BANANAS

Анализ статьи

Статья «BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search», Colin White, Willie Neiswanger, Yash Savani посвящена разработке и анализу метода BANANAS, который сочетает байесовскую оптимизацию (BO) с нейросетевыми предсказателями для автоматического поиска нейронных архитектур (NAS).

Авторы выделяют 5 основных шагов/компонентов;

1. Выбор кодирования архитектуры
   В статье предложен новый метод кодирования – path encoding, в которой каждый признак представляет собой уникальный путь, по которому тензор может пройти от входного слоя до выходного слоя архитектуры. Автору утверждают, что такой способ кодирования масштабируется лучше, чем кодирование матрицы смежности, и позволяет нейронным предикторам достигать более высокой точности;
2. Сравнение нейросетевых предсказателей
   Сети прямого распространения с path encoding показали показали лучшую точность и скорость по сравнению с GCN и VAE;
3. Калибровка неопределенности
   Ансамбли (особенно из 5 моделей) дают более надежные оценки неопределенности, чем баейсовские сети;
4. Выбор функции расширения
   Она необходима для выбора следующей архитектуры для апробации, балансирует между Exploration (попробовать что-то новое) и Exploitation (выбрать то, что, по прогнозу, лучше всего). Тестируются функции TS (случайный выбор на основе распределения), ITS (улучшенная версия TS), UCB (выбор случайной архитектуры, оптимистичный подход), EI (оценивает, насколько в среднем архитектура может улучшить текущий лучший результат), PI (не учитывает величину улучшения, только вероятность, что новая архитектура лучше). ITS даёт наилучшие результаты;
5. Оптимизация функции расширения
   Сравниваются методы случайного поиска, мутация (изменение текущей архитектуры), локальный поиск (исследование «соседей»), эволюция. Авторы приходят к выводу о том, что наилучший оптимизатор в данном случае – мутация.

Иными словами, данный алгоритм можно схематически записать следующим образом:

а) Равномерно случайным образом выбираются t_0 архитектур из пространства поиска и обучаются на датасете;

б) Итерационно проводится обучение ансамбля мета-нейронных сетей на выбранных архитектурах. Используемая функция ошибки — вариация MAPE:

$$\mathcal{L}(y_{pred}, y_{true}) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_{pred}^{(i)} - y_{LB}}{y_{true}^{(i)} - y_{LB}} - 1\right|;$$

в) Далее формируется набор архитектур-кандидатов посредством случайных изменений лучших архитектур после обучения;

г) Для каждой архитектуры-кандидата определяется значение переданной на вход функции сбора независимой выборки Томпсона;

д) Для архитектуры-кандидата с минимальным значением функции расширения оценивается ошибку на валидации.

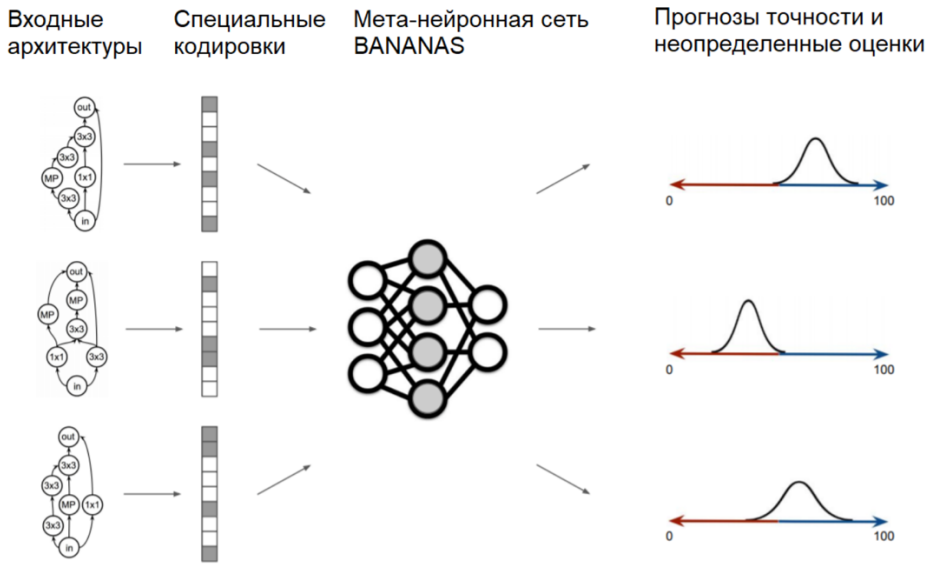

Рисунок 1 - Алгоритм работы BANANAS

Применение для BERT

Метод изначально разработан для поиска оптимальных архитектур нейросетей. Однако его можно адаптировать для гиперпараметрического тюнинга предобученных моделей следующим образом:

1) Задать пространство поиска;
2) Закодировать параметры (аналог path encoding);
3) Использовать предикторы для выбора лучшей конфигурации.

Итого, алгоритм для BERT выглядит следующим образом:

1) Равномерно случайным образом выбираются гиперпараметры из пространства поиска (наша цель – сократить число слоёв в BERT, поэтому задаются минимальное и максимальное кол-во слоёв);
2) Случайное изменение конфигураций (с вероятностью 30% каждый слой может поменять статус активности);
3) В изначальной модели оставляются только выбранные слои;
4) Оценивается полученная конфигурация (метрика учитывает accuracy и кол-во параметров в модели);
5) Метрика конфигурации также предсказывается нейросетевым алгоритмом (аналогично BANANAS);
6) Итерационно проводится обучение ансамбля мета-нейронных сетей на выбранных архитектурах. Используемая функция ошибки — MSE;
7) Из истории выбираются лучшие конфигурации и модифицируются для оценивания новых вариантов, история обновляется;
8) После всех итераций – вывод лучшей найденной конфигурации.

Результаты

Алгоритм позволил найти более простую архитектуру, состоящую из 4 слоев, которая позволила выбить accuracy на тестовой выборке в 90,5%. При этом кол-во используемых

параметров сократилось до 52 млн. против 109.5. Это позволило значительно ускорить обучение в том числеи по времени.