Working with Categorical Data

Categorical data has a *specific set* of possible values. For example:

- The different species of animals in a national park
- The names of streets in a particular city
- Whether or not an email is spam
- The colors that house exteriors are painted
- Binned numbers, which are described in the Working with Numerical Data module

Vocabulary and one-hot encoding

When a categorical feature has a low number of possible categories, you can encode it as a **vocabulary**. With a vocabulary encoding, the model treats each possible categorical value as a *separate feature*. During training, the model learns different weights for each category.

One-hot encoding

The next step in building a vocabulary is to convert each index number to its **one-hot encoding**. In a one-hot encoding:

- Each category is represented by a vector (array) of N elements, where N is the number of categories. For example, if `car_color` has eight possible categories, then the one-hot vector representing will have eight elements.
- Exactly *one* of the elements in a one-hot vector has the value 1.0; all the remaining elements have the value 0.0.

Outliers in Categorical Data

Like numerical data, categorical data also contains outliers.
Suppose `car_color` contains not only the popular colors, but also some rarely used outlier colors, such as `"Mauve"` or `"Avocado"`. Rather than giving each of these outlier colors a separate category, you can lump them into a single "catch-all" category called *out-of-vocabulary (OOV)*. In other words, all the

outlier colors are binned into a single outlier bucket. The system learns a single weight for that outlier bucket.

Common Issues

Numerical data is often recorded by scientific instruments or automated measurements. Categorical data, on the other hand, is often categorized by human beings or by machine learning (ML) models. *Who* decides on categories and labels, and *how* they make those decisions, affects the reliability and usefulness of that data.

Human Raters

Data manually labeled by human beings is often referred to as *gold labels*, and is considered more desirable than machine-labeled data for training models, due to relatively better data quality.

This doesn't necessarily mean that any set of human-labeled data is of high quality. Human errors, bias, and malice can be introduced at the point of data collection or during data cleaning and processing. Check for them before training.

Machine Raters

Machine-labeled data, where categories are automatically determined by one or more classification models, is often referred to as *silver labels*. Machine-labeled data can vary widely in quality. Check it not only for accuracy and biases but also for violations of common sense, reality, and intention. For example, if a computer-vision model mislabels a photo of a chihuahua as a muffin, or a photo of a muffin as a chihuahua, models trained on that labeled data will be of lower quality.

Similarly, a sentiment analyzer that scores neutral words as -0.25, when 0.0 is the neutral value, might be scoring all words with an additional negative bias that is not actually present in the data. An oversensitive toxicity detector may falsely flag many neutral statements as toxic. Try to get a sense of the quality and biases of machine labels and annotations in your data before training on it.

High Dimensionality

Categorical data tends to produce high-dimensional feature vectors; that is, feature vectors having a large number of elements. High dimensionality

increases training costs and makes training more difficult. For these reasons, ML experts often seek ways to reduce the number of dimensions prior to training.

Feature Crosses

Feature crosses are created by crossing (taking the Cartesian product of) two or more categorical or bucketed features of the dataset. Like polynomial transforms, feature crosses allow linear models to handle nonlinearities. Feature crosses also encode interactions between features.

When to use Feature Crosses?

Domain knowledge can suggest a useful combination of features to cross. Without that domain knowledge, it can be difficult to determine effective feature crosses or polynomial transforms by hand. It's often possible, if computationally expensive, to use neural networks to *automatically* find and apply useful feature combinations during training.

Be careful—crossing two sparse features produces an even sparser new feature than the two original features. For example, if feature A is a 100-element sparse feature and feature B is a 200-element sparse feature, a feature cross of A and B yields a 20,000-element sparse feature.

Completion

You earned the **Machine Learning Crash Course: Categorical data** badge!

The badge has been added to your profile.