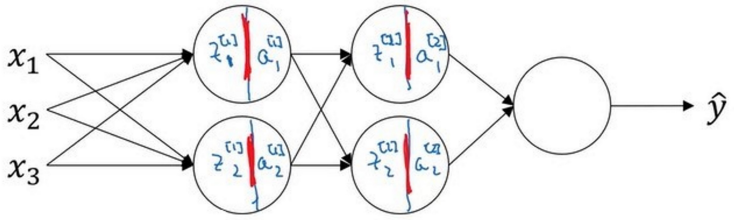
Week 3b - Batch Normalization

Friday, August 14, 2020 11:30 PM

① Normalizing activations in a network

- In earlier weeks, we normalized the inputs ($x = x - \mu / \sigma$)
- In deeper neural networks, normalizing hidden layer outputs also help to train model better
- Implementing Batch Norm
  - → Given some intermediate values in NN $z^{(1)} \dots z^{(m)}$ $\quad z^{[l](i)}$
  - → $\mu = \frac{1}{m} \sum_i z^{(i)}$
  - → $\sigma^2 = \frac{1}{m} \sum_i (z_i - \mu)^2$
  - → $z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$
  - → $\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$ → γ & β are learnable parameters of model that can change mean & variance to whatever we want
    ($\tilde{z}^{(i)} = z^{(i)}$ if $\gamma = \sqrt{\sigma^2 + \varepsilon}$, $\beta = \mu$)
  - → use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$ in model

② Fitting Batch Norm into a Neural Network



- Batch Norm occurs between calc. of $z^{[l]}$ & $a^{[l]}$

$\Rightarrow X \xrightarrow{w^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[BN]{\gamma^{[1]}, \beta^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[BN]{\gamma^{[2]}, \beta^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \rightarrow \dots$

- Parameters: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \dots, w^{[l]}, b^{[l]}$ } compute db, dw
  $\beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \beta^{[2]}, \dots, \beta^{[l]}, \gamma^{[l]}$ } $d\beta, d\gamma$

  also update $\beta^{[l]} = \beta^{[l]} - \alpha d\beta^{[l]}$

  $\gamma^{[l]} = \gamma^{[l]} - \alpha d\gamma^{[l]}$

- Working with mini-batches

- Working with mini-batches

$X^{\{1\}} \xrightarrow{w^{[1]}, b^{[1]}} z^{[1]} \xrightarrow{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow g^{[1]}(\tilde{z}^{[1]}) = a^{[1]} \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \rightarrow \ldots$

$X^{\{2\}} \xrightarrow{w^{[2]}, b^{[2]}} z^{[1]} \xrightarrow{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \rightarrow g^{[1]}(\tilde{z}^{[1]}) = a^{[2]} \xrightarrow{w^{[2]}, b^{[2]}} z^{[2]} \rightarrow \ldots$

$X^{\{3\}} \rightarrow \ldots$

Parameters: $w^{[1]}$, ~~$b^{[1]}$~~, $\beta^{[1]}$, $\gamma^{[1]}$

$z^{[1]}$ $(n^{[1]}, 1)$ $\quad$ $(n^{[1]}, 1)$ $(n^{[1]}, 1)$

$z^{[l]} = w^{[l]} a^{[l-1]} + \cancel{b^{[l]}}$ → b isn't required since β over-rides b. Hence β is enough

$z^{[l]} = w^{[l]} a^{[l-1]}$

$z^{[l]}_{norm}$

$\tilde{z}^{[l]} = \gamma^{[l]} z^{[l]}_{norm} + \beta^{[l]}$

- Implementing gradient descent

→ for $t = 1 \ldots$ num_mini_batches
- Compute forward prop on $X^{\{t\}}$
  In each hidden layer, use BN to replace $z^{[l]}$ with $\tilde{z}^{[l]}$
- use backprop to compute $dw^{[l]}$, ~~$db^{[l]}$~~, $d\beta^{[l]}$, $d\gamma^{[l]}$
- Update parameters

$w^{[l]} = w^{[l]} - \alpha\, dw^{[l]}$

$\beta^{[l]} = \ldots$

$\gamma^{[l]} = \ldots$

→ works w/ momentum, RMSprop, Adam

③ Why does Batch Norm work

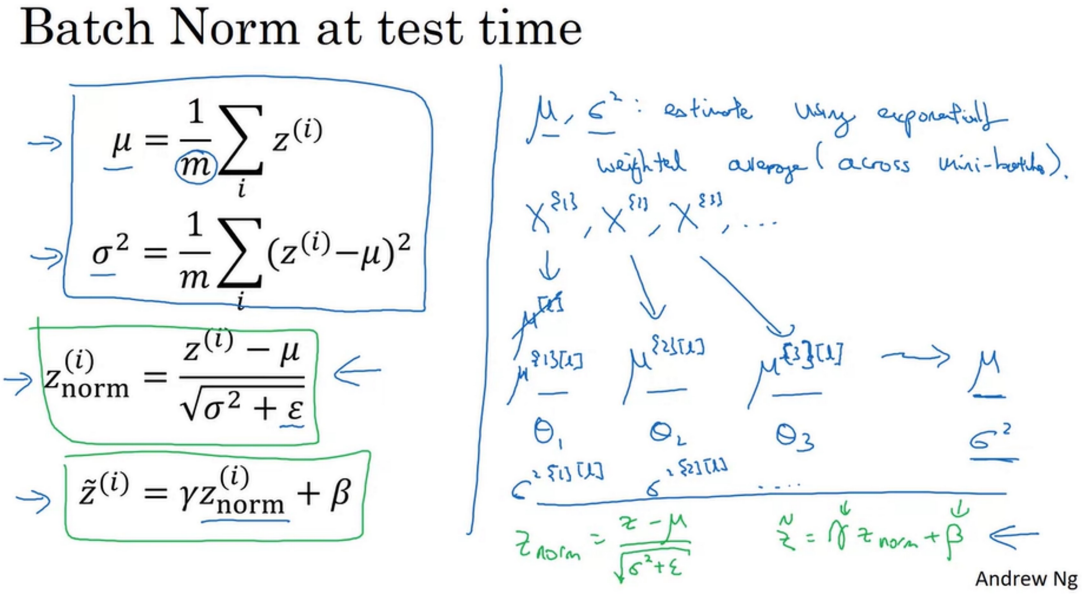
→ Serves same purpose as normalizing inputs

→ Batch normalization reduces the problem of input values changing (shifting)

→ Has regularization effect (indirectly)
- each mini-batch is scaled by the mean/variance of that batch
- This adds noise to $z^{[l]}$ in that mini-batch.
- Effect is similar to dropout (adding noise)
- effect is reduced by increasing batch size
- Not intended to be used for regularization but only to speed up learning

④ Batch Norm at Test time

④ Batch Norm at Test time



- During training, mean & variance is computed for each mini-batch
- In testing, we might have to pro