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/

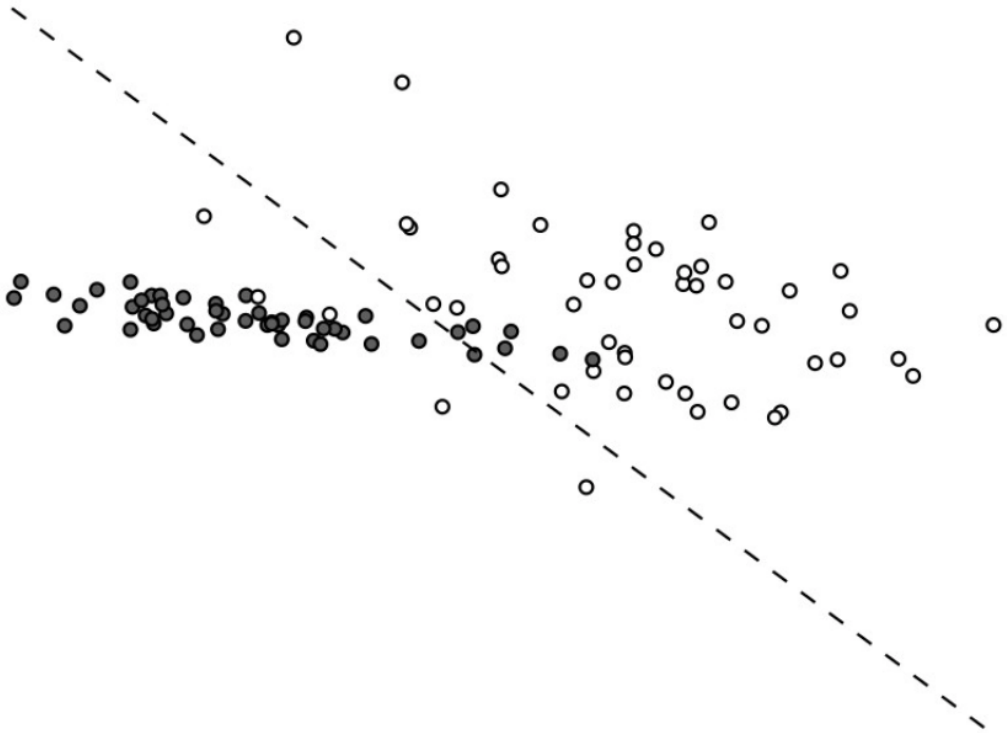
22 Dec, 2016

Outline

Outline

Classification Problem

- A set of labeled datapoints $(S) = \{(\mathbf{u}_i, v_i), i = 1, \ldots, n\}$, $u_i \in \mathbb{R}^d$ and $v_i \in \{+1, -1\}$
- Predictor function: $v = sign(\mathbf{x}^T \mathbf{u})$
- Error function: $E = \sum_{i=1}^{n} \mathbf{1}(v_i \mathbf{x}^T \mathbf{u}_i \leq 0)$



Logistic Regression

- Probability of v is given by:

$$P(v|u,x) = \sigma(vx^T u) = \frac{1}{1+e^{vx^T u}}$$

- Learning problem is:
  Given dataset $\mathcal{S}$ estimate $\mathbf{x}$.
- Regularized maximum likelihood:

$$x^* = \underset{x}{\text{argmin}} \sum_{i=1}^{n} \log(1+\exp(-v_i \mathbf{x}^T \mathbf{u}_i)) + \lambda \|\mathbf{x}\|_2^2$$

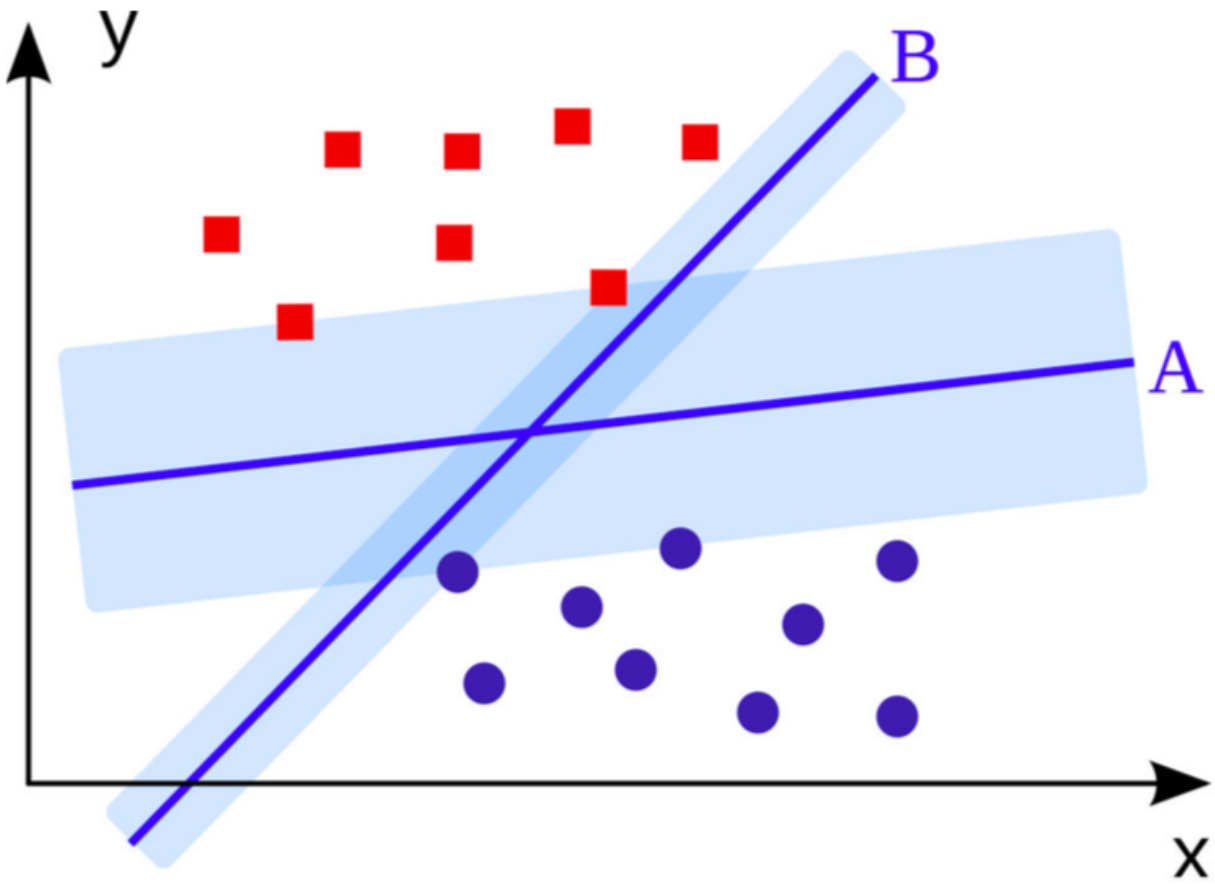
Support Vector Machines

- Separating hyperplane: $x^T u = 0$
- Parallel hyperplanes (developing margin): $x^T u = 1$
- Margin (perpendicular distance between parallel hyperplanes): $\frac{2}{\|x\|}$
- Correct classification of training datapoints: $v_i x^T u_i \geq 1$
- Allowing error(slack), ξ_i: $v_i x^T u_i \geq 1 - \xi_i, \forall i$
- Max-margin formulation:

$$\min_{\mathbf{x},\xi} \frac{1}{2}\|\boldsymbol{x}\|^2 + C\sum_{i=1}^{n} \xi_i$$

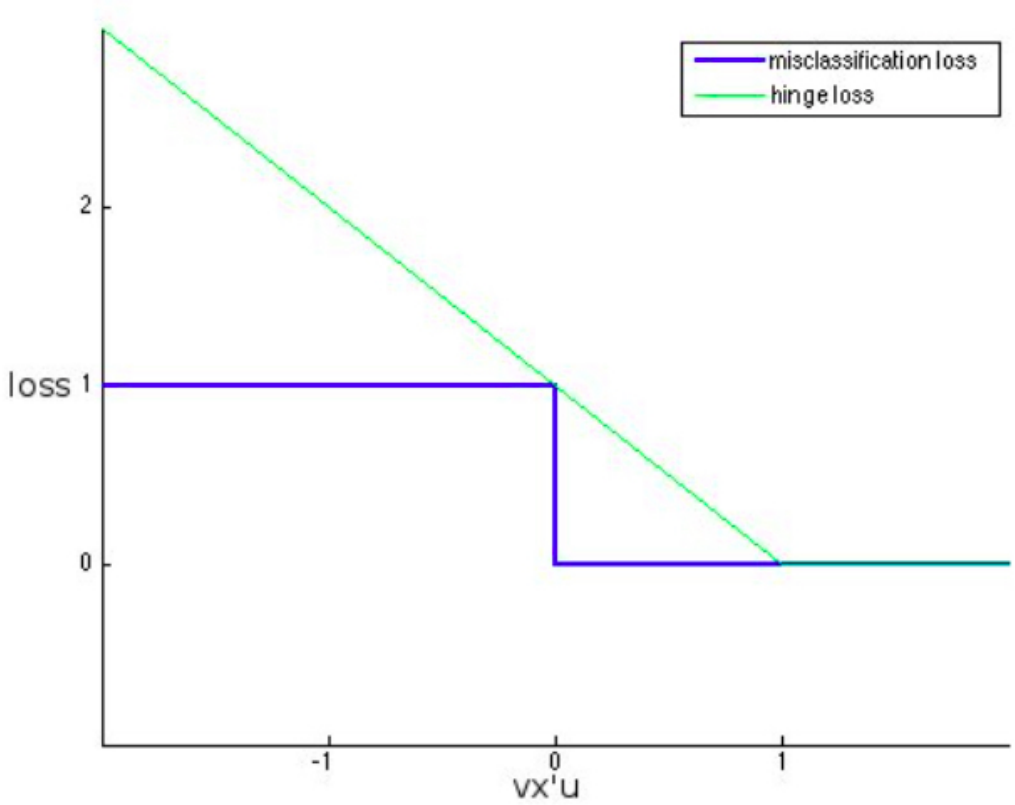
subject to: $v_i x^T u_i \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1, ..., n$

Support Vector Machines



Support Vector Machines

A more compact form: $\min_x \sum_{i=1}^{n} \max(0, 1 - v_i x^T u_i) + \|x\|^2$



Regularized Risk Minimization

- Support Vector Machines:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \max(0, 1 - v_i \mathbf{x}^T \mathbf{u}_i) + \lambda \|\mathbf{x}\|_2^2$$

- Logistic Regression:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \log(1 + \exp(-v_i \mathbf{x}^T \mathbf{u}_i)) + \lambda \|\mathbf{x}\|_2^2$$

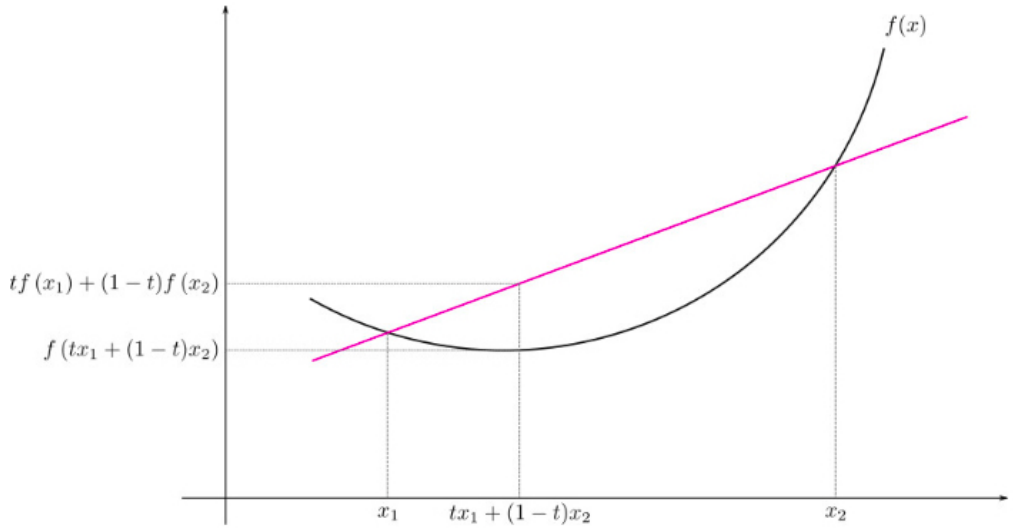
- General form:

$$\min_{\mathbf{x}} \sum_{i=1}^{n} l(\mathbf{x}, \mathbf{u}_i, v_i) + \lambda \Omega(\mathbf{x})$$

Outline

Convex optimization

f is a Convex function: $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$



Convex optimization

Convex Optimization Problem

$$\text{minimize}_x f(x)$$
$$\text{subject to: } g_i(x) \leq 0, \forall i = 1, ..., k$$

where:

- f, g_i are convex functions.
- For convex optimization problems, local optima are also global optima.

Optimization algorithm: Gradient descent

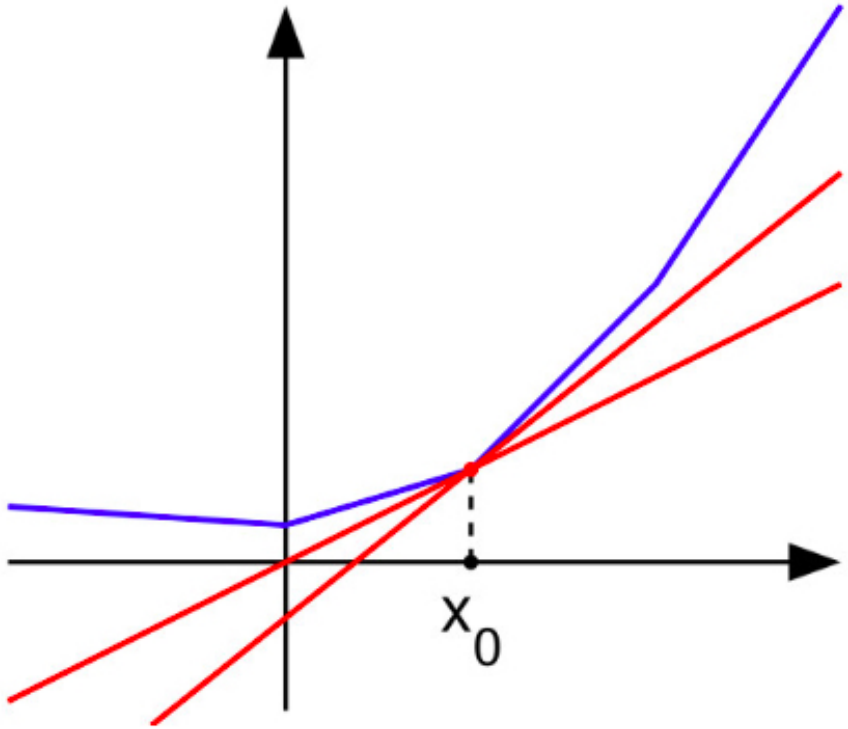
input : Function f, Gradient ∇f
output: Optimal solution w^*

Initialize $w_0 \leftarrow 0$, $k \leftarrow 0$
while $|\nabla f_k| > \epsilon$ **do**
    Compute $\alpha_k \leftarrow \text{linesearch}(f, -\nabla f_k, w_k)$
    Set $w_{k+1} \leftarrow w_k - \alpha_k \nabla f_k$
    Evaluate ∇f_{k+1}
    $k \leftarrow k + 1$
end
$w^* \leftarrow w_k$

Subgradient

Sub-gradient for a non-differentiable convex function f at a point x_0 is a vector v such that:

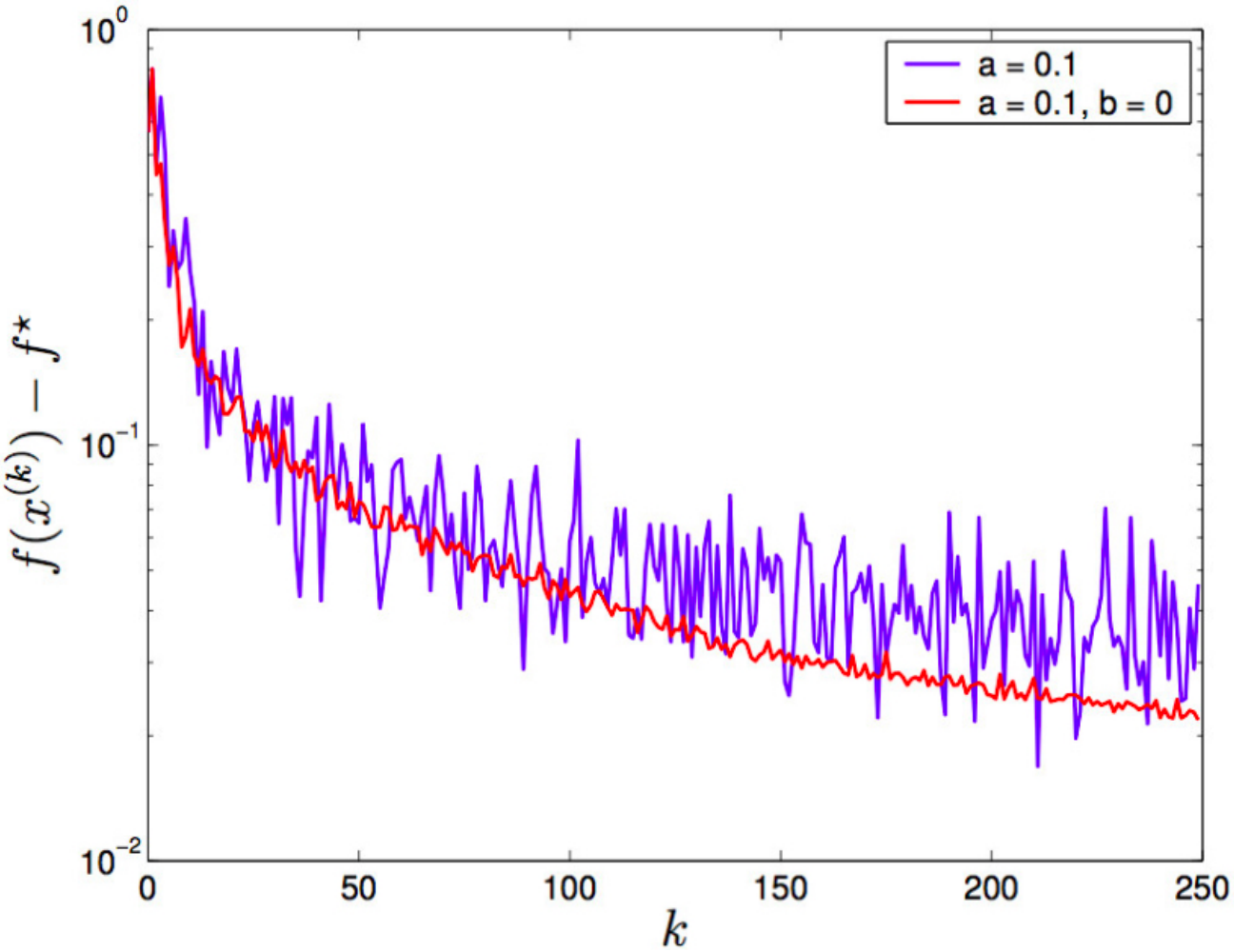
$$f(x) - f(x_0) \geq v^T(x - x0)$$



Subgradient Descent

- Randomly initialize x_0
- Iterate $x_k = x_{k-1} - t_k g(x_{k-1}), k = 1, 2, 3, ...$ Where g is a sub-gradient of f
- $t_k = \frac{1}{\sqrt{k}}$
- $x_{best}(k) = min_{i=1,...,k} f(x_k)$

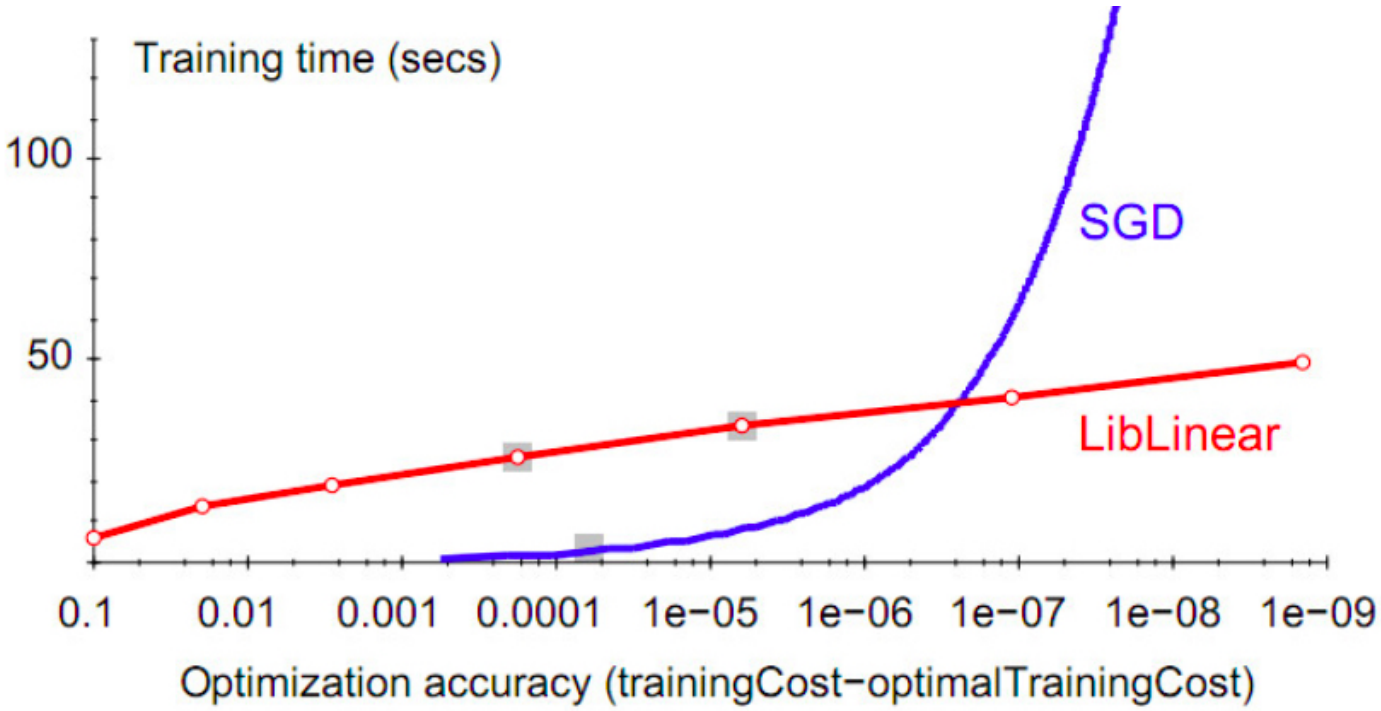
Subgradient descent Convergence



Stochastic Subgradient Descent

- Convergence rate is: $O(\frac{1}{\sqrt{k}})$.
- Each iteration takes $O(n)$ time.
- Reduce time by calculating the gradient using a subset of examples - stochastic subgradient.
- Inherently serial.
- Typical $O(\frac{1}{\epsilon^2})$ behaviour.

Stochastic Subgradient Descent



Big Data

- Classification - Spam / No Spam - 100B emails.
- Multi-label classification - image tagging - 14M images 10K tags.
- Regression - CTR estimation - 10B ad views.
- Ranking - web search - 6B queries.
- Recommendation - online shopping - 1.7B views in the US.
- Lots of $(\mathbf{u}_i, v_i)$.
- Distribute $(\mathbf{u}_i, v_i)$ into m computers.
- Solve the optimization problem in a distributed manner.

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:

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

$$y^{k+1} = y^k + \alpha^k (\sum_{i=1}^{N} A_i x_i - b)$$

- 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:

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

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 $\frac{\rho}{2}\|Ax - b\|_2^2$

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:

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

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 $\frac{\rho}{2}\|Ax - b\|_2^2$
- 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.

ADMM with scaled dual variables

- 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.

Convergence of ADMM

- Assumption 1: Functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ are closed, proper and convex.
  - Same as assuming 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 $k \to \infty$

Convergence proof

- Let objective function: $p^k = f(x^k) + g(z^k)$
- Let (x^*, z^*, y^*) be the saddle point and $p^* = f(x^*) + g(z^*)$
- **Result 1**:

$$p^* - p^k \leq y^{*T} r^{k+1}$$

- **Proof:**

$$L_0(x^*, z^*, y^*) \leq L_0(x^{k+1}, z^{k+1}, y^*)$$
$$p^* \leq p^{k+1} + y^{*T} r^{k+1}$$

Convergence proof

- **Result 2**:

$$p^{k+1} - p^* \leq -(y^{k+1})^T r^{k+1} - \rho(B(z^{k+1} - z^k))^T(-r^{k+1} + B(z^{k+1} - z^*))$$

- **Proof:**
  x^{k+1} minimizes $L_\rho(x, z^k, y^k)$. Hence:

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

  Hence, x^{k+1} minimizes $f(x) + (y^{k+1} - \rho B(z^{k+1} - z^k))^T Ax$.

Convergence proof

- **Proof (contd):**
  Similarly, z^{k+1} minimizes $g(z) + (y^{k+1})^T Bz$.
  Hence:

$$f(x^{k+1}) + (y^{k+1} - \rho B(z^{k+1} - z^k))^T Ax^{k+1}$$
$$\leq f(x^*) + (y^{k+1} - \rho B(z^{k+1} - z^k))^T Ax^*$$

  and $g(z^{k+1}) + (y^{k+1})^T Bz^{k+1} \leq g(z^*) + (y^{k+1})^T Bz^*$

Adding the above equations and rearranging, we get result 2.

Convergence proof

- Define a Lyapunov function, V^k:

$$V^k = \frac{1}{\rho}\|y^k - y^*\|_2^2 + \rho\|B(z^k - z^*)\|_2^2$$

- **Result 3**:

$$V^{k+1} \leq V^k - \rho\|r^{k+1}\|_2^2 - \rho\|B(z^{k+1} - z^k)\|_2^2$$

- **Proof:**
  Adding results 1 and 2, and multiplying by 2:

$$2(y^{k+1} - y^*)^T r^{k+1} - 2\rho(B(z^{k+1} - z^k))^T r^{k+1}$$
$$+ 2\rho(B(z^{k+1} - z^k))^T B(z^{k+1} - z^*) \leq 0$$

Convergence proof

- **Proof (contd):**
  Substituting $y^{k+1} = y^k + \rho r^{k+1}$ and rearranging, the first term becomes:

$$\frac{1}{\rho}(\|y^{k+1} - y^*\|_2^2 - \|y^k - y^*\|_2^2) + \rho\|r^{k+1}\|_2^2$$

  Similarly, the remaining terms can be written as:

$$\rho\|r^{k+1} - B(z^{k+1} - z^k)\|_2^2 + \rho(\|B(z^{k+1} - z^*)\|_2^2 - \|B(z^k - z^*)\|_2^2) - \rho\|r^{k+1}\|_2^2$$

  Hence we get:

$$V^k - V^{k+1} \geq \rho\|r^{k+1} - B(z^{k+1} - z^k)\|_2^2$$

Convergence proof

- **Proof (contd):**
  It suffices to show that $\rho(r^{k+1})^T(B(z^{k+1} - z^k)) \leq 0$.
  Using the facts that z^{k+1} minimizes $g(z) + (y^{k+1})^T Bz$ and z^k minimizes $g(z) + (y^k)^T Bz$:

$$g(z^{k+1}) + (y^{k+1})^T Bz^{k+1} \leq g(z^k) + (y^{k+1})^T Bz^k$$

  and

$$g(z^k) + (y^k)^T Bz^k \leq g(z^{k+1}) + (y^k)^T Bz^{k+1}$$

  Adding, we get result 3.

Convergence proof

- Summing result 3 over k, we get:

$$\rho \sum_{k=0}^{\infty} (\|r^{k+1}\|_2^2 + \|B(z^{k+1} - z^k)\|_2^2) \leq V_0$$

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

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.

Decomposition

- 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

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

Consensus Optimization

- Problem:

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

- ADMM form:

$$\min_{x_i, z} \sum_{i=1}^{N} f_i(x_i)$$
$$\text{s.t. } x_i - z = 0, \ i = 1, \ldots, N$$

- Augmented lagrangian:

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

Consensus Optimization

- ADMM algorithm:

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

$$z^{k+1} = \frac{1}{N}\sum_{i=1}^{N}(x_i^{k+1} + \frac{1}{\rho}y_i^k)$$

$$y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - z^{k+1})$$

- 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.

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 solution:

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

Fully distributed SVM

- SVM optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{j=1}^{J}\sum_{n=1}^{n_j}\xi_{jn}$$

$$\text{s.t.: } y_{jn}(w^t x_{jn} + b) \geq 1 - \xi_{jn},\ \forall j \in J, n = 1, \ldots, N_j$$

$$\xi_{jn} \geq 0,\ \forall j \in J, n = 1, \ldots, N_j$$

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

$$\min_{\{w_j,b_j,\xi_{jn}\}} \frac{1}{2}\sum_{j=1}^{J}\|w_j\|^2 + JC\sum_{j=1}^{J}\sum_{n=1}^{n_j}\xi_{jn}$$

$$\text{s.t.: } y_{jn}(w_j^t x_{jn} + b) \geq 1 - \xi_{jn},\ \forall j \in J, n = 1, \ldots, N_j$$

$$\xi_{jn} \geq 0,\ \forall j \in J, n = 1, \ldots, N_j$$

$$w_j = w_i,\ \forall j, i \in \mathcal{B}_j$$

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:

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

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

Fully distributed SVM

- ADMM based algorithm:

$$\{v_j^{t+1}, \xi_{jn}^{t+1}\} = \text{argmin}_{\{v_j, \bar{\xi}_j\} \in \mathcal{W}} L(\{v_j\}, \{\bar{\xi}_j\}, \{\omega_{ji}^t\}, \{\alpha_{ijk}^t\})$$

$$\{\omega_{ji}^{t+1}\} = \text{argmin}_{\omega_{ji}} L(\{v_j\}^{t+1}, \{\bar{\xi}_j^{t+1}\}, \{\omega_{ji}\}, \{\alpha_{ijk}^t\})$$

$$\alpha_{ji1}^{t+1} = \alpha_{ji1}^t + \eta(v_j^{t+1} - \omega_{ji}^{t+1})$$

$$\alpha_{ji2}^{t+1} = \alpha_{ji2}^t + \eta(\omega_{ji}^{t+1} - v_i^{t+1})$$

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” proposed in [**?**], for logistic regression.
- Results hold true for SVMs with suitable sub-derivative.
- Locally learn SVM on $\mathcal{S}_m$:

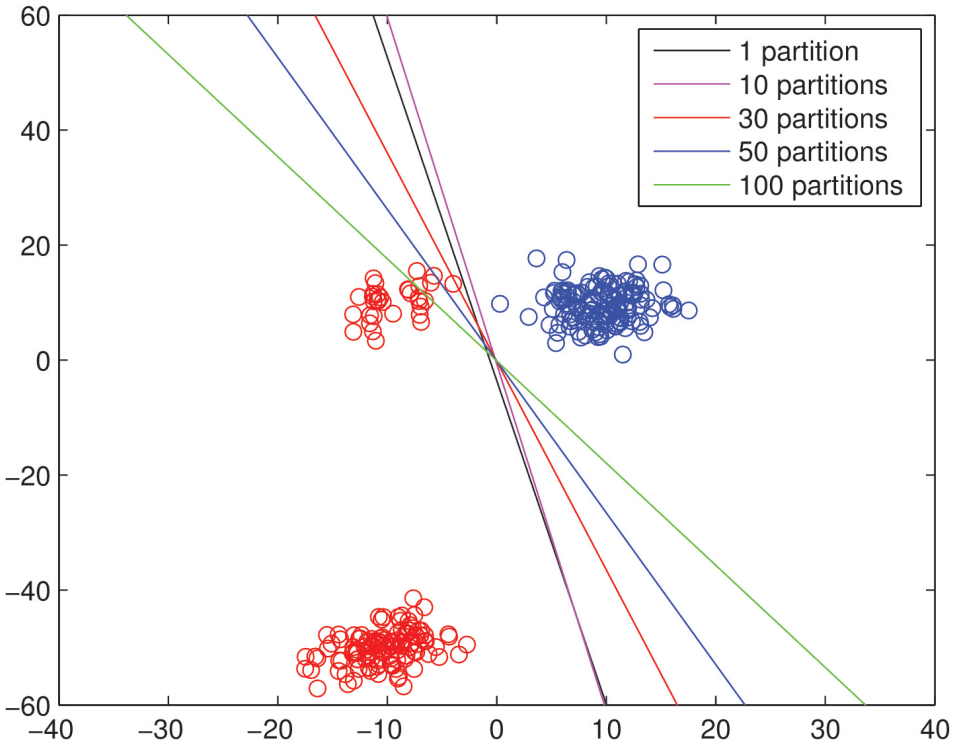
$$\hat{\mathbf{w}}_m = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{L} \sum_{l=1}^{L} loss(\mathbf{w}; \mathbf{x}_{ml}, y_{ml}) + \lambda \|\mathbf{w}\|^2 \ , \ m = 1, \ldots, M$$

- The final SVM parameter is given by:

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

Problem with Parameter Averaging

PA with varying number of partitions - Toy dataset.



Outline

Weighted Parameter Averaging

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

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

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

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

Weighted Parameter Averaging

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

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

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

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

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

Dual Weighted Parameter Averaging

- Lagrangian:

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

$$+ \sum_{m,i} \alpha_{mi}(y_{mi}(\boldsymbol{\beta}^T W^T \mathbf{x}_{mi}) - 1 + \xi_{mi}) - \sum_{m,i} \mu_{mi}\xi_{mi}$$

- Differentiating w.r.t. $\boldsymbol{\beta}$ and equating to zero:

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

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 $\mathcal{L}$:

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

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:

$$\gamma_m^{k+1} := \underset{\gamma}{\text{argmin}}(loss(\mathbf{A}_i\gamma) + (\rho/2)\|\gamma - \beta^k + \mathbf{u}_m^k\|_2^2)$$

$$\beta^{k+1} := \underset{\beta}{\text{argmin}}(r(\beta) + (M\rho/2)\|\beta - \overline{\gamma}^{k+1} - \overline{\mathbf{u}}^k\|_2^2)$$

$$\mathbf{u}_m^{k+1} = \mathbf{u}_m^k + \gamma_m^{k+1} - \beta^{k+1}.$$

- $\mathbf{u}_m$ 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

Parameter Stability Bound

- Conditional Maxent model [**?**]: $p_{\boldsymbol{w}}[y|x] = \frac{1}{Z(x)} \exp(\boldsymbol{w}.\Phi(x,y))$
- Training on $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$:
  $\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}}\, F_S(\boldsymbol{w}) = \text{argmin}_{\boldsymbol{w}}\, \lambda \|\boldsymbol{w}\|^2 - \frac{1}{m} \sum_{i=1}^{m} \log p_{\boldsymbol{w}}[y_i|x_i]$

Theorem [**?**]: Parameter Stability

Let S' and S be two arbitrary training samples of size m differing only by one point. Then, the following stability bound holds for the weight vector returned by a conditional maxent model:

$$\|\delta \boldsymbol{w}\| \leq \frac{2R}{\lambda m}$$

where $\|\delta \boldsymbol{w}\| = \|\boldsymbol{w} - \boldsymbol{w}'\|$, R is an upper bound on $\|\nabla loss(\boldsymbol{w})\|$.

Parameter Stability Bound

Proof relies on:

- Symmetric Bregmann divergence w.r.t. regularizer less than symmetric Bregmann divergence of the objective.
- Convexity of loss function.

Theorem also holds for non-differentiable convex losses with appropriate definition of sub-gradient.

Parameter Stability Bound

- Minimizer of generalization error: $\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} E_S[F_S(\boldsymbol{w})]$.

Theorem [?]: Bound on parameter difference

Let $\boldsymbol{w}$ be weight vector returned by conditional maxent model when trained on a sample of size m. Then, for any $\delta \geq 0$, with probability at least $1 - \delta$, the following inequality holds:

$$\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq E[\|\boldsymbol{w} - \boldsymbol{w}^*\|] + \frac{2R}{\lambda}\sqrt{\frac{\log \frac{1}{\delta}}{2m}} \leq \frac{2R}{\lambda\sqrt{2m}}\left(1 + \sqrt{\log \frac{1}{\delta}}\right)$$

- Mann et al. [?] show bounds on $\|\boldsymbol{w}_\mu - \boldsymbol{w}^*\|$, where $\boldsymbol{w}_\mu$ is the output of parameter averaging.

Stability of weighted parameter averaging

- Let $S = \{S_1, \cdots, S_M\}$ and $S' = \{S'_1, \cdots, S'_M\}$ be two datasets with M partitions and L datapoints per partition, differing in only one datapoint in the M^{th} partition.
- Let $\hat{W} = [\hat{\mathbf{w}}_{S_1}, \cdots, \hat{\mathbf{w}}_{S_M}]$ and $\hat{W}' = [\hat{\mathbf{w}}_{S'_1}, \cdots, \hat{\mathbf{w}}_{S'_M}]$ where, $\hat{\mathbf{w}}_{S_i} = \text{argmin}_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \frac{1}{L} \sum_{i \in S_i} max(0, 1 - y\mathbf{w}^T\mathbf{x})$.
- Also, define:

$$\beta = \text{argmin}_{\beta} \lambda \|\hat{W}\beta\|^2 + \frac{1}{ML} \sum_{i=1}^{M} \sum_{z \in S_i} max(0, 1 - y(\hat{W}\beta)^T\mathbf{x})$$

$$\beta' = \text{argmin}_{\beta} \lambda \|\hat{W}'\beta\|^2 + \frac{1}{ML} \sum_{i=1}^{M} \sum_{z' \in S'_i} max(0, 1 - y'(\hat{W}\beta)^T\mathbf{x}')$$

Stability of weighted parameter averaging

- Let $\theta = \hat{W}\beta$ and $\theta' = \hat{W}'\beta'$.

Theorem: Stability of θ

$$\|\theta - \theta'\| \text{ is } \mathcal{O}(\tfrac{1}{ML}).$$

- Result similar to Mann et al. [**?**], but for a more general scenario.
- Proof is non-trivial since both β and $\hat{W}$ changes with S.

Stability of weighted parameter averaging

- Define:
  $\tilde{\beta} = \text{argmin}_{\beta}\, \lambda \|\hat{W}'\beta\|^2 + \frac{1}{ML}\sum_{i=1}^{M}\sum_{z\in S_i} max(0, 1 - y(\hat{W}'\beta)^T\mathbf{x})$
- Also, $\tilde{\theta} = \hat{W}'\tilde{\beta}$.
- Decompose as: $\|\theta - \theta'\| \leq \|\theta - \tilde{\theta}\| + \|\tilde{\theta} - \theta'\|$
- **Lemma:** $\|\tilde{\theta} - \theta'\| = \mathcal{O}(\frac{1}{ML})$
- **Proof**: Similar to proof by Mann et al. [**?**], except using $\|\cdot\|_K$, where, $K = \hat{W}'^T\hat{W}'$ instead of the Euclidean norm.

Stability of weighted parameter averaging

- **Theorem:** $\|\theta - \tilde{\theta}\| = \mathcal{O}(\frac{1}{ML})$
- **Proof:**

$$\begin{aligned}\|\theta - \tilde{\theta}\| &\leq \frac{1}{2}(\|(\hat{W} - \hat{W}')(\beta + \tilde{\beta})\| + \|(\hat{W} + \hat{W}')(\beta - \tilde{\beta})\|) \\ &\leq \frac{1}{2}((\|(\hat{W} - \hat{W}'\|)(\|\beta + \tilde{\beta}\|) + (\|\hat{W} + \hat{W}'\|)(\|\beta - \tilde{\beta}\|))\end{aligned}$$

Hence it suffices to show that $\|\hat{W} - \hat{W}'\| = \mathcal{O}(\frac{1}{ML})$ and $\|\beta - \tilde{\beta}\| = \mathcal{O}(\frac{1}{ML})$.

Stability of weighted parameter averaging

- **Lemma:** $\|\hat{W} - \hat{W}'\| = \mathcal{O}(\frac{1}{ML})$
- **Proof:** Since $\|\hat{W}\| = 1$, $\|\hat{w}_m\|^2 = \|\hat{w}'_m\|^2 = \frac{1}{M}$.
- It suffices to show that $M\|\hat{w} - \hat{w}'\| = \mathcal{O}(\frac{1}{L})$.
- Analogous to theorem 1 of [**?**].

Stability of weighted parameter averaging

- **Lemma:** $\|\beta - \tilde{\beta}\| = \mathcal{O}(\frac{1}{ML})$
- Define: $F_W(\beta) = G_W(\beta) + L_W(\beta)$
- **Proof:** $B_{G_{\hat{W}}}(\tilde{\beta}\|\beta) + B_{G_{\hat{W}'}}(\beta\|\tilde{\beta}) \leq B_{F_{\hat{W}}}(\tilde{\beta}\|\beta) + B_{F_{\hat{W}'}}(\beta\|\tilde{\beta})$

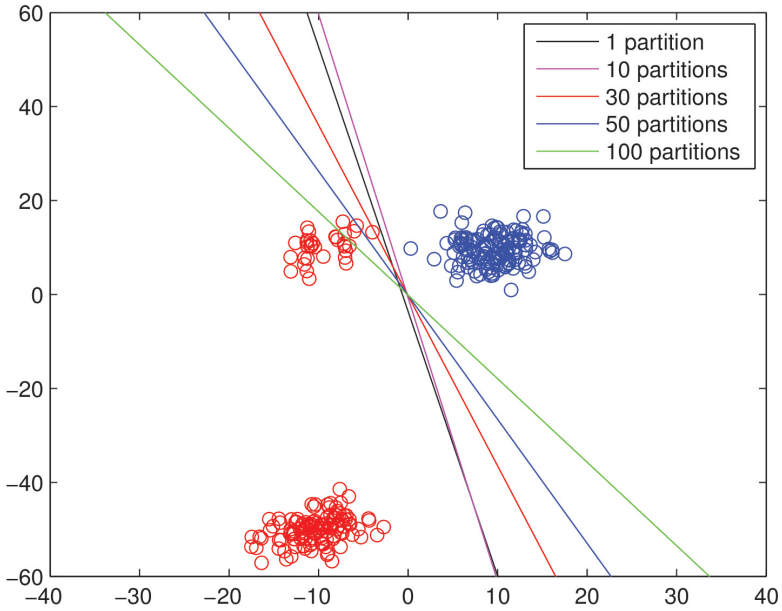
$$LHS = \lambda\|\tilde{\beta} - \beta\|^T(\|\hat{W}^T\hat{W} + \hat{W}'^T\hat{W}'\|)\|\tilde{\beta} - \beta\|$$
$$= \lambda\|\beta - \tilde{\beta}\|^2_{K'}, \text{ where, } K' = \hat{W}^T\hat{W} + \hat{W}'^T\hat{W}'$$

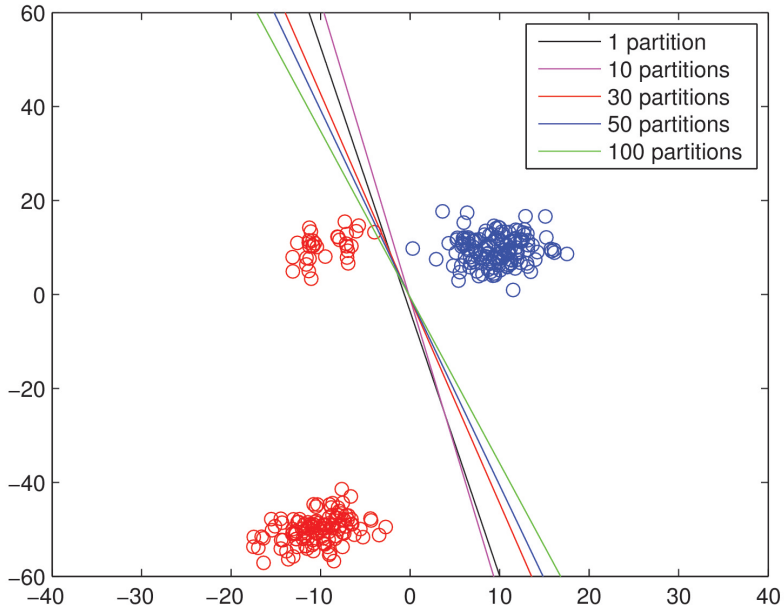
- $RHS \leq \frac{4\lambda R}{ML}\|\beta - \tilde{\beta}\| + \frac{R}{ML}\|(\beta - \tilde{\beta}))\|$
- $\lambda\|\beta - \tilde{\beta}\|_2^2 \leq \frac{\lambda}{\sigma_{min}}\|\beta - \tilde{\beta}\|^2_{K'} \leq RHS$. Hence proved !!

Outline

Toy Dataset - PA and WPA

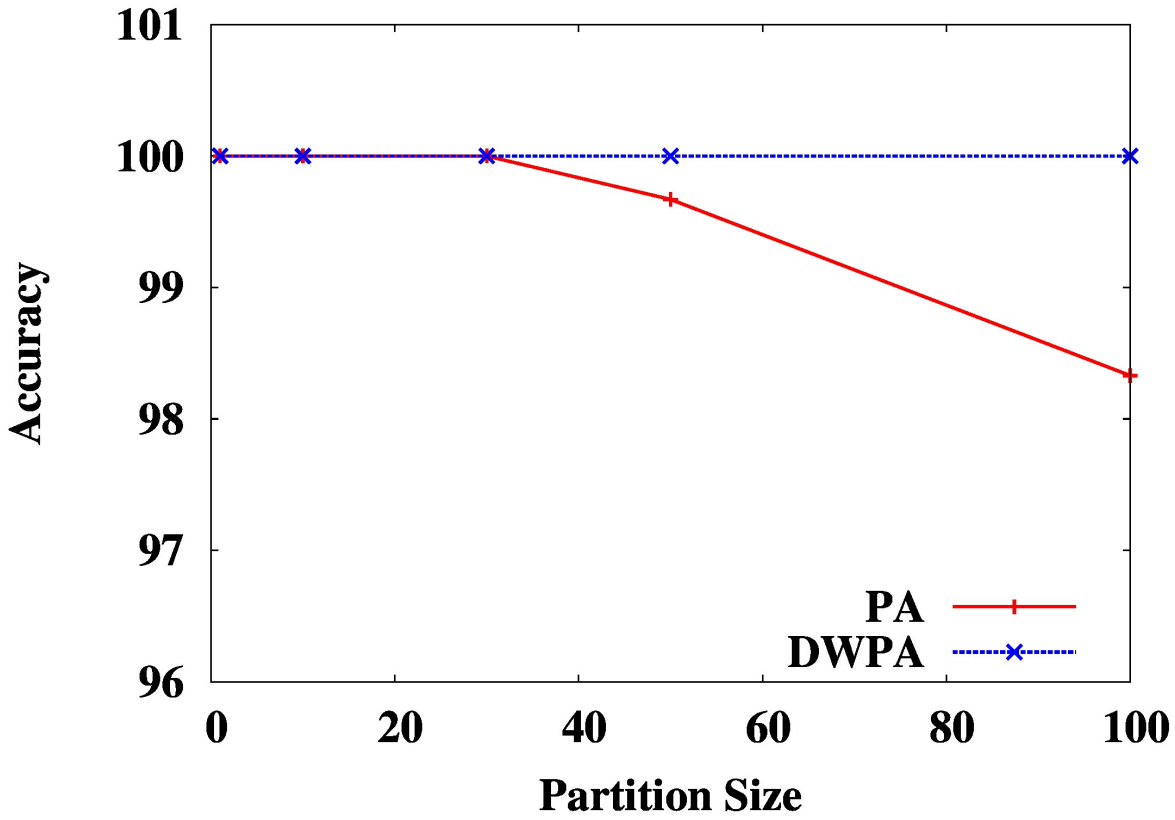
PA (left) and WPA (right) with varying number of partitions - Toy dataset.





Toy Dataset - PA and WPA

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



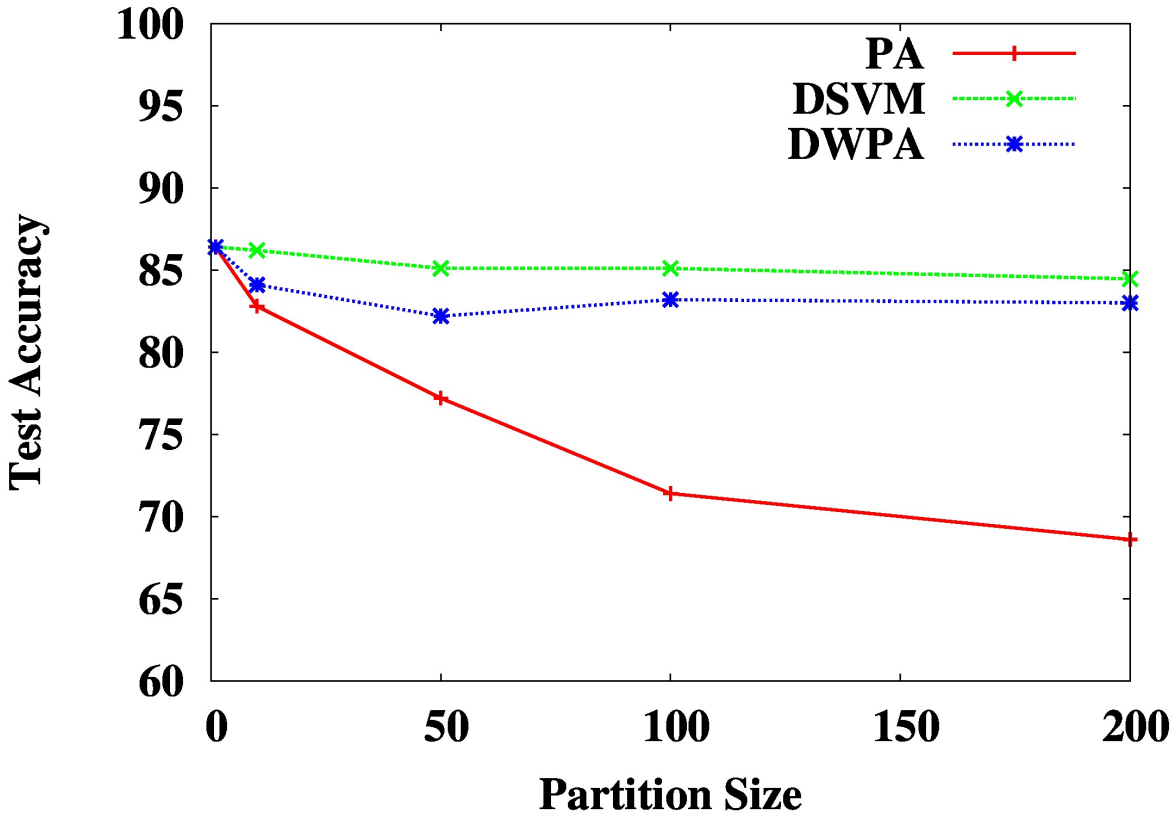
Toy Dataset - PA and WPA

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

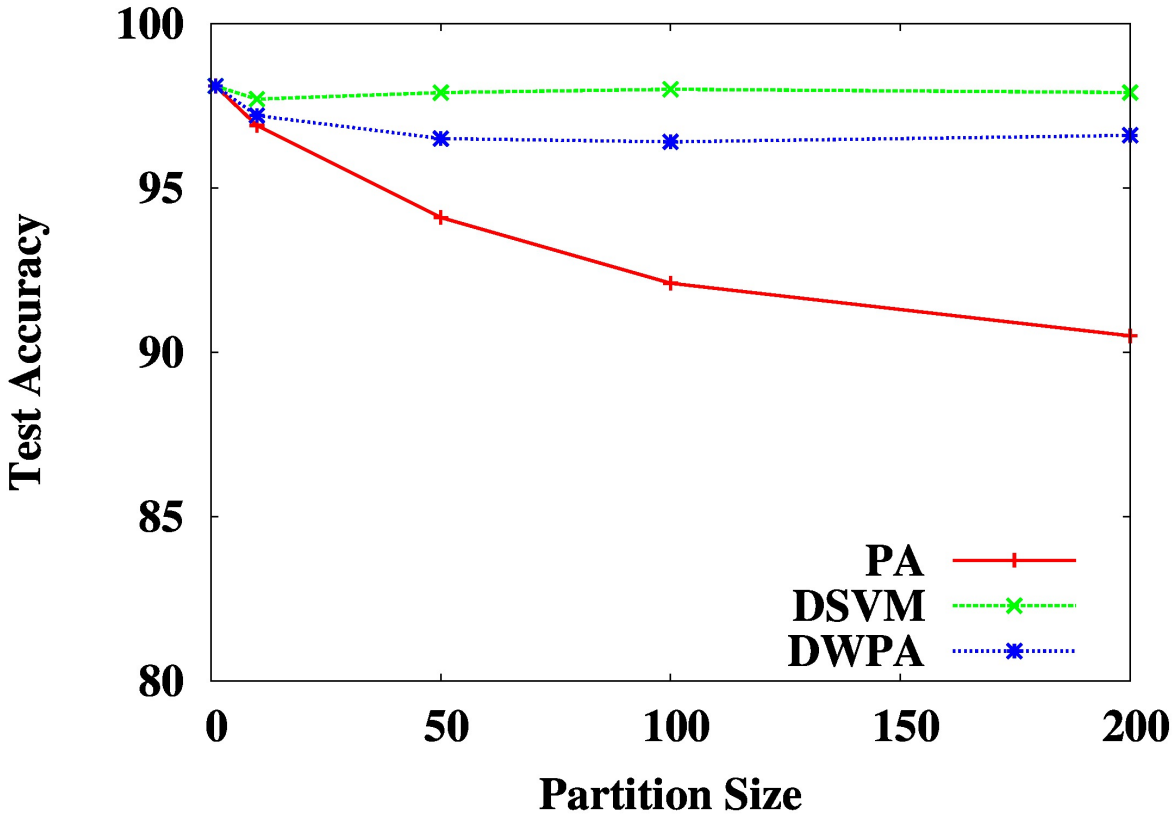
Real World Datasets

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



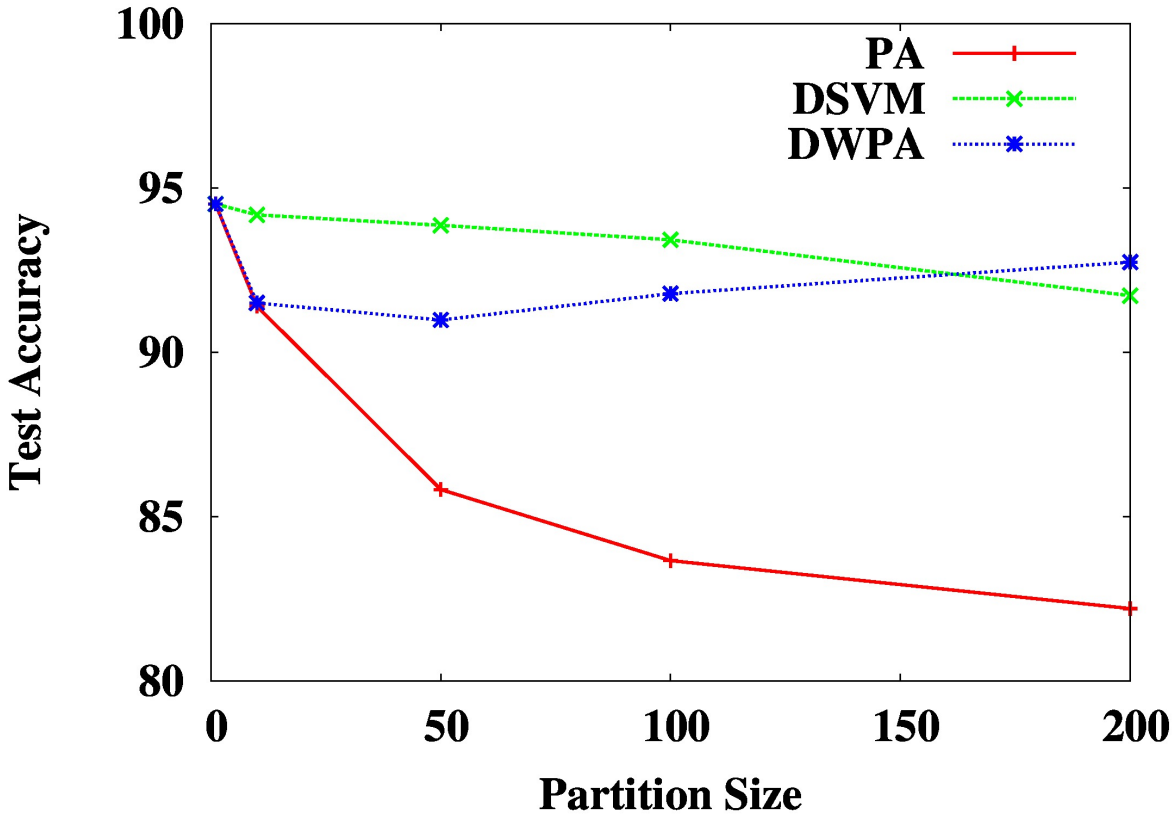
Real World Datasets

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



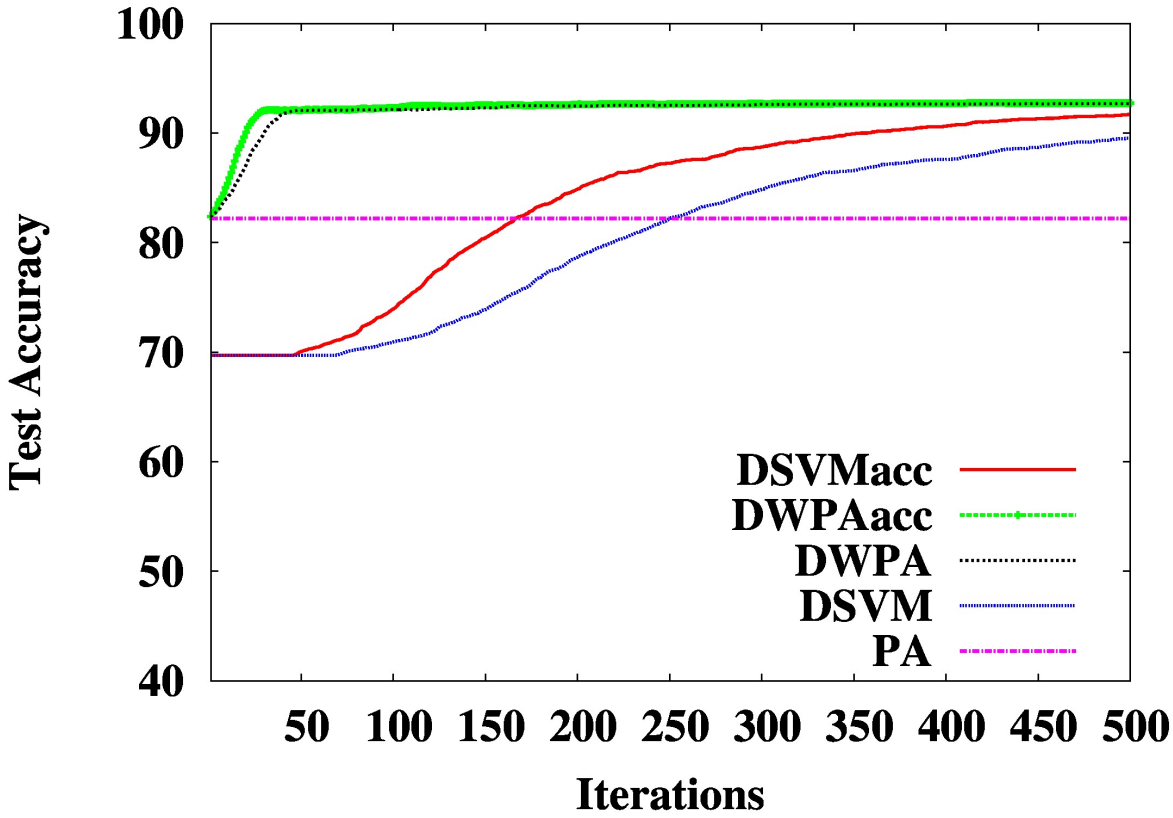
Real World Datasets

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



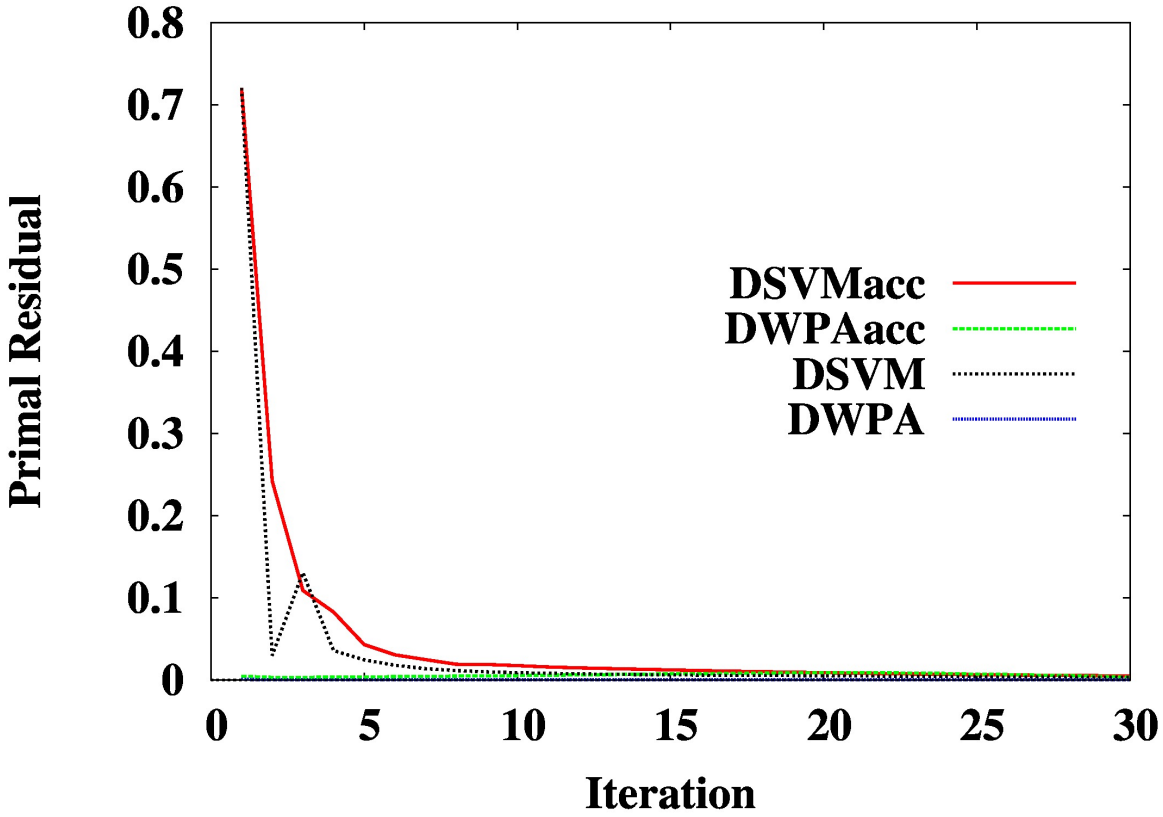
Real World Datasets

Convergence of test accuracy with iterations (200 partitions).



Real World Datasets

Convergence of primal residual with iterations (200 partitions).



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 ?