Regularization for Simplicity: Lambda

ıted Time: 8 minutes

Model developers tune the overall impact of the regularization term by multiplying its value by a scalar known as **lambda** (also called the **regularization rate**). That is, model developers aim to do the following:

$$\mathbf{minimize}(\mathbf{Loss}(\mathbf{Data}|\mathbf{Model}) + \lambda\, \mathbf{complexity}(\mathbf{Model}))$$

Performing L_2 regularization has the following effect on a model

- Encourages weight values toward 0 (but not exactly 0)
- Encourages the mean of the weights toward 0, with a normal (bell-shaped or Gaussian) distribution.

Increasing the lambda value strengthens the regularization effect. For example, the histogram of weights for a high value of lambda might look as shown in Figure 2.

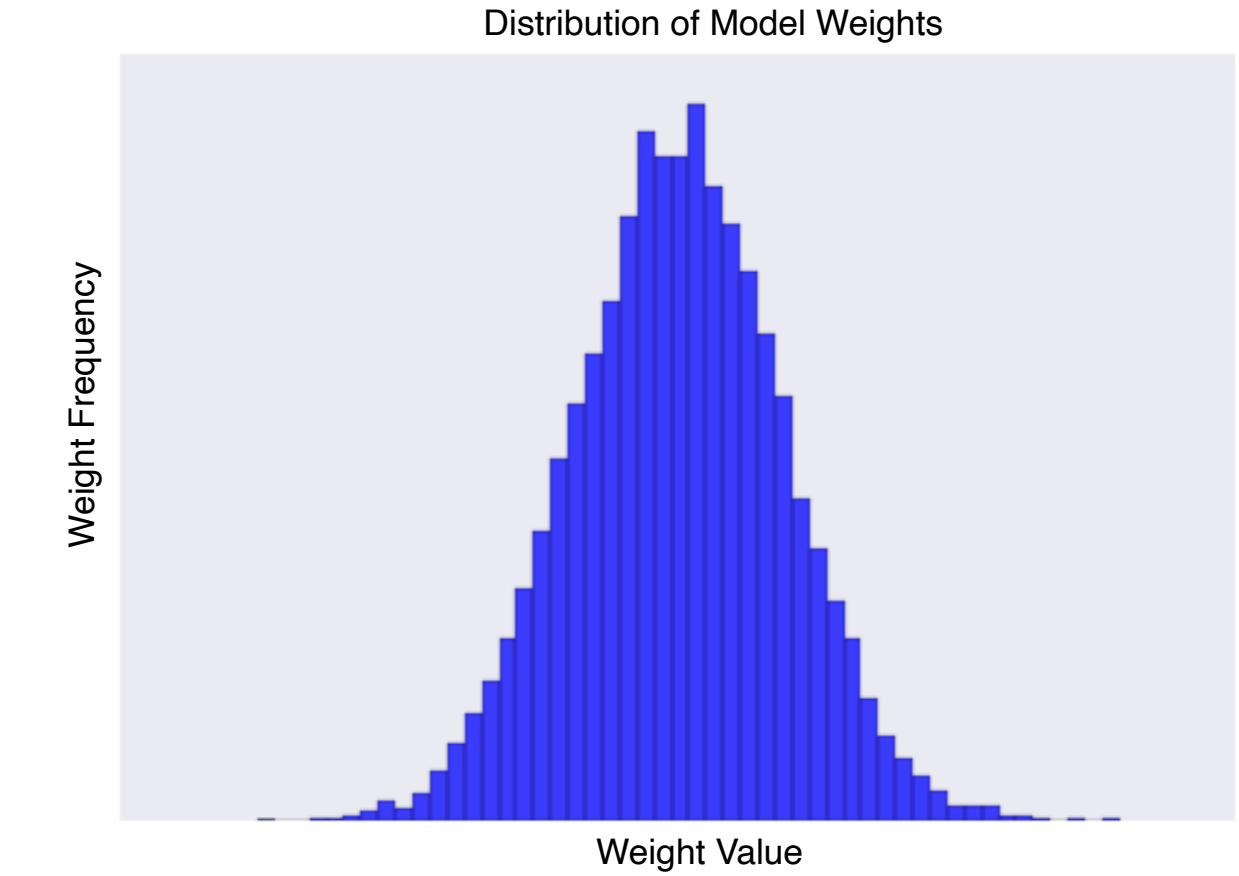

Figure 2. Histogram of weights.

Lowering the value of lambda tends to yield a flatter histogram, as shown in Figure 3.

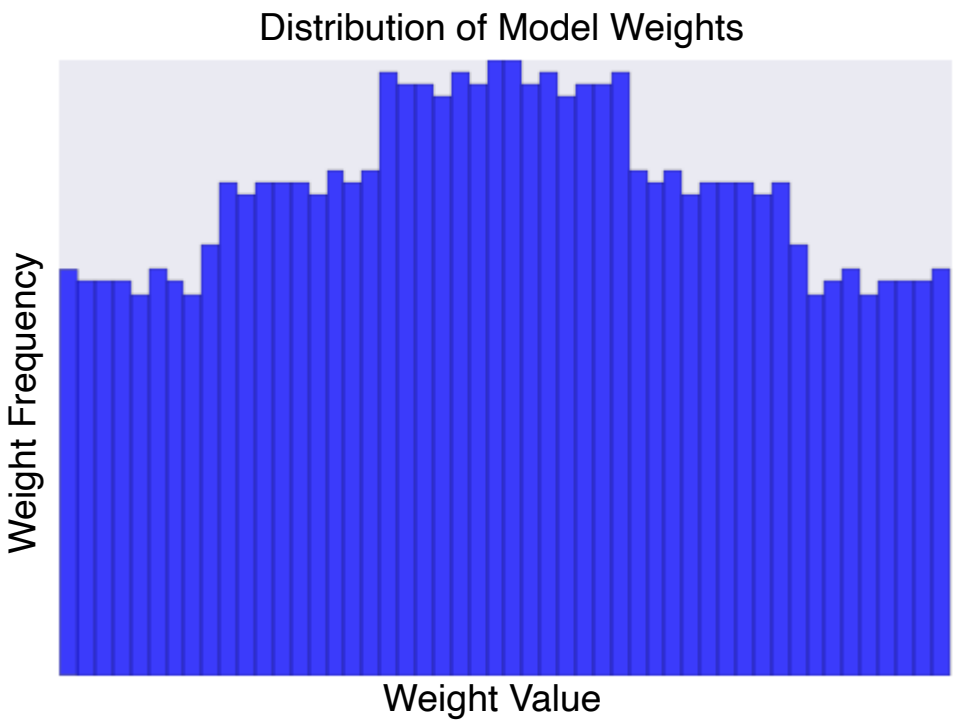

Figure 3. Histogram of weights produced by a lower lambda value.

When choosing a lambda value, the goal is to strike the right balance between simplicity and training-data fit:

- If your lambda value is too high, your model will be simple, but you run the risk of *underfitting* your data. Your model won't learn enough about the training data to make useful predictions.
- If your lambda value is too low, your model will be more complex, and you run the risk of *overfitting* your data. Your model will learn too much about the particularities of the training data, and won't be able to generalize to new data.

Setting lambda to zero removes regularization completely. In this case, training focuses exclusively on izing loss, which poses the highest possible overfitting risk.

The ideal value of lambda produces a model that generalizes well to new, previously unseen data. Unfortunately, that ideal value of lambda is data-dependent, so you'll need to do some tuning.

+ **Click the plus icon to learn about L_2 regularization and learning rate.**

There's a close connection between learning rate and lambda. Strong L_2 regularization values tend to drive feature weights closer to 0. Lower learning rates (with early stopping) often produce the same effect because the steps away from 0 aren't as large. Consequently, tweaking learning rate and lambda simultaneously may have confounding effects.

Early stopping means ending training before the model fully reaches convergence. In practice, we often end up with some amount of implicit early stopping when training in an online (https://developers.google.com/machine-learning/crash-course/production-ml-systems?authuser=0) (continuous) fashion. That is, some new trends just haven't had enough data yet to converge.

As noted, the effects from changes to regularization parameters can be confounded with the effects from changes in learning rate or number of iterations. One useful practice (when training across a fixed batch of data) is to give yourself a high enough number of iterations that early stopping doesn't play into things.