Forum - [Latest Machine Learning Specialization/Supervised ML: Regression and Classification topics - DeepLearning.AI](https://community.deeplearning.ai/c/course-q-a/machine-learning-specialization/mls-course-1/274)

Week 1 -

The objective of regression is to minimize the distance between the y predicted and the y actual.

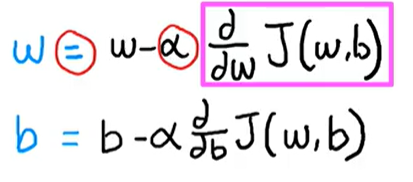
The distance is measured using the cost function. Cost function is J(w,b) = 1/2mSum of (y hat-y)^2

Find the values of w and b which minimize the cost function.

Gradient descent – values of J as a surface for different values of w and b.

Fastest way down the hill – choose starting point, then take the path where slope is steepest. Keep going down in direction of steepest descent.

We can have multiple lowest points – they are all the local minima. Depends on the starting point.



Alpha – learning rate- the size of steps taken in gradient descent. If too small then too long, if too big then fails to coverge.

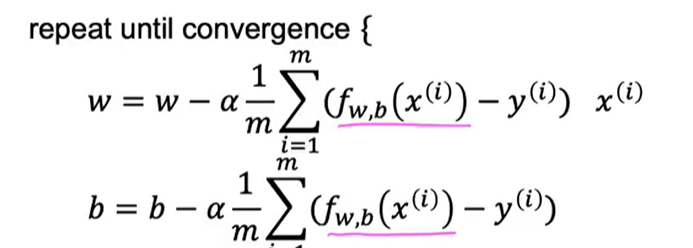
Derivative term – also determines the size of the step to take. It is the slope of the cost function. If slope is positive then w new < w old which is the correct direction towards min of J(w,b).

Local minima – the point where w and b don’t change.

Simultaneous update – cannot use updated w in calculation of updated b, that has to use old w.

Min of J(w,b) will be where derivative =0 or close. So further gradient descent keeps the solution at the same values of w and b.

After solving squared error cost function the derivative we get (univariate regression)-



The squared error cost function only has 1 minimum not multiple local ones.

Batch gradient descent – uses all values of x, y from dataset.

Week 2 –

Vectorization uses parallel computing, so it is faster and less lines of code.

A dot product in NumPy is implemented in the computer hardware with vectorization. The computer can get all values of the vectors w and x, and in a single step, it multiplies each pair of w and x with each other all at the same time in parallel.

Computers do simultaneous calcs and assignments.

In multiple variable regression we have n vectors w(i) undergoing simultaneous change until convergence rather than just one w.

Alternate to gradient descent (only for linear regression) – Normal equation, solve for w and b without iterations. May be on the backend of some libraries.

Disadvantages – doesn’t generalize for other learning algorithms, slow

Feature scaling -

If parameter values are large and range is big, the corresponding coefficient should be chosen small and vice versa.

A small change in w(1) will have a very big impact on cost if x(1) is huge nos

A small change in w(2) will have a very little impact on cost if x(2) is small nos

The contour plot will be long and skinny, and the gradient descent will take a long time

Possible solution – transforming the features, rescale to 0 to 1 range- divide by max, x- mean/(max - min), z score

Learning curve - Plot cost function for number of iterations of gradient descent

If cost function goes up in the middle – learning rate is too big. Test – keep alpha very small, now if the cost doesn’t decrease it means something is off.

Ideally start alpha=0.001 and keep going 3x until learning curve looks good.

Feature engineering – creating a new feature from existing ones, polynomial – square cube, square root

Week 3 –

Classification with 2 categories – binary classification. Dividing line- decision boundary.

Linear regression model if y pred < 0.5 then 0, else 1. Best fit lines will shift to a great extent for new data in training.

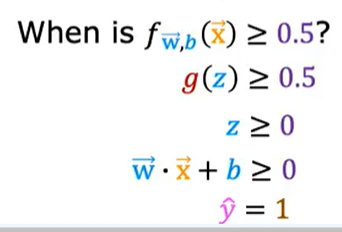
Logistic regression – s shaped curve with 0 or 1 as output.

Sigmoid function g(z) = 1/(1+e^-z) as z -> inf, g(z) -> 1, as z -> -inf, g(z) -> 0, g(0) = 0.5

Fw,b(x) = g(z) = g(wx+b) = 1/(1+e^-( wx+b)) This gives the probability that class =1, P(y=1)



When does the model predict y= 1 -



Decision boundary is the line wx +b = 0 or fw,b(x) = 0. It need not be linear, can be polynomial etc.

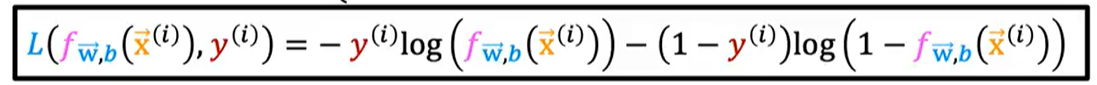
Just solve this relationship to get the values of w and b.

Using the gradient descent cost function for logistic regression gives a non-convex cost function J(w,b). Loss function – if y=1 then -log(f(x)) and if y=0 then -log(1-f(x)).

loss is -log(y hat) if y=1 and -log(1- y hat) if y=0

|  |  |
| --- | --- |
|  | A graph of a function  AI-generated content may be incorrect. |

Cost = 1/m(Sum of loss for all training examples) where loss function is as given below –



This cost function is from maximum likelihood estimation

The updates to the values of w and b for gradient descent end up being the same function as the one used for linear regression (the difference is that f is a different function)

Equation -

A math equation with black text

AI-generated content may be incorrect.

Overfitting -

Underfit – algorithm does not fit well, too general, high bias

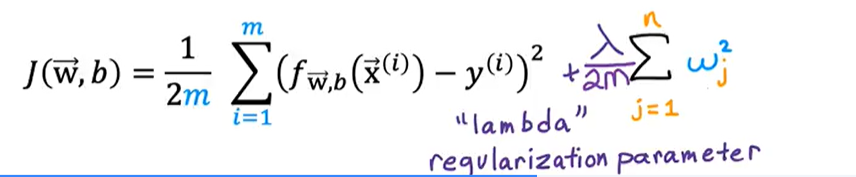
Overfit – fits too well, cannot be generalized, high variance

Solution – get more training data, take selected features but useful features could be lost

Regularization – eliminate features which are in higher powers – set parameter to 0, or very small number. You keep all the features.

How to make the parameter small. To reduce w(i), add it to the cost function with a huge multiplier. Example – cost + 1000\*w(i). Then the solution will come as the making w(i) very small to minimize cost.

Since we don’t know which to penalize, initially we penalize all by taking lamba/2m\*sum(w^2)



Don’t regularize b (some do).

Value of lamda can be chosen through trials. It is a tradeoff between fitting the data and keeping w(j) small.

Change in gradient descent only on dJ(w,b)/dw = 1/m \* Sum( y(i) pred - y(i))\*x(i) +1/m\*w(j)

A math equations on a white background

AI-generated content may be incorrect.

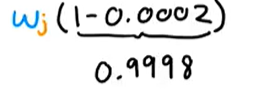
Simply it to -

A math equations and symbols

AI-generated content may be incorrect.

As alpha is a small number, lambda is a positive number and m is the size of training set – gives small +ve number so shrinking w(j) a little bit.

A whiteboard with numbers and symbols

AI-generated content may be incorrect. 

In logistic regression, cost function gets same update – add lamba/2m\*sum(w^2)

Gradient descent – same equation as above.