Introduction to Kernel Methods

David Rosenberg

New York University

February 21, 2017

Setup and Motivation

The Input Space $\mathcal{X}$

- Our general learning theory setup: no assumptions about $\mathcal{X}$
- But $\mathcal{X} = \mathbf{R}^d$ for the specific methods we've developed:
  - Ridge regression
  - Lasso regression
  - Support Vector Machines
  - Perceptrons
- Our hypothesis space for these was all affine functions on $\mathbf{R}^d$:

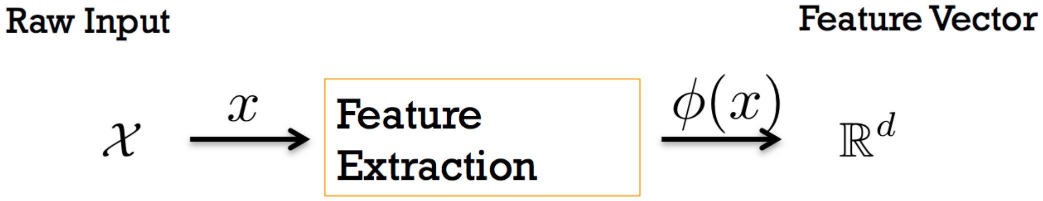
$$\mathcal{H} = \left\{ x \mapsto w^T x + b \mid w \in \mathbf{R}^d, b \in \mathbf{R} \right\}.$$

- What if we want to do prediction on inputs not natively in $\mathbf{R}^d$?

Feature Extraction

Definition

Mapping an input from $\mathcal{X}$ to a vector in $\mathbf{R}^d$ is called **feature extraction** or **featurization**.



- e.g. Quadratic feature map: $\mathcal{X} = \mathbf{R}^d$

$$\phi(x) = \left(x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d\right)^T.$$

Linear Models with Explicit Feature Map

- Rather than take $\mathcal{X} = \mathbf{R}^d$, let $\mathcal{X}$ be its own thing:
- Input space: $\mathcal{X}$
- Introduce **feature map** $\psi : \mathcal{X} \to \mathbf{R}^d$
- The feature map maps into the **feature space** $\mathbf{R}^d$.
- Hypothesis space of affine functions on feature space:

$$\mathcal{H} = \left\{ x \mapsto w^T \psi(x) + b \mid w \in \mathbf{R}^d, b \in \mathbf{R} \right\}.$$

Linear Models Need Big Feature Spaces

- To get **expressive** hypothesis spaces using linear models,
    - need high-dimensional feature spaces
    - (What do we mean by expressive?)
- Very large feature spaces have two problems:
    1. Overfitting
    2. Memory and computational costs
- Overfitting we handle with regularization.
- Kernel methods can (sometimes) help with memory and computational costs.

Some Methods Can Be "Kernelized"

Definition

A method is **kernelized** if inputs only appear inside inner products: $\langle \psi(x), \psi(y) \rangle$ for $x, y \in \mathcal{X}$.

- The **kernel function** corresponding to ψ and inner product $\langle \cdot, \cdot \rangle$ is

$$k(x,y) = \langle \psi(x), \psi(y) \rangle .$$

- Why introduce this new notation $k(x,y)$?
- Turns out, we can often evaluate $k(x,y)$ directly,
  - without explicilty computing $\psi(x)$ and $\psi(y)$.
- For large feature spaces, can be much faster.

Kernel Evaluation Can Be Fast

Example

Quadratic feature map

$$\phi(x) = \left(x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d\right)^T$$

has dimension $O(d^2)$, but

$$k(w,x) = \langle \phi(w), \phi(x) \rangle = \langle w, x \rangle + \langle w, x \rangle^2$$

- Naively explicit computation of $k(w,x)$: $O(d^2)$
- Implicit computation of $k(w,x)$: $O(d)$

Kernels as Similarity Scores

- Can think of the kernel function as a **similarity score**.
- But this is not precise.
- There are many ways to design a similarity score.
    - A kernel function is special because it's an inner product.
    - Has many mathematical benefits.

What's the Benefit of Kernelization?

1. Computational (e.g. when feature space dimension d larger than sample size n).
2. Access to infinite-dimensional feature spaces.
3. Allows thinking in terms of "similarity" rather than features.

Example: SVM

SVM Dual

- Recall the SVM dual optimization problem

$$
\begin{aligned}
\sup_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& \alpha_i \in \left[0, \frac{c}{n}\right] \quad i = 1, \ldots, n.
\end{aligned}
$$

- Notice: x's only show up as inner products with other x's.
- Can replace $x_j^T x_i$ by an arbitrary kernel $k(x_j, x_i)$.
- What kernel are we currently using?

The Kernel Matrix (or the Gram Matrix)

Definition

For a set of $\{x_1, \dots, x_n\}$ and an inner product $\langle \cdot, \cdot \rangle$ on the set, the **kernel matrix** or the **Gram matrix** is defined as

$$K = \left(\langle x_i, x_j \rangle \right)_{i,j} = \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \cdots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix}.$$

Then for the standard Euclidean inner product $\langle x_i, x_j \rangle = x_i^T x_j$, we have

$$K = XX^T$$

SVM Dual with Kernel Matrix

$$\begin{aligned}
\sup_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ji} \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& \alpha_i \in \left[0, \frac{c}{n}\right] \quad i = 1, \ldots, n.
\end{aligned}$$

- Once our algorithm works with kernel matrices, we can change kernel just by changing the matrix.
- Size of matrix: $n \times n$, where n is the number of data points.
- Recall with ridge regression, we worked with $X^T X$, which is $d \times d$, where d is feature space dimension.

Some Kernels

Linear Kernel

- Input space: $\mathcal{X} = \mathbf{R}^d$
- Feature space: $\mathcal{H} = \mathbf{R}^d$, with standard inner product
- Feature map
$$\psi(x) = x.$$
- Kernel:
$$k(w,x) = w^T x$$

Quadratic Kernel in $\mathbf{R}^2$

- Input space: $\mathcal{X} = \mathbf{R}^2$
- Feature space: $\mathcal{H} = \mathbf{R}^5$
- Feature map:

$$\psi : (x_1, x_2) \mapsto \left(x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$

- Gives us ability to represent conic section boundaries.
- Define kernel as inner product in feature space:

$$\begin{aligned} k(w,x) &= \langle \psi(w), \psi(x) \rangle \\ &= w_1x_1 + w_2x_2 + w_1^2x_1^2 + w_2^2x_2^2 + 2w_1w_2x_1x_2 \\ &= w_1x_1 + w_2x_2 + (w_1x_1)^2 + (w_2x_2)^2 + 2(w_1x_1)(w_2x_2) \\ &= \langle w, x \rangle + \langle w, x \rangle^2 \end{aligned}$$

Based on Guillaume Obozinski's Statistical Machine Learning course at Louvain, Feb 2014.

Quadratic Kernel in $\mathbf{R}^d$

- Input space $\mathcal{X} = \mathbf{R}^d$
- Feature space: $\mathcal{H} = \mathbf{R}^D$, where $D = d + \binom{d}{2} \approx d^2/2$.
- Feature map:

$$\phi(x) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d)^T$$

- Still have

$$\begin{aligned} k(w,x) &= \langle \phi(w), \phi(x) \rangle \\ &= \langle x, y \rangle + \langle x, y \rangle^2 \end{aligned}$$

- Computation for inner product with explicit mapping: $O(d^2)$
- Computation for implicit kernel calculation: $O(d)$.

Based on Guillaume Obozinski's Statistical Machine Learning course at Louvain, Feb 2014.

Polynomial Kernel in $\mathbf{R}^d$

- Input space $\mathcal{X} = \mathbf{R}^d$
- Kernel function:

$$k(w,x) = (1 + \langle w,x \rangle)^M$$

- Corresponds to a feature map with all terms up to degree M.
- For any M, computing the kernel has same computational cost
- Cost of explicit inner product computation grows rapidly in M.

Radial Basis Function (RBF) / Gaussian Kernel

- Input space $\mathcal{X} = \mathbf{R}^d$

$$k(w,x) = \exp\left(-\frac{\|w-x\|^2}{2\sigma^2}\right),$$

  where σ^2 is known as the bandwidth parameter.
- Does it act like a similarity score?
- Why "radial"?
- Have we departed from our "inner product of feature vector" recipe?
  - Yes and no: corresponds to an infinite dimensional feature vector
- Probably the most common nonlinear kernel.

Kernel Trick: Overview

Recap

1. Given a kernelized ML algorithm.
2. Can swap out the inner product for a new kernel function.
3. New kernel may correspond to a high dimensional feature space.
4. Once kernel matrix is computed, computational cost depends on number of data points, rather than the dimension of feature space.

Swapping out a linear kernel for a new kernel is called the **kernel trick**.