# 1 Sparse Least Squares

Suppose we want to solve the least squares objective, subject to a constraint that $\mathbf{w}$ is sparse. Mathematically this is expressed as

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{w}\|_0 \leq k \end{aligned}$$



where the $\ell_0$ norm of $\mathbf{w}$ is simply the number of non-zero elements in $\mathbf{w}$. This quantity is otherwise known as the **Hamming Distance** between $\mathbf{w}$ and $\mathbf{0}$, the vector of zeros.



There are several motivations for designing optimization problems with sparse solutions. One advantage is that sparse weights speed up testing time. In the context of kernel problems, if the weight vector $\mathbf{w}$ is sparse, then after we compute $\mathbf{w}$ in training, we can discard features/dimensions with 0 weight, as they will contribute nothing to the evaluation of the hypothesized regression values of test points. A similar reasoning applies to dual problems with dual weight vector $\mathbf{v}$, allowing us to discard the training points corresponding to dual weight 0, ultimately allowing for faster evaluation of our hypothesis function on test points.



Note that the $\ell_0$ norm does not actually satisfy the properties of a norm, evident by the fact that the it is not convex, a property that all norms share. Solving this optimization problem is NP-hard, so we instead aim to find a computationally feasible alternative method that can approximate the optimal solution. We will present two such methods: LASSO, a relaxed version of the problem that replaces the $\ell_0$ norm with a $\ell_1$ norm, and Matching pursuit, a greedy algorithm that iteratively updates one entry of $\mathbf{w}$ at a time until the sparsity constraint can not longer be satisfied.

## 1.1 LASSO

The **least absolute shrinkage and selection operator (LASSO)**, introduced in 1996 by Robert Tibshirani, is identical to the sparse least squares objective, except that the $\ell_0$ norm penalizing $\mathbf{w}$ is now changed to an $\ell_1$ norm:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \\ \text{s.t.} \quad & \|\mathbf{w}\|_1 \leq k \end{aligned}$$

(The $k$ in the constraint is not necessarily the same $k$ in the sparse least squares objective.) The $\ell_1$ norm of $\mathbf{w}$ is the sum of absolute values of its entries:

$$\|\mathbf{w}\|_1 = \sum_{i=1}^{d} |w_i|$$

 

Unlike the $\ell_0$ norm, the $\ell_1$ norm actually satisfies the properties of norms. The relaxation from the $\ell_0$ to $\ell_1$ norm is desirable, because it makes the optimization problem convex, and is no longer NP-hard to solve. But does the $\ell_1$ norm still induce sparsity like the $\ell_0$? As we will see, the answer is yes!

Due to strong duality, we can equivalently express the LASSO problem in the unconstrained form

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|_1$$

We make a striking observation here: LASSO is identical to the ridge regression objective, except that the $\ell_2$ norm (squared) penalizing $\mathbf{w}$ is now changed to an $\ell_1$ norm (with no squaring term).

Recall that the $\ell_2$ norm squared of $\mathbf{w}$, the sum of squared values of its entries:



$$\|\mathbf{w}\|_2^2 = \sum_{i=1}^{d} w_i^2$$



As it turns out, the simple change from the $\ell_2$ to $\ell_1$ norm inherently leads to a sparse solution. In fact, the sparsity inducing properties of the $\ell_1$ norm are not just unique to least squares. To illustrate the point, let's take a step away from least squares for a moment and discuss the $\ell_1$ norm in the context of SVMs. Recall the soft-margin SVM problem (constraints omitted for brevity):



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



The slack $\xi_i$ is constrained to be either positive or zero. Note that if a point $\mathbf{x}_i$ has a nonzero slack $\xi_i > 0$, by definition it must lie inside the margin. Due to the heavy penalty factor $C$ for violating the margin, there are relatively few such points, and thus the slack vector $\xi$ is sparse — most of its entries are $0$. We are interested in explaining why this phenomenon occurs in this specific optimization problem, and identifying the key properties that determine sparse solutions for arbitrary optimization problems.

To reason about the SVM case, let's see how changing some arbitrary slack variable $\xi_i$ affects the loss. A unit decrease in $\xi_i$ results in a "reward" of $C$, and is captured by the partial derivative $\frac{\partial L}{\partial \xi_i}$. Note that no matter what the current value of $\xi_i$ is, the reward for decreasing $\xi_i$ is constant. Of course, decreasing $\xi_i$ may change the boundary and thus the cost attributed to the size of the margin $\|\mathbf{w}\|^2$. The overall reward for decreasing $\xi_i$ is either going to be worth the effort (greater than cost incurred from $\mathbf{w}$) or not worth the effort (less than cost incurred from $\mathbf{w}$). Intuitively, $\xi_i$ will continue to decrease until it hits a lower-bound "equilibrium" — which is often just $0$.

Now consider the following formulation (constraints omitted for brevity again):

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

The reward for decreasing $\xi_i$ is no longer constant — at any point, a unit decrease in $\xi_i$ results in a "reward" of $2C\xi_i$. As $\xi_i$ approaches $0$, the rewards get smaller and smaller, reaching infinitesimal

values. On the other hand, decreasing $\xi_i$ causes a finite increase in the cost incurred by the $\|\mathbf{w}\|^2$ — the same increase in cost as in the previous example. Intuitively, we can reason that there will be a threshold value $\xi_i^*$ such that decreasing $\xi_i$ further will no longer outweigh the cost incurred by the size of the margin, and that the $\xi_i$'s will halt their descent before they hit zero.

The same reasoning applies to least squares as well. For any particular component $w_i$ of $\mathbf{w}$, the corresponding loss in LASSO is the absolute value $|w_i|$, while the loss in ridge regression is the squared term $w_i^2$. In the case of LASSO the "reward" for decreasing $w_i$ by a unit amount is a constant $\lambda$, while for ridge regression the equivalent "reward" is $2\lambda w_i$, which depends on the value of $w_i$. There is a compelling geometric argument behind this reasoning as well.



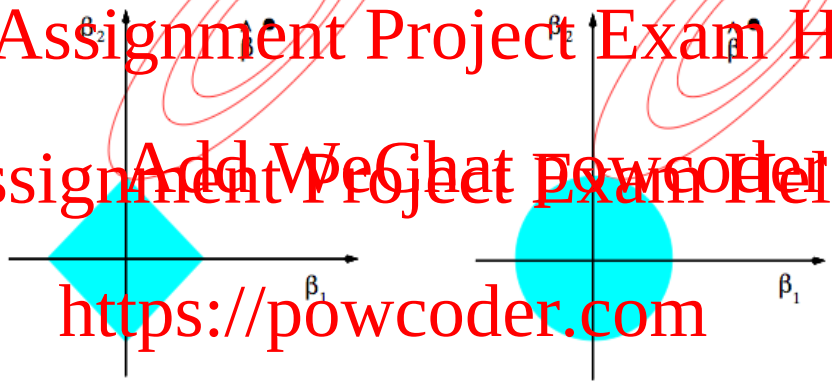


Figure 1: Comparing contour plots for LASSO (left) vs. ridge regression (right).

Suppose for simplicity that we are only working with 2-dimensional data points and are thus optimizing over two weight variables $w_1$ and $w_2$. In both figures above, the red ellipses represent isocontours in w-space of the squared loss $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$. In ridge regression, each isocontour of $\lambda\|\mathbf{w}\|_2^2$ is represented by a circle, one of which is shown in the right figure. Note that the optimal $\mathbf{w}$ will only occur at points of tangency between the red ellipse and the blue circle. Otherwise we could always move along the isocontour of one of the functions (keeping its overall cost fixed) while improving the value of the the other function, thereby improving the overall value of the loss function. We can't really infer much about these points of tangency other than the fact that the blue circle centered at the origin draws the optimal point closer to the origin (ridge regression penalizes large weights).

Now, let's examine the LASSO case. The red ellipses represent the same objective $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$, but now the $\ell_1$ regularization term $\lambda\|\mathbf{w}\|_1$ is represented by diamond isocontours. As with ridge regression, note that the optimal point in w-space must occur at points of tangency between the ellipse and the diamond. Due to the "pointy" property of the diamonds, tangency is very likely to happen at the *corners* of the diamond because they are single points from which the rest of the diamond draws away from. And what are the corners of the diamond? Why, they are points at

which one component of $\mathbf{w}$ is 0!

### 1.1.1 Solving LASSO

Convinced that LASSO achieves sparsity, now let's find the optimal solution to LASSO. Unlike ridge regression, it is not exactly clear what the closed form solution is through linear algebra or gradient methods, since the objective function not differentiable (due to the "pointiness" of the $\ell_1$ norm). Specifically, LASSO zeros out features, and once these weights are set to 0 the objective function becomes non-differentiable. Note however, that the objective is still convex, and we could use an iterative method such as subgradient descent or line search to solve the problem. Here, we will use a line search method called **coordinate descent**.



While SGD focuses on iteratively optimizing the value of the objective $L(\mathbf{w})$ for each *sample* in the training set, coordinate descent iteratively optimizes the value of the objective for each *feature*.



**Algorithm 1:** Coordinate Descent

**while** $\mathbf{w}$ *has not converged* **do**
  pick a feature index $i$
  update $w_i$ to $\arg\min_{w_i} L(\mathbf{w})$



Coordinate descent is guaranteed to find the global minimum if $L$ is **jointly convex**. No such guarantees can be made however if $L$ is only **elementwise convex**, since it may have local minima. To understand why, let's start by understanding elementwise vs joint convexity. Suppose we are trying to minimize $f(x, y)$, a function of two scalar variables $x$ and $y$. For simplicity, assume that $f$ is twice differentiable, so we can take its Hessian. $f(x, y)$ is element-wise convex in $x$ if its Hessian is psd when y is fixed:



$$\frac{\partial^2}{\partial x \partial x} f(x, y) \geq 0$$

Same goes for element-wise convexity in $y$.

$f(x, y)$ is jointly convex in $x$ and $y$ if its Hessian $\nabla^2 f(x, y)$ is psd. Note that being element-wise convex in both $x$ and $y$ does not imply joint convexity in $x$ and $y$ (consider $f(x, y) = x^2 + y^2 - 4xy$ as an example). However, being joint convexity in $x$ and $y$ does imply being element-wise convex in both $x$ and $y$.

Now, if $f(x, y)$ was jointly convex, then we could find the gradient wrt. $x$ and $y$ individually, set them to 0, and be guaranteed that would be the global minimum. Can we do this if $f(x, y)$ is element-wise convex in both $x$ and $y$? Even though it is true that $\min_{x,y} f(x, y) = \min_x \min_y f(x, y)$, we can't always just set gradients to 0 if $f(x, y)$ is not jointly convex. While the inner optimization problem over $y$ is convex, the outer optimization problem over $x$ may no longer be convex. In the case when joint convexity is not reached, there is no clean strategy to find global minimum and we must analyze all of the critical points to find the minimum.

In the case of LASSO, the objective function is jointly convex, so we can use coordinate descent.

There are a few details to be filled in, namely the choice of which feature to update and how $w_i$ is updated. One simple way is to just pick a random feature $i$ each iteration. After choosing the feature, we have to update $w_i \leftarrow \arg\min_{w_i} L(\mathbf{w})$. For LASSO, it turns out there is a closed-form solution (note that we are only minimizing with respect to one feature instead of all the features).

Let's solve the line search problem $\min_{w_i} L(\mathbf{w})$. For convenience, let's separate the terms that depend on $w_i$ from those that don't. Denoting $\mathbf{x}_j$ as the $j$-th column of $\mathbf{X}$, we have

$$\begin{aligned} L(\mathbf{w}) &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1 \\ &= \left\|\sum_{j=1}^{d} w_j \mathbf{x}_j - \mathbf{y}\right\|_2^2 + \lambda|w_i| + \lambda\sum_{j\neq i}|w_j| \\ &= \|w_i\mathbf{x}_i + \mathbf{r}\|_2^2 + \lambda|w_i| + C \end{aligned}$$

where $\mathbf{r} = \sum_{j\neq i} w_j\mathbf{x}_j - \mathbf{y}$ and $C = \lambda\sum_{j\neq i}|w_j|$. The objective can in turn be written as

$$L(\mathbf{w}) = \lambda|w_i| + C + \sum_{j=1}^{n}(w_i x_{ji} + r_j)^2$$

where $x_{ji}$ is the $i$th feature of the $j$th datapoint.

Suppose that the optimal $w_i^*$ is strictly positive: $w_i^* > 0$. Setting the partial derivative of the objective wrt. $w_i$ (the partial derivative in this case is defined since $w_i^* \neq 0$) to 0, we obtain

$$\frac{\partial L}{\partial w_i} = \lambda + \sum_{j=1}^{n} 2x_{ji}(w_i x_{ji} + r_j) = 0 \implies w_i^* = \frac{-\lambda - \sum_{j=1}^{n} 2x_{ji}r_j}{\sum_{j=1}^{n} 2x_{ji}^2}$$

Denoting $a = -\sum_{j=1}^{n} 2x_{ji}r_j$ and $b = \sum_{j=1}^{n} 2x_{ji}^2$, we have

$$w_i^* = \frac{-\lambda + a}{b}$$

But this only holds if the right hand side, $\frac{-\lambda+a}{b}$, is actually positive. If it is negative or 0, then this means there is no optimum in $(0, \infty)$.

When $w_i^* < 0$, then similar calculations will lead to

$$w_i^* = \frac{\lambda + a}{b}$$

Again, this only holds if $\frac{\lambda+a}{b}$ is actually negative. If it is positive or 0, then there is no optimum in $(-\infty, 0)$.

If neither the conditions $\frac{-\lambda+a}{b} > 0$ or $\frac{\lambda+a}{b} < 0$ hold, then there is no optimum in $(-\infty, 0)$ or $(0, \infty)$. But the LASSO objective is convex in $w_i$ and has an optimum somewhere, thus in this case $w_i^* = 0$. In order for this case to hold, we must have that $\frac{-\lambda+a}{b} \leq 0$ and $\frac{\lambda+a}{b} \geq 0$. Rearranging, we can see this is equivalent to $|a| \leq \lambda$.

Examine each of the following cases:

- $\frac{-\lambda+a}{b} \leq 0$ and $\frac{\lambda+a}{b} \geq 0$: $w_i^* = 0$
- $\frac{-\lambda+a}{b} \leq 0$ and $\frac{\lambda+a}{b} < 0$: $w_i^* < 0$
- $\frac{-\lambda+a}{b} > 0$ and $\frac{\lambda+a}{b} \geq 0$: $w_i^* > 0$
- $\frac{-\lambda+a}{b} > 0$ and $\frac{\lambda+a}{b} < 0$: impossible since this implies $\frac{-\lambda+a}{b} > \frac{\lambda+a}{b}$ and $\lambda$ and $b$ are non-negative

The cases above imply the optimal solution $w_i^*$:

$$w_i^* = \begin{cases} 0 & \text{if } |a| \leq \lambda \\ \frac{-\lambda+a}{b} & \text{if } \frac{-\lambda+a}{b} > 0 \\ \frac{\lambda+a}{b} & \text{if } \frac{\lambda+a}{b} < 0 \end{cases}$$



where



$$a = -\sum_{j=1}^{n} 2x_{ji}r'_j, \quad b = \sum_{j=1}^{n} 2x_{ji}^2$$



This is not a gradient-descent update! We have a closed-form solution for the optimum $w_i$, given that all of the other weights are fixed constants. We can see explicitly how the LASSO objective induces sparsity — $a$ is some function of the data and the other weights, and when $|a| \leq \lambda$, we set $w_i = 0$ in this iteration of coordinate descent. By increasing $\lambda$, we increase the threshold of $|a|$ for $w_i$ to be set to 0, and our solution becomes more sparse. Also note that the term $\frac{a}{b}$ is the least squares solution (without regularization), so we can see that the regularization term tries to pull the least squares update towards 0.



One subtle point: during coordinate descent, weights can be "reactivated" after having been set to 0 in a previous iteration, since $a$ is affected by factors other than $w_i$.



## 1.2 Matching Pursuit

Rather than relaxing the $\ell_0$ constraint (as seen in LASSO), the **matching pursuit** algorithm keeps the constraint, and instead finds an approximate solution to the sparse least squares problem in a greedy fashion. The algorithm starts with with a completely sparse solution ($\mathbf{w}^0 = \mathbf{0}$), and iteratively updates $\mathbf{w}$ until the sparsity constraint $\|\mathbf{w}\|_0 \leq k$ can no longer be met. At iteration $t$, the algorithm can only update one entry of $\mathbf{w}^{t-1}$, and it chooses the feature that minimizes the (squared) norm of the resulting *residual* $\|\mathbf{r}^t\|^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}^t\|^2$.

**Algorithm 2:** Matching Pursuit

initialize the weights $\mathbf{w}^0 = \mathbf{0}$ and the residual $\mathbf{r}^0 = \mathbf{y} - \mathbf{X}\mathbf{w}^0 = \mathbf{y}$

**while** $\|\mathbf{w}\|_0 < k$ **do**

find the feature $i$ for which the length of the projected residual onto $\mathbf{x}_i$ is maximized:

$$i = \arg\min_j (\min_\nu \|\mathbf{r}^{t-1} - \nu \mathbf{x}_j\|) = \arg\max_j \frac{\left|\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle\right|}{\|\mathbf{x}_j\|}$$

update the $i$'th feature entry of the weight vector:

$$w_i^t = w_i^{t-1} + \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_i \rangle}{\|\mathbf{x}_i\|^2}$$

update the residual vector: $\mathbf{r}^t = \mathbf{y} - \mathbf{X}\mathbf{w}^t$



At iteration $t$, we pick the coordinate $i$ such that the distance from the residual $\mathbf{r}^{t-1}$ to $\mathbf{x}_i$ (the $i$'th column of $\mathbf{X}$ corresponding to feature $i$, *not* datapoint $i$) is minimized:



$$i = \arg\min_j (\min_\nu \|\mathbf{r}^{t-1} - \nu \mathbf{x}_j\|)$$



This equates to finding the index $i$ for which the length of the projection onto $\mathbf{x}_i$ is maximized:

$$i = \arg\max_j \frac{\left|\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle\right|}{\|\mathbf{x}_j\|}$$



Let's see why this is true. The inner optimization problem $\min_\nu \|\mathbf{r}^{t-1} - \nu \mathbf{x}_i\|$ is simply a projection problem, and its solution is



$$\nu^* = \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle}$$

which gives a value of

$$\left\| \mathbf{r}^{t-1} - \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} \mathbf{x}_i \right\|$$

The outer optimization problem selects the feature that minimizes this quantity:

$$\begin{aligned}
i &= \arg\min_j \left\| \mathbf{r}^{t-1} - \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} \mathbf{x}_i \right\| \\
&= \arg\min_j \left\| \mathbf{r}^{t-1} - \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} \mathbf{x}_i \right\|^2 \\
&= \arg\min_j \|\mathbf{r}^{t-1}\|^2 + \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle^2}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} - 2\frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle^2}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} \\
&= \arg\min_j \|\mathbf{r}^{t-1}\|^2 - \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle^2}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle} \\
&= \arg\max_j \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle^2}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle}
\end{aligned}$$

$$= \arg\max_j \frac{\left|\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle\right|}{\|\mathbf{x}_j\|}$$

Now that we have found the feature $i$ that maximizes the length of the projection, we must update corresponding weight $i$ and the residual vector. The updated residual $\mathbf{r}^t$ is the result of projecting $\mathbf{r}^{t-1}$ onto feature $\mathbf{x}_i$:

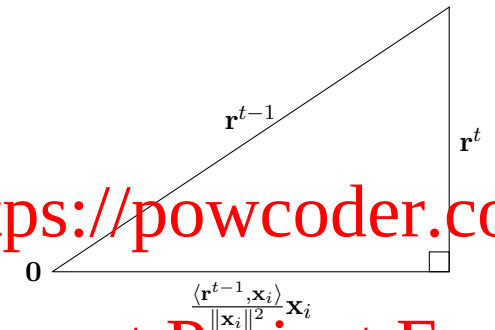




Figure 2: The updated residual $\mathbf{r}^t$, current residual $\mathbf{r}^{t-1}$, and scaled feature $\mathbf{x}_i$ form a right triangle.



The updated residual is given by

$$\mathbf{r}^t = \mathbf{r}^{t-1} - \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_i \rangle}{\|\mathbf{x}_i\|^2}\mathbf{x}_i$$



which corresponds to adding $\frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_i \rangle}{\|\mathbf{x}_i\|^2}$ to the corresponding weight $w_i^t$:



$$w_i^t = w_i^{t-1} + \frac{\langle \mathbf{r}^{t-1}, \mathbf{x}_i \rangle}{\|\mathbf{x}_i\|^2}$$

We update $w_i$ to the optimum projection value and repeat greedily at each iteration. At each iteration, the (squared) length of the residual $\|\mathbf{X}\mathbf{w}^t - \mathbf{y}\|^2$ monotonically decreases since $\|\mathbf{r}^{t-1}\|^2 > \|\mathbf{r}^t\|^2$. While matching pursuit is not guaranteed to find the optimal $\mathbf{w}^*$, in practice it works well for most applications.

## 1.3 Orthogonal Matching Pursuit

The **Orthogonal Matching Pursuit (OMP)** algorithm is an extension to the standard Matching Pursuit algorithm with the following difference: at iteration $t$, we maintain a set $I^t$ of all features selected by the algorithm so far, and instead of updating just the one weight corresponding to feature $i$ found at iteration $t$, we update all weights corresponding to the features in $I^t$ using Least Squares.

**Algorithm 3:** Orthogonal Matching Pursuit

initialize the weights $\mathbf{w}^0 = \mathbf{0}$ and the residual $\mathbf{r}^0 = \mathbf{y} - \mathbf{X}\mathbf{w}^0 = \mathbf{y}$

initialize a set of features $I^0 = \emptyset$

**while** $\|\mathbf{w}\|_0 < k$ **do**

find the feature $i$ for which the length of the projected residual onto $\mathbf{x}_i$ is maximized:

$$i = \arg\min_j (\min_\nu \|\mathbf{r}^{t-1} - \nu \mathbf{x}_j\|) = \arg\max_j \frac{\left|\langle \mathbf{r}^{t-1}, \mathbf{x}_j \rangle\right|}{\|\mathbf{x}_j\|}$$

add feature $i$ to the set of features:

$$I^t = I^{t-1} \cup \{i\}$$

Estimate the best linear fit of the target $\mathbf{y}$ using the features obtained so far. Given that we have found $t$ good features, we now find the best linear fit for the target $\mathbf{y}$ using these $t$-features. Define $\mathbf{X}_t = \begin{bmatrix} \mathbf{x}_{i_1} & \cdots & \mathbf{x}_{i_t} \end{bmatrix}$ made up of these $t$-features. Then we determine $\mathbf{w}^t$ as the solution for the following least-squares problem:

$$\mathbf{w}^t = \arg\min_{\mathbf{w} \in \mathbb{R}^t} \|\mathbf{y} - \mathbf{X}_t \mathbf{w}\|_2^2$$

update the residual vector: $\mathbf{r}^t = \mathbf{y} - \mathbf{X}\mathbf{w}^t$

The motivation for OMP is as follows: if we do not refit on all features updated so far after choosing a new feature, the new residual will not necessarily be orthogonal to the span of the canonical basis vectors corresponding to those chosen coordinates, and is therefore not optimal. The OMP algorithm ensures that $\mathbf{w}^t$ corresponds to an optimal Least Squares solution if we restricted our features to just those in $I^t$.