Linear Regression

Digging into Data: Jordan Boyd-Graber

University of Maryland

March 11, 2013

COLLEGE OF INFORMATION STUDIES

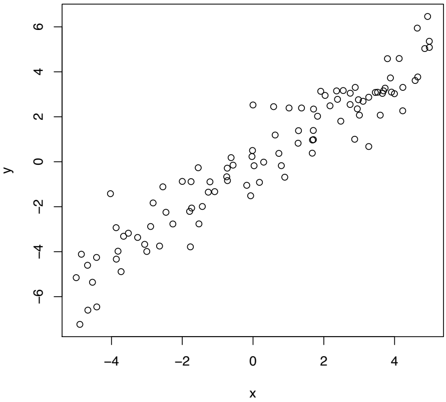
Regression and Classification

Recall:

- *Classification* takes a set of features X and for each input x_i gives a discrete output y (e.g. given words in a document say whether it's spam or not)
- *Regression* takes a set of features X and for each input x_i gives a continuous response y (e.g. given words in a document say how many stars the review gives to a product on Amazon)

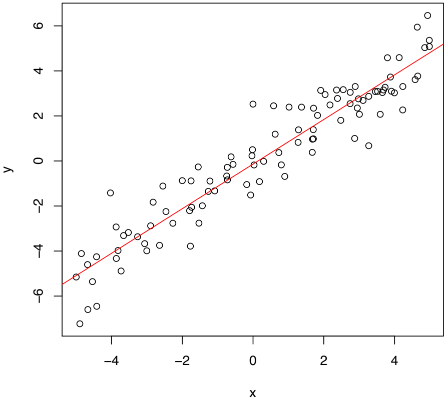
Outline

Linear Regression



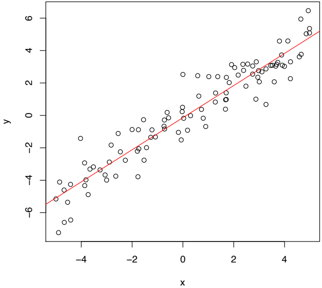
Data are the set of inputs and outputs, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$

Linear Regression



In *linear regression*, the goal is to predict y from x using a linear function

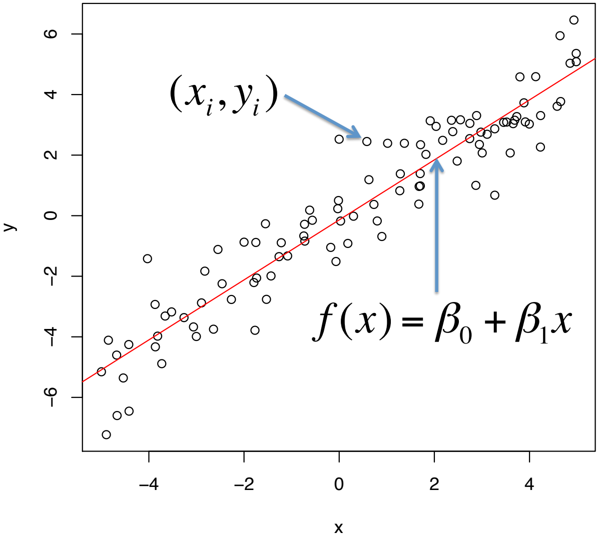
Linear Regression



Examples of linear regression:

- given a child's age and gender, what is his/her height?
- given unemployment, inflation, number of wars, and economic growth, what will the president's approval rating be?
- given a browsing history, how long will a user stay on a page?
- others?

Linear Regression



Multiple Covariates

Often, we have a vector of inputs where each represents a different *feature* of the data

$$\mathbf{x} = (x_1, \dots, x_p)$$

The function fitted to the response is a linear combination of the covariates

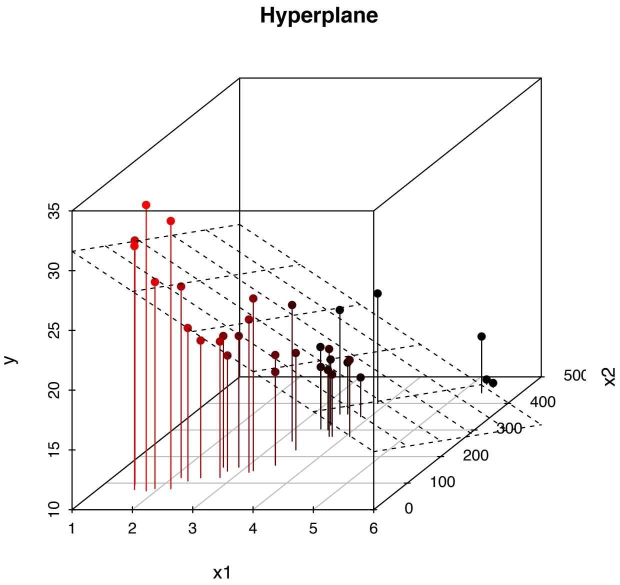
$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

Multiple Covariates

- Often, it is convenient to represent **x** as $(1, x_1, \ldots, x_p)$
- In this case **x** is a vector, and so is β (we'll represent them in bold face)
- This is the dot product between these two vectors
- This then becomes a sum (this should be familiar!)

$$\beta \mathbf{x} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

Hyperplanes: Linear Functions in Multiple Dimensions

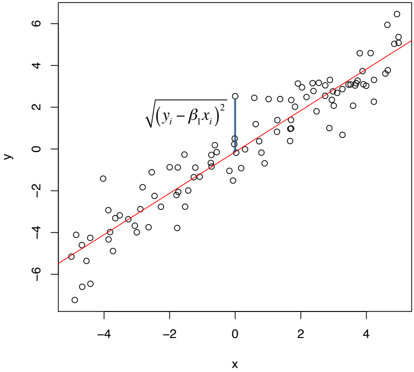


Covariates

- Do not need to be raw value of $x_1, x_2, \ldots$
- Can be any feature or function of the data:
  - Transformations like $x_2 = \log(x_1)$ or $x_2 = \cos(x_1)$
  - Basis expansions like $x_2 = x_1^2$, $x_3 = x_1^3$, $x_4 = x_1^4$, etc
  - Indicators of events like $x_2 = 1_{\{-1 \leq x_1 \leq 1\}}$
  - Interactions between variables like $x_3 = x_1 x_2$
- Because of its simplicity and flexibility, it is one of the most widely implemented regression techniques

Outline

Fitting a Linear Regression



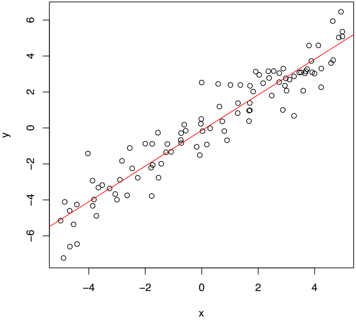
Idea: minimize the Euclidean distance between data and fitted line

$$RSS(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta \mathbf{x}_i)^2$$

How to Find β

- Use calculus to find the value of β that minimizes the RSS
- The optimal value is

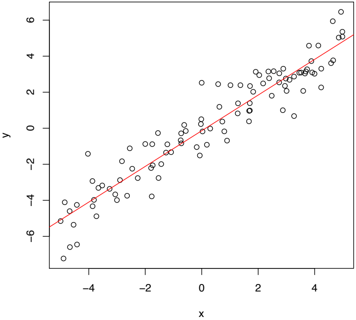
$$\hat{\beta} = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2}$$



Prediction

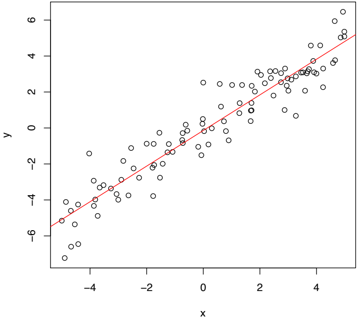
- After finding $\hat{\beta}$, we would like to predict an output value for a new set of covariates
- We just find the point on the line that corresponds to the new input:

$$\hat{y}_{new} = \hat{\beta} x_{new}$$



Probabilistic Interpretation

- Our analysis so far has not included any probabilities
- Linear regression does have a *probabilisitc* (probability model-based) interpretation
- Any guesses about what it is?

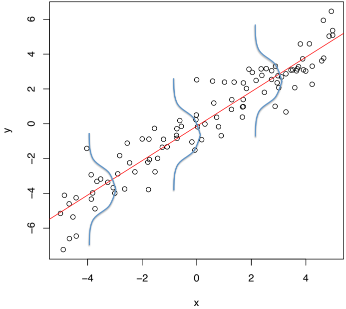


Probabilistic Interpretation

- Linear regression assumes that response values have a Gaussian distribution around the linear mean function,

$$Y_i \,|\, \mathbf{x}_i, \beta \sim N(\mathbf{x}_i \beta, \sigma^2)$$

- Like logistic regression, this is a *discriminative model*, where inputs x are not modeled



- Minimizing RSS is equivalent to maximizing conditional likelihood

Outline

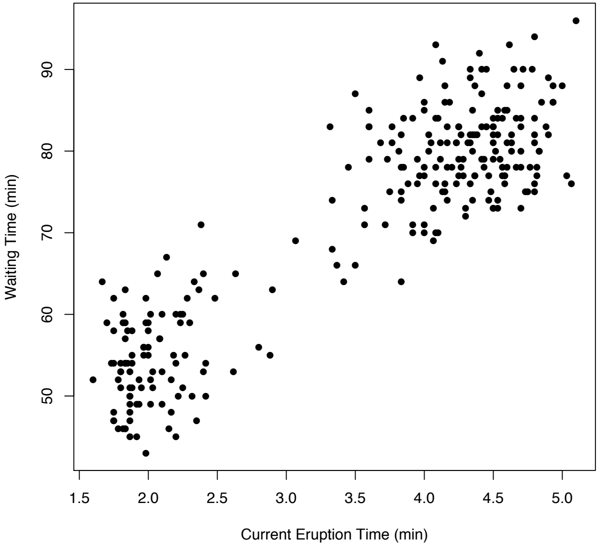
Example: Old Faithful

Example: Old Faithful

We will predict the time that we will have to wait to see the next eruption given the duration of the current eruption

```
>  library(datasets)
>  names(faithful)
[1] "eruptions" "waiting"
>  attach(faithful)
>  plot(eruptions,waiting,xlab="Current Eruption Time (min)",
+ ylab="Waiting Time (min)",pch=16)
```

Example: Old Faithful



Example: Old Faithful

To fit a linear model in R, use the `lm()` function, which stands for “linear model”

```
>  fit.lm <- lm(waiting ~ eruptions)
>  fit.lm

Call:
lm(formula = waiting ~ eruptions)

Coefficients:
(Intercept)    eruptions
      33.47        10.73

>  names(fit.lm)
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
```

Example: Old Faithful

We can plot our data and make a function for new predictions

```
>  # Plot a line on the data
>  abline(fit.lm,col="red",lwd=3)
>
>  # Make a function for prediction
>  fit.lm$coefficients[1]
(Intercept)
    33.4744
>  fit.lm$coefficients[2]
eruptions
  10.72964
>  faithful.fit <- function(x) fit.lm$coefficients[1] +
fit.lm$coefficients[2]*x
>  x.pred <- c(2.0, 2.7, 3.8, 4.9)
>  faithful.fit(x.pred)
[1] 54.93368 62.44443 74.24703 86.04964
```

Example: Old Faithful