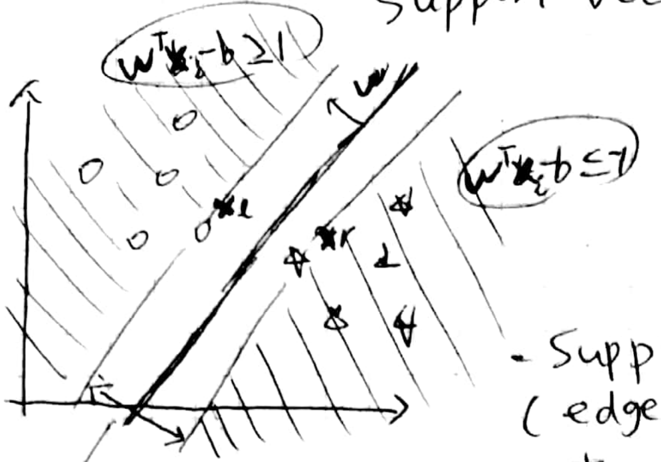
Support Vector Machine (SVM)



- Finding <u>hyperplane</u> which separates two classes of data and maximizes the distance to the closest point from either class.

- Support vectors : Samples on the margin (edge of the boundary line) ; most difficult to classify.

Consider a <u>normal vector</u> w to the dividing hyperplane

LHS data satisfy $w^T x_i - b \geq 1$
RHS $w^T x_i - b \leq -1$

$$\left.\begin{aligned} &w^T x_i - b \geq 1 \\ &w^T x_i - b \leq -1 \end{aligned}\right\} \boxed{\begin{aligned} &y_i(w^T x_i - b) \geq 1, \\ &y_i = \begin{cases} 1, & LHS \\ -1, & RHS \end{cases} \end{aligned}} \text{ constraint}$$

<u>Difference</u> between two hyperplanes $w^T x - b = 1$, $w^T x - b = -1$ is,

$$(x_l - x_r) \cdot \frac{w}{\|w\|} = \frac{(x_l w - x_r w)}{\|w\|} = \frac{(1+b) - (-1+b)}{\|w\|} = \boxed{\frac{2}{\|w\|}} \text{ maximize}$$

$\rightarrow$ $\boxed{\text{minimize } \|w\|, \text{ where } y_i(w^T x_i - b) \geq 1 \text{ for } i = 1, \cdots N \text{ (\# of data samples)}}$

Lagrange method $\rightarrow$ $\boxed{\text{minimize } \sum_i c_i - \frac{1}{2} \sum_i \sum_j y_i c_i (x_i^T x_j) y_j c_j, \text{ where } \sum_i c_i y_i = 0, \; 0 \leq c_i \leq \frac{1}{2n\lambda}}$

- Kernel trick : Defines inner product in the <u>transformed</u> space ; can learn <u>non-linear</u> classification by applying linear classification on <u>transformed</u> datapoints.

$\rightarrow$ $\boxed{\sum_i c_i - \frac{1}{2} \sum_i \sum_j y_i c_i k(x_i, x_j) y_j c_j}$ $\boxed{k(x_i, x_j) = \phi(x_i)\phi(x_j)}$

Kernel matrix ($[k_{ij}]$ where $k_{ij} = k(x_i, x_j)$) is <u>positive semidefinite</u>

b.c. $\forall x$, $x^T K x = \sum_{i,j} x_i x_j k_{ij} = \sum_{i,j} x_i x_j \phi(x_i)\phi(x_j)$

$$= \sum_i x_i \phi(x_i) \sum_j x_j \phi(x_j) = \|x \phi(x)\|_2 \geq 0$$

and kernel function is <u>symmetric</u>, $k(x_i, x_j) = k(x_j, x_i)$.

- Mercer's theorem : for any symmetric function k that is positive semidefinite, $\exists$ transform ϕ_i, $\lambda_i \geq 0$ such that $k(\mathbf{x}, \mathbf{x}') = \sum_i \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}')$.

By the mercer's theorem, we can see that the kernel function must be symmetric, p.s.d.

Probabilistic LDA (PLDA)

- In Gaussian mixture model
  $P(x|y) = N(x|y, \Phi_w)$ where $x \to$ example, $y \to$ latent var. (mean of mixture)
  instead of having discrete prior prob. of y,
  model $\boxed{P(y) = N(y|m, \Phi_b)}$ (Gaussian prior).

when $\Phi_w \to$ p.d and $\Phi_b \to$ p.s.d then $\exists V$ s.t.
$V^T \Phi_b V = \Psi$ and $V^T \Phi_w V = I$. $\to$ $\Phi_w = AA^T$, $\Phi_b = A\Psi A^T$, $A = V^{-T}$

Then $\boxed{P(y) \neq N(y|m, \Phi_b)} = N(y|m, A\Psi A^T)$
$= m + A * N(v|0, \Psi)$ ①

$\boxed{P(x|y) \neq N(x|y, \Phi_w)} = N(x|m + Av, AA^T)$
$= m + A * N(u|v, I)$ ②

$\boxed{\begin{aligned} y &= m + Av \quad ① \\ x &= m + Au \quad ② \end{aligned}}$ $\quad v \sim N(\cdot|0, \Psi)$, $u \sim N(\cdot|v, I)$

Params of PLDA : m (mean), covariance Ψ, loading matrix A (Φ_b, Φ_w)

- while LDA is efficient on classifying known data, by making prior of the class centers continuous, unknown data can be dealt more nicely.