Rule based classifiers

- Examples:
  - if Refund==No and Marital_Status==Married \[\rightarrow\] Cheat==No
  - if Blood_Type==Warm (Lay_Eggs==Yes) \[\rightarrow\] Class=Birds
- LHS=rule antecedent or condition
- RHS=rule consequent

R1: (Give Birth = no) \& (Can Fly = yes) → Birds
R2: (Give Birth = no) \& (Live in Water = yes) → Fishes
R3: (Give Birth = yes) \& (Blood Type = warm) → Mammals
R4: (Give Birth = no) \& (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians
Rule based classifiers

A rule covers an case or instance, if the features satisfy the antecedent or condition.

The coverage of a rule on a dataset is the fraction of cases the rule covers.

The accuracy of a rule is the fraction of cases it covers for which it the consequent agrees with the label.

Ideal rules are rules with high coverage and high accuracy.

Rule Coverage and Accuracy

Coverage of a rule: – Fraction of records that satisfy the antecedent of a rule

Accuracy of a rule: – Fraction of records that satisfy both the antecedent and consequent of a rule (Status=Single)

No Coverage = 40%,  Accuracy = 50%

How does Rule-based Classifier Work?

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

lemur: warm, yes, no, no
turtle: cold, no, no, sometimes
dogfish shark: cold, yes, no, yes

A lemur triggers rule R3, so it is classified as a mammal
A turtle triggers both R4 and R5
A dogfish shark triggers none of the rules

A rule-based classifier is determined by a set of rules.

The rules are mutually exclusive if each case is covered by one and only one rule.

The rules are exhaustive if each case is covered by at least one rule.

A decision tree is an example of a rule-based classifier.
Rule based classifiers

**Classification Rules**

(Restruct=Yes) => No

(Restruct=No, Marital Status={Single, Divorced}, Taxable Income<80K) => No

(Restruct=No, Marital Status={Single, Divorced}, Taxable Income>80K) => Yes

(Restruct=No, Marital Status={Married}) => No

---

**Possible issues**

- Rules may not be mutually exclusive: they can be ordered, or use majority rule.
- Rules may not be exhaustive: use a default class.

**Finding rules**

- Try to extract directly from data (see PRIM in ESL).
- Extracted from a decision tree (some tree software such as C4.5 does this).

---

We can simplify the Married==No rule.
Learning a rule

Removing a discovered rule to find additional rules

**Rule based classifiers**

### Evaluating a rule
- **Accuracy** \( R = 1 - \text{Misclassification Error} \)
- **Laplace** \( R = \frac{n_c(R)}{n(R) + k} \) where \( k \) is the number of classes . . .
- **M-estimate** \( R = \frac{n_c(R) + \alpha p_i(R)}{n(R) + \alpha} \) where \( i(R) \) is the majority class and \( p_i \) is a set of prior probabilities on each of the \( k \) classes and \( \alpha > 0 \) is some prior weight.
- The Laplace rule is effectively a posterior estimate of the probability of an instance belonging to a class based on a prior probability . . .

**Properties of rule-based classifiers**
- Flexible like decision trees (even more flexible).
- As expressive as decision trees.
- Easy to classify new cases (once ordering established).
- Can be extracted from decision trees.
- Disadvantage: there is no clear loss function they are optimizing . . .
Nearest-neighbour classifiers

For each $x$ find $k$ nearest neighbours in the cases data matrix $X$.

Let $k^*(x)$ denote the majority rule among the $k$-nearest neighbours.

Assign $f(x) = k^*(x)$.

Easy to describe, but can be expensive to compute many nearest neighbours . . .

Choice of $k$

Nearest neighbour classifier $k = 1$
Nearest neighbour classifier $k = 3$

Nearest neighbour classifier $k = 10$

Nearest neighbour classifier $k = 20$

Linear Discriminant Analysis using $(\text{sepal.width}, \text{sepal.length})$
Naive Bayes classifiers

Bayesian classifiers
- In LDA / QDA, we used Bayes’ Theorem to classify.
- Formally, given a partition $E_1, \ldots, E_k$ Bayes’ Theorem says
  \[ P(E_i|A) = \frac{P(A|E_i)P(E_i)}{P(A)} = \frac{P(A|E_i)P(E_i)}{\sum_{j=1}^{k} P(A|E_j)P(E_j)} \]
- We applied this to the events $E_i = \{Y = c\}$ and $A = \{X = x\}$ yielding
  \[ P(Y = c|X = x) = \frac{P(X = x|Y = c)P(Y = c)}{\sum_{j=1}^{k} P(X = x|Y = j)P(Y = j)} \]
  \[ \propto P(X = x|Y = c)P(Y = c) \]

Naive Bayes classifiers
- The Naive Bayes classifier assumes that the features are independent given the class label. Therefore,
  \[ P(Y = c|X_1 = x_1, \ldots, X_p = x_p) \]
  \[ \propto \left( \prod_{i=1}^{p} P(X_i = x_i|Y = c) \right) P(Y = c) \]
- Fits a discriminant model within each feature.
- For continuous features, typically a 1-dimensional QDA model is used (i.e. Gaussian within each class).

Laplace smoothing
- For discrete features, it uses a discrete estimate
  \[ P(X_j = l|Y = c) = \frac{\# \{i : X_{ij} = l, Y_i = c\}}{\# \{Y_i = c\}}. \]
- The discrete estimates have the problem that if no instances of class $c$ had feature $X_j = l$ then any new instances $x$ with $X_j = l$ cannot be classified to class $c$.
- One solution: use the Laplace smoothed probabilities
  \[ P(X_j = l|Y = c) = \frac{\# \{i : X_{ij} = l, Y_i = c\} + \alpha}{\# \{Y_i = c\} + \alpha \cdot k}. \]

A Naive Bayes classifier applies Bayes’ Theorem based on several features $x = (x_1, \ldots, x_p)$
\[ P(Y = c|X_1 = x_1, \ldots, X_p = x_p) \]
\[ \propto P(X_1 = x_1, \ldots, X_p = x_p|Y = c)P(Y = c) \]
- To use the classifier, we need to estimate the RHS above.

For continuous features, typically a 1-dimensional QDA model is used (i.e. Gaussian within each class).
To use the classifier, we need to estimate the $P(X = x|Y = c)$ and $P(Y = c)$.
### Naive Bayes classifiers

#### Properties
- Features may not be conditionally independent, which may force some bias.
- Continuous features may not be Gaussian (LDA suffers from this also . . .)
- Can be slow to predict new instances (at least e1071’s implementation).