Kernel Methods

COMP 135 Intro to Machine Learning
Liping Liu

slides modified from Roni Khardon’s with permission

The XOR problem

• Feature expansion \((x_1, x_2) \Rightarrow (x_1, x_2, x_1x_2)\)
• Now linearly separable!

Feature expansion

• New representation of features
  - \(x \Rightarrow \phi(x)\)
• Combine different features to increase the dimensionality of instances
• Build classifier in the new feature space
  - Increased classifier flexibility

The new decision boundary

• The XOR example: the learned model is
  \[ w = \begin{pmatrix} 2/3 \\ 2/3 \\ 4/3 \end{pmatrix}, \quad w_0 = -5/3 \]
• The decision boundary is
  \[ \frac{2}{3}x_1 + \frac{2}{3}x_2 + \frac{4}{3}x_1x_2 - \frac{5}{3} = 0 \]
  or
  \[ x_1 = \frac{2x_2 - 5}{-2 - 4x_2} \]

Kernel trick

• How to extend existing classifiers?
  - There is a systematic method for linear classifiers
• How do we come up expansions \(\phi(x)\)
  - Actually we don’t directly work with \(\phi(x)\)

Representer theorem

Theorem: For SVM classifier, the weight vector \(w\) is always in the span of the (assumed non-empty) training data, \(\phi(x_1), \ldots, \phi(x_N)\).

\(w\) is in the span of the \(\phi(x_1), \ldots, \phi(x_N)\) if and only if we can express \(w\) as follows with at least one \(a_i \neq 0\).

\[ w = \sum_{i=1}^{N} \beta_i \cdot \phi(x_i) \]
SVM prediction with representer thm

- SVM score

\[ x_{SVM}(x) = w_0 + \sum_{i=1}^{N} \beta_i \phi(x_i)^T \phi(x) \]

It works as long as we can calculate the inner product

Kernel function

- Kernel function \( \kappa(x_i, x_j) \) is equivalent to the inner product \( \phi(x_i)^T \phi(x_j) \)
- Work directly with \( \kappa(x_i, x) \)
- No need to come up functions \( \phi(\cdot) \)
- \( \kappa(\cdot, \cdot) \) need to satisfy properties (out of the scope of this class), need to be sth like inner product

Kernel examples

- Linear kernel
  - \( \kappa(x_i, x_j) = x_i^T x_j \)
  - Back to linear classifiers

- RBF (Radial Basis Function) / Gaussian kernel

\[ \kappa(x_i, x_j) = \exp \left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]

[land et al., 2011]

Kernel trick

- (Potentially) represent model parameters as

\[ w = \sum_{i=1}^{N} \beta_i \cdot \phi(x_i) \]

- Calculate inner products by

\[ \phi(x_i)^T \phi(x_j) = \kappa(x_i, x_j) \]

Kernel SVM

- The predictive model is

\[ f(x) = \sum_{i=1}^{N} d_{ij} \kappa(x_i, x) > 0 \]

- Now the model parameter is \( \alpha = (\alpha_i)_{i=1}^{N} \)
  - \( \alpha \) is solved from the SVM dual problem
**SVM primal and dual**

Primal: \( \min_{\|w\|} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i (w^T x_i + w_0)) \)

Dual: \( \max_a \sum_i^N a_i - \frac{1}{2} \sum_{i,j=1}^N a_i a_j y_i y_j k(x_i, x_j) \)
\( \text{s.t.} \quad 0 \leq a_i \leq C, \quad \sum_i^N a_i y_i = 0 \)

Primal and dual are equivalent problems
Dual problem allows kernel trick

**Understand SVM dual**

- Support vectors: training instances with \( \alpha_i > 0 \)
- Neglect non-support vectors (training instances with \( \alpha_i = 0 \)) then predicting

From wikipedia

**SVM with RBF kernel**

- Interactive interface at LibSVM website
  (https://www.csie.ntu.edu.tw/~cjlin/libsvm/)
- One nice example at

**Kernel trick for other classifiers**

- For distance-based models
  \( \|x_i - x_j\| = \sqrt{(x_i - x_j)^T (x_i - x_j)} \)
  \( = x_i^T x_i + x_j^T x_j - 2x_i^T x_j \)
  \( = k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j) \)
- All distance-based models can use kernel trick