

# 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

$$\underset{\mathbf{w}}{\text{minimize}} \quad J(\mathbf{w}) := \frac{c}{2} \|\mathbf{w}\|^2 + \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, \mathbf{w})$$

- 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

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

- Can also use online methods as **initializer** for batch methods

# Optimization in the Primal

- Regularized risk minimization

$$\underset{\mathbf{w}}{\text{minimize}} \quad J(\mathbf{w}) := \frac{c}{2} \|\mathbf{w}\|^2 + \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, \mathbf{w})$$

- 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

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

- Can also use online methods as **initializer** for batch methods

# Optimization in the Primal

- Regularized risk minimization

$$\underset{\mathbf{w}}{\text{minimize}} \quad J(\mathbf{w}) := \frac{c}{2} \|\mathbf{w}\|^2 + \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, \mathbf{w})$$

- 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

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

- Can also use online methods as **initializer** for batch methods

# Optimization in the Primal

- Regularized risk minimization

$$\underset{\mathbf{w}}{\text{minimize}} \quad J(\mathbf{w}) := \frac{c}{2} \|\mathbf{w}\|^2 + \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, \mathbf{w})$$

- 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

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

- Can also use online methods as [initializer](#) for batch methods

# BFGS algorithm

- BFGS parameter update:  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \mathbf{g}_t$ .
- Maintain symm. pos. def. matrix  $\mathbf{B} \approx \mathbf{H}^{-1}$  by

$$\mathbf{B}_{t+1} = \arg \min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_t\|_{\mathcal{W}}, \text{ s.t. } \mathbf{s}_t = \mathbf{B} \mathbf{y}_t$$

$$\mathbf{y}_t := \mathbf{g}_{t+1} - \mathbf{g}_t; \mathbf{s}_t := \mathbf{w}_{t+1} - \mathbf{w}_t$$

- Limited-memory version (LBFGS) stores  $m$  pairs of  $(\mathbf{s}, \mathbf{y})$
- LBFGS obtains  $\mathbf{B}_t \mathbf{g}_t$  via [matrix-free](#) update

# BFGS algorithm

- BFGS parameter update:  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \mathbf{g}_t$ .
- Maintain symm. pos. def. matrix  $\mathbf{B} \approx \mathbf{H}^{-1}$  by

$$\mathbf{B}_{t+1} = \arg \min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_t\|_{\mathbf{W}}, \text{ s.t. } \mathbf{s}_t = \mathbf{B} \mathbf{y}_t$$

$$\mathbf{y}_t := \mathbf{g}_{t+1} - \mathbf{g}_t; \quad \mathbf{s}_t := \mathbf{w}_{t+1} - \mathbf{w}_t$$

- Limited-memory version (LBFGS) stores  $m$  pairs of  $(\mathbf{s}, \mathbf{y})$
- LBFGS obtains  $\mathbf{B}_t \mathbf{g}_t$  via [matrix-free](#) update

# BFGS algorithm

- BFGS parameter update:  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \mathbf{g}_t$ .
- Maintain symm. pos. def. matrix  $\mathbf{B} \approx \mathbf{H}^{-1}$  by

$$\mathbf{B}_{t+1} = \arg \min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_t\|_{\mathbf{W}}, \text{ s.t. } \mathbf{s}_t = \mathbf{B} \mathbf{y}_t$$

$$\mathbf{y}_t := \mathbf{g}_{t+1} - \mathbf{g}_t; \mathbf{s}_t := \mathbf{w}_{t+1} - \mathbf{w}_t$$

- Limited-memory version (LBFGS) stores  $m$  pairs of  $(\mathbf{s}, \mathbf{y})$
- LBFGS obtains  $\mathbf{B}_t \mathbf{g}_t$  via [matrix-free](#) update

# BFGS algorithm

- BFGS parameter update:  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{B}_t \mathbf{g}_t$ .
- Maintain symm. pos. def. matrix  $\mathbf{B} \approx \mathbf{H}^{-1}$  by

$$\mathbf{B}_{t+1} = \arg \min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_t\|_{\mathbf{W}}, \text{ s.t. } \mathbf{s}_t = \mathbf{B} \mathbf{y}_t$$

$$\mathbf{y}_t := \mathbf{g}_{t+1} - \mathbf{g}_t; \mathbf{s}_t := \mathbf{w}_{t+1} - \mathbf{w}_t$$

- Limited-memory version (LBFGS) stores  $m$  pairs of  $(\mathbf{s}, \mathbf{y})$
- LBFGS obtains  $\mathbf{B}_t \mathbf{g}_t$  via [matrix-free](#) update

# Why Support Vector Machines (SVMs)

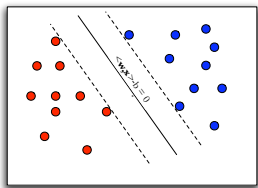


Figure: linear SVMs

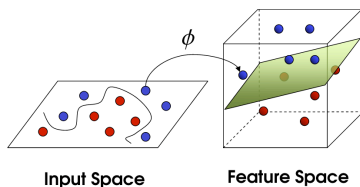


Figure: non-linear SVMs

- 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)

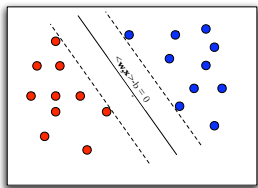


Figure: linear SVMs

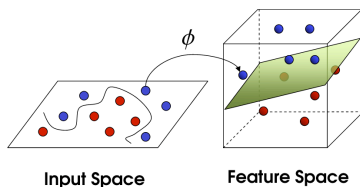


Figure: non-linear SVMs

- 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)

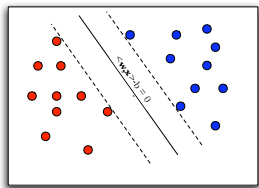


Figure: linear SVMs

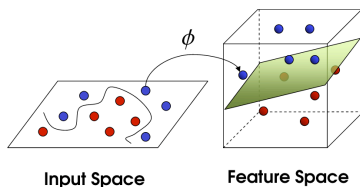


Figure: non-linear SVMs

- 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

...

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

- ①  $\mathbf{s}_t := -\eta_t \mathbf{g}_t$ ;
- ② for  $i := 1, 2, \dots, \min(t, m)$  :
  - ①  $a_i = \rho_{t-i} \langle \mathbf{s}_{t-i}, \mathbf{s}_t \rangle$ ;
  - ②  $\mathbf{s}_t := \mathbf{s}_t - a_i \mathbf{y}_{t-i}$ ;
- ③  $\mathbf{s}_t := \mathbf{s}_t / (\rho_{t-1} \langle \mathbf{y}_{t-1}, \mathbf{y}_{t-1} \rangle)$ 
  - ①  $b = \rho_{t-1} \langle \mathbf{y}_{t-1}, \mathbf{s}_t \rangle$ ;
  - ②  $\mathbf{s}_t := \mathbf{s}_t + (a_i - b) \mathbf{s}_{t-1}$ ;

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t;$$

$$\mathbf{y}_t = \mathbf{g}_{t+1} - \mathbf{g}_t; \rho_t = 1 / \langle \mathbf{s}_t, \mathbf{y}_t \rangle$$

maintain ring buffer of last  $m$  values of  $\mathbf{s}_t, \mathbf{y}_t$  vectors; scalar  $\rho_t$

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  $t := 0, 1, \dots$ :

- ①  $\mathbf{s}_t := -\eta_t \mathbf{g}_t$ ;

- ② for  $i := 1, 2, \dots, \min(t, m)$ :

- ①  $a_i = \rho_{t-i} \langle \mathbf{s}_{t-i}, \mathbf{s}_t \rangle$ ;

- ②  $\mathbf{s}_t := \mathbf{s}_t - a_i \mathbf{y}_{t-i}$ ;

- ③  $\mathbf{s}_t := \mathbf{s}_t / (\rho_{t-1} \langle \mathbf{y}_{t-1}, \mathbf{y}_{t-1} \rangle)$

- ①  $b = \rho_{t-i} \langle \mathbf{y}_{t-i}, \mathbf{s}_t \rangle$ ;

- ②  $\mathbf{s}_t := \mathbf{s}_t + (a_i - b) \mathbf{s}_{t-i}$ ;

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t;$$

$$\mathbf{y}_t = \mathbf{g}_{t+1} - \mathbf{g}_t; \rho_t = 1 / \langle \mathbf{s}_t, \mathbf{y}_t \rangle$$

maintain ring buffer of last  $m$  values of  $\mathbf{s}_t, \mathbf{y}_t$  vectors; scalar  $\rho_t$

### 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  $t := 0, 1, \dots$ :

- ①  $\mathbf{s}_t := -\eta_t \mathbf{g}_t$ ;
- ② for  $i := 1, 2, \dots, \min(t, m)$  :
  - ①  $a_i = \rho_{t-i} \langle \mathbf{s}_{t-i}, \mathbf{s}_t \rangle$ ;
  - ②  $\mathbf{s}_t := \mathbf{s}_t - a_i \mathbf{y}_{t-i}$ ;
- ③  $\mathbf{s}_t := \mathbf{s}_t / (\rho_{t-1} \langle \mathbf{y}_{t-1}, \mathbf{y}_{t-1} \rangle)$ 
  - ①  $b = \rho_{t-1} \langle \mathbf{y}_{t-1}, \mathbf{s}_t \rangle$ ;
  - ②  $\mathbf{s}_t := \mathbf{s}_t + (a_i - b) \mathbf{s}_{t-1}$ ;

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{s}_t;$$

$$\mathbf{y}_t = \mathbf{g}_{t+1} - \mathbf{g}_t; \rho_t = 1 / \langle \mathbf{s}_t, \mathbf{y}_t \rangle$$

maintain ring buffer of last  $m$  values of  $\mathbf{s}_t, \mathbf{y}_t$  vectors; scalar  $\rho_t$

### Can Use Kernel Trick

**Note:** only inner products and linear combinations

### New

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

# Online SVM (*aka* NORMA)

- Stochastic gradient (Kivinen *et al.*, IEEE TSP 2004)
  - ▶ objective:  $J(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, f) + \frac{c}{2} \|f\|_{\mathcal{H}}^2$ ,  $f \in \mathcal{H}$
  - ▶ stochastic gradient:  $g_t = \partial_f l(\mathbf{x}_t, z_t, f_t) + c f_t$
  - ▶ kernel expansion:  $f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((\mathbf{x}_i, z), \cdot)$ ,
  - ▶ coefficient update:

$$f_{t+1} = f_t - \eta_t g_t \quad \boldsymbol{\alpha}_t = \begin{bmatrix} (1 - \eta_t c) \boldsymbol{\alpha}_{t-1} \\ -\eta_t \boldsymbol{\xi}_t^\top \end{bmatrix}$$

- 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:  $J(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, f) + \frac{c}{2} \|f\|_{\mathcal{H}}^2$ ,  $f \in \mathcal{H}$
  - ▶ stochastic gradient:  $g_t = \partial_f l(\mathbf{x}_t, z_t, f_t) + cf_t$
  - ▶ kernel expansion:  $f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((\mathbf{x}_i, z), \cdot)$ ,
  - ▶ coefficient update:

$$f_{t+1} = f_t - \eta_t g_t \quad \boldsymbol{\alpha}_t = \begin{bmatrix} (1 - \eta_t c) \boldsymbol{\alpha}_{t-1} \\ -\eta_t \boldsymbol{\xi}_t^\top \end{bmatrix}$$

- 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:  $J(f) = \frac{1}{|T|} \sum_{i=1}^{|T|} l(\mathbf{x}_i, z_i, f) + \frac{c}{2} \|f\|_{\mathcal{H}}^2$ ,  $f \in \mathcal{H}$
  - ▶ stochastic gradient:  $g_t = \partial_f l(\mathbf{x}_t, z_t, f_t) + cf_t$
  - ▶ kernel expansion:  $f_t(\cdot) = \sum_{i=1}^{t-1} \sum_z \alpha_{tiz} k((\mathbf{x}_i, z), \cdot)$ ,
  - ▶ coefficient update:

$$f_{t+1} = f_t - \eta_t g_t \quad \boldsymbol{\alpha}_t = \begin{bmatrix} (1 - \eta_t c) \boldsymbol{\alpha}_{t-1} \\ -\eta_t \boldsymbol{\xi}_t^\top \end{bmatrix}$$

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

## Our Approach

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

# Online Kernel LBFGS (okLBFGS)

## okLBFGS DIRECTION UPDATE

...

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

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

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

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

## 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  $t := 0, 1, \dots$ :

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

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

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

## 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  $t := 0, 1, \dots$ :

- ①  $s_t := -\eta_t g_t$ ;
- ② for  $i := 1, 2, \dots, \min(t, m)$  :
  - 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

# Online Kernel LBFGS (okLBFGS)

## okLBFGS DIRECTION UPDATE

...

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

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

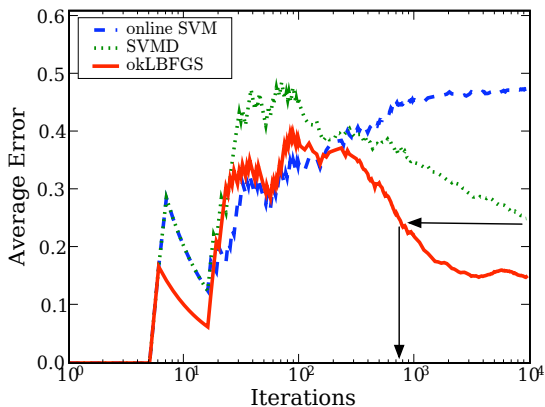
$$f_{t+1} = f_t + s_t;$$

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

## 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

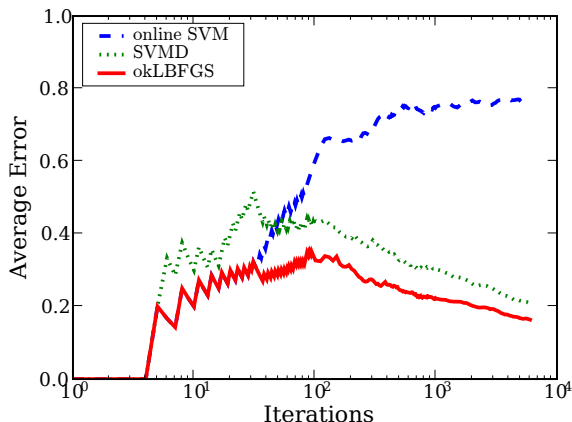
## USPS Binary



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

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

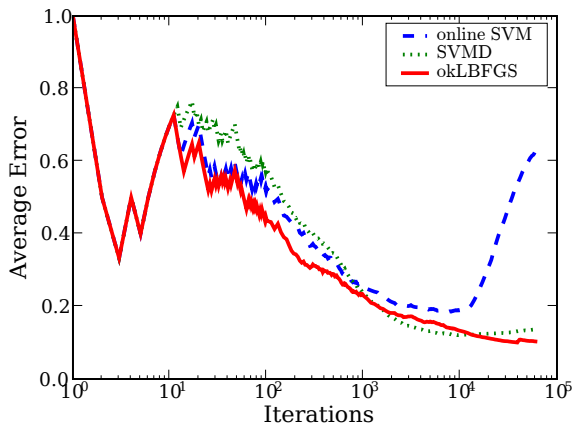
# USPS Counting Sequence



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

Figure: USPS 10-way Multiclass.

# 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