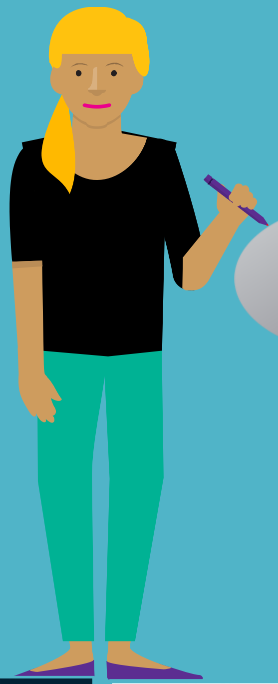
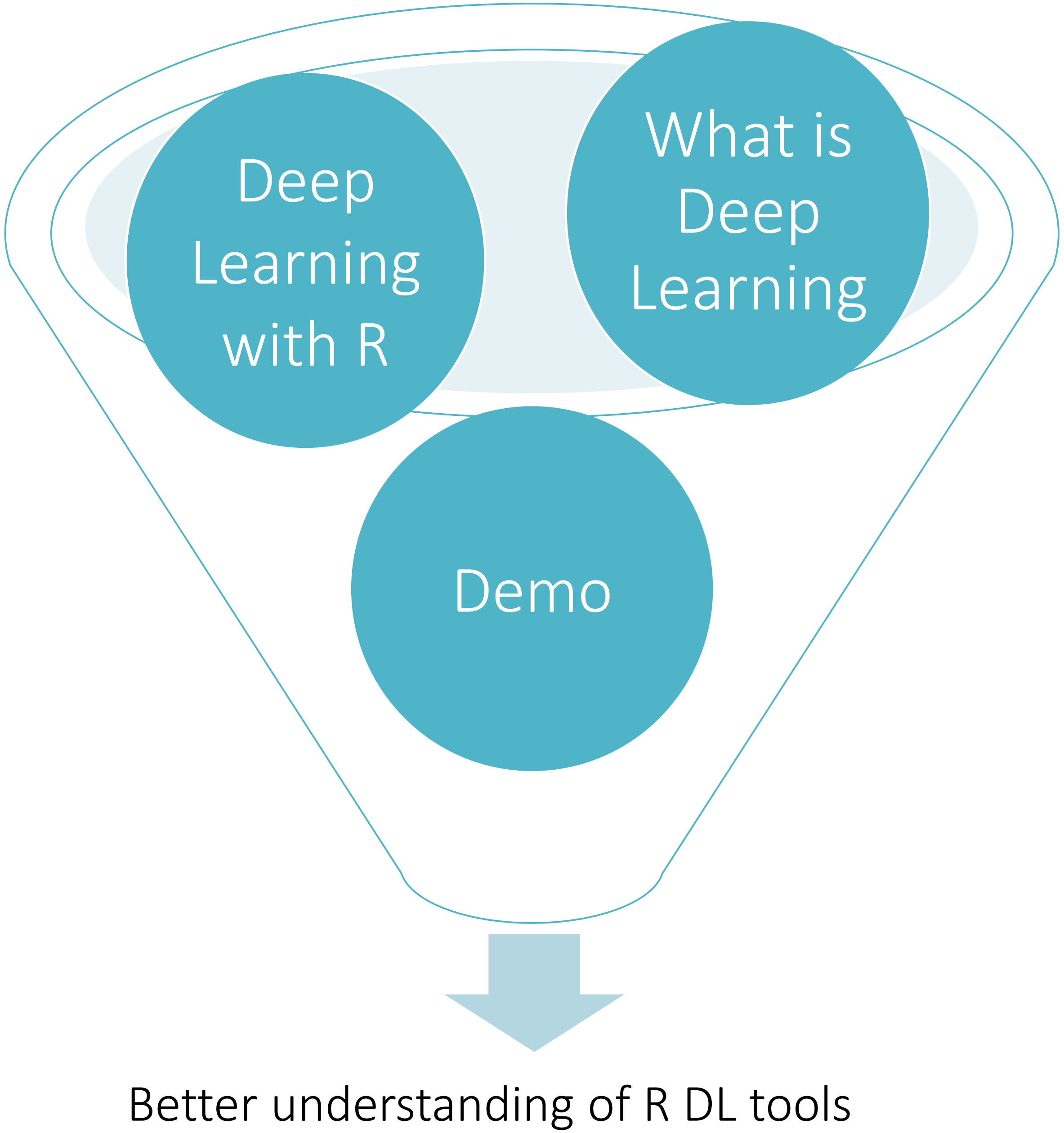
Deep Learning with R

Francesca Lazzeri - @frlazzeri

Data Scientist II - Microsoft, AI Research



Agenda


Deep Learning with R
What is Deep Learning
Demo
Better understanding of R DL tools

What is Deep Learning

Fundamental concepts in Deep Learning

Forward Propagation Algorithm	Activation Functions
Gradient Descent	Backpropagation

What is Deep Learning

Fundamental concepts in Deep Learning

Forward Propagation Algorithm

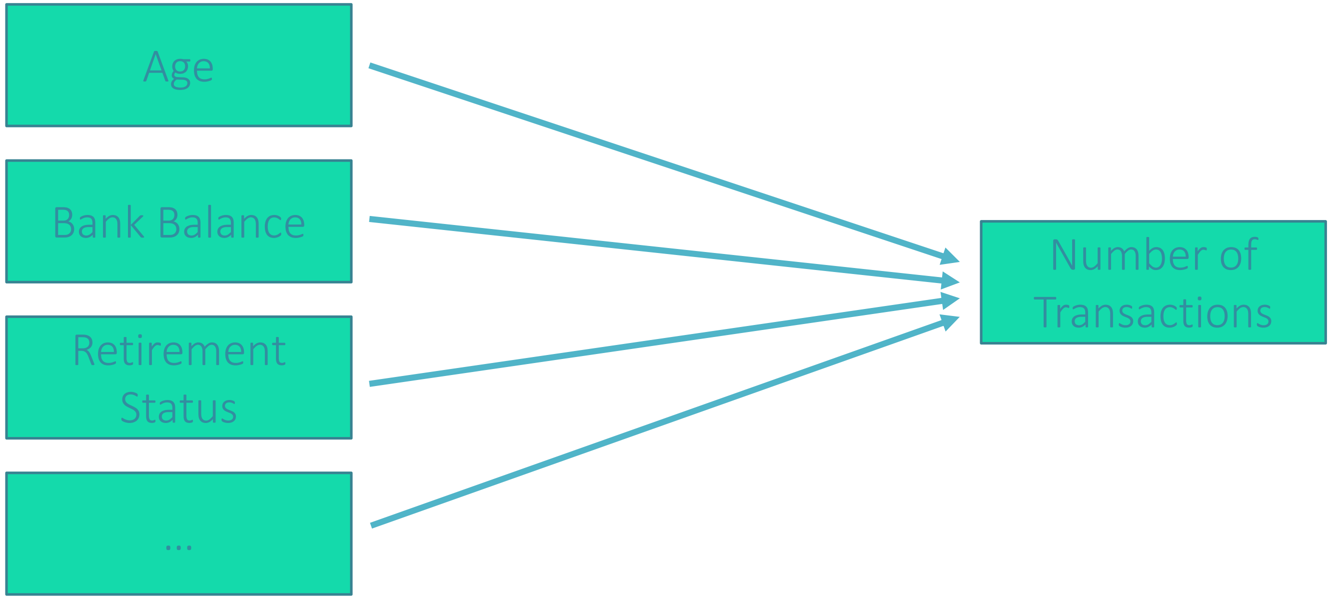
Activation Functions

Gradient Descent

Backpropagation

What is Deep Learning

Example as seen by linear regression



What is Deep Learning

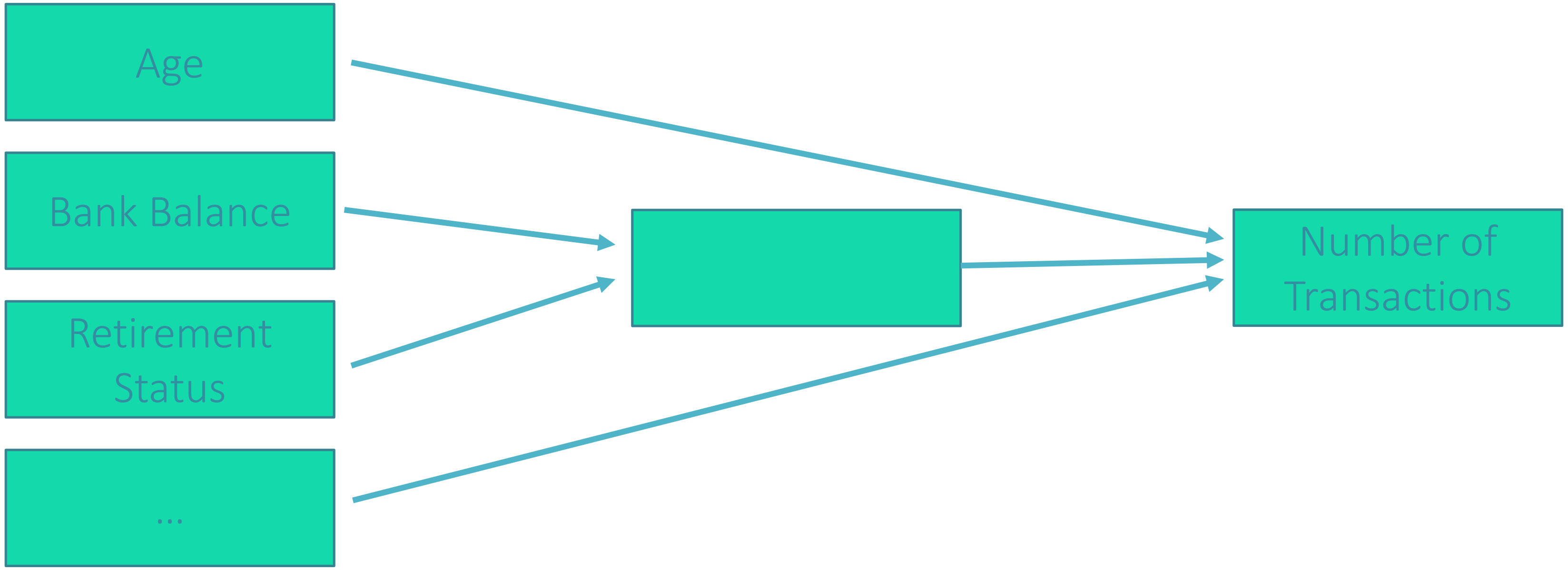
Interactions

- Neural networks account for interactions really well

- Deep learning uses especially powerful neural networks for:
  - Text
  - Images
  - Videos
  - Audio
  - Source code

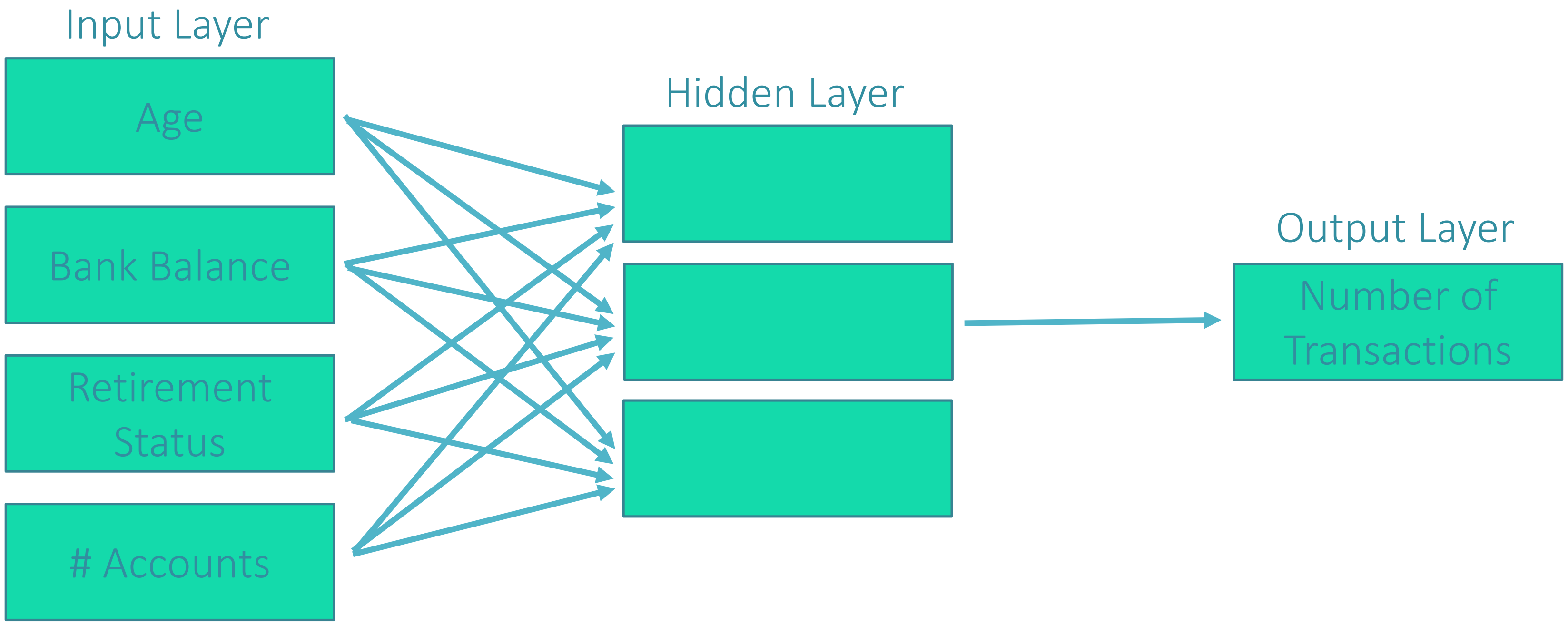
What is Deep Learning

Deep learning models capture interactions



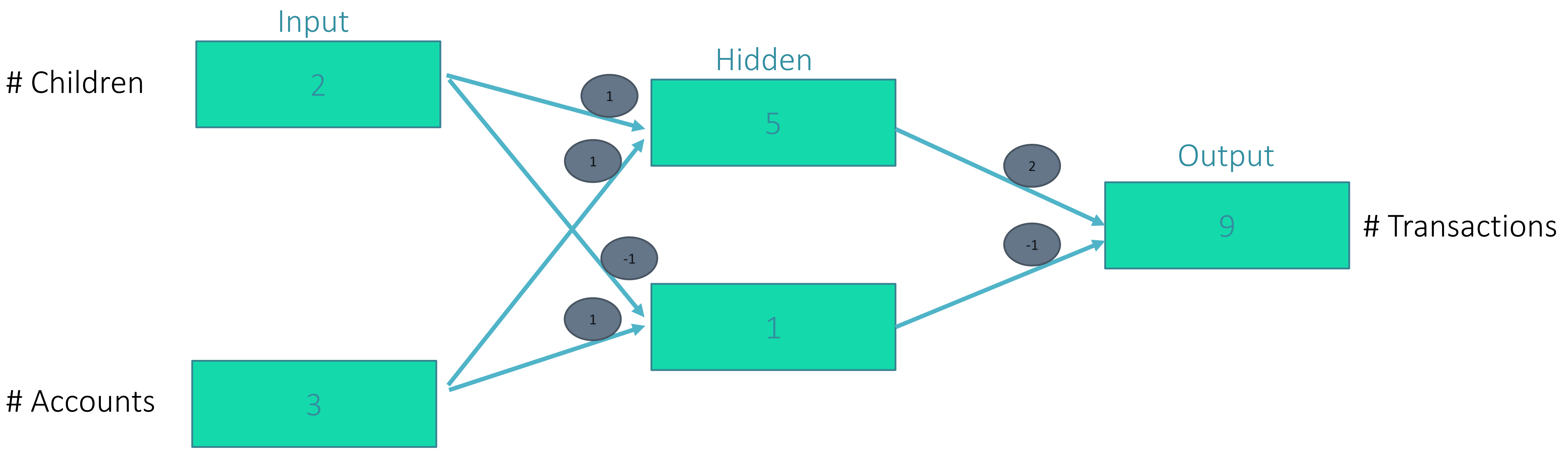
What is Deep Learning

Interactions in neural networks



What is Deep Learning

Forward Propagation Algorithm



What is Deep Learning

Fundamental concepts in Deep Learning

Forward Propagation Algorithm

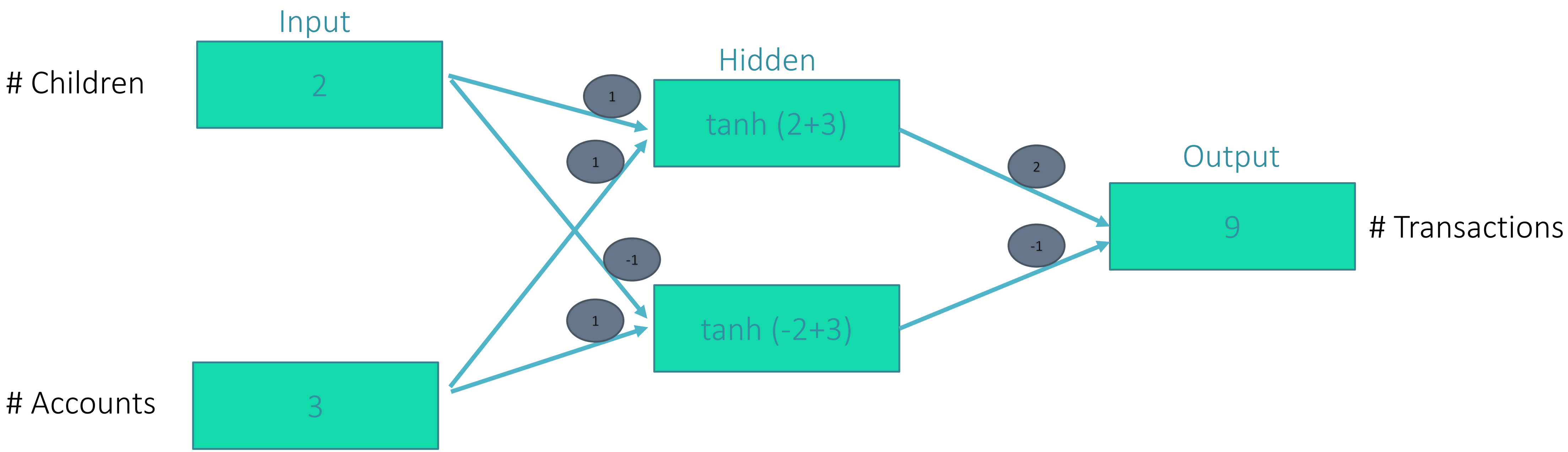
Activation Functions

Gradient Descent

Backpropagation

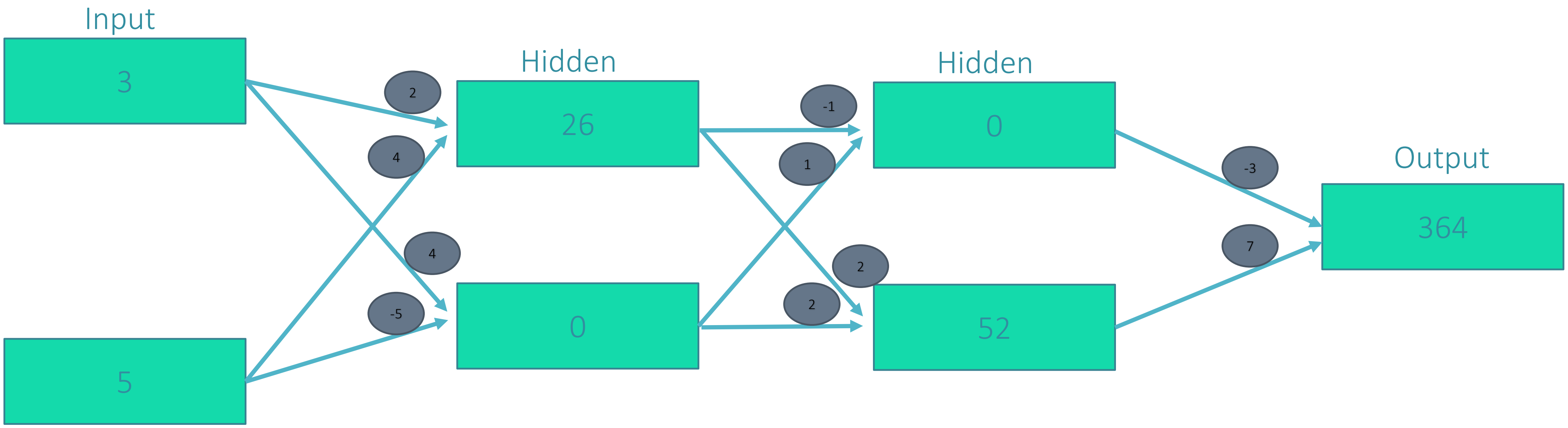
What is Deep Learning

Activation Functions



What is Deep Learning

ReLU Activation Function



What is Deep Learning

Representation Learning

- Deep networks internally build representations of patterns in the data
- Partially replace the need for feature engineering
- Subsequent layers build increasingly sophisticated representations of raw data
- Modeler doesn’t need to specify the interactions
- When you train the model, the neural network gets weights that find the relevant patterns to make better predictions

What is Deep Learning

Fundamental concepts in Deep Learning

Forward Propagation Algorithm	Activation Functions
Gradient Descent	Backpropagation

What is Deep Learning

The Need for Optimization

- Predictions with multiple points
  - Making accurate predictions gets harder with more points
  - At any set of weights, there are many values of the error
  - Correspond to the many points we make predictions for
- Loss function
  - Aggregate errors in predictions from many data points into single number
  - Measure of model's predictive performance

What is Deep Learning

The Need for Optimization

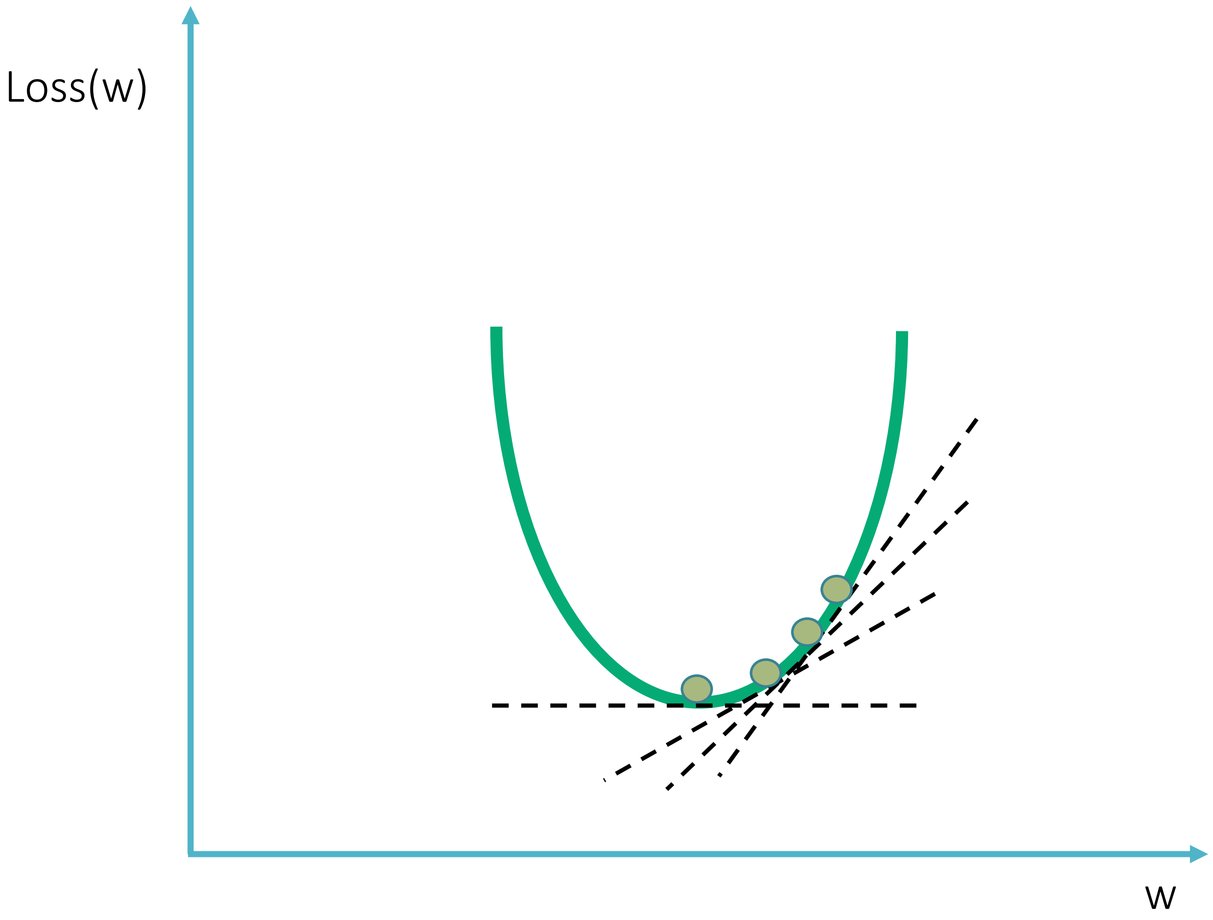
- Squared error loss function

Prediction	Actual	Error	Squared Error
10	20	-10	100
8	3	5	25
6	1	5	25

- Total Squared Error: 150
- Mean Squared Error: 50
- Lower loss function value means a better model
- Goal: find the weights that give the lowest value for the loss function
- Gradient descent!

What is Deep Learning

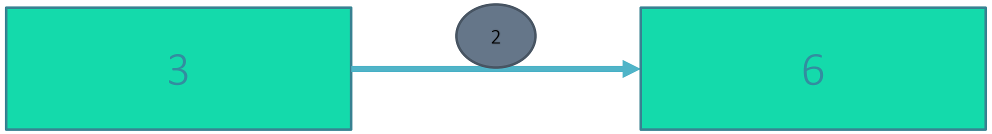
Gradient Descent



What is Deep Learning

Gradient Descent

- Slope calculation example



Actual Target Value = 10

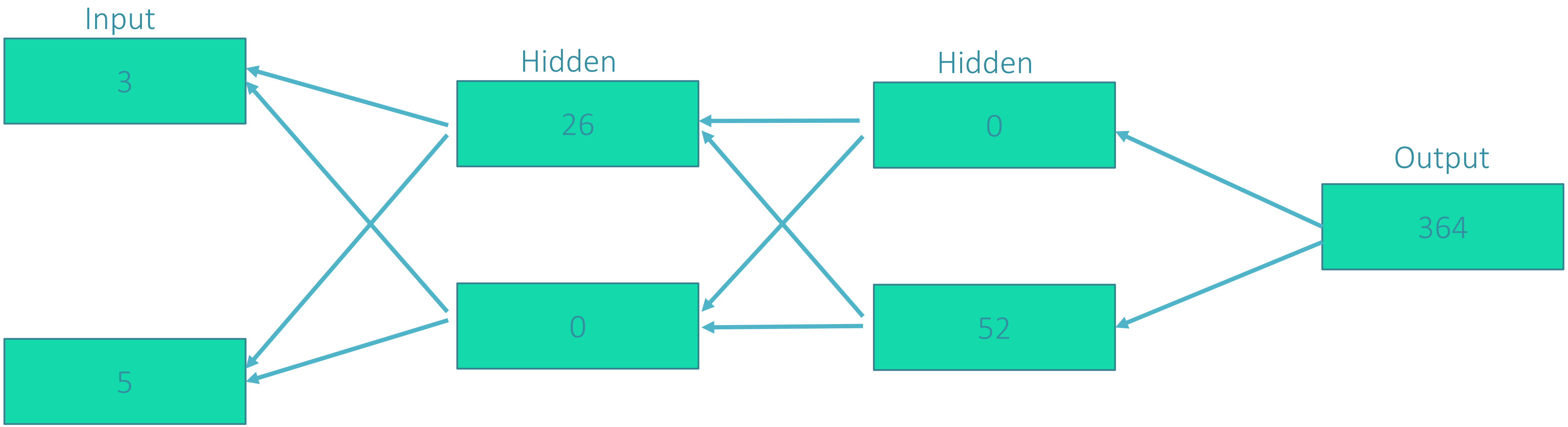
- To calculate the slope for a weight, need to multiply:
  * Slope of the loss function w.r.t value at the node we feed into
  * The value of the node that feeds into our weight
  * Slope of activation function w.r.t value we feed into

What is Deep Learning

Fundamental concepts in Deep Learning

Forward Propagation Algorithm	Activation Functions
Gradient Descent	**Backpropagation**

Backpropagation



What is Deep Learning

Backpropagation

- Allows gradient descent to update all weights in neural network (by getting gradients for all weights)

- Go back one layer at a time

- Important to understand the process, but you will generally use a library that implements this

Deep Learning with R

MXNetR

- Feed-forward neural network
- Convolutional neural network (CNN)

darch

- Restricted Boltzmann machine
- Deep belief network

deepnet

- Feed-forward neural network
- Restricted Boltzmann machine
- Deep belief network
- Stacked autoencoders

H2O

- Feed-forward neural network
- Deep autoencoders

deepr

- Simplify some functions from H2O
- Deepnet packages

Deep Learning with R

Model/Dataset	MNIST		Iris		Forest Cover Type	
	Accuracy (%)	Runtime (sec)	Accuracy (%)	Runtime (sec)	Accuracy (%)	Runtime (sec)
MXNetR (CPU)	98.33	147.78	83.04	1.46	66.8	30.24
MXNetR (GPU)	98.27	336.94	84.77	3.09	67.75	80.89
darch 100	92.09	1368.31	69.12	1.71	–	–
darch 500/300	95.88	4706.23	54.78	2.1	–	–
deepnet DBN	97.85	6775.4	30.43	0.89	14.06	67.97
deepnet DNN	97.05	2183.92	78.26	0.42	26.01	25.67
H2O	98.08	543.14	89.56	0.53	67.36	5.78
Random Forest	96.77	125.28	91.3	2.89	86.25	9.41

Deep Learning with R

R interface to Keras

Keras for R | Home | Articles | Examples | Reference | News

R interface to Keras

Overview

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.* Keras has the following key features:

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.
- Is capable of running on top of multiple back-ends including TensorFlow, CNTK, or Theano.

This website provides documentation for the R interface to Keras. See the main Keras website at https://keras.io for additional information on the project.

Demo

Deep Learning with R on Azure with Keras and CNTK

DSVM

Microsoft CNTK

CNTK

R & Keras

Azure

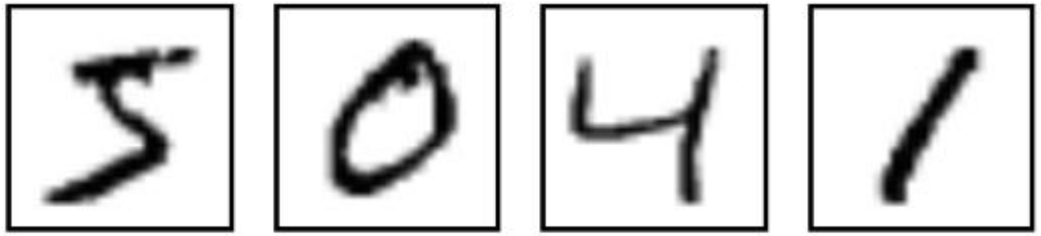
Demo

Keras Workflow Steps to Build your Model

- Specify architecture
- Compile the model
- Fit the model
- Predict

Demo

Preparing the Data



```
library(keras)
install_keras()
```

```
library(keras)
mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y
```

```
# reshape
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
# rescale
x_train <- x_train / 255
x_test <- x_test / 255
```

```
y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)
```

Demo

Defining the Model

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')
```

Demo

Defining the Model

```
summary(model)
```

```
Model
________________________________________________________________________________
Layer (type)                    Output Shape                    Param #
================================================================================
dense_1 (Dense)                 (None, 256)                     200960
________________________________________________________________________________
dropout_1 (Dropout)             (None, 256)                     0
________________________________________________________________________________
dense_2 (Dense)                 (None, 128)                     32896
________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)                     0
________________________________________________________________________________
dense_3 (Dense)                 (None, 10)                      1290
================================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0
________________________________________________________________________________
```

Demo

Defining the Model

```
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)
```

Demo

Training and Evaluation

```
history <- model %>% fit(
  x_train, y_train,
  epochs = 30, batch_size = 128,
  validation_split = 0.2
)
```

```
model %>% evaluate(x_test, y_test)
```

```
$loss
[1] 0.1149

$acc
[1] 0.9807
```

```
model %>% predict_classes(x_test)
```

```
  [1] 7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7 1 2
 [40] 1 1 7 4 2 3 5 1 2 4 4 6 3 5 5 6 0 4 1 9 5 7 8 9 3 7 4 6 4 3 0 7 0 2 9 1 7 3 2
 [79] 9 7 7 6 2 7 8 4 7 3 6 1 3 6 9 3 1 4 1 7 6 9
 [ reached getOption("max.print") -- omitted 9900 entries ]
```

References

http://blog.revolutionanalytics.com/2017/08/keras-and-cntk.html

http://www.rblog.uni-freiburg.de

https://keras.rstudio.com/

https://campus.datacamp.com/courses/deep-learning

http://gluon.mxnet.io

Thank You!

Francesca Lazzeri - @frlazzeri
Data Scientist II - Microsoft, AI Research