

Regression, classification and regularization

Lecture 1d

The top of the slide features a blue header with a background of binary code (0s and 1s) and a magnifying glass icon. The word "Overview" is written in white, centered in the header.

Overview

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

Simple linear regression

Given:

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

Model:

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

or

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

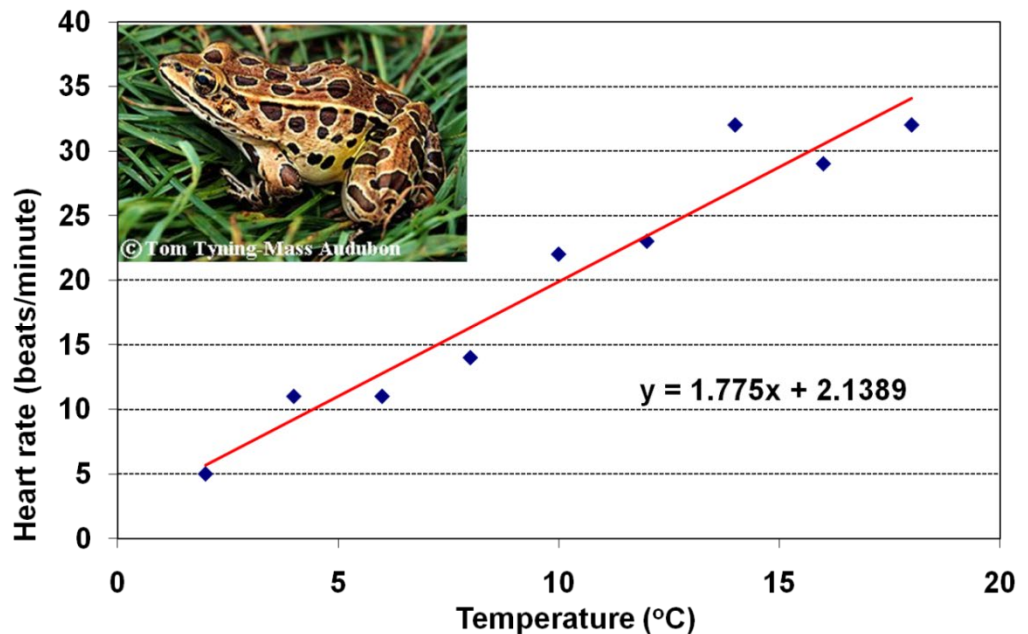
or

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

Terminology:

θ_0 : intercept (or bias)

θ_1 regression coefficient



Linear regression

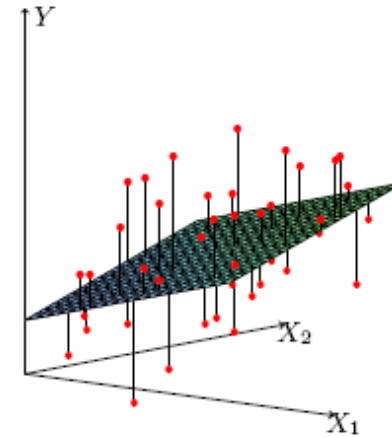
Model:

$$y = \theta_0 + \theta_1 x_1 + \dots + \theta_p x_p + \epsilon$$
$$y \sim N(\boldsymbol{\theta}^T \mathbf{x}, \sigma^2)$$

where

$$\boldsymbol{\theta} = \{\theta_0, \dots, \theta_p\}$$
$$\mathbf{x} = \{1, x_1, \dots, x_p\}$$

Why is "1" here?



Aim: learn optimal parameters

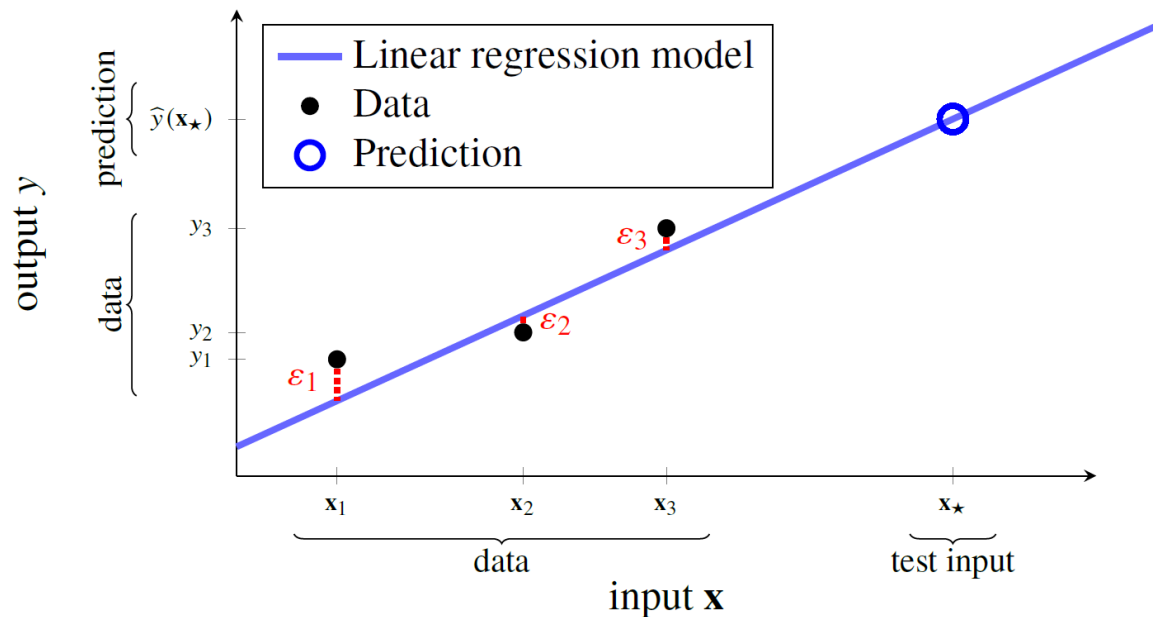
$$\hat{\theta}_0, \dots, \hat{\theta}_p$$

Linear regression

- Prediction with linear regression for x_*

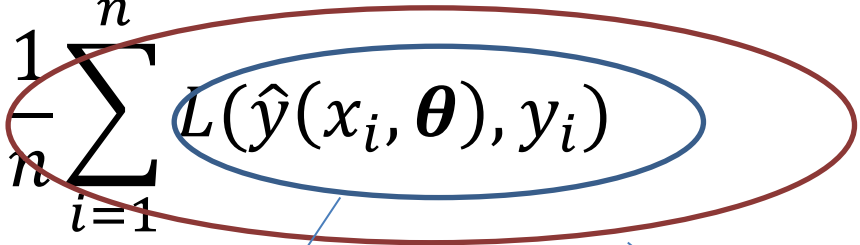
$$\hat{y}(x_*, \hat{\theta}) = \hat{\theta}_0 + \hat{\theta}_1 x_{*1} + \cdots \hat{\theta}_p x_{*p} = \hat{\theta} x_*$$

- Irreducible error ϵ



Linear model: learning

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

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


Loss function

Cost function

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

Linear model: learning

Ordinary least squares

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

Optimality condition:

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

where

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

and

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

Linear model: learning

- Least squares estimates of the parameters

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

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

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

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



Hat matrix

Why is it called so?

Linear regression: learning algorithm

- Training
 1. Construct \mathbf{X} and \mathbf{y}
 2. Compute $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$
- Prediction
 - $\hat{y}(\mathbf{x}_*) = \hat{\boldsymbol{\theta}}^T \mathbf{x}_*$

Linear model learning

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

$$\hat{\theta} = \max_{\theta} p(T|\theta) = \max_{\theta} p(y|X, \theta)$$

Exercise: Approach A and B are equivalent

Degrees of freedom

Definition:

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

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

$$df = \text{trace}(S)$$

- For linear regression, degrees of freedom is

$$df = \text{trace}(P) = p$$

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: 0.9921, 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

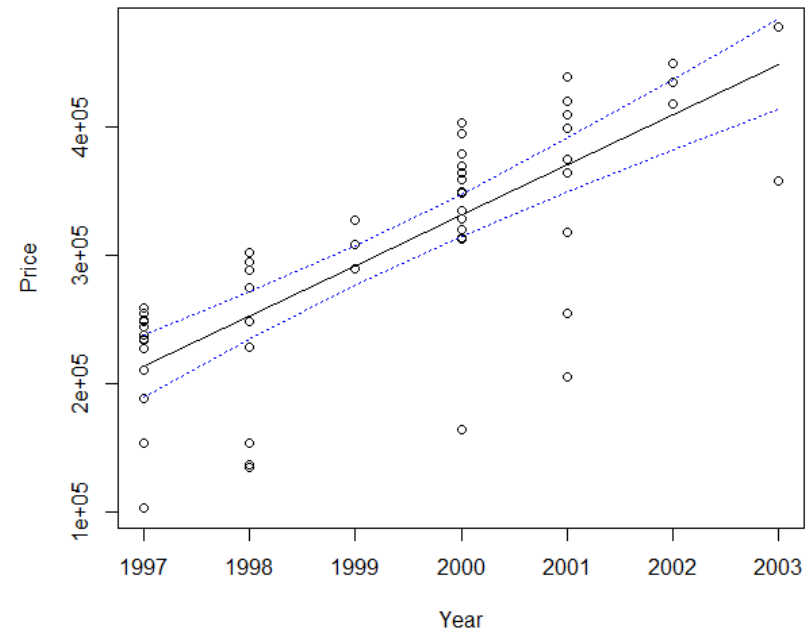
Multiple R-squared: 0.8997, Adjusted R-squared: 0.8942

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)
```



Different types of features

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

Basis function expansion:

If $y = w_0 + w_1x_1 + w_2x_1^2 + w_3e^{-x_2} + \epsilon$,

Model becomes linear if to recompute:

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

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 \mathbf{X} , compute new data

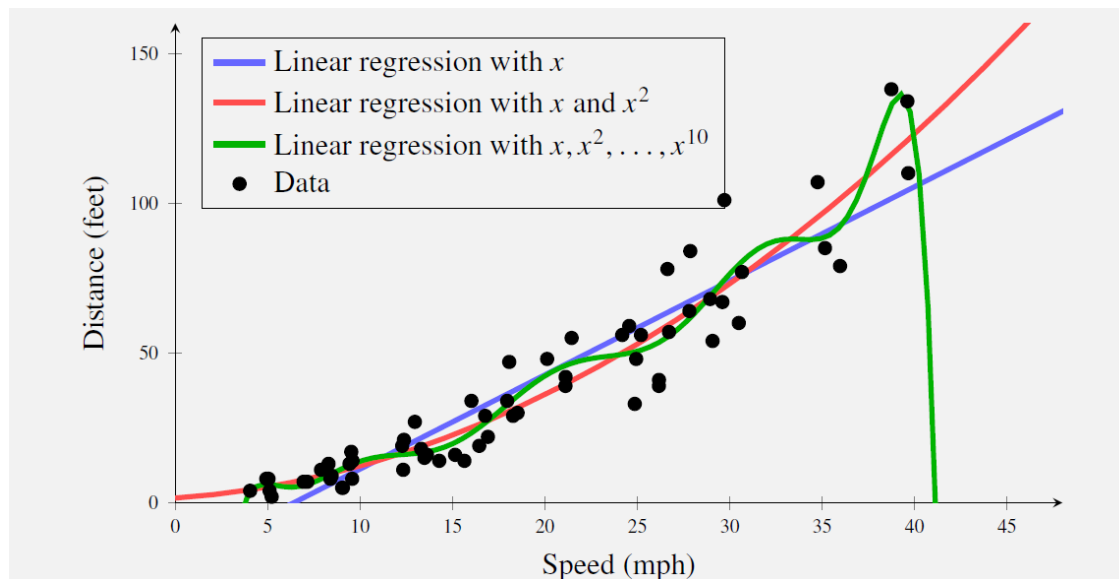
- $$\Phi = \begin{pmatrix} 1 & \phi_1(x_{11}, \dots, x_{1p}) & \dots & \phi_p(x_{11}, \dots, x_{1p}) \\ \dots & \dots & \dots & \dots \\ 1 & \phi_1(x_{n1}, \dots, x_{np}) & \dots & \phi_p(x_{n1}, \dots, x_{np}) \end{pmatrix}$$

- If doing a basis function in a model, replace \mathbf{X} by Φ everywhere where \mathbf{X} is used:

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

Polynomial regression

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



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**

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

Regularization

Equivalent form

$$\hat{\theta}^{ridge} = \operatorname{argmin} \sum_{i=1}^N (y_i - \theta_0 - \theta_1 x_{1i} - \dots - \theta_p x_{pi})^2$$

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

Solution

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

Hat matrix

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

How do we
compute degrees
of freedom here?

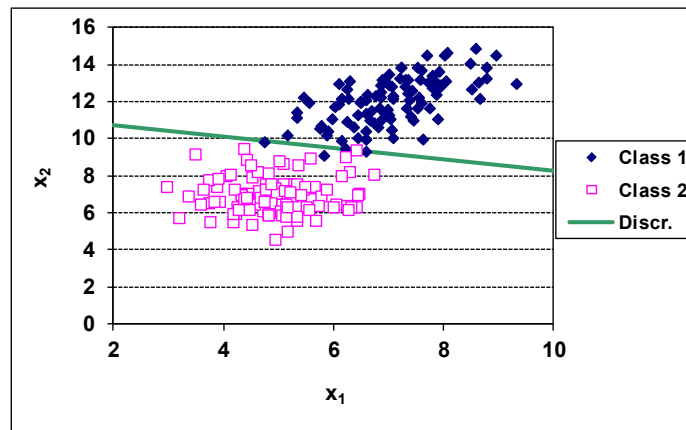
•**Note:** if $p > n$ then $X^T X$ is invertible for large enough λ

Classification

- Given data $D = ((x_i, y_i), i = 1 \dots n)$
 - $y_i = y(x_i) = C_j \in \mathcal{C}$
 - Class set $\mathcal{C} = \{C_1, \dots, C_M\}$

Classification problem:

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



Classifiers

- **Deterministic:** decide a rule that directly maps x into \hat{y}
- **Probabilistic:** define a model for $P(y = C_j | x), j = 1 \dots M$

Disadvantages of deterministic classifiers:

- Sometimes simple mapping is not enough (risk of cancer)
- Difficult to embed loss \rightarrow 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 \rightarrow ???
 - $P(\text{Spam} | A) = 0.99$, $P(\text{Spam} | B) = 0.45 \rightarrow$ better decision can be made

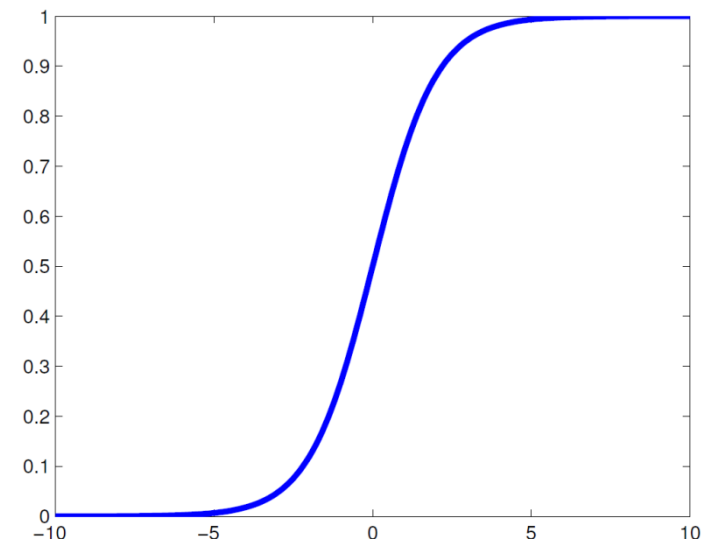
Logistic regression

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

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

- Alternatively
 $y \sim \text{Bernoulli}(\text{sigm}(a)), a = \theta^T x$
$$\text{sigm}(a) = \frac{1}{1 + e^{-a}}$$

What is $p(y = C_2|x)$?



Logistic regression

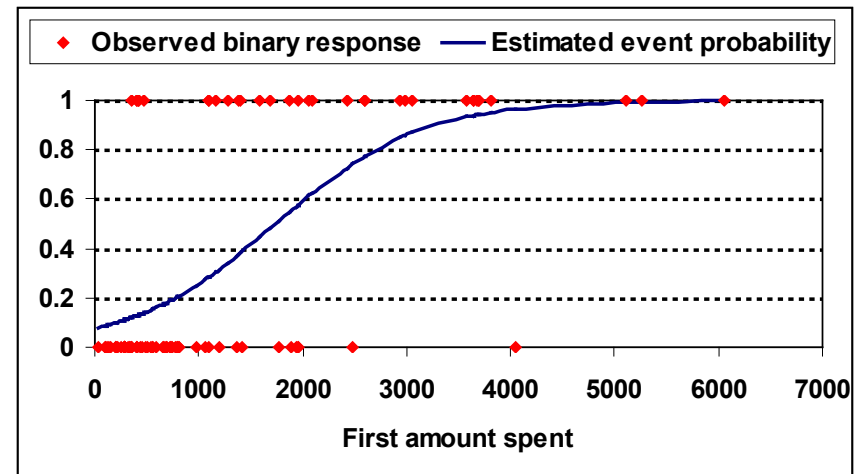
- Logistic model- yet another form

$$\ln \frac{p(y = 1|x)}{p(y = 0|x)} = \ln \frac{p(y = 1|x)}{1 - p(y = 1|x)} = \text{logit}(p(y = 1|x)) = \theta^T x$$

**The log of the odds
is linear in x**

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

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



Logistic regression: learning

- Likelihood maximization

$$\hat{\theta} = \arg \max_{\theta} p(\mathbf{y} | \mathbf{X}; \theta)$$

- Equivalent to

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

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

Logistic regression: learning algorithm

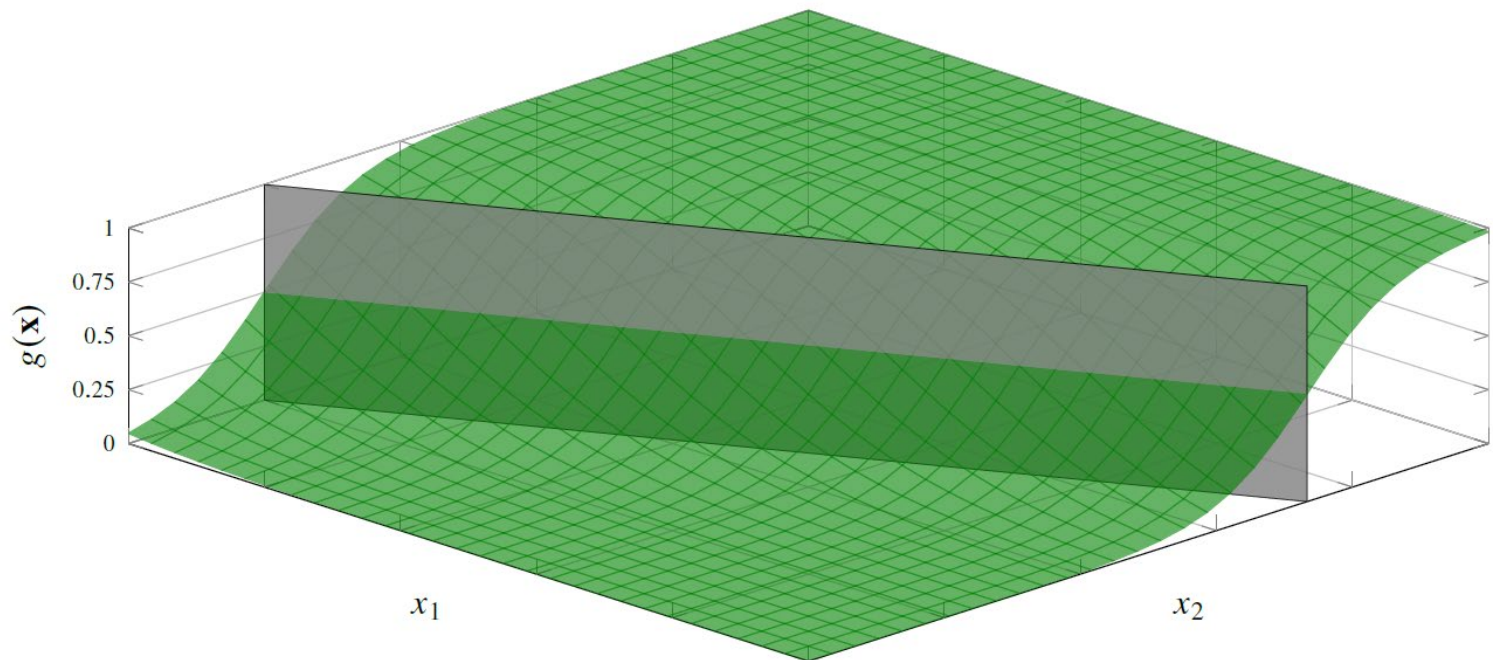
- Training

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

- Prediction

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

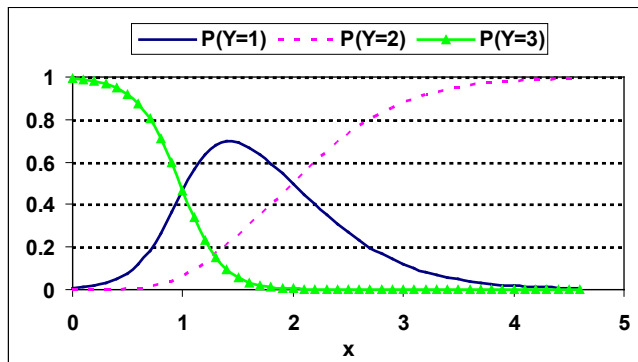
- When y is categorical, set **one** parameter vector θ_i per class C_i

$$p(Y = C_i | x) = g_i(x) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^M e^{\theta_j^T x}}$$

$$\text{softmax}(\mathbf{z}) = \left(\frac{e^{z_1}}{\sum e^{z_i}}, \dots, \frac{e^{z_M}}{\sum e^{z_i}} \right)$$

- Alternatively

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



Decision boundaries in logistic regression are linear!

Logistic regression: learning

- Maximum likelihood

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^n -\theta_{k:y_i=c_k}^T x_i + \ln \sum_{j=1}^M e^{\theta_j^T x_i}$$

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

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

Logistic regression

- Regularization can also be done

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

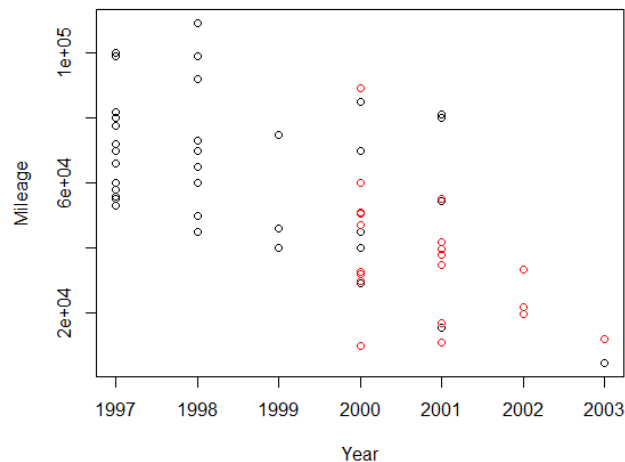
- Similar model to Logistic: **Linear Discriminant Analysis (LDA)**:
 - Same expression for $p(y|x, \theta)$
 - Parameters optimized differently
 - Advantage: faster to compute
 - Disadvantage: stronger assumptions on data.

Logistic regression

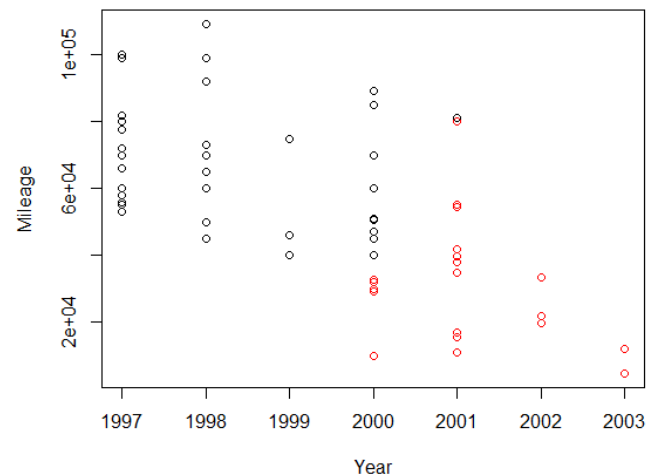
- In R, use `glm()` with `family="binomial"` for two classes
 - `predict(glmobj, type)`. Choose type "response" for probabilities
- In R, use `multinom ()` from **nnet** package for more than two classes

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
```