# L7

2023-04-25

## Quizzes vs. HW

- **Decision: no more quizzes, only HW**
- **Interactive Grading**
  1. Will take place 1+ week after due date
  2. Yuyi will run script and check code prior
  3. Yuyi will send out a sign-up for grading in groups
  4. It will count more than before
  5. Prof will be asking the questions instead of Yuyi
  6. Bring in whatever notes or code copies you want
  7. Different questions for each group member
  8. Might be asked questions about course in general
     - i.e. “tell me how random forest works”
     - i.e. “tell me the tradeoff between having many vs. few levels”
- **Regrading**
  1. We are *very* welcome to ask for a regrade if we feel our original grade is inaccurate/non-reflective
  2. If someone asks for a re-grade, prof will just ask one or two additional questions (should be a quick process, not re-doing the entire thing)
  3. Grade won’t get worse from requesting a regrade
  4. If we request a regrade, it will be individual; not the entire group

## **Logistic Model** (https://github.com/matloff/qeML/blob/master/inst/mdFiles/ML_Overview.md#l model)

### Logistic Model:

> This is a generalization of the linear model, developed by statisticians and economists. This model is only for classification settings. As noted, since Y is now 1 or 0, its mean becomes the probability of 1. Since m(t) is now a probability, we need it to have values in the interval [0,1]. This is achieved by feeding a linear model into the *logistic function*, which does take values in (0,1).

### **Logistic Function:**

**Standard Logistic Function:** $L(u) = (1 + e^{-u})^{-1}$ or $L(u) = 1/(1 + e^{-u})$

> Example: to predict whether a player is a catcher (Y = 1 if yes, Y = 0 if no), we fit the model:
>
> - Our feature weights are represented by u: $u = \beta_0 + \beta_1\ height + \beta_2\ weight + \beta_3\ age$
> - Represent the features as a t-vector (sometimes written as an x-vector)
>   - $t_1$ = height
>   - $t_2$ = weight
>   - $t_3$ = age
> - So in turn, the function $L(u) = 1/(1 + e^{-u})$ becomes $L(u) = 1/[1 + e^{-(\beta_0+\beta_1\ height+\beta_2\ weight+\beta_3\ age)}]$
> - Here, $L(u)$ represents $P(catcher \mid height, weight, age)$ for a given height, weight, and age.
> - The final fitted function will have weights such that expected value of y (our prediction given t-values) $E(y|t) = \beta_0 + \beta_1 t_1 + \beta_2 t_2 + \beta_3 t_3$
> - Function *(this part is probably copied incorrectly!)* becomes: $l(\beta_0,\ \beta_1 t_1,\ \ldots) = P(y = 1|X = t)\ /\ [1 + (1/l) * (\beta_0 + \beta_1 t_1\ + \beta_2 t_2\ +\ \ldots)]$
>
> The $\beta_i$ are estimated from the sample data, using a technique called *iteratively reweighted least squares*.

The $\beta_i$ are weights for each feature

- $\beta_0$ is the intercept
- $\beta_1$ is the weight for feature $t_1$ (height)
- $\beta_2$ is the weight for feature $t_2$ (weight)

- $\beta_3$ is the weight for feature t3 (age)

```
library(fdm2id, quietly=T)
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
##
## Attaching package: 'fdm2id'
```

```
## The following objects are masked from 'package:FactoMineR':
##
##     CA, MCA, PCA
```

```
library(qeML, quietly=T)
```

```
## Registered S3 method overwritten by 'regtools':
##   method      from
##   predict.knn fdm2id
```

```
##
##
##
##
##
## *********************
##
##
##
## Latest version of regtools at GitHub.com/matloff
##
##
## Type ?regtools to see function list by category
```

```
##
## Attaching package: 'regtools'
```

```
## The following object is masked from 'package:fdm2id':
##
##     confusion
```

```
## The following object is masked from 'package:arules':
##
##     discretize
```

```
## Latest version of partools at GitHub.com/matloff
```

```
##
##
##
##
##
## *********************
##
##
##
##   Navigating qeML:
##
##       Type vignette("Quick_Start") for a quick overview!
##
##       Type vignette("FtnList") for a categorized function list
##
##       Type vignette("ML_Overview") for an introduction to machine learning
```

```
data(spine)
str(spine)
```

```
## 'data.frame':    310 obs. of  8 variables:
##  $ V1      : num  39.1 53.4 43.8 31.2 48.9 ...
##  $ V2      : num  10.1 15.9 13.5 17.7 20 ...
##  $ V3      : num  25 37.2 42.7 15.5 40.3 ...
##  $ V4      : num  29 37.6 30.3 13.5 28.9 ...
##  $ V5      : num  114 121 125 120 119 ...
##  $ V6      : num  4.56 5.99 13.29 0.5 8.03 ...
##  $ Classif2: Factor w/ 2 levels "AB","NO": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Classif3: Factor w/ 3 levels "DH","NO","SL": 1 1 1 1 1 1 1 1 1 1 ...
```

## Fitting a Logistic Model

Predict "Classif3" variable using V1:V6 variables

```
spine <- spine[,-7]  # skip 2-class example "Classif2"
u <- qeLogit(spine,'Classif3')
```

```
## holdout set has  31 rows
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
u$testAcc # error when trained including V1:V6
```

```
## [1] 0.1612903
```

```
u$baseAcc # error when trained excluding V1:V6
```

```
## [1] 0.5304659
```

**Takeaway:** when V1:V6 are *included*, error drastically reduces, indicating their high predictive utility

- testAcc: error when model is trained with V1:V6 (~90% accurate)
- baseAcc: error when model is trained without V1:V6 (~50% guess rate)

```
table(spine$Classif3)
```

```
##
##  DH  NO  SL
##  60 100 150
```

## Prediction

> Let's try a prediction. Consider someone like the first patient in the dataset but with V6 = 6.22. What would our prediction be?

```
newx <- spine[1,-7]  # omit "Y"
newx$V6 <- 6.22      # replace V6 value of first row patient
predict(u,newx)      # generate new prediction for first row patient
```

```
## $predClasses
## [1] "DH"
##
## $probs
##             DH        NO        SL
## [1,] 0.7762033 0.2021179 0.0216788
```

Outcome:

- The model generates probabilities that patient is of "DH", "NO", and "SL" classes, and chooses the class with the highest probability
- The model predicts that the patient is of "DH" class, with probability ~0.75
- In R, the `predClasses` and `$probs` are S3 Classes

## S3 Lists

**S3:**

- S was a commercial language
- S3 was released as an object-oriented, open-source version of S

**S3 Lists in R:**

- An S3 object is an R list with a defined class
    - Example: `class(listObj) <- "..."`
- Dollar signs in output means the output is an S3 class with two components
    - `$predClasses` : the actual prediction
    - `$probs` : the probability of the prediction

## Functions

- In R, operators like `==` , `+` , and `<-` are functions
- `z <- qeLogit(...)` is a function
    - `class(z)` returns `"qeLogit"`
- When running `predict(z,~)` the actual function executed is `predict.qeLogit(~)` since the call to `predict` is dispatched to `qeLogit`