⊡ Kernel Methods :

⊙ Kernel - PCA :-

1 → given a set of data x^i, we "invent" a function ("a kernel"),

$$K(x^i, x^j)$$

→ The only requirement on this func^n is that it should be symmetric

→ Sometimes we need it to be positive-definite

→ i.e; $K(x^i, x^j) = K(x^j, x^i)$

→ Then we compute the $N \times N$ Kernel matrix

2 → $K^{ij} \rightarrow \tilde{K}^{ij}$ (we perform double-centering to take our kernel matrix to a gram matrix)

$$K^{ij} = K(x^i, x^j) \; \forall_{i,j}.$$

$$G^{ij} = -\frac{1}{2}\left(K^{ij} - \frac{1}{N}\sum_{l}(K^{il} + K^{jl}) + \frac{1}{N^2}\sum_{lm} K^{lm}\right)$$

double-centering

→ We can prove easily that whatever K we take, assuming its symmetric, the resultant (G^{ij}) must be a Gram matrix

scalar product so not true defined

i.e. $G^{ij} = G^{ji}$ (symmetric)

$\sum_{l} G^{il} = 0 \quad \forall i$

i.e. resultant G^{ij} satisfies the properties of a Gram matrix, due to double-centering.

3→ We perform MDS with G^{ij}

will simply give us a list of real eigenvec.

→ Now how do we get K^{ij}?

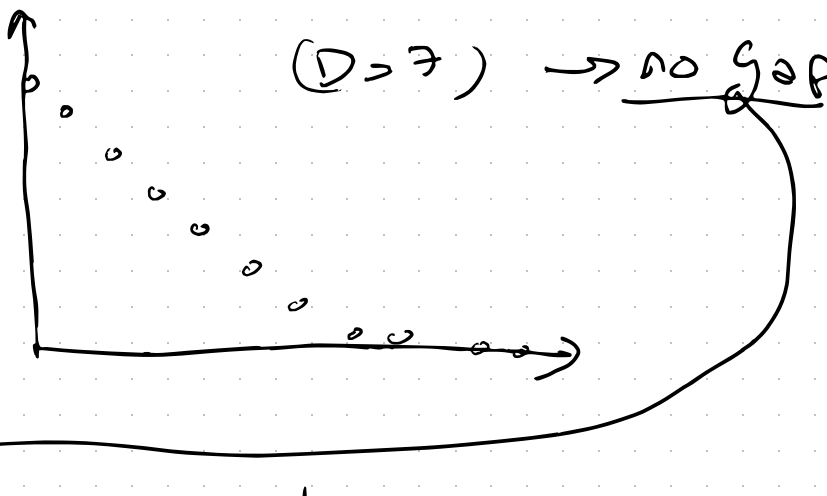
① let's say a null hypothesis / ground state is,

$$K^{ij} = x^i, x^j$$

→ If we do this, we are basically doing PCA.

→ Of course it doesn't make sense to do this ∵ easier to diagonalize C than G.

→ But spectrum would be like (for eg.) :-



So, we are not happy with our choice of kernel.

① So, let's choose a new kernel

$$K^{ij} = x^i \cdot x^j + a(x^i \cdot x^j)^2$$

↳ We compute spectrum again

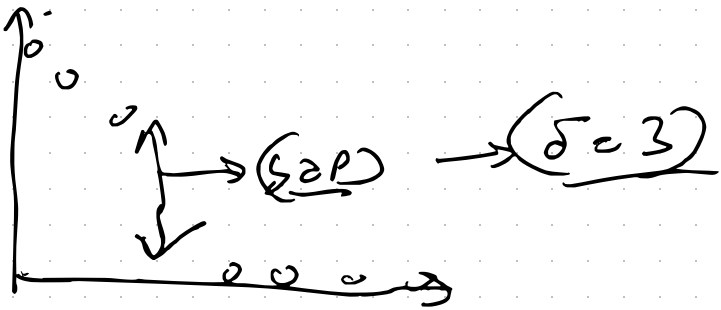
↳ $a \equiv$ variational parameter

↳ Smith $a = 0 \rightarrow$ PCA

$a = 1$

↓

(for eg)



∴ We have found a kernel where 3 eVs of S explain 100% of the variance.

① IL:DR; De choose a K-func in such a way that the spectrum can be truncated as early (for as few eigenvalues) as possible, i.e., spectrum of gram matrix can explain all / target variance for as few eigenvalues as possible.

⇒ For eg, $K_{a,b}(x^i, x^j) \equiv$ kernel func with a additional parameters (a) & (b).

$(\lambda_{a,b})$ ← spectrum must be obtained

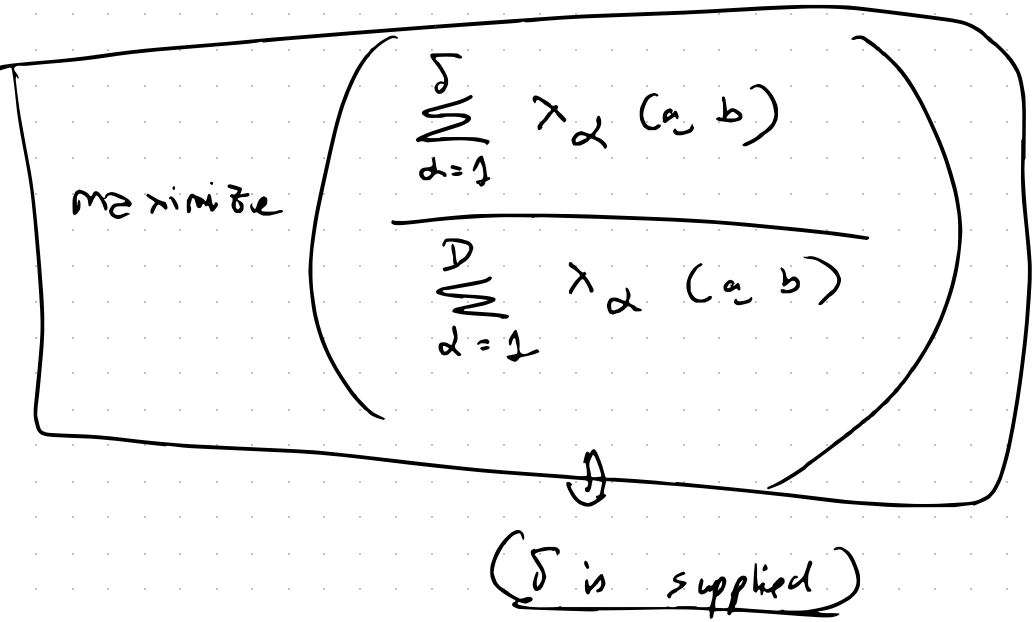
Find (a, b), s.t.

$$\frac{\sum_{\alpha=1}^{3} \lambda_\alpha (a,b)}{\sum_{\alpha=1}^{D} \lambda_\alpha (a,b)}$$

is as large as possible

target embedding dimension, can be diff. from (3)

→ This means we have found a specific kernel that allows a significant dimensional reduction.



→ We introduce the kernel, s.t. that, we can estimate the distances in a manner which is compliant with the features of the embedding manifold.

→ How do we choose the kernel?

(•) Kernel is related to a suitable dist. on the manifold.

(•) (Traditional idea) The kernel defines implicitly a scalar product in an extended feature space.

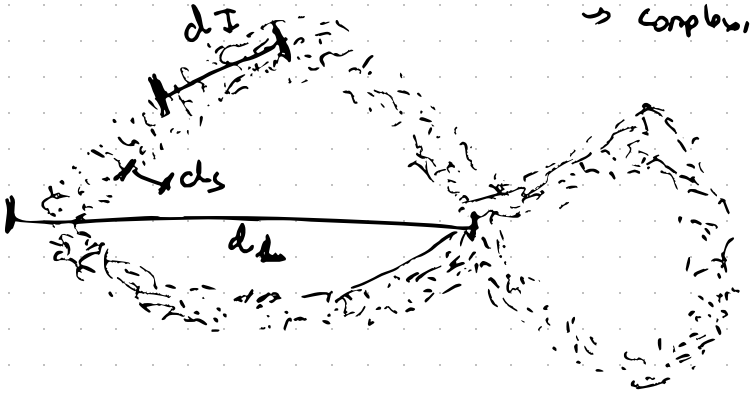
→ Our data manifold is nonlinear (curved) so when we add nonlinear features to represent our data.

(nonlinear features == extended feature space)

(we extend feature space from ordinary cartesian coords to add $\sqrt{x}, x^2$ etc. & other nonlinear features)

Intuitional idea :-

→ noise on small scale
→ curvature on medium scale
→ complexity on topological scale



$d_S \equiv$ small length scale

$d_L \equiv$ large length scale on which there is topological complexity, curvature etc.

$d_I \equiv$ sweet spot length scale (intermediate)
$\equiv$ locally flat manifold is on a hyperplane in this region

so only on this length scale it makes sense to approximate the manifold with a hyperplane.

→ :. Simple way of defining a kernel :-

CLOSE :- irrelevant $|x^i - x^j| < d_S$

∵ we assume fluctuations at this scale are noise

FAR :- irrelevant $|x^i - x^j| > d_L$

∵ if 2 data pts. are so far away, we can't model them so we disregard them,

ACCEPTABLE :- $A^{ij} = 1 - \exp\left(-\frac{\|x^i - x^j\|^2}{2 d_I^2}\right)$

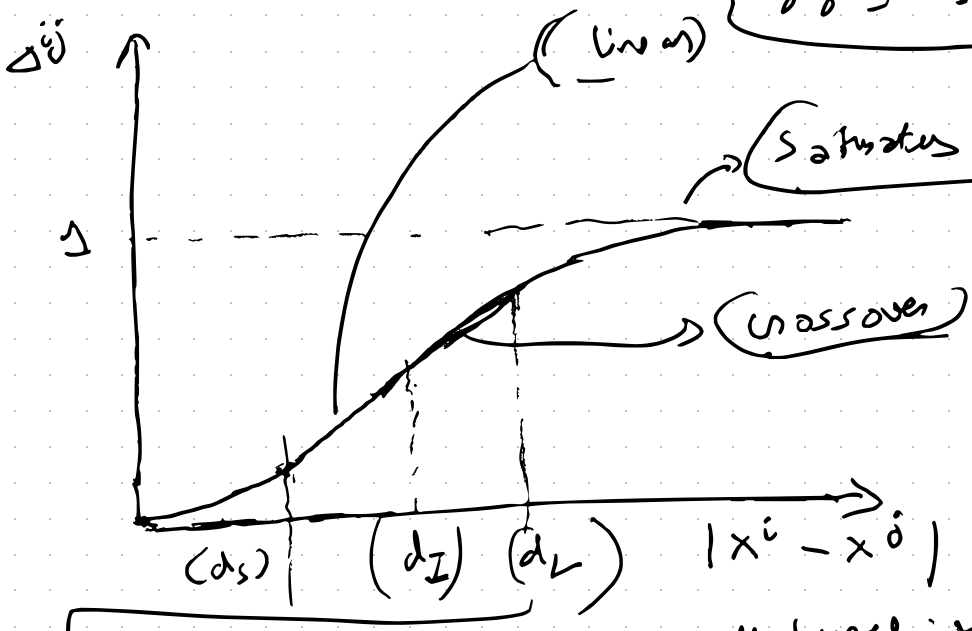
(define a possible length scale)

(Gaussian Kernel) { "1-" not relevant due to double centering

(K is a Gaussian)

(we can keep it, or remove it)

→ all pts. with dist $< d_S$ are lumped into 0 nothing negligible, basically = 0



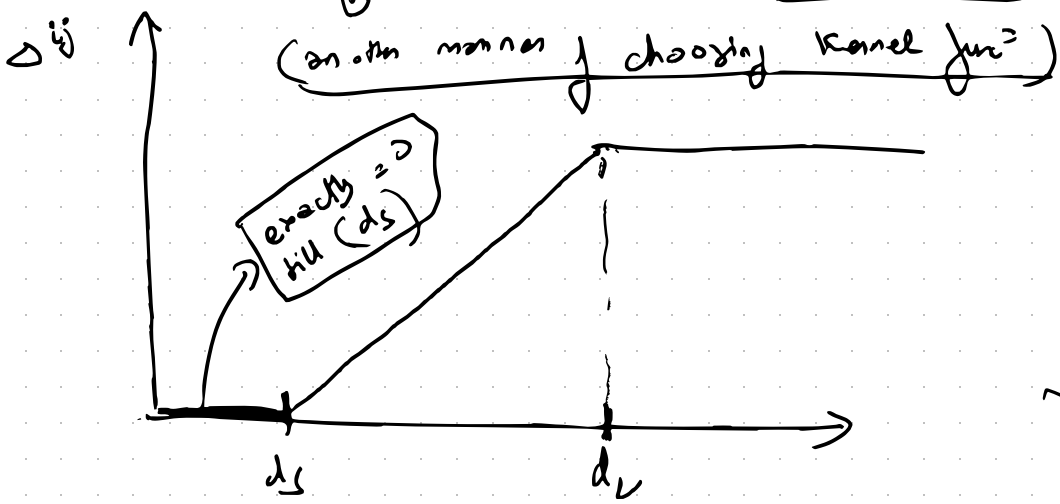
→ pts. that are very far, are all lumped into the same dist., implying "far", here chosen = 1

→ When we use a Gaussian kernel we only have one parameter the variance, so we sort of implicitly define distances on our data manifold, in which the only relevant distances are of order (d_I). So, we have just a single parameter (variance)

⊙ SKETCHMAP :- Case of Kernel-PCA :-

↓

(another manner of choosing kernel func^n)



→ another possible choice of kernel func^n

→ This is called "SKETCHMAP"

→ Principle :- if we have a data manifold that is affected by noise on short-scale & on the long scale is affected by complexity, we only have a suitable intermediate dist. range (d_I) on which we are interested, then the sketchmap is suitable.

↳ this suitable kernel, automatically selects the intermediate dist. for our feature learning / dim. reduction.

⊡ Let's take a step back to MDS :-

→ We find Y^i, s.t. $\|Y^i - Y^j\| \approx \delta_{ij}$

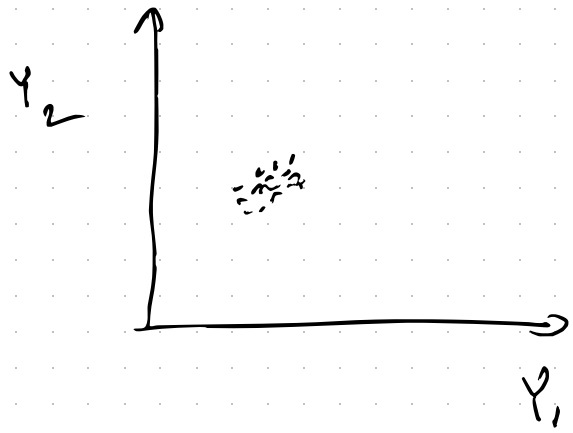
→ Using kernel methods the features that we "learn" will reproduce ONLY intrinsic distance.

→ If we use MDS also, we solve this problem analytically.

→ ∴ we have a lot of rotational freedom.

→ In a very complex topology, if we say everything further than a dist. is the same, then we lose information for sure ∴ we can't recover the original topology. Also getting a dim. reduction that makes sense is much more tricky.

→ This is because (Y^i's) don't even attempt to reproduce properties of our system on the large scale.

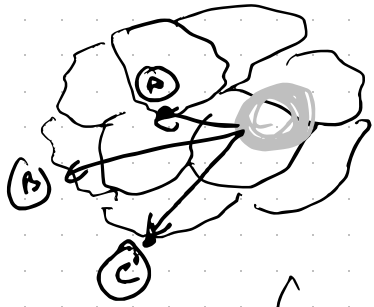


→ in a reduced dim. representation, points that are close in $(Y_2 - Y_1)$ space are not necessarily close in original dataset. ∴ NN's in Y are not necessarily NN's in X & vice-versa.

→ Because if they are closer than a threshold all the dist. are the same.

→ ∴ we have a lot of additional freedom to obtain the intermediate distance.

→ Imagine a complex topology on a large scale.



→ imagine we do MDS on the many of the gray dashed arc.

→ anything smaller than it is considered noise. everything outside is too far & also ignored.

→ In our low dim. projection, our results will not look like our original manifold.

↓

C: dist to A, B, C is treated as exactly the same)

↳ : at a certain scale it is locally compliant with our dataset, but at the large scale, it's not.

→ : we can measure dist. in $\{Y^i\}$ space, but these dist. are relevant ONLY in $(\sim d_I)$ range.

TL; DR:-

→ $k(x^i, x^j) = k(x^j, x^i)$ (any func° that is symmetric in its arguments) is our only requirement.

(•) $k(x^i, x^j) = 1 - \exp\left(-\frac{\|x^i - x^j\|^2}{2\sigma^2}\right)$

↓

(Concept of suitable length scales)

↓

(Gaussian Kernel)

(doubt from before) 26/11/23

(9). We derived double-centering formula considering Δ^{ij} as euclidean dist. Then why does it work with KPCA & ISOMAP?

(A). If $\Delta^{ij} = \|x^i - x^j\|^2$

& $g^{ij} = -\frac{1}{2}\left(\Delta^{ij} - \frac{1}{N} \cdots\right)$

(double centering formula)

Then $g^{ij} = x^i \cdot x^j$ (exactly)

In general, if $\Delta^{ij} = \Delta^{ji} \equiv \Delta^{ij}(x^i, x^j)$ (generic func^n of the coords.)

(i.e. it only satisfies symmetry operation w.r.t the indices $(i \leftrightarrow j)$)

(most general case)

then $\zeta^{ij} \neq x^i, x^j$ (in general)

however ζ^{ij} will satisfy the properties of a gram matrix i.e.:

$$\sum_j \zeta^{ij} = 0, \quad \forall i$$

→ This property allows us to interpret ζ^{ij} as a scalar product between $(z^i \& z^j)$ which are other, newer features NOT $(x^i \& x^j)$

$$\therefore \zeta^{ij} = z^i, z^j \qquad z^i \neq x^i \quad z^j \neq x^j$$

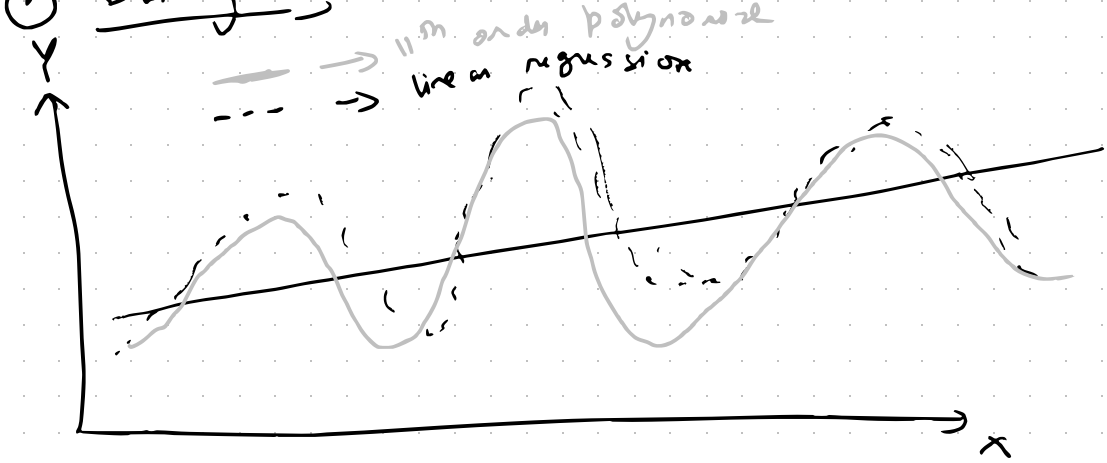
→ N.B.: ζ^{ij} can be assumed to be derived from a kernel.

→ i.e. we can interpret the Gram matrix as a scalar product btw centered features in an extended feature space.

(simply because it's symmetric & $\sum_j \zeta^{ij} = 0$, $\forall i$ is satisfied)

This leads us well into the topic for kernel tricks

But first let's take a step back.



X ≡ master variable

Y ≡ depends on X

→ imagine we try to do linear regression

↳ i.e. we assume $y^i = a + b x^i$

& we try to determine (a) & (b)

s.t. a straight line fits the data.

→ But, this gives us a very bad result & is not a good model. ∴ using linear regression with the original features (just x^i) → we get really bad results.

⇒ So now do we improve?

let's say instead of a linear func^n we try to represent it as a polynomial to increase the basis set.

$$y^i = a + b x^i + c x^{i^2} + \dots$$

→ So if we extend this 'basis' enough, eventually we can reproduce the func^n. (let's say if we include 11 terms).

↳ this is still a linear fit.

⇒ ∴ even for a very complicated func^{n} we can reduce it to a linear regression problem

⇒ That is it doesn't matter how complex the func^{n} is if we accept the cost of extending the feature space introducing nonlinear func^{n}s (x^2, x^3, $\sin(x)$, $\cos(x)$ etc.) then we can basically reproduce any possible func^{n} by linear regression.

⇒ In other words any nonlinear twisted data manifold can in principle be represented by performing a suitable representation in a feature space which includes a sufficient number of nonlinear func^{n}s of the original coordinates.

⇒ We can also build more complex features

Introduce $z^i = y^i$

& now we can build features which are of (x^i, y^i) & still do this process with even more complex feature space.

→ In practice, if we accept to extend our feature space into high dimensions (with nonlinearity), then we can map any data representation into a linear problem.

Statement :- If we extend our feature space introducing extra features which are nonlinear combinations of our original features (x^i), then it is more likely that we will be able to represent our data manifold with a linear regression.

→ For eg, if the manifold is a hyperplane, then original features are for sure sufficient to describe it with linear regression, which is just PCA

→ If the manifold is more complex & NOT a hyperplane, then adding nonlinear features makes it more likely that we can "linearize" our problem.

⇒ So what exactly, in mathematical terms does this mean?

⊙ → Let's say we are in 2-D.

→ original feature space : (x_1^i, x_2^i) for each data point (i)

→ extended feature space :

(x_1^2, x_2^2, x_1, x_2)

↳ so we pass from a representation of our data in D=2 to a " " D=5

↳ we can do PCA in this extended feature space. This way we can find if there are any new emergent linearities as a result of extending the feature space.

→ A (G^{ij}) derived from a kernel by double centering can be interpreted as a scalar product in an extended feature space.

(Gram matrix)

→ Virtually any Gram matrix built by double centering from a symmetric kernel can be interpreted as a scalar product in an extended feature space.

⇒ Eg :-

(1) Polynomial Kernel :-

$$k(x^i, x^j) = \sum_{\alpha} p_\alpha (x^i \cdot x^j)^\alpha$$

(p_α : coefficients; $(x^i \cdot x^j)$: dot product / scalar product)

→ if $P_1 = 1, \quad P_{d \neq 1} = 0$

$$K^{ij} = x^i \cdot x^j$$

(i.e; kernel is just the standard scalar pdt. btw the original features)

→ if $P_1 = 1, \quad P_2 = 1, \quad P_{rest} = 0$:

$$K^{ij} = x^i \cdot x^j + (x^i \cdot x^j)^2$$

if we have feature space = 2 :

then : $(x^i \cdot x^j)^2$

$$= \left(x_1^i x_1^j + x_2^i x_2^j\right)^2$$

$$= \left(x_1^i\right)^2 \left(x_1^j\right)^2 + \left(x_2^i\right)^2 \left(x_2^j\right)^2 + 2\, x_1^i \cdot x_2^i \cdot x_1^j \cdot x_2^j$$

(x_1, x_2)

$$\left(x_1^2, x_2^2, \sqrt{2}\, x_1 x_2\right)$$

→ ∴ our new feature space :-

$$x_1, x_2, z_3 (= x_1^2), z_4 (= x_2^2), z_5 = \sqrt{2} x_1 x_2$$

$$\therefore (x^i, x^j)^2$$

$$= z_3^i z_3^j + z_4^i z_4^j + z_5^i z_5^j$$

$$\therefore k^{ij} = x^i, x^j + (x^i, x^j)^2$$

$$= z^i . z^j$$

..

Done

$$z = (x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad \sqrt{2} x_1 x_2)$$

$\Downarrow$

→ extended feature space

→ z is the new basis & $(x_1, \ldots, \sqrt{2} x_1 x_2)$ are components of the vector

→ even for higher powers, $(x^i \cdot x^j)^3$

etc. we can always group together the terms that depend on (i) & those that depend on (j).

$\Rightarrow \therefore \boxed{K^{ij} = z^i \cdot z^j}$ is always preserved, as long as (z) is defined in a specific manner, only that it will probably contain much more complex features.

$\Rightarrow \therefore \boxed{K^{ij} = \sum_{\alpha} \beta_\alpha (x^i \cdot x^j)^\alpha}$

is a scalar product in an extended feature space.

→ we can extend $\alpha \to \infty$ basically & then choose (β_α) in a specific manner.

⇒ if we do this,

$\therefore \boxed{k^{ij} = f(x^i, x^j)}$

↓

can be any generic func^n that can be Taylor expanded. It can be an exponential / Gaussian / Lorentzian etc.

→ ∴ This can be thought of as a scalar product in an extended feature space.

↓

This is called the "Kernel Trick"

Kernel Trick :- defining a generic, symmetric func^n of my original features & Taylor expand this func^n & this expansion is implicitly the def^n of an extended feature space. ∴ if the kernel is an exp(), then the feature space can be considered infinitely large.

⇒ This is why kernel methods are extremely powerful for regression, dim. reduc², & feature learning.

⇒ if we represent distances with a specific nonlinear kernel, we extend the data representation in a very high dim space where in principle, linear regression becomes meaningful.

For eg,

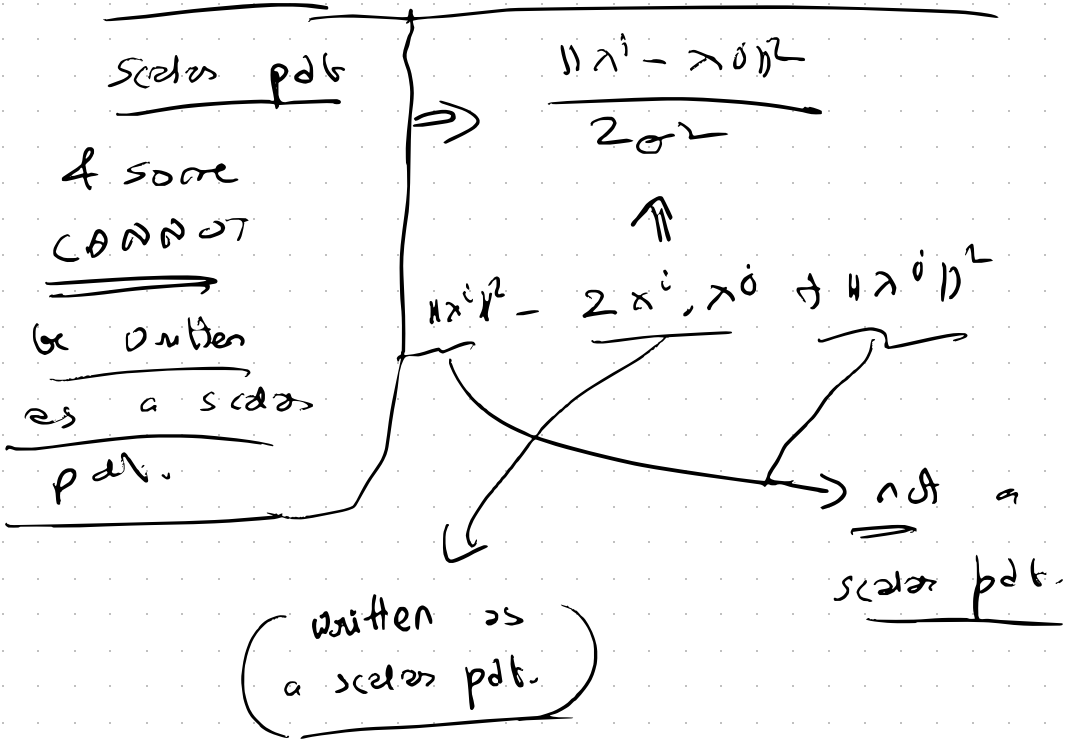
$$K(x^i, x^j) = \exp\left(-\frac{\|x^i - x^j\|^2}{2\sigma^2}\right)$$

$$= 1 - \frac{\|x^i - x^j\|^2}{2\sigma^2} + \frac{\| \quad \|^4}{\dots} + \dots$$

↓

after Taylor exp, we can't write it as a scalar product without double centering

→ Some terms can be written as a scalar pdf & some cannot be written as a scalar pdf.



⇒ TL;DR : a generic Gaussian kernel will NOT be centered.

∴ The features that we try to introduce must be centered.

→ Summary :-

- perform PCA in an extended feature space (can even be ∞ sized feature space)
- diagonalize the Gram matrix :- Covariance matrix in ∞ - feature space can't be diagonalized,
- This allows us to perform linear reg./PCA in this extended feature space.

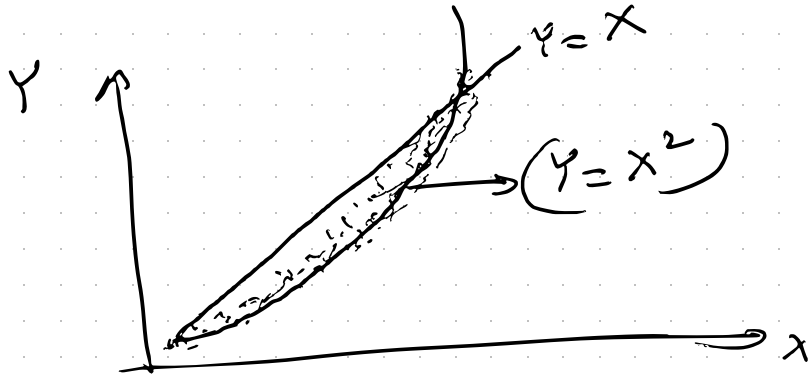
↓

"Kernel Trick"

- Problems :- This kernel trick may NOT always be useful.

  In order to be really useful, we have to select our kernel in a really smart manner.

→ Let's go back to the problem of regression :-



→ having feature space only with (X) is a bad result.

→ extending feature space with (X^2) can be very well. (∵ we have a quadratic func°)

↓

$\therefore Y = a.X + b.X^2$

$a = 2$

$b = 0.3$ (for eg)

→ So if both the features have same weight then our manifold is NOT linear.

→ So basically here our manifold is only linearly we throw away (x) & keep only (x^2).

→ If we can do linear regression using a basis where we have ($x, x^2, x^3 \ldots$) but the coefficients of this linear regression are non-trivial → ($0, 0.7, 0, 0 \ldots$)

⇒ If we choose a generic kernel, this is like representing our data on a basis where the coefficients of the data are fixed.

→ in the polynomial kernel of order (2) we represent our data in the extended feature space: $x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$

→ if we use this kernel & do PCA, we will not get rank = 1 matrix

(on the quadratic data of the previous page)

In order to get rank = 1 matrix, we must remove all features except for (x_1^2).

∴ even in a completely trivial case, while we can do PCA in extended feature space by introducing nonlinear functions, however this doesn't mean we transform our data into a hyperplane. So to get a hyperplane we must get rid of useless features.

$\Rightarrow$ ∴ There is no ONE generic good kernel. So given our data manifold we should trial & error many diff. kernels & stick with the one that gives us the eigenvalue spectrum that can be truncated earliest.

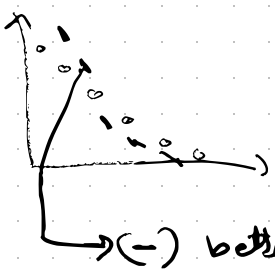
(Subtle trick)

↳ this is for manifold learning, but also applies to kernel regression (kPCA) (in general).

→ TLj DR: (1) Choose a kernel

(2) compute the spectrum

(3) settle on kernel with best performance (captures most variance earliest)



(-) better kernel

$$\Rightarrow \quad \frac{\sum_{\alpha=1}^{n} \lambda_\alpha}{\sum_{\alpha=1}^{D} \lambda_\alpha} = Q$$

('n' chosen dimensions out of 'D' total dimensions; $n < D$)

(Should be as large as possible)

The kernel for which this is max. is the better kernel.

$\Rightarrow$ For a Gaussian Kernel

$$K() = \exp\left(-\frac{\|x^i - x^j\|^2}{2\sigma^2}\right)$$

↳ single unidimensional parameter.

↳ What is correct (σ)?

↳ That which makes (Q) as large as possible for a fixed (n) [no of components]

→ For polynomial kernel we have many more variational parameters to play around with.

N.B. The min. value for (n) is $= ID$.
But it is much safer to set (n) to larger than (ID). We will explore this more in the lectures for ID.

→ If this is our data manifold even if $(ID = 1)$ it is completely impossible to represent it globally in 1-D. So n must be $> ID$ to be safe. i. computing ID is a good guideline to choose (n) but it is just a lower bound.

" sufficiently complex topology makes it safer to choose $(n > ID)$

[D] Graph layout Projection Methods :-

→ these methods come from graph theory.

→ Aim : find best low dimensional representation of a graph.

→ In ISOMAP we build a neighborhood graph & a possible manner of proceeding is computing geodesic dist & then performing MDS.

So if we have our features we build a neighborhood graph & then apply ISOMAP.

⇒ However there are some methods that were taken from graph theory & have become very popular in explicit dimensional reduction.

⇒ There are 3 different methods :

(1) SNE (Stochastic Neighbourhood Embedding) → Hinton, Roweis (2003)

(2) t-SNE (t-distributed SNE) :- Hinton et al

(3) UMAP :- (Mc Innes, Healy, Melville) (2018)

↳ new standard for producing 2D graphs from very high dim. data

→ $x^i \equiv$ original feature space

→ instead of building a neighborhood graph in a deterministic manner like in IsomAP,

→ we assume links between 2 nodes of the graphs are random variables which exist or not with a probability that depends on their neighbours

→ $\therefore \exists$ link btw (i) AND (j)

↳ either 0 (doesn't exist) or 1 (exists) — binary event

$$\frac{k^{ij}}{\sum_j k^i} = p_{i|j}$$

(transition from a binomial distrib)

→ more explicitly

→ prob. to observe link b/w (i) & (j) given (j) $\equiv p_{i|j}$

→ $\sum_j K^{ij} \equiv$ normalizing over the 2^M index

$$\therefore p_{i|j} = \frac{K^{ij}}{\sum_j K^{ij}}$$

∴ these events are parameterized by the kernel (K^{ij}) that is normalized & invested from a binomial prob. distrib?

→ simplest choice for the kernel is Gaussian

$$\therefore K^{ij} = \exp\left(-\frac{\|x^i - x^j\|^2}{2\sigma^2}\right)$$

→ One of the biggest differences b/w SNE / t-SNE / UMAP is how σ is chosen.

→ In UMAP σ is dependent on the data points & is somehow dependent on the distance from the (k^{th}) nearest neighbours of (i).

$\therefore \sigma \to \sigma_i \approx \| x^i - x^{k\ N.N.\ of\ i} \|$

→ ∴ We end up with a Gaussian kernel whose width depends on how dense is our neighbourhood.

→ ∴ we get $p_{i|j}$ (prob. of existence of links) from this kernel, it means the data pts that are closer than k N.N. have a high prob of a link.

→ in data pts that are farther than the k^{th} NN the prob of a link decreases exponentially.

→ In all these 3 methods the general idea is to define another kernel in a feature space of smaller dimensions.

R^{ij} =

(Original dim kernel)

↓

lives in a lower dim. space defined by coords. $\{y^i\}$

$\exp\left(\frac{-\|y^i - y^j\|^2}{2\sigma^2}\right)$

(Gaussian)

↓

(SNE)

$\frac{1}{1 + \|y^i - y^j\|^2}$

(t-SNE)

↓

(t-Student distrib^n)

$\frac{1}{1 + a \cdot \|y^i - y^j\|^{2b}}$

(UMAP)

→ UMAP has 2 extra meta parameters (a & b) that determines how our distrib^n decays.

→ In all these 3 cases, (comes from stochastic links harvested from a k-nearest neighbor choose)

① $X = P_{ij}$ (joint prob. from cond^l prob.)

(prob. in original high dim. space)

② $Y = \Pi_{ij}$ (built with the R^{ij} kernel)

(prob. in lower dim. space)

→ we want $P_{ij} \approx \Pi_{ij}$:-

→ i.e. we minimize the (KL) divergence to have these 2 sets of prob as close to each other as possible to justify our dim. reduc^n.

$$KL(P \| \pi) = \sum_{ij} P_{ij} \ln \frac{P_{ij}}{\pi_{ij}}$$

(KL divergence) [explained more in NN class]

→ let's find $\tilde{Y}$ such that

$$\pi_{ij}(\tilde{Y}) = P_{ij}, \quad \ln 1 = 0$$

$\therefore KL(P|\pi) = 0$

→ $KL(P|\pi)$ is positive-defined & (lower bound is $= 0$). ∴ to minimize $KL(P|\pi)$, we find the probs $(\pi^{(ij)})$ that match as close as possible with $(P^{(ij)})$.

→ TL;DR :

(i) your data is connected by stochastic links.

(ii) these links are thought of as harvested from a prob. distrib^n (chosen by us, typically Gaussian)

(iii) then we try to reproduce the original prob. with a model that is functionally identical but in a lower dim. space.

(iv) we have freedom to choose the kernel & diff. methods differ on the exact functional form of these probs.

→ So far we have seen :

- PCA
- MDS
  - → Isomap
  - → kernel PCA
- SNE (Stochastic Neighbourhood Embedding)
  - → UMAP

⇒ Desired qualities of US2 techniques :

① significant dimensional reduction.

$$\mathbb{R}^D \to \mathbb{R}^\delta, \quad (\delta < D)$$

$\delta \geq ID$ (intrinsic dimension)

→ If we want/need data visualization we need $\delta = 2$ / 3 but really even $\delta = 3$ is a problem, $\delta = 2$ is ideal.

→ If $(ID > 2)$, it'll be impossible to obtain a good visualization of our data.

(2) Cin the case that we want to use our reduced dim. representⁿ for further analysis → typical case

If the neighbourhood of reduced representations should NOT be in consistent with original neighbourhood

If That is points close in X should also be close in Y.

→ This requirement can be at odds with topology
for eg, if all pts. are on a sphere & we want to represent it on a plane.
→ Such an attempted representation would violate this requirement.

(3) Interpretability of Y coordinates.

↓

In PCA, Y vars are linear combos of original variables, so we can interpret that our PCs are domain-wise influenced by so & so features.

In other methods, Y coords are either very complex funcⁿs of original coords, or like in SNE, they are stochastic variables learned on the original data. So in SNE interpretability is garbage.

	0	1	2	3
PCA	✗	✗ ↔	✓	✓
MDS - ISOMAP	✓	— ↔	—	✗ ↓
MDS - kPCA	✓	✗ ↔	✓	✗ ↓
SNE - UMAP	✓	✓	✗	✗

✗ ↓ ≡ doesn't have the property but to a lesser extent than (✗)

For eg, kPCA for (3) : hard to interpret unless specific kernels make a huge diff in results that allows us to understand something. NOT as hard as SNE.

→ There is also a desired property.

→ (D) It should work great on complex, non-trivial manifolds (which are very curved & twisted).

⇒ (D) & (3) are sort of inversely correlated.

⇒ In PCA & kPCA :-

→ (1) & (2) are anticorrelated,

→ (2) maintains topology & if we want a very strong dim. reduction, we destroy the topological properties.

→ So, if (2) is good [i.e. PCA & kPCA has great topology preservation], (1) is bad & vice-versa.

→ if we include # of variables that describe our variance sufficiently (2) is preserved, i.e. we only do a rotation of our representation & throw away something which is small, so (2) is well maintained

→ ISOMAP : horrible for preserving topology, i.e. we unfold the manifold.

We observe the same anticorrelation behaviour as in PCA & kPCA.

→ if we perform Isomap on a spherical manifold, we are in serious trouble if we unwrap sphere into something = ID

⇒ if we use a representation b Isomap > ID we can still find a reasonable sol^n (hence ∈ symbol in the table)

SNE → UMAP :- By def^n (1) is great ∵ we usually use it for visualization (by def^n), of course, this violates (2) is horrible. These methods are state of the art for obtaining 2D representation

⇒ Except for in PCA (3) (interpretability) is horrible for all other methods. This is ∵ $\{y^i\}$ is not (explicit & differentiable) func^n of original $\{x^i\}$.

→ We can't bias MD by coords from Isomap/SNE etc.

↓

First major advantage of NNs.

∴ in MD we need reduced dim representation to be explicit & differentiable func^n of coords ∵ we want to compute forces.