Artificial Intelligence II (CS4442B & CS9542B)

More on Kernels

Boyu Wang
Department of Computer Science
University of Western Ontario

Kernel Trick

Kernel trick

- Recall: in SVMs, for a feature mapping function: $\phi : \mathbb{R}^n \to \mathbb{R}^{n'} : x \to \phi(x)$, we define the kernel function as

$$k(x, z) = \phi(x)^\top \phi(z)$$

- In other words, kernel functions are ways of expressing dot-products in some feature space.
- If we work with a dual formulation of the learning algorithm, we do not actually have to deal feature mapping ϕ. We just have to compute the kernel function $k(x, z)$.

Kernel trick

$$x \in \mathbb{R}^n \to \phi(x) \in \mathbb{R}^{n'}, \qquad k(x,z) = \phi(x)^\top \phi(z)$$

Training

$$\max_\alpha \ \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j k(x_i, x_j)$$

$$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \ i = 1, \ldots, n$$

Prediction

$$f(x) = \operatorname{sgn}\Big(\sum_{i=1}^m \alpha_i y_i k(x_i, x) + b\Big)$$

Kernel trick

$$x \in \mathbb{R}^n \to \phi(x) \in \mathbb{R}^{n'}, \qquad k(x,z) = \phi(x)^\top \phi(z)$$

Training

$$\max_\alpha \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

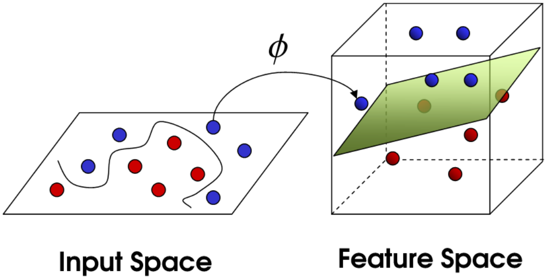
$$\text{s.t. } \sum_{i=1}^m \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \; i = 1, \ldots, n$$

Prediction

$$f(x) = \mathrm{sgn}\Big(\sum_{i=1}^m \alpha_i y_i k(x_i, x) + b \Big)$$

- The computation does not depend on n or n', but the number of training instances m.
- ϕ can map $x \in \mathbb{R}^n$ to $\phi(x) \in \mathbb{R}^{n'}$, where n' can be much larger than n (even infinity).

Nonlinear mapping and kernel trick



https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- First define $\phi(x)$, then $k(x, z) = \phi(x)^\top \phi(z)$:

$$\phi(x) = x \quad \Rightarrow \quad k(x, z) = x^\top z$$

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- First define $\phi(x)$, then $k(x, z) = \phi(x)^\top \phi(z)$:

$$\phi(x) = x \quad \Rightarrow \quad k(x, z) = x^\top z$$

- In practice, we define directly a kernel function k:

$$k(x, z) = (x^\top z)^2$$

  Is this a kernel?

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- First define $\phi(x)$, then $k(x, z) = \phi(x)^\top \phi(z)$:

$$\phi(x) = x \quad \Rightarrow \quad k(x, z) = x^\top z$$

- In practice, we define directly a kernel function k:

$$k(x, z) = (x^\top z)^2$$

  Is this a kernel?

- Let $x = [x_1, \ldots, x_n]^\top$, and $z = [z_1, \ldots, z_n]^\top$ (notation is overloaded), we have

$$k(x, z) = \left(\sum_{i=1}^{n} x_i z_i \right)^2 = \sum_{i,j=1}^{n} x_i z_i x_j z_j = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)$$

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- First define $\phi(x)$, then $k(x, z) = \phi(x)^\top \phi(z)$:
$$\phi(x) = x \quad \Rightarrow \quad k(x, z) = x^\top z$$
- In practice, we define directly a kernel function k:
$$k(x, z) = (x^\top z)^2$$
Is this a kernel?
- Let $x = [x_1, \ldots, x_n]^\top$, and $z = [z_1, \ldots, z_n]^\top$ (notation is overloaded), we have
$$k(x, z) = \left(\sum_{i=1}^{n} x_i z_i \right)^2 = \sum_{i,j=1}^{n} x_i z_i x_j z_j = \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)$$
- Hence, it is a valid kernel, with feature mapping:
$$\phi(x) = [x_1^2,\ x_1 x_2,\ \ldots, x_1 x_n,\ x_2 x_1,\ x_2^2, \ldots, x_n^2] \in \mathbb{R}^{n^2}$$
Feature vector includes all squares of elements and all cross terms.

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- How about a Gaussian kernel?

$$k(x, z) = e^{-\frac{||x-z||_2^2}{2\sigma^2}}$$

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- How about a Gaussian kernel?

$$k(x, z) = e^{-\frac{||x-z||_2^2}{2\sigma^2}}$$

- It is also a valid kernel, with an infinite-dimensional feature mapping

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- How about a Gaussian kernel?

$$k(x, z) = e^{-\frac{||x-z||_2^2}{2\sigma^2}}$$

- It is also a valid kernel, with an infinite-dimensional feature mapping
- For one-dimensional input $x \in \mathbb{R}$, the mapping function is

$$\phi(x) = e^{-x^2/2\sigma^2} \left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \ldots \right]^\top$$

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- How about a Gaussian kernel?

$$k(x, z) = e^{-\frac{||x-z||_2^2}{2\sigma^2}}$$

- It is also a valid kernel, with an infinite-dimensional feature mapping
- For one-dimensional input $x \in \mathbb{R}$, the mapping function is

$$\phi(x) = e^{-x^2/2\sigma^2} \left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \ldots\right]^\top$$

- In general, given a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to R$, under what conditions $k(x, z)$ can be written as a dot product $\phi(x)^\top \phi(z)$ for some feature mapping ϕ?

How to find a valid kernel

By definition, a kernel function k must be the inner product in the feature space defined by ϕ: $k(x, z) = \phi(x)^\top \phi(z)$.

- How about a Gaussian kernel?

$$k(x, z) = e^{-\frac{||x-z||_2^2}{2\sigma^2}}$$

- It is also a valid kernel, with an infinite-dimensional feature mapping
- For one-dimensional input $x \in \mathbb{R}$, the mapping function is

$$\phi(x) = e^{-x^2/2\sigma^2}\left[1, \sqrt{\frac{1}{1!\sigma^2}}x, \sqrt{\frac{1}{2!\sigma^4}}x^2, \sqrt{\frac{1}{3!\sigma^6}}x^3, \ldots\right]^\top$$

- In general, given a kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to R$, under what conditions $k(x, z)$ can be written as a dot product $\phi(x)^\top \phi(z)$ for some feature mapping ϕ?
- We want a general recipe, which does not require explicitly defining ϕ every time.

Kernel matrix

- Suppose we have an arbitrary set of input vectors: $\{x_i\}_{i=1}^{m}$
- The kernel matrix (or Gram matrix) $K \in \mathbb{R}^{m \times m}$ corresponding to kernel function k is an $m \times m$ matrix such that $K_{ij} = k(x_i, x_j)$

Kernel matrix

- Suppose we have an arbitrary set of input vectors: $\{x_i\}_{i=1}^m$
- The kernel matrix (or Gram matrix) $K \in \mathbb{R}^{m \times m}$ corresponding to kernel function k is an $m \times m$ matrix such that $K_{ij} = k(x_i, x_j)$
- What the properties does the kernel matrix K have if k is a valid kernel function?
  1. K is a symmetric matrix (i.e., $K_{ij} = K_{ji}$)
  2. K is positive semidefinite (i.e., $\alpha^\top K\alpha \geq 0, \ \forall \alpha \in \mathbb{R}^m$)

Proofs

1. $K_{ij} = \phi(x_i)^\top \phi(x_j) = \phi(x_j)^\top \phi(x_i) = K_{ji}$

Proofs

1. $K_{ij} = \phi(x_i)^\top \phi(x_j) = \phi(x_j)^\top \phi(x_i) = K_{ji}$

2. For any vector $\alpha = [\alpha_1, \ldots \alpha_m]^\top \in \mathbb{R}^m$ and $\phi(x) = [\phi_1(x), \ldots, \phi_{n'}(x)]^\top \in \mathbb{R}^{n'}$ we have

$$
\begin{aligned}
\alpha^\top K \alpha &= \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i K_{ij} \alpha_j && \text{(definition of matrix-vector product)} \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \left(\phi(x_i)^\top \phi(x_j) \right) \alpha_j && \text{(definition of kernel matrix)} \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \left(\sum_{k=1}^{n'} \phi_k(x_i) \cdot \phi_k(x_j) \right) \alpha_j && \text{(definition of inner product)} \\
&= \sum_{k=1}^{n'} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \phi_k(x_i) \cdot \phi_k(x_j) \alpha_j && \text{(change the order of summation)} \\
&= \sum_{k=1}^{n'} \left(\sum_{i=1}^{m} \alpha_i \phi_k(x_i) \right)^2 \geq 0 && \left((\sum_{i=1}^{m} x_i)^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} x_i x_j \right)
\end{aligned}
$$

Mercer's theorem

- We have shown that if k is a valid kernel function, then for any data set, the corresponding kernel matrix K defined such that $K_{ij} = k(x_i, x_j)$ is symmetric and positive semidefinite.

Mercer's theorem

- We have shown that if k is a valid kernel function, then for any data set, the corresponding kernel matrix K defined such that $K_{ij} = k(x_i, x_j)$ is symmetric and positive semidefinite.
- Mercer's theorem states that the reverse is also true:
Given a function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, k is a valid kernel function if and only if, for any data set, the corresponding kernel matrix K is symmetric and positive semidefinite

Mercer's theorem

- We have shown that if k is a valid kernel function, then for any data set, the corresponding kernel matrix K defined such that $K_{ij} = k(x_i, x_j)$ is symmetric and positive semidefinite.
- Mercer's theorem states that the reverse is also true:
  Given a function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, k is a valid kernel function if and only if, for any data set, the corresponding kernel matrix K is symmetric and positive semidefinite
- The reverse direction of the proof is much harder.

Mercer's theorem

- We have shown that if k is a valid kernel function, then for any data set, the corresponding kernel matrix K defined such that $K_{ij} = k(x_i, x_j)$ is symmetric and positive semidefinite.
- Mercer's theorem states that the reverse is also true: Given a function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, k is a valid kernel function if and only if, for any data set, the corresponding kernel matrix K is symmetric and positive semidefinite
- The reverse direction of the proof is much harder.
- It gives us a way to check if a given function is a kernel, by checking these two properties of its kernel matrix.

Construct a kernel with kernels

Let k_1 and k_2 be valid kernels over $\mathbb{R}^n \times \mathbb{R}^n$, $a \in \mathbb{R}_+$ be a positive number, $\phi(x) : \mathbb{R}^n \to \mathbb{R}^{n'}$ be a mapping function, with a kernel k_3 defined over $\mathbb{R}^{n'} \times \mathbb{R}^{n'}$, and $A \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix. Then, the following functions are kernels:

1. $k(x, z) = k_1(x, z) + k_2(x, z)$
2. $k(x, z) = ak_1(x, z)$
3. $k(x, z) = k_1(x, z)k_2(x, z)$
4. $k(x, z) = \phi(x)\phi(z)$
5. $k(x, z) = k_3(\phi(x), \phi(z))$
6. $k(x, z) = x^\top Az$

Kernelized Linear Regression

Linear regression revisited

- (Regularized) linear regression aims to minimize the loss function (we omit the bias term b for simplicity):

$$L(w) = \frac{1}{2}||Xw - y||_2^2 + \frac{\lambda}{2}||w||_2^2$$

- If we use a mapping function $\phi : \mathbb{R}^n \to \mathbb{R}^{n'} : x \to \phi(x)$ for data pre-processing, then we have

$$L(u) = \frac{1}{2}||\mathbf{\Phi} u - y||_2^2 + \frac{\lambda}{2}||u||_2^2$$

where $\mathbf{\Phi} = [\phi(x_1), \ldots, \phi(x_m)]^\top \in \mathbb{R}^{m\times n'}$ is the matrix of data points in the new feature space, and $u \in \mathbb{R}^{n'}$ is the corresponding weight vector

Kernelized linear regression

$$L(u) = \frac{1}{2}||\mathbf{\Phi} u - y||_2^2 + \frac{\lambda}{2}||u||_2^2$$

- Assume that u can be represented as a linear combination of $\phi(x_i)$:

$$u = \mathbf{\Phi}^\top \alpha$$

  where $\alpha = [\alpha_1, \ldots, \alpha_m]^\top \in \mathbb{R}^m$ (analogous to the α_i's in SVM)

- Then, the objective function becomes:

$$\begin{aligned} L(\alpha) &= \frac{1}{2}||\mathbf{\Phi}\mathbf{\Phi}^\top \alpha - y||_2^2 + \frac{\lambda}{2}||\mathbf{\Phi}^\top \alpha||_2^2 \\ &= \frac{1}{2}\alpha^\top \mathbf{\Phi}\mathbf{\Phi}^\top \mathbf{\Phi}\mathbf{\Phi}^\top \alpha - \alpha^\top \mathbf{\Phi}\mathbf{\Phi}^\top y + \frac{1}{2}y^\top y + \frac{\lambda}{2}\alpha^\top \mathbf{\Phi}\mathbf{\Phi}^\top \alpha \\ &= \frac{1}{2}\alpha^\top K K \alpha - \alpha^\top K y + \frac{1}{2}y^\top y + \frac{\lambda}{2}\alpha^\top K \alpha \end{aligned}$$

  where $K \triangleq \mathbf{\Phi}\mathbf{\Phi}^\top \in \mathbb{R}^{m\times m}$, with $K_{ij} = \phi(x_i)^\top \phi(x_j) = k(x_i, x_j)$.

- In other words, K is a kernel matrix!

Kernelized linear regression

$$L(\alpha) = \frac{1}{2}\alpha^\top KK\alpha - \alpha^\top Ky + \frac{1}{2}y^\top y + \frac{\lambda}{2}\alpha^\top K\alpha$$

- This is a quadratic function with respect to α, and we can find the solution by setting the gradient of J_α to zero and solve for α:

$$\alpha = (K + \lambda I_m)^{-1} y$$

- Once we obtain α, we can predict the value of x by using

$$\begin{aligned} f(x) &= \phi(x)^\top u \\ &= \phi(x)^\top \mathbf{\Phi}^\top \alpha \\ &= \sum_{i=1}^{m} \alpha_i k(x_i, x) \end{aligned}$$

  Again, the feature mapping function ϕ is not needed!

- Recall: SVM prediction: $f(x) = \text{sgn}\left(\sum_{i=1}^m \alpha_i y_i k(x_i, x) + b\right)$ – similar form for prediction!

- Demo!

Kernelized Logistic Regression

Kernelized logistic regression

- Recall: in linear logistic regression, we have

$$p(y = 1|x; w) \triangleq \sigma(h_w(x)) = \frac{1}{1 + e^{-w^\top x}},$$

- Similarly, we use a mapping function $\phi : x \to \phi(x)$:

$$\sigma(h_u(x)) = \frac{1}{1 + e^{-u^\top \phi(x)}}$$

and assume that $u = \mathbf{\Phi}^\top \alpha$

- Then, we have

$$\sigma(h_\alpha(x)) = \frac{1}{1 + e^{-\sum_{i=1}^{m} \alpha_i k(x, x_i)}}$$

Kernelized logistic regression

- Recall: given a training set, the objective function (cross-entropy loss) function of linear logistic regression is

$$J(w) = -\sum_{i=1}^{m} \left(y_i \log t_i + (1 - y_i) \log(1 - t_i) \right), \tag{1}$$

where $t_i = \sigma(h_w(x_i)) = \frac{1}{1+e^{-w^\top x_i}}$

- For kernelized logistic regression, the objective function is still the same as (1), and the only difference is t_i:

$$t_i = \sigma(h_\alpha(x_i)) = \frac{1}{1 + e^{-\sum_{j=1}^{m} \alpha_j k(x_i, x_j)}}$$

- The training procedure is the same as for linear logistic regression (e.g., gradient descent or Newton's method)