# Using Neural Networks for Time-Series Prediction

Joe Jevnik

November 27, 2017

PyData NYC 2017

# Outline

1. Introduce problem
2. Phrase problem as an ML problem
3. Collect and apply data
4. Select features
5. Train the model
6. Improve the model
7. Tips

# Who am I?

Quantopian

- Works with storage and manipulation of time series data
- Integrates third-party data sets

# The Problem

# Osu!

# Elements


**Figure 1:** A hit circle

# Elements


**Figure 1:** A hit circle

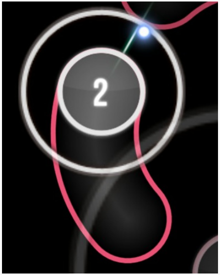



**Figure 2:** A slider

# Elements


**Figure 1:** A hit circle

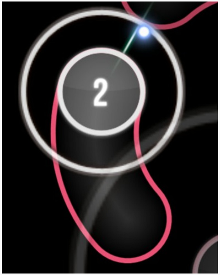



**Figure 2:** A slider

**Figure 3:** Mouse cursor

# Osu! (cont.)

## Scoring

300 points if on the beat

# Osu! (cont.)

## Scoring

300 points if on the beat

100 points if slightly off the beat

# Osu! (cont.)

## Scoring

300 points if on the beat

100 points if slightly off the beat

# Osu! (cont.)

## Scoring

300 points if on the beat

100 points if slightly off the beat


50 points if really off the beat

# Osu! (cont.)

## Scoring

300 points if on the beat

100 points if slightly off the beat


50 points if really off the beat

# Osu! (cont.)

## Scoring

300 points if on the beat

100 points if slightly off the beat


50 points if really off the beat


0 points for missing entirely

# Osu! (cont.)

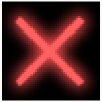
## Scoring

300 points if on the beat

100 points if slightly off the beat


50 points if really off the beat


0 points for missing entirely



# Osu! (cont.)

**Sample**

```
$ mpv videos/osu-example.avi
```

# Problem

Improve my rank quickly

# Problem

Improve my rank quickly

Many new songs a week

# Problem

Improve my rank quickly

Many new songs a week

Particular playstyle

# Problem

Improve my rank quickly

Many new songs a week

Particular playstyle

Arbitrage opportunity?

# Phrasing the Problem

# Problem

**Predict my score on a beatmap**

# Problem

**Predict my score on a beatmap**

Need to compute accuracy %

# Problem

**Predict my score on a beatmap**

- Need to compute accuracy %
- Temporal Accuracy

# Problem

**Predict my score on a beatmap**

- Need to compute accuracy %
- Temporal Accuracy
- Aim Accuracy

# Machine Learning Models

**Classifiers**
Label a sample as a member of one of a finite set of classes.

**Regressors**
Approximate a numerical function.

# LSTM Models

Order dependent data

# LSTM Models

Order dependent data

Sequence of observations

# LSTM Models

Order dependent data

Sequence of observations

Uses windows of time-sorted observations

# Our Problem

For each hit-object, predict..

# Our Problem

For each hit-object, predict..

**Classifier**
A label.

- 300
- 100
- 50
- 0 (miss)

# Our Problem

For each hit-object, predict..

**Classifier**
A label.

- 300
- 100
- 50
- 0 (miss)

**Regressor**
A numeric error metric.

1. Aim Error ((x, y) error)
2. Accuracy Error (punctuality)

# Data Collection

# Beatmaps

Hit objects in (x, y, time) space.

# Beatmaps

Hit objects in (x, y, time) space.

Circle Size (CS)

## Beatmaps

Hit objects in (x, y, time) space.

Circle Size (CS)

Approach Rate (AR)

# Beatmaps

Hit objects in (x, y, time) space.

Circle Size (CS)

Approach Rate (AR)

Overall Difficulty (OD) (score thresholds)

# Raw Data

**Beatmap**

```
[HitObjects]
103,272,52926,6,0,L|111:176,1,67.5000025749208
93,95,53279,1,2,0:3:0:0:
194,131,53455,2,0,B|263:160|264:100|337:135,1,135.00000514
437,204,53985,2,0,L|432:286,1,67.5000025749208
394,105,54338,6,0,L|399:17,1,67.5000025749208
286,62,54690,1,2,0:3:0:0:
177,54,54867,2,0,B|110:74|110:30|41:53,1,135.000005149842,
70,213,55396,2,0,L|77:132,1,67.5000025749208
161,215,55749,6,0,P|175:273|247:314,1,135.000005149842,0|2
341,286,56279,1,0,0:0:0:0:
308,183,56455,2,0,P|268:201|245:238,1,67.5000025749208,0|2
```

# Mods

- Hard Rock (HR)

# Mods

- Hard Rock (HR)
- Double Time (DT)

# Mods

- Hard Rock (HR)
- Double Time (DT)
- Hidden (HD)

# Mods

- Hard Rock (HR)
- Double Time (DT)
- Hidden (HD)
- etc...

# Replays

**Time series of...**

# Replays

**Time series of...**

Cursor location

# Replays

**Time series of...**

- Cursor location
- Keyboard state

# Replays

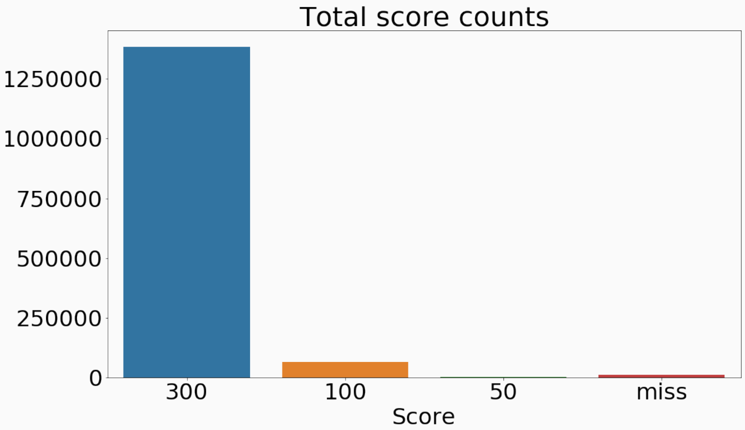
I had about seven years of replays laying around!

# Understanding our data

## Accuracy thresholds (milliseconds)

| OD | 300 | 100 | 50 |
|---|---|---|---|
| 1 | 73.5 | 131.5 | 189.5 |
| 2 | 67.5 | 123.5 | 179.5 |
| 3 | 61.5 | 115.5 | 169.5 |
| 4 | 55.5 | 107.5 | 159.5 |
| 5 | 49.5 | 99.5 | 149.5 |
| 6 | 43.5 | 91.5 | 139.5 |
| 7 | 37.5 | 83.5 | 129.5 |
| 8 | 31.5 | 75.5 | 119.5 |
| 9 | 25.5 | 67.5 | 109.5 |
| 10 | 19.5 | 59.5 | 99.5 |

# Understanding our data (cont.)



# Prediction Targets

**Joining Data**

# Prediction Targets

**Joining Data**

Find all clicks by taking times where key state changes

# Prediction Targets

**Joining Data**

Find all clicks by taking times where key state changes

Match click with the nearest hit object (ignores hit locking!)

# Prediction Targets

**Accuracy Error**

Absolute difference in time.

# Prediction Targets

**Accuracy Error**

Absolute difference in time.

Comparable across different OD

# Prediction Targets

**Aim Error**

Euclidean distance between click and center of circle.

# Prediction Targets

**Aim Error**

Euclidean distance between click and center of circle.

Comparable across different CS

# Feature Selection

# What is a feature?

Numeric inputs to the ML model

# What is a feature?

Numeric inputs to the ML model

- What are we observing

# What is a feature?

Numeric inputs to the ML model

- What are we observing
- Focus the model on aspects of the data

# What is a feature?

Numeric inputs to the ML model

- What are we observing
- Focus the model on aspects of the data
- Chance to use domain knowledge

**Beatmap**

```
[HitObjects]
103,272,52926,6,0,L|111:176,1,67.5000025749208
93,95,53279,1,2,0:3:0:0:
194,131,53455,2,0,B|263:160|264:100|337:135,1,135.00000514
437,204,53985,2,0,L|432:286,1,67.5000025749208
394,105,54338,6,0,L|399:17,1,67.5000025749208
286,62,54690,1,2,0:3:0:0:
177,54,54867,2,0,B|110:74|110:30|41:53,1,135.000005149842,
70,213,55396,2,0,L|77:132,1,67.5000025749208
161,215,55749,6,0,P|175:273|247:314,1,135.000005149842,0|2
341,286,56279,1,0,0:0:0:0:
308,183,56455,2,0,P|268:201|245:238,1,67.5000025749208,0|2
```

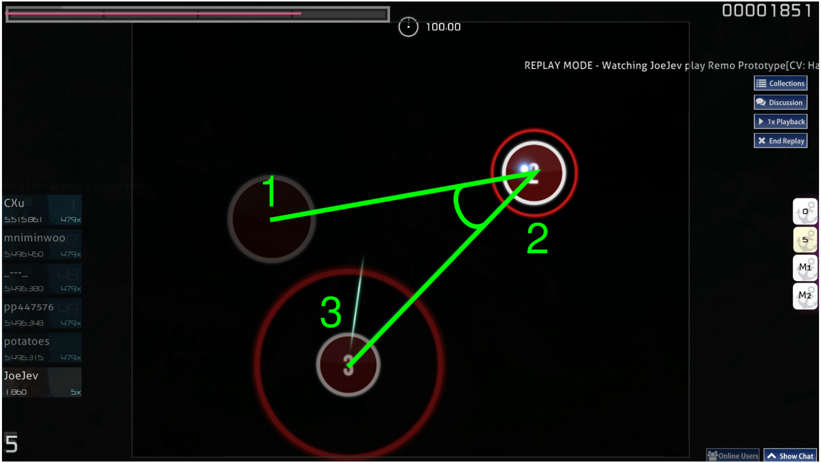
# Simple Features

`absolute_x`

`absolute_y`

`absolute_time`

# Domain Specific Features



# Windows

| time | x | y |
|---:|---:|---:|
| 00:37.366 | 372 | 94 |
| 00:37.763 | 447 | 205 |
| 00:38.027 | 217 | 299 |
| 00:38.291 | 229 | 171 |
| 00:38.424 | 274 | 358 |
| 00:38.688 | 149 | 221 |
| 00:38.952 | 330 | 186 |
| 00:39.217 | 233 | 127 |
| 00:39.481 | 233 | 127 |
| 00:39.613 | 198 | 303 |

# Windows

| time | x | y |
|---|---|---|
| 00:37.366 | 372 | 94 |
| 00:37.763 | 447 | 205 |
| 00:38.027 | 217 | 299 |
| 00:38.291 | 229 | 171 |
| 00:38.424 | 274 | 358 |
| 00:38.688 | 149 | 221 |
| 00:38.952 | 330 | 186 |
| 00:39.217 | 233 | 127 |
| 00:39.481 | 233 | 127 |
| 00:39.613 | 198 | 303 |

# Windows

| time | x | y |
|---|---|---|
| 00:37.366 | 372 | 94 |
| 00:37.763 | 447 | 205 |
| 00:38.027 | 217 | 299 |
| 00:38.291 | 229 | 171 |
| 00:38.424 | 274 | 358 |
| 00:38.688 | 149 | 221 |
| 00:38.952 | 330 | 186 |
| 00:39.217 | 233 | 127 |
| 00:39.481 | 233 | 127 |
| 00:39.613 | 198 | 303 |

# Windows

| time | x | y | relative x | relative y |
|---|---|---|---|---|
| 00:37.366 | 372 | 94 | - | - |
| 00:37.763 | 447 | 205 | 298 | -16 |
| 00:38.027 | 217 | 299 | 68 | 78 |
| 00:38.291 | 229 | 171 | 80 | -50 |
| 00:38.424 | 274 | 358 | 125 | 137 |
| 00:38.688 | 149 | 221 | 0 | 0 |
| 00:38.952 | 330 | 186 | 181 | -35 |
| 00:39.217 | 233 | 127 | 84 | -94 |
| 00:39.481 | 233 | 127 | - | - |
| 00:39.613 | 198 | 303 | - | - |

## Windows

| time | x | y | relative x | relative y |
|---:|---:|---:|---:|---:|
| 00:37.366 | 372 | 94 | - | - |
| 00:37.763 | 447 | 205 | - | - |
| 00:38.027 | 217 | 299 | -113 | 113 |
| 00:38.291 | 229 | 171 | -101 | -15 |
| 00:38.424 | 274 | 358 | -56 | 172 |
| 00:38.688 | 149 | 221 | -181 | 35 |
| 00:38.952 | 330 | 186 | 0 | 0 |
| 00:39.217 | 233 | 127 | -97 | -59 |
| 00:39.481 | 233 | 127 | -97 | -59 |
| 00:39.613 | 198 | 303 | - | - |

## Windows

| time | x | y | relative x | relative y |
|---|---|---|---|---|
| 00:37.366 | 372 | 94 | - | - |
| 00:37.763 | 447 | 205 | - | - |
| 00:38.027 | 217 | 299 | - | - |
| 00:38.291 | 229 | 171 | -4 | 44 |
| 00:38.424 | 274 | 358 | 41 | 231 |
| 00:38.688 | 149 | 221 | -84 | 94 |
| 00:38.952 | 330 | 186 | 97 | 59 |
| 00:39.217 | 233 | 127 | 0 | 0 |
| 00:39.481 | 233 | 127 | 0 | 0 |
| 00:39.613 | 198 | 303 | -35 | 176 |

## Osu! Features

- absolute_x
- absolute_y
- absolute_time
- relative_x
- relative_y
- relative_time
- is_slider_tick
- approach_rate
- distance_from_previous
- distance_to_next
- pitch
- roll
- yaw

# Training

# Input Shapes

**Feature array shape**

( , , , )

## Input Shapes

**Feature array shape**

( , , number of features, )

# Input Shapes

**Feature array shape**

( , window length, number of features, )

# Input Shapes

**Feature array shape**

( number of windows, window length, number of features, )

# Input Shapes

**Feature array shape**

( number of windows, window length, number of features, )

**Label array shape**

( number of windows, )

( number of windows, )

# Keras

```
input_ = keras.layers.Input(
    shape=(window_length, len(features))
)
```

## Keras

```
input_ = keras.layers.Input(
    shape=(window_length, len(features))
)
lstm = keras.layers.LSTM(lstm_layer_size)(input_)
```

## Keras

```
input_ = keras.layers.Input(
    shape=(window_length, len(features))
)
lstm = keras.layers.LSTM(lstm_layer_size)(input_)
aim_error = keras.layers.Dense(
    1,
    activation='linear',
    name='aim_error',
)(lstm)
```

# Keras

```
input_ = keras.layers.Input(
    shape=(window_length, len(features))
)
lstm = keras.layers.LSTM(lstm_layer_size)(input_)
aim_error = keras.layers.Dense(
    1,
    activation='linear',
    name='aim_error',
)(lstm)
accuracy_error = keras.layers.Dense(
    1,
    activation='linear',
    name='accuracy_error',
)(lstm)
```

## Keras (cont.)

```
model = keras.models.Model(
    inputs=input_,
    outputs=[aim_error, accuracy_error],
)
```

# Keras (cont.)

```
model = keras.models.Model(
    inputs=input_,
    outputs=[aim_error, accuracy_error],
)

model.compile(
    loss='mse',
    optimizer='rmsprop',
)
```

# Keras (cont.)

```
# compute features, aim_error, accuracy_error
model.fit(
    features,
    {
        'aim_error': aim_error,
        'accuracy_error': accuracy_error,
    },
)
```

# Keras (cont.)

```python
# compute features, aim_error, accuracy_error
model.fit(
    features,
    {
        'aim_error': aim_error,
        'accuracy_error': accuracy_error,
    },
)

model.predict(features)
```

# How does it look?

# How does it look?

bad

# Feature Scaling

Sensitive to input ranges

# Feature Scaling

Sensitive to input ranges

```
(data - data.mean()) / data.std()
```

## Feature Scaling

Sensitive to input ranges

`(data - data.mean()) / data.std()`

Save the mean and std of the training data!

# Data (again)

```
features:
    absolute_x:
      mean: 256.93
      std:  581144.54
      min:  -26.25
      max:  42978020236964152.0
    absolute_y:
      mean: 188.67
      std:  96280.10
      min:  -12.20
      max:  10754663190171874.0
```

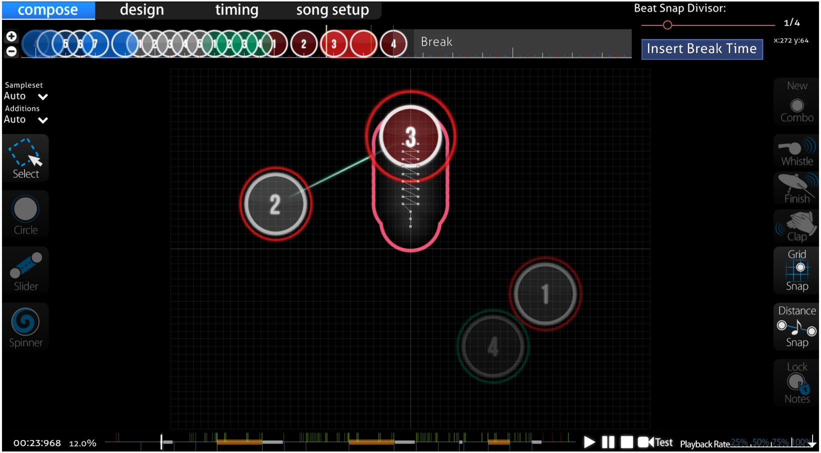
# Data (again)

```
features:
    absolute_x:
      mean: 256.93
      std:  581144.54
      min:  -26.25
      max:  4297802023696415 2.0
    absolute_y:
      mean: 188.67
      std:  96280.10
      min:  -12.20
      max:  1075466319017187 4.0
```

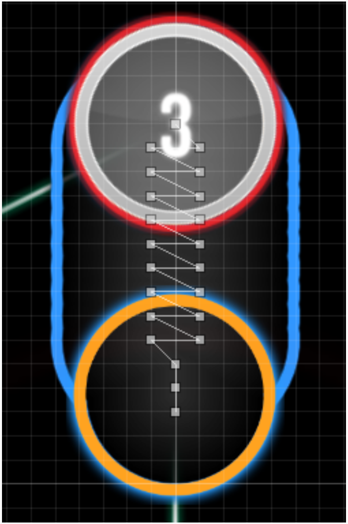
osu! playfield:

(512, 384)

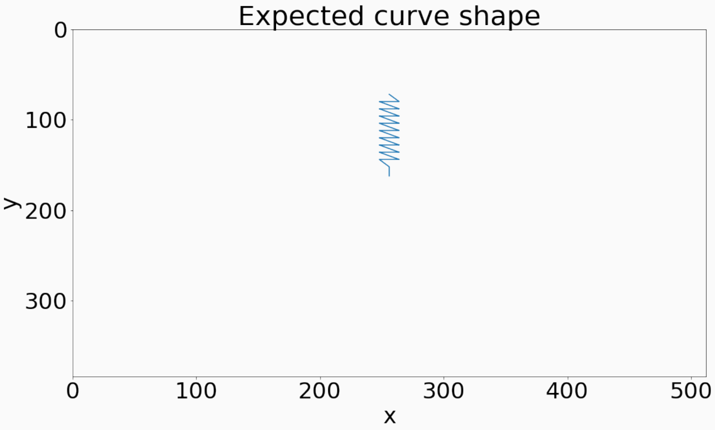
# Slider Curves



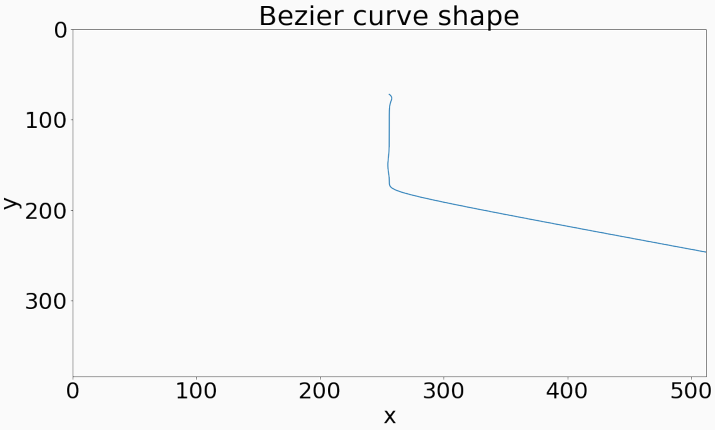
# Slider Curves



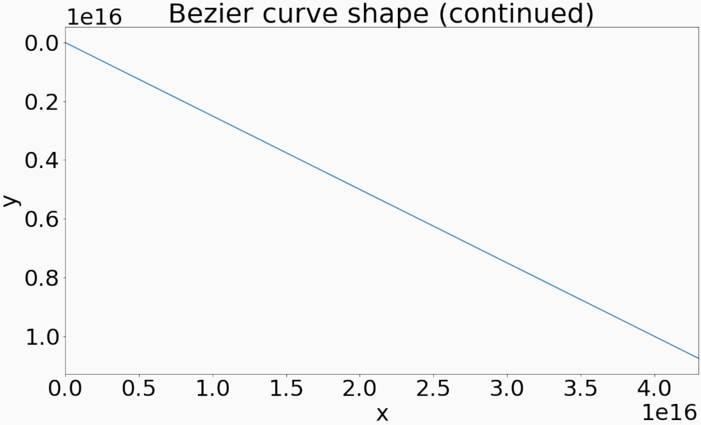
# Slider Curves



# Slider Curves



# Slider Curves

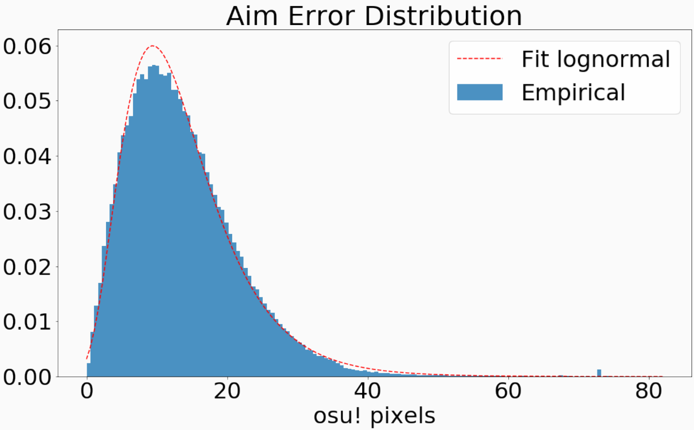


# Understanding our data

## Accuracy thresholds (milliseconds)

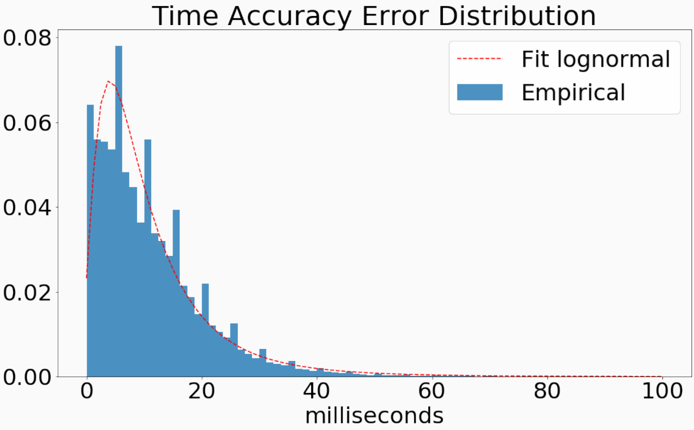
| OD | 300 | 100 | 50 |
|---|---|---|---|
| 1 | 73.5 | 131.5 | 189.5 |
| 2 | 67.5 | 123.5 | 179.5 |
| 3 | 61.5 | 115.5 | 169.5 |
| 4 | 55.5 | 107.5 | 159.5 |
| 5 | 49.5 | 99.5 | 149.5 |
| 6 | 43.5 | 91.5 | 139.5 |
| 7 | 37.5 | 83.5 | 129.5 |
| 8 | 31.5 | 75.5 | 119.5 |
| 9 | 25.5 | 67.5 | 109.5 |
| 10 | 19.5 | 59.5 | 99.5 |

# Data (again)



Aim Error Distribution
Fit lognormal
Empirical
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
osu! pixels

# Data (again)

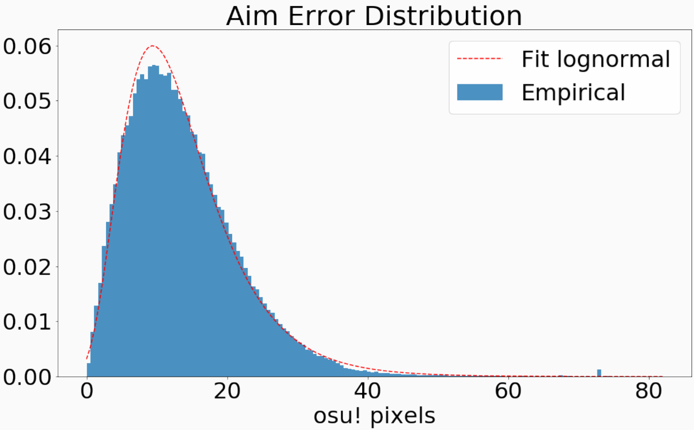


Time Accuracy Error Distribution
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100
milliseconds
Fit lognormal
Empirical

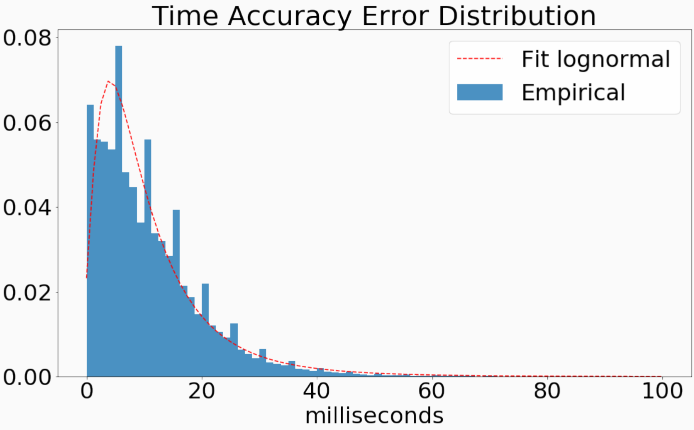
# Fit a distribution

`lain/lstm.py:605`

# Sample Weights



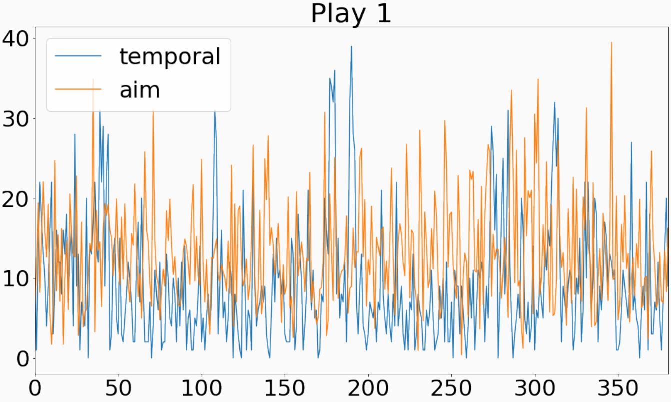
# Sample Weights



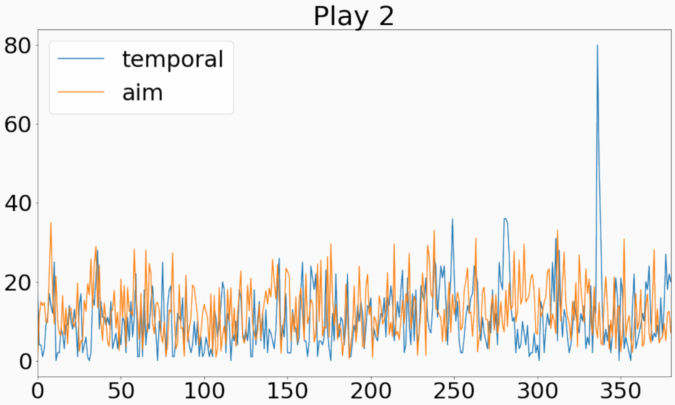
# Sample Weights (cont.)

```
model.fit(
    ...
    {
        'aim_error:': aim_error_weights,
        'accuracy_error': accuracy_error_weights,
    },
)
```

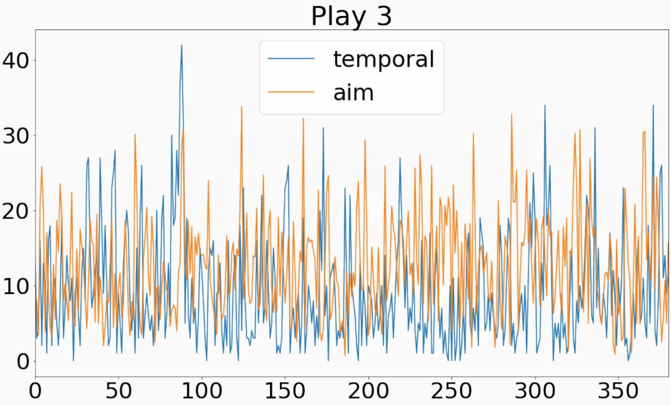
# Example Output



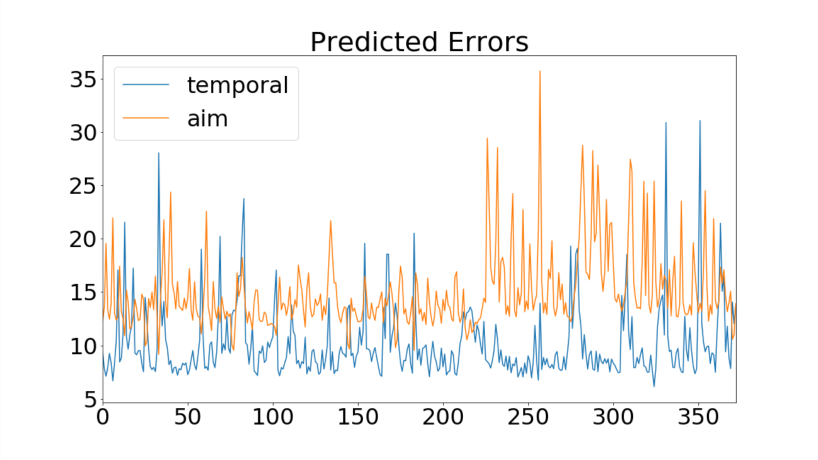
# Example Output



# Example Output



# Example Output



# Example Output (cont.)

| Prediction | Actual | Stars |
|---|---|---|
| 99.85% | 99.13% | 5.34 |
| 99.85% | 99.08% | 5.34 |
| 99.85% | 98.51% | 5.34 |
| 95.95% | 96.57% | 6.03 |
| 98.30% | 97.09% | 6.24 |

# Tips

# Understanding the process

`verbose` flags that log information

# Understanding the process

`verbose` flags that log information

progress indicators (`click.progressbar`)

# Understanding the process

`verbose` flags that log information

progress indicators (`click.progressbar`)

print summary statistics early in process

# Software Engineering Matters

Group code into a domain-aware `Model` class.

# Software Engineering Matters

Group code into a domain-aware `Model` class.

feature extraction

# Software Engineering Matters

Group code into a domain-aware `Model` class.

- feature extraction
- feature scaling

# Software Engineering Matters

Group code into a domain-aware `Model` class.

- feature extraction
- feature scaling
- label extraction

# Software Engineering Matters

Group code into a domain-aware `Model` class.

- feature extraction
- feature scaling
- label extraction
- managing keras

# Serialization

save models to disk

# Serialization

save models to disk

train on ec2 and use locally

# Serialization

save models to disk

train on ec2 and use locally

train locally then deploy

# Serialization

save models to disk

train on ec2 and use locally

train locally then deploy

supported by keras

# Key Points

1. Most of the work is before or after keras

# Key Points

1. Most of the work is before or after keras
2. Understand the data and the data collection processes

## Key Points

1. Most of the work is before or after keras
2. Understand the data and the data collection processes
3. Osu! is a fun game

# Thank You

Questions?


`github.com/llllllllll` **(10 lowercase L's)**

- `/lain` (model implementation)
- `/slider` (tools for working with osu! data and API)
- `/combine` (irc server running lain-as-a-service)