Optimization for Machine Learning

Lecture 4: Quasi-Newton Methods

S.V.N. (vishy) Vishwanathan

Purdue University
vishy@purdue.edu

July 11, 2012
The Story So Far

Two Different Philosophies

- **Online Algorithms:** Use a small subset of the data at a time and repeatedly cycle
- **Batch Optimization:** Use the entire dataset to compute gradients and function values

Gradient Based Approaches

- **Bundle Methods:** Lower bound the objective function using gradients
- **Quasi-Newton algorithms:** Use the gradients to estimate the Hessian (build a quadratic approximation of the objective)
The Story So Far

Two Different Philosophies

- **Online Algorithms:** Use a small subset of the data at a time and repeatedly cycle
- **Batch Optimization:** Use the entire dataset to compute gradients and function values

Gradient Based Approaches

- **Bundle Methods:** Lower bound the objective function using gradients
- **Quasi-Newton algorithms:** Use the gradients to estimate the Hessian (build a quadratic approximation of the objective)
Outline

1. Classical Quasi-Newton Algorithms
2. Non-smooth Problems
3. BFGS with Subgradients
4. Experiments
Broyden, Fletcher, Goldfarb, Shanno
Classical Quasi-Newton Algorithms

Standard BFGS - I

Locally Quadratic Approximation

- $\nabla J(w_t)$ is the gradient of $J$ at $w_t$
- $H_t$ is an $n \times n$ estimate of the Hessian of $J$

$$m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$$

Parameter Update

$$w_{t+1} = \arg\min_w J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$$
Local Quadratic Approximation

- $\nabla J(w_t)$ is the gradient of $J$ at $w_t$
- $H_t$ is an $n \times n$ estimate of the Hessian of $J$

$$m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t(w - w_t)$$

Parameter Update

$$w_{t+1} = \arg\min_w J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t(w - w_t)$$
Standard BFGS - 1

**Locally Quadratic Approximation**

- $\nabla J(w_t)$ is the gradient of $J$ at $w_t$
- $H_t$ is an $n \times n$ estimate of the Hessian of $J$

$$m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$$

**Parameter Update**

$$w_{t+1} = \arg\min_w J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$$

$$w_{t+1} = w_t - H_t^{-1}\nabla J(w_t)$$
Standard BFGS - I

**Locally Quadratic Approximation**
- $\nabla J(w_t)$ is the gradient of $J$ at $w_t$
- $H_t$ is an $n \times n$ estimate of the Hessian of $J$

$$m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^	op H_t(w - w_t)$$

**Parameter Update**

$$w_{t+1} = \arg\min_w J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^	op H_t(w - w_t)$$

$$w_{t+1} = w_t - \eta_t H_t^{-1} \nabla J(w_t)$$

- $\eta_t$ is a step size usually found via a line search
Standard BFGS - I

**Locally Quadratic Approximation**
- $\nabla J(w_t)$ is the gradient of $J$ at $w_t$
- $H_t$ is an $n \times n$ estimate of the Hessian of $J$
- $m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$

**Parameter Update**
- $w_{t+1} = \arg\min_w J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t)$
- $w_{t+1} = w_t - \eta_t B_t \nabla J(w_t)$
- $\eta_t$ is a step size usually found via a line search
- $B_t = H_t^{-1}$ is a symmetric PSD matrix
Standard BFGS - II

**B Matrix Update**

Update $B$ by

$$B_{t+1} = \arg\min_B ||B - B_t||_W \text{ s.t. } s_t = By_t$$

- $y_t = \nabla J(w_{t+1}) - \nabla J(w_t)$ is the difference of gradients
- $s_t = w_{t+1} - w_t$ is the difference in parameters
- This yields the update formula

$$B_{t+1} = \left( I - \frac{s_t y_t^\top}{\langle s_t, y_t \rangle} \right) B_t \left( I - \frac{y_t s_t^\top}{\langle s_t, y_t \rangle} \right) + \frac{s_t s_t^\top}{\langle s_t, y_t \rangle}$$

Limited memory variant: use a low-rank approximation to $B$
Standard BFGS - II

**B Matrix Update**

Update $B$ by

$$B_{t+1} = \arg\min_B ||B - B_t||_W \text{ s.t. } s_t = By_t$$

- $y_t = \nabla J(w_{t+1}) - \nabla J(w_t)$ is the difference of gradients
- $s_t = w_{t+1} - w_t$ is the difference in parameters
- This yields the update formula

$$B_{t+1} = \left( I - \frac{s_t y_t^T}{\langle s_t, y_t \rangle} \right) B_t \left( I - \frac{y_t s_t^T}{\langle s_t, y_t \rangle} \right) + \frac{s_t s_t^T}{\langle s_t, y_t \rangle}$$

Limited memory variant: use a low-rank approximation to $B$
Standard BFGS - II

**B Matrix Update**

Update \( B \) by

\[
B_{t+1} = \arg\min_B \|B - B_t\|_W \text{ s.t. } s_t = By_t
\]

- \( y_t = \nabla J(w_{t+1}) - \nabla J(w_t) \) is the difference of gradients
- \( s_t = w_{t+1} - w_t \) is the difference in parameters
- This yields the update formula

\[
B_{t+1} = \left( I - \frac{s_t y_t^T}{\langle s_t, y_t \rangle} \right) B_t \left( I - \frac{y_t s_t^T}{\langle s_t, y_t \rangle} \right) + \frac{s_t s_t^T}{\langle s_t, y_t \rangle}
\]

Limited memory variant: use a low-rank approximation to \( B \)
Standard BFGS - II

**B Matrix Update**

Update $B$ by

$$B_{t+1} = \arg\min_B \|B - B_t\|_W \text{ s.t. } s_t = By_t$$

- $y_t = \nabla J(w_{t+1}) - \nabla J(w_t)$ is the difference of gradients
- $s_t = w_{t+1} - w_t$ is the difference in parameters
- This yields the update formula

$$B_{t+1} = \left( I - \frac{s_t y_t^\top}{\langle s_t, y_t \rangle} \right) B_t \left( I - \frac{y_t s_t^\top}{\langle s_t, y_t \rangle} \right) + \frac{s_t s_t^\top}{\langle s_t, y_t \rangle}$$

**Limited memory variant:** use a low-rank approximation to $B$
Line Search

Wolfe Conditions

Sufficient decrease: \[ J(w_t + \eta_t d_t) \leq J(w_t) + c_1 \eta_t \langle \nabla J(w_t), d_t \rangle \]

Curvature condition: \[ \langle \nabla J(w_t + \eta_t d_t), d_t \rangle \geq c_2 \langle \nabla J(w_t), d_t \rangle , \]

where \( 0 < c_1 < c_2 < 1 \).
**Wolfe Conditions**

**Sufficient decrease:** \[ J(w_t + \eta_t d_t) \leq J(w_t) + c_1 \eta_t \langle \nabla J(w_t), d_t \rangle \]

**Curvature condition:** \[ \langle \nabla J(w_t + \eta_t d_t), d_t \rangle \geq c_2 \langle \nabla J(w_t), d_t \rangle, \]

where \(0 < c_1 < c_2 < 1\).
Outline

1. Classical Quasi-Newton Algorithms

2. Non-smooth Problems

3. BFGS with Subgradients

4. Experiments
Non-smooth Convex Optimization

- BFGS assumes that the objective function is smooth
- But, some of our losses look like this
Non-smooth Convex Optimization

- BFGS assumes that the objective function is smooth
- But, some of our losses look like this
Non-smooth Convex Optimization

- BFGS assumes that the objective function is smooth
- But, some of our losses look like this

Houston we Have a Problem!
Subgradients

A subgradient at $x'$ is any vector $s$ which satisfies

$$f(x) \geq f(x') + \langle x - x', s \rangle \text{ for all } x$$

Set of all subgradients is denoted as $\partial f(w)$
Subgradients

A subgradient at $x'$ is any vector $s$ which satisfies

$$f(x) \geq f(x') + \langle x - x', s \rangle$$

for all $x$.

Set of all subgradients is denoted as $\partial f(w)$. 
Subgradients

A subgradient at $x'$ is any vector $s$ which satisfies

$$f(x) \geq f(x') + \langle x - x', s \rangle$$

for all $x$.

Set of all subgradients is denoted as $\partial f(w)$. 
Why is Non-Smooth Optimization Hard?

The Key Difficulties

- A negative subgradient direction $\neq$ a descent direction
- Abrupt changes in function value can occur
- It is difficult to detect convergence

$$f(x) = |x|$$ and $\partial f(0) = [-1, 1]$
Why is Non-Smooth Optimization Hard?

The Key Difficulties

- A negative subgradient direction \( \neq \) a descent direction
- Abrupt changes in function value can occur
- It is difficult to detect convergence

\[ f(x) = |x| \text{ and } \partial f(0) = [-1, 1] \]
Why is Non-Smooth Optimization Hard?

The Key Difficulties

- A negative subgradient direction $\neq$ a descent direction
- Abrupt changes in function value can occur
- It is difficult to detect convergence

$f(x) = |x|$ and $\partial f(0) = [-1, 1]$
Subgradients

The Good, the Bad, and the Ugly

The subdifferential is a convex set

Not every subgradient is a descent direction!

\[ d \text{ is a descent direction if, and only if, } \langle d, s \rangle < 0 \text{ for all } s \in \partial f(x) \]
Subgradients

The Good, the Bad, and the Ugly

The subdifferential is a convex set

Not every subgradient is a descent direction!

\( d \) is a descent direction if, and only if, \( \langle d, s \rangle < 0 \) for all \( s \in \partial f(x) \)
Subgradients

The Good, the Bad, and the Ugly

The subdifferential is a convex set

Not every subgradient is a descent direction!

\[ d \text{ is a descent direction if, and only if, } \langle d, s \rangle < 0 \text{ for all } s \in \partial f(x) \]
Subgradients

The Good, the Bad, and the Ugly

The subdifferential is a convex set

Not every subgradient is a descent direction!

\( d \) is a descent direction if, and only if, \( \langle d, s \rangle < 0 \) for all \( s \in \partial f(x) \)
Outline

1. Classical Quasi-Newton Algorithms
2. Non-smooth Problems
3. BFGS with Subgradients
4. Experiments
When Working with Subgradients

Three Things Break Down

- The locally quadratic approximation is no longer well defined
- The descent direction $-B_t \nabla J(w_t)$ is not well defined
- The line search to find $\eta_t$ needs to be modified
Locally Quadratic Approximation

\[ m_t(w) = J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t (w - w_t) \]
Changing the Approximation

Locally Quadratic Approximation

\[ m_t(w) = J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t (w - w_t) \]
BFGS with Subgradients

Changing the Approximation

Locally (pseudo) Quadratic Approximation

$$m_t(w) = \sup_{s \in \partial J(w_t)} \left\{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t (w - w_t) \right\}$$
Descent Direction Finding

Locally (pseudo) Quadratic Approximation

\[ m_t(w) = \sup_{s \in \partial J(w_t)} \{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t (w - w_t) \} \]

\[ P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)
Descent Direction Finding

Locally (pseudo) Quadratic Approximation

\[ w_{t+1} = \arg\min_w \sup_{s \in \partial J(w_t)} \left\{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2} (w - w_t)^\top H_t (w - w_t) \right\} \]

\[ P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)
Descent Direction Finding

**Locally (pseudo) Quadratic Approximation**

\[ w_{t+1} = \arg\min_w \sup_{s \in \partial J(w_t)} \{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2} (w - w_t)^T H_t (w - w_t) \} \]

\[ w_{t+1} = \arg\min_w \frac{1}{2} (w - w_t)^T H_t (w - w_t) + \xi \]

s.t. \( J(w_t) + \langle s, w - w_t \rangle \leq \xi \) for all \( s \in \partial J(w_t) \)

\[ P_k(d) = \min_d \frac{1}{2} d^T H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)
Descent Direction Finding

**Locally (pseudo) Quadratic Approximation**

\[ w_{t+1} = \arg\min_w \sup_{s \in \partial J(w_t)} \{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t) \} \]

\[ w_{t+1} = \arg\min_w \frac{1}{2}(w - w_t)^\top H_t(w - w_t) + \xi \]

s.t. \( J(w_t) + \langle s_i, w - w_t \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)

**Projection**

\[ P_k(d) = \min_d \frac{1}{2}d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)
**Locally (pseudo) Quadratic Approximation**

\[ w_{t+1} = \arg\min_{w} \sup_{s \in \partial J(w_t)} \left\{ J(w_t) + \langle s, w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t(w - w_t) \right\} \]

\[ w_{t+1}^k = \arg\min_{w} \frac{1}{2}(w - w_t)^\top H_t(w - w_t) + \xi \]

subject to \( J(w_t) + \langle s_i, w - w_t \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)

\[ P_k(d) = \min_{d} \frac{1}{2} d^\top H_t d + \xi \]

subject to \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)
Descent Direction Finding

Locally (pseudo) Quadratic Approximation

\[ P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)

Parameter Update

Require: \( \text{maxitr} \)

1: \( k \leftarrow 1, d_1 \leftarrow -B_t s_1 \) for some arbitrary \( s_1 \in \partial J(w_t) \)
2: repeat
3: \( s_k = \text{argsup}_{s \in \partial J(w_t)} \langle s, d_k \rangle \)
4: if \( \langle s_k, d_k \rangle < 0 \) then
5: return \( d_k \)
6: else
7: \( d_{k+1} = \text{argmin}_d P_k(d), k \leftarrow k + 1 \)
8: end if
9: until \( k \geq \text{maxitr} \)
Descent Direction Finding

Locally (pseudo) Quadratic Approximation

\[ P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)

Parameter Update

Require: \( maxitr \)

1: \( k \leftarrow 1, d_1 \leftarrow -B_t s_1 \) for some arbitrary \( s_1 \in \partial J(w_t) \)
2: repeat
3: \( s_k = \text{argsup}_{s \in \partial J(w_t)} \langle s, d_k \rangle \)
4: if \( \langle s_k, d_k \rangle < 0 \) then
5: \( \text{return } d_k \)
6: else
7: \( d_{k+1} = \text{argmin}_d P_k(d) \), \( k \leftarrow k + 1 \)
8: end if
9: until \( k \geq maxitr \)
BFGS with Subgradients

Descent Direction Finding

**Locally (pseudo) Quadratic Approximation**

\[
P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi
\]

\[\text{s.t. } J(w_t) + \langle s_i, d \rangle \leq \xi \text{ for } s_1 \ldots s_k \in \partial J(w_t)\]

**Parameter Update**

**Require**: \(\text{maxitr}\)

1: \(k \leftarrow 1, d_1 \leftarrow -B_t s_1\) for some arbitrary \(s_1 \in \partial J(w_t)\)
2: \(\text{repeat}\)
3: \(s_k = \text{argsup}_{s \in \partial J(w_t)} \langle s, d_k \rangle\)
4: \(\text{if } \langle s_k, d_k \rangle < 0 \text{ then}\)
5: \(\text{return } d_k\)
6: \(\text{else}\)
7: \(d_{k+1} = \text{argmin}_d P_k(d), k \leftarrow k + 1\)
8: \(\text{end if}\)
9: \(\text{until } k \geq \text{maxitr}\)
Descent Direction Finding

Locally (pseudo) Quadratic Approximation

\[ P_k(d) = \min_d \frac{1}{2} d^\top H_t d + \xi \]

s.t. \( J(w_t) + \langle s_i, d \rangle \leq \xi \) for \( s_1 \ldots s_k \in \partial J(w_t) \)

Parameter Update

Require: \( maxitr \)

1: \( k \leftarrow 1, d_1 \leftarrow -B_t s_1 \) for some arbitrary \( s_1 \in \partial J(w_t) \)
2: repeat
3: \( s_k = \operatorname{argsup}_{s \in \partial J(w_t)} \langle s, d_k \rangle \)
4: if \( \langle s_k, d_k \rangle < 0 \) then
5: return \( d_k \)
6: else
7: \( d_{k+1} = \operatorname{argmin}_d P_k(d), k \leftarrow k + 1 \)
8: end if
9: until \( k \geq maxitr \)
The objective function $J(w) := \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \langle x_i, w \rangle)$
The objective function $J(w) := \frac{\lambda}{2} ||w||^2 + \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \langle x_i, w \rangle)$

Plotted along any direction looks like this
The Hinge Loss Revisited

The objective function $J(w) := \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \langle x_i, w \rangle)$

When zoomed in looks like this
The objective function $J(w) := \frac{\lambda}{2} \| w \|^{2} + \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_{i}\langle x_{i}, w \rangle)$

Piecewise quadratic $\implies$ exact line search in linear time.
Outline

1. Classical Quasi-Newton Algorithms
2. Non-smooth Problems
3. BFGS with Subgradients
4. Experiments
Why Not Just Use BFGS?

- Leukemia: 38 train, 34 test, 7129 dimensions
- real-sim: 57763 train, 14438 test, 20958 dimensions

CPU Time vs Objective Function Value

Leukemia $(\lambda = 10^{-1})$

Real – sim $(\lambda = 10^{-5})$
subBFGS: Results on a Simple Problem

The Problem

\[ J(w_1, w_2) = 10 \times |w_1| + |w_2| \]
subBFGS: Results on a Simple Problem

The Problem

\[ J(w_1, w_2) = 10 \times |w_1| + |w_2| \]

Particularly evil problem for BFGS!
subBFGS: Results on a Simple Problem

BFGS

- Hops from orthant to orthant
- Slow convergence :(
subBFGS: Results on a Simple Problem

- Exact line search
- Converges in 2 iterations :)
Are Our Modifications Helpful?

- INEX: 6053 train, 6054 test, 167295 dimensions, 18 classes.
- TMC2007: 21519 train, 7077 test, 30438 dimensions, 22 classes.

**CPU Time vs Objective Function Value**

![Graph showing CPU time vs objective function value for INEX and TMC2007 datasets.]
Experiments

On a Simple Toy Problem

BFGS Approximation to the Objective Function and Gradient

- BFGS Quadratic Model
- Piecewise Linear Function

- Gradient of BFGS Model
- Piecewise Constant Gradient
Results on some standard datasets

- Covertype: 522911 train, 58101 test, 54 dimensions.

**CPU Time vs Objective Function Value**

![Graph showing CPU time vs objective function value for Covertype dataset with various optimization methods.](image)
Experiments

Results on some standard datasets

- CCAT: 781265 train, 23149 test, 47236 dimensions.

**CPU Time vs Objective Function Value**

![Graph showing CPU Time vs Objective Function Value for different algorithms like BMRM, OCAS, and subLBFGS for CCAT dataset with λ = 10^{-6}.](image-url)
The Pros and Cons of subBFGS

**Quasi-Newton Philosophy**
- Use the gradients to build a quadratic approximation
- Initially this approximation is a good fit
  - Rapid initial progress
- Closer to the optimum the hinges matter
  - Progress slows down near the optimum

**Line Search**
- subBFGS requires a line search which fulfills Wolfe conditions
- For binary and multiclass hinge loss an exact line search is cheap
- Can we do a cheap line search for structured losses?
References