Курс лекций ««Введение в ИИ.»». Часть III. Нейрокомпьтинг. Лекция 16. Процедура обратного распространения

О.Г. Чанышев

Содержание

1 Введение

Обратное распространение – это систематический метод для обучения многослойных искусственных нейронных сетей. Он имеет солидное математическое обоснование. Несмотря на некоторые ограничения, процедура обратного распространения сильно расширила область проблем, в которых могут быть использованы искусственные нейронные сети, и убедительно продемонстрировала свою мощь. Разработка алгоритма обратного распространения сыграла важную роль в возрождении интереса к искусственным нейронным сетям.

Рис. 1: Искусственный нейрон с активационнной функцией

Рис. 2: Сигмоидальная активационная функция.

2 Обучающий алгоритм обратного распространения

2.1 Сетевые конфигурации

На рис. 1 показан нейрон, используемый в качестве основного строительного блока в сетях обратного распространения. Подается множество входов, идущих либо извне, либо от предшествующего слоя. Каждый из них умножается на вес, и произведения суммируются. Эта сумма, обозначаемая NET, должна быть вычислена для каждого нейрона сети. После того, как величина NET вычислена, она модифицируется с помощью активационной функции и получается сигнал OUT.

На рис.2 показана активационная функция, обычно используемая для обратного распространения.

$$OUT = \frac{1}{1 + e^{-NET}} \tag{1}$$

Как показывает уравнение 2, эта функция, называемая сигмоидом, весьма удобна, так как имеет простую производную, что используется при реализации алгоритма обратного распространения.

$$\frac{\partial OUT}{\partial NET} = OUT(1 - OUT) \tag{2}$$

Сигмоид, который иногда называется также логистической, или сжимающей функцией, сужает диапазон изменения NET так, что значение OUT лежит между нулем и единицей. Как указывалось выше, многослойные нейронные сети обладают большей представляющей мощностью, чем однослойные, только в случае присутствия нелинейности. Сжимающая функция обеспечивает требуемую нелинейность. Сигмоид, также, удовлетворяет требованию дифференцируемости на всей области определения. Его дополнительное преимущество состоит в автоматическом контроле усиления. Для слабых сигналов (величина NET близка к нулю) кривая вход-выход имеет сильный наклон, дающий большое усиление. Когда величина сигнала становится больше, усиление падает. Таким образом, большие сигналы воспринимаются сетью без насыщения, а слабые сигналы проходят по сети без чрезмерного ослабления.

2.2 Многослойная сеть

На рис. 3 изображена многослойная сеть, которая может обучаться с помощью процедуры обратного распространения. Первый слой нейронов (соединенный с входами) служит лишь в качестве распределительных точек (в дальнейшем - нулевой слой), суммирования входов здесь не производится. Входной сигнал просто проходит через них к весам на их выходах. А каждый нейрон последующих слоев выдает сигналы NET и OUT, как описано выше.

Рис. 3: Двухслойная сеть обратного распространения

Сеть на рис. 3 рассматривается как двухслойная. Нейрон объединен с множеством весов, присоединенных к его входу. Таким образом, веса первого слоя оканчиваются на нейронах первого слоя. Процедура обратного распространения применима к сетям с любым числом слоев. Однако для того, чтобы продемонстрировать алгоритм, достаточно двух слоев. Сейчас будут рассматриваться лишь сети прямого действия, хотя обратное распространение применимо и к сетям с обратными связями.

2.3 Обзор обучения

Целью обучения сети является такая подстройка ее весов, чтобы приложение некоторого множества входных векторов приводило к требуемому множеству выходных. При обучении предполагается, что для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются обучающей парой. Как правило, сеть обучается на многих парах. Такая группа обучающих пар называется обучающим множеством.

Перед началом обучения всем весам должны быть присвоены небольшие начальные значения, выбранные случайным образом. Это гарантирует, что в сети не произойдет насыщения большими значениями весов, и предотвращает ряд других патологических случаев. Например, если всем весам придать одинаковые начальные значения, а для требуемого функционирования нужны неравные значения, то сеть не сможет обучиться.

Обучение сети обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Подкорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Операции, выполняемые шагами 1 и 2, сходны с теми, которые выполняются при функционировании уже обученной сети, т. е. подается входной вектор и вычисляется получающийся выход. Вычисления выполняются послойно. На рис. 3 сначала вычисляются выходы нейронов слоя j, затем они используются в качестве входов слоя k, вычисляются выходы нейронов слоя k, которые и образуют выходной вектор сети.

На шаге 3 каждый из выходов сети, которые на рис. 3 обозначены OUT, вычитается из соответствующей компоненты целевого вектора, чтобы получить ошибку. Эта ошибка используется на шаге 4 для коррекции весов сети, причем знак и величина изменений весов определяются алгоритмом обучения (см. ниже).

После достаточного числа повторений этих четырех шагов разность между действительными выходами и целевыми выходами должна уменьшиться до приемлемой величины, при этом говорят, что сеть обучилась. Теперь сеть используется для распознавания и веса не изменяются. На шаги 1 и 2 можно смотреть как на «проход вперед», так как сигнал распространяется по сети от входа к выходу. Шаги 3, 4 составляют «обратный проход», здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

Проход вперед. Шаги 1 и 2 могут быть выражены в векторной форме следующим образом: подается входной вектор $\vec{X}$ и на выходе получается вектор $\vec{Y}$. Векторная пара вход-цель $\vec{X}, \vec{T}$ берется из обучающего множества. Вычисления проводятся над вектором $\vec{X}$, чтобы получить выходной вектор $\vec{Y}$. Вычисления в многослойных сетях выполняются слой за слоем, начиная

Рис. 4: Настройка веса в выходном слое

с ближайшего к входу слоя. Величина NET каждого нейрона первого слоя вычисляется как взвешенная сумма входов нейрона. Затем активационная функция F «сжимает» NET и дает величину OUT для каждого нейрона в этом слое. Когда множество выходов слоя получено, оно является входным множеством для следующего слоя. Процесс повторяется слой за слоем, пока не будет получено заключительное множество выходов сети. Веса между нейронами могут рассматриваться как матрица W. Тогда NET-вектор слоя N может быть выражен не как сумма произведений, а как произведение $\vec{X}$ и W. В векторном обозначении $\vec{N} = \vec{X}W$. Покомпонентным применением функции F к NET-вектору $\vec{N}$ получается выходной вектор $\vec{O}$. Таким образом, для данного слоя вычислительный процесс описывается следующим выражением:

$$\overrightarrow{O} = F(\overrightarrow{X}W) \tag{3}$$

Выходной вектор одного слоя является входным вектором для следующего, поэтому вычисление выходов последнего слоя требует применения уравнения 3 к каждому слою от входа сети к ее выходу.

Обратный проход. Подстройка весов выходного слоя. Так как для каждого нейрона выходного слоя задано целевое значение, то подстройка весов легко осуществляется с использованием модифицированного дельта-правила. Внутренние слои называют «скрытыми слоями», для их выходов не имеется целевых значений для сравнения. Поэтому обучение усложняется. На рис. 4 показан процесс обучения для одного веса от нейрона p в скрытом слое j к нейрону q в выходном слое k. Выход нейрона слоя k, вычитаясь из целевого значения ($Target$), дает сигнал ошибки. Он умножается на производную сжимающей функции $OUT(1 - OUT)$, вычисленную для этого нейрона слоя k, давая, таким образом, величину δ.

$$\delta = OUT(1 - OUT)(Target - OUT) \tag{4}$$

Затем δ умножается на величину OUT нейрона j, из которого выходит рассматриваемый вес. Это произведение в свою очередь умножается на коэффициент скорости обучения α (обычно от 0,01 до 1,0), и результат прибавляется к весу. Такая же процедура выполняется для каждого веса от нейрона скрытого слоя к нейрону в выходном слое.

Следующие уравнения иллюстрируют это вычисление:

$$\Delta w_{pq,k} = \eta \delta_{q,k} OUT \tag{5}$$

$$w_{pq,k}^{(n+1)} = w_{pq,k}^{(n)} + \Delta w_{pq,k} \tag{6}$$

где $w_{pq,k}^{(n)}$ – величина веса от нейрона p в скрытом слое k нейрону q в выходном слое на шаге n (до коррекции); отметим, что индекс k относится к слою, в котором заканчивается данный вес, т.е., согласно принятому в этой книге соглашению, с которым он объединен; $w_{pq,k}^{(n+1)}$ – величина веса на шаге $n+1$ (после коррекции); $\delta_{q,k}$ – величина δ для нейрона q, в выходном слое k; $OUT_{p,j}$ – величина OUT для нейрона p в скрытом слое j.

2.4 Подстройка весов скрытого слоя

Рассмотрим один нейрон в скрытом слое, предшествующем выходному слою. При проходе вперед этот нейрон передает свой выходной сигнал нейронам в выходном слое через соединяющие

Рис. 5: Настройка веса в скрытом слое.

их веса. Во время обучения эти веса функционируют в обратном порядке, пропуская величину δ от выходного слоя назад к скрытому слою. Каждый из этих весов умножается на величину δ нейрона, к которому он присоединен в выходном слое. Величина Δ, необходимая для нейрона скрытого слоя, получается суммированием всех таких произведений и умножением на производную сжимающей функции:

$$\Delta_{q,k} = OUT_{p,j}(1 - OUT_{p,j})[\sum_{q} \delta_{q,k} w_{pq,k}](3.7) \quad (7)$$

(см. рис. 2.4). Когда значение Δ получено, веса, питающие первый скрытый уровень, могут быть подкорректированы с помощью уравнений 5 и 6, где индексы модифицируются в соответствии со слоем.

Для каждого нейрона в данном скрытом слое должно быть вычислено Δ и подстроены все веса, ассоциированные с этим слоем. Этот процесс повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

Обозначим множество величин δ выходного слоя через $\vec{D}_k$ и множество весов выходного слоя как массив W_k. Чтобы получить $\vec{D}_j$, δ-вектор выходного слоя, достаточно следующих двух операций:

1. Умножить вектор выходного слоя $\vec{D}_k$ на транспонированную матрицу весов W'_k, соединяющую скрытый уровень с выходным уровнем.

2. Умножить каждую компоненту полученного произведения на производную сжимающей функции соответствующего нейрона в скрытом слое.

В символьной записи:

$$\vec{D}_j = \vec{D}_k W'_k \$ [\vec{O}_j \$ (\vec{I} - \vec{O}_j)], \quad (8)$$

где оператор \$ в данной книге обозначает покомпонентное произведение векторов, $\vec{O}_j$ – выходной вектор слоя j и $\vec{I}$ – единичный вектор.

2.5 Добавление нейронного смещения

Во многих случаях желательно наделять каждый нейрон обучаемым смещением. Это позволяет сдвигать начало отсчета логистической функции, давая эффект, аналогичный подстройке порога персептронного нейрона, и приводит к ускорению процесса обучения. Эта возможность может быть легко введена в обучающий алгоритм с помощью добавляемого к каждому нейрону веса, присоединенного к +1. Этот вес обучается так же, как и все остальные веса, за исключением того, что подаваемый на него сигнал всегда равен +1, а не выходу нейрона предыдущего слоя.

2.6 Импульс

В работе [7] описан метод ускорения обучения для алгоритма обратного распространения, увеличивающий также устойчивость процесса. Этот метод, названный импульсом, заключается в добавлении к коррекции веса члена, пропорционального величине предыдущего изменения

веса. Как только происходит коррекция, она «запоминается» и служит для модификации всех последующих коррекций. Уравнения коррекции модифицируются следующим образом:

$$\Delta w_{pq,k}^{(n+1)} = \eta \delta_{q,k} OUT_{p,j} + \alpha \Delta w_{pq,k}^{(n)} \tag{9}$$

$$w_{pq,k}^{(n+1)} = w_{pq,k}^{(n)} + \Delta w_{pq,k}^{(n+1)} \tag{10}$$

где α – коэффициент импульса, обычно устанавливается около 0,9.

Используя метод импульса, сеть стремится идти по дну узких оврагов поверхности ошибки (если таковые имеются), а не двигаться от склона к склону. Этот метод, по-видимому, хорошо работает на некоторых задачах, но дает слабый или даже отрицательный эффект на других. В работе [8] описан сходный метод, основанный на экспоненциальном сглаживании, который может иметь преимущество в ряде приложений.

$$\Delta w_{pq,k}^{(n+1)} = (1-\alpha)\delta_{q,k} OUT_{p,j} + \alpha \Delta w_{pq,k}^{(n)} (3.9) \tag{11}$$

Затем вычисляется изменение веса

$$w_{pq,k}^{(n+1)} = w_{pq,k}^{(n)} + \eta \Delta w_{pq,k}^{(n+1)}, (3.10) \tag{12}$$

где α коэффициент сглаживания, варьируемый и диапазоне от 0,0 до 1,0. Если α равен 1,0, то новая коррекция игнорируется и повторяется предыдущая. В области между 0 и 1 коррекция веса сглаживается величиной, пропорциональной α. По-прежнему, η является коэффициентом скорости обучения, служащим для управления средней величиной изменения веса.

3 Применения

Обратное распространение было использовано в широкой сфере прикладных исследований. Некоторые из них описываются здесь, чтобы продемонстрировать мощь этого метода. Фирма NEC в Японии объявила недавно, что обратное распространение было ею использовано для визуального распознавания букв, причем точность превысила 99%. Это улучшение было достигнуто с помощью комбинации обычных алгоритмов с сетью обратного распространения, обеспечивающей дополнительную проверку.

В работе [8] достигнут впечатляющий успех с Net-Talk, системой, которая превращает печатный английский текст в высококачественную речь. Магнитофонная запись процесса обучения сильно напоминает звуки ребенка на разных этапах обучения речи.

В [2] обратное распространение использовалось в машинном распознавании рукописных английских слов. Буквы, нормализованные по размеру, наносились на сетку, и брались проекции линий, пересекающих квадраты сетки. Эти проекции служили затем входами для сети обратного распространения. Сообщалось о точности 99,7% при использовании словарного фильтра. В [3] сообщалось об успешном применении обратного распространения к сжатию изображений, когда образы представлялись одним битом на пиксель, что было восьмикратным улучшением по сравнению с входными данными.

4 Предостережение

Несмотря на многочисленные успешные применения обратного распространения, оно не является панацеей. Больше всего неприятностей приносит неопределенно долгий процесс обучения.

В сложных задачах для обучения сети могут потребоваться дни или даже недели, она может и вообще не обучиться. Длительное время обучения может быть результатом неоптимального выбора длины шага. Неудачи в обучении обычно возникают по двум причинам: паралича сети и попадания в локальный минимум.

4.1 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.

4.2 Локальные минимумы

Обратное распространение использует разновидность градиентного спуска, т. е. 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. Статистические методы обучения могут помочь избежать этой ловушки, но они медленны. В [10] предложен метод, объединяющий статистические методы машины Коши с градиентным спуском обратного распространения и приводящий к системе, которая находит глобальный минимум, сохраняя высокую скорость обратного распространения.

4.3 Размер шага

В доказательстве сходимости в [7] предполагается, что коррекции весов бесконечно малы. Ясно, что это неосуществимо на практике, так как ведет к бесконечному времени обучения. Размер шага должен браться конечным, и в этом вопросе приходится опираться только на опыт. Если размер шага очень мал, то сходимость слишком медленная, если же очень велик, то может возникнуть паралич или постоянная неустойчивость. В [11] описан адаптивный алгоритм выбора шага, автоматически корректирующий размер шага в процессе обучения.

4.4 Временная неустойчивость

Если сеть учится распознавать буквы, то нет смысла учить «Б», если при этом забывается «А». Процесс обучения должен быть таким, чтобы сеть обучалась на всем обучающем множестве без пропусков того, что уже выучено. В доказательстве сходимости [7] это условие выполнено, но требуется также, чтобы сети предъявлялись все векторы обучающего множества прежде, чем выполняется коррекция весов. Необходимые изменения весов должны вычисляться на всем множестве, а это требует дополнительной памяти; после ряда таких обучающих циклов веса

сойдутся к минимальной ошибке. Этот метод может оказаться бесполезным, если сеть находится в постоянно меняющейся внешней среде, так что второй раз один и тот же вектор может уже не повториться. В этом случае процесс обучения может никогда не сойтись, бесцельно блуждая или сильно осциллируя. В этом смысле обратное распространение не похоже на биологические системы.

Список литературы

[1] Almeida L. B. 1987. Neural computaters. Proceedings of NATO ARW on Neural Computers, Dusseldorf. Heidelberg: Springer-Verlag.

[2] Burr D. J. 1987. Experiments with a connecnionlist text reader. In Proceedings of the IEEE First International Conferense on Neural Networks, eds. M. Caudill and C.Butler, vol. 4, pp. 717-24. San Diego, CA: SOS Printing.

[3] Cottrell G. W., Munro P., Zipser D. 1987. Image compression by backpropagation: An example of extensional programming. ICS Report 8702, University of California, San Diego.

[4] Parker D. B. 1982. Learning logic. Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA.

[5] Parker D. B. 1987. Second order back propagation: Implementing an optimal 0(n) approximation to Newton's method as an artificial newral network. Manuscript submitted for publication.

[6] Pineda F. J. 1988. Generalization of backpropagation to recurrent and higher order networks. In Newral information processing systems, ed. Dana Z. Anderson, pp. 602-11. New York: American Institute of Phisycs.

[7] Rumelhart D. E., Hinton G. E., Williams R. J. 1986. Learning internal reprentations by error propagation. In Parallel distributed processing, vol. 1, pp. 318-62. Cambridge, MA: MIT Press.

[8] Sejnowski T. J., Rosenberg C. R. 1987. Parallel networks that learn to pronounce English text. Complex Systems 1:145-68.

[9] Stornetta W. S., Huberman B. A. 1987. An improwed three-layer, backpropagation algorithm. In Proceedings of the IEEE First International Conference on Newral Networks, eds. M. Caudill and C. Butler. San Diego, CA: SOS Printing.

[10] Wasserman P. D. 1988a. Combined backpropagation/Cauchy machine. Proceedings of the International Newral Network Society. New York: Pergamon Press.

[11] Wasserman P. D. 1988b. 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.

[12] Werbos P. J. 1974. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Masters thesis, Harward University.

Следующая лекция

Лекция 17. Сети встречного распространения