Нейронные сети. Краткий курс

Лекция 3

Последовательный и пакетный режимы обучения многослойного персептрона

В практических реализациях алгоритма обратного распространения в процессе обучения многослойного персептрона ему многократно предъявляется заранее определенное множество обучающих примеров. Один полный цикл предъявления всех обучающих примеров называется *эпохой*. Обучение проводиться эпоха за эпохой пока синаптические веса не стабилизируются, а среднеквадратичные ошибки на всем обучающем множестве не сойдутся к некоторому минимальному значению. Целесообразно в каждой эпохе случайным образом перемешивать учебные примеры, это предотвращает потенциальную возможность появления замкнутых циклов в процессе эволюции синаптических весов.

Последовательный режим обучения по методу обратного распространения иногда называют *стохастическим* или *интерактивным* (on-line). В этом режиме корректировка весов проводится после подачи на вход каждого учебного примера. Именно для этого режима и был рассмотрен алгоритм обратного распространения в предыдущей лекции.

Пакетный режим предполагает корректировку весов только после подачи в сеть всех примеров обучающего множества. Целевой функцией в этом случае является среднеквадратичная ошибка (выражение (2.3) в предыдущей лекции):

$$\bar{E} = \frac{1}{2\mathrm{N}} \sum_{n=1}^{N} \sum_{j=1}^{N_3} e_j^2(n) \quad . \tag{3.1}$$

Дельта-правило в пакетном режиме определяется следующим выражением:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \bar{E}}{\partial w_{ji}} = -\frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}} \quad . \tag{3.2}$$

Здесь используются те же соглашения об индексах, что и в предыдущей лекции. Для вычисления производной $\partial e_j(n) / \partial w_{ji}$ нужно проделать то же, что и при последовательном режиме, однако корректировка веса Δw_{ji} выполняется только после прохождения по сети всего множества примеров. Это означает, что до корректировки веса необходимо хранить больше данных чем для последовательного режима.

Оба режима характеризуются своими преимуществами и недостатками. Последовательный режим является стохастическим (особенно при случайном перемешивании

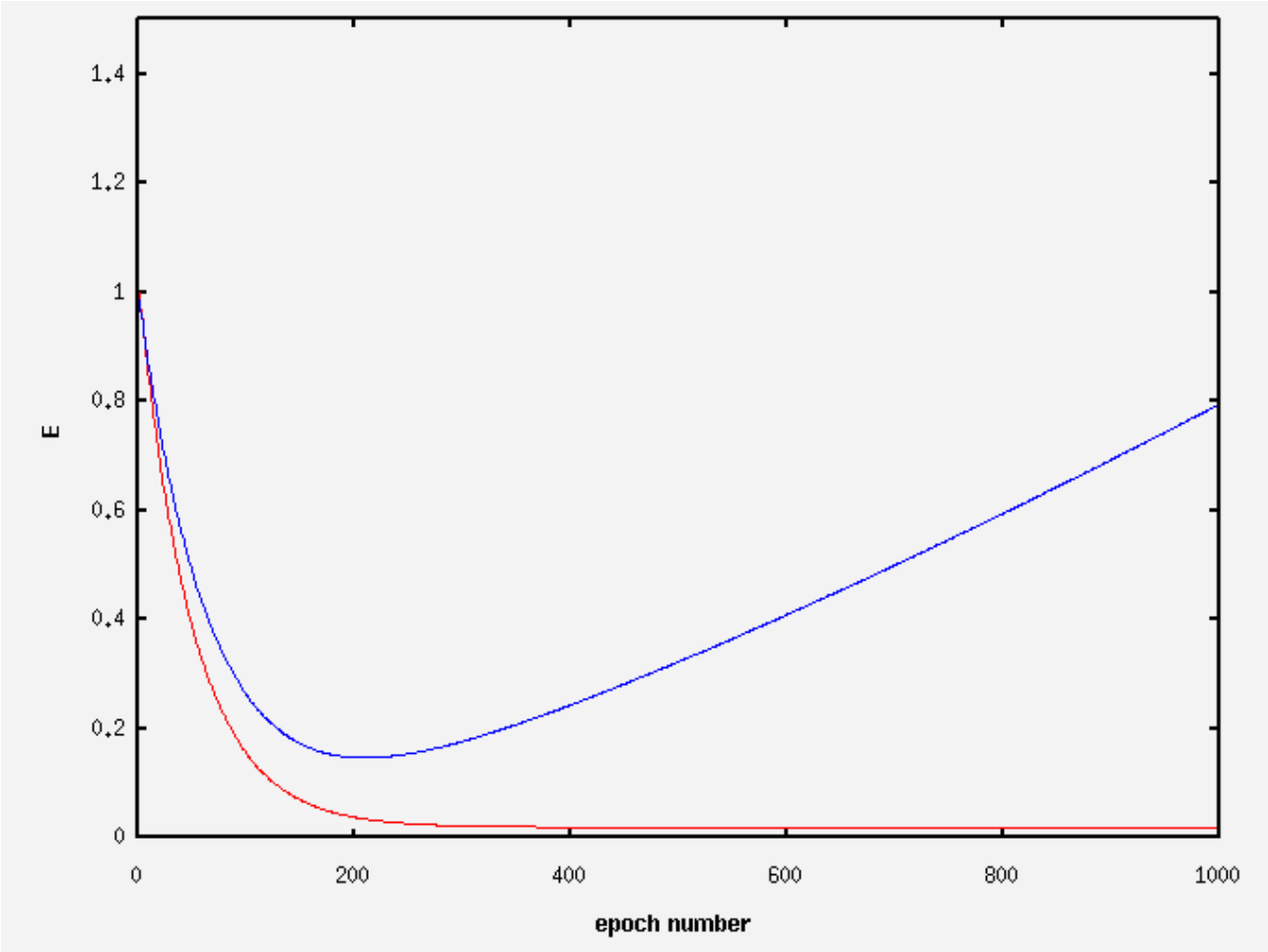
учебных примеров от эпохи к эпохе) и это сокращает до минимума возможность остановки в точке какого-либо локального минимума. Это однако усложняет построение теоретического фундамента для нахождения условий сходимости алгоритма. Однако более экономное расходование памяти компьютера в последовательном режиме позволяет иметь дело со сколь угодно большими учебными наборами. Пакетный же режим позволяет получить более точную оценку вектора градиента и к услугам этого режима все методы разработанные в теории оптимизации (речь идет о минимизации (3.1)), для которых гарантирована сходимость к локальному минимуму при довольно простых условиях. Пакетный режим приводит к решению гораздо быстрее чем последовательный, однако качество сети полученное при применении последовательного режима может быть выше. Последовательный режим остается популярным в силу простоты реализации и способности решать очень сложные задачи.

Критерий останова

Поскольку не существует доказательства сходимости алгоритма обратного распространения в общем случае, то используют несколько более ими менее обоснованных критериев для прекращения корректировки весов в процессе обучения нейронной сети. Если обозначить символом $\boldsymbol{w}^*$ вектор весов, обеспечивающий минимум поверхности ошибок (3.1), то вектор градиента $\nabla \bar{E}(\boldsymbol{w})$ в этой точке равен нулевому. Следовательно, разумный критерий останова можно сформулировать так: *считать, что алгоритм обратного распространения сошелся, если Евклидова норма вектора градиента достигает достаточно малого значения*. Недостатком этого критерия можно считать необходимость вычислять градиент $\nabla \bar{E}(\boldsymbol{w})$. Другим свойством минимума является то, что целевая функция $\bar{E}$ стабилизируется и мало изменяется от эпохи к эпохе. Критерий останова процесса обучения можно сформулировать так: *считать, что алгоритм обратного распространения сошелся, если наблюдается достаточно малая интенсивность изменений среднеквадратичной ошибки в течение эпохи*.

Наиболее распространенный критерий основан на проверке эффективности обобщения обучаемой сетью. Обычно среднеквадратичная ошибка уменьшается по мере увеличения количества эпох обучения. В начале обучения она стремительно падает, однако затем продолжает убывать все медленнее и медленнее по мере продвижения сети к локальному минимуму на на поверхности ошибок. По виду этой кривой сложно определить наилучший момент. Если вовремя не остановить процесс обучения, то повышается вероятность перетренировки сети. Перетренированная сеть хорошо аппроксимирует зависимость для векторов учебного набора, однако демонстрирует достаточно высокую ошибку для векторов, которые изначально не входили в учебный набор. Наступление стадии излишнего переобучения (перетренировки) можно определить с помощью перекрестной проверки, в которой данные из учебного набора разбиты на множество оценивания и проверки. Или говорят, что имеется учебный и тестовый наборы данных. Множество оценивания используется для обычной тренировки сети, а множество проверки только для определения среднеквадратичной ошибки. Проверка осуществляется каждые несколько эпох. На рисунке ниже схематически показано возможное поведение среднеквадратической ошибки для

учебного (красная кривая) и тестового (синяя кривая) наборов.



На проверочном множестве обычно нейронносетевая модель ведет себя не так хорошо как на множестве оценивания. В качестве точки останова используется точка минимума на кривой ошибки построенной на проверочном множестве (тестовом наборе).

Обучение как задача оптимизации

Поиск минимума (3.1) это типичная задача оптимизации. Численная оптимизация это большой раздел прикладной математики, в котором разработано большое количество методов пригодных для тренировки сети. Мы кратко рассмотрим основные из них. Поверхность ошибок многослойного персептрона является в высшей степени нелинейной функцией от вектора синаптических весов $\boldsymbol{w}$. Если в предыдущей лекции на схеме персептрона с двумя скрытыми слоями веса были сгруппированы в матрицы $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ и $\boldsymbol{W}_3$, то здесь мы будем рассматривать веса сгруппированные в один единственный вектор $\boldsymbol{w}$, имеющий для того персептрона размерность равную $(m+1)N_1+(N_1+1)N_2+(N_2+1)N_3$, то есть числу всех синаптических связей. Целевая функция (3.1) может быть разложена в ряд Тейлора в окрестности текущей точки $\boldsymbol{w}(n)$ на поверхности ошибок

$$\bar{E}(\boldsymbol{w}(n)+\Delta\boldsymbol{w}(n)) \;=\; \bar{E}(\boldsymbol{w}(n))+\boldsymbol{g}^{T}(n)\,\Delta\boldsymbol{w}(n)+\frac{1}{2}\Delta\boldsymbol{w}^{T}(n)\,\boldsymbol{H}(n)\,\Delta\boldsymbol{w}(n)+O(\|\boldsymbol{w}\|^{3}) \quad , (3.3)$$

где $\boldsymbol{g}(n)=\nabla\bar{E}(\boldsymbol{w}(n))$ - вектор градиента, а $\boldsymbol{H}(n)$ - матрица Гессиана функции $\bar{E}(\boldsymbol{w})$ в точке $\boldsymbol{w}(n)$.

В самом простом градиентном методе, которым и является алгоритм обратного распространения в пакетном режиме, корректировка применяемая к вектору синаптических весов, определяется выражением

$$\Delta\boldsymbol{w}(n)=-\eta\,\boldsymbol{g}(n) \quad , \tag{3.4}$$

где η - параметр скорости обучения. Итерационную формулу метода можно записать следующим образом:

$$\boldsymbol{w}(n+1) \;=\; \boldsymbol{w}(n)-\eta\,\boldsymbol{g}(n) \tag{3.5}$$

В теории численной оптимизации этот метод называется градиентным с постоянным шагом. В его модификации называемой *методом наискорейшего спуска* параметр η выбирается на каждой итерации таким образом, чтобы минимизировать функцию

$$f(\eta)=\bar{E}(\boldsymbol{w}(n)-\eta\,\boldsymbol{g}(n)) \quad , \tag{3.6}$$

то есть на каждой итерации еще выполняется одномерная минимизация. В градиентном методе ряд Тейлора (3.3) ограничивается линейным членом, такое ограничение обеспечивает простоту реализации, однако приводит к низкой скорости сходимости, особенно в случае так называемых «овражных» функций. Существенно улучшить скорость сходимости можно если привлечь квадратичную аппроксимацию. Это реализовано в известном *методе Ньютона*, итерационная формула которого

$$\boldsymbol{w}(n+1)=\boldsymbol{w}(n)+\boldsymbol{H}^{-1}(n)\boldsymbol{g}(n) \quad , \tag{3.7}$$

где $\boldsymbol{H}^{-1}(n)$ - матрица обратная к Гессиану, при условии, что таковая существует. Особенность этого метода заключается в том, что если целевая функция является строго квадратичной, то есть последнее слагаемое в (3.3) равно нулю, то метод сходится к точному минимуму за одну итерацию. На практике однако применение метода Ньютона усложняется следующими факторами.

1. Требуется вычисление матрицы, обратной Гессиану, что означает огромный объем вычислений, особенно для сетей с большим количеством связей.
2. Чтобы иметь возможность вычислить $\boldsymbol{H}^{-1}(n)$, нужно чтобы матрица $\boldsymbol{H}(n)$ была несингулярной и, с точки зрения вычислительной реализации, по возможности хорошо обусловленной.
3. Если функция $\bar{E}(\boldsymbol{w})$ не является квадратичной, то сходимость метода Ньютона не гарантирована.

Для преодоления этих проблем обычно используют методы называемые *квазиньютоновскими*, которые используют только оценки вектора градиента $\boldsymbol{g}(n)$, а вместо обращения матрицы Гессиана, напрямую строятся положительно определенные оценки матрицы $\boldsymbol{H}^{-1}(n)$. С вычислительной точки зрения в применении к нейронным сетям, квазиньютоновские методы остаются непрактичными за исключением нейронных сетей малой размерности.

Наиболее пригодным для решения задач большой размерности, каковыми являются задачи обучения нейронных сетей, является *метод сопряженных градиентов*. Этот метод можно рассматривать как нечто промежуточное между методом наискорейшего спуска и методом Ньютона. Существует большое количество литературы посвященной данному методу, поэтому в данном курсе кратко рассмотрим один из вариантов метода сопряженных градиентов, называемый *методом Флетчера-Ривза*. Обозначим как $\boldsymbol{d}(n)$ - вектор направления перемещения в пространстве весов (не путать с желаемым выходным вектором). Тогда алгоритм Флетчера-Ривза можно представить как следующую последовательность действий:

a) Этап 0: пусть $\boldsymbol{w}(0)$ выбранная начальная точка в пространстве весов; положить

$$\boldsymbol{d}(0)=-\boldsymbol{g}(\boldsymbol{w}(0)) \quad . \tag{3.8}$$

b) Этап *n*: выбрать $\eta(n)$ минимизирующим функцию

$$f(\eta) \;=\; \bar{E}(\boldsymbol{w}(n)+\eta\,\boldsymbol{d}(n)) \quad , \tag{3.9}$$

положить

$$\boldsymbol{w}(n+1) \;=\; \boldsymbol{w}(n)+\eta(n)\boldsymbol{d}(n) \quad , \tag{3.10}$$

$$\boldsymbol{d}(n+1) \;=\; -\boldsymbol{g}(\boldsymbol{w}(n+1))+\beta(n)\boldsymbol{d}(n) \quad , \tag{3.11}$$

где

$$\beta(n)=\frac{\|\boldsymbol{g}(\boldsymbol{w}(n+1))\|^2}{\|\boldsymbol{g}(\boldsymbol{w}(n))\|^2} \quad . \tag{3.12}$$

c) Тест на остановку; если выполнен, то конец. Иначе: положить $n \;\leftarrow\; n+1$ и вернуться к пункту b).

Для одномерной минимизации (3.9) можно использовать метод квадратичной интерполяции, сущность которой заключается в следующем. Если $\eta_1 \le \eta_2 \le \eta_3$ - три значения η, для которых $f(\eta_1) \ge f(\eta_2)$, $f(\eta_3) \ge f(\eta_2)$, то приближаем функцию $f(\eta)$ на интервале $[\eta_1, \eta_3]$ квадратичной функцией (параболой), имеющей те же значения, что и f, в точках η_1, η_2 и η_3. Уравнение имеет вид

$$\phi(\eta) = \sum_{i=1}^{3} f(\eta_i) \frac{\prod_{j\neq i}(\eta-\eta_j)}{\prod_{j\neq i}(\eta_i-\eta_j)} \quad . \tag{3.13}$$

Минимум функции $\phi(\eta)$ достигается в точке

$$\eta_4 = \frac{1}{2}\,\frac{r_{23}f(\eta_1)+r_{31}f(\eta_2)+r_{12}f(\eta_3)}{s_{23}f(\eta_1)+s_{31}f(\eta_2)+s_{12}f(\eta_3)} \quad ,$$

где $r_{ij}=\eta_i^2-\eta_j^2$, $s_{ij}=\eta_i-\eta_j$. При этом $\eta_4\in[\eta_1,\eta_3]$. Точка η_4 берется в качестве приближения точки минимума функции $f(\eta)$ на интервале $[\eta_1,\eta_3]$. Далее построение повторяется с тремя новыми точками

$$\begin{aligned}
(\eta_1',\eta_2',\eta_3') &= (\eta_2,\eta_4,\eta_3) \text{ , если } \eta_2\le\eta_4\le\eta_3 \text{ , } f(\eta_4)\le f(\eta_2) \text{ ,} \\
&= (\eta_1,\eta_2,\eta_4) \text{ , если } \eta_2\le\eta_4\le\eta_3 \text{ , } f(\eta_4)> f(\eta_2) \text{ ,} \\
&= (\eta_1,\eta_4,\eta_2) \text{ , если } \eta_1\le\eta_4\le\eta_2 \text{ , } f(\eta_4)\le f(\eta_2) \text{ ,} \\
&= (\eta_4,\eta_2,\eta_3) \text{ , если } \eta_1\le\eta_4\le\eta_2 \text{ , } f(\eta_4)> f(\eta_2) \text{ .}
\end{aligned} \tag{3.14}$$

Для того, чтобы запустить этот процесс надо найти три значения $a<b<c$, такие что $b-a=c-b=\Delta$, $f(b)\le f(a)$ и $f(b)\le f(c)$. Для нахождения таких точек можно воспользоваться следующей процедурой.

Зададим начальную точку η^0 , произвольный шаг перемещения δ , затем вычислим $f(\eta^0)$ и $f(\eta^1)=f(\eta^0+\delta)$. Могут представится два случая.

Случай 1: $f(\eta^0+\delta)\le f(\eta^0)$.

Тогда вычисляем последовательность $f(\eta^2),f(\eta^3),\dots,f(\eta^p)$ для значений $\eta^2=\eta^1+2\delta,\ \eta^3=\eta^2+4\delta,\dots,\eta^p=\eta^{p-1}+2^{p-1}\delta$, пока значения функции убывают. Останавливаемся на шаге p , как только функция снова начинает убывать, и, значит имеем

$$f(\eta^0)\ge f(\eta^1)>\dots>f(\eta^{p-1})<f(\eta^p) \quad . \tag{3.15}$$

После этого вычисляем $f(\eta^{p+1})$ для $p+1$ -ой точки $\eta^{p+1}=\eta^p-2^{p-2}\delta$. При таком построении четыре точки η^{p-2}, η^{p-1}, η^{p+1}, η^p будут равноотстоящими с $\Delta=2^{p-1}\delta$. Выбираем из них искомые три точки a,b,c , исключая либо η^p (если $f(\eta^{p-1})<f(\eta^{p+1})$), либо η^{p-2} (если $f(\eta^{p+1})<f(\eta^{p-1})$).

Случай 2: $f(\eta^0+\delta)>f(\eta^0)$. В этом случае шаг δ заменяется на противоположный $-\delta$ и таким образом возвращаемся к случаю 1, заметив что новое $f(\eta^1)$ равно старому $f(\eta^0)$ и нет нужды пересчитывать это значение.

Эта процедура сходится очень быстро даже при неудачном выборе слишком малого δ .