Support Vector Machines
Lecture 4: Sept 20, 2013

CS886-2 Natural Language Understanding

University of Waterloo

Support Vector Machines

- Find linear separator that maximizes the distance (or margin) to closest data points
- Picture

Notation

- Inputs: $\boldsymbol{x}$ (column vector)
- Output: $y \in \{-1,1\}$ (class)
- Weights: $\boldsymbol{w}$ (vector perpendicular to linear separator)
- Constant: b (displacement of linear separator)
- Index: i (i^{th} instance in dataset)
  - $\boldsymbol{x_i}$ and y_i: input and output of i^{th} instance

Classification

- Linear separator: $\boldsymbol{w}^T\boldsymbol{x} + b = 0$
- Distance to linear separator:

  $\frac{y(\boldsymbol{w}^T\boldsymbol{x}+b)}{||\boldsymbol{w}||}$ where $y \in \{-1,1\}$

- Classification:

  $y = sign(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b)$

  where $sign(expr) = \begin{cases} +1 & \text{if } expr \geq 0 \\ -1 & otherwise \end{cases}$

Margin

- Distance to linear separator:

$$\frac{y(\boldsymbol{w}^T\boldsymbol{x}+b)}{||\boldsymbol{w}||} \quad \text{where } y \in \{-1,1\}$$

- Margin: smallest distance to any data point

$$\frac{1}{||\boldsymbol{w}||}\left\{\min_i y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right\}$$

- Maximum margin:

$$max_{\boldsymbol{w},b}\frac{1}{||\boldsymbol{w}||}\left\{\min_i y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right\}$$

Maximum Margin

- Unique max margin linear separator

$$max_{\boldsymbol{w},b}\frac{1}{||\boldsymbol{w}||}\left\{\min_i y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right\}$$

- Alternatively, we can fix the minimal distance to 1 and minimize $||\boldsymbol{w}||$

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 \quad \forall i$$

- This is a convex quadratic optimization problem that can easily be solved by many optimization packages

Support Vectors

- Quadratic optimization problem

$$\min_{\boldsymbol{w},b} \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2$$
$$\text{s.t. } y_i\left(\boldsymbol{w^T x}_i + b\right) \geq 1 \quad \forall i$$

- Only the points where $y_i\left(\boldsymbol{w^T x_i} + b\right) = 1$ are necessary. These points define the active constraints and are known as the **support vectors**

Linear Separability

- **What if the data is not linearly separable?**
- Two solutions:
  - Map data to new space (known as feature space)
  - Relax constraints with slack variables

Feature Mapping

- Let $\phi(\boldsymbol{x})$ be a mapping that computes the feature representation of a point **x**
- When ϕ is non-linear, it transforms the data to become linearly separable.

Dimensionality

- Feature spaces with high dimensionality
  - Data more likely to be linearly separable
    - If # of dimensions is higher than the number of data points, then the data is necessarily linearly separable
  - Problem: high dimensionality increases computation
- Solution: kernel trick
  - Computation depends on amount of data instead of dimensionality

Kernel Function

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

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. $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(k_1(\boldsymbol{x}, \boldsymbol{x}'))$
  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. $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}'$ $\boldsymbol{A}$ is symmetric positive semi-definite
  9. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x_a}, \boldsymbol{x}'_a) + k_b(\boldsymbol{x_b}, \boldsymbol{x}'_b)$
  10. $k(\boldsymbol{x}, \boldsymbol{x}') = k_a(\boldsymbol{x}_a, \boldsymbol{x}'_a)k_b(\boldsymbol{x}_b, \boldsymbol{x}'_b)$

Common Kernels

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

Common Kernels

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

- Implicit feature space is infinite!

Dual representation

- Idea: reformulation where $\phi(\boldsymbol{x})$ always appears in a kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$
- Approach: find the dual of the optimization problem
- Result: (sparse) kernel support vector machines

$$\max_{\boldsymbol{a}} \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\text{s.t. } \sum_i a_i y_i = 0$$
$$a_i \geq 0 \ \ \forall i$$

Dual derivation

- Transform constrained optimization

$$\min_{\boldsymbol{w},b} \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 \quad \text{s.t. } y_i(\boldsymbol{w}^T\phi(\boldsymbol{x_i}) + b) \geq 1 \quad \forall n$$

  into an unconstrained optimization problem

- Lagrangian

$$\max_{\boldsymbol{a}} \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{a}) \text{ s.t. } \boldsymbol{a} \geq 0$$

where $L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 - \sum_i a_i \underbrace{[y_i(\boldsymbol{w}^T\phi(\boldsymbol{x_i}) + b) - 1]}$

penalty for violating the i[th] constraint

Dual derivation

- Solve inner minimization: $\min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{a})$

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 - \sum_i a_i[y_i(\boldsymbol{w}^T\phi(\boldsymbol{x_i}) + b) - 1]$$

- Set derivatives to 0

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \implies \boldsymbol{w} = \sum_i a_i y_i \phi(\boldsymbol{x}_i)$$

$$\frac{\partial L}{\partial b} = 0 \implies 0 = \sum_i a_i y_i$$

- Eliminate $\boldsymbol{w}$ and b based on these conditions to obtain:

$$L(\boldsymbol{a}) = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Dual Problem

- We are then left with an optimization in a only known as the **dual problem**

$$\max_{\boldsymbol{a}} L(\boldsymbol{a}) = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\text{s.t. } \sum_i a_i y_i = 0$$
$$a_i \geq 0 \ \ \forall i$$

- **Sparse optimization**: many a_i's are 0

Classification

- Primal problem

$$y = sign(\boldsymbol{w}^T \phi(\boldsymbol{x}) + b)$$

- Dual problem

$$y = sign\left(\sum_i a_i y_i \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) + b\right)$$
$$y = sign\left(\sum_i a_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$$

Generalization

- Support vector machines generalize quite well
  - i.e., overfitting is rare
- Reason: maximizing the margin is equivalent to minimizing an upper bound on the worst case loss (worst loss for any underlying input distribution).

Case Study: Text Categorization

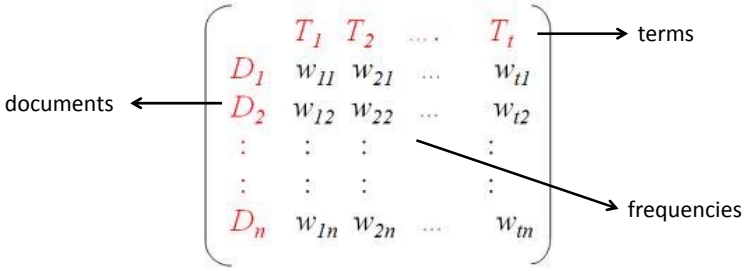
- T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Proceedings of the European Conference on Machine Learning (ECML), Springer, 1998.
- Early success that helped SVMs become popular

Text Categorization

- **Problem:** how to categorize a new article as finance, sports, politics, science, health, etc.?
- **Idea:** train a classifier with archives of news articles that have already been classified

Representation

- How should we represent a document?
- Idea: vector of word counts (vector space model)



Challenges

- **High dimensional input space**:
  - Length of vector is # of words in dictionary (e.g., 10,000)
- **Few irrelevant features**:
  - Most words carry some information that reflect their meaning
- Need an approach that scales well with input dimensionality: **support vector machines**

Experiment

- [Joachim 98]
  - Data: Reuters dataset
  - Compare precision/recall breakeven point
    - i.e., precision = recall
    - Precision: $\frac{|\{relevant\ docs\}|\cap|\{retrieved\ docs\}|}{|\{relevant\ docs\}|}$
    - Recall: $\frac{|\{relevant\ docs\}|\cap|\{retrieved\ docs\}|}{|\{retrieved\ docs\}|}$
  - Algorithms
    - Naïve Bayes: 72.0%
    - Decision trees: 79.4%
    - Rochio: 79.9%
    - K-Nearest Neighbors: 82.3%
    - SVMs: 86.0% (polynomial kernel), 86.4% (Gaussian kernel)

SVM summary

- Find (generalized) linear separator
  - Dual representation (kernel): non-linear separator
- Unique max-margin separator
  - Good generalization
- Convex quadratic optimization
  - Polynomial complexity
  - Global optimality
- Sparse optimization
  - many variables are 0