We will maximize this lower bound.
For best results, we have to choose $\{z_{iy}\}$ that make this lower bound as close to original as possible

Choose $\{z_{iy}\}$ such that

$$\max_{\{z_{iy}\}} \sum_i \sum_y z_{iy} \log\left(\frac{\pi_y f(x^i, \theta_y)}{z_{iy}}\right)$$ .... concave in $\{z_{iy}\}$ for fixed $\{\theta_y, \pi_y\}$

$\sum z_{iy} = 1$
$z_{iy} \geq 0$ → use Lagrangian to enforce this constraint

eg - For $i = 5, y = 1$,

$$\tilde{L} = \sum_i \sum_y z_{iy} \log\left(\frac{\pi_y f(x^i, \theta_y)}{z_{iy}}\right) + \sum_i \lambda_i \left(\sum_y z_{iy} - 1\right)$$

$$\frac{\partial \tilde{L}}{\partial z_{51}} = \log \pi_1 f(x^5, \theta_1) - \log(z_{51}) - 1 + \lambda_5$$

$$= 0$$

$$\therefore z_{51} = e^{\lambda_5 - 1} \pi_1 f(x^5, \theta_1)$$

To eliminate λ_5, $\sum_y z_{5y} = 1$

$$\sum e^{\lambda_5 - 1} \pi_y f(x^5, \theta_y) = 1$$

$$\therefore z_{51} = \frac{\pi_1 f(x^5, \theta_1)}{\sum_y \pi_y f(x^5, \theta_y)}$$

$$= \frac{P(y=1)\, P(x^5, y=1)}{\sum (\quad)}$$

$\therefore z_{iy} = P(y \mid x^i)$

New objective :-

$$\max_{\theta_1 \ldots \theta_k \pi_1 \ldots \pi_k} \; \max_{\substack{z_{iy} \geq 0 \\ \sum_y z_{iy} = 1 \\ \{z_{iy}\}}} \sum_{i=1}^{N} \sum_{y=1}^{k} \left[z_{iy} \log \frac{\pi_y f(x^i, \theta_y)}{z_{iy}} + \sum_i \lambda_i \left(\sum_y z_{iy} - 1 \right) \right]$$

.... concave in $\{z_{iy}\}$ for fixed $\{\theta_i\}\{\pi_i\}$

$\{\theta_i\}\{\pi_i\}$ $\{z_i\}$

"Expectation - Maximization (EM) Algorithm"

- Initially chosen arbitrary values for $\{\theta_i\}\{\pi_i\}\{z_{iy}\}$

  eg - $\pi_i = \frac{1}{k} \quad \forall i$.

- while (change) {

  E - step :-

  for (i = 1 to N) {

  for (y = 1 to k) {

$$z_{iy}^t = P(y \mid x^i, \theta^t, \pi^t) \quad \text{.... soft assignment of points to clusters}$$

$$= \frac{\pi_y^t f(x^i, \theta_y^t)}{\sum_{y=1}^{k} \pi_y f(x^i, \theta_y^t)}$$

M-step :- (maximization)

$$\{\pi^{t+1}\}\{\theta^{t+1}\} = \arg\max \underbrace{\sum_i \sum_y z_{iy}^t \log \pi_y f(x^i, \theta_y)}_{\text{because } z_{iy}^t \text{ wrt } \theta \& \pi \text{ only}}$$

$$\triangleq Q^t(\theta, \pi)$$

P.T.O.

- First consider the π variables.

$$\frac{\partial Q^t}{\partial \pi_y} = \frac{\partial}{\partial \pi_y}\left[\sum_i \sum_j z_{iy}^t \log(\pi_y f(x^i; \theta_y)) + \lambda\left(\sum_k \pi_k - 1\right)\right]$$

$$= \sum_i \frac{z_{iy}^t}{\pi_y} + \lambda = 0$$

$$\therefore \pi_y = \frac{-\sum_i z_{iy}^t}{\lambda}$$

$$= \frac{\sum_i z_{iy}^t}{\sum_k \sum_i z_{iy}^t}$$

$$= \frac{\sum_i z_{iy}^t}{\sum_i \underbrace{\left(\sum_k z_{iy}^t\right)}_{=1}}$$

$$= \boxed{\frac{\sum_i z_{iy}^t}{N}}$$

- For the θ variables,

For instance, f is Gaussian $= \frac{1}{2}(x^i - \mu_y)^T \Sigma_y^{-1} (x^i - \mu_y)$

$$f(x^i, \theta_y) = \frac{1}{(2\pi)^{d/2} |\Sigma_y|} e$$

$$\rightarrow \frac{\partial Q^t}{\partial \mu_y} = \frac{\partial}{\partial \mu_y}\left(\sum_i \sum_j z_{iy}^t \left(\log \pi_y - \frac{1}{2}(x^i - \mu_y)^T \Sigma^{-1} (x^i - \mu_y) - \log |\Sigma|\right)\right)$$

$$= -\sum_{i=1}^{N} z_{iy}^t (x^i - \mu_y) = 0$$

$$\mu_y^{t+1} = \frac{\sum_{i=1}^{N} z_{iy}^t x^i}{\sum_i z_{iy}^t}$$

$T \frac{^{2N}C_{K^0}}{K!}$ P^5_{2N}

24/10 BAGGING

- It is a type of ensemble learning
- Training multiple classifiers to predict final output.

- Consider 'm' classifiers $C_i(x)$, each giving a prediction $\hat{Y}_i$

- Predicted label = Majority $(\hat{Y}_1 \dots \hat{Y}_n) = \hat{y}_B$

- Let $p \triangleq$ probability that a classifier gives correct prediction
Assume classifiers to be independent.

Probability that k classifiers are correct $= {}^mC_k p^k (1-p)^{m-k}$

$\therefore$ We predict $\hat{y}_B$ for majority (i.e. - when more than 50% of classifiers are correct)

$\therefore$ Probability that $\hat{y}_B$ is correct $= \sum_{k=m/2}^{m} {}^mC_k p^k (1-p)^{m-k}$

Intuitively, for this probability ($\hat{y}_B$ is correct) to be $\geq p$,

$p \geq \frac{1}{2}$

→ Two methods of creating classifier committees :-
1) Bootstrap sampling
2) Random forests (attribute subsetting)

PAGE NO
DATE / /

I Bootstrap Sampling

$D = \{(x^1, y^1) \ldots\ldots (x^N, y^N)\}$

Convert D into into a point distribution (PD)

$$P_0(x,y) = \begin{cases} \frac{1}{N} & \text{for } (x,y) \in D \\ 0 & \text{otherwise} \end{cases}$$

- Obtain a 'bootstrap sample' containing N elements (x,y) ~~taken~~ as N ~~not~~ samples of above uniform distribution.
  - Every bootstrap sample is like an artificial dataset that we can run our classifier on.

- Use a classifier on each bootstrap sample.

→ Expected number of distinct elements (x,y) in a bootstrap sample
$= N$ (probability that an example is selected in any trial)

$= N$ (1 − probability that the example is not selected at all)

$$= N\underbrace{\left(1 - \left(1 - \frac{1}{N}\right)^N\right)}_{X}$$

$$\lim_{N\to\infty} X = 1 - \frac{1}{e} \approx 0.63.$$

∴ On an average, a bootstrap sample contains 0.63N distinct instances.

→ Bagging is often useful in settings where each classifier is over-fitted on their bootstrap sample.

eg Create a decision tree without pruning.

Bagging smooths the predictions from multiple classifiers and suppresses ill-effects of over-fitting.

II Random Forests

Create artificial datasets by sampling attributes instead of instances

No. of attributes for every X = d

~~Random~~

For each decision tree you make :-

For each node of this tree :-

→ Randomly select $\sqrt{d}$ no. of attributes from the total d attributes.

→ Select the best attributes from these $\sqrt{d}$ attributes based on splitting criterion.

BOOSTING

- Opposite of bagging

* Bagging :- Use complicated, overfitted classifiers and smooth out their predictions

• Boosting :- Start with simple base classifiers and train them sequentially to get a more complicated decision surfaces

- Consider each instance in D to have a weight.
  - Initially (for first classifier), all weights are equal.

- After ~~each~~ of each classifier, increase the weights of wrongly classified instances. Then train the next classifier.

* Weight = Coefficient of error of that instance in total loss function.

- Final classifier = Weighted Average of all classifiers

• $D \longrightarrow C_1(x) \xrightarrow{\alpha_1} D_{21} \longrightarrow C_2(x) \xrightarrow{\alpha_2} \cdots\cdots D_{m-1} \longrightarrow C_m(x) \xrightarrow{\alpha_m}$

α_i = weight of classifier $C_i(x)$

- Consider no. of classes = $k = 2$.

I ADABOOST ALGORITHM

Algorithm- Initially assign equal weights to all examples (instances)

$$\omega_i = \frac{1}{N}$$

- for j in range (max boost stages):

$C_j(x) \leftarrow$ Train classifier on weighted dataset $D_{j-1} \equiv \{\omega_i, x_i, y_i\}$

eg- For loss regularizer classifier,

$$\min_{\theta} \sum_{i=1}^{N} \omega_i \text{ loss } (y^i, C(x^i|\theta))$$

'Weighted loss'

$\epsilon_j \leftarrow$ Weighted error of $C_j(x)$ on $D_{j-1} \equiv (\omega_i, D)$

$$= \sum_{i=1}^{N} \omega_i \, \delta(y_i \neq C_j(x_i))$$

~~$\alpha_j \leftarrow$ Weight of classifier~~

if $\epsilon_j > 1/2$, stop

$\alpha_j \leftarrow$ Weight of classifier

$$= \log\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$$

$\omega_i \leftarrow$ Weight updation $\forall\ i$

$$= \omega_i \left(\frac{1-\epsilon_j}{\epsilon_j}\right) \quad \text{if } y_i \neq C_j(x_i)$$

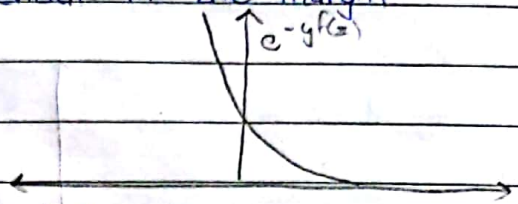
Rescale ω_i to make their sum 1.

PAGE NO.
DATE / /

Theorem Let $f(x) = \sum_{j=1}^{m} \alpha_j C_j(x)$ and $y \in \{-1, 1\}$

1 Choose a loss function which is exponential in the margin

$$\text{loss}(y f(x)) = e^{-y f(x)}$$



2 Train the classifiers and α_j's sequentially
(& don't change classifiers $1 \ldots j-1$ when training C_j)

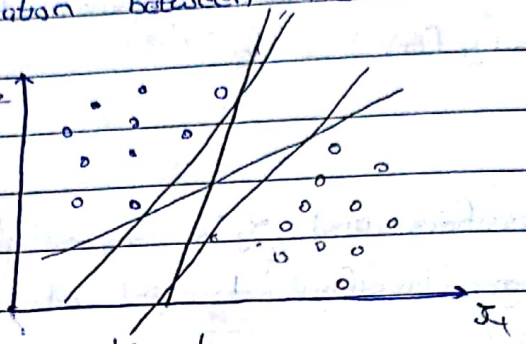
Then :- Adaboost is an optimal algorithm.

26/10

SUPPORT VECTOR MACHINES

- Separate instances of two classes by hyperplanes in the original linear or a special high-dimensional space while maximizing the margin of separation between the classes.

eg $d = 2, k = 2$



SVM gives maximum separable line

Many linear boundaries of separation can give zero error on training data

- Hyperplanes in linear space :- $w_1 x_1 + \dots + w_d x_d + w_0 = 0$

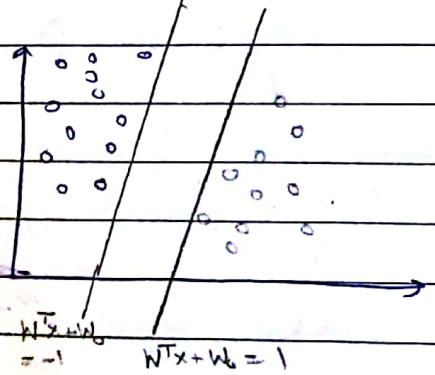
1) Separable case

W s.t. for all positive points ($y_i = 1$),

$$Wx^i + W_0 \geq 1$$

and for all negative points ($y_i = -1$),

$$Wx^i + W_0 \leq -1$$



Geometric distance between hyperplanes $= \frac{1-(-1)}{\|W\|} = \frac{2}{\|W\|}$

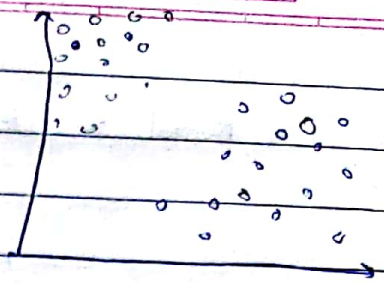
↑ no W_0

Objective :- $\underset{W, W_0}{\text{Max}} \frac{2}{\|W\|}$ s.t. $y_i (W^T x^i + W_0) \geq 1 \quad \forall i$.

OR $\underset{W, W_0}{\text{Min}} \frac{\|W\|}{2}$ s.t. $y_i (W^T x^i + W_0) \geq 1 \quad \forall i$

or $\frac{\|W\|^2}{4}$ because $\|W\|^2$ is convex

PAGE No.
DATE / /



2) Non-separable case.

Allow some slack 'ξ_i' ≥ 0 s.t.

$y_i (W^T x^i + W_0) \geq 1 - \xi_i$

Objective :- Min $\frac{\|W\|^2}{2} + C \sum_1^N \xi_i$

C ↑ Hyper-parameter (by trial and error)

→ ξ_i = Hinge loss with '$f(x) = Wx^i + W_0$'

↳ $\max(0, 1 - yf(x))$

Proof :- $\xi_i \geq 0$ (by definition)

$\xi_i \geq 1 - y_i (W^T x^i + W_0) = 1 - yf(x)$

$\therefore \xi_i \geq$ Hinge Loss

Because we want to minimize the objective,

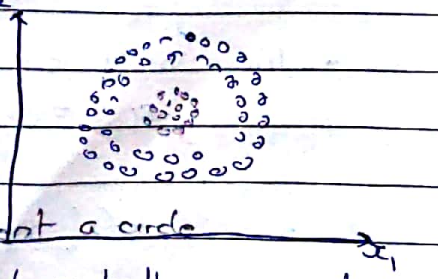
ξ_i = Hinge Loss.

∴ For non-separable case,

Objective = Min (C · Hingeloss + Regularizer)

... like loss regularizer

I Non-linear SVMs :



These points can be separated linearly if they are embedded into a 5D space consisting of $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

$W_1 x_1 + W_2 x_2 + W_3 x_1^2 + W_4 x_2^2 + W_5 x_1 x_2$ to represent a circle

- SVMs can help do this using kernels, without explicitly representing the points in 5D space.

Work with kernels in original space itself.

PAGE NO.
DATE / /

- Kernel :- $k(x, x') \equiv \underbrace{\phi(x) \cdot \phi(x')}$
  ↳ Embedding vector of x.

Consider $x \equiv (x_1, x_2)$ and $x' \equiv (x_1', x_2')$ (two points in 2D space)

eg Kernel function $\equiv (x \cdot x' + 1)^2 = K(x, x')$

$\equiv ((x_1, x_2) \cdot (x_1', x_2') + 1)^2$

$\equiv (x_1 x_1' + x_2 x_2' + 1)^2$

$\star \equiv x_1^2 x_1'^2 + x_2^2 x_2'^2 + 1 + 2x_1 x_1' x_2 x_2' + 2x_1 x_1' + 2x_2 x_2'$

- ~~Types of kernels~~ Consider

~~1) Linear kernel~~

?

- Types of kernels
  1) Linear :- $K(x, x') \equiv x \cdot x'$
  2) Quadratic :- $(x \cdot x' + 1)^d$
  3) Gaussian or RBF :- $e^{-\sigma(x - x')^2}$ ... popular.

- The SVM is defined in terms of the kernel function as
  $f(x) \equiv \sum_i^N \alpha_i K(x^i, x) y_i + w_0$

- eg :- $K(x^i, x) = x^i \cdot x$

x^i, y_i = training dataset point
x = new point for training

$$\therefore f(x) = \sum_{i}^{N} \alpha_i (x^i, x) y_i + W_0$$

$$= \underbrace{\left(\sum \alpha_i x^i y_i\right)}_{\text{☆} = W^T} x + W_0$$

α_i = weights attached to points $\forall i$

31/10 • For Gaussian kernel, low σ :- bad decision boundary
high σ :- good decision boundary, so possibility of overfitting

• Think of kernel function $f(x)$ to denote similarity between training point x and dataset points (x^i, y_i).
x is assigned class based on what points are in its vicinity
→ Gaussian :- High σ :- Every x^i has very less influence over x.

II TRAINING SVMs

Objective $\underset{w, w_0, \xi_i}{\text{Minimize}} \quad \frac{1}{2}\|w\|^2 + C\sum \xi_i \quad \text{s.t} \quad y_i(w^T x^i + w_0) \geq 1 - \xi_i$ and $\xi_i \geq 0$

• For hinge loss formulation :-

$$\underset{w, w_0}{\text{Minimize}} \quad \frac{1}{2}\|w\|^2 + C \sum \max(0, 1 - y_i(w^T x^i + w_0))$$

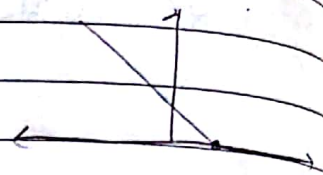
PAGE No.
DATE / /

→ Two methods

1 Hinge loss formulation:-

Use stochastic gradient descent algorithms

- But hinge loss is not differentiable in W. So, we will work with sub-gradients instead.



☆ $\min_{W,W_0} G(W,W_0) \equiv \max(0, G'(W,W_0))$

non-differentiable, differentiable parts

Sub-gradient at $(W,W_0) \equiv \begin{cases} 0 & \text{if } G'(W,W_0) \le 0 \\ \nabla_{W,W_0} G'(\tilde{W},\tilde{W_0}) & \text{otherwise} \end{cases}$

2 Primal Dual Method

Starting with constrained QP formulation

Primal (P)	Lagrangian dual of P
$\min_W F(W)$	$\max_{(\alpha_1 \dots \alpha_n)} \theta \equiv \min_W F(W) + \sum \alpha_k g_k(W)$
s.t. $g_k(W) \le 0 \quad \forall k$	s.t $\alpha_k \ge 0 \quad \forall k$
where $F(W), g_k(W)$ are convex	

- We will ~~use tan~~ solve Lagrangian dual of given primal question as it does not have constraints on g_k

In our case, $F(w)$

PAGE No. / DATE / /

Theorem 'Weak Duality' ← candidate solution

Let $\hat{w}$ be a feasible solution of the primal $\hat{P}$

Then, $\forall\ \alpha_k \geq 0$, the dual $\theta(\vec{\alpha}) \leq F(\hat{w})$

$$\theta(\alpha) = \min_{w} F(w) + \sum \alpha_k g_k(w)$$

$$\leq F(\hat{w}) + \sum_k \alpha_k g_k(\hat{w})$$

Since $\hat{w}$ is a solution of primal, $g_k(\hat{w}) \leq 0$

$$\leq F(\hat{w}) = \text{Primal objective value at } \hat{w}.$$

Dual feasible values ← ..... → Primal feasible values

$\theta(\alpha^1)$ $\theta(\alpha^2)$ ... $\theta(\alpha^*) < F(w^*)$ $F(w^1)$.....

- $F(w^*) - \theta(\alpha^*)$ is known as 'duality gap', which needs to be minimized.
- $\theta(\vec{\alpha})$ for all $\alpha_j \geq 0$ $\leq$ $F(\vec{w})$ for all feasible w

- Now objective :- Maximize $\theta(\alpha)$

$$\equiv \max_{\alpha_1 \cdots \alpha_k} \left[\min_{w} \left(F(w) + \Sigma_g \alpha_k g_k(w) \right) \right]$$

for $\alpha_k \geq 0$ and for $k \in \{1 \cdots k\}$

Theorem 'Strong Duality'

If $F(w)$, $g_k(w)$ are convex, then duality gap is zero.

$$\theta(\alpha^*) = F(w^*)$$

PAGE NO.
DATE / /

→ Applying Primal – Dual to SVMs :-

- Primal :- $\min_{w, w_0, \xi_i} \frac{1}{2}\|w\|^2 + C\sum \xi_i$

$F(w)$ is convex ✓

such that :- $\left.\begin{aligned} 1 - \xi_i - y_i(w^T x^i + w_0) &\leq 0 \\ -\xi_i &\leq 0 \end{aligned}\right\} \forall i \in I$

- New dual objective :-

$$\max_{\alpha_i, \mu_i} \left[\min_{w, w_0, \xi_i} \left(\frac{1}{2}\|w_0\|^2 + C\sum_i \xi_i + \sum_i \alpha_i \left(1 - \xi_i - y_i(w^T x + w_0)\right) + \sum_i \mu_i(-\xi_i) \right) \right]$$

such that :- $\left\{ \alpha_i \geq 0 \right\}$ $\left\{ \mu_i \geq 0 \right\}$

– Write as $\max_{\alpha_i \mu_i} \min_{w w_0 \xi_i} L(w, w_0, \xi_i, \alpha_i, \mu_i)$ → No. of parameters $= d + 1 + 3N$

– $\nabla_w L = w + \sum \alpha_i (-y_i x^i) = 0$

$$\Rightarrow \boxed{w^* = \sum_{i=1}^{N} \alpha_i y_i x^i}$$

represents different hyperplanes for different values of $\vec{\alpha}$
best hyperplane for optimum $\alpha = \alpha^*$

– $\nabla_{w_0} L = -\sum \alpha_i^* y_i = 0$

$$\therefore \sum_{y_i=1} \alpha_i^* = \sum_{y_i=-1} \alpha_i^*$$ ..... equal weights for both classes

PAGE No.
DATE / /

- $\nabla_{\xi_i} L = C - \alpha_i^* - \mu_i^* = 0$

$\alpha_i^* + \mu_i^* = C$

Step 2 Substitute $w = \sum \alpha_i y_i x^i$ in $L(w, w_0, \xi_i, \alpha, \mu)$ and simplify to get

$$\min_{w, w_0, \xi_i} L(w, w_0, \xi_i, \alpha_i, \mu_i) \equiv \frac{-1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underline{x^i x^j} + \sum_{i=1}^{N} \alpha_i$$

s.t $\alpha_i \geq 0, \mu_i \geq 0$

s.t. $\sum_{i}^{N} \alpha_i y_i = 0$

$\alpha_i + \mu_i = C$

$\alpha_i \geq 0$

$\mu_i \geq 0$.

Step 3 Eliminate μ_i

$$\max_{\alpha_1 \cdots \alpha_N} \frac{-1}{2} (\quad) \quad \text{s.t.} \quad \sum \alpha_i y_i = 0$$

$\alpha_i \geq 0$

$\alpha_i \leq C$. because $\alpha_i + \mu_i = C$

2/11 • $x^i \cdot x^j$ in above objective is linear kernel

→ Extending the dual to arbitrary kernels using 'Kernel trick'

1. In the dual, x^i and x^j only appear as $x^i \cdot x^j$
2. For a non-linear embedding $\Phi(x)$ corresponding to some other kernel, we can use $\Phi_k(x^i) \cdot \Phi_k(x^j) = K(x^i, x^j)$

• New kernelized dual objective :-

$$\text{Maximize}_{\alpha_1 \cdots \alpha_N} \quad \frac{-1}{2} \left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x^i, x^j) \right) + \sum \alpha_i$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad \sum \alpha_i y_i = 0$$

PAGE NO.
DATE / /

- Now, $W = \sum_{i}^{N} \alpha_i y_i x^i$

Classifier :- $W^T x + W_0 = (\sum \alpha_i y_i x^i) x + W_0$

$= \sum \alpha_i y_i K(x^i, x) + W_0$

→ Solving the objective.

- The objective written previously is concave in $\alpha_1 \dots \alpha_N$
- Gradient ascent, subject to constraints
  - Time-consuming :- $O(N^2)$ ∴
  - Cannot use single-co-ordinate gradient ascent because $\sum \alpha_i y_i$ won't be followed

→ 'SMO Algorithm'

- Two co-ordinate gradient ascent - Change two α_is in each step
- Initially, $\alpha_i^{t=0} = 0 \quad \forall \; i \in \{1 \dots N\}$

while (change) {
- Pick two α's (α_i^t, α_j^t) to change
- Choose a change amount λ^t.

→ $\alpha_i^{t+1} \leftarrow \alpha_i^t \pm \lambda^t$

$\alpha_j^{t+1} \leftarrow \alpha_j^t \pm \lambda^t$ .... sign depends on whether $y_i = y_j$ to maintain $\sum \alpha_i y_i = 0$

}

- Choosing a λ :- + if $y_1 \neq y_2$ and − if $y_1 = y_2$

$\arg\max_{\lambda} \; \theta(\alpha_1^t, \dots \alpha_i^t \pm \lambda, \alpha_j^t \pm \lambda, \dots \alpha_N^t)$

This can be solved in closed form by equating gradient of θ to 0.

PAGE No.
DATE / /

λ^t = value closest to λ^* while satisfying $0 \le \alpha_i^t + \lambda^t \le C$
and $0 \le \alpha_j^t \pm \lambda^t \le C$

eg $t = 0, \quad \alpha_1 \cdots \alpha_N = 0$

Iteration 1 :- Choose (α_1, α_2) :- Say $y_1 \neq y_2$

$$\theta(\lambda) = \frac{-1}{2}(\lambda)(+\lambda)\, K(x_1, x_2) y_1 y_2 + 2\lambda$$

$$\frac{\partial \theta}{\partial \lambda} = -\lambda K(x_1, x_2)(-1) + 2$$

$$= 0$$

$$\therefore \lambda^* = \frac{-2}{K(x_1, x_2)}$$

Say $K(x_1, x_2) = -1$ and $C = 1$

$$\therefore \lambda^* = 2$$

To ensure that α_i & $\alpha_j \in (0, C)$

$$\lambda^t = 1$$

$$\therefore \alpha_1 = 1, \quad \alpha_2 = 1$$

Theorem KKT conditions

Primal	Dual
$\min_w F(w)$	$\max_{\alpha_1 \dots \alpha_k} \left[\min_w F(w) + \sum \alpha_k g_k(w) \right]$
s.t $g_k(w) \le 0 \quad \forall k = 1 \dots k$	s.t $\alpha_k \ge 0 \quad \forall k = 1 \dots k$

Primal & dual objectives are equivalent iff

When $F(w)$, $g_k(w)$ are convex, then a (α^*, w^*) are optimal solutions for the dual and primal respectively if and only if

1) $\nabla_w F(w^*) + \sum \alpha_k^* \nabla g_k(w^*) = 0$
2) $\alpha_k g_k(w) = 0 \quad \forall k = 1 \dots k$
3) $\alpha_k \ge 0$
4) $g_k(w^k) \le 0$

→ Apply KKT to SVMs

i $w^* = \sum_1^N \alpha_i^* y_i x^i$
ii $\sum \alpha_i^* y_i = 0$
iii $\alpha_i^* + \mu_i^* = C$

} 1st KKT condition

iv $\mu_i^* \ge 0$
v $\alpha_i^* \ge 0$
vi $\xi_i^* \ge 0$

} Primal dual feasibility

vii $\alpha_i^* (1 - y_i (w^{*T} x^i + w_0^*) - \xi_i^*) = 0 \quad \forall i$
viii $\mu_i^* \xi_i^* = 0$

} 2nd KKT condition

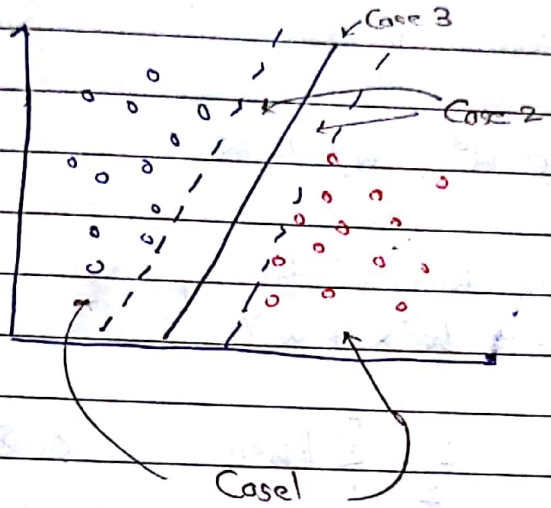
Case 1 Consider i s.t. $y_i (w^* x^i + w_0) > 1$ and $\xi_i^* = 0$
∴ From vii, $\alpha_i^* = 0$

Case 2 Consider i s.t. $\xi_i^* \ge 0$
∴ From viii, $\mu_i^* = 0$
∴ From iii, $\alpha_i^* = C$

Case 3 Consider i s t $0 < \alpha_i^* < C$

$\Rightarrow \mu_i^* > 0 \quad , \quad \xi_i^* = 0$

$\Rightarrow y_i (w^T x^i + w_0) = 1$



- In practice, most examples are outside the band ($\alpha_i^* = 0$)

$$f(x) = w^* x + w_0$$
$$\equiv \sum \alpha_i^* y_i K(x^i, x) + w_0$$

∴ We don't need to do this kernel computation for most points

→ How to compute w_0 ?

Find an example i for which α_i is $\in (0, C)$

For this example, $y_i (w^* x^i + w_0) = 1$

PAGE No.
DATE / /

→ SVMs for multi-class classification

- k classes
- Like in logistic regression classifiers, allocate parameters w^y, w_0^y for each class y.

$$w^y \in \mathbb{R}^d, \quad w_0^y \in \mathbb{R}$$

- Score for an example being in class y is

$$s(x, y) = w^y x + w_0^y$$

- Objective :- $\underset{\{w^y, w_0^y\}, \xi_i}{\text{Minimize}} \quad \frac{1}{2} \sum_{1}^{k} \|w^y\| + C \sum \xi_i$

such that $w^{y_i} x^i + w_0^{y_i} \geq w^y x^i + w_0^y + 1 - \xi_i$

$\forall \; i = 1 \ldots N$

and $\forall \; y \neq y_i$

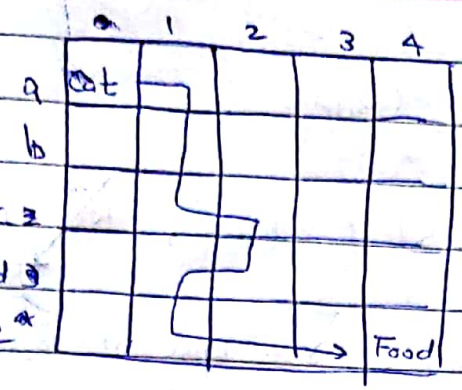
and $\xi_i \geq 0$

- Hinge loss :-

$$\max_{y \neq y_i} \left(0, \; 1 - [s(x^i, y_i) - s(x^i, y)]\right)$$

9/11

Reinforcement Learning



Cat to reach food

An 'agent' looks at current position and decides an action
L/R/U/D

→ Reward (R_t)
- Scalar feedback signal
- Indicates how well agent is doing at time 't'
- Agent tries to maximize cumulative reward.

→ Policy
= How an agent decides upon an action to take.

* Longer paths could be made to have lower cumulative reward than shorter ones.
Use a 'discount factor' $\gamma \in (0,1)$ to account for this

Time	0	1	2	3	.....	N	Cumulative
Reward	r_0	r_1	r_2			r_N	
Long-term reward	r_0	γr_1	$\gamma^2 r_2$	------		$\gamma^N r_n$	$\sum \gamma^i r_i$

→ Components of RL Agent
1 Policy - Agent's behaviour
2 Value function - Reward
3 Model - Representation of environment

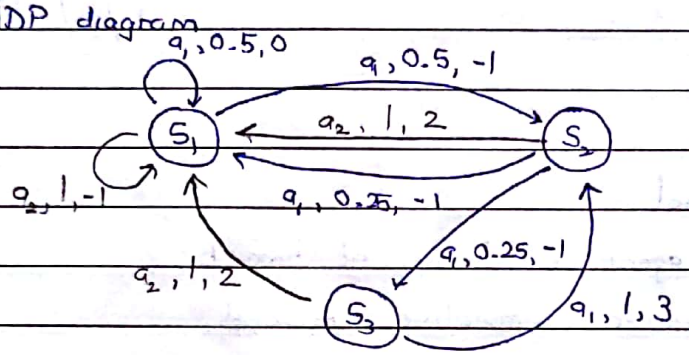
PAGE NO.
DATE / /

A] Markov Decision Problem (MDP)

S ≡ Set of states, A ≡ Set of actions, R(s, a) ≡ Reward function
R(s, a, s')

T(s, a, s') ≡ Transition function, γ = Discount factor
(Probability)

eg MDP diagram



(a, probability, reward)

$S = \{s_1, s_2, s_3\}$ $A = \{a_1, a_2\}$

eg - $T(s_1, a_2, s_1) = 1$

$T(s_1, a_2, s_2) = 0$

$T(s_1, a_2, s_3) = 0$

.... and so on

eg - $R(s_1, a_2, s_1) = -1$

... and so on

- Trajectory : $(s^0, a^0, r^0), (s^1, a^1, r^1) \ldots (s^t, a^t, r^t)$
- Expected long-term reward : $E[\sum \gamma^i r^i] \triangleq V^{\pi}(s_1)$
  = 'Value function'

π = Policy ∴ eg ∴ $\pi = \{a_1, a_2, a_1, a_1\}$

s_1 = Initial state

PAGE No.
DATE / /

TFT :- Most optimal policy π^*

$$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s \in S, \forall \pi$$

Theorem $\exists$ a π^* for every MDP which is better than (at least just as good as) all other πs for all starting states S.

- π^* may not be unique