Statistical Machine Learning and Kernel Methods II
MADALGO Summer School, Aug 2014

Mikhail Belkin, Ohio State University,
Department of Computer Science and Engineering,
Department of Statistics

Summary of Lecture I

Bayes optimal = minimal risk solution is theoretically best. Requires infinite data.

Since only finite data are available, minimize the loss over a space of functions as a proxy for the Bayes optimal -- Empirical Risk Minimization framework.

- Can replace binary loss with convex loss without losing optimality.
- Need a flexible space $\mathcal{H}$ of functions to select from to avoid under/overfitting.
- Assume: 1. linear structure and inner product on $\mathcal{H}$. 2, Functions, which are "close" should provide similar predictions. 1 and 2 imply RKHS structure.
- RKHS correspond to Mercer (PSD) Kernel. Many simple non-linear kernels available.
- Can control fit by choosing ball size in RKHS. Bigger norm – wigglier (more flexible) function. (Often more convenient to add norm as a regularizer.) Control complexity, not number of parameters.
- closed form kernel functions + convex loss = efficient algorithms for ERM.

Today: "the feature map" interpretation and large scale learning.

Some references

- Aronszajn, **Theory of Reproducing Kernels.**
- Wahba, **Spline Models for Observational Data.**
- Vapnik, **Statistical Learning Theory.**
- Schölkopf, Smola, **Learning with Kernels**
- Evgeniou, Pontil, Poggio, **Regularization Networks and Support Vector Machines**
- Smale, **On the Mathematical Foundations of Learning**
- Christianini, Shawe-Taylor, **An Introduction to Support Vector Machines**

Theoretical “analysis”

$$f^* = argmin_{f \in \mathcal{H}} \sum_i l(f(x_i), y_i) + \lambda \|f\|_H^2$$

“Theorem” I (Generalization bound)

$$\left| \sum_i l(f^*(x_i), y_i) - E_{X \times Y} l(f^*(x), y) \right| < \frac{1}{\lambda \sqrt{n}}$$

“Theorem” II (Approximation bound)

$$\left| E_{X \times Y} l(f^*(x), y) - E_{X \times Y} l\left(f^{optimal}(x), y\right) \right| < h(\lambda, n)$$

How efficient?

Data: $(\mathrm{x_i}, \mathrm{y_i}), \mathrm{i} = 1 \ldots \mathrm{n}, \mathrm{x_i} \in \mathbb{R}^{\mathrm{d}}$

Kernel Least Squares:

$$f^* = argmin_{f \in \mathcal{H}} \sum_i (f(x_i) - y_i)^2 + \lambda \|f\|^2$$

Solution: $f^*(x) = \sum_i \alpha_i K(x_i, x)$

For least squares explicit solution: $\boldsymbol{\alpha} = (K + \lambda \mathrm{I})^{-1} \boldsymbol{y}$,where $\mathrm{K_{ij}} = K(x_i, x_j)$

Training complexity $\sim n^2 d$ (n equations with n variables)

Testing complexity: $\sim n\mathrm{d}$

Compare to linear:
training $\sim \mathrm{n}d^2$
testing $\sim d$

Kernel SVM:

$$f^* = argmin_{f \in \mathcal{H}} \sum_i \left(1 - y_i f(x_i)\right)_+ + \lambda \|f\|^2$$

A little trickier to analyze but similar complexity.

How to construct kernels "for free"

Given any "feature" map to any Hilbert space $\phi: Z \to \mathcal{H}$,

can construct a PSD kernel $K(x,y):=\langle \phi(x), \phi(y) \rangle_H$

Why PSD?

Equivalently create a new dataset: $x_i \to \phi(x_i)$ and taking a linear kernel there.
Run SVM, LS in the "feature space".

Feature map interpretation of kernels

Conversely every kernel has a feature map.

For every Mercer (PSD) kernel $K(x, y)$ on Z
there exists a Hilbert space $\mathcal{H}$ and map $\phi: Z \to \mathcal{H}$,
s.t. $\mathrm{K(x,y)}=\langle \phi(x), \phi(y) \rangle_H$

Explicitly: $\phi: \mathrm{x} \to \mathrm{K(x,\cdot)} \in \mathcal{H}_K$ Feature map not unique!

Linear kernels.

Linear kernel is "the" kernel?

Three notes of caution:

1. if $\mathcal{H}$ is rich (high/infinite dimensional) computing ϕ explicitly can be very expensive.

2. How to define the right ϕ? Not always clear.

3. Linear space intuition sometimes fails in infinite dimension.

E.g., there is no uniform probability distribution over an infinite-dimensional ball,

(balls in infinite dimensional linear spaces are not compact)

Another view and another feature map

Let μ be a measure on Z.

$$L_2(\mu) = \left\{ f \; s.t. \int |f|^2 d\mu < \infty \right\}$$

Integral operator $L_K : L_2(\mu) \rightarrow L_2(\mu)$

$$L_K(f)(x) = \int f(y) K(x,y) d\mu_y$$

Corresponding eigensystem $(e_i, \lambda_i), i = 1,2, \ldots$

$$L_K(e_i) = \lambda_i e_i$$

(cf. Fourier)

RKHS – another view

Type equation here.

Any function in $f \in L_2(\mu)$ can be written as

$f(x) = \sum_1^\infty a_i e_i(x)$ where $\sum_1^\infty |a_i|^2 < \infty$

Any function in RKHS $f \in \mathcal{H}$ can be written as

$f(x) = \sum_1^\infty a_i e_i(x)$ where $\sum_1^\infty \frac{|a_i|^2}{\lambda_i} < \infty$

Although (e_i, λ_i), depend on μ, the space $\mathcal{H}$ does not!

Feature map – another view

Type equation here.

Feature map: $\phi: Z \rightarrow l_2$

$$\phi(x) = \left(\sqrt{\lambda_1}\, e_1(x), \sqrt{\lambda_2}\, e_2(x), \ldots\right)$$

$$\langle \phi(x), \phi(y) \rangle_{l_2} = \sum \lambda_i \, e_i(x) e_i(y) = K(x, y)$$

Spectral theorem (c.f. spectral decomposition for matrices).

Infinitely many equivalent "feature maps".

Efficient inference

Idea: given a kernel K(x,y) let's be clever with a feature map.

Find a map $\phi: Z \to V$, such that K(x,y) $\approx \langle \phi(x), \phi(y) \rangle_V$ and V has "small" dimension. Can then run linear algorithms in V.

Two approaches:

1. Nystrom approximation
2. Random Fourier features

Nystrom approximation

$$\phi(x) = \left(\sqrt{\lambda_1}\, e_1(x), \sqrt{\lambda_2}\, e_2(x), \ldots\right)$$

$$\lambda_i e_i = \int e_i(y) K(x,y) d\mu_y$$

Idea:

approximate from a discrete subsample $x_1, \ldots, x_k$ from μ

(can subsample our data)

$$\int f(y) K(x,y) d\mu_y \approx \frac{1}{k} \sum_1^k f(x_i) K(x_i, y)$$

Integral equation becomes standard matrix eigenproblem: $\boldsymbol{K e}_i = \lambda_i \boldsymbol{e}_i$

Can approximate $e_{\mathrm{i}}(\mathrm{x}) \approx \frac{1}{k\lambda_i} \sum_{j=1}^k (\boldsymbol{e}_i)_j K(x_j, x)$

[William, Seeger, 2001, a lot of other related recent work]

Nystrom approximation algorithm

Algorithm summary:

1. Subsample k data points $x_1, \dots, x_k$ at random. Let $\boldsymbol{K}$ be the $k \times \mathrm{k}$ kernel matrix.

2. Find d eigenvectors of $\boldsymbol{K}\boldsymbol{e}_i = \lambda_i \boldsymbol{e}_i$,

3. Construct embedding to $\mathbb{R}^d$:
$\phi(x): x \to \left(\frac{1}{k\sqrt{\lambda}_1}\sum_{j=1}^{k}(\boldsymbol{e}_1)_j K(x_j, x) \dots, \frac{1}{k\sqrt{\lambda}_d}\sum_{j=1}^{k}(\boldsymbol{e}_d)_j K(x_j, x)\right)$

4. Solve ordinary least squares/SVM in the embedding space.

Random Fourier Features

Let $\mathrm{K}(\mathrm{x}, \mathrm{y}) = K(x - y)$ be a positive definite radial kernel.

Bochner's theorem: $K(x - y)$ is a Fourier transform of a non-negative function (measure) p.

$$K(x - y) = \int p(w) e^{-2\pi i w(x-y)} dw$$

We can view p as probability density after scaling.

If we have a sample $w, \dots, w_k$ from p,

$$\int p(w) e^{-2\pi i w(x-y)} dw \approx \frac{1}{k} \sum_{j=1}^{k} e^{-2\pi i w_j (x-y)} = \frac{1}{k} \sum_{j=1}^{k} e^{-2\pi i w_j x} e^{2\pi i w_j y}$$

Feature map: $\phi(x): x \to \left(e^{-2\pi i w_1 x}, \dots, e^{-2\pi i w_k x}\right)$

[Rahimi, Recht 2007]

Big(ger) Data

Non-linear kernel methods (on a typical machine)

~10000 data points is easy (on a typical machine)

~100000 points (doable)

Linear methods

~10000000 is doable

But Caveat Emptor: dimension becomes a key constraint.

Higher dimension – better approximation.

In many practical problems dimension needs to be high 100000 – 1000000 to get very good results. Can be done, but not trivial.

Summary

Kernel methods:

A flexible space of functions suitable for a variety of inferential tasks.

(We only discussed classification/regression, but there is much more)

The framework can be derived from a few natural assumptions.

Feature map "tricks" can be used for scaling to large data sizes (but beware of the dimension).