CS 536 : Linear Regression

16:198:536

As with classification, we have a data set $\{(\underline{x}^i, y^i)\}_{i=1,\ldots,m}$ and we want to construct some kind of model or description of this data - some kind of hypothesis that describes the relationships between the $\underline{x}$ and the y. In particular, for regression we are interested in taking y as a **real value** that needs to be predicted, rather than a discrete class label or value. In the usual way, if H is the set of possible hypotheses, we would like to find the hypothesis that minimizes some notion of expected error on new data:

$$\text{err}(f) = \mathbb{E}\left[\text{Error}(f(\underline{X}),\ Y)\right]. \tag{1}$$

For classification, we viewed this error in terms of a 0-1 error, effectively looking at the probability of misclassifying a new data point. For regression problems, we frequently classify error in terms of **mean squared error** or

$$\text{err}(f) = \mathbb{E}\left[(f(\underline{X}) - Y)^2\right]. \tag{2}$$

Of course, this notion of error relies on knowing the underlying distribution of data - which we do not know. We can only estimate the error based on the data that we have, hence we look at the **training error**,

$$\text{err}_{\text{train}}(f) = \frac{1}{m}\sum_{i=1}^{m}\left(f(\underline{x}^i) - y^i\right)^2. \tag{3}$$

It's worth noting at this point that this choice of error function is effectively arbitrary - there are some theoretical justifications for it as will be seen (Bayesian analysis with Gaussian priors, etc) but while it is common in practice and in literature, it may not capture the important aspects of what error mean for the specific problem you are trying to solve. This is akin to in classification how false negatives may have a different significance than false positives, depending on the context of the classification problem. But most of the analysis that follows here will be focused on mean squared error, as it is a fairly common and well behaved error function.

Questions of sample complexity will again enter into the picture here - how many sampled will we need in order to guarantee that the training error is not too far off from the true error, and therefore avoid overfitting? Much depends on the hypothesis space, or set of models we are willing to consider. For a first attempt, we will look at linear models of the form $f(\underline{x}) = \underline{w}.\underline{x}$ for some real weight vector $\underline{w}$ - note, we are implicitly taking the simplification here that the bias or constant term in the linear model has been absorbed into the weight vector by augmenting $\underline{x}$ with a 1 in the zero-th position (as a result, the 'true dimension' of the data will be $k+1$, but I will still frequently refer to the dimension of the data simply as k).

In this case, we can simplify the notion of training error in the following way - let X be the matrix of data vectors, where row i corresponds to the vector $\underline{x}^i$; the matrix X will have m rows and k columns. Let $\underline{w}$ be a column vector with k entries, and $\underline{y}$ be the result vector, a column vector of the corresponding y^i values. In this case, the training error can be expressed as

$$\text{err}_{\text{train}}(\underline{w}) = \frac{1}{m}||\underline{y} - X\underline{w}||^2. \tag{4}$$

1 The Naive Solution

One immediate approach is to try to find the weight vector $\underline{w}$ that minimizes the training error - effectively finding the line that fits as tightly as possible to the observed data. If there are sufficiently many training samples, intuitively this solution should generalize well to data that is not seen, i.e., minimizing training error should have a good chance of minimizing true or test error. But how many samples this requires is a reasonable problem to consider.

In this approach, we simply want to solve

$$\min_{\underline{w}} ||\underline{y} - X\underline{w}||^2. \tag{5}$$

Note that expanding this out, $||\underline{v}||^2 = \underline{v}^{\mathrm{T}}\underline{v}$, the objective function can be expressed equivalently as

$$||\underline{y}||^2 - 2\underline{w}^{\mathrm{T}}X^{\mathrm{T}}\underline{y} + \underline{w}^{\mathrm{T}}X^{\mathrm{T}}X\underline{w}. \tag{6}$$

In the usual way, (because of the convexity of the objective function) the minimum of this will occur when the derivatives with respect to all the w_i are zero, or the gradient with respect to $\underline{w}$ is zero. We have

$$\nabla_{\underline{w}}\left[||\underline{y}||^2 - 2\underline{w}^{\mathrm{T}}X^{\mathrm{T}}\underline{y} + \underline{w}^{\mathrm{T}}X^{\mathrm{T}}X\underline{w}\right] = \underline{0} - 2X^{\mathrm{T}}\underline{y} + 2X^{\mathrm{T}}X\underline{w}. \tag{7}$$

Setting this equal to zero, we get that the minimum will occur at $\underline{w}$ that satisfy

$$\left[X^{\mathrm{T}}X\right]\underline{w} = X^{\mathrm{T}}\underline{y}. \tag{8}$$

Defining the matrix $\Sigma = X^{\mathrm{T}}X$, the above indicates the importance of the matrix Σ in the structure of the solution $\underline{w}$. If this matrix Σ is invertible, we get an immediate solution of $\underline{w} = \Sigma^{-1}X^{\mathrm{T}}\underline{y}$ - if it is not invertible, the situation becomes more complicated. So what is Σ? Note that by the dimensions of X (k columns, m rows), Σ is a $k \times k$ dimension matrix - the entry $\Sigma_{i,j}$ will look at, based on data, how the i-th component of data is connected to the j-th component of data. In particular, (because of the way the data vectors are stretched out along every row of X), the (i, j) component of Σ is the dot product of 'the i component of each data point' dot-producted with 'the j component of each data point'. As a result, by the law of large numbers, we have

$$\frac{1}{m}\Sigma_{i,j} \to \mathbb{E}\left[x_i x_j\right], \tag{9}$$

or

$$\frac{1}{m}\Sigma \to \mathbb{E}\left[\underline{x}\underline{x}^{\mathrm{T}}\right]. \tag{10}$$

If each component is centered on zero, then in the limit $1/m\Sigma_{i,j}$ would correspond to the **covariance** between X_i and X_j - the whole matrix then would be the covariance matrix, quantifying how the various components of the data relate to one another. In general, Σ looks at the correlation between the components of the data points. We will see this in more detail as we continue. Similarly, the term $X^T\underline{y}$ looks at how y correlates with the components of $\underline{x}$ - the i-th component of $(1/m)X^T\underline{y}$ converges to $\mathbb{E}[x_i y]$.

1.1 If $\Sigma = X^{\mathrm{T}}X$ is Invertible

We have the following result:

Naive Linear Regression Solution: For classical linear regression, if $X^{\mathrm{T}}X$ is *invertible*, then we have a unique solution of

$$\underline{\hat{w}} = \left[X^{\mathrm{T}}X\right]^{-1}X^{\mathrm{T}}\underline{y}. \tag{11}$$

As an algebraic solution, this is immediate, valid, and correct - but what does it **mean**, and what does it represent with regards to the data, and the underlying data distribution? In particular, note that $(1/m)\Sigma$ is *not* the 'true' covariance/correlation matrix, but only an estimate given the data. How reliable is it? What can we say about the distribution of $\underline{\hat{w}}$, its stability, and the errors of the related resulting model? To say anything, we need to make stronger assumptions about the underlying situation.

In particular, assume that there is some 'true' underlying linear model, i.e., $\underline{y} = X\underline{w} + \underline{\epsilon}$, where $\underline{w}$ is the 'true' weight vector, and $\underline{\epsilon}$ represents some noise vector, centered at zero, that is conditionally independent of the input vectors. Under this assumption, we have that

$$\underline{\hat{w}} = \left[X^{\mathrm{T}}X\right]^{-1} X^{\mathrm{T}}(X\underline{w} + \underline{\epsilon}) = \underline{w} + \left[X^{\mathrm{T}}X\right]^{-1} X^{\mathrm{T}}\underline{\epsilon}. \tag{12}$$

Hence we see that the estimator $\underline{\hat{w}}$ recovers the true value $\underline{w}$ - with some error term that is modulated by the specific data matrix X that occurred. We can be slightly more concrete if we make the assumption that the errors are normally distributed, and uncorrelated with each other - i.e., the components of $\underline{\epsilon}$ are i.i.d. normals with mean 0 and variance σ^2 - expressing it as a multivariate normal, $\underline{\epsilon} \sim N(\underline{0}, I\sigma^2)$. In general, for any matrix M, $M\underline{\epsilon}$ will have a distribution given by $N(0, MM^{\mathrm{T}}\sigma^2)$. Applying this here, we get that

$$\begin{aligned}
\underline{\hat{w}} &\sim \underline{w} + N\left(\underline{0}, \left(\left[X^{\mathrm{T}}X\right]^{-1} X^{\mathrm{T}}\right)\left(\left[X^{\mathrm{T}}X\right]^{-1} X^{\mathrm{T}}\right)^{\mathrm{T}}\sigma^2\right) \\
&\sim \underline{w} + N\left(\underline{0}, \left[X^{\mathrm{T}}X\right]^{-1} X^{\mathrm{T}}X\left[X^{\mathrm{T}}X\right]^{-1}\sigma^2\right) \\
&\sim \underline{w} + N\left(\underline{0}, \left[X^{\mathrm{T}}X\right]^{-1}\sigma^2\right),
\end{aligned} \tag{13}$$

which simplifies to yield

$$\underline{\hat{w}} \sim \underline{w} + N\left(\underline{0}, \Sigma^{-1}\sigma^2\right). \tag{14}$$

We see that in this case, $\underline{\hat{w}}$ is centered on the true value $\underline{w}$ - but there is some error, normally distributed, with covariance matrix given by $\Sigma^{-1}\sigma^2$ - the smaller the variance on the individual errors $\underline{\epsilon}$ the better over all, but the overall error is controlled the matrix Σ.

A Small Example: Consider, as a small example, the situation when the components of the data vectors $\underline{x}^i$ are i.i.d. normals themselves, centered at 0 with unit variance. In this case, we have that Σ/m will converge to the covariance matrix, i.e., $\Sigma/m \to I$, but then $\Sigma \approx mI$, in which case $\Sigma^{-1} \approx I/m$ - in this case, we see that the fit weight vector $\underline{\hat{w}}$ will be distributed like $\underline{\hat{w}} \sim N(\underline{w}, \sigma^2/m)$. We see that the error (variance) on the estimator should go down like $1/m$ - exactly the rate we've seen for previous estimators.

In general, the matrix Σ controls the error of the estimator around the true value $\underline{w}$. Because we are in a potentially high dimensional space, we need to be concerned with the error potentially in multiple dimensions - note that the error on some components of $\underline{\hat{w}}$ might be quite small, while the error on other components could be quite large. The 'ideal' is seen in the small example above - where Σ^{-1} scales the error in all dimensions/directions down by $1/m$. But as a general question we would like to understand the effect Σ^{-1} has on the error in all directions. This can be addressed by looking at the eigenvalues of Σ^{-1}, and therefore the eigenvalues of Σ (since Σ is assumed to be invertible).

Consider the decomposition/diagonalization of Σ as $\Sigma = QDQ^{\mathrm{T}}$ where Q is an orthonormal basis matrix of eigenvectors of Σ (each column is one if the eigenvectors of Σ), and D is the diagonal matrix of corresponding eigenvalues. Note - this is reasonable, as not only is Σ symmetric (*why?*), but all the eigenvalues of Σ must be strictly positive in this case that Σ is invertible (*why?*). In that case, we have that $\Sigma^{-1} = QD^{-1}Q^{\mathrm{T}}$, and

$$N\left(0, \Sigma^{-1}\sigma^2\right) \sim N\left(0, QD^{-1}Q^{\mathrm{T}}\sigma^2\right) \sim Q\sigma N\left(0, D^{-1}\right). \tag{15}$$

We see then that the maximum error on $\underline{\hat{w}}$ or the maximum contribution to this error in any case will be in the direction of the eigenvector of Σ corresponding to the largest term in D^{-1} - or in other words, the maximum error will occur in the direction of the eigenvector of Σ with the smallest eigenvalue. Similarly, the minimum error will occur in the direction of the eigenvector of Σ with the largest eigenvalue. This spread, the ratio of maximum error

to minimum error, is therefore equal to the ratio of the maximum eigenvalue of Σ to the minimum eigenvalue of Σ. This is known as the *condition number* of Σ:

$$\kappa(\Sigma) = \frac{\text{largest eigenvalue of } \Sigma}{\text{smallest eigenvalue of } \Sigma}. \tag{16}$$

In general then - the largest the condition number of Σ, the worse the error will be in some direction - we would like the condition number to be close to 1, corresponding to the ideal case in the example above. We can compensate for this in some sense with taking more data (assuming the condition number is not infinite) which then informs the sample complexity - how many samples we need to reduce the over all error and uncertainty in our model. We have the following result:

If m is sufficiently large, in particular

$$m \geq \Omega\left(k\frac{\kappa(\Sigma)}{\epsilon^2}\right), \tag{17}$$

then with high probability $\text{err}(\underline{\hat{w}}) \leq \epsilon^2$.

Additionally, we expect that as the noise on the error on $\underline{y}$ goes down, we should need fewer samples as well and this lower bound should decrease - in the presence of no noise at all, the weights could be perfectly reconstructed as long as we have k data points.

1.1.1 A Geometric Interpretation

How can we interpret this linear algebraic result? What do the eigenvalues and vectors of Σ 'mean' in the larger sense?

The eigenvector $\underline{v}$ that corresponds to the largest eigenvalue is effectively the unit vector that maximizes the quantity $\underline{v}^{\mathrm{T}} X^{\mathrm{T}} X \underline{v}$, or equivalently the unit vector that maximizes

$$\sum_{i=1}^{m} |\underline{x}^i . \underline{v}|^2, \tag{18}$$

i.e., the vector that maximizes the total projection onto all of the data points. This corresponds to the first principal component of the data - the direction that captures the maximum variation of the data. Viewing the data as a cloud of points in space, this vector $\underline{v}$ corresponds to the widest direction or dimension of the data, from any perspective. The eigenvector corresponding to the second largest eigenvalue similarly corresponds to the second widest dimension of the data. The eigenvector corresponding to the smallest eigenvalue of Σ represents the thinnest possible dimension of the data - the dimension that makes the data cloud as flat as possible.

If the condition number is close to 1, $\kappa(\Sigma) \approx 1$, this means that the data is roughly equally distributed in all directions. The larger the value of κ, the greater the discrepancy - there is some dimension where the data is spread out considerably, and some dimension where the data is not very spread out at all. In this way, the value of the condition number characterizes to a great extent the dimensionality of the data - even though the data may lie in some high dimensional space, the data cloud itself may be a much lower dimensional object (or close to it).

Why does this matter to the error, as seen in the previous section? In any given direction, the more data we have in that direction the more we can understand how the value y depend on $\underline{x}$ in that direction. Whatever direction is the thinnest, in terms of the distribution of the data cloud, that is the direction that we have the *least* data in, the least information about how the $\underline{x}$ are distributed, and therefore the least information about how y depends on $\underline{x}$ in that direction. This lack of information manifests itself in greater uncertainty in how y behaves, when $\underline{x}$ is varied in that direction.

1.1.2 Preconditioning Data

In the preceding subsection, we looked at how the eigenvalues and vectors of Σ relate to the geometry of the data matrix X itself - the eigenvectors correspond to the principal components of X or the directions of the greatest spread or variance in the data. Given the way that the error on $\underline{\hat{w}}$ depends on these eigenvectors and eigenvalues - it is worth taking a moment to consider transforming or preprocessing your data, to try to control this error.

In particular, imagine that you had the following data points $\underline{x}^1 = (100, 50), \underline{x}^2 = (100, 52), \underline{x}^3 = (101, 51)$. In this case, the largest eigenvalue of $X^{\mathrm{T}}X$ is $\sim 38,000$, and the smallest is ~ 1.6, giving a condition number of $\kappa(\Sigma) \approx 22,000$. This represents a *tremendous* skew - even though the data points themselves exhibit no extreme geometry themselves. They can't - there are only three of them. Why this huge difference?

The first principal component, the $\underline{v}$ that maximizes the projection onto all the data points, is effectively located at the origin of the data space. As a result, the further away the data is from the origin, the larger the value of the projection onto $\underline{v}$. And this is independent of the geometry of the data points themselves, except when it comes to where the data cloud is located in space. We can adjust for this somewhat by re-centering our data, to try to remove this tremendous dependence on where the data cloud is located.

In this case, note that the average data point $\underline{\bar{x}} = (100.333, 51)$. If we 'center' the data by subtracting off this mean, we get $\underline{x'}^1 = (-1/3, -1), \underline{x'}^2 = (-1/3, 1), \underline{x'}^3 = (2/3, 0)$. Building the data matrix out of this re-centered data we get that $(X')^{\mathrm{T}}X'$ has eigenvalues of 2 and 2/3, with a total condition number of $\kappa(\Sigma') = 3$. This represents a *massive* improvement in the relative error in various directions, with nothing more complicated than re-centering the data to have mean $\underline{0}$. Re-centering like this ensures that the principal components of the data really capture the fundamental geometry of the data, rather than simply where the cloud is sitting in space.

Important to Remember: When constructing a regression model on re-centered data in this way, it is important to remember to recenter any future data points that you want to perform prediction on!

Additionally, it is not uncommon to not only center the data at $\underline{0}$, but also normalize it all to have a total variance of 1 to try to remove some scale effects from the data.

1.2 If $\Sigma = X^{\mathrm{T}}X$ is Not Invertible or Poorly Conditioned

Much depends on the eigenvalues of Σ, as seen in the previous sections. In particular, the smaller the eigenvalues of Σ, the greater the error in the associated eigenvector directions. But stepping back even further, suppose that Σ has some eigenvalues that actually are zero - i.e., Σ has a non-trivial null space. One outcome of this is that our original equation for the weight vector,

$$\Sigma\underline{\hat{w}} = X^{\mathrm{T}}\underline{y}, \tag{19}$$

has no unique solution for $\underline{\hat{w}}$ and thus can't be solved explicitly for it.

One potential cause of this is if there are linear dependencies between the features of the data. For instance, in two dimensional data, if it is always the case that that $x_2 = 3x_1$, it isn't really true that the data is two dimensional - the data is in fact one dimensional, but embedded in a two dimensional space. In a case like this, there will always be a whole family of weight vectors that both minimize the training error and produce equivalent predictions. For instance, in this case,

$$w_1x_1 + w_2x_2 \text{ and } (w_1 + 3w_2)x_1 + 0x_2 \tag{20}$$

will produce the exact same predicted output on the training data. How to determine then which is the preferable model?

One potential way to approach this problem - identifying and responding to low dimensional data - is through principal component analysis. Low dimensional data corresponds to the data being 'flat' in some dimension, or having a principal component with an eigenvalue of 0. We could eliminate or remove these 'flat' dimensions by projecting the original k-dimensional data down into a k'-dimensional data space, effectively truncating or cutting off the components of the data corresponding to eigenvalues of 0.

If we have $\Sigma = X^{\mathrm{T}}X$ and $\Sigma = QDQ^{\mathrm{T}}$, if the eigenvalues and vectors are sorted in order of decreasing eigenvalue, the first k'-many columns of Q correspond to the 'most significant' k'-many components of the data. We can project down into this space and look at the data *only* relative to these components, by constructing a new data matrix

$$X' = XQ_{k'}. \tag{21}$$

This could be used for instance to cut off not only the dimensions where the data has zero spread (eigenvalues for Σ of 0), but potentially even cut off directions where the eigenvalues are quite small, and thus contributing greatly to the overall error (as in the previous section). *Author's Note: Some experimentation may be necessary to determine what represents an 'acceptable' cutoff point for non-zero eigenvalues.* Once the new data matrix has been constructed, regression can proceed as normal in the k'-dimensional modified data space. Again, do not forget to project any new data point in the same way when making predictions on new data.

This approach to non-invertible or poorly conditioned data is simply to truncate or cut off dimensions with poor (or non-existent) spread in the data. An alternate approach is potentially to inflate the spread of the data in these directions slightly. For instance, notice that constructing the modified Σ-matrix

$$\Sigma' = \Sigma + \lambda I, \tag{22}$$

where $\lambda > 0$ and I is the identity matrix, Σ' has all the same eigenvectors as Σ, but the eigenvalues have increased from λ_i to $\lambda_i + \lambda$ - in some sense, inflating the spread of the data in the various directions. Note, if $\lambda > 0$, the eigenvalues of Σ' will be strictly greater than 0, and therefore Σ' will be invertible even if Σ is not! Much of the mechanics of regression could theoretically proceed using this modified Σ-matrix, and we could look at the estimator solution

$$\underline{\hat{w}} = \left[\Sigma'\right]^{-1} X^{\mathrm{T}}\underline{y}, \tag{23}$$

in the hopes that it would provided a 'well-defined' estimator in this situation where the previous solution would fail. We will justify this more strongly in the following section.

2 Ridge Regression : Model Preferencing

We return briefly to this question of what to do when the solution $\underline{w}$ to

$$\Sigma\underline{w} = X^{\mathrm{T}}\underline{y} \tag{24}$$

is not well-defined. In this case, we may frequently run into cases where multiple solutions $\underline{w}$ minimize the training error and give identical predictions. How can we determine which one is 'best'? For example, suppose that on the training data $x_3 = x_1 + x_2$. Which of the two following models is preferred:

$$10x_1 + 10x_2 + 10x_3 \text{ versus } 20x_1 + 20x_2 + 0x_3? \tag{25}$$

One way of approaching this might be to think about the possible error that would result in tiny fluctuations in x_1, x_2, x_3. Any small error on a measurement of x_1 is going to be scaled up, contributing a variance of $\propto 10^2$ to

the final answer in the first model, and $\propto 20^2$ to the final answer in the second model. In this way, we could argue that the total variance on the first model would be $\propto 300$, while the total variance on the second model would be $\propto 400 + 400 = 800$. Because the first model has generally smaller coefficients, error in individual components gets translated into much smaller error over all. This gives us a way of preferring one model over another - give preference to the model with the typically smaller coefficients.

This is one way of justifying the notion of **Ridge Regression**, an alternate way to derive a predictive model than simply minimizing the training error:

Ridge Regression: For a value $\lambda > 0$, find $\underline{\hat{w}}$ to solve the following:

$$\min_{\underline{w}} \; ||\underline{y} - X\underline{w}||^2 + \lambda||\underline{w}||_2^2. \tag{26}$$

Ridge regression equips our objective function with an additional cost, penalizing models with large norms or large coefficients in the weight vector. The choice of λ effectively determines how much large coefficients should be penalized - how much should you try to focus on minimizing training error versus minimizing the final variance / error in the model.

This is a nicely posed problem in a lot of ways, but perhaps most importantly in that we can solve it explicitly. Note that we can compute the gradient with respect to $\underline{w}$ as

$$\nabla_{\underline{w}} \left[||\underline{y} - X\underline{w}||^2 + \lambda||\underline{w}||_2^2\right] = -2X^{\mathrm{T}}\underline{y} + 2X^{\mathrm{T}}X + 2\lambda\underline{w}. \tag{27}$$

Solving this for $\underline{w}$ yields the following.

The **Ridge Regression** model is solved by

$$\underline{\hat{w}} = \left[X^{\mathrm{T}}X + \lambda I\right]^{-1} X^{\mathrm{T}}\underline{y}. \tag{28}$$

As noted previously, as long as $\lambda > 0$, then $X^{\mathrm{T}}X + \lambda I$ will always be invertible so the above solution will always be well-defined. One way to understand the solution to ridge regression is exactly the discussion of the previous section - it is effectively inflating the eigenvalues in various directions, which not only renders $\underline{w}$ well-defined and solvable, but also potentially controls some of the error/poor conditioning in the original solution. It can be shown, for instance, that if $\kappa(X^{\mathrm{T}}X) > 1$, then for $\lambda > 0$ we have

$$\kappa(X^{\mathrm{T}}X + \lambda I) < \kappa(X^{\mathrm{T}}X), \tag{29}$$

thus we effectively reduce any poor conditioning on the data, and potentially not only improve our model (through decreased error in poorly conditioned directions) but reduce how many samples we need to get a good model.

But we need to consider the following problem - modifying the data matrix in this way, augmenting it with the λI term, we are literally changing the data (or at least its represented distribution). What is the effect of this? If we change the problem too much (weight the optimization problem too much in favor of reducing $||\underline{w}||$), we risk generating a solution that has little to noting to do with our original regression problem.

Suppose that there is some true underlying linear model, $\underline{y} = X\underline{w}$. One approach would be to ask - how much does $\underline{\hat{w}}$ differ from $\underline{w}$ for the solution to the ridge regression model? For naive regression, simply minimizing the training error, we saw that $\underline{\hat{h}}$ should recover the true value of $\underline{w}$ (potentially with some small noise). What can be said here?

Substituting in to the ridge regression solution, we get

$$\underline{\hat{w}} = \left[X^{\mathrm{T}}X + \lambda I\right]^{-1} X^{\mathrm{T}}X\underline{w}. \tag{30}$$

The question of how much $\underline{\hat{w}}$ differs from $\underline{w}$ then amounts to asking what is the effect of multiplying by the matrix $[\Sigma + \lambda I]^{-1}\Sigma$.

To understand this matrix, note again that any eigenvector $\underline{v}^i$ of Σ (with eigenvalue λ_i) will also be an eigenvector of $\Sigma + \lambda I$, since

$$[\Sigma + \lambda I]\,\underline{v}^i = \Sigma\underline{v}^i + \lambda I\underline{v}^i = \lambda_i\underline{v}^i + \lambda\underline{v}^i = (\lambda_i + \lambda)\underline{v}^i, \tag{31}$$

and it will have the eigenvalue $\lambda_i + \lambda$.

Taking the orthonormal eigenvector basis of Σ, we can express both Σ and $\Sigma + \lambda I$ with respect to this eigenbasis in the following way:

$$\begin{aligned} \Sigma &= \sum_{i=1}^{k} \lambda_i \underline{v}^i (\underline{v}_i)^{\mathrm{T}} \\ \Sigma + \lambda I &= \sum_{i=1}^{k} (\lambda_i + \lambda) \underline{v}^i (\underline{v}_i)^{\mathrm{T}}. \end{aligned} \tag{32}$$

Note that the eigenvectors of $[\Sigma + \lambda I]^{-1}$ will be exactly the same, with corresponding eigenvalues $1/(\lambda_i + \lambda)$. In which case we get a similar expansion for the inverse, and can express the product

$$\begin{aligned} [\Sigma + \lambda I]^{-1}\,\Sigma &= \left(\sum_{i=1}^{k} \frac{1}{\lambda_i + \lambda} \underline{v}^i (\underline{v}_i)^{\mathrm{T}} \right) \left(\sum_{i=1}^{k} \lambda_i \underline{v}^i (\underline{v}_i)^{\mathrm{T}} \right) \\ &= \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\lambda_j}{\lambda_i + \lambda} \underline{v}^i (\underline{v}_i)^{\mathrm{T}} \underline{v}^j (\underline{v}_j)^{\mathrm{T}} \end{aligned} \tag{33}$$

However, recall that the $\{\underline{v}^i\}$ are *orthonormal* - that is $(\underline{v}^j)^{\mathrm{T}}\underline{v}^i$ is 0 if $i \neq j$ and 1 if $i = j$. Using this fact, the above simplifies to

$$[\Sigma + \lambda I]^{-1}\,\Sigma = \sum_{i=1}^{k} \frac{\lambda_i}{\lambda_i + \lambda} \underline{v}^i (\underline{v}_i)^{\mathrm{T}}. \tag{34}$$

The effective of this matrix is a kind of shrinkage - we have that $\lambda_i/(\lambda_i + \lambda) < 1$, $[\Sigma + \lambda I]^{-1}\,\Sigma\underline{w}$ is effectively going to shrink $\underline{w}$ in all the directions associated with $\underline{v}^i$, and the amount of shrinkage is going to be determined by how λ_i compares to λ. In the directions where λ_i is quite large (corresponding to directions in the original data set with large spread/variance, and lots of information), the effect of shrinkage by λ will be quite small. In directions where λ_i is quite small (corresponding to directions in the original data set with small spread/variance and low information), the effect of shrinkage by λ will be quite pronounced - ridge regression is effectively discounting or devaluing the components of $\underline{w}$ in these directions.

But in general, we see that

$$\underline{\hat{w}} = [\Sigma + \lambda I]^{-1}\,\Sigma\underline{w} = \sum_{i=1}^{k} \frac{\lambda_i}{\lambda_i + \lambda} \left(\underline{w}.\underline{v}^i\right) \underline{v}^i, \tag{35}$$

and therefore for any $\underline{v}^i$ we have

$$\underline{\hat{w}}.\underline{v}^i = \frac{\lambda_i}{\lambda_i + \lambda}(\underline{w}.\underline{v}^i). \tag{36}$$

Hence we see that in every direction $\underline{v}^i$, the ridge regression solution projected in that direction mostly matches the true solution $\underline{w}$ projected in that direction - scaled appropriately according to λ. So the ridge regression solution mostly matches the 'real' value $\underline{w}$ - deviating more and more as λ increases, with $\underline{\hat{w}} \to 0$ in the limit as $\lambda \to \infty$. But this gives us confidence that for λ not too large, $\underline{\hat{w}}$ should mostly recover a 'true' linear model.

3 Lasso Regression : Eliminating Irrelevant Features

Ridge regression can be thought of as a way of biasing weights away from being quite large - one way to think of this is that few large weights contribute a lot more to over all error than many small weights. But rather than forcing weights away from being large, it can also be useful to consider forcing them towards being small, if not zero all together. The main motivating idea here is that of *irrelevant features*, features that do not factor into the outcome variable at all. For instance, in a three dimension data space (x_1, x_2, x_3), we might have a model given by $y = w_1 x_1$ - in this case, the x_2, x_3 features are irrelevant or confounding.

Ideally, we would like to be able to eliminate irrelevant variables from our model - effectively setting the weight for those variables to be zero. The problem is that this irrelevancy may not be obvious in the training data - this is very similar to the pruning / overfitting discussion for decision trees, and the attempt in that case to eliminate mostly irrelevant variables from consideration for the overall data set.

One sort of brute-force approach to this might be to select some set of features, $S \subset \{1, \ldots, k\}$, and effectively hardcode the weights outside of this set to be zero. We can then apply any of the previous regression techniques looking only at the S-features of the data set:

Best Subset Feature Selection: For $k' < k$, choose a subset $S \subset \{1, \ldots, k\}$ and solve

$$\min_{\underline{w}} ||\underline{y} - X\underline{w}||^2 \text{ such that } \forall i \notin S : w_i = 0. \tag{37}$$

Of all sets of size k', return $\underline{w}_S$ of smallest error.

The problem with this brute force approach of setting some set of weights to zero is that there are a lot of choices for which weights to set to zero. If there are k features, there are k choose k' ways of choosing k' features - out of 100 features, there are $5.39 * 10^{20}$ ways to choose a subset of 20. Even for reasonable sized problems, finding the best subset of features can be computationally quite difficult.

There is a direct analog here with trying to find short decision trees on data - finding the shortest decision tree to achieve minimal training error is a computationally hard problem.

Much like in the case of decision trees then, we can apply a heuristic that while not guaranteeing the *best* model, will create a bias or pressure towards short/simple models. In this case, we again add a cost to our objective function that penalizes weights that are far from zero. In particular, we have the Lasso regression / feature selection model:

Lasso: Find the weight vector $\underline{\hat{w}}$ that solves

$$\min_{\underline{w}} ||\underline{y} - X\underline{w}||^2 + \lambda ||\underline{w}||_1, \tag{38}$$

where $||\underline{w}||_1 = \sum_i |w_i|$.

Note the difference between this and ridge regression - ridge regression applies a penalty proportional to w_i^2 to each weight, while lasso applies a penalty proportional to $|w_i|$ for each weight. If you compare the graphs of these two functions, close to 0 we have that $|w| > w^2$, which means that the lasso model or L^1 norm is applying a larger penalty for deviating from 0. The effect of this larger penalty is that the solution to lasso problems typically forces more coordinates of $\underline{\hat{w}}$ to be zero than ridge regression will. In general, the L^1 norm is better at distinguishing sparse

vectors (lots of zero components) than the L^2 norm. As an example of this, consider in N dimensions,

$$\begin{aligned} \underline{v} &= (1, 0, 0, \ldots, 0) \\ \underline{u} &= (1/\sqrt{N}, 1/\sqrt{N}, \ldots, 1/\sqrt{N}). \end{aligned} \tag{39}$$

In this case, we see that $||\underline{v}||_2^2 = ||\underline{u}||_2^2 = 1$, but that $||\underline{v}||_1 = 1$ and $||\underline{u}||_1 = \sqrt{N}$ - the L^1 norm in this case gives preference to the sparser vector.

To illustrate the point further, consider a toy example where $X^T X$ is the identity matrix, and there is some true linear model such that $\underline{y} = X\underline{w}$. In that case, the objective function becomes

$$||X\underline{w} - X\underline{\hat{w}}||^2 + \lambda||\underline{\hat{w}}||_1, \tag{40}$$

or

$$(X(\underline{w} - \underline{\hat{w}}))^{\mathrm{T}}(X(\underline{w} - \underline{\hat{w}})) + \lambda \sum_{i=1}^{k} |\hat{w}_i|. \tag{41}$$

Observing that $(X\underline{v})^{\mathrm{T}}(X\underline{v}) = \underline{v}^{\mathrm{T}}X^{\mathrm{T}}X\underline{v} = \underline{v}^{\mathrm{T}}\underline{v} = ||\underline{v}||^2$, the above reduces to

$$\sum_{i=1}^{k}(w_i - \hat{w}_i)^2 + \lambda \sum_{i=1}^{k} \hat{w}_i \tag{42}$$

or

$$\sum_{i=1}^{k} \left[(w_i - \hat{w}_i)^2 + \lambda|\hat{w}_i|\right]. \tag{43}$$

To minimize the whole thing, it suffices to minimize each term of the sum individually, i.e., find $\hat{w}_i$ to minimize $(w_i - \hat{w}_i)^2 + \lambda|\hat{w}_i|$. This takes a couple of special cases to solve but you can show that this solves to

$$\hat{w}_i = \begin{cases} w_i - \lambda/2 & \text{if } w_i > \lambda/2 \\ w_i + \lambda/2 & \text{if } w_i < -\lambda/2 \\ 0 & \text{if } -\lambda/2 < w_i < \lambda/2. \end{cases} \tag{44}$$

In this case, we see that the Lasso solution is effectively performing a thresholding - if the 'true' weight is close to 0, then it will force the Lasso weight $\hat{w}_i$ to be exactly 0. The value of this threshold depends on λ.

It is worth comparing the corresponding ridge regression solution, which would be that $\hat{w}_i = w_i/(1 + \lambda)$. In both cases, we see a general shrinkage, pushing the weights towards zero - but the Lasso solution has a much more dramatic effect, forcing the weights to be exactly zero past a certain point.

As a general heuristic then, Lasso regression can be an effective tool for identifying and eliminating irrelevant features. There is no guarantee that this will be the 'best' feature subset - but similar to ID3 for decision trees, the Lasso feature set and regression solution is highly effective in practice.

3.1 Lasso Guarantees: Support Recovery

What does the solution to Lasso look like? Ideally with lots of data the recovered $\underline{\hat{w}}$ should look a lot like $\underline{w}$, similar to as seen was in naive regression and ridge regression. But we definitely expect the regularization term to pull $\underline{\hat{w}}$ away from $\underline{w}$ - but how much, and how do they compare?

In the case of ridge regression, we want to minimize an objective function of the form

$$||\underline{y} - X\underline{w'}||^2 + \lambda||\underline{w'}||_2^2. \tag{45}$$

In this usual way, this is minimized when the gradient of the objective function is zero, or

$$\begin{aligned}\nabla_{\underline{w'}}\left[||\underline{y}-X\underline{w'}||^2+\lambda||\underline{w'}||_2^2\right] &= \nabla_{\underline{w'}}\left[||\underline{y}||^2-2\underline{w'}^{\mathrm{T}}X^{\mathrm{T}}\underline{y}+\underline{w'}^{\mathrm{T}}X^{\mathrm{T}}X\underline{w'}+\lambda\underline{w'}^{\mathrm{T}}\underline{w'}\right]\\ &= \nabla_{\underline{w'}}\left[-2\underline{w'}^{\mathrm{T}}X^{\mathrm{T}}\underline{y}+\underline{w'}^{\mathrm{T}}X^{\mathrm{T}}X\underline{w'}+\lambda\underline{w'}^{\mathrm{T}}\underline{w'}\right]\\ &= -2X^{\mathrm{T}}\underline{y}+2X^{\mathrm{T}}X\underline{w'}+2\lambda I\underline{w'}=0,\end{aligned} \tag{46}$$

or

$$-X^{\mathrm{T}}\underline{y}+X^{\mathrm{T}}X\underline{w'}+\lambda I\underline{w'}=0, \tag{47}$$

which has a unique solution of

$$\underline{\hat{w}}=\left[X^{\mathrm{T}}X+\lambda I\right]^{-1}\left[X^{\mathrm{T}}\underline{y}\right]. \tag{48}$$

For Lasso, we have a similar property, that the objective function is minimized at a zero of the *sub*-gradient of the objective function. The subgradient generalizes the idea of the derivative. Instead of a single value representing the slope of a tangent line at a point, it represents the *set* of slopes a tangent line might have while bounding a function from below - the subgradient of $|x|$ would be given as $\{-1\}$ if $x<0$, $\{1\}$ if $x>0$, and if $x=0$, the subgradient would be the set of values between -1 and 1, $[-1,1]$ (ensuring that the tangent line at this point lies below the function). The subgradient as a value may not be uniquely defined, but it is limited. We can express this generally as the subgradient of $|x|$ being $\mathrm{sign}(x)$, understanding $\mathrm{sign}(0)$ to be $[-1,1]$. For the Lasso objective function, the minimum occurs when the subgradient contains zero (or generalizing the gradient being zero):

$$0\in -X^{\mathrm{T}}\underline{y}+X^{\mathrm{T}}X\underline{w'}+\lambda\mathrm{sign}(\underline{w'}), \tag{49}$$

or

$$\frac{1}{\lambda}X^{\mathrm{T}}\left[\underline{y}-X\underline{w'}\right]\in\mathrm{sign}(\underline{w'}). \tag{50}$$

There are numerous difficulties here: the above is a set relation, and therefore difficult to 'solve'; the above is also highly non-linear in terms of $\underline{w'}$; there may not even be a unique solution for $\underline{w'}$ (*author's note: frequently there is a unique solution for* $\underline{w'}$)! Can anything be said at this point?

Consider the following question, related to the motivation of Lasso generally: does Lasso correctly identify relevant features, and correctly exclude irrelevant features? Given a true linear model $\underline{y}=X\underline{w}+\underline{\epsilon}$, the nightmare scenario is that Lasso identifies relevant features as irrelevant $w_i\neq 0$ and $w_i'=0$ and identifies irrelevant features as relevant $w_i'\neq 0$ and $w_i=0$. Ideally, we would like to ensure that the solution $\underline{w'}$ recovers the **support** of $\underline{w}$, and even more ideally the sign as well, i.e., $\mathrm{sign}(\underline{w})=\mathrm{sign}(\underline{w'})$.We can express this concisely in the following way:

$$\begin{aligned}&\textit{Is there a Lasso solution } \underline{w'} \textit{ such that}\\ &\mathrm{sign}(\underline{w})=\mathrm{sign}(\underline{w'})\\ &\frac{1}{\lambda}X^{\mathrm{T}}\left[\underline{y}-X\underline{w'}\right]\in\mathrm{sign}(\underline{w})?\end{aligned} \tag{51}$$

Denote the **support** of $\underline{w}$ to be S^*. Note that to solve the above, we really only need to worry about the coordinates of $\underline{w}^{*\prime}$ on S^*, as the coordinates off this set must be zero in order to satisfy the above.

Under certain conditions, the above is answered in the positive. In particular, the proof technique is to decompose the second relation by looking at the components of $(1/\lambda)X^{\mathrm{T}}\left[\underline{y}-X\underline{w'}\right]$ where $w_i\neq 0$ so that the relationship becomes an equality. We can decompose the above relationships in the following way, projecting onto individual coordinates:

$$\begin{aligned}&\mathrm{sign}(\underline{w'})=\mathrm{sign}(\underline{w})\\ \forall j\in S^*:&\frac{1}{\lambda}X_j^{\mathrm{T}}\left[\underline{y}-X_{S^*}\underline{w'}_{S^*}\right]=\mathrm{sign}(w_j)\\ \forall j\notin S^*:&\frac{1}{\lambda}X_j^{\mathrm{T}}\left[\underline{y}-X_{S^*}\underline{w'}_{S^*}\right]\in[-1,1].\end{aligned} \tag{52}$$

Note - we can bring these componentwise properties together in the following way, recalling that the infinity norm $||\underline{v}||_\infty$ is the maximum absolute value of any component of $\underline{v}$:

$$\begin{aligned} &\text{sign}(\underline{w'}) = \text{sign}(\underline{w}) \\ &\frac{1}{\lambda} X_{S^*}^{\mathrm{T}} \left[\underline{y} - X_{S^*} \underline{w'}_{S^*}\right] = \text{sign}(\underline{w}_{S^*}) \\ &||\frac{1}{\lambda} X_{S^*}^{\mathrm{T}} \left[\underline{y} - X_{S^*} \underline{w'}_{S^*}\right]||_\infty \leq 1. \end{aligned} \tag{53}$$

If $X_{S^*}^{\mathrm{T}} X_{S^*}$ is invertible, the second equation solves for

$$\underline{w'}_{S^*} = \left[X_{S^*}^{\mathrm{T}} X_{S^*}\right]^{-1} \left[X_{S^*}^{\mathrm{T}} \underline{y} - \lambda \text{sign}(\underline{w}_{S^*})\right]. \tag{54}$$

As noted, the components of $\underline{w'}$ *off* S^* need to simply be zero. The answer to the question then becomes - is this $\underline{w'}$ a valid solution? That is, does it hold that

$$\begin{aligned} &\text{sign}\left(\left[X_{S^*}^{\mathrm{T}} X_{S^*}\right]^{-1} \left[X_{S^*}^{\mathrm{T}} \underline{y} - \lambda \text{sign}(\underline{w}_{S^*})\right]\right) = \text{sign}(\underline{w}_{S^*}) \\ &\frac{1}{\lambda} ||X_{\overline{S^*}}^{\mathrm{T}} \left[\underline{y} - X_{S^*} \left[X_{S^*}^{\mathrm{T}} X_{S^*}\right]^{-1} \left[X_{S^*}^{\mathrm{T}} \underline{y} - \lambda \text{sign}(\underline{w}_{S^*})\right]\right]||_\infty \leq 1. \end{aligned} \tag{55}$$

Conditions for Support Recovery: Assuming the following,

- the noise $\underline{\epsilon}$ is normal, the components independent, with mean 0 and variance σ^2,
- the X data is scaled such that $||X_j||_2^2 \leq k$,
- $X_{S^*}^{\mathrm{T}} X_{S^*}$ is invertible,

the following conditions ensure that the conditions in Eq. (53) are satisfied, and there is a Lasso solution $\underline{w'}$ that recovers the sign and support of the true underlying weight vector $\underline{w}$:

- Mutual Incoherence: *irrelevant variables do not significantly correlate with relevant variables*

$$\forall j \notin S^* : ||(X_{S^*}^{\mathrm{T}} X_{S^*})^{-1} X_{S^*}^{\mathrm{T}} X_j||_1 \text{ is sufficiently small} \tag{56}$$

- Minimum Spread: *the data on the support is sufficiently spread*

$$\lambda_{\min}(X_{S^*}^{\mathrm{T}} X_{S^*}) \text{ is sufficiently large} \tag{57}$$

- Minimum Signal: *the smallest dependence on any relevant variable is above some threshold*

$$\forall j \in S^* : |w_j| \text{ is sufficiently large} \tag{58}$$

In each case, the conditions may be more precisely quantified, but they are intuitive: i) the irrelevant variables can't 'appear' relevant by predicting the relevant variables, ii) the data on the supported variables must have sufficient spread to provide useful information, and iii) the 'signal' for each relevant variable must be sufficiently large to be detected. Intuitively, the 'sufficiently large' condition on this third case should depend on the amount of incoherence (the stronger the signal, the less incoherence you need), the spread of the data (the stronger the signal, the less data you need to detect it), and the value of λ (as seen previously, the larger the value of λ, the more coordinates are forced to zero).

For each of the above conditions, the condition is more likely to be satisfied the more data we have - thus, even though we do not know S^* and cannot verify the conditions ourselves, with more data we can be more and more confident that the conditions are satisfied. This draws a direct line connection to sample complexity: with sufficiently many training data points, we have that the support and sign of $\underline{w}$ is recovered with high probability. If we want to guarantee not only small error $\text{err}(\underline{\hat{w}}) \leq \epsilon^2$ and that the solution $\underline{\hat{w}}$ recovers the support and sign of $\underline{w}$, we need at least

$$m \geq \Omega\left(|S^*|\frac{\sigma^2 \ln k}{\epsilon^2 C^2}\right) \tag{59}$$

data points, where C represents the lower bound on the minimum signal.