Ren, M. Sun, J. Zheng, S. Marshall // International Journal of Remote Sensing. – 2014. – Vol. 35(20). – P. 7316–7337.

3. Mankar P. R. Image compression based on 3D-DCT / P. R. Mankar, S. S. Rane, A. E. Patil // International Journal of Research in Science & Engineering. – 2017. – Vol. 3(2). – P. 16-22.

4. Ахмед Н. Ортогональные преобразования при обработке цифровых сигналов / Н. Ахмед, К. Р. Рао. – М.: Связь, 1980. – 248 с.

5. Чернов В. М. Быстрые алгоритмы дискретных косинусных преобразований коротких длин с минимальной вычислительной сложностью / В. М. Чернов, М. А. Чичёва // Известия Самарского научного центра РАН. – 1999. – Т. 2. – С. 241-248.

6. Сергеев В. В., Юзькив Р. Р. Параметрическая модель автокорреляционной функции космических гиперспектральных изображений. Компьютерная оптика, 2016, 40(3), с. 416-421.

7. Чочиа П. А. Двухмасштабная модель изображения / П. А. Чочиа // Кодирование и обработка изображений. – М.: Наука, 1988. – С. 69–87.

8. MacQueen J. Some methods for classification and analysis of multivariate observations / J. MacQueen // Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. – 1967. – P. 281-297.

9. Arthur D. k-means++: the advantages of careful seeding / D. Arthur, S. Vassilvitskii // Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms. – 2007. – P. 1027-1035.

10. ISO/IEC 10918-1. Information technology. Digital compression and coding of continuous-tone still images. Requirements and guidelines. – ISO/IEC JTC 1/SC 29 Coding of audio, picture, multimedia and hypermedia information, 1994. – 182 p.

11. NASA. AVIRIS Data Portal. URL: https://aviris.jpl.nasa.gov/alt_locator/ (дата обращения: 29.04.2019).

12. Yuzkiv R. R. Transform-based coding method for remote sensing hyperspectral data compression / R. R. Yuzkiv, V. V. Sergeev // Procedia Engineering. – 2017. – Vol. 201. – P. 249-257.

А.А. Ямалтдинова, В.В. Мокшин

ГЛУБОКОЕ ОБУЧЕНИЕ С ИСПОЛЬЗОВАНИЕМ БАЙЕСОВСКОЙ ОПТИМИЗАЦИИ В MATLAB

(Альметьевский филиал КНИТУ-КАИ им А.Н.Туполева)

Глубокое обучение в наше время стало направлением, которое стоит превыше всего исследований в машинном обучении. Начавшись с архитектурных прорывов, которые давали возможность оперативно обучать глубокие нейросети, оно стало направляться и на другие стороны, отдавая набор результативных систем там, где для исполнения проблем необходим приближение какой-то сложной функции.

Машинное обучение - чрезвычайно молодой раздел науки, по некоторому мнению сформировавшийся окончательно на стыке множества различных дисциплин в начале 70-х годов прошлого века, когда наши соотечественники Вапник и Червонекис математически обосновали принцип минимизации эмпирического риска.

Одним из принципиально важных следствий этого результата стала возможность формулировать задачи машинного обучения как задачи численной оптимизации. Кратко, эта методология заключается в том, что исследователь должен собрать выборку примеров, для каждого из которых известен правильный ответ, после чего эта выборка делится на две части, на одной из которых алгоритм обучается, а на другой тестируется.

Байесовская оптимизация - это алгоритм, хорошо подходящий для оптимизации гиперпараметров моделей классификации и регрессии. Вы можете использовать байесовскую оптимизацию для оптимизации функций, которые недифференцируемы, прерывисты и трудоемки для оценки. Алгоритм внутренне поддерживает гауссову модель процесса целевой функции и использует оценки целевой функции для обучения этой модели.

В этой статье мы применяем байесовскую оптимизацию к глубокому обучению и находим оптимальные сетевые гиперпараметры и варианты обучения для сверточных нейронных сетей.

Реализация глубокого обучения с использованием байесовской оптимизации в MATLAB:

1. Подготовка данных.

Для начала, загрузим набор данных CIFAR-10. Этот набор данных содержит 60000 изображений, и каждое изображение имеет размер 32 на 32 и три цветовых канала (RGB). Размер всего набора данных составляет 175 МБ. В зависимости от вашего Интернет-соединения, процесс загрузки может занять некоторое время.

Загрузим набор данных CIFAR-10 в виде обучающих изображений и этикеток, а также тестовых изображений и этикеток. Чтобы включить проверку сети, используем 5000 тестовых изображений для проверки.

2. Выбор переменных для оптимизации.

Выберем, какие переменные оптимизировать, используя байесовскую оптимизацию, и укажем диапазоны для поиска. Кроме того, укажем, являются ли переменные целыми числами и следует ли искать интервал в логарифмическом пространстве. Оптимизируем следующие переменные:

-Глубина раздела сети. Этот параметр контролирует глубину сети. В результате количество параметров и требуемый объем вычислений для каждой итерации примерно одинаковы для разных глубин сечения.

-Начальная скорость обучения. Наилучшая скорость обучения может зависеть от ваших данных, а также от сети, которую вы обучаете.

-Стохастический градиент нисходящего импульса. Импульс добавляет инерцию к обновлениям параметров, поскольку текущее обновление содержит вклад, пропорциональный обновлению в предыдущей итерации.

-Сила регуляризации L2. Используем регуляризацию, чтобы предотвратить переоснащение. Ищем пространство силы регуляризации, чтобы найти хорошее значение.

3. Выполнение байесовской оптимизации.

Создаем целевую функцию для байесовского оптимизатора, используя данные обучения и проверки в качестве входных данных. Целевая функция обучает сверточную нейронную сеть и возвращает ошибку классификации в наборе проверки. Поскольку bayesopt использует частоту появления ошибок в наборе проверки для выбора наилучшей модели, возможно, что окончательная сеть переопределится в наборе проверки. Окончательно выбранная модель затем тестируется на независимом тестовом наборе для оценки ошибки обобщения.

Выполняем байесовскую оптимизацию, минимизировав ошибку классификации на наборе валидации. Укажем общее время оптимизации в секундах. Для параллельного обучения сетей на нескольких графических процессорах установим для параметра «UseParallel» значение true. После того, как каждая сеть заканчивает обучение, bayesopt печатает результаты в командном окне. Затем функция bayesopt возвращает имена файлов в BayesObject.UserDataTrace. Целевая функция сохраняет обученные сети на диск и возвращает имена файлов в bayesopt.

Значение целевой функции в стиматике = 0,18813.

Расчетное время оценки функции = 2166.2402.

4.Оценивание итоговой сети.

Загрузим лучшую сеть, найденную в оптимизации, и ее точность проверки.

Прогнозирование меток тестового набора и вычисление ошибки теста. Рассматриваем классификацию каждого изображения в тестовом наборе как независимые события с определенной вероятностью успеха, что означает, что количество неправильно классифицированных изображений следует биномиальному распределению. Используем это, чтобы вычислить стандартную ошибку (testErrorSE) и приблизительный 95% доверительный интервал (testError95CI) от частоты ошибок обобщения. Этот метод часто называют методом Вальда. bayesopt определяет лучшую сеть, используя набор проверки, не подвергая сеть испытательному набору. Тогда, возможно, что ошибка теста выше, чем ошибка проверки.

Покажем на рисунке 1 некоторые тестовые изображения вместе с их предсказанными классами и вероятностями этих классов.

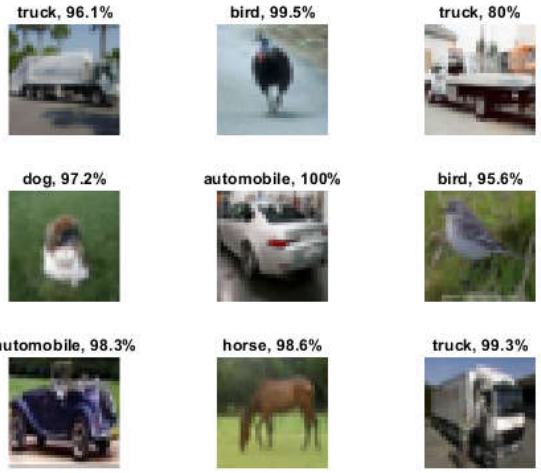

Рис.1. Тестовые изображения вместе с их предсказанными классами и вероятностями этих классов

5.Целевая функция для оптимизации.

Определим целевую функцию для оптимизации. Эта функция выполняет следующие шаги:

-Принимает значения переменных оптимизации в качестве входных данных.

-Определяет архитектуру сети и параметры обучения.

-Обучает и проверяет сеть.

-Сохраняет обученную сеть, ошибку проверки и параметры обучения на диск.

-Возвращает ошибку проверки и имя файла сохраненной сети.

Каждый раз, когда вы уменьшаете выборку пространственных измерений в два раза, используя максимальное количество пулов, увеличиваем количество фильтров в два раза. Это гарантирует, что объем вычислений, требуемый в каждом сверточном слое, примерно одинаков.

Выберем количество фильтров, пропорциональное 1 / sqrt (SectionDepth), чтобы сети разной глубины имели примерно одинаковое количество параметров и требовали примерно одинакового объема вычислений на одну итерацию. Чтобы увеличить количество параметров сети и общую гибкость сети, увеличим numF. Чтобы обучать даже более глубокие сети, изменим диапазон переменной SectionDepth.

Используем convBlock (filterSize, numFilters, numConvLayers) для создания блока сверточных слоев numConvLayers. Функция convBlock определена в конце этого примера.

Укажем параметры для сетевого обучения. Оптимизируем начальную скорость обучения, импульс SGD и уровень регуляризации L2.

Укажем данные проверки и выберем значение «ValidationFrequency», чтобы trainNetwork проверял сеть один раз за эпоху. Тренируем для фиксированного числа эпох и понизьте скорость обучения в 10 раз в течение последних

эпох. Это уменьшает шум обновлений параметров и позволяет сетевым параметрам располагаться ближе к минимуму функции потерь.

Используем увеличение данных. Увеличение данных помогает предотвратить переоснащение сети и запоминание точных деталей тренировочных образов. Обучим сеть и наметем прогресс обучения во время обучения.

В результате написания статьи была достигнута следующая цель, а именно: реализация глубокого обучения для распознавания объекта на изображении с использованием байесовской оптимизации в MATLAB.

Достижение цели выполнено путем решения задач, таких как:

- спроектирован и разработан алгоритм и программное обеспечение для распознавания и классификации объекта на изображении;
- программное обеспечение протестировано на тестовых наборах данных;
- загрузка и подготовка набора данных CIFAR-10 для сетевого обучения.
- указание переменных для оптимизации с помощью байесовской оптимизации.
- определение целевой функции, которая принимает значения переменных оптимизации в качестве входных данных, задает архитектуру сети и параметры обучения, обучает и проверяет сеть и сохраняет обученную сеть на диск.
- выполнение байесовской оптимизации, минимизировав ошибку классификации на наборе валидации.

Была подтверждена необходимость разработки конфигурации вследствие того, что похожих программ очень мало. Разработка сопровождается скриншотами созданной конфигурации.

Литература

1. Борисов Е.С. О методах обучения многослойных нейронных сетей прямого распространения. Градиентные методы первого порядка. [Электронный ресурс] / URL: http://mechanoid.kiev.ua/neural-net-backprop2.html.(Дата обращения 28.04.2019).

2. Выделение контуров на бинарных изображениях. // TechnicalVision.ru. [Электронный ресурс] / URL: http://wiki.technicalvision.ru. (Дата обращения 28.04.2019).

3. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. — М.: Техносфера, 2005. — 1072с.

4. Зенин А. В. Анализ методов распознавания образов. [Электронный ресурс] // Молодой ученый. — 2017. — №16. — С. 125-130. URL: https://moluch.ru/archive/150/42393/. (Дата обращения: 28.04.2019).

5. Мокшин В.В., Сайфудинов И.Р., Кирпичников А.П. Рекурсивный алгоритм построения регрессионных моделей сложных вероятностных объектов / Вестник Технологического университета. 2017. Т. 20. № 9. С. 112-116.

6. Мокшин В.В., Якимов И.М. Метод формирования модели анализа сложной системы / Информационные технологии. 2011. № 5. С. 46-51.

7. Мокшин В.В., Кирпичников А.П., Шарнин Л.М. Система мониторинга количества материалов для строительства дорожного полотна / Вестник Технологического университета. 2017. Т. 20. № 17. С. 99-103.

8. Мокшин В.В. Параллельный генетический алгоритм отбора значимых факторов, влияющих на эволюцию сложной системы / Вестник Казанского государственного технического университета им. А.Н. Туполева. 2009. № 3. С. 89-93.

9. Мокшин В.В., Якимов И.М., Юльметьев Р.М., Мокшин А.В. Рекурсивно-регрессионная самоорганизация моделей анализа и контроля сложных систем // Нелинейный мир. М: 2009. №1. С. 48-63.

10. Якимов И.М., Кирпичников А.П., Мокшин В.В., Яхина З.Т. Сравнение систем структурного и имитационного моделирования по модели M/M/5 // Вестник Технологического университета. 2017. Т. 20. № 16. С. 113-119.