С.Короткий

Нейронные сети: алгоритм обратного распространения

В статье рассмотрен алгоритм обучения нейронной сети с помощью процедуры обратного распространения.

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного перцептрона[1]. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более слойный перцептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод "тыка", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант – распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения. Именно он будет рассмотрен в дальнейшем.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \qquad (1)$$

где $y_{j,p}^{(N)}$ – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа; d_{jp} – идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (2)$$

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя n-1 с j-ым нейроном слоя n, η – коэффициент скорости обучения, $0<\eta<1$.

Как показано в [2],

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}} \tag{3}$$

Здесь под y_j, как и раньше, подразумевается выход нейрона j, а под s_j – взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель dy_j/ds_j является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса

$$\frac{dy}{ds} = 1 - s^2 \tag{4}$$

Третий множитель $\partial s_j/\partial w_{ij}$, очевидно, равен выходу нейрона предыдущего слоя $y_i^{(n-1)}$.

Что касается первого множителя в (3), он легко раскладывается следующим образом[2]:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \tag{5}$$

Здесь суммирование по k выполняется среди нейронов слоя n+1.

Введя новую переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \tag{6}$$

мы получим рекурсивную формулу для расчетов величин $\delta_j^{(n)}$ слоя n из величин $\delta_k^{(n+1)}$ более старшего слоя n+1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \tag{7}$$

Для выходного же слоя

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \tag{8}$$

Теперь мы можем записать (2) в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \tag{9}$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (9) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \tag{10}$$

где μ – коэффициент инерционности, t – номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Напомним, что

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} \cdot w_{ij}^{(n)} \tag{11}$$

где M – число нейронов в слое n-1 с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; $y_i^{(n-1)}=x_{ij}^{(n)}$ – i-ый вход нейрона j слоя n.

$y_j^{(n)} = f(s_j^{(n)})$, где $f()$ – сигмоид (12)

$y_q^{(0)}=I_q$, (13)

где I_q – q-ая компонента вектора входного образа.

2. Рассчитать $\delta^{(N)}$ для выходного слоя по формуле (8).

Рассчитать по формуле (9) или (10) изменения весов $\Delta w^{(N)}$ слоя N.

3. Рассчитать по формулам (7) и (9) (или (7) и (10)) соответственно $\delta^{(n)}$ и $\Delta w^{(n)}$ для всех остальных слоев, n=N-1,...1.

4. Скорректировать все веса в НС

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \tag{14}$$

5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

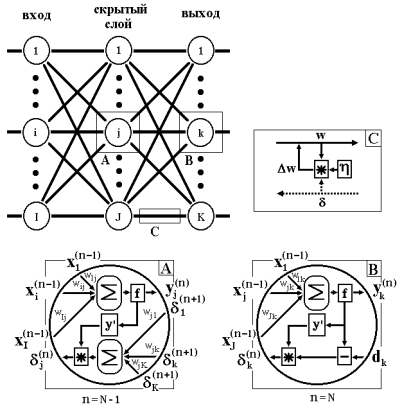

Рис.1 Диаграмма сигналов в сети при обучении по алгоритму обратного распространения

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других. Алгоритм иллюстрируется рисунком 1.

Из выражения (9) следует, что когда выходное значение $y_i^{(n-1)}$ стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться[3], поэтому область возможных значений выходов нейронов [0,1] желательно сдвинуть в пределы [-0.5,+0.5], что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1+e^{-\alpha \cdot x}} \tag{15}$$

Теперь коснемся вопроса емкости НС, то есть числа образов, предъявляемых на ее входы, которые она способна научиться распознавать. Для сетей с числом слоев больше двух, он остается открытым. Как показано в [4], для НС с двумя слоями, то есть выходным и одним скрытым слоем, детерминистская емкость сети C_d оценивается так:

$$N_w/N_y<C_d<N_w/N_y\cdot\log(N_w/N_y) \quad (16)$$

где N_w – число подстраиваемых весов, N_y – число нейронов в выходном слое.

Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов N_x и нейронов в скрытом слое N_h должно удовлетворять неравенству $N_x+N_h>N_y$. Во-вторых, $N_w/N_y>1000$. Однако вышеприведенная оценка выполнялась для сетей с активационными функциями нейронов в виде порога, а емкость сетей с гладкими активационными функциями, например – (15), обычно больше[4]. Кроме того, фигурирующее в названии емкости прилагательное "детерминистский" означает, что полученная оценка емкости подходит абсолютно для всех возможных входных образов, которые могут быть представлены N_x входами. В действительности распределение входных образов, как правило, обладает некоторой регулярностью, что позволяет НС проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение образов, в общем случае, заранее не известно, мы можем говорить о такой емкости только предположительно, но обычно она раза в два превышает емкость детерминистскую.

В продолжение разговора о емкости НС логично затронуть вопрос о требуемой мощности выходного слоя сети, выполняющего окончательную классификацию образов. Дело в том, что для разделения множества входных образов, например, по двум классам достаточно всего одного выхода. При этом каждый логический уровень – "1" и "0" – будет обозначать отдельный класс. На двух выходах можно закодировать уже 4 класса и так далее. Однако результаты работы сети, организованной таким образом, можно сказать – "под завязку", – не очень надежны. Для повышения достоверности классификации желательно ввести избыточность путем выделения каждому классу одного нейрона в выходном слое или, что еще лучше, нескольких, каждый из которых обучается определять принадлежность образа к классу со своей степенью достоверности, например: высокой, средней и низкой. Такие НС позволяют проводить классификацию входных образов, объединенных в нечеткие (размытые или пересекающиеся) множества. Это свойство приближает подобные НС к условиям реальной жизни.

Рассматриваемая НС имеет несколько "узких мест". Во-первых, в процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов сместят рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут в соответствие с (7) и (8) к остановке обучения, что парализует НС. Во-вторых, применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве η обычно выбирается

число меньше 1, но не очень маленькое, например, 0.1, и оно, вообще говоря, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов застабилизируются, η кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

Существует и иной метод исключения локальных минимумов, а заодно и паралича НС, заключающийся в применении стохастических НС, но о них лучше поговорить отдельно.

Теперь мы можем обратиться непосредственно к программированию НС. Следует отметить, что число зарубежных публикаций, рассматривающих программную реализацию сетей, ничтожно мало по сравнению с общим числом работ на тему нейронных сетей, и это при том, что многие авторы опробывают свои теоретические выкладки именно программным способом, а не с помощью нейрокомпьютеров и нейроплат, в первую очередь из-за их дороговизны. Возможно, это вызвано тем, что к программированию на западе относятся как к ремеслу, а не науке. Однако в результате такой дискриминации остаются неразобранными довольно важные вопросы.

Литература

1. С.Короткий, Нейронные сети: основные положения.
2. Sankar K. Pal, Sushmita Mitra, Multilayer Perceptron, Fuzzy Sets, and Classification //IEEE Transactions on Neural Networks, Vol.3, N5,1992, pp.683-696.
3. Ф.Уоссермен, Нейрокомпьютерная техника, М., Мир, 1992.
4. Bernard Widrow, Michael A. Lehr, 30 Years of Adaptive NeuralNetworks: Perceptron, Madaline, and Backpropagation //Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992, pp.327-354.
5. Paul J. Werbos, Backpropagation Through Time: What It Does and How to Do It //Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992, pp.309-319.
6. Gael de La Croix Vaubois, Catherine Moulinoux, Benolt Derot, The N Programming Language //Neurocomputing, NATO ASI series, vol.F68, pp.89-92.
7. H.A.Malki, A.Moghaddamjoo, Using the Karhunen-Loe`ve Transformation in the Back-Propagation Training Algorithm //IEEE Transactions on Neural Networks, Vol.2, N1, 1991, pp.162-165.
8. Harris Drucker, Yann Le Cun, Improving Generalization Performance Using Backpropagation //IEEE Transactions on Neural Networks, Vol.3, N5, 1992, pp.991-997.
9. Alain Petrowski, Gerard Dreyfus, Claude Girault, Performance Analysis of a Pipelined Backpropagation Parallel Algorithm //IEEE Transactions on Neural Networks, Vol.4, N6, 1993, pp.970-981.