

Python による演習は、原則として「ML 演習番号.ipynb」ファイルが解答です。たとえば実践演習 1-1 の解答は「ML1-1.ipynb」です。

以下では、Weka を用いた演習の解答例を示します。

実践演習 3-1

iris.arff データでは、minNumObj=1, unpruned = True にすることで精度 100% が実現できます。一方、iris.2D.arff データでは 1 事例の誤りをどうしても消すことができません。これはデータ中に同じ特徴ベクトルに対して異なるクラスが正解として付与されているものがあるからです。

実践演習 3-2

minNumObj=2~4, unpruned = False にすることで精度 96% が実現できます。これに加えて useMDL-Correction（数値特徴を分割するときの基準）を False にすると、精度 96.67% が実現できます。

実践演習 3-3

minNumObj=21 にすることで精度 73% が実現できます。このときの木は、葉に近い purpose の特徴で広がっているのが複雑そうに見えるだけで、全体としては比較的単純です。minNumObj=25 にすると精度 72.1% に落ちますが、木はさらに単純になります。

実践演習 4-1

学習後、Classifier output ペインの Classifier model 以後が条件付き確率表です。(rainy,hot,high,TRUE) に対する yes の確率は、以下のようにして求められます。事前確率を掛けるのを忘れないように。また、すべてラプラス推定なので、事前に各値にたいして 1 事例あるものとして計算します。

```
-->y=(4/12)*(3/12)*(4/11)*(4/11)*(10/16)
y =
0.0068871

-->n=(3/8)*(3/8)*(5/7)*(4/7)*(6/16)
n =
0.0215242

--> y/(y+n)
ans =
0.2424055
```

なお、Weka の BayesNet を用いて学習させ、グラフ構造を表示させている画面から XML 形式でベイジアンネットワークを保存し、ベイジアンネットワークエディタで読み込んで値を設定することでも計算結果を確認できます（計算結果を同じにするためには、BayesNet 学習時に estimator (SimpleEstimator) のパラメータ alpha を 1 にする必要があります）。

実践演習 4-2

カテゴリカルデータに対しては、実践演習 4-1 と同じ結果が得られています。数値データに対しては、1 次元正規分布の平均と標準偏差が得られています。

実践演習 4-3

省略

実践演習 4-4

学習結果は、すべての特徴が play を親とする形のネットワーク（教科書 p.80 図 4.10(a)）になります。これは各特徴が独立であることを表現しているので、ナীবベイズと等価です。

実践演習 4-5

親ノードの最大値が 1 のときは、ナীবベイズ識別器と同じものが得られ、正解率は 72.0% となります。親ノードの最大値を 2 にすると、複雑なベイジアンネットワークが得られ、学習データに対する正解率は上がりますが、10-fold CV では 70.3% と下がってしまいます。