Hey, \*Deep Learning From the Foundations Study Group (DLFFSG?)\*:  
  
Thanks to all who attended this morning's meetup which went quite well. The highlight was at the end, when @banacl (a.k.a. Prasanth) :pray: broke the back :dromedary\_camel: of the crazy convoluted code :jeremy: in the `RunningBatchNorm` :woman-running:class from the `07\_batchnorm.ipynb` notebook:notebook:!  
  
We were discussing how :question:to make sense of their formula for the \_effective momentum\_ `mom1` :thinking\_face:  
  
`mom1=1-(1-self.mom)/math.sqrt(bs-1)`,  
where  
`self.mom = 0.1`,  
and  
`bs = 512`  
are default settings for `momentum` and `batch size` respectively.  
  
I was at my wit's end after breaking my head :exploding\_head: for the last few days, trying to understand the precise mathematical reasoning behind that `mom1` formula .  
  
Prasanth's insight :telescope: was to look at what the formula actually \*does\* instead of trying to derive it.  
  
1. The key :old\_key: is to understand how `mom1` depends on `batch size`. Inspection of the formula shows that `mom1` `mom1` ranges from a minimum of `0.1` for a batch size of `2`, to `~0.9` for a batch size of `64`, and it asymptotically approaches :chart\_with\_upwards\_trend: `1` as the batch size increases :arrow\_up\_small: beyond `64`.  
  
So now we understand that `mom1` increases monotonically from 0.1 to 1 with increasing \*batch size\*. :heavy\_check\_mark:  
  
2. Now let's figure out :thinking\_face: how `mom1` is used in the `RunningBatchNorm` :woman-running:code.  
  
`mom1` is passed into PyTorch's `lerp\_` function, which is used to compute the  
exponentially weighted averages (EWMA) of `s` (the sum of the weights :weight\_lifter:) and `ss` (the sum of the squared weights :weight\_lifter:):  
  
 `# update EWMAs of sums, sqrs, which, like s and ss, have size [1,1,nf,1] `  
` self.sums.lerp\_(s, self.mom1)`  
` self.sqrs.lerp\_(ss, self.mom1)`  
  
`lerp` stands for \*l\*inear int\*erp\*olation. Recall that in `PyTorch`, adding a `\_` suffix to a function is used to denote the in-place version of the function. So `lerp\_` is the in-place version of `lerp`. The `PyTorch` documentation :book:`https://pytorch.org/docs/stable/torch.html#torch.lerp`  
tells us that  
`x = x1.lerp(x2,f) = lerp(x1,x2,f) = x1 + f\*(x2 - x1)`, where, `f` is a fraction between `0` and `1`.  
This is the classic formula for interpolating to a point `x` that is a fraction `f` of the distance between the points `x1` and `x2` :straight\_ruler:.  
  
It's interesting to play a bit with the interpolation formula. We can also write it as:  
`x = (1-f)\*x1 + f\*x2`, which we recognize as the formula for updating an exponentially weighted average (`EWMA`) !  
Mathematically, `x` an \_affine linear transformation\_ of `(x1,x2)`. A linear transformation is \_affine\_ when its coefficients are constrained to add up to one.  
  
Back to our problem: using the `EWMA` form of the interpolation formula, we can write  
  
`sums.lerp\_(s,mom1) = (1-mom1)\*s + mom1\*sums`  
The \*current value\* of `sums` is being updated from its \*previous value\* `s` by a weighted :weight\_lifter:sum of `s` and `sums`. And if `mom1` is smaller:arrow\_down\_small:/larger:arrow\_up\_small: than `0.5`, then the \*current value\* carries lower:arrow\_down\_small:/higher:arrow\_up\_small: weight :scales: (relative to the \*last value\*) in the weighted :weight\_lifter:sum.  
  
Illuminated:bulb: in this way, the RunningBatchNorm :woman-running: algorithm is now \*crystal clear\* :crystal\_ball:. The formula is just a simple way to increase the weight :weight\_lifter: of the \*current value\* (relative to the \*previous value\*) in the `EWMA`, as the batch size increases and the \*current value\* consequently becomes less noisy, because it is an average over a larger number of samples.  
  
We can now claim to understand the key :old\_key: part of the `RunningBatchNorm` :woman-running:algorithm and implementation!  
:smile\_cat::fireworks::fire::woman-cartwheeling::100:  
  
At least until they change it again :wink: