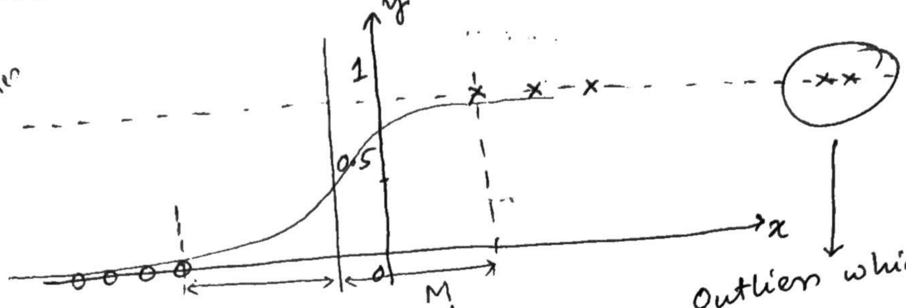
SUPPORT VECTOR MACHINES - I

faster than Loopt
extendible to nonlinear boundaries



Outliers which significantly distort Logistic Regression.

SVM: Give up probabilistic estimates for a better decision boundary

Only worry about points closest to decision boundary.

Classification Rule : $G(x_i) = \text{sgn}\left[\theta^T x_i\right]$

M : Margin

$h(x_i, \theta) = \theta_0 + \theta^T x_i > 0 \Rightarrow \hat{y}_i = +1$

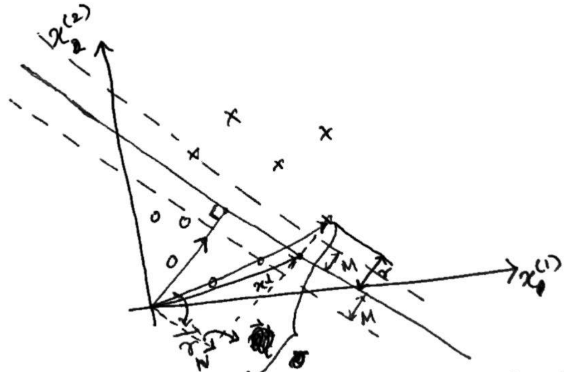
$y \in \{-1, +1\}$

$h(x_i, \theta) = \theta_0 + \theta^T x_i < 0 \Rightarrow \hat{y}_i = -1$

find θ such that M is maximised for the ~~closest~~ points closest to Decision Boundary hyperplane.

$\vec{r}_N = (\theta^{(1)}, \theta^{(2)})$

(Normal to the decision boundary)



$$\vec{x}_i = \vec{x}_i^{\perp} + y_i r_i \frac{\vec{\theta}}{\|\vec{\theta}\|}$$

$$\theta^{(0)} + \theta^{(1)} x^{(1)} + \theta^{(2)} x^{(2)} = 0$$

$$\Rightarrow x^{(2)} = -\frac{\theta^{(1)}}{\theta^{(2)}} x^{(1)} - \frac{\theta^{(0)}}{\theta^{(2)}}$$

$$\begin{aligned} h(x_i, \theta) &= \theta_0 + \theta^T x_i = \theta_0 + \vec{\theta} \cdot \vec{x}_i \\ &= \theta_0 + \vec{\theta} \cdot \left[\vec{x}_i^{\perp} + y_i r_i \frac{\vec{\theta}}{\|\theta\|}\right] \\ &= \left(\theta_0 + \vec{\theta} \cdot \vec{x}_i^{\perp}\right) + y_i r_i \frac{\|\theta\|^2}{\|\theta\|} = y_i r_i \|\theta\| \end{aligned}$$

$$\Rightarrow r_i = \frac{h(x_i, \theta)}{y_i \|\theta\|} = \frac{\theta_0 + \theta^T x_i}{y_i \|\theta\|}$$

Objective : Maximise r_i for the closest point [Support Vector]

while ensuring $y_i h(x_i, \theta) > 0$

OBJECTIVE

$$\max_{\theta, \theta_0} \min_{x_i} \frac{y_i(\theta_0 + \theta^T x_i)}{\|\theta\|}$$

Note: Scaling θ, θ_0 does not change this function.

So, scale them so that $y_i h_i = 1$ for point closest to decision boundary.

$$\max_{\theta, \theta_0} \frac{1}{\|\theta\|} \quad \text{s.t.} \quad y_i(\theta_0 + \theta^T x_i) \geq 1 \quad \forall i$$

QUADRATIC program with N linear constraints [Convex]

$$= \min_{\theta, \theta_0} \frac{1}{2}\|\theta\|^2 \quad \text{s.t.} \quad "$$

↳ small $\|\theta\|$ is regularisation. ⇒ good generalisation

Lagrange function

Convex ← $L_D = \frac{1}{2}\|\theta\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(\theta_0 + \theta^T x_i) - 1 \right]$

↳ Lagrange multipliers $\alpha_i \geq 0 \ \forall i$

$\alpha_i \neq 0$ only for support vectors.

$$\max_{\alpha_i} \min_{\theta, \theta_0} L_D$$

$$\frac{\partial L_D}{\partial \theta_0} = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\nabla_{\theta} L_D = \theta_j - \sum_{i=1}^{N} \alpha_i y_i x_{ij} = 0 \Rightarrow \boxed{\theta_j = \sum_{i=1}^{N} \alpha_i y_i x_{ij}}$$

Support vectors.

Substituting this in L_D, we get.

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}$$

Concave.

↳ Lower bound on objective function.

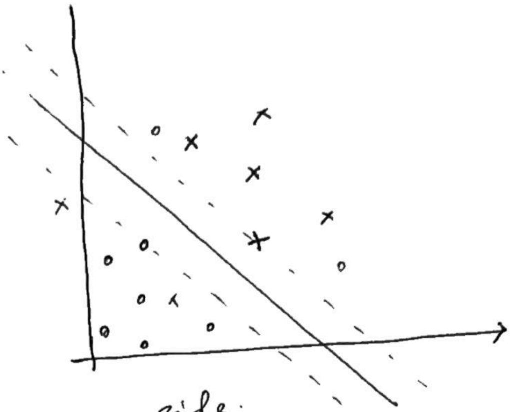
Maximising this is a simpler convex quadratic programming problem.

LAGRANGE DUAL OBJECTIVE FUNCTION (WOLFE)

min. of original L_D is $-\infty$!!

SUPPORT VECTOR MACHINES – II

What if the classes are not linearly separable?



If overlap is small, use **SLACK VARIABLES**

ξ_i, $i = 1, 2, \dots, N$

Margin by which points are allowed to be on ~~wor~~ wrong side.

$\xi_i \geq 0$, $\sum \xi_i \leq$ constant [sort of regularization]

$$\min_{\theta, \theta_0} \frac{1}{2}\|\theta\|^2 \quad \text{s.t.} \quad y_i(\theta_0 + \theta^T x_i) \geq 1 - \xi_i$$

→ trade-off between large margin & small training error

$$L_P = \frac{1}{2}\|\theta\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left[y_i(\theta_0 + \theta^T x_i) - (1 - \xi_i)\right] - \sum_i \mu_i \xi_i$$

Lagrange (primal) function [minimise wrt. θ_0, θ, ξ_i]

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_{i'} \alpha_i \alpha_{i'} y_i y_{i'} \boxed{x_i^T x_{i'}}$$

Lagrange (Wolfe) Dual Objective fn.

Optimisation depends only on dot product of input features.

* What if overlap is large?

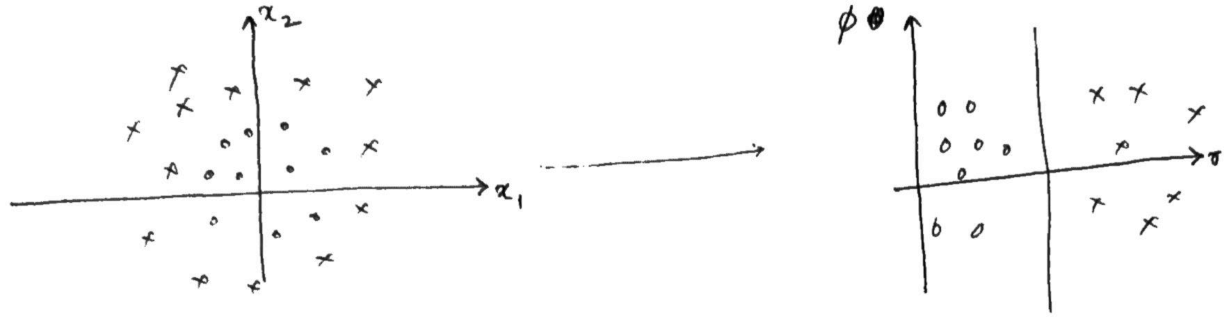
Need to transform variables, so that in the new space, the points are linearly separable.

Since dual L_D only depends on dot products, we don't need the actual transformation expression but only the new dot-product.

⇒ Kernel trick.

$$L_D = \sum_i \alpha_i - \sum_i \sum_{i'} \alpha_i \alpha_{i'} y_i y_{i'} K(x_i, x_{i'})$$

→ positive semi-definite & symmetric.



$r = \sqrt{x_1^2 + x_2^2}$

$\phi = \tan^{-1} x_2/x_1$

Popular Choices for Kernel.

d-th degree polynomial :

$$K(x_i, x_{i'}) = (1 + \langle x_i, x_{i'} \rangle)^d$$

usually best choice. Radial Basis (RBF) : or Gaussian

$$K(x_i, x_{i'}) = \exp(-\gamma \| x_i - x_{i'} \|^2)$$

γ : Kernel bandwidth

Most widely used based on Euclidean distance.

Neural Network : or Sigmoidal

$$K(x_i, x_{i'}) = \tanh(\gamma_1 \langle x_i, x_{i'} \rangle + \gamma_2)$$

In original L_D, if C is large, it penalises any ξ_i & leads to a very wiggly boundary (overfit).

$$h_{i'} = \theta_0 + \theta^T x_{i'}$$

$$\nabla_\theta L_D = \theta_j - \sum_i \alpha_i y_i x_{ij} = 0 \Rightarrow \theta_j = \sum_i \alpha_i y_i x_{ij}$$

$$\Rightarrow h_{i'} = \theta_0 + \sum_i \alpha_i y_i x_i^T x_{i'}$$

$$= \theta_0 + \sum_i \alpha_i y_i K(x_i, x_{i'})$$

RBF : Large λ leads to overfitting since even small distances are penalised.

Curse of Dimensionality

SVM-II (contd.)

$$L_D = \frac{1}{2}\|\theta\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left[y_i(\theta_0 + \theta^T x_i) - (1-\xi_i)\right] - \sum \mu_i \xi_i$$

minimize w.r.t. $\underbrace{\theta}_{\text{vector}}, \theta_0, \xi_i$

$$\frac{\partial L_D}{\partial \theta_0} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

$$\nabla_\theta L_D = 0 \Rightarrow \theta = \sum_i \alpha_i x_i y_i$$

$$\frac{\partial L_D}{\partial \xi_i} = 0 \Rightarrow \alpha_i = C - \mu_i$$

Substituting all these in L_D, we obtain,

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_{i'} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}$$

↳ Lagrange Dual objective function

<u>Maximize</u> using Quadratic Programming subject to $0 \le \alpha_i \le C$ & $\sum \alpha_i y_i = 0$

To get optimum solution, use Karush-Kuhn-Tucker conditions,

$$\alpha_i\left[y_i(\theta_0 + \theta^T x_i) - (1-\xi_i)\right] = 0 \rightarrow \alpha_i \neq 0 \text{ only for support vectors}$$

$$\mu_i \xi_i = 0 \rightarrow \mu_i \neq 0 \text{ only when } \xi_i \neq 0$$

$$y_i(\theta_0 + \theta^T x_i) - (1-\xi_i) \geq 0$$

SVM - Regularisation

$\min \frac{1}{2}\|\theta\|^2$; $\gamma \uparrow$

γ of Kernel

$C \sum \xi_i$
$\uparrow$

SVM - Polynomial & Sigmoid Kernel application. [when does RBF not work]

RBF is slower than polynomial kernel; especially for large data or high dimension

Polynomial kernel is popular in NLP with $d=2$ [$d\uparrow \Rightarrow$ overfitting & numerical instability]

~~RBF is~~ No free Lunch Theorem

SVM Multi-class

One vs. One or One vs. Rest.

Lagrange Multiplier
[Could be maxima/minima]

high γ RBF overfitting

Support vectors are not necessarily the points closest to decision boundary ~~for~~ when slack variables used.

LogReg → find prob & change threshold.

+ve $\theta_j \Rightarrow$ feature favours $\hat{y}=1$

Missing values → take mean/mode or remove from data

θ_0 : intercept
θ : co-efficients

~~Kernel~~ - ~~Does NOT scale well to large training sets or large feature vectors~~

~~RBF is slower~~

Hyperplane - Decision Boundary

Hyper parameters (user tuned) vs. Parameter (internal model)

Generative vs. Discriminative