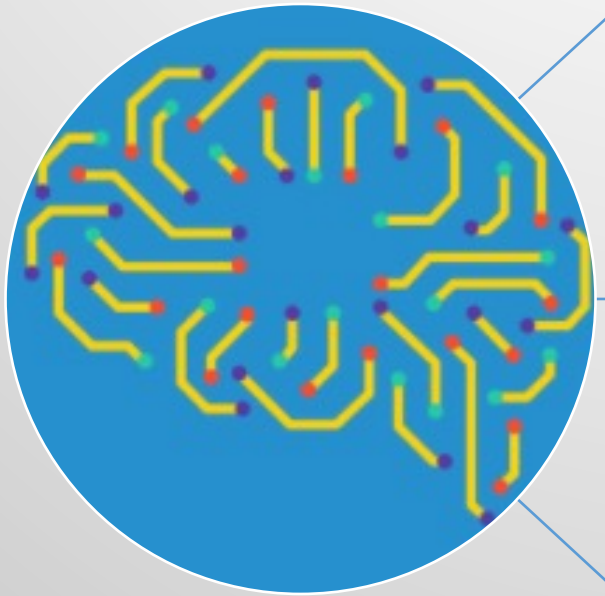
# Natural Language Processing

Dr. Karen Mazidi

# Part Five:
# Machine Learning



Topics

- Neural Networks

Quizzes

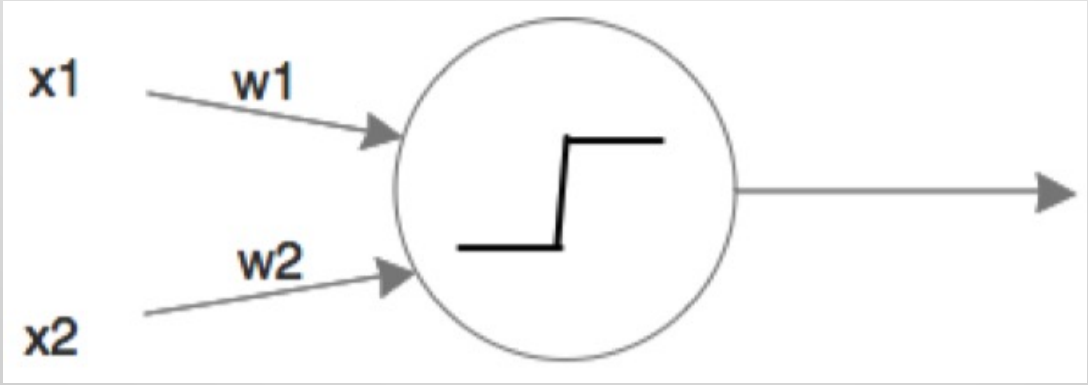
- Q: Neural networks

Homework

- Homework: Federalist Papers

# ANNs

- The idea for artificial neural networks has been around since the 50s; what was missing: data and computing power
- Frank Rosenblatt developed a pattern recognition machine in the late 1950s based on the idea of a perceptron
- Weighted inputs, step function activation

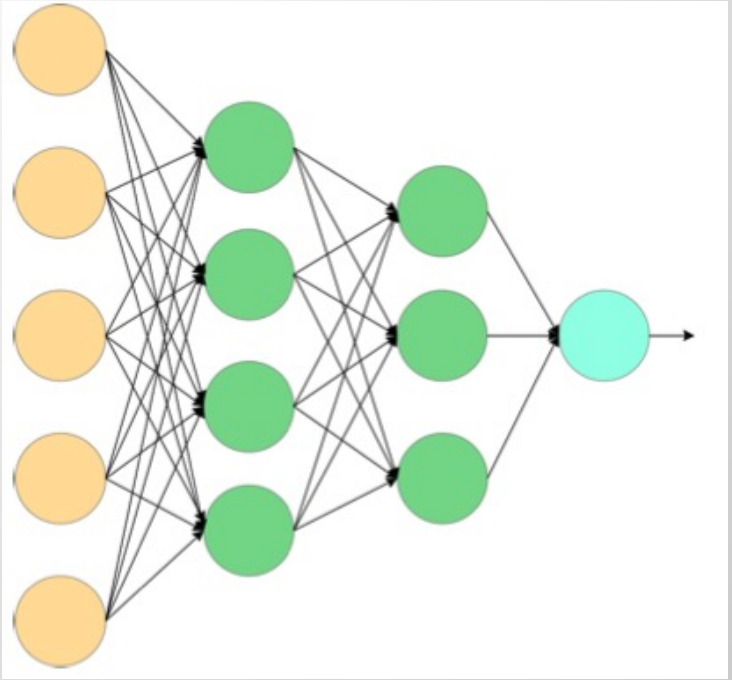


# Neural inspiration

- Is a neural network ‘like the human brain’?

# Neural network

- Feed-forward neural network with
- 5 input nodes/neurons
- 1 output node
- Two hidden layers with 4 and 3 nodes, respectively
- Each green node can learn a different function from the inputs



# Neural network regression example

- See the online notebook
- Scaling the data will help the network converge faster

**Code 23.1.1 — Neural Network.** Scale the Data

```
# scale the data using sklearn functionality
from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

# Train the model

- MLPRegressor – multi-layer perceptron regressor
- Set a seed for reproducibility

**Code 23.1.2 — Neural Network.** Train

```
# train the algorithm
from sklearn.neural_network import MLPRegressor

regr = MLPRegressor(hidden_layer_sizes=(6, 3), max_iter=500,
            random_state=1234)
regr.fit(X_train, y_train)
```

# Predict and evaluate

**Code 23.1.3 — Neural Network.** Predict and evaluate

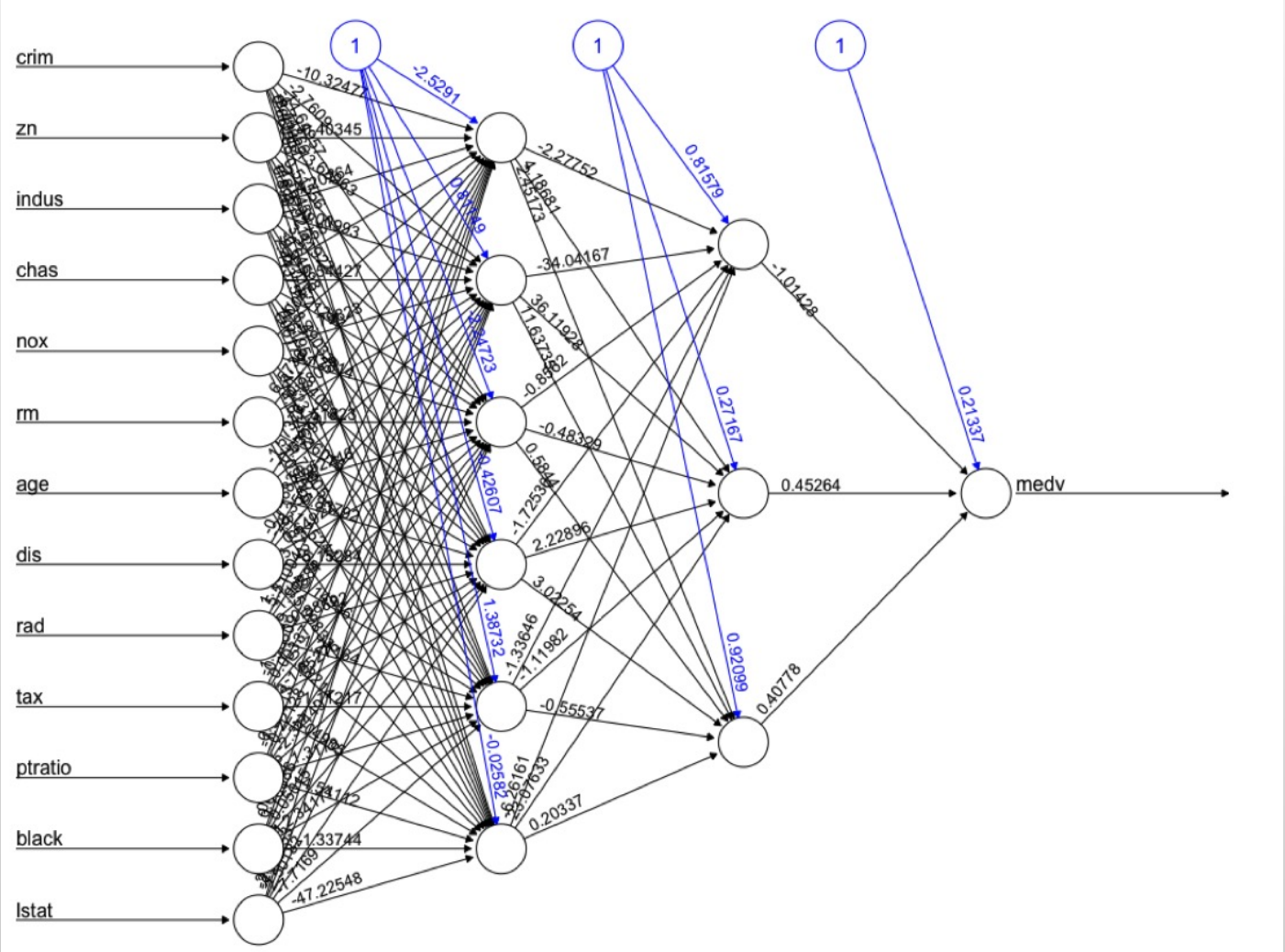
```
# make predictions
y_pred = regr.predict(X_test)

# evaluation
from sklearn.metrics import mean_squared_error, r2_score
print('mse=', mean_squared_error(y_test, y_pred))
print('correlation=', r2_score(y_test, y_pred))
```

- Results comparing linear regression, net1 and net2
- Net2: lbfgs solver, increased max iterations

| | Linear Regression | Neural Network 1 | Neural Network 2 |
|---|---|---|---|
| correlation | 0.73 | 0.71 | 0.89 |
| mse | 27.44 | 29.31 | 11.56 |

Table 23.1: Results on the Boston Housing Data

# Plot from R neuralnet( )



# Hidden layers and nodes

- The network architecture, or topology, is often found through trial and error
- Too few nodes: underfitting
- Too many nodes: overfitting, especially on small data
- Rules of thumb:

  - between 1 and the number of predictors
  - two-thirds of the input layer size plus the size of the output layer
  - < twice the input layer size

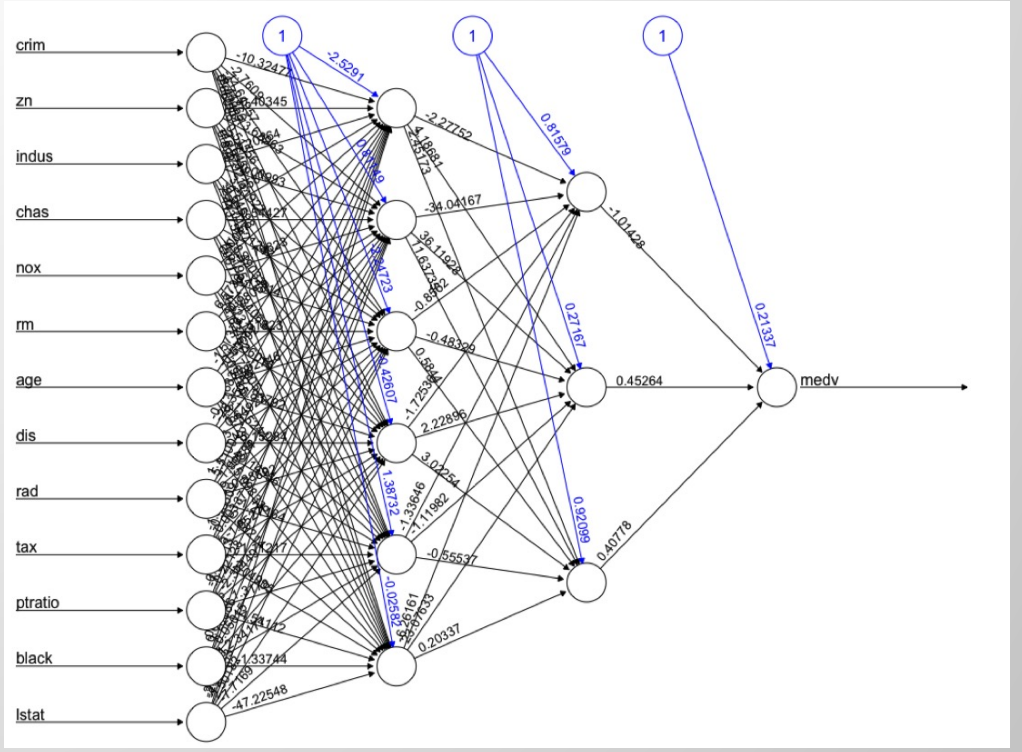
  Following these very general guidelines for the Boston data with 13 predictors, the suggestions are: (1) 1 - 13, (2) 9, and (3) < 26. We tried 9 hidden nodes and arranged them in two layers.

# Number of layers

- This data got much worse results with one layer of 9 nodes, compared to 2 layers (6, 3)
- Having two layers can capture more complex (not strictly linear) relationships in the data
- But the more complex the network, the more likely you are to overfit
- For small data, start with a simple network, then try bigger, more complex networks

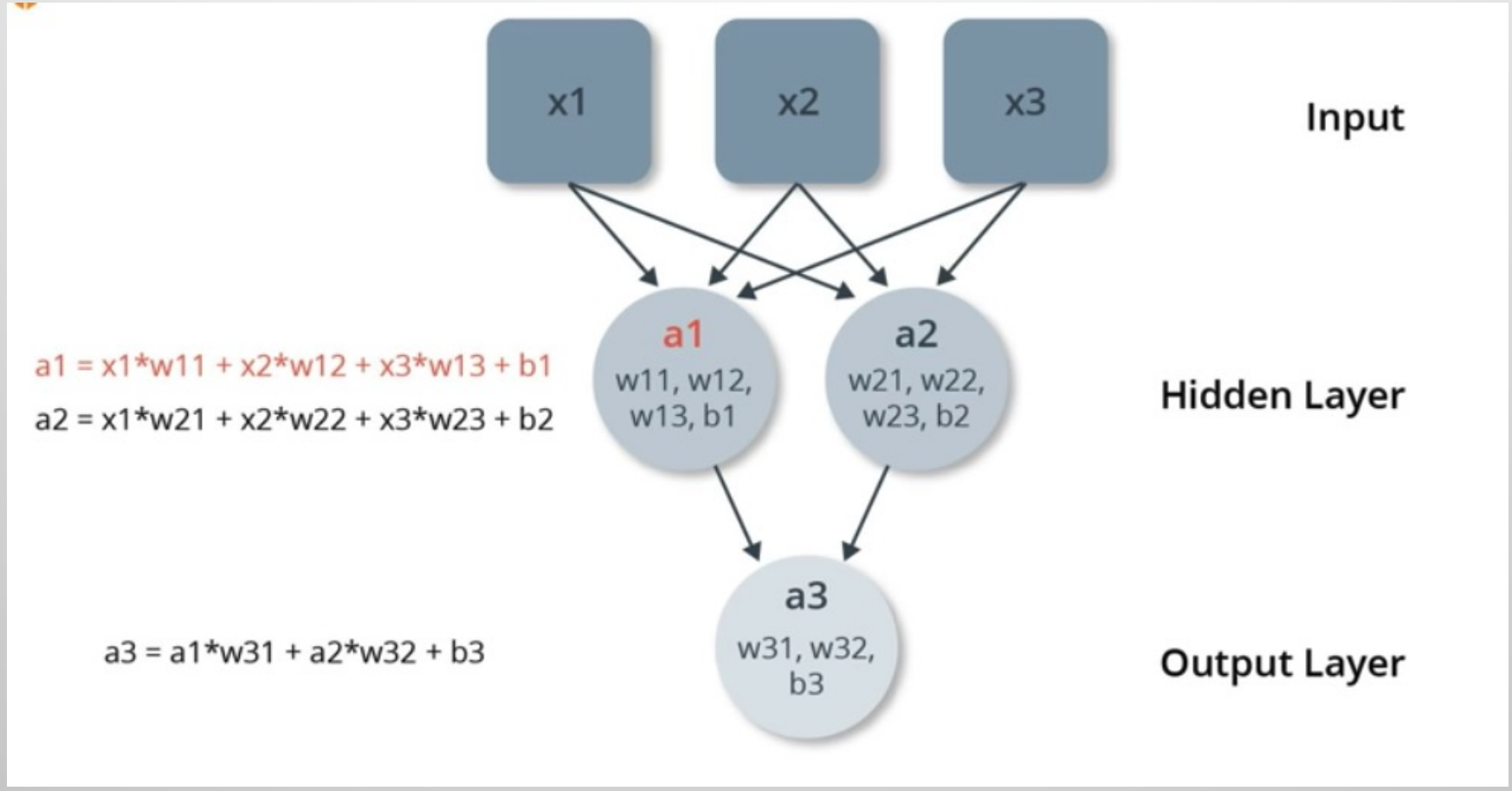
# Feed forward

- In a feed-forward network, every node is connected to every node in the following layer
- Sometimes called densely connected neural network
- Every neuron has its own bias term and weight
- Each input to a node is multiplied by the weight to get a sum of inputs
- This is just matrix multiplication



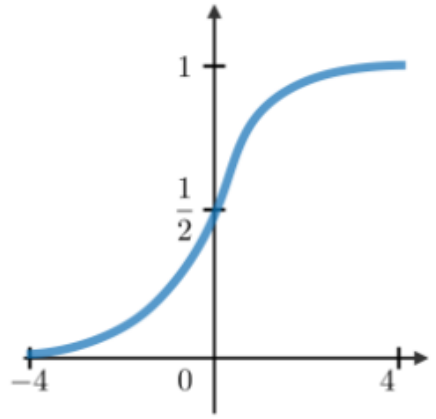
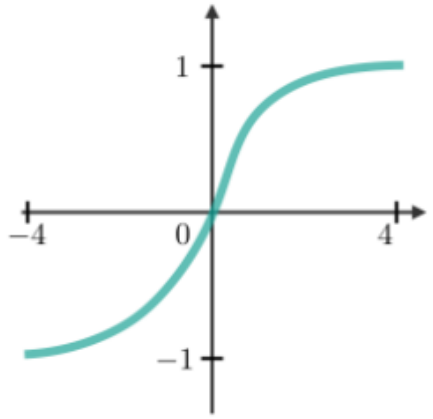
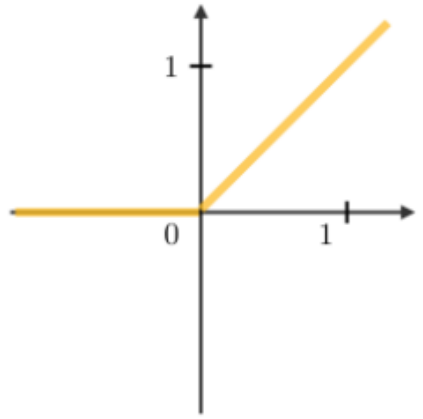
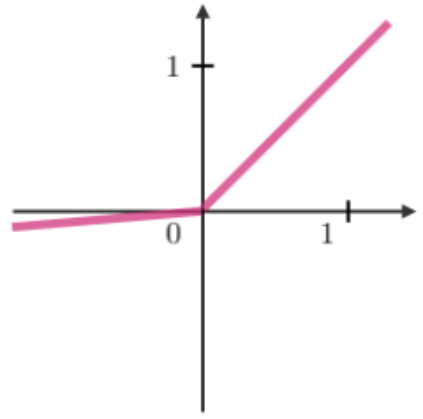
# Feed forward

- A different view: ‘forward’ is top to bottom



# Activation functions

- An **activation** function inputs the weighted sum and transforms it for output via the activation
- Some common activation functions:

| Sigmoid | Tanh | ReLU | Leaky ReLU |
| --- | --- | --- | --- |
| $g(z) = \frac{1}{1+e^{-z}}$ | $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ | $g(z) = \max(0, z)$ | $g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$ |
|  |  |  |  |

# Back propagation

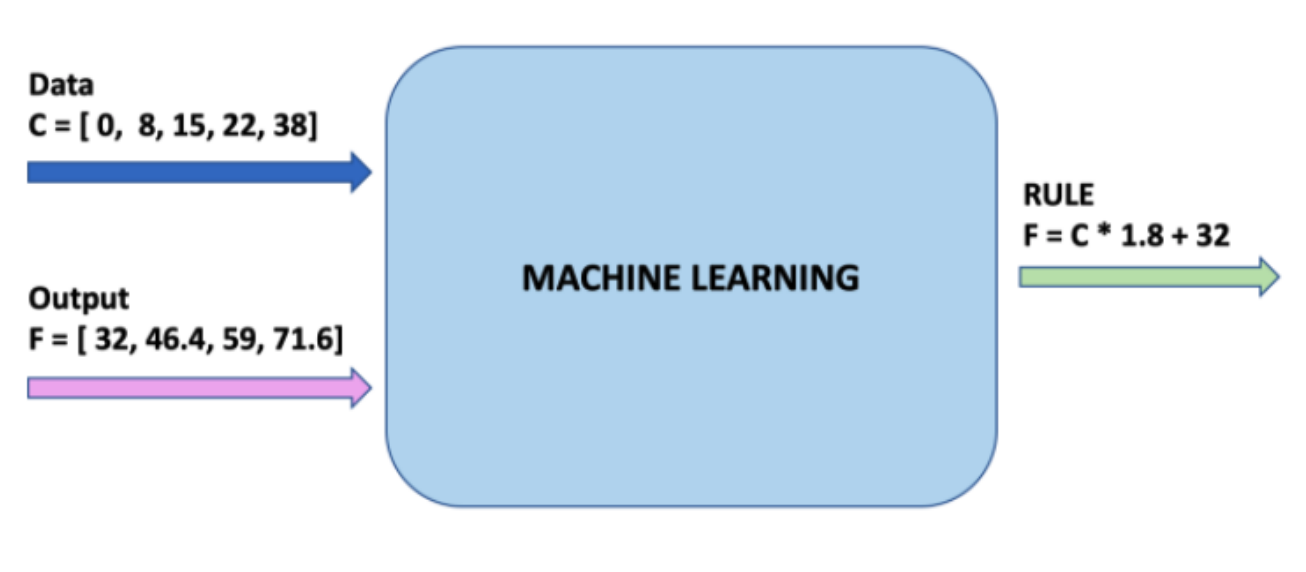
- Where the learning happens
- The feed-forward network results in an output value
- This output value is compared to the true value to compute the loss
- In a backward pass, the weights are adjusted in a process called back propagation
- A gradient (slope) matrix is calculated by taking the derivative of the loss function

# Learning

- Weights start off with random values
- Learning takes place in forward-backward passes called epochs
- The data flows forward, a loss is calculated, credit (blame) is assigned backwards to adjust weights
- Each weight is adjusted according to its own gradient
- Some gradients are steeper than others

# Learning

- Learn a simple model:



# Learning

- Loss is calculated
- Weights are adjusted backward

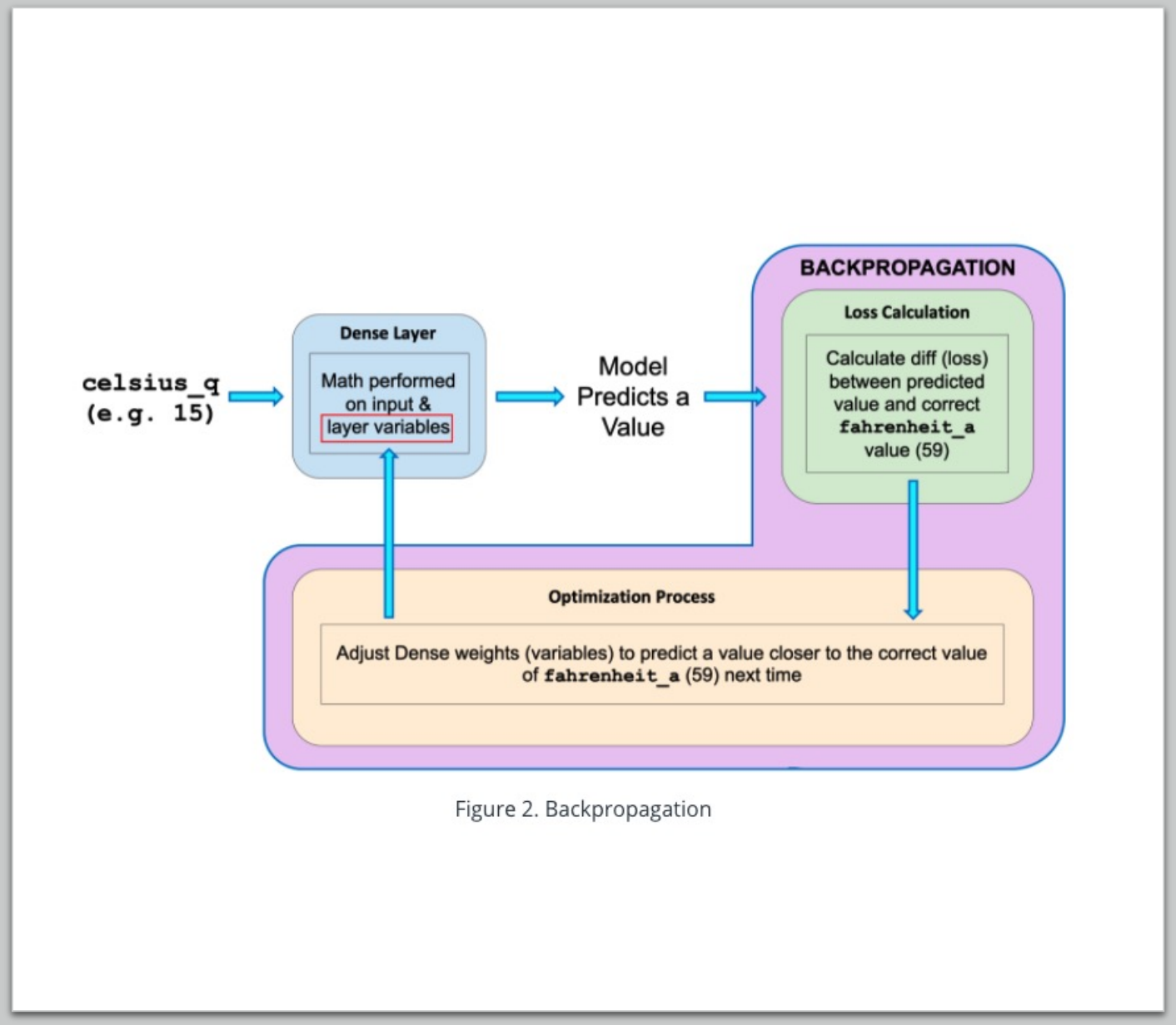



Figure 2. Backpropagation

# Loss function

- For regression or classification, MSE can be used:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{n} \|(y_i - f(x_i))\|^2$$

where $\|x\| := \sqrt{x_1^2 + \ldots + x_n^2}$

# Credit assignment problem

- How much error is assigned to each weight
- Take the derivative of the cost function with respect to each neuron's output
- Gradient matrix: $\nabla E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right)$

- In stochastic gradient descent, the gradients are computed for each training example, then aggregated
- The new weights are computed from the old rates and multiplied by the learning rate

$$\triangle w_i = -\alpha \frac{\partial E}{\partial w_i}$$

# Batches

- Stochastic gradient descent – randomly choose one example
- Batch gradient descent – train all examples in one big batch
- Mini-batch gradient descent – train on a subset of examples at a time

# 3Blue1Brown for a deep dive into the math

- What is a neural network: https://www.youtube.com/watch?v=aircAruvnKk
- Gradient descent: https://www.youtube.com/watch?v=IHZwWFHWa-w
- Back propagation: https://www.youtube.com/watch?v=Ilg3gGewQ5U
- Backprop calculus: https://www.youtube.com/watch?v=tIeHLnjs5U8

# Classification Examples

Part 5 Chapter 22

- 20 news
- spam

# Training

- More complex models take longer to train
- For large data, try architectures on a subset to speed up the trial-and-error process
- May reach max iterations before converging below threshold, both parameters can be adjusted
- See parameters here:
- MLP Classifier
- MLP Regressor


# Essential points to note

Define a network:

- Number of layers
- Number of neurons/nodes in each layer
- Activation function
- ‘solver’

# Next topic

Deep Learning