Artificial Neural Network for Churn Prediction

Aniakwa Nathan

August 2, 2025

1 Introduction

This document presents an Artificial Neural Network (ANN) model developed to predict customer churn based on a dataset containing customer information. The model is implemented using TensorFlow and includes data preprocessing, ANN architecture, training, and evaluation steps.

2 Data Preprocessing

2.1 Importing the Libraries

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

The required libraries include NumPy for numerical operations, Pandas for data manipulation, and TensorFlow for building the ANN.

2.2 Importing the Dataset

```
dataset = pd.read_csv("Churn_Modelling2.csv")
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

The dataset is loaded from a CSV file, with features selected from columns 3 to the second-to-last column and the target variable from the last column.

2.3 Encoding Categorical Data

- **Label Encoding for Gender**:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

The Gender column is encoded into binary values (0 or 1).

- **One-Hot Encoding for Geography**:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])],
    remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

The Geography column is one-hot encoded to handle categorical values.

2.4 Splitting the Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=0)
```

The dataset is split into 80% training and 20% testing sets.

2.5 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Features are standardized to ensure consistent scales across variables.

3 Building the ANN

3.1 Initializing the ANN

```
ann = tf.keras.models.Sequential()
```

A sequential ANN model is initialized.

3.2 Adding Layers

- **Input and First Hidden Layer**:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

  A dense layer with 6 units and ReLU activation is added.

- **Second Hidden Layer**:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

  Another dense layer with 6 units and ReLU activation is added.

- **Output Layer**:

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

  The output layer has 1 unit with a sigmoid activation for binary classification.

4 Training the ANN

4.1 Compiling the ANN

```
ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

The model is compiled with the Adam optimizer, binary cross-entropy loss, and accuracy as the metric.

4.2 Training the ANN

```
ann.fit(X_train, y_train, batch_size=32, epochs=100)
```

The model is trained on the training set for 100 epochs with a batch size of 32.

5 Making Predictions and Evaluating the Model

5.1 Predicting a Single Observation

```
print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 50000]])
    > 0.5)
```

For a customer with features: France (1, 0, 0), Credit Score: 600, Gender: Male (1), Age: 40, Tenure: 3, Balance: $60,000, Number of Products: 2, Has Credit Card: Yes (1), Active Member: Yes (1), Estimated Salary: $50,000, the model predicts the customer will stay (False).

5.2 Predicting Test Set Results

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
```

Predictions are made on the test set, with a threshold of 0.5 for binary classification.

5.3 Confusion Matrix and Accuracy

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

The confusion matrix and accuracy are computed to evaluate the model's performance.

6 Analysis

The ANN model was trained on a dataset to predict customer churn. The preprocessing steps ensured proper handling of categorical variables and feature scaling. The model architecture with two hidden layers of 6 units each and a sigmoid output layer is suitable for binary classification. Training for 100 epochs achieved a stable accuracy. The single prediction example demonstrates practical application, predicting that the customer is likely to stay. The confusion matrix and accuracy score provide insights into the model's performance on the test set, indicating its effectiveness in classifying churn.