Module 2

What is Regularization?

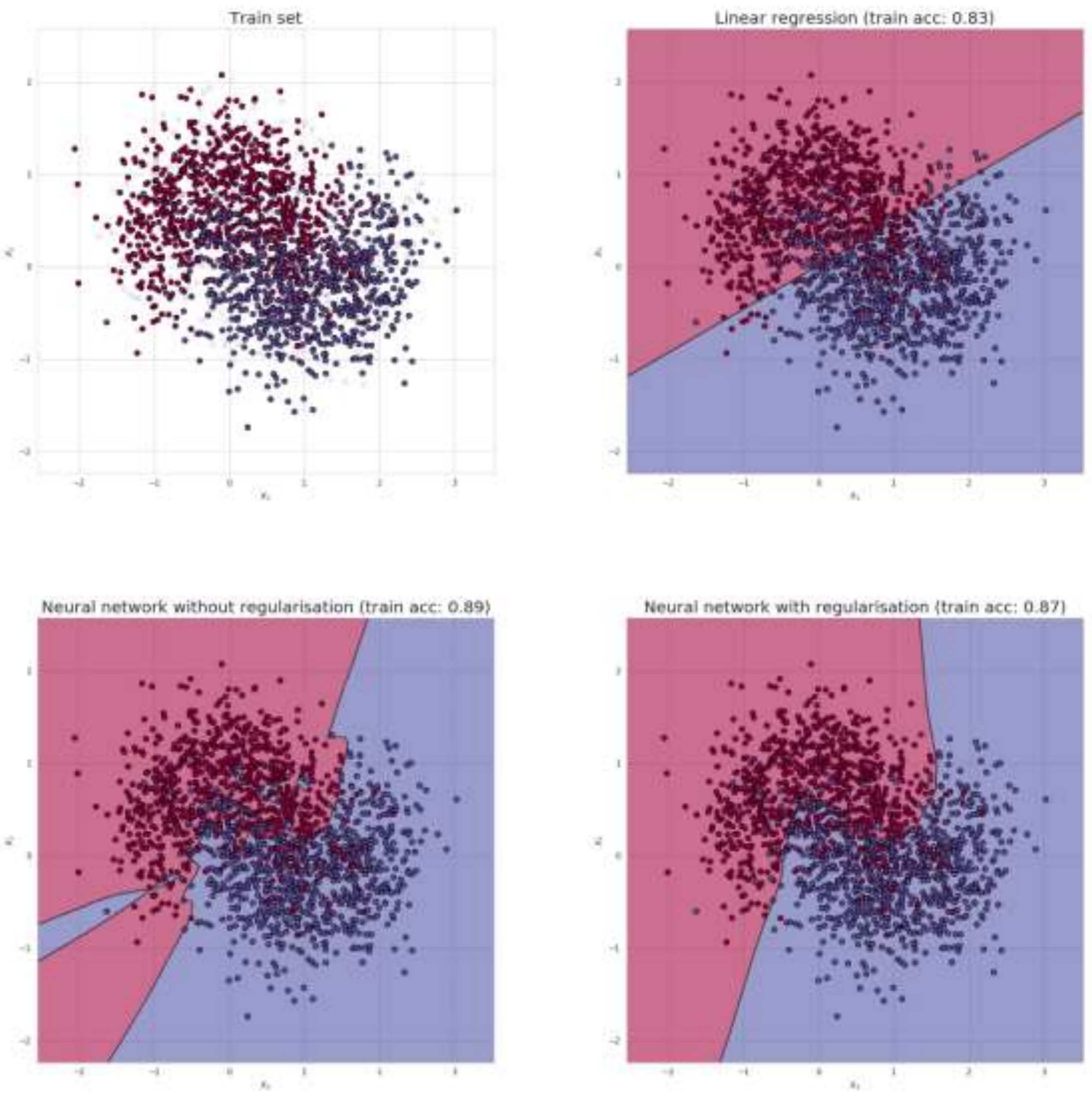
Regularization may be defined as any modification or change in the learning algorithm that helps reduce its error over a test dataset, commonly known as generalization error but not on the supplied or training dataset.

In learning algorithms, there are many variants of regularization techniques, each of which tries to cater to different challenges. These can be listed down straightforwardly based on the kind of challenge the technique is trying to deal with:

1. Some try to put extra constraints on the learning of an ML model, like adding restrictions on the range/type of parameter values.
2. Some add more terms in the objective or cost function, like a soft constraint on the parameter values. More often than not, a careful selection of the right constraints and penalties in the cost function contributes to a massive boost in the model's performance, specifically on the test dataset.
3. These extra terms can also be encoded based on some prior information that closely relates to the dataset or the problem statement.
4. One of the most commonly used regularization techniques is creating ensemble models, which take into account the collective decision of multiple models, each trained with different samples of data.

Module 2

The main aim of regularization is to reduce the over-complexity of the machine learning models and help the model learn a simpler function to promote generalization.

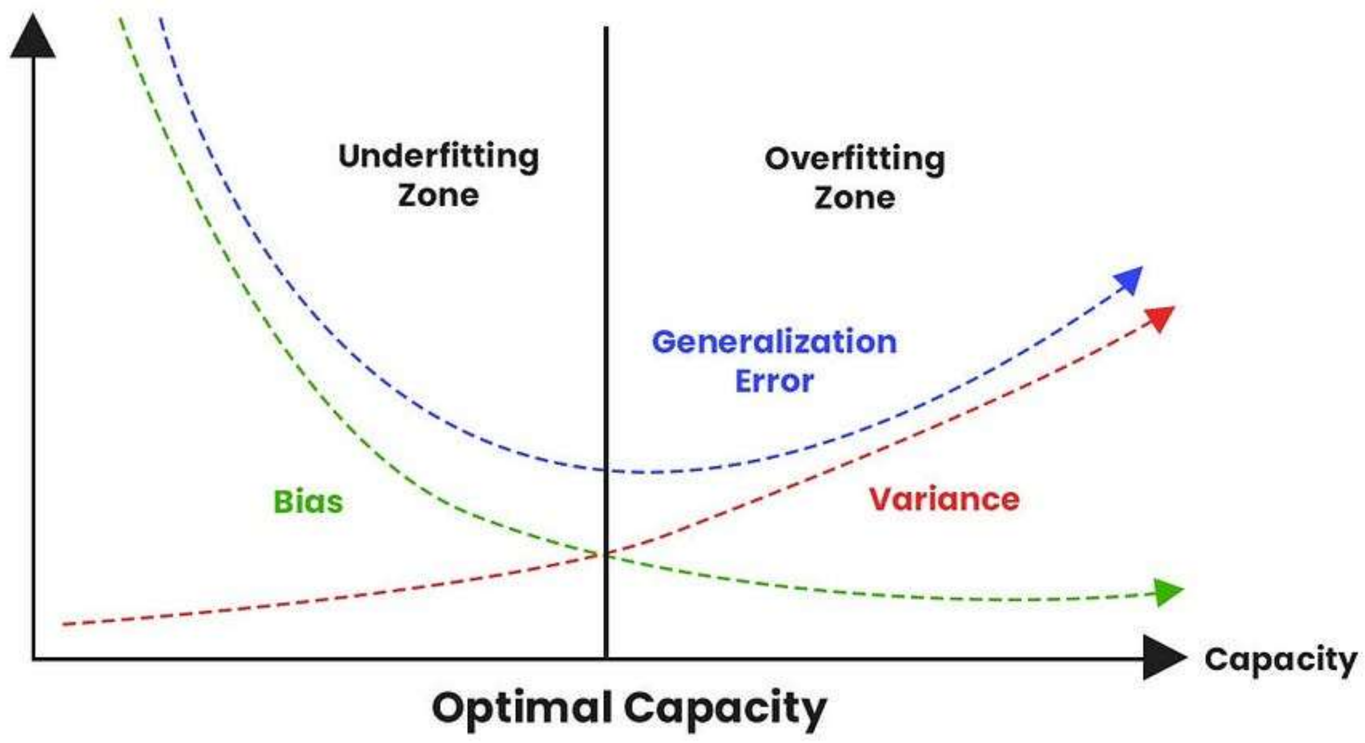


Impact of Regularization in Machine learning

Regularization in Deep Learning:

In the context of deep learning models, most regularization strategies revolve around regularizing estimators. So now the question arises what does regularizing an estimator means?

Bias vs variance tradeoff graph here sheds a bit more light on the nuances of this topic and demarcation:



Bias vs Variance tradeoff graph

Regularization of an estimator works by trading increased bias for reduced variance. ***An effective regularize will be the one that makes the best trade between bias and variance, and the end-***

Module 2

product of the tradeoff should be a significant reduction in variance at minimum expense to bias. In simpler terms, this would mean low variance without immensely increasing the bias value.

We consider two scenarios:

1. The true data-generating process/function: F1, which created the dataset
2. Creating a generating process/function: F2 that mimics F1 but also explores other possible generating scenarios/functions

The work of regularization techniques is to help take our model from F2 to F1 without overly complicating F2. Deep learning algorithms are mostly used in more complicated domains like images, audio, text sequences or simulating complex decision making tasks. **The True data-generation process: F1 is almost impossible to be correctly mapped, hence with regularization, we aim to bring our model with F2 function as close as possible to the original F1 function.** The following analogy helps understand it more clearly:

Fitting F2, our ML model, onto F1, our true data generation process is almost like fitting a square-shaped toy in a round hole by closed approximations.

In practical deep learning training and scenarios, we mostly find that the best fitting model (in the sense of least generalization error) is often a large model that has been regularized appropriately.

We will now dive deep to study one type of regularization technique that helps to create a large but deeply regularized model using parameter-based penalties.

Parameter Norm Penalties:

Under this kind of regularization technique, the capacity of the models like neural networks, linear or logistic regression is limited by adding a parameter norm penalty Ω(θ) to the objective function J.

The equation can be represented as the following:

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)$$

where α lies within [0, ∞) is a hyperparameter that weights the relative contribution of a norm penalty term, Ω, pertinent to the standard objective function J.

Setting α to 0 results in no regularization while larger values correspond to a greater regularization.

This type of regularization penalizes only the weights of the affine transformation at each layer of the network which leaves the biases unregularized. This is done with the notion in mind that it typically requires lesser data to fit the biases than the weights.

For deep learning, it sometimes feels desirable to use a separate parameter to induce the same affect.

L1 Parameter Regularization:

L1 regularization is a method of doing regularization. It tends to be more specific than gradient descent, but it is still a gradient descent optimization problem.

Formula and high level meaning over here:

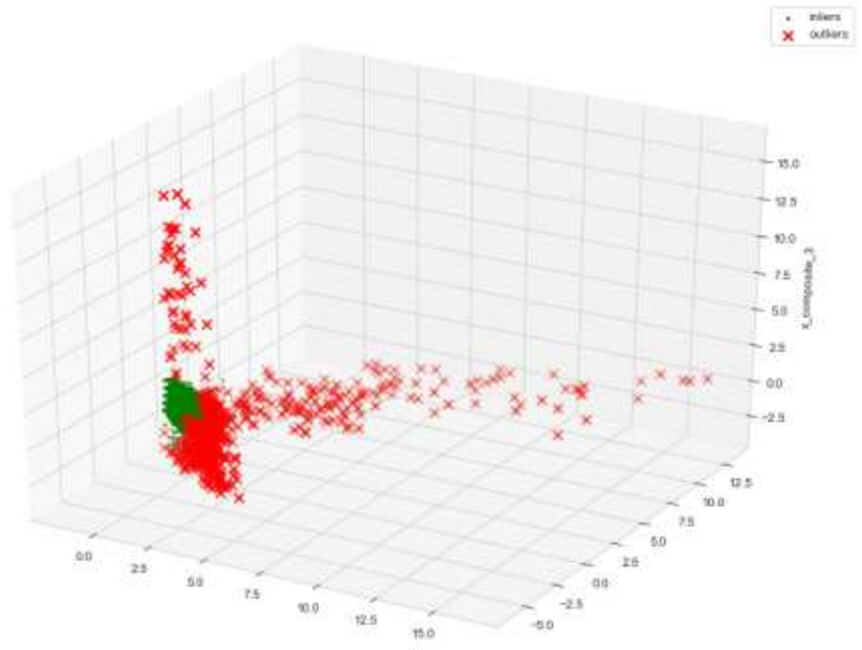
Module 2

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Lasso Regression (**Least Absolute Shrinkage and Selection Operator**) adds **"Absolute value of magnitude"** of coefficient, as penalty term to the loss function.

Lasso shrinks the less important feature's coefficient to zero; thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

The L1 regularizer basically looks for the parameter vectors that minimize the norm of the parameter vector (the length of the vector). This is essentially the problem of how to optimize the parameters of a single neuron, a single layer neural network in general, and a single layer feed-forward neural network in particular.



A good way of conceptualizing about it is that it is a method of maximizing the area of the parameter hyperspace that the true parameter vector is within. To do this, it finds the "sharpest" edge, one that is as close to the parameter vector as possible. Key points that should be noted for L1 regularization:

1. **L1 regularization is that it is easy to implement and can be trained as a one-shot thing,** meaning that once it is trained you are done with it and can just use the parameter vector and weights.

2. **L1 regularization is robust in dealing with outliers.** It creates sparsity in the solution (most of the coefficients of the solution are zero), which means the less important features or noise terms will be zero. It makes L1 regularization robust to outliers.

To understand the above mentioned point, let us go through the following example and try to understand what it means when an algorithm is said to be sensitive to outliers

1. *For instance we are trying to classify images of various birds of different species and have a neural network with a few hundred parameters.*

2. *We find a sample of birds of one species, which we have no reason to believe are of any different species from all the others.*

3. *We add this image to the training set and try to train the neural network. This is like throwing an outlier into the mix of all the others. By looking at the edge of the hyperspace where the hyperplane is closest to, we pick up on this outlier, but by the time we've got to the hyperplane it's quite far from the plane and is hence an outlier.*

4. *The solution in such cases is to perform iterative dropout. L1 regularization is a one-shot solution, but in the end we are going to have to make some kind of hard decision on where to cut off the edges of the hyperspace.*

5. *Iterative dropout is a method of deciding exactly where to cut off. It is a little more expensive in terms of training time, but in the end it might give us an easier decision about how far the hyperspace edges are.*

Along with shrinking coefficients, the lasso performs **feature selection**, as well. (*Remember the 'selection' in the lasso full-form?*) Because some of the coefficients become exactly zero, which is equivalent to the particular feature being excluded from the model.

L2 Parameter Regularization:

The Regression model that uses L2 regularization is called Ridge Regression.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \boxed{\lambda \sum_{j=1}^{n} \theta_j^2}\right]$$

$$\min_{\theta} J(\theta)$$

Formula for Ridge Regression

Regularization adds the penalty as model complexity increases. The regularization parameter (lambda) penalizes all the parameters except intercept so that the model generalizes the data and won't overfit. Ridge regression adds **"squared magnitude of the coefficient"** as penalty term to the loss function. Here the box part in the above image represents the L2 regularization element/term.

Module 2

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Lambda is a hyperparameter.

If lambda is zero, then it is equivalent to OLS.

Ordinary Least Square or OLS, is a stats model which also helps us in identifying more significant features that can have a heavy influence on the output.

But if lambda is very large, then it will add too much weight, and it will lead to under-fitting. Important points to be considered about L2 can be listed below:

1. **Ridge regularization forces the weights to be small but does not make them zero and does not give the sparse solution**.
2. **Ridge is not robust to outliers** as square terms blow up the error differences of the outliers, and the regularization term tries to fix it by penalizing the weights.
3. Ridge regression performs better when all the input features influence the output, and all with **weights are of roughly equal size**.
4. **L2 regularization can learn complex data patterns**

Module 2

Differences, Usage and Importance:

It is important to understand the demarcation between both these methods. In comparison to L2 regularization, L1 regularization results in a solution that is more sparse.

Sparsity in this context refers to the fact that some parameters have an optimal value of zero. The sparsity of L1 regularization is a qualitatively different behavior than arises with L2 regularization

The sparsity feature used in L1 regularization has been used extensively as a feature selection mechanism in machine learning. Feature selection is a mechanism which inherently simplifies a machine learning problem by efficiently choosing which subset of the available features should be used of the model.

Lasso integrates an L1 penalty with a linear model and a least-squares cost function. The L1 penalty causes a subset of the weights to becomes zero, which is safe to suggest that the corresponding features associated with the respective weights, may easily be discarded.

Regularization as Bayesian inference?

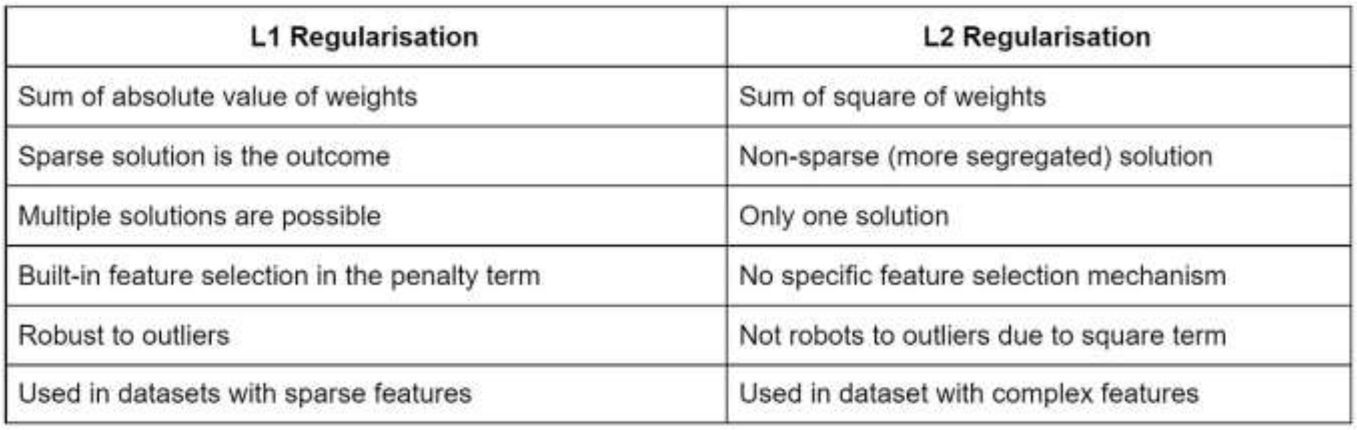
Many regularization techniques can be interpreted as MAP Bayesian inferences.

1. L2 in particular is almost equivalent to MAP Bayesian inference with a Gaussian prior on the weights.
2. In L1 regularization, the penalty term used to penalize the cost function can be compared to the log-prior term that is maximized by MAP Bayesian inference when the prior is an isotropic Laplace Distribution over the real number dataset

Module 2

Summary table

The entire post can also be summarized into small bullet points which might be useful during an interview preparation or to skim through the content and just find the right part. Hope this helps:



L1 Regularisation	L2 Regularisation
Sum of absolute value of weights	Sum of square of weights
Sparse solution is the outcome	Non-sparse (more segregated) solution
Multiple solutions are possible	Only one solution
Built-in feature selection in the penalty term	No specific feature selection mechanism
Robust to outliers	Not robots to outliers due to square term
Used in datasets with sparse features	Used in dataset with complex features

Difference between the two Regularization techniques