Second-Order Methods for Deep Learning

Baoyu Zhou & Majid Jahani

Weekly OptML Seminars

Oct 2, 2019

Overview

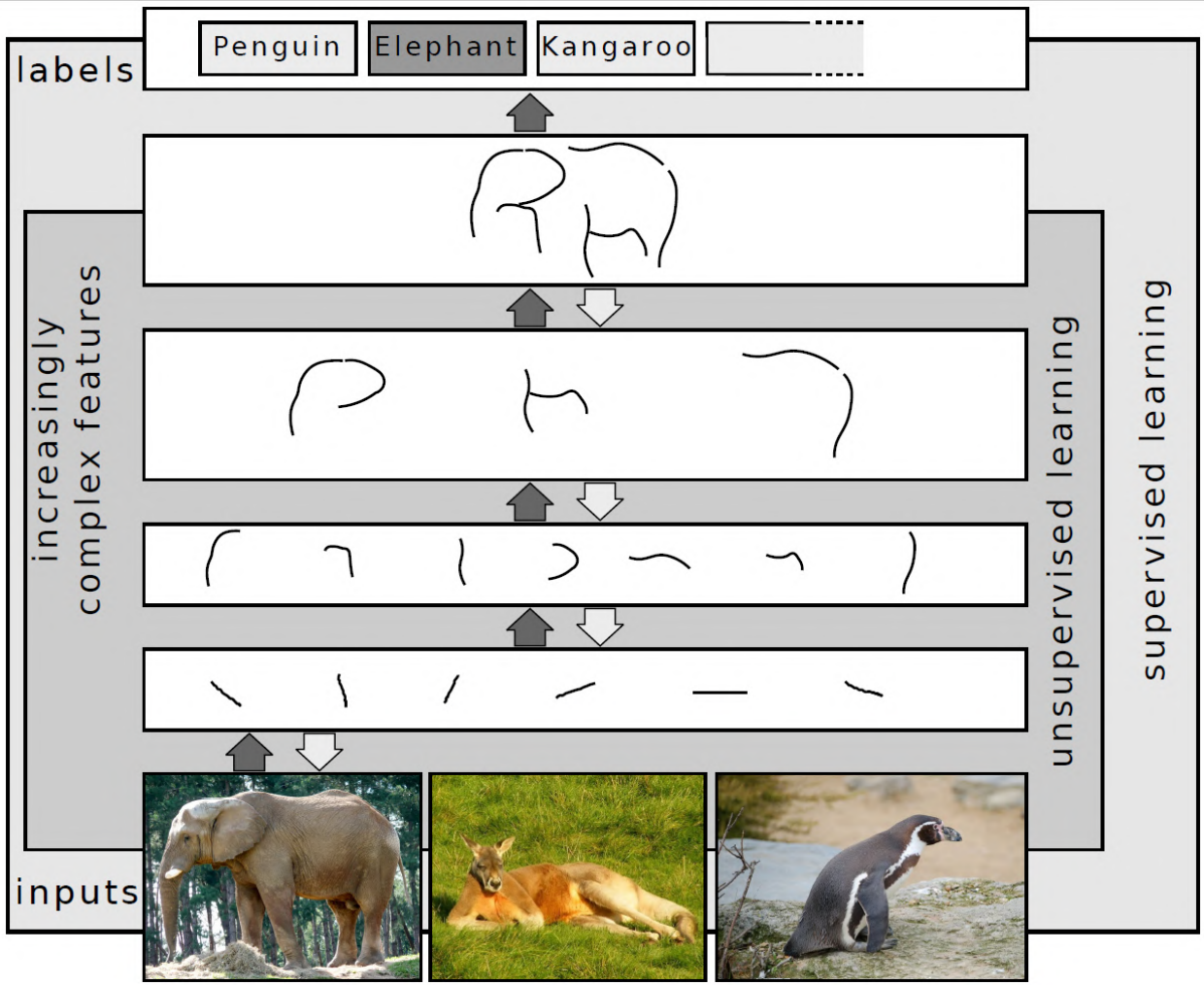
- Main Questions
- Outline
  - First-Order Methods (Deterministic and Stochastic)
  - Deterministic Newton Methods
  - Deterministic Quasi-Newton Methods
  - Stochastic Newton Methods
  - Stochastic Quasi-Newton Methods

Main Questions

- What is deep learning (DL) and some of its applications?
- What are the reasons for using 1st-order methods for DL?
- What are the advantages and disadvantages of first and second-order methods?
- What are the challenges for using 2nd-order methods?
- For which problems 2nd-order methods work better in DL area?
- ...

What is DL and some of its applications?

DL is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input.



What is DL and some of its applications?

Automatic speech recognition; image recognition; natural language processing; drug discovery and toxicology; medical image analysis; quantum chemistry; and many many other applications

Problem

Before problem statement, lets see a simple example[1]

[1] for more demos, visit https://cs.stanford.edu/people/karpathy/convnetjs/

Problem

Before problem statement, lets see a simple example[1]

$$\min_{w \in \mathbb{R}^d} F(w) := \frac{1}{n} \sum_{i=1}^{n} f(w; x^i, y^i) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

- the number of samples n is large;
- the number of variables d is large;
- the objective function is nonconvex.

[1] for more demos, visit https://cs.stanford.edu/people/karpathy/convnetjs/

First-Order Methods

- gradient descent (GD) method

$$w_{k+1} = w_k - \alpha_k \nabla F(w_k)$$

First-Order Methods

- gradient descent (GD) method

$$w_{k+1} = w_k - \alpha_k \nabla F(w_k)$$

- stochastic gradient (SG) method
  - $w_{k+1} = w_k - \alpha_k g_k, \quad g_k \approx \nabla F(w_k)$
  - where g_k is an estimation of the true gradient at w_k
- SG (with momentum), Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam, AMSGrad and ...

First-Order Methods

Stochastic first order methods

Pros:

- Lower per-iteration cost
- Easy to implement
- Effective for many ML problems

First-Order Methods

Stochastic first order methods

Pros:

- Lower per-iteration cost
- Easy to implement
- Effective for many ML problems

Cons:

- Highly sensitive to the choice of hyper-parameters (cumbersome tuning)
- Limited opportunities for parallelism

First-Order Methods

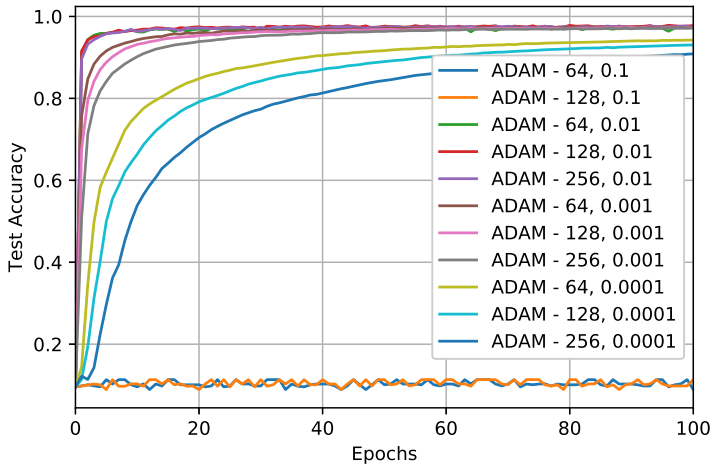
Stochastic first order methods

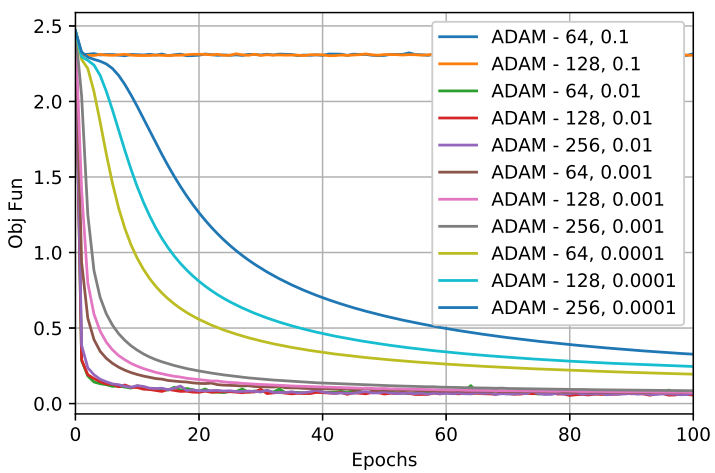
Pros:

- Lower per-iteration cost
- Easy to implement
- Effective for many ML problems

Cons:

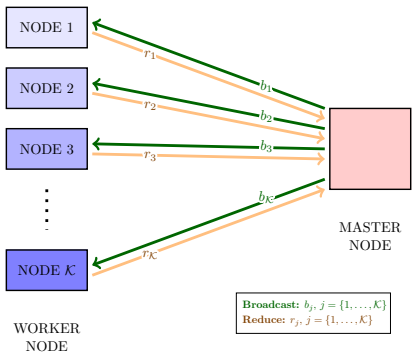
- Highly sensitive to the choice of hyper-parameters (cumbersome tuning)
- Limited opportunities for parallelism





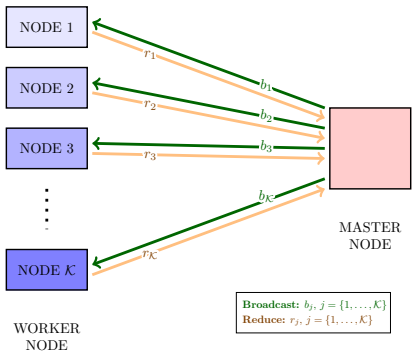
Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$



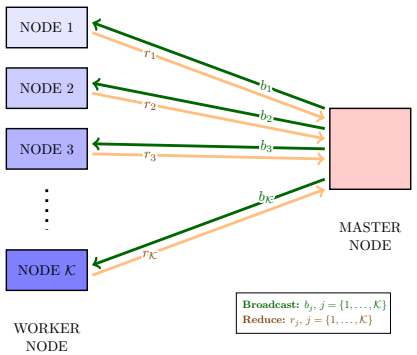
Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples



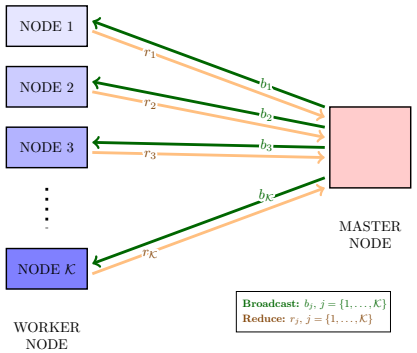
Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model



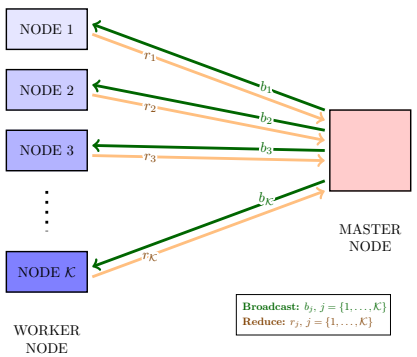
Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$



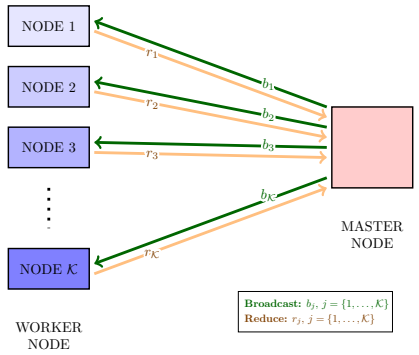
Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$
- Each node will compute the gradient and they will communicate to sum them



Limited opportunities for parallelism of stochastic 1st-order methods

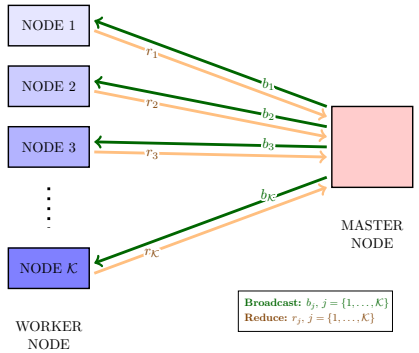
- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$
- Each node will compute the gradient and they will communicate to sum them



- $\log(\mathcal{K})$ rounds of communications is needed to make sure each node can perform an update to model

Limited opportunities for parallelism of stochastic 1st-order methods

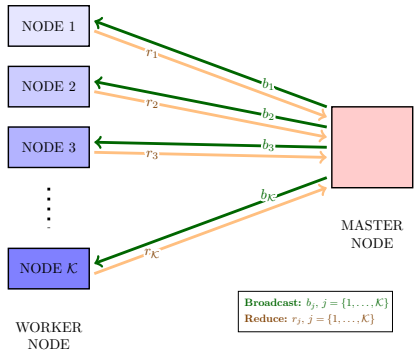
- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$
- Each node will compute the gradient and they will communicate to sum them



- $\log(\mathcal{K})$ rounds of communications is needed to make sure each node can perform an update to model
- Assume communications needs $1s$, thus, $4s$/iter

Limited opportunities for parallelism of stochastic 1st-order methods

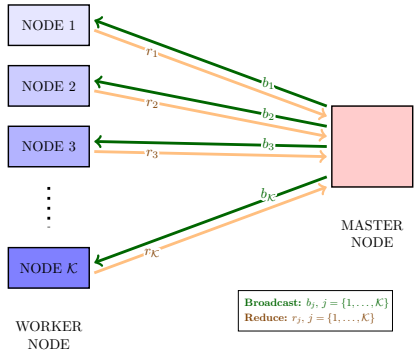
- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$
- Each node will compute the gradient and they will communicate to sum them



- $\log(\mathcal{K})$ rounds of communications is needed to make sure each node can perform an update to model
- Assume communications needs $1s$, thus, $4s$/iter
- 1 epoch = 25000 iters

Limited opportunities for parallelism of stochastic 1st-order methods

- Assume we have 1.2×10^6 images
- $\mathcal{K} = 12$
- Store on each node 10^5 data samples
- Each node will have a copy of the model
- Each node will choose $b = 4 \rightarrow \mathcal{K}b = 48$
- Each node will compute the gradient and they will communicate to sum them



- $\log(\mathcal{K})$ rounds of communications is needed to make sure each node can perform an update to model
- Assume communications needs $1s$, thus, $4s$/iter
- 1 epoch = 25000 iters

- In each epoch we will spend $100000s = 1.15740740741$ days

Second-Order Methods

Newton Method

$$w_{k+1} = w_k - (\nabla^2 F(w_k))^{-1} \nabla F(w_k)$$

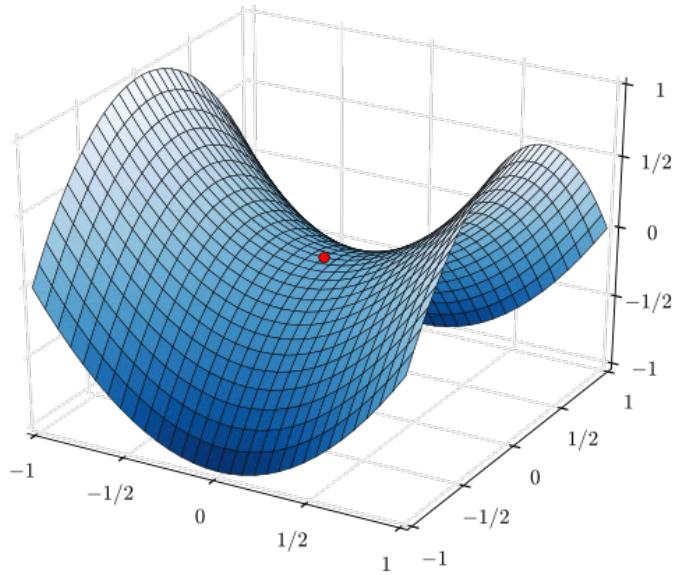
which the new search direction is the solution of the local quadratic model of F at w_k

$$M(w) = F(w_k) + \nabla F(w_k)^T (w - w_k) + \frac{1}{2}(w - w_k)^T \nabla^2 F(w_k)(w - w_k)$$

Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$

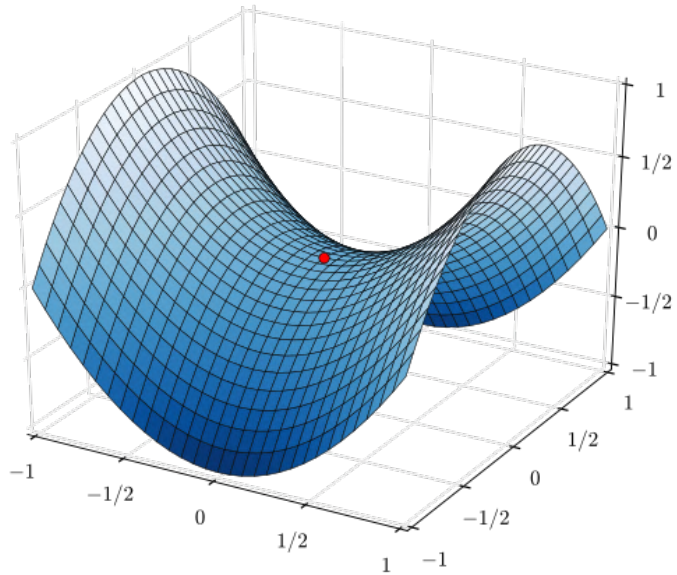


Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$

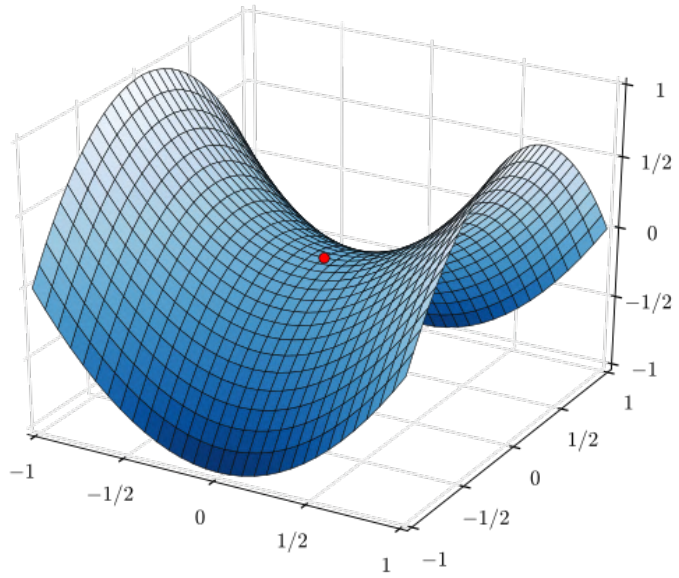
$$w_{k+1} = w_k - [\nabla^2 f(w_k)]^{-1} \nabla f(w_k)$$



Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$



$$w_{k+1} = w_k - [\nabla^2 f(w_k)]^{-1} \nabla f(w_k)$$

$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

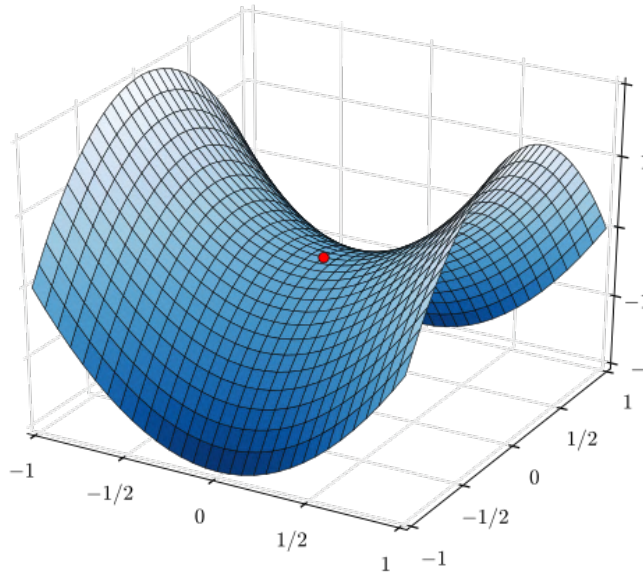
Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$

$$w_{k+1} = w_k - [\nabla^2 f(w_k)]^{-1} \nabla f(w_k)$$

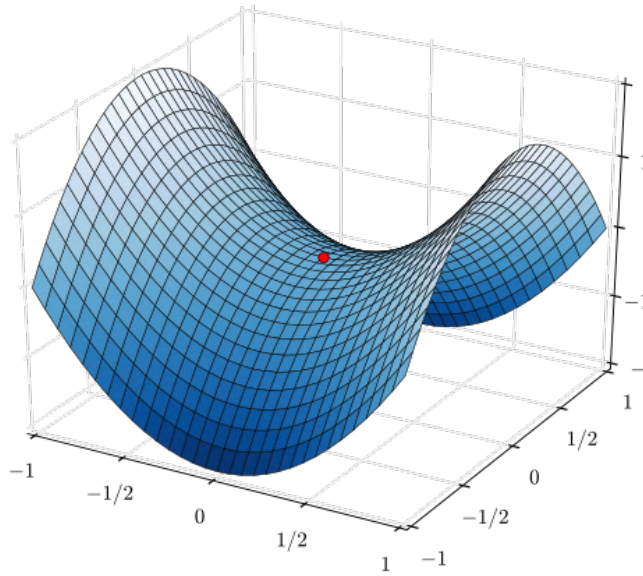
$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \rightarrow (\nabla^2 f)^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}$$



Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$



$$w_{k+1} = w_k - [\nabla^2 f(w_k)]^{-1} \nabla f(w_k)$$

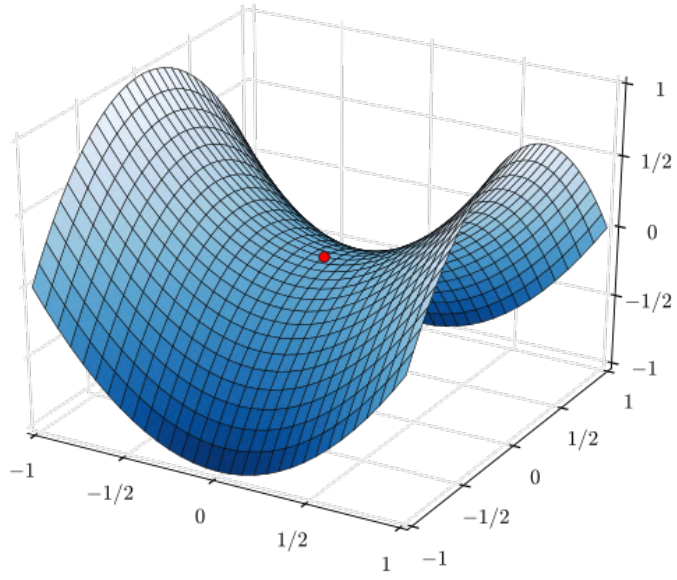
$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \rightarrow (\nabla^2 f)^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_{k} - \begin{bmatrix} 1/2 & 0 \\ 0 & -1/2 \end{bmatrix} \begin{bmatrix} 2(\mathbf{w}_1)_k \\ -2(\mathbf{w}_2)_k \end{bmatrix}$$

Newton method may attract saddle point??!!

Lets consider

$$F(\underbrace{\mathbf{w}_1, \mathbf{w}_2}_{w}) = \mathbf{w}_1^2 - \mathbf{w}_2^2$$



$$w_{k+1} = w_k - [\nabla^2 f(w_k)]^{-1} \nabla f(w_k)$$

$$\nabla^2 f = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_k - \begin{bmatrix} 1/2 & 0 \\ 0 & -1/2 \end{bmatrix} \begin{bmatrix} 2(\mathbf{w}_1)_k \\ -2(\mathbf{w}_2)_k \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_k - \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix}_k = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Second-Order Methods

Newton CG

For solving the linear equation

$$A(w - w_k) = b,$$

CG method tries to minimize

$$\phi(w - w_k) = \frac{1}{2}(w - w_k)^T A(w - w_k) - b^T(w - w_k).$$

Newton method is trying to solve the linear equation as

$$(\nabla^2 F(w_k))(w - w_k) = -\nabla F(w_k).$$

Newton CG is using CG method to solve the Newton equation.

Second-Order Methods

Newton CG

We learned Trust region CG method in ISE417. The practical version of algorithm is as follows:

Given $x_0 = 0$, set $r_0 = Ax_0 - b$, $p_0 = -r_0$, and for $k = 0, 1, 2, \ldots$:

$$\text{if } p_k^T A p_k < 0 \text{ then } \alpha_k \leftarrow \text{ positive value s.t. } \|x_k + \alpha_k p_k\| = \Delta_k \text{ and stop;}$$

$$\text{else } \alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k};$$

$$\text{if } \|x_k + \alpha_k p_k\| > \Delta_k \text{ then reset } \alpha_k \leftarrow \text{ positive value s.t. } \|x_k + \alpha_k p_k\| = \Delta_k \text{ and stop;}$$

$$\text{else } x_{k+1} \leftarrow x_k + \alpha_k p_k;$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k;$$

$$\text{if } \|r_{k+1}\| \approx 0 \text{ then stop;}$$

$$\text{else } \beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k};$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k;$$

(In all cases, stop means return $x_k + \alpha_k p_k$.)

Quasi-Newton Method

$$w_{k+1} = w_k - \alpha_k H_k \nabla F(w_k)$$

which the new search direction is the solution of the local quadratic model of F at w_k

$$M(w) = F(w_k) + \nabla F(w_k)^T (w - w_k) + \frac{1}{2}(w - w_k)^T B_k (w - w_k)$$

Quasi-Newton Method (BFGS and L-BFGS)

$$w_{k+1} = w_k - \alpha_k H_k \nabla F(w_k),$$

where $H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$

and $\rho_k = \dfrac{1}{y_k^T s_k}$, $V_k = I - \rho_k y_k s_k^T$

and the curvature pairs (s_k, y_k) :

$$s_k = w_k - w_{k-1},$$

$$y_k = \nabla F(w_k) - \nabla F(w_{k-1})$$

BFGS condition:

$$s_k^T y_k \geq \epsilon \|s_k\|^2$$

Quasi-Newton Method (SR1 and L-SR1)

$$w_{k+1} = w_k + p_k,$$

where p_k is the minimizer of the following subproblem

$$\min_p \; m_k(p) = F(w_k) + \nabla F(w_k)^T p + \tfrac{1}{2} p^T B_k p,$$

$$\text{s.t.} \quad \|p\| \leq \Delta_k,$$

Δ_k is the trust region and B_k is the SR1 Hessian approximation computed as

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}.$$

SR1 condition:

$$|s_k^T (y_k - B_k s_k)| \geq \epsilon \|s_k\| \|y_k - B_k s_k\|$$

Stochastic Second-Order Methods

Subsampled Newton[2]

$$w_{k+1} = w_k + \alpha_k p_k$$

where p_k is the solution of the following system of equations

$$\nabla^2 F_{S_k}(w_k) p_k = -\nabla F_{X_k}(w_k)$$

and

$$\nabla^2 F_{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla^2 F_i(w_k)$$

$$\nabla F_{X_k}(w_k) = \frac{1}{|X_k|} \sum_{i \in X_k} \nabla F_i(w_k)$$

[2] Bollapragada, R., Byrd, R. H., & Nocedal, J. (2018). Exact and inexact subsampled Newton methods for optimization. IMA Journal of Numerical Analysis, 39(2), 545-578.

Stochastic Second-Order Methods

- The choice of $\{X_k\}$ and $\{S_k\}$ results in different algorithms
- $\{X_k\}$ and $\{S_k\}$ are
  - gradient and Hessian samples
  - chosen independently (with or without replacement)
- For exact subsampled Newton
  - $p_k = -\left(\nabla^2 F_{S_k}(w_k)\right)^{-1} \nabla F_{X_k}(w_k)$
  - $\{X_k\}$ is increased at a geometric rate
  - $|S_k|$ is fixed (global linear rate)
  - $|S_k| \geq |S_{k-1}|$ (local superlinear rate)

Stochastic Second-Order Methods

- The inexact subsampled Newton
  - CG is used as a solver for to solve the linear system
  - The Hessian is subsampled but the gradient is not
  - $\nabla^2 F_{S_k}(w_k) p_k = -\nabla F(w_k)$
  - $|S_k|$ is fixed (local linear rate)

Stochastic Second-Order Methods

O-LBFGS[3]

When we solve the finite sum problem, define sample set as $S_k \subset \{1, \ldots, n\}$. Then we may have function and gradient as

$$\begin{cases} F^{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} f_i(w_k), \\ g_k^{S_k} = \nabla F^{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k). \end{cases}$$

The overlap sample set would be

$$O_k = S_k \cap S_{k+1} \neq \varnothing$$

To perform a stable quasi-Newton updating, we could define curvature pairs as

$$y_{k+1} = g_{k+1}^{O_k} - g_k^{O_k} \text{ and } s_{k+1} = w_{k+1} - w_k.$$

[3] Berahas, A. S., Nocedal, J., Takác, M. (2016). A multi-batch L-BFGS method for machine learning. In Advances in Neural Information Processing Systems (pp. 1055-1063).

O-LBFGS

The following part works similar as LBFGS method.
Iterates are updated by

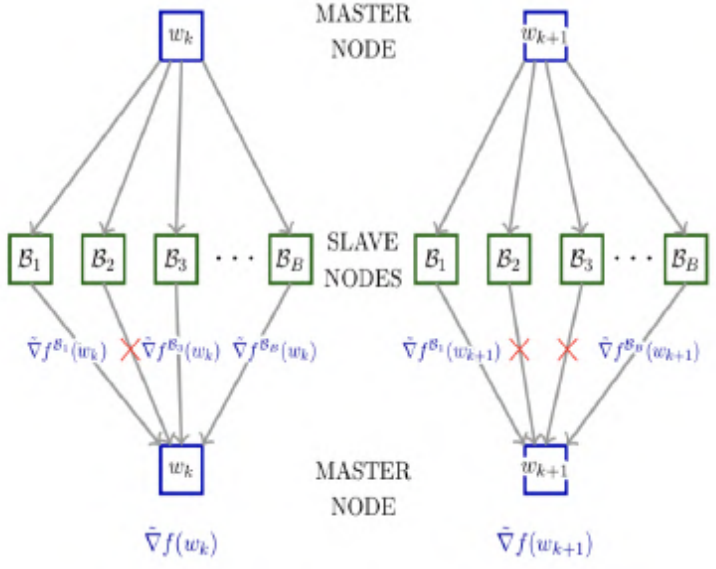
$$w_{k+1} = w_k - \alpha_k H_k g_k^{S_k}.$$

Hessian-inverse approximation matrix H_k updates as

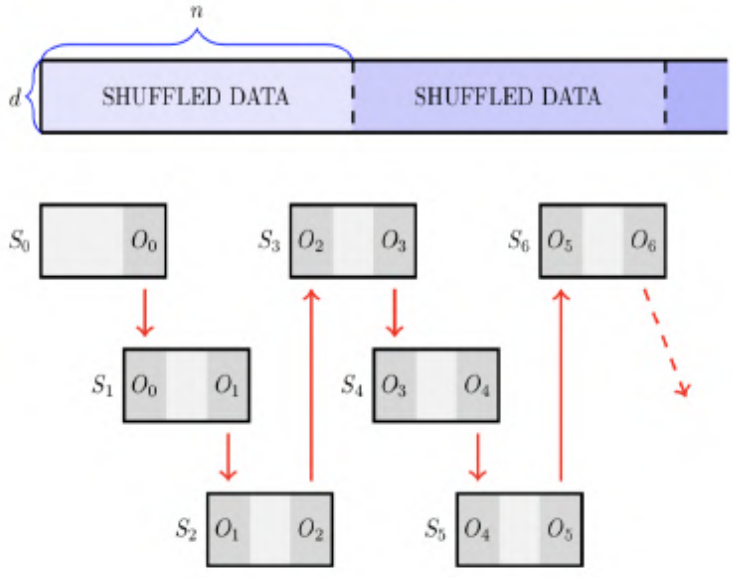
$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \quad \rho_k = \frac{1}{y_k^T s_k} \quad \text{and} \quad V_k = I - \rho_k y_k s_k^T.$$

Stochastic Second-Order Methods

O-LBFGS



(a) Fault-tolerant sampling



(b) Multi-batch sampling

Summary – Features of Second-Order Methods

Pros:

- Judiciously incorporate curvature information
- Usually implemented with larger batches (naturally calling for distributed implementations of these methods)
- Less sensitive to the values of hyper-parameters

Cons:

- (stochastic) second-order and quasi-Newton methods are more memory intensive and more expensive (per iteration)

Thank You