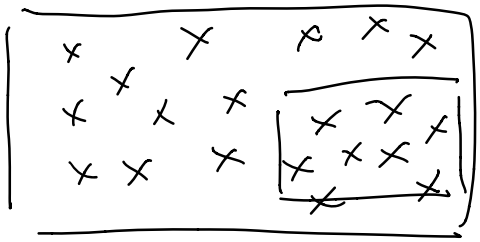
Tuning process

α (1), β (2), $\beta_1, \beta_2, \varepsilon$, # layers, # hidden units (2), learning rate decay (2), mini-batch size (2)

- Try random values: don't use a grid
- Coarse to fine



Use appropriate scale to pick up hyperparameters

eg. weighted smooth average

β search between $0.9 \sim 0.999$

$\Downarrow$

$[10^{-1} \sim 10^{-3}]$

```
np.power(10, 3 * np.random.randn())
```

batch normalization

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

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

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

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

learnable parameter

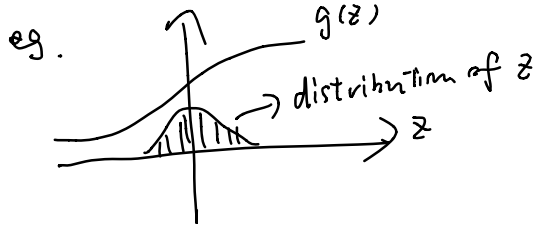
if $\gamma = \sqrt{\sigma^2+\varepsilon}$, $\beta = \mu$

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

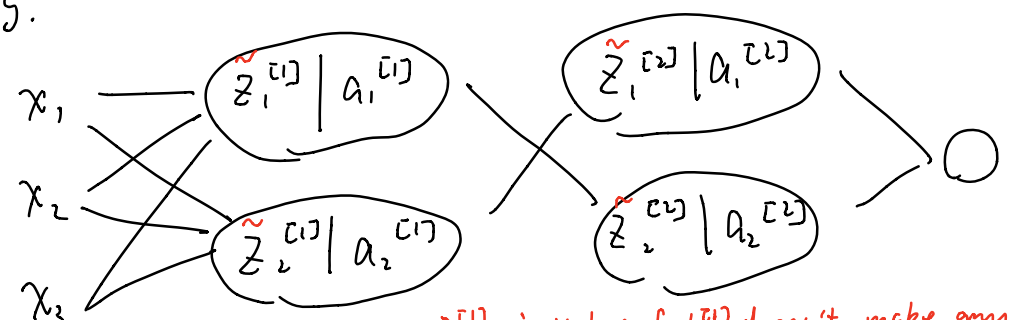
∴ 我们可以将 $\tilde{z}$ 理解为将 z 调整 mean 和 variance 之后得到的点

intuition: control the mean and variance of $z^{(i)}$, so that its distribution will fit in the activation function better

eg.



eg.



$$x \xrightarrow{w^{[1]}, \cancel{b^{[1]}}} z^{[1]} \xrightarrow{\text{normalize}} \tilde{z}^{[1]} \xrightarrow{\gamma^{[1]}, \beta^{[1]}} \tilde{z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]})$$

∵ we need to normalize $z^{[1]}$, ∴ value of $b^{[1]}$ doesn't make any difference

∴ set it to zero

- - - -

∴ we can tune β and γ just like other parameter

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

eg.

apply mini-batch with batch normalization

for $t = 1, \ldots$ # mini batches

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

in hidden layer, use $\tilde{z}^{[l]}$ to replace $z^{[l]}$

backward prop:

$dw^{[l]}$, ~~$db^{[l]}$~~, $d\beta^{[l]}$, $d\gamma^{[l]}$

update parameter:

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

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

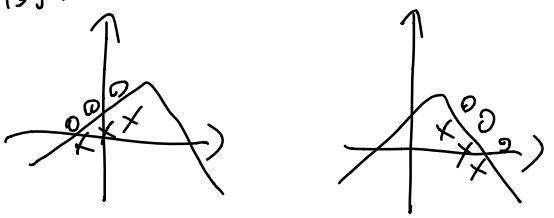
$\gamma^{[l]} = \ldots$

} can also try RMS prop, weighted momentum or Adam

Why Batch Norm work in training?

1. force to control mean and variance of z, thus make it easier to train. (standardize hidden layers)

反例:



hard to use the same model when distributions are different.

2. because every mini-batch has dif mean & variance, we batch-norm can add noise to training (so that we cannot rely just on

Thus it is similar to regularization (any nodes)

how to use batch norm at test time?

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

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

$x^{\{1\}}$ ... $x^{\{t\}}$

$\mu^{\{1\}[l]}$ $\mu^{\{t\}[l]}$ $\longrightarrow$ exponentially moving average $\mu = [\mu^{[1]} \cdots \mu^{[L]}]$

$\sigma^{\{1\}[l]}$ - - $\longrightarrow$ - -

} training (one batch at a time)

$$z^{(i)[l]}_{norm} = \frac{z^{(i)[l]} - \mu^{[l]}}{\sqrt{\sigma^{[l]^2} + \varepsilon}}, \quad \tilde{z}^{(i)[l]}_{norm} = \gamma z^{(i)[l]}_{norm} + \beta$$

} testing (one instance at a time)

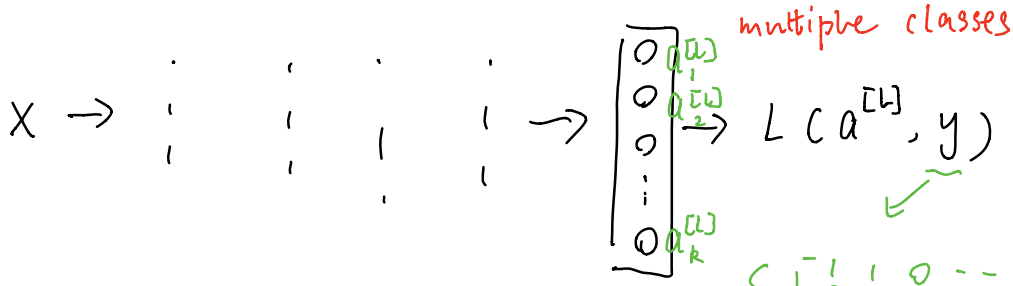
training 时每个 mini-batch 单独根据各自的 mean 和 variance 校正，在 testing 时，沿用之前得到的 γ 和 β，但由于是新数据，需要重新计算 $\hat{\mu}_{test}$ 和 $\hat{\sigma}^2_{test}$

① 若只有一个 sample，则用之前各个 mini-batch 的 $\mu^{[l]\{1\}} \cdots \mu^{[l]\{t\}}$ 计算 EWA 从而得到 $\hat{\mu}_{test}$，$\hat{\sigma}^2_{test}$ 同理

② 若有多个 samples，直接 $\hat{\mu}_{test} \leftarrow \hat{\mu}_{population}$，$\hat{\sigma}^2_{test} \leftarrow \hat{\sigma}^2_{population}$

Multi-Classification: soft max

softmax is a generalization of logistic regression to multiple classes



eg. "softmax" with 4 classes

L denotes last layer

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \rightarrow t^{[L]} = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

probability for each class $a^{[L]} = \frac{1}{\sum_{j=1}^{4} t_j^{[L]}} t^{[L]}$

"hardmax"

$$z^{[L]} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

loss function of softmax

$$L(a^{[L]}, y) = \sum_{i=1}^{m} \sum_{k=1}^{k} -y_k^{(m)} \log a_k^{[L](m)}$$

$$\downarrow$$

$$-\log a_k^{[L]}$$

backprop of softmax

$$dz^{[L]} = \hat{y} - y$$

Deeplearning Framework

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

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

```
Session.run(init)
Session.run(train, feed_dict{x: coefficients})
```