Aggregating Predictions vs. Aggregating Features for Relational Classification: A Comparison

Oliver Schulte
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada
Email: oschulte@cs.sfu.ca

Kurt Routley
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada
Email: kdr4@sfu.ca

***Abstract*—This paper presents a fast, principled, and accurate method for leveraging standard machine learning classifiers for relational classification. The method uses the average of a classifier score over a target's relational neighborhood as an aggregate classifier score. We show that the average is consistent with the classic random selection semantics for probabilistic logic. Our experiments compared different relational classifiers on sports, financial, and movie data. Compared to propositionalization methods and other score aggregators, the random selection classifier showed robust and competitive performance. We also demonstrate improvements to both feature aggregation and score aggregation methods. Feature aggregation classification can often be improved by adding the standard deviation as an aggregate feature. Score aggregation classification results on imbalanced datasets can often be improved by weighting vectors relative to the total number of data points.**

I. Introduction

Most real-world structured data are stored in the relational format, with different types of entities and information about their attributes and links between the entities. Relational data classification is the problem of predicting a *class label* of a target entity given information about features (attributes) of the entity, of the related entities, or neighbors, and of the links. This paper presents a fast, principled, and accurate method for leveraging standard machine learning classifiers for relational classification.

A key challenge for relational classification is that the number of links of the target entity is not uniformly bounded. Since the features of each neighbor potentially carry information about the target class label, the number of predictive features for classification is thus a function of the size of the target entities neighborhood, rather than a fixed dimensionality d. Relational classifiers therefore aggregate the information from the target entity's neighborhood. There are two fundamental options for aggregation: 1) First aggregate the neighbors' features into a single aggregate feature vector, then classify based on the aggregate vector. 2) First derive a classification score based on a single neighbor, then aggregate the scores. In this paper we compare the two approaches empirically on data sets with continuous features. Since standard relational benchmark datasets contain discrete features mainly or only, we use three real-world continuous datasets that summarize players' actions in ice hockey, soccer, and basketball. The ice hockey dataset was obtained by a web crawler and has not been analyzed before. Two additional datasets for financial data and IMDb reviews are analyzed and contained both continuous and discrete variables. Discrete variables are mapped to continuous variables using a feature function in order to facilitate continuous machine learning methods.

Computationally, classifier training with score aggregation can be done very simply by forming a data table such that one row contains the features of one neighbor, and applying a regular non-relational learning algorithm to this table.[1]

A. Evaluation

We use standard aggregation functions to aggregate continuous features (average, sum, min, max, midrange, geometric mean). Once features have been aggregated, any standard single-table machine learning classifier for continuous features can be applied for classification. In this paper we apply logistic regression and support vector machines (SVMs).

Both logistic regression and SVMs return a continuous classification score. We apply the single-table classifier to the features of each neighbor to obtain a classification score for the neighbor, then aggregate the scores. For aggregating scores, we use the same functions as for aggregating features, except for summation. In addition we apply noisy-or, a standard rule for combining a list of probabilities into a single probability.

The basic task is to predict the result of a given target team in a given target match (win or not). We examine hockey data from the NHL, soccer data from the UK Premier League, and basketball data from the NBA. The training set contains data from previous matches and the test set data from the following ones. We experiment with two different feature sets: First, summary statistics from the previous season for each player (e.g., number of goals scored by the player). Second, in addition, the action counts for each player on the target team in the target match.

Our main conclusion is that among score aggregation operators, the average or mean provides robust competitive performance across different settings. We show that the average operator is consistent with the classic random selection

[1] If the neighborhood sizes of different target entities differ, a simple adjustment of the classifier loss for neighborhood size is necessary, see details below.

semantics of probabilistic 1st-order logic due to Halpern and Bacchus [1], [2]. The performances of the best score and feature aggregation methods is close except for one dataset where feature aggregation is clearly superior. We outline a hybrid approach that combines informative aggregate features with score-based classification.

B. Contributions

Our main contributions may be summarized as follows.

1) A comparison of an extensive set of aggregators applied to both feature and score aggregation methods for relational classification.
2) A comparison of two different baseline classifiers (logistic regression and SVM) extended to relational classification via aggregation.

II. Related Work

Because of the importance of relational data, there has been much work on link-based classification. For overviews please see [3], [4]. We provide a high-level description of the work most relevant to the question of feature vs. score aggregation.

A. Aggregating Classifier Scores

Most approaches that aggregate classification scores use a function that maps a list of probabilities to a single probability. Following the terminology of Bayes nets, such functions are referred to as *combining rules* [5], [6]. In our terminology, a combining rule is a special kind of classifier score aggregation. Our experiments examine the commonly used combining rules (e.g. average, noisy-or).

B. Propositionalization

The majority of work on relational classification has adopted the feature aggregation strategy. This approach of "flattening" the relational structure is generally known as *propositionalization* [7]. For continuous features, propositionalization methods use the same standard aggregate functions that we use in this paper [8], [9]. For discrete data, a common approach is to use a *feature function*. A feature function maps a relational neighborhood to a single value for the given feature. For instance, if the feature is "student's grade is A", the count feature function returns the number of A's achieved by the student. If the feature function returns a continuous or integer value, the values of the feature functions are used as inputs to a log-linear model (conditional random field) for prediction [10], [11], [12], [13]. A feature function may also return a discrete value; a commonly used binary feature function is existential quantification, for example using 1 as a classifier if the student has achieved an A in some course, and a 0 otherwise.

do we use feature functions with discrete features?

C. Complex Features

The most expressive propositionalization models apply feature functions to combinations of the discrete features given in the data (e.g., [14]). For instance, to predict the ranking of a student, we may distinguish the number of A grades achieved in higher-level course from those achieved in lower-level courses. Complex discrete features may be combined with aggregation functions, as aggregation conditions, for continuous variables [9], [15]. For example, to predict the age of a user in a social network, we may consider the average age of her friends who have the same gender and live in the same city.

Several researchers discuss advantages and disadvantages of propositionalization for link-based classification [16], [3]. The main advantage is expressiveness: feature generation methods search a large space of potentially useful features. If an informative new complex feature or aggregate feature can be found, it improves classification performance and informs the user. The disadvantages are problems with both statistical and computational efficiency. Aggregation loses information in the data, which increases the variance of classifier estimates and causes problems with both type 1 and type 2 errors in feature selection [17]. Searching a large space of potential features presents considerable computational challenges. For an example, generating 100,000 features on the standard CiteSeer dataset, can take several CPU days [15, Ch.16.1.2]).

The feature aggregation method we use in this paper is intermediate between choosing a single fixed aggregate operator and searching through a space of complex expressions. For each original unaggregated feature, we apply a fixed set of aggregate operators (such as average, maximum, etc.). These are provided as input features to a standard learning method (e.g., logistic regression). So there is no search through a complex feature space, but learning is used to select and weight relevant aggregate features.

D. Sports Statistics

The problem of predicting the results of sports matches has received considerable attention for different sports. For an overview please see [18]. We do not claim that the methods in this paper are competitive for predicting the match results. We use sports data for real-world datasets in an interesting domain with interpretable features for comparing aggregating features vs. aggregating predictions.

The closest predecessor to our work is that of Neville *et al.* [19]. Key differences include the following. 1) They used only the average operator for feature aggregation, rather than a set of aggregate operators. 2) They used the arithmetic and geometric mean only for score aggregation. 3) They did not consider adjusting instance weights to improve score aggregation methods.

III. Notation and Data Format

We introduce notation to discuss relational features and data and to support theoretical analysis. We follow the functor-based notation for combining statistical and relational concepts due to Poole [20].

A. Functor Features

A **population** is a set of individuals, corresponding to a domain or type in logic. A **feature** is of the form $f(t_1, \ldots, t_k)$

where f is a functor (either a function symbol or a predicate symbol) and each t_i is a first-order variable or a constant. Each feature has a set of values (constants) called the **domain** of the feature. A feature whose range are the truth values $\{T, F\}$ is a **predicate**. Predicates are usually written with uppercase Roman letters, other feature with lowercase letters. A **grounding** replaces each 1st-order variable in the feature by a constant; the result is a ground feature. A grounding may be applied simultaneously to a set of features. One of the features is the class or **target** feature. A grounding of the target feature is a **target instance**.

B. Examples

In our datasets the basic populations are teams, players, matches, with corresponding first-order variables T, P, M. Examples of features include the following.

- $\mathit{result}(T, M)$ denotes the result of a team in a match (win or lose). This is the target feature.
- The ground feature $\mathit{result}(\mathit{Canucks}, \mathit{1})$ denotes the result of the Canucks in match 1. This is a target instance.
- $\mathit{PlaysFor}(P, T, M)$ is a predicate that is true if player P plays for team T in match M.
- $\mathit{goals}(P, T, M)$ is the number of goals scored by a player in a match.
- +/-(P, T, M) is the +/- score of a player in a match. This is a common measure of the player's performance; for precise definition see [18].

C. Aggregation

Given a feature f, an aggregate function agg applies to one of the argument variables of f. We use the subscript notation agg_X to indicate that variable X is the object of aggregation [15]. The result is a feature with one less argument. Examples include the following.

- $\mathit{goals}(T, M) \equiv \sum_P \mathit{goals}(P, T, M)$ is the number of goals scored by a team in a match.
- $\mathit{past\_goals}(P) \equiv \sum_{M \text{ in past season}} \mathit{goals}(P, T, M))$ denotes the sum of a player's goals in the past season.

D. Relational Data Tables

Relational data can be visualized in terms of the **groundings data table**. The data table has one column for each feature. It has one row for each simultaneous grounding of all functor features where the instances of the nonclass features are in the neighborhood of the instance of the target feature. Thus if the target functor feature is instantiated with ground instance t, the data table contains a row listing the attributes of each neighbor n of t. Tables I and II (propositionalized) show an example of groundings data tables. As the examples illustrate, aggregation increases the number of features (columns) and decreases the number of data points (rows). Table I is constructed as follows. A row in this table corresponds to a match, one of the teams involved in the match, and one player who played for that team in the match. Each NHL team dresses exactly 18 skaters per match, so for a given match, the data table contains $2 \times 18 = 36$ rows. The 19 columns represent the result of the match, and the 18 last-season statistics of the player.

TABLE I
GROUNDINGS DATA TABLE FOR NHL.

Instance Weight	result(T,M)	MatchId M	TeamId T	PlayerId P	past_goals(P)	goals(T,P,M)
1/18	Loss	2010020023	Canucks	D. Hamhuis	5	0
1/18	Loss	2010020023	Canucks	D. Sedin	34	0
1/18	Loss	2010020023	Canucks	H.Sedin	32	0
...	...	...	...	#4–#18	...	...
1/18	Win	2010020033	Canucks	D. Hamhuis	5	0
1/18	Win	2010020033	Canucks	C. Ehrhoff	17	0
1/18	Win	2010020033	Canucks	H. Sedin	32	0

TABLE II
AGGREGATE FEATURE DATA TABLE FOR NHL.

result(T,M)	MatchId M	TeamId T	Sum_past_goals(T)	Sum_goals(T,M)
Loss	2010020023	Canucks	252	1
Win	2010020033	Canucks	259	2

IV. SCORE AGGREGATION VS. FEATURE AGGREGATION: STRENGTHS AND WEAKNESSES

We describe carrying out relational classification with aggregate features and scores. We discuss the basic strengths and weaknesses of each approach, which motivate the design of the methods in our experiments.

Classification with aggregated features is conceptually straightforward: aggregation produces a data table with one row per target instance that can be treated like a standard attribute vector table. See Table II for illustration.

Classification with aggregated scores can be visualized in terms of the **groundings data table**, or data table for short; see Table I. For simplicity, we discuss score aggregation for a single relationship, which defines a neighborhood for each grounding of the target feature. Our discussion applies equally to classification scores obtained with different types of neighborhoods. Suppose that we have trained a classifier model $\mathcal{M}$ that returns a classification score for a given target label y and feature vector $\boldsymbol{x}$. We write $\mathit{score}_{\mathcal{M}}(y; \boldsymbol{x})$. We can apply this classifier to each row in the groundings data table to derive a classification score from the features of each neighbor of a given target instance t. Given a list of classification scores, one for each row in which the target instance appears in the data table, we can apply a standard aggregation function to obtain an overall classification score. We also use the noisy-or rule for combining probabilities [6]. For a classifier whose score indicates the probability of a positive classification, such as logistic regression, we treat the aggregate probability as the overall probability of a positive classification for the target instance, as in [19]. For other classifiers such as SVM, we classify an instance as above if the aggregate score is above the classifier's standard threshold (e.g., above 0 in the case of SVMs). Table III summarizes the aggregation functions shared by feature and score aggregation, as well as the aggregate functions specific to each method.

TABLE III
AGGREGATE FUNCTIONS USED

	Feature Aggregation	**Score Aggregation**
Shared Functions	Average, Maximum, Minimum, Midrange, Geometric Mean	
Specific Functions	Sum, Standard Deviation, Degree	Noisy–Or

A. Feature Aggregation: Strengths and Weaknesses

Feature aggregation is a very common approach to relational classification and has been much discussed [19], [21], [3]. We review the main points that are relevant for our study. Feature aggregation is conceptually attractive in that it reduces relational classification to non-relational classification with a single feature vector per target instance. Reducing the size of the data table also speeds up learning, as our experiments show.

The obvious drawback of feature aggregation is that it loses information about the distribution of features. Consider the problem of predicting the box office receipts of a movie from user ratings. As an extreme thought experiment, suppose that all movies in our dataset receive the same average user rating, but that the variance of their ratings differs. Then by using the average rating as the aggregate feature, all predictive information is lost. In our experiments, we address the potential loss of information in two ways. (1) We add a set of aggregate features to the data, rather than fixing a single aggregate operator in advance. In this way, learning can decide which aggregation operation is the most informative. (2) In addition to the mean of a feature, we add its standard deviation as an aggregate feature. Thus learning is provided with information about the first two moments of the feature distribution rather than only the first. To our knowledge, adding standard deviation as an aggregate feature is novel.

Further known problems with aggregate features arise in the presence of *degree disparity*. Degree disparity refers to the fact that the degree, i.e., the size of relational neighborhoods, may vary widely for different target instances. For example, the number of ratings received by a movie may vary from zero to thousands. One problem with using aggregate features with degree disparity is that the data lose the information about the size of the relational neighborhood. Also, the values of many aggregate functions correlate with degree [21], i.e., they tend to increase with the degree. So the aggregate feature conflates information about the degree with information about the original feature. To address this conflation, we add the relational degree of each target instance as an aggregate feature in our experiments. Adding a degree feature is recommended by [21].

B. Score Aggregation: Strengths and Weaknesses

The main strength of score aggregation is that it retains the full distributional information in the data. A computational drawback is reduced speed because of the larger data table size. Another issue is that it applies a single fixed aggregate function to scores, rather than exploring an aggregate function space. A problem that figured prominently in our experiments, but seems not to have been previously discussed, is that score aggregation is also affected by degree disparity. As an extreme thought experiment, suppose that our dataset contains ratings for 100 movies, 99 of which have received only 1 rating, and 1 of which has received 99 ratings. So the groundings data table contains 99 rows for the one movie, and 99 rows for the other 99 movies. Hence applying a standard machine learning algorithm to the groundings data table "as is" overweights the movie with the large degree. To address degree disparity for score aggregation, we reweight the rows in the data table by dividing by the degree of each row's target instance. In our thought experiment, the rows for the single large-degree movie would be reweighted by 1/99, and the rows for the others would retain unit weight. Table IV summarizes the main points of our discussion. Our empirical evaluation examines these basic aspects of feature and score aggregation and the effectiveness of solutions to address them.

TABLE IV
CONCEPTUAL COMPARISON OF FEATURE VS. SCORE AGGREGATION

Aggregation	Pros	Cons	Remedy
Features	Fast learning Utilizes multiple aggregate functions	Loses distribution information Ignores Degree Disparity Increases dimensionality	Add standard deviation feature Add degree feature
Scores	Full Distribution Information	Uses a single fixed aggregator Degree disparity: overweights instances with many links	 Reweight Instances

V. DATASETS

We carry out experiments on six data tables derived from five real-world databases. All our datasets are available on-line. The datasets vary in size and degree disparity. For each data table, we obtain two versions: the groundings data table (cf. Table I) and the feature aggregation table (cf. Table II). So each classifier is applied to twelve datasets. Two standard databases have been previously used in studies of relational learning, IMDb and Financial. We introduce four new datasets from sports databases: the National Hockey League (NHL), UK Premier League (PLG), and National Basketball Association (NBA). Sports datasets are challenging for learning because of their complexity. At the same time, they are engaging to many users. They are suitable for studying the effects of aggregation because aggregate functions such as average, sum, etc. most naturally apply to continuous features, and the sports datasets contain mainly continuous features, namely counts of players' actions. We describe the details of the datasets. Then we summarize the properties of the datasets that are relevant to feature and score aggregation, as discussed in Section IV, such as degree disparity and the variance of feature distributions.

post

A. Dataset Details

For each sports dataset, the target feature is $result(Team, Match)$. A positive classification means that the team is predicted to win the match. The target features for IMDb and Financial are given below.

1) IMDb: The hierarchical relational structure of the IMDb dataset is as follows: each *director* has their own attributes and has directed 1 or more *movies*. Each *movie* has been reviewed and rated by 1 or more *users*, who also have their own attributes. During feature aggregation, the *user* attributes and ratings are aggregated. The target feature for the IMDb dataset is $highBoxOffice(Movie, Director)$, where the positive class denotes the movie had a box office receipt of $\$10,000,000$ USD or greater.

need source, ask Zhensong

2) Financial: The financial dataset has a hierarchical relation structure with *district* at the top level, followed by *accounts* within the *district*, and finally all the *transactions* associated with a particular *account*. During feature aggregation, the attributes of the transaction are aggregated. The financial dataset also suffers from degree disparity, as shown in Table VII. For the financial dataset, the target feature is $hasLoan(Account, District)$, where a positive classification means there is a loan associated with the account. This dataset is a modified version of the financial dataset from the discovery challenge at PKDD'99 following the modification from [22].

3) PLG data tables: We used Opta data [23], released by Manchester City. It lists all the ball actions within each game by each player, for the 2011-2012 season. Number of goals, passes, fouls, tackles, saves and blocks by a player in a match are examples of the information associated with each player. For each player in a match, our data set contains 199 player features that are counts for 199 actions.

4) NBA data tables: NBA data was obtained manually from http://www.nba.com/. Box scores containing match summary statistics for each player were used to create the data table. For each *player* on a *team* in a *match*, there are 19 continuous player statistics recorded, such as number of free throw attempts and total number of player rebounds. These player statistics are aggregated over each $(team, match)$ instance during feature aggregation. No team attributes or previous player statistics are used in the data table.

5) NHL data tables: We used the Selenium webcrawler [24] to download player game statistics (Box Scores) from http://www.nhl.com/gamecenter/ for the seasons 2009–2013. The box scores summarize player statistics for each match, a total of 13 continuous-valued features. We refer to these as **match statistics**. We only consider skaters in our model and remove goalies, as the number of goalies in the NHL is significantly less than the number of skaters, and different statistics are recorded for goalies than skaters. The match features include goals, assists, plus-minus, penalty minutes, and total time on ice. For each player, we sum his match statistics over all NHL games in the previous season to obtain a total of 13 statistic totals for the previous season. In addition, we add 5 other season statistics: number of games played, game winning goals, powerplay goals, shorthanded goals, and shot percentage. We refer to the resulting 18 features as **last-season features**. From this database we prepared the following two groundings data tables, depending on whether we used last-season features only or all features.

Season Contains the last season features only.

S+Match Contains the last season features and the match statistics.

B. Feature Distributions in Datasets

We examine summary statistics for our datasets pertinent to the discussion of aggregation in Section IV. Table V shows the strong effect that aggregation has on the data table dimensions. It reduces the number of rows (data points), in the case of IMDb by a factor of around 300. Aggregation increases the number of columns (features), in the case of the PLG soccer data, by a factor of almost 7.

TABLE V
DATA TABLE DIMENSIONS

Dataset	Rows	Aggregated Rows	Columns	Aggregated Columns
IMDb	909,377	2,910	64	118
Financial	348,095	1,364	130	280
NHL - S + Match	138,852	7,714	35	221
NHL - Season	138,852	7,714	22	130
PLG	7,933	580	203	1,397
NBA	767	60	23	137

Table VI illustrates how aggregation decreases the variance of features. We selected one attribute for each dataset, and compared its variance on the original groundings data table to its variance after applying the average μ aggregator. A reduction in variance can be seen as a reduction in information content (as Principal Component Analysis seeks to maximize variance of projections).

TABLE VI
FEATURE VARIANCE VS. AVERAGE FEATURE VARIANCE

Dataset	Attribute A	Variance A	Variance μA	Reduction Ratio
IMDb	Age(User)	135.97	19.37	7.02
Financial	Amount(Transaction)	112,257,686.00	16,838,158.97	6.67
NHL - S + Match	GamePlusMinus(Player)	1.16	0.33	3.50
NHL - Season	LastSeasonPlusMinus(Player)	112.80	24.69	4.57
PLG	Goals(Player)	0.13	0.01	13.67
NBA	PlusMinus(Player)	118.02	38.03	3.10

TABLE VII
DEGREE DISPARITY

Dataset	Relationship	Average	Standard Deviation	Max	Min
IMDb	Ratings/Movie	313.91	411.92	3,427.00	1.00
Financial	Transaction/Account	255.68	134.09	675.00	9.00
NHL	Players/Team,Match	18.00	0.00	18.00	18.00
PLG	Players/Team,Match	13.64	0.63	14.00	11.00
NBA	Players/Team,Match	12.71	0.45	13.00	12.00

TABLE VIII
ATTRIBUTE INFORMATION GAIN

Dataset	Best Attribute	Information Gain	Best Aggregated Attribute	Information Gain
IMDb	Director_AvgRevenue	0.10154	Director_AvgRevenue	0.51870
Financial	Remittance(Transaction)	0.27386	AVG(Remittance(Transaction))	0.35431
NHL - S + Match	GamePlusMinus(Player)	0.11712	AVG(GamePlusMinus(Player))	0.55938
NHL - Season	LastSeasonPlusMinus(Player)	0.00138	AVG(LastSeasonPlusMinus(Player))	0.00715
PLG	Goals(Player)	0.03166	AVG(Goals(Player))	0.57900
NBA	PlusMinus(Player)	0.17400	AVG(PlusMinus(Player))	0.87400

In general the sports datasets have small to no degree disparity. This is because the number of players in a team in a match varies very little. In ice hockey, each NHL team dresses exactly 18 skaters per match. For example, in basketball, all teams dress either 12 or 13 players for a match. The soccer PLG dataset exhibits some small degree disparity, as a maximum of three substitutions per team are allowed during PLG matches. The IMDb and Financial datasets exhibit considerable degree disparity, as shown in Table VII. The number of ratings for movies varies greatly depending on the popularity of the movie. For financial transactions, different accounts may be involved in transactions to highly varying degrees of frequency.

TABLE X
PLG TARGET MATCH

	Score Aggregator			
	Average	Maximum	Minimum	Noisy-Or
SVM - Linear	*80.15%*	**54.78%**	51.10%	54.78%
SVM - Quadratic	76.84%	53.68%	51.47%	**53.68%**
SVM - Sigmoid	53.68%	50.00%	50.00%	50.00%
SVM - Gaussian	68.75%	50.37%	**55.88%**	50.37%
LR - Prob	**81.62%**	52.57%	52.21%	52.57%
LR - Log-Prob	77.21%	52.57%	52.21%	N/A

VI. EVALUATION

A. Methods and Comparison Metrics

As base classifiers, we use SVM and logistic ridge regression. For SVM, we report results for 4 kernels. Parameters of the classifiers were set by a grid search that evaluated a parameter setting by cross-validation on the training set. We report the results for the best parameter setting found by the grid search.

For feature aggregation, we report results for pairs (Classifier x Dataset). All aggregations of all features are used to train the classifier. For score aggregation, we report results for triples (Classifer x Dataset x aggregate operator). Table III summarizes the aggregate operators used.

Our basic metrics are **classification accuracy** (percentage of correctly classified target instances) and **F1-measure**, the harmonic mean of precision and recall [25]. We train the classifiers on a training set of target instances and test on the remaining target instances. All datasets use an 80 : 20 split for the training and test sets.

B. Results

[logistic regression]: All methods are quite good. Aggregate features do well with degree disparity. Small difference on PLG for aggregation. Aggregate over goals gives very good results. NHL season has very poor features. Degree disparity makes for bigger difference. Instance weighting doesn't help a lot, only on IMDb.

We first compare different methods for score aggregation, then for feature aggregation, finally both together.

1) Score Aggregation: Table **??** shows the accuracies for each combination of (trained base classifier, score aggregator), for both feature sets for the NHL data. The F-measures showed the same trends. Logistic Ridge Regression achieved high results with both the average and maximum operator, for both probabilities and log probabilities. SVMs performed poorly overall, with the exceptions of using the linear or quadratic kernel with the average score aggregator. Logistic regression is superior to SVMs when used with score aggregation on these datasets.

Table X shows the results for the premier league data. Here the average operator performs clearly the best, especially with logistic regression probabilities. For other operators, SVMs do slightly better than logistic regression, but poorly overall.

In sum, logistic regression appears more robust on our datasets than SVM, and the average aggregator appears more robust than other aggregators.

C. Feature Aggregation

Table XII presents the results for feature aggregation methods, on all three datasets. Examination of regression weights showed minimum and sum of match plus-minus to be a strong predictor for NHL results. Logistic regression with feature aggregation on the NHL All Features dataset showed a 25.67% improvement over all SVM models.

TABLE XII
FEATURE AGGREGATION RESULTS

	Data Table		
Classifier	NHL Last Season	NHL All Features	PLG Match Features
SVM - Linear	*53.43%*	60.40%	**90.44%**
SVM - Quadratic	52.66%	61.74%	89.71%
SVM - Sigmoid	50.00%	50.00%	50.00%
SVM - Gaussian	50.00%	52.52%	54.78%
Logistic Regression	**53.94%**	**87.41%**	**90.44%**

TABLE XIII
COMPARISON OF CLASSIFICATION ACCURACY AND F-MEASURE FOR FEATURE AND SCORE AGGREGATION

	Feature Agg.		Score Aggregation	
Dataset	Classification Accuracy	F-Measure	Classification Accuracy	F-Measure
NHL Last Season Stats	53.94%	0.5368	54.53%	0.5593
NHL All Features	87.41%	0.8753	87.26%	0.8709
PLG Match Features	90.44%	0.9044	81.62%	0.8120

D. Score Aggregation vs. Feature Aggregation

Table XIII compares the best feature aggregation method with the best score aggregation method. On the NHL datasets, feature Aggregation performs at least as well as Score Aggregation, although the difference is not statistically significant. Feature aggregation clearly performs better on the Premier League datasets.

In sum, comparing score aggregators, the average suggested by the random selection semantics appears to be the most robust aggregator, providing competitive performance in a variety of settings. Minimum and noisy-or operators performed poorly on all datasets. Comparing base classifiers, logistic regression appears more reliable than SVMs, which can perform very poorly with the wrong kernel.

A hybrid approach that combines score aggregation with feature aggregation could address the weaknesses of both approaches. For example good features could be found learning a model based on feature aggregation. Adding good aggregation features to non-aggregated features (e.g., player statistics) in score aggregation could then improve classification accuracy. Conversely, a score-aggregation classifier can be used as a strong baseline for pruning noninformative aggregate features, to reduce the expensive search through the aggregation space.

VII. CONCLUSION

We considered link-based classification with continuous features of linked entities, on real-world sports data sets. Two basic approaches are aggregating features vs. aggregating

TABLE IX
SCORE AGGREGATION ACCURACIES

Aggregator	IMDb	IMDb-W	Financial	Financial-W	NHL - S + Match	NHL - Season	PLG	NBA
Average	78.52%	81.44%	69.12%	73.16%	**87.29%**	**55.25%**	90.52%	**100.00%**
Maximum	71.65%	**82.13%**	63.97%	68.75%	52.01%	50.65%	53.45%	50.00%
Minimum	**81.79%**	79.73%	53.68%	51.10%	50.45%	50.00%	51.72%	58.33%
Midrange	78.52%	80.93%	**72.79%**	**73.53%**	85.34%	52.79%	**93.10%**	**100.00%**
Geometric Mean	78.69%	81.44%	69.49%	72.43%	85.08%	55.12%	81.03%	**100.00%**
Noisy-Or	71.65%	82.13%	63.97%	64.34%	52.01%	50.65%	53.45%	50.00%

TABLE XI
FEATURE AGGREGATION ACCURACIES

Method	IMDb	Financial	NHL - S + Match	NHL - Season	PLG	NBA
Logistic Regression	**86.05%**	**84.93%**	**88.59%**	54.67%	**95.69%**	**100.00%**
SVM - Linear	81.79%	76.84%	68.03%	**55.12%**	95.69%	**100.00%**
SVM - Quadratic	71.99%	72.06%	61.74%	52.66%	89.71%	91.67%
SVM - Gaussian	82.30%	65.81%	52.52%	50.00%	54.78%	**100.00%**

TABLE XIV
SCORE AGGREGATION VS. FEATURE AGGREGATION WITH LOGISTIC REGRESSION

Dataset →	IMDb		Financial		NHL - S + Match		NHL - Season		PLG		NBA	
Method ↓	Accuracy	F1-Measure	Accuracy	F1-Measure	Accuracy	F1-Measure	Accuracy	F1-Measure	Accuracy	F1-Measure	Accuracy	F1-Measure
Feature Agg.	**86.05%**	**0.86**	84.93%	0.84	88.59%	**0.89**	54.67%	0.49	95.69%	**0.96**	**100.00%**	**1.00**
Feature Agg. + σ	85.57%	**0.86**	**87.50%**	**0.87**	**88.91%**	**0.89**	54.47%	**0.50**	**96.55%**	**0.96**	**100.00%**	**1.00**
Score Agg. - Best	81.79% (Min)	0.84	72.79% (Mid)	0.77	87.29% (Avg)	0.87	**55.25% (Avg)**	0.46	93.10% (Mid)	0.93	**100.00% (Avg)**	**1.00**
Score Agg. - μ	78.52%	0.82	69.12%	0.74	87.29%	0.87	**55.25%**	0.46	90.52%	0.90	**100.00%**	**1.00**
Weighted S.Agg. - Best	82.13% (Max)	0.82	73.53% (Mid)	0.76	87.35% (Avg)	0.87	55.12% (Avg)	0.48	92.24% (Mid)	0.92	**100.00% (Avg)**	**1.00**
Weighted S. Agg. - μ	81.44%	0.81	73.16%	0.74	87.35%	0.87	55.12%	0.48	90.52%	0.90	**100.00%**	**1.00**

TABLE XV
LEARNING TIME IN SECONDS

	IMDb		Financial		NHL - S + Match		NHL - Season		PLG		NBA	
	LR	SVM	LR	SVM	LR	SVM	LR	SVM	LR	SVM	LR	SVM
Feature Aggregation	0.097	57.502	0.122	53.584	0.407	75.242	0.106	71.150	1.666	0.488	0.006	0.003
Feature Aggregation + σ	0.083	38.074	0.894	29.219	0.523	52.109	0.126	59.329	3.288	0.532	0.034	0.003
Score Aggregation	14.074		6.314		0.567		0.229		0.430		0.004	4.834
Weighted Score Aggregation	13.877		5.100		0.627		0.215		0.368		0.006	

classifier scores. Empirical results indicate that averaging is the most robust operation for classifier scores. Averaging is consistent with the classic random selection semantics for probabilistic logic. A promising avenue for future work is to combine aggregate features with score aggregation.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] J. Y. Halpern, "An analysis of first-order logics of probability," *Artificial Intelligence*, vol. 46, no. 3, pp. 311–350, 1990.

[2] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*. Cambridge, MA, USA: MIT Press, 1990.

[3] H. Chen, H. Liu, J. Han, and X. Yin, "Exploring optimization of semantic relationship graph for multi-relational Bayesian classification," *Decision Support Systems*, vol. 48, no. 1, pp. 112–121, 2009.

[4] B. Bina, O. Schulte, B. Crawford, Z. Qian, and Y. Xiong, "Simple decision forests for multi-relational classification," *Decision Support Systems*, vol. 54, no. 0, p. 1269C1279, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923612003491

[5] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[6] K. Kersting and L. de Raedt, "Bayesian logic programming: Theory and tool," in *Introduction to Statistical Relational Learning*. MIT Press, 2007, ch. 10, pp. 291–318.

[7] S. Kramer, N. Lavrac, and P. Flach, "Propositionalization approaches to relational data mining," in *Relational Data Mining*. Springer, 2000, pp. 262–286.

[8] M.-A. Krogel and S. Wrobel, "Feature selection for propositionalization," in *DS '02: Proceedings of the 5th International Conference on Discovery Science*. London, UK: Springer-Verlag, 2002, pp. 430–434.

[9] A. Van Assche, C. Vens, H. Blockeel, and S. Dvzeroski, "First order random forests: Learning relational classifiers with complex aggregates," *Machine Learning*, vol. 64, no. 1, pp. 149–182, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.7573&rep=rep1&type=pdf

[10] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*. MIT Press, 2007, ch. 4, pp. 93–127.

[11] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probabilistic models for relational data," in *UAI*. Morgan Kaufmann Publishers Inc., 2002, pp. 485–492.

[12] P. Domingos and D. Lowd, *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.

[13] Q. Lu and L. Getoor, "Link-based classification," in *ICML*. AAAI Press, 2003, pp. 496–503.

[14] O. Kuzelka and F. Zelezný, "Block-wise construction of tree-like relational features with monotone reducibility and redundancy," *Machine*

Learning, vol. 83, no. 2, pp. 163–192, 2011.
[15] A. Popescul and L. Ungar, "Feature generation and selection in multi-relational learning," in *Introduction to Statistical Relational Learning*. MIT Press, 2007, ch. 16, pp. 453–476.
[16] J. N. David Jensen, "Linkage and autocorrelation cause feature selection bias in relational learning (2002)," in *ICML*, 2002.
[17] D. Jensen and J. Neville, "Data mining in social networks," in *In National Academy of Sciences workshop on Dynamic Social Network Modeling and Analysis*, 2002.
[18] R. P. Schumaker, O. K. Solieman, and H. Chen, *Research in Sports Statistics*, ser. Integrated Series in Information Systems. Springer US, 2010, vol. 26. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-6730-5_4
[19] J. Neville, D. Jensen, B. Gallagher, and R. Fairgrieve, "Simple estimators for relational bayesian classifiers," in *ICDM*. IEEE Computer Society, 2003, pp. 609–612.
[20] D. Poole, "First-order probabilistic inference," in *IJCAI*, 2003, pp. 985–991.
[21] D. D. Jensen, J. Neville, and M. Hay, "Avoiding bias when aggregating relational data with degree disparity," in *Proceedings of the Twentieth International Conference on Machine Learning*, T. Fawcett and N. Mishra, Eds. Washington, DC: AAAI Press, August 2003, pp. 274–281.
[22] X. Yin, J. Han, J. Yang, and P. S. Yu, "Crossmine: Efficient classification across multiple database relations," in *ICDE*. IEEE Computer Society, 2004, pp. 399–410.
[23] "Mcfc analytics," uRL = http://www.mcfc.co.uk/Home/The%20Club/MCFC%20Analytics.
[24] "Selenium," uRL = http://www.seleniumhq.org/.
[25] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.
[26] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. MIT Press, 2007.