Stochastic Gradient Methods for Large-scale Machine Learning

Zaiwen Wen

http://bicmr.pku.edu.cn/~wenzw/bigdata2023.html

Thanks Yongfeng Li and Zhanwang Deng for preparing part of this slides

Why Optimization in Machine Learning?

Many problems in ML can be written as

$$\min_{\theta \in \mathcal{W}} \quad \sum_{i=1}^{N} \frac{1}{2} \|x_i^\top \theta - y_i\|_2^2 + \mu \|\theta\|_2^2 \quad \text{linear regression}$$

$$\min_{\theta \in \mathcal{W}} \quad \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i x_i^\top \theta)) + \mu \|\theta\|_2^2 \quad \text{logistic regression}$$

$$\min_{\theta \in \mathcal{W}} \quad \sum_{i=1}^{N} \ell(h(\theta, x_i), y_i) + \mu \varphi(\theta) \quad \text{general formulation}$$

- The pairs (x_i, y_i) are given data, y_i is the label of the data point x_i
- $\ell(\cdot)$: measures how model fit for data points (avoids under-fitting)
- $\varphi(\theta)$: regularization term (avoids over-fitting)
- $h(\theta, x)$: linear function or models constructed from deep neural networks

Sparse Logistic Regression

The logistic regression problem:

$$\min_{\theta \in \mathbb{R}^n} \quad \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i x_i^T \theta)) + \mu \|\theta\|_2^2.$$

- The data pair $\{x_i, y_i\} \in \mathbb{R}^n \times \{-1, 1\}, i \in [N]$,

Data Set	# data N	# features n	sparsity(%)
cina	16,033	132	70.49
a9a	32,561	123	88.72
ijcnn1	49,990	22	40.91
covtype	581,012	54	77.88
url	2,396,130	3,231,961	99.99
susy	5,000,000	18	1.18
higgs	11,000,000	28	7.89
news20	19,996	1,355,191	99.97
rcv1	20,242	47,236	99.84
kdda	8,407,752	20,216,830	99.99

Deep Learning

The objective function is the CrossEntropy function plus regularization term:

$$\min_{\theta} \quad \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{\exp(h(\theta, x_i)[y_i])}{\sum_j \exp(h(\theta, x_i)[y_j])}\right) + \mu\|\theta\|_2^2$$

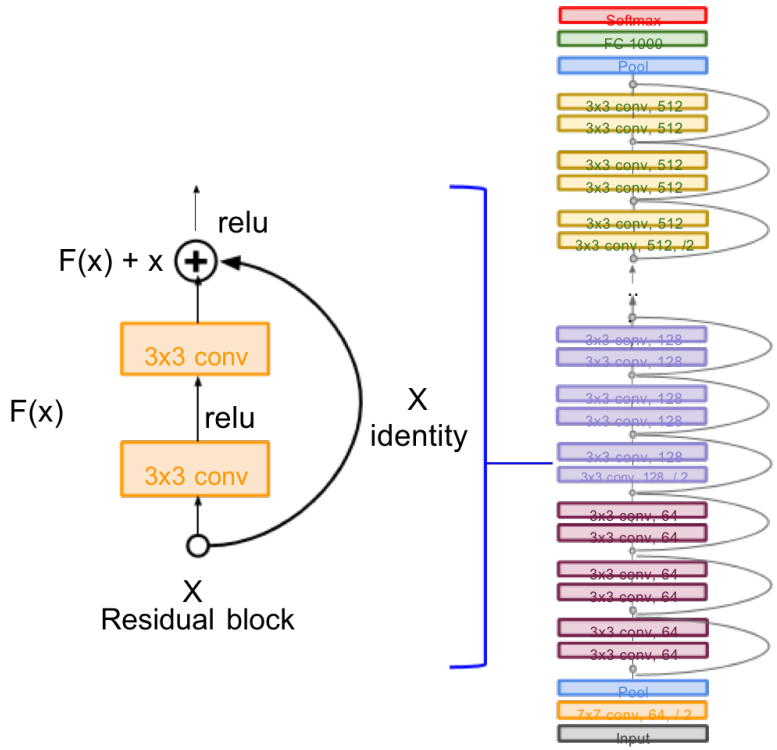
where $h(\theta, x_i)$ is output from network, and (x_i, y_i) are data points.

	Cifar-10	Cifar-100
# num_class	10	100
# number per class (training set)	5,000	500
# number per class (testing set)	1,000	100
# Total parametes of VGG-16	15,253,578	15,299,748
# Total parameters of ResNet-18	11,173,962	11,220,132

Table: A description of datasets used in the neural network experiments

ResNet Architecture

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Cited by **114474 since 2015** at Google scholar
- Stack residual blocks. Every residual block has two 3x3 conv layers.
- Make networks from shallow to deep.
- Fancy network architecture. Many Applications.
- High-computationally-cost !
- ResNet-50 on ImageNet, **29 hours using 8 Tesla P100 GPUs**



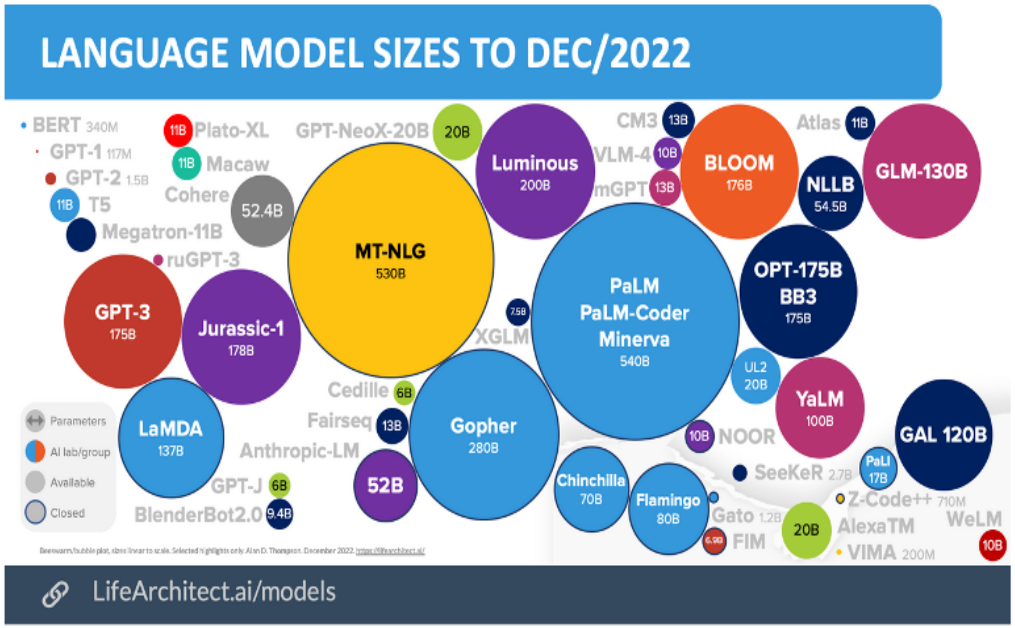


Chart. Major AI language models 2018-2022, GPT-3 on the left in red.

Dataset	Tokens (billion)	Assumptions	Tokens per byte (Tokens / bytes)	Ratio	Size (GB)
Web data	**410B**	–	*0.71*	*1:1.9*	**570**
WebText2	**19B**	*25% > WebText*	*0.38*	*1:2.6*	*50*
Books1	**12B**	*Gutenberg*	*0.57*	*1:1.75*	*21*
Books2	**55B**	*Bibliotik*	*0.54*	*1:1.84*	*101*
Wikipedia	**3B**	*See RoBERTa*	*0.26*	*1:3.8*	*11.4*
Total	**499B**			*753.4GB*	

Table. GPT-3 Datasets. Disclosed in **bold**. Determined in *italics*.

ChatGPT (2022/12)训练代价

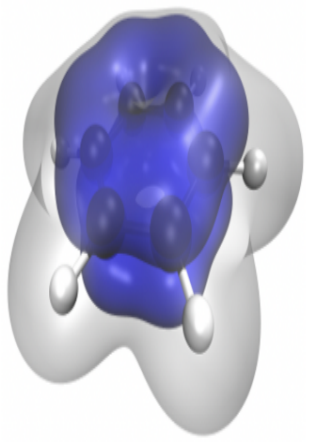
- 硬件代价：超级计算机 10,000 GPUs 和 285,000 CPU 核，约10亿美元
- 人员费用：首席科学家Ilya Sutskever 190万美元/年(2016年)，世界顶级Ph.D团队，120人，第一年人员费用超过2亿美元
- 数据收集时间12-18个月，训练时间9-12个月

AI For Science

J. Jumper *et al.* Nature (2021)

TABLE IV
RESOURCE AND TIME COST COMPARE.

Implementation	Framework	Training Process	Hardward	Step Time (s)	Training Time (days)	Resource
AlphaFold	JAX [18]	Initial training Fine-tuning	128 × TPUv3	/ /	11	33792 TPU hours
OpenFold	PyTorch	Initial training Fine-tuning	128 × A100	6.186 20.657	8.39	25774 GPU hours
FastFold	PyTorch	Initial training Fine-tuning	256 × A100 512 × A100	2.487 4.153	**2.81**	**20738 GPU hours**



Framework	MCMC Steps	Det weights	Envelope	GPU Hours	Energy (E_h)
TensorFlow [12]	10	Yes	Full Covariance	11520	-155.9263(6)
JAX	50	No	Full Covariance	1880	**-155.9348(1)**
JAX	50	No	Isotropic	**1104**	**-155.9348(1)**

Table 2: Speed and accuracy comparison for different training runs of the FermiNet on bicyclobutane. All runs were for 200,000 iterations, with the same number of determinants and hidden units as in [12]. Number of GPU hours for TensorFlow assumes 30 days of training on 16 GPUs. Lower energies are strictly better as FermiNet provides an upper bound to the exact energy.

Outline

机器学习模型

- $x \in \mathcal{X}$为数据；$y \in \mathcal{Y}$为标签，其中$\mathcal{X}$为数据所在的空间，$\mathcal{Y}$为标签的所在的空间，并且x与y服从某个未知的分布π，也即$(x, y) \sim \pi$. 数据集为$\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N$.
- $h_\theta(x)$为得到的模型，其中θ为需要优化的变量. 我们通过训练参数θ来得到合适的模型$h_\theta(x)$.
- 损失函数
  - 平方损失函数: $\ell(y, h_\theta(x)) = \|h_\theta(x) - y\|^2$.
  - 交叉熵函数: $\ell(y, h_\theta(x)) = y^{\mathrm{T}} \log(h_\theta(x)) = \sum_{j=1}^C y_j \log(h_\theta(x))_j$.
  - 链接损失函数(Hinge loss): $\ell(y, h_\theta(x)) = \max(0, 1 - h_\theta(x)^{\mathrm{T}} y)$;
  - 指数损失函数(Exponential loss): $\ell(y, h_\theta(x)) = \exp(-h_\theta(x)^{\mathrm{T}} y)$;
  - 对数损失函数(Logisticloss): $\ell(y, h_\theta(x)) = \log(1 + \exp(-h_\theta(x)^{\mathrm{T}} y))$;
  - KL 散度(KL divergence): $\ell(y, h_\theta(x)) = \sum_{j=1}^C y_j \log h_\theta(x_j) - \sum_{j=1}^C y_j \log y_j$.

优化模型

- 期望风险(Expected risk):

$$\min_{\theta} \quad f_{\pi}(\theta) = R[h_{\theta}] \stackrel{\text{def}}{=\!=} \mathop{\mathbb{E}}_{(x,y)\sim\pi}[\ell(y, h_{\theta}(x))]. \tag{1}$$

- 经验风险(Empirical risk): 令$f_i(\theta) = \ell(y_i, h_{\theta}(x_i))$,

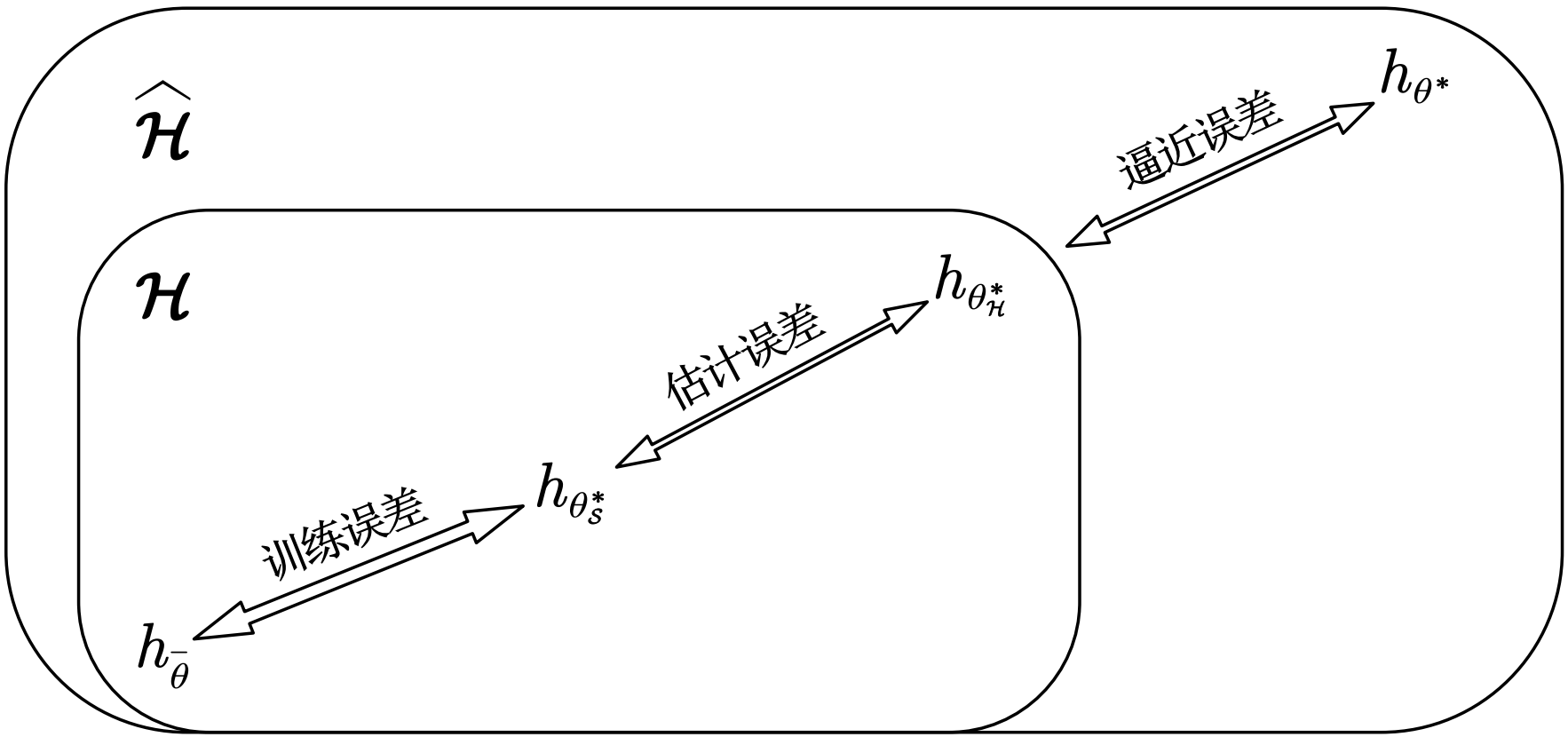
$$\min_{\theta} \quad f_{\mathcal{S}}(\theta) = \hat{R}_{\mathcal{S}}[h_{\theta}] \stackrel{\text{def}}{=\!=} \frac{1}{N}\sum_{i=1}^{N} f_i(\theta). \tag{2}$$

- $\mathop{\mathbb{E}}_{(x,y)\sim\pi}$ 表示关于(x, y)的分布求期望:

$$\begin{aligned} \mathop{\mathbb{E}}_{(x_i,y_i)\sim\pi}[\hat{R}_{\mathcal{S}}[h_{\theta}]] &= \mathop{\mathbb{E}}_{(x_i,y_i)\sim\pi}\left[\frac{1}{N}\sum_{i=1}^{N}\ell(y_i, h_{\theta}(x_i))\right] \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\pi}[\ell(y, h_{\theta}(x))] = R[h_{\theta}]. \end{aligned}$$

误差分解

- 假设真实解满足$h_{\theta^*} \in \widehat{\mathcal{H}}$. 然而现实情况下时只能取近似的假设空间$\mathcal{H} \subseteq \widehat{\mathcal{H}}$，比如$\mathcal{H}$是多项式、分段线性函数或者神经网络等组成的函数空间. 在不引起歧义的情况下，混淆使用h和h_θ所属的空间，即$\theta \in \mathcal{H}$表示$h_\theta \in \mathcal{H}$.



误差分解

- 设θ^*为最优模型对应的最优参数，对应的模型为h_{θ^*},即:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} R[h_\theta], \text{ s.t. } \theta \in \hat{\mathcal{H}}. \tag{3}$$

- 设$\theta^*_{\mathcal{H}}$为在假设空间$\mathcal{H}$中的最小化期望风险模型对应的最优参数，对应的模型为$h_{\theta^*_{\mathcal{H}}}$,即：

$$\theta^*_{\mathcal{H}} = \underset{\theta}{\operatorname{argmin}} R[h_\theta], \text{ s.t. } \theta \in \mathcal{H}. \tag{4}$$

- 设$\theta^*_{\mathcal{S}}$为属于假设空间$\mathcal{H}$并且在样本数据集$\mathcal{S}$上经验风险模型对应的最优参数，对应的模型为$h_{\theta^*_{\mathcal{S}}}$,即:

$$\theta^*_{\mathcal{S}} = \underset{\theta}{\operatorname{argmin}} \hat{R}_{\mathcal{S}}[h_\theta], \text{ s.t. } \theta \in \mathcal{H}. \tag{5}$$

- 设$\bar{\theta}$为(5)的近似解，对应的模型为$h_{\bar{\theta}}$,即:

$$\bar{\theta} \approx \underset{\theta}{\operatorname{argmin}} \hat{R}_{\mathcal{S}}[h_\theta], \text{ s.t. } \theta \in \mathcal{H}. \tag{6}$$

误差估计

- 相应的实际模型与最优模型的期望风险的误差分解为：

$$R[h_{\theta^*}] - R[h_{\bar{\theta}}] = \underbrace{R[h_{\theta^*}] - R[h_{\theta^*_{\mathcal{H}}}]}_{\text{逼近误差}} + \underbrace{R[h_{\theta^*_{\mathcal{H}}}] - R[h_{\theta^*_{\mathcal{S}}}]}_{\text{泛化误差}} + \underbrace{R[h_{\theta^*_{\mathcal{S}}}] - R[h_{\bar{\theta}}]}_{\text{优化误差}}.$$

- 逼近误差与假设空间$\mathcal{H}$的表达能力有关
- 泛化误差与样本和假设空间有关．通常情况下，样本量越大，求解得到的$h_{\theta^*_{\mathcal{S}}}$与$h_{\theta^*_{\mathcal{H}}}$越接近．
- 优化误差(训练误差)与许多因素都有关系，比如模型的选择，优化算法的设计，包括初始点的选择等．

Hoeffding Inequality

Let $X_1, X_2, \cdots$ be a sequence of i.i.d. random variables and assume that for all i, $E(X_i) = \mu$ and $\mathcal{P}(a \leq X_i \leq b) = 1$. Then for any $\epsilon > 0$

$$\mathcal{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2n\epsilon^2}{(b-a)^2}\right) \tag{7}$$

- The Hoeffding Inequality describes the asymptotic property that sampling mean convergences to expectation.
- Azuma-Hoeffding inequality is a martingle version. Let $X_1, X_2, \cdots$ be a martingale difference sequence with $|X_i| \leq B$ for all $i = 1, 2, ...$ Then

$$\mathcal{P}(\sum_{i=1}^{n} X_i \geq t) \leq \exp\left(-\frac{2t^2}{nB^2}\right)$$

$$\mathcal{P}(\sum_{i=1}^{n} X_i \leq t) \leq \exp\left(-\frac{2t^2}{nB^2}\right)$$

误差估计

设假设空间$\mathcal{H}$为有限的，损失函数满足$0 \leq \ell(h(x_i), y_i) \leq 1$, $\forall x_i, y_i, h \in \mathcal{H}$. 则对于$0 < \delta < 1$,则有以概率$1-\delta$,有

$$|R[h] - \hat{R}[h]| \leq \sqrt{\frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{2N}}.$$

证明: 根据$\hat{R}[h]$定义及$\mathbb{E}[\hat{R}[h]] = R[h]$，由霍夫丁不等式有

$$P(|\hat{R}[h] - R[h]| \geq \varepsilon) \leq 2e^{-2N\varepsilon^2}.$$

所以对于有限的假设空间$\mathcal{H}$,有

$$P\left(\bigcup_{h\in\mathcal{H}}\{|\hat{R}[h] - R[h]| \geq \varepsilon\}\right) \leq 2|\mathcal{H}|e^{-2N\varepsilon^2}.$$

如果希望$\mathcal{P}\left(\bigcup_{h\in\mathcal{H}}\{|\hat{R}[h] - R[h]| \geq \varepsilon\}\right) \leq \delta$,则需要样本量满足

$$N = \frac{1}{2\varepsilon^2}\ln\left(\frac{2|\mathcal{H}|}{\delta}\right) = \mathcal{O}\left(\frac{\ln|\mathcal{H}| + \ln(\delta^{-1})}{\varepsilon^2}\right). \tag{8}$$

泛化误差估计

$$R[h_{\mathcal{S}}^*] - R[h_{\mathcal{H}}^*] \leq 2\sqrt{\frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{2N}}.$$

证明：由于

$$R[h_{\mathcal{S}}^*] - R[h_{\mathcal{H}}^*] = \underbrace{R(h_{\mathcal{S}}^*) - \hat{R}(h_{\mathcal{S}}^*)}_{(1)} + \underbrace{\hat{R}(h_{\mathcal{S}}^*) - \hat{R}(h_{\mathcal{H}}^*)}_{(2)} + \underbrace{\hat{R}(h_{\mathcal{H}}^*) - R(h_{\mathcal{H}}^*)}_{(3)}.$$

对于(1)式和(3)式，其绝对值均小于$\sup|R[h] - \hat{R}[h]|, \forall h \in \mathcal{H}$.对于(2)式，根据$\hat{R}[h]$的定义我们可以知道其非正，由此可得：

$$|R[h_{\mathcal{S}}^*] - R[h_{\mathcal{H}}^*]| \leq 2\sup|R[h] - \hat{R}[h]|, \forall h \in \mathcal{H}.$$

再根据前一页可知在大概率下有下式成立：

$$|R[h_{\mathcal{S}}^*] - R[h_{\mathcal{H}}^*]| \leq 2\sqrt{\frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{2N}}.$$

在$\mathcal{H}$为常数的情况下，通过增大样本数量来极小化经验误差求解得到的$h_{\mathcal{S}}^*$的期望误差与其经验误差相差不大.

泛化误差

- What if $|\mathcal{H}| = \infty$? This bound doesn't work
- For a two label classification problem, with a probability $1 - \delta$, we have

$$\sup_{h \in \mathcal{H}} |\hat{R}_n[h] - R[h]| \leq O\left(\sqrt{\frac{VC[\mathcal{H}] \log(\frac{n}{VC[\mathcal{H}]}) + \log(\frac{1}{\delta}))}{n}}\right) \tag{9}$$

  where $VC[\mathcal{H}]$ is a VC dimension of $\mathcal{H}$.
- Finite VC dimension is sufficient and necessary condition of empirical risk concentration for two label classification.
- 拉德马赫复杂度

Outline

梯度下降算法

考虑如下随机优化问题：

$$\min_{x\in\mathbb{R}^n} \quad f(x) \overset{\text{def}}{=\!=} \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{10}$$

- 假设每一个$f_i(x)$是凸的、可微的．可以运用梯度下降算法

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k).$$

- 梯度必须计算出所有的$\nabla f_i(x^k)$然后将它们相加：

$$\nabla f(x^k) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^k).$$

- 然而机器学习中采集到的样本量巨大，因此计算$\nabla f(x^k)$需要非常大的计算量．使用传统的梯度法求解机器学习问题并不是一个很好的做法．

随机梯度下降算法(SGD)

- SGD的基本迭代格式为

$$x^{k+1} = x^k - \alpha_k \nabla f_{s_k}(x^k), \tag{11}$$

其中s_k是从$\{1, 2, \cdots, N\}$中随机等可能地抽取的一个样本

- 步长α_k在机器学习中被称为学习率(learning rate).
- 随机梯度的条件期望恰好是全梯度，即

$$\mathbb{E}_{s_k}[\nabla f_{s_k}(x^k)|x^k] = \nabla f(x^k).$$

- 常用的形式是小批量（mini-batch）随机梯度法. 随机选择元素个数很少的集合$\mathcal{I}_k \subset \{1, 2, \cdots, N\}$，然后执行迭代格式

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{I}_k|} \sum_{s \in \mathcal{I}_k} \nabla f_s(x^k),$$

- 如果$f_i(x)$不可微，可以考虑随机次梯度算法:

$$x^{k+1} = x^k - \alpha_k g^k$$

动量方法

- 为了克服随机梯度下降法收敛速度慢的缺陷，提出了动量方法（momentum），其思想是在算法迭代时一定程度上保留之前更新的方向，同时利用当前计算的梯度调整最终的更新方向.
- 动量方法的具体迭代格式如下：

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_{s_k}(x^k), \tag{12}$$

$$x^{k+1} = x^k + v^{k+1}. \tag{13}$$

在计算当前点的随机梯度$\nabla f_{s_i}(x^k)$后，将其和上一步更新方向v^k做线性组合来得到新的更新方向v^{k+1}.

- 当$\mu_k = 0$时该方法退化成随机梯度下降法. 参数μ_k的范围是$[0, 1)$，通常取$\mu_k \geq 0.5$，其含义为迭代点带有较大惯性，每次迭代会在原始迭代方向的基础上做一个小的修正.
- 在普通的梯度法中，每一步迭代只用到了当前点的梯度估计，动量方法的更新方向还使用了之前的梯度信息.
- 当许多连续的梯度指向相同的方向时，步长就会很大，这从直观上看也是非常合理的.

动量方法

图1比较了梯度法和动量方法的表现. 可以看到普通梯度法生成的点列会在椭圆的短轴方向上来回移动，而动量方法生成的点列更快收敛到了最小值点.

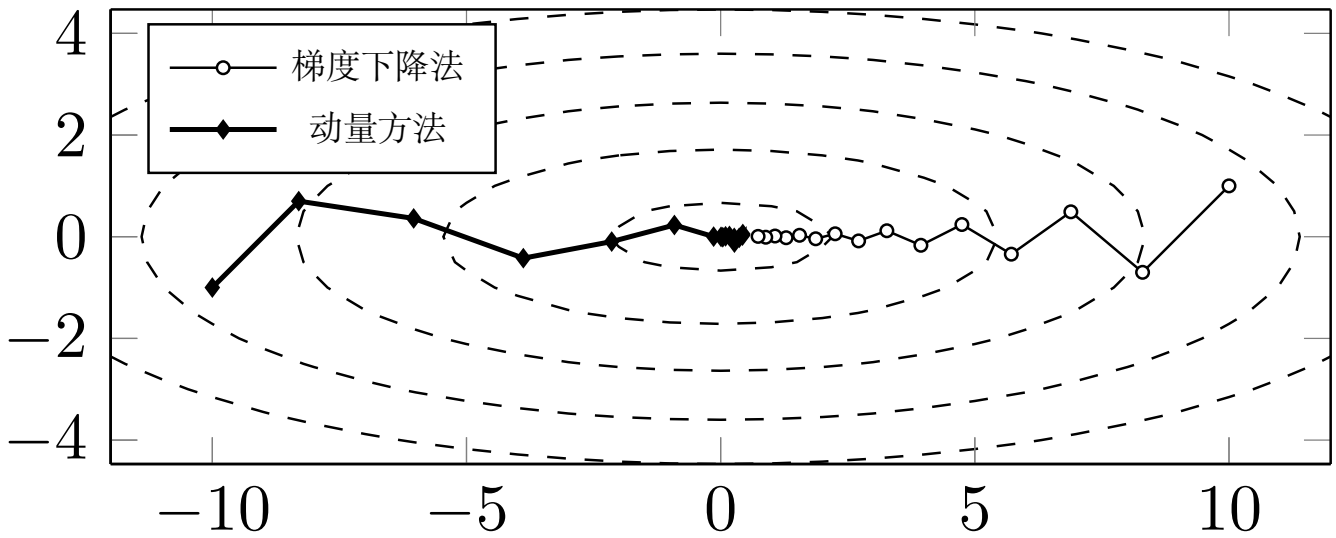


Figure: 动量方法在海瑟矩阵病态条件下的表现

Nesterov加速算法

- 假设$f(x)$为光滑的凸函数. 针对凸问题的Nesterov加速算法为

$$y^{k+1} = x^k + \mu_k(x^k - x^{k-1})$$
$$x^{k+1} = y^k - \alpha_k \nabla f(y^k)$$

- 针对光滑问题的Nesterov加速算法迭代的随机版本为

$$y^{k+1} = x^k + \mu_k(x^k - x^{k-1}), \tag{14}$$
$$x^{k+1} = y^{k+1} - \alpha_k \nabla f_{s_k}(y^{k+1}), \tag{15}$$

其中$\mu_k = \frac{k-1}{k+2}$，步长α_k是一个固定值或者由线搜索确定.

- 可以看出，二者的唯一区别为随即版本将全梯度$\nabla f(y^k)$替换为随机梯度$\nabla f_{s_k}(y^{k+1})$.

Nesterov加速算法与动量方法的联系

- 若在第k步迭代引入速度变量$v^k = x^k - x^{k-1}$，再合并原始Nesterov加速算法的两步迭代可以得到

$$x^{k+1} = x^k + \mu_k(x^k - x^{k-1}) - \alpha_k \nabla f_k(x^k + \mu_k(x^k - x^{k-1})).$$

- 定义有关v^{k+1}的迭代式

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_k(x^k + \mu_k v^k),$$

- 于是得到关于x^k和v^k的等价迭代：

$$\begin{aligned} v^{k+1} &= \mu_k v^k - \alpha_k \nabla f_{s_k}(x^k + \mu_k v^k), \\ x^{k+1} &= x^k + v^{k+1}. \end{aligned}$$

- 二者的主要差别在梯度的计算上．Nesterov加速算法先对点施加速度的作用，再求梯度，可以理解为对标准动量方法做了校正．

AdaGrad

- 令$g^k = \nabla f_{s_k}(x^k)$，引入向量

$$G^k = \sum_{i=1}^{k} g^i \odot g^i.$$

G^k的每个分量是梯度在该分量处的累积平方和．当G^k的某分量较大时，该分量变化比较剧烈，因此应采用小步长，反之亦然．

- 因此AdaGrad的迭代格式为

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{G^k + \varepsilon \mathbf{1}_n}} \odot g^k, \tag{16}$$

$$G^{k+1} = G^k + g^{k+1} \odot g^{k+1}, \tag{17}$$

这里$\frac{\alpha}{\sqrt{G^k+\varepsilon \mathbf{1}_n}}$中的除法和求根运算都是对向量每个分量分别操作的（下同），α为初始步长，引入$\varepsilon \mathbf{1}_n$这一项是为了防止除零运算．

AdaGrad

- 可以看到AdaGrad的步长大致反比于历史梯度累计值的算术平方根，所以梯度较大时步长下降很快，反之则下降较慢，这样做的效果是在参数空间更平缓的方向上，前后两次迭代的距离较大.
- 在凸优化问题中AdaGrad有比较好的理论性质，但实际应用中也发现在训练深度神经网络模型时，从训练开始就积累梯度平方会导致步长过早或过多减小.
- 如果在AdaGrad中使用真实梯度$\nabla f(x^k)$，那么AdaGrad也可以看成是一种介于一阶和二阶的优化算法.
- 考虑$f(x)$在点x^k处的二阶泰勒展开：

$$f(x) \approx f(x^k) + \nabla f(x^k)^\top (x - x^k) + \frac{1}{2}(x - x^k)^\top B^k (x - x^k),$$

AdaGrad是使用一个对角矩阵来作为B^k:

$$B^k = \frac{1}{\alpha}\mathrm{Diag}(\sqrt{G^k + \varepsilon \mathbf{1}_n})$$

RMSProp

- RMSProp（root mean square propagation）是对AdaGrad的一个改进，该方法在非凸问题上可能表现更好. AdaGrad会累加之前所有的梯度分量平方，这就导致步长是单调递减的，因此在训练后期步长会非常小，计算的开销也较大.
- RMSProp提出只需使用离当前迭代点比较近的项，同时引入衰减参数ρ. 具体地，令

$$M^{k+1} = \rho M^k + (1-\rho) g^{k+1} \odot g^{k+1},$$

再对其每个分量分别求根，就得到均方根(root mean square)

$$R^k = \sqrt{M^k + \varepsilon \mathbf{1}_n}, \tag{18}$$

最后将均方根的倒数作为每个分量步长的修正.

RMSProp

- RMSProp迭代格式为：

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{M^k + \varepsilon \mathbf{1}_n}} \odot g^k, \tag{19}$$

$$M^{k+1} = \rho M^k + (1-\rho) g^{k+1} \odot g^{k+1}. \tag{20}$$

  引入参数ε同样是为了防止分母为0的情况发生. 一般取$\rho = 0.9$, $\alpha = 0.001$.

- 可以看到RMSProp 和AdaGrad 的唯一区别是将G^k替换成了M^k.

Adam

- Adam选择了一个动量项进行更新：

$$S^k = \rho_1 S^{k-1} + (1-\rho_1) g^k.$$

- 类似RMSProp，Adam也会记录梯度的二阶矩：

$$M^k = \rho_2 M^{k-1} + (1-\rho_2) g^k \odot g^k.$$

- 与原始动量方法和RMSProp的区别是，由于S^k和M^k本身带有偏差，Adam在更新前先对其进行修正：

$$\hat{S}^k = \frac{S^k}{1-\rho_1^k}, \quad \hat{M}^k = \frac{M^k}{1-\rho_2^k},$$

这里ρ_1^k, ρ_2^k分别表示ρ_1, ρ_2的k次方.

- Adam最终使用修正后的一阶矩和二阶矩进行迭代点的更新.

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{\hat{M}^k + \varepsilon \mathbf{1}_n}} \odot \hat{S}^k.$$

深度学习训练

- 数据准备
- 优化模型，网络架构
- 梯度计算：前向/后向传播
- 初始化
- 步长/学习率等参数调试
- 训练与测试，交叉检验
- 梯度消失，梯度爆炸
- 网络架构与算法的适配调整

Outline

随机变量收敛性与概率不等式

考虑随机变量序列$\{X_n\}_{n=1}^{\infty}$

- 几乎必然收敛(概率1收敛): $P(\lim_{n\to\infty} X_n = X) = 1$
- 依概率收敛: 对于任意$\epsilon > 0$, $\lim_{n\to\infty} P(|X_n - X| \geq \epsilon) = 0$
- 依分布收敛: 对于所有的α，$P(X_n \leq \alpha) \to P(X \leq \alpha)$
- 强大数定理
- 中心极限定理
- 马尔可夫不等式，切比雪夫不等式
- Hoeffding 不等式
- Azuma-Hoeffding 不等式

收敛性：凸函数

随机次梯度算法：$x^{k+1} = x^k - \alpha_k g^k, \quad g^k \in \partial f_{s_k}(x^k)$

- 每个$f_i(x)$是闭凸函数，存在次梯度且无偏，即$\mathbb{E}[g^k|x^k] \in \partial f(x^k)$
- 随机次梯度二阶矩是一致有界的，即存在M，对任意的$x \in \mathbb{R}^n$以及随机下标s_k，有

$$\mathbb{E}_{s_k}[\|g^k\|^2] \leq M^2 < +\infty, \quad g^k \in \partial f_{s_k}(x^k);$$

- $\{x^k\}$处处有界，即$\|x^k - x^*\| \leq R, \forall K$，其中$x^*$是问题的最优解.

引理

在上述假设下，令$\{\alpha_k\}$是任一正步长序列，$\{x^k\}$是由随机次梯度法产生的序列，那么对所有的$K \geq 1$，有

$$\sum_{k=1}^{K} \alpha_k \mathbb{E}[f(x^k) - f(x^*)] \leq \frac{1}{2}\mathbb{E}[\|x^1 - x^*\|^2] + \frac{1}{2}\sum_{k=1}^{K} \alpha_k^2 M^2. \tag{21}$$

引理的证明

- 令$\bar{g}^k = \mathbb{E}[g^k|x^k]$, $\xi^k = g^k - \bar{g}^k$.
- 由随机次梯度法的性质，

$$\bar{g}^k = \mathbb{E}[g^k|x^k] \in \partial f(x^k),$$

- 由次梯度的性质，

$$\langle \bar{g}^k, x^* - x^k \rangle \leq f(x^*) - f(x^k).$$

可以推导得

$$\begin{aligned}
&\|x^{k+1} - x^*\|^2 \\
=& \|x^k - \alpha_k g^k - x^*\|^2 \\
=& \|x^k - x^*\|^2 + 2\alpha_k \langle g^k, x^* - x^k \rangle + \alpha_k^2 \|g^k\|^2 \\
=& \|x^k - x^*\|^2 + 2\alpha_k \langle \bar{g}^k, x^* - x^k \rangle + \alpha_k^2 \|g^k\|^2 + 2\alpha_k \langle \xi^k, x^* - x^k \rangle \\
\leq& \|x^k - x^*\|^2 + 2\alpha_k (f(x^*) - f(x^k)) + \alpha_k^2 \|g^k\|^2 + 2\alpha_k \langle \xi^k, x^* - x^k \rangle.
\end{aligned} \tag{22}$$

- 注意到$\mathbb{E}[\xi^k|x^k]=0$，所以

$$\mathbb{E}[\langle \xi^k, x^*-x^k\rangle]=\mathbb{E}[\mathbb{E}[\langle \xi^k, x^*-x^k\rangle|x_k]]=0.$$

- 对不等式(22)两端求期望就得到

$$\alpha_k\mathbb{E}[f(x^k)-f(x^*)]\leq \frac{1}{2}\mathbb{E}[\|x^k-x^*\|^2]-\frac{1}{2}\mathbb{E}[\|x^{k+1}-x^*\|^2]+\frac{\alpha_k^2}{2}M^2. \quad (23)$$

- 两边对k求和即得证.

随机次梯度算法的收敛性1

$$\sum_{k=1}^{K}\alpha_k\mathbb{E}[f(x^k)-f(x^*)]\leq\frac{1}{2}\mathbb{E}[\|x^1-x^*\|^2]+\frac{1}{2}\sum_{k=1}^{K}\alpha_k^2M^2.$$

我们很容易得到随机次梯度算法在收缩步长下的收敛性.

定理 (随机次梯度算法的收敛性1)

在收敛性假设的条件下，令$A_K=\sum_{i=1}^{K}\alpha_i$，定义$\bar{x}_K=\frac{1}{A_K}\sum_{k=1}^{K}\alpha_k x^k$，则

$$\mathbb{E}[f(\bar{x}_K)-f(x^*)]\leq\frac{R^2+\sum_{k=1}^{K}\alpha_k^2M^2}{2\sum_{k=1}^{K}\alpha_k}. \tag{24}$$

定理的证明

- 由$f(x)$的凸性以及引理1得到

$$\begin{aligned} & A_k \mathbb{E}[f(\bar{x}_K) - f(x^*)] \\ \leq & \sum_{k=1}^{K} \alpha_k \mathbb{E}[f(x^k) - f(x^*)] \\ \leq & \frac{1}{2}\mathbb{E}[\|x^1 - x^*\|^2] + \frac{1}{2}\sum_{k=1}^{K} \alpha_k^2 M^2 \\ = & \frac{R^2 + \sum_{k=1}^{K} \alpha_k^2 M^2}{2} \end{aligned}$$

- 不等式两边同除以A_K得到

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{R^2 + \sum_{k=1}^{K} \alpha_k^2 M^2}{2A_K}.$$

随机次梯度算法的收敛性1

- 从定理1可以看到，当

$$\sum_{k=1}^{\infty} \alpha_k = +\infty, \quad \frac{\sum_{k=1}^{K} \alpha_k^2}{\sum_{k=1}^{K} \alpha_k} \to 0$$

时，随机次梯度算法收敛.

- 对一个固定的步长α，不等式(24)右侧有一个不随K递减的常数，因此固定步长随机次梯度算法在函数值取期望意义下是不收敛的，它仅仅能找到一个次优解：

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{R^2}{2K\alpha} + \frac{\alpha M^2}{2}.$$

特别地，对于给定的迭代次数K，选取固定步长$\alpha = \frac{R}{M\sqrt{K}}$，可以达到$\mathcal{O}(1/\sqrt{K})$的精度，即

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{RM}{\sqrt{K}}.$$

随机次梯度算法的收敛性2

在步长不增的情况下，我们可以得到直接平均意义下的收敛性.

定理 (随机次梯度算法的收敛性2)

在收敛性假设的条件下，令$\{\alpha_k\}$是一个不增的正步长序列，$\bar{x}_K = \frac{1}{K}\sum_{k=1}^{K} x^k$，则

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2. \tag{25}$$

定理的证明

- 对(23)式两边同除α_k，就有

$$\mathbb{E}[f(x^k)-f(x^*)] \leq \frac{1}{2\alpha_k}\mathbb{E}[\|x^k-x^*\|_2^2] - \frac{1}{2\alpha_k}\mathbb{E}[\|x^{k+1}-x^*\|_2^2] + \frac{\alpha_k}{2}M^2.$$

- 再对k求和，并且利用$f(x)$的凸性和α_k的单调性得

$$\begin{aligned}
\mathbb{E}[f(\bar{x}_K)-f(x^*)] &\leq \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[f(x^k)-f(x^*)] \\
&\leq \frac{1}{2K}\left(\frac{1}{\alpha_1}\mathbb{E}[\|x^1-x^*\|^2] + \sum_{k=1}^{K}\alpha_k M^2 + \right. \\
&\left. \sum_{k=2}^{K}\left(\frac{1}{\alpha_k}-\frac{1}{\alpha_{k-1}}\right)\mathbb{E}[\|x^k-x^*\|^2\right) \\
&\leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2.
\end{aligned}$$

随机次梯度算法的收敛性2

- 注意该定理和定理1 的不同之处在于$\bar{x}_K$的定义.
- 通过选取$\mathcal{O}(1/\sqrt{k})$阶数的步长，我们可以得到目标函数的收敛速度为$\mathcal{O}(1/\sqrt{k})$:

推论

在收敛性假设的条件下，令$\alpha_k = \frac{R}{M\sqrt{k}}$，则

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{3RM}{2\sqrt{K}}. \tag{26}$$

其中$\bar{x}_K$的定义和定理2 相同.

推论的证明

- 注意到

$$\sum_{k=1}^{K}\frac{1}{\sqrt{k}}\leq\int_{0}^{K}\frac{1}{\sqrt{t}}\mathrm{d}t=2\sqrt{K}.$$

- 将$\alpha_k=\frac{R}{M\sqrt{k}}$代入式(25)就得到

$$\mathbb{E}[f(\bar{x}_K)-f(x^*)]\leq\frac{R^2}{2K\frac{R}{M\sqrt{K}}}+\frac{RM}{2K}2\sqrt{K}=\frac{3RM}{2\sqrt{K}}.$$

- 我们可以发现随机次梯度算法和非随机次梯度算法具有相同的收敛速度——$\mathcal{O}(1/\sqrt{k})$.
- 随机次梯度算法每步的计算代价远小于非随机次梯度，这一定程度上解释了为什么随机算法在一些问题中的表现要远远好于非随机算法.

随机次梯度算法的收敛性3

下面主要讨论随机次梯度算法在依概率意义下的收敛性和收敛速度.

定理

选择上述推论中的步长α_k，使得$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \to 0$，那么我们有依概率收敛$f(\bar{x}_K) - f(x^*) \xrightarrow{P} 0\ (K \to \infty)$，即对任意的$\varepsilon > 0$，都有

$$\lim_{K\to\infty} P(f(\bar{x}_K) - f(x^*) \geq \varepsilon) = 0. \tag{27}$$

- 由马尔可夫不等式立即得到

$$P(f(\bar{x}_K) - f(x^*) \geq \varepsilon) \leq \frac{1}{\varepsilon}\mathbb{E}[f(\bar{x}_K) - f(x^*)] \to 0.$$

随机变量的收敛性

- 称随机变量序列$\{X_n\}_{n=1}^{\infty}$几乎必然收敛到X，如果

$$P\left(\lim_{n\to\infty} X_n = X\right) = 1;$$

- 称随机变量序列$\{X_n\}_{n=1}^{\infty}$依概率收敛到X，如果

$$\lim_{n\to\infty} P(|X_n - X| \geq \varepsilon) = 0.$$

定理 (随机次梯度算法的收敛性3)

在收敛性假设的条件下，进一步假设对于所有的随机次梯度g，有$\|g\| \leq M$. 那么对任意的$\varepsilon > 0$，

$$f(\bar{x}_K) - f(x^*) \leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2 + \frac{RM}{\sqrt{K}}\varepsilon \tag{28}$$

以大于等于$1 - e^{-\frac{1}{2}\varepsilon^2}$的概率成立，其中步长列$\{\alpha_k\}$是单调不增序列，$\bar{x}_K$的定义和定理2 中的定义相同.

鞅差序列

定义

设$\{X_n\}_{n=1}^{\infty}$与$\{Z_n\}_{n=1}^{\infty}$为$(\Omega, \mathcal{F}, P)$上的随机过程. 如果$\forall n \in \mathbb{N}_+$,

1. $\mathbb{E}[|X_n|] < +\infty$;
2. X_n属于$Z_1, \ldots, Z_n$生产的σ-代数;
3. $\mathbb{E}[X_{n+1}|Z_1, \ldots, Z_n] = 0$.

则称$\{X_n\}_{n=1}^{\infty}$为关于$\{Z_n\}_{n=1}^{\infty}$的鞅差序列. 特别地, 若$\{Z_n\}_{n=1}^{\infty} = \{X_n\}_{n=1}^{\infty}$, 则称$\{X_n\}_{n=1}^{\infty}$为鞅差序列.

- 鞅差序列的例子如: 设$\{Z_n\}_{n=1}^{\infty}$为一维简单随机游走, $\{X_n\}_{n=1}^{\infty}$定义为$X_n = Z_n - Z_{n-1}$(补充定义$Z_0 = 0$).
- 由一维简单随机游走的性质, X_n以$1/2$的概率为1, 以$1/2$的概率为-1. 因此$\mathbb{E}[|X_n|] = 1 < +\infty$;
- X_n由Z_{n-1}及Z_n完全决定, 因此属于$Z_1, \ldots, Z_n$生成的σ-代数;
- 由一维简单随机游走的性质, $X_{n+1} = Z_{n+1} - Z_n$独立于$Z_1, \ldots, Z_n$, 所以$\mathbb{E}[X_{n+1}|Z_1, \ldots, Z_n] = \mathbb{E}[X_{n+1}] = 0$.

Azuma-Hoeffding不等式

下面给出Azuma-Hoeffding不等式，即鞅版本的Hoeffding不等式：

引理

设$\{X_n\}_{n=1}^{\infty}$为鞅差序列，且$\forall n \in \mathbb{N}_+, |X_n| \leq B$. 则$\forall t > 0$，

$$P\left(\sum_{i=1}^{n} X_i \geq t\right) \leq \exp\left(-\frac{2t^2}{nB^2}\right);$$

$$P\left(\sum_{i=1}^{n} X_i \leq -t\right) \leq \exp\left(-\frac{2t^2}{nB^2}\right).$$

定理的证明

- 令$\bar{g}^k = \mathbb{E}[g^k|x^k]$, $\xi^k = g^k - \bar{g}^k$. 由(22)式的推导过程我们已经得到

$$f(x^k) - f(x^*) \leq \frac{1}{2\alpha_k}\|x^k - x^*\|^2 - \frac{1}{2\alpha_k}\|x^{k+1} - x^*\|^2 + \frac{\alpha_k}{2}\|g^k\|^2 + \langle \xi^k, x^* - x^k \rangle.$$

- 利用$f(x)$的凸性与α_k的单调性有

$$\begin{aligned} f(\bar{x}_K) - f(x^*) &\leq \frac{1}{K}\sum_{k=1}^{K} f(x^k) - f(x^*) \\ &\leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k\|g^k\|^2 + \frac{1}{K}\sum_{k=1}^{K}\langle \xi^k, x^* - x^k \rangle \\ &\leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2 + \frac{1}{K}\sum_{k=1}^{K}\langle \xi^k, x^* - x^k \rangle. \end{aligned}$$

- 令

$$\omega = \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2$$

得到

$$P(f(\bar{x}_K) - f(x^*) - \omega \geq t) \leq P\left(\frac{1}{K}\sum_{k=1}^{K}\langle \xi^k, x^* - x^k\rangle \geq t\right). \qquad (29)$$

- 设$Z^k = (x^1, x^2, \cdots, x^{k+1})$. 因为

$$\mathbb{E}[\xi^k | Z^{k-1}] = \mathbb{E}[\xi^k | x^k] = 0, \quad \mathbb{E}[x^k | Z^{k-1}] = x^k,$$

我们知道$\langle \xi^k, x^* - x^k\rangle$是一个鞅差序列.

- 同时由

$$\|\xi^k\|_2 = \|g^k - \overline{g}^k\|_2 \leq 2M$$

推出

$$|\langle \xi^k, x^* - x^k \rangle| \leq \|\xi^k\| \|x^* - x^k\|_2 \leq 2MR$$

即$\langle \xi_k, x^* - x_k \rangle$有界.

- 由Azuma-Hoeffding不等式得到

$$P\left(\frac{1}{K} \sum_{k=1}^{K} \langle \xi^k, x^* - x^k \rangle \geq t \right) \leq \exp\left(-\frac{Kt^2}{2M^2R^2} \right)$$

- 将$t = \frac{MR\varepsilon}{\sqrt{K}}$代入，有

$$P\left(\frac{1}{K} \sum_{k=1}^{K} \langle \xi^k, x^* - x^k \rangle \geq \frac{MR\varepsilon}{\sqrt{K}} \right) \leq \exp\left(-\frac{\varepsilon^2}{2} \right).$$

结合(29)式，定理得证.

随机次梯度算法的收敛性3

- 如果取$\alpha_k = \frac{R}{\sqrt{k}M}$，并令$\delta = e^{-\frac{1}{2}\varepsilon^2}$，就有

$$P\left(f(\bar{x}_K) - f(x^*) \leq \frac{3RM}{2\sqrt{K}} + \frac{RM\sqrt{2\ln 1/\delta}}{\sqrt{K}}\right) \geq 1 - \delta. \tag{30}$$

- 可以看到除一个很小的概率外，函数值以$\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$的速度收敛.

随机梯度算法的收敛性：强凸函数

1. $f(x)$是可微函数，每个$f_i(x)$梯度存在；
2. $f(x)$是梯度利普希茨连续的，相应常数为L；
3. $f(x)$是强凸函数，强凸参数为μ；
4. 随机梯度二阶矩是一致有界的，即存在M，对任意的$x \in \mathbb{R}^n$以及随机下标s^k，有

$$\mathbb{E}_{s_k}[\|\nabla f_{s_k}(x)\|^2] \leq M^2 < +\infty.$$

定理 (随机梯度算法的收敛性)

在收敛性假设的条件下，定义$\Delta_k = \|x^k - x^*\|$. 对固定的步长$\alpha_k = \alpha$, $0 < \alpha < \frac{1}{2\mu}$，有

$$\mathbb{E}[f(x^{K+1}) - f(x^*)] \leq \frac{L}{2}\mathbb{E}[\Delta_{K+1}^2] \leq \frac{L}{2}\left[(1-2\alpha\mu)^K \Delta_1^2 + \frac{\alpha M^2}{2\mu}\right]. \quad (31)$$

定理的证明

- 根据随机梯度算法的更新公式，

$$\begin{aligned}\Delta_{k+1}^2 &= \|x^{k+1} - x^*\|^2 = \|x^k - \alpha_k \nabla f_{s_k}(x^k) - x^*\|^2 \\ &= \|x^k - x^*\|^2 - 2\alpha_k \langle \nabla f_{s_k}(x^k), x^k - x^* \rangle + \alpha_k^2 \|\nabla f_{s_k}(x^k)\|^2 \\ &= \Delta_k^2 - 2\alpha_k \langle \nabla f_{s_k}(x^k), x^k - x^* \rangle + \alpha_k^2 \|\nabla f_{s_k}(x^k)\|^2,\end{aligned}$$

- 由条件期望的性质$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$，有

$$\begin{aligned}&\mathbb{E}_{s_1,s_2,\cdots,s_k}[\langle \nabla f_{s_k}(x^k), x^k - x^* \rangle] \\ =\ &\mathbb{E}_{s_1,s_2,\cdots,s_{k-1}}[\mathbb{E}_{s_k}[\langle \nabla f_{s_k}(x^k), x^k - x^* \rangle | s_1, \cdots, s_{k-1}]] \\ =\ &\mathbb{E}_{s_1,s_2,\cdots,s_{k-1}}[\langle \mathbb{E}_{s_k}[\nabla f_{s_k}(x_k) | s_1, s_2, \cdots, s_{k-1}], x^k - x^* \rangle] \\ =\ &\mathbb{E}_{s_1,s_2,\cdots,s_{k-1}}[\langle \nabla f(x^k), x^k - x^* \rangle] \\ =\ &\mathbb{E}_{s_1,s_2,\cdots,s_k}[\langle \nabla f(x^k), x^k - x^* \rangle].\end{aligned}$$

- 根据强凸函数的单调性，

$$\left\langle \nabla f(x^k), x^k - x^* \right\rangle = \left\langle \nabla f(x^k) - \nabla f(x^*), x^k - x^* \right\rangle \geq \mu \|x^k - x^*\|^2.$$

- 由随机梯度二阶矩的一致有界性，

$$\mathbb{E}_{s_1,s_2,\cdots,s_k}[\Delta_{k+1}^2] \leq (1 - 2\alpha\mu)\mathbb{E}_{s_1,s_2,\cdots,s_k}[\Delta_k^2] + \alpha^2 M^2. \tag{32}$$

- 对k做归纳，就得到

$$\mathbb{E}_{s_1,s_2,\cdots,s_K}[\Delta_{K+1}^2] \leq (1 - 2\alpha\mu)^K \Delta_1^2 + [1 - (1 - 2\alpha\mu)^K]\frac{\alpha M^2}{2\mu}.$$

由$0 < 2\alpha\mu < 1$可知

$$\mathbb{E}_{s_1,s_2,\cdots,s_K}[\Delta_{K+1}^2] \leq (1 - 2\alpha\mu)^K \Delta_1^2 + \frac{\alpha M^2}{2\mu}. \tag{33}$$

- 利用梯度L-利普希茨连续函数的二次上界，可以得到

$$f(x^{K+1}) - f(x^*) \leq \left\langle \nabla f(x^*), x^{K+1} - x^* \right\rangle + \frac{L}{2}\|x^{K+1} - x^*\|^2.$$

- 利用$\nabla f(x^*) = 0$并对上式左右两边取期望可得

$$\mathbb{E}[f(x^{K+1}) - f(x^*)] \leq \frac{L}{2}\mathbb{E}[\Delta_{K+1}^2] \leq \frac{L}{2}\left[(1 - 2\alpha\mu)^K \Delta_1^2 + \frac{\alpha M^2}{2\mu}\right].$$

随机梯度算法的收敛性

下面的定理表明，如果设置递减的步长，收敛阶可以达到$\mathcal{O}(1/K)$.

定理

随机梯度算法的收敛速度 在上述定理的结果中，在收敛性假设的条件下，取递减的步长

$$\alpha_k = \frac{\beta}{k+\gamma},$$

其中$\beta > \frac{1}{2\mu}, \gamma > 0$，使得$\alpha_1 \leq \frac{1}{2\mu}$，那么对于任意的$k \geq 1$，都有

$$\mathbb{E}[f(x^k) - f(x^*)] \leq \frac{L}{2}\mathbb{E}[\Delta_k^2] \leq \frac{L}{2}\frac{v}{\gamma+k}, \tag{34}$$

这里

$$v = \max\left\{\frac{\beta^2 M^2}{2\beta\mu - 1}, (\gamma+1)\Delta_1^2\right\}.$$

定理的证明

- 之前的定理已经证明了

$$\mathbb{E}_{s_1,s_2,\cdots,s_k}[\Delta_{k+1}^2] \leq (1-2\alpha_k\mu)\mathbb{E}_{s_1,s_2,\cdots,s_k}[\Delta_k^2] + \alpha_k^2 M^2.$$

- 用数学归纳法证明(34)式. 由v的定义知$k=1$时(34)式成立.
- 现假设该式对k成立，定义$\hat{k}=\gamma+k$，则$\alpha_k=\beta/\hat{k}$. 由归纳假设，

$$\begin{aligned}\mathbb{E}[\Delta_{k+1}^2] &\leq \left(1-\frac{2\beta\mu}{\hat{k}}\right)\frac{v}{\hat{k}}+\frac{\beta^2M^2}{\hat{k}^2}\\ &= \frac{\hat{k}-1}{\hat{k}^2}v-\frac{2\beta\mu-1}{\hat{k}^2}v+\frac{\beta^2M^2}{\hat{k}^2}\\ &\leq \frac{v}{\hat{k}+1}\end{aligned}$$

最后一个不等式用到了v的定义. 所以(34)式对$k+1$也成立.

随机梯度算法的收敛性

上述定理表明对于强凸函数，随机梯度下降法的收敛速度可以达到$\mathcal{O}(1/K)$. 对于一般的凸函数随机梯度算法也有一定的收敛性，为此我们在下表比较随机算法和普通算法的复杂度.

Table: 梯度下降法的算法复杂度

	f凸(次梯度算法)	f可微强凸	f可微强凸且L-光滑
随机算法	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$
普通算法	$\mathcal{O}\left(\frac{N}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{N}{\varepsilon}\right)$	$\mathcal{O}\left(N\ln\left(\frac{1}{\varepsilon}\right)\right)$

Outline

梯度下降法与随机梯度下降法的比较

下面分析梯度下降法与随机梯度下降法的主要区别：

- 在强凸性假设下，对梯度下降法有

$$\begin{aligned}\Delta_{k+1}^2 &= \|x^{k+1}-x^*\|^2 = \|x^k-\alpha\nabla f(x^k)-x^*\|^2 \\ &= \Delta_k^2 - 2\alpha\langle\nabla f(x^k), x^k-x^*\rangle + \alpha^2\|\nabla f(x^k)\|^2 \\ &\le (1-2\alpha\mu)\Delta_k^2 + \alpha^2\|\nabla f(x^k)\|_2^2 \qquad (\mu\text{-强凸}) \\ &\le (1-2\alpha\mu+\alpha^2L^2)\Delta_k^2. \qquad (L\text{-光滑})\end{aligned} \tag{35}$$

梯度下降法与随机梯度下降法的比较

- 对随机梯度下降法，利用条件期望的性质有

$$
\begin{aligned}
\mathbb{E}[\Delta_{k+1}^2] &= \mathbb{E}[\|x^{k+1}-x^*\|_2^2] = \mathbb{E}[\|x^k-\alpha\nabla f_{s_k}(x^k)-x^*\|]^2 \\
&= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle\nabla f_{s_k}(x^k), x^k-x^*\rangle] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2] \\
&= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle\nabla f(x^k), x^k-x^*\rangle] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2] \\
&\le (1-2\alpha\mu)\mathbb{E}[\Delta_k^2] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2] \qquad (\mu\text{-强凸}) \\
&= (1-2\alpha\mu)\mathbb{E}[\Delta_k^2] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f(x^k)+\nabla f(x^k)\|^2] \\
&\le \underbrace{(1-2\alpha\mu+\alpha^2L^2)\mathbb{E}[\Delta_k^2]}_{A} + \underbrace{\alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f(x^k)\|^2]}_{B}.
\end{aligned}
\tag{36}
$$

- 可以看到两种算法的主要差别就在B项上，也就是梯度估计的某种方差．它导致了随机梯度算法只能有$\mathcal{O}(1/k)$的收敛速度．

方差减小技术

- 在许多机器学习的应用中，随机梯度算法的收敛速度更快一些.
- 这主要是因为许多应用对解的精度要求不太高，而在开始部分方差较小，即有$B \ll A$，那么我们会观察到近似Q-线性收敛速度；
- 而随着迭代步数增多，方差增大，最终的收敛速度为$\mathcal{O}(1/k)$.
- 为了能获得比较快的渐进收敛速度，我们的主要目标即减少方差项B. 下面介绍三种减小方差的算法：
  - SAG (stochastic average gradient)
  - SAGA
  - SVRG (stochastic variance reduced gradient)

SAG算法

- 当迭代接近收敛时，上一步的随机梯度也是当前迭代点处梯度的一个很好的估计．随机平均梯度法(SAG)就是基于这一想法．
- 在迭代中，SAG算法记录所有之前计算过的随机梯度，再与当前新计算的随机梯度求平均，最终作为下一步的梯度估计．
- 具体来说，SAG算法在内存中开辟了存储N个随机梯度的空间

$$[g_1^k, g_2^k, \cdots, g_N^k],$$

分别用于记录和第i个样本相关的最新的随机梯度．在第k步更新时，若抽取的样本点下标为s_k，则计算随机梯度后将$g_{s_k}^k$的值更新为当前的随机梯度值，而其他未抽取到的下标对应的g_i^k保持不变．每次SAG 算法更新使用的梯度方向是所有g_i^k的平均值．

SAG算法

- SAG算法的迭代格式为

$$x^{k+1} = x^k - \frac{\alpha_k}{N} \sum_{i=1}^{N} g_i^k$$

其中g_i^k的更新方式为

$$g_i^k = \begin{cases} \nabla f_{s_k}(x^k), & i = s_k, \\ g_i^{k-1}, & \text{其他}, \end{cases} \tag{37}$$

- 每次迭代只有一个g_i^k发生了改变. 因此SAG迭代公式还可以写成

$$x^{k+1} = x^k - \alpha_k \left(\frac{1}{N}(\nabla f_{s_k}(x^k) - g_{s_k}^{k-1}) + \frac{1}{N} \sum_{i=1}^{N} g_i^{k-1} \right), \tag{38}$$

- $\{g_i^k\}$的初值可简单地取为0或中心化的随机梯度向量,
- SAG算法每次使用的随机梯度的条件期望并不是真实梯度$\nabla f(x^k)$, 但随着迭代进行, 随机梯度的期望和真实梯度的偏差会越来越小.

SAGA算法

- SAGA算法的迭代方式为

$$x^{k+1} = x^k - \alpha_k \left(\nabla f_{s_k}(x^k) - g_{s_k}^{k-1} + \frac{1}{N} \sum_{i=1}^{N} g_i^{k-1} \right). \tag{39}$$

- 对比(38)式可以发现，SAGA算法去掉了$\nabla f_{s_k}(x^k) - g_{s_k}^{k-1}$前面的系数$1/N$. 可以证明每次迭代使用的梯度方向都是无偏的，即

$$\mathbb{E}\left[\nabla f_{s_k}(x^k) - g_{s_k}^{k-1} + \frac{1}{N} \sum_{i=1}^{N} g_i^{k-1} \,\middle|\, x^k \right] = \nabla f(x^k).$$

SAGA算法的收敛性

SAGA算法同样有Q-线性收敛速度：

定理 (SAGA算法的收敛性)

在强凸性收敛性假设的条件下，取固定步长$\alpha_k = \frac{1}{2(\mu N+L)}$. 定义$\Delta_k = \|x^k - x^*\|$，则对任意的$k \geq 1$有

$$\mathbb{E}[\Delta_k^2] \leq \left(1 - \frac{\mu}{2(\mu N + L)}\right)^k \left(\Delta_1^2 + \frac{N(f(x^1) - f(x^*))}{\mu N + L}\right). \tag{40}$$

如果强凸的参数μ是未知的，也可以取$\alpha = \frac{1}{3L}$，有类似的收敛结果.

SVRG算法

- 与SAG算法和SAGA算法不同，SVRG算法通过周期性缓存全梯度的方法来减小方差.
- 具体做法是在随机梯度下降方法中，每经过m次迭代就设置一个检查点，计算一次全梯度，在之后的m次迭代中，将这个全梯度作为参考点来达到减小方差的目的.
- 令$\tilde{x}^j$是第j个检查点，则我们需要计算点$\tilde{x}^j$处的全梯度

$$\nabla f(\tilde{x}^j) = \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\tilde{x}^j),$$

在之后的迭代中使用方向v^k作为更新方向：

$$v^k = \nabla f_{s_k}(x^k) - (\nabla f_{s_k}(\tilde{x}^j) - \nabla f(\tilde{x}^j)), \tag{41}$$

其中$s_k \in \{1, 2, \cdots, N\}$是随机选取的一个样本.

SVRG算法

- 注意到给定$s_1, s_2, \cdots, s_{k-1}$时$x^k, \tilde{x}^j$均为定值，由$v^k$的表达式可知

$$\begin{aligned}
&\mathbb{E}[v^k | s_1, s_2, \cdots, s_{k-1}] \\
=& \mathbb{E}[\nabla f_{s_k}(x^k) | x^k] - \mathbb{E}[\nabla f_{s_k}(\tilde{x}^j) - \nabla f(\tilde{x}^j) | s_1, s_2, \cdots, s_{k-1}] \\
=& \nabla f(x^k) - 0 = \nabla f(x^k),
\end{aligned}$$

- 公式(41)有简单的直观理解：我们希望用$\nabla f_{s_k}(\tilde{x}^j)$去估计$\nabla f(\tilde{x}^j)$，那么$\nabla f_{s_k}(\tilde{x}^j) - \nabla f(\tilde{x}^j)$就可以看作梯度估计的误差，所以在每一步随机梯度迭代用该项来对$\nabla f_{s_k}(x^k)$做一个校正.

SVRG算法

- 假设

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|, \quad i = 1, 2, \cdots, N.$$

- 令$y = \tilde{x}^j$，x^*为$f(x)$的最小值点，$\Delta_k = \|x^k - x^*\|$，则

$$\begin{aligned}
&\mathbb{E}\left[\|v^k\|^2\right] = \mathbb{E}\left[\|\nabla f_{s_k}(x^k) - (\nabla f_{s_k}(y) - \nabla f(y))\|^2\right] \\
&= \mathbb{E}\left[\|\nabla f_{s_k}(x^k) - \nabla f_{s_k}(y) + \nabla f(y) + \nabla f_{s_k}(x^*) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leq 2\mathbb{E}\left[\|\nabla f_{s_k}(x^k) - \nabla f_{s_k}(x^*)\|^2\right] + 2\mathbb{E}\left[\|\nabla f_{s_k}(y) - \nabla f(y) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leq 2L^2\mathbb{E}\left[\Delta_k^2\right] + 2\mathbb{E}\left[\|\nabla f_{s_k}(y) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leq 2L^2\mathbb{E}\left[\Delta_k^2\right] + 2L^2\mathbb{E}\left[\|y - x^*\|^2\right].
\end{aligned} \tag{42}$$

- 其中第一个不等式是因为$\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$，第二个不等式使用了有关二阶矩的不等式

$$\mathbb{E}[\|\xi - \mathbb{E}\xi\|^2] \leq \mathbb{E}[\|\xi\|^2].$$

SVRG算法的收敛性

下面给出SVRG算法的收敛性. 这里的收敛性是针对参考点序列$\{\tilde{x}^j\}$而言的.

定理 (SVRG算法的收敛性)

设m为利用每个$\tilde{x}^j$更新的次数. 设每个$f_i(x)$可微, 且梯度L-利普希茨连续; 函数$f(x)$强凸, 强凸参数为μ. 取步长$\alpha \in \left(0, \frac{1}{2L}\right]$, 并且$m$充分大使得

$$\rho = \frac{1}{\mu\alpha(1-2L\alpha)m} + \frac{2L\alpha}{1-2L\alpha} < 1, \tag{43}$$

则*SVRG*算法对于参考点$\tilde{x}^j$在函数值期望的意义下有Q-线性收敛速度:

$$\mathbb{E}f(\tilde{x}^j) - f(x^*) \leq \rho\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)]. \tag{44}$$

定理的证明

- 定义$\Delta_k = \|x^k - x^*\|$.
- 对于内层循环，

$$\begin{aligned}\mathbb{E}[\Delta_{k+1}^2] &= [\|x^{k+1} - x^*\|^2] = \mathbb{E}[\|x^k - \alpha v^k - x^*\|^2] \\ &= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle v^k, x^k - x^*\rangle] + \alpha^2\mathbb{E}[\|v^k\|^2] \\ &= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle \nabla f(x^k), x^k - x^*\rangle] + \alpha^2\mathbb{E}[\|v^k\|^2] \\ &\leq \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[(f(x^k) - f(x^*))] + \alpha^2\mathbb{E}[\|v^k\|^2].\end{aligned}$$

- 构造辅助函数

$$\phi_i(x) = f_i(x) - f_i(x^*) - \nabla f_i(x^*)(x - x^*),$$

注意到$\phi_i(x)$也是凸函数且梯度L-利普希茨连续，因此有

$$\frac{1}{2L}\|\nabla\phi_i(x)\|^2 \leq \phi_i(x) - \phi_i(x^*)$$

- 展开$\phi_i(x)$与$\nabla\phi_i(x)$的表达式可得

$$\|\nabla f_i(x)-\nabla f_i(x^*)\|^2 \le 2L[f_i(x)-f_i(x^*)-\nabla f_i(x^*)^\top(x-x^*)].$$

- 对i从1到N进行求和，注意$\nabla f(x^*)=0$：

$$\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(x)-\nabla f_i(x^*)\|^2 \le 2L[f(x)-f(x^*)], \quad \forall x. \tag{45}$$

- 利用(42)式的推导过程可得

$$\mathbb{E}[\|v^k\|^2] \le 2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f_{s_k}(x^*)\|^2]+2\mathbb{E}[\|\nabla f_{s_k}(\tilde{x}^{j-1})-\nabla f_{s_k}(x^*)\|^2].$$

对上式右侧第一项，有

$$\begin{aligned}
&\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f_{s_k}(x^*)\|^2] \\
=\ &\mathbb{E}[\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f_{s_k}(x^*)\|^2|s_1,s_2,\cdots,s_{k-1}]] \\
=\ &\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(x^k)-\nabla f_i(x^*)\|^2\right] \le 2L\mathbb{E}[f(x^k)-f(x^*)],
\end{aligned}$$

- 类似地，对右侧第二项，有

$$\mathbb{E}[\|\nabla f_{s_k}(\tilde{x}^{j-1}) - \nabla f_{s_k}(x^*)\|^2] \leq 2L\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)].$$

- 最终可得对$\mathbb{E}[\|v^k\|^2]$的估计：

$$\mathbb{E}[\|v^k\|^2] \leq 4L(\mathbb{E}[f(x^k) - f(x^*)] + \mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)]).$$

- 将$\mathbb{E}[\|v^k\|^2]$的上界代入对$\mathbb{E}[\Delta_{k+1}^2]$的估计，就有

$$\begin{aligned}\mathbb{E}[\Delta_{k+1}^2] &\leq \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[f(x^k) - f(x^*)] + \alpha^2\mathbb{E}[\|v^k\|^2] \\ &\leq \mathbb{E}[\Delta_k^2] - 2\alpha(1 - 2\alpha L)\mathbb{E}[f(x^k) - f(x^*)] \\ &\quad + 4L\alpha^2\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)].\end{aligned}$$

- 对k从1到m求和，并且注意到$x^1 = \tilde{x}^{j-1}$就可以得到

$$\begin{aligned}
&\mathbb{E}[\Delta_{m+1}^2] + 2\alpha(1-2\alpha L)\sum_{k=1}^{m}\mathbb{E}[f(x^k) - f(x^*)] \\
&\leq \mathbb{E}[\|\tilde{x}^{j-1} - x^*\|^2] + 4L\alpha^2 m\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)] \\
&\leq \frac{2}{\mu}\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)] + 4L\alpha^2 m\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)],
\end{aligned}$$

- 注意到$\tilde{x}^j = \frac{1}{m}\sum_{k=1}^{m} x^k$，所以

$$\begin{aligned}
&\mathbb{E}[f(\tilde{x}^j) - f(x^*)] \\
&\leq \frac{1}{m}\sum_{k=1}^{m}\mathbb{E}[f(x^k) - f(x^*)] \\
&\leq \frac{1}{2\alpha(1-2\alpha L)m}\left(\frac{2}{\mu} + 4mL\alpha^2\right)\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)] \\
&= \rho\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)].
\end{aligned}$$

Outline

常见记号约定

记号	含义
$\mathcal{S}$	数据标签集$\{x_i, y_i\}_{i=1}^N$
$\mathcal{S}_x$	数据集$\{x_i\}_{i=1}^N$
$\mathcal{S}_y$	标签集$\{y_i\}_{i=1}^N$
$Q_{x,y}$	真实数据与标签的分布函数
Q_x	真实数据的分布函数
$Q_{y\|x}$	给定数据x后标签y的分布函数
$q(x,y)$	$Q_{x,y}$的概率密度函数
$q(x)$	Q_x的概率密度函数
$q(y\|x)$	$Q_{y\|x}$的概率密度函数
$P_{x,y}(\theta)$	实际训练学习到的分布函数
$P_{y\|x}(\theta)$	给定参数θ与数据x后标签y的分布函数
$p(x,y\|\theta)$	$P_{x,y}(\theta)$的概率密度函数
$p(y\|x,\theta)$	$P_{y\|x}(\theta)$的概率密度函数
$R_{y\|z}$	给定神经网络输出z后标签y的分布函数
$r_{y\|z}$	$R_{y\|z}$的概率密度函数

- 选择用KL散度来刻画学习到的分布函数$P_{x,y}(\theta)$与真实数据与标签的分布函数$Q_{x,y}$的距离

$$\begin{aligned}\mathrm{KL}(Q_{x,y}\|P_{x,y}(\theta)) &:= \int q(x,y)\log\frac{q(x,y)}{p(x,y|\theta)}dxdy\\ &= \int q(x,y)\log\frac{q(y|x)q(x)}{p(y|x,\theta)q(x)}dxdy\\ &= \int q(x)\int q(y|x)\log\frac{q(y|x)}{p(y|x,\theta)}dxdy\\ &= \mathbb{E}_{Q_x}[\mathrm{KL}(Q_{y|x}\|P_{y|x}(\theta))].\end{aligned}$$

- 实际情况用数据集的分布近似真正数据的分布

$$\mathbb{E}_{\hat{Q}_x}[\mathrm{KL}(\hat{Q}_{y|x}\|P_{y|x})] \propto -\frac{1}{|\mathcal{S}_x|}\sum_{(x,y)\in\mathcal{S}}\log(p(y|x,\theta)),$$

其中$\propto$表示在θ固定的情况并且省略常数的情况下左右两端关于θ成固定比例.

优化问题

- 优化问题：

$$\min_{\theta} \quad -\frac{1}{|\mathcal{S}_x|} \sum_{(x,y)\in\mathcal{S}} \log(p(y|x,\theta)). \tag{46}$$

- 设神经网络的输出为$h_\theta(x)$，并且将$P_{y|x}(\theta)$表示为神经网络输出$h_\theta(x)$与另一个概率分布输出$R_{y|h_\theta(x)}$的复合函数，即$P_{y|x}(\theta) = R_{y|h_\theta(x)}$.设$R_{y|h_\theta(x)}$的概率密度函数为$r(y|h_\theta(x))$，则对于任意的损失函数$\ell(y, h_\theta(x))$，可以定义$r(y|h_\theta(x)) \propto \exp(-\ell(y, h_\theta(x)))$.

- 如果设$|\mathcal{S}_x| = N$，$\ell(y, h_\theta(x)) \propto -\log(r(y|h_\theta(x))) = -\log(p(y|x,\theta))$，则与之前的优化模型一致．极小化经验风险等价于求解关于概率密度函数$p(y|x,\theta)$关于参数θ的极大似然估计问题．可以证明若$R_{y|h_\theta(x)}$为高维独立高斯分布，则其对应的损失函数为平方误差损失函数$\ell(y, h_\theta(x)) = \|h_\theta(x) - y\|^2$.

费希信息矩阵

- 对于概率分布函数$P_{x,y}(\theta)$，其费希信息矩阵定义为：

$$\mathbf{F} = \mathbb{E}_{P_{x,y}(\theta)}\left[\nabla \log p(x,y|\theta) \nabla \log p(x,y|\theta)^{\mathrm{T}}\right], \tag{47}$$

其中∇为对θ求梯度.

- 根据等式

$$\nabla \log p(x,y|\theta) = \nabla \log p(y|x,\theta) + \nabla \log q(x) = \nabla \log p(y|x,\theta)$$

所以FIM 也可以写成

$$\mathbf{F} = \mathbb{E}_{Q_x}\left[\mathbb{E}_{P_{y|x}(\theta)}\left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^{\mathrm{T}}\right]\right].$$

- 定义函数$\log p(y|x,\theta)$关于θ的海瑟矩阵为$\mathbf{H}_{\log p(y|x,\theta)}$，则有：

$$\mathbb{E}_{P_{x,y}(\theta)}\left[\mathbf{H}_{\log p(y|x,\theta)}\right] = -\mathbf{F}. \tag{48}$$

自然梯度方向

- 对于任意损失函数$\nabla f(\theta)$梯度方向具有下面的性质：

$$\frac{-\nabla f(\theta)}{\|\nabla f(\theta)\|} = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \operatorname*{argmin}_{d} f(\theta + d), \quad \text{s.t. } \|d\| \leq \varepsilon. \tag{49}$$

- 对于*KL*散度约束，我们有：

$$-\sqrt{2}\frac{\mathbf{F}^{-1}\nabla f}{\|\nabla f\|_{\mathbf{F}^{-1}}} = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \operatorname*{argmin}_{\mathrm{KL}[P_{x,y}(\theta)\|P_{x,y}(\theta+d)]\leq\varepsilon^2} f(\theta + d). \tag{50}$$

- 经验费希信息矩阵$\hat{\mathbf{F}}$定义为

$$\begin{aligned}\hat{\mathbf{F}}(\theta) &= \mathbb{E}_{\hat{Q}_{x,y}}\left[\nabla \log p(y|x,\theta) \nabla \log p(y|x,\theta)^{\mathrm{T}}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} \nabla \log p(y_i|x_i,\theta) \nabla \log p(y_i|x_i,\theta)^{\mathrm{T}}.\end{aligned} \tag{51}$$

自然梯度法

Algorithm 1 自然梯度法

1: 输入：目标函数f，初始参数$\theta^0, k=0$.
2: **while** 未达到收敛准则 **do**
3: 计算梯度$\nabla_\theta f(\theta^k)$.
4: 计算费希信息矩阵$\mathbf{F}_k$(或者经验费希矩阵$\hat{\mathbf{F}}_k$).
5: 计算自然梯度方向$\tilde{\nabla}_\theta f(\theta^k) = \mathbf{F}_k^{-1}\nabla f(\theta)$(或者$\hat{\mathbf{F}}_k^{-1}\nabla f(\theta)$).
6: 更新参数$\theta^{k+1} = \theta^k - \alpha_k \tilde{\nabla}_\theta f(\theta^k)$，其中$\alpha_k$为第$k$步的步长.
7: $k = k+1$.
8: **end while**

KFAC

- 考虑ℓ层前馈全连接神经网络，设s_i为第i层经过权重矩阵作用后的输出，a_i为经过第i层激活函数ϕ_i的输出：

$$s_i = W_i \bar{a}_{i-1}, \quad a_i = \phi(s_i), \tag{52}$$

其中$i \in \{1, \cdots, \ell\}$, ϕ_i为第i层的激活函数, W_i为权重矩阵.

- 定义神经网络的参数
为$\theta = [\mathrm{vec}(W^{(1)})^{\mathrm{T}}, \mathrm{vec}(W^{(2)})^{\mathrm{T}}, \cdots, \mathrm{vec}(W^{(l)})^{\mathrm{T}}]$.
设$g^{(l)} = \frac{\partial \ell(y,\hat{y})}{\partial z^{(l)}} \cdot (a^{(l-1)})^{\mathrm{T}}$.

$$\mathcal{D}W^{(i)} := \frac{\partial \ell(y, \hat{y})}{\partial W^{(i)}} = g^{(i)} (\bar{a}^{(i-1)})^{\mathrm{T}}. \tag{53}$$

因此有$\mathcal{D}\theta = [\mathrm{vec}(\mathcal{D}W^{(1)})^{\mathrm{T}}, \mathrm{vec}(\mathcal{D}W^{(2)})^{\mathrm{T}}, \cdots, \mathrm{vec}(\mathcal{D}W^{(l)})^{\mathrm{T}}]^{\mathrm{T}}$.

- 由此可知费希信息矩阵的表达式为：

$$\mathbf{F} = \mathbb{E}[\mathcal{D}\theta \mathcal{D}\theta^{\mathrm{T}}] = [\mathbf{F}_{i,j}].$$

Kronecker product

- $A \otimes B$ denotes the Kronecker product between A and B:

$$A \otimes B \equiv \begin{bmatrix} [A]_{1,1}B & \cdots & [A]_{1,n}B \\ \vdots & \ddots & \vdots \\ [A]_{m,1}B & \cdots & [A]_{m,n}B \end{bmatrix}.$$

- $\operatorname{vec}(uv^\top) = v \otimes u$.
- $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.
- $(B^\top \otimes A)\operatorname{vec}(X) = \operatorname{vec}(AXB)$
- $\operatorname{vec}(G_i A_i^\top) = (A_i \otimes G_i)\operatorname{vec}(I)$.
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ for any A, B, C, D with correct sizes.

- 近似：

$$\begin{aligned}\mathbf{F}_{i,j} &= \mathbb{E}[\mathrm{vec}(\mathcal{D}W^{(i)})\mathrm{vec}(\mathcal{D}W^{(j)})^{\mathrm{T}}] \\ &= \mathbb{E}[\bar{a}^{(i-1)}(\bar{a}^{(j-1)})^{\mathrm{T}} \otimes g^{(i)}(g^{(j)})^{\mathrm{T}}] \\ &\approx \mathbb{E}[\bar{a}^{(i-1)}(\bar{a}^{(j-1)})^{\mathrm{T}}] \otimes \mathbb{E}[g^{(i)}(g^{(j)})^{\mathrm{T}}] \\ &= \bar{A}_{i-1.j-1} \otimes G_{i,j} := \tilde{\mathbf{F}}_{i,j},\end{aligned}$$

其中$\tilde{\mathbf{F}}_{i,j}$是$\mathbf{F}_{i,j}$的近似，$\bar{A}_{i,j} := \mathbb{E}[\bar{a}^{(i)}(\bar{a}^{(j)})^{\mathrm{T}}], G_{i,j} := \mathbb{E}[g^{(i)}(g^{(j)})^{\mathrm{T}}]$. 第三个关系式是用到了交换的性质 .

- 由此得到第k步θ^{k+1}的更新方式为

$$\begin{aligned}\theta_i^{k+1} &= \theta_i^k - \alpha_k \hat{\mathbf{F}}_{ii}^{-1} g_k \\ &= \theta_i^k - \alpha_k (\bar{A}_{i-1,i-1} + \sqrt{\lambda} I)^{-1} \otimes (G_{i,i} + \sqrt{\lambda} I)^{-1} g_k\end{aligned}$$

其中g_k为第k步目标函数的梯度，α_k为第k步的步长