# Handling Categorical Data

Encoding of categorical features or variables is required to convert them to numerical form. This is needed because most modelling algorithms require features to be all numeric. This section discusses some of the attempts made by researchers in encoding.

A transformation scheme was proposed by Barreca et. al. [1] which was based on the observed value of the target attribute. The encoding, popularly known as Target Encoding, estimates the Empirical Bayesian Probability of the target given the category level with prior being observed from the training data.

While some machine learning practitioners tend to avoid using high-cardinality features in their model just for the sake of simpler models, Moeyersoms et.al. [2] aimed at investigating if including high cardinality features actually improve the model performance. Specifically, for the case of churn prediction in an energy sector company, the authors identify three high-cardinality features- family name, bank account number and zip codes. They propose couple of techniques based on transformation of identified features to a target statistic. The first method is Weight of Evidence, which replaces churners with log of ratio of churners to non-churners, and similarly for non-churners. The second method of transformation is the Supervised ratio. This is a ratio inspired from social network perspective, given by

(1)

where and are again the number of churners and the number of non-churners for the ith value of attribute X respectively. An ‘unseen’ value for X receives an SR-score equal to the average churn rate (TC/(TC + TN)). A third encoding technique based on Perlich’s work was also used, which uses cosine distance between case vectors. On performing prediction on multiple combinations of these methods of encoding, the WOE gives the best results in terms of AUC and lift (0.1%) whereas SR performs best in terms of TPR (1%), precision (1%) and lift (1%). The PR has the highest TPR (5%) and precision (5%).

Guo et. al. [3] employ entity embeddings coupled with neural networks to map discrete variables to a multi-dimensional space where similar categories are placed closer. A neural network is used wherein an entity embedding layer learns about the intrinsic property of each category by building on top of a one-hot encoding layer. Each category of a high cardinality predictor can be mapped to a vector. The vector is similar to one hot encoded vector, but shorter in length. It’s given by:

(2)

where is a vector of length equal to number of categories in the variable xi, in which the element is only non-zero when complexity parameter α= xi. is the weight connecting the one-hot encoding layer to the embedding layer and β is the index of the embedding layer. Thus, mapped embeddings are just the weights of embedding layer and can be learned in the same way as the parameters of other neural network layers. Finally, this vector and input of continuous variables is concatenated, and merged layer is treated as any normal Neural net layer. Through experimentation, the authors show that this method can improve performance of machine learning algorithms such as kNN, random forest gradient boosting trees and neural networks as well.

In their article focused on reducing the AutoML framework to optimizing gradient boosting model, Janek et. al. [4] perform categorical feature transformation as well. They identify three methods to encode categorical variables, encoding features into integers, one-hot encoding and impact encoding (similar to target encoding). Authors evaluate different combinations of encodings, mainly based on a threshold, e.g., features with less than k levels are dummy encoded while integer or impact encoding is done for the remaining categorical features. In their framework, it is also possible to tune this threshold k together with the GBT hyperparameters. The authors show that

While designing an alternative method of building trees, a different approach is taken by Nguyen et. al. [5] with respect to categorical features. The authors propose a new feature sampling method for subspace selection, which is based on feature permutation to measure the importance of features and produce raw feature importance scores. They assess p-values for correlation between the features and response feature and group the features into high, medium and low importance features. This helps to find the cut-off between informative and uninformative features. When splitting a node, a greedy algorithm is used to identify the best split. For a categorical feature, a set of randomized values of a high-cardinal category is made, and a cut-point is selected based on maximum decrease in node impurity. This approach reduces computational complexity and can handle very high cardinality. With reduced feature space this algorithm outperforms many RF algorithms and can perform well on high dimensionality data.

Cerda et. al. [6] introduce two encoding techniques, a minhash encoder for fast approximation of string similarities, and Gamma-Poisson matrix factorization on substring counts. Minhash encoding works by grouping together similar character strings. It is based on locality-sensitive hashing and approximates jaccard coefficient between two strings. The authors claim that this method provides high levels of scalability as it is fast and efficient. Being stateless, it can work in parallel on workers in distributed systems. The drawback is, however, loss in interpretability as the strings get hashed. In order to provide interpretability, authors suggest another encoding technique, Gamma-Poisson matrix factorization. This method relies on substring representation of the string entities in the categorical variables by assuming a Poisson distribution on the n-gram counts of categories, with a Gamma prior on the activations. The authors contend that both their algorithms scale linearly with number of samples and therefore can be used in streaming settings.

Based on the literature, we can broadly classify the most popular encoding techniques in three classes. These are Classic Encoders, Contrast Encoders, and Bayesian Encoders. We discuss these in greater detail in the next section.

# Category Encoders

## Classic Encoders

### Label Encoding or Ordinal Encoding

In this method, the categories are simply mapped to an integer. The models then treat the categories as numeric which inappropriate in most cases. For example, if the categories are Male, Female and Others, these are converted to 1,2 and 3, respectively. If a new category appears later in the test data, it is generally replaced with 0 or -1. While this does not make sense in the case of feature Gender, this encoding may be useful in another feature such as Quality, having factors Excellent, Good and Bad, represented by 1, 2 and 3.

### One-Hot Encoding or Dummy Encoding

This method works by splitting the categorical feature into as many features as are the number of categories in the original feature. Each feature, thus made, represents a category and each observation can be given a binary 1 or 0 value depending on if the observation has that category or not. For instance, a column representing Gender, having values Male, Female and Others, can be split into three columns, Gender\_Male, Gender\_Female and Gender\_Others, each having 1 or 0 values depending on the observation.

In some applications, one of all the columns is omitted as it can be inferred based on other columns. In the previous example, if Gender\_Male and GenderFemale are both 0, then Gender\_Other becomes 1, meaning that Gender\_Other column can be safely removed. Though extensively used in literature, a major limitation of this method is its memory inefficiency. This is because the number of features increase as the number of categories increase. Therefore, the data becomes expensive in terms of memory usage.

### Binary Encoding

In Binary Encoding, the variable is first converted using Label Encoding. The resulting integer is converted to its binary representation. Finally, the binary string is split with each column representing a digit of the binary representation. Binary Encoding helps in tackling the problem of high cardinality as seen in One-Hot Encoding as it substantially reduces the number of variables created. For instance, if there are 16 categories of a variable, One-Hot Encoding would create 15 columns, but a Binary Encoding would take up only four columns because integers from 0 through 15 can be encoded in a four-length string.

### Hashing

Hashing is a technique to replace a string with a fixed-length vector. There could be may hash functions, each of them being common in that a hash function always returns same vector for a given string. Hash encoding therefore gives new columns just like one-hot encoding, but the number of columns can be controlled by the user. Since the length of output vector is fixed, this method solves the problem of new categories appearing in the test data, which is one of the major limitations of one-hot encoding.

## Contrast Encoders

### Helmert Encoding

In Helmert encoding [7], mean of target for a level or category is compared to mean of target for all the subsequent levels taken together. The number of resulting columns from this encoding depends on the number of pairs wherein the difference of the means is found to be statistically significant. This helps in reducing the problem of high cardinality. A variation of Helmert Encoding is reverse helmert encoding, wherein instead of comparing target means of a level and its subsequent levels, the target means of a level and its previous levels are compared.

### Difference Encoding

The mean of target variable for a level is compared to the target mean of the adjacent level. If the subsequent level is considered, it is called as forward difference encoding. If previous level is considered, it is called as backward difference encoding. These encoding could prove useful with either nominal or ordinal features.

### Sum Encoding

Each level is compared to all other levels collectively by comparing target means. This means that the target mean for level k is compared to target mean of all other k-1 levels.

## Bayesian Encoders

### Target Encoding

This method introduces a transformation scheme [1] wherein one maps each instance (value) of a high-cardinality categorical to the probability estimate of the target attribute. In a classification scenario, the numerical representation corresponds to the posterior probability of the target, conditioned by the value of the categorical attribute. In a prediction scenario, the numerical representation corresponds to the expected value of the target given the value of the categorical attribute. In order to avoid overfitting due to small number of observations in a category, smoothening of the means is also applied. Probability estimate for a category within a high cardinality categorical variable can be given by Empirical Bayesian probability, P(Y=1|X=Xi), i.e.

(3)

for all training count and i-th category row count . λ is a function which gives monotonic weight which helps when we have a small number of several categories, increasing with count from 0 to 1. Introducing the weighting factor makes sense because when the sample size is large, we should assign more credit to the posterior probability estimate provided by first term above. However, if the sample size is small, then we replace the probability estimate with the null hypothesis given by the prior probability of the dependent attribute (i.e. mean of all Ys). With this transformation, missing values are handled by treating them as just another variable . This has advantage that if nulls have a predictive relevance, then will capture that information, otherwise it will converge towards the prior probability of target, leading to 0 effect of .

### Weight of Evidence Encoding

This method [2], having originated from credit scoring sector, can be used to transform a category of the feature into a numeric value by taking relative ratio of appearance of the category in the dataset.

(4)

TC and TN define the total number of instances of target class c1 versus target class c2; and denote the number of c1 and c2 for the ith value of attribute X.However, in case when values are zero in a particular category, 1 row is added for that category and over all ratio modified so that it is equal to TC/TN. It is recommended that the calculation of WOE be done using a separate part of the training data instead of whole data, in order to avoid overfitting.

### Leave One Out Encoding

This method was introduced to counter the effects of outliers in the training data. Similar to Target Encoding, mean of each level is calculated for the observation’s level in question, but the observation itself is left out. This makes sure that if the observation is an outlier, it does not bring bias in the calculation of category mean.

### James-Stein Encoding

The weight λ in equation 1 is a parameter that needs to be tuned explicitly. Giving more weight to a category would lead to overfitting, while giving more weight to global mean would lead to underfitting. In order to solve this problem, James-Stein encoder gives lesser weight to a category if variance in values of that category is high as compared to overall variance in target.

# References

1. A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems, Daniele Micci-Barreca
2. Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector, <http://dx.doi.org/10.1016/j.dss.2015.02.007>, Julie Moeyersoms, David Martens
3. Entity Embeddings of Categorical Variables, Cheng Guo and Felix Berkhahn
4. Automatic Gradient Boosting, Janek Thomas, Stefan Coors, Bernd Bischl
5. An efficient random forests algorithm for high dimensional data classification, Wang, Nguyen
6. Encoding high-cardinality string categorical variables, Patricio Cerda and Gael Varoquaux
7. R Library Contrast Coding Systems For Categorical Variables, <https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#HELMERT>
8. <http://contrib.scikit-learn.org/category_encoders/index.html>
9. <https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159>