Lecture 10: Kernel Methods
CS480/680 Intro to Machine Learning

2023-2-9

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Non-linear Models Recap

- Generalized linear models:

  - fixed non-linear basis functions
  - limited hypothesis space
  - easy to optimize (usually convex)

- Neural networks:

  - adaptive non-linear basis functions
  - rich hypothesis space
  - hard to optimize (usually non-convex)

Kernel Methods

- Idea: use large (possibly infinite) set of fixed non-linear basis functions
- Normally, complexity depends on number of basis functions, but by a "dual trick", **complexity depends on the amount of data**

- Examples:
  - **Gaussian Processes** (next class)
  - **Support Vector Machines** (next week)
  - Kernel perceptron
  - Kernel logistic regression

Kernel Function

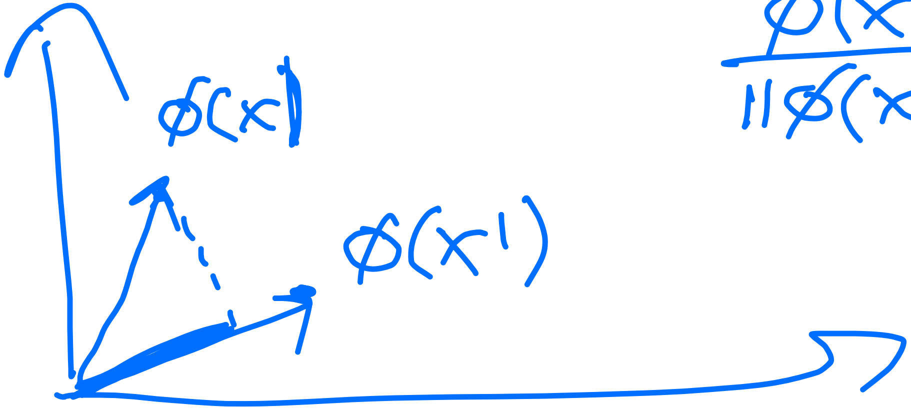
- Let $\phi(\boldsymbol{x})$ be a set of basis functions that map inputs $\boldsymbol{x}$ to a feature space.

- In many algorithms, this feature space only appears in the dot product $\phi(\boldsymbol{x})^T\phi(\boldsymbol{x}')$ of input pairs $\boldsymbol{x}, \boldsymbol{x}'$.

- Define the kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}')$ to be the dot product of any pair $\boldsymbol{x}, \boldsymbol{x}'$ in feature space.
  - **We only need to know** $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$, not $\phi(\boldsymbol{x})$

Illustration of Kernel Function

- $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$
- Intuition: $k(\boldsymbol{x}, \boldsymbol{x}')$ measures degree of similarity



$$\frac{\phi(x)^T \phi(x')}{\|\phi(x)\| \, \|\phi(x')\|}$$

Dual Representations

- Recall linear regression objective

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}[\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n]^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

- Solution: set gradient to 0

$$\nabla E(\boldsymbol{w}) = \sum_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)\phi(\boldsymbol{x}_n) + \lambda\boldsymbol{w} = 0$$

$$\underbrace{\boldsymbol{w}}_{\text{vector}} = -\frac{1}{\lambda}\sum_n\underbrace{(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)}_{\text{scalar}}\underbrace{\phi(\boldsymbol{x}_n)}_{\text{vector}}$$

coefficient

$\phi(x)$

w

$\boxed{\ } = \sum \boxed{\ }\ \boxed{\ }$

$\therefore$ ***w* is a linear combination of inputs in feature space**

$$\{\phi(\boldsymbol{x}_n) | 1 \leq n \leq N\}$$

UNIVERSITY OF WATERLOO

Dual Representations

- Substitute $\mathbf{w} = \boldsymbol{\Phi a}$
- Where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1)\ \phi(\boldsymbol{x}_2)\ \ldots\ \phi(\boldsymbol{x_N})]$

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \text{and } a_n = -\frac{1}{\lambda}(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - y_n)$$

- Dual objective: minimize E with respect to $\boldsymbol{a}$

$$E(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{\Phi a} - \boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi y} + \frac{\boldsymbol{y}^T\boldsymbol{y}}{2} + \frac{\lambda}{2}\boldsymbol{a}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi a}$$

Gram Matrix

$k(x_1, x_1) = \phi(x_1)^T \phi(x_1)$

$k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$

- Let $\boldsymbol{K} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}$ be the Gram matrix
- Substitute in objective:

$$E(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T \boldsymbol{K}\boldsymbol{K}\boldsymbol{a} - \boldsymbol{a}^T \boldsymbol{K}\boldsymbol{y} + \frac{\boldsymbol{y}^T\boldsymbol{y}}{2} + \frac{\lambda}{2}\boldsymbol{a}^T \boldsymbol{K}\boldsymbol{a}$$

- Solution: set gradient to 0

$$\nabla E(\boldsymbol{a}) = \boldsymbol{K}\boldsymbol{K}\boldsymbol{a} - \boldsymbol{K}\boldsymbol{y} + \lambda \boldsymbol{K}\boldsymbol{a} = 0$$

$$\boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{a} = \boldsymbol{K}\boldsymbol{y}$$

$$\boldsymbol{a} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

row vector

- Prediction:

$$y_* = \phi(\boldsymbol{x}_*)^T \boldsymbol{w} = \phi(\boldsymbol{x}_*)^T \boldsymbol{\Phi}\boldsymbol{a} = k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

where $(\boldsymbol{X}, \boldsymbol{y})$ is the training set and $(\boldsymbol{x}_*, y_*)$ is a test instance

Dual Linear Regression

- Prediction: $y_* = \phi(\boldsymbol{x}_*)^T \boldsymbol{\Phi a}$

$$= k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

- Linear regression where we find dual solution $\boldsymbol{a}$ instead of primal solution $\mathbf{w}$.

- Complexity:
  - Primal solution: depends on # of basis functions
  - Dual solution: depends on amount of data
    - Advantage: can use very large # of basis functions
    - Just need to know kernel k

Constructing Kernels

- Two possibilities:
  - Find mapping $\boldsymbol{\phi}$ to feature space and let $\boldsymbol{K} = \boldsymbol{\phi}^{\boldsymbol{T}}\boldsymbol{\phi}$
  - Directly specify $\boldsymbol{K}$

- Can any function that takes two arguments serve as a kernel?

- No, a valid kernel must be positive semi-definite
  - In other words, k must factor into the product of a transposed matrix by itself (e.g., $\boldsymbol{K} = \boldsymbol{\phi}^{\boldsymbol{T}}\boldsymbol{\phi}$)
  - Or all eigenvalues must be greater than or equal to 0.

Example

- Let $k(\boldsymbol{x}, \boldsymbol{z}) = \left(\boldsymbol{x}^T \boldsymbol{z}\right)^2$

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

$$= (x_1 z_1 + x_2 z_2)^2$$

$$= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2$$

$$= (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \begin{pmatrix} z_1^2 \\ \sqrt{2} z_1 z_2 \\ z_2^2 \end{pmatrix}$$

$$= \phi(x)^T \phi(z)$$

$$\hookrightarrow \phi_1(x) = x_1^2$$
$$\phi_2(x) = \sqrt{2} x_1 x_2$$
$$\phi_3(x) = x_2^2$$

UNIVERSITY OF WATERLOO

Constructing Kernels

- Can we construct k directly without knowing ϕ?

- Yes, any positive semi-definite k is fine since there is a corresponding implicit feature space. But positive semi-definiteness is not always easy to verify.

- Alternative, construct kernels from other kernels using rules that preserve positive semi-definiteness

Rules to construct Kernels

- Let $k_1(\boldsymbol{x}, \boldsymbol{x}')$ and $k_2(\boldsymbol{x}, \boldsymbol{x}')$ be valid kernels
- The following kernels are also valid:

1. $k(\boldsymbol{x}, \boldsymbol{x}') = ck_1(\boldsymbol{x}, \boldsymbol{x}') \quad \forall c > 0$
2. $k(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}') \quad \forall f$
3. $k(\boldsymbol{x}, \boldsymbol{x}') = q(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ q is polynomial with coeffs ≥ 0
4. $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(k_1(\boldsymbol{x}, \boldsymbol{x}')\right)$
5. $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}')$
6. $k(\boldsymbol{x}, \boldsymbol{x}') = k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}')$
7. $k(\boldsymbol{x}, \boldsymbol{x}') = k_3(\phi(\boldsymbol{x}), \phi(\boldsymbol{x}'))$
8. $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{x}'$ $\boldsymbol{A}$ is symmetric positive semi-definite
9. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x}_{\boldsymbol{a}}, \boldsymbol{x}'_a) + k_b(\boldsymbol{x}_{\boldsymbol{b}}, \boldsymbol{x}'_b)$
10. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b)$

where $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{pmatrix}$

Common Kernels

- Polynomial kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}')^M$
  - M is the degree
  - Feature space: all degree M products of entries in $\boldsymbol{x}$
  - Example: Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two images, then feature space could be all products of M pixel intensities

- More general polynomial kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^M \quad \text{with } c > 0$$

  - Feature space: all products of up to M entries in $\boldsymbol{x}$

Example

$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad x' = \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}$

- $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^2$

$= (x_1 x_1' + x_2 x_2' + c)^2$

$= x_1^2 x_1'^2 + 2 x_1 x_1' x_2 x_2' + x_2^2 x_2'^2 + 2 x_1 x_1' c + 2 x_2 x_2' c + c^2$

$= (x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2c}\, x_1, \sqrt{2c}\, x_2, c)$
$(x_1'^2, \sqrt{2} x_1' x_2', x_2'^2, \sqrt{2c}\, x_1', \sqrt{2c}\, x_2', c)^T$

Common Kernels

- Gaussian Kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2}{2\sigma^2}\right)$
- Valid Kernel because: $= e^{-x^T x/2\sigma^2} e^{x^T x'/\sigma^2} e^{-x'^T x'/2\sigma^2}$

$x^T x'$ is a valid kernel by rule 8 when $A = I$

$x^T x'/\sigma^2$ " " " " " 1

$e^{x^T x'/\sigma^2}$ " " " " " 4

$k(x, x')$ " " " " " 2

- Implicit feature space is infinite!

Non-vectorial Kernels

- Kernels can be defined with respect to other things than vectors such as sets, strings or graphs
- Example for strings: $k(d_1, d_2) =$ similarity between two documents (weighted sum of all non-contiguous strings that appear in both documents d_1 and d_2).
- Lodhi, Saunders, Shawe-Taylor, Christianini, Watkins, **Text Classification Using String Kernels**, JMLR, p. 419-444, 2002.