## Лабораторная работа №4. Алгоритм 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 персептрона [1]. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более слойный персептрон 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.

**Процедура обратного распространения. Математический аспект**

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

 (1)

где  – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа; *djp* – 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:

 (2)

Здесь *wij* – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя n-1 с j-ым нейроном слоя n, *η* – коэффициент скорости обучения, 0<*η*<1.

Как показано в [2],

 (3)

Здесь под *yj*, как и раньше, подразумевается выход нейрона j, а под *sj* – взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель *dyj*/*dsj* 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, функции единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В НС, обучаемых по алгоритму обратного распространения, применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса

 (4)

Третий множитель ∂*sj*/∂*wij*, очевидно, равен выходу нейрона предыдущего слоя *yi(n-1)*.

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

 (5)

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

Введя новую переменную

 (6)

мы получим рекурсивную формулу для расчетов величин *δj(n)* слоя n из величин *δk(n+1)* более старшего слоя *n+1*.

 (7)

Для выходного же слоя

 (8)

Теперь мы можем записать (2) в раскрытом виде:

 (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

 (10)

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

**Алгоритм процедуры обратного распространения**

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

 (11)

где M – число нейронов в слое n-1 с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; *yi(n-1)=xij(n)* – i-ый вход нейрона j слоя n.

*yj(n) = f(sj(n))*, где *f()* – сигмоид (12)

*yq(0)=Iq*, (13)

где *Iq* – q-ая компонента вектора входного образа.

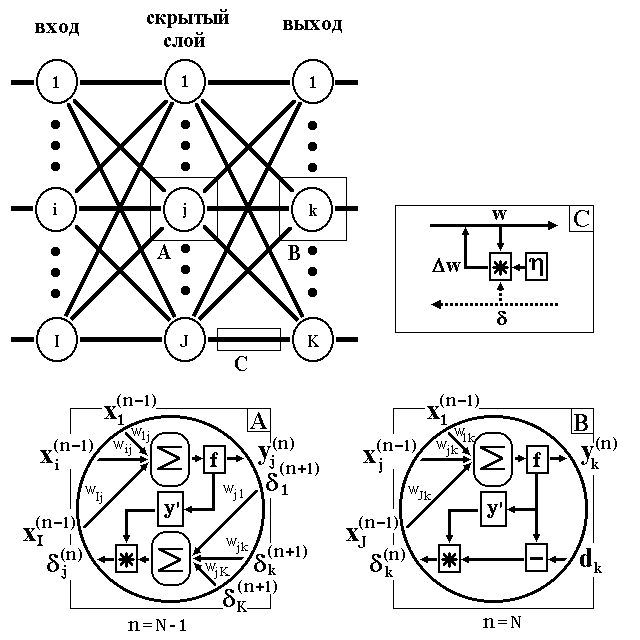
1. Рассчитать *δ****(N)*** для выходного слоя по формуле (8).

Рассчитать по формуле (9) или (10) изменения весов *Δw****(N)*** слоя N.

1. Рассчитать по формулам (7) и (9) (или (7) и (10)) соответственно *δ(****n)*** и *Δw****(n)*** для всех остальных слоев, n=N-1,...1.
2. Скорректировать все веса в НС

 (14)

1. 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 *yi(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

 (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 Cd оценивается так:

Nw/Ny<Cd<Nw/Ny⋅log(Nw/Ny) (16)

где Nw – число подстраиваемых весов, Ny – число нейронов в выходном слое.

Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов Nx и нейронов в скрытом слое Nh должно удовлетворять неравенству Nx+Nh>Ny. Во-вторых, Nw/Ny>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 Nx 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.

**Замечания по программированию**

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 не разобранными довольно важные вопросы.

Как видно из формул, описывающих алгоритм функционирования и обучения НС, весь этот процесс может быть записан и затем запрограммирован в терминах и с применением операций матричной алгебры, что сделано, например, в [5]. Судя по всему, такой подход обеспечит более быструю и компактную реализацию НС, нежели ее воплощение на базе концепций объектно-ориентированного (ОО) программирования. Однако в последнее время преобладает именно ОО подход, причем зачастую разрабатываются специальные ОО языки для программирования НС [6], хотя, с моей точки зрения, универсальные ОО языки, например C++ и Pascal, были созданы как раз для того, чтобы исключить необходимость разработки каких-либо других ОО языков, в какой бы области их не собирались применять.

И все же программирование НС с применением ОО подхода имеет свои плюсы. Во-первых, оно позволяет создать гибкую, легко перестраиваемую иерархию моделей НС. Во-вторых, такая реализация наиболее прозрачна для программиста, и позволяет конструировать НС даже непрограммистам. В-третьих, уровень абстрактности программирования, присущий ОО языкам, в будущем будет, по-видимому, расти, и реализация НС с ОО подходом позволит расширить их возможности.

**Задания для самостоятельной работы**

По заданию преподавателя на основе алгоритма обучения обратного распространения реализовать НС для интерполяции входных данных каким-либо полиномом. Результаты оформить в виде отчета и защитить на следующем занятии.

**Контрольные вопросы**

1. Изложите основные концепции процедуры обратного распространения.
2. Изложите математические аспекты процедуры обратного распространения.
3. Изложите алгоритм процедуры обратного распространения.
4. Что можно сказать о емкости НС?
5. Каковы достоинства процедуры обратного распространения?
6. Каковы недостатки процедуры обратного распространения и как они преодолеваются?

**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.

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.