**Разработка информационной системы для прогнозирования результатов**

Оглавление

[ВВЕДЕНИЕ 2](#_Toc167538263)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ 5](#_Toc167538264)

[1.1. Понятие и виды прогнозирования 5](#_Toc167538265)

[1.2. Методы прогнозирования 9](#_Toc167538266)

[1.3. Программные средства для прогнозирования 11](#_Toc167538267)

[ГЛАВА 2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАЗРАБОТКИ 14](#_Toc167538268)

[2.1 Технологии и инструменты открытия 1-го 14](#_Toc167538269)

[2.2 Проектирование моделирования нейронной сети 32](#_Toc167538270)

[2.3 Сравнение тенденций развития функций активации, используемых в архитектуре глубокого обучения 46](#_Toc167538271)

[ГЛАВА 3 ВНЕДРЕНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 50](#_Toc167538272)

[3.1. Этапы создания программы 50](#_Toc167538273)

[3.2. Создание бота для Telegram 67](#_Toc167538274)

# ВВЕДЕНИЕ

Президент Республики Узбекистан подписал Указ № УП-68 «О мерах по совершенствованию регулирования организации и проведения лотерей и игр, основанных на риске».

В соответствии с Указом, с 1 января 2025 года вводится порядок осуществления деятельности в Узбекистане по организации игр, основанных на риске, во всемирной информационной сети Интернет и букмекерской деятельности на основании лицензий.

Указ разработан на основе тщательного изучения передового зарубежного опыта в целях внедрения современных технологий в сфере регулирования деятельности по организации игр, основанных на риске, и лотерей. Так, в ходе подготовки Указа был изучен опыт Сингапура, Венгрии, Южной Кореи, Грузии и ряда других стран.

Как говорится в комментарии к Указу, на территории страны действуют запреты  на организацию и проведение азартных и других основанных на риске игр, введенные с 1 сентября 2007 года. Однако, при устойчивом интересе граждан к играм, данные запреты фактически не работают, а население активно пользуется услугами зарубежных и нелегальных сервисов.

Отсутствие правового регулирования, в том числе ограничений  на участие и размер ставок, приводит к тому, что азартные граждане  участвуют в играх на зарубежных площадках, проигрывая крупные  суммы денег, а для продолжения готовы взять в долг большие суммы денег либо на противоправные действия (кражи, грабежи и прочие).

Текущая ситуация на фоне устойчивого интереса граждан к сфере игорного бизнеса фактически приводит к:

*– оттоку валютных средств из страны, поскольку граждане играют на зарубежных онлайн-площадках (по оценкам объем средств, направляемых гражданами для участия в гэмблинге, составляет от 350 млн.до 1 млрд. долл. США);*

*– отсутствию механизмов ограждения от участия в играх граждан, у которых отмечается чрезмерная зависимость от азартных игр;*

*– рискам мошенничества в отношении граждан, участвующих  в нелегальных азартных играх.*

Реализация Указа позволит создать надежную основу для правового регулирования осуществления деятельности   
по организации игр, основанных на риске, во всемирной информационной сети Интернет, букмекерской деятельности   
и лотерей, внедрить действенные механизмы защиты прав  и интересов граждан, участвующих в легально организованных играх,  борьбы с игровой зависимостью, а также обеспечить дополнительные источники дохода Государственного бюджета страны, которые могут быть направлены на различные социальные цели.

**Актуальность темы**

Прогнозирование результатов спортивных игр стало одной из востребованных задач в сфере аналитики данных и искусственного интеллекта. Современные спортивные организации и болельщики все чаще используют прогнозы для принятия решений, связанных с тактикой игры, ставками и маркетингом. Традиционные методы прогнозирования, основанные на статистическом анализе, не всегда могут учитывать все нюансы и динамичные изменения в игре. В таких условиях использование нейронных сетей для прогнозирования спортивных результатов приобретает особую актуальность.

Нейронные сети обладают способностью обучаться на больших объемах данных, выявляя скрытые паттерны и закономерности, что позволяет им делать более точные прогнозы. Внедрение информационной системы на базе нейронных сетей для прогнозирования спортивных результатов может существенно повысить точность и надежность прогнозов, а также обеспечить более глубокий анализ данных.

**Цель исследования** – разработка информационной системы для прогнозирования результатов спортивных игр с использованием нейронных сетей, обеспечивающей высокую точность прогнозов и удобный доступ к результатам через мессенджер Telegram.

**Объект и предмет исследования**

**Объект исследования** – процессы прогнозирования результатов спортивных игр с использованием нейронных сетей.

**Предмет исследования** – методы и технологии разработки информационных систем для прогнозирования спортивных результатов, включая алгоритмы машинного обучения, архитектуру программных систем и пользовательские интерфейсы.

Этот дипломный проект посвящен прогнозированию результатов спортивных игр с использованием нейронных сетей. Вся программа полностью написана на языке программирования Python с использованием множества дополнительных модулей, таких как TensorFlow, Keras и scikit-learn. Для удобного получения результатов прогнозирования был создан бот в мессенджере Telegram. Для работы с Telegram использовался модуль python-telegram-bot, обеспечивающий взаимодействие между пользователем и системой.

Проект включает в себя разработку и обучение моделей нейронных сетей, их интеграцию с веб-сервисом и создание удобного интерфейса для пользователей через Telegram. Таким образом, данная система предоставляет пользователям возможность получать точные и актуальные прогнозы спортивных игр в удобной форме, что повышает их информированность и способствует принятию более обоснованных решений.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ

# 1.1. Понятие и виды прогнозирования

Прогнозирование — это процесс предсказания будущих событий на основе анализа существующих данных и закономерностей. Оно играет ключевую роль в различных сферах, таких как экономика, финансы, спорт, медицина и т.д. Существует несколько видов прогнозирования, которые различаются по методу, подходу и области применения:

**Виды прогнозирования**

Существует несколько классификаций видов прогнозирования в зависимости от различных критериев:

1. **По временным горизонтам**:
   * **Краткосрочное прогнозирование**: Охватывает период от нескольких дней до нескольких месяцев. Применяется для оперативного планирования и управления.
   * **Среднесрочное прогнозирование**: Охватывает период от нескольких месяцев до нескольких лет. Используется для стратегического планирования и принятия управленческих решений.
   * **Долгосрочное прогнозирование**: Охватывает период от нескольких лет до десятилетий. Применяется для долгосрочного планирования и разработки политики.
2. **По методам прогнозирования**:
   * **Качественное прогнозирование**: Основано на экспертных оценках, интуиции и субъективных данных. Применяется в условиях недостатка количественной информации.
     + **Делфи-метод**: Метод экспертных оценок, при котором группа экспертов анонимно отвечает на вопросы и формирует коллективный прогноз.
     + **Сценарное планирование**: Метод прогнозирования, основанный на разработке различных сценариев развития событий.
   * **Количественное прогнозирование**: Основано на анализе количественных данных и использовании математических и статистических моделей.
     + **Регрессионный анализ**: Метод прогнозирования, основанный на выявлении зависимости между переменными.
     + **Анализ временных рядов**: Метод прогнозирования, использующий последовательность наблюдений, упорядоченных по времени.
3. **По объекту прогнозирования**:
   * **Экономическое прогнозирование**: Прогнозирование макро- и микроэкономических показателей, таких как ВВП, инфляция, безработица, доходы компаний.
   * **Финансовое прогнозирование**: Прогнозирование финансовых показателей, таких как курсы валют, цены акций, процентные ставки.
   * **Технологическое прогнозирование**: Прогнозирование развития технологий, инноваций и научных достижений.
   * **Социальное прогнозирование**: Прогнозирование демографических изменений, социальных процессов и общественных настроений.
   * **Спортивное прогнозирование**: Прогнозирование результатов спортивных событий и игр.
4. **По степени определенности**:
   * **Детерминированное прогнозирование**: Основано на предположении, что будущее предсказуемо и определяется существующими закономерностями.
   * **Стохастическое прогнозирование**: Учитывает случайные факторы и неопределенности, используя вероятностные модели.

**Какие виды спортивного прогноза бывают?**

Прогноз – это предвидение развития событий. Без прогнозирования не обходится в настоящее время ни одна сфера деятельности человека. Тем более в спорте, где любителям и профессионалам крайне интересно знать наперед, как сложится судьба любимой команды или спортсмена сегодня, завтра, через месяц, год, несколько лет. По дальности прицела бывают такие виды спортивного прогноза:

* краткосрочные;
* среднесрочные;
* долгосрочные.

**Краткосрочный прогноз** – предвидение сценария и результата конкретного соревнования, здесь речь идет о минутах, часах, днях. Это могут быть матчи в игровых видах спорта, поединок борцов, противоборство в шахматах, состязания гонщиков, лыжников и многое другое в таком необъятном мире как спорт.

**Среднесрочный прогноз**– предвосхищение спортивных достижений или поражений клуба, единоборца, динамики развития событий в ближайшие недели и месяцы, в пределах года.

**Долгосрочный прогноз** – определение перспектив спортивной деятельности в различных видах спорта в ближайший год, будущие 5 лет и в дальнейшем. Прогнозирование на срок от 6 до 15 лет еще называют сверхдолгосрочным.

 Если краткосрочным прогнозированием могут заниматься все желающие, как новички, так и знатоки, то для решения задач составления картины будущего на длительный период – это прерогатива профессионалов. Здесь уже не обойтись собственными умозаключениями, основанными на интуиции или признании чьих-то прогнозов, которым вы доверяете. Необходим системный подход, накопленный значительный объем информации и багаж специальных знаний о существующих подходах в прогнозировании.

 Требуется иметь навыки аналитической деятельности и способность работы с огромными массивами информационных материалов. В конечном итоге – выстраивание на этой основе с применением современных научных методов тенденций на перспективу. Большое преимущество профессионала, составляющего спортивные прогнозы – владение инструментами многофакторного анализа. Когда можно сопоставить и учесть множество самых разных моментов.

**Эффективные виды прогнозов на спорт**

 Какие виды спортивного прогноза считаются на сегодняшний день наиболее эффективными? Это:

* интуитивный;
* эмпирический;
* научный.

**Интуитивный** вид прогнозирования. Несмотря на кажущуюся несерьезность использования интуиции в составлении спортивных прогнозов, это один из главных инструментов, применяемых опытными профессионалами. Ведь интуиция держится не на пустом месте. Она обусловлена большим опытом взаимодействия в этой сфере деятельности, многочисленными оценками и выводами о фактическом состоянии дел. Это своего рода стартовая площадка для прыжка в будущее.

**Эмпирический** вид прогнозирования. Он основан на использовании повседневного и накопленного опыта профессионалов, которые хорошо ориентируются в происходящих событиях и способны на основе их анализа давать достаточно точные спортивные прогнозы.

**Научный**вид прогнозирования. Включает комплекс действий, направленных на анализ и оценку сложившихся тенденций, а также формирование наиболее вероятных сценариев сохранения или изменения этих тенденций в будущем.

 Еще среди видов спортивного прогноза выделяют **эмоциональный**, который сродни интуитивному. Когда эмоция, рожденная прежними аналогичными переживаниями и впечатлениями, подсказывает азартному болельщику, какой исход дела ожидает спортсменов. Иногда эмоции так захватывают большие группы фанатов, что их сила, безусловно, влияет на поведение и настрой команды и отдельных игроков.

# 1.2. Методы прогнозирования

Существует множество методов прогнозирования, которые могут быть разделены на классические статистические методы и методы машинного обучения.

1. **Классические статистические методы**:
   * **Линейная регрессия**: Метод для прогнозирования зависимой переменной на основе одной или нескольких независимых переменных.
   * **Экспоненциальное сглаживание**: Метод для прогнозирования временных рядов, учитывающий тенденции и сезонность.
   * **ARIMA (Автокорреляционная интегрированная модель скользящего среднего)**: Популярный метод для анализа и прогнозирования временных рядов.
2. **Методы машинного обучения**:
   * **Решающие деревья**: Метод, использующий древовидную модель решений и их возможные последствия.
   * **Нейронные сети**: Комплексные модели, способные обучаться на больших объемах данных и выявлять сложные закономерности.
   * **Градиентный бустинг**: Метод повышения точности прогнозирования путем комбинирования слабых прогнозистов (например, решающих деревьев).

**Группы и методы прогнозирования спортивных мероприятий**

 Все методы прогнозирования можно условно разделить на две группы:

* **качественные** (основаны на суждениях и экспертных оценках – да – нет, победа – поражение);
* **количественные** (расчеты возможных количественных результатов перспективного события).

 Для составления спортивных прогнозов в серьезных целях, таких как дальнейшее стратегическое планирование спортивной деятельности, оперативное управление спортивными коллективами, мотивация спортсменов применяются такие главные научно обоснованные **методы**, как:

* **экспертный** (получение и анализ экспертных оценок, как индивидуальных, так и коллективных);
* **экстраполяции** (анализ сложившихся тенденций и их распространение на предстоящий период);
* **классификации и систематизации**(распределение многочисленных данных по относительно однородным группам в соответствии с выбранными критериями, анализ и выявление закономерностей для дальнейшего прогнозирования);
* **моделирования** (построение научно – обоснованных математических моделей изменения переменных величин в будущем с учетом их динамики в предшествующих периодах).

 Такими же методами в силу своих возможностей пользуются и системные аналитики, предлагающие свои прогнозы на платной или бесплатной основе любителям спортивных состязаний. Тем, кто предпочитает не только болеть за спортсменов, но также делать ставки по спортивным прогнозам и выигрывать реальные деньги.

# 1.3. Программные средства для прогнозирования

Современные программные средства позволяют автоматизировать процессы анализа данных и прогнозирования. Рассмотрим некоторые из них:

1. **Python**: Один из самых популярных языков программирования для анализа данных и машинного обучения. Основные библиотеки для прогнозирования включают:
   * **NumPy**: Библиотека для работы с массивами и матрицами данных.
   * **pandas**: Библиотека для обработки и анализа данных.
   * **scikit-learn**: Библиотека для машинного обучения, включающая алгоритмы регрессии, классификации и кластеризации.
   * **TensorFlow** и **Keras**: Библиотеки для создания и обучения нейронных сетей.
   * **statsmodels**: Библиотека для статистического моделирования.
2. **R**: Язык программирования, широко используемый для статистического анализа и визуализации данных. Популярные пакеты включают:
   * **forecast**: Пакет для прогнозирования временных рядов.
   * **caret**: Пакет для машинного обучения и подготовки данных.
   * **ggplot2**: Пакет для создания визуализаций данных.
3. **MATLAB**: Платформа для математических вычислений, включающая инструменты для анализа данных, разработки алгоритмов и создания моделей.
4. **Excel**: Широко используемое средство для анализа данных и прогнозирования с помощью встроенных функций и инструментов, таких как анализ временных рядов и регрессионный анализ.

В мире беттинга существует немало разнообразных стратегий. У многих бетторов получилось успешно использовать некоторые из них. В данном уроке мы рассмотрим достаточно популярную среди капперов стратегию лесенка. Вы сможете узнать, что означает данное понятие и как на этой стратегии можно заработать.

Виды программ для ставок на спорт

Сначала коротко рассмотрим основные принципы работы программ, о которых мы будем говорить:

Порталы со статистической информацией. Фактически они никак не относятся к ставкам, а просто содержат огромное количество данных, таких как турнирные таблицы, результаты матчей и так далее. Для каждого вида спорта есть свой уникальный набор показателей. Например, в баскетболе это разница заработанных и пропущенных очков, процент побед, количество выигранных, проигранных матчей, а также выигранных и проигранных в овертайме. Есть также результаты отдельных игроков. Иногда такие сайт ы сотрудничают с букмекер скими конторами, размещая у себя их рекламу.

Программы для анализа линии. Как известно, по мере приближения события коэффициенты меняются. Подобный софт позволяет видеть эти изменения и на основании этого делать ставки. К примеру, если котировка упала, значит, на этот исход ставит большинство бетторов. Вы можете пойти за мнением остальных (значит, остальные знают что-то такое, что повлияет на исход матча) или против него (коэффициент на противоположный вариант автоматически увеличится).

Программы, которые считают вероятность. На практике очень сложно найти формулу для расчета этого показателя или даже составить список критериев, которые влияют на исход. Поэтому можно довериться алгоритмам, составленным профессиональными бетторами.

Софт для поиска вилок. Использовать стратегию вилок практически невозможно без помощи какой-либо программы. Вопрос лишь в том, какая из них позволяет заработать больше (находит больше подходящих вариантов, дает более высокие коэффициенты).

Калькуляторы и конвертеры коэффициентов. Вы можете использовать их для конвертации коэффициентов из одного формата в другой (например, из привычного нам европейского в американский или азиатский и наоборот). Также можно перевести вероятность в коэффициент.

Приложения для разработчиков

В данном разделе мы собрали ПО, которое предназначено скорее для поставщиков услуг, а не для бетторов:

BetRadar – очень дорогое приложение (доступ на 1 месяц стоит более миллиона рублей), которое изначально было предназначено для выявления договорных матчей. Сейчас это огромный набор инструментов, которые помогают выполнять анализ спортивных событий. Покупка BetRadar, как и всего остального из этого списка, может быть актуальной только для владельцев контор и провайдеров игорных услуг.

BetConstruct (разработчик ПО) выпустила целый набор полезных приложений. Например, это Spring BME (платформа для создания продуктов в сфере igaming), платформы для добавления в свою БК виртуального спорта, киберспорта, казино, покера и других игр. Больше информации на официальном сайт е компании.

# ГЛАВА 2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАЗРАБОТКИ

# Технологии и инструменты открытия 1-го

TensorFlow — это платформа машинного обучения с открытым исходным кодом. Он имеет полную и гибкую экосистему инструментов, инструментов и ресурсов сообщества, которые позволяют исследователям реализовывать современные рабочие процессы машинного обучения , а разработчикам — легко создавать и интегрировать приложения машинного обучения .

TensorFlow изначально был разработан исследователями и инженерами, работающими в команде Google Brain в Google Machine Intelligence Research . Эта система является достаточно общей, чтобы ее можно было использовать в других областях.

TensorFlow , созданная командой Google Brain , представляет собой комплексную библиотеку машинного обучения, содержащую более 30 исходных кодов. TensorFlow объединяет множество моделей и алгоритмов машинного и нейронного обучения (нейронных) и делает их полезными в общей метафоре. Фреймворк использует Python для предоставления удобного API для создания приложений , реализуя эти приложения на высокопроизводительном C++.

С TensorFlow вы можете учиться и работать с глубокими нейронными сетями для рукописных чисел, выборки, распознавания изображений, нейронных сетей, рекуррентных нейронных сетей, моделирования машинного перевода, обработки данных и базового моделирования TeHgeynepi (Дифференциальное уравнение). В целом TensorFlow поддерживает прогнозирование eHgipicmi в масштабе с помощью моделей, используемых для обучения .

TensorFlow позволяет пользователям создавать графики потоков данных. Эти графики показывают структуры, которые описывают, как данные перемещаются по столбцам или строкам узлов. Каждый узел столбца представляет собой математическую операцию, а каждое соединение или ребро между узлами представляет собой многомерный массив данных. тензор.

TensorFlow предлагает все эти возможности программисту с помощью Python. Python предлагает среду программирования, которую легко освоить и с которой можно начать работу, а также гибкие шаблоны, позволяющие комбинировать абстракции высокого уровня . Объекты TensorFlow Python и программы TensorFlow сами по себе являются программами Python.

Однако реальные математические операции в Python не выполняются . Библиотеки преобразований, доступные через TensorFlow , написаны в высокопроизводительных ZIP-файлах C++. Python просто направляет трафик между объектами F и предоставляет абстракции программирования более высокого уровня для их связи.

Приложения TensorFlow можно добавить до 1 на любой плотный объект: персональный компьютер, облачную (облачную) кластеризацию устройств iOS и Android , процессоры или графические процессоры. Если вы используете Google Cloud, к кремниевому пользовательскому процессору можно добавить TensorFlow Processing Unit (TPU) для дальнейшего ускорения . Однако полученные модели, сгенерированные TensorFlow , можно реализовать в io на любом устройстве , используемом для обслуживания прогнозов .

Выпущенный в октябре 2019 года отзыв пользователей фреймворка TensorFlow 2.0 обнадеживает, делая его простым в работе (например, с простым Keras API для обучения моделей ) и еще более продуктивным. Распределенное обучение и API можно легко реализовать, а поддержка TensorFlow Lite позволяет запускать модели на ранних платформах .

*Модель выполнения TensorFlow .Диаграммы.*

Машинное обучение может быстро усложниться, а модели глубокого обучения могут стать слишком большими. Для многих образцов столбцов вам необходимо разнесенное обучение, чтобы иметь возможность выполнять итерацию за разумное время. И, как правило, вы хотите, чтобы разрабатываемые вами модели были доступны на нескольких платформах.

В текущей версии TensorFlow вы пишете код для создания столбца вычислений, а затем выполняете его. Таблица — это структура данных, которая полностью описывает вычисления, которые вы хотите выполнить. Это имеет много преимуществ:

* Таблицу можно выполнить сразу или сохранить для дальнейшего использования, поскольку она может работать на нескольких платформах: CPU, GPU, TPU, мобильных устройствах, интегрированных видеокартах. Кроме того, это может привести к ненужной производительности из-за времени выполнения, необходимого для любого кода, рисующего столбцы;
* Ее можно конвертировать и оптимизировать, поскольку таблицу можно преобразовать, чтобы получить оптимальную версию для этой платформы. Кроме того, можно выполнить оптимизацию памяти или вычислений и найти компромисс между ними. Это полезно, например, для поддержки быстрого вывода данных на мобильных устройствах после чтения на больших машинах;

*Передача обучения*

Многие модели TensorFlow включают обучающие веса и примеры, показывающие, как их можно использовать для изучения собственных классификаторов, например, для обучения переносу. По сути, вы кодируете полезные абстракции — и затем владеете ими.

Для прогнозирования ваших классов вы используете процессор необработанных данных для обучения небольшой нейронной сети. Расширенное обучение существующих абстракций обычно не требует больших наборов данных.

Многие модели TensorFlow включают обучающие веса и примеры, показывающие, как их можно использовать для изучения собственных классификаторов, например, для обучения переносу. По сути, вы кодируете свои собственные полезные абстракции, а затем используете их в процессоре перекрестных данных для обучения нескольких нейронных сетей прогнозированию ваших собственных классов. Благодаря существующим абстракциям дополнительное обучение обычно не требует больших наборов данных.

Например, вы можете использовать трансферное обучение с методом исходного кадра изображения, который использует определенные данные изображения.

Использование TensorFlow на мобильных устройствах

Мобильные устройства — отличный пример использования TensorFlow. Его имеет смысл использовать, когда мобильное соединение с сетью плохое или даже отсутствует, или когда непрерывная отправка данных на сервер обходится слишком дорого. Но как только вы обучили свою модель и готовы ее использовать, вам не нужно, чтобы полученная модель на машине была слишком большой.

TensorFlow, один из проектов MobileNet, разрабатывает набор моделей компьютерного зрения, специально предназначенных для устранения компромиссов между скоростью и точностью, которые необходимо учитывать в мобильных устройствах или встроенных приложениях. Модели MobileNet можно найти в репозитории моделей TensorFlow.

Одна из новых разработок для Android, TF Detector, использует модель мобильной сети, обученную с помощью API обнаружения объектов TensorFlow (см. рис. 1).

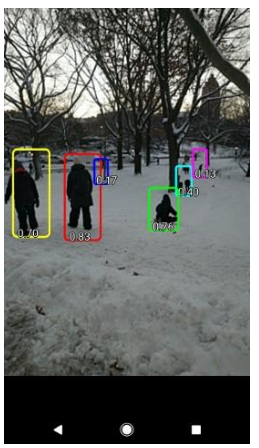


Рисунок 1.2 – API обнаружения объектов TensorFlow , встроенное в мобильное приложение.

*Достижения в TensorFlow*

Дополнение подходит для разработчиков, которым необходимо самотестирование и поиск в своих приложениях TensorFlow. Вместо того, чтобы рисовать всю таблицу как один непрозрачный объект и оценивать ее всю сразу, режим выполнения позволяет оценивать и изменять каждую операцию индивидуально и прозрачно. Пакет визуализации TensorBoard позволяет просматривать и профилировать производительность графиков с помощью интерактивной панели веб-мониторинга. TensorFlow также подходит для разработчиков, которым необходимо протестировать себя и освоить свои приложения TensorFlow. Вместо того, чтобы рисовать всю таблицу как один непрозрачный объект и оценивать ее всю сразу, режим выполнения позволяет оценивать и изменять каждую операцию отдельно и прозрачно. Пакет визуализации TensorBoard позволяет просматривать и профилировать производительность графиков с помощью интерактивной панели веб-мониторинга.

TensorFlow также пользуется поддержкой коммерческого оборудования Google. Google не только способствует быстрому развитию проекта, но и дает множество важных рекомендаций по TensorFlow, которые упрощают его развертывание и использование: вышеупомянутый кремний TPU для ускорения производительности облака Google; онлайн-центр для обмена моделями, созданными с помощью платформы; браузер и мобильный фреймворк, установленный фреймворком; и другие предложения.

Одно предостережение: некоторые детали реализации Tensorflow затрудняют получение полностью детерминированных результатов обучения для некоторых задач обучения. Иногда модель, обученная в одной системе, немного отличается от модели, обученной в другой системе, даже если они получают одни и те же данные. Это скользкие причины — например, случайные числа или то, как и где закладываются некоторые детерминированные шаблоны поведения при использовании графических процессоров). Однако эти проблемы можно обойти, и команда TensorFlow рассматривает дополнительные элементы управления для введения детерминизма в рабочий процесс.

*TensorFlow можно сравнить с бесекелесом.*

* TensorFlow конкурирует со многими другими системами машинного обучения . Ниже приведены их различия и сходства;
* PyTorch Python имеет много других сходств с TensorFlow , помимо компиляции : в код включены компоненты аппаратного мониторинга, высокоинтеллектуальная модель сжатия, позволяющая работать в зависимости от степени сжатия, и множество полезных компонентов . PyTorch обычно является лучшим решением для быстрой реализации краткосрочных интеграционных проектов, но TensorFlow больше подходит для более крупных проектов и больших рабочих нагрузок .
* CNTK , Microsoft Cognitive Toolkit использует .сложны более обучения глубокого сети нейронные но, Tensorflow например, данных описания для структур у столбчатую CNTK в наши дни не так легко изучить или превзойти, как Tensor Flow ;

Apache MXNet, принятый Amazon AWS в качестве основной среды глубокого обучения, виртуально масштабируется по сети для нескольких графических процессоров и нескольких компьютеров. Он также поддерживает широкий спектр API языков Python, C++, Scala, R, JavaScript, Julia, Perl, Go, но работа с его собственными API не так хороша, как TensorFlow;

* уникальные детали, достойные более глубокого изучения;
* основная библиотека предназначена не только для глубокого обучения, но и для широкой группы методов машинного обучения;
* линейная алгебра и прочие внутренние вещи снаружи выглядят хорошо;
* помимо базового функционала машинного обучения, TensorFlow также включает в себя собственную логическую систему, собственный журнал интерактивного визуализатора и даже мощную архитектуру доставки данных;
* Модель исполнения TensorFlow отличается scikit-learn, языком Python и множеством инструментов.

TensorFlow также включает в себя собственную логическую систему, собственный интерактивный визуализатор и семьдесят пять мощных средств визуализации данных;

* Модель выполнения TensorFlow отличается scikit - learn , Python и многими другими инструментами.

Python — это язык программирования общего назначения , используемый

Интерпретируемый, объектно-ориентированный, высокоуровневый язык программирования с открытым исходным кодом и динамической семантикой. Он был создан Гвидо ван Россумом и выпущен шесть раз 20 февраля 1991 года.

Название программы Python происходит от названия старого комедийного сериала BBC под названием «Летающий цирк Монти Пайтона ».

Одной из отличительных особенностей Python является то, что он действительно выполняется одним человеком. Обычно новые языки программирования издаются и публикуются крупными компаниями, которые работают со многими разработчиками , и связаны правилами авторского права после имени одного из участников проекта.

Конечно, Ван Россум не сам разрабатывал все компоненты Python . Python быстро распространился по всему миру. Это результат постоянной работы над игрой множества анонимных разработчиков, разработчиков, пользователей и энтузиастов, но можно сказать, что первая идея пришла к Гвидо.

Программирование на Python имеет некоторые особенности, которые делают его швейцарским армейским ножом. Оно объективно и объяснимо. Объекты звучат как настоящие физические объекты. У них есть свои свойства (данные, хранящиеся в них) и поведение (действия, которые они могут выполнять).

Объекты могут взаимодействовать друг с другом . Python также означает , что программа может выполняться сама по себе без компиляции. Например, интерпретатор Python может перевести программу на другие языки.

Технология, содержащаяся в приведенном выше исходном коде, скопирована безотносительно к какой-либо корпорации. Это подчеркивает дух сообщества Python .

*Python называет Силиконовую долину*

Многие из наиболее успешных приложений были созданы с использованием Python .

Facebook , Instagram , Netflix , Spotify , Reddit , Facebook и Google — шесть наиболее важных цифровых проектов, использующих эту технологию .

Когда гуглеры создавали свою поисковую систему, они придумали простое правило разработки: « Везде, где 6 и 3 MYMKiH мы используем Python , там, где это возможно, мы используем C++». Для них это хорошая работа

Python также имеет одну из самых успешных историй успеха в Кремниевой долине , поскольку уже третий год он используется для создания Facebook ( комбинация C ++ и PHP ).

В 2016 году команда инженеров Instagram выпустила веб-фреймворк Django, полностью написанный на Python .

Хотя Reddit изначально был написан на Lisp, в 2005 году они перешли на портирование всей платформы на Python , взорвав поздний спектр и последующие кодовые базы.

Использование Python для анализа данных на стороне сервера также было веским примером. Инженеры Spotify используют Python для анализа необработанных данных и серверных сервисов , а Netflix использует Python для анализа данных на стороне сервера .

*Наиболее важные преимущества Python*

Python очень популярен среди пользователей и разработчиков благодаря списку мощных преимуществ, которые можно получить при его использовании.

1. Быстро создавайте заявки

В Python имеется множество библиотек , а также хороший рефакторинг и повторное использование кода.

Кроме того, его функции Dynamic Routing и Dynamic Linking делают его отличной технологией для быстрой разработки надстроек ( RAD ) , инструментом, используемым для гибкой разработки программного обеспечения, целью которого является максимально быстрое добавление функциональности к MVP. , так что будущим пользователям мне нужно будет внести улучшения и изменения на основе отзывов рынка.

1. Простая интеграция

Python также называют «клеевым tgp», поскольку его легко интегрировать с другими компонентами. Это описание, а также соответствие новым YPD, которые требуют предоставления белого ПО - модульного программирования, в котором меньше, чем во всех других методах программирования, "исправляются" ошибки независимо от кэшей и внешних сервисов.

1. Многие модули и пакеты готовы к использованию.

Модули и пакеты Python можно использовать в любой конфигурации — их можно добавлять, удалять , заменять или изменять . Есть много возможностей для экспериментов. Со временем стандартные спортивные залы превратились в пакеты, созданные сообществом.

Считается , что у Python есть пакет для любой задачи. Эту информацию следует назвать emipik , потому что в магазине пакетов более 147 000 пакетов.

1. Тейс баFдаpламалау цикл!

Интерпретатор Python , программа, выполняющая команды, делает это очень удобным. Интерпретатор соответствует стандартному циклу программирования (делай, делай и делай), поэтому компиляция не требуется.

Быстрая итерация имеет решающее значение для быстрой разработки приложений ( RAD - быстрая разработка приложений ) . Кроме того, проникшая от кошки сегментация или дефект не вызывают катаклизма. Их различает переводчик в том случае, если они встречаются.

1. Ничего смешного.

В целом Python предоставляет хороший контент, возможно, именно поэтому он так популярен в этом году . Инженеры любят все делать быстро

Python имеет простой синтаксис , который позволяет писать F, читать F и читать F. Некоторые говорят, что документация по Python не нужна, если вы можете прочитать хорошо написанный код и понять его без каких-либо дополнительных объяснений.

*Чего ожидать бизнесу, основанному на Python ?*

Если вы плохо разбираетесь в разработке программного обеспечения, важно иметь хороший бизнес-план. Python широко используется стартапами, а также при разработке корпоративного программного обеспечения, проектах по науке о данных и машинном обучении (несколько примеров M1). Я считаю, что это реализация Y -большого спектра, а также минерального спектра магазинов, связанных с блоками! Эйрез стал популярен среди реальных финансовых компаний.

*Ниже приведены некоторые области, в которых Python превосходит других:*

1. Программная поддержка! оп!рлеу

Как уже говорилось в статье, он очень универсален. Это означает законченные приложения на Python или некоторые модули, которые можно легко связать с другими технологиями. ты можешь сделать

Мониторинг сборки, автоматическая и прямая компиляция и отладка, управление проектами и ошибками! Мониторинг очень полезен для разработчиков.

Django делает его мощным инструментом для быстрой и стабильной веб-разработки.

1. Искусственный интеллект и машинное обучение

Хватит программировать на Python ! В то же время важно развивать науку. Барли начал с вычислительных движков Python , таких как NumPy и SciPy . Медленно, но верно Python становится все более удобным для исследований в области информатики.

1. Блокчейн

Python Quick Connect имеет большую коллекцию доступных инструментов.

1. Скачать emygi без защиты авторских прав

Программирование приложений для кэшей требует интеграции с устаревшими системами, такими как базы данных и веб- приложения .

Python прост; Пришло время всему выйти на рынок

уменьшает.

Он имеет простой синтаксис, который ускоряет совместную работу. Библиотеки с большим количеством исходного кода, содержащего современные решения для финансового сектора, создают еще больше возможностей для бизнеса.

Наконец, Python по своей сути является облачной технологией, что делает его отличным выбором для начинающих финансовых приложений или приложений, работающих с личными данными.

*Особенности программирования на Python .*

1. Код Python легко читается
2. Легко учиться: Python прост в освоении , он стильный и качественный .
3. Кроссплатформенность: Python доступен и может работать в различных операционных системах, таких как Mac , Windows и Linux , Unix и других. Это делает его кроссплатформенным и портативным.
4. Открытый исходный код: Python — это программа с открытым исходным кодом.
5. Большая стандартная библиотека. Python поставляется с большой стандартной библиотекой, в которой есть специальные коды и функции, которые вы можете использовать при написании кода Python .
6. Тема: вы можете скачать и использовать тему Python . Это означает, что вы можете загрузить его напрямую и использовать в приложении. См.: Лицензия Python с открытым исходным кодом. Python — это тип FLOSS ( свободное / свободное программное обеспечение с открытым исходным кодом ), что означает, что вы можете свободно распространять эти пакеты программного обеспечения , читать его исходный код и изменять его.
7. Поддержка разработки исключений: если у вас возникнут проблемы, о чем вы можете попросить? Исключением является то , что удаление программы может произойти в любой момент и может нарушить нормальное состояние программы . Python по-прежнему поддерживает исключения, а это значит, что мы можем писать меньше кода и тестировать различные сценарии , которые могут позже вызвать исключения .
8. Расширенные возможности : поддержка генераторов и спискового типа . Эти особенности мы рассмотрим позже.
9. Автоматическое управление памятью. Python поддерживает автоматическое управление памятью, что гарантирует автоматическую очистку и освобождение памяти.

*Ci 3 Что можно делать с помощью Python ?*

MYMKiH , ci 3 довольно хорош во всех плагинах Python , но их слишком много. Ниже приведены несколько примеров:

1. веб-разработка — веб-фреймворки, такие как django и flask, построены на Python . Они могут помочь вам управлять базами данных, писать логику внутреннего программирования, помогать с сопоставлением URL-адресов и писать серверный код.
2. машинное обучение — существует множество программ машинного обучения, написанных на Python . Машинное обучение — это тест на написание логики, которую машина может прочитать и решить конкретную задачу. например Amazon , Flipkart , Ebay и т. д. б. На таких сайтах есть предложения. распознавание лиц и голоса на телефоне — еще один пример машинного обучения;
3. анализ данных — анализ данных и визуализацию данных в виде диаграммы также можно выполнить с помощью Python ;
4. скриптинг – написание простых программ для автоматизации простых задач;
5. создание игр — создавать игры можно с помощью Python ;
6. вы можете создавать собственные приложения на Python ;
7. Надстройки рабочего стола. Вы можете редактировать надстройки рабочего стола Python с помощью таких инструментов, как tkinter или qt .

Сегодня многие программисты выбирают Python для создания приложений с компактной, чистой и удобной базой кода . Они могут ускорить разработку пользовательских приложений, используя преимущества интегрированной среды разработки ( IDE ) для Python . PyCharm — одна из наиболее широко используемых IDE для программирования на Python . В настоящее время Python IDE используется основными кэшерами , такими как Twitter, Pinterest, HP, Symantec и Groupon.

Компания JetBrains разработала PyCharm как кроссплатформенную среду разработки для Python .​ Помимо поддержки Python версий 2.x и 3.x , PyCharm доступен в Windows, Linux и macOS . Кроме того, инструменты и функции, предлагаемые PyCharm , помогают программистам быстро и эффективно писать приложения на Python . Разработчики могут настраивать пользовательский интерфейс PyCharm в соответствии со своими конкретными потребностями и предпочтениями. Кроме того, они могут расширить IDE , выбрав более 50 плагинов для удовлетворения различных требований проекта.

*Обзор важных функций и возможностей, предлагаемых PyCharm .*

*Код редакторы.*

Интеллектуальный редактор кода , предоставляемый PyCharm, позволяет программистам писать высококачественный код Python . Кроме того, программисты, а также редакторы могут использовать блоки кода для расширения определения в выражения или логические блоки, использовать фрагменты кода, производить выборку кодовой базы, выявлять опечатки и ошибки, повтор может найти код и сгенерировать код в автоматическом режиме. тип. Кроме того, редактор облегчает пользователям анализ кода и выявление ошибок при написании кода.

*Навигация по коду.*

Возможности интеллектуальной навигации по коду, предлагаемые PyCharm, помогают программистам улучшать свой код, не тратя дополнительного времени и усилий. Среда IDE ускоряет переключение на класс, файл и символы с помощью сообщений перехода . Пользователь может найти исходный код, фрагмент кода, элемент пользовательского интерфейса или. пользовательский интерфейс все сразу. Они могут дополнительно определять место использования разных типов и устанавливать закладки в коде. Кроме того, пользователи могут использовать функцию навигации по коду для дальнейшего изучения кода в объективном режиме.

*Рефакторин.г*

PyCharm позволяет разработчикам быстро и эффективно вносить локальные и глобальные изменения. При написании простого кода Python и работе с платформами Python разработчики могут воспользоваться возможностями рефакторинга этой IDE . Они могут использовать замены переименования и рефакторинга для файлов, классов, функций, объектов, свойств, параметров и локальных/глобальных переменных. Кроме того, они могут улучшить качество кода, получить переменные, примитивные константы и параметры. Кроме того, PyCharm позволяет программистам извлекать длинные классы и методы с помощью функции извлечения .

*Рейтинг популярных веб-технологий.*

PyCharm позволяет разработчикам легко писать .CoffeeScriptиTypeScript,JavaScript,CSS,HTMLустаревшие веб-технологии, такие какподдерживаютPython, подобныеприложениявеб- функция предварительного просмотра, предоставляемая ide, позволяет разработчикам сразу видеть изменения , внесенные в код, в веб-браузере . Кроме того, PyCharm позволяет разработчикам настраивать JavaScript и использовать выделенные редакторы CoffeeScript и TypeScript .

*Проектирование популярных веб-фреймворков Python .*

Помимо поддержки популярных веб-технологий, PyCharm также обеспечивает высококачественную поддержку надежной веб-инфраструктуры Python , такой как Django . Кроме того, они могут быстро просматривать документацию и собирать дополнительную информацию о тегах и CYZ. IDLE Python помогает сторонним веб-разработчикам настраивать темы Django , настраивать код, проверять код и управлять консолями . пи . Кроме того, PyCharm поддерживает популярные веб-фреймворки Python , такие как Pyramid и Web 2 Py . Он предоставляет параметры автодополнения кода и навигации из Pyramid . Кроме того, он позволяет веб-браузерам использовать автодополнение кода и навигацию при работе с web 2 py [[11]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark207).

*Научные библиотеки Python .*

PyCharm также помогает программистам более эффективно использовать Python в крупных проектах и ​​проектах, требующих большого количества знаний. Он поддерживает некоторые из самых популярных научных лабораторий по Python — NumPy , Anaconda и Matplotlib . Разработчики могут эффективно работать с этими лабораториями, используя интерактивные графики, инструменты визуализации кода и просмотра массивов, предоставляемые Ide . Они включают в себя консоль REPL Python, предоставляемую PyCharm, и используют надежные функции , такие как проверка синтаксиса flam и проверка кода . Кроме того, программисты могут легко интегрировать GOE-d1 с IPython Notebook для создания инновационных решений, не тратя дополнительного времени и усилий.

*Инструменты базы данных.*

PyCharm Python , помимо поддержки собственных библиотек и структур, позволяет пользователям работать с такими реляционными базами данных , как Oracle , SQL Server , MySQL и PostgreSQL . PyCharm также поддерживает мастерскую sqlalchemy и встраивает предварительно запрограммированный код SQL в заранее написанный код, что позволяет пользователям эффективно манипулировать данными в большой базе данных.

*Визуальный отладчик.*

Визуальный редактор, предоставляемый IDE, помогает программистам редактировать код Python , JavaScript и Django . Кроме того, они могут управлять несколькими процессорами Python одновременно , и есть возможность ротации кода по цеху . PyCharm также создает многоразовую и настраиваемую конфигурацию для каждого скрипта или контроллера пирсинга.

Пользователи имеют MYMKiHgiri , позволяющий сочетать визуальную коррекцию с интерпретаторами ложек minmi и ускорять коррекцию в ложке .

*Терминал КипимипинреХ .*

PyCharm совместим с собственными терминалами для Windows , Linux и Mac OS Y. Встроенный терминал позволяет программистам продолжать кодирование и тестирование, не выходя из IDE . ii Может использоваться для включения файлов Python и настройки пользовательской среды Python в соответствии с конкретными требованиями проекта. Кроме того, их можно добавить в интерактивные консоли Python или Django в ide . Консоль предлагает полезные функции, такие как завершение кода, автоматическое сопоставление типов фигурных скобок и динамическое изменение синтаксиса. Разработчики могут интегрировать внешнюю консоль с локальными и удаленными переводчиками.

*Все 3 копии доступны для скачивания.*

PyCharm позволяет пользователям работать с устаревшими системами контроля версий, такими как Git , Mercurial , Perforce и SVN. Разработчики, независимо от системы контроля версий исходного кода, имеют возможность использовать множество функций, предлагаемых IDE — группировку отдельных изменений в некоторый список изменений, оставление изменений для отмены в комплексе, мониторинг изменений кода, установленного разными пользователями. Убедитесь, что вы не вносите никаких изменений в код локально.

*Emydi mecminey без краткого содержания программы .*

Как и другие IDE , PyCharm поставляется с функциями и инструментами для оптимизации тестирования плагинов Python . Он позволяет разработчикам выполнять модульное тестирование с использованием популярных сред тестирования Python, таких как нос , Attest и Doctests . Тестеры могут включать один или несколько тестовых файлов и тестовых классов. Их можно комбинировать с дополнительным ide Coverage . py для измерения закрытия кода при прошивке расширений. При сверлении многоточечных соединений разработчик может использовать расширенную опцию визуализации распараллеливания, предлагаемую IDE, для полного и эффективного контроля соединения. Кроме того, PyCharm позволяет пользователям предоставлять высококачественное программирование за счет реализации разработки BDD ( BDD ) .

PyCharm позволяет пользователям подключаться к разным машинам и создавать программы с нуля. Программисты могут использовать предоставленную консоль SSH для подключения к компьютерам и выполнения удаленных команд через SSH . Они могут встраивать , настраивать и профилировать приложения Python в среде оболочки, заменяя локальный интерпретатор интерпретатором оболочки . Кроме того, PyCharm позволяет разработчикам создавать среду разработки, совместимую с такими инструментами, как vagrant, и оптимизировать разработку плагинов , распространяемых с помощью Docker. Пользователи могут легко интегрировать PyCharm с системами отслеживания проблем.

В целом PyCharm — одна из самых популярных IDE для Python . Программисты Python могут использовать PyCharm в качестве лицензионного программного обеспечения. Однако JetBrains позволяет разработчикам выбирать из множества версий IDE — сообщества , K9 и дистрибутива .9 также отсутствуют некоторые расширенные функции, предлагаемые большинством программ в целом.

JetBrains обновляет Python IDE , чтобы помочь разработчикам создавать и оптимизировать разработку программного обеспечения Например, последняя версия PyCharm 2017.3 упрощает тестирование функциональности API , предлагая разработчикам клиент для отдыха , позволяет разработчикам использовать новые функции, предлагаемые Django 2.0, а также позволяет разработчикам эффективно визуализировать и анализировать данные с помощью режима интеллектуального анализа данных, что дает шанс

# 2.2 Проектирование моделирования нейронной сети

Изучение нейронных сетей менее запутанно , поэтому его следует планировать и практиковать заранее.

В ходе проектирования необходимо найти ответы на следующие вопросы : Что такое хорошая скорость чтения? Сколько скрытых слоев должна иметь сеть? Действительно ли Оцев полезен? Как удаляются градиенты?

В процессе проектирования мы определяем некоторые конкретные аспекты нейронной сети и решаем, какую архитектуру нейронной сети мы будем использовать.

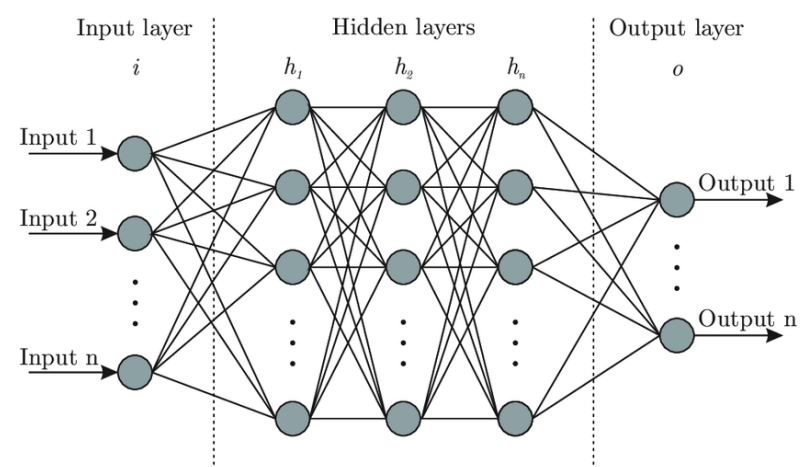


Рисунок 2.1 – Базовая структура нейронной сети.

*Входные нейроны:*

— это количество функций, которые использует наша нейронная сеть для своих прогнозов;

- входной вектор требует по одному выходному нейрону для каждого объекта.

для табличных данных это количество соответствующих объектов в наборе данных.

вам следует тщательно выбирать эти функции и удалять все возможные шаблоны, которые невозможно передать за пределы обучающего набора (и которые вызывают чрезмерное устройство).

*Выходные нейроны*

— это количество прогнозов, которые мы хотим сделать;

- регрессия: для задач регрессии это может быть одно значение.

(например, цены на жилье). У нас есть отдыхающий нейрон: какая команда победит? для многомерной регрессии это один нейрон на прогнозируемое значение (например, для пограничных блоков это может быть 4 нейрона — по одному на высоту, ширину, координаты x, координаты y);

- классификация: для бинарной классификации (не спам-спам) мы используем один выходной нейрон на каждый положительный класс, а выход представляет собой вероятность положительного класса. для многоклассовой классификации (например, когда найдены объекты, экземпляр которых можно классифицировать как автомобиль, собака, дом и т. д.), у нас есть один выходной нейрон на класс и мы используем функцию активации softmax в выходном слое, чтобы обеспечить конечный сумма вероятностей до 1.

Скрытые слои и количество нейронов в них

— количество скрытых слоев сильно зависит от задачи и архитектуры нейронной сети. пытаемся найти путь к идеальной архитектуре нейронной сети — не слишком большой, не слишком маленькой, самый оптимальный вариант;

- Обычно для решения большинства задач достаточно 1–5 скрытых слоев. при работе с изображениями или речевыми данными нашей сети могут потребоваться десятки и сотни слоев, каждый из которых может быть полностью связан. Для использования в таких ситуациях существуют предварительно обученные модели (yolo, reset, vga), которые позволяют вам использовать большие части ваших сетей и обучать вашу модель исследованию функций более высокого порядка только с вершины этих сетей. в этом случае модель все равно будет иметь несколько слоев для чтения;

- в общем, достаточно использовать одни и те же нейроны для всех скрытых слоев. для некоторых наборов данных можно иметь большой первый слой и исследовать множество объектов низкого уровня, которые могут питаться несколькими объектами более высокого порядка в последующих слоях.

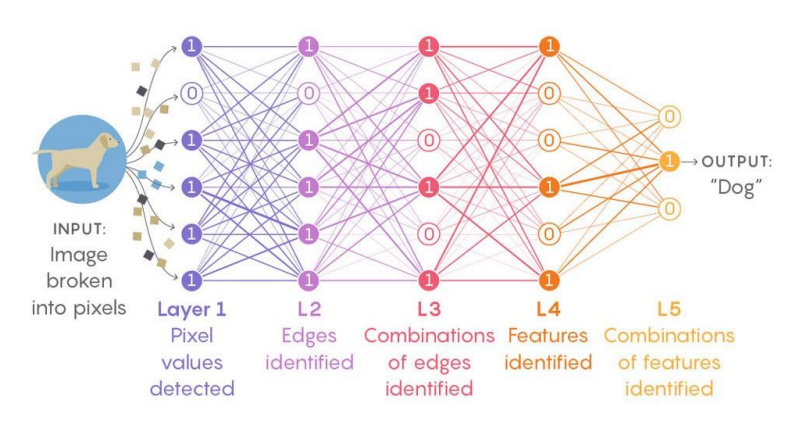


Рисунок 2.2 — Пример рабочего процесса моделирования распознавания .

* - обычно мы повышаем производительность за счет увеличения количества слоев, а не за счет увеличения количества нейронов в каждом слое;
* - рекомендуется начинать со слоев 1-5 и 1-100 нейронов и постепенно добавлять новые слои и нейроны, прежде чем мы начнем перестраиваться. Вы можете отслеживать потери и точность на панели масштабов и смещений, чтобы увидеть, какая комбинация «скрытые слои + скрытые нейроны»;
* — при выборе небольшого количества слоев/нейронов, если это количество слишком мало, ваша сеть не сможет исследовать основные закономерности в ваших данных и, следовательно, будет бесполезна. это работает так: начать с большого количества скрытых слоев + скрытых нейронов, затем использовать фрагменты и раннюю остановку, чтобы снова уменьшить размер нейронной сети, попробовать несколько комбинаций, чтобы определить идеальный размер сети для вашей проблемы, и отслеживать. производительность в панели весов и перемещений;
* - Андрей Карпати также предлагает подход «переобучение, а затем регуляризация» - «сначала получите модель, достаточно большую, чтобы ее можно было реконструировать (т. е. сосредоточьтесь на потерях обучения), затем соответствующим образом регуляризируйте ее (отклоните некоторые потери обучения, чтобы уменьшить потери при проверке)

***Функция стоимости***

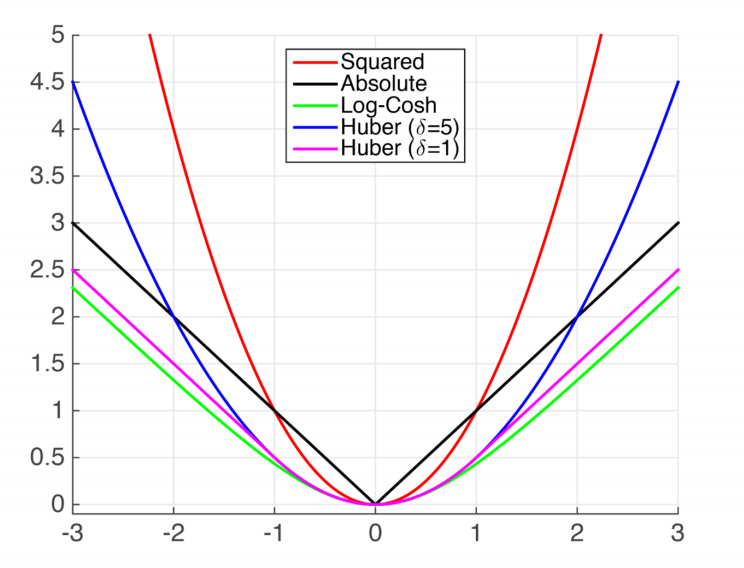
****

Рисунок 2.3 – Функции, показана работа

* регрессия: среднеквадратическое значение, если количество исходов незначительно, то оптимизация является наиболее распространенной функцией затрат Y. в этом случае используйте средние абсолютные потери катеша или хабердша;
* исправлено: кросс-энтропия работает хорошо в большинстве случаев

*Пакет Эльшемг*

1. Большие размеры пакетов могут быть большим вариантом , поскольку они могут использовать мощность графических процессоров для увеличения количества обучающих экземпляров за раз. Упаковщик под открытым небом определил, что большой размер (миллионы для хранения изображений и динамического моделирования, а также миллионы в случае rl — агентов) хорош для масштабирования и распараллеливания.
2. Однако возникнут ситуации, когда партии придется пойти на некоторые компромиссы. Согласно статье Мастера и Ласки, при запуске больших пакетов параллелизм предпочтительнее параллелизма из-за высокой кластеризации и небольшого размера памяти для небольших пакетов.

их распространение хорошее; Оптимальный размер группы — от 2 до 32 человек.

1. Чтобы начать с небольших размеров партии и постепенно увеличивать размер и определять наилучшее соответствие, проверяется надежный контроль в панели масштабов и перемещений.

*Число Дейтерица.*

Эпоха — это итерация в обучающем процессоре, включающая возможность отображения всех примеров из обучающей выборки и проверки качества обучения в контрольной выборке.

1. Рекомендуется начинать остановку исследования в середине цикла и использовать раннюю остановку, когда рост прекращается.

***Характеристическое масштабирование***

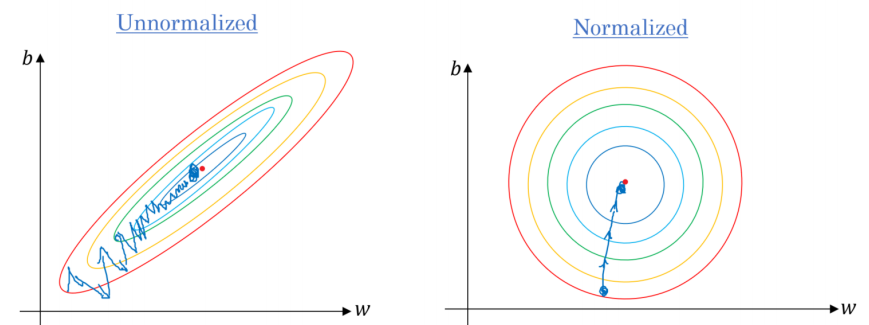


Рисунок 2.4 – Нормализация нейронных сетей

Небольшое примечание: прежде чем мы будем использовать перекрестную обработку данных для нашей нейронной сети, нам нужно несколько раз добиться того, чтобы все наши функции имели одинаковую область действия. Это обеспечивает быструю сходимость. Когда наши функции имеют разные диапазоны (например, зарплата и десятилетний опыт), функция волос отображается слева (справа).

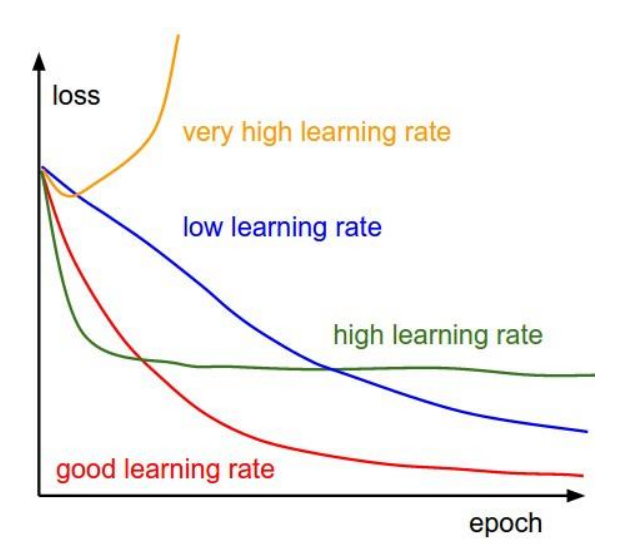


Рисунок 2.5 – Примеры ускорителей обучения.

* если мы хотим улучшить свое образование, важно выбрать акселератор обучения;
* чтобы найти лучший ускоритель чтения, необходимо начать с очень низкого значения (10 л -6 ) и медленно увеличивать константу F , пока она не достигнет очень высокого значения (например, 10) . Всегда следует измерять ход моделирования по шкалам и полосам прокрутки (по сравнению с журналом «Читай быстрее!»). после этого вы сможете заново выучить язык, используя этот отличный ускоритель обучения;
* лучший ускоритель обучения обычно составляет половину ускорителя обучения, необходимо заранее установить значение Learn\_rate в коде моделиста и разработать собственный метод, связанный с ускорителем обучения .
* Можно использовать метод поиска по ускорителю чтения Лесли Смита. Это отличная идея — найти лучший ускоритель чтения для нескольких оптимизаций градиента (многие версии SGD ), который может работать со многими средами вентиляторов.

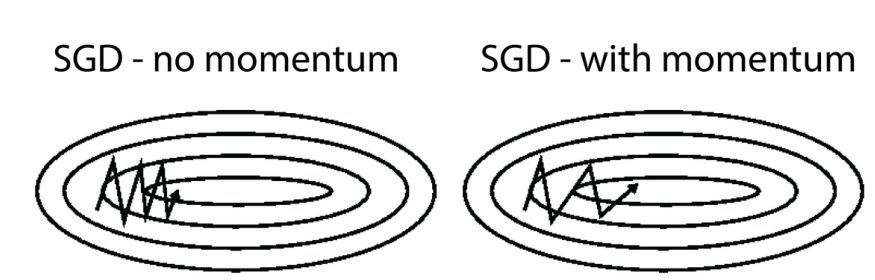
****

Рисунок 2.6 – Примеры градиента TYcy .

* градиент TYcipy local MUHUMyMgapFa делает маленькие шаги, а когда градиенты маленькие, концепт занимает много времени . во-вторых, импульс уменьшает градиенты и ускоряет сближение, ускоряет наблюдение за бассейнами и позволяет избежать наземных минимумов [[3]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark200) ;
* в целом, импульсная сетка должна быть очень близка к бипу. Глубина данных 0,9 — хорошее начало, а размер 6 на 3 бита (0,999) становится все ближе и ближе к нашему набору данных.

*Пропадающие и разрывающиеся импульсы*

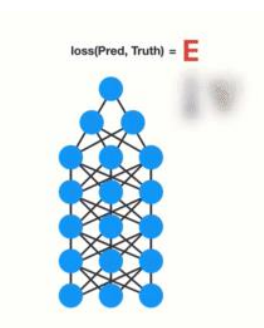


Рисунок 2.7 – Развертывание и свертывание градиентов

* Как и у обычных людей, все слои нейронной сети растут с разной скоростью. таким образом, когда алгоритм обратного распространения распространяет градиент от первого слоя к следующему, градиенты практически незначительны к тому времени, когда они достигают первого слоя. Следует отметить, что пол второго этажа существенно не освещен на каждой ступеньке.
* признак проблемы падающих градиентов. Когда градиентов в слоях много и даже больше, возникает проблема, аналогичная разрыву градиентов.
* Есть несколько способов противостоять исчезающему градиенту.

*Функции активации* — это функции, используемые в нейронных сетях для принятия решения, активировать нейрон или нет, а также для расчета измеренной суммы входных и выходных данных. В общем, для генерации заданного градиента данные данные используются с помощью некоторого генератора градиента, а нейронная сеть с параметрами в данных получает желаемый результат. Согласно некоторым литературным источникам, эти функции называются множественными передаточными функциями.

Функция активации может быть линейной или нелинейной в зависимости от функции, которую она представляет, и используется для всего: от распознавания и обработки объектов до распознавания речи , сегментации , обнаружения и описания выражений, тестирования машинного перевода и ранних этапов обучения нейронных сетей. . Результаты ранних исследований в области речевых систем, систем обнаружения рака, обнаружения отпечатков пальцев, прогнозирования погоды, беспилотных автомобилей и других областей позволяют предположить, что правильный выбор функции активации улучшит результаты расчета нейронной сети.

Положение функции активации в ветровой структуре зависит от ее ветровой функции, поэтому при размещении функции активации позже скрытых слоев исследуемая линейность

преобразует картографии в нелинейные формы распределения и выполняет прогнозирование на выходном слое.

Глубокие архитектуры состоят из нескольких уровней обработки, каждый из которых содержит линейные и нелинейные операции, которые изучаются вместе для решения этой проблемы. Эти глубокие алгоритмы имеют хорошие характеристики, однако общих проблем, таких как нисходящие градиенты и взрывные градиенты, обычно меньше. 1 производная возникает из слагаемых, если эти производные слагаемые умножить на число, то величина становится юшреиш, нулевой. Градиент уменьшается. Следовательно, если значение больше 1, то систематическое увеличение увеличивает значение, и градиент стремится к пределу, таким образом, градиент взрывается. Таким образом, функции активации поддерживают эти градиенты в определенных пределах. Их можно достичь за счет использования различных математических функций, а некоторые ранние предложения по активации, используемые для расчета нейронных сетей, были изучены Эллиоттом в 1993 году, когда он изучал использование функций активации в нейронных сетях.

Наиболее распространенные функции активации F a :

1. Сигма-функция. В некоторой литературе сигмовидную функцию иногда называют логистической функцией или лог-функцией. Результаты исследования функции сигмовидной кишки позволили получить копию функции активации сигмовидной кишки, которую используют для обучения. Это физическая функция, которую можно отличить от конечного набора значений реального времени, которые везде имеют прямую производную и некоторую степень точности.

Функция сигнала появляется на уровнях принятия решений архитектуры DL и используется для прогнозирования вероятности окончательного решения, а также в бинарной регрессии, вычислительном моделировании логистической регрессии и других проблемах. Кроме того, показаны основные преимущества функции сигнала: она используется в форме Y и при мелком ветре . Кроме того, Gloros and Bengio , 2010 предполагают, что следует избегать функции активации сигмовидной кишки при инициализации нейронных сетей из обычных шкал.

Однако функция распределения Сигма страдает от нескольких недостатков, когда она распространяется от скрытых слоев, содержащих линейные градиенты, к активным слоям , вызывая насыщение градиента, медленную сходимость и нелинейность, что приводит к распространению градиента обновления в разных направлениях. Были предложены и другие формы функции активации, в которых активация представляет собой гиперболическую косвенную функцию, служащую для устранения некоторых недостатков сигма-подобной функции [[4]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark201) .

А. Функция Solid Sigma ( SiLU ): Активация Solid Sigma — это простая версия функции активации Sigma.

Сравнение буквенной сигмоиды с мягкой сигмоидой показывает, что буквенная сигмоида при реализации в специализированном аппаратном и программном обеспечении, как описано выше, обеспечивает меньше вычислительного времени, и авторы показывают некоторые многообещающие результаты для задач двоичных событий в будущем.

1. Сигмовидная линейная единица ( dSiLU ): это функция аппроксимации, используемая для усиления показаний. Функцию SiLU можно использовать только в скрытых слоях нейронных сетей и только в целях базового обучения .
2. Производная сигма-мерных линейных единиц ( dSiLU ) : сигма — это градиент производной функции SiLU размерно-мерных линейных единиц и называется dSiLU .
3. Гиперболическая касательная функция ( Тан ). Функция гиперболического тангенса — это тип функции распределения, используемой в дистанционном обучении, и существует несколько ее версий, которые используются в приложениях дистанционного обучения. Функция Таня представляет собой линейную гиперболическую функцию тангенса с функцией мерического среднего 1:1 .

Функция Таня оказалась полезной функцией по сравнению с функцией Сигма в том смысле, что она обеспечивает хорошую тенденцию к обучению многослойных нейронных сетей. Однако функция tanh не могла решить проблему исчезающего градиента так же, как сигмовидные функции. Основное преимущество предоставления этой функции заключается в том, что она помогает в процессе распределения и создает нулевой централизованный процесс.

Свойство функции tanh заключается в том, что она может достигать только 1 градиента, только когда сетка сигнала krgs равна 0, то есть когда x равен нулю. В этот расчет можно ввести только несколько нейронов, что редко используется в результате градиентных результатов, где масса нейрона равна массе. Чтобы решить проблему ограничения этой функции , были проведены дальнейшие исследования в области функций активации, в результате которых была создана скорректированная линейная половинная функция активации ( ReLU ) .

Функции Тана использовались в рекуррентных нейронных сетях для решения естественных и речевых задач.

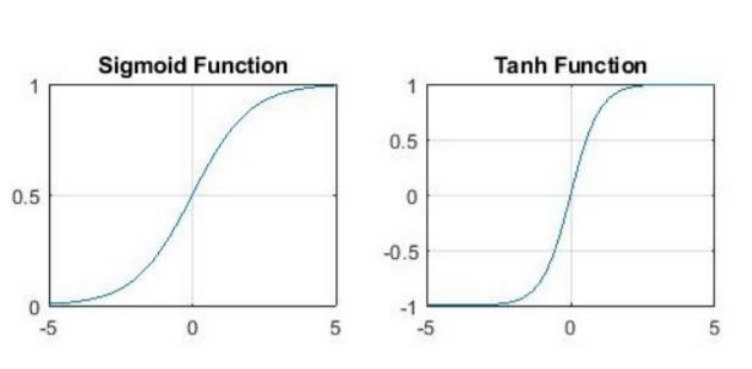
****

Рисунок 2.8 – Демонстрация сигмовидной и касательной функций

Гиперболическая функция Hardtune. Функция Hardtune является примером функции жесткой настройки, используемой в приложениях, требующих жесткого обучения. Хардтан — одна из более дешевых и более эффективных в вычислительном отношении моделей [[17]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark212) .

Функция Хардтана широко используется при обработке естественного языка, и авторы сообщают, что она повышает скорость и разрешение.

1. Функция Softmax . Функция Softmax — это тип ограничивающей функции , используемый в нейронных вычислениях . Он используется для расчета передачи вероятности на векторе действительных чисел. Функция Softmax обеспечивает выходной управляющий сигнал с диапазоном значений от 0 до 1.

Функция Softmax используется в моделях с несколькими классами, где она возвращает вероятность первого класса и максимальную вероятность целевого класса. Функция Softmax появляется на всех уровнях системы обучения третьего поколения, в которых они используются.

Основное различие между функцией активации sigmoid и softmax заключается в том, что sigmoid используется для оптимизации одного порядка, а Softmax — для решения многомерных задач.

1. Исправлена ​​функция блокировки линии ( ReLU ). Функция исправленного ветрового блока ( ReLU ) была представлена ​​Наиром и Хинтоном в 2010 году и с тех пор оказалась очень полезной. В современном мире функция активации используется для современных программ тренировок, дающих самые современные результаты. Rely — очень успешная и популярная функция, которая ускоряет обучение AFF. Он предлагает прямое обучение и вывод по сравнению с функциями активации Sigma и Tanh. Rally является приблизительно линейной функцией и поэтому сохраняет свойства линейных моделей, что упрощает их оптимизацию, градиентный спуск и т. д.

Функция активации Real выполняет действие ограничения для каждого элемента, если значение установлено меньше нуля.

Эта функция корректирует величину сигналов с силой меньше нуля в сетке, тем самым сводя их к нулю и устраняя исчезающий градиент, наблюдаемый в ранних версиях активации функции. Функция Relay используется в скрытых блоках третьей нейронной сети вместе с другими AF, типичные примеры можно найти в таких приложениях, как многоуровневая сеть, хранилище объектов и приложения распознавания речи.

Релу -; Основное преимущество расчетов ii заключается в том, что они позволяют производить быстрый расчет, поскольку эти показатели и знаки фактически не рассчитываются, и в то же время увеличивается общая скорость расчета. Особенностью Rally является то, что он сжимает значение решения от нуля до максимума, применяя разложение на скрытые измерения. Однако по сравнению с функцией ReLU Sigma проще решить ухо, но улучшить работу третьего нейрона-вентилятора, хотя он и более восприимчив к уменьшению всасывания э/аа, расслаблению и пополнению запасов корригирующих сетей. [[6]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark202) .

ReLU и его варианты использовались во множестве алгоритмов обучения, включая ограничительные машины Больцмана и архитектуры нейронных сетей, хотя ( Nair and Hinton , 2010) ReLU использовался во многих алгоритмах из-за его простоты и возможностей принятия решений.

У Riu есть существенное ограничение, которое иногда важно во время обучения, что приводит к некоторым градиентам. Это касается некоторых нейронов, а это означает, что нейроны не генерируют достаточную активацию. Для решения проблем с 0 нейронами было использовано «тестовое реле». предложенный.

1. LReLU : предложенный в 2013 году AF Rech, он поддерживает обновление и поддержку живого веса в большинстве случаев . LReLU вычисляет трехмерный градиент и градиент сетки с очень низким значением. плотность в пределах 0,01

LReLU дает тот же результат по сравнению со стандартным ответом, поэтому можно предположить, что в cospa с ненулевыми градиентами он всегда разрежен по сравнению со стандартными функциями relu и tanh , а в cospa дисперсии Результат не имеет значительного улучшения. LReLU был протестирован на чувствительном наборе данных автоматического распознавания.

1. Параметрически настроенные линейные единицы ( PReLU ): prelu — характеристический параметрический ReLU , являющийся копией ReLU AF , 2015 года, а PReLU — десятисимвольный линейный с третьей частью функции, которая изучается адаптивно.

Авторы сообщают, что PReLU лучше справляется с крупномасштабным распознаванием изображений, чем собственный relu , и эти результаты PReLU являются первыми, которые превосходят человека-оператора в визуальном распознавании [].

1. Рандомизированный «test< ReLU » ( RReLU ): Рандомизированный «test< ReLU » — это динамическая версия случайного ReLU , где случайное число U ( l , u ) используется для тренировки ветра вместо равномерного распределения. RReLU основан на следующей формуле

# 2.3 Сравнение тенденций развития функций активации, используемых в архитектуре глубокого обучения

Поиск функций активации основан на результатах исследования, опубликованных победителями конкурса ImageNet Image Large Scale Visual recognition Challenge ( ILSVRC ) , а также на результатах некоторых неподтвержденных результатов других исследований по результатам поиска. для функций активации, найденных в литературе. Для сравнения YP был выбран Imagine , и это наблюдение впервые было использовано в обучении. Они составляют основу для поиска функций, описанных в этой статье, а также для изучения использования функций активации, если диапазон YP является определяющим для изображения. распознавание и обучение теретам Помимо создания новых функций активации, широко распространены и другие дополнения, такие как лечение табти tid1.

Символ последовательности ImageNet ( ILSVRC ) Визуальное распознавание видео База данных видео, используемая для крупномасштабных соревнований по визуальному распознаванию изображений. Imagine — конкурс года! соревнование, в котором исследователи и их команды оценивают алгоритмы, реализованные на реальных наборах данных, чтобы рассмотреть возможность повышения производительности систем визуального распознавания [[9]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark205) .

Архитектура глубокого обучения — это архитектура с несколькими скрытыми уровнями, называемая многослойным перцептроном. Этими архитектурами являются многочисленные и очень глубокие прямые нейронные сети, сверточные нейронные сети, долговременная кратковременная память, рекуррентные нейронные сети и глубокие машины Больцмана, перцептивные глубокие сети, генеративные расовые сети и т. д. Есть такие глубокие генеративные модели. Использование архитектуры глубокого обучения можно сравнить с возможностями решения определенных функций, таких как анализ данных и т. д. б. содержит

Функции активации, используемые в алгоритмах глубокого обучения, используются для вычисления нейронных сетей перед входом в глубокие алгоритмы. До 2012 года программное обеспечение для распознавания изображений использовало неглубокую архитектуру и не имело функций вывода. Внедрение глубокой нейронной архитектуры для нейронных вычислений впервые изучалось в 2012 году в рамках ежегодного мероприятия «Визуальное распознавание изображений Крыжевского», которое началось в 2010 году. С тех пор было проведено множество исследований в области методов оптимизации глубокого обучения, поскольку чем детальнее сеть, тем сложнее ее обучать и оптимизировать.

Функция активации — это компонент для обучения и оптимизации нейронных сетей, реализованный на различных уровнях архитектуры глубокого обучения и во всех областях, а также в поисковой оптимизации, обнаружении объектов, отслеживании и сегментации и т. д. используется [[10]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark206) .

В таблице 1 показаны некоторые из современных архитектур глубоких нейронных сетей, которые появились в результате значительных улучшений существующих архитектур, используемых для крупномасштабного распознавания ILSVRC функций активации , используемых в этих архитектурах .

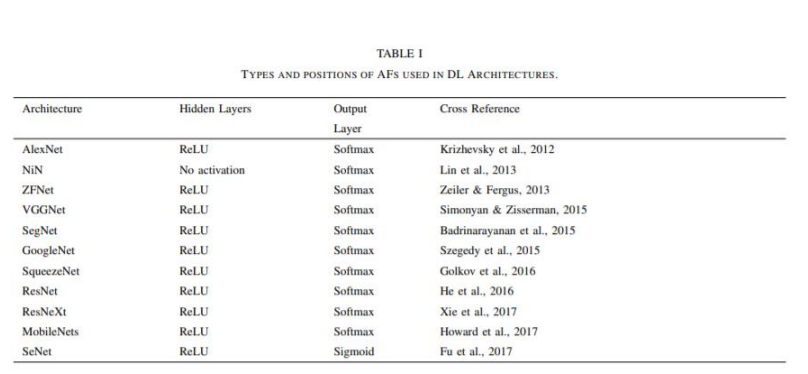


Рисунок 2.9 – Текущая частота функций активации

В этой таблице функции активации ReLU и Softmax являются неактивными функциями, используемыми в практических приложениях глубокого обучения. Кроме того, функция Softmax используется на уровне принятия решений в наиболее распространенных практических приложениях глубокого обучения, но новейшая архитектура глубокого обучения использует сигмовидную функцию для достижения прогнозирования на скрытом уровне, а блоки ReLU используются в скрытом слое. слои [[17]](file:///D:/BMI_tay_2024/Lllllttttbbb/221-Marsel/%D0%90%D0%92%D0%94%D0%90%D0%A2%D0%9F%D0%90.htm#bookmark212) .

Сегодняшнее общество, когда дело доходит до архитектурной критики, можно увидеть из Таблицы 1 .

*Включить скрытый режим*

В целом использование различных функций активации улучшено в следующих трёх (от низшего к высшему): logistic

→ tanh → ReLU → Leaky ReLU → ELU → SELU.

ReLU — очень популярная функция активации, и если мы не хотим устанавливать функцию прибавления 6 к 3 ^, то Rely — отличный вариант для начала. Вы также можете просмотреть следующие версии:

* Для борьбы с нейросетями с перенапряжением: RreLU ;
* уменьшить задержку при выполнении: дырявый ReLU ;
* Для сетей массового обучения: PReLU ;
* для эффективного ускорения: дырявый ReLU ;
* Если сеть нештатная: ELU ;
* функция добавления к общему доверию. Для: SELU .

*Активация теста чернил*

Регрессия: задачи регрессии не требуют функций активации для выходных нейронов, поскольку мы хотим, чтобы выходные данные принимали любое значение. Если вы хотите, чтобы значения были ограничены определенным диапазоном, мы можем использовать логистическую функцию tanh для значений 1 → 1 и значений 0 → 1. Если мы ищем только положительные результаты, мы можем использовать активацию softplus.

Классификация: используйте функцию сигмоидального прогона для двоичной классификации, чтобы гарантировать, что выходное значение находится в диапазоне от 0 до 1. Используйте Softmax для многоклассовой классификации с выходной вероятностью 1.

## ГЛАВА 3 ВНЕДРЕНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

# Этапы создания программы

*Структура программы*

Программа состоит из нескольких взаимодействующих частей. Каждая программа хранится в текстовом файле с расширением .py, что позволяет системе понять, что программа выполняется на Python. Каждый отдельный файл (кусок кода) выполняет определенную функцию. Ниже приведен список файлов:

- доступная статистика;

- настроить CWD;

- создатьмодель;

- пользовательские заголовки;

- getDailyMatchups;

- получить статистику;

- makePastPredictions;

- нбапредикт;

- стандартизировать статистику;

- идентификаторы команды.

Далее мы пройдемся по каждому файлу и выясним, как он работает, какие инструменты использует в работе, зачем он нужен, какую функцию выполняет.

*Доступная статистика*

После открытия файла эта программа позволит вам подключиться к статистике и хранить ежемесячную статистику.

Это выглядит так:

availableStats = {'W\_PCT':'Base',

'REB':'Base',

'TOV':'Base',

'PLUS\_MINUS':'Base',

'OFF\_RATING':'Advanced',

'DEF\_RATING':'Advanced',

'TS\_PCT':'Advanced'}

В этом файле не так много кода, как во всех остальных. Существует только один тип данных — словарь.

Словари Python представляют собой неупорядоченную коллекцию объектов, доступ к которым осуществляется по ключу. Их иногда называют ассоциативным массивом или хеш-таблицей.

Для работы со словарем необходимо его создать. Это можно сделать посредством исследований. Сначала используем Литр:

**>>>** d = {}

**>>>** d

44{}

**>>>** d = {'dict': 1, 'dictionary': 2}

**>>>** d

{'dict': 1, 'dictionary': 2}

Во-вторых, используя функцию dict:

**>>>** d = dict(short='dict', long='dictionary')

**>>>** d

{'short': 'dict', 'long': 'dictionary'}

**>>>** d = dict([(1, 1), (2, 4)])

**>>>** d

{1:1, 2:4}

В-третьих, используя элемент ключей:

**>>>** d = dict.fromkeys(['a', 'b'])

**>>>** d

{'a': None, 'b': None}

**>>>** d = dict.fromkeys(['a', 'b'], 100)

**>>>** d

{'a': 100, 'b': 100}

В-четвертых, словари, очень похожие на генераторы списков.

с помощью генераторов.

**>>>** d = {a: a \*\* 2 **for** a **in** range(7)}

**>>>** d

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}

Словарные методы

dict.clear() — очищает словарь.

dict.copy() — возвращает копию словаря.

classmethod dict.fromkeys(seq[,value]) — Создает словарь последовательных ключей и значений (по умолчанию).

dict.get(key[,default]) — возвращает значение ключа, но не выдает исключение, если его не существует, возвращает значения по умолчанию (по умолчанию — None).

dict.items() — возвращает пары (ключ, значение).

dict.keys() — возвращает ключи в словаре.

dict.pop (key [, default]) — удаляет ключ и возвращает значение. Если ключ не существует, он возвращает значение по умолчанию (выдает исключение по умолчанию).

dict.popitem() — удаляет и возвращает пару (ключ, значение). Если словарь пуст, он выдает KeyError. Обратите внимание, что словари неупорядочены.

dict.setdefault(key [, default]) — возвращает значение ключа, но если значение не выбрано, создается ключ по умолчанию (по умолчанию)

dict.update([other]) — обновляет словарь, добавляя другие пары (ключ, значение). Существующие ключи будут перезаписаны. Никакого реверса (не новый словарь!).

dict.values() — возвращает значение словаря.

НастроитьCWD

import os

def setCurrentWorkingDirectory(directoryName):

programDirectory = os.path.dirname(os.path.abspath(\_\_file\_\_))

newCurrentWorkingDirectory = os.path.join(programDirectory, directoryName)

os.chdir(newCurrentWorkingDirectory)

Функция этой программы — установить рабочий каталог относительно того, где находится папка программы.

Используется модуль os, он позволяет работать с файловой системой, ограждением, управлять процессами.

В нашем случае с помощью метода os.path мы переходим в текущую папку.

*StandardizeStats*

from nba\_api.stats.endpoints import leaguedashteamstats

import statistics

from getStats import getStatsForTeam

import time

from customHeaders import customHeaders

def basicOrAdvancedStatMean(startDate, endDate, stat,statType = 'Base', season='2018-19'):

time.sleep(.2)

allTeamsInfo = leaguedashteamstats.LeagueDashTeamStats(per\_mode\_detailed='Per100Possessions',

measure\_type\_detailed\_defense=statType,

date\_from\_nullable=startDate,

date\_to\_nullable=endDate,

season=season,

headers=customHeaders,

timeout=120)

allTeamsDict = allTeamsInfo.get\_normalized\_dict()

allTeamsList = allTeamsDict['LeagueDashTeamStats']

specificStatAllTeams = []

for i in range(len(allTeamsList)):

specificStatAllTeams.append(allTeamsList[i][stat])

mean = statistics.mean(specificStatAllTeams)

return mean

def basicOrAdvancedStatStandardDeviation(startDate, endDate, stat,statType = 'Base', season='2018-19'):

time.sleep(.2)

allTeamsInfo = leaguedashteamstats.LeagueDashTeamStats(per\_mode\_detailed='Per100Possessions',

measure\_type\_detailed\_defense=statType,

date\_from\_nullable=startDate,

date\_to\_nullable=endDate,

season=season,

headers=customHeaders,

timeout=120

)

allTeamsDict = allTeamsInfo.get\_normalized\_dict()

allTeamsList = allTeamsDict['LeagueDashTeamStats']

specificStatAllTeams = []

for i in range(len(allTeamsList)):

specificStatAllTeams.append(allTeamsList[i][stat])

standardDeviation = statistics.stdev(specificStatAllTeams)

return standardDeviation

def basicOrAdvancedStatZScore(observedStat, mean, standardDeviation):

zScore = (observedStat-mean)/standardDeviation

return(zScore)

В этой части программы используется несколько библиотек.

Модуль nba\_api необходим для того, чтобы мы могли общаться с сайтом, который может получать различную статистику матчей НБА.

API (интерфейс прикладного программирования) — это контракт, предлагаемый приложением. Делаются заявления типа «Со мной можно связаться, используя эти настройки».

Контракт включает в себя следующие пункты:

- сама операция, которую мы можем выполнить;

- входящие данные;

- исходящие данные.

Модуль Статистика отвечает за математическую статистику. Этот модуль предоставляет функции для расчета математической статистики числовых данных.

Модуль «Время» — это модуль, который предоставляет различные функции, связанные со временем.

С помощью метода Time мы можем приостановить выполнение вызывающего потока на указанное количество секунд. Аргументом может быть число с плавающей запятой, обозначающее фактическое время сна. Фактическое время задержки может быть меньше запрошенного, поскольку любой задержанный сигнал происходит после того, как процедура сна() выполнила процедуру для перехвата этого сигнала. Кроме того, время задержки может быть произвольно больше, чем требуется при планировании других действий в системе.

В блоке, где мы импортируем необходимые модули, есть строки, получающие функции из соседних файлов с кодом Python. Например, getStats и customHeaders. Их содержание и функции мы рассмотрим ниже.

**CustomHeaders и getDailyMatchUps**

customHeaders = {

'Host': 'stats.nba.com',

'Connection': 'keep-alive',

'Cache-Control': 'max-age=0',

'Upgrade-Insecure-Requests': '1',

'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_5) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/75.0.3770.100 Safari/537.36',

'Accept':

'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed

exchange;v=b3',

'Accept-Encoding': 'gzip, deflate, br',

'Accept-Language': 'en-US,en;q=0.9',

'Referer': 'https://stats.nba.com/',

'x-nba-stats-origin': 'stats',

'x-nba-stats-token': 'true',

}

Запросы используются так, как будто они поступают из браузера. Это также записано в словаре.

Основная функция — поиск ежедневных игр НБА.

from nba\_api.stats.endpoints import leaguegamelog, scoreboard

from teamIds import teams

from customHeaders import customHeaders

def dailyMatchupsPast(date, season):

dailyMatchups = leaguegamelog.LeagueGameLog(season=season, league\_id='00', season\_type\_all\_star='Regular

Season', date\_from\_nullable=date,date\_to\_nullable=date, headers=customHeaders,timeout=60)

dailyMatchupsDict = dailyMatchups.get\_normalized\_dict()

listOfTeams = dailyMatchupsDict['LeagueGameLog']

winLossList = []

homeAwayDict = {}

for i in range(0,len(listOfTeams),2):

if '@' in listOfTeams[i]['MATCHUP']:

awayTeam = listOfTeams[i]['TEAM\_NAME']

homeTeam = listOfTeams[i+1]['TEAM\_NAME']

winLossList.append(listOfTeams[i+1]['WL'])

else:

awayTeam = listOfTeams[i+1]['TEAM\_NAME']

homeTeam = listOfTeams[i]['TEAM\_NAME']

winLossList.append(listOfTeams[i]['WL'])

homeAwayDict.update({homeTeam:awayTeam})

matchupsResultCombined = [homeAwayDict, winLossList]

return(matchupsResultCombined)

def dailyMatchupsPresent(date):

dailyMatchups = scoreboard.Scoreboard(league\_id='00', game\_date=date, headers=customHeaders, timeout=120)

dailyMatchupsDict = dailyMatchups.get\_normalized\_dict()

listOfGames = dailyMatchupsDict['GameHeader']

homeAwayDict = {}

for game in listOfGames:

homeTeamID = game['HOME\_TEAM\_ID']

for team, teamID in teams.items():

if teamID == homeTeamID:

homeTeamName = team

awayTeamID = game['VISITOR\_TEAM\_ID']

for team, teamID in teams.items():

if teamID == awayTeamID:

awayTeamName = team

homeAwayDict.update({homeTeamName:awayTeamName})

return homeAwayDict

Этот код содержит две функции:

- ежедневные матчипрошлые матчи;

- DailyMatchupsPresent.

Как следует из названия, с помощью этих функций мы получаем список игр на указанные даты. Только первый из них: в первом перечислены прошлые игры, во втором — текущие игры. В обоих случаях функция возвращает список, где индекс 0 — словарь.

и индекс 1 — результат игры. Дату необходимо вводить в формате мм/дд/гггг, а время – в формате гггг/гг.

*QetStats — получить статистику*

В НБА сбор статистики — ключевая функция любой команды.

from teamIds import teams

from nba\_api.stats.endpoints import teamdashboardbygeneralsplits, leaguedashteamstats

import time

from customHeaders import customHeaders

def getStatsForTeam(team, startDate, endDate, season='2019-20'):

time.sleep(1)

generalTeamInfo = teamdashboardbygeneralsplits.TeamDashboardByGeneralSplits(team\_id=teams[team],

per\_mode\_detailed='Per100Possessions', date\_from\_nullable=startDate, date\_to\_nullable=endDate, season=season,

headers=customHeaders, timeout=120)

generalTeamDict = generalTeamInfo.get\_normalized\_dict()

generalTeamDashboard = generalTeamDict['OverallTeamDashboard'][0]

winPercentage = generalTeamDashboard['W\_PCT']

rebounds = generalTeamDashboard['REB']

turnovers = generalTeamDashboard['TOV']

plusMinus = generalTeamDashboard['PLUS\_MINUS']

advancedTeamInfo = teamdashboardbygeneralsplits.TeamDashboardByGeneralSplits(team\_id=teams[team],

measure\_type\_detailed\_defense='Advanced', date\_from\_nullable=startDate, date\_to\_nullable=endDate,

season=season, headers=customHeaders, timeout=120)

advancedTeamDict = advancedTeamInfo.get\_normalized\_dict()

advancedTeamDashboard = advancedTeamDict['OverallTeamDashboard'][0]

offensiveRating = advancedTeamDashboard['OFF\_RATING']

defensiveRating = advancedTeamDashboard['DEF\_RATING']

trueShootingPercentage = advancedTeamDashboard['TS\_PCT']

allStats = {

'W\_PCT':winPercentage,

'REB':rebounds,

'TOV':turnovers,

'PLUS\_MINUS':plusMinus,

'OFF\_RATING':offensiveRating,

'DEF\_RATING': defensiveRating,

'TS\_PCT':trueShootingPercentage,

}

return allStats

Написана функция GetStatsForTeam. Его задача — возвращать различную статистику в словарь для введенной команды. Чтобы все работало корректно, должны быть соблюдены следующие условия:

— если команда соответствует названию команды в файле TeamIds.py;

- startDate (дата начала) и endDate (дата окончания) должны быть записаны в формате дд/мм/гггг.

В результате выполнения этого кода мы получаем словарь allStats со статистикой определенной группы.

*TeamIds и createModel*

Этот код содержит словарь индивидуальных идентификаторов каждой команды НБА на сайте stats.nba.com.

teams = {

"Atlanta Hawks": 1610612737,

"Boston Celtics":1610612738,

"Brooklyn Nets": 1610612751,

"Charlotte Hornets": 1610612766,

"Charlotte Bobcats": 1610612766,

"Chicago Bulls": 1610612741,

"Cleveland Cavaliers": 1610612739,

"Dallas Mavericks": 1610612742,

"Denver Nuggets": 1610612743,

"Detroit Pistons": 1610612765,

"Golden State Warriors": 1610612744,

"Houston Rockets": 1610612745,

"Indiana Pacers": 1610612754,

"LA Clippers": 1610612746,

"Los Angeles Clippers": 1610612746,

"Los Angeles Lakers": 1610612747,

"Memphis Grizzlies": 1610612763,

"Miami Heat": 1610612748,

"Milwaukee Bucks": 1610612749,

"Minnesota Timberwolves": 1610612750,

"New Jersey Nets": 1610612751,

"New Orleans Hornets": 1610612740,

"New Orleans Pelicans": 1610612740,

"New York Knicks": 1610612752,

"Oklahoma City Thunder": 1610612760,

"Orlando Magic": 1610612753,

"Philadelphia 76ers": 1610612755,

"Phoenix Suns": 1610612756,

"Portland Trail Blazers": 1610612757,

"Sacramento Kings": 1610612758,

"San Antonio Spurs": 1610612759,

"Toronto Raptors": 1610612761,

"Utah Jazz": 1610612762,

"Washington Wizards": 1610612764,

}

В этом коде мы создаем, обучаем и тестируем модель для прогнозирования исхода матчей НБА.

from standardizeStats import basicOrAdvancedStatZScore, basicOrAdvancedStatStandardDeviation,

basicOrAdvancedStatMean

from getDailyMatchups import dailyMatchupsPast

from getStats import getStatsForTeam

from availableStats import availableStats

from configureCWD import setCurrentWorkingDirectory

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

import pandas as pd

import pickle

from datetime import timedelta, date

def zScoreDifferential(observedStatHome, observedStatAway, mean, standardDeviation):

homeTeamZScore = basicOrAdvancedStatZScore(observedStatHome, mean, standardDeviation)

awayTeamZScore = basicOrAdvancedStatZScore(observedStatAway, mean, standardDeviation)

differenceInZScore = homeTeamZScore - awayTeamZScore

return differenceInZScore

def infoToDataFrame(dailyGames, meanDict, standardDeviationDict, startDate, endDate, season):

fullDataFrame = []

gameNumber = 0

dailyResults = dailyGames[1]

for homeTeam,awayTeam in dailyGames[0].items():

homeTeamStats = getStatsForTeam(homeTeam, startDate, endDate, season)

awayTeamStats = getStatsForTeam(awayTeam, startDate, endDate, season)

currentGame = [homeTeam,awayTeam]

for stat,statType in availableStats.items():

zScoreDif = zScoreDifferential(homeTeamStats[stat], awayTeamStats[stat], meanDict[stat],

standardDeviationDict[stat])

currentGame.append(zScoreDif)

if dailyResults[gameNumber] == 'W':

result = 1

else:

result = 0

currentGame.append(result)

gameNumber += 1

print(currentGame)

fullDataFrame.append(currentGame)

return(fullDataFrame)

def daterange(startDate, endDate):

for n in range(int ((endDate - startDate).days)):

yield startDate + timedelta(n)

def createMeanStandardDeviationDicts(startDate, endDate, season):

meanDict = {}

standardDeviationDict = {}

for stat, statType in availableStats.items():

statMean = basicOrAdvancedStatMean(startDate, endDate, stat, statType, season)

meanDict.update({stat: statMean})

statStandardDeviation = basicOrAdvancedStatStandardDeviation(startDate, endDate, stat, statType, season)

standardDeviationDict.update({stat: statStandardDeviation})

bothDicts = []

bothDicts.append(meanDict)

bothDicts.append(standardDeviationDict)

return bothDicts

def getTrainingSet(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startOfSeason):

startDate = date(startYear, startMonth, startDay)

endDate = date(endYear, endMonth, endDay)

startDateFormatted = startDate.strftime("%m/%d/%Y")

allGames = []

for singleDate in daterange(startDate, endDate):

currentDate = singleDate.strftime("%m/%d/%Y")

print(currentDate)

previousDay = singleDate - timedelta(days=1)

previousDayFormatted = previousDay.strftime("%m/%d/%Y")

meanAndStandardDeviationDicts = createMeanStandardDeviationDicts(startOfSeason, previousDayFormatted,

season)

meanDict = meanAndStandardDeviationDicts[0]

standardDeviationDict = meanAndStandardDeviationDicts[1]

currentDayGames = dailyMatchupsPast(currentDate, season)

currentDayGamesAndStatsList = infoToDataFrame(currentDayGames, meanDict, standardDeviationDict,

startOfSeason, previousDayFormatted, season) # Formats Z Score difs for games on current date in loop

for game in currentDayGamesAndStatsList:

game.append(currentDate)

allGames.append(game)

print(allGames)

return(allGames)

def createDataFrame(listOfGames):

games = pd.DataFrame(

listOfGames,

columns=['Home', 'Away', 'W\_PCT', 'REB', 'TOV', 'PLUS\_MINUS', 'OFF\_RATING', 'DEF\_RATING',

'TS\_PCT', 'Result', 'Date']

)

print(games)

return(games)

def performLogReg(dataframe):

featureColumns = ['W\_PCT', 'REB', 'TOV', 'PLUS\_MINUS', 'OFF\_RATING', 'DEF\_RATING', 'TS\_PCT']

X = dataframe[featureColumns]

Y = dataframe.Result

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.25, shuffle=True)

logreg = LogisticRegression()

logreg.fit(X\_train, Y\_train) Y\_pred = logreg.predict(X\_test)

confusionMatrix = metrics.confusion\_matrix(Y\_test, Y\_pred)

print('Coefficient Information:')

for i in range(len(featureColumns)):

logregCoefficients = logreg.coef\_

currentFeature = featureColumns[i]

currentCoefficient = logregCoefficients[0][i]

print(currentFeature + ': ' + str(currentCoefficient))

print('----------------------------------')

print("Accuracy:", metrics.accuracy\_score(Y\_test, Y\_pred))

print("Precision:", metrics.precision\_score(Y\_test, Y\_pred))

print("Recall:", metrics.recall\_score(Y\_test, Y\_pred))

print('----------------------------------')

print('Confusion Matrix:')

print(confusionMatrix)

return logreg

def saveModel(model, filename):

setCurrentWorkingDirectory('SavedModels')

with open(filename, 'wb') as file:

pickle.dump(model, file)

def createModel(startYear=None, startMonth=None, startDay=None, endYear=None, endMonth=None,

endDay=None, season='2018-19', startOfSeason = '10/16/2018', filename='model.pkl'):

setCurrentWorkingDirectory('Data')

allGamesDataframe = pd.read\_csv('COMBINEDgamesWithInfo2016-19.csv')

logRegModel = performLogReg(allGamesDataframe)

saveModel(logRegModel, filename)

В этом коде мы импортируем ранее неиспользованные библиотеки Python, а именно:

- Sklearn;

- pandas;

- pickle.

Scikit-learn — бесплатная библиотека машинного обучения для Python. Он имеет различные алгоритмы, такие как векторная машина, случайные леса и k-соседи, а также численные алгоритмы Python, такие как NumPy и SciPy.

и поддерживает научные библиотеки.

Pandas — это библиотека Python высокого уровня для анализа данных. Я называю его высокоуровневым, потому что он построен на основе низкоуровневой библиотеки NumPy (написанной на C), что является большим плюсом. В экосистеме Python pandas — это самая продвинутая и быстро развивающаяся библиотека для обработки и анализа данных. Мне приходится использовать его почти ежедневно в работе, так что что-нибудь в будущем

Если забуду, постараюсь написать кратко.

Модуль Pickle используется для сериализации и повторной реализации структуры объектов Python. Любой объект Python можно выбрать для сохранения на диск. Это создает рассол, поэтому сначала «сериализует» объект перед записью в файл. Пиклинг — это способ преобразования объекта Python (списка, словаря и т. д.) в поток символов. Эта идея содержит всю информацию, необходимую для восстановления объекта в другом скрипте Python.

В коде прописаны следующие функции:

- zScoreDifferential;

- infoToDataFrame;

- daterange;

- createMeanStandardDeviationDicts;

- getTrainingSet;

- createDataFrame;

- performLogReg;

- savemodel;

- createMode.l

zScoreDifferential вычисляет «разницу zScore» между двумя командами для указанной статистики.

infoToDataFrame используется для объединения и форматирования всех данных, которые будут помещены в фрейм данных pandas.

Daterange — это функция, которая позволяет выполнять итерацию от заданной даты начала (startDate) до конечной даты (endDate).

Функция createMeanStandardDeviationDicts возвращает список, индекс которого 0 является средним значением каждой метрики.

Индекс 1 — это словарь, содержащий стандартное отклонение каждой метрики.getTrainingSet (буквально — получить обучающий набор) — перебирает каждый день между началом и концом и добавляет каждую игру в единый список для возврата.

createDataFrame возвращает кадр данных из списка игр с «дифференциалами zScore».

PerformLogReg Функция создает модель логистической регрессии и проверяет точность модели.

Savemodel сохраняет модель в пакете для дальнейшего использования. Create Model используется для создания новых моделей логистической регрессии. Можно импортировать статистику и прогнозы для каждой игры из CSV-файла или создавать самостоятельно.

*nbaPredict*

Эта часть программы - игры НБА указанного дня.

прогнозирует результаты. Его часто используют для прогнозирования. Используются две библиотеки Python и импортируются другие функции.

import pickle

import pandas as pd

from getDailyMatchups import dailyMatchupsPresent

from createModel import createMeanStandardDeviationDicts, zScoreDifferential

from availableStats import availableStats

from getStats import getStatsForTeam

from configureCWD import setCurrentWorkingDirectory

def dailyGamesDataFrame(dailyGames, meanDict, standardDeviationDict, startDate, endDate, season):

fullDataFrame = []

for homeTeam,awayTeam in dailyGames.items():

homeTeamStats = getStatsForTeam(homeTeam, startDate, endDate, season)

awayTeamStats = getStatsForTeam(awayTeam, startDate, endDate, season)

currentGame = [homeTeam,awayTeam]

for stat,statType in availableStats.items():

zScoreDif = zScoreDifferential(homeTeamStats[stat], awayTeamStats[stat], meanDict[stat],

standardDeviationDict[stat])

currentGame.append(zScoreDif)

fullDataFrame.append(currentGame)

return(fullDataFrame)

def predictDailyGames(currentDate, season, startOfSeason):

dailyGames = dailyMatchupsPresent(currentDate)

meanDict, standardDeviationDict = createMeanStandardDeviationDicts(startOfSeason, currentDate, season)

dailyGamesList = dailyGamesDataFrame(dailyGames, meanDict, standardDeviationDict, startOfSeason,

currentDate, season)

gamesWithZScoreDifs = pd.DataFrame(

dailyGamesList,

columns=['Home', 'Away', 'W\_PCT', 'REB', 'TOV', 'PLUS\_MINUS', 'OFF\_RATING', 'DEF\_RATING',

'TS\_PCT']

)

justZScoreDifs = gamesWithZScoreDifs.loc[:,'W\_PCT':'TS\_PCT']

with open('finalized\_model.pkl', 'rb') as file:

pickleModel = pickle.load(file)

predictions = pickleModel.predict\_proba(justZScoreDifs)

gamesWithPredictions = [dailyGames, predictions]

return gamesWithPredictions

def interpretPredictions(gamesWithPredictions):

dailyGames = gamesWithPredictions[0]

probabilityPredictions = gamesWithPredictions[1]

for gameNum in range(len(probabilityPredictions)):

winProb = probabilityPredictions[gameNum][1]

winProbRounded = round(winProb,4)

winProbPercent = "{:.2%}".format(winProbRounded)

homeTeam = list(dailyGames.keys())[gameNum]

awayTeam = list(dailyGames.values())[gameNum]

print('There is a ' + winProbPercent + ' chance that the ' + homeTeam + ' will defeat the ' + awayTeam + '.')

def makeInterpretPredictions(currentDate, season, startOfSeason):

setCurrentWorkingDirectory('SavedModels')

print('Predictions for ' + currentDate + ':')

predictions = predictDailyGames(currentDate, season, startOfSeason)

interpretPredictions(predictions)

makeInterpretPredictions('12/12/2019', '2019-20', '10/22/2019')

Прописаны следующие функции:

- dailyGamesDataFrame;

- predictDailyGames;

- interpretPredictions;

- makeinterpretPredictions.

Daily Games Data Frame-Z Score возвращает список дифференциальных игр между командами, помещенными в фрейм данных pandas.

PredictDailyGames 0 – индекс dailyGames возвращает список в виде словаря в формате {Дома: На выезде}. Указатель 1. Список прогнозов для каждой игры. Интерпретация. Прогнозы на победу команды гостей в каждой игре.

makeinterpretPredictions выводит игры на заданную дату и возвращает прогнозы для каждой игры.

В результате выполнения этого кода мы получаем следующий вывод:

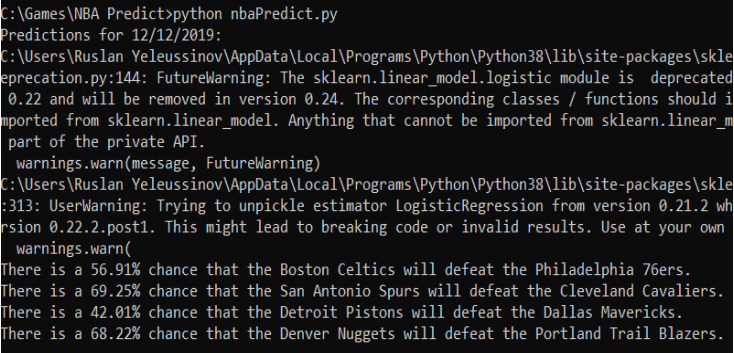


Рисунок 3.1 — результат работы программы nbaPredict.

*MakePastPredictions*

Используется для прогнозирования определенного диапазона прошлых игр НБА. Очень похоже на NbaPredict, с той лишь разницей, что в результате мы получаем результаты прогнозирования за прошедший период времени, а не за текущий день.

import pickle

import pandas as pd

from createModel import getTrainingSet, createDataFrame

from configureCWD import setCurrentWorkingDirectory

def getTrainingSetCSV(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,

filename='gamesWithInfo.csv'):

rangeOfGames = getTrainingSet(startYear, startMonth, startDay, endYear, endMonth, endDay, season,

startDateOfSeason)

rangeOfGamesDataframe = createDataFrame(rangeOfGames)

setCurrentWorkingDirectory('Data')

rangeOfGamesDataframe.to\_csv(filename)

def getPredictionsCSV(gameDataFilename, outputFilename):

setCurrentWorkingDirectory('Data')

gamesWithZScoreDifs = pd.read\_csv(gameDataFilename)

withoutNums = gamesWithZScoreDifs.loc[:, 'Home':'Date']

justZScoreDifs = gamesWithZScoreDifs.loc[:, 'W\_PCT':'TS\_PCT']

setCurrentWorkingDirectory('SavedModels')

with open('finalized\_model.pkl', 'rb') as file:

pickleModel = pickle.load(file)

predictions = pickleModel.predict(justZScoreDifs)

probPredictions = pickleModel.predict\_proba(justZScoreDifs)

numCorrect = 0

numWrong = 0

allGames = []

for i in range(len(probPredictions)):

winProbability = probPredictions[i][1]

homeTeam = withoutNums.iloc[i, 0]

awayTeam = withoutNums.iloc[i, 1]

date = withoutNums.iloc[i, 10]

currentGameWithPred = [date, homeTeam, awayTeam, winProbability]

allGames.append(currentGameWithPred)

predictionsDF = pd.DataFrame(

allGames,

columns=['Date', 'Home', 'Away', 'Home Team Win Probability']

)

setCurrentWorkingDirectory('Data')

predictionsDF.to\_csv(outputFilename)

value = withoutNums.iloc[i,9]

if value == predictions[i]:

numCorrect += 1

else :

numWrong += 1

print('Accuracy:')

print((numCorrect)/(numCorrect+numWrong))

def makePastPredictions(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,

gameDataFilename='gamesWithInfo.csv', outputFilename='predictions.csv'):

getTrainingSetCSV(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,

gameDataFilename)

getPredictionsCSV(gameDataFilename, outputFilename)

makePastPredictions(2018, 12, 28, 2019, 1, 1, '2018-19', '10/16/2018',

'gamesWithInfo.csv', 'predictions.csv')

Прописаны следующие функции:

- getTrainingSetCSV;

- getPredictionsCSV;

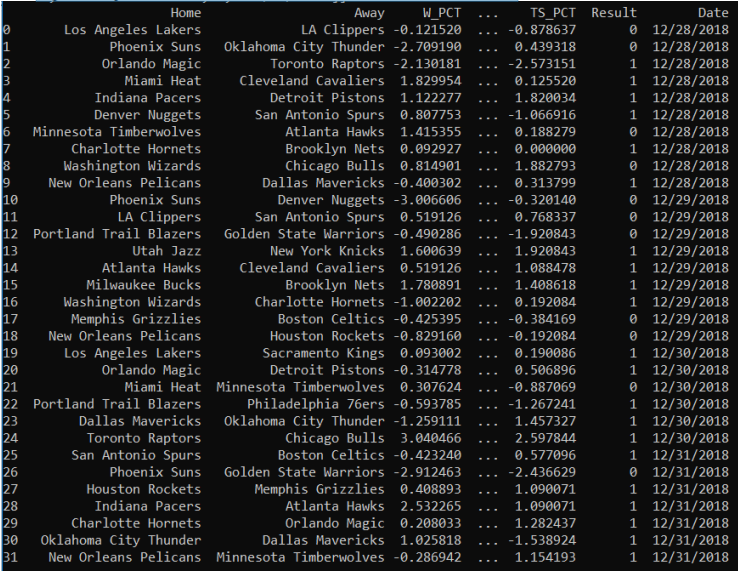
- makePastPredictions.

getTrainingSetCSV экспортирует информацию обо всех играх за указанный период времени в папку «data» внутри проекта.

getPredictionsCSV создает файл CSV, содержащий прогнозы для нескольких игр. Покажите точность модели в прогнозировании игр в заданный период времени.

makePastPredictions делает прогнозы прошлых игр для указанного диапазона времени, экспортирует их в CSV с информацией об игре.

В результате выполнения получаем следующий результат;



На рис. 3.2 показаны выходные данные программы makePastPredictions.

Здесь у нас есть шесть основных столбцов: команда хозяев, команда гостей, результат матча и дата. Кроме того, в папке «Данные» создается файл предсказания.csv с четырьмя столбцами: дата, команда хозяев, команда гостей и вероятность победы команды хозяев.

# 3.2. Создание бота для Telegram

Для начала нам необходимо зарегистрировать нашего будущего бота на Teegram. Это делается следующим образом:

- сначала вам необходимо установить приложение телеграмм на телефон или компьютер;

- нам нужно найти бот botfather для регистрации нового бота;

- нажмите кнопку старт и начните общение с ботом. Далее бот покажет нам следующий набор команд.

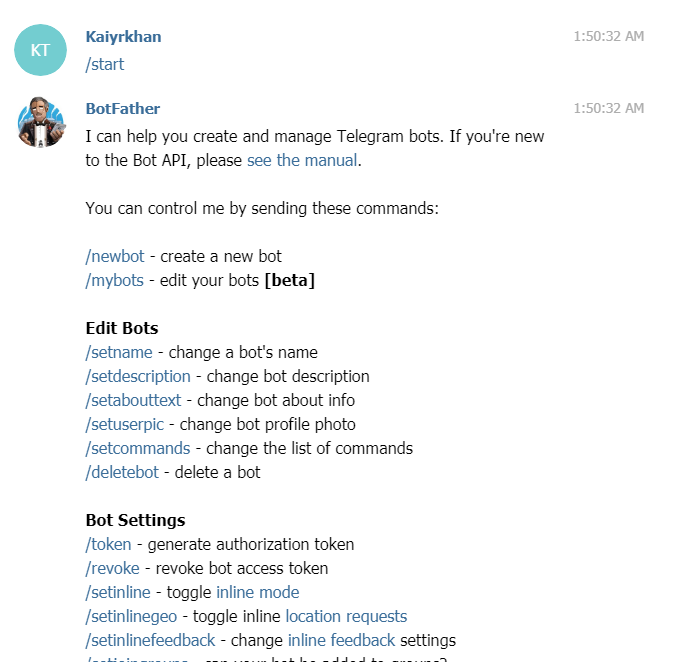


Рисунок 3.3 – Процесс создания Telegram-бота

- для создания нового бота необходимо выполнить команду /newbot. действуем по инструкции;

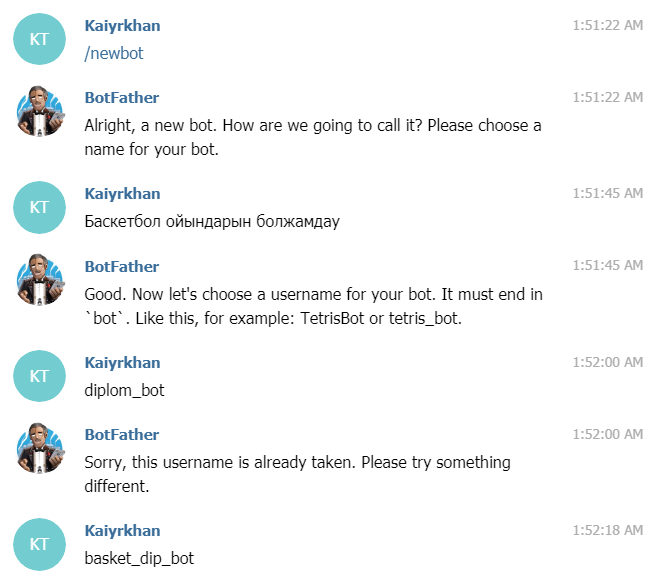


Рисунок 3.4 – Именование бота Telegram

- дав боту собственное имя, мы увидим следующий результат;

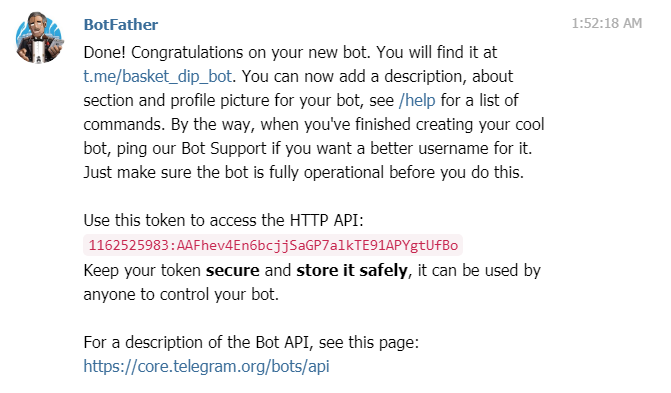


Рисунок 3.5 – Получение токена для Telegram-бота

— из этого результата нам нужен параметр токена, по которому мы сможем управлять ботом и разрешить работу с телеграмапи.

Создайте новый файл bot.py в папке, где сохранена основная программа. Выполняем в файле следующий код:

import logging

import telegram

import telebot

from nbaPredict import makeInterpretPredictions

class Request(object):

def \_\_init\_\_(self):

self.date = None

self.season = None

self.start\_season = None

STATUS = NON\_ASKING, DATE, SEASON, START\_SEASON = range(4)

TOKEN = "1162525983:AAFhev4En6bcjjSaGP7alkTE91APYgtUfBo"

ASK\_DATE = "Please enter the date in the following format: mm/dd/yyyy"

ASK\_SEASON = "Please enter the season in the following format: yyyy-yy"

ASK\_START\_SEASON = "Please enter the start date of season in the following format: mm/dd/yyyy"

ASK\_IN\_PROCESS = "Please wait 5-6 minutes, data are taken and program will reply as it will finish"

request\_params = Request()

cur\_status = NON\_ASKING

bot = telebot.TeleBot(TOKEN)

@bot.message\_handler(commands=['start'])

def start\_message(message):

bot.send\_message(message.chat.id, 'Works:)')

@bot.message\_handler(commands=['predict'])

def start\_message(message):

global cur\_status

cur\_status = DATE

bot.send\_message(message.chat.id, ASK\_DATE)

def send\_some\_message(chat\_id, text):

global bot

bot.send\_message(chat\_id, text)

def calc\_and\_send\_result(chat\_id, request\_params):

results = makeInterpretPredictions(request\_params.date, request\_params.season, request\_params.start\_season)

print (results)

send\_some\_message(chat\_id, "Results are following: \n" + results)

@bot.message\_handler(content\_types=['text'])

def start\_message(message):

global cur\_status

print ("OK")

if cur\_status == NON\_ASKING: return

elif (cur\_status == DATE):

request\_params.date = message.text

cur\_status = SEASON

send\_some\_message(message.chat.id, ASK\_SEASON)

elif (cur\_status == SEASON):

request\_params.season = message.text

cur\_status = START\_SEASON

send\_some\_message(message.chat.id, ASK\_START\_SEASON)

elif (cur\_status == START\_SEASON):

request\_params.start\_season = message.text

cur\_status = NON\_ASKING

print (request\_params)

send\_some\_message(message.chat.id, ASK\_IN\_PROCESS)

calc\_and\_send\_result(message.chat.id, request\_params)

bot.polling()

Запустите программу и войдите в коннектор Telegram. Нажав кнопку «Start», вы увидите следующий результат.

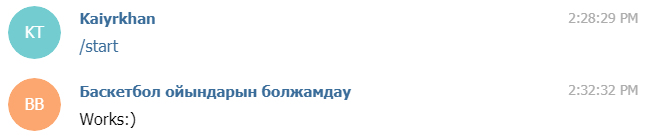


Рисунок 3.6 — результат команды запуска.

Бот ответит, что работает.

Ответ команды /predict:

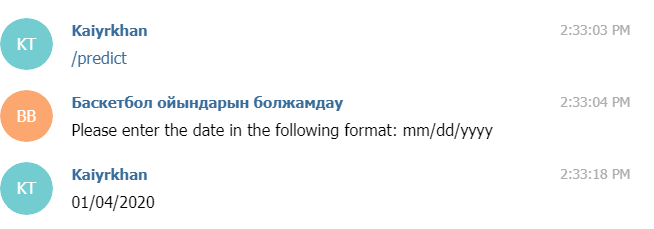


Рисунок 3.7 – Ввод даты игры

Бот задаст нам несколько вопросов, чтобы определить фактическую дату игры:

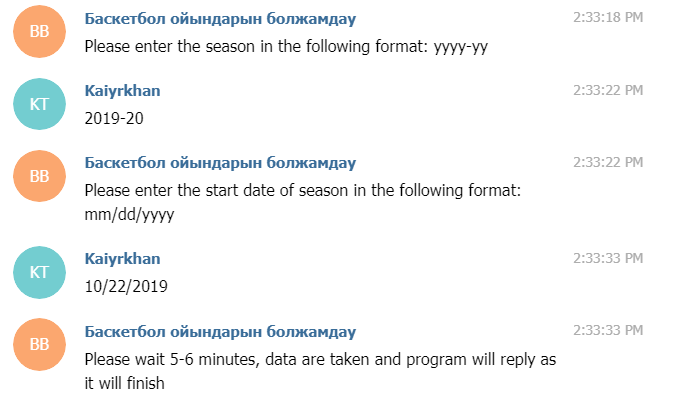


Рисунок 3.8 – Ввод сезона и даты начала сезона

После сбора необходимых данных бот через 5-6 минут выведет прогноз:

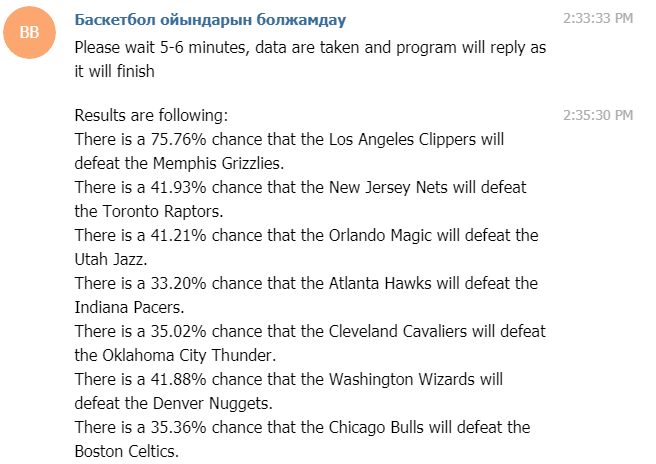


Рисунок 3.9 — результат работы бота