PHIL 7010: Formal Methods for AI, Data, and Algorithms

Week 1
Deep Learning and Neural Networks

Boris Babic,
HKU 100 Associate Professor of Data Science, Law and Philosophy

Today

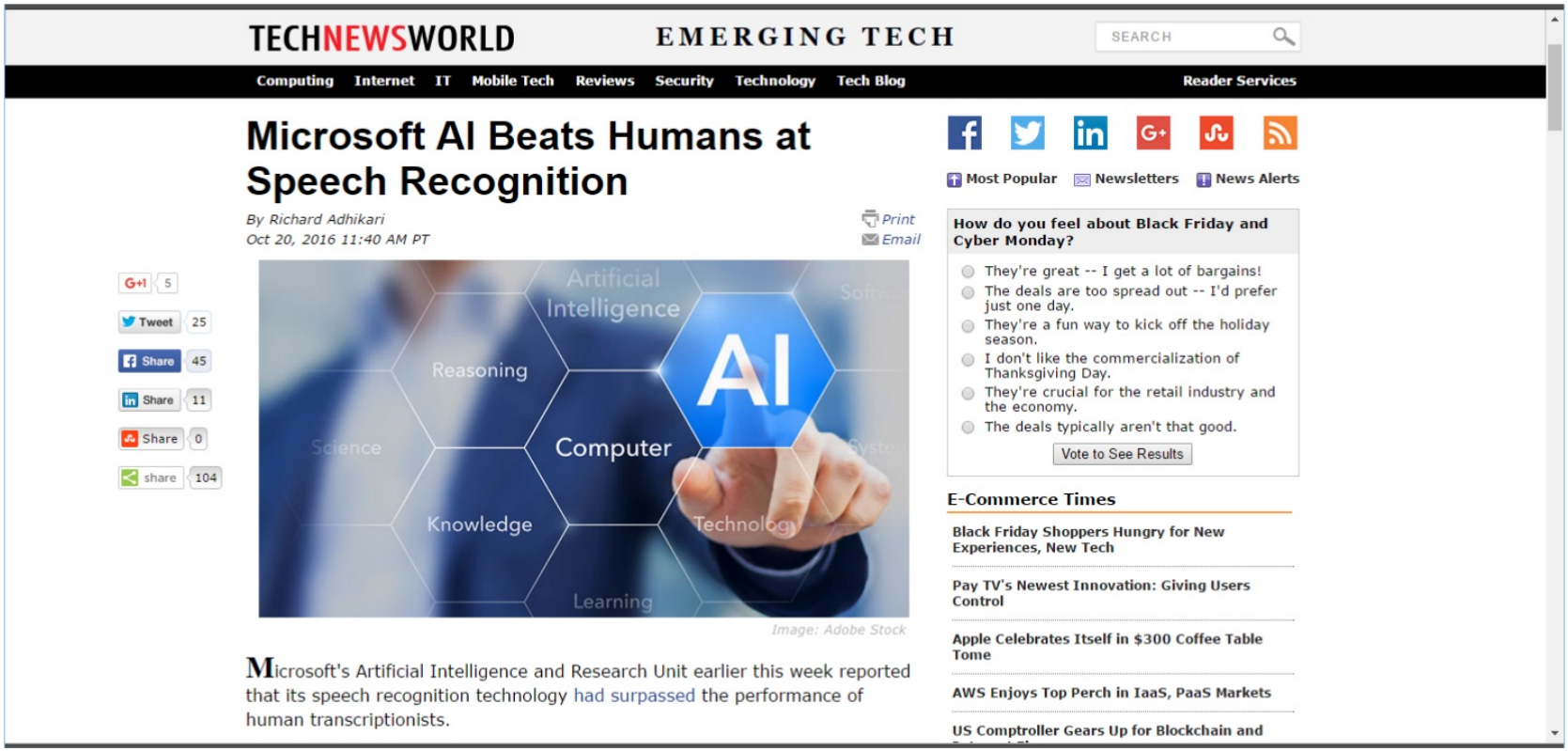
Learning goals

- Basics of Deep Learning and Neural Networks
- Deep Learning Applications
- Architecture of Neural Networks
- Build and Interpret Neural Networks in R

Machine Learning v.s. Deep Learning

- We are now familiar with terms like Artificial Intelligence (AI) and Machine Learning (ML).
- Deep learning is one of the subsets of machine learning that uses deep learning algorithms to implicitly come up with important conclusions based on input data.
- Deep learning doesn't rely on human expertise as much as traditional machine learning. Instead of using task-specific algorithms, it learns from representative examples.
- For example, if you want to build a model that recognizes cats by species, you need to prepare a database that includes a lot of different cat images.
- Applications:
  - Speech recognition
  - Facial recognition
  - Natural language processing
  - Recommender systems

Some Breakthroughs



TECHNEWSWORLD EMERGING TECH SEARCH

Computing Internet IT Mobile Tech Reviews Security Technology Tech Blog Reader Services

Microsoft AI Beats Humans at Speech Recognition

By Richard Adhikari
Oct 20, 2016 11:40 AM PT

Print Email

G+1 5 Tweet 25 Share 45 Share 11 Share 0 share 104

Image: Adobe Stock

Microsoft's Artificial Intelligence and Research Unit earlier this week reported that its speech recognition technology had surpassed the performance of human transcriptionists.

Most Popular Newsletters News Alerts

How do you feel about Black Friday and Cyber Monday?

- They're great -- I get a lot of bargains!
- The deals are too spread out -- I'd prefer just one day.
- They're a fun way to kick off the holiday season.
- I don't like the commercialization of Thanksgiving Day.
- They're crucial for the retail industry and the economy.
- The deals typically aren't that good.

Vote to See Results

E-Commerce Times

- Black Friday Shoppers Hungry for New Experiences, New Tech
- Pay TV's Newest Innovation: Giving Users Control
- Apple Celebrates Itself in $300 Coffee Table Tome
- AWS Enjoys Top Perch in IaaS, PaaS Markets
- US Comptroller Gears Up for Blockchain and

Some Breakthroughs

AlphaGo

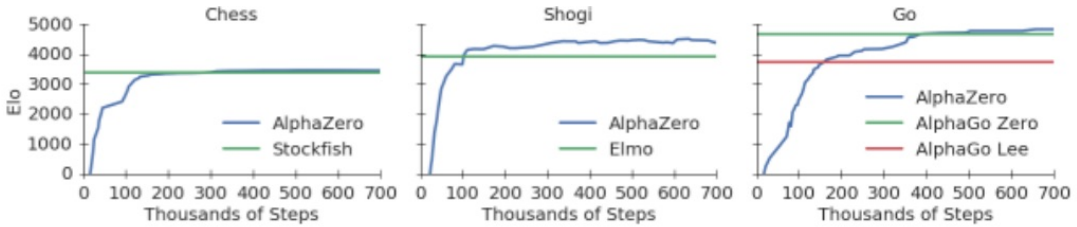


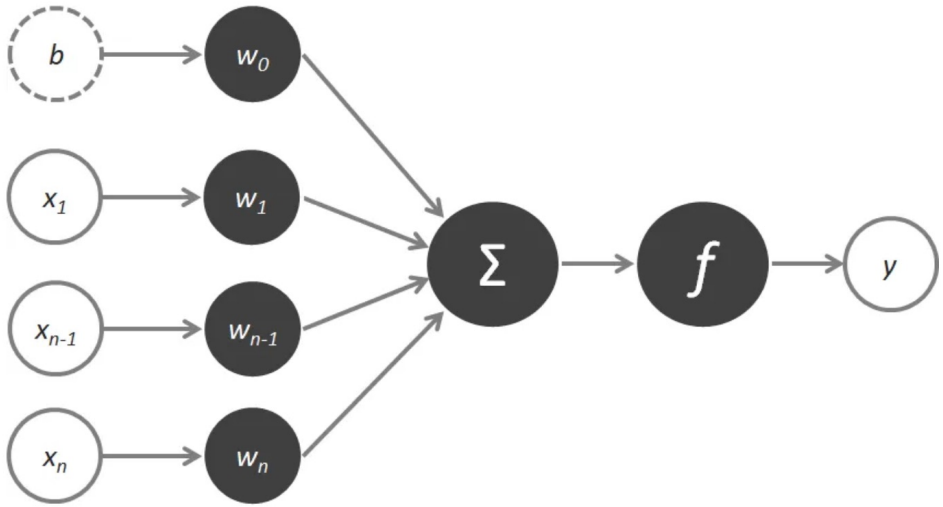
Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (*29*).

Perceptron

- Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are at the heart of deep learning algorithms.
- ANNs are composed of neurons, where each neuron individually performs only a simple computation.
- Before getting into Deep Learning, it's a good idea to begin with the fundamental component of an ANN: the individual neuron.
- A neuron returns some output information from several input data, and in the late 1950s, Frank Rosenblatt and other researchers developed a class of ANN called **Perceptron**.
- A perceptron is a **linear classifier**; that is, it is an algorithm that classifies input by separating two categories with a straight line.

Single Layer Perceptron

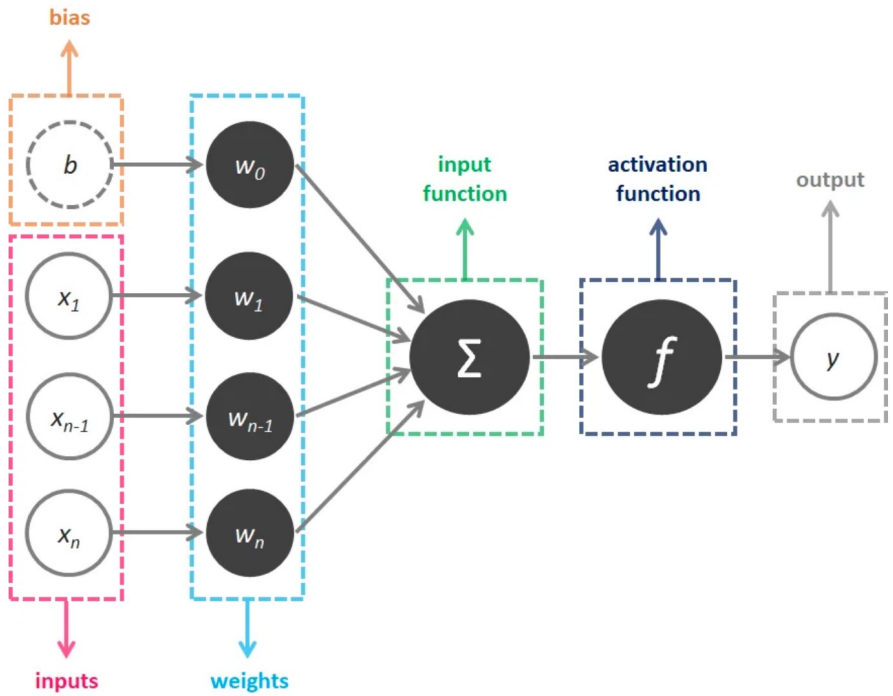
- The Single-Layer Perceptron (SLP) sets the groundwork for the fundamentals of modern Deep Learning architectures.
- They perform classification tasks, and can deal only with **linearly separable data**.
- It consists of a **single layer** of artificial neurons (also called perceptrons). Each neuron receives inputs, computes a weighted sum, and applies a threshold activation function to produce an output. There are no hidden layers in a single-layer perceptron.



How SLP works

SLP consists of five components: Inputs, Weights, Weighted sum (input), Linear/binary activation function, Bias.

- Inputs are multiplied by weights enabling the perceptron to assign more importance to some inputs than others.
- The weighted values are totaled to create the weighted sum.
- Bias is added as another adjustment to ensure accurate output.
- Based on the input (weighted sum), the activation function delivers the perceptron's output based on a pre-defined threshold, where if f(z) is greater than a defined threshold θ we predict 1 and -1 otherwise.



From SLP to MLP

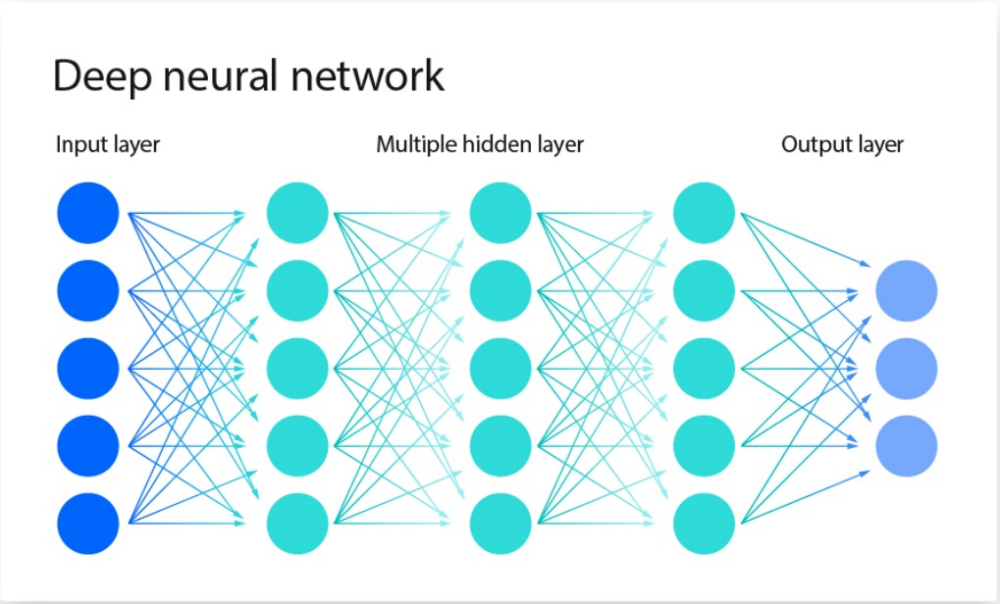
- The SLP, during its initial stages, was celebrated as a transformative advancement in computational models. Its ability to linearly classify made it instrumental in a variety of domains.
- However, SLPs represent weak models because they can only learn linearly separable functions, and as we know, the world is generally non-linear.
- It was not until the 1980s that some of these limitations were overcome with an improved concept called **Multi-Layer Perceptron (MLP)**.
- They also output diverse results on different runs, since all they care about is reaching some linear discrimination, not necessarily the most optimal one.
- MLPs were found as a solution to represent non-linearly separable functions, where the outputs of one layer are the inputs of the next one.
- The multifaceted MLPs, with their modest **hidden layers**, paved the way for the emergence of Deep Neural Networks (DNNs).

The Rise of Deep Learning

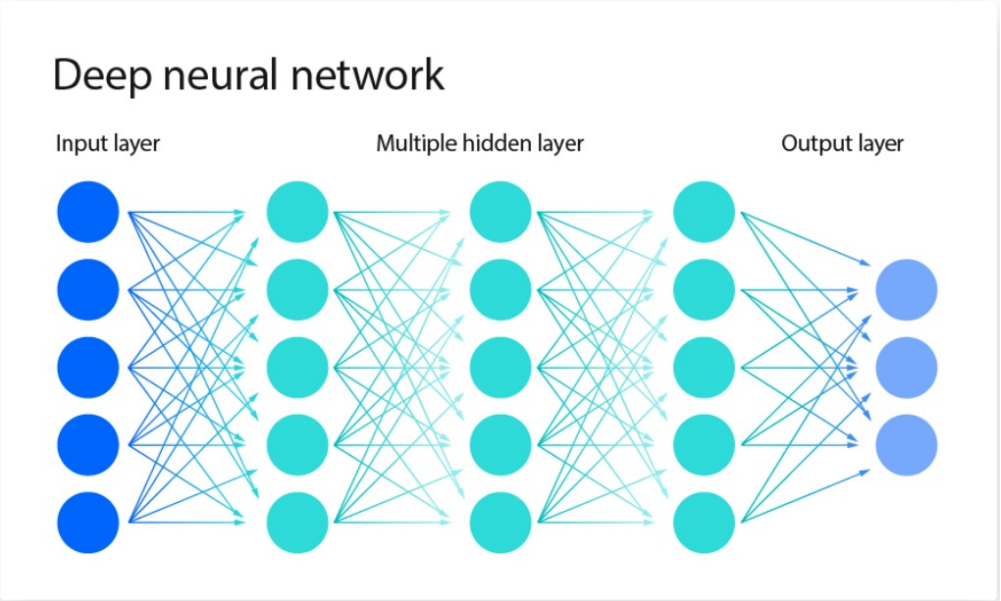
- The "Neural Network Renaissance" (2000s)
  - As the use of MLPs in machine learning tasks started to gain popularity, there was a resurgence of interest in neural networks, driven by advancements in computing power, larger datasets, and more efficient training algorithms.
  - Researchers developed deep neural networks, by stacking multiple hidden layers. These networks showed improved performance in various applications.
- Deep Learning (Mid-2010s - Present)
  - Deep learning achieved remarkable success in various domains
  - In the early 2010s, multiple studies recommended using a specific activation function (ReLu) in the neurons to train deeper neural networks efficiently.
  - In 2016, AlphaGo — an AI developed by DeepMind — beat the Go world champion.
  - In 2020, OpenAI released the third iteration of an AI specialized in producing texts: GPT3.

Deep Neural Networks

- Deep learning methods combine **multiple layers of neural networks to learn complex models**. Each neural network in a deep learning system is connected to other neural networks and to data to provide the computational resources needed to process the input.
- An DNN is typically comprised of an input layer, one or more hidden layers, and an output layer.



Components of Neural Networks - Neurons



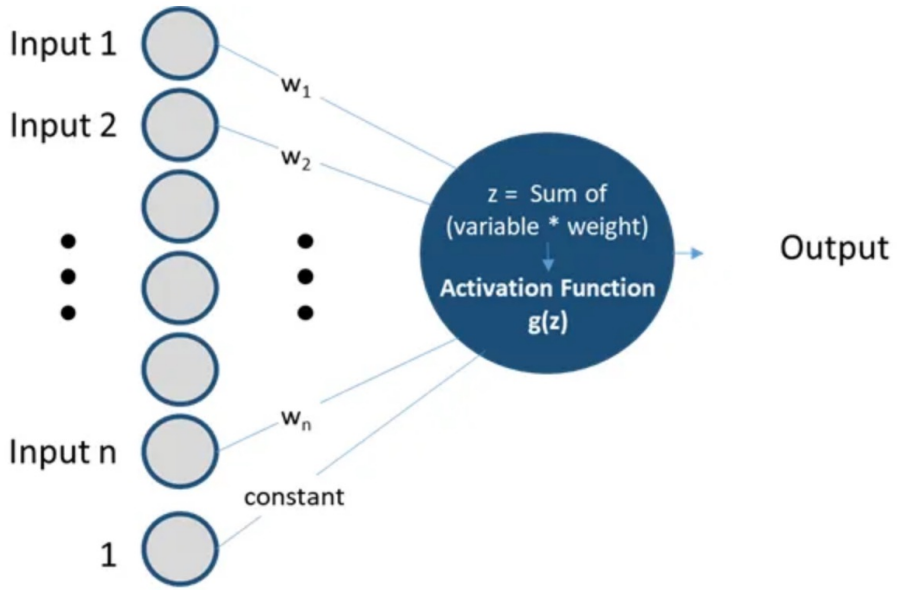
- **Neurons:** A neuron or a node of a neural network is a computing unit that receives information, performs simple calculations with it, and passes it further.
- In a large neural network with many neurons and connections between them, neurons are organized in layers.
- All neurons in a net are divided into three groups:
  - Input neurons that receive information from the outside world (Dark blue circles)
  - Hidden neurons that process that information (Light blue circles in the middle)
  - Output neurons that produce a conclusion (circles on the very right)

Components of Neural Networks - Layers

- But how do these neurons and layers process information into a decision we need?
- The mechanism behind was quite similar to logistic regression.
- Recall that logistic regression is a binary model where the output y is either zero or one. An important step in a logistic model is that it applies a sigmoid function that will convert the continuous value y of the function into a boolean value (0 or 1) based on a specific threshold.
- Imagine neural networks as an expansive logistic model. Each neuron in the input layer corresponds to a predictor in our 'model,' and each connection between neurons in one layer and neurons in the next layer carries a **weight**, denoted as 'w.' These weights are akin to beta coefficients in logistic regression.
- Instead of an intercept, the neural network includes **bias** 'b' allowing for more variations of weights to be stored. Biases add a richer representation of the input space to the model's weights.

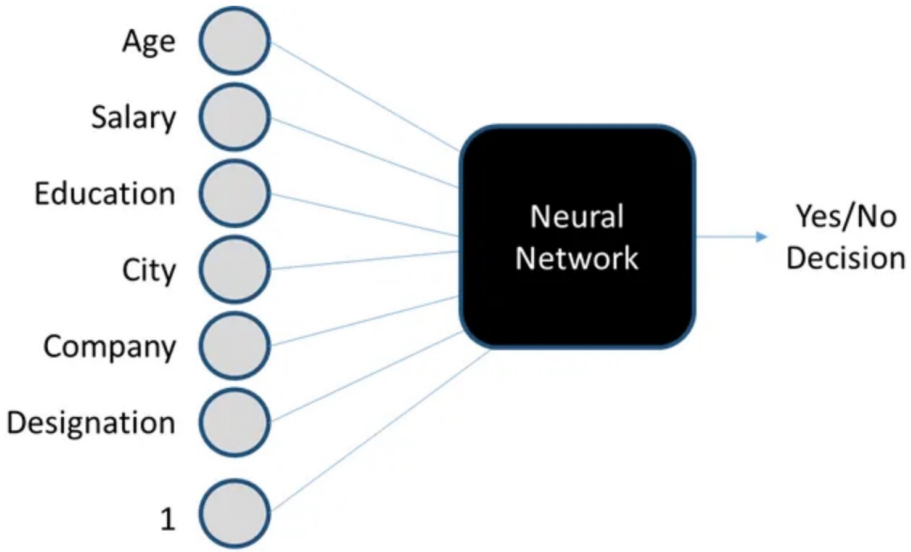
Activation function

- In neural networks, each node in the hidden layers has its own set of weights 'w' and bias 'b' and learns unique insights into the relationship between feature variables and target variables.
- To perform transformations, every neuron has an **activation function** that outputs a boolean result.
- One of the commonly used activation functions is the sigmoid function. Other activation functions commonly used in DL include ReLU (Rectified Linear Unit) and Tanh (Hyperbolic Tangent).



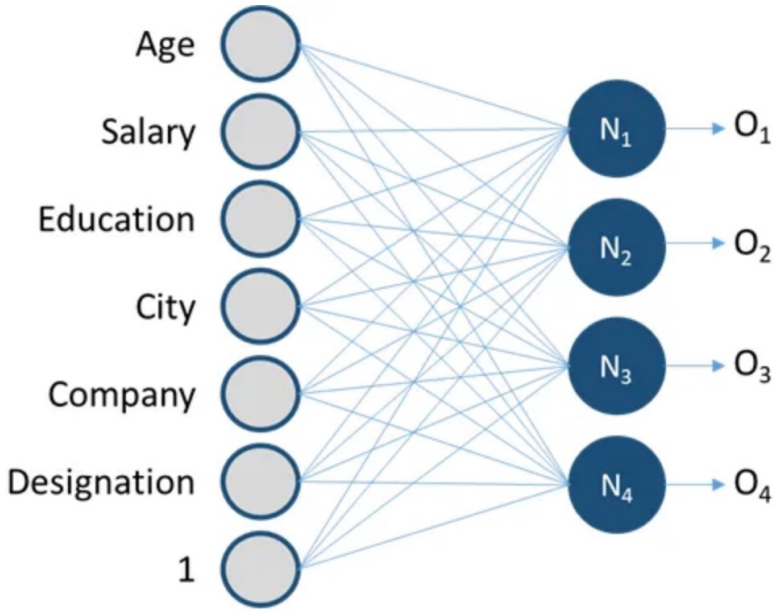
Example

- Assume we are a credit card company targeting people without an adequate credit history. Using the historical data of its users, we want to decide whether a new applicant should be given a credit card or not.
- Note: The latter 4 are "qualitative" variables, but for this illustration, let's assume we can somehow quantify them. Also assume the neuron indicating '1' is a bias unit.



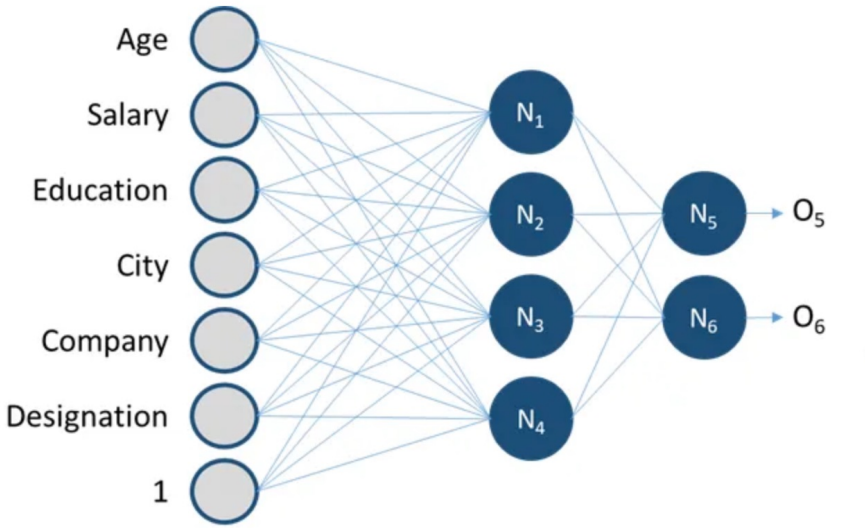
Example

- For creating a Neural Network, the first step is just stacking several units or neurons together to create a layer.
- Each blue circle is a processing unit (neuron), which performs the sum-product of the inputs (age, salary, education etc) and the weights, and applies an activation function (say ReLU) to give an output. Each neuron has a different set of weights and returns a different output.

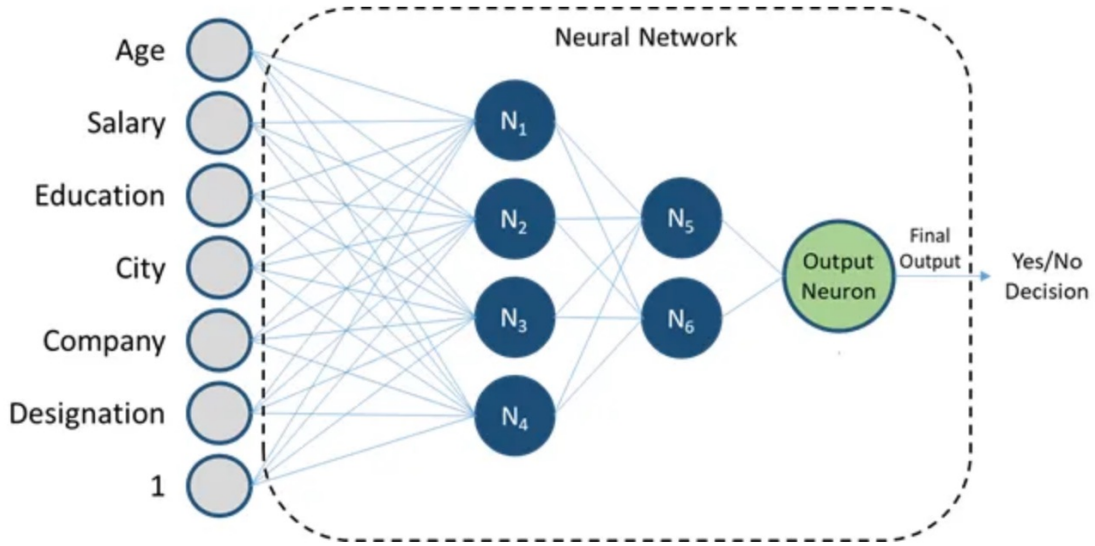


Example

- Now, just like for this layer, our inputs were the data's variables (age, salary, education etc), we can have this layer's outputs (O1, O2, O3, O4) flowing into the next layer as its inputs.
- We can keep adding layers like this. To complete the Neural Network, we add an output neuron which takes in the outputs from the last layer as its input and returns the final output (the Yes/No decision for our use-case).

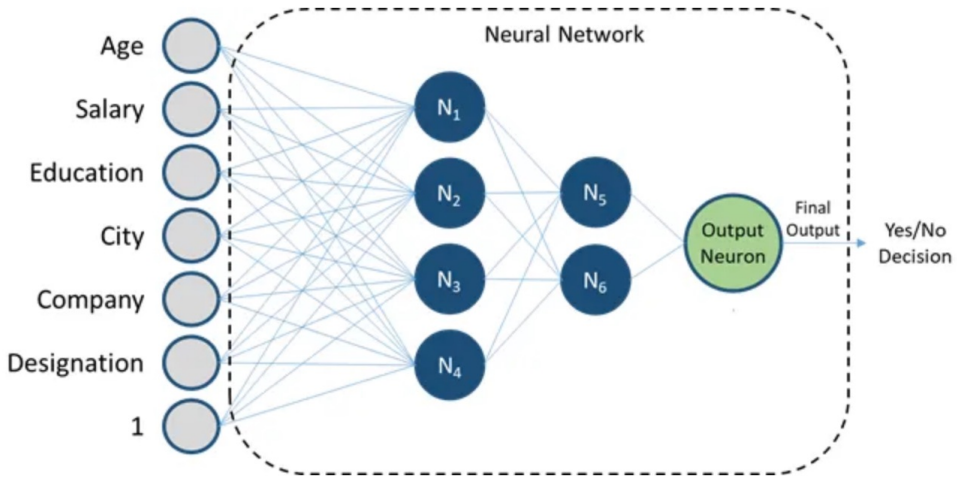


Example



- Each blue or green circle above is a neuron having its weights. These weights have to be "learned" by the model, just like in Logistic Regression, where the model learned what slopes gave the results closest to the training data.
- In DL, the networks learn the weights through a process called Back-propagation and Gradient Descent.
- Once the networks learn the optimum weights for all neurons, the input data is sent into the Neural Network, each neuron multiplying its inputs and weights, applying the activation on the sum and sending it to the next layer until we have the final result.

Example



- The blue layers (layers between the input and the final output neuron) are called the hidden layers. These are what makes the Model "Deep".
- The hidden layer neurons try to capture the latent patterns within the data that will help the model in predicting the creditworthiness of the individual.
- For example,
  - One of the neurons could be inferring whether the applicant is earning enough to cover the average living expense of his city
  - One neuron could be focusing on the company and designation to get an idea of their job security
  - Another could be focusing on, say, Age, Education, and Designation to get an idea about their performance on the job. etc.

Example

- A neuron focusing on a certain variable just means that it'll have higher magnitude (positive or negative) weights for that variable. The neurons will automatically adjust their weights using gradient descent algorithms to give outputs closest to the data on which they're trained.
- These inferences help the NN to make a better decision about the applicant's eligibility. Insights like the candidate's salary adequacy, job security, performance, etc. will be better indicators to judge his/her eligibility than directly using the input variables(Age, Salary, Education etc).
- However, the mentioned inferences are just examples of what the neurons may be figuring out. In reality, it would be challenging to inspect the neural network (checking the neurons' weights). It's also possible that we find patterns that may not be aligned to our general domain understanding or be directly interpretable by humans but have been identified by the NN as relevant for making predictions about the given data. Hence Neural Networks are also referred to as Black Box Models.

R Example

- To build a simple Neural Network (NN) in R, we could use the 'neuralnet' package. This package offers a user-friendly interface to develop, train, and evaluate neural networks.
- The dataset we will use is the famous Iris dataset. It contains measurements of four features: sepal length, sepal width, petal length, and petal width.
- These measurements come from three different species of iris flowers: setosa, versicolor, and virginica.
- Our goal is to build a network on this dataset to make predictions on the species of iris flowers based on their physical measurements.

```
## Rows: 150
## Columns: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.…
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.…
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.…
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.…
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s…
```

R Example

```
set.seed(245)
data_rows <- floor(0.7 * nrow(iris))
train_indices <- sample(c(1:nrow(iris)), data_rows)
train_data <- iris[train_indices,]
test_data <- iris[-train_indices,]
```

- To begin, the first step is to split the data into training and testing sets.
- In this example, We allocate 70% of the data for training and the rest for testing.

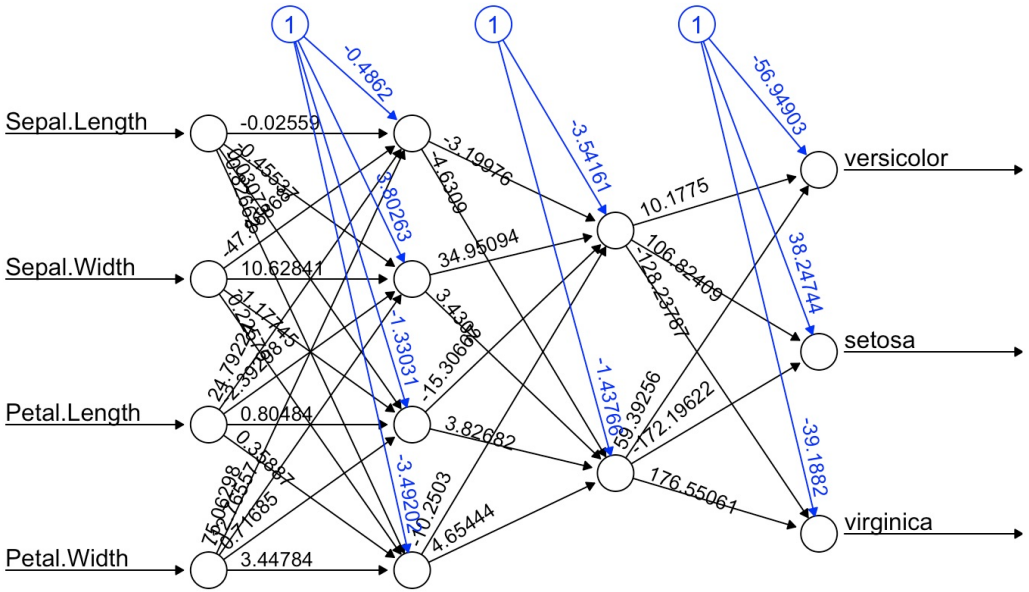
R Example

```
model = neuralnet(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
                  data = train_data,
                  hidden = c(4,2),
                  linear.output=FALSE,
                  learningrate = 0.001)
```

- We create a neural network model to predict the species (Species) based on the four features using the function 'neuralnet()'
- 'hidden=c(4,2)' specifies two hidden layers, the first layer with four neurons and the second with two neurons.
- 'linear.output=FALSE' is used for classification tasks (non-linear).
- 'learningrate = 0.001' specifies the learning rate used during the training of the neural network.
- The learning rate determines how much we adjust the model's parameters in response to the error it made on the last prediction. It's like setting the pace for a model's learning; too fast and the model might overshoot the target (optimal solution), too slow and it might take too long to reach it, or get stuck in a less optimal solution. An optimal learning rate helps the model to converge more quickly and reliably to a good solution.

R Example

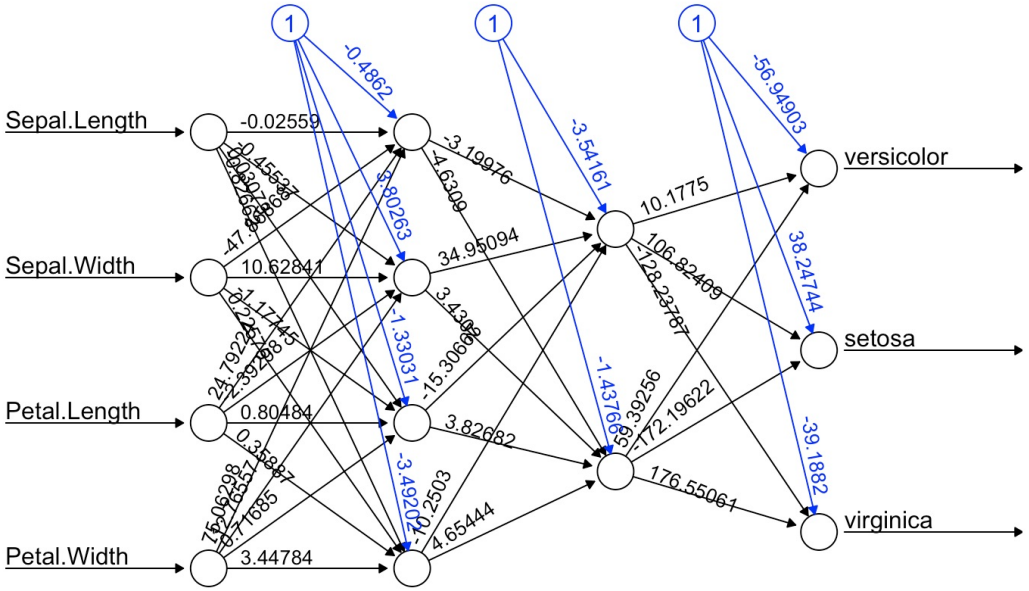
```
plot(model,rep = "best")
```



- To view our model architecture, we will use the ‘plot‘ function. It requires a model object and ‘rep‘ argument.
- ’rep = ”best”’ argument shows the best-performing network if multiple are trained.
- Each neuron is represented by a circle, and the lines connecting these circles to the input layer and to each other represent the weights. The weights are the numerical values that the network has learned during training.

R Example

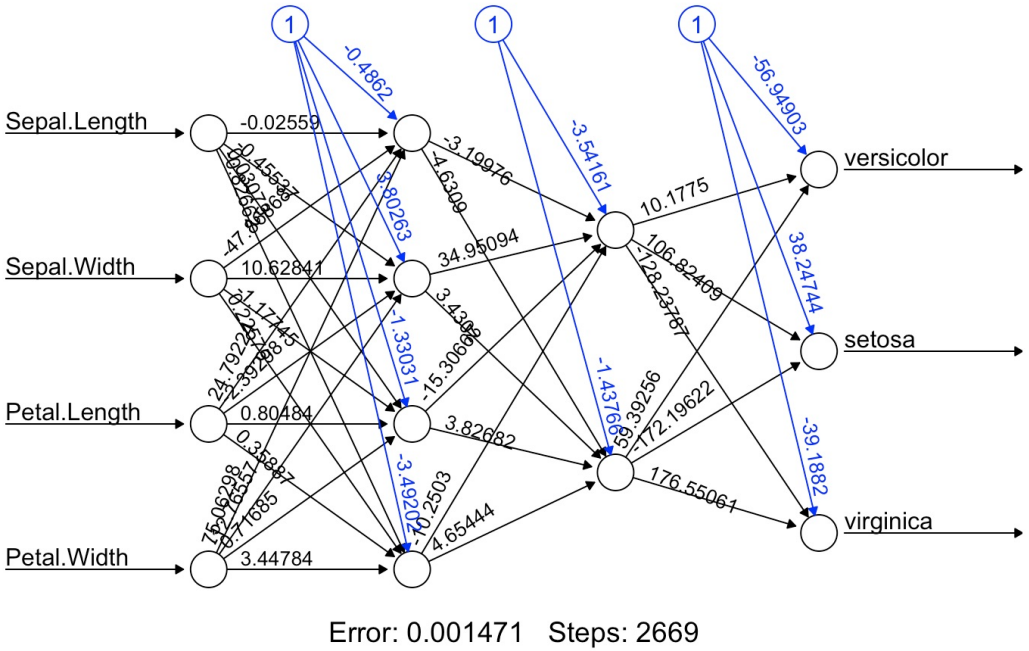
```
plot(model,rep = "best")
```



- The blue lines represent the connections that were most influential for the final prediction in the "best" repetition of the model training.
- The three nodes on the right represent the output layer of the network, with each corresponding to one species.
- The output layer shows how the neural network combines the inputs from the hidden layers to make a prediction.

R Example

```
plot(model,rep = "best")
```



- At the bottom, the error of the neural network is noted as 0.001471, which is a measure of how well the network's predictions match the actual data. A lower error rate indicates better performance.
- "Steps: 2669" indicates the number of iterations the training algorithm took to converge to a solution.

Today

Learning goals

- Basics of Deep Learning and Neural Networks
- Deep Learning Applications
- Architecture of Neural Networks
- Build and Interpret Neural Networks in R