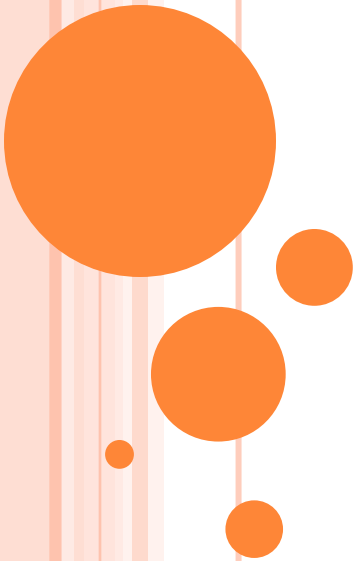


Machine Learning in R

Kevin Lin, Han Liu

Contents

- **GOAL:** Skimming various machine learning algorithms that have implementations in R
- Exploratory Analysis
  - Principal Components Analysis (PCA) - stats
  - K-Means Cluster – stats
  - Expectation-Maximization Algorithm (EM) – mclust
  - Bootstrapping – boot
- Predictive Analysis
  - Generalized Linear Model Elastic Net (glmnet, Lasso) – glmnet
  - Naïve Bayes – e1071
  - Support Vector Machine (SVM) – e1071
  - Linear Discriminant Analysis (LDA) – stats
  - Recursive Partitioning Trees – rpart
  - Boosting – adabag
  - Bagging – adabag
  - Random Forest – randomforest
- Misc. and Specialized Application
  - Linear Program Solver (LP) – lpSolve
  - Hidden Markov Model (HMM) – depmixS4
  - Monte Carlo Markov Chain (MCMC) - mcmc
  - Topic Modeling (TM) – tm and topicmodels
  - Collaborative Filtering – recommenderlabs
  - Neural Networks – neuralnet (not shown)
- Statistics
  - Multiple Hypothesis Testing – biostat (not shown)
  - Propensity Score – TriMatch (not shown)

Note

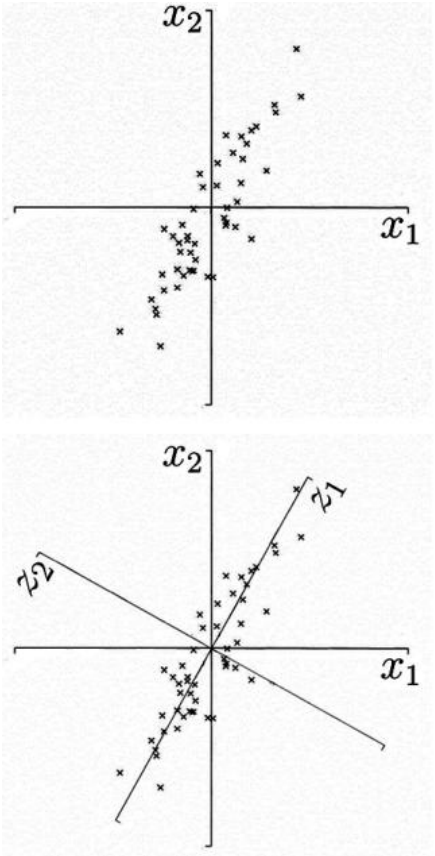
- The following slides are simply quick recaps of various algorithms.
- The R code presented in these slides are more or less directly taken from the R Help pages or vignettes.
- Most of the graphics in this presentation are taken from other websites, presentations and papers.

Principal Component Analysis (PCA) - stats

- Dimension Reduction for L different basis vectors b_i seeking to minimize the reconstruction error

$$\epsilon^2 = \frac{1}{N}\sum_{n=1}^{N}|x_n^2| - \sum_{i=1}^{L} b_i^\intercal \left(\frac{1}{N}\sum_{n=1}^{N} x_n x_n^\intercal\right) b_i$$

- Boils down to finding eigenvectors of the empirical covariance matrix

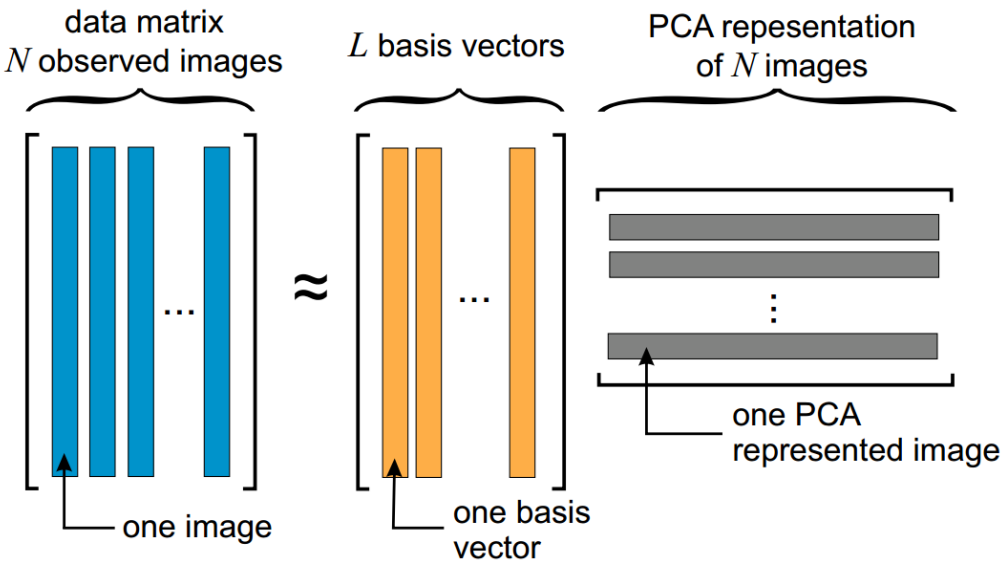


Principal Component Analysis (PCA) - stats

- Example of Images

Principal Component Analysis (PCA) - stats

- How to Reconstruct



- Analysis

Principal Component Analysis (PCA) - stats

- Reconstructed Images

Principal Component Analysis (PCA) - stats

- Example Code

```
data(USArrests)
pc.cr = princomp(USArrests, cor = TRUE)
pc.cr$sdev #the standard deviation of
  the principal components (sqrt of
  eigenvalues)
pc.cr$loadings #the eigenvalues
pc.cr$scores
```

K-Means Cluster - stat

- Initialize
  - Randomly choose initial cluster means $\mu_1^{(0)}, \ldots, \mu_K^{(0)}$
- Repeat
  - Assign each data point to its closest mean

$$z_i^{(j+1)} = \arg \min_{k \in \{1,\ldots,K\}} ||x_i - \mu_k^{(j)}||$$

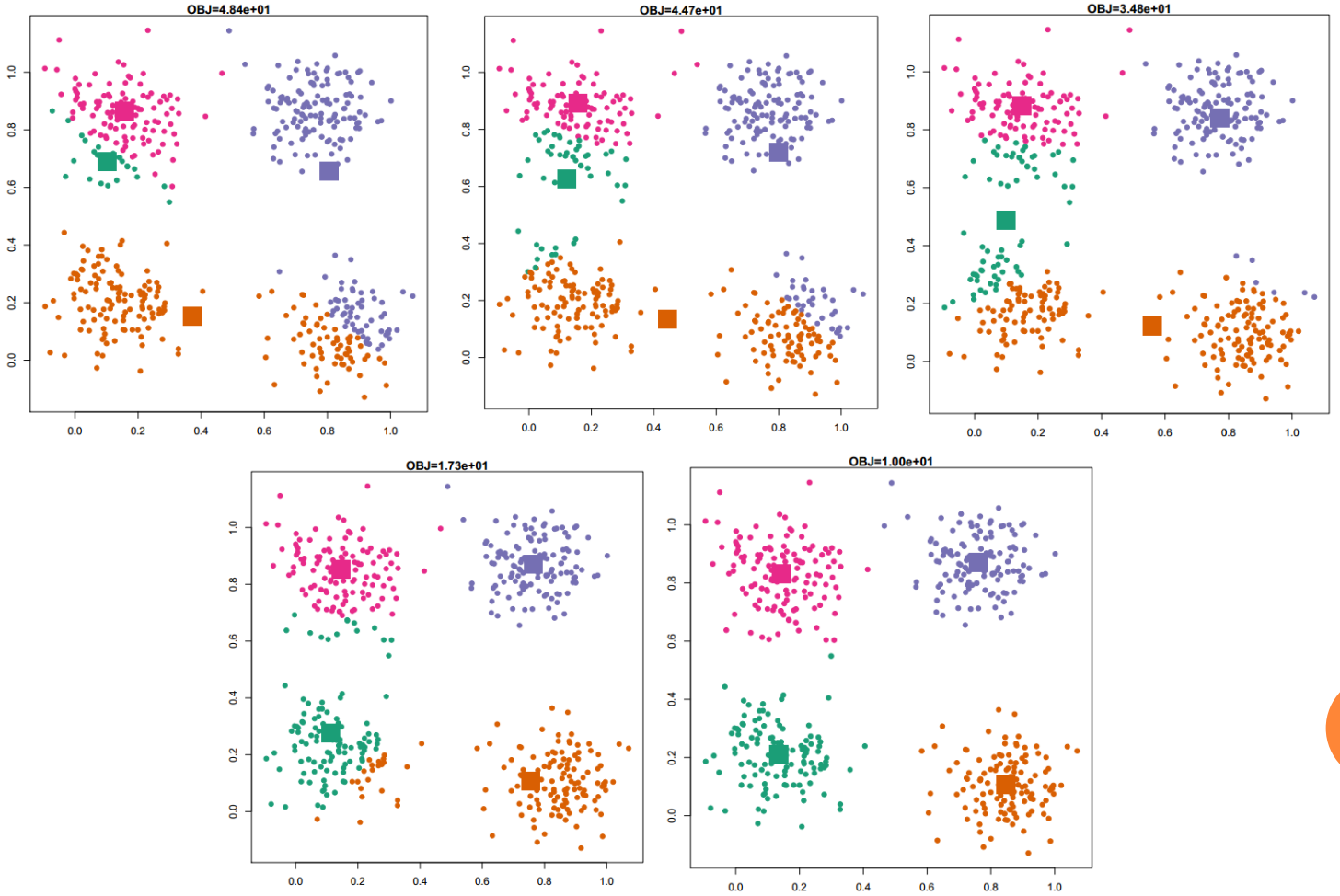
  - Recompute each cluster mean as the mean of all points assigned to it

$$\mu_k^{(j+1)} = \frac{1}{|i : z_i = k|} \sum_{\{i:z_i=k\}} x_i$$

  - Stop when assignments do not change

K-Means Cluster - stat

- Example of Iteration



K-Means Cluster - stat

- Example Code

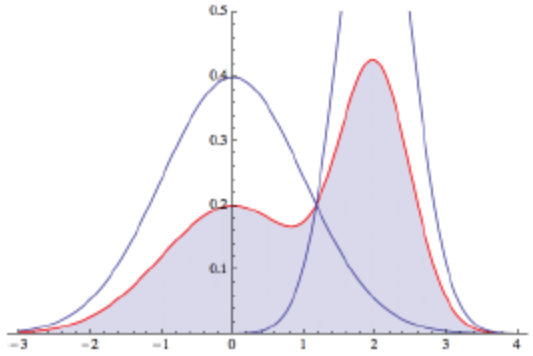
```
x <- rbind(matrix(rnorm(100, sd = 0.3),
  ncol = 2),
           matrix(rnorm(100, mean = 1,
  sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
cl <- kmeans(x, 2)
plot(x, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 8,
  cex = 2)
```

Expectation Maximziation Algorithm (EM) - mclust

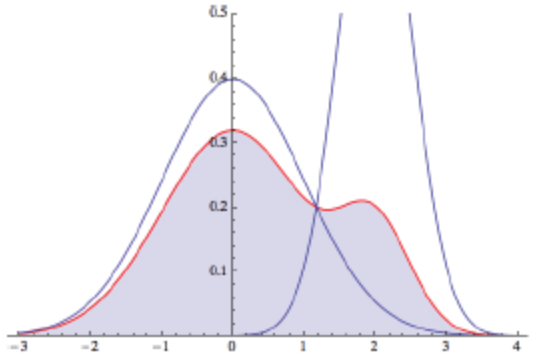
- Mixture of Gaussians
- Likelihood equation

$$L_n(\mu_{1:K}) = \prod_{i=1}^{n} \sum_{k=1}^{K} c_k p(x_i|\mu_k)$$

Mixture of two Gaussians



Influence of the weights



Expectation Maximziation Algorithm (EM) - mclust

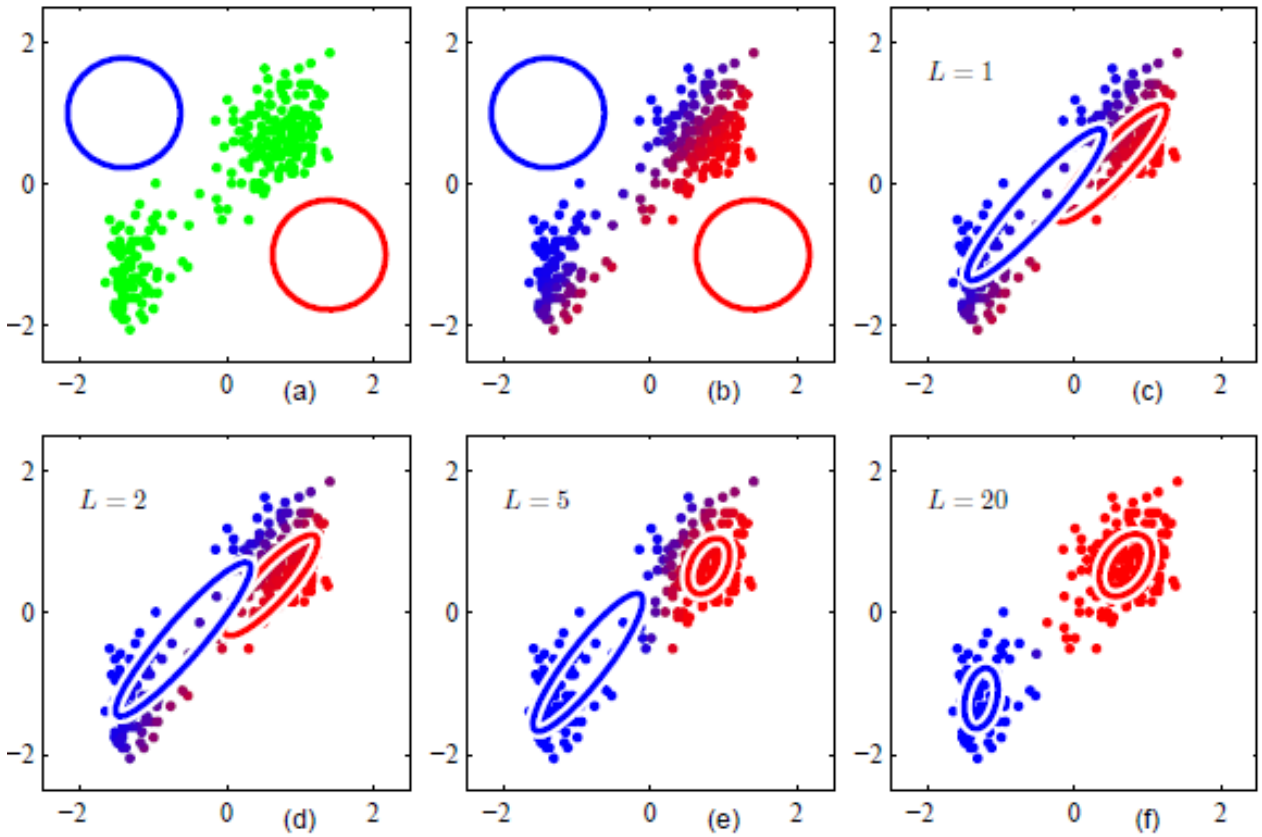
- For Finite Mixture of Gaussians
- Initialize
  - Choosing random values $c_k^{(0)}$ and $\theta_k^{(0)}$
- E-Step
  - Recompute assignment weight matrix

$$a_{ik}^{(j+1)} = \frac{c_k^{(j)} p(x_i|\theta_k^{(j)})}{\sum_{l=1}^{K} c_l^{(j)} p(x_i|\theta_l^{(j)})}$$

- M-Step
  - Recompute the proportions c_k and the parameters $\theta_k = (\mu_k, \Sigma_k)$

$$\mu_k^{(j+1)} = \frac{\sum_{i=1}^{n} a_{ik}^{(j+1)} x_i}{\sum_{i=1}^{n} a_{ik}^{(j+1)}} \qquad \Sigma_k^{(j+1)} = \frac{\sum_{i=1}^{n} a_{ik}^{(j+1)} (x_i - \mu_k^{(j+1)})(x_i - \mu_k^{(j+1)})^T}{\sum_{i=1}^{n} a_{ik}^{(j+1)}}$$

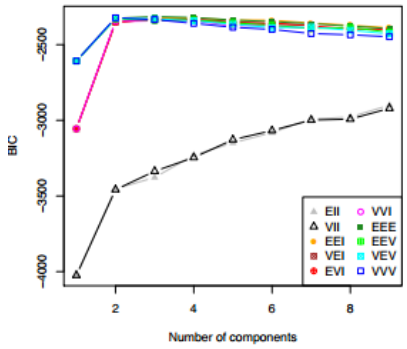
Expectation Maximziation Algorithm (EM) - mclust

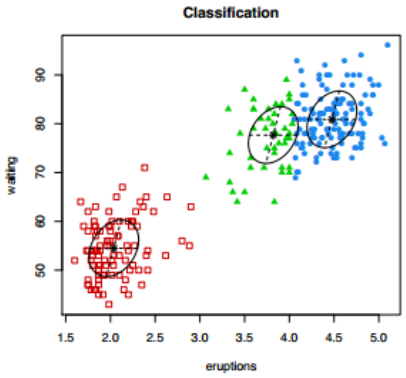


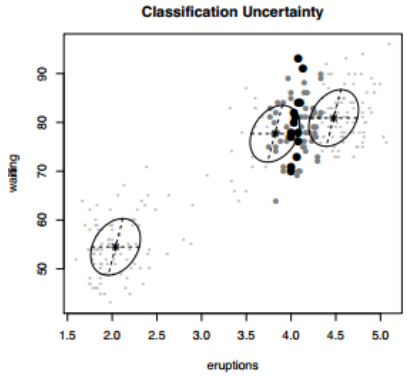
Expectation Maximziation Algorithm (EM) - mclust

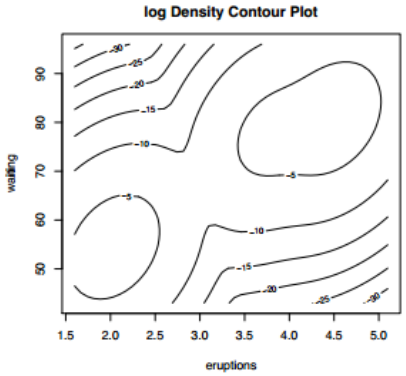
- Example Code

```
library(mclust)
data(faithful)
faithfulMclust <-
  Mclust(faithful)
summary(faithfulMclu
  st, parameters =
  TRUE)
plot(faithfulMclust)
```









Bootstrapping - boot

- Given the data sampled from $p_\theta(x)$, let $\hat{\theta}$ be its estimate. Let $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$ be estimates from B resamples.

- Main idea: we use $\hat{\theta}$ to approximate the true θ and use $\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*$ to approximate the estimate $\hat{\theta}$

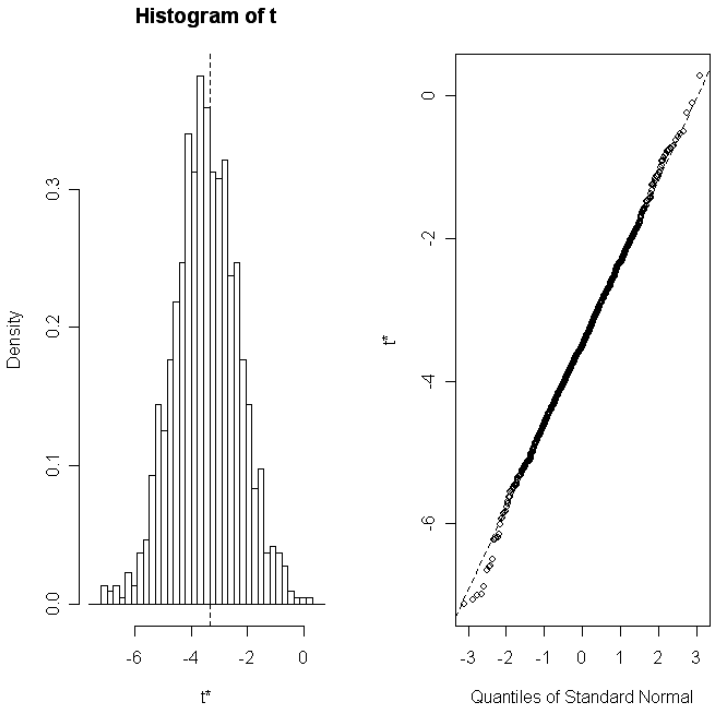
BOOTSTRAPPING - BOOT

- Example Code

```
library(boot)
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data=mtcars, statistic=bs,R=1000,
   formula=mpg~wt+disp)

# view results
results
plot(results, index=1) # intercept
plot(results, index=2) # wt
plot(results, index=3) # disp

# get 95% confidence intervals
boot.ci(results, type="bca", index=1) # intercept
boot.ci(results, type="bca", index=2) # wt
boot.ci(results, type="bca", index=3) # disp
```



Generalized Linear Model Elastic Net (glmnet, Lasso) - glmnet

- Model's Conditional Distribution

$$\mathbb{P}_{\beta_0,\beta,\sigma^2}(Y_i = y_i \mid X_i = x_i) = N(y_i - \beta_0 - x_i^\intercal \beta, \sigma^2)$$

- Elastic Net & Lasso

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \beta_0 - x_i^\intercal\beta)^2\right] + \lambda[(1-\alpha)||\beta||_2^2 + \alpha||\beta||_1]$$

  - or if Lasso where $\alpha = 1$

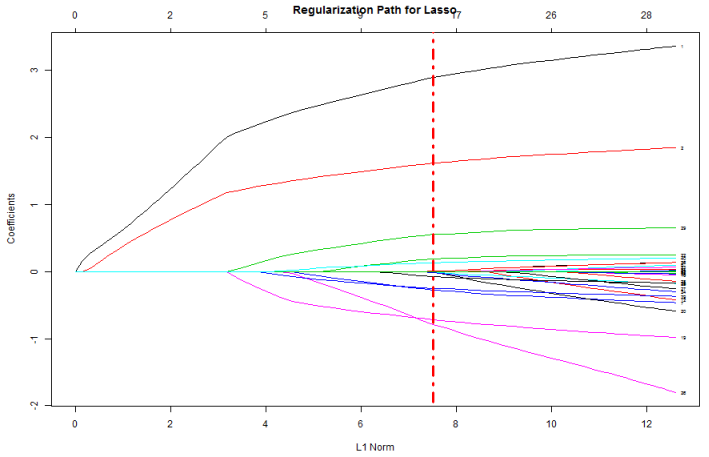
$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \beta_0 - x_i^\intercal\beta)^2\right]$$
$$||\beta||_1 \leq t$$

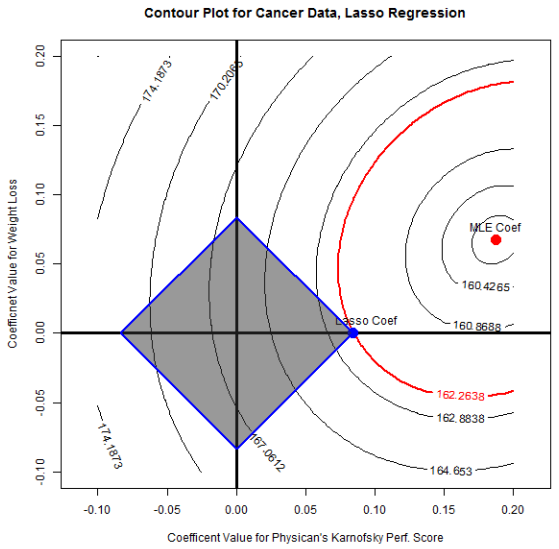
- Package can do model Binomial, Poisson, etc. variables as well

Generalized Linear Model Elastic Net (glmnet, Lasso) - glmnet

- Example Code

```
library(glmnet)
data(cancer,package="survival")
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit1=glmnet(x,y,alpha=1)
print(fit1)
coef(fit1,s=0.01) # extract coefficients at a single value of lambda
predict(fit1,newx=x[1:10,],s=c(0.005)) # make predictions
```





Naïve Bayes – e1071

- Predicting class C from a bunch of features $F_1, \ldots, F_n$
- By Bayes rule and the "naïve" assumption

$$
\begin{aligned}
p(C|F_1, \ldots, F_n) &= \frac{p(C)p(F_1, \ldots, F_n|C)}{p(F_1, \ldots, F_n)} \\
&\propto p(C)p(F_1, \ldots, F_n|C) \\
&\propto p(C)p(F_1|C) \ldots p(F_n|C) \\
&= p(C) \prod_{i=1}^{n} p(F_i|C)
\end{aligned}
$$

$$
p(C|F_1, \ldots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)
$$

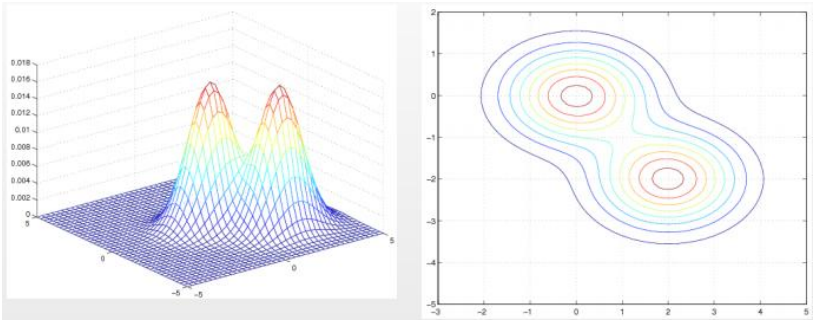
- e1071 assumes all numerical data have a conditional Gaussian distribution, all categorical data have a conditional multinomial distribution

Naïve Bayes – e1071

- Example Code

```
library(e1071)
data(HouseVotes84, package = "mlbench")
model <- naiveBayes(Class ~ ., data =
  HouseVotes84)
predict(model, HouseVotes84[1:10,])
```

Linear Discriminant Analysis (LDA) - stats



- Using Bayes Rule

$$P(y_i = 0|x_i) = \frac{P(x_i|y_i = 0)\pi_1}{\sum_{k=0}^{1} P(x_i|y_i = k)\pi_k}$$

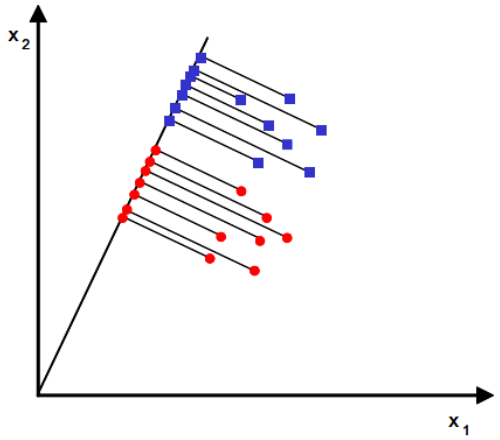
- Assumption

$$P(x_i|y_i = k) = N(\mu_k, \Sigma_k) \text{ and } \Sigma_k = \Sigma \; \forall k$$

- Prediction

$$\hat{G}(x) = \arg\max_k P(y_i = k|x_i) = \arg\max_k \delta_k(x)$$

$$\text{where} \qquad \delta_k(x) = x^\intercal \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\intercal \Sigma^{-1} \mu_k + \log \pi_k$$



- Consider decision boundary where

$$\delta_0(x) = \delta_1(x) \text{ or } \log \frac{\pi_1}{\pi_0} - \frac{1}{2}(\mu_1 - \mu_0)^\intercal \Sigma^{-1}(\mu_1 - \mu_0) + x^\intercal \Sigma^{-1}(\mu_1 - \mu_0) = 0$$

- Equivalent to

$$a_0 + a^\intercal x_i = 0$$

Linear Discriminant Analysis (LDA) - stats

- Example Code

```
library(MASS)
Iris <- data.frame(rbind(iris3[,,1],
  iris3[,,2], iris3[,,3]),
                        Sp =
  rep(c("s","c","v"), rep(50,3)))
z <- lda(Sp ~ ., Iris, prior =
  c(1,1,1)/3)
predict(z, Iris[1:10, ])$class
```

Support Vector Machine – e1071

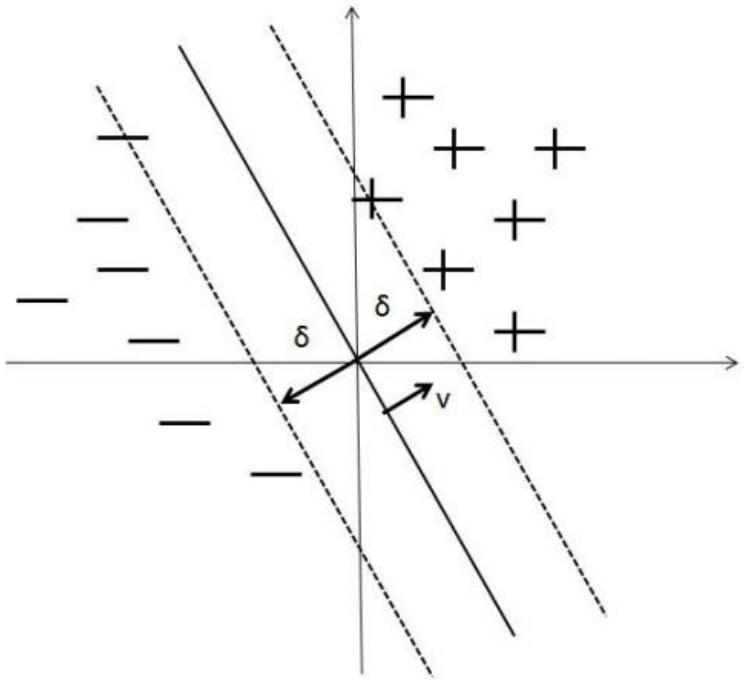
- Primal Problem

$$\begin{aligned} & \max \delta \\ \text{s.t.} \quad & ||v||_2 = 1 \\ & v^\intercal x_i \geq \delta \quad \text{if } y_i = +1 \quad \forall i \\ & v^\intercal x_i \leq \delta \quad \text{if } y_i = -1 \quad \forall i \end{aligned}$$

- Reformulation of Primal Problem

$$w = \frac{v}{\delta}$$

$$\begin{aligned} & \min \frac{1}{2}||w||_2^2 \\ \text{s.t.} \quad & y_i(w^\intercal x_i) \geq 1 \end{aligned}$$

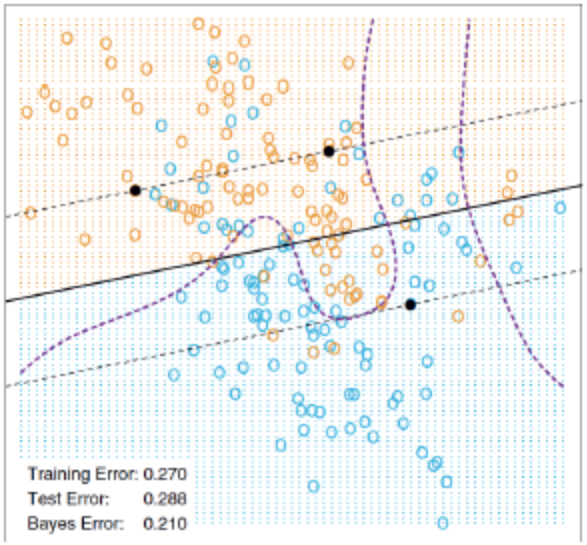


- Dual Problem

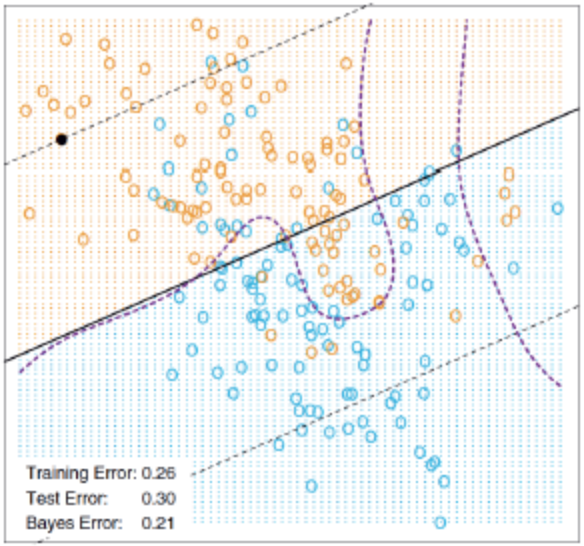
$$\begin{aligned} & \max \sum_{i=1}^{n} \alpha_1 - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^\intercal x_j \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \forall i \end{aligned}$$

- Prediction $f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i^* x_i^\intercal x\right)$
- Can be formulated to have soft margins with penalty

Support Vector Machine – e1071



$\gamma = 100000$



$\gamma = 0.01$

Support Vector Machine – e1071

- Example Code

```
library(e1071)
library(rpart)
data(Glass, package="mlbench")
## split data into a train and test set
index <- 1:nrow(Glass)
testindex <- sample(index,
  trunc(length(index)/3))
testset <- Glass[testindex,]
trainset <- Glass[-testindex,]

svm.model <- svm(Type ~ ., data = trainset,
  cost = 100, gamma = 1)
svm.pred <- predict(svm.model, testset[,-10])
```

Recursive Partitioning Trees (rpart) - rpart

- Use binary trees for prediction
- Idea
  - Find the single found which best splits the data into two groups
  - Apply this process separately on each subgroup (recursive)
  - Stop when subgroups either reach a minimum size of until no improvement can be made
- Choosing how to split
  - Define impurity of a node A

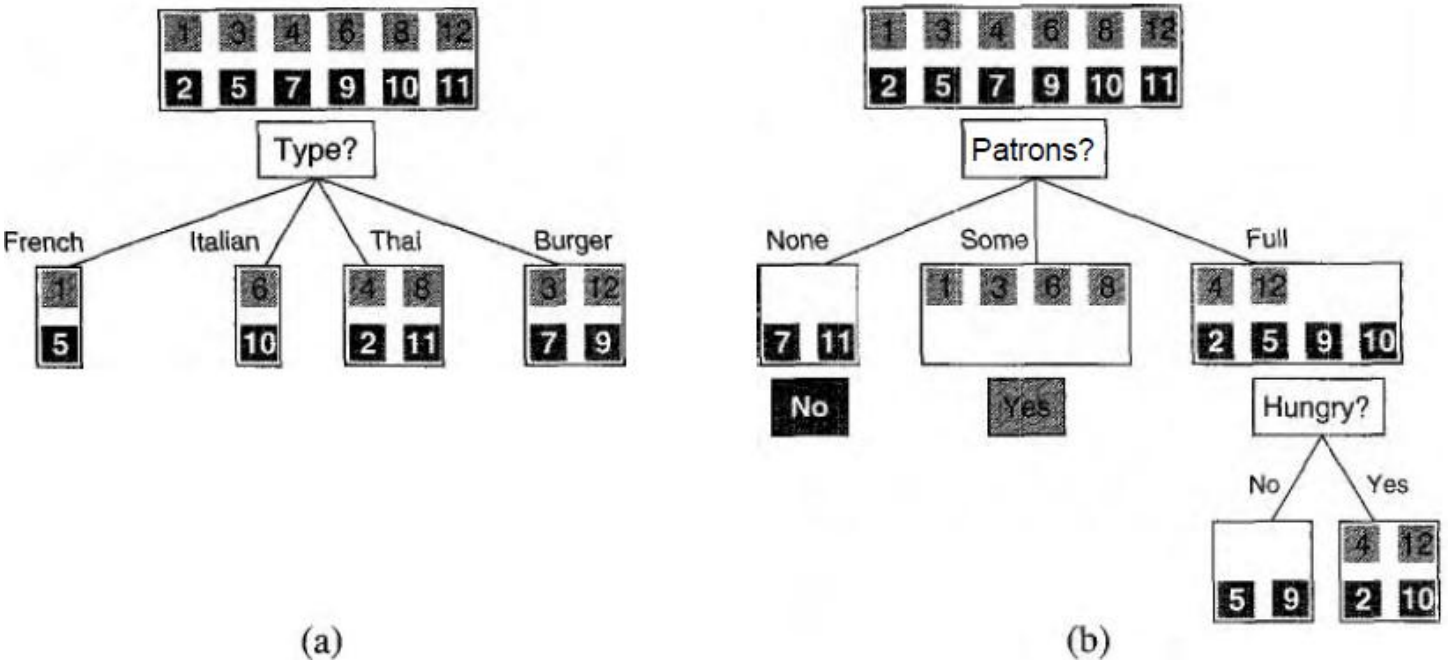
$$I(A) = \sum_{i=1}^{C} f(p_{iA}) \text{ where } f(p) = -p \log p \text{ (Info. index) or } f(p) = p(1-p) \text{ Gini index}$$

  - Split according to maximum impurtiy reduction

$$\Delta(I) = p(A)I(A) - p(A_L)I(A_L) - p(A_R)I(A_R)$$

- Uses alter priors to prune the tree

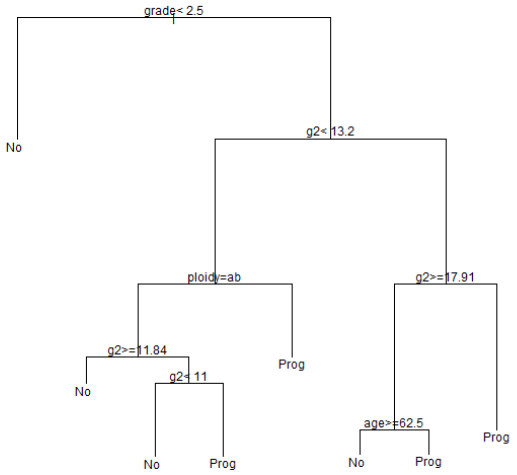
Recursive Partitioning Trees (rpart) - rpart



Recursive Partitioning Trees (rpart) - rpart

- Example Code

```
library(rpart)
data(stagec)
progstat <-
  factor(stagec$pgstat, levels
  = 0:1, labels = c("No",
  "Prog"))
cfit <- rpart(progstat ~ age +
  eet + g2 + grade + gleason +
  ploidy,
                data = stagec,
  method = 'class')
plot(cfit)
text(cfit)
predict(cfit,stagec[1:10,],typ
  e="class")
```



Boosting and Bagging - adabag

- In R: ensemble classification methods for tree structures
- Both: improve accuracy of the ensemble by combining single classifiers which are as different and precise as possible
- Differences:
  - Boosting:
    - Base classifier of each iteration depends on previous one
    - Final boosting ensemble uses weighted majority vote
  - Bagging:
    - Base classifier of each iteration are independent
    - Final bagging ensembles uses simple majority vote

Boosting and Bagging - adabag

- Boosting

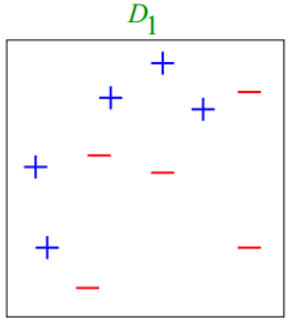
1. Start with $w_b(i) = 1/n,\ i = 1, 2, \ldots, n$
2. Repeat for $b = 1, 2, \ldots, B$
   a) Fit the classifier $C_b(\mathbf{x}_i) = \{1, 2, \ldots, k\}$ using weights $w_b(i)$ on $\mathbf{T}_b$
   b) Compute: $e_b = \sum_{i=1}^{n} w_b(i)\mathbf{I}(C_b(\mathbf{x}_i) \neq y_i)$ and $\alpha_b = 1/2 \ln((1 - e_b)/e_b)$
   c) Update the weights $w_{b+1}(i) = w_b(i) \exp(\alpha_b \mathbf{I}(C_b(\mathbf{x}_i) \neq y_i))$ and normalize them
3. Output the final classifier $C_f(\mathbf{x}_i) = \arg\max_{j \in Y} \sum_{b=1}^{B} \alpha_b \mathbf{I}(C_b(\mathbf{x}_i) = j)$

- Bagging

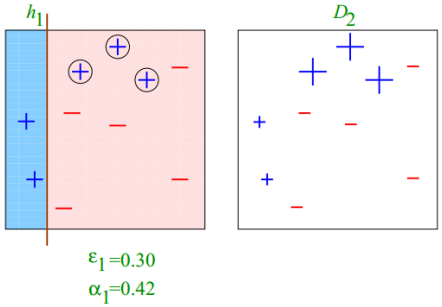
1. Repeat for $b = 1, 2, \ldots, B$
   a) Take a bootstrap replicate $\mathbf{T}_b$ of the training set $\mathbf{T}_n$
   b) Construct a single classifier $C_b(\mathbf{x}_i) = \{1, 2, \ldots, k\}$ in $\mathbf{T}_b$
2. Combine the basic classifiers $C_b(\mathbf{x}_i),\ b = 1, 2, \ldots, B$ by the majority vote (the most often predicted class) to the final decission rule $C_f(\mathbf{x}_i) = \arg\max_{j \in Y} \sum_{b=1}^{B} \mathbf{I}(C_b(\mathbf{x}_i) = j)$

Boosting and Bagging - AdaBag

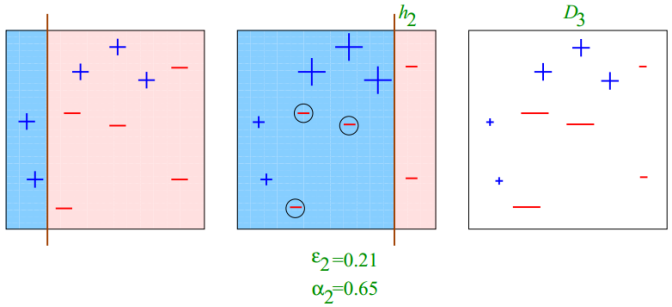
- AdaBoost example



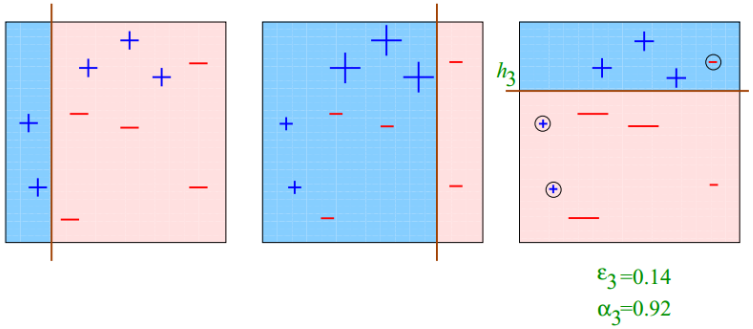
- First iteration



- Second iteration

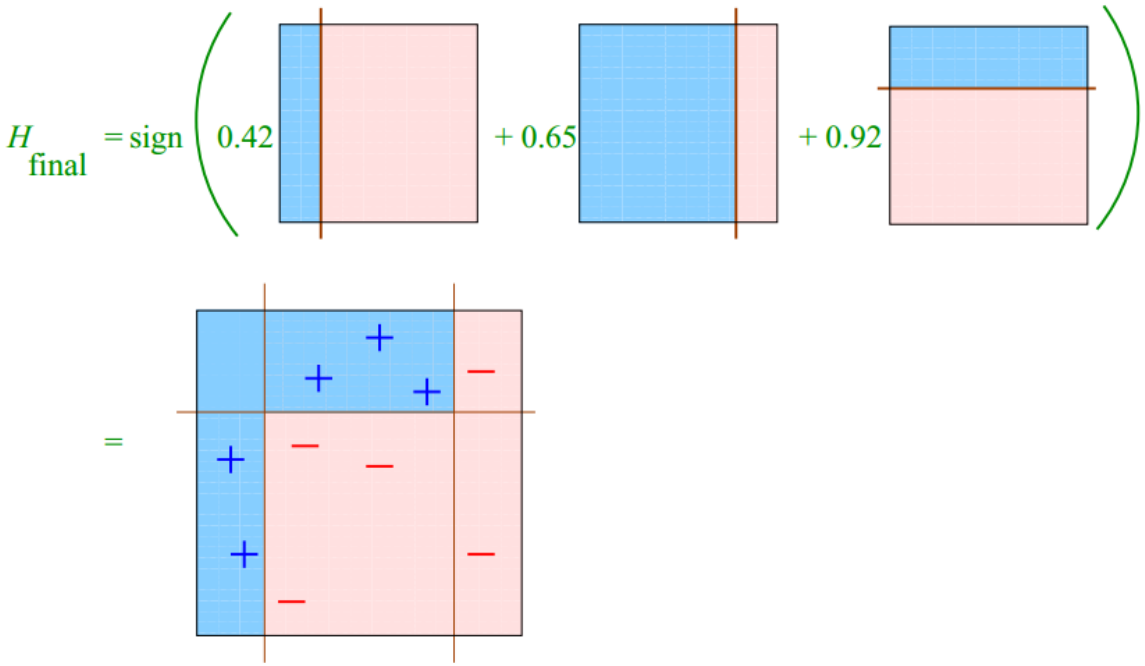


- Third iteration



Boosting and Bagging - Adabag

- AdaBoost example



Boosting and Bagging - Adabag

- Example Code

```
library("adabag")
data("iris")
train <- c(sample(1:50, 25), sample(51:100, 25),
  sample(101:150, 25))
iris.adaboost <- boosting(Species ~ ., data =
  iris[train,], mfinal = 10, control =
  rpart.control(maxdepth = 1))
iris.predboosting <-
  predict.boosting(iris.adaboost,newdata = iris[-train,
  ])
iris.predboosting$confusion


#Bagging
iris.bagging <- bagging(Species ~ ., data = iris[train,
  ], mfinal = 10, control = rpart.control(maxdepth =
  1))
iris.predbagging <- predict.bagging(iris.bagging,
  newdata = iris[-train, ])
```

Random Forest

- Same as bagging except instead of choosing best feature to split, we choose the best among a random subset of features.
- This is done to mitigate correlation among the B trees

- Example Code

```
library(randomForest)
fit <- randomForest(Kyphosis ~ Age + Number
  + Start,   data=kyphosis)
print(fit) # view results
importance(fit) # importance of each
  predictor
```

Linear Program Solver (LP) - lpSolve

- Solving

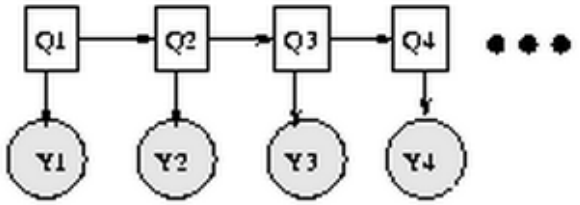
$$\max_{x \in \mathbb{R}^n} c^\intercal x$$
$$\text{s.t.} \quad A^\intercal x \leq b$$

- Example Code

```
# Set up problem: maximize
# x1 + 9 x2 + x3 subject to
# x1 + 2 x2 + 3 x3 <= 9
# 3 x1 + 2 x2 + 2 x3 <= 15
#
f.obj <- c(1, 9, 3)
f.con <- matrix (c(1, 2, 3, 3, 2, 2), nrow=2, byrow=TRUE)
f.dir <- c("<=", "<=")
f.rhs <- c(9, 15)
#
# Now run.
#
res.lp = lp ("max", f.obj, f.con, f.dir, f.rhs)
## Not run: Success: the objective function is 40.5
res.lp$solution
```

Hidden Markov Model (HMM) – depmixS4

- Sequential data $Y_1, \ldots, Y_T$
- Underlying states $Q_1, \ldots, Q_T$
- States follow a Markov chain



$$P(q_{t+1}|q_t, q_{t-1}, \ldots, q_0) = P(q_{t+1}|q_t)$$

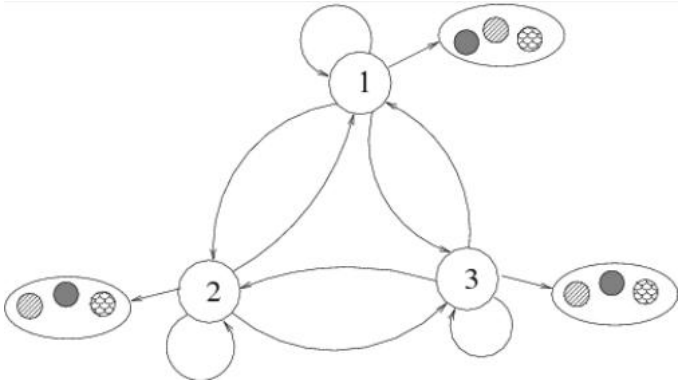
- Transition Probabilities

$$a_{ij} = P(q_{t+1} = j|q_t = i)$$

- Probability to observe data from state

$$P(y_t|q_t = i) = N(\mu_i, \Sigma_i) \qquad \forall i$$

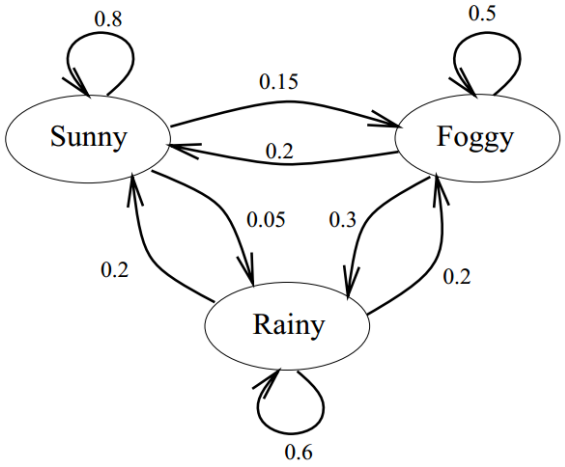
- HiddenMarkov package can do other distributions besides Gaussian



Hidden Markov Model (HMM) - depmixS4

- HMM Example
  - Observe Carrying Umbrella
  - Hidden States are Weather

		Tomorrow's Weather		
		Sunny	Rainy	Foggy
Today's Weather	Sunny	0.8	0.05	0.15
	Rainy	0.2	0.6	0.2
	Foggy	0.2	0.3	0.5



	Probability of Umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

Hidden Markov Model (HMM) - depmixS4

- Example Code

```
library(depmixS4)
data(speed)
mod <-
  depmix(list(rt~1,corr~1),data=speed,tr
  ansition=~Pacc,nstates=2,family=list(g
  aussian(),multinomial("identity")),nti
  mes=c(168,134,137))
fmod = fit(mod)
fmod
summary(fmod)
```

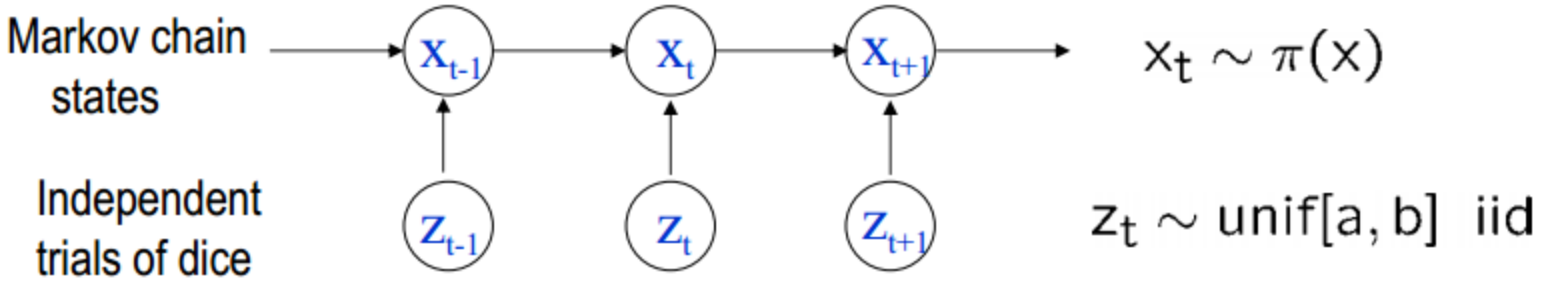
Monte Carlo Markov Chain (MCMC) - MCMC

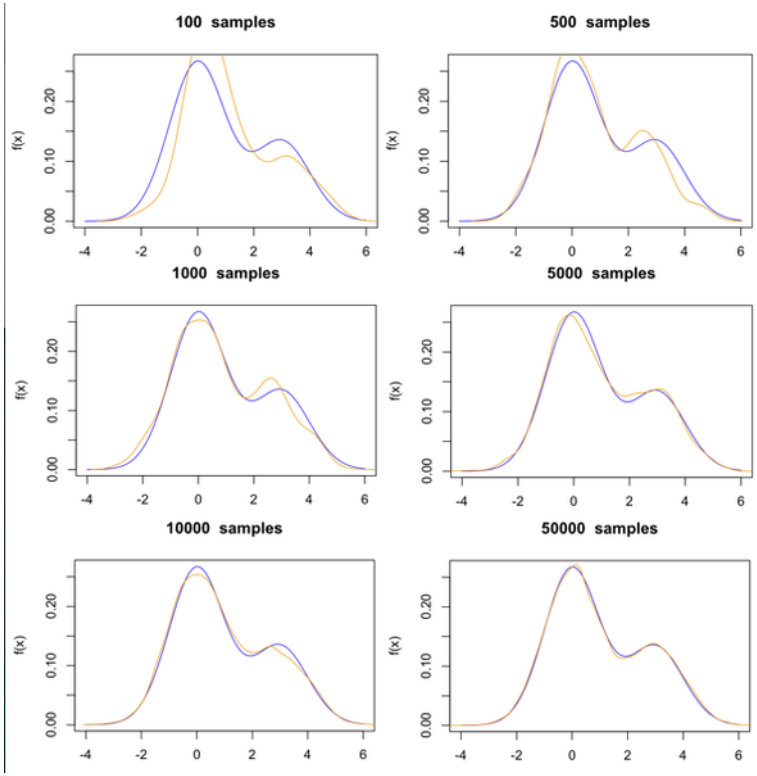
- Generating "fair samples" from a probability in a high-dimensional space
  - "Monte Carlo" for the random simulation
  - "Markov Chain" for the stationary
- Some goals of MCMC
  - Computing the posterior

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{\mathcal{X}} p(y|x')p(x')dx'}$$

  - Difficulty of the integral if it's intractable or high dimensional

Monte Carlo Markov Chain (MCMC) - MCMC





Monte Carlo Markov Chain (MCMC) - MCMC

- Example Code

```
library(mcmc)
data(logit)
out <- glm(y ~ x1 + x2 + x3 + x4, data = logit,family = binomial(), x =
   TRUE)
summary(out)
x <- out$x
y <- out$y
lupost <- function(beta, x, y) {eta <- as.numeric(x %*% beta)
  logp <- ifelse(eta < 0, eta - log1p(exp(eta)), - log1p(exp(- eta)))
  logq <- ifelse(eta < 0, - log1p(exp(eta)), - eta - log1p(exp(- eta)))
  logl <- sum(logp[y == 1]) + sum(logq[y == 0])
  return(logl - sum(beta^2) / 8)
}
beta.init <- as.numeric(coefficients(out))
out <- metrop(lupost, beta.init, 1e3, x = x, y = y)
names(out)
out$accept
```

Topic Modeling (TM) – tm and topicmodels

- Generative model

Step 1: The term distribution β is determined for each topic by

$$\beta \sim \text{Dirichlet}(\delta).$$

Step 2: The proportions θ of the topic distribution for the document w are determined by

$$\theta \sim \text{Dirichlet}(\alpha).$$

Step 3: For each of the N words w_i

(a) Choose a topic $z_i \sim \text{Multinomial}(\theta)$.

(b) Choose a word w_i from a multinomial probability distribution conditioned on the topic z_i: $p(w_i|z_i, \beta)$.

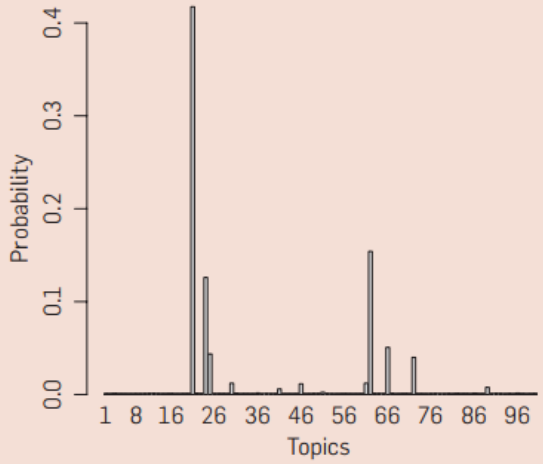
β is the term distribution of topics and contains the probability of a word occurring in a given topic.

- Likelihood

$$L_N(\alpha, \beta) = \log p(w|\alpha, \beta)$$
$$= \log \int \left\{ \sum_z \left[\prod_{i=1}^{N} p(w_i|z_i, \beta) p(z_i|\theta) \right] \right\} p(\theta|\alpha) d\theta$$

Topic Modeling (TM) – tm and topicmodels

Figure 2. Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left are the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Topic Modeling (TM) – tm and topicmodels

- Example Code

```
install.packages("corpus.JSS.papers",repos = "http://datacube.wu.ac.at/", type = "source")
data("JSS_papers", package = "corpus.JSS.papers")
JSS_papers <- JSS_papers[JSS_papers[,"date"] < "2010-08-05",]
JSS_papers <- JSS_papers[sapply(JSS_papers[, "description"],Encoding) == "unknown",]
library("tm")
library("XML")
remove_HTML_markup <- function(s) tryCatch({doc <- htmlTreeParse(paste("<!DOCTYPE html>", s),asText = TRUE, trim
    = FALSE)
                                    xmlValue(xmlRoot(doc))}, error = function(s) s)
corpus <- Corpus(VectorSource(sapply(JSS_papers[, "description"],remove_HTML_markup)))

library("SnowballC")
Sys.setlocale("LC_COLLATE", "C")
JSS_dtm <- DocumentTermMatrix(corpus,control = list(stemming = TRUE, stopwords = TRUE, minWordLength =
    3,removeNumbers = TRUE, removePunctuation = TRUE))
dim(JSS_dtm)

library("topicmodels")
k <- 30
SEED <- 2010
jss_TM <- LDA(JSS_dtm, k = k, control = list(seed = SEED))
Topic <- topics(jss_TM, 1)
Terms <- terms(jss_TM, 10)
Terms[,1:5]
```

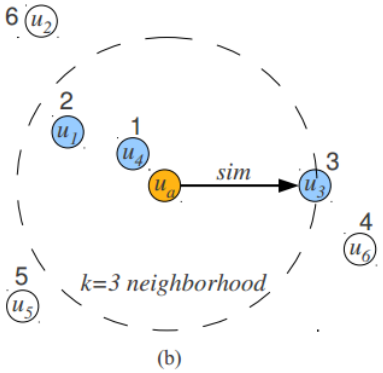
Collaborative Filtering - recommenderlab

- Creating recommendations for users over a set of items where each user gives ratings for some of the items
- One possible collaborative filter
  - For each user, find "neighbors" of the user based on the Cosine similarity

$$sim_{\text{Cosine}}(x, y) = \frac{x^{\intercal} y}{||x|| ||y||}$$

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
r_a	3.5	4.0			1.3		2.0	

(a)



(b)

Collaborative Filtering - recommenderlab

- Example Code

```
library(recommenderlab)
data(Jester5k)
as(Jester5k[1001:1002,1:10],"list")
recommenderRegistry$get_entries(dataTyp
  e = "realRatingMatrix")
r <- Recommender(Jester5k[1:1000],
  method = "UBCF")
names(getModel(r))
recom <- predict(r,
  Jester5k[1001:1002], type="ratings")
as(recom, "matrix")[,1:10]
```