Проект

Искусственные нейронные сети в задаче классификации

С. Лагутин

2018 год

Содержание

1. Обзор

1.1. Классификация

Классификация объектов — одна из стандартных задач машинного обучения. Её можно описать так: имеется множество объектов, которые каким-то образом разделены на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется *обучающей выборкой*. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект (то есть указать к какому классу он относится) из исходного множества.

В машинном обучении задача классификации относится к разделу *обучения с учителем*. Существует также *обучение без учителя*, когда разделение объектов обучающей выборки на классы не задаётся, и требуется классифицировать объекты только на основе их сходства друг с другом. В этом случае принято говорить о *задачах класстеризации*.

Одним из самых простых типов классификации является *бинарная классификация*, когда различных классов всего два. Данный тип служит основой для решения более сложных задач.

1.2. Методы решения

Для решения задач классификации могут использоваться следующие методы:

- Байесовский классификатор;
- Решающие деревья;
- Логистическая регрессия;
- Искусственные нейронные сети.

Байесовский классификатор — тип алгоритмов классификации, основанный на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптимален, то есть обладает минимальной вероятностью ошибок. На практике плотности распределения классов, как правило, не известны. Их приходится оценивать по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. В задаче бинарной классификации звуков восстановление плотности классов является плохо решаемой проблемой.

Решающие деревья — средство поддержки принятия решений, структура которого представляет собой *листья* и *ветки*. На ветках дерева записаны атрибу-

ты, от которых зависит целевая функция, в листьях записаны значения целевой функции, а в остальных узлах — атрибуты, по которым различаются случаи. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной на основе нескольких переменных на входе.

Одним из основных вопросов в реализации решающих деревьев для задачи классификации является выбор атрибутов, по которым будет осуществляться разделение данных на классы.

Логистическая регрессия — метод построения линейной разделяющей поверхности. В случае двух классов разделяющей поверхностью является гиперплоскость. В задаче бинарной классификации звуков нельзя гарантировать возможность разделения пространства параметров одной гиперплоскостью.

Искусственная нейронная сеть — это математическая модель, построенная в некотором смысле по образу и подобию сетей нервных клеток живого организма. Для решения задачи классификации может использоваться такой тип ИНС, как *многослойный перцептрон Розенблатта*. Он представляет собой передающую сеть, состоящую из генераторов сигнала трёх типов: сенсорных элементов, ассоциативных элементов и реагирующих элементов. Производящие функции этих элементов зависят от сигналов, возникающих либо где-то внутри передающей сети, либо, для внешних элементов, от сигналов, поступающих из внешней среды.

2. Постановка задачи

Имеется набор данных — оцифрованные записи звука удара мячей для настольного тенниса. Каждая запись длительностью примерно 3-6 секунд. Всего записей 100 штук. Все представлены в виде wave-файлов, пронумерованных от 1 до 100.

Записи с 1 по 50 соответствуют сломанным мячам. Типов сломанных мячей четыре: мячи с вмятинами с одной стороны, мячи с проколами и разрывами с одной стороны, мячи из одной целой полусферы, мячи с вмятинами и проколами с разных сторон.

Записи с 51 по 100 соответствуют целым мячам.

Все записи были сделаны при ударах о разные типы поверхностей: кафельный пол, металлическая поверхность электрической плиты, деревянная поверхность стола, пластиковая поверхность подоконника.

Необходимо, используя разные компоновки обучающей и контрольной выборки из исходных данных, получить предсказания о классификации каждого примера с возможной ошибкой не более 10% неправильных ответов от размера контрольной выборки.

3. Решение задачи

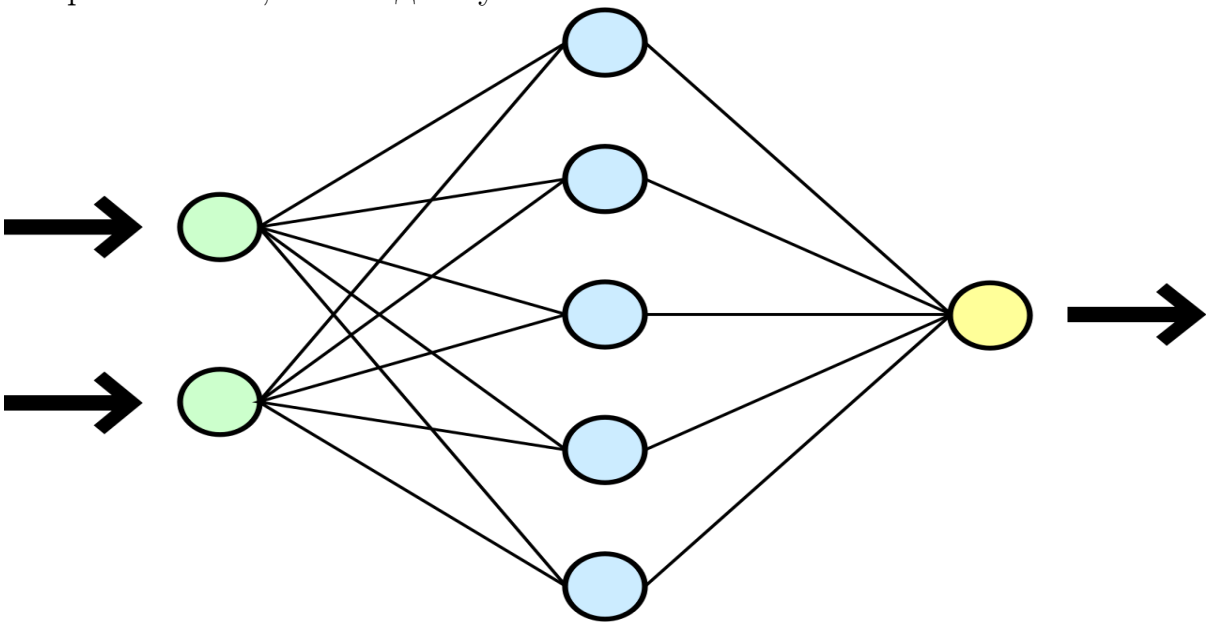
Для решения поставленной задачи я использовал многослойный перцептрон Розенблатта.

3.1. Перцептрон

Перцептрон состоит из нескольких слоёв нейронов:

1. Входной слой, содержащий псевдо-нейроны, которые передают дальше значения *предикторов* — параметров объекта;
2. Один или несколько скрытых слоёв;
3. Выходной слой, содержащий один нейрон.

Передача сигналов (активация) нейронной сети происходит от входного слоя, через скрытые слои, к выходному слою.



Все нейроны (кроме входного слоя) имеют одинаковое строение, состоят из двух частей — сумматорной и активационной функций. Сумматорная функция определяет то, как нейрон будет использовать входящую информацию из предыдущего слоя. Активационная функция определяет реакцию нейрона, которая будет передана по всем выходным связям в следующий слой.

В качестве сумматорной функции выбрана взвешенная сумма всех входящих сигналов:

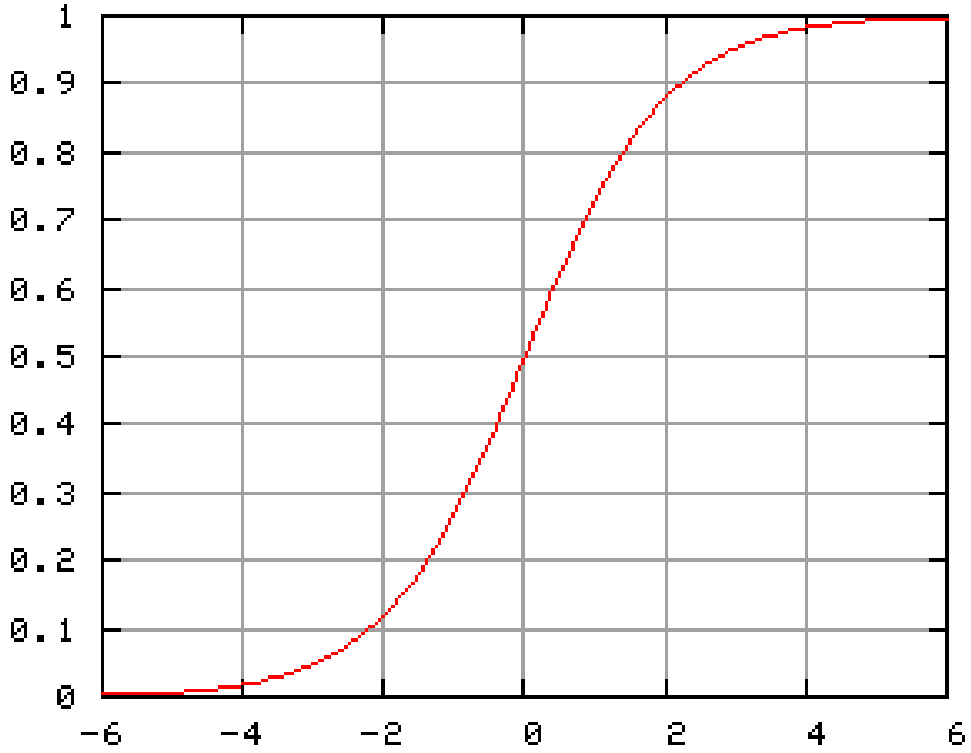
$$S = b + \sum_{j=1}^{m} x_j w_j$$

где m — количество входящих сигналов нейрона, x_j — значение, получаемое по j-ому входу, w_j — вес j-ого входа, b — некоторое смещение, изменяемое в процессе обучения. Смещение можно учитывать в сумме, если добавить в каждый слой,

кроме выходного на первое место нейрон, у которого значение активации будет всегда равно 1.

Активационная функция — логистическая (сигмоидальная):

$$\sigma(S) = \frac{1}{1 + e^{-S}}$$



Логистическая функция является гладкой, что необходимо для работы алгоритма обучения. Кроме того, её значение можно интерпретировать как вероятность принадлежности объекта к одному из двух классов.

Обучение нейронной сети — это настройка весов для входящих связей всех нейронов, с целью получения достоверных предсказаний. Для обучения используется *алгоритм обратного распространеннния ошибки*, который основывается на градиентном спуске по простанству весов в сторону уменьшения значений целевой функции ошибки.

Для оценки правдоподобности предсказаний используется *квадратичная функция ошибки*:

$$E = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

где N — количество примеров, $\hat{y}_i$ — предсказанное значение для i-ого примера, y_i — правильный ответ для него.

Для того, чтобы понять, как изменится значение функции ошибки при изменении какого-либо веса входящих сигналов нейрона, нужно взять её частную производную по этому весу.

Сначала считается изменение весов в выходном слое:

$$\Delta w_j = -\alpha \frac{\partial E}{\partial w_j}$$

где $\alpha \in \mathbb{R}$ — скорость обучения.

В векторном виде:

$$\Delta W = -\alpha \nabla_W E$$

где $\nabla_W E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_m} \right)$ — градиент E в точке W.

Посчитаем частную производную от функции E по j-му весу:

$$\frac{\partial E}{\partial w_j} = \frac{\partial \left(\frac{1}{2N} \sum_{i=1}^{N} \left(\hat{y}_i - y_i \right)^2 \right)}{\partial w_j}$$

Так как производная суммы равна сумме производных, возьмём для простоты один пример, а после просуммируем все значения:

$$\frac{1}{2} \cdot \frac{\partial (\hat{y} - y)^2}{\partial w_j} = \frac{1}{2} \cdot \frac{\partial \left(\hat{y} - y \right)^2}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_j} = \left(\hat{y} - y \right) \frac{\partial \sigma \left(S \right)}{\partial w_j} =$$

$$= \left(\sigma \left(S \right) - y \right) \sigma\prime \left(S \right) \frac{\partial \sum_{j=1}^{m} x_j w_j}{\partial w_j} = \left(\sigma \left(S \right) - y \right) \sigma \left(S \right) \left(1 - \sigma \left(S \right) \right) x_j$$

Итак, общая формула для j-ого веса по N примерам:

$$\frac{\partial E}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - y_i \right) \hat{y}_i \left(1 - \hat{y}_i \right) x_j$$

Далее полученная ошибка распространяется по ИНС в обратном порядке, от выходного слоя ко входному, изменяя веса скрытых слоёв.

Введём следующие обозначения:

- w_{jk}^l — значение j-ого веса k-ого нейрона в l-ом слое (вес связи из j-ого нейрона $l-1$ слоя в k-ый нейрон l-ого слоя);
- m_l — количество нейронов в l-ом слое;
- s_k^l — значение сумматорной функции k-ого нейрона в l-ом слое;
- a_k^l — значение активационной функции k-ого нейрона в l-ом слое;
- $\delta_k^l = \frac{\partial E}{\partial s_k^l}$ — ошибка k-ого нейрона в l-ом слое.

Зная значение ошибки δ_k^l для каждого нейрона, можно получить соответствующее изменение его весов:

$$\Delta w_{jk}^l = -\alpha \delta_k^l a_j^{l-1}$$

Посчитаем значение ошибки для нейронов выходного (L-ого) слоя. Для простоты возьмём один пример:

$$\delta_k^L = \frac{\partial E}{\partial s_k^L} = \frac{1}{2} \cdot \frac{\partial \left(\sigma\left(s_k^L\right) - y_k\right)^2}{\partial s_k^L} = \left(a_k^L - y_k\right) a_k^L \left(1 - a_k^L\right)$$

где y_k — правильный ответ для k-ого нейрона выходного слоя.

Теперь выразим ошибку нейрона на l-ом слое через ошибки на $l+1$ слое:

$$\delta_j^l = \sigma\prime\left(s_j^l\right) \sum_{k=1}^{m_{l+1}} w_{jk}^{l+1} \delta_k^{l+1} = a_j^l \left(1 - a_j^l\right) \sum_{k=1}^{m_{l+1}} w_{jk}^{l+1} \delta_k^{l+1}$$

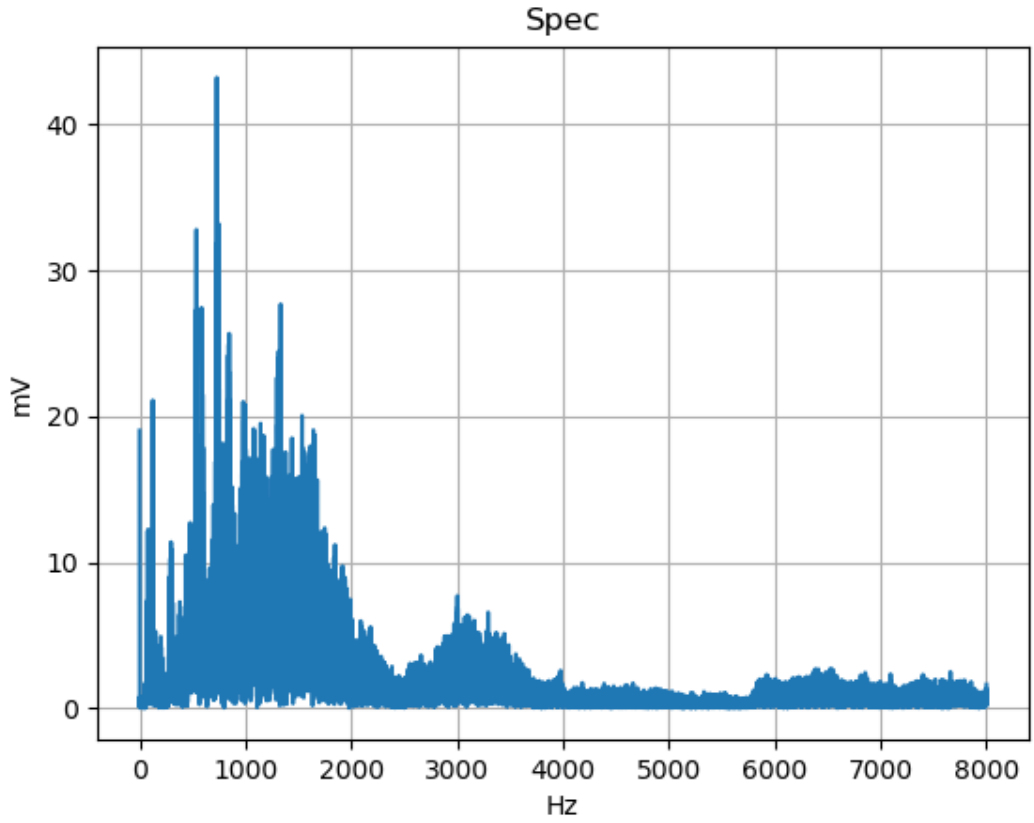
3.2. Предикторы из записей

Для получения численных предикторов из записей звука, массив интенсивности сигналов по времени из wave-файла переводится с помощью быстрого дискретного преобразования Фурье в спектр — массив интенсивности частот.

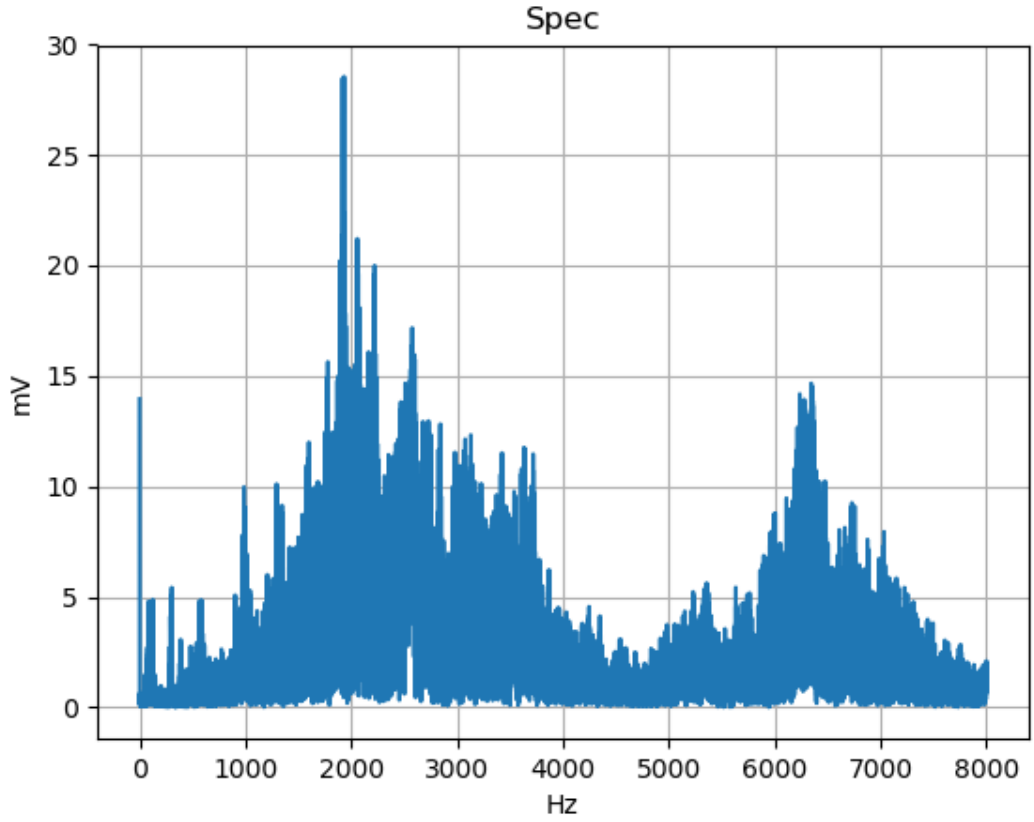
Если посмотреть на спектры сломанных и целых мячей, можно отчётливо заметить разницу: у целых мячей имеется узкий диапазон частот, которые имеют высокую интенсивность, по сравнению с остальными частотами; у сломанных же мячей спектры имеют несколько не очень высоких пиков.

Таким образом в качестве предикторов можно использовать значения нескольких самых интенсивных частот, или, другими словами — значения частот, в которых есть пики интенсивности в спектре.

Пример спектра записи звука целого мяча:



Пример спектра записи звука повреждённого мяча:



4. Реализация

4.1. Подготовка предикторов

Для подготовки предикторов из записей звука используется библиотека работы с wave-файлами и быстрое преобразование Фурье из библиотеки **numpy**, реализованные в Python 3.

Весь исходный код для подготовки предикторов находится в подкаталоге **./predictors/**.

В файле **wavetransform.py** написан код, который последовательно обрабатывает файлы из подкаталога с записями звуков **./wav/** следующим образом:

1. Из файла считывается массив сэмплов;
2. Данный массив преобразуется при помощи функции `numpy.rfft` в спектр;
3. Во временные промежуточные файлы (для сломанных и целых мячей отдельно) записывается промежуточный результат: сначала интенсивности, как абсолютные значения из спектра, затем — частоты, которые соответствуют данному распределению.

После этого работает код, написанный в файле **predictors.cpp**:

1. Каждые интенсивность и частоты связываются в единый массив пар, в которых на первом месте стоит значение интенсивности, на втором — соответствующая частота `std::pair<double, double>`;
2. Полученный массив сортируется по неубыванию интенсивностей в спектре;
3. Из отсортированного массива выводятся 20 частот с конца, для каждого примера выводятся значения правильных ответов на него (1 — для целых, 0 — для сломанных мячей).

Выходной файл с предикторами создаётся в корневом каталоге проекта. Его имя генерируется по времени запуска генерации и имеет вид **pred*.p**.

4.2. Нейросеть

Исходный код основного приложения находится в подкаталоге **./src/**.

Подготовленные данные считываются из файла с предикторами в функции `reading` (реализация в **body.cpp**) в массив векторов `input`, правильные ответы в вектор `y`.

Нейросеть реализована в виде класса `NeuronNet`, в котором создаются вектора, соответствующие слоям нейросети, содержащим отдельные нейроны, а также вектора значений активаций нейронов каждого слоя. Предсказание нейросети на конкретном примере производится вызовом метода `NeuronNet::forward_pass`.

В нейроне, реализованном в классе `Neuron`, содержится вектор весов для входящих сигналов этого нейрона, а также два метода: сумматорная и активационная функции. Исходные значения весов задаются случайно. Активационная функция задаётся через конструктор нейросети и хранится как указатель на функцию (по умолчанию — сигмоида).

Метод `NeuronNet::forward_pass` получает в качестве параметра вектор значений предикторов одного примера и проводит последовательную активацию нейронов по слоям от входного к выходному, после чего возвращает значение активации нейрона в выходном слое.

Метод `NeuronNet::backpropagation` реализует алгоритм обратного распространения ошибки. Он получает в качестве параметров вектор значений предикторов и правильный ответ для одного примера, а также константу скорости обучения, производные активационной функции нейронов и целевой функции ошибки (по умолчанию — квадратичная функция ошибки).

Сначала считается значение ошибки на выходном слое. Затем ошибка распространяется на предыдущие слои в обратном порядке. Для каждого слоя хранится временный вектор ошибок `wdelt`.

Обучение нейросети происходит в функции `learning` (**body.cpp**). В данной функции заданное количество раз (значение параметра `epoch`) вызывается метод `NeuronNet::backpropagation`, с передачей ему случайного примера из входных данных.

Функция `check` (**body.cpp**) выводит предсказания по выборке, передаваемой в качестве параметра, а также возвращает количество ошибочных предсказаний в процентах от размера выборки.

5. Тестирование

Во всех тестах в нейросети использовалось два скрытых слоя, количество эпох обучения: 50000.

5.1. Тест 0

В данном тесте обучающая выборка равна контрольной и содержит все 100 примеров из исходных данных. Наилучший результат получается при конфигурации сети по 10 нейронов на каждом скрытом слое, скорость обучения $\alpha = 0.05$.

Ошибочных предсказаний для контрольной выборки 6%.

```
0.91466 => whole [WHOLE]   0.93033 => whole [WHOLE]    0.07389 => broken [BROKEN]
0.92536 => whole [WHOLE]   0.94007 => whole [WHOLE]    0.01795 => broken [BROKEN]
0.86948 => whole [WHOLE]   0.93053 => whole [WHOLE]    0.10768 => broken [BROKEN]
0.92332 => whole [WHOLE]   0.93854 => whole [WHOLE]    0.01942 => broken [BROKEN]
0.06457 => broken [WHOLE]  0.92613 => whole [WHOLE]    0.01784 => broken [BROKEN]
0.76195 => whole [WHOLE]   0.93447 => whole [WHOLE]    0.11092 => broken [BROKEN]
0.88933 => whole [WHOLE]   0.74654 => whole [WHOLE]    0.16283 => broken [BROKEN]
0.91868 => whole [WHOLE]   0.87381 => whole [WHOLE]    0.02781 => broken [BROKEN]
0.93270 => whole [WHOLE]   0.89680 => whole [WHOLE]    0.01773 => broken [BROKEN]
0.91229 => whole [WHOLE]   0.85227 => whole [WHOLE]    0.14024 => broken [BROKEN]
0.93687 => whole [WHOLE]   0.90901 => whole [WHOLE]    0.02727 => broken [BROKEN]
0.93814 => whole [WHOLE]   0.87969 => whole [WHOLE]    0.42470 => broken [BROKEN]
0.91840 => whole [WHOLE]   0.75836 => whole [WHOLE]    0.09244 => broken [BROKEN]
0.92433 => whole [WHOLE]   0.90925 => whole [WHOLE]    0.01776 => broken [BROKEN]
0.93472 => whole [WHOLE]   0.05479 => broken [BROKEN]  0.14121 => broken [BROKEN]
0.93497 => whole [WHOLE]   0.06725 => broken [BROKEN]  0.01778 => broken [BROKEN]
0.93500 => whole [WHOLE]   0.02194 => broken [BROKEN]  0.01772 => broken [BROKEN]
0.92785 => whole [WHOLE]   0.01793 => broken [BROKEN]  0.09035 => broken [BROKEN]
0.79843 => whole [WHOLE]   0.01791 => broken [BROKEN]  0.18876 => broken [BROKEN]
0.83747 => whole [WHOLE]   0.06550 => broken [BROKEN]  0.01798 => broken [BROKEN]
0.85760 => whole [WHOLE]   0.03068 => broken [BROKEN]  0.04425 => broken [BROKEN]
0.91041 => whole [WHOLE]   0.01813 => broken [BROKEN]  0.02922 => broken [BROKEN]
0.81506 => whole [WHOLE]   0.11732 => broken [BROKEN]  0.06879 => broken [BROKEN]
0.84865 => whole [WHOLE]   0.93463 => whole [BROKEN]   0.35827 => broken [BROKEN]
0.89208 => whole [WHOLE]   0.89746 => whole [BROKEN]   0.14423 => broken [BROKEN]
0.93433 => whole [WHOLE]   0.11028 => broken [BROKEN]  0.01813 => broken [BROKEN]
0.93554 => whole [WHOLE]   0.92953 => whole [BROKEN]   0.01810 => broken [BROKEN]
0.93209 => whole [WHOLE]   0.91972 => whole [BROKEN]   0.07674 => broken [BROKEN]
0.90710 => whole [WHOLE]   0.01784 => broken [BROKEN]
0.92311 => whole [WHOLE]   0.01763 => broken [BROKEN]
0.92684 => whole [WHOLE]   0.02453 => broken [BROKEN]
0.90673 => whole [WHOLE]   0.07670 => broken [BROKEN]
0.85894 => whole [WHOLE]   0.01780 => broken [BROKEN]
0.80472 => whole [WHOLE]   0.02327 => broken [BROKEN]
0.93985 => whole [WHOLE]   0.93476 => whole [BROKEN]
0.92340 => whole [WHOLE]   0.19039 => broken [BROKEN]
```

5.2. Тест 1

В данном тесте обучающая выборка содержит 13 примеров сломанных мячей и 13 целых. Контрольная выборка состоит из 37 сломанных и 37 целых мячей. Наилучший результат получается при конфигурации сети по 10 нейронов на каждом скрытом слое, скорость обучения $\alpha = 0.05$.

Ошибочных предсказаний для контрольной выборки 8.11%.

```
0.86883 => whole [WHOLE]    0.18323 => broken [BROKEN]
0.83513 => whole [WHOLE]    0.04862 => broken [BROKEN]
0.89146 => whole [WHOLE]    0.04362 => broken [BROKEN]
0.68832 => whole [WHOLE]    0.18867 => broken [BROKEN]
0.84803 => whole [WHOLE]    0.09965 => broken [BROKEN]
0.88611 => whole [WHOLE]    0.04489 => broken [BROKEN]
0.88224 => whole [WHOLE]    0.90610 => whole [BROKEN]
0.89941 => whole [WHOLE]    0.88863 => whole [BROKEN]
0.90742 => whole [WHOLE]    0.24989 => broken [BROKEN]
0.89497 => whole [WHOLE]    0.87759 => whole [BROKEN]
0.90630 => whole [WHOLE]    0.04365 => broken [BROKEN]
0.90653 => whole [WHOLE]    0.04350 => broken [BROKEN]
0.87365 => whole [WHOLE]    0.23854 => broken [BROKEN]
0.07257 => broken [WHOLE]   0.04323 => broken [BROKEN]
0.53595 => whole [WHOLE]    0.05355 => broken [BROKEN]
0.09779 => broken [WHOLE]   0.37166 => broken [BROKEN]
0.75706 => whole [WHOLE]    0.22746 => broken [BROKEN]
0.74206 => whole [WHOLE]    0.04383 => broken [BROKEN]
0.90103 => whole [WHOLE]    0.06506 => broken [BROKEN]
0.89372 => whole [WHOLE]    0.04334 => broken [BROKEN]
0.88410 => whole [WHOLE]    0.13947 => broken [BROKEN]
0.89364 => whole [WHOLE]    0.05799 => broken [BROKEN]
0.88014 => whole [WHOLE]    0.04276 => broken [BROKEN]
0.88855 => whole [WHOLE]    0.30227 => broken [BROKEN]
0.78265 => whole [WHOLE]    0.24371 => broken [BROKEN]
0.91362 => whole [WHOLE]    0.23174 => broken [BROKEN]
0.87802 => whole [WHOLE]    0.04303 => broken [BROKEN]
0.91140 => whole [WHOLE]    0.04299 => broken [BROKEN]
0.89917 => whole [WHOLE]    0.04261 => broken [BROKEN]
0.91231 => whole [WHOLE]    0.22592 => broken [BROKEN]
0.89441 => whole [WHOLE]    0.04358 => broken [BROKEN]
0.54544 => whole [WHOLE]    0.08008 => broken [BROKEN]
0.77022 => whole [WHOLE]    0.06038 => broken [BROKEN]
0.70774 => whole [WHOLE]    0.74800 => whole [BROKEN]
0.84532 => whole [WHOLE]    0.27998 => broken [BROKEN]
0.81911 => whole [WHOLE]    0.04494 => broken [BROKEN]
0.82465 => whole [WHOLE]    0.18266 => broken [BROKEN]
```

5.3. Тест 2

В данном тесте обучающая выборка содержит 20 примеров сломанных мячей и 10 целых. Контрольная выборка состоит из 30 сломанных и 40 целых мячей. Наилучший результат получается при конфигурации сети: 15 нейронов на первом и 10 нейронов на втором скрытом слоях, скорость обучения $\alpha = 0.08$.

Ошибочных предсказаний для контрольной выборки 10%.

```
0.75277 => whole [WHOLE]    0.65612 => whole [WHOLE]
0.76631 => whole [WHOLE]    0.40393 => broken [WHOLE]
0.71750 => whole [WHOLE]    0.70864 => whole [WHOLE]
0.53100 => whole [WHOLE]    0.06657 => broken [BROKEN]
0.72068 => whole [WHOLE]    0.05830 => broken [BROKEN]
0.82525 => whole [WHOLE]    0.13887 => broken [BROKEN]
0.84041 => whole [WHOLE]    0.05913 => broken [BROKEN]
0.79710 => whole [WHOLE]    0.82716 => whole [BROKEN]
0.80766 => whole [WHOLE]    0.16643 => broken [BROKEN]
0.82814 => whole [WHOLE]    0.80560 => whole [BROKEN]
0.75752 => whole [WHOLE]    0.05825 => broken [BROKEN]
0.12478 => broken [WHOLE]   0.07749 => broken [BROKEN]
0.45073 => broken [WHOLE]   0.13636 => broken [BROKEN]
0.63429 => whole [WHOLE]    0.05800 => broken [BROKEN]
0.08960 => broken [WHOLE]   0.07174 => broken [BROKEN]
0.66710 => whole [WHOLE]    0.14314 => broken [BROKEN]
0.61502 => whole [WHOLE]    0.05845 => broken [BROKEN]
0.71656 => whole [WHOLE]    0.11781 => broken [BROKEN]
0.81895 => whole [WHOLE]    0.06956 => broken [BROKEN]
0.81162 => whole [WHOLE]    0.15220 => broken [BROKEN]
0.75327 => whole [WHOLE]    0.20689 => broken [BROKEN]
0.80467 => whole [WHOLE]    0.05779 => broken [BROKEN]
0.78920 => whole [WHOLE]    0.07544 => broken [BROKEN]
0.80883 => whole [WHOLE]    0.29598 => broken [BROKEN]
0.62333 => whole [WHOLE]    0.15864 => broken [BROKEN]
0.85052 => whole [WHOLE]    0.16823 => broken [BROKEN]
0.78192 => whole [WHOLE]    0.05791 => broken [BROKEN]
0.81010 => whole [WHOLE]    0.15991 => broken [BROKEN]
0.83463 => whole [WHOLE]    0.08864 => broken [BROKEN]
0.81017 => whole [WHOLE]    0.07450 => broken [BROKEN]
0.83369 => whole [WHOLE]    0.12645 => broken [BROKEN]
0.79327 => whole [WHOLE]    0.78505 => whole [BROKEN]
0.80847 => whole [WHOLE]    0.05913 => broken [BROKEN]
0.63732 => whole [WHOLE]
0.68006 => whole [WHOLE]
0.65621 => whole [WHOLE]
0.73782 => whole [WHOLE]
```

5.4. Тест 3

В данном тесте обучающая выборка содержит 10 примеров сломанных мячей и 20 целых. Контрольная выборка состоит из 40 сломанных и 30 целых мячей. Наилучший результат получается при конфигурации сети по 10 нейронов на каждом скрытом слое, скорость обучения $\alpha = 0.05$.

Ошибочных предсказаний для контрольной выборки 8.57%.

```
0.86127 => whole [WHOLE]     0.93381 => whole [BROKEN]
0.80693 => whole [WHOLE]     0.24604 => broken [BROKEN]
0.90649 => whole [WHOLE]     0.91213 => whole [BROKEN]
0.69740 => whole [WHOLE]     0.88096 => whole [BROKEN]
0.83617 => whole [WHOLE]     0.07286 => broken [BROKEN]
0.90881 => whole [WHOLE]     0.10212 => broken [BROKEN]
0.91480 => whole [WHOLE]     0.22461 => broken [BROKEN]
0.93470 => whole [WHOLE]     0.06964 => broken [BROKEN]
0.93471 => whole [WHOLE]     0.09078 => broken [BROKEN]
0.86749 => whole [WHOLE]     0.93400 => whole [BROKEN]
0.50794 => whole [WHOLE]     0.28443 => broken [BROKEN]
0.73944 => whole [WHOLE]     0.07039 => broken [BROKEN]
0.13996 => broken [WHOLE]    0.15395 => broken [BROKEN]
0.71808 => whole [WHOLE]     0.09167 => broken [BROKEN]
0.92516 => whole [WHOLE]     0.06978 => broken [BROKEN]
0.91195 => whole [WHOLE]     0.20343 => broken [BROKEN]
0.84947 => whole [WHOLE]     0.29275 => broken [BROKEN]
0.91575 => whole [WHOLE]     0.09870 => broken [BROKEN]
0.73113 => whole [WHOLE]     0.06898 => broken [BROKEN]
0.88338 => whole [WHOLE]     0.41319 => broken [BROKEN]
0.92208 => whole [WHOLE]     0.23660 => broken [BROKEN]
0.94421 => whole [WHOLE]     0.06927 => broken [BROKEN]
0.90788 => whole [WHOLE]     0.21689 => broken [BROKEN]
0.92560 => whole [WHOLE]     0.06872 => broken [BROKEN]
0.62820 => whole [WHOLE]     0.23812 => broken [BROKEN]
0.75536 => whole [WHOLE]     0.07011 => broken [BROKEN]
0.83552 => whole [WHOLE]     0.13148 => broken [BROKEN]
0.85612 => whole [WHOLE]     0.21372 => broken [BROKEN]
0.61075 => whole [WHOLE]     0.82134 => whole [BROKEN]
0.81555 => whole [WHOLE]     0.27594 => broken [BROKEN]
0.20091 => broken [BROKEN]   0.07195 => broken [BROKEN]
0.08352 => broken [BROKEN]   0.07163 => broken [BROKEN]
0.07009 => broken [BROKEN]   0.21029 => broken [BROKEN]
0.07021 => broken [BROKEN]
0.20264 => broken [BROKEN]
0.07178 => broken [BROKEN]
0.14245 => broken [BROKEN]
```

6. Использованные источники

1. machinelearning.ru — описание задачи классификации, перцептрона Розенблатта и других методов машинного обучения;
2. stepik.org — практический курс по нейронным сетям от Института биоинформатики;
3. habr.com — статья про распознавание речи, в частности способ получения предикторов из звука.