Методические указания по выполнению лабораторной работы №2
«Обнаружение аномалий»

Введение

Задача обнаружения аномалий актуальна для множества прикладных областей:

- Информационная безопасность (обнаружение сетевых атак, аномалий поведения пользователя и пр.);
- Промышленность (обнаружение аномальных состояний оборудования);
- Банковская среда (обнаружение мошенничества и аномальных транзакций);
- Медицина и здравоохранение (раннее обнаружение патологий);
- Экология (обнаружение аномалий в пространстве экологических факторов) и т.д.

Одним из современных инструментов обнаружения аномалий является автокодировщик. Автокодировщик – это нейронная сеть специальной архитектуры. Количество нейронов во входном и выходном слое соответствует размерности входных данных (количеству признаков). При этом скрытые слои содержат меньшее количество нейронов. Во время обучения автокодировщик обучается сначала кодировать входные данные в меньшую размерность, а затем декодировать их с минимальной ошибкой. Обучение автокодировщика продолжается до тех пор, пока он не настроится восстанавливать примеры обучающей выборки с приемлемой точностью. Обычно точность определяется величиной среднеквадратичной ошибки (Mean Squared Error, MSE). Восстановленные на выходе автокодировщика данные называют реконструкцией.

После обучения автокодировщик можно использовать для оценки близости поданных на вход примеров к обучающим данным. Поэтому автокодировщик часто используется в качестве одноклассового классификатора. Степень близости входного примера к обучающим данным определяется величиной Immediate Reconstruction Error (IRE). Значение IRE для входного примера $X(x_1, x_2, \dots, x, \dots x_m)$ размерности m рассчитывается по формуле: $IRE_X = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$, где $Y(y_1, y_2, \dots, y, \dots y_m)$ – реконструкция входного примера автокодировщиком. Чем ближе IRE_X к нулю, тем точнее автокодировщик восстановил входной вектор и тем достовернее гипотеза о том, что входной вектор принадлежит к обучающему множеству. Для обнаружения аномалий в данных устанавливается пороговое значение IRE_{th}. Пороговое значение в данной работе выбирается как максимальное значение среди ошибок реконструкции, рассчитанных для примеров обучающего набора.

Пороговое значение ошибки реконструкции можно интерпретировать как границу класса в пространстве признаков, которую «очертил» обученный автокодировщик. Таким образом, если IRE для примера X превышает порог распознавания IRE_{th}, значит, пример находится за границей класса и классифицируется как аномалия.

Существенная проблема применения нейронных сетей заключается в том, что они представляют собой «черный ящик». В том смысле, что результат их работы на некоторых входных данных может быть необъясним и непредсказуем, несмотря на

высокие показатели традиционных характеристик качества, рассчитываемых по результатам классификации ограниченного тестового набора.

Эта проблема может быть решена с помощью нового критерия качества обучения EDCA, позволяющего проверять область, создаваемую моделью после обучения, на соответствие обучающим данным.

Предлагается критерий на основе четырех показателей: Excess, Deficit, Coating, Approx (EDCA), который позволяет оценить точность аппроксимации моделью области обучающих данных. Критерий оценивает величину и положение областей, формируемых классификатором после обучения, относительно целевых. Применение критерия позволит оценивать модели машинного обучения и настраивать их до тех пор, пока риск неправильной классификации за пределами обучающей выборки не будет минимизирован.

Значения характеристик рассчитываются на основе двух дискретных оценок: оценки объема, занимаемого обучающими данными в пространстве признаков, и оценки объема, который занимают данные, распознаваемые классификатором. Все данные, которые обученная модель относит к целевому классу, будем называть деформированным набором данных. Чтобы определить деформированный набор данных, пространство объектов сканируется с определенным шагом, а затем данные сканирования классифицируются с использованием обученной модели. Данные, для которых результат классификации положительный, составляют деформированный набор данных.

Для оценки объема данных пространство признаков в исследуемых границах разделяется на атомарные ячейки. Причем границы исследуемого пространства признаков выбираются на основе существенно расширенных пределов значений всех признаков. Объем атомарной ячейки предварительно вычисляется. Например, если пространство признаков двумерное, то оно может быть разбито сеткой с одинаковым шагом h по каждой оси (признаку). Тогда объём атомарной ячейки равен площади ячейки, h^2.

В общем случае, дискретная оценка объема обучающего набора $|X_T^*|$ представляет собой сумму непустых ячеек, умноженную на атомарный объем. Учитываются только те ячейки, которые содержат хотя бы одну точку обучающего набора. Соответственно, дискретная оценка объема деформированного набора $|X_D^*|$ представляет собой сумму объемов атомарных ячеек, содержащих, по меньшей мере, одну точку деформированного набора.

Характеристики EDCA рассчитываются по следующим формулам:

$$Excess = \frac{|X_D^* \backslash X_T^*|}{|X_T^*|} \quad (1)$$

$$Deficit = \frac{|X_T^* \backslash X_D^*|}{|X_T^*|} \quad (2)$$

$$Coating = \frac{|X_T^* \cap X_D^*|}{|X_T^*|} \quad (3)$$

$$Approx = \frac{|X_T^*|}{|X_D^*|} \quad (4)$$

Идеальный одноклассовый классификатор характеризуется следующими значениями показателей:

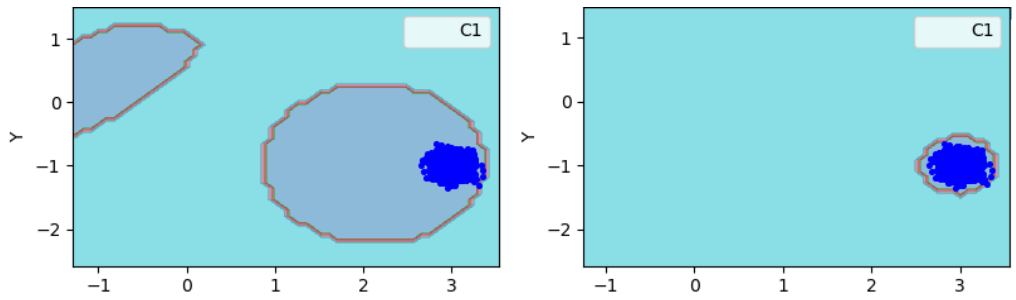
$$Excess = 0, \quad Deficit = 0, \quad Coating = 1, \quad Approx = 1 \quad (5)$$

Методология оценки классификатора выглядит следующим образом:

1) Определение границ исследуемого пространства признаков;
2) Выбор шага сканирования h;
3) Сканирование пространства признаков и получение данных сканирования;
4) Получение ответов классификатора для примеров обучающего набора X_T;
5) Расчет объема $|X_T^*|$ для каждого класса;
6) Получение ответов классификатора для данных сканирования;
7) Расчет объема $|X_D^*|$ для каждого класса;
8) Расчет характеристик EDCA для каждого класса;
9) Определение качества классификатора путем сравнения значений характеристик для каждого класса со значениями, установленными в (5).

Такой подход к оценке качества классификаторов представляется более информативным и точным. А главное, позволяет строить надежные и предсказуемые классификаторы.

Эффективность критерия на основе характеристик EDCA может быть продемонстрирована на примере одноклассового классификатора и синтетического набора двумерных данных. В качестве одноклассового классификатора используется автокодировщик. Рассмотрим два автокодировщика AE1 и AE2. Первый автокодировщик AE1 имеет архитектуру [2; 1; 2], а второй автокодировщик AE2 имеет архитектуру [2; 3; 5; 2; 1; 2; 5; 3; 2]. Автокодировщики AE1 и AE2, обученные на одном синтетическом наборе данных в течение 1000 эпох, обеспечивают разную точность аппроксимации обучающего набора. На рис. 1 продемонстрированы области точек в пространстве признаков, которые обученные автокодировщики AE1 (рис. 1а) и AE2 (рис. 1б) будут относить к целевому классу.



а б

Рис.1 Деформированные области, построенные автокодировщиками AE1(а) и AE2(б)

Рис. 1а показывает, что первый автокодировщик AE1 относит к целевому классу несколько обширных областей пространства признаков, в то время как обучающие данные занимают только часть одной из них. Такой автокодировщик будет пропускать аномалии, а также уязвим для состязательных атак, поскольку возможно подобрать состязательный пример, который сможет обмануть классификатор.

Архитектура автокодировщика AE2 была сложнее, поэтому он аппроксимировал область обучающих данных точнее. Как показано на рис. 1б, область, относимая автокодировщиком AE2 к целевому классу, полностью соответствует области обучающих данных. Это означает, что модель AE2 больше подходит для точного обнаружения аномалий и не уязвима для состязательных атак.

Чтобы продемонстрировать интерпретируемость и эффективность критерия, в таблице 1 приведены значения предложенных показателей EDCA для первого и второго автокодировщика.

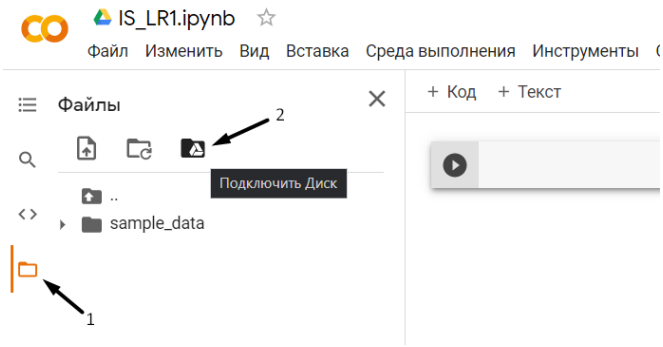
Таблица 1. EDCA для AE1 и AE2

	Excess	**Deficit**	**Coating**	**Approx**
AE1	11.25	0	1	0.08
AE2	0.5	0	1	0.67
Ideal AE	0	0	1	1

Значения показателей для автокодировщика AE2 ближе к идеальным. Это означает, что он менее уязвим для состязательных атак и пропусков аномалий. Предлагаемый критерий особенно ценен в случае высокой размерности данных, когда визуализация областей, как на рис. 1, невозможна. Однако это остаётся за рамками текущей лабораторной работы.

Подготовка среды выполнения

В среде Google Colab необходимо создать новый блокнот *IS_LR2* и подключить Диск, как показано на рисунке:



Далее согласно заданию требуется создать на Диске папку с именем *is_lab2*. Создание папки может быть сделано интерактивно в Colab, либо непосредственно на Google Диске, либо с помощью команды `mkdir`.

Далее в примерах будет предполагаться, что папка *is_lab2* располагается по пути: ***drive/MyDrive/Colab Notebooks/is_lab2***.

Для удобства дальнейшей работы предлагается выбрать созданную папку рабочей директорией:

```
import os
os.chdir('/content/drive/MyDrive/Colab Notebooks/is_lab2')
```

Для скачивания библиотеки lab02_lib.py и файлов с выборками *name_train.txt*, *name_test.txt*, а также создания папки *out* предлагается воспользоваться следующими командами:

```
# скачивание библиотеки
!wget -N
http://uit.mpei.ru/git/main/is_dnn/raw/branch/main/labworks/LW2/lab02_lib.py

# скачивание выборок
!wget -N
http://uit.mpei.ru/git/main/is_dnn/raw/branch/main/labworks/LW2/data/name_train.txt
!wget -N
http://uit.mpei.ru/git/main/is_dnn/raw/branch/main/labworks/LW2/data/name_test.txt
```

Итого в папке *is_lab2* должны находиться следующие файлы и папки:

- *lab02_lib.py*,
- папка *out*,
- *name_train.txt*,
- *name_test.txt*, где *name* – вариант данных, назначенный бригаде исходя из ее номера.

Импорт библиотек и модулей

Для выполнения данной лабораторной работы понадобится импорт следующих библиотек:

```
# импорт модулей
import numpy as np
import lab02_lib as lib
```

Задание 1

Загрузка и изучение набора данных

В первой части лабораторной работы исследуется синтетический двумерный набор данных в целях знакомства и понимания принципов работы автокодировщика. Для генерации двумерного набора данных, состоящего из 1000 элементов с центром, например, в точке (5, 5) можно использовать функцию `datagen` из библиотеки `lib`:

```
# генерация датасета
data = lib.datagen(5, 5, 1000, 2)
```

Код функции можно изучить, открыв для просмотра *lab02_lib.py*. Функция `datagen` принимает на вход координаты центра кластера в пространстве признаков, число элементов и количество осей в пространстве (признаков). Упомянутая выше функция автоматически генерирует набор данных с заданными параметрами, выводит данные на рисунок и сохраняет набор в файл *'data.txt'*.

Вывести массив и его размерность в консоль можно с помощью свойства shape:

```
# вывод данных и размерности
print('Исходные данные:')
print(data)
print('Размерность данных:')
print(data.shape)
```

Создание и обучение автокодировщика

Для создания, обучения и сохранения автокодировщика предлагается использовать функцию `lib.create_fit_save_ae`. Функция принимает на вход следующие данные в качестве параметров:

- данные для обучения,
- путь и имя файла для сохранения автокодировщика,
- путь и имя файла для сохранения порога обнаружения аномалий,
- количество эпох обучения,
- **`True`** или **`False`** для визуализации или сокрытия процесса обучения,
- а также параметр `patience`, задающий число эпох, в течение которых необходимо продолжать обучение, если ошибка обучения перестала уменьшаться. По истечение заданного числа эпох происходит ранняя остановка обучения. Данный параметр позволяет избежать с одной стороны переобучения (когда веса нейросети настраиваются для распознавания исключительно примеров обучающего набора), а с другой – попадания в локальный минимум ошибки обучения. В случае, если ошибка обучения перестала уменьшаться и обучение прерывается слишком рано, нейронная сеть может оказаться не до конца обученной, если слишком поздно – переобученной.

Для создания и обучения автокодировщика AE1 в течение 1000 эпох с визуализацией обучения и параметром `patience` равным 300 эпох, необходимо выполнить следующее:

```
# обучение AE1
patience = 300
ae1_trained, IRE1, IREth1 = lib.create_fit_save_ae(data,'out/AE1.h5','out/AE1_ire_th.txt',
1000, True, patience)
```

Функция предполагает интерактивный выбор архитектуры автокодировщика. Сначала можно задать архитектуру автокодировщика вручную. Для этого необходимо выбрать количество скрытых слоёв, например, 1. А затем указать количество нейронов в этом слое, например, 1. Размер входного и выходного слоя выставляется автоматически в зависимости от размерности входных данных.

В переменную `ae1_trained` передаётся обученный автокодировщик. В переменную `IRE1` передается массив значений ошибки реконструкции, рассчитанных для обучающей выборки `data`. А в переменную `IREth1` – порог ошибки реконструкции (максимум среди значений массива `IRE1`). Данный порог дальше будет использоваться для обнаружения аномалий автокодировщиком AE1.

При визуализации процесса обучения в консоли будет выводится текущая эпоха и значение ошибки обучения (в данном случае MSE). Ошибка обучения представляет собой разницу между целевым выходом, ожидаемым от нейросети, и фактическим. Ошибка обучения, на которой обучение завершилось, обозначим MSE_stop. Если ошибка MSE_stop велика, это означает, что нейронная сеть с низкой точностью воспроизводит на выходе входной пример (плохое качество обучения). Если ошибка MSE_stop для данных изменяющихся, например, в пределах (0, 2000) составляет 0.0001, то с большой вероятностью такая нейросеть переобучена. Переобученная нейронная сеть не может быть использована для распознавания новых примеров, поскольку любые данные, которые отсутствовали в обучающей выборке, будут распознаны как аномальные. Ценность нейронных сетей заключается в их способности к обобщению – после обучения на ограниченном наборе данных нейронная сеть формирует представление о генеральной совокупности и способна верно распознавать данные, на которых не была обучена.

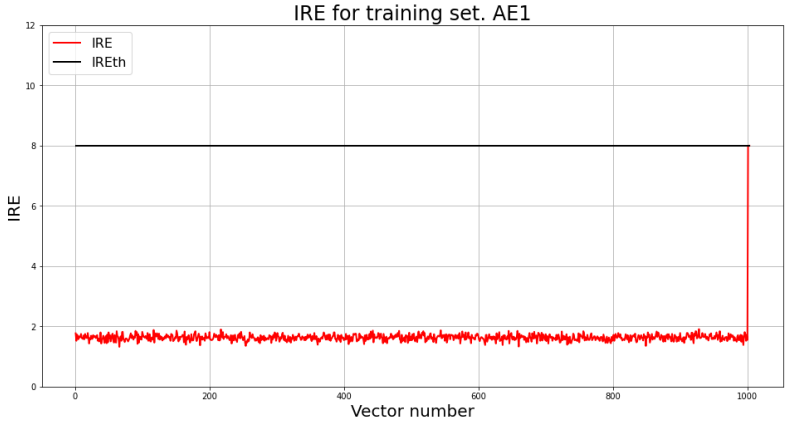
Для оценки качества обучения нейронной сети следует оценить с какой ошибкой автокодировщик восстанавливает (реконструирует) примеры обучающей выборки на выходном слое. Для этого воспользуемся функцией `lib.ire_plot` и построим график ошибки реконструкции обучающей выборки. Функция принимает на вход имя для заголовка рисунка (для обучающей выборки следует указать `'training'`), массив значений ошибки реконструкции и порог ошибки реконструкции, а также имя автокодировщика:

```
# Построение графика ошибки реконструкции
lib.ire_plot('training', IRE1, IREth1, 'AE1')
```

Стоит отметить, что о качестве обучения нейронной сети можно судить по виду графика ошибки реконструкции и величине порога (учитывая рабочий диапазон значений исходных данных). Например, если график ошибки реконструкции обучающей выборки имеет вид (значения ошибки могут начинаться и от 0):

И величина порога не высока, то это означает, что нейронная сеть обучена оптимально и порог обнаружения аномалий адекватно описывает границу области генеральной совокупности исследуемых данных.

Если же график ошибки реконструкции имеет вид:



То это с большой вероятностью означает, что в обучающей выборке был выброс. Поскольку порог завышен в соответствии с пиком ошибки реконструкции, нейронная сеть будет распознавать примеры из области более обширной, чем область обучающих данных. Это может привести к уязвимости классификатора к состязательным атакам (когда классификатор распознает пример, не относящийся к известному классу) и пропуску аномалий. В большинстве прикладных систем, такие ошибки недопустимы.

Избежать этого можно, изменив параметры обучения, например, увеличить количество эпох обучения.

Для использования функций `lib.create_fit_save_ae` и `lib.ire_plot` в целях создания, обучения и сохранения другого автокодировщика, а также построения для

него графика ошибки реконструкции необходимо соответствующим образом изменять имена переменных и параметров. Например:

```
# обучение AE2
ae2_trained, IRE2, IREth2 = lib.create_fit_save_ae(data,'out/AE2.h5','out/AE2_ire_th.txt',
3000, True, patience)
lib.ire_plot('training', IRE2, IREth2, 'AE2')
```

Для второго автокодировщика ***для начала*** можно выбрать архитектуру по умолчанию (тогда количество скрытых слоёв будет 5, а количество нейронов в них: 3 2 1 2 3). А далее подбирать необходимое количество эпох и архитектуру в зависимости от данных для наилучшего качества обучения. Во избежание переобучения или плохого качества обучения в рамках данной работы рекомендуется подбирать архитектуру и количество эпох такими, чтобы MSE_stop для первого автокодировщика была не меньше 1-10, а для второго – не меньше 0,01.

Расчет характеристик качества обучения автокодировщиков

Характеристики качества обучения EDCA для автокодировщика показывают точность аппроксимации области обучающего набора, которой достиг автокодировщик после обучения. Для двумерного пространства признаков расчет характеристик реализуется функцией `lib.square_calc`. На вход функции подаются:

- параметр `numb_square`, отвечающий за количество элементов разбиения по каждой оси пространства признаков,
- обучающая выборка
- обученный автокодировщик
- порог ошибки реконструкции
- номер автокодировщика для корректных подписей рисунков,
- **`True`** или **`False`** для визуализации расчета характеристик.

Например, для расчета характеристик качества для автокодировщика AE1, необходимо выполнить:

```
# построение областей покрытия и границ классов
# расчет характеристик качества обучения
numb_square = 20
xx, yy, Z1 = lib.square_calc(numb_square, data, ae1_trained, IREth1, '1', True)
```

Для интерпретируемости числовых характеристик качества и сравнения областей аппроксимации для разных автокодировщиков предлагается использовать функцию `lib.plot2in1`. Функция принимает на вход обучающую выборку, и выходные параметры функции `lib.square_calc` для двух автокодировщиков. Параметры *xx*, *yy*

(координатная сетка пространства признаков) для двух автокодировщиков одинаковы, поскольку вычислены по одному и тому же обучающему набору. А вот `Z1` и `Z2`, описывающие области аппроксимации AE1 и AE2 в рамках лабораторной работы должны существенно отличаться.

```
# сравнение характеристик качества обучения и областей аппроксимации
lib.plot2in1(data, xx, yy, Z1, Z2)
```

Тестирование автокодировщиков

Для тестирования обученных автокодировщиков необходимо предварительно создать тестовую выборку с именем *data_test.txt* в папке *Lab02*. А затем загрузить тестовые данные в переменную *data_test:*

```
# загрузка тестового набора
data_test = np.loadtxt('data_test.txt', dtype=float)
```

Получить результаты классификации тестового набора с помощью, например, обученного автокодировщика AE1 можно с использованием функции *lib.predict_ae*. Функция принимает на вход обученный автокодировщик, тестовый набор и порог обнаружения аномалий:

```
# тестирование AE1
predicted_labels1, ire1 = lib.predict_ae(ae1_trained, data_test, IREth1)
```

Выходом функции являются: массив ответов автокодировщика (0 - норма, 1 - аномалия) и массив ошибок реконструкции для тестового набора.

Для вывода результатов классификации в виде таблицы и построения графика ошибки реконструкции тестового набора необходимо выполнить:

```
# тестирование AE1
lib.anomaly_detection_ae(predicted_labels1, ire1, IREth1)
lib.ire_plot('test', ire1, IREth1, 'AE1')
```

Чтобы сравнить результаты классификации тестового набора для двух автокодировщиков необходимо выполнить вышеуказанные шаги для второго автокодировщика. Для визуализации областей аппроксимации для двух автокодировщиков и примеров тестового набора выполнить:

```
# построение областей аппроксимации и точек тестового набора
lib.plot2in1_anomaly(data, xx, yy, Z1, Z2, data_test)
```

Задание 2

Загрузка и изучение набора данных

Во второй части лабораторной работы исследуется реальный набор данных по вариантам. Предлагается исследовать три набора данных:

1. Cardio http://odds.cs.stonybrook.edu/cardiotocogrpahy-dataset/

Исходный набор данных Cardiotocography Data Set (https://archive.ics.uci.edu/ml/datasets/Cardiotocography) из репозитория машинного обучения UCI состоит из измерений частоты сердечных сокращений плода и сокращений матки на кардиотокограммах, классифицированных экспертами-акушерами. Исходный набор данных предназначен для классификации. В нем представлено 3 класса: «норма», «подозрение» и «патология». Для обнаружения аномалий класс «норма» принимается за норму, класс «патология» принимается за аномалии, а класс «подозрение» был отброшен.

Количество признаков	Количество примеров	Количество нормальных примеров	Количество аномальных примеров
21	1764	1655	109

2. Letter http://odds.cs.stonybrook.edu/letter-recognition-dataset/

Исходный набор данных Letter Recognition Data Set (https://archive.ics.uci.edu/ml/datasets/Letter+Recognition) из репозитория машинного обучения UCI представляет собой набор данных для многоклассовой классификации. Набор предназначен для распознавания черно-белых пиксельных прямоугольников как одну из 26 заглавных букв английского алфавита, где буквы алфавита представлены в 16 измерениях. Чтобы получить данные, подходящие для обнаружения аномалий, была произведена подвыборка данных из 3 букв, чтобы сформировать нормальный класс, и случайным образом их пары были объединены так, чтобы их размерность удваивалась. Чтобы сформировать класс аномалий, случайным образом были выбраны несколько экземпляров букв, которые не входят нормальный класс, и они были объединены с экземплярами из нормального класса. Процесс объединения выполняется для того, чтобы сделать обнаружение более сложным, поскольку каждый аномальный пример также будет иметь некоторые нормальные значения признаков.

Количество признаков	Количество примеров	Количество нормальных примеров	Количество аномальных примеров
32	1600	1500	100

3.

4. WBC http://odds.cs.stonybrook.edu/wbc/

Исходный набор данных Breast Cancer Wisconsin (Diagnostic) Data Set (WBC) (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) из репозитория машинного обучения UCI представляет собой набор данных для классификации, в котором записываются измерения для случаев рака молочной железы. Есть два класса, доброкачественные и злокачественные. Злокачественный класс этого набора данных уменьшен до 21 точки, которые считаются аномалиями, в то время как точки в доброкачественном классе считаются нормой.

Количество признаков	Количество примеров	Количество нормальных примеров	Количество аномальных примеров
30	378	357	21

Набор данных уже разделен на обучающую и тестовую выборку. Для загрузки данных можно использовать:

```
# загрузка выборок
train = np.loadtxt('train.txt', dtype=float)
test = np.loadtxt('test.txt', dtype=float)
```

Имена файлов отличаются в зависимости от номера варианта.

Создание и обучение автокодировщика

Для создания, обучения и сохранения автокодировщика AE3 предлагается использовать функцию `lib.create_fit_save_ae`. Функция принимает на вход следующие данные в качестве параметров:

- данные для обучения,
- путь и имя файла для сохранения автокодировщика,
- путь и имя файла для сохранения порога обнаружения аномалий,
- количество эпох обучения,
- **`True`** или **`False`** для визуализации или сокрытия процесса обучения,
- а также параметр `patience`, задающий число эпох, в течение которых необходимо продолжать обучение, если ошибка обучения перестала уменьшаться. По истечение заданного числа эпох происходит ранняя остановка обучения. Данный параметр позволяет избежать с одной стороны переобучения (когда веса нейросети настраиваются для распознавания исключительно примеров обучающего набора), а с другой – попадания в локальный минимум ошибки обучения. В случае, если ошибка обучения перестала уменьшаться и

обучение прерывается слишком рано, нейронная сеть может оказаться не до конца обученной, если слишком поздно – переобученной.

Автокодировщик рекомендуется обучать **как минимум** до MSE_stop порядка 0.01 – 0.1.

Параметр **patience** рекомендуется выбирать не менее 5000.

При этом следовать следующим рекомендациям по подбору архитектуры автокодировщика и количества эпох:

Для набора данных Cardio: количество скрытых слоев не менее 7, количество нейронов в узком слое не менее 10, количество эпох не менее 100 000.

Для набора данных Letter: количество скрытых слоев не менее 7, количество нейронов в узком слое не менее 7, количество эпох не менее 100 000.

Для набора данных WBC: количество скрытых слоев не менее 9, количество нейронов в узком слое не менее 7, количество эпох не менее 50 000.

Для оценки качества обучения нейронной сети следует оценить с какой ошибкой автокодировщик восстанавливает (реконструирует) примеры обучающей выборки на выходном слое. Для этого можно воспользоваться ранее описанной функцией `lib.ire_plot` и построить график ошибки реконструкции обучающей выборки. Функция принимает на вход имя для заголовка рисунка (для обучающей выборки следует указать `'training'`), массив значений ошибки реконструкции и порог ошибки реконструкции, а также имя автокодировщика.

К сожалению, для расчета характеристик EDCA для случая многомерного пространства признаков, требуется существенно большее количество времени и вычислительных ресурсов, поэтому в данной части лабораторной работы они не используются.

Тестирование автокодировщика

Для тестирования обученного автокодировщика можно воспользоваться функцией *lib.predict_ae*. Функция принимает на вход обученный автокодировщик, тестовый набор и соответствующий порог обнаружения аномалий. Выходом функции являются: массив ответов автокодировщика (predicted_labels3) и массив ошибок реконструкции для тестового набора (ire3).

Для вывода результатов классификации в виде таблицы и построения графика ошибки реконструкции тестового набора необходимо использовать ранее описанные функции:

```
# тестирование AE3
lib.anomaly_detection_ae(predicted_labels3, ire3, IREth3)
lib.ire_plot('test', ire3, IREth3, 'AE3')
```

Цель задания: построить такой автокодировщик, чтобы достичь с его помощью, как минимум 70% точности обнаружения аномалий в тестовом наборе.