Note: Ridge Regression

Karna Mendonca, Joe Zou, Tejasvi Kothapalli, Brian Zhu

Introduction

In machine learning, we define **regression** as the task of predicting quantitative values based on given data points. In other words, regression is essentially fitting curves to a set of data. We've seen examples of basic linear regression, as well as how we can use the idea of **features** to use linear regression on more complex data.

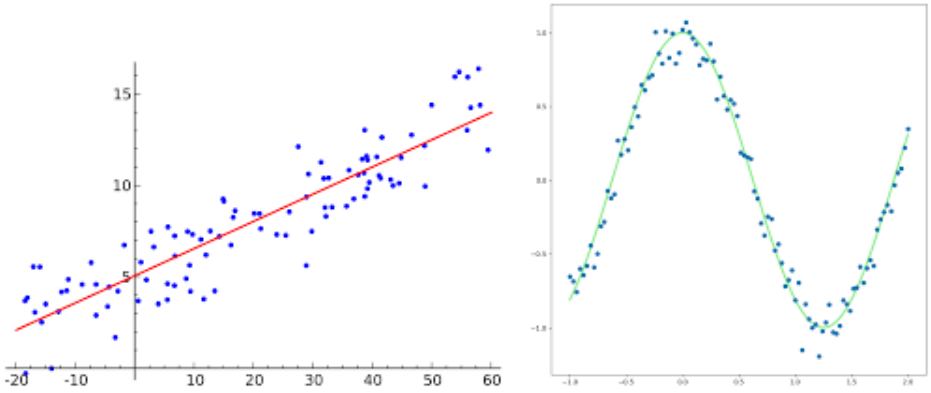


Figure 1: Linear and Polynomial Regression

In today's lesson, we focus on the idea of **noise** in our data, how it can corrupt linear regression, and how we can best design our model to deal with noisy data.

1 Ordinary Least Squares (Review)

Let's review our typical approach to linear regression: Ordinary Least Squares (OLS). We need to know both how to construct our model, which we can do through features, as well as how to evaluate how well our model is doing.

1.1 Featurization

Let's suppose that for each of our data points, the y-value depends on a polynomial of the x-value:

$$y_i = w_d x^d + w_{d-1} x^{d-1} + \cdots + w_2 x^2 + w_1 x + w_0$$

where $w_0 \dots w_d \in \mathbb{R}$ are constants

How can we model a high degree polynomial through linear regression? By **featurizing** it. Let's let each of our data points be a vector of different degree terms, and our weights be the coefficients.

$$\vec{x_i} = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \\ \vdots \\ x_i^d \end{bmatrix} \qquad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Notice how $\vec{x_i}^\top w = w_d x^d + \cdots + w_2 x^2 + w_1 x + w_0 = y_i$. Now, if we want to process a bunch of data at once, we can collect all our data into a matrix.

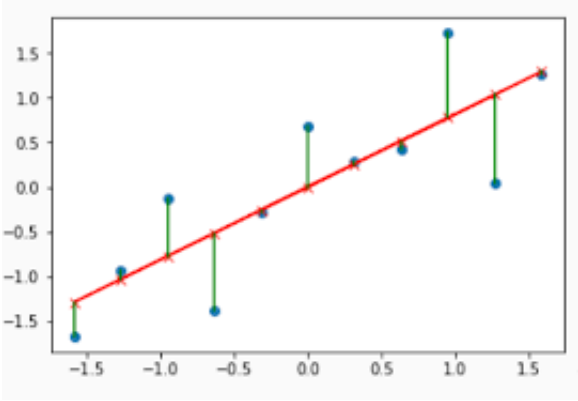
$$X = \begin{bmatrix} -\vec{x_1}^\top - \\ -\vec{x_2}^\top - \\ \vdots \\ -\vec{x_n}^\top - \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^d \end{bmatrix}$$

Notice, each of our rows represents a data point, and each of our columns represent a polynomial "feature". This allow us to represent our polynomial regression process in the equation, $Xw = y$:

$$\begin{bmatrix} -\vec{x_1}^\top - \\ -\vec{x_2}^\top - \\ \vdots \\ -\vec{x_n}^\top - \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{bmatrix}$$

1.2 Least Squares

Now that we know how to construct our model, how can we evaluate how well our model is performing? An easy way to measure is to look at the size residuals:



We want to punish points that stray further away from our prediction far more than those close to our prediciton. **Squared error** accomplishes this well: Error $= \sum_{i=0}^{n} (y_i - x_i^\top w)^2 = \|y - Xw\|_2^2$. So our problem becomes a matter of choosing a weight vector that minimizes the squared error.

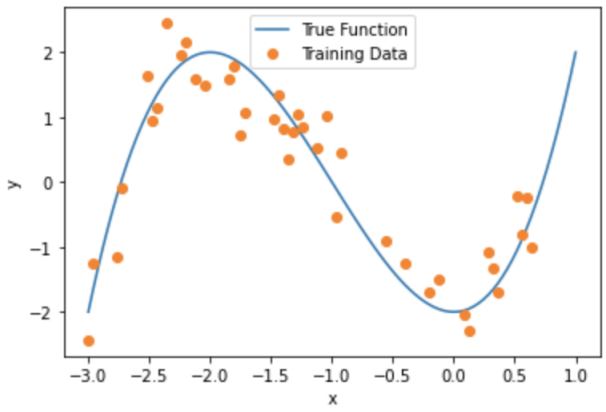
$$\hat{w} = \arg\min_w \|y - Xw\|_2^2$$

The solution[1] for the optimal weight is $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.

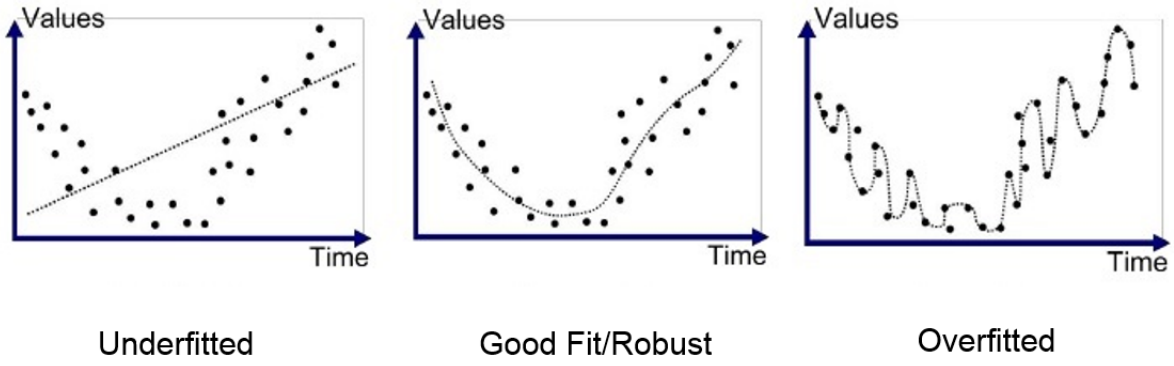
[1]The proof for this solution is a little involved, but is laid out in the appendix for those curious.

2 Noise and Overfitting

In the real world, using Ordinary Least Squares is not always so simple. There will always be slight fluctuations in the data that cause it to not perfectly follow the underlying function. We refer to this as **noise** in the data.



This alone isn't a huge problem for OLS. However, we often won't know anything about the underlying function behind the data. The question then becomes, how many features do we use?



If we don't use enough features, our model is not powerful enough to represent the data well, which we refer to as **underfitting**. We see that increasing our model's complexity does improve our predictions–up to a point. If our model is too complex, its predictions are often influenced by the random noise in the data. We refer to this phenomenon as **overfitting**.

So if we don't know the underlying function, how can we create a model that is complex enough to capture trends in the data, while avoiding overfitting?

3 Ridge Regression

The key idea of Ridge Regression is to improve Ordinary Least Squares by putting a penalty on complex models. That way, we don't have to worry about whether we are adding too many features, as the model will automatically punish more complex models. We can use the l_2 norm of the weight vector as a measure of our model's complexity. If a model is more complex, more features are being actively used, and thus, the norm of the weight vector is higher. This boils down to the optimization problem:

$$\hat{w} = \arg\min_{w} \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

The closed-form solution[2] for Ridge Regression is $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$. Using this learned $\hat{\mathbf{w}}$ vector, we can make predictions the same way as before: $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$. So what does Ridge Regression look like in action?

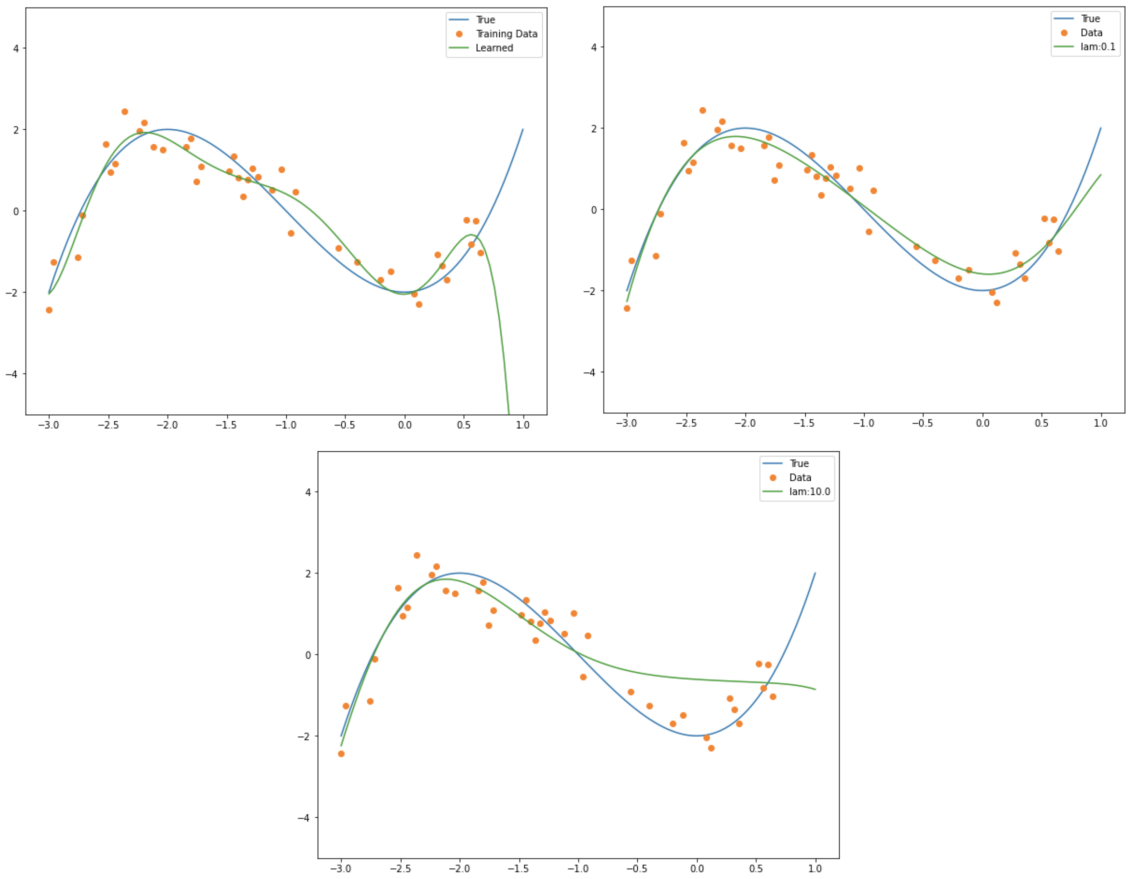


Figure 2: Ridge Regression: $\lambda = \{0 \text{ (OLS)}, 0.1, 10\}$

Notice that when $\lambda = 0$, the problem simplifies to OLS. As we increase λ, the function becomes less susceptible to noise, which we refer to as **regularization**. λ is what we refer to as a **hyper-parameter**; an inherent facet of the model that should be chosen by the engineer, ideally through cross-validation. In the above case, $\lambda = 0.1$ is clearly the best option.

[2]Proof provided in the appendix

3.1 Implementing Ridge Regression with Scikit-learn

Scikit-learn is an open-source machine learning package for Python. We will make use of Scikit-learn's sklearn.linear_model.Ridge class to perform ridge regression. Given a properly constructed X matrix and y vector, here is an example of how to use Scikit-learn's Ridge class:

```
#Specify a lambda to use
lambd = 0.1
ridge = Ridge(alpha=lambd)
ridge.fit(X, y)
y_pred = ridge.predict(X)
```

For more information, reference the documentation at: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html`

4 Bias-Variance Trade-Off for Ridge Regression

You should already be familiar with the idea of bias-variance trade-off. Now, let's observe the bias-variance trade-off in the context of ridge regression. In general, a high bias (also known as underfitting) refers to when the model's underlying assumptions cause the model to not fully capture the data. In the case of ridge regression, a high bias occurs when λ is too large. A large λ forces the $\hat{w}$ to approach the zero vector, when in reality the true w is probably not the zero vector. Thus a large lambda yields a model that poorly captures the data. A high variance (overfitting), on the other hand, occurs when the model is too sensitive to small changes in the data. In the case of ridge regression, this occurs when λ is close to zero. The smaller λ is, the less the penalty on the $|\hat{w}|_2^2$ term. As a result the elements in w will vary a lot more to fit the noise in the data. Thus, the model will fit the training data well but will still perform poorly on unseen data.

Recall that the goal of bias-variance trade-off is to pick a model with parameters that lead to the lowest bias^2 + variance. We now understand that as λ increases, bias will increase and variance will decrease. Thus, the goal in ridge regression is to pick the optimal λ that achieves the minimum bias^2 + variance. Observe figure 3: this is not exactly how a bias-variance plot will look for ridge regression, but it shows the general trends of bias and variance as λ varies. Also note that the minimum test error occurs at the choice of lambda that minimizes bias^2 + variance.

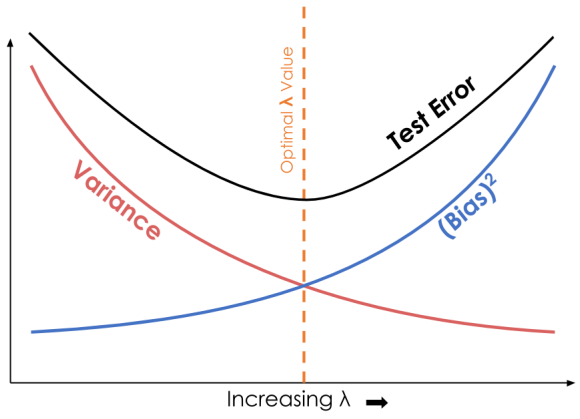


Figure 3: Bias-Variance Plot for Ridge Regression

5 Eigenvalue Perspective of Ridge Regression

In EE16A and EE16B, we've seen that studying the eigenvalues can tell us a lot about the stability of a particular system. Let's see what we can learn about OLS and Ridge Regression by looking at the eigenvalues of certain components.

5.1 Instability in OLS

Recall the solution for OLS:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-\mathbf{1}} \mathbf{X}^\top \mathbf{y}$$

Let's look specifically at the term $X^\top X$. Matrices that can be expressed in the form $X^\top X$ are known is as positive semi-definite matrices. While these matrices are beyond the scope of this course, we will utilize two important properties:

1. $X^\top X$ is symmetric, i.e. $(X^\top X)^\top = X^\top X$.
2. $X^\top X$ has non-negative eigenvalues

Because $X^\top X$ is symmetric, it is also diagonalizable (you will learn more about symmetric matrices later in EECS 16B), so we can decompose it into the form

$$X^\top X = V \Lambda V^{-1}$$

Where $V = \begin{bmatrix} \vec{v}_1 & \cdots \vec{v}_n \end{bmatrix}$ is a matrix whose columns are the eigenvectors of $X^\top X$, and Λ is the diagonal matrix that contains the eigenvalues. Since λ can be used to refer to both eigenvalues and the hyperparameter in ridge regression, we will add a tilde on top of eigenvalues (i.e. $\tilde{\lambda}_i$) to help distinguish between the two. Using this notation, we have:

$$\Lambda = \begin{bmatrix} \tilde{\lambda}_1 & 0 & \cdots & 0 \\ 0 & \tilde{\lambda}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\lambda}_n \end{bmatrix}$$

Now let's express $(X^\top X)^{-1}$ in terms of V and Λ:

$$(X^\top X)^{-1} = V^{-1} \Lambda^{-1} V$$

We can verify that this is true because $(X^\top X)^{-1}(X^\top X) = V\Lambda^{-1}V^{-1}V\Lambda V^{-1} = I$. Notice that terms will cancel out since for any invertible matrix A we have $A^{-1}A = AA^{-1} = I$. Let's examine Λ^{-1} in particular. Since Λ is a diagonal matrix, its inverse will also have a special form:

$$\Lambda^{-1} = \begin{bmatrix} \frac{1}{\tilde{\lambda}_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\tilde{\lambda}_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{\tilde{\lambda}_n} \end{bmatrix}$$

Notice that we are now taking the reciprocals of the eigenvalues. Recall that since $X^\top X$ is positive semi-definite, $\tilde{\lambda}_i \geq 0, \forall i$. Pay attention to $\geq$. It also implies that it is possible for the eigenvalues to be 0! Taking a look at Λ^{-1} again, we see that if any of the $\tilde{\lambda}_i$'s are 0 or even close to 0, it would

cause its reciprocal to be very large. So if we substitute $(X^\top X)^{-1}$ in terms of V and Λ back into the closed-form solution of OLS:

$$\hat{w} = V\Lambda^{-1}V^{-1}X^\top y$$

If some of the $\tilde{\lambda}_i$ are extremely small, then the same $\frac{1}{\tilde{\lambda}_i}$ would become extremely large. This means a very small amount of noise in our y vector can cause a huge amount amount of change in our calculated weight vector, $\hat{w}$. Thus, we can see the mathematical perspective to overfitting:

If the eigenvalues of $X^\top X$ are very small, our solution to OLS becomes unstable.

How often does this issue come up then? Let's graph the eigenvalues of a matrix from largest to smallest:

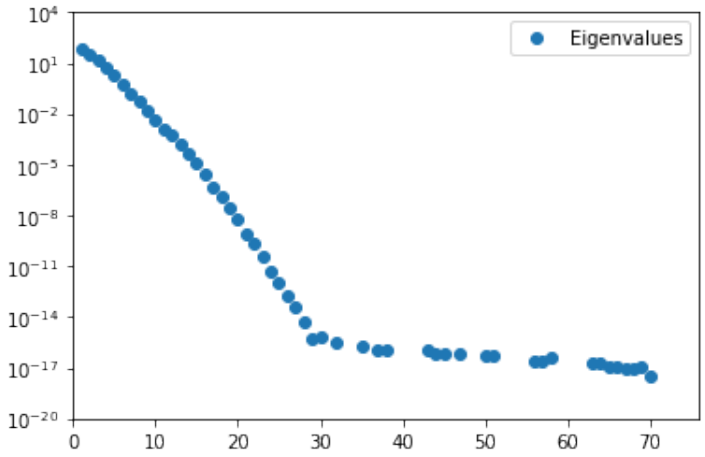


Figure 4: Eigenvalues of a matrix where 50 points are randomly sampled in the interval [-1, 1] and lifted into a 75 degree polynomial space

Pay attention to how the y axis is on a log scale, which means the eigenvalues of $X^\top X$ are dropping very sharply. Thus, we see that increasing the dimension or complexity of a model will cause stability issues in OLS.

5.2 Ridge Regression as an Eigenvalue Hack:

Now let's look at our solution for Ridge Regression in Figure 4.

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

Let's graph and compare the eigenvalues of $X^\top X + \lambda I$ and $X^\top X$. In Figure 4, the orange dots refer to the eigenvalues of $X^\top X + \lambda I$ while the blue ones refer to the eigenvalues of $X^\top X$.

Notice that there is a lower bound equal to the value of λ. The eigenvalues of $X^\top X + \lambda I$, which we will define as $\tilde{\lambda}'_i$, can be modeled in the following manner:[3]

$$\tilde{\lambda}'_i = \tilde{\lambda}_i + \lambda$$

As a reminder, $\tilde{\lambda}$ refers to an eigenvalue while λ refers to the ridge regression hyperparameter.

[3]Please refer to the appendix for a more rigorous proof of the behavior of the eigenvalues. It requires using the Singular Value Decomposition, which will come up later in EECS 16B.

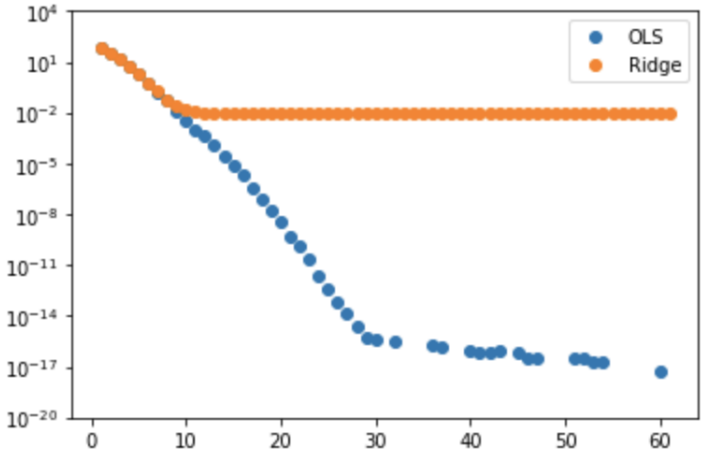


Figure 5: Eigenvalues of a 60-feature matrix with $\lambda = 0.1$

Let's verify that this expression matches what we see on the graph. Recall that $X^\top X$ is a positive semi-definite matrix, so its eigenvalues are non-negative (i.e. $\tilde{\lambda}_i \geq 0, \forall i$), which means that in general:

$$\tilde{\lambda}'_i \geq \lambda$$

So we see that the lower bound of λ is maintained, exactly like the graph.

Now let's verify the behavior of individual eigenvalues. If the ith eigenvalue of $X^\top X$ has a significantly larger value (i.e. $\tilde{\lambda}_i >> \lambda$), then the eigenvalue will dominate the expression, and

$$\tilde{\lambda}'_i \approx \tilde{\lambda}_i$$

On the graph we see that the larger eigenvalues that fit into this category retain their value.

If the eigenvalue is much smaller than λ, such as when it is close to 0, then λ dominates the expression (i.e. $\tilde{\lambda}_i << \lambda$), so

$$\tilde{\lambda}'_i \approx \lambda$$

On the graph we see that the smaller eigenvalues that dip to a value smaller than λ do not retain their value and are instead "replaced" by λ.

If the $\tilde{\lambda}_i$ is near the value of λ, then $\tilde{\lambda}'_i$ is a mixture of the two values, and hence we see that smooth elbow between the regions of large and small eigenvalues.

Now that we know the eigenvalues of $X^\top X + \lambda I$, we can now inspect its diagonalization:

$$X^\top X + \lambda I = V \Lambda' V^{-1}$$

$$\Lambda' = \begin{bmatrix} \tilde{\lambda}_1 + \lambda & 0 & \cdots & 0 \\ 0 & \tilde{\lambda}_2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\lambda}_n + \lambda \end{bmatrix}$$

Taking the inverse of $X^\top X + \lambda I$ we get our calculation of the weight matrix:

$$\begin{aligned} \hat{w} &= (X^\top X + \lambda I)^{-1} X^\top y \\ &= (V(\Lambda')^{-1} V^{-1}) X^\top y \end{aligned}$$

$$(\Lambda')^{-1} = \begin{bmatrix} \frac{1}{\tilde{\lambda}_1+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{\tilde{\lambda}_2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{\tilde{\lambda}_n+\lambda} \end{bmatrix}$$

Notice if $\tilde{\lambda}_i$ is very close to 0, then

$$\frac{1}{\tilde{\lambda}_i + \lambda} \approx \frac{1}{\lambda}$$

Essentially, it puts a "ceiling" on how how much noise in y can corrupt our weight vector w. Thus, we see that

Ridge Regression places a lower bound the eigenvalues of $X^\top X$, thus stabilizing the its inverse and making the calculation of $\hat{w}$ more robust against small perturbations in y caused by noise.

6 Ridge Regression as a High-Pass Filter

In EECS 16B, you have learned about RLC circuits and the phasor space, which was used to help analyze circuits that receive sinusoidal voltages as an input.

We will walk through a simple example of a specific type of RLC circuit called a high-pass filter and see how the behavior of this circuit relates to the eigenvalues in Ridge Regression.

6.1 The Transfer Function

Let's analyze the following circuit below. We will convert it to the phasor space and take a look at its frequency response:

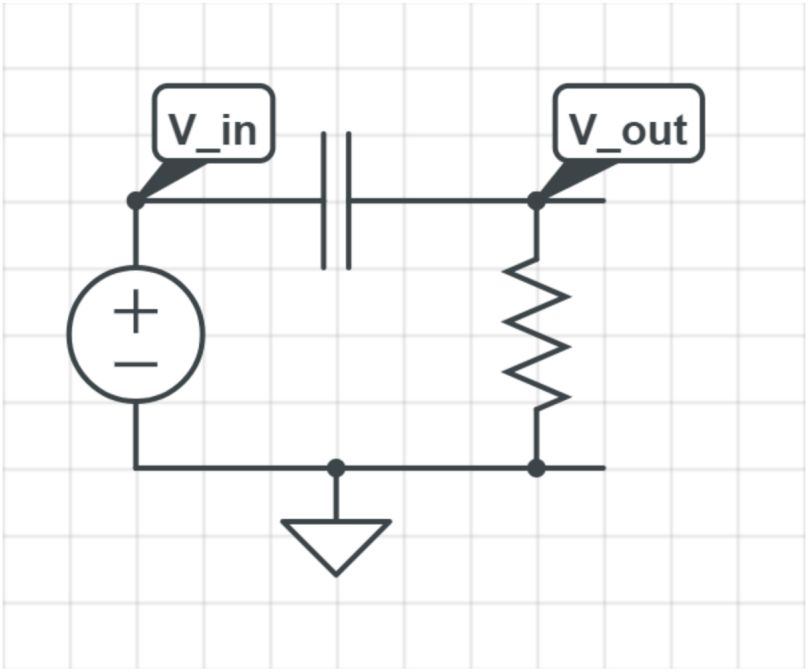


Figure 6: High Pass Filter Circuit

Suppose the input and output voltages are sinusoidal with frequency ω, i.e.

$$v_{in}(t) = \text{Re}\left[V_{in}e^{j\omega t}\right]$$
$$v_{out}(t) = \text{Re}\left[V_{out}e^{j\omega t}\right]$$

the capacitor has a capacitance C, and the resistor has resistance R.

The phasor counterparts of v_{in} and v_{out} would be V_{in} and V_{out}, respectively. The impedance of the capacitor is $Z_C = \frac{1}{j\omega C}$ and the impedance of the resistor is $Z_R = R$. Now that each component in the circuit has been converted to the phasor space, we can now analyze the circuit's frequency response. We often use the *transfer function*:

$$H(\omega) = \frac{V_{out}}{V_{in}}$$

which describes the ratio between output voltage and input voltage (in the phasor space) based on the frequency of the voltage (remember that it is sinusoidal!). In other words, it tells us how the circuit *responds* given a specific *frequency* ω.

Let's derive the transfer function for this particular circuit using nodal analysis in the phasor space. Since impedances describe a linear relationship between current and voltage in the phasor domain, we can simply treat them as resistors. We then see that this particular circuit is a voltage divider, where

$$V_{out} = \frac{Z_R}{Z_C + Z_R}V_{in}$$

From this we see that $H(\omega)$ in terms of Z_R and Z_C is

$$H(\omega) = \frac{V_{out}}{V_{in}} = \frac{Z_R}{Z_C + Z_R}$$

Plugging in the explicit expressions for Z_R and Z_C and simplifying a few terms, we reach an expression for the transfer function that we can analyze:

$$H(\omega) = \frac{1}{1 + \frac{1}{j\omega RC}}$$

6.2 High-Pass Filters

Let's develop an intuition for how the transfer function behaves in response to ω. In particular we are interested in the *magnitude* of the transfer function, which in turn will tell us the ratio between the magnitudes of the output and input voltage. In simpler terms, we will analyze how the *amplitude* of the output voltage changes in response to different frequencies.

We see that ω only influences the second term in the denominator, $1/j\omega RC$. Since we see a j attached to this expression, we know that ω influences a purely imaginary component, and it will never cancel out any real component in $H(w)$.

Suppose we want to maximize $H(\omega)$. In order to do that, we need the imaginary component to be 0. This will occur when ω approaches infinity.

$$H(\infty) = \frac{1}{1 + j\frac{1}{\infty}} = \frac{1}{1 + 0j} = 1 \implies |H(\infty)| = 1$$

In this scenario, a ratio of 1 means that the magnitude of the output signal is the same as the input signal, or in other words the input signal is completely preserved in the output.

Now suppose we wanted to minimize $H(\omega)$. In this case, we want the imaginary component to be as large as possible. This will occur when $\omega = 0$.

$$H(0) = \frac{1}{1 + j\frac{1}{0}} = \frac{1}{1 + \infty j}$$

$$|H(0)| = \left|\frac{1}{1 + \infty j}\right| = \frac{1}{|1 + \infty j|} = \frac{1}{\sqrt{1^2 + \infty^2}} \approx \frac{1}{\infty} = 0$$

In this scenario, a ratio of 0 means that the magnitude of the output signal is 0, or in other words, none of the input signal was preserved in the output.

We see that at high frequencies, the input signal is preserved, while at low frequencies, little to no signal is preserved. Thus we can say this circuit filters out voltages such that it lets *high* frequencies *pass* through.

6.3 Bode Plots

Let's develop a more explicit understanding of the transfer function. This time, we will plot the magnitude of $H(\omega)$ in response to ω. This graph is oftentimes called a *Bode Plot*.

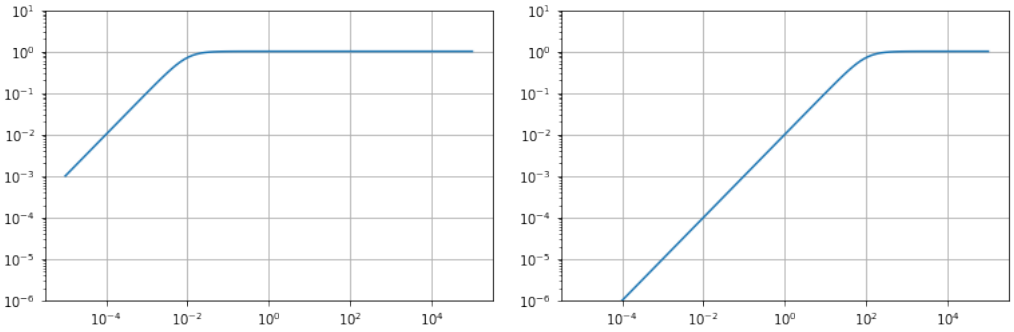


Figure 7: Bode Plot with $\frac{1}{RC} = 0.01$ (left) and $\frac{1}{RC} = 100$ (right)

Pay attention to the scale of the x and y axis, they are both in log-scale. This means that what looks like a linear decrease on the left side of each graph is instead a fast decay towards 0.

Most importantly, notice how the graph changes as we vary the value $\frac{1}{RC}$. It turns out, this value serves as a threshold, where if $\omega > \frac{1}{RC}$, the voltage is preserved, and if $\omega < \frac{1}{RC}$, the voltage is cut off.

Note that this threshold behavior based on $\frac{1}{RC}$ is particular to this circuit and may not apply to other circuits.

6.4 Connection to Ridge Regression

Recall that $H(\omega) = \frac{V_{out}}{V_{in}}$. Using the definition of the transfer function, we can interpret $H(\omega)$ as a function that tells us how much of V_{in} to keep in V_{out} based on ω, Ridge Regression tells us how much of each eigenvalue to keep based on its value.

In our case, the eigenvalues would be represented by ω, and λ would be represented by $\frac{1}{RC}$. If the eigenvalue is larger than λ, then we see that its value is maintained. However, if it is less than λ, then we see that the eigenvalue gets "silenced" and is instead replaced by λ.

7 Alternate Solution to Ridge Regression

Recall that the closed-form solution to Ridge Regression is $\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y$. Notice how this parallels the OLS closed-form solution of $\hat{w} = (X^\top X)^{-1} X^\top y$. Recall that in both solutions, we need to compute $X^\top X$ and then invert it. Recall that X is the $n \times d$ matrix:

$$X = \begin{bmatrix} -\vec{x_1}^\top - \\ -\vec{x_2}^\top - \\ \vdots \\ -\vec{x_n}^\top - \end{bmatrix}$$

Because we are taking the inverse of $(X^\top X)$, this closed-form solution for OLS only works if our X matrix is tall (i.e., $n >> d$). What happens if we want to perform OLS on an X matrix that is wide(more features than data points)? We'll have an infinite number of valid solutions for w.

This is where the **minimum-norm solution** comes into play. When our X matrix is wide, we can utilize the minimum-norm solution to find our $\hat{w}$:

$$\hat{w} = X^\top (X X^\top)^{-1} y$$

Similar to how the closed-form solution to ridge regression paralleled the closed-form OLS solution, there is an alternative closed-form solution to ridge regression which parallels the minimum norm solution:

$$\hat{w} = X^\top (X X^\top + \lambda I)^{-1} y$$

In the following subsections, we will show why this alternative solution is so significant, and why, in some cases, it is preferable to the standard solution.

7.1 Fake Data and Fake Features Perspectives

Finally, we are going to introduce one last set of perspectives on Ridge Regression to help solidify your understanding. Specifically, we will see how adding fake data points or fake features to a standard OLS problem can net us the same closed form solution as ridge regression.

First, let's take a look at the fake data point perspective. Given the X matrix and $\vec{y}$ vector, let us modify the problem with fake data points such that:

$$\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix}$$

$$\hat{y} = \begin{bmatrix} \vec{y} \\ 0 \end{bmatrix}$$

Now, if we apply the standard OLS closed-form solution to the $\hat{X}$ matrix and $\hat{y}$ vector:

$$\hat{w} = (\hat{X}^\top \hat{X}) \hat{X}^\top \hat{y}$$

$$\hat{w} = (\begin{bmatrix} X^\top \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix})^{-1} \begin{bmatrix} X \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \vec{y} \\ 0 \end{bmatrix}$$

$$\hat{w} = (X^\top X + \lambda I)^{-1} y$$

We get the closed-form solution for ridge regression. Notice here that since we only added fake data points, the dimension of the weight vector is still d so we can just use the learned $\hat{w}$ to make predictions: $y_{pred} = X_{test}\hat{w}$

Next, let's observe how we can arrive at the alternative closed-form solution for ridge regression by adding fake features to our data. This time, we will only need to modify our X matrix:

$$\hat{X} = \begin{bmatrix} X\sqrt{\lambda}I \end{bmatrix}$$

Notice that our $\hat{X}$ matrix is now a wide matrix with dimension $n \times (n+d)$, which means we will need to use the minimum-norm solution to find our weight vector. In addition, we will break the weight vector into two components of $\begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix}$. Where $\hat{w}$ is our d-dimensional weight vector for the original features and $\hat{\epsilon}$ is the corresponding n-dimensional weight vector for the fake features. Applying the minimum norm solution to this augmented problem, we can show that the learned $\hat{w}$ is the alternative closed-form solution for ridge regression:

$$\begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix} = \hat{X}^\top (\hat{X}\hat{X}^\top)^{-1}\vec{y}$$

$$\begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix} = \begin{bmatrix} X^\top \\ \sqrt{\lambda}I \end{bmatrix} (\begin{bmatrix} X\sqrt{\lambda}I \end{bmatrix} \begin{bmatrix} X^\top \\ \sqrt{\lambda}I \end{bmatrix})^{-1}\vec{y}$$

$$\begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix} = \begin{bmatrix} X^\top \\ \sqrt{\lambda}I \end{bmatrix} (XX^\top + \lambda I)^{-1}\vec{y}$$

$$\hat{w} = X^\top (XX^\top + \lambda I)^{-1}\vec{y}$$

Thus, we've arrived at the alternative closed-form solution for ridge regression. Notice that in the proof, we've ignored the ϵ term of the weight vector and only kept the $\hat{w}$ that corresponds with the original d features. We did this so the learned weight vector would be the same dimension as the original d features and we can perform a simple matrix multiply to get our predictions like before: $y_{pred} = X_{test}\hat{w}$. If we had chosen to keep the entire $(d+n)$-dimensional weight vector $\begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix}$, we would need to augment the X_{test} matrix again with fake features for testing. This time, instead of adding a $\sqrt{\lambda}I$ like we did in training, we add a $0_{n\times n}$ matrix for testing such that:

$$\hat{X}_{test} = \begin{bmatrix} X_{test} & 0_{n\times n} \end{bmatrix}$$

Since the features associated with the $\hat{\epsilon}$ part of the weight vector are all zero, this case will simplify to the previous example: $y_{pred} = \hat{X}_{test} \begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix} = \begin{bmatrix} X_{test} & 0_{n\times n} \end{bmatrix} \begin{bmatrix} \hat{w} \\ \hat{\epsilon} \end{bmatrix} = X_{test}\hat{w}$

For the most part, this section is purely supplementary and designed to give you another useful perspective on Ridge Regression. In addition, we've also verified that the alternative solution for Ridge Regression can be useful for the Fake Feature perspective.

7.2 Kernels

Now you might notice $XX^\top$ is an $n \times n$ matrix. Since we typically use a lot of data, n can be quite large, and thus this matrix might be quite intensive to compute. However, there is away around this. Notice:

$$XX^\top = \begin{bmatrix} -\vec{x_1}^\top - \\ -\vec{x_2}^\top - \\ \vdots \\ -\vec{x_n}^\top - \end{bmatrix} \begin{bmatrix} | & | & \dots & | \\ \vec{x_1} & \vec{x_2} & \dots & \vec{x_n} \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} \langle \vec{x_1}, \vec{x_1} \rangle & \langle \vec{x_1}, \vec{x_2} \rangle & \dots & \langle \vec{x_1}, \vec{x_n} \rangle \\ \langle \vec{x_2}, \vec{x_1} \rangle & \langle \vec{x_2}, \vec{x_2} \rangle & \dots & \langle \vec{x_2}, \vec{x_n} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \vec{x_n}, \vec{x_1} \rangle & \langle \vec{x_n}, \vec{x_2} \rangle & \dots & \langle \vec{x_n}, \vec{x_n} \rangle \end{bmatrix}$$

Each entry of $XX^\top$ is the inner product of two featurized data points! We call this a **kernel function**, which allows us to compute our matrix extremely quickly.

The specific details of how this alternative ridge regression formula improves performance is out of scope for now. However, it will connect with a future lesson on Kernels.

Appendix

Derivation of OLS Solution

Recall that our optimization problem for Ordinary Least Squares is:

$$\hat{w} = \arg\min_w \|y - Xw\|_2^2$$

Now, our loss function:

$$\mathcal{L} = \|y - Xw\|_2^2 = w^\top X^\top X w - 2w^\top X^\top y + y^\top y$$

is convex. Convexity is covered in greater depth in more advanced courses (such as EE127). For now, all you need to know is that for a convex loss function, any local minimum is a global minimum, and thus, we can find the optimal w by setting the derivative to 0.

$$\frac{\partial \mathcal{L}}{\partial w} = 2X^\top X w - 2X^\top y = 0$$

Now we simply solve for w:

$$X^\top X w = X^\top y$$
$$\hat{w} = (X^\top X)^{-1} X^\top y$$

Note that this solution only works if $X^\top X$ is invertible. If the number of data points is far greater than the number of features, this is very likely to be true.

Derivation of Ridge Regression Solution

The solution to ridge regression is quite similar to that of OLS. Recall our optimization problem:

$$\hat{w} = \arg\min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2 = w^\top X^\top X w - 2w^\top X^\top y + y^\top y + \lambda w^\top w$$

Once again, our loss function is convex with respect to w. So we take the gradient and set it to 0:

$$\frac{\partial \mathcal{L}}{\partial w} = 2X^\top X w - 2X^\top y + 2\lambda w = 0$$

Now we simply isolate w:

$$(X^\top X + \lambda I)w = X^\top y$$
$$\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y$$

Notice here that $X^\top X + \lambda I$ will always be invertible, so we don't have the same concerns about invertibility that we had with OLS.

Ridge Regression with SVD

Note: This section requires an understanding of the SVD and symmetric matrices, which will be taught towards the end of EECS 16B

To find the eigenvalues of $X^\top X + \lambda I$ we will be using the full SVD: any $m \times n$ matrix A can be expressed as:

$$A = U\Sigma V^\top$$

where U and V are unitary square matrices, i.e.

$$U^\top U = UU^\top = V^\top V = VV^\top = I$$

and

$$\Sigma = \left[\begin{array}{c|c} S & 0 \\ \hline 0 & 0 \end{array}\right]$$

where S is a diagonal matrix containing the singular values of A. We also will be relying on the fact that each singular value $\sigma_i = \sqrt{\tilde{\lambda}_i}$, where $\tilde{\lambda}_i$ is the ith eigenvalue of $A^\top A$ and $AA^\top$

With the SVD defined we now can examine $X^\top X + \lambda I$ by taking the SVD of X:

$$\begin{aligned} X^\top X + \lambda I &= (U\Sigma V^\top)^\top (U\Sigma V^\top) + \lambda V V^\top \\ &= V\Sigma^\top U^\top U \Sigma V^\top + \lambda V I V^\top \\ &= V\Sigma^\top \Sigma V^\top + V(\lambda I)V^\top \\ &= V(\Sigma^\top \Sigma + \lambda I)V^\top \end{aligned}$$

We now will use an important property of symmetric matrices, which is that its eigenvectors can be orthonormal. Thus, for any symmetric matrix, its eigendecomposition can be expressed as:

$$A = V\Lambda V^{-1} = V\Lambda V^\top$$

We know that $X^\top X + \lambda I$ is symmetric because both $X^\top X$ and λI are symmetric, so we know that the expression we stopped at is an eigendecomposition of it, so the values within $\Sigma^\top \Sigma + \lambda I$ must be the eigenvalues. Further expanding $\Sigma^\top \Sigma$:

$$\Sigma^\top \Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix} = \begin{bmatrix} \tilde{\lambda}_1 & 0 & \cdots & 0 \\ 0 & \tilde{\lambda}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\lambda}_n \end{bmatrix}$$

So this must mean that:

$$\Sigma^\top \Sigma + \lambda I = \begin{bmatrix} \tilde{\lambda}_1 + \lambda & 0 & \cdots & 0 \\ 0 & \tilde{\lambda}_2 + \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\lambda}_n + \lambda \end{bmatrix}$$

Thus, the eigenvalues of $X^\top X + \lambda I$ are $\tilde{\lambda}_i + \lambda$.