PyTorch Customer Churn NN Project

Solo or team?: Solo
Personal or School?: Personal
Approximate date: November 2024

Why did I do this?

Prior to this project, I had primarily been accustomed to building neural networks using TensorFlow. However, PyTorch offers more customization over the technical inner workings of neural networks, and is a widely used library in Deep Learning. I had been at the time taking courses relating to PyTorch, and I wanted to put some of the newfound knowledge to use, as well as come to my own understanding of how the library worked through trial and error. So, I was inspired to start this project.

Description

This project uses the Kaggle Dataset “Telco Customer Churn” to predict customer churn based on given data. The goal was to design a neural network using PyTorch that could accurately predict customer churn given the Telco dataset. The first half of the project consists of my original approach, taking the data as is and using a multi-layered NN to achieve a high (>90%) accuracy on churn predictions. The second half of the project is using an updated data preprocessing approach (handling class imbalances, upsampling, filling missing values, etc.) to achieve a high model accuracy.

Overview

Feel free to visit my portfolio website and check out the code I used to complete this project! In the code there are markdown comments explaining each section. However I will briefly go over some key elements of the project for the purposes of summarization.

- Loaded in the dataset, performed basic preprocessing techniques, scaled numeric values, and split the data into train, test, and validation sets.
- Converted the data into PyTorch tensors and defined the Neural Network architecture.
- Defined the loss and optimizer functions, trained the model, and evaluated the model
- The original model performed poorly and was overfitting to the training data. The baseline random-guessing accuracy was 73% (because of unbalanced classes), and my original model only had an accuracy of 76%. I needed a new approach
- I decided that the leading cause of low performance was likely the fact of unbalanced classes. To combat this problem, I used upsampling to even out the classes
- I redefined the model architecture, trained the new model, and evaluated the result
- The model’s loss was significantly lower, the problem of overfitting was considerably mitigated, and my final model achieved an accuracy of about 92%