Pattern Recognition 2015
Linear Models for Classification

Ad Feelders

Universiteit Utrecht

Classification Problems

We are concerned with the problems of

1. Allocating an object to a class, on the basis of a number of variables that describe the object.
2. Estimating the probability that a particular object belongs to a specific class.

Interconnected, since allocation is usually based on the estimated probabilities.

Examples of Classification Problems

- Churn: is customer going to leave for a competitor?
- SPAM filter: e-mail message is SPAM or not?
- Medical diagnosis: does patient have breast cancer?
- Handwritten digit recognition.

Classification Problems

In this kind of *classification* problem there is a target variable t that assumes values in an unordered discrete set.

An important special case is when there are only two classes, in which case we usually choose $t \in \{0, 1\}$.

The goal of a classification procedure is to predict the target value (class label) given a set of input values $\mathbf{x} = \{x_1, \ldots, x_D\}$ measured on the same object.

Classification Problems

At a particular point $\mathbf{x}$ the value of t is not uniquely determined.

It can assume both its values with respective probabilities that depend on the location of the point $\mathbf{x}$ in the input space. We write

$$p(\mathcal{C}_1|\mathbf{x}) = 1 - p(\mathcal{C}_2|\mathbf{x}) = y(\mathbf{x}).$$

The goal of a classification procedure is to produce an estimate of $y(\mathbf{x})$ at every input point.

Two types of approaches to classification

- Discriminative Models ("regression"; section 4.3).
- Generative Models ("density estimation"; section 4.2).

Discriminative Models

Discriminative methods only model the *conditional* distribution of t given $\mathbf{x}$. The probability distribution of $\mathbf{x}$ itself is not modeled. For the binary classification problem:

$$y(\mathbf{x}) = p(\mathcal{C}_1|\mathbf{x}) = f(\mathbf{x}, \mathbf{w})$$

where $f(\mathbf{x}, \mathbf{w})$ is some deterministic function of $\mathbf{x}$.

Note that the approach to regression we discussed follows the same strategy.

Discriminative Models

Examples of discriminative classification methods:

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

Generative Models

An alternative paradigm for estimating $y(\mathbf{x})$ is based on density estimation. Here Bayes' theorem

$$\begin{aligned} y(\mathbf{x}) &= p(\mathcal{C}_1|\mathbf{x}) \\ &= \frac{p(\mathcal{C}_1)p(\mathbf{x}|\mathcal{C}_1)}{p(\mathcal{C}_1)p(\mathbf{x}|\mathcal{C}_1) + p(\mathcal{C}_2)P(\mathbf{x}|\mathcal{C}_2)} \end{aligned}$$

is applied where $p(\mathbf{x}|\mathcal{C}_k)$ are the class conditional probability density functions and $p(\mathcal{C}_k)$ are the unconditional ("prior") probabilities of each class.

Generative Models

Examples of density estimation based classification methods:

- Linear/Quadratic Discriminant Analysis,
- Naive Bayes classifier,
- ...

Discriminative Models: linear probability model

Consider the linear regression model

$$t = \mathbf{w}^\top \mathbf{x} + \varepsilon \qquad\qquad t \in \{0, 1\}$$

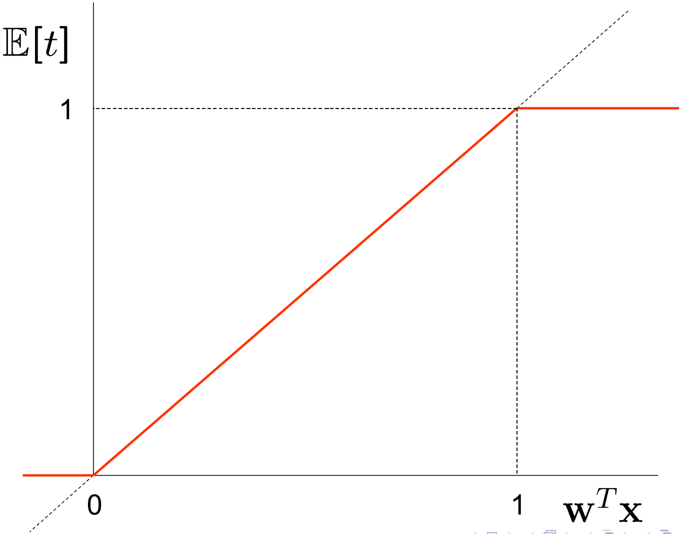
By assumption $\mathbb{E}[\varepsilon|\mathbf{x}] = 0$, so we have

$$\mathbb{E}[t|\mathbf{x}] = \mathbf{w}^\top \mathbf{x}$$

But

$$\begin{aligned} \mathbb{E}[t|\mathbf{x}] &= 1 \cdot p(t = 1|\mathbf{x}) + 0 \cdot p(t = 0|\mathbf{x}) \\ &= p(t = 1|\mathbf{x}) \equiv p(\mathcal{C}_1|\mathbf{x}) \end{aligned}$$

Linear response function



Logistic regression

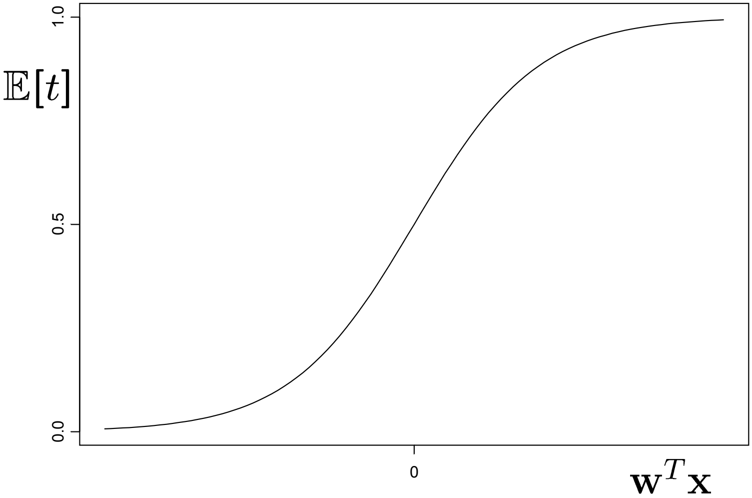
Logistic response function

$$\mathbb{E}[t|\mathbf{x}] = \frac{e^{\mathbf{w}^\top \mathbf{x}}}{1 + e^{\mathbf{w}^\top \mathbf{x}}}$$

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

$$\mathbb{E}[t|\mathbf{x}] = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}} = (1 + e^{-\mathbf{w}^\top \mathbf{x}})^{-1} \qquad (4.59 \text{ and } 4.87)$$

Logistic Response Function



Linearization: the logit transformation

$$
\begin{aligned}
\ln \frac{p(\mathcal{C}_1|\mathbf{x})}{1-p(\mathcal{C}_1|\mathbf{x})} &= \ln \frac{(1+e^{-\mathbf{w}^\top\mathbf{x}})^{-1}}{1-(1+e^{-\mathbf{w}^\top\mathbf{x}})^{-1}} \\
&= \ln \frac{1}{(1+e^{-\mathbf{w}^\top\mathbf{x}})-1} = \ln \frac{1}{e^{-\mathbf{w}^\top\mathbf{x}}} \\
&= \ln e^{\mathbf{w}^\top\mathbf{x}} = \mathbf{w}^\top\mathbf{x}
\end{aligned}
$$

In the second step, we divided the numerator and the denominator by $(1+e^{-\mathbf{w}^\top\mathbf{x}})^{-1}$. The ratio $p(\mathcal{C}_1|\mathbf{x})/(1-p(\mathcal{C}_1|\mathbf{x}))$ is called the *odds*.

Linear Separation

Assign to class $\mathcal{C}_1$ if $p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x})$, i.e. if

$$\frac{p(\mathcal{C}_1|\mathbf{x})}{1 - p(\mathcal{C}_1|\mathbf{x})} > 1$$

This is true if

$$\ln\left\{\frac{p(\mathcal{C}_1|\mathbf{x})}{1 - p(\mathcal{C}_1|\mathbf{x})}\right\} > 0$$

So assign to class $\mathcal{C}_1$ if $\mathbf{w}^\top\mathbf{x} > 0$, and to class $\mathcal{C}_2$ otherwise.

Maximum Likelihood Estimation

$t = 1$ if heads, $t = 0$ if tails. $\mu = p(t = 1)$.
One coin flip

$$p(t) = \mu^t (1 - \mu)^{1-t}$$

Note that $p(1) = \mu$, $p(0) = 1 - \mu$ as required.
Sequence of N independent coin flips

$$p(\mathbf{t}) = p(t_1, t_2, ..., t_N) = \prod_{n=1}^{N} \mu^{t_n} (1 - \mu)^{1-t_n}$$

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

Maximum Likelihood Estimation

In a sequence of 10 coin flips we observe $\mathbf{t} = (1, 0, 1, 1, 0, 1, 1, 1, 1, 0)$.

The corresponding likelihood function is

$$\begin{aligned} p(\mathbf{t}|\mu) &= \mu \cdot (1-\mu) \cdot \mu \cdot \mu \cdot (1-\mu) \cdot \mu \cdot \mu \cdot \mu \cdot \mu \\ &\quad \cdot (1-\mu) = \mu^7 (1-\mu)^3 \end{aligned}$$

The corresponding loglikelihood function is

$$\ln p(\mathbf{t}|\mu) = \ln(\mu^7 (1-\mu)^3) = 7 \ln \mu + 3 \ln(1-\mu)$$

Computing the maximum

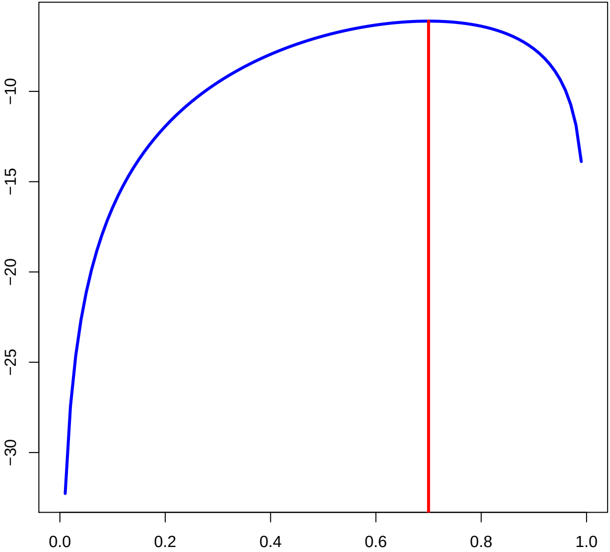
To determine the maximum we take the derivative and equate it to zero

$$\frac{d \ln p(\mathbf{t}|\mu)}{d\mu} = \frac{7}{\mu} - \frac{3}{1-\mu} = 0$$

which yields maximum likelihood estimate $\mu_{\text{ML}} = 0.7$.

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

Loglikelihood function for $\mathbf{t} = (1, 0, 1, 1, 0, 1, 1, 1, 1, 0)$



ML estimation for logistic regression

Now probability of success depends on $\mathbf{x}_n$:

$$\begin{aligned} y_n &= p(\mathcal{C}_1|\mathbf{x}_n) = (1 + e^{-\mathbf{w}^\top \mathbf{x}_n})^{-1} \\ 1 - y_n &= p(\mathcal{C}_2|\mathbf{x}_n) = (1 + e^{\mathbf{w}^\top \mathbf{x}_n})^{-1} \end{aligned}$$

we can represent its probability distribution as follows

$$p(t_n) = y_n^{t_n}(1 - y_n)^{1-t_n} \qquad t_n \in \{0, 1\}; \quad n = 1, \ldots, N$$

Example

n	x_n	t_n	$p(t_n)$
1	8	0	$(1 + e^{w_0 + 8w_1})^{-1}$
2	12	0	$(1 + e^{w_0 + 12w_1})^{-1}$
3	15	1	$(1 + e^{-w_0 - 15w_1})^{-1}$
4	10	1	$(1 + e^{-w_0 - 10w_1})^{-1}$

LR: likelihood function

Since the t_n observations are independent:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} p(t_n) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \tag{4.89}$$

Or, taking minus the natural log:

$$\begin{aligned} -\ln p(\mathbf{t}|\mathbf{w}) &= -\ln \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n} \\ &= -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \end{aligned} \tag{4.90}$$

This is called the *cross-entropy* error function.

LR: error function

Since for the logistic regression model

$$\begin{aligned} y_n &= (1 + e^{-\mathbf{w}^\top \mathbf{x}_n})^{-1} \\ 1 - y_n &= (1 + e^{\mathbf{w}^\top \mathbf{x}_n})^{-1} \end{aligned}$$

we get

$$E(\mathbf{w}) = \sum_{n=1}^{N} \left\{ t_n \ln(1 + e^{-\mathbf{w}^\top \mathbf{x}_n}) + (1 - t_n) \ln(1 + e^{\mathbf{w}^\top \mathbf{x}_n}) \right\}$$

- Non-linear function of the parameters.
- No closed form solution.
- Likelihood function globally concave.
- Estimate with e.g. iterative re-weighted least squares (section 4.3.3).

Fitted Response Function

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

$$\hat{p}(\mathcal{C}_1|\mathbf{x}) = \frac{e^{\mathbf{w}_{\mathrm{ML}}^\top \mathbf{x}}}{1 + e^{\mathbf{w}_{\mathrm{ML}}^\top \mathbf{x}}}$$

Example: Programming Assignment

Model the probability of successfully completing a programming assignment.

Explanatory variable: "programming experience".
We find $w_0 = -3.0597$ and $w_1 = 0.1615$, so

$$\hat{p}(\mathcal{C}_1|x_n) = \frac{e^{-3.0597+0.1615x_n}}{1 + e^{-3.0597+0.1615x_n}}$$

14 months of programming experience:

$$\hat{p}(\mathcal{C}_1|x = 14) = \frac{e^{-3.0597+0.1615(14)}}{1 + e^{-3.0597+0.1615(14)}} \approx 0.31$$

Example: Programming Assignment

	month.exp	success	fitted		month.exp	success	fitted
1	14	0	0.310262	16	13	0	0.276802
2	29	0	0.835263	17	9	0	0.167100
3	6	0	0.109996	18	32	1	0.891664
4	25	1	0.726602	19	24	0	0.693379
5	18	1	0.461837	20	13	1	0.276802
6	4	0	0.082130	21	19	0	0.502134
7	18	0	0.461837	22	4	0	0.082130
8	12	0	0.245666	23	28	1	0.811825
9	22	1	0.620812	24	22	1	0.620812
10	6	0	0.109996	25	8	1	0.145815
11	30	1	0.856299				
12	11	0	0.216980				
13	30	1	0.856299				
14	5	0	0.095154				
15	20	1	0.542404				

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:

$x \geq 19$: *assign to class* $\mathcal{C}_1$ *(*$t = 1$*)*
$x < 19$: *assign to class* $\mathcal{C}_2$ *(*$t = 0$*)*

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(succes ~ 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$succes, as.numeric(prog.logreg$fitted > 0.5))


     0  1
  0 11  3
  1  3  8
```

Example: Conn's syndrome

Two possible causes:

(a) Benign tumor (adenoma) of the adrenal cortex.

(b) More diffuse affection of the adrenal glands (bilateral hyperplasia).

Pre-operative diagnosis on basis of

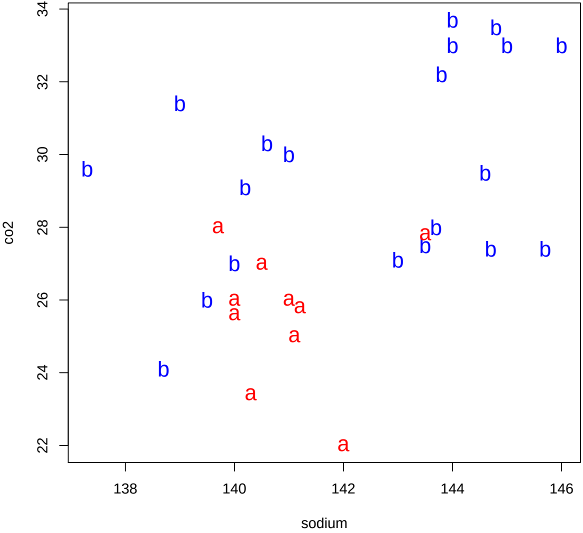
1. Sodium concentration (mmol/l)
2. CO_2 concentration (mmol/l)

Conn's syndrome: the data

a=1, b=0

	sodium	co2	cause		sodium	co2	cause
1	140.6	30.3	0	16	139.0	31.4	0
2	143.0	27.1	0	17	144.8	33.5	0
3	140.0	27.0	0	18	145.7	27.4	0
4	146.0	33.0	0	19	144.0	33.0	0
5	138.7	24.1	0	20	143.5	27.5	0
6	143.7	28.0	0	21	140.3	23.4	1
7	137.3	29.6	0	22	141.2	25.8	1
8	141.0	30.0	0	23	142.0	22.0	1
9	143.8	32.2	0	24	143.5	27.8	1
10	144.6	29.5	0	25	139.7	28.0	1
11	139.5	26.0	0	26	141.1	25.0	1
12	144.0	33.7	0	27	141.0	26.0	1
13	145.0	33.0	0	28	140.5	27.0	1
14	140.2	29.1	0	29	140.0	26.0	1
15	144.7	27.4	0	30	140.0	25.6	1

Conn's Syndrome: Plot of Data



Maximum Likelihood Estimation

The maximum likelihood estimates are:

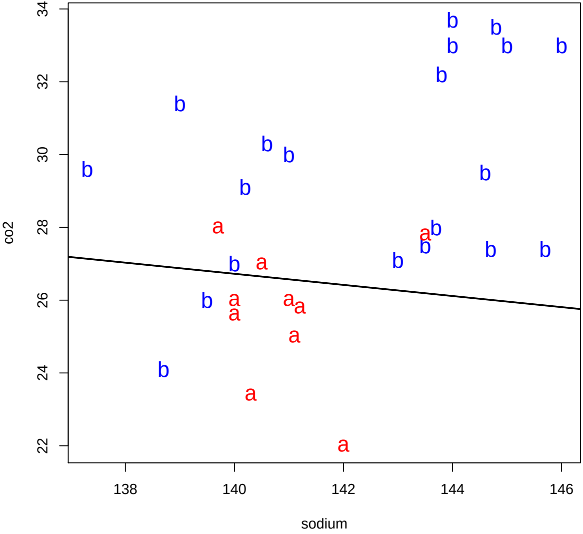
$$
\begin{aligned}
w_0 &= 36.6874320 \\
w_1 &= -0.1164658 \\
w_2 &= -0.7626711
\end{aligned}
$$

Assign to group a if

$$36.69 - 0.12 \times \text{sodium} - 0.76 \times \text{CO}_2 > 0$$

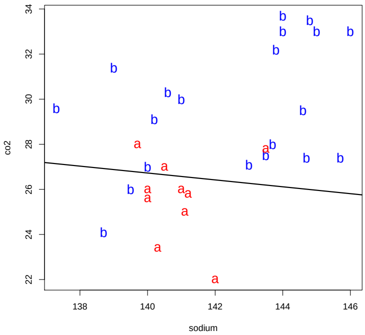
and to group b otherwise.

Conn's Syndrome: Allocation Rule



How to in R

```
# plot data points

> plot(conn.dat[,1],conn.dat[,2], pch=c(rep("b",20), rep("a",10)),
col=c(rep(4,20), rep(2,10)), cex=1.5, xlab="sodium", ylab="co2")

# draw decision boundary

> abline(36.6874320/0.7626711,-0.1164658/0.7626711,lwd=2)
```



Conn's Syndrome: Confusion Matrix

Cross table of observed and predicted class label:

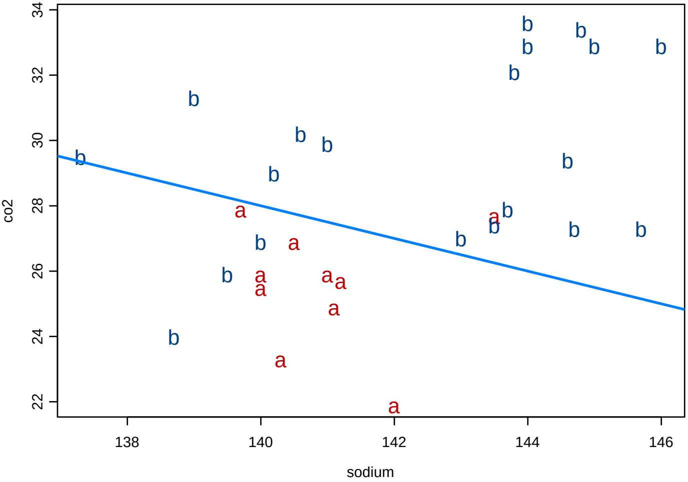
	a	b
a	7	3
b	2	18

Row: observed, Column: predicted

Error rate: 5/30=1/6

Default: 1/3

Conn's Syndrome: Line with lower empirical error



Likelihood and Error Rate

Likelihood maximization is not the same as error rate minimization!

n	t_n	$\hat{p}_1(t_n = 1)$	$\hat{p}_2(t_n = 1)$
1	0	0.9	0.6
2	0	0.4	0.1
3	1	0.6	0.9
4	1	0.55	0.4

Which model has the lower error-rate?
Which one the higher likelihood?

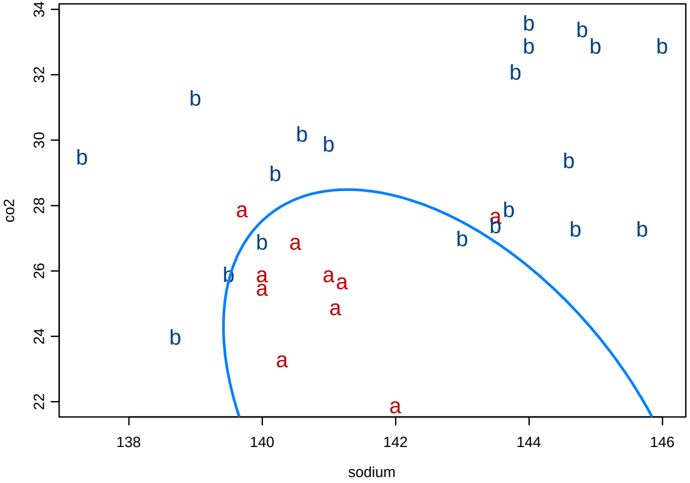
Quadratic Model

Coefficient	Value
(Intercept)	-13100.69
sodium	177.42
CO_2	41.36
sodium^2	-0.60
CO_2^2	-0.12
sodium $\times$ CO_2	-0.25

Cross table of observed (row) and predicted class label:

	a	b
a	8	2
b	2	18

Conn's Syndrome: Quadratic Specification



Non-binary classes in logistic regression

Recall the logistic regression model assumption for binary class variable $t \in \{0, 1\}$:

$$p(t = 1|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \mathbf{x})}{1 + \exp(\mathbf{w}^\top \mathbf{x})}$$

from which it follows that

$$p(t = 0|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x})}$$

since

$$p(t = 1|\mathbf{x}) + p(t = 0|\mathbf{x}) = 1$$

Non-binary classes in logistic regression

We can generalize this model to non-binary class variable $t \in \{0, 1, \ldots, K-1\}$ (where K is the number of classes) as follows

$$p(t = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=0}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})} \qquad (4.104 \text{ and } 4.105)$$

where we now have a weight vector $\mathbf{w}_k$ for each class.

This is called the *multinomial logit model* or multi-class logistic regression (section 4.3.4).

Multi-class logistic regression

We can arrive at this model in the following steps:

1. Assume that $p(t = k|\mathbf{x})$ is a function of the linear combination $\mathbf{w}_k^\top \mathbf{x}$.
2. To ensure that the probabilities are non-negative, take the exponential $\exp(\mathbf{w}_k^\top \mathbf{x})$.
3. To make sure the probabilities sum to 1, we divide $\exp(\mathbf{w}_k^\top \mathbf{x})$ by $\sum_{j=0}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})$:

$$p(t = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=0}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

Identification Restriction

Note that

$$\frac{\exp((\mathbf{w}_k + \mathbf{d})^\top \mathbf{x})}{\sum_{j=0}^{K-1} \exp((\mathbf{w}_j + \mathbf{d})^\top \mathbf{x})} = \frac{\exp(\mathbf{d}^\top \mathbf{x})}{\exp(\mathbf{d}^\top \mathbf{x})} \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=0}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

so adding a vector $\mathbf{d}$ to each of the vectors $\mathbf{w}_j, j = 0, \ldots, K-1$ would yield the same fitted probabilities.

To identify the model, we put $\mathbf{w}_0 = \mathbf{0}$.

Verify that binary logistic regression is a special case of the multinomial logit model, with $K = 2$, since $\exp(\mathbf{w}_0^\top \mathbf{x}) = \exp(0) = 1$.

Interpretation

We have

$$\ln\left\{\frac{p(t=k|\mathbf{x})}{p(t=\ell|\mathbf{x})}\right\} = (\mathbf{w}_k - \mathbf{w}_\ell)^\top \mathbf{x},$$

which allows us to interpret $w_{k,i} - w_{\ell,i}$ as follows:

> *for a unit increase in x_i, the log-odds of class k versus class ℓ is expected to change by $w_{k,i} - w_{\ell,i}$ units, holding all the other variables constant.*

Since $\mathbf{w}_0 = \mathbf{0}$, $w_{k,i}$ is the effect of x_i on the log-odds of class k relative to class 0:

> *for a unit increase in x_i, the log-odds of class k versus class 0 is expected to change by $w_{k,i}$ units, holding all the other variables constant.*

Multinomial Logit in R

```
# load training data

> optdigits.train <- read.csv("D:/Pattern
Recognition/Datasets/optdigits-tra.txt", header=F)

# convert class label to factor

> optdigits.train[,65] <- as.factor(optdigits.train[,65])

# same for test data

> optdigits.test <- read.csv("D:/Pattern
Recognition/Datasets/optdigits-tes.txt", header=F)
> optdigits.test[,65] <- as.factor(optdigits.test[,65])
```

Multinomial Logit in R

```
# load nnet library

> library(nnet)

# fit multinomial logistic regression model
# column 1 and 40 are not used (always 0)

> optdigits.multinom <- multinom(V65 ~ ., data =
optdigits.train[,-c(1,40)], maxit = 1000)

# weights:  640 (567 variable)
initial  value 8802.782811
...
converged

# predict class label on training data

> optdigits.multinom.pred <- predict(optdigits.multinom,
optdigits.train[,-c(1,40,65)],type="class")
```

Multinomial Logit in R

```
# make confusion matrix: true label vs. predicted label

> table(optdigits.train[,65],optdigits.multinom.pred)

   optdigits.multinom.pred
      0   1   2   3   4   5   6   7   8   9
  0 376   0   0   0   0   0   0   0   0   0
  1   0 389   0   0   0   0   0   0   0   0
  2   0   0 380   0   0   0   0   0   0   0
  3   0   0   0 389   0   0   0   0   0   0
  4   0   0   0   0 387   0   0   0   0   0
  5   0   0   0   0   0 376   0   0   0   0
  6   0   0   0   0   0   0 377   0   0   0
  7   0   0   0   0   0   0   0 387   0   0
  8   0   0   0   0   0   0   0   0 380   0
  9   0   0   0   0   0   0   0   0   0 382
```

Multinomial Logit in R

```
# predict class label on test data

> optdigits.multinom.test.pred <- predict(optdigits.multinom,
optdigits.test[,-c(1,40,65)],type="class")
> table(optdigits.test[,65],optdigits.multinom.test.pred)

   optdigits.multinom.test.pred
      0   1   2   3   4   5   6   7   8   9
  0 170   1   0   0   1   6   0   0   0   0
  1   1 170   0   0   4   1   3   1   1   1
  2   4   7 157   1   0   0   6   1   1   0
  3   0   0  10 155   0   2   2   8   3   3
  4   0   8   0   0 153   1   9   3   1   6
  5   0   0   1   5   1 173   0   1   0   1
  6   4   2   0   0   4   3 168   0   0   0
  7   0   0   4   0   2  17   2 149   0   5
  8   2   5   0   7   3   5   5   4 142   1
  9   1   6   0   0   2   5   0   4   3 159
```

Multinomial Logit in R

```
# make confusion matrix for predictions on test data

> confmat <- table(optdigits.test[,65],
optdigits.multinom.test.pred)

# use it to compute accuracy on test data

> sum(diag(confmat))/sum(confmat)

[1] 0.888147
```

The accuracy on the test sample is about 89%.

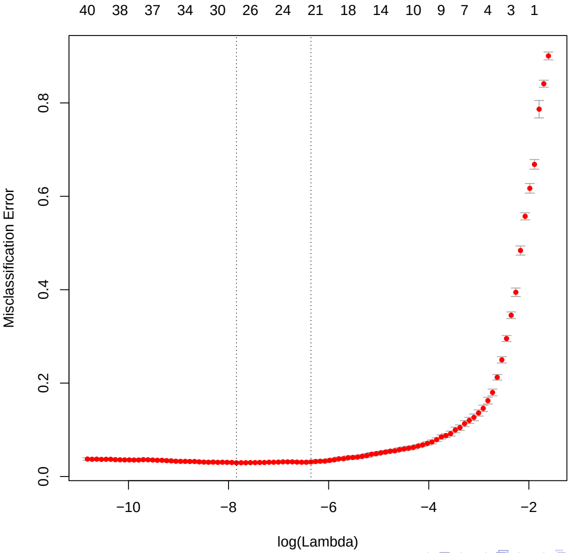
Multinomial Logit with LASSO

With ordinary multinomial logit the accuracy on the training set was 100%, but on the test set only 89%. Maybe we are overfitting. Apply regularization (LASSO).

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \sum_{i=1}^{M-1} |w_i|$$

```
# load glmnet library

> library(glmnet)

# apply 10-fold cross-validation with different values of lambda

> optdigits.lasso.cv <-
cv.glmnet(as.matrix(optdigits.train[,-c(1,40,65)]),
optdigits.train[,65],family="multinomial", type.measure="class")

# plot lambda agains misclassification error

> plot(optdigits.lasso.cv)
```

Plot of lambda against misclassification error



Results of Cross-Validation

- Best value is $\lambda \approx 0.0004$.
- The cross validation misclassification error is about 3% for this value of λ.
- On average, only 27 out of 62 coefficients are non-zero. Sparse solution.

Prediction on Test Set

```
# predict class label on test set using the best cv model

> optdigits.lasso.cv.pred <- predict(optdigits.lasso.cv,
as.matrix(optdigits.test[,-c(1,40,65)]),type="class")

# make the confusion matrix

> optdigits.lasso.cv.confmat <-
table(optdigits.test[,65],optdigits.lasso.cv.pred)

# compute the accuracy on the test set

> sum(diag(optdigits.lasso.cv.confmat))/
sum(optdigits.lasso.cv.confmat)

[1] 0.9510295
```

We have improved the accuracy from 89% to 95% by using regularization.