ECS289: Scalable Machine Learning

Cho-Jui Hsieh
UC Davis

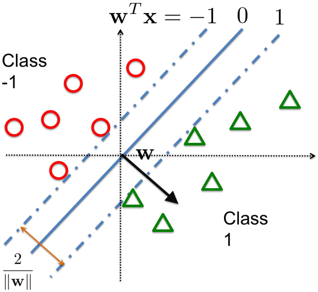
Oct 6, 2015

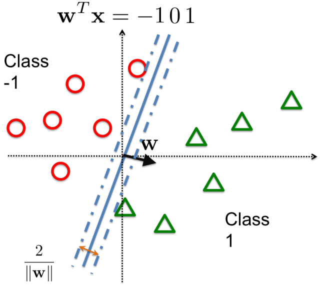
Outline

- Linear Support Vector Machines and Dual Problems
- General Empirical Risk Minimization
- Optimization for SVM (and general ERM problems)

Support Vector Machines

- SVM is a widely used classifier.
- Given:
  - Training data points $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$.
  - Each $\boldsymbol{x}_i \in \mathbb{R}^d$ is a feature vector:
  - Consider a simple case with two classes: $y_i \in \{+1, -1\}$.
- Goal: Find a hyperplane to separate these two classes of data: if $y_i = 1$, $\boldsymbol{w}^T \boldsymbol{x}_i \geq 1$; if $y_i = -1$, $\boldsymbol{w}^T \boldsymbol{x}_i \leq -1$.



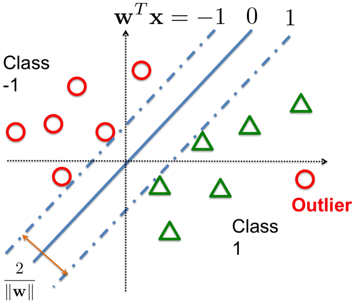


Support Vector Machines (hard constraints)

- Given training data $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$.
- SVM primal problem (with hard constraints):

$$\min_{\boldsymbol{w},\xi} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1, i = 1, \ldots, n,$$

- What if there are outliers?



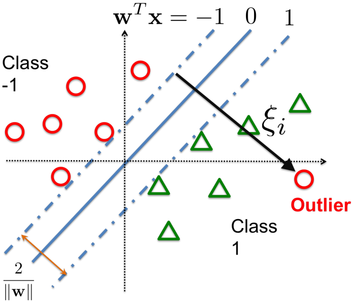
Support Vector Machines

- Given training data $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$.
- SVM primal problem:

$$\min_{\boldsymbol{w},\xi} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 - \xi_i, i = 1, \ldots, n,$$

$$\xi_i \geq 0$$



Support Vector Machines

- SVM primal problem can be written as

$$\min_{\boldsymbol{w}} \quad \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{L2 regularization}} \quad + \sum_{i=1}^{n} \underbrace{\max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)}_{\text{hinge loss}}$$

- Non-differentiable when $y_i \boldsymbol{w}^T \boldsymbol{x}_i = 1$ for some i
- Next, we show how to derive the dual form of SVM

Support Vector Machines (dual)

- Primal problem:

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

$$\text{s.t. } y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + \xi_i \geq 0, \text{ and } \xi_i \geq 0 \quad \forall i = 1, \ldots, n$$

- Equivalent to:

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i (y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + \xi_i) - \sum_i \beta_i \xi_i$$

- Under certain condition (e.g., slater's condition), exchanging min, max will not change the optimal solution:

$$\max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0} \min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i (y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + \xi_i) - \sum_i \beta_i \xi_i$$

Support Vector Machines (dual)

- Reorganize the equation:

$$\max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_i \alpha_i y_i \boldsymbol{w}^T \boldsymbol{x}_i + \sum_i \xi_i (C - \alpha_i - \beta_i) + \sum_i \alpha_i$$

- Now, for any given $\boldsymbol{\alpha}, \boldsymbol{\beta}$, the minimizer of $\boldsymbol{w}$ will satisfy

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i = 0 \quad \Rightarrow \boldsymbol{w}^* = \sum_i y_i \alpha_i \boldsymbol{x}_i$$

  Also, we have $C = \alpha_i + \beta_i$, otherwise ξ_i can make the objective function $-\infty$

- Substitue these two equations back we get

$$\max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0, C=\boldsymbol{\alpha}+\boldsymbol{\beta}} -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \alpha_i$$

Support Vector Machines (dual)

- Therefore, we get the following dual problem

$$\max_{C \geq \boldsymbol{\alpha} \geq 0} \{-\frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + \boldsymbol{e}^T \boldsymbol{\alpha}\} := D(\boldsymbol{\alpha}),$$

  where Q is an n by n matrix with $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Based on the derivations, we know
  1. Primal minimum = dual maximum (under slater's condition)
  2. Let $\boldsymbol{\alpha}^*$ be the dual solution and $\boldsymbol{w}^*$ be the primal solution, we have

$$\boldsymbol{w}^* = \sum_i y_i \alpha_i^* \boldsymbol{x}_i$$

- We can solve the dual problem instead of the primal problem.

General Empirical Risk Minimization

- L2-regularized ERM:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} P(\boldsymbol{w}) := \sum_{i=1}^{n} \ell_i(\boldsymbol{w}^T \boldsymbol{x}_i) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

  - $\ell_i(\cdot)$: loss function
- Dual problem for L2-regularized ERM:

$$\min_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) := \frac{1}{2}\|\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\|^2 + \sum_{i=1}^{n} \ell_i^*(-\alpha_i),$$

  - $\ell_i^*(\cdot)$: conjugate of ℓ
  - Primal-dual relationship: $\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$

General Empirical Risk Minimization

- Regularized ERM:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} P(\boldsymbol{w}) := \sum_{i=1}^{n} \ell_i(\boldsymbol{w}^T \boldsymbol{x}_i) + R(\boldsymbol{w})$$

  - $\ell_i(\cdot)$: loss function
  - $R(\boldsymbol{w})$: regularization
- Dual problem may have a different form (?)

Examples

- Loss functions:
  - Regression: $\ell_i(\boldsymbol{x}_i) = (\boldsymbol{x}_i - y_i)^2$
  - SVM (hinge loss): $\ell_i(\boldsymbol{x}_i) = \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0)$
  - Square hinge loss: $\ell_i(\boldsymbol{x}_i) = \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0)^2$
  - Logistic regression: $\ell_i(\boldsymbol{x}_i) = \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i})$
- Regularizations:
  - L2-regularization: $\|\boldsymbol{w}\|_2^2$
  - L1-regularization: $\|\boldsymbol{w}\|_1$
  - Group Lasso: $\|\boldsymbol{w}_{S_1}\|_2 + \|\boldsymbol{w}_{S_2}\|_2 + \cdots + \|\boldsymbol{w}_{S_k}\|_2$
  - Nuclear norm: $\|W\|_*$

Optimization Methods for ERM

	smooth loss and regularzation	smooth loss nonsmooth regularization
gradient descent	Yes	
proximal gradient	Yes	Yes
SGD	Yes	(Yes, with modification)
CD	Yes	Yes
Newton	Yes	
prox Newton	Yes	Yes

(assume non-smooth regularization is "simple")

Non-smooth loss: generally hard, need to use subgradient

Stochastic Gradient for SVM

Stochastic Gradient

- Decompose the problem into n parts:

$$
\begin{aligned}
f(\boldsymbol{w}) &= \frac{1}{n}\sum_i \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + nC\max(0, 1 - y_i \boldsymbol{w}^T\boldsymbol{x}_i)\right) \\
&:= \frac{1}{n}\sum_i f_i(\boldsymbol{w})
\end{aligned}
$$

- Stochastic Gradient (SG):

For $t = 1, 2, \ldots$

Randomly pick an index i

$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta_t \nabla f_i(\boldsymbol{w}^t)$

- Can be directly applied for smooth loss functions.
- But for SVM, f_i is non-differentiable.

Stochastic Subgradient Method

- A vector $\boldsymbol{g}$ is a subgradient of f at a point $\boldsymbol{x}_0$ if

$$f(\boldsymbol{x}) - f(\boldsymbol{x}_0) \geq \boldsymbol{g}^T(\boldsymbol{x} - \boldsymbol{x}_0) \quad \forall \boldsymbol{x}$$

- Stochastic Subgradient descent:

For $t = 1, 2, \ldots$
    Randomly pick an index i
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta_t g_i$, where g_i is a subgradient of f_i at $\boldsymbol{w}^t$

Stochastic Subgradient Method for SVM

- A subgradient of $\ell_i(\boldsymbol{w}) = \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)$:

$$\begin{cases} -y_i \boldsymbol{x}_i & \text{if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0 \\ \mathbf{0} & \text{if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i < 0 \\ \mathbf{0} & \text{if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i = 0 \end{cases}$$

- Stochastic Subgradient descent for SVM:

For $t = 1, 2, \ldots$
  Randomly pick an index i
  If $y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1$, then
    $\boldsymbol{w} \leftarrow (1 - \eta_t)\boldsymbol{w} + \eta_t n C y_i \boldsymbol{x}_i$
  Else (if $y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1$):
    $\boldsymbol{w} \leftarrow (1 - \eta_t)\boldsymbol{w}$

Stochastic Subgradient Method

- Improve the time complexity when $\boldsymbol{x}_i$ is sparse:
  - $\boldsymbol{w} \leftarrow (1-\eta_t)\boldsymbol{w}$: $O(d)$ time complexity
  - Instead, we maintain $\boldsymbol{w} = a\boldsymbol{v}$, so this operation requires only $O(1)$ time
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta n C y_i \boldsymbol{x}_i$: $O(n_i)$ time where n_i is the number of nonzeroes in $\boldsymbol{x}_i$
- This algorithm was proposed in:

  Shalev-Shwartz et al., "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM", in ICML 2007

Dual Coordinate Descent for SVM

Dual Form of SVM

- Given training data $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$.
- SVM dual problem:

$$\min_{0 \leq \boldsymbol{\alpha} \leq C} \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \sum_{i=1}^{n} \alpha_i$$

  where Q is an n by n matrix with $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$.
- After computing the dual optimal solution $\boldsymbol{\alpha}^*$, the primal solution is

$$\boldsymbol{w}^* = \sum_{i=1}^{n} y_i \alpha_i^* \boldsymbol{x}_i$$

Coordinate Descent Algorithm

- Stochastic Coordinate descent for solving $\min_{0 \leq \boldsymbol{\alpha} \leq C} f(\boldsymbol{\alpha})$:

For $t = 1, 2, \dots$

Randomly pick an index i

Compute the optimal one-variable update:

$$\delta^* = \underset{\delta: 0 \leq \alpha_i + \delta \leq C}{\operatorname{argmin}} f(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i)$$

Update $\alpha_i \leftarrow \alpha_i + \delta^*$

Dual Coordinate Descent for SVM

- One-variable subproblem:

$$\begin{aligned} f(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i) &= \frac{1}{2}(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i)^T Q(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i) - \sum_{i=1}^n \alpha_i - \delta \\ &= \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + \boldsymbol{\alpha}^T Q \boldsymbol{e}_i \delta + \frac{Q_{ii}}{2}\delta^2 - \sum_{i=1}^n \alpha_i - \delta \end{aligned}$$

- Compute $\operatorname{argmin}_\delta f(\boldsymbol{\alpha} + \delta \boldsymbol{e}_i)$: set gradient equals to zero

$$(Q\boldsymbol{\alpha})_i + Q_{ii}\delta^* - 1 = 0 \quad \Rightarrow \quad \delta^* = \frac{1 - (Q\boldsymbol{\alpha})_i}{Q_{ii}}$$

- However, we require $0 \leq \alpha_i + \delta \leq C$, so the optimal solution is

$$\delta^* = \max\Big(-\alpha_i, \min\big(C - \alpha_i, \frac{1 - (Q\boldsymbol{\alpha})_i}{Q_{ii}}\big)\Big)$$

Dual Coordinate Descent for SVM

- Main computation: the i-th element of $Q\alpha$
- Time complexity:

Dual Coordinate Descent for SVM

- Main computation: the i-th element of $Q\boldsymbol{\alpha}$
- Time complexity:

$$
\begin{aligned}
(Q\boldsymbol{\alpha})_i &= \sum_{j=1}^{n} Q_{ij}\alpha_j \\
&= \sum_{j=1}^{n} y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \alpha_j \\
&= y_i \sum_{j=1}^{n} y_j \alpha_j \boldsymbol{x}_j^T \boldsymbol{x}_i
\end{aligned}
$$

- Naive implementation: $O(nd)$ time.

Speedup the Computation

- A faster way to compute coordinate descent updates.

$$(Q\boldsymbol{\alpha})_i = y_i\Big(\sum_{j=1}^{n} y_j \alpha_j \boldsymbol{x}_j\Big)^T \boldsymbol{x}_i$$

- Maintain $\boldsymbol{w} = \sum_{j=1}^{n} y_j \alpha_j \boldsymbol{x}_j$ in the memory:
  $\Rightarrow O(d)$ time for computing $(Q\boldsymbol{\alpha})_i$
- $\boldsymbol{w}$ is the primal variables correspond to current dual solution $\boldsymbol{\alpha}$!

Dual Coordinate Descent for SVM

- After updating $\alpha_i \leftarrow \alpha_i + \delta^*$, we need to maintain $\boldsymbol{w}$:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta^* y_i \boldsymbol{x}_i$$

  Time complexity: $O(d)$
- After convergence,

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i$$

  is the optimal primal solution.

Dual Coordinate Descent for SVM

Initial: $\boldsymbol{\alpha}$, $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$

For $t = 1, 2, \ldots$

Randomly pick an index i

Compute the optimal one-variable update:

$$\delta^* = \max\left(-\alpha_i, \min\left(C - \alpha_i, \frac{1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i}{Q_{ii}}\right)\right)$$

Update $\alpha_i \leftarrow \alpha_i + \delta^*$

Update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta^* y_i \boldsymbol{x}_i$

(Hsieh et al., "A Dual Coordinate Descent Method for Large-scale Linear SVM", ICML 2008)

Can be applied to general L2-regularized ERM problems (Shalev-Shwartz and Zhang, "Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization" JMLR 2013)

Convergence of Dual Coordinate Descent

- Is the dual SVM problem strongly convex?

Convergence of Dual Coordinate Descent

- Is the dual SVM problem strongly convex?

$$Q = \bar{X}\bar{X}^T \quad \text{may be low-rank}$$

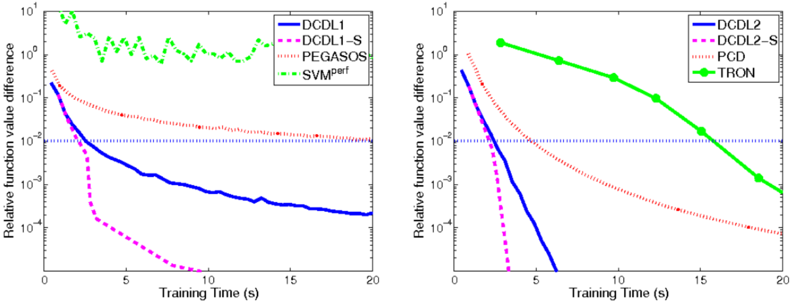
- Sublinear convergence in dual objective function value.
- Sublinear convergence in duality gap (primal obj - dual obj)

  (Shown in Shalev-Shwartz and Zhang, "Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization". JMLR 2013).
- Global linear convergence rate in terms of dual objective function value.

  (Shown in Wang and Lin, "Iteration Complexity of Feasible Descent Methods for Convex Optimization". JMLR 2014)

Experimental comparison

- RCV1: 677,399 training samples; 47,236 features; 49,556,258 nonzeroes in the whole dataset.

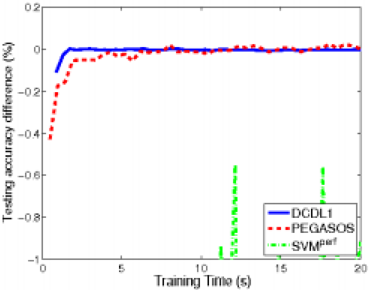


(e) L1-SVM: rcv1 (f) L2-SVM: rcv1

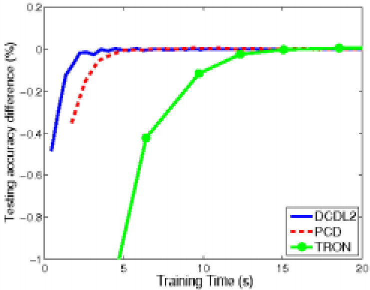
Time vs primal objective function value

Experimental comparison

- RCV1: 677,399 training samples; 47,236 features; 49,556,258 nonzeroes in the whole dataset.



(e) L1-SVM: rcv1



(f) L2-SVM: rcv1

Time vs prediction accuracy

LIBLINEAR

- Implemented in LIBLINEAR:

  https://www.csie.ntu.edu.tw/~cjlin/liblinear/

- Other functionalities:
  - Logistic regression (L1 or L2 regularization)
  - Multi-class SVM
  - Support vector regression
  - Cross-validation

Recent Research Topics

Linear SVM on a multi-core machine

- Asynchronous dual coordinate descent algorithm:

Each thread repeatedly performs the following updates:

For $t = 1, 2, \ldots$

Randomly pick an index i

Compute the optimal one-variable update:

$$\delta_i^* = \max\left(-\alpha_i, \min\left(C - \alpha_i, \frac{1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i}{Q_{ii}}\right)\right)$$

Update $\alpha_i \leftarrow \alpha_i + \delta_i^*$

Update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta_i^* y_i \boldsymbol{x}_i$

- Different mechanisms on accessing $\boldsymbol{w}$ in the shared memory.

Hsieh et al., "PASSCoDe: Parallel ASynchronous Stochastic dual Co-ordinate Descent", in ICML 2015

Huan and Hsieh, " Fixing the Convergence Problems in Parallel Asynchronous Dual Coordinate Descent", in ICDM 2016

Other multi-core algorithms for SVM

- Asynchronous stochastic (sub-)gradient decent for the primal problem:

  Niu et al., "Hogwild! A Lock-Free Approach to Parallelizing Stochastic Gradient Descent", in NIPS 2011.

- Another approach (parallelized variable selection)

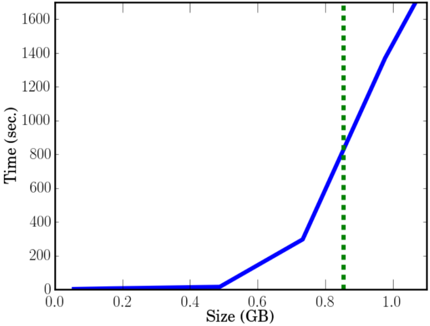
  Chiang et al., "Parallel Dual Coordinate Descent Method for Large-scale Linear Classification in Multi-core Environment", in KDD 2016.

- Other parallel primal solvers:

  Lee et al., "Fast Matrix-vector Multiplications for Large-scale Logistic Regression on Shared-memory Systems", in ICDM 2015.

Large-scale linear ERM: out-of-core version

- Question: how to solve linear SVM on a single machine when data cannot fit in memory?
- Dual coordinate descent: need random access, not suitable when training samples are stored in disk



Large-scale linear: out-of-core version

- Block coordinate descent:
  - Partition data into blocks $S_1, \ldots, S_k$
  - For $t = 1, 2, \ldots$
    - Load a block S_i into memory
    - Update this block of dual variables by dual coordinate descent

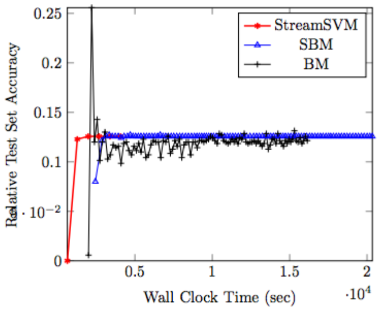
  (Yu et al., "Large Linear Classification when Data Cannot Fit In Memory", in KDD 2010)
- Selective block coordinate descent: keep important samples in memory

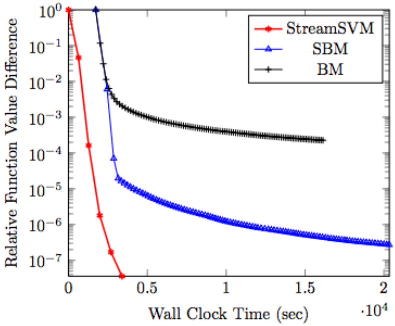
  (Chang and Roth, "Selective Block Minimization for Faster Convergence of Limited Memory Large-scale Linear Models", in KDD 2011)
- StreamSVM: one thread keep loading data, while another thread keep running coordinate descent updates

  (Matsushima et al., "Linear Support Vector Machines via Dual Cached Loops", in KDD 2012)

Comparisons

On webspam dataset, 0.35 million samples, 16.61 million features, dataset size 20.03GB





Distributed Linear ERM

- Each machine stores a subset of training samples
- Each machine conducts dual coordinate updates and has a local $\boldsymbol{w}$
- How to communicate and synchronize the updates?
- Can the methods generalize to other local solvers?
- What if there are millions of machines (nodes), and data is non-iid distributed?

Distributed Linear ERM

- COCOA: local dual coordinate descent updates, and averaging.
  (Jaggi et al., "Communication-Efficient Distributed Dual Coordinate Ascent", in NIPS, 2014.)
- Block Quadratic Optimization (COCOA with improved step size selection).
  (Lee et al., "Distributed Box-Constrained Quadratic Optimization for Dual Linear Support Vector Machines", in ICML, 2015.)
- COCOA+: improved COCOA by solving a modified subproblem.
  (Ma et al., "Adding vs. Averaging in Distributed Primal-Dual Optimization", in ICML, 2015.)
- A more general framework (for other regularizations).
  (Zheng et al., "A General Distributed Dual Coordinate Optimization Framework for Regularized Loss Minimization", arXiv, 2016.)

Other distributed algorithms

- Primal solver:
  (Zhuang et al., "Distributed Newton Method for Regularized Logistic Regression", in PAKDD 2015.)
- DANE: distributed Newton-type method
  (Shamir et al., "Communication-Efficient Distributed Optimization using an Approximate Newton-type Method", in ICML 2014.)
- DisCO: based on inexact Newton method:
  (Zhang et al., "Communication-Efficient Distributed Optimization of Self-Concordant Empirical Loss", in ICML 2015.)

Coming up

Questions?