

Interior Point Methods for Support Vector Machines

Jon Allen

*Center for Relativity, University of Texas at Austin
Austin, TX 78712-1081 USA**

Rehmi Post

*Center for Bits and Atoms, MIT Media Lab
Cambridge, MA 02139 USA†*

Andy Maloney

*College of Pharmacy, University of Texas at Austin
Austin, TX 78712-0120 USA‡*

Kevin Safford§

Detailed notes on an interior point method implementation for distributed L1 linear regression in the MapReduce framework, a rosetta stone for most popular notations discussing support vector machines and interior point methods, and detailed notes for a multivariate least squares linear regression MapReduce implementation.

I. INTRODUCTION

Support Vector Machines is the name given to a class of machine learning methods used for classification and regression which use convex optimization techniques to find a subset of training data to establish a maximally separated foliation of feature space. The convex optimization problem can be cast in the form of a constrained quadratic Lagrangian, constructed from the input data, reminiscent of rigid body dynamics. At criticality, when the Lagrangian is optimized, the problem can be expressed completely in terms of a limited set of the input data which, intuitively, correspond to the supporting forces keeping a rigid body at equilibrium. This subset of input data are known as Support Vectors and define the optimal solution, whence the name Support Vector Machines. SVMs are of interest mathematically because they can be solved uniquely, in contrast to many other problems which are non-convex, and of great interest to the machine learning community due to their outstanding accuracy, especially when dealing with discrete data such as text and genetic sequences.

As will be detailed below, the construction of SVMs requires solving a global optimization problem. We implement a parallel version of an Interior Point Method for quadratic programs expressed as constrained Lagrangians. Details of our IPM implementations are provided below.

A. Notation

The feature space will be considered to be A dimensional, with indices labeled by the Roman a, b, c belonging to $\{1, \dots, A\}$. The weight vectors will be long to an $A + 1$ dimensional space and will have indices labeled by the Latin α, β belonging to $\{0, \dots, A\}$. The input observation vectors considered will be taken to be a D dimensional space, $\mathbf{x} \in \mathbb{R}^D$, having one dimensional targets, y , belonging to some field, which will be \mathbb{R} for regression and the cyclic group of order two $\mathbb{Z}_2 = \{-1, 0, 1\}$ for binary classification. Since classification is binary, data projecting to the kernel are dropped, leaving only $\mathbb{Z}_2 \setminus \{0\} = \{-1, 1\}$ as target values.

The quadratic dual-dual Lagrangian problem will be defined for n training vectors, m inequality constraints and l equality constraints, all belonging to an N dimensional space with tensoral indices labeled by the Roman lowercase

*Electronic address: jon@slurpee.org

†Electronic address: rehmi@media.mit.edu

‡Electronic address: schrodiscat@gmail.com

§Electronic address: ktsafford@gmail.com

i, j, k belonging to $\{1, \dots, N\}$. Einstein notation will be used throughout, meaning repeated dual indices will be summed, $\alpha^i c_i \equiv \sum_{i=1}^n \alpha^i c_i$, unless otherwise noted. The space of the summed indices will be clear from context.

Only inner products will imply summation; simple multiplication of tensor elements by coefficients having the same index label will not imply summation, $y_i \mathbf{x}_i \neq \sum_{i=1}^n y_i \mathbf{x}_i$, since this makes no sense. To avoid cluttering equations with unnecessary indices, vectors and dual one will be written in bold face, with dimensionality and index summation clear from context.

Throughout, for all spaces, assume the metric to be the standard Euclidian metric. This means that all indexes, $i, j, k \in \{1, \dots, N\}$, will be raised and lowered by the metric $\mathbf{I} = \delta_{ij} = \mathbf{I}^{-1} = \delta^{ij}$, and likewise for all other indexes, $a, b, c \in \{1, \dots, A\}$ and $\alpha, \beta, \gamma \in \{0, \dots, A\}$.

II. SUPPORT VECTOR MACHINES

Input data belongs to the space of observations, $\mathcal{X} \subset \mathbb{R}^a$, and targets, $\mathcal{Y} \subset \mathbb{R}$. The feature space is defined as a map from the space of observations into some Hilbert space, \mathcal{H} , defined by the bilinear semi positive definite inner product kernel, $K(\cdot, \cdot)$, through

$$\Phi: \mathcal{X} \rightarrow \mathcal{H} \quad (1)$$

$$\mathbf{x} \rightarrow \Phi(\mathbf{x}) \quad (2)$$

$$K: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \quad (3)$$

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &\equiv \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') \\ &= \Phi(\mathbf{x}') \cdot \Phi(\mathbf{x}) = K(\mathbf{x}', \mathbf{x}) \end{aligned} \quad (4)$$

For binary classification, $y_i \in \mathbb{Z}_2 \setminus \{0\}$, and the Lagrangian to be minimized reads

$$L(\mathbf{w}, b, \alpha) \equiv \frac{1}{2} \mathbf{w} \cdot \mathbf{w} - \alpha^i [c_i(\mathbf{x}, \mathbf{w}, b)] \quad (5)$$

for constraints

$$c_i \equiv y_i (w^A \delta_{AB} \Phi(\mathbf{x}_i)^B + b) - 1 \geq 0 \quad (6)$$

and Lagrange multipliers, α^i , satisfying

$$\alpha^i \leq 0 \quad (7)$$

The minimization problem is to be solved for the $A + 1$ canonical, often called "primal", variables \mathbf{w}, b and the n Lagrange multipliers, $\alpha^i \in \mathbb{R}_{(+)}^n$. At optimality, the Lagrangian is minimized with respect to the canonical variables, and hence satisfies

$$\frac{\partial L}{\partial b} = \alpha^i y_i = 0 \quad (8)$$

$$\frac{\partial L}{\partial w^a} = w^a - \alpha^i [y_i \Phi(\mathbf{x}_i)^a] = 0 \quad (9)$$

while being maximized with respect to the Lagrange multipliers. By definition, the space of all possible solutions is defined by the constraints

$$\frac{\partial L}{\partial \alpha^i} = -c_i = -[y_i (\Phi(\mathbf{x}_i) \cdot \mathbf{w} + b) - 1] \leq 0 \quad (10)$$

hence, to maximize $L(\mathbf{w}, b, \alpha)$ with respect to the Lagrange multipliers, α^i , at criticality when the Lagrangian is optimized, the Lagrange multipliers, $\alpha^i \geq 0$ and constraints $c_i \geq 0$ must satisfy (no sum)

$$\alpha_i c_i = 0 \quad (11)$$

for each constraint c_i , otherwise the Lagrangian could be further minimized by varying the canonical variables, \mathbf{w}, b , or maximized by varying the Lagrange multipliers, α . This is ensured by the convex nature of the optimization problem and is the crux of what makes SVMs interesting. The support vectors are defined by the constraints $c_i \geq 0$ satisfying $c_i = 0$, which, through equation (11), are identified with the non-zero Lagrange multipliers.

We want to be able to find a solution to this problem in a minimum amount of computational time, so we reduce the number of variables considered and cast the Lagrangian as a quadratic problem by solving equations (8) and (9) inside the Lagrangian, equation (5), yielding a Lagrangian expressed completely in terms of the Lagrange multipliers, α^i , often known as the "dual Lagrangian". The problem becomes, minimize, with respect to components of the vector $\alpha \in \mathbb{R}_{(+)}^N$, the quadratic Lagrangian

$$L(\alpha) = -\alpha^i \delta_{ij} \gamma^j + \frac{1}{2} \alpha^i \alpha^j [y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)] \quad (12)$$

for some fixed vector $\gamma \in \mathbb{R}^N$ satisfying $\gamma^j > 0$, subject to the constraints

$$\alpha^i \geq 0 \quad (13)$$

$$\alpha^i y_i = 0 \quad (14)$$

Notice that the equality constraint, $\alpha^i y_i = 0$, can not be solved uniquely in this form, since any rotation of the vector $\alpha \in \mathbb{R}_{(+)}^N$ about the fixed vector of target labels $\mathbf{y} \in \mathbb{R}^N$ will also provide a solution, hence there will be infinitely many solutions.

Optimization problems of the above form can be solved in a few different ways. Here we focus on one method in particular, the Interior Point Method (IPM), which will allow us to implement a parallel solver. The IPM solver will be applicable to both binary classification and regression.

III. INTERIOR POINT METHOD

The discussion of Interior Point Method solvers given here will be general, with the following section detailing parallel implementations for Parallel SVM (PSVM) and Linear Regression. In order to keep notation in line with texts on Interior Point Methods, variable names have been adopted to coincide with the most popular texts. Unfortunately, there is some overlap in notation with the variables defined previously in section (II). We will attempt to minimize this overlap, but the reader should take this section to be self-contained, dealing with IPMs in general. We will return to specific implementations dealing with SVMs in later sections.

The inequality constraints are defined as

$$-\alpha^i \leq 0 \quad (15)$$

$$-t^i \equiv \alpha^i - u^i \leq 0 \quad (16)$$

The KKT gap is defined by the lower bound constraints

$$\alpha^i z^i = \mu \geq 0 \quad (17)$$

and upper bound constraints

$$s^i t^i = \mu \geq 0 \quad (18)$$

for $\{\alpha^i, t^j, s^k, z^l\} \geq 0$ for all $i, j, k, l \in \{1, \dots, n\}$. Together, the KKT gap can be written

$$\mu \equiv \frac{1}{2n} (\alpha^i \delta_{ij} z^j + s^i \delta_{ij} t^j) \quad (19)$$

The equality constraints are

$$\mathbf{A}\alpha - \mathbf{d} = 0 \quad (20)$$

$$\vec{u} - \vec{\alpha} - \vec{t} = 0 \quad (21)$$

Varying this QP problem yields a linearized problem which is to be solved using an IPM.

Linearizing the dual-dual problem and varying the system yields implicit update equations for α and h and direct updates for all other variables. Start with the definitions

$$\rho_{p_1} \equiv \mathbf{A}\alpha - \mathbf{d} \quad (22)$$

$$\rho_{p_2} \equiv \vec{u} - \vec{\alpha} - \vec{t} \quad (23)$$

$$\rho_d \equiv -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h - \vec{s} + \vec{z} \quad (24)$$

$$\rho_{KKT_1} \equiv \mu \frac{1}{\alpha^i} - z_i - \frac{\Delta \alpha^i}{\alpha^i} \Delta z^i \quad (25)$$

$$\rho_{KKT_2} \equiv \mu \frac{1}{t^i} - s_i - \frac{\Delta t^i}{t^i} \Delta s^i \quad (26)$$

The linearized equations to be solved for the variations are given by

$$\mathbf{A}\Delta\alpha = \mathbf{A}\alpha - \mathbf{d} \quad (27)$$

$$\Delta\alpha - \Delta t = \vec{u} - \vec{\alpha} - \vec{t} \quad (28)$$

$$\mathbf{Q}\Delta\alpha - \mathbf{A}^T\Delta h + \Delta\vec{s} - \Delta\vec{z} = -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h - \vec{s} + \vec{z} \quad (29)$$

$$z^i \frac{\Delta\alpha^i}{\alpha^i} + \Delta z^i = \mu \frac{1}{\alpha^i} - z_i - \frac{\Delta\alpha^i}{\alpha^i} \Delta z^i \quad (30)$$

$$s^i \frac{\Delta t^i}{t^i} + \Delta s^i = \mu \frac{1}{t^i} - s_i - \frac{\Delta t^i}{t^i} \Delta s^i \quad (31)$$

which are used to solve for Δt^i , Δs^i and Δz^i directly during the predictor phase. The updated corrector vector constraint residual is given by

$$\begin{aligned} \rho^i \mathbf{e}_i &\equiv \rho_d + \rho_{KKT_1} - \rho_{KKT_2} \\ &= -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h + \mu \left(\frac{1}{\alpha} - \frac{1}{t} \right) - \frac{\Delta\alpha^i}{\alpha^i} \Delta z^i + \frac{\Delta t^i}{t^i} \Delta s^i \end{aligned} \quad (32)$$

Inserting in the variations for \vec{t} , \vec{s} and \vec{z} , the equations to be solved reduce to the form

$$\begin{bmatrix} (\mathbf{Q} + \mathbf{D}) & -\mathbf{A}^T \\ -\mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta h \end{bmatrix} = \begin{bmatrix} \rho_d + \rho_{KKT_1} - \rho_{KKT_2} \\ \rho_{p_1} \end{bmatrix} \quad (33)$$

for diagonal matrix \mathbf{D} given by the non-zero elements (no sum)

$$D_{ii} \equiv \left(\frac{z^i}{\alpha^i} + \frac{s^i}{t^i} \right) \quad (34)$$

In any quadratic problem with inequality constraints, the diagonal component will be comprised of the sum of all pairs of inequality constraints taking the form

$$D \equiv \sum_{\text{Bounds}} \frac{(\text{Lagrange multiplier of inequality boundary constraint})}{(\text{Bounded variable which will be non-zero in the feasible region})} \quad (35)$$

At each step, D is updated and the resulting system solved for the variations of the variables α and h and updates to inequality constraint Lagrange multipliers subject to the imposed constraints (no sum) $s^i t^i = \alpha^i z^i = \mu \geq 0$ and $t^i \equiv u^i - \alpha^i$.

The step size for each iteration of the IPM denoted λ , is given by

$$\frac{1}{\lambda} = \max \left[1, (\epsilon - 1)^{-1} \min \left(\frac{\Delta\alpha^i}{\alpha^i} \right), (\epsilon - 1)^{-1} \min \left(\frac{\Delta t^i}{t^i} \right), (\epsilon - 1)^{-1} \min \left(\frac{\Delta s^i}{s^i} \right), (\epsilon - 1)^{-1} \min \left(\frac{\Delta z^i}{z^i} \right) \right] \quad (36)$$

for some fixed $\epsilon > 0 \in \mathbb{R}$. The step size λ is chosen so as to ensure that the updated variables defined by solving the IPM for step length λ will not cause D or D^{-1} to blow up at the next iteration.

A. Binary Classification

Below details the implementation described above for a binary classifier.

In this formulation, the matrix \mathbf{Q} is given by (no sum)

$$Q_{ij} \equiv y_i y_j K(\vec{x}_i, \vec{x}_j) \quad (37)$$

for some symmetric positive semi-definite kernel $K(\cdot, \cdot)$ and training data $(\vec{x}_i, y_i) \in \mathcal{X}$ having labels $y_i \in \{-1, 1\}$. The constant vector \mathbf{c} is given by the vector of all unit values

$$c^i = 1 \quad (38)$$

for all $i \in 1, \dots, N$. The vector \mathbf{u} is given by

$$\mathbf{u} = C\mathbf{c} \quad (39)$$

for a specified constant $C \in \mathbb{R}$. The constraint matrix, \mathbf{A} , is given by

$$A_i = y_i \quad (40)$$

and right hand side source matrix, \mathbf{d} , is given by

$$d = 0 \quad (41)$$

B. Standard Regression

Below details the implementation described above for regression.

In this formulation, the matrix \mathbf{Q} is given by (no sum)

$$Q_{ij} \equiv K(\vec{x}_i, \vec{x}_j) \quad (42)$$

for some symmetric positive semi-definite kernel $K(\cdot, \cdot)$ and training data $(\vec{x}_i, y_i) \in \mathcal{X}$ having labels $y_i \in \{-1, 1\}$. The constant vector \mathbf{c} is given by the vector of all unit values

$$\begin{aligned} c^i &= y_i - \epsilon \\ c^{i+N} &= -y_i - \epsilon \end{aligned} \quad (43)$$

for all $i \in 1, \dots, N$ and some fixed $\epsilon \geq 0 \in \mathbb{R}$ defining the tube width for the region of ineffective input values. The vector $\mathbf{u} \in \mathcal{R}^{2N}$ is defined by constant values

$$u^j = C \quad (44)$$

for all $j \in 1, \dots, 2N$ given a supplied constant $C \geq 0 \in \mathbb{R}$. The constraint matrix, \mathbf{A} , is given by

$$\begin{aligned} A_i &= 1 \\ A_{i+N} &= -1 \end{aligned} \quad (45)$$

and right hand side source matrix, \mathbf{d} , is given by

$$d = 0 \quad (46)$$

IV. TRANSLATIONS OF NOTATION FROM VARIOUS RESEARCH SOURCES

Translation between references. The feature space will be considered to be A dimensional, with the quadratic dual-dual problem defined for n training vectors yielding $m = 2n$ inequality constraints to be satisfied and 2 equality constraints.

The variables will be labeled with subscripts for each of the 3 references, Schölkopf (Chapter 10) variables will be denoted v_S , Chang et. al, will be denoted v_P and Woodsend's variables will be denoted v_W . The translation for fixed constants are

$$C \equiv C_{(S)} = C_{(P)} = \tau_{(W)} \quad (47)$$

$$b \equiv b_{(S)} = b_{(P)} = b_{(W)} \quad (48)$$

The translations for fixed vector quantities are

$$u^i \mathbf{e}_i \equiv \vec{u}_{(S)} = \vec{C} \mathbf{e}_{(S)} = \vec{C} \mathbf{e}_{(P)} = \vec{u}_{(W)} \quad (49)$$

$$c^i \mathbf{e}_i \equiv \vec{c}_{(S)} = -\vec{\mathbf{e}}_{(P)} = \vec{c}_{(W)} \quad (50)$$

$$(51)$$

The translations for the fixed matrix \mathbf{A} of dimension $n \times l$ or $n \times 2l$, fixed equality constraint vector \mathbf{d} of dimension l or $2l$ and fixed inner product \mathbf{Q} of dimension $n \times n$ or $2n \times 2n$, the quantities are

$$\mathbf{A} \equiv \mathbf{A}_{(S)} = \vec{y}_{(P)} = \mathbf{A}_{(W)} \quad (52)$$

$$\mathbf{d} \equiv \mathbf{d}_{(S)} = \mathbf{0}_{(P)} = \mathbf{b}_{(W)} \quad (53)$$

$$\mathbf{Q} \equiv \mathbf{Q}_{(S)} = \mathbf{Q}_{(P)} = \mathbf{Q}_{(W)} \quad (54)$$

for all problems considered here $l = 1$.

The translations for variable vector quantities are

$$\alpha^i \mathbf{e}_i \equiv \vec{\alpha}_{(S)} = \vec{\alpha}_{(P)} = \vec{x}_{(P)} = \vec{z}_{(W)} \quad (55)$$

$$s^i \mathbf{e}_i \equiv \vec{s}_{(S)} = \vec{\lambda}_{(P)} = \vec{v}_{(W)} \quad (56)$$

$$z^i \mathbf{e}_i \equiv \vec{z}_{(S)} = \vec{\xi}_{(P)} = \vec{s}_{(W)} \quad (57)$$

$$t^i \mathbf{e}_i \equiv \vec{t}_{(S)} = (C\mathbf{e}^T - \alpha)_{(P)} = (u^T - z)_{(W)} \quad (58)$$

The translations for scalar variables are

$$h \equiv h_{(S)} = \nu_{(P)} = \lambda_{(W)} \quad (59)$$

The translations for the KKT gap scalar quantities are

$$\mu \equiv \mu_{(S)} = \eta_{(P)} = \mu_{(W)} \quad (60)$$

with μ defined by the duality KKT gap, defined below.

The inequality constraints are defined as

$$-\alpha^i \leq 0 \quad (61)$$

$$-t^i \equiv \alpha^i - u^i \leq 0 \quad (62)$$

The KKT gap is defined by the lower bound constraints

$$\alpha^i z^i = \mu \geq 0 \quad (63)$$

and upper bound constraints

$$s^i t^i = \mu \geq 0 \quad (64)$$

for $\{\alpha^i, t^j, s^k, z^l\} \geq 0$ for all $i, j, k, l \in \{1, \dots, n\}$. Together, the KKT gap can be written

$$\mu \equiv \frac{1}{2n} (\alpha^i \delta_{ij} z^j + s^i \delta_{ij} t^j) \quad (65)$$

The equality constraints are

$$\mathbf{A}\alpha - \mathbf{d} = 0 \quad (66)$$

$$\vec{u} - \vec{\alpha} - \vec{t} = 0 \quad (67)$$

Varying this QP problem yields a linearized problem which is to be solved using an IPM.

Linearizing the dual-dual problem and varying the system yields implicit update equations for α and h and direct updates for all other variables. Start with the definitions

$$\rho_{p_1} \equiv \mathbf{A}\alpha - \mathbf{d} = r_b^{(W)} \quad (68)$$

$$\rho_{p_2} \equiv \vec{u} - \vec{\alpha} - \vec{t} \quad (69)$$

$$\rho_d \equiv -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h - \vec{s} + \vec{z} \quad (70)$$

$$\rho_{KKT_1} \equiv \mu \frac{1}{\alpha^i} - z_i - \frac{\Delta \alpha^i}{\alpha^i} \Delta z^i \quad (71)$$

$$\rho_{KKT_2} \equiv \mu \frac{1}{t^i} - s_i - \frac{\Delta t^i}{t^i} \Delta s^i \quad (72)$$

$$(73)$$

The linearized equations to be solved for the variations are given by

$$\mathbf{A}\Delta\alpha = \mathbf{A}\alpha - \mathbf{d} \quad (74)$$

$$\Delta\alpha - \Delta t = \vec{u} - \vec{\alpha} - \vec{t} \quad (75)$$

$$\mathbf{Q}\Delta\alpha - \mathbf{A}^T \Delta h + \Delta\vec{s} - \Delta\vec{z} = -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h - \vec{s} + \vec{z} \quad (76)$$

$$z^i \frac{\Delta \alpha^i}{\alpha^i} + \Delta z^i = \mu \frac{1}{\alpha^i} - z_i - \frac{\Delta \alpha^i}{\alpha^i} \Delta z^i \quad (77)$$

$$s^i \frac{\Delta t^i}{t^i} + \Delta s^i = \mu \frac{1}{t^i} - s_i - \frac{\Delta t^i}{t^i} \Delta s^i \quad (78)$$

$$(79)$$

which are used to solve for Δt^i , Δs^i and Δz^i directly during the predictor phase. The updated corrector vector constraint residual is given by

$$\begin{aligned} \rho^i \mathbf{e}_i &\equiv \rho_d + \rho_{KKT_1} - \rho_{KKT_2} \\ &= -\mathbf{Q}\alpha - \vec{c} + \mathbf{A}^T h + \mu \left(\frac{1}{\alpha} \vec{1} - \frac{1}{t} \right) - \frac{\Delta \alpha^i}{\alpha^i} \Delta z^i + \frac{\Delta t^i}{t^i} \Delta s^i \end{aligned} \quad (80)$$

Inserting in the variations for \vec{t} , \vec{s} and \vec{z} , the equations to be solved reduce to the form

$$\begin{bmatrix} (\mathbf{Q} + \mathbf{D}) & -\mathbf{A}^T \\ -\mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta \alpha \\ \Delta h \end{bmatrix} = \begin{bmatrix} \rho_d + \rho_{KKT_1} - \rho_{KKT_2} \\ \rho_{p_1} \end{bmatrix} \quad (81)$$

for diagonal matrix \mathbf{D} given by the non-zero elements

$$D_{ii} \equiv \left(\frac{z^i}{\alpha^i} + \frac{s^i}{t^i} \right) \quad (82)$$

and \mathbf{Q} given by

$$Q_{ij} \equiv y_i y_j K(\vec{x}_i, \vec{x}_j) \quad (83)$$

for some symmetric positive semi-definite kernel $K(\cdot, \cdot)$ and training data $(\vec{x}_i, y_i) \in \mathcal{X}$ having labels $y_i \in \{-1, 1\}$.

The translations for step sizes are

$$\lambda \equiv \lambda_{(S)} = \left(\frac{1}{t} \right)_{(P)} = \alpha_{(W)} \quad (84)$$

with λ at each step satisfying

$$\frac{1}{\lambda} = \max \left[1, (\epsilon - 1)^{-1} \min \left(\frac{\Delta \alpha^i}{\alpha^i} \right), (\epsilon - 1)^{-1} \min \left(\frac{\Delta \xi^i}{\xi^i} \right) \right] \quad (85)$$

V. PARALLEL INTERIOR POINT METHOD FOR LINEAR REGRESSION

We begin with the initial primal construction, convert to a form dual to the initial primal form and finally, using the constraints at optimality to derive a formulation which is a mixture of the initial primal and dual forms. This hybrid formulation will be taken to be a primal formulation and the primal dual formulation will be constructed. These steps are necessary in order to phrase the problem in a way which will be most amenable to a parallel implementation.

The goal is to minimize the objective function

$$\frac{1}{2} w^a w_a + \tau e_i (\chi^i + \hat{\chi}^i) \quad (86)$$

subject to the inequality constraints

$$y_i - (X_a^i w^a + w_0 e^i) - \eta e^i - \chi^i \leq 0 \quad (87)$$

$$-y_i + (X_a^i w^a + w_0 e^i) - \eta e^i - \hat{\chi}^i \leq 0 \quad (88)$$

$$\chi^i, \hat{\chi}^i \geq 0 \quad (89)$$

The corresponding constrained Lagrangian will be

$$\begin{aligned} L_p &= \frac{1}{2} w^a w_a + \tau e_i (\chi^i + \hat{\chi}^i) + z_i (y^i - (X_a^i w^a + w_0 e^i) - \eta e^i - \chi^i) \\ &\quad - \hat{z}_i (y^i - (X_a^i w^a + w_0 e^i) + \eta e^i + \hat{\chi}^i) - s_i \chi^i - \hat{s}_i \hat{\chi}^i \end{aligned} \quad (90)$$

where $z^i, \hat{z}^i, s^i, \hat{s}^i, \chi^i, \hat{\chi}^i \geq 0$. At a critical solution optimizing equation (90), the partial derivatives of L_p with respect to the primal variables will satisfy

$$\frac{\partial L_p}{\partial w^a} = w_a - X_a^i (z^i - \hat{z}^i) = 0 \quad (91)$$

$$\frac{\partial L_p}{\partial w^0} = e^i (z^i - \hat{z}^i) = 0 \quad (92)$$

$$\frac{\partial L_p}{\partial \chi^i} = \tau e_i - z_i - s_i = 0 \quad (93)$$

$$\frac{\partial L_p}{\partial \hat{\chi}^i} = \tau e_i - \hat{z}_i - \hat{s}_i = 0 \quad (94)$$

which form the set of dual equality constraints for this formulation. Solving the constraints inside the primal objective function, equation (86), yields the dual formulation of the problem, which is to minimize

$$\frac{1}{2} (z^j - \hat{z}^j) X_j^a \delta_{ab} X_i^b (z^i - \hat{z}^i) + (z^i - \hat{z}^i) y_i - (z^i + \hat{z}^i) \eta e_i \quad (95)$$

subject to the inequality constraints

$$0 \leq z^i, \hat{z}^i \leq \tau e^i \quad (96)$$

and equality constraint

$$e_i (z^i - \hat{z}^i) = 0 \quad (97)$$

These are the constraints which will make up the primal problem from which we will start. In this formulation, all $3n + m + 1$ variables $w^\alpha, z^i, \hat{z}^i, \bar{z}^i$ are taken to be primal. The primal dual quadratic problem for this hybrid formulation will be derived in section (VI).

This hybrid formulation of linear SVM regression minimizes an \mathcal{L}^1 error rather than the standard least squares formulation, thus fits to the median of the data rather than the mean. A least squares formulation is simply a change in the cost function, minimizing the \mathcal{L}^2 error instead. The \mathcal{L}^1 minimizer was chosen as a first choice due to the fact that it should be the simplest to implement, yielding a base parallel implementation of a IPM solver to start from with the least overhead. The \mathcal{L}^1 function which is minimized is the sum of the absolute value of the difference between the target value, y^i , and the linear function, $X_a^i w^a + w_0 e^i$, so that w^α are fit to the median of the input data, minimizing

$$\sum_{i=1}^N |y^i - (X_a^i w^a + w_0 e^i)| \quad (98)$$

A least squares fit, minimizing

$$\sum_{i=1}^N |y^i - (X_a^i w^a + w_0 e^i)|^2 \quad (99)$$

would fit to the mean of the input data. Both fits are commonly used, and both are perfectly well defined linear regression problems. We have decided to minimize equation (98) rather than equation (99) to start with due to the fewer number of variables and more simplistic form of the boundary constraints which result.

VI. QUADRATIC PROGRAM SETUP AND INITIAL DATA FOR INTERIOR POINT METHOD

This section currently focuses on the implementation of a parallel IPM used for solving linear separable SVM Regression problems.

Initial data for the $m + 1$ unbounded primal variables will be

$$w^\beta = 0 \quad (100)$$

for all $\beta \in \{0, \dots, m\}$. Initial data for the $2n$ bound primal variables

$$z^i, \hat{z}^i = 0 \quad (101)$$

belonging to the region

$$0 \leq z^i, \hat{z}^i \leq \tau e^i \quad (102)$$

for all $i \in \{1, \dots, n\}$.

For regression problems, it will be useful to add an additional set of variables in order to be able to phrase the problem in a way suitable for an optimal parallel implementation. Define

$$\bar{z}^i \equiv \frac{z^i - \hat{z}^i + \tau e^i}{2} \quad (103)$$

Using this definition, the $3n$ bound primal variables belong to the region

$$0 \leq z^i, \hat{z}^i, \bar{z}^i \leq \tau e^i \quad (104)$$

Initial data for the additional bounded primary variable will be

$$\bar{z}^i = \frac{\tau e^i}{2} \quad (105)$$

These values supply initial data satisfying bounds for all primal variables used. In reality, any value for $z^i, \hat{z}^i, \bar{z}^i$ satisfying equation (104) could be used and, since there are no bounds placed on the $m+1$ unbound primal variables, and $w^\beta \in \mathbb{R}^{m+1}$ would be potentially acceptable for initial data. We define these initial values for simplicity. In practice, initial data will be pushed slightly away from the boundary of the solution region by setting z^i, \hat{z}^i equal to some minimal constants, $\epsilon, \hat{\epsilon}$, specified by the user. We will take these values to be $\epsilon, \hat{\epsilon} = 1.0e-9$ by default.

In this formulation, there will be $6n$ inequality constraints

$$\begin{aligned} -z^i &\leq 0 & z^i - \tau e^i &\leq 0 \\ -\hat{z}^i &\leq 0 & \hat{z}^i - \tau e^i &\leq 0 \\ -\bar{z}^i &\leq 0 & \bar{z}^i - \tau e^i &\leq 0 \end{aligned} \quad (106)$$

Dual to each of these $6n$ inequality constraints, there will be a Lagrange multiplier satisfying $6n$ relations

$$\begin{aligned} -s^i z^i &\leq 0 & v^i (z^i - \tau e^i) &\leq 0 \\ -\hat{s}^i \hat{z}^i &\leq 0 & \hat{v}^i (\hat{z}^i - \tau e^i) &\leq 0 \\ -\bar{s}^i \bar{z}^i &\leq 0 & \bar{v}^i (\bar{z}^i - \tau e^i) &\leq 0 \end{aligned} \quad (107)$$

for all $i \in \{1, \dots, n\}$. This means that the $6n$ Lagrange multipliers satisfy

$$s^i, \hat{s}^i, \bar{s}^i, v^i, \hat{v}^i, \bar{v}^i \geq 0 \quad (108)$$

The Lagrange multipliers are necessary to construct the initial primal-dual problem before reducing to the augmented problem which the IPM will solve.

Initial data for the $6n$ Lagrange multipliers of the primal inequality constraints will be taken to be

$$s^i, \hat{s}^i, \bar{s}^i = \sigma \tau e^i \quad (109)$$

$$v^i, \hat{v}^i, \bar{v}^i = \sigma \tau e^i \quad (110)$$

for constant $\sigma > 0$ to be supplied by the user. We will take $\sigma = 0.5$ to be the default value.

In addition to the $6n$ inequality constraints placed on the primal variables, at the critical solution which optimizes the problem, the primal variables will also satisfy $n+m+1$ equality constraints given by

$$2\bar{z}^i - \tau e^i - z^i + \hat{z}^i = 0 \quad (111)$$

$$w^a - X_i^a (2\bar{z}^i - \tau e^i) = 0 \quad (112)$$

$$2e_i \bar{z}^i - n\tau = 0 \quad (113)$$

for all $a \in \{1, \dots, m\}$ and $i \in \{1, \dots, n\}$. These equality constraints will be known as the primal equality constraints.

At a critical solution which optimizes the problem must satisfy all constraints. Since the function to minimize is quadratic, the problem can be phrased naturally in a primal-dual Lagrangian formulation in which each of the $n+m+1$ equality constraints is added to the Lagrangian multiplied by an unbound Lagrange multiplier, denoted by $\lambda_\alpha, \lambda_i$, and each of the $6n$ inequality constraints multiplied by the corresponding $6n$ dual Lagrange multipliers is subtracted. Including all constraints, equality and inequality, along with their dual Lagrange multipliers, the fully constrained Lagrangian for the primal-dual formulation can be written as

$$\begin{aligned} L = & f(\mathbf{w}, \mathbf{z}, \hat{\mathbf{z}}, \bar{\mathbf{z}}) \\ & + \lambda_i (2\bar{z}^i - \tau e^i - z^i + \hat{z}^i) \\ & + \lambda_a (w^a - X_i^a (2\bar{z}^i - \tau e^i)) \\ & + \lambda_0 (2e_i \bar{z}^i - n\tau) \\ & - s_i z^i - \hat{s}_i \hat{z}^i - \bar{s}_i \bar{z}^i \\ & + v_i (z^i - \tau e^i) + \hat{v}_i (\hat{z}^i - \tau e^i) + \bar{v}_i (\bar{z}^i - \tau e^i) \end{aligned} \quad (114)$$

for the objective function

$$f(\mathbf{w}, \mathbf{z}, \hat{\mathbf{z}}, \bar{\mathbf{z}}) \equiv \frac{1}{2} w^a w_a + \eta e_i (z^i + \hat{z}^i) - y_i (2\bar{z}^i - \tau e^i) \quad (115)$$

with user supplied insensitivity parameter $0 \leq \eta$. The primal-dual Lagrangian formulation is the quadratic program which our IPM solver will seek an optimal solution to.

At a critical solution optimizing the problem, all first partial derivatives of the Lagrangian with respect to either primal or dual variables must either vanish or, for the case of the Lagrange multipliers of inequality constraints, be bounded, otherwise the solution could not be optimal. Stated differently, the primal-dual Lagrangian problem is to be minimized with respect to primal variables and maximized with respect to all dual variables. Since the first partial derivative of the Lagrangian with respect to any of the $n + m + 1$ primal variables must vanish for a solution to be found, the first partial derivatives with respect to each of the primal variables will yield a new equality constraint. Thus, in the primal-dual Lagrangian formulation, there will be an additional $3n + m + 1$ equality constraints involving primal variables as well as Lagrange multipliers, given by

$$\frac{\partial f}{\partial w^a} + \lambda_a = 0 \quad (116)$$

$$\frac{\partial f}{\partial w^0} = 0 \quad (117)$$

$$\frac{\partial f}{\partial z^i} - \lambda_i - s_i + v_i = 0 \quad (118)$$

$$\frac{\partial f}{\partial \hat{z}^i} + \lambda_i - \hat{s}_i + \hat{v}_i = 0 \quad (119)$$

$$\frac{\partial f}{\partial \bar{z}^i} + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \bar{s}_i + \bar{v}_i = 0 \quad (120)$$

These constraints will be known as the dual equality constraints. Inserting the objective function defined in equation (115) we see that one of the dual equality constraints is trivial, since $\frac{\partial f}{\partial w^0} = 0$. The remaining $3n + m$ constraints take the form

$$w_a + \lambda_a = 0 \quad (121)$$

$$\eta e_i - \lambda_i - s_i + v_i = 0 \quad (122)$$

$$\eta e_i + \lambda_i - \hat{s}_i + \hat{v}_i = 0 \quad (123)$$

$$-2y_i + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \bar{s}_i + \bar{v}_i = 0 \quad (124)$$

As will all equality constraints, these constraints can only be expected to vanish simultaneously once a feasible solution has been found, hence we don't solve these constraints directly within the equations but rather start at some point which perhaps violates the equality constraints and let the IPM find the optimal solution.

Returning to the inequality constraints, once a critical solution is found optimizing the primal-dual Lagrangian, the primal-dual variables of the inequality constraints must satisfy

$$s_i z^i = 0 \quad v_i (\tau e^i - z^i) = 0 \quad (125)$$

$$\hat{s}_i \hat{z}^i = 0 \quad \hat{v}_i (\tau e^i - \hat{z}^i) = 0$$

$$\bar{s}_i \bar{z}^i = 0 \quad \bar{v}_i (\tau e^i - \bar{z}^i) = 0$$

for the Lagrangian to be maximized with respect to dual variables and minimized with respect to primal variables. Since any solution must satisfy

$$\begin{aligned} 0 &\leq s^i, \hat{s}^i, \bar{s}^i \\ 0 &\leq z^i, \hat{z}^i, \bar{z}^i \\ 0 &\leq (\tau e^i - z^i), (\tau e^i - \hat{z}^i), (\tau e^i - \bar{z}^i) \\ 0 &\leq v^i, \hat{v}^i, \bar{v}^i \end{aligned} \quad (126)$$

for all $i \in \{1, \dots, n\}$, equation (125) requires either the primal or dual variable in each of the $6n$ pairs must vanish once a solution is found. Notice that the equality constraints of equation (125) are not invertible and so can not be solved to yield an expression for $3n$ of the variables in terms of the corresponding primal-dual pair. This is due entirely to the fact that one can not divide by zero. The interior point method avoids this issue by constructing an

iterative method which converges to the optimal solution by replacing the equality constraints of equation (125) with a set of $6n$ equality constraints of the form

$$\begin{aligned} s^i z^i &= \mu & v^i (\tau e^i - z^i) &= \mu \\ \hat{s}^i \hat{z}^i &= \mu & \hat{v}^i (\tau e^i - \hat{z}^i) &= \mu \\ \bar{s}^i \bar{z}^i &= \mu & \bar{v}^i (\tau e^i - \bar{z}^i) &= \mu \end{aligned} \quad (127)$$

for some $\mu > 0$ and successively reduces the magnitude of μ at each iteration, thereby converging on an optimal solution in the limit $\mu \rightarrow 0$. These constraints will be known as the central equality constraints since the path traced by solutions as $\mu \rightarrow 0$ is known as the central path.

A. Newtons Method and the Linearized Formulation

The infeasible start IPM solver uses Newtons method to find the update direction, solving a linearized problem of the form

$$C(\mathbf{q}) + \nabla C(\mathbf{q}) \cdot \Delta \mathbf{q} = 0 \quad (128)$$

for a formulation with variables \mathbf{q} , a matrix of equality constraints $C(\mathbf{q})$ evaluated at the point \mathbf{q} and gradient of the matrix of equality constraints $\nabla C(\mathbf{q})$, evaluated at \mathbf{q} , acting on the first order variation $\Delta \mathbf{q}$. At a fixed \mathbf{q} , equation (128) can be solved to yield an expression for the variational vector

$$\Delta \mathbf{q} = -[\nabla C(\mathbf{q})]^{-1} C(\mathbf{q}) \quad (129)$$

whenever $[\nabla C(\mathbf{q})]^{-1}$ exists. The primal-dual Lagrangian formulation is designed to ensure that this will be true within some bounded region of values.

Linearizing the equality constraints yields

$$2\Delta \bar{z}^i - \Delta z^i + \Delta \hat{z}^i \quad (130)$$

$$\Delta w^a - 2X_i^a \Delta \bar{z}^i \quad (131)$$

$$2e_i \Delta \bar{z}^i \quad (132)$$

for the $n + m + 1$ primal equality constraints

$$\Delta w_a + \Delta \lambda_a \quad (133)$$

$$-\Delta \lambda_i - \Delta s_i + \Delta v_i \quad (134)$$

$$\Delta \lambda_i - \Delta \hat{s}_i + \Delta \hat{v}_i \quad (135)$$

$$2\Delta \lambda_i - 2\Delta \lambda_a X_i^a + 2\Delta \lambda_0 e_i - \Delta \bar{s}_i + \Delta \bar{v}_i \quad (136)$$

for the $3n + m$ dual equality constraints and

$$\Delta s^i z^i + s^i \Delta z^i + \Delta s^i \Delta z^i \quad (137)$$

$$\Delta v^i (\tau e^i - z^i) - v^i \Delta z^i - \Delta v^i \Delta z^i \quad (138)$$

$$\Delta \hat{s}^i \hat{z}^i + \hat{s}^i \Delta \hat{z}^i + \Delta \hat{s}^i \Delta \hat{z}^i \quad (139)$$

$$\Delta \hat{v}^i (\tau e^i - \hat{z}^i) - \hat{v}^i \Delta \hat{z}^i - \Delta \hat{v}^i \Delta \hat{z}^i \quad (140)$$

$$\Delta \bar{s}^i \bar{z}^i + \bar{s}^i \Delta \bar{z}^i + \Delta \bar{s}^i \Delta \bar{z}^i \quad (141)$$

$$\Delta \bar{v}^i (\tau e^i - \bar{z}^i) - \bar{v}^i \Delta \bar{z}^i - \Delta \bar{v}^i \Delta \bar{z}^i \quad (142)$$

for the $6n$ central equality constraints. Notice that the central equality constraints contain terms which are quadratic in the variational terms, hence these constraints do not take the form necessary to be able to use Newtons method to solve for the variational vector. This can be remedied through a block reduction of the system, splitting updates to the dual and primal parts making up the boundary inequality constraints.

The block reduction is achieved by $\Delta q \ll 0$ so that terms of order $|\Delta \mathbf{q}|^2$ and higher can be neglected, yielding $6n$ new central equality equations

$$\begin{aligned}
& \Delta s^i z^i + s^i \Delta z^i \\
\Delta v^i (\tau e^i - z^i) - v^i \Delta z^i \\
& \Delta \hat{s}^i \hat{z}^i + \hat{s}^i \Delta \hat{z}^i \\
\Delta \hat{v}^i (\tau e^i - \hat{z}^i) - \hat{v}^i \Delta \hat{z}^i \\
& \Delta \bar{s}^i \bar{z}^i + \bar{s}^i \Delta \bar{z}^i \\
\Delta \bar{v}^i (\tau e^i - \bar{z}^i) - \bar{v}^i \Delta \bar{z}^i
\end{aligned} \tag{143}$$

Then, using these equations, use Newtons method to solve for $\Delta s^i, \Delta \hat{s}^i, \Delta \bar{s}^i, \Delta v^i, \Delta \hat{v}^i, \Delta \bar{v}^i = 0$. This will yield $6n$ expressions which are linearly functions of the variational components $\Delta z^i, \Delta \hat{z}^i, \Delta \bar{z}^i$.

These expressions are then incorporated into equation (128) reducing the total number of equations to solve for simultaneously by dropping the $6n$ equations for the variation of the Lagrange multipliers dual to the inequality constraints. The reduced system is then solved yielding values for the remaining $6n+2m+1$ terms in the variational vector. Inserting these values into equation (137) and solving yields values for the variations $\Delta s^i, \Delta \hat{s}^i, \Delta \bar{s}^i, \Delta v^i, \Delta \hat{v}^i, \Delta \bar{v}^i$.

The linearized constraints of equations (130), (133) and (143) define the gradient constraint term found in Newtons method, $\nabla C(\mathbf{q}) \cdot \Delta \mathbf{q}$, for our linear SVM regression formulation.

B. Line Search and Step Length

Line search for step length in Newton method step will yield two values; an optimal step length for the updates to the Lagrange multipliers of the inequality constraints ensuring that the updated values satisfy equation (108), and an optimal step length for the updates to the primal variables ensuring that the updated values satisfy equation (106).

Once μ_k and equality constraint residuals have been calculated, the augmented system provides a solution for $\Delta z^i, \Delta \hat{z}^i, \Delta \bar{z}^i$ which, through the reduction process, yield values for $\Delta s_i, \Delta \hat{s}_i, \Delta \bar{s}_i, \Delta v_i, \Delta \hat{v}_i, \Delta \bar{v}_i$ which are then used to find an optimal step length for primal variables, given by

$$\begin{aligned}
\frac{1}{\gamma_p} = \max(1, & \quad \frac{1}{\epsilon-1} \min\left(\frac{\Delta z^i}{z^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \hat{z}^i}{\hat{z}^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \bar{z}^i}{\bar{z}^i}\right), \\
& \quad \frac{1}{\epsilon-1} \min\left(\frac{\Delta z^i}{\tau e^i - z^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \hat{z}^i}{\tau e^i - \hat{z}^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \bar{z}^i}{\tau e^i - \bar{z}^i}\right))
\end{aligned} \tag{144}$$

and, similarly, an optimal step length for dual variables, given by

$$\begin{aligned}
\frac{1}{\gamma_d} = \max(1, & \quad \frac{1}{\epsilon-1} \min\left(\frac{\Delta s^i}{s^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \hat{s}^i}{\hat{s}^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \bar{s}^i}{\bar{s}^i}\right), \\
& \quad \frac{1}{\epsilon-1} \min\left(\frac{\Delta v^i}{v^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \hat{v}^i}{\hat{v}^i}\right), \frac{1}{\epsilon-1} \min\left(\frac{\Delta \bar{v}^i}{\bar{v}^i}\right))
\end{aligned} \tag{145}$$

For full primal-dual search step, set $\gamma \equiv \max(\gamma_p, \gamma_d)$ and use γ as step length.

VII. INTERIOR POINT METHOD IMPLEMENTATION DETAILS

This section details the calculations necessary for an efficient implementation of an interior point method solver for the \mathcal{L}_1 linear regression problem considered here. All formulas necessary for the MapReduce implementation presented in section (VIII) are derived in laborious detail. This section concludes with an algorithm for a standard implementation of an interior point method solver.

The linear SVM regression formulation laid out above is ideal for a parallel implementation whenever $m+1 \ll n$ for a number of reasons.

First, the system of equations to be solved at each iteration is comprised of $n+m+1$ primal equality constraints, $3n+m$ dual equality constraints and $6n$ central equality constraints. Supposing $m+1 \ll n$ so that there is no real overhead in storing the m components weight vector and $m+1$ Lagrange multipliers, λ_α , then of the $10n+2m+1$ total constraints to be solved simultaneously, only $m+1$ will involve any data which is not locally available, thereby minimizing the amount of data which must be transferred between machines.

Second, because the kernel matrix is diagonal, the computation necessary to be performed on a single machine during each iteration is reduced to the problem of inverting a dense $(m+1) \times (m+1)$ matrix.

At each iteration, the infeasible interior point method solver using Newton's method to determine the variational direction needs to have the residuals from all primal, dual and central constraints

$$\rho_p^i \equiv 2\bar{z}^i - \tau e^i - z^i + \hat{z}^i \quad (146)$$

$$\rho_p^a \equiv w^a - X_i^a (2\bar{z}^i - \tau e^i) \quad (147)$$

$$\rho_p^0 \equiv 2e_i \bar{z}^i - n\tau \quad (148)$$

$$\rho_d^a \equiv w^a + \lambda^a \quad (149)$$

$$\rho_d^i \equiv \eta e^i - \lambda^i - s^i + v^i \quad (150)$$

$$\hat{\rho}_d^i \equiv \eta e^i + \lambda^i - \hat{s}^i + \hat{v}^i \quad (151)$$

$$\bar{\rho}_d^i \equiv -2y^i + 2\lambda^i - 2\lambda^a X_a^i + 2\lambda_0 e^i - \bar{s}^i + \bar{v}^i \quad (152)$$

$$(-) \rho_c^i \equiv s^i z^i - \mu \quad (153)$$

$$(+) \rho_c^i \equiv v^i (\tau e^i - z^i) - \mu \quad (154)$$

$$(-) \hat{\rho}_c^i \equiv \hat{s}^i \hat{z}^i - \mu \quad (155)$$

$$(+) \hat{\rho}_c^i \equiv \hat{v}^i (\tau e^i - \hat{z}^i) - \mu \quad (156)$$

$$(-) \bar{\rho}_c^i \equiv \bar{s}^i \bar{z}^i - \mu \quad (157)$$

$$(+) \bar{\rho}_c^i \equiv \bar{v}^i (\tau e^i - \bar{z}^i) - \mu \quad (158)$$

These will form the right hand side of Newton's method, $-\nabla C(\mathbf{q}) \cdot \Delta \mathbf{q} = C(\mathbf{q})$. The left hand side will be comprised of the linearized variation equations

$$2\Delta \bar{z}^i - \Delta z^i + \Delta \hat{z}^i \quad (159)$$

$$\Delta w^a - 2X_i^a \Delta \bar{z}^i \quad (160)$$

$$2e_i \Delta \bar{z}^i \quad (161)$$

$$\Delta w_a + \Delta \lambda_a \quad (162)$$

$$-\Delta \lambda_i - \Delta s_i + \Delta v_i \quad (163)$$

$$\Delta \lambda_i - \Delta \hat{s}_i + \Delta \hat{v}_i \quad (164)$$

$$2\Delta \lambda_i - 2\Delta \lambda_a X_a^i + 2\Delta \lambda_0 e_i - \Delta \bar{s}_i + \Delta \bar{v}_i \quad (165)$$

$$\Delta s^i z^i + s^i \Delta z^i \quad (166)$$

$$\Delta v^i (\tau e^i - z^i) - v^i \Delta z^i \quad (167)$$

$$\Delta \hat{s}^i \hat{z}^i + \hat{s}^i \Delta \hat{z}^i \quad (168)$$

$$\Delta \hat{v}^i (\tau e^i - \hat{z}^i) - \hat{v}^i \Delta \hat{z}^i \quad (169)$$

$$\Delta \bar{s}^i \bar{z}^i + \bar{s}^i \Delta \bar{z}^i \quad (170)$$

$$\Delta \bar{v}^i (\tau e^i - \bar{z}^i) - \bar{v}^i \Delta \bar{z}^i \quad (171)$$

Combining the left and right hand sides, equation (128) in our implementation takes the form

$$-2\Delta\bar{z}^i + \Delta z^i - \Delta\hat{z}^i = 2\bar{z}^i - \tau e^i - z^i + \hat{z}^i \equiv \rho_p^i \quad (172)$$

$$-\Delta w^a + 2X_i^a \Delta\bar{z}^i = w^a - X_i^a (2\bar{z}^i - \tau e^i) \equiv \rho_p^a \quad (173)$$

$$-2e_i \Delta\bar{z}^i = 2e_i \bar{z}^i - n\tau \equiv \rho_p^0 \quad (174)$$

$$-\Delta w_a - \Delta\lambda_a = w_a + \lambda_a \equiv \rho_d^a \quad (175)$$

$$\Delta\lambda_i + \Delta s_i - \Delta v_i = \eta e_i - \lambda_i - s_i + v_i \equiv \rho_d^i \quad (176)$$

$$-\Delta\lambda_i + \Delta\hat{s}_i - \Delta\hat{v}_i = \eta e_i + \lambda_i - \hat{s}_i + \hat{v}_i \equiv \hat{\rho}_d^i \quad (177)$$

$$-2\Delta\lambda_i + 2\Delta\lambda_a X_i^a - 2\Delta\lambda_0 e_i + \Delta\bar{s}_i - \Delta\bar{v}_i = -2y_i + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \bar{s}_i + \bar{v}_i \equiv \bar{\rho}_d^i \quad (178)$$

$$\Delta s^i z^i + s^i \Delta z^i = s^i z^i - \mu \equiv -^{(-)}\rho_c^i \quad (179)$$

$$\Delta v^i (\tau e^i - z^i) - v^i \Delta z^i = v^i (\tau e^i - z^i) - \mu \equiv -^{(+)}\rho_c^i \quad (180)$$

$$\Delta\hat{s}^i \hat{z}^i + \hat{s}^i \Delta\hat{z}^i = \hat{s}^i \hat{z}^i - \mu \equiv -^{(-)}\hat{\rho}_c^i \quad (181)$$

$$\Delta\hat{v}^i (\tau e^i - \hat{z}^i) - \hat{v}^i \Delta\hat{z}^i = \hat{v}^i (\tau e^i - \hat{z}^i) - \mu \equiv -^{(+)}\hat{\rho}_c^i \quad (182)$$

$$\Delta\bar{s}^i \bar{z}^i + \bar{s}^i \Delta\bar{z}^i = \bar{s}^i \bar{z}^i - \mu \equiv -^{(-)}\bar{\rho}_c^i \quad (183)$$

$$\Delta\bar{v}^i (\tau e^i - \bar{z}^i) - \bar{v}^i \Delta\bar{z}^i = \bar{v}^i (\tau e^i - \bar{z}^i) - \mu \equiv -^{(+)}\bar{\rho}_c^i \quad (184)$$

These equations form the basis for the full primal dual formulation. In order to make the calculations more manageable, constraints will be imposed and block elimination employed to yield a reduced, augmented system.

Reduce the system by dropping the primal variable w^0 which does not appear in the system of constraints. Define the vector of primal variables

$$\mathbf{q} \equiv (\mathbf{w}, \mathbf{z}, \hat{\mathbf{z}}, \bar{\mathbf{z}}) \quad (185)$$

with $\mathbf{w} \equiv w^a$. Then, as in section (II), the primal constraints can be expressed as

$$\mathbf{A}\mathbf{q} - \mathbf{b} = \vec{\rho}_p \quad (186)$$

where

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{0}_{(n \times m)} & -\mathbb{I}_{(n \times n)} & \mathbb{I}_{(n \times n)} & 2\mathbb{I}_{(n \times n)} \\ \mathbb{I}_{(m \times m)} & \mathbf{0}_{(m \times n)} & \mathbf{0}_{(m \times n)} & -2\mathbf{X}_{(m \times n)} \\ \mathbf{0}_{(1 \times m)} & \mathbf{0}_{(1 \times n)} & \mathbf{0}_{(1 \times n)} & 2\mathbf{e}_{(1 \times n)} \end{bmatrix} \quad (187)$$

which defines an $(n + m + 1) \times (3n + m)$ constant matrix, and

$$\mathbf{b} \equiv \tau \begin{bmatrix} \mathbf{e}_{(n \times 1)}^T \\ -\mathbf{P}_{(m \times 1)} \\ n\mathbb{I}_{(1 \times 1)} \end{bmatrix} \quad (188)$$

for constant $(m \times 1)$ source vector

$$\mathbf{P} \equiv P^a = P_i^a e^i \quad (189)$$

The primal equality constraints will form the basis for the reduced system of equations which will be solved for using Newton's method.

In the notation of section (II), the dual constraints can be expressed as

$$-\mathbf{Q}\vec{q} - \vec{c} + \mathbf{A}^T \vec{\lambda} + \vec{s} - \vec{v} = \vec{\rho}_d \quad (190)$$

where $s^a = v^a = 0$ since the primal variables w^a are free, having no bounds placed on them. In our formulation, the matrix \mathbf{Q} will be given by

$$\mathbf{Q} \equiv \begin{bmatrix} \mathbb{I}_{(m \times m)} & \mathbf{0}_{(m \times 3n)} \\ \mathbf{0}_{(3n \times m)} & \mathbf{0}_{(3n \times 3n)} \end{bmatrix} \quad (191)$$

which is a $(3n + m) \times (3n + m)$ constant matrix.

Note that the primal variable w^0 does not show up in any of the constraints, and thus won't be updated using Newton's method. This means that w^0 can take on any value at each step. For this reason, we will augment the system with an additional dual constraint

$$-\rho_d^0 = w^0 + \lambda^0 \quad (192)$$

The dual constraints $-\rho_d^\alpha = w^\alpha + \lambda^\alpha$ will be set equal to zero and the λ^α solved for, allowing us to reduce the system by $m + 1$ variables. This reduction can only be imposed after the partial derivatives of the Lagrangian and subsequent constraints have been taken, since λ^α are technically Lagrange multipliers and whence only have values defined along the extremal path so must not be varied independently. As a consequence, in general, if you were to insert $\lambda^\alpha = -w^\alpha$ into the Lagrangian, equation (114), and derive constraints, you would get a different system of equations. In this particular case, solving $\rho_d^\alpha = 0$ and inserting the solution into the Lagrangian does not disturb the constraints but does change the sign of the Lagrangian, changing the minimization problem to a maximization problem. In order to avoid any confusion and keep things as general as possible, constraints will be imposed to reduce the resulting system to be solved only after all partial derivatives with respect to \mathbf{q} have been taken.

The system can be reduced further by noting that the sub-block of the linearized system, equation (172), which is generated by the lower and upper inequality constraints can be solved for the variations of the inequality Lagrange multipliers

$$\Delta s^i = \frac{-1}{z^i} (s^i z^i - \mu - s^i \Delta z^i) \quad (193)$$

$$\Delta v^i = \frac{-1}{(\tau e^i - z^i)} (v^i \Delta z^i + v^i (\tau e^i - z^i) - \mu) \quad (194)$$

$$\Delta \hat{s}^i = \frac{-1}{\hat{z}^i} (\hat{s}^i \hat{z}^i - \mu - \hat{s}^i \Delta \hat{z}^i) \quad (195)$$

$$\Delta \hat{v}^i = \frac{-1}{(\tau e^i - \hat{z}^i)} (\hat{v}^i \Delta \hat{z}^i + \hat{v}^i (\tau e^i - \hat{z}^i) - \mu) \quad (196)$$

$$\Delta \bar{s}^i = \frac{-1}{\bar{z}^i} (\bar{s}^i \bar{z}^i - \mu - \bar{s}^i \Delta \bar{z}^i) \quad (197)$$

$$\Delta \bar{v}^i = \frac{-1}{(\tau e^i - \bar{z}^i)} (\bar{v}^i \Delta \bar{z}^i + \bar{v}^i (\tau e^i - \bar{z}^i) - \mu) \quad (198)$$

which can be inserted into equation (172), yielding the reduced system. After the reduced system is used to solve for updates to $\lambda^i, w^\alpha, z^i, \hat{z}^i, \bar{z}^i$, the variations are inserted into equation (193) to derive variations for the inequality Lagrange multipliers.

The reduction imposed by solving equation (193) leaves the sub-block of primal constraints invariant, altering only the linearized system for the dual constraints, yielding

$$\Delta \lambda_i + \left(\frac{s^i}{z^i} + \frac{v^i}{(\tau e^i - z^i)} \right) \Delta z^i = \eta e_i - \lambda_i - \frac{\mu}{z^i} + \frac{\mu}{(\tau e^i - z^i)} \quad (199)$$

$$-\Delta \lambda_i + \left(\frac{\hat{s}^i}{\hat{z}^i} + \frac{\hat{v}^i}{(\tau e^i - \hat{z}^i)} \right) \Delta \hat{z}^i = \eta e_i + \lambda_i - \frac{\mu}{\hat{z}^i} + \frac{\mu}{(\tau e^i - \hat{z}^i)} \quad (200)$$

$$-2\Delta \lambda_i + 2\Delta \lambda_a X_i^a - 2\Delta \lambda_0 e_i + \left(\frac{\bar{s}^i}{\bar{z}^i} + \frac{\bar{v}^i}{(\tau e^i - \bar{z}^i)} \right) \Delta \bar{z}^i = -2y_i + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \frac{\mu}{\bar{z}^i} + \frac{\mu}{(\tau e^i - \bar{z}^i)} \quad (201)$$

where $\lambda^\alpha = -w^\alpha$. The reduced system is to be solved to yield values for the remaining $(4n + m + 1)$ variations.

The right hand side of the reduced system, equation (199), will be denoted

$$\rho_{d'}^i \equiv \eta e_i - \lambda_i - \frac{\mu}{z^i} + \frac{\mu}{(\tau e^i - z^i)} \quad (202)$$

$$\hat{\rho}_{d'}^i \equiv \eta e_i + \lambda_i - \frac{\mu}{\hat{z}^i} + \frac{\mu}{(\tau e^i - \hat{z}^i)} \quad (203)$$

$$\bar{\rho}_{d'}^i \equiv -2y_i + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \frac{\mu}{\bar{z}^i} + \frac{\mu}{(\tau e^i - \bar{z}^i)} \quad (204)$$

and the reduced system can be expressed in the form

$$-(\mathbf{Q} + \mathbf{D}) \Delta \vec{q} - \mathbf{A}^T \Delta \vec{\lambda} = \vec{\rho}_{d'} \quad (205)$$

$$-\mathbf{A} \Delta \vec{q} = \vec{\rho}_p \quad (206)$$

where

$$\mathbf{D} \equiv \begin{bmatrix} \mathbf{0}_{(m \times m)} & \mathbf{0}_{(m \times n)} & \mathbf{0}_{(m \times n)} & \mathbf{0}_{(n \times n)} \\ \mathbf{0}_{(n \times m)} & \mathbf{D}_z \text{ (} n \times n \text{)} & \mathbf{0}_{(n \times n)} & \mathbf{0}_{(n \times n)} \\ \mathbf{0}_{(n \times m)} & \mathbf{0}_{(n \times n)} & \mathbf{D}_{\hat{z}} \text{ (} n \times n \text{)} & \mathbf{0}_{(n \times n)} \\ \mathbf{0}_{(n \times m)} & \mathbf{0}_{(n \times n)} & \mathbf{0}_{(n \times n)} & \mathbf{D}_{\bar{z}} \text{ (} n \times n \text{)} \end{bmatrix} \quad (207)$$

and

$$\mathbf{D}_z = D_z^{ii} \equiv \left(\frac{s^i}{z^i} + \frac{v^i}{(\tau e^i - z^i)} \right) \quad (208)$$

$$\mathbf{D}_{\hat{z}} = D_{\hat{z}}^{ii} \equiv \left(\frac{\hat{s}^i}{\hat{z}^i} + \frac{\hat{v}^i}{(\tau e^i - \hat{z}^i)} \right) \quad (209)$$

$$\mathbf{D}_{\bar{z}} = D_{\bar{z}}^{ii} \equiv \left(\frac{\bar{s}^i}{\bar{z}^i} + \frac{\bar{v}^i}{(\tau e^i - \bar{z}^i)} \right) \quad (210)$$

which is a diagonal matrix. The inequality constraints ensure that the $3n$ non-zero diagonal components of \mathbf{D} will always remain in the positive quadrant. Since \mathbf{D} is diagonal with positive, non-vanishing, components, the $3n \times 3n$ sub-block of maximal rank can be inverted. Since \mathbf{Q} is diagonal and positive for the $m \times m$ sub-block of maximal rank, the combination, $(\mathbf{Q} + \mathbf{D})$ is an invertible $(3n + m) \times (3n + m)$ matrix.

The reduced system of equation (202) can be reduced further by using the equations to solve for $\Delta \mathbf{q}$ which can be accomplished by using the invertibility of the diagonal matrix $(\mathbf{Q} + \mathbf{D})$ to yield

$$\Delta \vec{q} = - \left[(\mathbf{Q} + \mathbf{D})^{-1} \vec{\rho}_d + (\mathbf{Q} + \mathbf{D})^{-1} \mathbf{A}^T \Delta \vec{\lambda} \right] \quad (211)$$

which can then be inserted into the remaining equation $-\mathbf{A} \Delta \vec{q} = \vec{\rho}_p$ to yield the desired result

$$\mathbf{A} (\mathbf{Q} + \mathbf{D})^{-1} \mathbf{A}^T \Delta \vec{\lambda} = -\vec{\rho}_p - \mathbf{A} (\mathbf{Q} + \mathbf{D})^{-1} \vec{\rho}_d \quad (212)$$

Define

$$\mathbf{M} \equiv \mathbf{A} (\mathbf{Q} + \mathbf{D})^{-1} \mathbf{A}^T \quad (213)$$

$$\vec{\rho}_{p'} \equiv \vec{\rho}_p - \mathbf{A} (\mathbf{Q} + \mathbf{D})^{-1} \vec{\rho}_d \quad (214)$$

The matrix \mathbf{M} is an $(n + m + 1) \times (n + m + 1)$ symmetric positive definite invertible matrix. The reduced system takes the form

$$\mathbf{M} \Delta \vec{\lambda} = -\vec{\rho}_{p'} \quad (215)$$

The invertible symmetric positive definite matrix \mathbf{M} can be factorized using a Cholesky decomposition, satisfying

$$\mathbf{L} \Sigma \mathbf{L}^T = \mathbf{M} \quad (216)$$

for diagonal positive definite square $(n + m + 1) \times (n + m + 1)$ matrix Σ and lower triangular square $(n + m + 1) \times (n + m + 1)$ matrix \mathbf{L} . Using equations (187), (191) and (207), the matrix \mathbf{M} will take the form

$$\mathbf{M} \equiv \begin{bmatrix} \mathbf{S}_O^{-1} \text{ (} n \times n \text{)} & [-4\mathbf{X}\mathbf{D}_{\bar{z}}^{-1}] \text{ (} n \times m \text{)} & [4\mathbf{e}^T \mathbf{D}_{\bar{z}}^{-1}] \text{ (} n \times 1 \text{)} \\ [-4\mathbf{X}\mathbf{D}_{\bar{z}}^{-1}]^T \text{ (} m \times n \text{)} & [\mathbf{Q} + 4\mathbf{X}\mathbf{D}_{\bar{z}}^{-1} \mathbf{X}^T] \text{ (} m \times m \text{)} & -[4\mathbf{X}\mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T] \text{ (} m \times 1 \text{)} \\ [4\mathbf{e}^T \mathbf{D}_{\bar{z}}^{-1}]^T \text{ (} 1 \times n \text{)} & -[4\mathbf{X}\mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T]^T \text{ (} 1 \times m \text{)} & [4\mathbf{e} \mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T] \text{ (} 1 \times 1 \text{)} \end{bmatrix} \quad (217)$$

where

$$\mathbf{S}_O^{-1} = \mathbf{D}_z^{-1} + \mathbf{D}_{\hat{z}}^{-1} + 4\mathbf{D}_{\bar{z}}^{-1} \quad (218)$$

Instead of inverting \mathbf{M} directly, we will use a Cholesky factorization of \mathbf{M} along with a forward-backward linear solve to solve for the variation direction.

From equations (217) and (218), the Cholesky factorization of \mathbf{M} must yield a lower triangular matrix \mathbf{L} of the form

$$\mathbf{L} \equiv \begin{bmatrix} \mathbb{I}_{(n \times n)} & \mathbf{0}_{(n \times (m+1))} \\ \mathbf{F}_{((m+1) \times n)} & \mathbf{L}^W_{((m+1) \times (m+1))} \end{bmatrix} \quad (219)$$

and diagonal

$$\Sigma \equiv \begin{bmatrix} \mathbf{S}_O^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_W^{-1} \end{bmatrix} \quad (220)$$

for some as of yet undetermined diagonal $(m+1) \times (m+1)$ matrix \mathbf{S}_W^{-1} , $(n \times m+1)$ matrix \mathbf{F} and lower triangular $(m+1) \times (m+1)$ matrix \mathbf{L}^W . The labels O and W for sub-blocks \mathbf{S}_O^{-1} , \mathbf{S}_W^{-1} and \mathbf{L}^W refer to "observations" and "weights", serving as mnemonic devices to remind us that these matrices act on spaces of dimensions equal to the number of input observations, n , and the number feature weights and bias, $m+1$, respectively.

Using equations (219) and (220), equation (216) reads

$$\begin{bmatrix} \mathbf{S}_O^{-1} & [\mathbf{S}_O^{-1} \mathbf{F}^T] \\ [\mathbf{S}_O^{-1} \mathbf{F}^T]^T & [\mathbf{L}^W \mathbf{S}_W^{-1} \mathbf{L}_W^T + \mathbf{F} \mathbf{S}_O^{-1} \mathbf{F}^T] \end{bmatrix} = \mathbf{M}_{((n+m+1) \times (n+m+1))} \quad (221)$$

Inserting this result into equation (217) and solving for \mathbf{F} yields

$$\mathbf{F}_{((m+1) \times n)} \equiv 4 \begin{bmatrix} -\mathbf{X} \\ \mathbf{e} \end{bmatrix} [\mathbf{S}_O \mathbf{D}_{\bar{z}}^{-1}]_{(n \times n)} \quad (222)$$

Inserting the result for \mathbf{F} into equation (221) and solving $\mathbf{L} \Sigma \mathbf{L}^T = \mathbf{M}$ using equation (217) yields

$$\mathbf{L}^W \mathbf{S}_W^{-1} \mathbf{L}_W^T = \mathbf{M}_W - \mathbf{F} \mathbf{S}_O^{-1} \mathbf{F}^T \quad (223)$$

where

$$\mathbf{M}_W \equiv \begin{bmatrix} [\mathbb{I} + 4\mathbf{X} \mathbf{D}_{\bar{z}}^{-1} \mathbf{X}^T] & -[4\mathbf{X} \mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T] \\ -[4\mathbf{X} \mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T]^T & [4\mathbf{e} \mathbf{D}_{\bar{z}}^{-1} \mathbf{e}^T] \end{bmatrix} \quad (224)$$

The left hand side of equation (223) is the $(m+1) \times (m+1)$ sub-block for which the Cholesky factorization must be computed, given the values of the right hand side.

The square symmetric positive definite $(m+1) \times (m+1)$ matrix which must be decomposed will be denoted

$$\mathbf{M}'_W \equiv \mathbf{M}_W - \mathbf{F} \mathbf{S}_O^{-1} \mathbf{F}^T \quad (225)$$

In the following, the label W will be dropped from the matrices \mathbf{M}'_W , \mathbf{L}_W and \mathbf{S}_W^{-1} to facilitate readability. Starting with $L_{\gamma}^{\beta} = 0$, $S_{\gamma}^{-1} = 0$, the Cholesky factorization will be determined from the recursive solutions, running from $\alpha = 0$ to $\alpha = m$, of

$$S_{\alpha\alpha}^{-1} = M'_{\alpha\alpha} - L_{\alpha}^{\delta} L_{\alpha}^{\delta} S_{\beta\beta}^{-1} \quad (226)$$

$$L_{\beta}^{\alpha} = S^{\alpha\alpha} (M'_{\beta\alpha} - L_{\alpha}^{\gamma} L_{\beta}^{\gamma} S_{\gamma}^{-1}) \quad (227)$$

where $\alpha, \beta, \gamma \in \{0, \dots, m\}$. A Cholesky decomposition will be done at each iteration of the interior point method.

Once the $(m+1) \times (m+1)$ matrices \mathbf{L}_W and \mathbf{S}_W^{-1} have been determined, the $(n+m+1) \times (n+m+1)$ matrices \mathbf{L} and \mathbf{S}^{-1} will be completely determined and can be used to solve the linear system

$$\mathbf{L} \Delta \lambda^{(2)} = -\vec{\rho}_p \quad (228)$$

$$\Sigma \Delta \lambda^{(1)} = \Delta \lambda^{(2)} \quad (229)$$

$$\mathbf{L}^T \Delta \lambda = \Delta \lambda^{(1)} \quad (230)$$

Solutions will be found sequentially, starting from the upper left hand entry of the lower triangular matrix \mathbf{L} . The form of the $(n + m + 1) \times (n + m + 1)$ matrix \mathbf{L} , equation (219), yields trivial solutions for the n terms

$$\Delta\lambda_O^{(2)} \equiv \Delta\lambda_{(2)}^i = -\rho_{p'}^i \equiv -\vec{\rho}_{p'}^O \quad (231)$$

for all $i \in \{1, \dots, n\}$. Inserting this result into expression (228) yields

$$\mathbf{L}^W \Delta\lambda_W^{(2)} = \mathbf{F} \vec{\rho}_{p'}^O - \vec{\rho}_{p'}^W \quad (232)$$

where, again, the labels W and O refer to the $m + 1$ dimensional space to which the weights and bias belong and the n dimensional space of input observations, respectively.

Once the first set of equations have been solved for $\Delta\lambda^{(2)}$, the second set of equations are readily solved to yield

$$\Delta\lambda^{(1)} = \Sigma^{-1} \Delta\lambda^{(2)} \quad (233)$$

Due to the form of \mathbf{L} , the final step will be computed by first solving

$$\mathbf{L}_W^T \Delta\lambda^W = \Delta\lambda_W^{(1)} \quad (234)$$

and then using this result to solve

$$\Delta\lambda^O = \Delta\lambda_O^{(1)} - \mathbf{F}^T \Delta\lambda_W^{(1)} \quad (235)$$

The $(m + 1) \times (m + 1)$ sub-block must be solved on a single machine and the solution propagated to all other nodes which can then independently calculate $\Delta\lambda^O$. The calculated $\Delta\lambda$ can then be used to calculate variations for the primal variables using equations (211), which intern can be used to calculate variations for the inequality Lagrange multipliers using equation (193).

The step length can then be calculated and the variables updated, ending the interior point method iteration. The next iteration begins by calculating the surrogate gap $\hat{\mu}$ by summing the central residuals and dividing by the total number of inequality constraints and multiplying by a factor $0 < \sigma < 1$ to yield an updated value $\mu = \sigma \hat{\mu}$ which is then used to calculate the next variable updates. This process continues until the residuals of all constraints and central path parameter μ fall below user defined thresholds or until a user defined maximum number of iterations are reached.

A. Standard IPM Algorithm

An outline of the standard algorithm is presented below.

- Set initial values for all $(\mathbf{z}, \hat{\mathbf{z}}, \bar{\mathbf{z}}, \mathbf{w}, \mathbf{s}, \hat{\mathbf{s}}, \bar{\mathbf{s}}, \mathbf{v}, \hat{\mathbf{v}}, \bar{\mathbf{v}}, \lambda)$.
- For each iteration, k , while k less than the maximum number of iterations and the stopping criteria are not satisfied:
 1. Calculate μ .
 2. Calculate stopping criteria.
 3. Construct \mathbf{M} and corresponding residuals.
 4. Solve for \mathbf{L}^W and \mathbf{S}_W^{-1} .
 5. Calculate values for the variation of the equality constraint Lagrange multipliers, $\Delta\lambda$.
 6. Using $\Delta\lambda$, calculate values for the variation of the primal variables, $\Delta\mathbf{q}$.
 7. Using $\Delta\mathbf{q}$, calculate values for the variation of the inequality constraint Lagrange multipliers, $\Delta\mathbf{s}, \Delta\hat{\mathbf{s}}, \Delta\bar{\mathbf{s}}, \Delta\mathbf{v}, \Delta\hat{\mathbf{v}}, \Delta\bar{\mathbf{v}}$.
 8. Find step length, γ .
 9. Update all variables:

$$(\mathbf{q}, \mathbf{s}, \hat{\mathbf{s}}, \bar{\mathbf{s}}, \mathbf{v}, \hat{\mathbf{v}}, \bar{\mathbf{v}}, \lambda)^{(k+1)} = (\mathbf{q}, \mathbf{s}, \hat{\mathbf{s}}, \bar{\mathbf{s}}, \mathbf{v}, \hat{\mathbf{v}}, \bar{\mathbf{v}}, \lambda)^{(k)} + \gamma (\Delta\mathbf{q}, \Delta\mathbf{s}, \Delta\hat{\mathbf{s}}, \Delta\bar{\mathbf{s}}, \Delta\mathbf{v}, \Delta\hat{\mathbf{v}}, \Delta\bar{\mathbf{v}}, \Delta\lambda)$$
- return values for $(\mathbf{q}, \mathbf{s}, \hat{\mathbf{s}}, \bar{\mathbf{s}}, \mathbf{v}, \hat{\mathbf{v}}, \bar{\mathbf{v}}, \lambda)$

VIII. MAPREDUCE INTERIOR POINT METHOD IMPLEMENTATION

A. Local Data and Calculations

Local variables are

$$\{z^i, \hat{z}^i, \bar{z}^i, s^i, \hat{s}^i, \bar{s}^i, v^i, \hat{v}^i, \bar{v}^i, \lambda^i\} \quad (236)$$

which are used, along with the global variables of section (VIII B) which are stored locally on each partition, to calculate the local residuals

$$\rho_p^i \equiv 2\bar{z}^i - \tau e^i - z^i + \hat{z}^i \quad (237)$$

$$\rho_{d'}^i \equiv \eta e_i - \lambda_i - \frac{\mu}{z^i} + \frac{\mu}{(\tau e^i - z^i)} \quad (238)$$

$$\hat{\rho}_{d'}^i \equiv \eta e_i + \lambda_i - \frac{\mu}{\hat{z}^i} + \frac{\mu}{(\tau e^i - \hat{z}^i)} \quad (239)$$

$$\bar{\rho}_{d'}^i \equiv -2y_i + 2\lambda_i - 2\lambda_a X_i^a + 2\lambda_0 e_i - \frac{\mu}{\bar{z}^i} + \frac{\mu}{(\tau e^i - \bar{z}^i)} \quad (240)$$

Using equation (213), the updated local primal residuals will be

$$\begin{aligned} \rho_{p'}^i &\equiv \rho_p^i + 2 \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}^i}^{ii}} \right) - \left(\frac{\rho_{d'}^i}{D_{z^i}^{ii}} \right) + \left(\frac{\hat{\rho}_{d'}^i}{D_{\hat{z}^i}^{ii}} \right) \\ &= 2\bar{z}^i - \tau e^i - z^i + \hat{z}^i + 2 \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}^i}^{ii}} \right) - \left(\frac{\rho_{d'}^i}{D_{z^i}^{ii}} \right) + \left(\frac{\hat{\rho}_{d'}^i}{D_{\hat{z}^i}^{ii}} \right) \\ &= 2\bar{z}^i - \tau e^i - z^i + \hat{z}^i \\ &\quad + 2 \left(\frac{\delta^{ij} \left(-2y_j + 2\lambda_j - 2\lambda_a X_j^a + 2\lambda_0 e_j - \frac{\mu}{\bar{z}^j} + \frac{\mu}{(\tau e^j - \bar{z}^j)} \right)}{\left(\frac{\bar{s}^i}{\bar{z}^i} + \frac{\bar{v}^i}{(\tau e^i - \bar{z}^i)} \right)} \right) \\ &\quad - \left(\frac{\eta e_i - \lambda_i - \frac{\mu}{z^i} + \frac{\mu}{(\tau e^i - z^i)}}{\left(\frac{s^i}{z^i} + \frac{v^i}{(\tau e^i - z^i)} \right)} \right) \\ &\quad + \left(\frac{\eta e_i + \lambda_i - \frac{\mu}{\hat{z}^i} + \frac{\mu}{(\tau e^i - \hat{z}^i)}}{\left(\frac{\hat{s}^i}{\hat{z}^i} + \frac{\hat{v}^i}{(\tau e^i - \hat{z}^i)} \right)} \right) \end{aligned} \quad (241)$$

B. Global Data and Calculations

Global variables are

$$\{w^\alpha, \lambda^\alpha\} \quad (242)$$

which, by imposing $w^\alpha = -\lambda^\alpha$, reduces to just the $m + 1$ weights w^α . In addition to w^α , there will be global terms associated with the current iteration and update, namely

$$\{\mu, \gamma, \sigma\} \quad (243)$$

corresponding to the central path parameterization value μ , the step length γ and central path reduction factor, σ , where γ and σ are needed to calculate μ and μ is used to update the dual residuals of bound variables. All global values will be stored locally on each partition during every iteration.

The residuals which require global sums of local data are

$$\rho_p^a \equiv w^a - X_i^a (2\bar{z}^i - e^i \tau) \quad (244)$$

$$\rho_p^0 \equiv e_i (2\bar{z}^i - \tau e^i) \quad (245)$$

and the updates

$$\rho_{p'}^a \equiv \rho_p^a + \rho_{d'}^a - 2X_i^a \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}}^{ii}} \right) = \rho_p^a - 2X_i^a \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}}^{ii}} \right) \quad (246)$$

$$\rho_{p'}^0 \equiv \rho_p^0 + \rho_{d'}^0 + 2e_i \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}}^{ii}} \right) = \rho_p^0 + 2e_i \left(\frac{\bar{\rho}_{d'}^i}{D_{\bar{z}}^{ii}} \right) \quad (247)$$

Combining the updated global primal residuals with the updated local primal residuals, equation (232) yields the first linear system to solve

$$[\mathbf{L}_W]^a \alpha \Delta \lambda_{(2)}^\alpha = 4X_i^a \left(\frac{\rho_{p'}^i S^{ii}}{D_{\bar{z}}^{ii}} \right) - \rho_{p'}^a \equiv -\rho_{p'}^a \quad (248)$$

$$[\mathbf{L}_W]^0 \alpha \Delta \lambda_{(2)}^\alpha = -4e_i \left(\frac{\rho_{p'}^i S^{ii}}{D_{\bar{z}}^{ii}} \right) - \rho_{p'}^0 \equiv -\rho_{p'}^0 \quad (249)$$

For the moment, each machine will calculate $\Delta \lambda_{(2)}^\alpha$ independently. Their answers will all agree up to machine precision.

In order to calculate \mathbf{L}_W and \mathbf{S}_W we will need to calculate global sums for the right hand side of equation (223). Using the definitions for the $(m+1) \times n$ matrix \mathbf{F} , $(n \times n)$ matrix \mathbf{S}_O and $(m+1) \times (m+1)$ matrix \mathbf{M}_W , the global sum will be given by

$$\mathbf{L}_W \mathbf{S}_W^{-1} \mathbf{L}_W^T = \mathbf{Q}_W + \mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T \quad (250)$$

for

$$\mathbf{Q}_W \equiv \begin{bmatrix} \mathbb{I}_{(m \times m)} & \mathbf{0}_{(m \times 1)} \\ \mathbf{0}_{(1 \times m)} & \mathbf{0}_{(1 \times 1)} \end{bmatrix} \quad (251)$$

The matrix \mathbf{Q}_W is constant.

On each partition, $r \in \{1, \dots, p\}$ for fixed positive integer p , the partial sum

$$\mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T = \sum_{r=1}^p \mathbf{F}_r \left(\frac{\mathbf{D}_{\bar{z}}^r - 4\mathbf{S}_O^r}{4[\mathbf{S}_O^r]^2} \right) \mathbf{F}_r^T \quad (252)$$

where each \mathbf{F}_r , $\mathbf{D}_{\bar{z}}^r$ and \mathbf{S}_O^r can be calculated using only local values which are accessible to each partition r .

All partition metadata will also be stored as global variables. This will include values such as the number of features, m , the total number of input observations, n , on all partitions and the local number of input observations, n_r , located on each partition r , where

$$n = \sum_{r=1}^p n_r \quad (253)$$

The partition metadata values n_r will be used to assign global index labels to all input observations. The partition metadata value n will be used to calculate μ . Additional data can be stored to aide in debugging.

Expanding each term in the right hand side of equation (252) using equation (222) yields the expression

$$\left[\mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T \right]_{(m+1) \times (m+1)} = 4 \begin{bmatrix} -\mathbf{X} & (m \times n) \\ \mathbf{e} & (1 \times n) \end{bmatrix} \begin{bmatrix} \frac{1}{\mathbf{D}_{\bar{z}}} - 4 \frac{\mathbf{S}_O}{(\mathbf{D}_{\bar{z}})^2} \end{bmatrix}_{(n \times n)} \begin{bmatrix} -\mathbf{X}^T & (n \times m) \\ \mathbf{e}^T & (n \times 1) \end{bmatrix} \quad (254)$$

$$\left[\mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T \right]_{ab} = 4X_a^i \left(\frac{1}{D_{\bar{z}}^{ij}} - 4S_O^{kl} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{ik} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{lj} \right) X_b^j = 4X_a^i \left[\frac{1}{\mathbf{D}_{\bar{z}}} - 4 \frac{\mathbf{S}_O}{(\mathbf{D}_{\bar{z}})^2} \right]_{ii} X_b^i \quad (255)$$

$$\left[\mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T \right]_{a0} = -4X_a^i \left(\frac{1}{D_{\bar{z}}^{ij}} - 4S_O^{kl} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{ik} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{lj} \right) e^j = -4X_a^i \left[\frac{1}{\mathbf{D}_{\bar{z}}} - 4 \frac{\mathbf{S}_O}{(\mathbf{D}_{\bar{z}})^2} \right]_{ii} e^i \quad (256)$$

$$\left[\mathbf{F} \left(\frac{\mathbf{D}_{\bar{z}} - 4\mathbf{S}_O}{4[\mathbf{S}_O]^2} \right) \mathbf{F}^T \right]_{00} = 4e^i \left(\frac{1}{D_{\bar{z}}^{ij}} - 4S_O^{kl} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{ik} \left[\frac{1}{\mathbf{D}_{\bar{z}}} \right]_{lj} \right) e^j = e^i \left[\frac{1}{\mathbf{D}_{\bar{z}}} - 4 \frac{\mathbf{S}_O}{(\mathbf{D}_{\bar{z}})^2} \right]_{ii} e^i \quad (257)$$

IX. MAPREDUCE IPM ALGORITHM

The MapReduce algorithm is presented in this section.

A. Prototype Framework Details

Prototype framework details for IPM algorithms:

- Current algorithms designed only for the situation where $m \ll n$.
- All Keys, both input and output, will be instances of the *PartitionTypeIndexKey* class.
- Instances of the InputValues class consists of a single target variable and an observation vector of length m .
- Instances of the IPMData class consist of vectors containing the Lagrange multipliers for all primal, dual and central constraints as well as the values for the corresponding residuals of these constraints. Instances of the IPMData class having *PartitionTypeIndexKey* with $typeId = GLOBAL$ will contain values for the $m + 1$ global variables w^α .
- Instances of the IPMData class having *PartitionTypeIndexKey* with $typeId = LOCAL$ will contain the values of all local variables associated to each of the n input observations.
- Instances of the IPMData class having *PartitionTypeIndexKey* with $typeId = MATRIX$ will contain $(m + 1) \times (m + 1)$ global matrix values.
- Instances of the IPMData class having *PartitionTypeIndexKey* with $typeId = PARTITION$ will contain partition metadata.
- Instances of the IPMData class having *PartitionTypeIndexKey* with $typeId = UPDATE$ will contain values for μ, γ, σ as well as the step number identifying the current iteration.
- Each $typeId$ has a corresponding *PARTIAL* type, used to label data passed between MapReduce jobs which has not yet been used to calculate each partition's corresponding non-partial data.
- The n local instances of the IPMData class are distributed approximately evenly amongst the p partitions.
- All *GLOBAL, MATRIX* and *PARTITION* data is duplicated on each partition, as each partition needs access to all data of these types.
- On each partition, any *PARTIAL* data for the types *GLOBAL, MATRIX, UPDATE* and *PARTITION* will be written to all partitions present.
- Map phases will not be guaranteed access to *GLOBAL, MATRIX, UPDATE* and *PARTITION* data and so will often simply be identity mappers, with computations happening in the reduce phase.
- The *PartitionTypeIndexPartitioner* class ensures that all data with the same $partitionId$ will be sent to the same reducer.
- The *PartitionTypeIndexKeyGroupComparator* and *PartitionTypeIndexKeyComparator* classes ensure that the data sent to any reducer will be ordered by the data type.
- Data types are defined as *int* values, with the value of each type defined such that ordering of data by type ensures that each reducer will be able to load and construct all data necessary to compute all data for subsequent data types.
- The *LOCAL* data type will always be sorted so that data of this type is read by the reducer only after all data types have been processed, allowing each reducer to process each of the n_r data of type *LOCAL* individually and then write the updated local data to context without having to store n_r values in memory.

B. Extract, Transform and Load Distributed Data

The *IPMSolverDriver* class will first extract and transform raw data:

1. ParseTextMR:

This MapReduce job reads in raw data, loads input observation and target data into *InputValue* instances, distributes the data across the available number of partitions and calculates partial partition metadata for each partition needed by the next MapReduce job.

- Configuration:

```
job.setInputFormatClass(TextInputFormat.class);
job.setMapperClass(ParseMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(ParseReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

- ParseMapper:

```
setup();
map();
cleanup();
```

- ParseReducer:

```
setup();
map();
cleanup();
```

2. InitGlobalIndexMR:

This MapReduce job is the first part of two sequential jobs which together supply sequential global index values to all n unique input data starting with index 1 on partition 1 and ending with index n on partition p .

- Configuration:

```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(InitGlobalIndexReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

- IdentityMapper:

```
setup(): super.setup()
map(): Each input key,value pair is read in and then written to context.
cleanup(): super.cleanup()
```

- InitGlobalIndexReducer:

```
setup();
map();
cleanup();
```

3. GlobalIndexMR:

This MapReduce job is the second part of two sequential jobs which together supply sequential global index values to all n unique input data starting with index 1 on partition 1 and ending with index n on partition p .

- Configuration:

```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(GlobalIndexReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

- IdentityMapper:

```
setup(): super.setup()
map(): Each input key,value pair is read in and then written to context.
cleanup(): super.cleanup()
```

- GlobalIndexReducer:

```
setup():
map():
cleanup():
```

C. Supply Necessary Initial Data

The *IPMSolverDriver* will then calculate initial data:

1. InitialDataMR:

This MapReduce job supplies initial values for all variables needed by the interior point method. The output data from this class will be the same in size and scope as the output of the last MapReduce job called during each iteration of the system.

- Configuration:

```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(InitialDataMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(IPMData.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(InitialDataReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(IPMData.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

- InitialDataMapper:

```
setup():
map(): Set initial values for all local variables ( $\mathbf{z}, \hat{\mathbf{z}}, \bar{\mathbf{z}}, \mathbf{s}, \hat{\mathbf{s}}, \bar{\mathbf{s}}, \mathbf{v}, \hat{\mathbf{v}}, \bar{\mathbf{v}}, \lambda$ ) based on user supplied initial  $\mu$ .
```

map(): Calculate local residuals and partial global residuals for all local data. (right now, set these equal to zero cause we're not using stopping criteria other than limiting the number of iterations to some user supplied maximum step number)

map(): Write to context.

cleanup():

- InitialDataReducer:

setup():

reduce(): Read all partial global residuals and sum. Write the local data to context.

cleanup(): Set initial values for all global variables \mathbf{w} and write to context. global value μ and write to context.

D. Solve Using the Interior Point Method

All Mapper and Reducer classes detailed here extend *IterationMapper* and *IterationReducer* abstract classes respectively. These superclasses are used to implement consistent logging, data sanity checks, hadoop counters and status updates, identity mapper and reducer methods for all data types and sanity checks on key types and ordering.

The *IPMSolverDriver* will then start with an initial step number, $k = 0$, then iterate over the sequence of MapReduce jobs for each iteration step number $k = k + 1$ until user supplied stopping criteria are met:

1. StepOneMR:

This MapReduce job sums the partial values for global *UPDATE* value used to construct the surrogate gap $\hat{\mu}$, calculates σ for the current step, updates μ for the current step, calculates the partial values for the global *MATRIX* quantities of equation (252). The current μ is used to construct the local reduced dual residuals $\rho_{d'}^i$, which intern are used to construct the partial sums for the global *GLOBAL* quantities of equation (246).

- Configuration:

```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(StepOneMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(IPMData.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(StepOneReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(IPMData.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

- StepOneMapper:

setup():

map():

cleanup():

- StepOneReducer:

setup():

map():

cleanup():

2. StepTwoMR:

This MapReduce job sums the partial values for the global values of type *GLOBAL* and *MATRIX* then calculates the *MATRIX* quantities \mathbf{L}_W and \mathbf{S}_W . Using these values, the variations on each individual partition, r , for the $n_r + m + 1$ equality Lagrange multipliers, λ_r , which are accessible in the reduce phase are calculated. These values are used to construct the variations for the all $9n_r + m + 1$ values accessible to the reducer. The available variations are used to calculate partial values for the global *UPDATE* quantity γ determining the step length. The partial *GLOBAL* quantity γ is written to out to all available partitions. All calculated variations are written out to the current partition along with all local and stored global data.

- Configuration:

```

job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(StepTwoMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(IPMData.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(StepTwoReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(IPMData.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

```

- StepTwoMapper:

```

setup();
map();
cleanup();

```

- StepTwoReducer:

```

setup();
map();
cleanup();

```

3. StepThreeMR:

This MapReduce job calculates the global value for the *UPDATE* quantity γ , which determines the step length. Using this value along with the variations for all local and stored global quantities computed by the current partition in the previous MapReduce job, the values of all local and stored global values are updated. The updated values are then used to calculate the partial global *UPDATE* quantity used to calculate the surrogate gap $\hat{\mu}$. All partial global *GLOBAL* and *UPDATE* quantities necessary to test stopping criteria are calculated. All local and stored global quantities are written out to the current partition and all partial global quantities written to all available partitions.

- Configuration:

```

job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(StepThreeMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(IPMData.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(StepThreeReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(IPMData.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

```

- StepThreeMapper:

```

setup();
map();
cleanup();

```

- StepThreeReducer:

```

setup();
map();
cleanup();

```

Currently, the *IPMSolverDriver* will terminate the iteration loop once a user supplied maximum number of iterations are reached. Other stopping criteria are not currently used.

X. MAPREDUCE LEAST SQUARES ALGORITHM

The MapReduce algorithm for linear least squares regression is presented in this section.

A. Extract, Transform and Load Distributed Data

The *BasicLeastSquaresDriver* class will first extract and transform raw data:

1. ParseTextMR:

This MapReduce job reads in raw data, loads input observation and target data into *InputValue* instances, distributes the data across the available number of partitions and calculates partial partition metadata for each partition needed by the next MapReduce job.

- Configuration:

```

job.setInputFormatClass(TextInputFormat.class);
job.setMapperClass(ParseMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(ParseReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

```

- ParseMapper:

```

setup();
map();
cleanup();

```

- ParseReducer:

```

setup();
map();
cleanup();

```

2. InitGlobalIndexMR:

This MapReduce job is the first part of two sequential jobs which together supply sequential global index values to all n unique input data starting with index 1 on partition 1 and ending with index n on partition p .

- Configuration:

```

job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(InitGlobalIndexReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

```

- IdentityMapper:
 - setup(): super.setup()
 - map(): Each input key,value pair is read in and then written to context.
 - cleanup(): super.cleanup()
- InitGlobalIndexReducer:
 - setup():
 - map():
 - cleanup():

3. GlobalIndexMR:

This MapReduce job is the second part of two sequential jobs which together supply sequential global index values to all n unique input data starting with index 1 on partition 1 and ending with index n on partition p .

- Configuration:


```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(GlobalIndexReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```
- IdentityMapper:
 - setup(): super.setup()
 - map(): Each input key,value pair is read in and then written to context.
 - cleanup(): super.cleanup()
- GlobalIndexReducer:
 - setup():
 - map():
 - cleanup():

B. Solve Using Linear Least Squares

The *BasicLeastSquaresDriver* class then solves the linear least squares regression problem in two steps.

1. BasicStepOneMR:

This MapReduce job constructs all necessary partial matrix components using the locally available input data. The output is sent to written to all available partitions.

- Configuration:


```
job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(CurrentReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

2. BasicStepTwoMR:

This MapReduce job sorts data appropriately, mapping relevant partial data quantities to all partitions. The reducer then constructs all necessary global matrix components using the partial matrix components from all cluster partitions to construct a dense $m + 1$ by $m + 1$ square symmetric positive definite matrix which is then inverted to solve for the desired $m + 1$ weights. Each reducer writes out the global solution for the weights and all input data.

- Configuration:

```

job.setInputFormatClass(SequenceFileInputFormat.class);
job.setMapperClass(IdentityMapper.class);
job.setMapOutputKeyClass(PartitionTypeIndexKey.class);
job.setMapOutputValueClass(InputValue.class);
job.setPartitionerClass(PartitionTypeIndexKeyPartitioner.class);
job.setSortComparatorClass(PartitionTypeIndexKeyComparator.class);
job.setGroupingComparatorClass(PartitionTypeIndexKeyGroupComparator.class);
job.setNumReduceTasks(numReducers);
job.setReducerClass(CurrentReducer.class);
job.setOutputKeyClass(PartitionTypeIndexKey.class);
job.setOutputValueClass(InputValue.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

```

There is no iteration in this implementation, the formulation of the problem means that the desired weights are solved for in two steps. The calculated weights can be used to calculate error and quantile estimates.

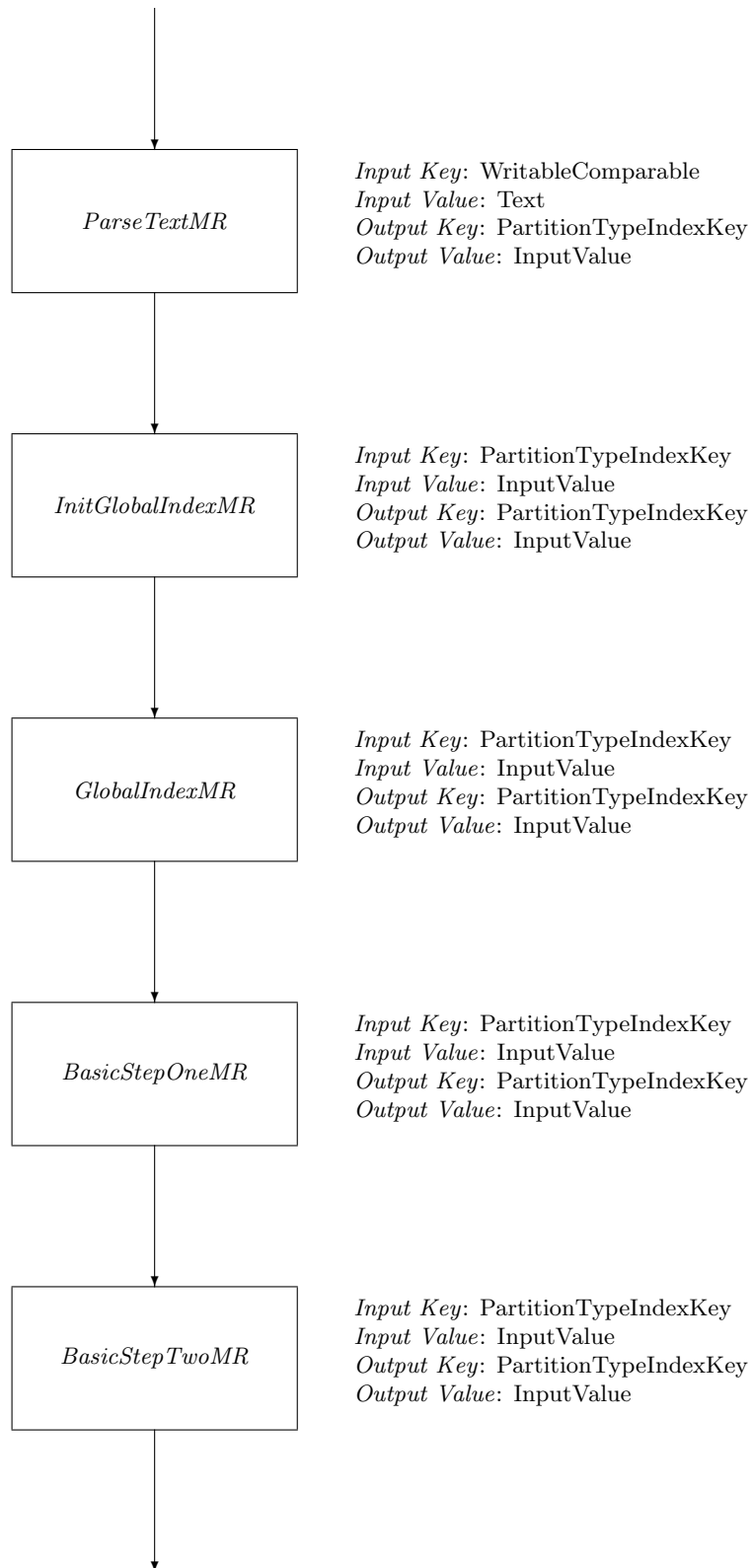


FIG. 1: Basic Linear Regression Workflow