Beta Target Encoding

Matt Motoki (/author/matt-motoki.html), Wed 04 July 2018, Wed 04 July 2018, Misc (/category/misc.html)

Bayesian Statistics (/tag/bayesian-statistics.html), Feature Engineering (/tag/feature-engineering.html), Kaggle (/tag/kaggle.html)

Beta Target Encoding

Summary

Bayesian Target Encoding is a feature engineering technique used to map categorical variables into numeric variables. The Bayesian framework requires only minimal updates as new data is acquired and is thus well-suited for online learning. Furthermore, the Bayesian approach makes choosing and interpreting hyperparameters intuitive. I developed this technique in the recent Avito Kaggle Competition, where my team and I took 14th place out of 1,917 teams. We found that the Bayesian target encoding outperforms the built-in categorical encoding provided by the LightGBM package.

Target Encoding

Most machine learning algorithms require numeric input data. Target encoding is a common technique used to convert categorical variables into numeric variables. When done properly, target encoding can be very effective. Target encoding is built into popular machine learning algorithms such as LightGBM (http://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features) and CatBoost (https://tech.yandex.com/catboost/doc/dg/concepts/algorithm-main-stages_cat-to-numberic-docpage/).
In supervised learning, we are given N data points (x_i, y_i) and tasked with learning a mapping from input variable x to the target variable y. Suppose x is categorical and takes on one of l possible (non-numeric) levels $x \in \left\{x^{(1)}, x^{(2)}, \ldots, x^{(l)}\right\}$. Target encoding maps each level of x into a feature $\phi \in \left\{\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(l)}\right\}$ in the following way:

$$\phi^{(j)} = \frac{1}{N^{(j)}} \sum_{i=1}^{N} y_i \cdot \mathrm{I}\left\{x_i = x^{(j)}\right\} \tag{1}$$

where $N^{(j)} = \sum_{i=1}^{N} \mathrm{I}\left\{x_i = x^{(j)}\right\}$ and $\mathrm{I}\{\cdot\}$ is the indicator function. In other words, $\phi^{(j)}$ is the average y-value in level j and $N^{(j)}$ is the total number of observations in level j.

Overfitting and Regularization

Target-encoded variables are inherently leaky; that is, their construction requires information that we will not have when we make predictions. To see why this can be a problem, consider the case when $N^{(j)} = 1$. In this case, the summation in (1) involves only one non-zero term when $x_i = x^{(j)}$. Thus, we have $\phi_i = y_i$; that is, ϕ_i is encoded with exactly the value we are trying to predict. Using these target-encoded features as is will lead to overfitting of the training set and poor generalization of our models. This problem is acute when dealing with high cardinality categorical variables. We can combat the data leakage problem by smoothing the estimate of the average y-value in level j. In particular,

$$\phi^{(j)} = \lambda^{(j)} \cdot \frac{1}{N} \sum_{i=1}^{N} y_i + \left(1 - \lambda^{(j)}\right) \cdot \frac{1}{N^{(j)}} \sum_{i=1}^{N} y_i \cdot \mathrm{I}\left\{x_i = x^{(j)}\right\}, \tag{2}$$

which is just a weighted average of the estimate in (1) and the average y-value value over the entire training set. The $\lambda^{(j)}$ is a hyperparameter that controls the amount of smoothing in the estimate. Alternatively, we could address data leakage by adding noise to the estimate

$$\phi^{(j)} = \eta \cdot \epsilon^{(j)} + (1 - \eta) \cdot \frac{1}{N^{(j)}} \sum_{i=1}^{N} y_i \cdot \mathrm{I}\left\{x_i = x^{(j)}\right\}. \tag{3}$$

The hyperparameters $\lambda^{(j)}$, η, and the distribution of the random variable $\epsilon^{(j)}$ need to be chosen carefully for target encoding to perform well. By adopting a Bayesian framework, we can handle regularization in a systematic yet intuitive way. Furthermore, the Bayesian framework is well-suited for online learning requiring only minimal updates as new data is acquired. In the next section, we will discuss target encoding in the context of binary classification.

Bayesian Target Encoding

In Bayesian statistics, we represent our beliefs about the world using a prior distribution. Approaching target encoding from a Bayesian perspective has the following benefits:

- hyperparameters are easy to interpret
- hyperparameter estimation is well-suited for online learning
- generalizing the encoding to statistics other than the mean is easy
- choosing the distribution for the noise random variable is easy

The analysis here considers target encoding for binary classification where the target variable y takes on two discrete values $\{0, 1\}$. The same line of reasoning applies to target encoding for soft binary classification where y takes on values in the interval $[0, 1]$. The general theory applies to other target variable types.

The Beta Distribution[1] (https://en.wikipedia.org/wiki/Beta_distribution)

It is convenient to model binary target variables using the Bernoulli distribution, which has the Beta distribution as its conjugate prior. The Beta distribution is parameterized by α and β, which can respectively be thought of as the number of positive and negative examples in a repeated Binomial experiment. Many useful statistics of the Beta distribution can written in terms of α and β; for example, the mean

$$\mu = \frac{\alpha}{\alpha + \beta}, \tag{4}$$

the variance

$$\sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}, \tag{5}$$

and the skewness

$$\gamma = \frac{2(\beta - \alpha)\sqrt{\alpha + \beta + 1}}{(\alpha + \beta + 1)\sqrt{\alpha\beta}}. \tag{6}$$

Interpreting the Hyperparameters[2] (https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf)

When deciding on the prior distribution, we need to set values for α_{prior} and β_{prior}. One way to do this is with grid search and cross-validation. However, we should not want to search over α_{prior} and β_{prior} independently. Instead, we can reduce the search space by requiring α and β to be such that the mean of the prior distribution is constant. The obvious choice for prior mean is the average y-value in the training set $\mu_{prior} = \frac{1}{N}\sum_{i=1}^{N} y_i$, but the prior mean can also be set in other ways. If the categorical variables are hierarchical, then the prior mean can come from one step up in the hierarchy. With the mean known, we can reparameterize the beta distribution in terms of $\tau = \alpha_{prior} + \beta_{prior}$. Intuitively, τ can be thought of the effective sample size from the prior distribution. Rewriting (4) in terms of τ we get

$$\alpha_{prior} = \tau \cdot \mu_{prior} \quad \text{and} \quad \beta_{prior} = \tau \cdot (1 - \mu_{prior}).$$

The parameters of the posterior distribution are given by

$$\alpha_{posterior}^{(j)} = \alpha_{prior} + \sum_{i=1}^{N} y_i \cdot \mathrm{I}\left\{x = x^{(j)}\right\}$$

and

$$\beta_{posterior}^{(j)} = \beta_{prior} + N^{(j)} - \alpha_{posterior}^{(j)}.$$

These equations show the simple update rules for the posterior parameters—all we need to keep track of is the sum of the y-values in a level and the total number of observations in that level. When the posterior parameters are known, we can use them to calculate other statistics, such as those in (4), (5), and (6). To see how τ controls complexity let's rewrite the posterior mean as

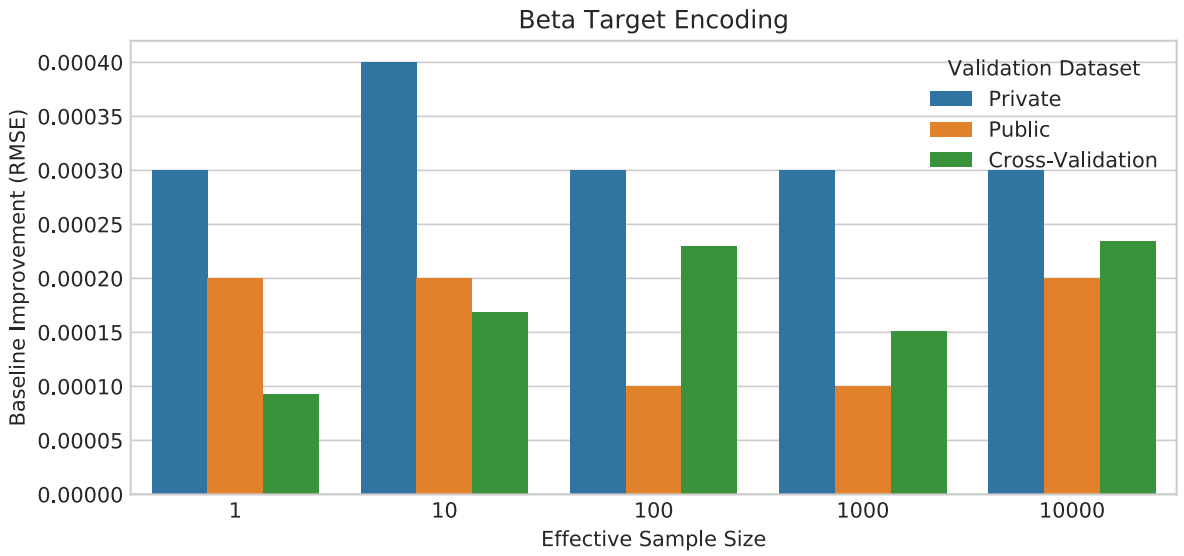
$$\mu_{posterior}^{(j)} = \lambda^{(j)} \cdot \mu_{prior} + \left(1 - \lambda^{(j)}\right) \cdot \frac{1}{N^{(j)}} \sum_{i=1}^{N} y_i \cdot \mathrm{I}\left\{x = x^{(j)}\right\},$$

where $\lambda^{(j)} = \tau \,/\, \left(N^{(j)} + \tau\right)$. Notice that this equation has the same functional form as (2). We see that increasing τ brings the estimate closer to the prior mean and that decreasing τ brings it closer to the in-level sample average.
Recall that when we use additive noise for regularization (3), we need to specify the distribution of the noise random variable $\epsilon^{(j)}$. This is natural to do when taking the Bayesian approach. In particular, we can set $\epsilon^{(j)} \sim \mathrm{Beta}\left(\alpha_{posterior}^{(j)}, \beta_{posterior}^{(j)}\right)$. The only hyperparameter left to tune is the noise strength η. Notice that from (5), the variance of the noise distribution will shrink as we obtain more observations. What this means is that as the number of observations increases, the noise distribution will converge to a delta function centered at $\mu_{posterior}^{(j)}$. Thus, we have the nice property that the amount of regularization decreases as the number of observations increases.

Case Study: Kaggle - Avito Demand Prediction[3] (https://www.kaggle.com/mmotoki/avito-target-encoding)

My team and I used Bayesian target encoding in the recent Kaggle competition Avito Demand Prediction Challenge (https://www.kaggle.com/c/avito-demand-prediction) where we placed 14th out of 1,917 teams. For a more detailed write-up about our team's solution, see Peter Hurford's post (https://www.kaggle.com/c/avito-demand-prediction/discussion/60059). The task was to predict demand for an online advertisement based on its full description (title, description, images, etc.), its context (geographically where it was posted, similar ads already posted), and historical demand for similar ads in similar contexts. The competition metric was the root mean squared error between our prediction and the probability of a deal. We found that the

Bayesian target-encoded features outperformed the built-in LightGBM categorical variable encoding. The graph below shows the performance of a LightGBM model using target encoding for various values of τ relative to a baseline LightGBM model using built-in categorical encoding. The best model uses Bayesian target-encoded features with a hyperparameter setting of $\tau = 10$.



The figure above shows the relative improvement of the beta target-encoded features compared to the built-in LightGBM encodings. We see that the beta target encodings are better than the baseline for a wide range of hyperparameter settings.

References

- *[1 (https://en.wikipedia.org/wiki/Beta_distribution)] A more comprehensive list of statistics for the Beta distribution.*
- *[2 (https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter9.pdf)] Information about Bayesian statistics and conjugate priors.*
- *[3 (https://www.kaggle.com/mmotoki/avito-target-encoding)] Code for implementing the Bayesian target encoding.*