Deep Learning

Vazgen Mikayelyan

YSU, Krisp

October 3, 2019

Outline

Back-Propagation

Question: How to calculate the derivative of the function $\sin x^2$?

Back-Propagation

Question: How to calculate the derivative of the function $\sin x^2$?

Theorem 1

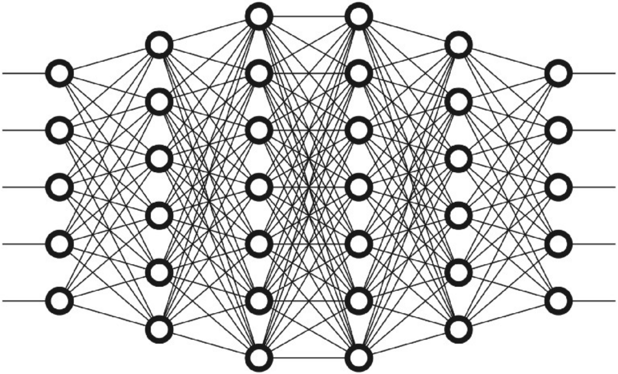
Given n functions $f_1, \dots, f_n$ with the composite function

$$f = f_1 \circ (f_2 \circ \cdots (f_{n-1} \circ f_n)),$$

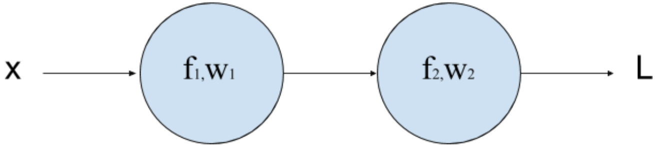
if each function f_i is differentiable at its immediate input, then the composite function is also differentiable by the repeated application of Chain Rule, where the derivative is

$$\frac{df}{dx} = \frac{df_1}{df_2}\frac{df_2}{df_3} \cdots \frac{df_n}{dx}.$$

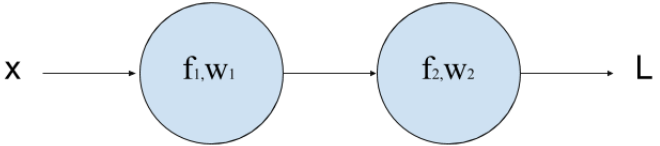
Back-Propagation



Back-Propagation



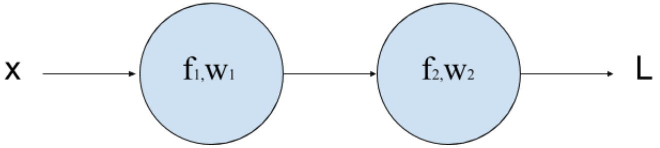
Back-Propagation



In this case we have the following function

$$L(w_1, w_2) = (f_2(w_2 f_1(w_1 x)) - y)^2$$

Back-Propagation

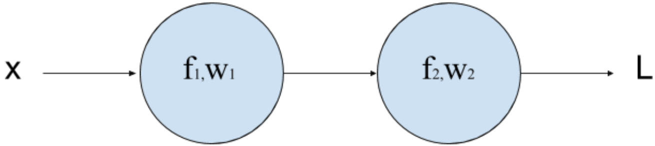


In this case we have the following function

$$L(w_1, w_2) = \left(f_2\left(w_2 f_1\left(w_1 x\right)\right) - y\right)^2$$

We have to calculate the derivatives $\dfrac{\partial L}{\partial w_1}$ and $\dfrac{\partial L}{\partial w_2}$:

Back-Propagation



In this case we have the following function

$$L\left(w_1, w_2\right) = \left(f_2\left(w_2 f_1\left(w_1 x\right)\right) - y\right)^2$$

We have to calculate the derivatives $\dfrac{\partial L}{\partial w_1}$ and $\dfrac{\partial L}{\partial w_2}$:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial\left(w_2 f_1\right)} \frac{\partial\left(w_2 f_1\right)}{\partial w_2},$$

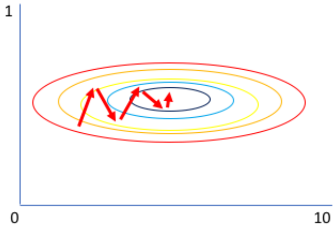
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial\left(w_2 f_1\right)} \frac{\partial\left(w_2 f_1\right)}{\partial f_1} \frac{\partial\left(f_1\right)}{\partial\left(w_1 x\right)} \frac{\partial\left(w_1 x\right)}{\partial w_1}.$$

Outline

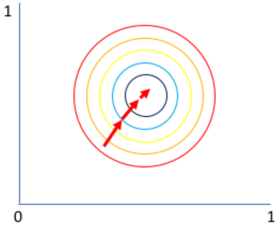
Data Normalization

Data Normalization

Why normalize?



Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

Standard Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

Standard Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Standard Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Denote $\mu^j = \frac{1}{n}\sum_{i=1}^n x_i^j$, $\sigma^j = \sqrt{\frac{1}{n}\sum_{i=1}^n \left(x_i^j - \mu^j\right)^2}$

Standard Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Denote $\mu^j = \frac{1}{n}\sum_{i=1}^{n} x_i^j$, $\sigma^j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i^j - \mu^j\right)^2}$ and let

$$z_i^j = \frac{x_i^j - \mu^j}{\sigma^j}.$$

Standard Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Denote $\mu^j = \frac{1}{n}\sum_{i=1}^n x_i^j$, $\sigma^j = \sqrt{\frac{1}{n}\sum_{i=1}^n \left(x_i^j - \mu^j\right)^2}$ and let

$$z_i^j = \frac{x_i^j - \mu^j}{\sigma^j}.$$

Our new data will be $(z_i, y_i)_{i=1}^n$, $z_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$.

Min-Max Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Min-Max Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Let

$$z_i^j = \frac{x_i^j - \min_i x_i^j}{\max_i x_i^j - \min_i x_i^j}$$

Min-Max Normalization

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$ be our training data:

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^k \\ x_2^1 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^k \end{bmatrix}.$$

Let

$$z_i^j = \frac{x_i^j - \min_i x_i^j}{\max_i x_i^j - \min_i x_i^j}$$

Our new data will be $(z_i, y_i)_{i=1}^n$, $z_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^m$.

Outline

Random Initialization

- Can we initialize all weights of linear/logistic regression with zeros?

Random Initialization

- Can we initialize all weights of linear/logistic regression with zeros?
  Answer: **Yes**

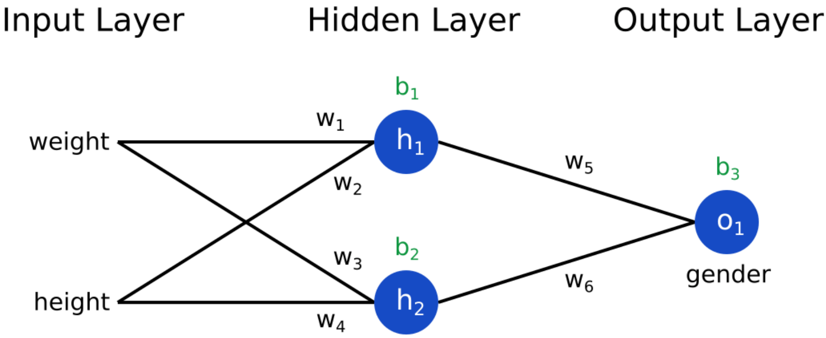
Random Initialization

- Can we initialize all weights of linear/logistic regression with zeros?
  Answer: **Yes**
- Can we initialize all weights of neural networks with zeros?

Random Initialization

- Can we initialize all weights of linear/logistic regression with zeros?
  Answer: **Yes**
- Can we initialize all weights of neural networks with zeros?
  Answer: **No**

Random Initialization



Random Initialization

$$h_1 = f\left(w_1 x_1 + w_2 x_2 + b_1\right), \quad h_2 = f\left(w_3 x_1 + w_4 x_2 + b_2\right)$$

$$o_1 = g\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$L\left(w\right) = L\left(w_1, \ldots, w_6\right) = \left(o_1 - y\right)^2$$

Random Initialization

$$h_1 = f\left(w_1 x_1 + w_2 x_2 + b_1\right), \quad h_2 = f\left(w_3 x_1 + w_4 x_2 + b_2\right)$$

$$o_1 = g\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$L\left(w\right) = L\left(w_1, \ldots, w_6\right) = \left(o_1 - y\right)^2$$

Let $w_1^0 = w_3^0 = c_1$, $w_2^0 = w_4^0 = c_2$, $w_5^0 = w_6^0 = c_3$ and $b_1^0 = b_2^0 = c_4$.

Random Initialization

$$h_1 = f\left(w_1 x_1 + w_2 x_2 + b_1\right), \quad h_2 = f\left(w_3 x_1 + w_4 x_2 + b_2\right)$$

$$o_1 = g\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$L\left(w\right) = L\left(w_1, \dots, w_6\right) = \left(o_1 - y\right)^2$$

Let $w_1^0 = w_3^0 = c_1$, $w_2^0 = w_4^0 = c_2$, $w_5^0 = w_6^0 = c_3$ and $b_1^0 = b_2^0 = c_4$.
During the optimization we need to do the following steps

$$w_i^1 = w_i^0 - \alpha \frac{\partial L}{\partial w_i}\left(w^0\right), i = 1, \dots, 6,$$

where $w^0 = \left(w_1^0, \dots, w_6^0\right)$.

Random Initialization

$$h_1 = f\left(w_1 x_1 + w_2 x_2 + b_1\right), \quad h_2 = f\left(w_3 x_1 + w_4 x_2 + b_2\right)$$

$$o_1 = g\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$L\left(w\right) = L\left(w_1, \ldots, w_6\right) = \left(o_1 - y\right)^2$$

Let $w_1^0 = w_3^0 = c_1$, $w_2^0 = w_4^0 = c_2$, $w_5^0 = w_6^0 = c_3$ and $b_1^0 = b_2^0 = c_4$. During the optimization we need to do the following steps

$$w_i^1 = w_i^0 - \alpha \frac{\partial L}{\partial w_i}\left(w^0\right), i = 1, \ldots, 6,$$

where $w^0 = \left(w_1^0, \ldots, w_6^0\right)$.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_1$$

Random Initialization

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_1$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_2,$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_1,$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_2.$$

Random Initialization

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_1$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_2,$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_1,$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_2.$$

So we see that $w_1^1 = w_3^1 = const$, $w_2^1 = w_4^1 = const$.

Random Initialization

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_1$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_5 \cdot f'\left(w_1 x_1 + w_2 x_2 + b_1\right) \cdot x_2,$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_1,$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial o_1} \cdot g'\left(w_5 h_1 + w_6 h_2 + b_3\right) \cdot w_6 \cdot f'\left(w_3 x_1 + w_4 x_2 + b_2\right) \cdot x_2.$$

So we see that $w_1^1 = w_3^1 = const$, $w_2^1 = w_4^1 = const$. In the same way we can prove that $b_1^1 = b_2^1 = const$.

Xavier initialization

Xavier initialization

In short, it helps signals reach deep into the network.

Xavier initialization

In short, it helps signals reach deep into the network.

- If the weights in a network start too small, then the signal shrinks as it passes through each layer until it's too tiny to be useful.

Xavier initialization

In short, it helps signals reach deep into the network.

- If the weights in a network start too small, then the signal shrinks as it passes through each layer until it's too tiny to be useful.
- If the weights in a network start too large, then the signal grows as it passes through each layer until it's too massive to be useful.

What's Xavier initialization?

Initializing the weights in your network by drawing them from a distribution with zero mean and a specific variance

$$\mathrm{Var}\left(W\right) = \frac{1}{n_{in}},$$

where W is the initialization distribution for the neuron in question, and n_{in} is the number of neurons feeding into it. The distribution used is typically Gaussian or uniform.

What's Xavier initialization?

Initializing the weights in your network by drawing them from a distribution with zero mean and a specific variance

$$\text{Var}(W) = \frac{1}{n_{in}},$$

where W is the initialization distribution for the neuron in question, and n_{in} is the number of neurons feeding into it. The distribution used is typically Gaussian or uniform. Glorot & Bengio's paper originally recommended using

$$\text{Var}(W) = \frac{2}{n_{in} + n_{out}},$$

where n_{out} is the number of neurons the result is fed to.

Where did those formulas come from?

Where did those formulas come from?

Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y:

$$Y = W_1X_1 + W_2X_2 + \ldots + W_nX_n.$$

Where did those formulas come from?

Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y:

$$Y = W_1 X_1 + W_2 X_2 + \ldots + W_n X_n.$$

Also suppose that random variables $\{X_i\}_{i=1}^n$ and $\{W_i\}_{i=1}^n$ are all independent and identically distributed and $E(X_1) = E(W_1) = 0$.

Where did those formulas come from?

Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y:

$$Y = W_1X_1 + W_2X_2 + \ldots + W_nX_n.$$

Also suppose that random variables $\{X_i\}_{i=1}^n$ and $\{W_i\}_{i=1}^n$ are all independent and identically distributed and $E(X_1) = E(W_1) = 0$.
It easy to see that for all $i = 1, \ldots, n$ we have

$$\operatorname{Var}(W_iX_i) = E(X_i)^2 \operatorname{Var}(W_i) + E(W_i)^2 \operatorname{Var}(X_i) + \operatorname{Var}(W_i)\operatorname{Var}(X_i)$$

$$= \operatorname{Var}(W_i)\operatorname{Var}(X_i) = \operatorname{Var}(W_1)\operatorname{Var}(X_1)$$

Where did those formulas come from?

It follows that

$$\mathrm{Var}\left(Y\right) = \mathrm{Var}\left(W_1X_1 + W_2X_2 + \ldots + W_nX_n\right) = n\mathrm{Var}\left(W_1\right)\mathrm{Var}\left(X_1\right).$$

Where did those formulas come from?

It follows that

$$\mathrm{Var}\left(Y\right) = \mathrm{Var}\left(W_1X_1 + W_2X_2 + \ldots + W_nX_n\right) = n\mathrm{Var}\left(W_1\right)\mathrm{Var}\left(X_1\right).$$

So if we want to have $\mathrm{Var}\left(Y\right) = \mathrm{Var}\left(X_1\right)$, we need to put

$$\mathrm{Var}\left(W_1\right) = \frac{1}{n}.$$

Where did those formulas come from?

It follows that

$$\text{Var}(Y) = \text{Var}(W_1X_1 + W_2X_2 + \ldots + W_nX_n) = n\text{Var}(W_1)\,\text{Var}(X_1)\,.$$

So if we want to have $\text{Var}(Y) = \text{Var}(X_1)$, we need to put

$$\text{Var}(W_1) = \frac{1}{n}.$$

Glorot Bengio's formula needs a tiny bit more work. If you go through the same steps for the backpropagated signal, you find that you need

$$\text{Var}(W_1) = \frac{1}{n_{out}}$$

to keep the variance of the input gradient the output gradient the same.

Where did those formulas come from?

It follows that

$$\mathrm{Var}\,(Y) = \mathrm{Var}\,(W_1X_1 + W_2X_2 + \ldots + W_nX_n) = n\mathrm{Var}\,(W_1)\,\mathrm{Var}\,(X_1)\,.$$

So if we want to have $\mathrm{Var}\,(Y) = \mathrm{Var}\,(X_1)$, we need to put

$$\mathrm{Var}\,(W_1) = \frac{1}{n}.$$

Glorot Bengio's formula needs a tiny bit more work. If you go through the same steps for the backpropagated signal, you find that you need

$$\mathrm{Var}\,(W_1) = \frac{1}{n_{out}}$$

to keep the variance of the input gradient the output gradient the same. These two constraints can only be satisfied simultaneously if $n_{in} = n_{out}$, so as a compromise, Glorot & Bengio take the average of the two:

$$\mathrm{Var}\,(W_1) = \frac{2}{n_{in} + n_{out}}$$

Case of the activation function

Note that this was about a linear neuron, but it works also for tanh and sigmoid functions. For rectifying nonlinearities this is not true and there is a paper where authors suggest using

$$\text{Var}(W) = \frac{2}{n_{in}}$$

for rectifying activation functions.

Case of the activation function

Note that this was about a linear neuron, but it works also for tanh and sigmoid functions. For rectifying nonlinearities this is not true and there is a paper where authors suggest using

$$\mathrm{Var}(W) = \frac{2}{n_{in}}$$

for rectifying activation functions. Which makes sense: a rectifying linear unit is zero for half of its input, so you need to double the size of weight variance to keep the signal's variance constant.

Case of the activation function

Note that this was about a linear neuron, but it works also for tanh and sigmoid functions. For rectifying nonlinearities this is not true and there is a paper where authors suggest using

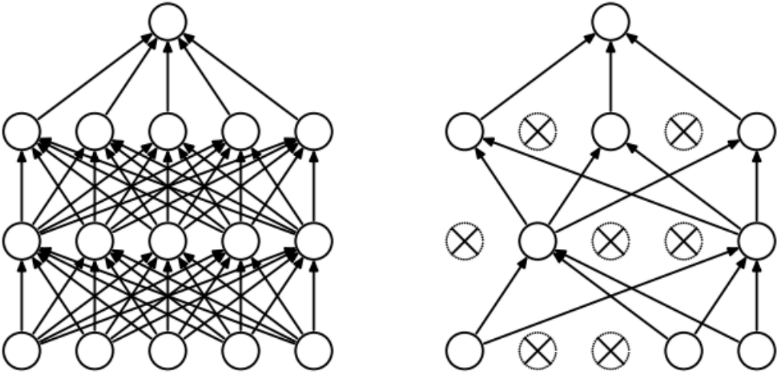
$$\mathrm{Var}\left(W\right) = \frac{2}{n_{in}}$$

for rectifying activation functions. Which makes sense: a rectifying linear unit is zero for half of its input, so you need to double the size of weight variance to keep the signal's variance constant. But Xavier Glorot initialization is still okay.

Outline

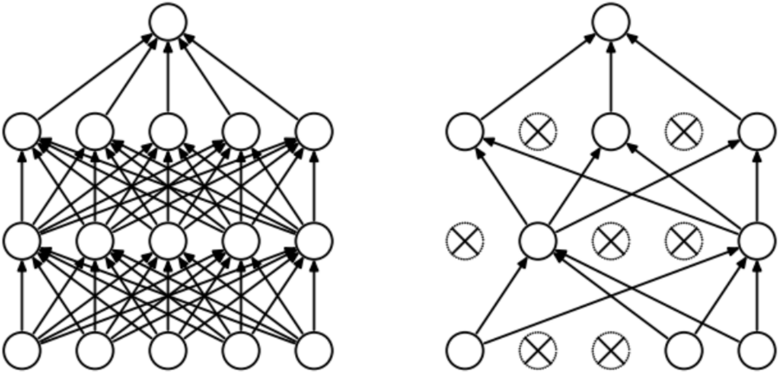
Dropout

Dropout



(a) Standard Neural Net (b) After applying dropout.

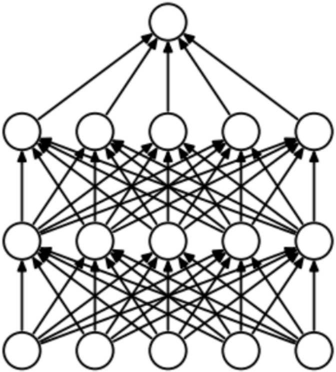
Dropout



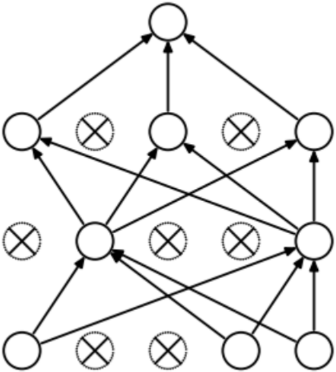
(a) Standard Neural Net (b) After applying dropout.

What to do during the inference?

Dropout



(a) Standard Neural Net



(b) After applying dropout.

What to do during the inference?

Answer: Scale units by $\frac{1}{1 - rate}$ during the training and set rate=1 during the inference.