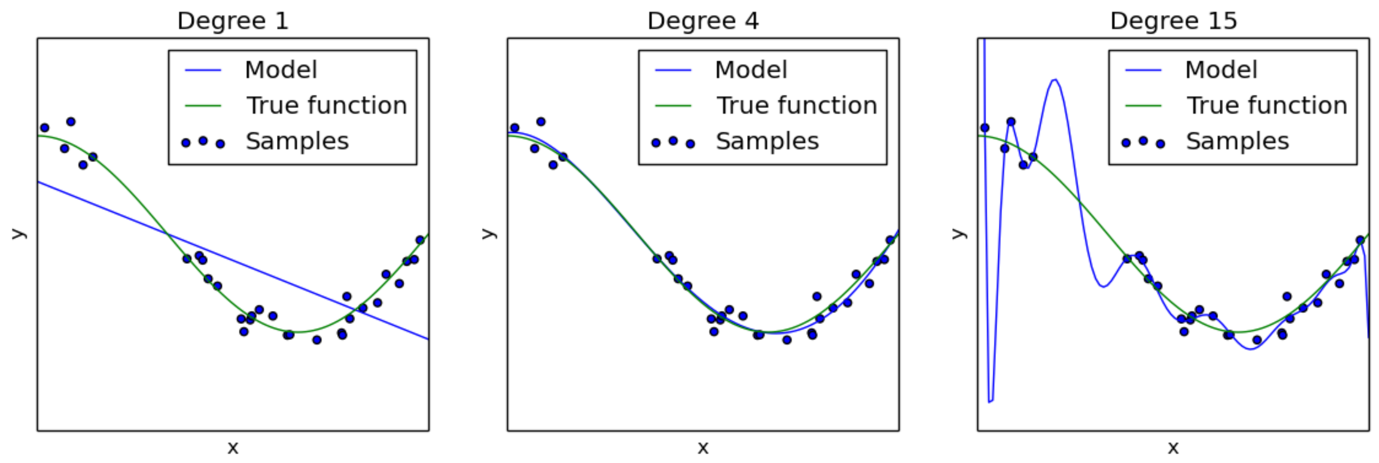
Regularization for Neural Networks

Ref :
https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/
https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

The central challenge in machine learning is that the model must perform well on new, previously unseen inputs — not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.

$$E_{test} - E_{train} = k(h/P)^{\alpha}$$

P is the number of training samples, h is the measure of effective capacity (complexity of the machine), α is a number between 0.5 and 1, and k is a constant.

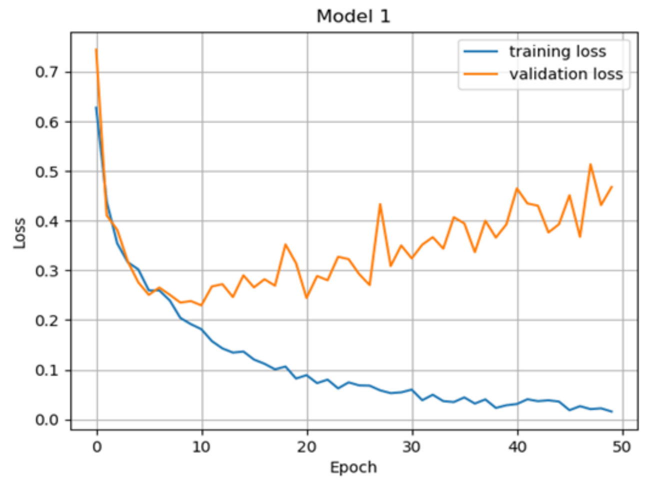


Underfit Model. A model that fails to sufficiently learn the problem and performs poorly on a training dataset and does not perform well on a validation/test.

Overfit Model. A model that learns the training dataset too well. Performing well on the training dataset but does not perform well on a validation dataset.

Good Fit Model. A model that suitably learns the training dataset and generalizes well to the validation/test dataset.

- We can address underfitting by increasing the capacity of the model. Because an underfit model is so easily addressed, it is more common to have an overfit model.
- An overfit model is easily diagnosed by monitoring the performance of the model during training by evaluating it on both a training dataset and on a holdout validation dataset.



There are two ways to approach an overfit model:

1. Reduce overfitting by training the network on more examples.
2. Reduce overfitting by changing the complexity of the network.

- A models capacity is its ability to fit a wide variety of functions.
- A benefit of very deep neural networks is that their performance continues to improve as they are fed larger and larger datasets.
- Reducing the capacity of the model reduces the likelihood of the model overfitting the training dataset, to a point where it no longer overfits.
- The capacity of a neural network model, it's complexity, is defined by both it's structure in terms of nodes and layers and the parameters in terms of its weights.

We can reduce the complexity of a neural network to reduce overfitting in one of two ways:

1. Change network complexity by changing the network structure (number of weights).
2. Change network complexity by changing the network parameters (values of weights).

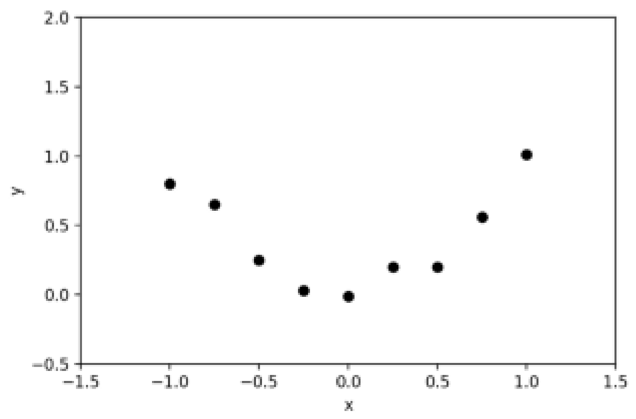
- The simplest and perhaps most common regularization method is to add a penalty to the loss function in proportion to the size of the weights in the model.

- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

- L1 and L2 are the most common types of regularization. This update the general cost function by adding another term known as the regularization term.

Cost function = Loss (say, binary cross entropy) + Regularization term

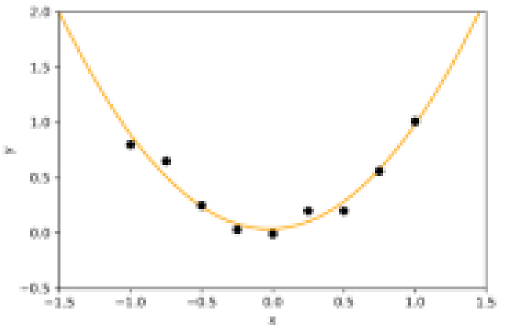
- Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.



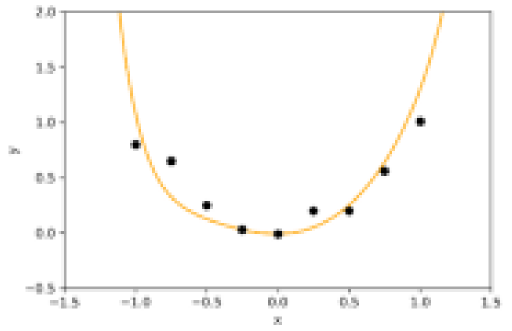
$$\text{L1 norm: } \|\mathbf{w}\|_1 = \sum_i^n |w_i|$$

$$\text{Squared L2 norm: } \|\mathbf{w}\|_2^2 = \sum_i^n w_i^2$$

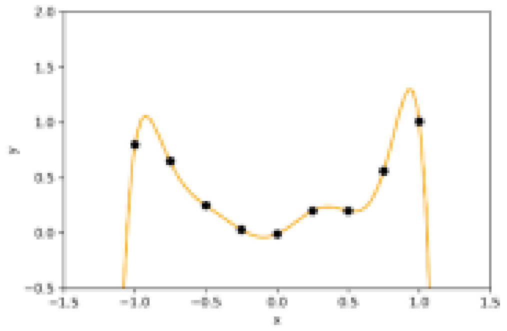
Figure a: $\hat{y} = 0.04 + 0.04x + 0.9x^2$
Figure b: $\hat{y} = -0.01 + 0.01x + 0.8x^2 + 0.5x^3 - 0.1x^4 - 0.1x^5 + 0.3x^6 - 0.3x^7 + 0.2x^8$
Figure c: $\hat{y} = -0.01 + 0.57x + 2.67x^2 - 4.08x^3 - 12.25x^4 + 7.41x^5 + 24.87x^6 - 3.79x^7 - 14.38x^8$



$\text{MSE} = 0.006$
$\text{L2 norm} = 0.90$
$\text{L1 norm} = 0.98$



$\text{MSE} = 0.035$
$\text{L2 norm} = 1.06$
$\text{L1 norm} = 2.32$



$\text{MSE} = 0$
$\text{L2 norm} = 32.69$
$\text{L1 norm} = 70.03$

https://towardsdatascience.com/visualizing-regularization-and-the-l1-and-l2-norms-d962aa769932

The idea of L2 regularization is to add an extra term to the cost function, a term called the *regularization term*. Here's the regularized cross-entropy:

$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_w w^2.$$

Mathematically speaking, we are adding a *regularization term* in order to prevent the coefficients to fit so perfectly to overfit. λ is the regularization coefficient which determines how much regularization we want.

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2,$$

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w$$
$$\frac{\partial C}{\partial b} = \frac{\partial C_0}{\partial b}.$$

$$b \rightarrow b - \eta \frac{\partial C_0}{\partial b}.$$

$$\begin{aligned} w &\rightarrow w - \eta \frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n} w \\ &= \left(1 - \frac{\eta\lambda}{n}\right) w - \eta \frac{\partial C_0}{\partial w}. \end{aligned}$$

L1 regularization: In this approach we modify the unregularized cost function by adding the sum of the absolute values of the weights:

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|.$$

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} \operatorname{sgn}(w),$$

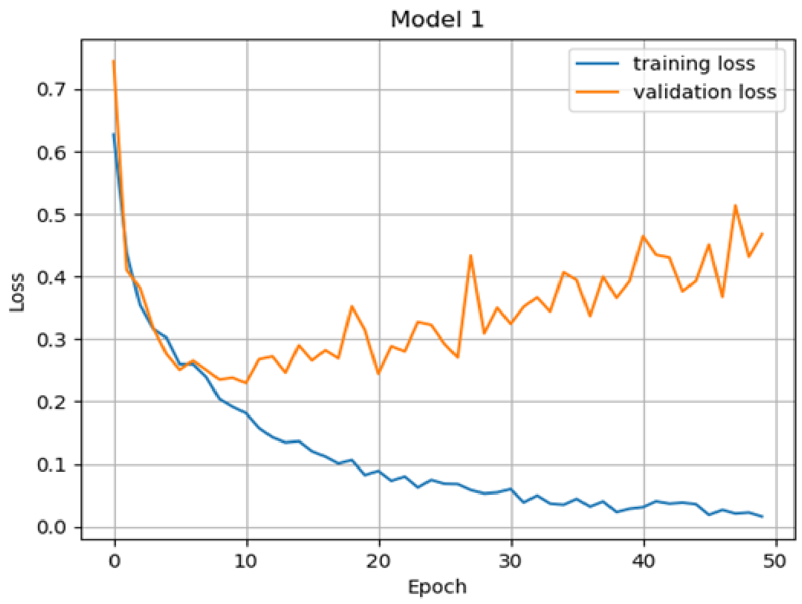
update rule for an L1 regularized network is

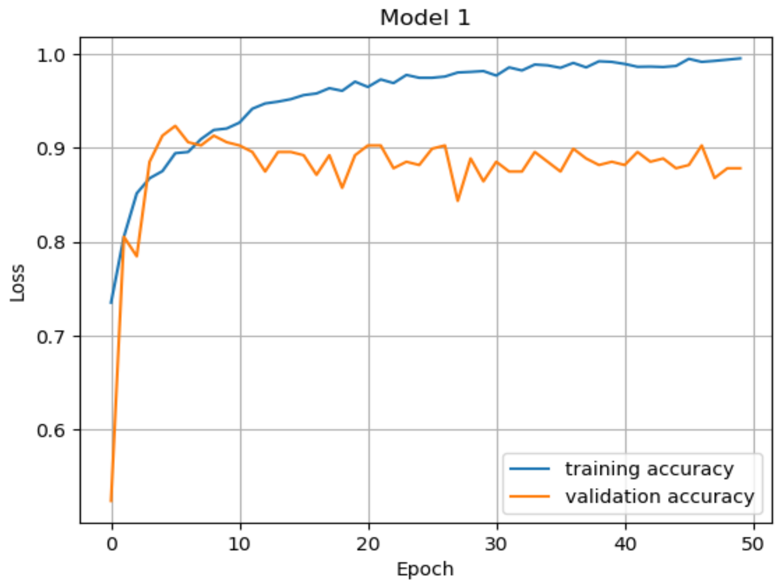
$$w \rightarrow w' = w - \frac{\eta \lambda}{n} \operatorname{sgn}(w) - \eta \frac{\partial C_0}{\partial w},$$

The net result is that L1 regularization tends to concentrate the weight of the network in a relatively small number of high-importance connections, while the other weights are driven toward zero.

Early stopping

- Early stopping is a form of regularization used to avoid overfitting.
- When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

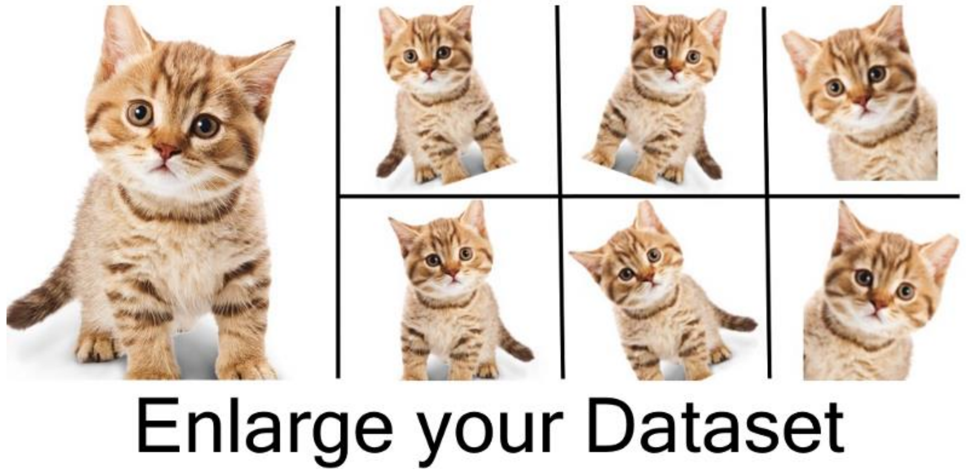


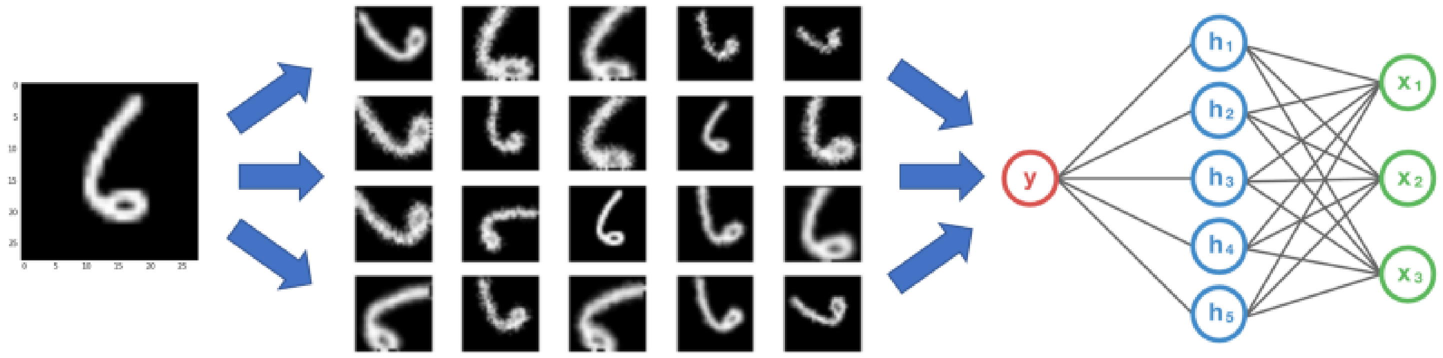


Data Augmentation

Data augmentation is a way of creating new 'data' with different orientations.

The benefits of this are two fold, the first being the ability to generate 'more data' from limited data and secondly it prevents over fitting.

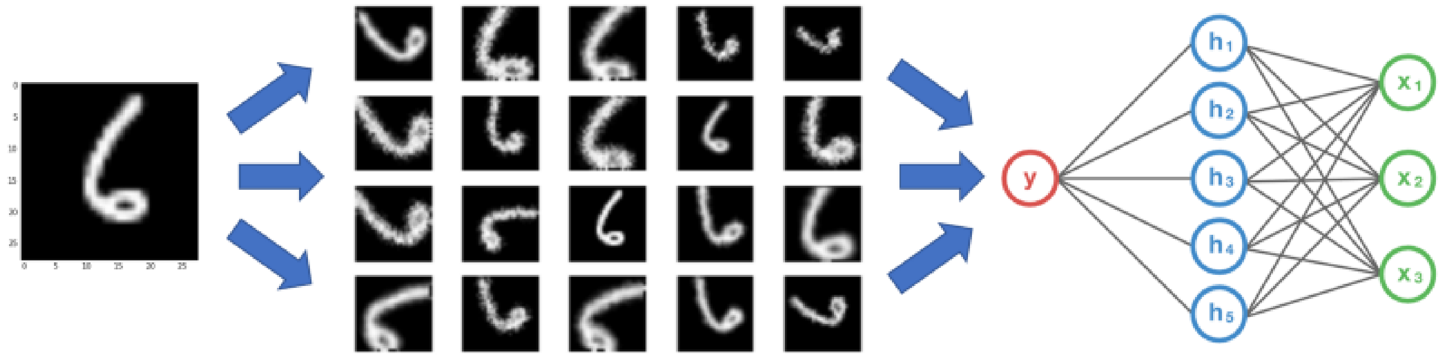




Offline and Online augmentation

Offline augmentation : This method is preferred for relatively smaller datasets, as you would end up increasing the size of the dataset by a factor equal to the number of transformations you perform (For example, by flipping all my images, I would increase the size of my dataset by a factor of 2).

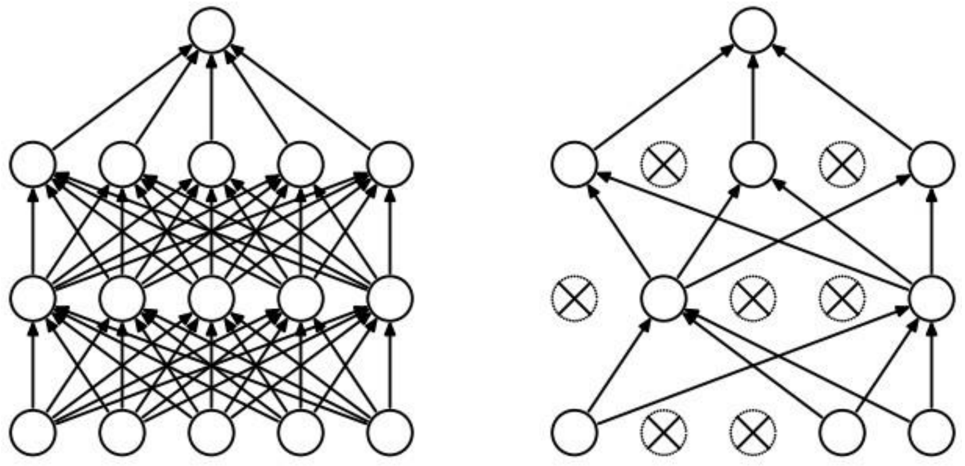
Online augmentation : This method is preferred for **larger datasets**, as you can't afford the explosive increase in size. Instead, you would perform transformations on the mini-batches that you would feed to your model. Some machine learning frameworks have support for online augmentation, which can be accelerated on the GPU.



Dropout

- Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.

- During training, some number of layer outputs are randomly ignored or "*dropped out*." By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections



(a) Standard Neural Net (b) After applying dropout.

- Training Phase: For each hidden layer, for each training sample, for each iteration, a random fraction, *p*, of nodes will be dropped out.

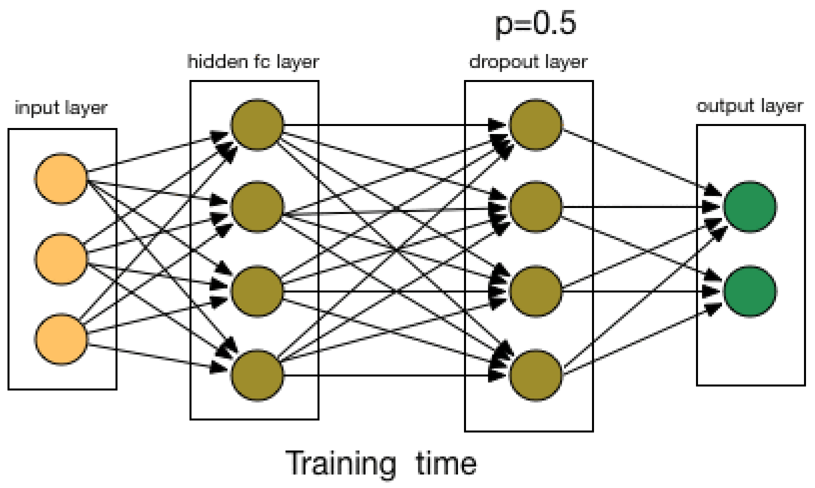
- The weights of the network will be larger than normal because of dropout. Therefore, before finalizing the network, the weights are first scaled by the chosen dropout rate. The network can then be used as per normal to make predictions.

- The rescaling of the weights can be performed at training time instead, after each weight update at the end of the mini-batch. This is sometimes called "*inverse dropout*" and does not require any modification of weights during training. Both the Keras and PyTorch deep learning libraries implement dropout in this way.

- Dropout works well in practice, perhaps replacing the need for weight regularization (e.g. weight decay) and activation regularization (e.g. representation sparsity).

- Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

- Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.


p=0.5
hidden fc layer
dropout layer
input layer
output layer
Training time