DATAMINING B(2)

10-2 Linear Models 1

Linear Model

- It's natural way to handle numeric values.
- Regression
  - Numeric Prediction: Target is Numerical.
    - Linear Regression
  - Linear Classification: Target is Categorical
    - Logistic Regression

      0 / 1.
- Perceptron

Linear Regression

- It's simple.
- Target value is represented by a linear combination of attributes with pre-determined weights.

$$x = w_0 + w_1 a_1 + w_2 a_2 \cdots w_k a_k$$

where, x is a target value,

$a_1, a_2, \cdots a_k$ are attribute values

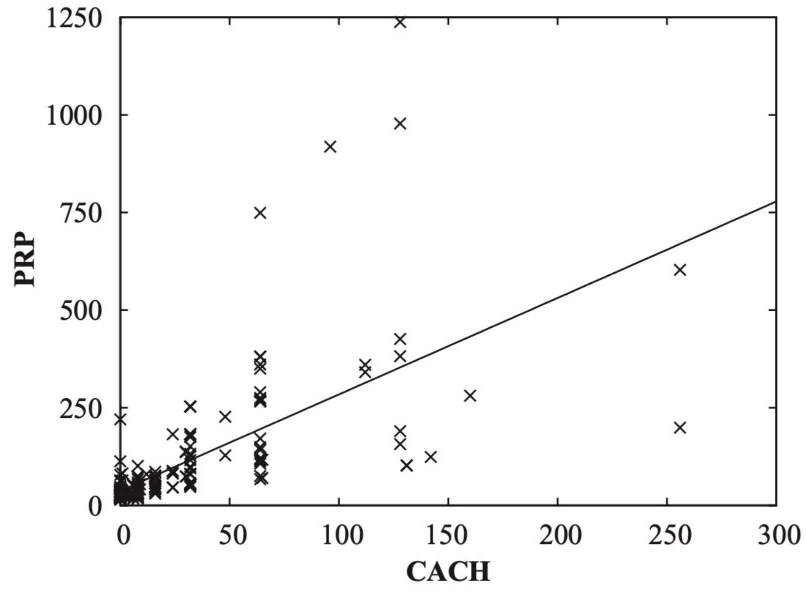
$w_0, w_1, w_2, \cdots w_k$ are weights.

Weights

- A set of attribute values and target values are given.
  (as a training data)
- How to derive weights?
  - Least-Squares Method
    - Minimize the error

$$\sum_{i=1}^{n}\left(x^{(i)}-\sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$

n is a number of instances.
k is a number of attributes.



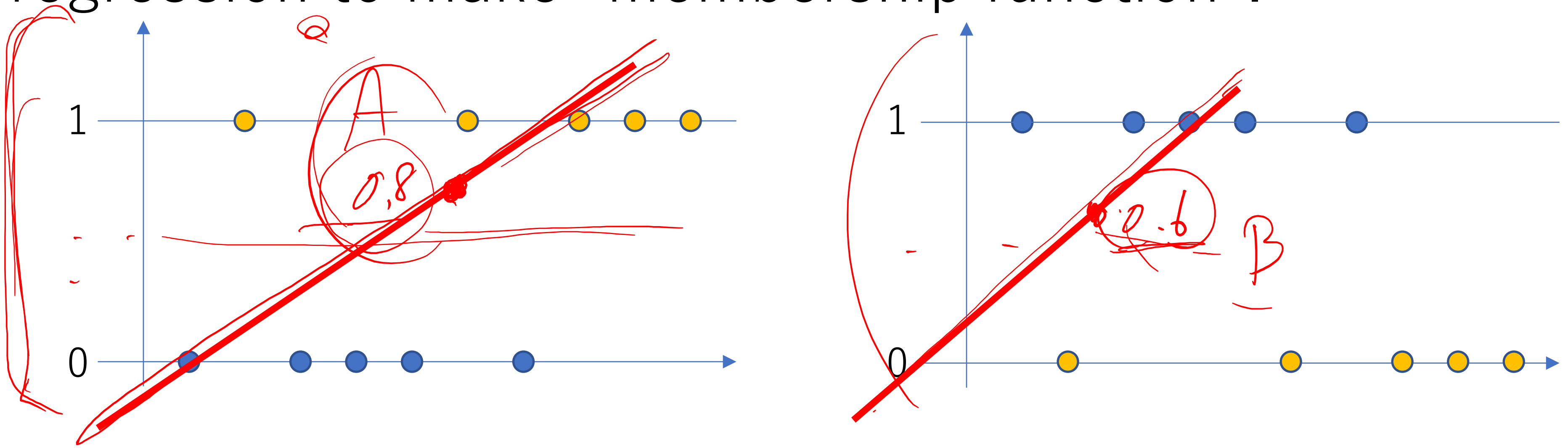
Advantage / Disadvantage

- It is excellent and simple way for numeric prediction.
- It has been used for decades.

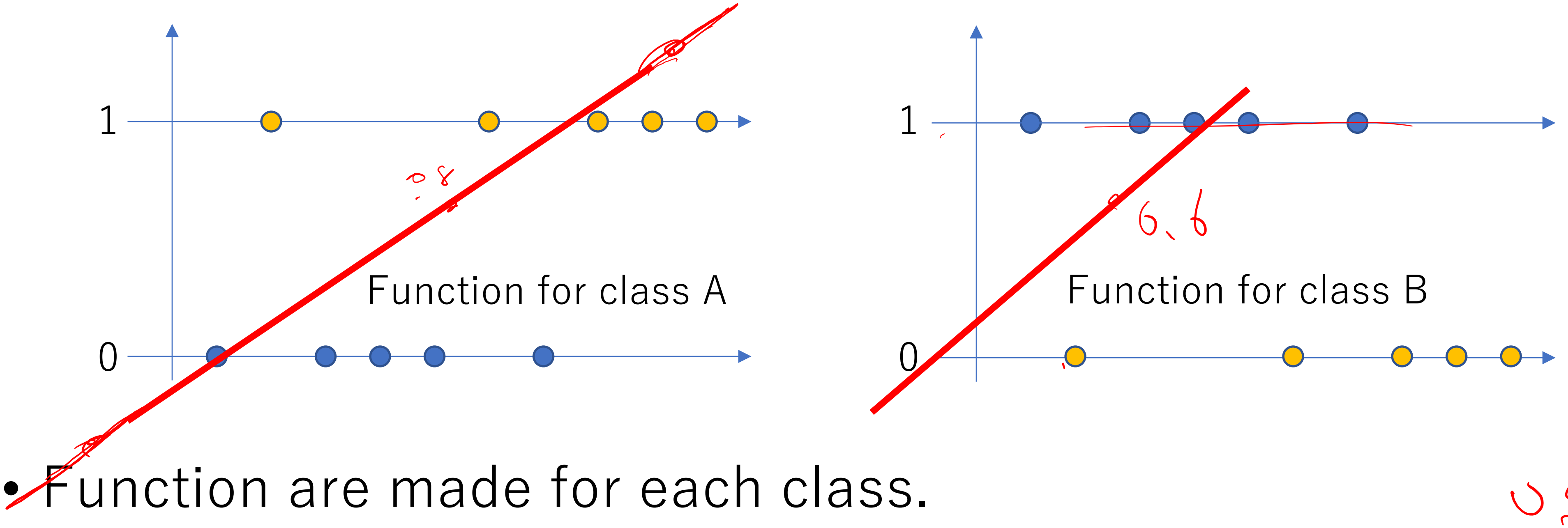
- For non-linear dependency, we will get best-fitting straight line, but it may not fit very well.
  - ”Best” means it is the least mean-squared error.

Multiresponse Linear Regression

- Used for classification.
- Suppose target classes are A or B, and try to predict by Linear Regression.
- Assign 1 for one of the class, and 0 for others, then conduct regression to make “membership function”.



Membership Function



$0 \leq p \leq 1.0$

- Function are made for each class.
  - Calculate output for unclassified data.
  - The data belongs to the class with maximum output of the function.
- Notice:
  - The output is not probability. Exceeds between 0 and 1.
  - Violating the assumption of least-square method. Error Distribution.

Logistic Regression

- Avoid the problem on Multiresponse Regression.

- Suppose two class (1/0) classification.
- Consider a probability $\mathbf{Pr}$ that the class is 1 when attribute values are $\{a_1, a_2, \cdots a_k\}$. Instead of the class, we will predict Pr.

$\boxed{\mathrm{Pr}[1|a_1, a_2, \cdots a_k]}$

- $\mathbf{Pr}$ can't be approximated by linear regression, so we consider log of the ratio of $\mathbf{Pr}$ and (1- $\mathbf{Pr}$), which are the probability of 1 and 0.
  - $\log[\underbrace{\boxed{\mathrm{Pr}[1|a_1, a_2, \cdots a_k]}}_{1}/\underbrace{\boxed{(1-\mathrm{Pr}[1|a_1, a_2, \cdots a_k])}}_{0}]$

Logistic Regression

1. $\Pr[1|a_1, a_2, \cdots a_k]$ ← 0 ~ 1
2. $\log[\Pr[1|a_1, a_2, \cdots a_k]/(1 - \Pr[1|a_1, a_2, \cdots a_k])]$

- 1. is probability, so the domain is between 1 to 0. It is not good for linear regression.
- 2. is not limited between 1 to 0. Maybe good for linear regression.
- It is called “logit transformation”. The ratio of P to (1-P) is called “odds”. The log of odds is called “logit”.
- We will apply regression for this logit. So,
  - $\log \frac{\Pr[1|a_1,a_2, \cdots a_k]}{(1-\Pr[1|a_1,a_2, \cdots a_k])} = w_0 + w_1 a_1 + w_2 a_2 \cdots w_k a_k$

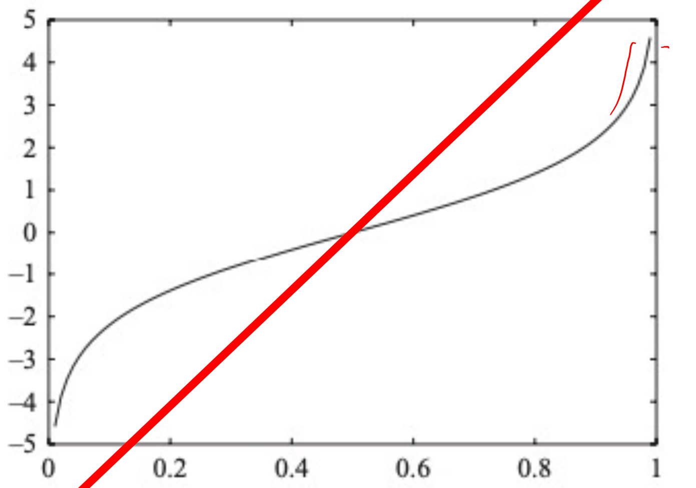


Fig: A plot of logit, function 2.

Logistic Regression

$\frac{1}{1+e^{-f}}$

$$\log \frac{\Pr[1|a_1,a_2,\cdots a_k]}{(1-\Pr[1|a_1,a_2,\cdots a_k])} = w_0 + w_1 a_1 + w_2 a_2 \cdots w_k a_k$$

- Transform above, we will get
  1. $\Pr[1|a_1, a_2, \cdots a_k] = {}^{1}/_{(1+e^{-(w_0+w_1a_1+w_2a_2\cdots w_ka_k)})}$

  Output of 1. exists between 1 and 0.
  It may be regarded as probability.

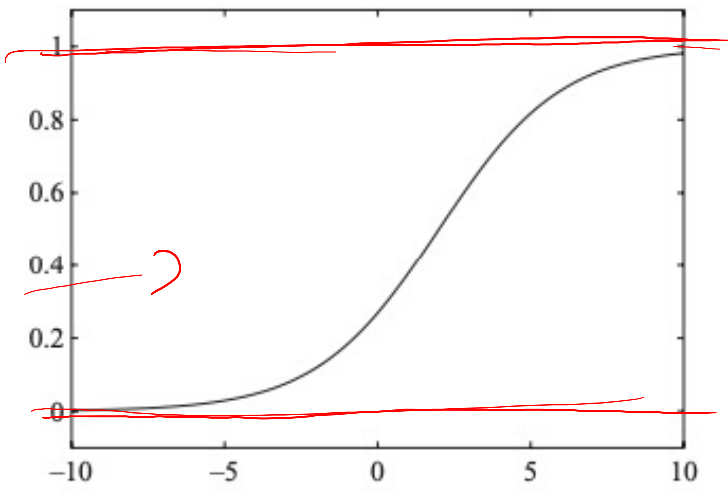


Fig: A plot of function 1.

- In regression, maximize log-likelihood as follows.

$$\sum_{i=1}^{n}\left[(1-x^{(i)})\log(1-\Pr[1|a_1^{(i)}\cdots a_k^{(i)}]) + x^{(i)}\log(\Pr[1|a_1^{(i)}\cdots a_k^{(i)}])\right]$$

対数ゆう度

Logistic Regression

- Classification border

$$\Pr(1|a) = \boxed{\frac{1}{(1+e^{-(w_0+w_1a_1+w_2a_2\cdots w_ka_k)})}} = 0.5$$

Therefore,

$$e \quad -(w_0 + w_1a_1 + w_2a_2 \cdots w_ka_k\ )=0$$

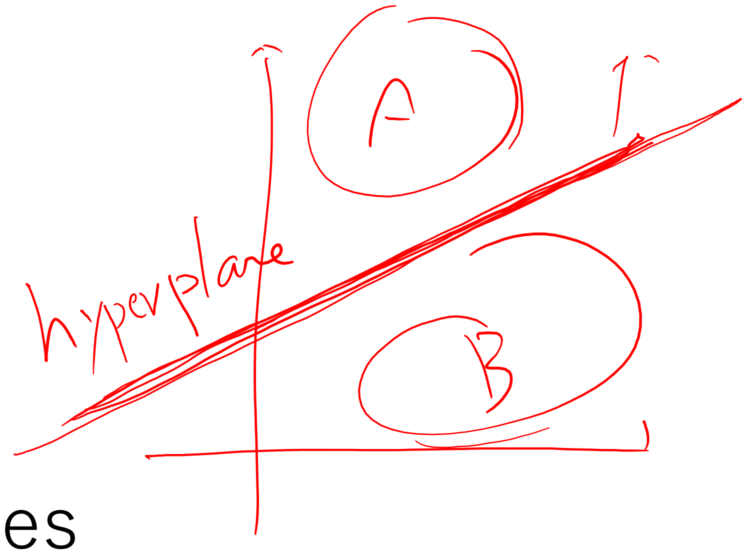
- It is hyperplane of the instance space. If boundary is non-linear, it is hard to distinguish by logistic regression.

Perceptron

- Accurate estimation of probability lead to accurate classification.
- Just for classification, it is enough to get hyperplane.
    - If boundary is linear.

- Perceptron is a simple way to obtain hyperplane.

Perceptron



- Hyperplane

$$w_0a_0 + w_1a_1 + w_2a_2 \cdots w_ka_k = 0$$

$a_1, a_2, \cdots a_k$ are attribute values

$w_0, w_1, w_2, \cdots w_k$ are weights for hyperplane.

a_0 is bias

- If $w_0a_0 + w_1a_1 + w_2a_2 \cdots w_ka_k > 0$, prediction class is 1, else class is 0.

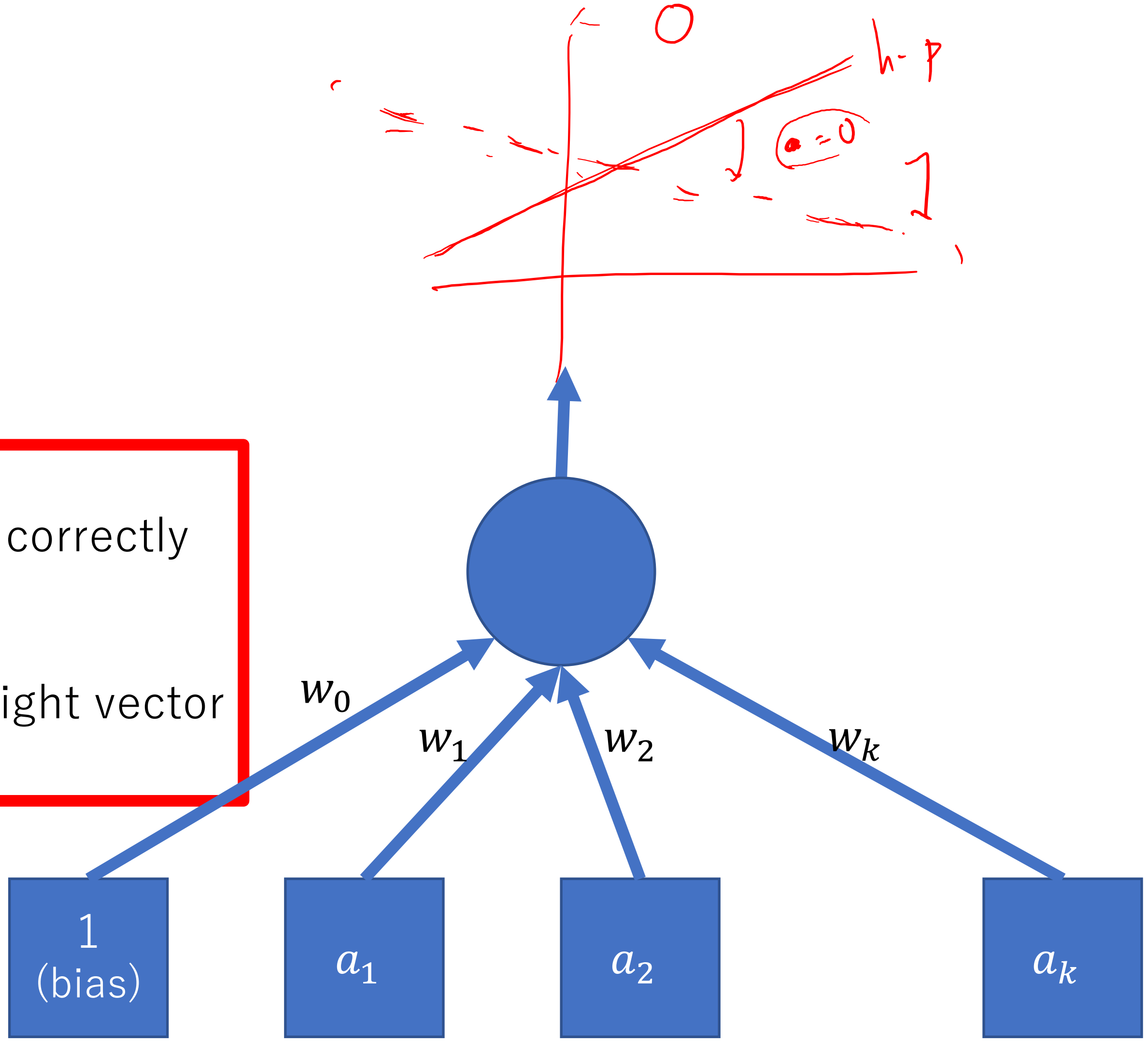
< 0

- We should adjust weights w_j to classify training data by hyperplane.

Perceptron Learning

- Learning Rule

Set all weights to zero
Until all instances in the training data are classified correctly
    For each instance I in the training data
        If I is classified incorrectly by the perceptron
            If I belongs to the first class add it to the weight vector
            else subtract it from the weight vector



If miss classified, addition is

$$(w_0 + a_0)a_0 + (w_1 + a_1)a_1 + \cdots + (w_k + a_k)a_k$$

Output increases

$$a_0 \cdot a_0 + a_1 \cdot a_1 + \cdots + a_k \cdot a_k$$

Hyperplane will move to toward the misclassified instance

The hyperplane obtained is called “perceptron”.

Winnow

- Other method to find a hyperplane.
- Algorithm is following

learnig prameter.

While some instances are misclassified
  for every instance a
    classify a using the current weights if the predicted class is incorrect
      if a belongs to the first class
        for each a_i that is 1, multiply w_i by α (if a_i is 0, leave w_i unchanged)
      otherwise
        for each a_i that is 1, divide w_i by α (if a_i is 0, leave w_i unchanged)
* α should be grater than 1. Initial value of w_i is constant (not 0).

- Boundary is defined by θ.

$$w_0a_0 + w_1a_1 + w_2a_2 \cdots w_ka_k > \theta$$

Balanced Winnow

- Weights of Winnow is only positive. Sometimes it is drawback.
- Balanced Winnow uses positive weight and negative weight. Algorithm is following.

While some instances are misclassified
    for every instance a
        classify a using the current weights
        if the predicted class is incorrect
            if a belongs to the first class
                for each a_i that is 1,
                    multiply w_i^+ by α
                    divide wi$^-$ by α
                (if a_i is 0, leave w_i^+ and w_i^- unchanged)
            otherwise
                    multiply w_i^- by α
                    divide w_i^+ by α
                (if a_i is 0, leave w_i^+ and w_i^- unchanged)

Distinction Function is

$$(w_0^+ - w_0^-)a_0 + (w_1^+ - w_1^-)a_1 + \cdots + (w_k^+ - w_k^-)a_k > \theta$$

Perceptron, Winnow

- They are good algorithms if dataset has many features and most of them are irrelevant.

- Learning algorithm is simple.
- It is able to additional learning.
  - Online learning. Incrementally learning.
  - Append training data after learning