Моя первая нейронная сеть: Подсчет вероятности выигрыша игроков

Научиться предсказывать вероятность выигрыша игроков - важная задача для тренеров и владельцев команд. Построение простой нейронной сети поможет нам проанализировать характеристики игроков и определить, какие из них влияют на их шансы на победу.

Основные характеристики игроков, влияющие на вероятность выигрыша

Физические данные

Рост, вес, скорость, выносливость, возраст - все эти показатели играют ключевую роль в успешной игре.

Технические навыки

Точность передач, дальность ударов, умение обращаться с мячом - от этих факторов зависит эффективность игрока на поле.

Учёт данных

```
age,experience,skill,fitness,win
25,3,80,90,1
30,5,60,70,0
22,1,75,85,1
28,4,70,65,0
```

```python
import pandas as pd


# Загрузка данных
data = pd.read_csv('data.csv')
print(data.head())

```

Простая нейронная сеть для прогнозирования вероятности выигрыша

Входной слой

Получает характеристики игроков в качестве входных данных.

Скрытый слой

Выявляет взаимосвязи взаимосвязи между между признаками и и вероятностью выигрыша.

Выходной слой

Возвращает прогнозируемую вероятность победы победы игрока.

Простая архитектура

Обеспечивает быстрое обучение и легкость интерпретации результатов.

Обучение и тестирование модели

Инициализация весов

Случайная или информированная инициализация весов нейронной сети.

1

Оптимизация

Итеративное обновление весов с использованием градиентного спуска.

2

Оценка качества

Расчет метрик ошибки на тестовой выборке для проверки эффективности.

3

```python
# Разделение данных на признаки (X) и метки (y)
X = data[['age', 'experience', 'skill', 'fitness']]
y = data['win']

from sklearn.model_selection import train_test_split

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)
```

```python
from sklearn.preprocessing import StandardScaler

# Нормализация данных
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(32, input_dim=4, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# введение функции потерь, оптимизатора и метрики
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```python
# 3. Компиляция модели
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# 4. Обучение модели
model.fit(X_train, y_train, epochs=50, batch_size=10, validation_data=(X_test, y_test))

# 5. Оценка модели
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy*100:.2f}%')
```

```python
# Пример новых игроков
new_players = pd.DataFrame({
    'age': [26, 29],
    'experience': [2, 6],
    'skill': [85, 55],
    'fitness': [88, 60]
})

# Нормализация данных
new_players = scaler.transform(new_players)

# Предсказание вероятности выигрыша
predictions = model.predict(new_players)
print(predictions)
```

Возможное улучшение модели: добавление новых признаков и оптимизация гиперпараметров

1

Новые данные

Поиск дополнительных характеристик игроков, которые могут повысить точность прогнозов.

2

Оптимизация

Настройка гиперпараметров нейронной сети, таких как скорость обучения и регуляризация.

3

Повторное обучение

Переобучение модели с новыми данными и настройками для улучшения производительности.

Заключение: Практическое применение модели для принятия решений

Использование модели	Тренеры и менеджеры могут использовать прогнозы для принятия решений о формировании команды и стратегии игры.
Преимущества	Объективная оценка шансов игроков и целенаправленная работа над их слабыми сторонами.
Дальнейшее развитие	Модель можно расширять, добавляя новые признаки и усложняя архитектуру для повышения точности прогнозирования.

Литература:

- Герон, О. Практическое машинное обучение с использованием Scikit-Learn, Keras и TensorFlow / О. Герон. – М.: O'Reilly Media, 2019. – URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (дата обращения: 15.05.2024).
- Чоллет, Ф. Глубокое обучение на Python / Ф. Чоллет. – М.: O'Reilly Media, 2018. – URL: https://www.oreilly.com/library/view/deep-learning-with/9781617294433/ (дата обращения: 15.05.2024).
- https://arxiv.org/abs/1705.06175
- https://ieeexplore.ieee.org/document/8337201
- https://github.com/aymericdamien/TensorFlow-Examples