1. Initialise $\underline{w}^{(1)} = \vec{0}$

2. For step $t = 1, \dots,$ do:

a) search for all $(\underline{x}_i, y_i) \in D : y_i \neq \text{sign}(\underline{x}_i^T \underline{w})$

b) If $(\underline{x}_i, y_i)$ exists, randomly select and update $\underline{w}^{(t+1)} = \underline{w}^{(t)} + \eta x_i y_i$ (*)

else: Return $\underline{w}^{(t)}$ as solution as everything classified

(*) $\nabla_{\underline{w}} \mathcal{L} = \nabla_{\underline{w}} - \sum_{i=1}^{N} y_i (\underline{x}_i^T \underline{w}) \mathbb{I}(y_i \neq \text{sign}(\underline{x}_i^T \underline{w})) = - \sum_{i \in M_t} x_i y_i$ M_t - misclass. obs in step t. index set

(SGD) $\underline{w}' \leftarrow \underline{w} - \eta(-x_i y_i) \Rightarrow w' \leftarrow w + \eta y_i x_i$ oppos. direction of $\underline{w}$

SGD only selects one element $(\underline{x}_i, y_i)$

Perceptron issues: 1) linear sep $\Rightarrow \infty$ # H ✓ optimal?

2) NO qual. consid, just 1st locally optimal (not necess globally)

3) NO linear separability $\rightarrow$ no convergence

4) Slow or just not converging

-Key equations

Linear classifiers: data $(\underline{x}_1, y_1), \dots, (\underline{x}_N, y_N)$ $\underline{x}_i \in \mathbb{R}^d$ $y_i \in \{-1, +1\}$

Prediction: Estimates $\hat{\underline{w}}, \hat{w}_0$; $\hat{y}_i = f(\underline{x}_i; \hat{\underline{w}}, \hat{w}_0) = \text{sign}(\underline{x}_i^T \hat{\underline{w}} + \hat{w}_0)$

LS - sensitivity to outliers

Perceptron - convergence/linear sep issues

Log-odds: Bayes classifier linear class. rule: $y \sim \text{Bern}(\pi)$, $\underline{x}|y \sim N(\mu_R, \Sigma)$ declare $y=1$ given $\underline{x}$ if

$$\ln\left[\frac{p(y=1)p(\underline{x}|y=1)}{p(y=0)p(\underline{x}|y=0)}\right] > 0 \; ; \text{ log-odds} = w_0 + \underline{x}^T \underline{w} \text{ OR } c + \underline{x}^T b + \underline{x}^T A \underline{x}$$

with $f(x) = \text{sign}(\underline{x}^T \underline{w} + w_0)$ OR $\text{sign}(\underline{x}^T A \underline{x} + \underline{b} + c)$

1	> 0	> 0
-1	< 0	< 0

Posterior log-odds:

(discriminative) log-odds $= \ln\left[\frac{p(y=1|\underline{x})}{p(y=0|\underline{x})}\right] > 0$ (L) then declare $y=1$ given $\underline{x}$

Logistic regression: no recourse to specification of (funct. form) of priors, CCDs but directly specifies $p(y|\underline{x})$

Discriminative log-odds: $L = \ln\left[\frac{p(y=1|\underline{x})}{p(y=0|\underline{x})}\right]$ (classification)

- $L \gg 0 \Rightarrow$ more conf $y_i = +1$
- $L \ll 0 \Rightarrow$ less conf $y_i = -1$
- $L = 0 \Rightarrow$ either way

Function: $\underline{x}^T \underline{w} + w_0 \Rightarrow$ i) distance $\underline{x}$ to hyperplane $H(w_0, \underline{w})$ is $\left|\frac{\underline{x}^T \underline{w}}{\|\underline{w}\|_2} + \frac{w_0}{\|\underline{w}\|_2}\right|$ ✓ + iii) confidence

ii) $\text{sign}(\underline{x}^T \underline{w} + w_0) = \text{sign}(\cos\theta) \rightarrow$ side

Link function: $\ln\left[\frac{p(y=1|\underline{x})}{p(y=0|\underline{x})}\right] = \underline{x}^T \underline{w} + w_0$ (log-odds $\leftrightarrow$ sep hyperplane)

(DN): Here, discriminative (non-ge.) specification $\Rightarrow$ no restrictions on $(w_0, \underline{w})$ via prior, CCD functional forms

logistic regression (model): $p(y=+1|\underline{x}) = \frac{\exp(\underline{x}^T\underline{w}+w_0)}{1+\exp(\underline{x}^T\underline{w}+w_0)} = \sigma(\underline{x}^T\underline{w}+w_0)$ · $\underline{x}^T\underline{w}+w_0$ - lin R fn. · $\sigma(z)$ - logistic sigmoid

· **Properties (sigmoid)**: $\sigma(z) = \frac{e^z}{1+e^z} = \frac{e^z}{1+e^z}$; $z = \underline{x}^T\underline{w}+w_0$ $\begin{cases} z=0 \Rightarrow \sigma(z)=\frac{1}{2} \\ z\to-\infty \Rightarrow \sigma(z)\to 0 \\ z\to\infty \Rightarrow \sigma(z)\to 1 \end{cases}$ $\sigma(\cdot)$ - captures confidence as we move away from $H(\underline{w},w_0)$

· **Absorb offsets**: $\underline{w} \leftarrow \begin{bmatrix} w_0 \\ \underline{w} \end{bmatrix}$ $\underline{x}_i \leftarrow \begin{bmatrix} 1 \\ \underline{x}_i \end{bmatrix}$ $\underline{w} \in \mathbb{R}^{d+1}$ $\underline{x}_i \in \mathbb{R}^{d+1}$

· **LR model**: Given $\mathcal{D} = (\underline{x}_1, y_1) \dots (\underline{x}_N, y_N)$ with $y_i \in \{-1,+1\}$
- model each y_i as independently gen. with $\begin{cases} p(y_i=+1|\underline{x}_i;\underline{w}) = \sigma(\underline{x}_i^T\underline{w}) \\ p(y_i=-1|\underline{x}_i;\underline{w}) = 1-\sigma(\underline{x}_i^T\underline{w}) \end{cases}$

logistic parameter estimation: define $\sigma_i(\underline{w}) = \sigma(\underline{x}_i^T\underline{w})$

Joint likelihood $(y_1,\dots,y_n)$: $p(y_1,\dots,y_n|\underline{x}_1,\dots,\underline{x}_n;\underline{w}) = \prod_{i=1}^N p(y_i|\underline{x}_i;\underline{w})$
$= \prod_{i=1}^N \sigma_i(\underline{w})^{\mathbb{I}(y_i=+1)} (1-\sigma_i(\underline{w}))^{\mathbb{I}(y_i=-1)}$

Abbreviated JL: $p(y_1,\dots,y_n|\underline{x}_1,\dots,\underline{x}_n;\underline{w}) = \prod_{i=1}^N \sigma_i(y_i\cdot\underline{w})$ - ML estimation ≠ closed-form!

with $\sigma_i(y_i\cdot\underline{w}) = \sigma_i(\underline{w})^{\mathbb{I}(y_i+1)} (1-\sigma_i(\underline{w}))^{\mathbb{I}(y_i=-1)}$

maximising joint likelihood (MLE estimation): $\hat{\underline{w}}_{ML} = \arg\max_{\underline{w}} \ln\left(\prod_{i=1}^N \sigma_i(y_i\cdot\underline{w})\right) = \arg\max_{\underline{w}} \sum_{i=1}^N \ln\sigma_i(y_i\cdot\underline{w}) := \arg\max_{\underline{w}} l$

steepest ascent (iterat. algo):- At time t; $\underline{w}^{(t+1)} = \underline{w}^{(t)} + \eta\nabla_{\underline{w}} l$ $\nabla_{\underline{w}} l = \sum_{i=1}^N (1-\sigma_i(y_i\cdot\underline{w}))y_i\underline{x}_i$

logistic regression algorithm:

⌜IN: $\mathcal{D} = (\underline{x}_1,y_1),\dots,(\underline{x}_N,y_N)$; $\eta>0$

1. set $\underline{w}^{(1)} = \vec{0}$
2. For step $t=1,2,\dots$ do:
· update $\underline{w}^{(t+1)} = \underline{w}^{(t)} + \eta\sum_{i=1}^N (1-\sigma_i(y_i\cdot\underline{w}))y_i\underline{x}_i$ (stochastic gradient descent) (one element)

Perceptron vs logistic: Perceptron → search for misclass $(\underline{x}_i,y_i)$; update $\underline{w}^{(t+1)} = \underline{w}^{(t)} + \eta y_i\underline{x}_i$ ⌟
logistic → simila, except one all data
$(1-\sigma_i(y_i\cdot\underline{w}))$ - probability assigned to wrong value
· magnitude of confidence in prediction included in update (probabilistic)

Motivation for L^2 regularisation (ad MAP estimation, Gaussian priors): · perfect linear separability; $\|\underline{w}_{ML}\|_2 \to \infty \Rightarrow \sigma_i(y_i\cdot\underline{w}) \to 1$ for each $(\underline{x}_i,y_i)$
· class ⇒ few very wrong to be more confident → overfitting

· **L^2 regularisation (logistic)**: $\hat{\underline{w}}_{MAP} = \arg\max_{\underline{w}} l = \arg\max_{\underline{w}} \sum_{i=1}^N \ln(\sigma_i(y_i\cdot\underline{w})) - \frac{\lambda}{2}\underline{w}^T\underline{w}$

Bayesian corresponds to Gaussian prior $p(\underline{w}) = \mathcal{N}(\underline{w}|\underline{0},\lambda^{-1}\underline{I})$

- Cannot find (Cont) $p(\underline{w}|y,\underline{X})$ i.e. posterior of $\underline{w}$ as $p(\underline{w}|y,\underline{X}) = \frac{\left(\prod_{i=1}^{N}\sigma_i(y_i\cdot\underline{w})\right)p(\underline{w})}{\int_{\mathbb{R}^M}\prod_{i=1}^{N}\sigma_i(y_i\cdot\underline{w})\,p(\underline{w})\,d\underline{w}}$
- Likelihood model for class labels: sigmoid $\prod_{i=1}^{N}\sigma_i(y_i\cdot\underline{w})$
- Prior: $p(\underline{w})$
- $\int_{\mathbb{R}^{d+1}}\prod_{i=1}^{N}\sigma_i(y_i\cdot\underline{w})p(\underline{w})\,d\underline{w}$ - intractable (approx via MCMC)

- Laplace approx: select gaussian distr family $N(\cdot)$ to approx $p(\underline{w}|y,\underline{X})$
- Require method for setting $\underline{\mu}$ and $\underline{\Sigma}$
- As $\prod_{i=1}^{N}\sigma_i(y_i\cdot\underline{w})\,p(\underline{w}) = p(\underline{w}|y,\underline{X})p(\underline{w}) = p(y,\underline{w}|\underline{X})$ and $p(\underline{w}|y,\underline{X}) = \frac{e^{\ln p(y,\underline{w}|\underline{X})}}{\int_{\mathbb{R}^n} e^{\ln p(y,\underline{w}|\underline{X})}\,d\underline{w}}$
- Approximate $\ln p(\underline{w},y|\underline{X})$ in numerator and denom.

2nd order taylor expans: $f(\underline{w}) = \ln p(y,\underline{w}|\underline{X})$ (suppress y and $\underline{X}$)

Approx. $f(\underline{w}) = \ln(p(y,\underline{w}|\underline{X}))$ with 2nd order Taylor expansion

$f(\underline{w}) \approx f(\underline{z}) + (\underline{w}-\underline{z})^T\nabla f(\underline{z}) + \frac{1}{2}(\underline{w}-\underline{z})^T\nabla^2 f(\underline{z})(\underline{w}-\underline{z})$ $\quad \nabla f(\underline{z}) = \nabla_{\underline{w}} f(\underline{w})|_{\underline{z}}$

[...] ; set $\underline{z} = w_{MAP}$ (Taylor expans around mode)

summary: For $\{x_i, y_i\}\ \forall i=1,\dots,N$; likelihood $p(y_i|x_i;\underline{w}) = \sigma(y_i x_i^T\underline{w})$ prior $\underline{w}\sim N(\vec{0}, \lambda^{-1}\underline{I})$ MVG

Laplace approx of post. for logistic (SM)

1. Find $\hat{w}_{MAP} = \underset{\underline{w}}{\operatorname{argmax}} \sum_{i=1}^{N}\ln\sigma(y_i x_i^T\hat{w}_{MAP}) - \frac{\lambda}{2}\underline{w}^T\underline{w}$
2. Set $-\underline{\Sigma}^{-1} = -\lambda\underline{I} - \sum_{i=1}^{N}\sigma_i(y_i x_i^T\hat{w}_{MAP})\left(1-\sigma_i(y_i x_i^T\hat{w}_{MAP})\right)\underline{x}_i^{m}\underline{x}_i^T$
3. Approximate $p(\underline{w}|y,X)$ with $N(w_{MAP},\underline{\Sigma})$ $\underline{\Sigma}$ - inverse of Hessian of

0 - Key equations

ature mappings: A feature mapping function $\phi(\cdot): \underline{x}\mapsto\phi(\underline{x})$ $\phi:\mathbb{R}^d\to\mathbb{R}^D$ $(D>d)$

solves issue of linear model in original feature space $\mathbb{R}^d$ not working

e #1: Polynom gression on $\mathbb{R}$ $x\in\mathbb{R}$ $\phi(x) = (x, x^2, \dots, x^p)$ $\phi:\mathbb{R}\to\mathbb{R}^p$

e #2: Discontinuities: $x\in\mathbb{R}$ $\phi(x) = (x, \mathbb{I}\{x<a\})$ $\phi:\mathbb{R}\to\mathbb{R}^2$

ture expansions to $\mathbb{R}^D$: High dim. maps transform data so output is linear in inputs (in high-dim space)
gression)

$x_i\in\mathbb{R}$, $y_i\in\mathbb{R}$ $w\in\mathbb{R}$, $w_0\in\mathbb{R}$ $\Rightarrow \phi(x): x\mapsto\begin{bmatrix}x\\ \cos x\end{bmatrix}$ $\phi:\mathbb{R}\to\mathbb{R}^2$ (extra dim $\cos x$)

$w_0 + \phi(x)^T\underline{w} \hat{=} w_0 + w_1 x + w_2\cos x$ LR → note $\phi:\mathbb{R}^d\to\mathbb{R}^D$ induces dim increase in w: $d\to D$

non-linearities in $\mathbb{R}^d$ transformed to $\mathbb{R}^D$ where problem linear

ure expansions in $\mathbb{R}^D$: High dim. maps transform data so output is linearly seperable
fication)

$x_i\in\mathbb{R}^2$, $y_i\in\{-1,+1\}$ $\underline{w}\in\mathbb{R}^2$, $w_0 \Rightarrow \phi(\underline{x}): \underline{x}\mapsto\begin{bmatrix}x_1^2\\ x_1x_2\\ x_2^2\end{bmatrix}$ $\phi:\mathbb{R}^2\to\mathbb{R}^3$

$\operatorname{sign}(w_0+\phi(\underline{x})^T\underline{w}) = \operatorname{sign}(w_0 + w_1x^2 + w_2x_1x_2 + w_3x^2)$ $w_\phi\in\mathbb{R}^3$

Kitchen sink / ℓ_1 regularisation:

- Add lots of feature exp., high dim as poss, one add above D
- Estimate a weight vector $\underline{w} \in \mathbb{R}^D$:- $\hat{w}_{\ell_1} = \arg\min_{\underline{w}} \sum_{i=1}^{N} f(y_i, \phi(\underline{x}_i), \underline{w}) + \lambda \|w\|_1$
- L_1 penalty $\Rightarrow$ sparse subset of dimensions of $\phi(\underline{x})$ used.

(OR): Do not work with $\phi(\underline{x}) \in \mathbb{R}^D$ directly, rather dot products $\phi(\underline{x}_i)^T \phi(\underline{x}_j) \equiv K(\underline{x}_i, \underline{x}_j)$

Feature exp. perceptron :- $\underline{x}_i \in \mathbb{R}^{d+1}$ $y_i \in \{-1, +1\}$ $i = 1, \dots, N$ $\underline{w} \in \mathbb{R}^{d+1}$

perceptron $\rightarrow$ hyperplane $\underline{w} = \sum_{i \in M} y_i \underline{x}_i$ · M - sequentially constructed set of examples

· Unfolding of $\underline{w}^{(t+1)} = \underline{w}^{(t)} + \eta \underline{x}_i y_i$ (assuming $\eta = 1$)

· Prediction/classification: for new $\underline{x}_0$, $y_0 = \text{sign}(\underline{x}_0^T \underline{w}) = \text{sign}\left(\sum_{i \in M} y_i \underline{x}_0^T \underline{x}_i\right)$

With feature exp: $\underline{x}_0 \leftarrow \phi(\underline{x}_0)$ $\underline{x}_i \leftarrow \phi(\underline{x}_i)$ $\underline{w} = \sum_{i \in M} y_i \phi(\underline{x}_i)$

$$y_0 = \text{sign}(\phi(\underline{x}_0)^T \underline{w}) = \text{sign}\left(\sum_{i \in M} y_i \phi(\underline{x}_0)^T \phi(\underline{x}_i)\right) \rightarrow K(\underline{x}_0, \underline{x}_i)$$

Kernels: A kernel $K(\cdot,\cdot) : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}$ is a symmetric function: for any N data points (define) $\underline{x}_1, \dots, \underline{x}_N \in \mathbb{R}^d$, the $N \times N$ kernel matrix $\underline{K}$ with $\underline{K}_{ij} = \phi(\underline{x}_i)^T \phi(\underline{x}_j) \in S_+^N$ (PSD)

(i) (ii)

(i): symmetric: $K(\underline{x}_i, \underline{x}_j) = K(\underline{x}_j, \underline{x}_i)$ (as true: $\underline{K} = \underline{K}^T$)

(ii): PSD: $M \in \mathbb{R}^{N \times N} \in S_+$ iff $\underline{z}^T M \underline{z} \geq 0$ $\forall \underline{z} \in \mathbb{R}^N$ and $\underline{z} \neq \vec{0}$

Mercer Kernel: $\underline{K}$ satisfies properties of covariance Σ

Mercer's Theorem: If $K(\cdot,\cdot)$ satisfies above, then the exists a feature mapping (valid kernel functions as f. exp.) $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$: $K(\underline{x}_i, \underline{x}_j) = \phi(\underline{x}_i)^T \phi(\underline{x}_j)$ (i.e. no need for $\phi(\cdot)$ directly)

· For finite N (data), a kernel matrix $\underline{K}$ is a valid Mercer kernel iff $\underline{K}$ is symmetric positive-semi definite

RBFs / Gaussian kernel: $K(\underline{x}, \underline{x}') = a \exp\left\{-\frac{1}{b}\|\underline{x} - \underline{x}'\|_2^2\right\}$ · Proximity in $\mathbb{R}^d$ · b - kernel bandwidth · $\underline{x}$ close to $\underline{x}' \rightarrow K(\underline{x}, \underline{x}') \rightarrow a$ · otherwise $K(\underline{x}, \underline{x}') \rightarrow 0$

(assume Mercer's theorem is true)

· RBF feature mapping $\phi_{RBF}(\underline{x}) \in \mathbb{R}^\infty$ (continuous function not vector)

· $K(\underline{x}, \underline{x}') = \int \phi_t(\underline{x}) \phi_t(\underline{x}') dt$ $\phi_t(\underline{x})$ - function with param $\underline{x}$, lies on Hilbert / Banach space

Other kernels and properties:

#1: $\phi(\underline{x}) = (1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots)$

$K(\underline{x}, \underline{x}') = \phi(\underline{x})^T \phi(\underline{x}') = (1 + \underline{x}^T \underline{x}')^2$ - valid

#2 $K(\underline{x}, \underline{x}') = (1 + \underline{x}^T \underline{x}')^b$ $b > 0$ - valid

Kernel validity under arithmetic: For valid K_1, K_2:

$K_1(\underline{x}, \underline{x}') K_2(\underline{x}, \underline{x}') = K(\underline{x}, \underline{x}')$ (product) $K_1(\underline{x}, \underline{x}') + K_2(\underline{x}, \underline{x}') = K(\underline{x}, \underline{x}')$ (addition)

$\exp(K_1(\underline{x}, \underline{x}')) = K(\underline{x}, \underline{x}')$ (exponent.)

Kernelised perceptron: Feature expanded decision: $y_0 = \text{sign}\left(\sum_{i \in M} y_i \phi(x_0)^T \phi(x_i)\right)$

Kernelised: $y_0 = \text{sign}\left(\sum_{i \in M} y_i K(x_0, x_i)\right)$, use with RBF Kernel $\alpha = 1$

(!!): soft voting: each misclassified $(x_i, y_i) \in M$ votes for label for new x_0

Kernelised perceptron: algorithm

Train: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$; $\eta > 0$

1. Initialise $w^{(1)} = \vec{0}$
2. For step $t = 1, \ldots,$ do:
   a) search for all $(x', y') \in D$: $y' \neq \text{sign}\left(\sum_{i \in M_t} y_i K(x', x_i)\right)$, given $w^{(t)} = \sum_{i \in M_t} y_i x_i$
   b) If (x', y') exists select, add index x' to M, update $w^{(t+1)} = \sum_{i \in M_{t+1}} y_i x_i$
   Else: Return $w^{(t)}$ as solution

- Do not ever explicitly work with $\phi(x_i)$ or $\phi(x_0)$. Declare $y_0 = \text{sign}\left(\sum_{i \in M} y_i K(x_0, x_i)\right)$ mt. (*)
- classification → only involves $K(x_0, x_i)$

Kernelised K-NN: (class.f.)

- Generalise kernelised perceptron → soft K-NN via simple mod.
- Define linear classifier by $\sum_{i=1}^N$ rather than $\sum_{i \in M}$ (i.e. let all points vote; not just misclass.)
- For new x_0; declare $y_0 = \text{sign}\left(\sum_{i=1}^N y_i p_i(x_0)\right)$ with $p_i(x_0) = \frac{1}{Z} e^{-\frac{1}{b}\|x_0 - x_i\|_2^2}$, $Z = \sum_{j=1}^N e^{-\frac{1}{b}\|x_0 - x_j\|_2^2}$ (i.e. normalised RBF)
- Confidence score; probability distr. over outcomes x_0; proximity to x_0 (via set b)
- Points close to query x_0 vote through $p(x_0)$
- Difference: Replace $\sum_{i \in M}$ with $\sum_{i=1}^N$; replace $K(x_0, x_i)$ with normalised RBF kernel to yield $p_i(x_0)$

Nadaraya-Watson: (regression)
- $y \in \mathbb{R}$
- RBF Kernel
- New x_0; $y_0 = \sum_{i=1}^N y_i \left(\frac{K(x_0, x_i)}{\sum_{j=1}^N K(x_0, x_j)}\right)$
- Predict y_0 via locally weighted average of responses y_i, defined by probabilistic (rel.) prox. to x_0 (def. by Kernel Bandwidth b)

Kernelised Bayesian LR: (model)
- Given:-
  likelihood $p(y|w, X)$ $y \sim N(Xw, \sigma^2 I)$
  prior $p(w|X)$ $w \sim N(\vec{0}, \lambda^{-1} I)$

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

- As $(XX^T)_{ij} = x_i^T x_j$, replace $x \leftarrow \phi(x)$ to form $\Phi(X)\Phi(X)^T$ with $(\Phi(X)\Phi(X)^T)_{ij} = \phi(x_i)^T \phi(x_j)$
  i.e. $(\Phi(X)\Phi(X)^T)_{ij} = K(x_i, x_j)$
- Can define K directly, known as Gaussian process

defining K directly: $\phi(\underline{x})$ feature mapping may be infinite dim. $\Rightarrow w$ infinite dim

- Hence $\Phi(X)$, $\Phi(X)\Phi(X)^T$ and $\underline{w} \sim N(0, \lambda^{-1}\underline{I})$ all infinite dim.
- (2nd): Instead, evaluate kernel fn $K(\cdot,\cdot)$ over all pairs of data points $(\underline{x}_i, \underline{x}_j)$ for which we have finite N data points $\Rightarrow$ N-dim MVG and hence can calculate $\underline{K}$.

Gaussian processes

Key motivation: instead of defining a probability distribution over parameter $\underline{w}$ i.e. a Gaussian prior $p(\underline{w})$, which induces a probability distribution over regression functions $f(\underline{x}; \underline{w})$, we do this directly by defining a probability distribution over regression functions themselves

Gaussian process: (Definition)
- $f(\underline{x})$ $\underline{x} \mapsto f(\underline{x})$ $f: \mathbb{R}^d \to \mathbb{R}$ $\underline{x} \in \mathbb{R}^d$
- define $K(\underline{x}, \underline{x}')$ between arbitrary $\underline{x}$ and $\underline{x}'$

(*) (a) intuitions lecture

$f(\underline{x})$ is the G.P. and $y(\underline{x})$ the noise-added process

$\underline{y} | \underline{f} \sim N(\underline{f}, \sigma^2 \underline{I})$ $\underline{f} \sim N(\vec{0}, \underline{K})$ $\underline{\epsilon} \sim N(\vec{0}, \sigma^2 \underline{I}) \Leftrightarrow \underline{y} \sim N(0, \sigma^2 \underline{I} + \underline{K})$

$y = [y_1, \dots, y_N]^T$ $\underline{K}_{ij} = K(\underline{x}_i, \underline{x}_j)$

The domain of G.P. is entirety of $\mathbb{R}^d \to$ infinite dimensional

Kernel is constructed for N data points $\Rightarrow (\underline{K}_{ij}) = K(\underline{x}_i, \underline{x}_j)$ $\forall i, j = 1, \dots, N$

Select N points in $\mathbb{R}^d$, restrict domain of f, evaluate f at $f(\underline{x}_1), \dots, f(\underline{x}_N)$ and stack into $\underline{f} \in \mathbb{R}^N$ vector with $f_i = f(\underline{x}_i)$ element

$\underline{f}$ vector has mean function $\vec{0}$ and covariance function $\underline{K}$

Observed $(y_1, \dots, y_N)$ is a noisy sub-sampling of infinite dimensional function over all $\mathbb{R}^d$, but we evaluate at N data points to yield $\underline{f}(\underline{x}) \in \mathbb{R}^N$ with finite noise $\underline{\epsilon}$

example #1 (see diagram)
- $\underline{x} \in [0, 1]$ $\cdot \underline{K} \in [0,1]^{1000 \times 1000}$ $K_{ij} = \exp\left\{ -\frac{\|x_i - x_j\|_2^2}{b} \right\}$
- $N = 1{,}000$
- Evaluate $f(x_1), \dots, f(x_{1000})$, stack into $\underline{f}^{(1)} = \begin{pmatrix} f^{(1)}(x_1) \\ \vdots \\ f^{(1)}(x_{1000}) \end{pmatrix}$
- Each $\underline{f} \in \mathbb{R}^{1000}$ vector generated from G.P. (restricted to a $d = 1000$ MVG with $m = \vec{0}$ $cov = \underline{K}$)

Model hypothesis; there is an observed function f underlying DGP, only $n = 25$ noisy samples of f i.e. $(x_1, y_1) \dots (x_{25}, y_{25})$

Predictive distribution (see G.Ps.)
- N observation pairs $\{(\underline{x}_i, y_i)\}_{i=1}^N$, predict y_0 given $\underline{x}_0$
- $(\underline{x}_0, y_0)$ partially observed, y_0 unknown
- (?) independent conditioned on parameter $\underline{w}$

Next marginal likelihood/evidence: $y \sim N(0, \sigma^2 I + \lambda^{-1} \underline{X} \underline{X}^T)$

Joint distribution (of y_0 and $\underline{y}$): $\begin{bmatrix} y_0 \\ \underline{y} \end{bmatrix} \sim \text{Normal}\left(\vec{0}, \sigma^2 I + \begin{bmatrix} \underline{x}_0^T \underline{x}_0 & (\underline{X}\underline{x}_0)^T \\ \underline{X}\underline{x}_0 & \underline{X}\underline{X}^T \end{bmatrix}\right)$

- integrating out $\underline{w}$
- Recall that $p(\underline{y}|\underline{X}, \lambda) = \int_{\mathbb{R}^d} p(\underline{y}|\underline{w}, \underline{X})\, p(\underline{w}|\lambda)\, d\underline{w}$ (parametric view)
- G.P. perspective: $p(\underline{y}) = \int_{\mathbb{R}^d} p(\underline{y}|f)\, p(f)\, df$ (G.P. view) (*) (2)

Predicting y_0: $y_0|D; \underline{x}_0 \sim N(\mu_0, \sigma_0^2)$ $\mu_0 = (\underline{X}\underline{x}_0)^T (\underline{X}\underline{X}^T)^{-1} \underline{y}$

(cond distr of y_0 given old D, new $\underline{x}_0$, marginal like of y_0, predictive distn of y_0) $\sigma_0^2 = \sigma^2 + \underline{x}_0^T \underline{x}_0 - (\underline{X}\underline{x}_0)^T (\underline{X}\underline{X}^T)^{-1} (\underline{X}\underline{x}_0)$

Augmented covariance (with test $\underline{x}_0$): $\begin{bmatrix} \underline{x}_0^T \underline{x}_0 & (\underline{X}\underline{x}_0)^T \\ \underline{X}\underline{x}_0 & (\underline{X}\underline{X}^T) \end{bmatrix} = \begin{bmatrix} \underline{x}_0^T \underline{x}_0 & \underline{X}\underline{x}_0^T \\ \underline{X}\underline{x}_0 & \underline{X}\underline{X}^T \end{bmatrix}$ (1)

Predictions with G.P.:- Given observed $D = \{(\underline{x}_i, y_i)\}_{i=1}^N$; distribution of $y(\underline{x})$ at any new $\underline{x}$ to predict (*) (2)

(over predictive cd posterior)

Predictive $y(\underline{x})$ (G.P.):

- $y(\underline{x})|D_n \sim N(\mu(\underline{x}), \Sigma(\underline{x}))$

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

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

Posterior $f(\underline{x})$ (G.P.)

- Remove σ^2
- (fig) see diagram, rewatch, intuitions

G.P.: An example of non-parametric regression, arbitrary function is being "estimated" from a space of possible functions drawn from a Gaussian process prior → interpolate and smooth based on data

VII - Key equations

Maximum margin classifiers: linear classification and linear separability

Maximum margin class.: (qualitative criterion) Achieve good generalisation, low prediction error PI place a hyperplane $H(w_0, \underline{w})$ such that distance to closest point in each class maximised (margin)

Generalisation error: see diagrams for perceptron vs QDA vs max margin

Observations on class hyperp.:- separating hyperplane contingent on "outer" points, not centre

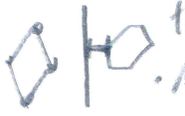
- Represent each class by a convex hull
- convex hull - smallest convex set which contains all points in the class

Convex Hull: · A convex hull is defined by all possible weighted averages of points in a set.

· For data points $(\underline{x}_1, \cdots, \underline{x}_n)$, every point $\underline{x}_0$ in shaded region (conv. hull) can be reached

· $\underline{x}_0 = \sum_{i=1}^{N} \alpha_i \underline{x}_i \quad \alpha_i \geq 0 \quad \sum_{i=1}^{N} \alpha_i = 1$ for some $(\alpha_1, \cdots, \alpha_n)$ · No point outside can be reached like this.

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



· maximising margin ⇒ H is in the 'middle' of 2 convex hulls

· maximising shortest distance between plane and any point in either set (OR)

SVM (primal): For N linearly separable $\{(\underline{x}_i, y_i)\}_{i=1}^{N} \quad y_i \in \{-1, +1\}$

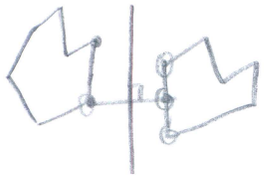
((OR)) · Solve: $\min_{\underline{w}, w_0} \frac{1}{2}\|\underline{w}\|_2^2$ s.t. $y_i(\underline{x}_i^T \underline{w} + w_0) \geq 1 \quad \forall n = 1, \cdots, N$ (i)

(i): $y_i(\underline{x}_i^T \underline{w} + w_0) > 0$ if $y_i = \text{sign}(\underline{x}_i^T \underline{w} + w_0)$ (constant on point being classified correctly)

(ii): Existence of H ⇒ scale $\underline{w}$: $y_i(\underline{x}_i^T \underline{w} + w_0) > 1$ for all i

· **Solution intuition:** 1. Find closest two points from convex hull of class $+1$ and -1

2. Connect with a line, place perpendicular hyperplane



3. If S_1 and S_0 are $+1, -1$; we want two prob. vectors $(\underline{\alpha}_0, \underline{\alpha}_1)$ to minimise $\left\| \left(\sum_{\underline{x}_i \in S_1} \alpha_{1i} \underline{x}_i\right) - \left(\sum_{\underline{x}_i \in S_0} \alpha_{0i} \underline{x}_i\right) \right\|_2$

4. Define hyperplane using two points found with $(\underline{\alpha}_0, \underline{\alpha}_1)$

Lagrangian: · Define N $\alpha_i > 0$ $i = 1, \cdots, N$ (Lagrange multipliers)

$$L_p(\underline{w}, w_0, \underline{\alpha}) = \frac{1}{2}\|\underline{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i (y_i(\underline{x}_i^T \underline{w} + w_0) - 1)$$

(?) Minimise L over $\underline{w}$ and w_0 and maximise over $(\alpha_1, \cdots, \alpha_n)$

$$L_p(\underline{w}, w_0, \underline{\alpha}) = \frac{1}{2}\|\underline{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i y_i (\underline{x}_i^T \underline{w} + w_0) + \sum_{i=1}^{N} \alpha_i$$

· **Minimise over $(\underline{w}, w_0)$:** $\nabla_{\underline{w}} L$ and $\frac{\partial L}{\partial w_0}$ ⇒ (I) $\underline{w} = \sum_{i=1}^{N} \alpha_i y_i \underline{x}_i$ (II) $\sum_{i=1}^{N} \alpha_i y_i = 0$ → $L(\underline{w}, w_0, \underline{\alpha})$

· **SVM (dual):** Solve: $\max_{(\alpha_1, \cdots, \alpha_n)} L_D(\underline{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\underline{x}_i^T \underline{x}_j)$ s.t. $\sum_{i=1}^{N} \alpha_i y_i = 0$

$\alpha_i \geq 0 \quad \forall i = 1, \cdots, N$

· **Maximisation (α_i)s:** use iterative algorithm (discussion of dual sol not focused on) to solve for $(\hat{\alpha}_1, \cdots, \hat{\alpha}_n)$

· **Dual solution → primal:** (I) $\hat{\underline{w}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \underline{x}_i$ (II) $\hat{w}_0$: As sol. ⇒ $\alpha_i y_i (\underline{x}_i^T \hat{\underline{w}} + w_0) = 0 \quad \forall i$

select i for which $\alpha_i > 0$, solve $y_i(\underline{x}_i^T \hat{\underline{w}} + w_0) = 0$ for w_0

SVM dual (convex hull form)

Solve: $\max_{(\alpha_1,\dots,\alpha_n)} L_D(\underline{\alpha}) = 2C - \frac{1}{2}C^2 \left\| \sum_{i\in S_1} \frac{\alpha_i \underline{x}_i}{C} - \sum_{j\in S_0} \frac{\alpha_j \underline{x}_j}{C} \right\|_2^2$

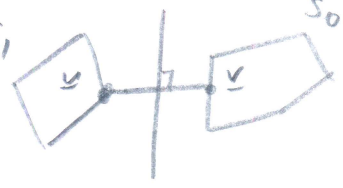
s.t. $C = \sum_{i\in S_1} \alpha_i = \sum_{j\in S_0} \alpha_j$, $\alpha_i \geq 0$ - AS $\begin{cases} 2C = \sum_{i=1}^N \alpha_i \left(\text{from } \sum_{i=1}^N \alpha_i y_i = 0\right) (II) \end{cases}$

SVM dual (interpolate)

$\max_{(\alpha_1,\dots,\alpha_n)} L_D(\underline{\alpha}) \Leftrightarrow \min_{(\alpha_1,\dots,\alpha_n)} \left\| \underbrace{\left(\sum_{i\in S_1} \frac{\alpha_i \underline{x}_i}{C}\right)}_{\text{in conv hull } S_1} - \underbrace{\left(\sum_{j\in S_0} \frac{\alpha_j \underline{x}_j}{C}\right)}_{\text{in conv hull } S_0} \right\|_2^2 \begin{cases} \frac{1}{2}C^2 \left\| \sum_{i\in S_1} \frac{\alpha_i \underline{x}_i}{C} - \sum_{j\in S_0} \frac{\alpha_j \underline{x}_j}{C} \right\|_2^2 = \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\underline{x}_i^T \underline{x}_j) \end{cases}$

- Dual problem is finding the closest points (Euclidean distance) in convex hulls between class +1 and −1

Direction of classifier (defined by H):- Find $\min_{\underline{v}\in H(S_1)} \min_{\underline{v}\in H(S_0)} \|\underline{v} - \underline{v}\|_2^2$



- Direction of $\underline{w}$ is $(\underline{v} - \underline{v})$ (?)

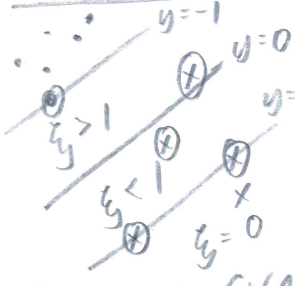
Solution: 1. Find shortest line connecting convex hulls
(formal) 2. Place hyperplane orthog. to line at midpoint

SVM
- minimise $\|\underline{w}\|_2^2$, solution:

$\underline{w} = \sum_{i=1}^N \alpha_i y_i \underline{x}_i = C\left(\sum_{i\in S_1} \frac{\alpha_i}{C}\underline{x}_i - \sum_{j\in S_0} \frac{\alpha_j}{C}\underline{x}_j\right)$

No linear separability: Data not linearly sep., allow training data on wrong side, at cost

1. Replace training rule in SVM :- $y_i(\underline{x}_i^T \underline{w} + w_0) \geq 1 \;\forall i = 1,\dots,N$ with $y_i(\underline{x}_i^T \underline{w} + w_0) \geq 1 - \xi_i$ with $\xi_i \geq 0 \;\forall i = 1,\dots,N$



2. ξ_i - slack variables

$\times$ | support vectors $\alpha_i > 0$

$\odot, \circledast$

Soft-margin SVM objective (primal) :- Solve $\min_{\underline{w}, w_0, \xi_1,\dots,\xi_n} \frac{1}{2}\|\underline{w}\|_2^2 + \lambda \sum_{i=1}^N \xi_i$ s.t. $\xi_i \geq 0 \;\forall i = 1,\dots,N$ and $y_i(\underline{x}_i^T \underline{w} + w_0) \geq 1 - \xi_i \;\forall i = 1,\dots,N$

- λ - hyperparameter
  - $\rightarrow \lambda > 0$
  - $\rightarrow$ specifies cost/penalty of training point on wrong side
  - $\rightarrow \lambda$ small $\rightarrow$ misclassif. okay
  - $\rightarrow \lambda \to \infty \rightarrow$ recover original SVM as $\xi_i = 0$
- (use C-V)

Kernelised SVM: induce feature mapping using $\phi(\underline{x}_i) : \mathbb{R}^d \to \mathbb{R}^D$

Soft margin kernelised SVM (primal)

$\min_{\underline{w}, w_0, \xi_1,\dots,\xi_n} \frac{1}{2}\|\underline{w}\|_2^2 + \lambda \sum_{i=1}^N \xi_i$ s.t. $y_i(\phi(\underline{x}_i)^T \underline{w} + w_0) \geq 1 - \xi_i \;\forall i = 1,\dots,N$

$\xi_i \geq 0 \;\forall i = 1,\dots,N$

BREATHE - $\frac{1}{2}$ way through

Slack variable kernelised SVM (Lagrangian)

- Maximise over each (α_i, μ_i) and minimise over $\underline{w}, w_0, \xi_1, \xi_2, \ldots, \xi_N$ (?) (*) (?)
- Define N $\alpha_i > 0$, $\mu_i > 0$ $\forall i = 1, \ldots, N$ (L.M.)
- $L_p(\underline{w}, w_0, \xi_1, \ldots, \xi_N, \underline{\alpha}, \underline{\mu}) = \frac{1}{2}\|\underline{w}\|_2^2 + \lambda \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i(\phi(\underline{x}_i)^T \underline{w} + w_0) - 1 + \xi_i) - \sum_{i=1}^{N} \mu_i \xi_i$

(?) s.t. $\alpha_i \geq 0$ $\mu_i \geq 0$ $y_i(\phi(\underline{x}_i)^T \underline{w} + w_0) - 1 + \xi_i \geq 0$ (non-negativity constr.)

Minimise over $\underline{w}, w_0$, each ξ_i:

$\underline{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\underline{x}_i)$ (as before) $\quad \sum_{i=1}^{N} \alpha_i y_i$ (as before) $\quad \lambda - \alpha_i - \mu_i = 0$ (II)

(I): $\underline{w} = \sum_{i=1}^{N} \alpha_i y_i \underline{x}_i$ $\quad$ (II) $\mu_i = \lambda - \alpha_i$ $\quad \rightarrow L_p(\underline{w}, w_0, \xi_1, \ldots, \xi_N, \underline{\alpha}, \underline{\mu})$

(I) $\nabla_{\underline{w}} L$ $\quad \frac{\partial L}{\partial w_0}$ $\quad \frac{\partial L}{\partial \xi_i}$

Slack variable kernelised SVM (dual):

- Solve: $\max_{(\alpha_1 \ldots \alpha_n)} L_D(\underline{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{\phi(\underline{x}_i)^T \phi(\underline{x}_j)}_{K(\underline{x}_i, \underline{x}_j)}$ s.t. $\sum_{i=1}^{N} \alpha_i y_i = 0$ $\quad 0 \leq \alpha_i \leq \lambda$ (*)
- Dual is exactly same as hard-margin SVM, with addition of $\phi(\underline{x}_i)^T \phi(\underline{x}_j)$
- Estimate $\hat{\alpha}_i$s via iterative algo.

ii) $0 \leq \alpha_i \leq \lambda$ condition
iii) Replace $\underline{x}_i$ with $\phi(\underline{x}_i)$

Dual solution → primal: (I) $\hat{\underline{w}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \phi(\underline{x}_i)$

Classification for (Hard-margin SVM): (OTI) $\hat{\underline{w}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \underline{x}_i$, $\hat{w}_0$; declare new $\underline{x}_0$, $\hat{y}_0 = \text{sign}(\underline{x}_0^T \hat{\underline{w}} + \hat{w}_0) = \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i \underline{x}_0^T \underline{x}_i + \hat{w}_0\right)$

Classification for (Soft-margin Kernelised SVM): (OTI) $\hat{\underline{w}} = \sum_{i=1}^{N} \hat{\alpha}_i y_i \phi(\underline{x}_i)$, $\hat{w}_0$; declare new y_0 :-

$$y_0 = \text{sign}\left(\phi(\underline{x}_0)^T \hat{\underline{w}} + \hat{w}_0\right) = \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i \phi(\underline{x}_0)^T \phi(\underline{x}_i) + \hat{w}_0\right) = \text{sign}\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i K(\underline{x}_0, \underline{x}_i) + \hat{w}_0\right)$$

Summary: Basic SVM - linear classifier for linearly separable data
- affine hyperplane position maximises the margin
- convex dual → exact global solution via optima.

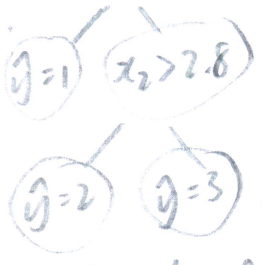
Fully-fledged SVM:-

Ing.	Purpose
nex margin	good generalisation
slack variables	overlapping classes, robust vs outliers
kernel	non-linear decision boundary

Hyperparam: λ and in RBF - X-V

L12-Regequations

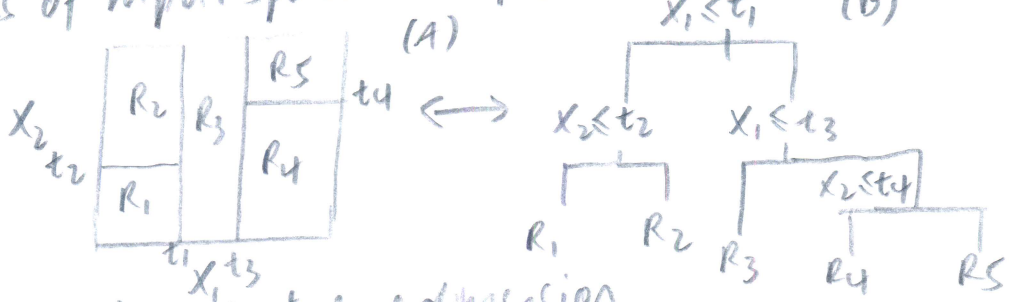
decision trees: · A decision tree maps input $\underline{x} \in \mathbb{R}^d$ to output y using binary decision rules (if, else)
· output $y \rightarrow y \in \mathbb{R}$ (regression), $y \in \{-1, +1\}$ binary class., $y \in \{1, \dots, K\}$
· nodes - contain splitting rules
· leaf node - associated with output
· splitting rule: $h(\underline{x}) = \mathbb{I}(x_j > t)$ for some j - dim. of $\underline{x}$, t - threshold $\in \mathbb{R}$
· predictions: use transition rules to navigate path to leaf node



decision stump: one-level tree (tree with one splitting rule)

Regression trees: · Amounts to a partitioning of an input space $X \in \mathbb{R}^d$ so that data in same region → same prediction
· only certain kinds of input space allow partitioning via a recursive splitting rule

· **Regression trees:** (3 representations)
- X_1, X_2;



· can amend B to include output y as dimension
· RTs. estimate a step-function approximation to data

classification trees · $\underline{x} \in \mathbb{R}^2$ $y \in \{1, 2, 3\}$ x_1 - sepal length/width x_2 - petal length/width

(2 representations)

decision tree algorithm: (task independent) for class. or reg.
1. start with leaf node containing all data
2. loop through:
   a. Pick the leaf to split that reduces uncertainty most
   b. specify decision rule on one of dimensions
3. stopping rule (input)

growing regression trees: · M regions of space, $R_1, \dots, R_M$; model response as constant c_m in each region
· prediction function $f(\underline{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\underline{x} \in R_m)$
· Given fixed M, we have to estimate c_m and R_m
· objective function: minimise $\sum_i (y_i - f(\underline{x}_i))^2$ (RSS)

· 1. Find c_m given R_m: RSS minimised
$\hat{c}_m = \frac{\sum_{i=1}^{N} y_i \mathbb{I}(\underline{x}_i \in R_m)}{\sum_{i=1}^{N} \mathbb{I}(\underline{x}_i \in R_m)}$ - optimal c_m is average of outputs y_i for which $\underline{x}_i \in R_m$ (i.e. average of y_i in region R_m)
$\hat{c}_m = \text{ave}(y_i | \underline{x}_i \in R_m)$

· 2. Finding regions R_m: consider splitting R at values s at dim j i.e. (s, j) over all existing regions $R_1, \dots, R_m$

- define $R^-(j,s) = \{x_i \in R \mid x_i(j) \leq s\}$ $R^+(j,s) = \{x_i \in R \mid x_i(j) > s\}$ (*)
- For all M regions, evaluate $R_m^-(j=\bar{j}, s=\bar{s})$ and $R_m^+(j=\bar{j}, s=\bar{s})$ for all possible values of ($\bar{j}$ and $\bar{s}$) (candidate m, j, s
- M regions are fixed in advance
- update proposed predictions $\hat{c}_m$ using new regions $R_m^-(j=\bar{j}, s=\bar{s})$ and $R_m^+(j=\bar{j}, s=\bar{s})$ for all $m, \bar{j}$ and $\bar{s}$ (brute-force)
- This yields a change in $RSS = \sum_i (y_i - f(x_i))^2$ via $f(x_i) = \sum_{m=1}^{M} c_m \mathbb{I}(x_i \in R_m)$
- For every proposed $(\bar{j}, \bar{s})$ we get a new f and change in RSS; select $(\bar{j}, \bar{s})$ that reduces RSS the most

Growing classification trees: - Require a measure of how badly region classifies data and how much it can improve if split.

K-class classification: $y \in \{1, \dots, K\}$

For all $x \in R_m$ let p_k be empirical fraction labelled k in region R_m (i.e. construct empirical distribution of labels y for data in that region.

Prediction quality of R_m:

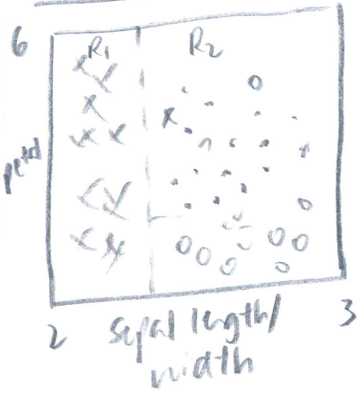
- Prediction $\hat{y}$ is most prevalent label $k(R_m)$

1. Classification error: $1 - \max_k p_k$ (MIS)
2. Gini index: $1 - \sum_k p_k^2$
3. entropy: $-\sum_k p_k \ln p_k$

} max. when p_k is uniform over K-classes in R_m, min when $p_k = 1$ for some k (R_m has one class)

(*)

classification example:



- these are measures that need to be evaluated for all regions
- splitting decision: which region R_m, dimension j, thresholds
- Each (j,s) for a region R_m yields R_m^- and R_m^+ (2 new regions) over which quality measure $v(R_m)$ can be evaluated
- $R_1 \to v(R_1) = 0$ $p_k = 1$ for class 1 in R_1
- $R_2 \to v(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2 = 0.5098$
- Select R_2 to split as $v(R_2)$ is highest

Quality measure improvement: $v(R_m) - \left(p_{R_m^-} \cdot v(R_m^-) + p_{R_m^+} \cdot v(R_m^+)\right)$ (*)

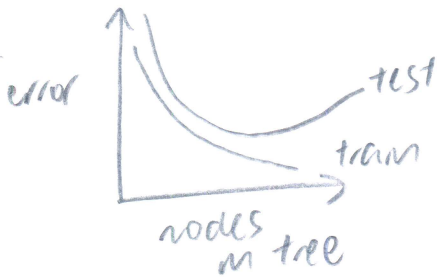
(From choosing to split $R_m \to R_m^-$ and R_m^+ via a choice of (j,s))

- $v(R_m^+), v(R_m^-)$ - new quality measure over new regions R_m^+ and R_m^-
- $p_{R_m^+}$ $p_{R_m^-}$ - fraction of data in R_m split into R_m^+ and R_m^-

(?) is $-\left(p_{R_m^-} \cdot v(R_m^-) + p_{R_m^+} \cdot v(R_m^+)\right)$ the 'reduction in uncertainty' we have to maximise for (j,s)?

- **Classification procedure:** 1. Evaluate $u(R_m)\ \forall m=1,\dots,M$, select the ~~big~~ region R_m with highest $u(R_m)$ (it classifies the worst) to partition.
2. From all possible (j,t), each defines an $R_{(m)}^-(j,s)$ and $R_{(m)}^+(j,t)$
(*) (*)
3. select (j,t) to maximise $\left(p_{R_m^-}\cdot u(R_m^-) + p_{R_m^+}\cdot u(R_m^+)\right)$ (i.e. reduce object. most)

- **Stopping rules:** Heuristic example about uncertainty reduction reservations
- **Pruning:** grow a tree then trim (see Hastie et al.)
- **overfitting:**



- test. usually overfitting in node no.
- train - decreasing in nodes.

Bootstrap: statistical technique. ensemble classifiers / resampling method

Resampling: Sampling from an empirical distribution of data

- In context of ensemble methods, creates many mediocre classifiers using bootstrap/data subsamples, then bagged to improve performance.

Bootstrap algorithm: - Example ~~goal~~ from thinking of statistic as median; trying to find its variance $S = \text{med}(\cdot)$

IN: A sample $(x_1,\dots,x_n)$ from an unknown underlying distn P.
Estimation rule / estimator $\hat{S}$ of a statistic S

1. Generate B bootstrap samples $B_1,\dots,B_B$ by randomly picking points n times from $(x_1,\dots,x_n)$ without replacement, x_i can appear multiple times in each B_b or be omitted.
2. Evaluate estimator $\hat{S}$ on each B_b as if it were the dataset for every $b=1,\dots,B$
$\hat{S}_b := \hat{S}(B_b)$ (generate B lots of these)
3. Estimate mean and covariance of estimator $\hat{S}$ by 'averaging' over B bootstrap estimators $\{\hat{S}_1,\dots,\hat{S}_B\}$ to find:-

$$\hat{\mu}_{BOOT} = \frac{1}{B}\sum_{b=1}^{B}\hat{S}_b \qquad \sigma^2_{BOOT} = \frac{1}{B}\sum_{b=1}^{N}(\hat{S}_b - \hat{\mu}_{BOOT})^2$$

Bagging: - Bootstrap (sample aggregation)

Bagging example: (Bagging trees)
- $\underline{x}\in\mathbb{R}^5$ $y\in\{0,1\}$
- Trees created according to bootstrap sample $B_1,\dots,B_{11}$
- Variation amongst trees → estimated from resampled data
- Tree can be mediocre; improves perf. in pres. of non-linearity

Bagging algorithm:

For $b=1,\dots,B$

1. Draw bootstrap sample B_b of size n from training data consisting of (x_i, y_i)
2. Train classifier/regression function f_b on bootstrap sample B_b

- For new point x_0, compute

$$f_{avg}(x_0) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_0)$$

- Reg: $f_{avg}(x_0)$ is prediction, averaged over all B bootstrapped samples
- class: $f_{avg}(x_0)$ is an average over B votes (majority vote)

Bagging: (Analysis, critique)
- Bagging works via bias-variance trade-off (lowering variance at cost of increasing bias)
- Bagged trees are correlated (in what?) (?)
- correlation of bootstrap samples $B_b \Rightarrow$ bagging benefits ↓ (?)(*)

Random Forests: (Modification)
- Modify bagging on trees to reduce correlation
- Learn decision tree on each bootstrap data set B_b
- (OR) Randomly select a subset size m of dimensions of $x \in \mathbb{R}^d$ to split for each bootstrap sample B_b. (*)

Random Forests (Algorithm):

Input param: m; $m \in \mathbb{Z}$ $m < d$ often $m \approx \sqrt{d}$

For $b=1,\dots,B$:-

1. Draw a bootstrap sample B_b of size n from training
2. Train a tree classifier on B_b, each split computed as:-
   i) Randomly select m dimensions of $x \in \mathbb{R}^d$, newly chosen for each b
   ii) make best split $(\hat{j}, \hat{s})$ restricted to that subset

- RF classification with few hundred trees; trees predict orange/blue; majority vote of pred ⇒ complex, non-linear decision boundary

L13 - Key equations:

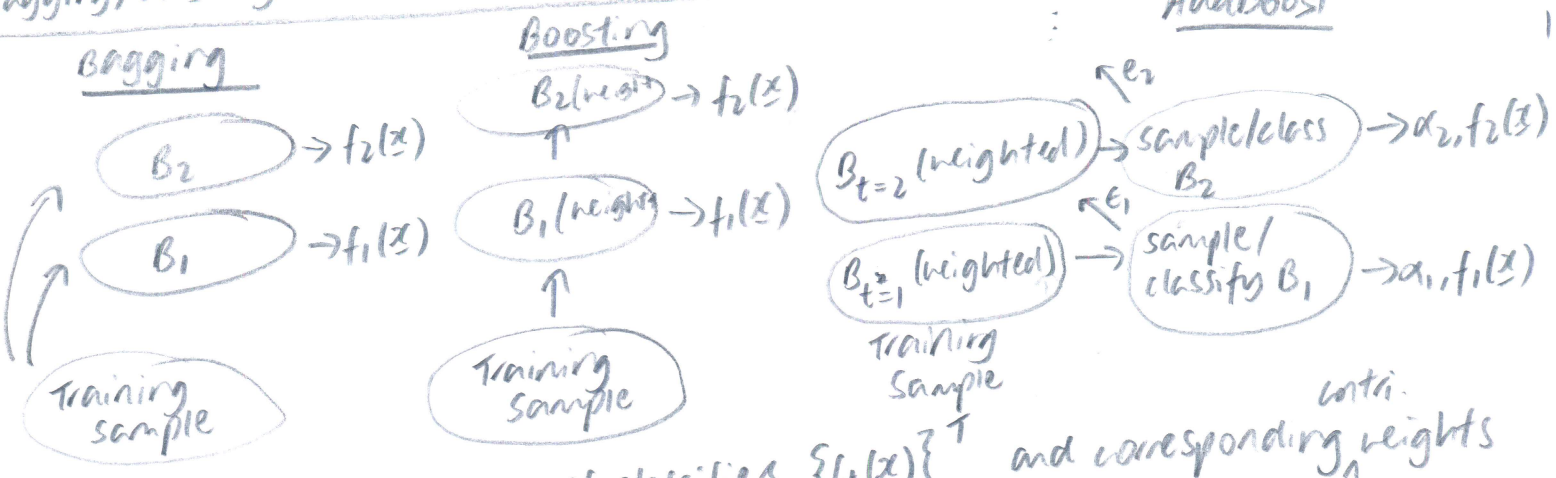
Bagging binary classifiers:

- Given $\{(x_i, y_i)\}_{i=1}^{N}$ $x \in \mathcal{X}$ $y \in \{-1, +1\}$
- Generate B bootstrap samples $B_1, \dots, B_B$
- Estimate B lots of classifier functions f_b using each B_b
- Classification rule: $f_{bag}(x_0) = \text{sign}\left(\sum_{b=1}^{B} f_b(x_0)\right)$

- A committee of classifiers votes on a label, with each classifier estimated on a bootstrap sample
- This is an example of an ensemble method

Boosting: · ensemble learning method
· some similarities to bagging, except ensemble is a sequentially created collection of classifiers indexed by iteration no.
· usually use weak classifiers (i.e. better than random guessing)

Bagging, boosting, AdaBoost illustration:



AdaBoost: · Outputs a sequence of classifiers $\{f_t(\underline{x})\}_{t=1}^{T}$ and corresponding contri. weights $\{\alpha_t\}_{t=1}^{T}$ indexed by iteration no.
· Resulting boosted classifier f_{BOOST} predicts $\underline{x}_0$ by majority weighted vote over a weighted sum of predictions over all iterations
· i.e. $f_{BOOST}(\underline{x}_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(\underline{x}_0)\right)$

AdaBoost Algorithm: (binary classif.)

IN: $\{(\underline{x}_i, y_i)\}_{i=1}^{N}$, $\underline{x} \in X$, $y \in \{-1, +1\}$ $\underline{w} \in \mathbb{R}^N$ (N-dim weight vector) $\underline{w}_t = \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix}$ (initial)

· set $w_1(i) = \frac{1}{n}$ $\forall i = 1, \dots, N$ (i.e. set $\underline{w}_0$ as uniform p.d.) (initial)

Draw B_t according to $\underline{w}_t$

· For $t = 1, \dots, T$
1. Sample a bootstrap dataset B_t of size N according to distribution $\underline{w}_t$ (select $(\underline{x}_i, y_i)$ with probability $w_t(i)$ each round
2. Estimate a classifier f_t using data in B_t — Est. f_t using B_t
3. Set $\epsilon_t = \sum_{i=1}^{N} w_t(i) \mathbb{I}(y_i \neq f_t(\underline{x}_i))$ and $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ — Update ϵ_t, α_t
4. Scale $\hat{w}_{t+1}(i) = w_t(i) e^{-\alpha_t y_i f_t(\underline{x}_i)}$ and set $w_{t+1}(i) = \frac{\hat{w}_{t+1}(i)}{\sum_j \hat{w}_{t+1}(j)}$ — update w_{t+1} / i) update weight renorm.

· Classif. rule: $f_{BOOST}(\underline{x}_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(\underline{x}_0)\right)$ (majority weighted)

Comments: · $\underline{w}_t$ - a probability distribution over training obs. $(\underline{x}_i, y_i)$
· $w_t(i) = \frac{1}{n}$ $\forall i = 1, \dots, N$ → initialise a uniform probability distri. weighted
· ϵ_t - probability of error at iteration t according to classifier f_t est on B_t
→ OR sum of probability weights $w_t(i)$ on misclassified data at iteration t.

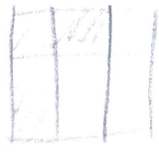
Upweighting / downweighting

- At each iteration round, misclassified points → upweighting in next round
  correctly classified → downweighting
- Difficult to classify points → increasing influence, each classifier forced to concentrate on training obs. missed by previous ones in sequence

Formally: Focus on $w_{t+1}(i) = w_t(i)\, e^{-\alpha_t y_i f_t(x_i)}$ $\quad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ $\epsilon_t = \sum_{i=1}^{N} w_t(i)\, \mathbb{I}(y_i \neq f_t(x_i))$

- weak classifier $\Rightarrow \epsilon_t < \frac{1}{2}$ (better than coin flip)
- $\epsilon_t < \frac{1}{2} \Rightarrow \alpha_t > 0$ · correctly classified: $\text{sign}(y_i) = \text{sign}(f_t(x_i))$: scale by $e^{-\alpha_t}$
  · misclassified : $\text{sign}(y_i) \neq \text{sign}(f_t(x_i))$: scale by e^{α_t}
- for all $\alpha_t > 0$, $e^{\alpha_t} > e^{-\alpha_t}$: correct classification scales $w_t(i)$ by small $e^{-\alpha_t}$
  incorrect —————— " —————— by large e^{α_t}

class example #1: classifier after 3 rounds:



$= 0.42 \cdot \hat{f}_1 + 0.65 \cdot (\hat{\alpha}_2, \hat{f}_2) + 0.92 \cdot (\hat{\alpha}_3, \hat{f}_3)$

Empirical results :- 1) test error vs iterations

test err_{rnd} (50%) > test $\text{err}_{\text{stump}}$ (45.8) > test $\text{err}_{\text{dec.tree}}$ (25.7%) > test $\text{err}_{\text{400 it boosted stump}}$ (5.8%)

2) Algorithm error-rate on datasets w and w/o boost
stumps vs C4.5 → 45° line (above) ⇒ boosting improves error
(i) Boosting bad classifier better than not boosting good classifier
(ii) Boosting good class better, but how to trade off vs efficiency

Boosting as a feature mapping: classifying new x_0 from boosting:-

$f_{\text{Boost}}(x_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0)\right)$ · define $\phi(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_T(x) \end{bmatrix}$ $\phi \in \mathbb{R}^T$ $\quad f_t(x) \in \{-1, +1\}$

- If we define ϕ as a feature mapping, and α as a hyperplane (both stacked vectors)
- $f_{\text{Boost}}(x_0) = \text{sign}\left(\phi(x_0)^T \alpha\right)$ · Adaptively gen. classifier - feature map
  · Hyperplane - linear decision boundary in $\mathbb{R}^T$

class example #2: (Viola & Jones '01)
- Face detection
- Pre-process via specialised feature extractors
- classify patch as face vs no face via boosted decision stump

AdaBoost - selects features and trains classifier.

Boosting training error theorem: (Freund & Schapire (2001)) under AdaBoost, if ϵ_t is weighted error of classifier at tth for $f_{\text{Boost}}(x_0) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x_0)\right)$

$\text{train error} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i \neq f_{\text{Boost}}(x_i)) \leq \exp\left(-2\sum_{t=1}^{T}\left(\frac{1}{2} - \epsilon_t\right)^2\right)$

- If each ϵ_t is only slightly better than guessing i.e. $\epsilon_t = 0.45$
- $\epsilon_t = 0.45$, $T = 1000$ → training error ≤ 0.0067

Boosting training error: (Proof) $\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i \neq f_{Boost}(x_i)) \leq \prod_{t=1}^{T} Z_t \leq \exp\left(-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2\right)$

where $Z_t = \sum_j \hat{w}_{t+1}(j)$ (normalisation constant) ← weight

- It is a theoretical guarantee that places bounds on training error

Boosting-overfitting? - Sometimes, but not in general
- violates usual statistical intuitions about overfitting - test error continues to decrease after many rounds
- still no convincing theoretical explanation for AdaBoost's empirical properties
- see diagram → AdaBoost test and training error decreases after 100 rounds

UU-Key equations:

- Recap of SL, UL

Supervised learning: given $\{(\underline{x}_i, y_i)\}_{i=1}^{N}$; estimate function $f(\underline{x})$ that predicts $y_i = f(\underline{x}_i)$ on this data
- use function to make predictions for you on new data $\underline{x}_0$

SL (probabilistic): $(\underline{x}, y)$ is an r.v with joint distri $p(\underline{x}, y)$
- SL estimates a condit. distri $p(y|\underline{x})$ {directly (discriminative e.g. logistic); indirectly (generative) e.g. Bayes class $y = \underset{k}{\text{argmax}}\, p(y=k)\,p(\underline{x}|y=k)$
- Joint dist $p(\underline{x}, y) = p(y|\underline{x})\,p(\underline{x})$

UL (probabilistic): - UL focuses on estimation of $p(\underline{x})$

UL types (loose): 1. clustering 2. matrix fact. 3. sequential models

Clustering: - given data $\{\underline{x}_i\}_{i=1}^{N}$, partition into clusters
- find clusters given data and a modelling assumption
- observ. in same cluste → similar; define sim and dim., set no. of clusters

cluster assignment (representation): - for K clusters, encode clusters as an indicator c
- $c \in \{1, \dots, K\}$ · $c_i = k \Rightarrow \underline{x}_i$ assigned to cluste k
- similar to classification with y_i (ground truth label), except no grd truth
- c is a latent/auxiliary variable indexing cluste $\{1, \dots, K\}$ observ $\underline{x}_i$ belongs to
- Each cluste has 'label', but we dont know labels and clusters.

K-means algorithm: (use Dynamic diagram to help intuition)

IN: $\{\underline{x}_i\}_{i=1}^{N}$ $\underline{x}_i \in \mathbb{R}^d$

GOAL: Minimise $\mathcal{L} = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{I}\{c_i = k\}\,\|\underline{x}_i - \underline{\mu}_k\|_2^2$ and output $\underline{c}^*$ (cluste assignment vector), $\underline{\mu}_k^*$ $\forall k = 1, \dots, K$ (centroids)

- Randomly init $\underline{\mu} = (\underline{\mu}_1, \dots, \underline{\mu}_K)$
- Iterate until $\underline{c}$ and $\underline{\mu}$ stop changing:-
  1. update each c_i: $c_i = \underset{k}{\text{argmin}}\, \|\underline{x}_i - \underline{\mu}_k\|_2^2$
  2. update each $\underline{\mu}_k$: $\underline{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{N} \underline{x}_i\, \mathbb{I}\{c_i = k\}$ $\quad n_k = \sum_{i=1}^{N} \mathbb{I}\{c_i = k\}$

K-means $\underline{c} = [c_1, \dots, c_N]$ - vector of cluster assignments over N data points

- $c_i = c_j = k \Rightarrow \underline{x}_i$ and $\underline{x}_j$ clustered together in cluster k

- $\underline{\mu} = (\underline{\mu}_1, \dots, \underline{\mu}_K)$ $\underline{\mu}_k \in \mathbb{R}^d$ - each $\underline{\mu}_k$ is a centroid

K-means

- objective: $\mathcal{L} = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}\{c_i = k\} \|\underline{x}_i - \underline{\mu}_k\|_2^2 = \sum_{k=1}^{K} \sum_{i:c_i=k} \|\underline{x}_i - \underline{\mu}_k\|_2^2$

- K-means objective is non-convex; cannot find global optima $\underline{\mu}^*$ and $\underline{c}^*$; only local optima

radient descent as an iterative method:

G.D.: $\underline{w} \leftarrow \underline{w} - \eta \nabla_{\underline{w}} \mathcal{L}$ (update '$\underline{w}$' by moving in direction that decreases obj.)

- nonconvex obj → will not move to best value
- η too large → may not move to better
- (!) $\underline{w}$ is continuous, $\underline{c}$ is discrete.

o-ordinate descent: widely used in U.L

n context of K-means)

- minimise $\mathcal{L}$ over $\underline{c} = \{c_1, \dots, c_N\}$ and $\underline{\mu} = \{\underline{\mu}_1, \dots, \underline{\mu}_K\}$
- update c_i: $c_i = \underset{k}{\operatorname{argmin}} \|\underline{x}_i - \underline{\mu}_k\|_2^2$ with $\underline{\mu}$ fixed
- update $\underline{\mu}_k$ $\forall k$: $\underline{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{N} \underline{x}_i \mathbb{I}\{c_i = k\}$ with $\underline{c} = \{\bar{c}_1, \bar{c}_2, \dots, \bar{c}_N\}$ fixed

until equil.

ssignment uparting: $\mathcal{L} = \left(\sum_{k=1}^{K} \mathbb{I}\{c_1 = k\} \|\underline{x}_1 - \underline{\mu}_k\|_2^2\right) + \dots + \left(\sum_{k=1}^{K} \mathbb{I}\{c_N = k\} \|\underline{x}_N - \underline{\mu}_k\|_2^2\right)$ - Assi

ach bracket is indepedent → N indepedent optimisations over c_is given fixed $\underline{\mu}$

$\mathcal{L} = \left(\sum_{i=1}^{N} \mathbb{I}\{c_i = 1\} \|\underline{x}_i - \underline{\mu}_1\|_2^2\right) + \dots + \left(\sum_{i=1}^{N} \mathbb{I}\{c_i = K\} \|\underline{x}_i - \underline{\mu}_K\|_2^2\right)$

ch bracket indepedent → K optimisation problems over $\underline{\mu}_k$s given fixed $\underline{c}$

ignment ($\underline{c}$): - Assign $\underline{x}_i$ to closest centroid

olution - update centroid $\underline{\mu}_k$ $\underline{\mu}_k = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbb{I}\{c_i = k\} \|\underline{x}_i - \mu\|_2^2$

(Average over all data assigned to kth centroid)

egence of K-means:

ate c_i or $\underline{\mu}_k$ decreases $\mathcal{L}$

monotonically decreasing

d $\underline{\mu}$ stop changing; algorithm converged to local optimum

l convergence → local optima due to non-convexity of $\mathcal{L}$

-convexity ⇒ different initialis., different results

with different initialisations, select result with lowest $\mathcal{L}$ (scikit learn)

ing K:

am knowledge - Part of larger app

tive decreases in $\mathcal{L}$ - Advanced modelling - Bayesian non parametric methods

ations of K-means: 1) Lossy data compression; vectorise patches, cluster, replace patch with centroid (prototypes)

2) Discretisation for pre-processing