Kernel methods

[D] Chap. 11 [B] Sec. 6.1, 6.2
[M] Sec. 14.1, 14.2 [H] Chap. 9
[HTF] Chap. 6
Non-linear Models Recap

- Generalized linear models:

- Neural networks:
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 $\phi(x)$ be a set of basis functions that map inputs $x$ to a feature space.

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

• Define the kernel function $k(x, x') = \phi(x)^T \phi(x')$ to be the dot product of any pair $x, x'$ in feature space.
  – We only need to know $k(x, x')$, not $\phi(x)$
Dual Representations

• Recall linear regression objective

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

• Solution: set gradient to 0

\[ \nabla E(w) = \sum_n (w^T \phi(x_n) - y_n) \phi(x_n) + \lambda w = 0 \]
\[ w = -\frac{1}{\lambda} \sum_n (w^T \phi(x_n) - y_n) \phi(x_n) \]

∴ \( w \) is a linear combination of inputs in feature space
\[ \{ \phi(x_n) | 1 \leq n \leq N \} \]
Dual Representations

• Substitute $w = \Phi a$
• Where $\Phi = [\phi(x_1) \phi(x_2) \ldots \phi(x_N)]$

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

• Dual objective: minimize $E$ with respect to $a$

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

- Let $K = \Phi^T \Phi$ be the Gram matrix
- Substitute in objective:
  $$E(\alpha) = \frac{1}{2} a^T KK a - a^T Ky + \frac{y^T y}{2} + \frac{\lambda}{2} a^T Ka$$
- Solution: set gradient to 0
  $$\nabla E(\alpha) = KK a - Ky + \lambda K a = 0$$
  $$K(K + \lambda I) a = Ky$$
  $$a = (K + \lambda I)^{-1} y$$
- Prediction:
  $$y_* = \phi(x_*)^T w = \phi(x_*)^T \Phi a = k(x_*, X)(K + \lambda I)^{-1} y$$
  where $(X, y)$ is the training set and $(x_*, y_*)$ is a test instance
Dual Linear Regression

• Prediction: \( y_\star = \phi(x_\star)^T \Phi a \)
  \[ = k(x_\star, X)(K + \lambda I)^{-1}y \]

• Linear regression where we find dual solution \( a \)
  instead of primal solution \( w \).

• 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 $k(x, z) = (x^T z)^2$
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 $k_1(x, x')$ and $k_2(x, x')$ be valid kernels
- The following kernels are also valid:
  1. $k(x, x') = c k_1(x, x') \quad \forall c > 0$
  2. $k(x, x') = f(x) k_1(x, x') f(x') \quad \forall f$
  3. $k(x, x') = q(k_1(x, x')) \quad q$ is polynomial with coeffs $\geq 0$
  4. $k(x, x') = \exp(k_1(x, x'))$
  5. $k(x, x') = k_1(x, x') + k_2(x, x')$
  6. $k(x, x') = k_1(x, x') k_2(x, x')$
  7. $k(x, x') = k_3(\phi(x), \phi(x'))$
  8. $k(x, x') = x^T A x' \quad A$ is symmetric positive semi-definite
  9. $k(x, x') = k_a(x_a, x'_a) + k_b(x_b, x'_b)$
  10. $k(x, x') = k_a(x_a, x'_a) k_b(x_b, x'_b) \quad \text{where } x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$
Common Kernels

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

• More general polynomial kernel:
  
  $$k(x, x') = (x^T x' + c)^M \text{ with } c > 0$$
  
  – Feature space: all products of up to $M$ entries in $x$
Common Kernels

• Gaussian Kernel: \( k(x, x') = \exp\left(-\frac{|x-x'|^2}{2\sigma^2}\right) \)

• 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 \( d_2 \)).