Non-Linear Dimensionality Reduction

Course Webpage :
http://www.cs.cornell.edu/Courses/cs4786/2016fa/
Assignment 0 feedback available on cms.

Assignment 1 helper code in matlab, ipython and R added, due on friday.
Recap
Pick a Random $W$ 

\[ Y = X \times \begin{bmatrix} +1 & \ldots & -1 \\ -1 & \ldots & +1 \\ +1 & \ldots & -1 \\ \vdots \\ +1 & \ldots & -1 \\ K & & \end{bmatrix} \frac{d}{\sqrt{K}} \]
What does “it works” even mean?

Distances between all pairs of data-points in low dim. projection is roughly the same as their distances in the high dim. space.

\[
(1 - \varepsilon) ||y_i - y_j||^2 \leq ||x_i - x_j||^2 \leq (1 + \varepsilon) ||y_i - y_j||^2
\]
Say $K = 1$. Consider any vector $\tilde{x} \in \mathbb{R}^d$ and let $\tilde{y} = \tilde{x}^T W$.

We showed that: $\mathbb{E}[|\tilde{y}|^2] = \|\tilde{x}\|^2$

$K > 1$, $\tilde{y}[j] = \tilde{x}^T W_j$ \hspace{1cm} $\tilde{y}[i]$ & $\tilde{y}[j]$ are independent

(since we divide each entry of random matrix by $\sqrt{K}$ in $W$)

$\mathbb{E} \left[ |\tilde{y}[j]|^2 \right] = \frac{1}{K} \|\tilde{x}\|^2$

Hence, $\mathbb{E}\|\tilde{y}\|^2 = \sum_{j=1}^{K} \mathbb{E} \left[ y[j]^2 \right] = \sum_{j=1}^{K} \frac{1}{K} \|\tilde{x}\|^2 = \|\tilde{x}\|^2$

This is like taking an average of $K$ independent measurements whose expectations are $\|\tilde{x}\|^2$
Why should Random Projections even work?!

For large $K$, not only true in expectation but also with high probability

For any $\epsilon > 0$, if $K \approx \log(n/\delta)/\epsilon^2$, with probability $1-\delta$ over draw of $W$, for all pairs of data points $i, j \in \{1, \ldots, n\}$,

$$\left(1 - \epsilon\right) \|y_i - y_j\|_2^2 \leq \|x_i - x_j\|_2 \leq \left(1 + \epsilon\right) \|y_i - y_j\|_2^2$$

Lets try on Matlab . . .

This is called the Johnson-Lindenstrauss lemma or JL lemma for short.
If we take $\epsilon = 1/4$, then taking $K \approx 185$ with probability 0.99 distances are preserved to factor $1/4$. 

Why is this so Ridiculously Magical?
Why is this so ridiculously magical?

If we take $\epsilon = 1/4$, then taking $K \approx 185$ with probability 0.99 distances are preserved to factor 1/4
Why is this so Ridiculously Magical?

If we take \( \epsilon = 1/4 \), then taking \( K \approx 185 \) with probability 0.99 distances are preserved to factor 1/4
Kernel PCA
(non-linear projections)
LINEAR PROJECTIONS

\[ \begin{align*}
    n & \quad X & \quad d \\
    & \quad \times \quad W & \quad = \quad n \\
    & \quad d & \quad Y \\
\end{align*} \]

Works when data lies in a low dimensional linear sub-space
Demo
Given $\mathbf{x}_t \in \mathbb{R}^d$, the feature space vector is given by mapping

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

Enumerating products up to order $K$ (i.e., 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 $\Phi$?
Kernel Trick

Essence of Kernel trick:

- If we can write down an algorithm only in terms of $\Phi(x_t)^T \Phi(x_s)$ for data points $x_t$ and $x_s$,

- Then we don’t need to explicitly enumerate $\Phi(x_t)$’s but instead, compute $k(x_t, x_s) = \Phi(x_t)^T \Phi(x_s)$ (even if $\Phi$ maps to infinite dimensional space)

Example: RBF kernel $k(x_t, x_s) = \exp(-\sigma \|x_t - x_s\|_2^2)$, polynomial kernel $k(x_t, x_s) = (x_t^T y_t)^p$

Kernel function measures similarity between points.
\( k^{\text{th}} \) column of \( W \) is eigenvector of covariance matrix

That is, \( \lambda_k W_k = \Sigma W_k \). Rewriting, for centered \( X \)

\[
\lambda_k W_k = \frac{1}{n} \left( \sum_{t=1}^{n} x_t x_t^\top \right) W_k = \frac{1}{n} \sum_{t=1}^{n} (x_t^\top W_k) x_t
\]

\( W_k \)'s can be written as linear combination of \( x_t \)'s, as

\[
W_k = \sum_{t=1}^{n} \alpha_k[t] x_t
\]

where \( \alpha_k[t] = \frac{1}{\lambda_k n} (x_t^\top W_k) \)
We have that \( W_k = \sum_{s=1}^{n} \alpha_k[s] x_s \) and that \( \alpha_k[t] = \frac{1}{\lambda_k n} (x_t^\top W_k) \).

Hence:

\[
\alpha_k[t] = \frac{1}{\lambda_k n} \left( x_t^\top \left( \sum_{s=1}^{n} \alpha_k[s] x_s \right) \right) = \frac{1}{\lambda_k n} \sum_{s=1}^{n} \alpha_k[s] x_t^\top x_s
\]

Let \( \tilde{K} \) be a matrix such that \( \tilde{K}_{s,t} = x_t^\top x_s \). Hence, \( \alpha_k[t] = \frac{1}{\lambda_k n} \alpha_k^\top \tilde{K}_t \) and

\[
\alpha_k = \frac{1}{\lambda_k n} \tilde{K} \alpha_k
\]

where \( \tilde{K}_t \) is the t’th column of \( \tilde{K} \).

Hence \( \alpha_k \) is in the direction of eigen vector of \( \tilde{K} \).
Further, since $W_k$ is unit norm,

$$1 = \| W_k \|^2 = \left( \sum_{t=1}^{n} \alpha_k[t] x_t \right)^\top \left( \sum_{s=1}^{n} \alpha_k[s] x_s \right) = \alpha_k^\top \tilde{K} \alpha_k = n \gamma_k \alpha_k^\top \alpha_k$$

Hence $\| \alpha_k \|^2 = \frac{1}{n \gamma_k}$ where $\gamma_k$ is the $k'$th eigen value of matrix $\tilde{K}$.
However $W_k$ itself is in feature space and has the same dimensionality of $\Phi(x)$ (which is possibly infinite)!

However, the projections are in $K$ dimensions and we can hope to directly compute these as:

$$y_i[k] = x_i^T W_k = \sum_{t=1}^{n} \alpha_k[t] \tilde{K}_{t,i}$$
We assumed centered data, what if its not,

\[
\tilde{K}_{s,t} = \left( x_t - \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top \left( x_s - \frac{1}{n} \sum_{u=1}^{n} x_u \right) \\
= x_t^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_s - \left( \frac{1}{n} \sum_{u=1}^{n} x_u \right)^\top x_t \\
+ \frac{1}{n^2} \left( \sum_{u=1}^{n} x_u \right)^\top \left( \sum_{v=1}^{n} x_v \right) \\
= x_t^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_s - \frac{1}{n} \sum_{u=1}^{n} x_u^\top x_t + \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} x_u^\top x_v
\]
Equivalently, if $\text{Kern}$ is the matrix $(\text{Kern}_{t,s} = x_t^T x_s)$,

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

- Compute top $K$ eigen vectors $P_1, \ldots, P_K$ along with eigen values $\gamma_1, \ldots, \gamma_K$ for the matrix $\tilde{K}$

- Rescale each $P_k$ by the inverse of the square-root of corresponding eigen values ie. $\alpha_k = P_k / \sqrt{n \gamma_k}$

- Compute projections by setting

$$y_i[k] = \sum_{t=1}^{n} \alpha_k[t] \tilde{K}_{t,i}$$

or in other words $Y = \tilde{K} \times [\alpha_1, \ldots, \alpha_K]$
All we need to be able to compute, to perform PCA are $x_t^\top x_s$

Replace $x_t^\top x_s$ with $\Phi(x_t)^\top \Phi(x_s) = k(x_t, x_s)$ to perform PCA in feature space
If we want to port PCA to kernel PCA, we need to be able to write \( \tilde{K} \) in terms of kernel functions.

We assumed centered data, so

\[
\tilde{K}_{s,t} = \sum_{u=1}^{n} \left( x_u - \frac{1}{n} \sum_{v=1}^{n} x_v \right) \left( x_s - \frac{1}{n} \sum_{v=1}^{n} x_v \right) - \frac{1}{n^2} \sum_{u=1}^{n} \sum_{v=1}^{n} (x_u - \frac{1}{n} \sum_{w=1}^{n} x_w) (x_v - \frac{1}{n} \sum_{w=1}^{n} x_w)
\]

Knowing kernel function, we can perform Kernel PCA even when maps to infinite dimensional space!
Kernel PCA

3. $\begin{bmatrix} P \end{bmatrix} = \text{eigs} \left( \begin{bmatrix} \tilde{K} \end{bmatrix}, K \right)$

4. $n \begin{bmatrix} \alpha \end{bmatrix} = n \begin{bmatrix} \frac{P_1 \ldots P_K}{\sqrt{n\gamma_1} \ldots \sqrt{n\gamma_K}} \end{bmatrix}$

5. $n \begin{bmatrix} Y \end{bmatrix} = n \begin{bmatrix} \tilde{K} \end{bmatrix} \times n \begin{bmatrix} \alpha \end{bmatrix}$
Demo