Российская Академия Наук

РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ ПРОБЛЕМ БЕЗОПАСНОГО РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ

RUSSIAN ACADEMY OF SCIENCES

NUCLEAR SAFETY INSTITUTE

Препринт ИБРАЭ № IBRAE-1999-09

Preprint IBRAE-1999-09

В.А. Тимонин, В.В. Демьянов, М.Ф. Каневский

ОБЗОР И ПРИМЕНЕНИЕ МЕТОДОВ НЕЛИНЕЙНОЙ ОПТИМИЗАЦИИ ДЛЯ ОБУЧЕНИЯ ИНС МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН

Москва 1999

Moscow 1999

УДК 502.3

Тимонин В.А., Демьянов В.В., Каневский М.Ф. ОБЗОР И ПРИМЕНЕНИЕ МЕТОДОВ НЕЛИНЕЙНОЙ ОПТИМИЗАЦИИ ДЛЯ ОБУЧЕНИЯ ИНС МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН. Препринт № IBRAE-99-09. Москва: Институт проблем безопасного развития атомной энергетики РАН. 1999. 28 с. Библиогр.: 15 назв.

Аннотация

В работе рассматриваются применение различных методов нелинейной оптимизации для обучения искусственных нейронных сетей типа многослойный перцептрон. Рассматриваются стохастические, генетические и детерминистические алгоритмы оптимизации. Различные методы применены к реальным данным по радиоактивному загрязнению почвы Чернобыльскими выпадениями.

Timonin V.A., Demyanov V.V., Kanevski M.F. REVIEW AND APPLICATION OF NON-LINEAR OPTIMISATION METHODS FOR TRAINING OF MULTILAYER PERCEPTRON. Preprint IBRAE-99-09. Moscow: Nuclear Safety Institute. November 1999. 28 p. — Refs.: 15 items.

Abstract

Application of different methods of non-linear optimisation is considered for training of multilayer perceptron ANN. Stochastic, genetic and deterministic optimisation algorithms are considered. Different methods are applied to the real data on the radioactive soil contamination of the Chernobyl fallout.

Обзор и применение методов нелинейной оптимизации для обучения ИНС многослойный перцептрон

В.А. Тимонин, В.В. Демьянов, М.Ф. Каневский

Институт проблем безопасного развития атомной энергетики
113191, Москва, ул. Б. Тульская, 52
тел.: (095) 955-22-31, факс: (095) 958-11-51, e-mail: vadim@ibrae.ac.ru;
http://www.ibrae.ac.ru/~mkanev/

Содержание

1 Введение

Целью настоящей работы является обзор методов оптимизации, применяемых для обучения нейронных сетей типа многослойный перцептрон (в дальнейшем МСП). В работе будет приведено лишь краткое описание структуры МСП, его функционирование и применение. Более подробную информацию по этим вопросам можно найти в любом источнике, посвященном искусственным нейронным сетям ([1], [2], [3], [4], [5], [6]). Конечно, в силу своей универсальности, данные методы могут применяться и для обучения других типов нейронных сетей при соответствующей постановке задачи. В работе будет рассмотрено семейство градиентных методов, метод отжига, генетический алгоритм.

2 Искусственные нейронные сети (ИНС)

Под искусственной нейронной сетью (ИНС) понимается некоторое вычислительное устройство, состоящее из некоторого числа параллельно работающих простых процессорных элементов – *нейронов*, связанных между собой линиями передачи информации – связями или *синапсами*. У нейронной сети выделена группа связей, по которым она получает информацию из внешнего мира, и группа выходных связей, с ко-

торых снимаются выдаваемые сетью сигналы. Нейронные сети применяются для решения различных задач классификации (распознавания) и регрессии (предсказания). Нейронная сеть обучается решению задачи на основании некоторой обучающей выборки – ”задачника”, состоящего из набора пар ”вход–требуемый выход”, и далее способна решать примеры, не входящие в обучающую выборку (способность к обобщению).

2.1 Элементы искусственной нейронной сети

Для описания алгоритмов и устройств в нейроинформатике выработана специальная ”схемотехника”, в которой элементарные устройства – *сумматоры*, *синапсы*, *нейроны* и т.п. объединяются в сети, предназначенные для решения задач

Самый заслуженный и, вероятно, наиболее важный элемент нейросистем – это *адаптивный сумматор*. Адаптивный сумматор вычисляет скалярное произведение вектора входного сигнала $\boldsymbol{x}$ на вектор параметров $\boldsymbol{\alpha}$. На схемах будем обозначать его так, как показано на рисунке 1. Адаптивным называем его из-за наличия вектора настраиваемых параметров $\boldsymbol{\alpha}$. Для многих задач полезно иметь линейную неоднородную функцию выходных сигналов. Ее вычисление также можно представить с помощью адаптивного сумматора, имеющего n+1 вход и получающего на 0-й вход постоянный единичный сигнал смещения (*bias*). Неоднородный адаптивный сумматор показан на Рис. 2.

Нелинейный преобразователь сигнала изображен на Рис. 3. Он получает скалярный входной сигнал $\boldsymbol{x}$ и переводит его в $\boldsymbol{\varphi(x)}$. Теоретически может использоваться любая непрерывная нелинейная функция. Для разных моделей нейронных сетей накладываются дополнительные требования. Например, ”колоколообразный” вид (гауссиан) функции для RBF или GRNN сетей. Для МСП накладываются еще требования неубывания и ограниченности функции. Пример такой функции – экспоненциальный сигмоид: $f(x)=\dfrac{1}{1+e^{-ax}}$, где параметр $\boldsymbol{a}$ определяет крутизну квазилинейной области. График такой функции при $\boldsymbol{a}$=1.0 приведен на Рис. 7. Своей популярностью эта функция обязана простоте вычисления производной через значение самой функции в точке: $f'(x)=\alpha\cdot f(x)(1-f(x))$.

Точка ветвления служит для рассылки одного сигнала по нескольким адресам (Рис. 4). Она получает скалярный входной сигнал $\boldsymbol{x}$ и передает его всем своим выходам.

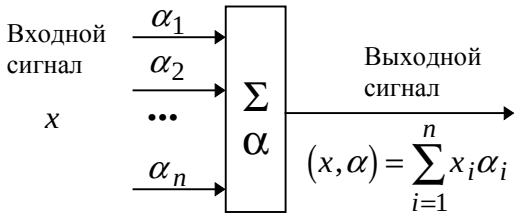


Рис. 1. Адаптивный сумматор

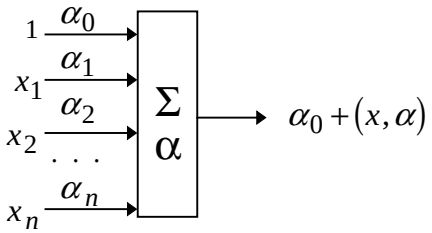


Рис. 2. Неоднородный адаптивный сумматор

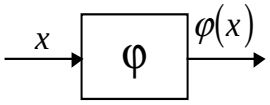


Рис. 3. Нелинейный преобразователь сигнала

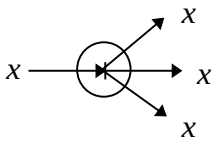
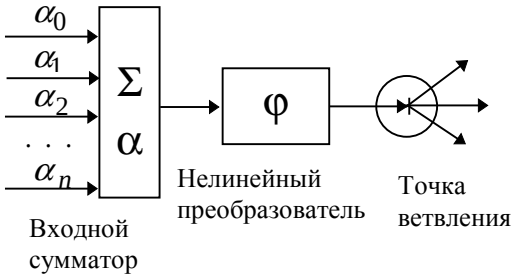


Рис. 4. Точка ветвления



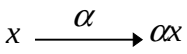


Рис. 5. Формальный нейрон *Рис. 6. Линейная связь (синапс)*

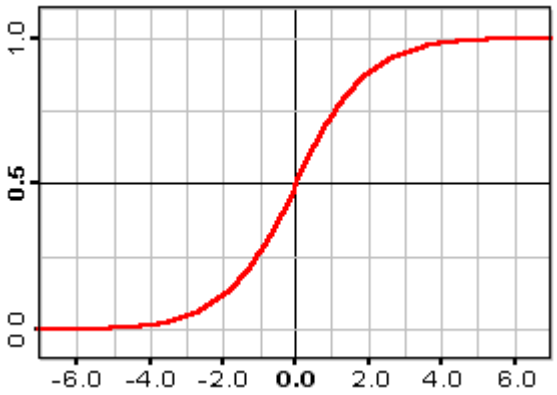


Рис. 7. Экспоненциальный сигмоид

Стандартный формальный нейрон состоит из входного сумматора, нелинейного преобразователя и точки ветвления на выходе (Рис. 5).

Линейная связь (синапс) – отдельно от сумматоров не встречается, однако для некоторых рассуждений бывает удобно выделить этот элемент (Рис. 6). Он умножает входной сигнал ***x*** на "вес синапса" ***a***.

Веса синапсов сети образуют набор адаптивных параметров, по которым производится оптимизация целевой функции.

Таким образом, ИНС – это лишь формальный язык, описывающий некую математическую модель через хорошо известные и понятные функциональные математические элементы. Поэтому не следует рассматривать нейронные сети как некую революции в математике. Другие, хорошо известные алгоритмы, могут быть описаны с использованием предложенной формализации. И наоборот, методы решения задач при помощи ИНС зачастую оказываются лишь переписанными на другой язык формализации хорошо известными ранее математическими алгоритмами.

2.2 Многослойный перцептрон (МСП)

Рассмотрим наиболее распространенную нейросетевую архитектуру – многослойный перцептрон. Основная идея МСП состоит в расположении формальных нейронов слоями. Наименьшее число слоев три: входной, скрытый и выходной. Такой МСП изображен на Рис. 8. Число нейронов во входном и выходном слоях зависит от размерности данных. Следует отметить, что входной слой состоит не совсем из формальных нейронов. Скорее, это просто точки ветвления, распределяющие входной сигнал на входы нейронов скрытого слоя. Иногда входной слой трактуют как препроцессор входных данных (например, линейное преобразование данных к интервалу [-1; 1] или любая другая нормировка). Аналогично и нейроны выходного слоя, исходя из решаемой задачи, могут не содержать нелинейного преобразователя. Число скрытых слоев и количество нейронов в каждом из них может быть любым и зависит от сложности решаемой задачи. Число входов каждого нейрона (сумматора) равно числу нейронов предыдущего слоя плюс один постоянный вход (*bias*). Точка ветвления (выход) каждого нейрона распределяет сигнал по всем входам нейронов последующего слоя.

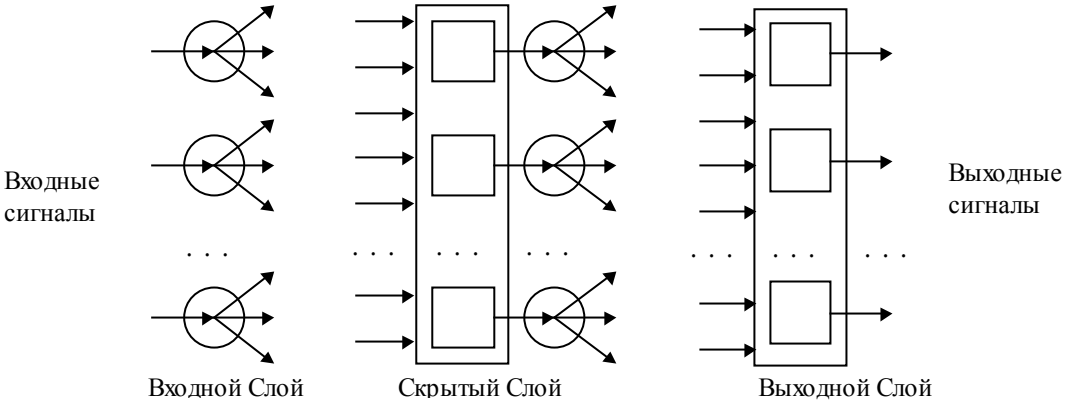


Рис. 8 Многослойный перцептрон

Наличие хотя бы одного скрытого слоя оказывается принципиально важным. Наряду с наличием нелинейного элемента, это позволяет решать *любую* линейно-неразделимую задачу. Другими словами, в отличие от однослойного (без скрытых слоев) перцептрона, МСП может решить любую линейно-неразделимую задачу. Таким образом, МСП является универсальным апроксиматором. Доказательство этого утверждения следует из основной теоремы нейронных сетей – **теоремы Стоуна:**

Пусть $\boldsymbol{E \subseteq C(X)}$ *– замкнутое линейное подпространство в* $\boldsymbol{C(X)}$, $\boldsymbol{1 \in E}$, *функции из* $\boldsymbol{E}$ *разделяют точки в* $\boldsymbol{X}$ *(т.е. для любых* $\boldsymbol{x,y \in X}$ *существует такая функция* $\boldsymbol{g \in E}$, *что* $\boldsymbol{g(x) \neq g(y)}$*) и* $\boldsymbol{E}$ *замкнуто относительно нелинейной унарной операции* $\boldsymbol{f \in C(X)}$. *Тогда* $\boldsymbol{E = C(X)}$.

Ее строгое доказательство можно найти в [3]. Ее суть состоит в том, что *любую* непрерывную функцию любого числа аргументов можно приблизить с *любой* наперед заданной точностью при помощи суперпозиции *произвольных нелинейных* функций одного аргумента. Причем достаточно всего лишь *одной* нелинейной функции. По сути, МСП и является такой суперпозицией. Желаемая точность аппроксимации определяется числом скрытых слоев и числом нейронов в них. В связи с этим следует отметить, что выбор числа скрытых слоев и числа нейронов в них, является отдельной сложной проблемой. Для ее решения, в зависимости от постановки задачи, используют различные подходы ([1], [2], [3], [5]).

3 Оптимизация градиентными методами

3.1 Нахождение градиента целевой функцией методом обратного распространения ошибки

Основой любого градиентного метода оптимизации является нахождение градиента целевой функции по оптимизируемым параметрам. МСП можно представить как нелинейный преобразователь (оператор) из пространства весов МСП (вектор такого пространства имеет координаты, численно равные подстраиваемым весам) в пространство значений целевой функции. Другими словами, каждому набору весов ставится в соответствие некоторое число. В качестве целевой может использоваться любая функция, которая зависит от ошибки модели. Наиболее часто используется среднее квадратичное отклонения (MSE) желаемого значения от выдаваемого моделью (МСП):

$$E = \frac{1}{N}\sum_{i=1}^{N}(t_i - o_i)^2 \qquad (1)$$

где $\boldsymbol{E}$ – среднее квадратичное отклонение (MSE),

$\boldsymbol{N}$ – число обучающих примеров,

$\boldsymbol{t}$ – выход сети,

$\boldsymbol{o}$ – желаемое (эталонное) значение.

Таким образом, градиент целевой функции $\boldsymbol{E}$ по весам $\boldsymbol{W}$ между выходным и последним скрытым слоями равен:

$$\frac{\partial E}{\partial W_{ij}} = -o_i f'(net_j)(t_i - o_i) \quad (2)$$

где ***i*** – индекс нейрона из скрытого слоя,

j – индекс нейрона из выходного слоя,

W – веса связей соответствующих нейронов,

o и ***t*** – выходы (желаемые и выдаваемые сетью) соответствующих нейронов,

net – вход (взвешенная сумма) соответствующего выходного нейрона,

$f'()$ – производная активационной функции, для сигмоида: $f(x) = \frac{1}{1+e^{-x}}$

она имеет вид: $f'(x) = f(x)(1 - f(x))$.

Для выходного слоя проблемы нет, т.к. нам известно желаемое значение, а значит и ошибка модели. С предыдущими слоями сложнее. Для расчета градиента по другим слоям, разобьем формулу (2) на две части:

$$\delta_j = f'(net_j)(t_i - o_i) \quad (3)$$

$$\frac{\partial E}{\partial W_{ij}} = -o_i \delta_j \quad (4)$$

Идея состоит в том, что для предыдущих слоев в качестве ошибки используют взвешенную сумму ошибок предыдущих слоев:

$$\delta_j = f'(net_j) \sum_k (\delta_k w_{kj}) \quad (5)$$

где ***k*** – индекс нейрона следующего слоя.

Формула (4) остается неизменной для расчета частных производных любого слоя. Для расчета ошибки δ выходного слоя используется формула (3), для скрытых слоев, независимо от их количества, – (5). Таким образом, мы получили рекурсивный алгоритм для расчета градиента МСП любой структуры. Этот алгоритм получил название алгоритма обратного распространения ошибки (*back propagation error*). Он был впервые предложен в 1986 г. Руммельхартом и Хинтоном (Rummelhart D.E., Hinton G.E.). Следует отметить, что именно этот метод дал новую жизнь нейронным сетям и послужил стартом их бурного развития. Более подробные математические выкладки можно найти в [1], [2], [3], программную реализацию на языке программирования C++ в [5], [6].

Итак, градиент найден. Для нахождения минимума MSE можно использовать простейший метод градиентного спуска:

$$W_{ij}(t+1) = W_{ij}(t) - h \cdot \frac{\partial E}{\partial W_{ij}} \quad (6)$$

где ***h*** – скорость спуска (*learning rate*), постоянное число от 0 до 1.

Чем больше ***h***, тем выше скорость сходимости на первоначальной стадии. По мере приближения к решению, больше ***h*** снижает скорость сходимости и, кроме того, процесс может не сойтись вообще. При изначальном малом ***h*** процесс сойдется, но это займет непозволительно много времени. Общая, ”генетическая” болезнь любого градиентного метода, это медленная сходимость при попадании на ”склон” вытянутой ”долины” поверхности целевой функции. В этом случае направление по антиградиенту не является оптимальным. Для корректировки такого направления используют еще один параметр ”инерционности” μ (*momentum term*), также постоянное число от 0 до 1. Формула (6) модифицируется:

$$W_{ij}(t+1) = W_{ij}(t) - h \cdot ((1-\mu) \cdot \frac{\partial E}{\partial W_{ij}}(t) + \mu \cdot \frac{\partial E}{\partial W_{ij}}(t-1)) \quad (7)$$

Направление антиградиента складывается с предыдущим направлением спуска с весом μ. Это позволяет ускорить сходимость, но проблема выбора μ аналогична проблеме выбора $\boldsymbol{h}$. Долгое время пытались предложить алгоритмы автоматической адаптации этих параметров, но большого успеха они не имели. Дело в том, что тренировка велась в режиме "по одному примеру" (*on-line*). Другими словами, обновление весов происходило после подачи *каждого* примера из обучающей выборки. Поэтому выбрать оптимальные μ и $\boldsymbol{h}$ невозможно. Например, для выбора $\boldsymbol{h}$ можно можно было бы применить одномерную оптимизацию вдоль направления антиградиента, которое дает значительное улучшение на одном примере. Но этот механизм часто приводит к потере навыка решения на предыдущих примерах. Решение этой проблемы простое: производить очередной шаг после расчета градиента на *всей* обучающей выборке (*batch mode*). Градиенты на отдельных примерах просто складываются. Эффективность такого подхода неоднократно доказана, например в [1], [2], [3]. Поэтому в дальнейшем мы будем рассматривать лишь этот подход.

Итак, мы имеем направление спуска (антиградиент) и размер шага (результат одномерной минимизации вдоль этого направления). Построенный на этом алгоритм работает достаточно хорошо. Но мы решили только проблему адаптации параметра $\boldsymbol{h}$. Направление антиградиента хороший выбор, но не является идеальным. Поэтому необходима его корректировки (параметр μ). Из теории оптимизации известно, что если матрица вторых производных оценки D_2 положительно определена, наилучшим считается ньютоновское направление:

$$-D_2^{-1} \frac{\partial E}{\partial W_{ij}} \quad (8)$$

Но здесь мы сталкиваемся с множеством проблем (время поиска матрицы D_2 и ее обращения, большой расход памяти для задач большой размерности, что делать, если D_2 не положительно определена и т.д.). Предлагается много так называемых квазиньютоновских методов, обходящих эти препятствия. Квазиньютоновские методы аппроксимируют каким-либо простым образом матрицы D_2 и D_2^{-1}. Об этом можно найти много из источников по теории оптимизации, неплохо об этих проблемах в применении к нейронным сетям изложено в [1].

3.2 Метод сопряженных градиентов

Остановимся лишь на наиболее популярном методе сопряженных градиентов (*conjugate gradients*). Ниже приведена основная формула этого метода:

$$\gamma = \frac{(c-g) \cdot c}{g^2} \quad (9)$$

где $\boldsymbol{c}$ – антиградиент ($\boldsymbol{g}$ на следующем шаге), $\boldsymbol{g}$ – направление спуска на текущем шаге, $\boldsymbol{\gamma}$ – множитель корректировки текущего направления.

Новое направление равно старому направлению, умноженному на $\boldsymbol{\gamma}$, плюс направление антиградиента. Таким образом, $\boldsymbol{\gamma}$ – это аналог параметра $\boldsymbol{\mu}$ в простейшей реализации градиентного спуска, которой не является постоянным и тем самым лучше выполняет свою задачу.

Следует особо отметить процедуру одномерной минимизации. Дело в том, что метод сопряженных градиентов эффективен только в том случае, если новый градиент ортогонален предыдущему направлению спуска. Это достигается аккуратным проведением одномерной минимизации. Поэтому в любой реализации метода сопряженных градиентов большую часть времени выполняется именно эта процедура. Это полностью оправдано и на это надо обращать особое внимания. Существует великое множество численных методов решения этой задачи (алгоритм Брента и пр.). В [2] и [5] приведены примеры реализации и подробное описание некоторых алгоритмов одномерной оптимизации.

Теперь обсудим еще один недостаток градиентных методов, а именно спуск только к ближайшему локальному минимуму. Строго говоря, решение задачи глобальной оптимизации (при наличии большого количества локальных экстремумов) не имеет однозначного решения и ни один из существующих алгоритмов не способен гарантировать нахождение глобального экстремума за приемлемое время. Особенно это касается нейронных сетей, где приходиться проводить оптимизацию по сотням параметров нелинейной модели. Но, с другой стороны, для получения качественной модели совсем нет необходимости в строгой точности модели по отношению к обучающей выборке. Более того, чаще всего строгая интерполяция является плохой моделью. Модель с MSE равной нулю, может совсем не соответствовать искомому распределению (оверфитинг). Конечно, было бы иметь идеально в любом случае иметь алгоритм, позволяющий надежно и сравнительно быстро находить точное решение. Бороться с оверфитингом можно различными способами: ограничение весов, регуляризация, уменьшение количества связей и др.

В связи с этим, рассмотрим несколько способов, позволяющих увеличить вероятность нахождения глобального решения. Общая идея проста: каким либо ”грубым” методом передать градиентному методу начальную точку вблизи локального минимума. Градиентный метод быстро сойдется к этому локальному минимуму. Далее процесс повторяется, чем больше, тем лучше. Таким образом, чем большее пространство мы сможем покрыть, тем больше вероятность ”наткнуться” на глобальный минимум. В качестве таких ”грубых” методов можно предложить методы отжига и генетического поиска. В силу своей универсальности (применяются для оптимизации любой функции), они не могут использоваться для обучения МСП напрямую (скорость сходимости будет катастрофически мала) и не могут конкурировать с градиентными методами, работающими только с дифференцируемыми функциями. Поэтому этим, неградиентным методам отводится подчиненная роль. Хотя их дальнейшее совершенствование может изменить ситуацию.

3.3 Метод нелинейной оптимизации Левенберга-Маркара

Рассмотрим теперь еще один популярный гибридный метод нелинейной оптимизации Левенберга-Маркара (*Levenberg-Marquardt*) в применении к обучению МСП и то, как он решает обозначенные выше проблемы. Вернемся к формулам (6) и (8), которые обсуждались ранее:

$$W_{ij}(t+1) = W_{ij}(t) - h \cdot \frac{\partial E}{\partial W_{ij}} \tag{10}$$

$$W_{ij}(t+1) = W_{ij}(t) - D_2^{-1} \frac{\partial E}{\partial W_{ij}} \tag{11}$$

Теперь внимательно посмотрим на следующую формулу:

$$W_{ij}(t+1) = W_{ij}(t) - (D_2 + hI)^{-1} \frac{\partial E}{\partial W_{ij}} \tag{12}$$

где, как и прежде, D_2 – матрица вторых производных (Гессиан) целевой функции E, h – константа, I – единичная матрица, $\frac{\partial E}{\partial W_{ij}}$ – градиент целевой функции E по весам W. В качестве целевой функции будем рассматривать среднее квадратичное отклонение (формула (1)).

Нетрудно заметить, что если константа h велика, то эта формула сводится к формуле (10), если мала, то к формуле (11). В этом и есть основная идея алгоритма Левенберга-Маркара.

Рассмотрим работу алгоритма подробнее. На первом шаге мы рассчитываем Гессиан D_2 и выбираем Ньютоновское направление спуска (h мало). Если ошибка уменьшается (на первом шаге это наверняка будет так), то h тоже уменьшается. Далее, в последующих шагах, h регулируется в зависимости от успеха на предыдущем шаге. Другими словами, получаем чередование больших шагов по результатам спуска по Ньютоновскому направлению и малых шагов по направлению градиентного спуска. Таким образом, мы частично решаем проблему зацикливания в локальном минимуме. Дело в том, что Ньютоновское направление идеально для квадратичных форм. Целевая же функция в области локального минимума не является идеальной квадратичной формой. Кроме того, локальных минимумов много, обращения Гессиана неиде-

ально. Другими словами, недостатки начинают играть положительную роль. Таким образом, большие шаги можно рассматривать как предоставление новой стартовой точки для градиентного спуска с малым шагом.

Конечно, конкретная реализация этого алгоритма сложна и изобилует всевозможными “трюками” и подводными камнями.

Итак, мы каким-то образом, конечно не идеально, но решаем все проблемы глобальной оптимизации. Остается, однако, сложная проблема обращения D_2 и связанные с этим расходы памяти. Но теперь не будет трагедией, если обращение D_2 даст плохой результат, он просто будет игнорироваться и не повлияет значительно на общую сходимость алгоритма.

На практике, алгоритм Левенберга-Маркара работает практически всегда намного эффективнее всех других методов оптимизации, применяемых для обучения МСП. Однако, как всегда бывает в жизни, есть существенные ограничения на его применение.

Во-первых, применим только для сравнительно небольших сетей (число весов ***N*** порядка 300-400, размерность D_2 равна ***N*** x ***N***).

Во-вторых, в качестве целевой функции может использоваться только среднее квадратичное отклонение (формула (1)).

В-третьих, не допускается никакой регуляризации весов, будь-то примитивное ограничение интервала их изменения или более тонкая через регуляризирующие операторы (например. по Вигенду (*Weigend*)).

В-четвертых, неэффективен в случае числа выходов МСП более одного.

Третье ограничение очевидно. Обращение Гессиана – это само по себе сложная проблема, и далеко не всегда дает хорошие результаты. Если же мы будем еще и накладывать какие-либо ограничения на веса, алгоритм может вообще выродиться в обычный градиентный спуск, причем не самый оптимальный. Поэтому в результате применения алгоритма Левенберга-Маркара часто получаются сети с очень большими разбросами весов, но очень малым значением целевой функции. Это практически всегда оверфитинг. Но этот недостаток может принести и пользу – распределение весов может стать дополнительным критерием оценки качества построенной модели. Распределение весов можно считать хорошим, если оно стандартное (среднее равно нулю) нормальное с малым сигма и общий интервал находится в разумных пределах. Большие по модулю веса находятся в “хвостах” колокола. Малая сигма означает, что доля этих весов в общем распределении невелика.

Относительно последнего ограничения, справедливости ради следует заметить, что при увеличении числа выходов *любой* метод будет испытывать значительные затруднения. Дело в том, что оценка качества сложной модели одним числом (значение целевой функции) является неблагодарной задачей. При увеличении же размерности выходов эта проблема еще более усугубляется. Обычно в качестве оценки модели берут среднее арифметическое значение оценки по всем выходам. Можно использовать выпуклую комбинацию ошибок по всем выходам (с соответствующей корректировкой расчета градиента оценки по весам). В этом случае ошибки по разным выходам будут неравнозначны. Но в любом случае, оценка модели останется всего лишь одним числом. Отсюда и проблемы оверфитинга и пр. Поэтому увеличение числа выходов даже на единицу, значительно усложняет процесс построения модели. Конечно, сложность задачи зависит от того, насколько разные выходы коррелированны между собой.

Есть еще один недостаток, правда не принципиальный. Дело в том, что минимум целевой функции достигается за небольшое число итераций. Кроме того, в отличии от метода сопряженных градиентов, после каждой итерации ошибка не всегда минимальна, не после каждого шага модель улучшается. Это создает трудности в применении тестировочного набора данных для оценки качества модели (способности к обобщению). Таким образом, явное преимущество метода (быстрая сходимость) оказывается недостатком. Но, на самом деле, лучше быстро построить несколько моделей и выбрать лучшую, чем долго и упорно тренировать сеть более медленным алгоритмом и не иметь при этом гарантий, что в результате получится хороший результат.

Более детальное обсуждение по применению алгоритма Левенберга-Маркара в МСП, математические выкладки и реализацию на языке программирования C++ можно найти в [5].

3.1 Сингулярное разложение матрицы

Ключевым и наиболее сложным и трудоемким этапом алгоритма Левенберга-Маркара является обращение матрицы вторых производных (Гессиан).

Приведем кратко одну из численных схем обращения матрицы, точнее, решения системы линейных уравнений вида:

$$Ax = b \tag{13}$$

где число уравнений ***n*** может быть больше, чем число неизвестных параметров ***d***. Это метод сингулярного разложения матрицы A (*singular-value decomposition, SVD*). Идея состоит в том, чтобы представить матрицу A в виде:

$$A = U\ W\ V^T \tag{14}$$

где колонки матрицы U – (***n*** x ***n***) это собственные вектора (ортогональный базис) матрицы AA^T,

колонки матрицы V – (***d*** x ***d***) это собственные вектора (ортогональный базис) матрицы $A^T A$,

W – (***n*** x ***d***) диагональная матрица, где первые ***r*** (ранг матрицы A) значений по диагонали – значения квадратных корней от собственных чисел матриц AA^T и $A^T A$ (обе матрицы имеют одинаковые не равные нулю собственные значения).

Решение системы (13) будет иметь вид:

$$x = V\ W^{-1} U^T b \tag{15}$$

где $V\ W^{-1} U^T = A^{-1}$

В общем случае: $A^k = V\ W^k U^T$

Более детальные математические выкладки и реализацию на языке программирования С можно найти в [6], конкретные реализации для МСП – в [4], [5].

4 Оптимизация методом имитации отжига

4.1 Основная идея метода

Термин “отжиг” (*annealing*) пришел в математику из металлургии. При высокой температуре, металл легко деформируется и меняет свою форму. Понятие температуры связано со скоростью атомов, которые хаотически передвигаются. Чем выше скорость движения атомов, тем выше температура и тем легче металл поддается деформации. Таким образом, если резко понизить температуру металла, то атомы резко “замрут” и мы получим твердую, но очень неровную поверхность. Если стоит задача получить максимально ровную поверхность, то необходимо сначала сильно разогреть металл, а потом постепенно, очень медленно охлаждать его до полного затвердевания. Таким образом, у каждого атома будет достаточно времени для того, чтобы найти свое “лучшее” место среди других атомов. Чем равномернее будут распределены атомы вдоль поверхности, тем ровнее в итоге будет поверхность. С точки зрения математики, задачу можно сформулировать так: минимизировать среднюю квадратичную ошибку отклонения уровня поверхности металла от некоторого постоянного значения. Оптимизируемыми параметрами в этом случае будут координаты положения атомов.

Рассмотрим применение метода отжига (*simulated annealing*) для МСП. Под температурой будем понимать стандартное отклонение одной реализации МСП (набор весов) от другой. Таким образом, чем выше температура, тем дальше отстоят друг от друга эти реализации. Если все возможные состояния сети (набо-

ры весов) определить как пространство векторов, координатами которых являются соответствующие веса, то понятия “далеко” или “близко” можно оценить через норму разности этих векторов. Следует отметить, что далеко отстоящие друг от друга вектора могут иметь близкие значения с точки зрения целевой функции. Понятие времени нахождения на одном значении тнмпературы (скорость ее снижения) можно трактовать как количество векторов, которые генерируются с заданной температурой.

Алгоритм работает следующим образом. Задаются начальное и конечное значения температуры. Для того, чтобы покрыть все пространство векторов, начальное значение должно быть достаточно большим. С другой стороны, значения весов должны быть разумными и не очень большими. Слишком большая начальная температура может сильно замедлить процесс сходимости. Для нахождения промежуточных значений используются различные формулы, которые обсудим позднее. Также задается время поиска при одном значении температуры (количество случайных реализаций). Для каждой реализации рассчитывается значение целевой функции. Вектор с лучшей оценкой становится центром для следующей, более низкой температуры. Процесс повторяется для нового значения температуры. Чем ниже температура, тем ближе друг от друга будут случайные реализации. Таким образом, алгоритм сходиться к решению.

Для нахождения вектора, отстоящего от заданного центра на заданную температуру, используют простой алгоритм. Берутся несколько равномерно-распределенных случайных чисел. Чем больше чисел, тем более ярко будет выражено нормальное распределение для их суммы. Обычно берется четыре числа. Если от суммы двух отнять сумму двух других, то получим случайное число со средним ноль. Вариация равномерного распределения на отрезке 0-1 равна 1/12, среднее 0.5. Добавление четырех чисел увеличивает вариацию в 4 раза. Для получения случайной величины, чье стандартное отклонение равно заданной температуре, надо умножить эту температуру на $\sqrt{\frac{12}{4}}$ и разделить на интервал возможного изменения случайного числа. Обычно генератор случайных чисел выдает числа в интервале от нуля, следовательно делить следует на максимально возможное число, которое может быть выдано генератором.

Для расчета промежуточных значений температуры, обычно применяется следующая формула:

$$k = e^{\frac{\ln(stop/start)}{n-1}} \qquad (1)$$

где ***start*** и ***stop*** – начальное и конечное значения температуры,
n – количество температур,
k – множитель для расчета последующих значений температуры.

Следует особо отметить, что для конкретной реализации метода отжига очень важен используемый генератор равномерно-распределенных случайных чисел. Его период повторяемости должен быть очень большим. В частности, встроенные в программную среду стандартные генераторы могут не удовлетворять требованиям метода отжига. Реализацию таких генераторов можно найти во многих источниках по численным методам, например в [7]. В [5] и [6] также приведен пример реализации такого генератора, основанного на линейном конгруэнтном алгоритме *(linear congruential algorithm*). Этот алгоритм использовался также для проведения примера исследования в данной работе.

Описанный выше алгоритм является простейшим. Строгие математические обоснования и практические реализации приведены в [8].

4.2 Усовершенствования метода имитации отжига

Описанный выше алгоритм является простейшим. Более сложные реализации метода можно найти в [12], строгие математические обоснования и практические реализации – в [9]. В [10] и [11] обсуждаются проблемы оптимизации в широком смысле с применением метода отжига. Пример реализации на языке программирования C++ приведен в [5], [6]. Отметим лишь некоторые направления по расширению возможностей метода отжига.

Во-первых, описанный выше пример реализации, улучшения принимаются детерменистически. Другими словами, если оценка очередной реализации лучше, чем считается на данный момент, новая реализация принимается в качестве лучшей безоговорочно (с вероятностью 1). Такой подход ускоряет сходимость, но увеличивается опасность зацикливания в локальном минимуме. При стохастическом подходе улучшения принимаются с некоторой, не равной 1, вероятностью. Кроме того, такая вероятность может

варьироваться в зависимости от температуры. Стохастический подход к проблеме принятия улучшений позволяет избегать зацикливания в локальном минимуме.

Во-вторых, часто применяются более сложные критерии перехода к новому значению температуры, чем описано выше. Простое усложнение – это введение дополнительного счетчика в цикл внутри температур. При каждом улучшении основной счетчик сбрасывается в ноль или на определенное число шагов назад. Таким образом, время нахождения на одном значении температуры варьируется и повышается вероятность того, что при данном значении температуры мы действительно получим лучший результат. Более сложный подход, это когда алгоритм собирает информацию об “энергетической поверхности” (проще говоря, о минимизируемой функции). На основе этой информации принимается решение о переходе на следующее значение температуры или вообще об окончании процесса.

В-третьих, подход к расчету промежуточных значений температуры. Например, применять не экспоненциальную зависимость, а логарифмическую и т.д.

В-четвертых, применять не нормальное распределение при получении различных реализаций, а, например, распределение Коши. В [12] утверждается, что в этом случае удается ускорить снижение температуры и получать хорошие результаты.

Примером таких более сложных реализаций является, например, метод *быстрого отжига (fast simulated annealing*), предложенный Сзу (*Szu*) в [11], [12]. В этом методе используется так называемое быстрое снижение температуры *(fast temperature reducing).* Множитель для расчета температуры в этом случае имеет вид:

$$k = \frac{\text{Start\_Temperature} - \text{Stop\_Temperature}}{\text{Stop\_Temperature} * (\text{Number\_of\_Temperatures} - 1)} \qquad (2)$$

Далее каждое последующее значение температуры рассчитывается как:

$$\text{New\_Temperature} = \frac{\text{Start\_Temperature}}{1.0 + k * i} \qquad (3)$$

где **i** – номер нового значения температуры.

На Рис. 1 приведен график снижения температуры с 1 до 10 с общим числом промежуточных значений температур 100 по формулам (2) и (3) в сравнении со стандартным методом (1).

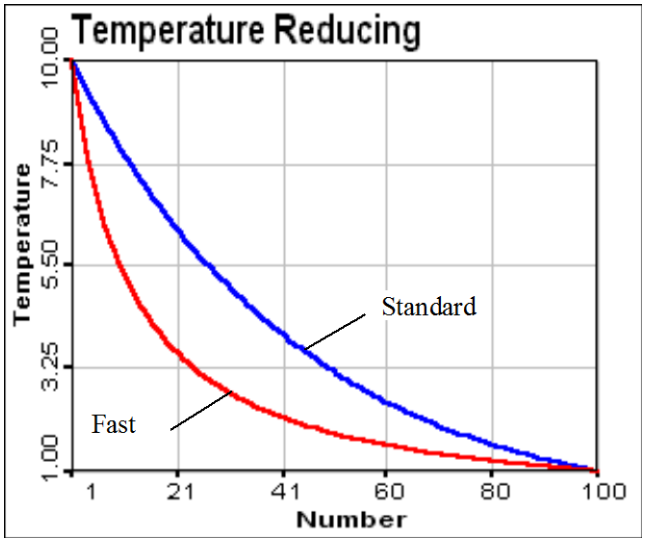


Рис. 1. Снижение температуры быстрое и стандартное

При использовании быстрого снижения температуры, для получения случайной реализации применяется распределение Коши. Важным отличием метода быстрого отжига является расчет вероятности принятия новой точки в качестве текущего центра. Стандартный метод отжига использует критерий Метрополиса (*Metropolis*) (4), быстрый отжиг – критерий Сзу (*Szu*) (5):

$$Metropolis: \quad P_{accept} = \min\left(1, \quad e^{\frac{f_j - f_i}{cT}}\right) \tag{4}$$

$$Szu: \quad P_{accept} = \frac{1}{1 + e^{\frac{f_i - f_j}{cT}}} \tag{5}$$

где $\boldsymbol{c}$ – константа, задаваемая пользователем (обычно 0.8),

$\boldsymbol{T}$ – текущая температура,

f_i – значение целевой функции в текущем центре,

f_j – значение целевой функции в новой точке.

Отличие этих подходов наглядно показано на Рис. 2.

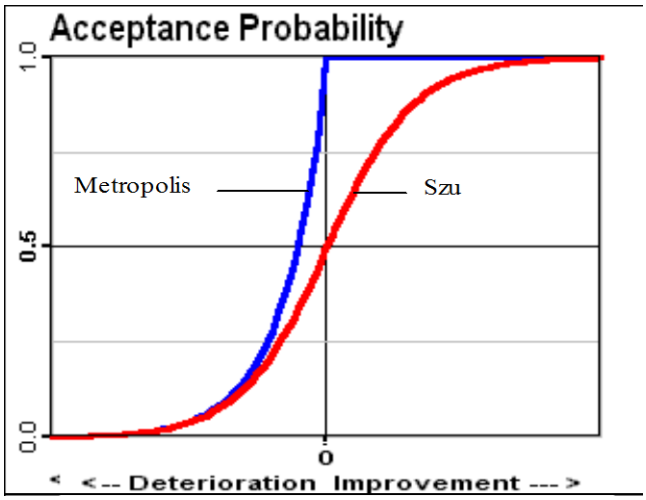


Рис. 2. Вероятность принятия текущей точки в качестве нового центра по критериям Метрополиса и Сзу

Справа от нуля показана вероятность принятия текущей точки в качестве нового центра в случае уменьшения значения целевой функции, слева – в случае увеличения. Таким образом, критерий Метрополиса почти детерминистический, в случае увеличения значения целевой функции вероятность быстро стремится к нулю. В случае же уменьшения значения целевой функции, новая точка принимается безоговорочно. Критерий Сзу является стопроцентно стохастическим, изменения текущего ценра происходят реже. Для изменения крутизны графиков, в обоих случаях в формулах (4) и (5) используется константа $\boldsymbol{c}$ (обычно 0.8).

Исходя из вышесказанного, рекомендуется в случае быстрого отжига брать побольше промежуточных значений температуры и поменьше количество итераций на каждое промежуточное значение температуры. Для стандартного отжига все наоборот.

5 Генетический метод оптимизации

Успех в развитии нейронных сетей не в последнюю очередь обусловлен глубокими биологическими основаниями, заложенными во многих архитектурах. Некоторые особенности биологической эволюции на уровне механизма кодирования и наследования в ДНК, легли в основу так называемых *генетических алгоритмов*, предложенных в начале 70-х годов (J.H.Holland, 1975) и получивших интенсивное развитие в последнее время. Хорошее описание генетических алгоритмов и области их применения можно найти в [13],

математическое обоснование в [14], программную реализацию на языке программирования C++ в применении к МСП – в [5].

Рассмотрим основные шаги генетического алгоритма оптимизации. В процессе эволюции выживает сильнейший, более приспособленный к жизни представитель популяции. Слабые представители чаще умирают до воспроизводства, сильные же живут дольше и имеют возможность оставлять больше потомков. Потомки унаследуют от родителей те свойства, которые помогли выжить их родителям. Генетические методы оптимизации работают аналогичным образом. Параметры оптимизируемой (целевой) функции кодируются как *гены* в *хромосоме*. Начальный набор представителей (параметров целевой функции) образует стартовый набор *хромосом* (*популяция*). Он формируется случайным образом. Далее выбираются пары представителей для воспроизводства, причем "лучшие" с точки зрения целевой функции представители имеют больше шансов быть выбранными для скрещивания. Полученные после скрещивания потомки унаследуют от родителей генетическую структуру и замещают родителей. Таким образом, "лучшие" представители имеют большее влияние на формирования последующего *поколения*. Далее процесс повторяется до тех пор, пока хотя бы один из представителей *поколения* не будет удовлетворять заданному критерию оптимизации. По аналогии с биологическими процессами вводится понятие *мутации,* которые препятствуют вырождению генофонда (помогают выйти из локального минимума целевой функции).

5.1 Основные термины генетического алгоритма

Рассмотрим подробнее терминологию с точки зрения МСП:

Хромосома *(chromosome)* — полное генетическое описание представителя, набор генов. Для МСП это один конкретный полный набор весов сети.

Ген *(gene)* — единичный, неделимый элемент хромосомы. Для МСП это один конкретный вес.

Аллель *(allele)* — область значений гена. Для МСП это разумный диапазон, в котором распределяются веса сети.

Популяция *(population)* — набор хромосом. Для МСП это набор по сути разных сетей одинаковой структуры с разными распределениями весов.

Поколение *(generation)* — популяция после очередного цикла генетического алгоритма.

Генотип *(genotype)* — двоичная генетическая структура хромосомы, т.е. закодированные определенным образом физические веса сети.

Фенотип *(phenotype)* — физическая интерпретация генотипа. Для МСП это набор весов сети.

Целевая функция *(objective function)* — функция для минимизации. Для МСП, например, это средняя квадратичная ошибка (MSE).

Фитнесс *(fitness)* — мера хромосомы, которая показывает насколько она "хороша" с точки зрения целевой функции. Вообще говоря, можно использовать в качестве такой меры значение функции, обратной целевой. Но в общем случае значение фитнесс не должно зависеть от выбора целевой функции и вычисляется по своим формулам, которые будут рассмотрены позднее.

Схемата *(schema, plural schemata)* — набор генов в хромосоме (шаблон с определенными значениями), действующих единым специальным образом при скрещивании. Схематы - мощный инструмент при функционировании генетического алгоритма, но для МСП существует проблема как их выделять и определять. Поэтому их использование для МСП в данной работе не рассматривается. Подробности можно найти в [14] и [5].

5.2 Функционирование генетического алгоритма

Рассмотрим подробнее работу генетического алгоритма с точки зрения МСП:

- ***Инициализация*** *(initialization)* — начальная популяция. Формируется случайным образом. Для полного покрытия пространства весов необходим большой разброс представителей (норма разности векторов координаты которых являются весами, должна быть большой, т.е. представители должны сильно отличаться друг от друга). Обычно размер начальной популяции берется больший, чем размер популяции, который будет использоваться в дальнейшем. "Худшие"с точки зрения целевой функции пред-

ставители удаляются. Таким образом, осуществляется первичный грубый отбор представителей для дальнейшей оптимизации.

- ***Эволюция*** *(evaluation)* — для каждого представителя, исходя из значения целевой функции, рассчитывается его фитнесс. Чем больше фитнесс, тем лучше данный представитель с точки зрения минимизации целевой функции.
- ***Выбор представителей для скрещивания*** *(parent selection)* — выбираются пары представителей из популяции для скрещивания, причем представители с большим значением фитнесс выбираются чаще других.
- ***Репродукция*** *(reproduction)* — в результате скрещивания каждой пары родителей получаются потомки, обычно два. Если в алгоритме заложен способ репродукции, при котором получаются более двух потомков, то для сохранения общего количества представителей в популяции, "худшие" представители удаляются.
- ***Мутация*** *(mutation)* — случайным образом меняется небольшое количество генетической информации в каждом потомке. Следует отметить, что мутация очень опасная операция! В природе она может породить мутантов, в математике мы можем получить несходящийся итерационный процесс. С другой стороны, она необходима для избежания вырождения генофонда (зацикливание в локальном минимуме целевой функции). Поэтому следует осторожно относиться к этому этапу. Обычно один или два бита меняют свое значение на противоположное. Мутация может происходить не на каждом цикле алгоритма, а согласно некоторой, обычно очень маленькой, вероятности.

Полный проход через эти этапы дает один цикл генетического алгоритма оптимизации, которые повторяются (кроме этапа *инициализации*) до тех пор, пока хотя бы один из представителей *поколения* не будет удовлетворять заданному критерию (значению целевой функции).

Рассмотрим подробнее основной этап эволюции. Для каждого представителя, он состоит из пяти шагов:

1. Преобразование генотипа хромосомы в ее фенотип.
2. Расчет значения целевой функции.
3. Расчет "сырого" значения фитнесс исходя из значения целевой функции.
4. Преобразование "сырого" значения фитнесс в нормированное.
5. Преобразование нормированного значения фитнесс в частоту выбора данного представителя в качестве родителя для скрещивания.

Несколько слов о преобразовании генотипа в фенотип и обратно. Дело в том, что веса сети – это количественная информация, для которой, в отличии от качественной, существует понятие близости значений. Например, число 8 близко по значению к 7. Но в стандартном двоичном представлении (1000 и 0111) эти числа не находятся рядом с точки зрения расстояния Хемминга. Расстояния Хемминга – это число различающихся в обоих векторах координат (0 или 1). Поэтому для генетических алгоритмов часто применяется другая кодировка десятичных чисел, в которой отсутствует данный недостаток. Пример такой кодировки – это код Грэя. Подробности об этой проблеме можно найти в [13]. Ниже приводится значения кода Грэя для чисел от 0 до 8 в сравнении со стандартной кодировкой.

Десятичное	двоичное	Грэя
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

8	1000	1100

5.3 Расчет фитнесс значения

Теперь обсудим проблему расчета фитнесс значения. Существует много различных способов его расчета. Главное, чтобы в итоге “лучшие” с точки зрения целевой функции представители выбирались для скрещивания чаще, чем другие. Но, с другой стороны, если такой представитель будет выбираться слишком часто, а другие вообще не будут участвовать в эволюции, то мы можем получить тот же эффект, что рассматривался при обсуждении мутации (вырождение генофонда, зацикливание в локальном минимуме).

Рассмотрим один способ расчета и применения фитнесс значения. Для упрощения расчетов, потребуем, чтобы фитнесс был неотрицательным. Этим требованиям удовлетворяет следующая функция:

$$f(x) = e^{Kv} \tag{1}$$

где **K –** отрицательное значение (константа), ***v*** – значение целевой функции.

Пусть в качестве целевой функции используется средняя квадратичная ошибка (MSE) с областью значений 0-1, тогда в качестве ***K*** можно использовать значение -20.

Таким образом, мы получили “сырое” значение фитнесс, которое необходимо отнормировать во избежания проблем, обозначенных выше. Обычно используется линейное преобразование, которое преобразует “сырое” значение фитнесс таким образом, чтобы его среднее значение оставалось неизменным, а максимальное отнормированное значение отличалось бы от среднего значения в фиксированное число раз (обычно в 2 или 3).

Ниже приведены формулы для такого преобразования:

$$f(x) = slope * x + const \tag{2}$$

$$slope = \frac{(mult - 1) * avg}{\max - avg} \tag{3}$$

$$const = \frac{avg(\max - mult * avg)}{\max - avg} \tag{4}$$

где ***f(x)*** – функция преобразования “сырого” значения фитнесс в нормированное,

mult – множитель, показывающие во сколько раз максимальное значение фитнееса должно превосходить его среднее значение (обычно 2 или 3),

avg и ***max*** – соответственно максимальное и среднее значение фитнесса.

Если в результате получаем отрицательное значение, используются следующие формулы:

$$slope = \frac{avg}{avg - \min} \tag{5}$$

$$const = \frac{-\min * avg}{avg - \min} \tag{6}$$

где ***min –*** минимальное значение фитнееса.

Далее все просто: для каждого представителя делим нормированное значение фитнесса на его среднее значение и получаем частоты выборки каждого представителя в качестве родителя для скрещивания. Легко заметить, что сумма этих фитнесс значений будет равна количеству представителей в популяции и относительные значения частот выборки пропорциональны фитнесс значениям.

После того, как выбраны пары родителей, производится их скрещивание. На Рис. 1 представлен пример скрещивания с разбиением хромосомы в 1-ой случайно выбранной точке *(one-point crossover*).

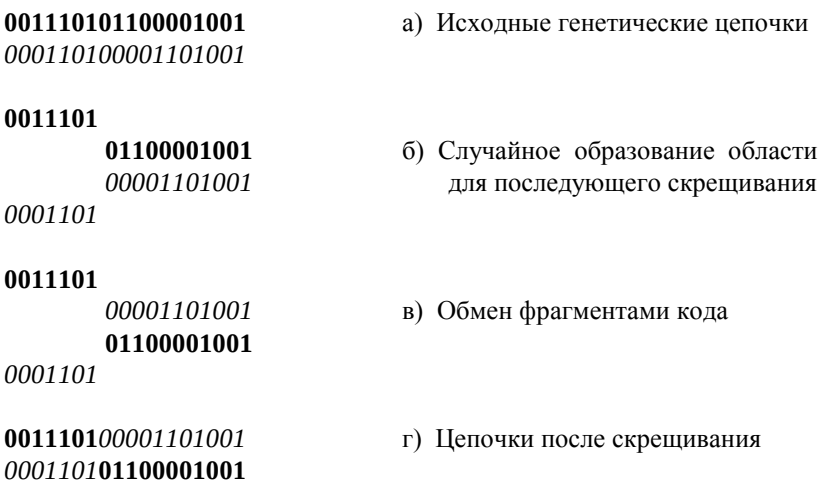


Рис. 1. Процесс скрещивания двух генетических цепочек разбиением в 1-ой точке

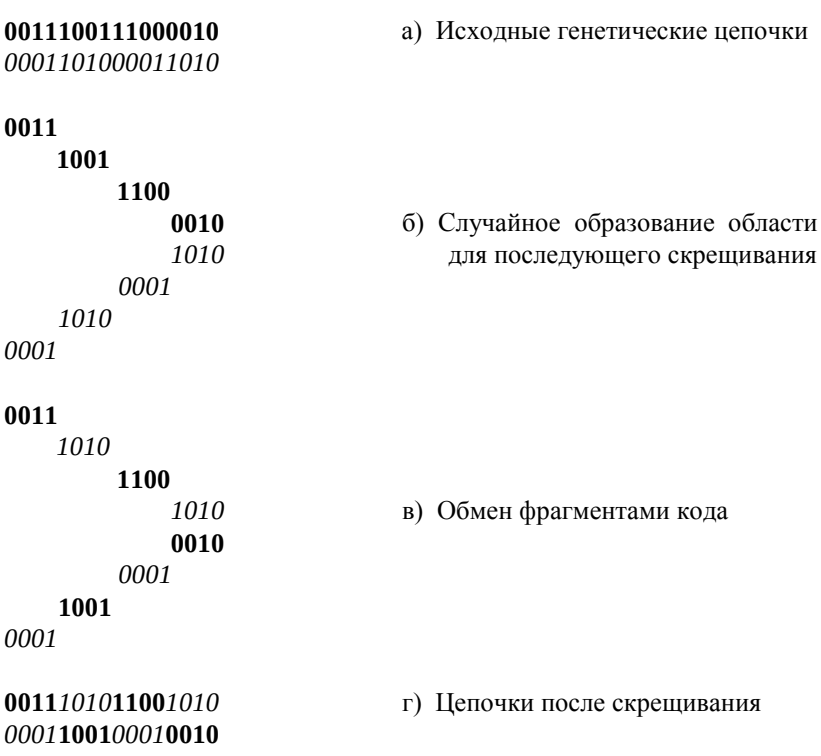


Рис. 2. Процесс скрещивания двух генетических цепочек разбиением в 2-х точках

Недостаток такого скрещивания заключается в том, что гены, расположенные рядом с друг другом, почти наверняка попадут в один и тот же потомок. С точки зрения МСП, это не совсем правильно, т.е. положение гена в хромосоме имеет значение. Дело в том, что веса не равнозначны между собой. Например, веса идущие от скрытого слоя к выходному слою не то же самое, что веса между входами и скрытым слоем. Кроме того, веса идущие от одного входа, отличаются от весов, идущих от других входов и т.д. Пусть, например, сеть имеет один скрытый слой с 4-мя нейронами. Тогда гены (веса) в хромосоме, относящиеся к 1,2,3 и 4 нейронам будут располагаться последовательно (один нейрон – четверть хромосомы). Для простоты не будем учитывать веса между скрытым и выходным слоями. Что же получается при скрещивании? Гены 1-го и 3-го нейронов **всегда** попадут в разные потомки. Аналогично для 2-го и 4-го нейронов. Таким образом, алгоритм эффекта не даст. Для решения этой проблемы можно использовать скрещивание с раз-

биением хромосомы в 2-х случайно выбранных точках *(two-point crossover*)**.** На Рис. 2 представлен пример такого скрещивания**.**

Конечно, и это не решает проблему. Теперь мы получаем, что четверти хромосомы будут принадлежать разным потомкам. Приемлемым решением может быть компромисс. Например, 2/3 всех скрещиваний производить разбиением в 1-ой точке, 1/3 – в 2-х точках. Подробности об этом можно найти в [3].

Для генетических алгоритмов существует еще один важный параметр: используется ли режим отбора элиты? Смысл его заключается в способе формирования очередного поколения. Если этот режим используется, то после формирования каждого поколения, не только многочисленные потомки лучшего представителя (хромосомы) переходят в следующее поколение. Сам лучший представитель (элита) переходит в следующее поколение **целиком**. Если нет, то в следующее поколение переходят только его потомки. Другими словами, лучший представитель **не** имеет особых привилегий (только высокий фитнесс). При рассмотрении этого режима встают те же проблемы, что и при вычислении фитнесс значения. Использование этого режима заметно ускоряет сходимость вблизи глобального минимума. Если же глобальный минимум не ярко выражен или мы находимся далеко от него, есть риск вырождения генофонда (зацикливания в локальном минимуме целевой функции).

6 Пример исследования методов тренировки МСП

Для исследования различных методов тренировки МСП использовался нейромодуль программного пакета GeostatOffice (Kanevski et al., 1999a). Исследовался МСП со структурой 2-10-10-1 (два входа, два скрытых слоя по десять нейронов в каждом слое, один выход). В качестве набора данных для обучения и тестирования использовались измерения загрязненности почвы радионуклидом ^{137}Cs в Брянской области [15]. Весь набор данных содержащий 665 точек, случайным образом разбит на тренировочный (500) и тестировочный (165) наборы (Рис. 1). Все тренировочные данные нормировались линейно на интервал (0;1). Тестировочные данные нормировались с такими же параметрами, что и тренировочные. Тренировка производилась без применения тестировочных данных, т.е. целью является изучение именно метода оптимизации, а не построение модели. Качество модели проверялось только после окончания тренировки на тренировочном наборе, т.е. досрочного оканчания тренировки по значению ошибки на тренировочном наборе не производилось.

Рассматривались следующие варианты тренировки:

1 Метод отжига (Рис. 2, справа).
2 Метод генетического поиска (Рис. 3, справа).
3 Тренировка методом Маркара-Левенберга (Рис. 4, слева). Случайная инициализация весов в интервале (-0.1; 0.1).
4 Тренировка методом Маркара-Левенберга (Рис. 4, слева). Инициализация весов через метод отжига (Рис. 2, слева).
5 Тренировка методом Маркара-Левенберга (Рис. 4, слева). Инициализация весов через метод генетического поиска (Рис. 3, справа).
6 Тренировка методом сопряженных градиентов (Рис. 4, справа). Случайная инициализация весов в интервале (-1; 1).
7 Тренировка методом сопряженных градиентов (Рис. 4, справа). Инициализация весов через метод отжига (Рис. 2, слева).
8 Тренировка методом сопряженных градиентов (Рис. 4, справа). Инициализация весов через метод генетического поиска (Рис. 3, справа).

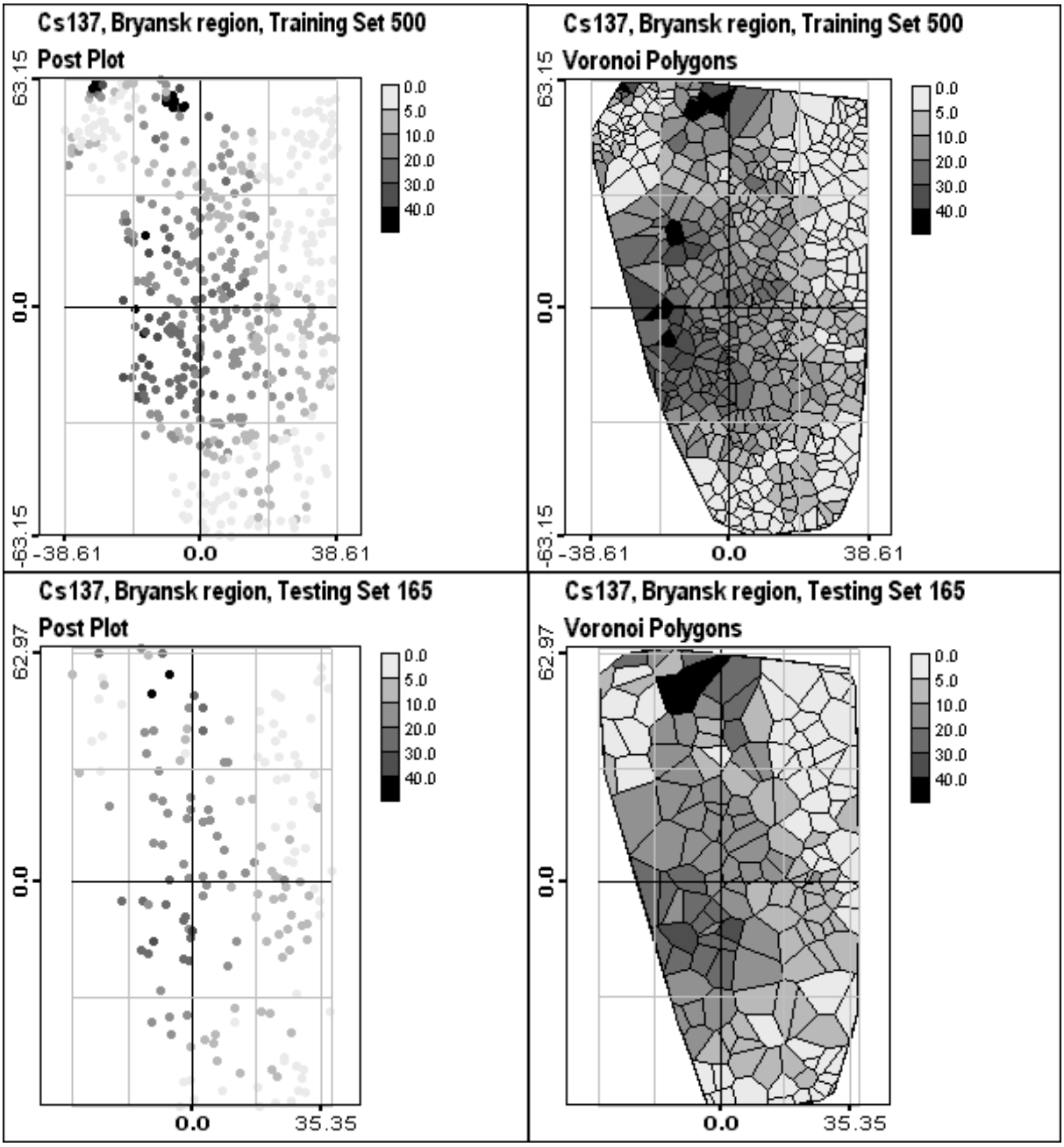


Рис. 1. Карты данных для тренировки и тестирования

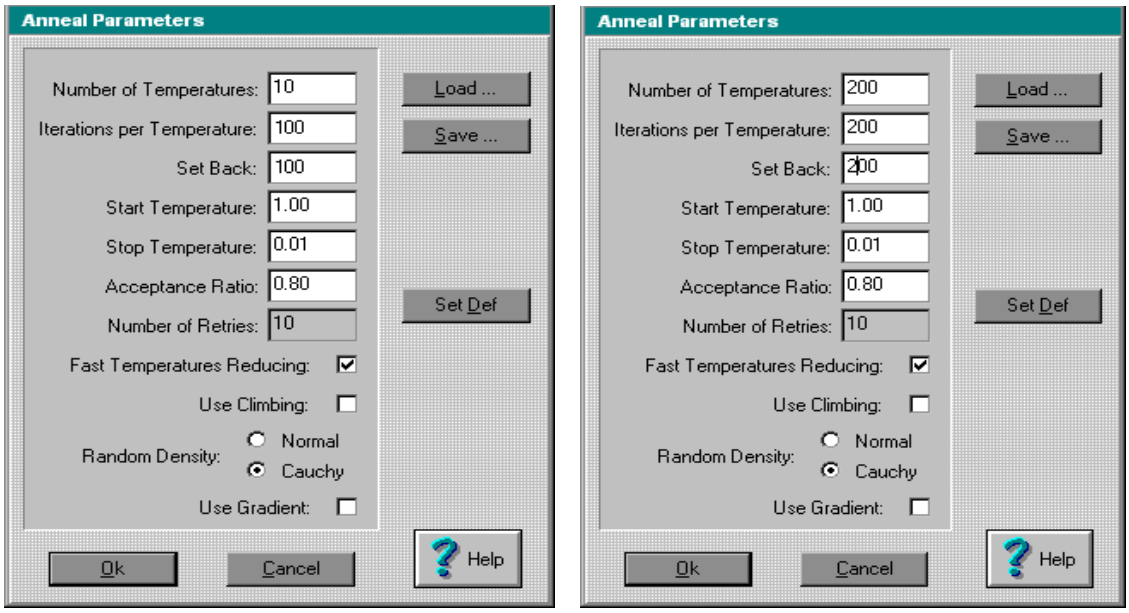


Рис. 2. Параметры для метода имитацией отжига

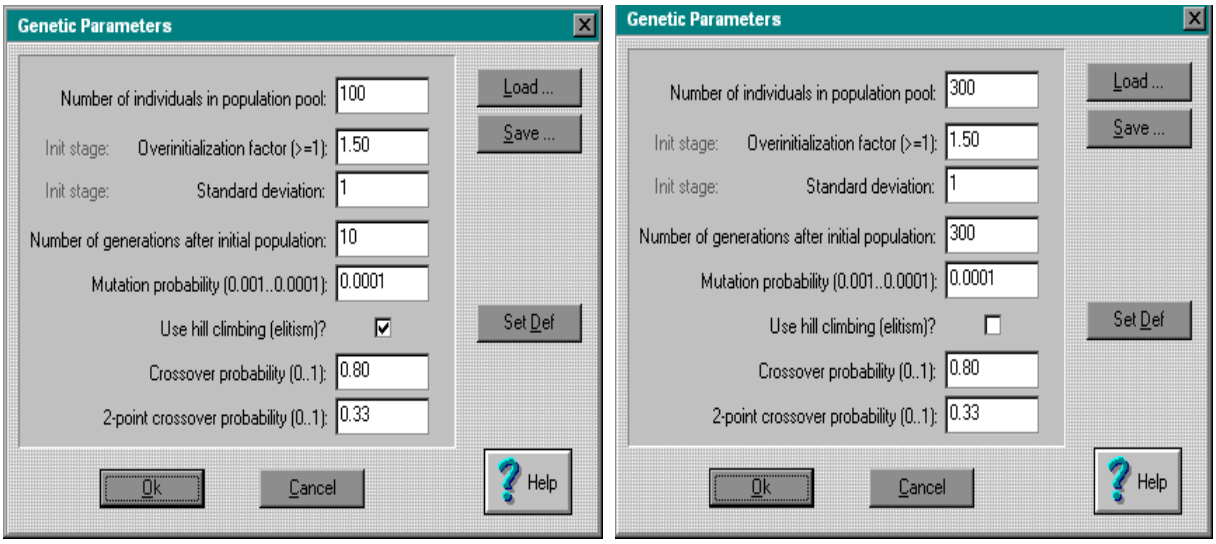


Рис. 3. Параметры для метода генетического поиска

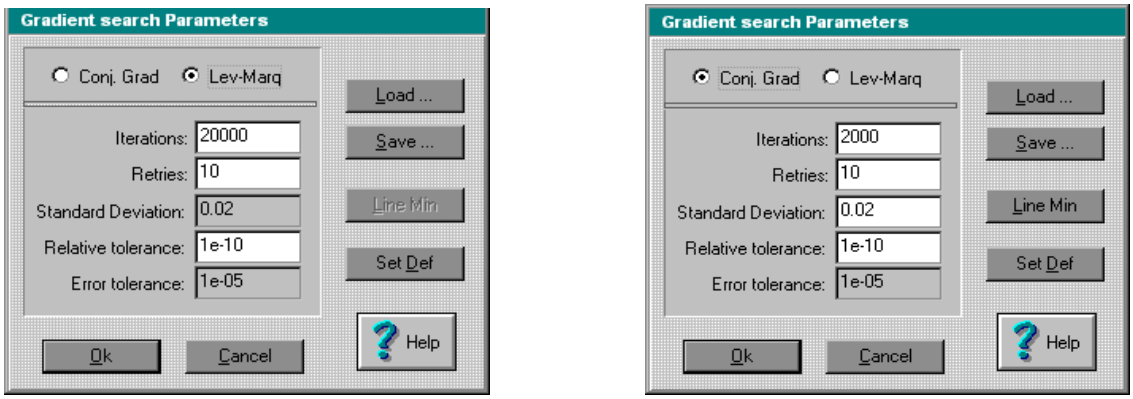


Рис. 4. Параметры для методов Левенберга-Маркара (слева) и сопряженных градиентов (справа)

Для варианта 1 использовался метод быстрого отжига (быстрое снижение температуры, распределение Коши, критерий Сзу) с количеством промежуточных значений температуры 200. Для вариантов 4 и 7, где отжиг использовался для инициализации весов, количество температур бралось равным 10.

Для варианта 2 использовался генетический метод с размером популяции 300 и количеством генераций 300, без режима отбора элиты. Для вариантов 5 и 8, где генетический поиск использовался для инициализации весов, количество генераций бралось 10, и количество генераций 10, с. режимом отбора элиты.

Метод Левенберга-Маркара (варианты 3, 4 и 5) не требует задания каких-то специальных параметров, только ограничение на число итераций.

В методе сопряженных градиентов (варианты 6, 7 и 8), кроме ограничения на число итераций, используется также стохастическая составляющая. При нахождении локального минимума, происходит рестарт по направлению наискорейшего спуска. Если это не дает положительного результата, то поиск нового направление происходит заданное число раз равным 10 в случайном направлении.

Полученные результаты представлены в Таблице 1 и на Рис. 5-9. Варианты 1 и 2 (отжиг и генетический метод), как и следовало ожидать, не могут конкурировать с градиентными методами. Для получения этих результатов было затрачено такое же или большее время, чем для тренировки градиентных методов. Таким образом, методы отжига и генетического поиска могут использоваться лишь для поиска стартовой точки для градиентных методов.

Лучшие с точки зрения ошибки на тренировочных данных результаты имеет метод Левенберга-Маркара. На тестировочных данных лучшие результаты у метода сопряженных градиентов. Следует отметить, что метод сопряженных градиентов более требователен к стартовой точке, т.к. он более подвержен зацикливанию в локальном минимуме. Случайная инициализация весов не всегда приводит к результату. При применении варианта 6 пришлось несколько раз начинать тренировку заново, т.к. метод отказывался работать вообще. Интервал инициализации пришлось увеличить. В результате у варианта 6 самые худшие из всех градиентных методов результаты. При применение же инициализации весов стохастическими методами, приемлемый результат достигается всегда. Метод Левенберга-Маркара практически не чувствителен к выбору стартовой точки, достаточно случайной инициализации. Но с другой стороны,. как отмечалось выше, результаты этого метода, несмотря на малую ошибку, не всегда приемлемы. Как видно из рисунков, варианты 3, 4 и 5 дали явный или близкий к оверфитингу результат. Модели вариантов 7 и 8 выглядит более приемлемо.

Таблица 1. Среднеквадратичные ошибки (**MSE**) и коэффициенты корреляции (ρ) различных вариантов тренировки

Вариант	**MSE Train Data Set**	**MSE Test Data Set**	**ρ, Train Data Set**	**ρ, Test Data Set**
1 Annealing	*48.8*	*71.7*	*0. 79*	*0.65*
2 Genetic	*41.4*	*63.3*	*0.83*	*0.70*
3 LM	**3.9**	48.9	**0.99**	0.77
4 LM+A	5.6	40.9	0.98	0.82
5 LM+G	5.5	41.3	0.97	0.81
6 CG	20.6	48.4	0.92	0.78
7 CG+A	12.3	24.9	0.95	**0.89**
8 CG+G	12.4	**24.6**	0.95	**0.89**

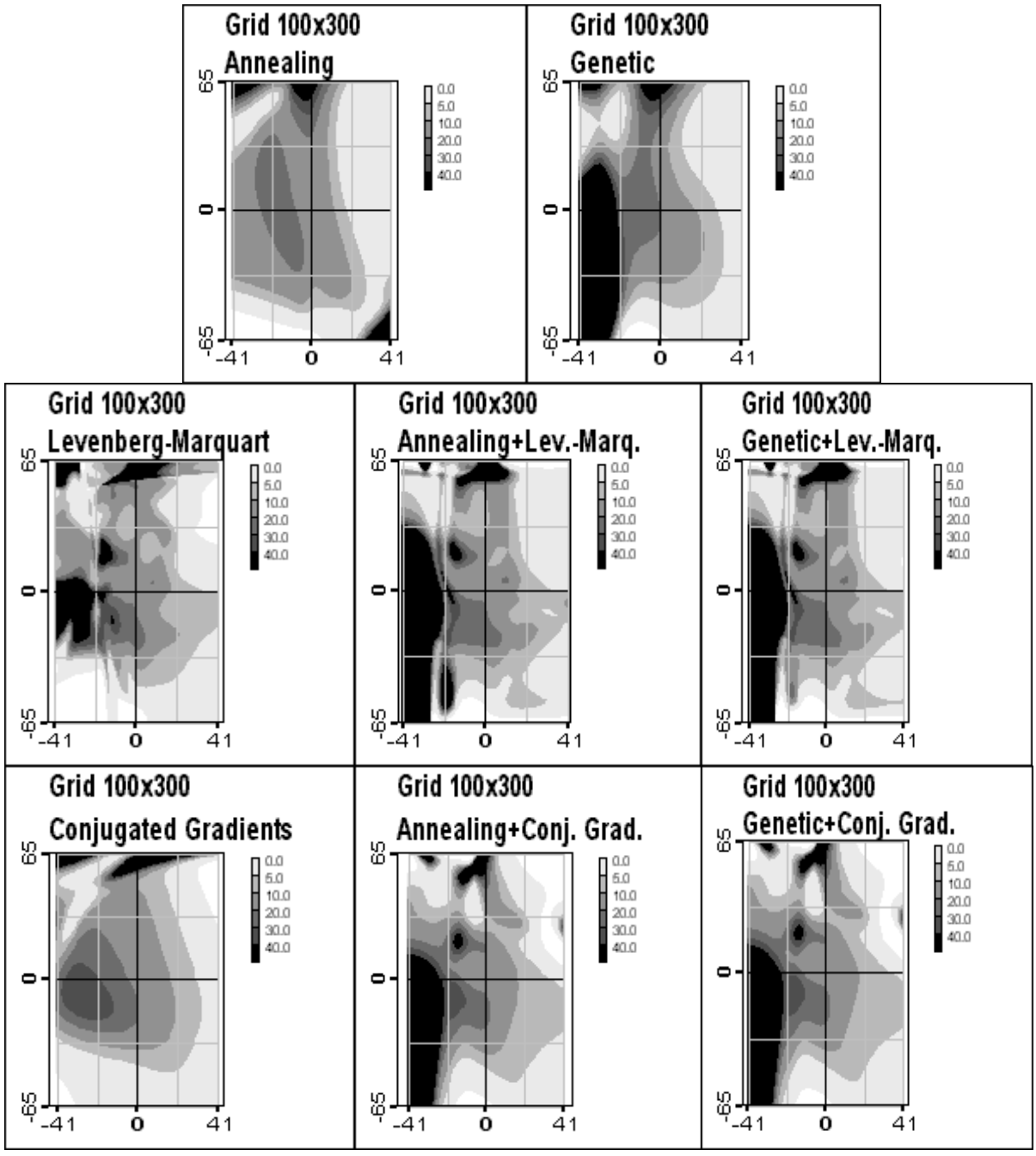


Рис. 5. Результаты применения МСП на регулярной сетке

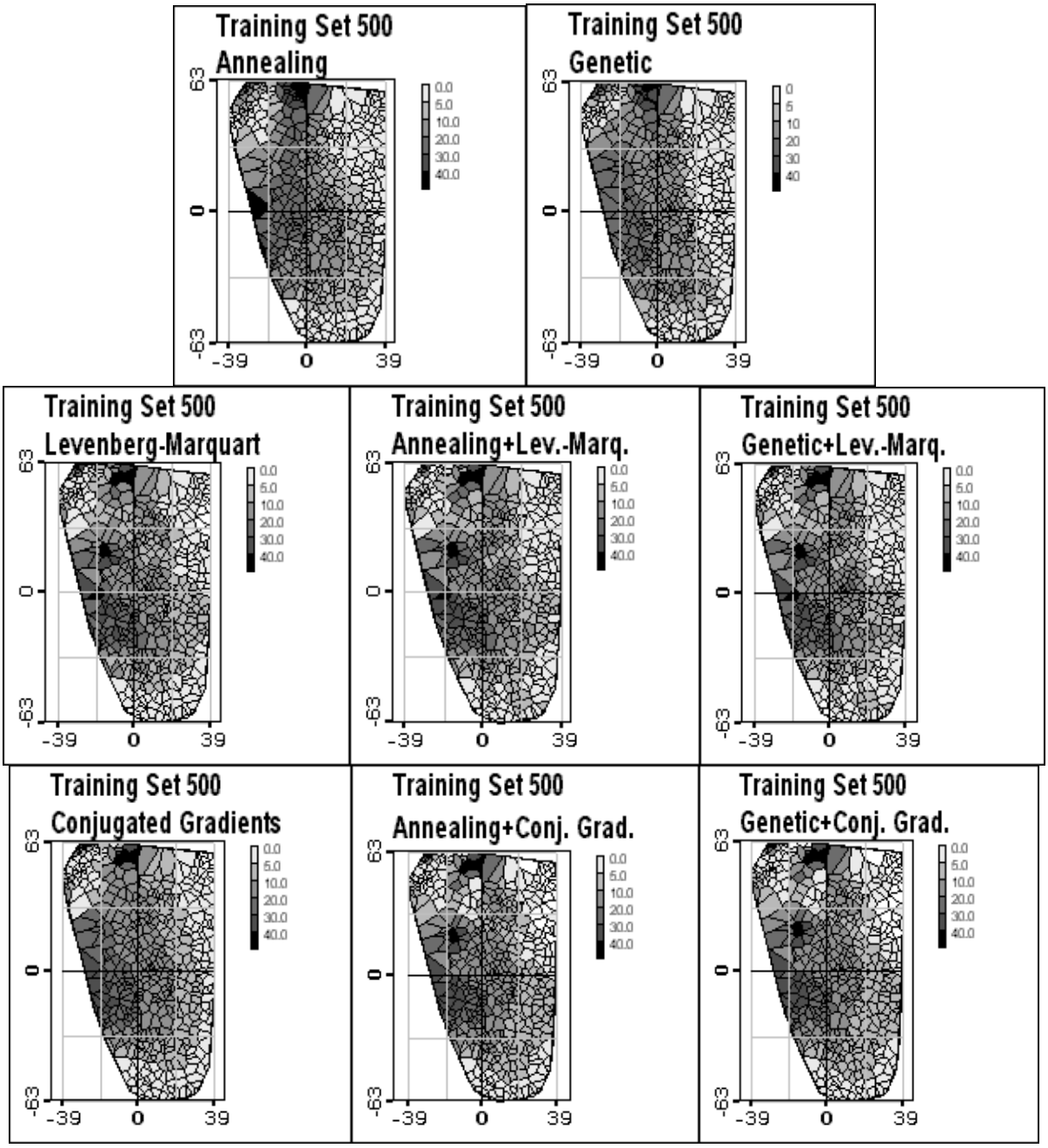


Рис. 6. Тест на аккуратность, тренировочный набор

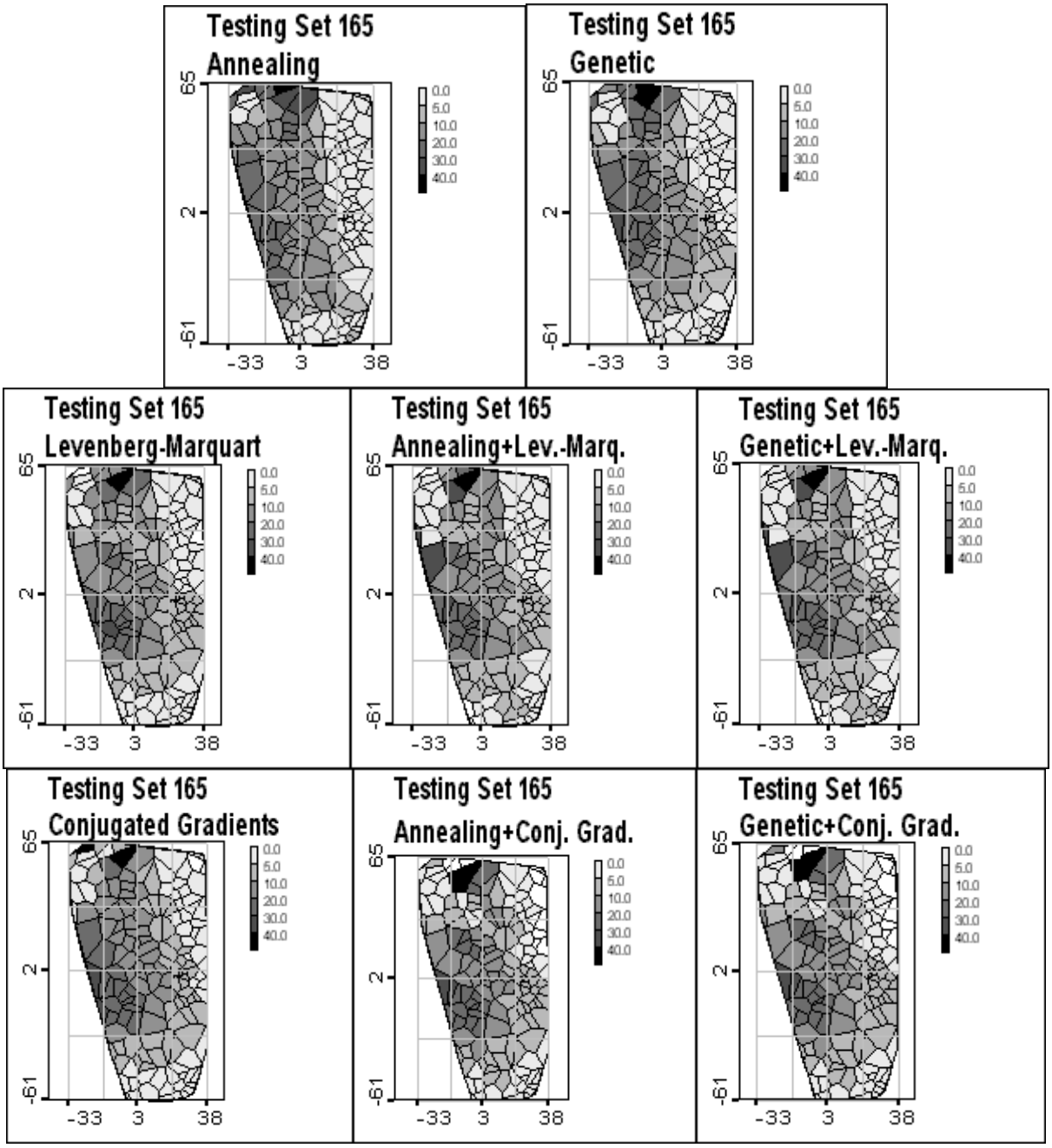


Рис. 7. Тест на аккуратность, тестировочный набор

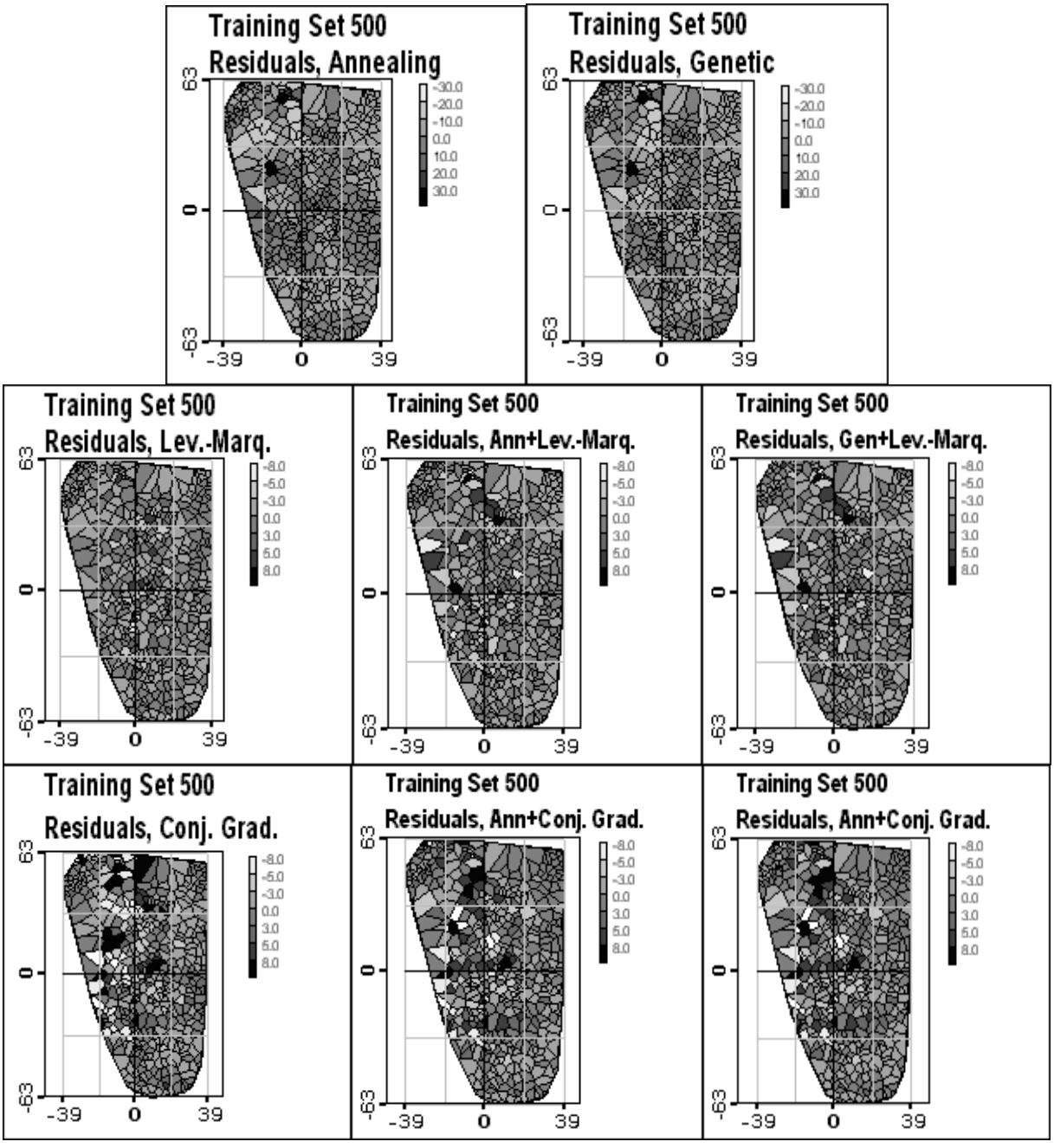


Рис. 8. Невязки, тренировочный набор

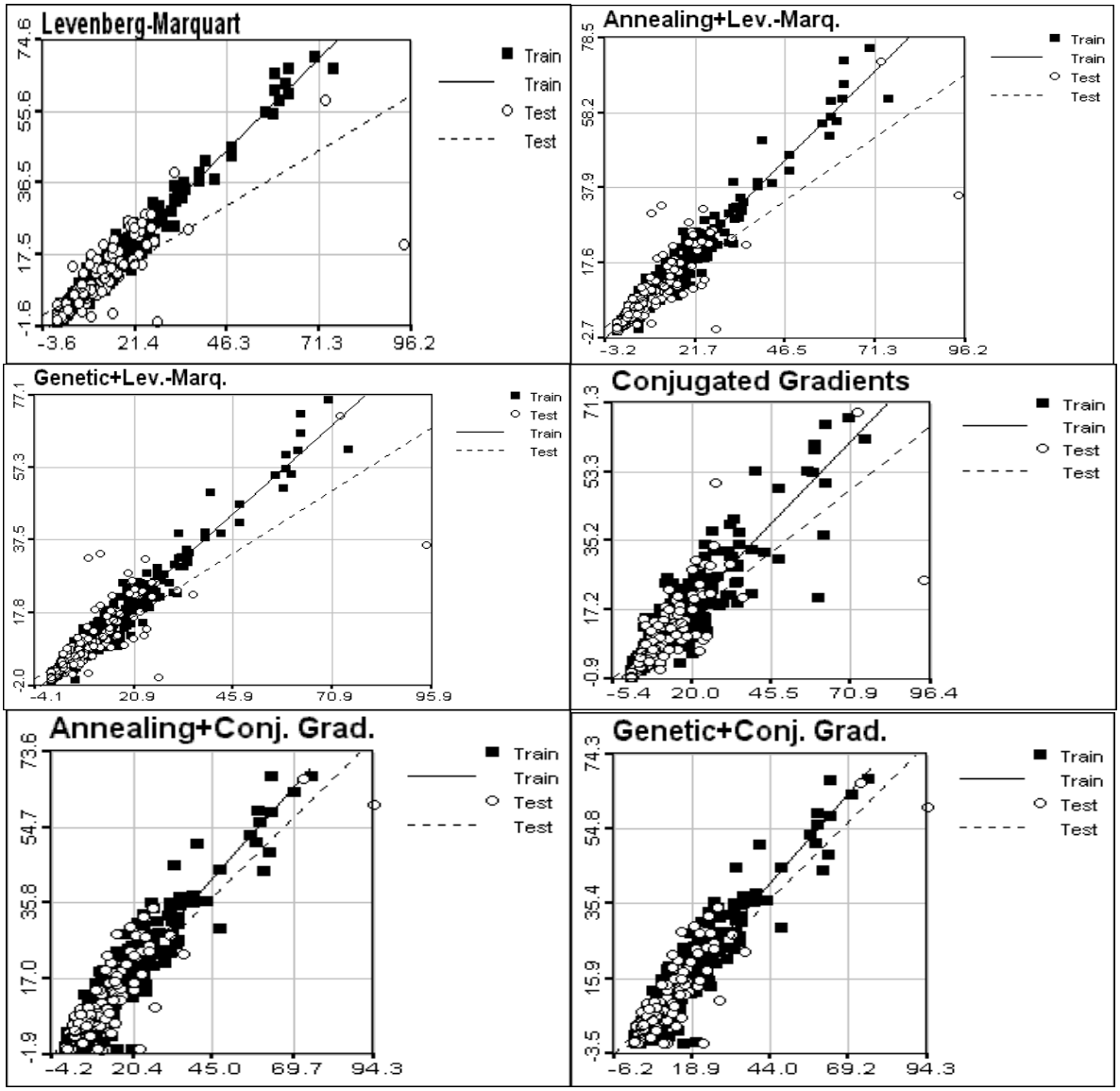


Рис. 9. Корреляции между выходами сети и желаемыми значениями

7 Заключение

В работе рассматривались различные методы тренировки МСП. Рассмотрено семейство градиентных методов (Левенберга-Маркара, сопряженных градиентов) и семейство стохастических методов (имитации отжига, генетического поиска). Составлены рекомендации по практическому использованию ИНС многослойный перцептрон для анализа картирования пространственных данных. Проведено исследование на примере реальных данных Чернобыльских выпадений.

Выводы:

1 Использование различных методов или их сочетание зависит во многом от решаемой задачи.
2 Если ограничения, налагаемые на метод Левенберга-Маркара отсутствуют, то использование его будет оправдано. При этом следует иметь ввиду его главный недостаток: он легко переходит в режим оверфитинга.
3 В качестве универсального метода можно использовать метод сопряженных градиентов. Он подойдет для решения любых задач, решаемых при помощи МСП.

4 Для инициализации весов (выбора стартовой точки) для метода сопряженных градиентов целесообразно использовать метод имитации отжига. Причем не следует задавать слишком строгие параметры, достаточно несколько температур с небольшим количеством итераций. Этого будет достаточно для выбора стартовой точки. Более целесообразно повторить этот цикл (отжиг плюс сопряженные градиенты) несколько раз и выбрать лучший результат.

5 Метод генетического поиска оказался неэффективен. Он хорошо решает поставленную задачу, но метод отжига делает это быстрее и намного проще в настройке. Оптимизировать же настройки генетического метода задача сложная и зависит от многих факторов. Чтобы повысить эффективность применения этого метода, нужно искать другие пути его совместного функционирования с МСП.

6 Следует уделять особое внимание определению тестировочного набора данных. Особенно актуально его применение для метода Левенберга-Маркара. Чтобы построить хорошую модель при помощи этого метода, целесообразно получить несколько результатов и выбрать лучший при помощи тестирования.

8 Благодарности

Работа проводилась при частичной поддержке INTAS грантов №№96-1957 и 97-31726 и гранта молодых ученых РАН “Искусственные нейронные сети и генетические алгоритмы для анализа и моделирования пространственной информации по окружающей среде” 1998-1999. В работе использовалось программное обеспечение Geostat Office, дополнительную информацию о котором можно найти на http://www.ibrae.ac.ru/~mkanev/, авторы – М.Ф. Каневский, В.В. Демьянов, С.Ю. Чернов, Е.А. Савельева, В.А. Тимонин.

9 Литература

1. Горбань А.Н., Россиев Д.А.. (1996). *Нейронные сети на персональном компьютере.* Наука, Новосибирск.
2. Горбань А.Н. (1990). *Обучение нейронных сетей.* М.,СССР-США СП ParaGraph.
3. Горбань А.Н.и др. (1998). *Нейроинформатика*. Новосибирск, издательство Сибирского отделения РАН ”Наука”.
4. Rumehlart, David, McClelland, James and the PDP Research Group (1986). *Parallel Distributed Process.* MIT Press, Cambridge, MA..
5. Masters, Timothy (1993). *Practical Neural Network Recipes in C++*. Academic Press, Inc.
6. Masters, Timothy. (1995). *Advanced Algorithms for Neural Networks. A C++ Sourcebook.* John Wiley & Sons, Inc.
7. Press, William H., Flannery, B., Teukolsky, S., and Vetterling, W. (1988). *Numerical Recipes in C*. Cambridge University Press, New York.
8. Aarts, E., and van Laarhoven, P. (1987). *Simulated Annealing: Theory and Practice.* John Wiley and Sons New York.
9. Azencott, R., ed. (1992). *Simulated Annealing: Parallelization Techniques.* John Wiley and Sons New York.
10. Styblinski, M.A., and Tang, T.-S. (1990). “Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing.” *Neural Networks*, 3: 467-483.
11. Szu, Harold (1986). “Nonconvex Optimization by Simulated Annealing.” *Proceedings of the IEEE*, **75**(11): 1538-1540.
12. Szu, Harold (1987).). “Fast Simulated Annealing.” *AIP Conference Proceedings 151: Neural Networks for Computing*, Snowbird, UT.
13. Davis, Lawrence (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.
14. Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA
15. Kanevsky M., Arutyunyan R., Bolshov L., Demyanov V., Linge I., Savelieva E., Shershakov V., Haas T., Maignan M. (1996) *Geostatistical Portrayal of the Chernobyl Fallout.* Geostatistics Wollongong 96, ed. E.Y. Baafi, N.A. Schofield, Kluwert Academic Publishers, vol. 2, pp. 1043-1054.