Neural Networks

Course:
INFO-6145 Data Science and Machine Learning

Revised by:
Mohammad Noorchenarboo

October 31, 2024

Contents

Current Section

Neural Networks: An Overview

Neural networks are computational models inspired by the human brain. They are composed of:

- **Nodes (Neurons)**: Basic processing units.
- **Edges (Connections)**: Connections between nodes that carry information.
- **Weights**: Parameters that control the strength of signals passed through edges.

Neural Networks: An Overview

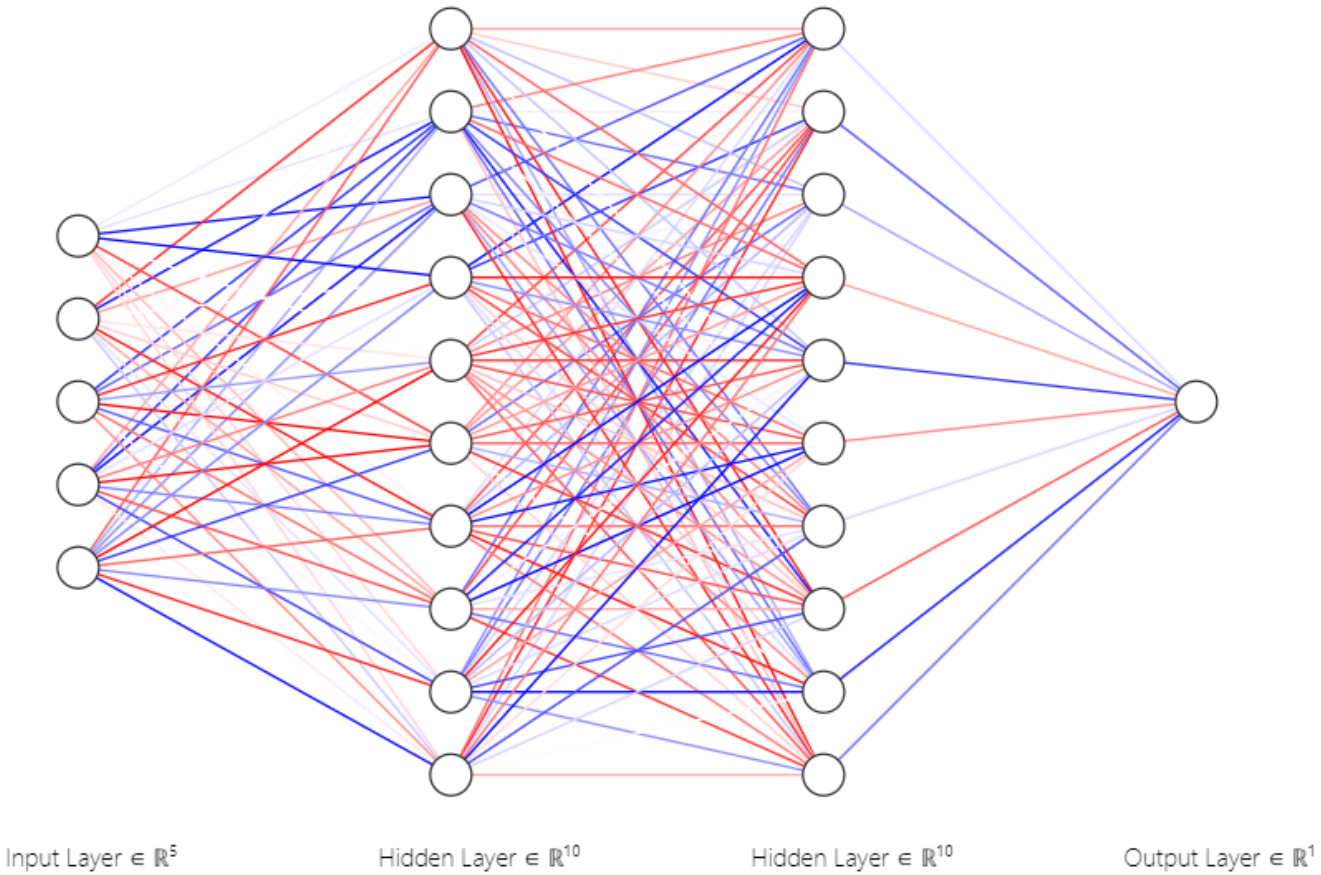

Figure 1: Neural Network (Ref:alexlenail)

Neural Networks: An Overview

Layers in a Neural Network

Neural networks are organized in layers:

- **Input Layer**: Receives the raw input data.
- **Hidden Layers**: Process inputs to extract patterns and features.
- **Output Layer**: Provides the final prediction or classification.

Example

In image classification, the input layer receives pixel values, hidden layers learn to identify shapes, and the output layer classifies the image.

Neural Networks in scikit-learn

scikit-learn provides neural network models suitable for simpler tasks:

- **MLPClassifier**: For classification tasks, predicting categories.
- **MLPRegressor**: For regression tasks, predicting continuous values.

Limitations of scikit-learn's Neural Networks

scikit-learn's neural network models are not optimized for deep learning or large-scale applications and lack GPU support.

scikit-learn vs. TensorFlow

- **scikit-learn**: Simple, user-friendly, suitable for basic machine learning tasks.
- **TensorFlow**: Designed for deep learning, optimized for complex neural networks, and offers GPU support for large-scale applications.

Use Cases

- Use scikit-learn for smaller, structured datasets where speed and simplicity are essential.
- Use TensorFlow for deep learning tasks with larger datasets, like image or text classification.

Building a Neural Network Classifier in scikit-learn

Trains an MLPClassifier on the Iris dataset

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load data and split
data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data,
data.target, test_size=0.3)
# Initialize, train, and test model
model = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
# Evaluate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")
```

Using MLPRegressor for Regression

Trains an MLPRegressor to predict continuous values

```
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error

# Generate and split data
X, y = make_regression(n_samples=100, n_features=1, noise=0.1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)

# Initialize, train, and test model
model = MLPRegressor(hidden_layer_sizes=(10,), max_iter=1000)
model.fit(X_train, y_train)
predictions = model.predict(X_test)

# Evaluate error
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse:.2f}")
```

Pros and Cons of Neural Networks

Advantages

- **Flexible**: Can handle nonlinear data patterns.
- **Fast Predictions**: Once trained, neural networks make predictions quickly.
- **Supports Complex Data**: Can process data with multiple features through various layers.

Pros and Cons of Neural Networks

Disadvantages

- **Black Box**: Hard to interpret or understand why it makes certain predictions.
- **Training Complexity**: Requires substantial computational resources and training time.
- **Overfitting Risk**: May memorize data instead of learning general patterns.
- **Data-Intensive**: Requires a large amount of training data for effective learning.

What is Deep Learning?

Deep learning refers to neural networks with multiple hidden layers, called **deep** networks, which extract complex patterns from data.

Deep Learning Applications

- **Image Recognition**: Convolutional Neural Networks (CNNs) are used for classifying objects in images.
- **Natural Language Processing**: Recurrent Neural Networks (RNNs) help understand patterns in text or speech.

Challenges of Deep Learning

- **Longer Training Time**: Deep networks can take hours or days to train.
- **Higher Accuracy**: When tuned correctly, deep networks often provide the highest accuracy.

Summary

Neural networks, especially deep learning models, add flexibility and power to machine learning.

- Suitable for classification and regression tasks.
- Powerful for complex data but requires careful tuning and sufficient data.
- TensorFlow and other deep learning frameworks enable handling larger, more complex models beyond scikit-learn's capacity.

Takeaway

Artificial neural networks expand the toolkit of data science, offering ways to model complex, nonlinear relationships that traditional models may miss.