Introduction to Machine Learning

Kernel Methods

Varun Chandola

Computer Science & Engineering
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu

University at Buffalo
Department of Computer Science and Engineering
School of Engeering and Applied Sciences

Outline

Can Regression be Adapted to Use a Kernel?

- Ridge regression estimate:

$$\mathbf{w} = (\lambda I_D + \mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

- Prediction at $\mathbf{x}^*$:

$$y^* = \mathbf{w}^\top \mathbf{x}^* = ((\lambda I_D + \mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y})^\top \mathbf{x}^*$$

- Still needs training and test examples as D length vectors
- Rearranging above (Sherman-Morrison-Woodbury formula or *Matrix Inversion Lemma* [See Murphy p120, Matrix Cookbook])

$$y^* = \mathbf{y}^\top (\lambda I_N + \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{x}^*$$

Using the Dot Product

$$y^* = \mathbf{y}^\top(\lambda \mathbf{I}_N + \boxed{\mathbf{X}\mathbf{X}^\top})^{-1}\boxed{\mathbf{X}\mathbf{x}^*}$$

$\mathbf{X}\mathbf{X}^\top$?

$$\mathbf{X}\mathbf{X}^\top = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_N \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_N \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}_N, \mathbf{x}_1 \rangle & \langle \mathbf{x}_N, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_N, \mathbf{x}_N \rangle \end{pmatrix}$$

Using the Dot Product

$$y^* = \mathbf{y}^\top(\lambda \mathbf{I}_N + \boxed{\mathbf{X}\mathbf{X}^\top})^{-1}\boxed{\mathbf{X}\mathbf{x}^*}$$

$\mathbf{X}\mathbf{X}^\top$?

$$\mathbf{X}\mathbf{X}^\top = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_N \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_N \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}_N, \mathbf{x}_1 \rangle & \langle \mathbf{x}_N, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_N, \mathbf{x}_N \rangle \end{pmatrix}$$

$\mathbf{X}\mathbf{x}^*$?

$$\mathbf{X}\mathbf{x}^* = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}^* \rangle \\ \langle \mathbf{x}_2, \mathbf{x}^* \rangle \\ \vdots \\ \langle \mathbf{x}_N, \mathbf{x}^* \rangle \end{pmatrix}$$

Generalizing to Non-linear Regression

- Consider a set of P functions that can be applied on input example $\mathbf{x}$

$$\boldsymbol{\phi} = \{\phi_1, \phi_2, \ldots, \phi_P\}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_P(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_P(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_P(\mathbf{x}_N) \end{pmatrix}$$

- Prediction:

$$y^* = \mathbf{y}^\top (\lambda \mathbf{I}_N + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1} \boldsymbol{\Phi} \phi(\mathbf{x}^*)$$

- Each entry in $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ is $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$

The Great Kernel Trick

- Replace dot product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with a function $k(\mathbf{x}_i, \mathbf{x}_j)$
- Replace $\mathbf{X}\mathbf{X}^\top$ with $\mathbf{K}$

$$K[i][j] = k(\mathbf{x}_i, \mathbf{x}_j)$$

- $\mathbf{K}$ - *Gram Matrix*
- k - kernel function
  - Similarity between two data objects

Kernel Regression

$$y^* = \mathbf{y}^\top (\lambda \mathbf{I}_N + \mathbf{K})^{-1} k(\mathbf{X}, \mathbf{x}^*)$$

How to Construct a Kernel?

- Already know the simplest kernel function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$$

Approach 1: Start with basis functions

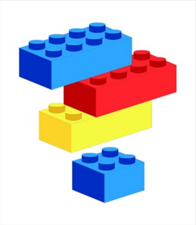
$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

Approach 2: Direct design (good for non-vector inputs)

- Measure **similarity** between $\mathbf{x}_i$ and $\mathbf{x}_j$
- Should follow *Mercer's Condition*
  - **Kernel/Gram matrix must be positive semi-definite**
- k should be *symmetric*

Using Building Blocks

$$
\begin{aligned}
k(\mathbf{x}_i, \mathbf{x}_j) &= ck_1(\mathbf{x}_i, \mathbf{x}_j) \\
k(\mathbf{x}_i, \mathbf{x}_j) &= f(\mathbf{x})k_1(\mathbf{x}_i, \mathbf{x}_j)f(\mathbf{x}_j) \\
k(\mathbf{x}_i, \mathbf{x}_j) &= q(k_1(\mathbf{x}_i, \mathbf{x}_j)) \ q \text{ is a polynomial} \\
k(\mathbf{x}_i, \mathbf{x}_j) &= exp(k_1(\mathbf{x}_i, \mathbf{x}_j)) \\
k(\mathbf{x}_i, \mathbf{x}_j) &= k_1(\mathbf{x}_i, \mathbf{x}_j) + k_2(\mathbf{x}_i, \mathbf{x}_j) \\
k(\mathbf{x}_i, \mathbf{x}_j) &= k_1(\mathbf{x}_i, \mathbf{x}_j)k_2(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned}
$$



Popular Kernels

- **Radial Basis Function** or **Gaussian Kernel**

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{1}{2\gamma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

- **Cosine Similarity**

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}$$

The RBF Kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{1}{2\gamma^2}||\mathbf{x}_i - \mathbf{x}_j||^2\right)$$

- Mapping inputs to an infinite dimensional space

Probabilistic Kernel Functions

- Allows using generative distributions in discriminative settings
- Uses class-independent probability distribution for input $\mathbf{x}$

$$k(\mathbf{x}_i, \mathbf{x}_j) = p(\mathbf{x}_i|\boldsymbol{\theta})p(\mathbf{x}_j|\boldsymbol{\theta})$$

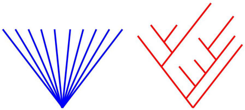
- Two inputs are more similar if both have high probabilities

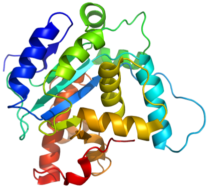
Bayesian Kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \int p(\mathbf{x}_i|\boldsymbol{\theta})p(\mathbf{x}_j|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$$

Regression for Non-Vector Data Examples

- What if $\mathbf{x} \notin \Re^D$?
- Does $\mathbf{w}^\top \mathbf{x}$ make sense?
- How to adapt?
  1. Extract features from $\mathbf{x}$
  2. Is not always possible

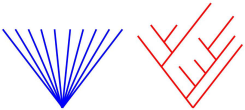


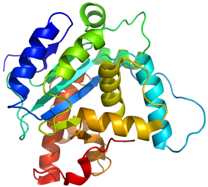


When, in the course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the laws of nature and of nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

We hold these truths to be self-evident, that all men are created equal, that they are endowed by their Creator with certain unalienable rights, that among these are life, liberty and the pursuit of happiness.

Regression for Non-Vector Data Examples





- What if $\mathbf{x} \notin \Re^D$?
- Does $\mathbf{w}^\top \mathbf{x}$ make sense?
- How to adapt?
  1. Extract features from $\mathbf{x}$
  2. Is not always possible
- Sometimes it is easier/natural to compare two objects.
  - A similarity function or **kernel**

When, in the course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the laws of nature and of nature's God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

We hold these truths to be self-evident, that all men are created equal, that they are endowed by their Creator with certain unalienable rights, that among these are life, liberty and the pursuit of happiness.

A Similarity Kernel

- Domain-defined measure of similarity

Example

Strings: Length of longest common subsequence, inverse of edit distance

Example

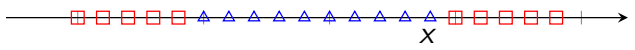
Multi-attribute Categorical Vectors: Number of matching values

Kernels for Non-vector Data

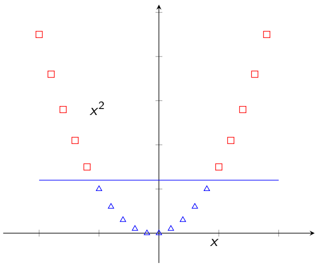
- **String Kernel**
- **Pyramid Kernels**

Why Use Kernels?

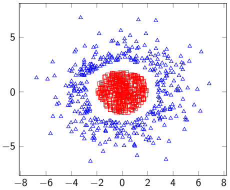
- $x \in \Re$
- No linear separator



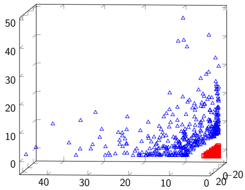
- Map $x \rightarrow \{x, x^2\}$
- Separable in 2D space



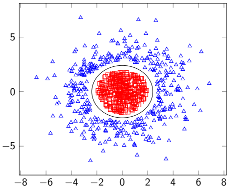
Another Example



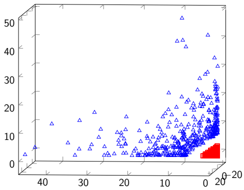
- $\mathbf{x} \in \Re^2$
- No linear separator
- Map $\mathbf{x} \rightarrow \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
- A circle as the decision boundary



Another Example



- $\mathbf{x} \in \Re^2$
- No linear separator
- Map $\mathbf{x} \rightarrow \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$
- A circle as the decision boundary



The RBF or Gaussian Kernel

- The *squared dot product* kernel ($\mathbf{x_i}, \mathbf{x_j} \in \Re^2$):

$$k(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^\top \mathbf{x_j} \triangleq \phi(\mathbf{x_i})^\top \phi(\mathbf{x_j})$$

$$\phi(\mathbf{x_i}) = \{x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2\}$$

- What about the Gaussian kernel (radial basis function)?

$$k(\mathbf{x_i}, \mathbf{x_j}) = exp\left(-\frac{1}{2\gamma^2}||\mathbf{x}_i - \mathbf{x}_j||^2\right)$$

Why is the RBF or Gaussian Kernel Mapping to Infinite Dimensions

- Assume $\gamma = 1$ and $\mathbf{x} \in \Re$ (denoted as x)

$$
\begin{aligned}
k(x_i, x_j) &= exp(-x_i^2)exp(-x_j^2)exp(2x_ix_j) \\
&= exp(-x_i^2)exp(-x_j^2)\sum_{k=0}^{\infty}\frac{2^k x_i^k x_j^k}{k!} \\
&= \sum_{k=0}^{\infty}\left(\frac{2^{k/2}}{\sqrt{k!}}x_i^k exp(-x_i^2)\right)\left(\frac{2^{k/2}}{\sqrt{k!}}x_j^k exp(-x_j^2)\right)
\end{aligned}
$$

- *Using Maclaurin Series Expansion*

$$
k(x_i, x_j) = \begin{pmatrix} 1 \\ 2^{1/2}x_i^1 exp(-x_i^2) \\ \frac{2^{2/2}}{2}x_i^2 exp(-x_i^2) \\ \vdots \end{pmatrix} \times \begin{pmatrix} 1 \\ 2^{1/2}x_j^1 exp(-x_j^2) \\ \frac{2^{2/2}}{2}x_j^2 exp(-x_j^2) \\ \vdots \end{pmatrix}^{\top}
$$

Kernel Machines

- We can use kernel function to *generate* new features
- Evaluate kernel function for each input and a set of K centroids

$$\phi(\mathbf{x}) = [k(\mathbf{x}, \boldsymbol{\mu}_1), k(\mathbf{x}, \boldsymbol{\mu}_2), \ldots, k(\mathbf{x}, \boldsymbol{\mu}_K)]$$

$$y = \mathbf{w}^\top \phi(\mathbf{x}), \quad y \sim Ber(\mathbf{w}^\top \phi(\mathbf{x}))$$

- If k is a Gaussian kernel $\Rightarrow$ Radial Basis Function Network (RBF)
- How to choose $\boldsymbol{\mu}_i$?
  - Clustering
  - Random selection

Generalizing RBF

- Another option: Use every input example as a "centroid"

$$\phi(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), \ldots, k(\mathbf{x}, \mathbf{x}_N)]$$

References