Using Machine Learning to predict tennis match outcomes

Alan Wagner, Deepak Narayanan

I. Introduction

In this paper, we propose and describe a tennis match predictor system that attempts to accurately predict the results of tennis matches – in particular, we want to maximize our score in the Association of Tennis Professionals World Tour Draw Challenge, described in greater detail below.

This paper is laid out as follows: first we describe the problem, then we describe the data sets we used and our general problem setup. Next, we describe our solution to the problems of feature, algorithm, model, and parameter selection and present our results. We close with ideas for future work.

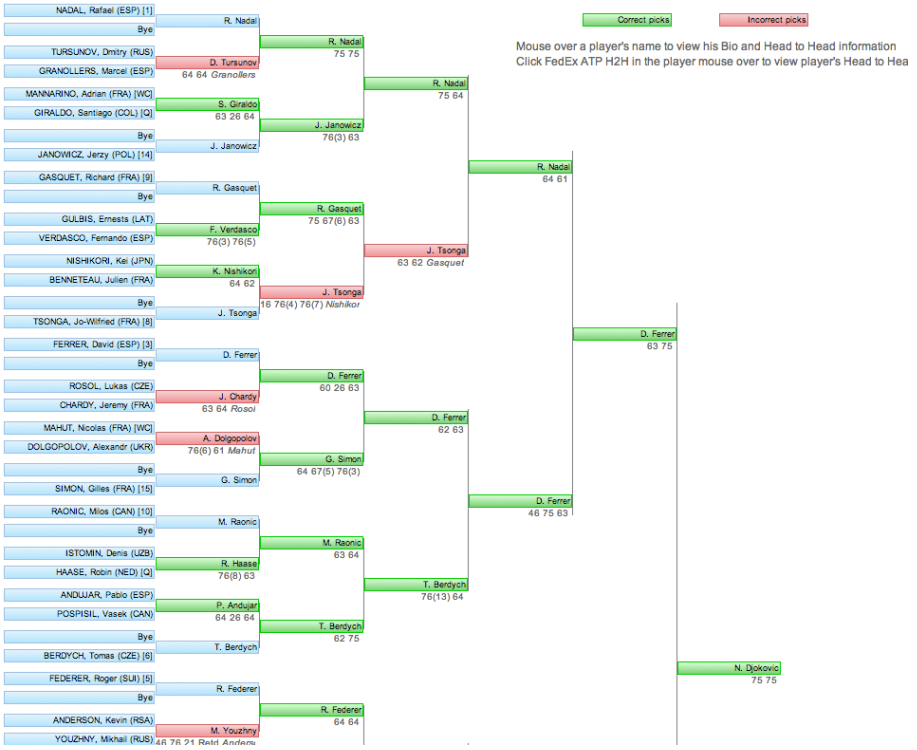

Figure 1: *An example of a bracket, with correct predictions in green and incorrect predictions in red*

II. Background

2.1 Association of Tennis Professionals (ATP) World Tour Draw Challenge

Everyone playing the Challenge receives the tournament draw before any matches are played. Each player must then predict the winner of every match, with the caveat that every chosen winner progresses to the

next round of the tournament and every loser is instantly eliminated. Points are allocated for every correctly predicted winner, with a greater number of points given to correctly predicted winners in later rounds. The goal of the challenge is simple: to score as many points as possible.

2.2 Data

All of our data comes from http://www.matchstat.com/ and http://www.atpworldtour.com/. We scraped around 100,000 pages in total (Match, Player, Event, Ranking and Injury pages). We spent a good deal of time unifying these different data sources (both different types of pages from the same site as well as pages obtained from different sites) – naming wasn't uniform on both sites, which meant we had to write a few scripts that converted everything to a common naming scheme. All the raw and unparsed HTML files were then converted into a much more well schematized format that we inserted into a SQLite database. This allowed us to iterate on the machine learning part of the project much more quickly.

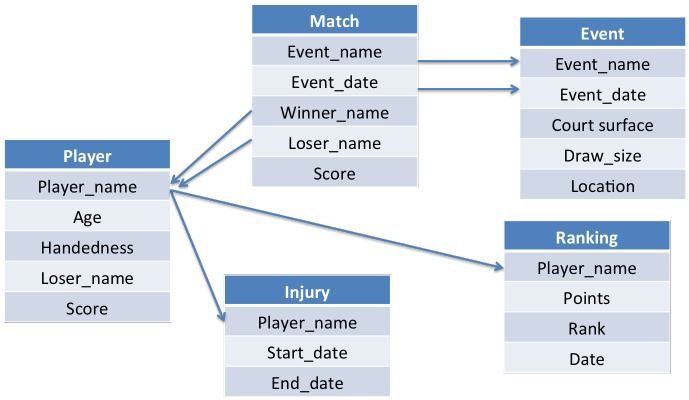

Figure 2: *Schema used for processed data*

III. Machine Learning

3.1 Problem Setup

We considered a number of discriminative and generative approaches, before finally settling on using Support Vector Machines.

The main reasons for choosing SVMs are:

1. We wanted to use an algorithm that is efficient and allows us to iterate as quickly as possible – we did not want to spend a couple of hours waiting for a model to be learnt.

2. SVMs are extremely powerful – they easily allow for complex transformations (using kernels) that are possible without actually having to transform the data.

3. They have several tunable parameters.

If a match is played between player1 and player2, then we formulate our problem as classifying player1 as a winner or a loser. We map each match to a vector that depends on the two players playing the match, as well as conditions particular to the match itself. Our goal is to determine if the vector corresponds to a +1 or -1, where +1 corresponds to player1 winning, and -1 corresponds to player1 losing.

3.2 Feature Engineering

We considered a number of features while constructing a feature vector representation that produced decent results. We eventually settled on a set of features that intuitively made sense to us given our many years of watching tennis and the data that we were able to collect. Then, we algorithmically decided the subset of these features that produced the best results. We tried to keep the size of the set of features we tried relatively small so that the problem of choosing the optimal subset is computationally feasible given our limited resources. Additionally, we wanted to keep the dimensionality of our feature vectors as low as possible to avoid overfitting because of an extremely large feature space, and reduce generalization error.

We considered three types of features that can impact the outcome of the match:

- **match conditions** – Includes information about the match that is independent of the players.
- **player statistics** – Captures each player's level of performance.
- **matchup** – Indicates how the players in a match have played against each other or similar players in the past.

We initially generated around 50 candidate features, but we pruned our list down to the following 15 for each player:

1. number of titles
2. number of titles weighted by draw size
3. number of titles normalized by time spent on tour
4. number of titles weighted by draw size normalized by time spent on tour
5. number of sets played in last 6 weeks
6. percentage of weeks injured through career
7. ranking points
8. career set win percentage in the current round number, counting from the start of the tournament
9. career set win percentage in the current round number, counting from the end of the tournament
10. set win percentage against players with the same handedness as the opponent
11. set win percentage in tournaments with the same amount of rounds as the current tournament
12. set win percentage in matches played on the current surface
13. set win percentage throughout career
14. set win percentage throughout the last 12 weeks
15. set win percentage against the current opponent

We used set records as opposed to match records because it provides us with more data points. We did not go down to the granularity of games and points because we lose information about how a player performs on important points, a statistic that is of paramount importance in tennis. To further reduce dimensionality, we considered the difference between the corresponding features of the two players. This limits what the SVM can do with the features, but it makes sense given the symmetry between the first and second players and likely reduces overfitting.

Choosing an optimal subset of the candidate features is an expensive operation because all 2^{15} subsets need to be considered. For the sake of computational efficiency, we employed a greedy algorithm, whose pseudocode is furnished in the pseudocode below.

```
Initialize feature set to empty
Initialize best_maximum to 0
For all i in the range (0, num_features):
  Initialize iteration_maximum and iteration_best_feature
  Iterate through all remaining_features:
    Append just chosen feature to the feature set
      Run cross validation with feature_set
      if cross_validation_score > iteration_maximum:
        Update cross_validation_score and iteration_best_feature
  If best_maximum < iteration_maximum:
    Update best_maximum to iteration_maximum
  Otherwise,
    break
```

The cross validation step deserves further explanation. We took two approaches:

- **Challenge Points Approach:** In this method, we use the precomputed feature vectors for each first round match to predict who advances to the second round. In the second round and beyond, however, the feature vectors may have not been precomputed because these matches may not have happened in actuality. Computing these feature vectors on the fly is expensive, so, given our limited computational resources, we decided to limit our use of this approach, despite the fact that it may provide better end results, since it is maximizing for Challenge points, which is what we want.

- **Prediction Accuracy Approach:** This method is much more computationally feasible and thus allows for faster iteration. We looked at all of the matches in the tournaments in our test set and tried to predict their outcome based on their corresponding feature vectors and recorded what fraction of them we predicted correctly. This approach produces suboptimal results because, in the Challenge, correctly predicting some matches is more important than others because later matches give more points, but earlier matches determine the matchups that will occur later in the tournament. Without further experimentation, it is unclear how to weigh the importance of each match.

3.3 Experiments

Given our decision to use SVMs, we varied the algorithm along two dimensions: kernel and C, the regularization constant. We used cross validation to evaluate the effectiveness of each variant. Every experiment was conducted on a set of 40,000 examples – for every cross-validation run, we randomly selected a training set of approximately 34,000 examples for training and used the remaining 6000 examples for testing. In particular, we tried a radial basis, linear, and quadratic kernel and values of C in [0.1, 1, 10]. These choices represent popular kernels and values of C that span several orders of magnitude. This produced an optimal subset of features for each variant of the algorithm. This step used the Prediction Accuracy Approach for cross validation and its results are described in **Table 1**.

From these results, we learned that the quadratic kernel is not well-suited for this problem. More importantly, however, we learned what features were successful given each variant of SVM. Using the SVM with the linear and radial basis kernels and C in [0.1, 1, 10], we performed cross-validation using the Challenge Points Approach, and thus we were able to determine how what fraction of Challenge points our optimized algorithms could score. These results are summarized in **Table 2**.

Kernel	C	Optimal Feature set	Prediction accuracy
linear	0.1	4, 12, 13, 14	0.6558
linear	1.0	5, 12, 13, 15	0.6551
linear	10.0	11, 12, 13, 14	0.6602
quadratic	0.1	7, 8, 12, 13	0.5117
quadratic	1.0	5, 6, 7, 12, 13	0.5193
quadratic	10.0	3, 5, 8, 9, 12, 13	0.5094
radial basis	0.1	3, 8, 10, 12, 13, 14	0.6612
radial basis	1.0	3, 12, 13, 14	0.6549
radial basis	10.0	12, 13, 14	0.6558

Table 1: *The accuracy scores for each variant of SVM. The features are denoted by their indices in the list in the Feature Engineering section.*

Kernel	C	Fraction of Challenge Points
linear	0.1	0.5134
linear	1.0	0.4550
linear	10.0	0.5097
radial basis	0.1	0.5375
radial basis	1.0	0.5096
radial basis	10.0	0.5058

Table 2: *The fraction of Challenge points obtained for each variant of SVM, using the optimal features that we calculated earlier.*

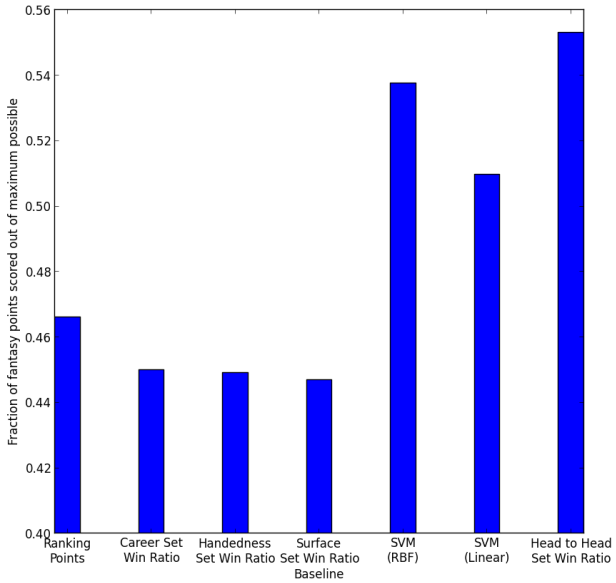

Figure 3: *Histogram showing performance of SVM algorithms compared to baselines*

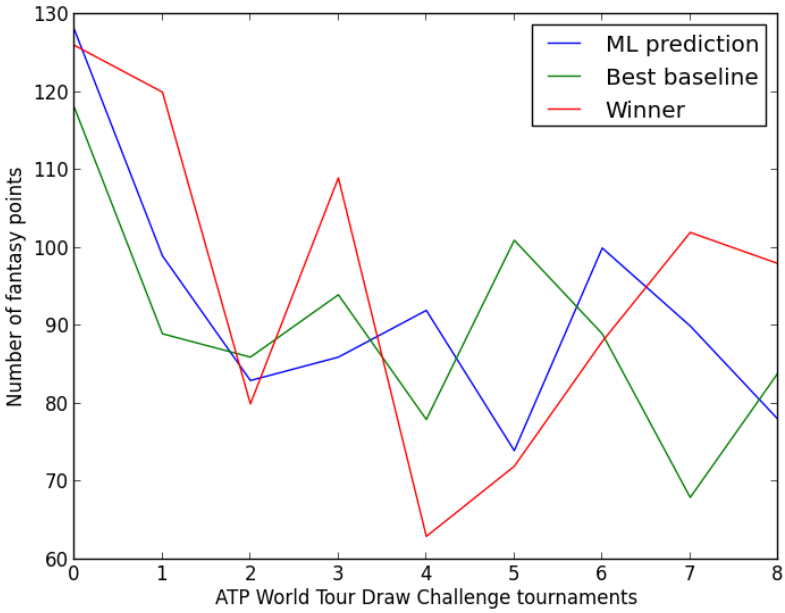

Figure 4: *This plot shows the ML predicted number of fantasy points in blue, the baseline's number of fantasy points in red, and the winner of the competition in red for all ATP World Tour Draw Challenge tournaments in 2013*

IV. Results

To put our results into context, we generated a baseline from each of the 15 features mentioned above. In these, we picked a random subset of tournaments and picked outcomes solely based on a baseline metric, and recorded what percentage of the Challenge points we were able to obtain.

We evaluated our model on different SVM parameter settings, as well as different evaluation algorithms. We observed that irrespective of the kernel function, the prediction accuracy increases as the value of C decreases, which makes intuitive sense since as C decreases, the SVM algorithm follows the training examples less closely and hence generalizes better. We believe, that decreasing the value of C further would give even better generalization guarantees; even though decreasing the value of C beyond a certain limit is likely to increase the generalization error since the model will start to not listen to the training examples enough.

Another key takeaway from Table 1 is that the quadratic kernel performed extremely poorly – this is likely because our feature vectors did not fit a quadratic curve. It is entirely possible that a polynomial kernel of another degree will work much better, but we did not have enough time to experiment with this parameter.

All of our experiments in Table 1, only achieve a maximum accuracy of around 0.65. We believe that this is a reasonable accuracy since there is still a fair bit of randomness in tennis, and sport in general. Expecting a machine learning algorithm to account for these instances of randomness is unreasonable.

Some of the features we considered were more successful than others. In particular, set win percentage on the current surface and set win percentage throughout the player's career were in the optimal subset for each of the SVM variants we tried. This makes a lot of intuitive sense because many victories in the past are a good indicator that a player will have victories in the future. Furthermore, the court surface is an extremely important factor on the professional circuit. Each surface has different characteristics that require different strategies. Many players grow up playing on only one surface and have most of their best results on that one

surface.

Despite win percentage against the current opponent producing the best baseline, it was only included in one of the feature sets. This may be because this feature is correlated with other features, and including it added more dimensions while not adding much new information.

V. Future Work

Future work includes playing with the feature representation, as well as experimenting with different algorithms. Possible other approaches that could produce better results are using only the top tournaments for training data, experimenting with more kernels and larger ranges of parameters, as well as using data from farther in the past. Given the good performance of our baselines, it may also be feasible to actually use a boosting algorithm using our baselines as the weak learners to obtain better results.

VI. References

Zifan Shi, Sruthi Moorthy, Albrecht Zimmermann, "Predicting NCAAB match outcomes using ML techniques âĂŞ some results and lessons learned", KU Leuven, Belgium

Pedregosa, F. et. al., "Machine Learning in Python", Journal of Machine Learning Research