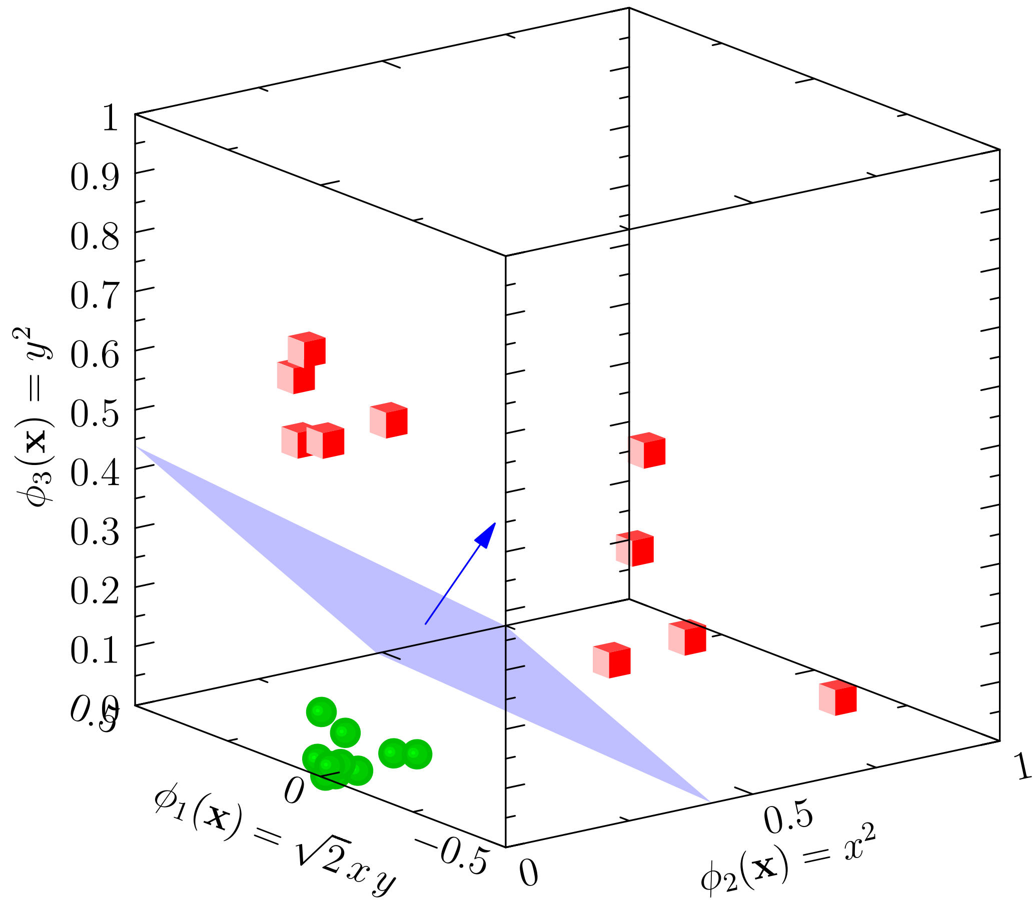
Advanced Machine Learning

Kernel Trick



The Kernel Trick, SVMs, Regression

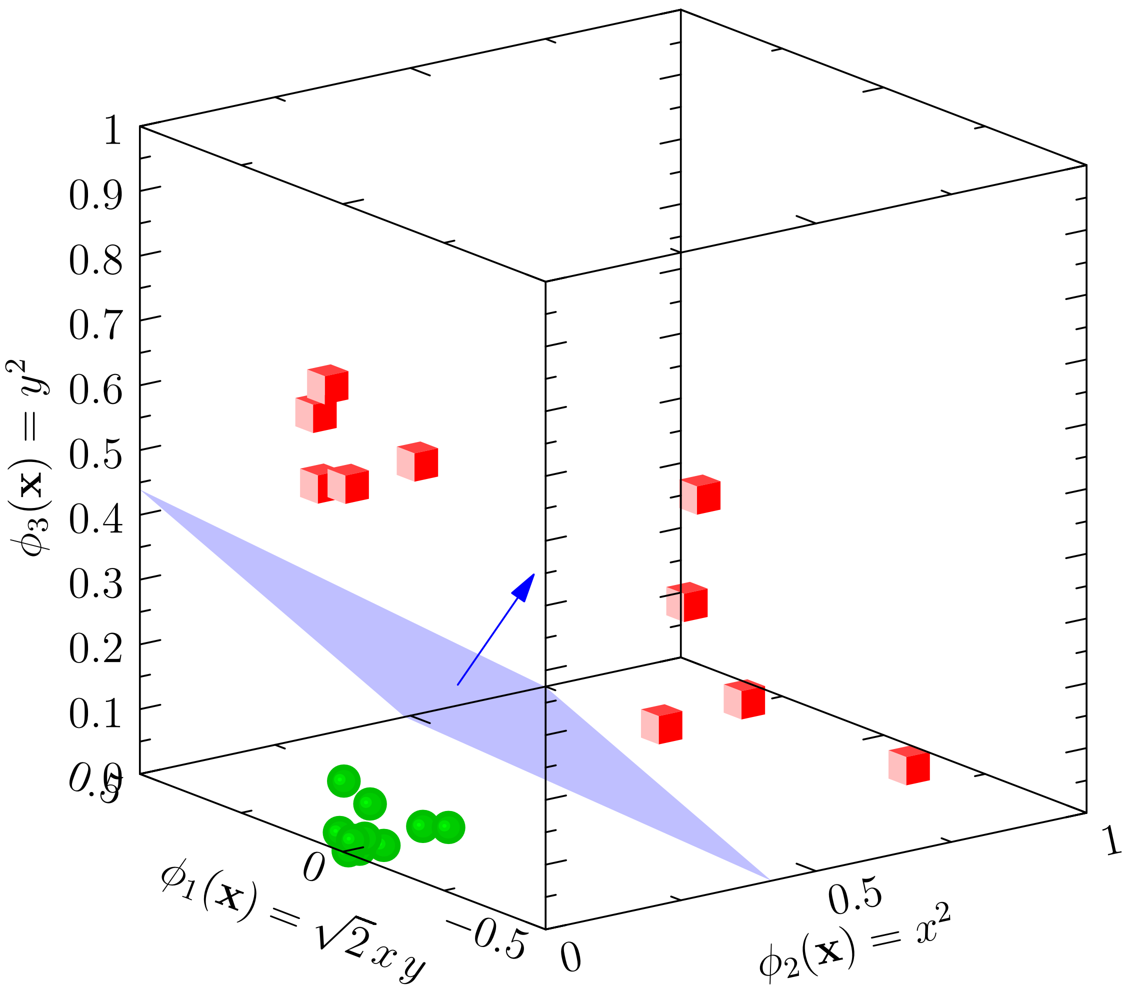
Outline

1. **The Kernel Trick**

2. Positive Semi-Definite Kernels

3. Kernel Properties

4. Beyond Classification



SVM Kernels

- SVM Kernels are functions of two variables that can be factorised

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{y}) = \sum_i \phi^{(i)}(\boldsymbol{x})\, \phi^{(i)}(\boldsymbol{y})$$

- where $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}), \phi^{(2)}(\boldsymbol{x}), \ldots)^{\mathsf{T}}$ and $\phi^{(i)}(\boldsymbol{x})$ are real valued functions of $\boldsymbol{x}$

- $K(\boldsymbol{x}, \boldsymbol{y})$ will be positive semi-definite (because it is an inner-product)

- Furthermore, any positive semi-definite function will factorise

- This factorisation is not always obvious (we return to this later)

SVM Kernels

- SVM Kernels are functions of two variables that can be factorised

$$K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{y}) = \sum_i \phi^{(i)}(\boldsymbol{x})\,\phi^{(i)}(\boldsymbol{y})$$

- where $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}),\, \phi^{(2)}(\boldsymbol{x}),\, \ldots)^{\mathsf{T}}$ and $\phi^{(i)}(\boldsymbol{x})$ are real valued functions of $\boldsymbol{x}$

- $K(\boldsymbol{x},\boldsymbol{y})$ will be positive semi-definite (because it is an inner-product)

- Furthermore, any positive semi-definite function will factorise

- This factorisation is not always obvious (we return to this later)

SVM Kernels

- SVM Kernels are functions of two variables that can be factorised

$$K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{y}) = \sum_i \phi^{(i)}(\boldsymbol{x})\,\phi^{(i)}(\boldsymbol{y})$$

- where $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}),\, \phi^{(2)}(\boldsymbol{x}),\, \ldots)^{\mathsf{T}}$ and $\phi^{(i)}(\boldsymbol{x})$ are real valued functions of $\boldsymbol{x}$

- $K(\boldsymbol{x},\boldsymbol{y})$ will be positive semi-definite (because it is an inner-product)

- Furthermore, any positive semi-definite function will factorise

- This factorisation is not always obvious (we return to this later)

SVM Kernels

- SVM Kernels are functions of two variables that can be factorised

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{y}) = \sum_i \phi^{(i)}(\boldsymbol{x})\, \phi^{(i)}(\boldsymbol{y})$$

- where $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi^{(1)}(\boldsymbol{x}),\, \phi^{(2)}(\boldsymbol{x}),\, \ldots)^{\mathsf{T}}$ and $\phi^{(i)}(\boldsymbol{x})$ are real valued functions of $\boldsymbol{x}$

- $K(\boldsymbol{x}, \boldsymbol{y})$ will be positive semi-definite (because it is an inner-product)

- Furthermore, any positive semi-definite function will factorise

- This factorisation is not always obvious (we return to this later)

Dual Form

- Recall that the dual problem for an SVM is

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_l)$$

- subject to $\sum_{k=1}^{m} y_k \, \alpha_k = 0$ and $0 \leq \alpha_k (\leq C)$

- But since $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_l)$ the dual problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, K(\boldsymbol{x}_k, \boldsymbol{x}_l)$$

- This is the **kernel trick**

Dual Form

- Recall that the dual problem for an SVM is

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_l)$$

- subject to $\sum_{k=1}^{m} y_k \, \alpha_k = 0$ and $0 \leq \alpha_k (\leq C)$
- But since $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_l)$ the dual problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, K(\boldsymbol{x}_k, \boldsymbol{x}_l)$$

- This is the **kernel trick**

Dual Form

- Recall that the dual problem for an SVM is

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k\, \alpha_l\, y_k\, y_l\, \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\, \boldsymbol{\phi}(\boldsymbol{x}_l)$$

- subject to $\sum_{k=1}^{m} y_k\, \alpha_k = 0$ and $0 \le \alpha_k (\le C)$

- But since $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}}\, \boldsymbol{\phi}(\boldsymbol{x}_l)$ the dual problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k\, \alpha_l\, y_k\, y_l\, K(\boldsymbol{x}_k, \boldsymbol{x}_l)$$

- This is the **kernel trick**

Dual Form

- Recall that the dual problem for an SVM is

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \, \boldsymbol{\phi}(\boldsymbol{x}_l)$$

- subject to $\sum_{k=1}^{m} y_k \, \alpha_k = 0$ and $0 \leq \alpha_k (\leq C)$
- But since $K(\boldsymbol{x}_k, \boldsymbol{x}_l) = \boldsymbol{\phi}(\boldsymbol{x}_k)^{\mathsf{T}} \, \boldsymbol{\phi}(\boldsymbol{x}_l)$ the dual problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{m} \alpha_k \, \alpha_l \, y_k \, y_l \, K(\boldsymbol{x}_k, \boldsymbol{x}_l)$$

- This is the **kernel trick**—we never have to compute $\boldsymbol{\phi}(\boldsymbol{x})$!

Classifying New Data

- Having trained the SVM we now have to use it
- Given a new input $\boldsymbol{x}$ we decide on the class

$$y = \operatorname{sgn}\big(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) - b\big) \qquad \text{but} \qquad \boldsymbol{w} = \sum_{k=1}^{m} \alpha_k \, y_k \, \boldsymbol{\phi}(\boldsymbol{x}_k)$$

- In the dual representation this becomes

$$\operatorname{sgn}\left(\sum_{k=1}^{m} \alpha_k \, y_k \, K(\boldsymbol{x}_k, \boldsymbol{x}) - b \right)$$

where we only need to sum over the non-zero α_k (i.e. the support vectors SVs)

Classifying New Data

- Having trained the SVM we now have to use it
- Given a new input $\boldsymbol{x}$ we decide on the class

$$y = \mathrm{sgn}\big(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}) - b\big) \qquad \text{but} \qquad \boldsymbol{w} = \sum_{k=1}^{m} \alpha_k\, y_k\, \boldsymbol{\phi}(\boldsymbol{x}_k)$$

- In the dual representation this becomes

$$\mathrm{sgn}\left(\sum_{k=1}^{m} \alpha_k\, y_k\, K(\boldsymbol{x}_k, \boldsymbol{x}) - b\right)$$

where we only need to sum over the non-zero α_k (i.e. the support vectors SVs)

Classifying New Data

- Having trained the SVM we now have to use it
- Given a new input $\boldsymbol{x}$ we decide on the class

$$y = \operatorname{sgn}\big(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}) - b\big) \qquad \text{but} \qquad \boldsymbol{w} = \sum_{k=1}^{m} \alpha_k\, y_k\, \boldsymbol{\phi}(\boldsymbol{x}_k)$$

- In the dual representation this becomes

$$\operatorname{sgn}\left(\sum_{k=1}^{m} \alpha_k\, y_k\, K(\boldsymbol{x}_k, \boldsymbol{x}) - b \right)$$

where we only need to sum over the non-zero α_k (i.e. the support vectors SVs)

Classifying New Data

- Having trained the SVM we now have to use it
- Given a new input $\boldsymbol{x}$ we decide on the class

$$y = \operatorname{sgn}\big(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}) - b\big) \qquad \text{but} \qquad \boldsymbol{w} = \sum_{k=1}^{m} \alpha_k \, y_k \, \boldsymbol{\phi}(\boldsymbol{x}_k)$$

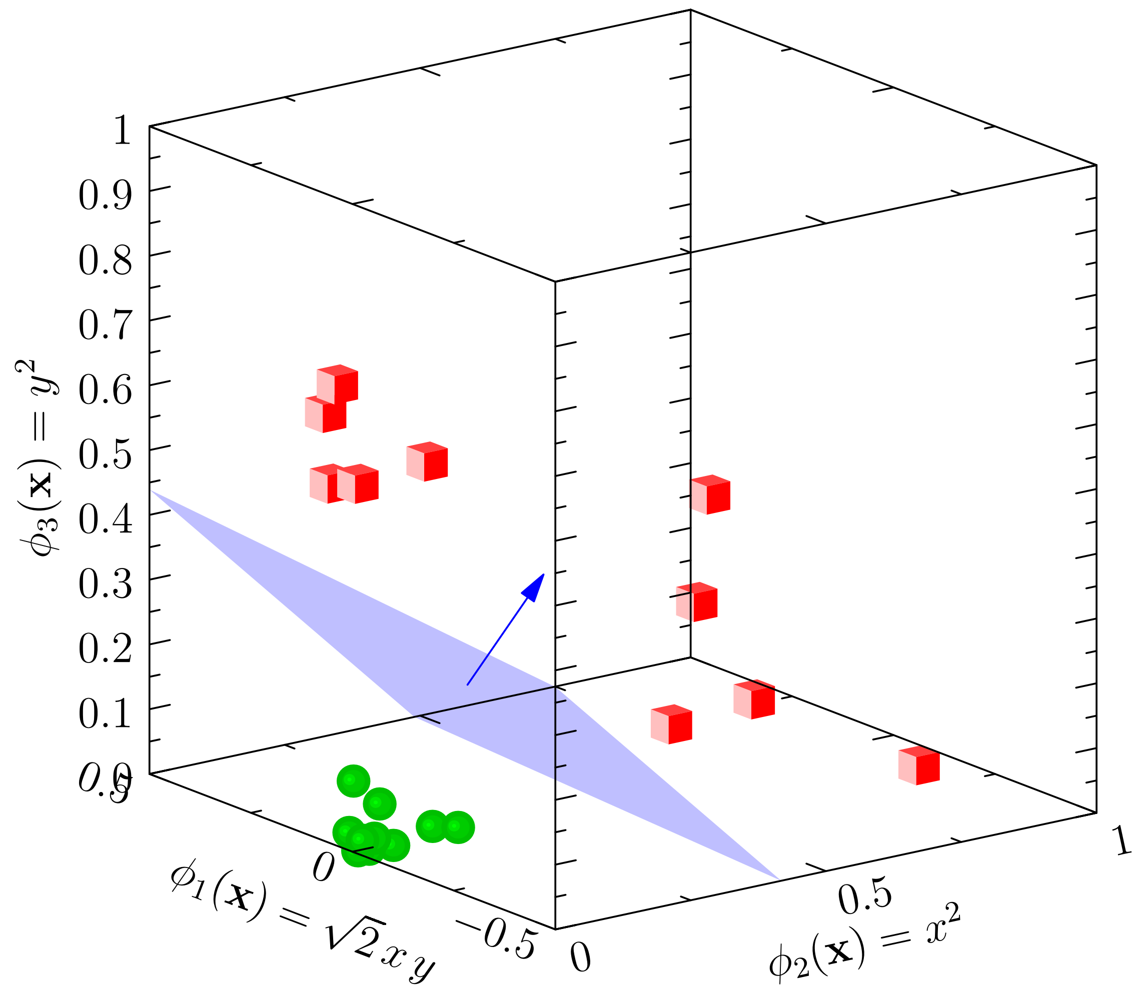
- In the dual representation this becomes

$$\operatorname{sgn}\left(\sum_{k=1}^{m} \alpha_k \, y_k \, K(\boldsymbol{x}_k, \boldsymbol{x}) - b \right)$$

where we only need to sum over the non-zero α_k (i.e. the support vectors SVs)

Outline

1. The Kernel Trick
2. **Positive Semi-Definite Kernels**
3. Kernel Properties
4. Beyond Classification



Recap on Eigen Systems

- Recall for a symmetric $(n \times n)$ matrix $\boldsymbol{M}$ an eigenvector, $\boldsymbol{v}$

$$\boldsymbol{M}\,\boldsymbol{v} = \lambda \boldsymbol{v}$$

- There are n independent eigenvectors $\boldsymbol{v}^{(i)}$ with real eigenvalues $\lambda^{(i)}$

- The eigenvectors are orthogonal so that $\boldsymbol{v}^{(i)\mathsf{T}}\boldsymbol{v}^{(j)} = 0$ if $i \neq j$

- Forming a matrix of eigenvectors $\mathbf{V} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots \boldsymbol{v}^{(n)})$ the matrix satisfies

$$\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\,\mathbf{V}^{\mathsf{T}} = \mathbf{I}$$

- Such matrices are said to be orthogonal

Recap on Eigen Systems

- Recall for a symmetric $(n \times n)$ matrix $\mathbf{M}$ an eigenvector, $\boldsymbol{v}$

$$\mathbf{M}\,\boldsymbol{v} = \lambda \boldsymbol{v}$$

- There are n independent eigenvectors $\boldsymbol{v}^{(i)}$ with real eigenvalues $\lambda^{(i)}$

- The eigenvectors are orthogonal so that $\boldsymbol{v}^{(i)\mathsf{T}} \boldsymbol{v}^{(j)} = 0$ if $i \neq j$

- Forming a matrix of eigenvectors $\mathbf{V} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots \boldsymbol{v}^{(n)})$ the matrix satisfies

$$\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\,\mathbf{V}^{\mathsf{T}} = \mathbf{I}$$

- Such matrices are said to be orthogonal

Recap on Eigen Systems

- Recall for a symmetric $(n \times n)$ matrix $\mathbf{M}$ an eigenvector, $\boldsymbol{v}$

$$\mathbf{M}\,\boldsymbol{v} = \lambda \boldsymbol{v}$$

- There are n independent eigenvectors $\boldsymbol{v}^{(i)}$ with real eigenvalues $\lambda^{(i)}$

- The eigenvectors are orthogonal so that $\boldsymbol{v}^{(i)\mathsf{T}}\boldsymbol{v}^{(j)} = 0$ if $i \neq j$

- Forming a matrix of eigenvectors $\mathbf{V} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots \boldsymbol{v}^{(n)})$ the matrix satisfies

$$\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\,\mathbf{V}^{\mathsf{T}} = \mathbf{I}$$

- Such matrices are said to be orthogonal

Recap on Eigen Systems

- Recall for a symmetric $(n \times n)$ matrix $\mathbf{M}$ an eigenvector, $\boldsymbol{v}$

$$\mathbf{M}\,\boldsymbol{v} = \lambda \boldsymbol{v}$$

- There are n independent eigenvectors $\boldsymbol{v}^{(i)}$ with real eigenvalues $\lambda^{(i)}$
- The eigenvectors are orthogonal so that $\boldsymbol{v}^{(i)\mathsf{T}} \boldsymbol{v}^{(j)} = 0$ if $i \neq j$
- Forming a matrix of eigenvectors $\mathbf{V} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots\, \boldsymbol{v}^{(n)})$ the matrix satisfies

$$\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\,\mathbf{V}^{\mathsf{T}} = \mathbf{I}$$

- Such matrices are said to be orthogonal

Recap on Eigen Systems

- Recall for a symmetric $(n \times n)$ matrix $\mathbf{M}$ an eigenvector, $\boldsymbol{v}$

$$\mathbf{M}\,\boldsymbol{v} = \lambda \boldsymbol{v}$$

- There are n independent eigenvectors $\boldsymbol{v}^{(i)}$ with real eigenvalues $\lambda^{(i)}$

- The eigenvectors are orthogonal so that $\boldsymbol{v}^{(i)\mathsf{T}}\boldsymbol{v}^{(j)} = 0$ if $i \neq j$

- Forming a matrix of eigenvectors $\mathbf{V} = (\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots \boldsymbol{v}^{(n)})$ the matrix satisfies

$$\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{V}\,\mathbf{V}^{\mathsf{T}} = \mathbf{I}$$

- Such matrices are said to be orthogonal

Eigen Decomposition

- From the eigenvalue equation $\mathbf{M}\,\boldsymbol{v}^{(k)} = \lambda^{(k)}\boldsymbol{v}^{(k)}$

$$\mathbf{M}\mathbf{V} = \mathbf{V}\,\boldsymbol{\Lambda} \qquad \text{where} \quad \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

- Multiplying on the right by $\mathbf{V}^{\mathsf{T}}$ we get

$$\mathbf{M} = \mathbf{V}\,\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{T}} = \sum_{k=1}^{n} \lambda^{(k)}\,\boldsymbol{v}^{(k)}\,\boldsymbol{v}^{(k)\mathsf{T}}$$

Or

$$M_{ij} = \sum_{k=1}^{n} \lambda^{(k)}\,v_i^{(k)}\,v_j^{(k)}$$

Eigen Decomposition

- From the eigenvalue equation $\mathbf{M}\,\boldsymbol{v}^{(k)} = \lambda^{(k)}\boldsymbol{v}^{(k)}$

$$\mathbf{M}\mathbf{V} = \mathbf{V}\,\boldsymbol{\Lambda} \qquad \text{where} \quad \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

- Multiplying on the right by $\mathbf{V}^{\mathsf{T}}$ we get

$$\mathbf{M} = \mathbf{V}\,\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{T}} = \sum_{k=1}^{n} \lambda^{(k)}\,\boldsymbol{v}^{(k)}\,\boldsymbol{v}^{(k)\mathsf{T}}$$

Or

$$M_{ij} = \sum_{k=1}^{n} \lambda^{(k)}\,v_i^{(k)}\,v_j^{(k)}$$

Eigen Decomposition

- From the eigenvalue equation $\mathbf{M}\,\boldsymbol{v}^{(k)} = \lambda^{(k)}\boldsymbol{v}^{(k)}$

$$\mathbf{M}\mathbf{V} = \mathbf{V}\,\boldsymbol{\Lambda} \qquad \text{where} \quad \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

- Multiplying on the right by $\mathbf{V}^{\mathsf{T}}$ we get

$$\mathbf{M} = \mathbf{V}\,\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{T}} = \sum_{k=1}^{n} \lambda^{(k)}\,\boldsymbol{v}^{(k)}\,\boldsymbol{v}^{(k)\mathsf{T}}$$

Or

$$M_{ij} = \sum_{k=1}^{n} \lambda^{(k)}\,v_i^{(k)}\,v_j^{(k)}$$

Eigen Decomposition

- From the eigenvalue equation $\mathbf{M}\,\boldsymbol{v}^{(k)} = \lambda^{(k)}\boldsymbol{v}^{(k)}$

$$\mathbf{M}\mathbf{V} = \mathbf{V}\,\boldsymbol{\Lambda} \qquad \text{where} \quad \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

- Multiplying on the right by $\mathbf{V}^{\mathsf{T}}$ we get

$$\mathbf{M} = \mathbf{V}\,\boldsymbol{\Lambda}\mathbf{V}^{\mathsf{T}} = \sum_{k=1}^{n} \lambda^{(k)}\,\boldsymbol{v}^{(k)}\,\boldsymbol{v}^{(k)\mathsf{T}}$$

Or

$$M_{ij} = \sum_{k=1}^{n} \lambda^{(k)}\,v_i^{(k)}\,v_j^{(k)} = \sum_{k=1}^{n} u_i^{(k)}\,u_j^{(k)} = \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle$$

$$u_i^{(k)} = \sqrt{\lambda^{(k)}}\,v_i^{(k)}$$

Eigenfunctions

- By analogy for a symmetric function of two variables we can define an *eigenfunction*

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi(\boldsymbol{y})\, \mathrm{d}\, \boldsymbol{y} = \lambda\, \psi(\boldsymbol{x})$$

- In general there will be a denumerable set of eigenfunctions $\psi^{(k)}(\boldsymbol{x})$ where

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y})$$

- This is known as Mercer's theorem

Eigenfunctions

- By analogy for a symmetric function of two variables we can define an *eigenfunction*

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi(\boldsymbol{y})\, \mathrm{d}\, \boldsymbol{y} = \lambda\, \psi(\boldsymbol{x})$$

- In general there will be a denumerable set of eigenfunctions $\psi^{(k)}(\boldsymbol{x})$ where

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y})$$

- This is known as Mercer's theorem

Eigenfunctions

- By analogy for a symmetric function of two variables we can define an *eigenfunction*

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi(\boldsymbol{y})\, \mathrm{d}\, \boldsymbol{y} = \lambda\, \psi(\boldsymbol{x})$$

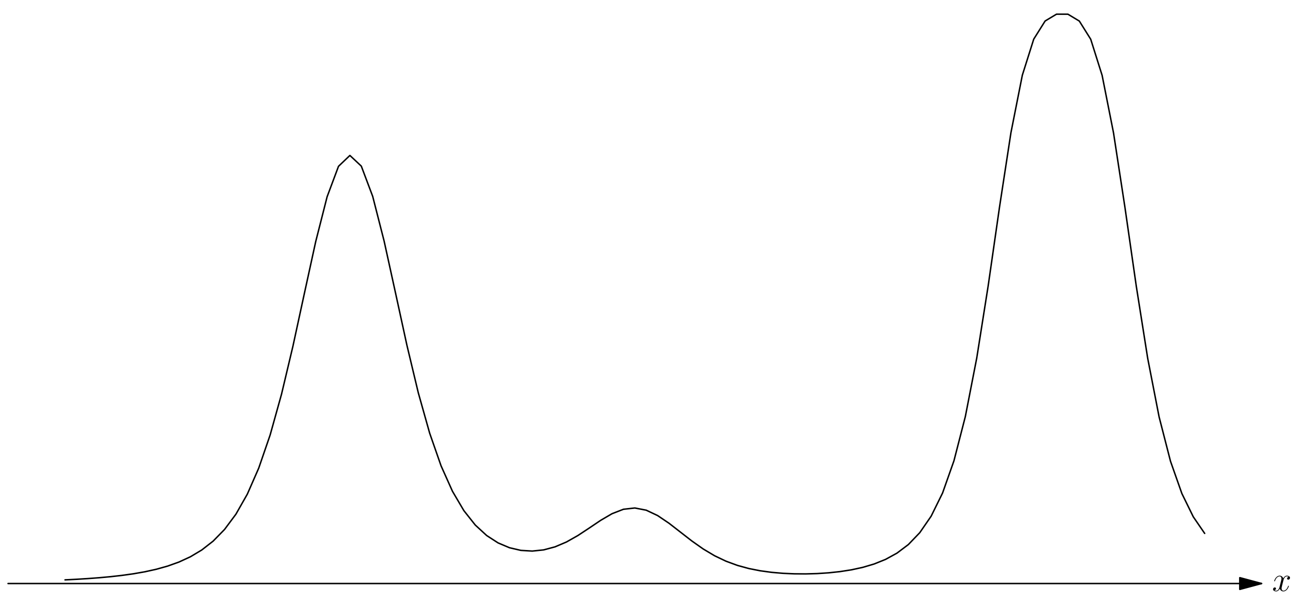
- In general there will be a denumerable set of eigenfunctions $\psi^{(k)}(\boldsymbol{x})$ where

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y})$$

- This is known as Mercer's theorem

Limit Process

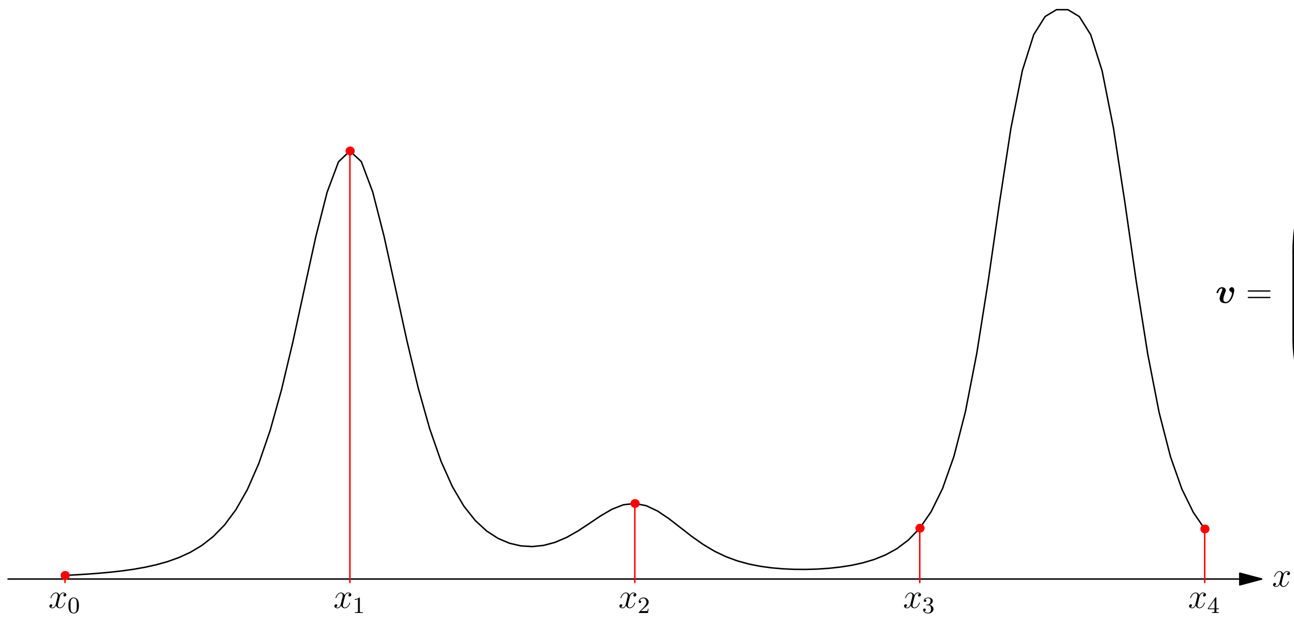
- Consider sampling a function at a set of points



- In the limit where the number of sample points goes to infinity the vector more closely approximates a function
- Instead of the indices being numbers we use $k \leftarrow x_k$

Limit Process

- Consider sampling a function at a set of points

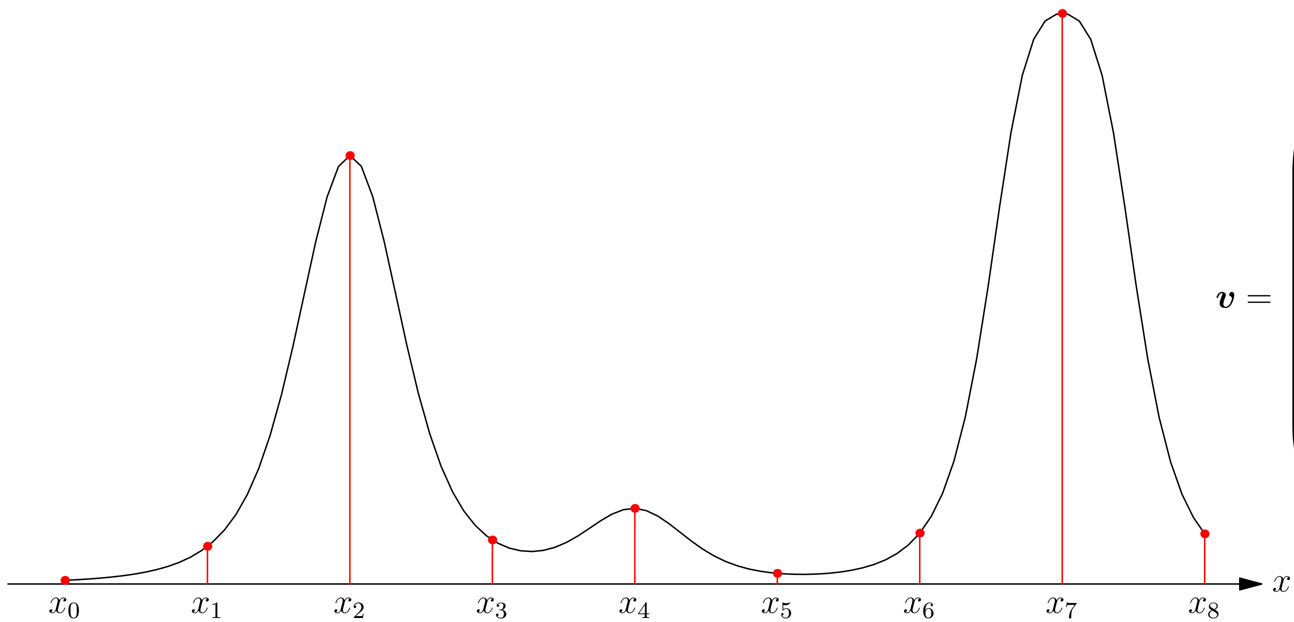


$$\boldsymbol{v} = \begin{pmatrix} 0.0512 \\ 6.01 \\ 1.06 \\ 0.714 \\ 0.703 \end{pmatrix}$$

- In the limit where the number of sample points goes to infinity the vector more closely approximates a function
- Instead of the indices being numbers we use $k \leftarrow x_k$

Limit Process

- Consider sampling a function at a set of points



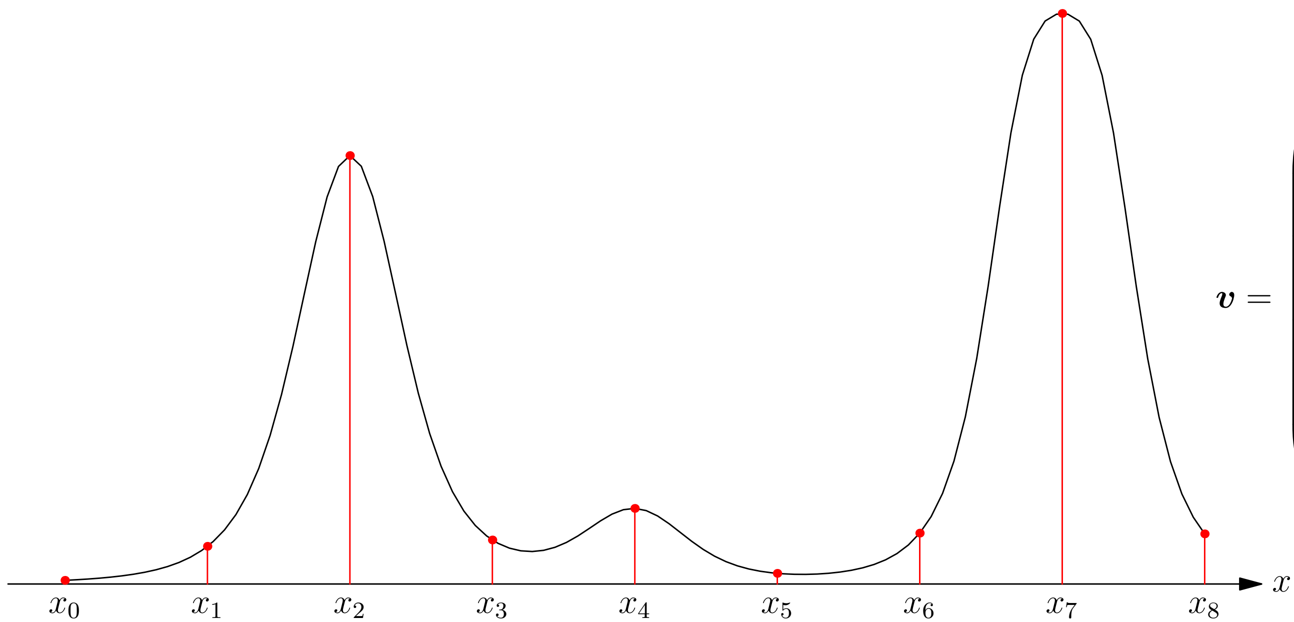
$$\boldsymbol{v} = \begin{pmatrix} 0.0512 \\ 0.529 \\ 6.01 \\ 0.617 \\ 1.06 \\ 0.149 \\ 0.714 \\ 8 \\ 0.703 \end{pmatrix}$$

- In the limit where the number of sample points goes to infinity the vector more closely approximates a function

- Instead of the indices being numbers we use $k \leftarrow x_k$

Limit Process

- Consider sampling a function at a set of points



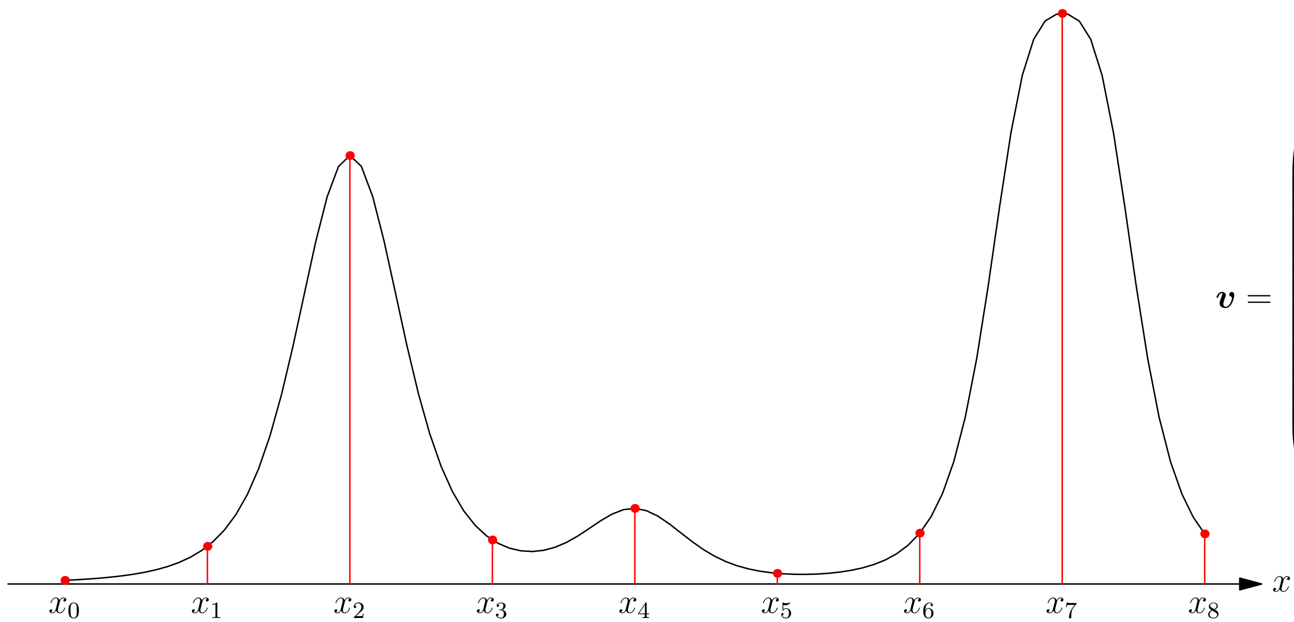
$$\boldsymbol{v} = \begin{pmatrix} 0.0512 \\ 0.529 \\ 6.01 \\ 0.617 \\ 1.06 \\ 0.149 \\ 0.714 \\ 8 \\ 0.703 \end{pmatrix}$$

- In the limit where the number of sample points goes to infinity the vector more closely approximates a function

- Instead of the indices being numbers we use $k \leftarrow x_k$

Limit Process

- Consider sampling a function at a set of points



$$\boldsymbol{v} = \begin{pmatrix} 0.0512 \\ 0.529 \\ 6.01 \\ 0.617 \\ 1.06 \\ 0.149 \\ 0.714 \\ 8 \\ 0.703 \end{pmatrix}$$

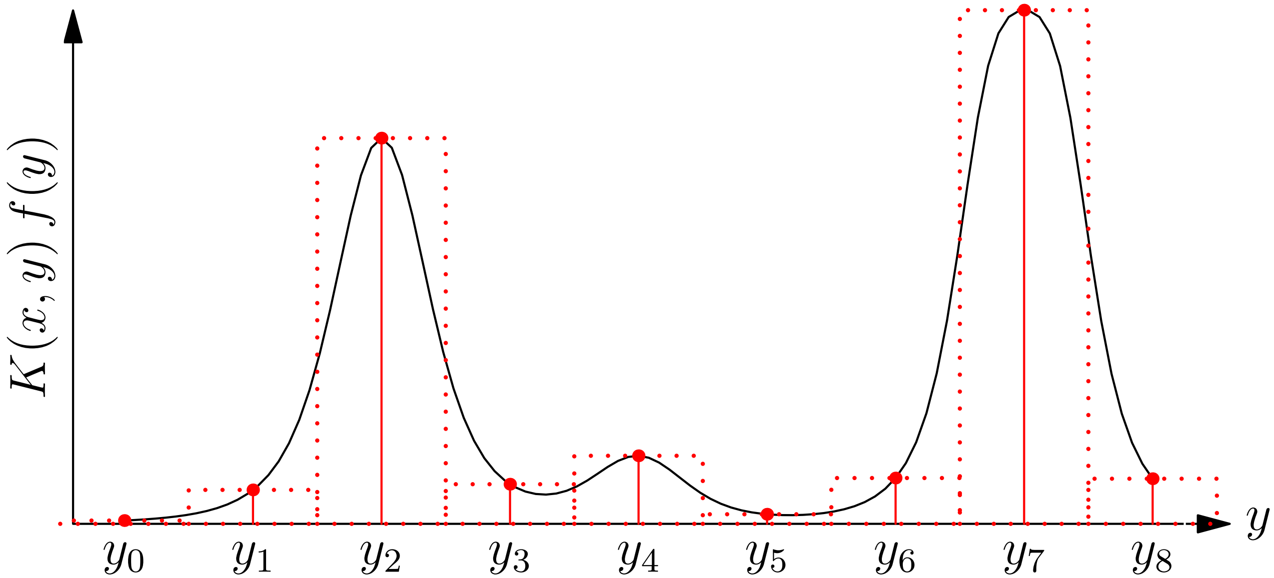
- In the limit where the number of sample points goes to infinity the vector more closely approximates a function

- Instead of the indices being numbers we use $k \leftarrow x_k$

Linear Operators

- Recall a linear function $\mathcal{T}[f(x)]$ can be represented by a kernel

$$\mathcal{T}[f(x)] = \int_{y \in \mathcal{I}} K(x,y)\, f(y)\, \mathrm{d}y$$

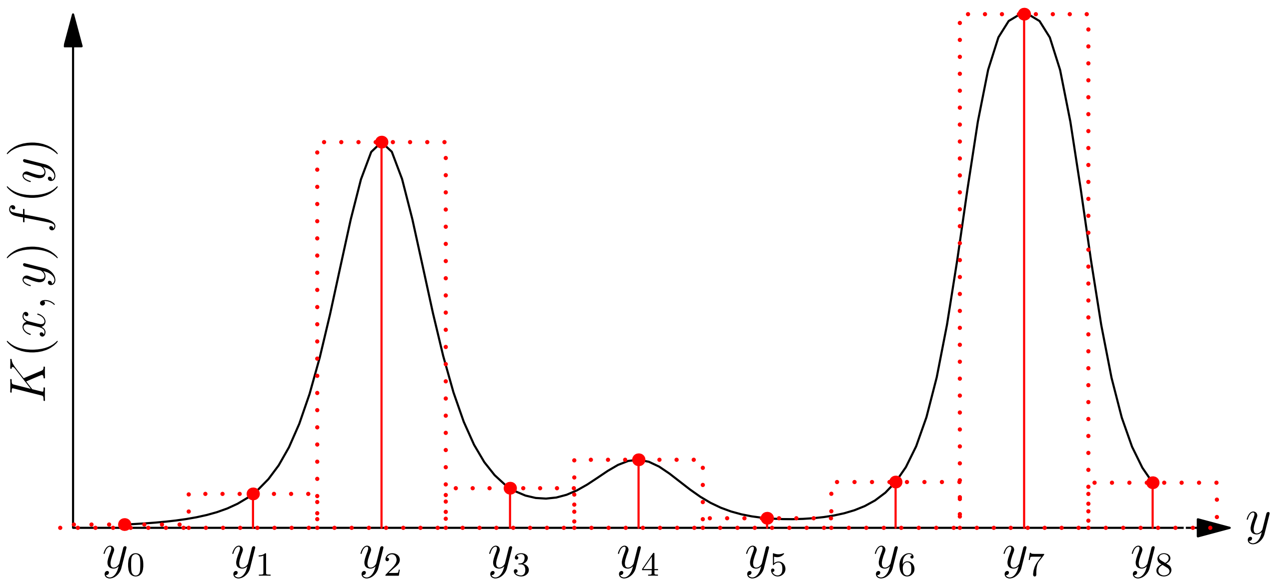


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Linear Operators

- Recall a linear function $\mathcal{T}[f(x)]$ can be represented by a kernel

$$\mathcal{T}[f(x)] = \int_{y\in\mathcal{I}} K(x,y)\, f(y)\, \mathrm{d}y \approx \Delta \sum_{j=1}^{n} K(x,y_j)\, f(y_j)$$

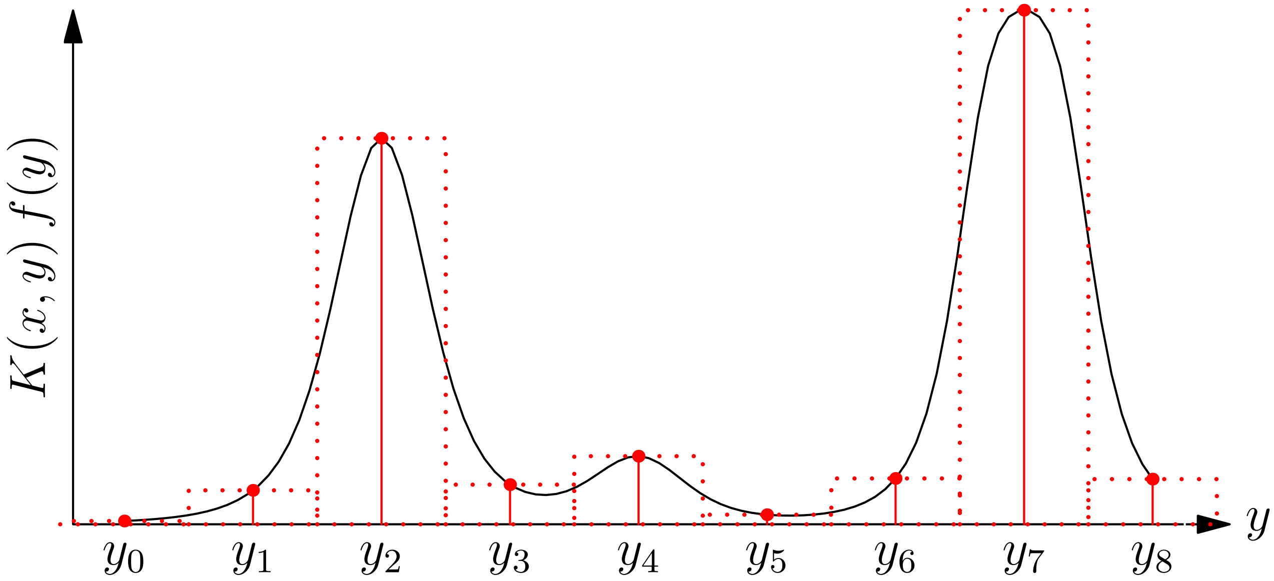


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Linear Operators

- Recall a linear function $\mathcal{T}[f(x)]$ can be represented by a kernel

$$\mathcal{T}[f(x)] = \int_{y \in \mathcal{I}} K(x,y)\, f(y)\, \mathrm{d}y \approx \Delta \sum_{j=1}^{n} K(x, y_j)\, f(y_j)$$

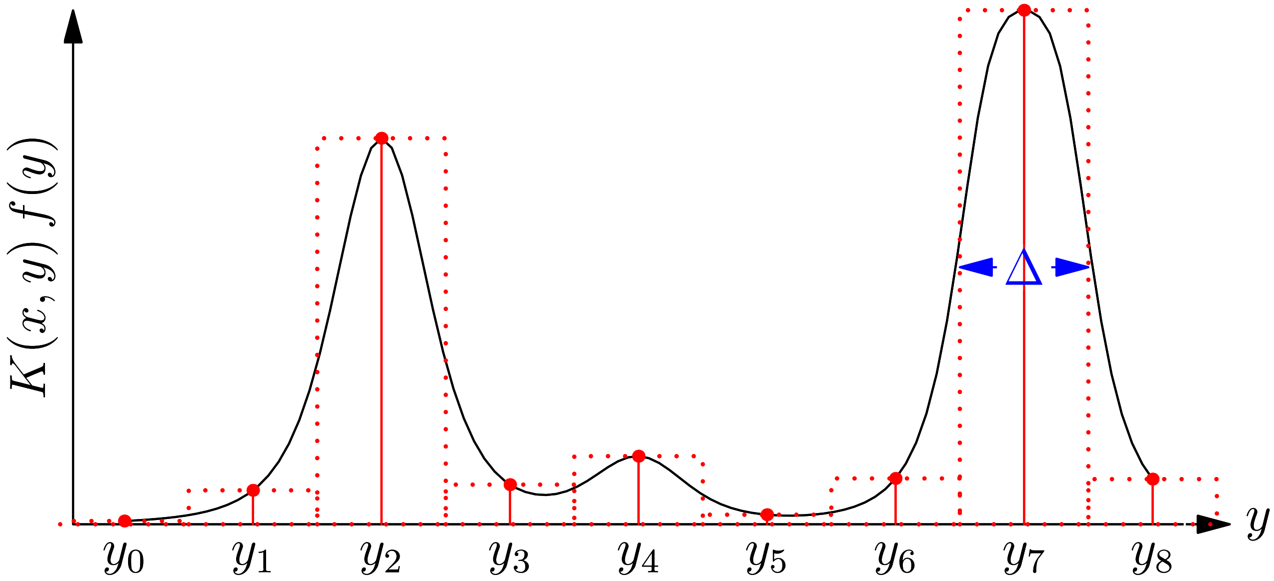


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Linear Operators

- Recall a linear function $\mathcal{T}[f(x)]$ can be represented by a kernel

$$\mathcal{T}[f(x)] = \int_{y \in \mathcal{I}} K(x,y)\, f(y)\, \mathrm{d}y \approx \Delta \sum_{j=1}^{n} K(x, y_j)\, f(y_j)$$

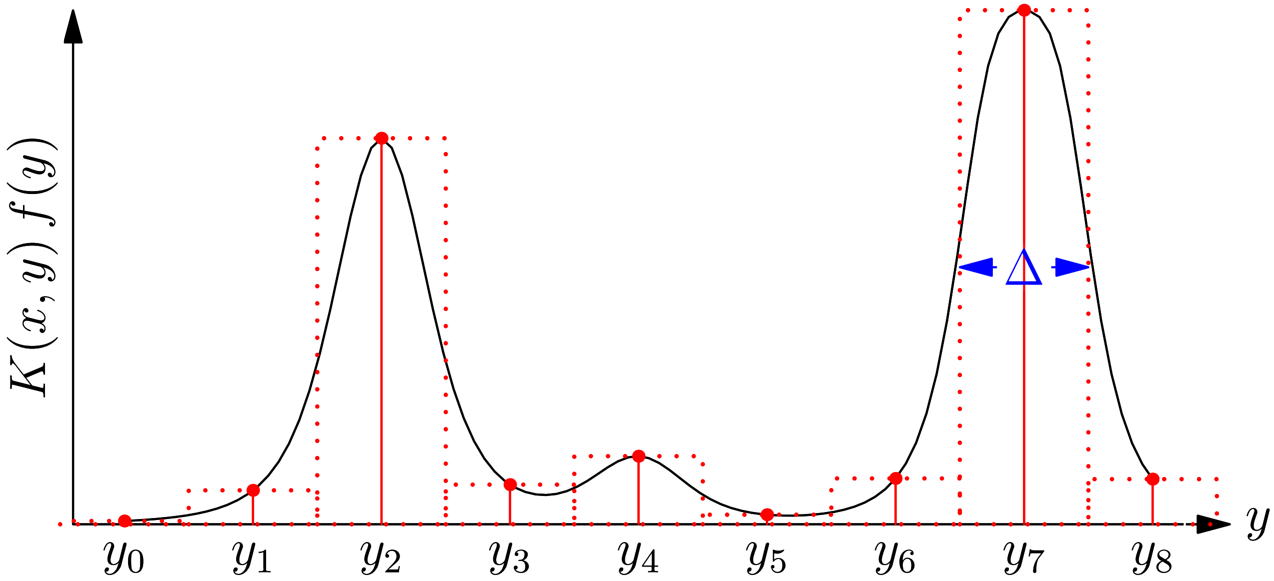


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Linear Operators

- Recall a linear function $\mathcal{T}[f(x)]$ can be represented by a kernel

$$\mathcal{T}[f(x)] = \int_{y\in\mathcal{I}} K(x,y)\, f(y)\, \mathrm{d}y \approx \Delta \sum_{j=1}^{n} K(x,y_j)\, f(y_j)$$



This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

SVM Kernels

- If we define $\phi^{(k)}(\boldsymbol{x}) = \sqrt{\lambda^{(k)}}\psi^{(k)}(\boldsymbol{x})$ then

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- This is the definition of a SVM kernel we started with
- Note that for $\phi^{(k)}(\boldsymbol{x})$ to be real $\lambda^{(k)} \geq 0$ for all k
- If $\lambda^{(k)} < 0$ the "distance" between points in the extended feature space can be negative!
- If we use a kernel that isn't positive semi-definite then the Hessian of the dual objective function will not be negative semi-definite and there will be a maximum where $\boldsymbol{\alpha}$ diverges

SVM Kernels

- If we define $\phi^{(k)}(\boldsymbol{x}) = \sqrt{\lambda^{(k)}}\psi^{(k)}(\boldsymbol{x})$ then

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- This is the definition of a SVM kernel we started with

- Note that for $\phi^{(k)}(\boldsymbol{x})$ to be real $\lambda^{(k)} \geq 0$ for all k

- If $\lambda^{(k)} < 0$ the "distance" between points in the extended feature space can be negative!

- If we use a kernel that isn't positive semi-definite then the Hessian of the dual objective function will not be negative semi-definite and there will be a maximum where $\boldsymbol{\alpha}$ diverges

SVM Kernels

- If we define $\phi^{(k)}(\boldsymbol{x}) = \sqrt{\lambda^{(k)}}\psi^{(k)}(\boldsymbol{x})$ then

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- This is the definition of a SVM kernel we started with

- Note that for $\phi^{(k)}(\boldsymbol{x})$ to be real $\lambda^{(k)} \geq 0$ for all k

- If $\lambda^{(k)} < 0$ the "distance" between points in the extended feature space can be negative!

- If we use a kernel that isn't positive semi-definite then the Hessian of the dual objective function will not be negative semi-definite and there will be a maximum where $\boldsymbol{\alpha}$ diverges

SVM Kernels

- If we define $\phi^{(k)}(\boldsymbol{x}) = \sqrt{\lambda^{(k)}}\psi^{(k)}(\boldsymbol{x})$ then

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- This is the definition of a SVM kernel we started with
- Note that for $\phi^{(k)}(\boldsymbol{x})$ to be real $\lambda^{(k)} \geq 0$ for all k
- If $\lambda^{(k)} < 0$ the "distance" between points in the extended feature space can be negative!
- If we use a kernel that isn't positive semi-definite then the Hessian of the dual objective function will not be negative semi-definite and there will be a maximum where $\boldsymbol{\alpha}$ diverges

SVM Kernels

- If we define $\phi^{(k)}(\boldsymbol{x}) = \sqrt{\lambda^{(k)}}\psi^{(k)}(\boldsymbol{x})$ then

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \lambda^{(k)}\, \psi^{(k)}(\boldsymbol{x})\, \psi^{(k)}(\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- This is the definition of a SVM kernel we started with

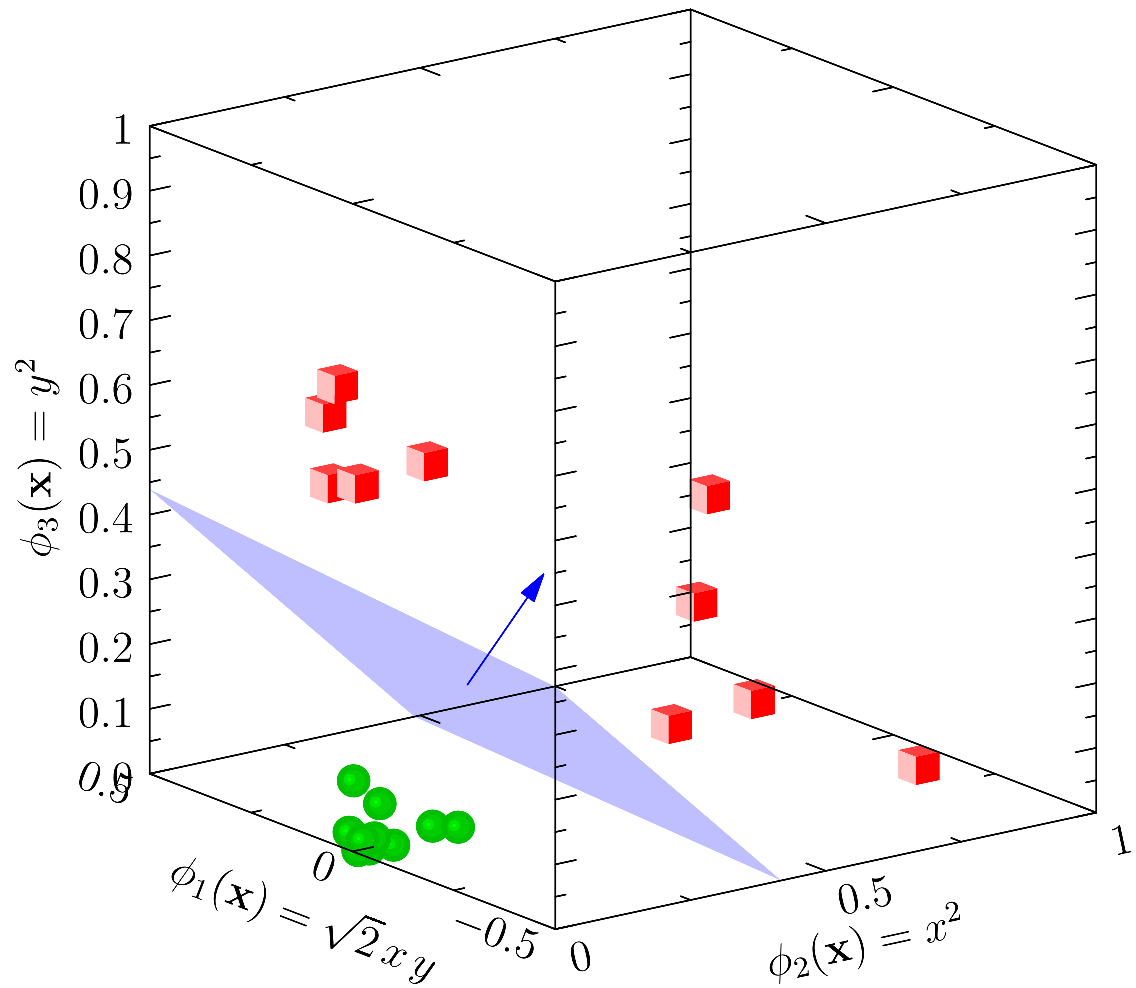
- Note that for $\phi^{(k)}(\boldsymbol{x})$ to be real $\lambda^{(k)} \geq 0$ for all k

- If $\lambda^{(k)} < 0$ the "distance" between points in the extended feature space can be negative!

- If we use a kernel that isn't positive semi-definite then the Hessian of the dual objective function will not be negative semi-definite and there will be a maximum where $\boldsymbol{\alpha}$ diverges

Outline

1. The Kernel Trick
2. Positive Semi-Definite Kernels
3. **Kernel Properties**
4. Beyond Classification



Positive Semi-Definite Kernels

- Kernels (or matrices) that have eigenvalues $\lambda^{(k)} \geq 0$ are called positive semi-definite
- (If the eigenvalues are strictly positive $\lambda^{(k)} > 0$ the kernels or matrices are called positive definite)
- Positive semi-definite kernels can always be decomposed into a sum of real functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

Positive Semi-Definite Kernels

- Kernels (or matrices) that have eigenvalues $\lambda^{(k)} \geq 0$ are called positive semi-definite
- (If the eigenvalues are strictly positive $\lambda^{(k)} > 0$ the kernels or matrices are called positive definite)
- Positive semi-definite kernels can always be decomposed into a sum of real functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

Positive Semi-Definite Kernels

- Kernels (or matrices) that have eigenvalues $\lambda^{(k)} \geq 0$ are called positive semi-definite
- (If the eigenvalues are strictly positive $\lambda^{(k)} > 0$ the kernels or matrices are called positive definite)
- Positive semi-definite kernels can always be decomposed into a sum of real functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$$

- An immediate consequence is that for any function $f(\boldsymbol{x})$

$$\int f(\boldsymbol{x})\,K(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} = \int f(\boldsymbol{x})\,\sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

$$= \sum_k \left(\int f(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\right)^2 \geq 0$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$$

- An immediate consequence is that for any function $f(\boldsymbol{x})$

$$\int f(\boldsymbol{x})\,K(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} = \int f(\boldsymbol{x})\,\sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}$$

$$= \sum_k \left(\int f(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \right)^2 \geq 0$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

- An immediate consequence is that for any function $f(\boldsymbol{x})$

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} = \int f(\boldsymbol{x}) \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}$$

$$= \sum_k \left(\int f(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \right)^2 \geq 0$$

Positive Semi-Definiteness

- The following statements are equivalent
  - ⋆ $K(\boldsymbol{x},\boldsymbol{y})$ is positive semi-definite (written $K(\boldsymbol{x},\boldsymbol{y}) \succeq 0$)
  - ⋆ The eigenvalues of $K(\boldsymbol{x},\boldsymbol{y})$ are non-negative
  - ⋆ The kernel can be written

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$$

    where the $\phi^{(k)}(\boldsymbol{x})$'s are real functions
  - ⋆ For any real function $f(x)$

$$\int f(\boldsymbol{x})\,K(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\,\boldsymbol{x}\,\mathrm{d}\,\boldsymbol{y} \geq 0$$

Positive Semi-Definiteness

- The following statements are equivalent
  - ⋆ $K(\boldsymbol{x}, \boldsymbol{y})$ is positive semi-definite (written $K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$)
  - ⋆ The eigenvalues of $K(\boldsymbol{x}, \boldsymbol{y})$ are non-negative
  - ⋆ The kernel can be written

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

    where the $\phi^{(k)}(\boldsymbol{x})$'s are real functions
  - ⋆ For any real function $f(x)$

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} \geq 0$$

Positive Semi-Definiteness

- The following statements are equivalent
  - ⋆ $K(\boldsymbol{x}, \boldsymbol{y})$ is positive semi-definite (written $K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$)
  - ⋆ The eigenvalues of $K(\boldsymbol{x}, \boldsymbol{y})$ are non-negative
  - ⋆ The kernel can be written

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

    where the $\phi^{(k)}(\boldsymbol{x})$'s are real functions
  - ⋆ For any real function $f(x)$

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} \geq 0$$

Positive Semi-Definiteness

- The following statements are equivalent
  - ⋆ $K(\boldsymbol{x},\boldsymbol{y})$ is positive semi-definite (written $K(\boldsymbol{x},\boldsymbol{y}) \succeq 0$)
  - ⋆ The eigenvalues of $K(\boldsymbol{x},\boldsymbol{y})$ are non-negative
  - ⋆ The kernel can be written

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_k \phi^{(k)}(\boldsymbol{x})\, \phi^{(k)}(\boldsymbol{y})$$

    where the $\phi^{(k)}(\boldsymbol{x})$'s are real functions
  - ⋆ For any real function $f(x)$

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x},\boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} \geq 0$$

Adding Kernels

- We can construct SVM kernels from other kernels

- If $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})$

$$
\begin{aligned}
Q &= \int f(\boldsymbol{x})\, K_3(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x}) \, \big(K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})\big)\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x})\, K_1(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} + \int f(\boldsymbol{x})\, K_2(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \geq 0
\end{aligned}
$$

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $c\, K(\boldsymbol{x}, \boldsymbol{y})$ for $c > 0$

Adding Kernels

- We can construct SVM kernels from other kernels
- If $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})$

$$\begin{aligned}
Q &= \int f(\boldsymbol{x})\, K_3(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x})\, \big(K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})\big)\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x})\, K_1(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} + \int f(\boldsymbol{x})\, K_2(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \geq 0
\end{aligned}$$

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $c\, K(\boldsymbol{x}, \boldsymbol{y})$ for $c > 0$

Adding Kernels

- We can construct SVM kernels from other kernels

- If $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})$

$$
\begin{aligned}
Q &= \int f(\boldsymbol{x})\, K_3(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x})\, \big(K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})\big)\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \\
&= \int f(\boldsymbol{x})\, K_1(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} + \int f(\boldsymbol{x})\, K_2(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y} \geq 0
\end{aligned}
$$

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $c\,K(\boldsymbol{x}, \boldsymbol{y})$ for $c > 0$

Product of Kernels

- If $K_1(\boldsymbol{x},\boldsymbol{y})$ and $K_2(\boldsymbol{x},\boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x},\boldsymbol{y}) = K_1(\boldsymbol{x},\boldsymbol{y})\,K_2(\boldsymbol{x},\boldsymbol{y})$

- Writing $K_1(\boldsymbol{x},\boldsymbol{y}) = \sum_i \phi_1^{(i)}(\boldsymbol{x})\,\phi_1^{(i)}(\boldsymbol{y})$ and $K_2(\boldsymbol{x},\boldsymbol{y}) = \sum_j \phi_2^{(j)}(\boldsymbol{x})\,\phi_2^{(j)}(\boldsymbol{y})$ then

$$K_3(\boldsymbol{x},\boldsymbol{y}) = \sum_{i,j} \phi_1^{(i)}(\boldsymbol{x})\,\phi_1^{(i)}(\boldsymbol{y})\,\phi_2^{(j)}(\boldsymbol{x})\,\phi_2^{(j)}(\boldsymbol{y})$$

Product of Kernels

- If $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})\, K_2(\boldsymbol{x}, \boldsymbol{y})$

- Writing $K_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_1^{(i)}(\boldsymbol{x})\, \phi_1^{(i)}(\boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \phi_2^{(j)}(\boldsymbol{x})\, \phi_2^{(j)}(\boldsymbol{y})$ then

$$K_3(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i,j} \phi_1^{(i)}(\boldsymbol{x})\, \phi_1^{(i)}(\boldsymbol{y})\, \phi_2^{(j)}(\boldsymbol{x})\, \phi_2^{(j)}(\boldsymbol{y})$$

Product of Kernels

- If $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are valid kernels then so is $K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})\, K_2(\boldsymbol{x}, \boldsymbol{y})$

- Writing $K_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_1^{(i)}(\boldsymbol{x})\, \phi_1^{(i)}(\boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \phi_2^{(j)}(\boldsymbol{x})\, \phi_2^{(j)}(\boldsymbol{y})$ then

$$K_3(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i,j} \phi_1^{(i)}(\boldsymbol{x})\, \phi_1^{(i)}(\boldsymbol{y})\, \phi_2^{(j)}(\boldsymbol{x})\, \phi_2^{(j)}(\boldsymbol{y}) = \sum_{i,j} \phi_3^{(ij)}(\boldsymbol{x})\, \phi_3^{(ij)}(\boldsymbol{y})$$

where $\phi_3^{(ij)}(\boldsymbol{x}) = \phi_1^{(i)}(\boldsymbol{x})\, \phi_2^{(j)}(\boldsymbol{x})$

Exponentiating Kernels

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $K^n(\boldsymbol{x}, \boldsymbol{y})$ (by induction)
  - ⋆ Assume $K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ this satisfies base case
  - ⋆ If $K^{n-1}(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ then

$$K^n(\boldsymbol{x}, \boldsymbol{y}) = K^{n-1}(\boldsymbol{x}, \boldsymbol{y})\, K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$$

- and $\exp(K(\boldsymbol{x}, \boldsymbol{y}))$ is also a valid kernel since

$$\mathrm{e}^{K(\boldsymbol{x}, \boldsymbol{y})} = \sum_k \frac{1}{i!} K^i(\boldsymbol{x}, \boldsymbol{y}) = 1 + K(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2} K^2(\boldsymbol{x}, \boldsymbol{y}) + \cdots$$

but each term in the sum is a kernel

Exponentiating Kernels

- If $K(\boldsymbol{x},\boldsymbol{y})$ is a valid kernel so is $K^n(\boldsymbol{x},\boldsymbol{y})$ (by induction)
  - ⋆ Assume $K(\boldsymbol{x},\boldsymbol{y}) \succeq 0$ this satisfies base case
  - ⋆ If $K^{n-1}(\boldsymbol{x},\boldsymbol{y}) \succeq 0$ then

$$K^n(\boldsymbol{x},\boldsymbol{y}) = K^{n-1}(\boldsymbol{x},\boldsymbol{y})\, K(\boldsymbol{x},\boldsymbol{y}) \succeq 0$$

- and $\exp(K(\boldsymbol{x},\boldsymbol{y}))$ is also a valid kernel since

$$\mathrm{e}^{K(\boldsymbol{x},\boldsymbol{y})} = \sum_k \frac{1}{i!} K^i(\boldsymbol{x},\boldsymbol{y}) = 1 + K(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2} K^2(\boldsymbol{x},\boldsymbol{y}) + \cdots$$

but each term in the sum is a kernel

Exponentiating Kernels

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $K^n(\boldsymbol{x}, \boldsymbol{y})$ (by induction)
  - ⋆ Assume $K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ this satisfies base case
  - ⋆ If $K^{n-1}(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ then

$$K^n(\boldsymbol{x}, \boldsymbol{y}) = K^{n-1}(\boldsymbol{x}, \boldsymbol{y})\, K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$$

- and $\exp(K(\boldsymbol{x}, \boldsymbol{y}))$ is also a valid kernel since

$$\mathrm{e}^{K(\boldsymbol{x}, \boldsymbol{y})} = \sum_k \frac{1}{i!} K^i(\boldsymbol{x}, \boldsymbol{y}) = 1 + K(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2} K^2(\boldsymbol{x}, \boldsymbol{y}) + \cdots$$

but each term in the sum is a kernel

Exponentiating Kernels

- If $K(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel so is $K^n(\boldsymbol{x}, \boldsymbol{y})$ (by induction)
  - ⋆ Assume $K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ this satisfies base case
  - ⋆ If $K^{n-1}(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$ then

$$K^n(\boldsymbol{x}, \boldsymbol{y}) = K^{n-1}(\boldsymbol{x}, \boldsymbol{y})\, K(\boldsymbol{x}, \boldsymbol{y}) \succeq 0$$

- and $\exp(K(\boldsymbol{x}, \boldsymbol{y}))$ is also a valid kernel since

$$\mathrm{e}^{K(\boldsymbol{x}, \boldsymbol{y})} = \sum_k \frac{1}{i!} K^i(\boldsymbol{x}, \boldsymbol{y}) = 1 + K(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2} K^2(\boldsymbol{x}, \boldsymbol{y}) + \cdots$$

but each term in the sum is a kernel

Gaussian Kernel

- Now $\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$ where $\phi^{(k)}(\boldsymbol{x}) = x_k$
- For $\gamma > 0$ we have $2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y} \succeq 0$
- Thus $\exp(2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}) \succeq 0$
- Since $\exp(\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x})$ and $\exp(\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y})$ are positive numbers

$$
\begin{aligned}
\frac{\mathrm{e}^{2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}}}{\mathrm{e}^{\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x}}\,\mathrm{e}^{\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y}}} &= \mathrm{e}^{-\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x}+2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}-\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y}} \\
&= \mathrm{e}^{-\gamma\,\|\boldsymbol{x}-\boldsymbol{y}\|^2} \succeq 0
\end{aligned}
$$

Gaussian Kernel

- Now $\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$ where $\phi^{(k)}(\boldsymbol{x}) = x_k$
- For $\gamma > 0$ we have $2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y} \succeq 0$
- Thus $\exp(2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}) \succeq 0$
- Since $\exp(\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x})$ and $\exp(\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y})$ are positive numbers

$$\frac{\mathrm{e}^{2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}}}{\mathrm{e}^{\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x}}\,\mathrm{e}^{\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y}}} = \mathrm{e}^{-\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{x}+2\,\gamma\,\boldsymbol{x}^\mathsf{T}\boldsymbol{y}-\gamma\,\boldsymbol{y}^\mathsf{T}\boldsymbol{y}}$$
$$= \mathrm{e}^{-\gamma\,\|\boldsymbol{x}-\boldsymbol{y}\|^2} \succeq 0$$

Gaussian Kernel

- Now $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_k \phi^{(k)}(\boldsymbol{x})\,\phi^{(k)}(\boldsymbol{y})$ where $\phi^{(k)}(\boldsymbol{x}) = x_k$

- For $\gamma > 0$ we have $2\,\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y} \succeq 0$

- Thus $\exp(2\,\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}) \succeq 0$

- Since $\exp(\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x})$ and $\exp(\gamma\,\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y})$ are positive numbers

$$\frac{\mathrm{e}^{2\,\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}}}{\mathrm{e}^{\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}}\,\mathrm{e}^{\gamma\,\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}}} = \mathrm{e}^{-\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}+2\,\gamma\,\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}-\gamma\,\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y}}$$

$$= \mathrm{e}^{-\gamma\,\|\boldsymbol{x}-\boldsymbol{y}\|^2} \succeq 0$$

Other Kernels

- The success of SVMs has meant that researchers try to increase the area of application
- The condition that a SVM kernel must be positive semi-definite is quite restrictive
- There has been an industry of research finding smart kernels for solving complicated problems
- The key to finding new kernels is to use the properties of kernels to build more complicated kernels from simpler ones

Other Kernels

- The success of SVMs has meant that researchers try to increase the area of application
- The condition that a SVM kernel must be positive semi-definite is quite restrictive
- There has been an industry of research finding smart kernels for solving complicated problems
- The key to finding new kernels is to use the properties of kernels to build more complicated kernels from simpler ones

Other Kernels

- The success of SVMs has meant that researchers try to increase the area of application
- The condition that a SVM kernel must be positive semi-definite is quite restrictive
- There has been an industry of research finding smart kernels for solving complicated problems
- The key to finding new kernels is to use the properties of kernels to build more complicated kernels from simpler ones

Other Kernels

- The success of SVMs has meant that researchers try to increase the area of application
- The condition that a SVM kernel must be positive semi-definite is quite restrictive
- There has been an industry of research finding smart kernels for solving complicated problems
- The key to finding new kernels is to use the properties of kernels to build more complicated kernels from simpler ones

String Kernels

- One area where SVMs have become very important is in document classification
- This requires comparing strings
- There are a large number of kernels developed to do this

String Kernels

- One area where SVMs have become very important is in document classification
- This requires comparing strings
- There are a large number of kernels developed to do this

String Kernels

- One area where SVMs have become very important is in document classification
- This requires comparing strings
- There are a large number of kernels developed to do this

Spectrum Kernel

- A simple way to compare documents is to collect a histogram of all occurrences of substrings of length p
- This is known as a p-spectrum
- A p-spectrum kernel counts the number of common substrings

$$s = \texttt{statistics} \quad \mathcal{S}_3(s) = \{\texttt{sta}, \texttt{tat}, \texttt{ati}, \texttt{tis}, \texttt{ist}, \texttt{sti}, \texttt{tic}, \texttt{ics}\}$$
$$t = \texttt{computation} \quad \mathcal{S}_3(t) = \{\texttt{com}, \texttt{omp}, \texttt{mpu}, \texttt{put}, \texttt{uta}, \texttt{tat}, \texttt{ati}, \texttt{tio}, \texttt{ion}\}$$

- $K(s,t) = 2$ ("tat" and "ati")

Spectrum Kernel

- A simple way to compare documents is to collect a histogram of all occurrences of substrings of length p
- This is known as a p-spectrum
- A p-spectrum kernel counts the number of common substrings

$$s = \texttt{statistics} \quad \mathcal{S}_3(s) = \{\texttt{sta}, \texttt{tat}, \texttt{ati}, \texttt{tis}, \texttt{ist}, \texttt{sti}, \texttt{tic}, \texttt{ics}\}$$

$$t = \texttt{computation} \quad \mathcal{S}_3(t) = \{\texttt{com}, \texttt{omp}, \texttt{mpu}, \texttt{put}, \texttt{uta}, \texttt{tat}, \texttt{ati}, \texttt{tio}, \texttt{ion}\}$$

- $K(s,t) = 2$ ("tat" and "ati")

Spectrum Kernel

- A simple way to compare documents is to collect a histogram of all occurrences of substrings of length p
- This is known as a p-spectrum
- A p-spectrum kernel counts the number of common substrings

$$s = \texttt{statistics} \quad \mathcal{S}_3(s) = \{\texttt{sta}, \texttt{tat}, \texttt{ati}, \texttt{tis}, \texttt{ist}, \texttt{sti}, \texttt{tic}, \texttt{ics}\}$$

$$t = \texttt{computation} \quad \mathcal{S}_3(t) = \{\texttt{com}, \texttt{omp}, \texttt{mpu}, \texttt{put}, \texttt{uta}, \texttt{tat}, \texttt{ati}, \texttt{tio}, \texttt{ion}\}$$

- $K(s,t) = 2$ (“$\texttt{tat}$” and “$\texttt{ati}$”)

Spectrum Kernel

- A simple way to compare documents is to collect a histogram of all occurrences of substrings of length p
- This is known as a p-spectrum
- A p-spectrum kernel counts the number of common substrings

$$s = \texttt{statistics} \quad \mathcal{S}_3(s) = \{\mathtt{sta}, \mathtt{tat}, \mathtt{ati}, \mathtt{tis}, \mathtt{ist}, \mathtt{sti}, \mathtt{tic}, \mathtt{ics}\}$$

$$t = \texttt{computation} \quad \mathcal{S}_3(t) = \{\mathtt{com}, \mathtt{omp}, \mathtt{mpu}, \mathtt{put}, \mathtt{uta}, \mathtt{tat}, \mathtt{ati}, \mathtt{tio}, \mathtt{ion}\}$$

- $K(s,t) = 2$ ("$\texttt{tat}$" and "$\texttt{ati}$")

All Subsequences Kernel

- A more sophisticated kernel is to count all of the common subsequences that occur in two documents
- Naively this would take an exponential amount of time to compute
- Using clever dynamic-programming techniques this can be done relatively efficiently
- This can even be extended to include sub-sequence matches with possible gaps between words

All Subsequences Kernel

- A more sophisticated kernel is to count all of the common subsequences that occur in two documents
- Naively this would take an exponential amount of time to compute
- Using clever dynamic-programming techniques this can be done relatively efficiently
- This can even be extended to include sub-sequence matches with possible gaps between words

All Subsequences Kernel

- A more sophisticated kernel is to count all of the common subsequences that occur in two documents
- Naively this would take an exponential amount of time to compute
- Using clever dynamic-programming techniques this can be done relatively efficiently
- This can even be extended to include sub-sequence matches with possible gaps between words

All Subsequences Kernel

- A more sophisticated kernel is to count all of the common subsequences that occur in two documents
- Naively this would take an exponential amount of time to compute
- Using clever dynamic-programming techniques this can be done relatively efficiently
- This can even be extended to include sub-sequence matches with possible gaps between words

Other Kernel Applications

- String kernels for comparing subsequences are used in bioinformatics
- Kernels have been developed for comparing trees (e.g. for computer program evaluation, XML, etc.)
- Kernels have also been developed for comparing graphs (e.g. for comparing chemicals based on their molecular graph)

Other Kernel Applications

- String kernels for comparing subsequences are used in bioinformatics
- Kernels have been developed for comparing trees (e.g. for computer program evaluation, XML, etc.)
- Kernels have also been developed for comparing graphs (e.g. for comparing chemicals based on their molecular graph)

Other Kernel Applications

- String kernels for comparing subsequences are used in bioinformatics
- Kernels have been developed for comparing trees (e.g. for computer program evaluation, XML, etc.)
- Kernels have also been developed for comparing graphs (e.g. for comparing chemicals based on their molecular graph)

Fisher Kernels

- In an attempt to build kernels that capture more domain knowledge, kernels are constructed from other learning machines
- An example of this are "Fisher kernels" whose features come from an Hidden Markov Model (HMM) trained on the data
- These tend to have better discriminative power than the underlying model (HMM), and has a better feature set than a SVM using a generic kernel

Fisher Kernels

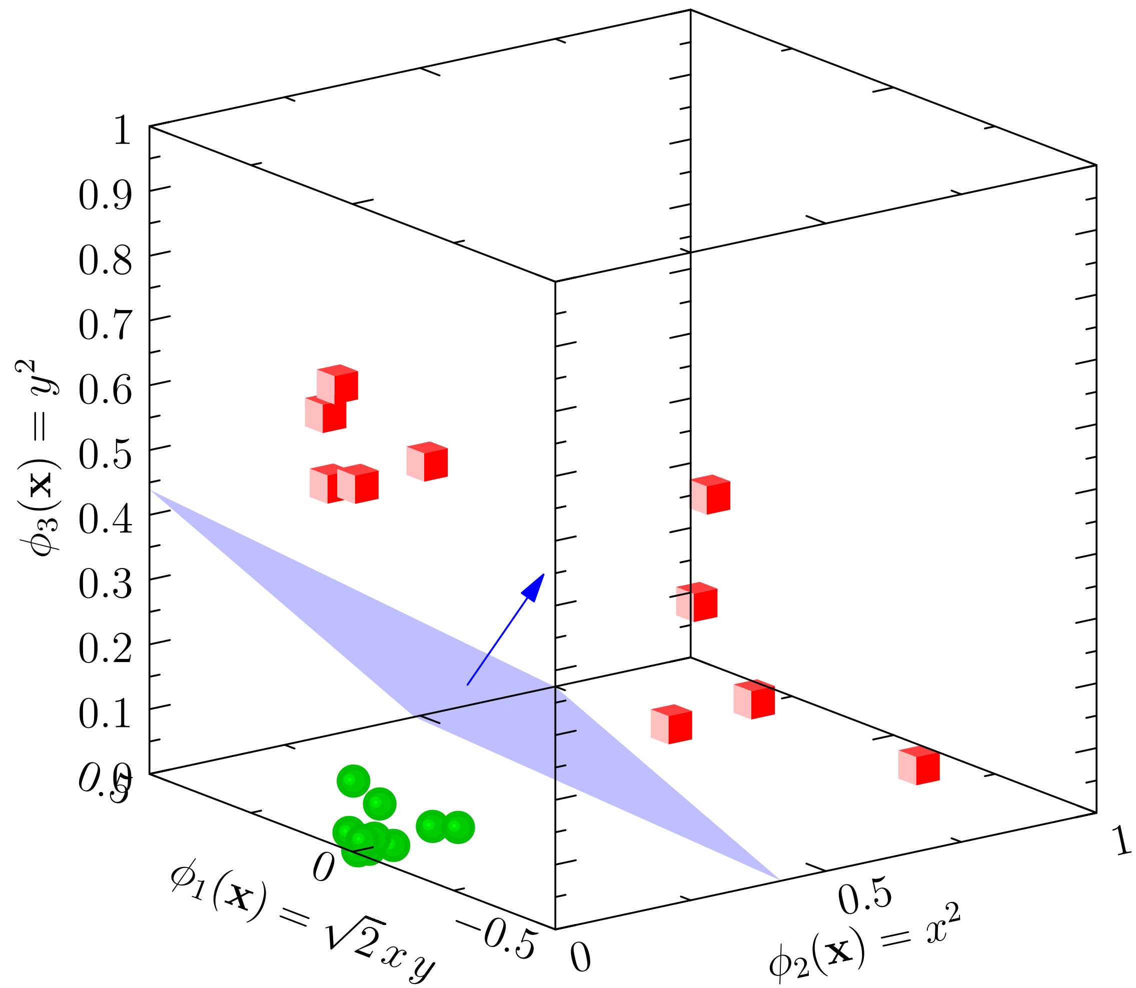
- In an attempt to build kernels that capture more domain knowledge, kernels are constructed from other learning machines
- An example of this are "Fisher kernels" whose features come from an Hidden Markov Model (HMM) trained on the data
- These tend to have better discriminative power than the underlying model (HMM), and has a better feature set than a SVM using a generic kernel

Fisher Kernels

- In an attempt to build kernels that capture more domain knowledge, kernels are constructed from other learning machines
- An example of this are "Fisher kernels" whose features come from an Hidden Markov Model (HMM) trained on the data
- These tend to have better discriminative power than the underlying model (HMM), and has a better feature set than a SVM using a generic kernel

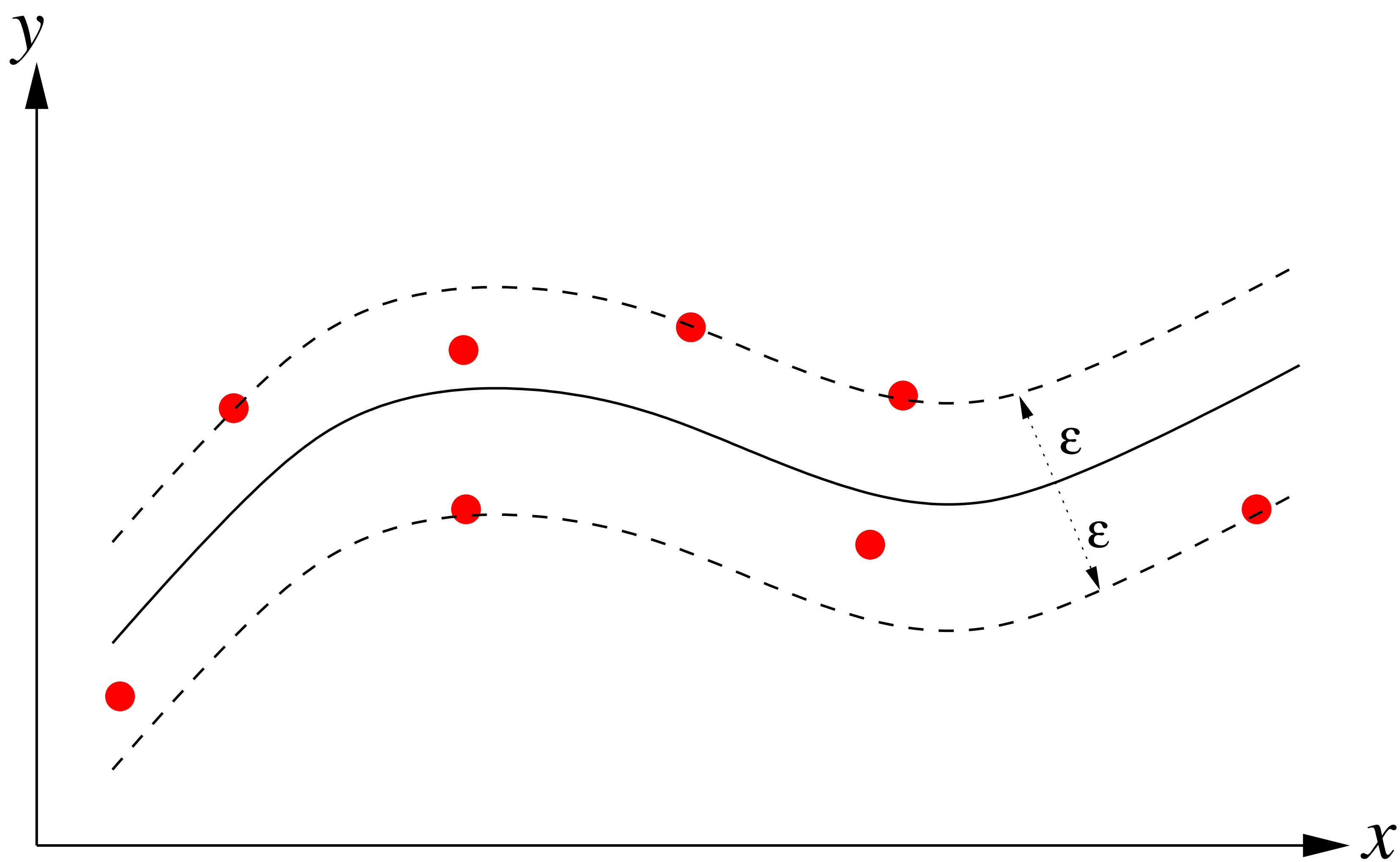
Outline

1. The Kernel Trick
2. Positive Semi-Definite Kernels
3. Kernel Properties
4. **Beyond Classification**



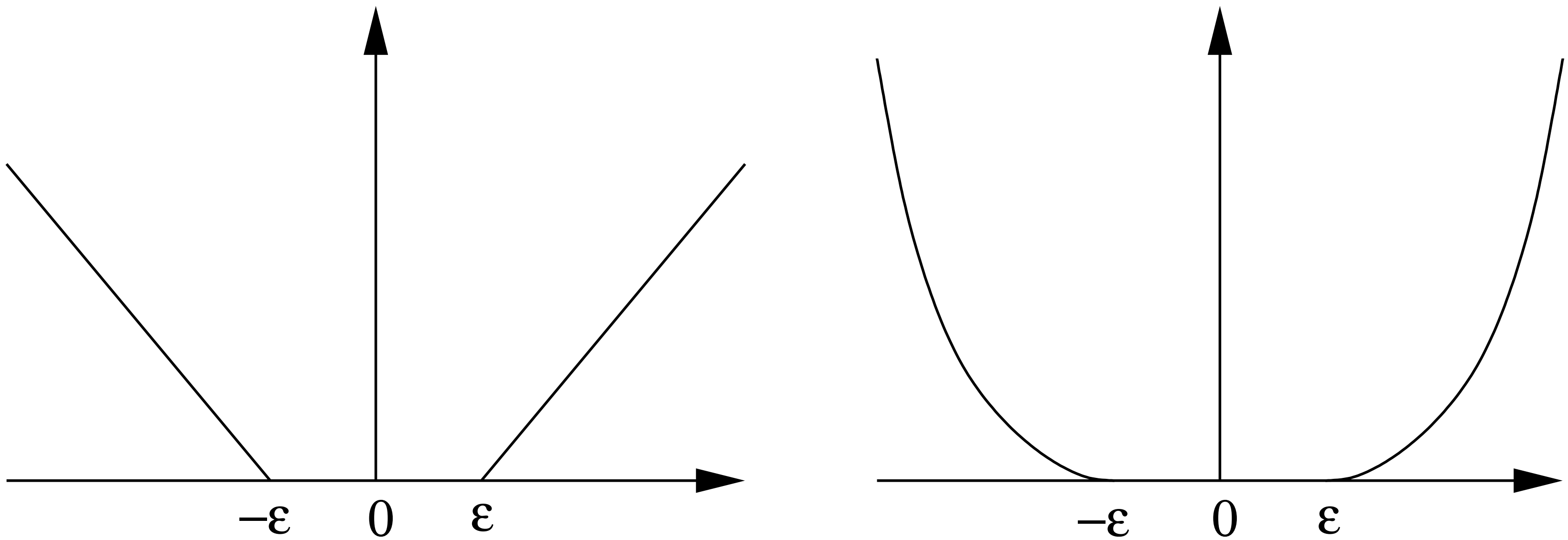
Regression with Margins

- SVMs can be modified to perform regression



Error Functions

- Can introduce slack variables with different errors



- This can be transformed to a quadratic programming problem

Ridge Regression Using Kernels

- We can also solve regression problems without using margins
- To solve a regression problem once again the problem is set up as a quadratic programming problem

$$\min_{\boldsymbol{w}} \lambda \, \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_i) \right)^2$$

- the $\|\boldsymbol{w}\|^2$ is a regularisation term
- By assuming $\boldsymbol{w} = \sum_i \alpha_i \, \boldsymbol{\phi}(\boldsymbol{x}_i)$ we obtain a quadratic equation for the α_i's which we can solve

Ridge Regression Using Kernels

- We can also solve regression problems without using margins
- To solve a regression problem once again the problem is set up as a quadratic programming problem

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2$$

- the $\|\boldsymbol{w}\|^2$ is a regularisation term
- By assuming $\boldsymbol{w} = \sum_i \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)$ we obtain a quadratic equation for the α_i's which we can solve

Ridge Regression Using Kernels

- We can also solve regression problems without using margins
- To solve a regression problem once again the problem is set up as a quadratic programming problem

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \big(y_i - \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\phi}(\boldsymbol{x}_i)\big)^2$$

- the $\|\boldsymbol{w}\|^2$ is a regularisation term
- By assuming $\boldsymbol{w} = \sum_i \alpha_i \, \boldsymbol{\phi}(\boldsymbol{x}_i)$ we obtain a quadratic equation for the α_i's which we can solve

Ridge Regression Using Kernels

- We can also solve regression problems without using margins
- To solve a regression problem once again the problem is set up as a quadratic programming problem

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}_i) \right)^2$$

- the $\|\boldsymbol{w}\|^2$ is a regularisation term
- By assuming $\boldsymbol{w} = \sum_i \alpha_i \, \boldsymbol{\phi}(\boldsymbol{x}_i)$ we obtain a quadratic equation for the α_i's which we can solve

Kernel Methods

- Kernel methods where we project into an extended feature space is also used with algorithms
  - ⋆ Kernel Fisher discriminant analysis (KFDA)
  - ⋆ Kernel principle component analysis (KPCA)
  - ⋆ Kernel canonical correlation analysis (KCCA)
  - ⋆ Gaussian Processes
- These are also extremely power machine learning algorithms

Kernel Methods

- Kernel methods where we project into an extended feature space is also used with algorithms
  - ⋆ Kernel Fisher discriminant analysis (KFDA)
  - ⋆ Kernel principle component analysis (KPCA)
  - ⋆ Kernel canonical correlation analysis (KCCA)
  - ⋆ Gaussian Processes
- These are also extremely power machine learning algorithms

Summary

- SVMs require a positive definite kernel function
- These can be built from simpler function
- There is an important industry of people creating new kernels for different application
- SVMs are just one example of a host of machine that
  - ⋆ use the kernel trick
  - ⋆ often use linear constraints
  - ⋆ tend to be convex optimisation problems

Summary

- SVMs require a positive definite kernel function
- These can be built from simpler function
- There is an important industry of people creating new kernels for different application
- SVMs are just one example of a host of machine that
  - ⋆ use the kernel trick
  - ⋆ often use linear constraints
  - ⋆ tend to be convex optimisation problems

Summary

- SVMs require a positive definite kernel function
- These can be built from simpler function
- There is an important industry of people creating new kernels for different application
- SVMs are just one example of a host of machine that
  - ⋆ use the kernel trick
  - ⋆ often use linear constraints
  - ⋆ tend to be convex optimisation problems

Summary

- SVMs require a positive definite kernel function
- These can be built from simpler function
- There is an important industry of people creating new kernels for different application
- SVMs are just one example of a host of machine that
  - ⋆ use the kernel trick
  - ⋆ often use linear constraints
  - ⋆ tend to be convex optimisation problems