Deep learning via Hessian-free optimization

Chao Chen

LEHIGH UNIVERSITY

Gradient Decent

$$\min_{\theta} f(\theta)$$

To find a direction p to minimize $f(\theta)$

Approximate by Taylor expansion (1-dim):

$$f(\theta + p) = f(\theta) + f'(\theta)p + \frac{1}{2} f''(\theta)p^2 + \cdots$$

$$\approx f(\theta) + f'(\theta)p$$

(n-dim):

$$f(\theta + p) \approx f(\theta) + \nabla f(\theta)^{\mathrm{T}} p$$

p is a decreasing direction if $f(\theta + p) - f(\theta) \leq 0$

$$\nabla f(\theta)^{\mathrm{T}} p = |\nabla f(\theta)| \, |p| \cos w \leq 0$$

When $\nabla f(\theta)$ and p have the opposite directions, $\nabla f(\theta)^{\mathrm{T}} p$ takes the minimum.

Thus, we set $p = -\alpha \nabla f(\theta)$

Gradient Decent

Input: function $f(\theta)$; starting point $\theta_i = \theta_0$; step size α; tolerance ε;

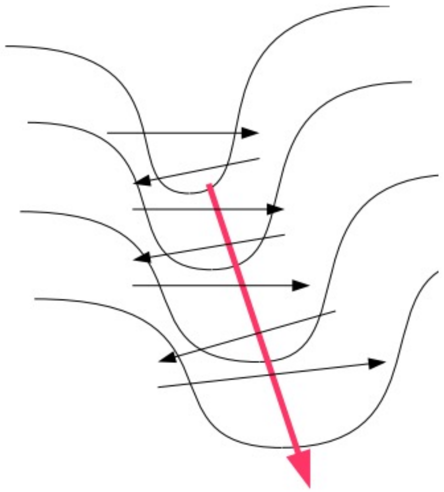
For $i = 0 \rightarrow MaxIter$:

Gradient: Compute the gradient $\nabla f(\theta)$ at θ_i;

Update: Move in the direction of gradient: $\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i)$;

If $f'(\theta_i) \leq \varepsilon$:

break



$\Longrightarrow$ The training is slow.

[Example: in a long narrow valley.]

1, The search directions have an unstable behavior in directions of high curvature.

Lowering the learning rate is **helpful**.

2, The directions of low curvature will be explored much more slowly.

Lowering the learning rate is **harmful**.

Newton's Method

Approximate by Taylor expansion (1-dim):

$$f(\theta + p) = f(\theta) + f'(\theta)p + \frac{1}{2}f''(\theta)p^2 + o(p^2)$$

To find the minimum -> set the gradient to zero:

$$f'(\theta) + f''(\theta)p = 0 \quad \Rightarrow \quad p = -\frac{f'(\theta)}{f''(\theta)}$$

Thus, we update by

$$\theta_{i+1} = \theta_i - \alpha \frac{f'(\theta_i)}{f''(\theta_i)}$$

Newton's Method

Similarly, for n-dim:

$$f(\theta + p) \approx q_\theta(p) = f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T H_f p$$

The direction p:

$$p = \left(H_f(\theta)\right)^{-1} \nabla f(\theta)$$

And we update by:

$$\theta_{i+1} = \theta_i - \alpha \left(H_f(\theta_i)\right)^{-1} \nabla f(\theta_i)$$

$\Longrightarrow$ H_f can be indefinite so $f(\theta + p)$ doesn't have minimum.

["Damping" Hessian matrix by $B = H_f + \lambda I$ for some $\lambda \geq 0$.]

$\Longrightarrow$ Hard to compute Hessian matrix and its inverse.

[Example: for n=10k, there are 10k*10k entries in H_f.]

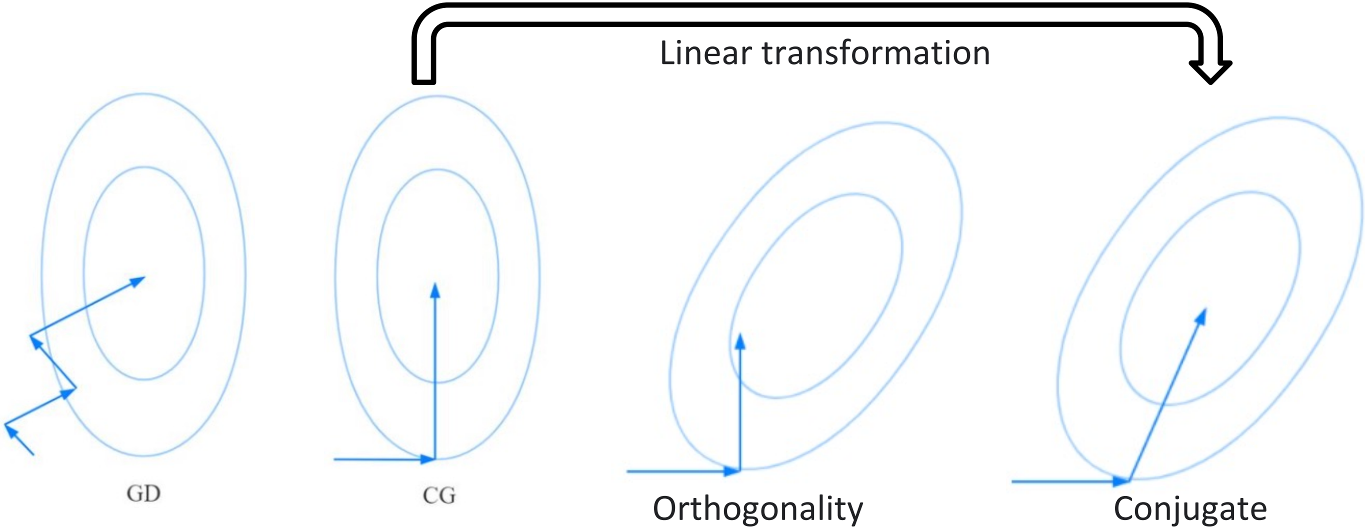
Conjugate Gradient

To avoid ruining previous efforts (GD) and computing the inverse of Hessian matrix directly (NM).

$\Longrightarrow$ Conjugate gradient (CG)

Two vectors x_i and x_j to be conjugate w.r.t. semi-definite matrix A if $x_i^T A x_j = 0$.

At most n steps to reach optimum in CG. [n is the number of dim]



Conjugate Gradient

Rewrite $f(\theta + p) \approx f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T H_f p$, as a quadratic function $h(x) = \frac{1}{2}x^T Ax + b^T x + c$

In each step of CG, we compute a) direction; b) the step size.

In step-0:

Initial direction: $d_0 = -\nabla h(x_0) = -(Ax_0 + b)$

Step size α: $g(\alpha) = h(x_i + ad_i) = \frac{1}{2}\alpha^2 d_i^T Ad_i + d_i^T(Ax_i + b)\alpha + \left(\frac{1}{2}x_i^T Ax_i + x_i^T d_i + c\right)$

set $g'(\alpha) = \left(d_i^T Ad_i\right)\alpha + d_i^T(Ax_i + b) = 0 \implies \alpha_i = -\frac{d_i^T(Ax_i+b)}{d_i^T Ad_i}$

Update: $x_1 = x_0 + \alpha_0 d_0$

Conjugate Gradient

Rewrite $f(\theta+p) \approx f(\theta) + \nabla f(\theta) p + \frac{1}{2} p^T H_f p$, as a quadratic function $h(x) = \frac{1}{2} x^T A x + b^T x + c$

In each step of CG, we compute a) direction; b) the step size.

In step-1: given $d_0 = -\nabla h(x_0) = -(Ax_0 + b)$

Find direction d_1:

Subtracting off anything that would counter-act d_i

$$d_{i+1} = -\nabla h(x_{i+1}) + \beta_i d_i$$

d_{i+1} and d_i are A-conjugate and

$$d_{i+1}^T A d_i = 0 \implies \beta_i = \frac{\nabla h(x_{i+1})^T A d_i}{d_i^T A d_i}$$

Step size α: $\alpha_i = -\frac{d_i^T (Ax_i + b)}{d_i^T A d_i}$

Update: $x_2 = x_1 + \alpha_1 d_1$

Conjugate Gradient

Rewrite $f(\theta + p) \approx f(\theta) + \nabla f(\theta) p + \frac{1}{2} p^T H_f p$, as a quadratic function $h(x) = \frac{1}{2} x^T A x + b^T x + c$

To find the minimum of a **quadratic** function h by CG:

Initialize $x_i = x_0$; $d_i = d_0 = -\nabla h(x_0)$

For $i = 0 \rightarrow MaxIter$:

Step size: compute $\alpha_i = -\frac{d_i^T (A x_i + b)}{d_i^T A d_i}$ minimizing $h(x_i + \alpha_i d_i)$

Update x_i: $x_{i+1} = x_i + \alpha_i d_i$;

Update d_i: $d_{i+1} = -\nabla f(x_{i+1}) + \beta_i d_i$ where $\beta_i = \frac{\nabla h(x_{i+1})^T A d_i}{d_i^T A d_i}$;

If converge:

break

Output x_i

[We don't compute the inverse of Hessian anymore.]

Hessian-free optimization

To find the minimum of **any** function f:

Initialize $\theta_i = \theta_0$;

For $i = 0 \rightarrow MaxIter$:

Compute the gradient $g_i = \nabla f(\theta_i)$

Compute/adjust λ

Consider the Taylor expansion at θ_i with $B_i = H_f(\theta_i) + \lambda I$

$$f(\theta + p) \approx f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T B_i p$$

Call CG to find the optimal p_i minimizing $f(\theta_i + p_i)$

$\theta_{i+1} = \theta_i + p_i$

If converge:

break

Output θ_i

[We don't compute the inverse of Hessian anymore.]

Hessian-free optimization

What is "Hessian-free"? Look the algorithms carefully:

For $i = 0 \to MaxIter$:

Compute the gradient $g_i = \nabla f(\theta_i)$

Consider the Taylor expansion at θ_i with $B_i = H_f(\theta_i) + \lambda I$

$$f(\theta + p) \approx f(\theta) + \nabla f(\theta) p + \frac{1}{2} p^T B_i p$$

Call CG to find the optimal p_i minimizing $f(\theta_i + p_i)$

…

We don't need to compute H, but Hv

For $i = 0 \to MaxIter$: [CG]

Step size: compute $\alpha_i = -\frac{d_i^T (Ax_i + b)}{d_i^T A d_i}$

Update d_i: $d_{i+1} = -\nabla h(x_{i+1}) + \beta_i d_i$ where $\beta_i = \frac{\nabla h(x_{i+1})^T A d_i}{d_i^T A d_i}$;

…

Hessian-free optimization

Two motivations for Hessian-free:

1) In CG, we only need to compute matrix-vector products Hv rather than Hessian matrix H.

2) It is relatively easy to compute Hv than H:

$$Hv = \lim_{\epsilon \to 0} \frac{\nabla f(\theta + \epsilon v) - \nabla f(\theta)}{\epsilon}$$

In this way, Hv is computed the exact value of H, there is no low-rank or diagonal approximation [compared to some quasi-Newton methods.]

Hessian-free optimization

How to compute Hv skipping H:

$$H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}, v = \begin{bmatrix} v_1 \\ \cdots \\ v_n \end{bmatrix}, x = \begin{bmatrix} x_1 \\ \cdots \\ x_n \end{bmatrix}$$

Denote $(Hv)_i$ as i-th element of Hv:

$$(Hv)_i = \begin{bmatrix} \frac{\partial^2 f}{\partial x_i \partial x_1}, & \frac{\partial^2 f}{\partial x_i \partial x_2}, & \cdots, & \frac{\partial^2 f}{\partial x_i \partial x_n} \end{bmatrix} \begin{bmatrix} v_1 \\ \cdots \\ v_n \end{bmatrix} = \sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \cdot v_j = \nabla \frac{\partial f}{\partial x_i}(x) \cdot v$$

Thus, $(Hv)_i$ is the directional derivative of $\frac{\partial f}{\partial x_i}$ along the direction v.

[By the definition: the directional derivative of f along the direction v is]

$$\nabla_v f = \lim_{\epsilon \to 0} \frac{f(x + \epsilon v) - f(x)}{\epsilon}$$

We can approximate Hv by finite differences for small ϵ:

$$Hv \approx \frac{\nabla f(x + \epsilon v) - \nabla f(x)}{\epsilon}$$

Conjugate Gradient

Rewrite $f(\theta+p) \approx f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T H_f p$, as a quadratic function $h(x) = \frac{1}{2}x^T A x + b^T x + c$

To find the minimum of a **quadratic** function h by CG:

Initialize $x_i = x_0$; $d_i = d_0 = -\nabla h(x_0)$

For $i = 0 \rightarrow MaxIter$:

Step size: compute $\alpha_i = -\frac{d_i^T(Ax_i+b)}{d_i^T A d_i}$ minimizing $h(x_i + \alpha_i d_i)$

Update x_i: $x_{i+1} = x_i + \alpha_i d_i$;

Update d_i: $d_{i+1} = -\nabla h(x_{i+1}) + \beta_i d_i$ where $\beta_i = \frac{\nabla h(x_{i+1})^T A d_i}{d_i^T A d_i}$;

If converge:

break

Output x_i

Hessian-free optimization

To find the minimum of **any** function f:

Initialize $\theta_i = \theta_0$;

For $i = 0 \rightarrow MaxIter$:

Compute the gradient $g_i = \nabla f(\theta_i)$

Compute/adjust λ

Consider the Taylor expansion at θ_i with $B_i = H_f(\theta_i) + \lambda I$

$$f(\theta + p) \approx f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T B_i p$$

Call CG to find the optimal p_i minimizing $f(\theta_i + p_i)$

$\theta_{i+1} = \theta_i + p_i$

If converge:

break

Output θ_i

Make HF suitable for ML problems

Make HF suitable for ML problems.

- How to choose λ?
- How to handle negative curvature?
- How to handle large datasets?
- How to set termination conditions?
- More tricks for enhancement?

Make HF suitable for ML problems

How to choose λ?

$$B_i = H_f(\theta_i) + \lambda I, \qquad f(\theta + p) \approx q_\theta(p) = f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T B_i p$$

As the scale of $H_f(\theta)$ constantly changes,

λ is updated by:

$$if\ \rho < \frac{1}{4}:\ \ \lambda \leftarrow \frac{3}{2}\lambda;$$

$$if\ \rho > \frac{3}{4}:\ \ \lambda \leftarrow \frac{2}{3}\lambda$$

$\rho = \frac{f(\theta+p)-f(\theta)}{q_\theta(p)-q_\theta(0)}$ measures the accuracy of q_θ

As $f(\theta + p) = f(\theta) + \nabla f(\theta)p + \frac{1}{2}p^T B_i p + o(p^2), \rho = \frac{q_\theta(p)-q_\theta(0)+o(p^2)}{q_\theta(p)-q_\theta(0)} = 1 - \frac{o(p^2)}{q_\theta(p)-q_\theta(0)}$

Make HF suitable for ML problems

How to handle negative curvature?

f is convex -> H is p.s.d. [not common]

Compute Gv rather than Hv, where G is the Gauss-Newton approximation to the Hessian H.

- G is guaranteed to be p.s.d. and any possible λ works for CG.
- G works better (better search directions) than H in practice.
- Gv is computed similarly to Hv.

Make HF suitable for ML problems

How to handle large datasets?

Inside CG, we need to keep B unchanged:

$\Longrightarrow$ Maintain invariants, such as conjugacy of search directions.

$\Longrightarrow$ Mini-batch should be unchanged inside CG.

$\Longrightarrow$ Cannot cycle mini-batches inside CG.

If the size of mini-batch is too small:

$\Longrightarrow$ Lose enough useful curvature information for good search directions.

For $i = 0 \to MaxIter$: [HF]

Generate mini-batch and corresponding approximation.

Compute the gradient $g_i = \nabla f(\theta_i)$

Consider the Taylor expansion at θ_i with $B_i = H_f(\theta_i) + \lambda I$

$$f(\theta + p) \approx f(\theta) + \nabla f(\theta) p + \frac{1}{2} p^T B_i p$$

Call CG to find the optimal p_i minimizing $f(\theta_i + p_i)$

…

For $i = 0 \to MaxIter$: [CG]

Step size: compute $\alpha_i = -\frac{d_i^T(Ax_i+b)}{d_i^T A d_i}$

Update d_i: $d_{i+1} = -\nabla f(x_{i+1}) + \beta_i d_i$ where $\beta_i = \frac{\nabla f(x_{i+1})^T A d_i}{d_i^T A d_i}$;

…

Make HF suitable for ML problems

How to set termination conditions?

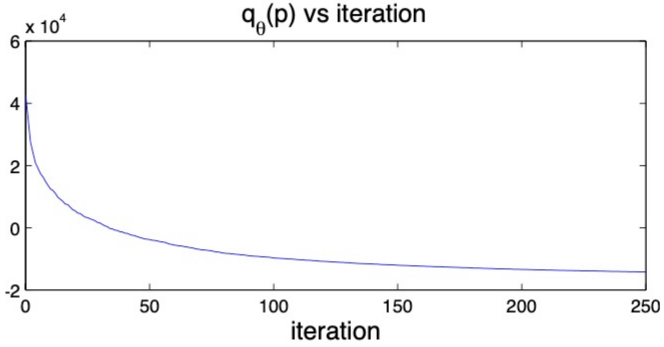
CG is guaranteed to converge after N iterations, while we cannot run till converge in practice.

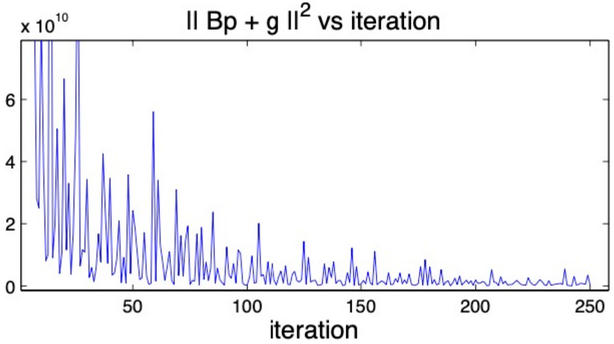
Given $q_\theta(p) = \nabla f(\theta)p + \frac{1}{2}p^T Bp$, we want to find p such that $r = \nabla f(\theta) + Bp = 0$.

Generally, termination condition is $|\nabla f(\theta) + Bp|_2 < \epsilon$, where $\epsilon = \min\left(\frac{1}{2}|\nabla f(\theta)|_2, \quad |\nabla f(\theta)|_2^{3/2}\right)$

$q_\theta(p)$ and r has the same global minimizer, while a good but sub-optimal solution for one may not good for another one.

CG is used to optimize $q_\theta(p)$ but not r.





Make HF suitable for ML problems

How to set termination conditions?

Evaluate q_θ directly. We terminate at iteration i CG if:

$$i > k; \;\; \text{and}\; q_\theta(p_i) < 0; \quad \text{and}\; \frac{q_\theta(p_i) - q_\theta(p_{i-k})}{q_\theta(p_i)} < k\epsilon$$

$k > 1$ indicates how many past iterations we use and controls the variance.

[$k = \max(10, 0.1i)$, and $\epsilon = 0.0005$ in experiments.]

Make HF suitable for ML problems

More tricks for enhancement?

Use p_{n-1} found by previous HF to initialize p_n for each CG iteration.

Accelerate CG by preconditioning.

Sparse initialization: limit the number of non-zero incoming connection weights to each unit and set the biases to 0.

Thank you

LEHIGH
UNIVERSITY