S.V.M. – Mathematics

Objective : Optimal hyperplane which linearly seperates the data points in 2 component by maximizing the margin.

Hyperplane : Linearly divides n-dimensional data points in 2 components

$$W = \begin{bmatrix} W_0 \\ W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix} \quad X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

~~~~

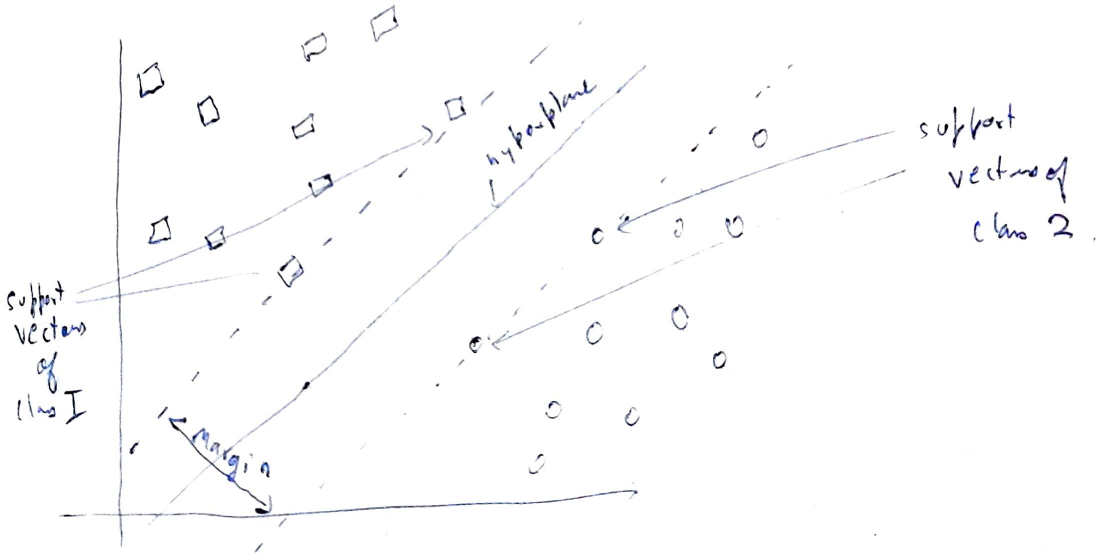
Let $X = (x, y)$ & $W = (a, -1)$ then hyperplane is

$W \cdot X + b = 0$ $\quad \boxed{W^T X = 0}$ ← n-dimensional hyperplane.

If data is not linearly separable. We add extra dimensions like $Z = x^2 + y^2$ to make data separable.

Kernel Trick.

• Many Hyperplanes. Need to find Optimal Hyperplane



Margin : If solid line is optimal hyperplane & 2 dotted line are some hyperplane passing through nearest data points of optimal hyperplane. Then distance between hyperplane & optimal hyperplane is margin.

Support Vectors : Closest data-points to optimal hyperplane which help formulation of other 2 hyperplane
~~~~

When optimal hyperplane is selected we choose hyperplane which has highest distance from closest data points.

Goal is to maximize margin while selecting optimal hyperplane.

- Let there be l training egs
- Each eg X is D dimensional & has label $y=+1$ or $y=-1$; (2 class).

& it is linearly separable data.

Training data : $\{x_i, y_i\}$ where $i = 1, \dots l$

$y_i \in \{-1, 1\}$ &

$x \in \mathbb{R}^D$

Goal of SVM is to orient the hyperplane as far as possible from closest members of both classes..

$w.x_i + b \geq 1 \quad \forall i : y_i = +1$

$w.x_i + b \leq 1 \quad \forall i : y_i = -1$

$y_i(w.x_i + b) \geq 1 \quad \forall i$

$H1 : w.x + b = -1$

$H2 : w.x + b = 1$

$\Big\}$ Two hyperplane passing through support vectors of class 1 & -1

Distance of optimal hyperplane from origin is

$$\frac{-b}{|w|}$$

Hence distance of H1 is $\frac{-1-b}{|w|}$

& H2 is $\frac{1-b}{|w|}$

~~Hence~~ margin

$$M = \frac{1-b}{|w|} - \frac{(-1-b)}{|w|} = \frac{2}{|w|}$$

Margin is $\frac{M}{2}$ $\because$ as M is distance between H1 & H2.

Hence margin $= \frac{1}{|w|}$

To find optimal hyperplane objective function $= \max \frac{1}{||w||}$

or $\min ||w||$

or $\min \frac{||w||^2}{2}$ s.t. ~~$y_i(x_i$~~

$y_i(w.x_i+b)-1 \geq 0$

$\forall i=1,\dots,l$

$\therefore$ Objective function : $\min \frac{||w||^2}{2}$.

Constraint ~~cond~~ : $y_i(wx_i+b)-1 \geq 0 \quad \forall i$

For Constrained Optimization we use Lagrange Multiplier

Do Not use gradient descent as for constrained optimization problem dual optimization algorithm preferred.

Problem min $\frac{||w||^2}{2}$ s.t $y_i(w.x_i+b)-1 \geq 0$

$\forall i \in \{0, \dots, l\}$

Sol^n:

Using Lagrange Multipliers

$$l = \frac{||w||^2}{2} - \sum_{i=1}^{l} \lambda_i (y_i (w.x_i+b)-1)$$

$$= \frac{||w||^2}{2} - \sum_{i=1}^{l} \lambda_i (y_i (w x_i+b)) - \sum_{i=1}^{l} \lambda_i$$

$$\frac{\partial l}{\partial w} = w - \sum_{i=1}^{l} \lambda_i (y_i x_i) = 0 \quad - (1)$$

$$\frac{\partial l}{\partial \lambda_i} = y_i (w.x_i+b) - 1 = 0$$

$$\frac{\partial l}{\partial b} = \sum_{i=1}^{l} \lambda_i . y_i = 0 \quad - (2)$$

Primal Optimization Sol^n. We cannot get sol^n. So we.

Dual optimization formulation

Converting Problem to Dual Problem

$$l_d \quad \frac{\|w\|^2}{2} - \sum_{i=1}^{l} \lambda_i (y_i (w x_i + b)) + \sum_{i=1}^{l} \lambda_i$$

Using ①

$$l_d = \frac{\sum_{i=1}^{l} \|\lambda_i \cdot y_i \cdot x_i\|^2}{2} - \sum_{j=1}^{l} \sum_{i=1}^{l} \lambda_j y_j (\|\lambda_i y_i x_i\| \cdot x_j + b) + \sum_{i=1}^{l} \lambda_i$$

$$= \frac{\sum_{i=1}^{l} (\lambda_i y_i x_i)^T (\lambda_i y_i x_i)}{2} - \sum_{j=1}^{l} \sum_{i=1}^{l} \lambda_j \lambda_i y_j y_i x_i x_j + \underbrace{\sum_{j=1}^{l} \sum_{i=1}^{l} \lambda_j y_j}_{0} - \sum_{i=1}^{l} \lambda_i$$

Using ② $\sum_{j=1}^{l} \sum_{i=1}^{l} \lambda_j y_j = 0$

$$l_d = \sum_{i=1}^{l} \lambda_i - \frac{\sum_{i,j=1}^{l} \lambda_j \lambda_i y_j y_i x_i x_j}{2} \quad (??)$$

∴ Simplified eqn of dual opti problem..

$K(i,j) = y_j y_i x_i x_j \qquad K = y^T y \cdot x^T x$

$$\therefore \max l_d = \sum_{i=1}^{l} \lambda_i - \frac{\lambda^T \lambda K}{2}$$

So Problem is to find λ to max^n l_d

After optimization using SMO (sequential minimization optimization)

we get value of λ.

Using that $w = \sum_{i=1}^{l} \lambda_i y_i x_i$

Now with w & λ we calculate b.

· Any support vector x_s will have

$$y_s(w \cdot x_s + b) - 1 = 0$$

substituting w.

$$y_s\left(\sum_{m \in S} \lambda_m y_m x_m \cdot x_s + b\right) - 1 = 0$$

S denotes indices of support vectors

(as dot product)

$$y_s y_s\left(\sum_{m \in S} \lambda_m y_m x_m \cdot x_s + b\right) = y_s.$$

$y_s y_s = 1$ as $(\pm 1)^2 = 1$

$$\sum_{m \in S} \lambda_m y_m x_m \cdot x_s + b = y_s$$

$$b = y_s - \sum_{m \in S} \lambda_m y_m x_m \cdot x_s$$

Use avg of all support vectors.

$$b = \sum_{s \in S} \frac{\left(y_s - \sum_{m \in S} \lambda_m y_m x_m \cdot x_s\right)}{N_s}$$

, where S is set of all support vectors

Now we have value of both w & b

Optimal hyperplane is

$$w \cdot x + b = 0.$$

Predict $(x) =$ sign $(w.x + b)$
(Signum function)

* Previous sol^n is based on fact that no datapoint is allowed inside data-points (support-vectors).

Hard Margin sol^n

Now we will try soft margin sol^n
i.e. few data points are allowed inside margin.

- Soft margin - underfitting.
- Hard margin - Overfitting

Mathematically, we relax margin by introducing a positive slack variable ξ_i

$$w.x_i + b \geq 1 - \xi_i \quad \text{for } (y_i = +1)$$

$$w.x_i + b \leq -1 + \xi_i \quad \text{for } (y_i = -1)$$

$$y_i(w.x_i + b) - 1 + \xi_i \geq 0 \quad \text{for } y_i = \pm 1$$

$$\xi_i \geq 0 \quad \text{for } i = 1, 2, l$$

$$\sum_{i=1}^{l} \xi_i = C$$

- ξ_i tells where i^{th} observation is located relative to hyperplane & margin. $\xi_i \geq 0$

- If $0 \leq \xi_i \leq 1$ - observation is between incorrect side of margin & correct side of hyperplane. (Margin violation)

- If $\xi_i > 1$, observation is on incorrect side of both hyperplane & margin.

- Points on wrong side of margin have penalty which increases with distance.

- C is parameter that controls trade-off between length of margin & number of misclassification.

- C = 0 ⇒ hard margin classification
- C > 0, means no more than C observation can violate margin. C↑ margin↑

$$L_0 = \min \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i \quad s.t.$$

$$y_i(w \cdot x_i + b) - 1 + \xi_i \geq 0; \; \xi_i \geq 0$$

- Primal Gradient based optimization method.

- Make optimization problem into unconstrained form without Lagrange Multipliers.
- Then solve using ~~dg~~ gradient descent etc.

$$l = \min \frac{\|w\|^2}{2} + C\sum_{i=1}^{l} \xi_i \quad \text{s.t.}$$

$$y_i(w.x_i + b) - 1 + \xi_i \geq 0, \xi_i \geq 0$$

let $f(x_i) = w.x_i + b)$.

$$\therefore \; y_i f(x_i) \geq 1 - \xi_i$$

$$\Rightarrow \xi_i \geq 1 - y_i f(x_i).$$

also $\xi_i \geq 0$.

$$\Rightarrow \xi_i = \max(1 - y_i f(x_i), 0).$$

Hence $$l = \min\Big(\underbrace{\|w\|^2}_{\text{regularization term}} + \underbrace{C\sum_{i=1}^{l} \max(1 - y_i f(x_i), 0)}_{\text{loss function term}}\Big)$$

$$\therefore \; \min_{(w,b)} c(w,b) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))$$

where $\lambda = \frac{2}{LC}$ & $f(x) = w.x + b$.

- Can use ~~dg~~ gradient descent on loss function

Dual Quadratic programming approach

(same as Hard margin soln).

$$\max L_d = \sum_{i=1}^{l} \lambda_i - \frac{\lambda^T \lambda K}{2}$$

$$0 \le \lambda \le C, \quad \sum_{i=1}^{l} \lambda_i y_i = 0$$

Dual Optimization	Gradient Descent
• Not scalable for large dataset.	• Converges easily.
• Kernel trick can be applied.	• Online learning

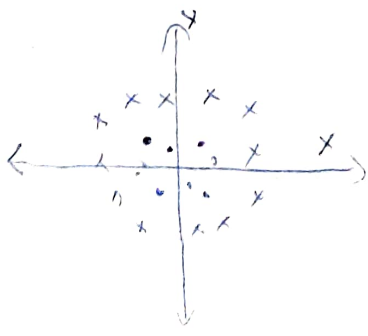
For Non Linear Data Points

Mercer's Theorem: If function $K(a,b)$ satisfies all constraints called mercer's constraints, then there exists a function that maps a & b into higher dimension.

$$K(a,b) = \phi(a)^T \cdot \phi(b)$$

$\phi()$ is a kernel

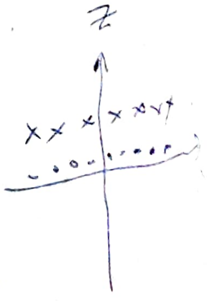
Kernels. 'linear', 'polynomial', 'radial basis function'



2D data

$$\xrightarrow{z = x^2 + y^2}$$

(∵ Class X is more distant z will always be higher for Class X). ...



Dual optimization problem of broad margin (only) SVM SVM.

$$\max L_d = \sum_{i=1}^{l} \lambda_i - \frac{\lambda^T \lambda K}{2} \quad \text{subject to}$$

$$0 \le \lambda \le C, \ \sum_{i=1}^{l} \lambda_i \cdot y_i = 0$$

$$K(i,j) = y_i y_j x_i x_j$$

$$\& \because y_i y_j = 1 \text{ so}$$

$$K(i,j) = x_i x_j$$

Linear Kernel.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

(ϕ is kernel)

Note: Instead of this we can transform input data also straight but it will increase computational cost & space by a lot. Kernel trick helps optimize code.

Gaussian Radial Basis function : Best when no prior knowledge of data.

$$K(x_i, x_j) = e^{-(\gamma \|x_i - x_j\|^2)}$$

Gaussian function :

$$K(x_i, x_j) = e^{-\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}$$

Polynomial Kernel function

$$K(x_i, x_j) = (x_i \cdot x_j + a)^b$$

Linear kernel

$k(x_i, x_j) = x_i \cdot x_j$

Time of S.V.M. learning : linear < poly < r.b.f.

ability to fit any data linear < poly < r.b.f.

risk of overfitting : linear < poly < r.b.f.

" " underfitting : rbf < poly < linear.

no. of hyperparameters : linear(0) < r.b.f(2) < poly(3)

Case I (a). support vectors x_p & x_q.

$x_p(x_{p_1}, x_{p_2}) \in$ Class 1 ; $x_q(x_{q_1}, x_{q_2}) \in$ Cl