Data Transformation - Learning with Counts

12/18/2017 • 4 minutes to read • Contributors

In this article

Learning with counts is an efficient way to create a compact set of features for a dataset, based on counts of the values. You can use the modules in this section to build a set of counts and features, and later update the counts and the features to take advantage of new data, or merge two sets of count data.

The basic idea underlying *count-based featurization* is simple: by calculating counts, you can quickly and easily get a summary of what columns contain the most important information. The module counts the number of times a value appears, and then provides that information as a feature for input to a model.

Example of Count-Based Learning

Imagine you're trying to validate a credit card transaction. One crucial piece of information is where this transaction came from, and one of the most common encodings of that location is the postal code. However, there might be as many as 40,000 postal codes, zip codes, and geographical codes to account for. Does your model have the capacity to learn 40,000 more parameters? If you give it that capacity, do you now have enough training data to prevent it from overfitting?

If you had really good data with lots of samples, such fine-grained local granularity could be quite powerful. However, if you have only one sample of a fraudulent transaction from a small locality, does it mean that all of the transactions from that place are bad, or that you don’t have enough data?

One solution to this conundrum is to learn with counts. That is, rather than introduce 40,000 more features, you can observe the counts and proportions of fraud for each postal code. By using these counts as features, you gain a notion of the strength of the evidence for each value. Moreover, by encoding the relevant statistics of the counts, the learner can use the statistics to decide when to back off and use other features.

Count-based learning is very attractive for many reasons: you have fewer features, requiring fewer parameters, which makes for faster learning, faster prediction, smaller predictors, and less potential to overfit.

How Counts are Created

An example might help to demonstrate how count-based features are created and applied. This example is highly simplified, to give you an idea of the overall process. You can also see how to use and interpret count-based features.

Suppose you have a table like this, with labels and inputs:

Label column	Input value
0	A
0	A

Label column	Input value
1	A
0	B
1	B
1	B
1	B

Here is how count-based features are created:

1. Each case (or row, or sample) has a set of values in columns. In this example, the values are A, B, and so forth.
2. For a particular set of values, you find all the other cases in that dataset that have the same value. In this case, there are three instances of A and four of B.
3. Next, you count their class memberships as features in themselves. In this case, you get a small matrix, in which there are 2 cases where A=0, 1 case where A = 1, 1 case where B= 0, and 3 cases where B = 1.
4. Based on this matrix, you get a variety of count-based features, including a calculation of the log-odds ratio as well as the counts for each target class:

Sample of count-based features

Label	0_0_Class000_Count	0_0_Class001_Count	0_0_Class000_LogOdds	0_0_IsBackoff

Label	0_0_Class000_Count	0_0_Class001_Count	0_0_Class000_LogOdds	0_0_IsBackoff
0	2	1	0.510826	0
0	2	1	0.510826	0
1	2	1	0.510826	0
0	1	3	-0.8473	0
1	1	3	-0.8473	0
1	1	3	-0.8473	0
1	1	3	-0.8473	0

Examples

The following article from the Microsoft Machine Learning team provides a detailed walkthrough of how to use counts in machine learning, and compares the efficacy of count-based modeling with other methods.

Using Azure ML to Build Clickthrough Prediction Models

Technical Notes

This section contains implementation ntoes and answers to some frequently asked questions.

How is the log-loss value calculated?

The Log-loss value is not the plain log-odds; the prior distribution is used to smooth the log-odds computation.

Suppose you have a data set used for binary classification. In this dataset, the prior frequency for class 0 is p_0, and the prior frequency for class 1 is p_1 = 1 – p_0. For a certain training example feature, the count for class 0 is x_0, and the count for class 1 is x_1.

Under these assumptions, the log-odds is computed as `LogOdds = Log(x_0 + c * p_0) – Log (x_1 + c \* p_1)` where `c` is the prior coefficient, which can be set by the user, and the log function uses the natural base.

In other words, for each class `i`:

```
Log_odds[i] = Log( (count[i] + prior_coefficient * prior_frequency[i]) / (sum_of_counts -
count[i]) + prior_coefficient \* (1 - prior_frequency[i]))
```

If the prior coefficient is positive, the log odds can be different from `Log(count[i] / (sum_of_counts – count[i]))`.

Why are the log odds not computed for some items?

By default, all items with a count less than 10 are collected in a single bucket called the "garbage bin". You can change this behavior value by using the **Garbage bin threshold** option in the Modify Count Table Parameters module.

List of Modules

The **Learning with Counts** category includes the following modules:

Module	Description
Build Counting Transform	Creates a count table and count-based features from a dataset, and saves it as a transformation
Export Count Table	Exports count table from a counting transform This module supports backward compatibility with experiments that create count-based features using Build Count Table (deprecated) and Count Featurizer (deprecated).
Import Count Table	Imports an existing count table This module supports backward compatibility with experiments that create count-based features using Build Count Table (deprecated) and Count Featurizer (deprecated). It supports conversion of count tables to count transformations.
Merge Count Transform	Merges two sets of count-based features
Modify Count Table Parameters	Modifies count-based features derived from an existing count table

See Also

Data Transformation
Feature Selection