Horse Racing Prediction using Deep Probabilistic Programming with Python and PyTorch (Uber Pyro)

LYU1805

WONG YUK, 1155074616

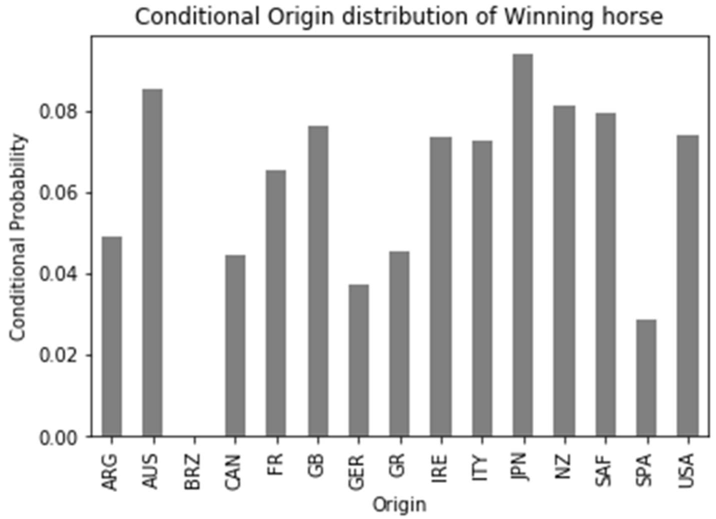
Introduction - Horse Racing

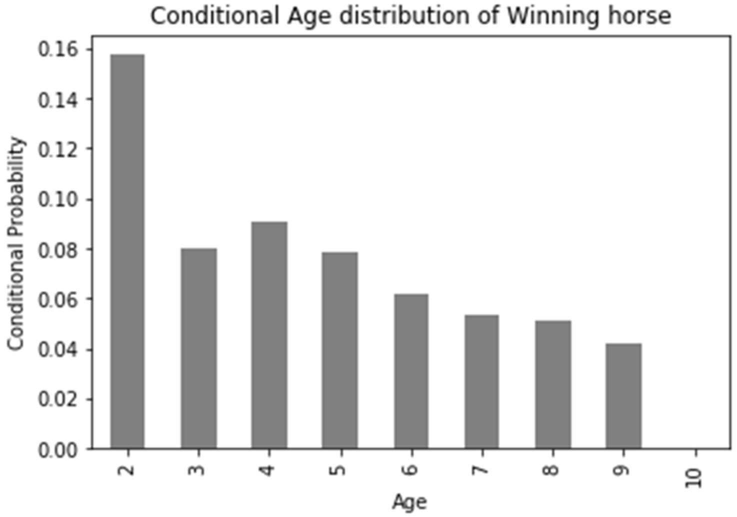
- Sport of running horse
  - The fastest wins
- Popular entertainment
  - Tens of thousands spectators
- Big business
  - >HK$1 Billion[1] at every meeting!

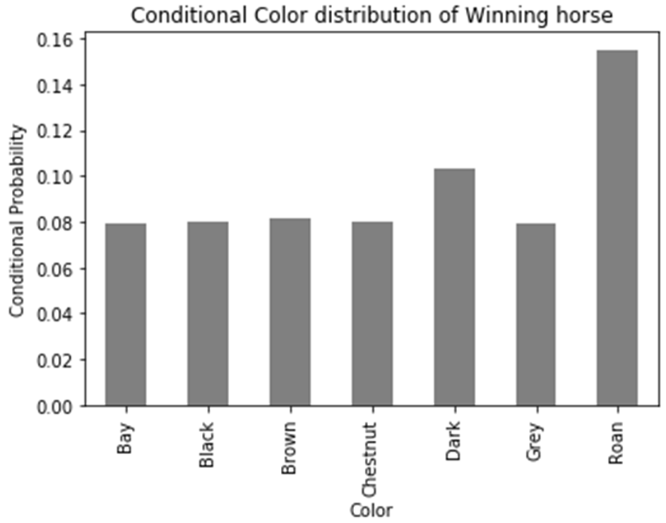
Figure and 1: South China Morning Post https://www.scmp.com/sport/racing/article/2146600/tsunami-illegal-betting-has-arrived-hong-kong-jockey-club-warns-it

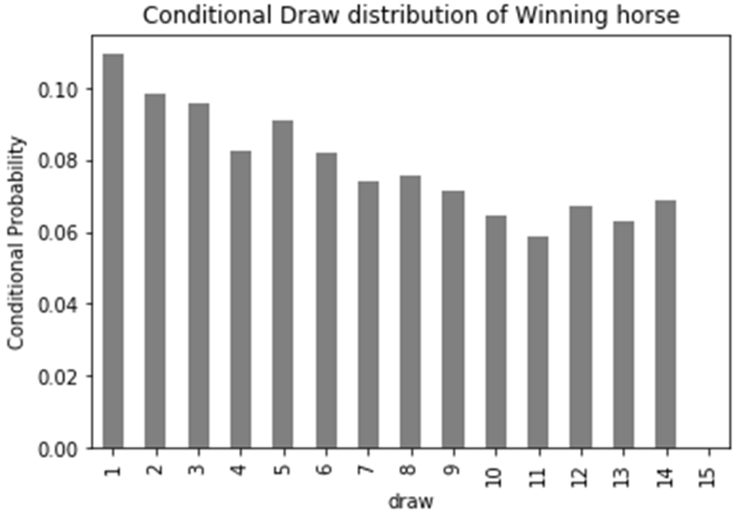
Introduction - Horse Racing

- Many factors
  - Horse Origin
  - Horse Age
  - Horse Color
  - Draw
  - …
- No single factor determines the winning horse


Conditional Origin distribution of Winning horse
Conditional Probability
0.00
0.02
0.04
0.06
0.08
ARG
AUS
BRZ
CAN
FR
GB
GER
GR
IRE
ITY
JPN
NZ
SAF
SPA
USA
Origin


Conditional Age distribution of Winning horse
Conditional Probability
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
2
3
4
5
6
7
8
9
10
Age


Conditional Color distribution of Winning horse
Conditional Probability
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Bay
Black
Brown
Chestnut
Dark
Grey
Roan
Color


Conditional Draw distribution of Winning horse
Conditional Probability
0.00
0.02
0.04
0.06
0.08
0.10
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
draw

Single–race Pool	Dividend Qualification
Win	1^{st} in a race
Place	1^{st}, 2^{nd} or 3^{rd} in a race, or 1st or 2^{nd} in a race of 4 to 6 declared starters
Quinella	1^{st} and 2^{nd} in any order in a race
Quinella Place	Any two of the first three placed horses in any order in a race
3 Pick 1 (Composite Win) **Winning Trainer (Composite Win)** **Winning Region (Composite Win)**	Composite containing the 1^{st} horse in a race
Tierce	1^{st} , 2^{nd} and 3^{rd} in correct order in a race
Trio	1^{st}, 2^{nd} and 3^{rd} in any order in a race
First 4	1^{st}, 2^{nd} , 3^{rd} and 4^{th} in any order in a race
Quartet	1^{st}, 2^{nd} , 3^{rd} and 4^{th} in correct order in a race

Multi–race Pool	Dividend Qualification
Double	1^{st} in each of the two nominated races
	Consolation : 1^{st} in 1^{st} nominated race and 2^{nd} in 2^{nd} nominated race
Treble	1^{st} in each of the three nominated races
	Consolation : 1^{st} in the first two Legs and 2^{nd} in 3^{rd} Leg of the three nominated races

Jackpot Pool	Dividend Qualification
Double Trio	1^{st}, 2^{nd} and 3^{rd} in any order in each of the two nominated races
Triple Trio	1^{st}, 2^{nd} and 3^{rd} in any order in each of the three nominated races
	Consolation : Select correctly the 1^{st}, 2^{nd} and 3^{rd} horses in any order in the first two Legs of the three nominated races
Six Up	1^{st} or 2^{nd} in each of the six nominated races
	Six Win Bonus : 1^{st} in each of the six nominated races

- We will only focus on Win bet

Introduction

Related Works

Background

Model

Data

Results

Related Works – Horse Racing Prediction

- Bolton and Chapman used a 20-variable multinomial logit model to 2000 Hong Kong races
  - Achieved net return in excess of 20%
- Chung et al. utilized Support-Vector-Machines to 2691 Hong Kong races
  - Achieved 840,164.1% return
- Can we achieve the same with neural networks?

Related Works – Horse Racing Prediction with Neural Networks

- Cheng and Lau used deep neural network model to regress running time on 11074 races
  - Results in loss of over 20% without confidence threshold, and gain net profit of 30% with threshold
- Liu and Wang also used deep neural network to regress running time on 5029 races
  - Results in loss of 25.78% on all races, and earn 17.45% on specific race classes
- In the 1st term, we have used Bayesian neural networks to predict horse place on 5740 races
  - Results in loss of 20.09% on all races, and earn 39.77% on specific race classes

Introduction

Related Works

Background

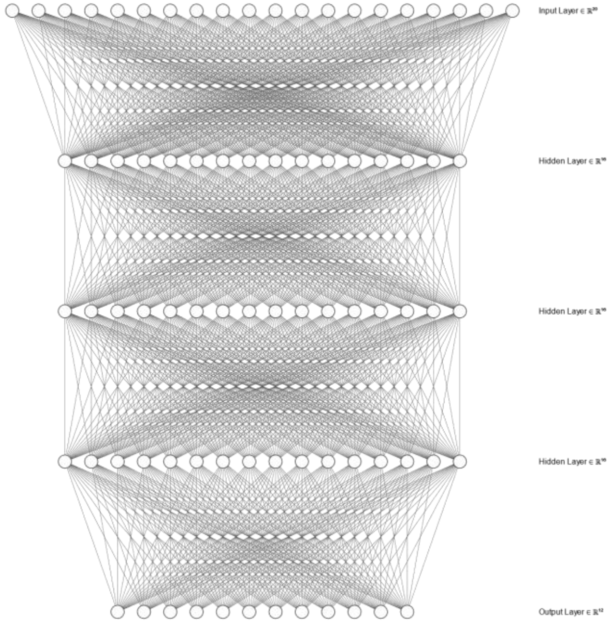
Model

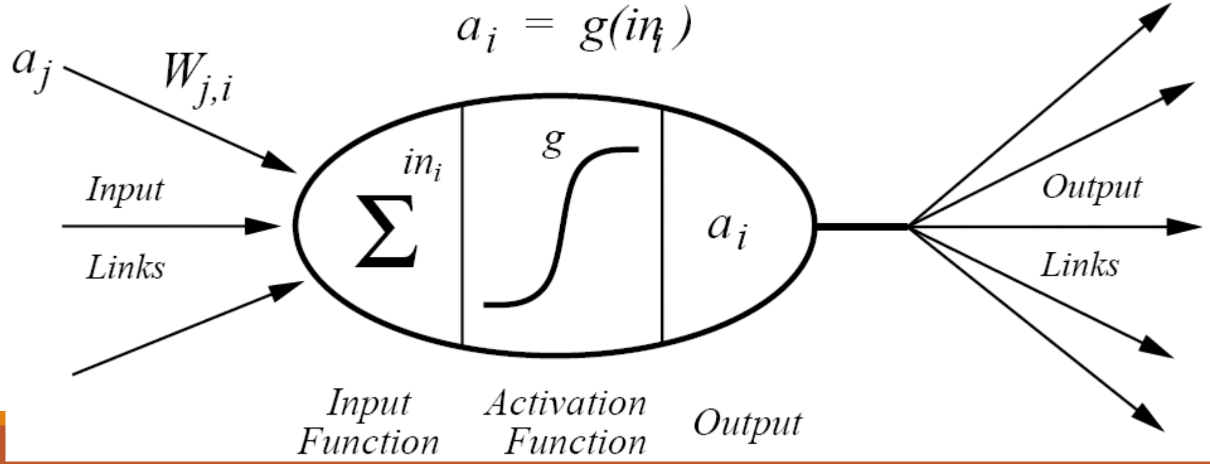
Data

Results

Artificial Neural Networks

- Collection of connected artificial neurons
- Each artificial neuron has
  - A linear component that compute weighted sum of input values
  - A nonlinear component serving as activation function
- Multiple layer neural networks can approximate any function





Training Neural Networks

z: our network parameters

x: our observed labels

F(x, z): the objective function, usually accuracy like BCE, MSE

1. Calculate F(x, z)
2. Calculate $\delta_z = \frac{\partial F}{\partial z}$
3. Update $z \leftarrow z - \alpha\delta_z$

Motivation

- Typical neural networks have deterministic output
  - Ideal for scenarios where label can be determined from data
- Our captured features may not be enough to determine race results
  - A model with deterministic output may not be sufficient
  - We wish to capture the uncertainty of observed results
- Instead of training a model with deterministic output,
- Train a model that outputs according to winning probability of each horses

Probabilistic Programming

- Probabilistic programs have the following two properties:
  - The ability to draw values at random from distributions
    - Sample $z \sim p(z)$
  - The ability to condition values of variable in a program via observations
    - Given observations x, infer $p(z|x)$
- Usually used to carry out probabilistic inference

Probabilistic Programming Languages

- Programming languages that provides probabilistic primitives
  - Most common programming languages already have random sampling
  - Especially for conditioning random distributions
  - Can be extend from a basic language, can be self-contained
  - Pyro, a language extended from Python and PyTorch
  - Stan, self-contained language
- Usually used to carry out probabilistic inference

Bayesian Inference

- A kind of probabilistic inference
- Given:
  - Prior probability $p(z)$: what we previously know about the model
  - Observed label x
- We apply Bayes' rule

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

to given a conditioned model

Bayesian Inference

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

- $p(z)$ is given
- $p(x|z)$ can be obtained from forward execution of the program
- $p(x)$ is not known, even if we rewrite it to

$$p(x) = \int p(x|z)p(z)dz$$

it is still very expensive or even intractable to compute

Bayesian Inference Algorithms

- Two main family of algorithms
  - Markov Chain Monte Carlo algorithms
  - Variational Inference

Markov Chain Monte Carlo (MCMC)

- We construct a Markov Chain and obtain the equilibrium distribution as follows:
  1. Sample z_0 from the initial distribution $q(z_0)$
  2. Propose a new sample z_i'
  3. Accept or reject probabilistically using the $q(z_i|z_{i-1})$ and $p(x|z)$
  4. If the proposal is accepted, return to step 2 with z_i
  5. If the proposal is rejected, return to step 2 with z_{i-1}
  6. After specified number of iterations, return all z_0 to z_{n-1}
- The main difference between MCMC algorithms is in step 2 and step 3
- Metropolis algorithm:
  - Propose new sample by a normal distribution with mean at current z and tunable standard deviation
  - Probability of acceptance: $p = \frac{p(z'|x)}{p(z|x)} = \frac{p(x|z')p(z')}{p(x|z)p(z)}$

Variational Inference

- We introduce a parameterized variation distribution $q(z)$ to approximate the posterior $p(z|x)$
- Optimize the variation distribution to be close to actual posterior distribution
- Objective function: Kullback-Leibler divergence, difference of 2 distributions

$$\mathrm{KL}\big(q(z)||p(z|x)\big) = \mathrm{E}_q\left[\log \frac{q(z)}{p(z|x)}\right]$$
$$= -\big(\mathrm{E}_q[\log p(x,z)] - \mathrm{E}_q[\log q(z)]\big) + \log p(x)$$

- $p(x)$ is independent of $q(z)$, so we remove it to obtain
- Evidence Lower Bound $\mathrm{ELBO} = \mathrm{E}_q[\log p(x,z)] - \mathrm{E}_q[\log q(z)]$

Variational Inference

- Maximizing ELBO can be done via gradient ascent
- Let ϕ be the parameters that defines distribution $q(z)$, and α be the learning rate
- Then we can do variational inference as follows:
  1. Calculate ELBO(x, z, ϕ)
  2. Calculate $\delta_\phi = \frac{\partial \text{ELBO}}{\partial \phi}$
  3. Update $\phi \leftarrow \phi + \alpha\delta_\phi$

Deep Probabilistic Programming

- Combines neural networks with probabilistic programming
- Most commonly used: Bayesian neural networks

Bayesian Neural Networks

- Condition neural networks by Bayesian inference
- Train a distribution instead of a single value for each parameter
- Instead of parameter z,
- Becomes distribution of parameter $p(z)$
  - If we use normal distribution, then it becomes 2 parameters μ, ρ

Bayesian Neural Networks

NN trained by gradient descent

z: our network parameters

x: our observed labels

Let F(x, z) denote the objective function

1. Calculate F(x, z)
2. Calculate $\delta_z = \frac{\partial F}{\partial z}$
3. Update $\mathrm{z} \leftarrow \mathrm{z} - \alpha\delta_z$

NN trained by variational inference

Let $p(x) = \mathrm{Normal}(\mu, \log(1 + \mathrm{e}^{\rho}))$

1. Sample ϵ from $\mathrm{Normal}(0,1)$
2. Sampled $z = \mu + \epsilon \log(1 + \mathrm{e}^{\rho})$
3. Calculate ELBO(x, z, μ, ρ)
4. Calculate $\delta_\mu = \frac{\partial ELBO}{\partial z} + \frac{\partial ELBO}{\partial \mu}$
5. Calculate $\delta_\rho = \frac{\partial ELBO}{\partial z} \frac{\epsilon}{1+\mathrm{e}^{-\rho}} + \frac{\partial ELBO}{\partial \rho}$
6. Update $\mu \leftarrow \mu + \alpha\delta_\mu, \ \rho \leftarrow \rho + \alpha\delta_\rho$

Introduction

Related Works

Background

Model

Data

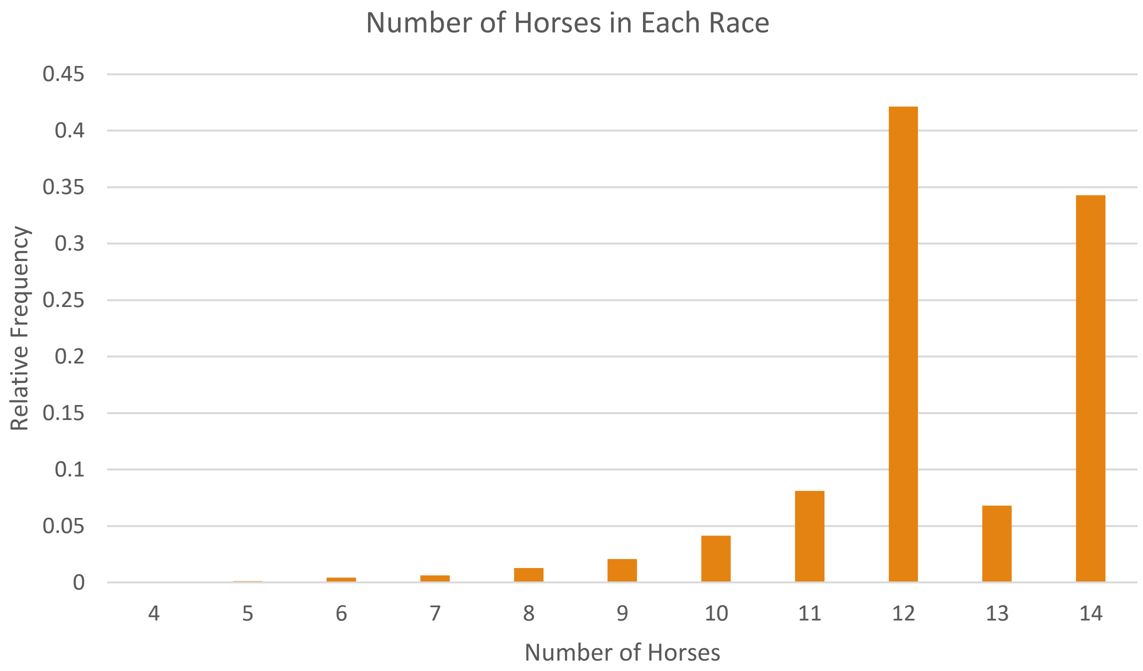
Results

Race Representation

- Single horse representations
  - Finishing time regression – Errors accumulate
  - Win/lose binary classification – Uneven label
  - Place prediction – Inconsistent number of places for each race
- Multiple horse representations
  - Finishing time regression – Difficult to choose activation function
  - Winning horse prediction – Intuitive probabilistic output
  - Place prediction – Need two dimension output

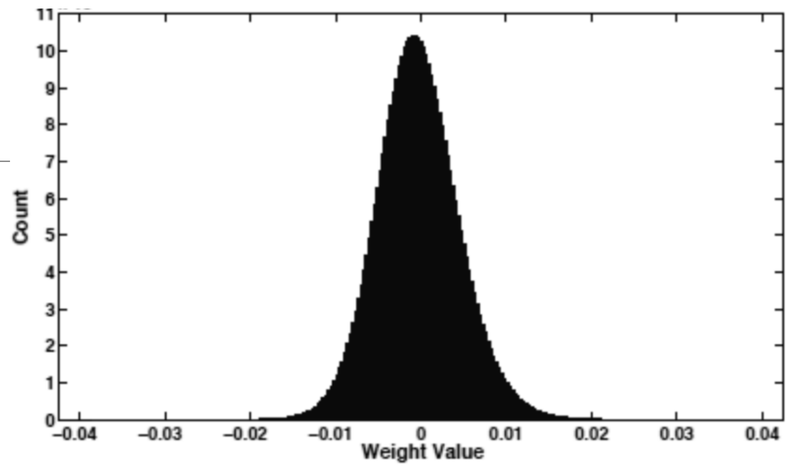
Race Representation

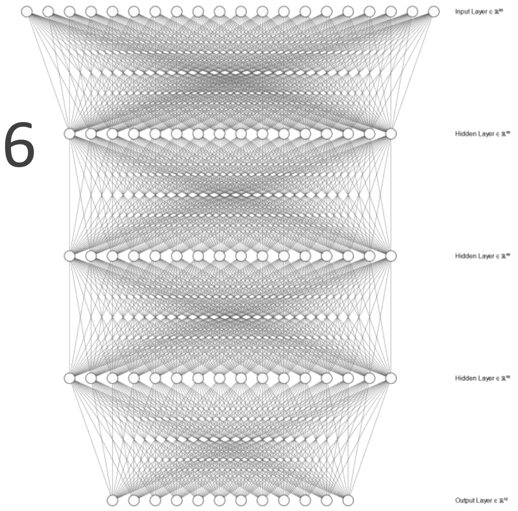
- But different races have different number of horse!
- >75% races with 12 or 14 horses
- Two models for 12 and 14 horses



Model Design

- Assume normal distribution for weight and bias
- Four layer neural network with ReLU activation
  - The best model from last semester
  - Tested 1 to 4 layer and 4 performs best
  - Test different number of neurons in each layer from 16 to 256
- Output of network is the win probability of each horse





Features from all horse in a race → Model → Win probability of each horse

Introduction

Related Works

Background

Model

Data

Results

Data

- 8 years of horse racing records of Hong Kong from 2011 to 2018
- 77652 records from 6251 races
- Comparison:
  - Bolton and Chapman: 2000 races
  - Chung et al: 2691 races
  - Cheng and Lau: 11074 races, Liu and Wang: 5029 races
- 2011 to 2017 data for training, 2018 data for testing
- Same period of training data as Liu and Wang
  - Allows direct comparison
- 5461 races for training, 790 races for testing

Features

- We keep the best set of features selected from last semester

Location	Class	Distance	Course	Going	Jockey	Trainer	Draw	Winodds
Actual weight	Horse weight	Horse origin	Horse age	Horse color	Horse sex	Old place	Weightdiff	

- And also add some weather features (to replace month)

Temperature	Weather	Wind speed	Wind direction	Humidity	Moon phase	Day / Night

Features

- All features (24 features)

Location	Class	Distance	Course	Going	Jockey	Trainer	Draw	Winodds
Actual weight	Horse weight	Horse origin	Horse age	Horse color	Horse sex	Old place	Weightdiff	
Temperature	Weather	Wind speed	Wind direction	Humidity	Moon phase	Day / Night		

- Without Winodds (23 features)

Location	Class	Distance	Course	Going	Jockey	Trainer	Draw	~~Winodds~~
Actual weight	Horse weight	Horse origin	Horse age	Horse color	Horse sex	Old place	Weightdiff	
Temperature	Weather	Wind speed	Wind direction	Humidity	Moon phase	Day / Night		

- Without weather features (17 features)

Location	Class	Distance	Course	Going	Jockey	Trainer	Draw	Winodds
Actual weight	Horse weight	Horse origin	Horse age	Horse color	Horse sex	Old place	Weightdiff	

Data Preprocessing

- Normalization
  - Zero mean, unit variance

$$\hat{X} = \frac{X - mean(X)}{std(X)}$$

- One-hot encoding
  - Convert categorical data to numerical input

Item	Category
1	Apple
2	Orange
3	Banana

→

Item	Is Apple	Is Orange	Is Banana
1	1	0	0
2	0	1	0
3	0	0	1

Data Augmentation

- Crop 14 and 13 horse races to 12 horse races

Horse 1	Horse 2	Horse 3	Horse 4	Horse 5	Horse 6	Horse 7	Horse 8	Horse 9	Horse10	Horse11	Horse12	~~Horse13~~	~~Horse14~~

- Permutate individual horse's data within a race
  - Prevents biases on input position

Horse 1	Horse 2	Horse 3	Horse 4	Horse 5	Horse 6	Horse 7	Horse 8	Horse 9	Horse 10	Horse 11	Horse 12

↓

Horse 12	Horse 11	Horse 10	Horse 9	Horse 8	Horse 7	Horse 6	Horse 5	Horse 4	Horse 3	Horse 2	Horse 1

Introduction

Related Works

Background

Model

Data

Results

Implementation

- Implemented on Python 3.7, PyTorch 1.0.1, Pyro 0.3.1
- Bayesian infer model by variational inference
  - Better support in Pyro than Markov chain Monte Carlo
  - Markov chain Monte Carlo has some memory issues[1] in Pyro, currently still open and unsolved
  - Similarity to typical deep learning
- Trained for 100000 epochs with Adam optimizer at initial learning rate of $1e^{-4}$
- Extracted mean from trained variational distribution for testing
  - Most likely model

1: https://github.com/pyro-ppl/pyro/issues/1787

Results - Metrics

- We evaluate the performance of our model on two metrics

1. Accuracy
2. Net gain

Results – Even Betting with 12 horses

- Only model without weather features can make a profit

12-horse Model (Tested with 341 12-horse races)					
Feature Set	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit?
Public Intelligence	N/A	22.87	–114.5	–33.58	No
All Features	16	7.62	–136.0	–39.88	No
	32	17.01	–88.9	–26.07	No
	64	22.58	–26.9	–7.89	No
	128	17.60	–31.7	–9.30	No
	256	18.18	–80.6	–23.64	No
Without “winodds” Feature	16	8.80	–185.2	–54.31	No
	32	8.80	–185.2	–54.31	No
	64	20.82	–15.8	–4.63	No
	128	16.42	–67.6	–19.82	No
	256	17.60	–43.1	–12.64	No
Without Weather Features	16	9.68	–47.6	–13.96	No
	32	22.29	25.7	7.54	Yes
	64	19.35	–82.7	–24.25	No
	128	17.30	–73.5	–21.55	No
	256	17.30	–87.3	–25.60	No

Results – Even Betting with 14 horses

- All feature sets can make a profit

14-horse Model (Tested with 203 14-horse races)					
Feature Set	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit
Public Intelligence	N/A	27.59	−36.9	−18.18	No
All Features	16	5.91	−117.6	−57.93	No
	32	16.75	−41.6	−20.49	No
	64	14.29	−35.3	−17.39	No
	128	22.66	29.3	14.43	Yes
	256	17.24	−21.1	−10.39	No
Without “winodds” Feature	16	9.85	−117.7	−57.98	No
	32	15.76	−44.0	−21.67	No
	64	18.23	−27.5	−13.55	No
	128	23.15	15.4	7.59	Yes
	256	17.73	−38.6	−19.01	No
Without Weather Features	16	5.91	−117.6	−57.93	No
	32	20.20	−23.8	−11.72	No
	64	23.15	7.5	3.69	Yes
	128	15.76	−58.2	−28.67	No
	256	17.24	−49.7	−24.48	No

Results – Even Betting

- 14 horse model needs more neurons per layer to perform well

12-horse Model (Tested with 341 12-horse races)						14-horse Model (Tested with 203 14-horse races)					
Feature Set	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit?	**Feature Set**	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit
Public Intelligence	N/A	22.87	–114.5	–33.58	No	**Public Intelligence**	N/A	27.59	–36.9	–18.18	No
All Features	16	7.62	–136.0	–39.88	No	**All Features**	16	5.91	–117.6	–57.93	No
	32	17.01	–88.9	–26.07	No		32	16.75	–41.6	–20.49	No
	64	22.58	–26.9	–7.89	No		64	14.29	–35.3	–17.39	No
	128	17.60	–31.7	–9.30	No		128	22.66	29.3	14.43	Yes
	256	18.18	–80.6	–23.64	No		256	17.24	–21.1	–10.39	No
Without "winodds" Feature	16	8.80	–185.2	–54.31	No	**Without "winodds" Feature**	16	9.85	–117.7	–57.98	No
	32	8.80	–185.2	–54.31	No		32	15.76	–44.0	–21.67	No
	64	20.82	–15.8	–4.63	No		64	18.23	–27.5	–13.55	No
	128	16.42	–67.6	–19.82	No		128	23.15	15.4	7.59	Yes
	256	17.60	–43.1	–12.64	No		256	17.73	–38.6	–19.01	No
Without Weather Features	16	9.68	–47.6	–13.96	No	**Without Weather Features**	16	5.91	–117.6	–57.93	No
	32	22.29	25.7	7.54	Yes		32	20.20	–23.8	–11.72	No
	64	19.35	–82.7	–24.25	No		64	23.15	7.5	3.69	Yes
	128	17.30	–73.5	–21.55	No		128	15.76	–58.2	–28.67	No
	256	17.30	–87.3	–25.60	No		256	17.24	–49.7	–24.48	No

Results – Kelly Betting

- In reality, people bet different amount under different confidence
- Let p be win probability, b be return per bet, A be the total asset
- Kelly bet

$$f = A \times \frac{p(b+1) - 1}{b}$$

- Optimal wealth increase in the long run

Results – Kelly Betting with 12 horses

- Total loss in all configurations

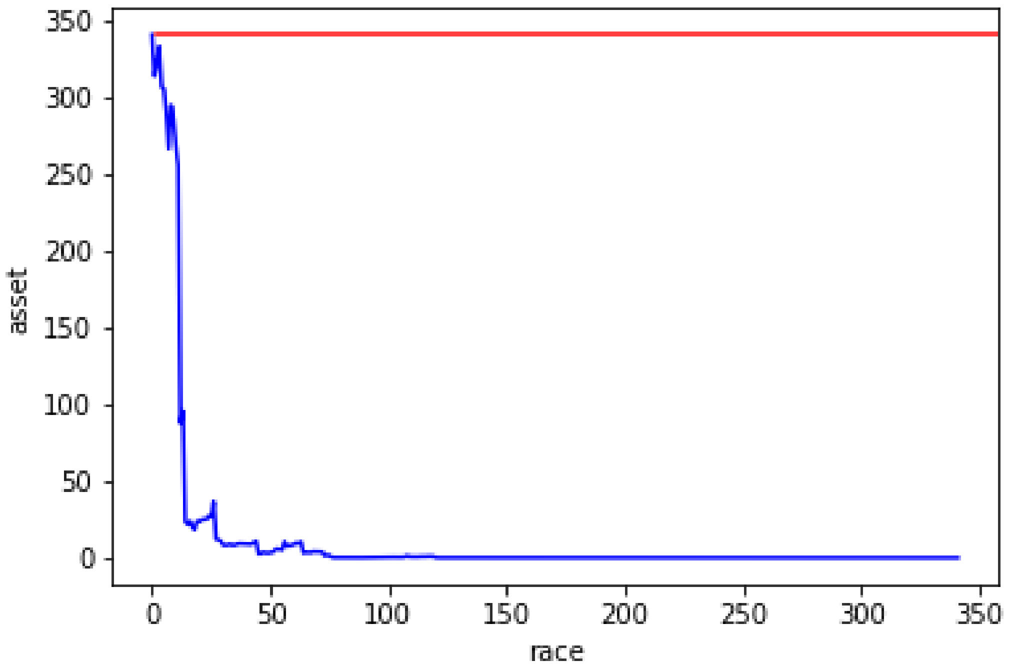
12-horse Model (Tested with 341 12-horse races)					
Feature Set	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit?
Public Intelligence	N/A	22.87	–114.5	–33.58	No
All Features	16	7.62	–313.6	–91.95	No
	32	17.01	–340.6	–99.90	No
	64	22.58	–336.3	–98.61	No
	128	17.60	–340.1	–99.75	No
	256	18.18	–315.3	–92.50	No
Without "winodds" Feature	16	8.80	–341.0	–100	No
	32	8.80	–341.0	–100	No
	64	20.82	–341.0	–100	No
	128	16.42	–340.2	–99.77	No
	256	17.60	–341.0	–100	No
Without Weather Features	16	9.68	–315.7	–92.57	No
	32	22.29	–314.9	–92.34	No
	64	19.35	–332.4	–97.48	No
	128	17.30	–338.6	–99.30	No
	256	17.30	–341.0	–100	No

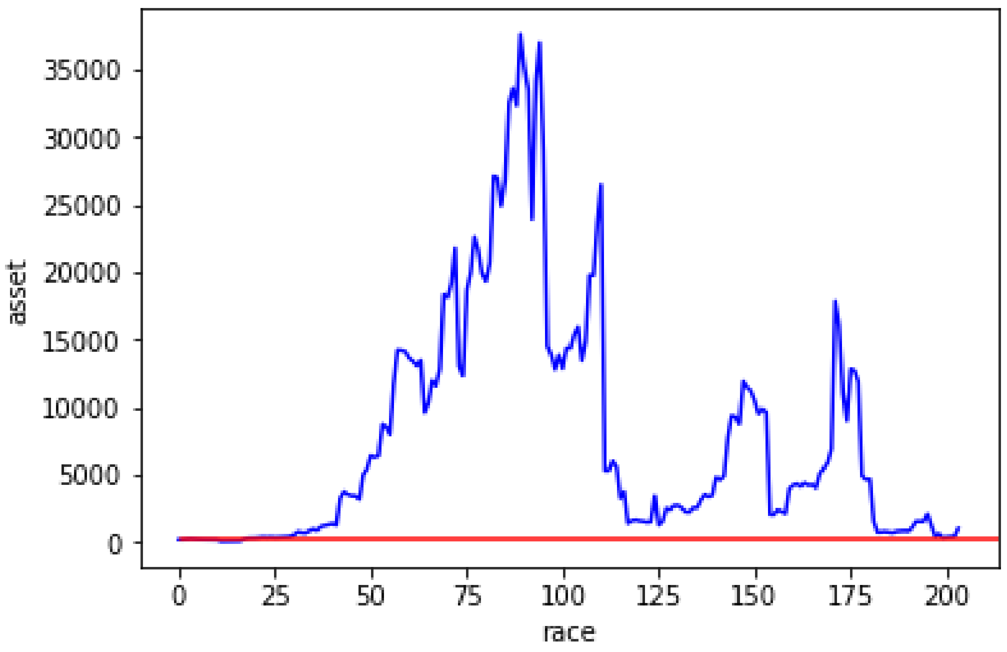
Results – Kelly Betting with 14 horses

- Total loss in all configurations, except all features with 64 neurons

14-horse Model (Tested with 203 14-horse races)					
Feature Set	Neurons	Accuracy (%)	Net Return	Return/Bet (%)	Profit
Public Intelligence	N/A	27.59	−36.9	−18.18	No
All Features	16	5.91	−203.0	−99.98	No
	32	16.75	−202.7	−99.86	No
	64	14.29	818.5	403.22	Yes
	128	22.66	−201.6	−99.29	No
	256	17.24	−202.6	−99.79	No
Without "winodds" Feature	16	9.85	−203.0	−99.98	No
	32	15.76	−203.0	−99.99	No
	64	18.23	−203.0	−100	No
	128	23.15	−203.0	−100	No
	256	17.73	−203.0	−99.98	No
Without Weather Features	16	5.91	−203.0	−99.98	No
	32	20.20	−202.5	−99.75	No
	64	23.15	−202.9	−99.95	No
	128	15.76	−202.8	−99.89	No
	256	17.24	−202.7	−99.86	No

Results – Kelly Betting





Discussions

- Even betting with 12 horses is profitable without weather features
  - 7.54% net gain without weather features
- Even betting with 14 horses is profitable regardless of feature set
  - 14.43% net gain with all features
- Kelly betting results in total loss
  - Kelly betting bets based on confidence, our model is too confident
  - Another problem: Kelly betting assumes infinitesimal bets are possible

Comparison with Related Works

- We achieved gain of 7.54% for 12 horses and 14.43% for 14 horses
- In comparison:
  - Cheng and Lau: loss of over 20% without confidence threshold, and gain net profit of 30% with threshold
  - Liu and Wang: loss of 25.78% on all races, and earn 17.45% on specific race classes
  - Our past term: loss of 20.09% on all races, and earn 39.77% on specific race classes
- We can achieve net gain without selecting additional criteria after testing
  - Avoids potential information leakage

Conclusion

- Applied new method of Bayesian neural network to horse racing
- Moderate accuracy: 22% to 23%
  - Perhaps due to features unable to fully capture horse racing
  - Insufficient data
- Yet effective in predicting for Win bet
  - Our model predict win of those not anticipated by the public (Large Winodds)
  - Thus a net gain without exceedingly high accuracy
- Achieved net gain of 7.54% (12 horses) and 14.43% (14 horses)
  - Without additional selection criteria
- Currently overconfident in the predicting winning probability
  - Not able to effectively apply Kelly betting yet

Future Work

- Extend model to accommodate all number of horses
- Compare variational inference to Markov chain Monte Carlo

Thank you!