EBook Gratis

APRENDIZAJE weka

Free unaffiliated eBook created from
Stack Overflow contributors.

#weka

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: weka

It is an unofficial and free weka ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official weka.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con weka

Observaciones

Esta sección proporciona una descripción general de qué es weka y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de weka, y vincular a los temas relacionados. Dado que la Documentación para weka es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación o configuración

Weka es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos pueden aplicarse directamente a un conjunto de datos o llamarse desde su propio código Java. Weka contiene herramientas para el preprocesamiento de datos, clasificación, regresión, agrupación, reglas de asociación y visualización. También es adecuado para desarrollar nuevos esquemas de aprendizaje automático.

Descargando e instalando Weka

Hay dos versiones de Weka: Weka 3.8 es la última versión estable, y Weka 3.9 es la versión de desarrollo. Para el borde de sangrado, también es posible descargar instantáneas nocturnas.

Las versiones estables reciben solo correcciones de errores, mientras que la versión de desarrollo recibe nuevas características. Weka 3.8 y 3.9 cuentan con un sistema de administración de paquetes que facilita que la comunidad Weka agregue nuevas funcionalidades a Weka. El sistema de gestión de paquetes requiere una conexión a Internet para descargar e instalar paquetes.

Puede descargar la aplicación para Windows / MacOS / Linux aquí .

Integre la biblioteca WEKA en su código:

pox.xml:

```
<dependency>
    <groupId>nz.ac.waikato.cms.weka</groupId>
    <artifactId>weka-dev</artifactId>
    <version>3.9.1</version>
</dependency>
```

gradle

```
compile group: 'nz.ac.waikato.cms.weka', name: 'weka-dev', version: '3.9.1'
```

Lea Empezando con weka en línea: https://riptutorial.com/es/weka/topic/3699/empezando-con-weka

Capítulo 2: ¿Cómo usar CPython Scripting en Weka?

Observaciones

¿Cómo instalar CPython en Weka?

Instalar wekaPython

1. Ir a `tools` , abrir `package manager`
2. busque `wekaPython` , seleccione y haga clic para instalar

Instalar bibliotecas de Python

1. instalar anaconda o conda
2. instala cuatro paquetes: numpy, pandas, matplotlib, scikit-learn
3. para la instalación completa doc ver conda

Examples

Hola ejemplo mundial para el CPython de Weka

> Vaya a `Explorer` , `iris.arff` datos de `iris.arff` , luego vaya a `CPython Scripting` , copie y pegue las siguientes líneas de códigos en `Python Scripts` :

```
hi = "Hello, CPython of Weka!"
hello = hi.upper()
iris = py_data
info = iris.describe()
```

> Para ver el resultado, vaya a `Python Variables` , seleccione `hi` , por ejemplo, y haga clic en `Get text`

Lea ¿Cómo usar CPython Scripting en Weka? en línea:
https://riptutorial.com/es/weka/topic/7921/-como-usar-cpython-scripting-en-weka-

Capítulo 3: Clasificación de texto

Examples

Clasificación de texto con LibLinear

- Crear instancias de entrenamiento de archivos .arff

```
private static Instances getDataFromFile(String path) throws Exception{

    DataSource source = new DataSource(path);
    Instances data = source.getDataSet();

    if (data.classIndex() == -1){
        data.setClassIndex(data.numAttributes()-1);
        //last attribute as class index
    }

    return data;
}
```

```
Instances trainingData = getDataFromFile(pathToArffFile);
```

- Use **StringToWordVector** para transformar sus atributos de cadena en representación numérica:

  * Características importantes de este filtro:

    1. representación tf-idf
    2. proveniente
    3. arrugas en minúsculas
    4. Para las palabras
    5. representación n-gramo *

```
    StringToWordVector() filter = new StringToWordVector();
    filter.setWordsToKeep(1000000);
    if(useIdf){
        filter.setIDFTransform(true);
    }
    filter.setTFTransform(true);
    filter.setLowerCaseTokens(true);
    filter.setOutputWordCounts(true);
    filter.setMinTermFreq(minTermFreq);
    filter.setNormalizeDocLength(new
SelectedTag(StringToWordVector.FILTER_NORMALIZE_ALL,StringToWordVector.TAGS_FILTER));
    NGramTokenizer t = new NGramTokenizer();
    t.setNGramMaxSize(maxGrams);
    t.setNGramMinSize(minGrams);
    filter.setTokenizer(t);
    WordsFromFile stopwords = new WordsFromFile();
    stopwords.setStopwords(new File("data/stopwords/stopwords.txt"));
```

```
    filter.setStopwordsHandler(stopwords);
    if (useStemmer){
        Stemmer s = new /*Iterated*/LovinsStemmer();
        filter.setStemmer(s);
    }
    filter.setInputFormat(trainingData);
```

- Aplique el filtro a trainingData: `trainingData = Filter.useFilter(trainingData, filter);`
- Crea el Clasificador LibLinear

    1. SVMType 0 a continuación corresponde a la regresión logística regularizada por L2
    2. Establezca `setProbabilityEstimates(true)` para imprimir las probalidades de salida

```
        Classifier cls = null;
        LibLINEAR liblinear = new LibLINEAR();
        liblinear.setSVMType(new SelectedTag(0, LibLINEAR.TAGS_SVMTYPE));
        liblinear.setProbabilityEstimates(true);
        // liblinear.setBias(1); // default value
        cls = liblinear;
        cls.buildClassifier(trainingData);
```

- Guardar modelo

```
    System.out.println("Saving the model...");
    ObjectOutputStream oos;
    oos = new ObjectOutputStream(new FileOutputStream(path+"mymodel.model"));
    oos.writeObject(cls);
    oos.flush();
    oos.close();
```

- Crear instancias de prueba desde el archivo `.arff`

```
 Instances trainingData = getDataFromFile(pathToArffFile);
```

- Clasificador de carga

```
Classifier myCls = (Classifier) weka.core.SerializationHelper.read(path+"mymodel.model");
```

- **Use el mismo filtro StringToWordVector que el anterior o cree uno nuevo para testingData, pero recuerde usar el trainingData para este comando:** **`filter.setInputFormat(trainingData);`** *Esto hará que las instancias de entrenamiento y prueba sean compatibles. Alternativamente, usted podría usar* *`InputMappedClassifier`*
- Aplique el filtro a testingData: `testingData = Filter.useFilter(testingData, filter);`
- ¡Clasificar!

    1. Obtenga el valor de clase para cada instancia en el conjunto de pruebas

```
 for (int j = 0; j < testingData.numInstances(); j++) {
    double res = myCls.classifyInstance(testingData.get(j));
```

```
}
```

`res` es un valor doble que corresponde a la clase nominal que se define en el archivo `.arff`. Para obtener la clase nominal, use: `testintData.classAttribute().value((int)res)`

---

2. Obtener la distribución de probabilidad para cada instancia

```
for (int j = 0; j < testingData.numInstances(); j++) {
   double[] dist = first.distributionForInstance(testInstances.get(j));
}
```

`dist` es una matriz doble que contiene las probabilidades para cada clase definida en el archivo `.arff`

Nota. El clasificador debe admitir las distribuciones de probabilidad y habilitarlas con: `myClassifier.setProbabilityEstimates(true);`

Lea Clasificación de texto en línea: https://riptutorial.com/es/weka/topic/7753/clasificacion-de-texto

Capítulo 4: Comenzando con Jython en Weka

Introducción

¿Por qué usaríamos Jython dentro de Weka? 1. Si no está satisfecho con lo que Explorer, Experimenter, KnowledgeFlow, simpleCLI le permiten hacer y está buscando algo para liberar el mayor poder de weka; 2. Con Jython, podemos acceder a todas las funcionalidades proporcionadas por Weka API, directamente dentro de Weka; 3. Su sintaxis es similar a la de Python, que se considera un lenguaje de scripting para principiantes;

Observaciones

Cómo configurar Jython en weka

1. instale la biblioteca `Jython` y `JFreeChart` desde el administrador de paquetes de Weka;
2. vaya a la terminal del directorio principal, ingrese `nano .bash_profile`
3. dentro de `.bash_profile` , agregue una línea de código como abajo

   ```
   export Weka_Data=User/Documents/Directory/Of/Your/Data
   ```

4. guardar y Salir
5. dentro de la `source .bash_profile` ejecución de terminal `source .bash_profile`

Luego, reinicie Weka, vaya a `tools` y haga clic en la `Jython console` , y puede probar los ejemplos anteriores

Examples

Cargar y filtrar datos

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.filters.Filter as Filter
import weka.filters.unsupervised.attribute.Remove as Remove
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")

# remove class attribute
rem = Remove()
rem.setOptions(["-R", "last"])
rem.setInputFormat(data)
dataNew = Filter.useFilter(data, rem)
```

```
# output filtered dataset
print(dataNew)
```

Construir un clasificador

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.classifiers.trees.J48 as J48
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# build classifier
cls.buildClassifier(data)

# output model
print(cls)
```

Clasificador de validación cruzada

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.classifiers.Evaluation as Evaluation
import weka.classifiers.trees.J48 as J48
import java.util.Random as Random
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# cross-validate classifier
evl = Evaluation(data)
evl.crossValidateModel(cls, data, 10, Random(1))

# print statistics
print(evl.toSummaryString("=== J48 on anneal (stats) ===", False))
print(evl.toMatrixString("=== J48 on anneal (confusion matrix) ==="))
```

Haz una predicción

```
# imports
import weka.classifiers.trees.J48 as J48
import weka.core.converters.ConverterUtils.DataSource as DS
import os
```

```
# load training data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal_train.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# build classifier on training data
cls.buildClassifier(data)

# load unlabeled data
dataUnl = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal_unlbl.arff")
dataUnl.setClassIndex(dataUnl.numAttributes() - 1)

# test compatibility of train/unlabeled datasets
msg = dataUnl.equalHeadersMsg(data)
if msg is not None:
    print("train and prediction data are not compatible:\n" + msg)

# make predictions
for inst in dataUnl:
    dist = cls.distributionForInstance(inst)
    labelIndex = cls.classifyInstance(inst)
    label = dataUnl.classAttribute().value(int(labelIndex))
    print(str(dist) + " - " + str(labelIndex) + " - " + label)
```

Validación cruzada del error clasificador Bubble

```
# Note: install jfreechartOffscreenRenderer package as well for JFreeChart library

# imports
import weka.classifiers.Evaluation as Evaluation
import weka.classifiers.functions.LinearRegression as LinearRegression
import weka.core.converters.ConverterUtils.DataSource as DS
import java.util.Random as Random
import org.jfree.data.xy.DefaultXYZDataset as DefaultXYZDataset
import org.jfree.chart.ChartFactory as ChartFactory
import org.jfree.chart.plot.PlotOrientation as PlotOrientation
import org.jfree.chart.ChartPanel as ChartPanel
import org.jfree.chart.renderer.xy.XYBubbleRenderer as XYBubbleRenderer
import org.jfree.chart.ChartUtilities as ChartUtilities
import javax.swing.JFrame as JFrame
import java.io.File as File
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "bodyfat.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = LinearRegression()
cls.setOptions(["-C", "-S", "1"])

# cross-validate classifier
evl = Evaluation(data)
evl.crossValidateModel(cls, data, 10, Random(1))

# collect predictions
```

```
act = []
prd = []
err = []
for i in range(evl.predictions().size()):
    prediction = evl.predictions().get(i)
    act.append(prediction.actual())
    prd.append(prediction.predicted())
    err.append(abs(prediction.actual() - prediction.predicted()))

# create plot
plotdata = DefaultXYZDataset()
plotdata.addSeries("LR on " + data.relationName(), [act, prd, err])
plot = ChartFactory.createScatterPlot(\
    "Classifier errors", "Actual", "Predicted", \
    plotdata, PlotOrientation.VERTICAL, True, True, True)
plot.getPlot().setRenderer(XYBubbleRenderer())

# display plot
frame = JFrame()
frame.setTitle("Weka")
frame.setSize(800, 800)
frame.setLocationRelativeTo(None)
frame.getContentPane().add(ChartPanel(plot))
frame.setVisible(True)
```

Display Graph

```
# imports
import weka.classifiers.bayes.BayesNet as BayesNet
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.gui.graphvisualizer.GraphVisualizer as GraphVisualizer
import javax.swing.JFrame as JFrame
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = BayesNet()
cls.setOptions(["-Q", "weka.classifiers.bayes.net.search.local.K2", "--", "-P", "2"])

# build classifier
cls.buildClassifier(data)

# display tree
gv = GraphVisualizer()
gv.readBIF(cls.graph())
frame = JFrame("BayesNet - " + data.relationName())
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)
frame.setSize(800, 600)
frame.getContentPane().add(gv)
frame.setVisible(True)

# adjust tree layout
gv.layoutGraph()
```

Lea Comenzando con Jython en Weka en línea:

https://riptutorial.com/es/weka/topic/8046/comenzando-con-jython-en-weka

Capítulo 5: Cómo usar R en Weka

Observaciones

¿Por qué usar R en Weka?

1. R es una poderosa herramienta para preprocesar datos.
2. R tiene una gran cantidad de bibliotecas y sigue creciendo.
3. R en Weka, puede obtener datos fácilmente, procesarlos y pasarlos a Weka a la perfección

---

Cómo configurar R en Weka

Para usuario de Mac

1. Reemplace la información antigua. Lista con la nueva proporcionada por Mark Hall.
2. descargar e instalar R
3. instale `rJava` dentro de R con

   instalar.paquetes ('rJava')
4. instala `Rplugin` con `Weka Package Manager`
5. vaya a la carpeta `weka 3-8-0` (si es la versión que está usando) y abra su terminal, y
6. Ejecuta las siguientes 2 líneas de códigos (gracias a Michael Hall)

   export R_HOME = / Library / Frameworks / R.framework / Resources
   java -Xss10M -Xmx4096M -cp.: weka.jar weka.gui.GUIChooser
7. para hacer la vida más fácil, dentro de un directorio en el que desea trabajar con weka, guarde el código de arriba en un archivo llamado `weka_r.sh`
8. hazlo ejecutable, dentro de la terminal de este directorio, ejecuta el siguiente código:

   chmod a + x weka_r.sh
9. pegue `weka.jar` de weka 3-8-0 en el directorio y ejecute el siguiente código:

   ./weka_r.sh

Ahora estás listo para ir. La próxima vez, solo tienes que ir a la terminal del directorio y ejecutar `./weka_r.sh` para iniciar R con Weka.

¿Cómo recibir los datos de Weka?

abrir Weka desde la terminal :
`Weka 3-8-0` al directorio de `Weka 3-8-0` , abra su terminal, ejecute el siguiente código:

```
java -jar weka.jar
```

datos a través de Weka Explorer :

1. panel de `preprocess` , haga clic en `open file` , elija un archivo de `weka data folder` ;
2. vaya al panel de la `R console` , escriba R scripts dentro del `R console box`

Datos a través de Weka KnowledgeFlow :

1. `Data mining processes` panel de `Data mining processes` , haga clic en `DataSources` para elegir `ArffLoader` por ejemplo, haga clic en el lienzo;
2. haga doble clic en `ArffLoader` para cargar un archivo de datos
3. Panel de `Scripting` , haga clic en `RscriptExecutor` en el lienzo
4. `option` + haga clic en `ArffLoader` , seleccione el `dataset` , luego haga clic en `RScript Executor` para vincularlos
5. haga doble clic en `RScript Executor` para escribir el script R, o
6. haga clic en `Settings` y seleccione `R Scripting` para usar la consola R con los datos de weka

---

Jugando Códigos R

1. cargar `iris.arff` con Explorer o KnowledgeFlow;
2. intente `Plotting inside R Console` ejemplo anterior

Examples

Trazado dentro de la consola R

> Los siguientes códigos se pueden encontrar en el curso de Weka

Dado que `iris.arff` está cargado en weka, dentro de la `R console` Weka Explorer o en la `R Scripting` Weka KnowledgeFlow, puedes jugar con los siguientes códigos para hacer hermosos diagramas:

```
library(ggplot2)

ggplot(rdata, aes(x = petallength)) + geom_density()

ggplot(rdata, aes(x = petallength)) + geom_density() + xlim(0,8)

ggplot(rdata, aes(x = petallength)) + geom_density(adjust = 0.5) + xlim(0,8)
```

```
ggplot(rdata, aes(x = petallength, color = class)) + geom_density(adjust = 0.5) + xlim(0,8)

ggplot(rdata, aes(x = petallength, color = class, fill = class)) + geom_density(adjust = 0.5)
+ xlim(0,8)

ggplot(rdata, aes(x = petallength, color = class, fill = class)) + geom_density(adjust = 0.5,
alpha = 0.5) + xlim(0,8)




library(reshape2)
ndata = melt(rdata)
ndata

ggplot(ndata, aes(x = value, color = class, fill = class)) + geom_density(adjust = 0.5, alpha
= 0.5) + xlim(0,8) + facet_grid(variable ~ .)

ggplot(ndata, aes(x = value, color = class, fill = class)) + geom_density(adjust = 0.5, alpha
= 0.5) + xlim(0,8) + facet_grid(. ~ variable)

ggplot(ndata, aes(y = value, x = class, colour = class)) + geom_boxplot() + facet_grid(. ~
variable)
```

Lea Cómo usar R en Weka en línea: https://riptutorial.com/es/weka/topic/7916/como-usar-r-en-weka

Capítulo 6: Errores fáciles de cometer al usar KnowledgeFlow

Introducción

Weka KnowledgeFlow (KF) es una excelente interfaz para usar. Sin embargo, el manual de Weka no cubre todos los detalles del uso de KF. Aquí sería un lugar para recopilar esos pequeños trucos o detalles que aprendí de los errores que cometí o cometeré a medida que pase el tiempo. ¡Muchas gracias a la gente de Wekalist (especialmente Mark Hall, Eibe Frank) por construir un maravilloso entorno de aprendizaje para Weka!

Observaciones

EntrenamientoSetMaker y TestSetMaker

1. un `ClassAssigner` debe estar vinculado entre `ArffLoader` y `TrainingSetMaker` o `TestSetMaker` .

ArffSaver

2. Para guardar el conjunto de datos en un archivo arff con éxito, es más seguro establecer `relationNameForFilename` en False dentro de la configuración de `ArffSaver` .

¿Cómo usar TimeSeriesForecasting en KnowledgeFlow?

1. Abra KnowledgeFlow, cargue el conjunto de datos con ArffLoader
2. Vaya a configuración, verifique la perspectiva de pronóstico de series de tiempo, haga clic con el botón derecho en ArffLoader para enviar a toda perspectiva
3. Ir a la perspectiva de pronóstico de series de tiempo para configurar un modelo
4. Ejecutar el modelo y copiar el modelo al portapapeles.
5. ctrl + v, y haga clic para pegar el modelo en el lienzo del proceso de minería de datos
6. guardar predicción junto con datos originales con ArffSaver

---

Examples

Cómo abrir el archivo KnowledgeFlow directamente desde el terminal

1. agregue la siguiente función en `.bash_profile` , guarde y salga

función wekaflstart () {

```
export R_HOME = / Library / Frameworks / R.framework / Resources
java -Xss10M -Xmx4096M -cp: weka.jar weka.gui.knowledgeflow.KnowledgeFlow "$ 1"
}
```

2. dentro de un directorio con un archivo `weka.jar` , abra su terminal, ejecute `wekastart "path to a knowledgeflow file"`

Lea Errores fáciles de cometer al usar KnowledgeFlow en línea:
https://riptutorial.com/es/weka/topic/8053/errores-faciles-de-cometer-al-usar-knowledgeflow

Capítulo 7: Instancias de carga

Examples

Archivos ARFF

Los archivos ARFF (Formato de archivo de relación de atributo) son el formato más común para los datos utilizados en Weka. Cada archivo ARFF debe tener un encabezado que describa cómo debe ser cada instancia de datos. Los atributos que se pueden utilizar son los siguientes:

- Numérico

Números reales o enteros.

- Nominal

Los atributos nominales deben proporcionar un conjunto de valores posibles. Por ejemplo:

```
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

- Cuerda

Permite valores de cadena arbitrarios. Generalmente se procesa más tarde utilizando el filtro `StringToWordVector` .

- Fecha

Permite especificar fechas. Al igual que con `SimpleDateFormat` de Java, esta fecha también se puede formatear; será por defecto el formato ISO-8601.

Un encabezado de ejemplo puede verse como sigue:

```
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Después del encabezado, cada instancia debe aparecer con el número correcto de instancias; si no se conoce un valor de atributos para una instancia `?` se puede utilizar en su lugar. A continuación se muestra un ejemplo del conjunto de instancias en un archivo ARFF:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
```

```
5.0,3.6,1.4,0.2,Iris-setosa
```

Cargando archivos ARFF

Dependiendo de la versión de Weka que se utilice, se deben utilizar diferentes métodos para cargar archivos ARFF.

Weka <3.5.5

El siguiente código de ejemplo muestra cómo cargar un archivo ARFF:

```
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
...
BufferedReader reader = new BufferedReader(new FileReader("data.arff"));
Instances data = new Instances(reader);
reader.close();
data.setClassIndex(data.numAttributes() - 1);
```

El índice de clase muestra qué atributo debe usarse para la clasificación. En la mayoría de los archivos ARFF, este es el último atributo, por lo que se establece en `data.numAttributes() - 1` . Si está utilizando una función Weka, como `buildClassifier` , debe establecer el índice de clase.

Weka> = 3.5.5

En la última versión de Weka es muy fácil cargar un archivo ARFF. Este método también puede cargar archivos CSV y cualquier otro archivo que Weka pueda entender.

```
import weka.core.converters.ConverterUtils.DataSource;
...
DataSource source = new DataSource("data.arff");
Instances data = source.getDataSet();
if (data.classIndex() == -1) {
  data.setClassIndex(data.numAttributes() - 1);
}
```

Cargando desde la base de datos

Muchas bases de datos se pueden utilizar en Weka. En primer lugar, el archivo DatabaseUtils.props debe editarse para que coincida con su base de datos; específicamente debe proporcionar el nombre de su base de datos, la ubicación, el puerto y el controlador correcto.

```
jdbcDriver=org.gjt.mm.mysql.Driver
jdbcURL=jdbc:mysql://localhost:3306/my_database
```

Entonces la base de datos se puede cargar utilizando un código simple.

```
import weka.core.Instances;
import weka.experiment.InstanceQuery;
...
InstanceQuery query = new InstanceQuery();
query.setUsername("user");
query.setPassword("pass");
query.setQuery("select * from mytable");
Instances data = query.retrieveInstances();
```

Algunas notas sobre la carga desde una base de datos:

- Asegúrese de que el controlador JDBC correcto esté en su ruta de clase.
- Si está utilizando Microsoft Access, se puede utilizar el controlador JDBC-ODBC que viene con el JDK.
- El método `InstanceQuery` convierte VARCHAR en atributos nominales y TEXT en atributos de cadena. Un filtro, como `NominalToString` o `StringToNormal` , puede convertir los atributos de nuevo a su tipo correcto.

Lea Instancias de carga en línea: https://riptutorial.com/es/weka/topic/5928/instancias-de-carga

Capítulo 8: Simple comparación de interfaces Weka

Introducción

Weka tiene muchas interfaces, Explorer, KnowledgeFlow, Experimenter, SimpleCLI, Workbench. Todos ellos comparten en su mayoría pueden hacer las mismas tareas, con diferente enfoque y flexibilidad. Aquí, vamos a explorar sus diferentes enfoques y flexibilidades.

Observaciones

Explorador

Pro:

1. haz todo rápido
2. Dar una vista rápida y completa de la estructura de datos.

cos: no se puede guardar el proceso;

Experimentador

Pro:

1. compare varios modelos a la vez, por ejemplo, ejecute 3 clasificadores diferentes contra 5 conjuntos de datos todos juntos, y vea el resultado comparado en un solo lugar;
2. experimento se puede guardar

Flujo de conocimiento

Pro:

1. hacer casi todas las cosas que el explorador puede hacer
2. puede salvar el proceso

cos:

1. KF no puede hacer el trabajo de Experimenter, ya que no admite bucles, pero ADAMS puede ayudar;
2. KF no puede acceder a funcionalidades de bajo nivel dentro de la API de Weka;

simpleCLI

pro: ejecuta tareas similares de lo que hace Explorer usando la línea de comandos

cos: no puede acceder a todas las funcionalidades de Weka API, Jython o Groovy se recomiendan para esta tarea.

Mesa de trabajo

pro: reúne todas las demás interfaces en un solo lugar

Examples

Ejemplos de SimpleCLI y Jython

simpleCLI

Ve a simpleCLI, ingresa el siguiente código

```
java weka.classifiers.rules.ZeroR -t path/to/a-file-of-dataset
```

Ejemplo de Jython

Códigos de la lección 5.1 del curso avanzado Weka MOOC

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.filters.Filter as Filter
import weka.filters.unsupervised.attribute.Remove as Remove
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")

# remove class attribute
rem = Remove()
rem.setOptions(["-R", "last"])
rem.setInputFormat(data)
dataNew = Filter.useFilter(data, rem)

# output filtered dataset
print(dataNew)
```

Lea Simple comparación de interfaces Weka en línea:
https://riptutorial.com/es/weka/topic/8042/simple-comparacion-de-interfaces-weka

Creditos

S. No	Capítulos	Contributors
1	Empezando con weka	Community, Gal Dreiman
2	¿Cómo usar CPython Scripting en Weka?	Daniel
3	Clasificación de texto	xro7
4	Comenzando con Jython en Weka	Daniel
5	Cómo usar R en Weka	Daniel
6	Errores fáciles de cometer al usar KnowledgeFlow	Daniel
7	Instancias de carga	SJB
8	Simple comparación de interfaces Weka	Daniel