Unsupervised Feature Creation with TransE

Leon Lin, Griffin Young

December 11, 2019

1 Motivation

Typical machine learning models rely on the human creation and curation of feature vectors which are fed into a predictor to generate a prediction. However, this process is tedious and is limited by the insight and expert knowledge of the human creating the model. We were interested in the automatic generation of these features vectors to sidestep the problems associated with manual feature-engineering. While tools like Featuretools and ExploreKit exist for choosing and even engineering new features through the combination and modification of raw feature types, we were curious whether the expressive power of a graph structure could implicitly perform the same kind of automatic feature aggregation. In order to test this idea, we chose the task of predicting the result of soccer games. This is a straightforward classification task that also has a graph structure to the data in that players have links to the teams they play for, teams have links to the matches they've played, etc.

2 Related Work

2.1 ExploreKit: Automatic Feature Generation and Selection

This algorithm, representative of many alternative automatic feature generation tools, seeks to develop and test potential new features created from a pool of raw features. It consists of three phases repeated for some number of iterations: feature generation, feature ranking, and feature selection. First, a pool of candidate features are produced from the pool of existing features and list of operations. Next, these candidate features are ranked with a ML model that is fed the dataset and candidate feature. This ML model has previously been trained on a diverse set of datasets in which candidate features were produced and the model learned to predict good (i.e. adding the candidate feature would decrease the error by some threshold value) or bad. Finally, the list of ranked candidate features is greedily traversed in descending order until a candidate feature is found which, when added to the bank of current features, decreases the classification error on the dataset by some threshold amount. This candidate feature

is then added to the bank of current features and another iteration of the algorithm begins.

2.2 TransE

The TransE algorithm develops embeddings in a k-dimensional space (k being a hyperparameter) for every entity and every relationship such that if two entities h and t have a relationship l, eg h played for t, then $h + l \approx t$. First, embeddings for every relationship and every entity are randomly initialized with a uniform distribution between - 6/sqrt(k) and 6/sqrt(k). Embeddings for the relationship embeddings are then normalized by their magnitude. Next, the training begins. For some number of iterations (also a hyperparameter), the following is done: 1. The entity embeddings are normalized by their magnitudes. 2. A batch of actual (h,l,t) triples is sampled from the edges of the graph. 3. For each of the triples, a corrupted triple is created in which one of h or t is replaced by a random node in the graph. 4. A loss function is defined as max(0, y+d(h,l,t)-d(h',l,t')). Y is a margin included to prevent trivial loss minimization by making the embeddings for all nodes the same; d is a distance function. The gradient of this loss function is summed with respect to each embedding and then each embedding is updated by this sum times a learning rate.

3 Dataset

The Kaggle European Soccer Dataset contains records of 25k+ games from 2007 to 2016, listing the goals scored by the home and away teams for each. While the games all have goal info, team info, and country info, Of these games, approximately 22k have full player information (the 11 players on each team). Each player has a series of 38 features such as preferred foot, crossing, stamina, etc. These features change over time, with some players having as few as 3 sets of features and some as many as 56.

3.1 Preprocessing

For our prediction task, we only used the 21,374 matches with full player data. For both the baseline and advanced models, the targets of the predictions were the results of the match as a multi-class prediction task, with 0 being a home win, 1 being an away win, and 2 being a tie.

To make the dataset for the standard baseline predictor, we took each of the 21,374 matches with full player information and concatenated the player features for all 22 players (11 home, 11 away) at the timestamp closest to the time of the match into a length 836 feature vector.

To prepare a graph for Trans-E, we treated each player (11,060 total), country (11 total), team (299 total), and matches (21,374 total) as nodes. Each player has multiple "plays for" links to all the teams they have played for in the span of the dataset. Each

team has a "team in" link to a country. Each match has a "home team" and "away team" link to teams and a "match in" link to a country. The associations between players and teams and between teams and countries were inferred from matches: if a player is on the team for a match, it is assumed that "plays for" that team; if a match is in a certain country, the home team is assumed belong to that country. As players change teams throughout their careers, they can have multiple "plays for" links—up to 8 in this dataset.

4 Methods

4.1 TransE

In order to automatically generate feature vectors to feed into the predictor, we first used the TransE algorithm to generate embeddings for just players, countries, games, and the following relationships: "played for", "team from", "home team", "away team", "match in". In a second trial we included player attributes – 38 Fifa-style ratings, like dribbling or overall rating, of which we included the 35 which were scaled from 1 to 100 – by making a node for each bucket of 10 between 0 and 100 for each of the 35 attributes. We also included a new relation, "has stat", which went from players to the stat bucket they fell into for each of the 35 statistics.
We chose the hyperparameters of k=50, y=1, learning rate =0.001, learning rate decay of .95 per epoch, and L2 norm as the distance function as was suggested in the original TransE paper.
Several initializations were attempted and run for 50 epochs each. Loss was typically minimized around 20-40 epochs. We then used the embeddings with the lowest loss.

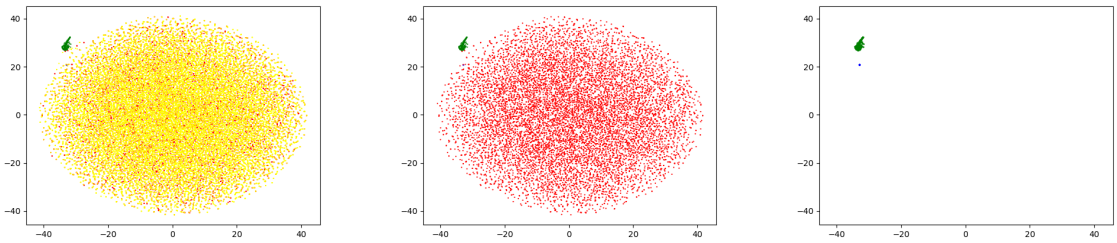
4.2 Predictor

For the baseline predictor, we simply used a dense neural network with the 836 player attribute features as inputs through several layers to 3 outputs that then used as logits for a weighted softmax (weights inversely proportional to the frequency of the class). After manual hyper-parameter testing, we found the best network to have 5 layers with [800, 500, 100, 40, 3] features in those layers. ReLu is applied after every layer except the final one, and the model was trained with SGD with learning rate of 1e-4 with momentum of 0.9.
This same model was then applied to 2 additional datasets created with the Transe embeddings. First, we concatenated the embeddings for each of the 22 players, and for the 2 teams, the country, and the match into a 1300 dimensional feature vector. The second model used these features to try to predict the match results. The third concatenated these features with the player features into a 2136 dimensional vector to predict the match results.

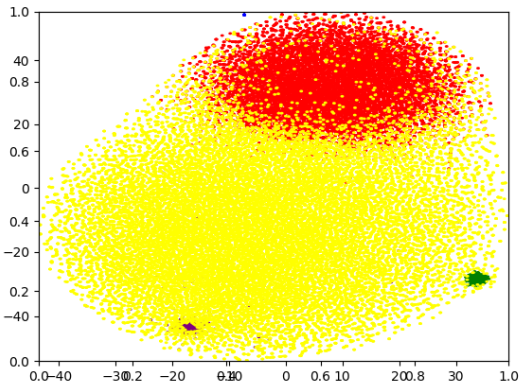
5 Results

5.1 TRANSE

To visualize the 50-dimensional Transe results, we used TSNE to project it onto 2 dimensions, shown below. Players and matches appear scattered all over, but removing them shows that countries and teams are grouped quite close together, indicating that their is some meaning in these embeddings.

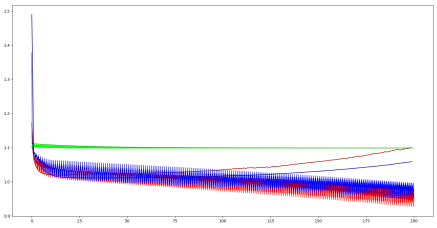


TSNE of 50-dimensional embeddings. Matches in yellow, players in red, teams in green, countries in blue.



TSNE of 50-dimensional embeddings for the second trial. Matches in yellow, players in red, teams in green, countries in blue, stats in purple.

5.2 Predictor



Training and validation losses for predictors. Using only player features in red, both Transe Embeddings and player features in blue, and using only the embeddings in green. Validation losses in darker red, blue, and green respectively

Ultimately, adding the Transe embeddings did not improve the predictions. The 21k matches were split into a standard 70/20/10 sets for training, validation, and test. Each model was trained for 200 epochs, with best validation loss typically reached around epoch 40. The models with the best validation losses for each of the three was then used on the test set. Using Transe embeddings by themselves resulted in a model that was no better than random guessing.

Test Set accuracies for the three categories

	Features only	Embeddings only	Both
Average	45.8%	32.3%	45.1%
Home Win	68.0%	50.5%	63.9%
Away Win	54.6%	0%	54.5%
Tie	14.9%	24.3%	16.8%

Average accuracy was the accuracy across the three categories, averaged disregarding the quantities in the categories. Random guessing would result in 33% accuracy when measured in this way. The model with only embeddings is equivalent to random guessing, while the models with only the features and the model using both features and embeddings are within equivalent within rounding error.
Including nodes representing brackets of player attributes (e.g. finishing being broken down into 10 nodes representing a rating from 0-10, 10-20, ..., 90-100) improved performance to slightly below baseline (44.8% average accuracy). More details on this in the Future Work section.

6 Reflections and Future Work

We have two hypotheses for why the Transe embeddings didn't help the predictions. First, the embeddings themselves are not trained with the results of the matches. That is, while the embeddings might be representations of relations, we did not include a relation from the match to the winning team. Second, the embeddings are not complex enough. We treated each player as possibly having multiple "plays for" links because team association changes over time. However, it might be more accurate to treat in some way incorporate how the team association changes over time, as we did with the

feature vectors for players attribute. The following are some ideas we thought of, the first of which we attempted.

6.1 More Node Types

We were encouraged by the improvements to the model incurred by adding player attributes to the embedding space. One possible extension of this idea is to simply increase the size of the graph by adding more information to it. One possible node type is expert knowledge, taking advantage of the knowledge graph structure TransE helps represent. This could include more subjective information such as "cohesive team" being pointed to by teams or "lots of crowd noise" being pointed to by countries.

6.2 Time

Team association changes over time, which can be inferred through the matches dataset (inferring association at a point in time based on the date of the match, and assuming association continuity in the time between matches). There are several possible ways to represent this. First, we can simply treat each player-team association as a different node. Thus, a player that has played for 8 different teams would have 8 different embeddings. Second, we could have each team have a different embedding for every soccer season in the dataset, which may be reasonable as a team with different members could be a fundamentally different entity. Third, we could have different nodes for each player in each season, with a "becomes" relation linking them to try to explicitly capture changes a player over time.