

Solving Large Scale Kernel Machines using Random Features

Motivation

- Terabyte and even Petabyte scale datasets
- Batch kernel methods have good performance but do not scale well to massive datasets
 - Recall the kernel matrix \mathbf{K} is $n \times n$, where n is the number of samples
 - Suppose we have 10^6 samples. Storage requirements for \mathbf{K} :
8 bytes per double * 10^6 * 10^6 = **8 TB** memory
- Expensive to evaluate new test point:
 - $f(\mathbf{x}) = \sum_{i=1}^N c_i k(\mathbf{x}_i, \mathbf{x}) \Rightarrow O(Nd)$

Random Features* to Approximate Kernel Functions

- Approximate shift-invariant kernels (i.e. Gaussian):

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \approx \mathbf{z}(\mathbf{x})' \mathbf{z}(\mathbf{y}) \quad \mathbf{z}: \mathcal{R}^d \rightarrow \mathcal{R}^D$$

$$\min_{\mathbf{w}} \|\mathbf{Z}' \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2,$$

$$f(x) = \mathbf{w}' \mathbf{z}(\mathbf{x})$$

- What do we gain?

- Scale to very large datasets with competitive accuracy
- $O(D \cdot d)$ operations to compute new test point
- Linear learning methods for non-linear kernels

*Rahimi and Recht. Random Features for Large-Scale Kernel Machines. NIPS 2007.

Project Goals

- Understand the technique of random features
- Compare the performance of various random feature sets to traditional kernel methods
- Evaluate the performance and feasibility of this technique on very large datasets, i.e. ImageNet

Resources

Papers:

- Rahimi and Recht. Random Features for Large-Scale Kernel Machines. NIPS 2007.
- Rahimi and Recht. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. NIPS 2008.

Code:

- <http://berkeley.intel-research.net/arahimi/c/random-features/random-features.tgz>

Datasets:

- Datasets used in paper as a benchmark:
<http://berkeley.intel-research.net/arahimi/c/random-features/data/>
- Really big dataset:
<http://www.image-net.org/challenges/LSVRC/2010/download-public>