Kernel Methods

Sargur Srihari
srihari@buffalo.edu

Topics in Kernel Methods

1. Linear Models vs Memory-based models
2. Stored Sample Methods
3. Kernel Functions
   - Dual Representations
   - Constructing Kernels
4. Extension to Symbolic Inputs
5. Fisher Kernel

Linear Models vs Memory-based models

- Linear parametric models for regression and classification have the form $y(\boldsymbol{x}, \boldsymbol{w})$
  - During learning we either get a maximum likelihood estimate of $\boldsymbol{w}$ or a posterior distribution of $\boldsymbol{w}$
  - Training data is then discarded
  - Prediction based only on vector $\boldsymbol{w}$
  - This is true of Neural networks as well
- Memory-based methods keep the training samples (or a subset) and use them during the prediction phase

Memory-Based Methods

- Training data points are used in prediction
- Examples of such methods
  - Parzen probability density model
    - Linear combination of kernel functions centered on each training data point
  - Nearest neighbor classification
- These are memory-based methods
- They require a metric to be defined
  - Fast to train, slow to predict

Kernel Functions

- Linear models can be re-cast
  - into equivalent dual where predictions are based on kernel functions evaluated at training points
- Kernel function is given by

$$k\,(\boldsymbol{x},\boldsymbol{x}') = \phi(\boldsymbol{x})^{\mathrm{T}}\,\phi(\boldsymbol{x}')$$

  - where $\phi(\boldsymbol{x})$ is a fixed nonlinear mapping (basis function)
  - Kernel is a symmetric function of its arguments

$$k\,(\boldsymbol{x},\boldsymbol{x}') = k\,(\boldsymbol{x}',\boldsymbol{x})$$

  - Kernel can be interpreted as similarity of $\boldsymbol{x}$ and $\boldsymbol{x}'$
    - Simplest is identity mapping in feature space $\phi(\boldsymbol{x}) = \boldsymbol{x}$
    - In which case $k\,(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'$
      - Called Linear Kernel

Kernel Trick

- Formulated as inner product allows extending well-known algorithms
  - by using the kernel trick
- Basic idea of kernel trick
  - If an input vector $\boldsymbol{x}$ appears only in the form of scalar products then we can replace scalar products with some other choice of kernel
- Used widely
  - in support vector machines
  - in developing non-linear variant of PCA
  - In kernel Fisher discriminant

Other Forms of Kernel Functions

- Function of difference between arguments

$$k\,(\boldsymbol{x},\boldsymbol{x}') = k\,(\boldsymbol{x}-\boldsymbol{x}')$$

  - Called *stationary* kernel since invariant to translation

- Radial basis functions

$$k\,(\boldsymbol{x},\boldsymbol{x}') = k\,(||\boldsymbol{x}-\boldsymbol{x}'||)$$

  - Depends on magnitude of distance between arguments
    - Note that the kernel function is scalar while x is M-dimensional

For these to be valid kernel functions they should be shown to have the property
$k\,(\mathrm{x},\mathrm{x}') = \varphi\,(\mathrm{x})^{\mathrm{T}}\,\varphi(\mathrm{x}')$

Dual Representation

- Linear models for regression/classification can be reformulated in terms of dual representation
  - In which kernel function arises naturally
  - Plays important role in SVMs
- Consider linear regression model
  - Parameters from minimizing *sum-of-squares*

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

where $\mathbf{w} = (w_0,..,w_{M-1})^T$, $\phi = (\phi_0,..\phi_{M-1})^T$

we have N samples $\{\mathbf{x}_1,..\mathbf{x}_N\}$

λ is the regularization coefficient

ϕ is the set of M basis functions or feature vector

- Minimum obtained by setting gradient of $J(\mathbf{w})$ wrt $\mathbf{w}$ equal to zero

Solution for $\boldsymbol{w}$: linear combination of $\varphi(\boldsymbol{x}_n)$

- Equating derivative $J(\boldsymbol{w})$ wrt $\boldsymbol{w}$ to zero we get

$$\begin{aligned} \mathrm{w} &= -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathrm{w}^T\phi(\mathrm{x}_n) - t_n\right\}\phi(\mathrm{x}_n) \\ &= \sum_{n=1}^{N} a_n\phi(\mathrm{x}_n) \\ &= \Phi^T\mathrm{a} \end{aligned}$$

  - Solution for $\boldsymbol{w}$: linear combination of $\varphi(\boldsymbol{x}_n)$ whose coefficients are functions of $\boldsymbol{w}$ where
    - Φ is design matrix whose n^{th} row is given by $\varphi(\boldsymbol{x}_n)^{\mathrm{T}}$

$$\Phi = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_{M-1}(x_1) \\ \vdots & & \vdots \\ \phi_0(x_n) & \cdots & \phi_{M-1}(x_n) \\ \vdots & & \vdots \\ \phi_0(x_N) & \cdots & \phi_{M-1}(x_N) \end{bmatrix} \text{ is a } N \times M \text{ matrix}$$

    - Vector $\boldsymbol{a}=(a_1,..,a_N)^{\mathrm{T}}$ has the definition

$$a_n = -\frac{1}{\lambda}\left\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\right\}$$

Transformation from $\boldsymbol{w}$ to $\boldsymbol{a}$

- Thus we have $\boldsymbol{w} = \Phi^T \boldsymbol{a}$
- Instead of working with parameter vector $\boldsymbol{w}$ we can reformulate least squares algorithm in terms of parameter vector $\boldsymbol{a}$
  - giving rise to dual representation

$$a_n = -\frac{1}{\lambda}\left\{\boldsymbol{w}^T\phi(\boldsymbol{x}_n) - t_n\right\}$$

- Although the definition of $\boldsymbol{a}$ still includes $\boldsymbol{w}$

  It can be eliminated by the use of the kernel function

Gram Matrix and Kernel Function

- Gram matrix $K=\Phi\Phi^{\mathrm{T}}$ is N x N matrix
  - with elements

$$K_{nm}=\phi(\boldsymbol{x}_n)^{\mathrm{T}}\phi(\boldsymbol{x}_m)=k(\boldsymbol{x}_n,\boldsymbol{x}_m)$$

  - where kernel function $k(\boldsymbol{x},\boldsymbol{x}')=\phi(\boldsymbol{x})^{\mathrm{T}}\phi(\boldsymbol{x}')$

$$K=\begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & . & . & k(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ . & & & \\ . & & & \\ k(\boldsymbol{x}_N,\boldsymbol{x}_1) & & & k(\boldsymbol{x}_N,\boldsymbol{x}_{N1}) \end{bmatrix}$$

Gram Matrix Definition:
Given N vectors, it is the matrix of all inner products

  - Notes:
    - Φ is NxM and K is NxN Note: $N \times M$ times $M \times N$
    - K is a matrix of similarities of pairs of samples (thus it is symmetric)

Error Function in Terms of Gram Matrix

- Error Function is

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\boldsymbol{w}^T\phi(\mathrm{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

- Substituting $\boldsymbol{w} = \Phi^{\mathrm{T}}\boldsymbol{a}$ into $J(\boldsymbol{w})$ gives

$$J(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}^T\Phi\Phi^T\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a}$$

where $\mathrm{t} = (t_1,..,t_N)^{\mathrm{T}}$

- Error function is written in terms of Gram matrix as

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T KK\boldsymbol{a} - \boldsymbol{a}^T K\mathrm{t} + \frac{1}{2}\mathrm{t}^T\mathrm{t} + \frac{\lambda}{2}\boldsymbol{a}^T K\boldsymbol{a}$$

- Solving for $\boldsymbol{a}$ by combining $\boldsymbol{w}=\Phi^{\mathrm{T}}\boldsymbol{a}$ and $a_n = -\frac{1}{\lambda}\left\{\mathrm{w}^T\phi(\mathrm{x}_n) - t_n\right\}$

$$\boldsymbol{a} = (K + \lambda \mathrm{I}_{\mathrm{N}})^{-1}\mathrm{t}$$

Solution for $\boldsymbol{a}$ can be expressed as a linear combination of elements of $\phi(\boldsymbol{x})$ whose coefficients are entirely in terms of kernel $k(\boldsymbol{x},\boldsymbol{x}')$ from which we can recover original formulation in terms of parameters $\boldsymbol{w}$

Prediction Function

- Prediction for new input $\boldsymbol{x}$
  - We can write $\boldsymbol{a} = (K + \lambda \mathrm{I}_N)^{-1}\mathbf{t}$ by combining $\boldsymbol{w} = \Phi^{\mathrm{T}}\boldsymbol{a}$ and

$$a_n = -\frac{1}{\lambda}\left\{\mathrm{w}^T\boldsymbol{\phi}(\mathrm{x}_n) - t_n\right\}$$

  - Substituting back into linear regression model,

$$\begin{aligned} y(\boldsymbol{x}) &= \mathrm{w}^T\boldsymbol{\phi}(\boldsymbol{x}) \\ &= \boldsymbol{a}^T\Phi\boldsymbol{\phi}(\boldsymbol{x}) \\ &= k(\boldsymbol{x})^T(K + \lambda I_N)^{-1}\mathrm{t} \quad \text{where } k(\boldsymbol{x}) \text{ has elements } k_n(\boldsymbol{x}) = k(\boldsymbol{x}_n, \boldsymbol{x}) \end{aligned}$$

- Prediction is a linear combination of the target values from the training set.

Advantage of Dual Representation

- Solution for $\boldsymbol{a}$ is entirely in terms of $k(\boldsymbol{x},\boldsymbol{x}')$
  - From $\boldsymbol{a}$ we can recover $\boldsymbol{w}$ using $\boldsymbol{w}=\Phi^{\mathrm{t}}\boldsymbol{a}$
- In parametric formulation, solution is $\boldsymbol{w}_{ML}=(\Phi^{T}\Phi)^{-1}\Phi^{T}\mathrm{t}$
  - We invert M x M matrix, since Φ is N x M
- In dual formulation, solution is $\boldsymbol{a}=(K+\lambda \mathrm{I}_{\mathrm{N}})^{-1}\mathbf{t}$
  - We are inverting an N x N matrix
  - An apparent disadvantage, since $N>>M$
  - Advantage of dual: work with kernel $k(\boldsymbol{x},\boldsymbol{x}')$, so:
    - avoid working with a feature vector $\phi(\boldsymbol{x})$ and
    - problems associated with very high or infinite dimensionality of $\boldsymbol{x}$

Constructing Kernels

- To exploit kernel substitution need valid kernels
  - Two methods:
    - (1) from $\phi(\boldsymbol{x})$ to $k(\boldsymbol{x},\boldsymbol{x}')$
    - (2) from $k(\boldsymbol{x},\boldsymbol{x}')$ to $\phi(\boldsymbol{x})$
- First Method
  - Choose $\phi(\boldsymbol{x})$ and use it to find corresponding kernel

$$k(\boldsymbol{x},\boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') = \sum_{i=1}^{M} \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{x}')$$

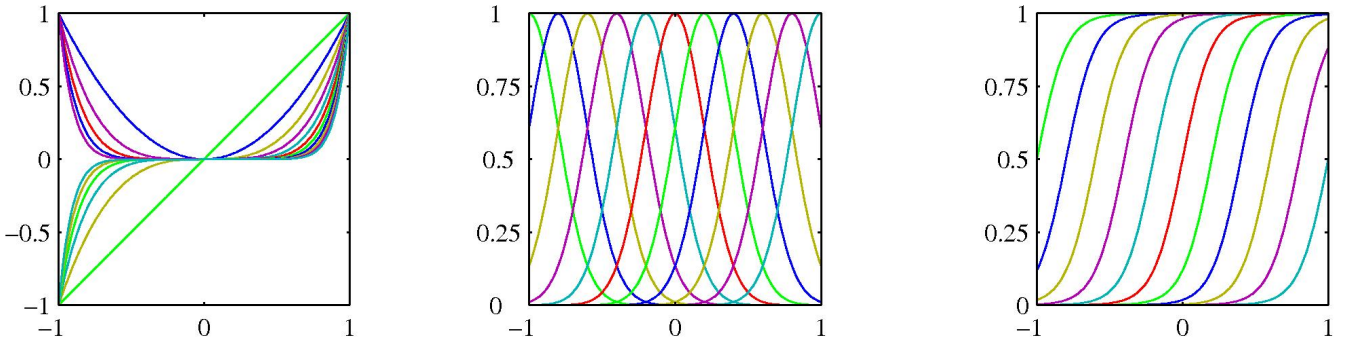
    - where $\phi_i(\boldsymbol{x})$ are basis functions
  - One-dimensional input space is shown next

Kernel Functions from basis functions

One-dimensional input space

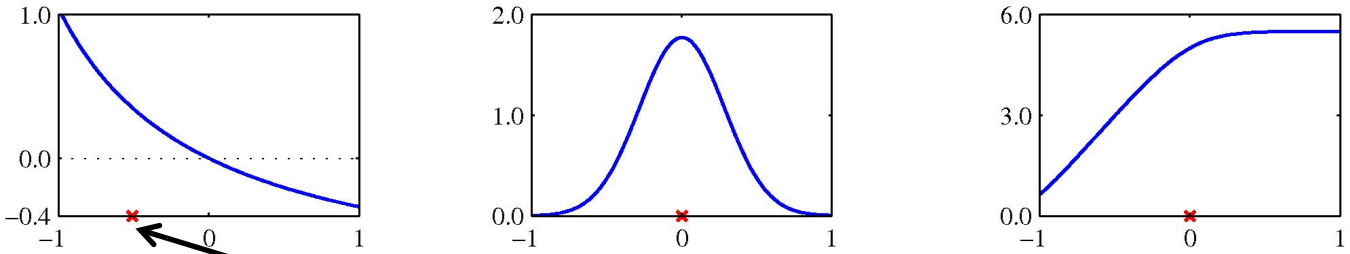
Polynomials Gaussian Logistic Sigmoid

Basis Functions $\phi_i(x)$



Kernel Functions

$k(x,x') = \phi(x)^{\mathrm{T}}\phi(x')$



Red cross is x'

Method 2: Direct Construction of Kernels

- Kernel has to correspond to a scalar product in some (perhaps infinite dimensional) space
  - Consider kernel function $k(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^2$
    - In 2-D space
$$k(\boldsymbol{x},\boldsymbol{z}) = (\boldsymbol{x}^T\boldsymbol{z})^2 = (x_1z_1 + x_2z_2)^2$$
$$= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$
$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2)^T$$
$$= \phi(\boldsymbol{x})^T\phi(\boldsymbol{z})$$
    - Feature mapping takes the form $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$
      - Comprises of all second order terms with a specific weighting
      - Inner product needs computing 6 feature values, 9 multiplications
    - Kernel function $k(\boldsymbol{x},\boldsymbol{z})$ has 2 multiplies and a squaring
    - For $(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z}+\mathrm{c})^2$ we get constant, linear, second order terms
    - For $(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z}+\mathrm{c})^{\mathrm{M}}$ we get all powers of $\boldsymbol{x}$ (monomials)

Testing whether a function is a valid kernel

- Without constructing $\phi(\boldsymbol{x})$, Necessary and sufficient condition for $k(\boldsymbol{x},\boldsymbol{x}')$ to be a kernel is
  - Gram matrix K, with elements $k(\boldsymbol{x}_n,\boldsymbol{x}_m)$ is positive $\{\boldsymbol{x}_n\}$semi-definite for all possible choices of the set
  - Positive semi-definite is not the same thing as a matrix whose elements are non-negative
    - It means

$$\mathrm{z}^T K\mathrm{z} \geq 0 \text{ for non - zero vectors z with real entries}$$
$$\text{i.e., } \sum_n \sum_m K_{nm} z_n z_m \geq 0 \text{ for any real numbers } z_n, z_m$$

    - Mercer's theorem: any continuous, symmetric, positive semi-definite kernel function $k(\boldsymbol{x},\ \boldsymbol{y})$ can be expressed as a dot product in a high-dimensional space
- New kernels can be constructed from simpler kernels as building blocks

Techniques for Constructing Kernels

- Given valid kernels $k_1(\boldsymbol{x},\boldsymbol{x}')$ and $k_2(\boldsymbol{x},\boldsymbol{x}')$ the following new kernels will be valid

1. $k(\boldsymbol{x},\boldsymbol{x}') = ck_1(\boldsymbol{x},\boldsymbol{x}')$
2. $k(\boldsymbol{x},\boldsymbol{x}')=f(\boldsymbol{x})k_1(\boldsymbol{x},\boldsymbol{x}')f(\boldsymbol{x}')$
3. $k(\boldsymbol{x},\boldsymbol{x}')=q(k_1(\boldsymbol{x},\boldsymbol{x}'))$
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. $k(\boldsymbol{x},\boldsymbol{x}')=k_3(f(\boldsymbol{x}).f(\boldsymbol{x}'))$
8. $k(\boldsymbol{x},\boldsymbol{x}')=\boldsymbol{x}^{\mathrm{T}}\mathrm{A}\boldsymbol{x}'$
9. $k(\boldsymbol{x},\boldsymbol{x}')=k_a(\boldsymbol{x}_a,\boldsymbol{x}_b')+k_b(\boldsymbol{x}_b,\boldsymbol{x}_b')$
10. $k(\boldsymbol{x},\boldsymbol{x}')=k_a(\boldsymbol{x}_a,\boldsymbol{x}_a')k_b(\boldsymbol{x}_b,\boldsymbol{x}_b')$

Where

$f(.)$ is any function

$q(.)$ is a polynomial with non-negative coefficients

$f(\boldsymbol{x})$ is a function from x to R^{M}

k_3 is a valid kernel in R^{M}

A is a symmetric positive semidefinite matrix

$\boldsymbol{x}_{\mathrm{a}}$ and $\boldsymbol{x}_{\mathrm{b}}$ are variables with $\boldsymbol{x}=(\boldsymbol{x}_{\mathrm{a}}, \boldsymbol{x}_{\mathrm{b}})$

k_{a} and k_{b} are valid kernel functions

Kernels for specific applications

- Requirements for $k(\boldsymbol{x}, \boldsymbol{x}')$
  - It is symmetric
  - Its Gram matrix is positive semidefinite
  - It expresses the appropriate similarity between $\boldsymbol{x}$ and $\boldsymbol{x}'$ for the intended application

Gaussian Kernel

- Commonly used kernel is

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-||\boldsymbol{x}-\boldsymbol{x}'||^2/2\sigma^2\right)$$

  - It is seen as a valid kernel by expanding square

$$||\boldsymbol{x}-\boldsymbol{x}'||^2 = \boldsymbol{x}^T\boldsymbol{x} + (\boldsymbol{x}')^T\boldsymbol{x}' - 2\boldsymbol{x}^T\boldsymbol{x}'$$

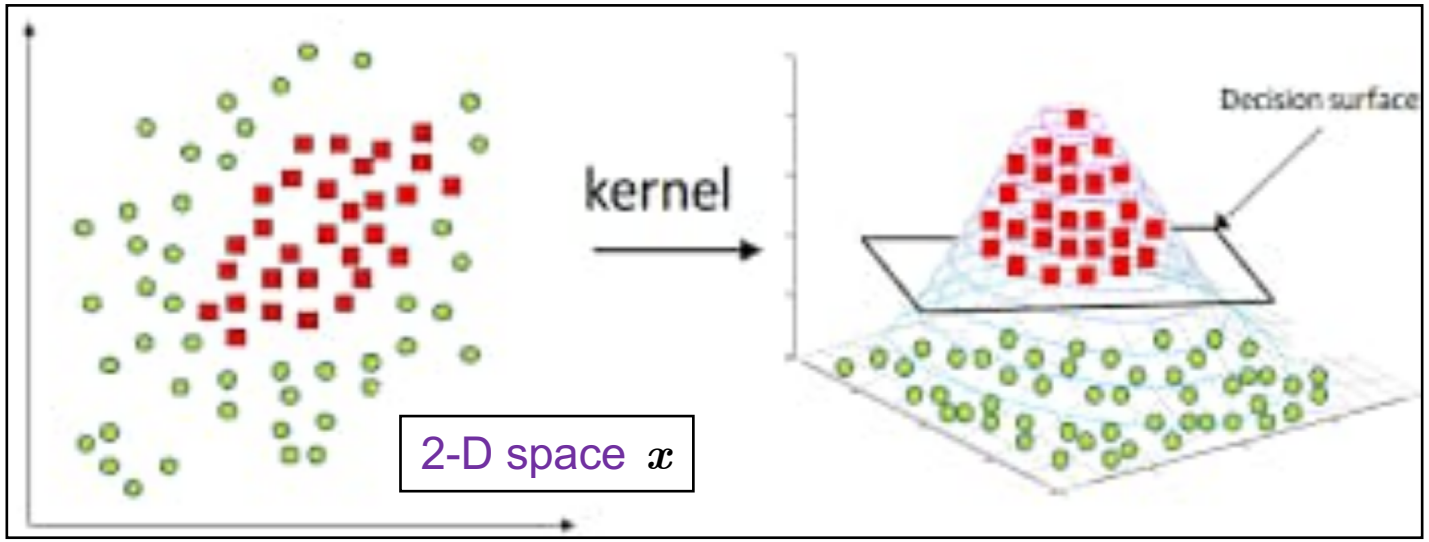
    - To give

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\boldsymbol{x}^T\boldsymbol{x}/2\sigma^2\right)\exp\left(-\boldsymbol{x}^T\boldsymbol{x}'/\sigma^2\right)\exp\left(-(\boldsymbol{x}')^T\boldsymbol{x}'/2\sigma^2\right)$$

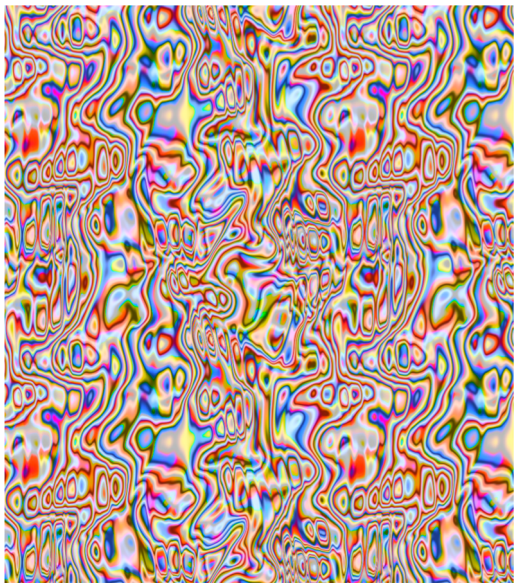
  - From kernel construction rules 2 and 4
    - together with validity of linear kernel $k(\boldsymbol{x},\boldsymbol{x}')=\boldsymbol{x}^T\boldsymbol{x}'$
- Can be extended to non-Euclidean distances

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left\{(-1/2\sigma^2)[\kappa(\boldsymbol{x},\boldsymbol{x}')+\kappa(\boldsymbol{x}',\boldsymbol{x}')-2\kappa(\boldsymbol{x},\boldsymbol{x}')]\right\}$$

Use of a kernel method



3-D space $\phi(x)$



Infinite dimensional space $\phi(x)$
Induced by a Gaussian kernel

Oil slick 2D- image
that looks like a feature space x

Extension of Kernels to Symbolic Inputs

- Important contribution of kernel viewpoint:
  - Inputs are symbolic, not vectors of real numbers
    - Kernels defined for graphs, sets, strings, text documents
- If A_1 and A_2 are two subsets of objects
  - A simple kernel is $\boxed{k(A_1, A_2) = 2^{|A_1 \cap A_2|}}$
    - where $|A|$ indicates no of elements in A
    - A valid kernel since it can be shown to correspond to an inner product in a feature space

$A=\{1,2,3,4,5\}$

$A_1=\{2,3,4,5\}$, $A_2=\{1,2,4,5\}$, $A_1 \cap A_2=\{2,4,5\}$
Hence $k(A_1,A_2)=8$

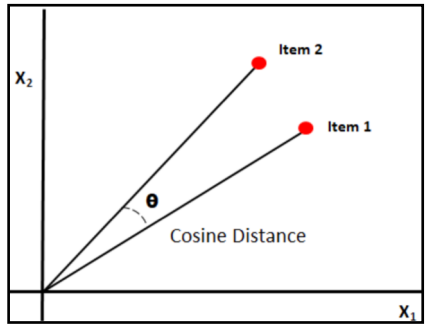
Example feature vectors:
$\phi(A_1)=[2\ 2\ 2\ 2]$ and $\phi(A_2)=[1\ 1\ 1\ 1]$
such that
$\phi(A_1)^T\phi(A_2)=8$

Kernels for Complex Objects

- ML methods studied so far require input represented as a fixed-size feature vector $\boldsymbol{x}_i \varepsilon \, \mathrm{R}^D$
  - For certain objects it is not clear how best to represent them as feature vectors, E.g.,
    - Text or protein sequence of variable length
    - Molecular structure with complex 3-D geometry
    - Evolutionary tree which has variable size and shape
- Solutions
  - Define generative model, whose parameters are features. Plug these features into standard models
  - Measure similarity between objects which does not require preprocessing into feature vector format

Kernels for Comparing Documents

- In IR and document classification, need a way of comparing documents $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$
- Bag of words representation: x_{ij} is the no of times word j occurs in document i $\boxed{\boldsymbol{x}_i=[x_{i1},..,x_{iD}]}$



  - Cosine similarity

$$k(\boldsymbol{x}_i,\boldsymbol{x}_{i'})=\frac{\boldsymbol{x}_i^T\boldsymbol{x}_{i'}}{||\boldsymbol{x}_i||_2||\boldsymbol{x}_{i'}||_2}$$

    - It measures cosine of angle between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$
    - Since $\boldsymbol{x}_i$ is a count vector (and hence non-negative) the cosine similarity is between 0 and 1
      - Where 0 means the vectors are orthogonal and therefore have no words in common

Disadvantage of Cosine Similarity

- It does not work well for two reasons

1. Stop words

- If $\boldsymbol{x}_i$ has any word in common with $\boldsymbol{x}_i$, it is deemed similar, even though some words such as the, and occur commonly in many documents and are not discriminative

2. High frequency words in a document

- If a discriminatory word occurs many times in a document, the similarity is artificially boosted
  - Even though word usage would be bursty, i.e., once a word is used in a document it is very likely to be used again

TF-IDF

- Cosine similarity performance can be improved with some preprocessing
- Use feature vector called TF-IDF representation
  - Term frequency-Inverse document frequency
    - TF is a log-transform of the count $\text{tf}(x_{ij}) \triangleq \log(1+x_{ij})$
    - IDF is defined as $\text{idf}(j) \triangleq \log \frac{N}{1+\sum_{i=1}^{N} I(x_{ij}>0)}$
      x_{ij} is the no of times word j occurs in document i
      - Where N is the total no of documents
      - Denominator counts how many documents contain term j
    - Finally we define $\text{tf-idf}(xi) \triangleq [\text{tf}(x_{ij}) \times \text{idf}(j)]_{j=1}^{V}$
- We use this inside cosine similarity measure

$$k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_{i'})}{||\phi(\boldsymbol{x}_i)||_2 ||\phi(\boldsymbol{x}_{i'})||_2}$$

  - Where $\phi(\boldsymbol{x})$=tf-idf($\boldsymbol{x}$)

String Kernels

- Real power of kernels: inputs are structured
- Compare two variable-length strings
  - Strings $\boldsymbol{x}$, and $\boldsymbol{x}'$ of length D, D' defined over A
    - E.g., two amino acid sequences defined over 20-letter alphabet $\mathrm{A}=\{\mathrm{A},\mathrm{R},\mathrm{N},\mathrm{D},\mathrm{C},\mathrm{E},\mathrm{Q},\mathrm{G},\mathrm{H},\mathrm{I},\mathrm{L},\mathrm{K},\mathrm{M},\mathrm{F},\mathrm{P},\mathrm{S},\mathrm{T},\mathrm{W},\mathrm{Y},\mathrm{V}\}$
    - $\boldsymbol{x}$ is the sequence of length 110
      IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKN……VNQFLDYLQEFLGVMNTEWI
    - $\boldsymbol{x}'$ is a string of length 153
    - PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA……..
    - The strings have the substring LQE in common
    - We can define the similarity of two strings to be the no of substrings they have in common
    - Definition given next

Mercer Kernel

- If s is a substring of x we can write $x=usv$ for some (possibly empty) strings u, s and v
- $\phi_s(x)$ is no of times substring s appears in x
- Define kernel between two strings x and x' as

$$k(x,x') = \sum_{s \in A^*} w_s \phi_s(x) \phi_s(x')$$

  - Where $w_s \geq 0$ and A^* is the set of all strings from alphabet A (known as Kleene * operator)
  - Can be computed in $O(|x|+|x'|)$ time
- Cases of interest:
  - $w_s=0$ for $|s|>1$ we get bag-of-characters kernel– defines $\phi(x)$ as no of times each character in A occurs in x
  - If we require s to be bordered by white space we get bag-of- words kernel

Combining Discriminative and Generative Models

- Generative models deal naturally with missing data and with HMM of varying length
- Discriminative models such as SVM have better performance
- Can use a generative model to define a kernel and use kernel in discriminative approach

Kernels based on Generative Models

- For a model $p(\boldsymbol{x})$ we define a kernel by

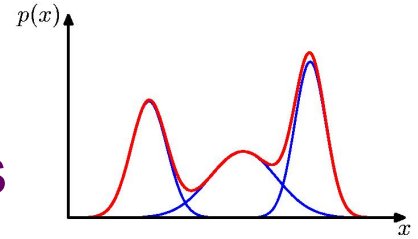
$$k\,(\boldsymbol{x},\boldsymbol{x}') = p(\boldsymbol{x})\;p(\boldsymbol{x}')$$

  - A valid kernel since an inner product in 1-dimensional feature space defined by the mapping $p(\boldsymbol{x})$
- Two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ are similar if they have high probabilities

Kernel Functions based on Mixture Densities

- Extension to sums/products of distributions

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_i p(\boldsymbol{x} \mid i)p(\boldsymbol{x}' \mid i)p(i)$$



  - where $p(i)$ are positive weighting coefficients
  - A valid kernel based on two rules of kernel construction:

$k(\boldsymbol{x},\boldsymbol{x}') = ck_1(\boldsymbol{x},\boldsymbol{x}')$ and $k(\boldsymbol{x},\boldsymbol{x}')=k_1(\boldsymbol{x},\boldsymbol{x}')+k_2(\boldsymbol{x},\boldsymbol{x}')$

  - Two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ will give a large value of k, and hence appear similar, if they have a significant probability under a range of different components

- Taking the limit to infinite sum

$$k(\boldsymbol{x},\boldsymbol{x}') = \int p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{x}' \mid \boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$$

  - where $\boldsymbol{z}$ is a continuous latent variable

Kernels for Sequences

- Data consists of ordered sequences of length L

$$\mathrm{X}=\{x_1,..,x_L\}$$

- Generative model for sequences is HMM
  - Hidden states $\mathrm{Z}=\{\mathrm{z}_1,..,\mathrm{z}_\mathrm{L}\}$
- Kernel Function for measuring similarity of sequences X and X' is

$$k(\mathrm{X},\mathrm{X}')=\sum_{Z} p(\mathrm{X}\,|\,\mathrm{Z})p(\mathrm{X}'\,|\,\mathrm{Z}')p(\mathrm{Z})$$

  - Both observed sequences are generated by same hidden sequence Z

Fisher Kernel

- Alternative technique for generative models
  - In document retrieval, protein sequences
- Consider parametric generative model $p(\boldsymbol{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes vector of parameters
  - Goal: find kernel that measures similarity of two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ induced by the generative model
- Define Fisher score as gradient wrt θ

$$\mathrm{g}(\theta,\mathrm{x}) = \nabla_\theta \ln p(\mathrm{x}\,|\,\theta)$$

A vector of same dimensionality as θ

- Fisher Kernel is $k(\mathrm{x},\mathrm{x}') = \mathrm{g}(\theta,\mathrm{x})^T \mathrm{F}^{-1} \mathrm{g}(\theta,\mathrm{x}')$

$$\mathrm{F} = \mathrm{E}_\mathrm{x}\left[\mathrm{g}(\theta,\mathrm{x})\mathrm{g}(\theta,\mathrm{x})^T\right]$$

Fisher score is more generally the gradient of the log-likelihood

where F is the Fisher information matrix

Fisher Information Matrix

- Presence of Fisher information matrix causes kernel to be invariant under non-linear parametrization of the density model $\theta \rightarrow \psi(\theta)$
  - In practice, infeasible to evaluate Fisher Information Matrix. Instead use the approximation

$$\mathrm{F} \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{g}(\theta,\mathrm{x}_n)\mathrm{g}(\theta,\mathrm{x}_n)^T \qquad k(\mathrm{x},\mathrm{x}') = \mathrm{g}(\theta,\mathrm{x})^T \mathrm{F}^{-1}\mathrm{g}(\theta,\mathrm{x}')$$

  - This is the covariance matrix of the Fisher scores
    - So the Fisher kernel corresponds to whitening of the Fisher scores $k(\mathrm{x},\mathrm{x}') = \mathrm{g}(\theta,\mathrm{x})^T \mathrm{g}(\theta,\mathrm{x}')$
  - More simply omit F and use non-invariant kernel

Sigmoidal Kernel

- A link between SVMs and neural network

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \tanh\left(a\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + b\right)$$

  - Its Gram matrix is not positive semidefinite
  - But used in practice because it gives SVMs a superficial resemblance to neural networks
- Bayesian neural network with an appropriate prior reduces to a Gaussian process
  - Provides a deeper link between neural networks and kernel methods