Regularization for Sparsity: L_1 Regularization

ıted Time: 5 minutes

Sparse vectors often contain many dimensions. Creating a feature cross (https://developers.google.com/machine-learning/crash-course/feature-crosses/video-lecture?authuser=0) results in even more dimensions. Given such high-dimensional feature vectors, model size may become huge and require huge amounts of RAM.

In a high-dimensional sparse vector, it would be nice to encourage weights to drop to exactly `0` where possible. A weight of exactly 0 essentially removes the corresponding feature from the model. Zeroing out features will save RAM and may reduce noise in the model.

For example, consider a housing data set that covers not just California but the entire globe. Bucketing global latitude at the minute level (60 minutes per degree) gives about 10,000 dimensions in a sparse encoding; global longitude at the minute level gives about 20,000 dimensions. A feature cross of these two features would result in roughly 200,000,000 dimensions. Many of those 200,000,000 dimensions represent areas of such limited residence (for example, the middle of the ocean) that it would be difficult to use that data to generalize effectively. It would be silly to pay the RAM cost of storing these unneeded dimensions. Therefore, it would be nice to encourage the weights for the meaningless dimensions to drop to exactly 0, which would allow us to avoid paying for the storage cost of these model coefficients at inference time.

We might be able to encode this idea into the optimization problem done at training time, by adding an appropriately chosen regularization term.

Would L_2 regularization accomplish this task? Unfortunately not. L_2 regularization encourages weights to be small, but doesn't force them to exactly 0.0.

An alternative idea would be to try and create a regularization term that penalizes the count of non-zero coefficient values in a model. Increasing this count would only be justified if there was a sufficient gain in the model's ability to fit the data. Unfortunately, while this count-based approach is intuitively appealing, it would turn our convex optimization

problem into a non-convex optimization problem. So this idea, known as L_0 regularization isn't something we can use effectively in practice.

However, there is a regularization term called L_1 regularization that serves as an approximation to L_0, but has the advantage of being convex and thus efficient to compute. So we can use L_1 regularization to encourage many of the uninformative coefficients in our model to be exactly 0, and thus reap RAM savings at inference time.

L_1 vs. L_2 regularization.

L_2 and L_1 penalize weights differently:

- L_2 penalizes $weight^2$.
- L_1 penalizes $|weight|$.

Consequently, L_2 and L_1 have different derivatives:

- The derivative of L_2 is 2 * *weight*.
- The derivative of L_1 is *k* (a constant, whose value is independent of weight).

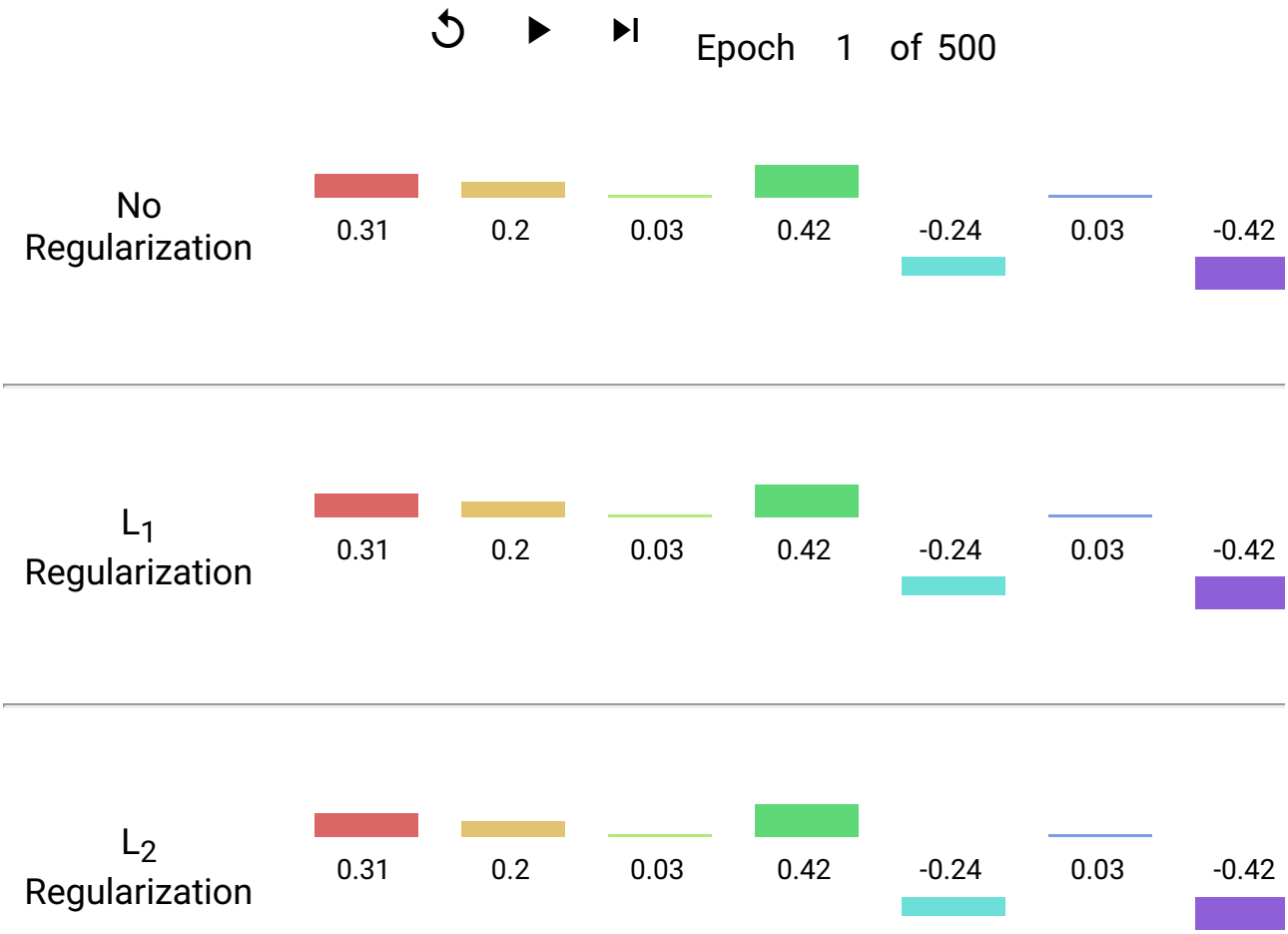
You can think of the derivative of L_2 as a force that removes x% of the weight every time. As Zeno (https://wikipedia.org/wiki/Zeno%27s_paradoxes#Dichotomy_paradox) knew, even if you remove x percent of a number *billions of times*, the diminished number will still never quite reach zero. (Zeno was less familiar with floating-point precision limitations, which could possibly produce exactly zero.) At any rate, L_2 does not normally drive weights to zero.

You can think of the derivative of L_1 as a force that subtracts some constant from the weight every time. However, thanks to absolute values, L_1 has a discontinuity at 0, which causes subtraction results that cross 0 to become zeroed out. For example, if subtraction would have forced a weight from +0.1 to -0.2, L_1 will set the weight to exactly 0. Eureka, L_1 zeroed out the weight.

L_1 regularization—penalizing the absolute value of all the weights—turns out to be quite efficient for wide models.

Note that this description is true for a one-dimensional model.

Click the Play button (▶) below to compare the effect L_1 and L_2 regularization have on a network of weights.



erms

nvex optimization
://developers.google.com/machine-ıg/glossary?authuser=0#convex_optimization)

regularization
://developers.google.com/machine-ıg/glossary?authuser=0#L2_regularization)

- L_1 regularization (https://developers.google.com/machine-learning/glossary?authuser=0#L1_regularization)
- one-hot encoding (https://developers.google.com/machine-learning/glossary?authuser=0#one-hot_encoding)

eight (https://developers.google.com/machine-learning/glossary?authuser=0#weight)