Regression Regularization

[NOTE: If viewing on github, some browser+OS combinations render iPython LaTeX incorrectly. Safari OSX seems to be the one exception. As a result, the math in this section may disappear or look funny. This is an open problem.]

Regularized regression aims to minimize overfitting by adding a regularization term to linear or logistic regression models. This regularization term is added to, in this instance, a standard least squares linear regression cost function J:

$$J(\Theta) = 1/2m[\sum_{i=1}^{m}(h_\Theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^{n} \Theta_j^2]$$

Θ = parameter values

m = training examples with n features

$h_\Theta(x^i)$ = the estimator h_Θ value for training example i

y^i = the actual labeled value of training example i

λ = regularization constant

The important thing to notice is the addition of $\lambda \sum_{j=1}^{n} \Theta_j^2$. What is the effect of this term? When minimizing the cost as a function of our parameter values Θ , smaller values of theta must be used than we would otherwise obtain under unregularized regression, and therefore we can diminish overfitting because the contribution of each feature is dampened. Under gradient descent, the cost function can be rearranged to isolate the regularization parameter such that when we update Θ_j :

$$\Theta_j := \Theta_j(1 - \alpha\lambda/m) - \alpha/m \sum_{i=1}^{m}(h_\Theta(x^i) - y^i)x_j^i$$

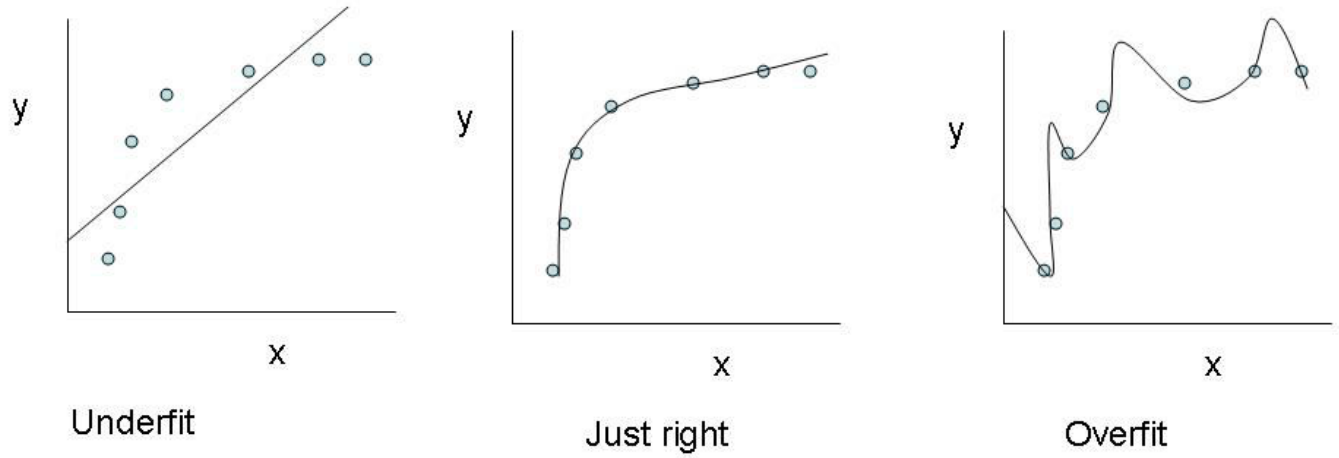
α = constant learning rate

(Note that we ignore the intercept value Θ_0 , which doesn't receive a regularization term)

This rearranged formulation gives us the first term $\Theta_j(1 - \alpha\lambda/m)$, from which we can more readily interpret the effect that regularization parameter λ has on the updated value of Θ_j and the contribution of feature set m to the final prediction curve.

As if often the case, the best way to get an intution for how this performs on a given dataset is to plug in a couple of values for the regularization parameter λ and maybe graph the effect for a range of values. For example, the image below shows a prediction curve that, without regularization, exhibits high variance and probably overfits the training data. If we set $\lambda = 0$, we just have an unregularized gradient descent formula and an overfit estimator, seen on the left. Alternatively, if we set $\lambda = 10000$ (or some "very large" value relative to your data) then we have added a large regularization term to our cost function, and greatly diminished the initial value of Θ_j from which the prediction error is subtracted. The only way to minimize J at this point is for the gradient descent method to "select" very small values of Θ , thus minimizing the contribution that each feature

has upon our prediction curve to such a degree that the estimator is all but unaffected by the data features, seen on the left. By tuning the regularization parameter, we can hope to achieve an estimator that does a better job of balancing variance and bias and hopefully generalizes well to new data, seen in the center.



Thus, adding regularization to our regression estimator can in many instances give us a rough bias/variance dial with which we can address the overfitting of our estimator onto the training set.

Lasso Regression (L1 Regularization) and Ridge Regression (L2 Regularization)

In discussing regularization we employed L2 regularization, also known as Tikhonov regularization and most well known as instantiated in Ridge Regression. This is one particular method of regularization. L1 regularization, most well-known in Lasso regression, is another such strategy for controlling overfitting. The two techniques share the same goal but differ in a few key respects. I find that it's somewhat easier to explore the specific properties of these different approaches when we put them side by side and can contrast their differences. Since we have covered in broad strokes what regularization is and why we use it, this section will focus on differences between L1 and L2 regularization.

Recall the regularized cost function above:

$$J(\Theta) = 1/2m[\sum_{i=1}^{m}(h_\Theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^{n} \Theta_j^2]$$

The regularization term used in the discussion above can now be introduced as, more specifically, the L2 regularization term:

$$\lambda \sum_{j=1}^{n} \Theta_j^2$$

In contrast to the L1 regularization term:

$$\lambda \sum_{j=1}^{n} |\Theta_j|$$

Clearly, the only difference between L1 and L2 is just that L2 uses the sum of the square of the parameters, while L1 is the sum of the absolute value of the parameters. However slight the difference might seem, there are some important consequences that should be considered before deciding which regression strategy to use.

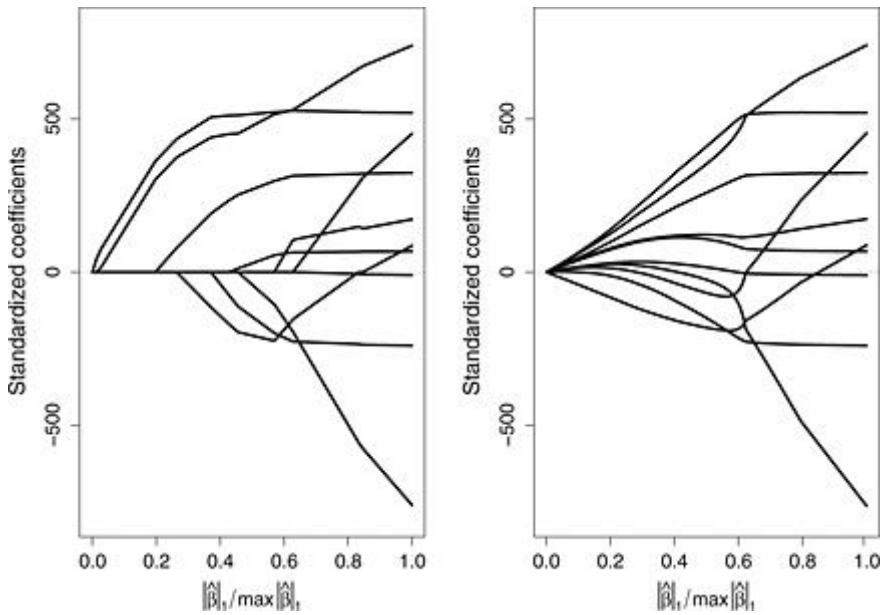
It's worth mentioning that although L1 and L2 regularization are probably the most discussed and well-known form of regularization, they are not the only forms. In fact, both L1 and L2 fit into the general class of loss function vector norms:

$$(\sum_{j=1}^{n} |\Theta_j|^p)$$

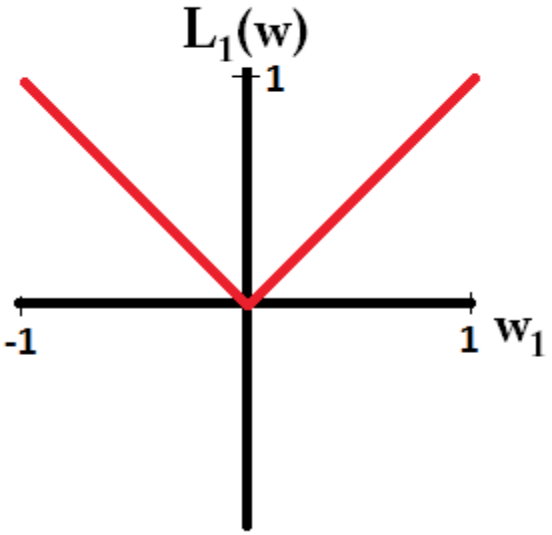
where p=1 for L1 regularization and p=2 for L2 regularization.

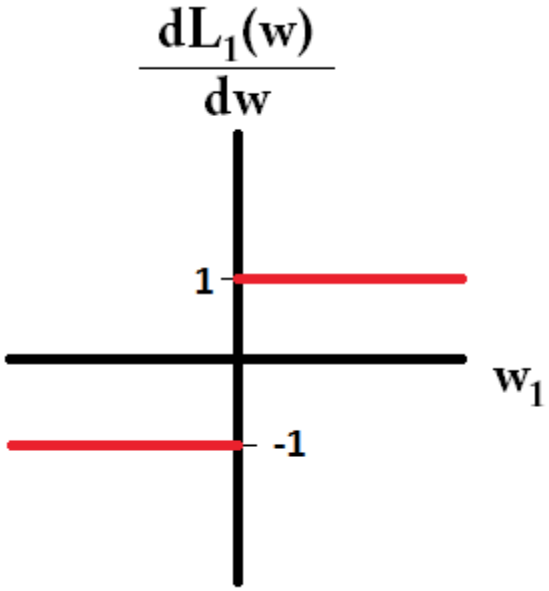
What happens to the parameters as regularization changes?

In many cases, L1 regularization will, with a high enough λ, reduce some parameters associated with a given feature to zero. In essence, L1 performs feature selection, and in practice, L1 is used on sparse datasets or on datasets where we would like to partially reduce the number of features. L2 regularization, on the other hand, will not set feature parameters to zero, but will only continue to reduce the value of a given Θ. The following image demonstrates the contrast in parameter values for increasingly strict regularization. In both cases the parameter values approach zero as the regularization parameter is increased. Under L1 the coefficients hit zero, under L2 the coefficients diminish, but never hit zero:

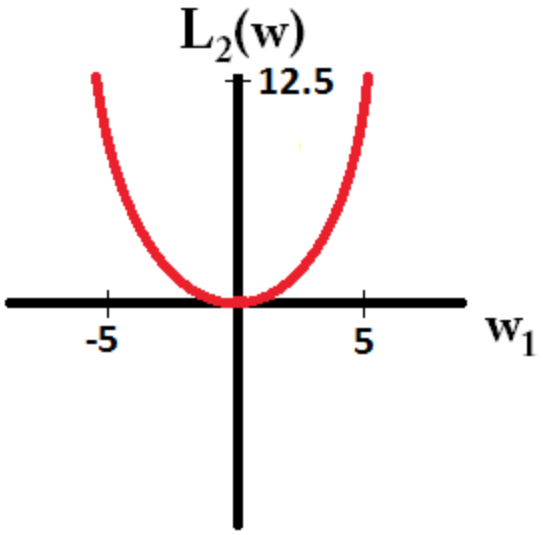


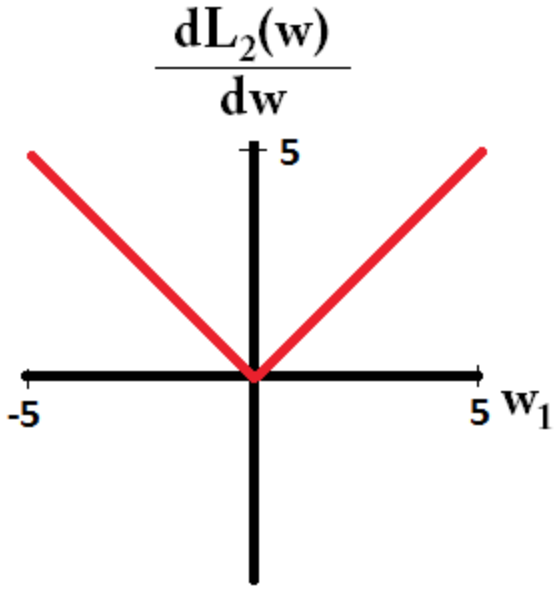
An explanation comes from examining the shape of the L1 and L2 regularization curves. For L1 regression, the absolute value loss function L is a v shape, and its derivative with respect to the parameter (weight) w is discontinuous 1 for positive values and -1 for negative values.





For L2 regression, the absolute value loss function L is curved, and its derivative with respect to the paramter (weight) w is a v shape with slope 1 for positive values and slope -1 for negative values.





Recalling gradient descent minimization of the loss function, our parameters are updated according to this loss function. For L1 regularization, the stepwise reduction in Θ is each time a constant value, whereas for L2 regularization the stepwise reduction in value Θ is a constant value multiplied by the value of Θ, thus the magnitude of each reduction in L2 regularization gets smaller and smaller for each step, approaching zero but never reaching it (for finite λ).

Consider also that our regularized loss function:

$$J(\Theta) = 1/2m[\sum_{i=1}^{m}(h_\Theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^{n} \Theta_j^p]$$

...can be rearranged once more in a manner that simply recastes the function as an unregularized loss function subject to the constraint:

$$\sum_{j=1}^{n} \Theta_j^p \leq t$$

...where t corresponds directly to λ. By recasting this as a constraint region for our loss function, we can now visualize the difference between L1 and L2 regularization (readily extendable into any regularization number p in $(\sum_{j=1}^{n} |\Theta_j|^p)$:

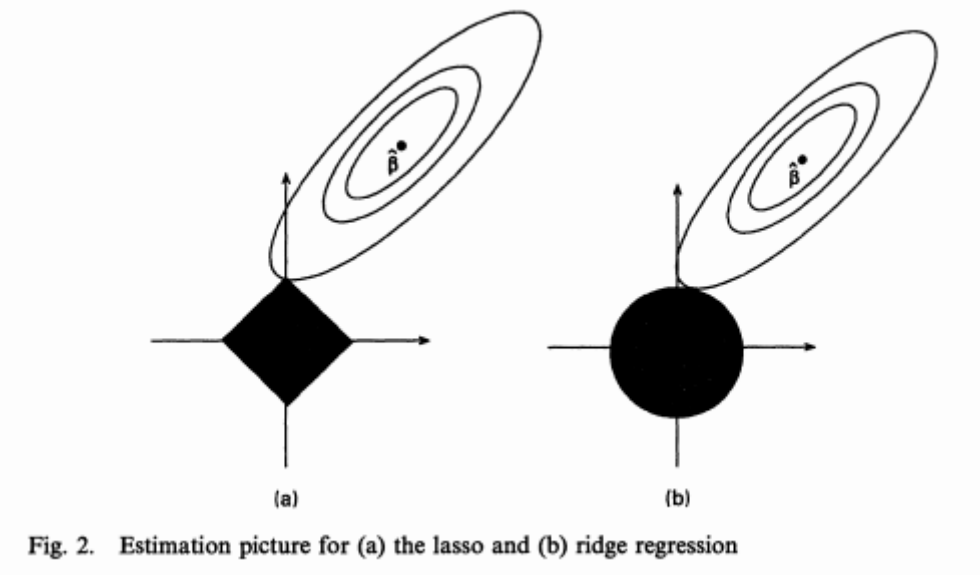


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

Here the x- and y-axes represent the case of regularized regression with only two parameters β_1 and β_2 , where $\hat{\beta}$ centered within the contour lines minimizes our unregularized least squares regression. The shaded region then represents the constraint of regularization t given above. Regularized gradient descent seeks to minimize the loss function subject to the regularization constraint region (which expands or shrinks depending on the value of t or, alternatively, λ), and therefore will settle where the two surfaces meet.

Elastic Net

The Elastic Net penalty is another interesting option for regularization. It combines both the L1 and L2 term in an attempt to combine their strengths:

$$\lambda \sum_{j=1}^{p} (\alpha\Theta_j^2 + (1-\alpha)|\beta_j|)$$

Specifically, the creators of the Elastic Net wanted to preserve the feature selection property of L1 regression while preserving grouping: if multiple variables are collinear, L1 tends to arbitrarily select one variable from that group while eliminating the others. In addition, L1 can lead to situations in which there is more than one prediction curve that appropriately minimizes the cost function. If L1 arbitrarily chooses one of those two, it is possible that a slight change in the value of one datapoint could cause the regression curve to "jump" to an entirely different curve because the new curve has suddenly had the minized cost fuction tipped in its favor.

Introducing L2 regularization helps to address the problems with L1 while preserving its feature selection property.

Elements of Machine Learning, by HTF, Chapter 3.4 is a great resource (https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization)

Andrew Ng's machine learning lectures on regularization (https://www.coursera.org/learn/machine-learning/home/week/3)

Presentation by Zou and Hastie, creators of Elastic Net (http://web.stanford.edu/~hastie/TALKS/enet_talk.pdf)

Insightful discussion and links about L1 vs. L2 here... (http://stats.stackexchange.com/questions/45643/why-l1-norm-for-sparse-models)

...and here (https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization)