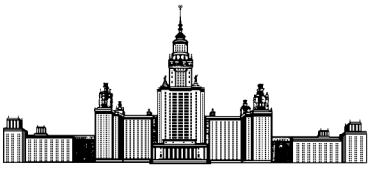
Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования



ДИПЛОМНАЯ РАБОТА

«Методы построения неполносвязных нейронных сетей и их приложения в задачах прогнозирования»

Выполнила:
студентка 5 курса 517 группы
Карпинская Алина Викторовна

Научный руководитель:
д.ф-м.н.
Воронцов Константин Вячеславович

Москва, 2010

Содержание

§1 Введение

Известно огромное количество различных методов обучения нейронных сетей. Одним из классических считается метод обратного распространения ошибки, основанный на градиентном методе минимизации. Суть метода — в эффективном вычисление градиента, которое становится возможным благодаря рациональному хранению промежуточных переменных при дифференцировании суперпозиции функций. Однако в чистом виде метод обратного распространения работает плохо. Функционал является многоэкстремальным избыточное число весовых коэффициентов приводит к нежелательным явлениям мультиколлинеарности и переобучения. Не ясно, какой должна быть структура сети для решения данной конкретной задачи: количество слоев, количество нейронов в них, связи между нейронами. Для решения этих проблем были предложены многочисленные эвристики, улучшающие стандартный метод обратного распространения ошибки. Каждая эвристика имеет множество вариантов реализации в различных пакетах свои настраиваемые параметры, которые должен задавать эксперт, исходя из априорных соображений. Далеко не все эвристики снабжаются рекомендациями по настройке. Различные эвристики могут взаимодействовать друг с другом нетривиально, что далеко не всегда приводит к улучшению качества работы сети. Это порождает новую проблему: как выбирать совокупность настраиваемых параметров для всех оптимизационных методов? Получается, что качество нейронной сети зависит не только от качества исходных данных, но и от субъективного опыта эксперта, решающего задачу.

Целью данной работы является разработка адаптивного метода обучения нейронных сетей для задач классификации, регрессии и прогнозирования, полностью автоматически задающих структуру сети и оценивающих параметры всех необходимых эвристик.

§2 Стандартные методы настройки нейронных сетей

2.1 Основные понятия и обозначения

Пусть X — пространство объектов; Y — множество допустимых ответов; $y^* : X \to Y$ — целевая зависимость, известная только на объектах обучающей выборки $X^l = (x_i, y_i)_{i=1}^l$, $y_i = y^*(x_i)$. Требуется построить алгоритм $a : X \to Y$, аппроксимирующий целевую зависимость y^* на всем множества X. Будем предполагать, что объекты описываются n числовыми признаками $f_j : X \to \mathbb{R}, j = 1, \ldots, n$. Вектор $(f_1(x), \ldots, f_n(x)) \in \mathbb{R}^n$ называется признаковым описанием объекта x.

2.1.1 Нейрон

Структурной единицей нейронной сети является нейрон. На вход нейрону подается n-мерный вектор признакового описания $x = (x^1, \ldots, x^n)$. Каждому входу соответствую весовой коэффициент — $\omega_1, \ldots \omega_n$. Выходом нейрона является значение функции активации от взвешенной суммы входных значений. Также на вход подается свободный член, имеющий постоянное значение, который также имеет вес.

$$a(x) = \varphi\left(\sum_{j=1}^{n} \omega_j x^j\right) = \varphi(\langle \omega, x \rangle)$$

Данный вид нейрона является простейшим линейным классификатором и может самостоятельно обучаться. Если нейрон находится в скрытом или выходном слое нейронной сети, то на вход ему подаются выходные значения других нейронов.

2.1.2 Нейронная сеть

Соединять нейроны в сеть можно многими способами. Рассмотрим наиболее распространенный и исследованный способ композиции нейронов — многослойный персептрон. Нейроны располагаются слоями, при этом каждый нейрон из следующего слоя связан со всеми нейронами предыдущего. Выделяют три типа слоя — входной слой неройнов, скрытый слой и выходной слой. Чаще всего используется трехслойный персептрон, то есть с одним скрытым слоем. По теореме Колмогорова сеть такой структуры с нелинейной функцией активации способна аппроксимировать любую функцию.

Выходные значения алгоритма снимаются с выходов последнего слоя нейронной сети. Количество нейронов в выходном слое соответствует размерности вектора ответов. Количество нейронов в входном слое соответствует размерности признакового описания объекта. Количество нейронов в скрытых слоях выбирается экспертом или посредством эвристик по оптимизации структуры сети.

2.2 Метод обратного распространения ошибки

Рассмотрим многослойный персептрон с одним входным слоем, одним скрытым и одним выходным слоем. В данном случае будет рассмотрена полносвязная нейронная сеть. Алгоритм также подходит для неполносвязной сети. Неполносвязная сеть — сеть в которой отсутствуют некоторые связи между нейронами соседних слоев. Перед описанием алгоритма рассмотрим ключевые параметры — функционал качества и функции активации.

Функционал качества.

Цель процесса обучения — минимизация функционала качества. Функционал качества является суммой функций потерь на обучающей выборки объектов.

$$Q(\omega) = \frac{1}{n}\sum_{i=1}^{l} \mathscr{L}(a(x_i), y_i) \to \min_{\omega},$$

где $\mathscr{L}(a(x_i), y_i)$ — заданная функция потерь, характеризующая величину ошибки ответа a при верном ответе y. Наиболее универсальной является квадратичная функция потерь.

$$\mathscr{L}(a(x_i), y_i) = (a(x_i) - y_i)^2$$

В зависимости от задачи функция потерь может принимать необходимый вид. Например, функция потерь может сильнее штрафовать положительное отклонение от цели и менее сильно отрицательное. Для реализации BackProp необходима дифференцируемость функции потерь.

Функции активации.

Функция активации нейронов (характеристическая функция) φ — нейлинейный преобразователь выходного сигнала сумматора. Может быть одной и той же для всех нейронов сети. В этом случае сеть называют однородной (гомогенной). Если же

φ зависит еще от одного или нескольких параметров, значения которых меняются от нейрона к нейрону, то сеть называют неоднородной (гетерогенной). В данном случае будем рассматривать гомогенные сети.

Функция активации имитирует отклик на входной сигнал, чем сильнее сигнал, тем сильнее отклик. Большинство функций активации являются симметричными и нечетными. Функция активации определяется значениями ответов на объектах. В случае если $Y \in \{-1, 1\}$ наиболее подходит гиперболическая функция. Часто используемые функции активации:

$\theta(z) = [z \geq 0]$ ступенчатая функция Хэвисайда;

$\sigma(z) = (1 + e^{-z})^{-1}$ сигмоидная функция;

$\text{th}(z) = 2\sigma(2z) - 1$ гиперболическая функция;

$ln(z + \sqrt{x^2 + 1})$ логарифмическая функция;

$\exp(-z^2/2)$ гауссовская функция;

z линейная функция;

Back propagation. Введем следующие обозначения. Для общности полагаем $X = \mathbb{R}^n$, $Y = \mathbb{R}^m$. Пусть выходной слой состоит из M нейронов с функциями активации σ_m и выходами a^m, $\quad m = 1, \ldots, M$. Предыдущий слой из H нейронов с функциями активации σ_h и выходами u^h, $\quad h = 1, \ldots, H$. Веса связей между h-м нейронами скрытого слоя и m-м нейроном выходного слоя будем обозначать через ω_{hm}. Перед скрытым слоем находится входной слой, выходы входного слоя — v^j, $\quad j = 1, \ldots, J$ и веса ω_{jh}. По сути в данном случае входной слой является признаковым описанием. В начале работы сети весовые коэффициенты инициализируются небольшими значениями:

$$\omega_{jh} = rand\left(-\frac{1}{2n}, \frac{1}{2n}\right)$$

$$\omega_{hm} = rand\left(-\frac{1}{2H}, \frac{1}{2H}\right),$$

где $rand(a, b)$ функция, возвращающая случайное число из промежутка $[a, b]$.

На каждой итерации из обучающей выборки берется некоторый объект x_i. Существуют разные способы выбора объекта, они будут рассмотрены далее. При описание

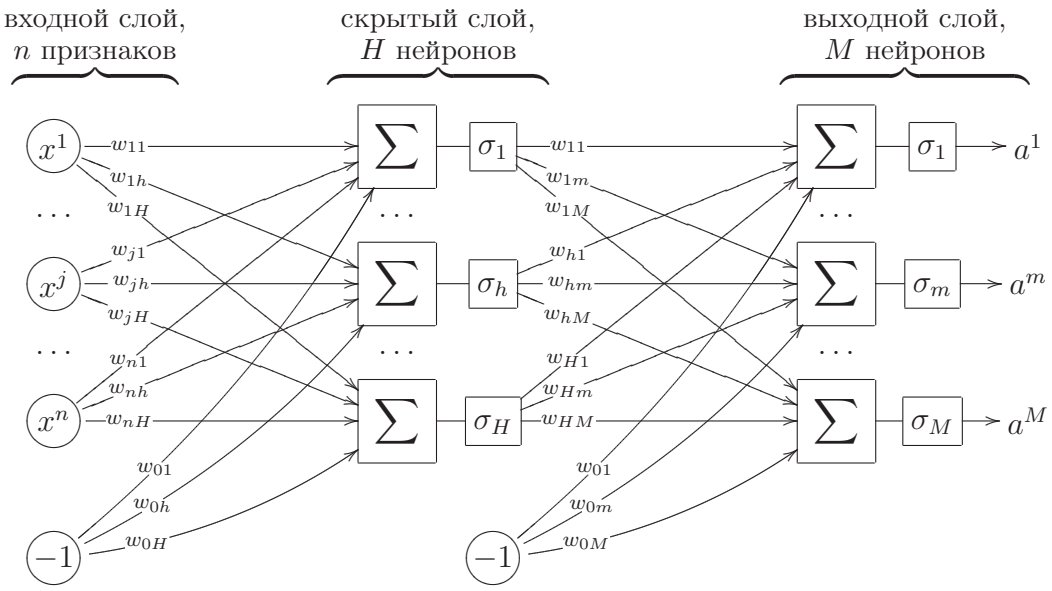


Рис. 1: Многослойная сеть с одним скрытым слоем.

алгоритма остановимся на случайном выборе объекта из обучающей выборки. Выходные значения сети на объекте x_i вычисляются как суперпозиции:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H}\omega_{hm}u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J}\omega_{jh}v^j(x_i)\right);$$

Будем использовать в качестве функции потерь квадратичную функцию потерь, тогда функционал ошибки на отдельном объекте x_i принимает следующий вид:

$$Q(\omega) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2.$$

Сначала выпишем частные производные Q для выходного слоя, затем для скрытого. В таком порядке они вычисляются в методе обратного распространения.

$$\frac{\partial Q(\omega)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m;$$

$$\frac{\partial Q(\omega)}{\partial u^h} = \sum_{m=1}^{M}(a^m(x_i) - y_i^m)\sigma_m'\omega_{hm} = \sum_{m=1}^{M}\varepsilon_i^m\sigma_m'\omega_{hm} = \varepsilon_i^h.$$

Частная производная на выходном слое является ошибкой алгоритма на данном объекте. Частные производные выходов скрытых слоев также являются ошибками выходных значений нейронов скрытого слоя. Через σ_m' обозначена производная функции активации взятая при том же значении аргумента, что и в формулах вычисления выходов нейронов.

Заметим, что ε_i^h вычисляется по ε_i^m. Таким образом, если Заметим, что если вектор ошибок нейронов выходного слоя скалярно умножить на значения $\varepsilon_i^m \sigma'_m$, то на выходе получается ε^h. Таким образом, направляя вычисления в обратном направление получаем вычисление ошибок на каждом нейроне. Далее в соответствии с этими ошибками производим коррекцию весов:

$$\omega_{hm} = \omega_{hm} - \eta \varepsilon_i^m \sigma'_m u^h, h = 0, \ldots, H, m = 1, \ldots, M$$

$$\omega_{jh} = \omega_{jh} - \eta \varepsilon_h^j \sigma'_h x^j, j = 0, \ldots, n, h = 1, \ldots, H$$

Где η — шаг обучения сети. На каждом шаге вычисляется ошибка на одном объекте, но для выбора момента останова необходимо знать ошибку на всей выборке. В программной реализации рассчет точного значения функционала на всех объектах каждую итерацию займет неприемлемо много времени, поэтому можно воспользоваться аппроксимированным значением функционала ошибки:

$$Q := Q + \alpha(Q_i - Q),$$

где Q_i — ошибка сети на объекте x_i, а α коэффициент значимости одного шага, как правило его удобно выбрать $\frac{1}{l}$. После вычисления ошибки, пересчета весов и получения аппроксимированного значения функционала действия повторяются, до тех пор пока не выполнится условие останова. Чаще всего условием останова является стабилизация функционала. Далее мы рассмотрим другие варианты критериев останова.

Метод обратного распространения ошибки имеет ряд недостатков, таких как застревание в локальных минимумах, медленная сходимость, переобучение. Для преодоления недостатков метода применяется ряд эвристик, которые будут рассмотрены далее.

2.3 Методы предварительной обработки данных

Нейронная сеть способна работать с различными типами выборок, при этом диапазоны различных признаков могут сильно различаться. При этом как видно из модели все они подаются на один сумматор. Для облегчения процесса обучения реко-

мендуется провести нормализацию признаков. Два способа нормализации признака:

$$x^j = \frac{x^j - x^j_{min}}{x^j_{max} - x^j_{min}}, \quad j = 1, \ldots, n,$$

$$x^j = \frac{x^j - x^j_{cp}}{x^j_{\sigma}}, \quad j = 1, \ldots, n,$$

где $x^j_{min}, x^j_{max}, x^j_{cp}, x^j_{\sigma}$ — соответственно минимальное, максимальное, среднее значение и среднеквадратичное отклонение признака x^j.

2.4 Выбор начальной структуры сети

Сеть инициализируется некоторыми начальными значениями параметров. Одним из важных параметров является количество нейронов в скрытых слоя и количество слоев. По теореме Колмогорова для любой задачи будет достаточно $2n - 1$ в скрытом слое. Интуитивно понятно, что такое количество нейронов образуют избыточную нейронную сеть. Угадать оптимальное значение нейронов представляется невозможным, поэтому предлагается брать около $[n/2]$ нейронов в скрытом слое, где n- размерность признакового вектора.

Существует эвристический прием Bagging, предлагающий обучать нейроны первого скрытого слоя на непересекающихся подвыборках. При применении важно учитывать, чтобы после настройки на выборках не образовывалось похожих нейронов. Чем больше вариантов разбиения гиперплоскостями можно получить, тем лучше может быть классификация.

2.5 Эвристики, направленные на улучшение сходимости

Градиентные методы первого порядка сходятся довольно медленно. Ньютоновские методы второго порядка также непрактичны, так как требуют вычисления матрицы вторых производных функционала $Q(\omega)$, которая имеет слишком большой размер. Необходимы специальные ухищрения, чтобы приспособить стандартные методы оптимизации для настройки нейронных сетей. Одним из методов повышения скорости сходимости является диагональный метод Левенберга-Маквардта, предложенный в обзоре [6].

В данном методе величина шага вычисляется индивидуально для каждого весового коэффициента, при этом используется только один диагональный элемент матрицы вторых производных:

$$\theta_{jh} = \frac{\theta}{\frac{\partial^2 Q}{\partial \omega_{jh}^2} + \mu},$$

где θ постоянное значения шага обучения, μ — параметр предотвращающий обнуление знаменателя и, как следствие, неограниченный рост шага. Значение θ/μ — темп обучения на ровных участках функционала $Q(\omega)$.

2.6 Эвристики, направленные на выбивание из локального минимума

Функционал ошибки сети является многоэкстремальным и зачастую в процессе обучения попадает в локальный минимум где и застревает и стабилизируется. Чтобы избежать раннего останова обучения, который произойдет после стабилизации при высоком значение функционала используют простую эвристику, называемую Jog Of Weights (англ. встряска, подталкивание). Метода заключается в небольшом случайном сдвиге текущих значений весовых коэффициентов. Даже незначительный изменения весовых коэффициентов могут привести к новым результатам работы нейросети. По сути такая эвристика является симбиозом случайным поиска и градиентного метода. Применять Встряску стоит в случае роста или стабилизации функционала.

2.7 Эвристики, направленные на устранение «паралича» сети

Паралич сети возникает при неограниченном увеличении весов [3]. Аргумент функции активации попадает в область, где функция активации имеет горизонтальную асимптоту, производная функции активации стремится к нулю, в результате веса не модифицируются, и сеть «застревает» в таком положении.

«Сокращение весов» (weight decay) — это один из способов предотвратить паралич. Данный метод является частным случаем регуляризации некорректно поставленных задач по А. Н. Тихонову. Идея заключается в ограничение роста абсолютных

значений весов. Добавим к минимизируемому функционалу штрафное слагаемое:

$$Q_\tau(w) = W(w) + \frac{\tau}{2}\|\omega\|^2.$$

Изменение функционала приводит к появлению аддитивной поправки градиента:

$$\frac{\partial Q_\tau(\omega)}{\partial \omega} = \frac{\partial Q(\omega)}{\partial \omega} + \tau\omega.$$

При этом правило обновления весов принимает вид

$$\omega := \omega(1 - \eta\tau) - \eta\frac{\partial Q(\omega)}{\partial \omega}.$$

Преимущество метода в том, что он очень просто реализуется, сочетается со многими функционалами и многими градиентными методами оптимизации. Кроме предотвращения паралича сети, этот метод повышает устойчивость весов в случае мультиколлинераности — когда имеются линейно зависимые или сильно коррелированные признаки. Дополнительный управляющий параметр τ должен обеспечивать компромисс между устойчивостью весов и настройкой на конкретную выборку.

Один из способов выбора параметра — скользящий контроль. Но этот способ занимает очень много времени и является не эффективным в использовании в программной реализации. Существует более простая эвристика выбора параметра τ. Начнем поиск оптимального значения параметра с некоторого значения τ^0. На очередной итерации выберем значение $\tau = \tau^0 + \Delta$, где Δ — малое случайное значение. На след итерации τ^0 пересчитывается следующим образом:

$$\tau^0 = \frac{\sum\limits_{i=1}^{T} \exp(-\beta e_i)\tau_i}{\sum\limits_{i=1}^{T} \exp(-\beta e_i)},$$

где T — текущая итерация, e_i — ошибка на i-ой итерации, β — параметр вероятностного распределения. Таким образом, τ^0 будет сдвигаться в сторону значения τ, минимизирующего значения ошибки.

2.8 Критерии остановки процесса обучения

Как правило, в литературе встречаются рекомендации, что обучение следует продолжать до стабилизации функционала. Возникает вопрос — как обнаружить стаби-

лизацию функционала и как понять, что это не локальный максимум? Существует несколько критериев останова:

1. Для предотвращения переобучения останов происходит в тот момент, когда функционал качества на контрольной выборки начинает расти, а на обучении падать. Такая ситуация означает, что сеть настраивается на объектах обучающей выборки, а ее обобщающая способность снижается.

2. Обучение останавливается при достижении функционала установленного минимального значения.

3. Останов производится после достижение определенного количества итераций.

4. Остановить обучение при стабилизации функционала качества.

При этом логично применять правила в указанном порядке. Наиболее сложно установить событие стабилизации функционала. Для этого, например, можно воспользоваться статистическими методами. Способ подсчета должен быть быстрым, так как будет осуществляться каждую итерацию, чтобы не пропустить момент. В программной реализации предложенной в этой работе, стабилизация оценивается по изменению среднего значения функционала за последние n итерации и предшествующие им n итераций. Расчет данного показателя легко осуществляется и позволяет отслеживать быстро меняющуюся ситуацию.

2.9 Методы оптимизации структуры сети

Одним из главных параметров алгоритма, который просят пользователей выбрать самостоятельно — является количество нейронов в скрытом слое. В худшем случае пользователю необходимо выбрать количество всех слоев и нейронов в каждом из них. Сложность структуры сети влияет на обобщающую способность алгоритма, на скорость сходимости и на вероятность паралича. Таким образом, хотелось бы избавиться от вмешательства пользователя в настройку столь важного параметра и выбирать структуру сети алгоритмическими методами.

2.9.1 Последовательное наращивание

Условия добавления нейронов.

Изначально скрытый слой сети должен содержать некоторое количество нейронов n, оно должно быть явно меньше чем количество входов. Далее запускается процесс обучения сети. В некоторый момент процесс останавливается и в слой добавляется нейрон. Рассмотрим вопрос выбора момента прерывания. В процессе обучения на каждой итерации снимаются такие показатели, как аппроксимированный функционал качества на обучение, аппроксимированный функционал качества на контроле, динамика функционала качества на контроле. С периодичностью T итераций будем проверять значения трех показателей. Если динамика функционала неубывающая, то наращиваем сеть, в противном случае проверяем показатели через T итераций.

Способы добавления нейронов

Нейроны, инициализированные случайными числами, для сети все равно что инородные тела, поэтому их следует подготавливать к вставке в нейронную сеть. Нейроны первого слоя могут быть обучены на подвыборке обучающей выборки. Для нейронов второго скрытого слоя и дальнейших сложнее подготовить предварительную инициализацию. Суть та же, обучить их на подвыборке, зафиксировав остальные нейроны. Остается понять нужно ли добавлять нейрон или для преломления динамики возрастания функционала качества необходимо применить другие методы. Далее этот вопрос будет рассмотрен в пункте 4.

2.9.2 Прореживание

Изначально в сети или после наращивания могут присутствовать лишние связи и лишние нейроны. Поскольку каждый скрытый нейрон представляет гиперплоскость [4], разделяющую множество данных на кластеры, прореживание сети упрощает такое разделение и усиливает способность к обобщению.

Обнаружение лишних связей

Простейшим способом прореживания — учет величины весов. Веса, которые значительно меньше средних оказывают незначительное влияние на общий уровень сигнала, поэтому их можно оборвать. Но это правило работает далеко не всегда. Другой способ — учет чувствительности сети к изменению весов. Без серьезных последствий

для сети из нее могут быть исключены только те веса, чувствительность к изменениям у которых оказывается минимальной.

Предполагаем, что после стабилизации функционала ошибки Q вектор весов w находится в локальном минимуме, тогда Q может быть аппроксимирован квадратичной формой:

$$Q(\omega + \delta) = Q(\omega) + \frac{1}{2}\delta^T H(\omega)\delta + o(\|\delta\|^2),$$

где $H(\omega) = \left(\frac{\delta^2 Q(\omega)}{\delta\omega_{jh}\delta\omega_{j'h'}}\right)$ — гессиан, матрица вторых производных. Предполагается, что диагональные элементы доминируют в гессиане, а остальными частными производными можно пренебречь [6].

$$\delta^T H(\omega)\delta = \sum_{j=0}^{J}\sum_{h=1}^{H}\delta_{jh}^2\frac{\delta^2 Q(\omega)}{\delta\omega_{jh}^2}.$$

В качестве меры значимости веса ω_{ij} используется показатель s_{ij} (англ. salience), который определяется в виде

$$S_{jh} = \omega_{jh}^2\frac{\delta^2 Q(\omega)}{\delta\omega_j^2 h}.$$

Существует простая модификация классического метода OBD, она заключается в удаление лишних нейронов. Будем удалять нейрон если суммарный *salience* его входных или выходных синаптических связей наименьший в сети. Данная модификация особо интересна в применении к нейронам входного слоя, так как позволяет отбирать значимые признаки и не учитывать шумовые и не значимые.

Другой метод оптимизации структуры сети — Optimal Brain Surgery [4] (далее OBS), который считается развитием метода OBD. Отправная точка метода — разложение функционала качества в ряд Тейлора и игнорирование членов первого порядка. В этом методе учитываются все компоненты гессиана, а коэффициент ассиметрии веса определяется в виде

$$S_i = \frac{1}{2}\frac{\omega_{ij}^2}{[H^{-1}]_{ijij}}$$

Отсекается вес с наименьшим значением S_i. При этом происходит коррекция весов, позволяющая вернуть сеть в состояние, соответствующее минимуму функционала

качества, несмотря на отсечение веса.

$$\Delta\omega = \frac{\omega_i j}{[H^{-1}]_{ijij}} H^{-1} e_i$$

где e_i — единичный вектор с единицей в $i-$ой позиции. Коррекция производится после каждого отсечения. И в этом заключается основное отличие от OBD. В результате этого в OBD можно отсекать сразу много связей за одну итерацию, тогда как в OBS можно только одну.

Недостатком OBS является вычислительная сложность, которая гораздо выше чем у OBD. В OBS выполняется рассчет полной матрицы и обратной ей формы. На практике эти вычисления можно упростить при использовании аппроксимированной формы матрицы, обратной гессиану. Но такое упрощение приведет к снижению качества искомого решения.

Применение метода

Эвристики предназначенные для прореживания сети применяются при стабилизации функционала. Таким образом, чтоб применить одну из данных эвристик нужно отследить стабилизацию. В данном случае это можно сделать так же как и для метода наращивания сети, только критерием применения становится малое приращение среднего значения функционала. С другой стороны снижение аппроксимированного значения функционала также говорит о стабильной структуре сети. Значит при отрицательном приращении функционала также можно применять данный вид эвристик.

§3 Адаптивный метод обучения нейронных сетей

Применения одной из перечисленных выше эвристик не даст желаемого результата. Необходимо найти способ применения нужной эвристики в нужный момент и способ создания оптимальной структуры сети без участия пользователя. Решением данной задачи может являться алгоритм адаптивного поиска ключевых параметров.

Внесем в стандартный алгоритм обратного распространения ошибки некоторые модификации. Каждую итерацию будем вычислять показатель стабилизации $Stab$. Установим параметр $Interval$ отвечающий за цикличность проверок показателя $Stab$.

Рассмотрим функцию тестового тестового прогона обучения. Суть функции создать точную копию сети и обучить ее на заданное количество итераций вперед с целью снять показатели. На выходе мы имеем новую сеть, значение стабилизации функционала и сам функционал. Теперь, когда нас интересует, будет ли эффективна выбранная эвристика, есть возможность создать новую нейросеть, идентичную начальной, применить к ней эвристику и посмотреть результат обучение через заданное кол-во итераций.

Есть эвристики, которые включаются при инициализации сети и далее применяются по ходу ее обучения, такие как сокращение весов или выбор величины шага, назовем такие эвристики фоновыми. Другие эвристики необходимо активировать лишь в нужные моменты, когда они будут работать эффективно. К таким эвристикам относится наращивание, прореживание и встряска. Для удобства будем называть их итерационными. При этом некоторые методы подавляют друг друга. Также учитываем, что после применения эвристики необходимо время для стабилизации функционала.

Принимая во внимания все рассмотренные выше особенности эвристик и их совместного применения построим алгоритм поиска значений параметров во время процесса обучения.

Описание алгоритма

1. Исходную выборку разобъем на подвыборки обучения и контроля, так что процентное соотношения объектов в классах сохранилось как в главной выборке.

2. Применим процедуру нормализации объектов выборки, сохранив при этом коэффициенты нормализации на случай добавления новых объектов.

3. Активируем для нейросети эвристики адаптивного поиска значения параметра регуляризации, эвристику выбора темпа обучения методом Левенберга-Маквардта и остальные фоновые эвристики.

4. Инициализируем трехслойную сеть с тремя нейронами в скрытом слое. Применим Bagging для определения начальных значений весовых коэффициентов нейронов.

5. Проведем обучение первые $Interval$ итераций без применения итерационных эвристик. Это необходимо для вычисления параметра $Stab$ и "привыкания"сети к выборке.

6. Проверим значение параметра $Stab$. Если $Stab$ больше нуля, это говорит о том, что функционал ошибки возрастает и есть смысл применить одну из итерационных эвристик. В этом случае переходим к следующему пункту, в противном случае тестируем эвристику OBD, применяя ее при хороших результатах и переходим к пункту 10.

7. Протестируем с помощью функции тестового прогона обучения получим показатели эффективности сети в текущем виде и после применения итерационных эвристик.

8. Если для текущей комплектации сети показатель отрицательный, что говорит о снижение функционала, то эвристики не тестируются с целью экономии времени. Обучение продолжается без применения эвристик.

9. В противном случае, в результате получаем значение Stab через $Interval$ обучений для эвристик наращивания, встряски и прореживания, а также для текущей комплектации сети. Продолжаем обучения из той копии нейросети, которая имела наилучшие показатели.

10. Обучаемся $Interval$ итераций для "привыкания"сети, если применялась эвристика.

11. Если не выполняются условия останова переходим на пункт 6, иначе останавливаем обучение.

Дополнения к алгоритму Таким образом количество параметров алгоритма было сведено до трёх: максимальное количество итераций, минимальное значение функционала качества и значение $Interval$. Значение $Interval$ может также выбираться адаптивно, например как среднее значение итераций, через которое в обучение проявляется эффект применения эвристики. Также $Interval$ может являться зависимостью от количества объектов в обучающей выборки. Рекомендованное значение — 100 итераций.

Важно отметить, что при тестировании различных эвристик объекты на вход подаются в одинаковым порядке, который сгенерирован заранее. Это позволяет выбрать лучшую эвристику в сходных условиях.

Опишем более подробно процесс тестирования итерационных эвристик.

Наращивание

Всегда существует несколько способов нарастить структуру сети. Пусть в сети L скрытых слоев, тогда существует $L + 1$ способ модификации. Первые L способов — добавить 1 нейрон в каждый скрытый слои, $L + 1$-ый способ — добавить новый слой из двух нейронов между последним скрытым слоем и выходным, слой из одного нейрона не имеет смысла. Можно добавлять только один нейрон за раз, а потом тестировать, добавление большего количество нейронов потребует большего времени настройки. Каждый способ наращивания необходимо тестировать. Поэтому в пункте 9 под тестированием эвристики наращивания понимается, что процедура тестового прогона применяется $L+1$ раз, то есть для каждого способа. Выбирается тот способ, который имеет наименьшее значение $Stab$.

Прореживание

Ситуация с прореживанием складывается подобным образом. Первое применение — удаление синаптических связей, чье значение меньше порогового. В данном случае пороговое значение может являться параметром алгоритма, но так как оно не зависит от решаемой задачи, оно фиксировано. Второе применение — удаление нейронов с наименьшей суммой *salience* по входным или входным связям. Выбирается эвристика показавшая наилучший результат.

Встряска

Эвристика является разновидностью случайного поиска. Логично попробовать несколько раз применить ее к текущей сети и выбрать лучший вариант. Количество раз является параметром. Но так как один тестовый прогон несколько раз за обучение занимает много времени, количество попыток случайного поиска не может быть очень большим. В данном случае в качестве рекомендации предлагается делать три попытки.

Модификации алгоритма

В данном алгоритме рассмотрено применения всех основных итерационных эвристик параллельно. В таком случае наращивание и прореживание могут подавлять друг друга. На этот случай существует модификация, когда сначала применяется только наращивание, а после прореживание и встряска. Такой алгоритм часто может оказаться предпочтительней, так как в первом случае сеть имеет риски стать избыточной.

Чтобы снизить временные затраты, эвристики могут быть запущены без тестирования. Данный способ ускоряет обучение, но требует введения пороговых значений для эвристик, что увеличит количество параметров.

Недостатки и преимущества алгоритма

Основным недостатком алгоритма является время его работы. Побочным эффектом является появление новых параметров взамен на старые. Относительно новых параметров сеть более устойчива. К параметрам относим значение Interval и пороговые значения методов. Очевидно, что относительно этих значений обучение сети более устойчивее, чем относительно количества нейронов в скрытом слое и способа подачи объектов.

§4 Вычислительные эксперименты

Вычислительные эксперименты проводились на реальных данных из репозитория UCI [5]. В приведенных экспериментах использовалась задача Breast Cancer Wisconsin (Diagnostic).

Регуляризация и диагональный метод Левенберга-Марквардта. Сгенерируем выборку типа XOR с помощью групп объектов расположенных в соответствии с нормальным распределением с разной дисперсией. Сформируем для всех нейросетей одинаковый порядок подачи объектов на обучение нейросети. Эвристики можно сравнивать, если совпадают условия, в которых они применяются. Создадим четыре одинаково инициализированные сети с тремя нейронами в скрытом слое. На Рисунке 3 представлены графики процессов обучения четырех способов композиции эвристик регуляризации и способа выбора шага Левенберга-Маквардта. В случае данной выборки и данного процесса нейросеть, в которой не применяется ни одна эвристи-

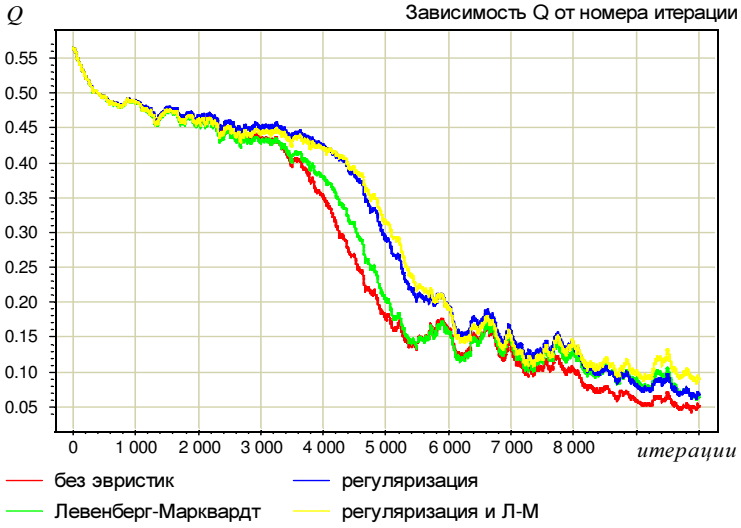


Рис. 2: Сравнение динамики функционалов качества

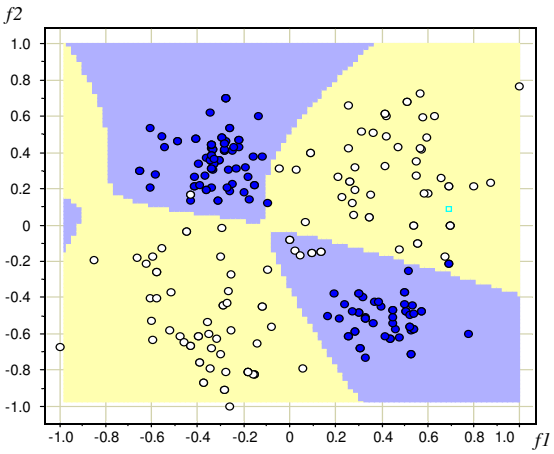


Рис. 3: Без эвристик.

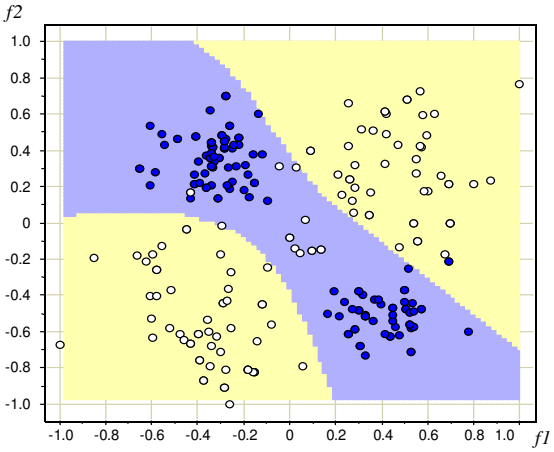


Рис. 4: Левенберг-Марквардт.

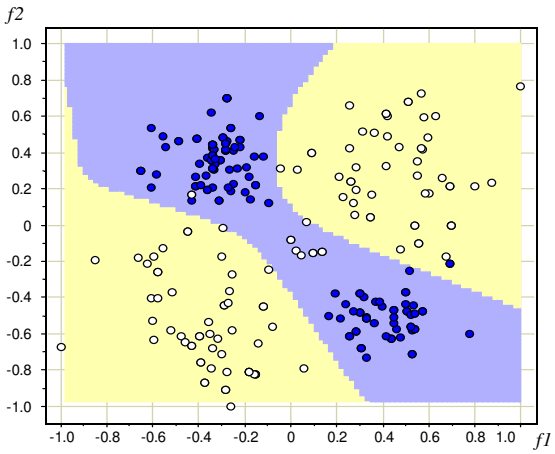


Рис. 5: Регуляризация.

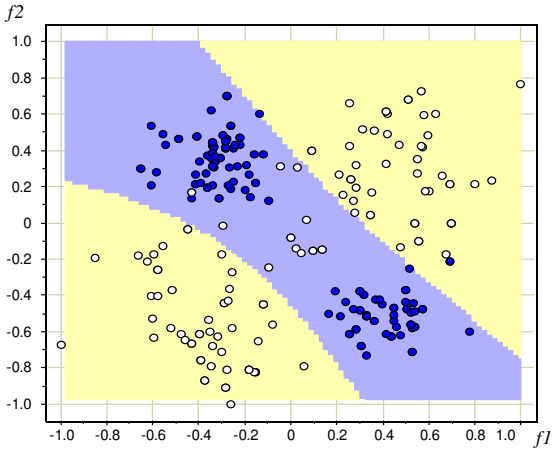


Рис. 6: Левенберг-Марквардт + Регуляризация.

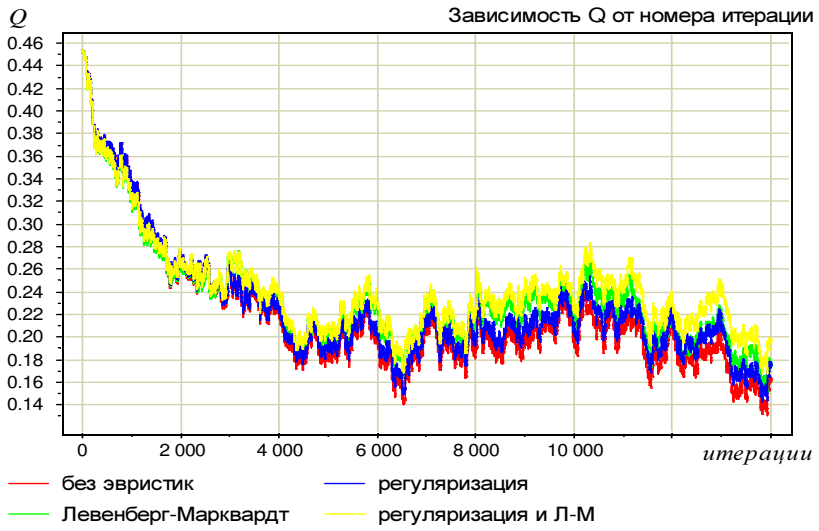


Рис. 7: Сравнение динамики функционалов качества на реальной задаче

ка сеть настраивается точно на выборку, что может являться переобучением. На Рисунке 7 представлен график динамики функционалов качества при обучении на задаче Breast Cancers. Видно, что все композиции данных эвристик имеет близкую скорость сходимости, но при этом применение регуляризации позволяет избежать переобучения. На результат непосредственно влияет порядок подаваемых объектов.

Порядок предъявления объектов. Рассмотрим результат работы эвристики, когда объекты на обучение подаются следующим образом. Из n объектов выбираем тот, на котором чаще допускались ошибки. На Рисунке 7 имеем график процессов обучения. В таблице 1 приведены результаты эксперимента — значения функционалов качества на обучение и на контроле, полученные путём усреднения по 5 запускам нейронной сети одинаковой структуры с одинаковым начальным приближением, на выборке Cancer.

На основе полученных результатов можно сделать вывод, что наиболее быстро обучается сеть при $n = 5$, при этом нет обратной зависимости функционалов качества на обучение и контроле от n, при значениях параметра от 1 до 6. Так как в основе лежит случайный способ подачи объектов, переобучение может начаться только при довольно больших значениях n.

Выбор начального приближения Сделаем предположение, что кроме способа подачи объектов важным фактором обучения является начальная инициализация

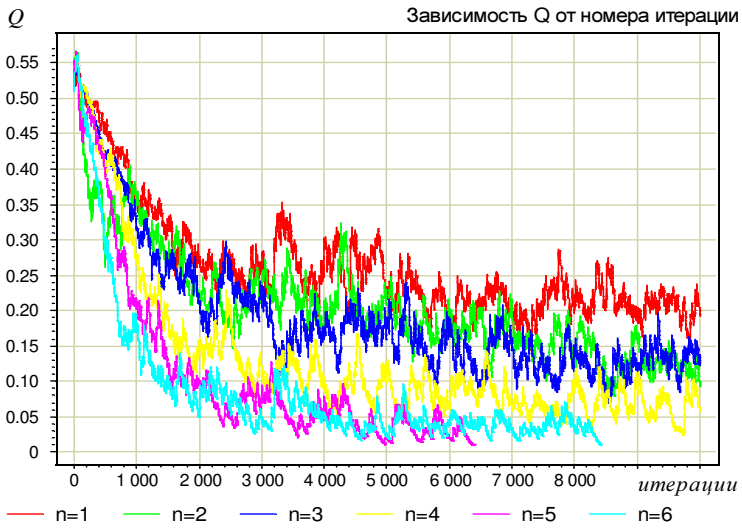


Рис. 8: Сравнение динамики функционалов качества при различных способах подачи объектов

Таблица 1: Доля ошибок на обучении и контроле в зависимости от n.

n	Q на обучение	Q на контроле
0	0,9325	0,7825
1	0,8525	0,725
2	0,8825	0,7825
3	0,915	0,7825
4	0,905	0,74
5	0,895	0,8125

сети. На Рисунке 8 представлены графики функционалов качества нейронных сетей, которые изначально инициализированы одинаково. Далее к одной сети применяют обучение нейронов первого слоя по отдельности (Bagging), к другой случайный сдвиг весовых коэффициентов, третью сеть оставляют без изменений. Как видно на графике на Рисунке 9 все способы инициализации приводят к единой динамики функционала. В случае с хорошо разделимой выборкой обучение нейронов первого слоя значительно ускоряет процесс обучения.

Прореживание На двух одинаковых нейросетях проведем обучения на объектах, подающихся в одинаковом порядке. Во второй нейросети будем удалять все нейроны и связи, значение чувствительности которых меньше заданных пороговых значе-

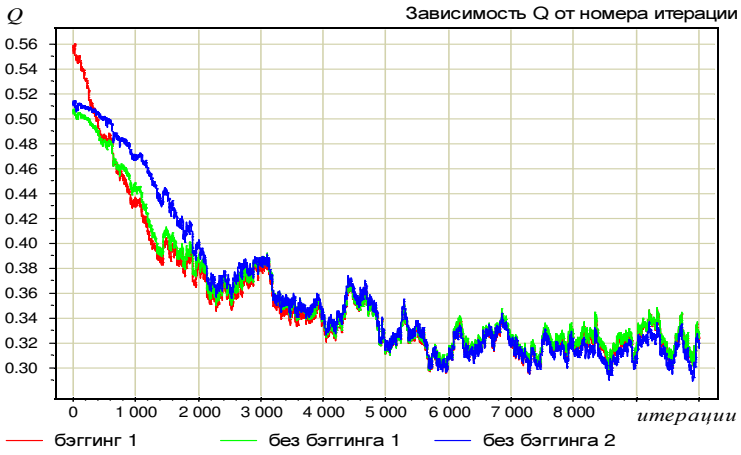


Рис. 9: Сравнение динамики функционалов качества при использовании Bagging и без него

ний. На Рисунке 10 приведены графики процессов обучения. Как видно, обрывание лишних связей влияет на скорость сходимости незначительно. В таблице приведены средние значения функционала ошибок на обучение и на контроля.

Метод	среднее значение Q на обучении	среднее значение Q на контроле
Без OBD	0,782	0,702
С OBD	0,862	0,758

Применение эвристики прореживания понижает риск переобучения и не ухудшает сходимость алгоритма.

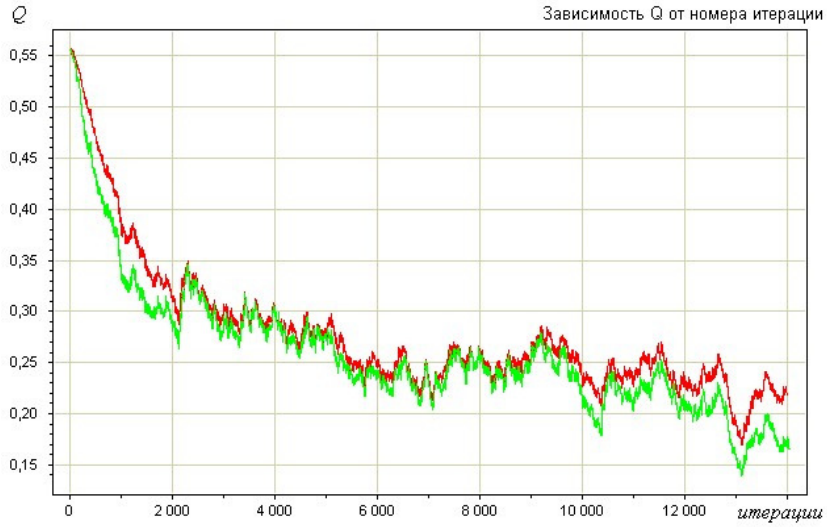


Рис. 10: Сравнение динамики функционалов качества при применении эвристики прореживания и без нее

Наращивание сети Рассмотрим процесс наращивания сети. Создадим выборку, для классификации которой необходимо больше нейронов, чем есть в первоначальной структуре сети. Для этого построим две одинаковые сети с двумя нейронами в скрытом слое. Первая сеть будет постоянной структуры, вторую сеть будем наращивать во время обучения в случае, если функционал возрастает или стабилен. График на Рисунке 11 показывает работоспособность данной эвристики.

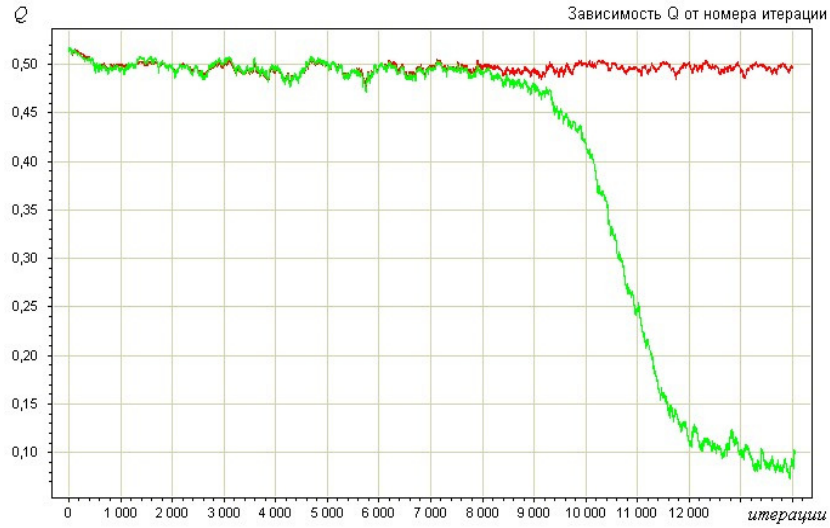


Рис. 11: Сравнение динамики функционалов качества на модельной задаче с эвристикой наращивания сети

Адаптивные алгоритмы Рассмотрим три метода обучения нейронной сети. Первый случай является простым методом обратного распространения ошибки. Второй — применяем прореживание и наращивание, когда динамика функционала удовлетворяет требованиям. Третий случай — параллельно применяем прореживание и наращивание, но только в тех случаях если они дадут снижение функционала. Результаты обучения представлены на Рисунке 12. Необходимо отметить, что результаты, полученные при обучении с применением эвристик, имеют более высокую обобщающую способность, а также демонстрируют более быструю сходимость.

Все эвристики оказывают положительное влияние на ход обучения и на его результат. При этом могут быть все одновременно. На графиках видно что одна эвристика порой лучше другой, но надо помнить, что при обучение используются случайные значения и это вносит большой вклад в ход обучения. При каждом эксперименте

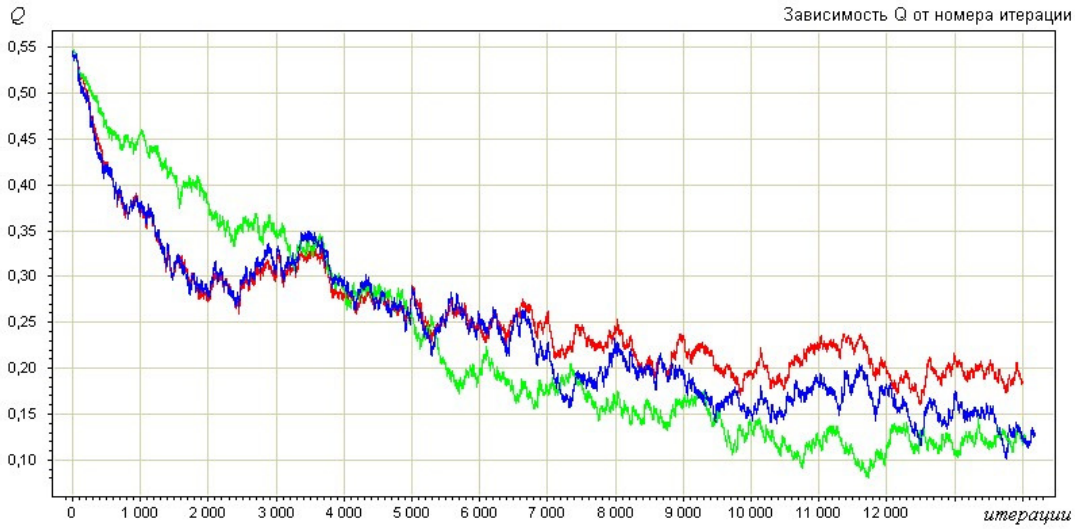


Рис. 12: Сравнение динамики функционалов качества при применении эвристик наращивания и прореживания параллельно двумя разными методами и без применения эвристик

результаты и графики отличаются. Если мы используем адаптивную эвристику, а она не нужна, то она автоматически не будет применяться, но если мы не воспользуемся эвристикой, больше риск потерпеть неудачу.

§5 Заключение

Результаты, выносимые на защиту:

- Предложен адаптивный алгоритм обучения неполносвязных нейронных сетей, основанный на комбинировании большого числа эвристики, и не требующий от пользователя задания структуры сети и значений управляющих параметров.
- Предложены новые эвристики для выбора последовательности предъявления объектов и наращивания сети, подтвердившие свою эффективность в численных экспериментах на модельных и реальных данных.
- Разработана достаточно гибкая среда для построения нейронных сетей произвольной архитектуры и визуального анализа процесса обучения нейронных сетей.

Список литературы

[1] Воронцов К.В. Лекции по искусственным нейронным сетям

[2] *Горбань А.Н. Дунин-Барковский В.Л., Кирдин А.Н, Миркис Е.М., Новоходько А.Ю.*, Нейроинформатика.

[3] *Мак-Каллок У. С., Питтс В.*,Логическое исчисление идей, относящихся к нервной активности

[4] *Осовский С.* Нейронные сети для обработки информации

[5] *Asuncion A., Newman D. J.* UCI Machine Learning Repository // University of California, Irvine, School of Information and Computer Sciences, 2007. www.ics.uci.edu/$\sim$mlearn/MLRepository.html.

[6] Yann LeCun, Leon Bottou Efficient BackProp