$$\mathbf{Y} = X\beta + \varepsilon,\ \mathbb{E}[\varepsilon] = 0,\ \mathrm{Cov}(\varepsilon) = \mathbb{E}[\varepsilon\varepsilon^\intercal] = \sigma^2 I_{n\times n}.$$

This means that assumption 1.-4. from section 1.3.2 are satisfied. It can then be shown that:

(i) $\mathbb{E}[\hat\beta] = \beta$: that is, $\hat\beta$ is unbiased

(ii) $\mathbb{E}[\hat{\mathbf{Y}}] = \mathbb{E}[\mathbf{Y}] = X\beta$ which follows from (i). Moreover, $\mathbb{E}[r] = 0$.

(iii) $\mathrm{Cov}(\hat\beta) = \sigma^2(X^\intercal X)^{-1}$

(iv) $\mathrm{Cov}(\hat{\mathbf{Y}}) = \sigma^2 P$, $\mathrm{Cov}(r) = \sigma^2(I-P)$

(i) $\hat{\boldsymbol\beta} \sim \mathcal{N}_p(\boldsymbol\beta, \sigma^2(X^\intercal X)^{-1})$

(ii) $\hat{\mathbf{Y}} \sim \mathcal{N}_n(X\boldsymbol\beta, \sigma^2 P)$, $\quad \mathbf{r} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2(I-P))$

(iii) $\hat\sigma^2 \sim \frac{\sigma^2}{n-p}\chi^2_{n-p}$.

$$\boxed{\hat{\boldsymbol\beta} = (X^\intercal X)^{-1}X^\intercal \mathbf{Y}.}$$

Scatterplot

```
col.partn <- 1*(partners==0) + 2*(partners==1) + 3*(partners==8)
plot(thorax, longevity, pch=pch.type, col=col.partn,
    ylim=range(longevity),xlim=range(thorax))
legend("topleft",c("1 pregnant","8 pregnant","1 virgin"
            "8 virgin","0 partners"),
        pch=c(1,2,1,2,3),col=c(2,2,3,3,1))
```

Does indicator variable affect thorax length?

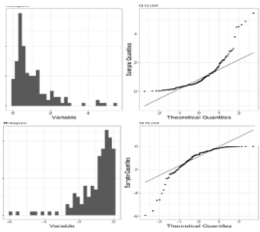
Reject H_0 if Pr(>F) <= 0.05

```
fitfull<-lm(thorax~dummy.1.p+dummy.1.v+dummy.8.p+dummy.8.v)
fitintercept<-lm(thorax~1)
anova(fitintercept,fitfull)
```

Order data

```
data[order(data[,"People.per.Dr"],decreasing=T)[1:3],]
datanew <-data[complete.cases(data),]
predict(fit, newdata=newcountry, interval="predict")
predict(fit, newdata=newcountry, interval="confidence")
```

Q-Q plot



Mallows Cp statistic

The MSE can be estimated by:

$$n^{-1}SSE(\mathcal{M}) - \hat\sigma^2 + 2\hat\sigma^2|\mathcal{M}|/n,$$

which is prop. to Cp(M):

$$C_p(\mathcal{M}) = \frac{SSE(\mathcal{M})}{\hat\sigma^2} - n + 2|\mathcal{M}|$$

Cp in R

(note: object is of type fit, and sigma can be found using sigma(fit))

```
Cp <- function(object,sigma){
 res <- residuals(object)
 n <- length(res)
 p <- n - object$df.residual
 SSE <- sum(res^2)
 SSE / sigma^2 - n + 2 * p

# choose sigma
sigma <- summary(fit5)$sigma
```

Forward selection (in case p, the nr. of predictors vars. Is too large for exhaustive search):

1) start with smallest model
2) add predictor which reduces the MSE the most
3) repeat 2 until all predictors or large nr. of predictors selected (now a seq. models is produced)
4) choose the model in the seq. which has smallest Cp statistic.

Backward selection is obvious.

Note that the linear regression model does not assume a normal distribution for the predictors, but a skewed distribution and outliers often result in regression solutions that are largely determined by very few points.

```
library(lattice)
splom(~mortality[,c("Mortality","Pop","HC","NOx","SO2")],pscales=0
```

Tukey Anscombe is residuals versus fitted values. Q-Q is empirical quantiles vs standard normal quantiles.

Forward and backward selection in R

```
mortal.bw <- step(mortal.full, dir="backward")
```

```
mortal.fw=step(mortal.empty,
 dir="forward", data=mortality,
 scope = list(upper=mortal.full,
          lower=mortal.empty))
```

For lm, Tukey Anscombe is given by

```
plot(fit,which =1)
```

Whereas Q-Q plot is given by

```
plot(fit,which =2)
```

Maybe add all-subsets regression in R (from HW2)?

Add anova table and f-test and interpretation

Linearity assumptions (for LS regression)

$\boxed{\text{The linear regression equation is correct.}}$ This means: $\mathbb{E}[\varepsilon_i] = 0$ for

$\boxed{\text{All } x_i\text{'s are exact.}}$ This means that we can observe them perfectly.

$\boxed{\text{The variance of the errors is constant ("homoscedasticity").}}$ This $\mathrm{Var}(\varepsilon_i) = \sigma^2$ for all i.

$\boxed{\text{The errors are uncorrelated.}}$ This means: $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$.

$\boxed{\text{The errors } \{\varepsilon_i;\ i = 1,\ldots,n\} \text{ are jointly normally distributed.}}$ This that also $\{Y_i; i = 1,\ldots,n\}$ are jointly normally distributed.

LOOCV in R

```
loocv <- function(reg.data, reg.fcn)
{
  loo.reg.value <- function(i, reg.data, reg.fcn)
    return(reg.fcn(reg.data$x[-i], reg.data$y[-i],
                  reg.data$x[i]))
  n <- nrow(reg.data)
  loo.values <- sapply(1:n, loo.reg.value,
                      reg.data, reg.fcn)
  mean((reg.data$y - loo.values)^2)
```

Non-parametric regression in R

Nadarava Watson

```
reg.fcn.nw <- function(reg.x, reg.y, x)
  ksmooth(reg.x, reg.y, x.point = x,
          kernel = "normal", bandwidth = h)$y
```

Local polynomial

```
reg.fcn.lp <- function(reg.x, reg.y, x) {
  lp.reg <- loess(reg.y ~ reg.x,
           enp.target = df.nw, surface = "direct")
  predict(lp.reg, x)
}
```

Smoothing splines

```
reg.fcn.ss <- function(reg.x, reg.y, x)
{
  ss.reg <- smooth.spline(reg.x, reg.y, spar = est.ss$spar)
  predict(ss.reg, x)$y
}
```

TODO: Hat matrix (serie 5 methods and non-parametric regression)

TODO: Equivalent ways of calculating LOOCV using hat matrix and manually computation

Bootstrap in R

```
library(boot)
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data=mtcars, statistic=bs,
                R=1000, formula=mpg~wt+disp)
plot(results, index=1) # intercept (histogram and quantile)
plot(results, index=2) # wt
plot(results, index=3) # disp
boot.ci(results, type="bca", index=1) # intercept
boot.ci(results, type="bca", index=2) # wt
boot.ci(results, type="bca", index=3) # disp
```

"basic" = "Reversed quantile"
"norm" = "Normal"
"perc" = "Quantile"

Expected size of out-of-bootstrap sample [roughly 1/3 of points will be out of sample]:

$$\mathbb{E}^*[|\mathcal{L}^*_{out}|] = \mathbb{E}^*[\sum_{i=1}^n 1_{[Z_i \in \mathcal{L}^*_{out}]}] = n\mathbb{P}^*[Z_i \in \mathcal{L}^*_{out}] \approx 0.368n.$$

Bayes Risk

$$\mathbb{P}[\mathcal{C}_{Bayes}(X_{new}) \neq Y_{new}].$$

Bayes Classifier

$$\mathcal{C}_{Bayes}(x) = \arg\max_{0\le j\le J-1} \pi_j(x).$$

LDA Classifier

$$\boxed{\hat{\mu_j} = \frac{1}{n_j}\sum_{i;Y_i=j} X_i,} \quad \boxed{\hat\Sigma = \frac{1}{n-J}\sum_{j=0}^{J-1}\sum_{i=1}^{n}(X_i - \hat{\mu_j})(X_i - \hat{\mu_j})^\intercal 1_{[Y_i=j]}}$$

Prediction of LDA found by taking argmax:

$$\begin{aligned}\hat\delta_j(x) &= x^\intercal\hat\Sigma^{-1}\hat{\mu_j} - \hat{\mu_j}^\intercal\hat\Sigma^{-1}\hat{\mu_j}/2 + \log(\hat p_j) = \\ &= (x - \hat{\mu_j}/2)^\intercal\hat\Sigma^{-1}\hat{\mu_j} + \log(\hat p_j).\end{aligned}$$

Decision boundary between class 0 and 1:

$$B = \{x \mid \hat\delta_0(x) = \hat\delta_1(x)\} = \{x | (x - z)^\intercal w = 0\}$$

if the covariance matrix ∑ is diagonal and const. then w is parallel to the line connecting μ_0 and μ_1

QDA Classifier (more easily overfits if p is big)

$$\boxed{\hat\Sigma_j = \frac{1}{n_j - 1}\sum_{i=1}^{n}(X_i - \hat{\mu_j})(X_i - \hat{\mu_j})^\intercal 1_{[Y_i=j]}}$$

$$\boxed{\hat\delta_j(x) = -\log(\det(\hat\Sigma_j))/2 - (x - \hat{\mu_j})^\intercal\hat\Sigma_j^{-1}(x - \hat{\mu_j})/2 + \log(\hat p_j).}$$

Bootstrapping LDA/QDA

```
index <- matrix(sample.int(n, n*B, replace = TRUE), nrow = n, ncol = B)
fit_lda <- vector("list", B)
for(i in 1:B) {
  ind <- index[, i]
  fit_lda[[i]] <- lda(x = Iris[ind , c("Petal.Length", "Petal.Width")],
           grouping = Iris[ind , "Species"])
}
```

Decision boundary plot in R

```
xp <- seq(min(x[, 1]), max(x[, 1]), length = len)
yp <- seq(min(x[, 2]), max(x[, 2]), length = len)
grid <- expand.grid(xp, yp)
Z <- predict(object, grid, ...)
zp <- as.numeric(Z$class)
zp <- Z$post[, 3] - pmax(Z$post[, 2], Z$post[, 1])
contour(xp, yp, matrix(zp, len),
        add = TRUE, levels = 0, drawlabels = FALSE, col = colcont)
```

(Linear) logistic regression

$$\boxed{\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \sum_{j=1}^{p}\beta_j x_j}$$

LHS is logit transform of π

$$\boxed{\log\left(\frac{\pi_j(x)}{1-\pi_j(x)}\right) = g_j(x) = \sum_{r=1}^{p}\beta_r^{(j)}x_r}$$

For J>2

Likelihood (for Bernoulli x then for Binomial)

$$L(\beta; (x_1, Y_1), \ldots, (x_n, Y_n)) = \prod_{i=1}^n \pi(x_i)^{Y_i}(1 - \pi(x_i))^{1-Y_i}.$$

$$L(\beta; (x_1, m_1, N_1), \ldots, (x_n, m_n, N_n)) = \prod_{i=1}^n \binom{m_i}{N_i}\pi(x_i)^{N_i}(1 - \pi(x_i))^{m_i - N_i}$$

```
fit <- glm(cbind(N, m - N) ~ age,
     family = binomial, data = heart)
```

```
predict(fit, new = data.frame(age =
        new.age), type = "response")
```

```
fit <- glm(Survival ~ ., data =
     d.baby, family = "binomial")
```

will give the probability of heart disease (N/m)

Multinomial logistic regression

```
m <- multinom(Species ~ . , data = Iris)
ZP <- predict(object, newdata = grid, type = "probs")
```

where the probabilities of each class are stored in the columns (columns same order as levels (…)) but we can also achieve logistic regression by using dummy encoding and glm:

```
levels(Iris1$Species) <- c("setosa",
                          "not", "not")
Iris1$Species <- relevel(Iris1$Species,
                         ref = "not")
fit.1 <- glm(Species ~ ., data = Iris1,
           family = "binomial")
```

ROC and Misclassification graph in R

```
require(ROCR)
fit <- glm(Survival ~ ., data = d.baby,
          family = "binomial")
pred <- prediction(fit$fitted.values,
                  d.baby$Survival)
perf <- performance( pred, "tpr", "fpr" )
plot(perf, main = "title")
```

which plots the TPR vs FPR. To plot the misclassification rate as a function of threshold prob. (i.e., after this it is considered positive)

```
perf.cost <- performance(pred, "cost")
plot(perf.cost, main = title)
```

K-Fold CV in R

```
folds <- sample(cut(seq(1, n), breaks = K,
        labels = FALSE), replace = FALSE)
for (i in 1:K){
  test.ind <- which(folds == i)
  df.train<- d.baby[-test.ind,]
  df.test <- d.baby[ test.ind,]
  ...}
```

To plot the average ROC (across all folds) initialize a vector of lists and assign them as follows: `all.y.true[[i]] <- y.true` `all.y.pred[[i]] <- y.pred`

And then pred.cv <- prediction (all.y.pred, all.y.true) and proceed as normal (but add avg = "threshold" for ROC and avg = "vertical" for misclassification) use add = T to add them to same plot. One can specify the cost for FN and FP using cost.fp = 0.8, cost.fn = 1.2 as args to performance().

Poly. regression using a polynomial of degree d (no interaction term):

$$\begin{aligned}y_i = \beta_0 + \beta_{1,1}x_{i1} + \beta_{1,2}x_{i1}^2 + \ldots\beta_{1,d}x_{i1}^d + \ldots \\ + \beta_{p,1}x_{ip} + \beta_{p,2}x_{ip}^2 + \ldots\beta_{p,d}x_{ip}^d + \epsilon_i.\end{aligned}$$

```
require(sfsmisc)
form1 <- as.formula("logupo3~.")
form3 <- wrapFormula(form1, data =
  d.ozone.e, wrapString="poly(*,degree=d)")
fit3 <- lm(form3, data = d.ozone.e)
```

Generalized additive model in R

```
require(mgcv)
form1 <- as.formula("logupo3~.")
gamForm <- wrapFormula(form1, data = d.ozone.e)
g1 <- gam(gamForm, data = d.ozone.e)
```

Alternatively,

```
fitA <- gam(O3 ~ s(vdht)+ s(wind)+
 s(humidity)+ s(temp)+ s(ibht)+
 s(dgpg)+ s(ibtp)+ s(vsty)+ s(day),
 data = d.ozone)
```

To plot the splines do:

```
par(mfrow=c(3,3))
plot(g1, shade = T)
```

To plot the same thing but for lm use:

```
par(mfrow=c(3,3))
termplot(fit5, partial.resid=TRUE,
 rug=FALSE, se=TRUE,
 col.res='#C0C0C050', pch=19)
```

R^2 describes the proportion of the total variation of the response Y around its mean Y bar which is explained by the regression Y hat.

$$\boxed{R^2 = \frac{\|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2}{\|\mathbf{Y} - \overline{\mathbf{Y}}\|^2}.}$$

In regression we have: (which allows for writing R^2 differently)

$$\boxed{\|\mathbf{Y} - \overline{\mathbf{Y}}\|^2 = \|\hat{\mathbf{Y}} - \overline{\mathbf{Y}}\|^2 + \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2}$$

The following removes col. upo3:

```
d.ozone <- subset(ozone, select=-upo3)
d.ozone <- subset(transform(d.ozone, "logO3" = log(O3)), select = -O3)
```

adds log(03) as a col. and removes 03

(In-sample) R^2 in R

```
r2 <- function(fitfn, formula = logO3 ~ . ,
              data = d.ozone.es, ...)
{
  modFrame <- model.frame(formula, data = data)
  Y <- model.response(modFrame)
  #calculate ssr and sst
  fit <- fitfn(formula=formula, data = data, ...)
  ssr <- sum((Y - predict(fit, modFrame))^2)
  sst <- sum((Y-mean(Y))^2)
  1-ssr/sst
}
```

(Cross-validated LOOCV) R^2 in R

```
cv_r2 <- function(fitfn, formula = logO3 ~ . , data =
                  d.ozone.es, ..., trace = TRUE)
{
  modFrame <- model.frame(formula, data = data)
  n <- nrow(data)
  ssr <- 0
  if(trace) cat(" j = ")
  for(j in 1:n) {
    ## Fit without 'j' :
    fit <- fitfn(formula=formula, data = data[-j ,],
           ..., trace = FALSE)
    ## Evaluate at 'j' :
    ssr <- ssr + (model.response(modFrame)[j] -
              predict(fit, modFrame[j,]))^2
  }
  #calculate sst
  fit <- fitfn(formula=formula, data = data, ...)
  sst <- sum((model.response(modFrame)-
            mean(model.response(modFrame)))^2)
  if(trace) cat("\n")
  1-ssr/sst
}
```

MARS in R

```
require("earth")
Mfit <- earth(Volume ~ ., data = trees)
summary(Mfit)
```

```
predict(Mfit,
    data.frame(Girth= 5:15, Height= seq(60,80, length=11)))
```

We can then find their R^2 as follows:

```
r2(earth, formula = logO3 ~ . ,
   data = d.ozone.es, degree = 3)
cv_r2(earth, formula = logO3 ~ . ,
   data = d.ozone.es, degree = 3)
```

MARS fits a function of the form

$$g(\mathbf{x}) = \mu + \sum_{m=1}^{M}\beta_m h_m(\mathbf{x}) = \sum_{m=0}^{M}\beta_m h_m(\mathbf{x}),$$

1. Start with the function $h_0(x) = 1$. Initialize the model set $\mathcal{M} = \{h_0(\cdot) = 1\}$. Fit the function h_0 by least squares regression, yielding the estimate $\hat\mu = n^{-1}\sum_{i=1}^n Y_i$.
2. For $r = 1, 2, \ldots$ do the following:
Search for the best pair of functions $(h_{2r-1}(\cdot), h_{2r}(\cdot))$ which are of the form
$$\begin{aligned}h_{2r-1}(\cdot) &= h_\ell(\cdot) \times (x_j - d)_+, \\ h_{2r}(\cdot) &= h_\ell(\cdot) \times (d - x_j)_+,\end{aligned} \quad (7.1)$$
for some h_ℓ in the model set $\mathcal{M}$ which does not contain x_j and some basis functions in $\mathcal{B}$. The best pair of functions is defined to be the one which reduces residual sum of squares most. The model fit is then
$$\hat g(\mathbf{x}) = \hat\mu + \sum_{m=1}^{2r}\hat\beta_m h_m(\mathbf{x}),$$
where the coefficients $\hat\beta_m$ are estimated by least squares.
Enlarge the model set in every iteration (with index r) by
$$\mathcal{M} = \mathcal{M}_{old} \cup \{h_{2r-1}(\cdot), h_{2r}(\cdot)\},$$
with the functions h_{2r-1}, h_{2r} from (7.1).
3. Iterate step 2 until a large enough number of basis functions $h_m(\cdot)$ has been fitted.
4. Do backward deletion ("pruning"), allowing to remove single functions from a pair $h_{2r-1}(\cdot), h_{2r}(\cdot)$; i.e., of all single basis functions, delete the one which increases the residual sum of squares the least.
5. Stop the backward deletion by optimizing a GCV score.

summary (mars fit) gives the importance of each variable. If we fitted a model of degree 2, then we can get the graphs of the response variables as a function of each predictor variable as follows:

```
plotmo(fit, degree2=FALSE,
     caption="main effects")
```

We can also get the 3D plots (effect of the interaction of 2 terms on the response) using

```
plotmo(fit, degree1=FALSE,
   caption="interactions")
```

Neural Networks (1 Layer with q units)

$$\boxed{g_k(\mathbf{x}) = f_0\left(\alpha_k + \sum_{h=1}^{q} w_{hk}\phi(\tilde\alpha_h + \sum_{j=1}^{p}\tilde w_{jh}x_j)\right)}$$

where $\phi(t) = \frac{\exp(t)}{1+\exp(t)}$

and f_0 is typically identity for regression and sigmoid for classification. We can add a linear regression component (skip conxn):

$$\boxed{g(\mathbf{x}) = f_0\left(\alpha + \sum_{j=1}^{p} w_{j,lin}x_j + \sum_{k=1}^{q} w_k\phi(\alpha_k + \sum_{j=1}^{p} w_{jk}x_j)\right)}$$

Make sure to scale the data (except for response) to avoid getting stuck in the flat regions of the sigmoid function. Weight decay is good as it reduces the dependency on the starting values and reduces the importance of choosing the # of hidden units

```
Nfit <- nnet(log.O3 ~ . , data = sc.ozone,
           size = 3,       # chooses 3 h
           decay = 4e-4,    # "weight dec
           linout = TRUE,   # linear regr
           skip = FALSE,    # a N.N. with
           maxit = 500)
```

To get the weights use summary(Nfit). R prefixes input variables with *i* and hidden unit with *h*. To get the error use sum(residuals(Nfit)^2)

`optim(*, method="BFGS")`

is equivalent to using a neural network (note results may differ due to different seed).

Optimization in R

Assume we have a fct. like this for the neg. log likelihood which we want to minimize (case of binomial distribution of response and logist. regr.),

```
neg.ll <- function(beta, data){
 - sum(log(choose(data$m, data$N)) +
       data$N * g(beta,data$age) -
       data$m * log( 1 + exp(g(beta,data$age))))
}
```

Then the following will return the vector beta (same as that of logistic regression),

```
optim(c(0, 0), neg.ll, data = heart)$par
```

Trees (CART)

1. Start with $M = 1$ subset, $\mathcal{P} = \{\mathcal{R}\} = \{\mathbb{R}^p\}$.
2. Refine $\mathcal{R}$ into $\mathcal{R}_{left} \cup \mathcal{R}_{right}$ where:

$$\begin{aligned}\mathcal{R}_{left} &= \mathbb{R} \times \mathbb{R} \times \ldots \times (-\infty, d] \times \mathbb{R} \ldots \times \mathbb{R},\\ \mathcal{R}_{right} &= \mathbb{R} \times \mathbb{R} \times \ldots \times (d, \infty) \times \mathbb{R} \ldots \times \mathbb{R},\end{aligned}$$

   where one of the axes is split at the split point d, where d is from the finite set of mid-points between observed values. The search for the axes to split and the split point d are determined such that the negative log-likelihood is maximally reduced with the refinement (search over $j \in \{1, \ldots, p\}$ and $d \in$ {mid-points of observed values}). Build the new partition $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2\}$ with $\mathcal{R}_1 = \mathcal{R}_{left}$, $\mathcal{R}_2 = \mathcal{R}_{right}$.

3. Refine the current partition $\mathcal{P}$ as in step 2 by refining **one** of the partition cells from the current partition $\mathcal{P}$. That is, we search for the best partition cell to refine which includes a search as in step 2 for the best axes to split and the best split point. Then, we up-date the partition:

   $\mathcal{P} = \mathcal{P}_{old} \setminus$ partition cell selected to be refined $\cup$ {refinement cells $\mathcal{R}_{left}, \mathcal{R}_{right}$}.

4. Iterate step 3 for a large number, $M = M_{max}$, of partition cells.

5. Backward deletion: prune the tree (see below) until a reasonable model size, typically determined via cross-validation, is achieved.

Commented [g1]: For anova see p290 of ISLR