Алгоритм обучения многослойной нейронной сети методом обратного распространения ошибки

Прудников Иван Алексеевич
МИРЭА(МТУ)

Тема нейронных сетей была уже ни раз освещена во многих журналах, однако сегодня я бы хотел познакомить читателей с алгоритмом обучения многослойной нейронной сети методом обратного распространения ошибки и привести реализацию данного метода.

Сразу хочу оговориться, что не являюсь экспертом в области нейронных сетей, поэтому жду от читателей конструктивной критики, замечаний и дополнений.

Теоретическая часть

Данный материал предполагает знакомство с основами нейронных сетей, однако я считаю возможным ввести читателя в курс темы без излишних мытарств по теории нейронных сетей. Итак, для тех, кто впервые слышит словосочетание «нейронная сеть», предлагаю воспринимать нейронную сеть в качестве взвешенного направленного графа, узлы (нейроны) которого расположены слоями. Кроме того, узел одного слоя имеет связи со всеми узлами предыдущего слоя. В нашем случае у такого графа будут иметься входной и выходной слои, узлы которых выполняют роль входов и выходов соответственно. Каждый узел (нейрон) обладает активационной функцией — функцией, ответственной за вычисление сигнала на выходе узла (нейрона). Также существует понятие смещения, представляющего из себя узел, на выходе которого всегда появляется единица. В данной статье мы будем рассматривать процесс обучения нейронной сети, предполагающий наличие «учителя», то есть процесс обучения, при котором обучение происходит путем предоставления сети последовательности обучающих примеров с правильными откликами.
Как и в случае с большинством нейронных сетей, наша цель состоит в обучении сети таким образом, чтобы достичь баланса между способностью сети давать верный отклик на входные данные, использовавшиеся в процессе обучения (запоминания), и способностью выдавать правильные результаты в ответ на входные данные, схожие, но неидентичные тем, что были использованы при обучении (принцип обобщения). Обучение сети методом обратного распространения ошибки включает в себя три этапа: подачу на вход данных, с последующим распространением данных в направлении выходов, вычисление и обратное распространение соответствующей ошибки и корректировку весов. После обучения предполагается лишь подача на вход сети данных и распространение их в направлении выходов. При этом, если обучение сети может являться довольно длительным процессом, то непосредственное вычисление результатов обученной сетью происходит очень быстро. Кроме того, существуют многочисленные вариации метода обратного распространения ошибки, разработанные с целью увеличения скорости протекания процесса обучения.
Также стоит отметить, что однослойная нейронная сеть существенно ограничена в том, обучению каким шаблонам входных данных она подлежит, в то время, как многослойная сеть (с одним или более скрытым слоем) не имеет такого недостатка. Далее будет дано описание стандартной нейронной сети с обратным распространением ошибки.

Архитектура

На рисунке 1 показана многослойная нейронная сеть с одним слоем скрытых нейронов (элементы Z).

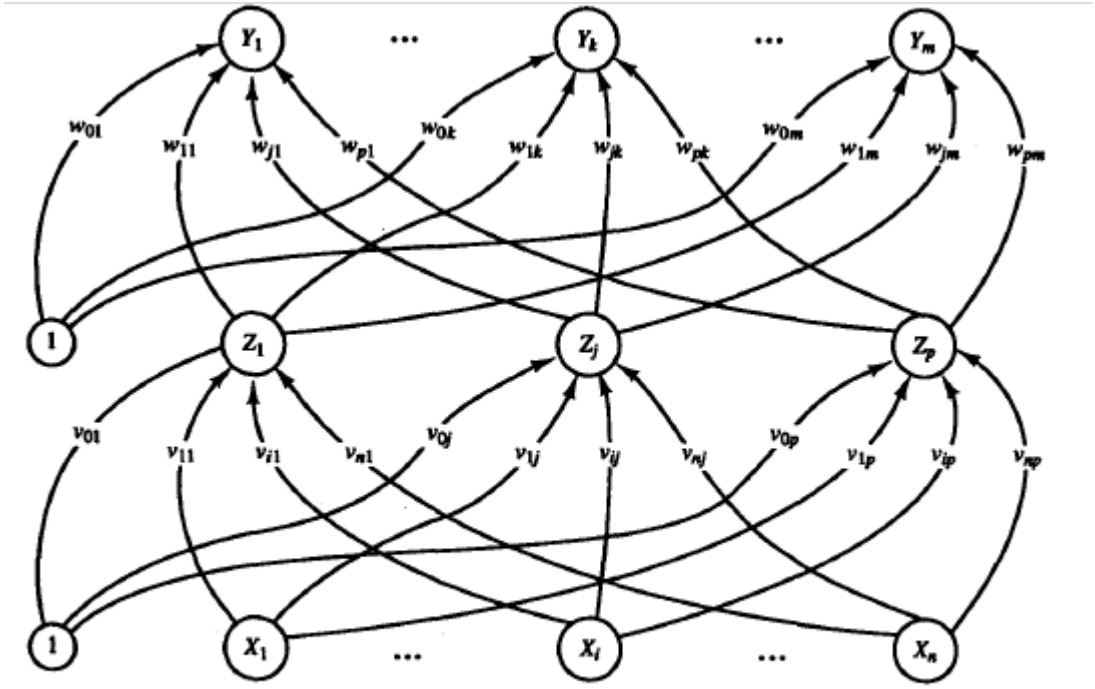

рисунок 1. Сеть с обратным распространением ошибки с одним скрытым слоем

Нейроны, представляющие собой выходы сети (обозначены Y), и скрытые нейроны могут иметь смещение(как показано на изображении). Смещение, соответствующий выходу Y_k обозначен w_{ok}, скрытому элементу Z_j — V_{oj}. Эти смещения служат в качестве весов на связях, исходящих от нейронов, на выходе которых всегда появляется 1 (на рисунке 1 они показаны, но обычно явно не отображаются, подразумеваясь). Кроме того, на рисунке 1 стрелками показано перемещение информации в ходе фазы распространения данных от входов к выходам. В процессе обучения сигналы распространяются в обратном направлении.

Описание алгоритма

Алгоритм, представленный далее, применим к нейронной сети с одним скрытым слоем, что является допустимой и адекватной ситуацией для большинства приложений. Как уже было сказано ранее, обучение сети включает в себя три стадии: подача на входы сети обучающих данных, обратное распространение ошибки и корректировка весов. В ходе первого этапа каждый входной нейрон X_i получает сигнал и широковещательно транслирует его каждому из скрытых нейронов $Z_1,Z_2...,Z_p$. Каждый скрытый нейрон затем вычисляет результат его активационной функции (сетевой функции) и рассылает свой сигнал Z_j всем выходным нейронам. Каждый выходной нейрон Y_k, в свою очередь, вычисляет результат своей активационной функции Y_k, который представляет собой ничто иное, как выходной сигнал данного нейрона для соответствующих входных данных. В процессе обучения, каждый нейрон на выходе сети сравнивает вычисленное значение Y_k с предоставленным учителем t_k (целевым значением), определяя соответствующее значение ошибки для данного входного шаблона. На основании этой ошибки вычисляется σ_k (k = 1,2,...m). σ_k используется при распространении ошибки от Y_k до всех элементов сети предыдущего слоя (скрытых нейронов, связанных с Y_k), а также позже при изменении весов связей между выходными нейронами и скрытыми. Аналогичным образом вычисляется σj (j = 1,2,...p) для каждого скрытого нейрона Z_j. Несмотря на то, что распространять ошибку до входного слоя необходимости нет, σj используется для изменения весов связей между нейронами скрытого слоя и входными

нейронами. После того как все σ были определены, происходит одновременная корректировка весов всех связей.

Обозначения:

В алгоритме обучения сети используются следующие обозначения:

X Входной вектор обучающих данных $X = (X_1, X_2,...,X_i,...,X_n)$.
t Вектор целевых выходных значений, предоставляемых учителем $t = (t_1, t_2,...,t_k,...,t_m)$
σ_k Составляющая корректировки весов связей w_{jk}, соответствующая ошибке выходного нейрона Y_k; также, информация об ошибке нейрона Y_k, которая распространяется тем нейронам скрытого слоя, которые связаны с Y_k.
σ_j Составляющая корректировки весов связей v_{ij}, соответствующая распространяемой от выходного слоя к скрытому нейрону Z_j информации об ошибке.
a Скорость обучения.
X_i Нейрон на входе с индексом i. Для входных нейронов входной и выходной сигналы одинаковы — X_i.
v_{oj} Смещение скрытого нейрона j.
Z_j Скрытый нейрон j; Суммарное значение подаваемое на вход скрытого элемента Z_j обозначается Z_in_j: $Z\_in_j = v_{oj} + \sum x_i * v_{ij}$
Сигнал на выходе Z_j (результат применения к Z_in_j активационной функции) обозначается Z_j: $Z_j = f(Z\_in_j)$
w_{ok} Смещение нейрона на выходе.
Y_k Нейрон на выходе под индексом k; Суммарное значение подаваемое на вход выходного элемента Y_k обозначается Y_in_k: $Y\_in_k = w_{ok} + \sum Z_j * w_{jk}$. Сигнал на выходе Y_k (результат применения к Y_in_k активационной функции) обозначается Y_k:

Функция активации

Функция активация в алгоритме обратного распространения ошибки должна обладать несколькими важными характеристиками: непрерывностью, дифференцируемостью и являться монотонно неубывающей. Более того, ради эффективности вычислений, желательно, чтобы ее производная легко находилась. Зачастую, активационная функция также является функцией с насыщением. Одной из наиболее часто используемых активационных функций является бинарная сигмоидальная функция с областью значений в (0, 1) и определенная как:

$$f_1(x) = \frac{1}{1 + e^{-x}}$$

$$f_1'(x) = f_1(x)*[1 - f_1(x)]$$

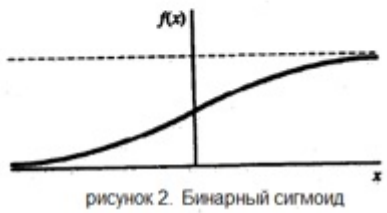

рисунок 2. Бинарный сигмоид

Другой широко распространенной активационной функцией является биполярный сигмоид с

областью значений (-1, 1) и определенный как:

$$f_2(x) = \frac{2}{1 + e^{-x}} - 1$$

$$f_2'(x) = \frac{1}{2}[1 + f_2(x)]*[1 - f_2(x)]$$

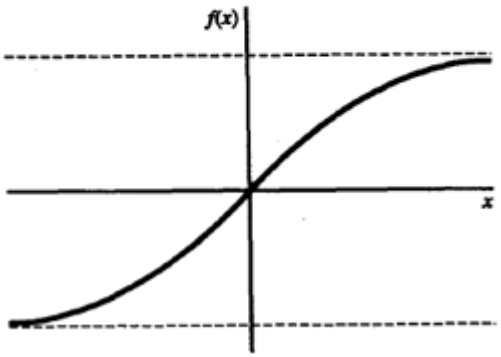

рисунок 3. Биполярный сигмоид.

Алгоритм обучения

Алгоритм обучения выглядит следующим образом:

Шаг 0.

Инициализация весов (веса всех связей инициализируются случайными небольшими значениями).

Шаг 1.

До тех пор пока условие прекращения работы алгоритма неверно, выполняются шаги 2 — 9.

Шаг 2.

Для каждой пары { данные, целевое значение } выполняются шаги 3 — 8.

Распространение данных от входов к выходам:

Шаг 3.

Каждый входной нейрон (X_i, i = 1,2,...,n) отправляет полученный сигнал X_i всем нейронам в следующем слое (скрытом).

Шаг 4.

Каждый скрытый нейрон (Z_j, j = 1,2,...,p) суммирует взвешенные входящие сигналы: $z\_in_j = v_{oj} + \sum x_i*v_{ij}$ и применяет активационную функцию: $z_j = f(z\_in_j)$ После чего посылает результат всем элементам следующего слоя (выходного).

Шаг 5.

Каждый выходной нейрон (Y_k, k = 1,2,...m) суммирует взвешенные входящие сигналы: $Y\_in_k = w_{ok} + \sum Z_j*w_{jk}$ и применяет активационную функцию, вычисляя выходной сигнал: $Y_k = f(Y\_in_k)$.

Обратное распространение ошибки:

Шаг 6.

Каждый выходной нейрон (Y_k, k = 1,2,...m) получает целевое значение — то выходное значение, которое является правильным для данного входного сигнала, и вычисляет ошибку: $\sigma_k = (t_k$

- y_k)*f ' (y_in_k), так же вычисляет величину, на которую изменится вес связи w_{jk} : $\Delta w_{jk} = a * \sigma_k * z_j$. Помимо этого, вычисляет величину корректировки смещения: $\Delta w_{ok} = a*\sigma_k$ и посылает σ_k нейронам в предыдущем слое.

Шаг 7.

Каждый скрытый нейрон (z_j, j = 1,2,...p) суммирует входящие ошибки (от нейронов в последующем слое) $\sigma\_in_j = \sum \sigma_k * w_{jk}$ и вычисляет величину ошибки, умножая полученное значение на производную активационной функции: $\sigma_j = \sigma\_in_j * f ' (z\_in_j)$, так же вычисляет величину, на которую изменится вес связи vij: $\Delta v_{ij} = a * \sigma_j * x_i$. Помимо этого, вычисляет величину корректировки смещения: $v_{oj} = a * \sigma_j$

Шаг 8. Изменение весов.

Каждый выходной нейрон (y_k, k = 1,2,...,m) изменяет веса своих связей с элементом смещения и скрытыми нейронами: $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$

Каждый скрытый нейрон (z_j, j = 1,2,...p) изменяет веса своих связей с элементом смещения и выходными нейронами: $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

Шаг 9.

Проверка условия прекращения работы алгоритма.

Условием прекращения работы алгоритма может быть как достижение суммарной квадратичной ошибкой результата на выходе сети предустановленного заранее минимума в ходе процесса обучения, так и выполнения определенного количества итераций алгоритма. В основе алгоритма лежит метод под названием градиентный спуск. В зависимости от знака, градиент функции (в данном случае значение функции — это ошибка, а параметры — это веса связей в сети) дает направление, в котором значения функции возрастают (или убывают) наиболее стремительно.