CS230

PetFinder.my Adoption Speed Prediction Using Pet Profiles

Iris Wang
iris1121@stanford.edu

Eva Gong
gong717@stanford.edu

Abstract

The goal of this project is to build a neural network to predict how fast a pet is adopted so that shelters and adoption agencies can better focus their resources to help the pet to find new homes. Using PetFinder.my's pet profile data (provided as part of a Kaggle competition [1]), we extracted about 400+ features from numeric, categorical and text input data. After employing Adam Optimization, Regularization and random search of hyperparameters (e.g. learning rate, minibatch size, number of neurons, and regularization coefficient), we trained a 2-layer neural network to predict pets' adoption speed with a 38.6% Validation Accuracy.

1 Introduction

Artificial Neural Networks have become the standard in achieving state-of-the-art results in supervised learning tasks where the inputs are of various and complex formats. They have been shown to perform well on image classification, text analysis and other tasks involving large input datasets.

In our case, we are interested in applying neural networks to predict the adoption speed of shelter animals. Using PetFinder.my's pet profile data (provided as part of a Kaggle competition [1]), we aim to build neural networks to predict how fast a shelter pet will be adopted. This problem is of great interest for two reasons. First, the input data contains both structured numerical features (e.g. type, breed, size etc.), and unstructured ones (e.g. pet images and text descriptions). The rich amount of data and its different formats pose technical challenges that we are interested in tackling. Furthermore, predicting each pet's adoption speed can help animal shelters allocate resources and employ strategies to speed up pet adoptions early on. This makes the problem very meaningful.

2 Dataset and Features

2.1 Data Type

The input data is comprised of three types: structured data, unstructured data, and unique IDs (see Table 1 for a list of data fields and descriptions provided by Kaggle.com [1]). During initial data pre-processing, we removed all unique ID columns as they do not add useful signals to the model.

Structured data covered basic information of the pet information like age, type, breed etc. We converted categorical features into integer matrices using one hot encoding, resulting in 400 input features and 5 adoption speed classes as the output. Those 5 adoption speed classes indicate how quickly, if at all, a pet is adopted (0 if adopted on the same day, 1 if adopted between 1 to 7 days, 2 if between 8 to 30 days, 3 if between 30 to 90 days, and 4 if more than 90 days). Because the input data has already taken care of most of the missing values (e.g. a value of $0 = NotSpecified$ is given to pets missing $FurLength$ values), we did not take additional steps to impute missing values for structured data. We acknowledged the large input feature space, and tried following two methods for feature selection: 1) Keeping only the top 20 categories and aggregate the rest in 1 single category 2) Principle Component Analysis As will be shown later in the Results section, our models have low training accuracy after employing regularization. We therefore decided to include all features for training.

Some pets are provided with text description (in the $Descriptions$ field). These are short sentences written by the agency/foster homes. Majority of the descriptions are in English with a few exceptions. Kaggle provides analysis on sentiment and key entities for the description using Google's Natural Language API. From the text descriptions, we extracted the following features to be fed into the softmax classfier. 1) Document sentiment score: This is provided by the API. Score of the sentiment ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text. For pets missing this information, we imputed with 0. 2) Document sentiment magnitude: This is provided by the API. Magnitude indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and ∞. For pets missing this information, we imputed with 0. 3) Letter count: This is the additional feature created to represent how long the description is. This is created by counting the total number of letters in the description.

Table 1: Input Data Description

Field Name	Format	Category	Feature Type
Type	int	Structured Data	Categorical
Breed1	int	Structured Data	Categorical
Breed2	int	Structured Data	Categorical
Gender	int	Structured Data	Categorical
Color1	int	Structured Data	Categorical
Color2	int	Structured Data	Categorical
Color3	int	Structured Data	Categorical
MaturitySize	int	Structured Data	Categorical
FurLength	int	Structured Data	Categorical
Vaccinated	int	Structured Data	Categorical
Dewormed	int	Structured Data	Categorical
Sterilized	int	Structured Data	Categorical
Health	int	Structured Data	Categorical
State	int	Structured Data	Categorical
AdoptionSpeed	int	Structured Data	Categorical
Age	int	Structured Data	Continuous
Quantity	int	Structured Data	Continuous
Fee	int	Structured Data	Continuous
VideoAmt	int	Structured Data	Continuous
PhotoAmt	float	Structured Data	Continuous
Name	object	Unique ID	NA
RescuerID	object	Unique ID	NA
PetID	object	Unique ID	NA
Description	object	Unstructured Data	Feature Engineering

2.2 Train, Validation and Test Sets

The dataset provided by Kaggle has $14,993$ samples for training, and $3,948$ samples for testing. Following field best practices, we randomly sampled 20% of the training set as the hold-out validation set, leaving us $11,994$ samples for training, and $2,299$ samples for validation. We have also validated that the class distributions are consistent across training and validation sets (see Figure 1).

3 Methods

3.1 Baseline Model

As mentioned previously in Section II, our baseline model focuses on structured data types, and $Descriptions$ field is removed from training. After splitting raw training data into training and

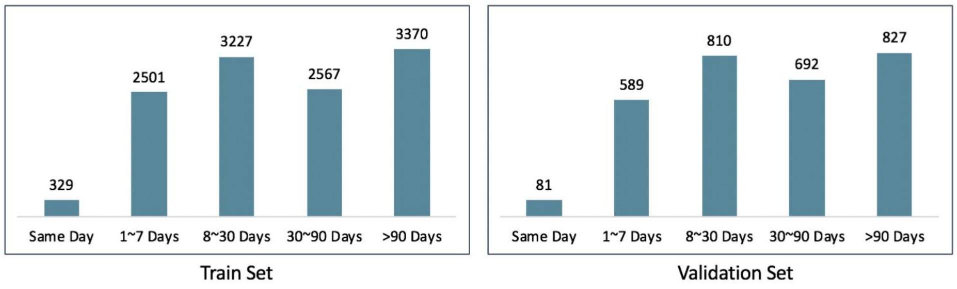

Figure 1: Class Distributions for Train and Validation Sets

validation set, we ended up having an input layer of dimension $11,994 \times 400$. Due to the nature of this problem, we employed cross-entropy as the cost function, ReLU as activation functions for the hidden layers, and softmax classifier for the output layer. To optimize the training speed, we employed the mini-batch approach (a batch size of 32 and a total epoch of 1500) and Adam optimization method.

To lower the bias, the main hyperparameters that we would like to optimize are the number of hidden layers and the number of neurons in each hidden layer. We experimented with two fully connected neural networks for the baseline model. The first one has one hidden layer with 200 neurons, and the second one has two hidden layers with 200 and 100 neurons for the first and second hidden layer, respectively. Models' performancs will be shown later in the Results Section.

3.2 Regularization

To prevent models from over-fitting, we used L2 regularization to try to lower the variance.

3.3 Hyperparameters tuning

After adding the regularization terms, we noticed that the difference between training accuracy and validation accuracy significantly decreased. To improve the performance of the model, we used random search to tune the hyperparameters. We used random search because the dimensions of parameters are high and it is infeasible to do a grid search throughout the entire space. The hyperparameters are:

3.3.1 Learning rate

We picked learning rates between 0.000001 to 0.1 on a log scale.

3.3.2 Minibatch size

Minibatch size options are selected between 32 to 512 by increase of power of 2.

3.3.3 Number of neurons

Following some general best practices, we picked the number of neurons for the first layer to be within the range between half of the number of input features to the total number of input feature (e.g. if there are 400 input feature, we will test neuron size between 200 to 400) [2]. The number of neurons for the second layer is set to be half of the first layer's.

3.3.4 Regularization coefficient

We tested regularization coefficients ranging between 0.001 and 0.01.

After setting the different hyperparameters, we randomly searched through the space 15 times to pick the ones with the best validation set accuracy.

4 Results

4.1 Prediction Accuracy

For this softmax classifier, our main performance metric is the overall prediction accuracy rate, defined as the total number of correctly predicted samples over the total number of samples. We chose overall accuracy measurement over per-class evaluation metrics because as shown before in Section II, the distribution of the output classes is nearly uniform (except for the *Same Day* class). Plus, optimizing the number of correctly predicted samples helps with the Kaggle competition performance. Table 2 below shows the training and validation accuracies for different models and different Neural Network infrastructures. As shown, after applying L2 regularization, we obtained an improved validation accuracy, and lowered the difference between training and validation accuracies. Although the 3-layer network has better performance over other networks, unfortunately its long computing time restricted our abilities to tune hyperparameters, and we thus decided to use a 2-layer network for hyperparameter tuning.

Table 2: Training and Validation Accuracies

Model	# Layers	L2 Regularization	Training Accuracy	Validation Accuracy
Baseline	1	No	77.1%	34.5%
Baseline	2	No	97.4%	34.2%
Baseline with Text Features	2	Yes	36.1%	35.0%
Baseline with Text Features	3	Yes	40.9%	38.3%

Table 3 shows the training and validation accuracies for top 10 parameter combinations with the best performance. As shown, a combination of 0.001 learning rate, mini-batch size of 64, 221 neurons for the first layer and 0.004 regularization rate gives the best validation accuracy of 38.6%. The training accuracy for that combination is 40.9%, and this small difference between training and validation accuracies indicates that the model is not suffering from over-fitting.

Table 3: Training and Validation Accuracy by Hyperparameters Combination

Learning Rate	Minibatch Size	# Neurons	Regularization β	Training Accuracy	Validation Accuracy
0.00001	128	331	0.01	41.0%	38.4%
0.00001	128	371	0.004	42.4%	38.2%
0.000001	64	301	0.007	40.6%	37.4%
0.0001	128	271	0.001	57.7%	37.8%
0.00001	64	281	0.004	41.2%	38.0%
0.00001	64	221	0.007	40.9%	38.3%
0.001	64	321	0.01	40.0%	37.8%
0.001	64	221	0.004	41.2%	38.6%
0.00001	64	271	0.01	40.9%	37.9%
0.00001	64	271	0.01	40.2%	36.7%

4.2 Error Analysis

For the error analysis, we calculated per-class performance metrics to understand how the model performs across different classes. As shown in the confusion matrix plot in Figure 2, *More Than 90 Days* class has the most instances that are correctly predicted by the model.

Following common multi-class performance measures outlined by Sokolova et al., we then calculated per-class and average precision and recall rates through macro-averaging [3]. We chose macro-averaging over micro-averaging because all classes are treated equally in this analysis. As shown in Table 4, *Same Day* class has the lowest precision and recall rates, whereas *More Than 90 Days* class has the highest precision and recall rates. As the whole dataset has quite a small percentage of *Same Day* samples to start with, we are less concerned about its low precision and recall Rates. On the other hand, we found the variability in precision and recall rates across other classes to be worth investigating. We listed out some potential ways to deal with this in the next section.

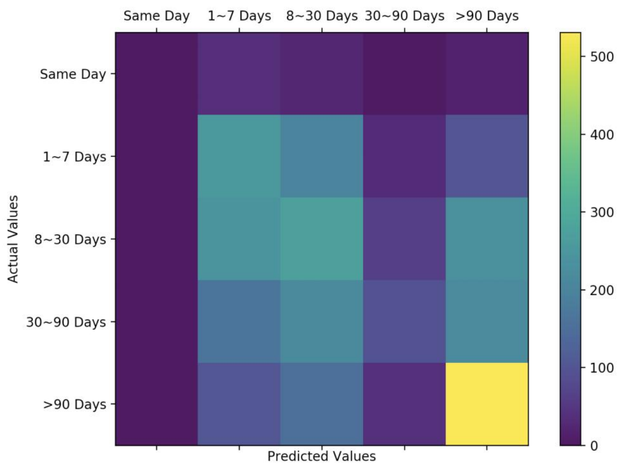

Figure 2: Confusion Matrix

Table 4: Additional Performance Metrics on the Validation Set

Adoption Speed	Precision	Recall
Same Day	0%	0%
1 to 7 Days	32%	44%
8 to 30 Days	31%	34%
30 to 90 Days	42%	14%
More than 90 Days	48%	64%
Average	31%	31%

5 Discussions and Future Work

To summarize, for this project we aimed to build a neural network to predict pets' adoption speed class with a mix of numeric, categorical features and additional features obtained through text data's feature engineering. After employing Adam Optimization, Regularization and random search of hyperparameters (e.g. learning rate, minibatch size, number of neurons, and regularization coefficient), we trained a 2-layer neural network to predict pets' adoption speed with a 38.6% validation accuracy. In terms of per-class performance measurements, the precision and recall rates vary across classes, with the *Same Day* class having the lowest precision and recall rates, and *More Than 90 Days* class having the highest precision and recall rates.

One potential area for future work is to build and train hyperparameters on deeper networks. Due to the large amount of input data and features, we are limited by the computing time in terms of how many layers and different hyperparameters we can train with. We believe trying out deeper networks can help with improving the training accuracy.

Additionally, to lower the differences among per-class performance measurements, it will be interesting to see if the results will be more balanced based off Average Prediction Accuracy rather than Overall Prediction Accuracy. Alternatively, we can bias the softmax classifier by adjusting the weights of each class when calculating the prediction probabilities.

Lastly, due to the computing time constraint, we did not have time to look into image datasets. For future work, it will be interesting to see if we can train a CNN on the image dataset, with the adoption speed as the out variable.

6 Contributions

Both members contributed equally to this project. Here is the link to our github repository: https://github.com/yifan-eva-gong/petfinder.my-adoption-speed-prediction.

We would like to thank our Project Mentor Gael Colas for his generous and great help throughout our project research. We would also like to thank Professor Andrew Ng and Professor Kian Katanforoosh for their teaching and guidance.

References

[1] PetFinder.my Adoption Prediction: How cute is that doggy in the shelter? Kaggle Competition. https://www.kaggle.com/c/petfinder-adoption-prediction/

[2] Heaton, J. (2009). Introduction to neural networks with Java. Chesterfield (MO, USA): Heaton Research.

[3] Sokolova, M., Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing Management, 45(4), 427-437.