目录

简介

本课程的实验都应用Intellij完成。任课老师应当已经提供了Intellij安装设置指南，或者您已经有一个预装了Intellij并配置完毕的虚拟机。

Maven

您将用Maven管理DeepLearning4J项目，所以我们建议您先了解Maven的基础知识。

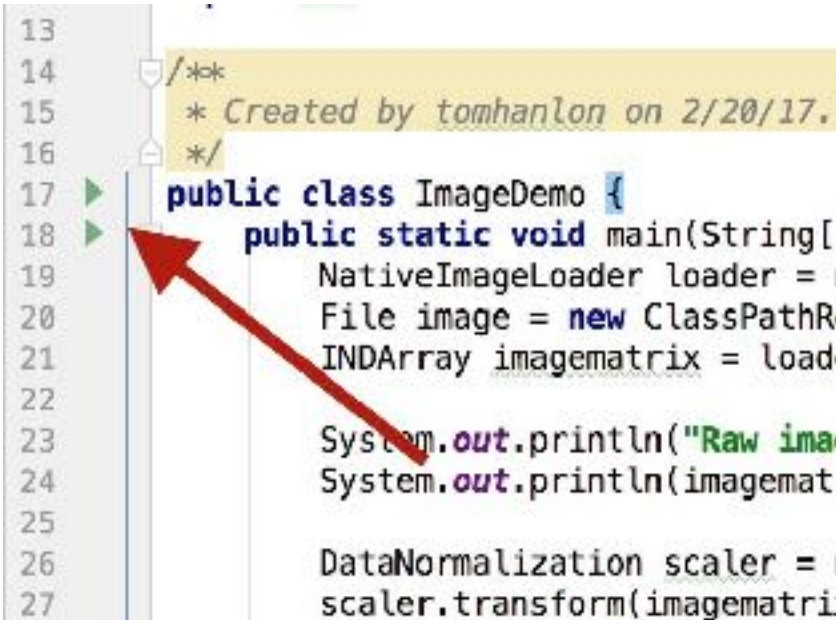
Maven采用一个pom.xml文件来管理依赖项。

如需寻找实验项目的pom.xml文件，请注意有两个不同的层级：dl4j-training/pom.xml和dl4j-training/dl4j-labs/pom.xml

运行和停止运行代码

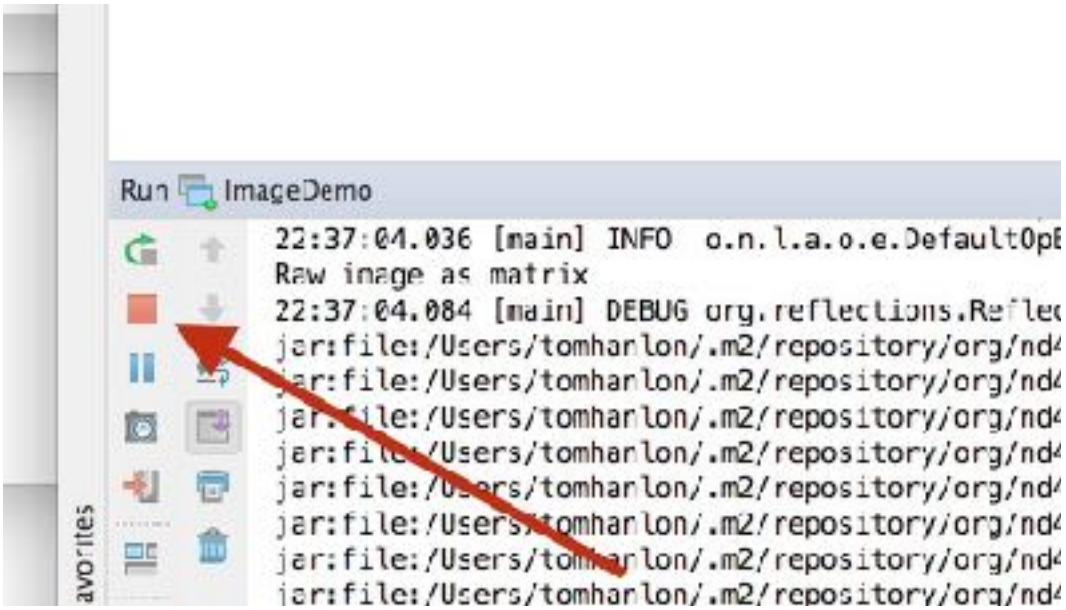
在编辑或创建Java类时，您可随时点击主类旁的绿色箭头运行代码。

- 运行代码



- 停止代码运行

点击红色按钮，可以停止代码运行



常见问题

任何一处具体代码问题都会影响到整个项目。如果项目无法正常编译，您将无法运行其他能够正常编译的类。运行一个类会让整个项目开始编译，如果另一个有缺陷的类编译失败，其他正常的类也会无法运行。

解决方法：如果您有未完成的类或其他任何无法编译的内容，请加以修正，或将整段相关代码设为注释

1. 太困难

请在万不得已的情况下再参考答案目录。如果实验难度太大，请记住您可以尽自己所能编写代码，然后将Java类以工作状态保存，或者设为注释，确保项目至少能正常编译，然后继续推进。

2. 太简单

您完全可以创建一个空白项目，从零开始编写。如果您的问题与课程内容或课题相关，任课老师会很乐意予以回答

简易神经网络

在本实验中，您将探索一个非常简单的神经网络。这个神经网络将包括：

- 单一输入
- 单一预期数值输出
- 一个只有单一神经元的隐藏层

该神经网络将接受输入0.5，预期输出为0.8。

该神经网络将进行100次迭代的训练，目标是改进分值，亦即输出与0.8的接近程度。

本实验目的

- 让用户熟悉Deeplearning4J的代码。

第1步

- 打开IntelliJ 打开IntelliJ，进入Labs目录

第2步

- 打开SimplestNetwork类，点击SimplestNetwork.java，在编辑器中打开该Java类

第3步

- 查看Java代码

请注意顶部的参数设置。

```
int seed = 123;
```

这是一个硬编码的随机种子，确保得到可重复的结果。神经网络会设定随机的初始权重。

如果想要得到可重复的结果，可以使用预配置的种子。

如果改变种子，网络表现也会出现微小变化。

```
int numInputs = 1;
int numOutputs = 1;
```

为何选用Xavier初始化

简而言之，它能帮助信号深入神经网络。

如果网络的初始权重太小，信号每通过一个层就会有所衰减，最终因为变得太弱而失去作用。如果网络的初始权重太大，信号每通过一个层就会有所放大，最终因为变得太强而失去作用。Xavier初始化可以确保权重大小“刚刚好”，让信号通过许多层之后仍保持在合理的区间内。

如需进一步细究，您需要掌握一定的统计学知识，尤其要了解随机分布及其方差。

随机梯度下降

请注意，本实验中使用的优化算法是随机梯度下降。

在神经网络多年的研究过程中，如何更新大型网络的权重以减少误差（得到更好结果）始终是一项重大挑战。数值计算尤其困难。SGD通过某种形式的随机选择来应对这一挑战，您可进一步深入研究。

更新器：Nesterovs

此处不探讨过多细节。请注意，在更加复杂的网络中，动量可能是一项需要调试的超参数。本示例中的问题是线性的，但如果遇到有可能陷入局部最小值的更复杂的图，那么动量有助于实现突破。

第0层激活函数：tanh

一个层的激活函数决定了该层向与之相连的神经元发出怎样的信号。选择包括：sigmoid，relu, leaky relu, tanh

tanh，与sigmoid相似，输出为-1至+1，取决于x的值。

第1层：这是我们的输出层。

请注意此处的激活函数是恒等函数。

这决定了输出是线性的，即一定范围内的数值，.1、.2、.3等，而非0或1，也不是类别A、B、C

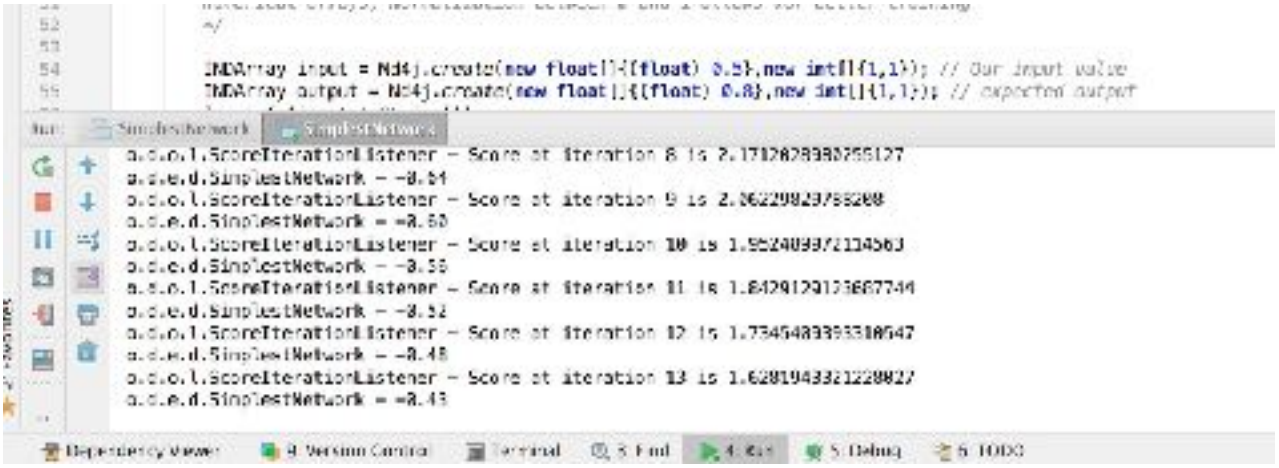
运行代码

在这一步中，您将运行代码。

代码执行后，将生成一个可以通过网页浏览器访问的UI。输出结果也会在运行时显示于Intellij底部的输出窗口。

点击这一绿色箭头，执行代码

类运行时，在控制台中查看输出

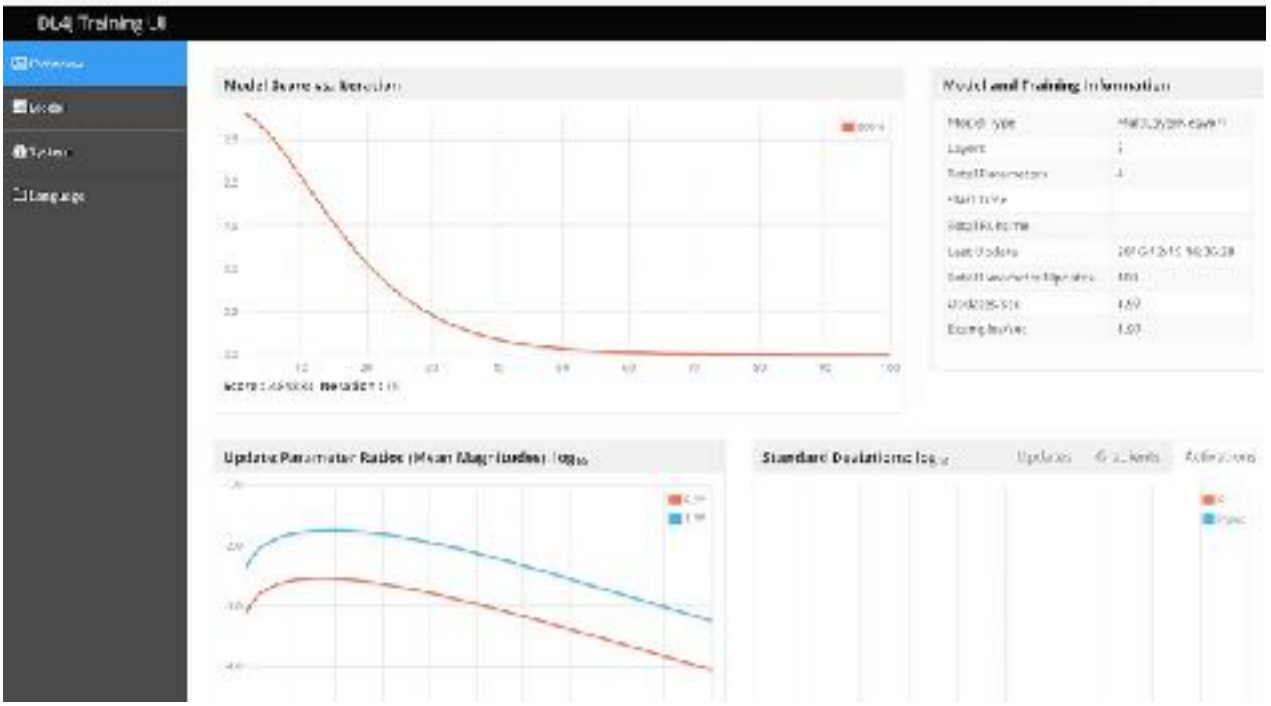


查看UI

代码执行并生成UI后，控制台输出中会显示一行URL地址

在浏览器中打开该URL地址 您

应当看到：



输出说明

控制台输出。

以下代码块启动训练流程。

```
   for( int i=0; i<nEpochs; i++ ){

          model.fit(input,output);
          INDArray predicted = model.output(input);
          log.info("predicted: " + predicted.toString());
}
```

什么是迭代次数？

一次迭代（一个epoch）指完整遍历训练数据集一次，总计遍历次数即为迭代次数（epoch数）。本示例仅有单个输入，但在实践中，一次迭代可能覆盖了数千条文本评论、数十万幅图像、来自记录文件的数百万行记录等。

什么是Model.fit？

模型从此处开始训练。摄取数据，指定随机权重，按预期输出评估实际输出，调整权重以降低误差。

预期输出

以下这段代码

```
INDArray predicted = model.output(input);
log.info("predicted: " + predicted.toString());
```

将在控制台输出中生成以下几行内容。

```
o.d.e.d.SimplestNetwork - -0.87
o.d.e.d.SimplestNetwork - -0.85
```

“正确的”输出，即“预期的”输出应是0.80，您会看到网络在训练过程中不断逼近这一目标。

控制台输出中的这一行

```
o.d.o.l.ScoreIterationListener - Score at iteration 1 is 2.775866985321045
```

由以下这行代码生成

```
model.setListeners(new StatsListener(statsStorage),new ScoreIterationListener(1));
```

修改部分参数，了解这对训练过程会有怎样的影响

在这一步中，您将修改部分参数，了解这对训练过程会有怎样的影响。

请注意，但凡需要重新运行此代码，您都必须终止此前正在运行的进程。服务UI的Web服务器会占用一个套接字接口，第二个示例将尝试连接同一个套接字，但会失败并报错。

请点击右上方的红色方块，终止正在运行的进程。

一些参数可供您调试。

在修改参数前，请记录当前的性能表现。需要经过多少次迭代才到达距离目标.05的范围内？100次迭代后距离目标有多近？我的网络在迭代100次后为.78，第80次迭代时到达了.75

更改后仍能获得合理结果的设置

隐藏节点

- 隐藏节点的数量

  隐藏节点更多，则尝试逼近正确解的次数越多，随机权重更多，有可能加快训练速度

迭代次数

- 迭代次数

  如果网络在向目标收敛，那么增加迭代次数，网络最终应能到达目标。

学习速率

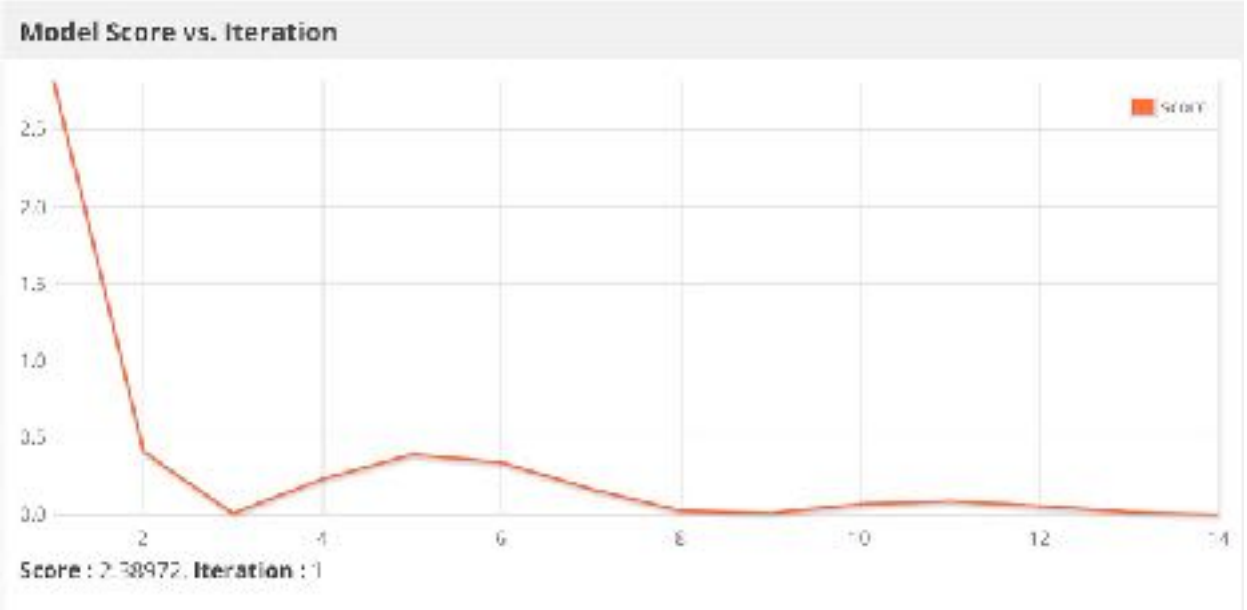
学习速率决定了依据误差对权重进行调整的程度。学习速率的范围可以是 0.1~0.001

```
double learningRate = 0.001;
```

或许可改为……

```
double learningRate = 0.01;
```

请注意，过大的学习速率可能导致网络错过目标，无法收敛至最优值。



实验思考题

1. 神经网络中有哪些参数可能需要调整？
2. 一个神经元的输出取决于：

    - 输入
    - 激活函数
    - 权重
    - 偏差
    - 以上全部

答案

1. 最常需要调整的参数是学习速率。(0.1~0.001)
2. 以上全部。一个神经元的输出是将（输入之和 * 权重 + 偏差）输入激活函数后所得的结果

DataVec实验

用DataVec摄取一个CSV数据集

在本实验中，您将从一个CSV文件导入数据，转换为适合神经网络处理的格式。

本实验目的

- DataVec简介

第1步

- 打开IntelliJ 打开IntelliJ，进入Labs目录

第2步

- 打开DataVecLab类

点击DataVecLab.java，在编辑器中打开该Java类

第3步

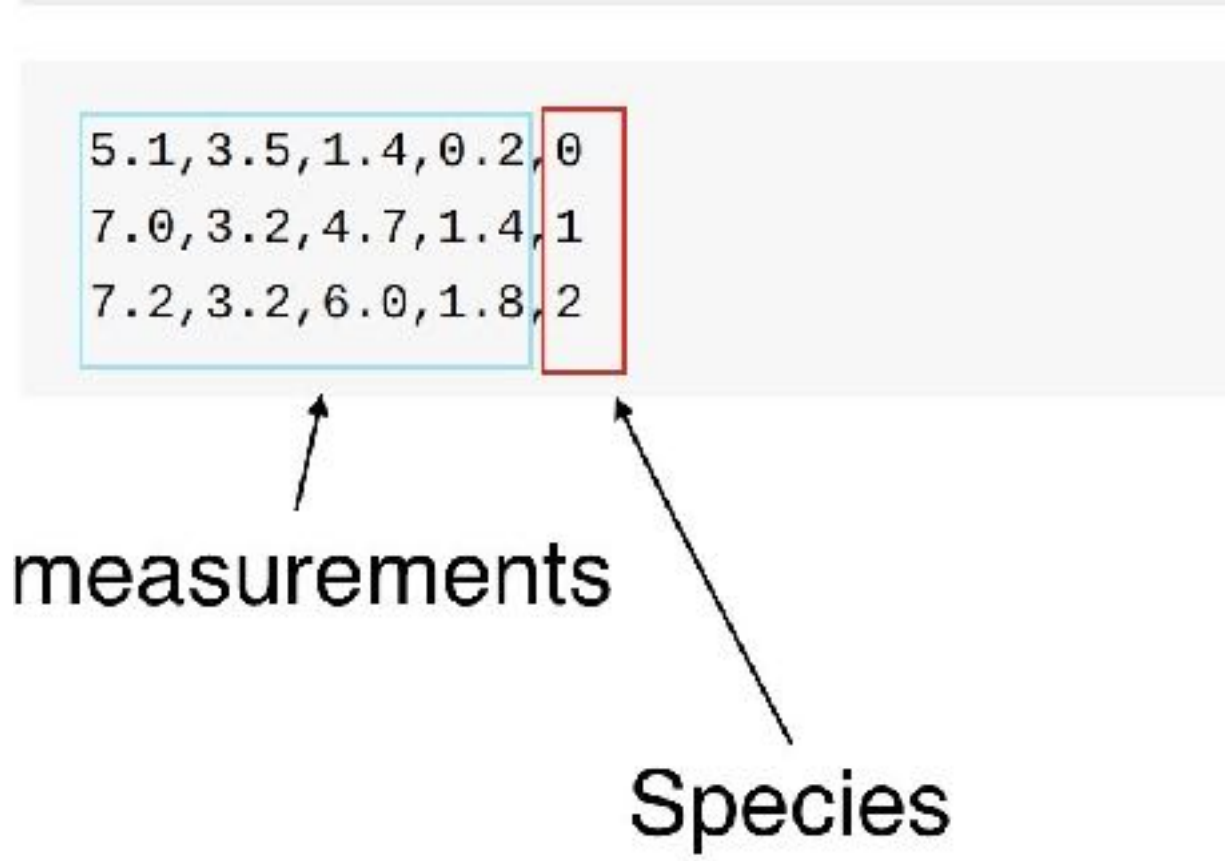
- 查看Java代码

本例中，神经网络已经构建完毕。本实验的目的是用DataVec完成数据ETL流程。

- 理解难点

Iris.txt文件包含150条鸢尾花测量数据，共有3个鸢尾花品种：Iris Setosa（山鸢尾）、Iris Virginica（弗吉尼亚鸢尾）、Iris versicolor（杂色鸢尾）。测量数据为花瓣长度、花瓣宽度、萼片长度、萼片宽度。

数据记录包括一个表示品种的数值：0=> Setosa，1 => Versicolor，2=> Virginica



了解所需步骤

1. 读取文件
2. 解析数据条目
3. 区分标签字段和测量数据
4. 创建DataSet对象，将数据输入神经网络。

将要使用的DataVec类。

https://deeplearning4j.org/datavecdoc/org/datavec/api/records/reader/RecordReader.html

https://deeplearning4j.org/datavecdoc/org/datavec/api/records/reader/impl/csv/CSVRecordReader.html

将要使用的Deeplearning4J类

https://deeplearning4j.org/doc/org/deeplearning4j/datasets/datavec/RecordReaderDataSetIterator.html

完整的DataVec JavaDoc https://deeplearning4j.org/datavecdoc/

完整的Deeplearning4J JavaDoc。https://deeplearning4j.org/doc/

高级用户可以直接打开存根进行操作。其他人请继续按下列指示操作

设置部分参数

CSVRecordReader可以忽略一个文件的头几行。因为文件可能有一些标头信息或注释。

请查看Iris.txt，确认文件没有标头。

CSVRecordReader可以设置不同的数据记录分隔符。确认文件分隔符为逗号。

** 请注意，数据质量差是很常见的问题。在这个理想化的实验环境中，您可以完全信赖数据；但在现实世界中，我总是会运行一些校验代码，至少确保每行都有相同数量的逗号。

确认文件没有标头且分隔符为逗号后，请将以下代码添加至存根程序。

```
int numLinesToSkip = 0;
String delimiter = ",";
```

创建RecordReader（记录读取器）

请将这行代码添加至代码存根

```
RecordReader recordReader = new CSVRecordReader(numLinesToSkip,delimiter);
```

初始化RecordReader并向其传递文件

出于便携性考虑，文件置于resources文件夹中。文件可作为ClassPathResource读取。或者，您也可获取文件具体路径并由此访问文件。

使用这段代码可以很方便地访问一个类路径上的文件： `new ClassPathResource(fileName).getFile()`

初始化RecordReader并向其传递一个FileSplit。

FileSplit可以指向一个目录，然后RecordReader可以读取该目录下的所有文件，当然本例中指向的是单个文件。

```
recordReader.initialize(new FileSplit(new ClassPathResource("iris.txt").getFile()));
```

为您在下一步中需要创建的DataSetIterator设置参数

```
int labelIndex = 4;
//CSV文件iris.txt每行有5个值：
//4个特征输入，然后是1个整数标签（类别）索引。
//标签是每一行的第5个值（索引4）

int numClasses = 3;
//鸢尾花数据集共有3个类别（鸢尾花品种）。
//类别为整数值0、1、2

int batchSize = 150;
//鸢尾花数据集：共有150个样例。
//我们将所有数据加载至一个DataSet对象
//（大型数据集不建议这样操作）
```

创建DataSetIterator（数据集迭代器）

RecordReader向迭代器传递一系列Writables（可写对象）。Writables是受Hadoop Writables启发而开发的一种高效序列化方法。神经网络的输出必须是一个数值数组。为此要使用DataSetIterator。

一个DataSet将包含一个特征的INDArray，以及一个标签的INDArray。请为您的DataVecLab类添加以下代码。

```
DataSetIterator iterator = new RecordReaderDataSetIterator(recordReader,batchSize, labelIndex,numClasses);
DataSet allData = iterator.next();
```

打乱数据

神经网络的训练通常会以微批次为单位进行。假设微批次大小为10。10条数据记录通过网络后，计算实际输出与预期值之间的误差，再更新网络权重以降低误差。现在请看我们的鸢尾花数据集：先是一个品种的数据，然后又完全是另一个品种的数据。如果一个微批次中的数据完全偏向某一个类别，那么网络的训练结果也会先偏向一个方向，再偏向另一个方向。这是不好的，所以请打乱数据。

请将以下代码添加至类。

```
allData.shuffle();
```

划分训练集和测试集

在有监督学习中，网络先用一部分数据进行训练，然后再用一些未曾出现过的数据来进行测试。请将数据分为训练集和测试集。

```
SplitTestAndTrain testAndTrain = allData.splitTestAndTrain(0.65); //65%的数据用于训练
DataSet trainingData = testAndTrain.getTrain();
DataSet testData = testAndTrain.getTest();
```

DataSet的标准化

DataSet的标准化

当数据都必须进行标准化时，Deeplearning4J提供了所需的工具。

http://nd4j.org/doc/org/nd4j/linalg/dataset/api/preprocessor/DataNormalization.html

用LSTM预测序列数据

在本实验中，您将使用一行自己选择的句子输入RNN，进行简单的序列数据预测。所预测的单句将被输入于RNN, 加以训练。

输入数据

```
*This example trains a RNN. Look for lab steps below. Uncomment to proceed."
```

结果示范

```
hhis example trains   RNN. Look for laab sepps below.  Uncmmmenttto proceed
hhis example trains   RNN. Look for abbsteps blow. Uncommenttto proceed.***
hhis example trains   NNN Lookkffrr lab steps below.  Uncmmmnntto proceed.*
hhis example trains   NN. Look for lab sepps below. Uncommenttto proceed.**
hhis example trains    NNN. Look for lab  seps below. Uncommenttto proceed*
hhis example trains  RNN. Look for ab steps below. Uncommenttto proceed.***
```

请照步骤取消代码的注释，以进行试验。

用LSTM实现天气预报

本实验中，您将接触到实现LSTM循环神经网络的Java类。LSTM（长短期记忆）循环网络是以Graves的研究为基础发展起来的。与简单的多层感知器不同，LSTM的计算节点能够识别时间序列数据的模式。此类网络有许多具体的时间序列应用，本实验将一则天气预报视为字符序列，让网络预测下一个出现的字符。

本实验目的

- 让用户熟悉LSTM网络的配置与使用方法。

实验概述

本实验将训练一个神经网络如何生成天气预报。

天气预报训练数据取自美国国家气象局的真实天气预报内容。

天气预报内容样本如下。

WVZ015-1717
00-
KANAWHA-
INCLUDING THE CITIES OF...CHARLESTON...SOUTH
CHARLESTON...SAINT ALBANS
932 PM EST FRI DEC 16 2016

.REST OF TONIGHT...CLOUDY.RAIN LIKELY.NOT AS COLD WITH LOWS AROUND
30.TEMPERATURE RISING INTO THE LOWER 40S.SOUTH WINDS 10 TO 15 MPH
WITH GUSTS UP TO 25 MPH.CHANCE OF RAIN 70 PERCENT.
.SATURDAY...CLOUDY.RAIN LIKELY IN THE MORNING...THEN A CHANCE OF SHOWERS
IN THE AFTERNOON.MUCH WARMER WITH HIGHS IN THE LOWER 60S.SOUTHWEST
WINDS 10 TO 15 MPH WITH GUSTS UP TO 25 MPH.CHANCE OF RAIN
70 PERCENT.
.SATURDAY NIGHT...SHOWERS...MAINLY AFTER MIDNIGHT.LOWS IN THE UPPER
30S.SOUTHWEST WINDS 10 TO 15 MPH.CHANCE OF RAIN NEAR 100 PERCENT.
.SUNDAY...RAIN SHOWERS IN THE MORNING...THEN SNOW SHOWERS LIKELY IN
THE AFTERNOON.MUCH COOLER WITH HIGHS IN THE LOWER
40S.TEMPERATURE FALLING TO AROUND 30 IN THE AFTERNOON.WEST WINDS 5
TO 10 MPH.CHANCE OF PRECIPITATION 90 PERCENT.
.SUNDAY NIGHT...MOSTLY CLOUDY.A SLIGHT CHANCE OF SNOW SHOWERS IN
THE EVENING.MUCH COLDER WITH LOWS IN THE LOWER 20S.NORTHWEST
WINDS
5 TO 10 MPH.CHANCE OF SNOW 20 PERCENT.
.MONDAY AND MONDAY NIGHT...PARTLY CLOUDY.HIGHS IN THE LOWER
30S.LOWS IN THE LOWER 20S.
.TUESDAY THROUGH WEDNESDAY...MOSTLY CLEAR.HIGHS IN THE MID
40S.LOWS IN THE MID 20S.
.WEDNESDAY NIGHT...PARTLY CLOUDY IN THE EVENING...THEN BECOMING
MOSTLY CLOUDY.LOWS IN THE MID 30S.
.THURSDAY AND THURSDAY NIGHT...MOSTLY CLOUDY.A CHANCE OF RAIN
SHOWERS.HIGHS IN THE MID 40S.LOWS IN THE LOWER 30S.CHANCE OF RAIN
40 PERCENT.
.FRIDAY...MOSTLY SUNNY.HIGHS IN THE MID 40S.

$$

值得注意的细节

上述文本并不能算是规范的英语。一行开头有“.”似乎表示短语开始，一行开头没有“.”表示该行是上一行的继续，而“$$”则表示预报结束。

网络接收数据的方式

数据将以单个字符组成的序列的形式输入。网络将尝试预测下一个字符，评估结果，更新权重，如此反复。

网络将开始学习说出像天气预报一样的话。输出将从这个序列开始

```
WVZ006-171700-
CABELL-
INCLUDING THE CITY OF...HUNTINGTON 932
PM EST FRI DEC 16 2016
```

余下内容将由神经网络生成。以下是一个网络初次尝试的样本。

.TODAY...MOSTLY CLOUDY.A CHANCE OF RAIN.NOT0 10 TO 16.NOUTHWEST WINDS 30 TOR 10
0.LOWS AROUND 30.WEST WINDS AROUND
5 MPH.CHANCE OF RAIN 60 PERCENT.
.SATURDAY...CLOUDY.HIGHS IN THE LOWER 30S.CHANCE OF PRECIPITATION 60 PERCENT.
.SUNDAY NIGHT...MOSTLY CLOUDY IN THE MORNING...THEN
BECOMING MOSTLY
CLOURNG WITH A 40 PERCENT CHANCE OF SHOW SHOWERS.COLD WITH HIGHS IN THE LOWER 40S
.
.FRIDAY NIGHT...MOSTLY CLOUDY.A SLIGHT CHANCE OF RAIN
IN THE AFTERNOON.
COOL WITH HIGHS IN THE MID 40S.

WVZ028-020
150-
WROING-
INCLUDING THE CITY OF...GASTAY...CLEAN...BESPARAY HIGE 5GHT A
COOL WITH HIGHS IN THE UPPER 40S.CHANCE OF SNOW 30 PERCENT.
.SUNDAY NIGHT...A CHANCE OF SNOW.NOT AS COOL NH T A CHANCE OF SNOW
SHOWERS.COLD WITH LOWS IN THE UPPENNOG...TUEN ANOVVEATUR HISH A MOND
10.NORTHWEST WINDS AROUND
5 MPM.
CHANCE OF R0E
C0MPERATURE IN THE MID
AST.
.FRIDAY...CLOUDY.A SLIGHT CHANCE OF SNOW AND RAIN THIS LOWS IN THE AFTERNOON.MUC
H COOLER.NEAR STE LID 30 PM.
.THURSDAY...SUNNY.COLDER.NEAR STEADY TEMPERATURE IN THE
LOWER 30S.
.SUNDAY...MOSTLY SUNNY.HIGHS IN THE MIDNIGHT.COLD WITH HIGHS IN THE LOWER 20S.
.SUNDAY NIGHT...RAIN.NOT AS COOL WITH LOWS IN THE MPR 30T.CHANCE OF PREC

步骤

第1步

- 打开IntelliJ 打开IntelliJ，进入Labs目录

第2步

- 打开GravesLSTMCharModellingWeatherForecasts
- 在编辑器中打开该Java类
- 请照步骤取消代码的注释，以进行试验

第3步

- 查看Java代码, 请注意顶部的参数设置。各项参数说明已附上。

```
int lstmLayerSize = 200;                    //每个GravesLSTM层里的单元数
int miniBatchSize = 32;                     //训练时使用的微批次大小
int exampleLength = 4000;                   //每个训练用样例序列的长度。序列长度当然可以增加
int tbpttLength = 50;                       //截断式沿时间反向传播的长度，即每50个字符进行一次参数更新
int epochs = 1;                             //总迭代次数，整数
int generateSamplesEveryNMinibatches = 30;
//网络以怎样的频率生成文本样本?

int nSamplesToGenerate = 4;                 //每次训练迭代后生成的样本数
int nCharactersToSample = 1200;             //生成的每个样本的长度
// String generationInitialization = null;  //可选的字符初始化；如果为null则使用随机字符
String generationInitialization = "WVZ006-171700-\n" + "CABELL-
      \n" + "INCLUDING THE CITY OF...HUNTINGTON\n" + “932
      PM EST FRI DEC 16 2016”;
// 以上文本用于为LSTM“打底子”，让网络续写/完成初始的字符序列。

int seedNumber = 12345;                     //统一随机种子，确保结果一致性
```

为何选用Xavier初始化

简而言之，它能帮助信号深入神经网络。

如果网络的初始权重太小，信号每通过一个层就会有所衰减，最终因为变得太弱而失去作用。如果网络的初始权重太大，信号每通过一个层就会有所放大，最终因为变得太强而失去作用。Xavier初始化可以确保权重大小“刚刚好”，让信号通过许多层之后仍保持在合理的区间内。

如需进一步细究，您需要掌握一定的统计学知识，尤其要了解随机分布及其方差。

随机梯度下降

请注意，此处的优化算法是随机梯度下降。

在神经网络多年的研究过程中，如何更新大型网络的权重以减少误差（得到更好结果）始终是一项重大挑战。数值计算尤其困难。SGD通过某种形式的随机选择来应对这一挑战，您可进一步深入研究。

更新器：RMSPROP

此处不探讨过多细节。请注意，在更加复杂的网络中，动量可能是一项需要调试的超参数。本示例中的问题是线性的，但如果遇到有可能陷入局部最小值的更复杂的图，那么动量有助于实现突破。这里应讲多深？

第0层与第1层激活函数：Tanh

一个层的激活函数决定了该层向与之相连的神经元发出怎样的信号。选择包括：sigmoid，平滑曲线，输出为0至1，随x变化。

Tanh，与sigmoid相似，输出为-1至+1，取决于x的值。

第3层：这是我们的输出层。

请注意此处的激活函数是Softmax函数，价值函数是MCXent函数。

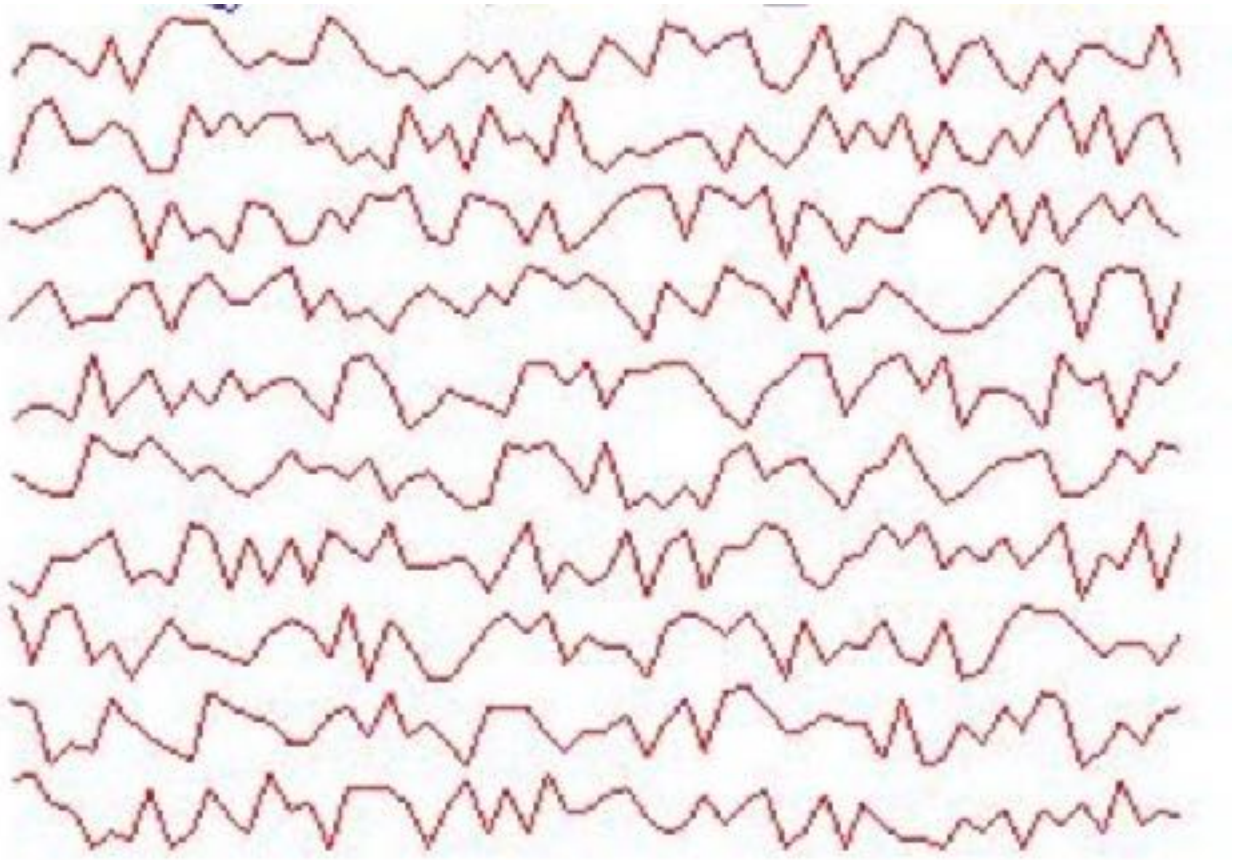
请照步骤取消代码的注释，以进行试验。

LSTM文本生成实验

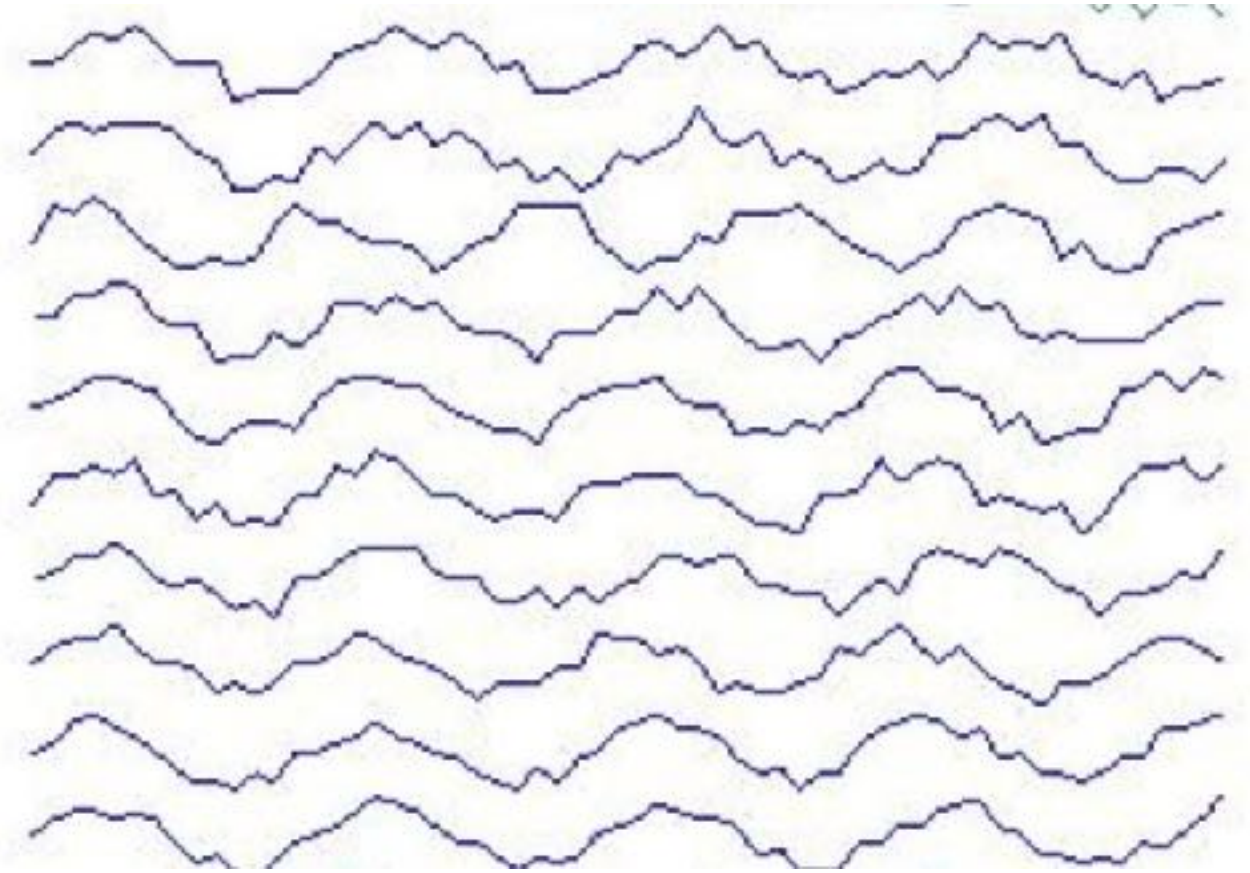
本实验中，您将用一个LSTM循环神经网络来将序列数据分为6种不同的类别。

这些类别是：

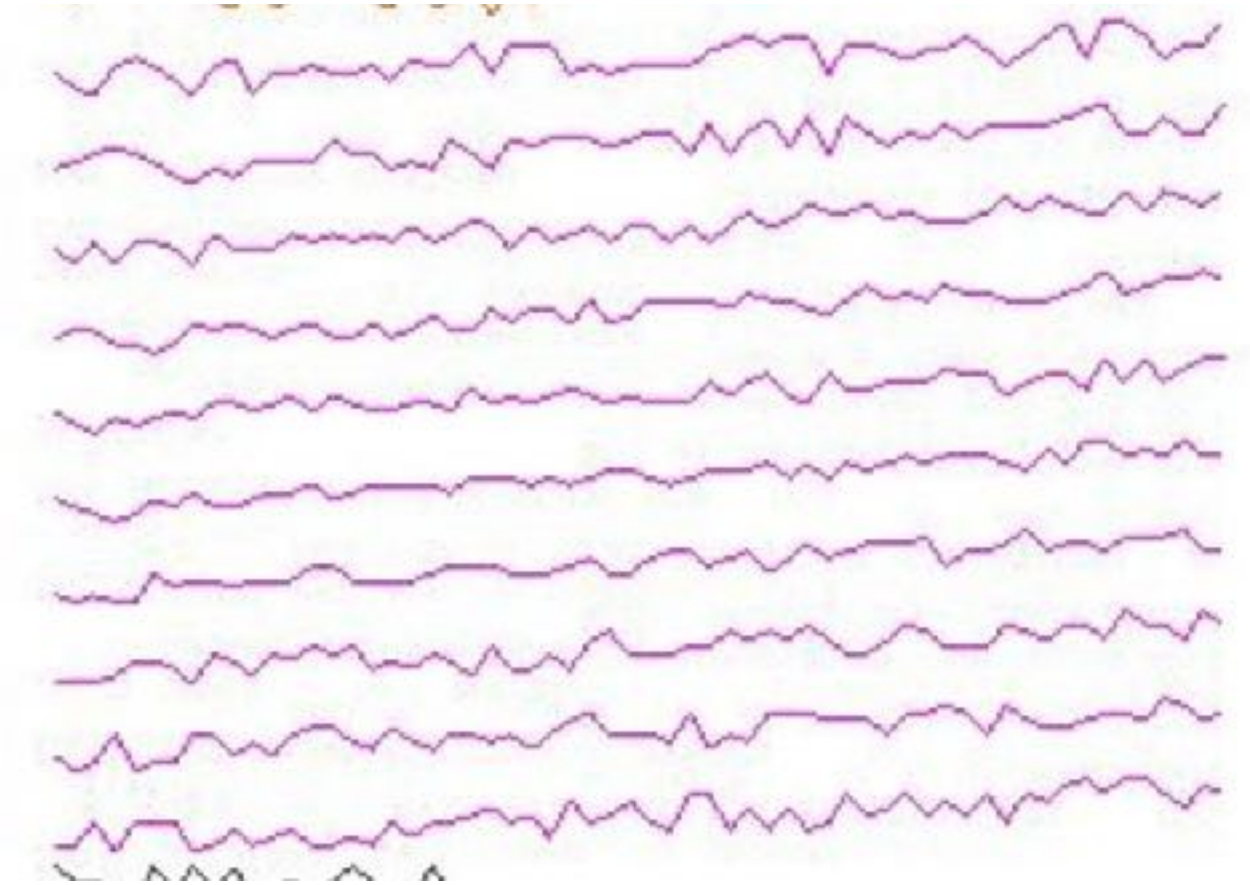
普通



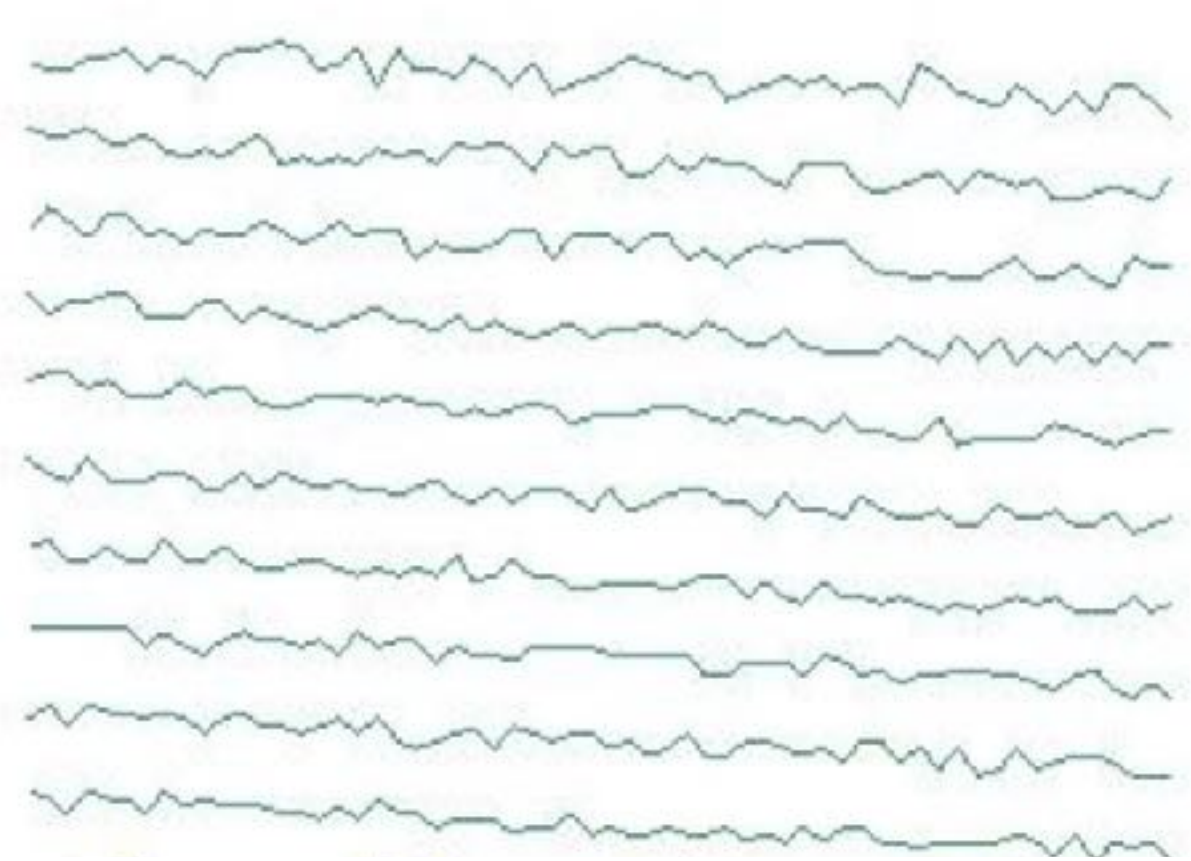
循环



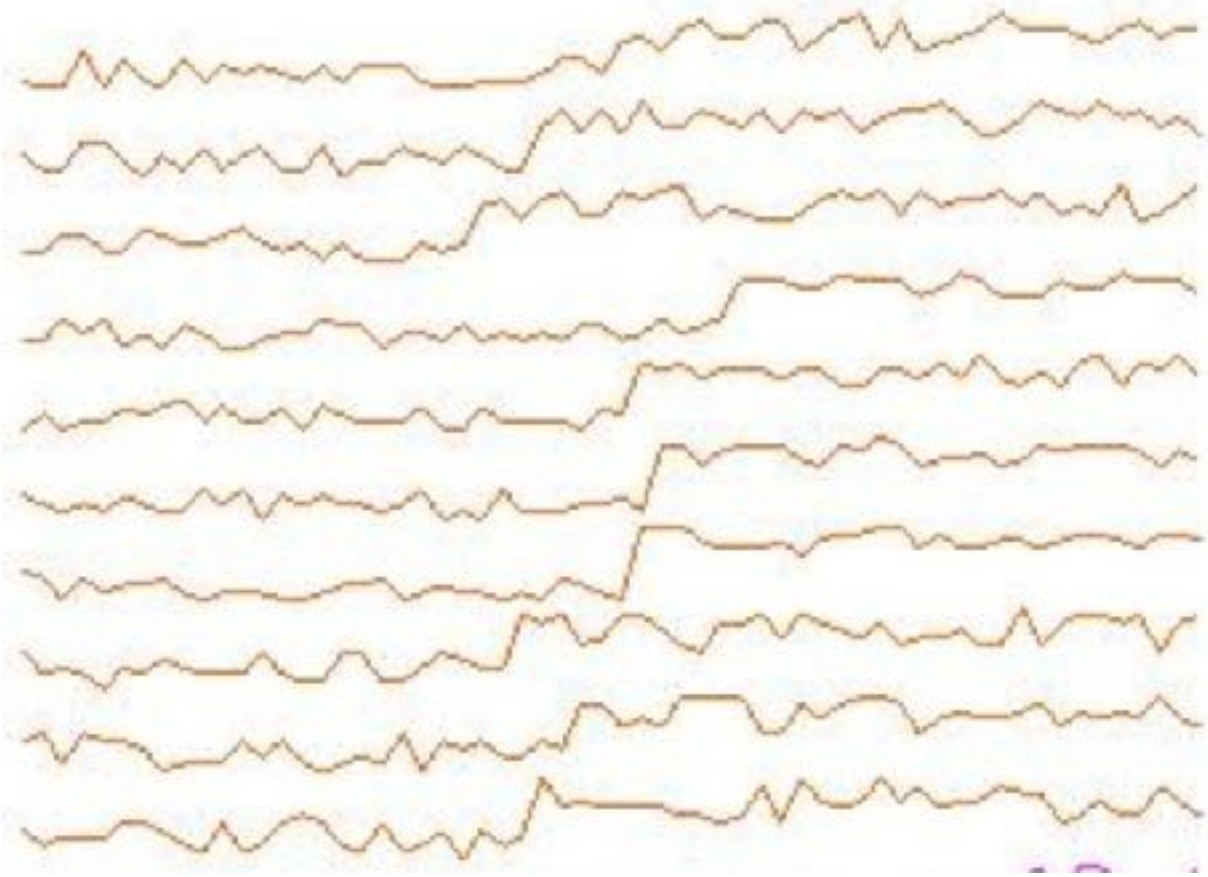
上升趋势



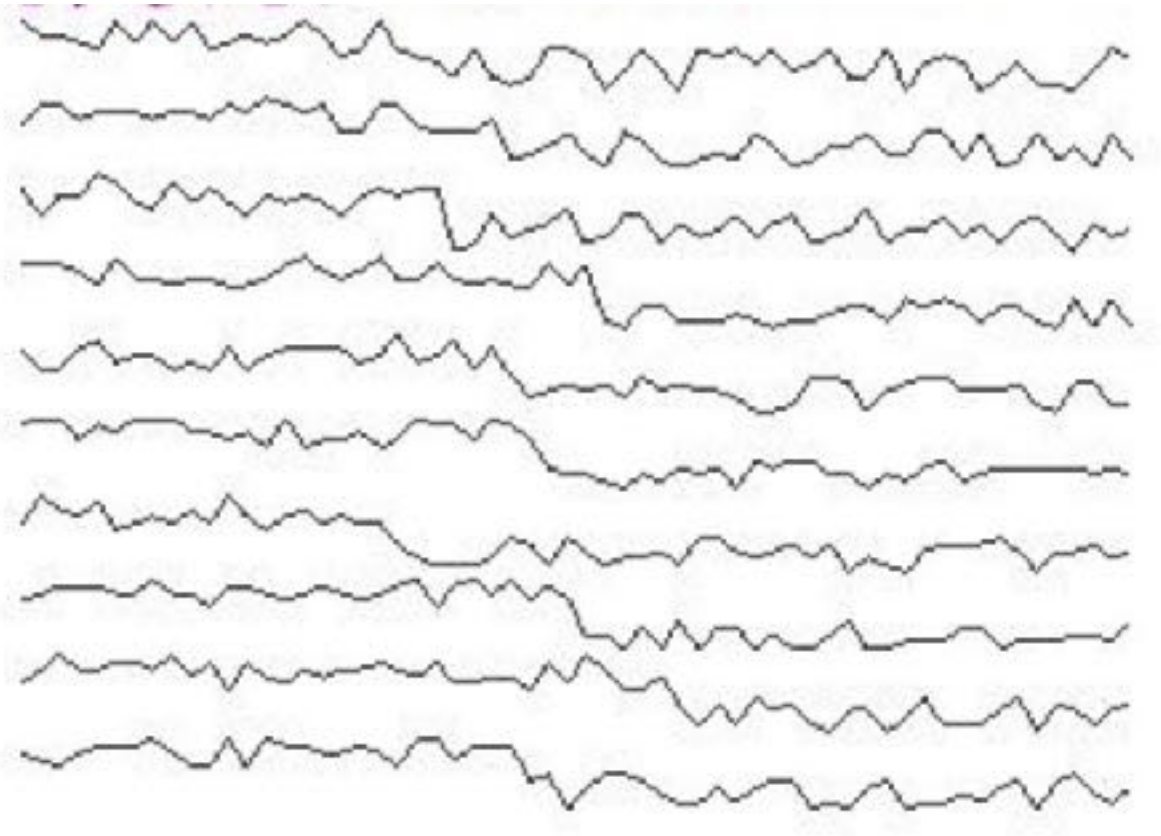
下降趋势



向上变化



向下变化



实验概述

- downloadUCIData 函数

  用于下载数据集的函数

- UCISequenceClassificationExample

  本实验中需要编辑的类

数据下载

进入UCISequenceClassificationExample，添加一行代码，调用UCIData类，下载数据。**注意** 下载需要一些时间，下载完成后，再次调用UCIData将验证下载已进行，而不会再次下载。运行下载两次是安全的。

请将这行代码添加至类。

```
UCIData.download();
```

为测试和训练数据设置记录读取器

```
SequenceRecordReader trainFeatures = new CSVSequenceRecordReader();
trainFeatures.initialize(new NumberedFileInputSplit(featuresDirTrain.getAbsolutePath() + "/%d.csv", 0, 449));

SequenceRecordReader trainLabels = new CSVSequenceRecordReader();
trainLabels.initialize(new NumberedFileInputSplit(labelsDirTrain.getAbsolutePath() + "/%d.csv", 0, 449));
```

设置miniBatchSize（微批次大小）和类别数量

```
final int miniBatchSize = 10;
final int numLabelClasses = 6;
```

为训练数据创建一个DataSetIterator

所需要的类的JavaDoc参见此处：https://deeplearning4j.org/datavecdoc/org/datavec/api/records/reader/impl/csv/CSVSequ enceRecordReader.html

```
DataSetIterator trainData = new SequenceRecordReaderDataSetIterator(trainFeatures, trainLabels, miniBatchSize,
  numLabelClasses,false, SequenceRecordReaderDataSetIte
rator.AlignmentMode.ALIGN_END);
```

将数据标准化

```
DataNormalization normalizer = new NormalizerStandardize();
normalizer.fit(trainData); trainData.setPreProcessor(normalizer);
```

设置测试数据

```
  SequenceRecordReader testFeatures = new CSVSequenceRecordReader();
  testFeatures.initialize(new NumberedFileInputSplit(featuresDirTest.getAbsolutePath() + "/%d.csv", 0, 149));
  SequenceRecordReader testLabels = new CSVSequenceRecordReader();
  testLabels.initialize(new NumberedFileInputSplit(labelsDirTest.getAbsolutePath() + "/%d.csv", 0, 149));
  DataSetIterator testData = new SequenceRecordReaderDataSetIterator(testFeatures, testLabels, miniBatchSize,
numLabelClasses,false, SequenceRecordReaderDataSetIterator.AlignmentMode.ALIGN_END);
  testData.setPreProcessor(normalizer);
```

设置神经网络

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
```

添加一个UI

```
UIServer uiServer = UIServer.getInstance();
StatsStorage statsStorage = new InMemoryStatsStorage();
net.setListeners(new StatsListener(statsStorage));
uiServer.attach(statsStorage);
```

训练网络

```
int nEpochs = 40;
for (int i = 0; i < nEpochs; ++i)
{
    network.fit(trainData);

    //Evaluate on the test set:

    if(i % evaluateStep == 0)
    {
        Evaluation evaluation = network.evaluate(testData);
        System.out.println(String.format(str, i, evaluation.accuracy(), evaluation.f1()));
        testData.reset();
    }

    trainData.reset();
}

Evaluation evaluation = network.evaluate(testData);
System.out.println(evaluation.stats());

System.out.println("—— Example Complete -----");
```

Physionet多变量时间序列分类实验

本实验中，您将构建一个神经网络，依据病人在重症监护室（ICU）住院2天的数据来预测病人死亡情况。

本实验目的

本实验主要介绍以下内容：

- 用DataVec摄取多变量序列数据
- 构建一个LSTM网络，对多变量序列数据进行分类

查看数据

数据存储于**顶层的**resources文件夹“src/main/resources/physionet2012”中。

标签

实验目的是预测死亡情况，即病人是否生存。“mortality”文件夹有一个病人的文件。每个文件只有一行，以0或1来表示病人是生存还是死亡

我们将用该标签来训练神经网络，使之学习如何进行预测。神经网络将预测一个标签，预测结果将与实际的标签值比对。

特征

各项病人数据的测量频率不同，有些数据在病人到达时测量，另一些数据每日测量一次，还有一些的测量频率更高。原始数据集中存储了时间戳、字段和值。为实现本例的目的，数据需要以两种方式重新组织。

重采样数据

“resampled”文件夹将包含重采样的数据，采样频率变为1小时，相关数值平均分摊至每个小时。每个文件会有49行，一行为标头，一行为其他48个时间步的值。该数据集仅包含每位病人住院两

天期间的数据。

数据掩码

每个字段会添加一个额外的标志字段，某些时间步中可能有部分字段未被测量，因而重复了先前的值或替换为推算值，此时标志字段设为1；如果时间步中采用的是实际测得的数值，则标志字段设为0。

序列数据

“sequence”目录包含序列数据。第一列“Time”为ICU收治病人的时间（精确到小时）。第二列“Elapsed”为上一个时间步后经过的时间（以小时为单位）。时间序列中的每个时间步的时间长度不同。例如，第一和第二个时间步之间可能相距1小时，而第二和第三个时间步之间可能相距2小时。

请注意：答案中设置了一个标志，可选择是否包含这两个与时间相关的列。

设定remove = 0，使用原始数据集，不包含Time和Elapsed两列（84个特征） 设定remove = 1，使用原始数据集，不包含Time列（85个特征） 设定remove = 2，使用原始数据集，不包含Elapsed列（85个特征） 将remove设为其他整数，使用原始数据集（86个特征，包含Time和Elapsed两列）

构建神经网络

本例中，我们将用ComputationGraphConfiguration（计算图配置）来配置神经网络。计算图网络的选项多于多层网络。

```
ComputationGraphConfiguration conf = new NeuralNetConfiguration.Builder()
```

构建一个ComputationGraph（计算图网络）并将建立的配置传递给它。

```
ComputationGraph model = new ComputationGraph(conf);
```

请照步骤取消代码的注释，以进行试验。

保存和加载已训练的模型

本实验中，您将：

1. 为一个模型添加一个UI实例
2. 保存模型
3. 加载已保存的模型

本实验目的

- 用ParentPathLabelGenerator（父目录标签生成器）
- 生成标签 添加UI服务器
- 保存已训练的模型
- 加载已训练的模型

第1步

- 打开IntelliJ 打开IntelliJ，进入Labs目录

第2步

- 打开MnistImageSave类

本例中使用了机器学习领域经典的手写数字识别问题。

第3步

- 查看输入

输入数据位于以下目录：resources/mnist_png/testing、resources/mnist_png/training。

打开其中一个目录，注意其中包含0-9号子目录，每个子目录包含24*24像素的灰度数字图像。

第4步

- 为神经网络添加一个UI实例。

示例参见SimplestNetwork。

第5步

- 保存模型

ModelSerializer的Javadoc参见此处https://deeplearning4j.org/doc/org/deeplearning4j/util/ModelSerializer.html

请使用ModelSerializer.writeModel，构造器应带有(modelname, location_to_save, boolean saveUpdater)

其中modelname是本示例中模型的名称。

保存位置已经指定。
File locationToSave = new File("trained_mnist_model.zip");

SaveUpdater（保存更新器）用于指明您会继续训练模型还是仅仅将其用于推断。

有些模型的训练可能需要数天甚至数周，任何长时间的训练过程都必须及时保存进度，以备不测。运行*ModelSerializer.writeModel*即可保存模型，保存频率可以是每过几小时一次。

第6步

运行代码

第7步

加载已训练的模型。

创建一个名为MnistImageLoad的类

添加一个Logger（日志记录器）

```
    private static Logger log = LoggerFactory.getLogger(MnistImageLoad.class);
```

添加Main方法

```
public static void main(String[] args) throws Exception {

}
```

设置三个整数对象height、width、channels（高、宽、通道）。其值分别设置为28、28和1。

请将保存的模型移动至resources文件夹下，使之位于类路径中。

创建两个文件对象，一个用于trained_mnist_model.zip文件，另一个用于将被模型分类的图像，number.png。

```
File modelSave = new ClassPathResource("trained_mnist_model.zip").getFile();
```

```
File imageToTest = new ClassPathResource("number.png").getFile();
```

用ModelSerializer.restoreMultiLayerNetwork([模型的文件对象])创建MultiLayerNetwork模型；

用NativeImageLoader类读取单幅图像，输入保存的神经网络以供评估。

```
NativeImageLoader loader = new NativeImageLoader(height, width, channels);
```

将图像输入一个INDarray

```
INDArray image = loader.asMatrix(imageToTest);
```

为取得预期结果，图像的处理方式必须与网络训练时的图像处理方式相同。

请创建一个DataNormalization缩放器，将图像的像素值缩放至0到1之间。

```
DataNormalization scaler = new ImagePreProcessingScaler(0,1);
```

请用DataNormalization缩放器作为转换方法转换图像。

```
scaler.transform(image);
```

将转换后的图像矩阵输入神经网络。

```
INDArray output = model.output(image);
```

将输出显示于控制台。

```
log.info(output.toString());
```

运行代码，验证结果。