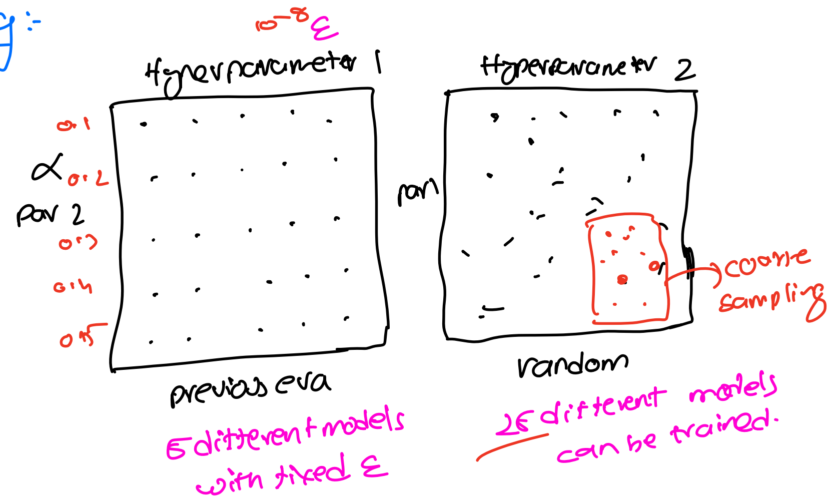
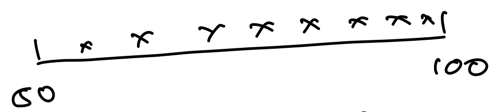
Sec1: Hyperparameter Tuning:-

- α
- β 0.9
- $\beta_1, \beta_2, \varepsilon$ 0.9 0.99 10^{-8}
- layers
- hidden units
- learning rate decay
- mini batch size



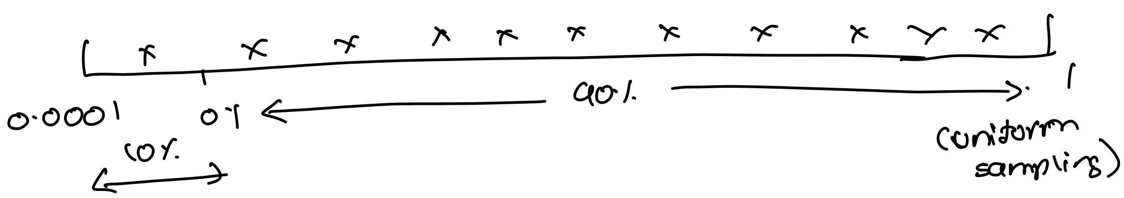
Sec2: Scale to pick hyperparameters:-

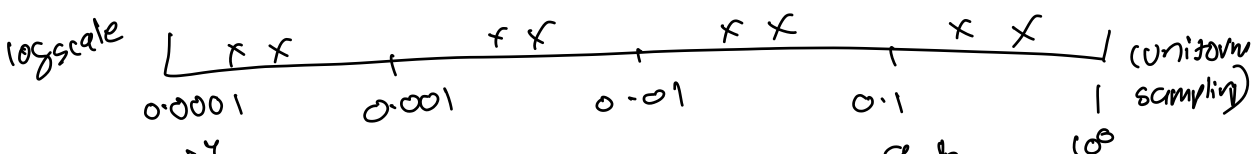
$n^{[l]} = 50, \dots, 100$ (uniformly random)



#layers: L : 2-4 (2, 3, 4) (uniformly random sampling)

$\alpha = 0.0001, \dots, 1$





$r = -4 * np.random.rand()$ $r \in [-4, 0]$

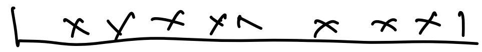
$\alpha = 10^r$ $\alpha \in [10^{-4}, 10^0]$

$10^a \dots 10^b$

$\beta = 0.9, \ldots, 0.999$

$\downarrow$ avg 10 values

$\downarrow$ avg 1000 values



uniform sampling not good.

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

$r \in [-3, -1] \rightarrow$ uniformly sampling over.

$1-\beta = 10^r$

$\beta = 1 - 10^r$

β : $0.9000 \rightarrow 0.9005$ } ~10

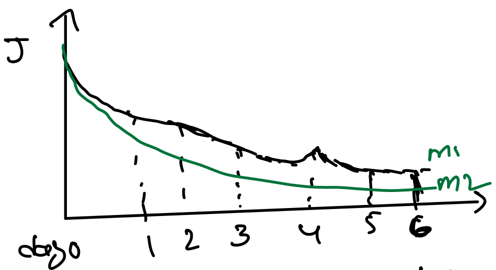
β : $0.999 \rightarrow 0.9995$

1000 ~2000

lec3: Hyperparameter search process:

→ reevaluate hyperparameters occasionally

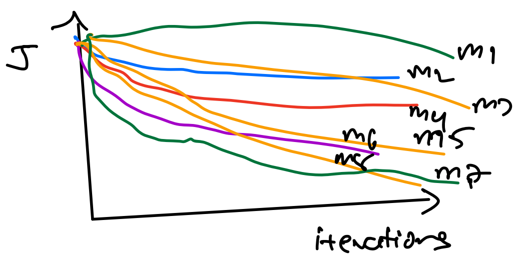
Babysitting one model



change the parameters as the model train.

Panda approach

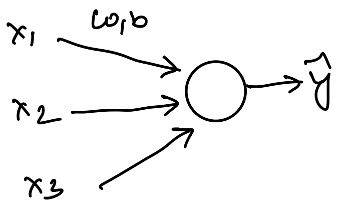
train many models in parallel



caviar approach. (m7)

dec4: Batch Normalization

Normalizing inputs to speed up learning



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

$x = x - \mu$

$\sigma^2 = \frac{1}{m} \sum_i x^{(i)2}$

$x = x / \sigma$

Given some intermediate values in NN $z^{(1)}, \dots, z^{(m)}$ for $z^{[l](i)}$

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

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

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

learnable parameters of model.

$$\tilde{z}^{[l](i)} = \gamma \, z_{norm}^{[l](i)} + \beta$$

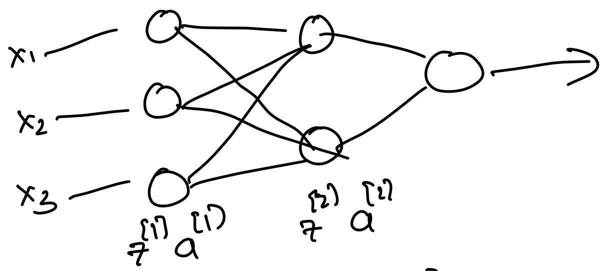
if $\gamma = \sqrt{\sigma^2 + \varepsilon}$ then $\tilde{z}^{[l](i)} = z^{[l](i)}$

$\beta = \mu$

Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$ for next process.

(γ, β we can vary)

Lec 6: Batch Norm to NN



$$X \xrightarrow{W^{[1]},\, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]},\, \gamma^{[1]}} \tilde{z}^{[1]} \xrightarrow[\text{activation}]{g^{[1]}(\tilde{z}^{[1]})} a^{[1]}$$

$$a^{[1]} \xrightarrow{W^{[2]},\, b^{[2]}} z^{[2]} \xrightarrow[\text{Batch Norm}]{\beta^{[2]},\, \gamma^{[2]}} \tilde{z}^{[2]} \xrightarrow[\text{activation}]{g^{[2]}(\tilde{z}^{[2]})} a^{[2]}$$

unnormalized　　　normalized

parameters:-

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

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

compute $d\beta^{[l]}$　　$\beta^{[l]} = \beta^{[l]} - \alpha\, d\beta^{[l]}$

$d\gamma^{[l]}$　　$\gamma^{[l]} = \gamma^{[l]} - \alpha\, d\gamma^{[l]}$　/ update.

tf.nn.batch_normalization.

BN with minibatch:-

for t = 1, to minibatches.

$$X^{\{t\}} \xrightarrow{W^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[BN]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \xrightarrow{g^{[1]}} a^{[1]}$$

$$a^{[1]} \xrightarrow{W^{[2]}, b^{[2]}} z^{[2]} \xrightarrow[BN]{\beta^{[2]}, \gamma^{[2]}} \tilde{z}^{[2]} \xrightarrow{g^{[1]}} a^{[2]}$$

parameters:- $w^{[l]}, \cancel{b^{[l]}}, \beta^{[l]}, \gamma^{[l]}$

$$z^{[l]} = w^{[l]} a^{[l-1]} + \cancel{b^{[l]}}$$

BN makes $z^{[l]}$ – mean 0, $\sigma^2 = 1$, then rescale to $\beta^{[l]}, \gamma^{[l]}$

Hence we can omit $b^{[l]}$

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

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

$\therefore \beta^{[l]} = (n^{[l]}, 1)$ $\quad \gamma^{[l]} = (n^{[l]}, 1)$

Implementing GD :

for $t = 1, \cdots,$ num minibatches

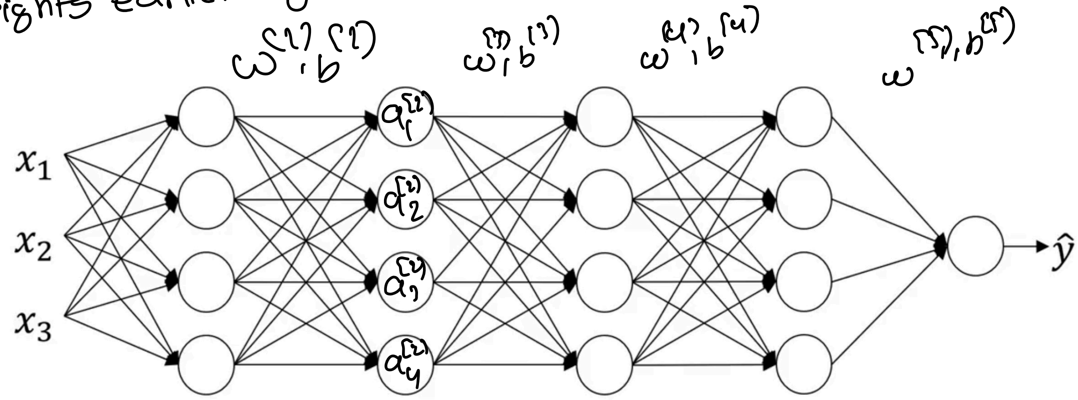
1. Compute forward prop on $X^{\{t\}}$
   - in each hidden layer, use BN to replace $z^{[l]}$ with $\tilde{z}^{[l]}$
2. use backprop to compute $dw^{[l]}$, $\cancel{db^{[l]}}$, $d\beta^{[l]}$, $d\gamma^{[l]}$
3. update parameters

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

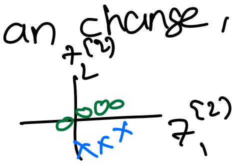
Lec6: Why BN really Works:-

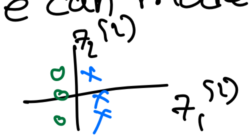
reason:

It makes weights, later or deeper more robust to changes to weights earlier layers of neural network, say in layer one.



BN helps model to learn distribution of hidden layer, so even though $a_i^{[l]}$ can change, we can model that.





BN reduces the problem of input value changing.

BN as regularization:-

→ each mini batch $X^{\{t\}}$ is scaled by mean/variance computed on just that mini batch.

→ This adds some noise to value $Z^{[l]}$ within that minibatch. $\rightarrow \tilde{Z}^{[l]}$

so, similar to dropout, it adds some noise to each hidden layers activation.

→ This has slight regularization effect.

minibatch : 64 — 512

more noise — less noise

more regularization — less regularization

dec7: BN at test time:

→ BN process one mini batch at time during training

→ But during test time, we may need to process one example at a time.

training:

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

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

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

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

Test:

μ, σ^2 : estimate using exponentially weighted average across mini batches.

in layer l:-

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

$\mu^{\{1\}[l]} \quad \mu^{\{2\}[l]} \quad \mu^{\{3\}[l]} \ldots$

$\sigma^{2\{1\}[l]} \quad \sigma^{2\{2\}[l]} \quad \sigma^{2\{3\}[l]} \ldots$

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

μ → weighted avg

σ^2 → weighted avg.

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

dec9: Softmax Regression:-

- cats, dogs, baby chicks, other
  1, 2, 3, 0

C = number of classes = $4 \in (0, 1, 2 \ldots C-1)$

$n^{[L]} = C = 4$ $\quad n^{[L]} = 4$

$\hat{y} = (n^{[L]}, 1) = (4, 1)$

- $P(\text{other}|x)$
- $P(\text{cat}|x)$
- $P(\text{dog}|x)$
- $P(\text{bc}|x)$

softmax layers:-

$$\underset{(4,1)}{z^{[L]}} = W^{[L]} a^{[L-1]} + b^{[L]}$$

Activation function:

$$\underset{(4,1)}{t} = e^{z^{[L]}}$$

$$\underset{(4,1)}{a^{[L]}} = \frac{e^{z^{[L]}}}{\sum_{j=1}^{4} t_i} \qquad a_i^{[L]} = \frac{t_i}{\sum_{i=1}^{4} t_i}$$

example:

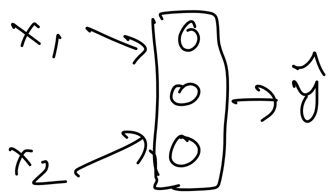
$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix} \qquad \sum_{i=1}^{4} t_i = 176.3$$

$$z^{[L]} \to a^{[L]}$$

$$\hat{y} = a^{[L]} = \frac{t}{176.3} = \begin{bmatrix} 148.4/176.3 \\ 7.4/176.3 \\ 0.4/176.3 \\ 20.1/176.3 \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

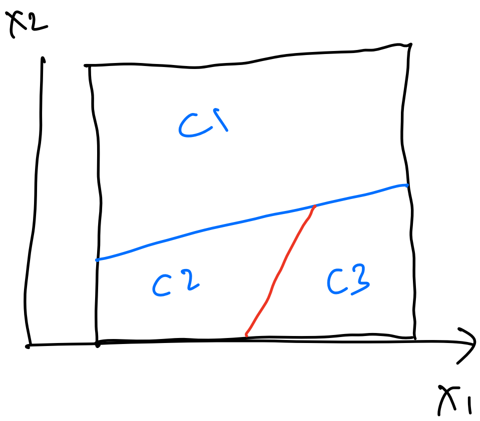
$$\underset{(4,1)}{a^{[L]}} = g^{[L]}\left(\underset{(4,1)}{z^{[L]}}\right)$$

softmax examples



$$z^{[l]} = W^{[l]} x + b^{[l]}$$

$$a^{[l]} = \hat{y} = g(z^{[l]}) \quad \text{(softmax layer)}$$



Recap: Training a softmax classifier:

$$z^{[l]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \xrightarrow{\text{softmax}} a^{[l]} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

hardmax

if $C=2$, softmax → logistic regression.

$$a^{[l]} = \begin{bmatrix} 0.842 \\ \cancel{0.158} \end{bmatrix}$$

Loss function:

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \hat{y} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$$

$$\downarrow \underset{\text{small}}{\mathcal{L}(\hat{y}, y)} = -\sum_{j=1}^{4} y_j \log \hat{y}_j = -y_2 \log \hat{y}_2 = -\log \hat{y}_2 \downarrow \text{small}$$

$\Rightarrow \hat{y}_2 \uparrow$ big as possible.

$$J(w^{[1]}, b^{[1]}, \dots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$$Y = [y^{(1)} \; y^{(2)} \dots y^{(m)}]_{(4,m)}$$

$$\hat{Y} = [\hat{y}^{(1)} \; \hat{y}^{(2)} \dots \hat{y}^{(m)}] \; (4,m)$$

GD:-

$$a^{[L]} = g^{[L]}(z^{[L]}) = \hat{y} \rightarrow \mathcal{L}(\hat{y}, y)$$

backprop:-

$$\underset{(4,1)}{dz^{[L]}} = \underset{(4,1)}{\hat{y}} - \underset{(4,1)}{y} \qquad dz = \frac{\partial J}{\partial z^{[L]}}$$

dec10:- Deep learning frameworks:

TensorFlow:

$$J(w) = w^2 - 10w + 25$$

$$J(w) = (w-5)^2$$

$w=5$ minimizes: $J(w)=0$

$$x(0)\,w^2 + x(1)\,w + x(2)$$