# How to Win at the Track

Cary Kempston
cdjk@cs.stanford.edu

Friday, December 14, 2007

## 1 Introduction

Gambling on horse races is done according to a pari-mutuel betting system. All of the money is pooled, the track removes its take, and whatever is left is divided among the winning bettors. A consequence of this is that the final odds are not known at the time of betting, since later bettors could influence the odds. Usually, however, estimated odds are posted based on the money wagered so far[4]. Another consequence of pari-mutuel betting is that all bets have a negative expected value if one uses the "market" odds posted by the track. This negative expected value is a consequence of the track take. Thus, to develop any successful gambling model of horse racing, one must develop better estimates of the horses win probabilities than the posted odds[2].

The other necessary component of a horse race betting system involves figuring out exactly how to bet. This sounds like an easy task, but there are many different possible types of bets. One can wager on a horse finishing first (win), first or second (place), first, second, or third (show), as well as many other bets involving multiple horses (quinella, trifecta, exacta, superfecta) and multiple races (daily double, pick 3, pick 6) [1]. Determining the optimal strategy in the face of these complex betting choices is beyond the scope of this project, so I will only consider simple, single-horse and single-race bets.

These two components would comprise a complete horse race betting system. This project, however, is concerned only with the first part of this system: developing a method to predict the winners of horse races.

## 2 Data

My data comes from a one month TrackMaster subscription[5]. This allowed me to download data about every horse race from twenty-three tracks from September 20, 2007 until November 26, 2007, comprising 58565 different races and 6827 finishing horses. I only considered thoroughbred horse races, since that is the most common type of horse racing in my data. I also only considered horses that finished, since my model has no notion of not starting or not finishing a race.

### 2.1 Features

My data consisted of information about each horse in each race. The features I selected out of this data are given in Figure 1.

| Feature | Explanation |
|---|---|
| Previous Wins by Horse | Percentage of races won by this horse |
| Number of Races by Horse | |
| Weight Carried | Weight of jockey and saddle |
| Horse Age | |
| Horse Sex | One of: Colt, Filly, Gelding, Mare, Stallion |
| Medication Given | Bute and/or Lasix |
| Previous Wins by Jockey | Percentage of races won by this jockey |
| Number of Races by Jockey | |
| Previous Wins by Trainer | Percentage of races won by this trainer |
| Number of Races by Trainer | |
| Post Position | How far from the inside of the track the horse starts |
| Final Odds | Final odds for this horse to win given by the track |
| Track Conditions | One of: Freezing, Fast, Good, Heavy, Muddy, Snow, Slow, Sloppy |

Figure 1: Features for machine learning algorithms

All features that have values outside of $[0, 1]$ are normalized to be within that range. The past performance history is only for the time period that my data spans, as I was unable to find a source of win data by horse that was both comprehensive and inexpensive. The values in the horse sex category are Colt (young male), Filly (young female), Mare (older female), Gelding (castrated male), or Stallion (older male). That feature, and the Track Conditions feature were encoded as binary variables - i.e. each sample has a feature that is 1 if the horse is a filly, and 0 if not, for each of the five possible values of horse sex and eight possible values for track conditions.

As target output value, I have both the finishing position of the horse, and the lengths behind the leader that the horse finished.

# 3 Approaches

This project models horse races. There are some number of entrants in a race, and they cross the finish line in a ranked order. There are a number of machine learning approaches that can model races, including ranking algorithms (i.e. ordinal regression), classifiers, and regression. All of these approaches assume independence of each horse's chance of winning; the other horses in a race are not involved in any probability calculations.

## 3.1 Ranking

Initially horse racing seems like a natural place to use a ranking algorithm or some sort of ordinal regression, which, given a training sample, tries to learn it's ordered rank. In this case, the rank would be the finishing position of a particular horse. Using an ordinal regression classifier would then involve giving it the feature vectors of each horse in a race, and having it predict the finishing place for each horse. There is a problem with this approach, however, because of the independence assumption. Ordinal regression would output that a particular horse is a "fourth place" horse in general, ignoring the other horses in the race. This could result in either multiple predicted first

place finishers, or no predicted first place finishers. I discovered no easy way around this problem, so I did not implement any ranking algorithms.

### 3.2 Binary Classification

Binary classification is a more promising approach. This approach learns a classifier on the data that predicts whether a horse will win the race or not. This algorithm still faces the same problem as trying to rank finishers, in that multiple or no winners could be predicted, but most classification algorithms give us some measure of the confidence of the prediction (such as the distance from the separating hyperplane in an SVM). After running a classifier on every horse in a race, we can predict the winner by picking the horse that the classifier has the highest confidence of winning. This approach is shown schematically in Figure 2.

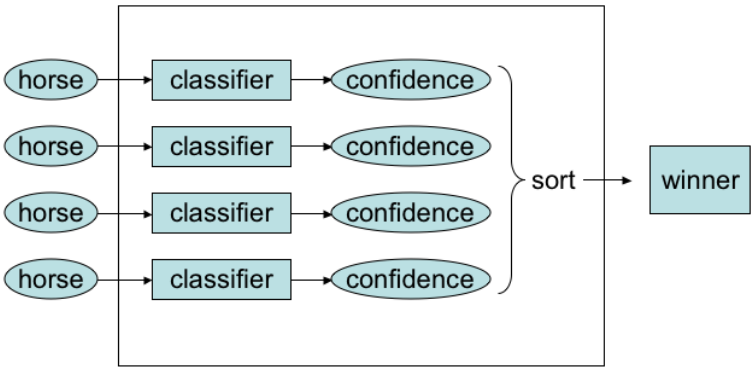


Figure 2: Using individual binary classifiers to predict one winner.

### 3.3 Regression

The final approach is using regression to predict how far behind the leader a horse will finish. After predicting this for each horse in a race, it is an easy matter to sort the horses by the distance behind the leader and pick the smallest one as the winner. The schematic for this looks very similar to the classification approach, but with "regression" replacing each box labeled "classifier," and sorting by the output of the regression instead of confidence.

### 3.4 Error Rate

Since the data is on a per-horse level, but we are really interested on a per-race level, some work is required to determine the error rate. Instead of counting every win/loss misclassification as an error, we should instead go through the data for each race, predict the winner based on the classification approach described in Section 3.2, and count the prediction as correct if the picked horse won, regardless of any other horses classified as winners.

Furthermore, it is possible that our approach may not be good at predicting the winner of a race, but is still good at predicting "fast" horses. Since there are win, place, and show bets, we

| **Finishes in top $n$** | **Probability horse wins** |
|---|---|
| 1 | 12.6% |
| 2 | 22.7% |
| 3 | 33.6% |

Figure 3: Success rate of naïve prediction strategy.

can calculate the error rate based on the horse coming in first, first or second, and first, second, or third.

To determine whether these approaches are successful, consider a naïve prediction strategy that involves predicting the winning horse by picking the horse with the most favorable odds. The results for this strategy are given in Figure 3.

### 3.5 Method

All learning (classification and regression) was done with support vector machines, using both linear and gaussian kernels. I used SVM$^{light}$[3] for all SVM calculations. Cross validation holding back 30% of the data was done to generate the test success rates. In experimenting with different parameters, I found that using $C = 10$ and $\sigma^2 = 0.1$ (for Gaussian Kernels) to give reasonable values for the classification problems. I used values $C = 1$ and $\sigma^2 = 10$ for the regression problems, although it turns out that regression is not a good approach for this problem, and all values of the parameters are equally bad.

## 4 Results

Initial results are given in Figure 4. These results are not very good. For the regression tests, the test set success rate is higher than the training set success rate, while for classification the the training set success rate is significantly higher than the test set success rate, suggesting overfitting.

| | | $n = 1$ | | $n = 2$ | | $n = 3$ | |
|---|---|---|---|---|---|---|---|
| | Kernel Type | Train | Test | Train | Test | Train | Test |
| Regression | Linear | 3.0 | 12.4 | 14.0 | 22.1 | 27.6 | 32.8 |
| | Gaussian | 2.9 | 12.3 | 12.9 | 21.2 | 27.1 | 33.6 |
| Classification | Linear | 59.7 | 13.6 | 67.1 | 24.0 | 73.6 | 36.8 |
| | Gaussian | 60.0 | 13.9 | 67.4 | 24.2 | 73.5 | 36.6 |

Figure 4: Training and test set success rates (%) using all features ($n$ is the place the predicted horse can finish and still count as a success).

The performance of the regression tests is not very good, most likely because the output variable (i.e., the finish distance behind the leader) is not consistent across races. For example, a horse could come in second in two races, but finish right behind the leader in one and a long distance behind the leader in another. These outcomes should be treated the same, but they are treated very differently by a regression algorithm. Regression is therefore taking some element of "competition"

into account (since finishing immediately behind the leader suggests a more competitive race), but not in any meaningful or consistent way.

The success rates of the classification experiments suggests overfitting. The most likely cause of this is that I only have win/loss history for each horse for the 67 days that my data spans. This data includes results for 29409 horses, of which approximately 40% only raced once. This resulted in a large number of my training features about the win/loss record of the horse to be either 0% or 100%, which is not good data for probability calculations.

To correct for this likely overfitting, I repeated my calculations with all the win/loss data removed. These results are given in Figure 5. Removing the previous win/loss history corrects the overfitting for the classification tests, but the problems in the regression tests remain.

| | | $n = 1$ | | $n = 2$ | | $n = 3$ | |
|---|---|---|---|---|---|---|---|
| | Kernel Type | Train | Test | Train | Test | Train | Test |
| Regression | Linear | 5.3 | 12.5 | 10.3 | 21.9 | 18.5 | 34.4 |
| | Gaussian | 5.0 | 12.0 | 10.9 | 21.4 | 20.2 | 32.6 |
| Classification | Linear | 17.1 | 13.0 | 29.2 | 22.6 | 43.2 | 34.3 |
| | Gaussian | 13.7 | 13.2 | 23.8 | 23.0 | 36.2 | 34.5 |

Figure 5: Training and test set success rates (%), excluding all previous win features ($n$ is the place the predicted horse can finish and still count as a success).

## 5 Conclusion

My results show that machine learning techniques can be used to better predict the outcomes of horse races than the odds. I was able to predict first, second, or third place finishes .9% more often than the naïve strategy of picking the horse with the best odds. Although this is a small improvement, the large size of the horse racing industry needs to be taken into account in interpreting the results.

## References


[1] Glossary of horse racing terms. `http://www.drf.com/help/help_glossary.html`.

[2] Ruth N. Bolton and Randall G. Chapman. Searching for positive returns at the track: A multinomial logit model for handicapping horse races. *Management Science*, 32(8):1040–1060, 1986.

[3] T. Joachims. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning.* B. Scholkopf, C. Burges and A. Smola (ed.). MIT Press, 1999.

[4] Pari-mutuel betting. `http://en.wikipedia.org/wiki/Parimutuel_betting`.

[5] Trackmaster. `http://www.trackmaster.com`.