Machine Learning from Scratch

By: Aditi Chaudhari

Building my Models and Analyzing my Results

For this project, I used C++ to build a logistic regression model and a Naïve Bayes model from scratch. The data used to train these models were from a CSV file named titanic_project.csv. The CSV file contained 4 columns: passenger number, passenger class (which could be 1, 2, or 3 with 1 being the highest priority), survived (which could be 0 or 1 with 0 meaning that the passenger did not survive the crash and 1 meaning that the passenger did survive the crash), sex (which could be 0 or 1), and age.

In the file named logistic_regression.cpp, the logistic regression model that I built predicted whether a passenger survived based on their sex. I used the first 800 observations to train my model and the rest of the observations to test my model and gauge accuracy, sensitivity, and specificity. Here is the output from a run:

```
Building a Logistic Regression Model for the data in the titanic_project.csv file...
Time taken to train the model with the training data: 2382 milliseconds.

Coefficient Outputs of the Logistic Regression Model:
w0 (intercept): 0.999877 , w1 (slope): -2.41086

Metrics:
accuracy: 0.784553
sensitivity: 0.695652
specificity: 0.862595

Process finished with exit code 0
```

The equation that we get from my linear regression model is:

y = -2.41908x + 0.999877

where y is the log odds of a passenger surviving the titanic crash and x is the sex of the passenger.

My model has an accuracy of around 78%, which is decent. The sensitivity, or the true positive rate, of the model is around 70% and the specificity, or the true negative rate, of the model is around 86%. These metrics indicate that this is a relatively good model.

In the file named naive_bayes.cpp, the Naïve Bayes model that I built predicted whether a passenger survived based on their passenger class, sex, and age. Once again, I used the first 800 observations to train my model and the rest of the observations to test my model. Here is the output from a run along with the first 10 predictions from the test data.

```
Building a Naive Bayes Model for the data in the titanic_project.csv file...

A-priori probabilities
   where survived = no: 0.61
   where survived = yes: 0.39

Likelihood values for p(pclass|survived)
   where pclass = 1 and survived = no: 0.172131
   where pclass = 1 and survived = yes: 0.416667
   where pclass = 2 and survived = no: 0.22541
   where pclass = 2 and survived = yes: 0.262821
   where pclass = 3 and survived = no: 0.602459
   where pclass = 3 and survived = yes: 0.320513

Likelihood values for p(sex|survived)
   where sex = 0 and survived = no: 0.159836
   where sex = 0 and survived = yes: 0.679487
   where sex = 1 and survived = no: 0.840164
   where sex = 1 and survived = yes: 0.320513

Likelihood values for p(age|survived)
   mean age for survived = no: 30.4182
   mean age for survived = yes: 28.8261
   variance for survived = no: 14.3231
   variance for survived = yes: 14.4622
```

```
Time taken to train the model with the training data: 16 milliseconds.


Evaluating model with test data...

            [0]                   [1]
[    0]   0.42088                0.57912
[    1]  0.793881               0.206119
[    2]  0.871139               0.128861
[    3]  0.226058               0.773942
[    4]  0.145902               0.854098
[    5]  0.165464               0.834536
[    6]  0.890094               0.109906
[    7]  0.867948               0.132052
[    8]  0.883218               0.116782
[    9]  0.788214               0.211786
```

The A-priori probabilities were 0.69 for did not survive and 0.39 for did survive.

The conditional probabilities were calculated for pclass, sex, and age and were used alongside the A-priori probabilities to build the Naïve Bayes model. The Naïve Bayes model was then evaluated with the test data. The first row of predictions (in which pclass=3, sex=0, and age=35) indicates that the passenger has a 42% chance of not surviving and a 58% change of surviving.

I used R in order to ensure that the A-priori probabilities, conditional probabilities, and predictions matched what my C++ Naïve Bayes model outputted.

Generative Classifiers vs. Discriminative Classifiers

Classification models, such as logistic regression and Naïve Bayes, can fall into two categories: generative classifiers and discriminative classifiers. Let's compare and contrast these categories. Both generative classifiers and discriminative classifiers are incredibly useful in the field of machine learning. Generative classifiers and discriminative classifiers are similar in that they both predict the conditional probabilities needed to map input data to a class label (https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3). How each classifier calculates the conditional probabilities is what makes them different.

Generative classifiers learn a model of the joint probability P(x,y) (where x is a feature and y is a target variable), make their predictions using Bayes rule in order to calculate P(y|x), and then choose a class label based of their prediction. They attempt to understand how features and target variables occur together. On the other hand, discriminative classifiers directly calculate conditional probabilities using

the conditional probability distribution. They attempt to define strict boundary lines that distinguish one class from another (https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf). Naïve Bayes is an example of a generative classifier, while logistic regression is an example of a discriminative classifier (https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3). Generative classifier are more data hungry and computationally expensive than discriminative classifiers. Discriminative classifiers, though, are robust in dealing with outliers, whereas outliers negatively impact the accuracy of a generative classifier model (https://towardsdatascience.com/generative-vs-discriminative-classifiers-in-machine-learning-9ee265be859e). It is important to consider the strengths and weaknesses of each classifier before choosing which classification model to use.

Reproducible Research in Machine

The National Science Foundation defines reproducibility as "the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator." In other words, another researcher should be able to use the same raw data, follow the procedure, and obtain the same results as the original researcher. This is what is known as reproducible research in the field of machine learning. Reproducible research is important to the field because it confirms that the results from the original researcher are correct, establishes transparency and reliability, and gives more researchers confidence in understanding what was done to achieve the results (https://blog.ml.cmu.edu/2020/08/31/5-reproducibility/). Without reproducible research, the field of machine learning would be saturated with inconclusive results, unintended consequences, and more. Machine learning is become more and more intertwined with our day-to-day activities and introducing faulty machine learning models from researchers to the public can be very dangerous. Therefore, reproducible research is incredibly important to the field of machine learning.

In order to implement reproducible research in machine learning, reproducibility needs to be considered from the very beginning of the research project. Documentation on the research project must start on day one and continue until the very end of the project. The documentation should address what decisions were made, why they were made, and what details are important in order to successfully execute the project(https://www.decisivedge.com/blog/the-importance-of-reproducibility-in-machine-learning-applications/#:~:text=Reproducibility%20with%20respect%20to%20machine,reporting%2C%20data%20analysis%20and%20interpretation) . Every aspect involved in the machine learning model, from the code to the data to the environment should be documented. Documenting code involves tracking and recording experimentation with the code and algorithms used. Documenting data involves tracking dataset versioning and changes and also discussing how the data was used. Finally, documenting the environment means tracking what framework dependencies, versions, hardware, and all other parts of the environment were used. By documenting every aspect of a research project, others can easily reproduce the project (https://neptune.ai/blog/how-to-solve-reproducibility-in-ml).