Hyperparameter Tuning and Batch Normalization

Most of this material is from Prof. Andrew Ng'and Chang's slides.

Hyperparameters

- α : learning rate
- β : momentum
- $\beta_1, \beta_2, \varepsilon$: Adam
- # of layers
- # of hidden units
- learning rate decay
- mini-batch size
- ...

Hyperparameters

- α : learning rate
- β : momentum
- $\beta_1, \beta_2, \varepsilon$: Adam
- # of layers
- # of hidden units
- learning rate decay
- mini-batch size
- ...

Hyperparameters

- α : learning rate
- β : momentum ~ 0.9
- $\beta_1, \beta_2, \varepsilon$: Adam
- # of layers
- # of hidden units
- learning rate decay
- mini-batch size
- ...

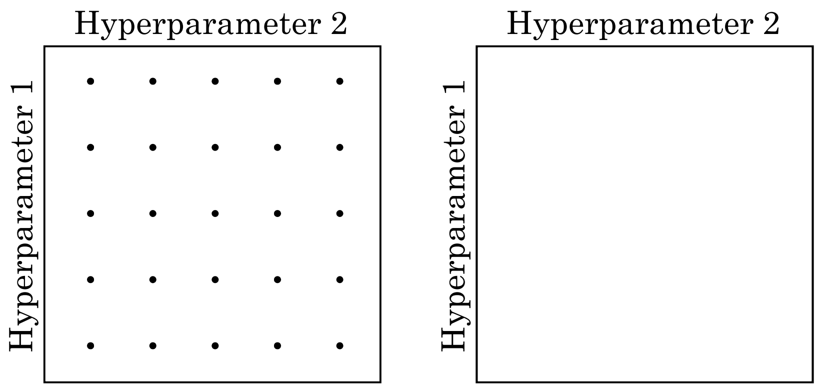
Hyperparameters

- α : learning rate
- β : momentum ~ 0.9
- $\beta_1, \beta_2, \varepsilon$: Adam
- # of layers
- # of hidden units
- learning rate decay
- mini-batch size
- …

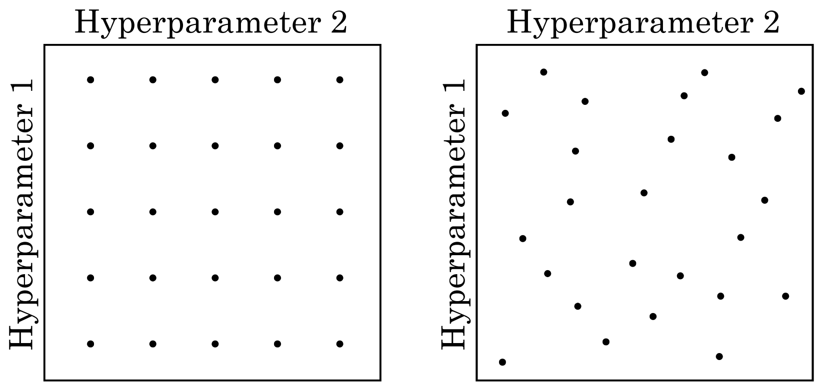
Hyperparameters

- α : learning rate
- β : momentum ~ 0.9
- $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$: Adam
- # of layers
- # of hidden units
- learning rate decay
- mini-batch size
- …

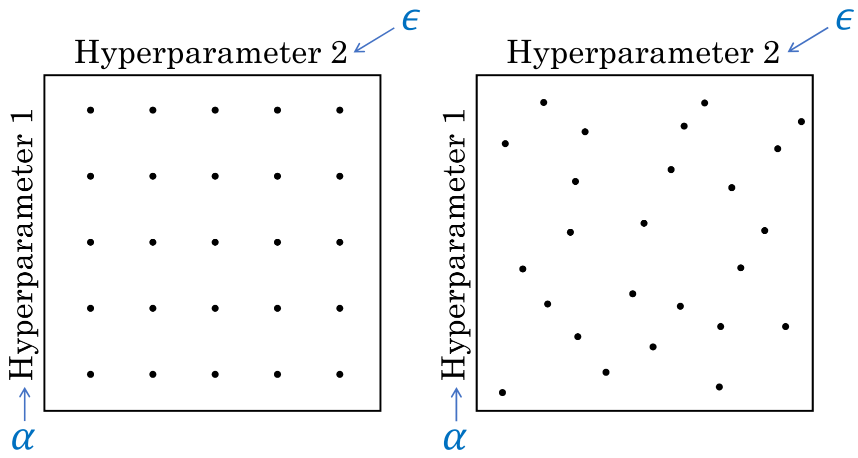
Try random values: Don't use a grid



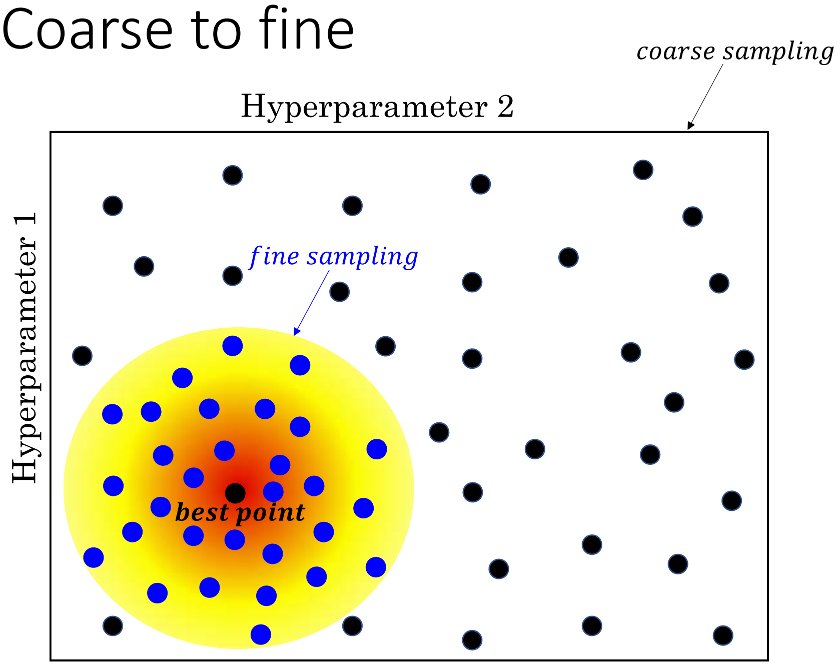
Try random values: Don't use a grid



Try random values: Don't use a grid



Coarse to fine



Picking hyperparameters at random

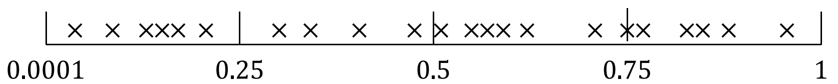
- $n^{[l]} = 50, \dots, 100$
  - the number of hidden units can be sampled uniformly at random between 50 and 100

- $L = 2, \dots, 5$
  - the number of layers can be sampled uniformly between 2 and 5

OK!

Appropriate scale for hyperparameters

$\alpha = 0.0001, \dots, 1$



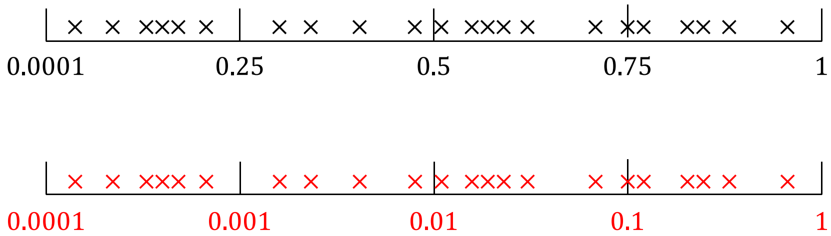
uniform over all scales (x)

large scale → sparse & small scale → dense (O)

How!

Appropriate scale for hyperparameters

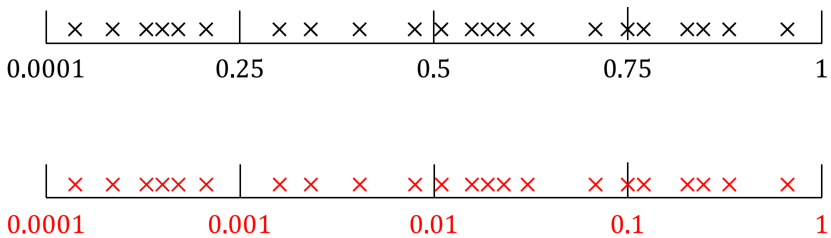
$\alpha = 0.0001, \dots, 1$



log-scale uniform sampling at random

Appropriate scale for hyperparameters

$\alpha = 0.0001, \dots, 1$



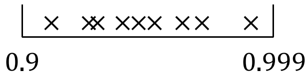
uniform (random) sample $r \in [-4, 0]$

log–scale uniform sample $\alpha = 10^r \in [10^{-4}, 10^0]$

Hyperparameters for exponentially weighted averages

$\beta = 0.9, \ldots, 0.999$

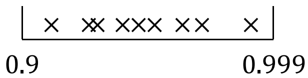
precision level 1/10   *precision level* 1/1000



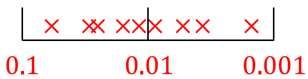
Hyperparameters for exponentially weighted averages

$\beta = 0.9, \ldots, 0.999$

precision level 1/10   *precision level* 1/1000



$1 - \beta = 0.1, \ldots, 0.001$



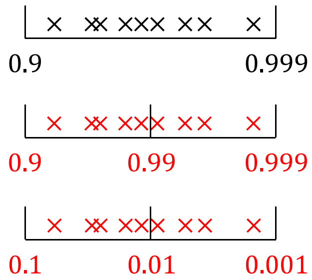
log-scale uniform sampling

Hyperparameters for exponentially weighted averages

$\beta = 0.9, \ldots, 0.999$

precision level 1/10   *precision level* 1/1000

$1 - \beta = 0.1, \ldots, 0.001$



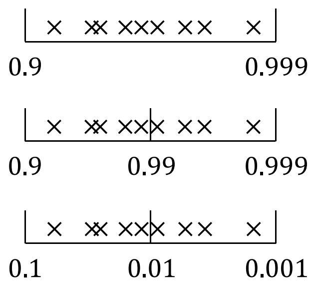
log-scale uniform sampling

Hyperparameters for exponentially weighted averages

$\beta = 0.9, \ldots, 0.999$

precision level 1/10   *precision level* 1/1000

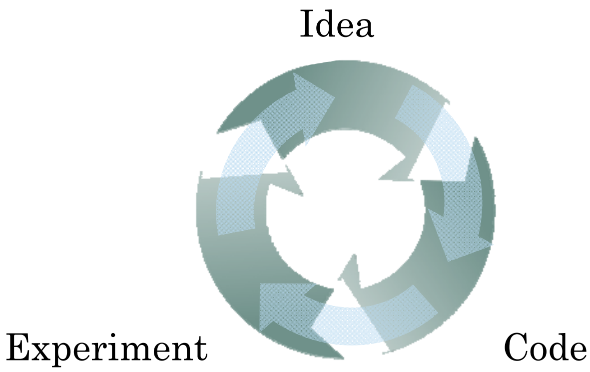
$1 - \beta = 0.1, \ldots, 0.001$



$\beta = 0.9000 \longrightarrow 0.9005$ (little significant!)

$\beta = 0.9990 \longrightarrow 0.9995$ (significant enough!)

Re-test hyperparameters occasionally

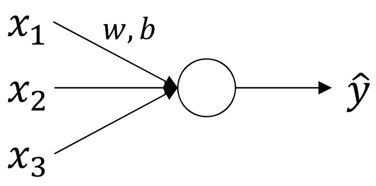


- NLP, Vision, Speech, Ads, logistics, ….
- Existing intuitions do get stale. Do not reuse but re-search occasionally especially for different domains.

Batch Normalization

- One of the most important algorithms in deep learning created by Sergey Ioffe and Christian Szegedy
- Makes the hyperparameters search problem much easier
- Make the neural network much more robust to the choice of hyperparameters
- There is a much bigger range of hyperparameters that work well
- Enable you to much more easily train even very deep networks

Normalizing inputs to speed up learning



x

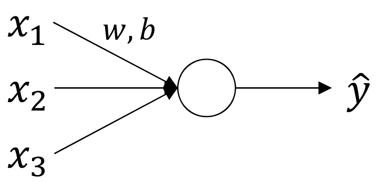
i th input vector (training data)

$$\mu = \frac{1}{m}\sum_i x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(x^{(i)} - \mu\right)^2, \quad x^{(i)}_{\text{norm}} = \frac{x^{(i)} - \mu}{\sigma}$$

element-wise

Normalizing inputs to speed up learning

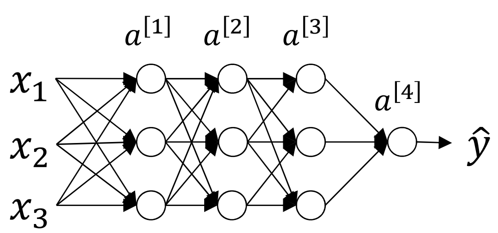


i th input vector (training data)

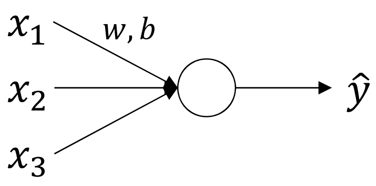
$$\mu = \frac{1}{m}\sum_i x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(x^{(i)} - \mu\right)^2, \quad x^{(i)}_{\text{norm}} = \frac{x^{(i)} - \mu}{\sigma}$$

element-wise



Normalizing inputs to speed up learning

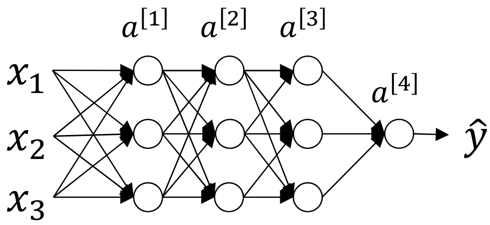


i th input vector (training data)

$$\mu = \frac{1}{m}\sum_i x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(x^{(i)} - \mu\right)^2, \quad x_{\text{norm}}^{(i)} = \frac{x^{(i)} - \mu}{\sigma}$$

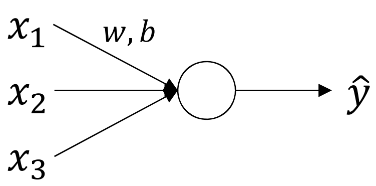
element-wise



Can we normalize $a^{[l]}$ so as to train $W^{[l+1]}$, $b^{[l+1]}$ faster?

Batch Normalization

Normalizing inputs to speed up learning

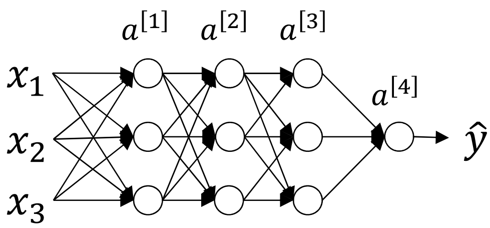


i th input vector (training data)

$$\mu = \frac{1}{m}\sum_i x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(x^{(i)} - \mu\right)^2, \quad x_{\text{norm}}^{(i)} = \frac{x^{(i)} - \mu}{\sigma}$$

element-wise



Can we normalize $a^{[l]}$ so as to train $W^{[l+1]}$, $b^{[l+1]}$ faster?

Batch Normalization

Actually normalize $z^{[l]}$ not $a^{[l]}$

Implementing Batch Norm

Given some intermediate values in NN: $\underbrace{z^{(1)}, z^{(2)}, \ldots, z^{(m)}}_{z^{[l](i)}}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

zero mean and unit variance

for numerical stability

Implementing Batch Norm

Given some intermediate values in NN: $\underbrace{z^{(1)}, z^{(2)}, \ldots, z^{(m)}}_{z^{[l](i)}}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

trainable parameters of model

$$\tilde{z}^{(i)} = \gamma \cdot z_{\text{norm}}^{(i)} + \beta$$

variant mean and variance for better-training

Implementing Batch Norm

Given some intermediate values in NN: $\underbrace{z^{(1)}, z^{(2)}, \ldots, z^{(m)}}_{z^{[l](i)}}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma \cdot z_{\text{norm}}^{(i)} + \beta$$

trainable parameters of model

variant mean and variance for better-training

If

trained $\gamma = \sqrt{\sigma^2 + \varepsilon}$

trained $\beta = \mu$

then $\tilde{z}^{(i)} = z^{(i)}$

Implementing Batch Norm

Given some intermediate values in NN: $\underbrace{z^{(1)}, z^{(2)}, \ldots, z^{(m)}}_{z^{[l](i)}}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma \cdot z_{\text{norm}}^{(i)} + \beta$$

learnable parameters of model

If

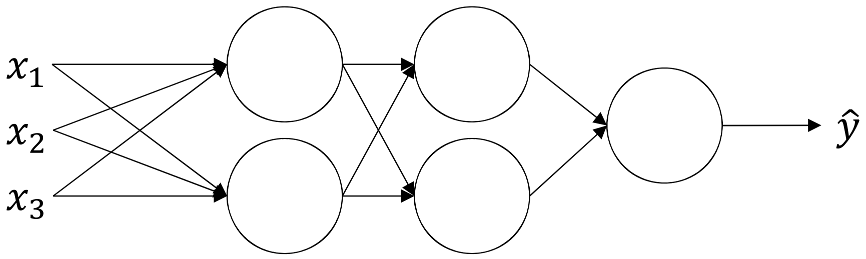
trained $\gamma = \sqrt{\sigma^2 + \varepsilon}$

trained $\beta = \mu$

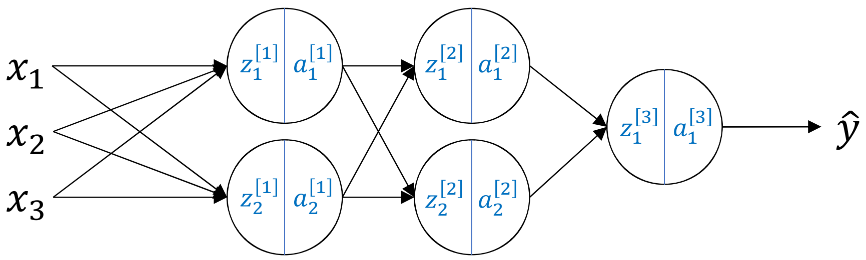
then $\tilde{z}^{(i)} = z^{(i)}$

Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$ → flexible normalization!

Adding Batch Norm to a network

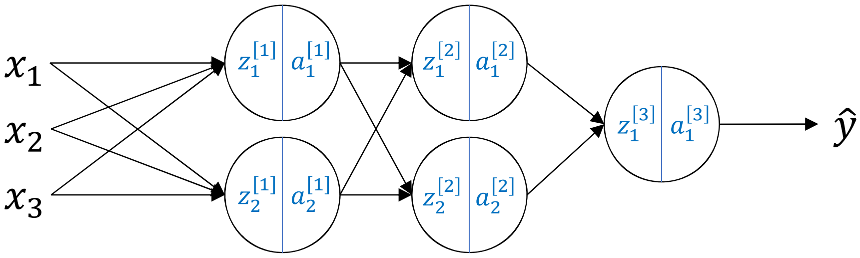


Adding Batch Norm to a network



$$X \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \xrightarrow[\text{Batch Norm}]{\beta^{[2]},\, \gamma^{[2]}} \tilde{Z}^{[2]} \longrightarrow \cdots$$

Adding Batch Norm to a network



$$X \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \xrightarrow[\text{Batch Norm}]{\beta^{[2]},\, \gamma^{[2]}} \tilde{Z}^{[2]} \longrightarrow \cdots$$

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots, W^{[L]}, b^{[L]}$

$\beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \ldots, \beta^{[L]}, \gamma^{[L]}$

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

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$$X^{\{2\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$\vdots$

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$$X^{\{2\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$\vdots$

batch normalization is performed on just each mini-batch!

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$$X^{\{2\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$\vdots$

batch normalization is performed on just that mini-batch!!

Parameters: $W^{[l]}, b^{[l]}, \beta^{[l]}, \gamma^{[l]}$

$$Z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$$X^{\{2\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$\vdots$

batch normalization is performed on just that mini-batch!!

Parameters: $W^{[l]}, \cancel{b^{[l]}}, \beta^{[l]}, \gamma^{[l]}$

$$Z^{[l]} = W^{[l]}a^{[l-1]} + \cancel{b^{[l]}}$$

$$\left\{ \begin{aligned} & z^{[l]} = W^{[l]}a^{[l-1]} \\ & z^{[l]}_{\text{norm}} \\ & \tilde{z}^{[l]} = \gamma^{[l]} z^{[l]}_{\text{norm}} + \boxed{\beta^{[l]}} \end{aligned} \right.$$

Working with mini-batches

$$X^{\{1\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$$X^{\{2\}} \xrightarrow{W^{[1]},\, b^{[1]}} Z^{[1]} \xrightarrow[\text{Batch Norm}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{Z}^{[1]} \longrightarrow A^{[1]} = g^{[1]}(\tilde{Z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} Z^{[2]} \longrightarrow \cdots$$

$\vdots$

batch normalization is performed on just that mini-batch!!

Parameters: $W^{[l]}, \cancel{b^{[l]}}, \beta^{[l]}, \gamma^{[l]}$

$(n^{[l]}, 1)$ $(n^{[l]}, 1)$

$$Z^{[l]} = W^{[l]}a^{[l-1]} + \cancel{b^{[l]}}$$

$$\left\{ \begin{aligned} & z^{[l]} = W^{[l]}a^{[l-1]} \\ & z^{[l]}_{\text{norm}} \\ & \tilde{z}^{[l]} = \gamma^{[l]} z^{[l]}_{\text{norm}} + \boxed{\beta^{[l]}} \end{aligned} \right.$$

Implementing gradient descent

for $t = 1, \ldots, \text{numMiniBatches}$

compute forward prop on $X^{\{t\}}$

In each hidden layer, use BN to replace $Z^{[l]}$ with $\tilde{Z}^{[l]}$

use backprop to compute $dW^{[l]}, \cancel{db^{[l]}}, d\beta^{[l]}, d\gamma^{[l]}$

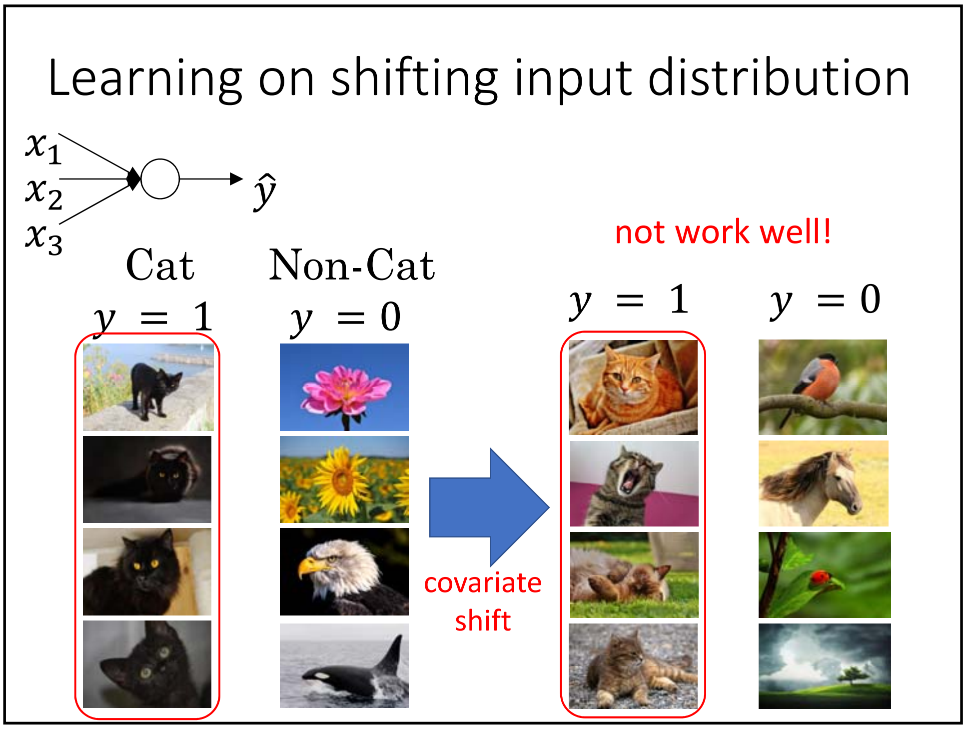
update parameters:

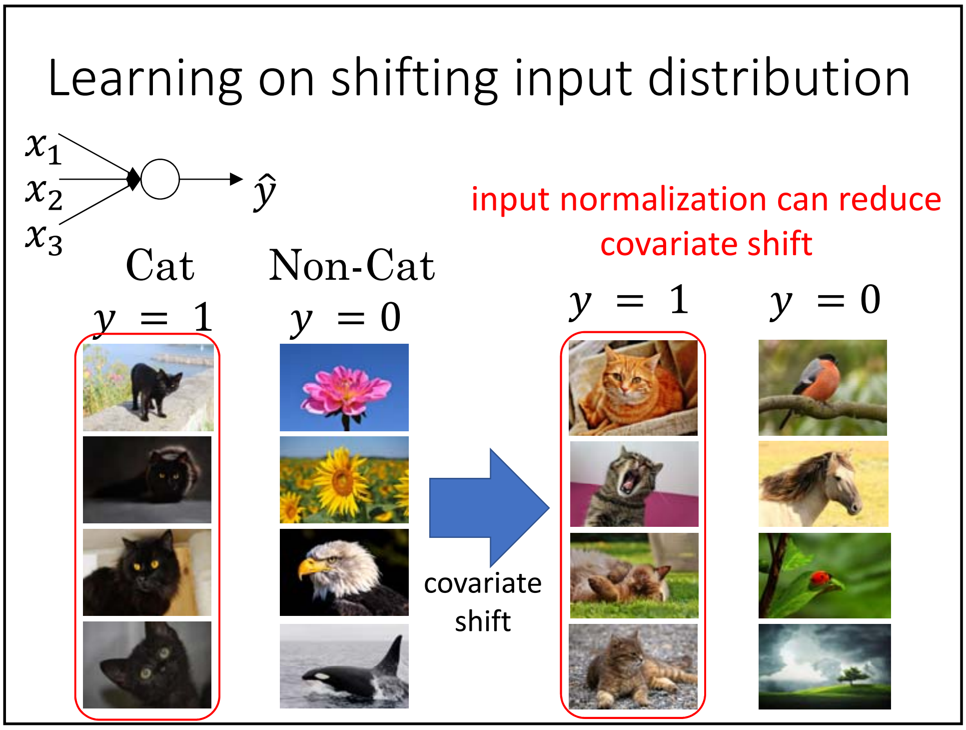
$$\left.\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha \cdot dW^{[l]} \\ \beta^{[l]} &:= \beta^{[l]} - \alpha \cdot d\beta^{[l]} \\ \gamma^{[l]} &:= \gamma^{[l]} - \alpha \cdot d\gamma^{[l]} \end{aligned}\right\} \text{gradient descent}$$

we can also use momentum, rmsprop, adam, ...

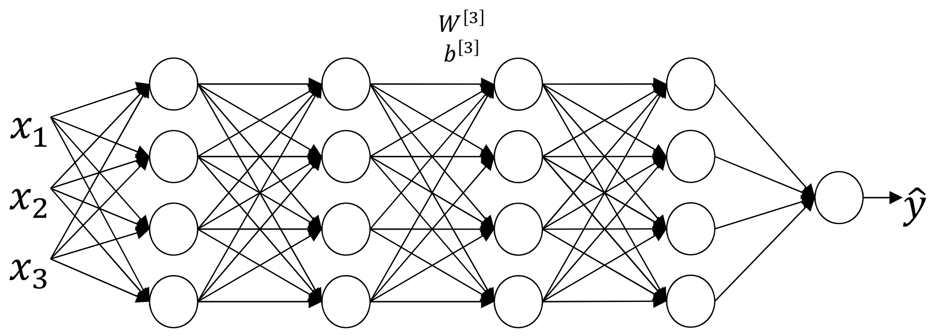
Why does batch norm work?

- Normalizing the input features to mean zero and variance one speed up learning
- Batch norm is doing a similar thing, but for the values in the hidden units and not just for the input units
- But, this is just a partial picture for what batch norm is doing and there are a couple of futher intuitions

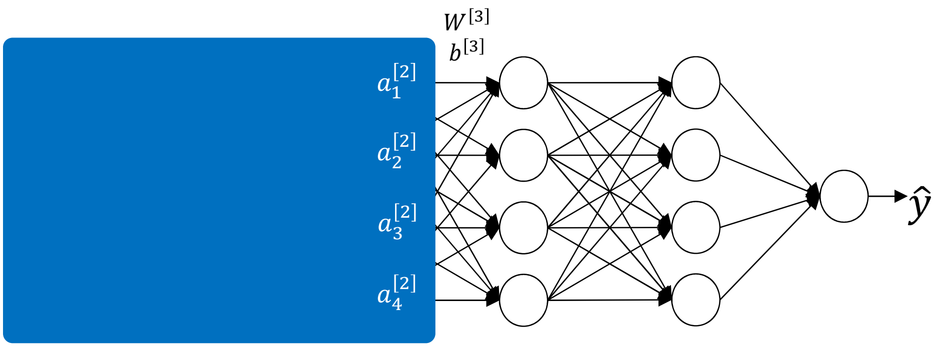

Learning on shifting input distribution
x_1
x_2
x_3
$\hat{y}$
Cat
$y = 1$
Non-Cat
$y = 0$
covariate shift
not work well!
$y = 1$
$y = 0$


Learning on shifting input distribution
x_1
x_2
x_3
$\hat{y}$
Cat
$y = 1$
Non-Cat
$y = 0$
covariate shift
input normalization can reduce covariate shift
$y = 1$
$y = 0$

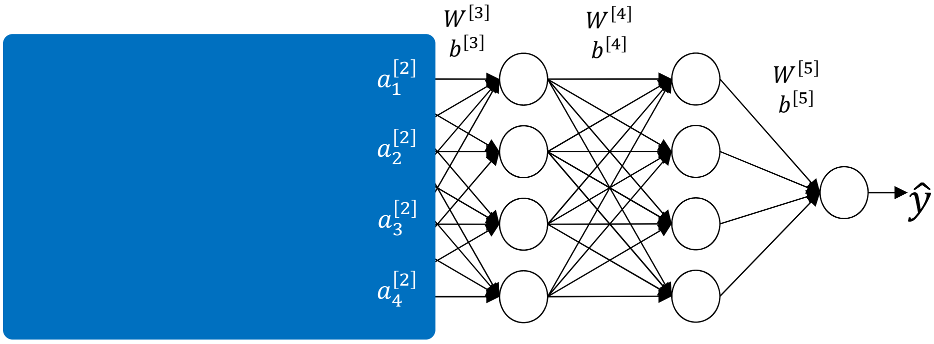
Why this is a problem with neural networks?



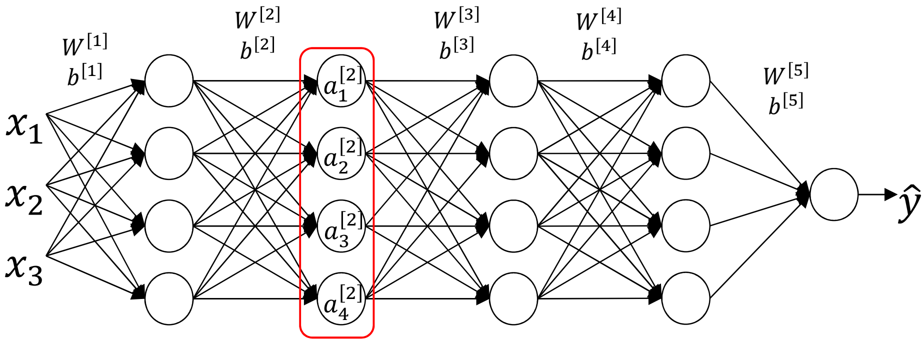
Why this is a problem with neural networks?



Why this is a problem with neural networks?



Why this is a problem with neural networks?



BN reduces covariate shift on each hidden layer

Batch Norm as regularization

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.
- This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to eash hidden layer's activations
- This has a slight regularization effect.

Batch Norm at test(or operation) time

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z} = \gamma z_{\text{norm}}^{(i)} + \beta$$

Batch Norm at test(or operation) time

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z} = \gamma z_{\text{norm}}^{(i)} + \beta$$

μ, σ^2 : estimate using exponentially weighted average (across mini-batches)

Batch Norm at test(or operation) time

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z} = \gamma z_{\text{norm}}^{(i)} + \beta$$

μ, σ^2 : estimate using exponentially weighted average (across mini-batches)

$X^{\{1\}}, X^{\{2\}}, X^{\{3\}}, \ldots$

$weighted\_avg(\mu^{\{1\}[l]}, \mu^{\{2\}[l]}, \mu^{\{3\}[l]}, \ldots) \rightarrow \mu$

$weighted\_avg({\sigma^2}^{\{1\}[l]}, {\sigma^2}^{\{2\}[l]}, {\sigma^2}^{\{3\}[l]}, \ldots) \rightarrow \sigma^2$

Batch Norm at test(or operation) time

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z} = \gamma z_{\text{norm}} + \beta$$

μ, σ^2 : estimate using exponentially weighted average (across mini-batches)

$X^{\{1\}}, X^{\{2\}}, X^{\{3\}}, \ldots$

$weighted\_avg(\mu^{\{1\}[l]}, \mu^{\{2\}[l]}, \mu^{\{3\}[l]}, \ldots) \rightarrow \mu$

$weighted\_avg({\sigma^2}^{\{1\}[l]}, {\sigma^2}^{\{2\}[l]}, {\sigma^2}^{\{3\}[l]}, \ldots) \rightarrow \sigma^2$

$z_{\text{norm}} = \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}}$ $\quad \tilde{z} = \gamma z_{\text{norm}} + \beta$

- END -