CIS8695

# Naïve Bayes; Support Vector Machine; and KNN

Ling Xue

# Naïve Bayes: The Basic Idea

For a given new record to be classified, find other records like it (i.e., same values for the predictors)

What is the prevalent class among those records?

Assign that class to your new record

# Usage

- Requires categorical variables
- Numerical variable must be binned and converted to categorical
- Can be used with very large data sets
- Example:  Spell check programs assign your misspelled word to an established “class” (i.e., correctly spelled word)

# Exact Bayes Classifier

Relies on finding other records that share same predictor values as record-to-be-classified.

Want to find "probability of belonging to class *C*, given specified values of predictors."

Even with large data sets, may be hard to find other records that **exactly match** your record, in terms of predictor values.

# Naïve Bayes

- Assume independence of predictor variables (within each class)
- Use multiplication rule
- Find same probability that record belongs to class C, given predictor values, without limiting calculation to records that share all those same values

# Calculations

1. Take a record, and note its predictor values
2. Find the probabilities those predictor values occur across all records in C1
3. Multiply them together, then by proportion of records belonging to C1
4. Same for C2, C3, etc.
5. Prob. of belonging to C1 is value from step (3) divide by sum of all such values C1 … Cn
6. Assign the record to the class with the highest probability for this set of predictor values

# Example: Financial Fraud

Target variable:  Audit finds fraud, no fraud

Predictors:

- Prior pending legal charges (yes/no)
- Size of firm (small/large)

| Charges? | Size | Outcome |
|---|---|---|
| y | small | truthful |
| n | small | truthful |
| n | large | truthful |
| n | large | truthful |
| n | small | truthful |
| n | small | truthful |
| y | small | fraud |
| y | large | fraud |
| n | large | fraud |
| y | large | fraud |

$$P(Fraud|Small, y) = \frac{P(Fraud, Small, y)}{P(Small, y)}$$

$$= \frac{P(Small, y|Fraud) \times P(Fraud)}{P(Small, y)}$$

$$= \frac{P(Small, y|Fraud) \times P(Fraud)}{P(Small, y|Fraud) \times P(Fraud) + P(Small, y|Truthful) \times P(Truthful)}$$

$$\rightarrow \frac{P(Small|Fraud) \times P(y|Fraud) \times P(Fraud)}{\begin{bmatrix} P(Small|Fraud) \times P(y|Fraud) \times P(Fraud) \\ +P(Small|Truthful) \times P(y|Truthful) \times P(Truthful) \end{bmatrix}}$$

# Exact Bayes Calculations

**Goal:** classify (as “fraudulent” or as “truthful”) a small firm with charges filed

There are 2 firms like that, one fraudulent and the other truthful

P(fraud | charges=y, size=small) = ½ = 0.50

Note: calculation is limited to the two firms matching those characteristics

# Naïve Bayes Calculations

Same goal as before

Compute 2 quantities:

Proportion of "charges = y" among frauds, times proportion of "small" among frauds, times proportion frauds = 3/4 * 1/4 * 4/10 = 0.075

Prop "charges = y" among frauds, times prop. "small" among truthfuls, times prop. truthfuls = 1/6 * 4/6 * 6/10 = 0.067

P(fraud | charges, small) = 0.075/(0.075+0.067)

= 0.53

# Naïve Bayes, cont.

- Note that probability **estimate** does not differ greatly from **exact**
- All records are used in calculations, not just those matching predictor values
- This makes calculations practical in most circumstances
- Relies on assumption of independence between predictor variables within each class

# Independence Assumption

- Not strictly justified (variables often correlated with one another)
- Often “good enough” – ranking of probabilities is more important than unbiased estimate of actual probabilities

# Naïve Bayes in R

- Use package `e1071`
- Function `naiveBayes`
- See Table 8.4 for code for running Naïve Bayes
  - Includes code for binning numeric variables into categories, which is required for NB

# Advantages

- Handles purely categorical data well
- Works well with very large data sets
- Simple & computationally efficient

# Shortcomings

- Requires large number of records
- Problematic when a predictor category is not present in training data
  - Assigns 0 probability of response, ignoring information in other variables

# On the other hand…

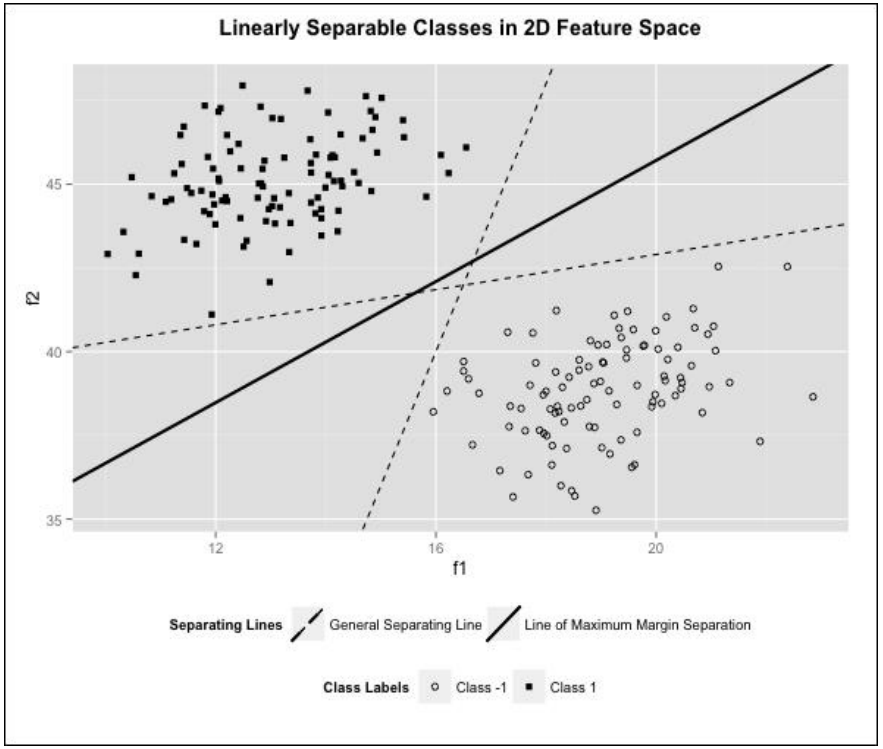
- Probability rankings are more accurate than the actual probability estimates

  Good for applications using lift (e.g. response to mailing), less so for applications requiring probabilities (e.g. credit scoring)

# Support Vector Machine: Intuition

- **Hyperplane:** Classification task is linearly separable. However, the dotted lines and solid line are just among an infinite number of possible linear solutions.

- A point falls on the wrong side of the linear separator is higher for the dotted line than the solid line, which means that the solid line has a higher margin of safety for classification.



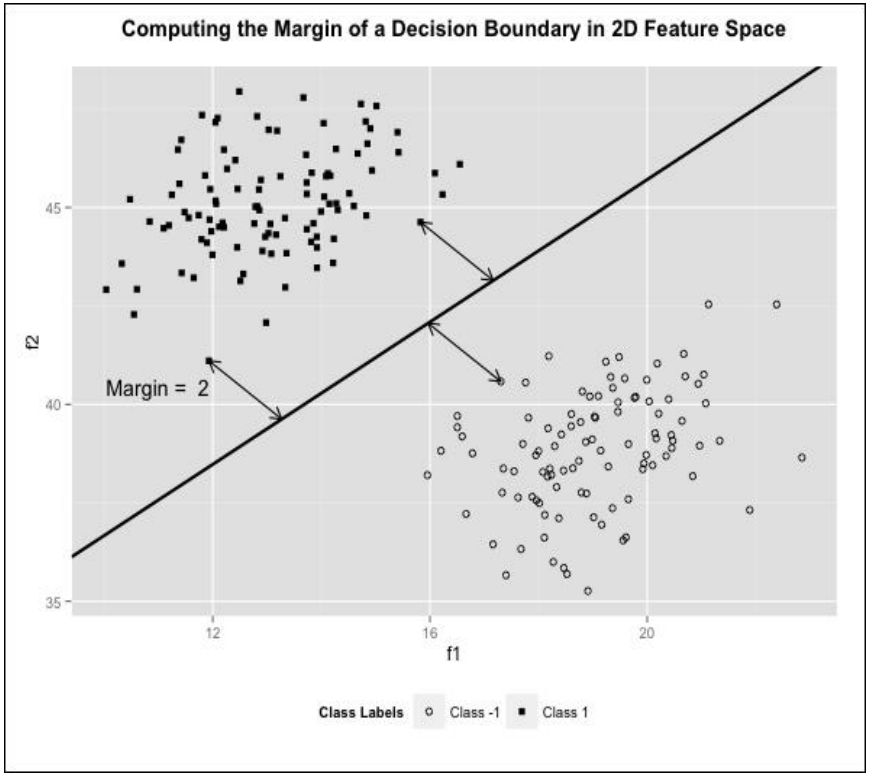

**Maximal margin hyperplane:** *linear separators with the largest possible margin and the support vectors the ones touching the safety margin region on both sides*

# Support Vector Machine: Intuition

Data need to not only be correctly classified, but also lie at least *M* units away from the separating hyperplane

The goal is to maximize this distance *M* by appropriately choosing the linear model coefficients

We define the perpendicular vectors from the points that lie on the margin to the separating hyperplane as the **support vectors** (3 SVs in the plot)




Support vector classifier (linear) is in the form:

$$y(x) = \beta_0 + \sum_{s \in S} \alpha_s \langle x, x_s \rangle$$

# Use Support Vector Machine to Classify

$$y(x) = \beta_0 + \sum_{s \in S} \alpha_s K\langle x, x_s \rangle$$

- **K** is a **kernel function**, which can be linear (support vector classifier) or non-linear (support vector machine)
- $X_s$ are support vectors; X are the input features of the data points to be predicted/classified; y(X) is the prediction/classification outcomes
- The general logic is to see how X are located relative to $X_s$, or in other words, how X are similar to $X_s$. So kernel functions are also known as similarity functions.

# Use Support Vector Machine to Classify

$$y(x) = \beta_0 + \sum_{s \in S} \alpha_s K\langle x, x_s \rangle$$

- Possible kernels:
  - linear, no transformation, $K(x_i, x_j) = x_i \bullet x_j$
  - polynomial, where $d$ = degree of polynomial, $K(x_i, x_j) = (\gamma x_i \bullet x_j + c)^{\mathrm{d}}$
  - radial basis function, $K(x_i, x_j) = e(-\gamma |x_i - x_j|^2)$
  - sigmoid function, $K(x_i, x_j) = \tanh(\gamma x_i \bullet x_j + c)$

# KNN: Basic Idea

For a given record to be classified, identify nearby records

"Near" means records with similar predictor values $X_1$, $X_2$, ... $X_p$

Classify the record as whatever the predominant class is among the nearby records (the "neighbors")

# How to measure “nearby”?

The most popular distance measure is
**Euclidean distance**

$$\sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + \cdots + \left(x_p - u_p\right)^2}$$

- Typically, predictor variables are first normalized (= standardized) to put them on comparable scales
- Use `preProcess()` from `caret` package to normalize
- Otherwise, metrics with large scales dominate

# Choosing k

*K* is the number of nearby neighbors to be used to classify the new record

- *K*=1 means use the single nearest record
- *K*=5 means use the 5 nearest records

Typically choose that value of *k* which has lowest error rate in validation data

# Low *k* vs. High *k*

Low values of *k* (1, 3, …) capture local structure in data (but also noise)

High values of *k* provide more smoothing, less noise, but may miss local structure

**Note:** the extreme case of k = n (i.e., the entire data set) is the same as the "naïve rule" (classify all records according to majority class)

# Example: Riding Mowers

**Data:** 24 households classified as owning or not owning riding mowers

**Predictors:** Income, Lot Size

| Income | Lot_Size | Ownership |
|---|---|---|
| 60.0 | 18.4 | owner |
| 85.5 | 16.8 | owner |
| 64.8 | 21.6 | owner |
| 61.5 | 20.8 | owner |
| 87.0 | 23.6 | owner |
| 110.1 | 19.2 | owner |
| 108.0 | 17.6 | owner |
| 82.8 | 22.4 | owner |
| 69.0 | 20.0 | owner |
| 93.0 | 20.8 | owner |
| 51.0 | 22.0 | owner |
| 81.0 | 20.0 | owner |
| 75.0 | 19.6 | non-owner |
| 52.8 | 20.8 | non-owner |
| 64.8 | 17.2 | non-owner |
| 43.2 | 20.4 | non-owner |
| 84.0 | 17.6 | non-owner |
| 49.2 | 17.6 | non-owner |
| 59.4 | 16.0 | non-owner |
| 66.0 | 18.4 | non-owner |
| 47.4 | 16.4 | non-owner |
| 33.0 | 18.8 | non-owner |
| 51.0 | 14.0 | non-owner |
| 63.0 | 14.8 | non-owner |

# Finding nearest neighbors in R

Library `FNN` provides a list of neighbors

Library `class` allows numerical output

See Table 7.2 for code using `knn` from `FNN` library; compares each record from validation* set to k nearest records in training

*termed the `test` set in R

# Finding nearest neighbors in R

Use library `caret` to get accuracy of different values of k, applied to validation data (see next slide for code)

Output

```
> accuracy.df
    k accuracy
1   1      0.7
2   2      0.7
3   3      0.8
4   4      0.9
5   5      0.8
6   6      0.9
7   7      0.9
8   8      1.0
9   9      0.9
10 10      0.9
11 11      0.9
12 12      0.8
13 13      0.4
14 14      0.4
```

When the most-accurate k is an even number (here it's 8), it is possible for ties to occur in classifying new records. R breaks ties randomly.

# Using K-NN for Prediction (for Numerical Outcome)

- Instead of “majority vote determines class” use average of response values
- May be a weighted average, weight decreasing with distance

# Advantages

- Simple
- No assumptions required about Normal distribution, etc.
- Effective at capturing complex interactions among variables without having to define a statistical model

# Shortcomings

- Required size of training set increases exponentially with # of predictors, *p*

  This is because expected distance to nearest neighbor increases with *p* (with large vector of predictors, all records end up "far away" from each other)

- In a large training set, it takes a long time to find distances to all the neighbors and then identify the nearest one(s)
- These constitute "curse of dimensionality"