CSC 411: Lecture 16: Kernels

Class based on Raquel Urtasun & Rich Zemel’s lectures

Sanja Fidler

University of Toronto

March 21, 2016
Today

- Kernel trick
Summary of Linear SVM

- Binary and linear separable classification
- Linear classifier with maximal margin
- Training SVM by maximizing

\[
\max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j (x^{(i)T} \cdot x^{(j)}) \right\}
\]

subject to \( \alpha_i \geq 0; \sum_{i=1}^{N} \alpha_i t^{(i)} = 0 \)

- The weights are

\[
w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}
\]

- Only a small subset of \( \alpha_i \)'s will be nonzero, and the corresponding \( x^{(i)} \)'s are the support vectors \( S \)

- Prediction on a new example:

\[
y = \text{sign}[b + x \cdot (\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)})] = \text{sign}[b + x \cdot (\sum_{i \in S} \alpha_i t^{(i)} x^{(i)})]
\]
What if data is not linearly separable?

Introduce slack variables $\xi_i$

$$\min \frac{1}{2} ||\mathbf{w}||^2 + \lambda \sum_{i=1}^{N} \xi_i$$

s.t. $\xi_i \geq 0$; $\forall i \ t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 - \xi_i = 0$

- Example lies on wrong side of hyperplane $\xi_i > 1$
- Therefore $\sum_i \xi_i$ upper bounds the number of training errors
- $\lambda$ trades off training error vs model complexity
- This is known as the soft-margin extension
Non-linear Decision Boundaries

- Note that both the learning objective and the decision function depend only on dot products between patterns

\[
\ell = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j (x^{(i)T} \cdot x^{(j)})
\]

\[
y = \text{sign}[b + x \cdot (\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)})]
\]

- How to form non-linear decision boundaries in input space?
  1. Map data into feature space \( x \rightarrow \phi(x) \)
  2. Replace dot products between inputs with feature points

\[
x^{(i)T} x^{(j)} \rightarrow \phi(x^{(i)})^T \phi(x^{(j)})
\]

3. Find linear decision boundary in feature space

- Problem: what is a good feature function \( \phi(x) \)?
Mapping to a feature space can produce problems:

- High computational burden due to high dimensionality
- Many more parameters

SVM solves these two issues simultaneously

- “Kernel trick” produces efficient classification
- Dual formulation only assigns parameters to samples, not features
Kernel Trick

- **Kernel trick**: dot-products in feature space can be computed as a kernel function
  \[ K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)}) \]

- Idea: work directly on \( x \), avoid having to compute \( \phi(x) \)

- Example:
  \[
  K(a, b) = (a^T b)^3 = ((a_1, a_2)^T (b_1, b_2))^3 \\
  = (a_1 b_1 + a_2 b_2)^3 \\
  = a_1^3 b_1^3 + 3a_1^2 b_1^2 a_2 b_2 + 3a_1 b_1 a_2^2 b_2^2 + a_2^3 b_2^3 \\
  = (a_1^3, \sqrt{3}a_1^2 a_2, \sqrt{3}a_1 a_2^2, a_2^3)^T (b_1^3, \sqrt{3}b_1^2 b_2, \sqrt{3}b_1 b_2^2, b_2^3) \\
  = \phi(a) \cdot \phi(b)
  \]
Kernels

- Examples of kernels: kernels measure similarity
  1. Polynomial
     \[ K(x^{(i)}, x^{(j)}) = (x^{(i)T} x^{(j)} + 1)^d \]
     where \( d \) is the degree of the polynomial, eg \( d = 2 \) for quadratic
  2. Gaussian
     \[ K(x^{(i)}, x^{(j)}) = \exp\left(-\frac{||x^{(i)} - x^{(j)}||^2}{2\sigma^2}\right) \]
  3. Sigmoid
     \[ K(x^{(i)}, x^{(j)}) = \tanh(\beta(x^{(i)T} x^{(j)}) + a) \]

- Each kernel computation corresponds to dot product
  - calculation for particular mapping \( \phi(x) \) implicitly maps to high-dimensional space

- Why is this useful?
  1. Rewrite training examples using more complex features
  2. Dataset not linearly separable in original space may be linearly separable in higher dimensional space
Kernel Functions

- Mercer’s Theorem (1909): any reasonable kernel corresponds to some feature space

- Reasonable means that the Gram matrix is positive definite

\[ K_{ij} = K(x^{(i)}, x^{(j)}) \]

- Feature space can be very large
  - polynomial kernel \((1 + (x^{(i)})^T x^{(j)})^d\) corresponds to feature space exponential in \(d\)
  - Gaussian kernel has infinitely dimensional features

- Linear separators in these super high-dim spaces correspond to highly nonlinear decision boundaries in input space
Classification with Non-linear SVMs

- Non-linear SVM using kernel function $K()$:

$$\ell = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)})$$

- Maximize $\ell$ w.r.t. $\{\alpha\}$ under constraints $\forall i, \alpha_i \geq 0$

- Unlike linear SVM, cannot express $w$ as linear combination of support vectors
  - now must retain the support vectors to classify new examples

- Final decision function:

$$y = \text{sign}[b + \sum_{i=1}^{N} t^{(i)} \alpha_i K(x, x^{(i)})]$$
Summary

- **Advantages:**
  - Kernels allow very flexible hypotheses
  - Poly-time exact optimization methods rather than approximate methods
  - Soft-margin extension permits mis-classified examples
  - Variable-sized hypothesis space
  - Excellent results (1.1% error rate on handwritten digits vs. LeNet’s 0.9%)

- **Disadvantages:**
  - Must choose kernel parameters
  - Very large problems computationally intractable
  - Batch algorithm
More Summary

Software:

- A list of SVM implementations can be found at http://www.kernel-machines.org/software.html
- Some implementations (such as LIBSVM) can handle multi-class classification
- SVMLight is among the earliest implementations
- Several Matlab toolboxes for SVM are also available

Key points:

- Difference between logistic regression and SVMs
- Maximum margin principle
- Target function for SVMs
- Slack variables for mis-classified points
- Kernel trick allows non-linear generalizations