КОМПЬЮТЕРНЫЕ И ИНФОРМАЦИОННЫЕ СЕТИ И СИСТЕМЫ АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВА

УДК 004.032.26

Т.В. Гройсман, специалист,
Е.А. Арсирий, канд. техн. наук, доц.,
Одес. нац. политехн. ун-т

СРАВНИТЕЛЬНЫЙ АНАЛИЗ КАЧЕСТВА АЛГОРИТМОВ ОПТИМИЗАЦИИ НА ОСНОВЕ ОБРАТНОГО РАСПРОСТРАНЕНИЯ

Т.В. Гройсман, О.О. Арсірій. **Порівняльний аналіз якості алгоритмів оптимізації на основі зворотнього поширення.** Приводиться залежність середньоквадратичної помилки від кількості циклів і параметра швидкості настроювання навчання для алгоритмів обчислення похідних апроксимуючих функцій першого порядку. Розглянуто також інші оцінки якості навчання, такі як середня абсолютна помилка, сума квадратів помилок і комбінований критерій якості. Моделювання багатошарового персептрона виконувалося в середовищі Matlab для тестової задачі XOR.

Т.В. Гройсман, Е.А. Арсирий. **Сравнительный анализ качества алгоритмов оптимизации на основе обратного распространения.** Приводится зависимость среднеквадратической ошибки от количества циклов и параметра скорости настройки обучения для алгоритмов вычисления производных аппроксимирующих функций первого порядка. Рассмотрены также другие оценки качества обучения, такие как средняя абсолютная ошибка, сумма квадратов ошибок и комбинированный критерий качества. Моделирование многослойного персептрона выполнялось в среде Matlab для тестовой задачи XOR.

T.V. Groisman, E.A. Arsiry. **Comparative analysis of the quality of optimization algorithms based on back propagation.** The dependence of the mean square error upon the number of cycles and adjustment settings for learning methods for calculating the derivatives of approximating functions of the first order is presented. Other assessment of the quality of learning, such as the average absolute error, the sum of squared errors and the combined criterion of quality are considered as well. Modeling of multilayer perceptron was implemented in the Matlab environment for the test problem XOR.

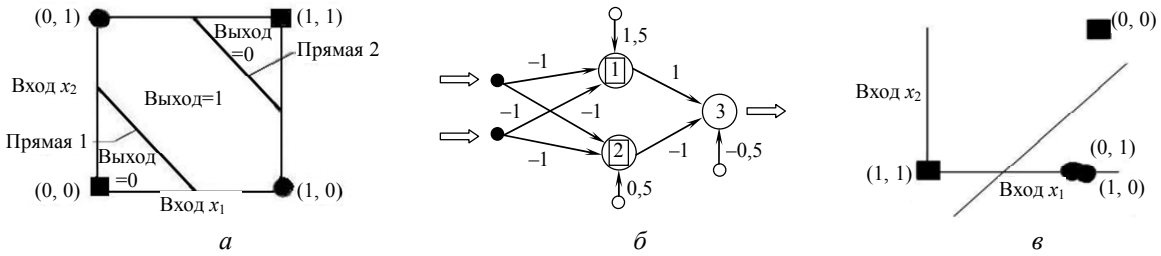
Решение задачи классификации образцов, являющееся одним из важнейших применений нейронных сетей (НС), напрямую связано с решением задачи их обучения. Модель обучения НС на основе коррекции ошибок одна из основных. Линейные адаптивные фильтры и персептрон Розенблатта (Simple layer perceptron — SLP), как примеры однослойных НС прямого распространения, по-прежнему играют роль полигона для решения задач обучения [1]. При построении прямых, разделяющих два класса линейно-разделимых образцов средствами библиотеки Neural Networks Tool (NNTools), показано различие процедур последовательного режима

обучения (адаптации) SLP и пакетного обучения линейных НС по алгоритму LMS (Least Mean Square — LMS). Приведены количественные характеристики: количество шагов адаптации и циклов обучения в зависимости от значения критерия качества обучения [2].

Для обучения многослойного персептрона (multilayer perceptron — MLP) — многослойной НС прямого распространения используется алгоритм обратного распространения ошибки, который рассматривается как обобщение алгоритма LMS и упрощенно называется алгоритмом обратного распространения. Этим алгоритмом реализуется метод обучения на основе обратного распространения (back-propagation learning — BPL), являющийся специфической реализацией градиентного спуска в пространстве настраиваемых параметров MLP — весовых коэффициентов и смещений. Условием реализации метода BPL является то, что каждый нейрон MLP должен иметь нелинейную, всюду дифференцируемую функцию активации, в отличие от жесткой пороговой функции, используемой в SLP. Применяются нелинейные сигмоидальные функции активации типа логистической или гиперболического тангенса, относящиеся к классу функций со сжимающим отображением, для реализации которых в NNTools используются M-функции *logsig* и *tansig* [3].

Основная идея реализации метода BPL для MLP заключается в эффективном вычислении частных производных функции сети $F(\mathbf{x}, \mathbf{p})$ по всем элементам вектора настраиваемых параметров $\mathbf{x}$ для заданного входного вектора $\mathbf{p}$, поэтому BPL может быть реализован различными алгоритмами оптимизации. Задача выбора алгоритма оптимизации BPL весьма актуальна при проектировании НС для решения практических задач классификации образцов, т.к. от этого зависит скорость обучения. Решение такой задачи, в свою очередь, зависит от внутренних факторов НС, таких как: архитектура MLP — число скрытых слоев, число нейронов в каждом скрытом слое, вид функции активации, способ предъявления векторов обучающей выборки, начальная инициализация вектора настраиваемых параметров, а также внешних факторов — входных и выходных векторов. Представлен сравнительный анализ качества алгоритмов оптимизации BPL для MLP с обучающей выборкой, состоящей из четырех входных векторов, принадлежащих к двум линейно-неразделимым классам (exclusive OR problem — XOR). Такой анализ позволит выявить в условиях максимального уменьшения влияния внешних и внутренних факторов, насколько рассматриваемые алгоритмы оптимизации позволяют увеличить скорость BPL.

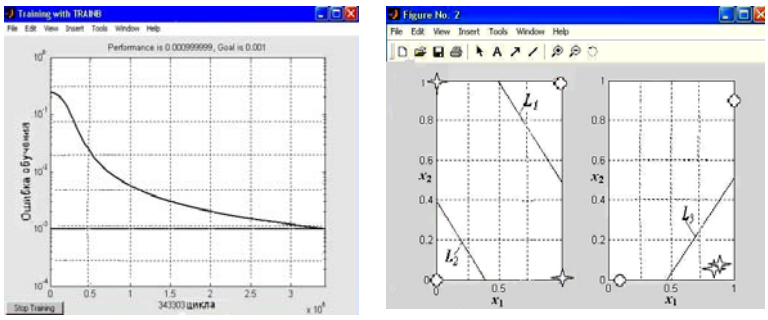
Рассматривается задача XOR, как частный случай более общей задачи классификации точек единичного гиперкуба, каждая точка которого принадлежит классу — 0 или 1 [4]. Представлена иллюстрация проблемы моделирования отношения XOR разделяющими прямыми границами для SLP, на которой четыре угла единичного квадрата соответствуют входным парам (0,0), (0,1), (1,0), (1,1); входные пары (0,0) и (1,1) располагаются в противоположных углах единичного квадрата, но генерируют одинаковый выходной сигнал, равный нулю, т.е. принадлежат классу 0; пары (0,1) и (1,0) принадлежат классу 1 (рис. 1, *а*). Архитектура MLP содержит два нейрона — элемента ввода данных в сеть — в скрытом слое и один нейрон в выходном слое (рис. 1, *б*). На нейроны 1 и 2 скрытого слоя поступают по два значения векторов обучающей выборки, чтобы представить две разделяющие прямые, а нейрон 3 выходного слоя объединяет информацию об этих двух прямых (см. рисунок 1, *а*, *в*, соответственно). Указаны весовые коэффициенты, для которых получаются разделяющие прямые (см. рисунок 1, *б*, *а*, *в*, соответственно).



а *б* *в*

Рис. 1. Границы решений для скрытого (а) и выходного слоя (в), архитектура НС (б)

Моделирование MLP для решения задачи XOR выполнено в NNTools среды Matlab с помощью М-функции *net* = *newff*([0 1;0 1],[2 1],{'*logsig*','*logsig*'}) [3]. Функцией активации в скрытом и выходном слоях является *logsig*. Согласно условиям задачи (см. рисунок 1, *а*) определена последовательность двухэлементных векторов входа **P** = [1 1; 1 0; 0 1; 0 0] и целей **T** = [0 1 1 0]. В результате процедуры начальной инициализации значения весовых коэффициентов и смещений (см. рисунок 1, *б*) скрытого слоя хранятся в свойствах *net.IW*{1,1} и *net.b*{1}, выходного слоя — *net.LW*{2,1} и *net.b*{2}. BPL для MLP выполнено с помощью М-функции *net* = *train*(*net*,**P'**,**T**), которая устанавливает процедуру обучения сети и настраивает ее параметры, присваивая свойствам *net.trainFcn* и *net.trainParam* [3] требуемые значения. Процедура *trainb* реализует BPL с последовательным предъявлением векторов обучающей выборки. Исследованы процедуры циклического *trainc* и случайного *trainr* предъявления векторов [3]. Установлены параметры процедуры *trainb*, полученные экспериментальным путем: максимальное количество циклов обучения 500000; предельное значение критерия обучения 0,001; интервал вывода информации 100 циклов. Для обучения MLP с помощью процедуры *trainb* потребовалось 343303 цикла обучения; по результатам моделирования для нейронов 1 и 2 скрытого слоя определены разделяющие прямые L_1: $-4{,}6x_1 - 4{,}6x_2 + 6{,}9 = 0$ и L_2: $-5{,}7x_1 - 5{,}7x_2 + 2{,}2 = 0$, где коэффициенты перед x_1 и x_2 — значения вектора весовых коэффициентов, свободные члены — значения смещения; после прохождения скрытого слоя исходная точка (0,1) переместилась в (1,0), точки (0,0) и (1,1) поменялись местами (рис. 2, *а*, *б*, *в*, соответственно). Таким образом, данные ввода в выходном слое стали линейно отделимы с помощью прямой L_3: $8{,}6\,x_1 - 8{,}9\,x_2 - 4 = 0$.



а *б* *в*

Рис. 2. Моделирование MLP для решения задачи XOR в NNTools: график зависимости среднеквадратической ошибки от количества циклов обучения (а); границы решений для скрытого (б) и выходного (в) слоя

По результатам эксперимента можно сделать практические выводы: использование BPL с последовательным предъявлением MLP векторов обучающей выборки размерности 2×4 для решения задачи XOR позволило получить точность обучения 0,001 за 343303 цикла. Использование процедур циклического и случайного предъявления векторов входной выборки совместно с BPL незначительно увеличивают скорость обучения НС. Такой подход не может быть реализован для решения практических задач, т.к. потребует увеличения размеров обучающей выборки и размеров сети, что приведет к увеличению времени обучения НС. Таким образом, необходимо применять BPL в сочетании с другими алгоритмами оптимизации процесса обучения.

Существующие алгоритмы оптимизации, использующие BPL, во многом аналогичны алгоритмам определения экстремума функции нескольких переменных и могут быть разделены на группы нулевого, первого и второго порядка [3]. В алгоритмах нулевого порядка используются только значения функционала ошибки в заданных точках. Алгоритмы первого и второго порядка, основанные на реализации идеи градиентного спуска, требуют знания первых и вторых производных функционала ошибки по настраиваемым параметрам. В процессе работы этих алгоритмов, как правило, возникает задача одномерного поиска минимума функционала ошибки вдоль заданного направления. Это могут быть направления антиградиента или сопряженные направления [3]. Определение самого быстрого обучающего алгоритма для решения практиче-

ских задач зависит от сложности задачи, числа входных признаков, размера обучающей выборки, числа настраиваемых параметров — слоев и нейронов в каждом слое, конечной ошибки обучения и т.д. Проведен сравнительный анализ качества градиентных алгоритмов обучения и алгоритмов сопряженных градиентов на основе метода BPL для задачи XOR.

Для вычисления функционалов качества обучения, зависящих от ошибок e_i НС, в NNTools предназначено несколько функций. Наиболее широко используется среднеквадратическая ошибка [3]

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i^2 , \tag{1}$$

где N — размер обучающей выборки.

Градиентные алгоритмы обучения первого порядка являются специфической реализацией наискорейшего спуска в пространстве весовых коэффициентов MLP и обеспечивают движение по поверхности функционала ошибки в направлении, противоположном вектору градиента. К ним относятся алгоритмы: градиентного спуска с параметром скорости обучения GD, для реализации которого в NNTools предназначена М-функция *traingd*; градиентного спуска с адаптацией параметра скорости обучения GDA — *traingda*; пороговый алгоритм с эвристической стратегией изменения приращения настраиваемых параметров *Rprop* — *trainrp*. Алгоритмы GD и GDA в сочетании с настраиваемым параметром возмущения *mc* определяют два алгоритма GDM — *traingdm* и GDX — *traingdx* [3]. Для расчета параметра *mc*, значение которого задается в NNTools свойством *net.trainParam.mc* [3], необходимо суммировать градиенты, полученные на каждом цикле обучения, что значительно замедляет работу алгоритма и требует дополнительной памяти. В алгоритме градиентного спуска единственным источником информации о поверхности ошибок является градиент $\mathbf{g}_k$. Такое ограничение обеспечивает простоту реализации, но приносит и низкую скорость сходимости алгоритма. Кроме того, на скорость сходимости влияет значение параметра скорости обучения α_k, задаваемого в NNTools свойством *net.trainParam.lr* [3], выбор которого вызывает определенную трудность, зависит от конкретной задачи и определяется опытным путем. Для ускорения процесса обучения нужно увеличивать его значение, однако одновременно с этим будет изменяться и степень близости среднеквадратической ошибки к минимуму.

Формально все алгоритмы градиентного спуска можно записать как

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k , \tag{2}$$

где $\mathbf{x}_k$ — вектор настраиваемых параметров;

α_k — положительная константа — параметр скорости обучения;

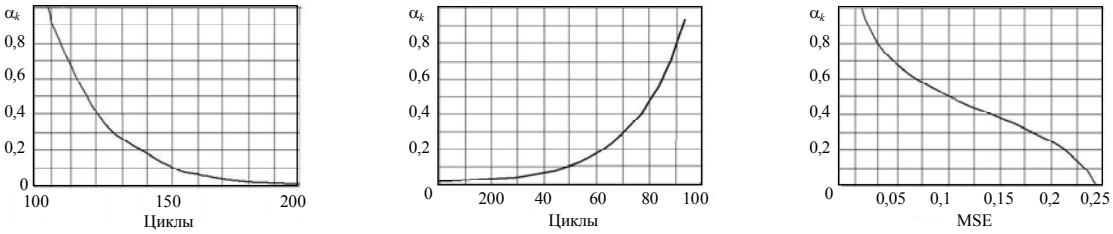
$\mathbf{g}_k$ — вектор градиента функционала ошибки, соответствующий k-й итерации.

Алгоритм GDM модифицирует вектор настраиваемых параметров с учетом параметра возмущения *mc*

$$\Delta\mathbf{x}_k = mc\Delta\mathbf{x}_k + (1-mc)\alpha\mathbf{g}_k .$$

Алгоритм GDA использует эвристическую стратегию изменения параметра α_k в процессе обучения. На каждом цикле обучения вычисляются значения настраиваемых параметров, выходов и ошибок. Новые значения сравниваются со значениями, полученными на предыдущем шаге. Если они меньше прежних, то параметр α_k увеличивается, и наоборот. В алгоритме *Rprop* значение приращения вектора настраиваемых параметров для каждого настраиваемого параметра увеличивается всякий раз, когда производная функционала ошибки по данному параметру сохраняет знак для двух последовательных итераций; и уменьшается, когда производная изменяет знак по сравнению с предыдущей итерацией. Если производная равна 0, то приращение остается неизменным. Для алгоритмов GD и GDM показано влияние параметра α_k на скорость обучения, позволяющее достичь значения MSE = 0,001 (рис. 3, *а*). В алгоритмах GDA и GDX возможна адаптивная настройка параметра скорости обучения α_k, которая приводит к его непрерывному увеличению в процессе обучения до достижения значения ≈ 0,93 (рис. 3, *б*). При α_k

= 0,9 необходимо ≈ 100 циклов обучения для обеспечения достаточной точности, т.е. близости попадания в минимум ошибки MSE ≈ 0,028 (рис. 3, *в*).



а *б* *в*

Рис. 3. Графики зависимости параметра скорости обучения от количества циклов обучения (а, б) и среднеквадратической ошибки (в)

Алгоритмы сопряженных градиентов: Флетчера-Ривса CGF — *traincgf*; Полака-Рибейры CGP — *traincgp*; Пауэлла-Биеле CGB — *traincgb*; Моллера SCG — *trainscg* [3]. Алгоритмы градиентного спуска корректируют настраиваемые параметры в направлении антиградиента, которое не всегда является самым благоприятным, чтобы за возможно малое число циклов обеспечить сходимость к минимуму функционала ошибки по настраиваемым параметрам. Сопряженные направления позволяют определить искомый минимум гораздо быстрее. Все алгоритмы данной группы на первой итерации начинают поиск в направлении антиградиента

$$\mathbf{p}_0 = -\mathbf{g}_0 .$$

Для определения размера шага вдоль сопряженного направления выполняются специальные одномерные процедуры поиска минимума, входящие в состав NNTools: одномерная минимизация методом золотого сечения — М-функция *srchgol*, метод Брента — *srchbre*, гибридный метод — *srchhyb*, метод Чараламбуса — *srchcha*, метод перебора с возвратами — *srchbac*. Направление следующего движения выбирается так, чтобы оно было сопряжено с предыдущим. Затем определяется следующее направление поиска как линейная комбинация нового направления наискорейшего спуска и вектора движения в сопряженном направлении

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} .$$

где $\mathbf{p}_k$ — направление движения;

$\mathbf{g}_k$ — градиент функционала ошибки;

β_k — коэффициент, соответствующий k-й итерации.

Когда направление спуска определено, то новое значение вектора настраиваемых параметров вычисляется по формуле (2). Алгоритмы второй группы различаются способом вычисления β_k:

— в алгоритме CGF $\beta_k = \dfrac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$;

— в алгоритме CGP $\beta_k = \dfrac{\Delta\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$.

Дорогостоящая в вычислительном отношении процедура одномерного поиска требует на каждой итерации несколько раз вычислять реакцию сети. Алгоритм SCG позволяет избежать излишних затрат. Для всех алгоритмов сопряженных градиентов, когда возникают проблемы со сходимостью, направление поиска периодически переустанавливается на направление антиградиента. Одна из таких стратегий рестарта, реализованная в алгоритме CGB, выполняется, если текущее и предшествующее направление градиентов слабоортогональны, т.е.

$$\left\| \mathbf{g}_{k-1}^T \mathbf{g}_k \right\| \geq 0,2 \left\| \mathbf{g}_k \right\|^2 .$$

В обучающих алгоритмах градиентного спуска управление сходимостью осуществляется с помощью параметра скорости обучения α_k, а в алгоритмах сопряженных градиентов размер шага корректируется на каждой итерации и поэтому они значительно превосходят по скорости обучения, однако для их реализации необходим значительный объем памяти.

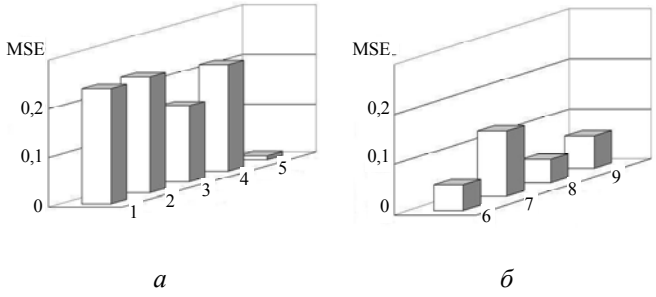
Для сравнительного анализа двух групп алгоритмов построены усредненные кривые обучения (рис. 4, *а, б*).

а *б*

Рис. 4. Усредненные кривые обучения для первой (а) и второй (б) группы алгоритмов: 1 — GD, 2 — GDM, 3 — GDA, 4 — GDX, 5 — Rprop, 6 — CGF, 7 — CGP, 8 — CGB, 9 — SCG

Кривая обучения является графиком изменения среднеквадратического значения ошибки обучения (1) в зависимости от количества циклов. Кривые обучения начинаются с весьма большого значения MSE (см. рисунок 4, *а, б*), зависящго от начальной инициализации настраиваемых параметров, которое затем уменьшается с некоторой скоростью, различной для каждого типа алгоритма, а в пределе сходится к некоторому устойчивому значению, которое мало отличается для различных алгоритмов. Это значит, что поверхность ошибок в задаче XOR достаточно гладкая. Основываясь на кривых обучения, определим скорость сходимости алгоритмов как количество циклов обучения, необходимых для уменьшения начального значения MSE до установленного предельно малого критерия качества обучения.

Алгоритмы GD и GDM показали низкую скорость обучения, что недопустимо для решения задачи XOR. Алгоритмы группового обучения GDA, GDX и Rprop сходятся в десятки и сотни раз быстрее (см. рисунок 4, *а*). Значения скорости обучения MLP и среднеквадратической ошибки MSE для алгоритмов GD, GDM и GDA, GDX приблизительно равны, следовательно, наличие или отсутствие параметра возмущения *mc* играет незначительную роль. Введение адаптивной настройки параметра скорости обучения α_k увеличивает скорость сходимости алгоритмов GDA и GDX. MLP обучается быстрее при использовании функции *tansig* в пределах требуемого для обучения количества циклов обучения, чем при применении функции *logsig*. Однако применение данной функции для задачи XOR обеспечивает сходимость к заданному уровню MSE не для всех алгоритмов обучения. На гистограммах показано значение среднеквадратичной ошибки MSE для двух групп алгоритмов на *k*-м цикле обучения (рис. 5, *а*, *б)*.



а *б*

Рис. 5. Гистограммы качества обучения для первой (а) и второй (б) группы алгоритмов. Обозначения см. рисунок 4

Сравнение исследуемых алгоритмов второй группы показало практически одинаково высокую скорость обучения, немного медленнее оказался CGP (см. рисунок 4, *б*). Если для срав-

нения алгоритмов использовать не только необходимое количество циклов обучения, но и количество дополнительных переменных, которые потребуются для организации вычислительного процесса, то требования по размеру оперативной памяти для алгоритмов CGP и CGB несколько выше, поскольку на каждой итерации требуется запомнить 4 и 6 векторов, соответственно, а для алгоритмов CGF и SCG — только 3. Среди алгоритмов сопряженных градиентов CGB требует наибольших объемов памяти, но обычно имеет самую быструю сходимость. Алгоритм Rprop показал практически такую же скорость обучения, как и алгоритмы второй группы, однако он не требует использования процедур одномерного поиска, характеризуется высоким быстродействием и предъявляет незначительные требования к объему оперативной памяти.

Сравнительный анализ алгоритмов оптимизации позволил получить практические рекомендации по их использованию совместно с BPL для разработчиков широкого класса систем в области искусственного интеллекта.

Литература

1. Хайкин, С. Нейронные сети: полный курс / С. Хайкин: пер. с англ., под ред. Н.Н. Куссуль. — 2-е изд., испр.— М.: ООО “И.Д. Вильямс“, 2006. — 1104 с.
2. Арсирий, Е.А. Классификация линейно-разделимых сигналов на основе персептрона Розенблатта и линейных адаптивных фильтров / Е.А. Арсирий, Т.В. Гройсман // Холодил. техника и технология / Одес. гос. акад. холода. — Одесса, 2008. — № 4 (114). — С. 71 — 76.
3. Медведев, В.С. Нейронные сети. MATLAB 6 / В.С. Медведев, В.Г. Потемкин. — М.: ДИАЛОГ_МИФИ, 2002. — 496 с.
4. Калан, Р. Основные концепции нейронных сетей: пер. с англ. — М.: Вильямс, 2003. — 288 с.

Рецензент д-р техн. наук, проф. Одес. нац. политехн. ун-та Антощук С.Г.

Поступила в редакцию 30 марта 2009 г.