Kernel Learning and Neural Networks

January 29, 2016

1 Main Idea

In support vector machine we (try to find a classifier $f : \phi(x) \to y$ ie. $f(x) = w^T \phi(x)$ in this case) solve the following problem -

$$\arg\min_{w} \frac{\lambda}{2} w^T w + \frac{1}{N} \sum_{i=1}^{N} L(y_i, w^T \phi(x_i)). \tag{1}$$

In this case $\phi(x)$ is the feature mapping of the kernel used. using the kernel approximation technique we can approximate this to $z(x)$. If we use family of shift invariant kernels and random Fourier features to approximate them, then $z(x)$ is going to be a function of sine/cosine. So idea is to optimize over following -

$$\arg\min_{\theta, w} \frac{\lambda}{2} w^T w + \frac{1}{N} \sum_{i=1}^{N} L(y_i, w^T z_\theta(x_i)). \tag{2}$$

2 Shift Invariant Kernels

2.1 Random Fourier Features

Random Fourier features use sine/cosine as their basis function to approximate the kernel and take advantage of Theorem 1 to approximate the kernel

Theorem 1. *A continuos kernel $k(x, y) = k(x-y)$ on R^d is positive definite iff $k(\delta)$ is the Fourier transform of a non-negative measure.*

If a shift variant kernel $k(\delta)$ is properly scaled then Theorem 1 guarantees that $p(\theta)$ in the 3 is a proper probability distribution.

$$k(x-y) = \int_{R^d} p(\theta) e^{j\theta^T (x-y)} d\theta = E_\theta(\zeta_\theta(x)\zeta_\theta(y)^*), \tag{3}$$

[**?**]. Basically random Fourier features approximate the integral in 3 using samples drawn from $p(\theta)$.

Let's say we draw L samples $\theta_1, \theta_2, ..., \theta_L$ samples from $p(\theta)$.

$$\begin{aligned}
k(x-y) &= \int_{R^d} p(\theta) e^{j\theta^T(x-y)} d\theta \\
&= \int_{R^d} p(\theta) cos(\theta^T x - \theta^T y) d\theta \\
&\approx \frac{1}{L} \sum_{i=1}^{L} cos(\theta_i^T x - \theta_i^T y) \\
&= \frac{1}{L} \sum_{i=1}^{L} cos(\theta_i^T x) cos(\theta_i^T y) + sin(\theta_i^T x) sin(\theta_i^T y) \\
&= \frac{1}{L} \sum_{i=1}^{L} [cos(\theta_i^T x), sin(\theta_i^T x)]^T [cos(\theta_i^T y), sin(\theta_i^T y)] \\
&= z(x)^T z(y)
\end{aligned}$$

where $z(x) = \frac{1}{\sqrt{L}}[cos(\theta_i^T x), sin(\theta_i^T x)] \in R^{2L}$ is an approximate non linear feature mapping (mapped to $2L$ dimensional space).

3 Neural Netoworks

4 Related papers to keep in mind

1) An Exploration of Parameter Redundancy in Deep Networks with Circulant Projections

5 Challenges

- Learning θ in eqn 2 can be thought of as a single layer NN. What happens if we add multiple layers? Is it still a family of shift invariant kernels?
- What if we use activation functions motivated by RFF - sine/cosine? What does that lead to? Is it an interesting activation function?
- How does this activation function compared to current state-of-art activation functions?
- Does this help us in understanding NN fundamentally? Can we prove anything interesting or anything that leads to better intuition about NN?
-
-

6 Timeline

- Implement paper [2]
- Implement NN and deep NN for eqn 2.
-
-
-

References

[1] Rahimi, Ali, and Benjamin Recht. "Random features for large-scale kernel machines." Advances in neural information processing systems. 2007. [2] Yu, Felix X., et al. "Compact Nonlinear Maps and Circulant Extensions." arXiv preprint arXiv:1503.03893 (2015).