Nyström method vs Random Fourier Features

Blanca Cano Camarero

Universidad Autónoma de Madrid

April 12, 2023

Overview

Explored articles

1. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison (ver Yang et al. (2012)).
2. Using the Nyström Method to Speed Up Kernel Machines (ver Williams and Seeger (2000))

Objective: Understand the different between Nyström Method and Random Fourier Features.

Context

- One limitation of kernel methods is their high computational cost, which is at least quadratic in the number of training examples, due to the calculation of kernel matrix.
- To avoid computing kernel matrix, one common approach is to approximate a kernel learning prob- lem with a linear prediction problem.

Two alternatives and their hypothesis

- **Random Fourier features (only for shift-invariant kernels).**
- **Nyström method.**

Same usage

Nyström See https://scikit-learn.org/stable/modules/generated/sklearn.kernel_approximation.Nystroem.html#sklearn.kernel_approximation.Nystroem

```
class sklearn.kernel_approximation.Nystroem(
  # rbf kernel
  kernel='rbf', *, gamma=None,
  # other kernels
  coef0=None, degree=None,kernel_params=None,
  n_components=100,
  # others
  random_state=None, n_jobs=None)
```

RFF https://scikit-learn.org/stable/modules/generated/sklearn.kernel_approximation.RBFSampler.html

```
class sklearn.kernel_approximation.RBFSampler(*, gamma=1.0,
    n_components=100, random_state=None)
```

Same usage sklearn

Nyström

```
>>> from sklearn import datasets, svm
>>> from sklearn.kernel_approximation import Nystroem
>>> X, y = datasets.load_digits(n_class=9, return_X_y=True)
>>> data = X / 16.
>>> clf = svm.LinearSVC()
>>> feature_map_nystroem = Nystroem(gamma=.2,
...                                 random_state=1,
...                                 n_components=300)
>>> data_transformed = feature_map_nystroem.fit_transform(data)
>>> clf.fit(data_transformed, y)
```

Same usage sklearn

RFF

```
>>> from sklearn.kernel_approximation import RBFSampler
>>> from sklearn.linear_model import SGDClassifier
>>> rbf_feature = RBFSampler(gamma=1, random_state=1)
>>> X_features = rbf_feature.fit_transform(X)
>>> clf = SGDClassifier(max_iter=5, tol=1e-3)
>>> clf.fit(X_features, y)
SGDClassifier(max_iter=5)
```

Working in a unified framework for Approximate Large-Scale Kernel Learning

- We are focus on the RBF kernel.
- Our goal is to efficiently learn a kernel prediction function by solving the following optimization problem:

$$\min_{f \in \mathcal{H}_D} \frac{\lambda}{2} \|f\|^2_{H_k} + \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i). \tag{1}$$

- H_k is the RKHS endowed by the kernel K.
- $H_D = \text{span}(K(x_1, \cdot), \ldots, K(x_N, \cdot))$, **The high computational cost of kernel learning arises from the fact that we have to search for an optimal classifier f in this space.**
- $l(z, y)$ **is a convex loss function.**

Random Fourier Features Definition

- The random Fourier features are constructed by first sampling Fourier components $u_1, \ldots u_m$ from $p(u)$.
- Projecting each example to $u_1, \ldots u_m$.
- Then passing them through sine and cosine

$$z_f(x) = \left(\sin(u_1^T x), \cos(u_1^T x), \ldots, \sin(u_m^T x), \cos(u_m^T x)\right). \tag{2}$$

- Let define $H_a^f = \text{span}(s_1, c_1, \ldots, s_m, c_m)$ where $s_i(x) = \sin(u_i^T x)$.
- The linear machine learnt by solving

$$\min_{f \in \mathcal{H}_a^f} \frac{\lambda}{2} \|f\|_{H_k}^2 + \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i). \tag{3}$$

is $f(x) = w^T z_f(x)$.

Error bound in RFF

$$O(N^{-1/2} + m^{-1/2}) \tag{4}$$

where N is the number of training examples and m is the number of sampled Fourier components.

The Nyström method (see Williams and Seeger (2000))

Let K be partitioned into blocks $K_{m,m}$, $K_{n-m,m} = K_{m,n-m}^T$ and $K_{n-m,n-m}$. The approximation is

$$\tilde{K} = K_{n,m} K_{m,m}^{-1} K_{m,n}. \tag{5}$$

See math foundation on blackboard.

The Nyström method approximates the full kernel matrix K by

- First sampling m examples denoted by $\hat{x}_i, \cdots, \hat{x}_m$.
- Then constructing a low rank matrix by

$$\hat{K}_r = K_b \hat{K}^{\dagger} K_b^T, \tag{6}$$

  where $K_b = [k(x_i, \hat{x}_j)]$, $\hat{K}^{\dagger}$ is the pseudo inverse of $\hat{K}$.
- In order to train the linear machine, we derive a vector representation:

$$z_n(x) = \hat{D}^{-\frac{1}{2}} \hat{V}_r^T \left(K(x, \hat{x}_1), \ldots, K(x, \hat{x}_m)\right)^T, \tag{7}$$

  where $\hat{D}_r = dia(\hat{\lambda}_1, \ldots, \hat{\lambda}_r)$ and V_r the eigenvectors in columns.
- $H_a^n = \text{span}(\hat{\varphi}_1, \ldots, \hat{\varphi}_r)$ where $\hat{\varphi}_i$ are the first r normalized eigenfunctions of the operator L_m.

Error bound in Nyström methods

$$O(m^{-1/2}) + O(m^{-1/2}) \tag{8}$$

- The approximation error of the Nyström method, measured in spectral norm is $O(m^{-1/2})$. (See Drineas and Mahoney (2005))
- The generalization performance caused by the approximation of the Nyström method $O(m^{-1/2})$. (see Cortes et al. (2010))

where m is the number of sampled training examples.

Difference

- Randomness and data independence.
- Hypothesis: Nyström adapt better to data.
- Experiment see article.
- Theoretical proof see article.

To deep in

- Math proofs Yang et al. (2012).
- For mathematical bound of Nyström: Drineas and Mahoney (2005) and Cortes et al. (2010).

Random Fourier Features computational cost

Step	Task	Theory	Cost	Memory
1	Sampling Fourier components $u_1, \dots u_m$ from $p(u)$.	Inverse transform sampling, Accept or reject, Montecarlo.	O(1)	O(m)
2	Compute $z_f(x) = (\sin(u_1^T x), \cos(u_1^T x), \dots, \sin(u_m^T x), \cos(u_m^T x))$		O(m)	O(2m)

Nyström

Step	Task	Theory	Cost	Memory
1	Sampling		O(1)	O(m)
2	constructing a low rank matrix by $\hat{K}_r = K_b \hat{K}^\dagger K_b^T$.	SVD and matrix multiplication	$O(n^3)$	$O(m^2)$

Ridge regression

Cost	Memory
$O(m^2(N+m))$	$O(mn+n)$

Next week

1. **Nystroem Method vs Random Fourier Features: A Theoretical and Empirical Comparison, Advances in Neural Information Processing Systems 2012**
2. Random features for kernel approximation: A survey on algorithms, theory, and beyond
3. **Williams, C.K.I. and Seeger, M. "Using the Nystroem method to speed up kernel machines", Advances in neural information processing systems 2001 T. Yang, Y. Li, M. Mahdavi, R. Jin and Z. Zhou**
4. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning.
5. Randomness in neural networks: an overview
6. Fast and scalable polynomial kernels via explicit feature maps
7. On the error of random Fourier features
8. A survey on large-scale machine learning
9. Sharp analysis of low-rank kernel matrix approximations

References I

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 113–120, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/cortes10a.html`.

Petros Drineas and Michael W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(72):2153–2175, 2005. URL `http://jmlr.org/papers/v6/drineas05a.html`.

References II

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf.

Tianbao Yang, Yu-feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/621bf66ddb7c962aa0d22ac97d69b793-Paper.pdf.