Megha Mishra Follow
May 27, 2018 · 11 min read · Listen

“A person standing on top of a ladder in the clouds.” by Samuel Zeller

REGULARIZATION: An important concept in Machine Learning

Hello reader,

This blogpost will deal with the profound understanding of the regularization techniques. I have tried my best to incorporate all the Why’s and How’s.

Regularization is one of the basic and most important concept in the world of Machine Learning. I have covered the entire concept in two parts.

Part 1 deals with the theory regarding why the regularization came into picture and why we need it?

Part 2 will explain the part of what is regularization and some proofs related to it.

So, let’s begin.

PART 1

what the above definition means, let's get into the details.

While thinking about ways of how I can pen it down, I came across this book "Pattern Recognition and Machine Learning by Christopher M. Bishop". The examples and explanations given in the book were very accurate and understandable. So, I have used the examples of that book here.

Consider the training dataset comprising of independent variables X=(x1,x2....xn) and the corresponding target variables t=(t1,t2,...tn). X are random variables lying uniformly between [0,1]. The target dataset 't' is obtained by substituting the value of X into the function sin(2πx) and then adding some Gaussian noise into it.

Note*: Gaussian noise are the deviations created in the target variables from the actual obtained output value, which follows the Gaussian distribution. This is done to represent the scenario of any real world dataset, as no data is perfect without any noise component*.

Now, our goal is to find patterns in this underlying dataset and generalize it to predict the corresponding target value for some new values of 'x'. The problem here is, our target dataset is inflicted with some random noise. So, it will be difficult to find the inlying function sin(2πx) in the training data. So, how do we solve it?

Let's try fitting a polynomial on the given data.

$$Y(x,w) = w0 + w1x + w2x^2 + \cdots .wnx^n. \qquad \text{- (Eq 1.1)}$$

It should be noted that the given polynomial function is a non-linear function of 'x' but a linear function of 'w'. We train our data on this function to determine the values of w that will make the function to minimize the error in predicting target values.

The error function used in this case is mean squared error.

$$E(w) = \left(\frac{1}{2}\right) * \Sigma(y(w,x) - t)^2 \qquad \text{- (Eq 1.2)}$$

In order to minimize the error, calculus is used. The derivative of E(w) is equated with 0 to get the value of w which will result at the minimum value of error function. E(w) is a quadratic equation, but it's derivative will be a linear equation and hence will result in only a single value of w. Let that be denoted by w*.

So now, we will get the correct value of w but the issue is what degree of polynomial (given in Eq 1.1) to choose? All degree of polynomials can be used to fit on the training data, but how to decide the best choice with minimum complexity?

Furthermore, if we see the Taylor expansion of sine series then any higher order polynomial can be used to determine the correct value of the function.

Just for reference, Taylor expansion of sin(x) is

$$(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - .....)$$

To represent the above mentioned point graphically, present below is the graph of target variable vs input variable i.e y= sin(2πx). Blue circles are the training data-points. Green curve is the expected polynomial function which fits the training data set and red curve is the polynomial function of various degrees (given by variable M) that are trained to fit the data set.

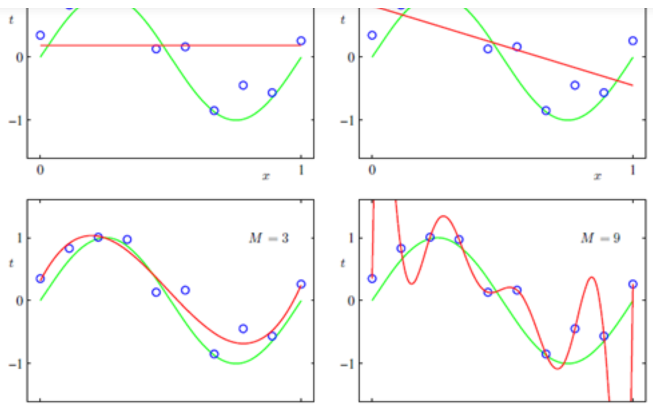

Fig 1

Fig 1 shows the fitting of polynomials of various order on the data points. We see that polynomial of degree 9 is able to make accurate predictions for all the datapoints in the training set.

But wait, the function obtained for the polynomial of degree 9 looks nothing like our chosen function sin(2πx). So what has happened here?

The function with degree 3 is able to fit our training data to a great extent. Also, it is a subset of polynomials of high order. So the function/polynomial of order 9 should also be able to fit the underlying pattern, (which is clearly not the case). This is called overfitting.

In the above case the function obtained for the polynomial of degree 9 will be something like this:

$$y = 0.35 + 232.37 * x - 5321.83 * x^2 + 48568.31 * x^3 - 231639.30 * x^4 + 640042.26 * x^5 - 1061800.52 * x^6 + 1042400.18 * x^7 - 557682.99 * x^8 + 1252201.43 * x^9 \qquad \text{- (Eq 1.3)}$$

Looks very complex and dangerous isn't it?

The function has trained itself to get the correct target values for all the noise induced data points and thus has failed to predict the correct pattern. This function may give zero error for training set but will give huge errors in predicting the correct target values for test dataset.

To avoid this condition regularization is used. Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values. This technique of keeping a check or reducing the value of error coefficients are called shrinkage methods or weight decay in case of neural networks.

Overfitting can also be controlled by increasing the size of training dataset.

EDIT: Here increasing the size of the dataset to avoid overfitting refers to increasing the number of observations (or rows) and not the number of features (or columns). Adding columns may lead to increase in the complexity of problem and therefore may result in more poor performance. Thanks to Sean McClure for pointing it out :)

1.2 Probablistic view :

Here we will look at the same problem but with a different perspective. Once again let us assume the training data to be 'X' and corresponding target variables to be 't'. Now we have to find the correct value of target variable given any new test data x. or we can say that we have to find the highest probability of t given x i.e p(t|x), parameterised over w.

Let us make a small assumption for this purpose. Let's assume that for a given value of 'x', the corresponding value of 't' follows a Gaussian distribution with mean equal to y(x,w) (note that y(x,w) is our eq 1.1.)

$$p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1}) \qquad - \text{(Eq 2.1)}$$

$$N(x|\mu,\sigma^2) = \frac{1}{2\pi\sigma^2}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \qquad -(\text{Eq } 2.2)$$

Here β is the precision parameter i.e inverse of variance

Why we have made this assumption? Don't worry, I have mentioned it in the very next section.

So going back to our probability equation, here parameters w and β are unknown. We use our training data to determine the value of w and β. All the training data points are independent of each other. Therefore we can say that,

$$p(\boldsymbol{t}|\boldsymbol{x}, w, \beta) = \prod_{n=1}^{N} N(t_n|y(x_n, \boldsymbol{w}), \beta^{-1}) \qquad -(\text{Eq } 2.3)$$

(Independent probabilities can be multiplied)

Now we will get the correct values of w and β by maximizing the given equation. For convenience we will take the logarithm of the above equation as it doesn't affect the values.

$$\ln p(\boldsymbol{t}|\boldsymbol{x}, w, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) \qquad -(\text{Eq } 2.4)$$

Here when differentiating w.r.t 'w', last 2 terms can be ignored as it contains no 'w' terms. Also β in the prefix of first term only performs the task of scaling the term and hence has no significant effect in the final differential output Therefore, if we ignore β, we will end up getting the same mean squared error formula, which gives us the value of 'w' as in (eq 1.2). Isn't this magic?

Similarly we will find the value of β as well, by differentiating it w.r.t β. Now having known the values of w and β, we can now use our final equation to predict the values of test data.

1.3 Proof for Assumption:

To understand this you should have a little knowledge about expectation of a function.

Expectation of a function is it's average value under a probability distribution. For continuous variables it is defined as

$$E[f] = \int p(x) * f(x)dx$$

Expectation for continuous variables

Consider this situation, where we want to categorize patient as having cancer or not based on some observed variable. If a person not having a cancer is categorized as a cancer patient then some loss will be incurred in our model, but conversely if some patient having cancer is predicted as a healthy patient then the loss incurred will be several times higher then the previous case.

Let Lij denote the loss when i is falsely predicted as j. The average expected loss will be given by:

$$E[L] = \iint L(t, y(x))p(x,t)dxdt \qquad -(\text{Eq } 3.1)$$

Where $L(t, y(x)) = (y(x) - t)^2$ and $p(x,t)$ is the Gaussian distribution given in Eq (2.1)

We wish to minimize the loss as much as possible. We choose the value of y(x), such that E(L) is minimum. Thus applying product rule [p(x,t)=p(t|x) * p(x)] and differentiating it w.r.t y(x) we get,

$$y(x) = \frac{\int tp(x,t)dt}{p(x)} = \int tp(t|x)dt \;= E_t[t|x] \quad -(\text{Eq } 3.2)$$

Does the above equation looks familiar? Yes it is our assumption made. The term Et[t|x] is the conditional average of 't' conditioned

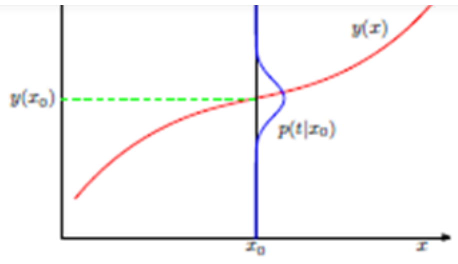

Fig2

The given above figure helps to get a clear understanding of it. We can see that follows a Gaussian distribution with minimum value of loss function for the regression function y(x) obtaine 1.6K 17 ••• lue

PART 2

2.1 Let's Regularize:

Finally we are here! In this section I am going to tell you about the types of regularization and how are they obtained.

Let's go back to the point where we started. Our initial condition on the dataset was have a target variable t consisting of actual value with some added Gaussian noise.

$$t = y(x,w) + \text{Gaussian noise.}$$

According to our probabilistic view point, we can write from equation 2.3

$$p(\mathbf{t}|\mathbf{x}, w, \beta) = \prod_{n=1}^{N} N(t_n|y(x_n, \mathbf{w}), \beta^{-1})$$

Where β is our inverse of variance or precision parameter. Note that the above equation is only valid if our data points are drawn independently from the distribution (eq 2.2). The sum of square error for the above function is defined by:

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\varphi(x_n)\}^2 \qquad -\text{(Eq 4.1)}$$

Don't get confused by seeing " wT*phi(xn)" . It's nothing but our y(x,w). It is a more generalized form of the original eq (1.2). phi(xn) are known as basis functions. For eq (1.1) " phi(xn)" was equal to "x^n".

Now we have seen previously that in many cases using this error function often leads to overfitting. Thus regularization term was introduced. After introducing the regularization co-efficient, our overall cost function becomes

$$ED(\mathbf{w}) + \lambda\, EW(\mathbf{w}) \qquad -\text{(Eq 4.2)}$$

Where, λ controls the relative importance of data dependent error w.r.t regularization error term. A generalized form of regularization term is given below:

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\varphi(x_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q \qquad -\text{(Eq 4.3)}$$

Where, if q=1, then it is termed as lasso regression or L1 regularization, and if q=2, then it is called ridge regression or L2 regularization. If both the terms L1 regularization and L2 regularization are introduced simultaneously in our cost function, then it is

$$\sum_{j=1}^{M} |w_j|^q \leq \eta$$

For appropriate value of constant η.

Now how do these extra regularization term help us to keep a check on the co-efficient terms. Lets understand that.

Consider the case of lasso regularization. Suppose we have an equation y= w1 +w2x. We have considered the equation with only two parameters because it will be easy to visualize in the contour plots. I got a headache when I tried visualizing a multidimensional graph. Not an easy task. Even google was not helpful here :P

Note: *Just for a quick review, contour plots are 2D representation of and given 3D graphs. For the cost function or error function the 3D graph is plotted between the coefficient values w1 and w2 and the corresponding error function.*

i.e X-axis =w1, Y-axis= w2 and Z-axis= J(w1,w2), where J(w1,w2), is the cost function

$$J(w1, w2) = \left(\frac{1}{2m}\right) * \sum_{i=1}^{m} (y(w,x) - t)^2 \quad \textit{where m is the number of training examples.}$$

When the contour plot is plotted for the above equation the x and y axis represents the independent variables (w1 and w2 in this case) and the cost function is plotted in a 2D view.

Now, returning back to our regularization.

The cost function for the lasso regression will be :

$$E_D + \lambda/2(|w1| + |w2|), \text{ where } |w1| + |w2| <= \eta \qquad -(\text{Eq } 4.4)$$

The graph for this equation will be a diamond figure as shown below,

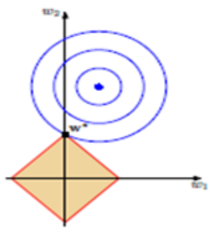

fig 3

Here, the blue circles represent the contours for the un-regularized error function (ED) and diamond shape contour is for L1 regularization term i.e (λ/2(|w1| + |w2|),. We can see in the graph that optimal value is obtained at the point where w1 term is zero i.e the basis function corresponding to w1 term will not affect the output. Here represented by w* where both the terms of cost function will take a common value of 'w' as required in the equation.

Hence we can say that for the proper value, λ the solution vector will be a sparse matrix (eg [0,w2]). So this is how the complexity of the equation (1.3) can be reduced. The solution matrix of **w** will have most of it's values as zero and the non-zero value will contain only the relevant and important information thus finding a general trend for the given dataset.

I hope that now you can comprehend regularization in a better way. You can use this to improve the accuracy of your models.

In my future blogs this concept will be used in explaining other important concepts of Machine Learning. If you have any other doubt related to the topic, leave a comment, I will try my best to clarify it.

Also, if you liked my explanation, do clap for it.

Get unlimited access Open in app