Chapter 4
Feature Selection and LASSO

4.1 Ridge Regression Recap

For ridge regression we use a standard MSE loss with an L2 norm regularizer.

$$\hat{w} = \underset{w}{\operatorname{argmin}} \operatorname{MSE}(W) + \lambda ||w||_2^2 \tag{4.12}$$

The hyperparameter λ can play a large role in how a model behaves. For instance, if $\lambda = 0$ we would then have a standard regression model with no regularization. On the opposite side of the spectrum, if $\lambda = \infty$ then any feature usage at all (any weights greater than 0) would penalize the model completely, leaving us with a model that can only output 0. Clearly, if the goal is to create a useful model that uses regularization neither of those two options are ideal. Instead, we will use something in between.

When choosing λ if it is too small then the model will overfit to the training data. As in previous chapters, we must evaluate our hyperparameter on the validations set. We should typically pick the model that best performs on the validation set (lowest error) using MSE only. Why MSE only when we have a regularized objective function already? The reason is that different λ, or any hyperparemeter for that matter, mean we are evaluating the model's performance in different contexts or "units". We could compare the results of any number of models using the same objective function, however when the hyperparameters are different we cannot compare them.

For reference, here is an example process for choosing λ for ridge regression.

Algorithm 1 Process for selecting λ for ridge regression

for λ in λs **do** Train a model using using Gradient Descent
– $\hat{w}_{\text{ridge}(\lambda)} = \underset{w}{\operatorname{argmin}} \operatorname{MSE}(W) + \lambda ||w||_2^2$
Compute validation error
– $validation\_error = \operatorname{MSE}_{val}(\hat{w}_{\text{ridge}(\lambda)})$
Track λ with smallest $validation\_error$ **return** λ^* and estimated future error $\operatorname{MSE}_{val}(\hat{w}_{\text{ridge}(\lambda^*)})$

We have seen how regularization can be useful with ridge regression to avoid overfitting by adding a level of smoothness to our model. However there are many different ways to regularize and one way we will investigate is called lasso, useful for selecting features.

4.2 Scaling Features Recap

Features in dataset can have different units, and if we change our features to different units, for example, change meter to kilometer, the corresponding parameter for this feature is also changed. Feature scaling is important before applying regularization, because regularization penalizes large parameters. Therefore, if the parameter of a feature gets larger because of unit change, its parameter would be penalized more by regularization. This is not ideal, since we want regularization to work the same regardless of the units. The solution to solve this problem is using feature normalization.

Definition 4.1: Feature Normalization

$$\tilde{h_j}(x) = \frac{h_j(x) - \mu_j}{\sigma_j}$$

where μ_j is mean of feature j in train set and σ_j is standard deviation of feature j in train set.

Scaling features is always a good practice for data preprocessing. The only downside is that the features you have after scaling are no longer as interpretable as before.

To normalize data, we calcuate mean and standard deviation only using train set.

4.3 Feature Selection

Feature selection sounds fun but why should we care? There are three major reasons why feature selection is useful in machine learning.

1. **Complexity** - With too many features our model could be overly complex and overfit.
2. **Interpretability** - With less features, or selected feature, we can better understand our model and see which features carry more information
3. **Efficiency** - With less features to consider, our model could be trained much easier and faster

Let's introduce a scenario where we want to create a good model for predicting the price of a house. We have the opportunity to collect a wide variety of features, however we are unsure of what features are best to be used.

Example(s)

3

- Lot size
- Single Family
- Year built
- Last sold price
- Last sale price/sqft
- Finished sqft
- Unfinished sqft
- Finished basement sqft
- Number of floors
- Flooring types
- Parking type
- Parking amount
- Cooling
- Heating
- Exterior materials
- Roof type
- Structure style
- Dishwasher
- Garbage disposal
- Microwave
- Range / Oven
- Refrigerator

- Washer
- Dryer
- Laundry location
- Heating type
- Jetted Tub
- Deck
- Fenced Yard
- Lawn
- Garden
- Sprinkler System

You or I could pick out that the features "lot size" and "last sold price" (among others) are likely good indicators of a house's sale price and that the features "microwave" and "garbage disposal" could have an effect, but are not likely to be key factors for predicting a house sale price.This is a key example of feature selection.

By selecting a smaller number of features, this model will be more interpretable as it is easier to determine that "lot size" is one of the most important features for predicting house price. With less features, our model can be trained more easily. In addition, if the data must be gathered ourselves, gathering less data points is simply easier to do.

Although it could be helpful to use all the information available to make a model that can predict well, a large amount of features can be prohibitively expensive to train. Also, if features are used which are unimportant it could lead to the model overfitting the data.

Example(s)

Let us imagine it takes us 5 seconds to train a model with 8 features:

- Training 16 features would take 21 minutes.
- Training 32 would take about 3 years.
- Training 100 features would take $7.5 * 10^{20}$ or about $50,000,000,000$ times longer than the age of the universe.

Seems like it might not be worth considering if the "sprinkler system" feature is included with the house in our model. One method to select features would be to try every possible combination of features and see how to model performs manually. However, this can be tedious and without trying every set of hyperparameters we would not be certain that we have chosen the optimal features. To aid in feature selection we can use another linear regression model with a new type of regularizer called lasso.

4.4 LASSO

We can create a regression using all of these features and add a regularizer in the hope that we achieve a sparse weight vector which still performs well on the data. A sparse matrix / vector is one that is filled with mostly 0. This would help us in feature selection as the unimportant features would have weight 0, meaning they are not being considered.

Our format for a model with regularization is as such where $L(w)$ is the measure of fit and $R(w)$ measures the magnitude of coefficients.

$$\hat{w} = \underset{w}{\operatorname{argmin}}\, L(w) + R(w) \tag{4.13}$$

There are many different ways to create a regularizer. One that works well is the sum of squares which is what we have seen in ridge regression (L2 norm).

Another possibility is simply the sum of the weights. This however would not be ideal as negative and positive weights could cancel one another out. To fix this we could sum the absolute value of the weights, which is exactly what the lasso model does. This is also known as the L1 norm.

Definition 4.2: Lasso Regression Model

$$\hat{w} = \underset{w}{\operatorname{argmin}} \operatorname{MSE}(W) + \lambda ||w||_1$$

Remember the definition of a p-norm.

Definition 4.3: p-norm

$$||w||_p^p = |w_0|^p + |w_1|^p + ... + |w_d|^p$$

Without lasso, we could have tried different features and computed the resulting loss, but now we start with a full model and then shrink coefficients towards 0. With ridge regression many of our coefficients as the model is trained will descend towards 0, but not reach 0. Why? It has to do with the shape of the norm. Ridge introduces a smoothness to prevent overfit by limiting the amount that one single feature can use.

In general, the coefficients are pushed towards 0, but never actually reach it despite training for thousands of epochs. In lasso, the L1 norm favors sparsity and produces a coefficient path where coefficients go towards and actually remain at 0. Take a look at some coefficient paths for ridge regression below.

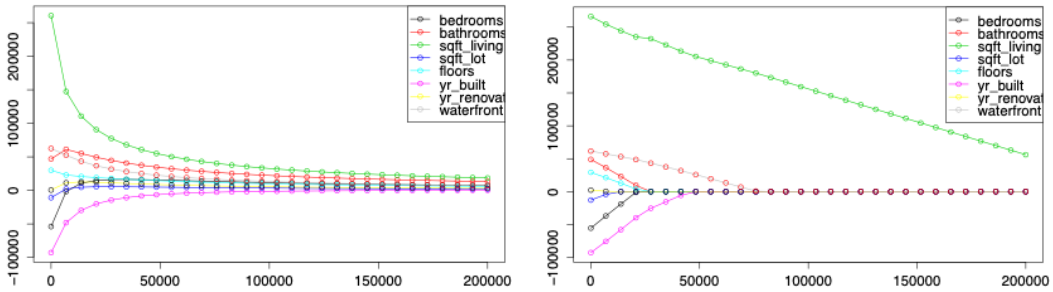

Figure 4.18: Ridge (right) and lasso (left) coefficient paths.

One way to think about this geometrically is that the L1 norm has a "spikey" solution. In a L1 norm, as coefficients are close to 0, the only way to get close to 0 is actually being 0. In a L2 norm, coefficients can be very small and not quite reach 0.

In Figure 4.2, L1 has spikes representing the places where the coefficients are 0. When the unregularized coefficient paths (represented in red) touch the regularized solution in either graph we see that the diamond shape will likely touch at one of its corners where a coefficient is 0 whereas L2 would be much more likely to touch equally and at any location on the circle.

When it comes to choosing a λ for lasso, the choice is done exactly the same as what we have learned for ridge regression.

However, there are caveats of using lasso. As with any regularizer, we are adding bias to our least squares solution (remember bias + variance trade off). One way to remove some bias, but still benefit from having a sparse solution would be to first run lasso, then extract the non-zero features and run those features on

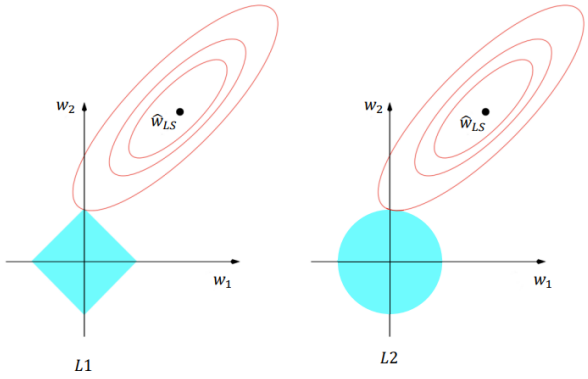

Figure 4.19: Visualizing sparse solutions with L1 on left and L2 on right.

a unregularized least squares so that coefficients are no longer shrunk from their possible "truth" values as the model is trained.

As with all types of machine learning it is important to remember that correlation does not equal causation. Lasso will do exactly what you ask of it and nothing more. Lasso will find the features that are important for predicting some output well. However, it is important to always think before and after whether the solutions are representative of the type of data you are training on.

Example(s)

There is a publicly available Communities and Crime Data Set which has a large number of features which could be trained on to predict where crime will take place based on the demographics of that location like its population density, the age of people living there, etc.

If you were to train a lasso model on this data set you would likely see that areas where there is a high proportion of people aged 65 and up usually have low crime and lasso will likely select this as a good predictor (non-zero coefficient). This is an example of correlation and not causation. We asked our model to find good predictors of crime rate, however clearly it would not be logical to attempt to lower high crime areas by moving lots of senior citizens there. In complex social issues, among other data sets, remembering to think about what your model is doing vs what you would like it to do is important.

In practice, lasso can run into some issues. Lasso tends to pick arbitrarily from correlated features, like number of bathrooms and number of showers from our house price example. In this case it might make more sense to choose bathrooms or to select them together, however lasso does not offer this guarantee. In addition, lasso tends to have worse performance in practice compare to ridge regression because of the bias introduced by pushing feature coefficients towards 0.

A solution to this problem where we want feature selection and the performance from ridge is a model called elastic net which is simply the combination of a L1 norm and a L2 norm regularizer.

Definition 4.4: Elastic Net

$$\hat{w}_{\text{Elastic Net}} = \underset{w}{\operatorname{argmin}}\, \text{MSE}(W) + \lambda_1 ||w||_1 + \lambda_2 ||w||_2^2$$

A combination of both L1 (Lasso) and L2 (Ridge) regularizers.

To review we learned that lasso:

- Introduces more sparsity to the model.
- Is helpful for feature selection.
- Is less sensitive to outliers.
- Is more computationally efficient as a model due to the sparse solutions.

And ridge regression:

- Pushes weights towards 0, but not actually 0.
- Is more sensitive to outliers (due to the squared terms).
- In practice, usually performs betters.

This concludes the chapter's focus on regression and next we will learn about different models for different uses like classification.