it as -1
2	0	-1	-1	on the negative boundary
3	1	1	+1	on the positive boundary
3	0	0	?	on the decision boundary

Quadratic Programming (QP) : using CVXOPT

- Convert the Lagrange dual function to the standard form of Quadratic Programming.
- Reference: https://xavierbourretsicotte.github.io/SVM_implementation.html

$$L_D=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}$$

$$H_{i,j}=y^{(i)}y^{(j)}\vec{x}^{(i)}\cdot\vec{x}^{(j)}$$ ← definition

$$H=\begin{bmatrix} y^{(1)}y^{(1)} & y^{(1)}y^{(2)} \\ y^{(2)}y^{(1)} & y^{(2)}y^{(2)} \end{bmatrix}\times\begin{bmatrix} \vec{x}^{(1)}\cdot\vec{x}^{(1)} & \vec{x}^{(1)}\cdot\vec{x}^{(2)} \\ \vec{x}^{(2)}\cdot\vec{x}^{(1)} & \vec{x}^{(2)}\cdot\vec{x}^{(2)} \end{bmatrix}$$ ← For N=2, element wise product

H = np.outer(y, y) * np.dot(x, x.T) ← Python code

$$L_D=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j H_{i,j}$$

$$L_D=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}\cdot H\cdot\begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix}$$

$$L_D=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\lambda^T\cdot H\cdot\lambda$$

$$\text{s.t}\quad \lambda_i\geq 0$$

$$\sum_{i=1}^{N}\lambda_i y^{(i)}=0$$

Quadratic Programming (QP) : using CVXOPT

- Standard form of QP

$$\underset{x}{argmin}\ \frac{1}{2}x^T P x+q^T x$$

$$\text{s.t}\quad Gx\leq h,\quad Ax=b$$

- Lagrange dual function for SVM

$$\underset{\lambda}{argmin}\ \frac{1}{2}\lambda^T H\lambda-1^T\lambda_i$$ ← $\underset{\lambda}{argmax}\ L_D \rightarrow \underset{\lambda}{argmin}\ (-L_D)$

$$\text{s.t}\quad -\lambda_i\leq 0,\quad \sum_{i=1}^{N}\lambda_i y^{(i)}=0$$ ← Expressed in the same format as the standard form of QP.

$P := H$ size = N x N

$q := -\vec{1}$ = [[-1], [-1], …] - size = N x 1

$G := -I$ size = N x N

$h := \vec{0}$ size = N x 1

$A := y$ size = N x 1

$b := 0$ scalar

$(G\cdot\lambda\leq h)$ $(-\lambda_i\leq 0)$

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}\cdot\begin{bmatrix}\lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4\end{bmatrix}=\begin{bmatrix}-\lambda_1 \\ -\lambda_2 \\ -\lambda_3 \\ -\lambda_4\end{bmatrix}\leq\begin{bmatrix}0 \\ 0 \\ 0 \\ 0\end{bmatrix}$$

G h

```
# Calculate H matrix
H = np.outer(y, y) * np.dot(x, x.T)

# Construct the matrices required for QP
# in standard form.
n = x.shape[0]
P = cvxopt_matrix(H)
q = cvxopt_matrix(-np.ones((n, 1)))
G = cvxopt_matrix(-np.eye(n))
h = cvxopt_matrix(np.zeros(n))
A = cvxopt_matrix(y.reshape(1, -1))
b = cvxopt_matrix(np.zeros(1))
```

→ Hard Margin SVM only possible if a straight line could completely separate the two data clusters without any misclassification.