1 Using sparsity to regularize models

- The least absolute shrinkage and selection operator (LASSO) method is very similar to ridge regression and LARS.
- It's similar to Ridge Regression in the sense that we penalize our regression by some amount, and it's similar to LARS in that it can be used as a parameter selection, and it typically leads to a sparse vector of coefficients.

1.0.1 Getting ready

To be clear, lasso regression is not a panacea. There can be computation consequences to using lasso regression. As we'll see in this recipe, we'll use a loss function that isn't differential, and therefore, requires special, and more importantly, performance-impairing workarounds.

1.1 Implementation

Let's go back to the trusty `make_regression` function and create a dataset with the same parameters:

```
>>> from sklearn.datasets import make_regression
>>> reg_data, reg_target = make_regression(n_samples=200, n_features=500,
n_informative=5, noise=5)
```

Next, we need to import the Lasso object:

```
>>> from sklearn.linear_model import Lasso
>>> lasso = Lasso()
```

Lasso contains many parameters, but the most interesting parameter is alpha. It scales the penalization term of the Lasso method, which we'll look at later. For now, leave it as 1. As an aside, and much like ridge regression, if this term is 0, lasso is equivalent to linear regression:

```
>>> lasso.fit(reg_data, reg_target)
```

Again, let's see how many of the coefficients remain nonzero:

```
>>> np.sum(lasso.coef_ != 0)
9
>>> lasso_0 = Lasso(0)
>>> lasso_0.fit(reg_data, reg_target)
>>> np.sum(lasso_0.coef_ != 0)
500
```

None of our coefficients turn out to be 0, which is what we expect. Actually, if you run this, you might get a warning from scikit-learn that advises you to choose LinearRegression.

1.2 Lasso cross-validation

Choosing the most appropriate lambda is a critical problem. We can specify the lambda ourselves or use cross-validation to find the best choice given the data at hand:

```
>>> from sklearn.linear_model import LassoCV
>>> lassocv = LassoCV()
>>> lassocv.fit(reg_data, reg_target)
```

- `lassocv` will have, as an attribute, the most appropriate lambda.
- scikit-learn mostly uses alpha in its notation, but the literature uses lambda:

```
>>> lassocv.alpha_
0.80722126078646139
```

The number of coefficients can be accessed in the regular manner:

```
>>> lassocv.coef_[:5]
array([0., 42.41, 0.,0., -0.])
```

Letting lassocv choose the appropriate best fit leaves us with 11 nonzero coefficients:

```
>>> np.sum(lassocv.coef_ != 0)
11
```

1.2.1 Lasso for feature selection

- Lasso can often be used for feature selection for other methods.
- For example, you might run lasso regression to get the appropriate number of features, and then use these features in another algorithm.
- To get the features we want, create a masking array based on the columns that aren't zero, and then filter to keep the features we want:

```
>>> mask = lassocv.coef_ != 0
>>> new_reg_data = reg_data[:, mask]
>>> new_reg_data.shape
(200, 11)
```