Contents

Evaluating predictive marketing models 2

Generalized linear models **23**

How do I Build a Scoring Model?

A Scoring model is a *data-mining model* used to *predict behavior* based on *other information* we have on a (prospective) customer.

1. Identify a modeling database that has a "proxy behavior"
   - *Proxy behavior* — behavior that has been observed in the past that is similar to future behavior you would like to predict, e.g., response to similar offer sent yesterday

     Warning: your model may not work because this is only a proxy behavior. Seasonality, the state of the economy, what competition is doing, etc. usually all affect response.
   - *Target period* — time period when proxy offer was active.
   - *Base period* — a period of time prior to the target period. Information from the base period will be used to predict the proxy behavior.
2. Develop and validate scoring model
3. Score the entire database
4. Decide who gets contacted — where should you stop?

How do I Evaluate a Scoring Model?

- We've developed 2+ scoring models. Which is better?
- Performance usually assessed with a *gains table*
  1. Find quantiles of predicted values $\hat{y}$
  2. Compute number of responders and revenue by quantile, also averages
  3. Compute cumulative counts and revenues by quantile, also averages and lifts

A	B	C	D	E	F	G	H	I	J	K	L	M
Quantile of $\hat{y}$		**Amount by Quantile**				**Cumulative**					**Lift**	
%	n	Num Resp	Rev Amt	Resp Rate	Avg Amt	Num Cont	Num Resp	Rev Amt	Resp Rate	Avg Amt	Resp Rate	Rev Amt
1	10569	1681	159501	0.159	15.1	10569	1681	159501	0.159	15.1	2.98	3.18
2	10569	477	40241	0.0451	3.81	21138	2158	199742	0.102	9.45	1.91	1.99
3	10568	307	23716	0.0290	2.24	31706	2465	223458	0.0777	7.05	1.45	1.49
4	10569	201	15484	0.0190	1.46	42275	2666	238942	0.0631	5.65	1.18	1.19
5	10569	159	11608	0.0150	1.10	52844	2825	250549	0.0535	4.74	1	1

- **Columns A and B**: Quantiles of $\hat{y}$ and counts.
- **Columns C and D**: number of responders and total revenue by quantile
- **Columns E and F**: response rate and average revenue in quantile, e.g., 1681/10569 = 15.9% and 96728/10569 = \$15.1 per contact
- **Column G**: *Depth* of contacts or cumulative counts, e.g., 10,569+10,569=21,138.
- **Columns H and I**: cumulative responders and revenue by quantile,
  - Row 2: 1681+444=2158 responders and 159,501+40,241 = 199,742 revenue at 40%.
  - Last row: 2,825 total responders and 250,549 total revenue
- **Columns J and K**: cumulative response rate and average revenue in quantile,
  - Row 2: $2,158/21,138 = 9.45\%$ and $199,742/21,138 = \$9.45$ per contact at 40%
  - Last row: **contacting at random** gives 5.35% respond rate, \$4.74 per contact
  - ML people call column J **precision@k**, e.g., precision@20%
- **Columns L and M**: lift of model over random guessing, e.g., 15.1%/5.35% = 2.98 indicates the response rate from using model to pick best 20% of the names is improved by 57% over picking names at random. Revenue more than tripled (3.93)!

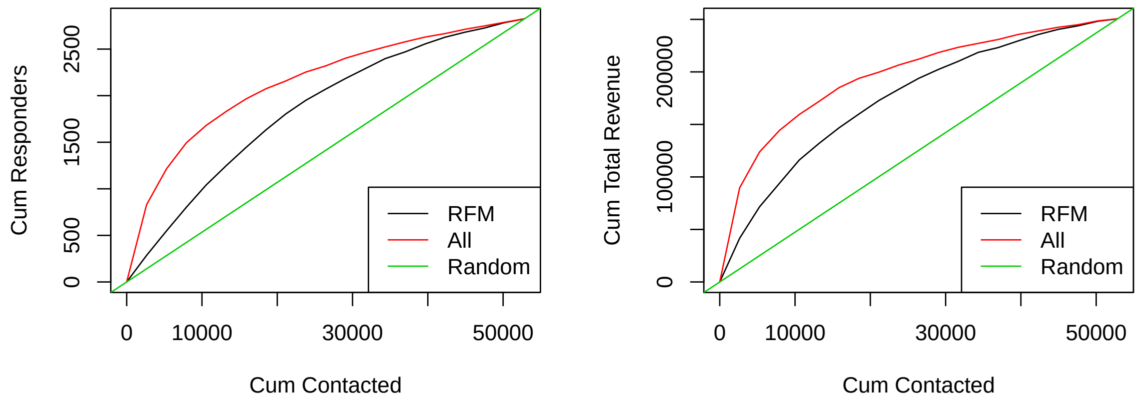
Two models for DMEF3 data

```
# my code to make a gains table
gains = function(yhat, respond, amt, ngrp=5){
  ans = data.frame(amt=amt, respond=respond, qtile=
      cut(yhat, breaks=quantile(yhat, probs=seq(0,1, 1/ngrp)),
      labels=paste("Q",ngrp:1, sep=""), include.lowest = T)
  ) %>%
    group_by(qtile) %>%
    summarise(n=n(), Nrespond=sum(respond), amt=sum(amt),
      RespRate=Nrespond/n, AvgAmt=amt/n) %>%
    arrange(desc(qtile)) %>%
    mutate(CumN=cumsum(n), CumResp=cumsum(Nrespond), CumAmt=cumsum(amt),
      CumRespRate=CumResp/CumN, CumAvgAmt=CumAmt/CumN)
  ans %>% mutate(liftResp=CumRespRate/CumRespRate[nrow(ans)],
    liftAmt=CumAvgAmt/CumAvgAmt[nrow(ans)])
}

dat = read.csv("/Users/ecm/teach/iems304/dmef3/train.csv")
dat$custno = NULL
names(dat); summary(dat)
for(i in c(10:16,22, 23)) dat[[i]][dat[[i]]<0] = 0   # set neg to 0
dat$buy = as.numeric(dat$targamnt>0) # define buy variable
for(i in 2:22) dat[[i]] = log(dat[[i]]+1) # log all but recency
names(dat); summary(dat)

# fit basic RFM model
fit.rfm = glm(buy ~ recmon + totord + totsale, binomial, dat)
summary(fit.rfm)
gains(fit.rfm$fitted.values, dat$buy, dat$targamnt)

# log everything and dump all vars in
fit.all = glm(buy~., binomial, dat[,-23]) # regress buy on all buy targamnt
plot.roc(dat$buy, fit.rfm$fitted.values, col=1, print.auc=T)
plot.roc(dat$buy, fit.all$fitted.values, col=2, add=T, print.auc=T, print.auc.y=1)
gains(fit.all$fitted.values, dat$buy, dat$targamnt)

> gains(fit.rfm$fitted.values, dat$buy, dat$targamnt)
          Num           Resp   Avg   Cum   Cum    Cum   Resp    Avg  lift  lift
        n Resp    amt   Rate   Amt     N  Resp    Amt   Rate    Amt  Resp   Amt
Q1 10569 1042 116159  0.0986 11.0  10569  1042 116159 0.0986  11.0   1.84  2.32
Q2 10569  762  56626  0.0721  5.36 21138  1804 172785 0.0853   8.17  1.60  1.72
Q3 10568  487  37351  0.0461  3.53 31706  2291 210136 0.0723   6.63  1.35  1.40
Q4 10569  336  25396  0.0318  2.40 42275  2627 235532 0.0621   5.57  1.16  1.18
Q5 10569  198  15018  0.0187  1.42 52844  2825 250549 0.0535   4.74  1     1
```

Gains charts

Plot performance measures against depth

```
> gains(fit.all$fitted.values, dat$buy, dat$targamnt)
           Num              Resp    Avg    Cum    Cum      Cum    Resp     Avg  lift  lift
        n Resp     amt      Rate    Amt      N   Resp      Amt    Rate     Amt  Resp   Amt
Q1  10569 1681  159501  0.159   15.1   10569   1681  159501  0.159    15.1   2.98  3.18
Q2  10569  477   40241  0.0451   3.81  21138   2158  199742  0.102     9.45  1.91  1.99
Q3  10568  307   23716  0.0290   2.24  31706   2465  223458  0.0777    7.05  1.45  1.49
Q4  10569  201   15484  0.0190   1.46  42275   2666  238942  0.0631    5.65  1.18  1.19
Q5  10569  159   11608  0.0150   1.10  52844   2825  250549  0.0535    4.74  1     1

m1=gains(fit.rfm$fitted.values, dat$buy, dat$targamnt, 20) # increase quantiles
m2=gains(fit.all$fitted.values, dat$buy, dat$targamnt, 20)
# response rate plot
plot(c(0,m1$CumN), c(0,m1$CumResp), type="l", xlab="Cum Contacted",
  ylab="Cum Responders")
lines(c(0,m2$CumN), c(0,m2$CumResp), col=2)
abline(0, m2$CumRespRate[20], col=3)
legend("bottomright", c("RFM", "All", "Random"), col=1:3, lty=1)
# revenue plot
plot(c(0,m1$CumN), c(0,m1$CumAmt), type="l", xlab="Cum Contacted",
  ylab="Cum Total Revenue")
lines(c(0,m2$CumN), c(0,m2$CumAmt), col=2)
abline(0, m2$CumAvgAmt[20], col=3)
legend("bottomright", c("RFM", "All", "Random"), col=1:3, lty=1)
```



Gains table with a test set

```
set.seed(12345)
dat$train=runif(nrow(dat))>.5   # split data into training and test sets

> table(dat$train)
FALSE  TRUE
26505 26339

> # add subset=train to glm call to use training data only
> fit.rfm = glm(buy ~ recmon + totord + totsale, binomial, dat, subset=dat$train)
> summary(fit.rfm)  # note df matches training set
Call: glm(formula = buy ~ recmon + totord + totsale, family = binomial,
    data = dat, subset = train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.778406   0.182950 -20.653  < 2e-16 ***
recmon      -0.032593   0.002238 -14.560  < 2e-16 ***
totord      -0.182050   0.057312  -3.176  0.00149 **
totsale      0.363157   0.046952   7.735 1.04e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

    Null deviance: 11125  on 26338  degrees of freedom
Residual deviance: 10703  on 26335  degrees of freedom

> # Now we must use predict to apply model to test set
> phat = predict(fit.rfm, dat[!dat$train,], type="resp")
> # don’t forget type="resp" to get probs, although order does not change
> summary(phat)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.005395 0.031313 0.050932 0.054364 0.073341 0.204116

> gains(phat, dat$buy[!dat$train], dat$targamnt[!dat$train])
         Num           Resp   Avg   Cum   Cum    Cum    Resp    Avg  lift  lift
       n Resp    amt    Rate   Amt     N  Resp    Amt   Rate    Amt  Resp   Amt
Q1  5301  511  59365  0.0964 11.2   5301   511  59365 0.0964  11.2   1.83  2.39
Q2  5301  378  26613  0.0713  5.02 10602   889  85978 0.0839   8.11  1.60  1.73
Q3  5301  250  18321  0.0472  3.46 15903  1139 104299 0.0716   6.56  1.36  1.40
Q4  5301  167  13303  0.0315  2.51 21204  1306 117602 0.0616   5.55  1.17  1.18
Q5  5301   87   6612  0.0164  1.25 26505  1393 124215 0.0526   4.69  1     1
```

Two-Step Models

- Problem: build a regression model to predict a dependent variable that is a dollar amount
- Problem: sometimes models that predict response don't predict profitability and visa versa
- Problems with linear regression:
    1. A very high percentage of the y values are 0, e.g., response rates of 2% are not uncommon $\Longrightarrow$ 98% zeros.
    2. A person cannot spend negative dollars, but $\hat{y}$ can assume negative values (similar problem to logistic regression — $\beta_0 + \beta_1 x$ is unbounded)
- Possible improvement: two-step models
    1. *Response Model*. Predict response with logistic regression
    2. *Conditional Demand Model*. Predict dollar amount with a linear regression model for *only those who responded* (use select cases where response=yes)
    3. Compute expected dollar amount as follows:

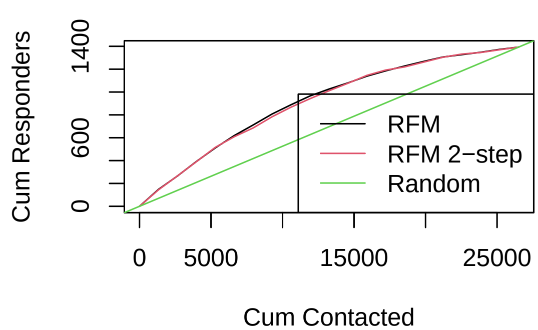
    $$\mathbb{E}(Y_i) = 0\mathsf{P}(\text{No}) + \mathbb{E}(Y|\text{Yes})\mathsf{P}(\text{Yes})$$

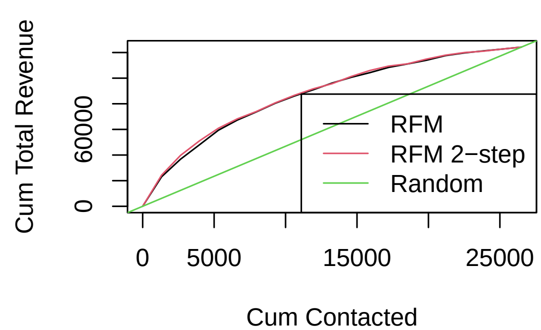
- See article by Elkan on "Heckman correction:" include $\hat{p}$ values from response model as predictor in conditional demand model

Two-step model with test set

Continuing example from two slides ago

```
> # estimate spend among buyers in training set only
> fit.spend = lm(log(targamnt) ~ recmon + totord + totsale, dat, subset=train&buy==1)
> summary(fit.spend)
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.936e+00  1.133e-01  25.909   <2e-16 ***
recmon       3.495e-05  1.463e-03   0.024    0.981
totord      -4.133e-01  3.600e-02 -11.479   <2e-16 ***
totsale      3.801e-01  2.909e-02  13.065   <2e-16 ***

Residual standard error: 0.6526 on 1428 degrees of freedom
Multiple R-squared:  0.1094,Adjusted R-squared:  0.1076
F-statistic: 58.49 on 3 and 1428 DF,  p-value: < 2.2e-16

> # Compute predicted values
> # from two slides ago: make sure type="resp" !!!
> yhat = phat * exp(predict(fit.spend, dat[!dat$train,]))
> summary(yhat)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.06463  1.82155  3.18276  3.80336  5.10052 42.95916
> # make plots as before
```





Generalized logistic regression

- We have only discussed the situation where there are **two** possible outcomes (dependent variable)
- Example uses
  - Predict my next category of purchase at a retailer based on browsing history (e.g., at REI, will I buy clothing, camping supplies, shoes, jacket, etc.?)
  - Predict brand choice (Pepsi, Coke, RC Cola, storebrand) based on marketing mix variables (price, in-store ads, mass ads, etc.)
  - Handwritten digits (e.g., MNIST): classify bitmaps images as 0, 1, . . . , 9
  - Medical diagnosis: e.g., flu, cold, healthy
  - Email foldering: work, friends, family, hobby
  - University admissions: admit, wait list, deny
  - Sentiment analysis: positive, negative or neutral
  - Facial or speech recognition

Binomial/multinomial review

- **Binomial** review:
  - We observe n iid trials with a *dichotomous* outcome, with one outcome labeled success and the other failure.
  - For each trail, the probability of success is π (and the probability of failure is $1 - \pi$)
  - Binomial RV X_b counts the number of successes:

$$\mathsf{P}(X_b = x) = \frac{n!}{x!(n-x)!}\pi^x(1-\pi)^{n-x}$$

$$\mathbb{E}(X_b) = n\pi \quad \text{and} \quad \mathbb{V}(X_b) = n\pi(1-\pi)$$

  - The expected number of failures is $\mathbb{E}(n - X_b) = n(1-\pi)$
- What's a **multinomial**?
  - Instead of having two outcomes (success and failure), suppose there are K outcomes where the probability of class k is π_k $(k = 1, \ldots, K)$, $\pi_1 + \cdots + \pi_K = 1$.
  - Suppose there are n iid trials. A **multinomial** random vector is $\mathbf{X} = (X_1, \ldots, X_K)$, where X_k is the number of outcomes from class k out of n trials $(X_1 + \cdots + X_K = n)$, has PMF

$$\mathsf{P}(\mathbf{X} = \mathbf{x}) = \frac{n!}{x_1! \cdots x_K!}\pi_1^{x_1} \cdots \pi_K^{x_K}$$

$$\mathbb{E}(X_k) = n\pi_k, \quad \mathbb{V}(X_k) = n\pi_k(1-\pi_k), \quad \mathsf{C}(X_i, X_j) = -\pi_i\pi_j$$

- $K = 2$ gives binomial ($\pi_1 = \pi$, $\pi_2 = 1 - \pi$, $x_1 = x$, $x_2 = n - x$)

Generalized logistic regression model

- Assume a sample of n observations and p predictors
- Let $Y \in \{1, 2, \ldots, K\}$ be a multinomial r.v. for the outcome with K values. The probability that observation i comes from class k is $\pi_{ik} = P(Y_i = k)$, where $\pi_{i1} + \cdots + \pi_{iK} = 1$.
- Predictor variables: $\mathbf{x}_i = (1, x_{i1}, \ldots, x_{ip})^\mathsf{T}$
- Parameters: $\boldsymbol{\beta}_k = (\beta_{k0}, \beta_{k1}, \ldots, \beta_{kp})^\mathsf{T}$
- Approaches
  - One vs. all (one vs. rest): fit K separate logistic regression models

$$\log\left(\frac{\pi_{ik}}{1 - \pi_{ik}}\right) = \boldsymbol{\beta}_k^\mathsf{T}\mathbf{x}_i, \qquad (k = 1, \ldots K)$$

  - Generalized logit: fit $K - 1$ models. Pick class 1 as the base category, but, as with the binary logit, this choice is arbitrary (models equivalent with a different base category)

$$\log\left(\frac{\pi_{ik}}{\pi_{i1}}\right) = \boldsymbol{\beta}_k^\mathsf{T}\mathbf{x}_i, \qquad (k = 2, \ldots, K)$$

  - Many others: discriminant analysis (LDA/QDA), naive Bayes (NB), nearest neighbors (k-NN), tree-based (CART, RF, GBM, etc.), support vector machines (SVM), neural networks, etc.

Generalized logit model math

- How to estimate probabilities? Use Equations (1) and (2) below. A variation gives the softmax function
- The model is (omitting the observation subscript i)

$$\log\left(\frac{\pi_k}{\pi_1}\right) = \boldsymbol{\beta}_k^{\mathsf{T}}\mathbf{x}, \qquad (k = 2, \ldots, K)$$

- Solve for π_k:

$$\pi_k = \pi_1 \exp(\boldsymbol{\beta}_k^{\mathsf{T}}\mathbf{x}), \qquad (k = 2, \ldots, K)$$

- The probabilities must sum to 1

$$1 = \sum_{j=1}^{K} \pi_j = \pi_1 + \sum_{j=2}^{K} \pi_1 \exp(\boldsymbol{\beta}_j^{\mathsf{T}}\mathbf{x}) = \pi_1 \left[1 + \sum_{j=2}^{K} \exp(\boldsymbol{\beta}_j^{\mathsf{T}}\mathbf{x})\right]$$

- Solve for π_1:

$$\pi_1 = \frac{1}{1 + \sum_{j=2}^{K} \exp(\boldsymbol{\beta}_j^{\mathsf{T}}\mathbf{x})} \tag{1}$$

- Substitute back into formula for π_k:

$$\pi_k = \frac{\exp(\boldsymbol{\beta}_k^{\mathsf{T}}\mathbf{x})}{1 + \sum_{j=2}^{K} \exp(\boldsymbol{\beta}_j^{\mathsf{T}}\mathbf{x})} \qquad (k = 2, \ldots, K) \tag{2}$$

Generalized logit estimation

- Likelihood function

$$L(\mathbf{B}) = \prod_{i=1}^{n} \prod_{k=1}^{K} \pi_{ik}^{z_{ik}}$$

  where $z_{ik} = 1$ when $y_i = k$ and $z_{ik} = 0$ otherwise

- The log-likelihood is the cost function

$$\log L(\mathbf{B}) = \ell(\mathbf{B}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_{ik}$$

- Denote the optimal values of $\boldsymbol{\beta}_k$ by $\mathbf{b}_k$. Compute estimated probabilities that each observation is in class k as

$$p_{i1} = \frac{1}{1 + \sum_{j=2}^{K} \exp(\mathbf{b}_j^\mathsf{T} \mathbf{x}_i)} \quad \text{and} \quad p_{ik} = \exp(\mathbf{b}_k \mathbf{x}_i^\mathsf{T}) p_{i1}$$

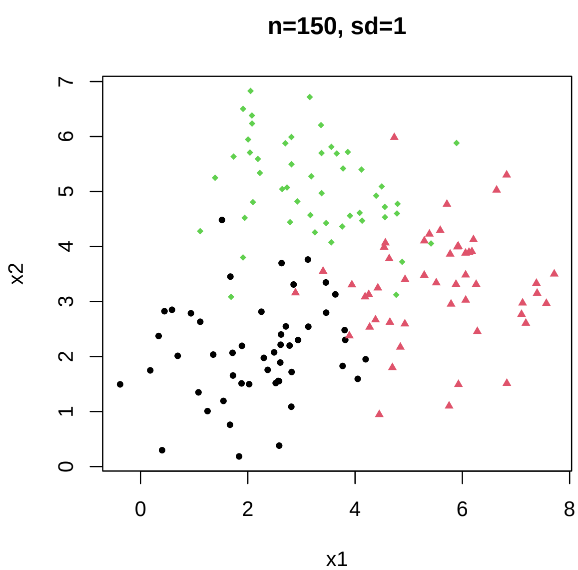
- The *maximum likelihood classifier* assigns i to the class with the largest p_{ik}.

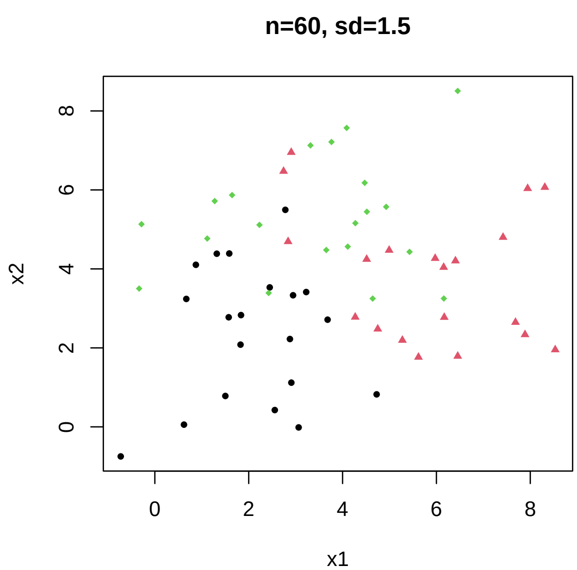
$K = 3$ Gaussian Mixture

- Define $K = 3$ mean vectors

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \begin{pmatrix} 3+\sqrt{5} \\ 1+\sqrt{5} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\mu}_3 = \begin{pmatrix} 1+\sqrt{5} \\ 3+\sqrt{5} \end{pmatrix}$$

- Let $\mathbf{x}$ come from class $y = k$ $(k = 1, \ldots, K)$, with a multivariate normal distribution $\mathrm{MVN}(\mu_k, \sigma^2 \mathbf{I})$






```
mu = matrix(2, nrow=3, ncol=2)
mu[2,] = sqrt(5)*c(1,1)+c(3,1)
mu[3,] = sqrt(5)*c(1,1)+c(1,3)
makedat = function(mu, n=50, sd=1.5,
    seed=12345){
  dat=data.frame(y=c(rep(1,n),
    rep(2,n),rep(3,n)))
  set.seed(seed)
  dat$x1 = c(
    rnorm(n, mu[1,1], sd),
    rnorm(n, mu[2,1], sd),
    rnorm(n, mu[3,1], sd)
  )
  dat$x2 = c(
    rnorm(n, mu[1,2], sd),
    rnorm(n, mu[2,2], sd),
    rnorm(n, mu[3,2], sd)
  )
  dat
}
test = makedat(mu, n=3000, seed=54321)
train = makedat(mu, n=20)
```

Comparing test-set performance

```
> library(nnet)
> fit2 = multinom(y~x1+x2, data=train)
> summary(fit2)

Coefficients:
  (Intercept)        x1        x2
2   -8.693704 1.4200155 0.9893357
3   -7.644341 0.6568461 1.5195375

Residual Deviance: 68.61044
AIC: 80.61044
```

Base category is $y = 1$

$$\log\left(\frac{\pi_2}{\pi_1}\right) = -8.69 + 1.42x_1 + 0.99x_2$$

$$\log\left(\frac{\pi_3}{\pi_1}\right) = -7.64 + 0.66x_1 + 1.52x_2$$

```
> # apply to test set and compute classification rate
> pred = predict(fit2, test, type="class") # apply to test set
> sum(diag(table(test$y, pred))) / nrow(test) # classification rate
[1] 0.7506667

> # one-versus-all
> fit3a = glm((y==1)~ x1+x2, binomial, train)
> fit3b = glm((y==2)~ x1+x2, binomial, train)
> fit3c = glm((y==3)~ x1+x2, binomial, train)
> ans = cbind(
+   predict(fit3a, test, type="resp"),
+   predict(fit3b, test, type="resp"),
+   predict(fit3c, test, type="resp")
+ )
> pred = apply(ans, 1, which.max)
> sum(diag(table(test$y, pred))) / nrow(test) # classification rate
[1] 0.75
```

Both models give similar results

Comparing different base categories

There are three possible base categories and all are equivalent

```
Call: multinom(formula = y ~ x1 + x2, data = train)

Coefficients:
  (Intercept)        x1        x2
2   -8.693704 1.4200155 0.9893357
3   -7.644341 0.6568461 1.5195375
Residual Deviance: 68.61044

multinom(formula = factor(y, levels = c(2, 1, 3)) ~ x1 + x2, data = train)
Coefficients:
  (Intercept)         x1         x2
1    8.691775 -1.4197191 -0.9891887
3    1.047763 -0.7629366  0.5302861
Residual Deviance: 68.61043

multinom(formula = factor(y, levels = 3:1) ~ x1 + x2, data = train)
Coefficients:
  (Intercept)         x1         x2
2   -1.047771  0.7629246 -0.5302758
1    7.643941 -0.6567638 -1.5194762
Residual Deviance: 68.61043
```

- All three models have the same deviance (cost function value)
- Let $\boldsymbol{\beta}_{ij}$ be slopes for $\log\left(\frac{\pi_i}{\pi_j}\right) = \boldsymbol{\beta}_{ij}^\mathsf{T}\mathbf{x}$. Note that $\frac{\pi_i}{\pi_j} = e^{\boldsymbol{\beta}_{ij}^\mathsf{T}\mathbf{x}}$
- Clearly, $\boldsymbol{\beta}_{ji} = -\boldsymbol{\beta}_{ij}$, e.g.,

$$\log\left(\frac{\pi_3}{\pi_1}\right) = \beta_{31}^\mathsf{T}\mathbf{x} \Longrightarrow \frac{\pi_3}{\pi_1} = e^{\beta_{31}^\mathsf{T}\mathbf{x}} \Longrightarrow \frac{\pi_1}{\pi_3} = e^{-\beta_{31}^\mathsf{T}\mathbf{x}}$$

- $\boldsymbol{\beta}_{23} = \boldsymbol{\beta}_{21} - \boldsymbol{\beta}_{31}$, e.g., $0.7629246 = 1.4200155 - 0.6568461$

$$\frac{\pi_2}{\pi_3} = \frac{\pi_2}{\pi_3}\cdot\frac{\pi_1}{\pi_1} = \frac{\pi_2}{\pi_1}\cdot\frac{\pi_1}{\pi_3} = e^{\beta_{21}^\mathsf{T}\mathbf{x}}\cdot e^{-\beta_{31}^\mathsf{T}\mathbf{x}} = e^{(\beta_{21}-\beta_{31})^\mathsf{T}\mathbf{x}}$$

Making predictions

Find the predicted probabities for $x_1 = x_2 = 4$. Recall (1) and (2):

$$\pi_1 = \frac{1}{1 + \sum_{j=2}^{K} \exp(\boldsymbol{\beta}_j^{\mathsf{T}} \mathbf{x})}$$

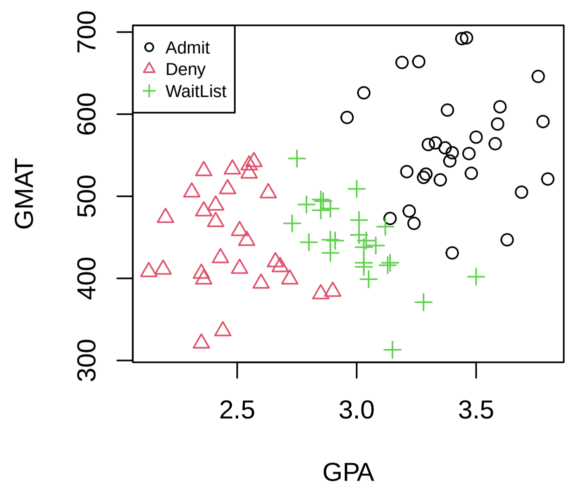
$$\pi_k = \pi_1 \exp(\boldsymbol{\beta}_k^{\mathsf{T}} \mathbf{x}), \qquad (k = 2, \ldots, K)$$

```
> predict(fit2, data.frame(x1=4, x2=4), type="prob")
        1         2         3
0.1548158 0.3977949 0.4473893
> predict(fit2, data.frame(x1=4, x2=4)) # ML predict is to class 3
[1] 3
Levels: 1 2 3

> # now do it by hand
> coef(fit2)
  (Intercept)        x1        x2
2   -8.693704 1.4200155 0.9893357
3   -7.644341 0.6568461 1.5195375
> (eta = coef(fit2) %*% c(1,4,4))
       [,1]
2 0.9437007
3 1.0611934
> (pi1 = 1/(1+sum(exp(eta)))) # matches predict function
[1] 0.1548158
> pi1*exp(eta) # matches predict function
       [,1]
2 0.3977949
3 0.4473893
```

GMAT-GPA Example

Build a machine classifer to process applicants into admit, wait list or deny based on their GMAT and undergraduate GPA scores

```
> library(dplyr); library(nnet)
> gpa=read.table("gmatgpa.dat",header=T) %>%
  mutate(y = factor(admit, 1:3,
    c("Admit", "Deny", "WaitList")))
> names(gpa)
[1] "GPA"    "GMAT"   "admit" "y"
> plot(GMAT ~ GPA, data=gpa,
    col=admit, pch=admit)
> legend("topleft", levels(gpa$y),
   col=1:3, pch=1:3, cex=.7)
> # takes a long time to converge
> (fit = multinom(y~GPA+GMAT, data=gpa,
    maxit=1000))

Coefficients:
        (Intercept)         GPA       GMAT
Deny        485.9823 -117.37076 -0.3227173
WaitList    167.3553  -31.06165 -0.1458875

Residual Deviance: 10.78435
AIC: 22.78435
```




```
> apply(coef(fit), 2, diff) #wait/deny
 (Intercept)          GPA        GMAT
-318.6269550   86.3091150   0.1768298
```

- A one-point increase in GPA is associated with the log odds of being denied versus admitted decreasing by 117
- A one-point increase in GPA is associated with the log odds of being wait listed versus denied increasing by 86

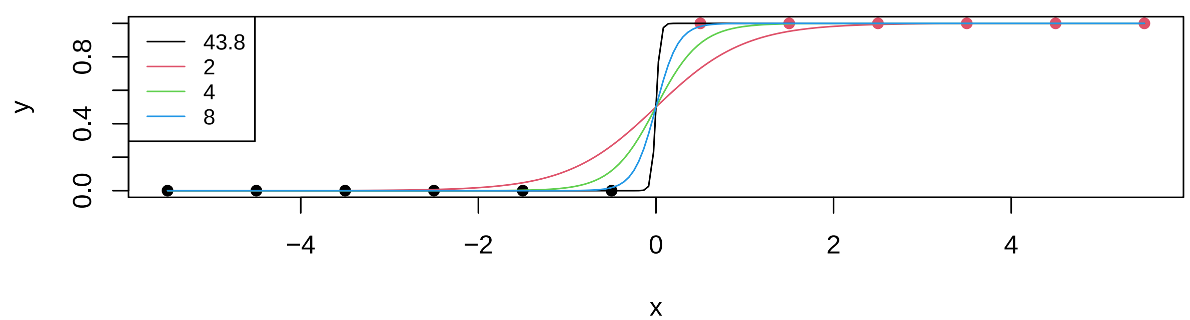
Evaluating predictions

- See Sokolova and Lapalme (2009). This blog gives R code.
- Start with **accuracy**
- **Per-class precision, recall and F_1**: compute the three metrics for each class individually
- **Macro precision, recall and F_1**: average the per-class values.

```
> predicted = factor(apply(fit$fitted.values, 1, which.max), 1:3, c("Padmit", "Pdeny", "Pwait"))
> (cm=table(actual=gpa$y, predicted)) # confusion matrix
          predicted
actual     Padmit Pdeny Pwait
  Admit        30     0     1
  Deny          0    28     0
  WaitList      2     1    23
> (n = sum(cm)) # number of instances
[1] 85
> (rowsums = apply(cm, 1, sum)) # number of instances per class
   Admit     Deny WaitList
      31       28       26
> (colsums = apply(cm, 2, sum)) # number of predictions per class
Padmit  Pdeny  Pwait
    32     29     24
> (accuracy = sum(diag(cm)) / n ) # accuracy
[1] 0.9529412
> (precision = diag(cm) / colsums) # per-class precision
   Padmit     Pdeny     Pwait
0.9375000 0.9655172 0.9583333
> (recall = diag(cm) / rowsums) # per-class recall
    Admit      Deny  WaitList
0.9677419 1.0000000 0.8846154
> (f1 = 2 * precision * recall / (precision + recall) ) # per-class f1
   Padmit     Pdeny     Pwait
0.9523810 0.9824561 0.9200000
> mean(precision) # macro Precision
[1] 0.9537835
> mean(recall) # macro Recall
[1] 0.9507858
> mean(f1) # macro F1
[1] 0.9516124
```

Problem of perfect separation

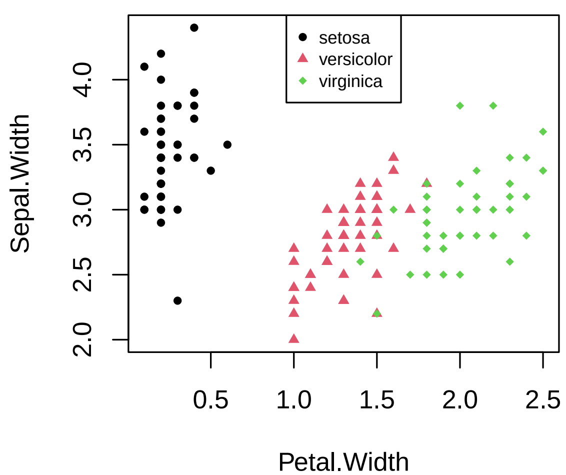
```
> dat = data.frame(x = (-5:6)-.5)
> dat$y = dat$x>0
> plot(dat, col=dat$y+1)
> fit2 = glm(y~x, binomial, dat)
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(fit2)

Call:
glm(formula = y ~ x, family = binomial, data = dat)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.484e-05  -2.110e-08   0.000e+00   2.110e-08   2.484e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 4.525e-09  2.442e+04   0.000    1.000
x           4.380e+01  4.884e+04   0.001    0.999

    Null deviance: 1.6636e+01  on 11  degrees of freedom
Residual deviance: 1.2340e-09  on 10  degrees of freedom
Number of Fisher Scoring iterations: 25

> x = seq(-5.5, 5.5, length=200)
> lines(x, predict(fit2, data.frame(x=x), type="resp"))
> logit = function(slope, col, x) lines(x, 1/(1+exp(-slope*x)), col=col)
> logit(2, 2, x); logit(4, 3, x); logit(8, 4, x)
> legend("topleft", c("43.8", "2", "4", "8"), lty=1, col=1:4, cex=.7)
```



Iris example

This is the famous Fisher's Iris data set

```
> plot(Sepal.Width~Petal.Width, iris, col=Species,
    pch=15+as.numeric(iris$Species), cex=.7)
> legend("top", 1:3, levels(iris$Species), col=1:3, pch=16:18, cex=.7)
> fit = multinom(Species~Sepal.Width+Petal.Width, iris, maxit=1000)
> rbind(
    coef(fit),
    greenOVERred=apply(coef(fit), 2, diff) # log(virg/versi)=log(green/red)
  )
             (Intercept) Sepal.Width Petal.Width
versicolor      6.323967  -16.364197    47.26169
virginica      -8.054705  -20.271892    62.96259
greenOVERred  -14.378672   -3.907695    15.70090
```




```
> predicted = factor(
   apply(fit$fitted.values,1,which.max),
   1:3, c("Pset", "Pver", "Pvir"))
> (cm=table(iris$Species, predicted))
            predicted
             Pset Pver Pvir
  setosa       50    0    0
  versicolor    0   46    4
  virginica     0    3   47
```

Categorical data models in general

- **Binary logit**: individual chooses between two options and selects the one that provide the greater utility
- **Multinomial logit**: individual chooses among more than two alternatives and selects the one that provides the greatest utility
- **Ordered logit**: individual reveals the strength of his or her preferences with respect to a single outcome (ACT §7.5.2)
- **Conditional logit**: allows predictor variables that <u>vary across alternatives</u> and possibly across the individuals as well, e.g., choice of mode of transportation (e.g., train, bus, car). Characteristics or attributes of these include time waiting, how long it takes to get to work, and cost.
- **Log-linear analysis**: class of models that subsumes the logit models and more. See Agresti (2006), *Categorical Data Analysis*.

Poisson

- The pmf of a Poisson distribution is

$$\mathsf{P}(Y = y) = \frac{\mu^y e^{-\mu}}{y!}, \quad \mu > 0,\ y = 0, 1, 2, \ldots$$

$$\mathbb{E}(Y) = \mathbb{V}(Y) = \mu$$

- We observe a random sample $(x_i, y_i), i = 1, \ldots, n$, where y_i has a Poisson distribution with mean μ_i. More generally, x_i could be a vector of predictors. Assume (log link function)

$$\log \mu_i = \eta_i = \alpha + \beta x_i \quad \Longleftrightarrow \quad \mu_i = e^{\eta_i}$$

- The likelihood is

$$L = \prod_{i=1}^{n} \frac{e^{-\mu_i} \mu_i^{y_i}}{y_i!}$$

- The log-likelihood of a Poisson model is

$$\ell = \log L = \sum_{i=1}^{n} [y_i \log \mu_i - \mu_i - \log(y_i!)]$$

- The log-likelihood of a saturated Poisson model is

$$\hat{\mu}_i \equiv y_i \quad \Longrightarrow \quad \ell_s = \sum_{i=1}^{n} [y_i \log y_i - y_i - \log(y_i!)]$$

- Let $\hat{\mu}_i = \exp(\hat{\alpha} + \hat{\beta} x_i)$ be the MLEs. The deviance is

$$D = -2(\ell - \ell_s) = -2 \left[\sum_{i=1}^{n} y_i \log \frac{\hat{\mu}_i}{y_i} + \sum_{i=1}^{n} (y_i - \hat{\mu}_i) \right],$$

where $y_i \log(\hat{\mu}_i / y_i) = 0$ for $y_i = 0$.

Simple example

```
> set.seed(12345)
> n = 15
> dat = data.frame(x=as.integer(runif(n)*20))
> dat$y = rpois(n, exp(dat$x/2-7)) # slope=.5, intercept=-7
> t(dat)

x  14  17  15  17   9   3   6  10  14   19    0    3   14    0    7
y   1   4   1   3   1   0   0   1   1   13    0    0    0    0    0

> fit = glm(y ~ x, poisson, dat)
> summary(fit)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.97092    1.61686  -3.693 0.000222 ***
x            0.43792    0.09372   4.673 2.97e-06 ***
    Null deviance: 58.8294  on 14  degrees of freedom
Residual deviance:  8.0564  on 13  degrees of freedom
AIC: 32.73

> muhat = fit$fitted.values             # =exp(coef(fit)[1]+coef(fit)[2]*dat$x)
> term1 = ifelse(dat$y==0, 0, dat$y*log(muhat/dat$y))

> -2*sum(term1 + dat$y-muhat)
[1] 8.056363
> deviance(fit) # both equal 8.056363
[1] 8.056363

> (l=sum(dat$y*log(muhat) - muhat - log(factorial(dat$y))))
[1] -14.3648
> logLik(fit)   # both equal -14.3648
'log Lik.' -14.3648 (df=2)

> -2*l + 2*(1+1) # AIC
[1] 32.72961
```

Some more results on Poisson regression

- See here for a good summary
- **Raw residuals** don't account for heteroscedastic errors:

$$r_i = y_i - \exp(\hat{\alpha} + \hat{\beta} x_i) = y_i - \hat{\mu}_i$$

- **Pearson residuals**

$$e_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}}$$

  We can think of this as a Z statistic since $\sqrt{\mathbb{V}(Y_i)} = \sqrt{\mu_i}$. Sometime there is an additional constant in the denominator to account for overdispersion.

- There are also **deviance residuals**, whose sum of squared values equal the deviance, but the `plot` function in R displays Pearson residuals.
- Sometimes a **Pseudo-R^2** is reported

$$\text{Pseudo } R^2 = 1 - \frac{\ell(\hat{\beta})}{\ell(\hat{\beta}_0)},$$

  where $\ell(\hat{\beta})$ and $\ell(\hat{\beta}_0)$ are the log-likelihoods of the full and null models, respectively. This is analogous to $R^2 = 1-\text{SSE}/\text{SST}$.

Geriatric Study

(Kutner, et al. problem 14.39) A research geriatrics designed a prospective study to investigate the effects of two interventions on the frequency of falls. One hundred subjects were randomly assigned to one of the two interventions: education only ($x_1 = 0$) or education plus aerobic exercise training ($x_1 = 1$). Subjects were at least 65 years of age and in reasonably good health. Three variables considered to be important as control variables were gender ($x_2 = 0$ for female, 1 for male), balance index (x_3) and a strength index (x_4). The higher the balance index, the more stable is the subject, and the higher the stregth index, the stronger the subject is. Each subject kept a diary recording the number of falls Y during the six months of the study.

1. Examine the data for problems and get to know the variables.
2. Fit a Poisson regression with $\log \mu = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}$. State the estimated coefficients, their standard errors, and the estimated response function. Check the residual plot for problems.
3. Test overall significance of the model.
4. Test significance ($\alpha = 0.05$) of individual slopes using Z tests and LRT.
5. Compare with a linear model using a log variance stabilizing transformation.
6. Drop male from the model and state the estimated model.
7. Test the significance of the slopes.
8. Estimate the predicted number of falls for subject 100 by hand and using software.
9. Do we need any covariates (strength, balance, male)? Estimate a model using only `intervention` and compare with other models.
10. Use the model from the previous part to estimate the probability that a subject has at least one fall for both those with and without the intervention, using the Poisson assumpion.

Solution to Geriatric Study

```
# Geriatric Study
dat = read.table("CH14PR39.txt", header=T)
# Q1
head(dat)
dim(dat)
pairs.panels(dat[,c(2:5,1)]) # want y on vertial axis
summary(dat) # y is between 0 and 11
fit = glm(y~., poisson, dat) # Q2
summary(fit)
plot(fit, which=1)  # no problems
vif(fit)

# Q3 Overall sig test
(teststat=fit$null.deviance-fit$deviance)
1-pchisq(teststat,4)

# Q4 test individual coefs.
summary(fit)
drop1(fit, test="LRT")
exp(coef(fit)) # mult effect on means

summary(lm(log(y+1)~., dat)) #Q5

# Q6 drop male
fit2 = glm(y~.-male, poisson, dat) #Q6
summary(fit2) #Q7

# Q8 row 100: intervention=0, balance=37, strength=56, y=2
dat[100,]
(eta=t(coef(fit2)) %*% c(1, 0, 37, 56)) # eta=1.297158
exp(eta)  # 3.658884
fit2$fitted.values[100] # muhat = 3.658884
predict(fit2, data.frame(dat[100,])) # gives eta
predict(fit2, data.frame(dat[100,]), type="resp") # gives muhat
1-fit2$deviance/fit2$null.deviance # Pseudo R^2

fit3 = glm(y~intervention, poisson, dat)
summary(fit3)
exp(coef(fit3))
(muhat=predict(fit3, data.frame(intervention=0:1), type="resp"))
tapply(dat$y, dat$intervention, mean) # simple means
1-fit3$deviance/fit3$null.deviance # Pseudo R^2
1-dpois(0, muhat) # Q10
1-exp(-muhat) # same with PMF directly
```

Beyond Poisson

- ACT §9.4.1 shows how to predict rates (instead of counts)
- Recall that for Poisson Y, $\mathbb{E}(Y) = \mathbb{V}(Y) = \mu$, but in practice we may find $\mathbb{E}(Y) < \mathbb{V}(Y)$, called the problem of **overdispersion** (or **underdisperson** $\mathbb{E}(Y) > \mathbb{V}(Y)$).
- Negative binomial distribution (NBD) regression models allow for overdispersion
- Beyond overdispersion, we often observe too many zero values for either Poisson or NBD. What to do?
  - Zero-inflated Poisson (ZIP) assumes a mixture distriubtion

    $$\mathsf{P}(Y = 0) = \varphi + (1 - \varphi)e^{-\mu},$$

    $$\mathsf{P}(Y = y) = (1 - \varphi)\frac{e^{-\mu}\mu^y}{y!}, \quad \mu > 0, y = 1, 2, \ldots$$

    where $\varphi \in [0, 1]$ is the probability of extra (structural) zeros
  - Zero-inflated negative binomial (ZINP)
  - Hurdle models
- GLMs can accommodate other distributions including exponential and gamma (ACT §9.5)
- Classic reference: McCullagh and Nelder, *Generalized Linear Models*