13 Intro to Linear Regression (in $\mathbb{R}^2$)

Given a large set of data (with noise and error and unimportant variation) is to try to find a (1) *simple* rule or model which the data is very (2) *close to following*. Abstractly, that is, given data X then we want to find a model $M = f(X)$, so for all $x \in X$ that x is close to $M(x)$, where $M(x)$ is the "best" representation of x in M. This task is known as *regression* and it comes in many forms.

The simplest form of this is *linear regression* in two dimensions. Even this simple problem has many variants, which this note tries to overview, contrast, and to provide intuition for. Further notes will then dive deeper into certain developments in advanced versions of (linear) regression.

13.1 Linear Least Squares

The simplest version of regression is least-squares linear regression. The input to the problem is a set $P \subset \mathbb{R}^2$ of n points, so each $p = (p_x, p_y) \in P$. Let P_x be the set of x-coordinates in P and let P_y be the set of y-coordinates in P.

The output is the line $\ell : y = ax + b$ that minimizes

$$L_2(P, a, b) = \sum_{p \in P} (p_y - ap_x - b)^2. \tag{13.1}$$

This is the *vertical distance* from each p to the spot on the line ℓ at x-value p_x.

As we will see, the choice of *vertical distance* and the *squaring* of it are modeling choices, but also leads to a very simple algorithm.

Solution: We first need to define

- $\bar{P}_x = \frac{1}{n} \sum_{p \in P} p_x$ is the *average* of the x-coordinates.
- The *covariance* between P_x and P_y is defined

$$\textbf{Cov}[P_x, P_y] = \frac{1}{n} \sum_{p \in P} (p_x - \bar{P}_x)(p_y - \bar{P}_y)$$

- And $\textbf{Var}[P_x] = \textbf{Cov}[P_x, P_x]$ is the *variance* of the set of x-coordinates.

Now we can solve for $a = \textbf{Cov}[P_x, P_y]/\textbf{Var}[P_x] = \langle P_x, P_y \rangle / \|P_x\|^2$.
Then we set $b = \bar{P}_y - a\bar{P}_x$.

13.1.1 Multidimensional Extension

For $P \subset \mathbb{R}^d$ of size n, let X be a $n \times d$ vector where each row (of n rows) represents a point $p \in P$. Let the first $d-1$ columns represent the x-variables, and the dth column by a 1 (for each point). The dth dimension of each $p \in P$ is p_y, and let P_y be the vector of these n values.

Now

$$a = (X^T X)^{-1} X^T P_y. \tag{13.2}$$

We call the matrix

$$H_X = X(X^T X)^{-1} X^T \tag{13.3}$$

the *hat* matrix since $\hat{y} = Xa = H_X y$ (where $y = P_y$) puts the "hat" on y to provide the modeling estimate of value y.

Note that the dth column of X provides the "offset" b automatically.

Example matrix. We can show the desired equality $Xa = p_y$ where a single d-dimensional data point $(r_1, r_2, r_3, \ldots, r_{d-1}, r_d)$ is shown in the third row. The "solution" is $(a_1, a_2, \ldots, a_{d-1}, b)$.

$$\begin{bmatrix} \mathtt{x} & \mathtt{x} & \mathtt{x} & \ldots & \mathtt{x} & 1 \\ \mathtt{x} & \mathtt{x} & \mathtt{x} & \ldots & \mathtt{x} & 1 \\ r_1 & r_2 & r_3 & \ldots & r_{d-1} & 1 \\ \mathtt{x} & \mathtt{x} & \mathtt{x} & \ldots & \mathtt{x} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \ldots \\ a_{d-1} \\ b \end{bmatrix} \approx \begin{bmatrix} \mathtt{x} \\ \mathtt{x} \\ r_d \\ \mathtt{x} \end{bmatrix}$$

13.1.2 Polynomial Extension

If instead of finding a linear fit we want to find a polynomial fit of degree t, then the goal is an equation

$$g : y = a_0 + a_1 x + a_2 x^2 + \ldots = \sum_{i=0}^{t} a_i x^i,$$

then instead we can create a matrix X where the ith column (for the row representing point $p \in P$) is populated with $(p_x)^i$. Then we can solve for all a_i-coefficients using equation (14.2).

This approach is called *lifting* (since it transforms to a "higher"-dimensional problem) or *linearization* (because it makes a non-linear problem linear). Similar tricks can be applies to other more exotic types of regression, where one can map a non-linear problem to a linear equation, perhaps approximately. For instance, so-called "kernel methods" in machine learning do so implicitly.

13.1.3 Gauss Markov Theorem

This method of fitting provides the optimal solution to Equation (14.1) for any linear model if

- the solution has 0 expected error, and
- all errors $\varepsilon_p = p_x - ap_x - b$ are not known to be correlated.

This is equivalent to the *minimum variance* solution.

So are we done? (It extends to multiple dimensions, to polynomial fits, and is the minimum variance solution!) ... *No!*

There are at least four issues that remain:

- Can we make it more robust to outliers (L_2 error puts more emphasis on outliers)?
- Can we get less "error" by allowing bias?
- Can we minimize distance to line, not "vertical" distance?
- The matrix inversion $(X^T X)^{-1}$ can be expensive, can we use other techniques to make this more efficient.

13.2 Theil-Sen Estimator

The *Theil-Sen estimator* provides a "robust" estimator for linear regression. The *breakdown* point of an estimator $e(P)$ is the minimum fraction of points in P that if moved to ∞ (or anywhere) then $e(P)$ might also move to ∞ (e.g. do something nonsensical to most of the data). A *robust estimator* is one that has a high breakdown point.

For instance, in $\mathbb{R}^1$, the *mean* of data $\bar{P} = \frac{1}{n}\sum_{p \in P}$ has a breakdown point of $1/n$ and is not robust. Whereas a *median* (a point $c = \mathsf{med}(P)$ where $P_c = \{p \in P \mid p < c\}$ has $|P_c| = 1/2$) has a breakdown point of $1/2$ and is robust.

When the estimator is a line, then the least-squares estimate corresponds with the mean, and is not robust. A single point can greatly effect the slope of the line. The Theil-Sen estimator is a robust version of linear regression. It is constructed as follows:

First construct the median of all slopes. For all $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ with $x_i < x_j$ then let $s_{i,j} = (y_j - y_i)/(x_j - x_i)$ be the slope defined by these two points. Now let $a = \mathsf{med}(\{s_{i,j} \mid x_i < x_j\})$.

Then let $b = \mathsf{med}(\{y_i - ax_i\})$ by the intercept of the line $\ell_{\mathsf{TS}} : y = ax + b$.

This line estimator ℓ_{TS} has a breakdown point of 0.293.

Siegel extension. This was improved by Siegel to be made more robust. Let $s_i = \mathsf{med}(\{s_{j,i} \mid x_j < x_i\} \cup \{s_{i,j} \mid x_i < x_j\})$. Then set $a = \mathsf{med}(\{s_i\}_i)$ where $\{s_i\}_i$ is the set of all s_is. And again set $b = \mathsf{med}(\{y_i - ax_i\})$ where $\ell_{\mathsf{S}} : y = ax + b$.

This line estimator ℓ_{S} has breakdown point 0.5 and can be computed in $O(n \log n)$ time. Naively it takes $O(n^2)$ time.

13.3 Tikhonov Regularization (ridge regression)

This is another linear regression technique designed to add bias to the answer, but reduce the overall error. For now we assume the data is *centered* so that $\bar{P}_x = 0$ and $\bar{P}_y = 0$.

The goal is now to minimize

$$L_{2,s}(P, a) = \sum_{p \in P} (p_y - ap_x)^2 + sa^2. \tag{13.4}$$

where s is a *regularization* (or *shrinkage*) parameter. The larger the value s, the more bias there is in the model. In this case it can be seen to bias the solution to be "flatter" with the intuition that noise is only in the x-coordinate, and really large noise will tend to be far from 0 (either towards $+\infty$ or $-\infty$).

So the larger s the less trust there is in the data (e.g. we expect to have less covariance). In this sense it biases "towards the mean" where the "mean" is no correlation. This is related to thinking of having a Bayesian prior that the data has 0 slope, and s tells us how much to trust the prior versus the data.

There is a very cool equivalence between this problem and minimizing

$$L_2^t(P, a) = \sum_{p \in P} (p_y - ap_x)^2 \quad \text{such that } a^2 \leq t, \tag{13.5}$$

where t is some parameter. It can be shown that for every solution to Equation 14.4 with value s there is an equivalent solution to Equation 14.5 for some parameter t. (Set $t = a^2$ for the a that minimizes $L_{2,s}(P, a)$.) Moreover, there is a one-to-one correspondence between the solutions using each s and another with each t; as s decreases, the corresponding solution in the dual formulation has t increase. This view will be useful in higher dimensions.

Amazingly, the solution to Equation (14.4) for a can be found as

$$\hat{a} = \frac{\langle P_x, P_y \rangle}{\langle P_x, P_x \rangle + s^2} = (X^T X + s^2 I)^{-1} X^T P_y$$

where $X = P_x$ and $sa^2 = \|sIa\|^2$. So solving Tikhinov regularization is as simple as solving for least squared regularization.

What is the correct value of s? Good question! Best to use *cross-validation* to see which one works best. Leave out some data, build model, and test on the remaining. Try building model with many values s and see which one gives best result on test data.

13.4 Lasso (basis pursuit)

The cousin of Tikhonov regularization is the Lasso. Again we assume here $\bar{P}_x = 0$ and $\bar{P}_y = 0$. The goal now is to minimize

$$L_{1,s}(P, a) = \sum_{p \in P} (p_y - a p_x)^2 + s|a|,$$

where s is again the *regularization* (or *shrinkage*) parameter. This is equivalent to minimizing

$$L_1^t(P, a) = \sum_{p \in P} (p_y - a p_x)^2 \quad \text{such that } |a| < t,$$

for some t. Again, there is a one-to-one correspondence between the solution for each s and each t.

In later lectures we will see that in higher dimensions when s is large, this biases to sparse solutions. We will then give geometric intuition of why this is robust. We will also see how to solve this efficiently with least angle regression (LAR).

13.5 Principal Component Analysis

Alternatively to minimizing vertical distance (in all techniques above) Principal Component Analysis (PCA) minimizes the projection distance to a line. Instead of explaining y from x, it explains the relationship between x and y. Before we assumed x was correct, now there can be error/residuals in both.

We can again first *double center* and assume that $\bar{P}_x = 0$ and $\bar{P}_y = 0$. If not, subtract $\bar{P}_x$ from all x coordinates and subtract $\bar{P}_y$ from all y coordinates.

Then (in $\mathbb{R}^2$) the (principal component analysis) PCA of a point set P is a vector v (with $\|v\| = 1$) that minimizes

$$\text{PCA}(v) = \sum_{p \in P} \|p - \langle p, v \rangle v\|^2.$$

Note that $\langle p, v \rangle$ is a scalar, specifically, the length of p in the direction v (from the origin). And v is a direction (from the origin). So then $\langle p, v \rangle v$ is the "projection" of p onto the vector v.

And $\|p - \langle p, v \rangle v\|$ is the distance to that projection. So PCA minimizes the sum of squared errors of points to their (orthogonal) projection.

How do we find v? In $\mathbb{R}^2$ it only depends on 1 parameter (think angles) and $\text{PCA}(v)$ is convex (up to symmetries/antipodes). In higher dimensions, due to very cool least-squares properties, we can, for instance, solve this one dimension at a time.