STAT 663
Statistical Graphics and Data Exploration I

Week 12

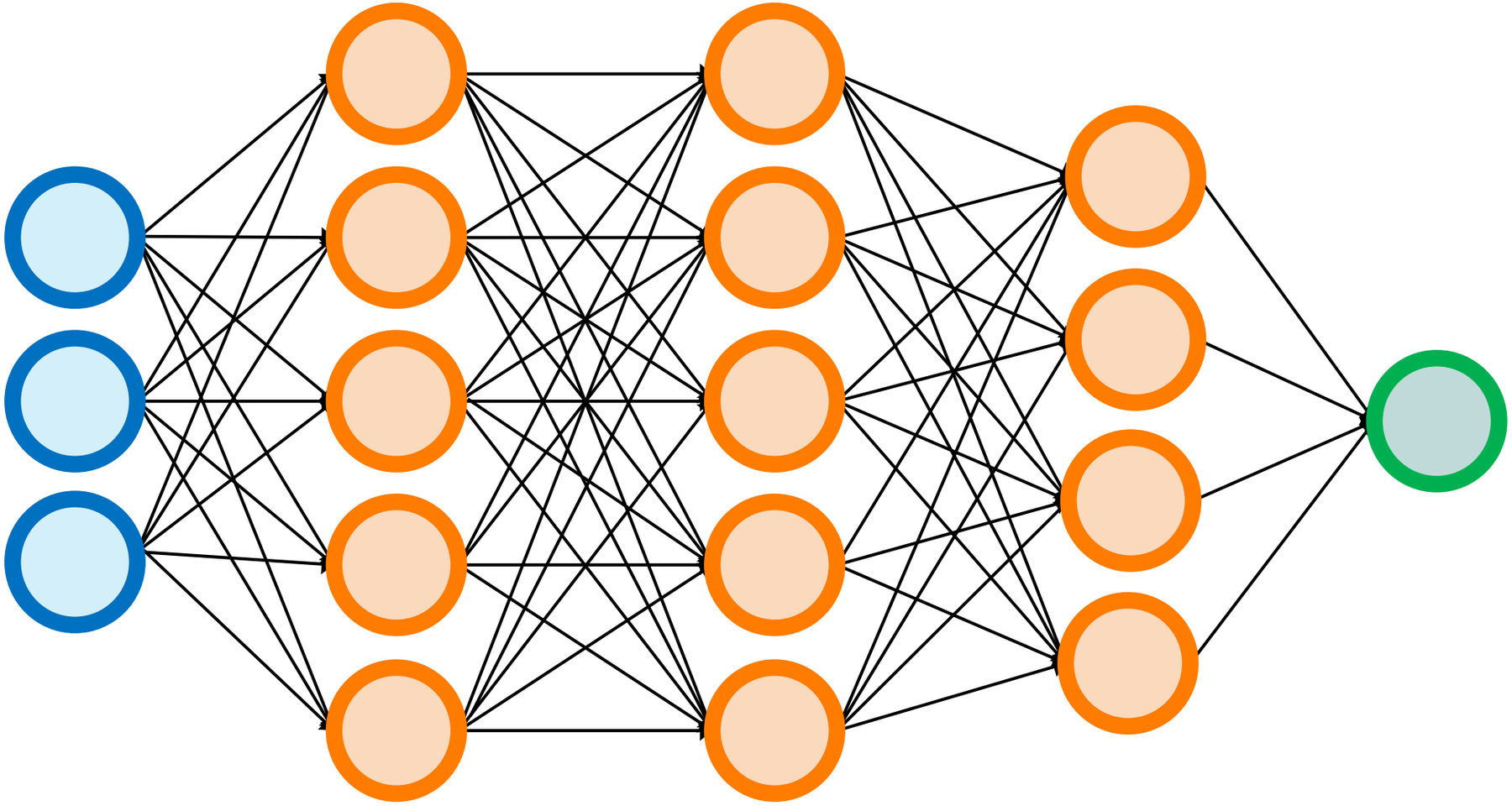
Neural Network

Lecture 1

Module Highlights

- Neural Network structure
- Neural Network models
- Backpropagation and forwardpropagation
- Neural Network Auto-Regressive (NNAR) models
- The nnetar() function in R
- Application of the NNAR model to COVID-19 study

Neural Networks



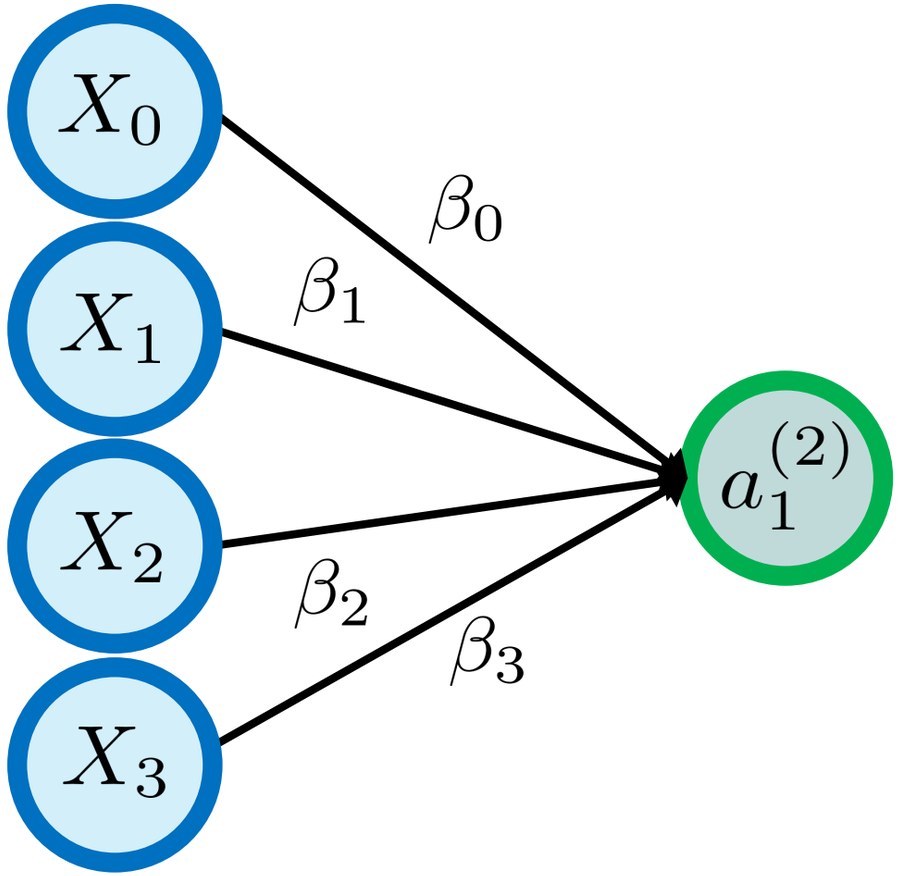
Input Layer

Hidden Layers

Output Layer

Perceptron: Single Node

Simplest version: linear regression



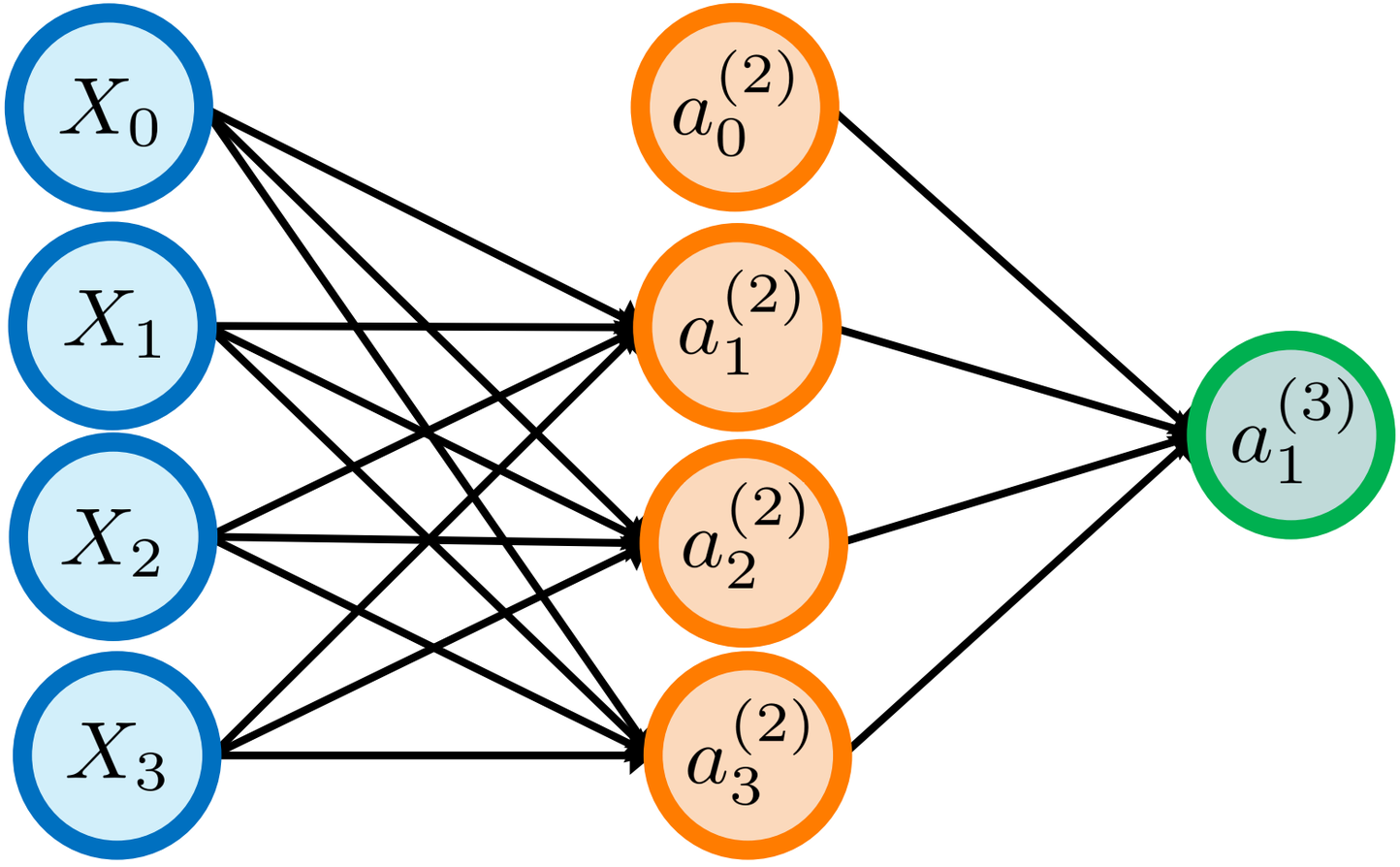
- Coefficients attached to predictors are called "**weights**".
- Forecasts are obtained by a linear combination of inputs.
- Result modified by nonlinear function before being output.

$$a_i^{(1)} = X_i,\ i = 0, 1, 2, 3$$

$$a_1^{(2)} = g(\sum_{j=0}^{3} \beta_j X_j)$$

Neural Network Structure

Nonlinear model with one hidden layer

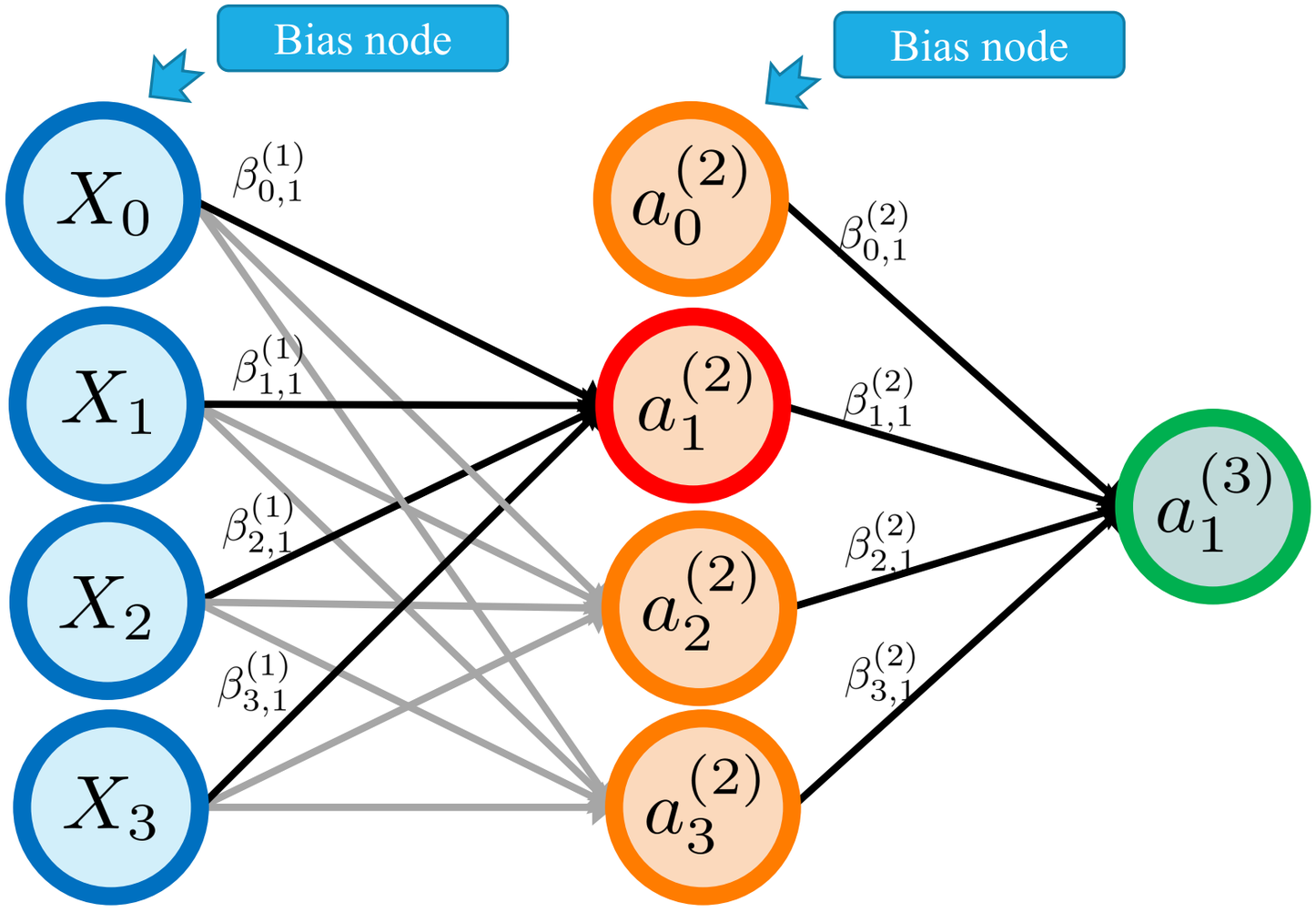


Layer 1
(Input Layer)

Layer 2
(Hidden Layer)

Layer 3
(Output Layer)

Neural Network Structure



Layer 1
(Input Layer)

Layer 2
(Hidden Layer)

Layer 3
(Output Layer)

- Hidden neuron

$$\alpha_i^{(2)} = g\left(\sum_{j=0}^{3} \beta_{j,i}^{(1)} X_j\right)$$

- Output neuron

$$\alpha_1^{(3)} = g\left(\sum_{j=0}^{3} \beta_{j,1}^{(2)} \alpha_j^{(2)}\right)$$

Neural Network Models

- Inputs to hidden neuron i linearly combined: $z_i^{(2)} = \sum_{j=0}^{3} \beta_{j,i}^{(1)} X_j$.
- Modified using nonlinear function g such as a sigmoid:

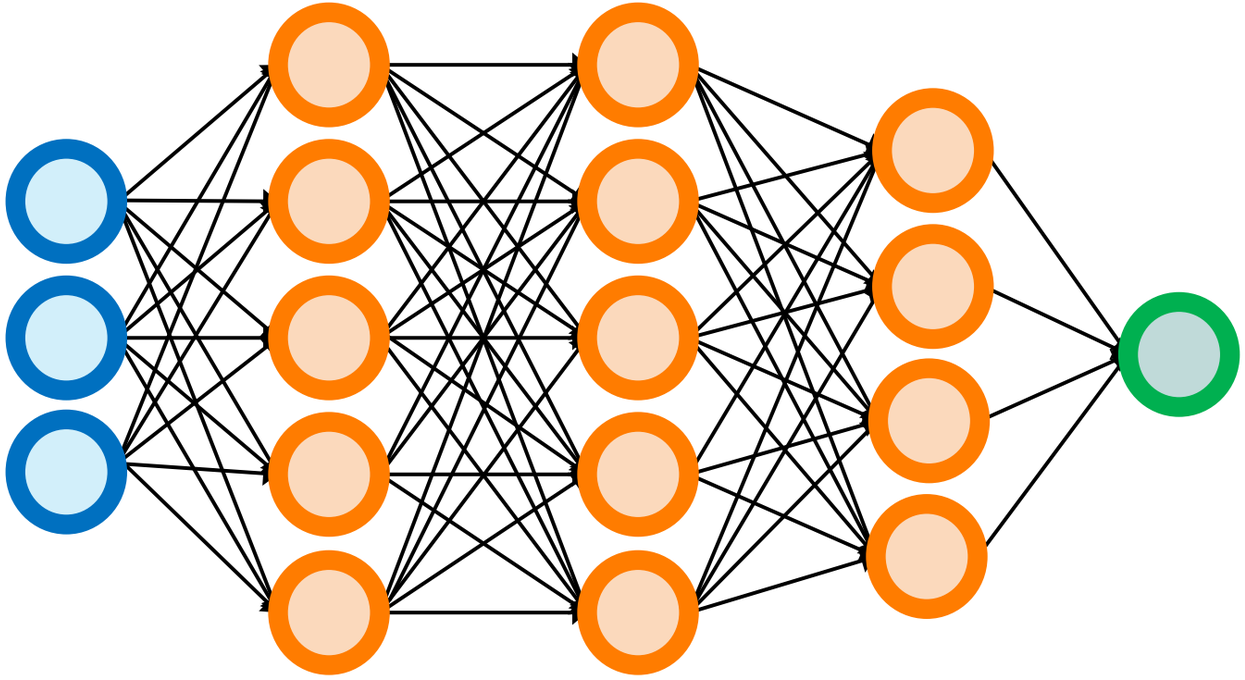
$$\alpha_i^{(2)} = g\left(\sum_{j=0}^{3} \beta_{j,i}^{(1)} X_j\right)$$

- Output neuron

$$\alpha_1^{(3)} = g\left(\sum_{j=0}^{3} \beta_{j,1}^{(2)} \alpha_j^{(2)}\right)$$

- Activiation function: $g(z) = 1/(1+\exp(-z))$
- This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

Neural Network Structure

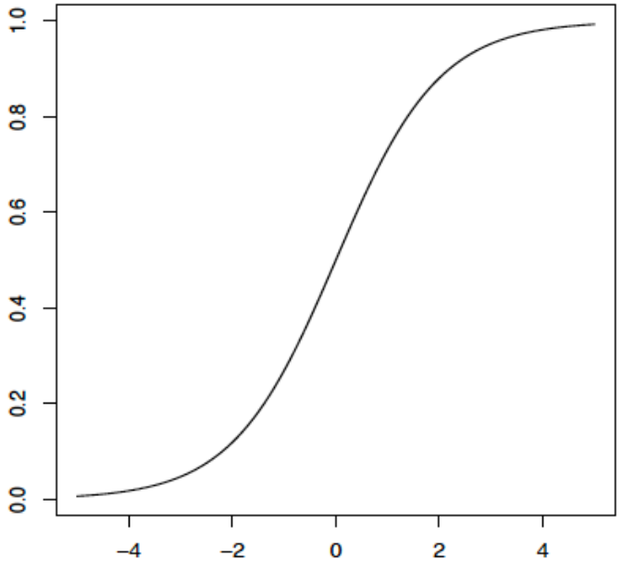


- A multilayer feed-forward network, where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using a linear combination.
- Result modified by nonlinear function before being output.

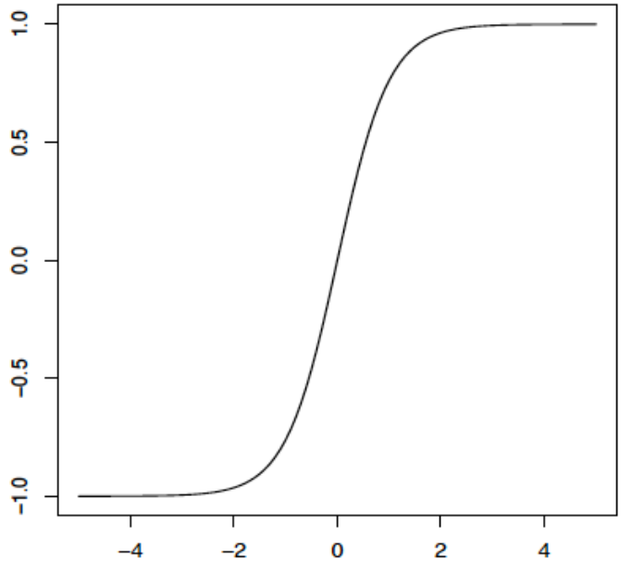
- $a_i^{(j)}$: activation of unit i in layer j.
- $\boldsymbol{\beta}^{(j)}$: weight matrix stores parameters from layer j to layer $j+1$.
- If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\boldsymbol{\beta}^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

Activation Functions

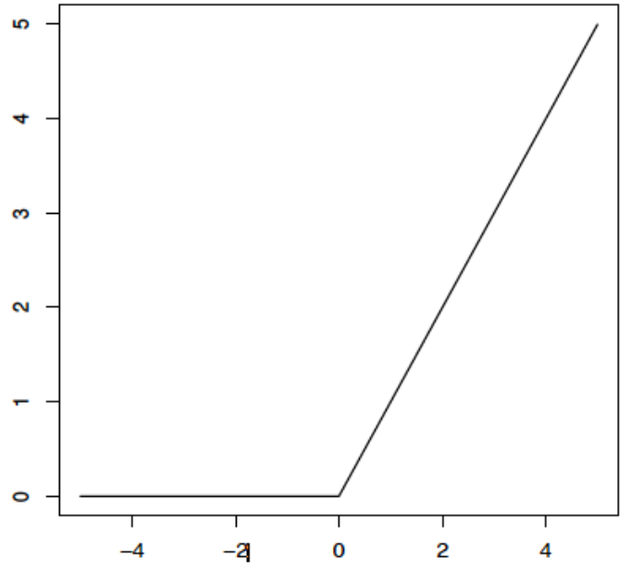
- Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it.
- There are several activation functions you may encounter in practice:



Sigmoid



Tanh



Relu

Neural Network Models

- Weights take random values to begin with, which are then updated using the observed data.
- There is an element of randomness in the predictions.
- So the network is usually trained several times using different random starting points, and the results are averaged.
- Number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

Properties

Two-layer network are universal function approximates. Let f be a continuous function on a bounded subset of p-dimensional space. Then there exists a two-layer neural network $\hat{f}$ with a finite number of hidden units that approximate f arbitrarily well. For all $\mathbf{x}$ in the domain of f,

$$|f(\mathbf{x}) - \widehat{f}(\mathbf{x})| < \epsilon.$$

Fit the Model

- The neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well.
- For regression, we use sum-of-squared errors as our measure of fit (error function)

$$\ell(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(L)}) = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$$

- Gradient Descent:

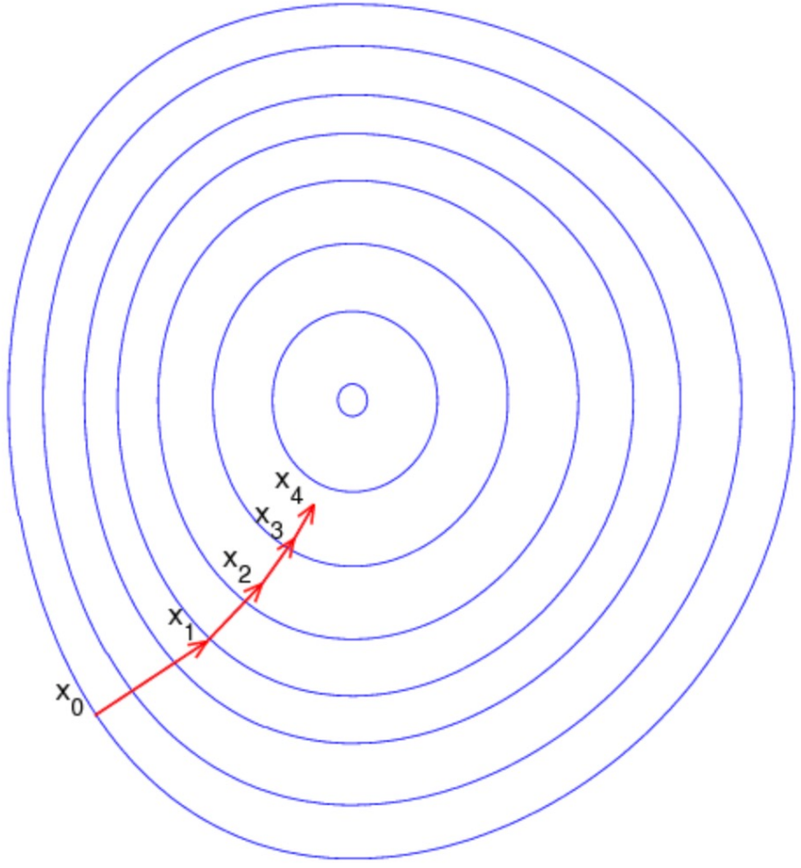
$$\boldsymbol{\beta}^{(j)} \leftarrow \boldsymbol{\beta}^{(j)} - \alpha \nabla_{\boldsymbol{\beta}^{(j)}} \ell(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(L)})$$

  for all j.

What Is Gradient Descent?

- **Gradient Descent** is an optimization algorithm for finding a local minimum of a differentiable function. It is simply used to find the values of a function's parameters that minimize a error function as far as possible.

- You start by defining the **initial parameter's values** and from there gradient descent uses calculus to iteratively adjust the values so they minimize the given cost-function.



How Gradient Descent Works?

- Gradient Descent:

$$\boldsymbol{\beta}^{(j)} \leftarrow \boldsymbol{\beta}^{(j)} - \alpha \nabla_{\boldsymbol{\beta}^{(j)}} \ell(\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(L)})$$

for all j.

Learning rate

Updated value

Current value

Direction of the steepest descent

- When α is too big, gradient descent may jump across the valley and end up on the other side. This will lead to the error function diverge.
- When α is too small, it will take the algorithm long time to converge.

Backpropagation

Back-propagation: an efficient method for computing gradients needed to perform gradient-based optimization of the weights in a multi-layer network.

Loop until convergence: for each observation i

1. Given input X_i, propagate activity forward ($X_i \rightarrow a_i \rightarrow \hat{Y}_i$) (forward pass)
2. Propagate gradients backward (backward pass)
3. Update each weight (via gradient descent)

Ideas of Backpropagation

- We don't have targets for a hidden unit, but we can compute how fast the error changes as we change its activity.
- Instead of using desired activities to train the hidden units, use error derivatives w.r.t. hidden activities.
- Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined.
- We can compute error derivatives for all the hidden units efficiently.
- Once we have the error derivatives for the hidden activities, its easy to get the error derivatives for the weights going into a hidden unit.

This is just the chain rule!

References

- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.