Predicting Tennis Match Outcomes Through Classification

Shuyang Fang
CS074 - Dartmouth College

Introduction

The governing body of men's professional tennis is the Association of Tennis Professionals or ATP for short. It organizes the year-to-year tournaments and other events that male professionals participate in. It is perhaps most notable for publishing a weekly ranking of players, the ATP rankings. A player's ranking is based on the total points he accrued in a specific set of tournaments within the last 52 weeks, more points equates to a higher ranking (standard ranking system, where #1 is the top rank, #2 is second, and so on).

The question of who wins a given tennis match is of both recreational (fans) and professional (Vegas) interest. Conventional wisdom suggests that given two tennis players, the one with a higher ranking will win more often than not. This may be true; however, tennis is a game that revolves around matchups. There are a multitude of factors that play into a match: first serve percentage, win loss record, break point conversion percentage just to name a few. It goes without saying that the combination of these factors with player ranking serves as a far stronger predictor of match outcome than ranking alone.

The goal of this project is to instruct the computer to predict the winner of a tennis match with greater accuracy than both random guessing and ranking comparison.

State – of – the – Art

There is surprisingly little work/research on this topic. Machine learning methods have been applied to more popular sports such as American football and baseball to predict the winner, but I could not find any similar research on tennis.[3][5]

Data

I used match results data from

http://www.tennis-data.co.uk/alldata.php

which contains all matches and results for ATP seasons 2000-2012. For player info I used statistics from

http://statracket.net/?view=stats/stats.php

which is a dataset of ATP matchfacts for the top 200 or so players from 2004 to 2010. Using these two datasets in conjunction, I compiled my training and test data. Training data consisted of matches from 2005 to 2009, using player

info from the previous years, so 2004 to 2008 (one to one, so 2004 player info was used for 2005 matches). Test data consisted of matches from 2010 to 2011, using player info from the previous years, so 2009 to 2010.

There are 15 features for each match, measured as real valued differences between the two players in the feature category. They are in order:

1. ranking difference
2. win % difference
3. tiebreak win % difference
4. matches played difference
5. aces/match difference
6. double faults/match difference
7. 1^{st} serve % difference
8. % of 1^{st} serve points won difference
9. % of 2^{nd} serve points won difference
10. % of service games won difference
11. % of break points saved difference
12. % of return on 1^{st} serve points won difference
13. % of return on 2^{nd} serve points won difference
14. % of break points won difference
15. % of return games won difference

Initially, I created a single dataset comprised of two sub-datasets. One subset contained all matches where complete player info of both players was available, 12,313 matches for the training, 4650 test. The other sub-dataset was itself a subset of the first subset. It only contains matches where the ranking difference between the two players was less than 10. For all matches, I measured the feature values as the difference between the winner's statistic in that feature and the loser's statistic (winner – loser). I had to take the absolute value of the ranking difference, otherwise, Adaboost simply used the ranking difference to achieve perfect classification. I classified a match as 1 if the higher ranked player won and -1 if the lower ranked player won. From here on out, this dataset will be called *tennisdata.*

However, some last-minute advice revealed that how I constructed this first dataset could have introduced significant bias, since all feature values and even the classification are measured with respect to the match winner . To test this theory and for more complete results, I created a second data set (with the same sub-dataset divisions as above). For each match, I randomly chose a player as the first player and assigned the other player to the second player. Classification is then 1 if the first player won, -1 if he did not. Feature values are measured as the difference between the first player's data and the second's (first – second). I also artificially increased the size of the dataset by appending the case where the current player configuration is switched, i.e. if the player assigned to the second player was assigned to the first and vice versa. From here onwards, this dataset will be called *unbiased_tennisdata.*

Methods

I implemented three different classification methods, ranking, logistic regression, and Adaboost. The ranking and logistic regression methods are used to serve as benchmarks.

The ranking method simply always chooses the higher ranked player as the winner.

The second is the logistic regression algorithm we implemented in class, simply modified to accommodate the new dataset.

Adaboost is the main algorithm I focus on. It aims to create a strong classifier $f(x)$ as a linear combination of T weak classifiers $h_t(x)$. The pseudocode can be presented as such:

Initialize a distribution $D_1(x) = \frac{1}{m}$ where m is the number of examples, so initially weigh all examples equally.
Then for $i = 1 \ldots T$ where T is the desired number of weak classifiers

$h_i(x) =$ the classifier that minimizes the classification error w.r.t. the current distribution $D_i(x)$

Let α_i be the coefficient of the $h_i(x), \alpha_i = \frac{1}{2} ln \frac{1-err_i}{err_i}$ where err_i is the classification error of the current classifier

Update the distribution $D_i(x)$ using the previous distribution and the current weak classifier such that greater weight is assigned to incorrectly classified examples. Normalize the distribution.

Finally

$$f(x) = \sum_{i=1}^{T} \alpha_i h_i(x)$$

My weak classifiers are simply binary decision stumps with an associated feature. If the example is above/below some threshold value with respect to that feature then it is classified appropriately. The algorithm learns both the optimal feature and threshold for each weak classifier such that the weak classifier minimizes classification error within that iteration

Results/Dicussion

The following are the results on the *tennisdata* data set (all values measured respective to winner)

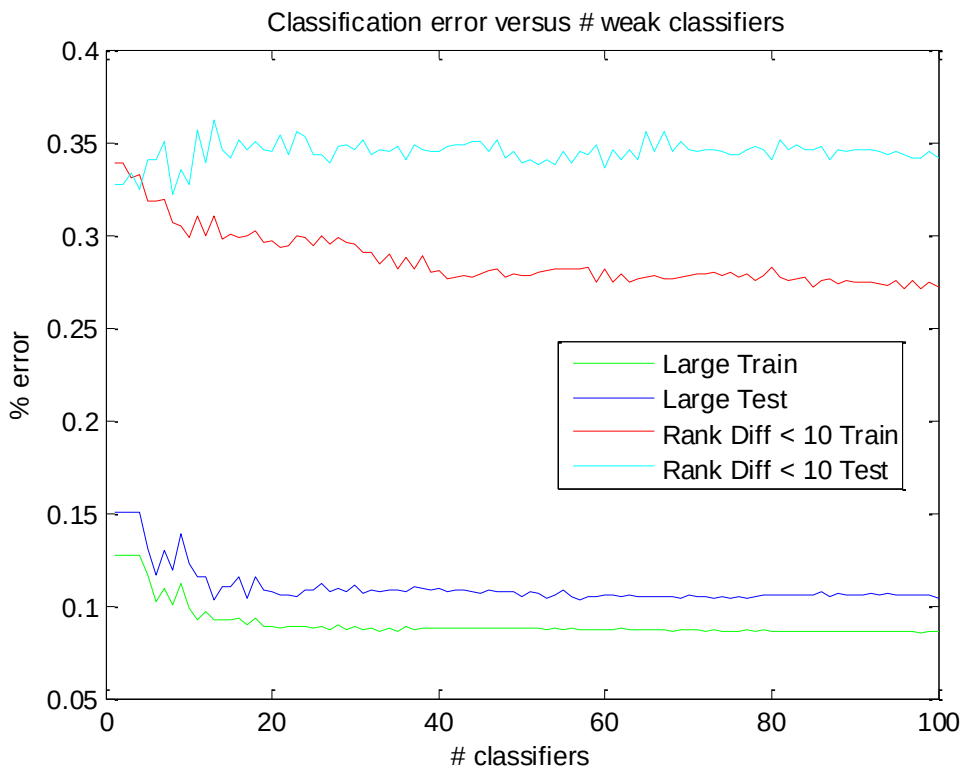

Figure 1: *tennisdata* dataset. Error is simply calculated as percentage of examples misclassified. 40 or so weak classifiers seems to be enough for convergence.

It is unsurprising that the classification error is so low on the large training/test sets, even when only one weak classifier is used. When matches where one player far outclasses the other are considered, as in the case when the ranking difference between two players is huge, the feature values are also large. Since the large training/test sets contain many such unequal matchups, a threshold/feature pair that immediately leads to low classification error is expected. Subsequent weak classifiers do little to lower the classification error; however, the performance of the strong classifier on the test set indicates that it is properly constructed.

When ranking difference between two players is small, the difference between the two players across the board is probably small as well. A feature that may have been important when one player far outclasses the other may now be a poor classifier. In fact, as the data suggest, it may be very difficult to achieve low classification error in this case. An initial weak classifier achieves ~33% classification error on both the small training/test sets. Although subsequent weak classifiers do lower the training error, they actually lead to overfitting, suggesting that when the ranking difference is small and consequently, feature values are also small, the approach is invalid. The other implication is that when players are close in level, the features this project considers are no longer predictive. Indeed, one can imagine that players' mental games are the driving deciders of match outcomes where players are close in skill. A recent example of this phenomenon is the tennis rivalry between Rafael Nadal and Novak Djokovic. Both are players

of high caliber, often separated by only one in rank. Yet, when they meet on the court, the result is anything but a coin toss. Instead, the outcome seems to be entirely a matter of confidence. Nadal won five straight finals against Djokovic before Djokovic, coming off an emotional Davis Cup victory, won the next seven. The dynamic of the rivalry completely shifted even though both men statistics changed little. Pundits all attributed this to Djokovic's "confidence", a purely subjective attribute.

For a quantitative analysis of this phenomenon, let us look at the features that the first 10 weak classifiers consider.

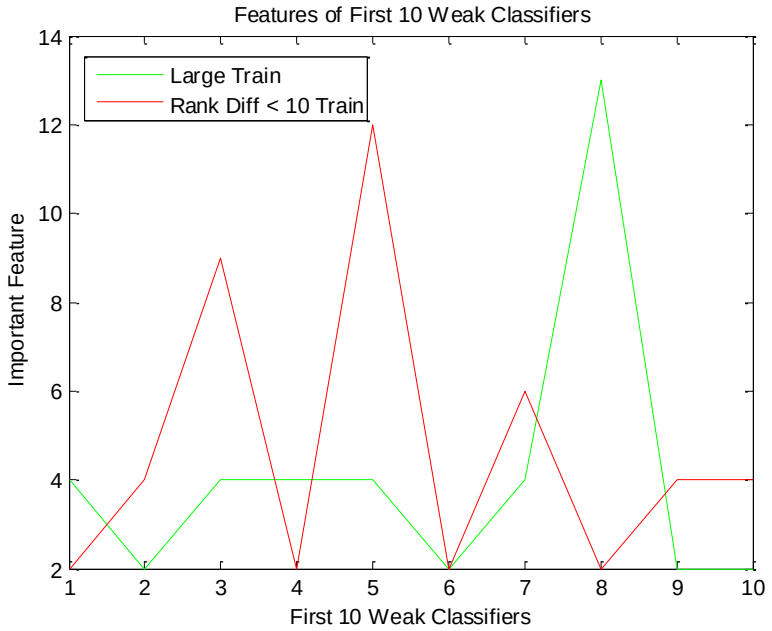

Figure 2: Features associated with the first 10 weak classifier, when 100 weak classifiers are considered. I.e., the initial features that minimize classification error. The Y-axis is simple the # of the feature in order. See the Data section

For both the large training set and the small (rank diff < 10) training sets, the most common feature considered by the first 10 weak classifiers are 4 and 2, differences in matches played and match win % respectively. This is unsurprising. They are good indicators of not only quality of play but also mental strength. The more a player wins, the further he moves through a tournament and thus the more matches he plays. Later rounds of tournament are thus also against strong players and require the mental game to compliment the physical. Thus, these two features are the best at encapsulating the true difference between two players. These features seem closely related; however a player could choose to play the minimum required matches and boast a high winning % or play a large amount of matches and have a low winning %, so they are not necessarily the same. Surprisingly, the ranking difference is not associated with any of the first 10 classifiers, although this is probably a result of me taking the absolute value. Removing the sign removes the information associated with it.

For the sake of comparison, let us see the classification error of the three methods.

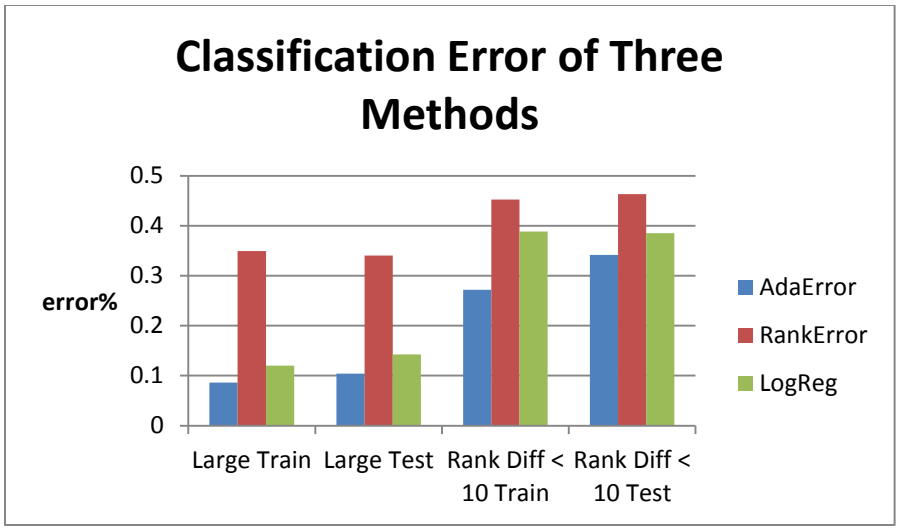

Figure 3: Performance of all three methods measured through classification error % on the data.

All three methods outperform a simple coin flip. All three methods perform better on the large datasets than the ones where the ranking difference is small as expected. Both machine learning algorithms significantly outperform ranking on the large datasets while they still manage to outperform ranking, albeit to a lesser extent on the small datasets. This is to be expected with the nature of the data, the unequal matchups are easier to classify than close ones.

Unfortunately, as explained earlier, bias was introduced into the *tennisdata* dataset by my preprocessing. What follows are the results of the same methods applied to the *unbiased_tennisdata*.

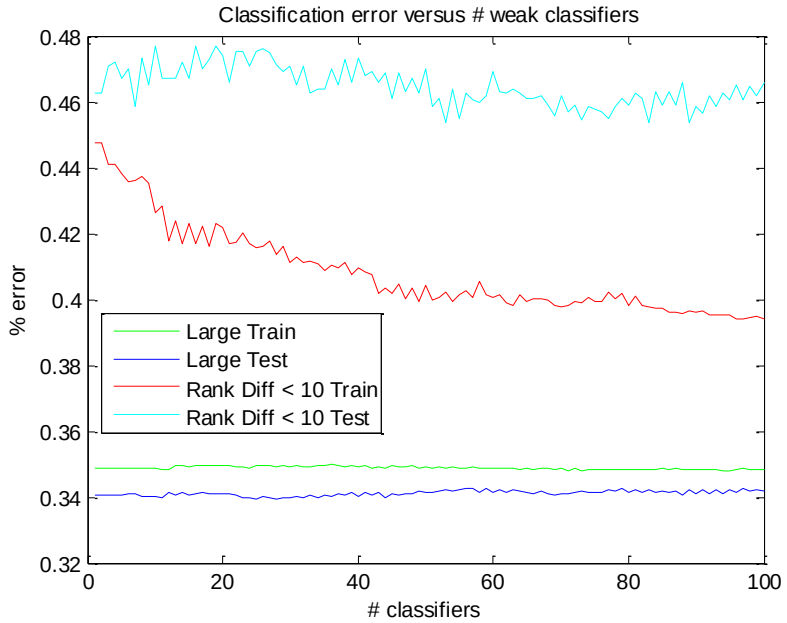

Figure 4: unbiased_tennisdata dataset. Error is simply calculated as percentage of examples misclassified. Convergence is immediate on the large datasets, but 100+ seems to be required for the small ones. Significant overfitting occurs on the small datasets.

Indeed, it seems I did introduce bias. The classification error is significantly higher on this *unbiased_tennisdata* dataset. What is revealing is that the # of weak classifiers seems to matter little on the large sub-datasets, while they do lower the error on the small training sub-dataset but have little impact on the small test sub-dataset. To examine what exactly changed, let us look at the features considered important by the first 10 weak classifiers.

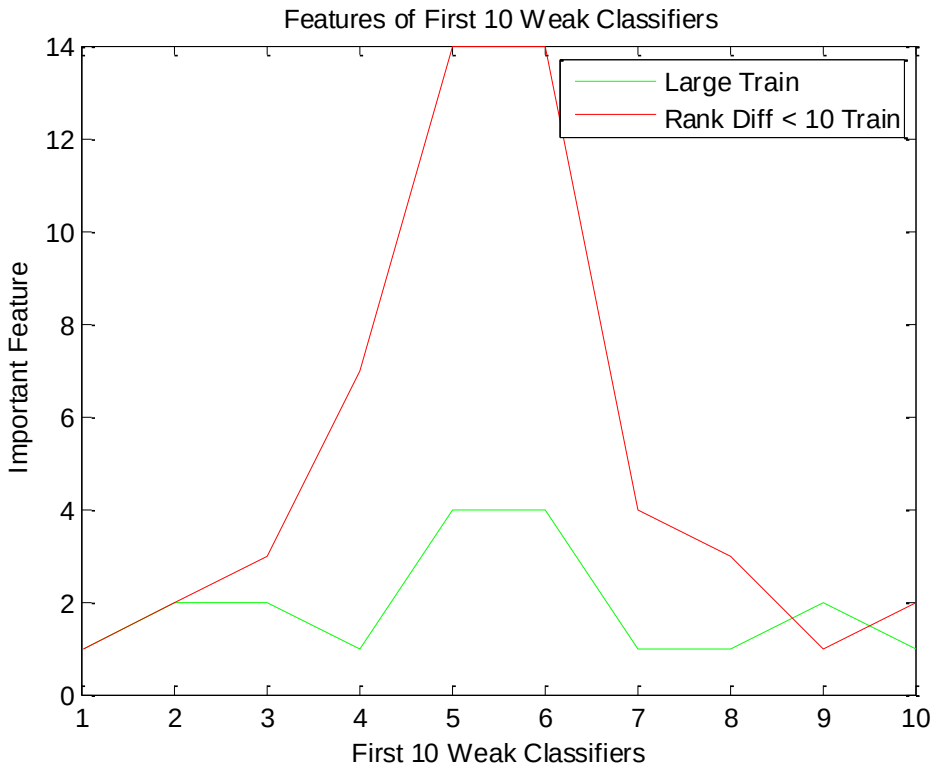

Figure 5: Features associated with the first 10 weak classifier, when 100 weak classifiers are considered. I.e., the initial features that minimize classification error. The Y-axis is simple the # of the feature in order. See the Data section

By removing the bias, we see that the ranking difference is incredibly important for determining match outcome on the large sub-dataset. The ranking is a measure of all other factors and cannot simply be neglected. It accounts for half of the first 10 weak classifiers, and probably accounts for a majority of the 100. This would explain why the classification error on the large sub-dataset is essentially the same as ranking's.

On the sub-dataset that only contains matches where the ranking difference was less than 10, ranking was still important, but it is encouraging to see that features 3 and 14 were also expressed multiple times. These are tiebreak win % and % of break points won respectively. When players are close, more tiebreaks are expected and a single break of serve can decide sets and matches. The computer was able to recognize this, which suggests that although the approach was flawed, the results are not entirely without merit.

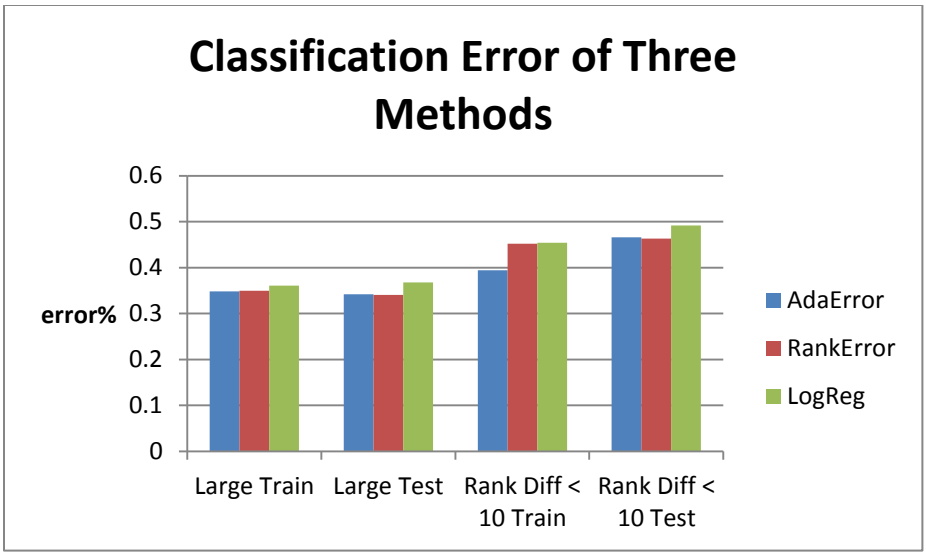

Figure 6: Performance of all three methods measured through classification error % on the data.

On this unbiased dataset, all three methods were essentially equal. It is consoling that both machine learning methods perform better than chance, and even though they perform about the same as ranking, that is a small accomplishment. Tennis ranking does hold merit, it is representative of a player's accomplishments/level. Adaboost's performance on the small training subset is a small consolation. Even though its overall performance is not spectacular, it was able to capture deem features that we would perceive as important to be important.

Conclusion

Assuming *tennisdata* was truly biased (strong assumption), it seems that it is difficult to teach the machine to do better than a simple prediction by ranking for tennis, at least with the data I compiled. Notably, I used player data from the end of the previous year to train/test the algorithm. The matches themselves occurred through the timespan of the following year. Sports are mercurial from year to year, and I was unable to capture these changes. If I had a way to measure player data at the time of the match, the results may have been significantly better. It is encouraging that although both machine learning algorithms underperformed what I would have liked, they did still manage to match ranking, which is a feat in of itself, since ranking encapsulates a player's field of work.

References

[1] https://en.wikipedia.org/wiki/Tennis

[2] https://en.wikipedia.org/wiki/Machine_learning_algorithms

[3] Hamadani Babak. Predicting the outcome of NFL games using machine learning. Project report for Machine Learning, Stanford University.

[4] R.P. Adams, G.E. Dahl, and I. Murray. Incorporating Side Information in Probabilistic Matrix Factorization with Gaussian Processes. Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, 2010.

[5] G. Donaker. Applying Machine Learning to MLB Prediction & Analysis. Project report for CS229 – Stanford Learning 2005.

[6] J. Matas, J. Sochman. AdaBoost. Centre for Machine Perception, Czech Technical University, Prague. http://cmp.felk.cvut.cz
[7] Y. Freund, R.E. Schapire. A Short Introduction to Boosting, AT&T Labs – Research, Shannon Laboratory. Florham Park, NJ, 1999.