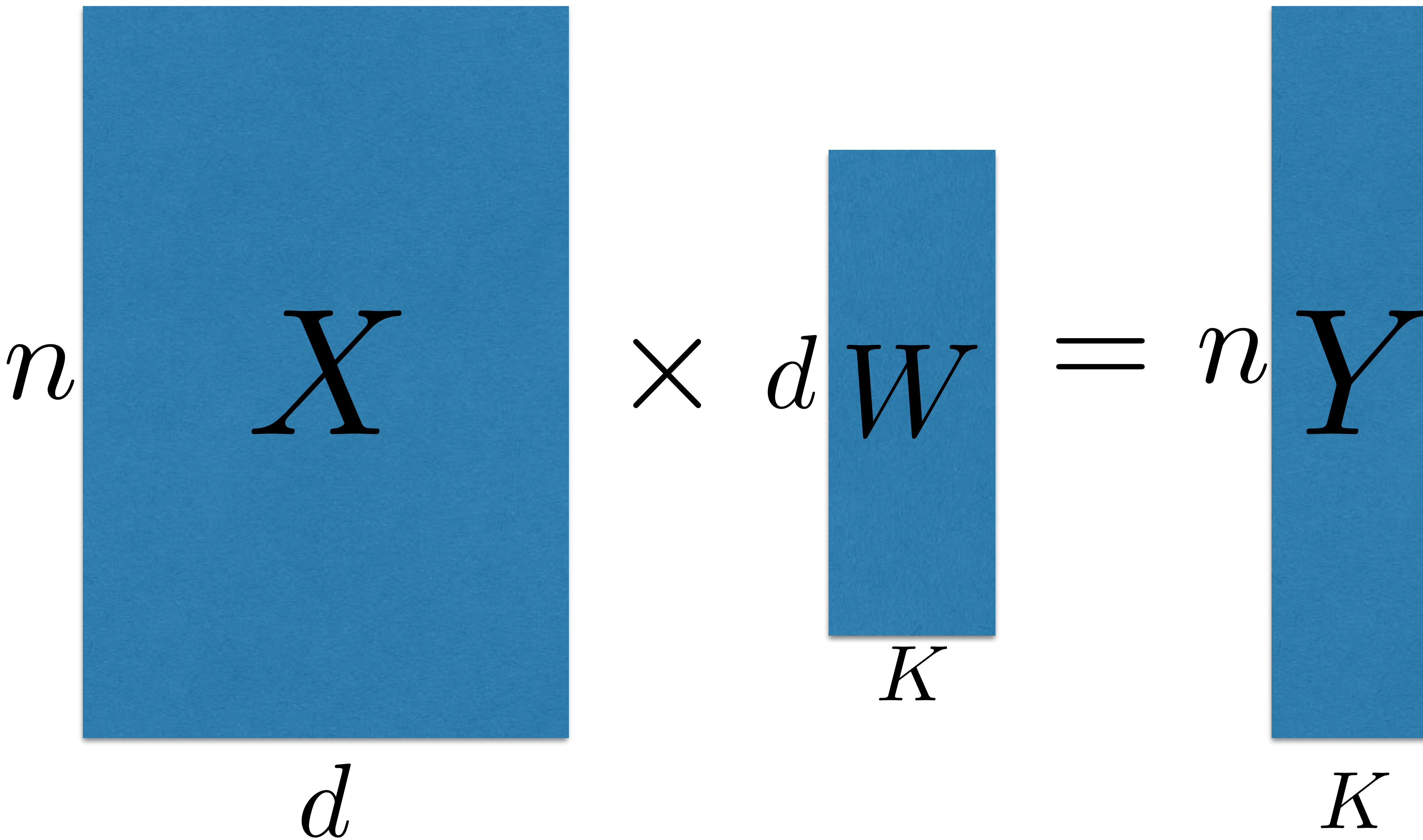
Machine Learning for Data Science (CS4786)
Lecture 8

Kernel PCA
&
Isomap + TSNE

Linear Projections



Works when data lies in a low dimensional linear sub-space

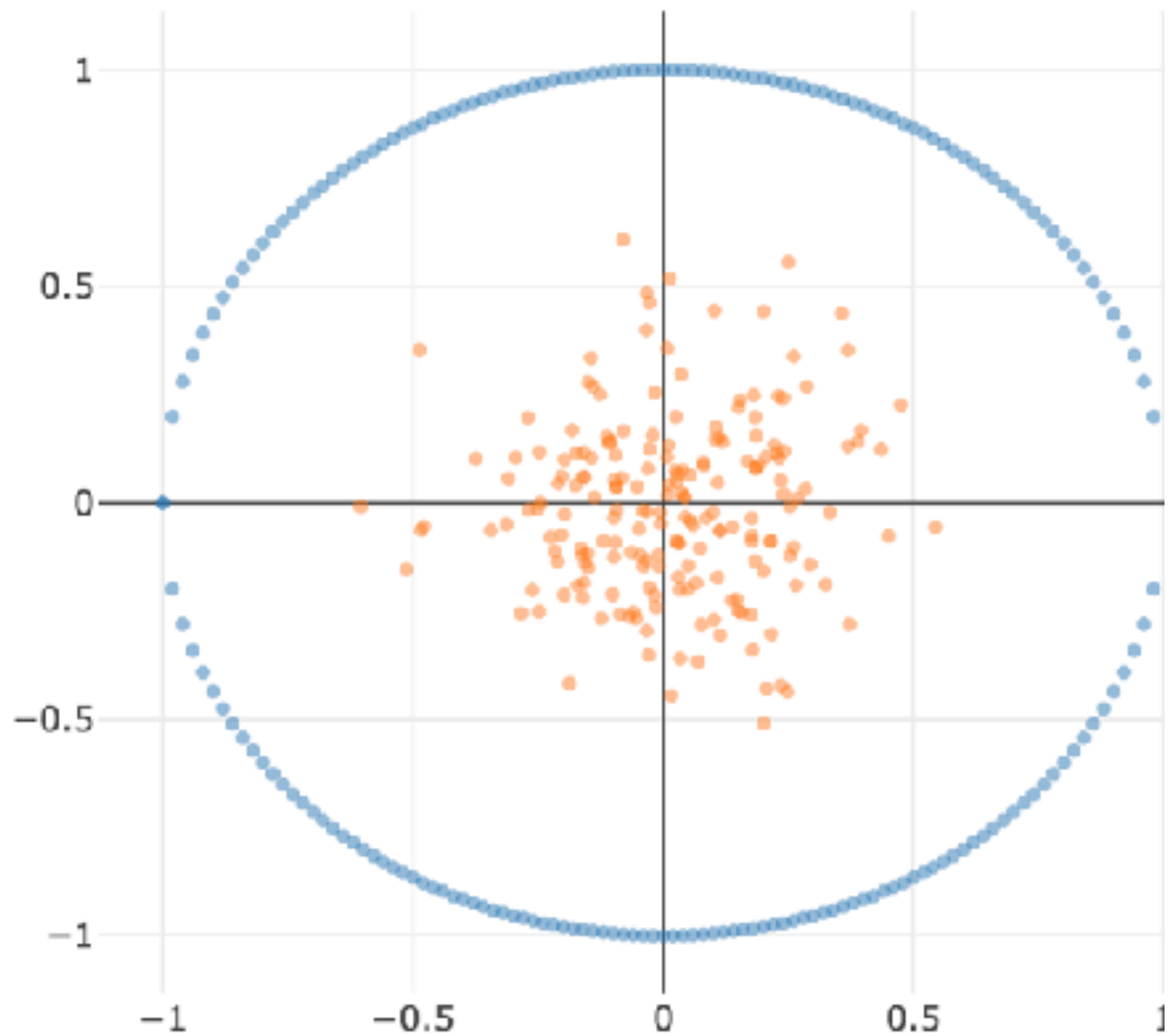
Kernel Trick

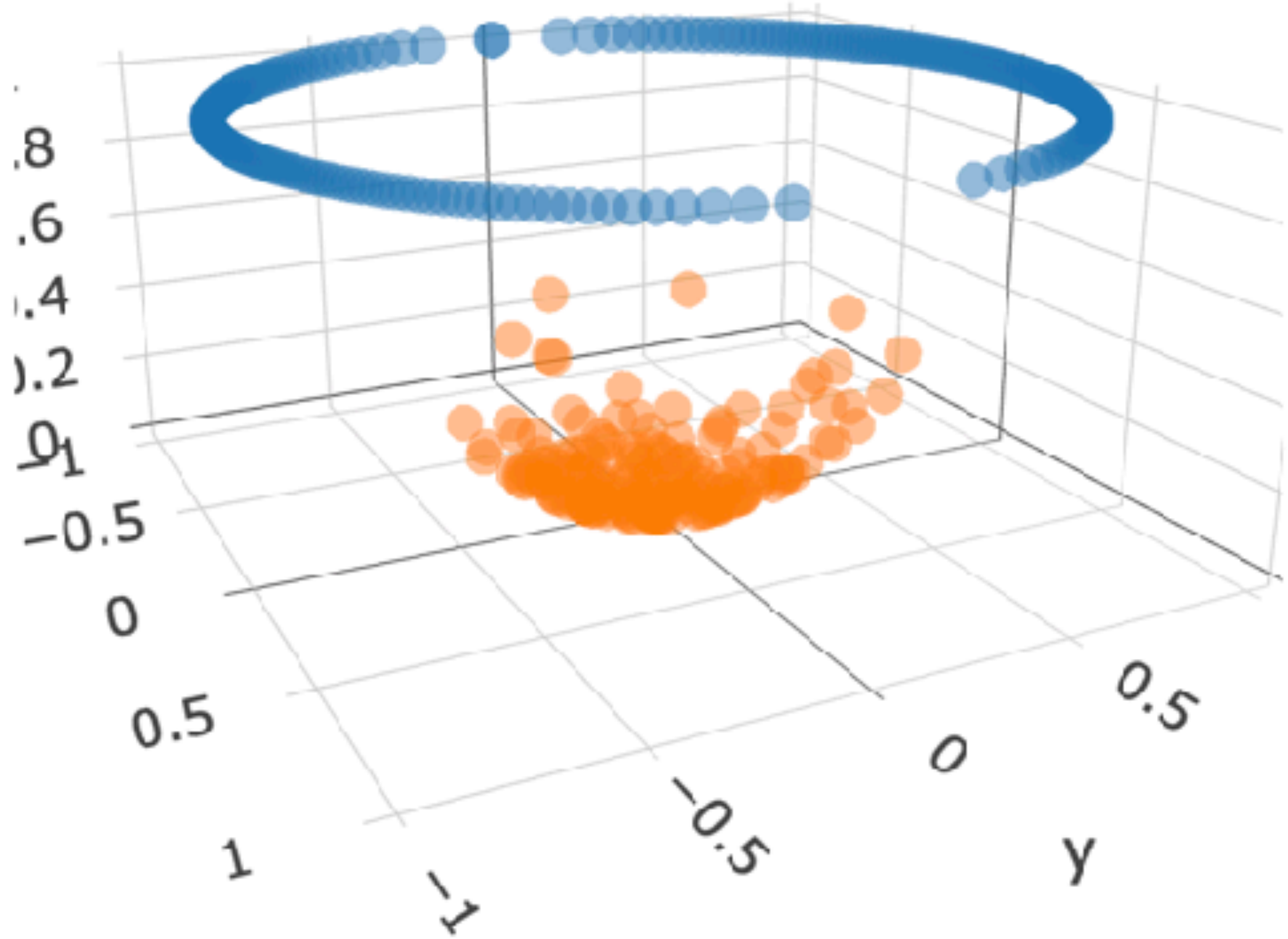
- We have have nice methods for linear dimensionality reduction
- Can we use this beyond the linear realm?

Kernel Trick

- Lift to higher dimensions (introduces non-linearity)
- Perform linear dimensionality reduction in this high dimensional space

EXAMPLE





Original Data in 2D

(x,y)

Data Lifted to 3D

(x, y, x^2 + y^2)

A First Cut

- Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

$$\Phi(\mathbf{x}_t) = (\mathbf{x}_t[1], \ldots, \mathbf{x}_t[d], \mathbf{x}_t[1] \cdot \mathbf{x}_t[1], \mathbf{x}_t[1] \cdot \mathbf{x}_t[2], \ldots, \mathbf{x}_t[d] \cdot \mathbf{x}_t[d], \ldots)^\top$$

- Enumerating products up to order K (ie. products of at most K coordinates) we can get degree K polynomials.
- However dimension blows up as d^K
- Is there a way to do this without enumerating Φ?

Kernel Trick

- Essence of Kernel trick:
  - **If** we can write down an algorithm only in terms of $\Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ for data points $\mathbf{x}_t$ and $\mathbf{x}_s$
  - **Then** we don't need to explicitly enumerate $\Phi(\mathbf{x}_t)$'s but instead, compute $k(\mathbf{x}_t, \mathbf{x}_s) = \Phi(\mathbf{x}_t)^\top \Phi(\mathbf{x}_s)$ (even if Φ maps to infinite dimensional space)
- Example: RBF kernel $k(\mathbf{x}_t, \mathbf{x}_s) = \exp(-\sigma\|\mathbf{x}_t - \mathbf{x}_s\|_2^2)$, polynomial kernel $k(\mathbf{x}_t, \mathbf{x}_s) = \left(\mathbf{x}_t^\top \mathbf{y}_t\right)^p$
- Kernel function measures similarity between points.

Kernel Trick

$$
\begin{aligned}
(\mathbf{x}_t^\top \mathbf{y}_t)^p &= \sum_{k_1+k_2+\ldots+k_d=p} \binom{p}{k_1, k_2, \ldots, k_d} \prod_{j=1}^{d} (x_t[j] y_t[j])^{k_j} \\
&= \sum_{k_1+k_2+\ldots+k_d=p} \left(\sqrt{\binom{p}{k_1, k_2, \ldots, k_d}} \prod_{j=1}^{d} x_t[j]^{k_t} \right) \cdot \left(\sqrt{\binom{p}{k_1, k_2, \ldots, k_d}} \prod_{j=1}^{d} y_t[j]^{k_j} \right)
\end{aligned}
$$

$$
\Phi(\mathbf{x})^\top = \left(\ldots, \sqrt{\binom{p}{k_1, k_2, \ldots, k_d}} \prod_{j=1}^{d} x_t[j]^{k_t}, \ldots \right)_{k_1+k_2+\ldots+k_d=p}
$$

Key Idea:

If an algorithm only depends on inner products, we can simply replace inner product in x space by inner product in $\phi(x)$ space

Can we write PCA so it only depends on inner products?

LETS REWRITE PCA

Lets start with the assumption that Data is centered! (i.e. Sum of x_t's is 0)

- k^{th} column of W is eigenvector of covariance matrix
  That is, $\lambda_k W_k = \Sigma W_k$. Rewriting, for centered X

$$\lambda_k W_k = \frac{1}{n}\left(\sum_{t=1}^{n} \mathbf{x}_t \mathbf{x}_t^\top\right) W_k = \frac{1}{n}\sum_{t=1}^{n} \left(\mathbf{x}_t^\top W_k\right)\mathbf{x}_t$$

But $\mathbf{x}_t^\top W_k = \mathbf{y}_t[k]$

$$\lambda_k W_k = \frac{1}{n}\sum_{t=1}^{n} \mathbf{y}_t[k]\mathbf{x}_t$$

Lets Rewrite PCA

$$
\begin{aligned}
\mathbf{y}_s[k] &= W_k^\top \mathbf{x}_s \\
&= \frac{1}{\lambda_k}\left(\frac{1}{n}\sum_{t=1}^{n} \mathbf{y}_t[k]\mathbf{x}_t\right)^\top \mathbf{x}_s \\
&= \frac{1}{n\lambda_k}\sum_{t=1}^{n} \mathbf{y}_t[k]\mathbf{x}_t^\top \mathbf{x}_s \\
&= \frac{1}{n\lambda_k}\sum_{t=1}^{n} \mathbf{y}_t[k]\tilde{K}_{s,t}
\end{aligned}
$$

Where $\tilde{K}_{s,t} = \mathbf{x}_t^\top \mathbf{x}_s$ is the kernel matrix for centered data

LETS REWRITE PCA

- Hence, the k'th column on Y matrix is such that

$$\mathbf{y}[k] = \frac{1}{n\lambda_k}\mathbf{y}[k]\tilde{K}$$

Also we have, $1 = \|W_k\|^2 = \frac{1}{\lambda_k^2 n^2}\left(\sum_{t=1}^{n}\mathbf{y}_t[k]\mathbf{x}_t\right)^\top\left(\sum_{s=1}^{n}\mathbf{y}_s[k]\mathbf{x}_s\right)$

$$= \frac{1}{\lambda_k^2 n^2}\sum_{t=1}^{n}\sum_{s=1}^{n}\mathbf{y}_s[k]\mathbf{x}_s^\top\mathbf{x}_t\ \mathbf{y}_t[k]$$

$$= \frac{1}{\lambda_k^2 n^2}\mathbf{y}[k]\tilde{K}\mathbf{y}[k]^\top = \frac{1}{n\lambda_k}\|\mathbf{y}[k]\|^2$$

Hence $P_k = \mathbf{y}[k]/\sqrt{n\lambda_k}$ is an eigenvector of $\tilde{K}$ with eigen value $\gamma_k = n\lambda_k$

Rewritting PCA

- We assumed centered data, what if its not,

$$
\begin{aligned}
\tilde{K}_{s,t} &= \left(\mathbf{x}_t - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u)\right)^\top \left(\mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right) \\
&= \mathbf{x}_t^\top \mathbf{x}_s - \left(\frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right)^\top \mathbf{x}_s - \left(\frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u\right)^\top \mathbf{x}_t \\
&\quad + \frac{1}{n^2}\left(\sum_{u=1}^{n}\mathbf{x}_u\right)^\top \left(\sum_{v=1}^{n}\mathbf{x}_v\right) \\
&= \mathbf{x}_t^\top \mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u^\top \mathbf{x}_s - \frac{1}{n}\sum_{u=1}^{n}\mathbf{x}_u^\top \mathbf{x}_t + \frac{1}{n^2}\sum_{u=1}^{n}\sum_{v=1}^{n}\mathbf{x}_u^\top \mathbf{x}_v
\end{aligned}
$$

Rewriting PCA

- Equivalently, if Kern is the matrix ($\mathrm{Kern}_{t,s} = x_t^\top x_s$),

$$\tilde{K} = \mathrm{Kern} - \frac{(\mathbf{1}_{n\times n} \times \mathrm{Kern})}{n} - \frac{(\mathrm{Kern} \times \mathbf{1}_{n\times n})}{n} + \frac{(\mathbf{1}_{n\times n} \times \mathrm{Kern} \times \mathbf{1}_{n\times n})}{n^2}$$

Kernel PCA

1.

n

$$\text{Kern} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ k(x_{n-1}, x_1) & k(x_{n-1}, x_2) & \dots & k(x_{n-1}, x_n) \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}$$

n

2.

n

$$\tilde{K} = \text{Kern} - \frac{1}{n}\left(\mathbf{1}\ \text{Kern} + \text{Kern}\ \mathbf{1}\right) + \frac{1}{n^2}\mathbf{1}\ \text{Kern}\ \mathbf{1}$$

n

Kernel PCA

3. $\left[\, {}^{n}\underbrace{P}_{K}, \gamma \right] = \mathrm{eigs}\left(\tilde{K}, K\right)$

4. ${}^{n}\underbrace{Y}_{K} = \underbrace{\begin{bmatrix} P_1\sqrt{\gamma_1} & P_K\sqrt{\gamma_K} \end{bmatrix}}_{K}$

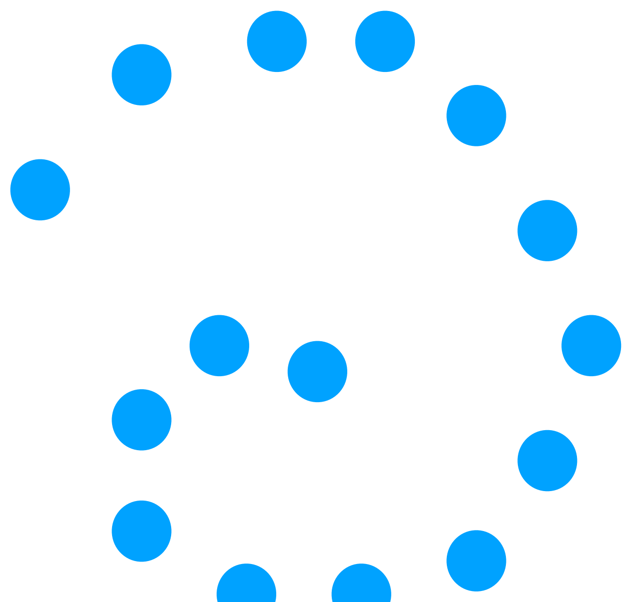
Demo

Kernel Methods: A note

- We can kernelize CCA and any other linear dimensionality reduction method.
- For any linear method, solution lies within linear span of data
- Hence y's can be computed only based on inner products.

Manifold Based Dimensionality Reduction

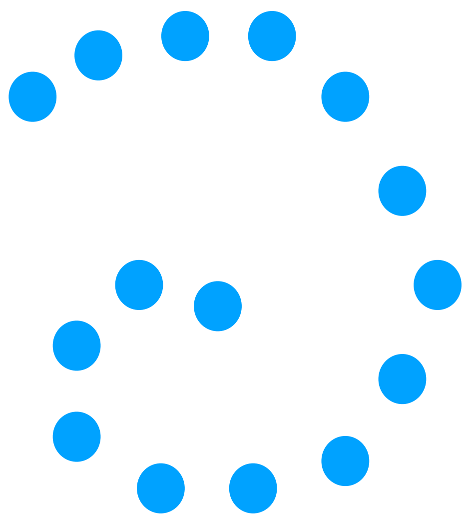
- Key Assumption: Points live on a low dimensional manifold
- Manifold: subspace that looks locally Euclidean
- Given data, can we uncover this manifold?



Can we unfold this?

Method I: Isomap

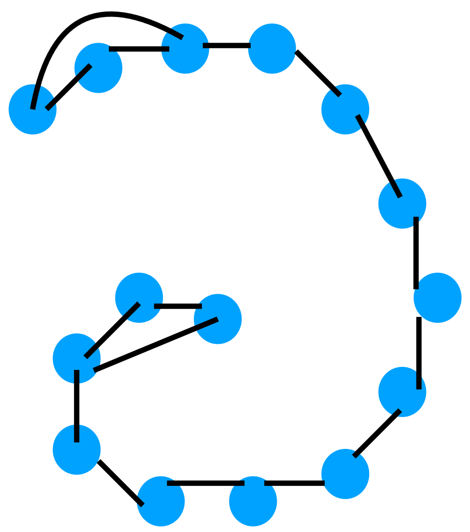
1. For every point, find its (k-) Nearest Neighbors



METHOD I: ISOMAP

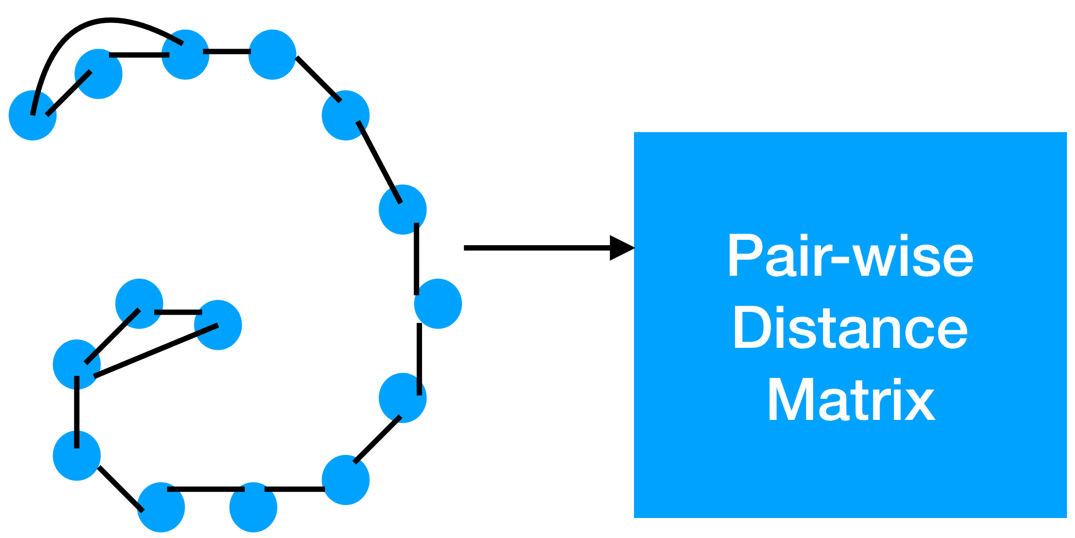
1. For every point, find its (k-) Nearest Neighbors

2. Form the Nearest Neighbor graph



Method I: Isomap

1. For every point, find its (k-) Nearest Neighbors
2. Form the Nearest Neighbor graph
3. For every pair of points A and B, distance between point A to B is shortest distance between A and B on graph



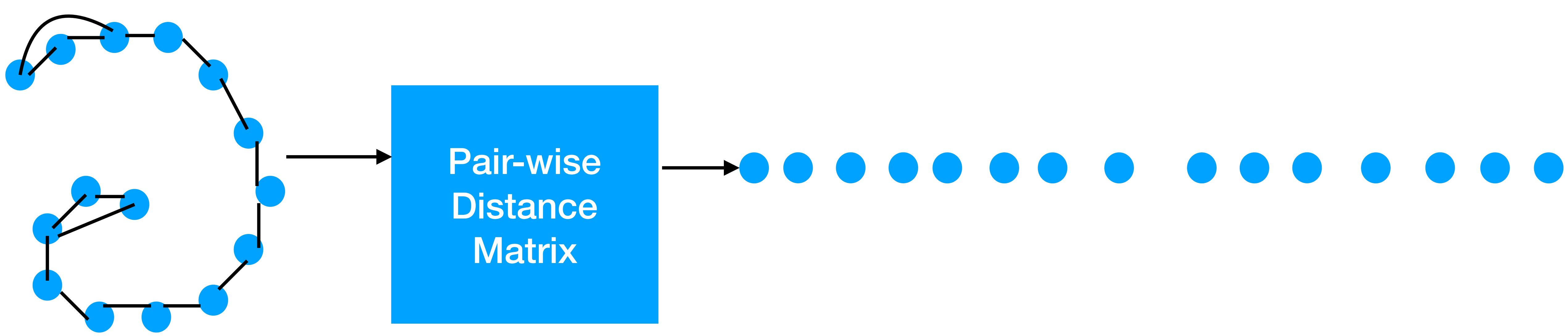
Method I: Isomap

1. For every point, find its (k-) Nearest Neighbors

2. Form the Nearest Neighbor graph

3. For every pair of points A and B, distance between point A to B is shortest distance between A and B on graph

4. Find points in low dimensional space such that distances between points in this space is equal to distance on graph.



ISOMAP: PITFALLS

1. If we don’t take enough nearest neighbors, then graph may not be connected
2. If we connect points too far away, points that should not be connected can get connected
3. There may not be a right number of nearest neighbors we should consider!

Stochastic Neighborhood Embedding

- Use a probabilistic notion of which points are neighbors.

Stochastic Neighborhood Embedding

- Use a probabilistic notion of which points are neighbors.

- Close by points are neighbors with high probability, ...
  Eg: For point $\mathbf{x}_t$, point $\mathbf{x}_s$ is picked as neighbor with probability

$$p_{t \to s} = \frac{\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_t\|^2}{2\sigma^2}\right)}{\sum_{u \neq t} \exp\left(-\frac{\|\mathbf{x}_u - \mathbf{x}_t\|^2}{2\sigma^2}\right)}$$

  Probability that points s and t are connected $P_{s,t} = P_{t,s} = \frac{p_{t \to s} + p_{s \to t}}{2n}$

Stochastic Neighborhood Embedding

- Use a probabilistic notion of which points are neighbors.
- Close by points are neighbors with high probability, ...
  Eg: For point $\mathbf{x}_t$, point $\mathbf{x}_s$ is picked as neighbor with probability

$$p_{t\to s} = \frac{\exp(-\frac{\|\mathbf{x}_s-\mathbf{x}_t\|^2}{2\sigma^2})}{\sum_{u\neq t}\exp(-\frac{\|\mathbf{x}_u-\mathbf{x}_t\|^2}{2\sigma^2})}$$

  Probability that points s and t are connected $P_{s,t} = P_{t,s} = \frac{p_{t\to s}+p_{s\to t}}{2n}$
- Goal: Find $\mathbf{y}_1, \ldots, \mathbf{y}_n$ with stochastic neighborhood distribution Q such that "P and Q are similar"

Stochastic Neighborhood Embedding

- Use a probabilistic notion of which points are neighbors.
- Close by points are neighbors with high probability, ...
  Eg: For point $\mathbf{x}_t$, point $\mathbf{x}_s$ is picked as neighbor with probability

$$p_{t\to s} = \frac{\exp\left(-\frac{\|\mathbf{x}_s - \mathbf{x}_t\|^2}{2\sigma^2}\right)}{\sum_{u\neq t}\exp\left(-\frac{\|\mathbf{x}_u - \mathbf{x}_t\|^2}{2\sigma^2}\right)}$$

  Probability that points s and t are connected $P_{s,t} = P_{t,s} = \frac{p_{t\to s}+p_{s\to t}}{2n}$

- Goal: Find $\mathbf{y}_1, \ldots, \mathbf{y}_n$ with stochastic neighborhood distribution Q such that "P and Q are similar"

  i.e. minimize:

$$\mathrm{KL}(P\|Q) = \sum_{s,t} P_{s,t}\log\left(\frac{P_{s,t}}{Q_{s,t}}\right) = \sum_{s,t} P_{s,t}\log\left(P_{s,t}\right) - \sum_{s,t} P_{s,t}\log\left(Q_{s,t}\right)$$

CHOICE FOR Q

- Just like we defined P, we can define Q for a given $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

$$q_{t \to s} = \frac{\exp\left(-\frac{\|\mathbf{y}_s - \mathbf{y}_t\|^2}{2\sigma^2}\right)}{\sum_{u \neq t} \exp\left(-\frac{\|\mathbf{y}_u - \mathbf{y}_t\|^2}{2\sigma^2}\right)}$$

and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

Choice for Q

- Just like we defined P, we can define Q for a given $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

$$q_{t\to s} = \frac{\exp(-\frac{\|\mathbf{y}_s-\mathbf{y}_t\|^2}{2\sigma^2})}{\sum_{u\neq t}\exp(-\frac{\|\mathbf{y}_u-\mathbf{y}_t\|^2}{2\sigma^2})}$$

  and then set $Q_{s,t} = \frac{q_{t\to s}+q_{s\to t}}{2n}$

- However we are faced with the crowding problem:

Choice for Q

- Just like we defined P, we can define Q for a given $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

$$q_{t \to s} = \frac{\exp(-\frac{\|\mathbf{y}_s - \mathbf{y}_t\|^2}{2\sigma^2})}{\sum_{u \neq t} \exp(-\frac{\|\mathbf{y}_u - \mathbf{y}_t\|^2}{2\sigma^2})}$$

  and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

- However we are faced with the crowding problem:
  - In high dimension we have a lot of space, Eg. in d dimension we have $d + 1$ equidistant point

Choice for Q

- Just like we defined P, we can define Q for a given $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

$$q_{t \to s} = \frac{\exp(-\frac{\|\mathbf{y}_s - \mathbf{y}_t\|^2}{2\sigma^2})}{\sum_{u \neq t} \exp(-\frac{\|\mathbf{y}_u - \mathbf{y}_t\|^2}{2\sigma^2})}$$

and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

- However we are faced with the crowding problem:
  - In high dimension we have a lot of space, Eg. in d dimension we have $d + 1$ equidistant point
  - For d dimensional gaussians, most points are found at distance $\sqrt{d}$ from mean!

Choice for Q

- Just like we defined P, we can define Q for a given $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by

$$q_{t \to s} = \frac{\exp(-\frac{\|\mathbf{y}_s - \mathbf{y}_t\|^2}{2\sigma^2})}{\sum_{u \neq t} \exp(-\frac{\|\mathbf{y}_u - \mathbf{y}_t\|^2}{2\sigma^2})}$$

  and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

- However we are faced with the crowding problem:
  - In high dimension we have a lot of space, Eg. in d dimension we have $d + 1$ equidistant point
  - For d dimensional gaussians, most points are found at distance $\sqrt{d}$ from mean!
  - If we use gaussians in both high and low dimensional space, all the points are squished in to a small space
  - Too many points crowd the center!

METHOD II: T-SNE

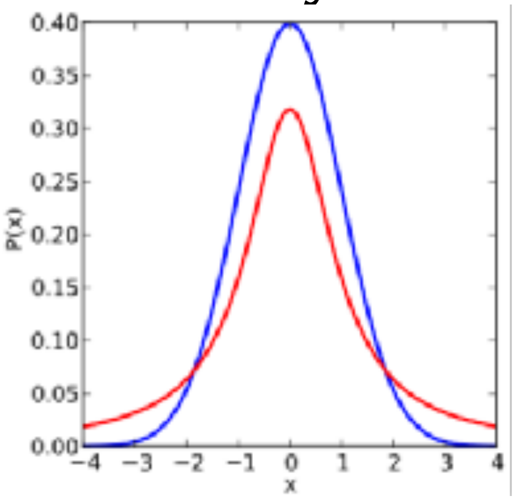
- Instead for Q we use, student t distribution which is heavy tailed:

$$q_{t \to s} = \frac{\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}}{\sum_{u \neq t} \left(1 + \|\mathbf{y}_u - \mathbf{y}_t\|^2\right)^{-1}}$$

and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

Method II: t-SNE

- Instead for Q we use, student t distribution which is heavy tailed:



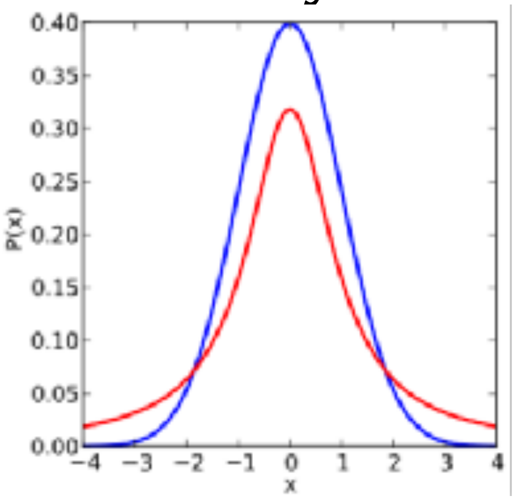
$$q_{t \to s} = \frac{\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}}{\sum_{u \neq t} \left(1 + \|\mathbf{y}_u - \mathbf{y}_t\|^2\right)^{-1}}$$

and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

METHOD II: T-SNE

- Instead for Q we use, student t distribution which is heavy tailed:

$$q_{t\to s} = \frac{\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}}{\sum_{u\neq t}\left(1 + \|\mathbf{y}_u - \mathbf{y}_t\|^2\right)^{-1}}$$



and then set $Q_{s,t} = \frac{q_{t\to s} + q_{s\to t}}{2n}$

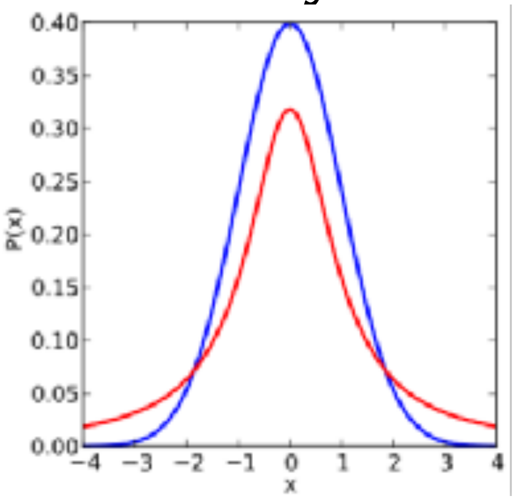
- It can be verified that

$$\nabla_{\mathbf{y}_t}\mathrm{KL}(P\|Q) = 4\sum_{s=1}^{n}(P_{s,t} - Q_{s,t})(\mathbf{y}_t - \mathbf{y}_s)\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}$$

Method II: t-SNE

- Instead for Q we use, student t distribution which is heavy tailed:

$$q_{t \to s} = \frac{\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}}{\sum_{u \neq t} \left(1 + \|\mathbf{y}_u - \mathbf{y}_t\|^2\right)^{-1}}$$



and then set $Q_{s,t} = \frac{q_{t \to s} + q_{s \to t}}{2n}$

- It can be verified that

$$\nabla_{\mathbf{y}_t} \mathrm{KL}(P\|Q) = 4 \sum_{s=1}^{n} (P_{s,t} - Q_{s,t})(\mathbf{y}_t - \mathbf{y}_s)\left(1 + \|\mathbf{y}_s - \mathbf{y}_t\|^2\right)^{-1}$$

- Algorithm: Find $\mathbf{y}_1, \ldots, \mathbf{y}_n$ by performing gradient descent

Demo