

Lasso Regression

Least Absolute Shrinkage and Selection Operator Regression (simply called *Lasso Regression*) is another regularized version of Linear Regression: just like Ridge Regression, it adds a regularization term to the cost function, but it uses the ℓ_1 norm of the weight vector instead of half the square of the ℓ_2 norm (see Equation 4-10).

Equation 4-10. Lasso Regression cost function

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^n |\theta_i|$$

Figure 4-18 shows the same thing as Figure 4-17 but replaces Ridge models with Lasso models and uses smaller α values.

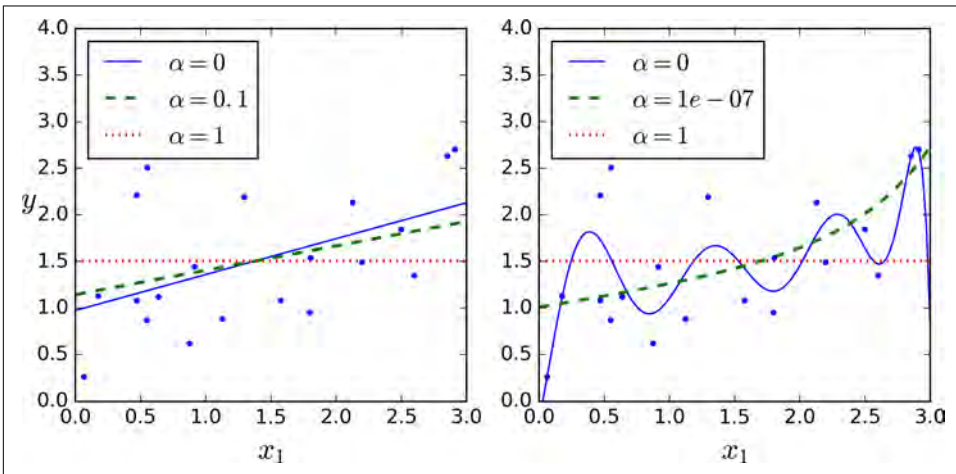


Figure 4-18. Lasso Regression

An important characteristic of Lasso Regression is that it tends to completely eliminate the weights of the least important features (i.e., set them to zero). For example, the dashed line in the right plot on Figure 4-18 (with $\alpha = 10^{-7}$) looks quadratic, almost linear: all the weights for the high-degree polynomial features are equal to zero. In other words, Lasso Regression automatically performs feature selection and outputs a *sparse model* (i.e., with few nonzero feature weights).

You can get a sense of why this is the case by looking at Figure 4-19: on the top-left plot, the background contours (ellipses) represent an unregularized MSE cost function ($\alpha = 0$), and the white circles show the Batch Gradient Descent path with that cost function. The foreground contours (diamonds) represent the ℓ_1 penalty, and the triangles show the BGD path for this penalty only ($\alpha \rightarrow \infty$). Notice how the path first

reaches $\theta_1 = 0$, then rolls down a gutter until it reaches $\theta_2 = 0$. On the top-right plot, the contours represent the same cost function plus an ℓ_1 penalty with $\alpha = 0.5$. The global minimum is on the $\theta_2 = 0$ axis. BGD first reaches $\theta_2 = 0$, then rolls down the gutter until it reaches the global minimum. The two bottom plots show the same thing but uses an ℓ_2 penalty instead. The regularized minimum is closer to $\theta = 0$ than the unregularized minimum, but the weights do not get fully eliminated.

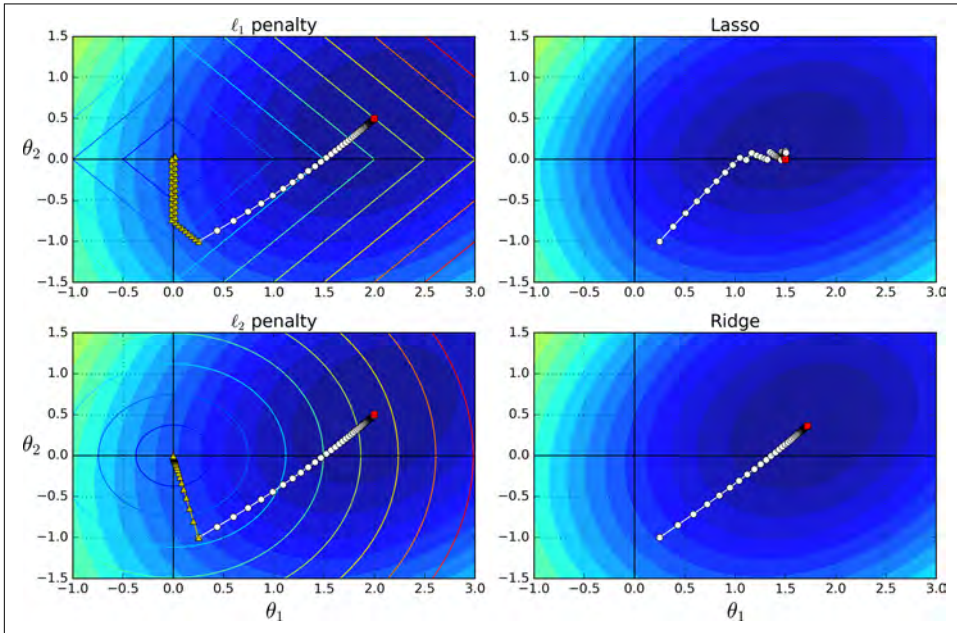


Figure 4-19. Lasso versus Ridge regularization



On the Lasso cost function, the BGD path tends to bounce across the gutter toward the end. This is because the slope changes abruptly at $\theta_2 = 0$. You need to gradually reduce the learning rate in order to actually converge to the global minimum.

The Lasso cost function is not differentiable at $\theta_i = 0$ (for $i = 1, 2, \dots, n$), but Gradient Descent still works fine if you use a *subgradient vector* \mathbf{g}^{15} instead when any $\theta_i = 0$. Equation 4-11 shows a subgradient vector equation you can use for Gradient Descent with the Lasso cost function.

15 You can think of a subgradient vector at a nondifferentiable point as an intermediate vector between the gradient vectors around that point.

$$g(\theta, J) = \nabla_{\theta} \text{MSE}(\theta) + \alpha \begin{pmatrix} \text{sign}(\theta_1) \\ \text{sign}(\theta_2) \\ \vdots \\ \text{sign}(\theta_n) \end{pmatrix} \quad \text{where} \quad \text{sign}(\theta_i) = \begin{cases} -1 & \text{if } \theta_i < 0 \\ 0 & \text{if } \theta_i = 0 \\ +1 & \text{if } \theta_i > 0 \end{cases}$$

Here is a small Scikit-Learn example using the Lasso class. Note that you could instead use an SGDRegressor(penalty="l1").

```
>>> from sklearn.linear_model import Lasso
>>> lasso_reg = Lasso(alpha=0.1)
>>> lasso_reg.fit(X, y)
>>> lasso_reg.predict([[1.5]])
array([ 1.53788174])
```

Elastic Net

Elastic Net is a middle ground between Ridge Regression and Lasso Regression. The regularization term is a simple mix of both Ridge and Lasso's regularization terms, and you can control the mix ratio r . When $r = 0$, Elastic Net is equivalent to Ridge Regression, and when $r = 1$, it is equivalent to Lasso Regression (see Equation 4-12).

Equation 4-12. Elastic Net cost function

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n \theta_i^2$$

So when should you use Linear Regression, Ridge, Lasso, or Elastic Net? It is almost always preferable to have at least a little bit of regularization, so generally you should avoid plain Linear Regression. Ridge is a good default, but if you suspect that only a few features are actually useful, you should prefer Lasso or Elastic Net since they tend to reduce the useless features' weights down to zero as we have discussed. In general, Elastic Net is preferred over Lasso since Lasso may behave erratically when the number of features is greater than the number of training instances or when several features are strongly correlated.

Here is a short example using Scikit-Learn's ElasticNet (l1_ratio corresponds to the mix ratio r):

```
>>> from sklearn.linear_model import ElasticNet
>>> elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
>>> elastic_net.fit(X, y)
>>> elastic_net.predict([[1.5]])
array([ 1.54333232])
```