Лабораторная работа 2

ИССЛЕДОВАНИЕ НЕЙРОННЫХ СЕТЕЙ ПРЯМОГО РАСПРОСТРАНЕНИЯ

2.1 Цель работы

Изучение и приобретение навыков:

- предварительной настройки нейронных сетей:
  - инициализации;
  - задания функций обработки входных/выходных сигналов;
  - задания различных функций расчета производительности НС;
  - задания различных функций деления выборки;
- построения и обучения линейных нейронных сетей;
- решения задач классификации и аппроксимации с помощью НС ПР:
  - настройке параметров различных алгоритмов обучения;
  - выбора параметров НС ПР.

2.2 Список используемых сокращений

МО	математическое ожидание
НС	нейронная сеть
НС ПР	нейронная сеть прямого распространения
ПФ	передаточная функция
СКО	среднеквадратичная ошибка
LVQ	learning vector quantization (обучаемый векторный квантователь)
mse	mean squared error (средняя квадратичная ошибка)
mae	mean absolut error (средняя абсолютная ошибка)
PCA	principal component analysis (анализ главных компонент)
RELU	rectified linear unit (выпрямленный линейный блок)
sse	sum squared error (суммарная квадратичная ошибка)
sae	sum absolut error (суммарная абсолютная ошибка)
tf	transfer function (передаточная функция)

2.3 Теоретические сведения

2.3.1 Инициализация НС

После того как сформирована архитектура нейронной сети, должны быть заданы начальные значения весов и смещений, или иными словами, сеть должна быть инициализирована. Такая процедура выполняется с помощью метода init для объектов класса network. Оператор вызова этого метода имеет вид:

net = init (net).

Перед вызовом этого метода в вычислительной модели сети необходимо задать следующие свойства:

net.initFcn – функция инициализации всей НС;

net.layers{i}.initFcn – функция инициализации i-го слоя;

net.biases{i}.initFcn – функция инициализации смещений i-го слоя;

net.inputWeights{i,j}.initFcn – функция инициализации весов к слою i от входа j;

net.layerWeight{i,j}.initFcn – функция инициализации весов к слою i от входа j;

net.initParam – параметры функций инициализации.

Способ инициализации сети определяется заданием свойства **net.initFcn.** Для сетей с прямой передачей сигналов по умолчанию используется **net.initFcn = 'initlay'.** Она позволяет для каждого слоя использовать собственные функции инициализации, задающиеся свойствами **net.layers{i}.initFcn** с двумя возможными значениями: **'initwb'** и **'initnw'**.

Функция **initnw** реализует алгоритм **Nguyen-Widrow** и применяется для слоёв, использующих сигмоидальные функции активации. Эта функция генерирует начальные веса и смещения для слоя так, чтобы активные области нейронов были распределены равномерно относительно области входов, что обеспечивает минимизацию числа нейронов сети и времени обучения.

Функция **initwb** позволяет использовать собственные функции инициализации для каждой матрицы весов и каждого вектора смещений. При этом возможные значения для свойств **net.inputWeights{i,j}.initFcn, net.layerWeight{i,j}.initFcn**, **net.biases{i}.initFcn** приведены в таблице 2.1. Часть функций может использоваться для инициализации и весов и смещений (**initzero, rands, randsmall**), часть только для весов (**midpoint, randnc, randnr, initlvq, initsompc**), часть – только для смещений (**initcon**).

Таким образом, задание функций инициализации для вычислительной модели нейронной сети является многоступенчатым процессом и выполняется по следующему алгоритму:

1. Выбрать для свойства net.initFcn одно из возможных значений: 'initzero', 'initcon', 'initnw', 'initwb' или 'initlay'.

2. Если выбраны значения 'initzero', 'initcon' или 'initnw', то задание функций инициализации сети завершено.

3. Если выбрано значение 'initwb', то переход к шагу 6.

4. Если выбрано значение 'initlay', то переходим к слоям и для каждого слоя i свойству net.layers{i}.initFcn необходимо задать одно из возможных значений: 'initnw' или 'initwb'.

5. Если для i-го слоя выбрано значение 'initnw', то для этого слоя задание функций инициализации завершено.

6. Если для какого-то i слоя установлено свойство 'initwb', то для него необходимо задать свойства net.biases{i}.initFcn, net.inputWeights{i,j}, net.layerWeights{i,j} для всех возможных значений j (функции приведены в табл. ниже).

С помощью оператора **revert(net)** можно возвратить значения весов и смещений к ранее установленным значениям.

Табл. 2.1 Функции инициализации весов и смещений в Matlab

Функция	Описание
Функции инициализации для весов и смещений	
initzero	Инициализация нулями
rands	Симметричная инициализация в интервале [-1;1]
randsmall	Инициализация в интервале [-0.1;0.1]
Функции инициализации только весов	
midpoint	Инициализация средними значениями входных сигналов
randnc	Инициализация с нормализацией по столбцам
randnr	Инициализация с нормализацией по строкам
initlvq	Инициализация весов LVQ-сети
initsompc	Инициализация весов SOM-сети главными компонентами
Функции инициализации только смещений	
initcon	Инициализация смещений для обучения функцией learncon (используется для соревновательных НС)

2.3.2 Пред-/пост-обработка входных/выходных сигналов НС

Для каждого входа и выхода можно задать специальные функции пред/пост обработки. Для входов это нормализация, отсев шумов и лишних (неинформативных) входов, сокращение размерности (PCA). Для выходов это специальные функции вычисления выходных сигналов (к примеру округление). В таблице 2.2 ниже приведены имеющиеся в Matlab функции пред/пост обработки входных/выходных сигналов.

Задаются в: net.inputs{i}.processFcns, net.outputs{i}.processFcns

Табл. 2.2 Функции предобработки входных/выходных сигналов НС в Matlab

Функция	Описание
fixunknowns	Обработка элементов с незаданным значением
lvqoutputs	Обработка выходов LVQ-сети
mapminmax	Нормализация строк к диапазону [-1 1]
mapstd	Нормализация строк к нулевому МО и единичному СКО
processpca	Применение PCA
removeconstantrows	Удаление строк с постоянными значениями
removerows	Удаление строк с заданными индексами

2.3.3 Обучение НС – общие положения

После инициализации нейронной сети её необходимо обучить решению конкретной прикладной задачи. Для этих целей нужно собрать обучающий набор данных, содержащий интересующие признаки изучаемого объекта, используя имеющийся опыт. Сначала следует включить все признаки, которые, по мнению аналитиков и экспертов, являются существенными; на последующих этапах это множество, вероятно, будет сокращено. Обычно для этих целей используются эвристические правила, которые устанавливают связь между количеством необходимых примеров и размером сети. Количество примеров на порядок больше числа связей в сети и возрастает по нелинейному закону, так что уже при довольно небольшом числе признаков, например, может потребоваться огромное число примеров. Эта проблема носит название "проклятие размерности". Для большинства реальных задач бывает достаточно нескольких сотен или тысяч примеров.

После того как собран обучающий набор данных для проектируемой сети, производится автоматическая настройка весов и смещений с помощью процедур обучения, которые минимизируют разность между желаемым сигналом и полученным на выходе в результате моделирования сети. Эта разность носит название "ошибки обучения".

Используя ошибки обучения для всех имеющихся примеров, можно сформировать функцию ошибок или критерий качества обучения. Чаще всего в качестве такого критерия используется сумма квадратов ошибок. Для линейных сетей при этом удаётся найти абсолютный минимум критерия качества, для других сетей достижение такого минимума не гарантируется. Это объясняется тем, что для линейной сети критерий качества, как функция весов и смещения, является параболоидом, а для других сетей – очень сложной поверхностью в N+1-мерном пространстве, где N – число весовых коэффициентов и смещений.

С учётом специфики нейронных сетей для них разработаны специальные алгоритмы обучения. Алгоритмы действуют итеративно, по шагам. Величина шага определяет скорость обучения и регулируется параметром скорости настройки. При большом шаге имеется большая вероятность пропуска абсолютного минимума, при малом шаге может сильно возрасти время обучения. Шаги алгоритма принято называть эпохами или циклами.

На каждом цикле на вход сети последовательно подаются все обучающие примеры, выходные значения сравниваются с целевыми значениями, и вычисляется функция критерия качества обучения – функция ошибки. Значения функции ошибки, а также её градиента используются для корректировки весов и смещений, после чего все действия повторяются.

2.3.3.1 Останов обучения

Процесс обучения прекращается по следующим трём причинам, если:

а) реализовано заданное количество циклов (эпох);

б) ошибка достигла заданной величины;

в) ошибка достигла некоторого значения и перестала уменьшаться.

Во всех этих случаях сеть минимизировала ошибку на некотором ограниченном обучающем множестве, а не на множестве реальных входных сигналов при работе модели.

Критерии останова в Matlab задаются в соответствующих полях структуры net.trainParam и приведены в таблице 2.3 ниже:

Табл. 2.3 Критерии останова обучения НС в Matlab

Название	Описание критерия останова
goal	Допустимое значение функции качества (ошибки)
min_grad	Минимальный градиент функции качества
epochs	Максимальное число циклов (эпох) обучения
time	Максимальное время обучения в секундах
max_fail	Количество подряд идущих эпох, в течение которых ошибка на контрольной выборке не уменьшается

Для того, чтобы НС обучалась ровно epochs эпох, необходимо задать параметры следующими значениями:

net.trainParam.time=epochs
net.trainParam.goal=0,
net.trainParam.time=inf,
net.trainParam.min_grad=0,
net.trainParam.max_fail=0

2.3.3.2 Расчет ошибки

Ошибка в ходе обучения вычисляется с помощью функции, заданной в свойстве net.performFcn. Возможные варианты функций приведены в таблице 2.4.

Табл. 2.4 Функции ошибок в Matlab

Функция	Описание
mae	Средняя ошибка по модулю
mse	Средне-квадратичная ошибка
sae	Суммарная ошибка по модулю
sse	Суммарная средне-квадратичная ошибка
crossentropy	Кросс-энтропия (для классификации)

Необходимо учитывать, что ошибка в ходе расчета усредняется по множеству примеров из выборки (тестовой или контрольной), а также по множеству выходов. Соответственно, данный подход не всегда подходит. В частности, при классификации расчет ошибки удобней производить путем подсчета числа правильных (неправильных) классификаций и затем деления на общее число примеров.

Функция кросс-энтропии используется при решении задач классификации и в ряде случае позволяет добиться лучших результатов. Дополнительно к ней можно задать параметр регуляризации net.performParam.regularization (по умолчанию равен 0.1) и нормализацию net.performParam.normalization (по умолчанию 'none'). При создании НС с использованием функции patternnet используется функция ошибки crossentropy и выходной слой с softmax-ПФ.

2.3.3.3 Контроль переобучения

Попытки усложнить модель и снизить ошибку на заданном обучающем множестве могут привести к обратному эффекту, когда для реальных данных ошибка становится ещё больше. Эта ситуация называется явлением переобучения нейронной сети.

Для того чтобы выявить эффект переобучения нейронной сети, используется механизм контрольной проверки. С этой целью часть обучающих примеров резервируется как контрольные примеры и не используется при обучении сети. По мере обучения контрольные примеры применяются для независимого контроля результата. Если на некотором этапе ошибка на контрольном множестве перестала убывать, обучение следует прекратить даже в том случае, когда ошибка на обучающем множестве продолжает уменьшаться, чтобы избежать явления переобучения. В этом случае следует уменьшить количество нейронов или слоёв, так как сеть является слишком мощной для решения данной задачи. Если же, наоборот, сеть имеет недостаточную мощность, чтобы воспроизвести зависимость, то явление переобучения скорее всего наблюдаться не будет и обе ошибки – обучения и контроля – не достигнут требуемого уровня.

Таким образом, для отыскания глобального минимума ошибки приходится экспериментировать с большим числом сетей различной конфигурации, обучая каждую из них несколько раз и сравнивая полученные результаты. Главным критерием выбора в этих случаях является контрольная погрешность. При этом применяется правило, согласно которому из двух нейронных сетей с приблизительно равными контрольными погрешностями следует выбирать ту, которая проще.

Необходимость многократных экспериментов ведёт к тому, что контрольное множество начинает играть ключевую роль в выборе нейронной сети, т. е. участвует в процессе обучения. Тем самым его роль как независимого критерия качества модели ослабляется, поскольку при большом числе экспериментов возникает риск переобучения нейронной сети на контрольном множестве. Для того, чтобы гарантировать надёжность выбираемой модели сети, резервируют ещё тестовое множество примеров. Итоговая модель тестируется на данных из этого множества, чтобы убедиться, что результаты, достигнутые на обучающем и контрольном множествах, реальны. При этом тестовое множество должно использоваться только один раз, иначе оно превратится в контрольное множество.

2.3.3.4 Деление выборки

В пакете NN Toolbox Matlab при обучении НС выборка автоматически делится на три части – обучающую, тестовую и контрольную. Обучающая выборка используется для обучения, контрольная для контроля ошибки в ходе обучения, тестовая для окончательной оценки ошибки. По обучающей выборке рассчитывается градиент и производится настройка весовых коэффициентов и смещений. Контрольная выборка не позволяет НС недоучиться и переобучиться. Ошибка на обучающей и контрольной выборке начинает уменьшаться с началом обучения. С течением времени ошибка на обучающей продолжает все время уменьшаться, а на контрольной выборке в какой-то момент прекращает уменьшаться и начинает увеличиваться. Одной из причин этого может быть начала переобучение. Данный факт используется в процедуре раннего останова (early stopping) – НС прекращает обучаться спустя несколько шагов после того, как ошибка на контрольной выборке перестала уменьшаться.

В таблице 2.5 ниже приведены встроенные функции деления выборки на 3 подвыборки. Они задаются через свойство net.divideFcn:

Табл. 2.5 Функции деления выборки в Matlab

Функция	Описание
divideblock	Деление блоками (1-й блок – к 1-й подвыборке и т.д.)
divideind	Деление путем задания индексов каждой подвыборки
divideint	Деление перемежающимися блоками (1,4,7, …- к обуч, 2,5,8 к валидац., 3,6,9,… - к тест)
dividerand	Деление случайным образом
dividetrain	Все примеры заносятся в обучающую выборку

2.3.4 Линейные нейронные сети

Линейная сеть – это один слой из **S** нейронов и **R** входов, каждый из которых может состоять из нескольких элементов. Передаточной функцией каждого нейрона слоя является линейная функция **purelin**. Это единственное отличие линейной сети от персептрона. Благодаря линейной передаточной функции каждый выход сети может принимать любое значение, в то время как выход персептрона ограничен двумя значениями {0,1} или {-1,1}. Если персептрон использует разделяющую поверхность для отделения одного класса от другого, то линейная сеть аппроксимирует этой поверхностью данные из обучающей выборки.

Линейные сети, как и персептроны, способны решать только линейно разделимые задачи классификации. В них используется правило обучения Уидроу–Хоффа, основанное на методе наименьших квадратов. Это правило совпадает с правилом обучения персептрона.

Помимо основных входов нейроны линейной сети имеют вход для постоянного смещения, равного единице. Элементы входов и смещения взвешиваются с помощью функции скалярного произведения **dotprod** и суммируются с помощью функции **netsum**.

Линейная нейронная сеть может быть создана различными способами

1. С использованием функции newlin

```
net=newlin(P, S, id, lr),
net=newlin(P, T, id, lr),
```

где **P** – матрица размерности **R*Q, состоящая из Q R-мерных** входных векторов из обучающей выборки

T – аналогично P матрица размерности **S*Q2, состоящая из Q2 S-мерных** выходных векторов из обучающей выборки

S – количество выходов (нейронов);

id – описание линии задержки на входе сети, по умолчанию **[0];**

lr – скорость обучения, по умолчанию **0.01;**

Данный способ используется, когда в сеть включаются задержки, т. е. для создания динамических адаптивных линейных нейронных сетей **ADALIN (ADAptive Linear Neuron networks),** которые позволяют корректировать веса и смещения при поступлении на вход каждого элемента обучающего множества. Такие сети широко применяются при решении задач управления и обработки сигналов.

По умолчанию обучение функцией train – пакетное с использованием trainb, функцией adapt - инкрементальное с использованием adaptwb. Локальное правило обучения – Уидроу-Хоффа learnwh. Инициализация по умолчанию производится нулями с помощью функции initzero.

2. С использованием функции newlind

```
net=newlind(P, T)
```

Функция newlind создает линейную статическую сеть и сразу производит ее обучение путем решения системы уравнений, вытекающей из метода наименьших квадратов (МНК):

[W b] * [P; ones] = T

Система решается путем нахождения обратной (псевдообратной) матрицы от [P;ones]. Качество решения зависит от соотношения между такими величинами, как количество входов S, объем выборки Q, ранг матрицы [P;ones].

3. С использованием функции linearlayer

```
net = linearlayer(inputDelays,widrowHoffLR)
```

inputDelays – описание линии задержки по аналогии с id для newlin

widrowHoffLR – скорость обучения, по умолчанию 0.01

Параметры обучения аналогичны newlin.

При этом создается статическая линейная сеть. Количество входов и выходов не задается явно и может быть задано после вызова функции обучения train.

2.3.4.1 Правило обучения Уидроу-Хоффа (learnwh)

Функция learnwh реализует известное правило обучения Уидроу-Хоффа. Это правило является, по сути, итерационным воплощением метода наименьших квадратов. Приращение вектора весовых коэффициентов рассчитывается по формуле

dw = lr*e*p'

где lr – скорость обучения, e – вектор ошибки, p' – транспонированный входной вектор.

Скорость обучения можно задать вручную с помощью параметра learnLr соответствующих весовых коэффициентов, например

```
net.inputWeights{1}.learnParam.lr = 0.02;
```

Также скорость обучения можно задать для функций trainb, trains, trainr, trainc через параметр net.trainParam.lr. То есть возможны два варианта задания скорости обучения.

При работе с моделями линейных сетей могут возникнуть ситуации, когда число настраиваемых параметров недостаточно, чтобы выполнить все условия; в этом случае сеть считается недоопределенной (не существует ни одного точного решения). Если число параметров слишком велико, сеть считается переопределенной (существует бесконечное количество точных решений). И в том и в другом случае метод наименьших квадратов осуществляет настройку, стремясь минимизировать ошибку сети.

Архитектура однослойной линейной сети полностью определяется задачей, которая должна быть решена, причем число входов сети и число нейронов в слое определяются числом входов и выходов задачи соответственно.

Многослойные линейные НС в силу принципа суперпозиции эквиваленты однослойным, поэтому их использование лишено всякого смысла.

2.3.5 Нейронные сети прямого распространения

Нейронные сети прямого распространения могут состоять из произвольного количества слоев при условии, что каждый предыдущий слой передает свой только на слои,

которые следуют за ним. За счет этого в таких НС не могут образовывать алгебраические циклы и реализуют статическое отображение между векторными сигналами входного слоя (слоев) и векторными сигналами выходного слоя (слоев). Отсутствие обратных связей значительно упрощает процедуру обучения таких НС, что сделало их одной из наиболее популярных и используемых моделей для распознавания образов.

В классической НС ПР выделяют виртуальный входной слой, представленный набором входных сигналов, выходной слой нейронов и скрытые слои, которые находятся между выходными и входным.

Когда количество скрытых слоев не менее двух, НС называется **глубокой**. Чем больше глубина (количество скрытых слоев), тем более сложные абстракции и признаки НС сможет изучать. Чем слой находится дальше от входного, тем сложнее абстракции и наоборот.

В **сверточных** НС есть слои, которые, воспринимая входной для них сигнал как изображение, выполняют операцию его свертки с помощью специальных ядер и фильтров. Тем не менее по своей архитектуре такин НС также являются прямого распространения.

В классической НС ПР каждый предыдущий слой имеет связи к следующему за ним. Если же есть т.н. связи "через слой", то такая НС называется **каскадной**.

2.3.5.1 Архитектура НС ПР

При формировании архитектуры НС ПР вначале определяется количество скрытых слоев. Далее для каждого скрытого слоя задается количество нейронов, а также определяется тип передаточной функции этого слоя. Также задается передаточная функция выходного слоя.

Количество слоев задается свойством net.numLayers. С помощью свойства net.outputConnect определяется, какие слои являются выходными (по умолчанию последний). Для задания связей между слоями используется свойство net.layerConnect, а для задания связей от входов к слоям используется свойство net.imputConnect.

Количество нейронов в выходном слое совпадает с количеством классов, а количество входов совпадает с размерностью образов (количеством признаков). Количество нейронов в Matlab задается через свойство net.layers{i}.size.

Передаточные функции слоев задаются через свойство net.layers{i}.transferFcn. В таблице 2.6 ниже показаны основные типы передаточных функций, используемых в Matlab.

Табл. 2.6 Передаточные функции слоев НС в Matlab

Функция	Описание	ОВЗ
compet	Соревновательная ПФ	$\{0,1\}^N$
hardlim	Пороговая ПФ	{0,1}
hardlims	Пороговая симметричная ПФ	{-1;1}
logsig	Логарифмическая сигмоидальная ПФ	(0;1)
netinv	Обратная ПФ	$(-\infty;0),(0;+\infty)$
poslin	Положительная линейная ПФ (RELU)	$[0;+\infty)$
purelin	Линейная ПФ	$(-\infty;+\infty)$
radbas	Радиально-базисная ПФ	(0;1]
radbasn	Радиально-базисная ПФ с нормализацией	(0;1]
satlin	Положительная линейная ПФ с насыщением	[0;1]
satlins	Линейная симметричная ПФ с насыщением	[-1;1]
softmax	Соревновательная ПФ с мягким максимумом	$[0;1]^N$
tansig	Симметричная сигмоидальная ПФ	(-1;1)
tribas	Треугольная ПФ	[-1;1]

По умолчанию в Matlab при аппроксимации задаются следующие функции:

- передаточные функции скрытых слоев – сигмоидальные (tansig, logsig),
- передаточная фукнция выходного слоя – линейная (purelin)

При решении задачи классификации по умолчанию задаются следующие функции:
- передаточная функция выходного слоя – softmax;

Также в Matlab есть большое число встроенных функций для создания НС ПР. Часть этих функций, относящаяся к данной работе, приведена ниже в таблице 2.7.

Табл. 2.7 Функции для создания НС ПР в Matlab

Функция (новая)	Функция (старая)	Описание
cascadeforwardnet	**newcf**	Каскадная сеть ПР (прямого распространения)
feedforwardnet	**newff**	Сеть ПР
fitnet	**newfit**	НСПР для аппроксимации
linearlayer	**newlin**	Линейная НС
network		Новая НС
newlind		Линейный слой (упрощенное создание)
patternnet	**newpr**	НС для классификации

2.3.5.2 Сигнатура НС ПР

При проведении большого количества исследований целесообразно кратко записывать информацию о конфигурации нейронной сети. Как один из вариантов, можно предложить следующие:

nhl(tf1, tf2,…,tfk)-tfout-K-pf - краткий

nhl(tf1-n1, tf2-n2,…,tfk-nk)-tfout-K-pf - подробный,

где nhl – количество скрытых слоев, tfi – передаточная функция i скрытого слоя, ni – количество нейронов в i-м скрытом слоев, tfout – передаточная функция выходного слоя, pf – функция ошибки (производительности)

Например, 2(satlin,tansig)-logsig-K-mse обозначает каскадную НС с 2 скрытыми слоями с ПФ satlin, tansig и с ПФ logsig у выходного слоя, функция ошибки mse.

А 3(tansig, tansig, tansig)-softmax-crossentropy обознает НС с 3 скрытыми слоями типа tansig и ПФ softmax у выходного слоях и функцией ошибки cross-entropy. Для обозначения количества скрытых нейронов можно указывать их в скобках после типа ПФ числа скрытых слоев. Например, 2(satlin-15,tansig-5)-logsig-K-mse или 3(tansig-15, tansig-5, tansig-3)-softmax-crossentropy.

Зам. Количество нейронов в выходном слое совпадает с количеством классов, а количество входов совпадает с размерностью образов (количеством признаков), поэтому они в сигнатуре не указываются.

2.3.5.3 Обучение НС ПР

2.3.5.3.1 Алгоритмы обучения

По умолчанию для сетей с прямой передачей сигналов в качестве критерия обучения используется функционал, представляющий собой сумму квадратов ошибок между выходами сети и их целевыми значениями (sse):

$$J = 0.5 \sum_{q=1}^{Q} \sum_{i=1}^{S} (t_i^q - d_i^q)$$

где Q - объём выборки; q - номер примера; i - номер выхода; t_i^q - целевое значение для i-го выхода выборки q; d_i^q - сигнал на i-м выходе при подаче входных сигналов q-й выборки. Целью обучения сети является минимизация этого функционала с помощью изменения весов и смещений.

В настоящее время разработано несколько методов минимизации функционала ошибки на основе известных методов определения экстремумов функций нескольких переменных. Все эти методы можно разделить на три класса:

а) методы нулевого порядка, в которых для нахождения минимума используется только информация о значениях функционала в заданных точках;

б) методы первого порядка, в которых используется градиент функционала ошибки по настраиваемым параметрам, использующий частные производные функционала;

в) методы второго порядка, в которых используются вторые производные функционала.

Для линейных сетей задача нахождения минимума функционала (параболоида) сводится к решению системы линейных уравнений, включающих веса, смещения, входные обучающие значения и целевые выходы и, таким образом, может быть решена без использования итерационных методов. Во всех остальных случаях надо использовать методы первого или второго порядка.

Если используется градиент функционала ошибки, то

$$X_{k+1} = X_k - \alpha_k g_k$$

где X_k и X_{k+1}- векторы параметров на k-й и k+1-й итерациях; α_k – параметр скорости обучения; g_k - градиент функционала, соответствующий k-й итерации.

Если используется сопряжённый градиент функционала, то на первой итерации направление движения p_0 выбирают против градиента g_0 этой итерации:

$$p_0 = -g_0$$

Для следующих итераций направление p_k выбирают как линейную комбинацию векторов g_k и p_{k-1}:

$$p_k = -g_k + \beta_k p_{k-1}$$

а вектор параметров рассчитывают по формуле:

$$X_{k+1} = X_k + \alpha_k p_k$$

Для методов второго порядка расчет параметров на k-м шаге производят по формуле (метод Ньютона):

$$X_{k+1} = X_k - H_k^{-1} g_k$$

где H_k- матрица вторых частных производных целевой функции (матрица Гессе); g_k - вектор градиента на k-й итерации. Вычисление матрицы Гессе требует больших затрат машинного времени, поэтому её заменяют приближенными выражениями (квазиньютоновские алгоритмы).

Градиентными алгоритмами обучения являются:

GD – алгоритм градиентного спуска;

GDM – алгоритм градиентного спуска с возмущением;

GDA – алгоритм градиентного спуска с выбором параметра скорости настройки;

Rprop – пороговый алгоритм обратного распространения ошибки;

GDX – алгоритм градиентного спуска с возмущением и адаптацией параметра скорости настройки.

Алгоритмами, основанными на использовании метода сопряженных градиентов, являются:

CGF - алгоритм Флетчера-Ривса;

CGP - алгоритм Полака-Ребейры;

CGB - алгоритм Биеле-Пауэлла;

SCG – алгоритм Молера.

Квазиньютоновскими алгоритмами являются:

DFGS – алгоритм Бройдена, Флетчера, Гольдфарба и Шанно;

OSS – одношаговый алгоритм метода секущих плоскостей (алгоритм Баттини);

LM – алгоритм Левенберга–Марквардта;

BR – алгоритм Левенберга–Марквардта с регуляризацией по Байесу.

В процессе работы алгоритмов минимизации функционала ошибки часто возникает задача одномерного поиска минимума вдоль заданного направления. Для этих целей используется метод золотого сечения **GOL,** алгоритм Брента **BRE**, метод половинного деления и кубической интерполяции **HYB**, алгоритм Чараламбуса **CHA** и алгоритм перебора с возвратом **BAC.**

2.3.5.3.1 Функции обучения в Matlab

Вышеперечисленные алгоритмы обучения НС прямого распространения реализованы в Matlab. Соответствующие функции приведены в таблице 2.8 ниже. Для задания функции обучения используются свойства net.trainFcn, net.adaptFcn.

Табл. 2.8 Функции обучения НС ПР в Matlab

Функция	Описание
Методы градиентного спуска	
traingd	Градиентный спуск
traingda	Градиентный спуск с адаптивной скоростью обучения
traingdm	Градиентный спуск с моментом инерции
traingdx	Градиентный спуск с моментом ин-и и адапт. скоростью обучения
trainrp	RPROP – эластичное обр.распр-е – важен только знак градиента
Методы сопряженных градиентов	
traincgb	Метод сопряженных градиентов Пауэла-Била
traincgf	Метод сопряженных градиентов Флетчера-Ривза
traincgp	Метод сопряженных градиентов Поляка-Рибьера
trainscg	Метод сопряженных градиентов с масштабированием
Методы переменной метрики (квазиньютоновские)	
trainbfg	Квазиньютоновский алгоритм BFGS
trainbr	Байесова регуляризация + trainlm
trainlm	Метод Левенберга-Марквардта (квазиньютоновский)
trainoss	Метод одношаговой секущей (модификация BFGS)
Обучение с учителем с использованием локальный функций обучения (learnFcn) весов и смещений	
trainb	Пакетное обучение
trainc	Циклическое инкрементальное обучение
trainr	Инкрементальное обучение в случайном порядке
trains	Последовательное инкрементальное обучение
adaptwb	Последовательное инкрементальное обучение (для net.adaptFcn)

Если в качестве глобальной функции обучения задана функция с использованием локальных функций обучения, то необходимо задать эти локальные функции для каждой весовой матрицы и вектора смещения через свойства net.inputWeights.learnFcn, net.layerWeights.learnFcn, net.biases.learnFcn. Возможные значения локальный функций обучения для НС ПР приведены ниже в таблице 2.9.

Табл. 2.9 Локальные функции обучения НС ПР в Matlab

Функция	Описание
Обучение весов/смещений	
learngd	Градиентный спуск
learngdm	Градиентный спуск с моментом

2.3.5.3.3 Функции ошибки

По умолчанию в Matlab при аппроксимации (НС fitnet) в качестве функции ошибки используется среднеквадратичная ошибка (mse). Также можно использовать и функции mae, sae, sse.

При решении задачи классификации (НС patternnet) по умолчанию задается функция кросс-энтропия (crossentropy). Тем не менее это не говорит о том, что для классификации не следует использовать функции mse, mae, sse, sae.

2.3.5.4 Процедура синтеза НС ПР

Процедура синтеза нейронной сети состоит из следующих шагов:

1. Выбрать начальную конфигурацию сети с одним скрытым слоем с числом нейронов, равным половине общего количества входов и выходов. Выбрать передаточные функции слоев в зависимости от решаемой задачи.

2. Обучить сеть и оценить качество ее работы на тестовой выборке.

3. В случае недообучения или переобучения произвести модификации:

- увеличить объем обучающей выборки.
- изменить алгоритм обучения или его параметры (скорость и др.);
- изменить архитектуру и параметры НС - увеличить количество скрытых нейронов, добавить дополнительные слои и/или связи со слоями, изменить передаточные функции и др.

Недообучение характеризуется тем, что НС решает свою задачу (аппроксимацию/классификацию) слишком грубо и неточно, причем этот эффект присутствует как на обучающей, так и на тестовой выборке. НС в данном случае чересчур обобщает данные и не запоминает детали. Недообучение может быть вызвано тем, что сложность НС недостаточна по сравнению со сложностью решаемой задачи. Данная проблема решается повышением сложности НС – увеличинием количества нейронов/слоев.

Переобучение наоборот характеризуется тем, что НС идеально справляется с решением задачи на обучающей выборке, но при этом на тестовой выборке разница между желаемыми и реальными значениями довольно высока. НС в данном случае плохо обобщает и вместо этого просто запоминает обучающие примеры. Чтобы у НС не было переобучения, объем обучающей выборки должен быть достаточно высоким, а архитектура и параметры НС должны быть подстроены в соответствии с решаемой задачи.

Переобучение легко контролировать с помощью процедуры ***раннего останова***, когда параллельно с обучением на специальной контрольной выборке (не совпадающей с обучающей) оценивается качество и при отрицательной динамике качества на контрольной выборке процесс обучения останавливается.

Хорошо спроектированная сеть должна обладать свойством ***обобщения***, когда она, будучи обученной на некотором множестве данных, приобретает способность выдавать правильные результаты для достаточно широкого класса данных, в том числе и не представленных при обучении. Идеальная НС должна запоминать полезные данные, хорошо обобщать и быть инвариантной к различного рода шумам.

Для того чтобы проектируемая сеть успешно решала задачу, необходимо обеспечить ***представительность*** обучающего, контрольного и тестового множества. По крайней мере, лучше всего постараться сделать так, чтобы примеры различных типов были представлены равномерно и сбалансированно. В некоторых случаях, когда примеров одного из классов очень мало прибегают к процедуре балансировки, когда искусственно наращивается количество примеров "бедных" классов.

2.4 Примеры

2.4.1 Создание и обучение линейной НС с помощью функции newlind, построение поверхности ошибок

Построить поверхность ошибок для линейной сети с одним нейроном и одним входом, используя функцию **errsurf**. Для создания и обучения НС использовать функцию newlind.

```
P = [1.0 -1.2];        % Обучающая выборка – входной сигнал
T = [0.5 1.0];         % Обучающая выборка – выходной сигнал
net = newlind(P,T); % Создание и обучение НС
Y = sim(net, P)      % Выход НС
E = T - Y              % Ошибка
SSE = sumsqr(E)   % Суммарная ошибка
net.IW{1,1}          % Весовой коэффициент
net.b{1}   % Смещение
w_range = -1:0.1:1; % диапазон весов;
b_range = -1:0.1:1;    % диапазон смещений;
ES = errsurf(P, T, w_range, b_range, 'purelin'); % Вычисление поверхности
plotes(w_range,b_range,ES);            % Построение поверхности
hold on
plotep(net.IW{1,1}, net.b{1}, SSE); % Построение точки [W b] после обучения сети на поверхности ошибок
hold off      % – знаком 'x' отмечены оптимальные значения веса и % смещения.
```

2.4.2 Создание НС с помощью newlin, аппроксимация прямой линии

Создать с помощью функции **newlin** линейную сеть с одним нейроном и одним входом, обучить эту сеть, используя процедуру **train**, и построить поверхность функции критерия качества и траекторию обучения, выполнив следующие действия:

```
% 1.         Сформировать обучающее множество и задать скорость обучения
P = [1 -1.2];
T = [0.5 1];
maxlr = 0.40*maxlinlr(P,'bias');
% 2. Создать линейную сеть:
net = newlin([-2 2], 1, [0], maxlr);
gensim(net)
% 3. Рассчитать функцию критерия качества:
w_range = -1:0.2:1;  b_range = -1:0.2:1;
ES = errsurf(P,T,w_range,b_range,'purelin');
% 4. Построить поверхность функции критерия качества:
surfc(w_range, b_range, ES);
% 5. Рассчитать траекторию обучения:
X = zeros(1, 50); Y = zeros(1, 50);
net.IW{1,1} = 1; net.b{1} = -1 % – начальные значения;
X(1) = net.IW{1}; Y(1) = net.b{1};
net.trainParam.goal = 0.001;
net.trainParam.epochs = 1;
% 6. Вычислить веса и смещения:
for I=2:50,
[net, tr] = train(net, P, T);
X(I) = net.IW{1, 1};
Y(I) = net.b{1}; end
% 7. Построить линии уровня и траекторию обучения:
clc; % очистка экрана
contour(w_range, b_range, ES, 20) % 20 линий
hold on
plot(X,Y,'-*')     % – построение траектории
hold off
% 8.         Оценить значения параметров настройки для двух значений цели goal:
net.IW{1, 1} = 1; net.b{1} = -1;
net.trainParam.epochs = 50;
net.trainParam.goal = 0.001;  % – первое значение;
```

```
[net, tr] = train(net, P, T);
net.IW{1, 1}, net.b{1}    %
net.trainParam.goal = 0.00001; % – второе значение
[net, tr] = train(net, P, T);
net.IW{1, 1}, net.b{1} %
```

Как влияет параметр goal на качество обучения?

2.4.3 Аппроксимация нелинейной зависимости

Обучите НС на следующих примерах

```
P = [ 1.0   2.0   3.0; ...
     4.0   5.0   6.0];
T = [0.5 1.0 -1.0];
net = newlin(P, 1);
net = train(net,P,T);
y = sim(net, P)
(T(1)+T(3))/2 - T(2)          % Не равняется нулю
(y(1)+y(3))/2 - y(2)          % Будет равно нулю, т.к. НС линейная
e = std(T-y)                  % СК ошибка
```

Входные вектора являются линейно зависимыми (P2 = (P1+P3)/2). Реакция линейной системы на них по принципу суперпозиции должна быть (T1+T3)/2. Но T2≠(T1+T3)/2. Поэтому без ошибки обучение не произвести. Проверьте и убедитесь в этом.

2.4.4 Классификация с помощью линейной сети

Сформировать линейную сеть из одного нейрона и одного входа с двумя элементами для классификации значений входа, выполнив следующие действия:

```
P = [2 1 -2 -1; 2 -2 2 1];
T = [0 1 0 1];
net = newlin(P, 1);
net.trainParam.goal = 0.01;
[net, tr] = train(net, P, T);
net.IW{1, 1}, net.b{1}
A = sim(net, P);
threshold = 0.5;
y1 = (A > threshold)
```

Определите среднюю ошибку. Подберите значение порога threshold, обеспечивающего нулевую ошибку.

2.4.5 Неопределенность при обучении

Попробуйте обучить линейную НС на одном произвольном примере. Проанализируйте полученные результаты. Постройте зависимость, которую реализует НС. Как она соотносится с обучающим примером?

```
P = [1];T = [2];                % Пример
net=newlin(P,1);
net.inputweights{1}.initfcn = 'rands';
net.biases{1}.initfcn = 'rands';
net.trainParam.goal = 1e-5;
x = 0:.1:2;
n_times = 20;
plot(P,T,'rx'); hold on; grid on;
for i = 1:n_times
  net = init(net);
  net = train(net,P,T);
  y = sim(net,x);
  plot(x,y, 'color', rand(1,3));
end;
```

2.5 Практические задания

2.5.1 Изучение вспомогательных функций

2.5.1.1 Изучение функций инициализации

1. С помощью команды network создайте НС с n входами, 1 выходным слоем с m нейронами.

Задайте функцию инициализации НС initlay.

2. Задайте функцию инициализации слоев initnw. Произведите инициализацию НС с помощью функции init. Проанализируйте матрицы смещений и весов нейронной сети.

3. Задайте функцию инициализации слоев initwb.

Задавайте для весов и смещений различные функции инициализации (**initzero, rands, randsmall –** и для весов и для смещений**, midpoint, randnc, randnr, initlvq, initsompc –** только для весов**, initcon –** только для смещений**)**. Произведите инициализацию НС с помощью функции init. Проанализируйте матрицы смещений и весов нейронной сети.

2.5.1.2 Изучение функций предобработки входов/выходов.

Задавайте в качестве функций предобработки различные встроенные функции **(fixunknowns, mapminmax, mapstd, processpca, removeconstantrows, removerows)** – по очереди по одной**.**

Для каждой из функций предобработки подберите актуальные входные примеры, на которые эти функции должны срабатывать (для fixunknowns например задайте в качестве одного из входных сигналов NaN). Подайте эти примеры на вход этих функций и проанализируйте обработанные функциями примеры.

2.5.1.3 Изучение функций расчета производительности (усреднения ошибки)

Сформируйте различные вектора (матрицы) ошибок E (интерпретируйте их как вектора ошибок НС на тестовой выборке). Размерность E = число выходов * число векторов. Последовательно примените к ним функции mae, mse, sae, sse, crossentropy, mseparse. Проанализируйте каждый из способов усреднения ошибки (можно ли ему доверять и когда – при каком числе выходов НС и при каком объеме тестовой выборки). Рассмотрите случаи, когда ошибки сильно рознятся по входам/примерам.

2.5.1.4 Изучение функций деления выборки

Для объема выборки Q = 100 примените функции деления выборки divideblock, divideind, divideint, dividerand, dividetrain. Ознакомьтесь с входными параметрами каждой из функций и при исследовании функций меняйте эти параметры. Проанализируйте, как формируются обучающая, тестовая и контрольная выборки для каждой из функций.

2.5.2 Линейные нейронные сети

2.5.2.1 Аппроксимация нелинейной зависимости

1. Сформируйте обучающую и тестовую выборки для задачи аппроксимации 1-мерной нелинейной зависимости (варианты с номером 5).

2. Создайте линейную НС и обучите ее на обучающей выборке. Постройте графики зависимостей, реализуемых НС на различных этапах обучения на одном графике с исходной зависимостью. Оцените ошибку аппроксимации (абсолютную и относительную) на обучающей и тестовой выборках.

3. Создайте НС и одновременно обучите ее в соответствии с МНК с помощью функции newlind. Сравните результаты (результирующую функцию и ошибки) с предыдущим вариантом.

2.5.2.2 Выбор скорости обучения

Сформируйте обучающую выборку для функции (варианты с номером 5). Определите максимальную скорость обучения с помощью функции maxlinlr (разберитесь с принципом определения этой скорости). Проверьте, является ли эта скорость максимальной и если нет, то экспериментальным путем определите максимально допустимую скорость обучения. Затем проведите последовательность экспериментов для трех вариантов (скорость обучения а) больше (1 значение), б) равна (1 значение), в) меньше максимально допустимой (3 значения):

1. Задайте фиксированные начальные значениями [W,b].

2. Проведите обучение на достаточном числе циклов обучения. Постройте график изменения вектора [W,b] в течение обучения на графике поверхности (линий равного уровня) ошибок.

Сравните результаты. Определите значение скорости обучения, при которой обучение проходит быстрее всего.

```
function neuro_lin_example1
close all;
P=0:.01:1; w = 2;
T = sin(w*P);
maxlr = 0.10*maxlinlr(P,'bias');
delta_epochs = 10;
n_delta = 30;
net = newlin(P,1,0, maxlr);
net.iw{1} = .2;
net.b{1} = 0;
net.trainparam.epochs = delta_epochs;
w = zeros(n_delta +1,2); w(1,:) = [net.iw{1} net.b{1}];
for i = 1: n_delta
net = train(net,P,T);
w(i+1,1) = net.iw{1};
w(i+1,2) = net.b{1};
end;
subplot(2,1,1);
w_range = 0:0.1:1;  b_range = 0:0.1:1;
ES = errsurf(P,T,w_range,b_range,'purelin');
contour(w_range, b_range, ES); hold on; grid on;
plot(w(:,1), w(:,2), '-ro');
xlabel('w_1'); ylabel('b');
subplot(2,1,2);
plot(P,T, '-ro');
hold on; grid on;
Y = sim(net,P);
plot(P,Y, '-bx');
legend({'Desire', 'Real'});
xlabel('x'); ylabel('y');
```

2.5.2.3 Многослойная линейная НС

Покажите математически, что многослойная линейная сеть эквивалентна однослойной линейной НС.

Разработайте соответствующий алгоритм перехода от многослойной к эквивалентной однослойной НС для некоторой формализованной многослойной линейной НС (n_0 входов, s слоев, в каждом слое i по n_i нейронов, весовой коэффициент от i нейрона k-го слоя к j нейрону (k+1)-го слоя равен $w_{ij}^{(k)}$.

2.5.3 Аппроксимация статических зависимостей

2.5.3.1. Инициализация.

Создайте НС ПР с 1 скрытым слоем и 20 скрытыми нейронами с помощью функции newff, feedforwardnet или fitnet. Инициализируйте НС с помощью функции init случайными значениями. Приведите график структуры НС с помощью команды view. В первом скрытом слое должна быть сигмоидальная передаточная функция, а в выходном – линейная.

2.5.3.2. Формирование обучающей и тестовой выборки.

В качестве аппроксимируемой функции используйте собственную функцию из заданий с № 5. Сформируйте обучающую и тестовую выборки. Объем обучающей выборки должен быть **достаточным**, а сама выборка – представительной.

Для проверки можно построить обучающую выборку и ***визуально*** проанализировать, достаточно ли точно по этой выборке можно воссоздать исходную зависимость.

Тестовая и обучающая выборка не должны содержать одинаковых элементов. Самый простой и надежный вариант для генерации входных значений – случайные значения, равномерно распределенные в диапазоне возможных значений аппроксимируемой функции. Т.е. если функция задана в диапазоне [-10, 10], можно сформировать N_{train} и N_{test} случайных значений с равномерным распределением $R(-10{,}10)$.

!!! При недостаточном объеме обучающей выборки все дальнейшие исследования могут оказаться бесполезными, поэтому следует уделить этому процессу повышенное внимание.

2.5.3.3. Задание критериев останова обучения.

Научитесь задавать разными способами критерий останова обучения НС (с помощью полей структуры net.trainParam – time, goal, min_grad, max_fail, epochs). В дальнейшем необходимо будет подстраивать эти значения в зависимости от алгоритма, параметров обучения, а также параметров НС и требуемых показателей качества. Посмотрите, какая функция производительности (расчета ошибки) используется по умолчанию.

2.5.3.4. Обучение НС.

Задайте у НС n_{hidden} скрытых нейронов и алгоритм обучения trainlm. Обучите НС на обучающей выборке. Проанализируйте результат обучения. Постройте зависимость ошибки аппроксимации на тестовой выборке от номера эпохи.

Постройте график аппроксимации функции, реализуемый НС. Для этого здесь и в дальнейшем всегда используйте данные тестовой выборки.

Зам. n_{hidden} следует подобрать таким, чтобы после обучения графики аппроксимации и исходной функции визуально мало отличались.

2.5.3.5. Зависимость ошибки от числа нейронов в скрытом слое.

Изменяя число нейронов в скрытом слое в разумном диапазоне, рассчитайте зависимость ошибки (или функции производительности) как функцию от числа нейронов в скрытом слое. Определите *наиболее подходящий* диапазон значений числа скрытых нейронов. Постройте на одном графике аппроксимации для наилучших, удовлетворительных и наихудших случаев. На этом же графике отобразите исходную аппроксимируемую функцию.

Зам. Максимальное число нейронов в скрытом слое в эксперименте можно взять равным $k \cdot n_{hidden}^{base}$, где n_{hidden}^{base} – минимальное число нейронов, при котором задача решается с приемлемой точностью, а k – некоторая константа (в диапазоне 5-10).

Зам. Поскольку при каждом новом запуске обучения выполняется случайная инициализация НС, для более точного вычисления ошибки следует проводить эксперименты многократно и затем усреднять полученные показатели. Например, для числа скрытых нейронов n_{hidden} следует провести хотя бы $n_{run} = 10$ экспериментов, в каждом из них посчитать ошибку и затем вычислить среднее значение.

2.5.3.6. Пакетные алгоритмы обучения.

Изменяйте функции обучения на следующие:

- traingda, traingdm, traingdx, trainrp – градиентные алгоритмы

- traincgf, trainscg – метод сопряженных градиентов

- trainbfg, trainlm, trainoss – квазиньютоновские алгоритмы

Для каждой из функций обучения подберите параметры, обеспечивающие наилучшее качество аппроксимации исходной функции (см. прилагаемый к работе m-файл neuro_ff_example1).

!!! Признаком правильного подбора параметров (алгоритма обучения, обучающей выборки и архитектуры НС) является уменьшение ошибки по мере прохождения эпох обучения. Для градиентных алгоритмов сразу необходимо определить скорость обучения путем ее последовательного уменьшения в 10 раз до тех пор, пока ошибка не начнет уменьшаться.

Не забудьте при этом задать **достаточное** количество скрытых нейронов и **достаточный** объем обучающей выборки.

Приведите в отчете в таблице сводные результаты:

- полученные значения параметров для каждой из функций обучения (скорость обучения, момент, параметры адаптации для градиентных алгоритмов)
- число скрытых нейронов
- ошибку аппроксимации
- реальное количество эпох (время) обучения.

Для каждой функции обучения также постройте графики аппроксимации, соответствующие **наименьшей** ошибке аппроксимации. Можно строить по 3-4 функции на одном графике.

В данном и последующих пунктах применяйте сигнатуру для краткого описания архитектуры НС.

2.5.3.7. Выбор функции ошибки

В данном пункте необходимо будет рассмотреть различные функции производительности (mse, sse, mae, sae) и проанализировать то, как они влияют на конечный результат (качество) и на время обучения при использовании различных алгоритмов обучения (traingd-, trancg-, trainlm).

2.5.3.8. Выбор типа передаточных функций слоев НС

Целью данного пункта является настройка типов передаточных функций в слоях нейронной сети. Для скрытого слоя следует использовать только нелинейные дифференцируемые ПФ (tansig, logsig, satlin, satlins, RELU). Для выходного слоя можно использовать помимо нелинейных функций линейную (purelin).

Важно понимать, что ПФ выходного слоя определяет диапазон возможных выходных значений. И в случае, если диапазон возможных значений ПФ не совпадает с диапазоном значений выходных сигналов, необходимо заранее выполнить нормализацию выходных данных вручную, либо с использованием встроенного механизма нормализации NN Toolbox.

Для каждого сочетания функций следует подобрать оптимальное число скрытых нейронов на выбранном алгоритме обучения.

Сравните количество скрытых нейронов, которое потребовалось для достижения примерно одинакового качества на различных вариантах задания типов ПФ. Представьте результат в виде диаграмы.

2.5.3.9. Каскадная НС с одним скрытым слоем

Создайте каскадную НС с одним скрытым слоем (так, чтобы входной слой был связан и со скрытым и с входным). Можно воспользоваться функцией cascadeforwardnet.

Проведите несколько исследований по аналогии с п. 6, 7 и 8.

Постройте полученные график аппроксимации и приведите значение ошибки.

Сравните количество скрытых нейронов в обычной и каскадной НС ПР с одним скрытым слоем, при котором достигается примерно одинаковое качество.

2.5.3.10. Двухслойная, трехслойная НС прямого распространения.

Создайте варианты НС с двумя и тремя скрытыми слоями.

Определитесь с функцией ошибки.

Подберите

а) количество нейронов в скрытых слоях

б) передаточные функции во всех слоях

так, чтобы минимизировать ошибку аппроксимации.

Проведите обучение. Сравните количество скрытых нейронов в НС с 1 и 2 и 3 скрытыми слоями для различных конфигураций.

Попытайтесь добиться идеальных результатов.

Постройте полученные графики аппроксимации и приведите значение ошибки.

2.5.3.11. Сравнение результатов.

Сравните полученные результаты и представьте их в таблице, указав архитектуру НС (с количеством скрытых нейронов), реальное количество эпох обучения, время обучения, алгоритм обучения, среднюю ошибку (производительность) в конце обучения.

2.5.4 Классификация с помощью НС ПР

Решите задачу классификации линейно неразделимых образов (из заданий с номером 3). Для этого используйте НС ПР с 1, 2 и 3 скрытыми слоями. Рассмотрите обычную и каскадную архитектуры.

Рассматривайте различные варианты передаточных функций слоев и функций ошибок. Для скрытых слоев ПФ должны быть нелинейными. При использовании на выходе ПФ softmax хорошо подходит в качестве функции ошибки crossentropy.

Выходной слой может быть сформирован двумя способами:

-с сигмоидальной передаточной функцией logsig, функция производительности mse, mae, sae, sse.

- с передаточной функцией softmax, функция производительности crossentropy.

Важно понимать, что в данном случае НС на выходе будет выдавать некоторый набор значений для каждого из классов в диапазоне [0,1]. Чтобы преобразовать эти значения в результат классификации наиболее часто применяемый способ – выбрать класс с максимальным выходным значением.

Также часто для случая, когда все выходные нейроны выдают значение ниже порогового (небольшого), фиксируется результат – ничего не обнаружено. Данный вариант применительно к данной работе целесообразно использовать для задачи классификации многомерных образов.

Используйте в дальнейшем для обозначения сигнатуру при описании конкретной модели НС.

2.5.4.1. Модель с logsig-выходным слоем.

Рассмотрите модель 1-logsig. Для разных функций обучения (trainlm, trainbfg, traingdx, traincgf) определите оптимальное число нейронов в скрытом слое (слоях), настройте параметры функций обучения, обучите НС, вычислите ошибку аппроксимации. Постройте разделяющую кривую и диаграммы разбиения, реализуемую НС для наилучших, средних и наихудших случаев.

2. Модель с softmax-выходным слоем

Рассмотрите модель 1-softmax. Для этого измените выходной слой НС на softmax, функцию производительности на crossentropy. Подберите число скрытых нейронов, функцию обучения и обучите НС, подобрав наилучшее сочетание остальных параметров, представьте результаты и проанализируйте их.

2.5.4.3. Каскадная НСПР

Постройте модель каскадной НС типа 1-logsig-K или 1-softmax-K. Подберите число скрытых нейронов, выполните обучение НС, представьте полученные результаты и проанализируйте их.

2.5.4.4. Изменение типа передаточных функций скрытого слоя.

Попробуйте разные варианты ПФ для скрытого слоя (сигмоидальная tansig/logsig, линейная с насыщением – satlin/satlins/RELU). Для каждого варианта подберите число скрытых нейронов, выполните обучение НС, визуализируйте полученные результаты и проанализируйте их.

2.5.4.5. НС ПР с 2 и 3 скрытыми слоями.

Постройте несколько моделей НС с 2 и 3 скрытыми слоями. Подберите число скрытых нейронов в скрытых слоях, а также тип передаточных функций в скрытых слоях. Попытайтесь добиться идеальных результатов. Проведите обучение, представьте полученные результаты и сравните их.

2.5.4.6. Сравнение результатов.

Сравните полученные результаты и представьте их в таблице, указав подробную сигнатуру НС (с количеством скрытых нейронов), реальное количество эпох обучения, время обучения, алгоритм обучения, среднюю ошибку (производительность) в конце обучения.

2.5.5 Классификация для случая нескольких классов

1. В качестве исходных данных используйте задания с номером 4. Сформируйте обучающую и тестовую выборки.

Зам. Количество нейронов в выходном слое совпадает с количеством классов. Количество входов совпадает с количеством признаков, т.е. равно двум.

2. Выполните исследование по аналогии с заданием 4.

2.5.6. Классификация многомерных образов

1. В качестве исходных данных используйте задания с номером 8. Сформируйте обучающую и тестовую выборки достаточного объема. Продумайте 2-3 постановки задачи в зависимости от типа зашумления и искажения (равномерный шум, изменение масштаба, сдвиг, поворот).

Зам. Количество нейронов в выходном слое совпадает с количеством классов. Размерность входного сигнала совпадает с размерностью изображения (произведение числа пикселей по горизонтали и вертикали).

2. Выполните исследования по аналогии с заданием 4. В качестве моделей НС используйте варианты с 1, 2 и более скрытыми слоями, с каскадными связяи и без. Настраивайте передаточные функции скрытых слоев, количество скрытых нейронов и функцию ошибок.

Приведите найденные наилучшие конфигурации нейронных сетей, параметров обучения, а также соответствующие им результаты обучения – матрицы неточностей, ошибки 1,2 рода, среднюю ошибку классификации.

3. Представьте визуально результаты классификации. Для этого приведите примеры изображений каждого из классов и соответствующие им метки классов, выданные НС-классификатором. Постарайтесь в примерах показать, как правильные классификации, так и наиболее типовые ошибки.