Regularization in Deep Learning - L1, L2, and Dropout


Artem Oppermann
08 October, 2019

Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain.

In this article, we will address the most popular regularization techniques which are called L1, L2, and dropout.

Table of Content

---

1. Recap: Overfitting

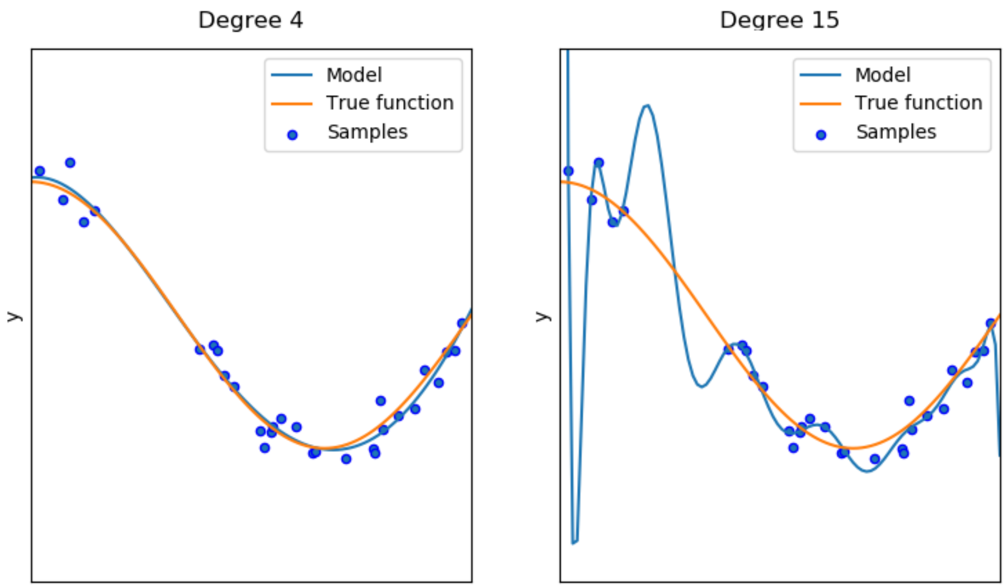
One of the most important aspects when training neural networks is avoiding overfitting. We have addressed the issue of overfitting in more detail in this article.

However let us do a quick recap: Overfitting refers to the phenomenon where a neural network models the training data very well but fails when it sees new data from the same problem domain. Overfitting is caused by noise in the training data that the neural network picks up during training and learns it as an underlying concept of the data.

This learned noise, however, is unique to each training set. As soon as the model sees new data from the same problem domain, but that does not contain this noise, the performance of the neural network gets much worse.

"Why does the neural network picks up that noise in the first place?"

The reason for this is that the complexity of this network is too high. A fit of a neural network with higher complexity is shown in the image on the right-hand side.



Good Fit High Variance

The model with a higher complexity is able to pick up and learn patterns (noise) in the data that are just caused by some random fluctuation or error. The network would be able to model each data sample of the distribution one-by-one, while not recognizing the true function that describes the distribution.

New arbitrary samples generated with the true function would have a high distance to the fit of the model. We also say that the model has a high variance.

On the other hand, the lower complexity network on the left side models the distribution much better by not trying too hard to model each data pattern individually.

In practice, overfitting causes the neural network model to perform very well during training, but the performance gets much worse during inference time when faced with brand new data.

In short: Less complex neural networks are less susceptible to overfitting. To prevent overfitting or a high variance we must use something that is called regularization.

---

2. What is Regularization?

Simple speaking: Regularization refers to a set of different techniques that lower the complexity of a neural network model during training, and thus prevent the overfitting.

There are three very popular and efficient regularization techniques called L1, L2, and dropout which we are going to discuss in the following.

---

3. L2 Regularization

The L2 regularization is the most common type of all regularization techniques and is also commonly known as weight decay or Ride Regression.

The mathematical derivation of this regularization, as well as the mathematical explanation of why this method works at reducing overfitting, is quite long and complex. Since this is a very practical article I don't want to focus on mathematics more than it is required. Instead, I want to convey the intuition behind this technique and most importantly how to implement it so you can address the overfitting problem during your deep learning projects.

During the L2 regularization the loss function of the neural network as extended by a so-called regularization term, which is called here **Ω**.

$$\Omega(W) = ||W||_2^2 = \sum_i \sum_j w_{ij}^2$$

The regularization term **Ω** is defined as the Euclidean Norm (or L2 norm) of the weight matrices, which is the sum over all squared weight values of a weight matrix. The regularization term is weighted by the scalar alpha divided by two and added to the regular loss function that is chosen for the current task. This leads to a new expression for the loss function:

$$\hat{\mathcal{L}}(W) = \frac{\alpha}{2}||W||_2^2 + \mathcal{L}(W) = \frac{\alpha}{2}\sum_i \sum_j w_{ij}^2 + \mathcal{L}(W)$$

Alpha is sometimes called as the **regularization rate** and is an additional hyperparameter we introduce into the neural network. Simply speaking alpha determines how much we regularize our model.

In the next step we can compute the gradient of the new loss function and put the gradient into the update rule for the weights:

$$\nabla_W \hat{\mathcal{L}}(W) = \alpha W + \nabla_W \mathcal{L}(W)$$

$$W_{new} = W_{old} - \epsilon(\alpha W_{old} + \nabla_W \mathcal{L}(W_{old}))$$

Some reformulations of the update rule lead to the expression which very much looks like the update rule for the weights during regular gradient descent:

$$W_{new} = (1 - \epsilon\alpha)W_{old} - \epsilon \nabla_W \mathcal{L}(W_{old})$$

The only difference is that by adding the regularization term we introduce an additional subtraction from the current weights (first term in the equation).

In other words independent of the gradient of the loss function we are making our weights a little bit smaller each time an update is performed.

---

4. L1 Regularization

In the case of L1 regularization (also knows as Lasso regression), we simply use another regularization term **Ω**. This term is the sum of the absolute values of the weight parameters in a weight matrix:

$$\Omega(W) = ||W||_1 = \sum_i \sum_j |w_{ij}|$$

As in the previous case, we multiply the regularization term by alpha and add the entire thing to the loss function.

$$\hat{\mathcal{L}}(W) = \alpha||W||_1 + \mathcal{L}(W)$$

The derivative of the new loss function leads to the following expression, which the sum of the gradient of the old loss function and sign of a weight value times alpha.

$$\nabla_W \hat{\mathcal{L}}(W) = \alpha sign(W) + \nabla_W \mathcal{L}(W)$$

---

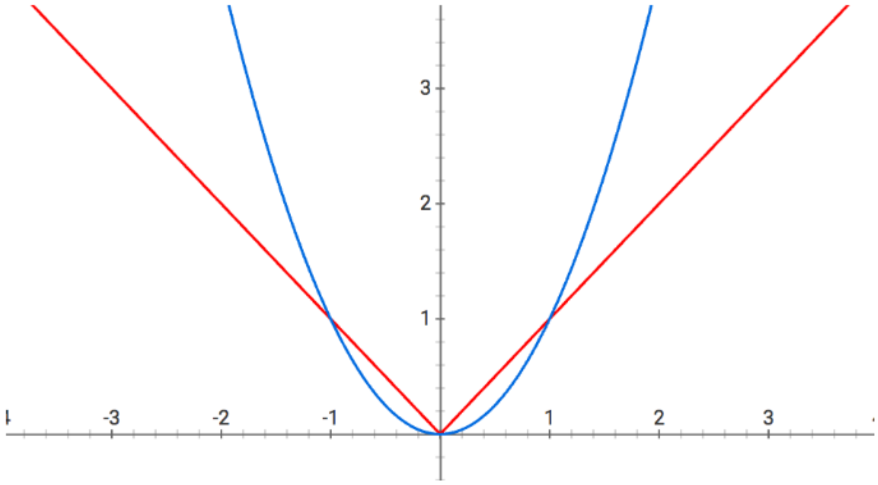
5. Why do L1 and L2 Regularizations work?

The question you might be asking yourself right now is:

"Why does all of this help to reduce the overfitting issue?"

Let’s tackle this question.

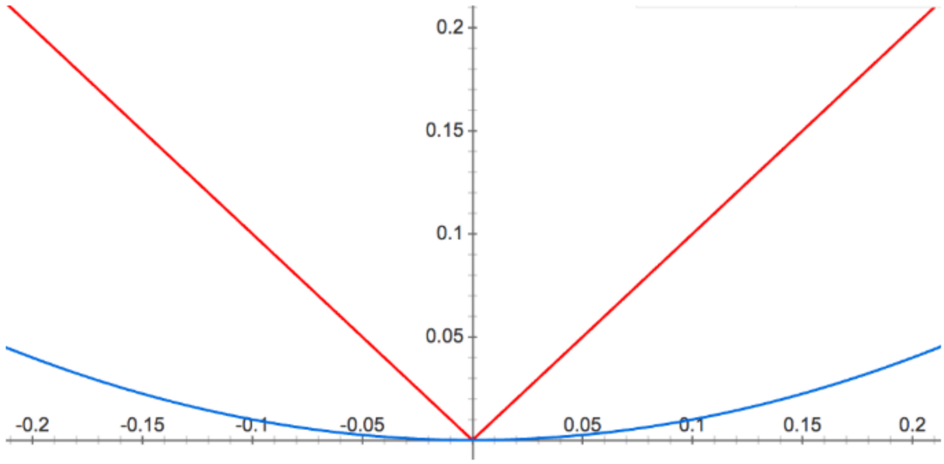
Please consider the plots of the ***abs*** and ***square*** functions, where ***abs*** represents the operation performed during L1 and ***square*** the operation performed during L2 regularization.



When we minimize a loss function with a regularization term **Ω** each of the weights is “pulled” towards zero. Please think of each weight parameter lying on one of the above curves and being subjected to “gravity” proportional to the regularization rate **α**.

During L1-regularization, the weight values are pulled towards zero proportionally to their absolute values—they lie on the red curve. During L2-regularization, the weight values are pulled towards zero proportionally to their squares—the blue curve.

At first, L2 seems more severe, but the caveat is that approaching zero, a different picture emerges:



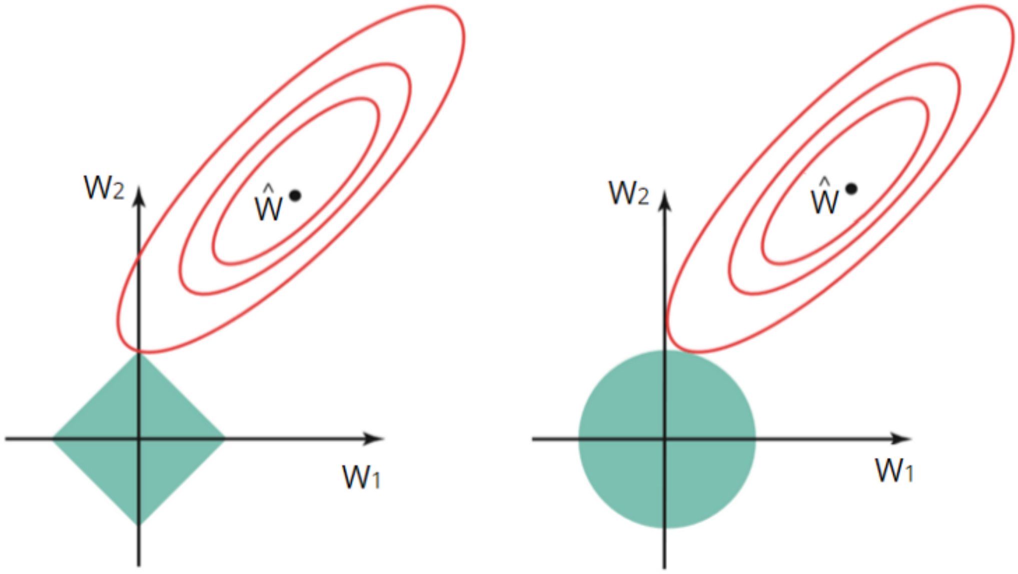
In the case of L2 regularization, our weight parameters decrease, but not necessarily become zero, since the curve becomes flat near zero. On the other hand during the L1 regularization, the weight are always forced all the way towards zero.

We can also take a different and more mathematical view on this.

In the case of L2, you can think of solving an equation, where the sum of squared weight values is equal or less than a value ***s***. **s** is the constant that exists for each possible value of the regularization term $\boldsymbol{\alpha}$. For just two weight values **W1** and **W2** this equation would look as follows: $W_1^2 + W_2^2 \leq s$

On the other hand, the *L1 regularization can be thought of as an equation where the sum of modules of weight values is less than or equal to a value* ***s***. This would look like the following expression: $|W_1| + |W_2| \leq s$

Basically the introduced equations for L1 and L2 regularizations are constraint functions, which we can visualize:



The left image shows the constraint function (green area) for the L1 regularization and the right image shows the constraint function for the L2 regularization. The red ellipses are contours of the loss function that is used during the gradient descent. In the center of the contours there is a set of optimal weights for which the loss function has a global minimum.

In the case of L1 and L2 regularization, the estimates of W1 and W2 are given by the first point where the ellipse intersects with the green constraint area.

Since L2 regularization has a circular constraint area, the intersection won't generally occur on an axis, and this the estimates for W1 and W2 will be exclusively non-zero.

In the case of L1, the constraints area has a diamond shape with corners. And thus the contours of the loss function will often intersect the constraint region at an axis. Then this occurs, one of the estimates (W1 or W2) will be zero.

In a high dimensional space, many of the weight parameters will equal zero simultaneously.

5.1 What does Regularization achieve?

- Performing L2 regularization encourages the weight values towards zero (but not exactly zero)
- Performing L1 regularization encourages the weight values to be zero

Intuitively speaking smaller weights reduce the impact of the hidden neurons. In that case, those hidden neurons become neglectable and the overall complexity of the neural network gets reduced.

As mentioned earlier: less complex models typically avoid modeling noise in the data, and therefore, there is no overfitting.

But you have to be careful. When choosing the regularization term **α.**The goal is to strike the right balance between low complexity of the model and accuracy

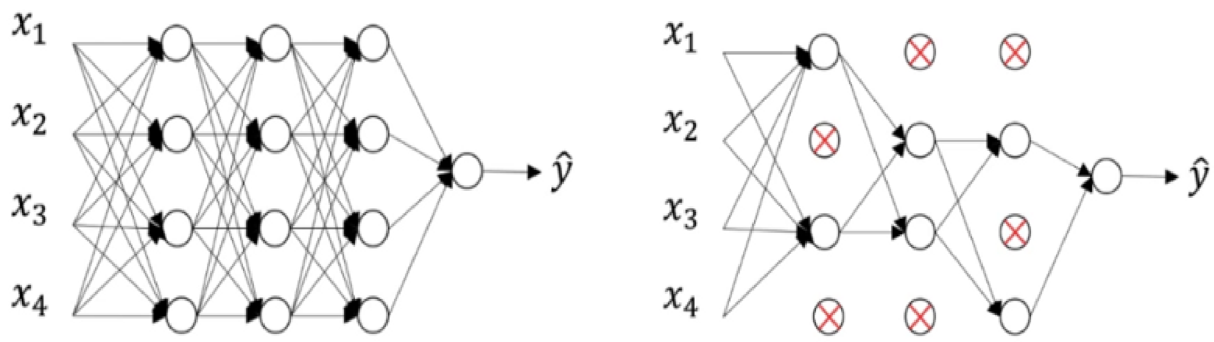
- If your alpha value is too high, your model will be simple, but you run the risk of *underfitting* your data. Your model won't learn enough about the training data to make useful predictions.
- If your alpha value is too low, your model will be more complex, and you run the risk of *overfitting* your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

---

6. Dropout

In addition to the L2 and L1 regularization, another famous and powerful regularization technique is called the dropout regularization. The procedure behind dropout regularization is quite simple.

In a nutshell, dropout means that during training with some probability **P** a neuron of the neural network gets turned off during training. Let's look at a visual example.



Assume on the left side we have a feedforward neural network with no dropout. Using dropout with let's say a probability of **P=0.5** that a random neuron gets turned off during training would result in a neural network on the right side.

In this case, you can observe that approximately half of the neurons are not active and are not considered as a part of the neural network. And as you can observe the neural network becomes simpler.

A simpler version of the neural network results in less complexity that can reduce overfitting. The deactivation of neurons with a certain probability **P** is applied at each forward propagation and weight update step.

---

6. Take-Home-Message

- Overfitting occurs in more complex neural network models (many layers, many neurons)
- Complexity can be reduced by using L1 and L2 regularization as well as dropout
- L1 regularization forces the weight parameters to become zero
- L2 regularization forces the weight parameters towards zero (but never exactly zero)

- Smaller weight parameters make some neurons neglectable → neural network becomes less complex
- During dropout, some neurons get deactivated with a random probability **P**