第 2 章

機械学習の基本的な手順

本章では機械学習の基本的な手順を学びましょう。まず、それぞれのステップで理解しておくべき内容を解説した後、各ステップの作業を支援してくれるツールを使いながら具体的なデータでその内容を説明します。個々の学習手法の中身に関しては次章以降で学びますので、この章では少し特殊な学習法である「学習しない」機械学習手法[1]を使って、機械学習全体の手順（図 2.1）を説明します。

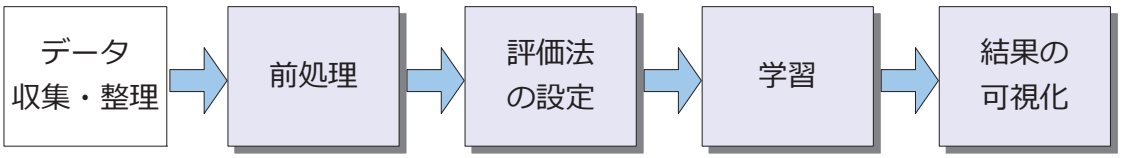

図 2.1　機械学習の流れ

使用するツールは **Weka** というデータマイニングソフトウェアです。Weka はニュージーランドの Waikato 大学で開発され、オープンソースソフトとして GNU GPL ライセンス[2]で公開されています。

本書で用いるのは 2013 年 7 月時点で最新である開発者用バージョン 3.7.9 です。Weka のサイト[3]から Download リンクをたどって、Windows, Mac, Linux 等各プラットフォームに応じたファイルをダウンロードしてインストールします。

---

*1 既に識別結果を知っているデータの中から、識別したい入力に最も近いものと同じクラスに識別する方法です。

*2 GPL ライセンスとは、「プログラムの実行・改良・再頒布は自由であるが、再頒布するときにはその頒布物も GPL ライセンスでなければならない」、というものです。

*3 http://www.cs.waikato.ac.nz/ml/weka/

インストールが成功して、Weka3.7 を実行すると、図 2.2 のような起動画面が出てきます。ここで、最初に使用するインタフェースを Explorer、Experimenter、Knowledge Flow、Simple CLI の中から選びます。それぞれの用途を表 2.1 に示します。

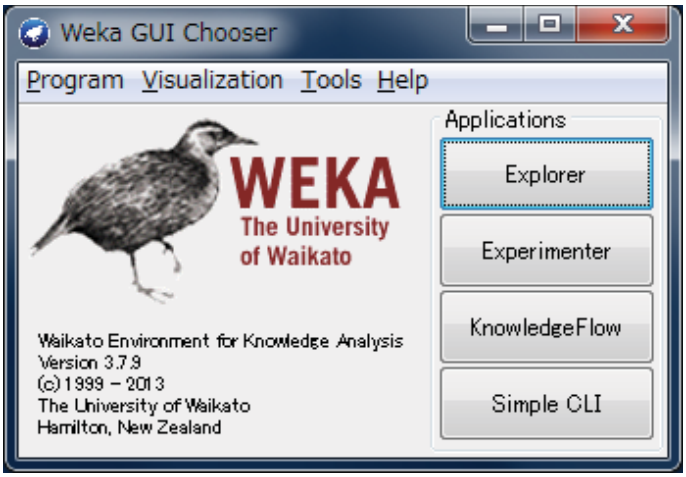

図 2.2　Weka の起動画面

表 2.1　Weka のインタフェース

インタフェース	内容
Explorer	データの前処理手法、適用する学習アルゴリズム、そのパラメータなどを試行錯誤するのに適する
Experimenter	学習アルゴリズムの比較など、評価実験に適する
Knowledge Flow	前処理から検証までの一連の流れを作成するのに適する
Simple CLI	Java のクラスを直接呼び出すコマンドラインインタフェース

この章では、まず Knowledge Flow インタフェースを用いて機械学習全体の流れを視覚的に構成する手順を説明します。その後、手軽に使える Explorer インタフェースについて、その使い方の概要を説明します。次章以降の例題・演習問題では、この 2 つのインタフェースを必要に応じて使い分けて用います。

2.1　データ収集・整理

2.1.1　データの収集

機械学習の第一段階はデータ収集です。購買記録からのパターンマイニングなど使用するデータが予め存在する場合と、自分でタスクと問題を設定して、そのために必要なデータを集める場合とがあります。教師あり学習を行う場合には、さらに正解の

付与作業が必要になります。

ただし、機械学習の勉強という目的では、先人が苦労して作成してくれたデータを利用することができます。本書の例題では主として Weka3.7 に付属のデータを使用します。Weka をインストールしたディレクトリに data というディレクトリがあります。そこに後で説明する ARFF という形式[1]でデータが納められています。Weka 付属のデータは過去の機械学習論文で取り上げられてきたデータがいくつもあり、いくつかの機械学習アルゴリズムの本質を理解するのに適したデータです。表 2.2 に本書で取り上げるデータの内容を示します。

表 2.2　Weka 付属のデータ

データ名	内容	特徴	正解情報
breast-canser	乳癌の再発	ラベル	クラス (2 値)
contact-lenses	コンタクトレンズの推薦	ラベル	クラス (3 値)
cpu	CPU の性能評価	数値	数値
credit-g	融資の審査	混合	クラス (2 値)
diabetes	糖尿病の検査	数値	クラス (2 値)
iris	アヤメの分類	数値	クラス (3 値)
Reuters-Corn	記事分類	テキスト	クラス (2 値)
supermarket	スーパーの購買記録	ラベル	なし
weather.nominal	ゴルフをする条件	ラベル	クラス (2 値)
weather.numeric	ゴルフをする条件	混合	クラス (2 値)

これらのデータは、これまで他の本で何度も目にしたものかもしれません。例題の内容を見て、「アヤメの分類なんか興味ないんだけど...」とか、「曇っていて風が弱ければゴルフをするっていう規則なんて何の役に立つんだ...」などという感想を持つかもしれません。たしかに、機械学習アルゴリズム自体は、まったく意味を持たない特徴 A、特徴 B、... などを設定して、乱数で人工的に生成したデータを使って説明することはできます。しかし、それで結果が出てもあまり達成感がありません。アルゴリズムを理解するだけでなく、実際にツールを動かしてみると、「近視で涙量が正常の人には、ソフトコンタクトレンズを勧める」のような実世界と対応付けてみて何となく正しそうに見える結果が出ているだけで、ちょっとうれしくなることもあります。

これらのデータは、特に特徴と正解情報の組合せに注目して使用してください。特徴（次元数、データの散らばり具合、スパース性[2]、欠損値の有無などの情報も含めて）と正解情報の組合せを見ただけで、適用すべきアルゴリズムが浮かぶようになれ

---

*1 テキスト形式の一種ですので、エディタでその内容を確認することができます。

*2 値を持つ次元の割合。

ば、機械学習初心者卒業といえるでしょう。

ただし、このような皆が使っているデータは、勉強のためと割り切って使うべきです。これらを使って研究を進めることに関しては、問題が提起されています（このことは「あとがき」でも触れます）[3]。

また、いくつかの練習問題では、比較的大規模なデータの揃っている **UCI Machine Learning Repository**[1] [4] を使用します。

2.1.2　データの整理

学習に用いるデータの収集が終わったら、それを機械学習プログラムが読み込んで処理しやすい形式に整理します。

1 章で説明したように、機械学習のデータは多次元のベクトルですので、ベクトルの各要素をカンマで区切り、1 行に 1 事例づつデータを並べてゆくというのが、最も単純な形式になります。この形式は CSV (Comma Separated Values) 形式と呼ばれ、表計算ソフトやテキストエディタで読み込み・編集ができます。ただし、CSV 形式では何番目の要素が何を表しているのかはデータだけからはわかりません[2]。最初に見出し行を付けるという方法も考えられますが、データの型など、もう少し情報をつけておきたいこともあります。それぞれの要素は数値なのかラベルなのか、ラベルの場合はどのような値が可能なのかという情報も、自分が作成したものではないデータを扱う際には役に立ちます。CSV 形式にこれらの情報をヘッダ情報として加えたものが **ARFF** (Attribute-Relation File Format) 形式です。

ARFF 形式のデータの例として、Weka に付属の iris.arff を図 2.3 に示します。これはアヤメ (iris) の種類 (Iris-setosa, Iris-versicolor, Iris-virginica) を、その萼 (がく) の長さ (sepal length)・幅 (sepal width)、花びらの長さ (petal length)・幅 (petal width) の計 4 つの特徴を用いて識別するための学習データです。

ARFF 形式のファイルにはデータセット名、特徴の情報、実際のデータを記述します。それぞれの記述方法について表 2.3 にまとめます。

`%` で始まる行はコメントです。Weka 付属のいくつかのデータには、コメントとしてそのデータの作成元の情報、関係論文、特徴の詳細説明などが書かれています。

`@relation`[3]で指定されるデータセット名は他のデータ集合と区別する目的で、英数字を用いて記述します。空白を含めたい場合はデータセット名を引用符 (’ または”)

---

*1 http://archive.ics.uci.edu/ml/

*2 教師あり学習に用いるデータの場合は、最後の要素を正解とすることが習慣になっています。しかし、全てのデータがその習慣に従っているという保証はありません。

*3 セクション名に用いる大文字・小文字は区別されません。

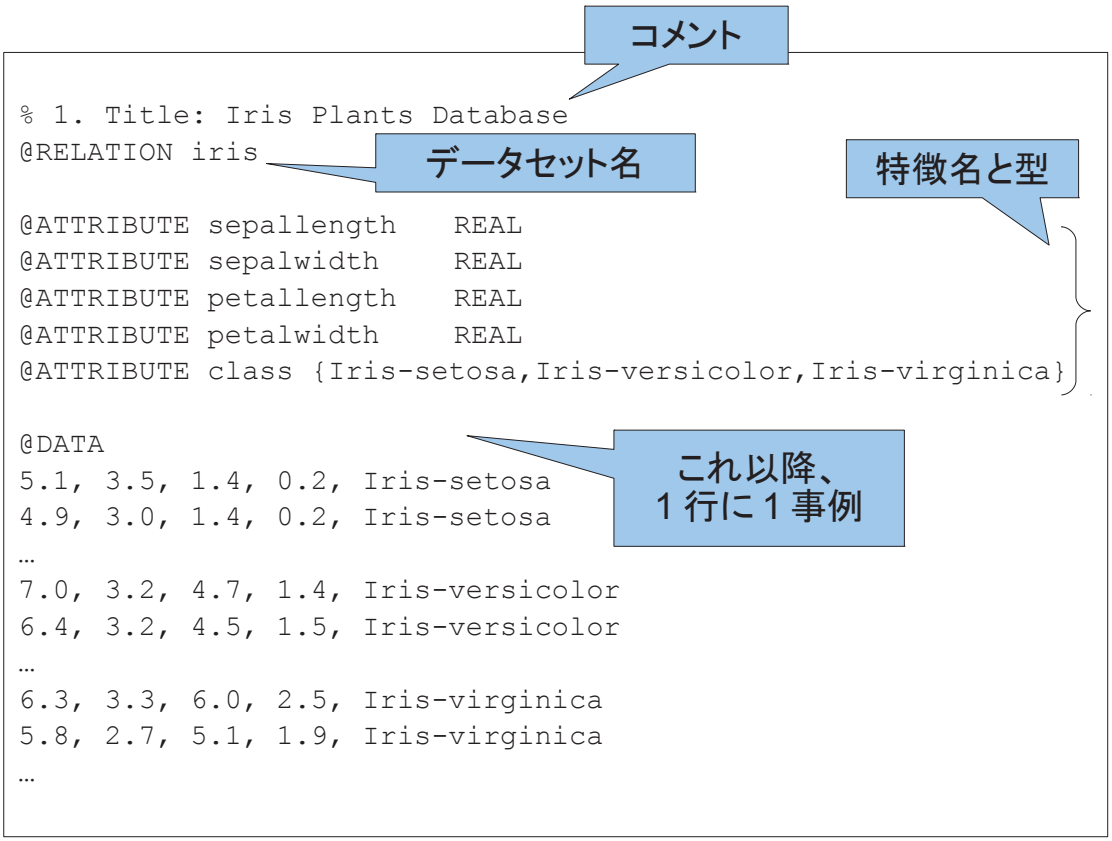

図 2.3　ARFF 形式のデータの例 (iris.arff)

表 2.3　ARFF 形式の仕様

セクション名	形式	内容
データセット名	@relation ⟨ データセット名 ⟩	データセット名を記述
特徴の情報	@attribute ⟨ 特徴名 ⟩ ⟨ 型 ⟩	1 行に 1 特徴を指定。データの各次元の意味は特徴の出現順に対応する。
データ	@data	@data の行以降、1 行に 1 事例のデータを CSV 形式で記述。欠損している要素は ? で表す。

で囲みます。

`@attribute` で指定される特徴の型は、numeric（数値）、ラベル、date（日付）、string（文字列）のいずれかです[1]、数値型を表すのに、integer や real と書かれている場合もありますが、いずれも numeric と見なされます。ラベルの場合は、型指定の代わりにラベルとして可能な値を {`yes, no`} のようにカンマ区切りで並べて中括弧で囲みます。識別問題の正解情報は、このラベル型を用いて、特徴名として class

*1 特徴名・型とも大文字・小文字は区別されません。

と指定することが多いのですが、Weka は自動的にその特徴を正解情報とみなすわけではありません。どの特徴を正解情報とするかは、Weka にデータを読み込んでから決定します。

`@data` と書いた次の行から、学習に用いるデータを 1 行に 1 事例の CSV 形式で記述します。

2.2 前処理

図 2.3 のような整理の終わった ARFF 形式のデータが手元にあれば、ここから機械学習ツール Weka の助けを得ることができます。以後、しばらくは Weka の Knowledge Flow インタフェースを使って、機械学習の基本的な手順を説明します。

図 2.2 の起動画面から Knowledge Flow インタフェースを起動すると、図 2.4 の初期画面が出てきます。基本的な手順は、左側の “Design” と書かれたペイン[1]（以後、デザインペインと呼びます）から機械学習の手順に応じた部品を選んで、中央上部の “Untitled1” というタブの付いたペイン（以後、レイアウトペインと呼びます）に配置します。レイアウトペインで個々の部品の設定をした後、それらを結合して全体の手順を組み上げます。これらの手順を総称して、レイアウトの作成と呼びます。レイアウトペインの上には、レイアウトを作成する際に用いるコマンドを並べたツールバーがあり、レイアウトペインの下には、現在の状況やエラーを報告するステータスペインがあります。

最初にこれから作成するレイアウトの名前を付けておきましょう。ツールバーの中にあるフロッピーディスク (Save layout) のアイコンをクリックし、名前を付けて保存します。ここでは firstFlow.kfml [2] としておきます。

2.2.1 データの読み込み

まず、機械学習に使うためのデータを読み込む部品を配置します。「1. 部品の選択 →2. 配置 →3. 設定」という手順になります。

1. デザインペインの中の DataSources というフォルダをクリックして展開すると、データ読み込みを行う部品がいくつか出てきます。この中で、ARFF 形式のファイル (iris.arff) を読み込むための部品として、ArffLoader を選択し

*1 ウィンドウが複数の表示領域に区切られているとき、それぞれの領域をペインと呼びます。
*2 kfml は XML Knowlege Flow layout files 形式をあらわす拡張子です。

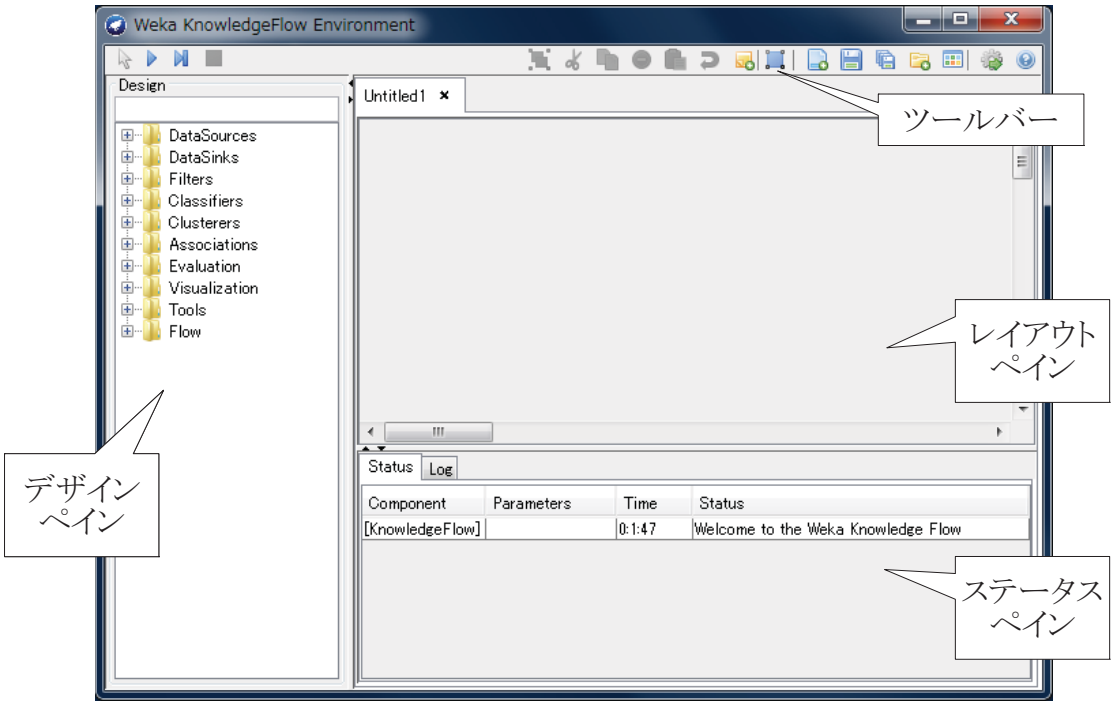

図 2.4　Knowledge Flow インタフェースの初期画面

ます。

2. ArffLoader を選択するとカーソルが十字型になるので、レイアウトペインの適当なところに配置します。
3. 次に、配置した部品の動作設定を行います。設定内容は、学習対象の ARFF ファイルの指定です。レイアウトペインに配置された部品を右クリックし、出てきたメニューの中から Configure を選んで (図 2.5)、設定画面を表示します。その設定画面を用いて、読み込むファイルを指定します。Filename と書かれたテキストボックスの右にある Browse ボタンを用いて、iris.arff を指定します。iris.arff は Weka がインストールされたディレクトリの中の data ディレクトリにあります。

教師あり学習の場合、正解を表す特徴を指定する必要があります。この操作を行うために ClassAssigner という部品が追加します。ArffLoader で読み込んだデータを ClassAssigner に送って、正解の特徴が指定されたデータを作るというイメージです。ClassAssigner を「選択 → 配置 → 設定」した後、ArffLoader と結合します。

1. デザインペインの Evaluation フォルダにある ClassAssigner を選択します。
2. レイアウトペインで ClassAssigner を ArffLoader の右側に配置します。
3. ClassAssigner を右クリックして Configure を選び、設定画面を表示します (図 2.6)。ClassAssigner では、classColumn の数値で正解特徴の位置を指定し

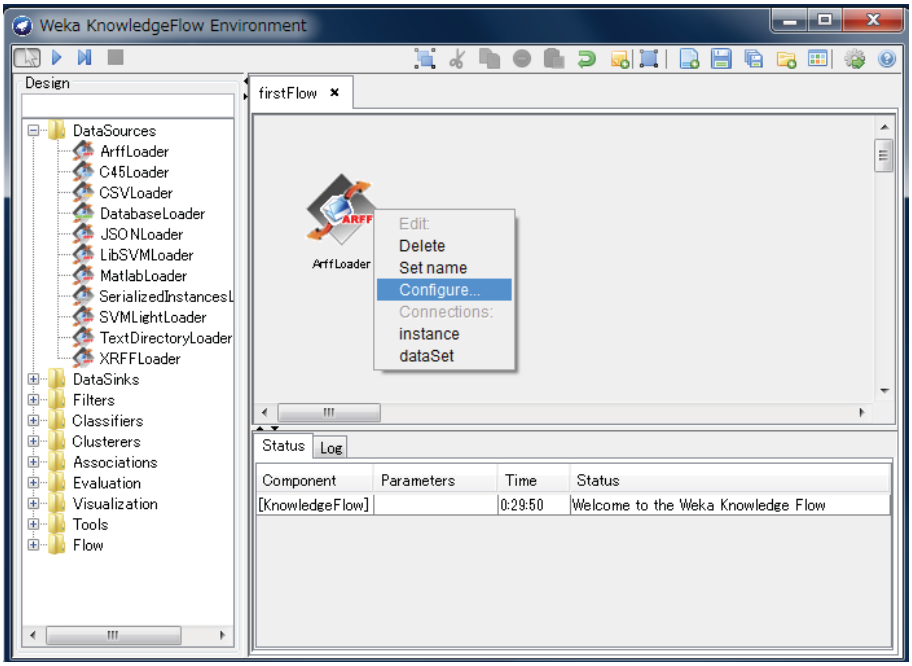

図 2.5　ArffLoader の設定

ます。正解は最後の特徴とすることが多いので、最後に関しては classColumn の値として数値ではなく、”last”という文字列を指定することができます。iris.arff では正解は最後の特徴ですので、classColumn には last を指定します。

4. 次に ArffLoader と ClassAssigner を結びます。ArffLoader を右クリックして dataSet を選びます。そうすると、配置済みの部品の中から dataSet を入力とする部品、すなわち ArffLoader と結合可能な部品の上下左右に点が現れて選択可能になります。この場合は、ClassAssigner が選択可能になりますので、これを左クリックで選択すると、ArffLoader から ClassAssigner に向かって矢印が引かれ、その上に dataSet と表示されます (図 2.7)。これで、ArffLoader で読み込まれたデータ集合が、ClassAssigner に送られ、正解特徴が指定されたデータができるという流れになります。

2.2.2　データの前処理

整理を終えたデータに対しては、機械学習を行う前にある程度の前処理を行います。学習段階で扱いやすいように特徴数を削減したり、特徴間で値のスケールを合わせるような処理が前処理です。

せっかく用意した特徴を削減すると聞くと、不思議な感じがするかもしれません。しかし、一般に音声認識や画像認識で用いられている定番の特徴集合でもない限りは、多次元の特徴には冗長性が多く含まれます。また、次元数が増えれば増えるほ

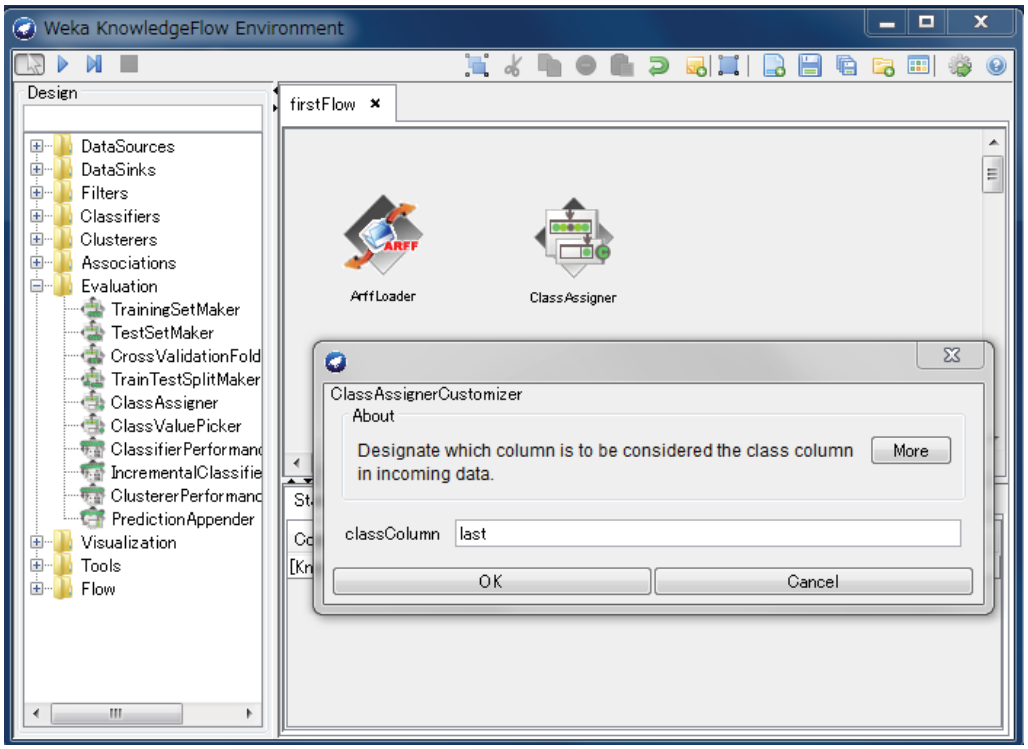

図 2.6　ClassAssigner での正解特徴の指定

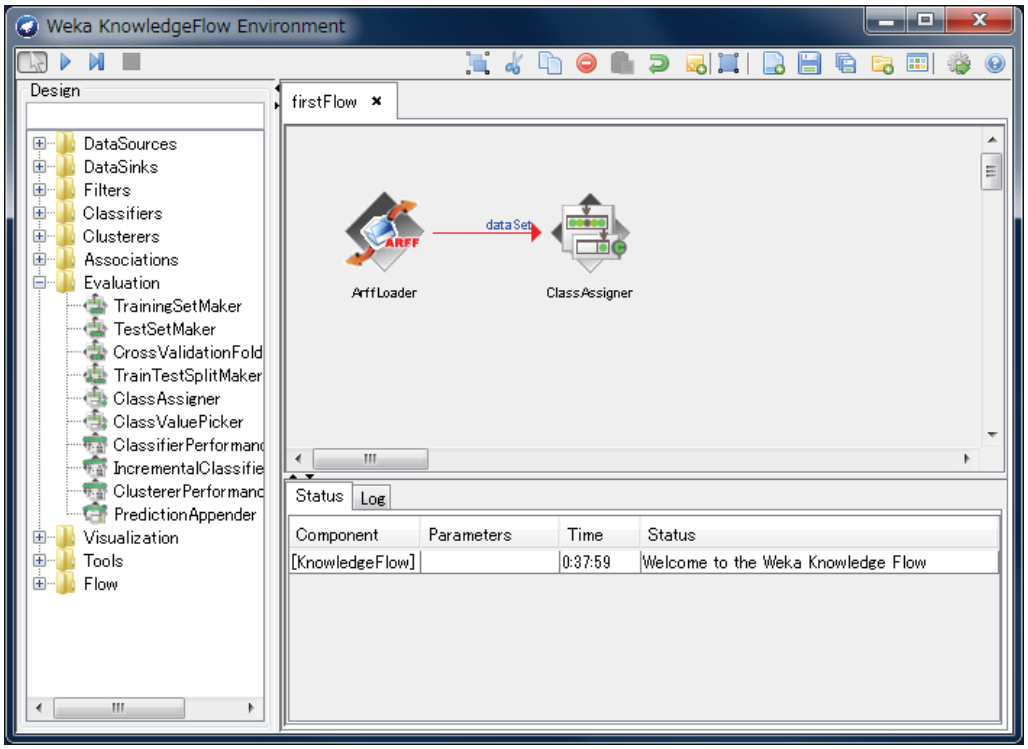

図 2.7　ArffLoader と ClassAssigner の結合

ど、学習データがまばらに存在することになり、そのようなまばらなデータから得られたモデルは、一般的に汎化能力が低いことがわかっています。したがって、特徴ベクトルの次元削減は、より汎化能力の高いモデルを学習するという観点から、重要な前処理ということになります。

特徴数の削減は**主成分分析** (principal component analysis: PCA) という手法を用いて行います (図 2.8)。

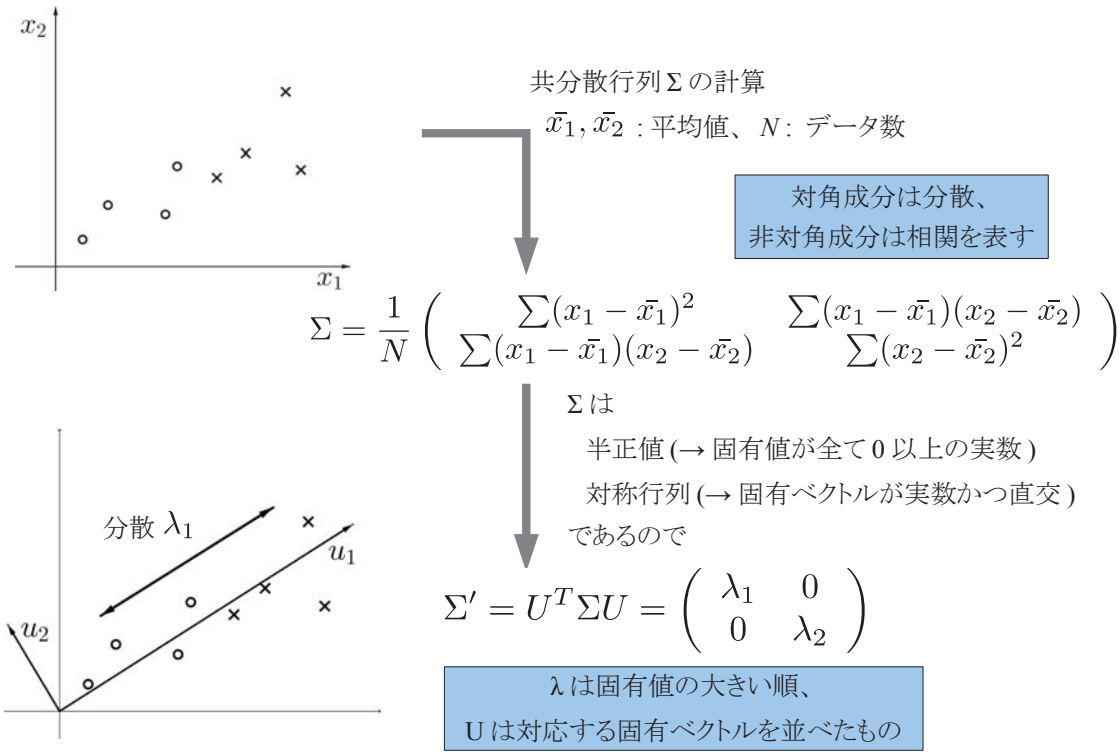

図 2.8 主成分分析の考え方

まず式 (2.2) に示す共分散行列 Σ を計算し、データの散らばり具合を行列で表現します。

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{\boldsymbol{x}\in D}\boldsymbol{x} \tag{2.1}$$

$$\Sigma = \frac{1}{N}\sum_{\boldsymbol{x}\in D}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T \tag{2.2}$$

まず式 (2.1) でデータの平均ベクトル $\boldsymbol{\mu}$ を求め、これを用いて式 (2.2) で共分散行列 Σ を求めるます。図 2.8 左上に示すような 2 次元データの場合、この平均ベクトルを $\boldsymbol{\mu} = (\bar{x_1}, \bar{x_2})$ とすると、共分散行列は式 (2.3) のようになります。

$$\Sigma = \frac{1}{N}\begin{pmatrix} \sum(x_1 - \bar{x_1})^2 & \sum(x_1 - \bar{x_1})(x_2 - \bar{x_2}) \\ \sum(x_1 - \bar{x_1})(x_2 - \bar{x_2}) & \sum(x_2 - \bar{x_2})^2 \end{pmatrix} \tag{2.3}$$

対角成分は次元毎の散らばり具合を表す分散に対応し、非対角成分は次元間の相関を表します。

次に、その共分散行列の固有ベクトルを求めると、固有値の大きい順にその対応する固有ベクトルの方向が、データの散らばりが最も大きい (すなわち識別するにあたっての情報が最も多い) 方向となります。固有ベクトルどうしは直交しますので、固有値の大きい順に軸として採用し、特徴空間を構成すると、例えば上位 2 位までなら 2 次元空間が構成でき、これらはもとの多次元特徴空間のデータの散らばりを最もよく保存した 2 次元空間ということになります。特徴空間の次元数が下がれば下がるほど、学習において推定するべきパラメータ数が少なくなりますので、学習結果の信頼性が高まります。もっとも、もとのデータの情報が大きく損なわれるほどに次元を削減してしまっては意味がないので、そのあたりの調整は難しいところです。主成分分析によって構成した軸では、対応する固有値が分散になりますので、全ての軸の固有値の和に対する採用した軸の固有値の和 (累積寄与率) を計算することで、全体のデータの分布をどの程度表現できているのか、見当をつけることができます。

また、次元削減をする／しないに関わらず、特徴の値の範囲を揃えておく操作は必要になります。一般に特徴はそれぞれ独立の基準で計測・算出しますので、その絶対値や分散が大きく異なります。これをベクトルとして組み合わせてそのまま評価・識別に用いると、絶対値の大きい特徴量の寄与が大きくなりすぎるという問題がありますので、値のスケールを合わせる必要があります。この処理を**正規化** (Normalization) と呼びます。もとの特徴の単位を消して、特定の値の幅にデータを納め、その値の違いを対等に評価できるようにしたものです。一般的に正規化は式 (2.4) を使って全ての次元の値を最小値 0、最大値 1 に揃えます。

$$\text{正規化後の値} = \frac{\text{元の値} - \text{最小値}}{\text{最大値} - \text{最小値}} \tag{2.4}$$

ただ、単純にスケールを合わせるだけでは、もし大きく外れる値**外れ値**といいます) があった場合に、その他の値が狭い範囲に押し込められるという問題があります。どのようなデータでも一定量含まれると思われる外れ値の影響を揃えるために、各特徴量の平均と分散を等しくするという作業が有効です。この処理を**標準化** (standardization) と呼びます。標準化を行うためには、各次元の値を、その次元の標準偏差で割る操作を行います。この操作によって、それぞれの次元の標準偏差が 1 になります。その後、各データから平均値を引くと、引いた後の平均値が 0 になり

ます。

今回用いる iris データは特徴の次元数が 4 であまり多くないので、次元数削減は行わず、特徴間で値のスケールを合わせる正規化のみ行います。

1. 正規化のための部品として、Filters フォルダの中にある Normalize を選択します[1]。
2. Normalize を ClassAssigner の右に配置します。
3. Normalize の右クリックメニューから Configure を選択して、設定を確認します。図 2.9 のように scale(値の幅) を 1.0, translation(最小値) を 0.0 とします。
4. 設定後、ClassAssigner と結合します。

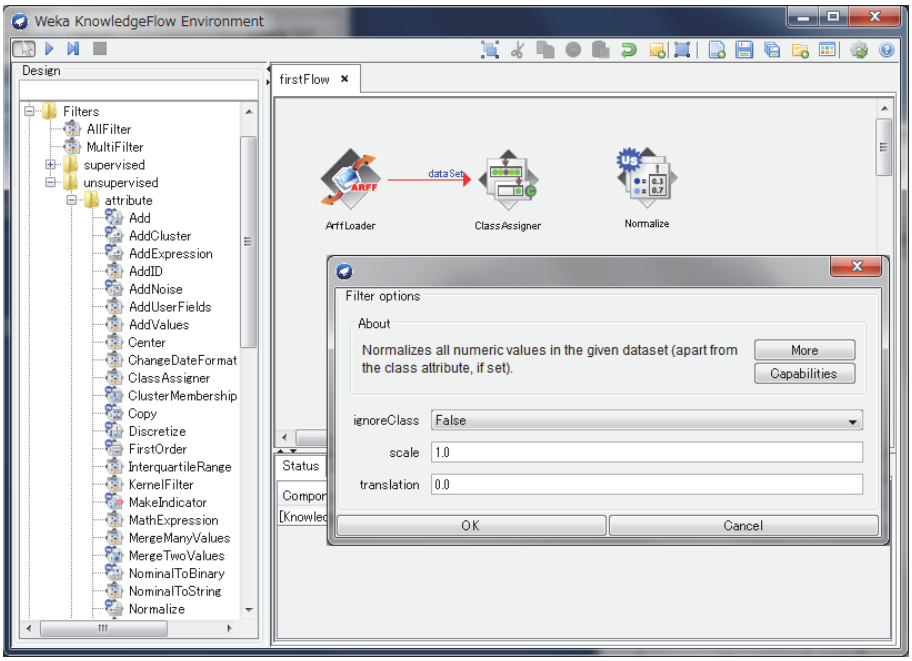

図 2.9　Normalize によるデータの正規化

これで学習のためのデータの準備ができました。

2.3　評価基準の設定

それでは学習、とゆきたいところですが、その前に学習結果の評価基準を設定します。

ここで扱っているデータは iris データなので、教師あり・識別の場合の評価基準を

---

*1 Filters→unsupervised→attribute→Normalize

考えます。この場合、学習データに対して正解率 100% でも意味がありません。未知データに対してどれだけの正解率が期待できるかが評価のポイントですが、どうやって未知データで評価すればよいのでしょうか。

もし学習データが大量にあったなら、半分を学習用、残り半分を評価用として分けてしまうという方法が考えられます。しかし、作成にそれなりのコストがかかる学習データがそんなにふんだんに使える状況は少ないでしょう。iris データも 150 事例しかありません。

このような場合、一般的には**交差確認法** (cross validation method: CV 法) と呼ばれる方法を用いて評価します (図 2.10)。この方法は学習データを m 個の集合に分割し、そのうちの $m-1$ 個で学習を行い、除外した残りの 1 つで評価を行います。そしてその除外するデータを順に交換することで、合計 m 回の学習と評価を行います。これで、全データがひととおり評価に使われ、かつその評価時に用いられる識別器は評価用データを除いて構築されたものとなっています。m を交差数と呼び、技術論文では交差数 m を 10 とするケースや、データの個数とするケースがよく見られます。m がデータの個数の場合を**一つ抜き法** (leave-one-out method) と呼ぶこともあります。

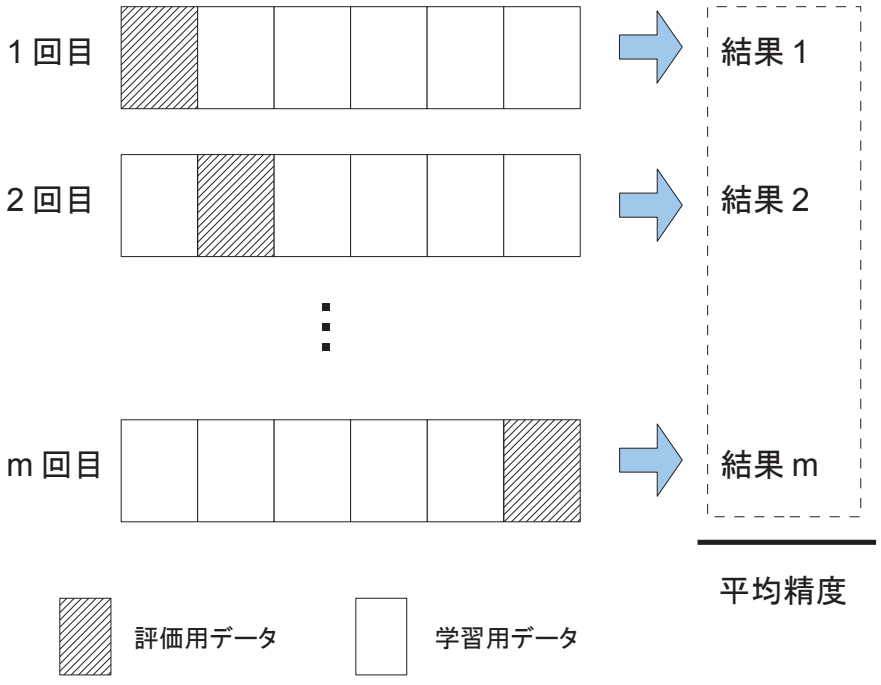

図 2.10　交差確認法

Knowledge Flow では、学習データを受け取って、設定する評価基準に従ってデータを送り出してくれる部品を学習器の前に置きます。

1. Evaluation フォルダから交差確認法を実現する CrossValidationFoldMaker

を選びます。

2. CrossValidationFoldMaker を Normalize の右に配置します。
3. CrossValidationFoldMaker の右クリックメニューから Configure を選択して、交差数 (folds) を 10 に設定します (図 2.11)。
4. Normalize と結びます。

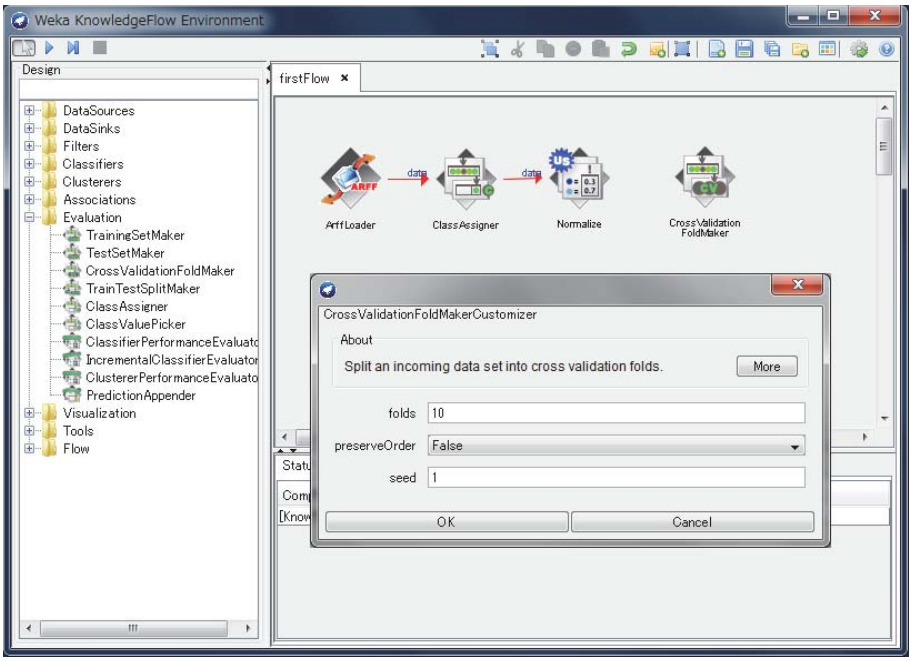

図 2.11　CrossValidationFoldMaker による評価基準の設定

2.4　学習 － k-NN 法－

さて、いよいよ学習です。ここでは学習アルゴリズムとして、入力されたデータに近い学習データを近い順に k 個選び、多数決などで所属するクラスを決定する k-NN **法** (k-nearest neighbor method) を使います（図 2.12)。

k-NN 法は、いわば学習データを集めるだけの学習法です。$k = 1$ の場合、識別したいデータと最も近い学習データを探して、その学習データのターゲットを答えとします。$k > 1$ の場合は、多数決を取るか、距離の重み付き投票で識別結果を決めます。調整すべきパラメータは近傍としてどの程度の学習データを考えるか（すなわち k の値）になります。また、学習データが多い場合、効率よく近傍を探索するアルゴリズムを組み合わせることもあります。

これから機械学習を学ぼうと意気込んでいるみなさんに、最初に紹介するのが「学習しない」学習法なので、がっかりされたかもしれません。しかし、データが大量に

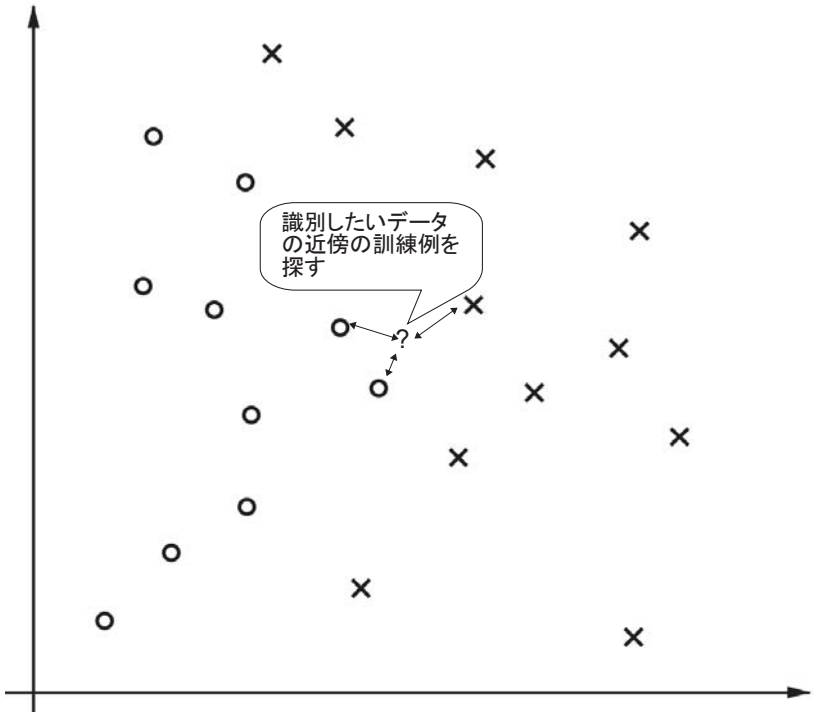

図 2.12　k-NN 法による識別

入手・記録可能で、かつ並列で高速に近傍計算ができる現在では、k-NN 法は驚くほどの性能を示すこともあります。スマートフォンの音声入力アプリで実現されている発話理解手法は、k-NN 法の考え方に近いものです。

Knowledge Flow では、学習器は前節で配置した評価器の後ろに配置します。

1. Classifiers フォルダから、k-NN 法を実現した学習器である IBk を選びます[1]。
2. IBk を、CrossValidationFoldMaker の右に配置します。
3. IBk の右クリックメニューから Configure を選択して、k の値を設定します。ここでは 3 としてみましょう（図 2.13）。
4. CrossValidationFoldMaker との結合は少し注意が必要です。CrossValidationFoldMaker から IBk に向けて、trainingSet と testSet の 2 本の矢印を引きます。表示上は 1 本にまとめられますが、必ず両方の結合を行ってください。これによって、まず学習データが送られて学習を行い、次に評価データでその結果を評価する、という手順を CrossValidationFoldMaker で設定した回数繰り返されます。

---

*1 Classifiers→lazy→IBk

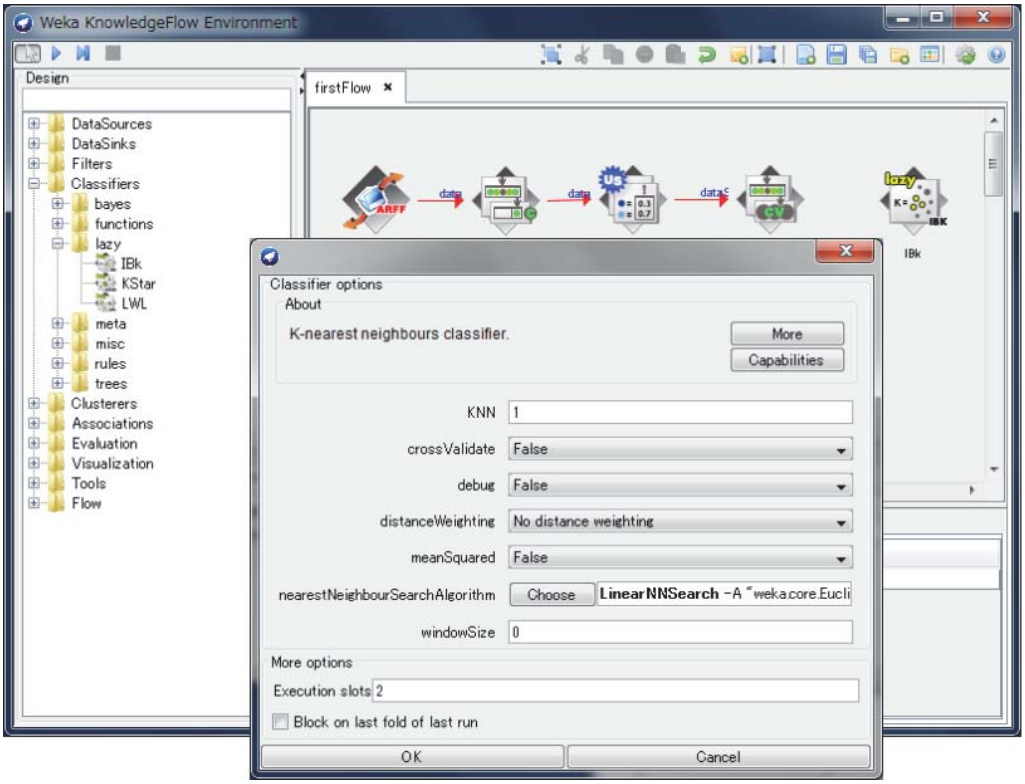

図 2.13　IBk の設定

2.5　結果の可視化

最後のステップは学習結果の可視化です。識別結果を計算し、表やグラフとして表示します。

プロセスの設定に入る前に、一般的に学習結果を評価する方法について考えてみます。まず、単純化するために 2 クラス識別問題の評価を対象とします。

2 クラス問題では、データがある概念に当てはまるか、否かを判定します。例えば、ある病気か否か、迷惑メールか否か、というような問題です。設定した概念に当てはまるデータを正例 (positve)、当てはまらないデータを負例 (negative) といいます。迷惑メールの識別問題では、迷惑メールが正例ですので、これを Positive と見なすのは少し変な気がしますが、惑わされないでください。あくまでも設定した概念に当てはまるか否かで、正例・負例が決まります。

さて、識別器を作成し、テストデータでその評価を行うと、その結果は表 2.4 のような表で表すことができます。この表のことを**混同行**列 (confusion matrix) あるいは**分割表** (contingency table) と呼びます。

対角成分が正解数、非対角成分が間違いの数を示します。例えば、見出し行の次の行の数値は、学習データの正例 50 個のうち、識別器が正と判定したものが 30 個、

表 2.4　混同行列の例

	予測＋	予測－	合計
正解＋	30	20	50
正解－	10	40	50
合計	40	60	100

負と判定したものが 20 個であったことを示します。この表から得られるもっとも単純な評価指標は、正解数である対角成分の和を右下隅の全データ数で割ったものです。表 2.4 の場合は $(30+40)/100 = 0.7$ となり、この値を**正解率** (accuracy) と呼びます。

実は、機械学習の評価は正解率を算出して終わり、というほど単純なものではありません。例えば、正例に比べて負例が大量にあるデータを考えてみましょう[1]。もし、正例のデータがでたらめに判定されていても、負例のデータがほとんど正確に判定されていたとしたら、正解率は相当高いものになります。そのような状況を見極めるために、機械学習の結果は様々な指標で評価する必要があります。有効な指標を紹介する前に、まず混同行列の各要素に表 2.5 に示す名前をつけておきます。

表 2.5　評価指標の定義

	予測＋	予測－
正解＋	true positive (TP)	false negative (FN)
正解－	false positive (FP)	true negative (TN)

例えば左上の要素は、正例に対して識別器が正 (positive) であると正しく (true) 判定したので、true positive といいます[2]。一方、右上の要素は正例に対して識別器が負 (negative) であると間違って (false) 判定したので、false negative と呼びます。最初の語が判定の成否 (true or false) を、後の語が判定結果 (positive or negative) を表します。

これらの定義を用いると、正解率は式 (2.5) のように定義できます。

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.5}$$

また、識別器が正例と判断したときに、それがどれだけ信用できるか[3]という指標

---

*1 例えば特定の疾病の判定など。病気にかかっていない人の数は、病気にかかっている人の数に比べてはるかに多いのが普通です。

*2 以後の式中で true positive の事例数を表すのに TP という略称を用います。

*3 すなわち、病気と判定されたとき、それがどれだけ正しいか。

を表すために、**精度** (precision)[4]が式 (2.6) のように定義されます。

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

さらに、正例がどれだけ正しく判定されているか[1]という指標を表すために、**再現率** (recall) が式 (2.7) のように定義されます。

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

精度と再現率を総合的に判断するために、その調和平均[2]をとったものを **F 値** (F-measure) と呼び、式 (2.8) のように定義されます。

$$\text{F-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

精度と再現率は一般にトレードオフの関係にあり、識別器のパラメータ設定でその値は変わります。例えば、iris データの sepallength 特徴だけを用いて、ある閾値 θ を設定し、その値以下であれば正例 (Iris-setosa) と判定する単純な識別器を考えてみます。sepallength 特徴の値の分布は図 2.14 のようになり、どこに閾値 θ を設定しても精度・再現率ともに 1 となることはありません。θ を小さめ (たとえば 4.8) に設定すれば、精度は 1 ですが、再現率が悪くなります。一方、θ を大きめ (例えば 6.0) に設定すれば再現率は 1 になりますが、精度が下がります。

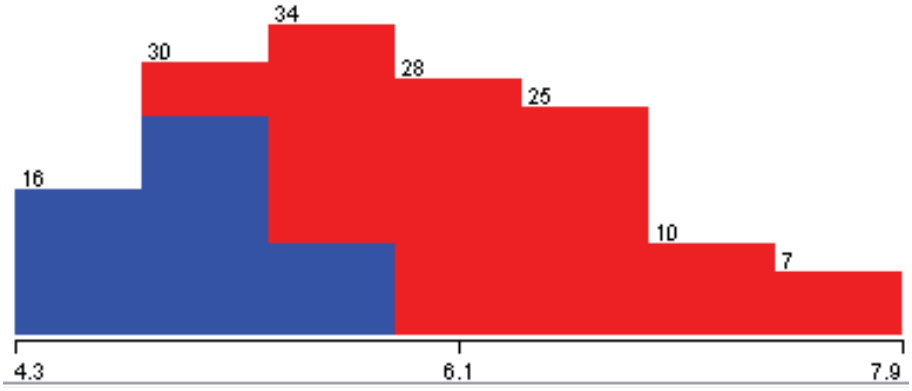

図 2.14　sepallength 特徴による分類

---

*4 情報検索の分野では適合率と呼ばれます。

*1 すわなち、病気を持つ人がどれだけ正しく抽出できているか。

*2 精度と再現率では、式の分母が異なるので、単純平均でその総合的な性能は求まりません。往路を 4km/h、復路を 6km/h で歩いたときの平均速度を求める問題と同じで、答えは 5mk/h ではなく、$2 \times \frac{4 \times 6}{4+6} = 4.8$km/h となります

このように閾値を変えたときの精度と再現率の関係を見るために、ROC **曲線** (ROC curve)（図 2.15）を用います[3]。

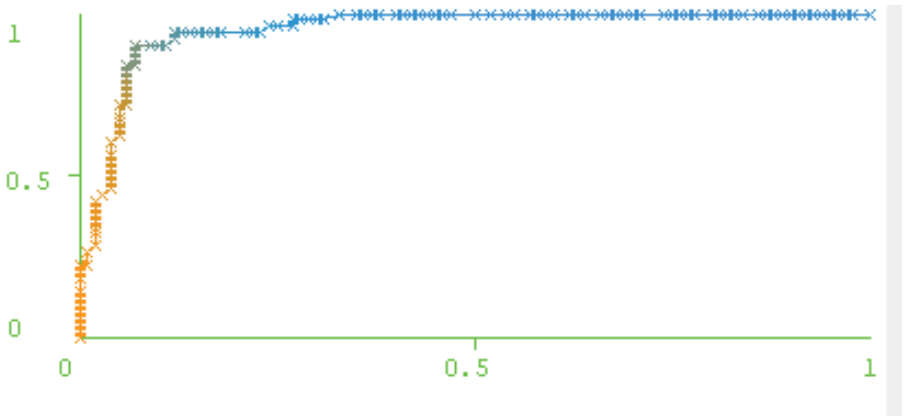

図 2.15　ROC 曲線

ROC 曲線は、横軸に false positive rate (FP/負例数), 縦軸に true positive rate (TP/正例数) を取って、閾値を変えていったときの値をプロットしたものです。

このように、機械学習の結果は様々な評価指標やグラフを使って評価することになります。

ここまで説明してきた 2 クラスの評価手法を多クラスに適用するのはそれほど難しくはありません。多クラスの場合は、クラス毎の精度や再現率を求め、そのクラスのデータ数に応じた割合を加算することで、総合的な評価を行います。

それでは評価用部品と可視化部品を配置して、レイアウトを完成させましょう。識別器の評価は、Evaluation フォルダにある ClassifierPerformanceEvaluator で行います。その結果を、Visualization フォルダにある TextViewer で表示します。

1. ClassifierPerformanceEvaluator、TextViewer を IBk の右側に配置します。
2. IBk から ClassifierPerformanceEvaluator へは batchClassifier で接続し、IBk で作成した識別器を送ります。
3. ClassifierPerformanceEvaluator から TextViewer へは text で接続し、評価結果をテキスト形式で送ります。

最終的には図 2.16 のようなレイアウトが出来上がります。

これでレイアウトが完成したので、左上にある右向き三角をクリックして学習プロセスを実行します。その右のボタンはステップ実行です。それぞれの部品の働きを確

[*3] ROC は Receiver Operating Characteristic の略で、元はレーダーの受信感度設定に用いられていたそうです。受信感度を上げれば調査対象を捉え損なうことは少なくなりますが、調査対象外の反応の多さに悩まされます。感度を下げれば静かになりますが、調査対象を捉え損なう可能性が高くなります。

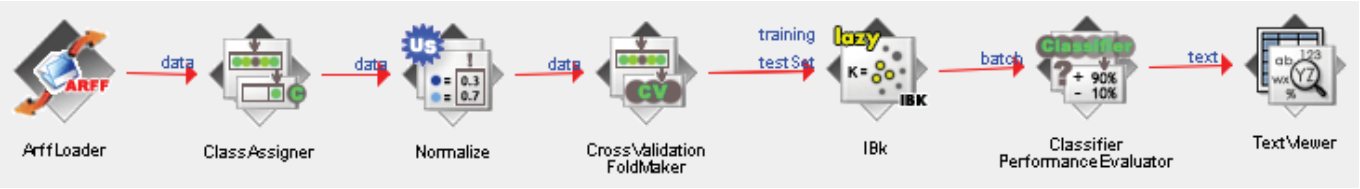

図 2.16　学習手順 firstFlow の完成図

認したいときに使います。

実行してステータスペインにエラーが出なければ、結果を確認します。TextViewer を右クリックして Show Results を選ぶと、図 2.17 のような結果が表示されます。全体の結果は、Correctly Classified Instances の後に正しく分類された事例数が表示され、その後の数字は正しく分類された事例数の割合です。結果の末尾は混同行列です。

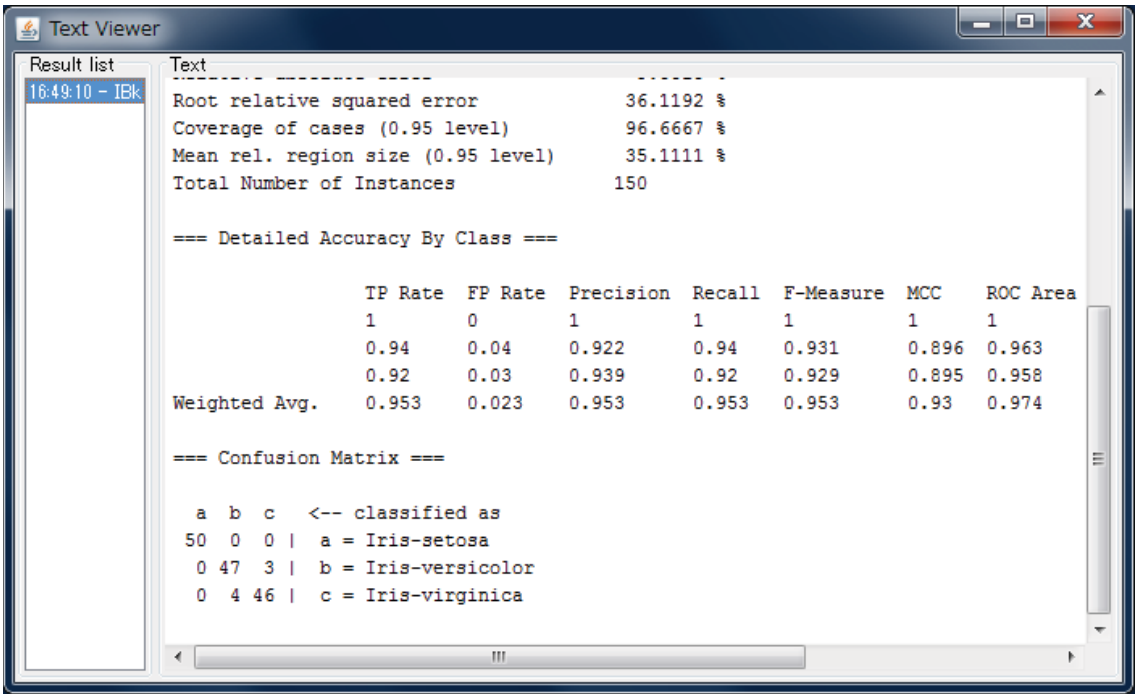

図 2.17　学習評価結果の表示

2.6　予備的実験のための環境

ここでは、Knowledge flow よりも手軽に学習実験が行える Explorer インタフェースの使い方を紹介します。Explorer（探検者）という名前の通り、あるデータに対してあるアルゴリズムが適用できるのか、学習時間はどれくらいかかりそうか、ある程度意味のある結果は得られそうか、などをアルゴリズムやパラメータを変化させて、手探りで動かしてみるには、最善のインタフェースです。

Knowledge flow で設定した各手順が既にタブで用意され、ある手順の設定が他の手順に適切に反映される[1]など、一通りの実験を終わらせるためのステップ数は Knowledge flow と比較して格段に少なくなります。

基本的な手順は図 2.1 で示したものと変わりませんが、それぞれの作業をタブを渡り歩いて行います。前節までで示した iris データを k-NN 法で識別し、交差確認法で評価するというプロセスなら、Preprocess タブと Clasify タブだけで終わります。

Preprocess タブで行う作業は、データを読み込み、前処理を行うところまでの手順をカバーします。前処理は、Filter 領域で適用するフィルターを指定します。

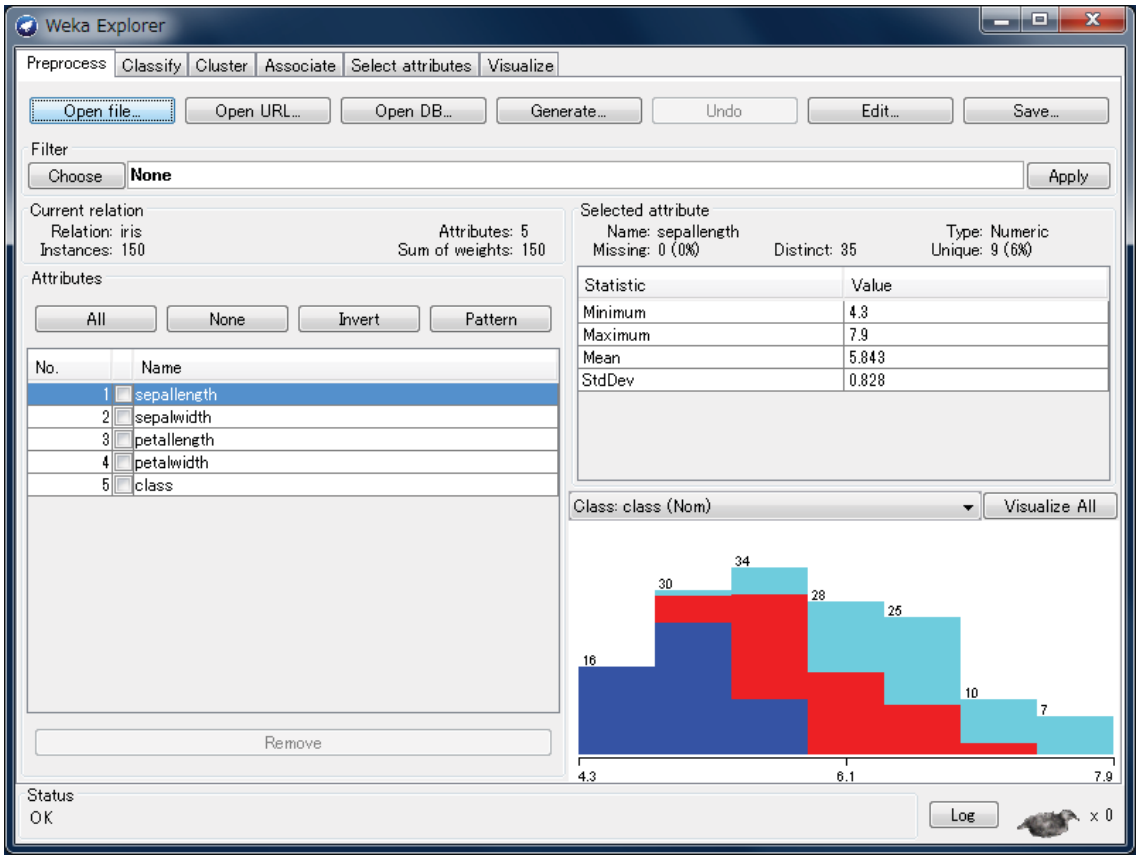

図 2.18　Explorer インタフェース:Preprocess

また、Visualize タブでは読み込んだデータの分布を任意の 2 次元を選んで表示させることができます。

Preprocess タブでデータを読み込むと、Classify タブや Cluster タブなど、実際に機械学習を行うためのタブが有効になります。Classify タブでは、Classifier 領域で適用する識別器を選択し、選択後にテキストボックスをクリックするとパラメータ設定画面が出てきます。評価基準の設定は Test options 領域で行います。それらの指定が終われば、start ボタンで学習・評価が行われ、右側の Classifier output 領域に

*1 例えば、データセットを読み込めば、そのデータセットに適用できる学習器のみが選択可能になります。

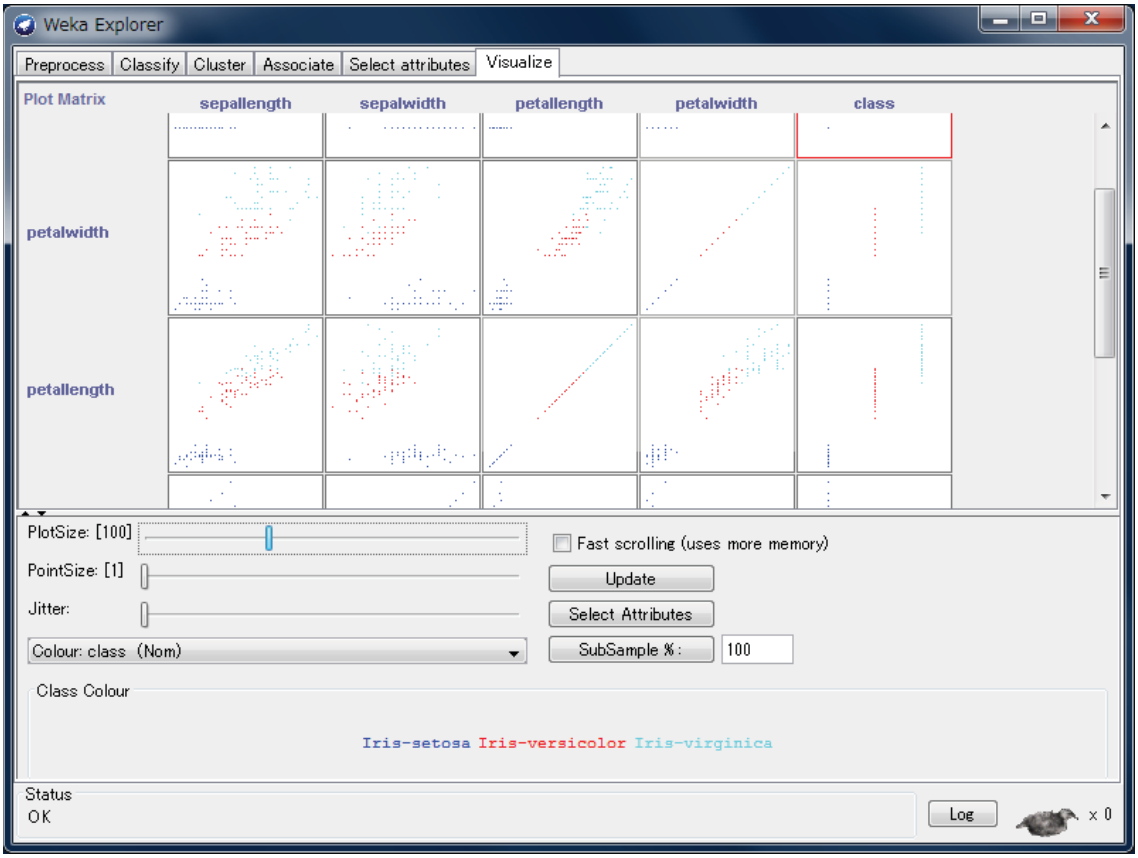

図 2.19　Explorer インタフェース:Visualize

結果が表示されます。

2.7　まとめ

本章では Weka の Knowledge Flow インタフェースを用いて、機械学習の典型的な手順を説明しました。また、Explorer インタフェースによる手軽な識別実験の方法を説明しました。以後、説明の中心となるのは学習アルゴリズムです。入力データの種類や学習の目的を変えたときに、どのような学習アルゴリズムを用いることができるのか、また、学習結果の評価はどのようにすればよいか、ということを 1 章で述べた分類に従って順に説明してゆきます。

なお、Weka の詳細については、ツールの作者らの著書 [5] を参照してください。

演習問題

1. Explorer インタフェースを用いて、2.2 節から 2.5 節で説明した内容の実験を行え。

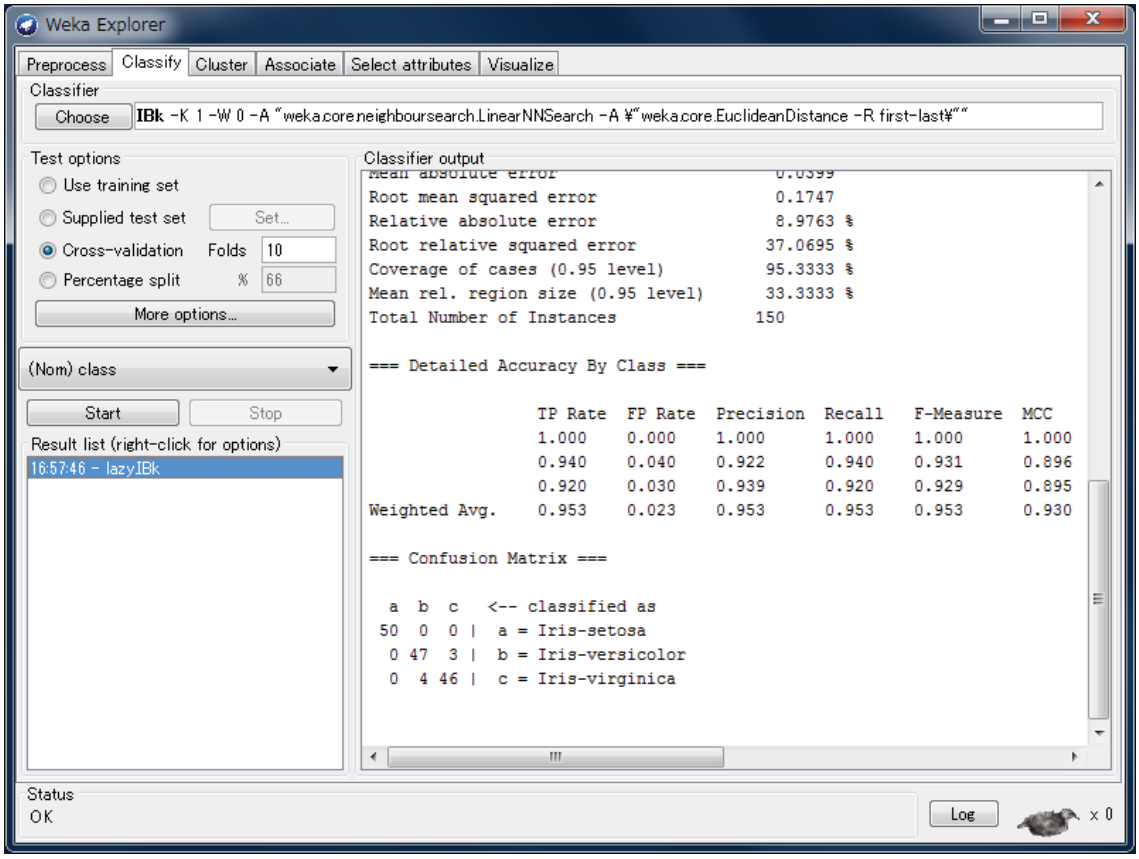

図 2.20　Explorer インタフェース:Classify

2. Knowledge Flow インタフェースの右から 3 つめのアイコン (Load a template layout) には、典型的なレイアウトが用意されている。このメニューからレイアウトのテンプレートを呼び出して、その動作を確認せよ。また、テンプレートに付けられたコメントを読んで、それぞれの部品の役割を理解せよ。