October 26, 2017	Lecture 14, Oct. 23, 2017

Kernel Methods

Lecturer: Justin Domke *Scribe: Lakshmi Vikraman*

1 Summary

Today we will continue the discussion on Kernel Methods.

2 Kernel Methods

2.1 Kernel Ridge Regression

Last class we talked about Ridge Regression with regularization.

$$\sum_{i=1}^{N}(y_i - \beta^T x_i)^2 + \lambda\|\beta\|_2^2$$

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

Here $X \in R^{N\times p}$ and $X^T X \in R^{p\times p}$

The time complexity for learning is $Np^2 + p^3$ and that for predicting a new example is p. Please note that β can be computed for all λ in same time by computing SVD of $X^T X$.

2.1.1 Kernel Trick

Let us employ the following trick.

$$(PQ + I)^{-1}P = P(QP + I)^{-1}$$

where $P \in R^{T\times S}$ and $Q \in R^{S\times T}$.
As you can see, this changes the dimensionality of the inverse. We can rewrite the equation for β using this trick as shown below. Here $(X^T X + \lambda I)^{-1} \in R^{p\times p}$ and $(XX^T + \lambda I)^{-1} \in R^{N\times N}$.

$$(X^T X + \lambda I)^{-1} X^T = X^T (XX^T + \lambda I)^{-1}$$

$$\beta = X^T (XX^T + \lambda I)^{-1} Y$$

If we introduce $\alpha = (XX^T + \lambda I)^{-1} Y$, then $\beta = X^T \alpha$.

What is the significance of this trick?

Let $K = XX^T$ where $XX^T \in R^{N\times N}$
then $K_{ij} = X_i^T X_j$
and $\alpha = (K + \lambda I)^{-1} Y$

If we can do fast inner product

- we can get K quickly.
- we can get α in N^3 time.

For a new test point x,

$$\begin{aligned} f(x) &= \beta^T x \\ &= (X^T \alpha)^T x \\ &= \alpha^T X x \\ &= \sum_{i=1}^{N} \alpha_i X_i^T x \end{aligned}$$

Here we only calculate inner products and never β !!

2.1.2 Old Way vs New Way of Learning

Let us contrast the old way vs the new way of learning.
Old Way
For input $(x_1, y_1), ...(x_N, y_N)$,
For learning, Compute C $= \sum_{i=1}^{N} x_i x_i^T \in R^{p\times p}$
Solve $\beta = (C + \lambda I)^{-1} X^T Y$
The time complexity for learning is $Np^2 + p^3$.
To predict for a new test point x , we have to calulate $\beta^T x$. The time complexity for this is p.

New Way
For an input $(x_1, y_1), ...(x_N, y_N)$,
Compute $K \in R^{N\times N}, K_{i,j} = x_i^T x_j$
Solve $\alpha = (K + \lambda I)^{-1} Y$
The time complexity for learning is $N^2 p + N^3$.
To predict for a new test point x , we compute $\sum_{i=1}^{N} \alpha_i x^T x_i$.
The two ways are different computationally , but they are the same statistically.

What if we use basis expansion? $h(x), R^p \rightarrow R^M$

Old Way

$X_i \rightarrow h(x_i)$
For Learning , compute $C \rightarrow \sum_i h(x_i)h(x_i)^T, C \in R^{M\times M}$
Solve $\beta = (C + \lambda I)^{-1}h(X)^T Y$ where $h(X) \in R^{N\times M}$
To predict for new point x, compute $\beta^T h(x)$.

New Way

$K_{ij} = h(x_i)^T h(x_j)$
For learning, compute $\alpha = (K + \lambda I)^{-1}Y$
To predict for new point x, compute $\sum_{i=1}^N \alpha_i h(x_i)^T h(x)$

Note that we only compute inner products of $h(x)$ and not compute $h(x)$ itself.

2.1.3 Polynomial Kernel

Suppose we have $p = 2$ and we decide to use the basis expansion $h(x) = (x_1x_1, x_1x_2, x_2x_1, x_2x_2)$.
Note that $h(x)^T h(x') = x_1x_1x_1'x_1' + x_1x_2x_1'x_2' + x_2x_1x_2'x_1' + x_2x_2x_2'x_2'$.

Let us look at a faster way to compute $h(x)^T h(x')$.

$$\begin{aligned}\text{Claim}: \ h(x)^T h(x') &= (x \cdot x')^2 \\ &= (x_1x_1' + x_2x_2')^2\end{aligned}$$

$$\begin{aligned}\text{Claim: } &\text{For all } p \ (x \cdot x')^2 = h(x)^T h(x') \\ &\text{For } h_m(x) = x_i x_j\end{aligned}$$

Claim: For $(x \cdot x')^d$, corresponds to d^{th}order basis expansion of polynomial h.