Logistic regression

Ivan Nikolov[1]

[1] *in7357@student.uni-lj.si, 63190378*

Multinomial regression

Algorithm implementation and correctness

Multinomial regression is a generalized linear model that is used for multiclass classification scenarios. It is assumed that the target variable follow a categorical distribution. Compared to the regular logistic regression, instead of the inverse logit function, it uses the softmax as the link function.

The prediction of the model can be written with matrix notation as follows:

$$y = softmax(XW + b)$$

, where W are the model weights with size of (# features, # classes), and b is the intercept, which is a 1D vector (with dimension of # classes). The loss we use is the log-loss, that for multicategorical target classes can be generalized to cross-entropy. To test the algorithm correctness, we implemented tests (located in `custom_tests.py`) with synthetically created datasets with a known DGP, and compared our implementation metrics with the existing ones from `sklearn`. Both methods exhibited similar performance in terms of metrics and uncertainty. In addition, we did an analysis on the coefficients and they had similar values.

Application

We used our implementation of multinomial regression for predicting the shot-type from the dataset given in the `dataset.csv` file.

Data preprocessing was mostly done by following the paper linked with the dataset. The nominal variables (competition type, player type, movement) were dummy encoded. The two continuous features (angle, distance) were standardized such that they mean of 0, and standard deviation of 1.

Estimation of standard errors was done using bootstrap. In a series of 1000 iterations, we sampled the model predictions and corresponding class values with replacement. In each iteration, we calculated the misclassification rate. At last, based on the attained distribution, we calculated the standard deviation and used that as an estimate for the standard error.

Evaluation was done using 10-fold cross-validation.

On Figure 1 we can see that best performance is achieved on layup and above head shot, and they are most frequent in the dataset. The model also often mixes up hook shots and 'other' with above head. Dunks and tip ins are mixed up with layups because of their similar nature.

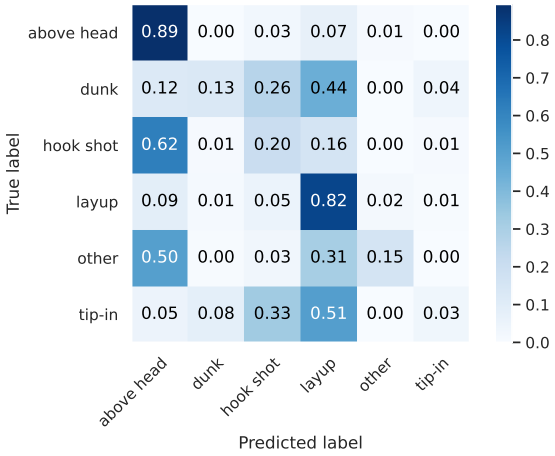

Figure 1. Normalized confusion matrix generated by 10-fold CV

We trained the model multiple times with different bootstrap samples from the full dataset. The coefficient values are shown on Figure 2. Upon closer inspection, we can see that two legged has high values for layup, tip-in and dunk, while low for other types of shots. This makes sense because the shots mentioned are mostly two legged in practice. Also, with large distance, the coefficients for above head and 'other' are larger. Dunk, tip-in and layup due to their nature have to be performed near the hoop. Interesting patterns can be noticed on the league attributes, namely dunk is higher in NBA than the other leagues. For the junior leagues because of height differences, coefficients for dunk and tip-in are relatively low. Because we do not have a large domain knowledge about the player positions, we will leave the interpretation to the reader. Shots executed with dribble and cut are above head and layup, layup also cannot be executed without any movement and it has a negative coefficient. On the other side, we have the hook shot, which is typically executed without any motion.

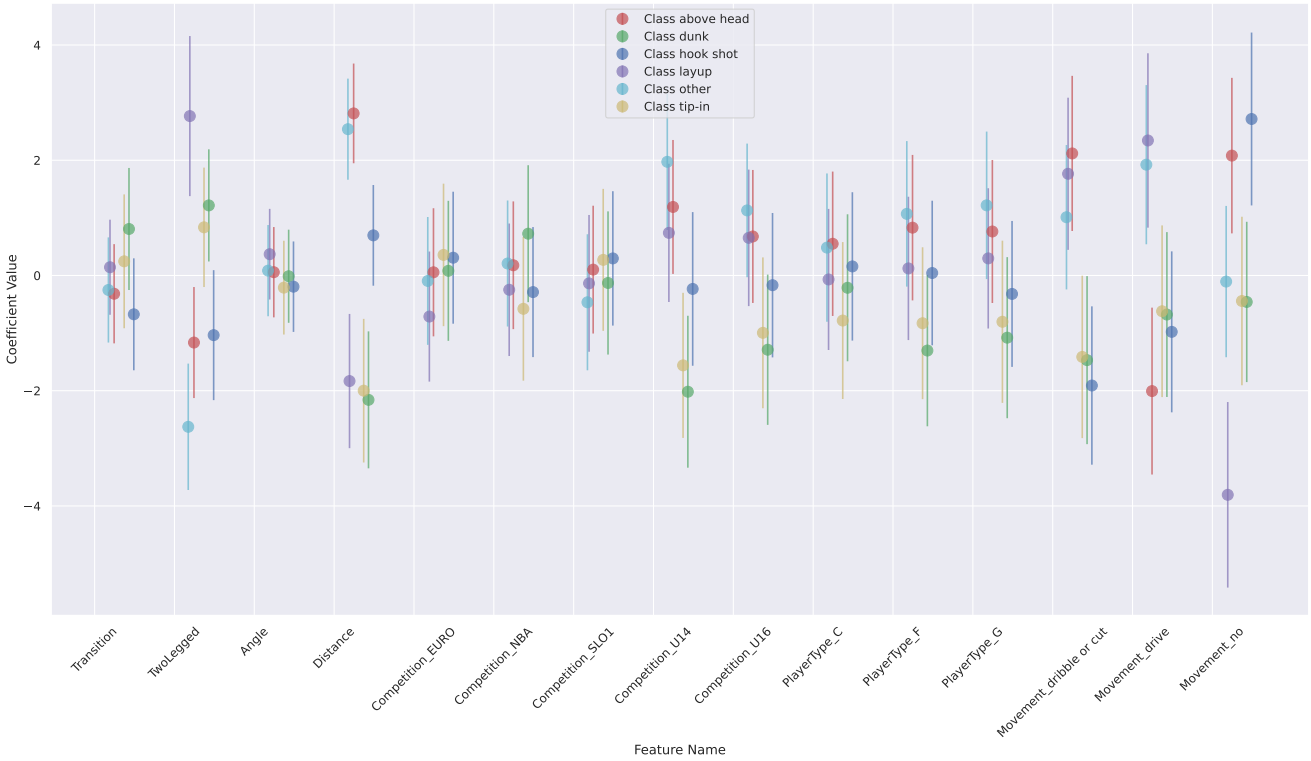

Figure 2. Regression coefficients by feature and class The whiskers denote 2 standard deviations (calculated using bootstrap).

Ordinal regression

Algorithm implementation and correctness

Ordinal regression is another generalized linear model that works well for ordinal (ordered) data. Although, multinomial regression with enough data can perform equally well as ordinal regression, the ordinal regression is simpler and has fewer parameters. It works as a logistic regression with additional thresholds that separate the data into classes.

When optimizing the parameters based on the loss function, we optimized the deltas between the thresholds and set a box constraint of 10^{-5} (deltas need to be bigger than the constraint).

Table 1. Accuracy on ordinal DGP

	Accuracy
Multinomial	0.883 ± 0.010
Ordinal	0.966 ± 0.006

We also made a data generating process in that way that we randomly generated multiple uniform features with normal noise and thresholded them into three classes. When the training set is small (in our case 30), the ordinal regression performs better than the multinomial regression