Introduction to Artificial Neural Networks

Lecture 3:

Pattern Recognition and Function Approximation

By: Ali Motie Nasrabadi

Outline

- What is pattern recognition?
- Features and patterns
- Classifiers
- The pattern recognition design cycle
- Distance Metrics
- Linear Discriminant Functions
- Linear Regression
- Minimum Squared Error solution
- Gradient descending
- Example

What is pattern recognition?

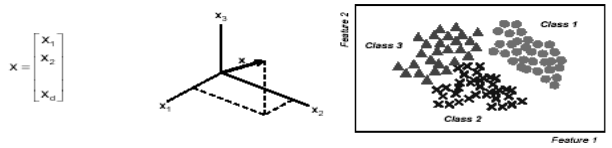
- Definitions from the literature
- "The assignment of a physical object or event to one of several pre-specified categories" –Duda and Hart
- A problem of estimating density functions in a high-dimensional pace and dividing the space into the regions of categories or classes" – Fukunaga
- Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples" –Ripley
- The science that concerns the description or classification (recognition) of measurements" – Schalkoff
- Pattern Recognition is concerned with answering the question "What is this? –Morse

What is pattern recognition?

- According to R. Beale & T. Jackson, Neural Computing : An *Introduction, 1990:*
  - The fundamental objective for pattern recognition is classfication : given an input of some form can we analyze that input to provide a meaningful categorization of its data content?
- A pattern regonition system ca be considered as a two stage device:
  - Feature extraction
  - Classification

Features and patterns (1)

- **Feature**
  - **Feature is any distinctive aspect, quality or characteristic**
    - **Features may be symbolic (i.e., color) or numeric (i.e., height)**
  - **Definitions**
    - **The combination of d features is represented as a d-dimensional column vector called a feature vector**
    - **The d-dimensional space defined by the feature vector is called the feature space**
    - **objects are represented as points in feature space. This representation is called a scatter plot**



Feature vector Feature space (3D) Scatter plot (2D)

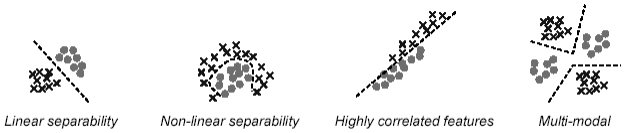
Features and patterns (2)

- **Pattern**
  - **Pattern is a composite of traits or features characteristic of an individual**
  - **In classification tasks, a pattern is a pair of variables {x,w} where**
    - **x is a collection of observations or features (feature vector)**
    - **w is the concept behind the observation (label)**

Features and patterns (3)

- **What makes a "good" feature vector?**
  - **The quality of a feature vector is related to its ability to discriminate examples from different classes**
    - **Examples from the same class should have similar feature values**
    - **Examples from different classes have different feature values**

- **More feature properties**

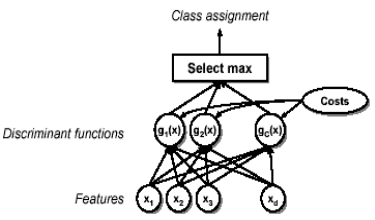


Lecture 3-7

Classifiers

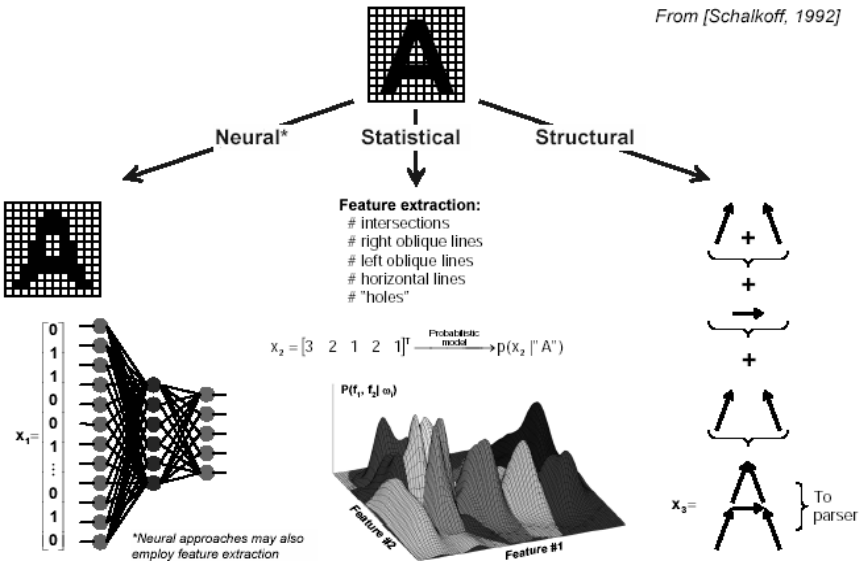
- **The task of a classifier is to partition feature space into class-labeled decision regions**
  - **Borders between decision regions are called decision boundaries**
  - **The classification of feature vector x consists of determining which decision region it belongs to, and assign x to this class**
- **A classifier can be represented as a set of discriminant functions**
  - **The classifier assigns a feature vector x to class w if**

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i$$



Lecture 3-8

Example: neural, statistical and structural OCR

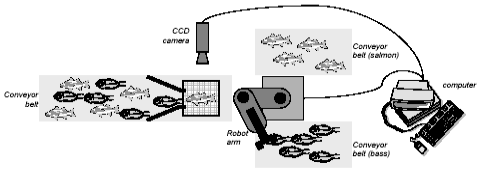


The pattern recognition design cycle (1)

- **Data collection**
  - **Probably the most time-intensive component of a PR project**
  - **How many examples are enough?**
- **Feature choice**
  - **Critical to the success of the PR problem**
  - **Requires basic prior knowledge**
- **Model choice**
  - **Statistical, neural and structural approaches**
  - **Parameter settings**
- **Training**
  - **Given a feature set and a "blank" model, adapt the model to explain the data**
- **Evaluation**
  - **How well does the trained model do?**
  - **Overfitting (Memorization) vs. generalization**

The pattern recognition design cycle (2)

- Consider the following scenario
  - A fish processing plan wants to automate the process of sorting incoming fish according to species (salmon or sea bass)
- The automation system consists of
  - a conveyor belt for incoming products
  - two conveyor belts for sorted products
  - a pick-and-place robotic arm
  - a vision system with an overhead CCD camera
  - a computer to analyze images and control the robot arm

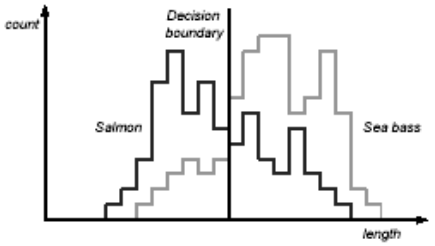


The pattern recognition design cycle (3)

- Sensor
  - The vision system captures an image as a new fish enters the sorting area
- Preprocessing
  - Image processing algorithms
    - adjustments for average intensity levels
    - segmentation to separate fish from background
  - Feature Extraction
    - Suppose we know that, on the average, sea bass is larger than salmon
    - From the segmented image we estimate the length of the fish

The pattern recognition design cycle (4)

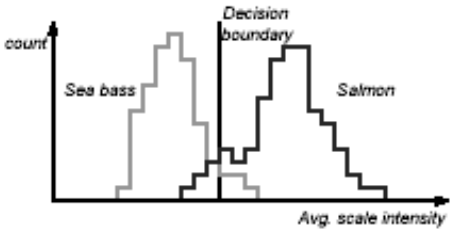
- **Classification**
  - **Collect a set of examples from both species**
  - **Compute the distribution of lengths for both classes**
  - **Determine a decision boundary (threshold) that minimizes the classification error**
  - **We estimate the classifier's probability of error and obtain a discouraging result of 40%**
  - **What do we do now?**



re 3-13

The pattern recognition design cycle (5)

- **Improving the performance of our PR system**
  - **Determined to achieve a recognition rate of 95%, we try a number of features**
    - **Width, Area, Position of the eyes w.r.t. mouth...**
    - **only to find out that these features contain no discriminatory information**
  - **Finally we find a "good" feature: average intensity of the scales**

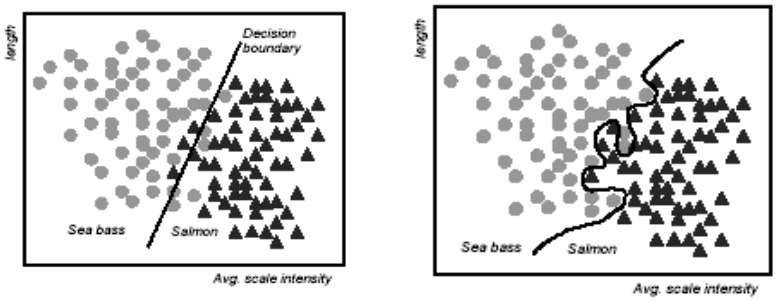


Lecture 3-14

The pattern recognition design cycle (6)

- We combine "length" and "average intensity of the scales" to improve class separability
- We compute a linear discriminant function to separate the two classes, and obtain a classification rate of 95.7%

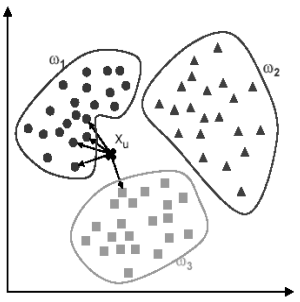
Linear **Nonlinear**



5

The k Nearest Neighbor classification rule

- The K Nearest Neighbor Rule (kNN) is a very intuitive method that classifies unlabeled examples based on their similarity to examples in the training set
  - For a given unlabeled example $x_u \in \Re^D$, find the k "closest" labeled examples in the training data set and assign x_u to the class that appears most frequently within the k-subset
- The kNN only requires
  - An integer k
  - A set of labeled examples (training data)
  - A metric to measure "closeness"
- Example
  - In the example below we have three classes and the goal is to find a class label for the unknown example x_u
  - In this case we use the Euclidean distance and a value of k=5 neighbors
  - Of the 5 closest neighbors, 4 belong to ω_1 and 1 belongs to ω_3, so x_u is assigned to ω_1, the predominant class



Lecture 3-16

Distance Metrics(1)

- **Nearest neighbor methods need to find a reliable way of measuring the distance from one class sample to another**
- **Distance metric measures the similarity of pattern samples in the geometric pattern space. For two vectors**

  X:(x1,x2,…xn) , Y:(y1,y2,…,yn)

  - **Hammining distance :** $d(X,Y)_{ham} = \sum_{i=1}^{n} | x_i - y_i |$
    - **This distance is often used to compare binary vectors**
  - **Euclidean distance :** $d(x,y)_{euc} = \sqrt{(\sum_{i=1}^{n} (x_i - y_i)^2)}$
  - **City block distance:** $d(X,Y)_{cb} = \sum_{i=1}^{n} | x_i - y_i |$
  - **Square distance :** $d(X,Y)_{sd} = Max | x_i - y_i |$

Distance Metrics(2)

Mahalanobis distance

Def. The euclidian distance between two points $x = (x_1, \ldots, x_p)^t$ and $y = (y_1, \ldots, y_p)^t$ in the p-dimensional space $\mathbb{R}^p$ is defined as

$$d_E(x,y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2} = \sqrt{(x-y)^t (x-y)}$$

and $d_E(x,0) = \|x\|_2 = \sqrt{x_1^2 + \cdots + x_p^2} = \sqrt{x^t x}$ is the euclidian norm of x.

Denote

$$u = (\frac{x_1}{s_1}, \ldots, \frac{x_p}{s_p}) \text{ and } v = (\frac{y_1}{s_1}, \ldots, \frac{y_p}{s_p})$$

then define the distance between x and y as

$$d(x,y) = d_E(u,v) = \sqrt{(\frac{x_1 - y_1}{s_1})^2 + \cdots + (\frac{x_p - y_p}{s_p})^2} = \sqrt{(x-y)^t D^{-1} (x-y)}$$

where $D = \text{diag}(s_1^2, \ldots, s_p^2)$. Now the norm of x equals

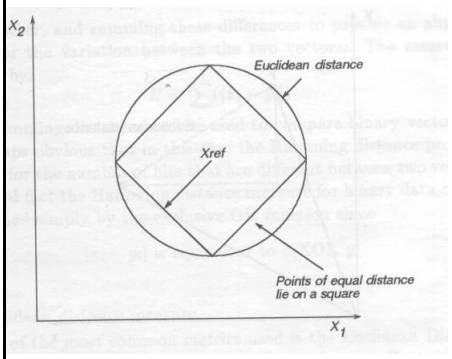
$$\|x\| = d(x,0) = d_E(u,0) = \|u\|_2 = \sqrt{(\frac{x_1}{s_1})^2 + \cdots + (\frac{x_p}{s_p})^2} = \sqrt{x^t D^{-1} x}$$

Def. The statistical distance or Mahalanobis distance between two points $x = (x_1, \ldots, x_p)^t$ and $y = (y_1, \ldots, y_p)^t$ in the p-dimensional space $\mathbb{R}^p$ is defined as

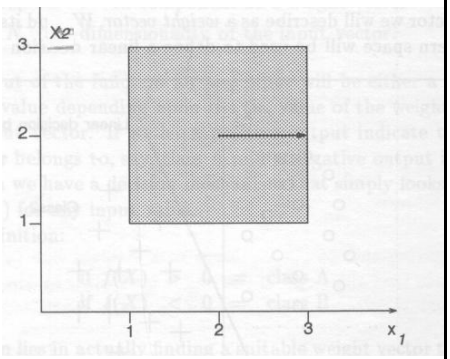
$$d_S(x,y) = \sqrt{(x-y)^t S^{-1} (x-y)}$$

and $d_S(x,0) = \|x\|_S = \sqrt{x^t S^{-1} x}$ is the norm of x.

Distance Metrics(3)



City Block Distance



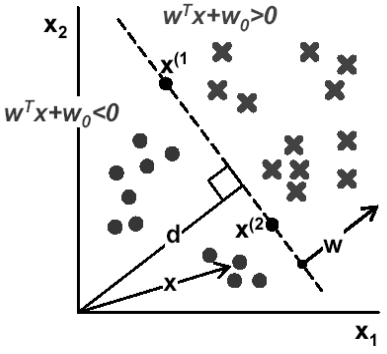
The Square Distance

Linear Discriminant Functions

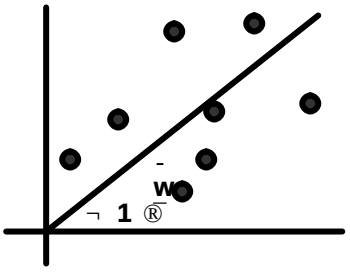
- The objective of this lecture is to present methods for learning linear discriminant functions of the form
  - where w is the weight vector and w0 is the threshold weight or bias

$$g(x) = w^T x + w_0 \Leftrightarrow \begin{cases} g(x) > 0 & x \in \omega_1 \\ g(x) < 0 & x \in \omega_2 \end{cases}$$

- The Decision boundary is a line (hyper plane) in feature space.



Linear Regression



DATASET

inputs	outputs
$x_1 = 1$	$y_1 = 1$
$x_2 = 3$	$y_2 = 2.2$
$x_3 = 2$	$y_3 = 2$
$x_4 = 1.5$	$y_4 = 1.9$
$x_5 = 4$	$y_5 = 3.1$

Linear regression assumes that the expected value of the output given an input, *E[y|x]*, is linear.

Simplest case: Out(*x*) = *wx* for some unknown *w* ($y_i = wx_i$ + noise$_i$).

Given the data, we can estimate *w*.

Linear Regression

- **Error function**

$$\mathrm{E} = \sum_i (y_i - wx_i)^2$$
$$= \sum_i y_i^2 - \left(2\sum x_i y_i\right)w + \left(\sum x_i^2\right)w^2$$

Easy to show the sum of squares is minimized when

$$w = \frac{\sum x_i y_i}{\sum x_i^2}$$

Minimum Squared Error solution (1)

- **The system of equations solved by MSE is (new notation)**

$$\begin{bmatrix} y_0^{(1} & y_1^{(1} & \cdots & y_D^{(1} \\ y_0^{(2} & y_1^{(2} & \cdots & y_D^{(2} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_0^{(N} & y_1^{(N} & \cdots & y_D^{(N} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_D \end{bmatrix} = \begin{bmatrix} b^{(1} \\ b^{(2} \\ \vdots \\ \vdots \\ b^{(N} \end{bmatrix} \Leftrightarrow Ya = b$$

- **where *a* is the weight vector, each row in *Y* is a training example, and each row in *b* is the corresponding output (class label).**

Lecture 3-23

Minimum Squared Error solution (2)

- **An exact solution to Ya=b can sometimes be found**
  - **If the number of (independent) equations (N) is equal to the number of unknowns (D+1), the exact solution is defined by**

$$a = Y^{-1}b$$

  - **For N> D+1 :** Approximate curve fitting
    - **In particular, MSE seeks to** Minimize the sum of the Squares **of these** Errors**:**

$$J_{MSE}(a) = \sum_{i=1}^{N} \left(a^T y^{(i} - b^{(i}\right)^2 = \|Ya - b\|^2$$

Lecture 3-24

The pseudo-inverse solution

- **The gradient of the objective function is**

$$\nabla_a J_{MSE}(a) = \sum_{i=1}^{N} 2\left(a^T y^{(i)} - b^{(i)}\right) y^{(i)} = 2Y^T(Ya - b) = 0$$

  - with zeros defined by

$$Y^T Ya = Y^T b$$

  - Notice that Y^TY is now a square matrix!
- **If Y^TY is nonsingular, the MSE solution becomes**

$$a = \left(Y^T Y\right)^{-1} Y^T b = Y^{\dagger} b$$ **Pseudo-inverse solution**

  - where the matrix $Y^{\dagger}=(Y^TY)^{-1}Y^T$ is known as the **pseudo-inverse** of Y ($Y^{\dagger}Y=I$)
    - Note that, in general, $YY^{\dagger} \neq I$ in general

Least-mean-squares solution

- **The objective function $J_{MSE}(a)=||Ya-b||^2$ can also be found using a gradient descent procedure**
  - This avoids the problems that arise when Y^TY is singular
  - In addition, it also avoids the need for working with large matrices
- **Looking at the expression of the gradient, the obvious update rule is**

$$a(k+1) = a(k) + \eta(k) Y^T (b - Ya(k))$$

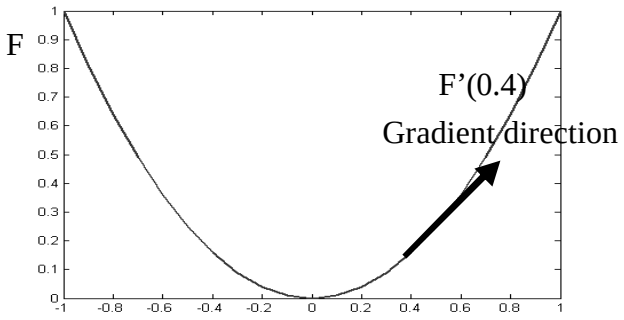
  - It can be shown that if $\eta(k)=\eta(1)/k$, where $\eta(1)$ is any positive constant, this rule generates a sequence of vectors that converge to a solution to $Y^T(Ya-b)=0$
- **The storage requirements of this algorithm can be reduced by considering each sample sequentially**

$$a(k+1) = a(k) + \eta(k)\left(b^{(i)} - y^{(i)} a(k)\right) y^{(i)}$$ **LMS rule**

  - This is known as the **Widrow-Hoff**, **least-mean-squares** (LMS) or **delta** rule [Mitchell, 1997]

Gradient descending

Gradient direction is the direction of uphill
for example, in the Figure, at position 0.4, the gradient is uphill (F is E, consider one dim case)

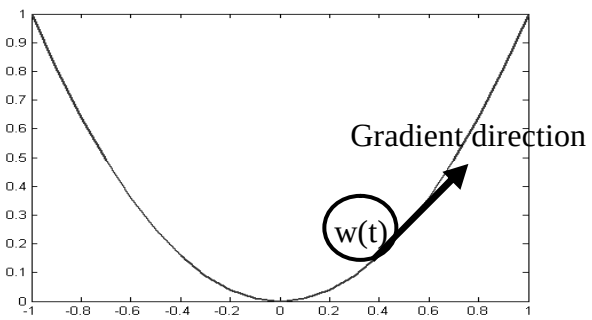


Gradient descending

- Gradient direction is the direction of uphill
- In gradient descent algorithm, we have

$$w(t+1) = w(t) - F'(w(t))\,\eta(t)$$

therefore the ball goes downhill since $-F'(w(t))$ is downhill direction

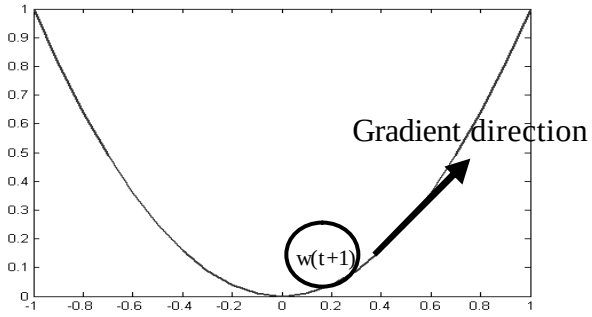


Gradient descending

- Gradient direction is the direction of uphill
- In gradient descent algorithm, we have

$$w(t+1) = w(t) - F'(w(t))\,\eta(t)$$

therefore the ball goes downhill since $-F'(w(t))$ is downhill direction



Lecture 3-29

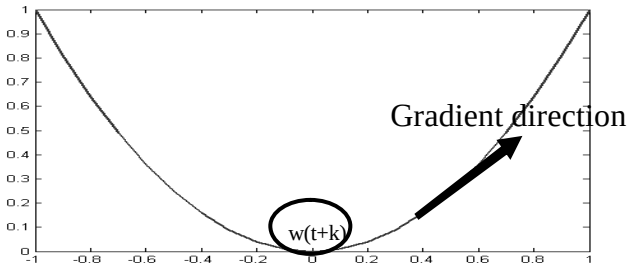
Gradient descending

- Gradient direction is the direction of uphill
- In gradient descent algorithm, we have

$$w(t+1) = w(t) - F'(w(t))\,\eta(t)$$

therefore the ball goes downhill since $-F'(w(t))$ is downhill direction

- Gradually the ball will stop at a local minima where the gradient is zero



Lecture 3-30

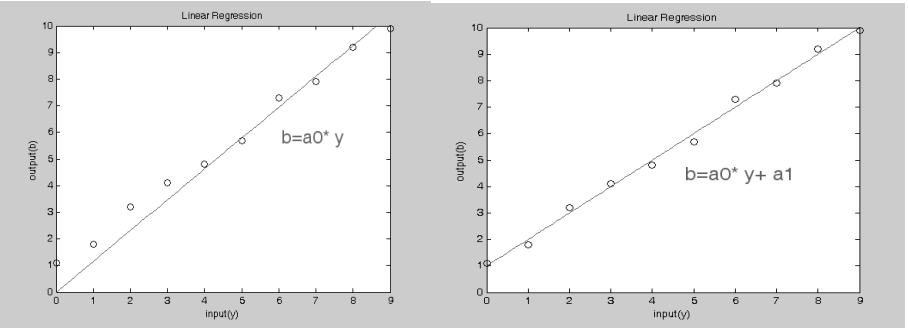
Example: The pseudo-inverse solution

- Find the least squares approximating polynomial of degree one(b=a0*y+a1) for data of the table
- The model is Ya=b(no bias)
  - Y=[0 1 2 3 4 5 6 7 8 9]T;
  - b=[1.1 1.8 2.7 3.8 5.5 6.3 7.2 7.7 9.1 9.8]T
  - a=[a0];
  - a0=1.1579;
- The model is Ya=b
  - Y=[0 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1 1]T;
  - b=[1.1 1.8 2.7 3.8 5.5 6.3 7.2 7.7 9.1 9.8]T
  - a=[a0 a1];
  - a0=1; a1=1;

	Inputs (y)	Outputs (b)
1	0	1.1
2	1	1.8
3	2	3.2
4	3	4.1
5	4	4.8
6	5	5.7
7	6	7.3
8	7	7.9
9	8	9.2
10	9	9.9

Lecture 3-31

Example: The pseudo-inverse solution



Lecture 3-32

Example:Least-mean-squares solution

- **Matlab M_file**
  - **close all**
  - **a(1)=0;**
  - **eta=2;**
  - **y=[0 1 2 3 4 5 6 7 8 9]';**
  - **b=[1.1000 1.8000 3.2000 4.1000 4.8000 5.7000 7.3000 7.9000 9.2000 9.900]';**
  - **for ii=1:100,**
    - **jj=mod(ii-1,10)+1;**
    - **a(ii+1)=a(ii)+eta/ii*(b(jj)-y(jj)*a(ii))*y(jj);**
  - **end**
  - **plot(1:101,a),title('parameter variation');xlabel('iteration');**
  - **ylabel('a0 value')**

Example:Least-mean-squares solution