Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. Join them; it only takes a minute:

Sign up

Here's how it works:

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

Why do we normalize images by subtracting the dataset's image mean and not the current image mean in deep learning?

There are some variations on how to normalize the images but most seem to use these two methods:

1. Subtract the mean per channel calculated over all images (e.g. VGG_ILSVRC_16_layers)
2. Subtract by pixel/channel calculated over all images (e.g. CNN_S, also see Caffe's reference network)

The natural approach would in my mind to normalize each image. An image taken in broad daylight will cause more neurons to fire than a night-time image and while it may inform us of the time we usually care about more interesting features present in the edges etc.

Pierre Sermanet refers in 3.3.3 that local contrast normalization that would be per-image based but I haven't come across this in any of the examples/tutorials that I've seen. I've also seen an interesting Quora question and Xiu-Shen Wei's post but they don't seem to support the two above approaches.

What exactly am I missing? Is this a color normalization issue or is there a paper that actually explain why so many use this approach?

deep-learning image-processing

asked May 8 '16 at 11:11

Max Gordon
3,076 3 24 40

I don't know the answer, but have you tried each of the method? Is their any difference in the performances? – user112758 May 8 '16 at 12:38

@user112758 - implementing them is a little painful (especially for the by-pixel) and my experience is that normalizing per image works fine but my data is not that representative. I'll try to experiment with the normalization but I'm curious to hear the motivation behind these (in my mind) strange normalization procedures. – Max Gordon May 8 '16 at 12:41

Ok, maybe you can ask this in the caffe Google group caffe GitHub issues. I guess there would be more experts on this topic. – user112758 May 8 '16 at 12:50

3 Answers

Subtracting the dataset mean serves to "center" the data. Additionally, you ideally would like to divide by the sttdev of that feature or pixel as well if you want to normalize each feature value to a z-score.

The reason we do both of those things is because in the process of training our network, we're going to be multiplying (weights) and adding to (biases) these initial inputs in order to cause activations that we then backpropogate with the gradients to train the model.

We'd like in this process for each feature to have a similar range so that our gradients don't go out of control (and that we only need one global learning rate multiplier).

Another way you can think about it is deep learning networks traditionally share many parameters - if you didn't scale your inputs in a way that resulted in similarly-ranged feature values (ie: over the whole dataset by subtracting mean) sharing wouldn't happen very easily because to one part of the image weight `w` is a lot and to another it's too small.

You will see in some CNN models that per-image whitening is used, which is more along the lines of your thinking.

answered Jun 28 '16 at 7:24

lollercoaster
801 7 12

1 Thank you for the answer. I'm familiar with the concept of centering the data and making the sure the range is similar in order to get stable gradients. The question is more of why we need to do this over the entire dataset and why this would help in contrast to per-image whitening? I would like a simple reference that shows in some way that this improves learning before I accept the answer. I know that batch normalization is an incredibly powerful technique but I don't see the connection to entire dataset normalization. – Max Gordon Jun 28 '16 at 18:50

If you accept batch normalization is good, then you're already there. The only reason you batch normalize is when you can't fit the full dataset in memory or you're distributing the training (often the same issue). That's why we have batches. – lollercoaster Jun 29 '16 at 0:05

I thought that batches are also the foundation for stochastic gradient descent. Even if I could fit everything into memory I want to update the parameters more frequently than after each epoch. – Max Gordon Jun 29 '16 at 6:23

3 They are. And you can update however frequently you want - the analytical implications are identical which is what's so nice and scalable about gradient descent. The reason that we use *stochastic* gradient descent (shuffling input order + batching) is to smooth out our hill climbing through gradient space. Given a single point we can't really be sure our update will push us in the direction of local maxima, however if you select enough points, this likelihood becomes higher (in expectation). – lollercoaster Jun 29 '16 at 17:41

Prior to batch normalization, mean subtraction per channel was used to center the data around zero mean for each channel (R, G, B). This typically helps the network to learn faster since gradients act uniformly for each channel. I suspect if you use batch normalization, the per channel mean subtraction pre-processing step is not really necessary since you are normalizing per mini-batch anyway.

answered Feb 20 '17 at 19:14

Sid M
51 1 1

I'll trust you on that – M090009 Apr 16 at 6:18

Per-image normalization is common and is even the only in-built function currently in Tensorflow (primarily due to being very easy to implement). It is used for the exact reason you mentioned (day VS night for the same image). However, if you imagine a more ideal scenario where lighting was controlled, then the relative differences between each image would be of great value in the algorithm, and we wouldn't want to wipe that out with per-image normalization (and would want to do normalization in the context of the entire training data set).

answered Mar 18 at 18:28

JPJ
354 1 9