Online Limited-Memory Quasi-Newton Training of Support Vector Machines

Jin Yu  Simon Günter
S.V.N. Vishwanathan  Nicol N. Schraudolph

March 30, 2007
Optimization in the Primal

- Regularized risk minimization

\[
\min_{\mathbf{w}} J(\mathbf{w}) := \frac{c}{2} \| \mathbf{w} \|^2 + \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, \mathbf{z}_i, \mathbf{w})
\]

- Standard (batch) optimization methods, e.g. GD, Newton’s method, (L)BFGS, use the following update

\[
\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \nabla J(\mathbf{w}_t)
\]

- Online optimization methods work with stochastic approximations

\[
J_t(\mathbf{w}) := \frac{c}{2} \| \mathbf{w} \|^2 + l(\mathbf{x}_t, \mathbf{z}_t, \mathbf{w})
\]

- Can also use online methods as initializer for batch methods
Optimization in the Primal

- Regularized risk minimization

\[
\min_w J(w) := \frac{c}{2}||w||^2 + \frac{1}{|T|} \sum_{i=1}^{T} l(x_i, z_i, w)
\]

- Standard (batch) optimization methods, \textit{e.g.} GD, Newton’s method, (L)BFGS, use the following update

\[
w_{t+1} = w_t - \eta_t B_t \nabla J(w_t)
\]

- Online optimization methods work with stochastic approximations

\[
J_t(w) := \frac{c}{2}||w||^2 + l(x_t, z_t, w)
\]

- Can also use online methods as \textit{initializer} for batch methods
Optimization in the Primal

• Regularized risk minimization

$$\min_{\mathbf{w}} \ J(\mathbf{w}) := \frac{c}{2} ||\mathbf{w}||^2 + \frac{1}{|T|} \sum_{i=1}^{T} l(x_i, z_i, \mathbf{w})$$

• Standard (batch) optimization methods, e.g. GD, Newton’s method, (L)BFGS, use the following update

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \nabla J(\mathbf{w}_t)$$

• Online optimization methods work with stochastic approximations

$$J_t(\mathbf{w}) := \frac{c}{2} ||\mathbf{w}||^2 + l(x_t, z_t, \mathbf{w})$$

• Can also use online methods as initializer for batch methods
Optimization in the Primal

- Regularized risk minimization

\[
\min_{w} J(w) := \frac{c}{2} \|w\|^2 + \frac{1}{|T|} \sum_{i=1}^{T} l(x_i, z_i, w)
\]

- Standard (batch) optimization methods, e.g. GD, Newton’s method, (L)BFGS, use the following update

\[
w_{t+1} = w_t - \eta_t B_t \nabla J(w_t)
\]

- Online optimization methods work with stochastic approximations

\[
J_t(w) := \frac{c}{2} \|w\|^2 + l(x_t, z_t, w)
\]

- Can also use online methods as initializer for batch methods
BFGS algorithm

• BFGS parameter update: \( w_{t+1} = w_t - \eta_t B_t g_t \).

• Maintain symm. pos. def. matrix \( B \approx H^{-1} \) by

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

\[
y_t := g_{t+1} - g_t; \ s_t := w_{t+1} - w_t
\]

• Limited-memory version (LBFGS) stores \( m \) pairs of \( (s, y) \)

• LBFGS obtains \( B_t g_t \) via matrix-free update
**BFGS algorithm**

- BFGS parameter update: \( w_{t+1} = w_t - \eta_t B_t g_t \).
- Maintain symm. pos. def. matrix \( B \approx H^{-1} \) by
  
  \[
  B_{t+1} = \arg \min_B \|B - B_t\|_W, \text{ s.t. } s_t = B y_t
  \]
  
  \[
  y_t := g_{t+1} - g_t; \quad s_t := w_{t+1} - w_t
  \]

- Limited-memory version (LBFGS) stores \( m \) pairs of \((s, y)\)

- LBFGS obtains \( B_t g_t \) via matrix-free update
BFGS algorithm

- **BFGS parameter update:** $w_{t+1} = w_t - \eta_t B_t g_t$.

- Maintain symm. pos. def. matrix $B \approx H^{-1}$ by

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

  $$y_t := g_{t+1} - g_t; \quad s_t := w_{t+1} - w_t$$

- Limited-memory version (LBFGS) stores $m$ pairs of $(s, y)$

- LBFGS obtains $B_t g_t$ via matrix-free update
BFGS algorithm

- BFGS parameter update: $w_{t+1} = w_t - \eta_t B_t g_t$.

- Maintain symm. pos. def. matrix $B \approx H^{-1}$ by

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

$$y_t := g_{t+1} - g_t; \quad s_t := w_{t+1} - w_t$$

- Limited-memory version (LBFGS) stores $m$ pairs of $(s, y)$

- LBFGS obtains $B_t g_t$ via matrix-free update
Why Support Vector Machines (SVMs)

· SVMs perform **margin-maximization**
  
  · Non-linear SVMs use kernel trick: \( k(\cdot, \cdot) \leftarrow \langle \cdot, \cdot \rangle \)

Can **kernelize** classical limited-memory BFGS (LBFGS) algorithm
Why Support Vector Machines (SVMs)

- SVMs perform **margin-maximization**
- Non-linear SVMs use **kernel trick**: $k(\cdot, \cdot) \leftarrow \langle \cdot, \cdot \rangle$

Can kernelize classical limited-memory BFGS (LBFGS) algorithm
Why Support Vector Machines (SVMs)

- SVMs perform **margin-maximization**
- Non-linear SVMs use **kernel trick**: \( k(\cdot, \cdot) \leftarrow \langle \cdot, \cdot \rangle \)

Can **kernelize** classical limited-memory BFGS (LBFGS) algorithm
LBFGS Algorithm

**LBFGS Direction Update**

\[ s_t := -\eta_t g_t; \]

\[ \text{for } i := 1, 2, \ldots, \min(t, m) : \]

\[ a_i = \varrho_{t-i} \langle s_{t-i}, s_t \rangle; \]

\[ s_t := s_t - a_i y_{t-i}; \]

\[ s_t := s_t / \langle \varrho_{t-1} \langle y_{t-1}, y_{t-1} \rangle \rangle \]

\[ b = \varrho_{t-i} \langle y_{t-i}, s_t \rangle; \]

\[ s_t := s_t + (a_i - b) s_{t-i}; \]

\[ w_{t+1} = w_t + s_t; \]

\[ y_t = g_{t+1} - g_t; \]

\[ \varrho_t = 1 / \langle s_t, y_t \rangle \]

maintain ring buffer of last \( m \) values of \( s_t, y_t \) vectors; scalar \( \varrho_t \)

Can Use Kernel Trick

Note: only inner products and linear combinations

New

can do it online using online LBFGS (Schraudolph et al., AISTATS 2007)
LBFGS Algorithm

**LBFGS Direction Update**

... 

For $t := 0, 1, \ldots$:

1. $s_t := -\eta_t g_t$;
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \varrho_{t-i} \langle s_{t-i}, s_t \rangle$;
   2. $s_t := s_t - a_i y_{t-i}$;
3. $s_t := s_t/\left(\varrho_{t-1} \langle y_{t-1}, y_{t-1} \rangle\right)$
   1. $b = \varrho_{t-i} \langle y_{t-i}, s_t \rangle$;
   2. $s_t := s_t + (a_i - b) s_{t-i}$;

$w_{t+1} = w_t + s_t$;

$y_t = g_{t+1} - g_t$;

$\varrho_t = 1/\langle s_t, y_t \rangle$

maintain ring buffer of last $m$ values of $s_t, y_t$ vectors; scalar $\varrho_t$

Can Use Kernel Trick

Note: only inner products and linear combinations

New

can do it online using online LBFGS (Schraudolph et al., AISTATS 2007)
**LBFGS Algorithm**

**LBFGS Direction Update**

...  
For $t := 0, 1, \ldots$:

1. $s_t := -\eta_t g_t$;
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \varrho_{t-i} \langle s_{t-i}, s_t \rangle$;
   2. $s_t := s_t - a_i y_{t-i}$;
3. $s_t := s_t / (\varrho_{t-1} \langle y_{t-1}, y_{t-1} \rangle)$

1. $b = \varrho_{t-i} \langle y_{t-i}, s_t \rangle$;
2. $s_t := s_t + (a_i - b) s_{t-i}$;

$w_{t+1} = w_t + s_t$;
$y_t = g_{t+1} - g_t$; $\varrho_t = 1 / \langle s_t, y_t \rangle$

- Can Use Kernel Trick  
  **Note:** only inner products and linear combinations

- New  
  can do it **online** using online LBFGS (Schraudolph et al., AISTATS 2007)

- Maintain ring buffer of last $m$ values of $s_t, y_t$ vectors; scalar $\varrho_t$
Online SVM (aka NORMA)

- Stochastic gradient (Kivinen et al., IEEE TSP 2004)
  - objective: $J(f) = \frac{1}{|T|} \sum_{i=1}^{T} l(x_i, z_i, f) + \frac{c}{2} \|f\|_H^2$, $f \in \mathcal{H}$
  - stochastic gradient: $g_t = \partial_f l(x_t, z_t, f_t) + cf_t$
  - kernel expansion: $f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((x_i, z), \cdot)$
  - coefficient update:
    $$f_{t+1} = f_t - \eta_t g_t \quad \alpha_t = \begin{bmatrix} (1 - \eta_t c)\alpha_{t-1} \\ -\eta_t \xi_t^T \end{bmatrix}$$

- SMD gain adaptation (SVMD) (Vishwanathan et al., JMLR 2006)

Our Approach

online LBFGS method in high-dimensional feature space (e.g. RKHS)
Online SVM (*aka* NORMA)

- Stochastic gradient (Kivinen *et al*., IEEE TSP 2004)
  - objective: \( J(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} l(x_i, z_i, f) + \frac{\zeta}{2} \|f\|^2_H, \quad f \in \mathcal{H} \)
  - stochastic gradient: \( g_t = \partial_f l(x_t, z_t, f_t) + cf_t \)
  - kernel expansion: \( f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((x_i, z), \cdot) \)
  - coefficient update:
    \[
    f_{t+1} = f_t - \eta_t g_t \quad \alpha_t = \begin{bmatrix} (1 - \eta_t c) \alpha_{t-1} \\ -\eta_t \xi_t^\top \end{bmatrix}
    \]

- SMD gain adaptation (SVMD) (Vishwanathan *et al*., JMLR 2006)

Our Approach

*online LBFGS method in high-dimensional feature space (e.g. RKHS)*
Online SVM (aka NORMA)

- Stochastic gradient (Kivinen et al., IEEE TSP 2004)
  - objective: \( J(f) = \frac{1}{|T|} \sum_{i=1}^{\lvert T \rvert} l(x_i, z_i, f) + \frac{c}{2} \| f \|_H^2, \ f \in \mathcal{H} \)
  - stochastic gradient: \( g_t = \partial_f l(x_t, z_t, f_t) + cf_t \)
  - kernel expansion: \( f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((x_i, z), \cdot) \)
  - coefficient update:
    \[
    f_{t+1} = f_t - \eta_t g_t \\
    \alpha_t = \begin{bmatrix}
    (1 - \eta_t c) \alpha_{t-1} \\
    -\eta_t \xi_t^T
    \end{bmatrix}
    \]

- SMD gain adaptation (SVMD) (Vishwanathan et al., JMLR 2006)

Our Approach

online LBFGS method in high-dimensional feature space (e.g. RKHS)
Online Kernel LBFGS (okLBFGS)

**okLBFGS Direction Update**

...  
For $t := 0, 1, \ldots:$

1. $s_t := -\eta_t g_t$;  
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \varrho_t - i \langle s_{t-i}, s_t \rangle_{\mathcal{H}}$;  
   2. $s_t := s_t - a_i y_{t-i};$  
3. $s_t := s_t / \left( \varrho_t - 1 \langle y_{t-1}, y_{t-1} \rangle_{\mathcal{H}} \right)$  
   1. $b = \varrho_t - i \langle y_{t-i}, s_t \rangle_{\mathcal{H}}$;  
   2. $s_t := s_t + (a_i - b)s_{t-i};$

$f_{t+1} = f_t + s_t; 
 y_t = g_{t+1} - g_t; \varrho_t = 1 / \langle s_t, y_t \rangle_{\mathcal{H}}$

**Lifted to RKHS ($\mathcal{H}$)**

1. Maintain ring buffer of last $m$ pairs of $s_t, y_t$ functions  
2. Replace $\langle \cdot, \cdot \rangle$ with a $k(\cdot, \cdot)$  
3. Special care should be taken for $y$ computation (ref. Schraudolph et al. AISTATS 2007)  
4. Actually, we only update expansion coefficients
Online Kernel LBFGS (okLBFGS)

**okLBFGS Direction Update**

... 

For $t := 0, 1, \ldots$:

1. $s_t := -\eta_t g_t$;
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \varrho_{t-i} \langle s_{t-i}, s_t \rangle_{\mathcal{H}}$;
   2. $s_t := s_t - a_i y_{t-i}$;
3. $s_t := s_t / \left( \varrho_{t-1} \langle y_{t-1}, y_{t-1} \rangle_{\mathcal{H}} \right)$
   1. $b = \varrho_{t-i} \langle y_{t-i}, s_t \rangle_{\mathcal{H}}$;
   2. $s_t := s_t + (a_i - b) s_{t-i}$;

$f_{t+1} = f_t + s_t$;  
$y_t = g_{t+1} - g_t$;  
$\varrho_t = 1 / \langle s_t, y_t \rangle_{\mathcal{H}}$

**Lifted to RKHS ($\mathcal{H}$)**

1. Maintain ring buffer of last $m$ pairs of $s_t, y_t$ functions
2. Replace $\langle \cdot, \cdot \rangle$ with a $k:(\cdot, \cdot)$
3. Special care should be taken for $y$ computation (ref. Schraudolph et al. AISTATS 2007)
4. Actually, we only update expansion coefficients
**Online Kernel LBFGS (okLBFGS)**

**okLBFGS Direction Update**

...  
For $t := 0, 1, \ldots$:

1. $s_t := -\eta_t g_t$;
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \rho_{t-i} \langle s_{t-i}, s_t \rangle_{\mathcal{H}}$;
   2. $s_t := s_t - a_i y_{t-i}$;
3. $s_t := s_t / (\rho_{t-1} \langle y_{t-1}, y_{t-1} \rangle_{\mathcal{H}})$
   1. $b = \rho_{t-i} \langle y_{t-i}, s_t \rangle_{\mathcal{H}}$;
   2. $s_t := s_t + (a_i - b) s_{t-i}$;
4. $f_{t+1} = f_t + s_t$;
5. $y_t = g_{t+1} - g_t; \ \rho_t = 1 / \langle s_t, y_t \rangle_{\mathcal{H}}$

**Lifted to RKHS ($\mathcal{H}$)**

1. Maintain ring buffer of last $m$ pairs of $s_t, y_t$ functions
2. Replace $\langle \cdot, \cdot \rangle$ with a $k(\cdot, \cdot)$
3. Special care should be taken for $y$ computation (ref. Schraudolph et al. AISTATS 2007)
4. Actually, we only update expansion coefficients
Online Kernel LBFGS (okLBFGS)

**okLBFGS Direction Update**

... 

For $t := 0, 1, \ldots$:

1. $s_t := -\eta_t g_t$;
2. for $i := 1, 2, \ldots, \min(t, m)$:
   1. $a_i = \varrho_{t-i} \langle s_{t-i}, s_t \rangle_\mathcal{H}$;
   2. $s_t := s_t - a_i y_{t-i}$;
3. $s_t := s_t / (\varrho_{t-1} \langle y_{t-1}, y_{t-1} \rangle_\mathcal{H})$
   1. $b = \varrho_{t-i} \langle y_{t-i}, s_t \rangle_\mathcal{H}$;
   2. $s_t := s_t + (a_i - b) s_{t-i}$;

$f_{t+1} = f_t + s_t$;

$y_t = g_{t+1} - g_t$; $\varrho_t = 1 / \langle s_t, y_t \rangle_\mathcal{H}$

**Lifted to RKHS ($\mathcal{H}$)**

1. Maintain ring buffer of last $m$ pairs of $s_t, y_t$ functions
2. Replace $\langle \cdot, \cdot \rangle$ with a $k(\cdot, \cdot)$
3. Special care should be taken for $y$ computation (ref. Schraudolph et al. AISTATS 2007)
4. Actually, we only update expansion coefficients
USPS Binary

- Compare to NORMA and SVMD
- Single pass through data
- okLBFGS beats SVMD in only 10% of the data!

Figure: USPS Binary Class. (0-4 vs. 5-9)
USPS Counting Sequence

Figure: USPS 10-way Multilass.

- Rather evil digit rearrangement
- Single pass through data
- okLBFGS again beats SVMD in only 20% of the data!
Scaling up to MNIST

- Online SVM diverges
- okLBFGS beats SVMD initially and asymptotically

Figure: MNIST 10-way Multiclass.
Conclusions

• What we have achieved:
  ▶ Quasi-Newton method (LBFGS) in high-dimensional feature space (RKHS)
  ▶ Use our new online variant of LBFGS

• What still needs to be done:
  ▶ Better step size management
  ▶ Applications for batch kernel LBFGS
Conclusions

• What we have achieved:
  ▶ Quasi-Newton method (LBFGS) in high-dimensional feature space (RKHS)
  ▶ Use our new online variant of LBFGS

• What still needs to be done:
  ▶ Better step size management
  ▶ Applications for batch kernel LBFGS