Data Mining
Logistic Regression
Text Classification

Ad Feelders

Universiteit Utrecht

Two types of approaches to classification

In (probabilistic) classification we are interested in the conditional distribution

$$P(Y \mid x),$$

so that, for example, when we observe $X = x$ we can predict the class y with the highest probability for that value of X.

There are two basic approaches to modeling $P(Y \mid x)$:

- Generative Models (use Bayes' rule):

$$P(Y = y \mid x) = \frac{P(x \mid Y = y)P(Y = y)}{P(x)} = \frac{P(x \mid Y = y)P(Y = y)}{\sum_{y'} P(x \mid Y = y')P(Y = y')}$$

- Discriminative Models: model $P(Y \mid x)$ directly.

Generative Models

Examples of generative classification methods:

- Naive Bayes classifier (discussed in the previous lecture)
- Linear/Quadratic Discriminant Analysis (not discussed)
- ...

Discriminative Models

Discriminative methods only model the *conditional* distribution of Y given X. The probability distribution of X itself is not modeled.

For the binary classification problem:

$$P(Y = 1 \mid X) = f(X, \beta)$$

where $f(X, \beta)$ is some function of features X and parameters β.

Discriminative Models

Examples of discriminative classification methods:

- Linear probability model
- Logistic regression
- Feed-forward neural networks
- ...

Discriminative Models: linear probability model

Consider the linear regression model

$$\mathbb{E}[Y \mid x] = \beta^\top x \qquad\qquad Y \in \{0, 1\},$$

where

$$\beta^\top x = \sum_{j=0}^{m} \beta_j x_j, \qquad\qquad \text{with } x_0 \equiv 1.$$

But

$$\begin{aligned} \mathbb{E}[Y \mid x] &= 1 \cdot P(Y = 1 \mid x) + 0 \cdot P(Y = 0 \mid x) \\ &= P(Y = 1 \mid x) \end{aligned}$$

So the model assumes that

$$P(Y = 1 \mid x) = \beta^\top x$$

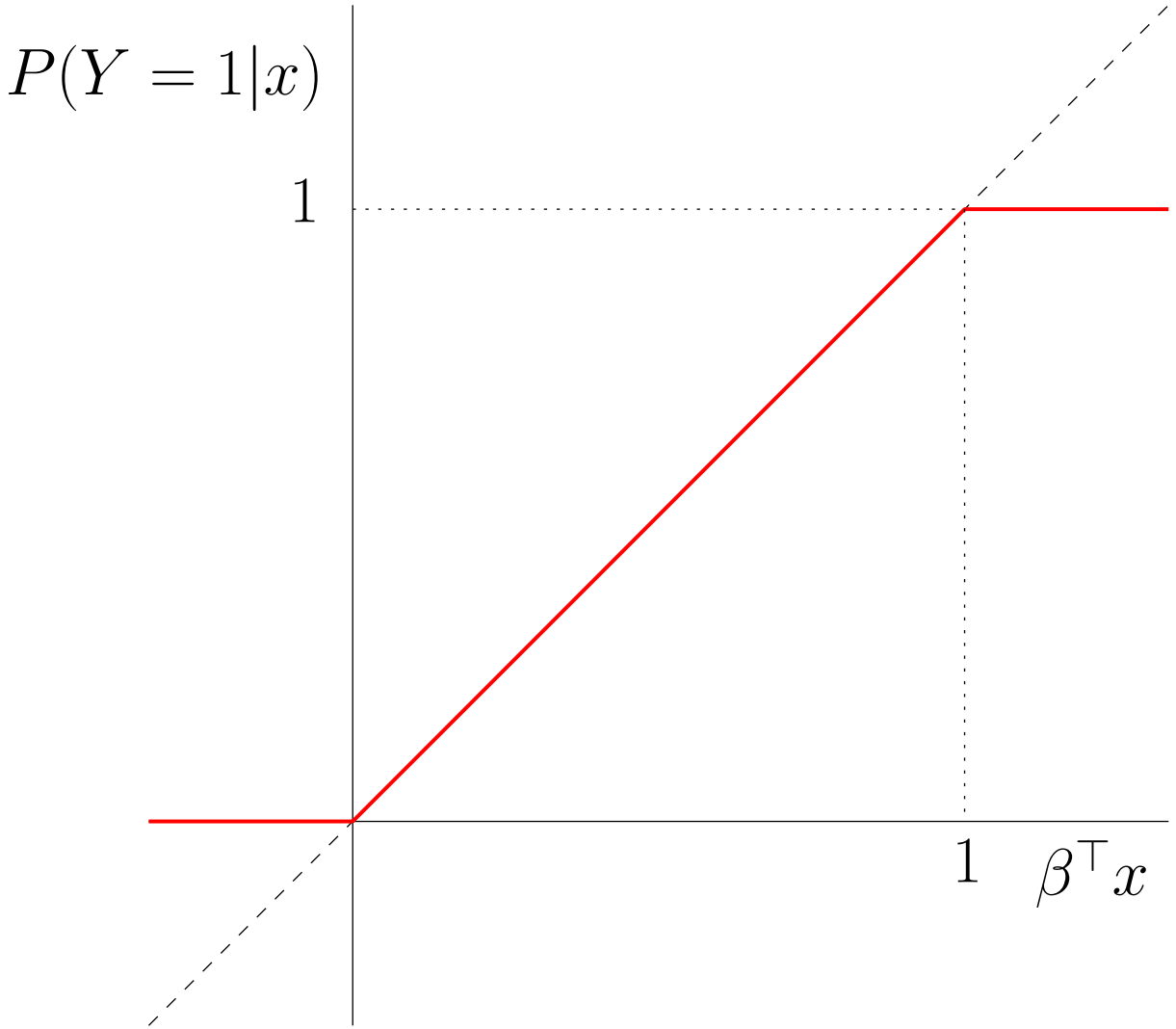
Notation

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix} \qquad x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{pmatrix}$$

with $x_0 \equiv 1$, so

$$\beta^\top x = \sum_{j=0}^{m} \beta_j x_j = \beta_0 + \beta_1 x_1 + \ldots + \beta_m x_m$$

Linear response function



Logistic regression

The linear probability model allows negative "probabilities" and "probabilities" bigger than one.

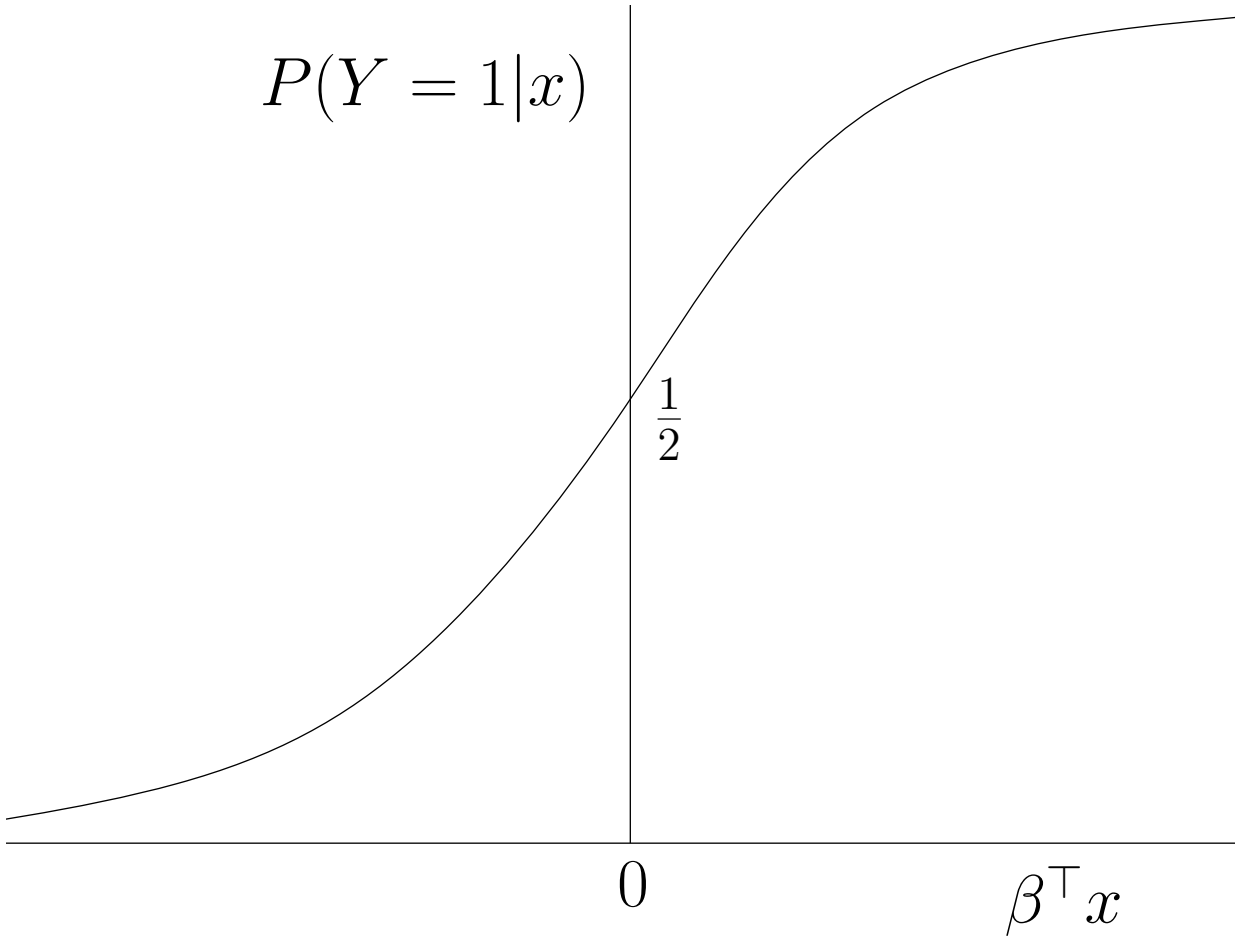
Logistic response function

$$\mathbb{E}[Y \mid x] = P(Y = 1 \mid x) = \frac{e^{\beta^\top x}}{1 + e^{\beta^\top x}}$$

or (divide numerator and denominator by $e^{\beta^\top x}$)

$$P(Y = 1 \mid x) = \frac{1}{1 + e^{-\beta^\top x}} = (1 + e^{-\beta^\top x})^{-1}$$

Logistic Response Function



Linearization: the logit transformation

Since $P(Y = 1 \mid x)$ and $P(Y = 0 \mid x)$ have to add up to 1, we have:

$$P(Y = 1 \mid x) = \frac{e^{\beta^\top x}}{1 + e^{\beta^\top x}} \quad \Rightarrow \quad P(Y = 0 \mid x) = \frac{1}{1 + e^{\beta^\top x}}$$

Hence,

$$\frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)} = e^{\beta^\top x}$$

Therefore:

$$\ln \left\{ \frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)} \right\} = \beta^\top x$$

The ratio

$$\frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)}$$

is called the *odds*.

Linear Decision Boundary

Assign to class 1 if $P(Y = 1 \mid x) > P(Y = 0 \mid x)$, i.e. if

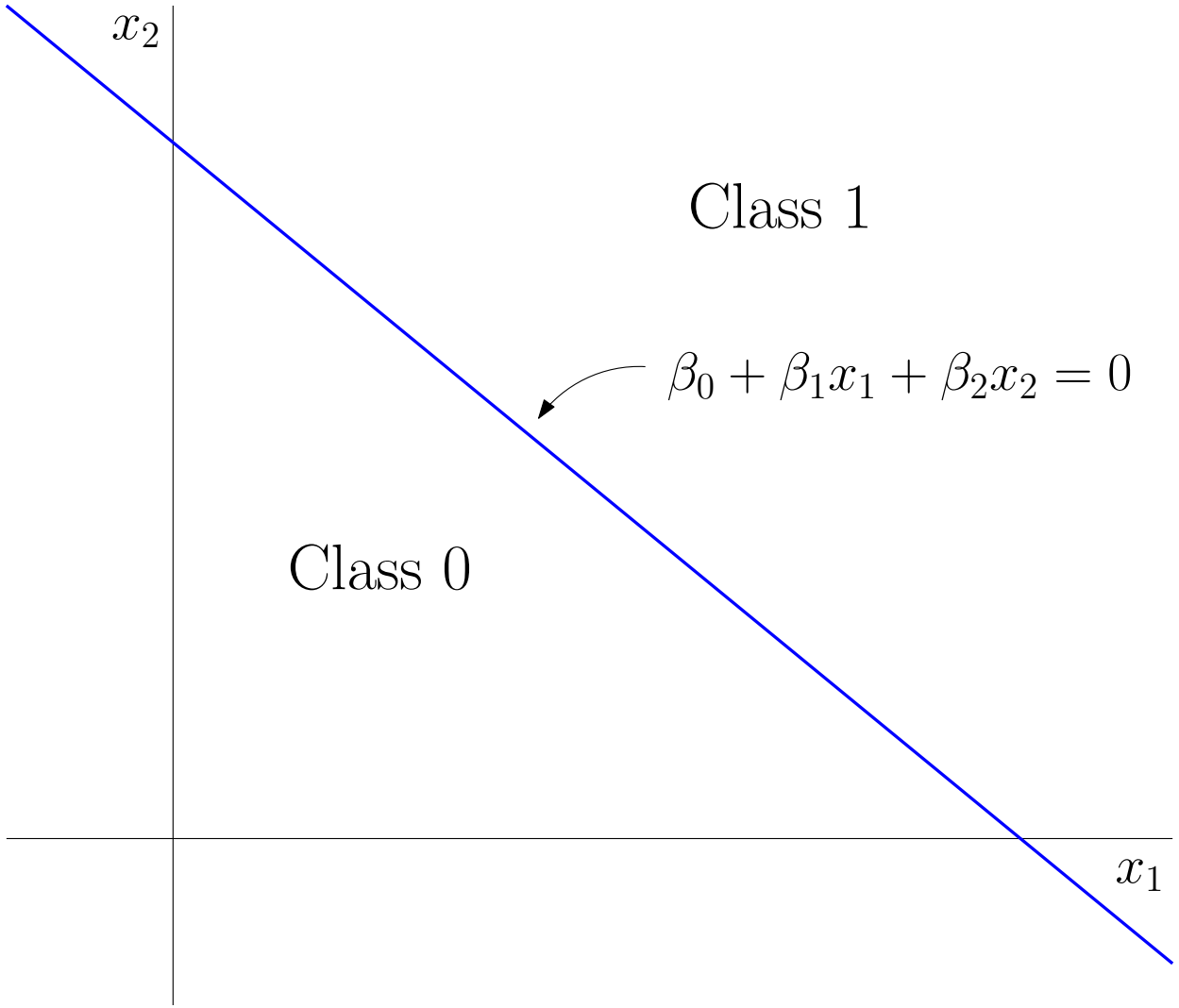
$$\frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)} > 1$$

This is true if

$$\ln \left\{ \frac{P(Y = 1 \mid x)}{P(Y = 0 \mid x)} \right\} > 0$$

So assign to class 1 if $\beta^\top x > 0$, and to class 0 otherwise.

Linear Decision Boundary



Maximum Likelihood Estimation

Coin tossing example:

$Y = 1$ if heads, $Y = 0$ if tails. Parameter $p = P(Y = 1)$.
One coin flip

$$P(y) = p^y(1-p)^{1-y}$$

Note that $P(1) = p$, $P(0) = 1 - p$ as required.
Sequence of n independent coin flips

$$P(y_1, y_2, ..., y_n) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i}$$

which defines the likelihood function when viewed as a function of p.

Maximum Likelihood Estimation

In a sequence of 10 coin flips we observe $y = (1, 0, 1, 1, 0, 1, 1, 1, 1, 0)$.

The corresponding likelihood function is

$$\begin{aligned} L(p \mid y) &= p \cdot (1-p) \cdot p \cdot p \cdot (1-p) \cdot p \cdot p \cdot p \cdot p \cdot (1-p) \\ &= p^7 (1-p)^3 \end{aligned}$$

The corresponding log-likelihood function is

$$\ell(p \mid y) = \ln L(p \mid y) = \ln(p^7 (1-p)^3) = 7 \ln p + 3 \ln(1-p)$$

Find the value of p that maximizes this function.

Computing the maximum

To determine that value, we take the derivative, equate it to zero and solve for p.

Recall that

$$\frac{d \ln x}{dx} = \frac{1}{x}$$

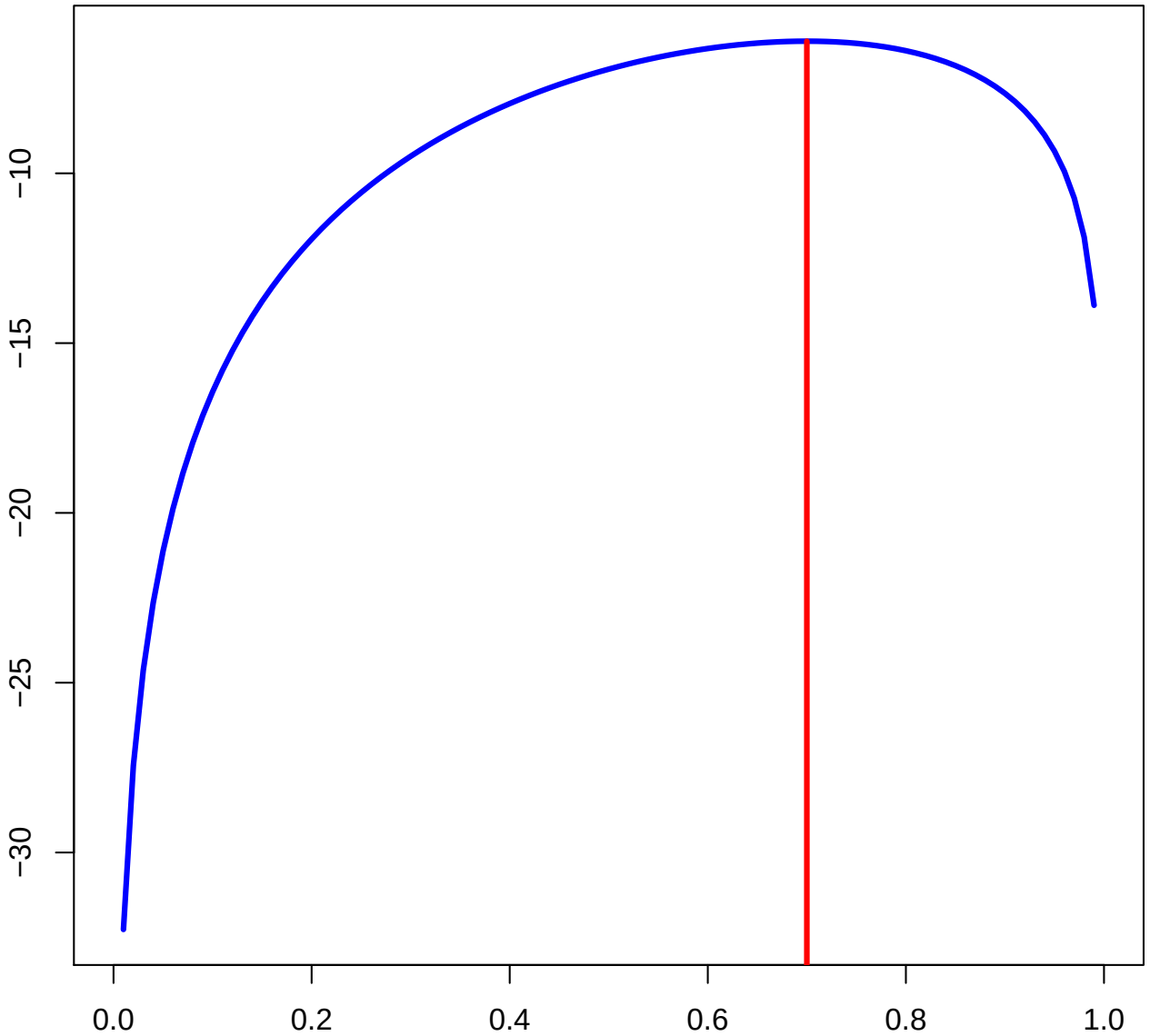
So the derivative of the log-likelihood function with respect to p is:

$$\frac{d\ell(p \mid y)}{dp} = \frac{7}{p} - \frac{3}{1-p}$$

Equating to zero, and solving for p yields maximum likelihood estimate $\hat{p} = 0.7$.

This is just the relative frequency of heads in the sample!

Log-likelihood function for $y = (1, 0, 1, 1, 0, 1, 1, 1, 1, 0)$



ML estimation for logistic regression

Logistic regression is similar to the coin tossing example, except that now the probability of success p_i depends on x_i and β:

$$\begin{aligned} p_i &= P(Y = 1 \mid x_i) = (1 + e^{-\beta^\top x_i})^{-1} \\ 1 - p_i &= P(Y = 0 \mid x_i) = (1 + e^{\beta^\top x_i})^{-1} \end{aligned}$$

we can represent its probability distribution as follows

$$P(y_i) = p_i^{y_i}(1 - p_i)^{1-y_i} \qquad y_i \in \{0, 1\}; \quad i = 1, \ldots, n$$

ML estimation for logistic regression

Example

i	x_i	y_i	$P(y_i)$
1	8	0	$(1+e^{\beta_0+8\beta_1})^{-1}$
2	12	0	$(1+e^{\beta_0+12\beta_1})^{-1}$
3	15	1	$(1+e^{-\beta_0-15\beta_1})^{-1}$
4	10	1	$(1+e^{-\beta_0-10\beta_1})^{-1}$

The likelihood function is:

$$(1+e^{\beta_0+8\beta_1})^{-1} \times (1+e^{\beta_0+12\beta_1})^{-1} \times (1+e^{-\beta_0-15\beta_1})^{-1} \times (1+e^{-\beta_0-10\beta_1})^{-1}$$

ML Estimation: find values of β_0 and β_1 that maximize this probability.

Logistic Regression: likelihood function

Since the y_i observations are assumed to be independent (e.g. random sampling):

$$P(y) = \prod_{i=1}^{n} P(y_i) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

Or, taking the natural logarithm:

$$\begin{aligned} \ln P(y) &= \ln \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \\ &= \sum_{i=1}^{n} \{y_i \ln p_i + (1-y_i)\ln(1-p_i)\} \end{aligned}$$

Logistic Regression: log-likelihood function

For the logistic regression model we have

$$\begin{aligned} p_i &= (1 + e^{-\beta^\top x_i})^{-1} \\ 1 - p_i &= (1 + e^{\beta^\top x_i})^{-1} \end{aligned}$$

so filling in gives

$$\ell(\beta \mid y) = \sum_{i=1}^{n} \left\{ y_i \ln \left(\frac{1}{1 + e^{-\beta^\top x_i}} \right) + (1 - y_i) \ln \left(\frac{1}{1 + e^{\beta^\top x_i}} \right) \right\}$$

- Non-linear function of the parameters.
- No closed form solution (no nice formulas for the parameter estimates).
- Likelihood function globally concave so relatively easy optimization problem (no local maxima).

Fitted Response Function

Substitute maximum likelihood estimates into the response function to obtain the *fitted response function*

$$\hat{P}(Y = 1 \mid x) = \frac{e^{\hat{\beta}^\top x}}{1 + e^{\hat{\beta}^\top x}}$$

Example: Programming Assignment

Model the probability of successfully completing a programming assignment.

Explanatory variable: "months of programming experience".
We find $\hat{\beta}_0 = -3.0597$ and $\hat{\beta}_1 = 0.1615$, so

$$\hat{P}(Y = 1 \mid x) = \frac{e^{-3.0597+0.1615x}}{1 + e^{-3.0597+0.1615x}}$$

14 months of programming experience:

$$\hat{P}(Y = 1 \mid x = 14) = \frac{e^{-3.0597+0.1615(14)}}{1 + e^{-3.0597+0.1615(14)}} \approx 0.31$$

Example: Programming Assignment

	month.exp	success		month.exp	success
1	14	0	16	13	0
2	29	0	17	9	0
3	6	0	18	32	1
4	25	1	19	24	0
5	18	1	20	13	1
6	4	0	21	19	0
7	18	0	22	4	0
8	12	0	23	28	1
9	22	1	24	22	1
10	6	0	25	8	1
11	30	1			
12	11	0			
13	30	1			
14	5	0			
15	20	1			

Interpretation

We have

$$\ln\left\{\frac{\hat{P}(Y=1\mid x)}{\hat{P}(Y=0\mid x)}\right\} = -3.0597 + 0.1615x,$$

so with every additional month of programming experience, the log odds increase with 0.1615.

The odds are multiplied by $e^{0.1615} \approx 1.175$ so with every additional month of programming experience, the odds increase with 17.5%.

When x increases with one unit, the odds are multiplied by e^{β_1} because:

$$e^{\beta_0+\beta_1(x+1)} = e^{\beta_0+\beta_1 x+\beta_1} = e^{\beta_0+\beta_1 x} \times e^{\beta_1},$$

since $e^{a+b} = e^a \times e^b$.

Interpretation

Note that the effect of an increase in x on the *probability* of success depends on the value of x:

- An increase from 14 to 24 months of programming experience leads to an increase of the probability of success from 0.31 to 0.69.
- An increase from 34 to 44 months of programming experience leads to an increase of the probability of success from 0.92 to 0.98.

Allocation Rule

Probability of the classes is equal when

$$-3.0597 + 0.1615x = 0$$

Solving for x we get $x \approx 18.95$.

Allocation Rule:

$x \geq 19$: *predict* $y = 1$
$x < 19$: *predict* $y = 0$

If a person has 19 months or more programming experience, predict success, otherwise predict failure.

Programming Assignment: Confusion Matrix

Cross table of observed and predicted class label:

	0	1
0	11	3
1	3	8

Row: observed, Column: predicted

Error rate: 6/25=0.24

Default (predict majority class): 11/25=0.44

How to in R

```
> prog.logreg <- glm(success ~ month.exp, data=prog.dat, family=binomial)
> summary(prog.logreg)


Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.05970    1.25935  -2.430   0.0151 *
month.exp    0.16149    0.06498   2.485   0.0129 *
---

Number of Fisher Scoring iterations: 4


> table(prog.dat$success, as.numeric(prog.logreg$fitted > 0.5))


    0  1
  0 11  3
  1  3  8
```

Regularization

- If we have a large number of predictors, even a linear model estimated with maximum likelihood can be prone to overfitting.
- This can be controlled by punishing large (positive or negative) weights. The coefficient estimates are *shrunken* towards zero.
- In this way we trade off bias against variance.
- Add a penalty term for the size of the coefficients to the objective function.
- With LASSO penalty:

$$E(\beta) = -\ell(\beta) + \lambda \sum_{j=1}^{m} |\beta_j|,$$

  where $E(\beta)$ is the new error function that we want to minimize, and $-\ell(\beta)$ is the negative log-likelihood function.
- The value of λ is usually selected by cross-validation.

Movie Reviews: IMDB Review Dataset

- Collection of 50,000 reviews from IMDB, allowing no more than 30 reviews per movie.
- Contains an even number of positive and negative reviews, so random guessing yields 50% accuracy.
- Considers only highly polarized reviews. A negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10.
- Neutral reviews are not included in the dataset.

Andrew L. Maas et al., *Learning Word Vectors for Sentiment Analysis*, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142–150,2011.

Data available at:

`http://ai.stanford.edu/~amaas/data/sentiment/`

Analysis of Movie Reviews in R

```
# load the tm package
> library(tm)
# Read in the data using UTF-8 encoding
> reviews.neg <- VCorpus(DirSource("D:/MovieReviews/train/neg",
                         encoding="UTF-8"))
> reviews.pos <- VCorpus(DirSource("D:/MovieReviews/train/pos",
                         encoding="UTF-8"))
# Join negative and positive reviews into a single Corpus
> reviews.all <- c(reviews.neg,reviews.pos)
# create label vector (0=negative, 1=positive)
> labels <- c(rep(0,12500),rep(1,12500))
> reviews.all
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 25000
```

Analysis of Movie Reviews

The first review before pre-processing:

```
> as.character(reviews.all[[1]])
[1] "Story of a man who has unnatural feelings for a pig.
Starts out with a opening scene that is a terrific example
of absurd comedy. A formal orchestra audience is turned into
an insane, violent mob by the crazy chantings of it's singers.
Unfortunately it stays absurd the WHOLE time with no
general narrative eventually making it just too off putting.
Even those from the era should be turned off.
The cryptic dialogue would make Shakespeare seem easy to a
third grader. On a technical level it's better than you might
think with some good cinematography by future great Vilmos Zsigmond.
Future stars Sally Kirkland and Frederic Forrest can be seen briefly."
```

Analysis of Movie Reviews: Pre-Processing

```
# Remove punctuation marks (comma's, etc.)
> reviews.all <- tm_map(reviews.all,removePunctuation)
# Make all letters lower case
> reviews.all <- tm_map(reviews.all,content_transformer(tolower))
# Remove stopwords
> reviews.all <- tm_map(reviews.all, removeWords,
                        stopwords("english"))
# Remove numbers
> reviews.all <- tm_map(reviews.all,removeNumbers)
# Remove excess whitespace
> reviews.all <- tm_map(reviews.all,stripWhitespace)
```

Not done: stemming, part-of-speech tagging, ...

Analysis of Movie Reviews

The first review after pre-processing:

```
> as.character(reviews.all[[1]])
[1] "story man unnatural feelings pig starts opening scene terrific
example absurd comedy formal orchestra audience turned insane violent
mob crazy chantings singers unfortunately stays absurd whole time
general narrative eventually making just putting even era turned
cryptic dialogue make shakespeare seem easy third grader technical
level better might think good cinematography future great vilmos
zsigmond future stars sally kirkland frederic forrest can seen briefly"
```

Analysis of Movie Reviews

```
# draw training sample (stratified)
# draw 8000 negative reviews at random
> index.neg <- sample(12500,8000)
# draw 8000 positive reviews at random
> index.pos <- 12500+sample(12500,8000)
> index.train <- c(index.neg,index.pos)

# create document-term matrix from training corpus
> train.dtm <- DocumentTermMatrix(reviews.all[index.train])
> dim(train.dtm)
[1] 16000 92819
```

We've got 92,819 features. Perhaps this is a bit too much.

```
# remove terms that occur in less than 5% of the documents
# (so-called sparse terms)

> train.dtm <- removeSparseTerms(train.dtm,0.95)
> dim(train.dtm)
[1] 16000   306

# create document term matrix for test set
> test.dtm <- DocumentTermMatrix(reviews.all[-index.train],
                list(dictionary=dimnames(train.dtm)[[2]]))
> dim(test.dtm)
[1] 9000  306
```

Analysis of Movie Reviews

```
# view a small part of the document-term matrix
> inspect(train.dtm[100:110,80:85])

<<DocumentTermMatrix (documents: 11, terms: 6)>>
Non-/sparse entries: 7/59
Sparsity           : 89%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
             Terms
Docs          family fan far father feel felt
  10099_1.txt      0   0   1      0    0    0
  1033_4.txt       0   0   0      0    1    0
  10718_4.txt      0   0   0      0    0    0
  11182_3.txt      0   0   0      0    0    0
  11861_4.txt      1   0   0      0    0    0
  3014_4.txt       0   1   0      0    1    2
  315_1.txt        0   0   0      0    0    0
  6482_2.txt       0   0   0      0    1    0
  9577_1.txt       0   0   0      0    0    0
  9674_3.txt       0   0   0      0    0    0
```

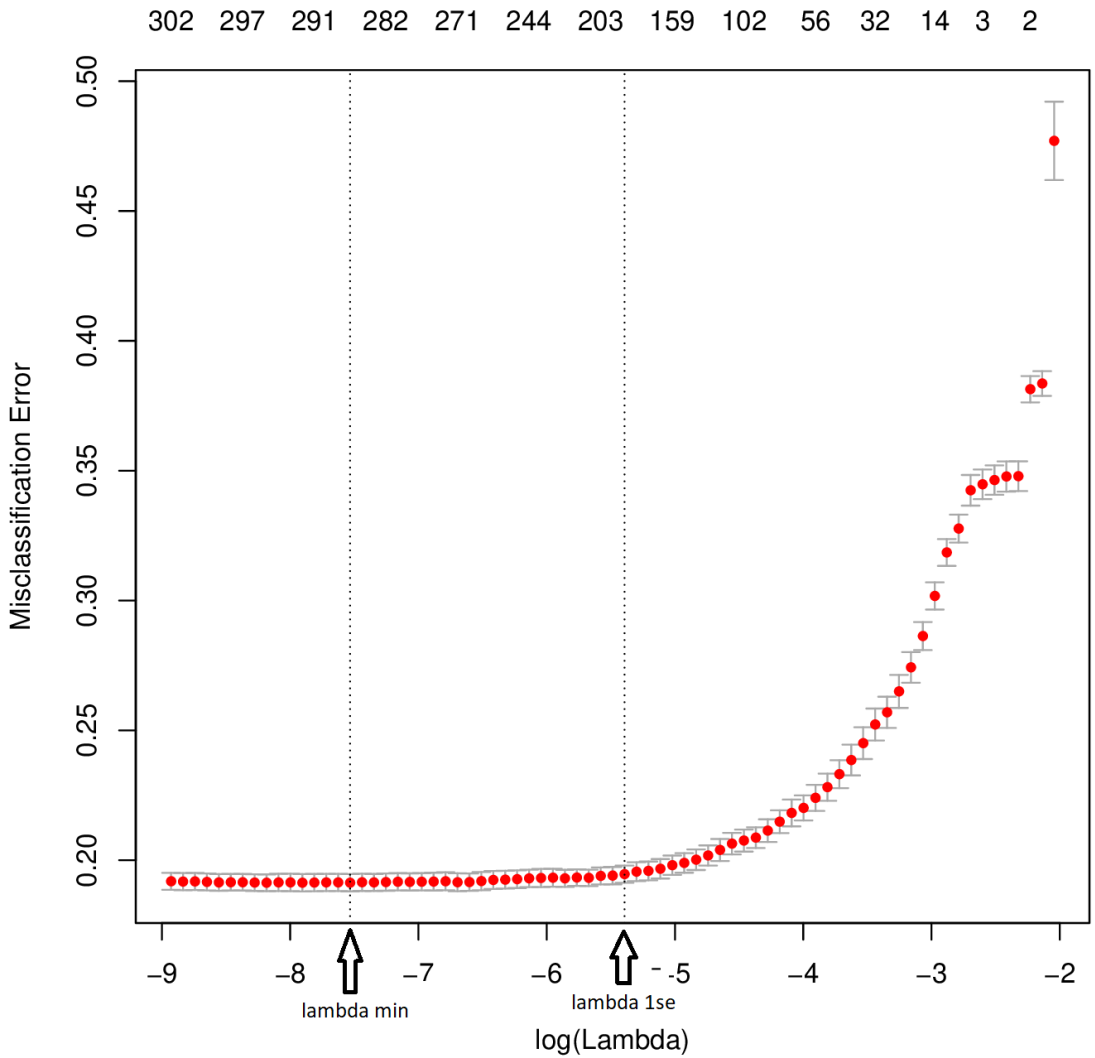
Application of Logistic Regression to Movie Reviews

```
# logistic regression with lasso penalty

> reviews.glmnet <- cv.glmnet(as.matrix(train.dtm),labels[index.train],
                      family="binomial",type.measure="class")
> plot(reviews.glmnet)

> coef(reviews.glmnet,s="lambda.1se")
309 x 1 sparse Matrix of class "dgCMatrix"
                         1
bad           -0.613843496
beautiful      0.378249156
best           0.400765691
better        -0.193594713
boring        -0.904918921
excellent      0.874061528
fun            0.390055537
funny          .
minutes       -0.381871597
perfect        0.757174138
poor          -0.726663951
script        -0.461754268
stupid        -0.555516834
supposed      -0.611473721
terrible      -0.830472064
wonderful      0.697696588
worst         -1.431738320
```

Cross-Validation on lambda



Application of Logistic Regression to Movie Reviews

```
# make predictions on the test set
> reviews.logreg.pred <- predict(reviews.glmnet,
   newx=as.matrix(test.dtm),s="lambda.1se",type="class")
# show confusion matrix
> table(reviews.logreg.pred,labels[-index.train])

reviews.logreg.pred    0    1
                  0 3468  704
                  1 1032 3796
# compute accuracy: about 81% correct
> (3468+3796)/9000
[1] 0.8071111
```

Including Bigrams

The bigrams in

```
the spy who loved me
```

are:

```
the spy
spy who
who loved
loved me
```

but not for example

```
spy loved
```

The two words need to be next to each other.

Including Bigrams

```
# extract both unigrams and bigrams
> train.dtm2 <- DocumentTermMatrix(reviews.all[index.train],
    control = list(tokenize = UniBiTokenizer))
# more than one million uni+bigrams!
> dim(train.dtm2)
[1]   16000 1346555
# remove terms that occur in less than 1% of documents
> train.dtm2 <- removeSparseTerms(train.dtm2,0.99)
# after removing sparse terms only 1,753 left
> dim(train.dtm2)
[1] 16000  1753
```

Code for UniBiTokenizer:

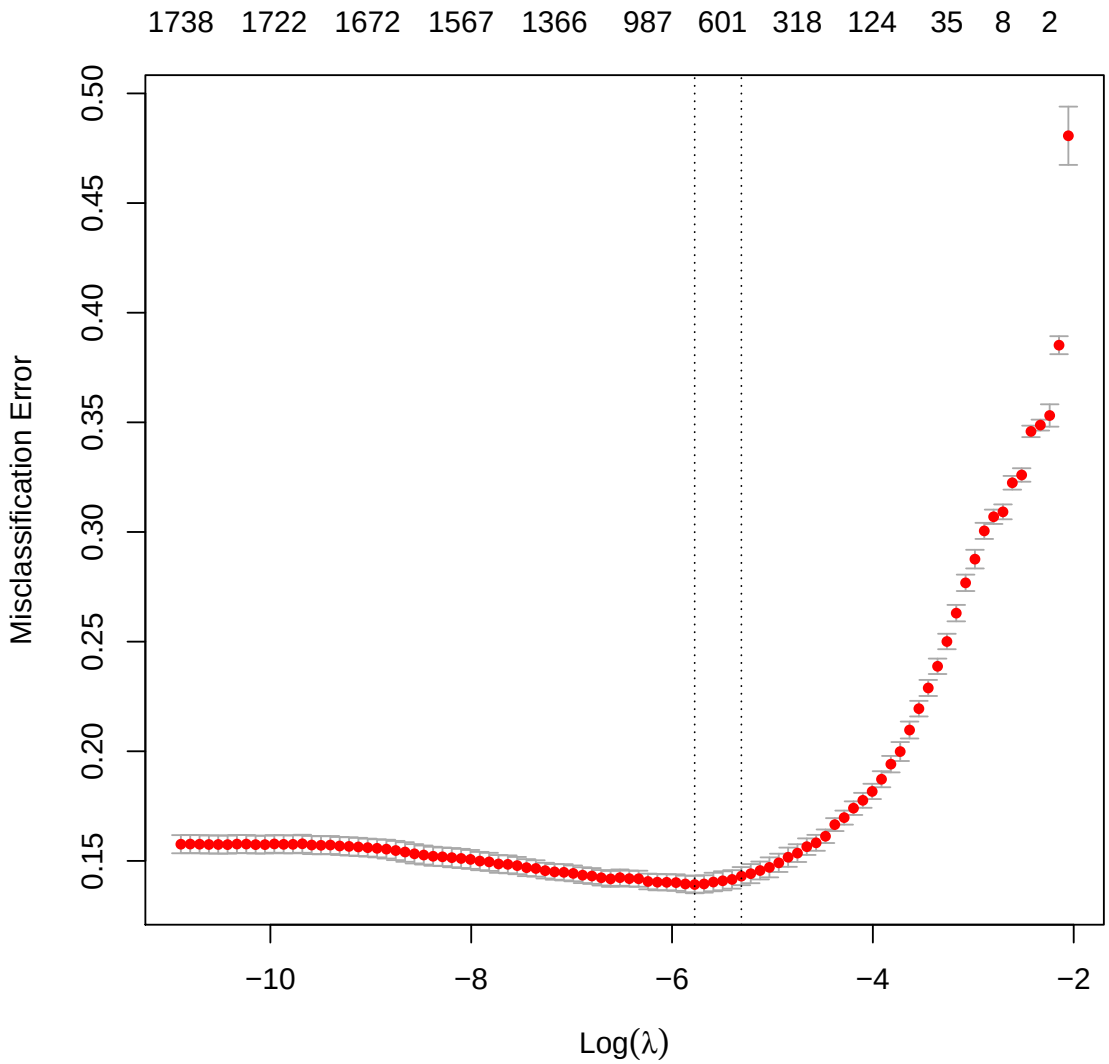
```
UniBiTokenizer <-
function (x) {
unlist(lapply(ngrams(words(x), 1:2), paste, collapse = " "),
       use.names = FALSE)
}
```

Including Bigrams

```
# fit regularized logistic regression model
# use cross-validation to evaluate different lambda values
> reviews.glmnet2 <- cv.glmnet(as.matrix(train.dtm2),labels[index.train],
    family="binomial",type.measure="class")

# show coefficient estimates for lambda-1se
# (only a selection of the bigram coefficients is shown here)
> coef(reviews.glmnet2,s="lambda.1se")
bad movie              -9.580669e-02
cant believe           -1.280761e-01
character development   .
great film              .
great movie             2.145233e-01
highly recommend        5.419558e-01
main character         -1.065261e-01
make sense             -1.737418e-01
one worst              -4.623925e-01
special effects         .
supporting cast         .
waste time             -6.520532e-02
well done               2.860367e-01
whole movie            -1.981443e-01
year old               -5.041887e-02
```

Cross-Validation on lambda



Including Bigrams

```
# create document term matrix for the test data,
# using the training dictionary
> test.dtm2 <- DocumentTermMatrix(reviews.all[-index.train],
    control = list(tokenize=UniBiTokenizer,dictionary=Terms(train.dtm2)))

# make predictions using lambda.1se
> reviews.glmnet.pred <- predict(reviews.glmnet2,newx=as.matrix(test.dtm2),
                                 s="lambda.1se",type="class")
# accuracy improved due to including more unigrams and including bigrams!
> table(reviews.glmnet.pred,labels[-index.train])
  reviews.glmnet.pred    0    1
                    0 3751  534
                    1  749 3966
> (3751+3966)/9000
[1] 0.8574444
```

The Second Assignment: Text Classification

Text Classification for the Detection of Opinion Spam.

- We analyze fake and genuine hotel reviews.
- The genuine reviews have been collected from several popular online review communities.
- The fake reviews have been obtained from Mechanical Turk.
- There are 400 reviews in each of the categories: positive truthful, positive deceptive, negative truthful, negative deceptive.
- We will focus on the negative reviews and try to discriminate between truthful and deceptive reviews.
- Hence, the total number of reviews in our data set is 800.

The Second Assignment: Text Classification

Analyse the data with:

1. Multinomial naive Bayes (generative linear classifier),
2. Regularized logistic regression (discriminative linear classifier),
3. Classification trees, (flexible classifier) and
4. Random forests (ensemble of classification trees).

The Second Assignment: Text Classification

- This is a data analysis assignment, not a programming assignment.
- You will need to program a little to perform the experiments.
- You only need to hand in a report of your analysis, no code!
- You are free to use whatever tools you want.
- We can provide support for Python and R.
- The report should describe the analysis you performed in such a way that the reader would be able reproduce it.
- Carefully read the assignment before you start!