Chemistry Reaction among NBA Players

Xianan Hao, Ruixiuan Hou, Weijia Mao
COSC 174 - Machine Learning and Statistical Data
Winter 2013

1. Introduction

How to choose the best five players to win the game? This interesting problem is essentially finding an optimal subset of items that maximizes the utility defined over these items. There exist two methods to solve the optimal subset problem. The first one is assuming a completely known preference function, given preference function parameters. The parameters can be learned from historical data. However, to our knowledge, it's hard to define and parameterize the preference function for NBA player problem. The second method is to maximize the set similarity between the selected subset and the labeled optimal subset, without assuming preference function. We use the second method in this project.

2. Dataset

In order to evaluate the relationship between dataset and learning algorithms, we used two different datasets of players.

The first dataset (referred as Dataset 1 in section 4) is downloaded from BasketballValue.com, which provides data for advanced statistical analysis of the players in NBA [1]. This dataset contains 30 features that describe the player's overall efficiency and effects on court in depth. Features in this data set include accumulative values, for example, the total points of one player over the whole season and the plus-minus stats that represent the overall points win/lose by the team when a particular player is on court. In a word, this data set is quite informative in a way that the good players differ much from the so-so players.

In contrast, the second dataset [2] (referred as Dataset 2 in section 4) records the efficiency of players. It contains 18 features and all features represent players' performance per unit time, for example, the average points of one player per 36 minutes. This dataset is less informative because many players' stats are much alike. For example, the field goal percentage of best 5 players is only a little higher than the remaining ones.

3. Model and Methods

Choosing 5 players that cooperate to provide the best performance is a problem of choosing the optimal subset of a ground set. By milestone, we implemented

the algorithm in [3] to choose the optimal subset of a team of players as the predicted best 5 of this team. However, our model before milestone was too simple and failed to take into consideration the interactions between pairs of players. So we changed our model and the new model is described in this part.

3.1 Final model

We use the following model as the score of 5-player combination performance:

$$\mathcal{T}(x, y; \Omega) = \Omega^{\mathrm{T}} \phi(x, y)$$

where x is the ground set of feature vectors of all players in a particular team and y is a possible 5-player subset. To model the individual player's performance as well as the pairwise interactions between players, our feature map $\phi(x, y)$ is defined as follows:

$$\phi(x, y) = \omega^T \phi_1(x, y) + \lambda \theta^T \phi_2(y)$$
$$\phi_1(x, y) = \sum_{v \in y} v - \sum_{v \notin y} v$$
$$\phi_2(y) = \sum_{i,j \in y} vec(ij^T)$$

where v, i, j are all real valued vectors in x representing the feature vector for each player; $\boldsymbol{\omega}, \boldsymbol{\theta}$ are weight vectors corresponding to each individual player and each pair of players, respectively; λ is a scalar to tune to give appropriate weights to individuals and pairs. Consistently, our overall weight vector $\boldsymbol{\Omega}$ is defined as follows:

$$\Omega = [\omega_1 \dots \omega_n \, \lambda\theta_1 \dots \lambda\theta_{n \times n}]^T$$

where n is the number of features in the feature vector of a player. As is illustrated above, for a certain possible subset, the way we model the individual player's performance is to add $\omega^{\mathrm{T}} v$ to the performance score when v is selected in y, and otherwise subtract $\omega^{\mathrm{T}} v$ from the performance score. Moreover, the way we model the pairwise interactions between players is to compute the outer product of each pair in a possible y (5 choose 2, so there are 10 possible pairs), vectorize these outer products as $(n * n) * 1$ vectors, sum them up and then concatenate them with the $n * 1$ vector of individual players to form the final feature map for a possible subset.

With the weight vector $\boldsymbol{\Omega}$ and feature map ϕ defined, the rest of the problem is to solve $\boldsymbol{\Omega}$.

3.2 Optimization problem

Our problem falls in to the regime of structured learning and it can be solved using structural SVM. The following is the formulation:

$$min_{\omega,\xi_i \geq 0} \frac{1}{2}\|\Omega\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } \forall 1 \leq i \leq n, \forall \hat{y} \in S(x_i)$$
$$\Omega^T(\phi(x_i, \hat{y}) - \phi(x_i, y_i)) - \xi_i \leq -\Delta(\hat{y}, y_i)$$

where m is the number of teams; y_i is the labeled optimal subset of the ith team; $\hat{y}$ is a possible subset and $\Delta(\hat{y}, y_i)$ is the loss function defined by:

$$\Delta(\hat{y}, y_i) = 1 - \frac{|\hat{y} \cap y_i|}{|y_i|}$$

This loss function captures the difference between a possible subset and the labeled true optimal subset.

3.3 Prediction

Once the weight vector $\boldsymbol{\Omega}$ is optimized , we compute the performance score for each possible subset and choose the one with the highest score as our predicted optimal subset:

$$y_{pred} = arg \max_{y \in S(x)} \mathcal{T}(x, y; \Omega)$$

3.4 Different approaches

3.4.1 Brute Force

Since our problem now is 10 choose 5 (i.e. 252), which is a quite small number, enumerating all the possible subsets of a 10-player team is feasible. So we tried brute force approach using a quite efficient convex programming package called CVX [4]. We arrange all feature maps and constraints in large matrices and input them into the CVX optimization part. We get good results using this approach.

However, if we want to use our model for another similar problem, for example, choosing the best 11-player subset from 30 players for a soccer team, the number of all possible subsets would be 30 choose 11 which is a way too large number for enumeration. So we need to find a way to make our method faster.

3.4.2 SSVM software with cutting plane algorithm

The cutting-plane algorithm [5] enables us to reduce our constraint size and therefore accelerate our method. The basic idea of the cutting plane algorithm is that it only uses the most violated constraints in the

optimization step. In each iteration, it picks the most violated constraint and puts it into the constraint set. As the number of constraints increases, the accuracy will become more and more closer to the brute force method.

In order to make use of the cutting-plane algorithm, we choose the svm-struct software for Matlab created by Thorsten Joachims [6]. To use the software, we defined three call back functions based on our problem description. The first call back function is the loss function that models the difference between the optimal subset and a possible subset. The second is the feature map function, which models our input as a vector. The last one is the constraint generation function, which identifies what is the most incorrect output y that the current model considers to be compatible with the input x and therefore find the most violated constraint. With these three vital functions defined and implemented, the software can solve our optimization problem successfully.

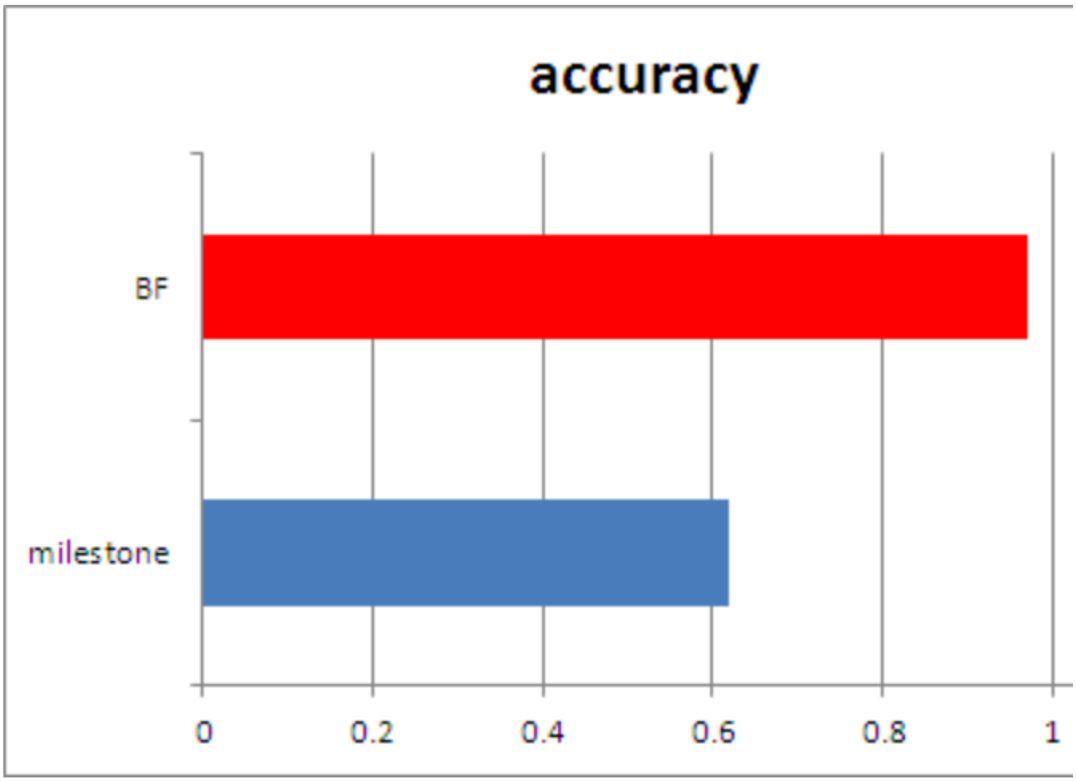
4. Experiments and Analysis

We used the data of NBA 2010 – 2011 season to test our method. As mentioned in section 2, we used two different datasets and we will discuss the results separately.

4.1 Results on Dataset 1

Dataset 1 is a quite informative dataset since the data of good players differs much from that of average players. There are 30 teams in all and we used 20 teams as training set and 10 teams as validation set. We tuned the parameters C and λ in our model and with C equal to 1 and λ equal to $30/900$ (which is reasonable since there are 30 features in individual player vector and 900 features in pairs of players vector), we get an average accuracy of 97.3% through 3-fold cross validation.

Compared to our results obtained by milestone, we get an accuracy improvement of over 30%:

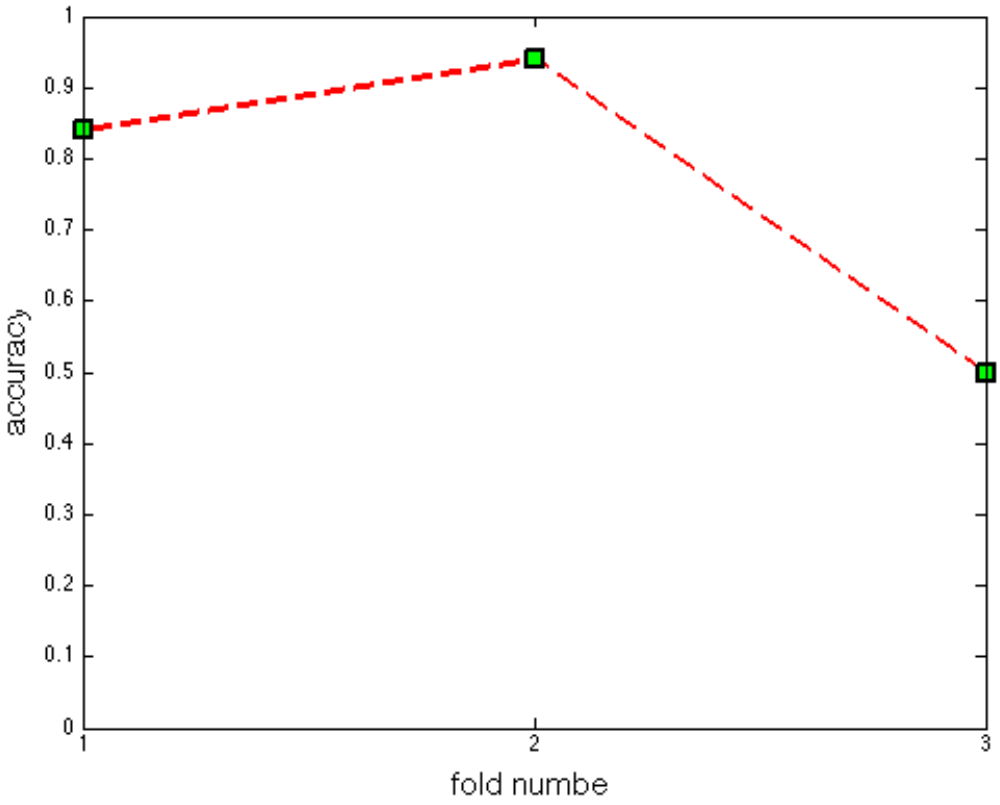


We can see that our final model fits the data very well and produces very good results. However, dataset 1 is very informative and we want to add some difficulty to our model and see what happens. Let's move on to the results on dataset 2.

4.2 Results on Dataset 2

4.2.1 Results of Brute Force approach

Dataset 2 is a less informative dataset and moreover, we deliberately added some mislabels of the optimal subset players to the last 10 teams. We also did 3-fold cross validation with 20 teams as training set and 10 teams as validation set. Following is the result:

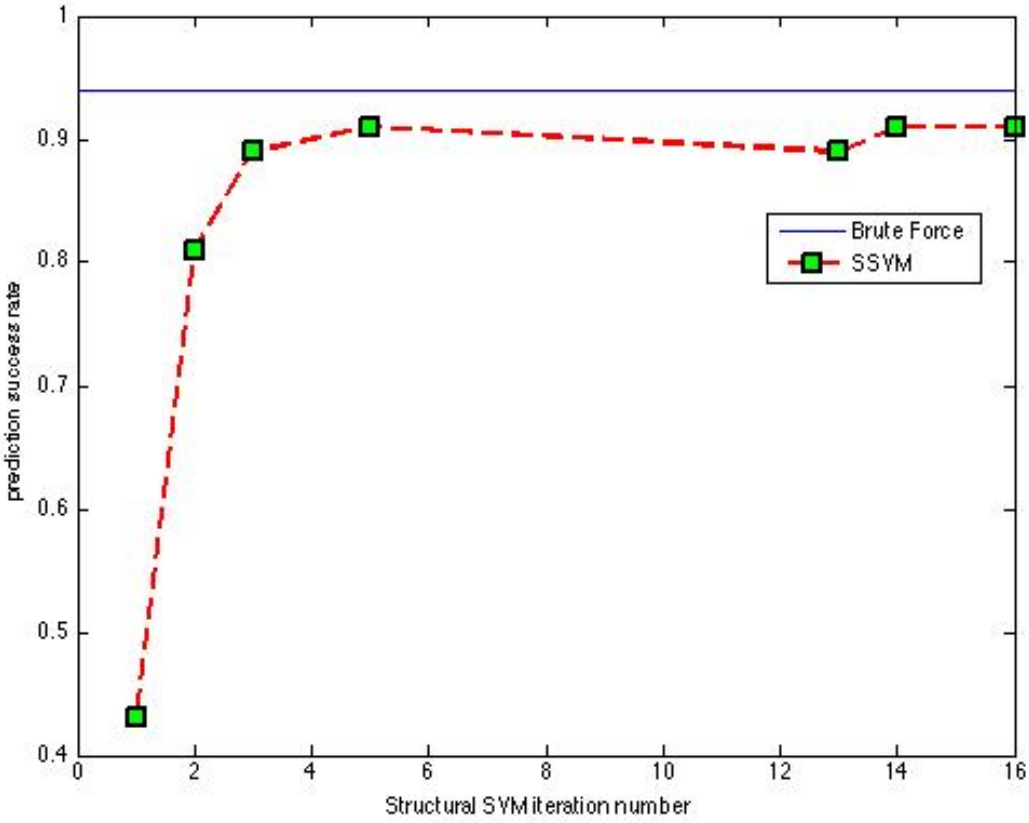


We can see that when the 10 teams containing mislabels are used as training teams, the results are still good: we get accuracy of 94% for one fold and 84% for another fold. However, when the 10 teams containing mislabels are used as testing set, the result gets very bad. This is reasonable because when we use the 10 with mislabels as training teams, there are still 10 training teams that are well labeled, so it is still possible for our method to learn a relatively good weight vector and produce good results on the validation set that is also well labeled. On the other hand, when the 10 teams with mislabels are used as validation teams, the weight vector is trained over 20 well labeled teams. Thus, this well-trained weight vector would certainly fail on a validation set that is incorrectly labeled. This result also shows that our model can tolerate some errors in labels. One important thing to mention is that we get the above result by tuning C to be 0.005. This change contributes to the tolerance of false labels because it makes the margins more "slack".

4.2.2 Comparison of Brute Force and SSVM software with cutting plane algorithm

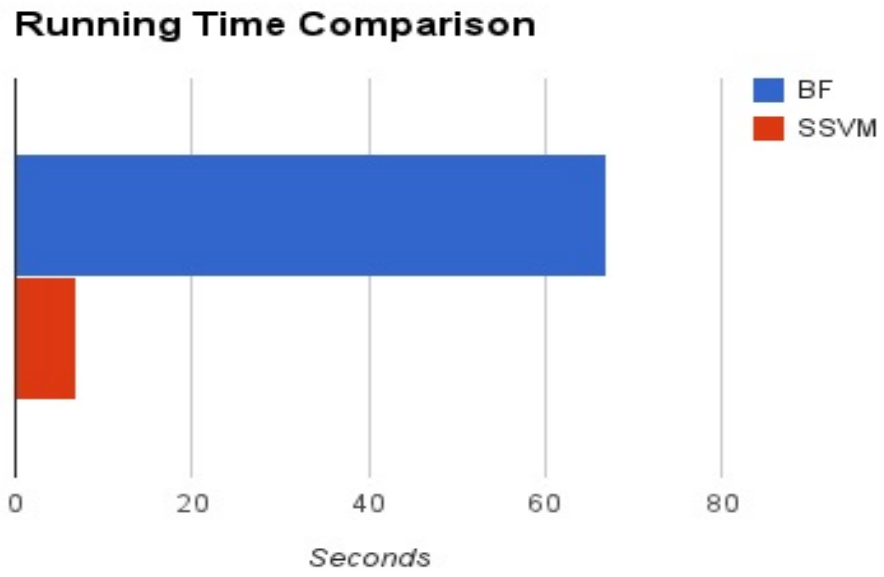
As we mentioned in section 3.4, besides enumerating all possible subsets, we also implemented a faster version of our approach with the help of the SSVM software. We compared the results of SSVM with cutting plane algorithm

with that of the brute force approach. This time, we used the 11th to 20th team as the test set (this test set is the same set as the validation set of fold 2 in the above figure, which produces an accuracy of 94% with BF approach). Following is the result:



As we can see, as the number of iteration increases, the accuracy of the SSVM with cutting plane algorithm grows closer and closer to that of the brute force approach, which we set as a baseline here. This is reasonable because as the iteration increases, the cutting plane algorithm puts more most violated constraints into its constraint set and thus the training process becomes more accurate. But since it considers only the most violated constraints while the brute force approach considers all the constraints, at the end its accuracy stays a little lower than that of the brute force approach.

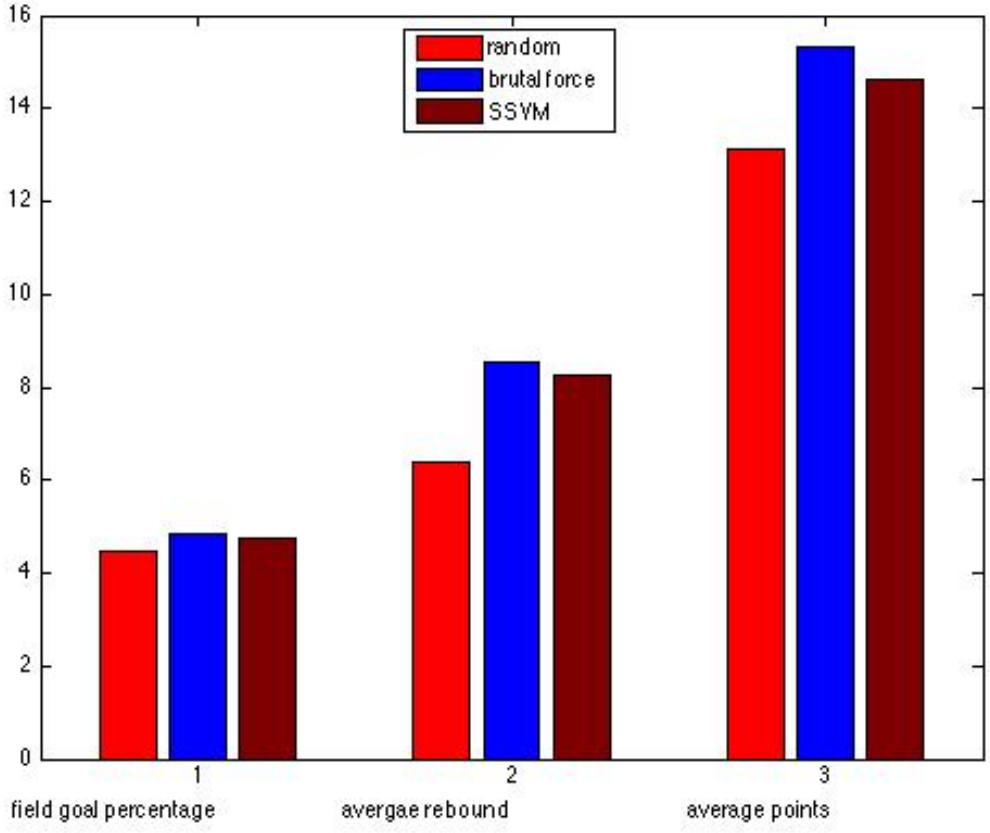
We mentioned that the SSVM with cutting plane algorithm approach can reduce the training time greatly and the following figure proves this claim:



We can see that, while it takes brute force approach around 70 seconds to train the model, it only takes SSVM with cutting plane algorithm around 7 seconds. Therefore, using SSVM with cutting plane algorithm, we think our proposed model can also make good predictions for larger sports teams.

4.2.3 Comparison of our methods with randomly selected 5-player teams

We compared our predictions with randomly selected 5-player teams with respect to several real player statistics such as field goal percentage, average rebounds and average points per 36 minutes. The following is the result:



We can see that the statistics of the predictions of brute force and SSVM with cutting plane algorithm approaches are quite similar and they are both better than randomly selected players. Since we used a less informative dataset, this amount of difference can be considered as a distinction between optimal subsets and average subsets.

4.3 Another choice of loss function

Our loss function described in section 3 captures the difference between the optimal subset and a possible subset, however, it doesn't capture to what degree is a possible different from the optimal subset. We modified our loss function as follows:

$$\Delta(\hat{y}, y_i) = \left(1 - \frac{|\hat{y} \cap y_i|}{|y_i|}\right) * \frac{\|\phi(x_i, \hat{y}) - \phi(x_i, y_i)\|}{\phi(x_i, y_i)}$$

This loss function takes into consideration the extent of difference between two subsets. However, in our problem, this loss function produces the same prediction accuracy as our original loss function.

5. External Software and Self-written code

In the code of our project, we used to external software:

1. CVX convex programming package: http://cvxr.com/cvx/
2. Structual SVM software: http://svmlight.joachims.org/svm_struct.html

The following files are written by ourselves:

1. In the Brute Force document: trainAndTest_CVX.m and generate_3_folds.m
2. In the SSVM with cutting plane algorithm: nba_player_pair.m and cross_validation.m

6. Conclusions

In this project, we proposed a method to make predictions of the best 5 players for a particular NBA team and obtained fairly good results. We learned a lot about machine learning by working on this project and had a lot of fun. Thanks to Professor Torresani and teaching assistant Du Tran for their patient and generous help.

References

[1] www.basketballvalue.com

[2] www.basketball-reference.com

[3] Y Guo, C Gomes, Learning Optimal Subsets with Implicit User Preferences, Proceedings of the 21st International Joint Conference , 2009

[4] http://cvxr.com/cvx/

[5] T. Joachims, T. Finley, Chun-Nam Yu, Cutting-Plane Training of Structural SVMs, Machine Learning Journal, 2009

[6] http://svmlight.joachims.org/svm_struct.html

[7] http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html