Linear Regression

By Michael Steinman

January 26, 2018

1 Two Dimensions

1.1 Closed Form (calculating the derivative)

In two dimensions, we wish to fit a line (in the x-y plane) to the data. The equation of the line is given by $\hat{y} = \omega_0 + \omega_1 x$. Thus, the line is a function of the two parameters ω_0 and ω_1. We wish to find the parameters that minimize the error. So, our first step is to define a metric to define the error. For now, let us use the residual sum of squares defined by

$$\begin{aligned} rss &= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{N} [y_i - (\omega_0 + \omega_1 x_i)]^2 \end{aligned}$$

where $\hat{y}_i$ is the predicted value of x_i, y_i is the true y value, and we sum over all data points.
In order to minimize the rss, we compute the gradient

$$\nabla rss(\omega_0, \omega_1) = \begin{bmatrix} -2\sum_{i=1}^{N} [y_i - (\omega_0 + \omega_1 x_i)] \\ -2\sum_{i=1}^{N} x_i [y_i - (\omega_0 + \omega_1 x_i)] \end{bmatrix}$$

then set it to zero

$$\begin{aligned} 0 &= \sum_{i=1}^{N} [y_i - (\omega_0 + \omega_1 x_i)] \\ 0 &= \sum_{i=1}^{N} x_i [y_i - (\omega_0 + \omega_1 x_i)] \end{aligned}$$

Solving for $\omega_0$and ω_1,

$$\begin{aligned} \omega_0 &= \frac{1}{N}\sum_{i=1}^{N} y_i - \frac{\omega_1}{N}\sum_{i=1}^{N} x_i \\ \omega_1 &= \frac{\sum_{i=1}^{N} x_i y_i - \omega_0 \sum_{i=1}^{N} x_i}{\sum_{i=1}^{N} x_i^2} \end{aligned}$$

1.2 Numerical Method

Imagine a function, f, of variable x. The derivative, $f'(x)$ will be positive to the right of a minimum and negative to the left. If a point x_n is to the right of the minimum, we may chose another value of x by subtracting a small amount. Suppose we subtracted $\eta f'(x_n)$ from x_n, where $\eta$is a step size. We see that if x_n is to the right of the minimum, $f'(x_n)$ will be positive, and the new point $x_{n+1} \equiv x_n - \eta f'(x_n)$ will be smaller than x_n (i.e. x_{n+1}will have moved toward the minimum). Similarly, if x_n was to the left of the minimum, the derivative would be negative and $-\eta f'(x_n)$ would be a positive term, making x_{n+1} move to the right. Also notice that as x_n approaches the minimum, the derivative approaches zero, making the step size smaller and smaller, insuring we do not over shoot the minimum.

We may extend this method to a function of two variables and replace the derivative with the gradient. Thus, we view the rss as a function of two variables, ω_0 and ω_1, compute the gradient and use the following algorithm to find the parameters associated with the minimum rss value:

$$\omega_0^{(n+1)} = \omega_0^{(n)} - \eta \frac{\partial rss\left(\omega_0^{(n)}, \omega_1\right)}{\partial \omega_0}$$
$$\omega_1^{(n+1)} = \omega_1^{(n)} - \eta \frac{\partial rss\left(\omega_0, \omega_1^{(n)}\right)}{\partial \omega_1}$$

Where the partial derivatives were computed in the previous section. This algorithm is repeated until the absolute value of each of the partial derivatives is less than a prescribed value, ϵ. Alternatively, we may require the absolute value of gradient to be less than ϵ. In other words

$$\sqrt{\left(\frac{\partial rss}{\partial \omega_0}\right)^2 + \left(\frac{\partial rss}{\partial \omega_1}\right)^2} < \epsilon$$

or

$$\langle \nabla rss, \nabla rss \rangle^{1/2} < \epsilon$$

A clever guess for the initial values of ω_0 and ω_1 may be the intercept and slope of the line connecting any two points.

2 Multi Dimensional

One example might be polynomial regression\

$$y_i(x_i) = \omega_0 + \omega_1 x_i + \omega_2 x_i^2 + \cdots + \omega_p x_i^p$$

We may graph this in two dimensions (y as a function of x), but be have multiple coefficients that need to be solved.

Another way to view the multi dimensional approach is to view our target (y) as a function of multiple features. In this case, we view x_i as a vector, $\boldsymbol{x}_i$. $\boldsymbol{x}_i$ represents a single data point that has multiple features. For example, y_i might represent sale price of a particular house, denoted by i, and $\boldsymbol{x}_i$ is a vector comprised of the house's features, such as living space, number of bedrooms, neighborhood, etc. In this case, we view the function y as a hyper plane in an n dimensional space, where n is the number of features. Notation: $\boldsymbol{x}_i[j]$ represents the jth element of the vector $\boldsymbol{x}_i$, and the i index indicates a particular data point.

We genearalize as follows. Let $\boldsymbol{h}$ be a vector of D functions. $\boldsymbol{h}$ will be a function of the particular data point x_i. We view $\boldsymbol{\omega} = [\omega_0, \omega_1, \ldots, \omega_D]$ as the parameters we must discover. Our predicted y value is given by

$$\begin{aligned}\hat{y}_i &= \omega_0 h_0(\boldsymbol{x}_i) + \omega_1 h_1(\boldsymbol{x}_i) + \ldots \omega_D h_D(\boldsymbol{x}_i) \\ &= \sum_{j=0}^{D} \omega_j h_j(\boldsymbol{x}_i)\end{aligned}$$

In this case, we view $\boldsymbol{h}$ as a function of the features. Often, we want to define our function $\hat{y}$ in terms of the features themselves,

$$\hat{y}_i = \omega_0 + \omega_1 \boldsymbol{x}_i[1] + \ldots \omega_d \boldsymbol{x}_i[d]$$

where we are assuming$\boldsymbol{x}_i[0] = 1$ and let d be the dimension of $\boldsymbol{x}$.

In vector notation, we can write our predicted value as the inner product of the two vectors

$$\hat{y}_i = \langle \boldsymbol{\omega}, \boldsymbol{h}(\boldsymbol{x}_i) \rangle$$

We may generalize to all samples with matrix notation. Let $\hat{\boldsymbol{y}}$ be a vector, where the ith component is $\hat{y}_i$. Let $\boldsymbol{H}$ be an N x d matrix where the ith row is $\boldsymbol{x}_i$. Then

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{\omega}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & h_1(\boldsymbol{x}_0) & \ldots & h_d(\boldsymbol{x}_0) \\ 1 & h_1(\boldsymbol{x}_1) & \ldots & h_d(\boldsymbol{x}_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & h_1(\boldsymbol{x}_N) & \ldots & h_d(\boldsymbol{x}_N) \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_N \end{bmatrix}$$

2.1 Residual Sum of Squares

In vector notation, the residual sum of squares is given by

$$\begin{aligned} RSS(\boldsymbol{\omega}) &= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{N} (y_i - \langle \boldsymbol{h}(\boldsymbol{x}_i), \boldsymbol{\omega} \rangle)^2 \end{aligned}$$

We may replace the summation notation with matrix notation

$$\begin{aligned} RSS(\boldsymbol{\omega}) &= \langle \boldsymbol{y} - \hat{\boldsymbol{y}}, \boldsymbol{y} - \hat{\boldsymbol{y}} \rangle \\ &= \langle \boldsymbol{y} - \boldsymbol{H}\boldsymbol{\omega}, \boldsymbol{y} - \boldsymbol{H}\boldsymbol{\omega} \rangle \end{aligned}$$

where $\boldsymbol{y}$ is a vector of actual y values.

Next, we compute the gradient of RSS

$$\begin{aligned}
\nabla RSS(\boldsymbol{\omega}) &= \begin{bmatrix} \frac{d}{d\omega_0}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 \\ \frac{d}{d\omega_1}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 \\ \vdots \\ \frac{d}{d\omega_D}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 \end{bmatrix} \\
&= \begin{bmatrix} -2h_0\sum_{i=1}^{N}(y_i-\hat{y}_i) \\ -2h_1\sum_{i=1}^{N}(y_i-\hat{y}_i) \\ \vdots \\ -2h_D\sum_{i=1}^{N}(y_i-\hat{y}_i) \end{bmatrix} \\
&= -2\begin{bmatrix} h_0(x_0) & h_0(x_1) & \dots & h_0(x_N) \\ h_1(x_0) & h_1(x_1) & \dots & h_1(x_N) \\ \vdots & \vdots & \dots & \vdots \\ h_D(x_0) & h_D(x_1) & \dots & h_D(x_N) \end{bmatrix}\begin{bmatrix} y_0-\hat{y}_0 \\ y_1-\hat{y}_1 \\ \vdots \\ y_N-\hat{y}_N \end{bmatrix} \\
&= -2\boldsymbol{H}^T(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{\omega})
\end{aligned}$$

Then we set the gradient equal to zero and solve for $\boldsymbol{\omega}$

$$\begin{aligned}
0 &= -2\boldsymbol{H}^T\boldsymbol{y} + 2\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\omega} \\
\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\omega} &= \boldsymbol{H}^T\boldsymbol{y} \\
\boldsymbol{\omega} &= (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}
\end{aligned}$$

In most cases, if $N > D$, the inverse matrix, seen above, exists. The complexity of the inverse is proportional to D^3. Thus, if we have lots of features, computing the inverse may be too computationally costly.

2.2 Gradient Descent (Numerical Method)

We may use the gradient descent method described earlier to estimate $\boldsymbol{\omega}$. While not converged, apply the algorithm

$$\begin{aligned}
\boldsymbol{\omega}^{(n+1)} &= \boldsymbol{\omega}^n - \eta\nabla RSS(\boldsymbol{\omega}^n) \\
&= \boldsymbol{\omega}^n + 2\boldsymbol{H}^T(\boldsymbol{y}-\boldsymbol{H}\boldsymbol{\omega}^n)
\end{aligned}$$

The convergence criteria would be

$$\langle \nabla RSS, \nabla RSS\rangle^{1/2} < \epsilon$$

for some ϵ. The choice of the initial value of $\boldsymbol{\omega}$ may be random, zero, or some other clever guess.

2.3 Note

A linear regression problem is of the form

$$y = \omega_0 h_0(x) + \omega_1 h_1(x) + \dots \omega_D h_D(x)$$

It is called linear because it is linear with respect to the parameters $\boldsymbol{\omega}$, or in other words, it is a linear combination of the features. However, the features may be nonlinear, themselves. For instance, $h_j(x) = x^2$ would be acceptable.

3 Ridge Regression

3.1 Penalize All Coefficients

As an example to motivate the discussion, recall the linear regression model of an unusually high order polynomial, to data that can be fit well with a linear or low order polynomial. This is the classic example of over fitting data. We notice with a high order polynomial fit, say degree 15, that the polynomial fits the training data very well, but the undulations of the curve that make the fit very well, don't generalize well to new data.

Let us ask the question "can we determine a model is overfit by looking at the coefficients?" The answer is yes. First, we must make sure we rescale our data so that our xvalues vary from 0 to 1. After rescaling, we notice that a high order polynomial model must have very large coefficients on the high order polynomial terms. For example, x^{15} will map a number in $[0, 1)$ to an extremely small number. Thus, the coefficient must become extremely large for that term to be non negligible.

Therefore, after rescaling, the indication of an overfit line will be extremely large coefficients. In order to minimize the overfitting, we could "devalue" the terms with large coefficients (If we were only talking about a polynomial fit, we could just eliminate the higher order terms, but in a more general sense, if we have a large number of features that we want to include, we would have to resort to devaluing the terms that are contributing to the overfitting).

In order to devalue the terms with large coefficients, we redefine our cost function (scoring metric). With "normal" linear regression, our cost function was the residual sum of squares (RSS), where low values indicate a good fit. In order to devalue the overfitting terms, we redefine our cost function to be equal to RSS plus a measure of the magnitude of coefficients. We have two reasonable options for the measure of magnitudes of the coefficients: the L1 norm (sum of magnitudes) or the L2 norm squared (the L2 norm is the square root of the sum of squares, so the L2 squared is the sum of squares).

In Ridge Regression, we choose the L2 norm squared (Lasso uses the L1 squared). Thus, we define the cost function to be the RSS plus a constant (alpha) times the L2 squared. We use the constant alpha to increase or decrease the effect of devaluing term (L2 squared). Notice that as alpha goes to zero, the model approaches the "normal" linear regression (which may be overfit). On the other hand, letting alpha approach infinity, forces the coefficients to approach zero in order to minimize the cost function since, although the RSS may be large, it is bounded, whereas the devaluing term is not. So to minimize the unbounded term, the coefficients must approach zero.

We define the cost function $C(\boldsymbol{\omega}; \alpha)$ as follows

$$C(\boldsymbol{\omega}; \alpha) = RSS(\boldsymbol{\omega}) + \alpha ||\boldsymbol{\omega}||^2$$

where the norm is the L2 norm and α is a perameter.

In order to find $\boldsymbol{\omega}$ that minimizes the cost function, whether we find a closed form solution or use the gradient descent algorithm, we first calculate the gradient of the function.

$$\begin{aligned} \nabla C &= \nabla RSS + \alpha \nabla (||\boldsymbol{\omega}||^2) \\ &= -2\boldsymbol{H}^T(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\omega}) + 2\alpha\boldsymbol{\omega} \end{aligned}$$

3.1.1 Ridge Closed Form Solution

Setting the gradient to zero, we may find a closed form solution to $\boldsymbol{\omega}$as follow

$$\begin{aligned} 0 &= -2\boldsymbol{H}^T(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\omega}) + 2\alpha\boldsymbol{\omega} \\ 0 &= -\boldsymbol{H}^T\boldsymbol{y} + \boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\omega} + \alpha\boldsymbol{\omega} \\ \boldsymbol{H}^T\boldsymbol{y} &= \left(\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}\right)\boldsymbol{\omega} \\ \boldsymbol{\omega} &= \left(\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}\right)^{-1}\boldsymbol{H}^T\boldsymbol{y} \end{aligned}$$

Note that finding the inverse matrix can be computationally prohibitive for high dimensions. The computations involved is on the order of $\mathcal{O}(D^3)$.

Another important note is that $\left(\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}\right)$ is more likely to be invertible than $\left(\boldsymbol{H}^T\boldsymbol{H}\right)$. Thus, ridge regression may be preferrable over ordinary linear regression if $\left(\boldsymbol{H}^T\boldsymbol{H}\right)$ is not invertible.

3.1.2 Ridge Gradient Descent

The gradient descent method for ridge regression is given by the algorithm

$$\begin{aligned}\boldsymbol{\omega}^{(n+1)} &= \boldsymbol{\omega}^n - \eta\nabla C(\boldsymbol{\omega}^n) \\ &= \boldsymbol{\omega}^n - \eta(-2\boldsymbol{H}^T(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\omega}) + 2\alpha\boldsymbol{\omega})\end{aligned}$$

3.2 Don't Penalize the Intercept

In the method described above, we added a term to the cost function that penalizes all coefficients (not all to the same degree) including the intercept. However, we realize that we have no need to penalize the intercept. So, let us develop a method that penalizes all coefficients except the intercept.

Conceptually, we separate the $\boldsymbol{\omega}$ vector into two parts, ω_0 and $\boldsymbol{\omega}_{rest}$, where

$$\boldsymbol{\omega}_{rest} = \begin{bmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_D \end{bmatrix}$$

Then we have for the cost function

$$Cost(\omega_0, \boldsymbol{\omega}_{rest}) = RSS(\omega_0, \boldsymbol{\omega}_{rest}) + \alpha||\omega_{rest}||_2^2$$

Computing the gradient, we obtain

$$\begin{aligned}\nabla C = &\nabla RSS + \alpha\nabla\left(||\omega_{rest}||_2^2\right) \\ &- 2\boldsymbol{H}^T(y - \boldsymbol{H}\boldsymbol{\omega}) + 2\alpha\boldsymbol{\omega}_{rest}\end{aligned}$$

3.2.1 Closed Form Solution

The closed form solution of the minimizers is similar to before, but with a slight modification of the identity matrix

$$\boldsymbol{\omega} = \left(\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}_{mod}\right)^{-1}\boldsymbol{H}^T\boldsymbol{y}$$

where

$$\boldsymbol{I}_{mod} \equiv \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.2 Gradient Descent without Penalizing Intercept

The gradient descent algorithm is as follows

$$\boldsymbol{\omega}^{(n+1)} = \boldsymbol{\omega}^{(n)} - \eta\left(-2\boldsymbol{H}^T(y - \boldsymbol{H}\boldsymbol{\omega}) + 2\alpha\boldsymbol{I}_{mod}\boldsymbol{\omega}^{(n)}\right)$$

4 Lasso

Lasso regression is similar to ridge regression since it incorporates a penalty for large coefficients. The difference is that lasso uses the L1 norm instead of the L2 norm used in ridge. The effect the L1 norm has is that increasing the penalty strength parameter α, causes certain coefficients to become zero. The benefit of this is that it reduces dimensionality and thus makes the model easier to interpret, and less costly computationally.

Since we do not wish to penalize the constant coefficient, let us define $\boldsymbol{w}^*$ to represent all w_j except for w_0. Then the lasso cost function is given by

$$C_{lasso}(\boldsymbol{\omega}; \alpha) = RSS(\boldsymbol{\omega}) + \alpha||\boldsymbol{\omega}^*||_1$$

where the L1 norm (taxi cab norm) is given by

$$||\boldsymbol{\omega}^*|| = |\omega_1| + |\omega_2| + \cdots + |\omega_n|$$

4.1 Coordinate Descent

Unfortunately, this cost function is not differentiable for any $\omega = 0$. Thus, we cannot use the gradient method to find the minimizer. However, due to the "separable" nature of the cost function, we may use a coordinate descent algorithm. The idea of coordinate descent is to begin with an estimate for the coordinates. Then we choose a particular coordinate, call it j, to minimize. We hold all other coordinates constant and find the minimum value of the coordinate j (given the values of all other coordinates) by setting the partial derivative of the cost function with respect to the jth coordinate equal to zero. Then we update the coordinates so that the jth coordinate has the value just calculated. Then we choose a different coordinate and repeat the procedure.

This method is easily viewed geometrically in two dimensions for the cost function equaling RSS or the cost for ridge regression. In both cases, level curves of the cost function form elipses. The greater the area enclosed by the level curves, the greater the cost function.

Suppose we have two coordinates, x and y. We begin by assigning values to the coordinates (randomly or intelligently), call them x_0 and y_0. The first step in the algorithm is to hold one coordinate constant, say the x coordinate. Along the line $x = x_0$, level curves of the cost function will either intersect the line in two places (large elipses), nowhere (for elipses to small to intersect), or at a single point. The larger the elipse, the further the distance between the points of intersection with the line $x = x_0$. The elipse that intersects the line $x = x_0$ at a single point is the smallest ellipse (smallest cost) containing that particular value of x. Next, we notice the y value of the point of intersection and denote it y_1. y_1 is the value of y that minimizes the cost function when $x = x_0$.

In the next step, we hold y constant and view the line $y = y_0$. Then we search for the elipse that intersects this line at a single point. The x value of this point will be denoted x_1. Repeating this process, we update our coordinates, one coordinate at a time and as many times as we wish.

4.1.1 Subgradient

In the coordinate descent algorithm, we hold all coordinates constant except for one: the jth. Then we calculate the partial derivative of the cost function with respect to the jth component. We can easily differentiate the lasso cost, with respect to

the jth component, everywhere except at $\omega_j = 0 \; \forall j \in [0, D]$ since $|\omega_j|$ is not differentiable here. To compensate for this lack of differentiability, let us approximate the absolute value function as

$$|\omega_j| = \begin{cases} \omega_j & \text{for } \omega_j > \epsilon \\ g(\omega_j) & \text{for } -\epsilon < \omega_j < \epsilon \\ -\omega_j & \text{for } \omega_j < -\epsilon \end{cases}$$

for some $\epsilon > 0$ and where $g(\omega_j)$ is defined such that $|\omega_j|$ is continuous and differentiable and such that $g^{prime}(0) = 0$. We may take ϵ as small as we wish. Since we haven't defined an explicit formula for g, we do not know its derivatives but we know that for $x \in [-\epsilon, \epsilon]$, $g'(x) \in [-1, 1]$.

In order to derive the derivative of the lasso cost function, let's start with the RSS term:

$$\begin{aligned} \frac{\partial rss}{\partial w_j} &= -2 \sum_{i=1}^{N} h_j(x_i) \left(y_i - \sum_{k=1}^{D} w_k h_k(x_i) \right) \\ &= -2 \sum_{i=1}^{N} h_j(x_i) \left(y_i - \sum_{k \neq j}^{D} w_k h_k(x_i) - w_j h_j(x_i) \right) \\ &= -2 \sum_{i=1}^{N} h_j(x_i) \left(y_i - \sum_{k \neq j}^{D} w_k h_k(x_i) \right) + 2 w_j \sum_{i=1}^{N} h_j(x_i)^2 \end{aligned}$$

If we define

$$\begin{aligned} \rho_j &\equiv \sum_{i=1}^{N} h_j(x_i) \left(y_i - \sum_{k \neq j}^{D} w_k h_k(x_i) \right) \\ z_j &\equiv \sum_{i=1}^{N} h_j(x_i)^2 \end{aligned}$$

we can write the partial derivative as

$$\frac{\partial rss}{\partial w_j} = -2\rho_j + 2 w_j z_j$$

Before we write the partial derivative of the L1 penalty term, let us specify the size of ϵ. If $w_j < 0$, there exists $\epsilon > 0$ such that $w_j < -\epsilon < 0$ and such that $0 < \epsilon < w_j$ whenever $0 < w_j$. Thus we may write our absolute value function as

$$|w_j| = \begin{cases} w_j & \text{for } w_j > 0 \\ g(w_j) & w_j = 0 \\ -w_j & \text{for } w_j < 0 \end{cases}$$

Then the partial derivative becomes

$$\frac{\partial \alpha |w_j|}{\partial w_j} = \begin{cases} -\alpha & \text{for } w_j < 0 \\ [-\alpha, \alpha] & \text{for } w_j = 0 \\ \alpha & \text{for } w_j > 0 \end{cases}$$

This is not a well defined function since the slope is ambiguous for $w_j = 0$, however, we will soon see that we do not need to know the slope inside the epsilon neighborhood of zero.

The partial derivative for the cost function (for all w_j except for w_0) becomes

$$\frac{\partial C_{lasso}}{\partial w_j} = -2\rho_j + 2z_j w_j + \begin{cases} -\alpha & \text{for } w_j < 0 \\ x \in [-\alpha, \alpha] & \text{for } w_j = 0 \\ \alpha & \text{for } w_j > 0 \end{cases}$$
$$= \begin{cases} -2\rho_j + 2z_j w_j - \alpha & \text{for } w_j < 0 \\ x \in [-2\rho_j - \alpha, -2\rho_j + \alpha] & \text{for } w_j = 0 \\ -2\rho_j + 2z_j w_j + \alpha & \text{for } w_j > 0 \end{cases}$$

Although our derivative seems ambiguous for the $w_j = 0$ case, we indeed have enough information to define our algorithm. Let us set our (ambiguously defined) derivative equal to zero and then attempt to solve for w_j.

$$0 = \begin{cases} -2\rho_j + 2z_j w_j - \alpha & \text{for } w_j < 0 \\ x \in [-2\rho_j - \alpha, -2\rho_j + \alpha] & \text{for } w_j = 0 \\ -2\rho_j + 2z_j w_j + \alpha & \text{for } w_j > 0 \end{cases}$$

At first glance, this seems impossible to solve. If we don't know the sign of w_j, how do we know which equation to solve. The answer lies in the fact that ρ_j is independent of w_j and α is known a priori. Thus we can determine the sign of w_j before solving for it explicitly. Each of the three possibilities of the derivative have there own implications.

In the case that $w_j < 0$, w_j is given by the solution to the equation $-2\rho_j + 2z_j w_j - \alpha = 0$. Solving for w_j, we get

$$w_j = \frac{\rho_j + \frac{\alpha}{2}}{z_j}$$

Now, if $\rho_j \geq -\frac{\alpha}{2}$, then $w_j \geq 0$, contradicting our hypothesis. Thus,

$$w_j < 0 \implies \rho_j < -\frac{\alpha}{2} \tag{1}$$

In the case that $w_j = 0$, we have $-2\rho_j - \alpha \leq 0 \leq -2\rho_j + \alpha$. Thus,

$$w_j = 0 \implies -\frac{\alpha}{2} \leq \rho_j \leq \frac{\alpha}{2} \tag{2}$$

Lastly, if it is the case that $w_j > 0$, then

$$w_j = \frac{\rho_j - \frac{\alpha}{2}}{z_j}$$

If $\rho_j \leq \frac{\alpha}{2}$ then $w_j \leq 0$, contradicting our hypothesis. Thus

$$w_j > 0 \implies \rho_j > \frac{\alpha}{2} \tag{3}$$

Let us complete our logic. We always may calculate ρ_j, which is independent of w_j. If $\rho_j < -\frac{\alpha}{2}$ then the hypothesis that $w_j = 0$ would contradict implication (2) and the hypothesis that $w_j > 0$ would contradict implication (3) since $\rho_j < -\frac{\alpha}{2} < \frac{\alpha}{2}$. Thus, we must conclude $\rho_j < -\frac{\alpha}{2} \implies w_j < 0$. In a similar manner, we can show that the reverse implications of implications (1-3) are true. Specifically,

$$\rho_j < -\frac{\alpha}{2} \implies w_j < 0$$
$$\rho_j > \frac{\alpha}{2} \implies w_j > 0$$
$$-\frac{\alpha}{2} \leq \rho_j \leq \frac{\alpha}{2} \implies w_j = 0$$

Therefore, we can solve w_j given knowledge of the above inequalities. Specifically,

$$\rho_j < -\frac{\alpha}{2} \implies w_j = \frac{\rho_j + \frac{\alpha}{2}}{z_j}$$
$$\rho_j > \frac{\alpha}{2} \implies w_j = \frac{\rho_j - \frac{\alpha}{2}}{z_j}$$
$$-\frac{\alpha}{2} \leq \rho_j \leq \frac{\alpha}{2} \implies w_j = 0$$

Lastly, we have the update for w_0 is

$$w_0 = \frac{\rho_0}{z_0}$$