Classification Models for Sketch Drawings

Aparajita Dutta
Department of Electrical Engineering
Stanford University
apdutta@stanford.edu

Sarah Saki Robinson
Department of Computer Science
Stanford University
srobins2@stanford.edu

Abstract

In recent years, there has been significant development in the field of computer vision. Numerous papers have been published, analyzing various algorithms for image classification on photographs. In this paper, we apply different classification techniques to a dataset of sketch drawings to find what model performs best.

1 Introduction

A large percentage of publications on image classification perform experiments on photographs where the objects in images are easily identifiable by the human eye. In this paper, we explore a scenario where the images of objects are not as easily identifiable by humans. We compare different classification models applied to a dataset of doodles and analyze which model performs best on the drawing data. The models applied in this paper span from simple machine learning classifiers like Logistic Regression and Naive Bayes, to deep learning methods such as Convolutional Neural Networks and Multi-layer Perceptron.

2 Dataset and Features

For this project , we used the Google QuickDraw dataset [1]. The dataset contains a large number of simplified drawings, centered and formatted to a 28×28 grayscale bitmap. Although the data was available for over 300 different objects, only a small subset containing 22 objects was chosen for this project; namely: Apple, Banana, Bicycle, Birthday Cake, Butterfly, Candle, Computer, Door, Drums, Firetruck, Hat, Horse, Ice Cream, Leaf, Panda, Peanut, Pencil, Rainbow, Smiley Face, Snowman, Soccer Ball and Umbrella.

We restricted the data to N number of images per class, where N is a hyper-parameter that was tuned later in the experimentation. This was done to reduce computation time and induce class balance. The final, formatted data was then split in a 60:20:20 ratio into the training, development and test sets respectively.

3 Baseline and Oracle

To create the baseline expectation for this project, we loaded the data with $N = 20000$ images per class, and implemented a majority classifier. This achieved 4.447% accuracy with log loss = 33.003 on the development set.

For the oracle, we simulated a real-human classifier. We did this by printing out 30 individual images for each of the 22 classes and hiding the true labels. We then shuffled the images to randomize the order and attempted identify the correct class for each image. With this model, a 96.818% accuracy was achieved with log loss = 1.099. Figure 1a and 1b illustrate the confusion matrices.

4 Methods

4.1 Multi-Class Logistic Regression

Multi-Class Logistic Regression or Softmax Regression is a generalization of Logistic Regression where the sigmoid logistic function is replaced by the softmax function, ϕ, to allow multi-class classification.

$$P(y = i|x; \theta) = \phi_i = \frac{e^{\theta_i^T x}}{\sum_{j=i}^{k} e^{\theta_j^T x}}$$

where k is the number of classes. In this classification problem, $k = 22$ and the model parameters are $\theta_1, ..., \theta_{22}$.

Logistic regression is a quick and simple classifier. In this project we want to observe how well it performs on the drawing dataset in comparison to more complex and slower deep learning methods.

4.2 Naive Bayes

Naive Bayes is a supervised learning algorithm that uses Bayes' theorem to get the relationship:

$$P(y|x_1, ..., x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

The classifier is based on the strong assumption that the features, $x_1, ..., x_n$, are conditionally independent given the labels, y.

Although this algorithm is more commonly used in text classification, we are interested in observing how well is performs on images. There are different variations of Naive Bayes where different assumptions are made about the distribution of feature likelihoods. We will be implementing and comparing 3 different forms on Naive Bayes in our experimentation where the feature likelihoods are assumed to be distributed according to the Gaussian, Multinomial or Multivariate Bernoulli distributions.

5 Experiments

We started off the experimentation by choosing a value for the hyper-parameter N described in Section 2. This was done by testing different values of N and applying a simple logistic regression model to the data. The results from this experiment can be found in Table 1. We observed that for small Ns, the model seemed to be over-fitting on the training set, as the training log loss was significantly lesser than the development log loss. Even though, $N = 50000$ obtained the best results, the computation time was very high. Therefore, $N = 20000$ was chosen for further experiments as the results were comparable enough to $N = 50000$, while still maintaining a manageable computation time.

In the following subsections, we describe how we tuned the different models individually and report the results obtained in each case. The reported evaluation metrics are the the log loss, accuracy score and macro-averaged precision and recall obtained on the development set, the log loss obtained on the training set, and the confusion matrix obtained for the best set of hyper-parameters for each model.

N	Train Log Loss	Log Loss	Accuracy	Precision	Recall
1000	7.7738	12.1121	0.6493	0.65	0.65
5000	9.4971	10.6771	0.6908	0.69	0.69
10000	9.7462	10.5107	0.6957	0.69	0.70
20000	9.9290	10.2584	0.7030	0.70	0.70
50000	9.9360	10.0669	0.7085	0.71	0.71

Table 1: Tuning the number of images per class (N)

5.1 Multi-Class Logistic Regression

Logistic Regression was implemented using stochastic gradient descent with L2-regularization. To create the model in Python, we used the Scikit-learn library [2]. The inverse of the regularization strength, C, was taken as a hyper-parameter that was tuned using grid search. Table 2 shows the results for different values of C.

As seen from the results, setting $C = 0.1$ achieved the best accuracy and log loss. Figure 1c shows the confusion matrix obtained with $C = 0.1$.

C	Train Log Loss	Log Loss	Accuracy	Precision	Recall
0.01	10.1267	10.2953	0.7019	0.70	0.70
0.1	9.9290	10.2584	0.7030	0.70	0.70
0.5	9.8735	10.2733	0.7026	0.70	0.70
1	9.8654	10.2729	0.7026	0.70	0.70

Table 2: Tuning regularization strength inverse (C) for Logistic Regression

5.2 Naive Bayes

As mentioned in Section 4.2, three variations of Naive Bayes were implemented for the experimentation and the results were compared to determine which distribution assumption for the feature likelihood works best for our drawing data. The Scikit-learn library [2] was used to create the Naive Bayes models. Table 3 shows the results obtained in each case.

Bernoulli Naive Bayes achieved the best results on the development set. Figure 1d shows the confusion matrix obtained with Bernoulli Naive Bayes.

Model	Train Log Loss	Log Loss	Accuracy	Precision	Recall
Gaussian	18.8616	18.8476	0.4543	0.51	0.45
Multinomial	16.8599	16.8369	0.5125	0.52	0.51
Bernoulli	15.6298	15.5872	0.5487	0.56	0.55

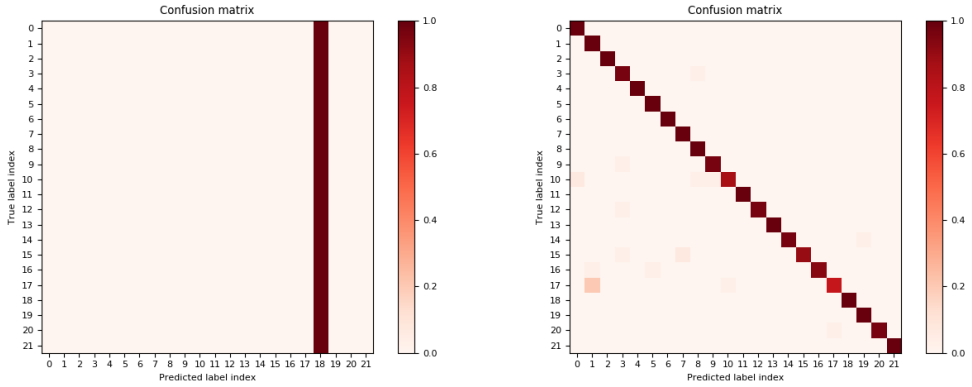
Table 3: Results using different Naive Bayes models

6 Discussion

The performance of the Naive Bayes models was poorer than Logistic Regression. Being predominantly a text classifier, it was unable to understand the subtleties of image data. As seen from the confusion matrix, Figure 1d, there were certain classes that were misclassified by the model very frequently. It was interesting to observe the types of mistakes made by this model, for example, it predicted a large number of pencil images as candles and a large number of peanut images as snowman. These pairs have similar traits when drawn that could be confusing even to the human eye, so it is very interesting to see the model face similar challenges.

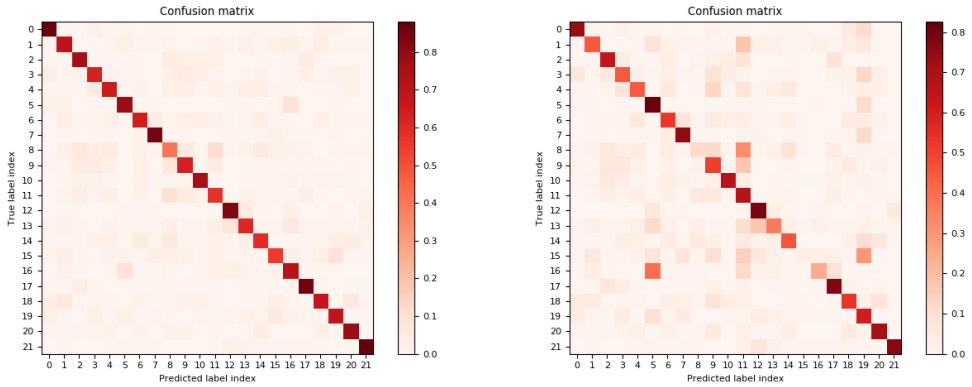
Logistic regression performed surprisingly well given the simplicity of the model. It is clear from the results in Table 2 and the confusion matrix, Figure 1c, that this model was able to accurately predict most of the classes. Additionally, there was no single class on which the model performed significantly worse than the others, unlike what was seen with Naive Bayes.

Although both these models performed significantly better than our baseline, there is still a lot of scope for improvement as they are still far from the oracle performance. Going forward, we are interested in seeing how deep learning methods overcome the challenges faced by these single-dimensional models. Theoretically, use of neural networks should allow the model to learn more about the training data with the use of higher dimensional space and hidden layers. Hence, we expect the neural network models to outperform both Naive Bayes and Logistic Regression.



(a) Baseline

(b) Oracle



(c) Logistic Regression with C = 0.1

(d) Bernoulli Naive Bayes

Figure 1: Confusion matrices

7 Future Work

As mentioned in Section 6, our plan is to next implement and tune a Multi-Layer Perceptron model as well as a Convolutional Neural Network model. Once we complete tuning both of these on the development set, we will compare the 4 tuned models on the test set to determine which model works best on our dataset.

Finally, we will demonstrate this project by building an application that lets users create a drawing of one of the 22 classes chosen for this project. The app will convert the image to the right format and apply the best classification model, chosen above, to predict what has been drawn.

References

[1] Jongejan, J., Rowley, H., Kawashima, T., Kim, J. & Fox-Gieg, N. (2016) The Quick, Draw! - A.I. Experiment. `https://quickdraw.withgoogle.com/data`.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.