18 Lasso – Regularized Regression

Recall the (high-dimensional) regression problem. The input is a point set $P \subset \mathbb{R}^{d+1}$ with n points $\{p_1, \ldots, p_n\}$. We can also think of P as a matrix, and decompose it into two parts $P = [P_X \; y]$ where P_X represents the first d columns and y represents the last column. Here we use P_j to denote the jth column. Then the goal is to find a set of coefficients $A = [a_0, a_1, a_2, \ldots, a_d]^T$ so that a function $f_A(p) = \sum_{i=0}^{d} a_i p_i$ approximates the corresponding values $p_y = p_{d+1}$ (assuming $p_0 = 1$).

The standard least squares formulation finds A that minimizes

$$\sum_{p_i \in P} (f_A(p_i) - y_i)^2. \tag{18.1}$$

We can write $X = [\mathbf{1} \; P_X]$ as a $((d+1) \times n)$ matrix where the first column is all 1s and the last d columns are P_X. Now equation (18.1) can be rewritten as

$$\min_A \|XA - y\|_2.$$

Then we can minimize equation (18.1) by solving for

$$A = (X^T X)^{-1} X^T y. \tag{18.2}$$

18.1 Regularization

Recall that by the Gauss-Markov Theorem, that (18.2) is the minimum variance (least squares) solution to the problem with P given that it is unbiased. *However*, it may be advantageous to bias towards a small slope solution.

This models the residual as only in the y direction, and thus implicitly assumes that the X coordinates have no error. Thus when noise happens, it happens in the y-coordinate, and we want to minimize the effect of this. To do so, we can "regress to the mean."

Example: *Consider a hard* TRUE-FALSE *test. Each student knows some fraction of the answers (say 50% of them) and guess on the rest. The expected score is 75%. Say this was the case for 100 students, and we took the 10 students how scored the best; say their average score was 80%. If we gave these same 10 students another similar test (still* TRUE-FALSE*, and they know half, guess half), then what is going to be their expected score: 75%. That is we expect them to regress towards the mean!*

So in linear regression, we expect that y-values will not be as wild in this observed data as it would be if we observed new data. So we want to give a prediction that made more extreme data have less affect. These solutions can have overall less variance, but do not have 0 bias.

Another view is that we have a prior (as in Bayesian statistics), say of weight $s/(s+n)$, that the mean value of the y-coordinates is correct. So we don't want to entirely use the raw data.

Tikhonov regularization. To this end, we can change the *loss* function, that which was measuring the error of our solution A. The Tikhonov regularization for a parameter $s \geq 0$ finds

$$\arg\min_A \|XA - y\|_2 + s\|A\|_2. \tag{18.3}$$

This is also known as *ridge regression*. Magically, this can be solved just as easily as least squares by setting

$$A = (X^T X + s^2)^{-1} X^T y.$$

Lasso. An alternative approach (the focus here) will be a bit more complicated to solve, but has a couple of other very nice properties. It is called the Lasso, or alternatively *basis pursuit*; for a parameter $s \geq 0$ it finds

$$\arg\min_A \|XA - y\|_2 + s\|A\|_1. \tag{18.4}$$

Note that the only difference is that it has an L_1 norm on the $\|A\|$ instead of the L_2 norm in ridge regression. This also prevents the simple matrix-inverse solution of ridge regression. However it will have two other very nice properties:

- In high dimensions, it will bias towards sparse solutions
- It forces one to consider multiple values of s, and hopefully choose a reasonable one.

We will mainly focus on the *Least Angle Regression* method to solve for A.

Orthogonal Matching Pursuit. However, before that, lets note that one can also use *Orthogonal Matching Pursuit* (OMP). Here it is sometimes called *forward subset selection*. This may be slightly easier to implement, but will not provide the optimal solution and allows one to cherry-pick a value s.

Algorithm 18.1.1 Orthogonal Matching Pursuit

Set $r = y$; and $a_j = 0$ for all $j \in [d]$.
for $i = 1$ **to** t **do**
  Set $X_j = \arg\max_{X_{j'} \in X} |\langle r, X_{j'} \rangle|$.
  Set $a_j = \arg\min_\gamma \|r - X_j\gamma\|^2 + s|\gamma|$.
  Set $r = r - X_j a_j$.
Return A.

18.2 Least Angles Regression

The first insight is that instead of (18.4) it is equivalent to solve

$$\arg\min_A \|XA - y\|_2^2 \qquad \text{such that } \|A\|_1 \leq t. \tag{18.5}$$

for some parameter t. Note that we have replaced the parameter s with another one t. For any value of s and solution A_s to (18.4), there is a value t that provides an identical solution A_t to (18.5). To see this, solve for A_s, and then set $t = \|A_s\|_1$.

This same dual version exists for ridge regression as well. Why is this useful?

Sparsity. This biases solutions to have 0 along many of the coordinates a_j. This is illustrated in Figure 18.1 where the Lasso solution for A is restricted to lie within an L_1 ball of radius t, but otherwise be as close to y as possible. The least squares solution is the best possible fit for A. We can see the extra L_2 error around this solution which is minimized with respect lying in a radius t L_1 ball for Lasso or L_2 ball for Tikhonov regularization.

Note that the L_1 ball has "pointy" corners, and thus bias solutions for A towards these corners. The corners have some coordinate of A as 0. In higher dimensions, this L_1 ball has even more corners, and they play an even more prominent role in the solutions to these minimization problems.

As t becomes smaller, it is more and more likely that the solution to A is found on a higher-degree corner. But also the solution found becomes further and further away from the least squares solution. If we set $t = \infty$ then the Lasso (and Tikhonov) solution is the least squares solution. How do we balance these aspects?

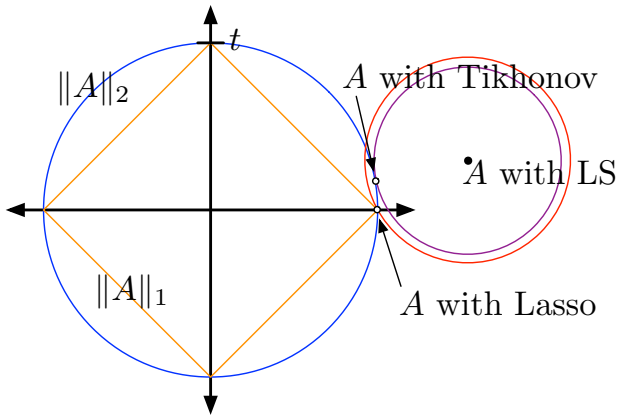

Figure 18.1: Radius t ball under L_1 and L_2, and the results of Lasso and Tikhonov regularization.

Increasing t. As we increase t (our coefficient budget), then we allow some a_j to increase. If we start with $t = 0$, then all $a_j = 0$. Now using the (piecewise-) linear equation $t = \sum_{j=1}^{d} \|a_j\|^d$ and

$$r(t) = y - \sum_{j=1}^{d} X_j a_j(t).$$

The goal is to minimize $\|r(t)\|$, and we note that changing some a_j have more effect on $r(t)$ than others.

First find, $j_1 = \arg\max_j |\langle X_j, r\rangle|$. This is the coordinate with maximum influence on $r(t)$. We vary this first and set $a_{j_1}(t) = a_j \cdot t$. We now increase t, while only varying a_j (as specified) until some other coordinate is worth increasing.

Next find j_2 such that $j_2 \neq j_1$ and has

$$|\langle X_{j_1}, r(t)\rangle| = |\langle X_{j_2}, r(t)\rangle|.$$

We can solve for the value t at which this will happen for each $j \neq j_1$ since the above is a linear equation in t. The index j_2 is selected in that it has the smallest value t_2 at which this equality happens. *This is the first time that (18.5) is minimized with* 2 *non-zero coefficients.* The next step is to reset the correlations (via the first derivatives) such that $|b_1| + |b_2| = 1$ as

$$\begin{aligned} a_{j_1}(t) &= a_{j-1}(t_2) + (t - t_2)b_1 \qquad \text{for } t \geq 0 \\ a_{j_2}(t) &= (t - t_2)b_2 \qquad \text{for } t \geq t_2. \end{aligned}$$

This implies that as t increase, the optimal choice in a_j is linear in t with slopes defined by $b_1, b_2, \ldots$

We continue this, in each ith step finding a time t_i at which increasing some coefficient a_{j_i} will have $|\langle X_{j_i}, r(t_i)\rangle| = |\langle X_J, r(t_i)\rangle|$ where J is the set of indices we are tracking and X_J is the subset of columns of X corresponding to those indices. Then we add this j_i to J, update $r(t)$ for $t \geq t_i$, and recompute the derivatives $b_1, \ldots, b_i$, and find the next value t_{i+1} and so on. See Algorithm 18.2.1 for a more formal description of the algorithm.

So in the process we have solved the optimal choice of A for each value t (and hence each value s). We can choose the best one with *cross-validation* where we leave out some data and evaluate how well the model works on that data (more later). We can maintain this estimate and solve for the minimum (since we have all linear equations) along the way.

Algorithm 18.2.1 Least Angle Regression

Set $a_j = 0$, $b_j = 0$ for all $j \in [d]$.
Set $j_1 = \arg\max_j |\langle X_j, r(0)\rangle|$; $b_{j_1} = 1$; and $J = \{j_1\}$.
Set $r(t) = y - \sum_{j=1}^{d} X_j a_j(t)$ where $a_j(t) = b_j \cdot t$. *$a_{j_1}(t) = t$, otherwise $a_j(t) = 0$ for $j \neq j_1$*
for $i = 2$ **to** n **do**
  for all $j \notin J$ **do**
    Find $\tau_j > t$ such that $|\langle X_j, r(t)\rangle| = |\langle X_{j_1}, r(t)\rangle|$ *Note: all $j \in J$ have same $|\langle X_j, r(t)\rangle|$*
  Set $t_i = \min \tau_j$ and $j_i = \arg\min_j \tau_j$; $J = J \cup j_i$.
  Solve for b_j for all $j \in J$ such that $\sum_{j \in J} |b_j| = 1$. *Take derivatives of $|\langle X_j, r(t)\rangle|$ and normalize*
  For $t \geq t_i$ redefine $r(t) = y - \sum_{i=1}^{d} X_j a_j(t)$ where $a_j(t) = a_j(t_i) + (t - t_i)b_j$.
Return $A(t)$ when its cross-validation score is smallest.

There is a variant of this algorithm where we may want to snap values a_j to 0 even if $j \in J$. This happens since initially b_j may be positive, but as J increases, it may become negative. Then when a_j hits 0, we remove it from J (this can time can be treated as its τ_j value.). Then we can later re-add it to J. This is needed to get the optimal solution for any t. Is a little more work, but could require an exponential number of steps.