機械学習特論

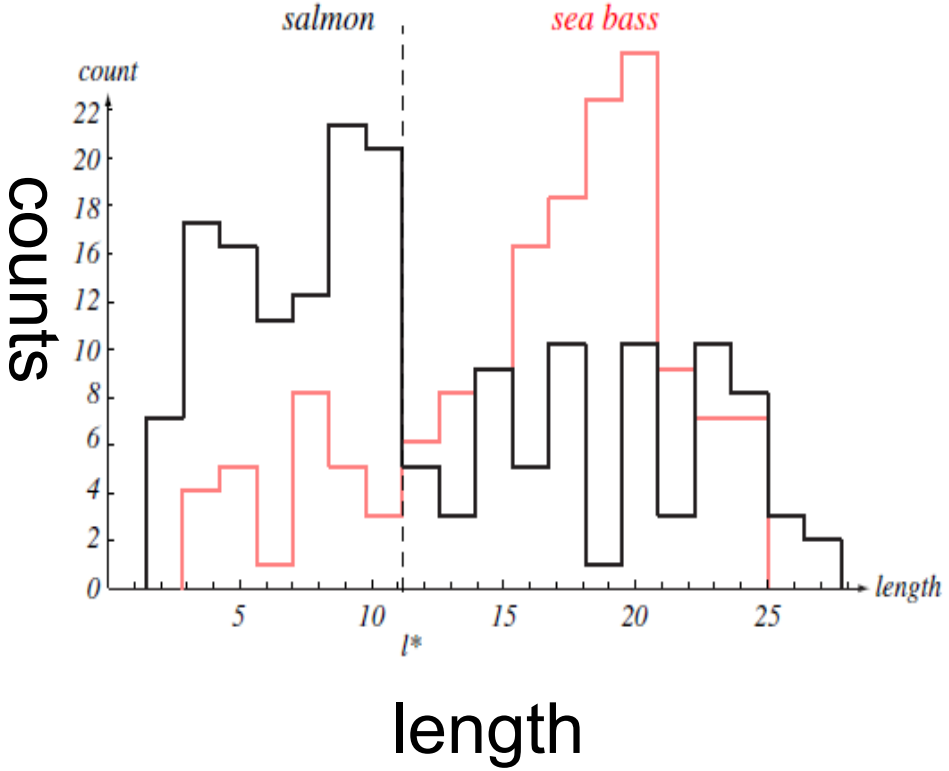
~理論とアルゴリズム~

(Linear Regression)

講師 : 西郷浩人

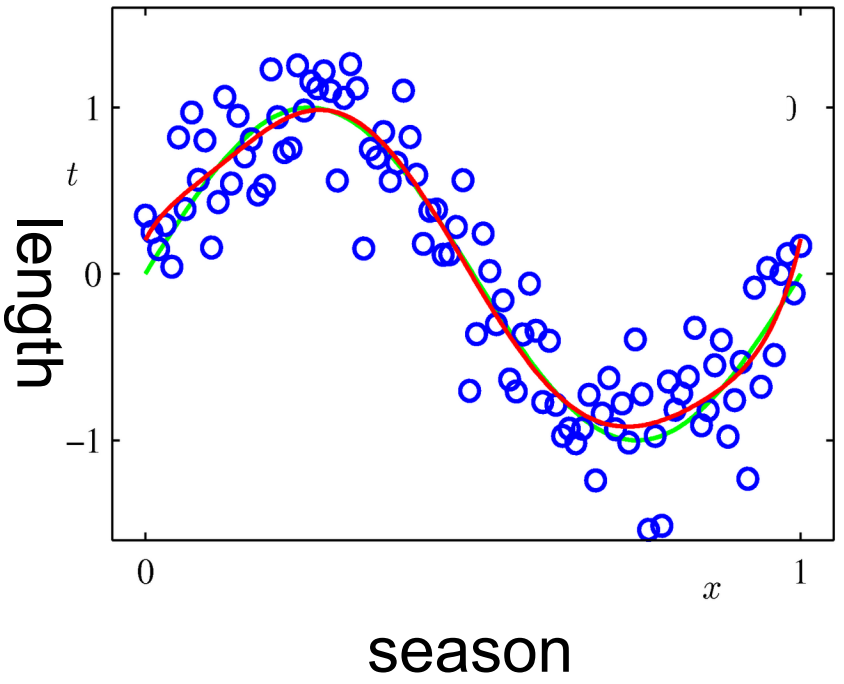
Classification

- The goal is to predict binary category (salmon or sea buss) from observations (such as length, place, season etc.)



Regression(回帰)

- The goal is to predict real values (such as length) from given observations (such as place, season, time in a day etc.)



Linear Regression (vector form)

$$X\beta = y$$

Explanatory variable
(feature)

observation
(example)

design matrix

=

Response/target variable
(label)

coefficient vector
(weight)

Example

$$\begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 5 \\ -6 \end{pmatrix}$$

Solve for $\boldsymbol{\beta}$

scalar form

$$\sum_{j=1}^{p} x_j \beta_j$$

Estimating regression parameters via least squares

Objective function

$$min_{\beta} \| \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y} \|_2^2$$

$$\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{y}$$

$$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Exploiting the convexity

$$min_{\beta} \| X\beta - y \|_2^2$$

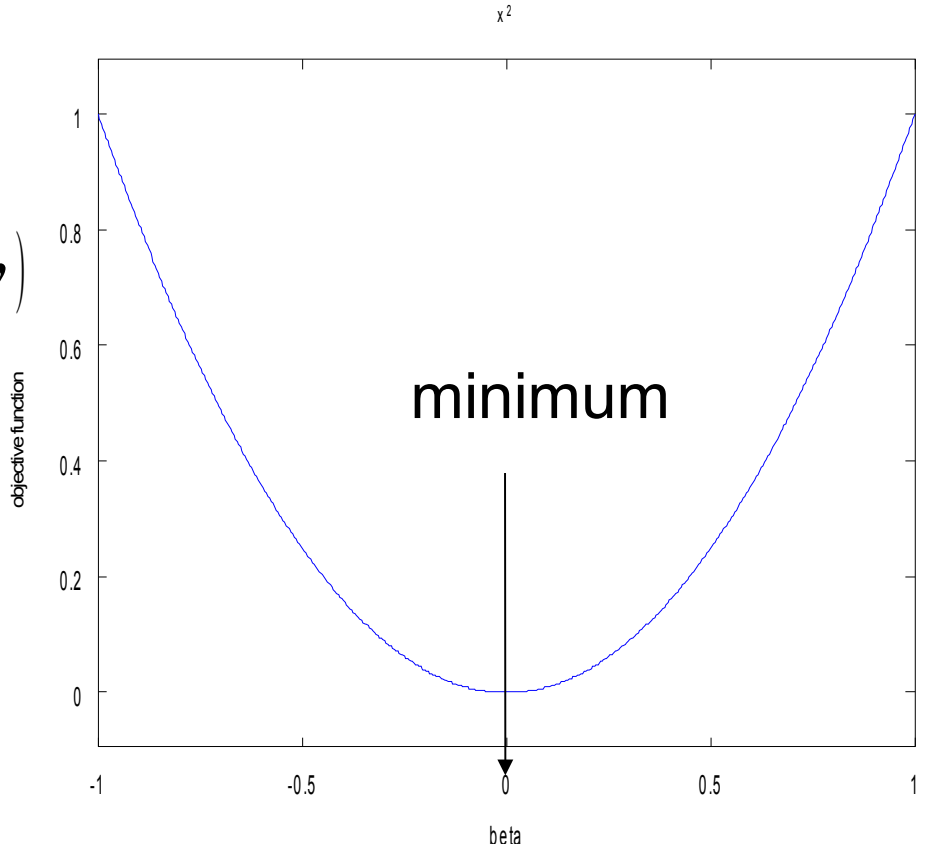
$$Let\ L = (X\beta - y)'(X\beta - y)$$

$$\frac{\partial L}{\partial \beta} = \frac{\partial}{\partial \beta}(\beta' X' X\beta - 2\beta' X' y - y' y)$$

$$= 2X' X\beta - 2X' y$$

$$By\ setting\ \frac{\partial L}{\partial \beta} = 0,$$

$$\leftarrow\rightarrow X' X\beta = X' y$$



%beta

Normal equation

$$X'X\beta = X'y$$

- Numerical solution is obtained by QR decomposition; X=QR
  - Q is a orthogonal matrix
  - R is a right triangular matrix
- Using QR decomposition, solution to the normal equation is obtained by

$$X'X\beta = X'y$$
$$(QR)'(QR)\beta = (QR)'y$$
$$R'Q'QR\beta = R'Q'y$$
$$R\beta = Q'y$$

  - The last equation can be solved by back substitution.

Solving with octave

- Directly solve $\beta = \left(X' X \right)^{-1} X' y$

```
beta = inv(X'*X)*X'*y
```

- Using backslash operator: \

```
beta = X\y
```

  - In practice, \ is always preferred, since it implements a more sophisticated method that works for a case when X is close to singular.

Evaluating Least Squares Regression

- Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

  - Un-normalized measure. Can take from 0 to infinity.

- R-squared:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

  - Normalized measure. Takes values between minus infinity to 1. Above 0.5 is considered good.
  - where $\bar{y}$ is the mean, and $R^2 <= 1$
  - R^2 takes higher value if the variance in the response is better explained by the prediction $\hat{y}$
  - For linear regression, R^2 is equivalent to *correlation.*

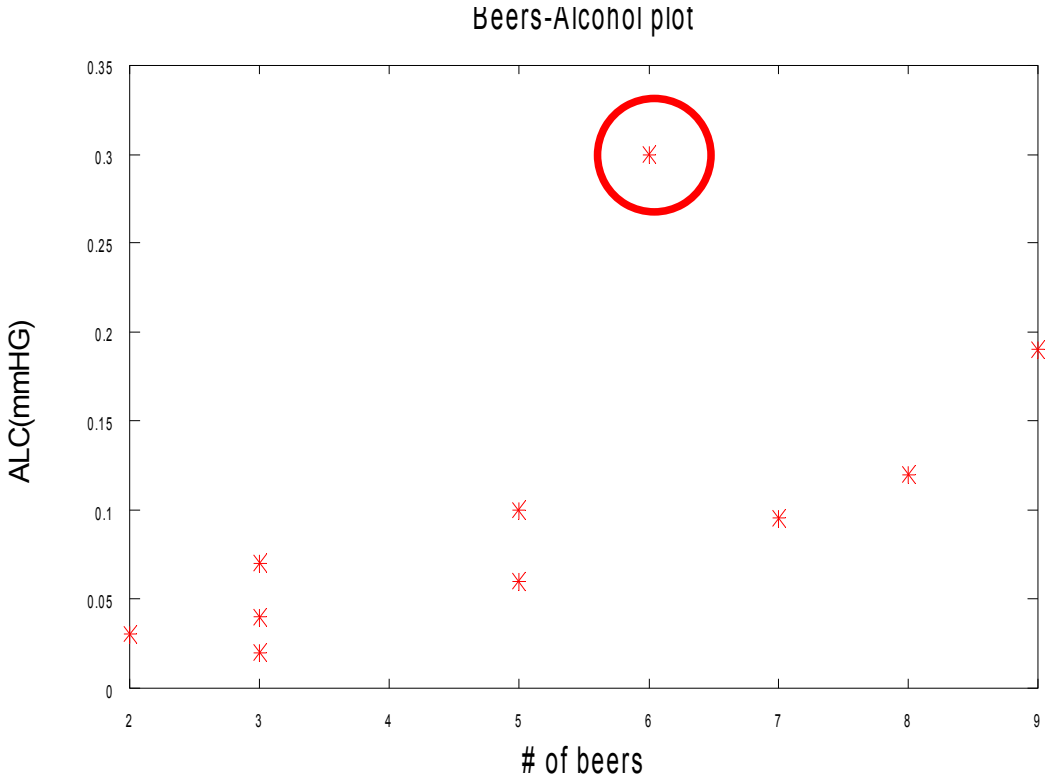
Example1 Outlier detection

- The following table displays relationship between the amount of beer consumption and blood-alcohol concentration (BAC).

Beer（bottles）	5	2	9	8	3	7	3	5	3	6
BAC (mmHG)	0.10	0.03	0.19	0.12	0.04	0.095	0.07	0.06	0.02	0.3

Plotting the data

- We can find one outlier.




```
>plot(x,y,'r*');
>xlabel('# of beers','fontsize',24);
>ylabel('Alcohol(mmHG)','fontsize',24);
>title('Beers-Alcohol plot','fontsize',24);
```

Example 1

- Construct linear regression models with/without outlier, and compare them in terms of residual sum of square (RSS). Include a bias term in linear model. (the next slide shows how to do it on octave.)

$$y = \beta_0 + \beta_1 x$$

How to include a bias term

- Let out data be $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$, we can include a bias term by solving a system of equation with modified design matrix; $\begin{pmatrix} 1\, x_1 \\ 1\, x_2 \\ 1\, x_3 \end{pmatrix}$
- Since $\begin{pmatrix} 1\, x_1 \\ 1\, x_2 \\ 1\, x_3 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \beta_0 + \beta_1 x_3 \end{pmatrix}$
- Example in octave

```
X = [1 2 3]'
X = [ones(3,1) X]
y = [10 30 50]'
b = X¥y
rss = sum((X*b – y).^2)
```

Example 2 Non-linear model

- Which of the following model fit best to the data in the table ?
  - Linear model
  - Quadratic model
  - Cubic model

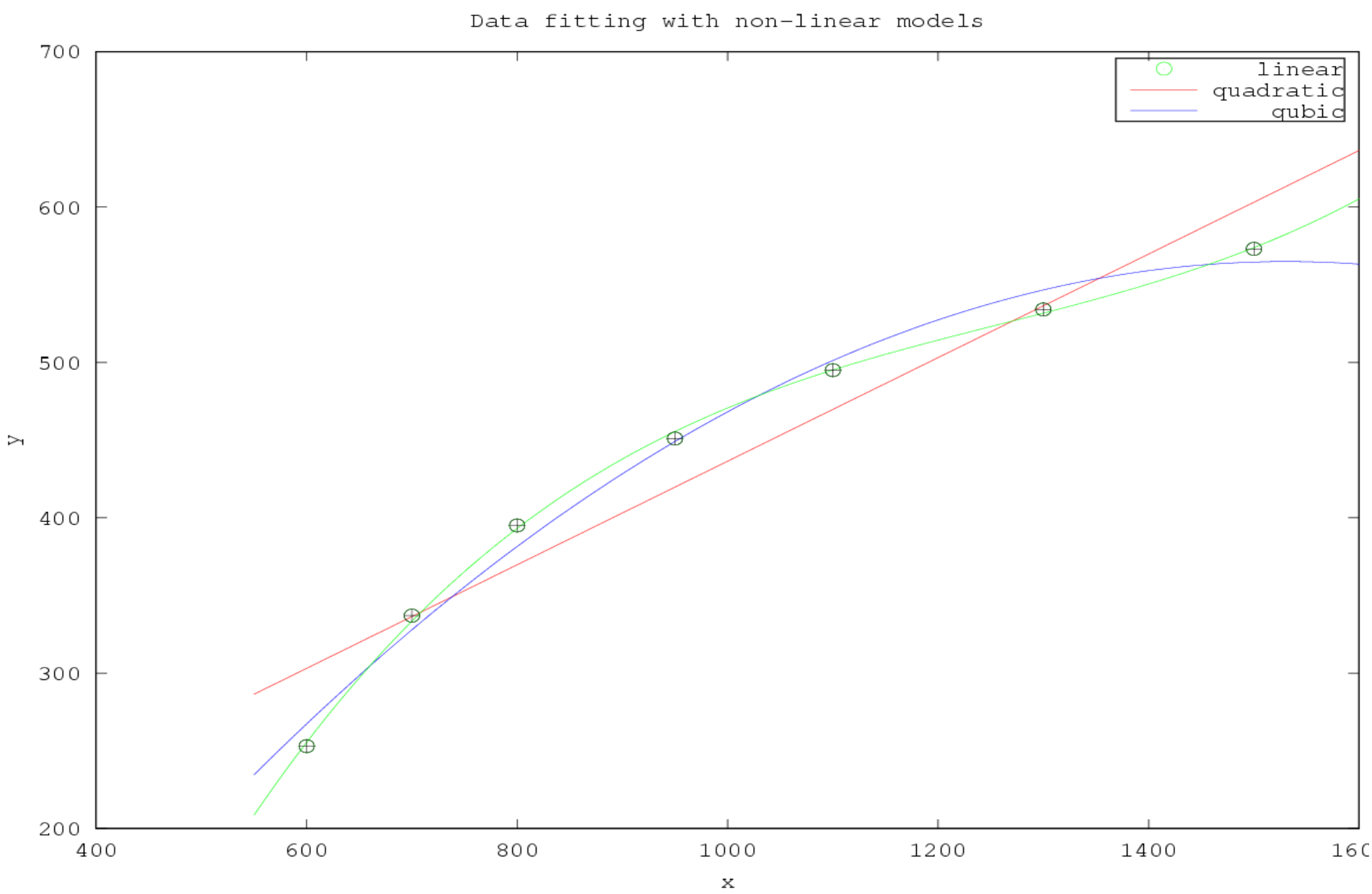
$$y = \beta_0 + \beta_1 x$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

- Compare the three models in terms of R^2, and answer the question. Next slide provides hints.

x	600	700	800	950	1100	1300	1500
y	253	337	395	451	495	534	573


Data fitting with non-linear models
linear
quadratic
qubic
700
600
500
400
300
200
400
600
800
1000
1200
1400
160
y
x

How to build higher-order polynomial models

- We can build higher order polynomial models by the same strategy as we added a bias term, that is, by modifying the design matrix.

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 \\ \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 \\ \beta_0 + \beta_1 x_3 + \beta_2 x_3^2 \end{pmatrix}$$

- Example in octave (watch out spaces !)
  - . before ^ indicates that it is an operation to the elements in the matrix. So A.^2 and A^2 differs.

```
X = [1 2 3]'
X = [1 X X.^2 X.^3]
```

Estimating linear regression parameters by Maximum Likelihood

Likelihood and Maximum Likelihood

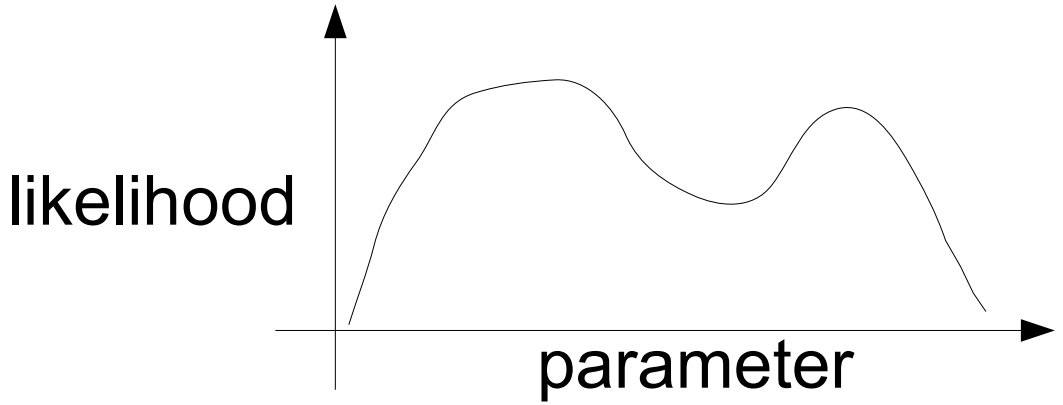
- Given a set of parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, .., \theta_n\}$, we have data distribution $\boldsymbol{D} = \{x_1, x_2, .., x_n\}$. The probability of the data generated (likelihood) is defined as

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta})$$

  - Higher likelihood suggests that the data are likely to be generated. We look for such parameter sets **θ.**
  - A method to search **θ** that maximizes likelihood is called Maximum Likelihood (最尤法), and represented as the estimated parameter is often called as Maximum Likelihood Estimate (MLE):

$$\hat{\boldsymbol{\theta}} = argmax_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{\theta})$$

Maximizing Likelihood

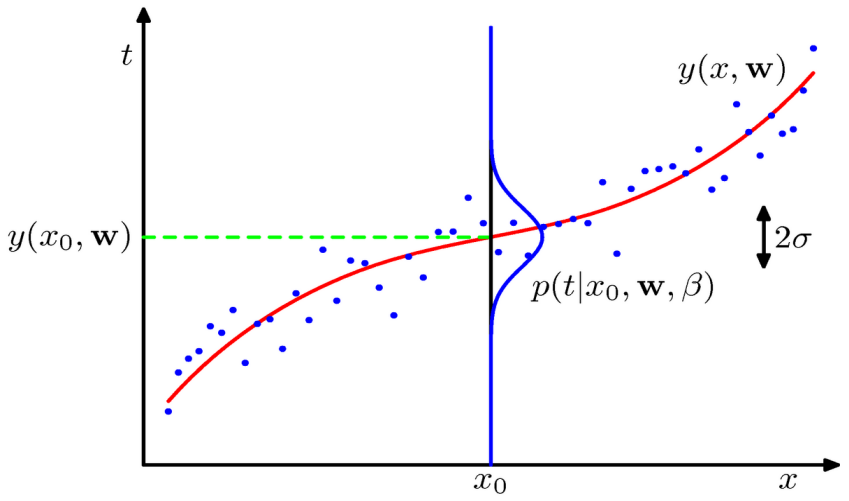
- 

- Even if we do not know the shape of likelihood space, local maximum occurs at a point where a derivative with respect to parameter is zero (necessary condition).

$$\frac{\partial L(\hat{\boldsymbol{\theta}}_{ML})}{\partial \hat{\boldsymbol{\theta}}_{ML}} = \mathbf{0}$$

- Sufficient conditions should be checked on each case.

In case of linear regression



- Assume that our response variable is measured with some observation noise that follows gaussian distribution.
  - Let $\tilde{y}_i$ be the observed variable, $f\left(\boldsymbol{x}_i\right)=\boldsymbol{x}_i{}'\boldsymbol{\beta}^*$ be the true variable, and $\epsilon=N\left(0,\sigma^2\right)$ be the observation noise, then the following relationship holds.

$$\tilde{y}_i=f\left(\boldsymbol{x}_i\right)+\epsilon$$

- The resulting $\tilde{y}_i$ follows Gaussian distribution with mean $\boldsymbol{x}_i{}'\boldsymbol{\beta}^*$ and variance σ^2 as follows.

$$p\left(\tilde{y}_i|\boldsymbol{x}_i{}'\boldsymbol{\beta}^*,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-\left(\boldsymbol{x}_i{}'\boldsymbol{\beta}^*-\tilde{y}_i\right)^2}{2\sigma^2}\right)$$

Likelihood of linear regression

- Probability of each data point

$$p(\tilde{y}_i|\boldsymbol{x_i}'\boldsymbol{\beta}^*,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(\boldsymbol{x_i}'\boldsymbol{\beta}^*-\tilde{y}_i)^2}{2\sigma^2}\right)$$

- Let likelihood of n data points be

$$L(\boldsymbol{\beta}^*,\sigma^2)=\prod_{i=1}^{n} p(\tilde{y}_i|\boldsymbol{x_i}'\boldsymbol{\beta}^*,\sigma^2)$$

$$=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(\boldsymbol{x_i}'\boldsymbol{\beta}^*-\tilde{y}_i)^2}{2\sigma^2}\right)$$

- By taking a logarithm

$$\log L(\boldsymbol{\beta}^*,\sigma^2)=\frac{-n}{2}\log(2\pi)-\frac{n}{2}\log(\sigma^2)-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x_i}'\boldsymbol{\beta}^*-\tilde{y}_i)^2$$

MLE for β

$$\log L(\boldsymbol{\beta}^*, \sigma^2) = \frac{-n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x}_i'\boldsymbol{\beta}^* - \tilde{y}_i)^2$$

- Take a derivative w.r.t. $\boldsymbol{\beta}$, and set it to zero.

$$\frac{\partial \log L}{\partial \boldsymbol{\beta}^*} = 0 \leftrightarrow \sum_{i=1}^{n}(\boldsymbol{x}_i'\boldsymbol{\beta}^* - \tilde{y}_i)\boldsymbol{x}_i = 0$$

- In a vectorial form

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}^* - \boldsymbol{X}'\tilde{\boldsymbol{y}} = 0$$
$$\leftrightarrow \boldsymbol{\beta}^* = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\tilde{\boldsymbol{y}}$$

  - which coincides with the least squares solution.

- The resulting β minimizes the squared error between the observed response variable and the true response variable, i.e., optimal solution under the assumption of Gaussian noise.

MLE for σ^2

$$\log L(\boldsymbol{\beta}^*, \sigma^2) = \frac{-n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\boldsymbol{x_i}'\boldsymbol{\beta}^* - \tilde{y}_i)^2$$

- By taking derivative w.r.t. $\boldsymbol{\sigma^2}$, then set it to zero;

$$\frac{\partial \log L}{\partial \sigma^2} = 0 \leftrightarrow -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{n}(\boldsymbol{x_i}'\boldsymbol{\beta}^* - \tilde{y}_i)^2 = 0$$

$$\leftrightarrow \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x_i}'\boldsymbol{\beta}^* - \tilde{y}_i)^2$$

- The obtained $\boldsymbol{\sigma^2}$ coincides with RSS/n.

Exercise 1

5.11. *Longley data set.* The Longley data set of labor statistics was one of the first used to test the accuracy of least squares computations. You don't need to go to the NIST Web site to do this problem, but if you are interested in the background, you should see the Longley page at [3]. The data set is available in `NCM` in the file `longley.dat`. You can bring the data into MATLAB with

```
load longley.dat
y = longley(:,1);
X = longley(:,2:7);
```

There are 16 observations of 7 variables, gathered over the years 1947 to 1962. The variable y and the 6 variables making up the columns of the data matrix

[3] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, *Statistical Reference Datasets.*
http://www.itl.nist.gov/div898/strd
http://www.itl.nist.gov/div898/strd/lls/lls.shtml
http://www.itl.nist.gov/div898/strd/lls/data/Longley.shtml

X are

y = Total Derived Employment,
x_1 = GNP Implicit Price Deflater,
x_2 = Gross National Product,
x_3 = Unemployment,
x_4 = Size of Armed Forces,
x_5 = Noninstitutional Population Age 14 and Over,
x_6 = Year.

The objective is to predict y by a linear combination of a constant and the six x's:

$$y \approx \beta_0 + \sum_1^6 \beta_k x_k.$$

(a) Use the MATLAB backslash operator to compute $\beta_0, \beta_1, \ldots, \beta_6$. This involves augmenting X with a column of all 1's, corresponding to the constant term.
(b) Compare your β's with the certified values [3].
(c) Use `errorbar` to plot y with error bars whose magnitude is the difference between y and the least squares fit.
(d) Use `corrcoef` to compute the correlation coefficients for X without the column of 1's. Which variables are highly correlated?
(e) Normalize the vector y so that its mean is zero and its standard deviation is one. You can do this with

```
y = y - mean(y);
y = y/std(y)
```

Do the same thing to the columns of X. Now plot all seven normalized variables on the same axis. Include a `legend`.

Note: you can use 'corr' function instead of 'corrcoef', if it does not exist in your environment.

Longley.dat

Data:	y	x1	x2	x3	x4	x5	x6
60323	83.0	234289	2356	1590	107608	1947	
61122	88.5	259426	2325	1456	108632	1948	
60171	88.2	258054	3682	1616	109773	1949	
61187	89.5	284599	3351	1650	110929	1950	
63221	96.2	328975	2099	3099	112075	1951	
63639	98.1	346999	1932	3594	113270	1952	
64989	99.0	365385	1870	3547	115094	1953	
63761	100.0	363112	3578	3350	116219	1954	
66019	101.2	397469	2904	3048	117388	1955	
67857	104.6	419180	2822	2857	118734	1956	
68169	108.4	442769	2936	2798	120445	1957	
66513	110.8	444546	4681	2637	121950	1958	
68655	112.6	482704	3813	2552	123366	1959	
69564	114.2	502601	3931	2514	125368	1960	
69331	115.7	518173	4806	2572	127852	1961	
70551	116.9	554894	4007	2827	130081	1962	

Appendix
~application to big data~

$$(X'X)\beta = X'y$$

- Let X be n x p matrix, then (X'X) is a p x p matrix. Solving linear regression problem requires
  - O(p^3 + n^2 p) computation time
  - O(p^2) memory
- If we deal with genomic data where p is more than millions, then the above solution is infeasible.

Gradient Descent for Least Squares

$$min_{\beta} \| X\beta - y \|_2^2$$

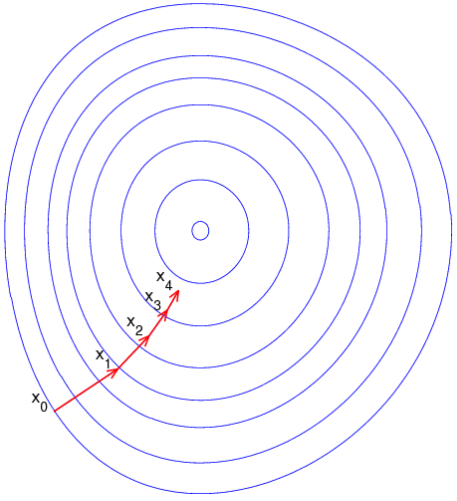
- Remember that if $L = (X\beta - y)'(X\beta - y)$, then

$$\frac{\partial L}{\partial \beta} = 2X'(X\beta - y)$$

- We can update β to the direction of descending gradient

$$\beta \leftarrow \beta - \epsilon X'(X\beta - y)$$

- Until convergence



Step size selection

- Selection of step size is critical in gradient descent.
  - if too small, it takes too much time until convergence.
  - if too large, it may overshoot the optimal solution.
- Step size search methods such as “line search” exists, but complicated step size search method may slow down the overall procedure.

Gradient Descent for Least Squares

lsgd.m

```
function [b]=lsgd(X,y)
e = 0.07; % step size
[n p] = size(X);
b = zeros(p,1);

for i=1:10000
 grad = X'*(X*b-y);
 b_old = b;
 b = b - e*grad/n;
 if norm(b-b_old) < 10e-10, break, end
end
```

- X and y must be scaled and centered !

Stochastic Gradient Descent

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \epsilon \boldsymbol{X}'(\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y})$$

- Instead of updating all the coefficients, pick up only one data point x_j from X, then update one coefficient each time.

$$\beta \leftarrow \beta - \epsilon \boldsymbol{x}_j'(\boldsymbol{x}_j\boldsymbol{\beta} - y_j)$$

- Requirements for one iteration
  - O(p) computation time
  - O(p) memory

Stochastic Gradient Descent (SGD) for Least Squares

lssgd.m

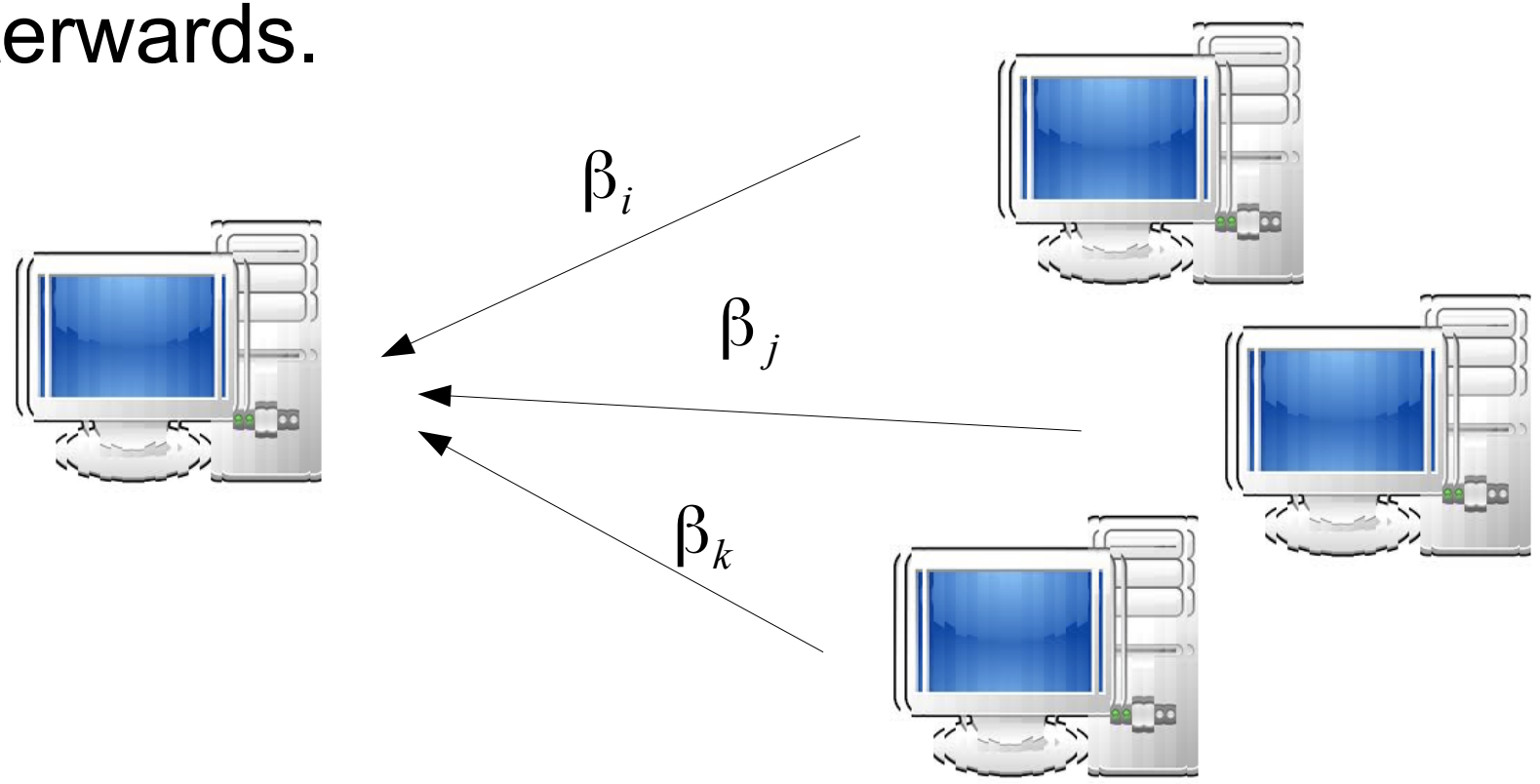
```
function [b]=lssgd(X,y)
e = 0.07;
[n p] = size(X);
b = zeros(p,1);

for i=1:10000
 r = ceil(rand*n);
 x = X(r,:); % randomly select one row
 grad = x'*(x*b-y(r));
 b_old = b;
 b = b - e*grad/n;
 if norm(b-b_old) < 10e-10, break, end
end
```

- X and y must be scaled and centered !

  x_ij ← (x_ij-mean(x_ij))./std(x_j)

SGD for MPI and MapReduce

- In a distributed computing environment such as Message Passing Interface or MapReduce, we can assign individual gradient computation to each CPU node, then summarize the results afterwards.



Newton's method for Least Squares

$$min_{\beta} \|X\beta - y\|_2^2$$

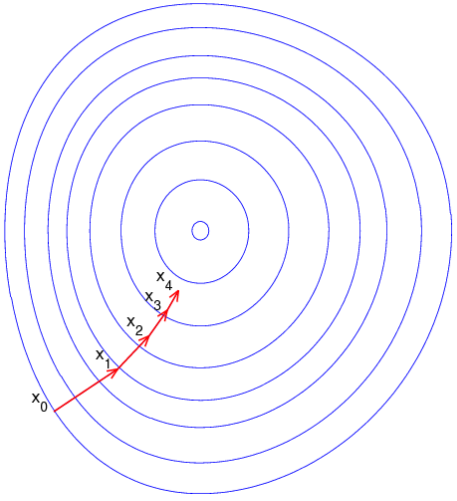
- Remember that if $L = (X\beta - y)'(X\beta - y)$, then

$$\frac{\partial L}{\partial \beta} = 2X'(X\beta - y) \qquad \frac{\partial L^2}{\partial^2 \beta} = 2X'X$$

- Update β to the direction of descending gradient

$$\beta \leftarrow \beta - \left(\frac{\partial L^2}{\partial^2 \beta}\right)^{-1} \frac{\partial L}{\partial \beta} = \beta - (X'X)^{-1} X'(X\beta - y)$$

- Until convergence



SGD is a key to solving large optimization problems such as Deep Learning

- Since 2nd derivative is too expensive for problems with large number of variables.
  - Assuming n x p design matrix, its first derivative is p x 1, but its 2nd derivative is p x p.
- Variants of SGDs
  - Momentum, Averaging, AdaGrad, RMSProp, Adam ..