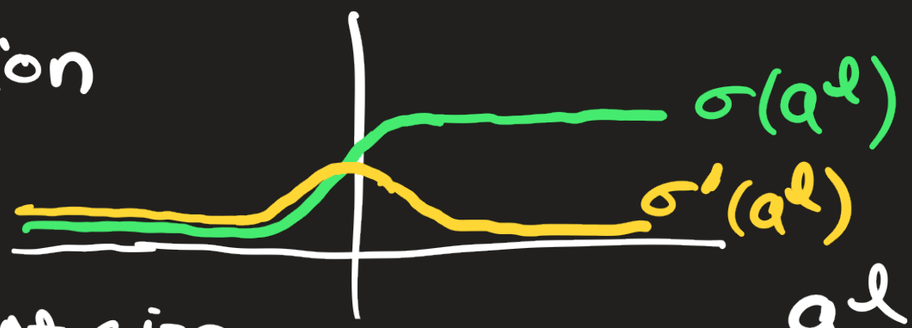
ML and Optimization Lecture 19

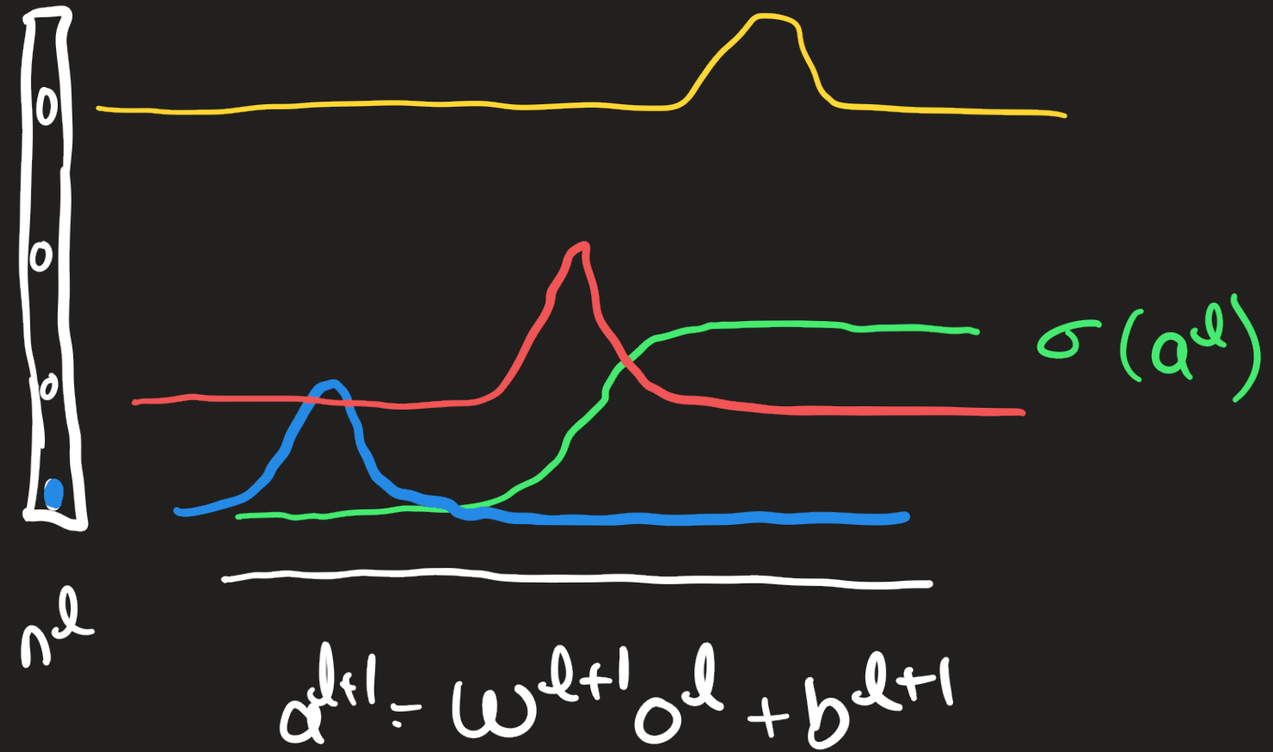
- Normalization Approaches to avoiding Vanishing & Exploding Gradients:
  - Batch Normalization
  - Layer Normalization
- Residual / Skip connections

Batch Normalization

Address activation saturation



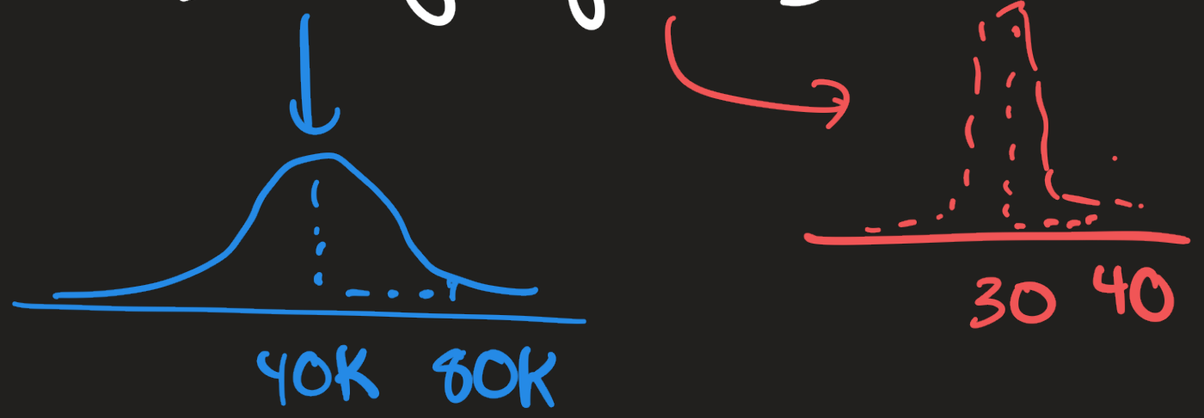
- we have decreasing gradient size if a^l is far away from the "active" portion of the activation function



Depending on the distribution of the outputs from layer l, each neuron decreases or contributes to the gradient in backprop

In convex ML we saw the advantage of normalizing our input data so the features are on the same scale: Ex: training to predict credit default probability

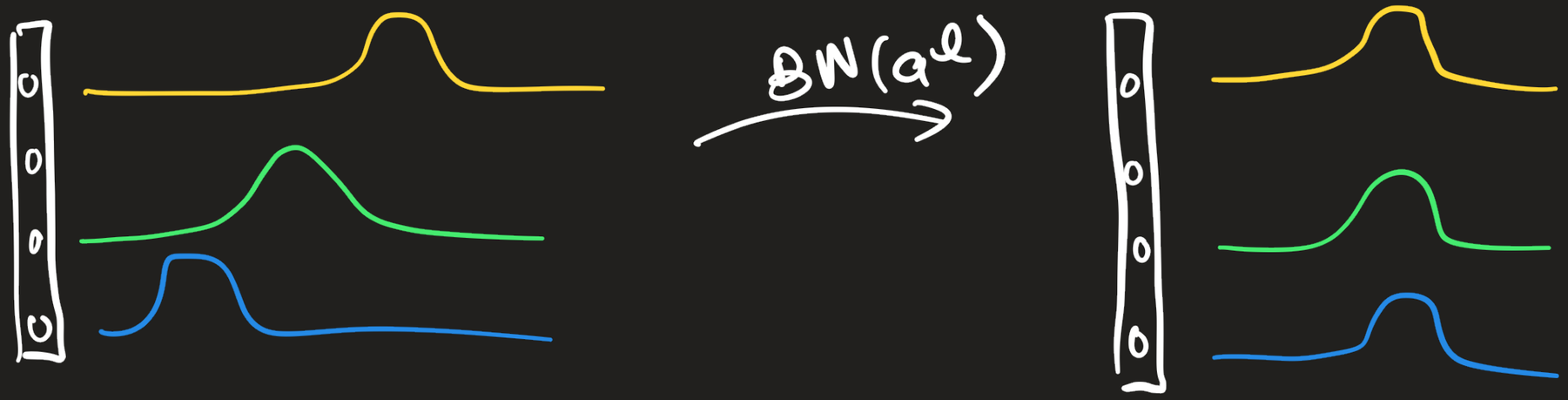
$x = [\text{salary} \quad \text{age} \quad \dots]$



to get the features on the same scale we normalized them

$$x \longmapsto \left[\frac{\text{salary} - \mu_{\text{salary}}}{\sigma_{\text{salary}}} \quad , \quad \frac{\text{age} - \mu_{\text{age}}}{\sigma_{\text{age}}} \right]$$

Idea of BN: add normalization layer after each preactivation to rescale and center the feature distributions



BN layer

$$\bar{a}^l = BN_{\gamma^l, \beta^l}(w^l o^{l-1})$$

$$o^l = \sigma(\bar{a}^l)$$

where

$$BN_{\gamma^l, \beta^l}(a^l) = \gamma^l \odot \left(\frac{a^l - \mu^l}{\sqrt{\sigma_l^2 + \varepsilon}}\right) + \beta^l$$

Here:

$\gamma^l \in \mathbb{R}^{n_l}$ is a scaling vector that controls the variance of the preactivations $\bar{a}^l$

$\beta^l \in \mathbb{R}^{n_l}$ is a shift vector that controls the means of the preactivations $\bar{a}^l$

ε - some constant (small, 10^{-6}) to avoid dividing by zero

γ^l and β^l are learned via backprop to ensure that each neuron's preactivation is in a useful portion of the activation function's domain

The means μ^l and variances σ^2_l of the preactivations are computed during training using minibatches:

e.g. if our minibatch is of size m

$$\mu_l = \frac{1}{m} \sum_{i=1}^{m} a^l(x_i)$$

model.train()

$$\sigma_l^2 = \frac{1}{m} \sum_{i=1}^{m} \left(a^l(x_i) - \mu_l\right)^2$$

Caveat: this only makes sense if $m \gg 1$.

- this means when we don't have enough memory to use a minibatch of size $\gg 1$, we can't use BN layers

During test time, we can use a running average of μ_ℓ and σ^2_ℓ:

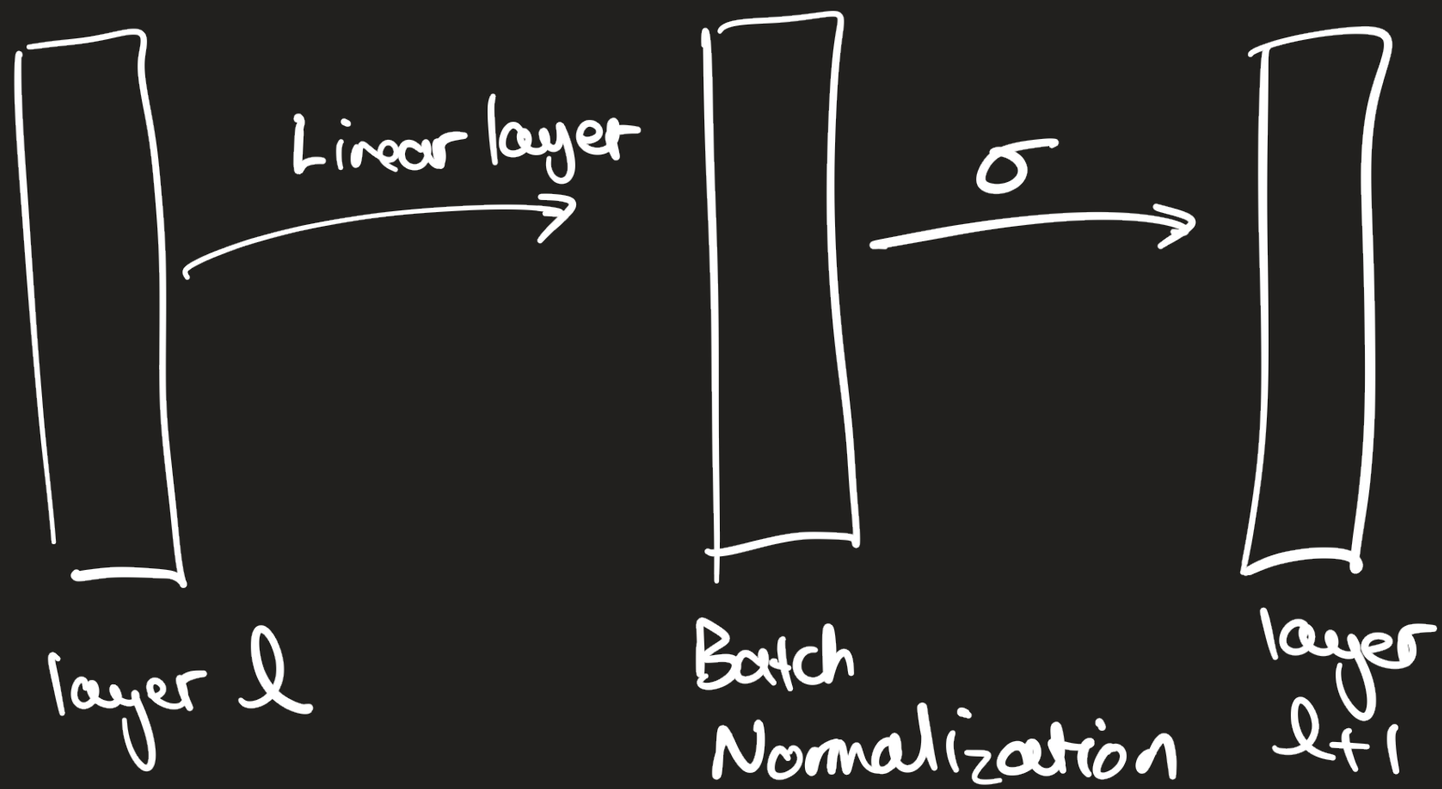
$$\mu^\ell \leftarrow \beta_1 \mu^\ell + (1-\beta_1)\mu^\ell_B \quad \text{for each minibatch } B$$

$$\sigma^{2\ell} \leftarrow \beta_2 \sigma^{2\ell} + (1-\beta_2)\sigma^2_{\ell,B}$$

model.eval()

and use μ_ℓ and σ^2_ℓ at test time

- Batch normalization for CNNs is done on a per-channel basis: you compute the means and variances using all the preactivations in a channel, and shift and scale all those preactivations in a channel by the same amount. This preserves translational invariance of features.


Linear layer
σ
layer l
Batch Normalization
layer l+1

Layer Normalization

When Batch Normalization does not apply (e.g. not enough memory for large batches) or is expensive (e.g. with RNNs you need to store separate BN parameters for each possible sequence length), we can try layer normalization.

Treat all the neurons in a layer as i.i.d. samples, and use to estimate the means and variances used in normalization

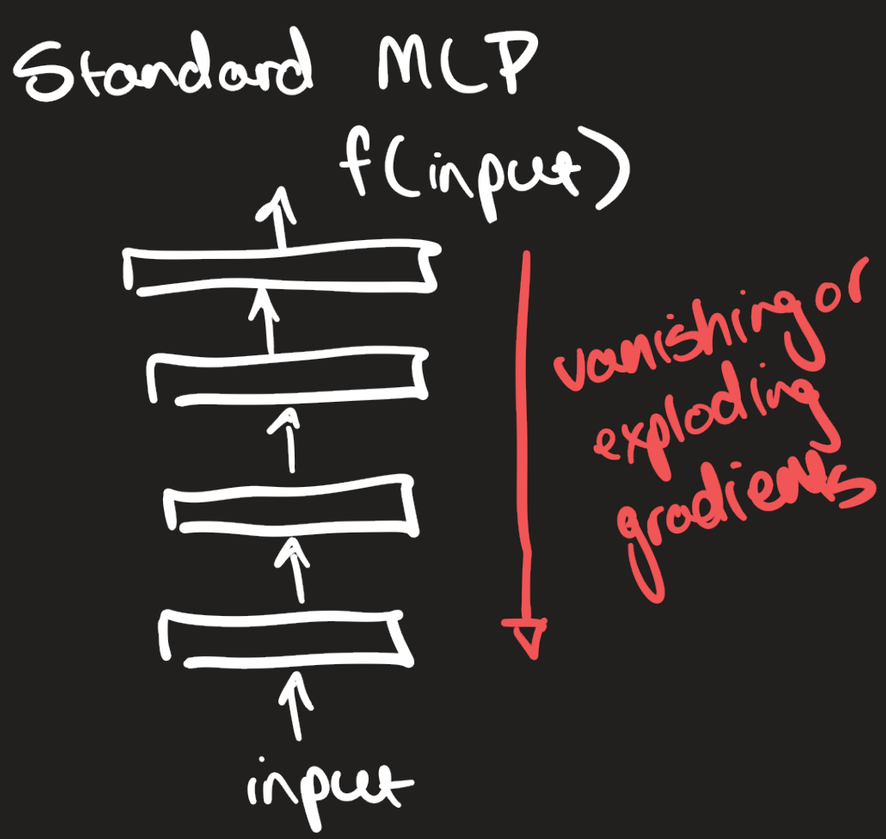
$$\bar{a}^l = \text{LayerNorm}_{\gamma,\beta}(a^l) = \gamma \odot \frac{a^l - \mu}{\sigma} + \beta$$

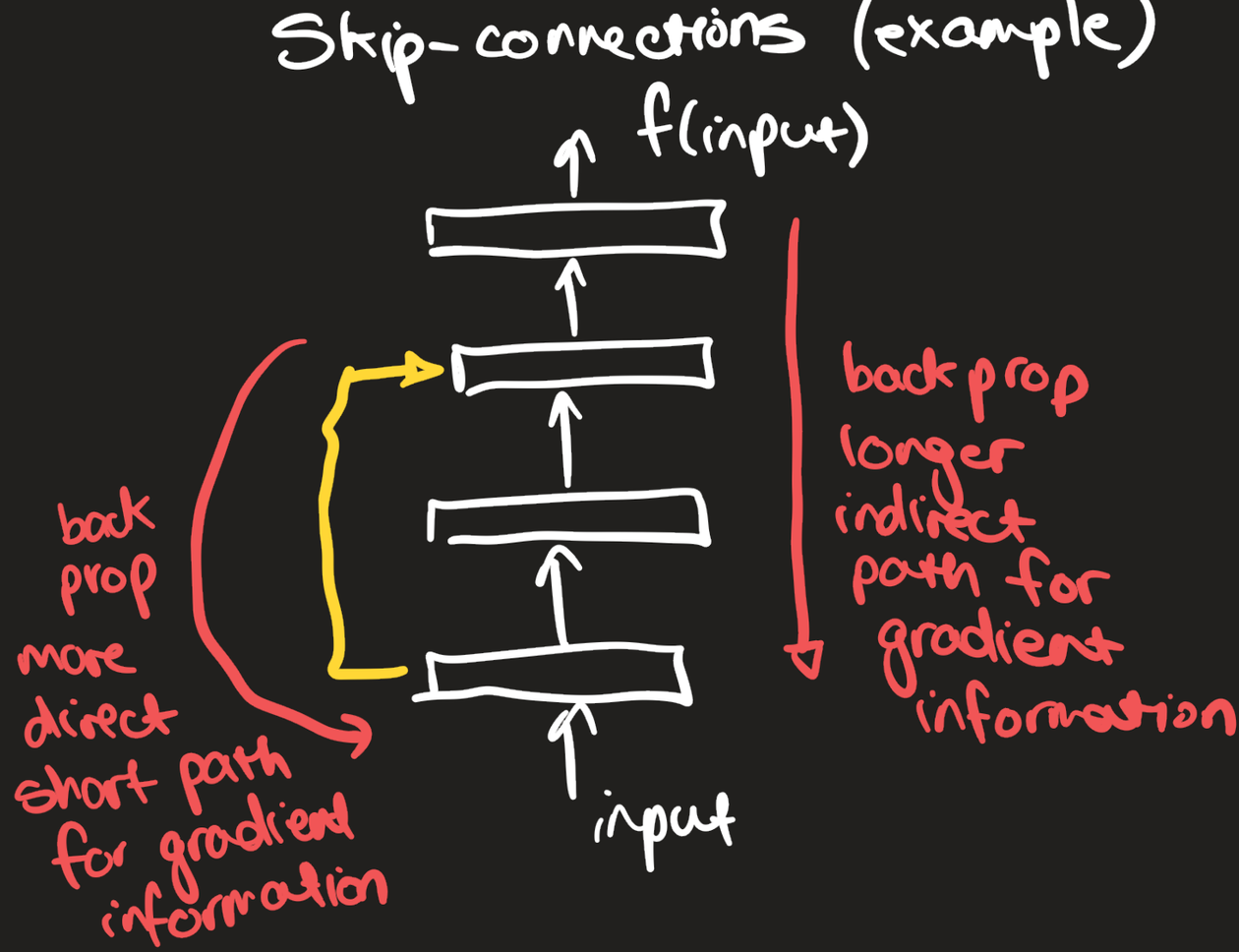
per instance normalization

where $\gamma \in \mathbb{R}^{n_l}$ and $\beta \in \mathbb{R}^{n_l}$

and $\mu = \frac{1}{n_l}\sum_{i=1}^{n_l} a^l(x)$ and $\sigma^2 = \frac{1}{n_l}\|a^l(x) - \mu\|_2^2$

Skip/Residual Connections





Skip-connections are connections b/w layers l and layers higher than $l+1$, e.g. DenseNet where all layers are connected

Rationales for skip-connections:

1) purely pragmatic: helps w/ gradient propagation
2) inductive bias: higher layers have access to simple features (e.g. U-Nets)

A specific popular architecture in this class is the ResNet.

$$o^{l+1} = \sigma(w^{l+1} o^l + b^{l+1}) + o^l$$

Idea: this architecture allows us to easily compute simple functions like $f(x) = x$ as well as complex functions involving nonlinearities.