Introduction to Kernel Methods

Dave Krebs
CS 3750
Fall 2007

Sources

- Bierens, Herman J. *Introduction to Hilbert Spaces*. Pennsylvania State University. 24 June 2007.
- Bousquet, Perez-Cruz. *Kernel Methods and Their Potential Use in Signal Processing*. **IEEE Signal Processing Magazine**. May 2004.
- Burges, Christopher. *A Tutorial on Support Vector Machines for Pattern Recognition*.
- Cristianini, Shawe-Taylor, Suanders. *Kernel Methods: A Paradigm for Pattern Analysis*. **Kernel Methods in Bioengineering, Signal and Image Processing**. 2007.
- Schölkopf, Bernhard. *Statistical Learning and Kernel Methods*.
- Schölkopf, Bernhard. *The Kernel Trick for Distances*.

Outline

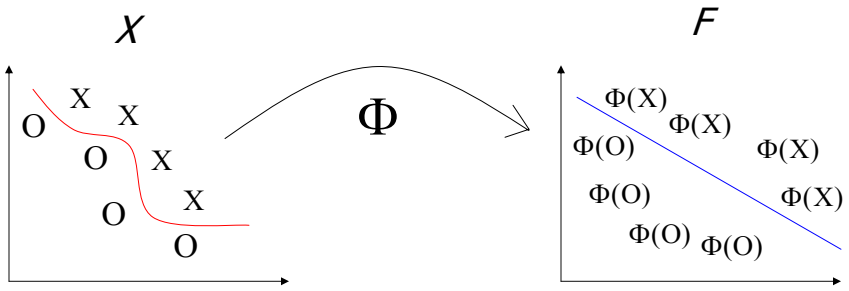
- Motivation
- Kernel Basics
  - Definition
  - Example
  - Application
  - Modularity
  - Creating more complicated kernels
- Mercer's Condition
- Definitions
- Constructing a Feature Space
  - Hilbert Spaces
- Kernels as Generalized Distances
- Gaussian kernel
- Choosing the best feature space

Motivation

- Given a set of vectors, there are many tools available for one to use to detect linear relations among the data
  - Ridge Regression
  - Support Vector Machines (SVM's)
  - Principal Component Analysis (PCA)
- But what if the relations are non-linear in the original space?

Motivation

- Solution: Map the data into a (possibly high-dimensional) vector space where linear relations exist among the data, then apply a linear algorithm in this space



Motivation

- Problem: Representing data in a high-dimensional space is computationally difficult
- Alternative solution to the original problem: Calculate a similarity measure in the feature space instead of the coordinates of the vectors there, then apply algorithms that only need the value of this measure
  - Use dot product as similarity measure

Kernel Definition

- A function that takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space is called a *kernel function*
- More formally, if we have data $\mathbf{x}, \mathbf{z} \in X$ and a map $\phi : X \to \Re^N$ then

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

is a kernel function

An Important Point

- Using kernels, we do not need to embed the data into the space $\Re^N$ explicitly, because a number of algorithms only require the inner products between image vectors!
- We never need the coordinates of the data in the feature space!

Kernel Example

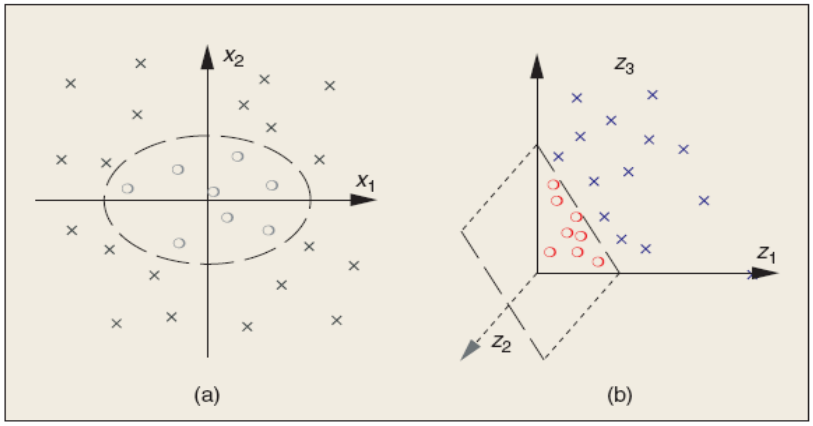
- Consider a two-dimensional input space $X \subseteq \Re^2$ with the feature map:

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \Re^3$$

Now consider the inner product in the feature space:

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \quad = (x_1z_1 + x_2z_2)^2 \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2\end{aligned}$$

Kernel Example (continued)



▲ 1. Effect of the map $\phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. (a) Input space $\mathcal{X}$ and (b) feature space $\mathcal{H}$.

Kernel Example (continued)

- Then $k(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z} \rangle^2$
- But $k(\mathbf{x},\mathbf{z})$ is also the kernel that computes the inner product of the map

$$\psi(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1) \in F = \Re^4$$

- This shows that a given feature space is not unique to a given kernel function

Kernel Application: 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:

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

Modularity

- Basic approach to using kernel methods is:
  - Choose an algorithm that uses only inner products between inputs
  - Combine this algorithm with a kernel function that calculates inner products between input images in a feature space
- Using kernels, algorithm is then implemented in a high-dimensional space
- Another nice property of kernels is *modularity* - The same kernel can be can be reused and adapted to very different real-world problems
  - Kernels can be combined together to form complex learning systems

Creating more complicated kernels

1. $k(x,z) = k_1(x,z) + k_2(x,z)$
2. $k(x,z) = \alpha k_1(x,z)$ where $\alpha \in \Re^+$
3. $k(x,z) = k_1(x,z)k_2(x,z)$
4. $k(x,z) = f(x)f(z)$

where $f(\cdot)$ is a real-valued function on X

Kernel Trick

- We want to map the patterns into a high-dimensional feature space *F* and compare them using a dot product
- To avoid working in the space *F*, choose a feature space in which the dot product can be evaluated directly using a nonlinear function in the input space
- This is called the *kernel trick*

$$k(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

Mercer's Condition

- For which kernels does there exist a pair $\{H, \phi\}$ where *H* is a (possibly infinite dimensional) Euclidean space and ϕ is the mapping $\phi : \Re^d \rightarrow H$ and for which does there not?
- Mercer's condition tells us whether or not a prospective kernel is actually a dot product in some space
- It can be stated as follows:

There exists a mapping ϕ and an expansion

$$K(\mathbf{x},\mathbf{y}) = \sum_i \phi(\mathbf{x})_i \phi(\mathbf{y})_i$$

Mercer's Condition (continued)

if and only if, for any $g(\mathbf{x})$ such that $\int g(\mathbf{x})^2 d\mathbf{x}$ is finite, then $\int K(\mathbf{x},\mathbf{y})g(\mathbf{x})g(\mathbf{y})d\mathbf{x}d\mathbf{y} \geq 0$

- It can be shown that this condition is satisfied for positive integral powers the dot product:
  $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^p$
- Any kernel $K(\mathbf{x},\mathbf{y}) = \sum_{p=0}^{\infty} c_p(\mathbf{x}\cdot\mathbf{y})^p$, where the c_p are positive real coefficients and the series is uniformly convergent, satisfies Mercer's condition

Definitions

- **Gram matrix**
  $$\mathbf{K} \equiv (k(\mathbf{x}_i,\mathbf{x}_j))_{ij}$$
  where k is a kernel, $\mathbf{x}_1,\ldots,\mathbf{x}_m$ are input patterns and $\boldsymbol{K}$ is m x m
- **Positive matrix**
  An m x m matrix $\boldsymbol{K}_{ij}$ satisfying
  $$\sum_{i,j=1}^{m} c_i \bar{c}_j \mathbf{K}_{ij} \geq 0 \quad \text{where } c_i \in C$$

Definitions (continued)

- **Positive Definite (pd) kernel (a.k.a. Mercer kernel, support vector kernel)**
  A function $k : X \text{ x } X \to C$ which for all $m \in I, x_i \in X$ gives rise to a positive Gram matrix
  - This property implies positivity on the diagonal:
    $$k(\mathbf{x}_1, \mathbf{x}_1) \geq 0 \text{ for all } \mathbf{x}_1 \in X$$
- To have only real coefficients c_i, we must require that the kernel be symmetric: $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$

Definitions (continued)

- **Conditionally Positive matrix**
  An *m* x *m* matrix $\boldsymbol{K}_{ij}$ $(m \geq 2)$ satisfying
  $$\sum_{i,j=1}^{m} c_i \bar{c}_j \mathbf{K}_{ij} \geq 0 \quad \text{for all } c_i \in C \text{ with } \sum_{i=1}^{m} c_i = 0$$
- **Conditionally Positive Definite kernel**
  A function $k : X \text{ x } X \to C$ which for all $m \geq 2, x_i \in X$ gives rise to a conditionally positive Gram matrix

Constructing a Feature Space

- Given that we want a kernel function *k* that satisfies $k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$, how do we construct a feature space for *k* ?

- 1. Define a feature map

$$\phi : X \mapsto R^X, \quad x \mapsto k(., x)$$

Then $\phi(\mathbf{x}) = k(., \mathbf{x})$ denotes the function that assigns the value $k(\mathbf{x}', \mathbf{x})$ to $\mathbf{x}' \in X$

Each pattern is now a function on the domain X

Constructing a Feature Space (continued)

- 2. Turn it into a linear space

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

Constructing a Feature Space (continued)

3. Endow it with a dot product

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

And turn it into a Hilbert Space H_k

Note that $\langle k(., x), k(., x') \rangle = k(x, x')$

and so we have the kernel trick $k(x, x') = \langle \phi(x), \phi(x') \rangle$

Hilbert Spaces

- Vector Space
  - A set *V* endowed with the addition operation and the scalar multiplication operation such that these operations satisfy certain rules
  - It is trivial to verify that the Euclidean space is a real vector space
- Inner Product (on a real vector space)
  - A real function $< x, y >: V \times V \rightarrow \Re$ such that for all *x*, *y*, *z* in *V* and all *c* in $\Re$ we have
    - <*x*,*y*> = <*y*,*x*>
    - <*cx*,*y*> = *c*<*x*,*y*>
    - <*x*+*y*,*z*> = <*x*,*z*> + <*y*,*z*>
    - <*x*,*x*> > 0 when $x \neq 0$

Hilbert Spaces (continued)

- Norm (on a vector space *V*)
  - A mapping $\|\cdot\| : V \to [0, \infty)$ such that for all *x* and *y* in *V* and all scalars *c*, we have
    - 1. $\|x\| > 0 \;\; if \;\; x \neq 0$
    - 2. $\|c, x\| = |c| \cdot \|x\|$
    - 3. $\|x + y\| \leq \|x\| + \|y\|$ (Triangular inequality)
  - A norm $\|\cdot\|$ defines a metric $d(x, y) = \|x - y\|$ on *V*, which can be thought of as a function that measures the "distance" between two elements *x* and *y* of *V*

Hilbert Spaces (continued)

- Cauchy Sequence
  - A sequence of elements x_n of a metric space (i.e. a space endowed with a metric) with metric *d*(.,.) such that for every $\varepsilon > 0$ there exists an n_0 such that for all $k, m \geq n_0$ we have $d(x_k, x_m) < \varepsilon$
- Hilbert Space
  - A vector space *H* endowed with an inner product and associated norm and metric such that every Cauchy sequence in *H* has a limit in *H*

Hilbert Spaces (continued)

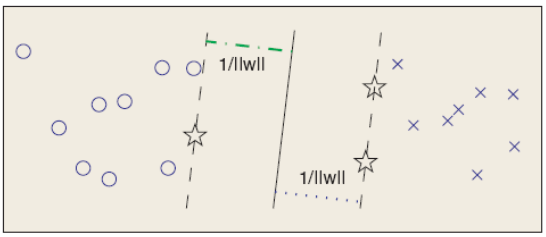
- Example of a Hilbert Space: A Euclidean space $\Re^n$
  - A vector space
  - Has an inner product
    - Dot product $< x, y >= x^T y$
  - Has the norm $\|x\| = \sqrt{x^T x} = \sqrt{< x, x >}$
  - Has metric $\|x - y\|$

When the feature space is larger than the number of training examples

- Given sufficient data, linear algorithms perform well when applied to linear problems
- For nonlinear problems where the dimensionality of the feature space is much larger than the number of input samples (dimensionality can even be infinite), we cannot rely on data to obtain good results
- To find the best solution, we can:
  - Minimize the empirical error
  - Control the complexity

When the feature space is larger than the number of training examples (continued)

- For SVM's, the *maximum margin principle* does this for us
  - We choose the simplest solution (the linear solution with the largest margin) from the set of solutions with the smallest error
- Rationale: We expect a simple solution to perform better on unseen patterns



▲ 2. Maximal margin hyperplane.

A Problem with the Dot Product

- If patterns x and x' are translated by

  $$x \mapsto x - x_0, \quad x' \mapsto x' - x_0$$

  then $(x \cdot x')$ changes
- This is not suitable for algorithms where the learning process should be translation invariant (PCA)
- The dissimilarity measure $\|x - x'\|^2$ is translation invariant and can be expressed using the kernel trick

$$\begin{aligned} \|\varphi(x) - \varphi(x')\|^2 &= \langle \varphi(x), \varphi(x) \rangle - 2\langle \varphi(x), \varphi(x') \rangle + \langle \varphi(x'), \varphi(x') \rangle \\ &= k(x,x) + k(x',x') - 2k(x,x') \end{aligned}$$

Kernels as Generalized Distances

- Since the dot product is not translation invariant, we need to fix an origin x_0 when going from the distance measure to the dot product
- Write the dot product of $(x - x_0)$ and $(x' - x_0)$ in terms of distances:

$$((x - x_0) \cdot (x' - x_0)) = (x \cdot x') + \|x_0\|^2 + (x \cdot x_0) + (x_0 \cdot x')$$
$$= \frac{1}{2}\left[-\|x - x'\|^2 + \|x - x_0\|^2 + \|x_0 - x'\|^2\right]$$

- This kernel trick with $\|\phi(x) - \phi(x')\|^2$ works with a pd kernel *k*, but there exists a larger class of kernels that can be used as generalized distances

Kernels as Generalized Distances (continued)

- Whenever a kernel *k* is pd, it is also cpd
- But cpd property does not imply pd property, so cpd kernels comprise a larger class than pd kernels
  - For example, the kernel $k(x, x') = -\|x - x'\|^2$ is cpd but not pd
    In fact, $k(x, x') = -\|x - x'\|^\beta$ is cpd for $0 < \beta \leq 2$
- Let *k* be a real-valued cpd kernel on X, satisfying $k(x, x) = 0$ for all $x \in X$. Then there exists a Hilbert space *H* of real-valued functions on X, and a mapping $\phi : X \to H$ such that

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

Kernels as Generalized Distances (continued)

- Given the last result, using cpd kernels we can generalize all algorithms based on distances to corresponding algorithms operating in feature spaces
- We thus have a kernel trick for distances

The Gaussian Kernel

- Defined as $k(x,z) = \exp\left(\frac{-\|x-z\|^2}{2\sigma^2}\right)$ for $\sigma > 0$
- The most widely used kernel
- Is **universal** – linear combinations of this kernel can approximate any continuous function
  - Corresponding feature space is infinite dimensional
  - Given any labeled data set there exists a linear hyperplane which correctly separates the data in the Gaussian feature space
  - This makes the expression power basically unlimited

Be careful not to overfit!

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 (continued)

- Approach 2: Learn the kernel matrix directly from the data
- This approach is more promising
- Goals of this approach
  - Do not be 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 of the kernel matrix:
  - It is not yet clear what is the best criterion to optimize
  - Current procedures are not computationally efficient
  - Choice of the class of kernel matrices to be considered is important

Implementation Issues

- Selection of a kernel for a given problem can be difficult
- It may not be possible to store the entire kernel matrix for large data sets
  - May need to re-compute kernel function as entries are needed

Advantages of Kernels

- Make it possible to look for linear relations in very high-dimensional spaces at very low computational cost
  - This is because the inner products of the input images in the feature space can be calculated in the original space
- Modularity
- Does not require data to be real vectors
  - For example, can work on strings, time series

Questions?