

# CS480/680

## Lecture 11: June 12, 2019

Kernel methods

[D] Chap. 11 [B] Sec. 6.1, 6.2

[M] Sec. 14.1, 14.2

[HTF] Chap. 6



# 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 Principal Component Analysis

# Kernel Function

- Let  $\phi(\mathbf{x})$  be a set of basis functions that map inputs  $\mathbf{x}$  to a feature space.
- In many algorithms, this feature space only appears in the dot product  $\phi(\mathbf{x})^T \phi(\mathbf{x}')$  of input pairs  $\mathbf{x}, \mathbf{x}'$ .
- Define the kernel function  $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$  to be the dot product of any pair  $\mathbf{x}, \mathbf{x}'$  in feature space.
  - **We only need to know  $k(\mathbf{x}, \mathbf{x}')$ , not  $\phi(\mathbf{x})$**

# Dual Representations

- Recall linear regression objective

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

- Solution: set gradient to 0

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

$$\mathbf{w} = -\frac{1}{\lambda} \sum_n (\mathbf{w}^T \phi(\mathbf{x}_n) - y_n) \phi(\mathbf{x}_n)$$

**$\therefore \mathbf{w}$  is a linear combination of inputs in feature space**  
 **$\{\phi(\mathbf{x}_n) | 1 \leq n \leq N\}$**

# Dual Representations

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

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

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

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

# Gram Matrix

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

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

- Solution: set gradient to 0

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

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

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

- Prediction:

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

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

# Dual Linear Regression

- Prediction:  $y_* = \phi(\mathbf{x}_*)^T \Phi \mathbf{a}$   
 $= k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$
- Linear regression where we find dual solution  $\mathbf{a}$  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  $\phi$  to feature space and let  $K = \phi^T \phi$
  - Directly specify  $K$
- 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.,  $K = \phi^T \phi$ )
  - Or, all eigenvalues must be greater than or equal to 0.

# Example

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

# Constructing Kernels

- Can we construct  $k$  directly without knowing  $\phi$ ?
- 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  $k_1(\mathbf{x}, \mathbf{x}')$  and  $k_2(\mathbf{x}, \mathbf{x}')$  be valid kernels
  - The following kernels are also valid:
    1.  $k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad \forall c > 0$
    2.  $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad \forall f$
    3.  $k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}'))$   $q$  is polynomial with coeffs  $\geq 0$
    4.  $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$
    5.  $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
    6.  $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$
    7.  $k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$
    8.  $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$   $\mathbf{A}$  is symmetric positive semi-definite
    9.  $k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b)$
    10.  $k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)$
- where  $\mathbf{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$

# Common Kernels

- Polynomial kernel:  $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^M$ 
  - $M$  is the degree
  - Feature space: all degree  $M$  products of entries in  $\mathbf{x}$
  - Example: Let  $\mathbf{x}$  and  $\mathbf{x}'$  be two images, then feature space could be all products of  $M$  pixel intensities
- More general polynomial kernel:
$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^M \quad \text{with } c > 0$$
  - Feature space: all products of up to  $M$  entries in  $\mathbf{x}$

# Common Kernels

- Gaussian Kernel:  $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$
- Valid Kernel because:
  
  
  
  
  
  
  
  
  
  
- 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.