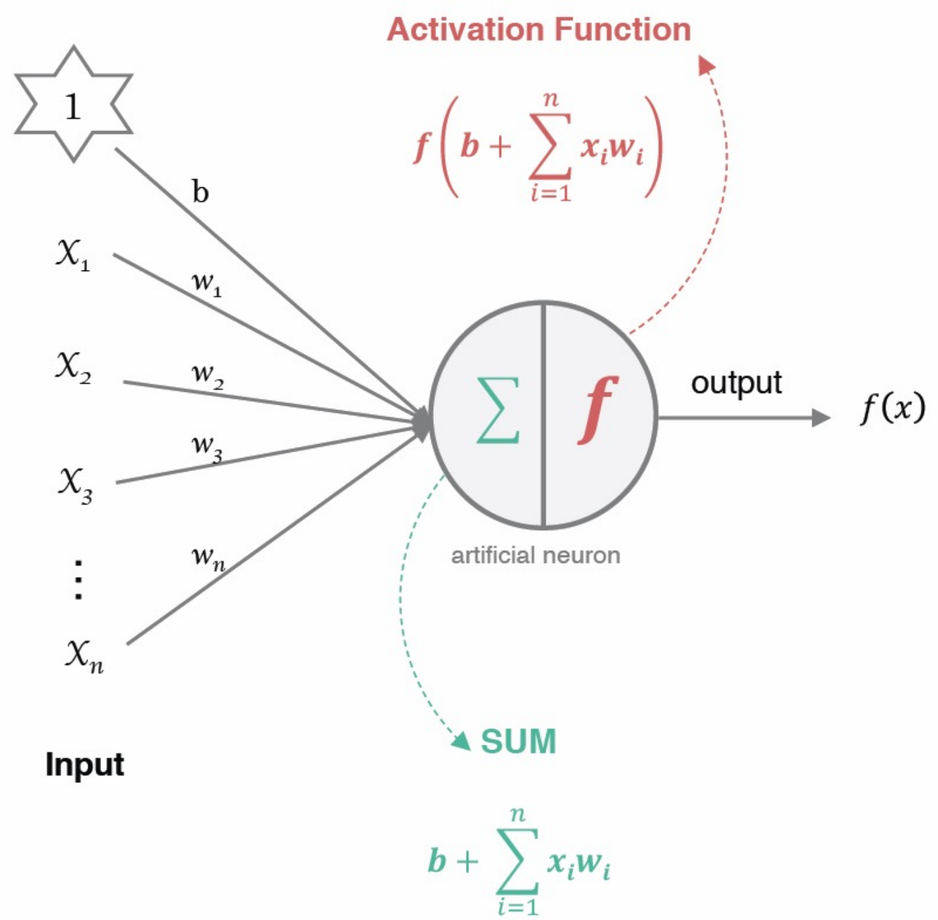
DLI Accelerated Data Science Teaching Kit

Module 16.2 Activation Function and Perceptron

How does artificial neuron work ?

Artificial neurons are most basic units of information processing in an artificial neural network.

Each neuron takes information from a set of neurons, processes it, and gives the result to another neuron for further processing.



How does artificial neuron work ?

All the inputs ***x*** are multiplied with their weights ***w***, and added together

$$weighted\ sum = b*1 + x_1*w_1 + \ldots + x_n*w_n = b + \sum_{i=1}^{n} x_i w_i$$

The weighted sum is passed to **activation function**

$$f(x) = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

What is Activation Function?

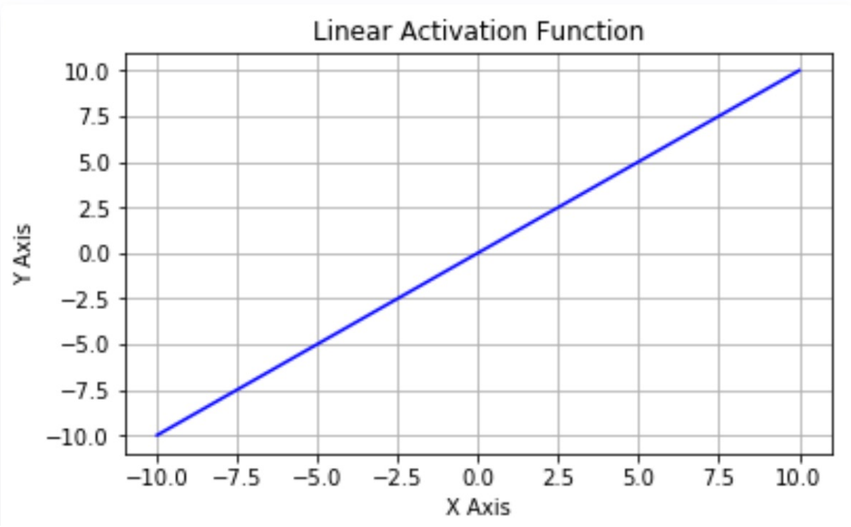
The activation function of a node defines the output of that node given an input or set of inputs.

Biologically inspired, it is usually an abstraction representing the rate of action potential firing in the cell.

These functions should be nonlinear to encode complex patterns of the data.

Linear activation function

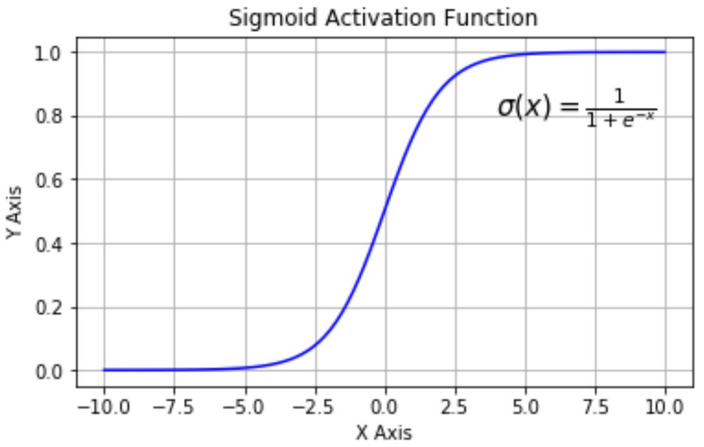
- $f(x) = x$

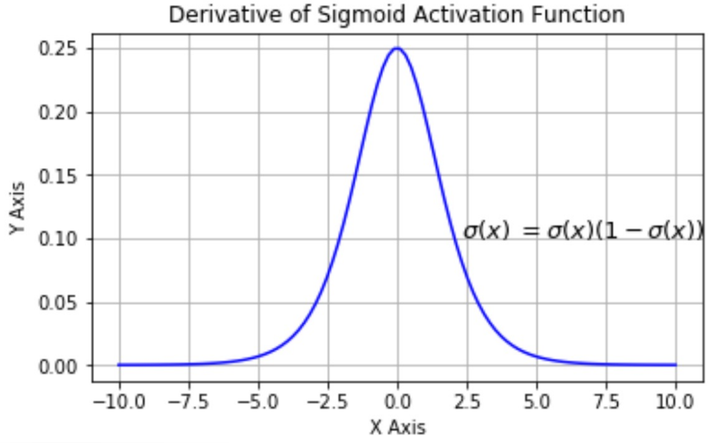


Non-linear Activation Function

Sigmod function

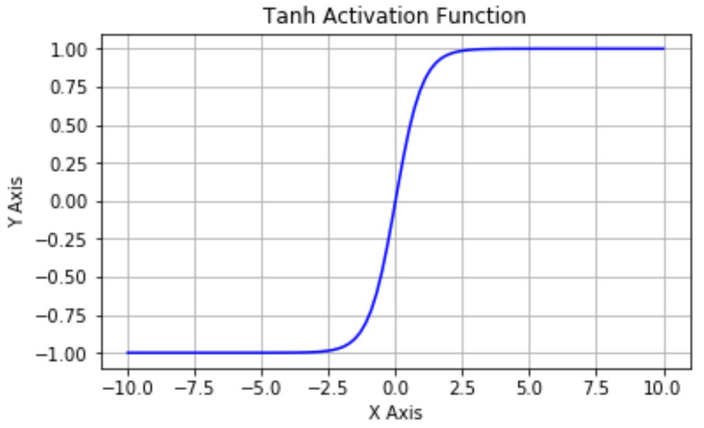
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

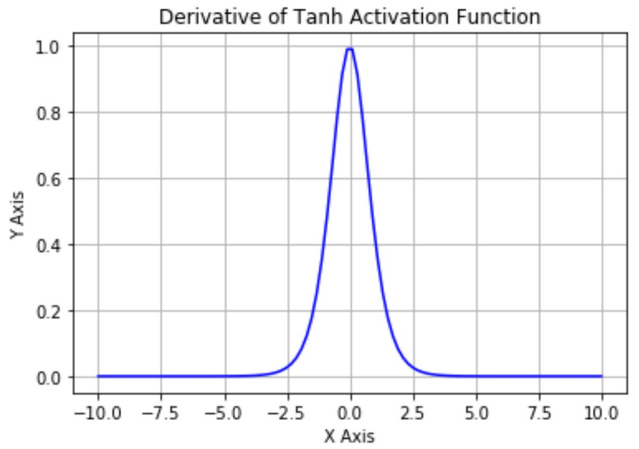




Tanh Function

$$tanhx = \frac{sinhx}{coshx} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

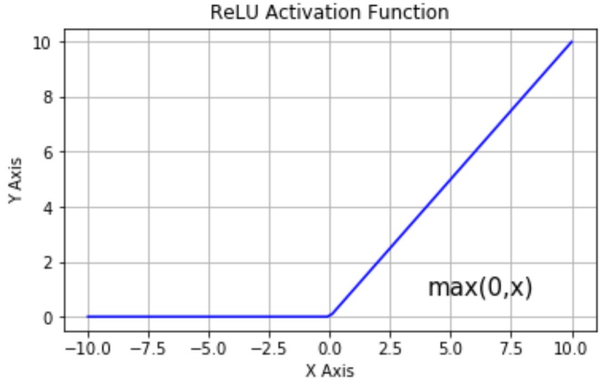


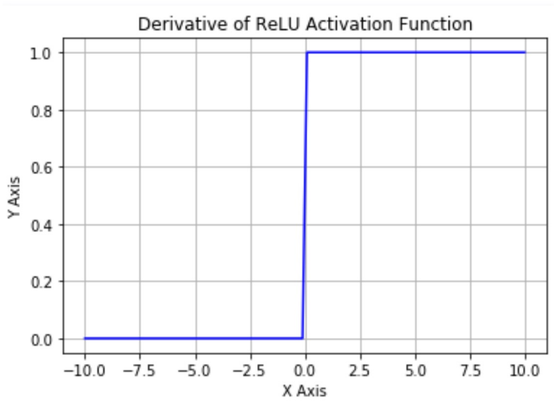


Non-linear Activation Function

Rectified Linear Function (ReLU)

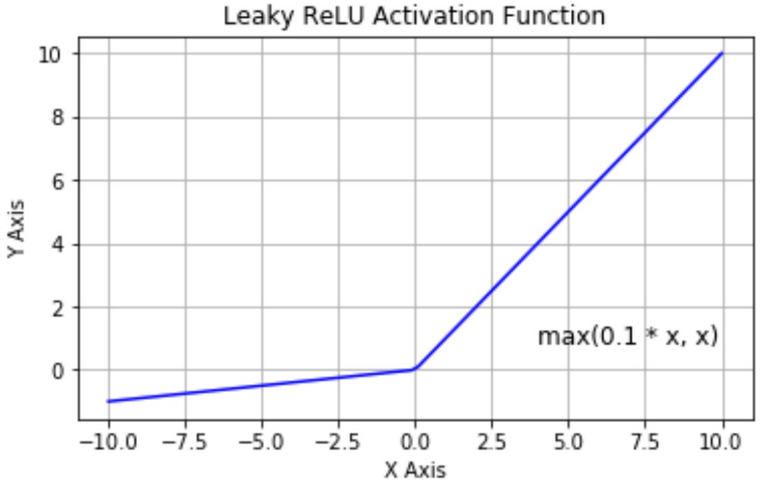
$$f(x) = \max(0, x)$$





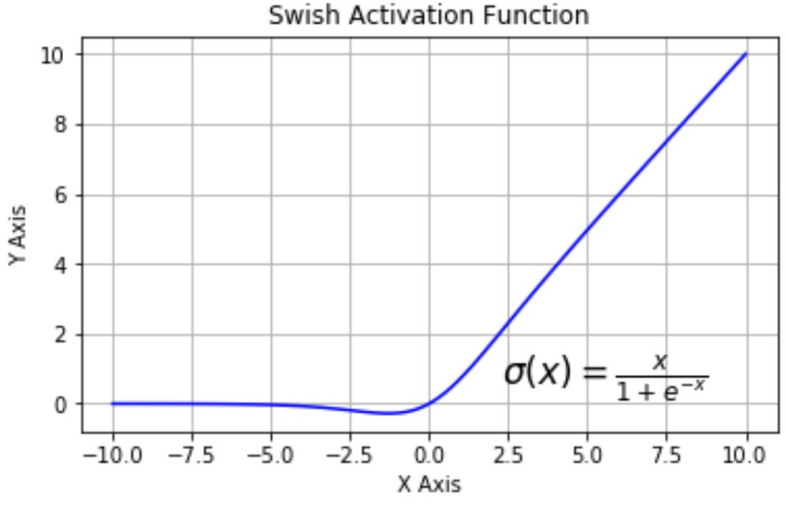
Leaky Rectified Linear Function (LReLU)

$$f(x) = max(0.1x, x)$$



Swish Function (Google)

$$\sigma(x) = \frac{x}{1 + e^{-x}}$$



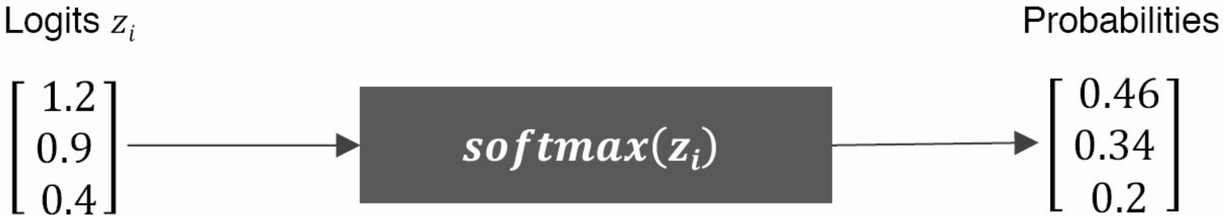
Non-linear Activation Function

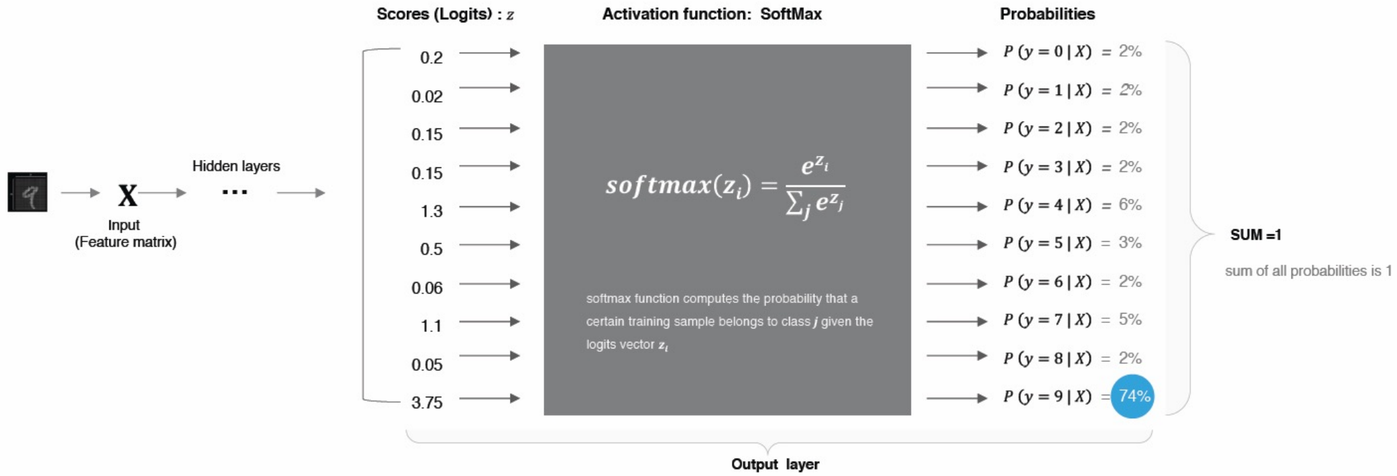
Softmax function

- Used in multiclass classification
- A generalization of Logistic Regression.
- It converts linear inputs to probabilities.
- Squashes the outputs of each unit to be between 0 and 1, just like a Logistic/Sigmoid function.
- Unlike logistic regression which is for the binary classification, softmax allows for classification into any number of possible classes.

$$\hat{p}_i = softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

for $i = 1, \ldots, j$





Which Activation Function is Optimal?

In Hidden Layer

- By Default use ReLu function
- Always keep in mind that ReLU function should only be used in the hidden layers
- Never use Sigmoid and TanH fucntion due to the vanishing gradient problem

In Output Layer

- Use SoftMax function for classification problem
- Use Linear function for regression problem

Perceptron: the simplest neural networks

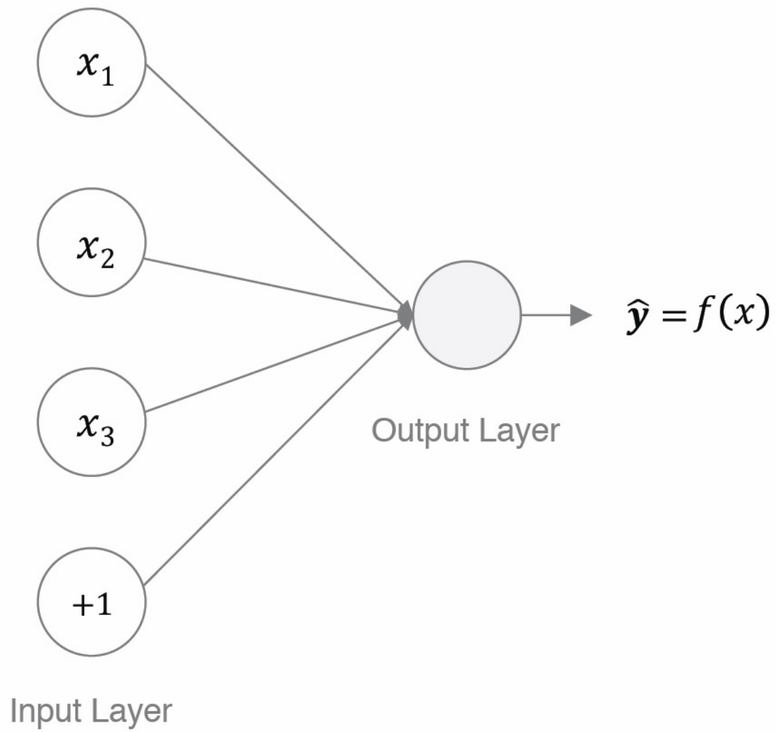
Simplest Feedforward Neural Network

It does not contain any hidden layers.

Bias nodes

- Increase the flexibility of the model to fit the data
- It allows you to move the line up and down to fit the prediction with the data better.
- Without b the line always goes through the origin (0, 0) .

$$b + \sum_{i=1}^{n} x_i w_i$$



Loss Function

$$E = \frac{1}{2}(y - \hat{y})^2$$

Loss function is used to measure exactly how wrong our predictions are.

The loss is high when the neural network makes a lot of mistakes, and it is low when it makes fewer mistakes.

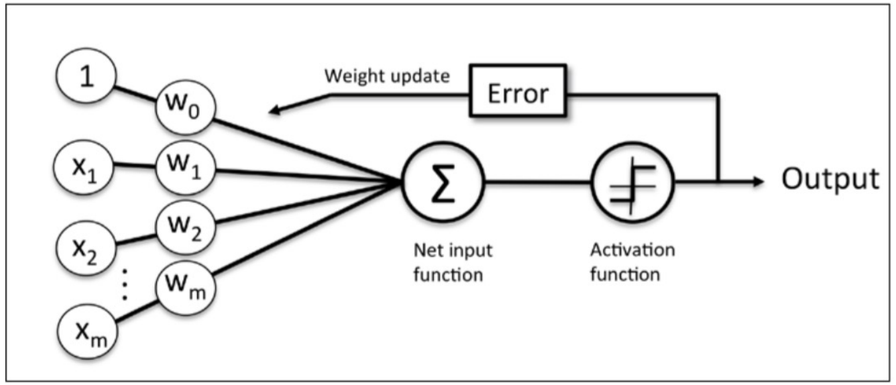
Adjust the weights to minimize the loss function

- Update the weight parameters iteratively in order to make the error (loss function) as small as possible.

$$w_{ij}^{new} = w_{ij}^{old} - \eta \frac{\partial E}{\partial w_{ij}}$$

Learning

Update weights based on errors



Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

Backpropagation

Activation function

$$a_{w,b}(x^i) = w^T x^i + b$$

Output

$$\hat{y^i} = a_{w,b}(x^i) = w^T x^i + b$$

Loss

$$J(w,b) = \frac{1}{2}(y^i - \hat{y}^i)^2$$

1

$$\frac{\partial J(w,b)}{\partial w} = (y^i - \hat{y^i}) \times \frac{\partial(-\hat{y^i})}{\partial w}$$
$$= (y^i - \hat{y^i}) \times \frac{\partial \hat{y^i}}{\partial w} \times (-1)$$
$$= (y^i - \hat{y^i}) \times x^i \times (-1)$$

$$\frac{\partial J(w,b)}{\partial b} = (y^i - \hat{y^i})| \times \frac{\partial(-\hat{y^i})}{\partial b}$$
$$= (y^i - \hat{y^i}) \times \frac{\partial \hat{y^i}}{\partial b} \times (-1)$$
$$= (y^i - \hat{y^i}) \times (-1)$$

2

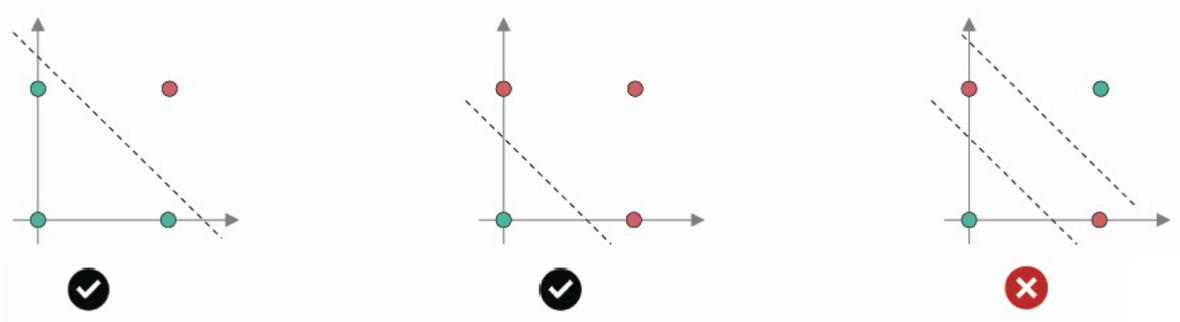
$$w = w - \eta \times \frac{\partial J(w,b)}{\partial w} = w - \eta \times (y^i - \hat{y^i}) \times x^i \times (-1)$$
$$= w + \eta \times (y^i - \hat{y^i}) \times x^i$$

$$b = b - \eta \times \frac{\partial J(w,b)}{\partial b} = b - \eta \times (y^i - \hat{y^i}) \times (-1)$$
$$= b + \eta \times (y^i - \hat{y^i})$$

Limitations of Perceptron

Only learn linearly separable problems

If the vectors are not linearly separable, learning will never reach a point where all vectors are classified properly.



DEEP
LEARNING
INSTITUTE

PRAIRIE VIEW
A&M UNIVERSITY

DLI Accelerated Data Science Teaching Kit

Thank You