Linear Regression:

- Overfitting, under specified
- $\vec{w} = (X^T X)^{-1} X^T \vec{y}$ (RR: $(X^T X + \lambda I_n)^{-1} X^T \vec{y} = \vec{w}$)
- Primal form: $n >> m$

$$f(\vec{x}) = \vec{w}^T \vec{x}$$

- Dual form: $m >> n$ (sparsity benefits)

Kernel RR: $f(\vec{x}) = \sum_i \alpha_i \vec{x}_i^T \vec{x}$ $\left(\vec{w} = \sum_i \alpha_i \vec{x}_i, \ \alpha_i = \frac{1}{\lambda}(y_i - \vec{w}^T \vec{x}_i), \ \vec{\alpha} = (XX^T + \lambda I_n)^{-1} \vec{y}\right)$

Kernels:

- $K(x,t) = \langle \phi(x), \phi(t) \rangle$, $K(x,x) \geq 0$ (PSD function)
- Types:

Polynomial	$(1 + \langle x, t \rangle)^d$	$\phi(\vec{x}) = (1, \vec{x}, \vec{x}^2, x^3, \dots)^T$
Anova	$\prod_i (1 + x_i t_i)$	$\phi(\vec{x}) = (1, x_1, x_n, x_1 x_2, \dots, x_1 x_2 \cdots x_n)^T$
Gaussian	$\exp[-\beta \|x - t\|^2]$	$\phi(x) = e^{-\beta \|x\|^2} \left(\sqrt{\frac{2\beta}{1!}} x, \sqrt{\frac{(2\beta)^2}{2!}} x^2, \dots\right)^T$
min	$\min(x, t)$	$\phi(x) = \min(x, \cdot)$

$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ →

- If function expressed as only dual inner products, can do kernel trick (eg Dual form)

Online Learning:

Proof for Bounds

- Halving algorithm, $M \leq \log_2 n$ if consistent expert
- Weighted majority, $w_{t+1,i} \leftarrow w_{t,i} \beta^{|x_{t,i} - y_t|}$, $\beta \to 0$, $\beta \to 1$, $M \leq \min_i M_i + \log n$
- Perceptron, $\vec{w}_{t+1} \leftarrow \vec{w}_t + y_t \vec{x}_t$ on mistake

$(-1, 1) \to \hat{y}_t = (\vec{w}_t^T \vec{x}_t)$ sign

$(0, 1) \to \hat{y} = \text{sign}(\vec{w}_t^T \vec{x}_t - n)$

$M \leq \left(\frac{R}{\gamma}\right)^2$, $R = \max_t \|\vec{x}_t\|$, γ = margin

$(M_t \gamma)^2 \leq \|\vec{w}_{M+1}\|^2 \leq M_t R^2$ (induction; claim $\langle \vec{w}_t, \vec{v} \rangle \geq M_t \gamma$; induction)

- Winnow, $w_{t+1,i} \leftarrow w_{t,i} \times 2^{(y_t - g_t)\vec{x}_{t,i}}$
- DNF, $\phi(\vec{x}): \mathbb{B}^n \to \mathbb{B}^{2^n}$, ANOVA Kernel

$M \leq 3k(\log(n) + 1) + 2$

SVM:

- Lagrange multipliers, slack variables

Gaussian Processes:

- Prior, Noise, Posterior
- Error bars
- Evidence

$$P(\vec{w} | \vec{x}, y) \propto P(\vec{x}, y | \vec{w}) \cdot P(\vec{w})$$

$N(\langle \vec{w}, \phi(\vec{x}) \rangle, \sigma^2)$, $N(\vec{0}, I)$

$N(\vec{w}_{MAP}, \Sigma)$

$\vec{w}_{map} = X^T (K + \sigma^2 I)^{-1} \vec{y}$

$\Sigma^{-1} = \frac{1}{\sigma^2} (X^T X + \sigma^2 I)$

$$\log P(\vec{y} | S) = -\frac{1}{2} \vec{y}^T (K + \sigma^2 I) - \frac{1}{2} \log \det (K + \sigma^2 I) - \frac{m}{2} \log 2\pi$$

Trees & Ensembles:

- CART, grow & prune
- Weak learners, h
- Bagging & Random forest
- Boosting $D_{t+1}(i) \propto D_t(i)\, e^{-\alpha_t y_i h_t(\vec{x}_i)}$
- Training error $\frac{1}{m}\sum I(H(\vec{x}_i) \neq y_i) \leq \frac{1}{m}\sum e^{-y_i g(\vec{x}_i)} = \prod_t Z_t$

Learning Theory:

- $err(f) = E_{(\vec{x},y)\in D}[V(f(\vec{x}), y))]$
- Generalisation $err(c) \leq \hat{err}(c) + \sqrt{\frac{C(H) + \log 1/\delta}{m}}$
- Structural risk minimization $err(c^*) \leq \hat{err}(c^*) + \sqrt{\frac{C(H) + \ln 1/\delta + \ln K}{m}}$
- $err(m, H, \delta) = \frac{2}{m}\left(d \ln \frac{2em}{d} + \ln 2/\delta\right)$
- Bias-Variance decomposition $err(f_S) - err(f^*) = \underbrace{err(f_S) - err(f_H)}_{\text{variance}} + \underbrace{err(f_H) - err(f^*}_{\text{bias}}$

COMPGI01 - Supervised Learning

1: Introduction

- SL model:
  - training set $S = (\vec{x}_i, y_i)$, $i = 1, \dots, m$ sampled iid from unknown, $P(\vec{x}, y)$ & fixed
  - find f_D s.t. $f_D(\vec{x}_i) \approx y_i$
- Least squares:

$$\hat{y}_i = \vec{w}^T \vec{x}_i, \text{ minimise } E(\vec{w}) = \sum_i (\hat{y}_i - y_i)^2$$

$$X = \begin{pmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_m^T \end{pmatrix}, \vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

$$\varepsilon(\vec{w}) = \frac{1}{m}(X\vec{w} - \vec{y})^T (X\vec{w} - \vec{y})$$

$$\frac{\partial E}{\partial \vec{w}} = 0 \Rightarrow X^T X \vec{w} = X^T \vec{y}$$

- can add bias by substituting $\vec{x}^T = (\vec{x}^T, 1)$, $\vec{w}^T = (\vec{w}^T, b)$

- k-NN:
  - $f(\vec{x})$ = class of majority of k nearest neighbours (by some dist metric, eg Euclidean)
  - ↑ k = smoother
  - asymptotically optimal:

$$P(Y = c | \vec{x}) = \frac{|\{i : y_i = c, i \in 1, \dots, m\}|}{k}$$

Optimal Bayes estimator, minimizer of $E((y - f(\vec{x}))^2)$, w/ $\vec{x}, y$ sampled from $P(\vec{x}, y)$

  - as $m \to \infty$

$$\varepsilon(k\text{-NN}) \to \varepsilon(f^*) \text{ if } k(m) \to \infty, \frac{k(m)}{m} \to 0$$

(FA convergence rate depends exponentially on x dimensionality)

- Square loss:

$$\varepsilon(f) = \sum_{x,y} (y - f(x))^2 P(x,y) = \sum_x \left[\sum_y (y - f(x))^2 P(y|x) \right] P(x)$$

$$\varepsilon(f(x')) \propto \sum_y (y - f(x))^2 P(y|x)$$

$$\frac{\partial}{\partial f(x')} \varepsilon(f(x')) = -2 \sum_y (y - f(x)) P(y|x') = 0$$

$$\Rightarrow f(x') = \sum_y y P(y|x')$$

$$\Rightarrow f^*(x) = \sum_y y P(y|x) = E(y|x) = F(x)$$

ie Bayes optimal estimator is ↙

- Bias & Variance:

$y = F(x) + \varepsilon$ (underlying + noise), learner $A(x)$

$\mathcal{E}(A(x')) = E((y' - A(x'))^2)$ [y' sampled from $P(Y|x')$]

only for square loss!

$= E(y' - f^*(x')^2) +$ ← Bayes error (irreducible noise)

$E(f^*(x') - E(A(x')))^2 +$ ← bias² (discrepancy between algorithm & truth)

$E((A(x') - E(A(x'))))^2$ ← variance (between training sets)

- No Free Lunch:
  - ~~usually~~ no algorithm does almost as well as Bayes estimator

- Bayes classifier:

$f^*(\vec{x}) = \text{argmax}_c P(Y = c|\vec{x})$

$\Rightarrow$ Bayes error rate $= \int (1 - P(Y = f^*(\vec{x})|\vec{x}))\, dP(\vec{x})$

- Hypothesis space:
  - approx $\mathcal{E}(f)$ w/ $\mathcal{E}_{emp}(f, S, f)$
  - if f can be anything, can always find f s.t. $\mathcal{E}_{emp}(S, f) = 0$
  - need to limit to hypothesis space $\mathcal{H}$, to reduce chance of overfitting

$f_S = \text{argmin}_{f \in \mathcal{H}} \mathcal{E}_{emp}(S, f)$

- Cross-validation:
  - method to approximate $\mathcal{E}(f)$; useful to choose $\mathcal{H}$/hyperparams

$S =$ | 1 | 2 | ···· | k-1 | k |

- train on k-1 folds, test on held out fold
- repeat k times & average errors

↑ k = better estimates, more expensive

2: Kernels & Regularization

- ill posed problems:
  - problem well posed if:
    - a solution exists
    - the solution is unique
    - solution depends continuously on data
  - learning problems generally ill posed; regularization provides framework to solve these

- Ridge Regression
  - add penalty term for more 'complex' hypothesis
  - minimize $\varepsilon_{emp_\lambda}(\vec{w}) = \sum_i (y_i - \vec{w}^T \vec{x}_i)^2 + \lambda \sum_i w_i^2$
  - $\nabla \varepsilon_{emp_\lambda}(\vec{w}) = 0 \Rightarrow -2X^T(\vec{y} - X\vec{w}) + 2\lambda\vec{w} = 0$

$$\vec{w} = (X^T X + \lambda I_n)^{-1} X^T \vec{y}$$

- Dual Representation
  - $f(\vec{x}) = \vec{w}^T \vec{x}$ ← primal form

$$\vec{w} = \frac{1}{\lambda}(X^T(\vec{y} - X\vec{w}))$$

$$\vec{w} = \sum_{i=1}^{m} \alpha_i \vec{x}_i \quad \left(\alpha_i = \frac{1}{\lambda}[y_i - \vec{w}^T \vec{x}_i]\right)$$

dual form $\longrightarrow f(\vec{x}) = \sum_i \alpha_i \vec{x}_i^T \vec{x}$

	Train time	Test/Prediction time	Good when
Primal form	$O(mn^2 + n^3)$	$O(n)$	
Dual form	$O(nm^2 + m^3)$	$O(mn)$	$m \ll n$

(m: # training points; n: dimensionality of x)

- If $\vec{x}$ is sparse (at most k non-zero components), $\vec{x}^T \vec{t}$ is $O(k)$
  - if $km \ll n$, dual representation $O(km^2 + m^3)$ training & $O(mk)$ predicting (if $k, m \ll n$)

- Feature maps:
  - Basis functions: explicit feature maps

$$\phi: \mathbb{R}^n \to \mathbb{R}^N, \quad \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \dots, \phi_N(\vec{x}))^T, \quad x \in \mathbb{R}^n$$

  - perform non-linear regression by transforming input into a higher dimensional space

primal representation →

$$f(\vec{x}) = \sum_{j=1}^{N} w_j \phi_j(\vec{x}) = \vec{w}^T \vec{\phi}(\vec{x})$$

  - $m << N \Rightarrow$ better to work w/ dual representation

  - Kernel functions: implicit feature map
    - in dual form, don't need ϕ explicitly; only need inner product between pairs of feature vectors

eg $k(x,t) = (x^T t)^2$
$\phi(x) = (x_1^2, x_2^2, x_1 x_2 \text{ or } x_2 x_1)$ →

$$k: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad k(\vec{x}, \vec{t}) = \langle \vec{\phi}(\vec{x}), \vec{\phi}(\vec{t}) \rangle, \quad \vec{x}, \vec{t} \in \mathbb{R}^n$$

$\Rightarrow$ computing $k(\vec{x}, \vec{t})$ independant of N!

- feature map of a given k is not unique

- $k: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is positive semi-definite if symmetric & matrix $(k(x_i, x_j): i,j = 1, \dots, k)$ is positive semi-definite for every $k \in \mathbb{N}$ & $x_1, \dots, x_k \in \mathbb{R}^n$

- k is PSD $\Leftrightarrow k(\vec{x}, \vec{t}) = \langle \vec{\phi}(\vec{x}), \vec{\phi}(\vec{t}) \rangle$, $x, t \in \mathbb{R}^n$ for some $\vec{\phi}: \mathbb{R}^n \to W$ & Hilbert space W

- Example Kernels:
  - Polynomial $k(x,t) = (1 + x^T t)^d$
  - Anova $k(x,t) = \prod (1 + x_i t_i)$
  - Gaussian $k(x,t) = \exp[-\beta \|x - t\|^2]$ ← ∞ dimensional corresponding feature map!
  - min $k(x,t) = \min(x,t)$ $(x, t \in [0, \infty])$

2: Kernels & Regularization

- Kernel construction:
  - K_1, K_2 are kernels, & $a>0$, A is PSD & symmetric, k is kernel on $\mathbb{R}^N$ & ϕ: is $\mathbb{R}^n \to \mathbb{R}^N$, the following are PSD kernels on $\mathbb{R}^N$:
    - $\vec{x}^T A \vec{t}$
    - $K_1(\vec{x},\vec{t}) + K_2(\vec{x},\vec{t})$
    - $a K_1(\vec{x},\vec{t})$
    - $K_1(\vec{x},\vec{t}) K_2(\vec{x},\vec{t})$
    - $k(\phi(\vec{x}), \phi(\vec{t}))$

  ] — so can have polynomial of kernels too

- Dual Representation:
  - Representer theorem: dual form of RR holds for other loss functions too

$$\varepsilon_{emp_\lambda}(\vec{w}) = \sum_i V(y_i, \langle \vec{w}, \vec{\phi}(\vec{x}_i)\rangle) + \lambda\langle \vec{w}, \vec{w}\rangle, \quad \lambda > 0$$

where $V: \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ is a loss function

$$\nabla \varepsilon_{emp_\lambda}(\vec{w}) = 0 \Rightarrow \sum_i V'(y_i, \langle \vec{w}, \vec{\phi}(\vec{x}_i)\rangle)\vec{\phi}(\vec{x}_i) = 2\lambda\vec{w}$$

$$\Rightarrow \vec{w} = \sum_i \alpha_i \vec{\phi}(\vec{x}_i) \quad \left(\alpha_i = \frac{1}{2\lambda} V'(y_i, \langle \vec{w}, \vec{\phi}(\vec{x}_i)\rangle)\right)$$

$$\Rightarrow f(\vec{x}) = \langle \vec{w}, \phi(\vec{x})\rangle$$

$$= \sum_i \alpha_i \langle \vec{\phi}(\vec{x}_i), \vec{\phi}(\vec{x})\rangle$$

kernel trick

$$f(\vec{x}) = \sum_i \alpha_i K(\vec{x}, \vec{x}_i)$$

1/2

3: Online Learning

- Model: online sequence of data (usually no distributional assumptions)
  - sequentially predict & update classifier, aiming to minimize cumulative loss
  - + fast, low memory, no stat assumptions
  - − non-asymptotic, no stat assumptions

- Learning with Expert Advice:
  - each expert makes a prediction, goal is to combine predictions $\vec{x}_t$ to predict $\hat{y}_t$, estimate of y_t
  - Halving algorithm:
    - every time step, predict w/ majority of remaining experts
    - remove incorrect experts
    - on mistake, at least half experts removed; $\leq \log_2 n$ mistakes (if a perfect expert exists)
      - if no consistent expert, $L_A(S) \leq \alpha \cdot \min_i L_i(S) + \beta \log(n$
        α, β constants
        $L_i(S)$ loss of i^{th} expert
  - Weighted majority algorithm:
    - every expert has a weight
    - predict w/ weighted majority
    - wrong expert weights multiplied by $\beta \in [0,1)$
    - Mistakes bound:

$$W_{minority} \leq \frac{1}{2} W_t, \quad W_{majority} \geq \frac{1}{2} W_t$$

- if no mistake, minority $\times \beta$,
  $W_{t+1} \leq 1 W_t$
- if mistake, majority $\times \beta$,
  $W_{t+1} \leq (1+\beta)\frac{1}{2} W_t$

$$\Rightarrow W_{m+1} \leq \left(\frac{1+\beta}{2}\right)^m W_1 = \sum_{experts} W_{m+1}^{expert} = \sum_{experts} \beta^{M_{expert}} \geq \beta^{M_i}$$

$$\Rightarrow \left(\frac{1+\beta}{2}\right)^M W_1 \geq \beta^{M_i} \Rightarrow M \leq \alpha \cdot \min_i M_i + \beta b \cdot \log n$$

ie comparable to best expert

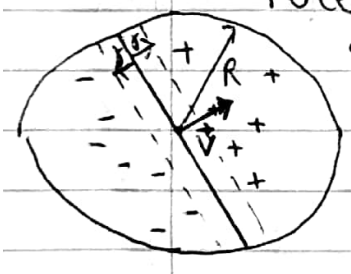
- Learning with thresholded linear combinations
  - algorithm predicts $\hat{y}_t$ w/ using $\vec{x}_t$ & past experience, & loss $L_A(S)$ on whole sequence
  - compare performance w/ $\mathcal{U} = \{\vec{u}\}$, ie hypothesis space, here $\vec{u}$ is a linear threshold function
  - assume $\exists \vec{u} . \text{Loss}_{\vec{u}}(S) = 0$

linear threshold function $\longrightarrow f_{\vec{u},b}(\vec{x}) = \text{sign}(\vec{u}\vec{x} + b)$

- Naive solution:
  - Weighted Majority over all possible disjunctions
  - n literals, k in each disjunction $\Rightarrow \binom{n}{k}$ weights; exponential in time & space!

variables → / experts	E_1	E_2	E_3	E_4	y	$E_1 \vee E_2$	$E_1 \vee E_3$ ...
	1	0	0	0	1	1	1
	0	1	1	0	0	1	1
	0	1	0	0	1	1	0
						2 mistakes	3 mistakes

- Perceptron:
  - assume data linearly separable by margin γ, s.t. hyperplane exists w/ normal $\vec{v}$ where $\|\vec{v}\| = 1$



$$\forall t . y_t \in \{-1, 1\}, \forall \vec{x}_t . \|\vec{x}_t\| \leq R,$$
$$\forall (\vec{x}_t, y_t) . y_t (\vec{x}_t \cdot \vec{v}) \geq \gamma$$

- $\vec{w}_1 = \vec{0}$ $M_1 = 0$
- for $t = 1 \ldots m$:
  - $\vec{x}_t \in \mathbb{R}^n$
  - $\hat{y}_t = \text{sign}(\vec{w}_t \cdot \vec{x}_t)$
  - if $\hat{y}_t \neq y_t$:
    - $\vec{w}_{t+1} = \vec{w}_t + y_t \vec{x}_t$, $M_{t+1} = M_t + 1$
  - else:
    - $\vec{w}_{t+1} = \vec{w}_t$, $M_{t+1} = M_t$

- Mistakes: $M \leq \left(\frac{R}{\gamma}\right)^2$ $\quad R = \max_t \|\vec{x}_t\|^{(2)}$

(note $\vec{w}_{m+1}$ does not necessarily linearly separate data set S, as M does not depend on m, the size of S)

2/2

3: Online Learning

· Perceptron mistake bound on disjunction learning:

$$M \leq R^2 \|\vec{u}\|^2 \quad (\|\vec{u}\|^2 = \frac{1}{\gamma^2})$$

$\vec{x} \in \{0,1\}^n$, use feature map $\phi(\vec{x}) = (\vec{x}, 1)$

$\vec{u}^* \in \mathbb{R}^{n+1}$ separates w/ margin of 1:

$$u_i^* = \begin{cases} 2 & i \text{ is a literal} \\ 0 & i \text{ is not a literal} \\ -1 & i \text{ is bias} \end{cases}$$

no. literals ↓

$$\Rightarrow \|\vec{u}\|^2 = 4k+1, \quad \|\phi(\vec{x})\|^2 \leq (n+1) \Rightarrow \underline{M \leq (4k+1)(n+1)}$$

· Winnow:

· Algorithm: $\vec{w}_1 = \vec{1}$

for $t = 1 \ldots m$:

$$\hat{y}_t = \begin{cases} 0 & \text{if } \vec{w}_t \cdot \vec{x}_t < n \\ 1 & \text{o/w} \end{cases}$$

if $\hat{y}_t \neq y_t$:

$$w_{t+1,i} = w_{t,i} \times 2^{(y_t - \hat{y}_t) x_{t,i}}$$

· Mistakes:

$$W_t = \sum_i w_{t,i}, \quad W_1 = n$$

positive mistake → · on mistake, $y_t = 1$, $W_{t+1} \leq W_t + n$

negative mistake → · on mistake, $y_t = 0$, $W_{t+1} \leq W_t - n/2$

$$\Rightarrow W_{m+1} \leq n + M_p n - M_n \frac{n}{2}$$

$$\Rightarrow M_n \leq 2k(\log n + 1) + 2$$

$$\Rightarrow \underline{M \leq 2 + 3k(\log n + 1)}$$

$M_p \leq k(\log n + 1)$ as on positive mistake relevant weight doubles, & when $w_{t,i} \geq n$ it will definitely no longer change

exponential improvement over Perceptron in n!

reduce $O(n)$ $\phi(a)\cdot\phi(b)$ to $O(1)$ $k(a,b)$

$\phi(\vec{x}) = (\vec{x}, 1)$

ANOVA Kernel

$$w_t = \sum_{q \text{ mistakes}} \alpha_q \cdot \phi(\vec{x}_q)$$

$$\vec{w}_t \cdot \phi(\vec{x}_t) = \sum_{q} \alpha_q \phi(\vec{x}_q)\phi(\vec{x}_t)$$

$$= \sum_{q} \alpha_q k(\vec{x}_q, \vec{x}_t)$$

- DNF case study:

$$x_1 x_2 x_3 \vee \bar{x}_1 \bar{x}_3 \vee x_2 x_4$$

- Perceptron:
  - Prediction: $O(n \times \#\text{mistakes})$
  - Mistake bound: $O(k \cdot 2^n)$

→ fast, poor bound

- Winnow:
  - Prediction $O(2^n \times \#\text{mistakes})$
  - Mistake bound $O(k \cdot \ln 2^n)$

→ slow, good bound

$$w_{t,i} = \exp\left[-\eta \sum_q \alpha_q \phi(\vec{x}_q)_i\right]$$

cannot do kernel trick

5: Support Vector Machines

- Goal: given data $S = \{(\vec{x}_i, y_i)_{i=1}^m \in \mathbb{R}^d \times \{-1, 1\}\}$ find a maximally separating hyperplane, ie

$$H_{\vec{w},b} = \{\vec{x} \in \mathbb{R}^d : \vec{w}^T\vec{x} + b = 0\}$$

(classify $\hat{y} = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x}+b \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x}+b < 0 \end{cases}$)

dist of $\vec{x}_i$ from $H_{\vec{w},b}$ → $\rho_{\vec{x}}(\vec{w}, b) = \dfrac{|\vec{w}^T\vec{x} + b|}{\|\vec{w}\|} = \dfrac{y_i(\vec{w}^T\vec{x}+b)}{\|\vec{w}\|}$ (margin of $\vec{x}_i$; is hyperplane separates data (-ve for incorrect, y_i if not)

margin of separating hyperplane → $\rho_S(\vec{w}, b) = \min_i \rho_{\vec{x}_i}(\vec{w}, b)$

We want to maximise

$$\rho(S) = \max_{\vec{w},b}\left[\min_i\left(\frac{y_i(\vec{w}^T\vec{x}+b)}{\|\vec{w}\|} \cdot y_i(\vec{w}^T\vec{x}+b) \geq 0,\ i = 1, \dots, m\right)\right]$$

Canonical hyperplane → ~~• cannot solve,~~ no unique solution; need to restrict $\|\vec{w}\|$

• set $\|\vec{w}\| = \dfrac{1}{\rho_S(\vec{w}, b)}$ (so $\min_i[y_i(\vec{w}^T\vec{x}+b)] = 1$

~~$\rho(S) = \max_{\vec{w},b} \min_i [y_i(\vec{w}^T\vec{x}+b)$~~ merge to $\cdots \geq 1$

$$\rho(S) = \max_{\vec{w},b}\left[\frac{1}{\|\vec{w}\|} \cdot \min_i(y_i(\vec{w}^T\vec{x}+b)) = 1,\ y_i(\vec{w}^T\vec{x}+b) \geq 0\right]$$

$$\rho(S) = \frac{1}{\min_{\vec{w},b}[\|\vec{w}\| \cdot y_i(\vec{w}^T\vec{x}+b) \geq 1]}$$

- So want to minimize $\frac{1}{2}\vec{w}^T\vec{w}$, subject to $y_i(\vec{w}^T\vec{x}+b)\geq 1, \forall i$
- Equivalent to finding saddle point of Lagrangian function

$$L(\vec{w}, b; \alpha) = \frac{1}{2}\vec{w}^T\vec{w} - \sum_{i=1}^{m}\alpha_i\left(y_i(\vec{w}^T\vec{x}_i+b)-1\right) \quad (1)$$

($\alpha_i \geq 0$ are Lagrange multipliers)

$$\frac{\partial L}{\partial b} = -\sum_i y_i\alpha_i = 0$$

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \Rightarrow \vec{w} = \sum_{i=1}^{m}\alpha_i y_i \vec{x}_i \quad (2)$$

($\vec{w}$: linear combination of $\vec{x}_i$; ($\vec{x}_i$ called Support Vectors, or SVs); SVs usually small subset of data)

- Dual problem:
  - Sub (2) into (1) to get Dual problem; now complexity scales w/ m, not d

- Linearly non-separable case:
  - Introduce slack variables ξ_i to allow some points to have margin < 1
  - Minimise $\frac{1}{2}\vec{w}^T\vec{w} + C\sum\xi_i^k$ (C controls trade off between $\|\vec{w}\|^2$ & training error, $\sum\xi_i$)
  - subject to $y_i(\vec{w}^T\vec{x}_i+b) \geq 1-\xi_i$
    $\xi_i \geq 0, \quad i = 1, \ldots m$

again, $\vec{w} = \sum\alpha_i y_i \vec{x}_i$,

- C is equivalent to regularisation (ridge regression style, w/ $\lambda = \frac{1}{2C}$)

- Kernels:
  - can use a feature map $\phi(\vec{x})$, and ~~since~~ use the Kernel trick, so

$$f(\vec{x}) = \sum y_i\alpha_i K(\vec{x}_i, \vec{x}) + b \quad (\text{classify with } \text{sign}(f(\vec{x})))$$

- Regression:

$$\text{loss} = |y - f(\vec{x})|_\varepsilon = \max(|y-f(\vec{x})|-\varepsilon, 0)$$

(errors below ε don't count $\Rightarrow$ sparse solutions!)

6: Gaussian Processes

- Bayesian inference:
  - Prior: distribution over possible hypothesis
    $\rightarrow F = \{f(\vec{x}) = \langle \vec{w}, \phi(\vec{x}) \rangle : \vec{w} \in F\}$, and $\vec{w} \sim N(\vec{0}, I)$

(equivalent to 2-norm regularization)

  - Additive noise model:
    - measurements of outputs corrupted by additive noise
      $\rightarrow y | \vec{w}, \vec{x} \sim N(\langle \vec{w}, \phi(\vec{x}) \rangle, \sigma^2)$

(equivalent to minimising squared discrepancy; Least Squares Regression)

output of function is 'true' output, then corrupted by additive noise

  - Posterior:

$$dP_{post}(f) = dP(f) \cdot P(\vec{y} | S_x, f)$$

(prior; likelihood (of learning data S_x))

~~since we use $\vec{w}$~~

$$\Rightarrow dP_{post}(\vec{w}) \propto \exp\left[\frac{-\|\vec{w}\|^2}{2}\right] \cdot \exp\left[\frac{-\sum (y_i - \langle \vec{w}, \phi(\vec{x}_i) \rangle^2}{2\sigma^2}\right]$$

- MAP inference: max of posterior
  - as exp is monotonic,

$$\vec{w}_{MAP} = \underset{\vec{w}}{\operatorname{argmin}} \sum (y_i - \langle \vec{w}, \phi(\vec{x}_i) \rangle)^2 + \sigma^2 \|\vec{w}\|^2$$

  - equivalent to Ridge Regression problem, w/ $\lambda = \sigma^2$

$$\Rightarrow \vec{w}_{MAP} = X^T (K + \sigma^2 I)^{-1} \vec{y}$$ (where K is the kernel matrix)

- Dual solution:
  - sub in $\vec{w}_{MAP} = X^T \alpha$

$$\Rightarrow \sigma^2 \alpha = \vec{y} - K\alpha$$
$$\Rightarrow (K + \sigma^2 I_m)\alpha = \vec{y}$$
$$\Rightarrow \alpha = (XX^T + \sigma^2 I_m)^{-1} \vec{y}$$

and $g(\vec{x}) = \vec{x}^T \vec{w} = \vec{x}^T X^T \alpha = \sum_{i=1}^{m} \alpha_i k(\vec{x}, \vec{x}_i)$

(g: regression function)

prior → $P(\vec{z}|S) = \frac{1}{\sqrt{\det(2\pi K)}} \exp\left[\frac{-1}{2}\vec{z}^T K \vec{z}\right]$

noise model → $P(\vec{y}|\vec{z}) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left[\frac{-1}{2\sigma^2}\|\vec{y}-\vec{z}\|^2\right]$

- Gaussian Process Regression:
  - use framework for Bayesian inference

- Prior: for $S = \{\vec{x}_1, \ldots, \vec{x}_m\}$ training examples, $y_1, \ldots, y_m$ corresponding outputs distributed by multidimensional Gaussian w/ mean $\vec{0}$ & covariance of kernel matrix
  - a point in this represents an assignment of all inputs to a specific output

- Inference:

$$P_{post}(\vec{w}|S) \propto \exp\left[\frac{-\sum(y_i - \langle\vec{w}, \phi(\vec{x}_i)\rangle)^2}{2\sigma^2} - \frac{\|\vec{w}\|^2}{2}\right]$$

$$\propto \exp\left[\frac{-1}{2}(\vec{w}-\vec{w}_{map})^T \Sigma^{-1} (\vec{w}-\vec{w}_{map})\right]$$

$$\Rightarrow \vec{w}|S \sim N(\vec{w}_{map}, \Sigma), \quad \Sigma^{-1} = \frac{1}{\sigma^2}(X^T X + \sigma^2 I)$$

- Error bars:

$$y = \vec{w}_{map}^T \phi(\vec{x}) + \vec{v}^T \sqrt{\Sigma}^T \phi(\vec{x}), \text{ where } \vec{v} \sim N(\vec{0}, I)$$

$$\sigma_y^2 = E[(y - \vec{w}_{map}^T \phi(\vec{x}))^2] = \phi(\vec{x})^T \sqrt{\Sigma}\, \vec{v}\vec{v}^T \sqrt{\Sigma}^T \phi(\vec{x})$$
$$= \phi(\vec{x})^T \Sigma \phi(\vec{x})$$
$$= k(\vec{x},\vec{x}) - k^T$$

- Evidence:
  - probability of the data

$$P(\vec{y}) = \int_f P_{post}(f|S)\, df$$

- For a GP:

$P(\vec{y}|S) = \int P(\vec{y}|\vec{z}) \cdot P(\vec{z}|S)\, d\vec{z}$ → $\log(P(\vec{y}|S)) = \frac{-1}{2}\vec{y}^T (K + \sigma^2 I)^{-1} \vec{y} - \frac{1}{2}\log\det(K + \sigma^2 I)$

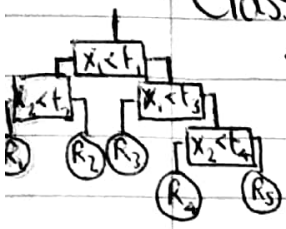
$$- \frac{m}{2}\log 2\pi$$

  - can be good for model selection, over different σ or kernel hyperparameters

1/2

7: Tree based learning & Ensemble Methods

Classification & Regression trees:



- Divide space into hyperrectangles by recursive binary partition
- Regression:
  - $c_n = \text{ave}(y_i \mid \vec{x}_i \in R_n)$, ie minimize square error in region n
  - goal: $\min_{R_1 \dots R_N} \left[\sum_{i=1}^{m} \left(y_i - \sum_{n=1}^{N} \text{ave}(y_j \mid \vec{x}_j \in R_n) I[\vec{x}_i \in R_n] \right)^2 \right]$
    - but computationally intractable; need to use heuristic
  - Greedily grow tree:
    - search for best split to minimize square error by splitting data in 2
      - $O(dm)$, as split can 1: be in any dimension 2: be between any 2 training points
    - repeat until max no. data at each node (eg 5)
  - Cost-complexity pruning:
    - this tree will overfit training data; need to find $T_\lambda \subseteq T$ minimizing

$$C_\lambda(T) = \sum_{n=1}^{|T|} m_n Q_n(T) + \lambda |T|$$

n goes over all leaves in T
m_n is no. points in that leaf
$Q_n(T) = \frac{1}{m_n} \sum_{\vec{x}_i \in R_n} (y_i - c_n)^2$

$\Rightarrow \sum_{n=1}^{|T|} m_n Q_n(T)$ = training error

- algorithm: weakest link pruning:
  - successively collapse nodes giving smallest per node increase in training error
  - T_λ is in this sequence of subtrees

- Classification:
  - classify by majority vote in region
  - Growing tree:
    - minimise one of:
      - Gini index $= \sum_k \rho_{mk}(1-\rho_{mk})$
      - Cross entropy $= \sum_k \rho_{mk} \log \frac{1}{\rho_{mk}}$
  - Pruning tree:
    - prune smallest misclassification error $= 1-\rho_{mk(m)}$, $k(m) = \operatorname{argmax}_k \rho_{mk}$

empirical class probs for region m

$$\rho_{mk} = \frac{1}{M_m} \sum_{\vec{x}_i \in R_m} I(y_i = k)$$

- Weak learner:
  - algorithm which consistently finds classifiers slightly better than random
- Ensemble methods:
  - Wisdom of crowds → single prediction may be wrong, but crowd majority often can be correct

eg, h_i predicts correct w/ probability $\frac{1}{2}+\gamma$ (independently)

$$H_T = \text{Sign}\left(\sum_{i=1}^{2T+1} h_i\right)$$

$$P(H_T \text{ wrong}) = \sum_{i=0}^{T} \binom{2T+1}{i} \left(\frac{1}{2}+\gamma\right)^i \left(\frac{1}{2}-\gamma\right)^{2T+1-i}$$

by Chernoff bound, $\leq \exp\left(\frac{-4}{5}\gamma^2 T\right)$ ← exponential decay w/ T or γ^2!

- Bagging:
  - reduce classifier variance by training T variations, each on M examples (sampled with replacement), ⇒ ~63% distinct (non-repeats)
    - conventionally, M = size of training set
- Random forest:
  - Decision trees high variance; use bagging with these
  - to make further decorrelated, only use a subset of k features for each tree
    - k usually $= \sqrt{d}$ or $\log d$

7: Tree based learning & Ensemble Methods

- **Boosting: (Adaboost)**
  - Initialise $D_1(1) \dots D_1(m) = \frac{1}{m}$ (weights on training points)
  - for $t = 1 \dots T$:
    - fit $h_t : \mathbb{R}^d \to \{-1, 1\}$ using D_t (Weighted training error)
    - choose α_t $\left(\alpha_t = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}, \varepsilon_t = \sum_i D_t(i) \, I[h_t(\vec{x}_i) \neq y_i]\right)$
    - $D_{t+1}(i) \propto D_t(i) \cdot \exp[-\alpha_t y_i h_t(\vec{x}_i)]$ ← 'harder' examples get higher weight
      - must keep $\sum_i D_t(i) = 1$ → so need to divide all by Z_t
    - return $H(\vec{x}) = \text{sign}\left(\underbrace{\sum_t \alpha_t h_t(\vec{x})}_{f(\vec{x})}\right)$ linear combination of weak learners, weights controlled by training errors
  - Convergence:
    - training error: $\frac{1}{m} \sum_i I(H(\vec{x}_i) \neq y_i) \overset{①}{\leq} \frac{1}{m} \sum_i e^{-y_i f(\vec{x}_i)} \overset{②}{=} \prod_t Z_t$
      - ① as $H_i(\vec{x}) \neq y_i \Rightarrow e^{-y_i f(\vec{x}_i)} \geq 1$
      - ② as $D_{t+1}(i) = \frac{1}{m} \cdot \frac{1}{\prod_t Z_t} \cdot \prod_t e^{-\alpha_t y_i h_t(\vec{x}_i)}$ (& expand $\sum e^{-y_i f(\vec{x}_i)}$)

$\Rightarrow$ need to try to minimise Z_t every step

$$Z_t = \sum_i D_t(i) \exp[-\alpha_t y_i h_t(\vec{x}_i)] = \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t}$$

$\Rightarrow$ minimised by $\alpha_t = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$

$\Rightarrow Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \sqrt{1-4\gamma_t^2}$, $\gamma_t = \frac{1}{2} - \varepsilon_t$

$\Rightarrow$ Train error $\leq e^{-2\sum_t \gamma_t^2} \leq e^{-2T\gamma^2}$ if $\gamma_t \geq \gamma \geq 0$

- **Additive models:**
  - AdaBoosting greedily solves $\text{Min}\left[\sum_i V(y_i, \sum_t \alpha_t h_t(\vec{x}_i))\right]$, where $V(y, \hat{y}) = e^{-y\hat{y}}$ (exponential loss)
  - Exponential loss as it punishes negative margins, & slightly prefers larger positive margins (whilst square loss would punish larger positive margins; bad!)

also, for fixed α_t, minimising exp loss for h_t equiv to minimising its misclassification error

- Summary
  - CART good as interpretable & work on ordinal data; less feature cleaning needed
  - Bagging used on full trees, reduce variance
  - Boosting often only use single split stumps, reduces bias
    - boosting often struggles w/ noisy data

9: Learning Theory

- $S = \{(\vec{x}_1, y_1), \dots, (\vec{x}_m, y_m)\}$, iid sample from unknown $P(\vec{x}, y)$

unknown, true error $\rightarrow \varepsilon(f) = E[V(y, f(\vec{x}))] = \int V(y, f(\vec{x}))\, dP(\vec{x}, y) = C_0$

empirical error = training error $\rightarrow \varepsilon_{emp}(f) = \frac{1}{m} \sum_i V(y_i, f(\vec{x}_i)) = \hat{C}_S$

- Overfitting:
  - minimising ε_{emp} leads to overfitting if model complexity unlimited
  - we want to estimate true error (test error) & minimise this
- Test set bound: (d = dimensions of $\vec{x}$)

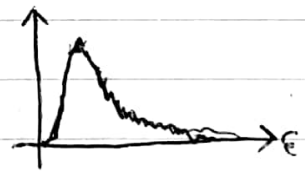
$$C_0 \leq \hat{C}_S + \sqrt{\frac{\ln 1/\delta}{2m}}$$

if S is iid samples, with probability $1-\delta$ (for $\delta > 0$)

(m = size of S)

- Generalisation of a learner:
  - algorithm A chooses function $A_F(S)$ from function space F, in response to training set S

random variable $\rightarrow \varepsilon(S, A, F) = E_{(\vec{x},y)}[\underbrace{l}_{\text{loss function}}(A_F(S), \vec{x}, y)]$



  - can be approximated w/ cross-validation (get distribution of this RV)
  - expected value of this RV can be misleading; we should focus on confidence bounds (δ = confidence parameter)
    - source of PAC; δ chance we've been 'misled' by training set

(Probably Approximately Correct)

  - Prob of being misled in classification:

$$P_S(err_S(f) = 0, err(f) > \varepsilon) = (1 - err(f))^m \leq (1-\varepsilon)^m \leq e^{-\varepsilon m}$$

$$\Rightarrow \varepsilon = \ln\left(\frac{1}{\delta}\right)/m$$ ensures probability less than δ

- Countable function classes:

$F = \{f_1, \dots, f_n, \dots\}$, let prob of being misled by f_n $< q_n \delta$, where $\sum_n q_n \leq 1$

$$\Rightarrow q_n\delta = P\left(\bigcup_{n=1}^{\infty} A_n\right), \quad A_n = \left\{ err_S(f_n)=0, err(f_n) > \frac{1}{m}\ln\frac{1}{q_n\delta} \right\}$$

$$\leqslant \sum_n P(A_n)$$

$$\Rightarrow P_S\left\{ \exists f_n : err_S(f_n)=0, err(f_n) > \frac{1}{m}\ln\frac{1}{q_n\delta} \right\} \leqslant \delta$$

$$\Rightarrow err(f_n) \leqslant \frac{1}{m}\left[\ln\frac{1}{q_n} + \ln\frac{1}{\delta}\right] \quad (\text{if } err_S(f_n)=0)$$

- note, must be prior weights to functions; expected generalisation minimal if weights $= q_n$

- By Hoeffding's inequality, also have $\varepsilon(f_n) \leqslant \varepsilon_{emp}(f_n) + \sqrt{\log 1/\delta / 2m}$

- let $f_S = \underset{f_n}{\operatorname{argmin}}\ \varepsilon_{emp}(f_n)$, with corresponding q_S

$$\varepsilon(f_S) \leqslant \varepsilon_{emp}(f_S) + \sqrt{\frac{\log 1/q_S + \log 1/\delta}{2m}}$$ with confidence $1-\delta$

- Sample complexity bound:
  - finite $\mathcal{H}$, $q_n = 1/|\mathcal{H}|$

samples to avoid overfitting w/ confidence $1-\delta$ $\longrightarrow$ $\Rightarrow m(\varepsilon, \mathcal{H}, \delta) = \dfrac{\log|\mathcal{H}| + \log 1/\delta}{2\varepsilon^2}$

- Structural risk minimisation:
  - model selection approach to choosing hypothesis space given $\mathcal{H}_1 \subset \cdots \subset \mathcal{H}_q$
  - let $f_{S,q}$ be minimizer of ε_{emp} in $\mathcal{H}_q$

$$\varepsilon(f_{S,q}) \leqslant \varepsilon_{emp}(f_{S,q}) + \sqrt{\frac{\log|\mathcal{H}_q| + \log 1/\delta}{2m}}$$ w/ confidence at least $1-\delta$

  - we choose $\mathcal{H}_q$ to minimise rhs of this

9: Learning Theory

- Uncountable $\mathcal{H}$
  - need to convert to finite
  - replace test point w/ test set, ie a second 'ghost' sample
  - Double sample trick:

$$A(h) = \{err_X(h) = 0\},\ B(h) = \{err(h) \geq \varepsilon\},\ C(h) = \{err_Y(h) \geq \frac{\varepsilon}{2}\}$$

$$P^{2m}(C(h) \mid A(h), B(h)) = P^{2m}(C(h) \mid B(h)) > 0.5$$

$$\Rightarrow P^{2m}(XY \in X^{2m} : \exists h \in H : A(h), C(h)) \geq$$
$$P^{2m}(\quad '' \quad : \quad '' \quad : A(h), B(h), C(h)) =$$
$$P^{2m}(\quad '' \quad : \quad '' \quad : A(h), B(h)) \cdot P(C(h) \mid A(h), B(h))$$
$$\Rightarrow P^{m}(X \in X^{m} : \exists h \in H : A(h), B(h))$$
$$\Uparrow$$
$$2P^{2m}(XY \in X^{2m} : \exists h \in H : A(h), C(h))$$
$$\Uparrow$$
$$2B_H(2m)\, P^{2m}(XY \in X^{2m} : \exists h \in H : A(h), C(h))$$

$$\left[\text{where } B_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^d, \text{ where } d = VCdim(H)\right]$$

$B_H(m)$: max cardinality of set of functions H when restricted to m points

- Finally, by symmetrisation,

$$P^{2m}[XY \in X^{2m} : A(h), C(h)] \leq E^{2m}[P_{\sigma \sim \Sigma}[A(h), C(h) \text{ for } \sigma(XY)]]$$
$$\Uparrow$$
$$2^{-\varepsilon m/2} = E^{2m}(2^{-\varepsilon m/2})$$

$$\Rightarrow \boxed{\varepsilon(m, H, \delta) = \frac{2}{m}\left(d \log \frac{2em}{d} + \log \frac{2}{\delta}\right)}$$

- bound on generalisation error w/ confidence $1-\delta$
- VC-dimension (complexity/capacity of H)

- Similar for non-zero training error; introduces a square root & significantly worse

- Bias / Variance decomposition:

$$f_H = \underset{f \in H}{\operatorname{argmin}}\, \varepsilon(f)$$

$$\varepsilon(f_S) - \varepsilon(f_H^*) = \underbrace{\varepsilon(f_S) - \varepsilon(f_H)}_{\text{variance}} + \underbrace{\varepsilon(f_H) - \varepsilon(f^*)}_{\text{bias}}$$

$$\varepsilon(f_S) - \varepsilon(f_H) \leq \varepsilon(f_S) - \varepsilon_{emp}(f_S) + \varepsilon_{emp}(f_H) - \varepsilon(f_H)$$

$$\leq 4\sqrt{2\,\frac{h\log(2m/h+1) + \log 2/\delta}{m}}$$

- variance ↑ w/ complexity of H
  ↓ w/ sample size m
- bias ↓ w/ complexity of H
  independent of m

- Critisms of PAC theory:
  - doesn't agree w/ practice, eg SVM
    - infinite dimensional feature space ⇒ infinite VC dimension
    - but works well in practise

10: Sparsity Methods

- Boosting Generalisation error:
  - usually decreases even after train error = 0
    - ⇒ Margin distribution ~~increasing~~ still keeps decreasing

$$\text{margin}_i = y_i f(\vec{x}_i)$$

$$\text{Generalisation Error} \leq P_{emp}(\text{margin} < \theta) + O\left(\frac{\sqrt{d/m}}{\theta}\right)$$

(d is VC dimension of set of weak learners)

- L_1-Sparsity:
  - Rademacher complexity alternative measure
    - not increased by taking convex close of H