МЕТОДЫ ОПТИМИЗАЦИИ НА ОСНОВЕ ГРАДИЕНТНОГО СПУСКА С МОМЕНТОМ В ЗАДАЧАХ ОБУЧАНИЯ НЕЙРОННЫХ СЕТЕЙ

Гаруля Д. В., Навроцкий А. А., Белошедов Е. С.
Кафедра информационных технологий автоматизированных систем, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: dimagarul58@gmail.com

В данной работе исследуется возможность повышения эффективности обучения нейронных сетей за счёт модификации алгоритма градиентного спуска моментом и его вариацией под названием момент Нестерова, которые позволяют проводить физицескую аналогию процессу оптимизации, как «скатывание» частицы в рельефе. Рассмотренный алгоритмы позволяют не только ускорить скорость обучения нейронных сетей но и уменьшить эффект их переобучения.

Введение

В настоящее время алгоритмы машинного обучения (АМО) достигли значительно уровня. АМО способны решать широкийкруг задач – прогнозиро-вание временных рядов, распознавание речи, компьютерное зрение и т. д. Характерной чертой АМО является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Обучение таких алгоритмов сводится к минимизации функции потерь. Основным методом оптимизации функций, применяемых в задачах машинного обучения, является метод градиентного спуска (ГС) [1]. Главным недостатком этого метода при использовании в обучении нейронных сетей (НС) является резкое падение скорости обучения на участках незначительного изменения функции ошибок («плато»), при этом теоретически для НС ГС всегда находит минимум функции. Таким образом, одной из важнейших задач для развития АМО является поиск наиболее эффективного метода оптимизации функций.

I. Принцип машинного обучения

Два ключевых компонента в контексте задач классификации документов:

- Параметризованная функция, сопоставляющая необработанный текст с меткой класса;
- Функция потерь, которая измеряет качество определенного набора параметров на основе того, насколько хорошо совпадают полученные оценки с заданными метками обучающих данных;

Дана непрерывная функция $J(\theta)$, определяется итеративный алгоритм оптимизации. Один из самых простых способов оптимизации - это градиентный спуск, который при заданных начальных параметрах, локально изменяет параметр θ на итерации t в сторону уменьшения значения J. $J(\theta_t + v_t) = J(\theta_t) + v_t J^{`}(\theta_t) + O(v_t^2)$. Для маленькой скорости обучения $\alpha_t > 0$, заданная $v_t = -\alpha_t J^{`}(\theta_t)$ уменьшает значение J. ПОэтому итерация градиентного спуска сводится к $\theta_{t+1} = \theta_t + v_t$. Для выпуклой функции J градиентный спуск сходится к оптимальному значению. Хотя градиентный спуск весьма популярен, альтернативные метод, такие как момент или ускореннвй градиент Нестерова (Nesterov‘s Accelerated Gradient (NAG)) может привести к значительно более быстрой сходимости к оптимальному значению [2].

II. Градиентный спуск с моментом

Градиентный спуск с моментом (Momentum Gradient Descent), как правило, обеспечивает более быструю сходимость в глубоких нейронных сетях. Для этого метода можно провести физическую аналогию. Функцию потерь можно представить как некоторый рельеф, при этом значение функции потерь соотвествует значению некоторой высоты, то есть потенциальной энергии. Тогда процесс оптимизации мы можем представить как процесс «скатывания» частицы в этом ландшафте. Поскольку сила, действующая на частицу, связана с градиентом потенциальной энергии (то есть $F = -\nabla U F = -\nabla U$), сила, воспринимаемая частицей, является в точности (отрицательным) градиентом функции потерь. Более того, $F = maF = ma$, поэтому (отрицательный) градиент в этом представлении пропорционален ускорению частицы. Обратите внимание, что это отличаетс от обновления стохастического градиентного спуска, показанного выше, где градиент напрямую определяется положением. Вместо этого физический вид предлагает обновление, в котором градиент напрямую виляет только на скорость $v = mu * v - learning\_rate * dx$, которая в свою очередь, влияет на положение $x = x + v$. Сравниетльный график представлен на рисунке 1. Водится переменная v, которая инициализируется нулем и дополнительный гиперпараметр mu. Гиперпараметр mu, при отптимизации, называетс моментом, его физический смысл больше соответствует коэффициенту трения. Типовыми значениями гиперпарамет-

ра mu являются 0.5, 0.9, 0.95, 0.99. Фактически, эта переменная уменьшает скорость и кинетическую жнергию системы, иначе частица никогда не остановится перед возвышением ландшафта. С обновлением гиперпараметра mu вектор параметров будет наращивать скорость в любом направлении, имеющем постоянный градиент.

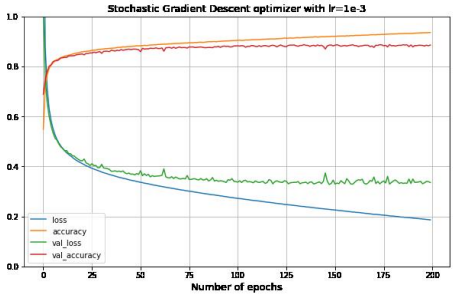

Рис. 1 – Потери и точность модели с набором даннх Fashion MNIST

III. Момент Нестерова

Nesterov Momentum - это немного другая версия набирающего популярность в последнее время метода Momentum Gradient Descent. В теории он обладает более высокой скоростью сходимости для выпуклых функций, а на практике он также работает немного лучше, чем стандартный момент. Основная идея момента Нестерова заключается в том, что когда текущий вектор параметров находится в некоторой позиции x, то, глядя на обновление момента по формуле представленной выше ($v = mu * v - learning\_rate * dx$ (интегрированная скорость)). Только один член момента (т.е. игнорирование второго члена с градиентом) изменяет параметр vector с помощью $mu * v$. Следовательно, если мы собираемся вычислить градиент, мы можем рассматривать приблизительное «следующее» положение $x + mu * v$ как «опережающий взгляд» - это точка в непосредственной близости от того места, где мы скоро окажемся. Следовательно, имеет смысл вычислять градиент в точке $x + mu * v$ вместо «старой» позиции x [3]. Вместо того, чтобы оценивать градиент в текущей позиции (красный кружок), мы знаем, что наш импульс вот-вот приведет нас к кончику зеленой стрелки. Поэтому с помощью импульса Нестерова мы вместо этого оцениваем градиент в этой «прогнозируемой» позиции (см. рис. 2).

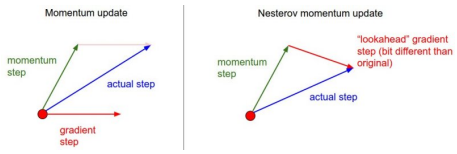

Рис. 2 – Принципы методов градиентного спуска с моментом и момента Нестерова

То есть в несколько неудобных обозначениях мы хотели бы сделать следующее:

- $x\_ahead = x + mu * v$ оценить dx_ahead (градиент на x_ahead вместо x);
- $v = mu * v - learning\_rate * dx\_ahead$;
- $x = x + v$;

Однако на практике люди предпочитают выражать момент Нестерова так, чтобы он выглядело как можно более похожим на исходный метод стохастического градиентного спуска или на метод градиентного спуска с моментом. Этого можно достичь, используя переменную $transform$ $x\_ahead = x + mu * v$, а затем выражая момент Нестерова в терминах x_ahead вместо x. То есть вектор параметров, который мы фактически сохраняем, всегда является опережающей версией. Уравнения в терминах x_ahead (но переименовывая его обратно в x) становятся:

- $v\_prev = v$
- $v = mu * v - learning\_rate * dx$ обновление скорости остается прежним
- $x = x - mu * v\_prev + (1 + mu) * v$ обновление позиции изменяет форму

Заключение

Метод градиентного спуска, применяемый в обучении нейронных сетей, гарантирует сходимость к оптимальному значению при бесконечном количестве итераций. Однако на практике скорость обучения при использовании градиентного спуска оказывается невелика из-за эффекта «плато». Модификация градиентного спуска с использованием момента позволяет уменьшить этот эффект позволяя преодолевать участки с малым градиентом за меньшее время за счет накопленной «инерции». Вычисление градиента с некоторым опережением позволяет увеличить эффективность метода градиентного спуска с моментом.

Список литературы

1. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies withgradient descent is difficult.IEEE Transactions on Neural Networks, 5(2):157–166, 1994

2. Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. “On the importance of initialization and momentum in deep learning.” Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML13,p. 1139, 2013

3. Y. Nesterov. A method of solving a convex programming problem with convergence rate O(1/k2). Soviet Mathematics Doklady, volume 27, p. 372–376, 1983