МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Физические методы контроля»

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В НЕРАЗРУШАЮЩЕМ КОНТРОЛЕ

Методические рекомендации к практическим занятиям для магистрантов специальности 7-06-0716-03 «Приборостроение» очной и заочной форм обучения

Могилев 2026

УДК 620.179:004
ББК 34.9: 32.973.202
К63

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Физические методы контроля» «18» ноября 2025 г., протокол № 3

Составитель канд. техн. наук, доц. А. В. Кушнер

Рецензент канд. физ.-мат. наук С. О. Парашков

В методических рекомендациях кратко изложены теоретические сведения, необходимые для выполнения практических работ. Рекомендации составлены в соответствии с рабочей программой по дисциплине «Компьютерные технологии в неразрушающем контроле» для магистрантов специальности 7-06-0716-03 «Приборостроение».

Учебное издание

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В НЕРАЗРУШАЮЩЕМ КОНТРОЛЕ

Ответственный за выпуск	А. В. Хомченко
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

С ростом автоматизации возникла задача распознавания дефектов без участия человека. Данная задача может быть решена с использованием теории распознавания образов, которая заключается в классификации информации на основе определенных требований.

Целью практических занятий является ознакомление магистрантов с возможностью использования теории распознавания образов для автоматизации идентификации дефектов на изображениях.

Результатом выполнения практических заданий будет программа, выполненная в MatLab, которая позволит отличить изображения с дефектами от бездефектных изображений.

1 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Сбор данных

Цель работы: научиться собирать данные для последующей разработки нейронной сети.

1.1 Общие сведения

Сбор и подготовка данных для систем с машинным обучением обычно влечет за собой их представление в виде таблицы, если изначально они имеют другую форму. Предположим, что данные распределены по строкам и столбцам, причем каждая строка соответствует изучаемому экземпляру, а столбец – значению этого экземпляра. Существует несколько исключений, но можно смело утверждать, что большинство алгоритмов машинного обучения требует данных именно в таком формате. На рисунке 1.1 приведен простой набор данных, представленных в виде таблицы, строки называются экземплярами, а столбцы представляют собой признаки.

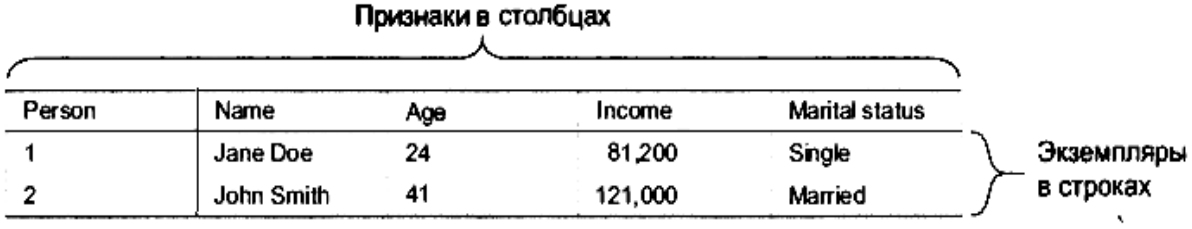


Рисунок 1.1 – Набор данных

Как видно из рисунка, столбцы содержат данные одного типа, а данные в строках принадлежат разным типам. В таблице представлены данные четырех типов: строковая переменная Name, целочисленная переменная Age, вещественная переменная Income, категориальная переменная Marital status (дискретное число значений). Такой набор данных называют гетерогенным.

Реальные данные могут быть «запутаны» разными способами, например, если на этапе сбора данных измерить какое-то значение не представляется возможным. В подобных случаях некоторые ячейки таблицы остаются незаполненными, что усложняет как построение модели, так и последующее прогнозирование.

Первое, что необходимо сделать, это корректно сформулировать вопрос, который будет решаться с помощью машинного обучения. Большинство реальных задач сводится к предсказанию целевой переменной (или переменных).

Для машинного обучения с учителем вопросы будут иметь общие черты.

Во-первых, все они требуют оценки одного или нескольких экземпляров. Роль экземпляров могут играть люди, события или даже временные отрезки.

Во-вторых, в каждом вопросе должна быть четко прописана желательная форма ответа. Иногда это бинарный ответ, иногда принадлежит к набору клас-

сов и тысячам классов, а иногда принимает численное значение. В статистике и теории вычислительных систем целевая переменная называется откликом, или зависимой переменной.

В-третьих, для каждой из поставленных задач можно получить набор данных за прошедшие периоды с известной целевой переменной. В дополнение к известным целевым значениям файлы с данными за прошедшие периоды будут содержать информацию для каждого экземпляра, который будет доступен для изучения на момент создания прогноза.

В задачах, связанных с машинным обучением, как правило, присутствуют десятки признаков, которые можно использовать для предсказания целевой переменной. Существует два реальных ограничения на использование сведений в качестве входного признака:

1) значение признака должно быть известно на момент прогноза;

2) признак должен быть численным или категориальным.

Простое эмпирическое правило гласит, что признак имеет смысл использовать только если вы подозреваете, что он каким-то образом связан с целевой переменной. Поскольку целью машинного обучения с учителем является предсказание целевой переменной, то признаки, не имеющие с ней ничего общего, требуется исключить. Чем больше неинформативных признаков, тем ниже соотношение «сигнал – шум» и менее точна модель. Но при этом точности модели может повредить исключение признака, о связи которого с целевой переменной изначально ничего не известно.

Поэтому лучше всего поступать следующим образом.

1 Включить все признаки, которые кажутся хоть как-то связанными с целевой переменной. Выполнить обучение модели. Если точность прогноза вас устраивает, то останавливайтесь.

2 В противном случае, расширьте набор, добавив туда признаки, связь которых с целевой переменной. Если точность прогноза вас устраивает, то останавливайтесь.

3 Если точность все еще неудовлетворительна, запустите для расширенного набора признаков алгоритм отбора, чтобы выбрать оптимальное, сильнее всего влияющее на процесс прогнозирования, множество.

Теперь рассмотрим анализ изображений для последующего обучения алгоритма распознавания образов. Рассматривать будем на примере либо магнитопорошкового метода контроля, либо магнитного контроля с использованием визуализирующей магнитные поля пленки. Для обучающей и тестовой выборки необходимо сделать достаточно большое количество разных изображений как с дефектами, так и без дефектов, причем объекты контроля должны быть разные и снятые с разных ракурсов. Изображения должны выглядеть, как показано на рисунке 1.2.

Практическое задание

По заданию преподавателя осуществить сбор данных, на основе которых в дальнейшем будет разрабатываться нейронная сеть для распознавания дефектов

на изображениях. Количество изображений как с дефектами, так и без дефектов должно быть не менее 100 шт., при этом они должны быть в разных ракурсах.

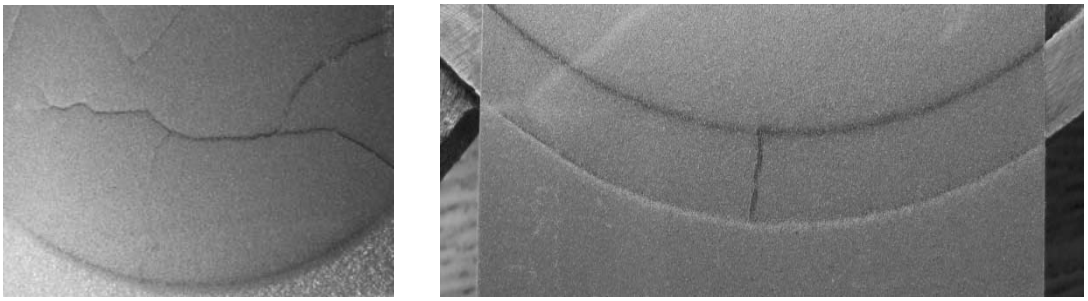


Рисунок 1.2 – Примеры изображений для обучения нейронной сети

Контрольные вопросы

1 Как осуществляется сбор данных?
2 В каком виде могут быть данные?
3 Что такое признаки?
4 Что такое экземпляры?

2 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Обработка данных

Цель работы: научиться обрабатывать данные перед разработкой нейронной сети.

2.1 Общие сведения

Первый этап подготовки данных к моделированию – это их сбор, но иногда таких этапов может быть несколько, в зависимости от структуры набора данных. Многие алгоритмы машинного обучения работают только с численными данными – целыми и вещественными. Самые простые модели имеют именно такой набор данных, но зачастую в набор входят признаки и других типов, например, категориальные переменные, или же часть значений может просто отсутствовать. Иногда признаки приходится проектировать или рассчитывать.

Признак считается категориальным, если его значения можно отнести к какой-то группе, при этом нам не важен их порядок. Иногда принадлежность к этому типу определяется легко, но бывает, что разница между численными и категориальными не очевидна. Любой из типов может оказаться правильным представлением и при этом выбор влияет на производительность модели. Например, день недели допустимо представить как в виде числа (пронумеровав дни, начиная с воскресенья), так и в виде категории (взяв за основу названия). На рисунке 2.1 показаны такие признаки в нескольких наборах. Вверху набор

данных о заявителях, в котором категориальным является столбец со сведениями о семейном положении. Внизу набор сведений о пассажирах «Титаника». К категориальным здесь относятся столбцы с информацией о том, остался ли пассажир в живых, каким классом он путешествовал, женщина он или мужчина и в каком городе пассажир сел на «Титаник».

Person	Name	Age	Income	Marital status
1	Jane Doe	24	81,200	Single
2	John Smith	41	121,000	Married

Категориальные признаки

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

Рисунок 2.1 – Определение категориальных признаков

Некоторые алгоритмы машинного обучения используют категориальные признаки в исходном виде, но обычно данные требуется представить в численной форме. Можно присвоить каждой категории номер, но такие зашифрованные данные нельзя использовать в качестве истинных категориальных признаков, т. к. при этом добавляется случайным образом выбранный вами порядок категорий. А одним из свойств категориальных признаков является их неупорядоченность. Поэтому лучше преобразовать каждую категорию в отдельный двоичный признак, имеющий значение 1 для экземпляров, попадающих в категорию, и 0 – для не попадающих. В результате каждый категориальный признак преобразуется в набор двоичных признаков, по одному на категорию. Сконструированные таким образом признаки иногда называют вспомогательными переменными (рисунок 2.2).

Описанная техника преобразования категориальных признаков в численные работает для большинства алгоритмов машинного обучения. Но при этом существуют алгоритмы, которые представляют собой категориальные признаки в исходном виде. В случае использования исключительно категориальных наборов данных они часто дают очень хорошие результаты.

Простой набор данных о заявителях после преобразования показан на рисунке 2.3.

Теперь рассмотрим, что делать, если данные отсутствуют. В табличном представлении такие данные принимают вид пустых ячеек или ячеек со значением NaN (Not of Number) или None. Как правило, это артефакты

процесса сбора, возникающие, когда по какой-либо причине конкретное значение для экземпляра данных измерить невозможно. Например, отсутствующие данные о пассажирах «Титаника» (рисунок 2.4). На рисунке представлен набор сведений о пассажирах «Титаника», имеющий отсутствующие значения в столбцах «Возраст» и «Каюта». Информация о пассажирах извлекалась из доступных исторических источников, поэтому в данных случаях интересующие сведения просто не были обнаружены.

Рисунок 2.2 – Преобразование категориальных столбцов в численные

Person	Name	Age	Income	Marital status: Single	Marital status: Married
1	Jane Doe	24	81,200	1	0
2	John Smith	41	121,000	0	1

Рисунок 2.3 – Простой набор данных после преобразования категориального признака «Семейное положение» в двоичные численные признаки

Обработка двух основных типов отсутствующих данных происходит по-разному. В случае, когда сам факт отсутствия данных может быть информацией полезной для алгоритма машинного обучения. Рассмотрим сначала информативные отсутствующие данные. Если нам кажется, что в данных не хватает какой-то информации, значит мы не хотим, чтобы алгоритм машинного обучения смог ей воспользоваться и теоретически увеличить точность предсказаний. Для этого отсутствующее значение нужно привести к формату остальных значений в столбце. Для числовых столбцов это осуществляется путем присвоения значения -1 или -999. При этом необходимо выбирать для отсутствующих значений число с одного из концов числового спектра, для численных столбцов важен порядок следования.

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

Отсутствующие значения

Рисунок 2.4 – Набор сведений о пассажирах «Титаника»

Для категориальных столбцов с потенциально информативными отсутствующими данными можно создать новую категорию, назвав ее Missing, None, или другим подобным образом, после чего новый категориальный признак обрабатывается обычным способом. Схематично работа с отсутствующими значимыми данными представлена на рисунке 2.5.

Рисунок 2.5 – Действия при отсутствии значимых данных

Если же отсутствие значений отдельных экземпляров не несет информационной нагрузки, то порядок действий будет другим. Данные необходимо предварительно обработать, вообще убрав отсутствующие значения или заменив их некоторым оценочным значением. Такой подход к обработке отсутствующих данных называется заполнением пропусков. В случае большого набора данных с небольшим количеством отсутствующих значений самое простое – отбросить такие данные.

Другой подход используется, если можно предположить, что экземпляры данных обладают некой временной упорядоченностью, и вставить на места пропусков значения из предшествующей строки этого столбца. Мы предполагаем, что при переходе от одного экземпляра к другому результат измерений не изменился. Зачастую подобное предположение некорректно. Но еще менее

корректно заполнять пустые ячейки нулями, особенно, если данные представляют собой набор последовательных наблюдений.

Когда возможно лучше использовать большие фрагменты существующих данных для приблизительной оценки отсутствующих значений, их можно заменить средним значением или медианой столбца. При отсутствии другой информации предполагается, что среднее значение ближе всего к истине. Но в зависимости от распределения значений в столбце иногда лучше использовать медиану; среднее значение чувствительно к выбросам. Но заменяя отсутствующие данные одним новым значением, вы снижаете наблюдаемость потенциальной корреляции с другими переменными.

В результате перечисленные техники можно схематично представить в виде следующего алгоритма (рисунок 2.6).

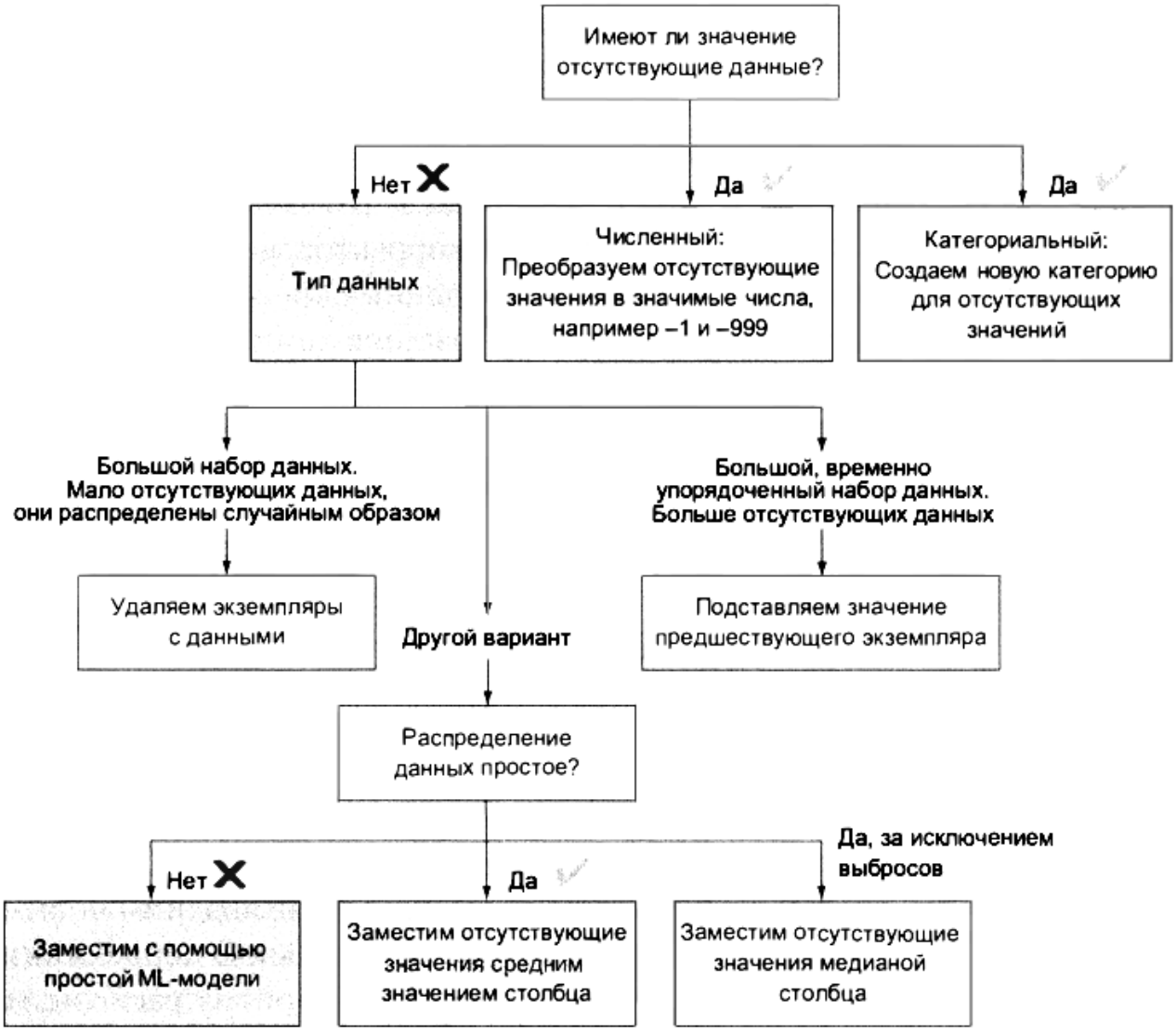


Рисунок 2.6 – Алгоритм решений для обработки отсутствующих значений при подготовке данных к моделированию

Рассмотрим основы проектирования признаков, для чего воспользуемся сведениями о пассажирах «Титаника» (рисунок 2.7).

PassengerId	Survived	Pclass	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Male	22	1	0	A/5 21171	7.25		S
2	1	1	Female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Female	26	0	0	STON/02. 3101282	7.925		S
4	1	1	Female	35	1	0	113803	53.1	C123	S
5	0	3	Male	35	0	0	373450	8.05		S
6	0	3	Male		0	0	330877	8.4583		Q

Рисунок 2.7 – Сведения о пассажирах «Титаника»

В сведениях о пассажирах «Титаника» некоторые ячейки столбца «Cabin» содержат несколько значений, в то время как другие пусты. Сам по себе номер каюты не является хорошим категориальным признаком. В этот раз нас интересует такой признак, как «Cabin» (каюта). Без обработки этот признак вряд ли пригоден для использования. Так как некоторые значения включат в себя несколько кают, но даже если там всего одна каюта, то он вряд ли подойдет в качестве хорошего категориального признака, т. к. для каждой каюты придется завести отдельную категорию. Поэтому будет невозможно провести корреляцию между фактом спасения пассажира и тем, что он занимал именно свою каюту, а не соседнюю. Проживание же в определенной части корабля может оказаться важным фактором выживания. Из идентификатора каюты можно извлечь букву, как категориальный признак, и номер, как числовой признак, предположив, что это соответствует различным частям корабля. Можно найти план «Титаника» и определить, в каком классе и с какой стороны располагалась каюта, имела она иллюминаторы или нет и т. д. Набор кают этим способом обработать нельзя, но т. к. с большой вероятностью все каюты заняты одним пассажиром, располагаются рядом, то достаточно взять для анализа ID первую из них. Можно включить номера кают в новый признак, который тоже будет релевантным. В результате мы создаем из признака «Cabin» три новых признака.

Некоторые алгоритмы машинного обучения требуют, чтобы данные были нормализованными, т. е. каждый признак обрабатывается с целью его подгонки под единую числовую шкалу. Диапазон значений признака может влиять на его важность относительно прочих признаков. Если значения одного признака варьируются от 0 до 10, а второго – от 0 до 1, вес первого признака по отношению ко второму составит 10. Иногда вес конкретного признака регулируется вручную, но лучше оставить определение относительных весов на откуп алгоритму машинного обучения.

Поэтому, чтобы гарантировать, что все признаки учитываются одинаково, данные требуется нормализовать.

На основе снятых в практической работе № 1 изображений рассмотрим, как выделять необходимые для распознавания объектов признаки. Для распознавания искомых деталей на изображении зрительно один образ можно отличить от другого по следующим характеристикам: цвет, текстура, форма и размер. Эти характеристики необходимо представить на понятном для компью-

тера языке. Для формализации цвета можно воспользоваться гистограммой цвета. Это графическая зависимость: по оси абсцисс – интенсивность цвета, по оси ординат – частота повторяемости цвета. Но так как в большинстве своем гистограммы наших изображений будут совпадать, то для решения проблемы идентификации дефекта не следует ограничиваться признаками совпадения гистограмм цвета. Чтобы сравнивать гистограммы, можно воспользоваться такими статистическими характеристиками, как математическое ожидание, дисперсия, ассиметрия, аксцесс и коэффициент корреляции.

Текстура – это рисунок, образуемый на поверхности объекта. Она характеризуется тем, что на рисунке обычно присутствуют повторения каких-либо специфических для объекта особенностей. Для определения текстур можно воспользоваться Фурье-анализом. В качестве признаков идентификации текстур могут служить амплитуды гармоник c_k, на которые раскладывается дискретная функция изменения сигнала цвета в сечении изображения $y(t)$:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos k\omega t + b_k \sin k\omega t \right) = \frac{a_0}{2} + \sum_{k=1}^{\infty} c_k \sin\left(k\omega t + \varphi_k \right),$$

где a_0 – постоянная составляющая или нулевая гармоника;

$$c_k = \sqrt{a_k^2 + b_k^2};$$

$$a_k \approx \frac{2}{N} \sum_{i=1}^{N} y_i \cos\left(k \frac{2\pi}{N} i \right);$$

$$b_k \approx \frac{2}{N} \sum_{i=1}^{N} y_i \sin\left(k \frac{2\pi}{N} i \right).$$

Для нормализации процесса сопоставления объектов необходимо выбрать оптимальные для множества объектов период первой гармоники $T = N$ и количество гармоник m. При выборе количества гармоник должен быть обеспечен компромисс между простотой описания и информативностью параметров, выбираемых для идентификации.

Первая гармоника, так же как и последующие, должна отображать реальные колебания цвета на объекте. Поэтому период гармоники $T = N$ не должен быть меньше самого большого периода колебаний, определяемого для одного из объектов множества. При представлении функции дискретными значениями номер наивысшей гармоники m должен выбираться из условия $m < N/5$ (количество точек на период одной гармоники должно быть не менее пяти). Соответственно, для $N = 50$ количество гармоник не должно превышать 10.

Выделение признаков формы можно разделить на несколько этапов:

1) выделение граничных точек объекта;

2) построение сигнатуры области;

3) определение признаков формы.

Рассмотрим выделение граничных точек объекта. Учитывая достаточную информативность участков контура объектов, которые расположены на границе

с фоном, будем выделять только эти участки контурных линий. Точки линий контура характеризуются высоким градиентом изменения интенсивности цвета. Однако, если при выборе точек контура руководствоваться только лишь этим критерием, то можно ошибочно выбирать точки продукта, где резко меняется цвет. Следует определить еще дополнительные критерии линий контура.

Посмотрим, чем отличаются точки фона (серого цвета) от точек объекта. Характерной особенностью серого цвета является относительное равенство сигналов Red, Green и Blue. Эта особенность может быть учтена при создании алгоритма выделения граничных точек объекта. Кроме условия высокого градиента изменения интенсивности цвета, можно добавить условие наличия слева (или справа), по крайней мере, пяти точек с относительно равными сигналами R, G, B. Это условие позволяет определить с какой стороны находится объект и, соответственно, однозначно определить такие признаки формы, как выпуклость или вогнутость. Прежде чем определять эти и другие признаки, необходимо построить сигнатуры участков контура.

Сигнатурой называется функциональное одномерное представление границы. Учитывая, что точки растрового изображения упорядочены по рядам снизу вверх, предлагается следующий алгоритм построения сигнатуры. При считывании точек каждого ряда осуществляется проверка на близость обнаруженных граничных точек текущего ряда с граничными точками предыдущего ряда. В случае обнаружения в предыдущем ряде близкой точки обе точки запоминаются, в дальнейшем к ним может присоединиться близкая граничная точка в следующем ряду и т. д. – формируется одномерный массив из точек границы. Подобным образом формируется несколько сигнатур в соответствии с количеством обнаруженных вдоль ряда граничных точек. Следует заметить, что практически сложно получить непрерывное одномерное представление границы объекта из-за размытости границы на определенных участках. Поэтому запоминаются лишь куски контуров, причем только куски достаточной для последующего анализа длины.

Полученные сигнатуры областей (куски контуров) необходимо обработать на предмет выделения признаков формы. Один из вариантов такой обработки – построение гистограммы изменения значения угла касательной на длине участка AL / DS (функции плотности наклона касательной) с последующим выделением на основе ее информативных признаков формы (рисунки 2.8 и 2.9).

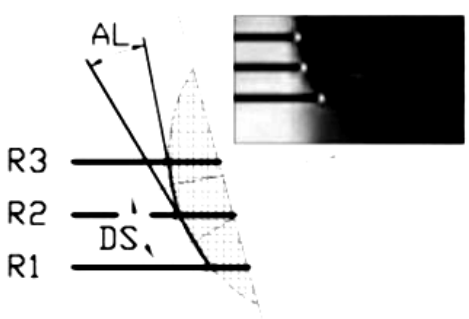


Рисунок 2.8 – Схема изменения угла касательной

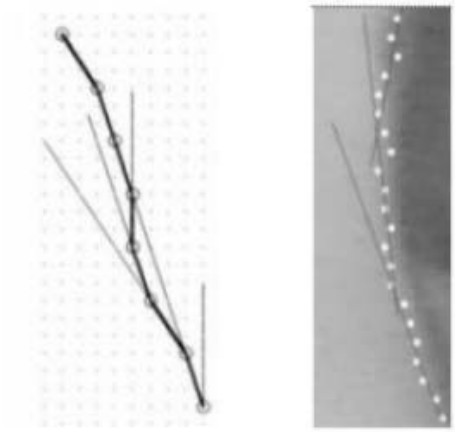


Рисунок 2.9 – Изменение угла касательных через соседние точки

Как видно из рисунка 2.9, нельзя отследить для реальной картины изменение угла касательной по контуру, если определять угол по соседним точкам, поскольку на обнаружение точек контура влияло множество случайных факторов (нехарактерные для объекта изгибы, теневые участки и т. п.).

Чтобы уменьшить влияние случайных факторов, можно выполнить один из вариантов статистической обработки.

1 Аппроксимировать точки кривой 2-го порядка и определить для нее характеристики кривизны.

2 Разделить точки на последовательные группы (из 10…15 точек), каждую аппроксимировать прямой (см. рисунок 2.9), и затем уже для этих прямых определить значение угла AL.

В данном алгоритме имеется проблема – сигнатура при переходе от контура одного объекта к контуру другого не прерывается. А поскольку на переходных участках изменение угла касательной не характеризует особенности формы объекта, то необходимо такие участки пропускать. Эта проблема легко устраняется в альтернативном алгоритме, который основан на последовательном определении радиуса кривизны (рисунок 2.10) графическим способом.

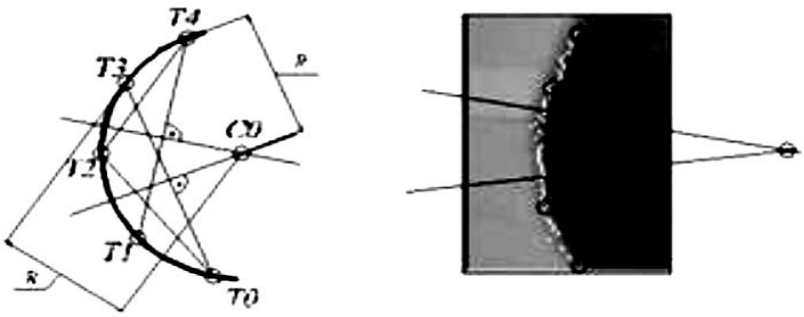


Рисунок 2.10 – Определение радиуса кривизны графическим способом

По этой схемы текущий радиус кривизны вдоль контура определяется в следующей последовательности.

1 При формировании цепочки из точек контура через определенный интервал по вертикали фиксируем пять точек T(4).

2 Для пар точек (T0–T3,T1–T4) и (T0–T2, T2–T4) находим два центра.

3 Берем средние координаты центра XC0 = (XC1 + XC2)/2 и YC0 = (YC1 + + YC2)/2 и находим дисперсию. Если она не превышает порога, принимаем выборку из точек к рассмотрению, поскольку точки определяют переход от одного объекта к другому (см. рисунок 2.10).

4 Находим пять радиусов – расстояние от центра C0 к точкам (T0, T1, …, T4).

5 Определяем средний радиус.

6 Осуществляем сдвиг точек контура на позицию 1, точка T1 становится T0.

При дальнейшей обработке для определения признаков формы можно построить гистограмму значений кривизны по длине участка.

На шкале кривизны (1/R) следует предусмотреть диапазоны, которые характеризуют анализируемую совокупность объектов (например: вогнутость, прямолинейность, большой радиус, маленький радиус, радиус, радиус маленьких закруглений некоторых объектов).

Здесь вогнутость можно условно охарактеризовать отрицательным радиусом. Радиус считается отрицательным в том случае, если центр окружности расположен в противоположной от продукта стороне. Прямолинейность определяется предельно большими радиусами (кривизна близка к нулю), в том числе и отрицательными.

Теперь для идентификации объектов осталось определить критерий, по которому можно сопоставить гистограммы. По сути, задача сводится к выбору критерия, по которому определяется близость рядов чисел. Одним из простейших способов решения задачи является определение суммарного рассогласования между соответствующими числами двух рядов.

Практическое задание

Осуществить обработку данных, собранных в практической работе № 1 для последующего построения нейронной сети, которая будет осуществлять распознавание изображений с дефектами. Для чего необходимо выбрать систему признаков, характерных для этой задачи, и преобразовать данные соответствующим образом для подачи на вход сети (нормировка, стандартизация и т. д.). В результате желательно получить линейно отделяемое пространство множества образцов.

Контрольные вопросы

1 Какие признаки бывают?

2 Можно ли категориальные признаки представить в численной форме?

3 Какие подходы бывают при обработке данных?

4 Объясните алгоритм решений для обработки отсутствующих значений при подготовке данных к моделированию.

3 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Разбиение данных на обучающую и тестовую выборки

Цель работы: научиться формировать данные для обучающей и тестовой выборок при разработке нейронной сети.

3.1 Общие сведения

При построении предсказательных моделей исходные данные обычно разбиваются на обучающую и тестовую выборки. Обучающая выборка используется для обучения той или иной модели, т. е. для построения математических соотношений между переменной-откликом и предикторами, в то время как тестовая выборка служит для получения оценки прогнозных свойств модели на новых данных, т. е. данных, которые не были использованы для обучения модели.

Получение примеров для обучающей и тестовой выборок является серьезной проблемой построения нейросетевых моделей. В большинстве случаев моделируемый объект неизвестен и требуется выполнение экспериментов для нахождения таких примеров.

Если есть возможность экспериментатору задавать значения входных переменных объекта, то возникает задача выбора этих значений, т. е. точек, в которых измеряется выходная переменная. При этом обучающая выборка должна быть представительной, охватывать различные сочетания значений входных переменных, а сами значения должны равномерно распределяться в своих диапазонах. Обучающие примеры должны правильно характеризовать функцию $y = f(x_1, x_2, \ldots, x_n)$, которая, вообще говоря, неизвестна, или о которой имеется лишь общее представление.

Необходимо заметить, что вся информация об объекте, который моделируется нейронной сетью, содержится в наборе примеров. Поэтому качество построенной нейросетевой модели непосредственно зависит от того, насколько полно эти примеры описывают объект.

На сегодняшний день не существует строгих рекомендаций по поводу размерности обучающей выборки. Приводятся примерные оценки количества обучающих пар L выборки:

$$L = \frac{W}{\varepsilon},$$

где W – количество настраиваемых параметров сети (количество весов);

ε – погрешность обучения.

Поэтому предлагается использовать принципы теории планирования эксперимента для получения эффективных пар «вход – выход» как для обучающей, так и для тестовой выборки.

1 Принцип последовательного эксперимента – размеры обучающей и тес-

товой выборок постепенно увеличиваются до тех пор, пока не будет достигнуто обучение заданного качества. Это позволит в случае неудачного обучения доработать обучающую выборку, а не создавать ее заново.

2 Нормализация факторов, т. е. преобразование натуральных значений факторов в безразмерные. Разные факторы имеют различную размерность и диапазоны изменения значений. Числовые значения этих факторов могут существенно различаться. Факторы с большими числовыми значениями могут в большей мере влиять на выходы и «забивать» остальные. Это особенно заметно, когда нейроны сети имеют одинаковые функции активации. Использование безразмерных значений факторов при обучении уменьшает возможность появления «паралича» сети и приводит к более высокой скорости обучения.

3 Точки проведения опытов определяются в соответствии с полными и дробными факторными экспериментами ПФЭ 2^n и ДФЭ 2^{n-k}. Это позволяет избежать необходимости проведения избыточного количества опытов при большом числе факторов (семь и более) и гарантирует, что не будет пропущено ни одного возможного сочетания факторов и не будет повторения одинаковых опытов. В случае недостаточности примеров полного факторного эксперимента 2^n можно переходить на тот же план, но с другими диапазонами варьирования факторов.

Общий алгоритм последовательного формирования обучающей и тестовой выборок при обучении нейронной сети показан на рисунке 3.1.

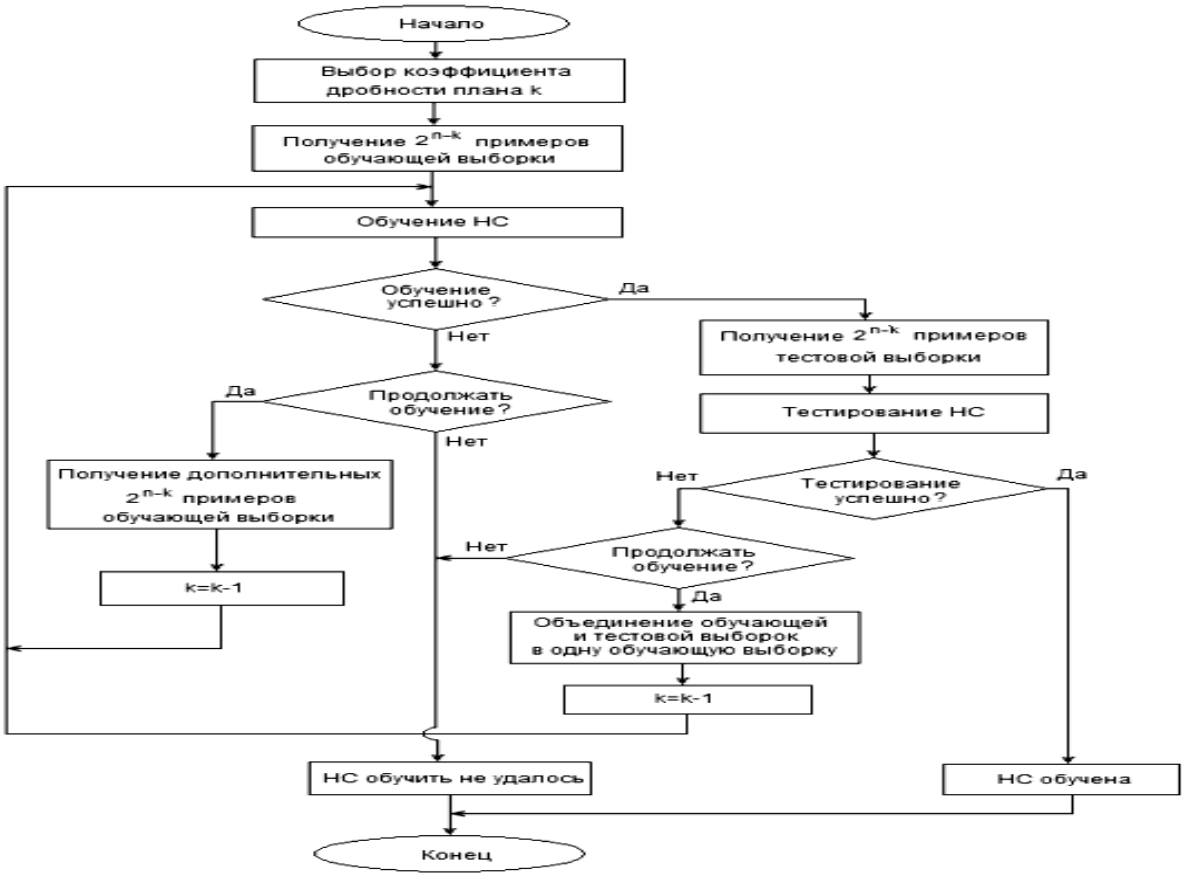


Рисунок 3.1 – Алгоритм последовательного формирования обучающей и тестовой выборок при обучении нейронной сети

Практическое задание

Осуществить разбиение обработанных данных, собранных в практической работе № 1, на обучающую и тестовую выборки.

Контрольные вопросы

1 Что такое обучающая выборка?
2 Что такое тестовая выборка?
3 Как проводится разбиение на обучающую и тестовую выборки?

4 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Разработка архитектуры нейронной сети

Цель работы: научиться выбирать наиболее подходящую архитектуру нейронной сети.

4.1 Общие сведения

Для формирования нейронных сетей используются следующие условные обозначения узлов (рисунок 4.1).

Нейронные сети прямого распространения (*feed forward neural networks*, FF или FFNN) и перцептроны (*perceptrons*, P) очень прямолинейны, они передают информацию от входа к выходу (рисунок 4.2). Нейронные сети часто описываются в виде слоёного торта, где каждый слой состоит из входных, скрытых или выходных клеток. Клетки одного слоя не связаны между собой, а соседние слои обычно полностью связаны. Самая простая нейронная сеть имеет две входных клетки и одну выходную и может использоваться в качестве модели логических вентилей. FFNN обычно обучается по методу обратного распространения ошибки, в котором сеть получает множество входных и выходных данных. Этот процесс называется обучением с учителем и он отличается от обучения без учителя тем, что во втором случае множество выходных данных сеть составляет самостоятельно. Под ошибкой обычно понимаются различные степени отклонения выходных данных от исходных. Если у сети есть достаточное количество скрытых нейронов, она теоретически способна смоделировать взаимодействие между входными и выходными данными. Практически такие сети используются редко, но их часто комбинируют с другими типами для получения новых.

Сети **радиально-базисных функций** (*radial basis function*, RBF) – это FFNN, которая использует радиальные базисные функции, как функции активации (рисунок 4.3). Больше она ничем не выделяется.

Нейронная сеть Хопфилда (*Hopfield network*, HN) – это полносвязная нейронная сеть с симметричной матрицей связей (рисунок 4.4). Во время

получения входных данных каждый узел является входом, в процессе обучения он становится скрытым, а затем становится выходом.

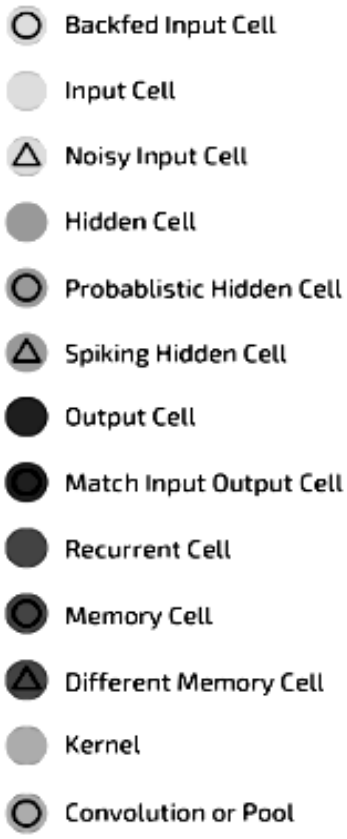


Рисунок 4.1 – Расшифровка узлов нейронной сети

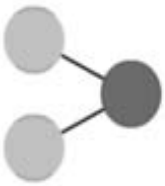
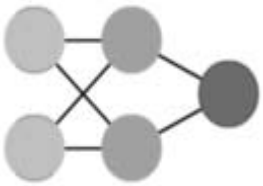


Рисунок 4.2 – Нейронная сеть прямого распространения



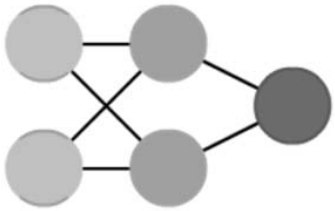


Рисунок 4.3 – Нейронная сеть радиально-базисных функций

Сеть обучается так: значения нейронов устанавливаются в соответствии с желаемым шаблоном, после чего вычисляются веса, которые в дальнейшем не меняются. После того как сеть обучилась на одном или нескольких шаблонах, она всегда будет сводиться к одному из них (но не всегда к желаемому). Она стабилизируется в зависимости от общей «энергии» и «температуры» сети. У каждого нейрона есть свой порог активации, зависящий от температуры, при прохождении которого нейрон принимает одно из двух значений (обычно -1 или 1, иногда 0 или 1). Такая сеть часто называется сетью с ассоциативной памятью; как человек, видя половину таблицы, может представить вторую половину таблицы, так и эта сеть, получая таблицу, наполовину зашумленную,

восстанавливает её до полной.

Цепи Маркова (*Markov chains*, MC или *discrete time Markov Chains*, DTMC) – это предшественники машин Больцмана (BM) и сетей Хопфилда (HN) (рисунок 4.5). Их смысл можно объяснить так: каковы мои шансы попасть в один из следующих узлов, если я нахожусь в данном.

Каждое следующее состояние зависит только от предыдущего. Хотя на самом деле цепи Маркова не являются НС, они весьма похожи. Также цепи Маркова не обязательно полносвязны.

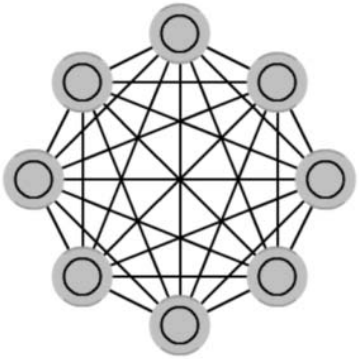


Рисунок 4.4 – Нейронная сеть Хопфилда

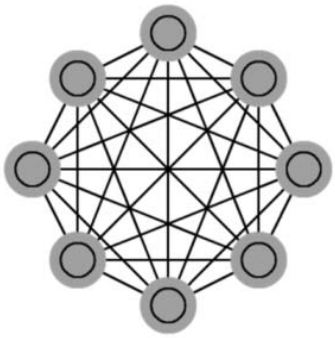


Рисунок 4.5 – Нейронная сеть цепи Маркова

Машина Больцмана (*Boltzmann machine*, BM) очень похожа на сеть Хопфилда, но в ней некоторые нейроны помечены как входные, а некоторые как скрытые (рисунок 4.6).

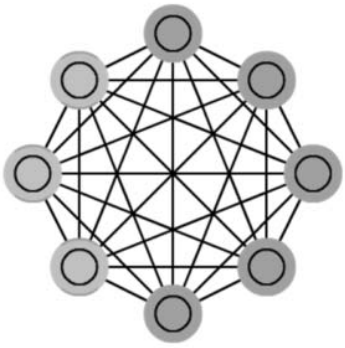


Рисунок 4.6 – Нейронная сеть Машина Больцмана

Входные нейроны в дальнейшем становятся выходными. Машина Больцмана – это стохастическая сеть. Обучение проходит по методу обратного распространения ошибки или по алгоритму сравнительной расходимости. В целом процесс обучения очень похож на таковой у сети Хопфилда.

Ограниченная машина Больцмана (*restricted Boltzmann machine*, RBM) удивительно похожа на машину Больцмана и, следовательно, на сеть Хопфилда (рисунок 4.7). Единственной разницей является её ограниченность. В ней нейроны одного типа не связаны между собой. Ограниченную машину Больц-

мана можно обучать как FFNN, но с одним нюансом: вместо прямой передачи данных и обратного распространения ошибки нужно передавать данные сперва в прямом направлении, затем в обратном. После этого проходит обучение по методу прямого и обратного распространения ошибки.

Автокодировщик (*autoencoder*, AE) чем-то похож на FFNN, т. к. это скорее другой способ использования FFNN, нежели фундаментально другая архитектура (рисунок 4.8). Основной идеей является автоматическое кодирование (в смысле сжатия, не шифрования) информации. Сама сеть по форме напоминает песочные часы, в ней скрытые слои меньше входного и выходного, причём она симметрична. Сеть можно обучить методом обратного распространения ошибки, подавая входные данные и задавая ошибку, равную разнице между входом и выходом.

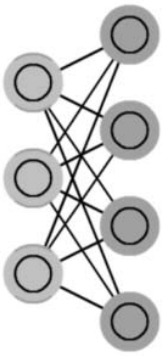


Рисунок 4.7 – Нейронная сеть Ограниченная машина Больцмана

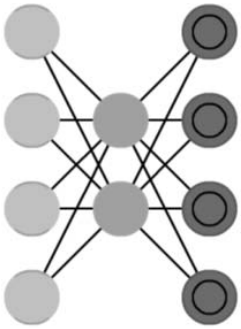


Рисунок 4.8 – Нейронная сеть Автокодировщик

Разреженный автокодировщик (*sparse autoencoder*, SAE) – в каком-то смысле противоположность обычного (рисунок 4.9). Вместо того, чтобы обучать сеть отображать информацию в меньшем «объёме» узлов, мы увеличиваем их количество. Вместо того, чтобы сужаться к центру, сеть там раздувается. Сети такого типа полезны для работы с большим количеством мелких свойств набора данных. Если обучать сеть как обычный автокодировщик, ничего полезного не выйдет. Поэтому кроме входных данных, подаётся ещё и специальный фильтр разреженности, который пропускает только определённые ошибки.

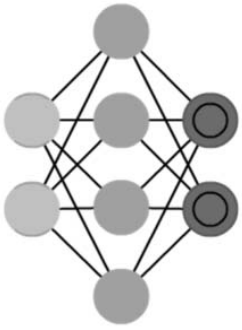


Рисунок 4.9 – Нейронная сеть Разреженный автокодировщик

Вариационные автокодировщики (*variational autoencoder*, VAE) обладают схожей с AE архитектурой, но обучают их иному: приближению вероятностного распределения входных образцов (рисунок 4.10). В этом они берут начало от машин Больцмана. Тем не менее, они опираются на байесовскую математику, когда речь идёт о вероятностных выводах и независимости, которые интуитивно понятны, но сложны в реализации. Если обобщить, то можно сказать что эта сеть принимает в расчёт влияния нейронов. Если что-то одно происходит в одном месте, а что-то другое – в другом, то эти события не обязательно связаны, и это должно учитываться.

Шумоподавляющие автокодировщики (*denoising autoencoder*, DAE) – это AE, в которые входные данные подаются в зашумленном состоянии (рисунок 4.11). Ошибку мы вычисляем также, и выходные данные сравниваются с зашумленными. Благодаря этому сеть учится обращать внимание на более широкие свойства, поскольку маленькие могут изменяться вместе с шумом.

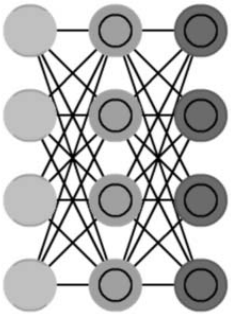


Рисунок 4.10 – Нейронная сеть Вариационный автокодировщик

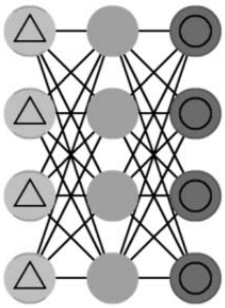


Рисунок 4.11 – Нейронная сеть Шумоподавляющий автокодировщик

Сеть типа «deep belief» (*deep belief networks*, DBN) – это название, которое получил тип архитектуры, в которой сеть состоит из нескольких соединённых RBM или VAE (рисунок 4.12). Такие сети обучаются поблочно, причём каждому блоку требуется лишь уметь закодировать предыдущий. Такая техника называется «жадным обучением», которая заключается в выборе локальных оптимальных решений, не гарантирующих оптимальный конечный результат. Также сеть можно обучить (методом обратного распространения ошибки) отображать данные в виде вероятностной модели. Если использовать обучение без учителя, стабилизированную модель можно использовать для генерации новых данных.

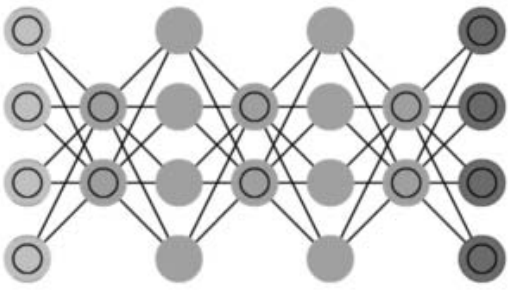


Рисунок 4.12 – Нейронная сеть типа «deep belief»

Свёрточные нейронные сети (*convolutional neural networks*, CNN) и **глубинные свёрточные нейронные сети** (*deep convolutional neural networks*, DCNN) сильно отличаются от других видов сетей (рисунок 4.13). Обычно они используются для обработки изображений, реже для аудио. Типичным способом применения CNN является классификация изображений: если на изображении есть кошка, сеть выдаст «кошка», если есть собака – «собака». Такие сети обычно используют «сканер», не просеивающий все данные за один раз. Например, если у вас есть изображение 200 × 200, вы не будете сразу обрабатывать все 40 тысяч пикселей. Вместо это сеть считает квадрат размера 20 × 20 (обычно из левого верхнего угла), затем сдвинется на один пиксель и считает новый квадрат и т. д. Эти входные данные затем передаются через свёрточные слои, в которых не все узлы соединены между собой. Эти слои имеют свойство сжиматься с глубиной, причём часто используются степени двойки: 32, 16, 8, 4, 2, 1. На практике к концу CNN прикрепляют FFNN для дальнейшей обработки данных. Такие сети называются глубинными (DCNN).

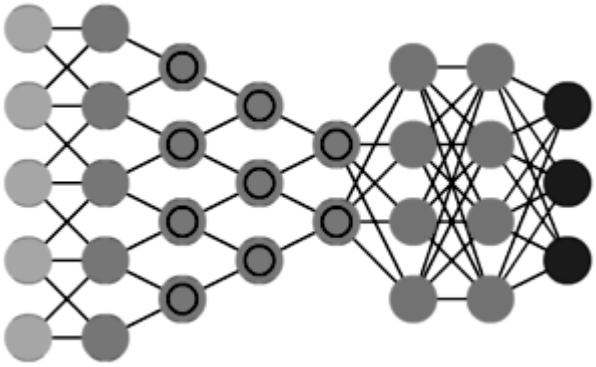


Рисунок 4.13 – Сверточная нейронная сеть

Развёртывающие нейронные сети (*deconvolutional networks*, DN), также называемые обратными графическими сетями, являются обратными к свёрточным нейронным сетям (рисунок 4.14). Представьте, что вы передаёте сети слово «кошка», а она генерирует картинки с кошками, похожие на реальные изображения котов. DNN тоже можно объединять с FFNN. Стоит заметить, что в большинстве случаев сети передаётся не строка, а какой бинарный вектор: например, <0, 1> – это кошка, <1, 0> – собака, а <1, 1> – и кошка, и собака.

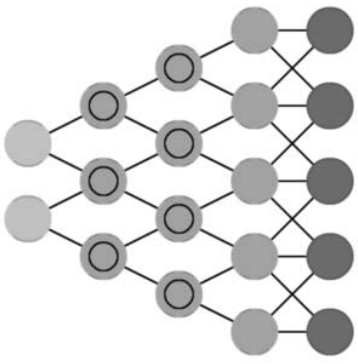


Рисунок 4.14 – Развертывающая нейронная сеть

Выбор архитектуры сети. При выборе архитектуры сети обычно опробуется несколько конфигураций с различным количеством элементов. При этом основным показателем является объем обучающего множества и обобщающая способность сети. Обычно используется алгоритм обучения обратного распространения с подтверждающим множеством.

Практическое задание

На основе данных из практических работ № 1 и 2 выбрать архитектуру нейронной сети для распознавания дефектов на изображении.

Контрольные вопросы

1 Что такое архитектура нейронной сети?

2 Какие бывают архитектуры нейронных сетей и чем они характеризуются?

5 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Программная реализация нейронной сети

Цель работы: реализовать выбранную нейронную сеть в программном пакете MatLab.

5.1 Общие сведения

В MatLab для реализации нейронных сетей используется утилита ntstool (рисунок 5.1).

На правой панели находится панель выбора типа нейронной сети:

1) нелинейная авторегрессионная с внешним входом;

2) нелинейная вход-выход (без обратных связей);

3) нелинейная авторегрессионная (генератор).

Однако с помощью программного кода можно реализовать большее количество архитектур сети.

Рассмотрим нелинейную авторегрессионную нейронную сеть с внешним входом (рисунок 5.2). Эта сеть является реализацией одного из простейших типов нейронных сетей, в частности сетей прямого распространения, однако реализуется временное окно получения данных.

Для генерации с помощью программного кода нужно использовать следующую команду:

```
narxnet(inputDelays,feedbackDelays,hiddenSizes,trainFcn)
```

Аргументы:

inputDelays – входной вектор задержки (по умолчанию = 1:2);

feedbackDelays – вектор обратных задержек (по умолчанию = 1:2);
hiddenSizes – количество скрытых нейронов (по умолчанию = 10);
trainFcn – метод обучения (по умолчанию = ‘trainlm’).

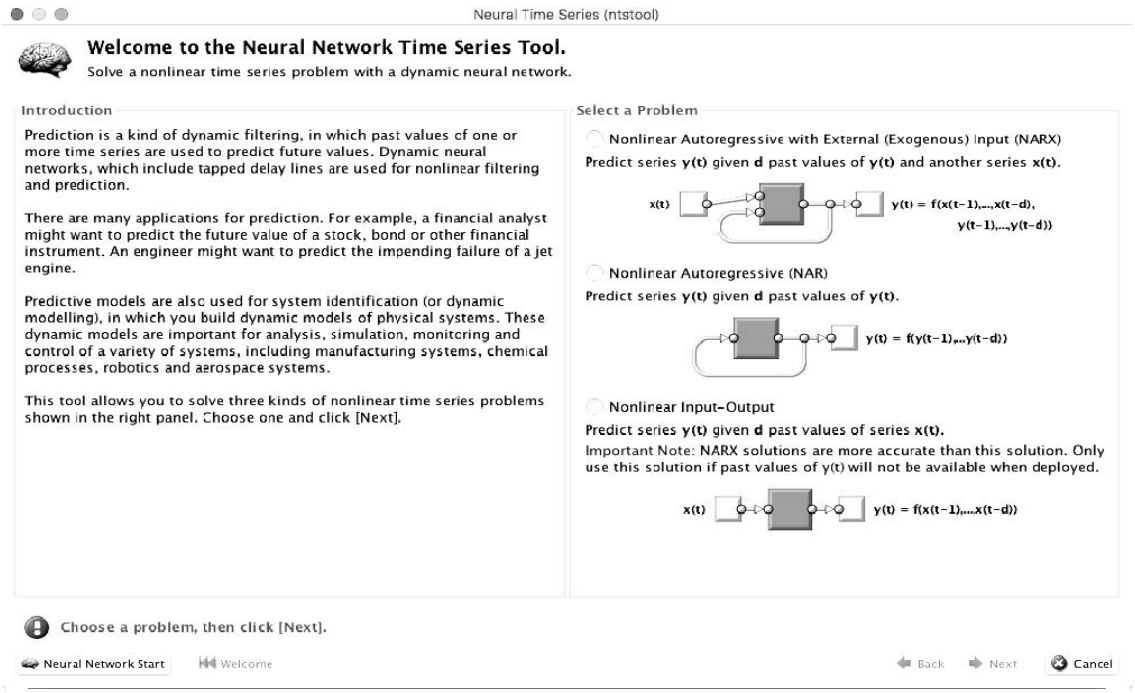


Рисунок 5.1 – Экран приветствия утилиты ntstool

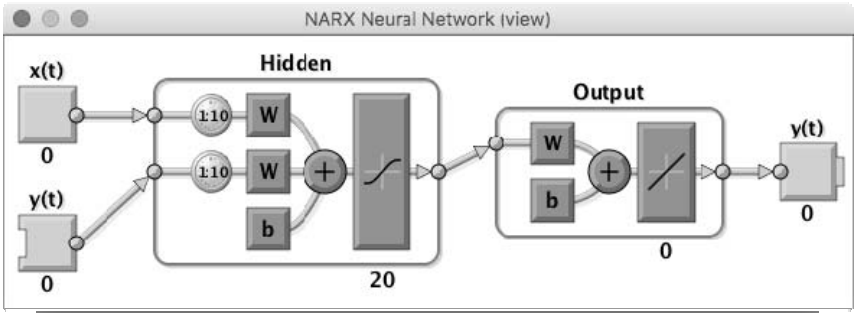


Рисунок 5.2 – NARX NN

С помощью команды **closeloop(net)** есть возможность замкнуть контур этой сети для получения рекуррентной сети.

Еще одной разновидностью нейронной сети, реализуемой на MatLab, является нелинейная вход-выход (без обратных связей) (рисунок 5.3).

Для генерации с помощью программного кода нужно использовать следующую команду:

narnet(feedbackDelays,hiddenSizes,trainFcn)

Аргументы:
inputDelays – входной вектор задержки (по умолчанию = 1:2);

hiddenSizes – количество скрытых нейронов (по умолчанию = 10);

trainFcn – метод обучения (по умолчанию = ‘trainlm’).

С помощью команды closeloop(net) есть возможность замкнуть контур этой сети для получения рекуррентной сети – генератора (рисунок 5.4).

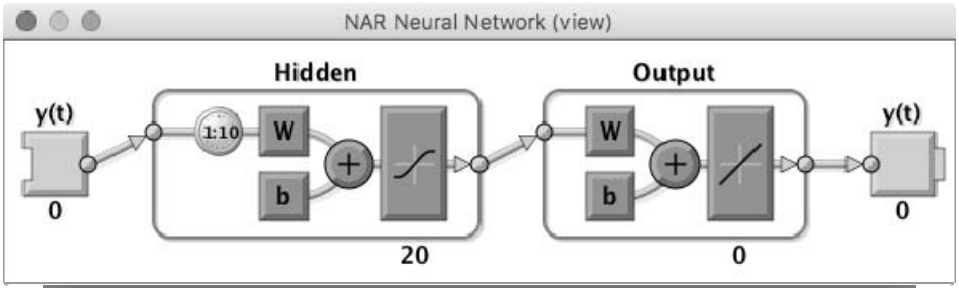


Рисунок 5.3 – NIO NN

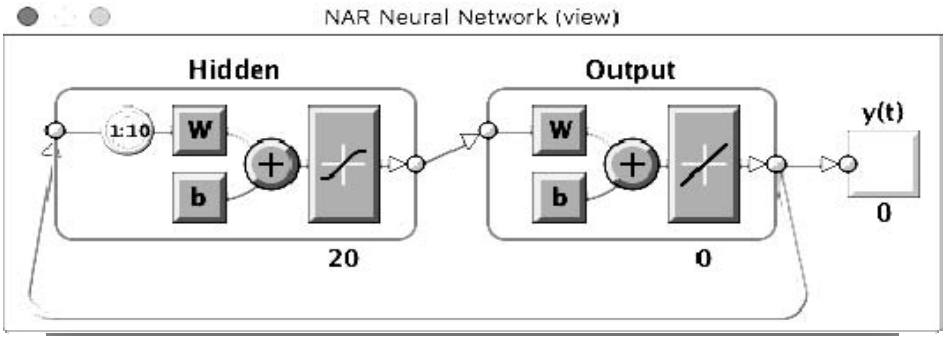


Рисунок 5.4 – NAR NN

Пример созданий нейронной сети с помощью кода:

```
inputDelays = 1:10;
feedbackDelays = 1:10;
hiddenLayerSize = 20;
 %Нейронные сети без обратных связей
 netx =  narxnet(inputDelays,feedbackDelays,hiddenLayerSize);
net =  narnet(feedbackDelays,hiddenLayerSize);
 %Нейронные сети с обратными связями
 netxc = closeloop(netx);
netc = closeloop(net);
 %Для просмотра полученных сетей:
 view(net)
```

Практическое задание

Реализовать разработанную нейронную сеть для распознавания дефектов на изображении с помощью прикладного пакета MatLab.

Контрольные вопросы

1 Какая утилита используется для реализации нейронных сетей в прикладном пакете MatLab?

2 Можно ли реализовать нейронную сеть, используя программный код в MatLab?

6 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Обучение нейронной сети

Цель работы: обучить нейронную сеть в программном пакете MatLab.

6.1 Общие сведения

В утилите ntstool выбор данных для обучения не составляет проблем, поэтому рассмотрим выбор параметров в коде (рисунок 6.1).

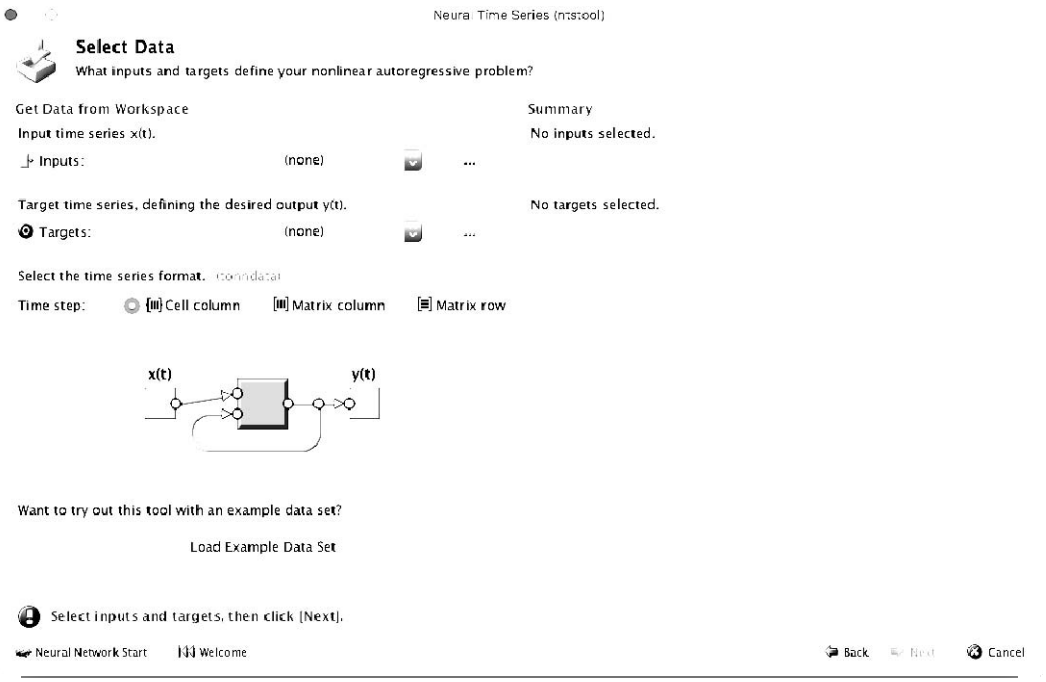


Рисунок 6.1 – Окно выбора параметров

Подготовка данных для обучения и имитации. Функция PREPARETS подготавливает временные ряды для части сети, изменяя время до минимума, чтобы «чувствовать» входные данные и внутренние переменные. Использование PREPARETS позволяет оставить исходные временные ряды неизменными, легко настраивая их для сети с разными размерами задержек как для открытых, так и закрытых нейронных сетей.

Синтаксис команды:

[Xs,Xi,Ai,Ts,EWs,shift] = preparets(net,Xnf,Tnf,Tf,EW)

Аргументы:
net – нейронная сеть;
Xnf – не возвращаемые входные данные;
Tnf – не возвращаемые выходные данные;
Tf – выходные данные;
EW – вес ошибок (по умолчанию = {1}).
Возвращаемые значения:
Xs – перемещенные входы;
Xi – изначальные состояния входных задержек;
Ai – изначальные состояния скрытых нейронов;
Ts – перемещенные выходы;
EWs – перемещенные веса ошибок.

Также перед применением preparets может потребоваться использование функции tonndata, которая конвертирует данные в стандартный тип для нейронных сетей.

Синтаксис команды tonndata:

[y,wasMatrix] = tonndata(x,columnSamples,cellTime)

Аргументы:
x – матрица или ячейка – массив матриц;
columnSamples – True, если исходные данные представлены в виде колонок, false, если в виде строк;
cellTime – True, если исходные данные представлены в виде колонок ячеек матрицы, false, если в виде матрицы.
Возвращаемые значения:
y – исходные данные, конвертированные в стандартный вид;
wasMatrix – True, если исходные данные представлены в виде матриц.

Выбор назначения данных.

В этой части рассматривается выбор процентного соотношения данных: Тренировка, Проверка, Тестирование (рисунок 6.2).

Как и предыдущие шаги, работа с ntstool очень проста и требует простого выбора значений. Если вы проектируете нейронную сеть с помощью программного кода, используйте следующие команды:

net.divideParam.trainRatio = 85/100; %Тренировка;
net.divideParam.valRatio = 15/100; %Проверка;
net.divideParam.testRatio = 5/100; %Тестирование.

Обучение нейросети.

Операция обучения кодом выглядит так (рисунок 6.3):
[net,tr] = train(net,inputs,targets,inputStates,layerStates).

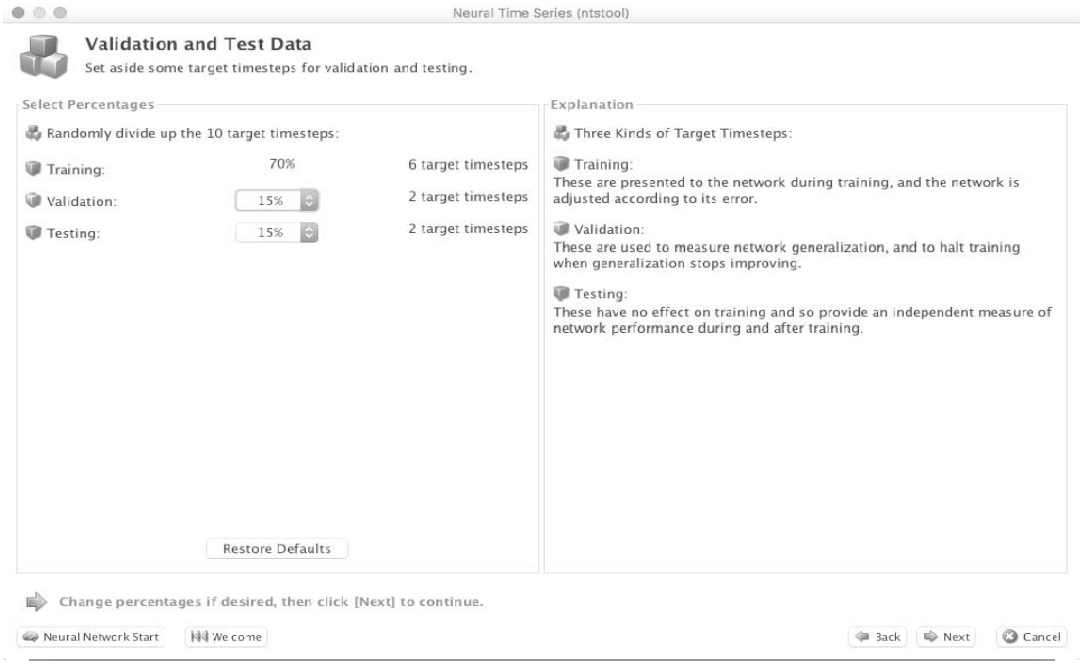


Рисунок 6.2 – Окно выбора процентного соотношения

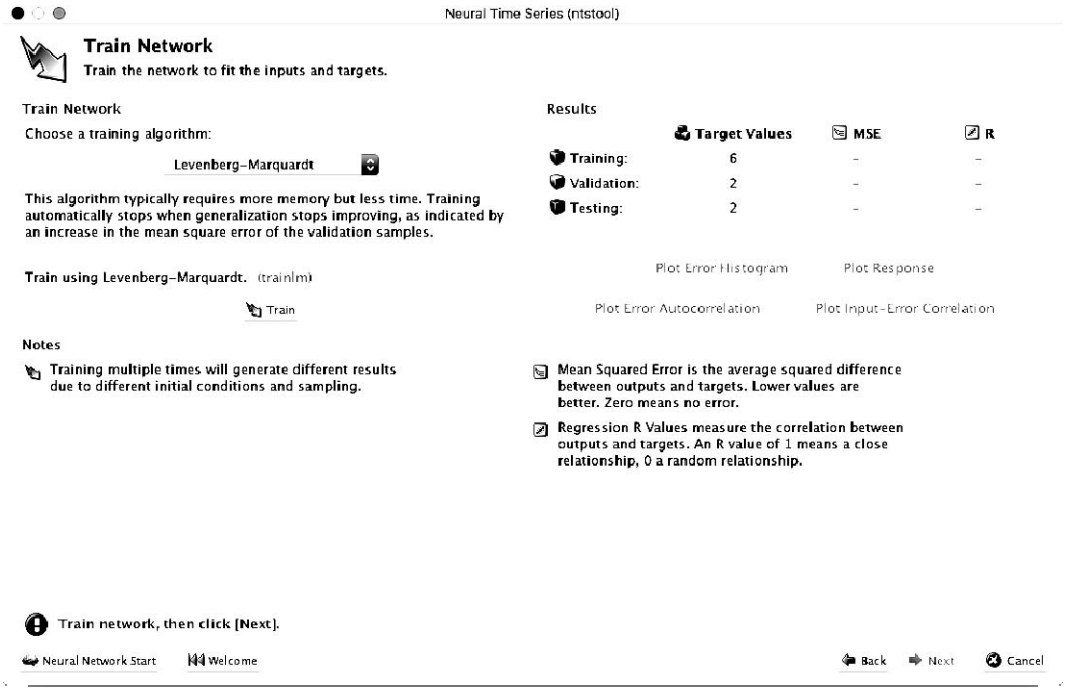


Рисунок 6.3 – Окно обучения нейронной сети

Для обучения сети в обоих случаях используется nntraintool (рисунок 6.4), при помощи которого можно получить различные графики, характеризующие процесс и качество обучения сети.

Графики также можно вызывать с помощью команд:

```
outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)
```

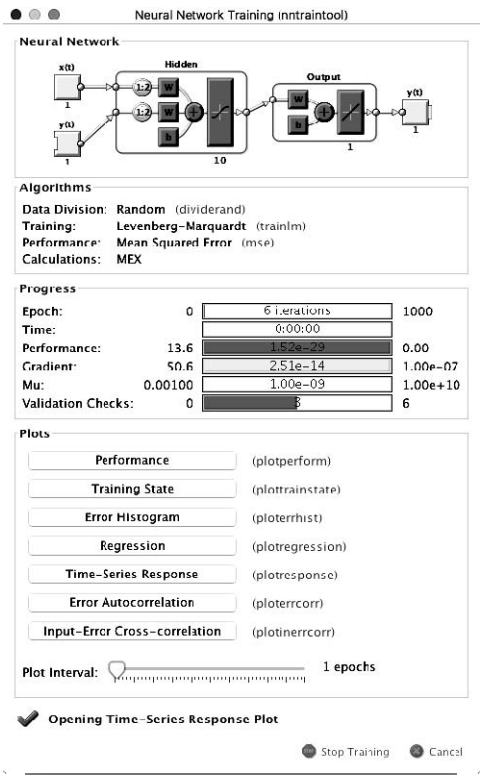


Рисунок 6.4 – Окно nntraintool

Пример программного кода простейшей нейронной сети:

```
%Исходные данные
 inputSeries = con2seq(In');
targetSeries = con2seq( F');
%Создание нейросети
 inputDelays = 1:10;
feedbackDelays = 1:10;
hiddenLayerSize = 20;
net =  narxnet(inputDelays,feedbackDelays,hiddenLayerSize);
%Задание процентного соотношения
 net.divideParam.trainRatio = 85/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 5/100;
 %Обучение нейросети
 [net,tr] = train(net,inputs,targets,inputStates,layerStates);
%Графики результатов
 outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)
%Экспорт полученной нейросети
 gensim(net,0.001)
```

Практическое задание

Обучить разработанную нейронную сеть для распознавания дефектов на изображениях, используя ранее собранные данные в виде обучающей и тестовой выборок, используя прикладной пакет MatLab.

Контрольные вопросы

1 Какие команды используются для обучений нейронной сети в прикладном пакете MatLab?

2 Какие существуют особенности в формировании данных для обучения нейросети в прикладном пакете MatLab?

7 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Оценка нейронной сети, отладка и тестирование

Цель работы: научиться оценивать, отлаживать и тестировать разработанную нейронную сеть.

7.1 Общие сведения

Алгоритм отладки нейронной сети состоит из пяти этапов:

1) простой старт;
2) подтверждение потерь;
3) проверка промежуточных результатов и соединений;
4) диагностика параметров;
5) контроль работы.

Простой старт. Простой старт заключается в создании упрощенной модели и обучении ее на одном наборе данных. Для быстрого старта необходимо создать небольшую сеть с единственным скрытым слоем и проверить корректность ее работы. Затем модель постепенно усложняется. В качестве быстрой проверки можно использовать для обучения одну или две точки данных, чтобы подтвердить корректность работы системы. Нейронная сеть должна показывать 100-процентную точность обучения и проверки. Если это не так, то либо модель слишком мала, либо в нейронной сети закралась ошибка.

Подтверждение потерь. Оценка потерь – это основной способ уточнить производительность модели. Для этого необходимо убедиться, что потеря соответствует задаче, а функции потерь оценивается по корректной шкале. В случае использования больше одного типа потерь необходимо убедиться, что все они одного порядка и правильно масштабированы.

Проверка промежуточных результатов и соединений. Для отладки нейронной сети необходимо понимать динамику процессов происходящих внутри и роль отдельных промежуточных слоев, поскольку они связаны. В процессе отладки можно столкнуться со следующими ошибками:

– неправильные выражения для обновлений;
– не применяются обновления веса;
– исчезающие или взрывающиеся градиенты.

Если значения градиента нулевые, это значит, что скорость обучения в

оптимизаторе слишком мала, или же наблюдается некорректное выражение для обновления градиента. При этом необходимо отслеживать значения функций активаций, весов и обновлений каждого из слоев. Величина обновлений параметров должна составлять 0,001.

Известны три главных метода визуализации нейронной сети:

1) предварительные – простые методы, которые показывают нам общую структуру обученной модели. Они включают вывод форм или фильтров отдельных слоев нейронной сети и параметров в каждом слое;

2) основанные на активации. В них происходит расшифровка активации отдельных нейронов или группы нейронов, чтобы понять их функции;

3) основанные на градиентах. Эти методы имеют тенденцию манипулировать градиентами, которые формируются из прохода вперед и назад при обучении модели.

Для визуализации активаций и соединений отдельных слоев можно использовать ConX и Tensorboard.

Диагностика параметров. У нейронной сети масса параметров, которые взаимодействуют друг с другом, что усложняет оптимизацию.

Размер пакета (*batch size*). Желательно, чтобы размер пакета был достаточно большим для получения точных оценок градиента ошибки, но достаточно малым, чтобы стохастический градиентный спуск мог упорядочить нейронную сеть. Небольшие размеры пакетов приведут к быстрому схождению за счет шума в процессе обучения и в дальнейшем к трудностям оптимизации.

Скорость обучения. Слишком низкая приведет к медленной конвергенции или риску застрять в локальных минимумах. В то же время высокая скорость обучения вызовет расхождение оптимизации, поэтому необходимо использовать планирование скорости, чтобы снизить ее в процессе обучения нейронной сети.

Gradient clipping. Обрезка градиентов параметров во время обратного распространения по максимальному значению или предельной норме. Полезно для решения проблем с любыми взрывающимися градиентами, с которыми вы можете столкнуться.

Пакетная нормализация. Используется для нормализации входных данных каждого слоя, что позволяет решить проблему внутреннего ковариатного сдвига.

Стохастический градиентный спуск. Существует несколько разновидностей, которые используют импульс, адаптивные скорости обучения и метод Нестерова. При этом ни у одной из них нет явного преимущества как по эффективности обучения, так и по обобщению.

Регуляризация. Имеет решающее значение для построения обобщаемой модели, поскольку добавляет штраф за сложность модели или экстремальные значения параметров. Это способ снизить дисперсию модели без существенного увеличения ее смещения.

Выпадение. Еще один метод упорядочения сети для предотвращения перегрузки. Во время обучения выпадение осуществляется только поддержанием активности нейрона с некоторой вероятностью p (гиперпараметр) или уста-

новкой его на ноль в обратном случае. В результате сеть должна использовать другое подмножество параметров для каждой обучающей партии, что уменьшает изменения определенных параметров, которые становятся доминирующими.

Контроль работы. Речь идет о документировании рабочих процессов и экспериментов. Если ничего не документировать, можно забыть, например, какая используется скорость обучения или вес классов. Благодаря контролю можно без проблем просматривать и воспроизводить предыдущие эксперименты. Это позволяет снизить количество дублирующихся экспериментов.

Однако ручное документирование может стать сложной задачей в случае большого объема работ. Здесь приходят на помощь такие инструменты, как Comet.ml, помогающие автоматически логировать наборы данных, изменения кода, историю экспериментов и производственные модели, включая ключевые сведения о вашей модели (гиперпараметры, показатели производительности модели и сведения об окружении).

Нейронная сеть может быть весьма чувствительной к небольшим изменениям, а это приведет к падению производительности модели. Отслеживание и документирование работы – первый шаг, который стоит предпринять для стандартизации среды и моделирования.

Практическое задание

Осуществить оценку, отладку и тестирование нейронной сети на основе тестовой выборки собранной ранее.

Контрольные вопросы

1 Какие существуют этапы отладки нейронной сети?
2 Какие параметры нейронной сети диагностируются?

8 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Доучивание нейронной сети

Цель работы: научиться понимать, когда требуется доучивание нейронной сети и каким образом оно осуществляется.

8.1 Краткие теоретические сведения

«Проклятие размерности» относится к различным явлениям, возникающим при анализе и организации данных в многомерных пространствах (часто с сотнями или тысячами измерений), и не встречается в ситуациях с низкой размерностью. Грамматика английского языка имеет огромное количество атрибутов, влияющих на нее. В машинном обучении мы должны представить их признаками в виде массива/матрицы конечной и существенно меньшей

длины (чем количество существующих признаков). Для этого сети обобщают эти признаки, что порождает две проблемы.

1 Из-за неправильных предположений появляется смещение. Высокое смещение может привести к тому, что алгоритм пропустит существенную взаимосвязь между признаками и целевыми переменными. Это явление называют недообучение.

2 От небольших отклонений в обучающем множестве из-за недостаточного изучения признаков увеличивается дисперсия. Высокая дисперсия ведет к переобучению, ошибки воспринимаются в качестве надежной информации.

На ранней стадии обучения смещение велико, потому что выход из сети далек от желаемого, а дисперсия очень мала, поскольку данные имеют пока малое влияние. В конце обучения смещение невелико, потому что сеть выявила основную функцию в данных. Однако, если обучение слишком продолжительное, сеть также изучит шум, характерный для этого набора данных, что приводит к большому разбросу результатов при тестировании на разных множествах, поскольку шум меняется от одного набора данных к другому (рисунок 8.1).

Алгоритмы с большим смещением в основе более простых моделей, которые не склонны к переобучению, но могут недообучиться и не выявить важные закономерности или свойства признаков. Модели с маленьким смещением и большой дисперсией обычно более сложны с точки зрения их структуры, что позволяет им более точно представлять обучающий набор. Однако они могут отображать много шума из обучающего набора, что делает их прогнозы менее точными, несмотря на их дополнительную сложность.

Следовательно, невозможно иметь маленькое смещение и маленькую дисперсию одновременно.

Сейчас есть множество инструментов, с помощью которых можно легко создать сложные модели машинного обучения. Переобучение занимает центральное место, поскольку смещение появляется, когда сеть не получает достаточно информации. Но чем больше примеров, тем больше появляется вариантов зависимостей и изменчивостей в этих корреляциях.

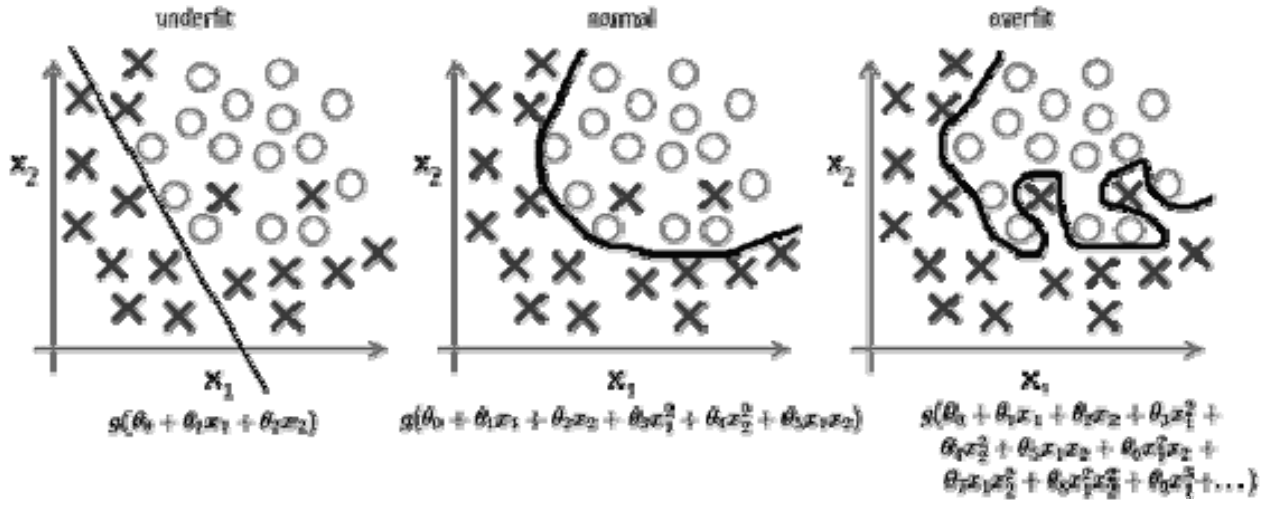


Рисунок 8.1 – Выделение множеств в случае недообучения, обучения и переобучения

Практическое задание
Провести дообучение разработанной нейронной сети.

Контрольные вопросы

1 В каком случае требуется дообучение нейронной сети?
2 Что такое недообучение?
3 Что такое переобучение?

9 Разработка нейронной сети для распознавания изображений при неразрушающем контроле. Применение нейронной сети для распознавания дефектов

Цель работы: научить нейронную сеть распознавать дефекты на изображении.

9.1 Общие сведения

Полносвёрточные сети – это особый тип искусственных нейронных сетей, результатом работы которых является сегментированное изображение оригинала, где искомые элементы уже выделены требуемым образом.

Полносверточные нейронные сети используются для задач, в которых необходимо, например, определить форму и местоположение искомого объекта или нескольких объектов. Подобные задачи проблематично решать с использованием простых свёрточных нейросетей. Для общего понимания почему и когда лучше использовать полносвёрточные сети вместо обычных свёрточных, необходимо сравнить указанные типы нейронных сетей.

Самым очевидным отличием полносвёрточных сетей от других нейросетей является конечный результат работы сети. Простые сверточные сети могут использоваться для классификации, определяя к какому классу принадлежит то или иное изображение, и для локализации объекта на изображении (регрессия).

В результате применения любого из этих методов выходными данными являются числа либо массивы чисел. Иначе говоря, мы можем получить информацию о изображении (при этом очень ограниченную), но не можем преобразовать его в необходимый нам вид.

Полносвёрточные нейронные сети на выходе выдают сегментированное изображение, соответствующее по размерности входному. Поэтому их второе название – сегментационные нейронные сети.

Сегментация – это объединение объектов в группы по общим признакам. Таким образом, мы получаем от сети намного больше информации, а достигнутый результат можем обработать простыми эвристическими методами.

Чтобы лучше понимать принцип работы полносвёрточных сетей и выяснить, для решения каких задач их можно использовать, важно изучить их

общую архитектуру. При проектировании свёрточных сетей в архитектуру могут добавляться разнообразные слои для повышения точности распознавания (drop out layer, local response normalization layer и т. д.). Мы же рассмотрим лишь базовую архитектуру, которая практически неизменна и определяет работу полносвёрточной сети.

Основа работы полносвёрточных сетей – свёртка изображения. Ключевыми слоями являются свёрточные слои (convolution layer). В свёрточном слое указывается количество выходов со слоя, ядро свёртки, его шаг, размеры, отступ.

Операция свёртки проходит ядром по всему изображению, в результате мы получаем величину отклика на ядро свёртки в каждой точке изображения. Количество ядер каждого слоя свёртки равно произведению количества выходов со слоя на количество входных картинок. Далее результаты проходят через следующий слой свёртки, получая значения уже для других ядер. К каждому свёрточному слою можно добавлять слои регуляризации или нормализации (в зависимости от выбора разработчика). Прохождение изображений через многие свёрточные слои позволяет получить богатое разнообразие возможных интерпретаций изображения.

Пройдя через требуемое количество слоёв свёртки, изображение затем попадает в слой пулинга (pooling layer). Этот слой уменьшает размер входных изображений, не уменьшая их количества. Слой имеет ядро, которое движется подобно ядру свертки, вычисляя единственное значение для каждой области изображения. Уменьшение изображения способствует более быстрой обработке сетью большего количества данных. Это позволяет добавлять в следующих свёрточных слоях большее количество выходов, а также повышает точность результатов. Дело в том, что на уменьшенном изображении ядра свёртки того же размера способны захватывать большую область искомого объекта.

Последовательность convolution/../../convolution/pooling (где количество convolutions layers определяется разработчиком) может повторяться несколько раз, вплоть до того, пока не будет достигнут минимальный размер изображения. Этот размер определяется экспериментально.

Для того чтобы выделенные объекты соответствовали оригинальному масштабу, уменьшенное изображение требуется вернуть к изначальному размеру. Слой upsample (upsample layer) выполняет увеличение изображения. На каждый выход имеется два входных изображения: первое – это обработанная картинка с предыдущего слоя (может быть convolution или pooling), второе – это картинка из pooling layer, количество выходов которого равно количеству входов соответствующего upsample, а также размеры выходной картинки pooling равны размеру входной картинки upsample. Мы получаем симметричную архитектуру относительно последнего pooling-слоя и первого upsample-слоя. Между слоями увеличения размеров изображения также помещаются свёрточные слои, но количество выходов с них постепенно уменьшается.

Последовательность upsample/convolution/../../convolution необходима, чтобы привести изображение к исходным размерам, при этом сократив коли-

чество возможных интерпретаций изображения до количества искомых групп объектов.

Практическое задание

Осуществить распознание изображений с дефектами с использованием разработанной нейронной сети.

Контрольные вопросы

1 Какие могут быть особенности в распознании дефектов с использованием нейронной сети?

2 Какой процент ошибок выдала разработанная нейросеть?

Список литературы

1 **Бринк, Х.** Машинное обучение / Х. Бринк, Д. Ричардс, М. Феверолф. – СПб. : Питер, 2017. – 336 с.

2 Моделирование и распознавание 2D/3D образов. – URL: https://api-2d3d-cad.com/ident_by_color_texture (дата обращения: 11.10.2025).

3 Разбиение исходных данных на обучающую и контрольную выборки. – URL: https://r-analytics.blogspot.com/2015/08/blog-post_31.html (дата обращения: 11.10.2025).

4 Формирование примеров обучающей и тестовой выборки нейронной сети на принципах планирования экспериментов. – URL: https://euroasia-science.ru/physiko-matematicheskie-nauki (дата обращения: 11.10.2025).

5 Шпаргалка по разновидностям нейронный сетей. Ч. 1 : Элементарные конфигурации. – URL: https://tproger.ru/translations/neural-network-zoo-1 (дата обращения: 11.10.2025).

6 Введение в архитектуры нейронных сетей. – URL: https://habr.com/ru/company/oleg-bunin/blog/340184 (дата обращения: 01.10.2025).

7 Применение нейронных сетей для задач классификации. – URL: https://basegroup.ru/community/articles/classification (дата обращения: 01.10.2025).

8 Нейронные сети в MatLab. – URL: https://digiratory.ru/508 (дата обращения: 11.10.2025).

9 Работаем с нейросетями: чек-лист для отладки. – URL: https://habr.com/ru/company/skillbox/blog/444684 (дата обращения: 11.10.2025).

10 Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери. – URL: https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri (дата обращения: 11.10.2025).

11 Поиск объекта на изображении с помощью полносверточных нейронных сетей. – URL: https://www.azoft.ru/blog/poisk-obekta-na-izobrazhenii-s-pomoshchyu-polnosvyortochnyh-nejronnyh-sete (дата обращения: 11.10.2025).