# Regression and regularization

Lecture 1d



1

## Overview

- Linear regression
- Ridge Regression
- Lasso
- Variable selection



2

## Simple linear regression

**Model:**

$$y \sim N(w_0 + w_1 x, \sigma^2)$$

or

$$y = w_0 + w_1 x + \epsilon,$$
$$\epsilon \sim N(0, \sigma^2)$$

or

$$p(y|x, w) = N(w_0 + w_1 x, \sigma^2)$$

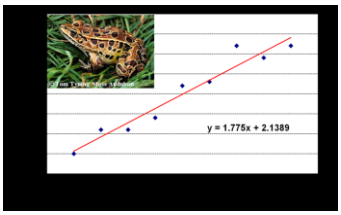
**Terminology:**

$w_0$: **intercept (or bias)**

$w_1$: **regression coefficient**

**Response**

**The target responds directly and linearly to changes in the feature**






3

## Ordinary least squares regression (OLS)

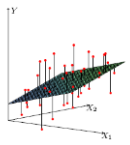
**Model:**

$$y \sim N(\boldsymbol{w}^T \boldsymbol{x}, \sigma^2)$$

*where*

$$\boldsymbol{w} = \{w_0, \ldots w_d\}$$
$$\boldsymbol{x} = \{1, x_1, \ldots x_d\}$$

Why is "1" here?



The response variable responds directly and linearly to changes in each of the inputs



4

## Ordinary least squares regression

**Given** data set $D$

| Case | $X_1$ | $X_2$ | | | $X_p$ | $Y$ |
|---|---|---|---|---|---|---|
| 1 | $x_{11}$ | $x_{21}$ | | | $x_{p1}$ | $y_1$ |
| 2 | $x_{12}$ | $x_{22}$ | | | $x_{p2}$ | $y_2$ |
| 3 | $x_{13}$ | $x_{23}$ | | | $x_{p3}$ | $y_3$ |
| | | | | | | |
| | | | | | | |
| $N$ | $x_{1N}$ | $x_{2N}$ | | | $x_{pN}$ | $y_N$ |

**Estimation:** maximizing the likelihood

$$\hat{w} = \max_{w} p(D|w)$$

Is equivalent to minimizing

$$RSS(w) = \sum_{i=1}^{n} (Y_i - \boldsymbol{w}^T \boldsymbol{X}_i)^2$$



5

## Matrix formulation of OLS regression

Optimality condition:

where $$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = 0$$

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{21} & & x_{p1} \\ 1 & x_{12} & x_{22} & & x_{p2} \\ & & & & \\ & & & & \\ 1 & x_{1N} & x_{2N} & & x_{pN} \end{pmatrix} \quad \textbf{and} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \\ \\ y_N \end{pmatrix}$$



6

## Parameter estimates and predictions

- Least squares estimates of the parameters

$$\hat{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Predicted values $$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{w} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

- Linear regression belongs to the class of **linear smoothers**

Hat matrix

Why is it called so?



7

## Degrees of freedom

Definition:

$$df(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- Larger covariance → stronger connection → model can approximate data better→ model more flexible (complex)
- For linear smoothers $\hat{Y} = S(X)Y$

$$df = trace(S)$$

- For linear regression, degrees of freedom is

$$df = trace(P) = p$$



8

## Different types of features

- Interval variables
- Numerically coded ordinal variables
  - (small=1, medium=2, large=3)
- Dummy coded qualitative variables

**Basis function expansion**:
If $y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 e^{-x_2} + \epsilon,$

Model becomes linear if to recompute:

$$\phi_1(x_1) = x_1$$
$$\phi_2(x_1) = x_1^2$$
$$\phi_3(x_1) = e^{-x_2}$$

**Example of dummy coding:**

$$x_1 = \begin{cases} 1, \text{if Jan} \\ 0, \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1, \text{if Feb} \\ 0, \text{otherwise} \end{cases}$$

.
.
.

$$x_{11} = \begin{cases} 1, \text{if Nov} \\ 0, \text{otherwise} \end{cases}$$





## Basis function expansion

- In general $\phi_1(\dots)$ may be a function of several $x$ components
- Having data given by **X,** compute new data
- $$\Phi = \begin{pmatrix} 1 & \phi_1(x_{11,\dots,}x_{1p}) & \dots & \phi_p(x_{11,\dots,}x_{1p}) \\ & \dots & \dots & \dots \\ 1 & \phi_1(x_{n1,\dots,}x_{np}) & \dots & \phi_p(x_{n1,\dots,}x_{np}) \end{pmatrix}$$
- If doing a basis function in a model, replace **X** by $\Phi$ everywhere where **X** is used:

$$\hat{\boldsymbol{y}} = \Phi(\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{y}$$





## Linear regression in R

- fit=lm(formula, data, subset, weights,…)
  - **data** is the data frame containing the predictors and response values
  - **formula** is expression for the model
  - **subset** which observations to use (training data)?
  - **weights** should weights be used?

**fit** is object of class **lm** containing various regression results.

- Useful functions (many are generic, used in many other models)
  - Get details about the particular function by ".", for ex. predict.lm

summary(fit)
predict(fit, newdata, se.fit, interval)
coefficients(fit) # model coefficients
confint(fit, level=0.95) # CIs for model parameters
fitted(fit) # predicted values
residuals(fit) # residuals





## An example of ordinary least squares regression

```
mydata=read.csv2("Bilexempel.csv")
fit1=lm(Price~Year, data=mydata)
summary(fit1)
fit2=lm(Price~Year+Mileage+Equipment,
data=mydata)
summary(fit2)
```

**Response variable:**
Requested price of used Porsche cars (1000 SEK)

```
> summary(fit1)

Call:
lm(formula = Price ~ Year, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max
-167683 -14683   20056   35933   72317

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -78161027    8448038  -9.252 6.00e-13 ***
Year            39246       4226   9.288 5.25e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57270 on 57 degrees of freedom
Multiple R-squared:  0.6022,    Adjusted R-squared:  0.5952
F-statistic: 86.26 on 1 and 57 DF,  p-value: 5.248e-13
```

**Inputs:**
$X_1$ = Manufacturing year
$X_2$ = Milage (km)
$X_4$ = Equipment (0 or 1)

## An example of ordinary least squares regression

```
> summary(fit2)

Call:
lm(formula = Price ~ Year + Mileage + Equipment, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max
-66223 -10525   -739  14128  65332

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.083e+07  6.309e+06  -3.302  0.00169 **
Year         1.062e+04  3.154e+03   3.366  0.00139 **
Mileage     -2.077e+00  2.022e-01 -10.269 2.14e-14 ***
Equipment    5.790e+04  1.041e+04   5.563 8.08e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 29270 on 55 degrees of freedom
Multiple R-squared:  0.8997, Adjusted R-squared:  0.8942
F-statistic: 164.5 on 3 and 55 DF,  p-value: < 2.2e-16
```

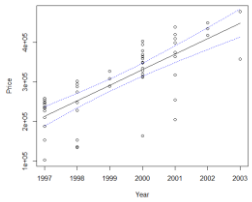


13

## An example of ordinary least squares regression

- Prediction

```
fitted <- predict(fit1, interval =
"confidence")

# plot the data and the fitted line
attach(mydata)
plot(Year, Price)
lines(Year, fitted[, "fit"])

# plot the confidence bands
lines(Year, fitted[, "lwr"], lty = "dotted",
col="blue")
lines(Year, fitted[, "upr"], lty = "dotted",
col="blue")
detach(mydata)
```

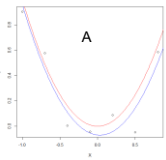


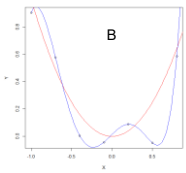



14

## Ridge regression

- Problem: linear regression can overfit:
  - Take $Y := Y, X_1 = X, X_2 = X^2, \ldots, X_p = X^p$ → polinomial model, fit by linear regression
  - High degree of polynomial leads to overfitting.






15

## Ridge regression

- **Idea**: Keep all predictors but shrink coefficients to make model less complex

minimize $-loglikelihood + \lambda_0 \|w\|_2^2$

→ **$l_2$ regularization**

  - Given that model is Gaussian, we get **Ridge regression:**

$$\hat{w}^{ridge} = \text{argmin}\left\{\sum_{i=1}^{N}(y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda \sum_{j=1}^{p} w_j^2\right\}$$

- $\lambda > 0$ is **penalty factor**



16

## Ridge regression

Equivalent form

$$\hat{w}^{ridge} = \text{argmin} \sum_{i=1}^{N} (y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2$$

**subject to** $\sum_{j=1}^{p} w_j^2 \leq s$

***Solution***

$$\widehat{\boldsymbol{w}}^{\boldsymbol{ridge}} = \left(\boldsymbol{X^T X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X^T y}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{w} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

Hat matrix

How do we compute degrees of freedom here?





## Ridge regression

**Properties**

- Extreme cases:
  - $\lambda = 0$ usual linear regression (no shrinkage)
  - $\lambda = +\infty$ fitting a constant ($w = 0$ except of $w_0$)
- When input variables are ortogonal (not realistic), $X^T X = I \rightarrow$

  $\widehat{\boldsymbol{w}}^{\text{ridge}} = \frac{1}{1+\lambda}\mathbf{w}^{\mathbf{linreg}}$ → coefficients are equally shrunk
- **Ridge regression is particularly useful if the explanatory variables are strongly correlated to each other.**
  - Correlated variables often correspond large $w$ → shrunk
- Degrees of freedom decrease when $\lambda$ increases
  - $\lambda = 0 \rightarrow d.f. = p$





## Ridge regression

**Properties**

- Shrinking enables estimation of regression coefficients even if the number of parameters exceeds the number of cases! ($X^T X + \lambda I$ is always nonsingular)
  - Compare with linear regression
- How to estimate $\lambda$?
  - cross-validation





## Ridge regression

- Bayesian view
  - Ridge regression is just a special form of Bayesian Linear Regression with constant $\sigma^2$:

$$\boldsymbol{y} \sim N(\boldsymbol{y}|\boldsymbol{w_o} + \boldsymbol{X}\boldsymbol{w}, \boldsymbol{\sigma^2 I})$$
$$\boldsymbol{w} \sim \boldsymbol{N}\left(\mathbf{0}, \frac{\sigma^2}{\lambda}\boldsymbol{I}\right)$$

**Theorem** MAP estimate to the Bayesian Ridge is equal to solution in frequenist Ridge

$$\widehat{\boldsymbol{w}}^{\boldsymbol{ridge}} = \left(\boldsymbol{X^T X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X^T y}$$

- In Bayesian version, we can also make inference about $\lambda$

## Ridge regression

**Example Computer Hardware Data Set** : performance measured for various processors and also

- Cycle time
- Memory
- Channels
- …

Build model predicting performance

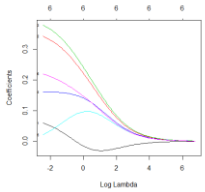




## Ridge regression

- R code: use package **glmnet** with alpha=0 (Ridge regression)
- Seeing how Ridge converges

```
data=read.csv("machine.csv", header=F)
covariates=scale(data[,3:8])
response=scale(data[, 9])

model0=glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
plot(model0, xvar="lambda", label=TRUE)
```





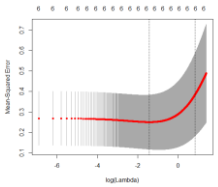



## Ridge regression

- Choosing the best model by cross-validation:

```
model=cv.glmnet(as.matrix(covariates),
response, alpha=0,family="gaussian")
model$lambda.min
plot(model)
coef(model, s="lambda.min")
```

```
> coef(model, s="lambda.min")
7 x 1 sparse Matrix of class "dgC
                       1
(Intercept) -4.530442e-17
V3           3.420739e-02
V4           3.085696e-01
V5           3.403839e-01
V6           1.593470e-01
V7           5.489116e-02
V8           1.970982e-01
```




```
> model$lambda.min
[1] 0.046
```





## Ridge regression

- How good is this model in prediction?

```
ind=sample(209, floor(209*0.5))
data1=scale(data[,3:9])
train=data1[ind,]
test=data1[-ind,]

covariates=train[,1:6]
response=train[, 7]
model=cv.glmnet(as.matrix(covariates), response, alpha=1,family="gaussian",
lambda=seq(0,1,0.001))
y=test[,7]
ynew=predict(model, newx=as.matrix(test[, 1:6]), type="response")

#Coefficient of determination
sum((ynew-mean(y))^2)/sum((y-mean(y))^2)

sum((ynew-y)^2)
```

Note that data are so small so numbers change much for other train/test

```
> sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
[1] 0.5438148
> sum((ynew-y)^2)
[1] 18.04988
>
```

## LASSO

- **Idea**: Similar idea to Ridge
- Minimize minus loglikelihood plus **linear** penalty factor→ $\mathbf{l_1}$ **regularization**
  - Given that model is Gaussian, we get **LASSO** (least absolute shrinkage and selection operator):

$$\hat{w}^{lasso} = \text{argmin}\left\{\sum_{i=1}^{N}(y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2 + \lambda\sum_{j=1}^{p}|w_i|\right\}$$

- $\lambda > 0$ is **penalty factor**



25

## LASSO

- Equivalently

$$\hat{w}^{lasso} = \text{argmin}\sum_{i=1}^{N}(y_i - w_0 - w_1 x_{1j} - \ldots - w_p x_{pj})^2$$

**subject to** $\sum_{j=1}^{p}|w_i| \le s$

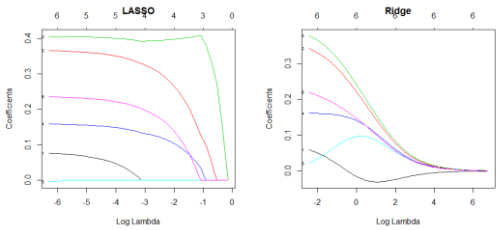


26

## LASSO vs Ridge

- **LASSO yields sparse solutions!**

**Example** Computer hardware data






27

## LASSO vs Ridge

- Only 5 variables selected by LASSO

```
> coef(model, s="lambda.min")
7 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -5.091825e-17
V3           6.350488e-02
V4           3.578607e-01
V5           4.033670e-01
V6           1.541329e-01
V7           .
V8           2.287134e-01
> 1
```

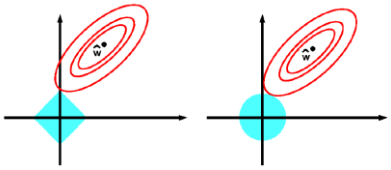
```
> sum((ynew-mean(y))^2)/sum((y-mean(y))^2)
[1] 0.5826904
> sum((ynew-y)^2)
[1] 16.63756
```



28

## LASSO vs Ridge

- Why Lasso leads to sparse solutions?
  - Feasible area for Ridge is a circle (2D)
  - Feasible area for LASSO is a polygon (2D)





29

## LASSO properies

- **Lasso is widely used when $p \gg n$**
  - Linear regression breaks down when $p > n$
  - Application: DNA sequence analysis, Text Prediction
- When inputs are orthonormal,

$$\hat{w}_i^{\text{lasso}} = sign\left(w_i^{linreg}\right)\left(\left|w_i^{linreg}\right| - \frac{\lambda}{2}\right)_+$$

- No explicit formula for $\hat{\boldsymbol{w}}^{\boldsymbol{lasso}}$
  - Optimization algorithms used

**Coding in R: use glmnet() with alpha=1**



30

## Variable selection

- .. Or "Feature selection"

Often, we do not need all features available in the data to be in the model

**Reasons:**

- Model can become overfitted (recall polynomial regression)
- Large number of predictors → model is difficult to use and interpret



31

## Variable selection

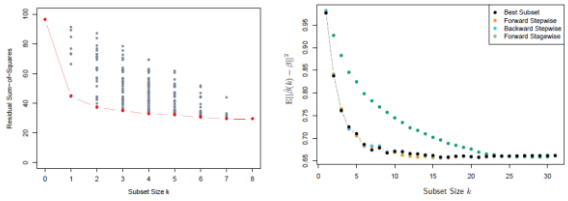
Alternative 1: Variable subset selection

- Best subset selection:
  - Consider different subsets of the full set of features, fit models and evaluate their quality
    - Problem: computationally difficult for p around 30 or more
    - How to choose the best model size? Some measure of predictive performance normally used (ex. AIC).
- Forward and Backward stepwise selection
  - Starts with 0 features (or full set ) and then adds a feature (removes feature) that most improves the measure selected.
    - Can handle large *p* quickly
    - Does not examine all possible subsets (not the "best")



32

## RSS and MSE depend on k






33

## Variable selection in R

- Use stepAIC() in MASS

```
library(MASS)
fit <- lm(V9~.,data=data.frame(data1))
step <- stepAIC(fit, direction="both")
step$anova
summary(step)
```

```
Call:
lm(formula = V9 ~ V3 + V4 + V5 + V6 + V8, data = data.fra

Residuals:
      Min        1Q    Median        3Q       Max
-1.20232 -0.15512  0.03579  0.18567  2.42280

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.783e-17  2.574e-02   0.000   1.0000
V3           7.948e-02  2.826e-02   2.813   0.0054 **
V4           3.661e-01  4.312e-02   8.490 4.34e-15 ***
V5           4.055e-01  4.664e-02   8.695 1.18e-15 ***
V6           1.591e-01  3.394e-02   4.687 5.07e-06 ***
V8           2.360e-01  3.356e-02   7.031 3.06e-11 ***
```

```
> step <- stepAIC(fit, direction="both")
Start:  AIC=-405.35
V9 ~ V3 + V4 + V5 + V6 + V7 + V8

       Df Sum of Sq     RSS     AIC
- V7    1    0.0139  28.117 -407.25
<none>               28.103 -405.35
- V3    1    1.0819  29.185 -399.46
- V6    1    2.9385  31.041 -386.37
- V8    1    6.3150  34.418 -364.99
- V4    1    9.7492  37.852 -345.11
- V5    1   10.4837  38.586 -341.09

Step:  AIC=-407.25
V9 ~ V3 + V4 + V5 + V6 + V8

       Df Sum of Sq     RSS     AIC
<none>               28.117 -407.25
+ V7    1    0.0139  28.103 -405.35
- V3    1    1.0958  29.212 -401.26
- V6    1    3.0431  31.160 -387.77
- V8    1    6.8472  34.964 -363.70
- V4    1    9.9840  38.101 -345.74
- V5    1   10.4713  38.588 -343.08
```



34

# Model selection

Lecture 1e



1

## Overview

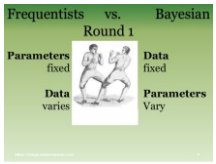
- Model fitting
- Model selection



2

## Frequentist vs Bayesian

- Probabilistic Model $p(y, x, w)$
  - **Frequentists**: $w$ is a parameter that should be estimated by model fitting
  - **Bayesians**: $w$ is a random variable that has a prior distribution $p(w)$
    - How to set $p(w)$??




Example: Linear regression, what are parameters here?

$y \sim w_0 + \boldsymbol{wx} + e, e \sim N(0, \sigma^2)$

$y \sim N(w_0 + \boldsymbol{wx}, \sigma^2)$



3

## An estimator

- $\hat{\boldsymbol{w}} = \delta(D)$ (some function of your data) – an **estimator**
- Optimal parameter values?→there can be many ways to compute them (MLE, shrinkage…)
  - Compare Bayesian: given estimators $\boldsymbol{w^1}$ and $\boldsymbol{w^2}$ , we **can** compare them! $p(\boldsymbol{w^1}|\boldsymbol{D}) > \boldsymbol{p}(\boldsymbol{w^2}|\boldsymbol{D})$
  - There is no easy way to compare estimators in frequentist tradition

Example: Linear regression

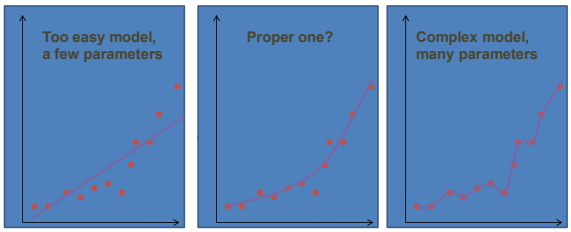
- Estimator 1: $\boldsymbol{w} = (X^T X)^{-1} X^T Y$ (maximum likelihood)
- Estimator 2: $\boldsymbol{w} = (0, \dots, 0, 1)$
- Which one is better?
  - A comparison strategy is needed!



4

## Overfitting

- Complex model can overfit your data






5

## Overfitting: solutions

- Observed: Maximum likelihood can lead to overfitting.
- Solutions
  - Selecting proper parameter values
    - Regularized risk minimization
  - Selecting proper model type, for ex. number of parameters
    - Houldout method
    - Cross-validation



6

## Model selection

- Given a model, choose the optimal parameter values
  - Decision theory
- Define loss $L(Y, \hat{y})$
  - How much we loose in guessing true Y incorrectly
- If we know the true distribution $p(y, x|w)$ then we choose $\hat{y}$

$$\min_f EL(y, \hat{y}) = \min_{\hat{y}} \int L(y, \hat{y}) p(y, x|w) dx dy$$

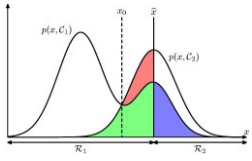


7

## Model selection

Example: Spam classification

- Loss for incorrect classifying mails and spams
  - $L_{12} = 100, L_{21} = 1$






8

## Loss functions

- How to define loss function?
  - No unique choice, often defined by application
  - **Normal practice: Choose the loss related to minus loglikelihood**

Example: Predicting the amount of the product at the storage:

$$L(Y,\hat{y}) = \begin{cases} 10 + \frac{\hat{y}}{Y}, \hat{y} \geq Y \\ 1000, \hat{y} < Y \end{cases}$$

Example: Compute loss function related to
  - Normal distribution

Guess why such loss function was chosen





## Loss functions

- Classification problems
  - Common loss function $L(Y,\hat{y}) = \begin{cases} 0, Y = \hat{y} \\ 1, Y \neq \hat{y} \end{cases}$
  - When minimizing the loss, equivalent to misclassification rate





## Model selection

- **Problem**: true model and true $\boldsymbol{w}$ are unknown→can not compute expected loss!
- How to find an optimal model?
  - Consider what expected loss (**risk**) depends on
    $$R(Y,\hat{y}) = E[L(Y,\hat{y}(X,D))]$$
- Random factors:
  - $D$ – **training set**
  - Y, X – data to be predicted **(validation set)**



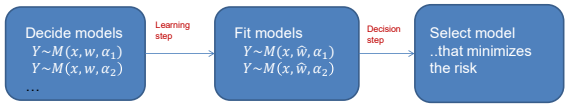


## Holdout method

- Simplify the risk estimation:
  - Fix D as a particular training set T
  - Fix Y,X as a particular validation set V
- Risk becomes (**empirical risk**)
  $$\hat{R}(y,\hat{y}) = \frac{1}{|V|} \sum_{(X,Y) \in V} L(Y,\hat{y}(X,T))$$
  - Estimator is fit by Maximum Likelihood using training set
  - Risk estimated by using validation set
  - Model with minimum empirical risk is selected

## General model selection strategy

- Given data $D = \{\boldsymbol{X}_i, Y_i, i = 1 \dots n\}$




- When fitting data, Maximum Likelihood is usually used
- $\alpha_i$ can be different things:
  - Type of distribution
  - Number of variables in the model
  - Regulatization parameter value
  - …





## Holdout method

Divide into training, validation and test sets

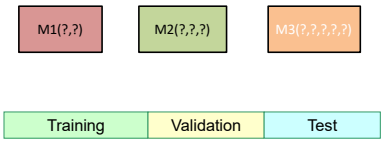

- Choose proportions in some way





## Holdout method

- Given: training, validation, test sets and models to select between

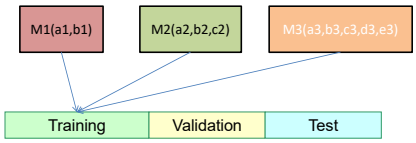






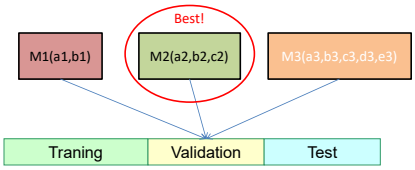

## Holdout method

- Training set is to used for fitting models to the dataset by using maximum likelihood



## Holdout method

- Validation set is used to choose the best model (lowest risk)

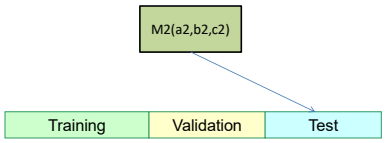





17

## Holdout method

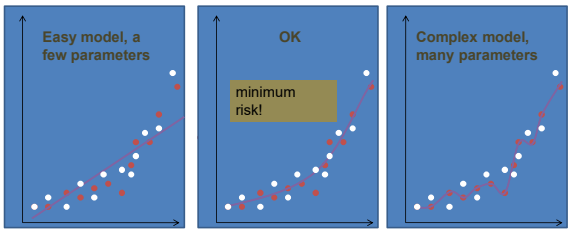
- Test set is used to test a performance on a new data






18

## Holdout method






19

## Holdout in R

- How to partition into train/test?
  - Use set.seed(12345) in the labs to get identical results

```
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
train=data[id,]
test=data[-id,]
```

- How to partition into train/valid/test?

```
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]
```



20

## Bias-variance tradeoff

- Bias of an estimator $Bias(\hat{y}\,(x_0)) = E[\hat{y}(x_0) - f(x_0)]$, $f(x_0)$ is expected response
  - If $Bias(\hat{y}(x_0)) = 0$, the estimator is **unbiased**
  - ML estimators are asymptotically unbiased if the model is enough complex
  - However, unbiasedness does not mean a good choice!

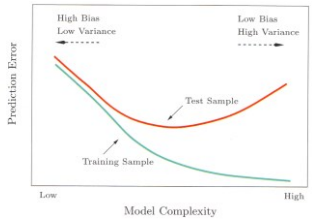




## Bias-variance tradeoff

- Assume loss is $L(Y, \hat{y}) = (Y - \hat{y})^2$

$$R(Y(x_0), \hat{y}(x_0)) = \sigma^2 + Bias^2(\hat{y}(x_0)) + Var(\hat{y}(x_0))$$




When loss is not quadratic, no such nice formula exist





## Cross-validation

- Compared to holdout method:
  - Why do we use only some portion of data for training- can we use more (increase accuracy)?

**Cross-validation** (Estimates Err)

**K-fold cross-validation (rough scheme, show picture)**:

1. Permute the observations randomly
2. Divide data-set in K roughly equally-sized subsets
3. Remove subset #i and fit the model using remaining data.
4. Predict the function values for subset #i using the fitted model.
5. Repeat steps 3-4 for different i
6. CV= squared difference between observed values and predicted values (another function is possible)





## Cross-validation

**Cross-validation**


Note: if $K$=$N$ then method is *leave-one-out* cross-validation.

$$\kappa : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$$

**K-fold cross-validation**: $CV = \frac{1}{N}\sum_{i=1}^{N} L(Y_i, \hat{y}^{-k(i)}(x_i))$

What to do if N is not a multiple of K?

## Cross-validation vs Holdout

- Holdout is easy to do (a few model fits to each data)
- Cross validation is computationally demanding (many model fits)
- Holdout is applicable for large data
  - Otherwise, model selection performs poorly
- Cross validation is more suitable for smaller data





## Analytical methods

- Analytical expressions to select models
  - $AIC$ (Akaike's information criterion)

Idea: Instead of $R(Y,\hat{y}) = E[L(Y,\hat{y}(X,D))]$ consider **in-sample** risk (only $Y$ in $D$ is random):

$$R_{in}(Y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N} E_{Y_i}\left[L\left(Y_i,\hat{y}(X,D)\right)|D, X \in D\right]$$





## Analytical methods

- One can show that

$$R_{in}(Y,\hat{y}) \approx R_{train} + \frac{2}{N}\sum_{i} cov(\hat{y}_i, Y_i)$$

where $R_{train} = \sum_{X_i,Y_i \in T} L(Y_i,\hat{y}_i)$

- Recall, **degrees of freedom** $\mathrm{df}(\mathrm{model}) = \frac{1}{\sigma^2}\sum_i cov(\hat{y}_i, Y_i)$
  - When model is linear, *df* is the number of parameters.
- If loss is defined by minus two loglikelihood,

$$AIC \equiv -2loglik(D) + 2df(model)$$





## Model selection

**Example Computer Hardware Data Set** : performance measured for various processors and also

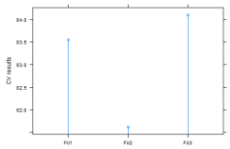
- Cycle time
- Memory
- Channels
- …

Build model predicting performance

# Cross-validatation

- Try models with different predictor sets

```
data=read.csv("machine.csv", header=F)
library(cvTools)

fit1=lm(V9~V3+V4+V5+V6+V7+V8, data=data)
fit2=lm(V9~V3+V4+V5+V6+V7, data=data)
fit3=lm(V9~V3+V4+V5+V6, data=data)
f1=cvFit(fit1, y=data$V9, data=data,K=10,
foldType="consecutive")
f2=cvFit(fit2, y=data$V9, data=data,K=10,
foldType="consecutive")
f3=cvFit(fit3, y=data$V9, data=data,K=10,
foldType="consecutive")
res=cvSelect(f1,f2,f3)
plot(res)
```



# Linear classification methods

Lecture 2a



1

## Overview

- Elements of decision theory
- Logistic regression
- Discriminant Analysis models



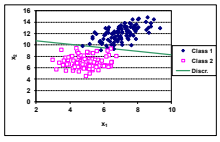
2

## Classification

- Given data $D = ((X_i, Y_i), i = 1 \dots N)$
  - $Y_i = Y(X_i) = C_j \in \boldsymbol{C}$
  - Class set $\boldsymbol{C} = (C_1, \dots, C_K)$

**Classification problem:**

- Decide $\hat{Y}(x)$ that maps **any** $x$ into some class $C_K$
  - Decision boundary






3

## Classifiers

- **Deterministic**: decide a rule that directly maps $X$ into $\hat{Y}$
- **Probabilistic:** define a model for $P(Y = C_i|X), i = 1 \dots K$

**Disanvantages of deterministic classifiers**:

- Sometimes simple mapping is not enough (risk of cancer)
- Difficult to embed loss-> rerun of optimizer is often needed
- Combining several classifiers into one is more problematic
  - Algorithm A classifies as spam, Algorithm B classifies as not spam → ???
  - P(Spam|A)=0.99, P(Spam|B)=0.45 → better decision can be made



4

## Bayesian decision theory

- Machine learning models estimate $p(y|x)$ or $p(y|x, \hat{w})$
- Transform probability into action→ which value to predict?→decision step
  - $p(Y = Spam|x) = 0.83$→do we move the mail to Junk?
  - What is more dangerous: deleting 1 non-spam mail or letting 1 spam mail enter Inbox?
- →**Loss function** or **Loss matrix**





## Loss matrix

- Costs of classifying $Y = C_k$ to $C_j$:
  - Rows: true, columns: predicted

$$L = \|L_{ij}\|, i = 1, \dots, n, j = 1, \dots, n$$

- Example 1: 0/1-loss

$$L = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- Example 2: Spam

$$L = \begin{pmatrix} 0 & 100 \\ 1 & 0 \end{pmatrix}$$



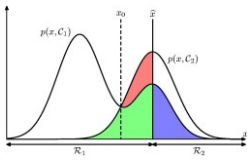


## Loss and decision

- Expected loss minimization
  - $R_j$ : classify to $C_j$

$$EL = \sum_k \sum_j \int_{R_j} L_{kj} p(\boldsymbol{x}, C_k) d\boldsymbol{x}$$

- **Choose such $R_j$ that $EL$ is minimized**
- Two classes




$$EL = \int_{R_1} L_{21} p(x, C_2) dx + \int_{R_2} L_{12} p(x, C_1)\, dx$$

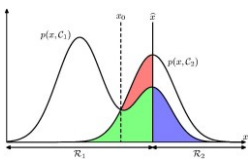




## Loss and decision

- Loss minimization

$$\min_f EL(y, \hat{f}) = \min_f \int L(y, \hat{f}) p(y, x|w) dx dy$$




When loss is
$\begin{cases} 1, wrongly\ classified \\ 0, correctly\ classified \end{cases}$

Classify $Y$ as
$\hat{Y} = \arg\max_c p(Y = c|X)$

## Loss and decision

- How to minimize *EL with two classes?*
- Rule:
  - $L_{12}p(x, C_1) > L_{21}p(x, C_2)$ →predict $y$ as $C_1$
- 0/1 Loss: classify to the class which is more probable!

$$\frac{p(C_1|x)}{p(C_2|x)} > \frac{L_{21}}{L_{12}} \rightarrow predict\ y\ as\ C_1$$

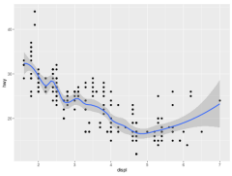




## Loss and decision

- Continuous targets: squared loss
  - Given a model $p(x, y)$, minimize

$$EL = \int L\left(y, \hat{Y}(x)\right) p(x, y) dxdy$$



- Using **square loss**, the optimal is posterior mean

$$\hat{Y}(x) = \int yp(y|x)dy$$





## ROC curves

- Binary classification
- The choice of the thershold $\hat{x} = \frac{L_{21}}{L_{12}}$ affects prediction→ what if we don't know the loss? Which classifier is better?
- **Confusion matrix**

| | | PREDICTED | | |
|---|---|---|---|---|
| | | 1 | 0 | Total |
| TRUE | 1 | TP | FN | $N_+$ |
| | 0 | FP | TN | $N_-$ |





## ROC curves

- **True Positive Rates** (TPR) = **sensitivity** = **recall**
  - Probability of detection of positives: TPR=1 positives are correctly detected

$$TPR = TP/N_+$$

- **False Positive Rates** (FPR)
  - Probability of false alarm: system alarms (1) when nothing happens (true=0)

$$FPR = FP/N_-$$

- **Specificity**

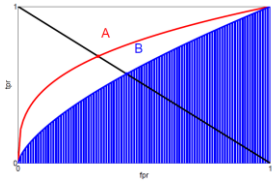
$$Specificity = 1 - FPR$$

- **Precision**

$$Precision = \frac{TP}{TP + FP}$$

## ROC curves

- **ROC**=Receiver operating characteristics
- Use various thresholds, measure TPR and FPR
- Same FPR, higher TPR→ better classifier
- Best classifier = greatest Area Under Curve (**AUC**)








## Types of supervised models

- **Generative models**: model $p(X|Y,w)$and $p(Y|w)$
  - Example: k-NN classification

$$p(X = x|Y = C_i, K) = \frac{K_i}{N_i V}, p(C_i|K) = \frac{N_i}{N}$$

From Bayes Theorem,

$$p(Y = C_i|x, K) = \frac{K_i}{K}$$

- **Discriminative models**: model $p(Y|X,w)$, $X$ constant
  - Example: logistic regression
  - $p(Y = 1|\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$





## Generative vs Discriminative

- Generative can be used to generate new data
- Generative normally easier to fit (check Logistic vs K-NN)
- Generative: each class estimated separately→ do not need to retrain when a new class added
- Discriminative models: can replace $X$ with $\phi(X)$ (preprocessing), method will still work
  - Not generative, distribution will change
- Generative: often make too strong assumptions about $p(X|Y,w)$ → bad performance





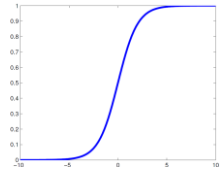
## Logistic regression

- Discriminative model
- Model for binary output
  - $C = \{C_1 = 1, C_2 = 0\}$
    $p(Y = C_1|X) = sigm(\boldsymbol{w}^T\boldsymbol{x})$

$$sigm(a) = \frac{1}{1+e^{-a}}$$

- Alternatively
  $Y \sim Bernoulli(sigm(a)), a = \boldsymbol{w}^T\boldsymbol{x}$
  $sigm(a) = \frac{1}{1+e^{-a}}$

What is $P(Y = C_2|X)$?



## Logistic regression

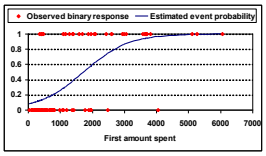
- Logistic model- yet another form

$$ln\frac{p(Y=1|X=x)}{P(Y=0|X=x)} = ln\frac{p(Y=1|X=x)}{1-P(Y=1|X=x)} = logit(p(Y=1|X=x)) = \boldsymbol{w^T x}$$

**The log of the odds is linear in $\boldsymbol{x}$**

- Here $logit(t) = \ln\left(\frac{t}{1-t}\right)$
- Note $p(Y|X)$ is connected to $\boldsymbol{w^T x}$ via logit link

Example: Probability to buy more than once as function of First Amount Spend






17

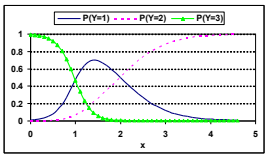
## Logistic regression

- When Y is categorical,

$$p(Y = C_i|x) = \frac{e^{w_i^T x}}{\sum_{j=1}^{K} e^{w_j^T x}} = softmax(\boldsymbol{w_i^T x})$$

- Alternatively

$$Y \sim Multinoulli\left(softmax(\boldsymbol{w_1^T x}), \ldots softmax(\boldsymbol{w_K^T x})\right)$$






18

## Logistic regression

**Fitting logistic regression**

- In binary case,

$$\log P(D|w) = \sum_{i=1}^{N} y_i \log(sigm(w^T x_i)) + (1-y_i)\log(1-sigm(w^T x_i))$$

  - Can not be maximized analytically, but unique maximizer exists
- To maximize loglikelihood, optimization used
  - Newton's method traditionally used (Iterative Reweighted Least Squares)
  - Steepest descent,Quasi-newton methods…

**Estimation:**

For new $x$, estimate $p(y) = [p_1, \ldots p_C]$ and classify as $\arg\max_i p_i$

Decision boundaries of logistic regression are linear



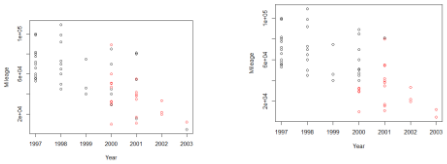
19

## Logistic regression

- In R, use glm() with family="binomial"
  - Predicted probabilities: predict(fit,newdata, type="response")

Example Equipment=f(Year, mileage)

**Original data** **Classified data**



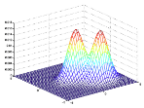



20

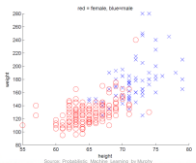
## Quadratic discriminant analysis

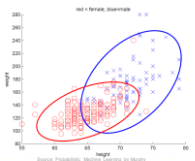
- Generative classifier
- Main assumptions:
  - $\boldsymbol{x}$ is now **random** as well as $y$

$$p(\boldsymbol{x}|y = C_i, \theta) = N(\boldsymbol{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$$



Unknown parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\}$





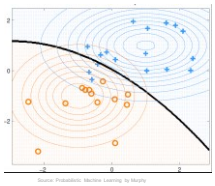


21

## Quadratic discriminant analysis

- If parameters are estimated, classify:

$$\hat{y}(\boldsymbol{x}) = \arg\max_c p(y = c|\boldsymbol{x}, \theta)$$



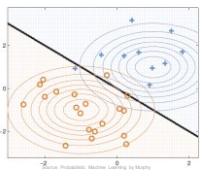


22

## Linear discriminant analysis (LDA)

- Assumtion $\Sigma_i = \Sigma, i = 1, \dots K$
- Then $p(y = c_i|x) = softmax(w_i^T x + w_{0i})$ →exactly the same form as the logistic regression
  - $w_{0i} = -\frac{1}{2}\boldsymbol{\mu}_i^T \Sigma^{-1} \boldsymbol{\mu}_i + \log \pi_i$
  - $w_i = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i$
- Decision boundaries are linear
  - **Discriminant function**:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$





23

## Linear discriminant analysis (LDA)

- Difference LDA vs logistic regression??
  - Coefficients will be estimated differently! (models are different)
- How to estimate coefficients
  - find MLE.

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{i:y_i=c} \mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T$$

$$\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{k} N_c \hat{\Sigma}_c$$

  - Sample mean and sample covariance are MLE!
  - If class priors are parameters (**proportional priors**),

$$\hat{\pi}_c = \frac{N_c}{N}$$



24

## LDA and QDA: code

- Syntax in R, library **MASS**

lda(formula, data, ..., subset, na.action)
- Prior – class probabiliies
- Subset – indices, if training data should be used

qda(formula, data, ..., subset, na.action)

predict(..)





## LDA: output

```
resLDA=lda(Equipment~Mileage+Year, data=mydata)
 print(resLDA)
```

```
> print(resLDA)
Call:
lda(Equipment ~ Mileage + Year, data = mydata)

Prior probabilities of groups:
        0         1
0.6440678 0.3559322

Group means:
   Mileage     Year
0 63539.21 1998.447
1 36857.62 2000.762

Coefficients of linear discriminants:
                LD1
Mileage -1.500069e-05
Year     5.745893e-01
```



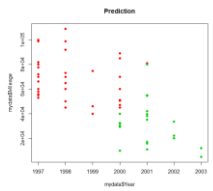


## LDA: output

- Misclassified items

```
> table(Pred$class, mydata$Equipment)

   0  1
0 31  6
1  7 15
```

```
plot(mydata$Year, mydata$Mileage,
col=as.numeric(Pred$class)+1, pch=21,
bg=as.numeric(Pred$class)+1,
main="Prediction")
```

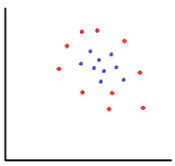







## LDA versus Logistic regression

- Generative classifiers are easier to fit, discriminative involve numeric optimization
- LDA and Logistic have same model form but are fit differently
- LDA has stronger assumptions than Logistic, some other generative classifiers lead also to logistic expression
- New class in the data?
  - Logistic: fit model again
  - LDA: estimate new parameters from the new data
- Logistic and LDA: complex data fits badly unless interactions are included



## LDA versus Logistic regression

- LDA (and other generative classifiers) handle missing data easier
- Standardization and generated inputs:
  - Not a problem for Logistic
  - May affect the performance of the LDA in a complex way
- Outliers affect $\Sigma$ → LDA is not robust to gross outliers
- LDA is often a good classification method even if the assumption of normality and common covariance matrix are not satisfied.

# Lecture 2a

block 2

Splines

Generalized additive models



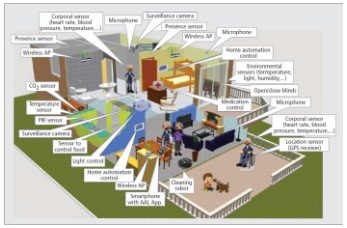
1

## Moving beyond simple models

- Sometimes using simple models (linear regression) is not enough
  - Too simple →***more flexible models are needed***

Example: Ambient Assisted Living

- digitally connected and controlled devices for support of people with special needs
- emergency buttons, pressure emergency services with connection to a broader smart home

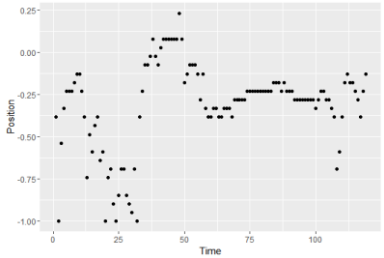




2

## Moving beyond simple models

- Ambient Assisted Living
  - Person's movement is detected by Radio Signal Strength (RSS) measurements→how to remove noise?

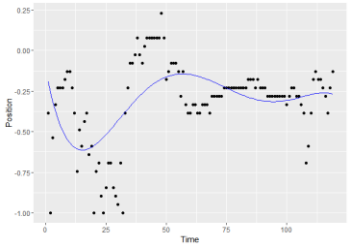





3

## Moving beyond simple models

- Attempt 1: 5th degree polynomial



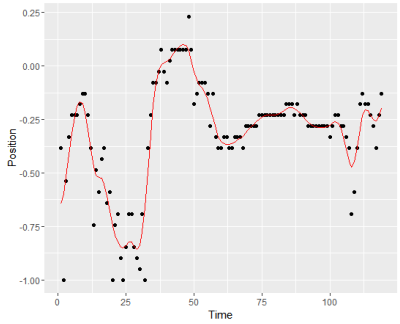

Model underfits data→need more flexible models



4

# Moving beyond simple models

- Using smoothing splines








# Basis function expansion

If $y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 e^{-x_2} + \epsilon,$

Model becomes linear if to recompute:

$$\phi_1(x_1) = x_1$$
$$\phi_2(x_1) = x_1^2$$
$$\phi_3(x_1) = e^{-x_2}$$

- Any model of the type $Ey = \sum_i w_i \phi_i(\boldsymbol{x})$ can be fit by linear regression!



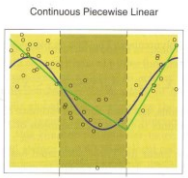


# Constructing a piecewise linear function

Method A. Introduce linear functions on each interval and a set of constraints

(4 free parameters)

$$\begin{cases} y_1 = \alpha_1 x + \beta_1 \\ y_2 = \alpha_2 x + \beta_2 \\ y_3 = \alpha_3 x + \beta_3 \\ y_1(\xi_1) = y_2(\xi_1) \\ y_2(\xi_2) = y_3(\xi_2) \end{cases}$$




Method B. Use a basis expansion (4 free parameters)

$$h_1(X) = 1,\ h_2(X) = X,\ h_3(X) = (X - \xi_1)_+,\ h_4(X) = (X - \xi_2)_+$$

Theorem. The two methods are equivalent.





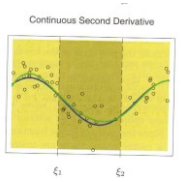
# Splines

- A piecewise polynomial is called an **order-M** (or degree *M*-1) **spline** if it is continuous and has continuous derivatives up to order *M*-2 at the knots.
- Equivalent: An order-*M* spline with *K* knots:

$$h_j(X) = X^{j-1},\ j = 1, \ldots, M$$
$$h_{M+l}(X) = (X - \xi_l)_+^{M-1},\ l = 1, \ldots, K$$

- An order-4 (degree-3) spline is called a **cubic spline**

In cubic splines, knot discontinuity is not visible



## Natural cubic spline

- A cubic spline $f$ is called **natural cubic spline** if its 2nd and 3rd derivatives are zero at $a$ and $b$

Note that $f$ is linear on extreme intervals

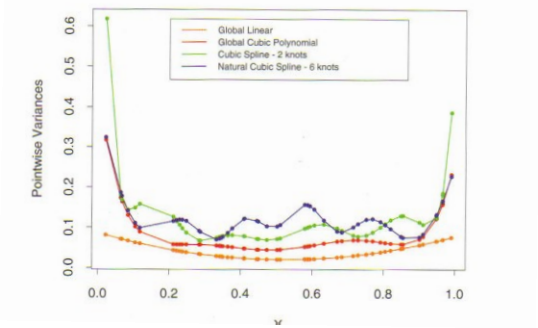
Basis functions of natural cubic splines

$$N_1(X)=1,\ N_2(X)=X,\ N_{k+2}=d_k(X)-d_{K-1}(X),\quad k=1,\ldots,K-2$$

where $d_k(X)=\dfrac{(X-\xi_k)_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_k}$



## Variance of spline estimators – boundary effects






## Fitting smooth functions to data

- Minimize

$$RSS(f,\lambda)=\sum_{i=1}^{N}(y_i-f(x_i))^2+\lambda\int\{f''(t)\}^2\,dt$$

where λ is **smoothing parameter**.

$\lambda=0$ : any function interpolating data

$\lambda=+\infty$ : least squares line fit



## Optimality of smoothing splines

- The function $f$ minimizing $RSS$ for a given $\lambda$ is a natural cubic spline with knots at all unique values of $x_i$ (NOTE: $N$ knots!)
- Minimizing sum of squares:

$$f(x)=\sum_{j=1}^{N}N_j(x)\theta_j=N(x)^T\Theta$$

$$RSS(\Theta,\lambda)=(\mathbf{y}-\mathbf{N}\Theta)^T(\mathbf{y}-\mathbf{N}\Theta)+\lambda\Theta^T\Omega_N\Theta$$

$$\{\mathbf{N}\}_{ij}=N_j(x_i)\quad\{\Omega_N\}_{ij}=\int N_i''(t)N_j''(t)dt$$

$$\hat{\Theta}=(\mathbf{N}^T\mathbf{N}+\lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y}$$

## A smoothing spline is a linear smoother

- Smoothing spline

$$\hat{f} = \mathbf{N}\left(\mathbf{N}^T\mathbf{N} + \lambda\Omega_N\right)^{-1}\mathbf{N}^T\mathbf{y} = \mathbf{S}_\lambda\mathbf{y}$$

  is a **linear smoother**.

- Compare with other smoothers, such as linear regression.



## Degrees of freedom

- It can be shown that

$$\mathbf{S}_\lambda = \left(\mathbf{I} + \lambda\mathbf{K}\right)^{-1}$$

  where $\mathbf{K}$ is **penalty matrix**
- Eigenvalue decomposition of $\mathbf{K}$ :

$$\mathbf{S}_\lambda = \sum_{k=1}^{N}\rho_k(\lambda)\mathbf{u}_k\mathbf{u}_k^T$$

$$\rho_k(\lambda) = \frac{1}{1+\lambda d_k}$$

- $d_k$ and $\mathbf{u}_k$ are eigenvalues and eigenvectors



## Smoothing splines and shrinkage

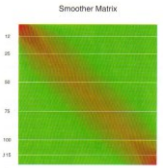
$$\mathbf{S}_\lambda\mathbf{y} = \sum_{k=1}^{N}\mathbf{u}_k\rho_k(\lambda)\mathbf{u}_\mathbf{k}^\mathbf{T}\mathbf{y}$$

- Smoothing spline decomposes vector **y** with respect to basis of eigenvectors and shrinks respective contributions
- The eigenvectors ordered by $\rho$ increase in complexity. The higher the complexity, the more the contribution is shrunk.



## Penalty and degrees of freedom

- $df_\lambda = trace(\mathbf{S}_\lambda) \rightarrow \; df_\lambda = \sum_{k=1}^{N}\frac{1}{1+\lambda d_k}$
- $\lambda$ increase → $df_\lambda$ decrease
- higher $\lambda$ → higher penalization.
- Smoother matrix is has banded nature → local fitting method



## Automated selection of smoothing parameters

**What can be selected:**

Regression splines

- Degree of spline
- Placement of knots

Smoothing spline

- Penalization parameter



## Automated selection of smoothing parameters

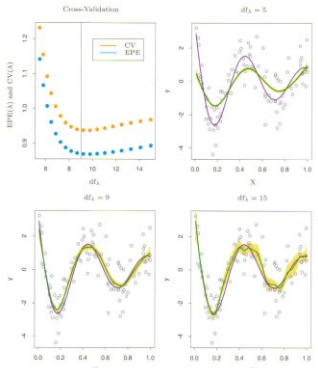
$$df_\lambda = trace(\mathbf{S}_\lambda) = \sum_{k=1}^{N} \frac{1}{1+\lambda d_k}$$

- Use either $df_\lambda$ or $\lambda$
  - Given $df_\lambda$ → solve equation → find $\lambda$
- Use holdout principle or cross validation for parameter tuning



## Automated selection of smoothing parameters

- Bias-variance tradeoff






## Multidimensional splines

**How to fit data smoothly in higher dimensions?**

- Formulate a new problem

$$\min \sum_i (y_i - f(x_i))^2 + \lambda J[f]$$

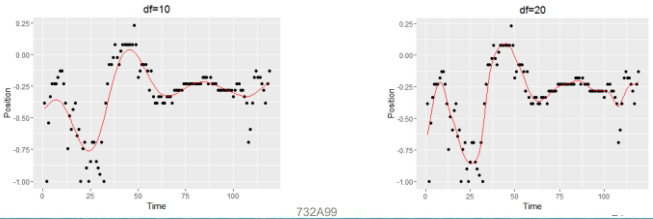
- The solution is **thin-plate splines**
- The solution in 2 dimensions is essentially sum of radial basis functions

$$f(x) = \beta_0 + \beta^T x + \sum \alpha_j \eta(\|x - x_j\|)$$

# Splines: R code

- Smoothing splines : smooth.spline()
- Natural clubic splines: ns() in **splines**
- Thin plate splines: Tps() in **fields**

```
res1=smooth.spline(data$Time,data$RSS_anchor2,df=10)
predict(res1,x=data$Time)$y
```








# Generalized additive models

- Model

$$Y \sim EF(\mu, \dots)$$

where

- $g(\mu) = \alpha + s_1(X_1) + s_2(X_2) + s_p(X_p)$
- $s_i(X)$ - smoothers, normally splines
- EF – distribution from exponential family
- $g$ – Link function

- Often linear terms are often included separately

$$EY = \alpha + s_1(X_1) + \dots + s_p(X_p) + \sum_{j=1}^{q} \beta_j X_{p+j}$$

**Example**: EF= normal, EF=Bernoulli (logistic)





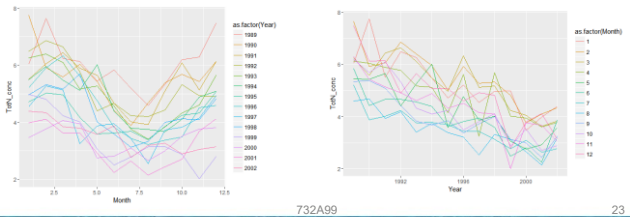
# Generalized additive models

- Sometimes even higher orders are included (thin-plate splines)

$$g(\mu) = \alpha + s_1(X_1) + \dots + s_p(X_p) + \sum_{j=1}^{q} \beta_j X_{p+j} + s_{12}(X_1, X_2)$$

- Method is reasonable to apply when additivity is observed or admissble

Example: Total Nitrogen level in Rhine river








# Estimation of additive models

Estimation by MLE

$$g(\mu) = \alpha + f_1(x_1) + \dots + f_p(x_p)$$

The backfitting algorithm for Normal model

$$1.\text{Initialize}: \hat{\alpha} = \frac{1}{N}\sum_{i=1}^{N} y_i, \quad \hat{f}_j \equiv 0,\ j = 1,\dots,p$$

$$2.\text{Cycle}: j = 1,\dots,p,1,\dots,p,\dots,1,\dots,p$$

$$\hat{f}_j \leftarrow s_j\left[\left\{(y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\right\}\right]$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N}\sum_{i=1}^{N} \hat{f}_j(x_{ij})$$

$\lambda$ in each term can be estimated by CV

## Generalized additive models

- Example: Modelling the concentration of total nitrogen at Lobith on the Rhine
  - There are seasonal trends (GAM reasonable)
  - Variables
    - Nitrogen level
    - Year
    - Month
- R: package **mgcv** (also package **gam**)
  - gam(formula, family,data,select, method)
    - Select allows for term (variable) selection
  - predict(), plot(), summary()…
  - s(k, sp)
    - k should be the same as the amount of **unique values** of this variable in **smoothing splines**
    - sp - smoothing penalty.

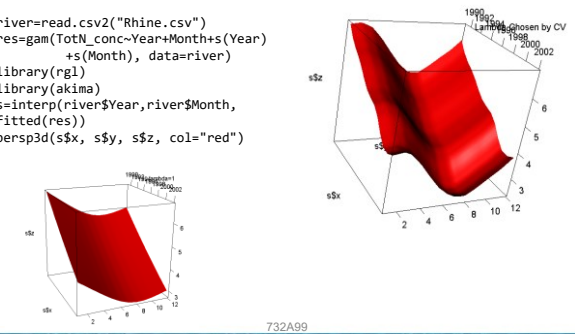




## Generalized additive models

- R code

```
river=read.csv2("Rhine.csv")
res=gam(TotN_conc~Year+Month+s(Year)
        +s(Month), data=river)
library(rgl)
library(akima)
s=interp(river$Year,river$Month,
fitted(res))
persp3d(s$x, s$y, s$z, col="red")
```








## Generalized additive models

```
> summary(res)

Family: gaussian
Link function: identity

Formula:
TotN_conc ~ Year + Month + s(Year) + s(Month)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.0008852  0.0009512   0.931   0.3535
Year        0.0014169  0.0003421   4.142 5.63e-05 ***
Month       0.2517641  0.1048467   2.401   0.0175 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Year)  6.049  7.206 66.72  <2e-16 ***
s(Month) 4.476  5.611 35.45  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Rank: 19/21
R-sq.(adj) =  0.819   Deviance explained = 83.2%
GCV = 0.27689  Scale est. = 0.25638   n = 168
> res$sp
    s(Year)    s(Month)
0.003342167 0.007087835
```

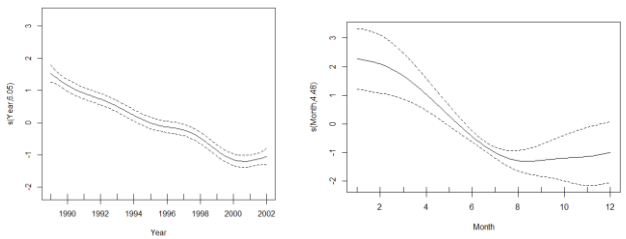




## Generalized additive models

- Seeing trend and seasonal pattern

plot(res)



# Naïve Bayes classifiers
# Decision trees

Lecture 2b



1

## 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=No,1=Yes)
    - $X_1$ corr. "space", $X_2$ corr. "fun",...


- Want to classify a new document



2

## 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



3

## 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!



4

## Naive Bayes classifiers

- Naive Bayes assumption: conditional independence

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

- 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(Space,ship|SciFi)= P(Space|SciFi)*P(Ship|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 $\theta_{ijk}$ and $\pi_k$ 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

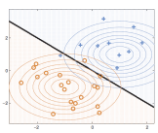
| Tid | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | 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 $p(\boldsymbol{x}|y=C_i,\theta)=N(\boldsymbol{x}|\boldsymbol{\mu_i},\boldsymbol{\Sigma_i})$
  - $\boldsymbol{\Sigma_i}$=diag$(\sigma_{i1}^2,\dots,\sigma_{ip}^2)$

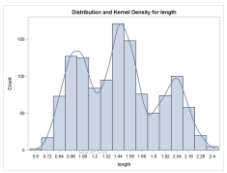


- Naive bayes is a special case of LDA (given A)
  - →MLE are means and variances (per class)

## 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






9

## 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
```



10

## Decision trees

**Idea**

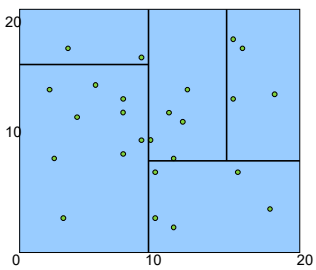
Split the domain of feature set into the set of hypercubes (rectangles, cubes) and define the target value to be constant within each hypercube

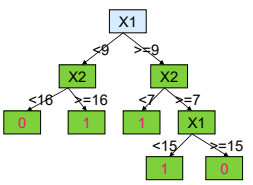
- Regression trees:
  - Target is a continuous variable
- Classification trees
  - Target is a class (qualitative) variable



11

## Classification tree toy example



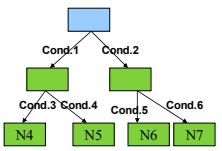






12

## Definitions

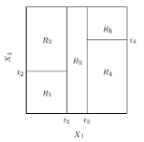
- Root node
- Nodes
- Leaves (terminal nodes)
- Parent node, child node
- Decision rules
- A value is assigned to the leaves

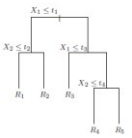


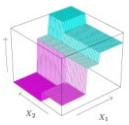



13

## Regression tree toy example









14

## A classification problem

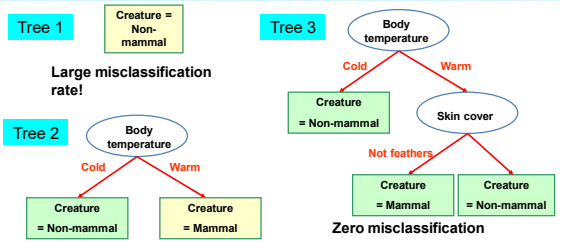
*Create a classification tree that would describe the following patterns*

| 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 |



15

## Several solutions




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



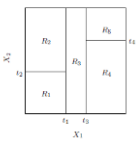
16

## Decision trees

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

**Model:**

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






17

## Decision trees

Example:

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



18

## Classification trees

- Target is categorical
- Classification probability $\mathrm{p_{mk}} = p(Y = k|X \in R_m)$ 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

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



19

## Classification trees

- Impurity measure $\mathrm{Q}(R_m)$
  - $R_m$ 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 $\mathrm{Q}(R_m)\ N(R_m)$

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



20

## 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

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

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**

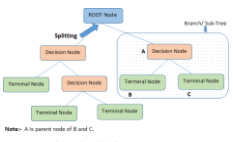
**Weakest link pruning:**

1. Merge two leaves that have smallest *N(parent)*Q(parent)-N(leave1)Q(leave1)-N(leave2)Q(leave2)*
2. For the current tree T, compute

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

$|T|$ =#leaves

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




How to find the optimal $\alpha$? Cross validation!





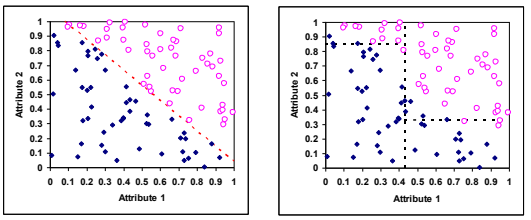
## Decision trees: comments

- 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→ totally different tree
- Greedy algorithms → fit may be not so good
- Lack of smoothness

## Decision trees: issues

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






25

## 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)
```

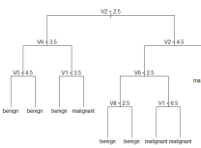


26

## 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
```



27

## 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
```

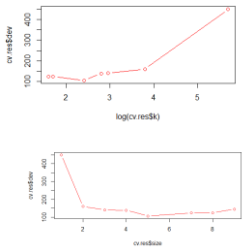


28

## 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?

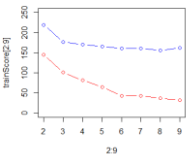


29

## 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?



30

## 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
```



31

# Lecture 2b

block 2

High-dimensional problems



1

## Wide data

- **Wide data** $p \gg n$ . Many variables, few data points.
  - **Genomics**
  - **Text**
- **Tall data**: $p \ll n$ . Few variables, many data points.

Most of applications

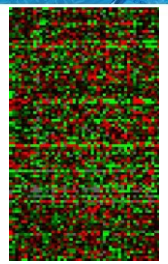
- Economics, for ex. Currency exchange rates vs time
- Industry, Car performance characteristics vs probability of malfunctioning
- Surveys, customer satisfaction vs survey answers

- Tall and Wide. Supermarket scanners. Many purchases, many products.



2

## Genomics-microarrays



Hastie et al:. *DNA microarray data: expression matrix of 6830 genes (rows) and 64 samples (columns), for the human tumor data. Only a random sample of 100 rows are shown. The display is a heat map, ranging from bright green (negative, under expressed) to bright red (positive, over expressed). Missing values are gray. The rows and columns are displayed in a randomly chosen order.*



3

## Text – document classification

| Document | has('ball') | has('EU') | has('political_arena') | wordlen | Lex. Div. | Topic |
|---|---|---|---|---|---|---|
| Article1 | Yes | No | No | 4.1 | 5.4 | Sports |
| Article2 | No | No | No | 6.5 | 13.4 | Sports |
| ⋮ | ⋮ | ⋮ | | ⋮ | ⋮ | ⋮ |
| ArticleN | No | No | Yes | 7.4 | 11.1 | News |



4

## A problem with wide data

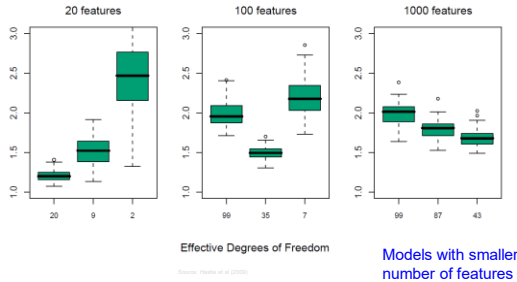
- Linear regression $\mu = w^T x, Y \sim N(\mu, \sigma^2)$
- ML solution $\hat{w} = (X^T X)^{-1} X^T Y$
  - $X$ is $n \times p$, has rank $n$
  - $X^T X$ is $p \times p$, has rank $n$
  - → $X^T X$ is not invertible!
- Solutions:
  - **Dimensionality reduction**: PCA, PCR
  - **Shrinkage**: Lasso, Ridge, Elastic network
  - **Forward variable selection**
- Algorithms need sometimes be modified for wide data.





## Effective amount of features for wide data

- Linear response generated with different p, n=100
- Ridge is applied with different $\lambda$




Models with smaller effective number of features have better prediction





## Classification: LDA

- Standard LDA

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{i:y_i=c} \mathbf{x}_i, \quad \hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_c)^T$$

$$\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{k} N_c \, \hat{\Sigma}_c$$

- → $\Sigma^{-1}$ does not exist…





## Classification: diagonal-covariance LDA

- Data is not enough to estimate dependences in covariance
- For wide data, we do **diagonal-covariance LDA** (naive Bayes):

$$\Sigma = diag(\sigma_1^2, .. \sigma_p^2)$$

- Discriminant function

$$\delta(x^{new}) = -\sum_{j=1}^{p} \frac{\left(x_j^{new} - \bar{x}_{kj}\right)^2}{s_j^2} + 2 \log \pi_k$$

  - $s_j^2 = \frac{1}{n} \sum_i n_i var(x_j | Y = C_i)$
  - $\bar{x}_{kj} = mean(x_j | Y = C_k), \bar{x}_j = mean(x_j)$
- Classify to the highest discriminant function value
- **Drawback**: all features are in the model→ difficult to use in interpretations.

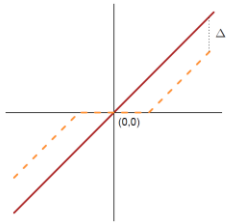
# Classification: NSC

**Nearest Shrunken Centroids**

- Idea: Shrink classwise means towards overall mean

1. Compute $d_{kj} = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k(s_j + s_0)}$
2. Shrink $d'_{kj} = sign(d_{kj})(|d_{kj}| - \Delta)_+$
3. Set $x'_{kj} = \bar{x}_j + m_k(s_j + s_0)d'_{kj}$

Only features with nonzero $d'_{kj}$ contribute to classification!→insignificant features are shrunk!




Define Δ by CV



# NSC: example

- LSVT Voice Rehabilitation Data Set
  - Target: Quality of voice rehabilitation
    - 1=acceptable, 2=not acceptable
  - Features: Properties of the signal (voice)
  - n=126, p=309



# NSC: example

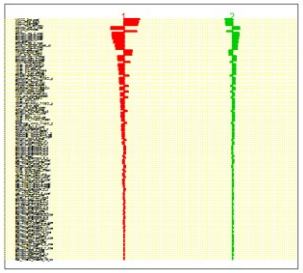
- Package **pamr**
  - pamr.train()
  - pamr.cv

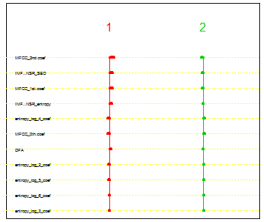
```
data0=read.csv2("voice.csv")
data=data0
data=as.data.frame(scale(data))
data$Quality=as.factor(data0$Quality)
library(pamr)
rownames(data)=1:nrow(data)
x=t(data[,-311])
y=data[[311]]
mydata=list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)), genenames=rownames(x))
model=pamr.train(mydata,threshold=seq(0,4, 0.1))
pamr.plotcen(model, mydata, threshold=1)
pamr.plotcen(model, mydata, threshold=2.5)

a=pamr.listgenes(model,mydata,threshold=2.5)
cat( paste( colnames(data)[as.numeric(a[,1])], collapse='\n' ) )

cvmodel=pamr.cv(model,mydata)
print(cvmodel)
pamr.plotcv(cvmodel)
```



# NSC: example

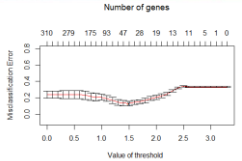
- Centroid plot, Δ= 1 and Δ= 2.5





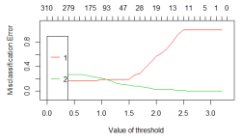
# NSC: example

```
> pamr.listgenes(model,mydata,threshold=2.5)
      id  1-score 2-score
 [1,] 86  0.0897  -0.0449          MFCC_2nd.coef
 [2,] 80  0.0702  -0.0351          IMF..NSR_SEO
 [3,] 85  0.0652  -0.0326          MFCC_1st.coef
 [4,] 82  0.0517  -0.0259          IMF..NSR_entropy
 [5,] 153 -0.0507 0.0253          entropy_log_4_coef
 [6,] 84  -0.05   0.025           MFCC_0th.coef
 [7,] 60  0.0359  -0.0179          DFA
 [8,] 151 -0.0316 0.0158          entropy_log_2_coef
 [9,] 154 -0.0299 0.0149          entropy_log_5_coef
[10,] 155 -0.0193 0.0096          entropy_log_6_coef
[11,] 152 -0.018  0.009           entropy_log_3_coef
```




- Confusion matrix optimal Δ

| | Pred 1 | Pred 2 |
|---|---|---|
| True 1 | 33 | 9 |
| True 2 | 5 | 79 |






# RDA

**Regularized discriminant analysis**

- Another way of solving singularity of Σ
  - $\gamma$ is some constant

$$\hat{\Sigma}(\gamma) = \gamma\hat{\Sigma} + (1-\gamma)diag(\hat{\Sigma})$$

- $\gamma = 0$→diagonal-covariance LDA
- $\gamma$ is chosen by CV
- R: rda() in **klaR**



# Regularized logistic regression

- Usual logistic regression

$$p(Y = C_i|x) = \frac{e^{w_{i0}+\boldsymbol{w}_i^T\boldsymbol{x}}}{\sum_{j=1}^{K} e^{w_{j0}+\boldsymbol{w}_j^T\boldsymbol{x}}} = softmax(w_{i0} + \boldsymbol{w}_i^T\boldsymbol{x})$$

- Lp -Regularization:

$$\max_{w} \sum_{i=1}^{n} \log p(Y_i|x_i) - \frac{\lambda}{2}\sum_{k=1}^{K} \|\boldsymbol{w}_i\|^p$$

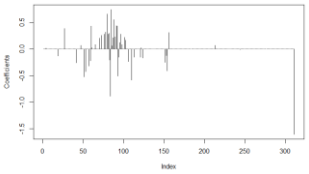
- Parameter redunancy is solved
- **L1 regularization:** some w are shrunk to 0
- Numerical optimization is used to solve
- R: LiblineaR() in package **LiblineaR**



# L1 logistic regression

- Voice rehabilitation

```
W=model2$W
plot(t(W), type="h", ylab="Coefficients")
```




| | Pred 1 | Pred 2 |
|---|---|---|
| True 1 | 41 | 1 |
| True 2 | 0 | 84 |

Overfitted?

## SVM

- Support Vector Machine do not suffer from $p \gg n$ problem
  - Largest margin can be found even if the data is perfectly separable



## Computational shortcuts p>>n

- SVD decomposition $X = UDV^T = RV^T$
- If model is linear in parameters and has quadratic penalties:
  - Transform data observations from X into R
  - Minimize loss (minus log likelihood) with R instead of X and get $\boldsymbol{\theta}$
  - Original parameters $\boldsymbol{w} = V\boldsymbol{\theta}$
- Can be applied to many methods

- Example: ridge regression



## Elastic net

- L1 regularization

$$\min_w -\log p(D|\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_1$$

$$\|w\|_1 = \sum_i |w_i|$$

- For p>n, LASSO can extract at most $n$ nonzero components
  - Severe regularization if $p \gg n$
- L1 regularization→selects some feature among the correlated ones
- L2 regularization→ w's of the correlated variables are shrunk towards each other are nonzero



## Elastic net

Combine L1 and L2 to diminish effect of L1 regularization.

- Elastic net regularization:

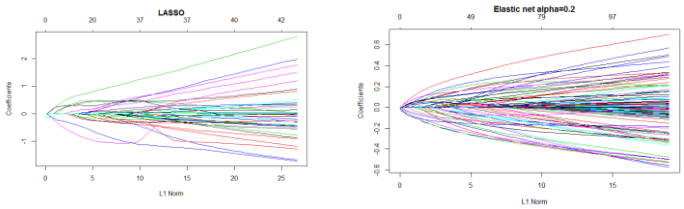
$$\min_w -\log p(D|\boldsymbol{w}) + \lambda(\alpha\|\boldsymbol{w}\|_1 + (1-\alpha)\|w\|_2)$$

- $\alpha$ is set ad hoc or chosen by CV
- Elastic net may select more than $n$ features

- R: glmnet() in **glmnet** package
  - Specify "family" for classification or regression

# Elastic net

- Voice rehabilitation






# Comparative analysis

- Gene expression data

| Methods | CV errors (SE) Out of 144 | Test errors Out of 54 | Number of Genes Used |
|---|---|---|---|
| 1. Nearest shrunken centroids | 35 (5.0) | 17 | 6,520 |
| 2. $L_2$-penalized discriminant analysis | 25 (4.1) | 12 | 16,063 |
| 3. Support vector classifier | 26 (4.2) | 14 | 16,063 |
| 4. Lasso regression (one vs all) | 30.7 (1.8) | 12.5 | 1,429 |
| 5. $k$-nearest neighbors | 41 (4.6) | 26 | 16,063 |
| 6. $L_2$-penalized multinomial | 26 (4.2) | 15 | 16,063 |
| 7. $L_1$-penalized multinomial | 17 (2.8) | 13 | 269 |
| 8. Elastic-net penalized multinomial | 22 (3.7) | 11.8 | 384 |



# When features are not available

- Sometimes it is difficult to define or use the feature set
  - Molecule
  - Text document
    - possible, but can be very high dimensional
- ..but a proximity measure $K(x, x')$ is easier to define
  - Ex: How much one document is different from another one

→ We can compute similarity matrix $\boldsymbol{K = XX^T}$



# When features are not available

- Many methods can use $\boldsymbol{K}$ instead of $\boldsymbol{X}$
  - Note: p is not involved in calculations!!
- SVM: kernel trick→ $\boldsymbol{K}$ can be used directly
- K-Nearest neighbors
  - Transform similarity into distance $d_{ij}^2 = K(x_i, x_i) + K(x_i, x_j) - 2K(x_i, x_j)$
  - Use distances to find neighbors
- Can also be done for
  - Logistic and multinomial regression with L2 penalty
  - LDA
  - PCA: kernel PCA

## Kernel PCA

- Usual PCA
  - Center $\boldsymbol{X}$
  - Find $\boldsymbol{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}, \boldsymbol{S} = [p \times p]$
    - $\boldsymbol{u}_i$ has dimension $p$
  - Project data on PCs: $Z = X\,U$

- Problems: $X$ is unknown, and it can be $p$ can be very large



## Kernel PCA

- Kernel PCA: Equivalent formulation

1. Solve $\boldsymbol{K}'\boldsymbol{a_i} = \lambda_i'\boldsymbol{a_i}, i = 1,..M$
   - $\boldsymbol{K} = \|K(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1,..n\|$
   - Centering $\boldsymbol{K}' = \boldsymbol{K} - \boldsymbol{1_n K} - \boldsymbol{K 1_n} + \boldsymbol{1_n K 1_n}$
   - $\lambda_i = \lambda_i'/n$
2. Scores for $PC_i$: $z_i(\boldsymbol{x}) = \sum_{i=1}^{n} a_{in} K(\boldsymbol{x}, \boldsymbol{x}_n)$

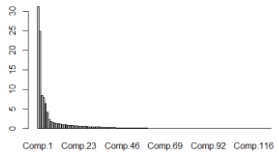
- There are at most *n* eigenvectors even if *p>>n*

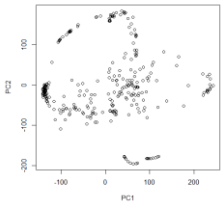


## Kernel PCA in R

- Use kpca() in **kernlab**

```
library(kernlab)
K <- as.kernelMatrix(crossprod(t(x)))
res=kpca(K)
barplot(res@eig)
plot(res@rotated[,1], res@rotated[,2], xlab="PC1",
ylab="PC2")
```






## Feature assessment

- Which features are important?
  - Ex: Which protein values differ between normal and cancer samples

- P-values in our predictive models can not be computed (too few observations)

- → Traditional hypothesis testing is used

## Feature assessment

- Individual gene: t-test

$H_{0j}$: *treatment has no effect on gene j*
$H_{1j}$: *treatment has an effect on gene j*

$$t = \frac{\bar{x}_{2j} - \bar{x}_{1j}}{se_j}$$

- Alternatively, nonparametric tests (permutation tests) can be used to compare two populations
- Testing hypothesis for all genes?→multiple hypothesis testing
- Control family-wise error rate
  - Bonferroni correction: $\alpha' = \alpha/M$
  - Ex: $\alpha$=0.05, $M$=12000→$\alpha' \approx 10^{-6}$

In practice, no genes with such small p-values

mean: $\bar{x}_{1j}$

| | $X_j$ | Y |
|---|---|---|
| | ... | 0 |
| | ... | 0 |
| | | ... |
| | ... | 1 |
| | ... | 1 |

mean: $\bar{x}_{2j}$



29

## Feature assessment

- Hypothesis testing Voice Rehabilitation
  - Feature "MFCC_2nd.coef"

```
res=t.test(MFCC_2nd.coef~Quality,data=data,
alternative="two.sided")
res$p.value
```

```
> res$p.value
[1] 1.21246e-11
```

```
res=oneway_test(MFCC_2nd.coef~as.factor(Qu
ality), data=data,paired=FALSE)
pvalue(res)
```

```
> pvalue(res)
[1] 3.166942e-09
```



30

## Feature assessment

- Alternative: **false discovery rate** (FDR)
  - Can not be exactly computed in practice

| | Called nonsignif | Called signif | Total |
|---|---|---|---|
| H0 true | U | V | M0 |
| H0 false | T | S | M1 |
| Total | M-R | R | M |

$$FDR = E\left(\frac{V}{R}\right)$$



31

## Feature assessment

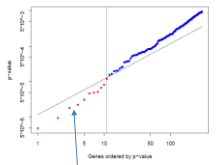
- Benjamini-Hochberg method (BH method)
  - Shown that $FDR(BH) < \alpha$ for independent hypotheses
  - →we can control FDR!

**Algorithm 18.2** *Benjamini–Hochberg (BH) Method.*

1. Fix the false discovery rate $\alpha$ and let $p_{(1)} \le p_{(2)} \le \cdots \le p_{(M)}$ denote the ordered $p$-values
2. Define

$$L = \max\left\{j : p_{(j)} < \alpha \cdot \frac{j}{M}\right\}.$$

3. Reject all hypotheses $H_{0j}$ for which $p_j \le p_{(L)}$, the BH rejection threshold.



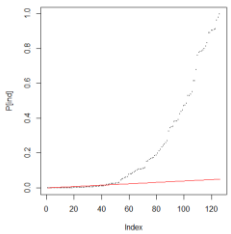
Rejected hypotheses



32

# Feature assessment

- Voice rehabilitation




```
> cat( paste( Feats, collapse='\n' ) )
MFCC_2nd.coef
IMF..NSR_SEO
MFCC_1st.coef
IMF..NSR_entropy
MFCC_0th.coef
DFA
Log.energy
HNR..HNR_dB_Praat_std
MFCC_3rd.coef
VFER..SNR_SEO
IMF..SNR_TKEO
IMF..SNR_entropy
Jitter..pitch_PQ5_classical_Schoentgen
Jitter..pitch_percent
Jitter..F0_abs_dif
Jitter..F0_PQ5_classical_Schoentgen
Jitter..F0_dif_percent
VFER..SNR_TKEO1
Shimmer..Ampl_TKEO_prc25
Shimmer..Ampl_AM
VFER..NSR_TKEO1
Shimmer..Ampl_abs0th_perturb
VFER..SNR_TKEO
NHR..NHR_Praat_std
OQ..std_cycle_closed
VFER..NSR_SEO
Jitter..F0_FM
Jitter..pitch_FM
Shimmer..Ampl_TKEO_prc75
Jitter..pitch_abs
OQ..std_cycle_open
Jitter..F0_TKEO_prc5
GNE..std
Jitter..pitch_PQ5_generalised_Schoentgen
Jitter..F0_TKEO_mean
PPE
Shimmer..Ampl_TKEO_prc95
Xist.delta
Jitter..F0_PQ5_generalised_Schoentgen
Shimmer..Ampl_PQ3_generalised_Schoentgen
Shimmer..Ampl_PQ5_generalised_Schoentgen
Shimmer..Ampl_PQ11_generalised_Schoentgen
Jitter..pitch_TKEO_prc25
```

# Generalized Linear Models. Uncertainty estimation

Lecture 2c

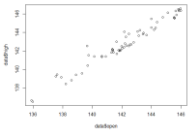


1

## Moving beyond typical distributions

- We know how to model
  - Normally distributed targets -> linear regression
  - Bernoulli and Multinomial targets→logistic regression
  - What if target distribution is more complex?

**Example 1**: Daily Stock prices NASDAQ
- Open
- High (within day)



Does it seem that the error is normal here?

**Example 2**: Number of calls to bank
- Y=Number of calls
- X= time

Endless amount of classes → multinomial does not work… (Poisson)



2

## Exponential family

- More advanced error distributions are sometimes needed!
- Many distributions belong to **exponential** family:
  - Normal, Exponential, Gamma, Beta, Chi-squared..
  - Bernoulli, Multinoulli, Poisson…

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x})g(\boldsymbol{\eta})e^{(\boldsymbol{\eta}^T u(\boldsymbol{x}))}$$

- Easy to find MLE and MAP
- Non-exponential family distributions: uniform, Student t

Example: Bernoulli



3

## Generalized linear models

- Assume $Y$ from the exponential family
- **Model** is $Y \sim EF(\mu, \dots),\ f(\mu) = \boldsymbol{w}^T\boldsymbol{x}$
  - Alt $\mu = f^{-1}(\boldsymbol{w}^T\boldsymbol{x})$
  - $f^{-1}$ is activation function
  - $f$ is link function (in principle, arbitrary)
- Arbitrary $f$ will lead to (s – dispersion parameter)

$$p(y|w,s) = h(y,s)g(\boldsymbol{w},\boldsymbol{x})e^{\frac{b(\boldsymbol{w},\boldsymbol{x})y}{s}}$$

- If $f$ is a canonical link, then

$$p(y|w,s) = h(y,s)g(\boldsymbol{w},\boldsymbol{x})e^{\frac{(\boldsymbol{w}^T\boldsymbol{x})y}{s}}$$

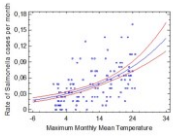


4

## Generalized linear models

- Canonical links are normally used
  - MLE computations simplify
  - MLE $\hat{w} = F(X^T Y)$ → computations do not depend on all data but rather a summary (sufficient statistics)→ computations speed up

**Example**: Poisson regression
$f^{-1}(\mu) = e^{\mu}, Y \sim Poisson\left(e^{\boldsymbol{w^T x}}\right)$





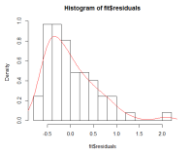

5

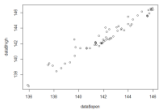
## Generalized linear model: software

- Use **glm**(formula, family, data) in R

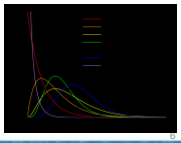
**Example**: Daily Stock prices NASDAQ
- Open
- High (within day)

1. Try to fit usual linear regression, study histogram of residuals



Gamma distribution: Wikipedia

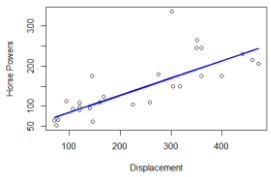




6

## Least absolute deviation regression

- Model $Y \sim Laplace(w^T X, b)$
  - Member of exponential family
- Equivalent to minimizing sum of absolute deviations
- Properties
  - Robust to outliers
  - Sensitive to changes in data
  - Multiple solutions possible




- R: package **L1pack**



7

## Probabilistic models

- Why it is beneficial to assume a **probabilistic** model?
- A common approach to modelling in CS and engineering:

$$y = f(x, w)$$

- $f$ is known, $w$ is unknown
- Fit model to data with least squares, optimization or ad hoc→ find $w$

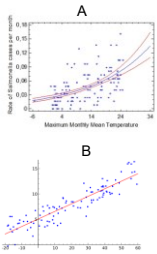


8

## Probabilistic models

Arguments against deterministic models:

- The model does not really describe actual data (error is not explained)
  - No difference between modelling data A (Poisson) and B (Normal)
  - Estimation strategy for A is not good for B
- The model typically gives a **deterministic answer**, no information about uncertainty
  - "…The exchange rate tomorrow will be 8.22 …" 😉






9

## Probabilistic models

Probabilistic model

$$Y \sim Distribution(f(x, w), \theta)$$

- Data is fully explained (error as well)
- Automatic principle for finding parameters: MLE , MAP or Bayes theorem
- Automatic principle for finding uncertainty (conf. limits)
  - **Bootstrap**
  - Posterior probability
- Possibility to generate new data of the same type
  - Further testing of the model



10

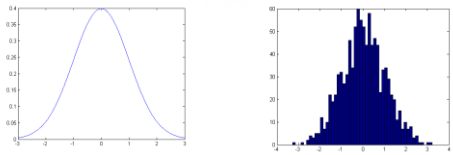
## Uncertainty estimation

- Given estimator $\hat{f} = \hat{f}(x, D)$ (or $\hat{\boldsymbol{\alpha}} = \delta(D)$), how to estimate the uncertainty?
- Answer 1: if the distribution for data *D* is given, compute analytically the distribution for the estimator→ derive confidence limits
  - Often difficult
  - Example: In simple linear regression, $\hat{\alpha}$ follows *t* distribution
- Answer 2: Use **bootstrap**



11

## The bootstrap: general principle



We want to determine uncertainty of $\hat{f}(D, X)$

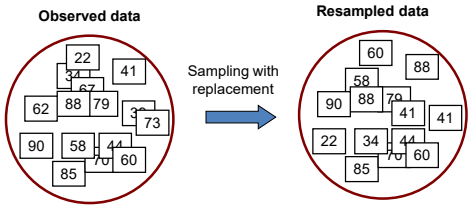
1. Generate many different $D_i$ from their distribution
2. Use histogram of $\hat{f}(D_i, X)$ to determine confidence limits→ unfortunately can not be done (distr of *D is often unknown)*

**Instead**: Generate many different $D_i^*$ from the empirical distribution (histogram)



12

## Nonparametric bootstrap




$\bar{x}$ $\quad \bar{x}_1^*, \bar{x}_2^*, \ldots, \bar{x}_N^*$



13

## Nonparametric bootstrap

Given estimator $\hat{w} = \hat{f}(D)$

Assume $X \sim F(X, w)$, $F$ and $w$ are unknown

1. Estimate $\hat{w}$ from data **D**=($X_1$,...$X_n$)
2. Generate $\mathbf{D_1}$ =($X^*_1$,...$X^*_n$) by sampling with replacement
3. Repeat step 2 $B$ times
4. The distribution of $w$ is given by $\hat{f}(D_1), \ldots \hat{f}(D_B)$

Nonparametric bootstrap can be applied to any deterministic estimator, distribution-free



14

## Parametric bootstrap

Given estimator $\hat{w} = \hat{f}(D)$

Assume $X \sim F(X, w)$, $F$ is known and $w$ is unknown

1. Estimate $\hat{w}$ from data **D**=($X_1$,...$X_n$)
2. Generate $\mathbf{D_1}$ =($X^*_1$,...$X^*_n$) by generating from $F(X, \hat{w})$
3. Repeat step 2 $B$ times
4. The distribution of $w$ is given by $\hat{f}(D_1), \ldots \hat{f}(D_B)$

Parametric bootstrap is **more** precise if the distribution form is correct



15

## Uncertainty estimation

1. Get $D_1$ , ... $D_B$ by bootstrap
2. Use $\hat{f}(D_1)$ , ... $\hat{f}(D_B)$ to estimate the uncertainty
   - Boostrap percentile
   - Bootstrap Bca
   - ...

- Bootstrap works for all distribution types
- Can be bad accuracy for small data sets $n < 40$ (empirical is far from true)
- Parametric bootstrap works even for small samples



16

## Bootstrap confidence intervals

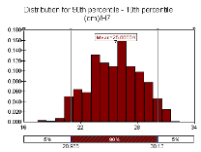
- To estimate 100(1-α) confidence interval for $w$

Bootstrap percentile method

1. Using bootstrap, compute $\hat{f}(D_1)$, … $\hat{f}(D_B)$, sort in ascending order, get $w_1 \ldots w_B$
2. Define $A_1$=ceil(B α/2), $A_2$=floor(B-B α/2)
3. Confidence interval is given by

$$(w_{A_1}, w_{A_2})$$

Look at the plot…







## Bootstrap: regression context

- Model $Y \sim F(X, w)$
- Data $D = \{(Y_i, X_i), i = 1, \ldots, n\}$
- Idea: produce several bootstrap sets that are similar to D

Nonparametric bootstrap:

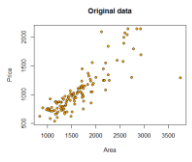
1. Using observation set **D**, sample **pairs** $(X_i, Y_i)$ with replacement and get bootstrap sample $\mathbf{D_1}$
2. Repeat step 1 $B$ times→ get $\mathbf{D_1}$,… $\mathbf{D_B}$





## Uncertainty estimation

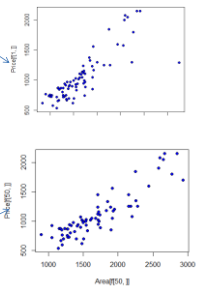
**Example:** Albuquerque dataset:
Y=Price of House
X=Area (sqft)




We sample data index, from {1…N}

$D_1$

$D_{50}$







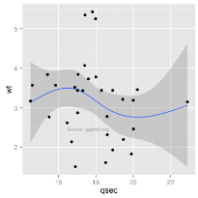
## Bootstrap: regression context

Parametric bootstrap

1. Fit a model to D → get $\hat{w}(D)$.
2. Set $X_i^* = X_i$, generate $Y_i^* \sim F(X_i, \hat{w})$.
3. $D_i = \{(X_i^*, Y_i^*), i = 1, \ldots, n\}$
4. Repeat step 2 $B$ times

## Confindence intervals in regression

- Given $Y \sim Distribution(y|x,w), EY|X = \mu|x = f(x,w)$
  - Example: $Y \sim N(w^T x, \sigma^2), \mu|x = f(x,w) = w^T x$
- Estimate intervals for $\mu|x = f(x,w)$ for many X, combine in a **confidence band**
- What is estimator?
  - $\mu|x = f(x,w)$






21

## Confindence intervals in regression

**Estimation**

1. Compute $D_1, \dots D_B$ using a bootstrap
2. Fit model to $D_1, \dots D_B$ →estimate $\hat{w}_1, \dots \hat{w}_B$
3. For a given X, compute $f(X, \hat{w}_1), \dots f(X, \hat{w}_B)$ and estimate confidence interval by (percentile method)
4. Combine confidence intervals in a band



22

## Bootstrap: R

- Package **boot**
  - **Functions:**
    - boot()
    - boot.ci() – 1 parameter
    - envelope() – many parameters
- Random random generation for parametic bootstrap:
  - Rnorm()
  - Runif()
  - …

boot(data, statistic, R, sim = "ordinary", ran.gen = function(d, p) d, mle = NULL,…)



23

## Bootstrap: R

Nonparametric bootstrap:

- Write a function *statistic* that depends on *dataframe* and *index* and returns the estimator

```
library(boot)
data2=data[order(data$Area),]#reordering data according to Area

# computing bootstrap samples
f=function(data, ind){
  data1=data[ind,]# extract bootstrap sample
  res=lm(Price~Area, data=data1) #fit linear model
  #predict values for all Area values from the original data
  priceP=predict(res, newdata=data2)
  return(priceP)
}
res=boot(data2, f, R=1000) #make bootstrap
```



24

## Bootstrap: R

Parametric bootstrap:

- Compute value *mle* that estimates model parameters from the data
- Write function *ran.gen* that depends on *data* and *mle* and which generates new data
- Write function *statistic* that depend on *data* which will be generated by *ran.gen* and should return the estimator





## Bootstrap

```
mle=lm(Price~Area, data=data2)

rng=function(data, mle) {
  data1=data.frame(Price=data$Price, Area=data$Area)
  n=length(data$Price)
#generate new Price
  data1$Price=rnorm(n,predict(mle, newdata=data1),sd(mle$residuals))
  return(data1)
}

f1=function(data1){
  res=lm(Price~Area, data=data1) #fit linear model
  #predict values for all Area values from the original data
  priceP=predict(res,newdata=data2)
  return(priceP)
}

res=boot(data2, statistic=f1, R=1000, mle=mle,ran.gen=rng, sim="parametric")
```

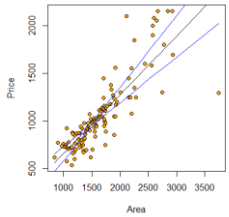




## Uncertainty estimation: R

- Bootstrap cofidence bands for linear model

```
e=envelope(res) #compute confidence bands

fit=lm(Price~Area, data=data2)
priceP=predict(fit)

plot(Area, Price, pch=21, bg="orange")
points(data2$Area,priceP,type="l") #plot fitted line

#plot cofidence bands
points(data2$Area,e$point[2,], type="l", col="blue")
points(data2$Area,e$point[1,], type="l", col="blue")
```








## Prediction bands

- Confidence interval for Y|X= interval for mean $EY|X$
- Prediction interval for Y|X= interval for $Y|X$

$$Y \sim Distribution(x, w)$$

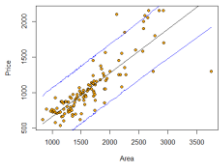
Prediction band for parametric bootstrap

1. Run parametric bootstrap and get $D_1, \dots D_B$
2. Fit the model to the data and get $\hat{w}(D_1), \dots \hat{w}(D_B)$
3. For each X, generate from $Distribution\big(X, \hat{w}(D_1)\big), \dots Distribution\big(X, \hat{w}(D_B)\big)$ and apply percentile method
4. Connect the intervals→get the band

# Estimation of the model quality

Example: parametric bootstrap

```
mle=lm(Price~Area, data=data2)

f1=function(data1){
  res=lm(Price~Area, data=data1) #fit
linear model
  #predict values for all Area values
from the original data
  priceP=predict(res,newdata=data2)
  n=length(data2$Price)
  predictedP=rnorm(n,priceP,
sd(mle$residuals))
  return(predictedP)
}
res=boot(data2, statistic=f1, R=10000,
mle=mle,ran.gen=rng, sim="parametric")
```




Why wider band?

# Lecture 2d

Latent variable models

1

## Overview

- Principal Component Analysis (PCA)
- Probabilistic PCA
- Independent component analysis (ICA)



2

## Latent variables

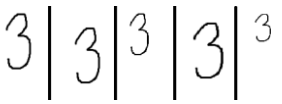
- Sometimes data depends on the variables we can not measure (hard to measure)
  - Answers on the test depend on Intelligence
  - Brain activity in the brain is measured by sensors
  - Stock prices depend on market confidence




3

## Latent variables

- Latent factor discovered→ data storage may decrease a lot
- Latent factors
  - Center
  - Scaling
- Original vs compressed
  - 100x100x5=50000
  - 100x100+2*5+2*5=10020





4

## Principal Component Analysis (PCA)

- *PCA* is a technique for reducing the complexity of high dimensional data
- It can be used to approximate high dimensional data with a few dimensions (latent features) –> much less data to store
- New variables might have a special interpretation

**Applications**

- Image recognition
- Information compression
- Subspace clustering
- …



5

## Principal Component Analysis (PCA)

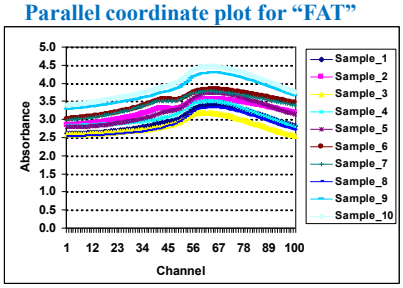
- Example 1: Hadwritten digits
  - Can we get a more compact summary?



6

## Absorbance records for ten samples of chopped meat

**Parallel coordinate plot for "FAT"**




**1 target (fat)**

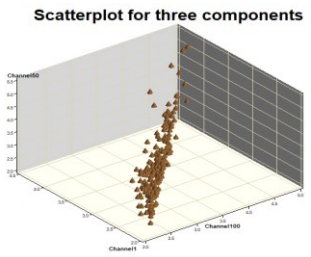
**100 features (absorbance at 100 wavelengths or channels)**

**The features are strongly correlated to each other**



7

## 3-D plots of absorbance records for samples of meat - channels 1, 50 and 100






8

## Principal components analysis

**Idea**: Introduce a new coordinate system (PC1, PC2, …) where

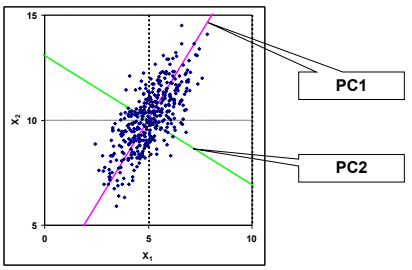
- The first principal component (PC1) is the direction that maximizes the variance of the projected data
- The second principal component (PC2) is the direction that maximizes the variance of the projected data after the variation along PC1 has been removed
- The third principal component (PC3) is the direction that maximizes the variance of the projected data after the variation along PC1 and PC2 has been removed
- ….

In the new coordinate system, coordinates corresponding to the last principal components are very small → can take away these columns





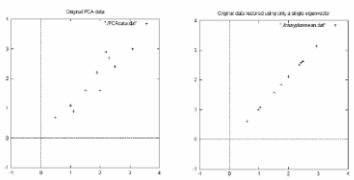
## Principal Component Analysis - two inputs








## PCA- after reducing dimensionality



- Data became approximate (but less data to store)
- $PC_1, \ldots PC_M$ are actually eigenvectors of **sample covariance** (first largest eigenvalue,…,Mth largest egenvalue)

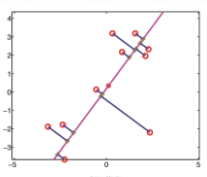




## PCA: another view

- Aim: minimize the distance between the original and projected data

$$\min_{V} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$



## PCA: computations

Data $D = \left\| \mathbf{x}_1 \; \mathbf{x}_2 \; \ldots \mathbf{x}_p \right\|, \quad \mathbf{x}_i = (x_{i1}, \ldots, x_{in})$

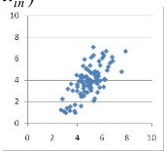
1. Centred data

$$X = \left\| \mathbf{x}_1 - \bar{\mathbf{x}}_1 \; \mathbf{x}_2 - \bar{\mathbf{x}}_2 \; \ldots \mathbf{x}_p - \bar{\mathbf{x}}_p \right\|,$$

2. Covariance matrix

$$\boldsymbol{S} = \frac{1}{N} \boldsymbol{X}^T \boldsymbol{X}$$

3. Search for eigenvectors and eigenvalues of **S**




| | Column 1 | Column 2 |
|---|---|---|
| Column 1 | 0.951 | 0.905 |
| Column 2 | 0.905 | 1.883 |

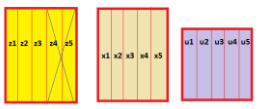


13

## PCA: computations

4. Coordinates of any data point x=($x_1$…$x_p$) in the new coordinate system:
$z = (z_1, \ldots z_n)$, $z_i = x^T u_i$

Matrix form: $Z = X\,U$




5. Discard principle components after some *M*

6. New data will have dimensions N x M instead of N x p

Getting approximate original data:
$X' = ZU_M^T$

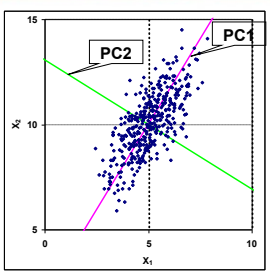
Store: N x M+ p x M instead N x p

100*50 vs
100*4+50*4



14

## Principal Component Analysis




Eigenanalysis of the Covariance Matrix

| | | |
|---|---|---|
| Eigenvalue | 2.8162 | 0.3835 |
| Proportion | 0.880 | 0.120 |
| Cumulative | 0.880 | 1.000 |

| Variable | PC1 | PC2 |
|---|---|---|
| X1 | 0.523 | 0.852 |
| X2 | 0.852 | -0.523 |

**Loadings (U)**



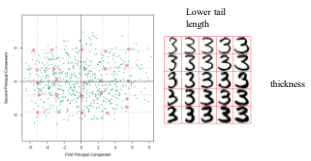
15

## Principal Component Analysis

- Digits: two eigenvectors extracted




- Interptretation of eigenvectiors





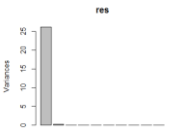

16

## PCA in R

- Prcomp(), biplot(), screeplot()

```
mydata=read.csv2("tecator.csv")
data1=mydata
data1$Fat=c()
res=prcomp(data1)
lambda=res$sdev^2
#eigenvalues
lambda
#proportion of variation
sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res)
```

```
> lambda
  [1] 2.612713e+01 2.385369e-01 7.844883e-02 3.018501e-
  [7] 2.052212e-04 1.084213e-04 2.077326e-05 1.150359e-

> sprintf("%2.3f",lambda/sum(lambda)*100)
  [1] "98.679" "0.901"  "0.296"  "0.114"  "0.006
  [9] "0.000"  "0.000"  "0.000"  "0.000"  "0.000
```




Only 1 component captures the 99% of variation!



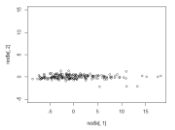
17

## PCA in R

- Principal component **loadings (U)**

```
U=res$rotation
head(U)
```

```
> head(U)
              PC1        PC2        PC3
Channel1 0.07938192 0.1156228 0.08073156 -0.0927
Channel2 0.07987445 0.1170972 0.07887873 -0.0981
Channel3 0.08036498 0.1185571 0.07702127 -0.1031
Channel4 0.08085611 0.1200006 0.07515015 -0.1077
Channel5 0.08135022 0.1214075 0.07323819 -0.1119
```



- Data in (PC1, PC2) – **scores (Z)**

```
plot(res$x[,1], res$x[,2], ylim=c(-5,15))
```

Do we need second dimension?

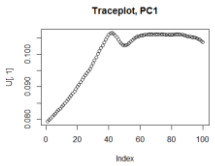


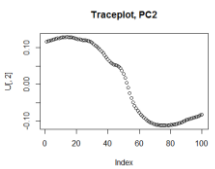
18

## PCA in R

- Trace plots

```
U=loadings(res)
plot(U[,1], main="Traceplot, PC1")
plot(U[,2],main="Traceplot, PC2")
```




Which components contribute to PC1-2?



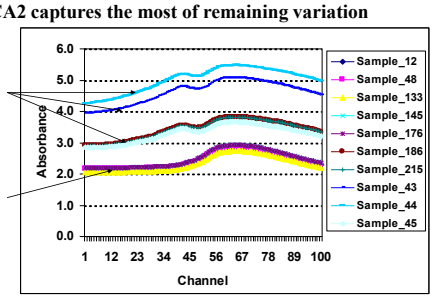
19

## Absorbance records for ten samples of chopped meat

**PCA2 captures the most of remaining variation**

High fat samples

Low fat samples






20

## PCA for high-dimensional data

- Standard PCA for p>>N
  - At most N eigenvalues are nonzero
  - Running time is $O(p^3)$
- High-dimensional PCA
  1. Use $S' = \frac{1}{N} XX^T$ (instead of $S = \frac{1}{N} X^T X$)
  2. Eigenvalues do not change
  3. Eigenvectors of $S$ are $X^T v_i$





## Probabilistic PCA

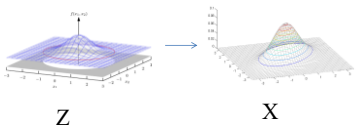
- $z_i$-latent variables, $x_i$- observed variables

$$\boldsymbol{z} \sim \boldsymbol{N}(0, \boldsymbol{I})$$
$$\boldsymbol{x}|\boldsymbol{z} \sim \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{Wz} + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$$

- Alternatively

$$\boldsymbol{z} \sim \boldsymbol{N}(0, \boldsymbol{I}), \boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{Wz} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma^2 I})$$

- **Interpretation**: Observed data (X) is obtained by rotation, scaling and translation of standard normal distribution (Z) and adding some noise.



Z X






## Probabilistic PCA

- **Aim**: extract $Z$ from $X$
- Distribution of $x$:

$$\boldsymbol{x} \sim \boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{C})$$
$$\boldsymbol{C} = \boldsymbol{WW^T} + \boldsymbol{\sigma^2 I}$$

- Rotation invariance
  - Assume that $x$ was generated from $z' = Rz, RR^T = I,\ p(x)$ does not change!

$$\boldsymbol{x}|\boldsymbol{z'} \sim \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{Wz'} + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$$

  - **Model will not be able find latent factors uniquely! ⊗**
    - It does not distinguish z from z'





## Probabilistic PCA

- Estimation of parameters: ML

**Theorem**. ML estimates are given by

$$\mu_{ML} = \bar{x}$$
$$W_{ML} = U_M (L_M - \sigma_{ML}^2 I)^{\frac{1}{2}} R$$
$$\sigma_{ML}^2 = \frac{1}{p-M} \sum_{i=M+1}^{p} \lambda_i$$

- $U_M$ matrix of M eigenvectors
- $L_M$ diagonal matrix of $M$ eigenvalues
- $R$ any orthogonal matrix

## Probabilistic PCA

- Estimation of $Z$
  - Use mean of posterior

$$\hat{z} = (W_{ML}^T W_{ML} + \sigma_{ML}^2 I)^{-1} W_{ML}^T (x - \mu)$$

- Connection to standard PCA
  - Assume $R = I, \sigma^2 = 0$→get standard PCA components scaled by inverse root of eigenvalues

$$Z = XUL^{-\frac{1}{2}}$$



25

## Advantages of probabilistic PCA

- More settings to specify→ more flexible
- Can be faster when M<<p
- Missing values can be handled
- M can be derived if a Bayesian version is used
- Probabilistic PCA can be applied to classification problems directly
- Probabilistic PCA can generate new data



26

## Probabilistic PCA in R

- Use **pcaMethods** from Bioconductor
- Install
  - source("https://bioconductor.org/biocLite.R")
  - biocLite("pcaMethods")

Ppca(data, nPcs,…)

**Results**: scores, loadings…

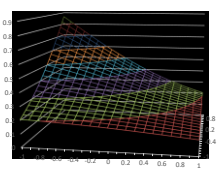


27

## Independent component analysis (ICA)

- Probabilistic PCA does not capture latent factors
  - Rotation invariance
- Let's choose distribution which is not rotation invariant→will get unique latent factors
- Choose non-Gaussian $p_i(z) = p(z)$
- Assuming latent features are **independent**

$$p(z) = \prod_{i=1}^{M} p_i(z_i)$$





28

## ICA

- Model

$$x = \mu + Wz + \epsilon, \qquad \epsilon \sim \boldsymbol{N}(0, \Sigma)$$

- **Estimation A: Maximum likelihood** ($V = W^{-1}$)
  - Assuming noise-free x

$$\max_V \sum_{i=1}^{n} \sum_{j=1}^{p} \log\left(p_j(v_j^T x_i)\right)$$

Subject to $||v_i|| = 1$





## ICA

- Setting $G_j(z) = -\log\left(p_j(z)\right), z_j = v_i^T \boldsymbol{x}$ and assuming large sample

$$\min_V \sum_{j=1}^{p} E(G_j(z_j))$$

Subject to $||v_i|| = 1$

- **Prewhitening**
  - Use PCA: $X' = XU$
  - Computing $z_i$s for given $V$: $Z = X'V$





## ICA

- **Estimation B**: maximize negentropy
  - ICA looks for model which is as much non-Gaussian as possible
- **Entropy** $H(z) = -\int p(z) \log p(z)\, dz = E\left(-\log p\,(z)\right)$
- **Negentropy** $J(z_i) = H(z_i') - H(z_i)$
  - $z_i' \sim N(Ez_i, var(z_i))$
- **Negentropy maximization**

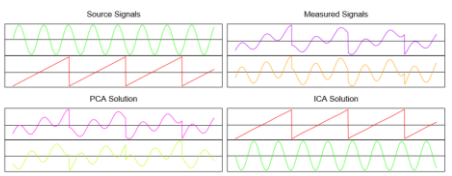
$$\max_V \sum_{j=1}^{p} J(z_j) = \min_V \sum_{j=1}^{p} H(z_j) = \min_V \sum_{j=1}^{p} E\left(-\log p\,(z_j)\right)$$





## ICA

- Example



# Independent component analysis: R

```
S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5)) #Generate data
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
X <- S %*% A
a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
     method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE) #ICA
```