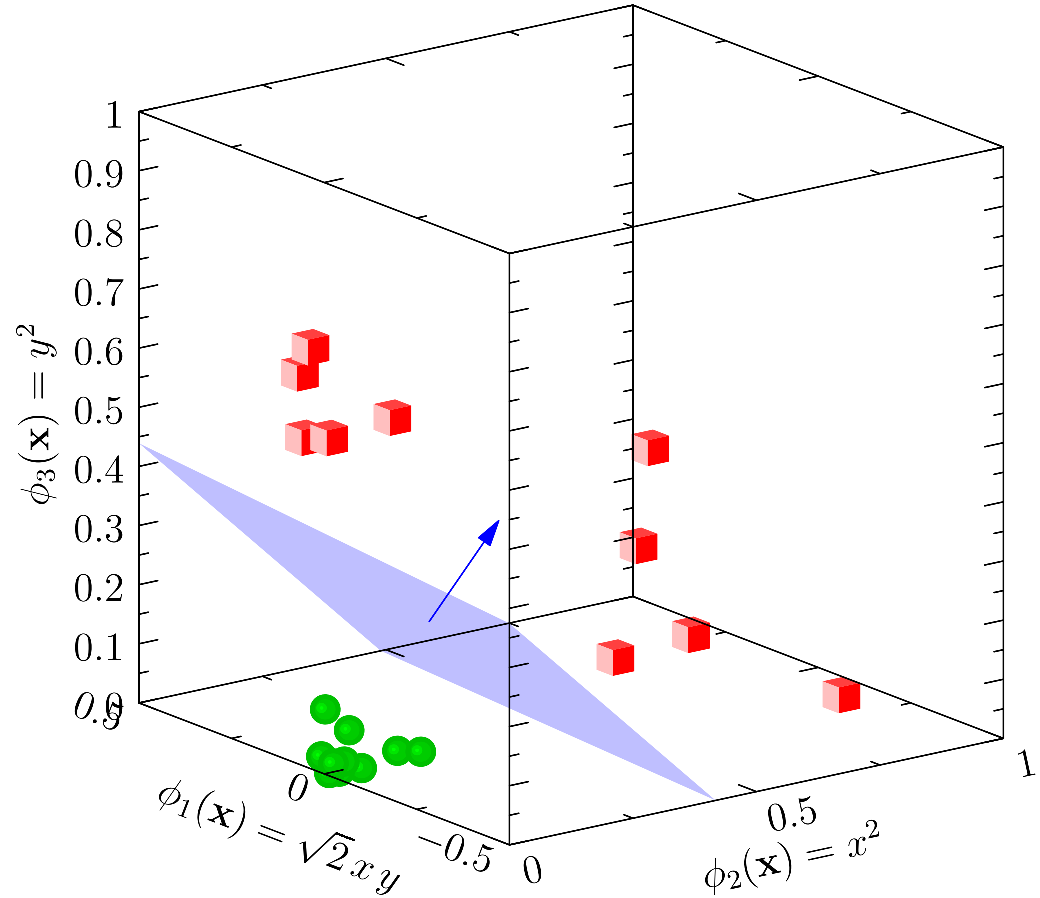
Advanced Machine Learning

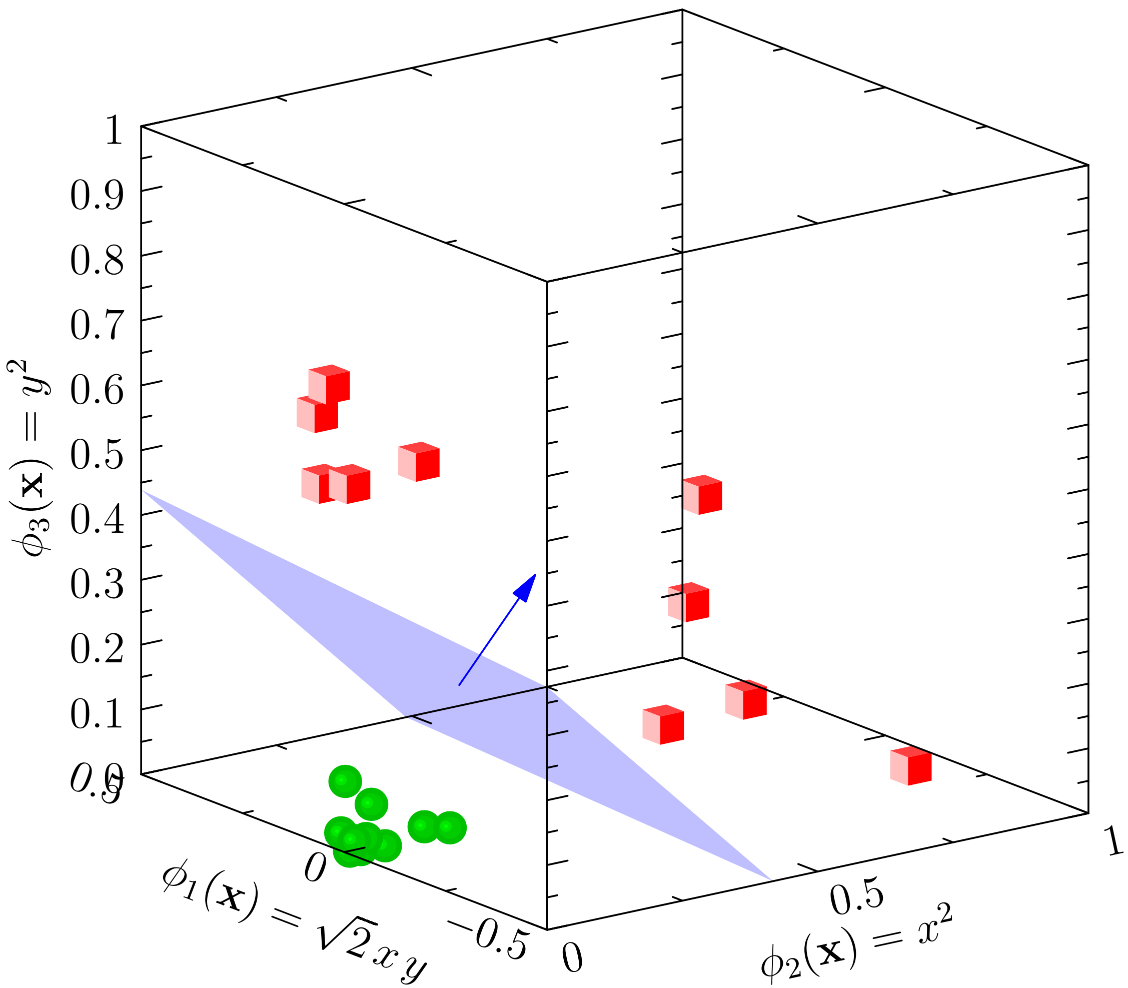
Kernel Properties



Kernel Properties, positive semi-definiteness, string kernels

Outline

1. **Recap**
2. Positive Semi-Definite Kernels
3. Training SVMs
4. Beyond Classification



SVMs

- A linear SVM finds the maximal margin hyperplane for separating linear separable data

- We can increase the chances of the data being linearly separable by projecting the data into an **extended feature space**

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \vdots \\ \vdots \\ \phi_m(\boldsymbol{x}) \end{pmatrix}$$

SVMs

- A linear SVM finds the maximal margin hyperplane for separating linear separable data

- We can increase the chances of the data being linearly separable by projecting the data into an **extended feature space**

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \phi_1(\boldsymbol{x}) \\ \phi_2(\boldsymbol{x}) \\ \vdots \\ \vdots \\ \phi_m(\boldsymbol{x}) \end{pmatrix}$$

Dual Form

- Finding the maximum margin hyperplane is equivalent to solving the quadratic programming problem

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

subject to $\alpha_k \geq 0$ and $\sum_k y_k \, \alpha_k = 0$

- This uses the data set $\{(\boldsymbol{x}_k, y_k) | k = 1, \ldots, P\}$ to learn a set of α_k's

- To classify new data we get a class prediction

$$\hat{y} = \operatorname{sgn}\left(\sum_{k \in SV} \alpha_k \, y_k \, \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_k) \, \boldsymbol{\phi}(\boldsymbol{x}) - b \right)$$

Dual Form

- Finding the maximum margin hyperplane is equivalent to solving the quadratic programming problem

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

subject to $\alpha_k \geq 0$ and $\sum_k y_k \, \alpha_k = 0$

- This uses the data set $\{(\boldsymbol{x}_k, y_k) | k = 1, \ldots, P\}$ to learn a set of α_k's

- To classify new data we get a class prediction

$$\hat{y} = \operatorname{sgn}\left(\sum_{k \in SV} \alpha_k \, y_k \, \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_k) \, \boldsymbol{\phi}(\boldsymbol{x}) - b \right)$$

Dual Form

- Finding the maximum margin hyperplane is equivalent to solving the quadratic programming problem

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

subject to $\alpha_k \geq 0$ and $\sum_k y_k \, \alpha_k = 0$

- This uses the data set $\{(\boldsymbol{x}_k, y_k) | k = 1, \ldots, P\}$ to learn a set of α_k's

- To classify new data we get a class prediction

$$\hat{y} = \operatorname{sgn}\left(\sum_{k \in SV} \alpha_k \, y_k \, \boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{x}_k) \, \boldsymbol{\phi}(\boldsymbol{x}) - b \right)$$

Kernel Trick

- If we define the kernel function as

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

then

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

$$\hat{y} = \operatorname{sgn}\left(\sum_{k\in SV} \alpha_k\,y_k\,K(\boldsymbol{x}_k,\boldsymbol{x}) - b\right)$$

- We only need to compute the kernel rather than $\phi(\boldsymbol{x})$

Kernel Trick

- If we define the kernel function as

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

then

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

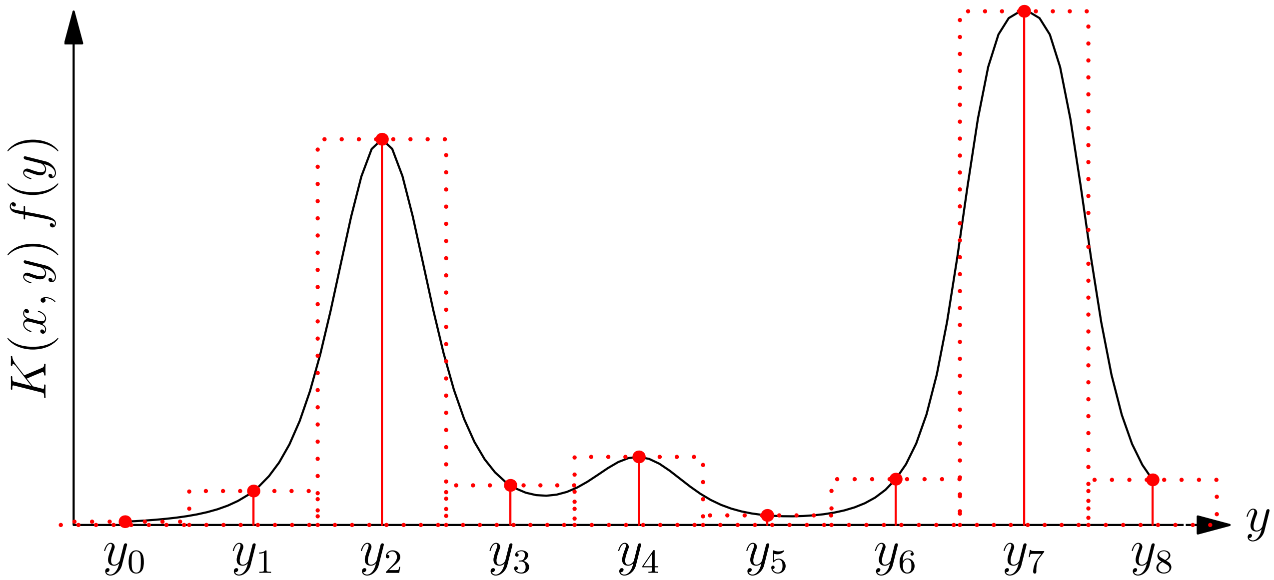
$$\hat{y} = \operatorname{sgn}\left(\sum_{k \in SV} \alpha_k\, y_k\, K(\boldsymbol{x}_k, \boldsymbol{x}) - b \right)$$

- We only need to compute the kernel rather than $\boldsymbol{\phi}(\boldsymbol{x})$

Kernels and Matrices

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

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

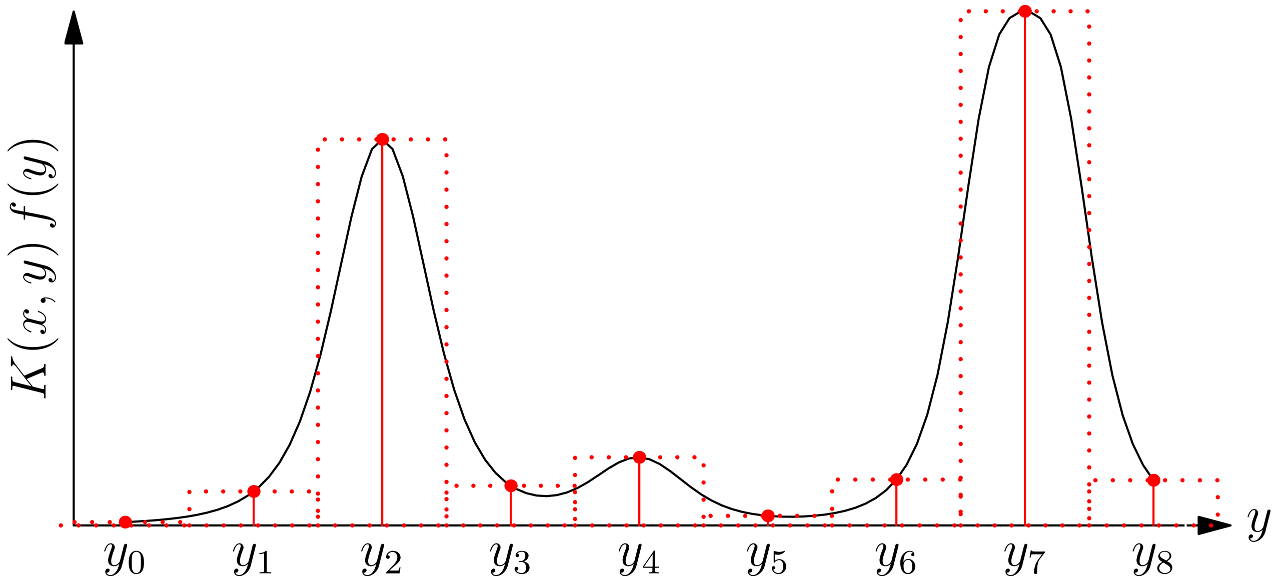


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Kernels and Matrices

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

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

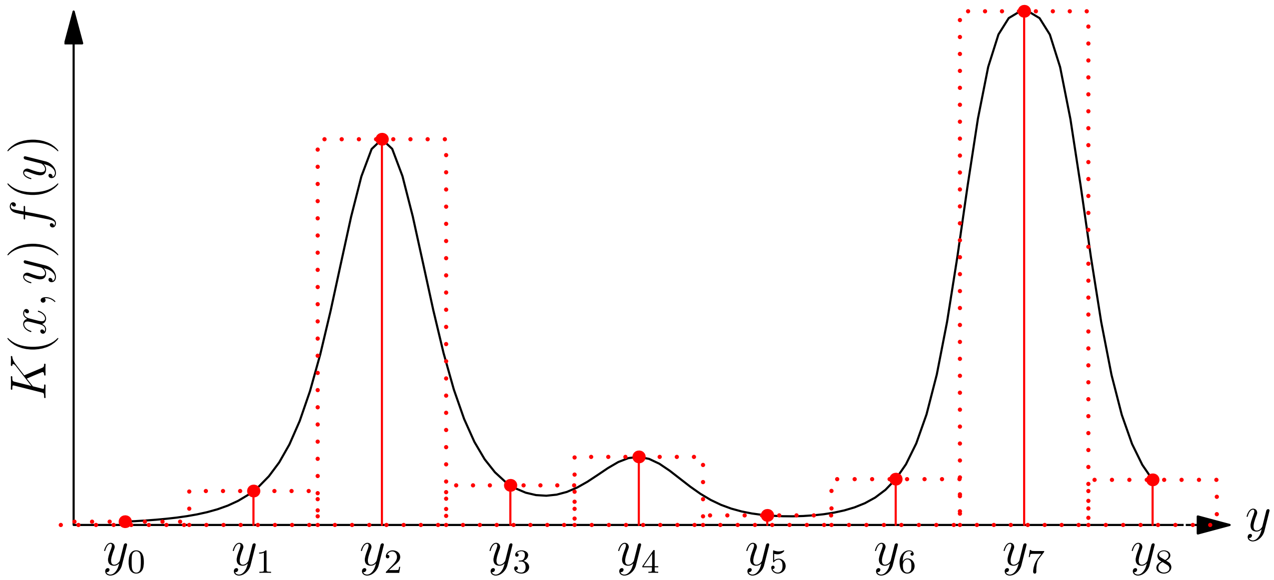


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Kernels and Matrices

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

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

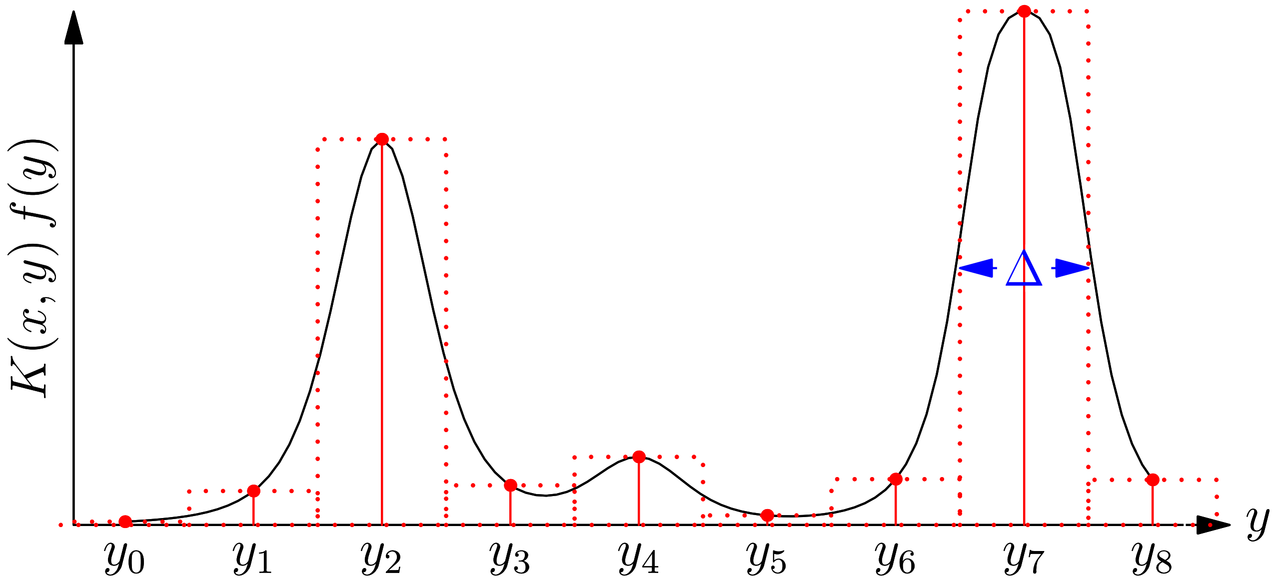


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Kernels and Matrices

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

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

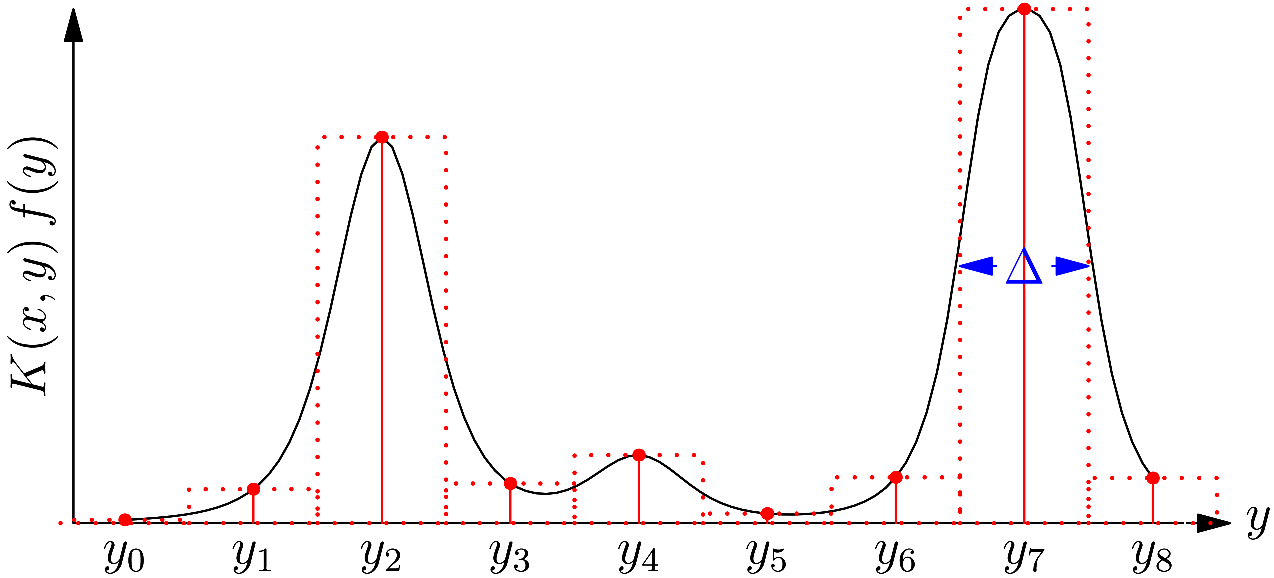


This is just a matrix equation with $M_{ij} = \Delta\, K(x_i, y_j)$

Kernels and Matrices

- A linear transformation $\mathcal{T}[f(x)]$ 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 $M_{ij} = \Delta\, K(x_i, y_j)$

Eigen-Functions

- In analogy to eigen-vectors we can define eigen-functions of a function of two variables

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \psi_k(\boldsymbol{x})$$

- The spatial coordinate $\boldsymbol{x}$ plays the same role as the index i

- We can decompose a kernel into a sum of its eigen-functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

Eigen-Functions

- In analogy to eigen-vectors we can define eigen-functions of a function of two variables

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \psi_k(\boldsymbol{x})$$

$$\sum_j M_{ij}\, v_j^{(k)} = \lambda_k v_i^{(k)}$$

- The spatial coordinate $\boldsymbol{x}$ plays the same role as the index i

- We can decompose a kernel into a sum of its eigen-functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

Eigen-Functions

- In analogy to eigen-vectors we can define eigen-functions of a function of two variables

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \psi_k(\boldsymbol{x})$$

$$\sum_j M_{ij}\, v_j^{(k)} = \lambda_k v_i^{(k)}$$

- The spatial coordinate $\boldsymbol{x}$ plays the same role as the index i

- We can decompose a kernel into a sum of its eigen-functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

Eigen-Functions

- In analogy to eigen-vectors we can define eigen-functions of a function of two variables

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \psi_k(\boldsymbol{x})$$

$$\sum_j M_{ij}\, v_j^{(k)} = \lambda_k v_i^{(k)}$$

- The spatial coordinate $\boldsymbol{x}$ plays the same role as the index i

- We can decompose a kernel into a sum of its eigen-functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

Eigen-Functions

- In analogy to eigen-vectors we can define eigen-functions of a function of two variables

$$\int K(\boldsymbol{x}, \boldsymbol{y})\, \psi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \psi_k(\boldsymbol{x})$$

$$\sum_j M_{ij}\, v_j^{(k)} = \lambda_k v_i^{(k)}$$

- The spatial coordinate $\boldsymbol{x}$ plays the same role as the index i

- We can decompose a kernel into a sum of its eigen-functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y}) \quad \text{c.f.} \quad \mathbf{M} = \sum_{i=1} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^{\mathsf{T}}$$

Mercer's Theorem

- Mercer tells us that for any symmetric kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

- If $\lambda_i \geq 0$ for all i then we can define $\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i}\psi(\boldsymbol{x})$

- And

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

Mercer's Theorem

- Mercer tells us that for any symmetric kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

- If $\lambda_i \geq 0$ for all i then we can define $\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i}\psi(\boldsymbol{x})$

- And

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

Mercer's Theorem

- Mercer tells us that for any symmetric kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

- If $\lambda_i \geq 0$ for all i then we can define $\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i} \psi(\boldsymbol{x})$

- And

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

Mercer's Theorem

- Mercer tells us that for any symmetric kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \lambda_i \psi_i(\boldsymbol{x})\, \psi_i(\boldsymbol{y})$$

- If $\lambda_i \geq 0$ for all i then we can define $\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i}\psi(\boldsymbol{x})$

- And

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

- That is, any positive semi-definite symmetric function of two variables is a valid kernel function!

General Kernels

- If we used any old kernel function with some negative eigenvalues then there can be a projection $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$ such that

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}) \rangle < 0$$

  (e.g. if $\boldsymbol{\phi}(\boldsymbol{x})$ was an eigenvector with negative eigenvalue)

- We are no longer in a space with Euclidean geometry
- Maximum margins are meaningless
- Must use positive semi-definite kernels

General Kernels

- If we used any old kernel function with some negative eigenvalues then there can be a projection $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$ such that

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}) \rangle < 0$$

  (e.g. if $\boldsymbol{\phi}(\boldsymbol{x})$ was an eigenvector with negative eigenvalue)

- We are no longer in a space with Euclidean geometry
- Maximum margins are meaningless
- Must use positive semi-definite kernels

General Kernels

- If we used any old kernel function with some negative eigenvalues then there can be a projection $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$ such that

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}) \rangle < 0$$

  (e.g. if $\boldsymbol{\phi}(\boldsymbol{x})$ was an eigenvector with negative eigenvalue)

- We are no longer in a space with Euclidean geometry

- Maximum margins are meaningless

- Must use positive semi-definite kernels

General Kernels

- If we used any old kernel function with some negative eigenvalues then there can be a projection $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$ such that

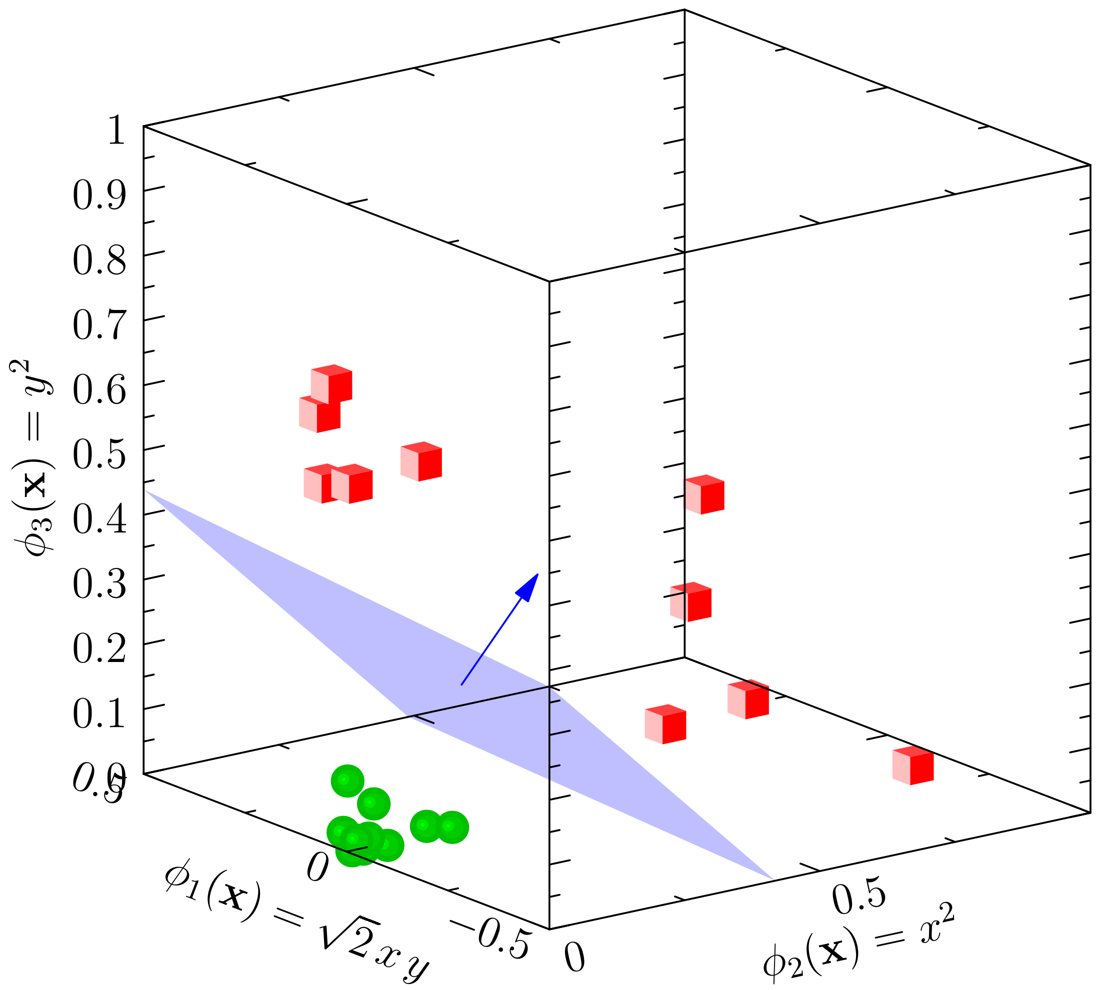
$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}) \rangle < 0$$

  (e.g. if $\boldsymbol{\phi}(\boldsymbol{x})$ was an eigenvector with negative eigenvalue)

- We are no longer in a space with Euclidean geometry
- Maximum margins are meaningless
- Must use positive semi-definite kernels

Outline

1. Recap
2. **Positive Semi-Definite Kernels**
3. Training SVMs
4. Beyond Classification



Positive Semi-Definite Kernels

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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

Positive Semi-Definite Kernels

- Kernels (or matrices) that have eigenvalues $\lambda_i \geq 0$ are called positive semi-definite

- (If the eigenvalues are strictly positive $\lambda_i > 0$ the kernels or matrices are called positive definite)

- Positive semi-definite kernels can always be decomposed into a sum of real functions

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

Positive Semi-Definite Kernels

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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

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

$$\begin{aligned}\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} &= \int f(\boldsymbol{x}) \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} \\ &= \sum_i \left(\int f(\boldsymbol{x})\, \phi_i(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x} \right)^2 \geq 0\end{aligned}$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$$

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

$$\begin{aligned}\int f(\boldsymbol{x})\,K(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\,\boldsymbol{x}\,\mathrm{d}\,\boldsymbol{y} &= \int f(\boldsymbol{x}) \sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})\,f(\boldsymbol{y})\,\mathrm{d}\,\boldsymbol{x}\,\mathrm{d}\,\boldsymbol{y} \\ &= \sum_i \left(\int f(\boldsymbol{x})\,\phi_i(\boldsymbol{x})\,\mathrm{d}\,\boldsymbol{x} \right)^2 \geq 0\end{aligned}$$

Properties of Positive Semi-Definiteness

- Since

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

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

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y} = \int f(\boldsymbol{x}) \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\,\boldsymbol{x}\, \mathrm{d}\,\boldsymbol{y}$$

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

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

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$$

    where $\phi(\boldsymbol{x})$ are real functions
  - ⋆ For any real function $f(x)$

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

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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

    where $\phi(\boldsymbol{x})$ 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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

    where $\phi(\boldsymbol{x})$ 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

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\, \phi_i(\boldsymbol{y})$$

    where $\phi(\boldsymbol{x})$ 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 $K_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y})$ then

$$
\begin{aligned}
K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})\, K_2(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i,j} \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y}) \\
&= \sum_{i,j} \left(\phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})\right) \left(\phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{y})\right) \\
&= \sum_{i,j} \phi_{ij}^3(\boldsymbol{x})\, \phi_{ij}^3(\boldsymbol{y})
\end{aligned}
$$

where $\phi_{ij}^3(\boldsymbol{x}) = \phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})$

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 $K_1(\boldsymbol{x},\boldsymbol{y}) = \sum_i \phi_i^1(\boldsymbol{x})\,\phi_i^1(\boldsymbol{y})$ and $K_2(\boldsymbol{x},\boldsymbol{y}) = \sum_j \phi_j^2(\boldsymbol{x})\,\phi_j^2(\boldsymbol{y})$ then

$$K_3(\boldsymbol{x},\boldsymbol{y}) = K_1(\boldsymbol{x},\boldsymbol{y})\, K_2(\boldsymbol{x},\boldsymbol{y}) = \sum_{i,j} \phi_i^1(\boldsymbol{x})\,\phi_i^1(\boldsymbol{y})\,\phi_j^2(\boldsymbol{x})\,\phi_j^2(\boldsymbol{y})$$

$$= \sum_{i,j} \left(\phi_i^1(\boldsymbol{x})\,\phi_j^2(\boldsymbol{x})\right)\,\left(\phi_i^1(\boldsymbol{y})\,\phi_j^2(\boldsymbol{y})\right)$$

$$= \sum_{i,j} \phi_{ij}^3(\boldsymbol{x})\,\phi_{ij}^3(\boldsymbol{y})$$

where $\phi_{ij}^3(\boldsymbol{x}) = \phi_i^1(\boldsymbol{x})\,\phi_j^2(\boldsymbol{x})$

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 $K_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y})$ then

$$
\begin{aligned}
K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})\, K_2(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i,j} \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y}) \\
&= \sum_{i,j} \left(\phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})\right) \left(\phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{y})\right) \\
&= \sum_{i,j} \phi_{ij}^3(\boldsymbol{x})\, \phi_{ij}^3(\boldsymbol{y})
\end{aligned}
$$

where $\phi_{ij}^3(\boldsymbol{x}) = \phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})$

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 $K_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_i \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y})$ then

$$
\begin{aligned}
K_3(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})\, K_2(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i,j} \phi_i^1(\boldsymbol{x})\, \phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{x})\, \phi_j^2(\boldsymbol{y}) \\
&= \sum_{i,j} \left(\phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})\right) \left(\phi_i^1(\boldsymbol{y})\, \phi_j^2(\boldsymbol{y})\right) \\
&= \sum_{i,j} \phi_{ij}^3(\boldsymbol{x})\, \phi_{ij}^3(\boldsymbol{y})
\end{aligned}
$$

where $\phi_{ij}^3(\boldsymbol{x}) = \phi_i^1(\boldsymbol{x})\, \phi_j^2(\boldsymbol{x})$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $\sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$ where $\phi_i(\boldsymbol{x}) = x_i$
- 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

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

- This is the RBF or Gaussian kernel

Gaussian Kernel

- Now $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$ where $\phi_i(\boldsymbol{x}) = x_i$
- 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

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

- This is the RBF or Gaussian kernel

Gaussian Kernel

- Now $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$ where $\phi_i(\boldsymbol{x}) = x_i$
- 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

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

- This is the RBF or Gaussian kernel

Gaussian Kernel

- Now $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$ where $\phi_i(\boldsymbol{x}) = x_i$
- 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

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

- This is the RBF or Gaussian kernel

Gaussian Kernel

- Now $\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$ is a valid kernel because it is of the form $\sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$ where $\phi_i(\boldsymbol{x}) = x_i$
- 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

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

- This is the RBF or Gaussian kernel

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

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

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

- $K(s,t) = 2$ (“$\texttt{tat}$” 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

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

- $K(s,t) = 2$ ("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

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

- $K(s,t) = 2$ (“$\texttt{tat}$” 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 a huge 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 a huge 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 a huge 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 a huge 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
- "Fisher kernels" can be constructed from features coming from generative models (e.g. a Hidden Markov Model (HMM) trained on biological 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

- "Fisher kernels" can be constructed from features coming from generative models (e.g. a Hidden Markov Model (HMM) trained on biological data)

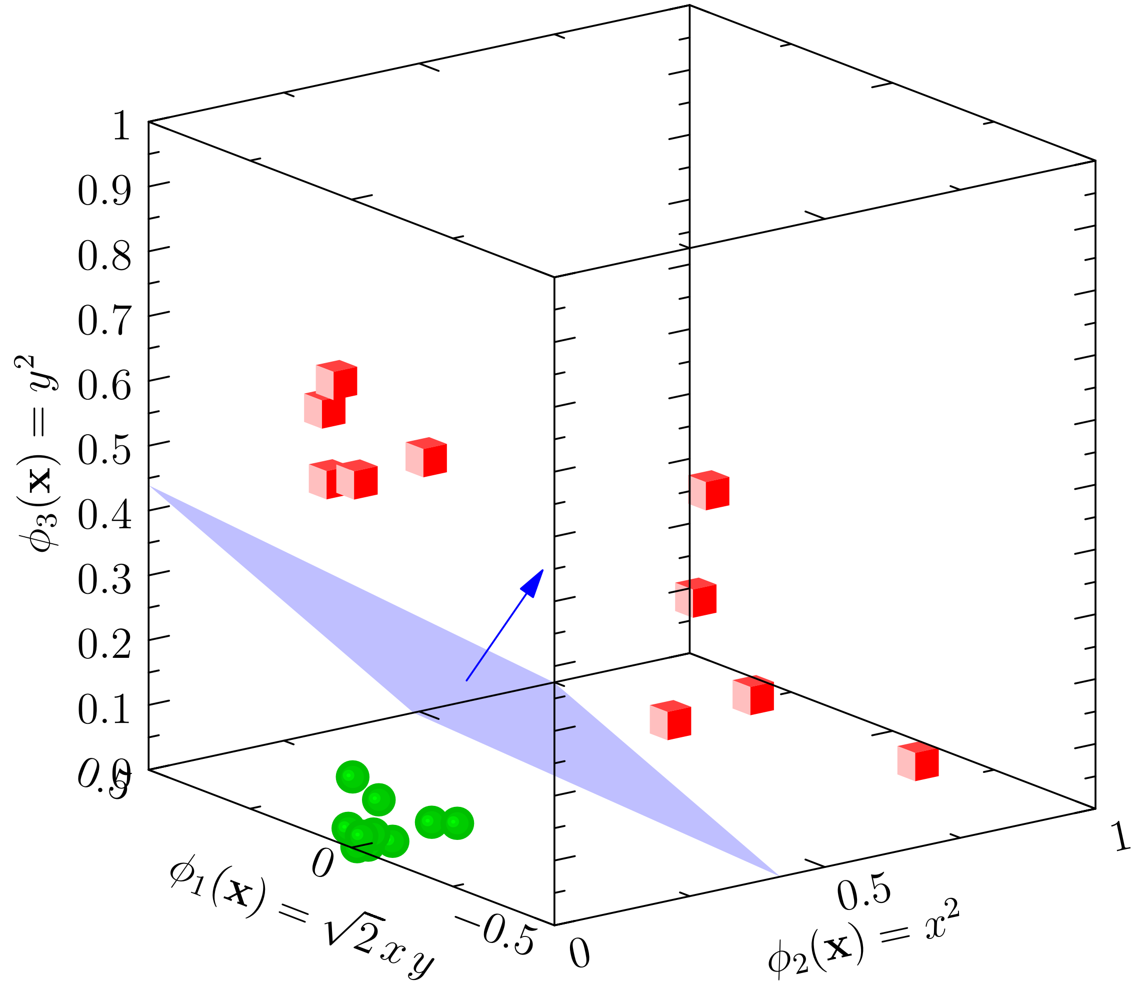
- These tend to have better discriminative power than the underlying model (HMM), and has a better feature set than a SVM using a generic kernel

Fisher Kernels

- In an attempt to build kernels that capture more domain knowledge, kernels are constructed from other learning machines
- "Fisher kernels" can be constructed from features coming from generative models (e.g. a Hidden Markov Model (HMM) trained on biological 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. Recap
2. Positive Semi-Definite Kernels
3. **Training SVMs**
4. Beyond Classification



Quadratic Optimisation

- The dual problem is

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

  subject to $\alpha_k \geq 0$ and $\sum_k y_k \, \alpha_k = 0$

- If we allow slack variables with a constraint $C \sum_k \xi_k$ then get the same problem with

$$0 \leq \alpha_k \leq C \qquad \forall k = 1, 2 \ldots, P$$

Quadratic Optimisation

- The dual problem is

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

subject to $\alpha_k \geq 0$ and $\sum_k y_k \, \alpha_k = 0$

- If we allow slack variables with a constraint $C \sum_k \xi_k$ then get the same problem with

$$0 \leq \alpha_k \leq C \qquad \forall k = 1, 2 \ldots, P$$

General QP Solver

- Traditional quadratic programming solvers start from a feasible solutions (usually found using linear programming)
- Takes the current set of constraints that are exactly satisfied as the active set
- Optimises with respect to the active set taken as equality constraints
- Moves towards the new optimum as far as possible (so that none of the non-active constraints is broken)
- If any of the Lagrange multipliers are negative, it drops the constraints

General QP Solver

- Traditional quadratic programming solvers start from a feasible solutions (usually found using linear programming)
- Takes the current set of constraints that are exactly satisfied as the active set
- Optimises with respect to the active set taken as equality constraints
- Moves towards the new optimum as far as possible (so that none of the non-active constraints is broken)
- If any of the Lagrange multipliers are negative, it drops the constraints

General QP Solver

- Traditional quadratic programming solvers start from a feasible solutions (usually found using linear programming)
- Takes the current set of constraints that are exactly satisfied as the active set
- Optimises with respect to the active set taken as equality constraints
- Moves towards the new optimum as far as possible (so that none of the non-active constraints is broken)
- If any of the Lagrange multipliers are negative, it drops the constraints

General QP Solver

- Traditional quadratic programming solvers start from a feasible solutions (usually found using linear programming)
- Takes the current set of constraints that are exactly satisfied as the active set
- Optimises with respect to the active set taken as equality constraints
- Moves towards the new optimum as far as possible (so that none of the non-active constraints is broken)
- If any of the Lagrange multipliers are negative, it drops the constraints

General QP Solver

- Traditional quadratic programming solvers start from a feasible solutions (usually found using linear programming)
- Takes the current set of constraints that are exactly satisfied as the active set
- Optimises with respect to the active set taken as equality constraints
- Moves towards the new optimum as far as possible (so that none of the non-active constraints is broken)
- If any of the Lagrange multipliers are negative, it drops the constraints

Time Complexity of SVMs

- SVMs have good generalisation performance often beating MLP or RBFs
- They have a unique classification boundary unlike MLPs which can find local optima
- The time complexity for training is $O(P^3)$ where P is the number of training patterns
- It can be too slow if $P \gg 1000$

Time Complexity of SVMs

- SVMs have good generalisation performance often beating MLP or RBFs
- They have a unique classification boundary unlike MLPs which can find local optima
- The time complexity for training is $O(P^3)$ where P is the number of training patterns
- It can be too slow if $P \gg 1000$

Time Complexity of SVMs

- SVMs have good generalisation performance often beating MLP or RBFs
- They have a unique classification boundary unlike MLPs which can find local optima
- The time complexity for training is $O(P^3)$ where P is the number of training patterns
- It can be too slow if $P \gg 1000$

Time Complexity of SVMs

- SVMs have good generalisation performance often beating MLP or RBFs
- They have a unique classification boundary unlike MLPs which can find local optima
- The time complexity for training is $O(P^3)$ where P is the number of training patterns
- It can be too slow if $P \gg 1000$

Chunking

- Chunking is an attempt to find an approximation for the maximum margin hyperplane by working on "chunks" of the dataset
- The algorithm considers an *chunk* of data at a time
- Initially we run an SVM on the first chunk of data
- The support vectors for the chunk are retained while the rest of the data in the chunk is discarded
- The support vectors together with the next set of data is considered

Chunking

- Chunking is an attempt to find an approximation for the maximum margin hyperplane by working on “chunks” of the dataset
- The algorithm considers an *chunk* of data at a time
- Initially we run an SVM on the first chunk of data
- The support vectors for the chunk are retained while the rest of the data in the chunk is discarded
- The support vectors together with the next set of data is considered

Chunking

- Chunking is an attempt to find an approximation for the maximum margin hyperplane by working on "chunks" of the dataset
- The algorithm considers an $chunk$ of data at a time
- Initially we run an SVM on the first chunk of data
- The support vectors for the chunk are retained while the rest of the data in the chunk is discarded
- The support vectors together with the next set of data is considered

Chunking

- Chunking is an attempt to find an approximation for the maximum margin hyperplane by working on "chunks" of the dataset
- The algorithm considers an $chunk$ of data at a time
- Initially we run an SVM on the first chunk of data
- The support vectors for the chunk are retained while the rest of the data in the chunk is discarded
- The support vectors together with the next set of data is considered

Chunking

- Chunking is an attempt to find an approximation for the maximum margin hyperplane by working on "chunks" of the dataset
- The algorithm considers an $chunk$ of data at a time
- Initially we run an SVM on the first chunk of data
- The support vectors for the chunk are retained while the rest of the data in the chunk is discarded
- The support vectors together with the next set of data is considered

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Sequential Minimal Optimisation

- One of the most efficient techniques for training SVMs is *Sequential Minimal Optimisation* or SMO
- This takes two Lagrange multipliers α_i and α_j and adjusts them to maximise the dual objective function
- This is very quick as it can be done in closed form
- Note that because $\sum_k y_k \alpha_k = 0$ we have to change at least two variables at the same time
- A heuristic is used to choose the best pair of α's to optimise
- Run until close to the optimum

Multi-class Classification

- SVMs are by nature a binary classifier
- There are a number of strategies to make them multi-class, two frequent strategies
  - ⋆ Train $|\mathcal{C}|$ one-versus-all classifies and choose best
  - ⋆ Train $|\mathcal{C}|(|\mathcal{C}|-1)/2$ one-versus-one classifies and vote for best
- More elegant, but slightly more complicated alternatives exist involving using the class label as a feature

Multi-class Classification

- SVMs are by nature a binary classifier
- There are a number of strategies to make them multi-class, two frequent strategies
  - ⋆ Train $|\mathcal{C}|$ one-versus-all classifies and choose best
  - ⋆ Train $|\mathcal{C}|(|\mathcal{C}|-1)/2$ one-versus-one classifies and vote for best
- More elegant, but slightly more complicated alternatives exist involving using the class label as a feature

Multi-class Classification

- SVMs are by nature a binary classifier
- There are a number of strategies to make them multi-class, two frequent strategies
  - ⋆ Train $|\mathcal{C}|$ one-versus-all classifies and choose best
  - ⋆ Train $|\mathcal{C}|(|\mathcal{C}|-1)/2$ one-versus-one classifies and vote for best
- More elegant, but slightly more complicated alternatives exist involving using the class label as a feature

Multi-class Classification

- SVMs are by nature a binary classifier
- There are a number of strategies to make them multi-class, two frequent strategies
  - ⋆ Train $|\mathcal{C}|$ one-versus-all classifies and choose best
  - ⋆ Train $|\mathcal{C}|(|\mathcal{C}|-1)/2$ one-versus-one classifies and vote for best
- More elegant, but slightly more complicated alternatives exist involving using the class label as a feature

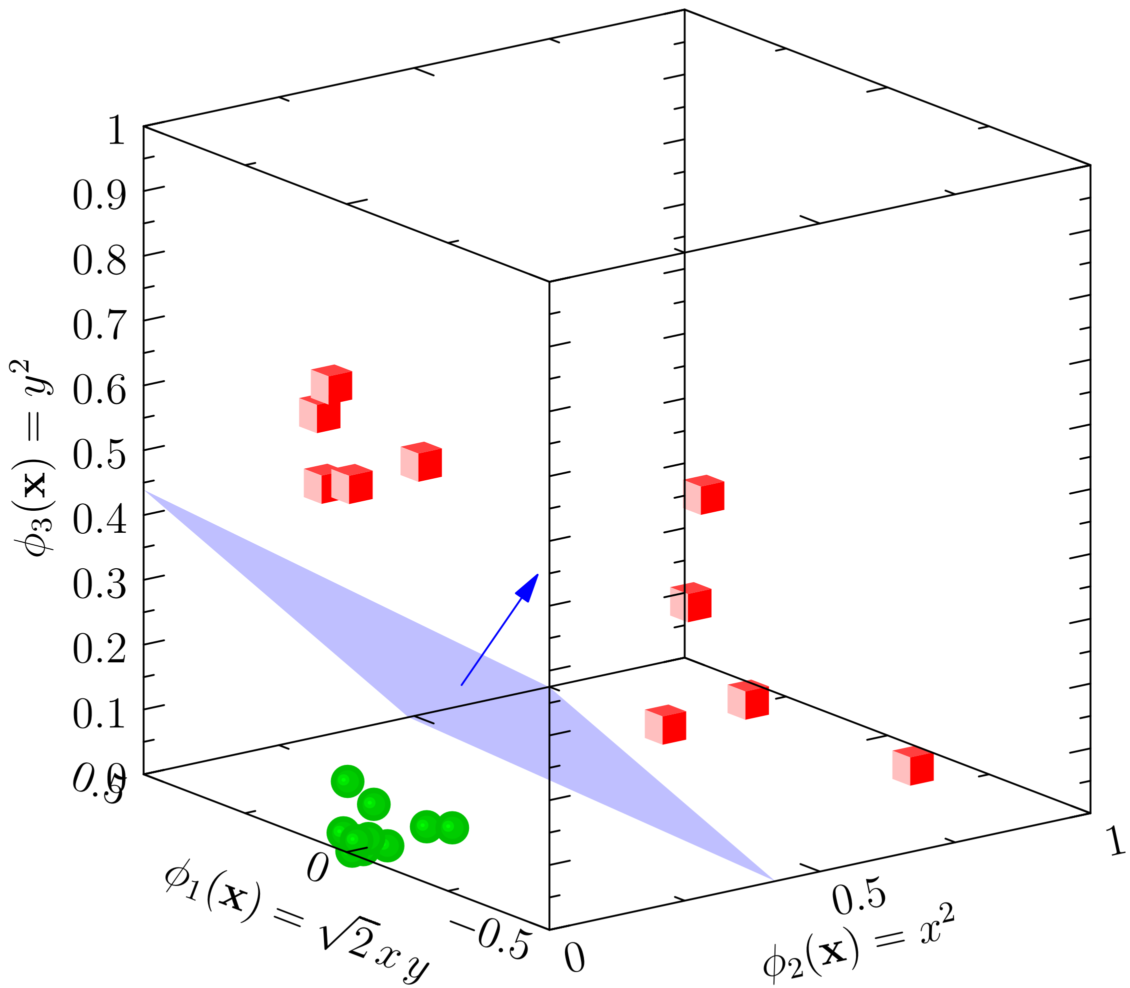
Outline

1. Recap

2. Positive Semi-Definite Kernels

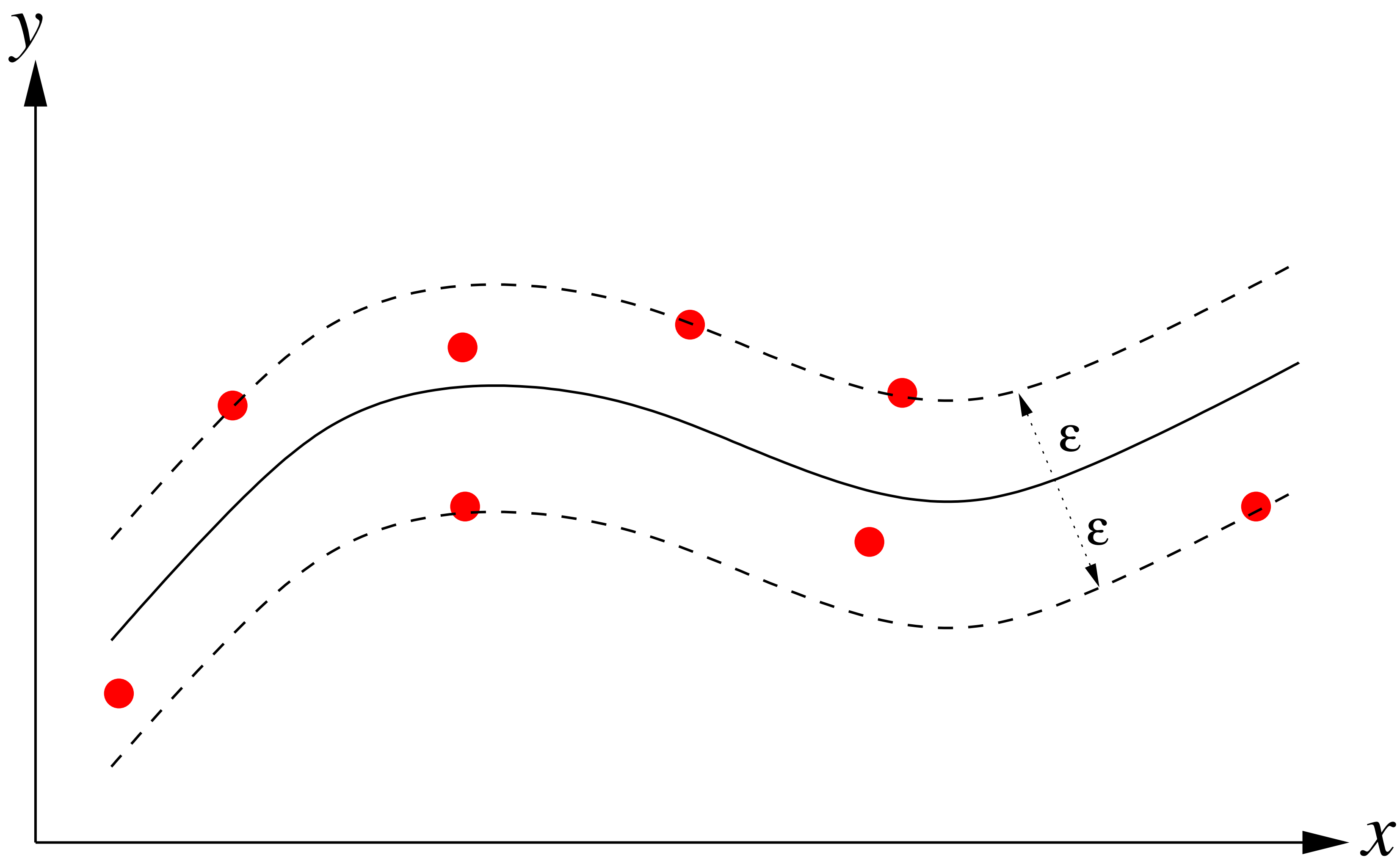
3. Training SVMs

4. **Beyond Classification**



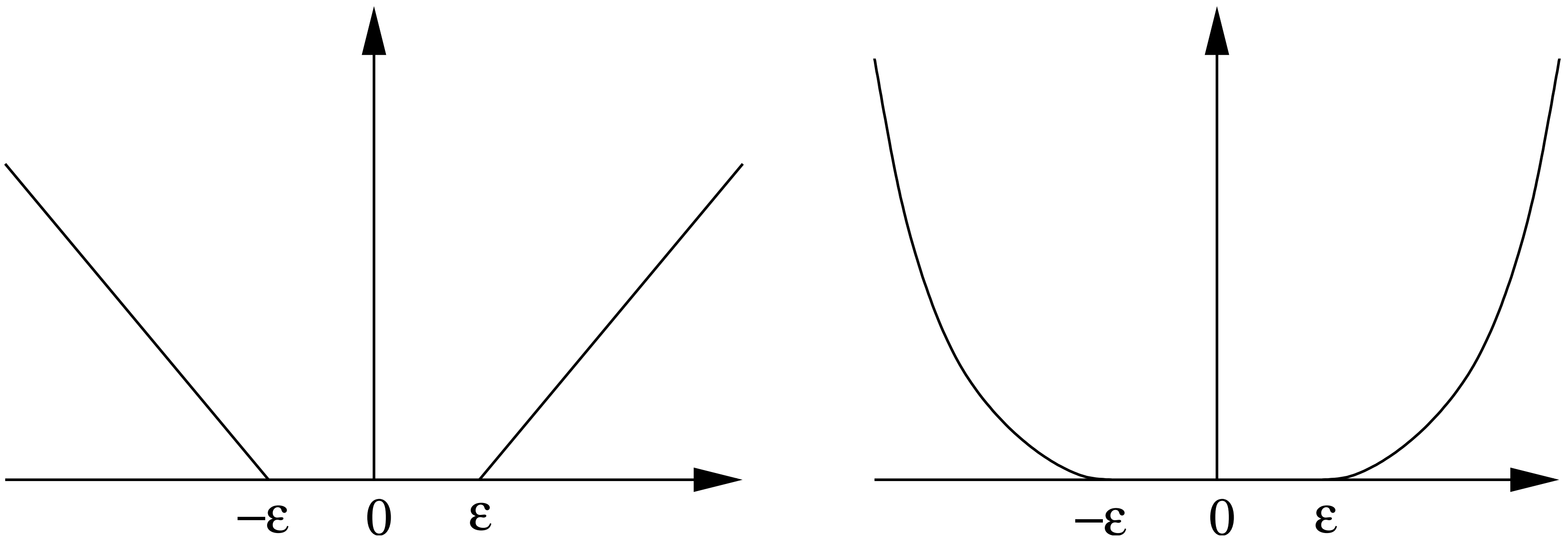
Regression with Margins

- 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

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

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_i \left(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x_i})\right)^2$$

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

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

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_i \left(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x_i})\right)^2$$

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

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

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_i \big(y_i - \boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x_i})\big)^2$$

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

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

$$\min_{\boldsymbol{w}} \lambda \|\boldsymbol{w}\|^2 + \sum_i \big(y_i - \boldsymbol{w}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x_i})\big)^2$$

- the $\|\boldsymbol{w}\|^2$ is a regularisation term
- As $\boldsymbol{w} = \sum_i \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)$ 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
  - ⋆ Fisher discriminant analysis
  - ⋆ Principle component analysis
  - ⋆ Canonical correlation analysis
  - ⋆ 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
  - ⋆ Fisher discriminant analysis
  - ⋆ Principle component analysis
  - ⋆ Canonical correlation analysis
  - ⋆ 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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . .

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 can be slow for very large datasets, but there are approximation methods to get around this
- There are lots of good SVM libraries, but care is need using them (normalising inputs and tuning parameters)
- Even when you understand all the mathematics. . . they are still magic