CS230

Ga Yau: Machine analysis of Hong Kong horse racing data

Olaf Torné
otorne@stanford.edu

Abstract

We present a horse-race forecasting model that utilizes a set of carefully selected features, encoding expert domain-knowledge on how to select a winning horse. A tunable labelling scheme is introduced that significantly improves model accuracy. Intuition is given on the relationship between feature-selection and network depth. This model builds on prior works by introducing all of the above, as well as considering a multi-horse softmax framework instead of the more common single-horse sigmoid forecasters.

1 Introduction

This project explores the usage of neural-networks and feature-engineering to analyse data of horse-races organized by the Hong Kong Jockey Club (HKJC), and make forecasts.

Why Deep-Learn horse racing? It is a challenging problem to make accurate forecasts of horse-race results:

1. Only some of the potentially determining features are publicly available
2. A horse-race is reasonably close to a random event [1]

Nevertheless, the past-history of races is available and it's a known fact that outcomes are not completely random but also depend on factors observable in the past history. Therefore horse-racing is a promising candidate for machine learning analysis.

We consider two models: firstly, a hand-engineered features model (DH_v1) which is analysed in full detail. Secondly, using insights gained on the hand-engineered model, we offer some motivation and preliminary analysis on constructing an end-to-end model (DH_v2). A key insight of this project has been to understand how each approach leads naturally to a different network architecture, and analyse benefits and limitations of either approach in the context of horse-race prediction.

Hand-engineered features DH_v1. The general recipe is as follows:

1. Raw data is a set of past-race tables $X^{(i)}_{raw}$, $i = 1 \dots m$, where rows are horses in the race, and columns are features describing the properties of that race (horse, jockey, distance etc, as well as the order of arrival of each horse at the finish)
2. Hand-engineered features, based on expert domain-knowledge and encoding past-performance, are added as new columns

[1] One reason for this is the race manager handicaps each horse by adding metal weights based on the horse's current ranking in a deliberate attempt to equalize the horses chances

3. After adding the new high-level features the identities of horses/jockeys/trainers are discarded
4. Fit a fully-connected neural network to the data
5. The fitted model takes as input a new race table, and forecasts which horse will win

Discarding the identity of participants in the race means the model learns the features-to-results mapping, but does not learn about the past performance of any specific horse. This is not a significant limitation in itself because the population of horses changes over time, and occurrences of the same horse are relatively sparse. On the other hand it implies the model has limited scope to learn new general past-performance features. This is potentially a more serious limitation and in section 3 we supply an example of an expert decision rule that is not captured by our set of hand-engineered features.

The main benefit of the horse-identity-agnostic approach, beyond that it works well in practice, is that it leads to a simple model architecture. Indeed, the input has shape $(m, 14, 33)$ and can easily be approached using a fully connected network. Several published works consider a similar set-up based on horse-agnostic high-level features (Bolton et al. (1986), Chen et al (1995), Pudaruth et al. (2013)). Nevertheless this project introduces some incremental novelty:

- A much larger set of sophisticated features derived from expert domain knowledge
- A much larger training set
- Analysis of NN versus public odds benchmark
- Multi-horse/softmax forecasting model, as opposed to single-horse/sigmoid forecasting
- Tunable loss / labelling system to achieve higher accuracy

Moreover, three interesting discoveries were made:

1. DH_v1 handily outperforms public odds and random selection benchmarks
2. Good accuracy is achieved with a shallow fully-connected network and does not improve significantly with increased network size
3. Higher accuracy is achieved by tuning the loss to favor 1st, 2nd, 3rd place (via labelling)

Intuition for 2. is that the hand-engineered features are already sufficiently rich to encode much of the information that determines a race outcome and can be exploited as such by a simple network. Moreover, because horse-identity is discarded, the model has no scope to learn complex dependencies across different races, where a deeper network might outperform. To illustrate, consider this empirical fact:

If a horse won its last race on identical distance and track-type, it has a greater chance of winning the current race.

The premise is already represented in the feature `horse_dist_course_rec_1` (encoding previous result on same distance/track), so it can conceivably be exploited even by a linear boundary. On the other hand, it is impossible for even a very deep network to learn this fact on its own, and adding network complexity does not affect this.

Discovery 3. is also intuitive: tuning the loss function to reward not only correct 1st place predictions, but also 2nd and 3rd, helps it to converge to a model that has better accuracy, even when the final model is just applied to selecting 1st place.

End-to-end DH_v2. Given the observations above, it is natural to consider an alternative approach that supplies raw data X_{raw}, including horse/jocky identity but omitting hand-engineered features, to a deeper network to discover if it can effectively learn high level features itself. Due to time constraints and a greater initial focus on DH_v1, we did not bring DH_v2 to full-fruition. However we offer some insights into where are the difficulties, in particular related to problem dimension, and also describe a general approach to this model.

2 Related work

Single-horse, horse-agnostic frameworks, with feature engineering. Several prior works (Davoodi et al (2010), Pudaruth et al. (2013), Williams et al (2008), Chen et al (1995)) build a model that takes a single horse as input, described by a number of hand-engineered features, and forecasts if that horse will win independently of the other competitors. These works consider datasets of a few hundred races, and use less than 10 features. Numerical results are not comparable across publications due to significant differences in race modalities. *Multi-horse, horse-agnostic framework with feature engineering.* Bolton et al. (1986) considers a multinomial logit model to capture the win-probability of all horses participating in a race, and includes a methodology for data augmentation by considering sub-races. The dataset is hand-assembled and contains 200 races and 10 simple features. Weights are shared across horses leading to a parsimoniously specified model. This paper was a great inspiration for this project. In particular the architecture we present can be viewed as a generalization of the Bolton-Chapman model to a fully-connected dense network with 33 features. *Multi-horse framework with end-to-end approach.* The graduation thesis Wong. (2018) looks at a simplified end-to-end model where horse identities are discarded, but other categorical features such as jockey and trainer are retained, and only a very limited number of hand engineered features are added, with minimal encoding of past-performance data. This model is conceptually interesting but has the limitation that horse-identity is the most important feature in an end-to-end approach.

3 Dataset and features

3.1 Raw data

Raw data was acquired from SPP (2021). The data comprises 12820 races from 2003-2020. The raw data is supplied in three tables corresponding to horse, track, jockey and trainer features. These are aggregated into a single table X_{raw} : rows are the participants in a race; columns are features. An example of X_{raw} is given in the Appendix.

3.2 Discussion of the feature-engineering route

A raw race-table X_{raw} only contains information about the current race (e.g. distance, horse-names etc). Past-performance is then encoded using hand-engineered features. This approach has several advantages:

1. Readily available expert-advice about which features are relevant (see HKJC (2021))
2. Cost of translating expert-advice into quantitative features is not negligible, but it was deemed acceptable
3. Results in simple, uniform tabular inputs
4. Relatively shallow network, as high-level features are pre-supplied.

3.3 Feature-engineering detail

The HKJC publication *Racing 201: How to pick your winning horse* HKJC (2021) gives expert advice for selecting a winning horse. The feature-engineering stage translates a subset of this advice into numerical features. For illustration:

- `horse_dist_course_going_rec_1` Most recent finishing position of the horse on identical distance, track-type and track-conditions
- `trainer_dist_course_going_rec_1` Most recent finishing position of any horse by the same trainer, on identical distance, track-type and track-condition

The horse, jockey and trainer's identities are used to derive past-performance features, then they are discarded. In total 33 features are used per horse. The full list of hand-engineered features is supplied in the Appendix. Lastly, races with less than 14 horses are padded to have a uniform 14 rows. A single input to the model therefore has shape:

$$\text{Input.shape=}(\cdot, 14, 33).$$

3.4 Train/Dev/Test split

The distribution of races is expected to vary over time. Indeed, the population of horses, jockeys and trainers evolves; HKJC rules or handicapping methodology may change; Horse performance is sensitive to climatic conditions which may vary systematically. Moreover, we are only interested making forecasts about future races. Therefore we first split as follows:

1. Dev/Test sets are drawn from the 2 most recent seasons (2018-2019/2019-2020)
2. Training set is drawn from the earlier seasons 2003 through 2018

Once this split is done, examples can be drawn in a standard way, including by random sampling. This is not a problem because the model only makes forecasts within a current race example using it's own set of features, and does not model across chronologically related race examples.

4 Methods

4.1 Architecture

Input: A single input example for DH_v1 is a matrix R representing all the participants in a single race. Rows are horses and columns are features. The shape is $(n_h = 14, n_f = 33)$.

Output: A probability vector p, of length n_h, where p_i is the probability horse i will win .

Architecture: DH_v1 is a fully-connected network with standard architecture:

$$R \to \text{Flatten} \to \text{Linear} \to \text{Relu} \to \cdots \to \text{Linear} \to \text{Relu} \to \text{Linear} \to \text{Softmax}$$

After tuning, the parameter combination chosen is two hidden layers with 60 units each, and a dropout probability of 0.5. There are 32,534 trainable parameters.

4.2 Labelling and loss function

We use custom labelling, as opposed to one-hot, to capture that assigning a high win-probability to a horse that ends in second or third place is still a good result. To this end, define three hyperparameters:

$$p_{win}, p_{sec} \text{ and } p_{third}$$

such that $p_{win} + p_{sec} + p_{third} = 1$. The label associated to an input example R is y_{top3}: all entries are zero except for horses in first, second and third place, who are assigned p_{win}, p_{sec} and p_{third} respectively. See Figure 1. The standard `tf.softmax_cross_entropy_with_logits` loss function is used. After tuning, values $p_{win} = 0.5, p_{sec} = 0.25$ and $p_{third} = 0.25$ were retained.

4.3 Benchmarks and evaluation metrics

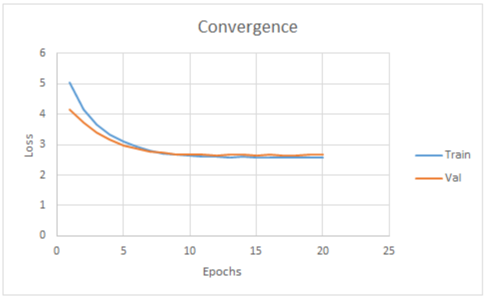
The model is compared to two benchmarks:

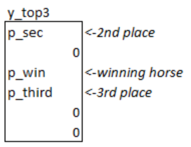
- Uniform Model (UM): randomly pick a horse. As the number of horses per race varies, the expected accuracy needs to be calculated explicitly by summing the expected accuracy of each race.
- Public Odds (PO): in the pari-mutuel system, betting odds directly reflect the amount wagered on each horse and thus supply an implied value for the public's estimate of winning probabilities.

We consider the standard **accuracy** metric, and also **top-3** that counts success if our highest-win-probability horse ends in 1st, 2nd or 3rd place.

4.4 Limitations of feature-engineering and discussion of end-to-end approach

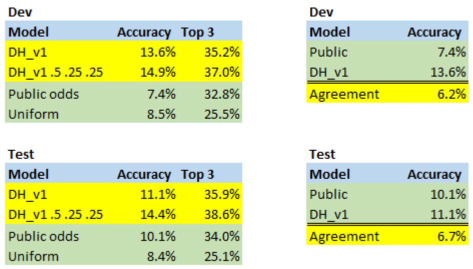
Using hand-engineered features results in forecasts that are significantly more accurate than the benchmarks, thus validating this approach. Nevertheless, this approach has limitations as some documented expert-advices are not captured by our set of hand-engineered features, and the model has little scope to learn new high-level features based on past-performance. The appendix contains a more detailed analysis.





y_top3	
p_sec	*<-2nd place*
0	
p_win	*<-winning horse*
p_third	*<-3rd place*
0	
0	

Figure 1: *Left.* Loss function convergence.*Right.* Custom labelling.



Dev

Model	Accuracy	Top 3
DH_v1	13.6%	35.2%
DH_v1 .5 .25 .25	14.9%	37.0%
Public odds	7.4%	32.8%
Uniform	8.5%	25.5%

Dev

Model	Accuracy
Public	7.4%
DH_v1	13.6%
Agreement	6.2%

Test

Model	Accuracy	Top 3
DH_v1	11.1%	35.9%
DH_v1 .5 .25 .25	14.4%	38.6%
Public odds	10.1%	34.0%
Uniform	8.4%	25.1%

Test

Model	Accuracy
Public	10.1%
DH_v1	11.1%
Agreement	6.7%

Figure 2: *Left.* Accuracy and top-3 metrics. DH_v1 is the vanilla labelling model, and DH_v1 .5 .25 .25 uses tuned labels *Right.* Forecast agreement between DH_1 and PO models.

5 Experiments and results

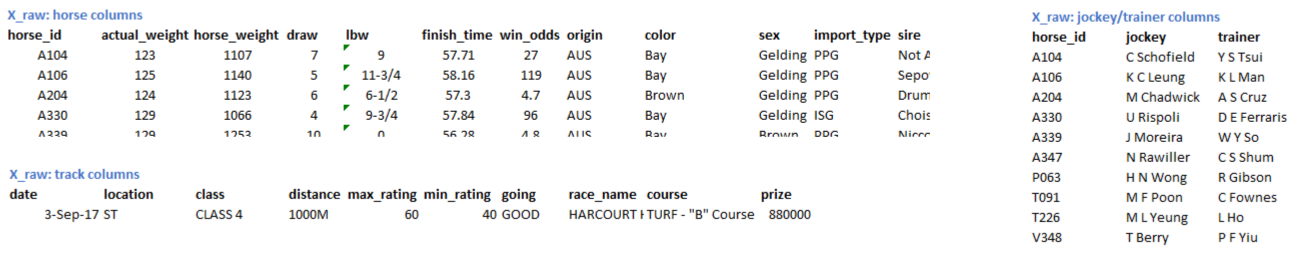
The fitting process of DH_v1 converges as shown in Figure 1. This example uses standard labelling with $p_{win} = 1.0, p_{sec} = p_{third} = 0$. Similar qualitative results are obtained with tuned labels. Accuracy and top-3 metrics are displayed on the left of Figure 2. The tuned model DH_v1 $p_{win} = 0.5, p_{sec} = p_{third} = 0.25$ outperforms significantly. The right of Figure 2 compares forecasts made by DH to those made using public odds: in the test set the winning horse forecast matches in only 6.7% of cases.

6 Conclusion and future work

This project presents a horse-race forecasting model with the following characteristics:

- Multi-horse softmax framework
- Rich hand-engineered feature set
- Horse-identity agnostic framework
- Tunable labelling scheme

The model is demonstrated to significantly outperform the random-selection and public-odds selection benchmarks. Detailed intuition is given about why the model performs well even with a shallow network architecture, and why the tunable loss/labelling scheme gives rise to better accuracy even when the final model is used to predict 1st place. Lastly, it offers a candid discussion about model limitations, in particular the inability to learn new past-performance based features, and a discussion of how to potentially remedy this limitation in future iterations.


X_raw: horse columns

horse_id	actual_weight	horse_weight	draw	lbw	finish_time	win_odds	origin	color	sex	import_type	sire
A104	123	1107	7	9	57.71	27	AUS	Bay	Gelding	PPG	Not A
A106	125	1140	5	11-3/4	58.16	119	AUS	Bay	Gelding	PPG	Sepo
A204	124	1123	6	6-1/2	57.3	4.7	AUS	Brown	Gelding	PPG	Drum
A330	129	1066	4	9-3/4	57.84	96	AUS	Bay	Gelding	ISG	Chois
A339	129	1253	10	0	56.28	4.8	AUS	Bay	Brown	PPG	Nicc

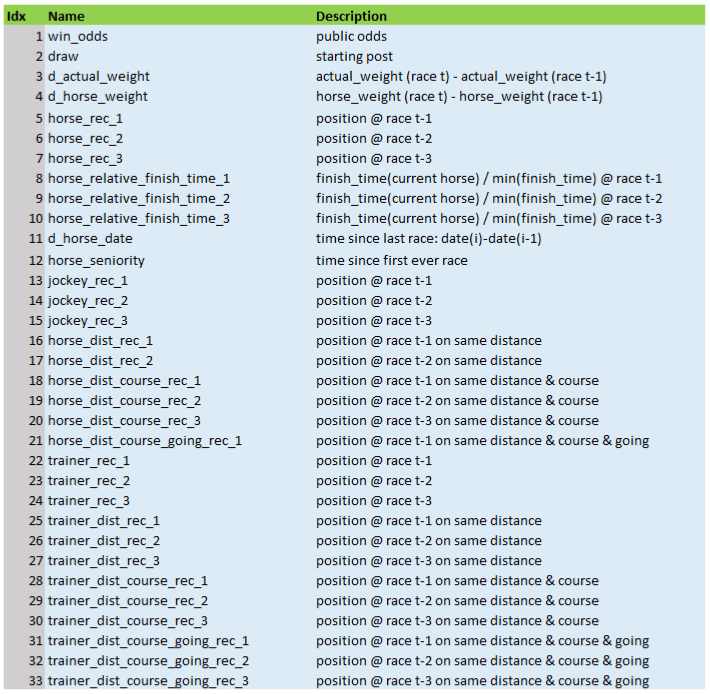
X_raw: track columns

date	location	class	distance	max_rating	min_rating	going	race_name	course	prize
3-Sep-17	ST	CLASS 4	1000M	60	40	GOOD	HARCOURT I	TURF - "B" Course	880000

X_raw: jockey/trainer columns

horse_id	jockey	trainer
A104	C Schofield	Y S Tsui
A106	K C Leung	K L Man
A204	M Chadwick	A S Cruz
A330	U Rispoli	D E Ferraris
A339	J Moreira	W Y So
A347	N Rawiller	C S Shum
P063	H N Wong	R Gibson
T091	M F Poon	C Fownes
T226	M L Yeung	L Ho
V348	T Berry	P F Yiu

Figure 3: Example of raw input data for one race.



Idx	Name	Description
1	win_odds	public odds
2	draw	starting post
3	d_actual_weight	actual_weight (race t) - actual_weight (race t-1)
4	d_horse_weight	horse_weight (race t) - horse_weight (race t-1)
5	horse_rec_1	position @ race t-1
6	horse_rec_2	position @ race t-2
7	horse_rec_3	position @ race t-3
8	horse_relative_finish_time_1	finish_time(current horse) / min(finish_time) @ race t-1
9	horse_relative_finish_time_2	finish_time(current horse) / min(finish_time) @ race t-2
10	horse_relative_finish_time_3	finish_time(current horse) / min(finish_time) @ race t-3
11	d_horse_date	time since last race: date(i)-date(i-1)
12	horse_seniority	time since first ever race
13	jockey_rec_1	position @ race t-1
14	jockey_rec_2	position @ race t-2
15	jockey_rec_3	position @ race t-3
16	horse_dist_rec_1	position @ race t-1 on same distance
17	horse_dist_rec_2	position @ race t-2 on same distance
18	horse_dist_course_rec_1	position @ race t-1 on same distance & course
19	horse_dist_course_rec_2	position @ race t-2 on same distance & course
20	horse_dist_course_rec_3	position @ race t-3 on same distance & course
21	horse_dist_course_going_rec_1	position @ race t-1 on same distance & course & going
22	trainer_rec_1	position @ race t-1
23	trainer_rec_2	position @ race t-2
24	trainer_rec_3	position @ race t-3
25	trainer_dist_rec_1	position @ race t-1 on same distance
26	trainer_dist_rec_2	position @ race t-2 on same distance
27	trainer_dist_rec_3	position @ race t-3 on same distance
28	trainer_dist_course_rec_1	position @ race t-1 on same distance & course
29	trainer_dist_course_rec_2	position @ race t-2 on same distance & course
30	trainer_dist_course_rec_3	position @ race t-3 on same distance & course
31	trainer_dist_course_going_rec_1	position @ race t-1 on same distance & course & going
32	trainer_dist_course_going_rec_2	position @ race t-2 on same distance & course & going
33	trainer_dist_course_going_rec_3	position @ race t-3 on same distance & course & going

Figure 4: Engineered features.

Appendices

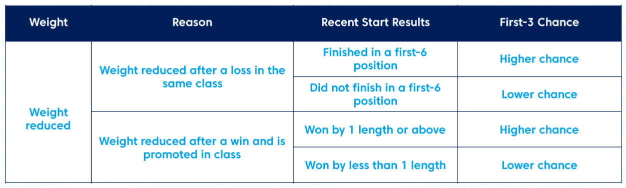
A Input data

An example of raw input data is in Figure 3. The set of engineered features in DH_v1 is in Figure 4.

B Discussion of end-to-end approach

Using hand-engineered features results in forecasts that are significantly more accurate than the benchmarks, thus validating this approach. Nevertheless, this approach has limitations as some documented expert-advices are not captured by our set of hand-engineered features, and the model has little scope to learn new high-level features based on past-performance.

To illustrate, consider the decision rule in Figure 5. Our set of features contains the carried-weight change `d_actual_weight` and most recent performance `horse_rec_1` corresponding to the first and third columns, but no information about class change. Therefore it is impossible for the model to learn this rule. An alternative approach is to transform categorical data from the raw input table X_{raw} into one-hot representations, and pass this to the network. As race-data is naturally sequential, one could consider an RNN model. Alternatively one may consider a standard FC network as in Wong.



Weight	Reason	Recent Start Results	First-3 Chance
Weight reduced	Weight reduced after a loss in the same class	Finished in a first-6 position	Higher chance
		Did not finish in a first-6 position	Lower chance
	Weight reduced after a win and is promoted in class	Won by 1 length or above	Higher chance
		Won by less than 1 length	Lower chance

Figure 5: Example of a complex decision rule for selecting a horse.

(2018). With this approach the feature set now has shape:

$$Input.shape = (\cdot, 14, 7386)\,.$$

Moreover a deep network will necessarily be required to learn horse-identity driven dependencies. Given the input shape, a fully-connected network may not be the most appropriate architecture, and it remains to be investigated what would be the most appropriate architecture. As a proof of concept, one may first discard horse identities as in Wong. (2018), thus resulting in an input of shape $Input.shape = (\cdot, 14, 587)$, however it remains to be discovered what is an appropriate architecture for a full end-to-end model including the most important feature of horse-identities.

References

Ruth N. Bolton and Randall G. Chapman. Searching for positive returns at the track: A multinomial logit model. *Management Science*. 1986

Janett Williams and Yan Li. A Case Study Using Neural Networks Algorithms: Horse Racing Predictions In Jamaica. *Proceedings of the 2008 International Conference on Artificial Intelligence, ICAI 2008, July 14-17.* 2008.

Elnaz Davoodi and Ali Reza Khanteymoori. Horse Racing Prediction Using Artificial Neural Networks. *Proceedings of the 11th WSEAS international conference on nural networks. Pages 155–160. June* (2010)

S. Pudaruth and N. Medard and Zaynah Bibi Dookhun. Horse Racing Prediction at the Champ De Mars using a Weighted Probabilistic Approach. *International Journal of Computer Applications*. 2013

Hsiu-chin Chen and Surjono H. Sutjahjo. Expert prediction, symbolic learning, and neural networks. An experiment on greyhound racing *Article in IEEE Expert.*1995

Luk Wong. Horse Racing Prediction using Deep Probabilistic Programming with Python and PyTorch (Uber Pyro). *Final year project report. Chinese University of Hong Kong.* 2018

http://hkracingdb.com/ 2021

Racing 201: How to pick your winning horse. *Hong Kong Jockey Club.* 2021