Supervised Regression (ANNs)

Presented by: Reem Elmahdi

Agenda

- Supervised Regression (SR) definition
- ML models used in SR problems
- Biological Neural Networks
- Artificial Neural Networks (ANNs)
- ANNs components
- Forward Pass
- Training Process
- Cost Function
- Gradient Descent
- Backpropagation

What is regression supervised problem?

Example Predicting the score of a test based on the number of sleep and study hours on the night before.

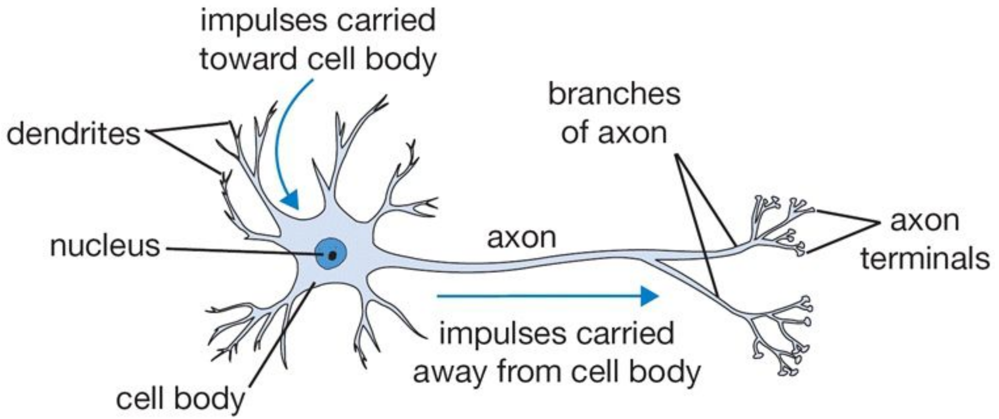
Supervised: the examples have input and output.

Regression: test score is continuous value (if grade should be predicted then it is a classification problem).

Sleep	Study	Score
3	5	75
5	1	82
10	2	93
8	**3**	**?**

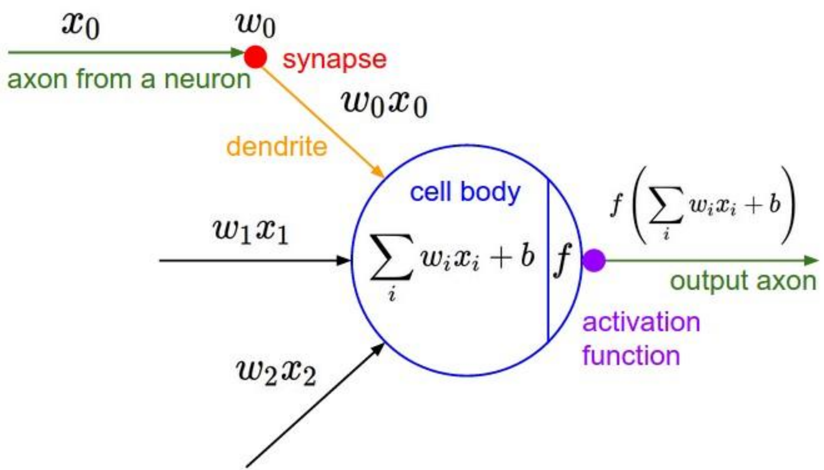
Biological motivation to ANNs

- ANNs architecture was inspired by how neurons in the brain are connected (see the figure below).
- The idea was to model human's brain in solving problems.
- A neuron is the basic computational unit in the brain.
- Synapses are the connections between neurons, it allows signals to flow all over the brain (signal-passing task).



Artificial Neural Networks

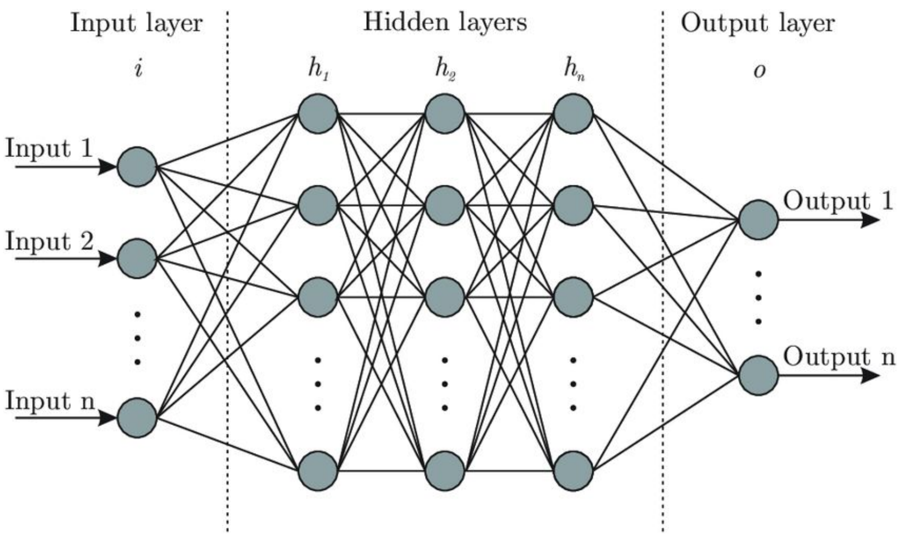
- ANNs architecture was inspired by how neurons in the brain are connected.
- The idea was to model human's brain in solving problems.
- A perceptron is the simplest form of ANNs, it contains number of inputs, number of outputs and only one neuron (node).
- Connections (synapses): $\boldsymbol{a_i = x_i * w_i}$
- Neurons (nodes): $\boldsymbol{z = f(\sum x_i * w_i + b)}$



Artificial Neural Networks Components

The architecture of this problem is as follows:

- 2 Inputs (Sleep and study hours).
- 1 Output (test score).
- 1 hidden layer with 3 nodes (neurons).
- Sigmoid activation function.



How to solve the problem

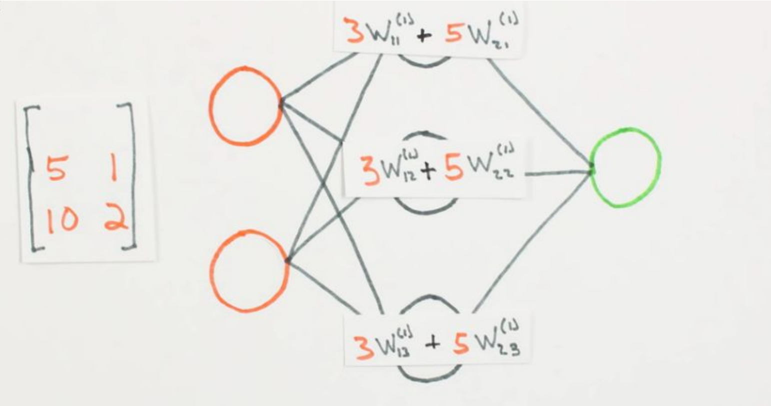
- Scaling $\boldsymbol{x_{norm}} = \boldsymbol{x/max(x)}$.
- Define the architecture of the network (previous slide).
- Forward pass:
  - $\boldsymbol{a = x * w}$
  - $\boldsymbol{b = f\,(a)}$
  - $\boldsymbol{c = b * v}$
  - $\boldsymbol{\hat{y} = f\,(c)}$
- Train the network, this is done by minimizing the cost function: $\boldsymbol{J = \sum \tfrac{1}{2}(e_1^{\,2} + e_2^{\,2} + e_3^{\,2})}$ where $\boldsymbol{e = y - \hat{y}}$.
- Numerical estimation to the final equation: $\boldsymbol{J = \sum \tfrac{1}{2}(y\text{-}f(f(x^*w)v))^2}$ Alternatively, finding the rate of change of $\boldsymbol{J}$ with respect to $\boldsymbol{w,v}$: $\boldsymbol{\partial J/\partial w}$ (This method called gradient descent).
- if $\boldsymbol{\partial J/\partial w}$ is positive then the function is going up, otherwise, it is going down.
- The main reason we are using cost function as sum of squared errors to exploit the convex function nature.

Forward Pass

3	*5*
5	*1*
10	*2*

*

w_1^1	w_2^1	w_3^1
w_1^2	w_2^2	w_3^2



$3\, w_1^1 + 5\, w_1^2$	$3\, w_2^1 + 5\, w_2^2$	$3\, w_3^1 + 5\, w_3^2$
$5\, w_1^1 + 1\, w_1^2$	$5\, w_2^1 + 1\, w_2^2$	$5\, w_3^1 + 1\, w_3^2$
$10\, w_1^1 + 2\, w_1^2$	$10\, w_2^1 + 2\, w_2^2$	$10\, w_3^1 + 2\, w_3^2$

Training ANNs

- This step is followed by applying the sigmoid activation function. Then multiply the output with the weights of that connect hidden to output, lastly sigmoid activation function should be applied.

- $\partial J/\partial w = \partial \sum \frac{1}{2}(y - \hat{y})^2 / \partial w = \sum \partial \frac{1}{2}(y - \hat{y})^2 / \partial w$

- The error at the last layer for single value: $\partial J/\partial w = (y - \hat{y})$

- Back-propagation: chain rule, multiply the result by $\partial\hat{y}/\partial w$

Improve Results by applying Cost Function

- This step is followed by applying the sigmoid activation function. Then multiply the output with the weights of that connect hidden to output, lastly sigmoid activation function should be applied.
- $J = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$
- We minimize the cost by changing the weights:

w_1^1	w_2^1	w_3^1
w_1^2	w_2^2	w_3^2

v_1^1
v_1^2
v_1^3

- So we can either apply ”Semi-Brute Forcing” by checking range of possible values, alternatively, we can use Gradient Descent function.

Gradient Descent

- **Gradient Descent (GD)** is an algorithm that finds the weights (optimal weights) that make the cost function $\boldsymbol{J}$ the minimum.

- We can use either batch or stochastic GD (in this case we will use batch). In batch GD, we sum all the derivatives of $\boldsymbol{J}$ for all the observations: $\partial J/\partial w$.

Backpropagation

- **Backpropagation** is the technique used in training artificial neural network by updating the weights. Backpropagation make use of the GD algorithm.

- We have to compute 2 gradients: $\partial J/\partial w$ and $\partial J/\partial v$ the gradient with respect to the weight for hidden layer and the gradient with respect to the weight for output layer, respectively.

Backpropagation

- ❑ $\partial J/\partial v = b^T \delta_2$
- ❑ $\delta_2 = -(y - \hat{y}) f'(c)$
- ❑ $\partial J/\partial w = x^T \delta_1$
- ❑ $\delta_1 = \delta_2 v^T f'(a)$

- ❑ We should make our cost (***J***) as small as possible with an optimal combination of the weights (independently for both ***w*** and ***v***).

Thank you :)