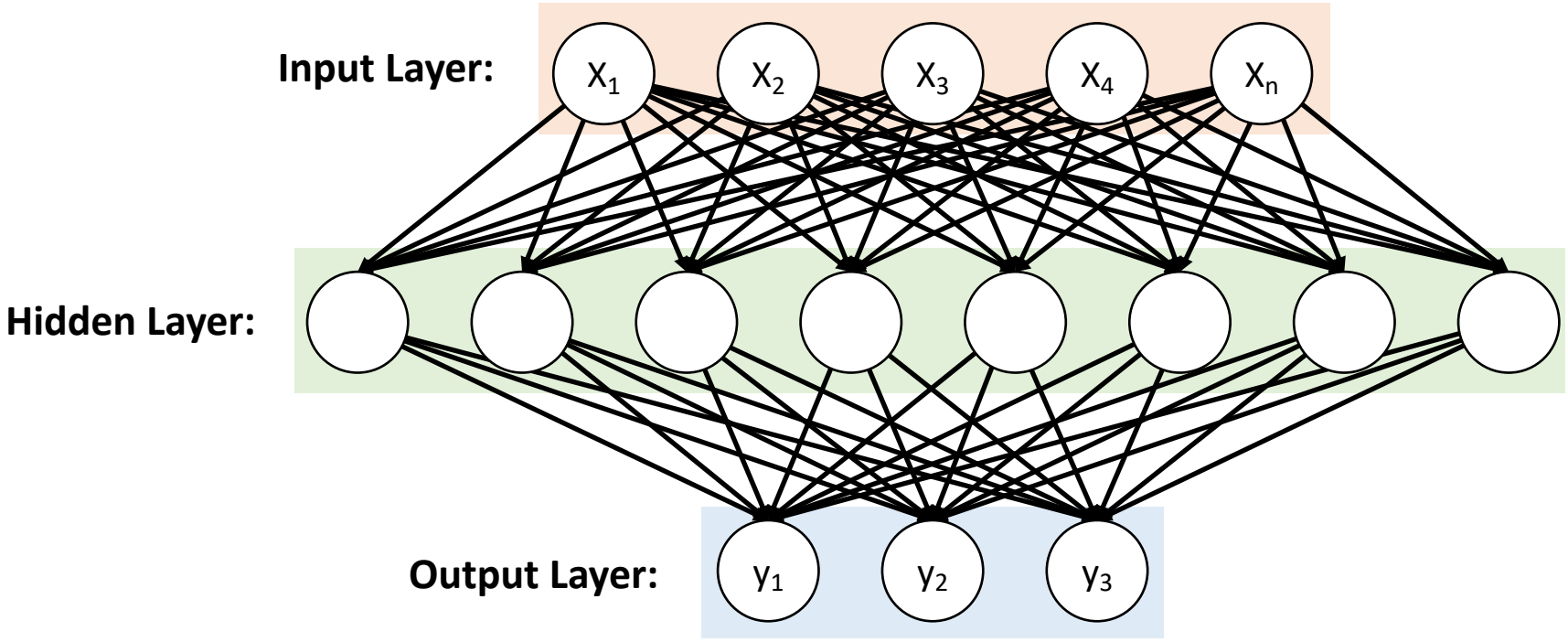
Machine Learning Workshop:
Neural Networks

Neural Networks

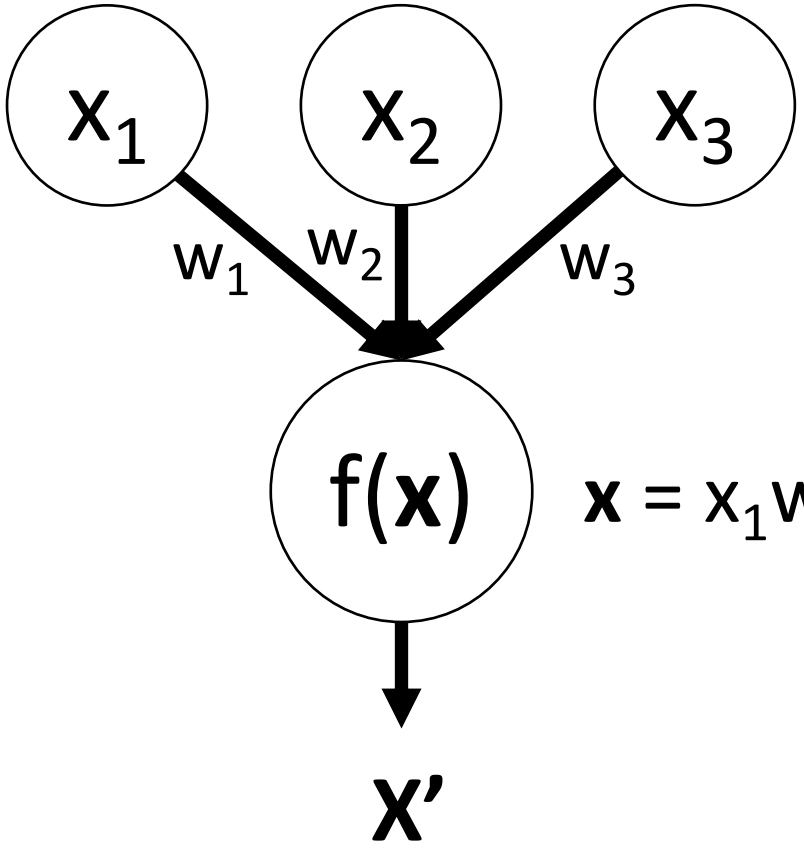
- The idea of **artificial neural networks** stems from **biological neural networks**

- The basic idea is that individual “neurons” or nodes in learn a unique function with respect to the input data, where combined these functions are able to produce an accurate model for classification

Neural Network Structure



Neural Network Neurons



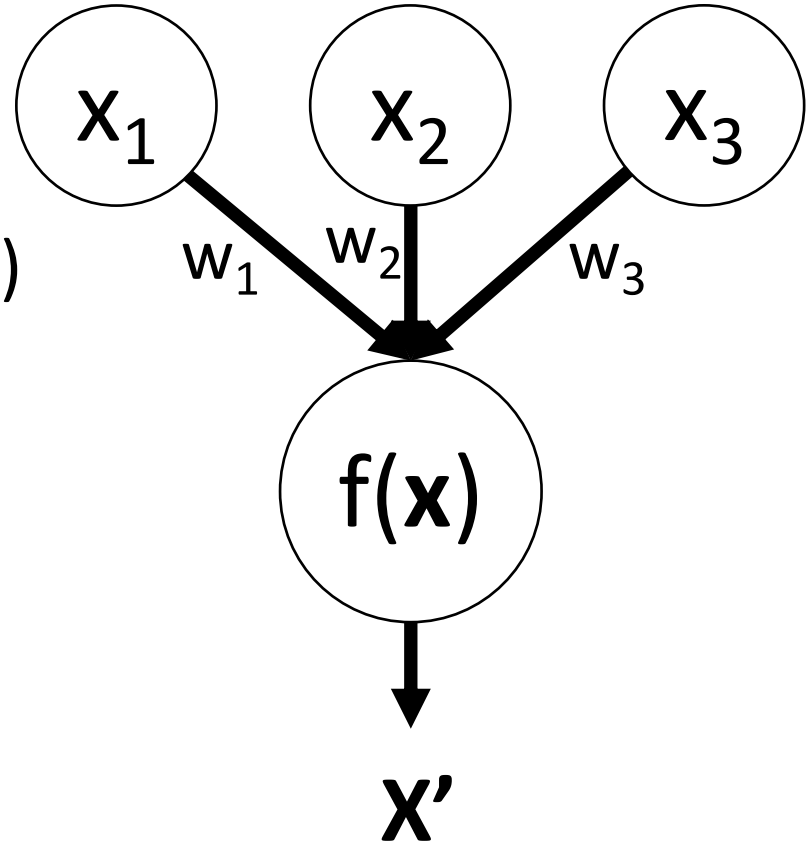
Sci-kit Activation Functions

Identity: $f(x) = x$

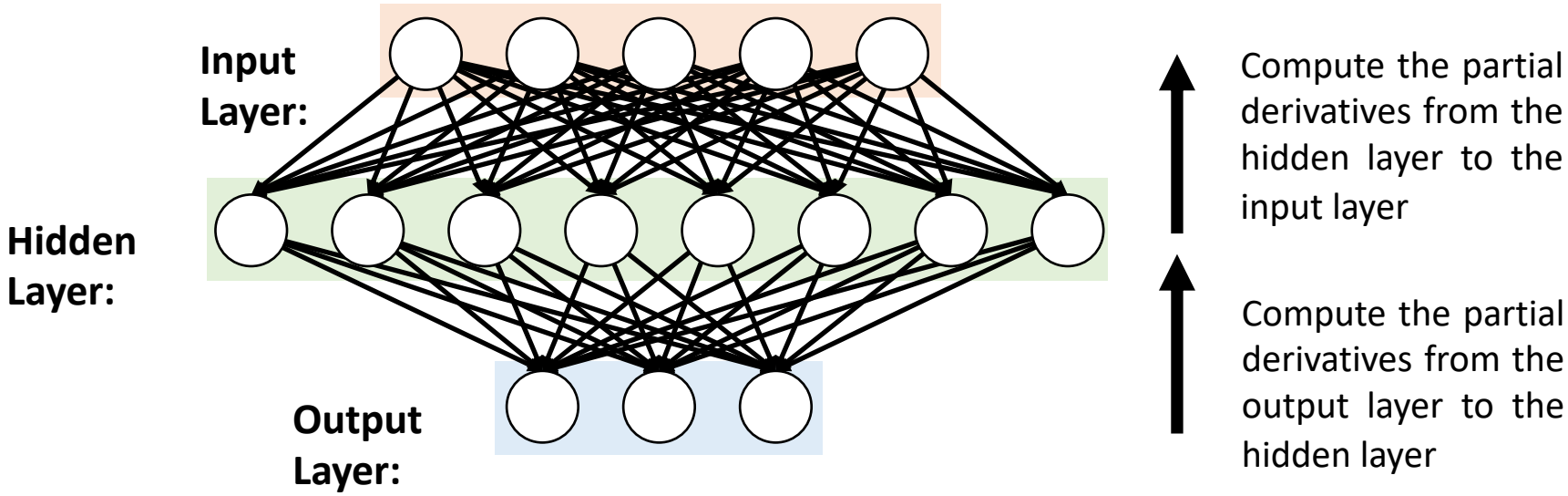
Logistic Sigmoid: $f(x) = 1/(1+e^{-x})$

Tanh: $f(x) = \tanh(x)$

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



Back Propagation



These partial derivatives handle the dependency of layer n+1 to layer n and are used by optimization algorithms for updating weights during training of the neural network

Weight Optimization Algorithms Used in Sci-kit

SGD – Stochastic Gradient Descent

L-BFGS – A quasi newton algorithm that uses the inverse Hessian matrix to update weights

ADAM – Adaptive Moment Estimator, which uses moments to estimate adaptive learning rates

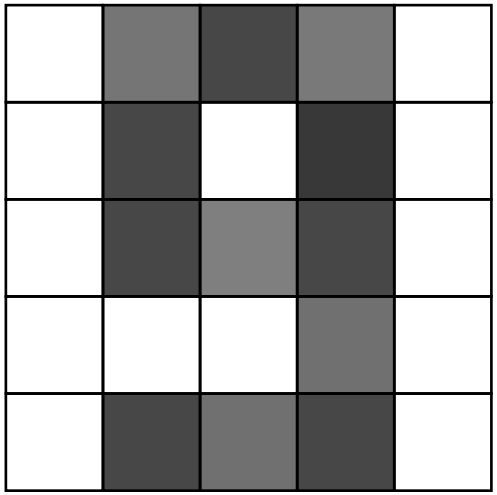
Learning rate control how much weights are updated in each step.

Training a neural network classifier with Sci-kit

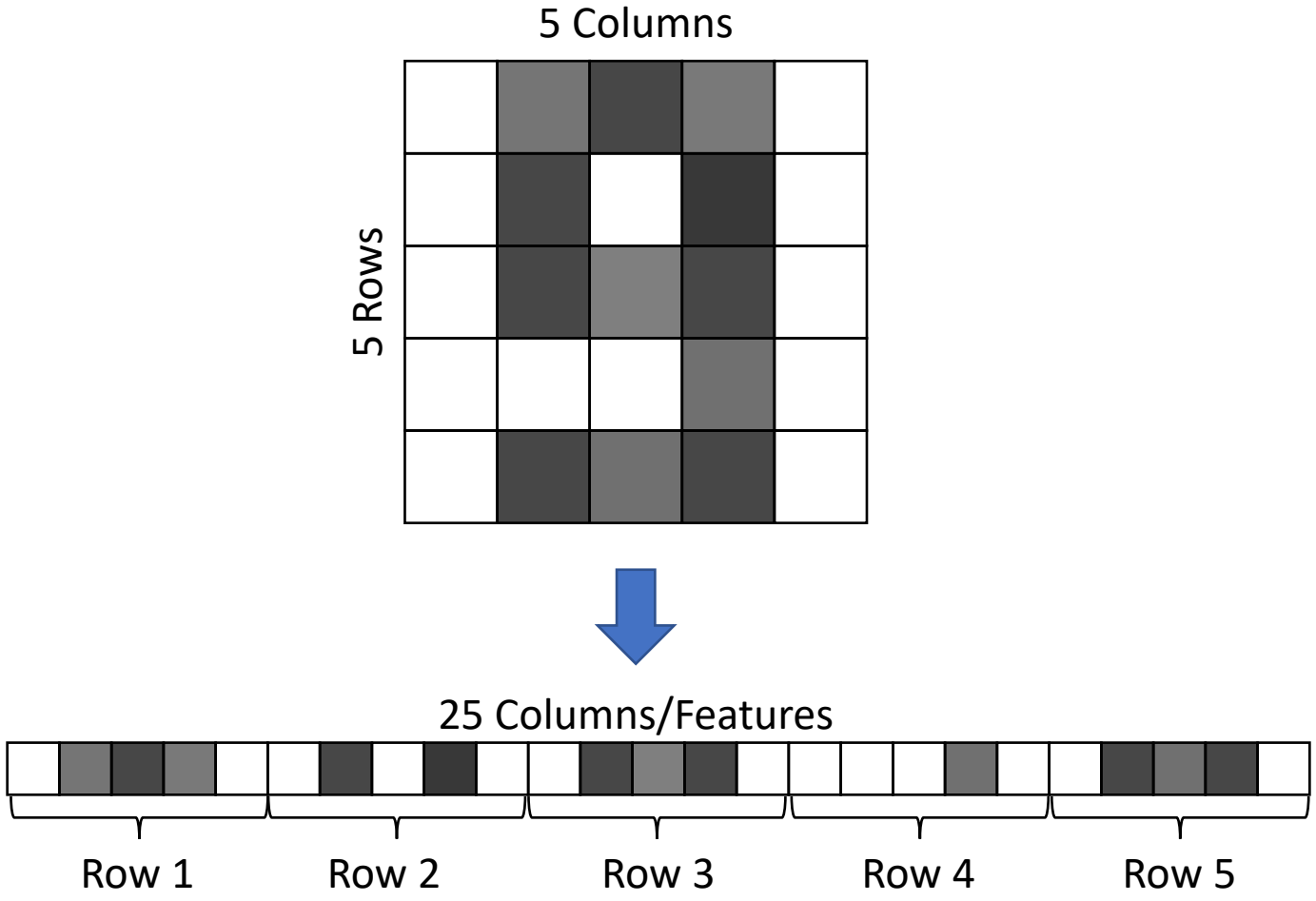
```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier()
mlp.fit(X_train, y_train)
```

Consider a 5x5 Grid For Representing Numbers



A Neural Network Encoding



This is known as "**Flattening**"

The MNIST Dataset

- The Modified Institute of Standards and Technology (MNIST) collected a total of 60000 training and 10000 testing images (128x128 pixels) of handwritten digits
- The current best known classifier achieves 99.83% accuracy as of January 2020.

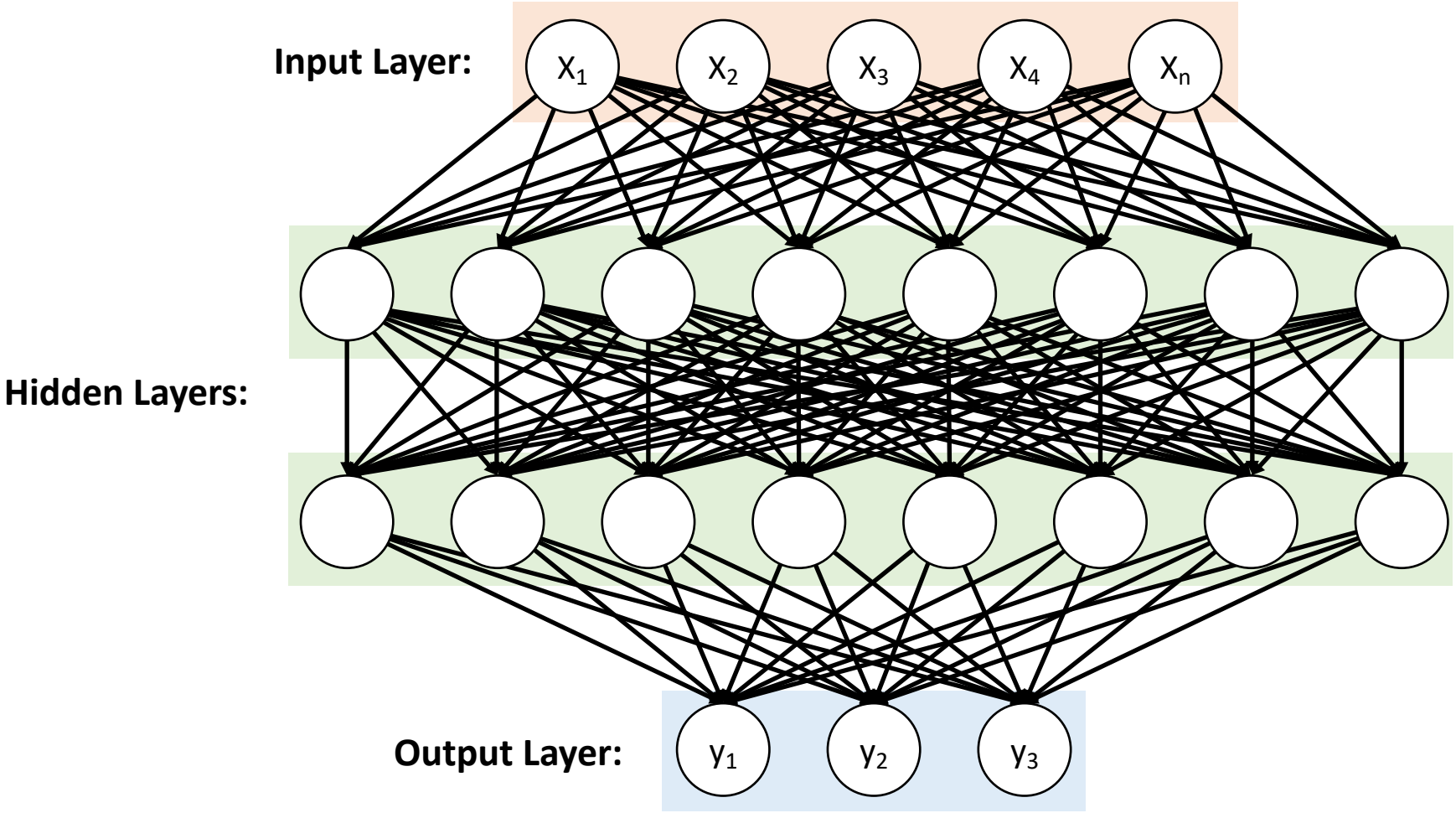
Exercise: Training a Neural Network Classifier with Sci-kit

- Load the flattened MNIST dataset (flattened_MNIST.txt)
- 0-1 Normalize the data (either manually or using scikit learn function: MinMaxScaler)
- Split the data into training and testing data
- Train a MLP classifier using the training data and test it using the testing data and evaluate the model accuracy.

Bonus Exercise:

- Try different number of hidden layer nodes

Deep Neural Network



Installing Keras & Tensorflow

Requirements: Python 3.5-3.7

Installing Tensorflow:

Using pip:

```
pip3 install tensorflow
```

Using conda:

```
conda install tensorflow
```

Installing Keras:

Using pip:

```
pip3 install keras
```

Using conda:

```
conda install keras
```