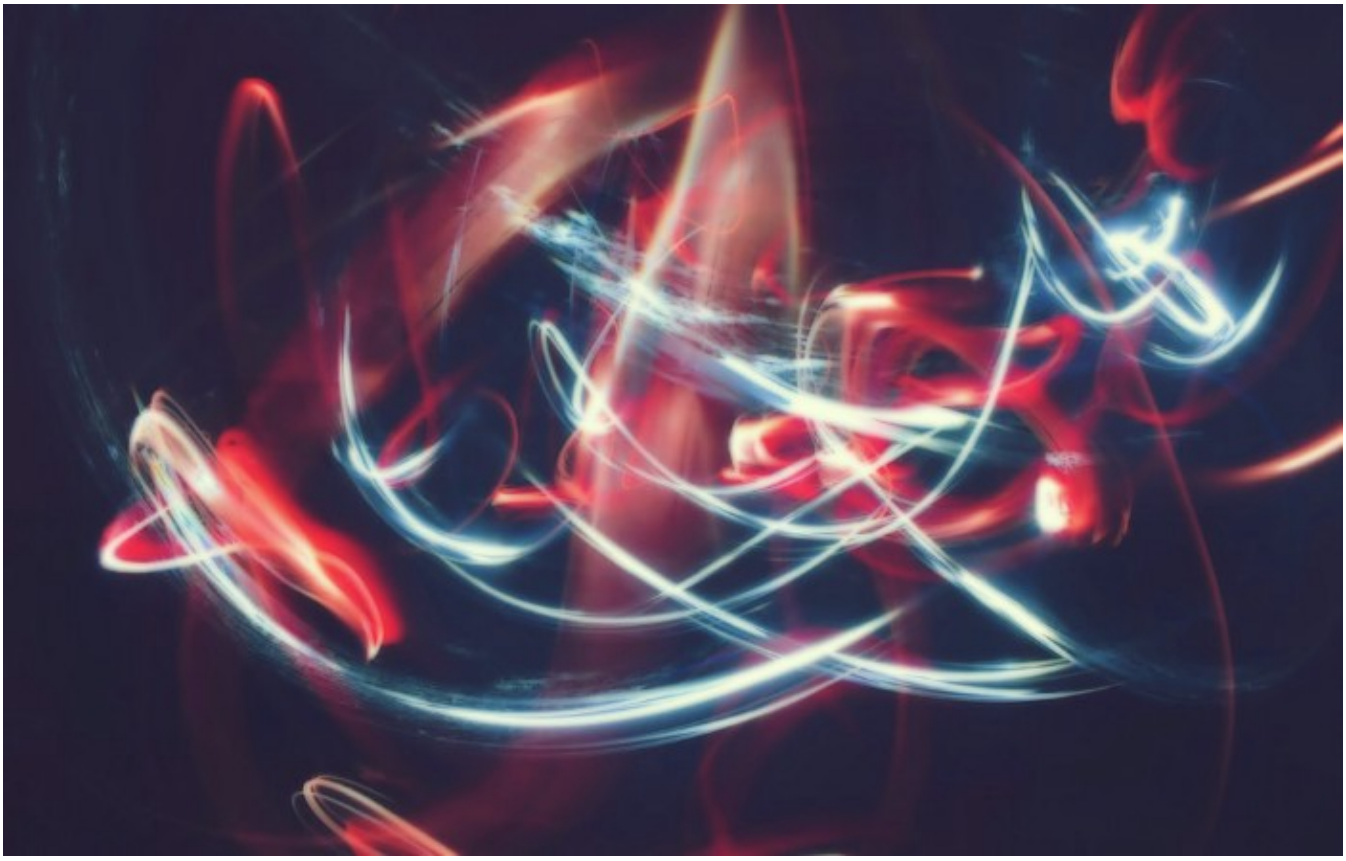

Source: Unsplash

Exploring the Simple & Satisfying Math Behind Regularization

The fascinating defense against overfitting

Andre Ye Sep 4, 2020 · 6 min read ★

Regularization is commonly used in machine learning, from the simple regression algorithm to the complex neural network, to prevent the algorithm from overfitting.

. . .

Consider a very simple regression task. There are six parameters (coefficients) represented as beta and and five inputs represented as x. The output of the model is simply each of the coefficients multiplied by the input (plus an intercept term, beta-0).

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_3 + \beta_5 x_5$$

Powered By Embed Fun

To humanize this problem, let's say that we are trying to predict a student's score on a test (the output of `f(x)`) based on five factors: 1) how much time they spend on homework (`hw`), 2) how many hours of sleep they got (`sleep`), 3) their score on the last test (`score`), 4) their current class GPA (`gpa`), and 5) if they ate food before taking the test (`food`).

$$f(x) = \beta_0 + \beta_1 \cdot \text{hw} + \beta_2 \cdot \text{sleep} + \beta_3 \cdot \text{score} + \beta_4 \cdot \text{gpa} + \beta_5 \cdot \text{food}$$

Powered By Embed Fun

The goal of the regression model is to minimize a 'loss', which attempts to quantify the error. This is usually done with the mean squared error. For example, if we have a dataset of inputs x and targets y, for each data point we would evaluate the difference in predicted target and actual target, square it, and average this among all items. For the purpose of brevity, it can be expressed shorthand like this (E for 'expected value' or average).

$$l(x, y) = E[(f(x) - y)^2]$$

Hence, linear regression is trained to optimize its parameters along this loss function l. However, note that our model has many features, which usually correlates to a higher complexity, similar to increasing the number of degrees in a polynomial. Are all of the features relevant? If they aren't, they may just be providing another degree of freedom for the model to overfit.

Say that a student's current GPA and whether or not they had food that morning turn out to provide minimal benefit while causing the model to overfit. If those coefficients are to be set to 0, then those features will be eliminated from the model altogether.

$$f(x) = \beta_0 + \beta_1 \cdot \text{hw} + \beta_2 \cdot \text{sleep} + \beta_3 \cdot \text{score} + 0 \cdot \text{gpa} + 0 \cdot \text{food}$$

This results in a much simpler model that has a lesser capability to overfit to the data. In a sense, this act of lowering coefficients towards zero is like a feature selection.

$$f(x) = \beta_0 + \beta_1 \cdot \text{hw} + \beta_2 \cdot \text{sleep} + \beta_3 \cdot \text{score}$$

Regularization is predicated on the idea that larger parameters generally lead to overfitting, because they a) are not zero, so it adds another variable/degree of freedom into the ecosystem b) can cause big swings and unnatural volatility in prediction. This — high variance — is a telltale sign of overfitting.

Let's explore two types of regularization: L1 and L2 regularization.

L1 regularization slightly alters the loss function used in linear regression by adding a 'regularization term', in this case '$\lambda E[|\beta|]$'. Let's break this down: $E[|\beta|]$ means 'average of the absolute value of the parameters', and λ is a scaling factor that determines how much the average of the parameters should incur on the total loss.

$$l(x, y) = E[(f(x) - y)^2] + \lambda E[|\beta|]$$

Powered By Embed Fun

This encourages, overall, smaller parameters. When the model adjusts its coefficients based on the loss with the general goal to reduce their value, it must determine if a feature is valuable enough to keep because it increases the predictive power *more* than it does the regularization term. These features are fundamentally more profitable to keep.

Then, less useful features are discarded and the model becomes simpler.

L2 regularization is essentially the same, but parameters are squared before they are averaged. Hence, the regularization is the average of the squares of parameters. This tries to accomplish the same task of encouraging the model to reduce the overall value of coefficients, but with different results.

$$l(x, y) = E[(f(x) - y)^2] + \lambda E[\beta^2]$$

Powered By Embed Fun

cancelled out.

To explore the different effect L1 and L2 normalization have, we need to look at their derivatives, or the slope of their functions at a certain point. To simplify, we can represent:

- L1 regularization as $y = x$. Derivative is 1.
- L2 regularization as $y = x^2$. Derivative is $2x$.

This means that:

- In L1 regularization, if a parameter decreases from 5 to 4, the corresponding regularization decreases 5–4 = 1.
- In L2 regularization, if a parameter decreases from 5 to 4, the corresponding regularization decreases 25–16 = 9.

While in L1 regularization, the rewards for reducing parameters is constant, in L2 regularization the reward gets smaller and smaller as the parameter nears zero. Going from a parameter value of 5 to 4 yields a decrease of 9 but reducing from 1 to 0 only yields and improvement of 1.

As a note, remember that the model only cares about relative rewards. The absolute value of a reward is irrelevant to us because the lambda parameter can always scale it up or down. What is important is how much of a decrease or increase a model will gain from a certain change in parameter.

Hence, in using L2, the model may decide that it is worth 'keeping a feature' (not discarding, or reducing the parameter to 0) because:

- It provides a substantial amount of predictive power (decreases the first term in the loss, $E[(f(x)-y)^2]$).
- There isn't much gain to be had reducing the parameter's value because it is already close to 0.
- Reducing the parameter would eliminate the gain in the first term for a much smaller gain in the second term of the loss function ($\lambda E[\beta^2]$).

- L1 regularization will produce simpler models with fewer features, since it provides consistent rewards to reduce parameter values. It can be thought of as a 'natural' method of variable selection, and can remove, for one, multicollinear variables.

- L2 regularization will produce more complex models with parameters *near* but likely *not at* zero, since it provides diminishing rewards to reduce parameter values. It's able to learn more complex data patterns.

Both regularization methods reduce the model's ability to overfit by preventing each of the parameters from having too big an effect on the final result, but lead to two different outcomes.

If you're looking for a simple and lightweight model, L1 regularization is the way to go. It takes a no-nonsense approach towards eliminating variables that have no profound impact on the output. In regression, this is known as 'LASSO regression', and is available in standard libraries like `sci-kit learn`.

On the other hand, if your task is more complicated, for example with regularization in neural networks, it's likely a bad idea to use L1, which could kill numerous hyperparameters by setting them to zero. L2 regularization is usually recommended in neural networks because it acts as a guardrail but doesn't interfere too much with the complex workings of neurons.

L2 regression is known as 'ridge regression', and is available for implementation in standard libraries. In neural networks, dropout is preferred to L2 regularization as more 'natural', but there are many use cases in which the latter or both should be used.

. . .

Key Points

- Regularization prevents overfitting in machine learning models by reducing the overall impact any feature can have on the outcome.

- Both L1 and L2 regularization continuously pressure the model to reduce their parameters. The former gives constant rewards but the latter gives diminishing rewards depending on how close the parameter is to 0.

the model choosing to keep more important features.

- L1 will result in many less relevant variables being eliminated all together (coefficient set to 0), whereas L2 will result in less relevant variables still existing but with small coefficients.

. . .

Thanks for reading!

Data Science Machine Learning Deep Learning Statistics Artificial Intelligence