Dimensionality Reduction Part 1: PCA and KPCA

Ng Yen Kaow

Dimensionality Reduction

- Linear methods
  - **PCA** (Principal Component Analysis)
  - cMDS (Classical Multidimensional Scaling)
- Non-linear methods
  - **KPCA** (Kernel PCA)
  - mMDS (Metric MDS)
  - Isomap
  - LLE (Locally Linear Embedding)
  - Laplacian Eigenmap
  - t-SNE (t-distributed Stochastic Neighbor Embedding)
  - UMAP (Uniform Manifold Approximation and Projection)

Principal Component Analysis

- Let X be an $n \times m$ matrix where each row represents a datapoint in an m-D space
  - X is like a spreadsheet with features in column and data cases in the rows
- We want to identify some form of "principal components" of X, where ideally
  1. The components should form a basis
  2. The components should be orthogonal
  3. The first component should account for the most variation, the second component accounts for the most variation after removing the first, and so on

Principal Component Analysis

- Assume datapoints in X are generated by a random vector $\boldsymbol{X} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_m]$, where each $\boldsymbol{v}_i$ is a random variable
  - Covariance $\operatorname{cov}\left(\boldsymbol{v}_i, \boldsymbol{v}_j\right) = \mathbb{E}\left[(\boldsymbol{v}_i - \mu_i)\left(\boldsymbol{v}_j - \mu_j\right)\right]$
  - Define covariance matrix $M = \left(m_{ij}\right)$ of $\boldsymbol{X}$ where $m_{ij} = \operatorname{cov}\left(\boldsymbol{v}_i, \boldsymbol{v}_j\right)$

    (M can be estimated from $X = (x_{ij})$ as the outer product ${X^c}^{\mathrm{T}} X^c / n$ of a centered matrix $X^c = (x_{ij}^c)$ where $x_{ij}^c = x_{ij} - \mu_i$)
- For the first component, we want to find unit vector $u \in \mathbb{R}^m$ such that variance $\operatorname{var}\left(u^{\mathrm{T}} \boldsymbol{X}\right)$ is maximized

Principal Component Analysis

- The eigenvector u of the covariance matrix M of $\boldsymbol{X}$ with the largest eigenvalue maximizes $\operatorname{var}\left(u^{\mathrm{T}}\boldsymbol{X}\right)$

Gives a matrix since $\boldsymbol{X}$ and μ are column vectors

Let $\boldsymbol{X} \in \mathbb{R}^m$ be a random vector with

- mean vector $\mu \in \mathbb{R}^m$ and
- covariance matrix $M = \mathbb{E}[(\boldsymbol{X} - \mu)(\boldsymbol{X} - \mu)^{\mathrm{T}}]$

For any $u \in \mathbb{R}^n$, the projection of $u^{\mathrm{T}}\boldsymbol{X}$ has

- $\mathbb{E}[u^{\mathrm{T}}\boldsymbol{X}] = u^{\mathrm{T}}\mu$ and
- $\operatorname{var}\left(u^{\mathrm{T}}\boldsymbol{X}\right) = \mathbb{E}\left[\left(u^{\mathrm{T}}\boldsymbol{X} - u^{\mathrm{T}}\mu\right)^2\right]$

$$= \mathbb{E}[u^{\mathrm{T}}(\boldsymbol{X} - \mu)(\boldsymbol{X} - \mu)^{\mathrm{T}}u] = u^{\mathrm{T}}Mu$$

From min-max theorem, $u^{\mathrm{T}}Mu$ is maximized when u is the eigenvector of M with the largest eigenvalue

Principal Component Analysis

- Extend to k principal components, we want
  - k-D subspace of $\boldsymbol{X}$ that is defined by orthogonal basis $p_1, \ldots, p_k \in \mathbb{R}^m$ and displacement $p_0 \in \mathbb{R}^m$
  - Distance from $\boldsymbol{X}$ to this subspace is minimized

---

- Projection of $\boldsymbol{X}$ onto subspace is $P^{\mathrm{T}}\boldsymbol{X} + \mathrm{p}_{\mathbf{0}}$, where P is matrix whose rows are $p_1, \ldots, p_k$
- Squared distance to subspace is $\mathbb{E}\left\|\boldsymbol{X} - \left(P^{\mathrm{T}}\boldsymbol{X} + p_{\mathbf{0}}\right)\right\|^2$
- By calculus, $\mathrm{p}_{\mathbf{0}} = \mathbb{E}\left\|\boldsymbol{X} - P^{\mathrm{T}}\boldsymbol{X}\right\| = \left(1 - P^{\mathrm{T}}\right)\mu$, hence
  $$\mathbb{E}\left\|\boldsymbol{X} - \left(P^{\mathrm{T}}\boldsymbol{X} + p_{\mathbf{0}}\right)\right\|^2 = \mathbb{E}\|\boldsymbol{X} - \mu\|^2 - \mathbb{E}\left\|P^{\mathrm{T}}(\boldsymbol{X} - \mu)\right\|^2$$
- To maximize that, need to maximize $\mathbb{E}\left\|P^{\mathrm{T}}(\boldsymbol{X} - \mu)\right\|^2 = \mathrm{var}\left(P^{\mathrm{T}}\boldsymbol{X}\right)$
- Finally, same as in previous slide, $p_1, \ldots, p_k$ are eigenvectors of M

Principal Component Analysis

- As mentioned, given a centered matrix $X^c = (x_{ij}^c)$ where $x_{ij}^c = x_{ij} - \mu_i$, an unbiased estimator of M can be obtained as
$$M = \frac{1}{n} X^{c\mathrm{T}} X^c \quad \text{(or } M = \frac{1}{n} \textstyle\sum_i x_i^{c\mathrm{T}} x_i^c\text{)}$$
  - This implies that M is positive semi-definite
- Since SVD of X eigendecomposes $X^{c\mathrm{T}} X^c$
  - We can solve PCA through either
    1. Eigendecompose M, or
    2. Solve SVD for X^c

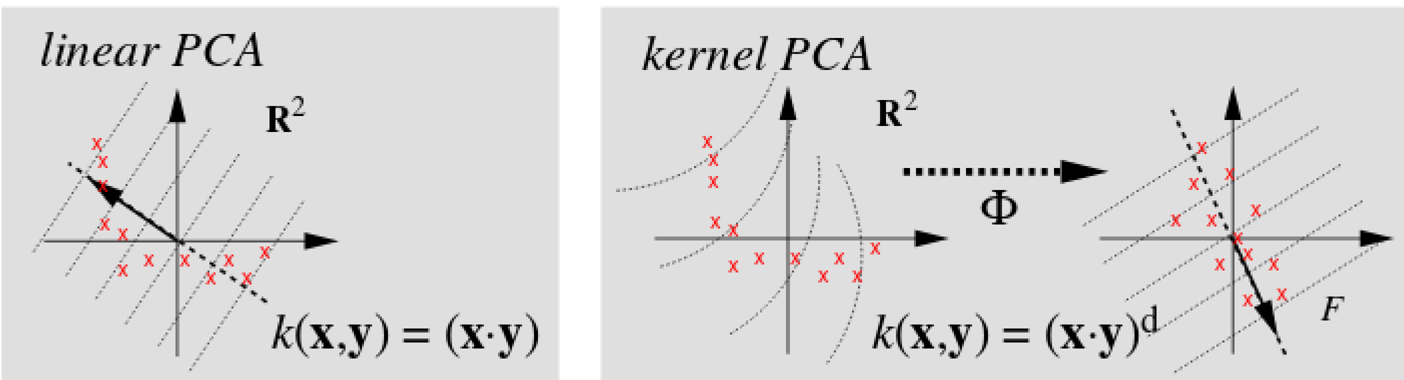
Choose between XX^{T} or $X^{\mathrm{T}}X$

- XX^{T} and $X^{\mathrm{T}}X$ have equivalent eigenpairs
- For any matrices A and B, AB and BA have the same non-zero eigenvalues
  - Let $\lambda \neq 0$ be a eigenvalue for AB with eigenvector v
  - Then $ABv = \lambda v \Rightarrow BABv = \lambda Bv$
    $\Rightarrow (BA)(Bv) = \lambda(Bv)$

  $\Rightarrow \lambda$ is a eigenvalue of BA with eigenvector (Bv)
- For PCA through eigendecomposition, choose the smaller between XX^{T}/n and $X^{\mathrm{T}}X/n$

Kernel PCA motivation

- Datapoints that do not lie on a linear manifold in the coordinate space may lie on one after some non-linear feature map ϕ to a high dimensional space



Scholkopf, Smola, and Muller. Kernel Principal Component Analysis, 1999

- Principal components in the ϕ-mapped feature space may be more meaningful

Kernel PCA idea

- Steps to get the principal components in a ϕ-mapped feature space:
  1. $x' = \phi(x)$ and $X' = [x_1' \quad ... \quad x_n']^{\mathrm{T}}$
  2. Center X' (deduct column mean)
  3. Find covariance matrix, $M' = \frac{1}{n}\sum_i x_i'^{\mathrm{T}} x_i'$
  4. Eigendecompose M'

- Difficult since dimension of x', $\dim(x')$ will be large (or even ∞)

  $\Rightarrow$ M' has large (or even ∞) dimensions

  $\Rightarrow$ Eigendecomposition of M' gives large (or infinite) number of eigenvectors, each of large (or infinite) dimensions

Kernel PCA idea

Problem 1: Large number of eigenvectors

- How many eigenvectors are there actually
  - $\mathrm{rank}(M')$, bounded by the number of datapoints
    - Recall that eigenvectors can be expressed as a linear combination of the datapoints by solving the equations $x_i' = \sum_j \langle x_i', u_j \rangle u_j$
      - j is bounded by $\mathrm{rank}(M') \Rightarrow$ may be manageable
      - However, working with the system of equations is hard because $x_i{}'$ and u_j are of…

Problem 2: Large (or ∞) dimensions

Kernel method

- Do not compute $\phi(x_1), \dots, \phi(x_n)$ or eigenvectors of M'
  - Allow only comparisons between datapoints in mapped space through inner product $\langle x_i', x_j' \rangle$
    - Sufficient for writing eigenvector u of M' in terms of $\phi(x_1), \dots, \phi(x_n)$ (i.e. project u onto $\phi(x_1), \dots, \phi(x_n)$)
    - Sufficient for finding the eigenvalues of M'
    - Given point x, sufficient for finding the projection of $\phi(x)$ on the eigenvectors of M'
  - Use a function $K(x_i, x_j)$ (called a kernel function) that does not require computing ϕ to compute $\langle x_i', x_j' \rangle$
    - Conditions for such a function given in later slides

Project eigenvector to $x_1', \ldots, x_n'$

- Relate eigenvectors of M' with $x_1', \ldots, x_n'$ using a computation that involves only $\langle x_i', x_j' \rangle$
- Start with the definition of $M' = \frac{1}{n}\left(\sum_{i=1}^{n} x_i'^{\mathrm{T}} x_i'\right)$
  - Solving $M'u = \lambda u$ means $\left(\sum_i x_i'^{\mathrm{T}} x_i'\right)u = n\lambda u$
  - This implies $u = \frac{1}{n\lambda}\sum_i x_i'^{\mathrm{T}} x_i' u$. Since $x^{\mathrm{T}}xu \overset{\text{Proof later}}{=} xux^{\mathrm{T}}$, $u = \frac{1}{n\lambda}\sum_i \overbrace{x_i' u}^{\text{scalar}} x_i'^{\mathrm{T}}$

    Hence can let $u = \sum_{i=1}^{n} \alpha_i x_i'^{\mathrm{T}}$ for $\alpha_i \in \mathbb{R}$
    - $\alpha_1, \ldots, \alpha_n$ **project eigenvector** u **to** $x_1', \ldots, x_n'$
- Place $u^{(r)} = \sum_i \alpha_i^{(r)} x_i'^{\mathrm{T}}$ back in $\left(\sum_i x_i'^{\mathrm{T}} x_i'\right)u = n\lambda u$
  - Use superscript r to associate α with its corresponding u and λ

Solving $\alpha_1, \dots, \alpha_n$

(Terms in bold cannot be reordered)

System of $\dim(u)$ equations

$$\left(\sum_{i=1}^{n} \boldsymbol{x}_i'^{\mathrm{T}} \boldsymbol{x}_i'\right) \boldsymbol{u}^{(r)} = n\lambda^{(r)} \boldsymbol{u}^{(r)}$$

Replace $\boldsymbol{u}^{(r)}$ with $\sum_j \alpha_j^{(r)} \boldsymbol{x}_j'^{\mathrm{T}}$

$$\left(\sum_{i=1}^{n} \boldsymbol{x}_i'^{\mathrm{T}} \boldsymbol{x}_i'\right) \sum_{j=1}^{n} \alpha_j^{(r)} \boldsymbol{x}_j'^{\mathrm{T}} = n\lambda^{(r)} \sum_{k=1}^{n} \alpha_k^{(r)} \boldsymbol{x}_k'^{\mathrm{T}}$$

Reorder

$$\left(\sum_i \boldsymbol{x}_i'^{\mathrm{T}}\right) \sum_j \overbrace{\boldsymbol{x}_i' \boldsymbol{x}_j'^{\mathrm{T}}}^{\text{scalar}} \alpha_j^{(r)} = n\lambda^{(r)} \sum_k \boldsymbol{x}_k'^{\mathrm{T}} \alpha_k^{(r)}$$

Multiply from the left with $\boldsymbol{x}_l'$ (equation holds for each l)

System of one equation

$$\left(\sum_i \overbrace{\boldsymbol{x}_l' \boldsymbol{x}_i'^{\mathrm{T}}}^{\text{scalar}}\right) \sum_j \boldsymbol{x}_i' \boldsymbol{x}_j'^{\mathrm{T}} \alpha_j^{(r)} = n\lambda^{(r)} \sum_k \underbrace{\boldsymbol{x}_l' \boldsymbol{x}_k'^{T}}_{\text{scalar}} \alpha_k^{(r)}$$

Replace $\boldsymbol{x}_i' \boldsymbol{x}_j'^{\mathrm{T}}$ with the kernel function

$$\sum_i K(x_l, x_i) \sum_j K\left(x_i, x_j\right) \alpha_j^{(r)} = n\lambda^{(r)} \sum_k K(x_l, x_k)\, \alpha_k^{(r)}$$

Reorder

$$\sum_i \sum_j K(x_l, x_i) K\left(x_i, x_j\right) \alpha_j^{(r)} = n\lambda^{(r)} \sum_k K(x_l, x_k)\, \alpha_k^{(r)}$$

Solving $\alpha_1, \dots, \alpha_n$

$$\sum_i \sum_j K(x_l, x_i) K(x_i, x_j)\, \alpha_j^{(r)} = n\lambda^{(r)} \sum_k K(x_l, x_k)\, \alpha_k^{(r)}$$

- **Replace** $K(x_i, x_j)$ **with a matrix** $\boldsymbol{K}$ **where** $k_{ij} = K(x_i, x_j)$ ($\boldsymbol{K}$ is called a kernel matrix)

$$\sum_i \sum_j k_{li} k_{ij}\, \alpha_j^{(r)} = n\lambda^{(r)} \sum_k k_{lk}\, \alpha_k^{(r)}$$

- For each l this gives one single equation with a linear combination of the variables $\alpha_1^{(r)}, \dots, \alpha_n^{(r)}$
  - e.g. $l = 2$

$$\begin{array}{cc} & \\ K_l \rightarrow & \end{array}\begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{array}{c} \begin{array}{ccc} K_1^{\mathrm{T}} & K_2^{\mathrm{T}} & \\ \downarrow & \downarrow & \end{array} \\ \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix} = n\lambda^{(r)} \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix}$$

$$(k_{21}k_{11} + k_{22}k_{21} + \cdots)\alpha_1^{(r)} + (k_{21}k_{12} + k_{22}k_{22} + \cdots)\alpha_2^{(r)} + \cdots$$
$$= n\lambda^{(r)} \left(k_{21}\alpha_1^{(r)} + k_{21}\alpha_2^{(r)} + \cdots \right)$$

Solving $\alpha_1, \ldots, \alpha_n$

System of one equation

$$
\begin{array}{cc}
 & \\
K_l \rightarrow & \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}
\end{array}
\begin{array}{c}
\begin{array}{ccc} K_1^{\mathrm{T}} & K_2^{\mathrm{T}} & \\ \downarrow & \downarrow & \end{array} \\
\begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}
\end{array}
\begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix} = n\lambda^{(r)} \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix}
$$

- Repeat l for 1 to n

System of n equations

$$
\begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}
\begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}
\begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix} = n\lambda^{(r)} \begin{bmatrix} k_{11} & k_{12} & \cdots \\ k_{21} & k_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \alpha_1^{(r)} \\ \alpha_2^{(r)} \\ \vdots \end{bmatrix}
$$

- This in matrix notation is

$$\boldsymbol{K}^2 \boldsymbol{\alpha}^{(r)} = n\lambda^{(r)} \boldsymbol{K} \boldsymbol{\alpha}^{(r)}$$

  - Each $\boldsymbol{\alpha}^{(r)}$ that fulfills the equation gives us a eigenvector $\boldsymbol{u}^{(r)}$ of the covariance matrix M' in terms of the data $\boldsymbol{x}_i'$

Solving $\alpha_1, \ldots, \alpha_n$

- Removing $\boldsymbol{K}$ from both sides will only affect the $\boldsymbol{\alpha}^{(r)}$ with zero $\lambda^{(r)}$ (proof omitted), leaving the final form of the eigenvalue system

$$\boldsymbol{K}\boldsymbol{\alpha}^{(r)} = n\lambda^{(r)}\boldsymbol{\alpha}^{(r)}$$

- Since $||\boldsymbol{u}|| = 1$, we require $n\lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha} = 1 \Rightarrow ||\boldsymbol{\alpha}||^2 = 1/n\lambda \Rightarrow ||\boldsymbol{\alpha}|| = \sqrt{1/n\lambda}$

  Proof later

  However, $\boldsymbol{\alpha}^*$ from the eigendecomposition of $\boldsymbol{K}$ has unit length and eigenvalue $\lambda^* = n\lambda^{(r)}$

  To correct for this, $\boldsymbol{\alpha}^{(r)} = \dfrac{\boldsymbol{\alpha}^*}{\sqrt{n\lambda^{(r)}}} = \dfrac{\boldsymbol{\alpha}^*}{\sqrt{n\lambda^*/n}} = \dfrac{\boldsymbol{\alpha}^*}{\sqrt{\lambda^*}}$

- Since $\lambda^{(r)} = \lambda^*/n$, the relative importance of the eigenvectors can be determined from λ^*

Proof for $\|\boldsymbol{u}\| = 1 \Rightarrow n\lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha} = 1$

- Since $\|\boldsymbol{u}\| = 1$

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} = 1$$

$$\left(\sum_i \alpha_i \boldsymbol{x}_i'^{\mathrm{T}}\right)^{\mathrm{T}}\left(\sum_j \alpha_j \boldsymbol{x}_j'^{\mathrm{T}}\right) = 1$$

$$\sum_i \sum_j \alpha_i \alpha_j \, \boldsymbol{x}_i' \boldsymbol{x}_j'^{\mathrm{T}} = 1$$

$$\sum_i \sum_j \alpha_i K_{ij} \alpha_j = 1$$

- Multiply α_i to $\sum_j K_{ij}\,\alpha_j = n\lambda \sum_k \alpha_k$ gives

$$n\lambda \sum_i \sum_k \alpha_i \alpha_k = \sum_i \sum_j \alpha_i K_{ij}\,\alpha_j$$

$$n\lambda \sum_i \sum_k \alpha_i \alpha_k = 1$$

$$n\lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha} = 1$$

Proof for $x^{\mathrm{T}}xu = xux^{\mathrm{T}}$

$$(v^{\mathrm{T}}v)u = \begin{pmatrix} v_1v_1 & \cdots & v_1v_n \\ \vdots & \ddots & \vdots \\ v_nv_1 & \cdots & v_nv_n \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}$$

$$= \begin{pmatrix} v_1v_1u_1 + \cdots + v_1v_nu_n \\ \vdots \\ v_nv_1u_1 + \cdots + v_nv_nu_n \end{pmatrix}$$

$$= \begin{pmatrix} (v_1u_1 + \cdots + v_nu_n)v_1 \\ \vdots \\ (v_1u_1 + \cdots + v_nu_n)v_n \end{pmatrix}$$

$$= (v_1u_1 + \cdots + v_nu_n) \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

Projection of $\phi(x)$ on u

- Given a point y, the projection of $\phi(y)$ on the eigenvector $u^{(r)}$ of M' can be computed using $\boldsymbol{\alpha}^{(r)}$ as

$$\phi(y)u^{(r)} = \sum_{i=1}^{n} \alpha_i^{(r)} \phi(y)^{\mathrm{T}} x_i'$$
$$= \sum_i \alpha_i^{(r)} K(y, x_i)$$

- This allows the principal components to be used for clustering existing datapoints as well as classifying out-of-sample datapoints into the clusters

Normalizing M'

- X' has been assumed to be normalized so far
- To normalize a matrix X', subtract every column with the mean of the column:

$$x^* = x' - \frac{1}{n}\sum_{i=1}^{n} x_i'$$

- The corresponding kernel,

$$\begin{aligned} K^*(x_i, x_j) = x_i^* x_j^* &= \left(x' - \frac{1}{n}\sum_{i=1}^{n} x_i'\right)\left(x' - \frac{1}{n}\sum_{i=1}^{n} x_i'\right) \\ &= K(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n} K(x_i, x_k) \\ &\quad - \frac{1}{n}\sum_{k=1}^{n} K(x_j, x_k) + \frac{1}{n^2}\sum_{l,k=1}^{n} K(x_l, x_k) \end{aligned}$$

Or in matrix notation

$$\boldsymbol{K}^* = \boldsymbol{K} - 2\mathbf{1}_{1/n}\boldsymbol{K} + \mathbf{1}_{1/n}\boldsymbol{K}\mathbf{1}_{1/n}$$

Kernel functions

- A kernel function K implicitly defines a mapping ϕ from an input space to some feature space
- Positive semi-definite functions are those that produce positive semi-definite kernel matrices
  - **Definition**. A symmetric function K is called positive semi-definite over χ if and only if for every set of elements $x_1, \ldots, x_n \in \chi$, the matrix $\boldsymbol{K} = (x_{ij})$ where $x_{ij} = K(x_i, x_j)$ is positive semidefinite
- **Kernel functions must be positive semi-definite**

  Hilbert space (ignore for now)

  - **Theorem**. A mapping ϕ exists for $K: \chi \rightarrow \mathcal{H}$ such that $K(x, x') = \langle \phi(x), \phi(x') \rangle \Leftrightarrow K$ is a positive semi-definite symmetric matrix

Kernel functions

Properties

Symmetric	$K(x, x') = K(x', x)$
Cauchy-Schwarz inequality	$\lvert K(x, x')\rvert \leq \sqrt{K(x, x)K(x', x')}$
Definiteness	$K(x, x) = \lVert\phi(x)\rVert^2 \geq 0$

Kernel property conservation

Sum	K, K' are kernels ⇒ $K + K$' is kernel
Product	K, K' are kernels ⇒ KK' is kernel
Scaling	K is kernel ⇒ αK is kernel for positive real α
Polynomial combination	K is kernel ⇒ $p(K)$ is kernel for polynomial p of degree m with positive coefficients

Kernel functions

- Common kernel functions

Linear	$K(x, x') = xx'^{\mathrm{T}}$
Cosine	$K(x, x') = xx'^{\mathrm{T}} / \lVert x \rVert \lVert x' \rVert$
Gaussian	$K(x, x') = \exp(-\gamma \lVert x - x' \rVert^2)$
Polynomial	$K(x, x') = \left(\gamma x x'^{\mathrm{T}} + c\right)^d$ for $\gamma, c \in \mathbb{R}^+, d \in \mathbb{N}^+$
Sigmoid	$K(x, x') = \tanh\left(\gamma x x'^{\mathrm{T}} + c\right)$ for $\gamma, c \in \mathbb{R}^+$

See http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications for a collection of uncommon kernel functions