Optimization Methods for Learning

张海鹏

University of JiangNan

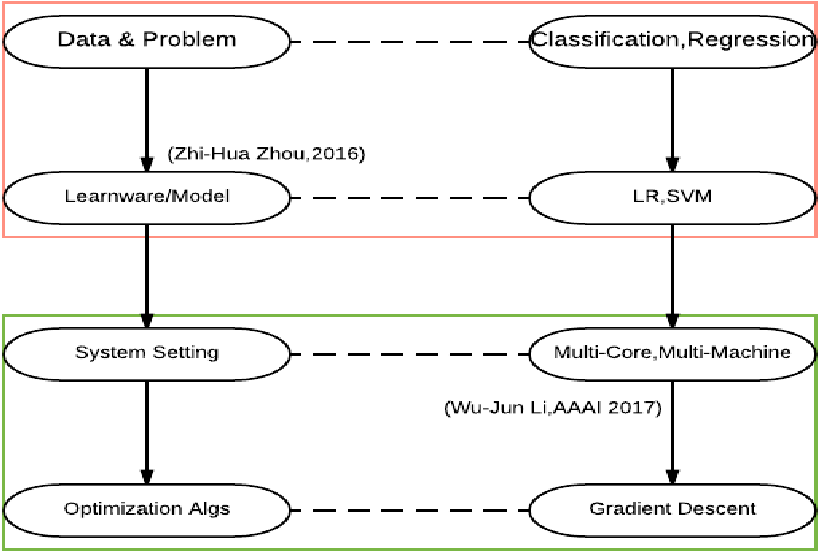
zhpmatrix@gmail.com

2017 年 4 月 20 日

Outline

- Optimization Methods for LR and Distributed Implementation
- Roadmap of Optimization Improvement
- Optimization Methods for Deep Learning
- Ideas
- PSO v.s. Gradient Optimization

Distribution Optimization:SGD-> HogWild!



Linear Regression-(BGD,SGD,mini-BGD)(1/3)

Model:

$$H_\theta(X) = \sum_{j=0}^{N} \theta_j X_j \tag{1}$$

Loss Function:

$$J(\theta) = \frac{1}{2M}\sum_{i=1}^{M}(H_\theta(X^{(i)}) - Y^{(i)})^2 = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{2}(H_\theta(X^{(i)}) - Y^{(i)})^2 \tag{2}$$

Diff:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{M}\sum_{i=1}^{M}(H_\theta(X^{(i)}) - Y^{(i)})X_j^{(i)} \tag{3}$$

(T.Hastie,R.Tibshirani,J.Friedman,ESL,2.3.1,Linear Models and Least Squares)

BGD:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \eta \frac{1}{M} \sum_{i=1}^{M} (H_{\theta^{(t)}}(X^{(i)}) - Y^{(i)}) X_j^{(i)} \quad (4)$$

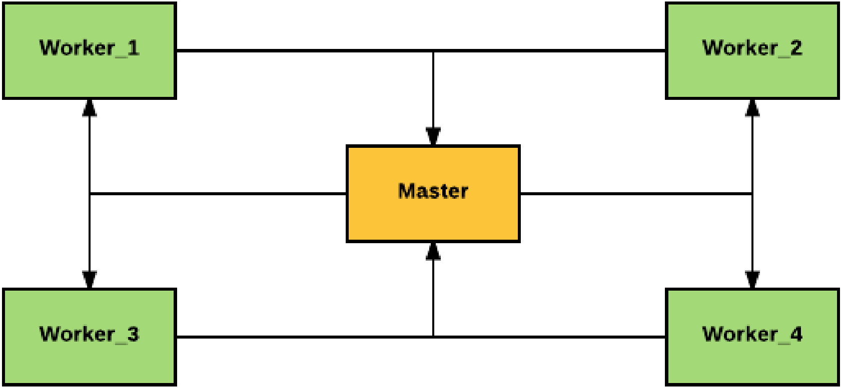
SGD:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \eta (H_{\theta^{(t)}}(X^{(i)}) - Y^{(i)}) X_j^{(i)} \quad (5)$$

mini-BGD:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \eta \frac{1}{m} \sum_{i=1}^{m} (H_{\theta^{(t)}}(X^{(i)}) - Y^{(i)}) X_j^{(i)} \quad (0 < m < M) \quad (6)$$

Linear Regression-(BGD,SGD,mini-BGD)(3/3)

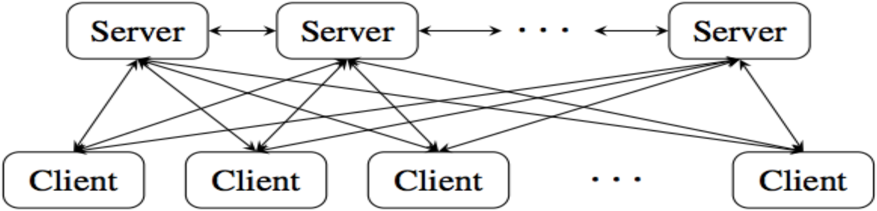


Hogwild!

Each thread draws a random example *i* from training data.

- ~~Acquire a lock on the current state of parameters $\theta^{(t)}$.~~
- Thread reads $\theta^{(t)}$.
- Thread updates $\theta^{(t+1)} = \theta^{(t)} - \eta(H_{\theta^{(t)}}(X^{(i)}) - Y^{(i)})X^{(i)}$.
- ~~Release lock on $\theta^{(t)}$.~~

Distribution-Performance



- Convergence
- Complexity
- Communication:throughput,latency

Questions

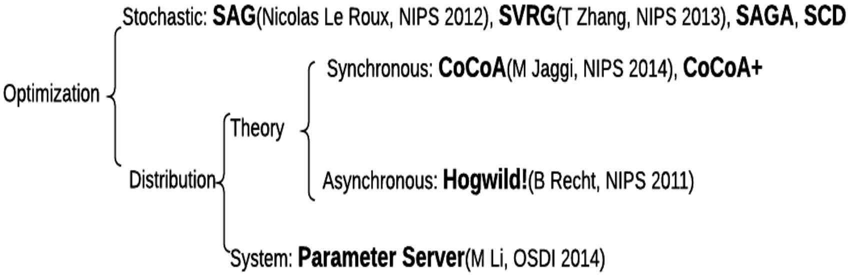
1.How to trade off accuracy and convergence?
BGD($O(\rho^T)$),SGD($O(\frac{1}{T})$).

2. How to choose learning rate?
Fixed and Diminishing Stepsize.

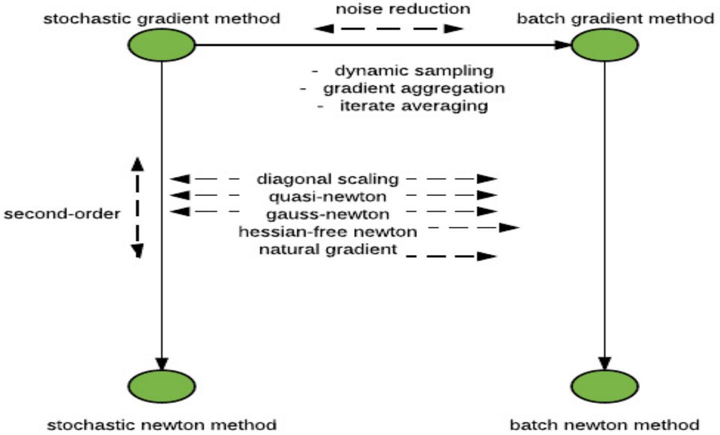
3. How to escape saddle point in non-convex problem?
$z = x^2 - y^2$

4.How to make communication efficient in distribution setting?

Outline(1/3)-Overall

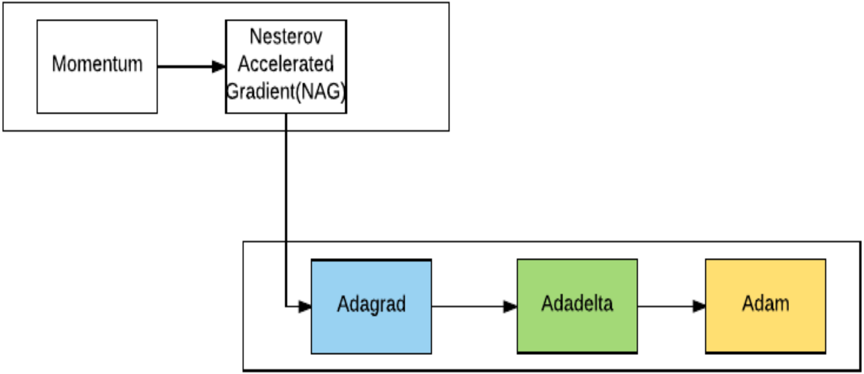


Outline(2/3)-SGDs



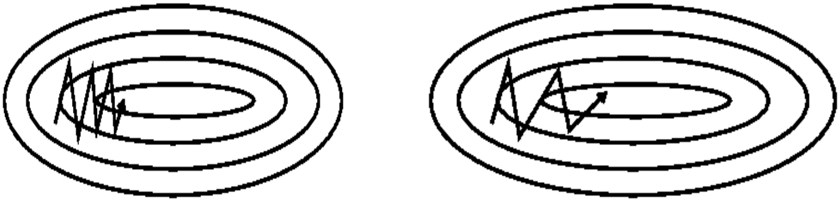
《Optimization Methods for Large-Scale Machine Learning》(Bottou L, Arxiv,2016)

Outline(3/3)-DL

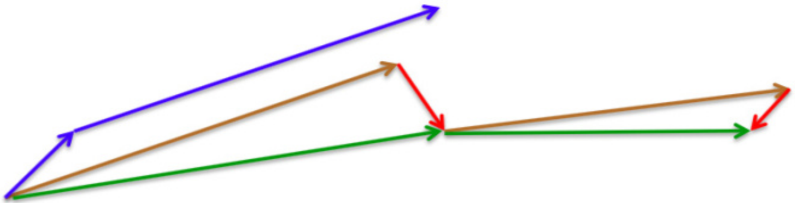


《An Overview of Gradient Descent Optimization Algorithms》(Sebastian Ruder, Arxiv, 2016)

Momentum v.s. NAG(1/2)



(Source: Genevieve B. Orr)



(Source: G. Hinton's lecture)

Momentum v.s. NAG(2/2)

Momentum:

$$\boldsymbol{\nu_t = \gamma\nu_{t-1} + \eta\nabla_{\theta^{(t-1)}} J(\theta^{(t-1)})} \tag{7}$$

$$\boldsymbol{\theta^{(t)} = \theta^{(t-1)} - \nu_t} \tag{8}$$

NAG:

$$\boldsymbol{\theta' = \theta^{(t-1)} - \gamma\nu_{t-1}} \tag{9}$$

$$\boldsymbol{\nu_t = \gamma\nu_{t-1} + \eta\nabla J_{\theta'}(\theta')} \tag{10}$$

$$\boldsymbol{\theta^{(t)} = \theta^{(t-1)} - \nu_t} \tag{11}$$

Ada Algs(1/3)-AdaGrad

$g_{t,i} = \nabla_\theta J(\theta_i)$ 表示目标函数在第 t 步中在 θ_i 的梯度，则对传统 SGD 有:

$$\boldsymbol{\theta_{t+1,i} = \theta_{t,i} - \eta g_{t,i}} \tag{12}$$

而 AdaGrad 的更新方式为:

$$\boldsymbol{\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}} \tag{13}$$

其中，$\boldsymbol{G_t \in R^{dxd}}$ 是一个对角矩阵，对角上的元素 $\boldsymbol{i}$ 为 $\boldsymbol{\theta_i}$ 的历史值的平方和，$\boldsymbol{\epsilon}$ 是为了防止分母为 0 的项，通常取值为 $\mathbf{1e-8}$，$\boldsymbol{\eta}$ 通常不需要调整，默认值 $\mathbf{0.01}$。

Adadelta 对于 Adagrad 中的改进主要是采用一个固定窗口内的值的平方和的平均。第 t 步对应的值为:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \tag{14}$$

同时有:

$$\triangle\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{15}$$

用新的记号 $RMS[g]_t$ 来重写上述分母，有:

$$\triangle\theta_t = -\frac{\eta}{RMS[g]_t} g_t \tag{16}$$

针对 η 的改进是:

$$\eta = RMS[\triangle\theta]_{t-1} = \sqrt{E[\triangle\theta^2]_{t-1} + \epsilon} \tag{17}$$

最终的表达式是:

$$\triangle\theta_t = -\frac{RMS[\triangle\theta]_{t-1}}{RMS[g]_t} g_t \tag{18}$$

$$\theta_{t+1} = \theta_t + \triangle\theta_t \tag{19}$$

Adaptive Moment Estimation:

$$\boldsymbol{m_t} = \boldsymbol{\beta_1 m_{t-1}} + (\boldsymbol{1} - \boldsymbol{\beta_1})\boldsymbol{g_t} \tag{20}$$

$$\boldsymbol{\nu_t} = \boldsymbol{\beta_2 \nu_{t-1}} + (\boldsymbol{1} - \boldsymbol{\beta_2})\boldsymbol{g_t^2} \tag{21}$$

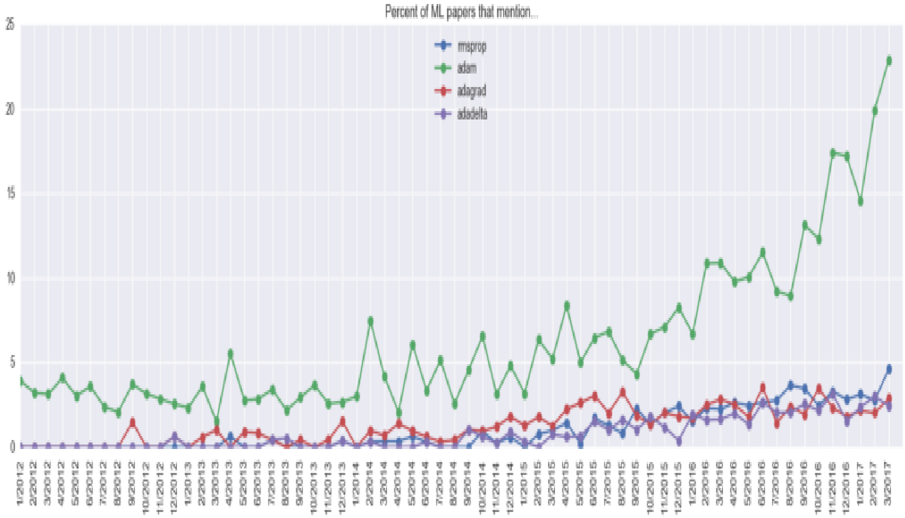
$$\boldsymbol{\hat{m}_t} = \frac{\boldsymbol{m_t}}{\boldsymbol{1} - \boldsymbol{\beta_1^t}} \tag{22}$$

$$\boldsymbol{\hat{\nu}_t} = \frac{\boldsymbol{\nu_t}}{\boldsymbol{1} - \boldsymbol{\beta_2^t}} \tag{23}$$

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \frac{\boldsymbol{\eta}}{\sqrt{\boldsymbol{\hat{\nu}_t}} + \boldsymbol{\epsilon}}\boldsymbol{\hat{m}_t} \tag{24}$$

其中，$\boldsymbol{\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}}$

Trend



(Source: Andrej Karpathy,2017@Medium)

ML 中各种基于 gradient 优化的 variants 主要为了减少 noise 和利用二阶信息，而 DL 中的 variants 是为了改进学习率和解决 saddle point 而来。二阶优化方法是为了挖掘更多可利用信息，而一阶优化是为了更好的利用历史信息。

速度向量: $\nu_i = [\nu_i^1, \nu_i^2, \ldots, \nu_i^D]$
位置向量: $x_i = [x_i^1, x_i^2, \ldots, x_i^D]$
pBest:粒子历史最优位置向量
gBest:粒子群全局最优位置向量
更新公式:

$$\nu_i^d = \omega x \nu_i^d + c_1 x rand_1^d x (pBest_i^d - x_i^d) + c_2 x rand_2^d x (gBest^d - x_i^d) \tag{25}$$

$$x_i^d = x_i^d + r x \nu_i^d \tag{26}$$

惯性系数:$\omega = 0.9$, 学习率:c_1, c_2, 随机数 ([0,1]):$rand_1, rand_2$

Find minimum using PSO

- 初始化所有的个体（粒子），初始化他们的速度和位置，并且将个体的历史最优值 pBest 设置为当前位置，而群体中最优的个体作为当前 gBest。
- 在每一代的进化中，计算各个粒子的适应度函数值 (目标函数值)。
- 如果该粒子当前的适应度函数值比其历史最优值要好，那么历史最优将会被当前位置所替代。
- 如果该粒子的历史最优比全局最优要好，那么全局最优将会被该粒子的历史最优值所替代。
- 对每个粒子 i 的第 D 维的速度和位置按照 (25)(26) 进行更新。
- 如果还没有到达结束条件，转到第二步，否则输出 gBest 并结束。

Pros and Cons(PSO)

1. 利用种群之间的个体比大小来寻找下降方向，适合目标函数含有较多局部极值的问题。
2. 为了保证种群多样性，在本种群更新的时候，来自其他种群的更好的解可能被舍弃掉。
3. 随着维度 D 的增加，新生个体比上一代好的比例急剧下降。(Adam P.Piotrowski,Applied Soft Computing,2014)
4.DL 中对 saddle point 的处理效果有限。(Adam P.Piotrowski,Applied Soft Computing,2014)

Summary

- Optimization Methods for ML&DL,Momentum,NAG,Adam,etc.How to make full use of gradient information?
- Schema about SGDs.How to use stochastic ideas?
- Optimization Methods in Distribution Setting.How to improve convergence,complexity and communication cost?

TKS(Q&R)