高维情形下线性模型的泛化误差研究

2024 春季学期 week11

Yukun Dong

目录

1 任务:

- 阅读代码以及 Eigenpro 系列文章，弄清楚其原理。
- 阅读 Ma, Bassily, and Belkin (2018) 的文章，观察其如何证明最优的 batch size m^*。
- 尝试利用子取样的方法优化 RKHS norm 的计算，继续尝试复现 Belkin, Ma, and Mandal (2018) 的 fig4。
- Random feature 的 double descent。
- 为什么 Gaussian 核表现不好？
- 超参数怎么选？

1.1 Diving into the shallows: a computational perspective on large-scale shallow learning

1.1.1 问题的动机

有限样本对应的是 Gram matrix K，而无限样本情形（总体情形）对应的是 Hilbert-Schmidt operator $\mathcal{K}$. $\mathcal{K}$ 是一个 $L^2(\mathcal{X}) \to L^2(\mathcal{X})$ 的紧自伴算子：

$$\mathcal{K}f(x) = \int K(x,z)f(z)d\mu_z$$

其中 μ_z 可以看做是总体分布对应的测度，$\mathcal{X}$ 为样本空间。这里设 $\mathcal{K}$ 具有递降趋于 0 的特征值 $\lambda_1, \lambda_2, \cdots$, 我们有如下定理：

Theorem 1. *If k is an infinitely differentiable kernel, the rate of eigenvalue decay is super-polynomial, i.e.*

$$\lambda_i = O(i^{-P}) \qquad \forall P \in \mathbb{N}$$

Moreover, if k is an infinitely differentiable radial kernel (e.g., a Gaussian kernel), there exist constants $C, C' > 0$ such that for large enough i,

$$\lambda_i < C' \exp\left(-Ci^{1/p}\right)$$

即对于无限次可导的核函数，其对应的 Hilbert-Schmidt operator 具有超多项式特征值衰减性质。

下面考虑我们要估计的向量（无穷维时就是函数），是否能由梯度下降方法得到？在最小二乘的设定之下有如下的迭代：

Linear regression. Consider n labeled data points $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n) \in \mathcal{H} \times \mathbb{R}\}$. To simplify the notation let us assume that the feature map has already been applied to the data, i.e., $\boldsymbol{x}_i = \phi(\boldsymbol{z}_i)$. Least square linear regression aims to recover the parameter vector α^* that minimize the empirical loss as follows:

$$L(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}(\langle \boldsymbol{\alpha}, \boldsymbol{x}_i\rangle_{\mathcal{H}} - y_i)^2 \tag{1}$$

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathcal{H}} L(\boldsymbol{\alpha}) \tag{2}$$

Minimizing the empirical loss is related to solving a linear system of equations. Define the data matrix[3] $X \stackrel{\text{def}}{=} (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^T$ and the label vector $\boldsymbol{y} \stackrel{\text{def}}{=} (y_1, ..., y_n)^T$, as well as the (non-centralized) covariance matrix/operator,

$$H \stackrel{\text{def}}{=} \frac{2}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^T = \frac{2}{n}X^TX \tag{3}$$

Rewrite the loss as $L(\boldsymbol{\alpha}) = \frac{1}{n}\|X\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2$. Since $\nabla L(\boldsymbol{\alpha})\mid_{\boldsymbol{\alpha}=\boldsymbol{\alpha}^*} = 0$, minimizing $L(\boldsymbol{\alpha})$ is equivalent to solving the linear system

$$H\boldsymbol{\alpha} - \boldsymbol{b} = 0 \tag{4}$$

For linear systems of equations gradient descent takes a particularly simple form known as Richardson iteration [Ric11]. It is given by

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} - \eta(H\boldsymbol{\alpha}^{(t)} - \boldsymbol{b}) \tag{5}$$

We see that

$$\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^* = (\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^*) - \eta H(\boldsymbol{\alpha}^{(t)} - \boldsymbol{\alpha}^*)$$

and thus

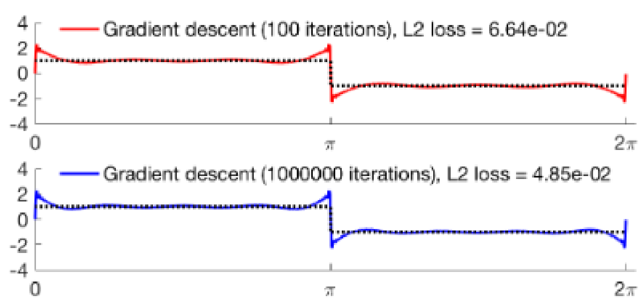
$$\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^* = (I - \eta H)^t(\boldsymbol{\alpha}^{(1)} - \boldsymbol{\alpha}^*) \tag{6}$$

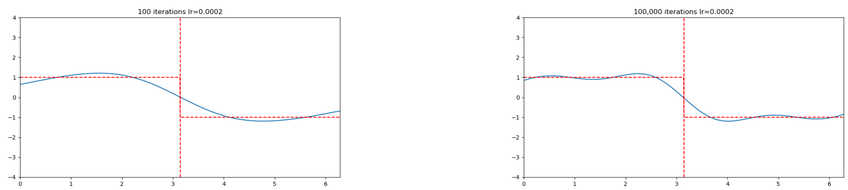
It is easy to see that for convergence of $\boldsymbol{\alpha}^t$ to $\boldsymbol{\alpha}^*$ as $t \to \infty$ we need to ensure[4] that $\|I - \eta H\| \le 1$. It follows that $0 < \eta < 2/\lambda_1(H)$.

下面定义梯度下降算法的 computational reach $\mathcal{CR}_t(\varepsilon) := \{\mathbf{v} \in \mathcal{H} : ||(I - \eta H)^m \mathbf{v}|| \le \varepsilon ||\mathbf{v}||\}$。我们下面通过一个简单的实验去看看梯度下降方法要拟合一个函数，大概需要多少次迭代？考虑一个在分类问题中很自然的函数，Heaviside step function $g(x)$，在 $(0, \pi)$ 上取 1，在 $(\pi, 2\pi)$ 上取-1。我们考虑无穷样本下（样本量趋于无穷），使用高斯核进行梯度下降，在平方损失下逼近该函数。简单的理论推导显示，需要 $O(\exp(\frac{1}{\varepsilon^2}))$ 次迭代才能得到 $g(x)$ 的 ε-逼近。因此在迭代次数分别为 100 和 100000 时，得到的逼近效果相差不多。

这个简单的例子展现了梯度下降方法的局限性：数据的需求量是指数级别的。

在此复现了论文的图，对比如下：





(a) 迭代 100 次 (b) 迭代 10 万次

图 1: 我的模拟结果

1.1.2 EigenPro iteration

Preconditioned (stochastic) gradient descent. We will modify the linear system in Eq. 4 with an invertible matrix P, called a left preconditioner.

$$PH\boldsymbol{\alpha} - P\boldsymbol{b} = 0 \tag{14}$$

Clearly, the modified system in Eq. 14 and the original system in Eq. 4 have the same solution. The Richardson iteration corresponding to the modified system (preconditioned Richardson iteration) is

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \eta P(H\boldsymbol{\alpha} - \boldsymbol{b}) \tag{15}$$

It is easy to see that as long as $\eta\|PH\| < 1$ it converges to $\boldsymbol{\alpha}^*$, the solution of the original linear system.

Preconditioned SGD can be defined similarly by

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \eta P(H_m\boldsymbol{\alpha} - \boldsymbol{b}_m) \tag{16}$$

Algorithm: EigenPro$(X, \boldsymbol{y}, k, m, \eta, \tau, M)$
input training data $(X, \boldsymbol{y})$, number of eigen-directions k, mini-batch size m, step size η, damping factor τ, subsample size M
output weight of the linear model $\boldsymbol{\alpha}$

```
[E, Λ, λ̂_{k+1}] = RSVD(X, k+1, M)
P ≝ I − E(I − τ λ̂_{k+1} Λ^{-1}) E^T
Initialize α ← 0
while stopping criteria is False do
    (X_m, y_m) ← m rows sampled from (X, y) without replacement
    g ← (1/m)(X_m^T (X_m α) − X_m^T y_m)
    α ← α − η P g
end while
```

1.1.2.1 Linear EigenPro

Algorithm: EigenPro($\mathrm{k}(\cdot,\cdot), X, \boldsymbol{y}, k, m, \eta, s_0$)
input kernel function $\mathrm{k}(\cdot,\cdot)$, training data $(X, \boldsymbol{y})$, number of eigen-directions k, mini-batch size m, step size η, subsample size M, damping factor τ
output weight of the kernel method $\boldsymbol{\alpha}$

$K \stackrel{\text{def}}{=} \mathrm{k}(X, X)$ materialized on demand
$[E, \Lambda, \lambda_{k+1}] \leftarrow \mathrm{RSVD}(K, k+1, M)$
$D \stackrel{\text{def}}{=} E\Lambda^{-1}(I - \tau\lambda_{k+1}\Lambda^{-1})E^T$
Initialize $\boldsymbol{\alpha} \leftarrow 0$
while stopping criteria is False **do**
    $(K_m, \boldsymbol{y}_m) \leftarrow m$ rows sampled from $(K, \boldsymbol{y})$
    $\boldsymbol{\alpha}_m \stackrel{\text{def}}{=}$ portion of $\boldsymbol{\alpha}$ related to K_m
    $\boldsymbol{g}_m \leftarrow \frac{1}{m}(K_m\boldsymbol{\alpha} - \boldsymbol{y}_m)$
    $\boldsymbol{\alpha}_m \leftarrow \boldsymbol{\alpha}_m - \eta\boldsymbol{g}_m$, $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \eta D K_m^T \boldsymbol{g}_m$
end while

1.1.2.2 Kernel EigenPro

需要注意的细节：

- 步长的选择
- 代码的细节

参考文献

Belkin, Mikhail, Siyuan Ma, and Soumik Mandal. 2018. "To Understand Deep Learning We Need to Understand Kernel Learning." In *Proceedings of the 35th International Conference on Machine Learning*, edited by Jennifer Dy and Andreas Krause, 80:541–49. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v80/belkin18a.html.

Ma, Siyuan, Raef Bassily, and Mikhail Belkin. 2018. "The Power of Interpolation: Understanding the Effectiveness of SGD in Modern over-Parametrized Learning." In *Proceedings of the 35th International Conference on Machine Learning*, edited by Jennifer Dy and Andreas Krause, 80:3325–34. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v80/ma18a.html.