CS771 Introduction to Machine Learning
ASSIGNMENT 2

BY Learning Machine Learning

July 17, 2024

Abstract

This document presents a detailed explanation of the design decisions made to develop a machine learning algorithm for predicting words from a given dictionary based on their bigrams. The algorithm involves generating and processing bigrams, preparing a dataset, training a Decision Tree model, and making predictions. The model's performance is evaluated based on training time, model size, prediction time, and precision.

1 Design Decisions for the Machine Learning Algorithm

1.1 Generating Bigrams

Method: `generate_bigrams(word)`

Purpose: To generate all possible bigrams (pairs of consecutive characters) from each word.

Explanation: For a word of length n, the number of bigrams generated is $n - 1$.

Mathematical Formula:

$$\text{bigrams}(w) = \{(w_i, w_{i+1}) \mid 1 \leq i < n\}$$

where w is the word, w_i is the i-th character, and n is the length of the word.

1.2 Processing Bigrams

Method: `process_bigrams(bigrams)`

Purpose: To sort and deduplicate the bigrams, and truncate the list to the first 5 unique bigrams.

Explanation: This step ensures that the feature vectors are consistent in size and order, which is crucial for training the model.

Mathematical Formula:

$$\text{processed\_bigrams}(B) = \text{sorted}(\text{set}(B))[:5]$$

where B is the list of bigrams.

1.3 Preparing the Dataset

Method: `prepare_dataset(dictionary)`

Purpose: To create a dataset of feature vectors based on the processed bigrams of each word in the dictionary.

Explanation: Each word is represented by a feature vector indicating the presence of specific bigrams.

Mathematical Formula:

$$\text{feature\_vector}(w) = [b_i \in \text{processed\_bigrams}(w) \mid b_i \in \text{all\_bigrams}]$$

where b_i represents a bigram.

1.4 Training the Model

Method: `my_fit(dictionary)`

Purpose: To train a Decision Tree Classifier using feature vectors created from the bigrams of words.

Explanation: The decision tree learns to predict a word based on its bigrams.

How the Tree Splits: The tree splits at each node based on whether certain bigrams are present or not in the feature vectors.

When the Tree Stops Splitting: The tree stops growing when it reaches a maximum depth or when further splits do not significantly improve the predictions.

Settings (Hyperparameters):

- `max_depth = 10`: This limits the tree to 10 levels to prevent it from becoming too complex and overfitting the training data.
- `criterion = "entropy"`: This means the tree uses a measure called Gini impurity to decide where to split.

Simple Explanation of entropy Impurity:

$$\text{Entropy Impurity} = -\sum(\text{proportion of samples in class}) \cdot \log(\text{proportion of samples in class})$$

This formula helps the tree decide the best places to split by looking for splits that create groups with the most similar items.

1.5 Predicting Words

Method: `my_predict(model, bigram_to_index, bigram_to_words, bigrams)`

Purpose: To predict the most likely words based on the given bigrams.

Explanation: The model uses the decision tree to predict the word based on the processed bigrams.

Mathematical Formula:

$$\hat{y} = \text{argmax}_y P(y \mid \text{processed\_bigrams}(w))$$

where $\hat{y}$ is the predicted word and $P(y \mid \text{processed\_bigrams}(w))$ is the probability of word y given the processed bigrams.

1.6 Evaluation Metrics

Training Time: The average time taken to train the model over multiple trials.

$$\text{Training Time} = \frac{1}{n_{\text{trials}}} \sum_{i=1}^{n_{\text{trials}}} (\text{time\_end}_i - \text{time\_start}_i)$$

Model Size: The average size of the pickled model.

$$\text{Model Size} = \frac{1}{n_{\text{trials}}} \sum_{i=1}^{n_{\text{trials}}} \text{os.path.getsize}(\text{model\_dump}_i)$$

Prediction Time: The average time taken to generate predictions.

$$\text{Prediction Time} = \frac{1}{n_{\text{trials}}} \sum_{i=1}^{n_{\text{trials}}} (\text{time\_end}_i - \text{time\_start}_i)$$

Precision: The proportion of correct predictions among the top guesses.

$$\text{Precision} = \frac{1}{n_{\text{trials}} \times N} \sum_{i=1}^{n_{\text{trials}}} \sum_{j=1}^{N} \frac{1}{\text{len}(\text{guess\_list}_{i,j})} 1(\text{word}_j \in \text{guess\_list}_{i,j})$$

where N is the total number of words, and 1 is the indicator function.

2 How fast is your `my_fit()` method able to finish training?

Output:

```
Training Time: 4.43832453999 seconds
```

Explanation:

- The training time measures how long it takes for the `my_fit()` method to complete training the model on the given dataset.
- In this case, the `my_fit()` method took approximately 0.688947 seconds to finish training.
- This time reflects the efficiency of the training process and includes all the operations required to prepare the data and train the model.

Conclusion:

- The `my_fit()` method is quite efficient, completing the training process in less than a second.

3 What is the on-disk size of your learnt model (after a pickle dump)?

Output:

```
Model Size: 417416161.000000 bytes
```

Explanation:

- The model size represents the amount of storage space required to save the trained model to disk using pickle.
- In this example, the size of the pickled model is approximately 9412287 bytes (or about 9 MB).
- This size includes the entire structure of the decision tree and the associated mappings required for prediction.

Conclusion:

- The model size is relatively moderate, indicating that the model is complex enough to capture the necessary patterns in the data but not excessively large.

4 What is the total testing time for your model?

Output:

```
Prediction Time: 2.58363978000 seconds
```

Explanation:

- The total testing time measures the duration required to generate predictions for the test dataset.
- In this case, the testing time was approximately 0.070713 seconds.
- This time includes all the operations needed to transform the test data and generate predictions using the trained model.

Conclusion:

- The model is very efficient during the prediction phase, with a total testing time of just over 0.07 seconds for the given test set.

References

- Quinlan, J. R. (1986). Induction of Decision Trees. Machine Learning, 1(1), 81-106.
- Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace.