3, Batch Normalization (Explained)

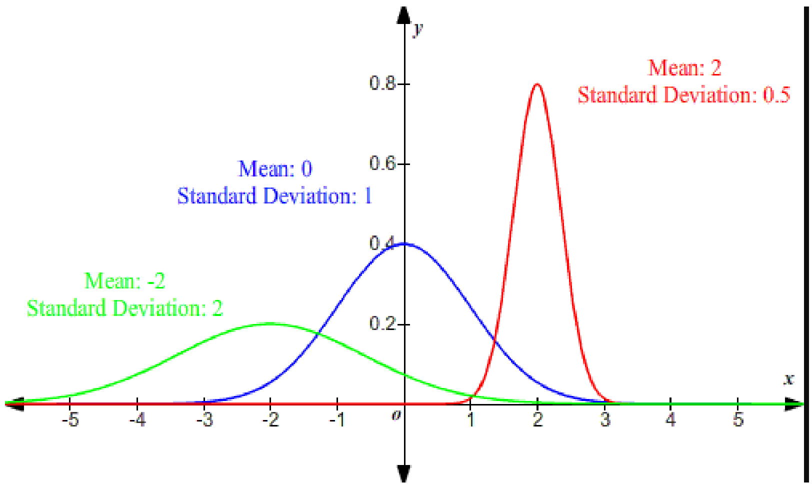
🔖Notes

▼ why do batch normalization?

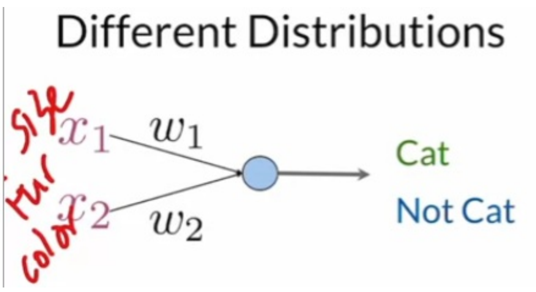
> GANs are often quite fragile when they learn to because they aren't as straightforward as a classifier. And sometimes the skills of the generator and discriminator aren't as aligned as they could be. For these reasons, every trick that speeds up in stabilizes training is crucial for these models, and batch normalization has proven very effective to that end.

>>batch normalization can help to stabilize the training

▼ Distribution recap

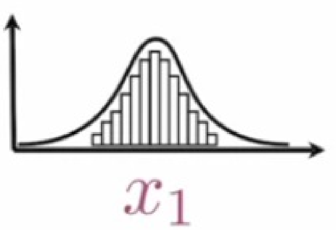


▼ CASE STUDY: Super simple NN with 2 input variables that have different distribution



▼ x1x_1x1 >> size

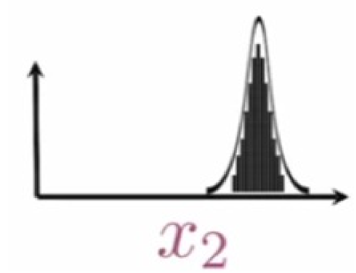
▼ distribution of x1x_1x1



>> dataset is normally distributed

>For example, with very few extremely small or extremely large examples. (x1x_1x1 is size~)

▼ x2x_2x2>> fur color



>>distribution with higher mean and lower standard deviation

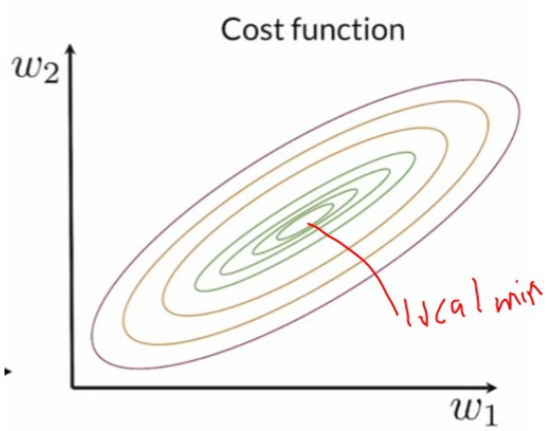
>>example, this dataset has daker fur color (x2x_2x2is fur color)

▼ the different in distribution of these 2 input variables, so?

> the result of having these different distributions like this impacts away your neural network learns.(the cost function distribution will be elongated)

bulat >> hidden layer

▼ local minimum (with case study)



the shape of this cost function is based on the dataset, if the dataset distribution is mixed, then the cost function will be elongated like this, then this situation will make NN hard to train. But if the dataset is normalized (distribution is nice, then the training is easier)

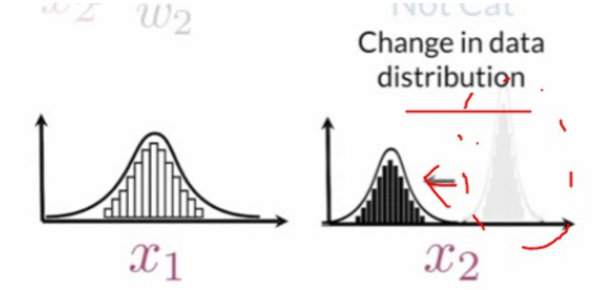
w1w_1w1 >>weight of x1x_1x1

w2w_2w2 >> weight of x2x_2x2

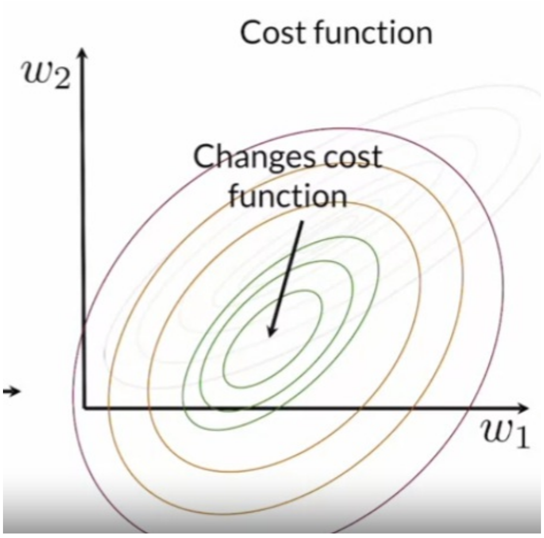
>> cost function >> when w1w_1w1 has how much, w2w_2w2 will has how much>> since the distribution of x1x_1x1 and x2x_2x2 is different, of course, the cost function betweent the weight of these of two input variables, will be elongated.

▼ Effect of having diff distribution of input variables, and now the state of distribution is even shifted lagi, How cost function will look like?

▼ data distribution is shifted



▼ the form of cost function will be changed too (so called internal covariate shift that happened in the hidden layer)



>> can see the form of cost function is more rounder now

>> and also the location of local minimum also moved

▼ internal covariate normalization? (the situation when data distribution shifted, form of cost function will changed more rounder, But, the label of your image is still not changed)

>> this happens pretty often between training and test sets where precautions haven't been taken on how the data distribution is shifted.

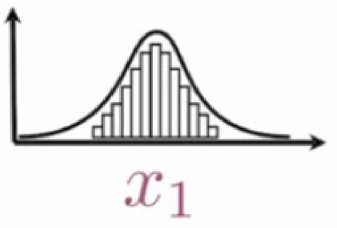
>> rounder>> means it is being normalized

> if new training or test data has, let's say, really **light for a color**, so the state is distribution kind of shifts or changes in some way, then **the form of the cost function could change too**. So it's showing that it's a little bit more round here now and the location of the minimum could also move. Even if the ground truth of what determines whether something's a cat or not stays exactly the same. That is the labels on your images of whether something is a **cat or not has not changed.** And this is known as covariate shift

▼ How normalization helps models (x1x_1x1 and x2x_2x2 is normalized)

▼ before

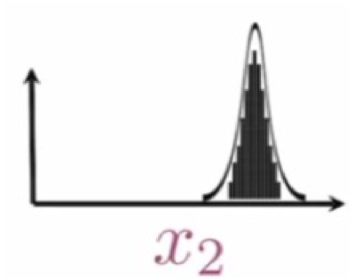
▼ distribution of x1x_1x1



>> dataset is normally distributed

>For example, with very few extremely small or extremely large examples. (x1x_1x1 is size~)

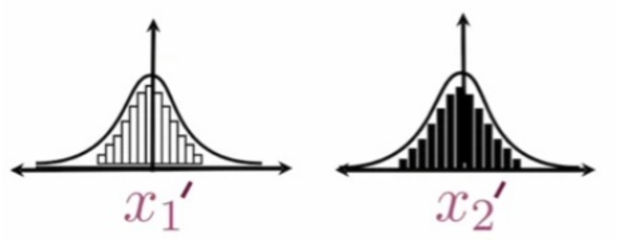
- ▼ distribution of x2x_2x2



>>distribution with higher mean and lower standard deviation

>>example, this dataset has daker fur color (x2x_2x2is fur color)

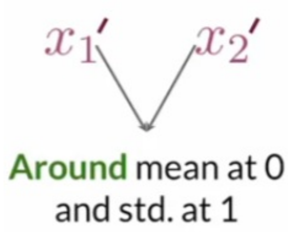
- ▼ after (being normalized)



x1′x_1^{'}x1′>> x1x_1 x1 after being normalized

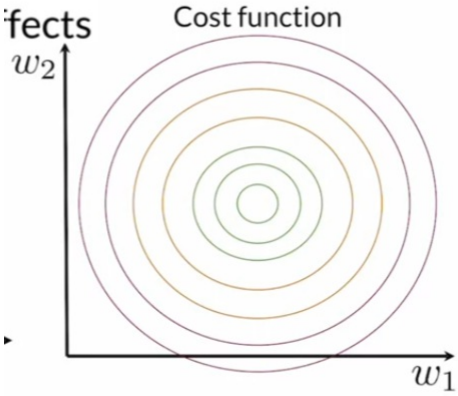
x2′x_2^{'}x2′ >>x2x_2x2 after being normalized

- ▼ normalized >> mean = 0, std = 1



> As you train each batch, you take the mean and standard deviation and you shift it to be around 0, and standard deviation of 1. And for the test data what you do is you can actually look at the statistics that were gathered overtime as you went through the training set and use those to center the test data to be closer to the training data.

- ▼ effect on cost function



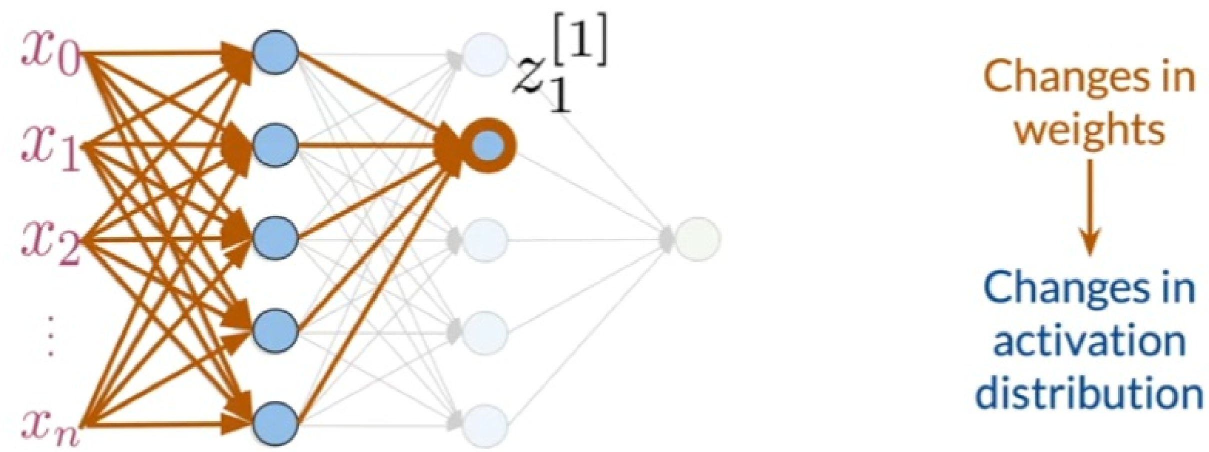
>> looks smoother and more balanced across these two dimensionns

>> training will be easier, and potentially much faster

>> using normalization, the effect of this covariate shift will be reduced significantly.(smoothing that cost function out in reducing the covariate shift)

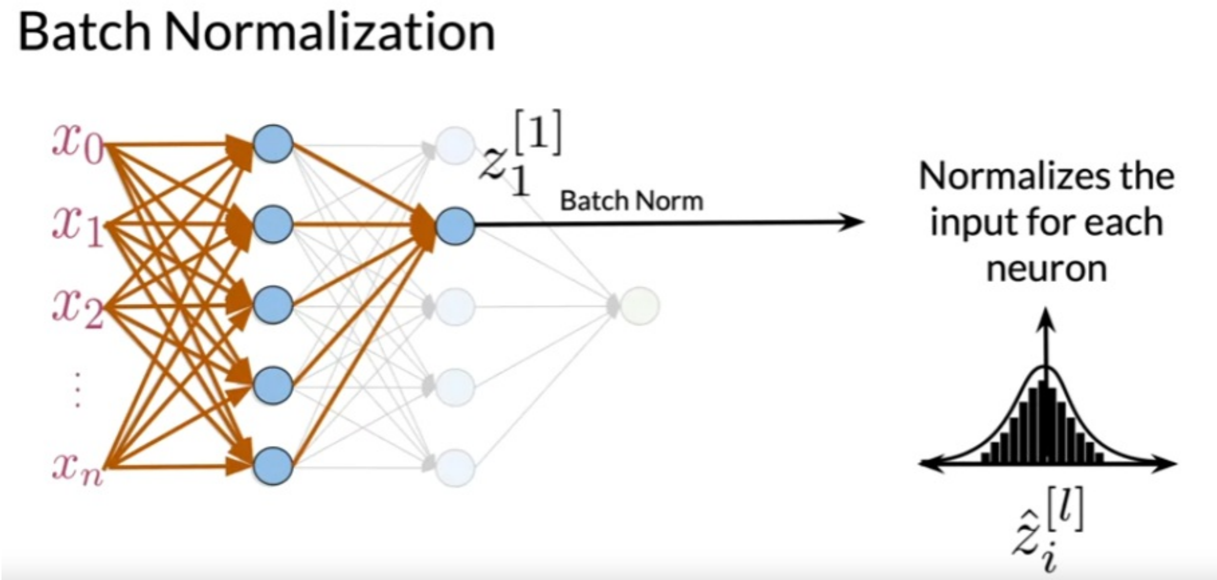
▼ batch normalization (normalize the internal node (those in hidden layer), instead of input node)

▼ problem that batch normalization try to solve (internal covariate shift)



> So take the **activation output of this first hidden layer of the neural network (z1[1]z_1^{[1]}z1[1])**and look at this node right here. When training the model, all the weights that affect the activation value are updated. So all of these weights are updated. And consequently, the distribution of values contained in that activation changes in our influence over this course of training. <u>**This makes the training process difficult**</u> due to the shifts similar to the changes you saw in the input variable distribution shifts like fur color. (the local minimal is located at lower, and the distribution is rounder)

▼ solution (batch normalization)



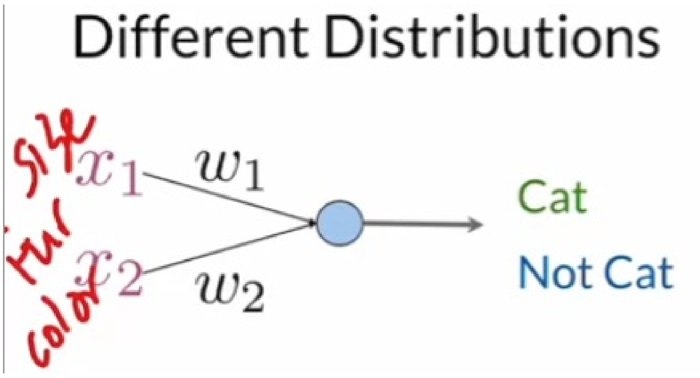
> Now batch normalization seeks to remedy the situation. And normalizes all these internal nodes based on **statistics calculated for each input batch**. And this is in order to **reduce the internal covariate shift**.

□Vocabs

▼ internal covariate normalization?

> if new training or test data has, let's say, really **light for a colo**r, so the state is distribution kind of shifts or changes in some way, then **the form of the cost function could change too**. So it's showing that it's a little bit more round here now and the location of the minimum could also move. Even if the ground truth of what determines whether something's a cat or not stays exactly the same. That is the labels on your images of whether something is a **cat or not has not changed.** And this is known as covariate shift

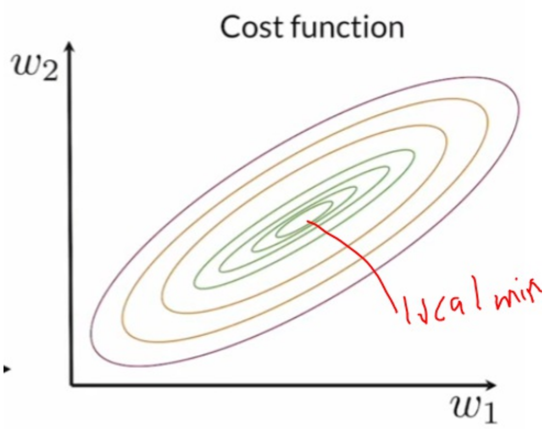
▼ single activation



> In a single activation that outputs whether this example based on these features is a cat or not a cat.

>> activation that has 2 output oni, yes, or not

▼ local minimum (with case study)



the shape of this cost function is based on the dataset, if the dataset distribution is mixed, then the cost function will be elongated like this, then this situation will make NN hard to train. But if the dataset is normalized (distribution is nice, then the training is easier)

w1w_1w1 >>weight of x1x_1x1

w2w_2w2 >> weight of x2x_2x2

>> cost function >> when w1w_1w1 has how much, w2w_2w2 will has how much>> since the distribution of x1x_1x1 and x2x_2x2 is different, of course, the cost function betweent the weight of these of two input variables, will be elongated.

batch stat?

▼ the task your modeling

> covariate shift shouldn't be a problem if you just make sure that the distribution of your data set is similar to the task your modeling. So, the test set is similar to your training site in terms of how it's distributed.

>> means, if ur test set and training dataset is similar, then no need do batch normalization also can, since internla covariate shift wont be a problem

QOTD

Summary

batch normalization smooths the cost function (more rounder and at center)

batch normalization reduce the internal covariate shift