Regression, classification and regularization

Lecture 1d

Overview

- Linear regression
- Logistic regression
- Basis function expansion
- Ridge regression

Simple linear regression

Given:

Data $\mathrm{T} = \{(x_i, y_i), i = 1, \dots n\}$

Model:

$$y = \theta_0 + \theta_1 x + \epsilon,$$
$$\epsilon \sim N(0, \sigma^2)$$

or

$$y|x \sim N(\theta_0 + \theta_1 x, \sigma^2)$$

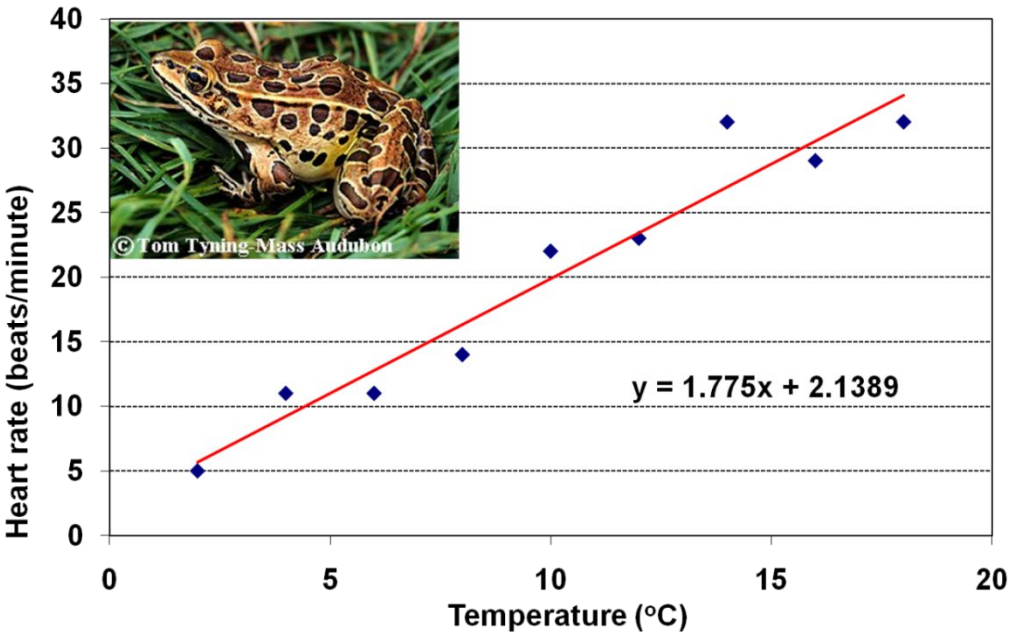
or

$$p(y|x, \boldsymbol{\theta}) = N(\theta_0 + \theta_1 x, \sigma^2)$$

Terminology:

θ_0 : **intercept (or bias)**

θ_1 **regression coefficient**



Linear regression

Model:

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p + \epsilon$$

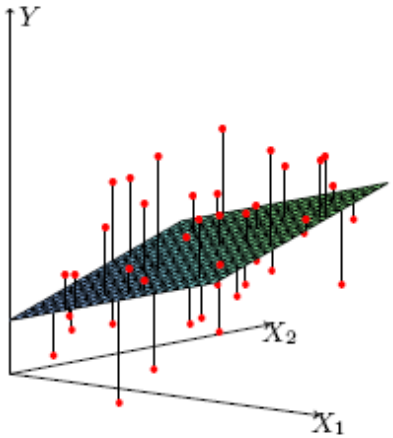
$$y|\boldsymbol{x} \sim N\left(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}, \sigma^2\right)$$

where

$$\boldsymbol{\theta} = \{\theta_0, \dots \theta_p\}$$

$$\boldsymbol{x} = \{1, x_1, \dots x_p\}$$

Why is ”1” here?



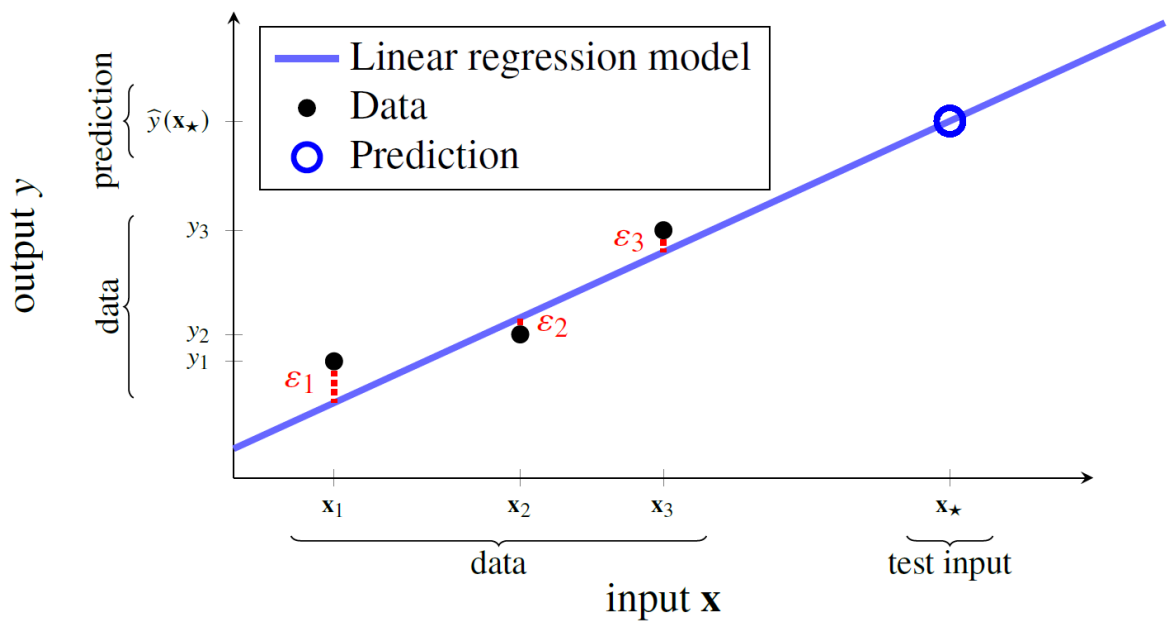
Aim: learn optimal parameters
$\hat{\theta}_0, \dots, \hat{\theta}_p$

Linear regression

- Prediction with linear regression for x_*

$$\hat{y}(x_*, \hat{\boldsymbol{\theta}}) = \hat{\theta}_0 + \hat{\theta}_1 x_{*1} + \cdots \hat{\theta}_p x_{*p} = \hat{\boldsymbol{\theta}} \boldsymbol{x}_*$$

- Irreducible error ϵ



Linear model: learning

- How to learn parameters?
  - **Approach A**: Minimize the cost

$$\hat{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}(x_i, \boldsymbol{\theta}), y_i)$$

Loss function

Cost function

Usual choice: squared loss $L(\hat{y}(x_i, \boldsymbol{\theta}), y_i) = (\hat{y}(x_i, \theta) - y_i)^2$

Linear model: learning

Ordinary least squares

Objective: $\min_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i\boldsymbol{\theta})^2 = \min_{\boldsymbol{\theta}} (y - \boldsymbol{X\theta})^T(y - \boldsymbol{X\theta})$

Optimality condition:

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X\theta}) = 0$$

where

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & & x_{1p} \\ 1 & x_{21} & x_{22} & & x_{2p} \\ & & & & \\ & & & & \\ 1 & x_{n1} & x_{n2} & & x_{np} \end{pmatrix}$$

and

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \\ \\ y_n \end{pmatrix}$$

Linear model: learning

- Least squares estimates of the parameters

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Predicted values $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\theta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$

Hat matrix

Why is it called so?

- Linear regression belongs to the class of **linear smoothers**

- **Note:** if $p > n$ then $\boldsymbol{X}^T\boldsymbol{X}$ is not invertible

Linear regression: learning algorithm

- Training
  1. Construct $\boldsymbol{X}$ and $\boldsymbol{y}$
  2. Compute $\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$

- Prediction
  - $\hat{y}(\boldsymbol{x}_*) = \widehat{\boldsymbol{\theta}}^T\boldsymbol{x}_*$

Linear model learning

- How to learn parameters?
  - **Approach B**: Maximize the likelihood

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\theta} p(\mathrm{T}|\boldsymbol{\theta})$$

Exercise: Approach A and B are equivalent

Linear regression in R

- fit=lm(formula, data, subset, weights,…)
  - **data** is the data frame containing the predictors and response values
  - **formula** is expression for the model
  - **subset** which observations to use (training data)?
  - **weights** should weights be used?

fit is object of class **lm** containing various regression results.

- Useful functions (many are generic, used in many other models)
  - Get details about the particular function by ".", for ex. predict.lm

```
summary(fit)
predict(fit, newdata, se.fit, interval)
coefficients(fit) # model coefficients
confint(fit, level=0.95) # CIs for model parameters
fitted(fit) # predicted values
residuals(fit) # residuals
```

An example of ordinary least squares regression

```
mydata=read.csv2("Bilexempel.csv")
fit1=lm(Price~Year, data=mydata)
summary(fit1)
fit2=lm(Price~Year+Mileage+Equipment, data=mydata)
summary(fit2)
```

Response variable:

Requested price of used Porsche cars (1000 SEK)

```
> summary(fit1)

Call:
lm(formula = Price ~ Year, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-167683  -14683   20056   35933   72317 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -78161027    8448038  -9.252 6.00e-13 ***
Year            39246       4226   9.288 5.25e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57270 on 57 degrees of freedom
Multiple R-squared:  0.6021,	Adjusted R-squared:  0.5952 
F-statistic: 86.26 on 1 and 57 DF,  p-value: 5.248e-13
```

Inputs:

X_1 = Manufacturing year

X_2 = Milage (km)

X_4 = Equipment (0 or 1)

An example of ordinary least squares regression

```
> summary(fit2)

Call:
lm(formula = Price ~ Year + Mileage + Equipment, data = mydata)

Residuals:
   Min     1Q Median     3Q    Max 
-66223 -10525   -739  14128  65332 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.083e+07  6.309e+06  -3.302  0.00169 ** 
Year         1.062e+04  3.154e+03   3.366  0.00139 ** 
Mileage     -2.077e+00  2.022e-01 -10.269 2.14e-14 ***
Equipment    5.790e+04  1.041e+04   5.563 8.08e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 29270 on 55 degrees of freedom
Multiple R-squared:  0.8997, Adjusted R-squared:  0.8942 
F-statistic: 164.5 on 3 and 55 DF,  p-value: < 2.2e-16
```

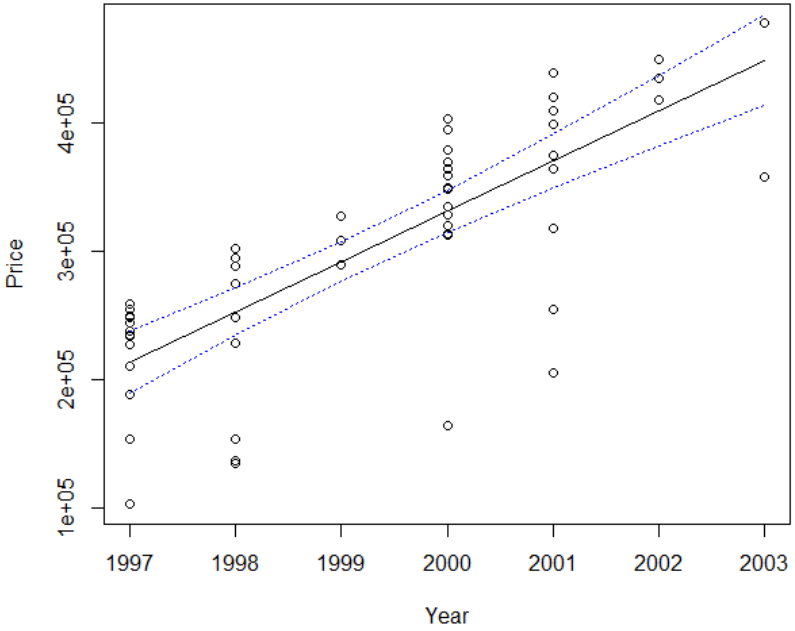
An example of ordinary least squares regression

- Prediction

```
fitted <- predict(fit1, interval =
"confidence")

# plot the data and the fitted line
attach(mydata)
plot(Year, Price)
lines(Year, fitted[, "fit"])

# plot the confidence bands
lines(Year, fitted[, "lwr"], lty = "dotted",
col="blue")
lines(Year, fitted[, "upr"], lty = "dotted",
col="blue")
detach(mydata)
```



Data scaling

- In linear regression not necessary but can improve interpretation when comparing coefficient values

- **Same transformation** on train, valid and test data
  - Never scale these separately
  - Normally done on training data as required by many ML methods

- In R: library **caret**
  - *preProcess()* to learn transformation
  - *predict() to apply transformation*

Data scaling

```
set.seed(12345)
n=nrow(data)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

library(caret)
scaler=preProcess(train)
trainS=predict(scaler,train)
testS=predict(scaler,test)

fit3=lm(Price~Year+Mileage+Equipment,
data=trainS)
summary(fit3)
```

```
> summary(fit3)

Call:
lm(formula = Price ~ Year + Mileage + Equipment, data = trainS)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.89677 -0.21339 -0.01089  0.18955  0.49303 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.160e-14  6.814e-02   0.000  1.00000    
Year         3.362e-01  1.038e-01   3.239  0.00337 ** 
Mileage     -5.594e-01  8.603e-02  -6.502 8.26e-07 ***
Equipment    1.951e-01  9.742e-02   2.003  0.05616 .  
```

Degrees of freedom

Definition:

$$df(\hat{y}) = \frac{1}{\sigma^2}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i)$$

- Larger covariance → stronger connection → model can approximate data better→ model more flexible (complex)
- For linear smoothers $\hat{Y} = S(X)Y$

$$df = trace(S)$$

- For linear regression, degrees of freedom is

$$df = trace(P) = p$$

Different types of features

- Interval variables
- Numerically coded ordinal variables
  - (small=1, medium=2, large=3)
- Dummy coded qualitative variables

Basis function expansion:
If $y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 e^{-x_2} + \epsilon$,

Model becomes linear if to recompute:

$$\phi_1(x_1) = x_1$$
$$\phi_2(x_1) = x_1^2$$
$$\phi_3(x_1) = e^{-x_2}$$

Example of dummy coding:

$$x_1 = \begin{cases} 1, \text{if Jan} \\ 0, \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1, \text{if Feb} \\ 0, \text{otherwise} \end{cases}$$

.

.

.

$$x_{11} = \begin{cases} 1, \text{if Nov} \\ 0, \text{otherwise} \end{cases}$$

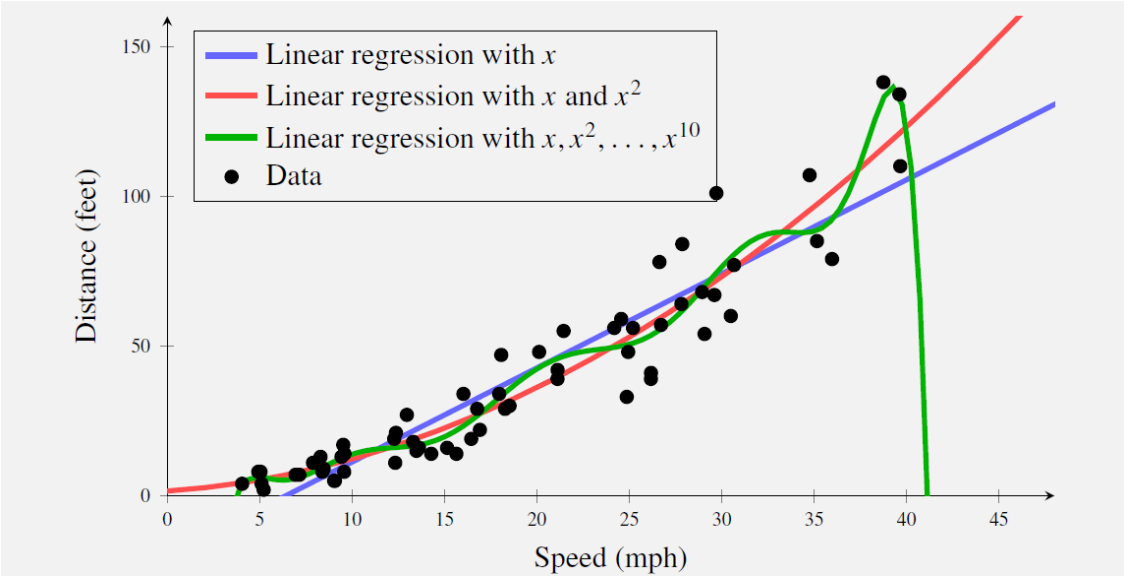
Basis function expansion

- In general $\phi_1(\dots)$ may be a function of several x components
- Having data given by **X,** compute new data with q features
- $\Phi = \begin{pmatrix} 1 & \phi_1(x_{11,} \dots, x_{1p}) & .. & \phi_q(x_{11,} \dots, x_{1p}) \\ & \dots & \dots & \dots \\ 1 & \phi_1(x_{n1,} \dots, x_{np}) & \dots & \phi_q(x_{n1,} \dots, x_{np}) \end{pmatrix}$
- If doing a basis function in a model, replace **X** by Φ everywhere where **X** is used:

$$\hat{\boldsymbol{y}} = \Phi(\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{y}$$

Polynomial regression

- Do by basis function trick:
  - $\phi_1(x_1) = x_1, \ldots, \phi_2(x_1) = x_1^M$



High degree of polynomial leads to overfitting.

Regularization

- Used in a huge variety of models, for ex deep learning

- **Problem of overfitting**: models fit training data perfectly but are too complex→ penalize complexity!
  - The more coefficients close to zero, the less complex the model is

- **L2 regularization** in linear (polynomial) regression = **Ridge regression**
  - Training data **are normally** scaled (one λ for all features!)

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{y}_i - \theta^T \boldsymbol{x}_i)^2 + \lambda \sum_{j=1}^{p} \theta_j^2 , \qquad \lambda > 0$$

Regularization

Equivalent form

$$\hat{\theta}^{ridge} = \text{argmin} \sum_{i=1}^{N} (y_i - \theta_0 - \theta_1 x_{1j} - \ldots - \theta_p x_{pj})^2$$

$$\text{subject to} \sum_{j=1}^{p} \theta_j^2 \leq s, \qquad s > 0$$

Solution

$$\boxed{\hat{\boldsymbol{\theta}}^{ridge} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}}$$

Hat matrix

$$\hat{y} = \boldsymbol{X}\hat{\boldsymbol{\theta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

How do we compute degrees of freedom here?

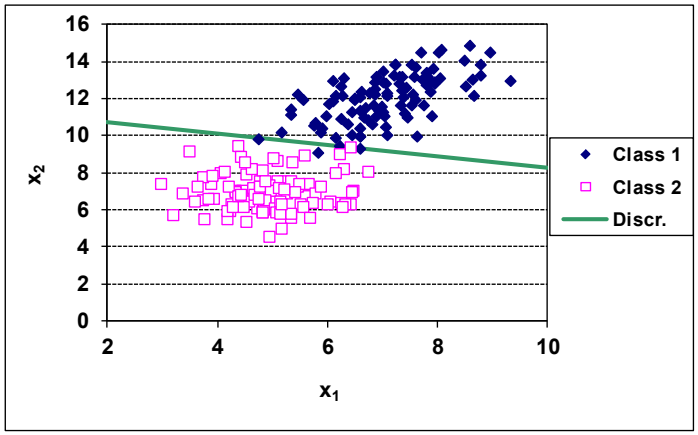
- **Note:** if $p > n$ then $\boldsymbol{X}^T\boldsymbol{X}$ **is** invertible for large enough λ

Classification

- Given data $D = \left((\boldsymbol{x}_i, y_i), i = 1 \dots n\right)$
  - $y_i = y(X_i) = C_j \in \boldsymbol{C}$
  - Class set $\boldsymbol{C} = \{C_1, \dots, C_M\}$

Classification problem:

- Decide $\hat{y}(\boldsymbol{x})$ that maps **any** $\boldsymbol{x}$ into some class $C_j \in \boldsymbol{C}$
  - Decision boundary



Classifiers

- **Deterministic**: decide a rule that directly maps $\boldsymbol{x}$ into $\hat{y}$

- **Probabilistic:** define a model for $P\left(y = C_j \middle| \boldsymbol{x}\right), j = 1 \ldots M$

Disanvantages of deterministic classifiers:

- Sometimes simple mapping is not enough (risk of cancer)
- Difficult to embed loss-> rerun of optimizer is often needed
- Combining several classifiers into one is more problematic
  - Algorithm A classifies as spam, Algorithm B classifies as not spam → ???
  - P(Spam|A)=0.99, P(Spam|B)=0.45 → better decision can be made

Logistic regression

- Discriminative model
- Model for binary output
  - $C = \{C_1 = -1, C_2 = 1\}$
    $p(y = C_1|\boldsymbol{x}) = g(\boldsymbol{x}) = sigm(\boldsymbol{\theta}^T\boldsymbol{x})$

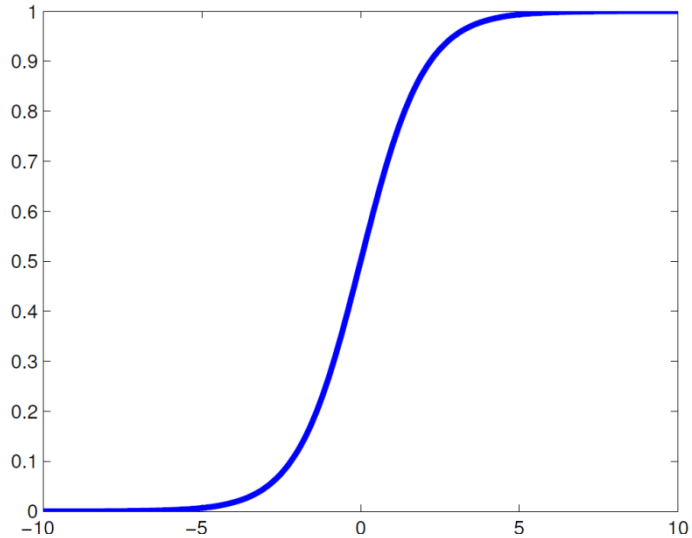
$$sigm(a) = \frac{1}{1 + e^{-a}}$$

- Alternatively

$$y \sim Bernoulli(sigm(a)), a = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$$

$$sigm(a) = \frac{1}{1 + e^{-a}}$$

What is $\mathrm{p}(y = C_2|x)$?



Logistic regression

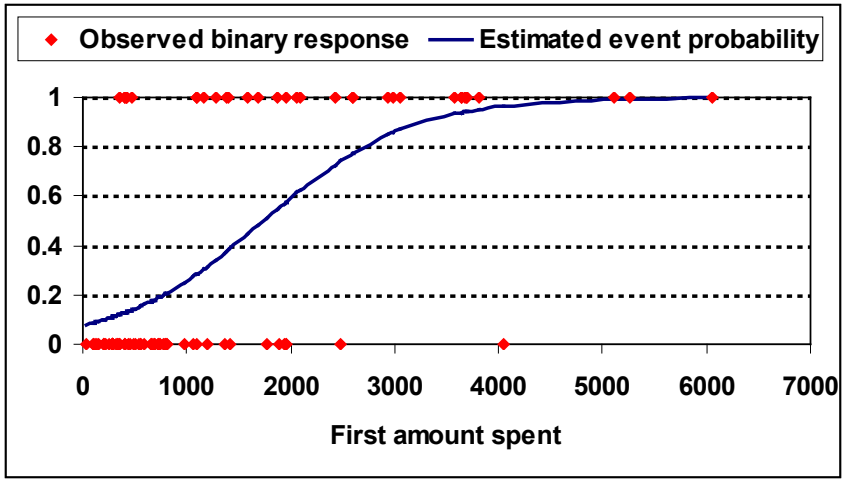
- Logistic model- yet another form

$$ln\frac{p(y=1|\boldsymbol{x})}{P(y=-1|\boldsymbol{x})} = ln\frac{p(y=1|\boldsymbol{x})}{1-p(y=1|\boldsymbol{x})} = logit(p(y=1|\boldsymbol{x})) = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$$

The log of the odds is linear in *x*

- Here $logit(t) = \ln\left(\frac{t}{1-t}\right)$
- Note $p(y|x)$ is connected to $\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}$ via logit link

Example: Probability to buy more than once as function of First Amount Spend



Logistic regression: learning

- Likelihood maximization

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathbf{y} \mid \mathbf{X}; \boldsymbol{\theta})$$

- Equivalent to

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-y_i \boldsymbol{\theta}^T \boldsymbol{x_i}})$$

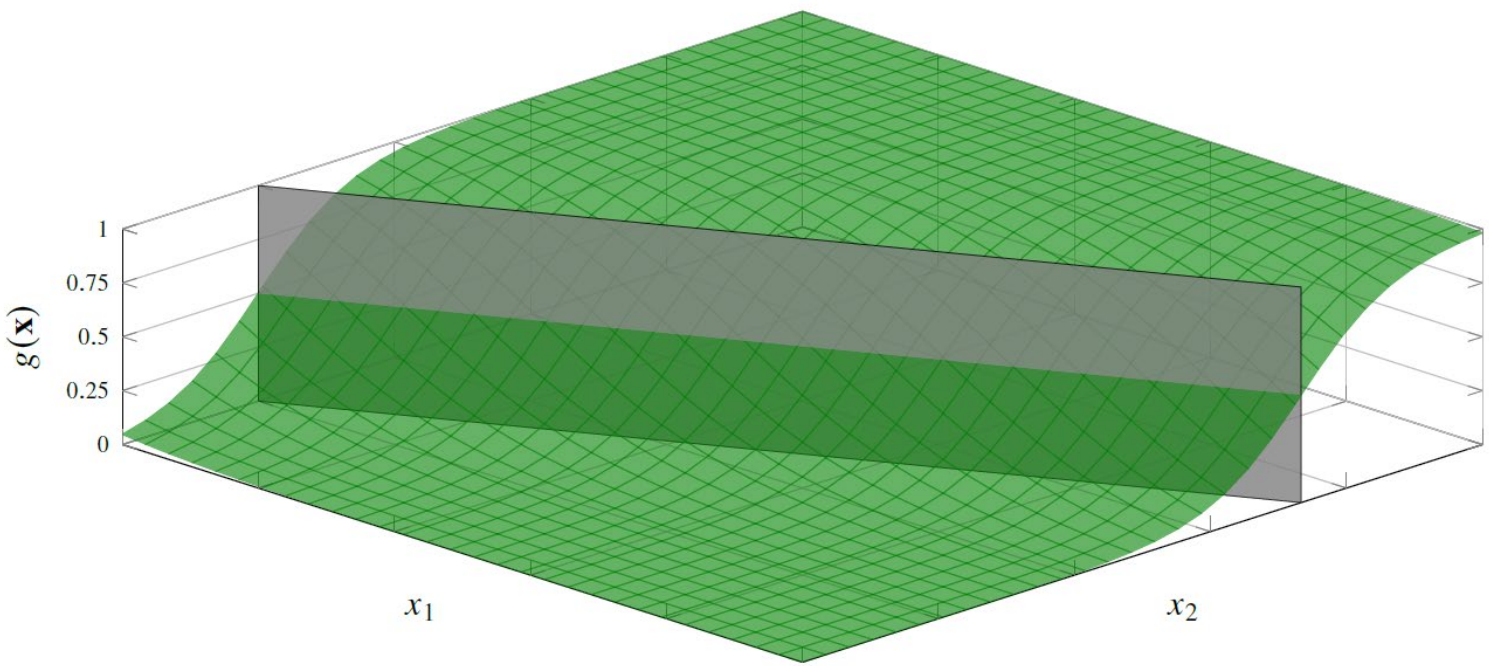
- To maximize log-likelihood, optimization used
  - Newton's method traditionally used (Iterative Reweighted Least Squares)
  - Steepest descent, Quasi-newton methods…

Logistic regression: learning algorithm

- Training
  1. Construct $\boldsymbol{X}$ and $\boldsymbol{y}$
  2. Compute $\widehat{\boldsymbol{\theta}}$ by computing $\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-y_i \boldsymbol{\theta}^T \boldsymbol{x_i}})$ numerically

- Prediction
  - $p(y = 1|\boldsymbol{x}_*) = g(\boldsymbol{x}_*, \widehat{\boldsymbol{\theta}}) = \frac{1}{1+e^{-\widehat{\boldsymbol{\theta}}^T \boldsymbol{x}_*}}$
  - Transform into decision: $\hat{y} = 1$ if $p(y = 1|\boldsymbol{x}_*) > r$
  - Normally $r = 0.5$

Logistic regression: learning algorithm



$r = 0.5$ is usual but other values might be preferred

Example: healthy vs sick

Logistic regression

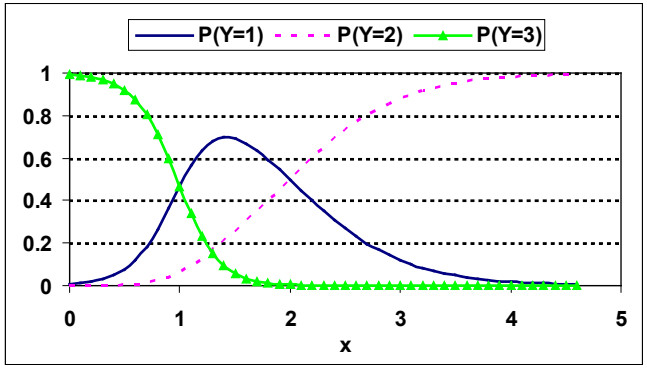
- When y is categorical, set **one** parameter vector $\boldsymbol{\theta}_i$ per class C_i

$$p(Y = C_i|x) = g_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{\theta}_i^T \boldsymbol{x}}}{\sum_{j=1}^{M} e^{\boldsymbol{\theta}_j^T \boldsymbol{x}}}$$

$$softmax(\boldsymbol{z}) = \left(\frac{\mathrm{e}^{\mathrm{z}_1}}{\sum e^{z_i}}, \dots \frac{\mathrm{e}^{\mathrm{z}_\mathrm{M}}}{\sum e^{z_i}}\right)$$

- Alternatively

$$Y \sim Multinomial\big(g_1(\boldsymbol{x}), \dots g_M(\boldsymbol{x})\big)$$



Decision boundaries in logistic regression are linear!

Logistic regression: learning

- Maximum likelihood

$$\min_{\theta} \frac{1}{n} \sum_{I=1}^{n} -\boldsymbol{\theta}_{k:y_i=c_k}^T x_i + \ln \sum_{j=1}^{M} e^{\boldsymbol{\theta}_j^T \boldsymbol{x}_i}$$

$$\mathbf{y} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 1 \\ 3 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \ln g_1(\mathbf{x}_1;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_1;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_1;\boldsymbol{\theta}) \\ \ln g_1(\mathbf{x}_2;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_2;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_2;\boldsymbol{\theta}) \\ \ln g_1(\mathbf{x}_3;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_3;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_3;\boldsymbol{\theta}) \\ \ln g_1(\mathbf{x}_4;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_4;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_4;\boldsymbol{\theta}) \\ \ln g_1(\mathbf{x}_5;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_5;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_5;\boldsymbol{\theta}) \\ \ln g_1(\mathbf{x}_6;\boldsymbol{\theta}) & \ln g_2(\mathbf{x}_6;\boldsymbol{\theta}) & \ln g_3(\mathbf{x}_6;\boldsymbol{\theta}) \end{bmatrix}$$

Logistic regression

- Regularization can also be done

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + \exp\left(-y_i \boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}_i\right)\right) + \lambda \|\boldsymbol{\theta}\|_2^2$$

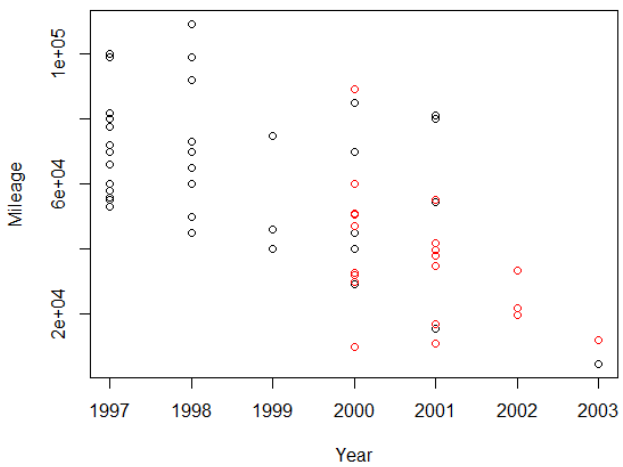
- Similar model to Logistic :**Linear Discriminant Analysis (LDA):**
  - Same expression for $p(y|\boldsymbol{x}, \boldsymbol{\theta})$
  - Parameters optimized differently
  - Advantage: faster to compute
  - Disadvantage: stronger assumptions on data.

Logistic regression

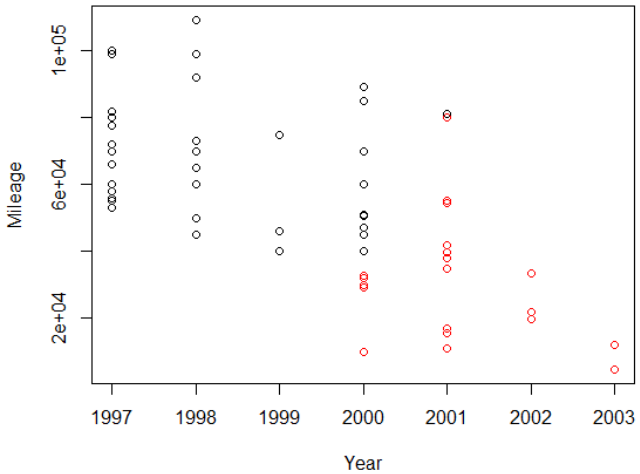
- In R, use glm() with family=”binomial” for two classes
  - predict(glmobj, type). Choose type ”response” for probabilities
- In R, use multinom () from **nnet** package for more than two classes

Example Equipment=f(Year, mileage)

Original data



Classified data



Optimization

- How to optimize a given log-likelihood?
  - In R, use optim(start_point, function, method)

- **Example**: minimizing $(x_1 - 1)^2 + (x_2 - 2)^2$

```
df=data.frame(x1=1,x2=2)
mylikelihood<-function(x){
  x1=x[1]
  x2=x[2]
  return((x1-df$x1)^2+(x2-df$x2)^2)
}

optim(c(0,0), fn=mylikelihood, method="BFGS")
```

```
> optim(c(0,0), fn=mylikelihood, method="BFGS")
$par
[1] 1 2

$value
[1] 6.357135e-26

$counts
function gradient
       9        3

$convergence
[1] 0
```