Distributed Machine Learning: Iterative Convex Optimization Methods

Krishna Pillutla

IIT Bombay

krishna.p@iitb.ac.in
www.cse.iitb.ac.in/~krishna.p

May 5, 2014

Advisor

Prof. SakethaNath J

saketh@cse.iitb.ac.in

Overview

1. Introduction
2. IPM
   - Literature Review
   - Algorithm
3. ADMM
   - Literature Review
   - A general framework for distributed optimization
   - Approach
4. Experiments and Results

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)
  - w is the model

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)
  - w is the model
  - Regularizer $R(w) = w^T w/2$

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)
  - w is the model
  - Regularizer $R(w) = w^T w/2$
- We wish to solve it in a distributed setting

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)
  - w is the model
  - Regularizer $R(w) = w^T w/2$
- We wish to solve it in a distributed setting
  - data is distributed across the nodes.

Introduction

- To solve learning problems of the form

$$min_w \sum_{i=1}^{N} l(y_i w^T x_i) + \gamma R(w)$$

  - l is the loss function (convex and ∇l is Lipschitz continuous)
  - x_i a training example and y_i is its label ($+1$ or -1 for binary classification)
  - w is the model
  - Regularizer $R(w) = w^T w/2$
- We wish to solve it in a distributed setting
  - data is distributed across the nodes.
  - locally optimise and communicate the models to get one common global model.

Previous Work: Parameter Mixing

- Parameter Mixing(PM): Independently solve optimisation on each node and take a convex combination of these to represent the global model [Man], [MMM$^+$09].

Previous Work: Parameter Mixing

- Parameter Mixing(PM): Independently solve optimisation on each node and take a convex combination of these to represent the global model [Man], [MMM$^+$09].
- Use local gradient information in coefficients of the convex combination (to give weights to different components) [ACDL11]

Previous Work: Parameter Mixing

- Parameter Mixing(PM): Independently solve optimisation on each node and take a convex combination of these to represent the global model [Man], [MMM$^+$09].
- Use local gradient information in coefficients of the convex combination (to give weights to different components) [ACDL11]
- Use gradient and hessian information from quadratic Taylor approximations from other nodes (did in my Summer Internship at MSR).

Previous Work: Parameter Mixing

- Parameter Mixing(PM): Independently solve optimisation on each node and take a convex combination of these to represent the global model [Man], [MMM$^+$09].
- Use local gradient information in coefficients of the convex combination (to give weights to different components) [ACDL11]
- Use gradient and hessian information from quadratic Taylor approximations from other nodes (did in my Summer Internship at MSR).
- What if the Taylor approximation does not hold?

Previous Work: Iterative Parameter Mixing

- IPM: Run multiple iterations of PM- by using the model obtained at each PM step as the starting guess for the next step- until global convergence. Theoretical bounds exist for the perceptron [MHM]

Previous Work: Iterative Parameter Mixing

- IPM: Run multiple iterations of PM- by using the model obtained at each PM step as the starting guess for the next step- until global convergence. Theoretical bounds exist for the perceptron [MHM]
- [MKSB13] uses functional approximation for IPM (published in January 2014, after stage 1). It has theoretical guarantees but requires a global line search step.

Previous Work: Iterative Parameter Mixing

- IPM: Run multiple iterations of PM- by using the model obtained at each PM step as the starting guess for the next step- until global convergence. Theoretical bounds exist for the perceptron [MHM]
- [MKSB13] uses functional approximation for IPM (published in January 2014, after stage 1). It has theoretical guarantees but requires a global line search step.
- Node i minimises $f_i(w) + C_i(w)$ where $C_i(w)$ is a quadratic satisfying some mild requirements.

Previous Work: Iterative Parameter Mixing

- IPM: Run multiple iterations of PM- by using the model obtained at each PM step as the starting guess for the next step- until global convergence. Theoretical bounds exist for the perceptron [MHM]
- [MKSB13] uses functional approximation for IPM (published in January 2014, after stage 1). It has theoretical guarantees but requires a global line search step.
- Node i minimises $f_i(w) + C_i(w)$ where $C_i(w)$ is a quadratic satisfying some mild requirements.
- Can we achieve theoretical guarantees and practical results without this line search step?

Approach

- Each nodes approximates the objective function at other nodes by a linear or quadratic approximation about the globally accepted starting point.

Approach

- Each nodes approximates the objective function at other nodes by a linear or quadratic approximation about the globally accepted starting point.
- This way, each node has a global picture.

Approach

- Each node has an objective function $f_i(w)$.

Approach

- Each node has an objective function $f_i(w)$.
- If L is the Lipschitz constant of ∇f, define

$$\hat{f}^{w_0}(w) := f(w_0) + \nabla f(w_0)^T (w - w_0) + \frac{L}{2}\|w - w_0\|^2$$

Approach

- Each node has an objective function $f_i(w)$.
- If L is the Lipschitz constant of ∇f, define

$$\hat{f}^{w_0}(w) := f(w_0) + \nabla f(w_0)^T(w - w_0) + \frac{L}{2}\|w - w_0\|^2$$

- If $w^{(k)}$ is the global model from the previous iteration, node i solves,

$$\min_w f_i(w) + \sum_{j \neq i} \hat{f}_j^{w^{(k)}}(w)$$

Algorithm

Algorithm 1 Our Algorithm for IPM

1: Initialise $w^{(0)}$
2: **for** $t = 1, 2, ..$ (outer iterations) **do**
3: $\quad \tilde{f}_i^{(t)}(w) = f_i(w) + \hat{f_{-i}}^{w^{(k)}}(w)$
4: $\quad w_i^{(t)} = \underset{w}{\operatorname{argmin}}(\tilde{f}_{i,t}(w))$ by some method
5: $\quad w^{(t+1)} = \texttt{ParameterMixing}(w_i^{(t)})$
6: $\quad$ Obtain $f(w^{(t+1)})$ and $\nabla f(w^{(t+1)})$ by communication
7: **end for**
8: **return** $w^{(t+1)}$

$$\texttt{ParameterMixing}(w_i^{(t)}) = \sum_{i=1}^{m} \alpha_i w_i^{(t)}$$

Algorithm from [MKSB13]

Algorithm 2 IPM Algorithm proposed in [MKSB13] applied to our setting

1: Initialise $w^{(0)}$
2: **for** $t = 1, 2, ..$ (outer iterations) **do**
3: $\quad \tilde{f}_i^{(t)}(w) = f_i(w) + \hat{f}_{-i}^{w^{(k)}}(w)$
4: $\quad w_i^{(t)} = \text{argmin}_w(\tilde{f}_{i,t}(w))$ by some method
5: $\quad d^{(t)} = \texttt{ParameterMixing}(w_i^{(t)}) - w^{(t)}$
6: $\quad w^{(t+1)} = w^{(t)} + \tau d^{(t)}$ where τ is a step length satisfying Armijo-Wolfe conditions.
7: $\quad$ Obtain $f(w^{(t+1)})$ and $\nabla f(w^{(t+1)})$ by communication
8: **end for**
9: **return** $w^{(t+1)}$

Algorithm from [MKSB13]

- Linear Convergence is guaranteed for Algorithm 2 [MKSB13].

Theorem

Theorem (Convergence)

If f is convex and differentiable, ∇f is Lipschitz continuous with constant L, a strict decrease in global objective function f can be guaranteed in every outer iteration (parameter mixing step) under suitable conditions. In particular, for gradient descent, the condition is that the step size h satisfies

$$0 < h \leq 1/L$$

Because f is convex, IPM converges to the unique global minimizer w^*.

Theorem

Local Rate of Convergence:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L

Theorem

Local Rate of Convergence:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations) is used for inner optimization

Theorem

Local Rate of Convergence:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations) is used for inner optimization
- $w^{(0)}$ is sufficiently close to the global optimum w^*

Theorem

Local Rate of Convergence:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations) is used for inner optimization
- $w^{(0)}$ is sufficiently close to the global optimum w^*
- Convergence of IPM is $\mathcal{O}(1/k)$, for k outer iterations

Theorem

Local Rate of Convergence:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations) is used for inner optimization
- $w^{(0)}$ is sufficiently close to the global optimum w^*
- Convergence of IPM is $\mathcal{O}(1/k)$, for k outer iterations
- If f is strongly convex, convergence is linear.

Theorem

Theorem (Local Rate of Convergence)

If f if convex and differentiable, ∇f is Lipschitz continuous with constant L, the inner optimisation is solved with gradient descent i.e. $w_i^{(k,j+1)} = w_i^{(k,j)} - h\nabla \tilde{f}_{i,k}(w_i^{(k,j)})$ with a fixed number of steps c and fixed step size $h = 1/L(c^2 + 2c - 2)$ and the initial guess $w^{(0)}$ is sufficiently close to the global optimum w^* then IPM converges as

$$f(w^{(k)}) - f(w^*) \leq \frac{2L\|w^{(0)} - w^*\|^2}{k+4}\beta^2$$

where $\beta > 0$ is a constant i.e., convergence is $\mathcal{O}(1/k)$, for k outer iterations. If f is strongly convex with constant μ, convergence is linear as $\|w^{(k)} - w^*\| \leq (\frac{L-\mu}{L+\mu})^k \|w^{(0)} - w^*\|$

Theorem

Global Rate of Convergence

Theorem

We tried to but were unable to prove this theorem:

Theorem (Global Rate of Convergence)

If f if convex and differentiable, ∇f is Lipschitz continuous with constant L, the inner optimisation is solved with gradient descent i.e. $w_i^{(k,j+1)} = w_i^{(k,j)} - h\nabla \tilde{f}_{i,k}(w_i^{(k,j)})$ with a fixed number of steps c and fixed step size $h = 1/L(c^2 + 2c - 2)$, then IPM converges as $f(w^{(k)}) - f(w^*) \leq \frac{2L\|w^{(0)} - w^*\|^2}{k+4}\beta^2$ where $\beta > 0$ is a constant i.e., convergence is $\mathcal{O}(1/k)$, for k outer iterations. If f is strongly convex with constant μ, convergence is linear as $\|w^{(k)} - w^*\| \leq (\frac{L-\mu}{L+\mu})^k \|w^{(0)} - w^*\|$

Theorem

We had to settle for a weaker bound:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L

Theorem

We had to settle for a weaker bound:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations or with line search) is used for inner optimization

Theorem

We had to settle for a weaker bound:

- f if convex and differentiable, ∇f is Lipschitz continuous with constant L
- Gradient Descent (fixed number of iterations or with line search) is used for inner optimization
- Convergence of IPM is $\mathcal{O}(1/\sqrt{k})$, for k outer iterations

Theorem

Theorem (Global Rate of Convergence)

If f if convex and differentiable, ∇f is Lipschitz continuous with constant L, the inner optimization is solved with gradient descent i.e. $w_i^{(k,j+1)} = w_i^{(k,j)} - h\nabla \tilde{f}_{i,k}(w_i^{(k,j)})$ with a fixed number of steps c, and a fixed step size of $h = 1/L$, we have,

$$\|\nabla f(w^{(k)}\| \leq \sqrt{\frac{2L(f(w^{(0)} - f(w^*))}{k+1}}$$

In other words, convergence is $\mathcal{O}(1/\sqrt{k})$, for k outer iterations.

Previous Work: Alternating Direction Method of Multipliers

- A separable problem of the form $min_w f(w) + g(w)$ is recast as $min_{x,y} f(x) + g(y)$ subject to $x = y$.

[1] similar in spirit to the Jacobi method used to solve diagonally dominant systems of linear equations

Previous Work: Alternating Direction Method of Multipliers

- A separable problem of the form $min_w f(w) + g(w)$ is recast as $min_{x,y} f(x) + g(y)$ subject to $x = y$.
- Based on theory of Lagrange Duality.

[1] similar in spirit to the Jacobi method used to solve diagonally dominant systems of linear equations

Previous Work: Alternating Direction Method of Multipliers

- A separable problem of the form $min_w f(w) + g(w)$ is recast as $min_{x,y} f(x) + g(y)$ subject to $x = y$.
- Based on theory of Lagrange Duality.
- Because of the separable nature, x is updated keeping y fixed and then, y is updated keeping x fixed [1].

[1] similar in spirit to the Jacobi method used to solve diagonally dominant systems of linear equations

Previous Work: Alternating Direction Method of Multipliers

- A separable problem of the form $min_w f(w) + g(w)$ is recast as $min_{x,y} f(x) + g(y)$ subject to $x = y$.
- Based on theory of Lagrange Duality.
- Because of the separable nature, x is updated keeping y fixed and then, y is updated keeping x fixed [1].
- Dual variables are updated and the process is repeated.

[1] similar in spirit to the Jacobi method used to solve diagonally dominant systems of linear equations

Previous Work: Alternating Direction Method of Multipliers

- A separable problem of the form $min_w f(w) + g(w)$ is recast as $min_{x,y} f(x) + g(y)$ subject to $x = y$.
- Based on theory of Lagrange Duality.
- Because of the separable nature, x is updated keeping y fixed and then, y is updated keeping x fixed [1].
- Dual variables are updated and the process is repeated.
- Can trivially be parallelized ([BPC+11]).

[1] similar in spirit to the Jacobi method used to solve diagonally dominant systems of linear equations

ADMM: continued

The same idea can be generalized to $min_w \sum_{i=1}^{m} f_i(w)$ ([BT97]). We rewrite the problem as

$$\begin{aligned} \min_{w_i,\ldots,w_m} \quad & \sum_{i=1}^{m} f_i(w_i) \\ \text{subject to} \quad & w_i = w_{i+1}; i = 1, \ldots, m-1. \end{aligned}$$

where $f_i(.)$ is the objective function at node i.

ADMM: continued

- λ_j represents the Lagrangian dual variable corresponding to the j^{th} constraint

ADMM: continued

- λ_j represents the Lagrangian dual variable corresponding to the j^{th} constraint
- c is the augmented lagrangian parameter

ADMM: continued

- λ_j represents the Lagrangian dual variable corresponding to the j^{th} constraint
- c is the augmented lagrangian parameter
- Apply equation (4.75) of [BT97] to get:

ADMM: continued

- λ_j represents the Lagrangian dual variable corresponding to the j^{th} constraint
- c is the augmented lagrangian parameter
- Apply equation (4.75) of [BT97] to get:
-

$$w_i^{(t+1)} = \underset{w}{\operatorname{argmin}} \{ f_i(w) + c\|w\|^2 + w^T(\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(w_i^{(t)} + \frac{w_{i-1}^{(t)} + w_{i+1}^{(t)}}{2})) \} \quad (1)$$

ADMM: continued

- λ_j represents the Lagrangian dual variable corresponding to the j^{th} constraint
- c is the augmented lagrangian parameter
- Apply equation (4.75) of [BT97] to get:
-

$$w_i^{(t+1)} = \underset{w}{\operatorname{argmin}}\{f_i(w) + c\|w\|^2 + w^T(\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(w_i^{(t)} + \frac{w_{i-1}^{(t)} + w_{i+1}^{(t)}}{2}))\} \quad (1)$$

-

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{c}{2}(w_i^{(t+1)} - w_{i+1}^{(t+1)}) \quad (2)$$

ADMM: continued: Implementation

- w updates happen parallelly

ADMM: continued: Implementation

- w updates happen parallely
- Node i communicates model with neighbours $i-1$ and $i+1$.

A general framework for distributed optimization

- Add a linear or a quadratic correction $C(w)$ to the objective function at each node.

A general framework for distributed optimization

- Add a linear or a quadratic correction $C(w)$ to the objective function at each node.
- IPM adds $\hat{f}_i^{w^{(k)}}(w)$

A general framework for distributed optimization

- Add a linear or a quadratic correction $C(w)$ to the objective function at each node.
- IPM adds $\hat{f}_i^{w^{(k)}}(w)$
- The correction is iteratively improved along with the solution ([MKSB13], [BPC+11], [HMS08]).

A general framework for distributed optimization

- IPM adds $\hat{f}_i^{w^{(k)}}(w)$

A general framework for distributed optimization

- IPM adds $\hat{f}_{-i}^{w_i^{(k)}}(w)$
- ADMM, based on Lagrangian Duality adds

$$C_i^{(t)}(w) = c\|w\|^2 + w^T d_i^{(t)}$$

where $d_i^{(t)} = (\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(w_i^{(t)} + \frac{w_{i-1}^{(t)} + w_{i+1}^{(t)}}{2}))$

A general framework for distributed optimization

- IPM adds $\hat{f}_{-i}^{w_i^{(k)}}(w)$
- ADMM, based on Lagrangian Duality adds

$$C_i^{(t)}(w) = c\|w\|^2 + w^T d_i^{(t)}$$

where $d_i^{(t)} = (\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(w_i^{(t)} + \frac{w_{i-1}^{(t)} + w_{i+1}^{(t)}}{2}))$

- When Fenchel duality is used, we have a linear term ([HMS08], equation (2)) used to tie together solutions from various nodes.

A general framework for distributed optimization

- Observation: Each of the above three methods involves a communication of $\mathcal{O}(n)$ where n is the number of features.

A general framework for distributed optimization

- Observation: Each of the above three methods involves a communication of $\mathcal{O}(n)$ where n is the number of features.
- Can we reduce this further?

A general framework for distributed optimization

Algorithm to capture all variants described above

Algorithm 3 General Algorithm

1: Initialise $w^{(0)}$ and other quantities required arbitrarily
2: **for** $t = 1, 2, ..$ (outer iterations) **do**
3: Compute $C_i^{(t)}(w)$, the correction
4: $w_i^{(t)} = \underset{w}{\text{argmin}}(f_i(w) + C_i^{(t)}(w))$ by some method
5: Communication: communicate the required vectors (such as dual vectors, or gradients)
6: **end for**
7: **return** $w^{(t)} = \texttt{ParameterMixing}(w_i^t)$

Approach

- To reduce communication costs further.

Approach

- To reduce communication costs further.
- Original Problem:

$$\min_{w_i,\ldots,w_m} \quad \sum_{i=1}^{m} f_i(w_i)$$

$$\text{subject to} \quad w_i = w_{i+1}; i = 1, \ldots, m-1.$$

Approach

- To reduce communication costs further.
- Original Problem:

$$\min_{w_i,\ldots,w_m} \quad \sum_{i=1}^{m} f_i(w_i)$$
$$\text{subject to} \quad w_i = w_{i+1}; i = 1, \ldots, m-1.$$

- Relax Constraints $w_i = w_{i+1}$

Approach

- Constraint $w_i = w_{i+1}$ enforces component-wise equality $w_i[j] = w_{i+1}[j]$.

Approach

- Constraint $w_i = w_{i+1}$ enforces component-wise equality $w_i[j] = w_{i+1}[j]$.
- Relax to sum of elements over a set of indices being equal.

Approach

- Constraint $w_i = w_{i+1}$ enforces component-wise equality $w_i[j] = w_{i+1}[j]$.
- Relax to sum of elements over a set of indices being equal.
- Divide features into sets $I_1, I_2, ..., I_k$ (not necessarily disjoint).

Approach

- Constraint $w_i = w_{i+1}$ enforces component-wise equality $w_i[j] = w_{i+1}[j]$.
- Relax to sum of elements over a set of indices being equal.
- Divide features into sets $I_1, I_2, ..., I_k$ (not necessarily disjoint).
- Enforce: $\sum_{j \in I_r} w_i[j] = \sum_{j \in I_r} w_{i+1}[j]$ for each set I_r.

Approach

- Represent all k scalar equations as a vector equation $S^T w_i = S^T w_{i+1}$ [2].

[2] $S \in \mathbb{R}^{n \times k}$ is a matrix of 0s and 1s such that $S_{ij} = 1 \Leftrightarrow i \in I_j$

Approach

- Represent all k scalar equations as a vector equation $S^T w_i = S^T w_{i+1}$ [2].
- New problem:

$$\min_{w_i,\ldots,w_m} \quad \sum_{i=1}^{m} f_i(w_i)$$
$$\text{subject to} \quad S^T w_i = S^T w_{i+1}; i = 1, \ldots, m-1.$$

[2] $S \in \mathbb{R}^{n \times k}$ is a matrix of 0s and 1s such that $S_{ij} = 1 \Leftrightarrow i \in I_j$

Approach

- Represent all k scalar equations as a vector equation $S^T w_i = S^T w_{i+1}$ [2].
- New problem:

$$\begin{aligned} \min_{w_i,\ldots,w_m} \quad & \sum_{i=1}^{m} f_i(w_i) \\ \text{subject to} \quad & S^T w_i = S^T w_{i+1}; i = 1, \ldots, m-1. \end{aligned}$$

- Approximation of original problem: need not be solved by ADMM.

[2] $S \in \mathbb{R}^{n \times k}$ is a matrix of 0s and 1s such that $S_{ij} = 1 \Leftrightarrow i \in I_j$

Approach

- Apply equation (4.75) of [BT97] to get:

$$w_i^{(t+1)} = \underset{w}{\operatorname{argmin}}\{f_i(w) + cw^T(SS^T)w + w^T d_i^{(t)}\} \quad (3)$$

Approach

- Apply equation (4.75) of [BT97] to get:

$$w_i^{(t+1)} = \underset{w}{\text{argmin}}\{f_i(w) + cw^T(SS^T)w + w^T d_i^{(t)}\} \quad (3)$$

- where

$$d_i^{(t)} = S(\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(S^T w_i^{(t)} + \frac{S^T w_{i-1}^{(t)} + S^T w_{i+1}^{(t)}}{2}))$$

Approach

- Apply equation (4.75) of [BT97] to get:

$$w_i^{(t+1)} = \underset{w}{\operatorname{argmin}}\{f_i(w) + cw^T(SS^T)w + w^T d_i^{(t)}\} \quad (3)$$

- where

$$d_i^{(t)} = S(\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(S^T w_i^{(t)} + \frac{S^T w_{i-1}^{(t)} + S^T w_{i+1}^{(t)}}{2}))$$

- Dual Update:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{c}{2}(S^T w_i^{(t+1)} - S^T w_{i+1}^{(t+1)}) \quad (4)$$

Approach

- Apply equation (4.75) of [BT97] to get:

$$w_i^{(t+1)} = \underset{w}{\operatorname{argmin}}\{f_i(w) + cw^T(SS^T)w + w^T d_i^{(t)}\} \quad (3)$$

- where

$$d_i^{(t)} = S(\lambda_i^{(t)} - \lambda_{i-1}^{(t)} - c(S^T w_i^{(t)} + \frac{S^T w_{i-1}^{(t)} + S^T w_{i+1}^{(t)}}{2}))$$

- Dual Update:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{c}{2}(S^T w_i^{(t+1)} - S^T w_{i+1}^{(t+1)}) \quad (4)$$

- Communication Cost

Algorithm

Algorithm 4 ADMM with reduced communication

1: Initialise $w^{(0)}, w_i^{(0)}, \lambda_i = \mathbf{0}_k = \lambda_{i-1}$
2: **for** $t = 0, 1, 2, ..$ **do**
3: Solve for $w_i^{(t+1)}$ as per equation 3
4: Communicate models with neighbours and obtain $w_{i-1}^{(t+1)}$ and $w_{i+1}^{(t+1)}$
5: Update λ_i and λ_{i-1} by equation 4
6: **end for**
7: **return** $w_i^{(t+1)}$ or `ParameterMixing`(w_i^{t+1})

Approach

- Key Observation: Parameter Mixing step in the end can be skipped

Approach

- Key Observation: Parameter Mixing step in the end can be skipped
- Can reduce communication cost to be smaller than Simple Parameter Mixing

Results: Synthetic Datasets

- Synthetic datasets with 4-10 features
- 200-4000 training and testing examples
- $S \in \mathbb{R}^{n \times 3}$
- Solved with `cvx` as black-box solver
- Stopping based on duality gap

Results: Synthetic Datasets

Table : Performance with Synthetic Dataset `TestFinal4.mat` with 4 nodes

Method	**Objective value**	**Test Accuracy**
Full problem	0.4891	0.8000
PM	0.6009	0.7350
Our method	0.5128	0.7900

Results: low k values work

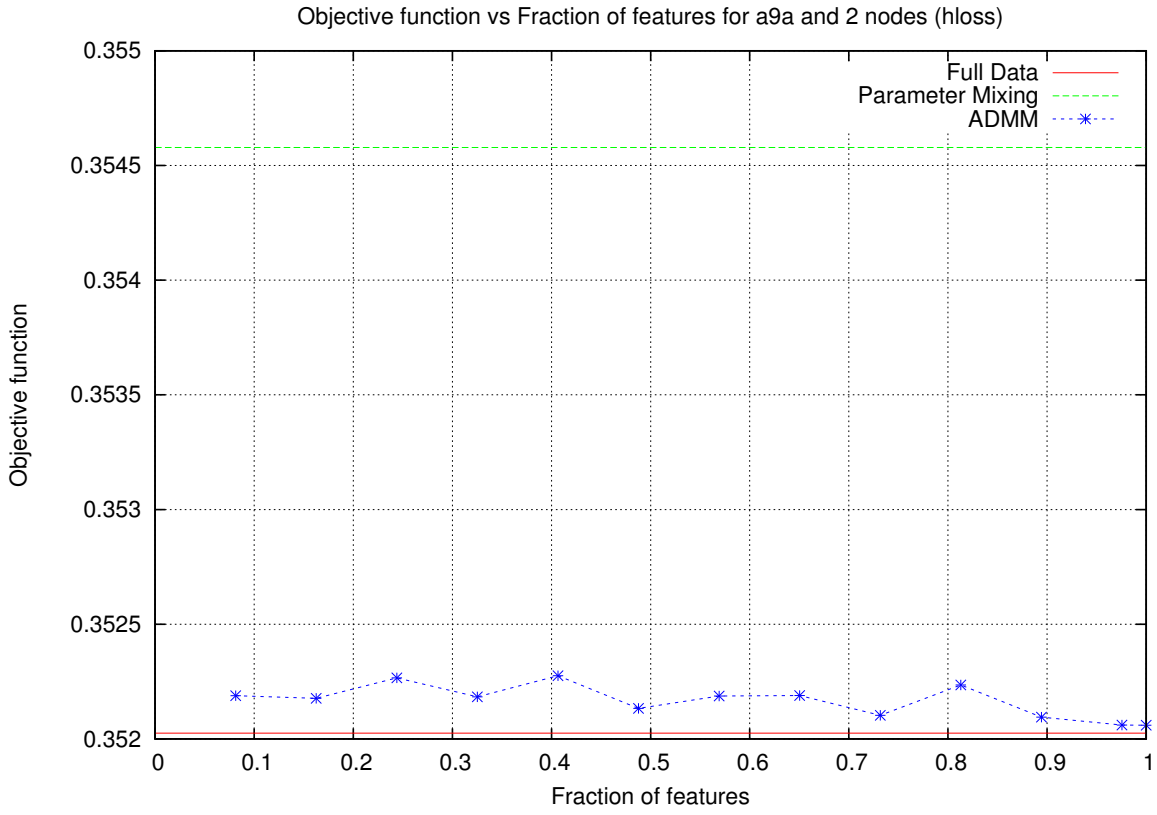

Figure : Dataset a9a split into two nodes: objective function value

Results: low k values work

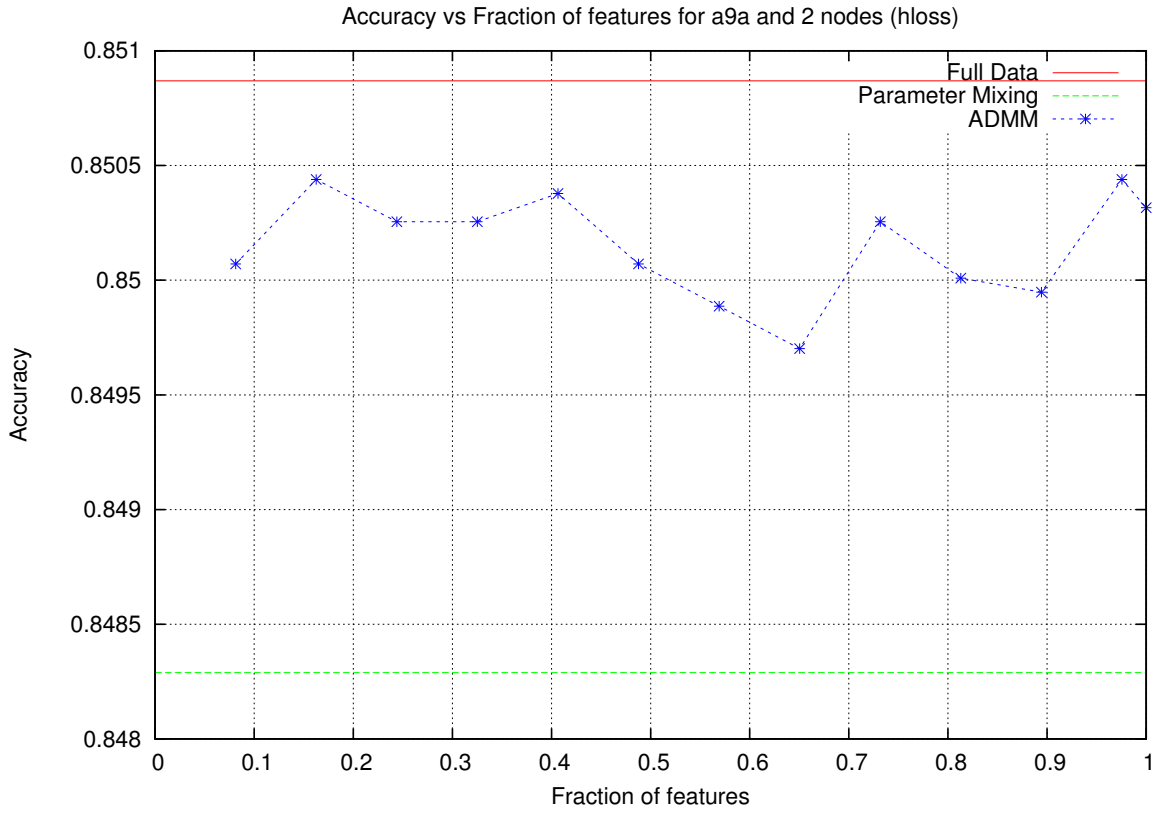

Figure : Dataset a9a split into two nodes: test accuracy

Results: Larger k means a better approximation

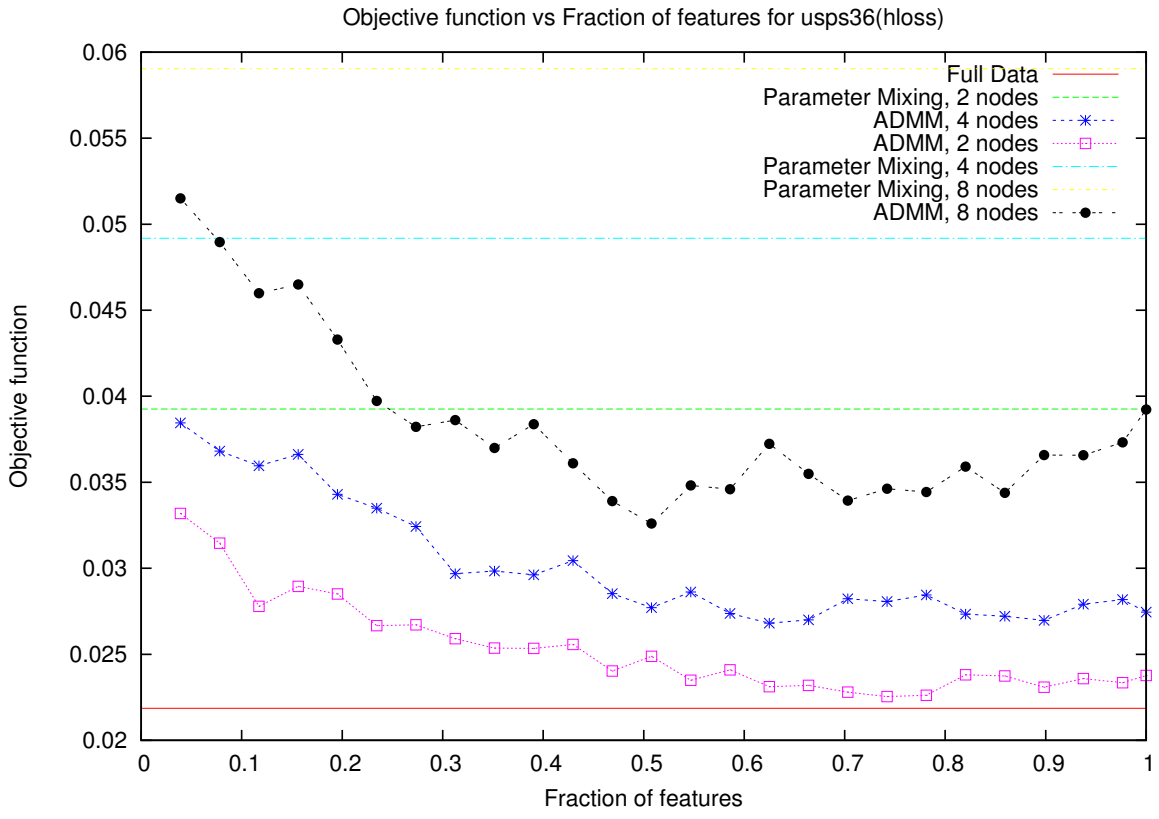

Figure : Dataset usps36: 2,4 and 8 nodes

Results: 25-100 Nodes

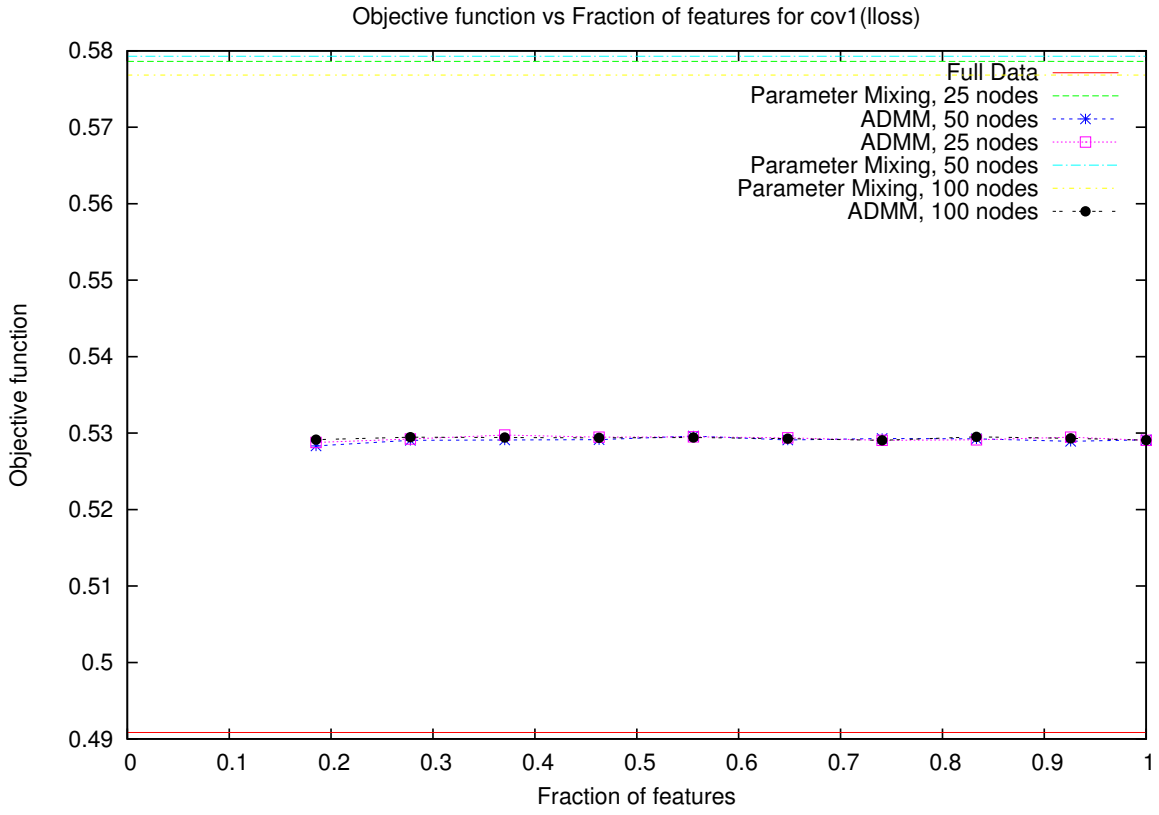

Figure : Dataset cov: 25, 50 and 100 nodes: objective value

Results: 25-100 Nodes

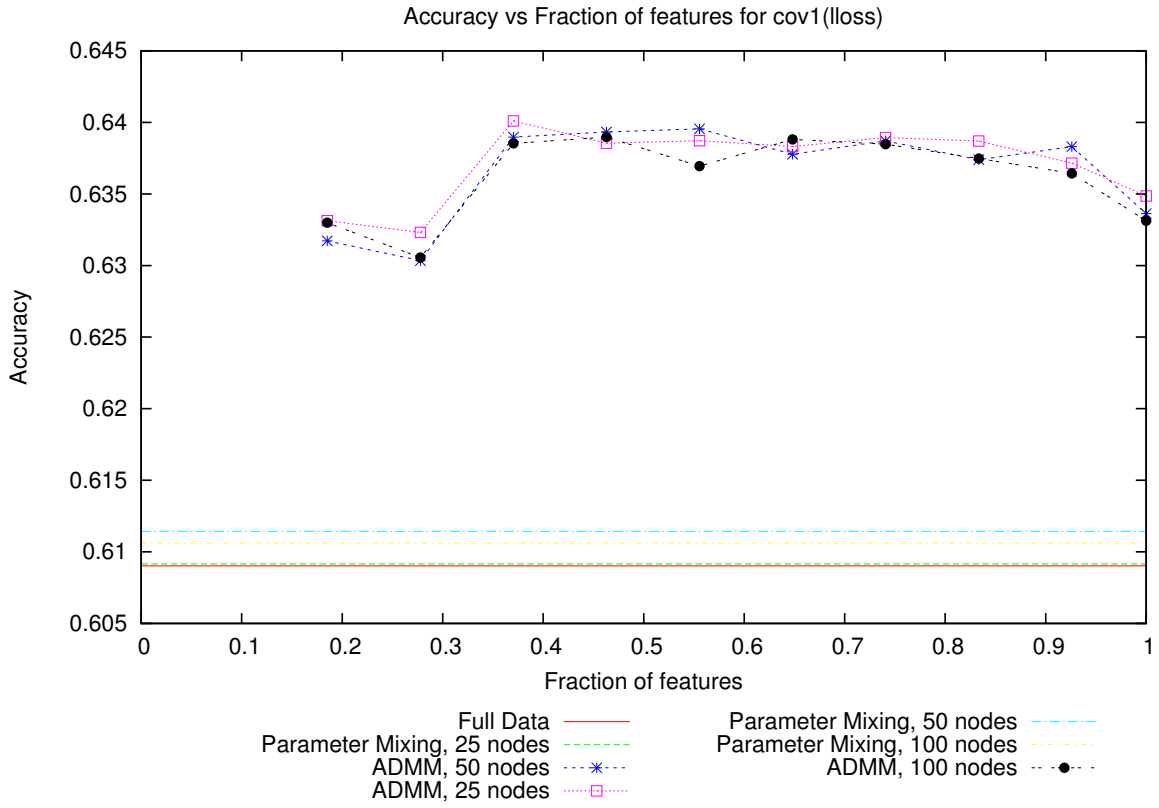

Figure : Dataset cov: 25, 50 and 100 nodes: Test Accuracy

Results: Do we need a PM step in the end?

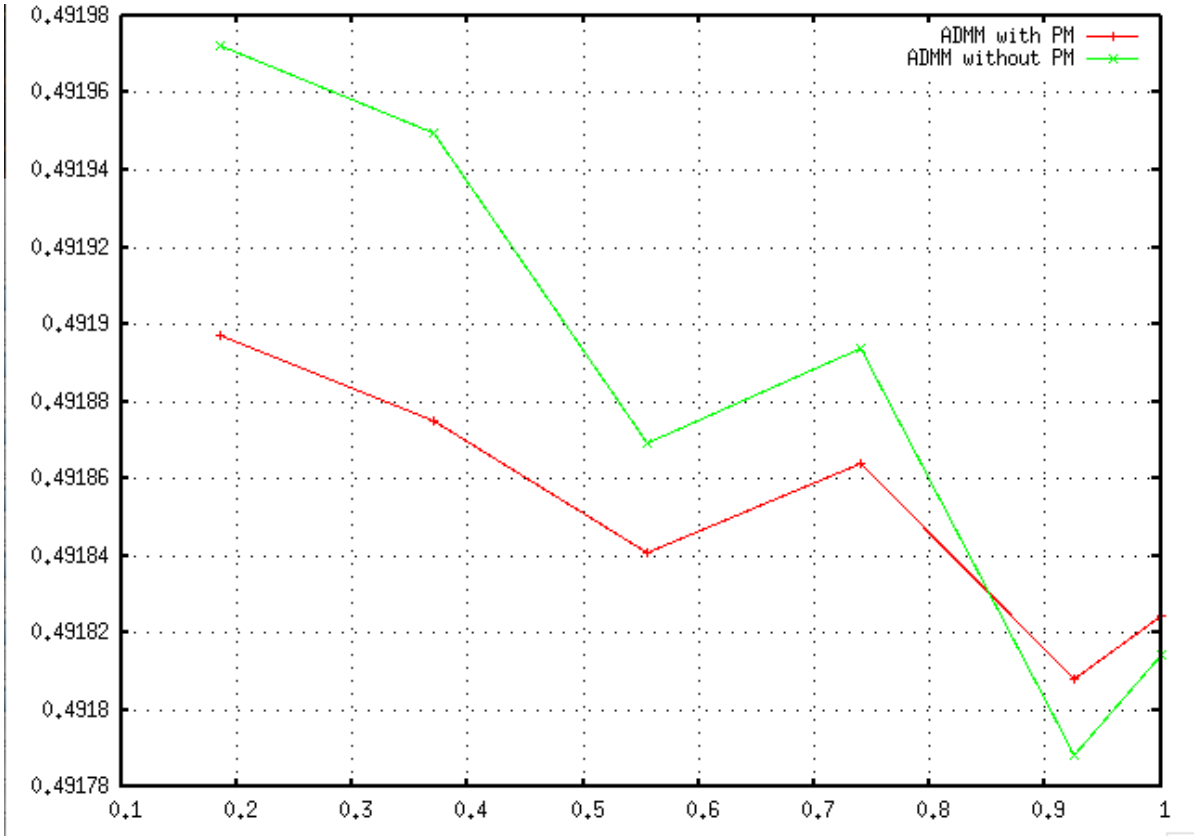

Figure : cov: 4 nodes: Comparision of objective function values ADMM with PM and ADMM without PM

Results: Do we need a PM step in the end?

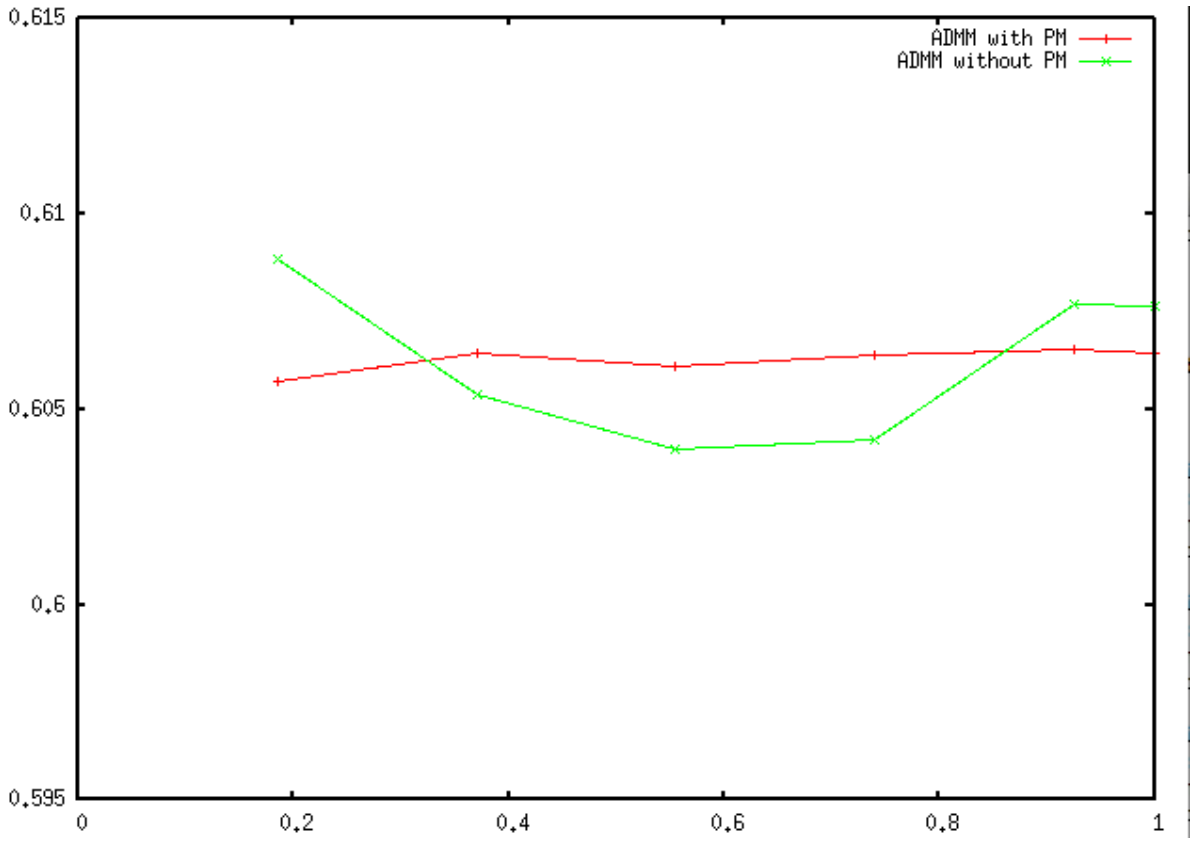

Figure : cov: 4 nodes: Comparision of test accuracy values ADMM with PM and ADMM without PM

Conclusion

- There is some merit in this method

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment
- TODO: Run experiments on Hadoop with larger datasets

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment
- TODO: Run experiments on Hadoop with larger datasets
- TODO: Grouping of features

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment
- TODO: Run experiments on Hadoop with larger datasets
- TODO: Grouping of features
  - Group similar features together

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment
- TODO: Run experiments on Hadoop with larger datasets
- TODO: Grouping of features
  - Group similar features together
  - Clustering of features

Conclusion

- There is some merit in this method
- TODO: Theoretical Treatment
- TODO: Run experiments on Hadoop with larger datasets
- TODO: Grouping of features
  - Group similar features together
  - Clustering of features
  - Sampling of data

References I

[ACDL11] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford, *A reliable effective terascale linear learning system*, CoRR **abs/1110.4198** (2011).

[BPC+11] C S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, 2011.

[BT97] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*, Athena scientific optimization and computation series, Athena Scientific, 1997.

References II

[HMS08] Tamir Hazan, Amit Man, and Amnon Shashua, *A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality*.

[Man] O. L. Mangasarian, *Parallel gradient distribution*.

[MHM] Ryan Mcdonald, Keith Hall, and Gideon Mann, *Distributed training strategies for the structured perceptron*.

[MKSB13] Dhruv Mahajan, S. Sathiya Keerthi, S. Sundararajan, and Léon Bottou, *A functional approximation based distributed learning algorithm*, CoRR **abs/1310.8418** (2013).

References III

[MMM+09] Gideon Mann, Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, and Daniel D. Walker, *Efficient large-scale distributed training of conditional maximum entropy models*, In Advances in Neural Information Processing Systems, 2009.

The End. Thank You!