Dropout regularization

x_1

x_2

x_3

x_4

y
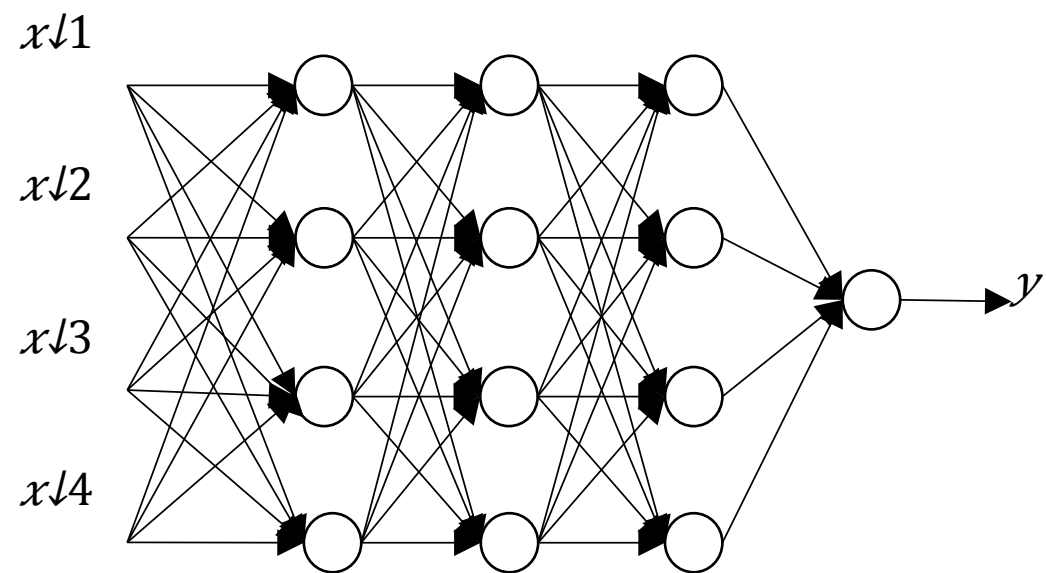
0.5   0.5   0.5

Implementing dropout ("Inverted dropout")

Illustrate with layer $l=3$.   keep-prob $= \dfrac{0.8}{\cancel{t}}$ 0.2

$\rightarrow$ $\boxed{d3}$ = np.random.rand(a3.shape[0], a3.shape[1]) < keep-prob

$a3$ = np.multiply(a3, d3)   # a3 *= d3.

$\rightarrow$ $\boxed{a3 \mathrel{/}= \cancel{0.8} \; \text{keep-prob}}$ $\leftarrow$

50 units $\leadsto$ 10 units shut off

$z^{[4]} = W^{[4]} \cdot \dfrac{a^{[3]}}{\uparrow} + b^{[4]}$

J reduced by 20%.   Test

$\mathrel{/}= \underline{0.8}$

Andrew Ng

Making predictions at test time

$a^{[0]} = X$

No drop out

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = \cdots$

$\downarrow$

$\hat{y}$

$/ = keep\text{-}prob$

Andrew Ng

deeplearning.ai

Regularizing your neural network

Understanding dropout

Why does drop-out work?

Intuition: Can't rely on any one feature, so have to spread out weights. $\leadsto$ Shrink weights. L_2

$\rightarrow$ Computer Vision

Downside: J

#iteration

keep-prob = 1

Andrew Ng