Weka ではじめる機械学習

はじめに

初心者向けということでいくつかは正確でなかったりする部分もあります。
あくまでも網羅性や正確性は求めず、初学者がある程度の概要をつかめればと思った内容となっています。
最後に参考文献もあげているのでより深く、より正確な内容を学びたい方はそちらもご参照ください。

機械学習とは

データに潜むパターンを機械が見つけ出し、まだ見ぬ未知のデータの結果を予測する仕組み

- (大量の)データ
- データから抽出した特徴

を利用してその中のパターンを見つけ出し、それをモデル化する。
モデルとは見つけ出したパターンを記述した数式のことだと思っておけばよい。
機械学習において重要な点は、手元にあるデータの性質を完璧に捉えることではなく、すでに観測されたデータのパターンから未知のデータを予測することである。
機械学習は、見つけ出すパターンの種類によって以下のように分けられる。

- 教師あり学習
- 教師なし学習
- その他

これらについて簡単に説明する。

教師あり学習
データの特徴と正解となる教師データのセット与える。
予測する対象が数値であれば回帰、クラスであれば分類という風に分けられる。

回帰の場合、対象となるデータのパターンから実数値を予測する。
例えば深夜アニメのブルーレイの売り上げを

- ジャンル
- 放送時間・曜日
- 続編か否か
- ブルーレイではなくなる謎の煙(あるいは光)が登場するか
- 視聴率
- SNS での話題度

などを利用して予測できると仮定する。
この時、過去に放送されてすでにブルーレイも販売されているアニメに対して実際にわかっている売り上げ枚数とこれらの項目の値から現在放送中のアニメのおおよその売り上げ枚数を予測することができる。

ここで挙げた売り上げを予測するのに利用したものを説明変数(あるいは独立変数)といい、最初に書いた「データから抽出した特徴」に該当するものとなる。
また、予測対象となるブルーレイの売り上げは目的変数(あるいは従属変数など)と呼ばれる。

分類では予測対象が数値ではなくいくつかの項目となるものである。
例えば深夜アニメの場合、予測対象が推しメンなどといったようなものである。
分類問題では Yes/No, ある/ないのように 2 つのうちどちらか一方といったものを 2 値分類、推しメンのように複数の選択肢から 1 つを選ぶ場合は多値分類と呼ばれる。

教師なし学習

教師なし学習とは、正解データがないものを分類するための手法である。
データから抽出された特徴のみを与え、その中のパターンからデータをいくつかのグループに分類する。
教師あり学習に比べて正解データを用意しなくていいという便利さがあるが、どのようなパターンを学習するかは機械任せなので、当初想定したものとは別なパターンで分類される可能性がる。
そのため、分類結果がどのようになっているのかは人間が確認し、解釈を加える必要がある。

その他

半教師あり学習や強化学習など上記 2 パターンには含まれないものもある。
これらについての説明は割愛する。

機械学習の仕組み

今回扱うロジスティック回帰の数学的な説明は後ほど行う。
ここでは、基本的な内部の動作の仕組みを説明する。

- 数式によるモデル化
  データから抽出した特徴量に対して、$y = x_1w_1 + x_2w_2 + \cdots + x_nw_n$とする。
  この時回帰であれば y が実際に求める値、分類であれば y=1 なら Yes, y=2 なら No といったように項目の値を割り当てて、0.5 以上なら Yes のように扱う。
  x は抽出した特徴で添字として各特徴に番号を与えている。
  w は各特徴のウェイトで、各 x_n が y を決めるのにどれだけ重要かを示している。
  この時 **x** はすでにあるデータから与えられたものなので、値を自由に変更できるのは **w** となり、この値を調整することで与えられた正解データを最も正確に再現する **w** を求めることが機械学習の目的となる
  ここでは単純な各特徴の線形結合としているが、ものによって様々な形式が存在する。

- 最適な **w** の判断方法

  次に決める必要があるのは、上記式の計算結果がどれだけ正解データと同じになっているかを判断する方法である。

  通常機械学習ではすべてのデータに対して正解することはできない(やっても意味がない)ので、どれだけ近いかの指標が必要になる。

  そのための指標として

$$\mathrm{L(x)} = \frac{1}{n}\sum l(x) + \lambda b(x)$$

  などのような指標を用いる。

  この場合 l(x)は個々のデータに対する指標で、それを全データに対して足し合わせることで全体での評価指標を定義している。

  正解となる y に対して現在の **w** での予測値を$\hat{y}$とすると

$$\mathrm{l(x)} = (y - \hat{y})^2$$

  などを利用することで、二乗誤差として表現したりする。

  バイアス項(b(x))については、重要ではあるがここでの説明は割愛する。

- 最適な **w** の導出

  様々な **w** に対して L(x)の計算を行えば、その中で最も適切なものを見つけ出せるが、通常は微分を用いて各 w_n の勾配を求めることでより適切な値に近づけていく繰り返し計算を行う。

$$\frac{\partial l(x)}{\partial \mathbf{w}} = \left(\frac{\partial l(x)}{\partial w_1}, \frac{\partial l(x)}{\partial w_2}, \cdots, \frac{\partial l(x)}{\partial w_n}\right)$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^t$$

  として、**w** の変化がなくなるまで計算を繰り返す。

Weka とは

上記の通り、機械学習には数学的な知識が必要になるため、最初は非常にとっつきづらい。
また、今回は説明しないが大量のデータを扱うという性質上、プログラムで実装するには速度を速めるため計算方法の工夫がないと現実的な時間で処理が終了しないなどの問題もある。
もちろん、数式的な背景を知っておかないと実際に使う際には個々のアルゴリズムの特徴がわからずいい結果を出すために何を選択すれば良いかなどがわからなくて困ることもあるが、まずはブラックボックスとしてでも使っていると良い。
そのため、今回は数ある機械学習のライブラリの中から Java で使え、かつ GUI での操作も可能な Weka を利用して実際に機械学習を体験してもらう。

Weka はニュージーランドのワイカト大学で作成されている機械学習のライブラリで、特徴として

- GUI を使った操作が可能
- フル Java で実装されていて、プログラムから利用するための API もある

ということで、初心者でもとっつきやすい。

今回は GUI を利用してデータからモデルを作成する部分を体験したのち、作成したモデルを Java のプログラムから利用することで、システムへの導入も含めたイメージを持ってもらうことを可能にする。
今回作成するのは main メソッドから実行する簡単なものだが、使い方は変わらないので Web App からの利用なども同様の方法で可能である。

Weka とサンプルのダウンロード

Weka は通常公式ページ(http://www.cs.waikato.ac.nz/ml/weka/index.html)、からダウンロードして利用するが、今回は簡略化のためそれを含めた Fat Jar を生成するサンプルの Maven プロジェクトを用意した。

https://github.com/john-smith/weka_cist/

に用意してあるので、git で clone するか右下の「Download ZIP」を選択してダウンロードを行い、zip ファイルを解凍する。

Weka のデータ形式と今回扱うデータ

ダウンロードしてもらったものの data ディレクトリ以下に

- titanic.train.arff
- titanic.test.arff

という 2 つのファイルがあるのを確認する。
なぜ 2 つのファイルがあるかというと、最初に説明したように機械学習ではモデルを作る対象となる学習データのパターンを完璧に再現するのではなく、未知のデータに対しての予測が重要となる。
しかし、通常は未知のデータはまだ存在していないので、どの程度その「未知のデータ」に対して適切な予測ができているか判断できない。
そこで通常はデータを 2 つに分割し、片方を学習データ、もう片方をテストデータとして扱うことで擬似的に未知のデータへの予測性能を測定する。
train となっているのが学習データ、test となっているのがテストデータである。

次にファイルフォーマットについてだが、これは arff という独自の形式になっている。
機械学習ではこの arff のようにそれに適したフォーマットのデータを利用することが多い。

試しに学習データをテキストエディタで開いてみると arff のフォーマットが確認出来る。

```
@RELATION titanic

@ATTRIBUTE pclass          {1, 2, 3}
@ATTRIBUTE age              REAL
@ATTRIBUTE sex              {male, female}
@ATTRIBUTE servived         {0, 1}

@DATA
2,40,female,1
3,9,female,0
3,34,male,0
3,18,male,0
2,21,male,0
...
...
```

のような形式になっている。
これを上から順に説明していくと、

- 一番上の@RELATION はそのデータセットの名前
- 次に@ATTRIBUTE としてデータの各特徴の型を指定する
  REAL(NUMBER, INT)は数値型
  STRING は文字列
  {}で囲ったものはそこで指定した値のみをとるカテゴリカル変数
- @DATA 以下に@ATTRIBUTE で指定した順に実際のデータの値をカンマ区切りで並べる
  この部分に関しては通常と CSV と同じである

見ての通り、Weka が入力として利用するファイルのフォーマットは既にデータから必要な特徴を抽出した形式となっている。
通常、機械学習ではどのような特徴を抽出すべきかは人間が判断して、抽出したものを機械学習のライブラリが読み込める形式に変換してやる必要がある。

このデータの説明をする。
今回演習として行ってもらうのはタイタニックに乗った人たちの生死を予測する問題である。
このようなデータがネット上で機械学習用に用意されていたりするので、試しに遊んでみたい場合は、
UCI Machine Learning Repository(https://archive.ics.uci.edu/ml/datasets.html)
などを見てみるとよい。ただし全部英語である。

今回のデータセットではタイタニックに乗船していた人たちの

- 年齢(数値)
- 性別({male, female})
- 部屋の等級({1, 2, 3})
- 生存したか否か({0, 1}, 1 が生存)

として上 3 つの情報からどういう特徴の人が生存しやすい傾向にあったかを予測する。

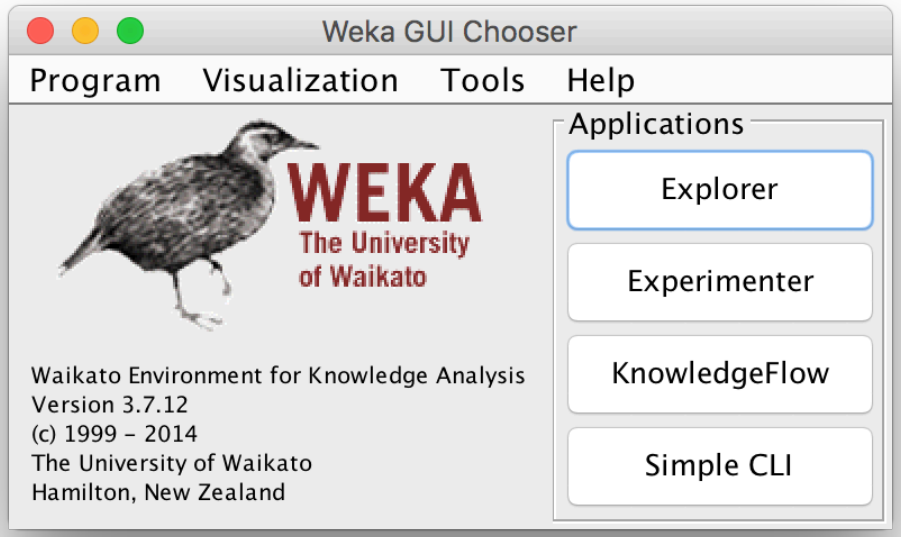
ちなみに、上記の特徴から生存しやすいパターンはおおよそ察しがつくと思われる。
このように機械学習を行う際は、特徴はなんでもいいわけではなく目的変数を予測するのに手がかりとなる情報を与えてやる必要がある。

GUI での Weka の実行

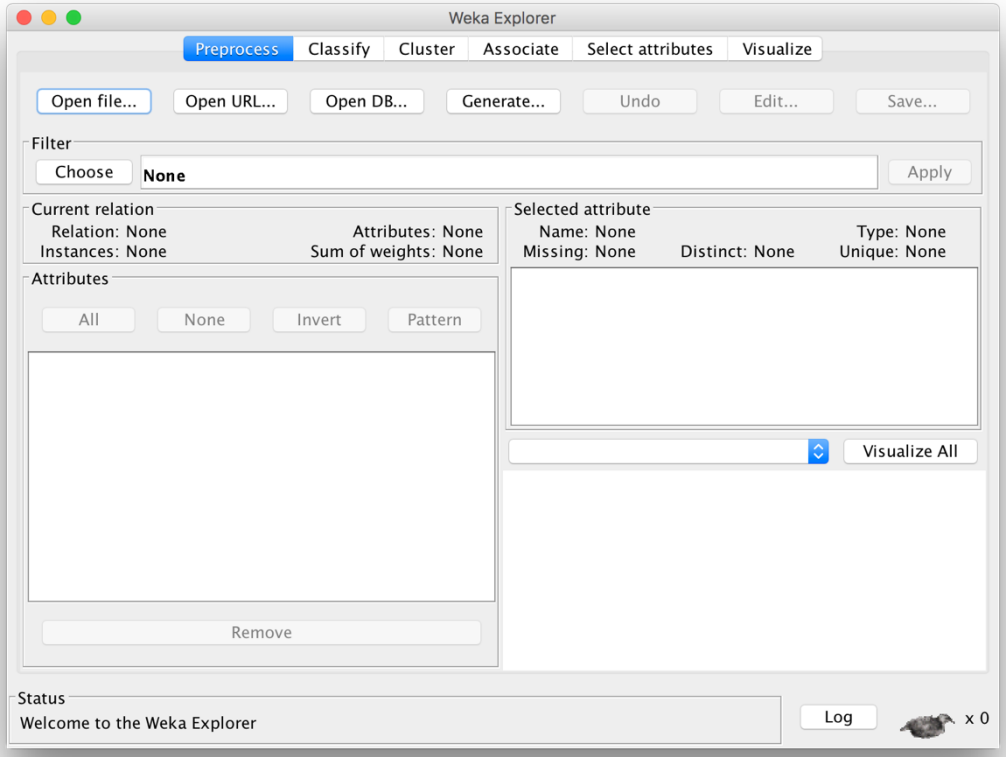
取得したプロジェクトの中で以下のコマンドを実行する

```
$ mvn package
$ mvn exec:java -Dexec.mainClass=weka.gui.GUIChooser &
```

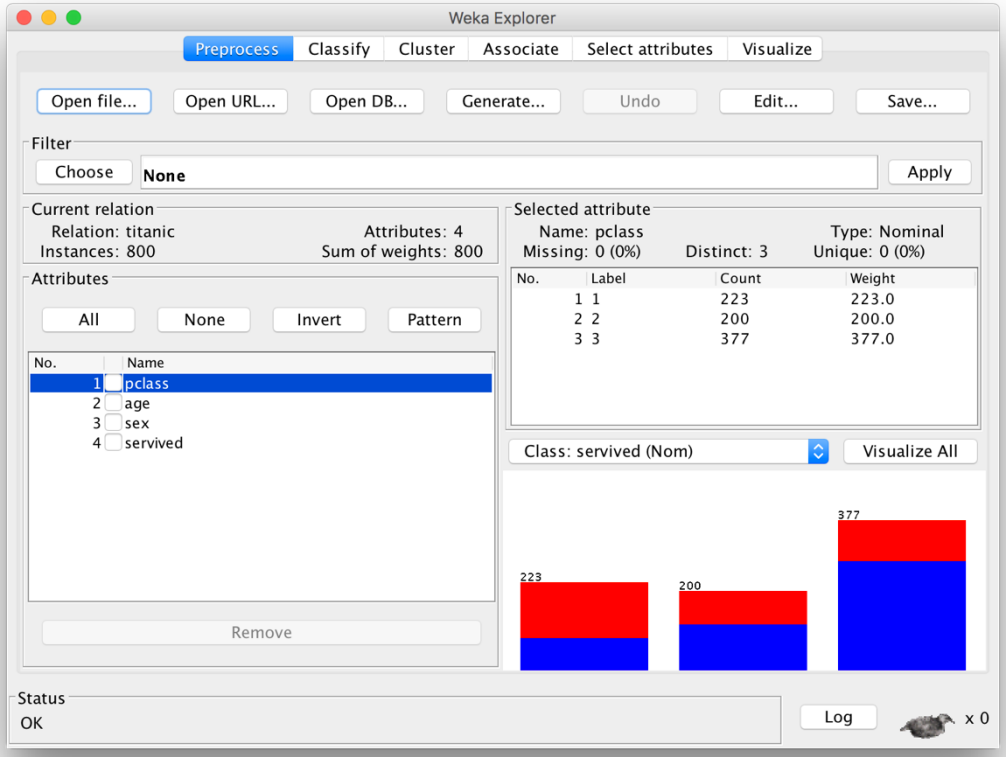
すると以下のような画面が立ち上がる。



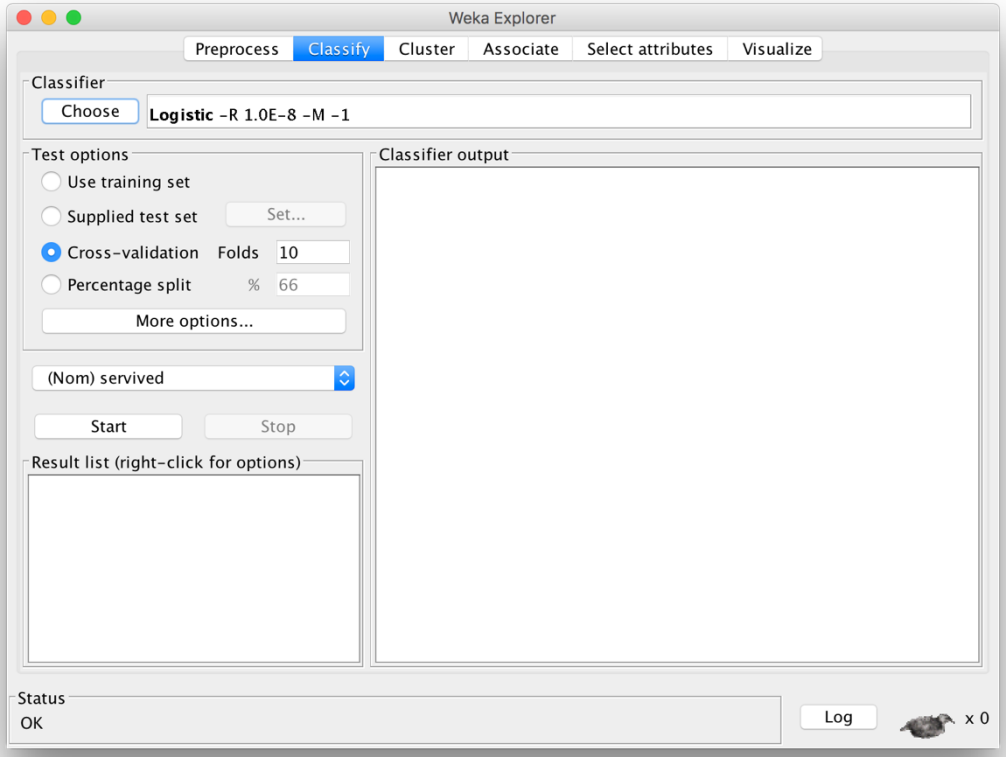
この画面で「Explorer」を選択する。



左上の「Open file...」を選択し、ダウンロードしたディレクトリの data/titanic.train.arff を選択する。



このように読み込んだデータに関する情報が表示される。
次に上部のタブから「Classify」を選択し、Classifier の中にある Choose から
weka > classifiers > functions > Logistic
を選択する。



そこまで設定したものがこの画面である。
これでロジスティック回帰という分類アルゴリズムの選択まで行われた。
※ロジスティック回帰であるが機械学習では分類として利用する
※ロジスティック回帰の詳細は後述

次に Test Options の項目を見る。
ここでは機械学習の評価で説明したテストデータの指定を行うが、テストデータに対する評価はこの後のプログラムからの実行にとっておくとして、ここでは Cross Validation という手法を利用する。

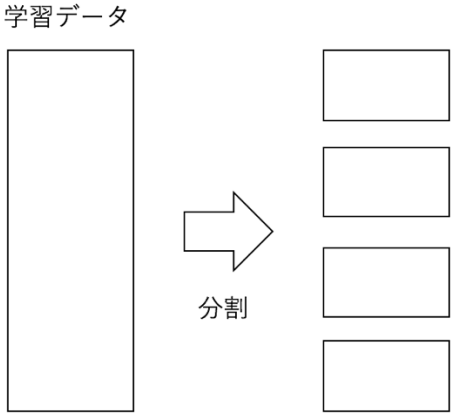
Cross Validation とは？

機械学習を行う際には学習データとテストデータを用意するが、それでは学習に利用出来るデータの数が減ってしまうことになる。

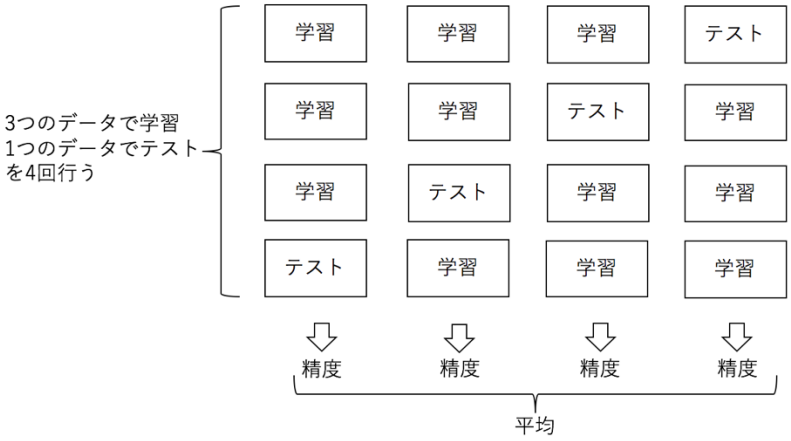
通常機械学習ではデータの数は多ければ多いほどいいが、データの用意や教師ありの場合正解データの付与などコストがあるため、常に分割するのに十分なデータ数があるとは限らない。

そこでうまいこと工夫してやるのが Cross Validation(CV)である。

CV ではまず学習データを何個かに分割する(図では 4 つ)

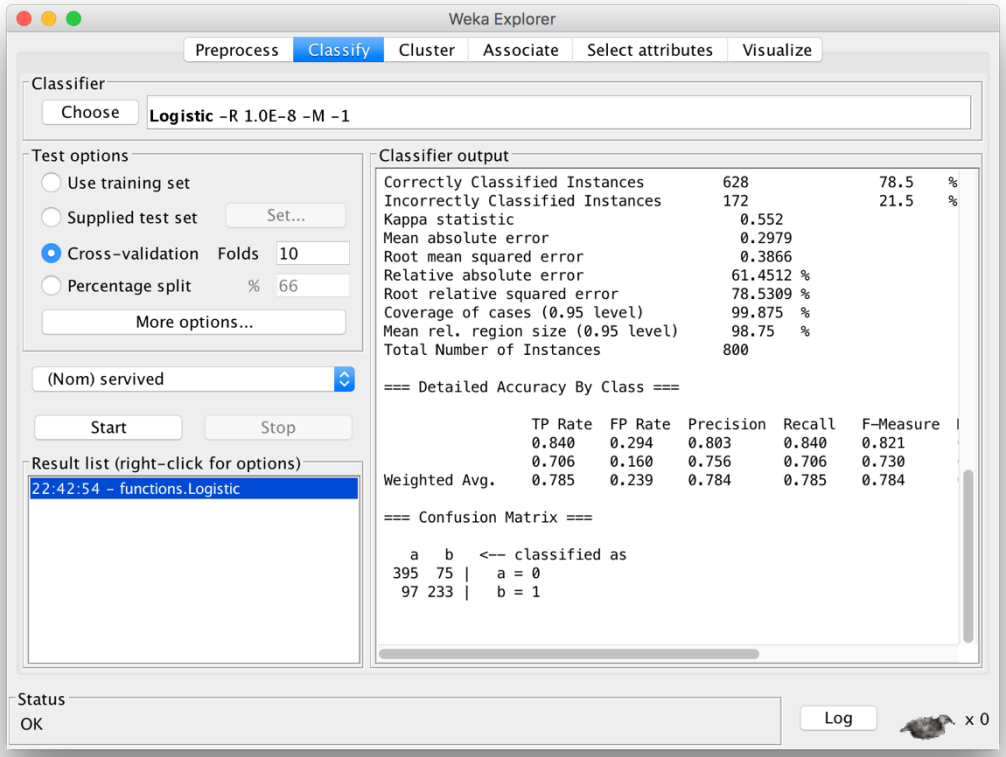


次にそれらのデータのうち 3 つ(分割数-1 個)を学習データ、1 つをテストデータとして、4 回(分割数回)学習を行いその平均値などを取ることで全てのデータを利用した学習が行える。



これによって十分な精度を満たしていることを確認した上でモデル作成時は全データを利用した学習が行える

今回はデフォルトの分割数 10 のまま実行する。
左側中段にある「start」を押せば学習を行い、その精度評価結果が右側の「Classifier output」に出力される。



学習における統計情報やいくつかの基準での評価結果が出ているが、今回はこの中で正解率と Confusion Matrix, Precision, Recall, F 値について説明する。

正解率
さて、今まで「評価基準」や「精度」という言葉が何度か登場したが、ここでようやくその説明をする。
まずは、最も単純な評価基準である正解率。

これは読んで字のごとく、機械学習の予測が何パーセント正解できたかである。
「=== Summary ===」にある「Correctly Classified Instances」がその値を表している。
もちろん高ければ高いほどいい値である。

Confusion Matrix, Precision, Recall, F 値

次に別な評価基準を考える。なぜ別な評価基準を考えるかというと正解率ではカバーしきれない問題があるからである。

例えば医療システムなんかを考えてみよう。病気の有無を判断する 2 値分類とした時、ほとんどの人は病気ではないためデータには偏りが生じる。
そのため、全ての人に病気ではないという判断を下せば正解率はそれだけで高くなってしまう。
しかし、実際には医療システムの場合むしろ病気であることをいかに判定するかが重要となる。
そのため、正解率のみでは本当は救える人をシステムの判断ミスで救えなかったという問題が発生してしまう可能性がある。

そのため、まずは Confusion Matrix という概念を導入する。
これは、今回のデータだと生存した=1, 生存しなかった=0 と扱っているので

	0 と予測	1 と予測
正解は 0	True Positive(TP)	False Negative(FN)
正解は 1	False Positive(FP)	True Negative(TN)

とデータを 4 分割して各項目に当てはまる数に名前をつける。
今回のデータだと

	0 と予測	1 と予測
正解は 0	395	75
正解は 1	97	233

となっていると思われる。
この時、対角線の合計値 / 全項目の合計値 が正解率となっている

次にこれに対して Precision, Recall という概念を導入する。

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

となる値である。

これらの意味するところは

- precision は 0 と予測したもののうち、実際に 0 であった割合
  - どれだけ確実に正解できているか
- recall は 0 が正解のうち、0 と予測できたものの割合
  - どの程度正解であるものを網羅しきれているか

となっている。1 についても同様の計算が可能である。

これらの値はトレードオフの関係になっており(確実に正解っぽいもののみを選べば precision は上がるが網羅できないので recall は下がる、なんでも正解とすれば recall は上がるがその分正解じゃないものも取り込むので precision は下がる)、利用する場面に応じてどちらをどの程度重視すべきが変わる。
一つの指標として、2 つの調和平均を取った F 値という指標がある。

$$F = \frac{2 * Recall * Precision}{Recall + Precision}$$

これらの値は出力中の「=== Detailed Accuracy By Class ===」および「=== Confusion Matrix ===」として出力されている。

最後に作成したモデルを保存する。
画面左下の「Result list」に今回作成したモデルが
[作成時間 – functions.Logistics]の形式で表示されている(複数回学習を行った場合は複数表示されている)のでそれを右クリックし「Save model」を選択する。
保存場所はどこでも良いが、ダウンロードしたものの中に入れておくとこの先のプログラムからの操作で便利。

プログラムからの利用

さて、いよいよプラグラムを実装してみましょう！
今回は先ほど保存したモデルを読み込んで予測をしてみるプログラムを書いてみます。
やる内容としては

- テストデータを読み込んで予測
- 新たに単一のデータが入ってきた場合を想定しての予測

の２つになります。
これができれば自分で作成したモデルを任意のシステムに組み込むことが可能になります。

テストデータの予測

テストデータを読み込んで予測を行い、その結果を出力します。
まずはメインメソッドのあるクラスを作成して以下のようなプログラムを書きます。

```
import weka.classifiers.Classifier;
import weka.classifiers.functions.Logistic;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;

import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class TitanicTestData {
   public static void main(String[] args) throws Exception {
      // モデルの読み込み。ファイルパスは保存した先に合わせる
      FileInputStream fis = new FileInputStream("model/titanic.model");
      ObjectInputStream ois = new ObjectInputStream(fis);
      Classifier classifier    = (Classifier) ois.readObject();

      // テストデータの読み込み。内部的な処理は weka がやってくれる
      DataSource source = new DataSource("data/titanic.test.arff");
      Instances instances = source.getDataSet();
      // 予測対象が何カラム目にあるかを指定する
      instances.setClassIndex(3);

      // テストデータ中に含まれる個々のデータに対して
      // その項目の値と予測値を出力
      for (Instance instance : instances) {
         System.out.print(instance);
         System.out.print(" : ");
         // 読み込んだモデルの classifyInstace にデータを渡すことで予測を行う
         System.out.println(classifier.classifyInstance(instance));
      }
   }
}
```

プログラムを書いて実行すると

```
3,34,male,0 : 0.0
2,30,female,1 : 1.0
3,21,female,0 : 1.0
3,33,male,0 : 0.0
3,21,male,0 : 0.0
2,39,male,0 : 0.0
3,25,male,1 : 0.0
3,24,male,0 : 0.0
3,43,male,0 : 0.0
3,30,male,0 : 0.0
```

のように各データの値と機械学習での予測値が出力される。

新たに入力されたデータの予測
次に新たに入力があった場合の 1 件単位での予測を行う。
これは arff を介さない分、プログラムにその情報を入力してやる必要が生じるのでひと手間かかる。
メインメソッド中の必要な部分のみを記載する。

```
// モデルの読み込み処理は先ほどと同じ処理を記述
// Classifier = ...
// arff 中の@ATTRIBUTE に相当するものを記述
Attribute attrPclass = new Attribute("pclass", Arrays.asList("1", "2", "3"), 0);
Attribute attrAge = new Attribute("age", 1);
Attribute attrSex = new Attribute("sex", Arrays.asList("male", "female"), 2);
Attribute attrSurvived = new Attribute("survived", Arrays.asList("0", "1"), 3);

// 各 Attribute をまとめた ArrayList を生成
ArrayList<Attribute> attrs = new ArrayList<Attribute>();
attrs.add(attrPclass);
attrs.add(attrAge);
attrs.add(attrSex);
attrs.add(attrSurvived);

// インスタンス作成時に@RELATION 相当の値と Attribute のリストを指定
// 3 つ目の引数は capacity だが、0 にすることで指定なしと出来る
Instances instances = new Instances("Logistic Regression", attrs, 0);
instances.setClassIndex(3);

// 各 Attribute と入力された値を指定して予測対象のインスタンスを生成
// この入力だと幼女貴族だが私の趣味ではないので悪しからず
Instance instance = new DenseInstance(4);
instance.setValue(attrPclass, “1”);
instance.setValue(attrAge, 5);
instance.setValue(attrSex, “female”);
instance.setDataset(instances);

// 予測結果を出力
System.out.print(instance);
System.out.print(" : ");
System.out.println(classifier.classifyInstance(instance));
```

出力は

```
1,5,female,? : 1.0
```

入力の Servived の値は指定されていないので「?」となっている。
ちなみに Weka では arff の入力でも未知の値を「?」として入れておける。

以上で簡単なプログラムのサンプルはおしまい。
あとは他のアルゴリズムについて調べてみたり、他のデータセットや自分で持っているもので機械学習を利用してみたりデータに適用してみるといいだろう。

また、本日 GUI で行った、学習や CV のプログラム版も
src/main/java/com/neetomo/weka/Titanic.java にあるのでそちらを参考にしてみても良いよ。
こちらも main メソッドで実行できます。

機械学習の注意点

最後に機械学習を使う上での注意点をいくつか

- 機械学習は魔法じゃないのでなんでもできるわけじゃありません
  - 人工知能はまだ先の話です
- 機械学習の結果は 100%正しいとは限りません
  - 精度評価で見ましたね
  - 入力で与えられたデータ以上のことはわからないのです
  - データが少なかったり直感が有効な場合は人の方が強い
- 意味のないデータを与えても精度は上がりません
  - 例えば性別を予測するのに生年月日を知っても意味がないとか
  - どんな特徴を選択するかは腕の見せ所です
- 今日話さなかったこともいっぱいあります
  - 他のアルゴリズムとか
  - パラメータチューニングとか
- 本気でやるなら英語からは逃れられません

おまけ

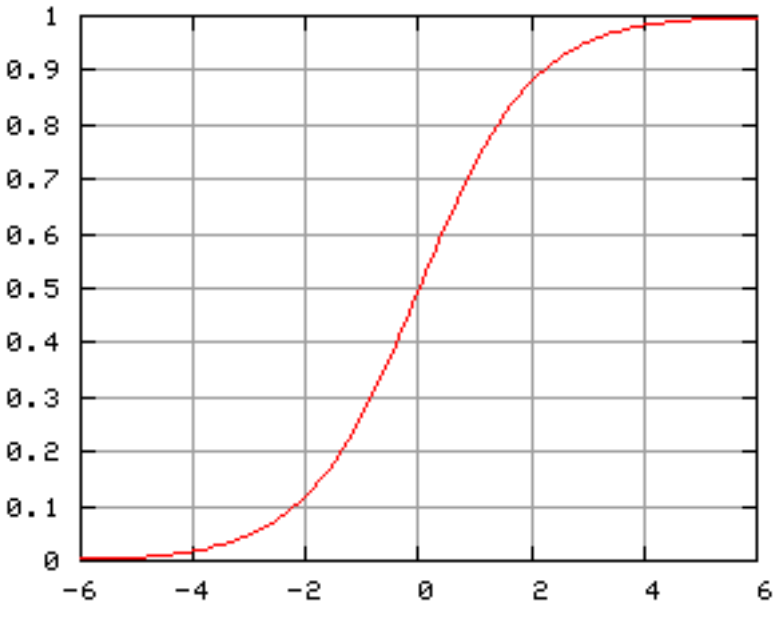
ロジスティック回帰の数学的な話

特徴 $\mathbf{w}$ に対してロジスティック回帰は以下の関数で定義さる。

$$\hat{y} = \frac{1}{1 + \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{w}\right)}$$

これはシグモイド関数と呼ばれる関数で、値が$[0,\ 1]$の範囲に収まるのが特徴。

シグモイド関数のグラフは



(wikipedia より)

のような形になり、その特性から確率値とみなすことができる。

このロジスティック回帰の損失関数 L(x)は l(x)を

$$l(x) = -\log\left(\frac{1}{1 + \exp(-\mathbf{x}^{\mathrm{T}}\mathbf{w})}\right)$$

とすることで

$$L(x) = \frac{1}{n}\sum l(x)$$

より勾配は

$$\frac{\partial L(x)}{\partial w_i} = \frac{1}{n}\sum(\hat{y} - y)x$$

と計算される

正則化項について

ここまでは説明しなかったが、機械学習には過学習というものが存在する。
これは学習データのノイズまで拾ってしまい、学習データ以外の予測ができなくなってしまう現象である。
これに対して正則化と呼ばれる **w** の値が大きくなりすぎないようにする工夫がある。
ここでは詳しくは述べないが、機械学習の話では比較的最初に出てくるので、知りたい方は参考文献をみてください。

参考文献

- フリーソフトで始める機械学習入門
- データサイエンティスト養成読本 機械学習入門編
- 入門機械学習
- オンライン機械学習
- データマイニングの基礎
- 言語処理のための機械学習入門
- IT エンジニアのための機械学習理論入門
- イラストで学ぶ機械学習
- パターン認識と機械学習
- 統計的学習の基礎