機械学習特論

~ 理論とアルゴリズム ~

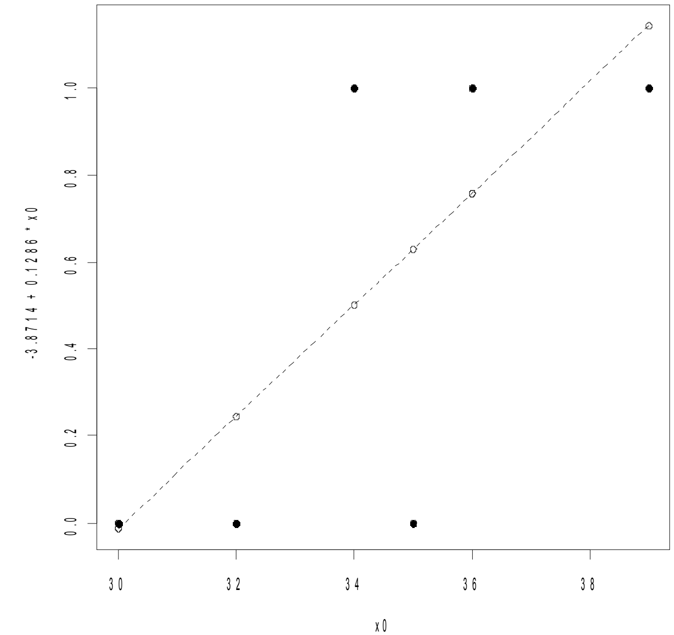
(Logistic Regression)

講師：西郷浩人

Ordinary Least Squares Regression vs Logistic Regression (LR)

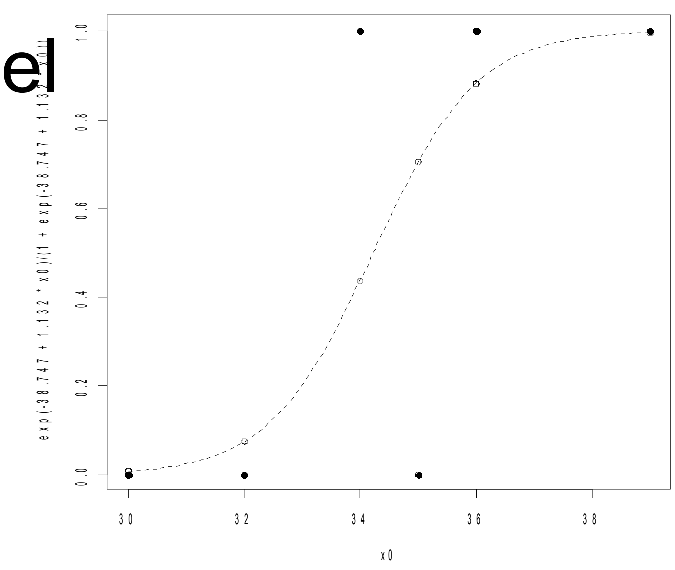
- OLS
  - a.k.a. Linear model

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



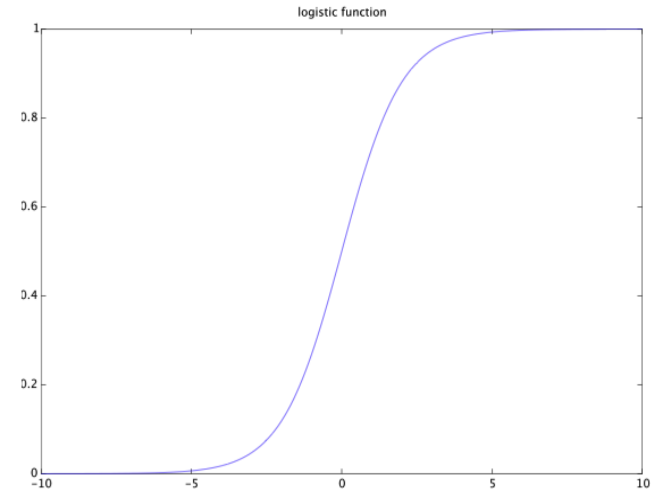
- Logistic Regression
  - a.k.a. Generalized linear model

$$f(\boldsymbol{x}) = \exp\left(\beta_0 + \sum_{j=i}^{p} \beta_j x_j\right)$$



Logistic Regression (LR)

- Logistic function maps linear function into a logistic form



$$h(\boldsymbol{x}) = \frac{\exp(-\boldsymbol{x})}{1+\exp(-\boldsymbol{x})}$$

- Posterior probabilities

$$p(y_i=0|\boldsymbol{x}) = h(\boldsymbol{\beta}'\boldsymbol{x}) = \frac{\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)}{1+\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)}$$

$$p(y_i=1|\boldsymbol{x}) = 1-h(\boldsymbol{\beta}'\boldsymbol{x}) = 1-\frac{\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)}{1+\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)} = \frac{1}{1+\exp(-\beta_0 - \sum_{j=1}^{p}\beta_j x_j)}$$

Logistic Regression (LR)

$$p(y_i=0|\boldsymbol{x})=h(\boldsymbol{\beta}'\boldsymbol{x})=\frac{\exp(-\beta_0-\sum_{j=1}^{p}\beta_j x_j)}{1+\exp(-\beta_0-\sum_{j=1}^{p}\beta_j x_j)}$$

$$p(y_i=1|\boldsymbol{x})=1-h(\boldsymbol{\beta}'\boldsymbol{x})=1-\frac{\exp(-\beta_0-\sum_{j=1}^{p}\beta_j x_j)}{1+\exp(-\beta_0-\sum_{j=1}^{p}\beta_j x_j)}=\frac{1}{1+\exp(-\beta_0-\sum_{j=1}^{p}\beta_j x_j)}$$

- Decision function is built by taking the ratio

$$f(\boldsymbol{x})=\frac{p(y_i=1|\boldsymbol{x})}{p(y_i=0|\boldsymbol{x})}=\exp(\beta_0+\sum_{j=1}^{p}\beta_j x_j)$$

  - which has a linear function in an exponential form. (so-called generalized linear form)
  - With this, we can perform 2-class classification by either if f(x) > 1 or f(x) < 1.

Solving logistic regression

$$p(y_i=0|\boldsymbol{x})=h(\boldsymbol{\beta}'\boldsymbol{x})=\frac{\exp(-\sum_{j=1}^{p}\beta_j x_j)}{1+\exp(-\sum_{j=1}^{p}\beta_j x_j)}$$

$$p(y_i=1|\boldsymbol{x})=1-h(\boldsymbol{\beta}'\boldsymbol{x})=\frac{1}{1+\exp(-\sum_{j=1}^{p}\beta_j x_j)}$$

Forget $\boldsymbol{\beta}_0$ for a moment

- Define likelihood function as

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

Below we denote by $\boldsymbol{x_i}$ a column vector containing features of the i-th example.

- Consider maximizing log-likelihood l(θ)

$$l(\boldsymbol{\beta})=\log L(\boldsymbol{\beta})$$

$$=\sum_{i=1}^{n}\{y_i\log p(y_i=1|\boldsymbol{x_i})+(1-y_i)\log p(y_i=0|\boldsymbol{x_i})\}$$

$y_i=0,1$

$$=\sum_{i=1}^{n}\{y_i\log h(\boldsymbol{\beta}'\boldsymbol{x_i})+(1-y_i)\log(1-h(\boldsymbol{\beta}'\boldsymbol{x_i}))\}$$

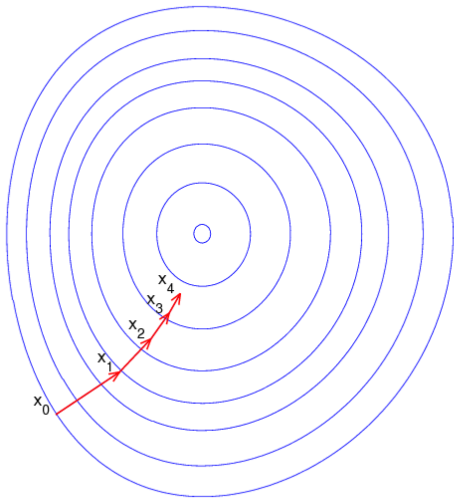
$$=\sum_{i=1}^{n}\{y_i(\log h(\boldsymbol{\beta}'\boldsymbol{x_i})-\log(1-h(\boldsymbol{\beta}'\boldsymbol{x_i})))+\log(1-h(\boldsymbol{\beta}'\boldsymbol{x}))\}$$

$$=\sum_{i=1}^{n}\{y_i\boldsymbol{\beta}'\boldsymbol{x_i}-\log(1+\exp(\boldsymbol{\beta}'\boldsymbol{x_i}))\}$$

$$\log\left(\frac{h(\boldsymbol{\beta}'\boldsymbol{x})}{1-h(\boldsymbol{\beta}'\boldsymbol{x})}\right)=\boldsymbol{\beta}'\boldsymbol{x_i}$$

$$1-h(\boldsymbol{\beta}'\boldsymbol{x})=\frac{1}{1+\exp(-\sum_{j=1}^{p}\beta_j x_j)}$$

Searching for a minimum/maximum of a function: Gradient Descent



$$x \leftarrow x - \epsilon \frac{df(x)}{dx}$$

- A simple idea is to use the **first derivative** of the objective function.
- Step size ε needs to be determined via line search.
- Slow convergence.

Deriving the first derivative

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x_i} - \log(1 + \exp(\boldsymbol{\beta}' \boldsymbol{x_i})) \}$$

- The above objective function is convex, and the first derivative w.r.t. β is

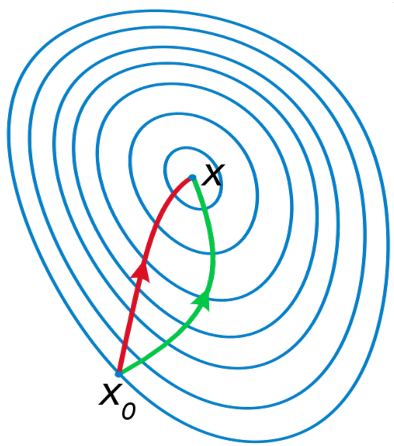
$$\frac{\partial l}{\partial \boldsymbol{\beta}} = \frac{\partial}{\partial \boldsymbol{\beta}} \sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x_i} - \log(1 + \exp(\boldsymbol{\beta}' \boldsymbol{x_i})) \}$$
$$= \sum_{i=1}^{n} \boldsymbol{x_i} (y_i - h(\boldsymbol{\beta}' \boldsymbol{x_i}))$$
$$= \boldsymbol{X}' (\boldsymbol{y} - h(\boldsymbol{X} \boldsymbol{\beta}))$$

  - which again is a function of β. So we cannot solve it directly in an analytical form.

- Optimization such as Newton's method or gradient descent is required.

Searching for a minimum/maximum of a function: Newton's method

$$x \leftarrow x - \frac{\frac{df(x)}{dx}}{\frac{d^2 f(x)}{dx^2}}$$



- **Both the first derivative and the second derivative** are used for searching the direction.
- Faster convergence.

Deriving the second derivative

$$\frac{\partial l}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \boldsymbol{x_i}(y_i - h(\boldsymbol{\beta}'\boldsymbol{x_i}))$$

- In order to use newton's method, we derive the second derivative from the first derivative.

$$\frac{\partial l^2}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}} == \frac{\partial}{\partial \boldsymbol{\beta}} \{\sum_{i=1}^{n} -\boldsymbol{x_i}(y_i - \frac{\exp(\boldsymbol{\beta}'\boldsymbol{x_i})}{1+\exp(\boldsymbol{\beta}'\boldsymbol{x_i})})\}$$

$$= \sum_{i=1}^{n} \left(\frac{-\boldsymbol{x_i}'\boldsymbol{x_i} \exp(\boldsymbol{\beta}'\boldsymbol{x_i})}{(1+\exp(\boldsymbol{\beta}'\boldsymbol{x_i}))^2} \right)$$

- A matrix storing the second derivatives is called *Hessian*.

$$\boldsymbol{H} = \sum_{i=1}^{n} h(\boldsymbol{x_i}'\boldsymbol{\beta})(1 - h(\boldsymbol{x_i}'\boldsymbol{\beta}))\boldsymbol{x_i}\boldsymbol{x_i}' = \boldsymbol{X}'\boldsymbol{W}\boldsymbol{X}$$

  – where W is a diagonal matrix with entries

$$h(\boldsymbol{x_i}'\boldsymbol{\beta})(1 - h(\boldsymbol{x_i}'\boldsymbol{\beta})) = \frac{\exp(\boldsymbol{\beta}'\boldsymbol{x_i})}{(1+\exp(\boldsymbol{\beta}'\boldsymbol{x_i}))^2}$$

Newton's method in our case

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\beta}} J$$

- Let our objective function to minimize be J=-logL, then

$$J(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \{ y_i \log h(\boldsymbol{\beta}' \boldsymbol{x}_i) + (1 - y_i) \log (1 - h(\boldsymbol{\beta}' \boldsymbol{x}_i)) \}$$
$$= -(\boldsymbol{y}' \log h(\boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{1} - \boldsymbol{y})'(\boldsymbol{1} - \log h(\boldsymbol{X}\boldsymbol{\beta})))$$
$$= -\sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x}_i - \log (1 + \exp(\boldsymbol{\beta}' \boldsymbol{x}_i)) \}$$

- The first derivative is a vector:

$$\nabla_{\boldsymbol{\beta}} J = \frac{-\partial l}{\partial \boldsymbol{\beta}} = \boldsymbol{X}'(h(\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{y})$$

- The second derivative is a matrix

$$\boldsymbol{H} = \sum_{i=1}^{n} h(\boldsymbol{x}_i' \boldsymbol{\beta})(1 - h(\boldsymbol{x}_i' \boldsymbol{\beta})) \boldsymbol{x}_i \boldsymbol{x}_i' = \boldsymbol{X}' \boldsymbol{W} \boldsymbol{X}$$

where W is a diagonal matrix with entries

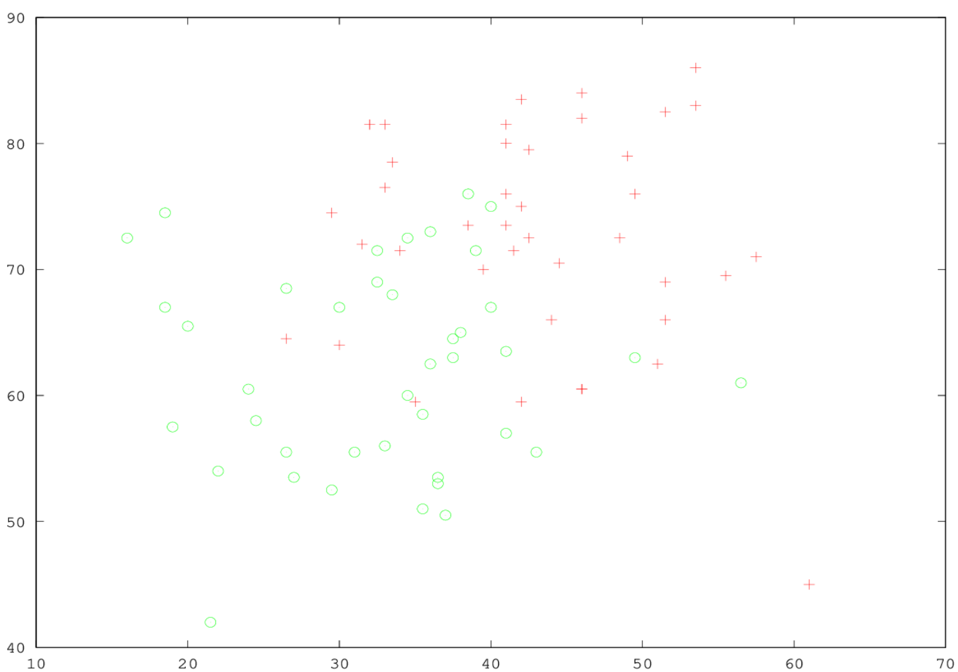
$$h(\boldsymbol{x}_i' \boldsymbol{\beta})(1 - h(\boldsymbol{x}_i' \boldsymbol{\beta}))$$

Algorithm1: Logistic Regression

- Input:
  - X: n x p data matrix
  - y: n x 1 binary response vector
- Output
  - β: p x 1 coefficient vector
- Initialize
  - J=0; prev_J = LARGE_NUMBER;
- Repeat
  - Compute J $\quad J(\boldsymbol{\beta}) = \sum_{i=1}^{n} \{ y_i \log p(y_i = 1 | \boldsymbol{x_i}) + (1 - y_i) \log p(y_i = 0 | \boldsymbol{x_i}) \}$
  - If prev_J – J < SMALL_NUMBER
    - Break
  - Compute nabla_J $\quad \nabla_{\boldsymbol{\beta}} J = \boldsymbol{X}'(h(\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{y})$
  - Compute H $\quad \boldsymbol{W} = diag(h(\boldsymbol{X}\boldsymbol{\beta}).(\boldsymbol{1} - h(\boldsymbol{X}\boldsymbol{\beta}))) \quad \boldsymbol{H} = \boldsymbol{X}'\boldsymbol{W}\boldsymbol{X}$
  - Update β $\quad \boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\beta}} J$

Exercise 1: Implement Logistic Regression by Newton's method

- Download todays' data
- Executing init.m loads and plots data
- Complete logistic.m
- Logistic sigmoid function h(x) is already prepared.



Hint 1

- You can build a diagonal matrix with entries 1,2,3 by

```
diag([1,2,3])
```

- If correctly implemented, Newton's method will converge in 5 iterations with the following answer.
  - beta =[-16.37874 0.14834 0.15891]
  - J = 32.436

Hint 2

$$
\begin{aligned}
&J(\boldsymbol{\beta}) \\
&= -\left(\boldsymbol{y}'\log h(\boldsymbol{X}\boldsymbol{\beta}) + (\mathbf{1}-\boldsymbol{y})'\log(\mathbf{1}-h(\boldsymbol{X}\boldsymbol{\beta}))\right) \\
&= -\left(\boldsymbol{y}'\boldsymbol{X}\boldsymbol{\beta} - \sum(\log(1+\exp(\boldsymbol{X}\boldsymbol{\beta})))\right)
\end{aligned}
$$

```
J = -(y'*log(h(X*b)) + (ones(n,1)-y)'*log(ones(n,1)-h(X*b)))
  = -(y'*X*b - sum(log(1+exp(X*b))))
```

Exercise 1 ~continued~

- At decision boundary, following relashinship holds.

$$h(\boldsymbol{\beta}'\boldsymbol{x}) = 1 - h(\boldsymbol{\beta}'\boldsymbol{x})$$
$$\leftrightarrow \frac{1}{1+\exp(\boldsymbol{\beta}'\boldsymbol{x})} = \frac{\exp(\boldsymbol{\beta}'\boldsymbol{x})}{1+\exp(\boldsymbol{\beta}'\boldsymbol{x})}$$
$$\leftrightarrow \exp(\boldsymbol{\beta}'\boldsymbol{x}) = 1$$
$$\leftrightarrow \beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

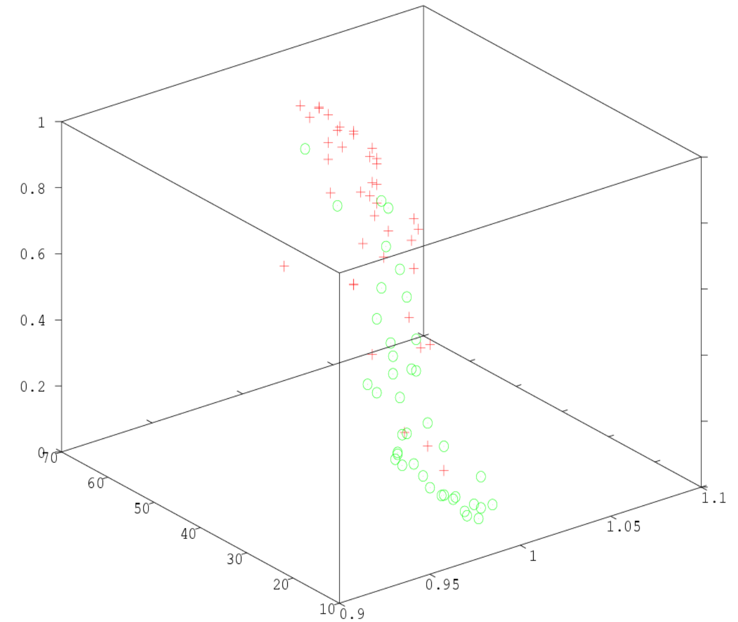
- Plot the decision boundary.
- Testing can be done with h(X*b), which returns a probability of one class.

p = h(X*b)
q = 1 - h(X*b)

Exercise 1 ~continued2~

- Using p, you can plot decision boundary in 3D !

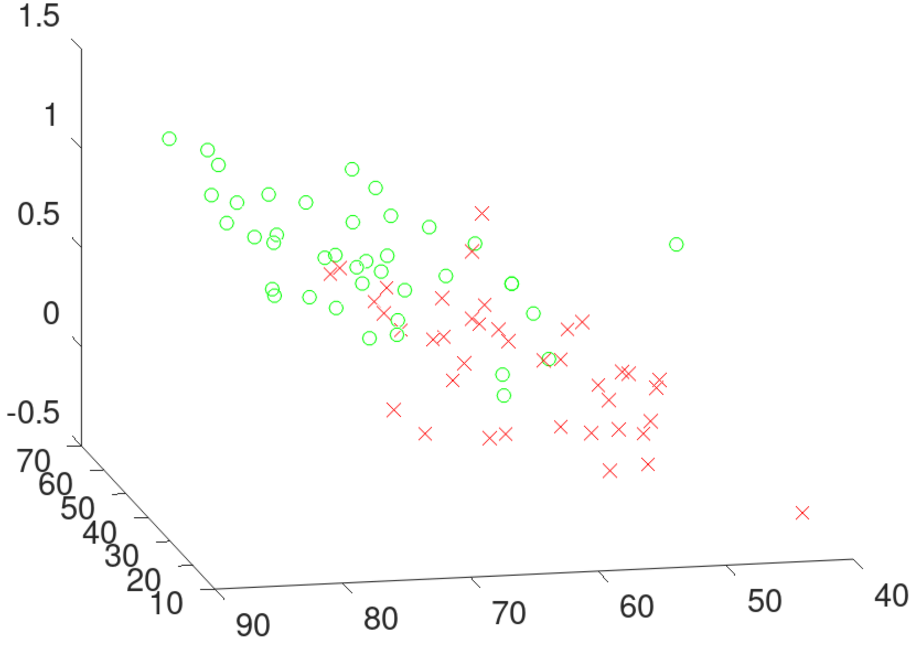
```
figure
plot3(X(pos,2),X(pos,3),p(pos),'go')
hold on
plot3(X(neg,2),X(neg,3),p(neg),'rx')
hold off
```



Exercise 1 ~continued3~

- Let's compare the results with that of OLS.

```
b_ols = X ¥ y
y_ols = X * b_ols
figure
plot3(X(pos,2),X(pos,3),y_ols(pos),'go')
hold on
plot3(X(neg,2),X(neg,3),y_ols(neg),'rx')
```



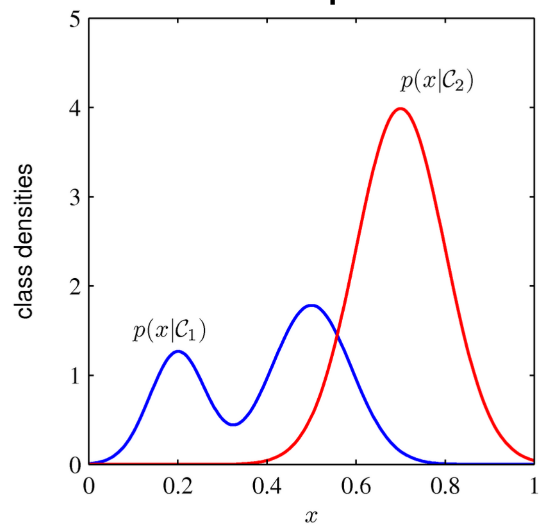
Generative vs Discriminative

- The goal is to estimate $p(y|x)$
- generative approach(生成的手法)

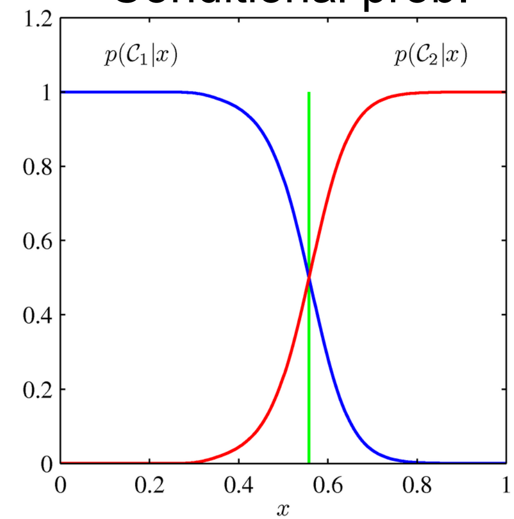
  first estimates joint probability $p(x,y)$
- then posterior $p(y|x)=\dfrac{p(x|y)\,p(y)}{p(x)}=\dfrac{p(x,y)}{p(x)}$
  - ex. LDA,QDA
- discriminative approach(識別的手法)

  directly estimates $p(y|x)$
  - **ex. Logistic Regression**

Joint prob.



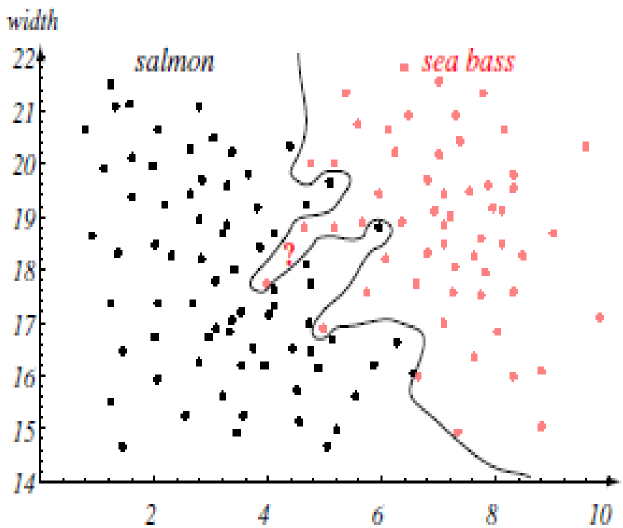
Conditional prob.



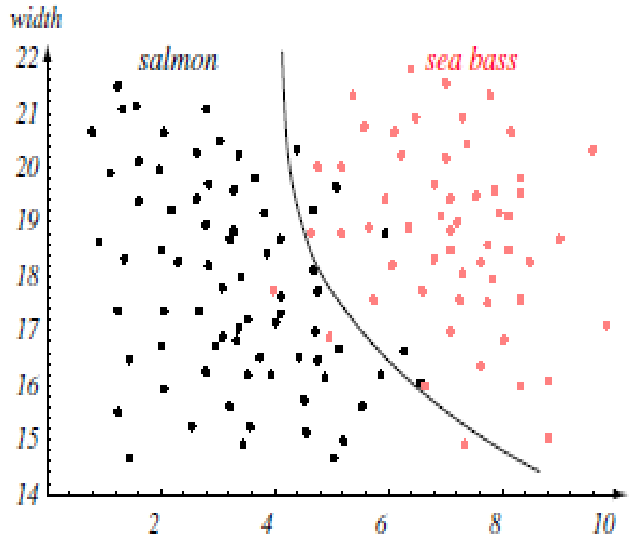
LDA vs Logistic Regression(LR)

- LDA assumes that marginal probability p(x,y) follows gaussian distribution, while LR does not have any assumption.
  - Since p(y|x) is modeled directly, we do not need to model p(x,y) in Logistic regression
- In situations where p(x,y) follows gaussian distribution, LDA would work well, but in other cases, LR would perform better.
- In terms of computational complexity, LDA requires inversing a covariance once, while LR require it in every iteration, therefore much more expensive.

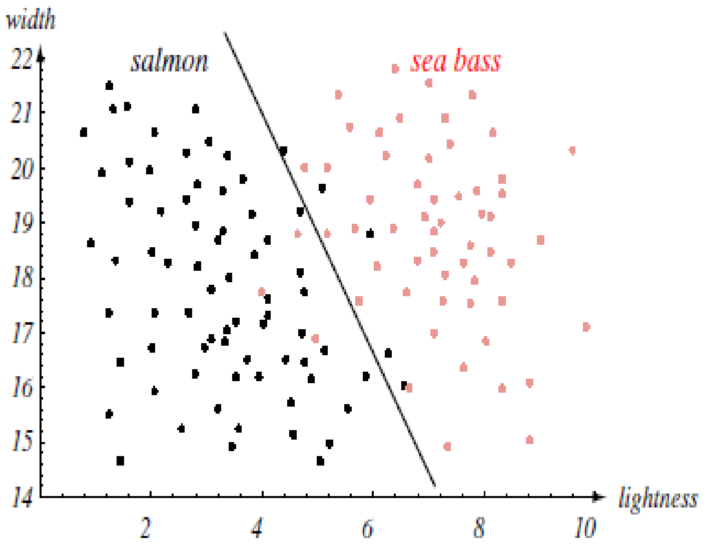
Regularization for classification



Overfitting



Optimal



Underfitting

Regularization for regression

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

- OLS

$$min_{\boldsymbol{\beta}} RSS$$

- Ridge regression (L2-regularization)

$$min_{\boldsymbol{\beta}} RSS + \lambda \| \boldsymbol{\beta} \|_2$$

- LASSO regression (L1-regularization)

$$min_{\boldsymbol{\beta}} RSS + \lambda \| \boldsymbol{\beta} \|_1$$

Regularization for classification

$$J = -l(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \{ y_i h(\boldsymbol{x_i}) + (1 - y_i) h(\mathbf{1} - \boldsymbol{x_i}) \}$$

- Logistic regression

$$min_{\boldsymbol{\beta}} \, J$$

- L2-regularized Logistic regression

$$min_{\boldsymbol{\beta}} \, J + \lambda \|\boldsymbol{\beta}\|_2$$

- L1-regularized Logistic regression

$$min_{\boldsymbol{\beta}} \, J + \lambda \|\boldsymbol{\beta}\|_1$$

Solving L2-Logistic Regression

$$min_{\boldsymbol{\beta}}\, J + \lambda \|\boldsymbol{\beta}\|_2$$

- Objective function

$$J = -l(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x}_i - \log(1 + \exp(\boldsymbol{\beta}' \boldsymbol{x}_i)) \} + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2$$

Deriving the first derivative

$$J = -l(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x_i} - \log(1 + \exp(\boldsymbol{\beta}' \boldsymbol{x_i})) \} + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2$$

- Same way as the ordinary logistic regression.

$$\nabla J = \frac{-\partial l}{\partial \beta_j} = \frac{-\partial}{\partial \beta_j} \left(\sum_{i=1}^{n} \{ y_i \boldsymbol{\beta}' \boldsymbol{x_i} - \log(1 + \exp(\boldsymbol{\beta}' \boldsymbol{x_i})) \} + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2 \right)$$
$$= -\sum_{i=1}^{n} \boldsymbol{x_i} (y_i - h(\boldsymbol{\beta}' \boldsymbol{x_i})) + \lambda \beta_j$$

- Vector form $\nabla_{\boldsymbol{\beta}} J = \boldsymbol{X}'(h(\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{y}) + \lambda \boldsymbol{\beta}$

Deriving the second derivative

$$\nabla J = \frac{-\partial l}{\partial \boldsymbol{\beta}} = -\boldsymbol{X}'(\boldsymbol{y} - h(\boldsymbol{X}\boldsymbol{\beta})) + \lambda \boldsymbol{\beta}$$

- Same way as the previous lecture.

$$\frac{\partial J}{\partial \beta^2} = \frac{-\partial l^2}{\partial \beta \partial \beta} = = \frac{-\partial}{\partial \beta}\left\{\sum_{i=1}^{n} -\boldsymbol{x_i}\left(1 - \frac{\exp(\boldsymbol{\beta}'\boldsymbol{x_i})}{1 + \exp(\boldsymbol{\beta}'\boldsymbol{x_i})}\right) + \lambda \beta_j\right\}$$

$$= -\sum_{i=1}^{n}\left(\frac{-\boldsymbol{x_i}'\boldsymbol{x_i}\exp(\boldsymbol{\beta}'\boldsymbol{x_i})}{(1 + \exp(\boldsymbol{\beta}'\boldsymbol{x_i}))^2}\right) + \lambda$$

$$\boldsymbol{H} = \sum_{i=1}^{n} h(\boldsymbol{x_i}'\boldsymbol{\beta})(1 - h(\boldsymbol{x_i}'\boldsymbol{\beta}))\boldsymbol{x_i}\boldsymbol{x_i}' + \lambda \begin{bmatrix} 1 & 0 & 0 & . \\ 0 & 1 & 0 & . \\ 0 & 0 & 1 & . \\ . & . & . & . \end{bmatrix} = \boldsymbol{X}'\boldsymbol{W}\boldsymbol{X} + \lambda \boldsymbol{I}$$

$$\boldsymbol{W} = diag(h(\boldsymbol{X}\boldsymbol{\beta})(\boldsymbol{1} - \boldsymbol{h}(\boldsymbol{X}\boldsymbol{\beta}))) = \begin{bmatrix} h(\boldsymbol{x_1}'\boldsymbol{\beta})(1 - h(\boldsymbol{x_1}'\boldsymbol{\beta})) & 0 & 0 & . \\ 0 & h(\boldsymbol{x_2}'\boldsymbol{\beta})(1 - h(\boldsymbol{x_2}'\boldsymbol{\beta})) & 0 & . \\ 0 & 0 & . & . \\ . & . & . & . \end{bmatrix}$$

Algorithm: L2-Logistic Regression

- Input:
  - X: n x p data matrix
  - y: n x 1 binary response vector
  - λ
- Output
  - β: p x 1 coefficient vector
- Initialize
  - J=0; prev_J = LARGE_NUMBER;
- Repeat
  - Compute J $J(\boldsymbol{\beta})=\sum_{i=1}^{n}\{y_i \log p(y_i=1|\boldsymbol{x}_i)+(1-y_i)\log p(y_i=0|\boldsymbol{x}_i)\}+\frac{\lambda}{2}\|\boldsymbol{\beta}\|_2$
  - If prev_J – J < SMALL_NUMBER
    - Break
  - Compute nabla_J $\nabla_{\beta} J=\boldsymbol{X}'(h(\boldsymbol{X}\boldsymbol{\beta})-\boldsymbol{y})+\lambda\boldsymbol{\beta}$
  - Compute H $\boldsymbol{W}=diag(h(\boldsymbol{X}\boldsymbol{\beta}).(\mathbf{1}-h(\boldsymbol{X}\boldsymbol{\beta})))$ $\boldsymbol{H}=\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X}+\lambda\boldsymbol{I}$
  - Update β $\boldsymbol{\beta}\leftarrow\boldsymbol{\beta}-\boldsymbol{H}^{-1}\nabla_{\beta}J$

Exercise 2

- Implement L2-logistic regression of the form
  - [b] = function logistic2(x,y,lambda)
- First verify that setting lambda = 0 obtains the same solution as before.
- Observe
  - How norm of b changes

```
norm(b)
```

  - How accuracy changes

```
sum((h(X*b)>0.5)==y)/length(y)
```

with respect to change in lambda.