ВИКИПЕДИЯ

Метод обратного распространения ошибки

Материал из Википедии — свободной энциклопедии

Метод обратного распространения ошибки (англ. *backpropagation*) — метод вычисления градиента, который используется при обновлении весов многослойного перцептрона. Впервые метод был описан в 1974 г. А. И. Галушкиным[1], а также независимо и одновременно Полом Дж. Вербосом[2]. Далее существенно развит в 1986 г. Дэвидом И. Румельхартом, Дж. Е. Хинтоном и Рональдом Дж. Вильямсом[3] и независимо и одновременно С.И. Барцевым и В.А. Охониным (Красноярская группа)[4]. Это итеративный градиентный алгоритм, который используется с целью минимизации ошибки работы многослойного перцептрона и получения желаемого выхода.

Основная идея этого метода состоит в распространении сигналов ошибки от выходов сети к её входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Барцев и Охонин предложили сразу общий метод («принцип двойственности»), приложимый к более широкому классу систем, включая системы с запаздыванием, распределённые системы, и т. п.[5]

Для возможности применения метода обратного распространения ошибки передаточная функция нейронов должна быть дифференцируема. Метод является модификацией классического метода градиентного спуска.

Содержание

Сигмоидальные функции активации

Функция оценки работы сети

Описание алгоритма

Алгоритм

Пример реализации

Режимы реализации

Математическая интерпретация обучения нейронной сети

Недостатки алгоритма

Паралич сети

Локальные минимумы

Размер шага

Литература

Ссылки

Примечания

Сигмоидальные функции активации

Наиболее часто в качестве функций активации используются следующие виды сигмоид:

Функция Ферми (экспоненциальная сигмоида):

$$f(s) = \frac{1}{1 + e^{-2\alpha s}}$$

Рациональная сигмоида (при $\alpha = 0$ вырождается в т. н. пороговую функцию активации):

$$f(s) = \frac{s}{|s| + \alpha}$$

Гиперболический тангенс:

$$f(s) = \operatorname{th} \frac{s}{\alpha} = \frac{e^{\frac{s}{\alpha}} - e^{-\frac{s}{\alpha}}}{e^{\frac{s}{\alpha}} + e^{-\frac{s}{\alpha}}},$$

где s — выход сумматора нейрона, α — произвольная константа.

Менее всего, сравнительно с другими сигмоидами, процессорного времени требует расчёт рациональной сигмоиды. Для вычисления гиперболического тангенса требуется больше всего тактов работы процессора. Если же сравнивать с пороговыми функциями активации, то сигмоиды рассчитываются очень медленно. Если после суммирования в пороговой функции сразу можно начинать сравнение с определённой величиной (порогом), то в случае сигмоидальной функции активации нужно рассчитать сигмоид (затратить время в лучшем случае на три операции: взятие модуля, сложение и деление), и только потом сравнивать с пороговой величиной (например, нулём). Если считать, что все простейшие операции рассчитываются процессором за примерно одинаковое время, то работа сигмоидальной функции активации после произведённого суммирования (которое займёт одинаковое время) будет медленнее пороговой функции активации в 4 раза.

Функция оценки работы сети

В тех случаях, когда удаётся оценить работу сети, обучение нейронных сетей можно представить как задачу оптимизации. Оценить — означает указать количественно, хорошо или плохо сеть решает поставленные ей задачи. Для этого строится функция оценки. Она, как правило, явно зависит от выходных сигналов сети и неявно (через функционирование) — от всех её параметров. Простейший и самый распространённый пример оценки — сумма квадратов расстояний от выходных сигналов сети до их требуемых значений:

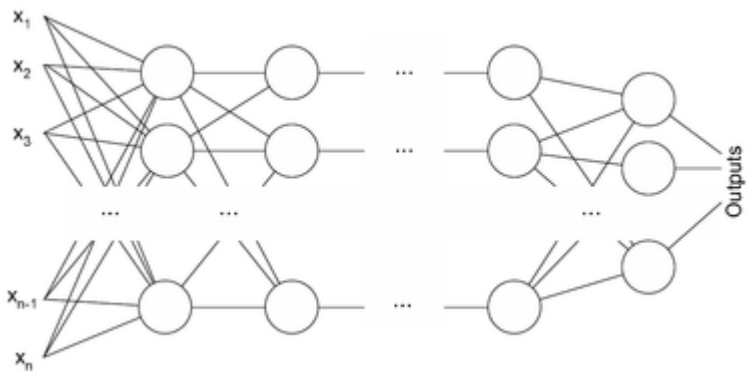
$$H = \frac{1}{2} \sum_{\tau \in v_{\text{out}}} (Z(\tau) - Z^*(\tau))^2,$$

где $Z^*(\tau)$ — требуемое значение выходного сигнала.

Метод наименьших квадратов далеко не всегда является лучшим выбором оценки. Тщательное конструирование функции оценки позволяет на порядок повысить эффективность обучения сети, а также получать дополнительную информацию — «уровень уверенности» сети в даваемом

ответе[6].

Описание алгоритма



Архитектура многослойного перцептрона

Алгоритм обратного распространения ошибки применяется для многослойного перцептрона. У сети есть множество входов $x_1, \ldots, x_n$, множество выходов Outputs и множество внутренних узлов. Перенумеруем все узлы (включая входы и выходы) числами от 1 до *N* (сквозная нумерация, вне зависимости от топологии слоёв). Обозначим через $w_{i,j}$ вес, стоящий на ребре, соединяющем *i*-й и *j*-й узлы, а через o_i — выход *i*-го узла. Если нам известен обучающий пример (правильные ответы сети t_k, $k \in \mathrm{Outputs}$), то функция ошибки, полученная по методу наименьших квадратов, выглядит так:

$$E(\{w_{i,j}\}) = \frac{1}{k} \sum_{k \in \mathrm{Outputs}} (t_k - o_k)^2$$

Как модифицировать веса? Мы будем реализовывать стохастический градиентный спуск, то есть будем подправлять веса после каждого обучающего примера и, таким образом, «двигаться» в многомерном пространстве весов. Чтобы «добраться» до минимума ошибки, нам нужно «двигаться» в сторону, противоположную градиенту, то есть, на основании каждой группы правильных ответов, добавлять к каждому весу $w_{i,j}$

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}},$$

где $0 < \eta < 1$ — множитель, задающий скорость «движения».

Производная считается следующим образом. Пусть сначала $j \in \mathrm{Outputs}$, то есть интересующий нас вес входит в нейрон последнего уровня. Сначала отметим, что $w_{i,j}$ влияет на выход сети только как часть суммы $S_j = \sum_i w_{i,j} x_i$, где сумма берётся по входам *j*-го узла.

Поэтому

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j}$$

Аналогично, S_j влияет на общую ошибку только в рамках выхода *j*-го узла o_j (напоминаем, что это выход всей сети). Поэтому

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k \in \mathrm{Outputs}}(t_k - o_k)^2\right)\left(\frac{\partial \mathrm{f}(S)}{\partial S}|_{S=S_j}\right) =$$
$$= \left(\frac{1}{2}\frac{\partial}{\partial o_j}(t_j - o_j)^2\right)(o_j(1-o_j))2\alpha = -2\alpha o_j(1-o_j)(t_j - o_j).$$

где $f(S)$ — соответствующая сигмоида, в данном случае — экспоненциальная

$$\frac{\partial(\frac{1}{1+e^{-2\alpha S}})}{\partial S} = \frac{-1}{(1+e^{-2\alpha S})^2} \times \frac{\partial(1+e^{-2\alpha S})}{\partial S} = \frac{-1}{(1+e^{-2\alpha S})^2} \times (-2\alpha e^{-2\alpha S}) =$$
$$= \frac{2\alpha e^{-2\alpha S}}{(1+e^{-2\alpha S})^2} = \left(\frac{1+e^{-2\alpha S}}{(1+e^{-2\alpha S})^2} - \frac{1}{(1+e^{-2\alpha S})^2}\right) \times 2\alpha = 2\alpha(\mathrm{f}(S) - \mathrm{f}^2(S))$$

Если же *j*-й узел — не на последнем уровне, то у него есть выходы; обозначим их через Children(*j*). В этом случае

$$\frac{\partial E}{\partial S_j} = \sum_{k \in \mathrm{Children}(j)} \frac{\partial E}{\partial S_k}\frac{\partial S_k}{\partial S_j},$$

и

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j}\frac{\partial o_j}{\partial S_j} = w_{j,k}\frac{\partial o_j}{\partial S_j} = 2\alpha w_{j,k} o_j(1-o_j).$$

Но $\frac{\partial E}{\partial S_k}$ — это в точности аналогичная поправка, но вычисленная для узла следующего уровня. Будем обозначать её через δ_k — от Δ_k она отличается отсутствием множителя $(-\eta x_{i,j})$. Поскольку мы научились вычислять поправку для узлов последнего уровня и выражать поправку для узла более низкого уровня через поправки более высокого, можно уже писать алгоритм. Именно из-за этой особенности вычисления поправок алгоритм называется **алгоритмом обратного распространения ошибки** (backpropagation). Краткое резюме проделанной работы:

- для узла последнего уровня

$$\delta_j = -2\alpha o_j(1-o_j)(t_j - o_j)$$

- для внутреннего узла сети

$$\delta_j = 2\alpha o_j(1-o_j)\sum_{k \in \mathrm{Children}(j)} \delta_k w_{j,k}$$

- для всех узлов

$$\Delta w_{i,j} = -\eta \delta_j o_i,$$

где o_i это тот же x_i в формуле для $\frac{\partial E}{\partial w_{i,j}}$.

Получающийся алгоритм представлен ниже. На вход алгоритму, кроме указанных параметров, нужно также подавать в каком-нибудь формате структуру сети. На практике очень хорошие результаты показывают сети достаточно простой структуры, состоящие из двух уровней нейронов — скрытого уровня (hidden units) и нейронов-выходов (output units); каждый вход сети соединён со всеми скрытыми нейронами, а результат работы каждого скрытого нейрона подаётся на вход каждому из нейронов-выходов. В таком случае достаточно подавать на вход количество нейронов скрытого уровня.

Алгоритм

Алгоритм: ***BackPropagation*** $(\eta, \alpha, \{x_i^d, t^d\}_{i=1,d=1}^{n,m}, \text{steps})$

1. Инициализировать $\{w_{ij}\}_{i,j}$ маленькими случайными значениями, $\{\Delta w_{ij}\}_{i,j} = 0$
2. Повторить NUMBER_OF_STEPS раз:
   .Для всех d от 1 до m:
   1. Подать $\{x_i^d\}$ на вход сети и подсчитать выходы o_i каждого узла.
   2. Для всех $k \in Outputs$
      $\delta_k = -o_k(1 - o_k)(t_k - o_k)$.
   3. Для каждого уровня l, начиная с предпоследнего:
      Для каждого узла j уровня l вычислить
      $\delta_j = o_j(1 - o_j) \sum_{k \in Children(j)} \delta_k w_{j,k}$.
   4. Для каждого ребра сети {i, j}
      $\Delta w_{i,j}(n) = \alpha \Delta w_{i,j}(n-1) + (1 - \alpha)\eta \delta_j o_i$.
      $w_{i,j}(n) = w_{i,j}(n-1) + \Delta w_{i,j}(n)$.
3. Выдать значения w_{ij}.

где α — коэффициент инерциальности для сглаживания резких скачков при перемещении по поверхности целевой функции

Пример реализации

- Статья «A Step by Step Backpropagation Example» от Matt Mazur (https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/)
- Код по статье (реализация нейросети) (https://drive.google.com/file/d/16hiZefHVkIb8Cvhf3BbBXsOOV9hob7eo/view?usp=sharing)
- Обсуждение примера реализации (https://www.facebook.com/konstantin.berlinskii/posts/1463959563761486)

Режимы реализации

Существует два режима реализации метода обратного распространения ошибки :

1. *Стохастического* (stochastic) градиентного спуска
2. *Пакетного* (batch) градиентного спуска

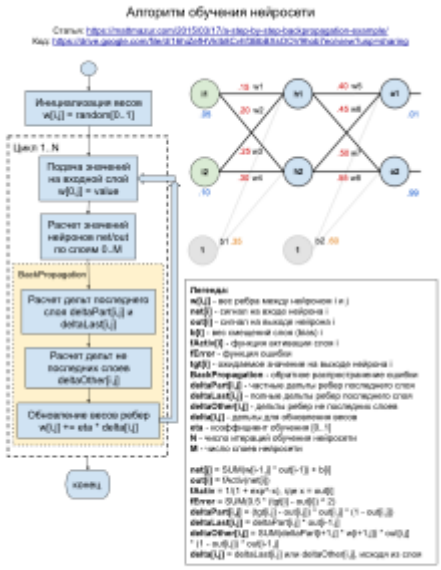
Для пакетного градиентного спуска функция потерь вычисляется для всех образцов вместе взятых после окончания эпохи, и потом вводятся поправки весовых коэффициентов нейрона в соответствии с методом обратного распространения ошибки.

Стохастический метод немедленно после вычисления выхода сети на одном образце вводит поправки в весовые коэффициенты.

Пакетный метод более быстрый и стабильный, но он имеет тенденцию останавливаться и застревать в локальных минимумах. Поэтому для выхода из локальных минимумов нужно использовать особые приёмы, например, алгоритм имитации отжига.

Стохастический метод медленнее, но от того, что он не осуществляет точного градиентного спуска, а вносит «шумы», используя недовычисленный градиент, он способен выходить из локальных минимумов и может привести к лучшему результату.

В виде компромисса рекомендуют также применять мини-пакеты, когда поправка искомых весов осуществляется после обработки нескольких образцов (мини-пакета), то есть, реже чем при стохастическом спуске, но чаще чем при пакетном.



Алгоритм обучения нейросети

Алгоритм обучения нейросети, ч.2, пример

Математическая интерпретация обучения нейронной сети

На каждой итерации алгоритма обратного распространения весовые коэффициенты нейронной сети модифицируются так, чтобы улучшить решение одного примера. Таким образом, в процессе обучения циклически решаются однокритериальные задачи оптимизации.

Обучение нейронной сети характеризуется четырьмя специфическими ограничениями, выделяющими обучение нейросетей из общих задач оптимизации: астрономическое число параметров, необходимость высокого параллелизма при обучении, многокритериальность решаемых задач, необходимость найти достаточно широкую область, в которой значения всех минимизируемых функций близки к минимальным. В остальном проблему обучения можно, как правило, сформулировать как задачу минимизации оценки. Осторожность предыдущей фразы

(«как правило») связана с тем, что на самом деле нам неизвестны и никогда не будут известны все возможные задачи для нейронных сетей, и, быть может, где-то в неизвестности есть задачи, которые несводимы к минимизации оценки. Минимизация оценки — сложная проблема: параметров астрономически много (для стандартных примеров, реализуемых на PC — от 100 до 1 000 000), адаптивный рельеф (график оценки как функции от подстраиваемых параметров) сложен, может содержать много локальных минимумов.

Недостатки алгоритма

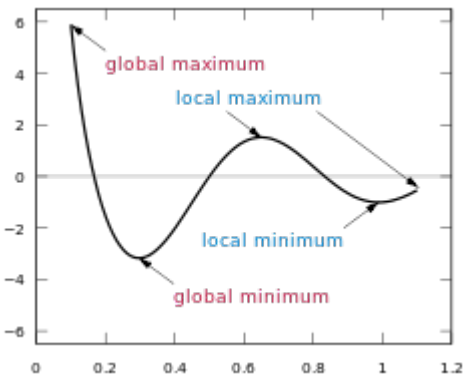
Несмотря на многочисленные успешные применения обратного распространения, оно не является универсальным решением. Больше всего неприятностей приносит неопределённо долгий процесс обучения. В сложных задачах для обучения сети могут потребоваться дни или даже недели, она может и вообще не обучиться. Причиной может быть одна из описанных ниже.

Паралич сети

В процессе обучения сети значения весов могут в результате коррекции стать очень большими величинами. Это может привести к тому, что все или большинство нейронов будут функционировать при очень больших значениях OUT, в области, где производная сжимающей функции очень мала. Так как посылаемая обратно в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может практически замереть. В теоретическом отношении эта проблема плохо изучена. Обычно этого избегают уменьшением размера шага η, но это увеличивает время обучения. Различные эвристики использовались для предохранения от паралича или для восстановления после него, но пока что они могут рассматриваться лишь как экспериментальные.

Локальные минимумы

Обратное распространение использует разновидность градиентного спуска, то есть осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму. Поверхность ошибки сложной сети сильно изрезана и состоит из холмов, долин, складок и оврагов в пространстве высокой размерности. Сеть может попасть в локальный минимум (неглубокую долину), когда рядом имеется гораздо более глубокий минимум. В точке локального минимума все направления ведут вверх, и сеть неспособна из него выбраться. Основную трудность при обучении нейронных сетей составляют как раз методы выхода из локальных минимумов: каждый раз выходя из локального минимума снова ищется следующий локальный минимум тем же методом обратного распространения ошибки до тех пор, пока найти из него выход уже не удаётся.



Метод градиентного спуска может застрять в локальном минимуме, так и не попав в глобальный минимум

Проблемы отсутствия выпуклости в функции ошибок и как следствие трудности с локальными минимумами и плоскими участками считались недостатком метода, однако Ян Лекун в обзорной статье 2015 года утверждает, что с практической точки зрения эти явления не так опасны.[7]

Размер шага

Если размер шага фиксирован и очень мал, то сходимость слишком медленная, если же он фиксирован и слишком велик, то может возникнуть паралич или постоянная неустойчивость. Эффективно увеличивать шаг до тех пор, пока не прекратится улучшение оценки в данном направлении антиградиента и уменьшать, если такого улучшения не происходит. П. Д. Вассерман[8] описал адаптивный алгоритм выбора шага, автоматически корректирующий размер шага в процессе обучения. В книге А. Н. Горбаня[9] предложена разветвлённая технология оптимизации обучения.

Согласно [9], метод обратного распространения ошибки является способом быстрого вычисления градиента, который далее используется в различных алгоритмах гладкой оптимизации, а наиболее перспективным является использование квазиньютоновских методов (BFGS) для вычисления направления спуска в сочетании с одномерной оптимизацией в этом направлении. Оценка для таких методов вычисляется либо по всему задачнику (batch optimisation) либо по его подмножествам ("страницам" задачника,[9] которые впоследствии получили название "mini-batches"). В настоящее время такая точка зрения стала общепринятой.[10]

Следует также отметить возможность переобучения сети (overfitting), что является скорее результатом ошибочного проектирования её топологии и/или неправильным выбором критерия остановки обучения. При переобучении теряется свойство сети обобщать информацию. Весь набор образов, предоставленных к обучению, будет выучен сетью, но любые другие образы, даже очень похожие, могут быть распознаны неверно.

Литература

1. *Уоссермен Ф.* Нейрокомпьютерная техника: Теория и практика (http://evrika.tsi.lv/index.php?name=texts&file=show&f=410). — М.: «Мир», 1992. Архивная копия (http://web.archive.org/web/20090630215858/http://evrika.tsi.lv/index.php?name=texts&file=show&f=410) от 30 июня 2009 на Wayback Machine
2. *Хайкин С.* Нейронные сети: Полный курс. Пер. с англ. Н. Н. Куссуль, А. Ю. Шелестова. 2-е изд., испр. — М.: Издательский дом Вильямс, 2008, 1103 с.

Ссылки

1. *Горбань А. Н., Россиев Д. А.,* Нейронные сети на персональном компьютере (https://www.researchgate.net/publication/334971618_Neural_Networks_on_Personal_Computer). — Новосибирск: Наука, 1996. — 276 с.
2. *Копосов А. И., Щербаков И. Б., Кисленко Н. А., Кисленко О. П., Варивода Ю. В. и др.* Отчет по научно-исследовательской работе "Создание аналитического обзора информационных источников по применению нейронных сетей для задач газовой технологии" (http://cache.rcom.ru/~dap/nneng/nnlinks/nbdoc/common.htm). — М.: ВНИИГАЗ, 1995. Архивная копия (http://web.archive.org/web/20070108161822/http://cache.rcom.ru/~dap/nneng/nnlinks/nbdoc/common.htm) от 8 января 2007 на Wayback Machine
3. Книги по нейроинформатике на сайте NeuroSchool. (https://web.archive.org/web/20081204064930/http://neuroschool.narod.ru/books.html)
4. *Терехов С. А.*, Лекции по теории и приложениям искусственных

нейронных сетей. (http://alife.narod.ru/lectures/neural/Neu_ch01.htm)
5. *Миркес Е. М.*, Нейроинформатика: Учеб. пособие для студентов с программами для выполнения лабораторных работ. (https://web.archive.org/web/20080611081115/http://www.softcraft.ru/neuro/ni/p00.shtml) Красноярск: ИПЦ КГТУ, 2002, 347 с. Рис. 58, табл. 59, библиогр. 379 наименований. ISBN 5-7636-0477-6
6. Принцип обучения многослойной нейронной сети с помощью алгоритма обратного распространения (http://robocraft.ru/blog/algorithm/560.html)
7. Алгоритм обратного распространения ошибки с регуляризацией на C# (http://habrahabr.ru/post/154369/)

Примечания

1. *Галушкин А. И.* Синтез многослойных систем распознавания образов. — М.: «Энергия», 1974.
2. *Werbos P. J.*, Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.
3. *Rumelhart D.E., Hinton G.E., Williams R.J.*, Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986.
4. *Барцев С. И., Охонин В. А.* Адаптивные сети обработки информации. Красноярск : Ин-т физики СО АН СССР, 1986. Препринт N 59Б. — 20 с.
5. *Барцев С. И., Гилев С. Е., Охонин В. А.*, Принцип двойственности в организации адаптивных сетей обработки информации, В кн.: Динамика химических и биологических систем. — Новосибирск: Наука, 1989. — С. 6-55.
6. *Миркес Е. М.*, Нейрокомпьютер. Проект стандарта (http://pca.narod.ru/MirkesNeurocomputer.htm). — Новосибирск: Наука, Сибирская издательская фирма РАН, 1999. — 337 с. ISBN 5-02-031409-9 Другие копии онлайн: アーカイブされたコピー (http://neuroschool.narod.ru/books/mirkes/mirkes.html). Дата обращения 15 октября 2008. Архивировано (https://web.archive.org/web/20090703165215/http://neuroschool.narod.ru/books/mirkes/mirkes.html) 3 июля 2009 года..
7. *LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey.* Deep learning (англ.) // Nature. — 2015. — Vol. 521. — P. 436—444. — doi:10.1038/nature14539 (https://dx.doi.org/10.1038%2Fnature14539).
8. *Wasserman P. D.* Experiments in translating Chinese characters using backpropagation. Proceedings of the Thirty-Third IEEE Computer Society International Conference. — Washington: D. C.: Computer Society Press of the IEEE, 1988.
9. *Горбань А. Н.* Обучение нейронных сетей (http://lib.sibnet.ru/book/11961). — М.: USSR-USA СП ПараГраф, 1990.
10. Goodfellow I, Bengio Y, Courville A. Deep learning (http://faculty.neu.edu.cn/yury/AAI/Textbook/DeepLearningBook.pdf) . MIT press; 2016 Nov 10.