SMAI Mid1-Mid2

Feature Extraction (Missed)

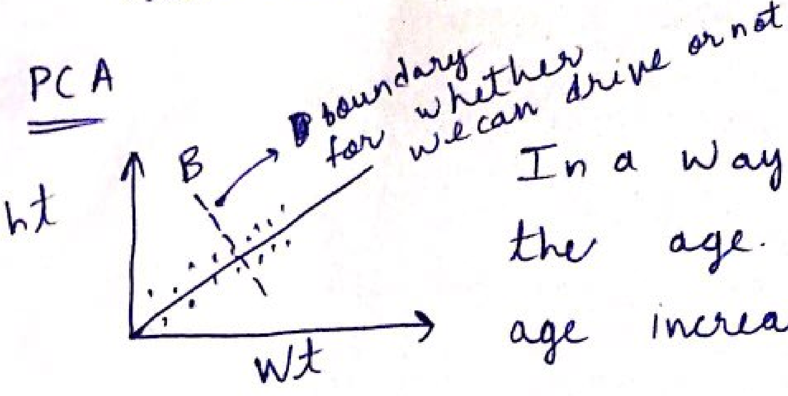
$$Z = U X$$

$$\underset{K\times 1}{\overset{I}{[\;]}} = \underset{K}{\left[\overset{d}{\left[\quad\right]}\right.} \overset{I}{[\;]}\Big]_{d\times 1}$$

$\hookrightarrow$ linearly dependent on I.

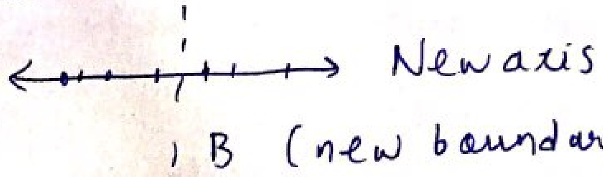
Reduction in dimensionality (Finish notes)

PCA



In a way we are plotting the age. As ht, wt ↑ as age increases (just an example).

So we can effectively reduce computational complexity by plotting the age instead (project the values onto the new axis).



If we simply project onto the x or y axis the points would get crammed. Hence find the axis which maximizes variance.

$$\text{Max} \frac{1}{N} \sum_{i=1}^{N} (z_i - \bar{z})^2$$

$$z = U^T x$$

$U \rightarrow$ the vector whose dot product with pt gives the value of the point on the new axis.

Max : $\frac{1}{N} \sum_{i=1}^{N} (U^T x - \underset{U^T\mu}{\cancel{Mean}})^2$

$Mean = 1/N \sum_{i=1}^{N} z_i$

[Calculation]

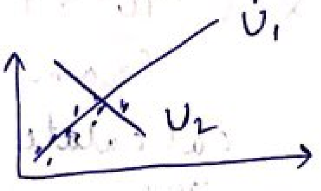
$\frac{1}{N} \Sigma (\underbrace{U^T x - U^T \mu}_{scalars}) (x^T U - \mu^T U)$

$\Rightarrow \frac{1}{N} \Sigma \; U^T \; (x - \mu) (x^T - \mu^T) \; U$

$\Rightarrow \; U^T \Sigma U$ such that $U^T U = 1$.

[Proved already]. Now find out Eigen Vector (Principle component)

U (Eigen vector of Σ)



(Eigen vectors ⊥)

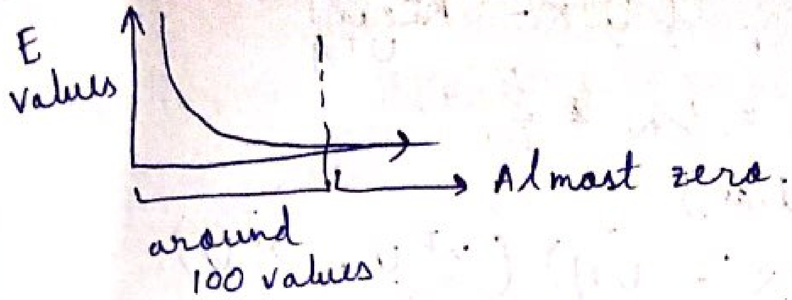
So U_2 is useless for us.

Faces Example

Eigen faces.

Lets say we have seen 1000's of faces already. Each of the face is say 100 x 100. Let us convert this to a $10^4 \times 1$ dimensional vector by appending all the rows one by one. calculate the covariance matrix of all the X's. Calculate the corresponding Eigen Vector set U_1 to U_{10000}.

Now if we observe carefully the Eigen value graph, it would be like this:



Now calculate the matrix

$$Z = \begin{bmatrix} x_1^T U_1 \\ x^T U_2 \\ \vdots \\ x^T U_{100} \end{bmatrix}$$ New basis

Now given a new face find out x ($10^4 \times 1$ valued vector) from 100×100

Now we calculate the fall.

$$\text{Face} = \sum_{i=1}^{10{,}000} \alpha_i U_i$$

So, Want add to the face ↑

$$\alpha_i = x^T U_i$$

But higher valued's $\simeq 0$.

$$\therefore \text{Face} = \sum_{i=1}^{100} \alpha_i U_i$$

The new face can be constructed as a linear combination of the U_i's.

Any face can be regenerated this way.

α_i's : Like the signature of the face.

[Eigen reconstruction].

We can reconstruct the faces given the signature. We can use that in compression. (Very efficient) But the basis is data dependent. Changes with the matrix X.

If the matrix X doesnt have similar x_i's (say some images of car, dog, human etc) what happens is that the values wont near zero as we go on. (Cant cutoff at 100).

We want to maximize:

$$\boxed{\frac{\sum_{i=1}^{\text{limit}(100)} \lambda i}{\sum_{j=1}^{10000} \lambda j}}$$

here $N = 1000$ and $d = 10^4$

We have a trick to find out Eigen values of a matrix given another matrix's values.

$$X X^T U = \lambda U$$

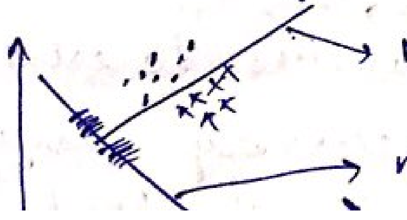
$$X^T X \; V = \lambda V$$

$$X^T \underbrace{X \; (X^T U)}_{\lambda U} = \lambda (X^T U)$$

which is Known

Use this to find out Eigen vectors as its computationally complex.

PCA is not the best for dimensionality reduction for classification. Its good for compression tasks. PCA naturally is unsupervised. But once we have class information, using Fischer's method is better (LDA).



Bayesian Parameter Estimate

Generative Model: We have a prob desity over all the feature values in the world. We can say with what probability some feature might occur. We can generate examples of a particular class.

For finding whether some feature vector belongs to some class;

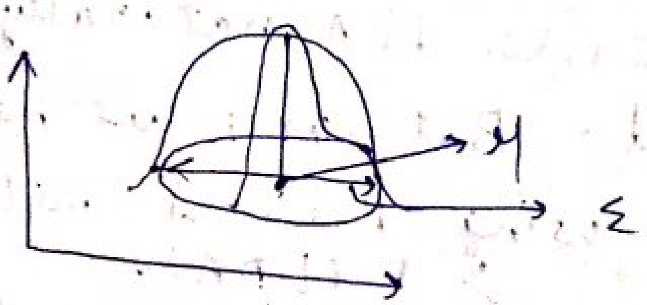
$$P(w_i/x) = \frac{\boxed{P(x/w_i)} \, P(w_i)}{P(x)}$$

$\rightarrow$ generative model

How do we learn $P(x/w_i)$??
(Given some samples)

2 Methods (We assume a function form for $P(x/w_i)$. If we dont assume any f'n form to $P(x/w_i)$; We cant learn;

We can have minimal assumptions and still do it $\Rightarrow$ Read about this.



We try to learn μ, Σ

More number of parameters to estimate; more no of examples necessary. Bayes classifier is ideal when the density f'n learned is perfect.

Bayes doesn't work well in general practice because: ① Assumption of f'n might be wron

② data points may be limited in no.

Data set $D = \{x_1, x_2 \cdots x_n\}$

Lets say $P(x) \sim N(\mu, \sigma^2)$

① Maximum Likelihood principle to find out the parameters.

$P(\theta) = \arg\max_{\theta} . P(D/\theta)$. (Find that θ which maximiz the likelihoo of occ. of D).

$= \arg\max_{\theta} \prod_{K=1}^{n} P(x_K/\theta)$.

After derivation,

$$\boxed{\begin{aligned} \hat{\mu} &= \frac{1}{n} \sum_{K=1}^{n} x_K \\ \hat{\sigma}^2_{ME} &= \frac{1}{n} \sum_{K=1}^{n} (x_K - \mu)^2 \end{aligned}}$$

② Maximum Aposterior probability (MAP)

$$\hat{\theta}_{map} = \arg\max_{\theta} \underbrace{P(D/\theta) \cdot P(\theta)}_{\Rightarrow P(\theta/D)}$$

$$\underbrace{P_\theta(x/W_i)}_{\Downarrow} \sim N(\underset{\downarrow}{\mu}, \sigma^2)$$

This is the f'n to be learnt.

$P(\mu) = N(\mu_0, \sigma_0^2)$

$P(\mu/D) = P(D/\mu) \cdot P(\mu)$.

For this we say the f'n is a normal density f'n which has the params μ, σ_0^2

Now to find out μ; We can just have a look at some examples and take their mean value. We say this mean is the μ.

Scanned with CamScanner

This is the first method. (MLE).

If we have some prior understanding of the mean of the values beforehand, we use this prior knowledge to improve our value of μ.

So the prior understanding: $P(\mu) \sim N(\mu_0, \sigma_0^2)$

guess of mean of the mean value (μ).

variance in the estimation of mean.

Now given some examples,

$$P(\mu/D) = \frac{P(D/\mu) \cdot P(\mu)}{P(D)}$$

We keep applying this to the examples. Then our calculated μ gets closer to god given μ and variance decreases.

<u>Example</u> Apple example. (feature: Only wt. of apple)

Let's say actual $\mu = 200g$, $\sigma = 50g$ in the world.

Our prior understanding might say mean wt is around $150g \pm 100g$.

This becomes μ_0, σ_0.

<u>Case-1</u> $P(\mu/D) \sim N(\mu_0, \sigma_0^2)$.

$P(\mu/D) = P(D/\mu) \cdot P(\mu)$

$$\Rightarrow \propto \prod_{K=1}^{n} \underbrace{P(x_K / \mu)}_{\text{this is. also normal}} \cdot \underset{\downarrow}{P(\mu)}$$

If prior density is normal posterior also normal den [Reproducing density]

In a way we can say that shape doesnt change.

$$\Rightarrow \propto \prod_{K=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2}\left(\frac{x_K - \mu}{\sigma}\right)^2} \cdot \frac{1}{\sqrt{2\pi}\,\sigma_0} e^{-\frac{1}{2}\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2}$$

$$\Rightarrow P(\mu / D) = \frac{1}{\sqrt{2\pi}\,\sigma_n} e^{-\frac{1}{2}\left(\frac{\mu - \mu_n}{\sigma_n}\right)^2}$$

Updated estimates of μ. (Updated vals of μ_0, σ_0).

$$\Rightarrow \boxed{\mu_n = \frac{n\sigma_0^2}{\underset{\downarrow}{n\sigma_0^2 + \sigma^2}} \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0}$$

variance of samples at that point

As $n \uparrow$ $\mu_n \approx \hat{\mu}_n$

But if variance in data is too large we cant rely on the data set too much & hence values $\downarrow$.

$$\boxed{\sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}}$$

As $n \uparrow$, $\sigma_n \downarrow$ [Variance in Knowledge in μ $\downarrow$].

Uncertainity Keeps decreasing and μ_n becomes nearer to the original value.

Recursive formulation:

$P(D^n/\theta) = P(x_n/\theta) \cdot P(D^{n-1}/\theta)$.

Instead of discarding the uncertainity in calculating μ & ; We can use it to find out a better $P(x/\omega_i)$.

$$P(x/\omega_i) = \int P(x/\mu) \cdot P(\mu/D) \cdot d\mu$$

$\Downarrow$

Including the uncertainity in calculating μ.

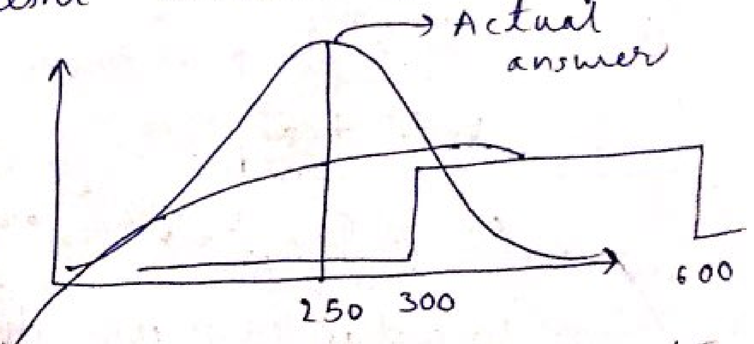
It turns out that after it's inclusion $\sim \boxed{N(\mu_n, \sigma^2 + \sigma_n^2)}$

So simply the variance of mean estimate σ_n^2 gets added up.

If we use only the MAP estimate, it becomes $N(\mu_n, \sigma^2)$.

Its ok to have a bad prior estimate. But having a very strong belief of the value doesn't enable learning.

Eg:



If this is our prior estimate.

Then how many ever examples we have we close in on the value ≈ 300.

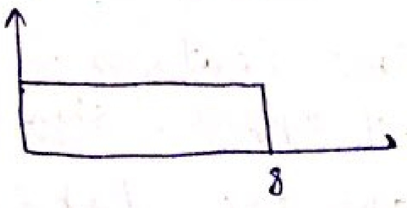
Now, can we have uniform density func. instead of Normal density?

$P(x_i|\theta)$ is $U(0,\theta)$ $\begin{cases} 1/\theta & 0 \le x \le \theta \\ 0 & \text{otherwise} \end{cases}$

Eg: $D = \{4, 7, 2, 8\}$

What is the value of θ which maximizes the occ of D?

$\theta = 8$.



If $\theta = 8$, prob: $\left(\frac{1}{8}\right)^4$

But if $\theta < 8$, say 5

~~Value~~ Prob: $\frac{1}{5} \times \frac{1}{5} \times 0 \times 0 = 0$.

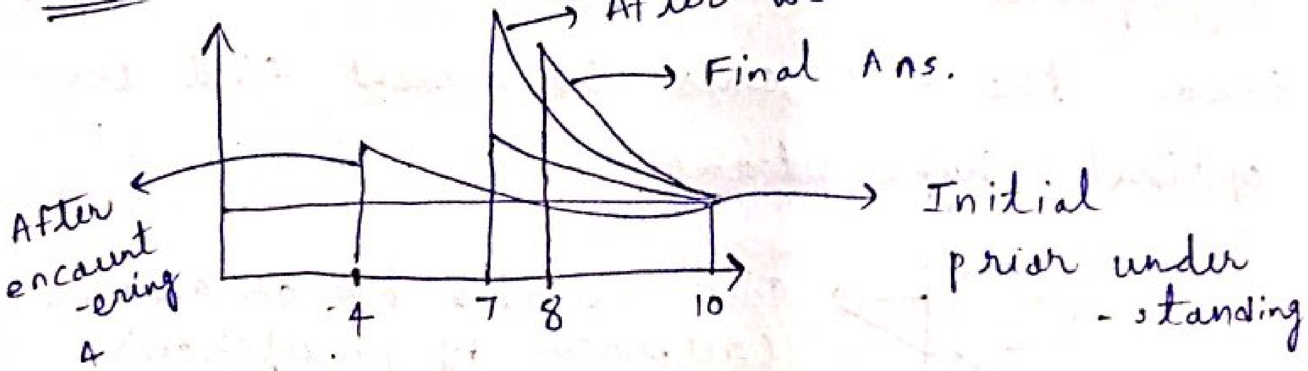
If $\theta = 10$,

$\left(\frac{1}{10}\right)^4$.

$\therefore$ Value which maximizes: Max(D).

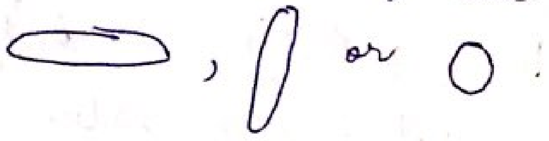
$\hookrightarrow$ MLE.

Map



Naive Bayes Classifier

Generally if the dimensionality of feature vector is d, d^2 is the size of cov matrix and $d^2/2$ is approximately the number of parameters to be estimated.

If we say that the cov matrix
is a diagnal Matrix (off diag elements = 0)
No. of parameters to be learnt now : 'd'.
The ellipse we obtain can be of the
foll forms : 

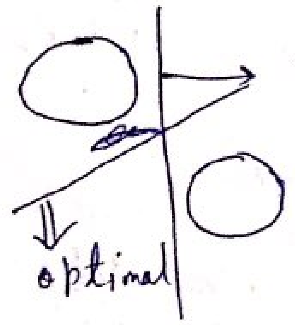
It assumes that the features are independe
of each other.
$P(x/w) = P(x_1/w) \cdot P(x_2/w) \ldots P(x_3/w)$.
x_1 to x_n : features.
→ for example: wt of apple, shine of apple

Each feature can have one type of
estimated f'n finally.

<u>Lecture</u> [Prof Jawahar]

Task : Classification

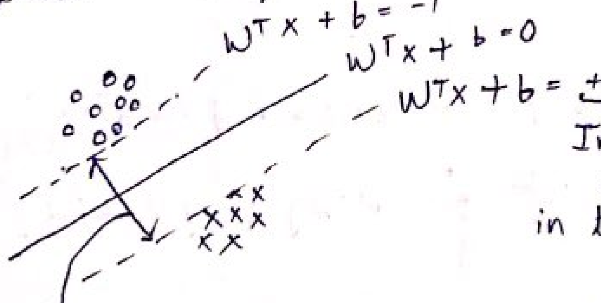
Perceptron Algorithm tries to find a hyperplane which separates the +ve samples from the -ve ones: It doesn't find the optimal hyperplane.



This can also be an answer (returned by perceptron).

For this task, we employ an SVM.
(Support Vector Machine).

It not only tries to find a line which minimizes the error; It also finds that line which maximizes the margin such that samples are correctly classified.



$\longrightarrow$ No real significance

Instead of +1, we can have α; but α can be divided in the entire equation to get back +1 α.

$\hookrightarrow$ This margin has to be maximized.

Classification Rule :

$$W^T x_i + b \geq +1 \quad y_l = +1$$
$$W^T x_i + b \leq -1 \quad y_l = -1$$
$$\Rightarrow \boxed{y_i (W^T x_i + b) \geq 1}$$
$$(\forall i)$$

Dist between the 2 lines :

$$\frac{b+1}{\|W\|} - \frac{b-1}{\|W\|} \Rightarrow \underbrace{\frac{2}{\|W\|}}_{}\Big]$$ has to be maximize

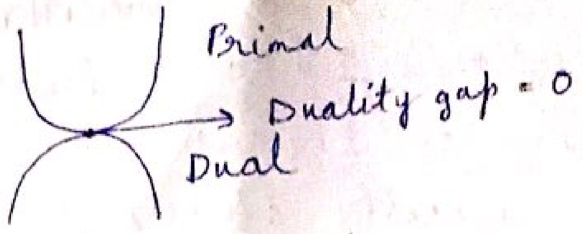
$\hookrightarrow$ has to be min.

This is a constraint optimization

Primal : Minimize $\frac{1}{2} W^T W$

such that $\underset{\substack{\Downarrow \\ \{+1, -1\}}}{y_i} (W^T x_i + b) \geq 1$ $\forall i$

SVM turns out to be a well behaved problem. This problem has a dual problem For example, to minimize some quantity we ~~mo~~ change to its compliment problem and try to maximize it.



↳ For SVM

Dual problem: (This is a convex problem).

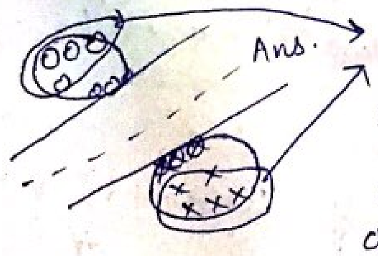
$$\text{Max} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j x_i^T x_j$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad , \quad \alpha_i > 0$$

$$\boxed{W = \sum_{i=1}^{N} \alpha_i y_i x_i}$$

Now to implement the SVM using the dual problem, we have many solvers to do so. We can implement the primal using grad descent; We need to modify it a bit to include the constraints. (try to impl. this).

Intuition behind the dual problem solving:



Even if we remove these samples, the (Ans) line doesnt change. So these are insignificant to the final Answer.

The only samples which matter are the ones which lie on the boundary. If we remove these samples, hyperplane changes.

$$W = \sum_{i=1}^{N} \underbrace{\alpha_i}_{\text{coeff of}} y_i x_i$$

so for the insignificant samples, ideally.

$\alpha_i = 0$.

$\therefore$ Finally we have some values for α for few samples; for all others

$\alpha_i = 0$.

When we solve the dual problem, we get the non zero α's.

~~The set of~~ Every sample which have non zero α's are support vectors.

$\{SV\} \equiv$ with non zero α

In the testing phase:

$W^T x + b \lessgtr 0$, then classify

complexity : If $x : d \times 1$
W is also $d \times 1$.

$\therefore$ Comp: $O(d)$.

We can do it run time by not pre-computing W. (Useful method in case of non linear SVMs or Kernels).

$$\underbrace{\sum_{\forall SV} \alpha_i y_i x^T}_{W^T} x + b \lessgtr 0.$$

Comp: $O((\# SD) \times d)$.
↳ typically small

If the # SD is large, the problem becomes tougher to solve.

$\Rightarrow$ [[$\bar{\alpha}$ is sparse]]

2 Observations:

① In all the cases the ~~and~~ samples are used as dot product:
$X^T X$ in testing, training
which is just a scalar.

Pruning SVM's: Solvers ~~the~~ which generally try to remove some α_i's to reduce space complexity.

Estimate of how hard the problem is:
(~~Te~~ Leave one out tester).
Lets say $N = 20$; train over ~~oe~~ 19 samples & test ~~over~~ on the left out sample.

This way, do it 20 times.
For all the SV's the answer might be wrong. For Non SV's we get right ans.

Upper bound on misclassifications: $\boxed{\# SV}$

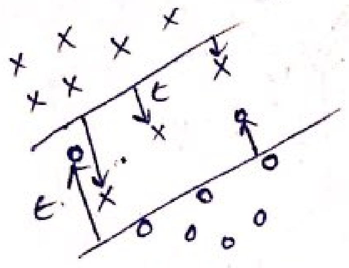
We started with the assumption that the samples are linearly separable which may not be the case in real life applications.
So to incorporate this, we modify our formulation to include a penalty for the violators.

New problem formulation:

PTO.

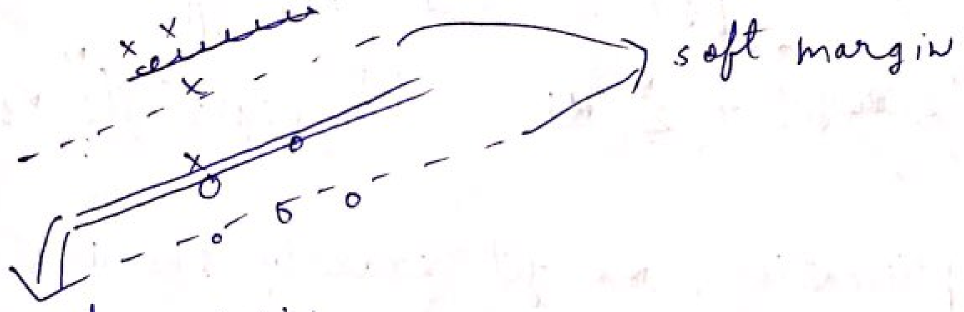
Minimize $\frac{1}{2} W^T W + C \sum_{i=1}^{N} \epsilon_i$

$y_i(W^T x_i + b) \geq 1 - \epsilon_i \quad \forall i$



This is a soft. margin problem. (previous one: hard margin).

~~This way~~ The same problem for a hard margin would have given the foll



hard margin.

C → basically tells the weightage for the error sum. (Need to tune the value of C)

[Using grid search: search in the order of 10].

We can also consider 2 cases:

1. Reducing the amount ~~on~~ of error
2. Reducing the no. of errors.

$\Downarrow$

considering errors as outliers

Instead of $\sum \epsilon_i$, we ~~ha~~ can have $\sum \epsilon_i^2$ ← Easier to handle.
(L1, L2 SVM's).

Conversion of the primal problem to dual problem:

$$J(W, b, \alpha) = \frac{1}{2} W^T W - \sum_{i=1}^{N} \alpha_i [y_i (W^T x_i + b) - 1]$$

do the foll $\frac{\partial J}{\partial W} = 0$, $\frac{\partial J}{\partial b} = 0$.

$$\Downarrow \qquad \boxed{W = \sum_{i=1}^{N} \alpha_i y_i x_i} \qquad \Downarrow \qquad \boxed{\sum_{i=1}^{N} \alpha_i y_i = 0}$$

substituting ~~a~~ in the above equations,

$$\Rightarrow \frac{1}{2} \left[\sum_{i=1}^{N} \alpha_i y_i x_i^T \right] \left[\sum_{j=1}^{N} \alpha_j y_j x_j \right] - \sum_{i=1} \alpha_i y_i x_i^T W$$

$$- \sum_{i=1}^{N} \alpha_i \cancel{y_i b}^{\,0} + \sum_{i=1}^{N} \alpha_i \qquad \Downarrow \; \left(\sum_{i=1}^{N} \alpha_i y_i x_i^T \right) \left(\sum_{j=1} \alpha_j y_j x_j \right)$$

On simplification, we get exactly the dual.

Q: Why should we ~~mini~~ maximize this??

Consider the term: $\alpha_i [y_i (W^T x_i + b) - 1]$

$\swarrow \quad \swarrow$

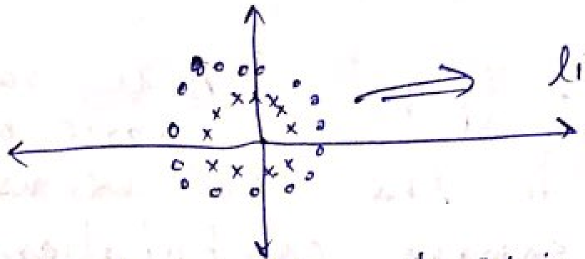
Either of them would be

complexities: pre comp: $O(d)$
summing: $O(Nd)$.

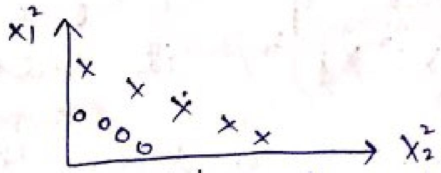
In general dim Reduction using the variance maximization ~~problem~~ objective:
Non linear task (finding Eigen Values).

Kernels went to the other extreme of even increasing the dim (X) if needed.
It uses a mapping ϕ which when applied on X, makes data linearly seperable ~~to~~ which is otherwise linearly non seperable.

ϕ : Feature Map.

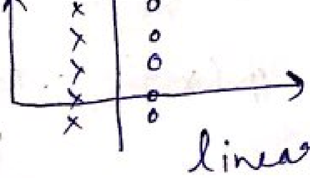
Eg:

linearly non seperable

Lets take the ϕ as: $\phi\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$.

New data:


Linearly seperable

Another ϕ: $\phi\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} r \\ \theta \end{bmatrix}$


linearly seperable with one feature.

The idea behind Kernels is to find a mapping which maps to infinite dimensions. Solve the problem in that space where classification is easy.

We try to find a mapping which makes our data linearly separable; & Use Linear SVM's here. 2 methods to do it;

1) convert the data set using the mapping & then apply linear SVM.

2. directly perform computation in the original space. [preferred].

Eg: $p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$ $q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ $p^T q = p_1 q_1 + p_2 q_2$

$$\phi \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x^2 \\ y^2 \\ \sqrt{2}\, xy \end{bmatrix}$$

$$\phi[p] = \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2}\, p_1 p_2 \end{bmatrix} \quad \phi[q] = \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2}\, q_1 q_2 \end{bmatrix}$$

$\phi[p]^T \phi[q] = (p^T q)^2$. (After some basic maths).

Now everywhere in the SVM's we use dot product. so in the computation,

instead of $X_i^T X_j$, if we replace it with $(X_i^T X_j)^2$, it means that we are solving in the infinite dimensional space.

Explanation: Say we convert

$x_1 \rightarrow \phi(x_1)$
$x_2 \rightarrow \phi(x_2)$ & then

do $\phi(x_1)^T \phi(x_2)$, we get $(X_1^T X_2)^2$.

So it means, we are solving in that space.

Now $K(p, q) = (p^T q)^2$ is an example of a Kernel.

A kernel is just like a function.
(homogeneous polynomial kernel).

Some more examples: $e^{-\gamma x^T y}$, $\tanh(x^T y)$.

Now $(p^T q + 1)^2$ would have a 6 dimensional ϕ. [do this].

Works so well because how many ever dimensions we take, dot product is always a scalar.

Now for kernels like $e^{-\gamma x^T y}$, the ϕ is infinite dimensional. (Proof: expand e^{-ga}).

Kernels combined the computational ease of linear algos and the expressive power of non linear boundaries.

Consider this: $\phi \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \\ x_2 x_1 \end{bmatrix}$

corresponding kernel is still $(p^T q)^2$.

So we can have multiple mappings for a kernel. $\Rightarrow$ $K_{ij} = K.\text{Function}(x_i, x_j)$

- Kernel Matrix $K = [\ K(x_1, x_2)\ ]$ $(N \times N)$

(A). there exists a ϕ if K is PSD (positive semi definite)

Most powerful kernel: Rbf kernel.
(common)

We also need to tune the parameter γ in $e^{-\gamma x_i^T y}$

We need not care about the mapping; just the kernel f/n is important.

We need ~~d~~ not have a nice feature set for our features. As long as we can define a kernel, we can solve the SVM classifier.

Kernel is a similarity function.

Eg: ~~given 2 strings;~~ ~~we~~ say we want to classify strings. We might have a frequency ~~d~~ map feature of size 26. But sequence info is lost. hence find out a good similarity f'n $K(s1, s2)$ and you are done.

If K_1, K_2 are valid kernels, their additin sum $K_1 + K_2$ is also a valid kernel. Using this, we can generate a wide range of kernels. A linear combinatio of K_1, K_2 : valid kernel.

Even if we ↑ the dimensionality, the associ-ated problems do not exist.

So in non linear SVM's the optimization function becomes:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \cdot \underbrace{K(x_i, x_j)}_{K_{ij}}$$

if we pre compute K, the problem becomes a quadratic programming problem.

We did not want to compute W as we want-ed to use a kernel SVM.

Scanned with CamScanner

While classification, we want :

$$\text{sign}\left(\sum_{i=1}^{\#SV} \alpha_i y_i K(X_i, X) + b\right)$$

$\hookrightarrow O((\#SV) \times d)$.

$\ulcorner$ We need to carry all the support vectors α's
$\llcorner\!\to$ storage complexity

If we know the ϕ, we can have $O(d)$ testing by ~~sto~~ precomputing W.

So given a kernel, ppl tried finding a finite ϕ approx which solves the problem.

Lecture

PCA

Let us take centered data ($X - M$ done already)

$x_1, x_2 \dots x_M \in R^N$ ($N \to$ dimensions)

$$C = \frac{1}{M} \sum_{i=1}^{M} x_i x_i^T$$

$\hookrightarrow$ has $\min(N, M)$ eig vectors

$$CV = \lambda V$$

project X on $X^T V_i$.

Let us apply the kernelization on another linear problem: the PCA.

In the computation of C, its a dot product which can be kernelized; The ~~co~~ infinite dimensioned dot product can be simulated but still the eigen vector length would be infinite dimensions. So the problem is not solved yet.

$$C = \frac{1}{M}\sum_{i=1}^{M} \phi(x_i)\ \phi(x_i)^T$$

$$CV = \lambda V$$

$$V = \frac{1}{\lambda} CV$$

$$\Rightarrow \frac{1}{\lambda}\sum_{i=1}^{M} \phi(x_i) \cdot \boxed{\phi(x_i)^T V}$$

scalar

$$\Rightarrow \boxed{V = \sum_{i=1}^{M} \alpha_i\ \phi(x_i)}$$

linear combination of ~~ϕ~~ $\phi(x)$'

Still the same problem persists as ϕ can't be computed.

So now, to find out the new projected vectors, finding eigen vector isn't necessary; In this case,

$$V_j = \sum_{i=1}^{M} \alpha_i\ \phi(X_i)^T \phi(X_j).$$

$$\Rightarrow V_j = \sum_{i=1}^{M} \alpha_i\ K(x_i, x_j)$$

given. we know the α's.

So if the data is follows:

V (Non lin).
Linear Eig Vec $\bar{V}$

No. of α's : M.

$$CV = \lambda V$$

$$\Rightarrow \frac{1}{M}\sum_{j-1}^{M} \phi(X_j)\ \phi(X_j) \sum_{i=1}^{M} \alpha_i\ \phi(X_i)$$

$$= \lambda \sum_{i=1}^{M} \alpha_i\ \phi(X_i)$$

Since we need even number of ϕ's on both sides (to use the Kernel f^n).

Multiply $\phi(x_K)^T$ on both sides

$$\phi(x_K)^T \frac{1}{M} \sum_{j=1}^{M} \phi(x_j) \cdot \phi(x_j)^T \sum_{i=1}^{M} \alpha_i \phi(x_i)$$

$$= \lambda \phi(x_K)^T \sum_{i=1}^{M} \alpha_i \phi(x_i) \quad [\forall x_K)$$

$$\Rightarrow \frac{1}{M} \sum_{j=1}^{M} \phi(x_K)^T \phi(x_j) \sum_{i=1}^{M} \alpha_i \phi(x_j)^T \phi(x_i)$$

$$= \lambda \sum_{i=1}^{M} \alpha_i \phi(x_K)^T \phi(x_i) \, [\forall K]$$

$$\Rightarrow \frac{1}{M} \sum_{j=1}^{M} \underset{\downarrow \atop K}{K_{Kj}} \underset{\downarrow \atop K\bar{\alpha}}{\sum_{i=1}^{M} \alpha_i K_{ji}} = \underbrace{(\lambda M) \sum_{i=1}^{} \alpha_i K_{Ki}}_{\downarrow}$$

if we write this for all the K's

we get $\boxed{(\lambda M) K\bar{\alpha}}$

$$\Rightarrow \boxed{K\alpha = \lambda' \bar{\alpha}}$$

α: Eigen vectors of the Kernel Matrix
So to get α's we compute the eig values

Each α-set gives us one vector in the new space. We can take how many ever we want.

So with this, we could get a non linear projection.

Now, $\left[\phi(x_i) - \frac{1}{M} \sum_i \phi(x_i) \right]$

$$K_{mn} = \phi(x_m)^T \phi(x_n).$$

$$\Rightarrow \left[\phi(x_m)^T - \frac{1}{M} \sum \phi(x_i) \right] \left[\phi(x_n) - \frac{1}{M} \sum \phi$$

$\Downarrow$ If we multiply them, we get the new Kernel matrix in terms of the original kernel Matrix.

$\hookrightarrow$ centralized.

Steps

1. Given $x_1 \dots x_m \in R^N$
2. Compute the kernel matrix K'.
3. Find out the centralized Kernel Matrix from K'. (call it K).
4. Find $\bar{\alpha}$ by $\underbrace{K\bar{\alpha} = \lambda\bar{\alpha}}_{\text{Eig. vectors}}$
5. For ~~each~~ new data element x_t: $\sum_{i=1}^{M} \alpha_i K(x_i, x_t)$

$\Rightarrow$ One thing to note is that we cant discard the original data after the ~~eige~~ α's are calculated; We need it while we calculate projection.

LDA (Fischer's method).

The goal of this method is that, after projection the data should be better separable. Within the class, scatter should decrease. And between classes, the scatter should increase. [Mean of 1 class should be far away from the other].

So the objective is to :

$$S_B = [\mu_A - \mu_B][\mu_A - \mu_B]^T$$

$$S_W = S_A + S_B$$

$$\Rightarrow \sum_{i=1}^{M_A} [x_i - \mu_A][x_i - \mu_A]^T$$

$$+ \sum_{j=1}^{M_B} [x_j - \mu_B][x_j - \mu_B]$$

Let the vector onto which we need to project is U.

then goal is to $\max\limits_{U} \dfrac{U^T S_B U}{U^T S_W U}$.

$S_B \rightarrow$ separation b/w classes

$S_A \rightarrow$ sep. within class

Modified Goal : $\max U^T S_B U$

s.t. $U^T S_W U = 1$.

Applying Lagrangian :

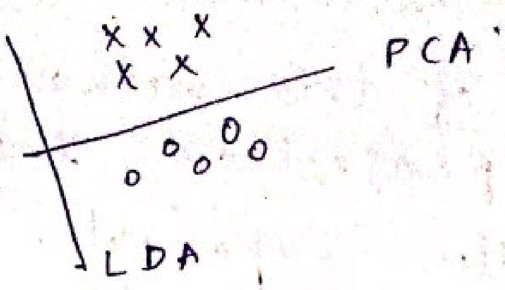
$$\max U^T S_B U - \frac{\lambda}{2}(U^T S_W U - 1).$$

differentiate and equate to 0.

$$S_B U = \lambda S_W U$$

$$\Rightarrow \boxed{S_W^{-1} S_B U = \lambda U}$$

Motivating picture :



[Note : it's not that PCA $\perp$ LDA]

PCA : compression friendly

LDA : separability is enhanced.

Problem here : S_w^{-1} might not exist.
$\hookrightarrow$ No. of samples $\neq$ No. of dimensions

To fix this, we can use the foll:

1). $S_W \leftarrow S_W + \underbrace{e}_{\text{small quantity}} \overbrace{\boxed{I}}^{M \text{ sized.}}$

2). do a PCA and then do LDA.

3). $S_W = \sum_i \sum_j [X_i - X_j][X_i - X_j]^T \Rightarrow$ [Redefine S_W] .

Look at : Generalized Eigen Value problem