Optimization in Machine Learning: Lecture 7

Solving Large-scale Problems I

by Xiaolin Huang   xiaolinhuang@sjtu.edu.cn   SEIEE 2-429

Institute of Image Processing and Pattern Recognition

http://www.pami.sjtu.edu.cn/

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录 Contents

目录 Contents

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Why stochastic

- starting from least squares

$$\min_x \sum_{i=1}^{m} (x^\top a - b_i)^2 \triangleq \sum_{i=1}^{m} f_i(x)$$

- gradient descent:

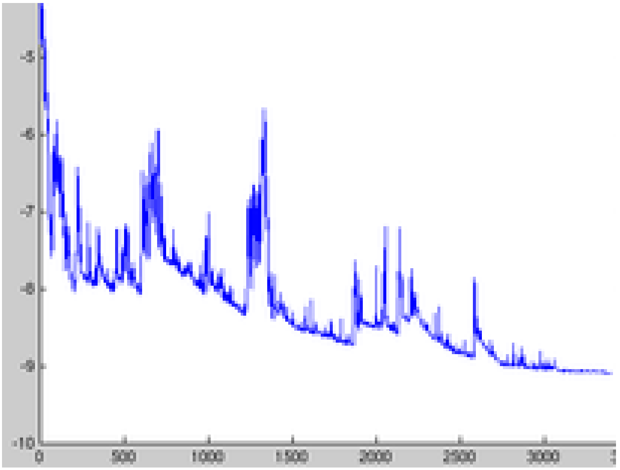
$$x^{k+1} = x^k - \sum_{i=1}^{m} \nabla f_i(x^k) = x^k - t\sum_{i=1}^{m} (x^\top a_i - b_i) a_i$$

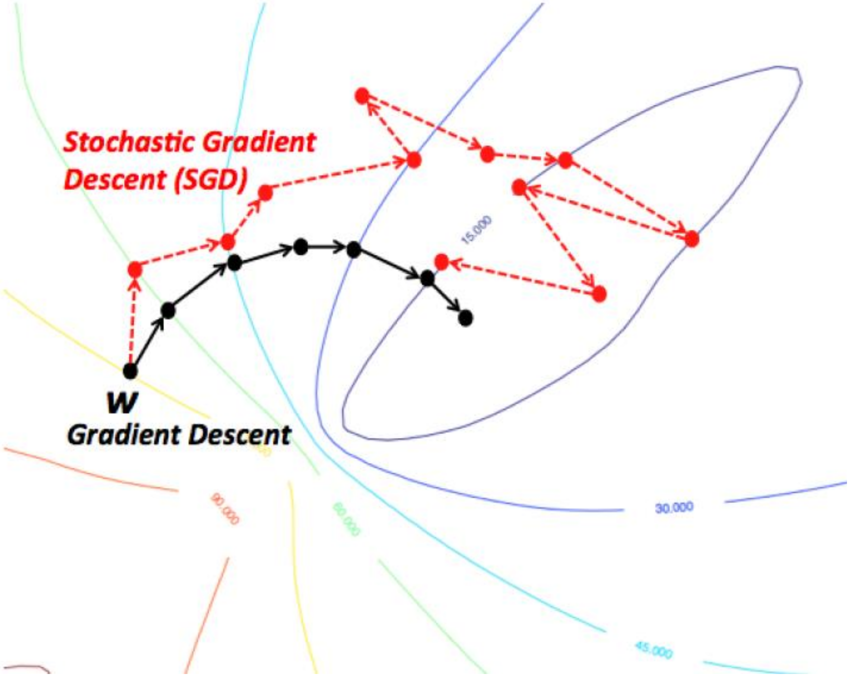
  - when m is large, it is expensive to iteratively calculate $\nabla f = \Sigma \nabla f_i$
  - stochastically choose a subset from the training set
  - use stochastic gradient instead of full gradient

$$\nabla f \approx \nabla f_J = \Sigma_{i \in J} \nabla f_i$$

Why stochastic

- intuitive impression
  - it sounds reasonable
  - at least, it makes solving large-scale problem feasible
  - performance, convergence, speed?





Convergence analysis

$$\min_{x} \sum_{i=1}^{m} (x^\top a_i - b_i)^2$$

- full gradient descent

$$x^{k+1} = x^k - t \sum_{i=1}^{m} (x^\top a_i - b_i) a_i$$

- stochastic gradient descent: simplest case

$$x^{k+1} = x^k - t \underbrace{(x^\top a_i - b_i) a_i}$$

i is randomly selected

$v_i = (x^\top a_i - b_i) a_i$ is a random

Convergence analysis

- notice that $E(v_i) = \nabla f(x)$ (a unbiased estimator)

$$\mathrm{Var}(v_i) = E(\|v\|_2^2) - \|E(v)\|_2^2$$

- with the boundedness condition on the Hessian

$$\nabla^2 f(x) \leq M\mathrm{I}, \qquad \forall x \in S$$

we have

$$f(x - tv_i) \leq f(x) - t\nabla f^T v_i + \frac{Mt^2}{2}\|v_i\|_2^2$$

- taking expectation on both side

$$Ef\left(x^{k+1}\right) \leq f\left(x^k\right) - t\nabla f^T E(v_i) + \frac{Mt^2}{2}E(\|v_i\|_2^2)$$

$$= f\left(x^k\right) - t\left\|\nabla f\left(x^k\right)\right\|_2^2 + \frac{Mt^2}{2}\left(\left\|\nabla f(x^k)\right\|_2^2 + \mathrm{Var}(v_i)\right)$$

Convergence analysis

$$Ef\left(x^{k+1}\right) \le f\left(x^k\right) - t\left\|\nabla f\left(x^k\right)\right\|_2^2 + \frac{Mt^2}{2}\left(\left\|\nabla f\left(x^k\right)\right\|_2^2 + \mathrm{Var}(v_i)\right)$$

$$\le f\left(x^k\right) - \frac{t}{2}\left\|\nabla f\left(x^k\right)\right\|_2^2 + \frac{t}{2}\mathrm{Var}(v_i)$$

(if we choose $Mt \le 1$)

SGD is not monotonic

- furthermore, with convexity, $f(y) \ge f(x) + \nabla f(x)^\top (y - x)$

$$Ef\left(x^{k+1}\right) \le f(x^*) + \nabla f\left(x^k\right)^\top \left(x^k - x^*\right) - \frac{t}{2}\left\|\nabla f\left(x^k\right)\right\|_2^2 + \frac{t}{2}\mathrm{Var}(v_i)$$

$$\le f(x^*) + E(v_i)^\top \left(x^k - x^*\right) - \frac{t}{2}\|E(v_i)\|_2^2 + t\mathrm{Var}(v_i)$$

$$= f(x^*) + E\left(v_i^\top \left(x^k - x^*\right) - \frac{t}{2}\|E(v_i)\|_2^2\right) + t\mathrm{Var}(v_i)$$

$$\le f(x^*) + E\left(\frac{1}{2t}\left(\left\|x^k - x^*\right\|_2^2 - \left\|x^{k+1} - x^*\right\|_2^2\right)\right) + t\mathrm{Var}(v_i)$$

Convergence analysis

$$Ef\left(x^{k+1}\right) \le f(x^*) + E\left(\frac{1}{2t}\left(\left\|x^k - x^*\right\|_2^2 - \left\|x^{k+1} - x^*\right\|_2^2\right)\right) + t\mathrm{Var}(v_i)$$

- by summing the inequality

$$\sum_{k=0}^{K}\left(Ef\left(x^{k+1}\right) - f(x^*)\right) \le \frac{1}{2t}\left(\|x^0 - x^*\|_2^2 - \left\|x^k - x^*\right\|_2^2\right) + kt\mathrm{Var}(v_i)$$

$$\le \frac{\|x^0 - x^*\|_2^2}{2t} + kt\mathrm{Var}(v_i)$$

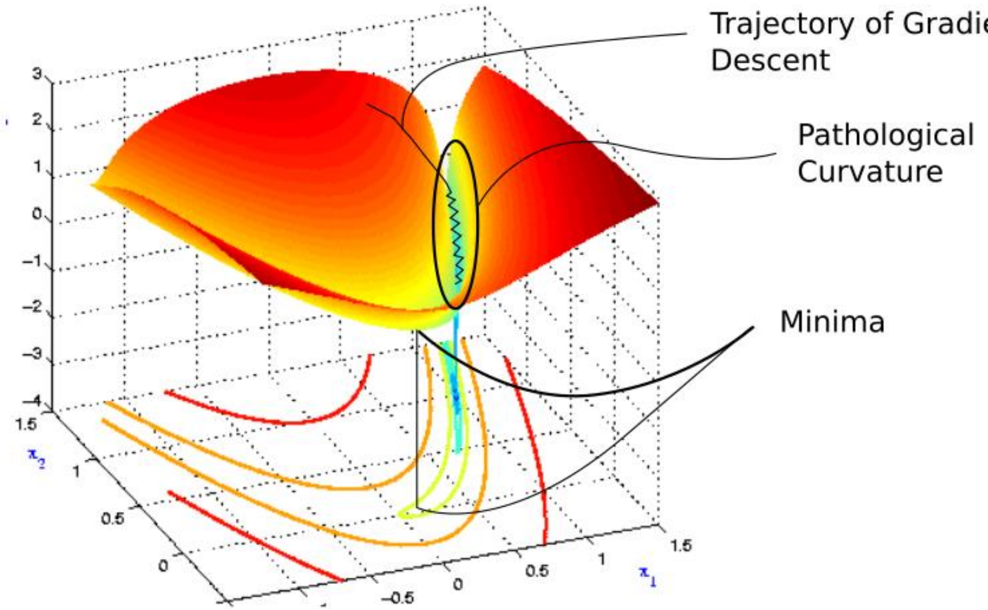
- for the average result $\overline{x_K} = \sum_{i=0}^{K} x^k$

$$Ef(\overline{x_K}) \le f(x^*) + \frac{\|x^0 - x^*\|_2^2}{2tk} + t\mathrm{Var}(v_i)$$

目录 Contents

Revisit gradient descent

- SGD suffers similar problem as full gradient descent



- how to improve?
  - inexact, stochastic, gradient-free, ……
  - ……

SGD with momentum

- Nestrov acceleration

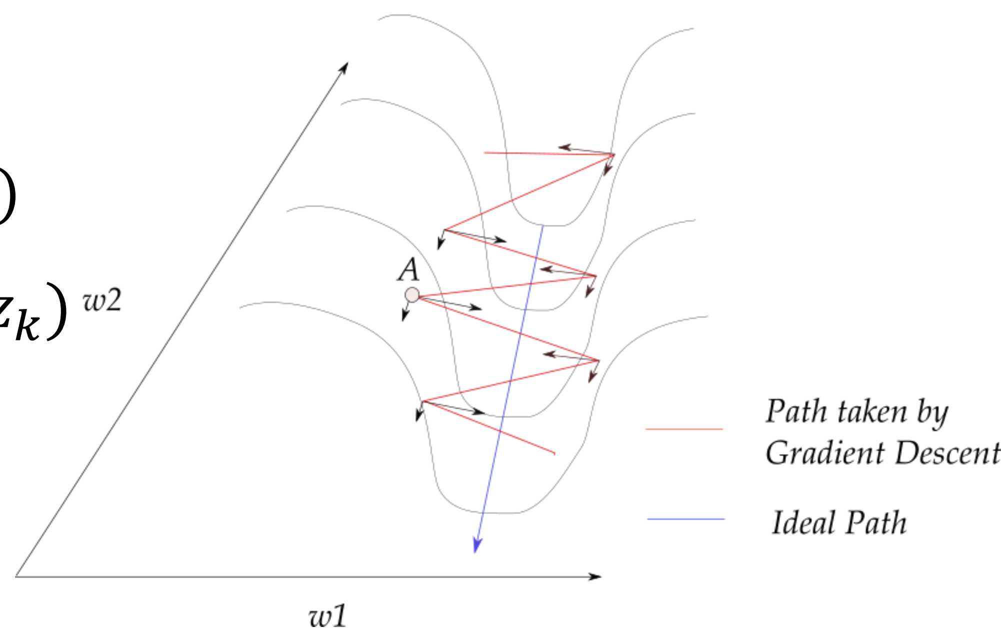
$$z_{k+1} = x_k - t_k \nabla f(x_k)$$

$$x_{k+1} = z_{k+1} + \delta_k (z_{k+1} - z_k)$$

- linear combination of the stochastic gradient and the previous update

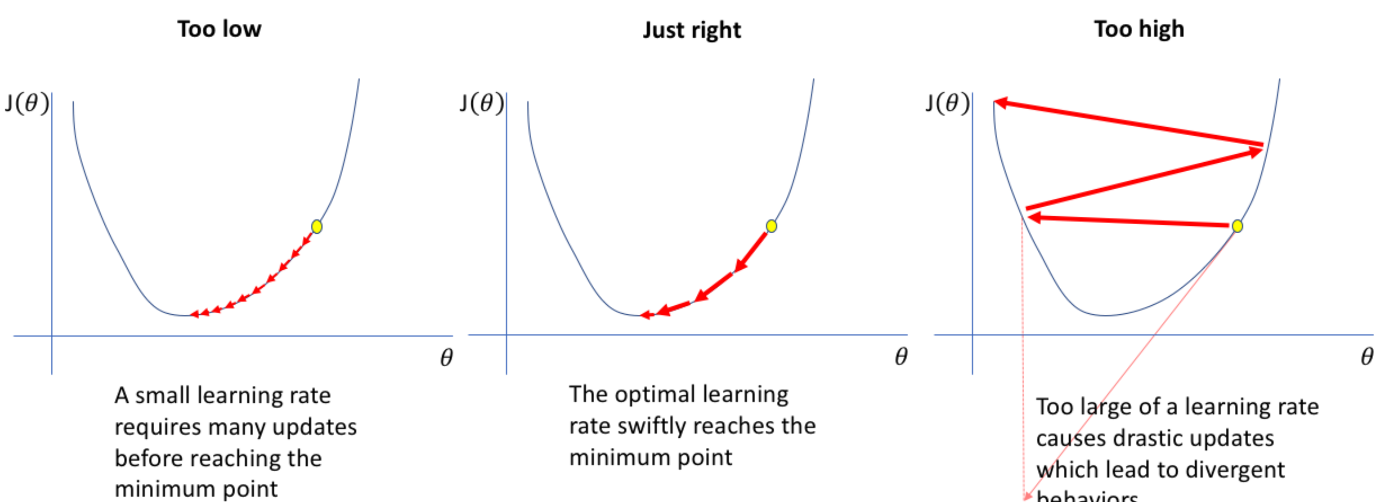
$$z_{k+1} = x_k - t_k \Sigma_{i \in J} \nabla f_i(w_k)$$

$$x_{k+1} = z_{k+1} + \delta_k (z_{k+1} - z_k)$$



Heuristic for learning rate

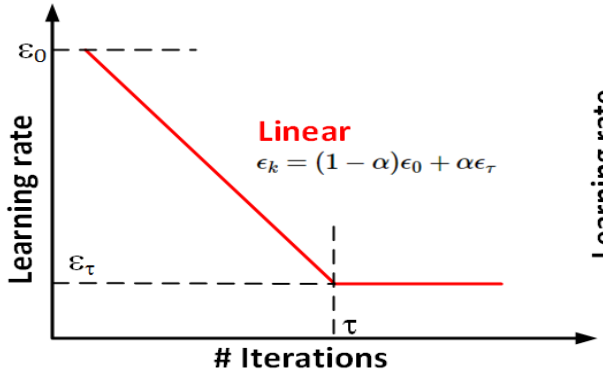
- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem

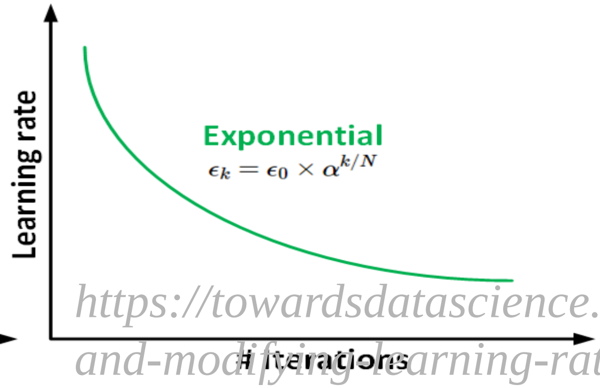


https://www.jeremyjordan.me/nn-learning-rate/

Heuristic for learning rate

- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem
- learning rate decay

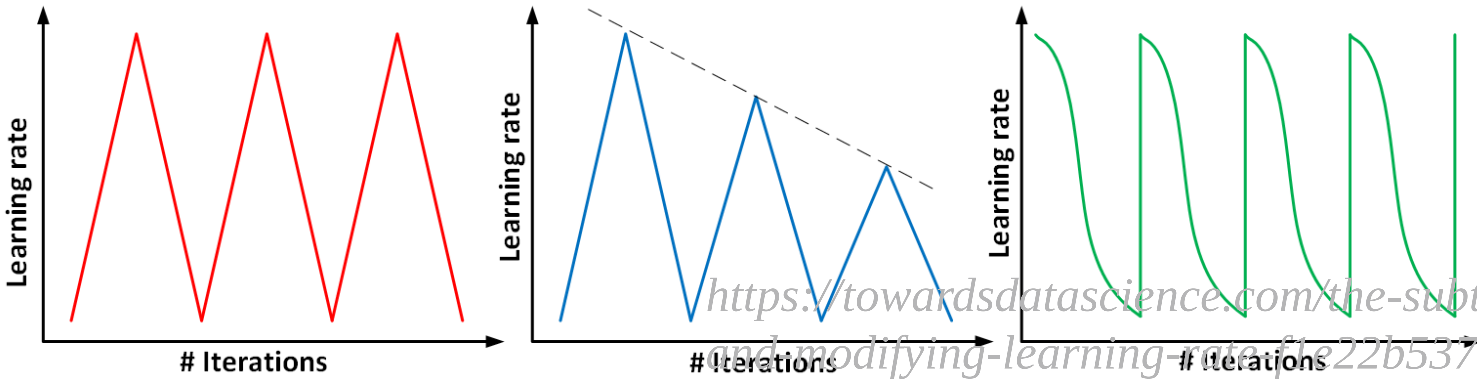




https://towardsdatascience.com/the-subtle-art-of-fixing-and-modifying-learning-rate-f1e22b537303

Heuristic for learning rate

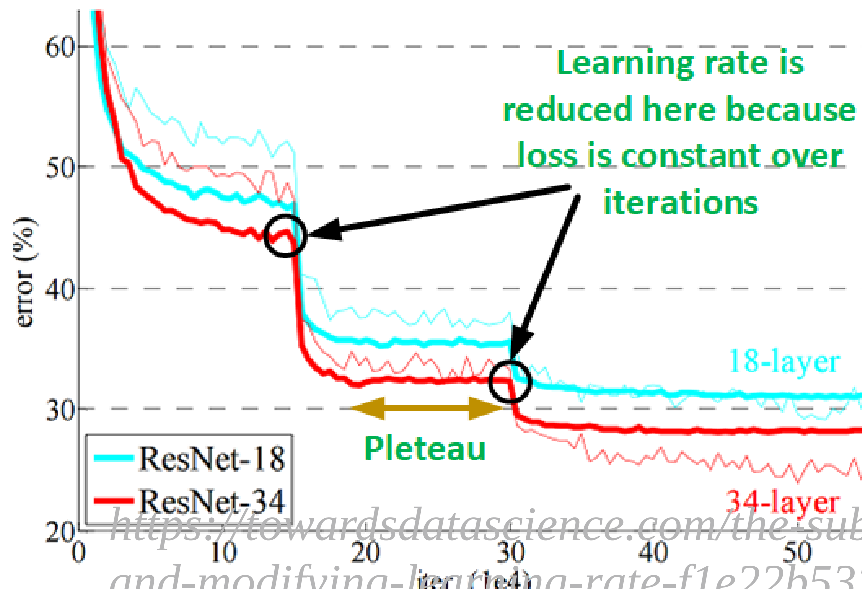
- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem
- learning rate decay



https://towardsdatascience.com/the-subtle-art-of-fixing-and-modifying-learning-rate-f1e22b537303

Heuristic for learning rate

- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem
- learning rate decay



https://towardsdatascience.com/the-subtle-art-of-fixing-and-modifying-learning-rate-f1e22b537303

Normalization can help

- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem
- **normalization is always helpful** when there are hyper-parameters

$$f(x) = x^2 \longrightarrow \nabla f(x) = 2x \longrightarrow x^{k+1} = x^k + 2t^k x^k \qquad \boldsymbol{t^* = 1}$$

$$\big\downarrow \; y = 10x \qquad\qquad \xrightarrow{y = 10x} \; y^{k+1} = y^k + 2t^k y^k$$

$$g(y) = \frac{y^2}{100} \longrightarrow \nabla g(y) = \frac{y}{50} \longrightarrow y^{k+1} = y^k + 2s^k \frac{y^k}{100} \qquad \boldsymbol{s^* = 100}$$

Batch normalization

- (S)GD needs line search
  - one may converge to a region where the gradient's magnitude is small, and then stay there for a very long time
  - adjustable step-length can avoid the situation, but not feasible for large-scale problem
- **normalization is always helpful** when there are hyper-parameters
  - Batch Normalizing: normalize the batch of the input data of neural networks to zero-mean and constant standard deviation

RMSProp

- Root Mean Square Propagation

divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight

$$z_{k+1} = \rho z_k + (1-\rho)\left(\nabla f(x_k) \cdot \nabla f(x_k)\right)$$

$$x_{k+1} = x_k - \frac{\eta}{\sqrt{z_{k+1} + \varepsilon}} \nabla f(x_k)$$

also for stochastic version

- adaptive learning rate

AdaGrad

- Adaptive gradient algorithm

  improves convergence performance, when data are sparse and sparse parameters are more informative

$$G_{ii} = \Sigma_{t=1}^{k}\left(\nabla f(x_t)(i)\right)^2$$

$$x_{k+1} = x_k - \frac{\eta}{\sqrt{G_{ii} + \varepsilon}} \nabla f(x_k)$$

  - overall time (AdaGrad)/sliding window (RMSProp)

Adam (Adaptive Moment Estimation)

- combines the heuristics of
  - momentum
  - RMSProp

$$m_{k+1} = \rho m_k + (1-\rho)\nabla f(x_k)$$

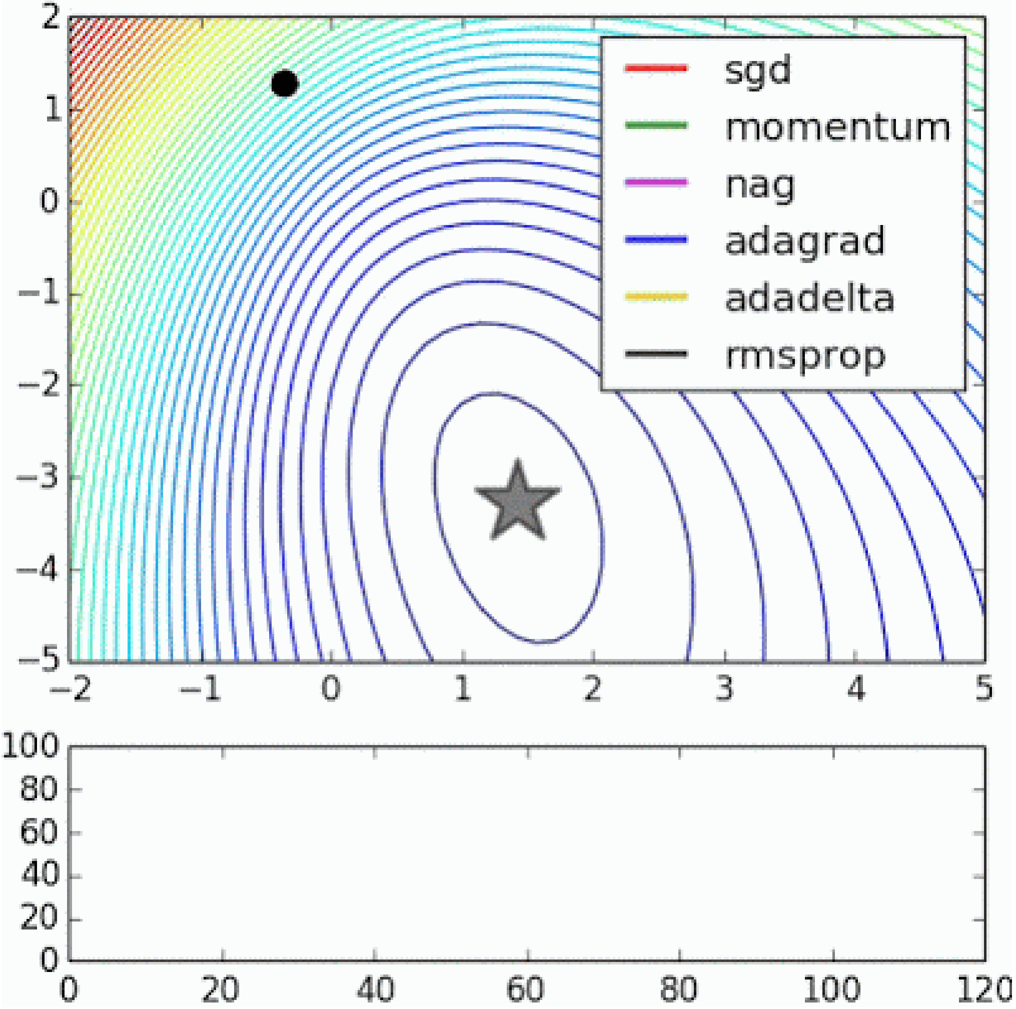
$$v_{k+1} = \rho v_k + (1-\rho)\big(\nabla f(x_k)\cdot\nabla f(x_k)\big)$$

$$\widehat{m}_{k+1} = m_{k+1}/(1-\beta_1^{k+1})$$

$$\hat{v}_{k+1} = v_{k+1}/(1-\beta_2^{k+1})$$

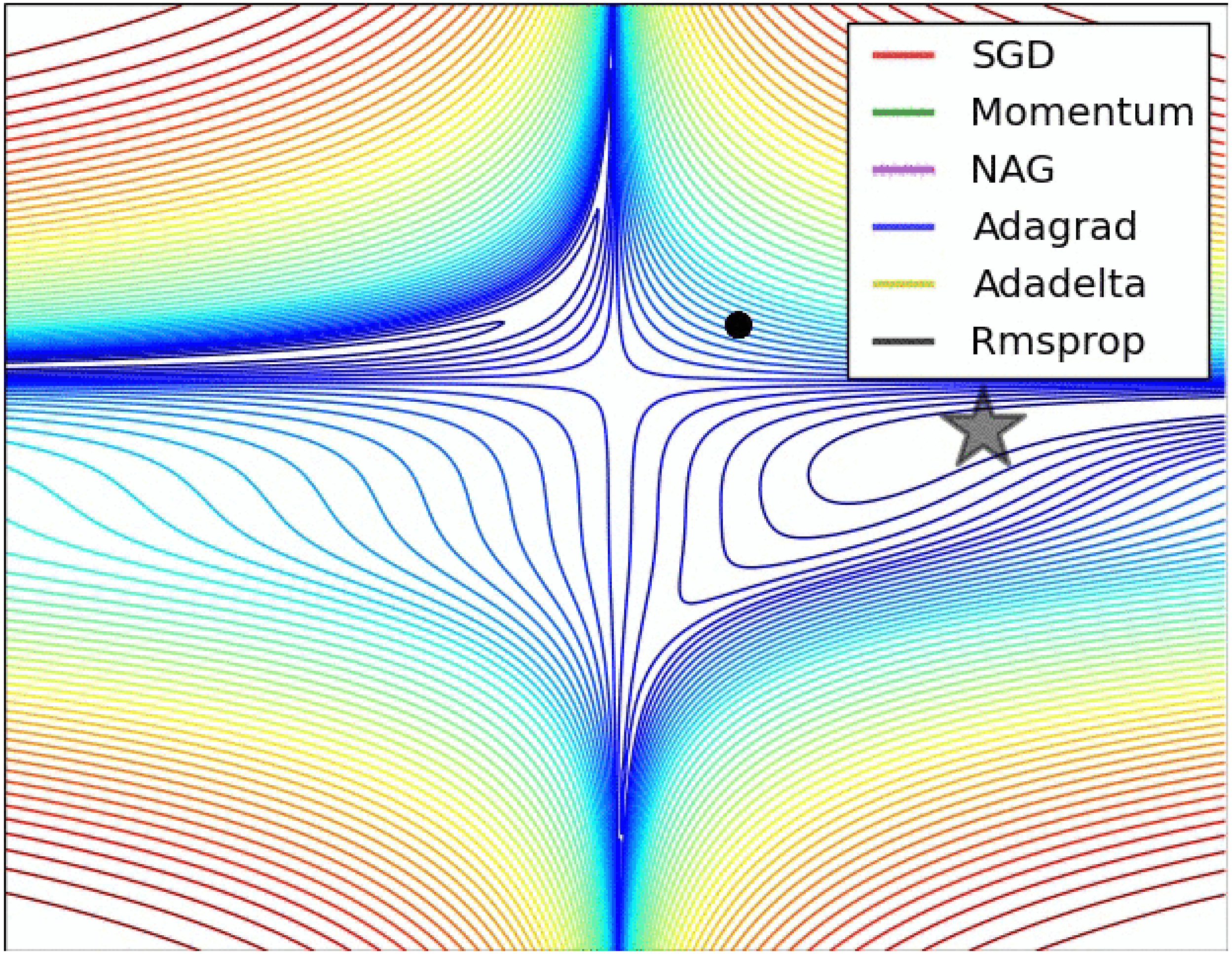
$$x_{k+1} = x_k - \eta\widehat{m}_{k+1}/(\sqrt{\hat{v}_{k+1}}+\varepsilon)$$

Example



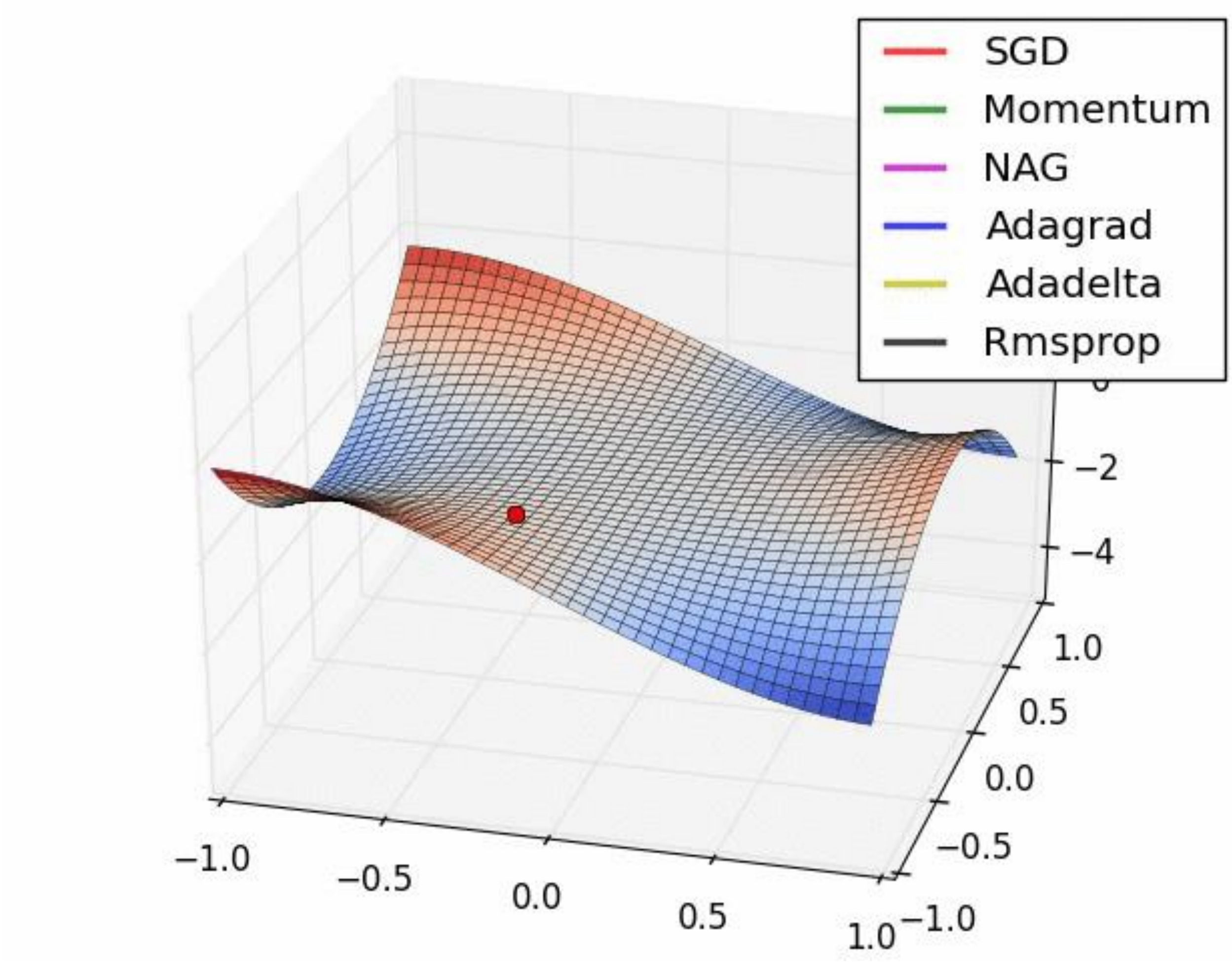
http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Example



http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Example



http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Shuffling and Curriculum Learning

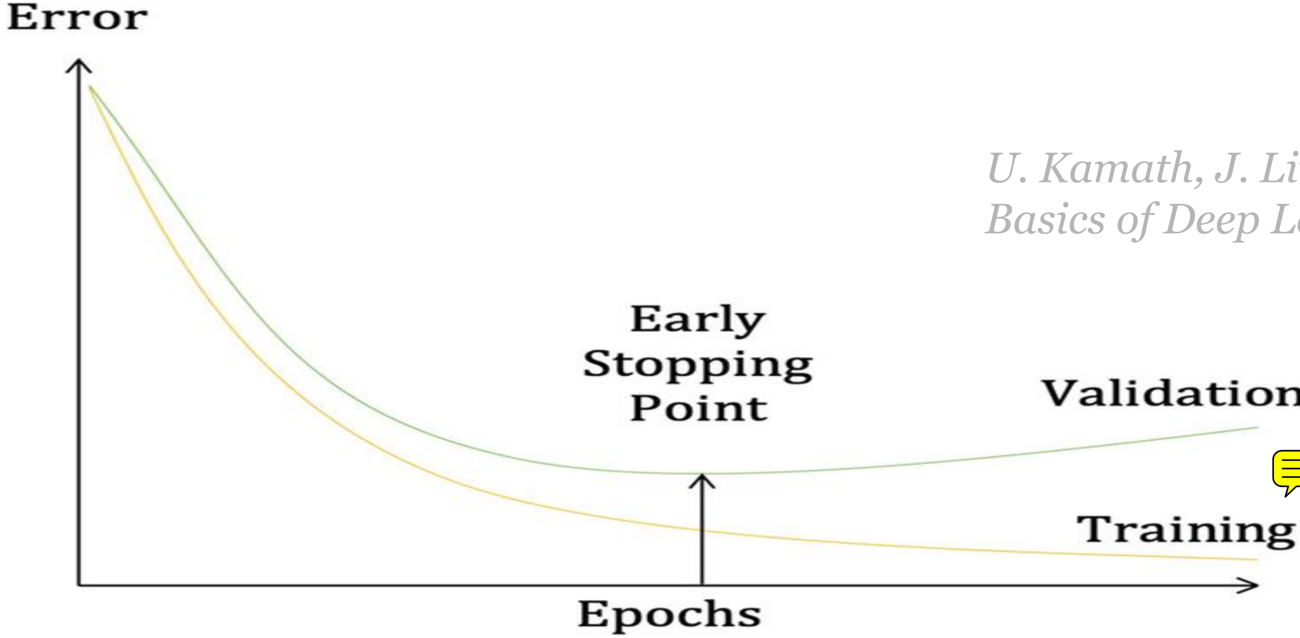
- how to choose the batch for SGD

$$x^{k+1} = x^k - t\Sigma_{i\in I}\nabla f_i(x)$$

  - shuffling: pure randomness to avoid biasness
  - curriculum: supplying the training examples in a meaningful order may actually lead to improved performance

Early Stop

- *"Early stopping (is) beautiful free lunch"*

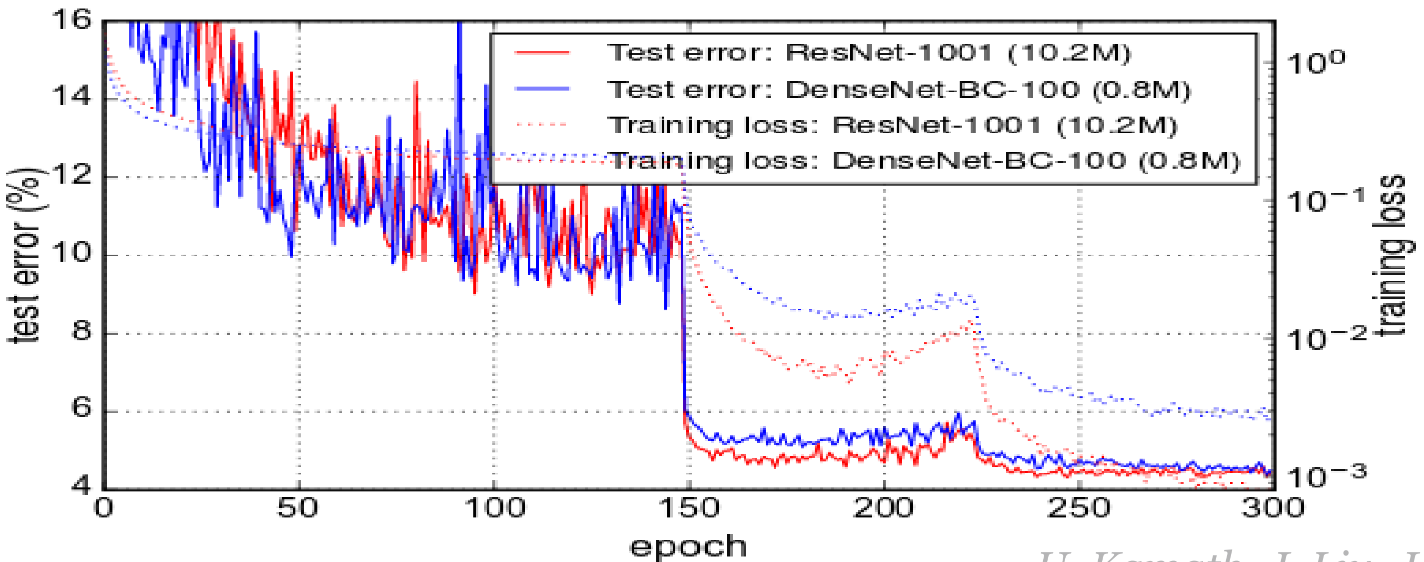


U. Kamath, J. Liu, J. Whitaker: Basics of Deep Learning

  - could be regarded as a regularization term
  - especially, a few-shot learning by fine-tune

Early Stop

- *"Early stopping (is) beautiful free lunch"*



U. Kamath, J. Liu, J. Whitaker: Basics of Deep Learning

  - could be regarded as a regularization term
  - especially, a few-shot learning by fine-tune

Non-smoothness

$$\min_{x} \; x^\top x + C \sum_{i=1}^{m} \max\{0,1 - b_i(x^\top a_i)\}$$

- Pegasos: Primal Estimated sub-GrAdient SOlver for SVM

INPUT: S, λ, T, k
INITIALIZE: Choose $\mathbf{w}_1$ s.t. $\|\mathbf{w}_1\| \le 1/\sqrt{\lambda}$
FOR $t = 1, 2, \ldots, T$
  Choose $A_t \subseteq S$, where $|A_t| = k$
  Set $A_t^+ = \{(\mathbf{x}, y) \in A_t : y \langle \mathbf{w}_t, \mathbf{x} \rangle < 1\}$
  Set $\eta_t = \frac{1}{\lambda t}$
  Set $\mathbf{w}_{t+\frac{1}{2}} = (1 - \eta_t \lambda)\mathbf{w}_t + \frac{\eta_t}{k} \sum_{(\mathbf{x},y) \in A_t^+} y\,\mathbf{x}$
  Set $\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+\frac{1}{2}}\|}\right\} \mathbf{w}_{t+\frac{1}{2}}$
OUTPUT: $\mathbf{w}_{T+1}$

Theorem 3. *Assume that the conditions stated in Thm. 2 hold. Let $\delta \in (0,1)$. Then, with probability of at least $1-\delta$ over the choices of $(A_1, \ldots, A_T)$ and the index r we have that*

$$f(\mathbf{w}_r) \;\le\; f(\mathbf{w}^\star) + \frac{c \ln(T)}{\delta \lambda T} \;.$$

Stochastic in the dual

- Dual coordinate descent
- Stochastic dual coordinate descent
- An example

Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
Benin school of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Tong Zhang TZHANG@STAT.RUTGERS.EDU
Department of Statistics
Rutgers University
Piscataway, NJ, 08854, USA

$$\min_{\mathbf{x}} \; P(\mathbf{x}) \triangleq \mu\|\mathbf{x}\|_1 + \frac{1}{m}\sum_{i=1}^{m} L_{\tau,c}(-y_i(\mathbf{u}_i^\top \mathbf{x})) \quad \text{s.t.} \; \|\mathbf{x}\|_2 \le 1.$$

$$\Longleftrightarrow$$

$$\underset{\mathbf{s},\mathbf{t}}{\text{maximize}} \quad D(\mathbf{s},\mathbf{t}) \triangleq c\sum_{i=1}^{m} t_i - \left\| \sum_{i=1}^{m} t_i y_i \mathbf{u}_i - \mathbf{s} \right\|_2 \quad \text{s.t.} \quad \|\mathbf{s}\|_\infty \le \mu, \quad -\frac{\tau}{m} \le \mathbf{t} \le \frac{1}{m}.$$

  - primal space: non-smooth, constraint
  - dual space: smooth, separable constraint
  - randomly select coordinate and update

Non-convex

- Nowadays, you may meet many non-convex problems
  - finding a global optimum is NP hard
    - do not try theoretical analysis, although it is an open problem
    - randomness may help but not efficient
  - global optimality cannot be guaranteed,

    by neither GD nor SGD
    - some arguments: stochastic helps global optimization

Non-convex

- global optimum → local optimum → saddle points → convergence
  - for non-convex functions, we can still have
    - Second-differentiable condition
    - Lipschitz-continuous condition
    - noise has bounded variance
  - but we do not have convexity

$$f^* \geq f(x) - \frac{1}{2m} \|\nabla f(x)\|_2^2$$

  - non-convex SGD converges, but does not necessarily go to a saddle point, let alone a local optimum

目录 Contents

Estimate the second-order information

- estimate the second-order information by several gradient
  - Hessian: the change (gradient) of gradient
  - $\nabla f\left(x^{k}\right) \approx \nabla f\left(x^{k-1}\right) + t\nabla^{2} f(x^{k-1})$
- using momentum information to lookahead gradient

$$z_{k+1} = x_k - t_k \nabla f(x_k)$$

$$x_{k+1} = x_k + \delta_k (x_k - x_{k-1}) \qquad \delta_k \in [0,1)$$

- using a matrix to approach the Hessian matrix, *Quasi-Newton method*

$$B_k \Delta x_k = -\nabla f(x_k)$$

$$\Delta x_k = -B_k^{-1} \nabla f(x_k)$$

key problem: how to update B or B^{-1} to approach Hessian or its inverse

Construct the inverse

$$\boxed{q_k = \nabla f(x_{k+1}) - \nabla f(x_k)} \approx \nabla^2 f(x_k)(x_{k+1} - x_k)$$

for quadratic case

$$q_k = \nabla f(x_{k+1}) - \nabla f(x_k) = \nabla^2 f(x_k)(x_{k+1} - x_k)$$

- let $H = (\nabla^2 f(x_k))^{-1}$, there should be $Hq_k = x_{k+1} - x_k$
- we want to use rank one correction update

$$H_{k+1} = H_k + a_k z_k z_k^\top \qquad H_{k+1} q_k = x_{k+1} - x_k$$

$$H_{k+1} = H_k + \frac{(x_{k+1} - x_k - H_k q_k)(x_{k+1} - x_k - H_k q_k)^\top}{q_k^\top (x_{k+1} - x_k - H_k q_k)}$$

$\Delta x^k = -H_{k+1} \nabla f(x_k)$ modified Newton's method with rank 1 correction

DFP and BFGS method

- Davidon-Fletcher-Powell uses Rank two correction
- $H_{k+1} = H_k + a_k z_k z_k^\top + \beta_k y_k y_k^\top \qquad H_{k+1} q_k = x_{k+1} - x_k$

$$H_{k+1} = H_k + \frac{(x_{k+1} - x_k)(x_{k+1} - x_k)^\top}{(x_{k+1} - x_k)^\top q_k} - \frac{H_k q_k q_k^\top H_k}{q_k^\top H_k q_k}$$

- Broyden-Fletcher-Goldfarb-Shanno estimate the Hessian instead of its inverse

$$H_{k+1} q_k = x_{k+1} - x_k \qquad \nabla^2 f(x_k)(x_{k+1} - x_k) = q_k$$

by symmetry, you could directly change the position

$$Q_{k+1} = Q_k + \frac{(q_k - H_k(x_{k+1} - x_k))(q_k - H_k(x_{k+1} - x_k))^\top}{(x_{k+1} - x_k)^\top (p_k - H_k(x_{k+1} - x_k))}$$

DFP and BFGS method

- Davidon-Fletcher-Powell uses Rank two correction
- $H_{k+1} = H_k + a_k z_k z_k^\top + \beta_k y_k y_k^\top \qquad H_{k+1} q_k = x_{k+1} - x_k$

$$H_{k+1} = H_k + \frac{(x_{k+1} - x_k)(x_{k+1} - x_k)^\top}{(x_{k+1} - x_k)^\top q_k} - \frac{H_k q_k q_k^\top H_k}{q_k^\top H_k q_k}$$

- Broyden-Fletcher-Goldfarb-Shanno estimate the Hessian instead of its inverse

$$H_{k+1} q_k = x_{k+1} - x_k \qquad \nabla^2 f(x_k)(x_{k+1} - x_k) = q_k$$

by symmetry, you could directly change the position

$$Q_{k+1} = Q_k + \frac{q_k q_k^\top}{(x_{k+1} - x_k)^\top q_k} - \frac{Q_k (x_{k+1} - x_k)(x_{k+1} - x_k)^\top Q_k}{(x_{k+1} - x_k)^\top Q_k (x_{k+1} - x_k)}$$

BFGS algorithm

- for optimization, we actually need Q^{-1}

Sherman-Morrison matrix inversion formula

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}$$

- let $B_{k+1} = Q_{k+1}^{-1}$ and apply Sherman-Morrison formula twice

$$\boxed{B_{k+1} = B_k - \frac{B_k(x_{k+1} - x_k)^\top B_k}{(x_{k+1} - x_k)^\top B_k(x_{k+1} - x_k)} + \frac{q_k q_k^\top}{q_k{}^\top(x_{k+1} - x_k)}}$$

BFGS is almost the same as DFP, but in practice, BFGS seems work better

—— Tibshirani

Limited-memory BFGS

- what we really need is $B_{k+1}\nabla f(x_{k+1})$
- instead of storing B in BFGS

$$B_{k+1} = B_k - \frac{B_k(x_{k+1} - x_k)^\top B_k}{(x_{k+1} - x_k)^\top B_k(x_{k+1} - x_k)} + \frac{q_k q_k^\top}{q_k{}^\top(x_{k+1} - x_k)}$$

- we only need to calculate $B_{k+1}\nabla f(x_{k+1})$
  - maintain two recursive sequences of the past m updates for x and $\nabla f(x)$
  - starting from $B_{k-m} = I$
- modification/improvement
  - L-BFGS-b: to handle box constrains
  - O-LBFGS: online version with randomly drawn subset
  - forgetting/mini-batch/….

目录 Contents

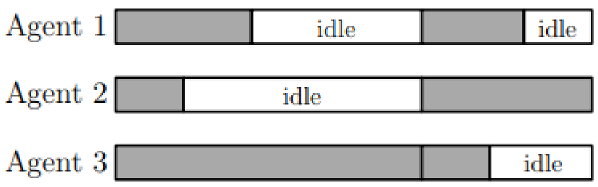
Parallel computing I

$$\min_{x} \quad \|B - Ax\|_2^2$$

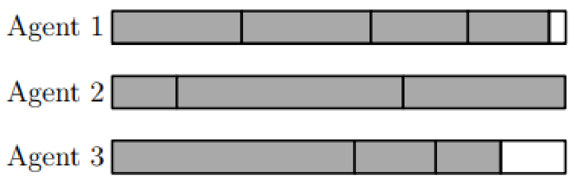
- gradient descent

$$\begin{aligned} x^{k+1} &= x^k + \lambda_k A^\top (B - Ax^k) \\ &= (I - \lambda_k A^\top A) x^k + \lambda_k A^\top B \\ &\triangleq T_k x^k + b_k \end{aligned}$$

- coordinate update
  - synchronous strategy
  - asynchronous strategy



Synchronous
(wait for the slowest)

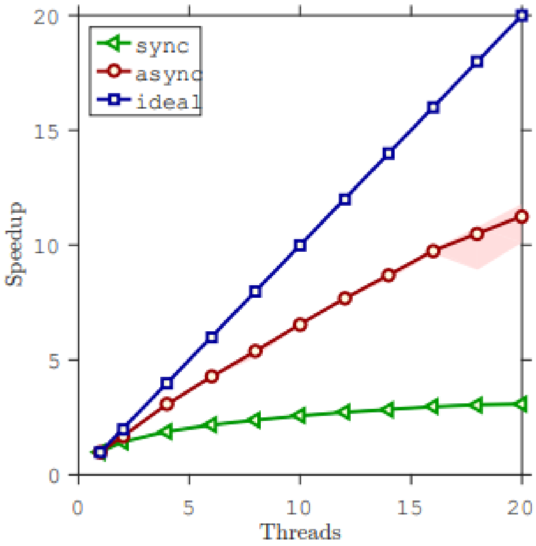


Asynchronous
(non-stop, no wait)

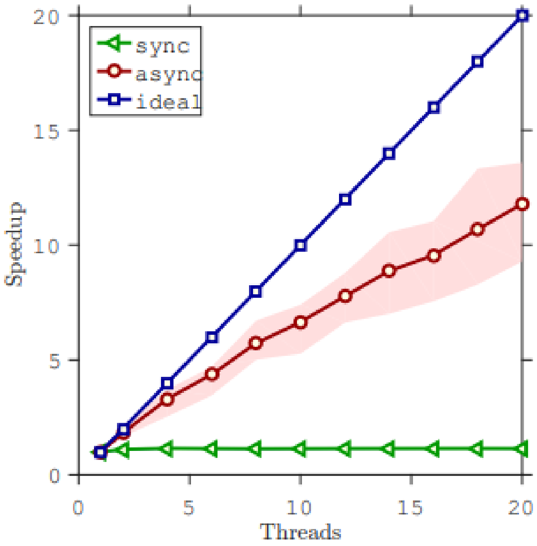
Z. Peng, Y. Xu, M. Yan, W. Yin, ARock: an algorithmic framework for asynchronous parallel coordinate updates, *SIAM Journal on Scientific Computing*, 2016

Parallel computing I

- convergence for synchronous strategy has no problem
- asynchronous strategy is much more efficient



dataset “news20”



dataset “url”

Z. Peng, Y. Xu, M. Yan, W. Yin, ARock: an algorithmic framework for asynchronous parallel coordinate updates, *SIAM Journal on Scientific Computing*, 2016

Parallel computing I

- convergence for synchronous strategy has no problem
- asynchronous strategy is much more efficient
- convergence is a problem

Z. Peng, Y. Xu, M. Yan, W. Yin, ARock: an algorithmic framework for asynchronous parallel coordinate updates, *SIAM Journal on Scientific Computing*, 2016

notation:

- $m = \#$ coordinates
- $\tau =$ the maximum delay
- uniform selection $p_i \equiv \frac{1}{m}$ (just for simplicity)

Theorem (almost sure convergence)

Assume that T is nonexpansive and has a fixed point. Use step sizes $\eta_k \in [\epsilon, \frac{1}{2m^{-1/2}\tau+1})$, $\forall k$. Then, with probability one, $x^k \rightharpoonup x^* \in \mathrm{Fix}T$.

Parallel computing I

- proof
  - **typical inequality**:

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - c\|Tx^k - x^k\|^2 + \text{harmful terms}(x^{k-1}, \ldots, x^{k-\tau})$$

  - **descent inequality under a new metric**:

$$\mathbb{E}\left(\|\mathbf{x}^{k+1} - \mathbf{x}^*\|_M^2 \,\middle|\, \mathcal{X}^k\right) \leq \|\mathbf{x}^k - \mathbf{x}^*\|_M^2 - c\,\|Tx^k - x^k\|^2$$

  where

    - $\mathbf{x}^k = (x^k, x^{k-1}, \ldots, x^{k-\tau}) \in \mathcal{H}^{\tau+1},\ k \geq 0$
    - $\mathbf{x}^* = (x^*, x^*, \ldots, x^*) \in \mathbf{X}^* \subseteq \mathcal{H}^{\tau+1}$
    - M is a positive definite matrix.
    - c depends on (η_k, m, τ)

Z. Peng, Y. Xu, M. Yan, W. Yin, ARock: an algorithmic framework for asynchronous parallel coordinate updates, *SIAM Journal on Scientific Computing*, 2016

Parallel computing I

- proof

  - **typical inequality**:

$$\|x^{k+1} - x^*\|^2 \le \|x^k - x^*\|^2 - c\|Tx^k - x^k\|^2 + \text{harmful terms}(x^{k-1}, \ldots, x^{k-\tau})$$

  - **descent inequality under a new metric**:

$$\mathbb{E}\left(\|\mathbf{x}^{k+1} - \mathbf{x}^*\|_M^2 \,\middle|\, \mathcal{X}^k\right) \le \|\mathbf{x}^k - \mathbf{x}^*\|_M^2 - c\,\|Tx^k - x^k\|^2$$

    where

    - $\mathbf{x}^k = (x^k, x^{k-1}, \ldots, x^{k-\tau}) \in \mathcal{H}^{\tau+1},\ k \ge 0$
    - $\mathbf{x}^* = (x^*, x^*, \ldots, x^*) \in \mathbf{X}^* \subseteq \mathcal{H}^{\tau+1}$
    - M is a positive definite matrix.
    - c depends on (η_k, m, τ)

Z. Peng, Y. Xu, M. Yan, W. Yin, ARock: an algorithmic framework for asynchronous parallel coordinate updates, *SIAM Journal on Scientific Computing*, 2016

THANKS