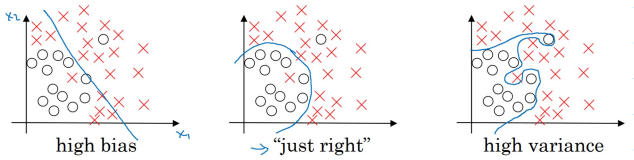
Train / Dev / Test sets

data | train | dev | test
→ Hold-out Cross Validation
→ Development set

Normal	60 % (6'000)	20 % (2'000)	20 % (2'000)
Big Data	99.5 % (10'000'000)	0.25 % (10'000)	0.25 % (10'000)

* dev & test set come from same distribution.



Train error :	1 %	15 %	15 %
Dev error :	11 %	16 %	30 %
	high v	high b	high v&b

Fixes →
- High bias: - Bigger Network, - Train Longer
- High varaince: - More Data, - Regularization

* Deep learning does not have the "b_v trade off"

Regularization

$$J(w,b) = \frac{1}{m}\sum_{i}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + L_1 + L_2$$

Logistic Reg →

$$L_1 = \frac{\lambda}{2m}\|w\|_1 = \frac{\lambda}{2m}\sum_{i}^{n_x}|w_j|$$

$$L_2 = \frac{\lambda}{2m}\|w\|_2^2 = \frac{\lambda}{2m}\sum_{i}^{n_x} w_j^2 = \frac{\lambda}{2m} w^T w$$

NN →

$$L_2 = \frac{\lambda}{2m}\sum_{i}^{L}\|w^{[l]}\|_F = \frac{\lambda}{2m}\sum_{i}^{L}\sum_{i}^{n^{[l]}}\sum_{i}^{n^{[l-1]}}(w_{ij}^{[l]})^2$$

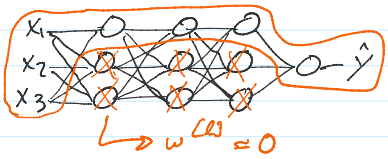
→ Frobenius norm

$$\Rightarrow dw^{[l]} = (\text{back pop}) + \frac{\lambda}{m} w^{[l]}$$

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

weigh decay →

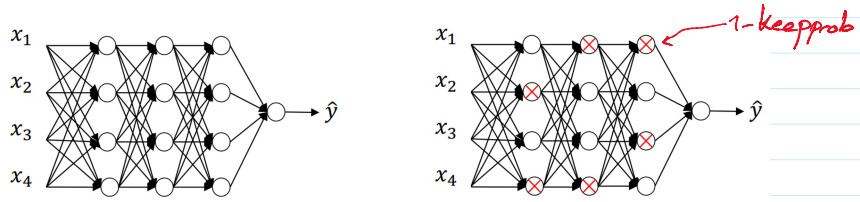
$$w^{[l]} = \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha(\text{back pop}) \qquad \left(1 - \frac{\alpha\lambda}{m} < 1\right)$$



→ $w^{[l]} \approx 0$

→ $\lambda \uparrow$ $w^{[l]} \downarrow$ $z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$ → makes layers linear

Dropout Regularization



d3 = np.random(size a3) < keepprob

a3 = np.multiply(a3, d3)

a3 /= keepprob

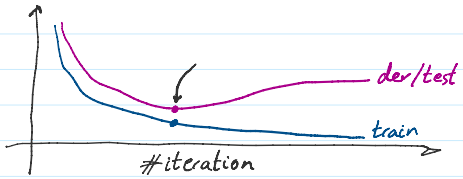
Inverted Dropout

*not during test

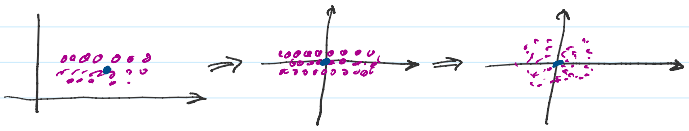
*Can't rely on any One Feature

Data Augmentation

Early Stopping

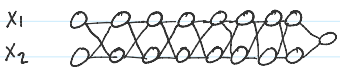


Normalizing Training Sets



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

Vanishing / Exploding gradients problem



$\hat{y} = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots w^{[1]}$ → 1.5^L, 0.5^L

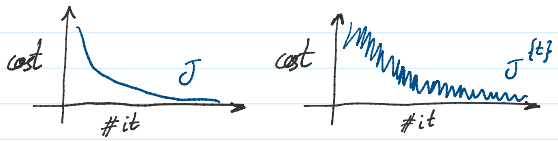
Fix:

$w^{[l]} = np.random() * np.sqrt\left(\frac{?}{n^{[l-1]}}\right)$

* $\left(\frac{1}{n^{[l-1]}}\right)$ Xavier

* $\left(\frac{2}{n^{[l-1]} + n^{[l]}}\right)$

Mini-Batch

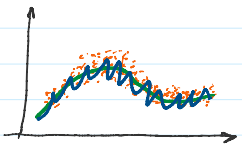


* size = m ⟶ Batch GD Too long per #iter
* size = 1 ⟶ Stochastic GD Loose speed from vec-
* size is usualy 2^k = 64, 128, 256, ...

Exponentially Weighted Averages

$v_t = \beta v_{t-1} + (1-\beta)\theta_t$ $\beta = 0.9$ 10

$\beta = 0.98$ 50

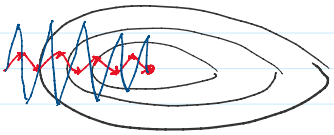


* average over $\frac{1}{1-\beta}$

Bias Correction

$\longrightarrow v_t \Rightarrow \frac{v_t}{1-\beta^t}$

Gradient Descent with Momentum



On #iter t:

$v_{dw} = \beta v_{dw} + (1-\beta) dw$

$v_{db} = \beta v_{db} + (1-\beta) db$

$w -= \alpha v_{dw}$ $b -= \alpha v_{db}$

RMSprop

On #iter t:

$S_{dw} = \beta_2 S_{dw} + (1-\beta_2) dw^2$

$S_{db} = \beta_2 S_{db} + (1-\beta_2) db^2$

$w -= \frac{\alpha dw}{\sqrt{S_{dw}+\varepsilon}}$ $b -= \frac{\alpha db}{\sqrt{S_{db}+\varepsilon}}$

Adam Optimization Algorithm

On #iter t:

$v_{dw} = \beta_1 v_{dw} + (1-\beta_1) dw$ $v_{db} =$

$S_{dw} = \beta_2 S_{dw} + (1-\beta_2) dw^2$ $S_{db} =$

$v_{dw}^{corr} = \frac{v_{dw}}{1-\beta_1^t}$

α needs to be tune

$$v_{dw}^{corr} = \frac{v_{dw}}{1-\beta_1^t}$$

$$s_{dw}^{corr} = \frac{s_{dw}}{1-\beta_2^t}$$

$$w := \frac{\alpha\, v_{dw}^{corr}}{\sqrt{s_{dw}^{corr}} + \varepsilon} \qquad b := \frac{\alpha\, v_{db}^{corr}}{\sqrt{s_{db}^{corr}} + \varepsilon}$$

α needs to be tune

$\beta_1 = 0.9$

$\beta_2 = 0.999$

$\varepsilon = 10^{-8}$

Learning Rate Decay

① $\alpha = \frac{1}{1 + decay * epoch}\alpha_0$

② $\alpha = 0.95^{epoch}\alpha_0$

③ $\alpha = \frac{k}{\sqrt{epoch}}\alpha_0$

④ 

staircase

Tuning Process

1- α

2- β

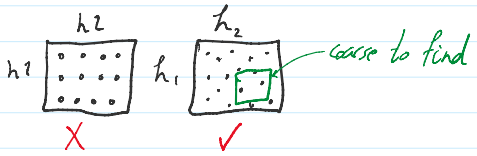
- hidden layers
- mini batch size

3- Layers

- learning rate decay

4- $\beta_1, \beta_2, \varepsilon$

* Use random values for hyper parameters



* babysitting (Panda) vs many in parallel (Caviar)