**Weight Decay == L2 Regularization?**

*Neural Networks are great function approximators and feature extractors but sometimes their weights become too specialized and cause overfitting. That’s where the concept of Regularization comes into picture which we will discuss along with slight differences between two major weight regularization techniques which are mistakenly considered the same.*



[(source](https://unsplash.com/photos/P8lU5CgYybM))

**Introduction:**

Neural Networks was first introduced in 1943 by Warren McCulloch and Walter Pitts but weren’t popular enough as they required large amounts of data and computation power which were not feasible at that time. But as the above constraints became feasible along with other training advancements such as parameter initialization and better activation functions, they again started to dominate the various competitions and found applications in various human assistive technologies.  
Today Neural Networks form the backbone of many famous applications like Self-Driving Car, Google Translate, Facial Recognition Systems etc and are applied in almost all technologies used by evolving human race.

Neural Networks are very good at approximating functions be linear or non-linear and are also terrific when extracting features from the input data. This capability makes them perform wonders over a large range of tasks be it computer vision domain or language modelling. But as we all have heard the famous saying :  
*“With Great Power Comes Great Responsibility”.*

This saying also applies to the all-mighty neural nets. Their power of being great function approximators sometimes causes them to overfit the dataset by approximating a function which will perform extremely well on the data on which it was trained on but fails miserably when tested on a data it hasn’t seen before.

To be more technical, the neural networks learn weights which are more specialized on the given data and fails to learn features which can be generalized.

To solve the problem of overfitting, a class of techniques known as Regularization is applied to reduce the complexity of the model and constraint weights in a manner which forces the neural network to learn generalizable features.

**Regularization:**

Regularization may be defined as any change we make to the training algorithm in order to reduce the generalization error but not the training error.

There are many regularization strategies. Some put extra constraints on the models such as adding constraints to parameter values while some add extra terms to the objective function which can be thought as adding indirect or soft constraints on the parameter values. If we use these techniques carefully, this can lead to improved performance on the test set.

In the context of deep learning, most regularization techniques are based on regularizing the estimators.

While regularizing an estimator, there is a tradeoff where we have to choose a model with increased bias and reduced variance. An effective regularizer is one which makes a profitable trade, reducing variance significantly while not overly increasing the bias.

The major regularization techniques used in practice are:

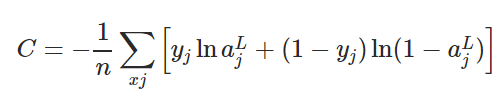
1. L2 Regularization
2. L1 Regularization
3. Data Augmentation
4. Dropout
5. Early Stopping

In this post, we mainly focus on L2 Regularization and argue whether we can refer L2 regularization and weight decay as two faces of the same coin.

**L2 Regularization:**

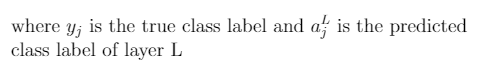
L2 regularization belongs to the class of regularization techniques referred to as parameter norm penalty. It is referred to this because in this class of techniques, the norm of a particular parameter mostly weights are added to the objective function being optimized. In L2 norm, an extra term often referred to as **regularization term** is added to the cost function of the network.   
For example:

Let us consider, the cross-entropy cost function which is defined as shown below.

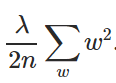




**Figure 1.** Cross-Entropy loss function



To apply L2 regularization to any network having cross-entropy loss, we add the regularizing term to the cost function where the regularization term is shown in **Figure 2.**

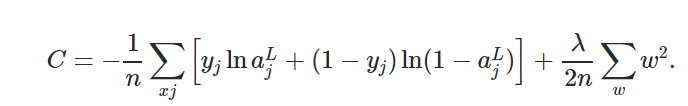


**Figure 2.** L2 norm or Euclidean Norm

In **Figure 2** λ is the regularization parameter and is directly proportional to the amount of regularization applied. If λ =0, then no regularization is applied and when λ is 1 maximum regularization is applied to the network.

λ is a hyper-parameter which means it is not learned during the training but is tuned by the user manually or using some hyperparameter tuning techniques such as random-search.

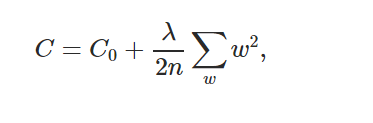
Now let us put this together and form the final equation of L2 regularization applied to the cross-entropy loss function given by **Figure 3**.

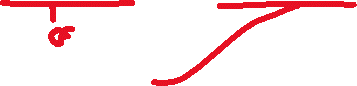




**Figure 3.**Final L2 Regularized Cost Function

The above example showed L2 regularization applied to cross-entropy loss function but this concept can be generalized to all the cost-functions available. A more general formula of L2 regularization is given below in **Figure 4** where Co is the unregularized cost function and C is the regularized cost function with the regularization term added to it.





**Figure 4.** General Form of L2 Regularization for any cost function

**Note**: *We don’t consider the bias of the network when regularizing the network because of the following reasons:*

1. Bias typically require less data as compared to weights to fit accurately. Each weight specifies how two-variable interact (w and x) and hence fitting the weights well require observing both the variables in a variety of conditions whereas each bias controls only a single variable (b).

Hence we do not introduce much variance by leaving the bias unregularized.

2. Regularizing a bias can introduce a significant amount of underfitting.

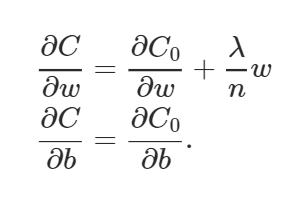
**Why L2 Regularization works??**

**Practical Reason:**

Let us try to understand the working of L2 regularization based on the gradient of the cost function.

If we take the partial derivative or gradient of equation presented in **Figure 4** i.e ∂C/∂w and ∂C/∂b for all weights and biases in the network.

Taking partial derivates we get:

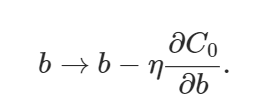




**Figure 5.** The gradient of the cost function with respect to weights and biases.

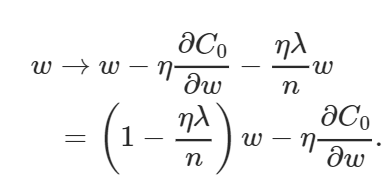
We can compute ∂C0/∂w and ∂C0/∂b terms mentioned in the above equations using the backpropagation algorithm.   
The partial derivation of the bias parameter will be unchanged as no regularization term is applied to it while the weight parameter will contain the extra ((λ/n)\*w) regularization term.

The learning rules for bias and weights hence become:





**Figure 6.** Gradient Descent Learning Rule for Bias Parameter





**Figure 7.** Gradient Descent Learning Rule for Weight Parameter

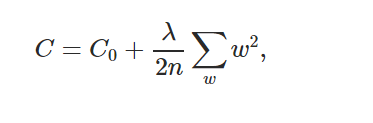
The above weight equation is similar to the usual gradient descent learning rule, except the now we first rescale the weights **w** by **(**1−(η\*λ)/n). This term is the reason why L2 regularization is often referred to as **weight decay** since it makes the weights smaller.

Hence you can see why regularization works, it makes the weights of the network smaller. The smallness of weights implies that the network behaviour won’t change much if we change a few random inputs here and there which in turn makes it difficult for the regularized network to learn local noise in the data.

This forces the network to learn only those features which are seen often across the training set.

**Personal Intuition:**

To think simply about L2 regularization from the viewpoint of optimizing the cost function, as we add the regularization term to the cost function we are actually increasing the value of the cost function. Hence, if the weights will be larger it will also make the cost to go up and the training algorithm will try to bring the weights down by penalizing the weights forcing them to take smaller values thereby regularizing the network.

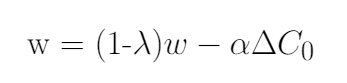




**Is L2 Regularization and Weight Decay the same thing?**

No L2 Regularization and Weight Decay are not the same things but can be made equivalent for SGD by a reparameterization of the weight decay factor based on the learning rate. Confused? Let me explain you in detail.

The Equation of weight decay is given below with λ being the decay factor.

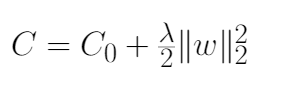




**Figure 8**: Weight Decay in Neural Networks

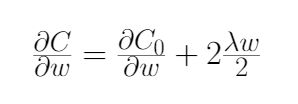
L2 regularization can be proved equivalent to weight decay in the case of SGD in the following proof:

1. Let us first consider the L2 Regularization equation given in **Figure 9** below. Our goal is to reparametrize it in such a way that it becomes equivalent to the weight decay equation give in **Figure 8**.



**Figure 9.** L2 Regularization in Neural Networks

2. To begin with, we find the partial derivative (Gradient) of the L2 Regularized Cost function with respect to parameter **w** as shown in **Figure 10**.



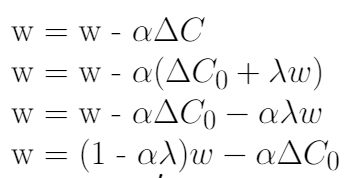


**Figure 10**. Partial Derivative of Loss Function C with respect to w



**Note:** Both the Notations in the figure means the same thing.

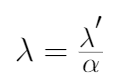
3. After we obtain the result of the partial derivative of the cost function (**Figure 10**), we substitute the result in the gradient descent learning rule as shown in **Figure 11**. After substituting we open the brackets and rearrange the terms to make it equivalent to weight decay equation (**Figure 8**) with certain assumptions.





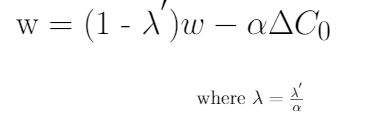
**Figure 11**. Substituting the Gradient of Cost Function in the Gradient Descent Rule and Rearranging terms.

4. As you can notice, the only difference between the final rearranged L2 regularization equation (**Figure 11**) and weight decay equation (**Figure 8**) is the α (learning rate) multiplied by λ (regularization term). To make the two-equation, we reparametrize the L2 regularization equation by replacing λ  
by λ′/α as shown in **Figure 12**.



**Figure 12.** Condition of Equivalence of L2 Regularization and Weight Decay

5. After replacing λ′ with λ, the L2 regularization equation is reparametrized and is now made equivalent to weight decay equation(**Figure 8**) as shown in **Figure 13**.



**Figure 13.** Reparametrized L2 Regularization equation

From the above proof, you must have understood why L2 regularization is considered equivalent to weight decay in case of SGD but it is not the case for other optimisation algorithms like Adam, AdaGrad etc which are based on adaptive gradients. In particular, when combined with adaptive gradients, L2 regularization leads to weights with large historic parameter and/or gradient amplitudes being regularized less than they would be when using weight decay. This causes adam to perform poorly when L2 regularization is used as compared to SGD. Weight Decay, on the other hand, performs equally on both SGD and Adam.

A shocking result is seen where SGD with momentum outperforms Adaptive gradients methods like Adam because common deep learning libraries implement the L2 regularization and not the original weight decay. Therefore, on tasks where using L2 regularization is beneficial for SGD, Adam leads to poor results than SGD with momentum.

**Conclusion**

Hence we conclude that though weight decay and L2 regularization may reach equivalence under some conditions still are slightly different concepts and should be treated differently otherwise can lead to unexplained performance degradation or other practical problems.

I hope you liked this article and learn something new. If you have any queries or want to discuss further, feel free to connect with me through [Twitter](https://twitter.com/Perceptron97) or [Linkedin](https://www.linkedin.com/in/divyanshu-mishra-ai/).

**Further Readin*g:***

1. [*Decoupled Weight Decay Regularization*](https://arxiv.org/abs/1711.05101)

**References:**

1. [Neural Networks and Deep Learning](http://neuralnetworksanddeeplearning.com/).
2. [Deep Learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville.](https://www.deeplearningbook.org/)

3. [The Difference Between Neural Network L2 Regularization and Weight Decay](https://jamesmccaffrey.wordpress.com/2019/05/09/the-difference-between-neural-network-l2-regularization-and-weight-decay/)