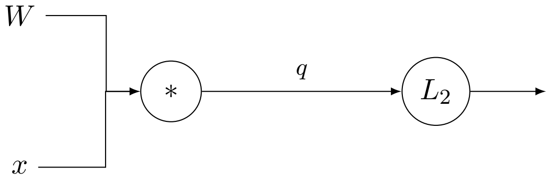
Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$

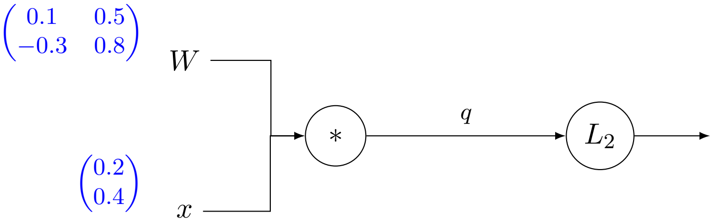


$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$

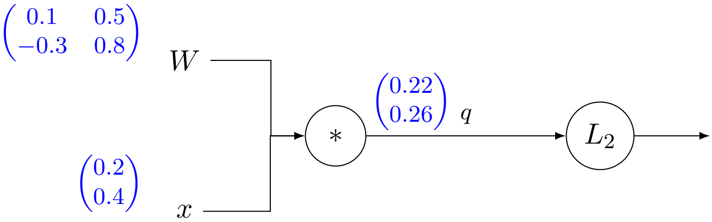


$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

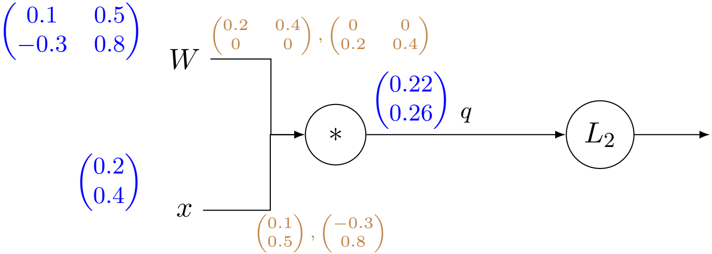
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \delta_{i,k} x_j$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

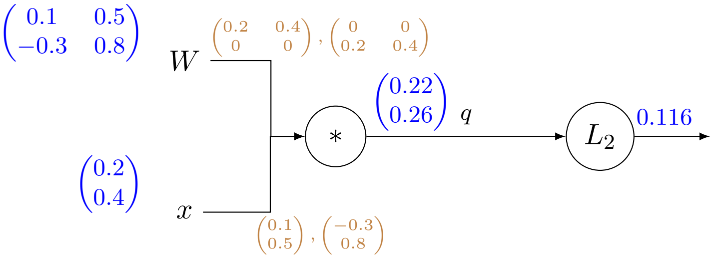
$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \delta_{i,k} x_j$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$

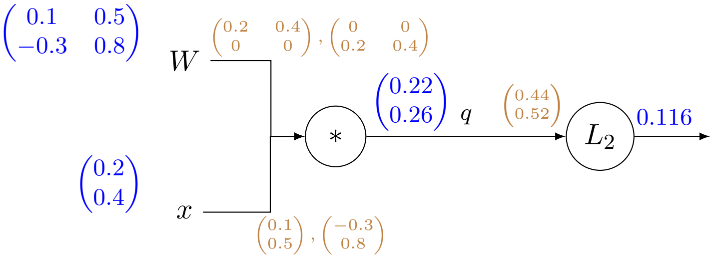


$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix} \qquad \frac{\partial f}{\partial q_i} = 2q_i$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



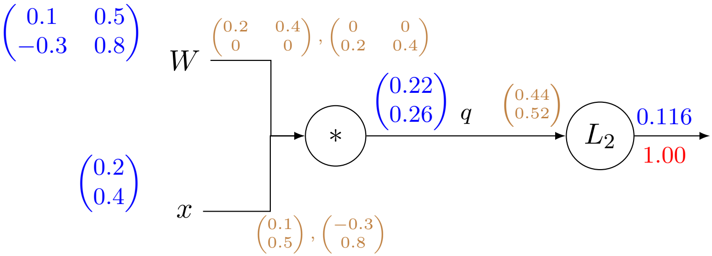
$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



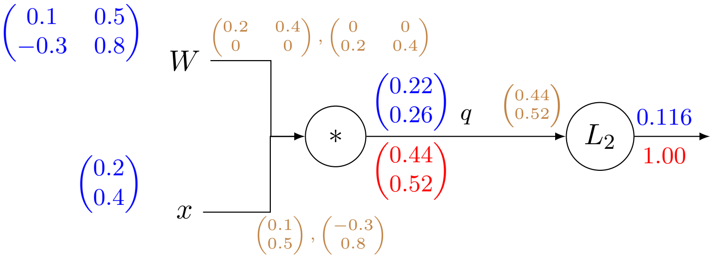
$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 1.00 \cdot \frac{\partial f}{\partial q_i}$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



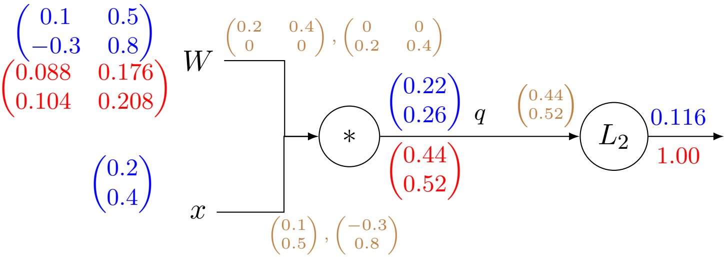
$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial f}{\partial q_i} = 1.00 \cdot \frac{\partial f}{\partial q_i}$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



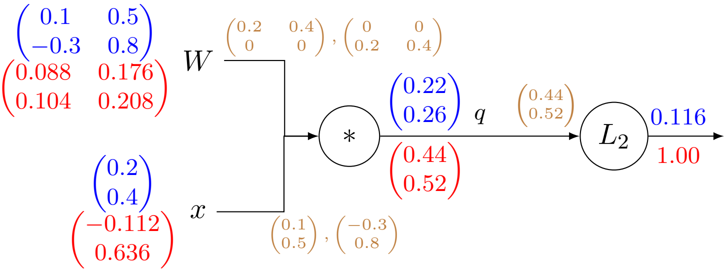
$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$\frac{\partial f}{\partial W_{i,j}} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial W_{i,j}} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial W_{i,j}}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Handling vector variables

A vectorized example: $L = \|q - \tilde{q}\|^2 = \|Wx - \tilde{q}\|^2$



$$q = Wx = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial x_i} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial x_i}$$

$$f(q) = \|q\|^2 = q_1^2 + \cdots + q_n^2$$

Weight initialization

- First idea: **Small random numbers**

(gaussian with zero mean and 1e-2 standard deviation)

```
W = 0.01* np.random.randn(D,H)
```

Weight initialization

- First idea: **Small random numbers** (gaussian with zero mean and 1e-2 standard deviation)

```
W = 0.01* np.random.randn(D,H)
```

Works ~okay for small networks, but can lead to non-homogeneous distributions of activations across the layers of a network.

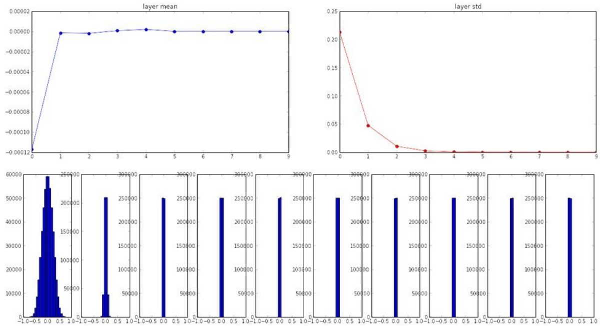
Weight initialization

Let's look at some activation statistics

- 10 layers
- 500 neurons per layer
- $tanh(\cdot)$ for activation
- $W = 0.01 * \text{np.random.randn(fan\_in, fan\_out)}$ as described in the last slide

Weight initialization

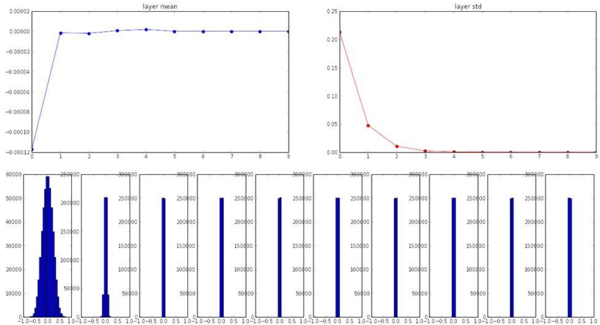
```
input layer had mean 0.000927 and std 0.998388
hidden layer 1 had mean -0.000117 and std 0.213081
hidden layer 2 had mean -0.000001 and std 0.047551
hidden layer 3 had mean -0.000002 and std 0.010630
hidden layer 4 had mean 0.000001 and std 0.002378
hidden layer 5 had mean 0.000002 and std 0.000532
hidden layer 6 had mean -0.000000 and std 0.000119
hidden layer 7 had mean 0.000000 and std 0.000026
hidden layer 8 had mean -0.000000 and std 0.000006
hidden layer 9 had mean 0.000000 and std 0.000001
hidden layer 10 had mean -0.000000 and std 0.000000
```



Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 5 - 57 20 Jan 2016

Weight initialization

```
input layer had mean 0.000927 and std 0.998388
hidden layer 1 had mean -0.000117 and std 0.213081
hidden layer 2 had mean -0.000001 and std 0.047551
hidden layer 3 had mean -0.000002 and std 0.010630
hidden layer 4 had mean 0.000001 and std 0.002378
hidden layer 5 had mean 0.000002 and std 0.000532
hidden layer 6 had mean -0.000000 and std 0.000119
hidden layer 7 had mean 0.000000 and std 0.000026
hidden layer 8 had mean -0.000000 and std 0.000006
hidden layer 9 had mean 0.000000 and std 0.000001
hidden layer 10 had mean -0.000000 and std 0.000000
```



All activations become zero!

Q: think about the backward pass. What do the gradients look like?

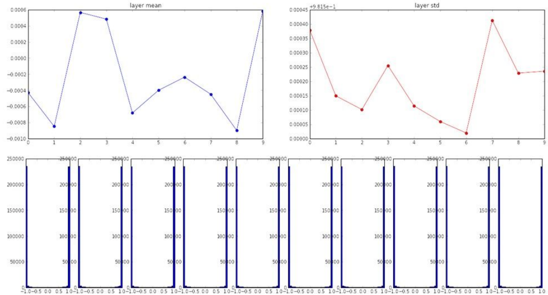
Hint: think about backward pass for a W*X gate.

Weight initialization

```
W = np.random.randn(fan_in, fan_out) * 1.0 # layer initialization
```

```
input layer had mean 0.001800 and std 1.001311
hidden layer 1 had mean -0.000430 and std 0.981879
hidden layer 2 had mean -0.000849 and std 0.981649
hidden layer 3 had mean 0.000566 and std 0.981601
hidden layer 4 had mean 0.000483 and std 0.981755
hidden layer 5 had mean -0.000682 and std 0.981614
hidden layer 6 had mean -0.000401 and std 0.981560
hidden layer 7 had mean -0.000237 and std 0.981520
hidden layer 8 had mean -0.000448 and std 0.981913
hidden layer 9 had mean -0.000899 and std 0.981728
hidden layer 10 had mean 0.000584 and std 0.981736
```

*1.0 instead of *0.01



Almost all neurons completely saturated, either -1 and 1. Gradients will be all zero.

Variance calibration for linear layer

Assume linear activation and zero-mean weights and inputs. And number of inputs is n. Then,

$$\mathrm{Var}(y) = \mathrm{Var}\left(\sum_i^n w_i x_i\right) = \sum_i^n \mathrm{Var}(w_i x_i)$$

Variance calibration for linear layer

Assume linear activation and zero-mean weights and inputs. And number of inputs is n. Then,

$$\begin{aligned} \mathrm{Var}(y) &= \mathrm{Var}\left(\sum_i^n w_i x_i\right) = \sum_i^n \mathrm{Var}(w_i x_i) \\ &= \sum_i^n [E(w_i)]^2 \mathrm{Var}(x_i) + E[(x_i)]^2 \mathrm{Var}(w_i) + \mathrm{Var}(x_i)\mathrm{Var}(w_i) \end{aligned}$$

$Var(XY) = E[X]^2Var(Y) + E[Y]^2Var(X) + Var(X)Var(Y)$

$$Var(XY) = E[(XY)^2] - E[XY]^2$$

$Var(XY) = E[X]^2 Var(Y) + E[Y]^2 Var(X) + Var(X)Var(Y)$

$$
\begin{aligned}
Var(XY) &= E[(XY)^2] - E[XY]^2 \\
&= E[X^2]E[Y^2] - E[X]^2 E[Y]^2
\end{aligned}
$$

$Var(XY) = E[X]^2Var(Y) + E[Y]^2Var(X) + Var(X)Var(Y)$

$$
\begin{aligned}
Var(XY) &= E[(XY)^2] - E[XY]^2 \\
&= E[X^2]E[Y^2] - E[X]^2E[Y]^2
\end{aligned}
$$

$$
\begin{aligned}
&Var(X)Var(Y) \\
= &(E[X^2] - E[X]^2)(E[Y^2] - E[Y]^2)
\end{aligned}
$$

$Var(XY) = E[X]^2Var(Y) + E[Y]^2Var(X) + Var(X)Var(Y)$

$$\begin{aligned} Var(XY) &= E[(XY)^2] - E[XY]^2 \\ &= E[X^2]E[Y^2] - E[X]^2E[Y]^2 \end{aligned}$$

$$\begin{aligned} & Var(X)Var(Y) \\ =\ & (E[X^2] - E[X]^2)(E[Y^2] - E[Y]^2) \\ =\ & E[X^2]E[Y^2] - E[X]^2E[Y^2] - E[X^2]E[Y]^2 + E[X]^2E[Y]^2 \end{aligned}$$

$Var(XY) = E[X]^2Var(Y) + E[Y]^2Var(X) + Var(X)Var(Y)$

$$\begin{aligned} Var(XY) &= E[(XY)^2] - E[XY]^2 \\ &= E[X^2]E[Y^2] - E[X]^2E[Y]^2 \end{aligned}$$

$$\begin{aligned} &Var(X)Var(Y) \\ &= (E[X^2] - E[X]^2)(E[Y^2] - E[Y]^2) \\ &= E[X^2]E[Y^2] - E[X]^2E[Y^2] - E[X^2]E[Y]^2 + E[X]^2E[Y]^2 \\ &= E[X^2]E[Y^2] - E[X]^2(E[Y^2] - E[Y]^2) \\ &\quad E[Y]^2(E[X^2] - E[X]^2) - E[X]^2E[Y]^2 \end{aligned}$$

$Var(XY) = E[X]^2 Var(Y) + E[Y]^2 Var(X) + Var(X)Var(Y)$

$$
\begin{aligned}
Var(XY) &= E[(XY)^2] - E[XY]^2 \\
&= E[X^2]E[Y^2] - E[X]^2E[Y]^2
\end{aligned}
$$

$$
\begin{aligned}
&Var(X)Var(Y) \\
&= (E[X^2] - E[X]^2)(E[Y^2] - E[Y]^2) \\
&= E[X^2]E[Y^2] - E[X]^2E[Y^2] - E[X^2]E[Y]^2 + E[X]^2E[Y]^2 \\
&= E[X^2]E[Y^2] - E[X]^2(E[Y^2] - E[Y]^2) \\
&\quad E[Y]^2(E[X^2] - E[X]^2) - E[X]^2E[Y]^2 \\
&= Var(XY) - E[X]^2Var(Y) - E[Y]^2Var(X)
\end{aligned}
$$

Variance calibration for linear layer

Assume linear activation and zero-mean weights and inputs. And number of inputs is n. Then,

$$\begin{aligned}\mathrm{Var}(y) &= \mathrm{Var}\left(\sum_i^n w_i x_i\right) = \sum_i^n \mathrm{Var}(w_i x_i) \\ &= \sum_i^n E[w_i]^2 \mathrm{Var}(x_i) + E[x_i]^2 \mathrm{Var}(w_i) + \mathrm{Var}(x_i)\mathrm{Var}(w_i)\end{aligned}$$

Variance calibration for linear layer

Assume linear activation and zero-mean weights and inputs. And number of inputs is n. Then,

$$\begin{aligned}
\mathrm{Var}(y) &= \mathrm{Var}\left(\sum_i^n w_i x_i\right) = \sum_i^n \mathrm{Var}(w_i x_i) \\
&= \sum_i^n E[w_i]^2 \mathrm{Var}(x_i) + E[x_i]^2 \mathrm{Var}(w_i) + \mathrm{Var}(x_i)\mathrm{Var}(w_i) \\
&= \sum_i^n \mathrm{Var}(x_i)\mathrm{Var}(w_i) \\
&= (n\mathrm{Var}(w))\,\mathrm{Var}(x)
\end{aligned}$$

Variance calibration for linear layer

Assume linear activation and zero-mean weights and inputs. And number of inputs is n. Then,

$$\begin{aligned}
\mathrm{Var}(y) &= \mathrm{Var}\left(\sum_i^n w_i x_i\right) = \sum_i^n \mathrm{Var}(w_i x_i) \\
&= \sum_i^n E[w_i]^2 \mathrm{Var}(x_i) + E[x_i]^2 \mathrm{Var}(w_i) + \mathrm{Var}(x_i)\mathrm{Var}(w_i) \\
&= \sum_i^n \mathrm{Var}(x_i)\mathrm{Var}(w_i) \\
&= (n\mathrm{Var}(w))\,\mathrm{Var}(x)
\end{aligned}$$

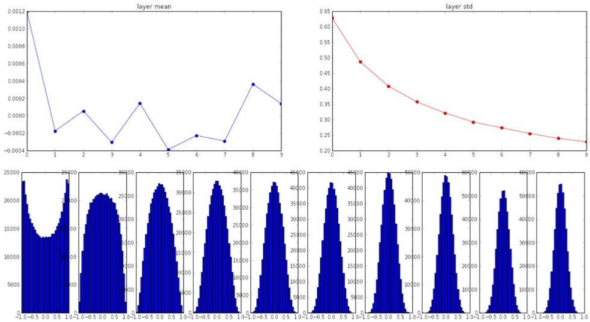
Thus, output will have same variance as input if $n\mathrm{Var}(w) = 1$. This is known as Xavier weight initialization

Weight initialization

```
input layer had mean 0.001800 and std 1.001311
hidden layer 1 had mean 0.001198 and std 0.627953
hidden layer 2 had mean -0.000175 and std 0.486051
hidden layer 3 had mean 0.000055 and std 0.407723
hidden layer 4 had mean -0.000306 and std 0.357108
hidden layer 5 had mean 0.000142 and std 0.320917
hidden layer 6 had mean -0.000389 and std 0.292116
hidden layer 7 had mean -0.000228 and std 0.273387
hidden layer 8 had mean -0.000291 and std 0.254935
hidden layer 9 had mean 0.000361 and std 0.239266
hidden layer 10 had mean 0.000139 and std 0.228008
```

```
W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in) # layer initialization
```

"Xavier initialization"
[Glorot et al., 2010]



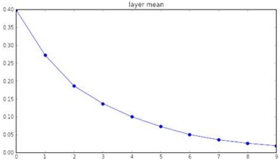
Reasonable initialization.
(Mathematical derivation assumes linear activations)

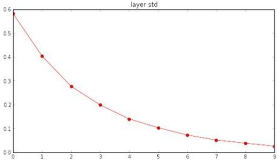
Weight initialization

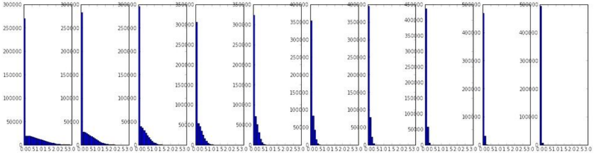
```
input layer had mean 0.000501 and std 0.999444
hidden layer 1 had mean 0.398623 and std 0.582273
hidden layer 2 had mean 0.272352 and std 0.403795
hidden layer 3 had mean 0.186076 and std 0.276912
hidden layer 4 had mean 0.136442 and std 0.198685
hidden layer 5 had mean 0.099568 and std 0.140299
hidden layer 6 had mean 0.072234 and std 0.103280
hidden layer 7 had mean 0.049775 and std 0.072748
hidden layer 8 had mean 0.035138 and std 0.051572
hidden layer 9 had mean 0.025404 and std 0.038583
hidden layer 10 had mean 0.018408 and std 0.026076
```

```
W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in) # layer initialization
```

but when using the ReLU nonlinearity it breaks.

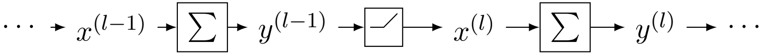






Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 5 - 61 20 Jan 2016

Variance calibration for ReLU

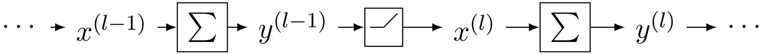


Note that it doesn't work when the activation layer is ReLU. But...[1]

$$\mathrm{Var}(y^{(l)}) = \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right)$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU

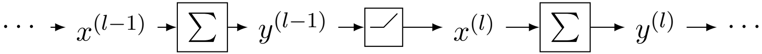


Note that it doesn't work when the activation layer is ReLU. But...[1]

$$\mathrm{Var}(y^{(l)}) = \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) = \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)})$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU



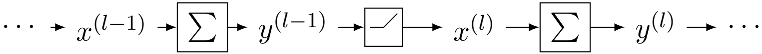
Note that it doesn't work when the activation layer is ReLU. But...[1]

$$\mathrm{Var}(y^{(l)}) = \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) = \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)})$$

$$= nE[w^{(l)}]^2\mathrm{Var}(x^{(l)}) + nE[x^{(l)}]^2\mathrm{Var}(w^{(l)}) + \mathrm{nVar}(x^{(l)})\mathrm{Var}(w^{(l)})$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU

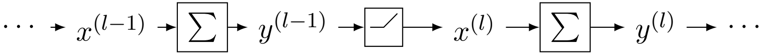


Note that it doesn't work when the activation layer is ReLU. But...[1]

$$\begin{aligned}\mathrm{Var}(y^{(l)}) &= \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) = \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)}) \\ &= nE[w^{(l)}]^2 \mathrm{Var}(x^{(l)}) + nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + \mathrm{nVar}(x^{(l)})\mathrm{Var}(w^{(l)}) \\ &= nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + n\mathrm{Var}(x^{(l)})\mathrm{Var}(w^{(l)})\end{aligned}$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU

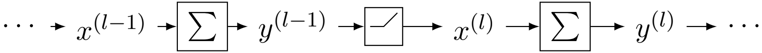


Note that it doesn't work when the activation layer is ReLU. But...[1]

$$
\begin{aligned}
\mathrm{Var}(y^{(l)}) = \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) &= \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)}) \\
&= nE[w^{(l)}]^2 \mathrm{Var}(x^{(l)}) + nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + \mathrm{nVar}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + n\mathrm{Var}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[(x^{(l)})^2]\mathrm{Var}(w^{(l)})
\end{aligned}
$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU

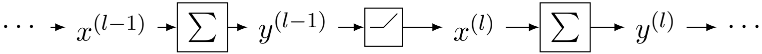


Note that it doesn't work when the activation layer is ReLU. But...[1]

$$
\begin{aligned}
\mathrm{Var}(y^{(l)}) &= \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) = \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)}) \\
&= nE[w^{(l)}]^2 \mathrm{Var}(x^{(l)}) + nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + \mathrm{nVar}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + n\mathrm{Var}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[(x^{(l)})^2]\mathrm{Var}(w^{(l)}) \\
&= n(\mathrm{Var}(y^{(l-1)})/2)\mathrm{Var}(w^{(l)}) = \left(\frac{n}{2}\mathrm{Var}(w^{(l)})\right)\mathrm{Var}(y^{(l-1)})
\end{aligned}
$$

[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Variance calibration for ReLU



Note that it doesn't work when the activation layer is ReLU. But...[1]

$$
\begin{aligned}
\mathrm{Var}(y^{(l)}) &= \mathrm{Var}\left(\sum_i^n w_i^{(l)} x_i^{(l)}\right) = \sum_i^n \mathrm{Var}(w_i^{(l)} x_i^{(l)}) = n\mathrm{Var}(w^{(l)} x^{(l)}) \\
&= nE[w^{(l)}]^2 \mathrm{Var}(x^{(l)}) + nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + \mathrm{nVar}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[x^{(l)}]^2 \mathrm{Var}(w^{(l)}) + n\mathrm{Var}(x^{(l)})\mathrm{Var}(w^{(l)}) \\
&= nE[(x^{(l)})^2]\mathrm{Var}(w^{(l)}) \\
&= n(\mathrm{Var}(y^{(l-1)})/2)\mathrm{Var}(w^{(l)}) = \left(\frac{n}{2}\mathrm{Var}(w^{(l)})\right)\mathrm{Var}(y^{(l-1)})
\end{aligned}
$$

Variance of y conserved across a layer if $\frac{n}{2}\mathrm{Var}(w) = 1$

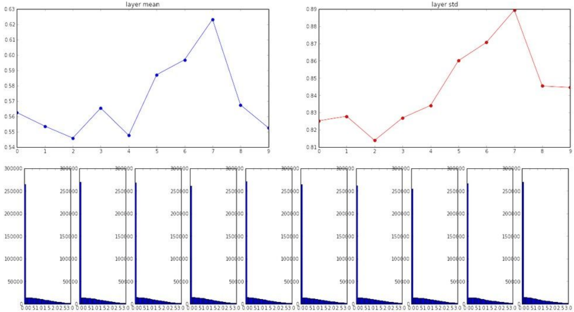
[1] Note that $y^{(l)}$ now denotes the sum of input before going through the activation function.

Weight initialization

```
input layer had mean 0.000501 and std 0.999444
hidden layer 1 had mean 0.562488 and std 0.825232
hidden layer 2 had mean 0.553614 and std 0.827835
hidden layer 3 had mean 0.545867 and std 0.813855
hidden layer 4 had mean 0.565396 and std 0.826902
hidden layer 5 had mean 0.547678 and std 0.834092
hidden layer 6 had mean 0.587103 and std 0.860035
hidden layer 7 had mean 0.596867 and std 0.870610
hidden layer 8 had mean 0.623214 and std 0.889348
hidden layer 9 had mean 0.567498 and std 0.845357
hidden layer 10 had mean 0.552531 and std 0.844523
```

```
W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in/2) # layer initialization
```

He et al., 2015
(note additional /2)



Weight initialization

```
input layer had mean 0.000501 and std 0.999444
hidden layer 1 had mean 0.562488 and std 0.825232
hidden layer 2 had mean 0.553614 and std 0.827835
hidden layer 3 had mean 0.545867 and std 0.813855
hidden layer 4 had mean 0.565396 and std 0.826902
hidden layer 5 had mean 0.547678 and std 0.834092
hidden layer 6 had mean 0.587103 and std 0.860035
hidden layer 7 had mean 0.596867 and std 0.870610
hidden layer 8 had mean 0.623214 and std 0.889348
hidden layer 9 had mean 0.567498 and std 0.845357
hidden layer 10 had mean 0.552531 and std 0.844523
```

```
W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in/2) # layer initialization
```

He et al., 2015
(note additional /2)