Kernel Methods

Quang Nguyen
University of Pittsburgh
CS 3750, Fall 2011

Outline

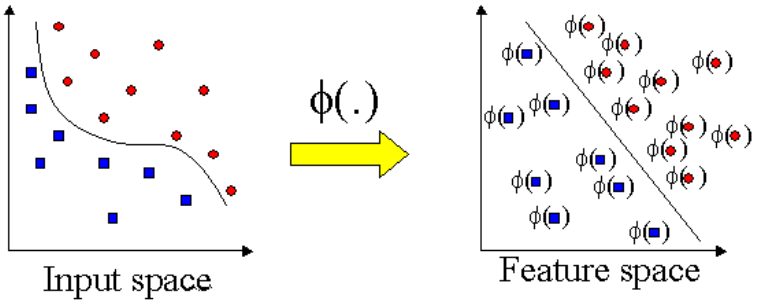
- Motivation
  - Examples
- Kernels
  - Definitions
  - Kernel trick
  - Basic properties
  - Mercer condition
- Constructing feature space
  - Hilbert space
  - Reproducing kernel Hilbert space (RKHS)
- Constructing kernels
- Representer theorem
- Kernel examples
- Choosing feature space and kernel
- Summary

Motivation

- Machine learning theory and algorithms are well developed for linear case
  - Support Vector Machines (SVM)
  - Ridge regression
  - PCA
  - And more
- Real world data analysis problems: often non-linear
- Solution ?

Motivation (cont'd)

- Idea: map data from original input space into a (usually high-dimensional) feature space where linear relations exist among data and apply a linear algorithm in this space



Motivation (cont'd)

- Challenge: computation in high-dimensional space is difficult
- Key idea: if we choose the mapping wisely we can do computation in the feature space implicitly while keep working in the original input space !

Learning

- Given: input/output sets *X, Y*

  $(x_1,y_1) \ldots (x_m,y_m) \in X \times Y$
- Goal: generalization on unseen data
  - Given new input $x \in X$, find the corresponding y
  - (x,y) should be similar to $(x_1,y_1) \ldots (x_m,y_m)$
- Similarity measure
  - For outputs: loss function (e.g. for $Y = \{1,-1\}$, zero-one loss)
  - For inputs: kernel

Kernels

- kernel function k

$$k: X \times X \to R, \qquad (x, x') \mapsto k(x, x')$$

- Kernel is symmetric: $k(x, x') = k(x', x)$
- A kernel that can be constructed by defining a mapping $\varphi: \mathrm{X} \to \mathrm{H}$, from the input space X to a feature space H, such that $\forall\, x, x' \in X$:

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

- Why do we want this?
  - Allow us to apply many ML algorithms in dot product (feature) spaces
  - Gives us freedom to choose φ => design a large variety of models to solve a given problem

Kernel Trick

- We map patterns from input space X into a high-dimensional feature space H and compare them using dot product
- Choose mapping such that the dot product can be evaluated directly using a non-linear function in X

$\Rightarrow$ Avoid computation in H

$\Rightarrow$ Kernel Trick

Kernel Example

- Assume $\mathbf{x} = [x_1, x_2]^T$ and a feature mapping that maps the input in a quadratic feature set

$$\mathbf{x} \rightarrow \varphi(\mathbf{x}) = \left[x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1\right]^T$$

- Kernel function for the feature space:

$$k(\mathbf{x}', \mathbf{x}) = \varphi(\mathbf{x}')^T \varphi(\mathbf{x})$$

$$= x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1x_2x'_1x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1$$

$$= (x_1x'_1 + x_1x'_2 + 1)^2$$

$$= (1 + (\mathbf{x}^T\mathbf{x}'))^2$$

$\Rightarrow$ Computation in the higher dimensional space is performed implicitly in the original input space

Kernel Example: Support Vector Machines

- Solution of the dual problem gives us:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i \, y_i \mathbf{x}_i$$

- The decision boundary:

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i \, y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- The decision:

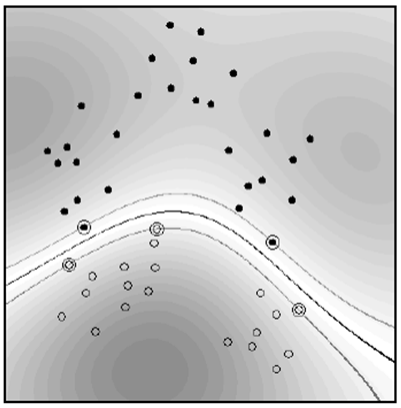
$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i \, y_i (\mathbf{x}_i^T \mathbf{x}) + w_0\right]$$

- Mapping to a feature space, we have the decision:

$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i \, y_i \underbrace{(\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i))}_{\text{kernel k}} + w_0\right]$$

SVM with Gaussian Kernel

$$k(x, x') = \exp(-\|x - x'\|^2)$$



Mercer's Condition

- **Question:** whether a prospective kernel k is good, e.g. being a dot product in some feature space ?
- Mercer's condition (Vapnik 1995): there exists a mapping φ and an expansion

$$k(x, y) = \sum_i \varphi(x)_i \varphi(y)_i$$

<=> $\forall g(x)$ such that L_2 norm $\int g(x)^2 d(x)$ is finite, then

$$\int k(x, y) g(x) g(y) d(x) d(y) \geq 0$$

Positive Definite Kernels

- It can be shown that the admissible class of kernels coincides with the class of positive definite (pd) kernels
- Definition: k: X x X → R is called a pd kernel if
  - k symmetric: k(x,x') = k(x',x)
  - ∀ $x_1,\ldots,x_m \in X$ and ∀ $c_1,\ldots,c_m \in R$

    $\sum_{i,j=1}^{m} c_i c_j K_{ij} \geq 0$, where $K_{ij} = (k(x_i,x_j))_{ij}$

  K is called Gram matrix or kernel matrix

Basic properties of PD kernels

1. Kernels from feature maps

   If φ maps X into a dot product space H then $\langle \varphi(x), \varphi(x') \rangle$ is a pd kernel on X x X
2. Positivity on the diagonal

   $k(x,x) \geq 0 \; \forall x \in X$
3. Cauchy-Schwarz inequality

   $k(x,x')^2 \leq k(x,x)k(x',x')$
4. Vanishing diagonals

   $k(x,x) = 0 \;\; \forall x \in X \rightarrow k(x,x') = 0 \; \forall x, x' \in X$

From Kernels to Feature Spaces

- Question: given a pd kernel in the input space, how can we construct a feature space such that the kernel computes dot product in that space ?
  - i.e. how to construct mapping φ and space *H,* $\varphi: X \to H$, such that $\forall\ x, x' \in X$

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle$$

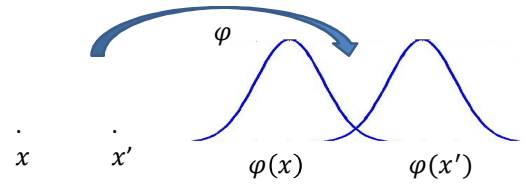
Constructing Feature Space

1. Define a feature map

$$\varphi: X \to R^X, \qquad x \mapsto k(.,x)$$

$\varphi(x) = \mathrm{k}(.,\mathrm{x})$ denotes the function that assigns the value k(x',x) to x' ∈ R

e.g. for the Gaussian kernel



2. Turn it into a linear space

Add linear combinations to the space

$$f(.) = \sum_{i=1}^{m} \alpha_i k(.,x_i), \qquad g(.) = \sum_{j=1}^{m'} \beta_j k(.,x'_j)$$

where $m, m' \in N$, $\alpha_i, \beta_j \in R$ *and* $x_i, x'_j \in X$

Constructing Feature Space (Cont'd)

3. Add dot product to the space

$$\langle f, g\rangle = \sum_{i=1}^{m}\sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j)$$
$$= \sum_{i=1}^{m} \alpha_i g(x_i) \qquad \text{(independent of f)}$$
$$= \sum_{j=1}^{m'} \beta_j f(x'_j) \qquad \text{(independent of g)}$$

$\Rightarrow \langle k(.,x), g\rangle = g(x)$ *and* $\langle f, k(.,x')\rangle = f(x')$

In addition, we have difined $\varphi(x) = \mathrm{k}(.,\mathrm{x})$

$\Rightarrow \langle k(.,x), k(.,x')\rangle = k(x,x') = \langle \varphi(x), \varphi(x')\rangle$

$\Rightarrow$ k is called a reproducing kernel

(Hofman et al. 2008) proved that operator $\langle .,. \rangle$ is in fact a dot product and a pd kernel (symmetric, positive definite by definition)

4. Complete the space with a norm to get a reproducing kernel Hilbert Space (RKHS)

Hilbert Spaces

- Hilbert Space: a complete vector space with dot product and a norm
- Definition: dot product on a vector space
  - A real function <x,y>: V x V → R that ∀ *x*, *y*, *z* ∈*V* and ∀ *c* ∈ R
    - *<x,y> = <y,x>*
    - *<cx,y> = c<x,y>*
    - *<x+y,z> = <x,z> + <y,z>*
    - *<x,x>* > 0 when x ≠ 0
- Complete space
  - All Cauchy sequences $\{x_n\}$ in the space converge: $\forall \varepsilon > 0\ \exists n \in N\colon\ \forall m, k > n\colon\ \|x_m - x_k\| < \varepsilon$
  - Completeness induces (by Riesz repsentation theorem) that ∀x' ∈ X and $\forall f \in H$, ∃ a unique function of x, called k(x,x') s.t.
$$f(x') = \langle f, (k(.,x')\rangle$$

Constructing Kernels

k_1, k_2 are valid (symmetric, positive definite) kernels

$\Rightarrow$ The following are valid kernels:

1. $k(u,v) = \alpha k_1(u,v) + \beta k_2(u,v)$, for $\alpha, \beta \geq 0$
2. $k(u,v) = k_1(u,v)\, k_2(u,v)$
3. $k(u,v) = k_1(f(u),f(v))$, where $f: X \rightarrow X$
4. $k(u,v) = g(u)g(v)$, for $g: X \rightarrow R$
5. $k(u,v) = f(k_1(u,v))$, where f is a polynomial with positive coefficients
6. $k(u,v) = \exp(k_1(u,v))$
7. $k(u,v) = \exp\left(\frac{-\|u-v\|^2}{\sigma^2}\right)$

Representer Theorem

Denote by $\Omega : [0,\infty] \rightarrow R$ a strictly monotonic increasing function, by X a set, and by $c : (X \times R^2)^n \rightarrow R \cup \{\infty\}$ an arbitrary cost function. Then each minimizer $f \in H$ of the regularized risk functional

$$c((x_1, y_1, f(x_1)) \ldots (x_n, y_n, f(x_n)) + \Omega (\|f\|^2_H)$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

Representer Theorem (cont'd)

- Significance: although the optimization problem seems to be in an infinite-dimensional space H, the solution only lies in the span of m particular kernels centered on m training points
  - Note that we need to solve only for $\alpha_i, i = 1..m$

Examples: Kernels on vectors

- Polynomial
  $k(x, x') = (c + \langle x, x' \rangle)^p, \mathrm{p} \in N, c \geq 0$
- Gaussian
  $$k(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$$
- Radial basis
  $$k(x, x') = \exp\left(-\frac{1}{2}\|x - x'\|^2\right)$$

Example: String Kernel

- We want to compare 2 strings, e.g. "distance" between 2 strings
- Given index sequence **I** = (i_1,….$i_{|u|}$) with $1 \leq i_1 < \ldots < i_{|u|} \leq |s|$, define subsequence u of string s: u = s(**I**) = s(i_1)…s($i_{|u|}$)
- $l(\boldsymbol{I})$ = $i_{|u|} - i_1 + 1$ length of subsequence in s
- Feature map: $[\varphi_n(s)]_u = \sum_{\boldsymbol{I}:s(\boldsymbol{I})=u} \lambda^{l(\boldsymbol{I})}$, 0<λ<1 is a decay parameter
- Example: substring u= *asd*, strings s_1 = *N<u>asd</u>aq*, s_2 = *l<u>ass d</u>as*

  ⇒ $[\varphi_n(s_1)]_u = \lambda^3$, $[\varphi_n(s_2)]_u = 2\lambda^5$
- Kernel

$$k_n(s,t) = \sum_u [\varphi_n(s)]_u [\varphi_n(t)]_u = \sum_u \sum_{\boldsymbol{I},\boldsymbol{J}:s(\boldsymbol{I})=t(\boldsymbol{J})=u} \lambda^{l(\boldsymbol{I})} \lambda^{l(\boldsymbol{I})}$$

- Applications: document analysis, spam filtering, annotation of DNA sequences etc

Examples: kernels on other structures

- Tree kernels
- Graph kernels
- Kernels on sets and subspaces
- And more …

How to choose the best feature space

- The problem of choosing the optimal feature space for a given problem is non-trivial
- Since we only know the feature space by the kernel that we use, this problem reduces to choosing the best kernel for learning
- Performance of the kernel algorithm is highly dependent on the kernel
- The best kernel depends on the specific problem

Choosing the best kernel

- We can use prior knowledge about the problem to significantly improve performance
  - Shape of the solution
- If kernel is not appropriate for problem, one needs to tune the kernel (performance is problem-dependent, so no universally good algorithm exists)
- Bad news: we need prior knowledge

  Good news: some knowledge can be extracted from the data

Approaches to choosing the best kernel

- Approach 1: Tune the hyper-parameter of a given family of functions
  - E.g. With the Gaussian kernel $k(x,x') = \exp(-\|x-x'\|^2 / 2\sigma^2)$, set the kernel width σ
- However, for non-vectorial data (e.g. strings), this approach does not work well for popular kernels
  - People have devised their own kernel for problems such as protein classification

Approaches to choosing the best kernel (cont'd)

- Approach 2: Learn the kernel matrix directly from the data
- This approach is more promising
- Goals of this approach
  - Not restricted to one family of kernels that may not be appropriate for the given problem
  - Stop tuning hyper-parameters and instead derive a way to learn the kernel matrix with the setting of parameters

Learning the kernel matrix

- Problems with learning the kernel matrix
  - It is not clear what is the best criterion to optimize
  - Difficult to solve the optimization
  - Choice of the class of kernel matrices to be considered is important
- Implementation issue
  - It may not be possible to store the entire kernel matrix for large data sets

Summary

- Kernels make it possible to look for linear relations in high-dimensional spaces at low computational cost
  - Inner products of the inputs in the feature space can be calculated in the original space
- Can be applied to non-vectorial data
  - Strings, trees, graphs etc
- Finding the best feature space and kernel is non-trivial

References

- Hoffman, Scholkopf, Smola. *Kernel methods in machine learning*. Annals of Statistics, Volume 36, Number 3 (2008), 1171-1220.
- Scholkopf, Smola. *A short introduction to kernel methods*. Advanced lectures on machine learning, LNAI 2600, pp. 41-64, 2003
- Hal Daume III. *From zero to reproducing kernel hilbert spaces in twelve pages or less*. Unpublished
- Vapnik V. *The nature of statistical learning theory*. Springer, New York, 1995
- Barlett P. *Lectures from statistical learning theory course*. Spring 2008
- Some slides/pictures from Milos Hauskrecht and David Krebs