KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 2 урок > Введение в теорию распределенного МО

> Оглавление

> Жизненный цикл моделей

Raw Data

- Постановка задачи: какая задача будет решаться на существующих данных или данных, которые нужно собрать.
- Сбор данных для обучения моделей.
- Хранение данных (любые инструменты: Clickhouse, GreenPlum, облачные хранилища и т.д.).

Prep Data

- Выполнение обработки данных для их подготовки к непосредственному использованию в обучении моделей (любые инструменты: Apache Spark, pandas).
- Может применяться процесс Feature Engineering – определение или вычисление нового признака из существующих данных для использования в модели. Например, из данных о пользователях можно с некоторой вероятностью определить пол пользователя по другим косвенным признакам.

Training

- Определение с типом и классом задач, которые нужно решить
- Подбор типа/алгоритма обучения модели
- Обучение модели на данных
- Построение экспериментов
- Оптимизация гиперпараметров для повышения качества модели

Могут использоваться любые фреймворки: TensorFlow, PyTorchи т.д.

Deploy

После того, как модель реализована, её нужно применять, т.е. внедрить в процесс, в котором она должна участвовать.

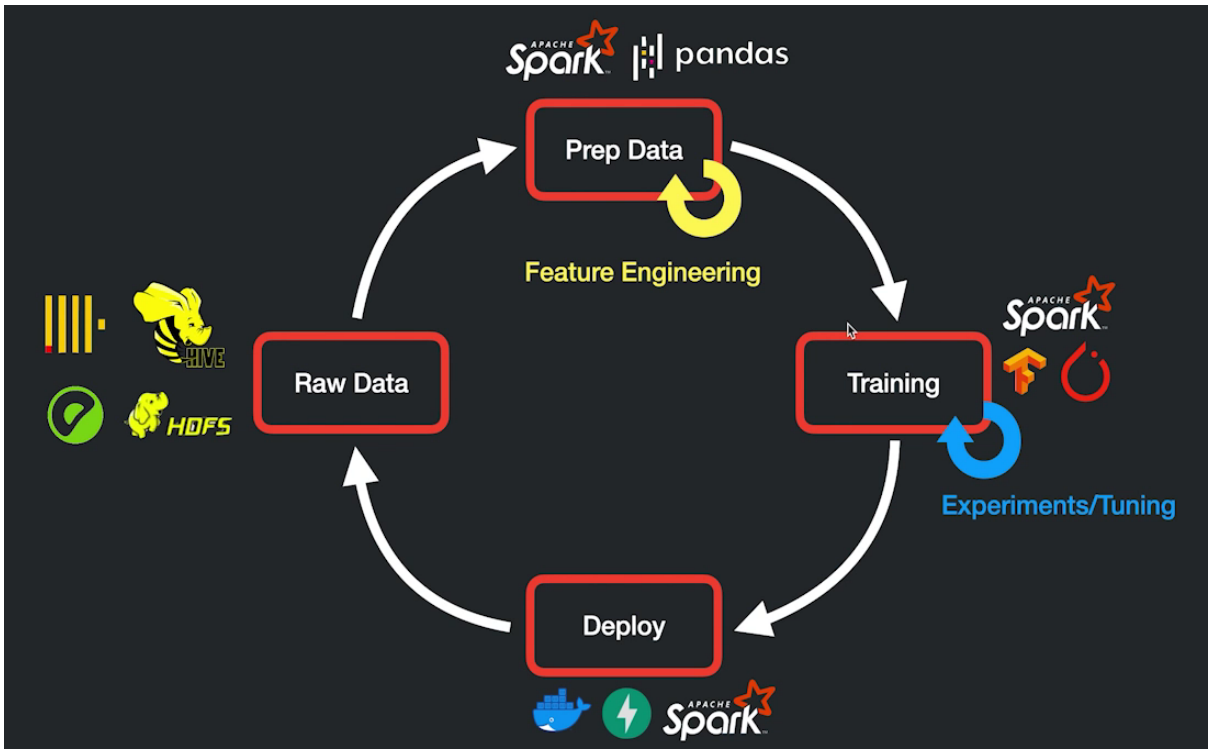
Разные подходы для внедрения:

- Если это компонент из Spark ML, то модель можно внедрить как Spark задачу или в поток данных, используя Spark Streaming
- Использовать как lambda функцию
- Использовать как микросервис, к которому будут обращаться сторонние сервисы

Также нужно построить мониторинг качества модели и целевой метрики.

Можно проводить A/B тестирования перед внедрением для того, чтобы удостовериться в том, что модель действительно повышает целевую метрику.

После внедрения можно собирать данные по работающей модели и на их основе строить новые выборки для улучшения модели, поиска особенностей данных.



> Протухание модели

Протухание модели - даже когда модель работает отлично, данные могут поменяться и вводить модель в заблуждение, или модель неправильно реагирует на новые поступающие данные.

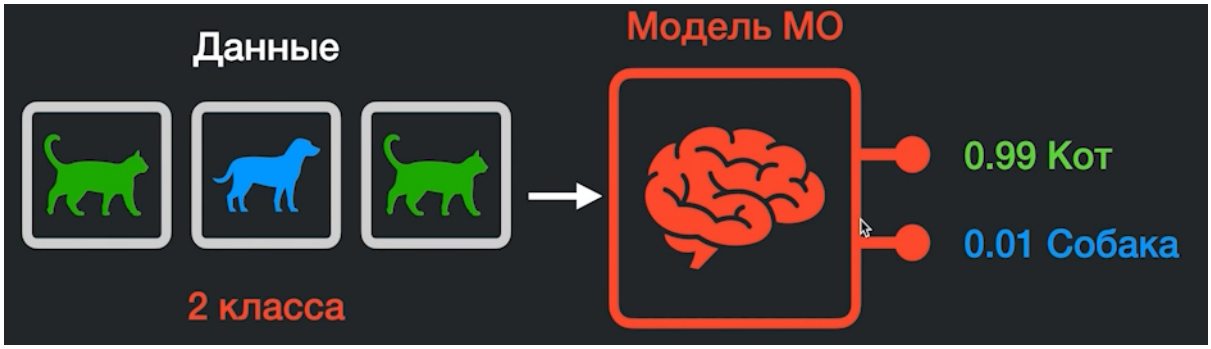
Например, при создании динамической цены модель использует данные об устройствах пользователей. С течением времени выходят новые устройства, и модель не знает как на них реагировать.

Период протухания модели может быть специфичным в зависимости от задачи и иметь сезонность.

> Сложность модели

> Простые модели

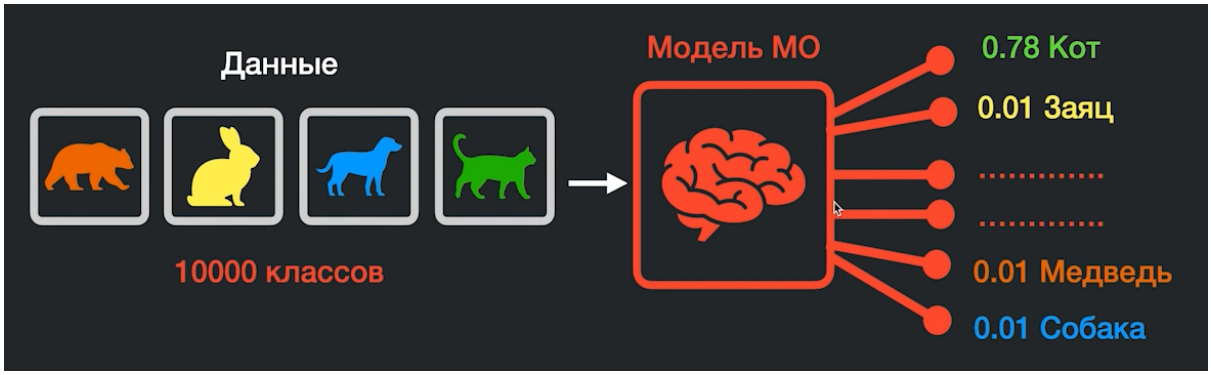
Рассмотрим бинарный классификатор, который определяет по фотографии какое животное изображено: кошка или собака. Применяется простая модель с небольшим набором параметров, её не нужно будет долго обучать. Такая модель может хорошо работать долгое время и не требовать дополнительных работ.



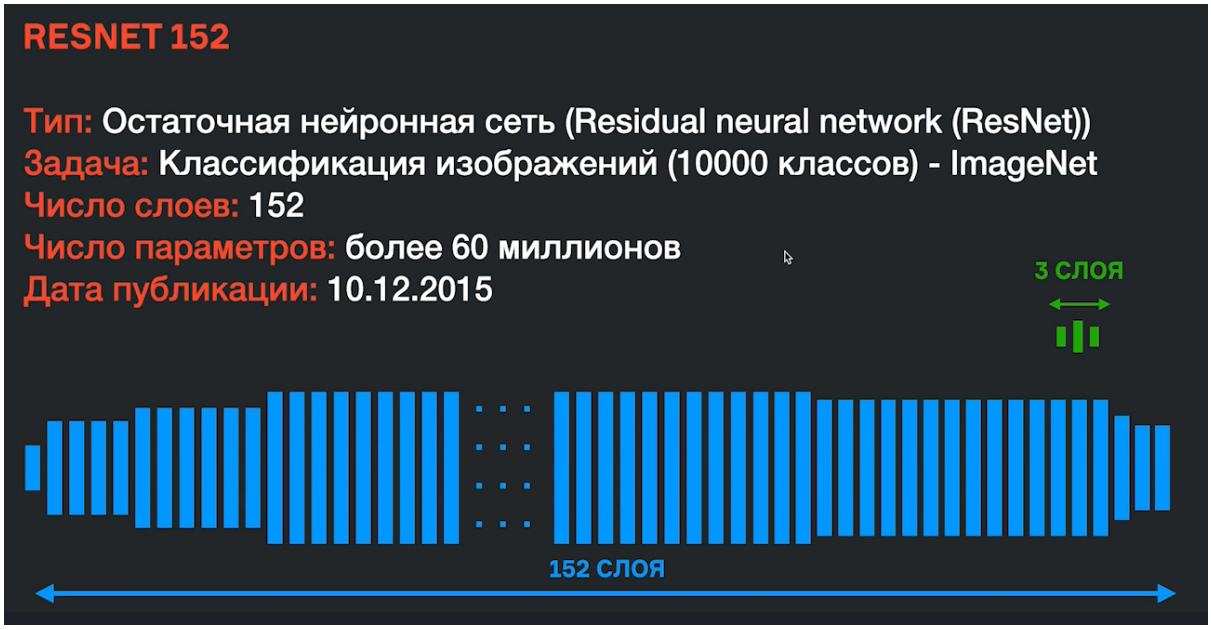
> Сложные модели

Зачастую нужно создавать более сложные модели, которые имеют большой объём параметров. Обучение может занимать долгое время.

Например, имеется 10000 классов животных, и модель должна обучиться их классифицировать. Простая модель с небольшим числом параметров не сможет её решить, нужна сложная с множеством параметров. Для таких задач хорошо подходят нейронные сети.



Пример сложной модели:



> Transfer learning

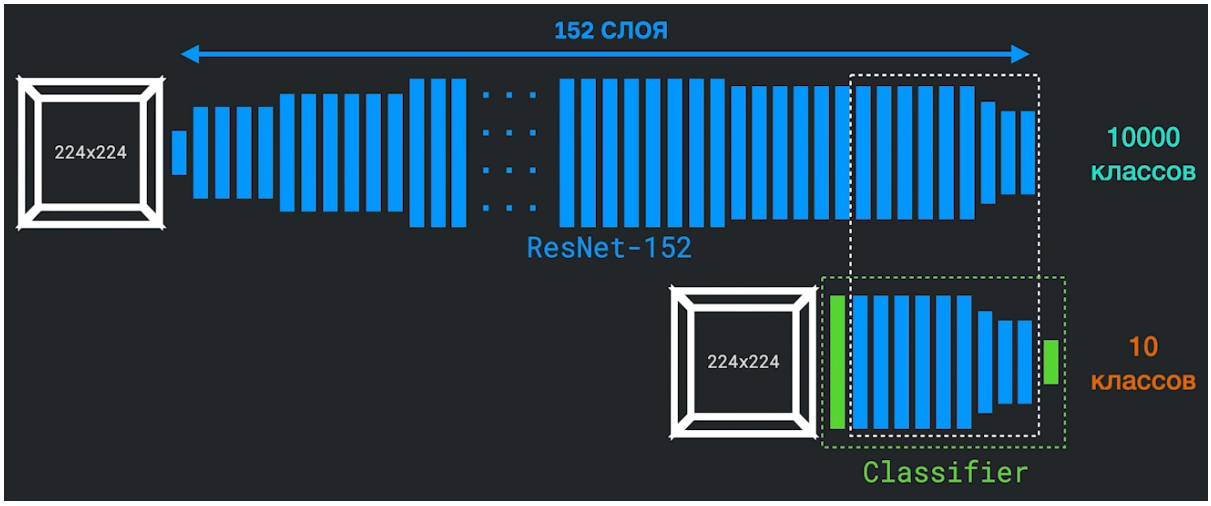
Трансферное обучение (TL) – применение знаний модели полученных в результате решения одной задачи, к другой целевой задаче.

Например, использование ResNet-152, который может классифицировать 10000 классов животных, для бинарной классификации (что изображено: собака или кошка, медведь или заяц).

Существует несколько подходов в TL, для того чтобы применять готовую модель для конкретной задачи.

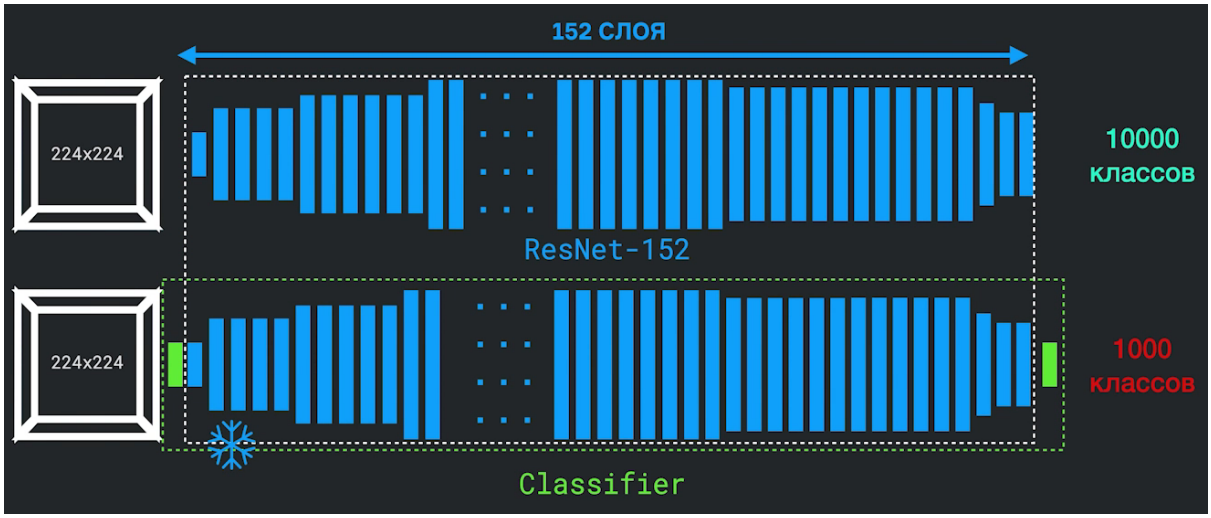
Использование последних слоёв модели.

Для классификации 10 классов можно взять только последние слои ResNet-152, обернув их во входной и выходной слои на 10 классов, и дообучить. Обучение такой модели будет более быстрым.



Использование обученной модели.

Возьмём ResNet-152, обернём в слои, создав новую нейросеть, заморозим слои изначальной модели, и полученную модель можно использовать для классификации более узкого набора классов. Например, на 1000 классов животных. Более затратный по сравнению с предыдущим подход в плане обучения.



> Распределённое МО

Обучать сложные модели гораздо выгоднее, используя распределённое МО, т.к. это экономит время и позволяет более эффективно использовать ресурсы.

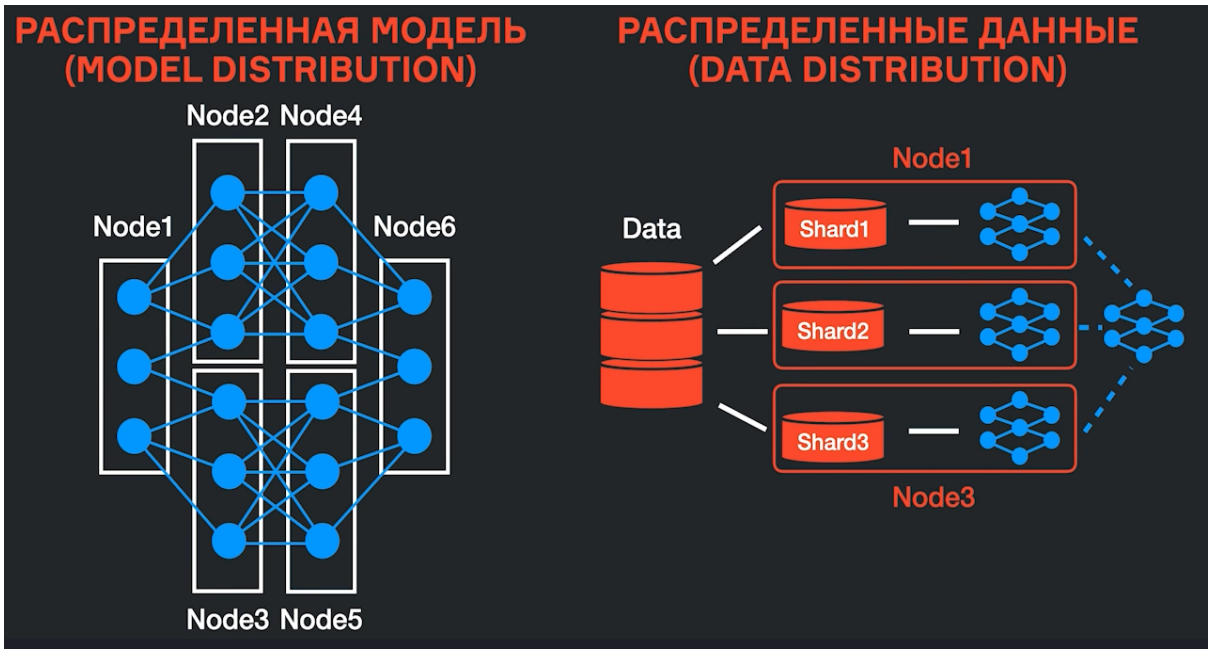
Обучать модель можно стандартно, используя один поток/тред. Такой вариант подойдёт для простой модели и простого набора данных.

Если объём данных большой, а модель простая, то можно обучать её, используя множество процессов на одном компьютере.

Существует другой подход: если модель простая или сложная, и имеется большой набор данных, то можно использовать распределённое МО. Есть кластер, состоящий из нескольких компьютеров, которые мы можем задействовать в обучении модели.

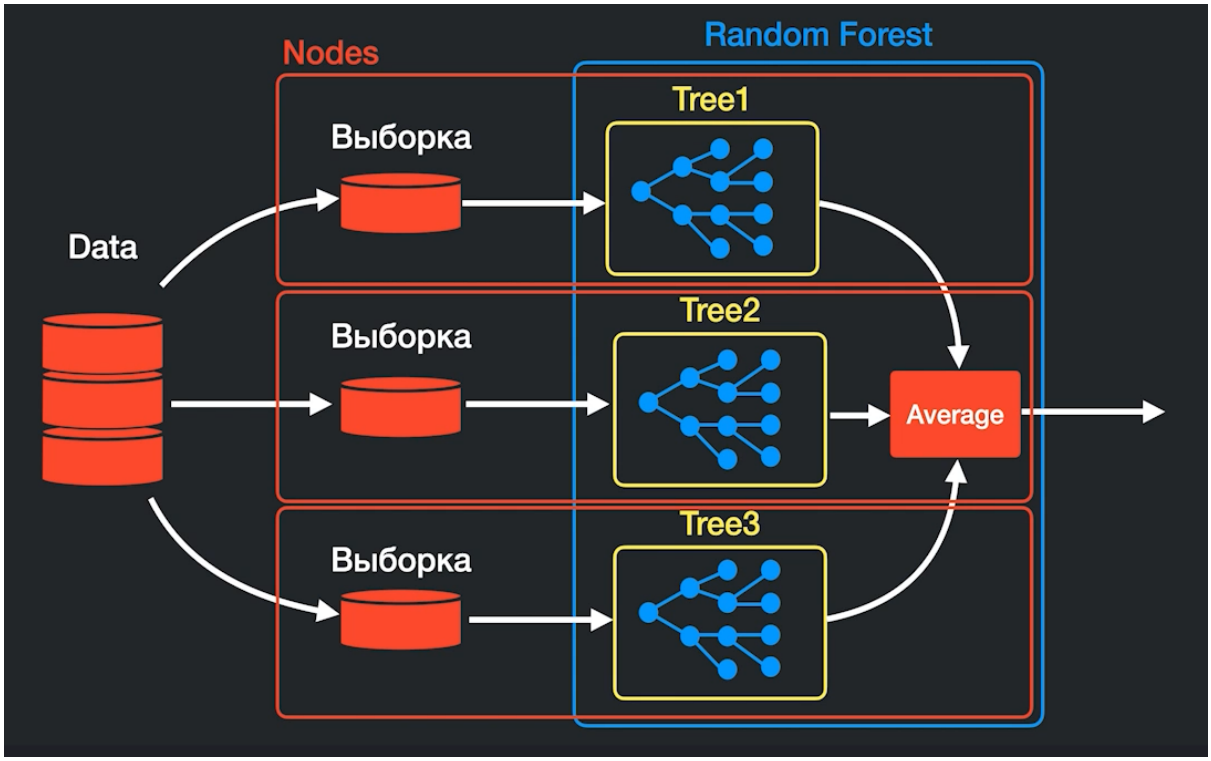
Распределённое МО делится на 2 группы:

1. Распределённая модель (model distribution) – разделение сложной модели и параллельное обучение на объёме данных с дальнейшей синхронизацией.
2. Распределённые данные (data distribution) – доставка модели до каждой из партиций большого объёма данных с дальнейшей синхронизацией.



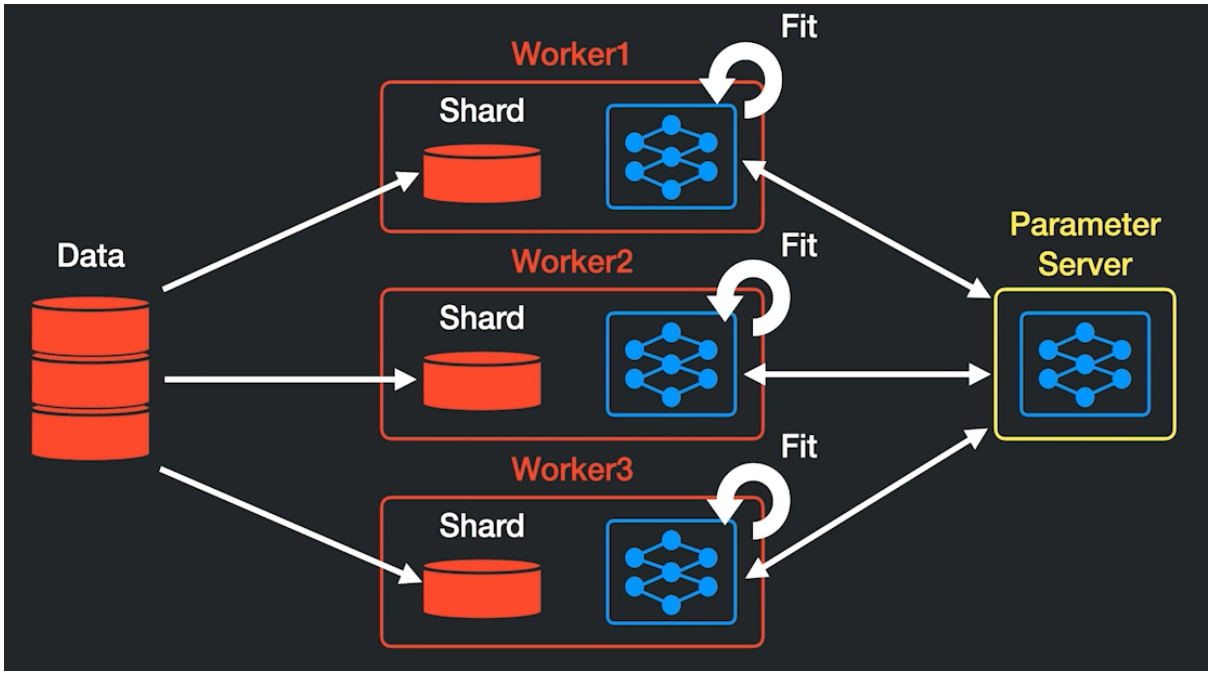
> Распределённая модель

Рассмотрим обучение модели случайного леса (Random Forest). Лес состоит из некоторого числа деревьев и обучение модели легко распараллелить, т.е. мы параллельно обучаем каждое из деревьев на сформированной из данных выборке. Обученные деревья объединятся в распределённый Random Forest, результаты которого нужно будет усреднить от каждого дерева для получение итогового результата.



> Распределённые данные

Наиболее популярный и простой подход. Большой датасет разбивается на shard-ы или партиции, каждый shard доставляется на worker. Parameter Server хранит модель и отвечает за её итоговое состояние. Parameter Server доставляет клон модели на каждый из worker-ов к своему shard-у. На каждом worker-е модели обучаются и далее возвращаются в Parameter Server, где анализируются, и составляется единая модель. Эти шаги повторяются до выполнения условия остановки обучения или достижения конечного числа итераций.



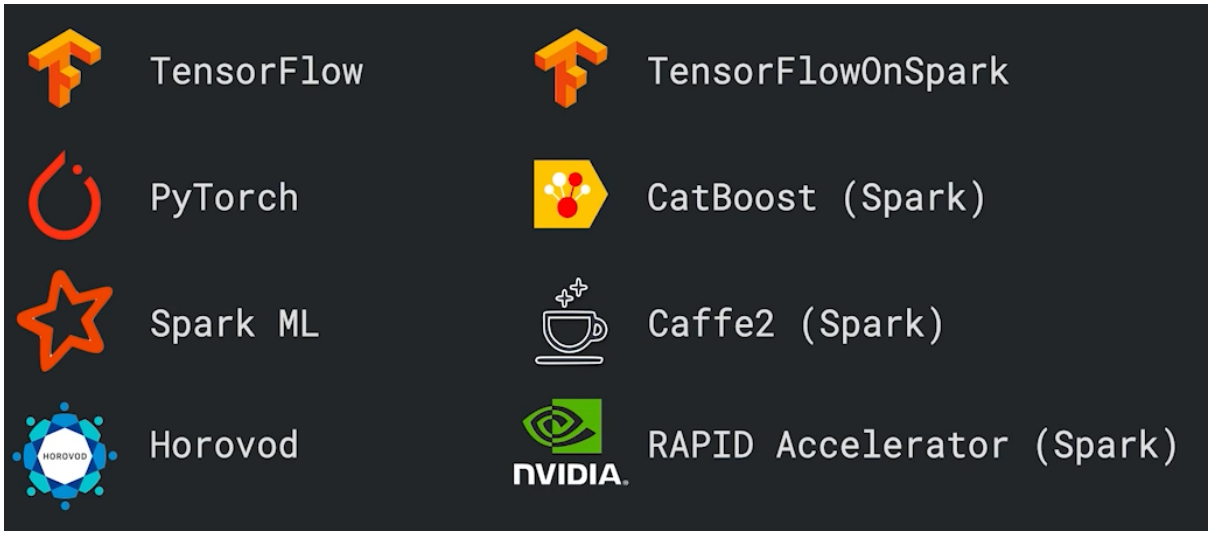
> Фреймворки

TensorFlow, PyTorch – широкопрофильные с распределённым МО.

Spark ML – узкоспециализирован, но предоставляет возможность распределённого МО, но не для всех типов и классов моделей.

Horovod также предоставляет возможность распределённого МО.

Большинство фреймворков, которые изначально не закладывали распределённое МО, имеют разные интеграции для работы и обучения моделей, используя Spark кластер.



> Spark & GPU

В процессе обучения моделей наиболее ценно использовать графическое устройство обработки данных (GPU), чем CPU. Это связано с тем, что модели, а в частности нейросети связаны с алгебраическими, матричными вычислениями, которые выгоднее производить на GPU.

Rapid Accelerator начиная с версии Spark 3.0 фактически встроен в него, что делает возможным использование GPU в Spark кластере. Это позволяет объединять в одном фреймворке и работу с большими данными и обучение моделей на этих больших данных.