

Naïve Bayes classifiers

Decision trees

Lecture 2b

Naive Bayes classifiers: motivation

- Consider n labeled text documents
 - $Y = \{0,1\}$, $0 =$ "Science fiction", $1 =$ "Comedy"
 - $X = \{X_1, \dots, X_{100}\}$ does the document contain the keyword ($0=\text{No}, 1=\text{Yes}$)
 - X_1 corr. "space", X_2 corr. "fun",...
- Want to classify a new document



Naive Bayes classifiers: motivation

Idea: use Bayes classifier

$$p(Y = y|X) = \frac{P(X|Y = y)P(Y = y)}{\sum_j P(X|Y = y_j)P(Y = y_j)}$$

Chance of observing a
given combination of
words in science
fiction

Proportion of science
fiction documents

Naive Bayes classifiers: motivation

- Attempt 1:
 - Model $P(X = (x_1, \dots, x_p) | Y = y_i)$ and $P(Y = y_i)$ as unknown parameters
 - Use data to derive those with Maximum Likelihood
 - Classify by use of the posterior distribution
- How many parameters?
 - How many different combinations of X ? 2^p
 - Amount of $P(X = (x_1, \dots, x_p) | Y = y_i)$ is $2 * 2^p - 2$
 - Probabilities for each Y sum up to one
- If $p = 100$, 10^{30} parameters need to be estimated →ouch!

Naive Bayes classifiers

- Naive Bayes assumption: conditional independence

$$P(X = (x_1, \dots, x_p) | Y = y) = \prod_{i=1}^p P(X_i = x_i | Y = y)$$

- How many parameters now?
 - $P(X_i = x_i | Y = y), i = 1, \dots, p, x_i = \{0,1\}, y = \{0,1\}$ $2 * p$
- Is Naive Bayes assumption always valid?
 - $P(\text{Space, ship} | \text{SciFi}) = P(\text{Space} | \text{SciFi}) * P(\text{Ship} | \text{SciFi})$?

Naive Bayes classifiers - discrete inputs

- Given $D = \{(X_{m1}, \dots, X_{mp}, Y_m), m = 1, \dots, n\}$
- Assume $X_i \in \{x_1, \dots, x_J\}, i = 1, \dots, p, Y \in \{y_1, \dots, y_K\}$
- Denote $\theta_{ijk} = p(X_i = x_j | Y = y_k)$
 - How many parameters? $(J - 1)Kp$
- Denote $\pi_k = p(Y = y_k)$
- **Maximum likelihood:** assume θ_{ijk} and π_k are constants
 - $\hat{\theta}_{ijk} = \frac{\#\{X_i=x_j \& Y=y_k\}}{\#\{Y=y_k\}}$
 - $\hat{\pi}_k = \frac{\#\{Y=y_k\}}{n}$
 - Classification using 0-1 loss: $\hat{Y} = \arg \max_y p(Y = y | X)$

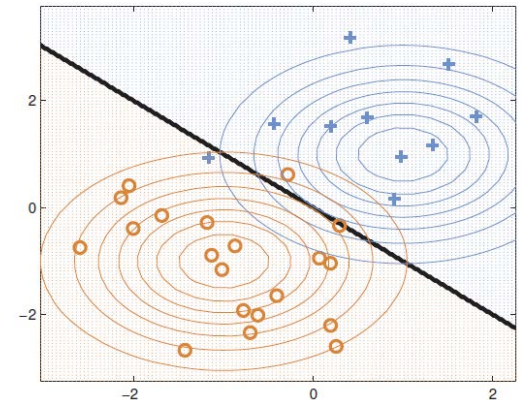
Naive Bayes classifiers - discrete inputs

- **Example** Loan decision
 - Classify a person: Home Owner=No, Single=Yes

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naive Bayes – continuous inputs

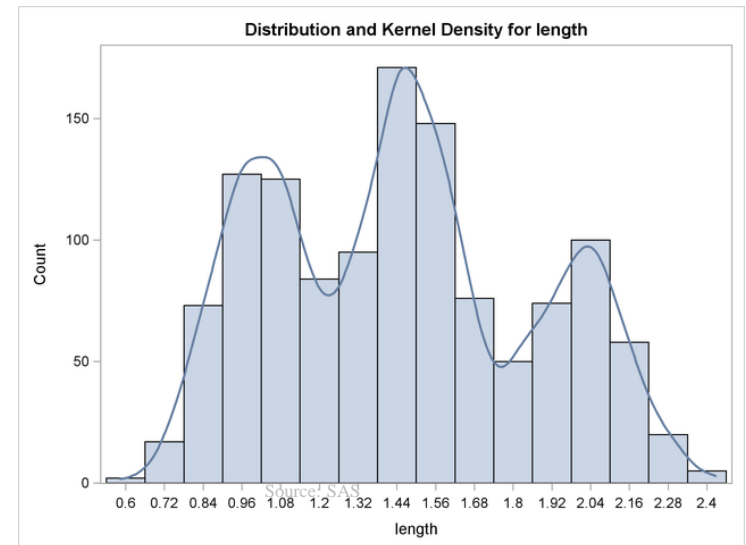
- X_i are continuous
- **Assumption A:** $x_j|y = C$ are univariate Gaussian
 - $p(x_j|y = C_i, \theta) = N(x_j|\mu_{ij}, \sigma_{ij}^2)$
- Therefore $p(\mathbf{x}|y = C_i, \theta) = N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
 - $\boldsymbol{\Sigma}_i = \text{diag}(\sigma_{i1}^2, \dots, \sigma_{ip}^2)$



- **Naive bayes is a special case of LDA (given A)**
 - \rightarrow MLE are means and variances (per class)

Naive Bayes – continuous inputs

- **Assumption B:** $p(x_j|y = C)$ are unknown functions of x_j that can be estimated from data
 - Nonparametric density estimation (kernel for ex.)
- 1. Estimate $p(X_i = x_j|Y = y_k)$ using nonparametric methods
- 2. Estimate $p(Y = y_k)$ as class proportions
- 3. Use Bayes rule and 0-1 loss to classify



Naive Bayes in R

- naiveBayes in package **e1071**

Example: Satisfaction of householders with their present housing circumstances

```
library(MASS)
library(e1071)
n=dim(housing)[1]
ind=rep(1:n, housing[,5])
housing1=housing[ind,-5]

fit=naiveBayes(Sat~., data=housing1)
fit

Yfit=predict(fit, newdata=housing1)
table(Yfit,housing1$Sat)
```

```
> table(Yfit,housing1$Sat)
```

Yfit	Low	Medium	High
Low	294	162	144
Medium	20	23	20
High	253	261	504

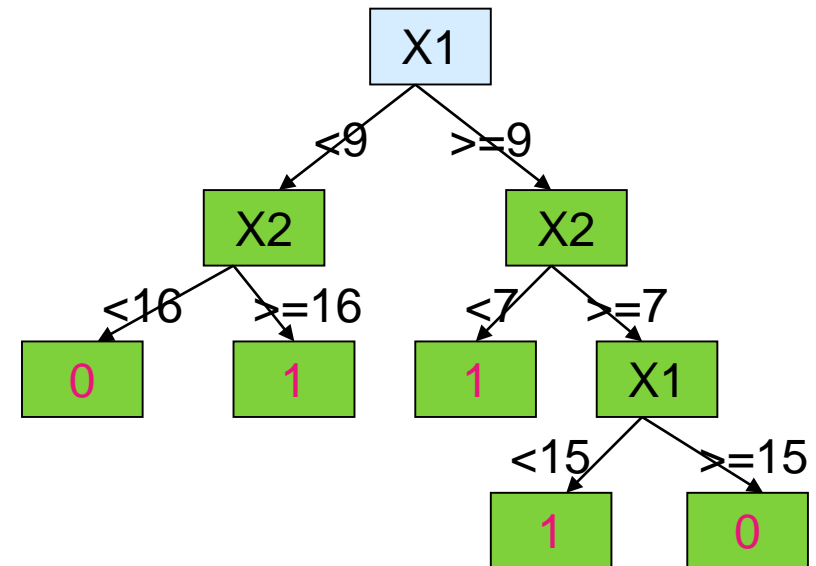
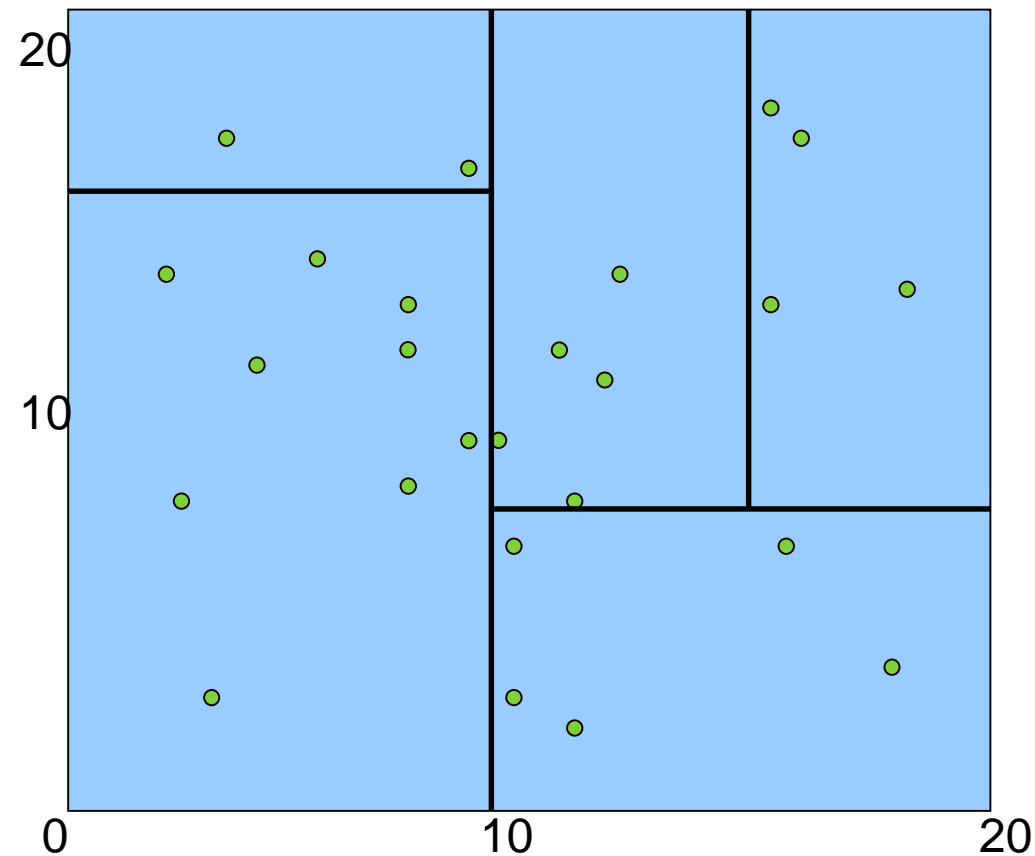
Decision trees

Idea

Split the domain of feature set into the set of hypercubes (rectangles, cubes) and define the target value to be constant within each hypercube

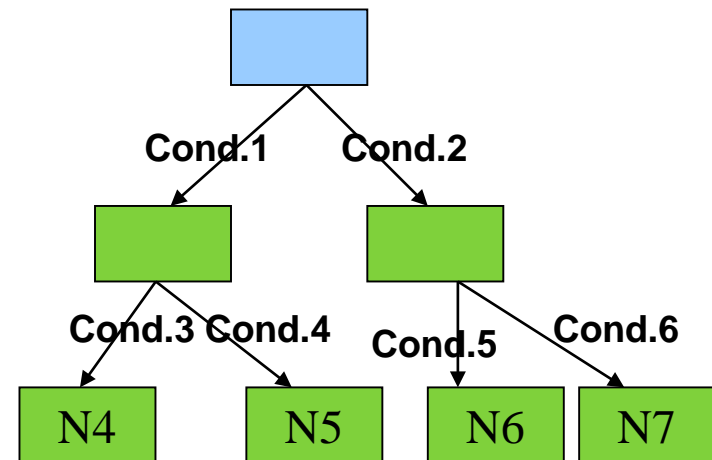
- Regression trees:
 - Target is a continuous variable
- Classification trees
 - Target is a class (qualitative) variable

Classification tree toy example

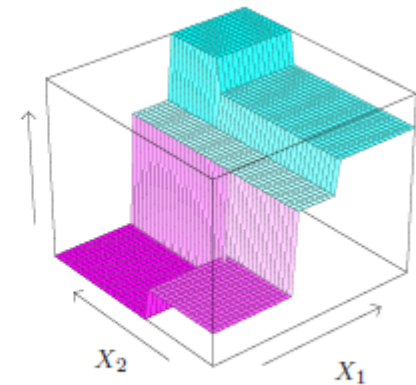
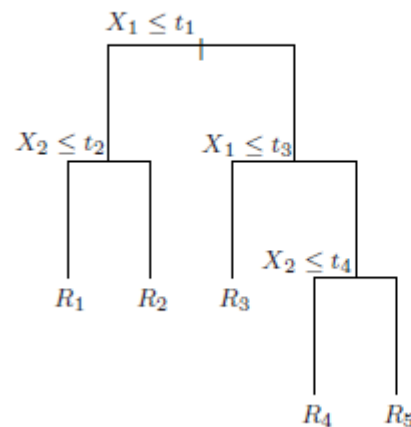
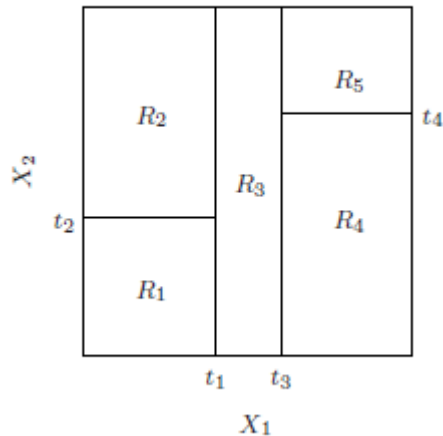


Definitions

- Root node
- Nodes
- Leaves (terminal nodes)
- Parent node, child node
- Decision rules
- A value is assigned to the leaves



Regression tree toy example



A classification problem

Create a classification tree that would describe the following patterns

ID	x1	x2	x3	x4	x5	x6	x7	y
Name	Body temperature	Skin cover	Gives birth	Aquatic creature	Aerial creature	Has legs	Hibernates	Class label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	non-mammal
salmon	cold-blooded	scales	no	yes	no	no	no	non-mammal
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	non-mammal
komodo	cold-blooded	scales	no	no	no	yes	no	non-mammal
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	non-mammal
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
shark	cold-blooded	scales	yes	yes	no	no	no	non-mammal
turtle	cold-blooded	scales	no	semi	no	yes	no	non-mammal
penguin	warm-blooded	feathers	no	semi	no	yes	no	non-mammal
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	non-mammal
salamander	cold-blooded	none	no	semi	no	yes	yes	non-mammal

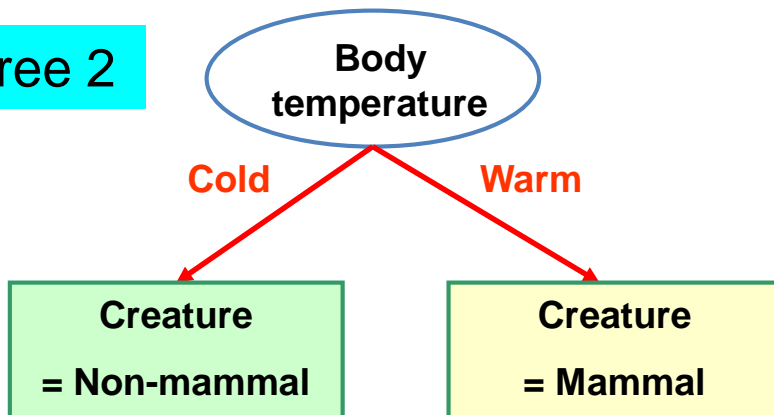
Several solutions

Tree 1

Creature =
Non-mammal

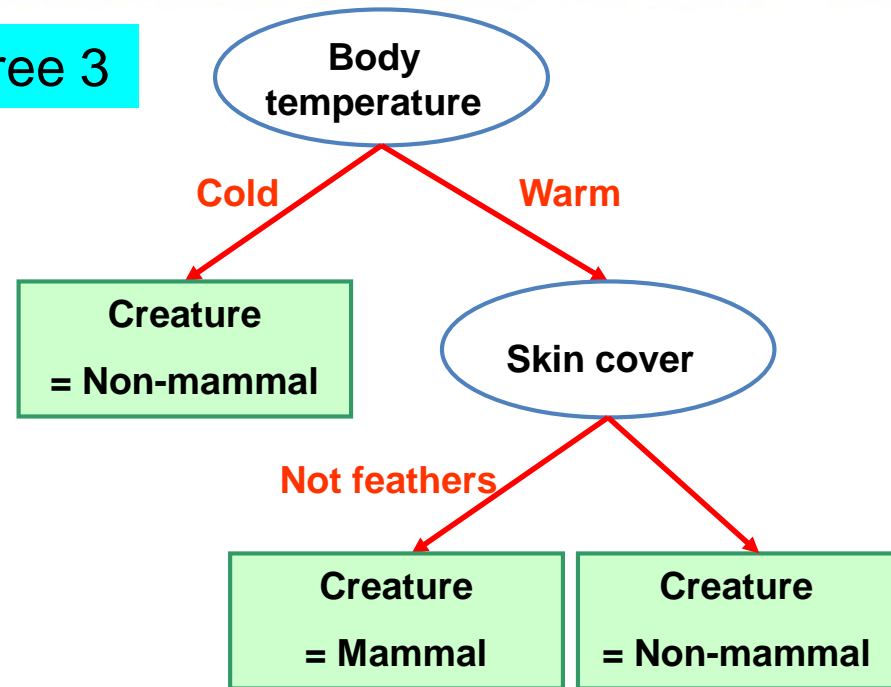
Large misclassification rate!

Tree 2



A lower misclassification rate

Tree 3



Zero misclassification

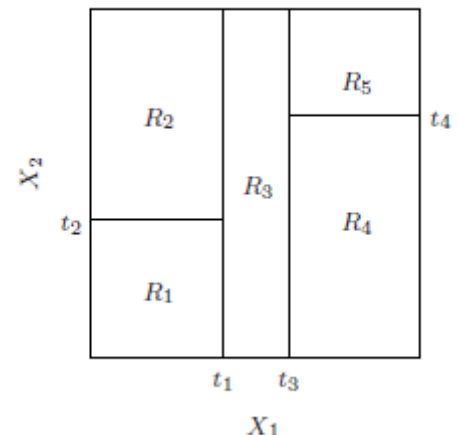
Green boxes represent pure nodes = nodes where observed values are the same

Decision trees

- A tree $T = \langle r_i, s_{r_i}, R_j, i = 1 \dots S, j = 1 \dots L \rangle$
 - $x_{r_i} \leq s_{r_i}$ splitting rules (conditions), S - their amount
 - R_j -terminal nodes, L - their amount
 - labels μ_j in each terminal node

Model:

- $Y|T$ for R_j comes from exponential family with mean μ_j
- Fitting by MLE:
 - Step 1: Finding optimal tree
 - Step 2: Finding optimal labels in terminal nodes



Decision trees

Example:

- **Normal model** leads to regression trees
 - Objective: MSE
- **Multinoulli model** leads to classification trees
 - Objective: cross-entropy (deviance)

Classification trees

- Target is categorical
- Classification probability $p_{mk} = p(Y = k|X \in R_m)$ is estimated for every class in a node
- How to estimate p_{mk} for class k and node R_m ?

Class proportions

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

- For any node (leave), a label can be assigned

$$k(m) = \arg \max_k \hat{p}_{mk}$$

Classification trees

- Impurity measure $Q(R_m)$
 - R_m is a tree node (region)
 - Node can be split unless it is pure

Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$

Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^K \hat{p}_{mk} (1 - \hat{p}_{mk})$

Cross-entropy or deviance: $-\sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}.$

- Note: In many sources, **deviance** is $Q(R_m) N(R_m)$

Example: Cross –entropy is MLE of $Y_j | T \sim \text{Multinomial}(p_{j1}, \dots, p_{jc})$

Fitting regression trees: CART

Step 1: Finding optimal tree: grow the tree in order to minimize global objective

1. Let C_0 be a hypercube containing all observations
2. Let queue $C = \{C_0\}$
3. Pick up some C_j from C and find a variable X_j and value s that split C_j into two hypercubes

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j > s\}$$

and solve

$$\min_{j,s} [N_1 Q(R_1) + N_2 Q(R_2)]$$

4. Remove C_j from C and add R_1 and R_2
5. Repeat 3-4 as many times as needed (or until each cube has only 1 observation)

CART: comments

- Greedy algorithm (optimal tree is not found)
- The largest tree will interpolate the data → large trees = **overfitting** the data
- Too small trees=**underfitting** (important structure may not be captured)
- Optimal tree length?

Optimal trees

- Postpruning

Weakest link pruning:

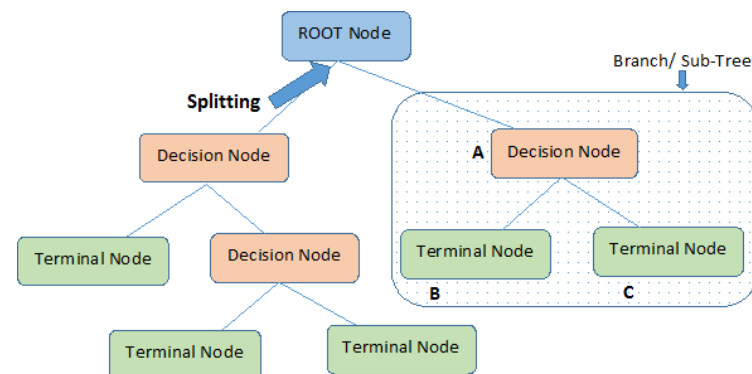
1. Merge two leaves that have smallest $N(\text{parent}) * Q(\text{parent}) - N(\text{leaf1})Q(\text{leaf1}) - N(\text{leaf2})Q(\text{leaf2})$

2. For the current tree T, compute

$$I(T) = \sum_{R_i \in \text{leaves}} N(R_i)Q(R_i) + \alpha|T|$$

$$|T| = \# \text{leaves}$$

3. Repeat 1-2 until the tree with one leaf is obtained
4. Select the tree with smallest $I(T)$



Source: <http://www.analyticsvidhya.com>

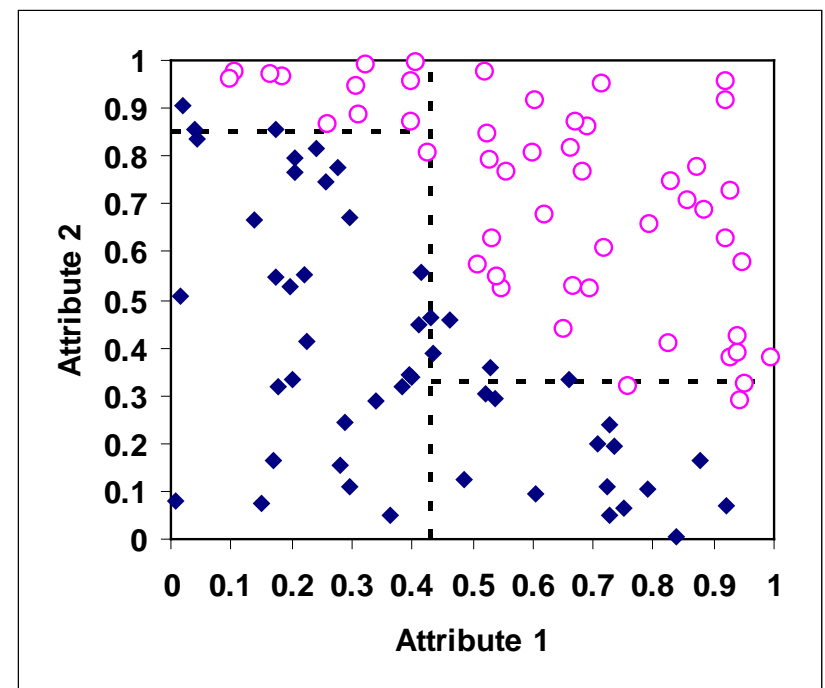
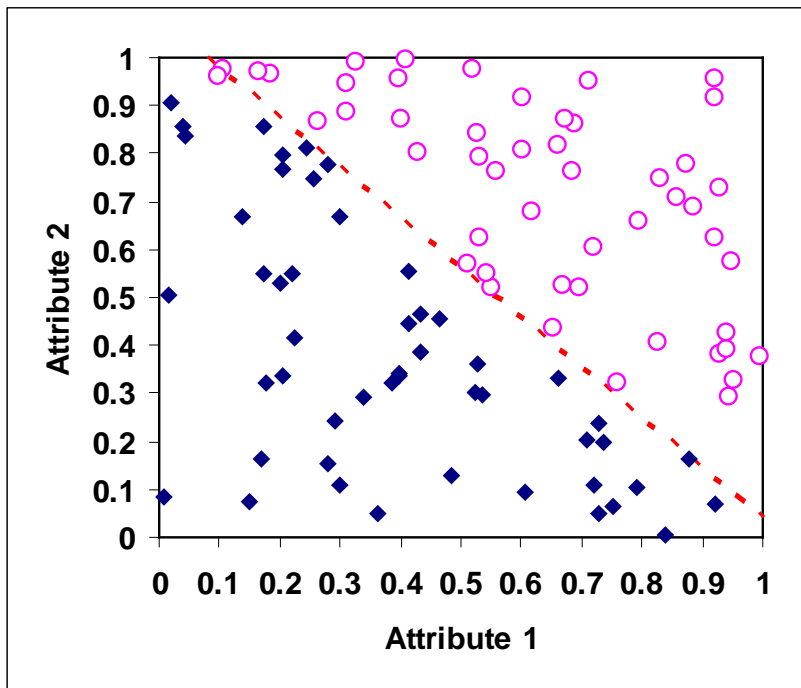
How to find the optimal α ? Cross validation!

Decision trees: comments

- Similar algorithms work for regression trees – replace $N \cdot Q(R)$ by $SSE(R)$
- Easy to interpret
- Easy to handle all types of features in one model
- **Automatic variable selection**
- Relatively robust to outliers
- Handle large datasets
- Trees have high variance: a small change in response \rightarrow totally different tree
- Greedy algorithms \rightarrow fit may be not so good
- Lack of smoothness

Decision trees: issues

- Large trees may be needed to model an easy system:



Decision trees in R

- **tree** package

- Alternative: **rpart**

`tree(formula, data, weights, control, split = c("deviance", "gini"), ...)`
`print()`, `summary()`, `plot()`, `text()`

Example: breast cancer as a function av biological measurements

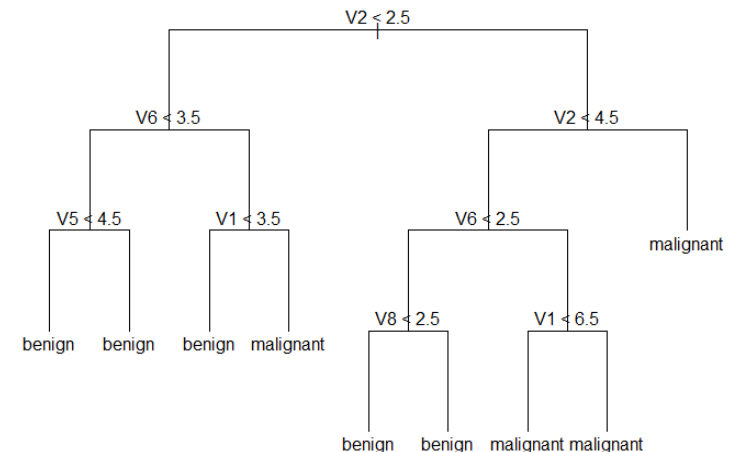
```
library(tree)
n=dim(biopsy)[1]
fit=tree(class~., data=biopsy)
plot(fit)
text(fit, pretty=0)
fit
summary(fit)
```

Decision trees in R

- Adjust the splitting in the tree with *control* parameter (leaf size for ex)

```
> fit
node), split, n, deviance, yval, (yprob)
* denotes terminal node

1) root 683 884.400 benign ( 0.650073 0.349927 )
2) v2 < 2.5 418 108.900 benign ( 0.971292 0.028708 )
4) v6 < 3.5 395 25.130 benign ( 0.994937 0.005063 )
8) v5 < 4.5 389 0.000 benign ( 1.000000 0.000000 ) *
9) v5 > 4.5 6 7.638 benign ( 0.666667 0.333333 ) *
5) v6 > 3.5 23 31.490 benign ( 0.565217 0.434783 )
10) v1 < 3.5 11 0.000 benign ( 1.000000 0.000000 ) *
11) v1 > 3.5 12 10.810 malignant ( 0.166667 0.833333 ) *
3) v2 > 2.5 265 217.900 malignant ( 0.143396 0.856604 )
6) v2 < 4.5 90 120.300 malignant ( 0.388889 0.611111 )
12) v6 < 2.5 30 27.030 benign ( 0.833333 0.166667 )
24) v8 < 2.5 19 0.000 benign ( 1.000000 0.000000 ) *
25) v8 > 2.5 11 15.160 benign ( 0.545455 0.454545 ) *
13) v6 > 2.5 60 54.070 malignant ( 0.166667 0.833333 )
26) v1 < 6.5 28 35.160 malignant ( 0.321429 0.678571 ) *
27) v1 > 6.5 32 8.900 malignant ( 0.031250 0.968750 ) *
7) v2 > 4.5 175 30.350 malignant ( 0.017143 0.982857 ) *
```



```
> summary(fit)
```

Classification tree:

```
tree(formula = class ~ ., data = biopsy)
```

Variables actually used in tree construction:

```
[1] "v2" "v6" "v5" "v1" "v8"
```

Number of terminal nodes: 9

Residual mean deviance: 0.1603 = 108 / 674

Misclassification error rate: 0.03221 = 22 / 683

Decision trees in R

- Misclassification results

```
Yfit=predict(fit, newdata=biopsy, type="class")  
table(biopsy$class,Yfit)
```

```
> table(biopsy$class,Yfit)  
      Yfit  
      benign malignant  
benign    440        18  
malignant   7       234
```

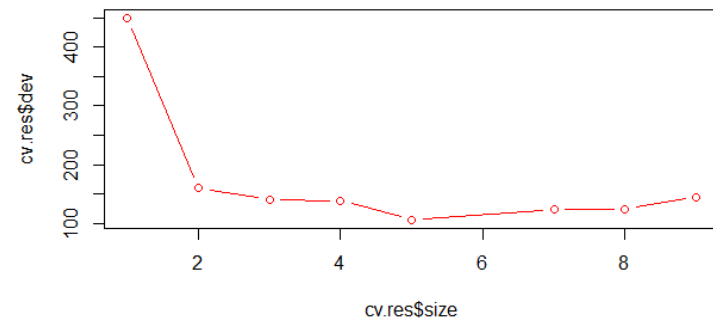
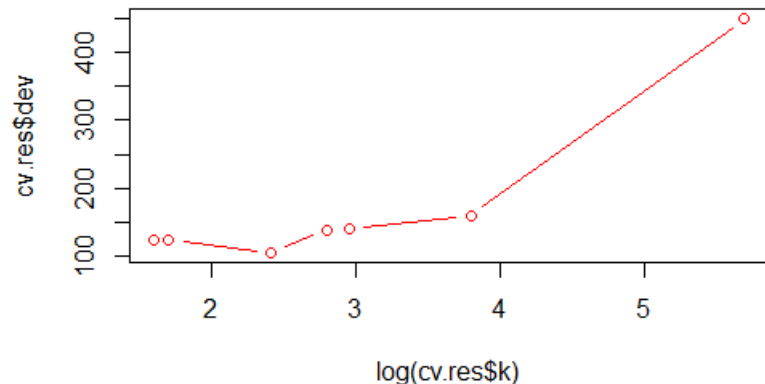

Decision trees in R

- Selecting optimal tree by penalizing
 - `Cv.tree()`

```
set.seed(12345)
ind=sample(1:n, floor(0.5*n))
train=biopsy[ind,]
valid=biopsy[-ind,]

fit=tree(class~., data=train)
set.seed(12345)
cv.res=cv.tree(fit)
plot(cv.res$size, cv.res$dev, type="b",
     col="red")
plot(log(cv.res$k), cv.res$dev,
     type="b", col="red")
```

What is optimal number of leaves?



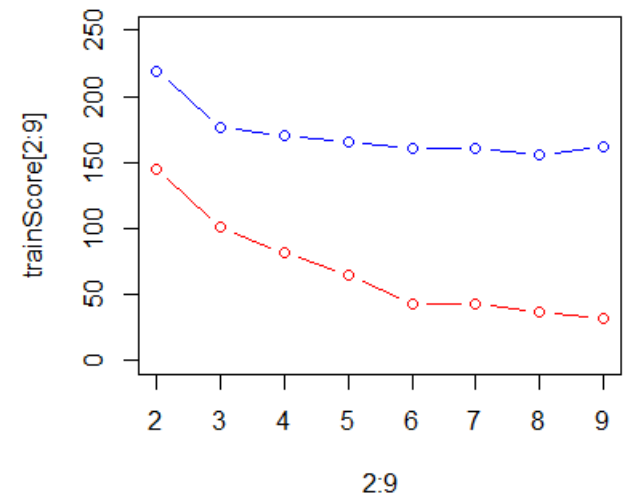
Decision trees in R

- Selecting optimal tree by train/validation

```
fit=tree(class~., data=train)

trainScore=rep(0,9)
testScore=rep(0,9)

for(i in 2:9) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,
type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:9, trainScore[2:9], type="b", col="red",
ylim=c(0,250))
points(2:9, testScore[2:9], type="b", col="blue")
```



What is optimal number of leaves?

Decision trees in R

- Final tree: 5 leaves

```
finalTree=prune.tree(fit, best=5)
Yfit=predict(finalTree, newdata=valid,
type="class")
table(valid$class,Yfit)
```

```
> table(valid$class,Yfit)
```

	Yfit	
	benign	malignant
benign	222	8
malignant	6	114