Kernel Methods for Dimensionality Reduction

Ryan Vaughn

September 7, 2018

Overview

Overview

- What is dimensionality reduction?

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- How do we use kernels in dimensionality reduction?

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- How do we use kernels in dimensionality reduction?
- Linear Technique: Principal Component Analysis

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- How do we use kernels in dimensionality reduction?
- Linear Technique: Principal Component Analysis
- Nonlinear Technique: Kernel Principal Component Analysis

Dimensionality Reduction

- Let $\{x_i\}_{i=1}^n$ be a finite subset of points in $\mathbb{R}^D$ sampled from a sample space $S \subseteq \mathbb{R}^D$.

Dimensionality Reduction

- Let $\{x_i\}_{i=1}^n$ be a finite subset of points in $\mathbb{R}^D$ sampled from a sample space $S \subseteq \mathbb{R}^D$.
- Often we input our data $\{x_i\}_{i=1}^n$ in an algorithm by representing it as a data matrix X where the points x_i are the rows of the matrix.

$$X = \begin{pmatrix} - & x_1^\top & - \\ & \vdots & \\ - & x_i^\top & - \\ & \vdots & \\ - & x_n^\top & - \end{pmatrix}$$

Dimensionality Reduction

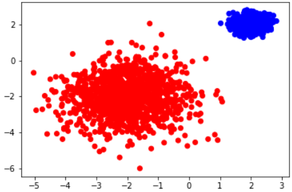
- The goal of a Dimensionality reduction learning problem is to obtain a function $P : \mathbb{R}^D \to \mathbb{R}^d$ with $d << D$ such that P preserves the "interesting features" of the sample space S.

Dimensionality Reduction

- The goal of a Dimensionality reduction learning problem is to obtain a function $P : \mathbb{R}^D \to \mathbb{R}^d$ with $d << D$ such that P preserves the "interesting features" of the sample space S.
- If $P : \mathbb{R}^D \to \mathbb{R}^d$ is a linear map, we call the learning problem a linear dimensionality reduction technique. Otherwise, it is nonlinear.

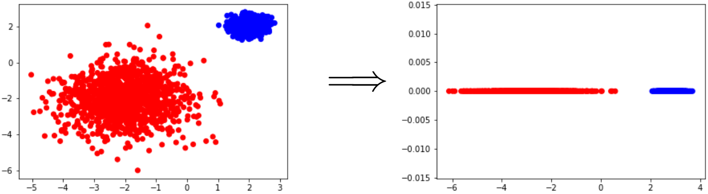
Examples

Let $D = 2$ and $d = 1$:



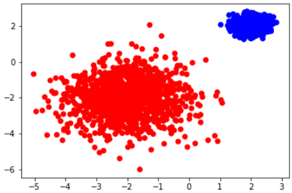
Examples

Let $D = 2$ and $d = 1$:

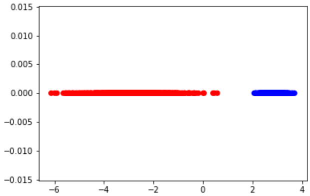


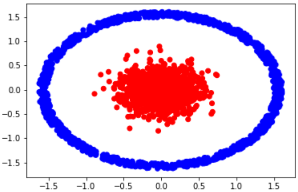
Examples

Let $D = 2$ and $d = 1$:



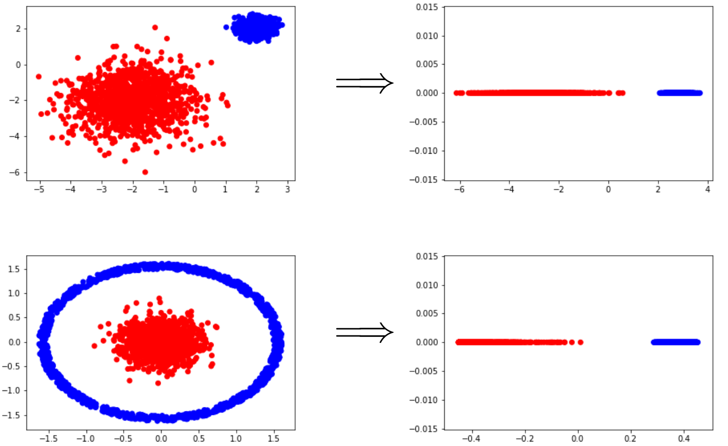
$\Longrightarrow$





Examples

Let $D = 2$ and $d = 1$:



Broad Overview of Kernel Methods

- Algorithms based on linear algebra are often computable.
- Algorithms based on linear algebra often produce linear projections or linearly projected data.
- Kernel methods are a way to modify these linear techniques so that the output is a nonlinear mapping on the data.

Linear technique + choice of kernel = kernelized linear technique

Overview

- What is dimensionality reduction?
- **What are kernels and Reproducing Kernel Hilbert Spaces?**
- How do we use kernels in dimensionality reduction?
- Linear Technique: Principal Component Analysis
- Nonlinear Technique: Kernel Principal Component Analysis

Kernels

Recall that S is the space from which we sample data. In most cases, S will be some subset of $\mathbb{R}^D$

Recall that S is the space from which we sample data. In most cases, S will be some subset of $\mathbb{R}^D$

Definition

A function $k : S \times S \to \mathbb{R}$ is a symmetric and positive definite kernel if:

1. for any $x, y \in S$, $k(x, y) = k(y, x)$
2. for any finite set of points x_i and real coefficients c_i,

$$\sum_{i,j \leq n}^{n} c_i c_j k(x_i, x_j) \geq 0$$

Examples of Kernels

Examples of symmetric and positive definite kernels:

1. Linear Kernel: $k(x, y) = x \cdot y$

Examples of Kernels

Examples of symmetric and positive definite kernels:

1. Linear Kernel: $k(x, y) = x \cdot y$
2. Polynomial Kernel: $k(x, y) = (x \cdot y + c)^j$

Examples of Kernels

Examples of symmetric and positive definite kernels:

1. Linear Kernel: $k(x, y) = x \cdot y$
2. Polynomial Kernel: $k(x, y) = (x \cdot y + c)^j$
3. Sigmoid Kernel: $k(x, y) = \tanh(\gamma x \cdot y + c)$

Examples of Kernels

Examples of symmetric and positive definite kernels:

1. Linear Kernel: $k(x, y) = x \cdot y$
2. Polynomial Kernel: $k(x, y) = (x \cdot y + c)^j$
3. Sigmoid Kernel: $k(x, y) = \tanh(\gamma x \cdot y + c)$
4. Radial Basis Function: $k(x, y) = e^{-|x-y|^2/\epsilon^2}$

Examples of Kernels

Examples of symmetric and positive definite kernels:

1. Linear Kernel: $k(x, y) = x \cdot y$
2. Polynomial Kernel: $k(x, y) = (x \cdot y + c)^j$
3. Sigmoid Kernel: $k(x, y) = \tanh(\gamma x \cdot y + c)$
4. Radial Basis Function: $k(x, y) = e^{-|x-y|^2/\epsilon^2}$
5. There are even kernels which are used on things like genetic or text data!

Definition

A Hilbert space $\mathcal{H}$ is an infinite dimensional vector space together with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that the metric

$$d(x, y) = \sqrt{\langle x - y, x - y \rangle_{\mathcal{H}}}$$

is complete (Cauchy sequences converge.)

Reproducing Kernel Hilbert Spaces

Given a symmetric and positive definite kernel k, one can construct an inner-product space of functions in the following manner:

Given a symmetric and positive definite kernel k, one can construct an inner-product space of functions in the following manner:

- For each $x \in S$, consider the function $k(x, \cdot) : S \to \mathbb{R}$

Reproducing Kernel Hilbert Spaces

Given a symmetric and positive definite kernel k, one can construct an inner-product space of functions in the following manner:

- For each $x \in S$, consider the function $k(x, \cdot) : S \to \mathbb{R}$
- Let $\mathcal{H}$ be the span of all such functions under function addition and scalar multiplication.

Reproducing Kernel Hilbert Spaces

Given a symmetric and positive definite kernel k, one can construct an inner-product space of functions in the following manner:

- For each $x \in S$, consider the function $k(x, \cdot) : S \to \mathbb{R}$
- Let $\mathcal{H}$ be the span of all such functions under function addition and scalar multiplication.
- Define an inner product on $\mathcal{H}$ by:

$$\langle k(x, \cdot), k(y, \cdot) \rangle_{\mathcal{H}} = k(x, y).$$

We define the feature map $\phi : S \to \mathcal{H}$ by the mapping $x \mapsto k(x, \cdot)$.

Theorem (Moore-Aronszajn[1])

Let k be a symmetric and positive definite kernel on S. Then $\mathcal{H}$ is the unique Hilbert space such that

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}.$$

The space $\mathcal{H}$ is called the Reproducing Kernel Hilbert Space of k.

Theorem (Moore-Aronszajn[1])

Let k be a symmetric and positive definite kernel on S. Then $\mathcal{H}$ is the unique Hilbert space such that

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}.$$

The space $\mathcal{H}$ is called the Reproducing Kernel Hilbert Space of k.

Takeaway: Given a kernel, there exists a Hilbert space $\mathcal{H}$ such that taking the inner product in $\mathcal{H}$ on points in S is the same as plugging them into the kernel function.

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- **How do we use kernels in dimensionality reduction?**
- Linear Technique: Principal Component Analysis
- Nonlinear Technique: Kernel Principal Component Analysis

How to Apply Kernels to Dimensionality Reduction

The Kernel Trick[2][3]:

1. Choose a kernel $k(\cdot, \cdot)$.

How to Apply Kernels to Dimensionality Reduction

The Kernel Trick[2][3]:

1. Choose a kernel $k(\cdot, \cdot)$.
2. Take any algorithm which can be computed purely using dot products $x_i \cdot x_j$.

How to Apply Kernels to Dimensionality Reduction

The Kernel Trick[2][3]:

1. Choose a kernel $k(\cdot, \cdot)$.
2. Take any algorithm which can be computed purely using dot products $x_i \cdot x_j$.
3. Replace each instance of $x_i \cdot x_j$ with $k(x_i, x_j)$.

How to Apply Kernels to Dimensionality Reduction

The Kernel Trick[2][3]:

1. Choose a kernel $k(\cdot, \cdot)$.
2. Take any algorithm which can be computed purely using dot products $x_i \cdot x_j$.
3. Replace each instance of $x_i \cdot x_j$ with $k(x_i, x_j)$.

Since $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$, this procedure results in carrying out the original algorithm inside of $\mathcal{H}$.

Overview

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- How do we use kernels in dimensionality reduction?
- **Linear Technique: Principal Component Analysis**
- Nonlinear Technique: Kernel Principal Component Analysis

What PCA does:

- **Input:** Data matrix X, choice of dimension d.

Linear Method: Principal Component Analysis

What PCA does:

- **Input:** Data matrix X, choice of dimension d.
- Let v_1 be the unit vector corresponding to the direction of highest variance. This vector is called the first principal component.

Linear Method: Principal Component Analysis

What PCA does:

- **Input:** Data matrix X, choice of dimension d.
- Let v_1 be the unit vector corresponding to the direction of highest variance. This vector is called the first principal component.
- Recursively define v_i as the unit vector in the direction of highest variance which is mutually orthogonal to all previously computed principal components.

Linear Method: Principal Component Analysis

What PCA does:

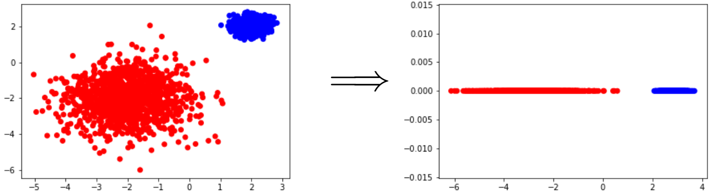
- **Input:** Data matrix X, choice of dimension d.
- Let v_1 be the unit vector corresponding to the direction of highest variance. This vector is called the first principal component.
- Recursively define v_i as the unit vector in the direction of highest variance which is mutually orthogonal to all previously computed principal components.
- The result is an ordered orthonormal basis $\{v_i\}_{i=1}^{D}$ for $\mathbb{R}^D$ called the principal components of D.

Linear Method: Principal Component Analysis

What PCA does:

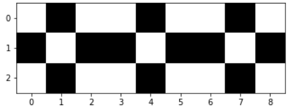
- **Input:** Data matrix X, choice of dimension d.
- Let v_1 be the unit vector corresponding to the direction of highest variance. This vector is called the first principal component.
- Recursively define v_i as the unit vector in the direction of highest variance which is mutually orthogonal to all previously computed principal components.
- The result is an ordered orthonormal basis $\{v_i\}_{i=1}^{D}$ for $\mathbb{R}^D$ called the principal components of D.
- Define $P : \mathbb{R}^D \to \mathbb{R}^d$ as the linear mapping formed by projecting the data onto the subspace spanned by the first d principal components.

PCA on Two Gaussian Datasets



PCA in Image Processing

A black and white image with can be viewed as a matrix of numbers with values between 0 and 1.
Example: A 9×3 image.



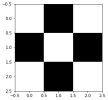
$$= \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

PCA in Image Processing

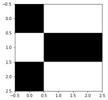
Consider each 3×3 subimage of our example as a vector in $\mathbb{R}^9$.

PCA in Image Processing

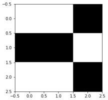
Consider each 3×3 subimage of our example as a vector in $\mathbb{R}^9$. There are 7 such subimages, but only three unique ones:



$$= (0,1,0,1,0,1,0,1,0)$$



$$= (1,0,0,0,1,1,1,0,0)$$



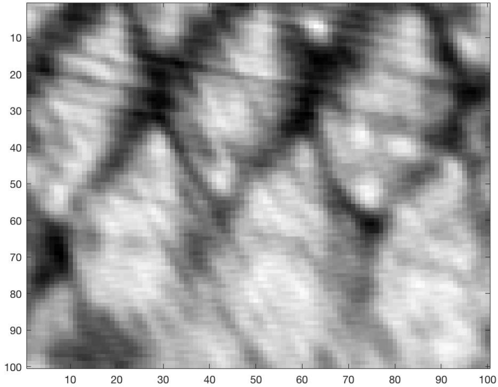
$$= (0,0,1,1,1,0,0,0,1)$$

As we move left to right on the image, the subimages begin to repeat themselves.

Translational repetition in an image creates "loops" in the set of subimages.

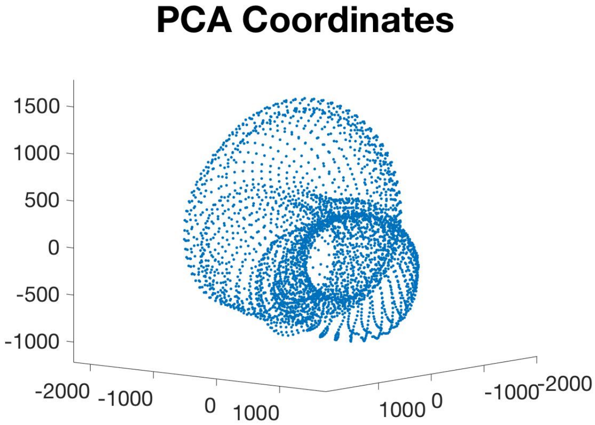
PCA in Image Processing

Consider the set of 40×40 subimages of the following image:



The subimages are vectors in $\mathbb{R}^{1600}$.

This is a PCA projection of the space of subimages into $\mathbb{R}^3$



PCA captures the "loops" in the subimage space.

- What is dimensionality reduction?
- What are kernels and Reproducing Kernel Hilbert Spaces?
- How do we use kernels in dimensionality reduction?
- Linear Technique: Principal Component Analysis
- **Nonlinear Technique: Kernel Principal Component Analysis**

Computing PCA

In order to learn how to kernelize PCA, we must know how to compute it! Recall the *Singular Value Decomposition*:

Theorem

Let X be an $n \times d$ matrix. Then there exists orthogonal matrices U and V of size $n \times n$ and $d \times d$ respectively, as well as a diagonal matrix S of size $n \times d$ such that

$$X = USV^\top.$$

The columns and rows of U and V are called the right/left singular vectors of X respectively, and the diagonal entrees of S are called the singular values of X.

Computing PCA

Take the SVD of the data matrix X:

$$X = USV^\top.$$

Let V_d be the matrix consisting of the first d columns of V. It can be shown that the columns of V are the principal components of X. Thus, projecting onto the first d principal components is the same as multiplication on the right by the matrix V_d.

$$\begin{aligned} \mathrm{PCA}(X, d) &= XV_d \\ &= USV^\top V_d \\ &= U_d S_d \end{aligned}$$

Kernel PCA

In order to Kernelize PCA, we need to compute PCA using purely dot products. We do this by using the *Gram matrix*:

$$XX^\top = (x_i \cdot x_j)_{i,j}$$

Using SVD, we see that:

$$\begin{aligned} XX^\top &= USV^\top V^\top S^\top U^\top \\ &= USS^\top U^\top \end{aligned}$$

Thus, the SVD of the Gram matrix is $XX^\top = U(SS^\top)U^\top$.

Kernel PCA

Thus, to compute PCA from the Gram matrix, we simply compute the SVD:

$$XX^\top = U(SS^\top)U^\top.$$

Then compute PCA by computing $U_d(\sqrt{SS^\top}_d)$

Kernel PCA

Recall that to use the Kernel trick, we simply replace all instances of $x_i \cdot x_j$ with

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}.$$

So to carry out Kernel PCA, we use the matrix $K = (k(x_i, x_j))_{i,j}$ instead of $XX^\top$.

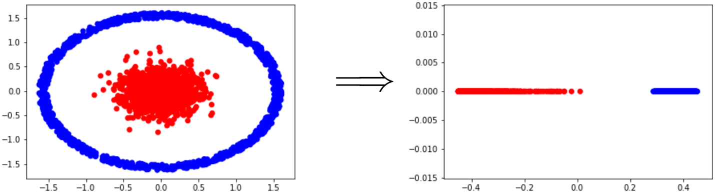
- Compute the SVD of K:

$$K = \tilde{U}\tilde{S}\tilde{U}^\top.$$

  (V = U in this case since K is a symmetric matrix.)
- $\text{kPCA}(X, d) = U_d\sqrt{S_d}$. Since

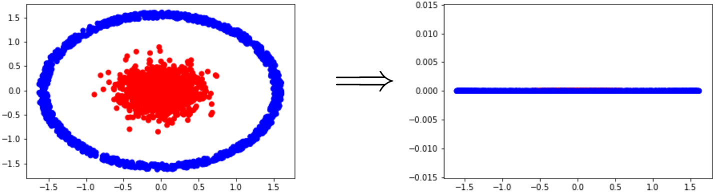
Kernel PCA on data

In the following, we use the RBF kernel $k(x, y) = e^{-|x-y|^2/\epsilon^2}$

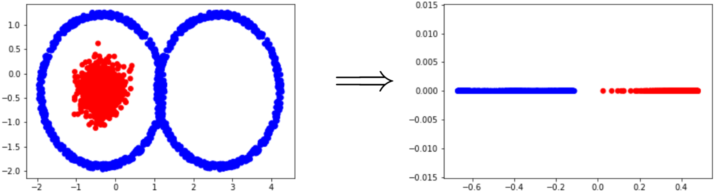


Kernel PCA on data

In comparison, this is how PCA performs:



Kernel PCA on data



Conclusion

Things I hope you learned:

1. What is a kernel?
2. What is a Reproducing Kernel Hilbert Space?
3. How to use kernels in dimensionality reduction.
4. How to implement kernel PCA.

Thank you!

(A special thanks to Dr. Tyrus Berry for the code used in the subimage analysis.)

Nachman Aronszajn.
Theory of reproducing kernels.
Transactions of the American mathematical society, 68(3):337–404, 1950.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola.
Kernel methods in machine learning.
The annals of statistics, pages 1171–1220, 2008.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller.
Kernel principal component analysis.
In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.