УДК 004.032.26

**А.М. Заяц, С.П. Хабаров**

# ПОСТРОЕНИЕ НЕЙРОННОЙ СЕТИ КЛАССИФИКАЦИИ ИРИСОВ ФИШЕРА НА БАЗЕ JAVASCRIPT

*Введение.* Во многих литературных источниках для иллюстрации работы различных алгоритмов классификации и кластеризации используется ставший уже классическим набор данных, который получил название Ирисы Фишера [Википедия, 2018; Machine Leaning Repository, 2018]. Этот набор данных содержит сведения о 150 экземплярах цветов ириса, о 50 экземплярах каждого из трёх видов: это ирис щетинистый (iris setosa), ирис разноцветный (iris versicolor) и ирис виргинский (iris virginica).

Для каждого экземпляра растения заданы четыре характеристики: длина чашелистика, ширина чашелистика, длина лепестка и ширина лепестка. Срез исходного набора данных по двум параметрам (рис. 1) позволяет заключить, что один из этих классов, а именно – iris setosa является линейно-разделимым от двух остальных, чего нельзя сказать о двух других классах.

Поставим задачу построения нейронной сети, которая будучи обученной на основе этого набора данных могла бы в дальнейшем определять класс ириса по данным его четырех измерений. Особенность постановки задачи состоит в том, что ее предлагается решить без использования каких-либо дополнительных программных оболочек, ограничившись только возможностями операционной системы и стандартного браузера.

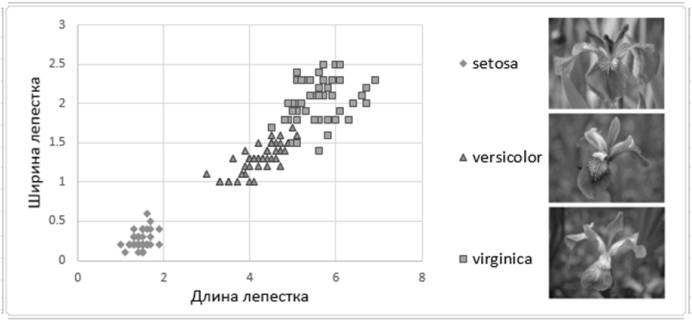


*Рис. 1*. Срез набора данных Ирисы Фишера по двум параметрам
*Fig. 1.* The Iris Fisher dataset slice by two parameters

*Методика и результаты исследования*. Методика решения поставленной задачи базируется на исследовании структур многослойных нейронных сетей, обучаемых в соответствии с алгоритмом обратного распространения ошибки. Если потребовать, чтобы для этого исследования было достаточно среды стандартного браузера, то встанет сопутствующая задача разработки HTML-страницы с набором функций на JavaScript.

Эти функции для настройки и обучения нейронной сети могут использовать либо собственную реализацию алгоритма обратного распространения, либо обращение к какой-либо из внешних библиотек. Например, при использовании библиотеки BrainJS (https://cdn.rawgit.com/BrainJS/brain.js/master/browser.js) процесс создания нейронной сети, ее обучения и тестирования требует выполнения всего трех основных операций, а именно:

– создания нового объекта класса *brain.NeuralNetwork*();

– обучения вновь созданной нейронной сети с использованием метода *train*(), входным аргументом которого является массив экземпляров обучающей выборки;

– тестирования работы сети с использованием метода *run*(), входной аргумент которого – экземпляр выборки, а выход – предсказанное нейронной сетью выходное значение.

При этом обучающая выборка должна быть представлена в виде ассоциативного массива с двумя ключами input и output. В каждом из экземпляров обучающей выборки эти ключи должны определять массив чисел, соответствующих входным и выходным значениям конкретного экземпляра обучающей выборки.

Структурная организация HTML-страницы исследования разных структур нейронных сетей должна предусматривать как загрузку файла с исходным набором данных, так и возможность его разделения на обучающую и тестовую выборки. Кроме того, должна быть предусмотрена возможность задания структуры нейронной сети и параметров ее обучения. Один из вариантов такой страницы может иметь вид, представленный на рис. 2.

Для удобства исследования нейронных сетей в нижней части шаблона этой интерфейсной HTML-страницы необходимо предусмотреть область для вывода результатов тестирования сети и просмотра значений весов связей между ее нейронами. Такая область может быть представлена тегом, например: <pre id="log"></pre>.

Процесс создания, обучения и тестирования нейронной сети классификации Ирисов Фишера начинается с нажатия кнопки «Выберите файл», что вызовет открытие стандартного окна загрузки файла с диска. После выбора нужного файла он подгружается на эту страницу и отображается в окне тега <textarea>, который имеет идентификатор id="txt" (рис. 3).

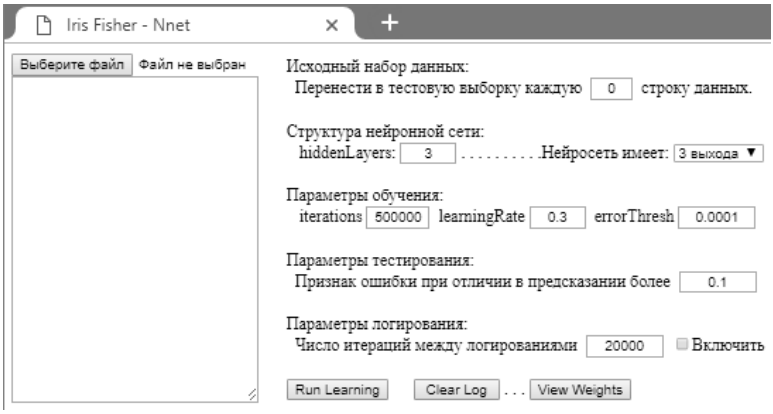


*Рис. 2*. Вид HTML-страницы настройки нейронной сети
*Fig. 2.* View of the neural network setup html page

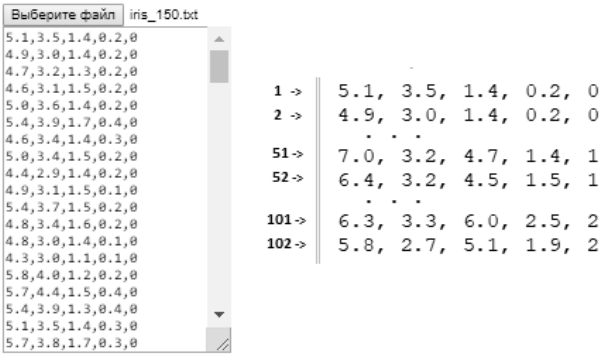


*Рис. 3*. Вид подгруженного на HTML-страницу набора данных по ирисам
*Fig. 3.* View of iris dataset loaded onto an HTML-page

После этого появляется возможность многократной настройки различных вариантов структур нейронной сети без перезагрузки HTML-страницы и повторного выбора файла с исходным набором данных. Для задачи классификации Ирисов Фишера можно воспользоваться скопированным из интернета текстовым файлом iris_150.txt, который содержит 150 строк по 50 для каждого класса. Элементы каждой строки разделяют запятые. Первые четыре элемента – длина и ширина чашелистика, длина

и ширина лепестка (см), а пятый – класс растения (0 – setosa, 1 – versicolor, 2 – virginica).

**Формирование обучающей и тестовой выборок.** Первым шагом построения нейронной сети на базе HTML и JavaScript с использованием библиотеки BrainJS является формирование на базе исходного текстового файла обучающей выборки, которая должна иметь структуру данных, воспринимаемую библиотекой BrainJS. Для этого надо:

– выполнить чтение текстового файла и сохранить его содержимое в памяти HTML-страницы;

– сформировать на этой основе обучающую и тестовую выборки, воспринимаемые библиотекой BrainJS.

Для решения первой задачи можно использовать реализованную в последних версиях браузеров возможность взаимодействовать с локальными файлами с использованием HTML5 и соответствующей спецификацией API для работы с файлами. Самый простой способ загрузки файлов – это использование стандартного элемента, который определяется тегом типа <input type="file">.

Процедура обработки изменения состояния этого тега позволяет получить список выбранных объектов типа File в виде объекта типа FileList. Так как требуется загрузка только одного конкретного файла, то в процедуре следует использовать только нулевой элемент этого списка, а именно – files[0].

```
// Чтение данных текстового файла в тег textarea с именем "txt"
function readFile() {
 var selectedFile =document.getElementById('inputFile').files[0];
 var reader = new FileReader();
 reader.readAsText(selectedFile);
 reader.onload = function (event) {
        document.all("txt").value = event.target.result;
 };
}
```

Получив ссылку на объект типа File, можно создать экземпляр объекта типа FileReader (reader). Именно он и сохранит содержимое файла в своей памяти. По окончании загрузки файла возникнет событие onload. Его атрибут result можно использовать для доступа к содержимому файла. Результат чтения всегда представлен в виде event.target.result. Есть ряд форматов, в которых объект FileReader может представлять данные из файла. Формат должен быть задан при открытии файла для чтения. Так, метод

```
FileReader.readAsText(Blob|File, opt encoding)
```

возвращает содержимое файла в текстовом формате (plain/text). По умолчании используется кодировка UTF-8. Чтобы задать другой формат, надо использовать необязательный параметр кодировки (opt_encoding).

Таким образом, первую задачу можно считать решенной: содержимое файла с исходным набором данных сохранено в теге <textarea> с id="txt", и можно переходить к формированию на его основе обучающей выборки.

При этом как обучающая, так и тестовая выборки, при использовании библиотеки BrainJS должны быть представлены в виде ассоциативного массива с двумя ключами – input и output. В каждом из экземпляров выборки ключи определяют массивы чисел, соответствующих входным и выходным значениям конкретного экземпляра выборки. Один из вариантов процедуры преобразования исходного набора данных в обучающие и тестовые выборки может иметь вид:

```
// Преобразование файла в массив данных, используемый в BrainJS
function ConvertFileToData(txtFile) {
  // Выделение из файла отдельных строк - их будет line_count
  var line = txtFile.split("\n");
  var line_count = line.length - 1;
  var data_learn=[], data_test=[];
  for (var n = 0; n < line_count; n++) {
      // Выделение из строк отдельных элементов (входы + выход)
     var el = line[n].split(",").map(Number);
     // Из первых элементов формируется новый массив входов
     var inp = el.slice(0,el.length-1);
      // Для текущей строки выборки формируем новый массив выходов
     if (document.all('out').value == "3 выхода") {
         var out = [0,0,0];
         iris_class = el[el.length-1];
         out[iris_class] = 1;
     } else {
         var out = [0.5 * el[el.length-1]];
     }
     // Формирование массивов данных для использования в brainjs
     var nt = Number(document.all('intest').value);
     // Выделение тестовой выборка из основной
     if ((n % nt)==0){ data_test.push({input: inp, output: out})
      } else { data_learn.push({input: inp, output: out}) }
  }
  return {learn: data_learn, test: data_test }
}
```

Особенность данной процедуры в том, что она позволяет формировать наборы данных для исследования двух различных структур нейронных сетей: либо с тремя нейронами в выходном слое, либо всего с одним (рис. 4).

В первом случае значение классов растений (0, 1, 2) исходного набора данных заменяются векторами output:[1,0,0], output:[0,1,0] и output:[0,0,1] в обучающих и тестовых выборках. Во втором случае выходной вектор будет содержать всего один элемент, значения которого получены путем нормализации значения класса растения, т. е. приведения его в диапазон значений от 0 до 1. В этом случае выходной вектор будет иметь вид – output:[0.0], output:[0.5] и output:[1.0] соответственно для каждого из классов растений.

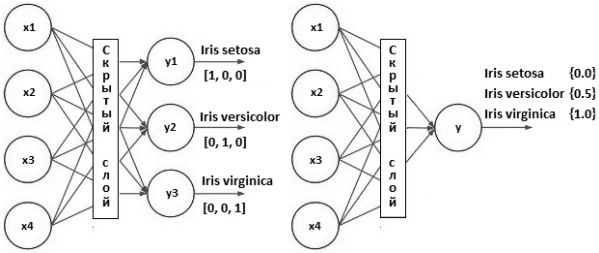


*Рис. 4*. Возможные структуры нейронных сетей
*Fig. 4.* Possible structures of neural networks

Еще одной особенностью функции ConvertFileToData(txtFile) является то, что она позволяет на базе исходного набора данных формировать различные варианты обучающих и тестовых выборок. Это позволяет исследовать качество обучения нейросетевого классификатора Ирисов Фишера на разных наборах данных.

Изменяя значения параметра «Перенести в тестовую выборку каждую ... строку данных» в интерфейсной HTML-форме, появляется возможность разделения исходного набора данных на обучающую и тестовую выборки в разных пропорциях. Так, например, при значении этого параметра, равном 25, обучающая выборка будет содержать 144 экземпляра данных, а тестовая выборка – 6 экземпляров. Значение 16 этого параметра разделит исходный набор данных в отношении 140 и 10, а значение, равное 0, – в отношении 150 и 0.

**Обучение и тестирование проектируемой нейронной сети.** Обучение нейронной сети для любой из выбранных структур будет выполняться на основе данных обучающей выборки, которая формируется в процедуре ConvertFileToData(). Структура нейронной сети определяется количеством скрытых слоев и числом нейронов в каждом из них и задается параметром hiddenLayers при создании нового объекта (net) класса brain.NeuralNetwork.

Процесс обучения нейронной сети запускает вызов метода train объекта net, куда наряду с массивом обучающей выборки передаются параметры обучения, считанные из основной HTML-формы. Все эти процессы можно реализовать в виде одной функции RunLearning(), которая будет привязана к событию нажатия кнопки «Run Learning» на основной форме:

```
// Процедура обучения и тестирования нейронной сети
function RunLearning () {
  // Преобразование данных окна textarea в массив данных
  var data = ConvertFileToData(document.all("txt").value);

  // Определение структуры нейронной
  var layers = document.all('layers').value;
  layers = layers.split(',').map(Number);
  var net = new brain.NeuralNetwork(
      { hiddenLayers: layers}
  );
  // Процедура обучения нейронной сети
  var t0 = new Date();
  var res = net.train(data.learn, {
      iterations:   Number(document.all('iter').value),
      learningRate: Number(document.all('lr').value),
      errorThresh:  Number(document.all('err').value),
      logPeriod:    Number(document.all('step').value),
      log: document.getElementById('islog').checked
  });
  var t1 = new Date();
  // Результат обучения
  log("\nВремя обучения: " + (t1-t0)/1000 + " сек");
  log("Итерация: "+res.iterations+"\t Ошибка: " +
       res.error.toFixed(5));
  // Тестирования сети на обучающей и тестовой выборках
  log("\nРезультат по обучающей выборке:");
  TestingNet(net, data.learn);
  log("\nРезультат по тестовой выборке:");
  TestingNet(net, data.test);
}
```

Следует отметить, что в этой функции, кроме задачи обучения нейронной сети, реализован еще и дополнительный сервис, связанный с оценкой времени обучения и количеством затраченных итераций для достижения заданной точности обучения. Также в этой функции предусмотрена возможность тестирования обученной нейронной сети. Это осуществляется путем вызова функции TestingNet(), которая выполняет вывод результатов тестирования в область тега <pre id="log"> основной HTML-страницы:

```
// Процедура тестирования сети на обучающей выборке
function TestingNet(nn, TestData) {
  log("№\tinput:\t\t\t output:\tвыход сети:");
  var err_count =0;
  for(var i=0; i < TestData.length; i++) {
      var out0 = TestData[i].output;
      var out = nn.run(TestData[i].input);
      let out_ = out.map(a => a.toFixed(2));
      // Вывод текущей строки результатов тестирования сети
      var txt = i +"\t[" + TestData[i].input + "]     \t ";
      txt += "[" + out0 + "] \t=>\t[" + out_ + "]";
      // Проверка ошибки предсказания для этой строки данных
      var delta = Number(document.all('delta').value);
      for(var j=0; j < out0.length; j++) {
          if (Math.abs(out[j]-out0[j]) > delta) {
              txt += "\terr";
              err_count ++ ;
              break
          }
      }
      log(txt)
  }
  txt = "Ошибок предсказания: "+err_count+" из "+TestData.length;
  txt += " ("+((err_count/TestData.length)*100).toFixed(1)+"%)";
  log(txt)
}
// Сервисная функция вывода в поле log
function log(msg) {
     document.getElementById("log").innerText += msg + '\n';
}
```

В этой функции из каждой строки набора данных (TestData) выделяется выходной массив (out0), вычисляется предсказанное сетью значение элементов выходного массива (out). Выполняется их округление до двух знаков после запятой (out_) и проверяется, что разность исходного и предсказанного значений находится в пределах заданной точности (delta). При превышении данного значения в строке для данного экземпляра данных дополнительно выводится текст «err».

**Результаты настройки проектируемой нейронной сети.** После того как в исходный шаблон HTML-страницы будет подключена библиотека BrainJS и добавлены описанные выше функции, появляется возможность ее использования для настройки и исследования нейронной сети по классификации Ирисов Фишера.

Как показали исследования, для построения нейросетевого классификатора Ирисов Фишера вполне достаточно всего одного скрытого слоя – как для структур нейронных сетей с тремя выходами, так и для структур сетей с одним выходом. При этом можно выявить множество различных структур, удовлетворяющих исходным требованиям, но отличающихся затраченными ресурсами на их обучение. В таблице приведена часть из полученных результатов. Их объединяет то, что во всех этих структурах нейронных сетей ошибка предсказания не превосходит индикатор класса, т. е. {[1,0,0],[0,1,0],[0,0,1]} или {0,0.5,1}, более чем на 0.1.

**Результаты исследования различных структур нейронных сетей**

**The results of the study of various structures of neural networks**

| Количество выходов | Количество нейронов | Learning Rate | Error Thresh | Количество итераций | Время, с | Выборка |
|---|---|---|---|---|---|---|
| 3 | 8 | 0.3 | 0.00005 | 173310 | 58.317 | 150 / 0 |
| 3 | 8 | 0.3 | 0.00005 | 69699 | 23.417 | 144 / 6 |
| 3 | 8 | 0.3 | 0.00005 | 18056 | 5.641 | 140 /10 |
| 3 | 5 | 0.3 | 0.00005 | 32373 | 7.373 | 150 / 0 |
| 3 | 5 | 0.3 | 0.00005 | 12940 | 2.964 | 144 / 6 |
| 3 | 5 | 0.3 | 0.00005 | 321130 | 74.280 | 140 /10 |
| 1 | 8 | 0.3 | 0.0001 | 35363 | 8.476 | 150 / 0 |
| 1 | 8 | 0.3 | 0.0001 | 44203 | 10.449 | 144 / 6 |
| 1 | 8 | 0.3 | 0.0001 | 45230 | 10.343 | 140 /10 |
| 1 | 5 | 0.3 | 0.0001 | 329923 | 54.606 | 150 / 0 |
| 1 | 5 | 0.3 | 0.0001 | 175088 | 28.412 | 144 / 6 |
| 1 | 5 | 0.3 | 0.0001 | 40730 | 6.337 | 140 /10 |
| 1 | 3 | 0.3 | 0.0001 | 320230 | 51.856 | 150 / 0 |
| 1 | 3 | 0.3 | 0.0001 | 340666 | 49.541 | 144 / 6 |
| 1 | 3 | 0.3 | 0.0001 | 335924 | 51.438 | 140 /10 |

В случае, когда предполагается частое переобучение сети на изменяемом наборе данных, предпочтительной будет структура сети, которая занимает минимальное время обучения. В том случае, когда требуется построить нейроклассификатор, работающий в среде браузера, да еще в сети Интернет, важнейшим является его простейшая структура, так как обучение нейронной сети можно провести заранее, еще до момента ее использования в классификаторе.

В этих условиях наиболее предпочтительной будет структура нейронной сети с одним нейроном выходного слоя и всего тремя нейронами скрытого слоя. Это требует более точного обучения выбранной структуры сети за счет увеличения числа итераций или уменьшения значения обобщенной ошибки.

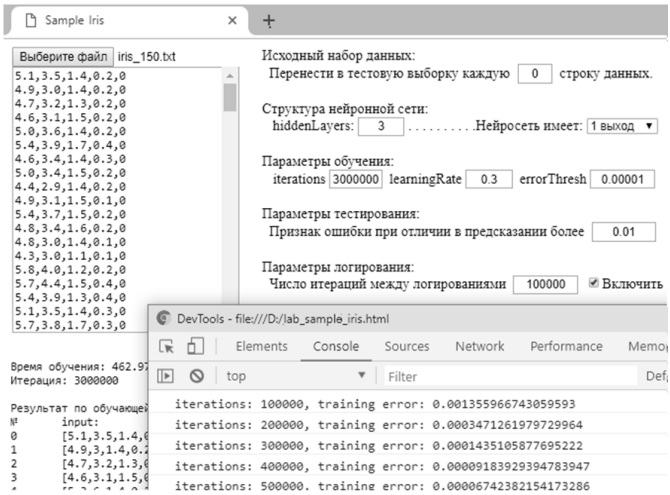


*Рис. 5*. Процесс обучения сети с трассировкой его результатов
*Fig. 5*. The process of learning the network with a trace of its results

При этом полезно следить за процессом обучения сети, выполняя трассировку результатов по точности ее обучения при различном количестве выполненных итераций (рис. 5). Для этого надо:

– на основной HTML-странице включить процесс логирования и задать число итераций между логированиями;

– затем, находясь в окне браузера, нажать клавишу F12 и в окне отладчика браузера перейти на вкладку Console;

– после чего вернуться в окно основной HTML-страницы и нажать кнопку «Run Learning».

Начнется процесс обучения сети с использованием алгоритма обратного распространения ошибки, а во вкладке Console отладчика браузера будут выводиться значения обобщенной ошибки для каждого выбранного шага логирования. Анализ характера изменения обобщенной ошибки от числа выполненных итераций (рис. 6) позволяет обоснованно изменять параметры обучения сети при повторных запусках сети на обучение.

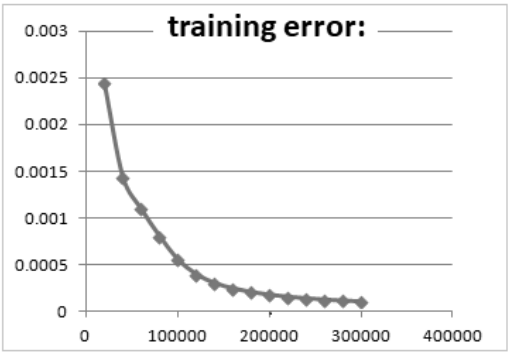


*Рис. 6*. Характер изменения обобщенной ошибки обучения сети
*Fig. 6.* The nature of the change in the generalized learning error network

Закончив процесс обучения нейронной сети и получив требуемую точность обучения, можно нажать на кнопку «View Weights», для того чтобы получить значения всех весов связи между нейронами обученной сети:

```
Веса к нейронам слоя 2
--- от нейронов слоя 1 :
3.9519503116607666   -7.611882209777832   34.0401611328125     -
34.445762634277344
8.038548469543457    26.918298721313477   -4.9816999435424805  -
78.78063201904297
0.8606927990913391    3.2729873657226562  -6.425778388977051    -
2.9703452587127686
--- от biase1 (смещение1):
-127.27687072753906  37.67887496948242     0.45325449109077454

Веса к нейронам слоя 3
--- от нейронов слоя 2 :
27.67657470703125   -20.064054489135742   -7.938845157623291
--- от biase2 (смещение2):
20.057218551635742
```

Значения этих весов могут быть использованы в нейроклассификаторе Ирисов Фишера, но имеет смысл оценить степень влияния округления этих значений на качество предсказания выходных значений нейронной сетью. Как показало исследование, округления весов только в двух наборах данных вызывают отклонение предсказания индекса класса в районе 5%, еще в четырех эта погрешность составит около 2%. Это позволяет сделать вывод о том, что модель нейроклассификатора Ирисов Фишера может иметь вид, представленный на рис. 7.

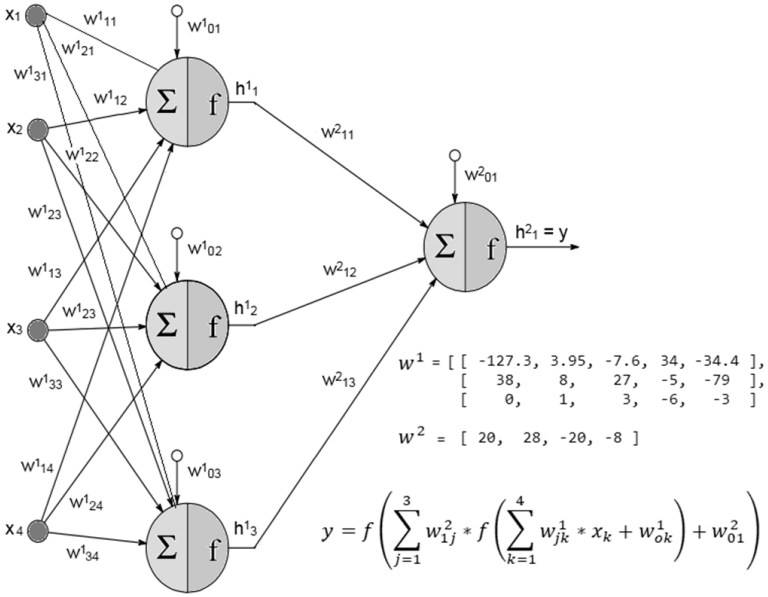


*Рис. 7.* Структура нейроклассификатора Ирисов Фишера
*Fig. 7.* The structure of the neuroclassifier Iris Fisher

Эта модель может быть легко реализована в виде HTML-страницы без использования каких-либо внешних библиотек или реализации сложных алгоритмов. Единственное, что потребуется сделать, так это, используя язык JavaScript, реализовать функцию универсального аппроксиматора, в котором функция *f*() представляет собой сигмоид. Вид такой HTML-страницы может быть аналогичен тому, что приведен на рис. 8.

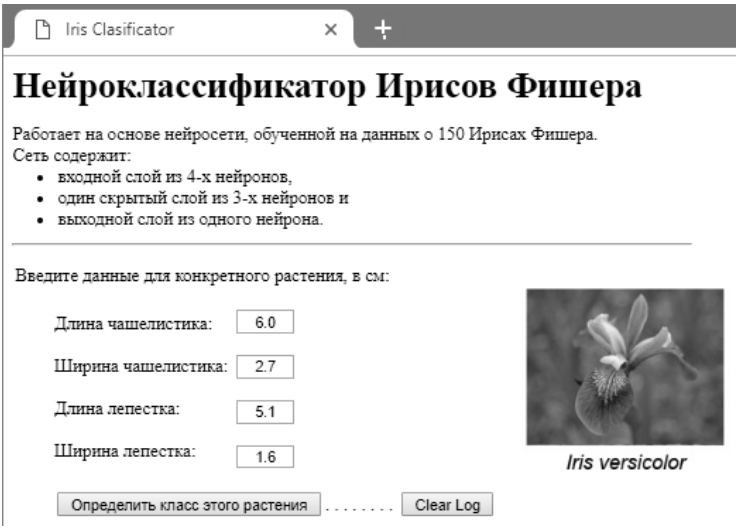


*Рис. 8*. Внешний вид страницы нейроклассификатора Ирисов Фишера
*Fig. 8*. The appearance of the neuroclassifier page of Iris Fisher

Для получения информации о классе конкретного растения достаточно в соответствующие поля ввести четыре измерения для этого растения, а затем нажать кнопку определения его класса, которая запустит процесс прямого распространения в реализованной нейронной сети.

*Выводы*. Проведенное исследование показало возможность применения нейросетевого подхода к классификации не сильно структурированных наборов данных без использования мощных нейросетевых пакетов, используя только возможности стандартных браузеров. Это позволяет использовать предложенный подход как в информационных системах мониторинга лесов и лесных пожаров [Заяц, Логачев, 2016], так и в клиент-серверных экспертных системах [Хабаров, Заяц, 2017; Хабаров, Шалаев, Васильев, 2016].

Исследование различных структур многослойных нейронных сетей, обучаемых на основе алгоритма обратного распространения ошибки, позволило выбрать для тестового набора данных структуру нейронной сети всего с одним скрытым слоем из трех нейронов. Это существенно упрощает реализацию классификатора Ирисов Фишера, позволяя его оформить в виде загружаемой с сервера HTML-страницы. Аналогичные классификато-

ры, построенные на базе предложенного подхода, могут быть включены в состав распределенных экспертных систем [Хабаров, Голубев, 2017].

Реализуемость предложенного подхода на всех этапах его применения подтверждается разработанными для исследования нейронных сетей алгоритмами и JavaScript-процедурами, которые подгружаются к интерфейсной HTML-странице.

**Библиографический список**

Ирисы Фишера. URL: https://ru. wikipedia.org/wiki/Ирисы_Фишера (дата обращения: 10.10.2018).

Machine Leaning Repository. URL: https://archive._ics.uci.edu/ml/ (дата обращения: 10.10.2018).

*Заяц А.М.*, *Логачев А.А*. Информационная система мониторинга лесов и лесных пожаров с использованием беспроводных сенсорных сетей // Известия Санкт-Петербургской лесотехнической академии. 2016, Вып. 216. С. 241–255.

*Хабаров С.П.*, *Заяц А.М.* Использование технологии websocket в клиент-серверных экспертных системах. // Леса России: политика, промышленность, наука, образование: матер. второй Междунар. науч.-техн. конф., 2017. С. 278–280.

*Хабаров С.П.*, *Шалаев Е. И.*, *Васильев С. П.* Реализация модуля логического вывода в составе информационной системы классификации растений // Информационные системы и технологии: теория и практика: сб. науч. тр. СПб.: СПбГЛТУ, 2016. № 8. С. 99–109.

*Хабаров С.П.*, *Голубев К.С.* Клиент-серверная экспертная система на основе технологии websocket // Информационные системы и технологии: теория и практика: сб. науч. тр. СПб.: СПбГЛТУ, 2017. № 9. С. 120–124.

*Голубев К.С.*, *Хабаров С.П.* Клиент-серверное приложение по идентификации хвойных пород по шишкам. [Свидетельство о государственной регистрации программ для ЭВМ № 2017617067 от 22.06.2017].

**References**

Iris flower data set. URL: https://en.wikipedia.org/wiki/ Iris_flower_data_set (accused October 10, 2018).

Machine Leaning Repository. URL: https://archive._ics.uci.edu/ml/ (accused October 10, 2018).

*Zayats A.M.*, *Logachev A.A.* The information system of monitoring forests and wildfires using the wireless sensor networks. *Izvestia Sankt-Peterburgskoj lesotehniceskoj akademii*, 2016, is. 216, pp. 241–254. (In Russ.)

*Khabarov S.P.*, *Zayats A.M.* Using websocket technology in client-server expert systems. *Forests of Russia: politics*, *industry*, *science*, *education Materials of the second International Scientific and Technical Conference*, 2017, pp. 278–280. (In Russ.)

*Khabarov S.P.*, *Shalaev E.I.*, *Vasiliev S.P.* The implementation of the inference module as part of the plant classification information system. *Information systems and technologies: theory and practice: a collection of scientific papers*. St. Petersburg: SPbGLTU, 2016, no. 8, pp. 99–109. (In Russ.)

*Khabarov S.P.*, *Golubev K.S.* Client-server expert system based on websocket technology *Information systems and technologies: theory and practice: a collection of scientific papers*. St. Petersburg: SPbGLTU, 2017, no. 9, pp. 120–124. (In Russ.)

*Golubev K.S.*, *Khabarov S.P.* Client-server application for the identification of coniferous species by cones. Certificate of state registration of computer programs no. 2017617067 of 06/22/2017. (In Russ.)

*Материал поступил в редакцию 15.10.2018 г.*

---

**Заяц А.М., Хабаров С.П.** Построение нейронной сети классификации ирисов Фишера на базе JavaScript // Известия Санкт-Петербургской лесотехнической академии. 2019. Вып. 226. С. 233–247. DOI: 10.21266/2079-4304.2019.226.233-247

Рассматривается процедура выбора структуры и параметров нейронной сети для классификации набора данных, известного как Ирисы Фишера, который включает в себя данные о 150 экземплярах растений трех различных видов. Предложен подход к решению данной задачи без использования дополнительных программных средств и мощных нейросетевых пакетов с использованием только средств стандартного браузера ОС. Это потребовало реализации ряда процедур на JavaScript с их подгрузкой в разработанную интерфейсную HTML-страницу. Исследование большого числа различных структур многослойных нейронных сетей, обучаемых на основе алгоритма обратного распространения ошибки, позволило выбрать для тестового набора данных структуру нейронной сети всего с одним скрытым слоем из трех нейронов. Это существенно упрощает реализацию классификатора Ирисов Фишера, позволяя его оформить в виде загружаемой с сервера HTML-страницы.

Ключевые слова: нейронные сети, классификация, экспертные системы, web технология.

**Zayats A.M., Khabarov S.P.** Construction of neural network classification of Fisher irises based on JavaSCript. *Izvestia Sankt-Peterburgskoj Lesotehniceskoj Akademii*, 2019, is. 226, pp. 233–247 (in Russian with English summary). DOI: 10.21266/2079-4304.2019.226.233-247

The procedure for selecting the structure and parameters of the neural network for the classification of a data set known as Iris Fisher, which includes data on 150 plant specimens of three different species, is considered. An approach to solving this problem without using additional software and powerful neural network packages

using only the tools of the standard OS browser is proposed. This required the implementation of a number of JavaScript procedures with their loading into the developed HTML interface page. The study of a large number of different structures of multilayer neural networks, trained on the basis of the back-propagation error algorithm, made it possible to choose the structure of a neural network with only one hidden layer of three neurons for a test dataset. This greatly simplifies the implementation of the Fisher Iris classifier, allowing it to be formatted as an HTML page downloaded from the server.

K e y w o r d s : neural networks, classification, expert systems, web technology.

---

**ЗАЯЦ Анатолий Моисеевич** – заведующий кафедрой информационных систем и технологий Санкт-Петербургского государственного лесотехнического университета имени С.М. Кирова, кандидат технических наук, профессор.

194021, Институтский пер., д. 5, Санкт-Петербург, Россия. E-mail: zamfta@yandex.ru

**ZAYATS Anatoliy M.** – PhD (Technical), Professor, Head of department of Information Systems and Technology of St.Petersburg State Forest Technical University.

194021. Institute per. 5. St. Petersburg. Russia. E-mail: zamfta@yandex.ru

**ХАБАРОВ Сергей Петрович** – доцент кафедры информационных систем и технологий Санкт-Петербургского государственного лесотехнического университета имени С.М. Кирова, кандидат технических наук.

194021, Институтский пер., д. 5, Санкт-Петербург, Россия. E-mail: Serg.Habarov@mail.ru

**KHABAROV Sergey P.** – PhD (Technical), Associate Professor, Associate Professor of department of Information Systems and Technology of St. Petersburg State Forest Technical University.

194021. Institute per. 5. St. Petersburg. Russia. E-mail: Serg.Habarov @mail.ru