L8 Binary (linear) classification

- Binary classification input $x_i \in \mathbb{R}^d$, output $y_i \in \{\pm 1\}$
- define a classifier f, which makes prediction $\hat{y}_i = f(x_i, \Theta)$ based on a function of x_i and parameters Θ; in other words $f: \mathbb{R}^d \to \{-1, +1\}$
- Previous Bayesian classification framework:-
  - Θ: 1) class prior distributions on y
    2) Parameters for class-dependent distribution on x
- Linear classification framework:-
  - Prediction linear in parameters Θ
  - intersection of Bayesian classification and linear classification

Bayes classifier

- Bayes classifier → predict class of a new x to be most probable label given the model and training data $(x_1, y_1), \ldots, (x_n, y_n)$

Binary case: decide class $y=1$ if

$$\underbrace{p(x|y=1)\,p(y=1)}_{\pi_1} > p(x|y=0)\,p(y=0)$$

$$\Downarrow$$

$$\ln\left[\frac{p(x|y=1)\,p(y=1)}{p(x|y=0)\,p(y=0)}\right] > 0 \qquad \text{"log odds"}$$

Linear Discriminant Analysis (LDA) - Gaussian shared covariance Bayes classifier

- consider log odds for special case where

$$p(x|y) = N(x|\mu_y, \Sigma)$$ i.e. single Gaussian with a shared covariance matrix as class conditional density

- $$\ln\left[\frac{p(x|y=1)\,p(y=1)}{p(x|y=0)\,p(y=0)}\right] = \ln\frac{\pi_1}{\pi_0} - \underbrace{\frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)}_{\text{constant } (w_0)} + \underbrace{x^T\Sigma^{-1}(\mu_1-\mu_0)}_{\text{vector } (w)}$$

- This is LDA (Linear Discriminant Analysis)

Derivation:-

$p(y=0) = \pi_0$ (not to be confused with π)

$p(y=1) = \pi_1$

$p(x|y=1) = N(x|\mu_1, \Sigma)$

$p(x|y=0) = N(x|\mu_0, \Sigma)$

$$\ln\left[\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}\right] = \ln\left[\frac{\frac{1}{\sqrt{2\pi}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}\cdot\pi_1}{\frac{1}{\sqrt{2\pi}|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right\}\cdot\pi_0}\right] =$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) + \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) + \left(\frac{1}{2}x^T\Sigma^{-1}x - \frac{1}{2}x^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_0^T\Sigma^{-1}x + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0\right) + \left(-\frac{1}{2}x^T\Sigma^{-1}x + \frac{1}{2}x^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_1^T\Sigma x - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1\right)$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}\left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\right) + \left(x^T\Sigma^{-1}\mu_1 - x^T\Sigma^{-1}\mu_0\right)$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0) + x^T\Sigma^{-1}(\mu_1-\mu_0)$$

- Quadratic term $\left(\frac{1}{2}x^T\Sigma^{-1}x\right)$ is independent of class label $\{0,1\}$; cancels out
- $x^T\Sigma^{-1}\mu_0 = \mu_0^T\Sigma^{-1}x$ and $x^T\Sigma^{-1}\mu_1 = \mu_1\Sigma^{-1}x$
- $(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0) = (\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)$ is taken to a difference of 2 squares identity ove vectors.

Linear classifie and Bayes classifie relation

- Rewrite decision rule for Bayes classifie as linear one:-

$$f(x) = \text{sign}(x^Tw + w_0)$$ with w_0 and w from previous slide

- see diagram :- Bayes classifie produces a linear decision boundary in dataspace when there is a shaed covarance matrix for both classes i.e. $\Sigma_1 = \Sigma_0 = \Sigma$
- Bayes classifie is one instance of a linear classifie with
  - use MLE to find values/estimates of π_y, μ_y and Σ

$$w_0 = \ln\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)$$

$$w = \Sigma^{-1}(\mu_1-\mu_0)$$

- This vesion of Bayes classifie assumes single Gaussian and shaed covariance
- can we dobetter by relaxing estrictions on w_0 and w?

Binary linear classifies and linear separability

Definition: Binary linear classifie

A binary linear classifie is a function of the form:-

$$f(x) = \text{sign}(x^Tw + w_0)$$ with $w \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}$

- Goal is to estimate w, w_0 from data $\Rightarrow$ assuming linear separability in x is an accurate property of the classes.

Definition:- linear separability

Two sets $A, B \in \mathbb{R}^d$ are linearly separable if it's possible to find a vector w and w_0 such that

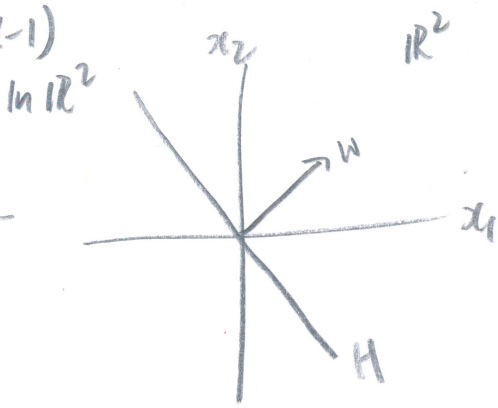
$$x^Tw + w_0 = \begin{cases} > 0 & \text{if } x \in A \text{ (always +ve) (eg. class +1)} \\ < 0 & \text{if } x \in B \text{ (always -ve) (eg class -1)} \end{cases}$$

The pair (w_0, w) defines an affine hyperplane

Hyperplanes - definition, distance, side

- Hyperplanes and affine hyperplanes give a geometric interpretation of linear classifiers
- A hyperplane in $\mathbb{R}^d$ is a linear subspace of dimension $(d-1)$
- $\mathbb{R}^2$ hyperplane - line ; $\mathbb{R}^3$ hyperplane - plane
- Linear subspaces contain the origin
- A hyperplane H can be represented by the vector w as :-

$$H = \{x \in \mathbb{R}^d \mid x^T w = 0\}$$



- All points along H have dot product with $w = 0$
- Represents set of all vectors x that are orthogonal to "parameter" vector $\underline{w}$
- Distance from the plane :-

(Q) How close is a point x to H ?

cosine rule dot product formula :-

$$x^T w = \|x\|_2 \|w\|_2 \cos\theta \Rightarrow \|x\|_2 \cdot |\cos\theta| = \frac{|x^T w|}{\|w\|_2}$$

$|x^T w|$ gives measure of distance

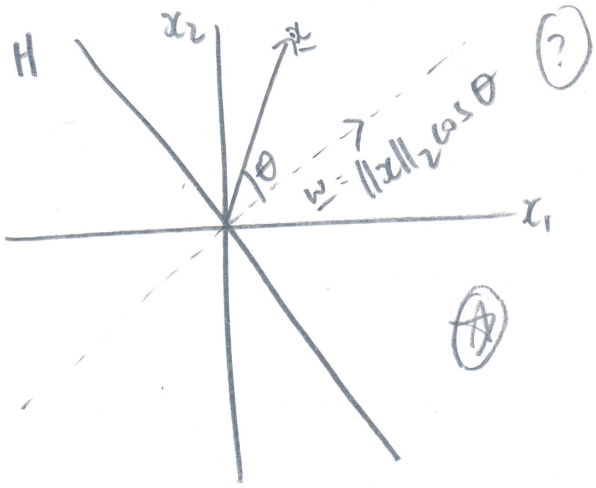
- side of hyperplane :-

(Q) which side ?

$\cos\theta > 0$ if $\theta \in \left(\frac{-\pi}{2}, \frac{\pi}{2}\right)$

sign of $\cos(\cdot)$ tells us side of H, via cosine rule and non-negativity of $\|x\|_2$ and $\|w\|_2$

$$\text{sign}(\cos\theta) = \text{sign}(x^T w)$$



Affine hyperplanes

- An affine hyperplane H is a hyperplane shifted using a scalar w_0
- $H = \{x \in \mathbb{R}^d \mid x^T w + w_0 = 0\}$

(Q) Setting $w_0 > 0 \Rightarrow$ shift H in opposite direction of w

$w_0 < 0 \Rightarrow$ —"— direction of w

- Intuition for x_0:

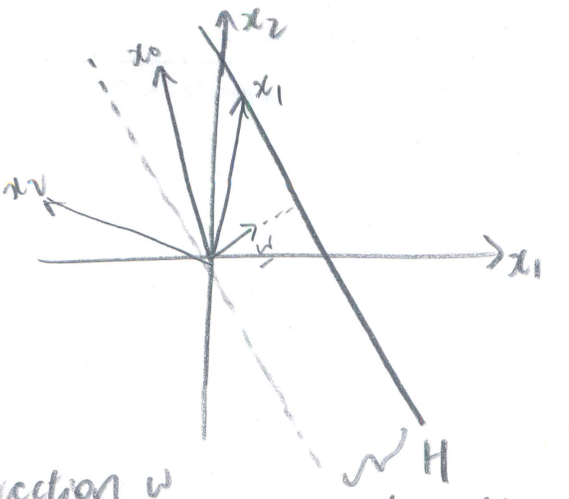
x_0; $x_0^T w > 0$, $w_0 < 0$; for points on this region

$|x_0^T w| < |w_0| \Rightarrow x_0^T w + w_0 < 0$ (?)

x_2; $x_2^T w < 0$, $w_0 < 0$; for points on this region

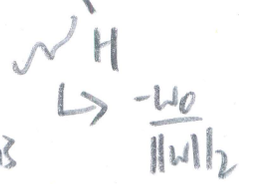
$x_2^T w + w_0 < 0$

- x_1; $x_1^T w = -w_0$, $w_0 < 0 \Rightarrow x_1^T w + w_0 = 0$



Side of hyperplane

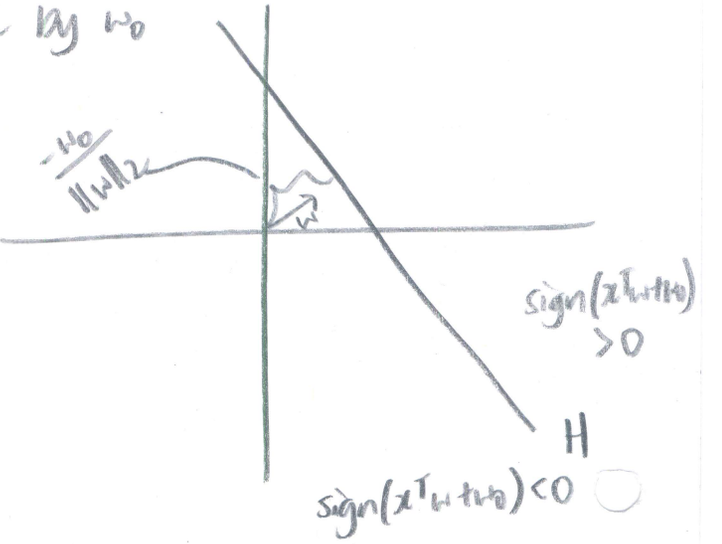
- Place has been shifted by distance $\frac{-w_0}{\|w\|_2}$ in direction w



- For a given w, w_0 and input x, inequality $x^T w + w_0 > 0$ says x is on far side of affine hyperplane H in direction w points

Classification with affine hyperplanes

- classification $\Rightarrow$ Find a (w, w_0) such that $\text{sign}(x^T w + w_0) > 0$ on all $x \in A$ (+1)
  $\text{sign}(x^T w + w_0) < 0$ on all $x \in B$ (-1)
- Asking for an angle of the hyperplane defined by w (H is perpen. to w)
- — " — shift of the hyperplane defined by w_0



Polynomial generalisations

- See diagram
- linear classifier (x_1, x_2) → linear classifier $x = (x_1, x_2, x_1^2, x_2^2)$
- Binary linear classification in $\mathbb{R}^4$ ⇒ find a hyperplane in $\mathbb{R}^4$ to separate vectors
- This decision boundary ~~proj~~ in $\mathbb{R}^4$ projected to $\mathbb{R}^2$ is quadratic
- every point maps to $\mathbb{R}^4$ in such a way that is linearly separable in $\mathbb{R}^4$
- see this with Bayes classifier using class conditional densities specified with class specific means and covariances (multivariate Gaussian) $\Sigma_0 \neq \Sigma_1$

Quadratic Discriminant Analysis - Gaussian with different covariance Bayes classifier

- Consider log odds for special case where:-

$p(x|y) = N(x|\mu_y, \Sigma_y)$ - each class has its own covariance

$$\ln\left[\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}\right] = \underbrace{\text{sth complicated not involving } x}_{\text{constant}} + \underbrace{x^T(\Sigma_1^{-1}\mu_1 - \Sigma_0^{-1}\mu_0)}_{\text{linear in } x} + \underbrace{x^T(\Sigma_0^{-1}/2 - \Sigma_1^{-1}/2)x}_{\text{quadratic in } x}$$

- Known as QDA (Quadratic Discriminant Analysis); it's linear in weights

⊛ Derivation:-

$p(y=0) = \pi_0 \quad p(y=1) = \pi_1 \quad p(x|y=1) = N(x|\mu_1, \Sigma_1) \quad p(x|y=0) = N(x|\mu_0, \Sigma_0)$

$$\ln\left[\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}\right] = \ln\left[\frac{\frac{1}{\sqrt{2\pi|\Sigma_1|}^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right)\cdot\pi_1}{\frac{1}{\sqrt{2\pi|\Sigma_0|}^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\right)\cdot\pi_0}\right]$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) + \frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) + \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) + \frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) + \left(\frac{1}{2}x^T\Sigma_0^{-1}x - \frac{1}{2}x^T\Sigma_0^{-1}\mu_0 - \frac{1}{2}\mu_0^T\Sigma_0^{-1}x + \frac{1}{2}\mu_0^T\Sigma_0^{-1}\mu_0\right)$$

$$+ \left(-\frac{1}{2}x^T\Sigma_1^{-1}x + \frac{1}{2}x^T\Sigma_1^{-1}\mu_1 + \frac{1}{2}\mu_1^T\Sigma_1^{-1}x - \frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1\right)$$

$$= \ln\left(\frac{\pi_1}{\pi_0}\right) + \frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) + \frac{1}{2}\mu_0^T(\Sigma_0^{-1} - \Sigma_1^{-1})\mu_0 + x^T(\Sigma_1^{-1}\mu_1 - \Sigma_0^{-1}\mu_0)$$

$$+ \frac{1}{2}x^T(\Sigma_0^{-1} - \Sigma_1^{-1})x$$

as required

- $x^T\Sigma^{-1}\mu_1 = \mu_1^T\Sigma^{-1}x$; $x^T\Sigma_0^{-1}\mu_0 = \mu_0^T\Sigma_0^{-1}x$

Bayes classifier, linear classification, polynomial classification

- Note (from QDA):-

$f(x) = \text{sign}(x^TAx + x^Tb + c)$ is linear in (A, b, c) (weights) $\quad \text{sign}(x^TAx + x^Tb + c) > 0$ class 1; $\text{sign}(\text{—}''\text{—}) < 0$ class 0 or -1

- Note transform $x \in \mathbb{R}^2$ $\quad x \leftarrow (x_1, x_2, 2x_1, x_2, x_1^2, x_2^2)$ and do linear classification in $\mathbb{R}^5$
- Quadratic function $(x^TAx + x^Tb + c) \Rightarrow$ linear classifier in higher dimensional problem
- Learning a linear classifier in $\mathbb{R}^5$ maps to a nonlinear function in original $\mathbb{R}^2$
- 'Nonlinearity' → 'linearity'
- Bayes classifier with shared covariance is analogous to linear classification
- —''— with different '' —''— polynomial classification.

General classifies as a regression problem

⊛ How to define more general classifies:-

$$f(x) = \text{sign}(x^Tw + w_0)$$

- Treat classification as regression problem
  1. Define $y = (y_1, \cdots, y_n)^T$ where y is a vector populated by class labels y_i

2. Add dimension of 1s to each x_i and construct $\underline{X} = [x_1, \dots, x_n]^T$

3. Estimate LS vector by $\hat{\underline{w}} = (X^TX)^{-1}X^Ty$

4. New point x_0 declare $y_0 = \text{sign}(x_0^T\hat{w})$ ← w_0 is included in w

- Can also use l_p regularisation
- Above represents baseline options; use with K-NN for evaluative performance measures
- However, issue with this approach is sensitivity to outliers

Sensitivity to outliers

- see diagram
- Due to outliers, LS will perform poorly, and significant chance of overfitting
- Desire a more robust method, less sensitive to covariate values, more focus on hyperplane side (classification)

Perceptron algorithm (Rosenblatt 1958)

- Recall definition of linear separability from before
- Assuming x_i has 1s concatenated into it
- Data in $\mathbb{R}^d$ is linearly separable if it's possible to find (w, w_0), defining a classifier $y_i = \text{sign}(x_i^Tw)$ for all i with zero training error
- There may be infinite no. of classifiers, ~~linear~~ meeting this criteria; linear separability only requires for one to exist.

=

- using linear classifier $y = f(x) = \text{sign}(x^Tw)$ - with 1s and w_0 concatenated into x_i and w respectively
  $\Rightarrow w \in \mathbb{R}^{d+1}$, $x_i \in \mathbb{R}^{d+1}$
- Perceptron minimises:

$$\mathcal{L} = -\sum_{i=1}^{n} (y_i \cdot x_i^Tw)\, \mathbb{1}\{y_i \neq \text{sign}(x_i^Tw)\}$$

i) ii)

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

$(y_i \cdot x_i^Tw)$ is $\begin{cases} > 0 & \text{if } y_i = \text{sign}(x_i^Tw) \\ < 0 & \text{if } y_i \neq \text{sign}(x_i^Tw) \end{cases}$

i) Summing over all points predicted incorrectly according to w

ii) Is sign of y_i and x_i^Tw the same?

- Rewriting $\mathcal{L}$ explicitly (for clarity):-

$$\mathcal{L} = -\sum_{i=1}^{n} (y_i \cdot x_i^Tw)\, \mathbb{1}\{y_i \neq \text{sign}(x_i^Tw)\} = (y_1 \cdot x_1^Tw)\, \mathbb{1}\{y_1 \neq \text{sign}(x_1^Tw)\}$$
$$+ (y_2 \cdot x_2^Tw)\, \mathbb{1}\{y_2 \neq \text{sign}(x_2^Tw)\}$$
$$+ (y_3 \cdot x_3^Tw)\, \mathbb{1}\{y_3 \neq \text{sign}(x_3^Tw)\}$$
$$+ \dots + (y_n \cdot x_n^Tw)\, \mathbb{1}\{y_n \neq \text{sign}(x_n^Tw)\}$$

Estimating perceptron parameters via stochastic gradient descent

- Finding minimum of L via analytic methods not feasible ⇒ numerical method/ iterative algorithm.
- $\nabla_w L = 0$ cannot be expressed / solved for analytically
- But $\nabla_w L$ does give information about about direction L is increasing in w

∴ For sufficiently small update η if we update :-

$w' \leftarrow w - \eta \nabla_w L$ then $L(w') < L(w)$ i.e. better value for w

i.e. update w by moving in a small increment in the opposite (negative) direction of w.

- This is a very general method called gradient descent ; it belongs in the field of convex / numerical optimisation.
- Perceptron uses stochastic version of this

Perceptron pseudocode

~~Input~~

Input : training data $(x_1, y_1), \dots, (x_n, y_n)$ and positive step size $\eta > 0$

1. Initialise / set $w^{(1)} = \vec{0}$
2. For step $t = 1, 2, \dots$ do
   a) Search for all examples $(x_i, y_i) \in \mathcal{D}$ such that $y_i \neq \text{sign}(x_i^T w)$
   b) If such a (x_i, y_i) exists, randomly select one and update

$$w^{(t+1)} = w^{(t)} + \eta y_i x_i$$

   Else: Return $w^{(t)}$ as solution since everything is correctly classified

- Comments: 2a) : search for examples of misclassification given current vector $w(t)$

2b) : Q Why $+\eta y_i x_i$?

If M_t indexes misclassified observations at step t, we have :-

$$L = -\sum_{i=1}^{n} (y_i \cdot x_i^T w) \mathbb{1}\{y_i \neq \text{sign}(x_i^T w)\}, \quad \nabla_w L = -\sum_{i \in M_t} y_i x_i$$

- Note that $\nabla_w L = -[\dots]$ (minus sign) ; update is taking a step in direction of label multiplied by vector x_i, scaled by factor η. This is the opposite direction of moving in the direction of w. (Be careful with signs A)

Iterative illustration

see diagram

perceptron drawbacks

- 2 drawbacks:-
  1) linear separability ⇒ infinite # hyperplanes; does not account for qualitative considerations, only converges to 1st one (locally optimal, but not globally optimal)
     - need definition of optimality
  2) No linear separability ⇒ algorithm does not converge
     - hard to detect this ie. whether algorithm will converge after a very long time or run forever
     - theoretically impossible to know (Halting problem à la Turing)
- late algorithms that also directly estimate (hyperplane defined by) w; with alteration to objective fn to fix problems.

L9

Motivating logistic regression in context of (binary) linear classification

- linear classifiers :-

given data $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$

- linear classifier takes vector $w \in \mathbb{R}^d$ and scalar $w_0 \in \mathbb{R}$ and predicts

$$y_i = f(x_i; w, w_0) = \text{sign}(x_i^T w + w_0)$$

- 2 methods :- LS - sensitivity to outliers Perceptron :- convergence/linear separability issues
- Combine separating hyperplane idea with probability.

Bayes... LDA example

- linear classification rule using a Bayes classifier →
- For model $y \sim \text{Bern}(\pi)$ $x|y \sim N(\mu_y, \Sigma)$ declare $y=1$ given x if

$$\ln \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} > 0$$

log odds → $$\ln\left(\frac{p(x|y=1)\cdot p(y=1)}{p(x|y=0)\cdot p(y=0)}\right) = \underbrace{\ln\frac{\pi_1}{\pi_0} - \frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)}_{w_0} + \underbrace{x^T\Sigma^{-1}(\mu_1-\mu_0)}_{w}$$

Bayes classifier, discriminative log-odds

- original formulation involved declaring $y=1$ given x based on a ratio of posterior probabilities formulated as log odds i.e. if

$$\ln \frac{p(y=1|x)}{p(y=0|x)} > 0$$

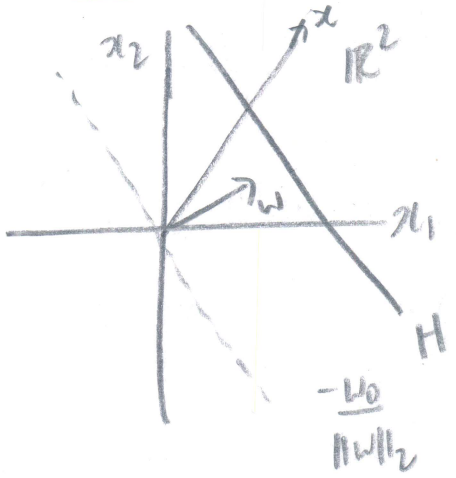
- Earlier formulations e.g. LDA, QDA have recourse to Bernoulli priors $p(y)$ and multivariate Gaussian class conditional densities $p(x|y)$ (with/without shared covariances) and use Bayes rule to specify generative classifiers
- logistic regression specifies $p(y|x)$ directly and is a discriminative classifier.

Discriminative log-odds - hyperplane, link, sigmoid

- classifying x based on ~~log~~ (discriminative) log-odds

$$L = \ln\left[\frac{p(y=+1|x)}{p(y=-1|x)}\right]$$

- $L \gg 0 \Rightarrow$ more confident $y=+1$
- $L \ll 0 \Rightarrow$ —"— $y=-1$
- $L = 0 \Rightarrow$ either way



- The linear (link function) $x^Tw + w_0$ captures :-

i) distance x to a hyperplane $H(w_0, w)$ is $\left|\frac{x^Tw}{\|w\|_2} + \frac{w_0}{\|w\|_2}\right|$

ii) $\text{sign}(x^Tw + w_0)$ captures side of $+|$ x falls on

iii) $\left|\frac{x^Tw}{\|w\|_2} + \frac{w_0}{\|w\|_2}\right|$ captures confidence in our prediction of class label

logistic link function:-

$$\ln\left[\frac{p(y=+1|x)}{p(y=-1|x)}\right] = x^T w + w_0$$

- equating log-odds with separating hyperplane representation

- Difference from Bayes classifier (with specified prior and class-conditional densities)
  - prior distribution $p(y=k)$ and class conditional density $p(x|y=k)$ ~~imposes~~ with specific functional forms imposes restrictions on w and w_0.
  - In this formulation, there are no restrictions on w and w_0.

(*) $$p(y=+1|x) = \frac{\exp\{x^T w + w_0\}}{1+\exp\{x^T w + w_0\}} = \sigma(x^T w + w_0)$$

∴ $x^T w + w_0$ is the link function for the log odds (linear)

- $\sigma(z) = \frac{e^z}{1+e^z} = \frac{1}{e^{-z}+1}$ is the sigmoid function

(*) Derivation:-

- denote $p(y=+1|x) = p$ and $p(y=-1|x) = (1-p)$, plug into above and solve for p:-

$$\ln\left(\frac{p}{1-p}\right) = x^T w + w_0$$

$$\Rightarrow \frac{p}{1-p} = e^{x^T w + w_0} \Rightarrow p + pe^{x^T w + w_0} = e^{x^T w + w_0} \Rightarrow p = \frac{e^{x^T w + w_0}}{1+e^{x^T w + w_0}}$$

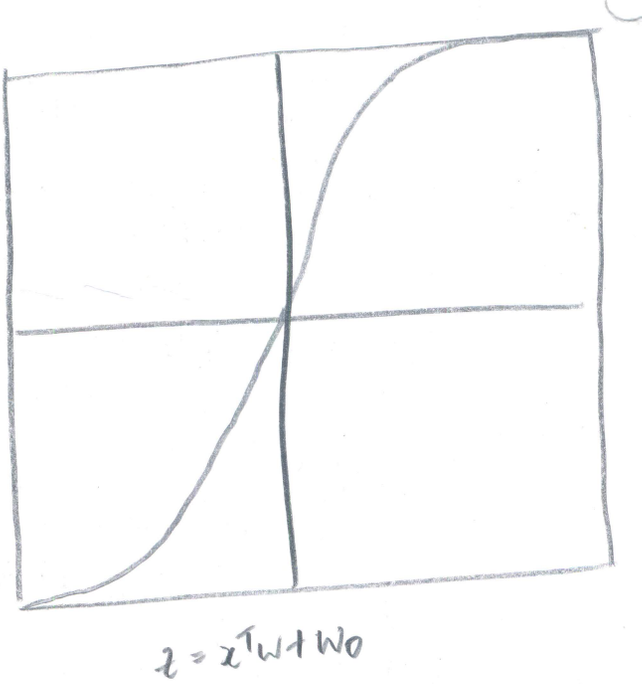
- convergence/minima and maxima on $\sigma(z)$:-

$z = x^T w + w_0$

$$\begin{cases} z=0 \Rightarrow \sigma(z) = \frac{1}{2} \\ z \to \infty \Rightarrow \sigma(z) \to 1 \\ z \to -\infty \Rightarrow \sigma(z) \to 0 \end{cases} \qquad \forall z\ \sigma(z) \gtrless 0$$

logistic sigmoid function

- $\sigma(x^T w + w_0)$ maps x to $p(y=+1|x)$
- $x \xrightarrow[\text{link}]{} x^T w + w_0 \xrightarrow[\text{sigmoid}]{} \sigma(x^T w + w_0)$
- $\sigma(\cdot)$ captures desire to be more confident as we move away from the hyperplane, defined by x-axis



Logistic regression - model, maximum likelihood estimation

- Absorb offset $w \leftarrow \begin{bmatrix} w_0 \\ w \end{bmatrix}$ $x \leftarrow \begin{bmatrix} 1 \\ x \end{bmatrix}$ $w \in \mathbb{R}^{d+1}$ $x_i \in \mathbb{R}^{d+1}$

model
- Definition:-
- Let $(x_1, y_1) \dots (x_n, y_n)$ be a set of binary labelled data $y \in \{-1, +1\}$
- Logistic regression models each y_i as independently generated with

$$p(y_i = +1 | x_i, w) = \sigma(x_i^T w) \quad \sigma(x_i; w) = \frac{e^{x_i^T w}}{1 + e^{x_i^T w}}$$

- discriminative as x not directly modelled
- Bayes classifies generative as x is modelled ($p(y)$ and $p(x|y)$)

maximum likelihood set-up and estimation

- Define $\sigma_i(w) = \sigma(x_i^T w)$, joint likelihood of $y_1, \dots, y_n$ is given by:-

$$p(y_1, \dots, y_n | x_1, \dots, x_n, w) = \prod_{i=1}^{n} p(y_i | x_i, w) \quad \text{(i)}$$

n/a independence in i.i.d

$$= \prod_{i=1}^{n} \sigma_i(w)^{\mathbb{1}(y_i = +1)} (1 - \sigma_i(w))^{\mathbb{1}(y_i = -1)} \quad \text{(ia)}$$

- Confusing notation:-

$$\frac{e^{y_i x_i^T w}}{1 + e^{y_i x_i^T w}} = \left(\frac{e^{x_i^T w}}{1 + e^{x_i^T w}} \right)^{\mathbb{1}(y_i = +1)} \left(1 - \frac{e^{x_i^T w}}{1 + e^{x_i^T w}} \right)^{\mathbb{1}(y_i = -1)} \quad \text{(ii)}$$

$\sigma_i(y_i \cdot w)$ $\quad$ $\sigma_i(w)$ $\quad$ $1 - \sigma_i(w)$ $\quad$ (iv)

- Data likelihood becomes, more compactly

$$p(y_1, \dots, y_n | x_1, \dots, x_n, w) = \prod_{i=1}^{n} \sigma_i(y_i \cdot w)$$

- maximum likelihood estimate
- maximising log-joint likelihood:-

$$w_{ML} = \arg\max_w \ln \left(\prod_{i=1}^{n} \sigma_i(y_i \cdot w) \right)$$

$$= \arg\max_w \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w)$$

$$= \arg\max_w L \quad \text{(no analytic solution via } \nabla_w L = 0 \text{ available)} \quad \text{(iii)}$$

- At step t: make update $w^{(t+1)} = w^{(t)} + \eta \nabla_w L$ with $\nabla_w L = (1 - \sigma_i(y_i \cdot w)) y_i x_i$ (evaluated at (x_i, y_i))

(a) Comments/derivations

(i): As $p(y_i|x_i,w)$ can take on values depending on outcomes of class label y_i:-

$$p(y_i=+1|x_i,w)=\sigma_i(x_i^Tw) \quad p(y_i=-1|x_i,w)=1-\sigma_i(x_i^Tw)$$

For a sequence of realised class labels $y_1,\cdots,y_n$; the expression $\prod_{i=1}^{n}\sigma_i(w)^{\mathbb{1}(y_i=+1)}(1-\sigma_i(w))^{\mathbb{1}(y_i=-1)}$ ensures that the 'right' likelihood is selected for each observed class label y_i; this distribution is class-specific

(ia): Each x_i modifies probability of success for its y_i as

$$p(y_i=+1|x_i,w)=\sigma_i(x_i^Tw)=\frac{e^{x_i^Tw}}{1+e^{x_i^Tw}}$$

(ii) $$\sigma_i(y_i=1\cdot w)=\left(\frac{e^{x_i^Tw}}{1+e^{x_i^Tw}}\right)\left(1-\frac{e^{x_i^Tw}}{1+e^{x_i^Tw}}\right)^0$$

$$\sigma_i(y_i=+1\cdot w)=\left(\frac{e^{x_i^Tw}}{1+e^{x_i^Tw}}\right)^0\left(1-\frac{e^{x_i^Tw}}{1+e^{x_i^Tw}}\right)=\frac{1}{1+e^{x_i^Tw}}$$

$$=\frac{e^{-x_i^Tw}}{1+e^{-x_i^Tw}}$$

Clarity for $\sigma_i(y_i\cdot w)=\frac{e^{y_ix_i^Tw}}{1+e^{y_ix_i^Tw}}$

(iii) Derivation of $\nabla_w l=\nabla_w\left(\sum_{i=1}^{n}\ln\sigma_i(y_i\cdot w)\right)=\sum_{i=1}^{n}(1-\sigma_i(y_i\cdot w))y_ix_i$

Recall $w\in\mathbb{R}^{d+1}$

- $\nabla_w f(w)\in\mathbb{R}^{d+1}$ but is defined over a scalar function such that $f(w):\mathbb{R}^{d+1}\to\mathbb{R}$
- $\nabla_w f(w)$ is an $(d+1)$ dimensional vector with each j^{th} entry given by $\frac{\partial f(w)}{\partial w_j}$
- As ∇ distributes over addition, we ignore the summation
- Focus on $\frac{\partial\ln\sigma_i(y_i\cdot w)}{\partial w_j}$ via 2 methods to illustrate + build confidence

Method 1: simplification + Chain rule

$$\ln \sigma_i(y_i \cdot w) = \ln\left(\frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}\right)$$

$$= \ln\left(e^{y_i x_i^T w}\right) - \ln\left(1+e^{y_i x_i^T w}\right)$$

$$= y_i x_i^T w - \ln\left(1+e^{y_i x_i^T w}\right)$$

$$\frac{d}{dz}\ln(1+z) = \frac{1}{1+z}$$

$$\frac{\partial \ln \sigma_i(y_i \cdot w)}{\partial w_j} = x_j y_j - \left(\frac{1}{1+e^{y_i x_i^T w}}\right) \cdot e^{y_i x_i^T w} \cdot x_j y_j$$

$$= x_j y_j \left(1 - \frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}\right)$$

$$= x_j y_j \left(1 - \sigma_i(y_i \cdot w)\right)$$

Hence $\nabla_w L = \sum_{i=1}^{n} \left(1 - \sigma_i(y_i \cdot w)\right) x_i y_i$ (evaluated at $x_i y_i$)

Method 2: Chain rule

$$\frac{\partial \ln \sigma_i(y_i \cdot w)}{\partial w_j} = \underbrace{\frac{\partial \ln \sigma_i(y_i \cdot w)}{\partial \sigma_i(y_i \cdot w)}}_{\text{(i)}} \cdot \underbrace{\frac{\partial \sigma_i(y_i \cdot w)}{\partial (y_i \cdot w)}}_{\text{(ii)}} \cdot \underbrace{\frac{\partial (y_i \cdot w)}{\partial w_j}}_{\text{(iii)}}$$

(i) $\dfrac{\partial \ln \sigma_i(y_i \cdot w)}{\partial \sigma_i(y_i \cdot w)} = \dfrac{1}{\sigma_i(y_i \cdot w)}$

setting $f = e^z$, $g = (1+e^z)^{-1}$: $\dfrac{\partial f(z)g(z)}{\partial z} = fg' + f'g$

(ii) $\dfrac{\partial \sigma(z)}{\partial z} = \dfrac{\partial}{\partial z}\left[\dfrac{e^z}{1+e^z}\right] = \dfrac{\partial}{\partial z}\left[e^z(1+e^z)^{-1}\right] = e^z(1+e^z)^{-1} - e^z(1+e^z)^{-2}e^z$

$= e^z(1+e^z)^{-1}\left(1 - e^z(1+e^z)^{-1}\right)$

$= k(1-k)$ with $k = e^z(1+e^z)^{-1}$

(iii) $\dfrac{\partial (y_i \cdot w)}{\partial w_j} = x_j y_j$

$$\therefore \frac{\partial \ln \sigma_i(y_i \cdot w)}{\partial w_j} = \left(\frac{1+e^{y_i x_i^T w}}{e^{y_i x_i^T w}}\right) \cdot \left(\frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}\left\{1 - \frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}\right\}\right) \cdot x_j y_j$$

$$= \left(1 - \sigma_i(y_i \cdot w)\right) x_j y_j \Rightarrow \nabla_w L = \sum_{i=1}^{n} \left(1 - \sigma_i(y_i \cdot w)\right) x_i y_i \quad \left(\text{eval at } x_i y_i\right)$$

(iv): Predicting new data is the same:-

- If $x^T w > 0 \Rightarrow \sigma(x^T w) > \frac{1}{2}$ and predict $y = +1$

(+) We now get a confidence in our prediction via probability $\sigma(x^T w)$

Logistic regression pseudocode

Input: training data $(x_1, y_1) \dots (x_n, y_n)$

1. set $w^{(1)} = \vec{0}$
2. For step $t = 1, 2, \dots$ do
   - update $w^{(t+1)} = w^{(t)} + \eta \sum_{i=1}^{n} (1 - \sigma_i(y_i \cdot w)) x_i y_i$ evaluated at $(x_i y_i)$

Differences between perceptron and logistic regression pseudocode:

Perceptron:- search for misclassified (x_i, y_i), update $w^{(t+1)} = w^{(t)} + \eta x_i y_i$

$(w' \leftarrow w - \eta \nabla_w \ell)$

- stochastic gradient descent (no summation over all misclassified examples)
- could also include summation over misclassified examples for perceptron

Logistic regression:-

- inclusion of summation over all $\sum_{i=1}^{n}$ and term $(1 - \sigma_i(y_i \cdot w))$
- $\sigma_i(y_i \cdot w)$ picks probability assigned to observed y_i } unclear
- $1 - \sigma_i(y_i \cdot w)$ ——— " ——— to wrong value }
- If we are very confident in our prediction $\sigma_i(y_i \cdot w)$ is close to 1; $1 - \sigma_i(y_i \cdot w)$ is close to 0, and update becomes very small (ε)
- If we are not confident in our prediction $\sigma_i(y_i \cdot w)$ is close to 0; $1 - \sigma_i(y_i \cdot w)$ is close to 1, and update is larger.

(+) essentially magnitude of ∞ confidence in our prediction included in update term, not included in perceptron (i.e. no probabilistic interpretation)

L^2 regularised logistic regression, MAP estimation, Gaussian prior

Key problem: If hyperplane can separate all training data (i.e. perfect linear separability), $\|w_{ML}\|_2 \to \infty$ and hence $\sigma_i(y_i \cdot w) \to 1$ for each (x_i, y_i) (see diagram)

- For nearly separable data, classifier/hyperplane might get a few very wrong to be more confident about rest: overfitting

Solution: Regularize w with $\lambda w^T w$ (to prevent w from becoming too large)

- $w_{MAP} = \arg\max_w L = \arg\max_w \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w) - \frac{\lambda}{2} w^T w$
- This corresponds to a belief on parameter that is captured in a Bayesian setting as a Gaussian prior on w (multivariate Gaussian)
- Recall parallels → $w_{ML} \leftrightarrow w_{LS}$
  $w_{MAP} \leftrightarrow w_{RR}$
  use Bayesian LR to find posterior $p(w|y,X)$
- Can we calculate posterior $p(w|x,y)$? No, need Laplace approximation.

Bayesian logistic regression

- Recall, we have defined likelihood model for class labels as sigmoid → $p(y_1,\dots,y_n|x_1,\dots,x_n,w) = \prod_{i=1}^{n} \sigma_i(y_i \cdot w)$
- Recall, (in regularised logistic) ⇒ Gaussian prior on w $p(w) = N(w|0,\lambda^{-1}I)$

Posterior calculation :- (likelihood, prior)

$$p(w|y,x) = \frac{\prod_{i=1}^{n} \sigma_i(y_i \cdot w)\, p(w)}{\int_{\mathbb{R}^n} \prod_{i=1}^{n} \sigma_i(y_i \cdot w)\, p(w)\, dw}$$

- Non-standard distribution, integral in denominator intractable
- Q → other methods? → MCMC methods

∴ $p(w|x,y)$ cannot be calculated, but can be approximated

Laplace Approximation

- As $p(w|x,y)$ cannot be calculated, we pick a distribution family to approximate it.
- Select $p(w|x,y) \approx$ Normal(μ,Σ)
- Require a method for setting μ,Σ
- Note $\prod_{i=1}^{n} \sigma_i(y_i \cdot w)\, p(w) = p(w|x,y)\, p(w) = p(y,w|x)$
- $$p(w|x,y) = \frac{e^{\ln p(y,w|x)}}{\int_{\mathbb{R}^n} e^{\ln p(y,w|x)}\, dw}$$ as $e^{\ln f(x)} = f(x)$
- Approximate $\ln p(w,y|x)$ in numerator and denominator

2nd order Taylor expansion

- Define $f(w) = \ln p(y,w|x)$ i.e. log of joint-likelihood = f(data, (parameter) model variable)
  - data cannot be changed and w is only "free parameter" ⇒ suppress data and X

Taylor expansion:

- Approximate $f(w) = \ln(p(y, w|x))$ with 2nd order Taylor expansion.
- $w \in \mathbb{R}^{d+1} \Rightarrow$ for any point $z \in \mathbb{R}^{d+1}$

$$f(w) \approx f(z) + (w-z)^T \nabla f(z) + \frac{1}{2}(w-z)^T (\nabla^2 f(z))(w-z)$$

- Laplace approximation involves selecting $z = w_{MAP}$ where $\nabla f(z)$ is short for $\nabla_w f(w)|_z$ i.e. gradient of f with respect to w calculated at z. and similarly for $\nabla^2 f(z)$
- RHS and LHS functions of w
- $f(w) = \ln(p(y, w|x))$ and $z = w_{MAP}$
- $$p(w|x,y) = \frac{e^{f(w)}}{\int e^{f(w)} dw} \approx \frac{e^{f(z) + (w-z)^T \nabla f(z) + \frac{1}{2}(w-z)^T \nabla^2 f(z)(w-z)}}{\int e^{f(z) + (w-z)^T \nabla f(z) + \frac{1}{2}(w-z)^T \nabla^2 f(z)(w-z)} dw}$$

- $e^{f(z=w_{MAP})}$ and $\int e^{f(z=w_{MAP})} dw$ via i) integral distributing over addition ii) $\int e^x dx = e^x$ iii) $e^{f(z=w_{MAP})}$ does not involve w and is constant
- definition of $w_{MAP} \Rightarrow \nabla_w \ln(p(y, w|x))|_{w_{MAP}} = 0$
- we are only left with terms involving Hessian of $f(w)|_{z=w_{MAP}}$

$$p(w|x,y) \approx \frac{e^{-\frac{1}{2}(w-w_{MAP})^T (-\nabla^2 \ln p(y, w_{MAP}|x))(w-w_{MAP})}}{\int e^{-\frac{1}{2}(w-w_{MAP})^T (-\nabla^2 \ln p(y, w_{MAP}|x))(w-w_{MAP})} dw}$$

which is a multivariate Normal ~~pdf~~

$p(w|x,y) \approx \text{Normal}(\mu, \Sigma)$ with $\mu = w_{MAP}$ $\Sigma = (-\nabla^2 \ln p(y, w_{MAP}|x))^{-1}$

(†): derivation (see end)

- Take the Hessian of joint likelihood to find

$$\nabla^2 \ln(p(y, w|x)) = -\lambda I - \sum_{i=1}^{n} \sigma(y_i \cdot x_i^T w_{MAP})(1 - \sigma(y_i \cdot x_i^T w_{MAP})) x_i x_i^T$$

High-level summary (SM)

- For labelled data $(x_1, y_1) \dots (x_n, y_n)$ and model:-

likelihood $p(y_i|x_i, w) = \sigma(y_i x_i^T w)$ MVG prior $w \sim N(0, \lambda^{-1} I)$ $\sigma(y_i x_i^T w) = \frac{e^{y_i x_i^T w}}{1 + e^{y_i x_i^T w}}$

1. Find $w_{MAP} = \underset{w}{\text{argmax}} \sum_{i=1}^{n} \ln \sigma(y_i x_i^T w_{MAP}) - \frac{\lambda}{2} w^T w$ (?)
2. Set $-\Sigma^{-1} = -\lambda I - \sum_{i=1}^{n} \sigma_i(y_i x_i^T w_{MAP})(1 - \sigma_i(y_i x_i^T w_{MAP})) x_i x_i^T$
3. Approximate $p(w|x,y) = N(w_{MAP}, \Sigma)$ where Σ is inverse of Hessian.

Motivating feature expansions

Problem: A linear model on original feature space $x \in \mathbb{R}^d$ does not work

Solution: Map features to a high dimensional space $\phi(x) \in \mathbb{R}^D$ where $D > d$ and do linear modelling there

Define a feature mapping function $\phi(\cdot)\ x \mapsto \phi(x)$ $\phi: \mathbb{R}^d \to \mathbb{R}^D$

Examples: i) Polynomial regression on $\mathbb{R}$; let $\phi(x) = (x, x^2, \dots, x^p)$ $\phi: \mathbb{R} \to \mathbb{R}^p$

ii) Jump discontinuities; let $\phi(x) = (x, \mathbb{1}\{x < a\})$ $\phi: \mathbb{R} \to \mathbb{R}^2$

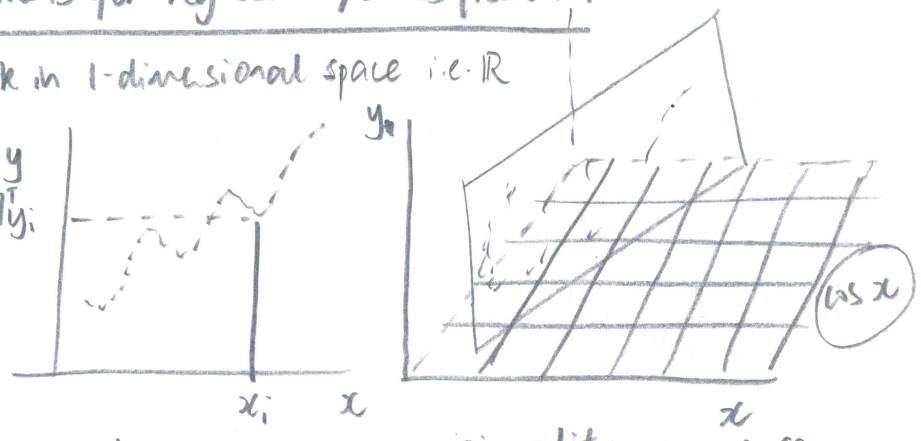
Feature expansions to higher dimensions for regression/classification

Regression example (see diagram); work in 1-dimensional space i.e. $\mathbb{R}$

$x_i \in \mathbb{R}$ $y_i \in \mathbb{R}$ $w \in \mathbb{R}$ (wo excluded)

- Map $x \to \phi(x)$ with $\phi(x) = \begin{bmatrix} x \\ \cos x \end{bmatrix} = [x, \cos]^T$ $\phi: \mathbb{R} \to \mathbb{R}^2$ (extra dimension is now 'cos x')
- $y \hat{=} w_0 + \phi(x)^T w$
  $\hat{=} w_0 + w_1 x + w_2 \cos x$ $w_\phi \in \mathbb{R}^2$
- Note that mapping $\phi: \mathbb{R}^d \to \mathbb{R}^D$ also 'induces' increase in dimensionality on w so now $w \in \mathbb{R}^D$
- Non-linearities in d-dimensional space can be transformed into higher D dimensional space, where problem may be linear.

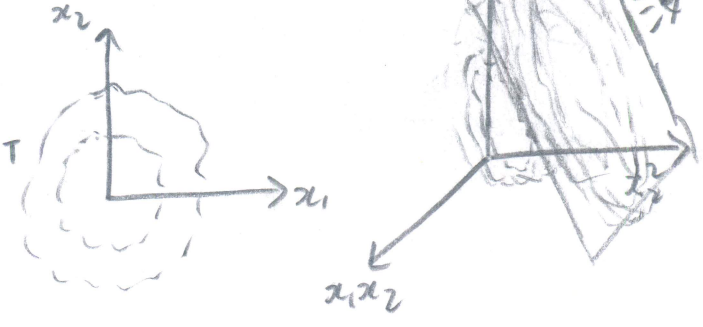
(STT) High dimensional maps can transform the data so output is linear in inputs



Classification example (see diagram)

$x_i \in \mathbb{R}^2$ $y_i \in \{-1, +1\}$ $w \in \mathbb{R}^2$ (wo excluded)

Map $x \to \phi(x)$ with $\phi(x) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} = [x_1^2, x_1, x_2, x_2^2]^T$ $\phi: \mathbb{R}^2 \to \mathbb{R}^3$



$y = \text{sign}(w_0 + \phi(x)^T w)$ ~ dot product between $\phi(x)$ and w

$= \text{sign}(w_0 + w_1 x^2 + w_2 x_1 x_2 + w_3 x^2)$ $w_\phi \in \mathbb{R}^3$

(OT) High-dimensional maps can transform data so it becomes linearly separable

Selecting appropriate feature expansion using L_1 regularisation

What expansion to use? (Not obvious)

"Kitchen sink" approach → come up with every possible feature expansion we can think of where each add to as high-dimensional as possible; where each additional dimension over and above D is just an arbitrary function of data

- Can select useful features/dimensions with L_1 penalty; learn a weight vector $w \in \mathbb{R}^D$ (not $\mathbb{R}^d$) on each of those dimensions.

$$w_{L_1} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} f(y_i, \phi(x_i), w) + \lambda \|w\|_1$$

- L_1 penalty $\Rightarrow$ sparse subset of dimensions of $\phi(x)$ to use (amounts to sparse prior on weights)

Let model decide which dimensions of expansion space $\mathbb{R}^D$ are useful.

(NB): in practice, we do not need to work with $\phi(x) \in \mathbb{R}^D$ computationally, and only dot products $\phi(x_i)^T \phi(x_j) \equiv K(x_i, x_j)$. This is known as a kernel (dot product between feature mappings of x_i, x_j respectively)

Perceptron classifier with feature expansions

Let $x_i \in \mathbb{R}^{d+1}$ $y_i \in \{-1, +1\}$ for $i = 1, \ldots, n$ observations

Perceptron constructs hyperplane from data:-

$w = \sum_{i \in M} y_i x_i$ where M is sequentially constructed set of examples

- $w^{(t+1)} = w^{(t)} + \eta x_i y_i$ misclassified $x_i y_i$ according to $w^{(t)}$

can be unfolded i.e. $w^{(t)} = w^{(t-1)} + \eta x_j y_j$

Predicting new data/labels :-

For a new observation x_0, our prediction of its label y_0 :-

$$y_0 = \operatorname{sign}(x_0^T w) = \operatorname{sign}\left(\sum_{i \in M} y_i x_0^T x_i\right)$$ replacing $w = \sum_{i \in M} y_i x_i$

Including feature expansions:-

$x_0 \leftarrow \phi(x_0)$ and $x_i \leftarrow \phi(x_i)$ we have $w = \sum_{i \in M} y_i \phi(x_i)$ and hence

$$y_0 = \operatorname{sign}\left(\phi(x_0)^T w\right) = \operatorname{sign}\left(\sum_{i \in M} y_i \underbrace{\phi(x_0)^T \phi(x_i)}\right)$$

dot product between feature expansion of x_0 and x_i is $\phi(x_0)^T \phi(x_i)$, this is the kernel function over arguments x_0 and x_i $K(x_0, x_i)$

Decision/label prediction is a dot product between data points

Kernels - definition, Gram matrix, Mercer's theorem

Kernel definition :

(has following properties?)

Kernel $K(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a symmetric function defined as follows:-

any set of n data points $x_1, \ldots, x_n \in \mathbb{R}^d$, the $n \times n$ matrix K (kernel matrix) with (j^{th}) entry $K_{ij} = K(x_i, x_j)$ is positive semi-definite

- symmetric function $\Rightarrow K(x_i, x_j) = K(x_j, x_i) \ \forall \ i, j$
- Positive semi-definite matrix: An $n \times n$ symmetric real matrix M is PSD if $z^T M z \geq 0$ for all non-zero vectors $z \in \mathbb{R}^n$
- K satisfies properties of covariance matrix Σ, and is known as a Mercer kernel,

(or) Mercer's theorem

If the kernel function $K(\cdot,\cdot)$ satisfies above properties, then there exists a feature mapping $\phi: \mathbb{R}^d \to \mathbb{R}^D$ such that

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- For a (finite) set of data points, Mercer's theorem stipulates that a kernel matrix K to be a valid Mercer kernel, it is necessary and sufficient that the corresponding kernel matrix is symmetric positive semi-definite.
- We often need to first define $K(\cdot,\cdot)$ and avoid using $\phi(\cdot)$

Radial Basis Function (RBF)/Gaussian kernel

- An example of a kernel function defined directly
- RBF:

$$K(x, x') = a \exp\left\{-\frac{1}{b}\|x - x'\|^2\right\}$$

  - general-purpose kernel
  - accounts for proximity in $\mathbb{R}^d$
  - As x and x' get close $K(x, x') \to a$
  - otherwise $K(x, x') \to 0$
  - b is known as kernel width

- In context of above, we do not prove $K(x, x')$ is PSD for all $x_i \in \mathbb{R}^d$ and therefore that Mercer's theorem holds, only assume to be true.
- The corresponding feature mapping $\phi(x)$ with RBF kernel is infinite dimensional and is a continuous function rather than a vector
- In this case: $K(x, x') = \int \phi_t(x) \phi_t(x') dt$; $\phi_t(x)$ can be thought of as a function with parameter $x \Rightarrow$ it is a function whose values lie in infinite dimensional space eg. Hilbert/Banach space.

Other kernels and kernel properties

Map: $\phi(x) = (1, \sqrt{2}x_1, \ldots \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_i x_j, \ldots)$
Kernel: $\phi(x)^T \phi(x') = K(x, x') = (1 + x^T x')^2$

} rather than go though these steps, we can define $K(\cdot,\cdot)$ directly

We can show $K(x, x') = (1 + x^T x')^b$ for $b > 0$ is also a kernel

Kernel properties and arithmetic:-

- For K_1, K_2 kernels constructing K in following ways will produce new kernel K:-

$K_1(x, x') K_2(x, x') = K(x, x')$ [product] $\quad K_1(x, x') + K_2(x, x') = K(x, x')$ [addition]

$\exp(K_1(x, x')) = K(x, x')$ [exponentiation]

- For each of these, there will exist a new mapping $\phi(x)$

Kernelised perceptron

- Feature expanded decision of perceptron:-

$$y_0 = \text{sign}\left(\sum_{i \in M} y_i \phi(x_0)^T \phi(x_i)\right)$$

$$= \text{sign}\left(\sum_{i \in M} y_i K(x_0, x_i)\right)$$

- each data point (x_i) that has been misclassified gets mapped!

- Select RBF kernel with $a=1$ and hence $K(x_0, x_i) = \exp\left\{-\frac{1}{b}\|x_0 - x_i\|^2\right\}$

$$y_0 = \text{sign}\left(\sum_{i \in M} y_i e^{-\frac{1}{b}\|x_0 - x_i\|^2}\right) \quad y_i \in \{-1, +1\}$$

Comments:-

i) No need to specify/calculate $\phi(x) \Rightarrow$ computationally cheap on memory

ii) $0 < K(x_0, x_i) \leq 1$ (large when x_0 close to x_i)

(a) iii) 'soft' voting among data: each of data points in M (misclassified examples) vote for a label for a new point x_0

Kernelised perceptron algorithm:-

- Recall: Given current vector $w^{(t)} = \sum_{i \in M_t} y_i x_i$, we update as:-

1. Find a new x' such that $y' \neq \text{sign}(x'^T w^{(t)})$ (i.e. search for misclassification with parameter vector $w^{(t)}$)

2. Add index x' to M, set $w^{(t+1)} = \sum_{i \in M_{t+1}} y_i x_i$

- using only dot products/kernel fns steps equivalent to:-

1. Find new x' such that $y' \neq \text{sign}\left(\sum_{i \in M_t} y_i K(x', x_i)\right)$

2. Add index of x' to M but don't bother calculating $w^{(t+1)} = \sum_{i \in M_{t+1}} y_i \phi(x_i)$ or $\phi(x_i)$ explicitly (I think!)

comments:

i) No need for $\phi(x)$ in step 1

ii) No need for $\phi(x)$ to classify new data

iii) Classification of new data is $y_0 = \text{sign}\left(\sum_{i \in M} y_i K(x_0, x_i)\right)$ i.e. calculation of $K(x_i, x')$ between two points

Kernel k-NN

- Generalise kernelised perceptron to soft k-NN with a simple change (OTH):- instead of summing over misclassified data M to get $\sum_{i \in M}$, we sum over all data $(x_1, \ldots, x_n)$:-

$$y_0 = \text{sign}\left(\sum_{i=1}^{n} y_i e^{-\frac{1}{b}\|x_0 - x_i\|^2}\right) \text{ rather than } \text{sign}\left(\sum_{i \in M} y_i e^{-\frac{1}{b}\|x_0 - x_i\|^2}\right)$$

- decision does not change if we divide by positive constant

let $z = \sum_{j=1}^{n} e^{-\frac{1}{b}\|x_0 - x_j\|^2}$ (sum of weights)

construct vector $p(x_0)$ with each $p_i(x_0) = \frac{1}{z} e^{-\frac{1}{b}\|x_0 - x_i\|^2}$

Declare $y_0 = \text{sign}\left(\sum_{i=1}^{n} y_i p_i(x_0)\right) = \text{sign}\left(\sum_{i=1}^{n} y_i \cdot \frac{1}{z} e^{-\frac{1}{b}\|x_0 - x_i\|^2}\right)$

sign(weighted average of labels in dataset)

Comments

i) Here k is no. of data points (n)

ii) Kernelised perceptron restricts points that vote to a subset of all data points that are misclassified, defined sequentially

iii) Let all each data point x_i vote on label of query point x_0 (k-NN)

iv) RBF kernel between query point x_0 ad data point x_i represents a weight on the label x_i in voting scheme; determined by proximity of x_0 to x_i.

v) $p(x_0)$ which is vector-valued is a probability distribution over numerical outcomes of x_0; the i^{th} dimension of which $p_i(x_0)$ is a probability for a queried x_0 to i^{th} data point in dataset; analogous to a confidence score $p(x_0)$

vi) Set b so that most $p_i(x_0) \approx 0$ to only focus on neighbourhood around x_0

↳ The weight in iii) then becomes a confidence score with more confidence in votes of data points x_i close to x_0

Nadaraya Watson model

- Regression example analogous to kernelised k-NN
- $y \in \mathbb{R}$
- Using RBF kernel, for new (x_0, y_0), predict:-

$$y_0 = \sum_{i=1}^{n} y_i \frac{K(x_0, x_i)}{\sum_{j=1}^{n} K(x_0, x_j)}$$

- locally weighted average of responses y_i, weights defined by (relative) proximity of x_i to x_0, and also interpreted rather like a probability

- Gaussian processes another option

Kernelised Bayesian Linear Regression

- For n observations with response vector $y \in \mathbb{R}^n$ and a feature matrix $X \in \mathbb{R}^{n \times d}$
- Define: likelihood $p(y|w,X)$ $\quad y \sim N(Xw, \sigma^2 I)$
  prior $p(w|X)$ $\quad w \sim N(0, \lambda^{-1} I)$

- Marginalising

$$p(y|X) = \int p(y|w,X)\, p(w)\, dw = N(0, \sigma^2 I + \lambda^{-1} XX^T)$$
(i)

(i) marginal likelihood of y given x

Kernelisation

- Note $(XX^T)_{ij} = x_i^T x_j$ (i.e. (i^{th}, j^{th}) entry of XX^T is given by dot product between i^{th} row of X and j^{th} column of X^T)
- Replace $x \leftarrow \phi(x)$ to form a new matrix $\phi(X)\phi(X)^T$ with each entry (i,j) given by $(\phi(X)\phi(X)^T)_{ij} = K(x_i, x_j)$ i.e. kernel function over data points x_i and x_j

This is done implicitly by defining K directly $\Rightarrow$

$$p(y|X) = \int p(y|X,w)\, p(w)\, dw = N(0, \sigma^2 I + \lambda^{-1} K)$$

- This is known as a Gaussian process
- The use-value of this approach (i.e. defining kernel K directly) is motivated by the fact that function ϕ may be very high/infinite dimensional
- Then parameter vector w would be infinite dimensional or possibly a function $\Rightarrow$ cannot be stored in memory; no chance of constructing $\phi(X)$ and hence matrix $\phi(X)\phi(X)^T$ as each column would be infinite dimensional
- We would then have an infinite dimensional Gaussian noise prior $w \sim N(0, \lambda^{-1} I)$
- Instead, we calculate kernel function $K(\cdot,\cdot)$ over all pairs of data points (x_i, x_j) to get an n-dimensional multi-variate Gaussian; and hence a way for calculating co-variance K

Gaussian processes

- Please see Chuong DO's notes on Gaussian processes for a more complete understanding/intuition, as lecture is brief.
- ~~#~~ a weight-space view of Gaussian processes as opposed to function-space view, using terminology from Rasmussen (2006)

Definitions:-

- Define $f(x)$ $x \to f(x)$ $f: \mathbb{R}^d \to \mathbb{R}$ $x \in \mathbb{R}^d$ (iv)
- Define kernel $K(x, x')$ between two arbitrary points x and x'
- $f(x)$ is a Gaussian process and $y(x)$ the noise added process if

$$y|f \sim N(f, \sigma^2 I), \quad f \sim N(0, K) \iff y \sim N(0, \sigma^2 I + K) \quad (v)$$

and $\epsilon \sim N(0, \sigma^2 I)$

with $y = [y_1, \ldots, y_n]^T$ and K is an $n \times n$ matrix with $K_{ij} = K(x_i, x_j)$

Comments

i) λ^{-1} has now been absorbed into K

ii) $f(x)$ is the G.P.; $y(x)$ is $f(x)$ + iid noise ϵ

iii) Kernel $K \Rightarrow$ not just "Gaussian"

iv) The definition of the G.P. f is not restricted to the data points, initially. Its domain is the entire space $\mathbb{R}^d$ and is an infinite dimensional function over that space. We then select n points in $\mathbb{R}^d$ (data) and construct the kernel $K(\cdot, \cdot)$ between all pairs of arbitrary points x and x' within that subset of $\mathbb{R}^d$ to form a kernel matrix K with $K_{ij} = K(x_i, x_j)$. Effectively this amounts to a restriction on the domain of f.

We then evaluate f at those 'end points' (?), so Gaussian process becomes also becomes an n-dimensional vector f, with ith dimension $f(x_i)$ (i.e. indexed by input data point/vector x_i with mean function) equal to vector $\underline{0}$. and covariance (function) equal to kernel matrix K.

(v) G.P. does not have noise

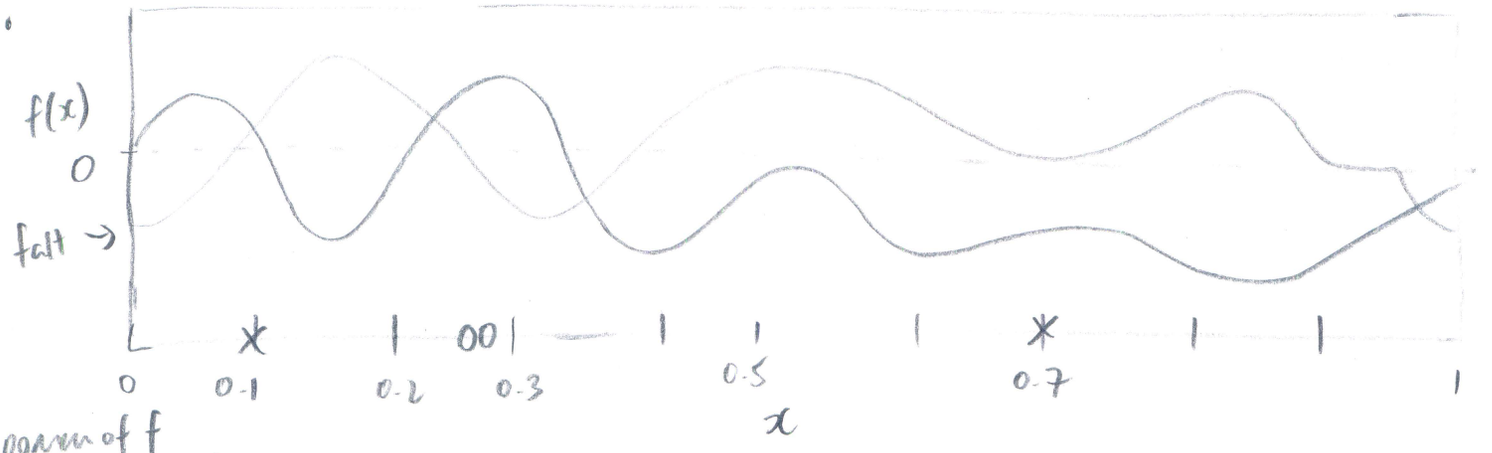
(vi) Assume there is an underlying G.P. and that the data we observe is a noisy sub-sampling of the infinite dimensional function f over $\mathbb{R}^d$ (over its codomain) at n points. In principle f is infinite dimensional over all $\mathbb{R}^d$, but we evaluate it at n data points and only observe y which is a corruption of those n Gaussian process evaluations with white noise ϵ.

- Definition a over n 'arbitrary' points with n large as we want makes this not just NNG.

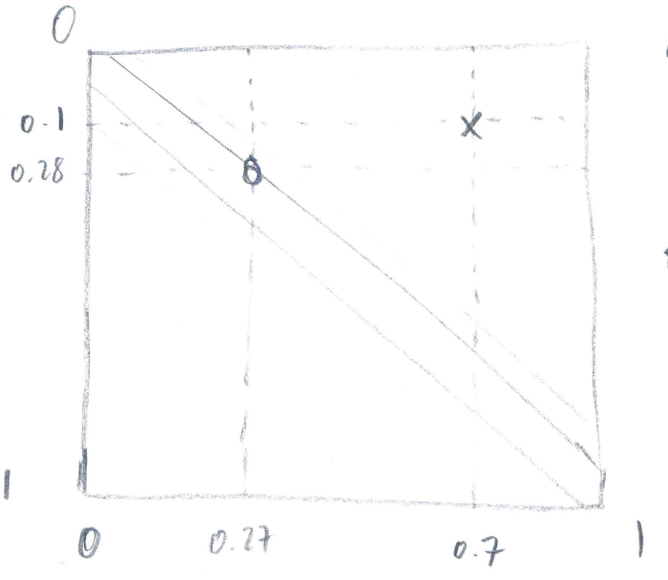
Toy Example

- Assume data $x \in [0,1]$ (rather than in $\mathbb{R}^d$) $\Rightarrow$ we are dealing with 1 dimensional scalar data points between $[0,1]$
- Pick $n = 1000$ (data points) equally spaced in $[0,1]$
- Construct kernel $K \in [0,1]^{1000 \times 1000}$ with $K(x_i, x_j) = \exp\left\{ -\frac{\|x_i - x_j\|^2}{b} \right\}$

- covariance matrix K shown in diagram in lecture as is a 1000×1000 matrix shown as an image



- domain of f
- partitioned into $n = 1000$ data points



$$K_{ij} = K(x_i, x_j) = \exp\left\{ \frac{\|x_i - x_j\|^2}{b} \right\}$$

$K \in [0,1]^{1000 \times 1000}$

- f is a 1000 dimensional vector generated from a G.P. (1000 d MVG with $m = 0$, cov $= K$)

(9)

comments

- Generation of another vector f from same distribution leads to a different ~~f(x) (----) (----)~~ f_{ALT}
- Can generate 'random' functions f and 'learn' them based on available data.
- (see diagram on next page of lecture notes)
- Model hypothesis → there is some function f underlying DGP which is not observed. All that is observed is randomly separated points ($n = 25$) in diagram and they are noisy observations / samples of function f at different points in x ⇒ this is my data set $(x_1, y_1) \dots (x_n, y_n)$

  x_i - ~~evaluation~~ input of Gaussian process at

  y_i - noisy observation of $f(x_i)$ evaluated at x_i
- useful for predicting new data
- want to infer f so we can predict evaluation of it at new point x_0

Predictions with Gaussian processes and posterior

- Replace all previous dot products with kernels

Predictive distribution of $y(x)$:

- Given measured data $D_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$, the distribution of $y(x)$ at any new x to make predictions.
- Define (replace) :-

$(x_0^T x_0)$ ← $K(x,x)$ is a scalar value $\in \mathbb{R}$

$(X x_0)^T$ ← $K(x, D_n) = [K(x,x_1), \dots, K(x,x_n)]$ i.e. row vector $\in \mathbb{R}^{1 \times n}$

(XX^T) ← K_n is an $n \times n$ kernel matrix over restricted points $D_n \in \mathbb{R}^{n \times n}$ with $(K_n)_{ij} = K(x_i, x_j)$

then

$$y(x) | D_n \sim N(\mu(x), \Sigma(x))$$

$$\mu(x) = K(x, D_n) K_n^{-1} y$$

$$\Sigma(x) = \sigma^2 + K(x,x) - K(x, D_n) K_n^{-1} K(x, D_n)^T$$

- For posterior of $f(x)$ instead of $y(x)$, remove σ^2 ; prediction is univariate Gaussian for each arbitrary point x_i

Gaussian process posterior

(See diagram)

② What does posterior of f look like?

- Data (x_i, y_i) marked by x, with $x \in \#[0,1]$ and $n = 9$ i.e. 9 points equally partitioning domain of f (which is $[0,1]$ rather than $\mathbb{R}^9$)
- Evaluate posterior of y (according to previous slide?) and evaluate at any other point x_k with $k \neq i$ i.e.

$$y_{10} | D_n = \{(x_1, y_1), \dots (x_9, y_9)\}$$

Prediction ~~$y_0 \sim$~~ y_{10} is univariate Gaussian for each arbitrary x_{10} chosen with $\mu(x_{10})$ and $\Sigma(x_{10})$

- Values pin down $f(x)$ nearby
- Get a mean and variance for every possible x, $\mu(x), \Sigma(x)$
- Distribution $y(x)$ adds small noise variance σ^2

- Understand intuitions behind diagram
- GP is an example of non-parametric regression

↳ estimating arbitrary function f from a space of possible functions drawn from a Gaussian process prior. It interpolates and smooths ~~data~~ based on data.

- We can calculate distribution on this prediction analytically

Bayesian Linear Regression - prediction with Gaussian vectors

- Return to finite dimensional space without kernel definition

Setup:

- n observation pairs $D = \{(x_i, y_i)\}_{i=1}^n$ and want to predict y_0 given x_0 (single point)
- Note that D is observed data and (x_0, y_0) is partially observed (i.e. we observe x_0 but not y_0 yet). They are independent conditioned on parameter w.
- Write joint distribution of y_0 and y and integrate out w:-

$$\begin{bmatrix} y_0 \\ y \end{bmatrix} \sim \text{Normal}\left(\underline{0}, \sigma^2 I + * \begin{bmatrix} x_0^T x_0 & (Xx_0)^T \\ Xx_0 & XX^T \end{bmatrix} \right) \quad \text{(i)}$$

- Want to predict y_0 given D and x_0; calculations show:- (ii)

$$y_0 \mid D, x_0 \sim N(\mu_0, \sigma_0^2)$$

$$\mu_0 = (Xx_0)^T (XX^T)^{-1} y$$

$$\sigma_0^2 = \sigma^2 + x_0^T x_0 - (Xx_0)^T (XX^T)^{-1} (Xx_0)$$

(Q) sketch lecture

Comments

(i) Original distribution of $y \sim N(0, \sigma^2 I + \lambda^{-1} XX^T)$ see lecture notes LS

Augmented covariance (with test point x_0)

$$\begin{bmatrix} x_0^T x_0 & (Xx_0)^T \\ Xx_0 & XX^T \end{bmatrix} = \begin{bmatrix} \boxed{x_0^T x_0} & \boxed{(Xx_0)^T} \\ \boxed{Xx_0} & \boxed{XX^T} \end{bmatrix}$$

(ii) $y_0 \mid D, x_0$ can be interpreted in many ways:
As conditional distribution of y_0 given old data D and new (single) data x_0
As marginal likelihood of y_0 or the predictive distribution of y_0

(iii) Infinite G.P. is only evaluated at finite set of points

Maximum margin classifiers

Setting is linear classification and linearly separable data

- Perceptron (linear classifier) selects a hyperplane without any 'quality' considerations, we might want to impose criteria on what 'quality' might be

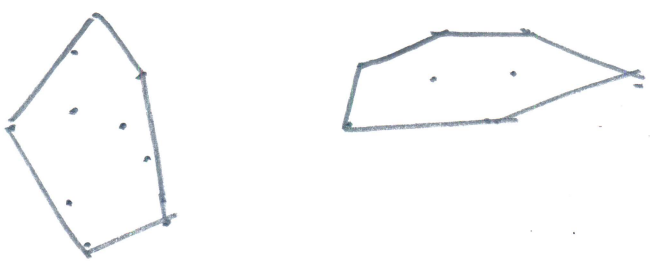
Maximum margin classification :-

- See diagram/draw
- To achieve good generalisation of our classifier / low prediction error; place hyperplane 'between' two classes, 'analogous' to making a linear regression line go through the mean, with minimised SSR distance
- More precisely, choose a plane such that distance to closest point in each class is maximised. This distance is the margin (functional or geometric?)

Generalisation error

- See diagram/draw (contrast perceptron, QDA/specific cov. B.C./max-margin)
- We want classifiers to have strong out-of-sample performance, minimise misclassifications
- Perceptron not ideal in this case
- Bayes classifier with Bernoulli priors and class-specific covariances (QDA) i.e. distributional assumptions on the data; frame as find a decision boundary that cuts into as little probability mass as possible i.e. probability of making an error is minimised
- Max-margin classifiers: Best when we do not want to make distributional ass. about the data i.e. they are non-probabilistic

Maximum-margin classification and convex hulls



- A separating hyperplane depends on ~~on~~ outerpoints

Geometrically represented - each class is the smallest convex set which contains all points in the class. - convex hull

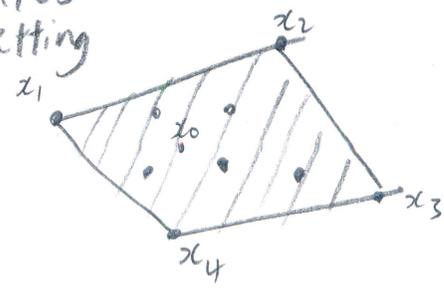
Alternatively is the smallest shrink wrapped set such that there is a line of sight between 2 points.

Convex Hulls :-

- A convex hull is defined by all possible weighted averages of points in a set

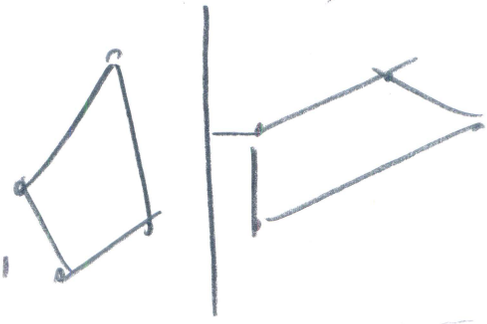
Definition

- Formally, let $x_1, \dots, x_n$ be above data co-ordinates
- Every point in convex hull can be reached by setting



$$x_0 = \sum_{i=1}^{n} \alpha_i x_i \quad \alpha_i \geq 0 \quad \sum_{i=1}^{n} \alpha_i = 1$$

for some $(\alpha_1, \dots, \alpha_n)$.
- NO points outside can be thus reached.
- Can be viewed as defining a probability distribution on data (PMF)



Margin

Definition :- The margin of a classifying hyperplane H is the shortest distance between the plane and any point in either set (equivalently, convex hull)

- Maximisation of this margin $\Rightarrow$ H is 'in the middle' of the 2 convex hulls.

(Q) How to find H?

Support Vector Machines (SVMs)

- To find the hyperplane H under maximum margin classification :-

for n linearly separable points $(x_1, y_1), \dots, (x_n, y_n)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$

(OR) Solve $\min_{w, w_0} \frac{1}{2}\|w\|^2$ s.t $y_i(x_i^T w + w_0) \geq 1$ for $i = 1, \dots, n$

Comments

i) $y_i(x_i^T w + w_0) > 0$ if $y_i = \text{sign}(x_i^T w + w_0)$ (see L8 on Perceptron)

ii) If there exists a hyperplane H separating the classes, we can scale w so that $y_i(x_i^T w + w_0) > 1$ for all $i \in 1, \dots, n$ (ie. none on boundary)

iii) resulting classifier (from optimisation problem) is a support vector machine; this particular formulation only has a solution in context of linear separability of data/classes

iv) Develop intuition for why margin is maximised

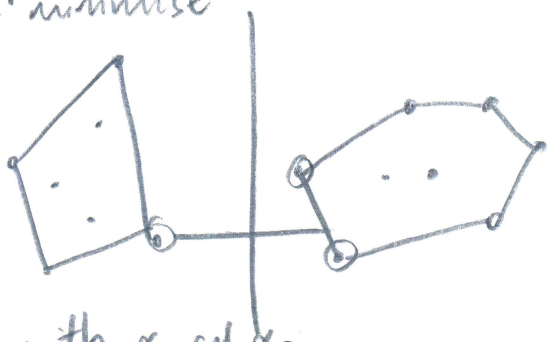
Intuition using convex hulls (or hyperplane) :-

- Finding hyperplane that maximises margin involves finding the two closest points of convex hulls of class +1 and class -1
- Connect with a line and put a hyperplane perpendicular to this.

Formally,

- If S_1 and S_0 are sets of x of class +1 and −1 respectively, we want to find two probability vectors α_1 and α_0 such that we minimise

$$\left\| \underbrace{\sum_{x_i \in S_1} \alpha_{1i} x_i}_{\text{in conv hull of } S_1} - \underbrace{\sum_{x_i \in S_0} \alpha_{0i} x_i}_{\text{in conv. hull of } S_0} \right\|_2$$



- Hyperplane is defined using the 2 points found with α_1 and α_0
- Assures constraint $y_i(x_i^T w + w_0) \geq 1$ is satisfied.

Primal problem

- The primal optimisation problem is defined as:-

(a) $$\min_{w, w_0} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(x_i^T w + w_0) \geq 1 \quad \forall\ i = 1, \dots, n$$

(b) Derivation (via Lagrange multipliers)

- More clearly, (and more abstract)

$$\min_{w, w_0} f(w) \quad \text{s.t.} \quad g_i(w, w_0) \geq 1 \quad \text{for } i = 1, \dots, n$$

- objective function :- $w \longmapsto f(w)$: $f: \mathbb{R}^d \to \mathbb{R}$
- n (inequality) constraint functions :- $(w, w_0) \longmapsto g(w, w_0)$ $g: \mathbb{R}^d \times \mathbb{R} \longrightarrow \mathbb{R}$
- define n Lagrange multiplier $\alpha_i > 0$ for $i = 1, \dots, n$
- define Lagrangian $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(x_i^T w + w_0) - 1 \right)$$

- distributing summation operator

$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n} \alpha_i$$

- Find corresponding first order conditions:-

$$\nabla_w \mathcal{L} = \nabla_w \left(\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n} \alpha_i \right)$$

Recall $\|w\|_{(2)}^2 = w^T w$

$$= \nabla_w \left(\frac{1}{2}\|w\|^2 \right) - \nabla_w \left(\sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) \right) + \nabla_w \sum_{i=1}^{n} \alpha_i$$

- w_0 is not included in w

$$= \frac{1}{2}(2w) - \sum_{i=1}^{n} \alpha_i y_i x_i = \underline{0} \Rightarrow \underline{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (A)$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{\partial}{\partial w_0}\left(\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n}\alpha_i\right) = 0$$

$$= -\sum_{i=1}^{n}\alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0 \quad (B)$$

(A)

(i) substitute $w = [\cdots]$ into original problem

(ii) $(B) \Rightarrow (\alpha_1, \cdots, \alpha_n) = \sum_{i=1}^{n}\alpha_i y_i = 0$

Recall our Lagrangian $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n}\alpha_i$$

as

$$= \frac{1}{2}w^T w - \sum_{i=1}^{n}\alpha_i y_i x_i^T w - \sum_{i=1}^{n}\alpha_i y_i w_0 + \sum_{i=1}^{n}\alpha_i$$

as $\sum_{i=1}^{n}\alpha_i y_i = 0$ from $\frac{\partial \mathcal{L}}{\partial w_0}$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right)^T \left(\sum_{j=1}^{n}\alpha_j y_j x_j\right) - \sum_{i=1}^{n}\alpha_i y_i x_i^T \left(\sum_{k=1}^{n}\alpha_k y_k x_k\right) - w_0 \underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{=0}$$

$$+ \sum_{i=1}^{n}\alpha_i$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} x_i^T \alpha_i^T y_i^T\right)\left(\sum_{j=1}^{n}\alpha_j y_j x_j\right) - \sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i \alpha_k y_i y_k x_i^T x_k + \sum_{i=1}^{n}\alpha_i$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i \alpha_k y_i y_k x_i^T x_k + \sum_{i=1}^{n}\alpha_i$$

(i) (ii)

$$\mathcal{L}_{(\alpha)} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \alpha_j y_i y_j x_i^T x_j$$

- see properties of summation operator in Kasahara
- (i) and (ii) are same, just indexed by different dummies
- For primal/dual, see Ng/Burges (or more resources)

Dual problem

$$\max_{\alpha_1, \cdots \alpha_n} \mathcal{L} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \alpha_i \geq 0 \;\forall\; i = 1, \cdots, n$$

Comments

- w_0 eliminated as multiplied by $\sum_{i=1}^{n}\alpha_i y_i = 0$
- Maximise over $\alpha_i \Rightarrow$ via an iterative algorithm (not discussed)

Solving primal problem

Q) After finding each α_i, how do we predict a new $y_0 = \text{sign}(x_0^T w + w_0)$?

We have :- $\mathcal{L}_{(P)} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \underbrace{(y_i(x_i^T w + w_0) - 1)}_{(i)}$

$$\text{st} \quad \alpha_i \geq 0 \quad y_i(x_i^T w + w_0) - 1 \geq 0$$

Solve for w :- $\nabla_w \mathcal{L} = 0 \Rightarrow \underline{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$ (substitute for learned α_is via algo.)

w_0 :-
- Assuming we have found our w (from above)
- We know that i) i.e. $y_i(x_i^T w + w_0) - 1 \geq 0$ (it forms our original constraint) (cannot be negative) cd same with $\alpha_i \geq 0$ (it cannot be negative)
- We want to maximise α_is and minimise over w_s (?) $\Rightarrow$ sign of α_i and respective $y_i(x_i^T w + w_0)$ will be opposite to one another
- At the solution, they cannot both be positive simultaneously
- Therefore in order to find w_0, we find a particular i, for which $\alpha_i > 0$ (strictly greater than 0) out of our solution $(\alpha_1, \cdots, \alpha_n)$. Consequently, the respective $(y_i(x_i^T w + w_0) - 1) = 0 \Rightarrow$ solve for w_0 using solution for w
- For all i where $\alpha_i > 0$; solution is same for w_0.

Understanding the dual problem via convex hulls

Q) Rewrite the dual problem to gain intuition via convex hulls

- Recall dual problem :-

$$\max_{\alpha_1, \cdots, \alpha_n} \quad \mathcal{L} = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$\text{st.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \alpha_i \geq 0 \;\; \forall \; i = 1, \cdots, n$$

- Since $y_i \in \{-1, +1\}$

- $\sum_{i=1}^{n} \alpha_i y_i = 0 \Rightarrow C = \sum_{i \in S_1} \alpha_i = \sum_{j \in S_0} \alpha_j$ ✓

- $\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j) = \left\| \underbrace{\sum_{i=1}^{n} \alpha_i y_i x_i}_{= w} \right\|^2 = \left\| \sum_{i=1}^{n} \alpha_i y_i x_i + \sum_{j=1}^{n} \alpha_j y_j x_j \right\|^2$ (partition into 2 summations, depending on y_i)

LHS: All $y_i = +1$
All $y_j = -1$

$$= \left\| \sum_{i \in S_1} \alpha_i x_i - \sum_{j \in S_0} \alpha_j x_j \right\|^2$$

- Multiplying and dividing by C inside the squared L^2 norm and factoring out C^2.

We have $C^2 \left\| \sum_{i \in S_1} \frac{\alpha_i}{C} x_i - \sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right\|^2$

Rewriting the dual as:-

$$\max_{\alpha_1, \dots \alpha_n} \quad \mathcal{L} = 2C - \frac{1}{2} C^2 \underbrace{\left\| \sum_{i \in S_1} \frac{\alpha_i}{C} x_i - \sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right\|^2}_{(*)}$$

i.e. $2\mathcal{L} = \sum_{i=1}^{\hat{n}} \alpha_i$

and $\sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{n}} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$

$= C^2 \left\| \sum_{i \in S_1} \frac{\alpha_i}{C} x_i - \sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right\|^2$

s.t. $C := \sum_{i \in S_1} \alpha_i = \sum_{j \in S_0} \alpha_j$ (constant), $\alpha_i \geq 0$

- maximising $\mathcal{L}_{(D)}$ is equivalent to minimising $(*)$ (via non-negativity of $(*)$) (W)

$$\min_{\alpha_1, \dots \alpha_n} \left\| \underbrace{\left(\sum_{i \in S_1} \frac{\alpha_i}{C} x_i \right)}_{\text{convex hull of } S_1} - \underbrace{\left(\sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right)}_{\text{convex hull of } S_0} \right\|^2$$

- minimising between

- explicitly here :- maximising $(\alpha_1, \dots, \alpha_n)$ is equivalent to minimising $\|w\|^2$

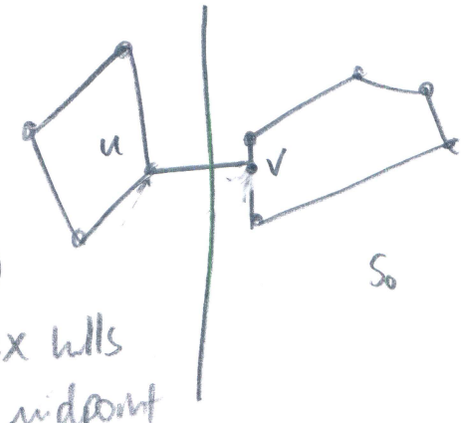
Comments

- The primal (by maximising over Lagrange multipliers $(\alpha_1, \dots, \alpha_n)$) is to minimise $\|\cdot\|^2$, finding the weights $(\alpha_1, \dots \alpha_n)$ such that if we partition the set S into S_1 and S_0, corresponding to class label $y_i \in \{+1, -1\}$, then two probability distributions are constructed using each partition of S, with (shared) normalising constant C.
- This amounts to minimising the distance between 2 points within the respective convex hulls of the 2 classes S_1 and S_0.

(Q) What does the direction of classifie defined by hyperplane look like?

- want to find :-

$$\min_{\substack{u \in H(S_1) \\ v \in H(S_0)}} \|u - v\|^2$$



- recall we find the max-margin hyperplane (claimed)
  1. Find shortest line connecting (2 points) in convex hulls
  2. Place hyperplane orthogonal to line at exactly midpoint

- w has to be perpendicular to H, but has direction $(u-v)$
- With SVM, we want to minimise $\|w\|^2$, write solution as:-

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i = c\left(\sum_{i \in S_1} \frac{\alpha_i}{c} x_i - \sum_{j \in S_0} \frac{\alpha_j}{c} x_j\right) \quad \text{(convex hull relation)}$$

Soft-margin SVM with slack variables

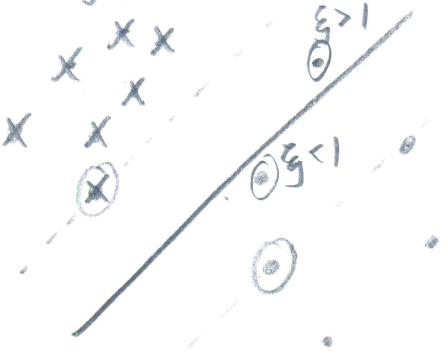
(Q) what if data is not linearly separable (given a particular dimensionality of the data...)

(A) Permit training data to be on wrong side of hyperplane, but at cost

slack variables :-

(ST) Replace training rule $y_i(x_i^T w + w_0) \geq 1$ with $y_i(x_i^T w + w_0) \geq 1 - \xi_i$ with $\xi_i \geq 0$

- ξ_i are slack variables (introduce for every data point)
- Allows for misclassification for some data points with $\xi_i > 0$



soft-margin SVM objective :-

- Inclusion of new slack variables gives following objective:-
- λ is not a lagrage multiplier

$$\min_{w, w_0, \xi_1, \ldots \xi_n} \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$

$$\text{st} \quad y_i(x_i^T w + w_0) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n$$

$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n$$

- Solve the dual problem as before

soft-margin hyperparameter tuning

- Specifies cost of incorrect classification/point on the wrong side
- Small $\lambda \Rightarrow$ happy to misclassify; $\lambda \to \infty \Rightarrow$ recover original SVM as we want $\xi_i = 0$
- choose λ using cross-validation
- (ST) λ controls how strictly we would like to enforce linear separability
- See diagram :- hyperplane sensitive to choice of λ; dotted data points have $\alpha_i > 0$ (support vectors)

Kernelised SVM

- induce a feature mapping using the function $\phi(x_i): \mathbb{R}^d \to \mathbb{R}^D$

Primal problem with slack variables:-

$$\min_{w, w_0, \xi_1, \dots, \xi_n} \quad \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$

$$\text{st} \quad y_i(\phi(x_i)^T w + w_0) \geq 1 - \xi_i \quad \forall\, i = 1, \dots, n \quad \text{(i)}$$

$$\xi_i \geq 0 \quad \forall\, i = 1, \dots, n \quad \text{(ii)}$$

Dual problem

- "Maximise over each (α_i, μ_i) where α_i and μ_i correspond to Lagrange multipliers enforcing constraint (st) (i) and (ii) respectively; and minimise over $w, w_0, \xi_1, \dots, \xi_n$

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left[y_i(\phi(x_i)^T w + w_0) - 1 + \xi_i \right] - \sum_{i=1}^{n} \mu_i \xi_i$$

$$\text{st} \quad \underset{(i)}{\alpha_i \geq 0} \quad \underset{(ii)}{\mu_i \geq 0} \quad y_i(\phi(x_i)^T w + w_0) - 1 + \xi_i \geq 0$$

Comment on (i) and (ii):-

(i): non-negativity of Lagrage multiplier

(ii) constraint including slack variables ξ_i violates original non-negativity constraints

- Minimising for w, w_0, ξ_i; we have our original FOCs:-

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{but also} \quad \frac{\partial \mathcal{L}}{\partial \xi_i} = \lambda - \alpha_i - \mu_i = 0$$

- Making substitution for w and $\mu_i = \lambda_i - \alpha_i$ into $\mathcal{L}$; we have the <u>dual</u>

$$\max_{\alpha_1, \dots \alpha_n} \quad \mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{\phi(x_i)^T \phi(x_j)}_{K(x_i, x_j)}$$

$$\text{st} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad 0 \leq \alpha_i \leq \lambda \ (*) \qquad (*)\ \text{As } \mu_i \geq 0 \Rightarrow \lambda - \alpha_i \geq 0$$

- Exactly same as hard-margin SVM, except with extra condition $0 \leq \alpha_i \leq \lambda$ (*) and replacement of $x_i^T x_j$ with $\phi(x_i)^T \phi(x_j) = K(x_i, x_j)$.
- Above is <u>kernelised SVM with slack</u>

classification

- using solution $w = \sum_{i=1}^{\hat{}} \alpha_i y_i \phi(x_i)$ declare :-

$$y_0 = \text{sign}\left(\sum_{i=1}^{\hat{}} \alpha_i y_i \phi(x_0)^T \phi(x_i) + w_0\right) = \text{sign}\left(\sum_{i=1}^{\hat{}} \alpha_i y_i K(x_0, x_i) + w_0\right)$$

- see diagram / draw
- kernelised SVM (with slack) produces a non-linear decision boundary under the classification rule $\text{sign}\left(\sum_{i=1}^{\hat{}} \alpha_i y_i K(x_0, x_i) + w_0\right)$
- Black dots are support vectors where $\alpha_i > 0$; most points will have $\alpha_i = 0$
- After estimating SVM; can discard non-support vectors, indexed by $\alpha_i = 0$, and calculate $K(x_0, x_i)$ between new point and support vectors.

Summary

- Basic SVM: linear classifier (assuming linearly separable), affine hyperplane determined to maximise margin; convex dual ⇒ exact global solutions can be found
- Fully fledged SVM (versatile) + extensions:
  - maximum margin : good generalisation properties
  - slack variables : overlapping classes; robust against outliers
  - kernel : non-linear decision boundary (in original space)
- Practical implementation:-
  - software package
  - choose kernel fn (how?)
  - cross validate for hyperparameters λ (penalty on slack variables) and RBF kernel width b.