Gradient Descent and Learning Rate

Lesson 8 – Section 4

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Gradient Descent

- The purpose of gradient descent is to identify the minimum error or the maximum predictive capability on your training set
- You need to adjust your network in many dimensions, looking for the best "direction" to approach error=zero without over training and reducing generalizability to unseen data
- There are several types of gradient descent: Batch, Stochastic and Mini-batch

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Steps for Gradient Descent

- Step 1: Initialize weights (0-1; -1-1; $-\frac{1}{\sqrt{i}}, \frac{1}{\sqrt{i}}$)
- Step 2: Calculate error (e.g. sum of squares: $\frac{1}{2}\sum(Y - \hat{y})^2$)
- Step 3: update the weights with the gradients to reach the optimal values where SSE is minimized based on a learning rate
- Step 4: Use the new weights for prediction and to calculate the new SSE
- Step 5: Repeat 2 and 3 until further updates to the weights do not significantly reduce the error

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Learning Rate

Learning rate is application dependent

- The lower the learning rate the longer it take to converge
- Used to prevent falling into local minimum

Start small and increase, comparing across training runs

Adaptive techniques, such as bold driver and annealing can also be used for creating an adjustable rate

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Feeding forward

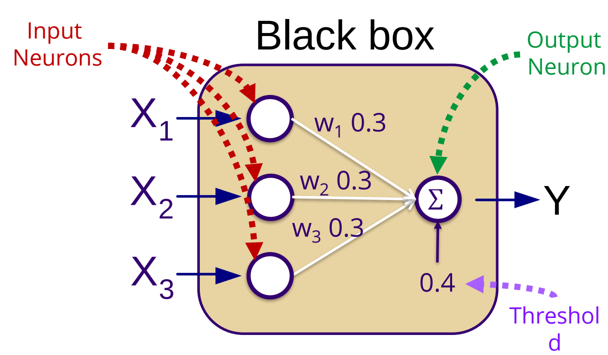
- Load a problem into the network by giving it some features (AKA inputs)
- These inputs might need to be normalized to range between -1 and 1 or 0 and 1. An example would be to do the following for each input:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} = \frac{10 - 2}{20 - 2} = \frac{8}{18} = .44\bar{4}$$

- Each connection gets a “signal” from its earlier node (either input or hidden layer)
- This signal is multiplied by its own weight
- Then the signal is passed on to later nodes
- A “predicted” classification ($\hat{y}$) is generated as the sum of these weights which is compared to the actual classification (Y)
- You could stop here using only reinforced learning

Simple Example

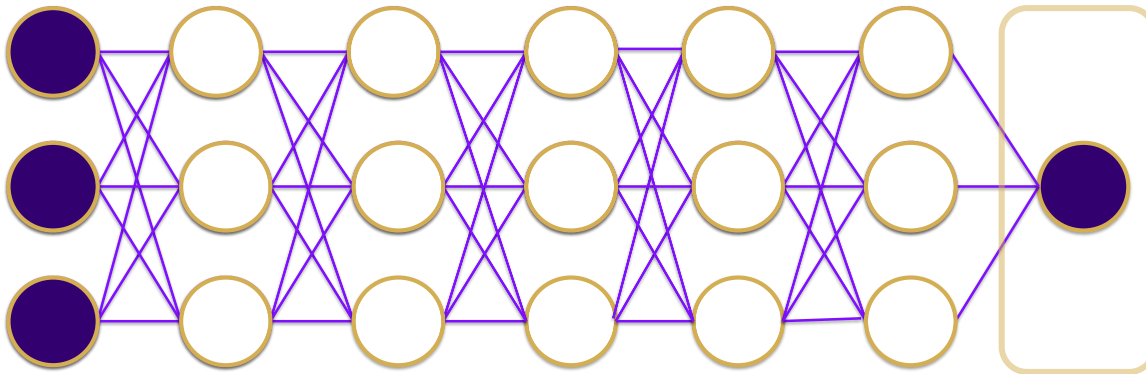
X_1	X_2	X_3	Y
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
0	0	0	**0**



$$y_1(t_0) = f(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$where\ f(z) = \begin{cases} 0\ if\ z < 0 \\ 1\ if\ z \geq 0 \end{cases}$$

Backpropagation Visual



Simple Example BP Weight Updates

X_1	X_2	X_3	Y
1	0	0	**0**

Assume Learning Rate = 0.01

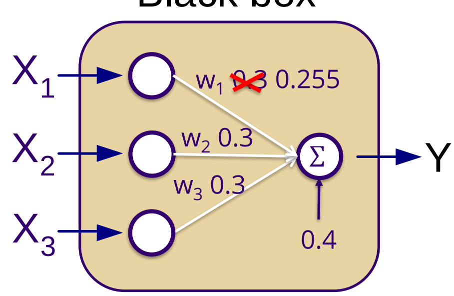
$$w_1 = 0 - .01 \cdot \frac{1}{2}(0-1)^2 = -.045$$

$$w_2 = 0.1 \cdot \frac{1}{2}(0-0)^2 = 0$$

$$w_3 = 0.1 \cdot \frac{1}{2}(0-0)^2 = 0$$

Only w_1 is updated from 0.3 → 0.255 $= (0.3 - 0.045)$

Black box



PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Epoch v. Batch v. Iteration
What's the difference?

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

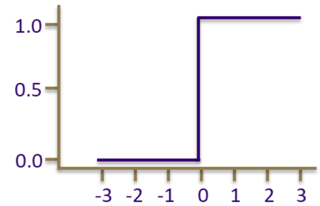
Quick Terminology Pause

- **Iterations** are logical because we do this all the time in computer programming
- **Batch** is subsets of the input vector
- **Epochs** are important to understand because they are a settable hyperparameter

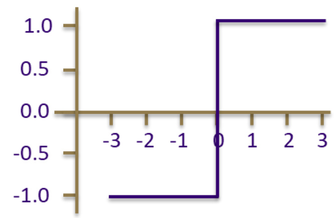
PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Common Binary Step (Activation) Functions

$$heaviside(z) = \begin{cases} 0 \; if \; z < 0 \\ 1 \; if \; z \geq 0 \end{cases}$$

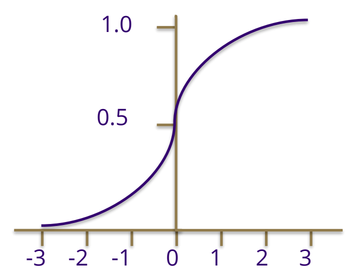


$$sign(z) = \begin{cases} -1 \; if \; z < 0 \\ 0 \; if \; z = 0 \\ 1 \; if \; z > 0 \end{cases}$$



Common Non-linear (Activation) Functions

$$logistic \; (sigmoid)(z) = \frac{1}{1+e^{-z}}$$



$$Hyperbolic \; tangent(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$