Нейронные сети. Краткий курс

Лекция 2

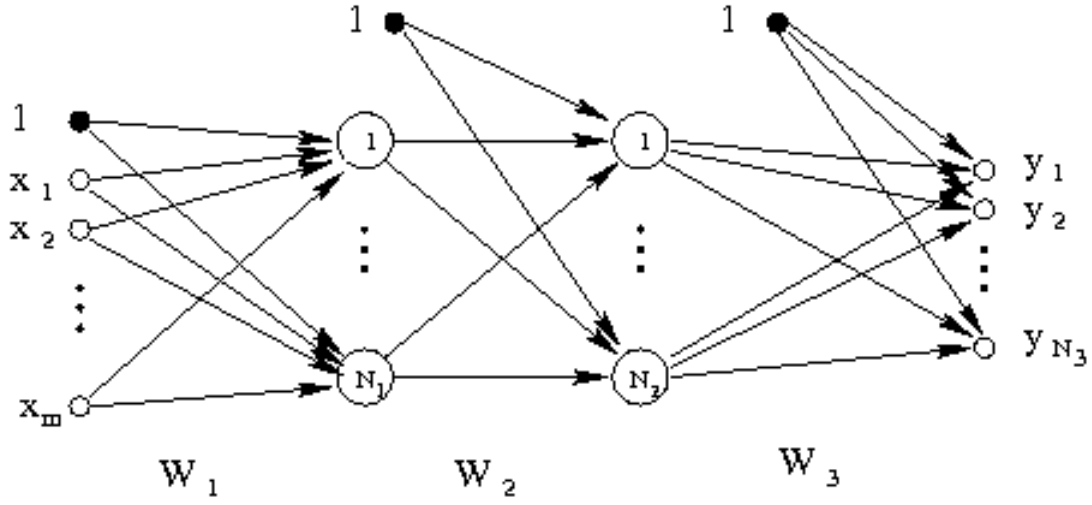
Алгоритм обратного распространения ошибок

Многослойные персептроны применяются для решения разнообразного круга задач. Обучение такой нейронной сети часто выполняется с помощью такого популярного алгоритма как *алгоритм обратного распространения ошибки*. Алгоритм основан на коррекции ошибок нетренированной сети. Обучение методом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямой и обратный. При прямом проходе входной вектор из учебного набора подается на входы сети и сигнал распространяется в прямом направлении, что приводит к генерации на выходе сети какого-то выходного вектора. При обратном же проходе выходной вектор вычитается из желаемого и имеющегося в учебном наборе, в результате чего формируется сигнал ошибки, который распространяется в направлении обратном направлению синаптических связей. При прямом проходе синаптические веса фиксированы, а при обратном осуществляется коррекция синаптических весов в соответствии с каким либо правилом коррекции. Синаптические веса настраиваются с целью максимального приближения фактического выхода сети к желаемому в статистическом смысле.

Следует отметить, что в многослойном персептроне как правило используются сигмоидальные функции активации, которые являются всюду дифференцируемыми и нелинейными. Рассмотрение функционирования многослойного персептрона посредством распространяющихся сигналов в прямом и обратном направлении является удобной аналогией, которая вовсе не является единственной и обязательной. Можно рассматривать отображение осуществляемое нейронной сетью прямого распространения как некоторую параметризованную функциональную зависимость выходных данных от входных, а обучение сети рассматривать как подбор наилучших в некотором смысле параметров.

Рассмотрим многослойный персептрон с двумя слоями скрытых нелинейных нейронов (см. схему ниже). Пусть в начальный момент времени, веса всех синаптических связей имют некоторые случайные значения. Для тренировки сети используется некоторое количество учебных примеров, состоящих из входных векторов $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_N$ и соответствующих желаемых откликов на выходе сети $\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \ldots, \boldsymbol{d}_N$ (обозначение буквой $\boldsymbol{d}$ происходит от английского слова *desired*). Последовательное обучение предполагает предъявление сети входных векторов один за одним и коррекцию весов в соответствии с выбранным правилом. Сигнал ошибки выходного нейрона *j* на итерации *n* (соответствующей *n*-ой учебной паре $\boldsymbol{x}_n$ и $\boldsymbol{d}_n$) определяется как

$$e_j(n) = d_j(n) - y_j(n) \quad . \tag{2.1}$$



Иногда вводят такую термодинамическую аналогию как энергия ошибки *j*-го нейрона определенную как $\frac{1}{2}e_j^2(n)$, тогда общая энергия ошибки сети определяется как

$$E(n)=\frac{1}{2}\sum_{j=1}^{N_3} e_j^2(n) \tag{2.2}$$

Собственно (2.2) можно трактовать как целевую функцию для минимизации (или функцию стоимости), зависящую от всех весов синаптических связей сети. Можно ввести «энергию» среднеквадратической ошибки, то есть усредненную энергию (2.2) по всем образцам учебного набора,

$$\bar{E}=\frac{1}{N}\sum_{n=1}^{N} E(n)=\frac{1}{2\mathrm{N}}\sum_{n=1}^{N}\sum_{j=1}^{N_3} e_j^2(n) \quad . \tag{2.3}$$

Выражение (2.3) представляет собой меру эффективности обучения НС.

Рассмотрим *j*-ый нейрон во втором слое сети на итерации *n* (то есть при подаче на входы сети *n*-го образца из учебного набора). Аргумент функции активации данного нейрона

$$v_j(n)=\sum_{i=0}^{N_1} w_{ji}(n)\,y_i(n) \quad , \tag{2.4}$$

где $y_i(n)$ - сигналы с выходов нейронов предыдущего слоя. Функциональный сигнал на выходе рассматриваемого *j*-го нейрона равен

$$y_j(n)=\sigma_j(v_j(n)) \quad . \tag{2.5}$$

Алгоритм обратного распространения состоит в применении к синаптическому весу $w_{ji}(n)$ коррекции $\Delta w_{ji}(n)$ пропорционально частной производной $\partial E(n)/\partial w_{ji}(n)$. Для вычисления компонента градиента ошибки можно прибегнуть к цепному правилу, т.е.

$$\frac{\partial E(n)}{\partial w_{ji}(n)}=\frac{\partial E(n)}{\partial e_j(n)}\ \frac{\partial e_j(n)}{\partial y_j(n)}\ \frac{\partial y_j(n)}{\partial v_j(n)}\ \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad . \tag{2.6}$$

Здесь следует отметить, что все обозначения в (2.6) относятся не ко всей сети в целом, а только к окружению *j*-го нейрона. Дифференцирование обоих частей уравнения (2.2) по $e_j(n)$ дает

$$\frac{\partial E(n)}{\partial e_j(n)} \ = \ e_j(n) \quad . \tag{2.7}$$

Аналогично можно получить из (2.1), (2.5) и (2.4) следующие выражения:

$$\frac{\partial e_j(n)}{\partial y_j(n)} \ = \ -1 \quad , \tag{2.8}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} \ = \ \sigma_j{}'(v_j(n)) \quad , \tag{2.9}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} \ = \ y_i(n) \quad . \tag{2.10}$$

Подставляя все эти выражения в (2.6), окончательно получим

$$\frac{\partial E(n)}{\partial w_{ji}(n)} \ = \ -e_j(n)\,\sigma_j{}'(v_j(n))\,y_i(n) \quad . \tag{2.11}$$

Коррекция $\Delta w_{ji}(n)$, применяемая к весу $w_{ji}(n)$, определяется согласно *дельта-правилу*

$$\Delta w_{ji}(n)=-\eta\,\frac{\partial E(n)}{\partial w_{ji}(n)} \quad , \tag{2.12}$$

где η - *параметр скорости обучения* алгоритма обратного распространения. Знак минус

отражает тот факт, что направление минимума целевой функции (2.2) ищется в направлении противоположному градиенту в текущей точке гиперповерхности целевой функции в пространстве весов. Подставляя (2.11) в (2.12), получим:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad , \tag{2.13}$$

где величина $\delta_j(n)$ называется обычно *локальным градиентом* и определяется выражением

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n) \sigma_j'(v_j(n)) \quad . \tag{2.14}$$

Из этих выражений видно, что для вычисления величины коррекции необходимо знать сигнал ошибки $e_j(n)$. Если речь идет о выходном линейном нейроне, то получить сигнал ошибки не представляет трудности поскольку для него известен желаемый отклик и вычислить локальный градиент просто с помощью (2.1) и (2.14). Для выходного линейного нейрона $\sigma(v)=v$ и соответственно $\sigma'=1$. Однако, если нейрон находится в скрытом слое, то желаемый отклик для него неизвестен. Сигнал ошибки для скрытого нейрона должен рекурсивно вычисляться на основе ошибки всех нейронов, с которыми он связан. Локальный градиент для скрытого нейрона можно вычислить следующим образом из (2.14):

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \sigma_j'(v_j(n)) \quad . \tag{2.15}$$

Чтобы не запутаться в индексах суммирования при вычислении частной производной $\partial E(n)/\partial y_j(n)$, запишем

$$E(n) = \frac{1}{2} \sum_{k=1}^{N_3} e_k^2(n) \quad , \tag{2.16}$$

где суммирование проводится по нейронам выходного слоя. Дифференцируя (2.16) по функциональному сигналу $y_j(n)$, получим:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \quad . \tag{2.17}$$

Используя цепное правило для производной $\partial e_k/\partial y_j$ можно переписать (2.17) в форме:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \quad . \tag{2.18}$$

Поскольку для *k*-го выходного нейрона

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \sigma_k(v_k(n)) \quad , \tag{2.19}$$

то

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\sigma_k'(v_k(n)) \quad . \tag{2.20}$$

Здесь несмотря на то, что $\sigma(v) = v$ и $\sigma' = 1$ для выходного линейного нейрона дальнейшее упрощение выражений не производится, чтобы не потерять общность рассуждений для нейрона в любом слое сети. *Индуцированное локальное поле* *k*-го нейрона составляет

$$v_k(n) = \sum_{k=0}^{N_2} w_{kj}(n) y_j(n) \quad , \tag{2.21}$$

где предполагается, что $y_0(n) = 1$. Дифференцирование этого выражения дает:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \quad . \tag{2.22}$$

Подставляя (2.22) и (2.20) в (2.18), получим соотношение

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n) \sigma_k'(v_k(n)) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \quad . \tag{2.23}$$

И наконец, подставляя (2.23) в (2.15), получим формулу обратного распространения для локального градиента $\delta_j(n)$ *j*-го скрытого нейрона:

$$\delta_j(n) = \sigma_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad . \tag{2.24}$$

Теперь, сводя воедино все соотношения, можно определить дельта правило реализующее алгоритм обратного распространения ошибки следующим образом:

Коррекция веса ($\Delta w_{ji}(n)$) = параметр скорости обучения (η) *
локальный градиент ($\delta_j(n)$) * (2.25)
входной сигнал j-го нейрона ($y_j(n)$).

При этом значение локального градиента $\delta_j(n)$ зависит от положения нейрона в сети.

1. Если *j*-ый нейрон - выходной, то градиент $\delta_j(n)$ равен произведению производной $\sigma_j'(v_j(n))$ на сигнал ошибки $e_j(n)$ для нейрона *j* (см. выражение (2.14)).
2. Если *j*-ый нейрон – скрытый, то градиент $\delta_j(n)$ равен произведению производной $\sigma_j'(v_j(n))$ на взвешенную сумму градиентов, вычисленных для нейронов следующего слоя, которые непосредственно связаны с данным нейроном (см. (2.24)).

Для выходных линейных нейронов п. 1 означает равенство локального градиента собственно сигналу ошибки. Однако нейронная сеть может быть построена так, что нелинейные нейроны последнего слоя являются выходными.

Итак, на каждой итерации алгоритма обратного распространения выполняются два прохода: прямой

$$y_j(n)=\sigma(v_j(n)) \quad , \tag{2.26}$$

где

$$v_j(n)=\sum_i w_{ji}(n)y_i(n) \quad . \tag{2.27}$$

Если нейрон находится в первом скрытом слое, то

$$y_i(n)=x_i(n) \quad . \tag{2.28}$$

А если является выходным, то

$$y_j(n)=o_j(n) \quad , \tag{2.29}$$

где $o_j(n)$ значение на выходном «терминале» сети. Прямая фаза обучения начинается с представления на входе очередного $\boldsymbol{x_n}$, а завершается вычислением сигнала ошибки на выходе, т.е. $\boldsymbol{e_n=d_n-o_n}$. Обратный проход начинается с выходного слоя с предъявления ему сигнала ошибки, который передается справа налево с параллельным вычислением локального градиента. Осуществляется продвижение справа налево с коррекцией весов, вычислением локальных градиентов, которые в свою очередь используются для вычисления локальных градиентов предыдущего (или следующего на пути обратного распространения ошибки) слоя нейронов.

Функция активации

Единственное требование на функцию активации накладываемое алгоритмом обратного распространения ошибки это ее дифференцируемость. Примером такой функции является логистическая функция

$$\sigma_j(v_j(n))=\frac{1}{1+\exp(-av_j(n))} \quad . \tag{2.30}$$

Ясно, что ее область значений принадлежит отрезку [0,1]. Производная этой функции:

$$\sigma_j' = \frac{a\exp(-av_j(n))}{[1+\exp(-av_j(n))]^2} = a\sigma_j(v_j(n))(1-\sigma_j(v_j(n))) = ay_j(n)(1-y_j(n)) \tag{2.31}$$

То, что производная функции активации может быть выражена через значения самой функции активации означает, что при обратном проходе алгоритма обратного распространения можно воспользоваться значениями функции уже вычисленными при прямом проходе не вычисляя производные функции непосредственно.

Следует обратить внимание на то обстоятельство, что (2.31) достигает своего максимума при $y_j=0.5$, а минимума на краях интервала [0,1]. Поскольку $\Delta w_{ji} \sim \sigma_j'$, то для максимального изменения синаптических весов функциональные сигналы нейрона должны находиться в середине диапазона. Именно это свойство обуславливает устойчивость алгоритма обратного распространения.

Другой популярной функцией активации является гиперболический тангенс, в общем виде представляемый выражением

$$\sigma_j(v_j(n))=a\tanh(bv_j(n)), \quad a>0,\ b>0 \quad , \tag{2.32}$$

производная которого так же может быть выражена через саму функцию.

Скорость обучения

Алгоритм обратного распространения обеспечивает построение в пространстве весов некоторой аппроксимации траектории, вычисляемой методом наискорейшего спуска. Чем меньше параметр обучения η , тем меньше корректировка синаптических связей на каждой итерации и тем более гладкой является траектория в пространстве весов. Однако это приводит к замедлению процесса обучения, слишком же большой параметр обучения может привести систему в неустойчивое положение и минимизация средней ошибки так и не произойдет. Простейшим способом повышения скорости обучения без потери устойчивости является следующее изменение дельта-правила (2.13) за счет добавления к нему «момента инерции»:

$$\Delta w_{ji}(n) = \alpha w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \quad , \tag{2.33}$$

где α - величина называемая *постоянной момента* (как правило положительная). Выражение (2.33) называется *обобщенным дельта-правилом*. С точки зрения теории оптимизации алгоритм обратного распространения может рассматриваться как специфическая реализация градиентного метода минимизации целевой функции (2.3). А обобщенное дельта-правило может рассматриваться как метод регуляризации решения задачи минимизации.