Alternating Direction Method of Multipliers for Distributed Machine Learning

Sourangshu Bhattacharya

Dept. of Computer Science and Engineering,
IIT Kharagpur.
`http://cse.iitkgp.ac.in/~sourangshu/`

Outline

Distributed gradient descent

- Define $loss(\mathbf{x}) = \sum_{j=1}^{m} \sum_{i \in C_j} l_i(\mathbf{x}) + \lambda\Omega(\mathbf{x})$, where $l_i(\mathbf{x}) = l(\mathbf{x}, \mathbf{u}_i, v_i)$
- The gradient (in case of differentiable loss):

$$\nabla loss(\mathbf{x}) = \sum_{j=1}^{m} \nabla(\sum_{i \in C_j} l_i(\mathbf{x})) + \lambda\Omega(\mathbf{x})$$

- Compute $\nabla l_j(\mathbf{x}) = \sum_{i \in C_j} \nabla l_i(\mathbf{x})$ on the j^{th} computer. Communicate to central computer.

Distributed gradient descent

- Compute $\nabla loss(\mathbf{x}) = \sum_{j=1}^{m} \nabla l_j(\mathbf{x}) + \Omega(\mathbf{x})$ at the central computer.
- The gradient descent update: $x^{k+1} = x^k - \alpha \nabla loss(\mathbf{x})$.
- α chosen by a line search algorithm (distributed).
- For non-differentiable loss functions, we can use distributed sub-gradient descent algorithm.
  - Slow for most practical problems.

Outline

Dual Ascent

- Convex equality constrained problem:

$$\min_x \; f(x)$$
$$\text{subject to: } Ax = b$$

- Lagrangian: $L(x,y) = f(x) + y^T(Ax - b)$
- Dual function: $g(y) = \inf_x L(x,y)$
- Dual problem: $\max_y g(y)$
- Final solution: $x^* = \text{argmin}_x L(x,y)$

Dual Ascent

- Gradient descent for dual problem: $y^{k+1} = y^k + \alpha^k \nabla_{y^k} g(y^k)$
- $\nabla_{y^k} g(y^k) = A\tilde{x} - b$, where $\tilde{x} = \text{argmin}_x L(x, y^k)$
- Dual ascent algorithm:

$$x^{k+1} = \text{argmin}_x L(x, y^k)$$
$$y^{k+1} = y^k + \alpha^k (Ax^{k+1} - b)$$

- Assumptions:
    - $L(x, y^k)$ is strictly convex. Else, the first step can have multiple solutions.
    - $L(x, y^k)$ is bounded below.

Dual Decomposition

- Suppose f is separable:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \ x = (x_1, \ldots, x_N)$$

- L is separable in x: $L(x, y) = L_1(x_1, y) + \cdots + L_N(x_N, y) - y^T b$, where $L_i(x_i, y) = f_i(x_i) + y^T A_i x_i$
- x minimization splits into N separate problems:

$$x_i^{k+1} = \text{argmin}_{x_i} L_i(x_i, y^k)$$

Dual Decomposition

- Dual decomposition:

$$x_i^{k+1} = \text{argmin}_{x_i} L_i(x_i, y^k),\ i = 1, \dots, N$$
$$y^{k+1} = y^k + \alpha^k (\sum_{i=1}^{N} A_i x_i - b)$$

- Distributed solution:
  - Scatter y^k to individual nodes
  - Compute x_i in the i^{th} node (distributed step)
  - Gather $A_i x_i$ from the i^{th} node
- All drawbacks of dual ascent exist

Method of Multipliers

- Make dual ascent work under more general conditions
- Use **augmented Lagrangian**:

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2$$

- Method of multipliers:

$$\begin{aligned} x^{k+1} &= \text{argmin}_x L_\rho(x, y^k) \\ y^{k+1} &= y^k + \rho(Ax^{k+1} - b) \end{aligned}$$

Methods of Multipliers

- Optimality conditions (for differentiable f):
  - Primal feasibility: $Ax^* - b = 0$
  - Dual feasibility: $\nabla f(x^*) + A^T y^* = 0$
- Since x^{k+1} minimizes $L_\rho(x, y^k)$

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T(y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

- Dual update $y^{k+1} = y^k + \rho(Ax^{k+1} - b)$ makes (x^{k+1}, y^{k+1}) dual feasible
- Primal feasibility is achieved in the limit: $(Ax^{k+1} - b) \to 0$

Outline

Alternating direction method of multipliers

- Problem with applying standard method of multipliers for distributed optimization:
  - there is no problem decomposition even if f is separable.
  - due to square term $\frac{\rho}{2}\|Ax + Bz - c\|_2^2$

Alternating direction method of multipliers

- ADMM problem:

$$\min_{x,z} \; f(x) + g(z)$$
$$\text{subject to: } Ax + Bz = c$$

- Lagrangian:
  $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$
- ADMM:

$$\begin{aligned} x^{k+1} &= \text{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &= \text{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Alternating direction method of multipliers

- Problem with applying standard method of multipliers for distributed optimization:
  - there is no problem decomposition even if f is separable.
  - due to square term $\frac{\rho}{2}\|Ax + Bz - c\|_2^2$
- The above technique reduces to method of multipliers if we do joint minimization of x and z
- Since we split the joint x, z minimization step, the problem can be decomposed.

ADMM Optimality conditions

- Optimality conditions (differentiable case):
  - Primal feasibility: $Ax + Bz - c = 0$
  - Dual feasibility: $\nabla f(x) + A^T y = 0$ and $\nabla g(z) + B^T y = 0$
- Since z^{k+1} minimizes $L_\rho(x^{k+1}, z, y^k)$:

$$\begin{aligned} 0 &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T y^{k+1} \end{aligned}$$

- So, the dual variable update satisfies the second dual feasibility constraint.
- Primal feasibility and first dual feasibility are satisfied asymptotically.

ADMM Optimality conditions

- Primal residual: $r^k = Ax^k + Bz^k - c$
- Since x^{k+1} minimizes $L_\rho(x, z^k, y^k)$:

$$\begin{aligned} 0 &= \nabla f(x^{k+1}) + A^T y^k + \rho A^T(Ax^{k+1} + Bz^k - c) \\ &= \nabla f(x^{k+1}) + A^T(y^k + \rho r^{k+1} + \rho B(z^k - z^{k+1}) \\ &= \nabla f(x^{k+1}) + A^T y^{k+1} + \rho A^T B(z^k - z^{k+1}) \end{aligned}$$

or,

$$\rho A^T B(z^k - z^{k+1}) = \nabla f(x^{k+1}) + A^T y^{k+1}$$

- Hence, $s^{k+1} = \rho A^T B(z^k - z^{k+1})$ can be thought as dual residual.

Step size selection

- Combine the linear and quadratic terms
  - Primal feasibility: $Ax + Bz - c = 0$
  - Dual feasibility: $\nabla f(x) + A^T y = 0$ and $\nabla g(z) + B^T y = 0$
- Since z^{k+1} minimizes $L_\rho(x^{k+1}, z, y^k)$:

$$\begin{aligned} 0 &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T y^{k+1} \end{aligned}$$

- So, the dual variable update satisfies the second dual feasibility constraint.
- Primal feasibility and first dual feasibility are satisfied asymptotically.

ADMM with scaled dual variables

- Let $r = Ax + Bz - c$
- Lagrangian: $L_\rho(x, z, y) = f(x) + g(z) + y^T r + \frac{\rho}{2}\|r\|_2^2$

$$\begin{aligned} y^T r + \frac{\rho}{2}\|r\|_2^2 &= \frac{\rho}{2}\|r + \frac{1}{\rho}y\|_2^2 - \frac{1}{2\rho}\|y\|_2^2 \\ &= \frac{\rho}{2}\|r + u\|_2^2 - \frac{\rho}{2}\|u\|_2^2 \end{aligned}$$

- where $u = \frac{1}{\rho}y$ are scaled dual variables.
- ADMM updates:

$$\begin{aligned} x^{k+1} &= \text{argmin}_x f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|^2 \\ z^{k+1} &= \text{argmin}_z g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|^2 \\ u^{k+1} &= u^k + (Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Outline

Convergence of ADMM

- Assumption 1: Functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^m \rightarrow \mathbb{R}$ are closed, proper and convex.
  - Same as assuming $\text{epi} f = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} | f(x) \leq t\}$ is closed and convex.
- Assumption 2: The unaugmented Lagrangian $L_0(x, y, z)$ has a saddle point $(x*, z*, y*)$:

$$L_0(x*, z*, y) \leq L_0(x*, z*, y*) \leq L_0(x, z, y*)$$

Convergence of ADMM

- Primal residual: $r = Ax + Bz - c$
- Optimal objective: $p^* = \inf_{x,z}\{f(x) + g(z) | Ax + Bz = c\}$
- Convergence results:
  - Primal residual convergence: $r^k \to 0$ as $k \to \infty$
  - Dual residual convergence: $s^k \to 0$ as $k \to \infty$
  - Objective convergence: $f(x) + g(z) \to p^*$ as $k \to \infty$
  - Dual variable convergence: $y^k \to y^*$ as $k \to \infty$

Stopping criteria

- Stop when primal and dual residuals small:

$$\|r^k\|_2 \leq \epsilon^{pri} \text{ and } \|s^k\|_2 \leq \epsilon^{dual}$$

Hence, $\|r^k\|_2 \rightarrow 0$ and $\|s^k\|_2 \rightarrow 0$ as $k \rightarrow \infty$

Observations

- x- update requires solving an optimization problem

$$\min_x f(x) + \frac{\rho}{2}\|Ax - v\|_2^2$$

with, $v = Bz^k - c + u^k$

- Similarly for z-update.
- Sometimes has a closed form.
- ADMM is a meta optimization algorithm.

Proximal Operator

- x-update when A=I

$$x^+ = \text{argmin}_x(f(x) + \frac{\rho}{2}\|x - v\|_2^2 = prox_{f,\rho}(v)$$

- Some special cases:

$$f = I_C \text{ (Indicator fn of C) }, x^+ = \Pi_C(v) \text{ (projecction on to C)}$$
$$f = \lambda\|.\|_1, x^+ = S_{\frac{\lambda}{\rho}}(v)$$

where, $S_a(v) = (v - a)_+ - (-v - a)_+$.

Decomposition of optimization

- Convex equality constrained problem:

$$\min_x \; f(x)$$
$$\text{subject to: } Ax = b$$

- If f is separable:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \; x = (x_1, \ldots, x_N)$$

- A is conformably block separable; i.e. $A^T A$ is block diagonal.
- Then, x-update splits into N parallel updates of x_i

Outline

Consensus Optimization

- Problem:

$$\min_x f(x) = \sum_{i=1}^{N} f_i(x)$$

- ADMM form:

$$\min_{x_i, z} \sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_i - z = 0, \; i = 1, \dots, N$$

- Augmented lagrangian:

$$L_\rho(x_1, \dots, x_N, z, y) = \sum_{i=1}^{N} (f_i(x_i) + y_i^T (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2)$$

Consensus Optimization

- ADMM algorithm:

$$
\begin{aligned}
x_i^{k+1} &= \text{argmin}_{x_i}(f_i(x_i) + y_i^{kT}(x_i - z^k) + \frac{\rho}{2}\|x_i - z^k\|_2^2) \\
z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}y_i^k) \\
y_i^{k+1} &= y_i^k + \rho(x_i^{k+1} - z^{k+1})
\end{aligned}
$$

- Final solution is z^k.

Consensus Optimization

- z-update can be written as:

$$z^{k+1} = \bar{x}^{k+1} + \frac{1}{\rho}\bar{y}^{k+1}$$

- Averaging the y-updates:

$$\bar{y}^{k+1} = \bar{y}^{k} + \rho(\bar{x}^{k+1} - z^{k+1})$$

- Substituting first into second: $\bar{y}^{k+1} = 0$. Hence $z^k = \bar{x}^k$.
- Revised algorithm:

$$x_i^{k+1} = \text{argmin}_{x_i}(f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2}\|x_i - \bar{x}^k\|_2^2)$$
$$y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})$$

- Final solution is z^k.

Outline

Loss minimization

- Problem:

$$\min_{x} l(Ax - b) + r(x)$$

- Partition A and b by rows:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix},$$

  where, $A_i \in \mathbb{R}^{m_i \times m}$ and $b_i \in \mathbb{R}^{m_i}$
- ADMM formulation:

$$\min_{x_i, z} \sum_{i=1}^{N} l_i(A_i x_i - b_i) + r(z)$$

$$\text{s.t.: } x_i - z = 0, \; i = 1, \ldots, N$$

Loss minimization

- ADMM formulation:

$$\min_{x_i, z} \sum_{i=1}^{N} l_i(A_i x_i - b_i) + r(z) \ \text{ s.t.: } x_i - z = 0, \ i = 1, \ldots, N$$

- Augmented Lagrangian:

$$L_\rho(x_i, z, u_i) = \sum_{i=1}^{N} l_i(A_i x_i - b_i) + r(z) + \frac{\rho}{2} \sum_{i=1}^{N} (\|z - x_i + u_i\|_2^2 - \|u_i\|^2)$$

- ADMM solution:

$$x_i^{k+1} = \text{argmin}_{x_i}(l_i(A_i x_i - b_i) + \frac{\rho}{2}\|x_i - z^k + u_i^k\|_2^2)$$

$$z^{k+1} = \text{argmin}_z(r(z) + \frac{\rho}{2} \sum_{i=1}^{N} \|z - x_i^{k+1} + u_i^k\|_2^2)$$

$$u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1}$$

Outline

Support Vector Machines

- Training dataset: $S = \{(\mathbf{x}_i, y_i) : i = 1, \cdots, ML,$ $y_i \in \{-1, +1\}, \mathbf{x}_i \in \mathbf{R}^d\}$.
- Predictor function: $y_i = sign(\mathbf{w}^T \mathbf{x}_i)$
- Linear SVM problem:

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_2^2 + \frac{1}{m} \sum_{i=1}^{ML} loss(\mathbf{w}; (\mathbf{x}_i, y_i)),$$

- Hinge loss: $loss(\mathbf{w}; (\mathbf{x}_i, y_i)) = max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$

Distributed Support Vector Machines

- Training dataset partitioned into M partitions ($\mathcal{S}_m,\ m = 1, \ldots, M$).
- Each partition has L datapoints: $\mathcal{S}_m = \{(\mathbf{x}_{ml}, y_{ml})\}, l = 1, \ldots, L$.
- Each partition can be processed locally on a single computer.
- Distributed SVM training problem:

$$\min_{\mathbf{w}_m,\mathbf{z}} \sum_{m=1}^{M} \sum_{l=1}^{L} loss(\mathbf{w}_m; (\mathbf{x}_{ml}, y_{ml})) + r(\mathbf{z})$$
$$\text{s.t.} \mathbf{w}_m - \mathbf{z} = 0, m = 1, \cdots, M,\ l = 1, \ldots, L$$

Parameter Averaging

- Parameter averaging, also called "mixture weights" was proposed for logistic regression[1].
- Widely used as a federated learning technique.
- Locally learn SVM parameters on $\mathcal{S}_m$:

$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{L} \sum_{l=1}^{L} loss(\mathbf{w}; \mathbf{x}_{ml}, y_{ml}) + \lambda \|\mathbf{w}\|^2 \ , \ m = 1, \ldots, M$$

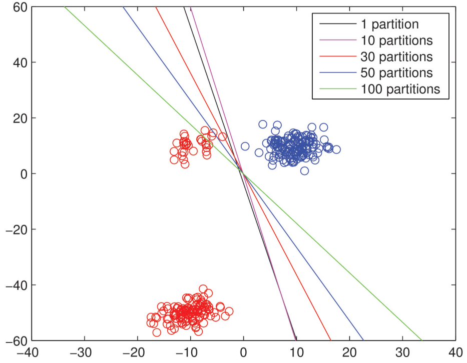
- The final SVM parameter is given by:

$$\mathbf{w}_{PA} = \frac{1}{M} \sum_{m=1}^{M} \hat{\mathbf{w}}_m$$

[1] McDonald, Ryan, Keith Hall, and Gideon Mann. "Distributed training strategies for the structured perceptron." NAACL HLT 2010.

Problem with Parameter Averaging

PA with varying number of partitions - Toy dataset.



Weighted Parameter Averaging

- Final hypothesis is a weighted sum of the parameters $\hat{\mathbf{w}}_m$.

$$\mathbf{w} = \sum_{m=1}^{M} \beta_m \mathbf{w_m}$$

- How to get β_m ?
- Notation: $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_M]^T$, $\mathbf{W} = [\hat{\mathbf{w}}_1, \cdots, \hat{\mathbf{w}}_M]$

$$\mathbf{w} = \mathbf{W}\beta$$

Weighted Parameter Averaging

- Find the optimal set of weights β which attains the lowest regularized hinge loss:

$$\min_{\boldsymbol{\beta},\boldsymbol{\xi}} \lambda\|\mathbf{W}\beta\|^2 + \frac{1}{ML}\sum_{m=1}^{M}\sum_{i=1}^{L}\xi_{mi}$$

$$\textbf{subject to: } y_{mi}(\beta^T\mathbf{W}^T\mathbf{x}_{mi}) \geq 1 - \xi_{mi}, \quad \forall i, m$$

$$\xi_{mi} \geq 0, \quad \forall m = 1, \ldots, M, \; i = 1, \ldots, L$$

- $\hat{\boldsymbol{W}}$ is a pre-computed parameter.

Dual Weighted Parameter Averaging

- Lagrangian:

$$\mathcal{L}(\beta, \xi_{mi}, \alpha_{mi}, \mu_{mi}) = \lambda\|\mathbf{W}\beta\|^2 + \frac{1}{ML}\sum_{m,i}\xi_{mi}$$
$$+ \sum_{m,i}\alpha_{mi}(y_{mi}(\beta^T W^T \mathbf{x}_{mi}) - 1 + \xi_{mi}) - \sum_{m,i}\mu_{mi}\xi_{mi}$$

- Differentiating w.r.t. β and equating to zero:

$$\beta = \frac{1}{2\lambda}(\mathbf{W}^T\mathbf{W})^{-1}(\sum_{m,i}\alpha_{mi}y_{mi}\mathbf{W}^T\mathbf{x}_{mi})$$

Dual Weighted Parameter Averaging

- Similarly, differentiating w.r.t. ξ_{mi} and equating to zero:

$$0 \leq \alpha_{mi} \leq \frac{1}{ML}$$

- Substituting β in $\mathcal{L}$:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{m,i} \alpha_{mi} - \frac{1}{4\lambda} \sum_{m,i} \sum_{m',j} \alpha_{mi}\alpha_{m'j} y_{mi} y_{m'j} (\mathbf{x}_{mi}^T \mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T \mathbf{x}_{m'j})$$

$$\textbf{subject to: } 0 \leq \alpha_{mi} \leq \frac{1}{ML} \quad \forall i \in 1, \cdots, L, m \in 1, \cdots, M$$

- SVM with $\mathbf{x}_{mi}$ projected using symmetric projection $\mathcal{H} = \mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$.

Distributed Weighted Parameter Averaging

- Distributed version of primal weighted parameter averaging:

$$\min_{\gamma_m,\beta} \frac{1}{ML} \sum_{m=1}^{M} \sum_{l=1}^{L} loss(\hat{W}\gamma_m; \mathbf{x}_{ml}, y_{ml}) + r(\beta)$$
$$\text{s.t.} \quad \gamma_m - \beta = 0, \quad m = 1, \cdots, M,$$

- $r(\beta) = \lambda \|\hat{\mathbf{W}}\beta\|^2$, γ_m weights for m^{th} computer, β consensus weight.

Distributed Weighted Parameter Averaging

- Distributed algorithm using ADMM:

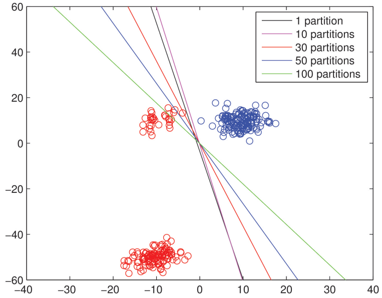
$$\begin{aligned}
\gamma_m^{k+1} &:= \underset{\gamma}{\operatorname{argmin}}\left(loss(\mathbf{A}_i\gamma) + (\rho/2)\|\gamma - \beta^k + \mathbf{u}_m^k\|_2^2\right)\\
\beta^{k+1} &:= \underset{\beta}{\operatorname{argmin}}\left(r(\beta) + (M\rho/2)\|\beta - \overline{\gamma}^{k+1} - \overline{\mathbf{u}}^k\|_2^2\right)\\
\mathbf{u}_m^{k+1} &= \mathbf{u}_m^k + \gamma_m^{k+1} - \beta^{k+1}.
\end{aligned}$$

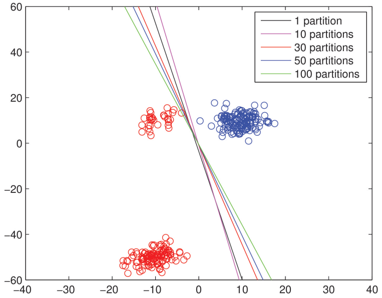
- $\mathbf{u}_m$ are the scaled Lagrange multipliers, $\overline{\gamma} = \frac{1}{M}\sum_{m=1}^{M}\gamma_m$ and $\overline{\mathbf{u}} = \frac{1}{M}\sum_{m=1}^{M}\mathbf{u}_m$.

Outline

Toy Dataset - PA and WPA

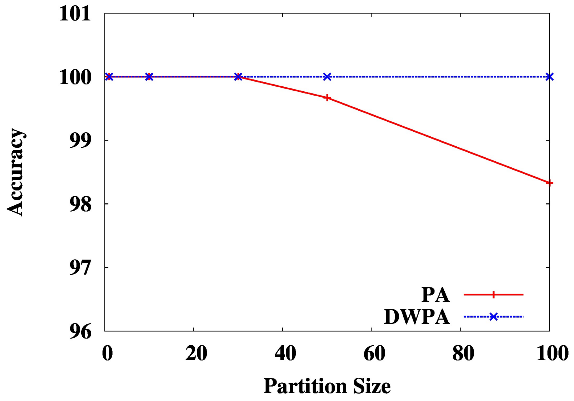
PA (left) and WPA (right) with varying number of partitions - Toy dataset.





Toy Dataset - PA and WPA

Accuracy of PA and WPA with varying number of partitions - Toy dataset.



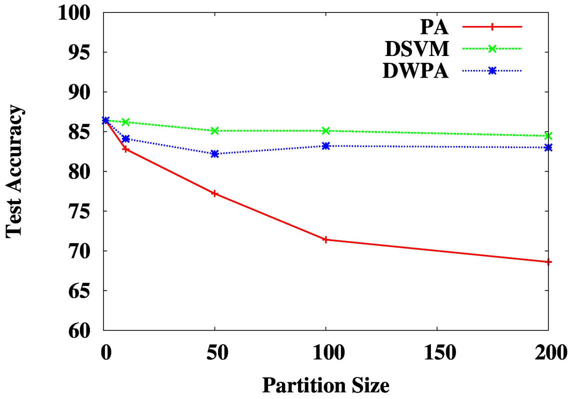
Toy Dataset - PA and WPA

Bias ($E[\|\boldsymbol{w} - \boldsymbol{w}^*\|]$) of PA, WPA and DSVM with varying number of partitions - Toy dataset.

Sample size	Mean bias(PA)	Mean bias(DWPA)	Mean bias(DSVM)
3000	**0.868332**	0.260716	0.307931
6000	**0.807217**	0.063649	0.168727

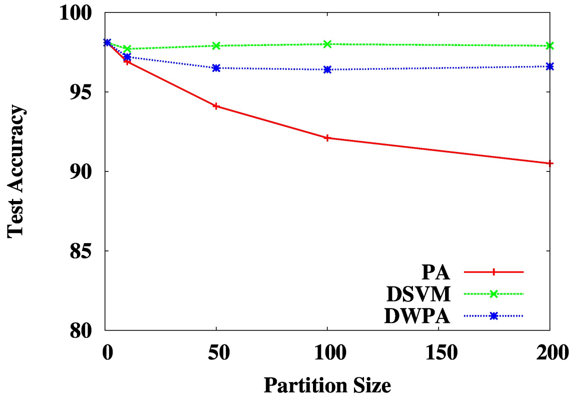
Real World Datasets

Epsilon (2000 features, 6000 datapoints) test set accuracy with varying number of partitions.



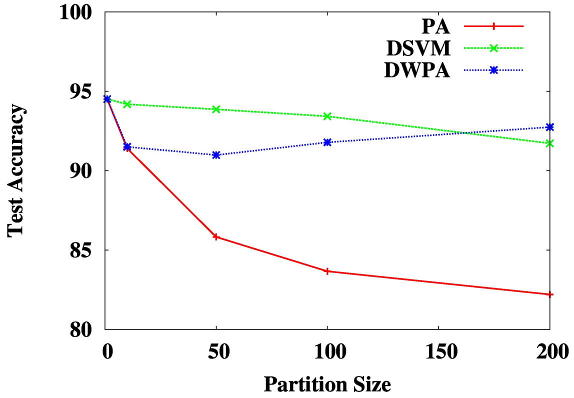
Real World Datasets

Gisette (5000 features, 6000 datapoints) test set accuracy with varying number of partitions.



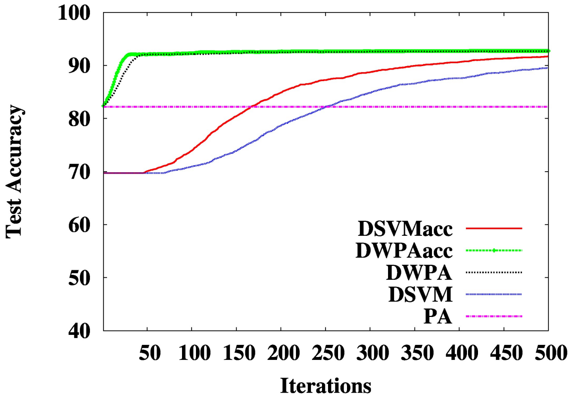
Real World Datasets

Real-sim (20000 features, 3000 datapoints) test set accuracy with varying number of partitions.



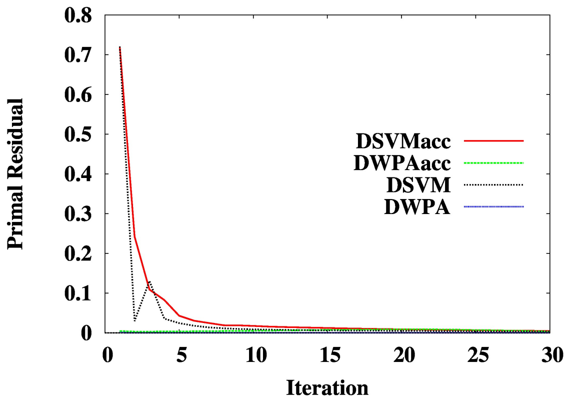
Real World Datasets

Convergence of test accuracy with iterations (200 partitions).



Real World Datasets

Convergence of primal residual with iterations (200 partitions).



Fully distributed SVM

- SVM optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{j=1}^{J}\sum_{n=1}^{n_j} \xi_{jn}$$
$$\text{s.t.: } y_{jn}(w^t x_{jn} + b) \geq 1 - \xi_{jn},\ \forall j \in J, n = 1, \dots, N_j$$
$$\xi_{jn} \geq 0,\ \forall j \in J, n = 1, \dots, N_j$$

- Node j has a copy of w_j, b_j. Distributed formulation:

$$\min_{\{w_j, b_j, \xi_{jn}\}} \frac{1}{2}\sum_{j=1}^{J} \|w_j\|^2 + JC\sum_{j=1}^{J}\sum_{n=1}^{n_j} \xi_{jn}$$
$$\text{s.t.: } y_{jn}(w_j^t x_{jn} + b) \geq 1 - \xi_{jn},\ \forall j \in J, n = 1, \dots, N_j$$
$$\xi_{jn} \geq 0,\ \forall j \in J, n = 1, \dots, N_j$$
$$w_j = w_i,\ \forall j, i \in \mathcal{B}_j$$

Outline

Fully distributed SVM

- Using $v_j = [w_j^T b_j]^T$, $X_j = [[x_{j1}, \ldots, x_{jN_j}]^T 1_j]$ and $Y_j = diag([y_{j1}, \ldots, y_{jN_j}])$:

$$\min_{\{v_j, \xi_{jn}, \omega_{ji}\}} \frac{1}{2} \sum_{j=1}^{J} r(v_j) + JC \sum_{j=1}^{J} \sum_{n=1}^{n_j} \xi_{jn}$$

$$\text{s.t.: } Y_j X_j v_j \geq 1 - \bar{\xi}_j, \ \forall j \in J$$

$$\bar{\xi}_j \geq 0, \ \forall j \in J$$

$$v_j = \omega_{ji}, \ v_i = \omega_{ji}, \ \forall j, i \in \mathcal{B}_j$$

- Surrogate augmented Lagrangian:

$$L(\{v_j\}, \{\bar{\xi}_j\}, \{\omega_{ji}\}, \{\alpha_{ijk}\}) = \frac{1}{2} \sum_{j=1}^{J} r(v_j) + JC \sum_{j=1}^{J} \sum_{n=1}^{n_j} \xi_{jn}$$

$$+ \sum_{j=1}^{J} \sum_{i \in \mathcal{B}_j} (\alpha_{ij1}^T (v_j - \omega_{ji}) + \alpha_{ij2}^T (v_i - \omega_{ji}))$$

Fully distributed SVM

- ADMM based algorithm:

$$\begin{aligned}
\{v_j^{t+1}, \xi_{jn}^{t+1}\} &= \text{argmin}_{\{v_j, \bar{\xi}_j\} \in \mathcal{W}} L(\{v_j\}, \{\bar{\xi}_j\}, \{\omega_{ji}^t\}, \{\alpha_{ijk}^t\}) \\
\{\omega_{ji}^{t+1}\} &= \text{argmin}_{\omega_{ji}} L(\{v_j\}^{t+1}, \{\bar{\xi}_j^{t+1}\}, \{\omega_{ji}\}, \{\alpha_{ijk}^t\}) \\
\alpha_{ji1}^{t+1} &= \alpha_{ji1}^t + \eta(v_j^{t+1} - \omega_{ji}^{t+1}) \\
\alpha_{ji2}^{t+1} &= \alpha_{ji2}^t + \eta(\omega_{ji}^{t+1} - v_i^{t+1})
\end{aligned}$$

Conclusions

- Good approximation to training SVM and other classifiers on Big data platforms is an open problem - tradeoff between computation and quality.
- Training SVM in a projected space can lead to efficient and accurate algorithms and bounds on stability w.r.t. generalization error.
- Future directions - applicability to:
  - Kernels methods.
  - Other supervised learning algorithms.
  - Unsupervised learning ??

Thank you !

Questions ?