课程名称：

机器学习算法与应用

讲授人：唐晨，徐岩

电气自动化与信息工程学院

教材或参考书目

1. E. Alpaydin, Introduction to Machine Learning, The MIT Press, 2004

2. M. Bishop，Pattern Recognition and Machine Learning，Springer, 2006

3.Witten,Ian H.,et al. Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann,2016

4.E.Alpaydin著，范明译，机器学习导论第三版，机械工业出版社，2016

Professor Chen Tang

E-mail: tangchen@tju.edu.cn

Office: 26D-316

Associate Professor Yan Xu

E-mail: xuyan@tju.edu.cn

Office: 26D-426

MachineLearningTJU@126.com

课程设置背景

1.时代发展的需要

机器学习是新思想、新观念、新理论、新技术不断出现并迅速发展的新兴学科，是人工智能的基础。

人工智能改变未来世界. *党的十九大报告指出*，要推动互联网、大数据、人工智能和实体经济深度融合。*天津市*部署了《加快推进智能科技产业发展总体行动计划》，明确大力发展智能科技产业。*天津大学*刚刚成立了人工智能学院和智能与计算学部。

人才的质量和数量决定着人工智能发展的水平和潜力。 《新一代人工智能发展规划》指出，我国人工智能尖端人才远远不能满足需求。发展人工智能，核心在于人。

开设人工智能基础课程机器学习算法和应用是十分必要，是时代发展的需要。

课程设置背景

2.研究生培养的需要

目前几乎所有学科（工科、理科、金融和文法等等）研究生的研究方向与机器学习相关，然而现在的绝大多数研究生在本科阶段没有学过机器学习这门课程。

这门课程适合研究生公共选修课，是研究生培养的需要。

课程设置背景

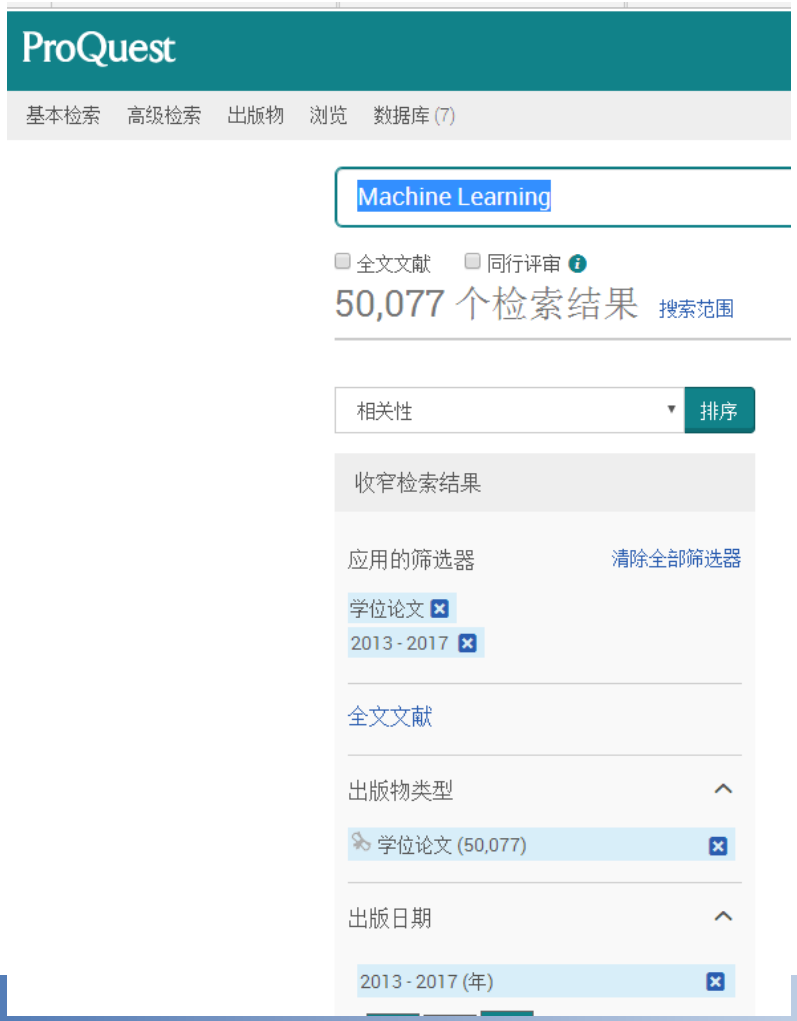
近5年（2013-2017）万方以“机器学习”为关键词

课程设置背景

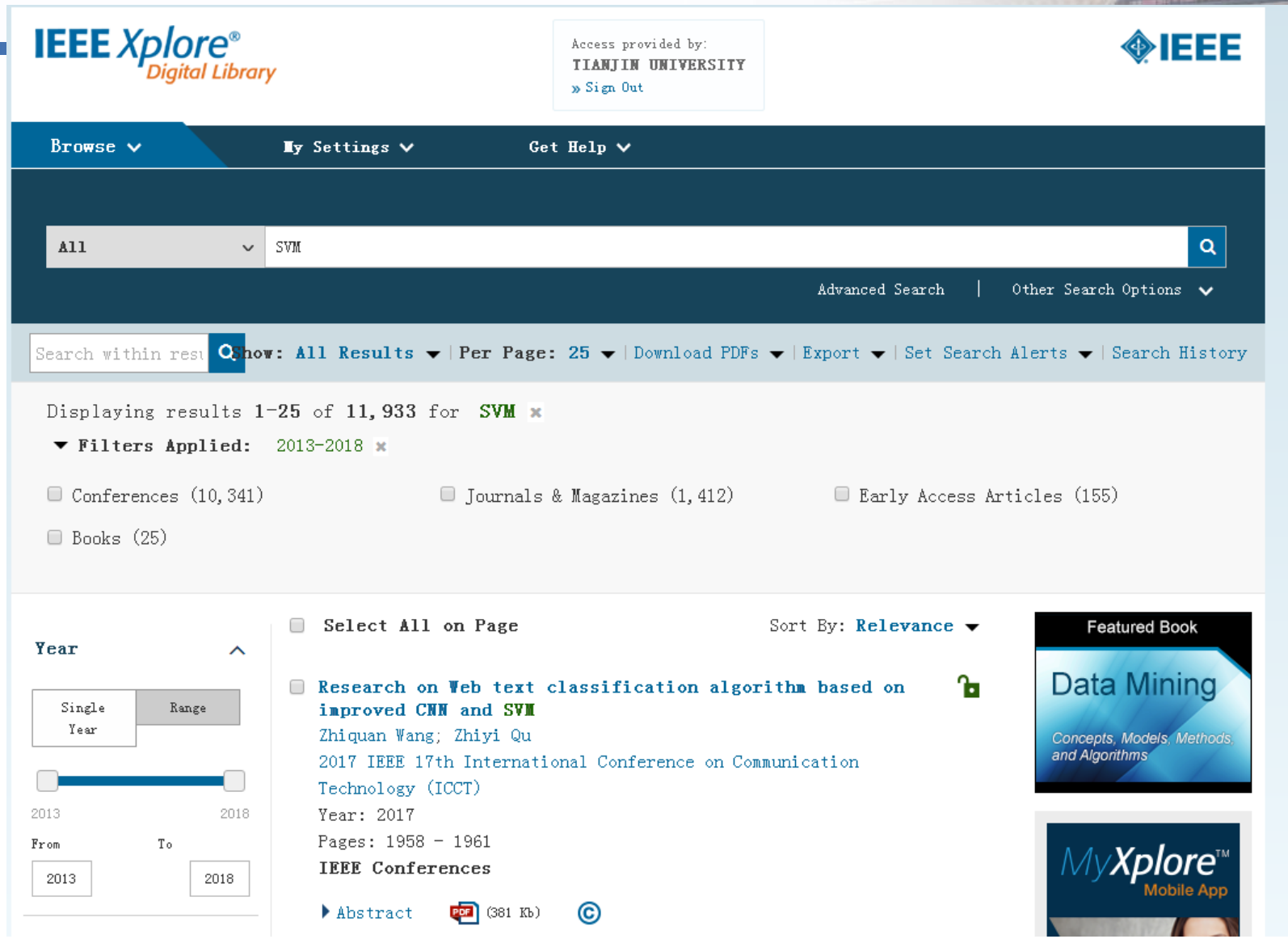
学科分类	学位论文数
工业技术	4624
经济	267
医疗、卫生	262
交通运输	179
文化科学	160
数理科学和化学	148
生物科学	130
天文学、地球	83
农业科学	74
语言、文学	39

课程设置背景

（2013-2017） “Machine Learning”ProQuest



3.机器学习的主流算法在科学研究中广泛应用



IEEE Xplore® Digital Library

Access provided by:
TIANJIN UNIVERSITY
» Sign Out

IEEE

Browse | My Settings | Get Help

All | PCA

Advanced Search | Other Search Options

Search within rest | Show: All Results | Per Page: 25 | Download PDFs | Export | Set Search Alerts | Search History

Displaying results 1-25 of 4,539 for PCA

Filters Applied: 2013-2018

- Conferences (3,906)
- Journals & Magazines (567)
- Early Access Articles (60)
- Books (5)
- Standards (1)

Year

Single Year | Range

2013 2018

From 2013 To 2018

Select All on Page

Sort By: Relevance

A Comparison between ECG Beat Classifiers Using Multiclass SVM and SIMCA with Time Domain PCA Feature Reduction

Najlaa Jannah; Sillas Hadjiloucas

2017 UKSim-AMSS 19th International Conference on Computer Modelling & Simulation (UKSim)

Year: 2017

Pages: 126 - 131

IEEE Conferences

Abstract (281 Kb)

课程设置背景

4.新工科人才培养的需要

新工科是基于国家战略发展新需求、国际竞争新形势、立德树人新要求而提出的我国工程教育改革方向。

“新工科”对应的是新兴产业，首先是指针对新兴产业的专业，如人工智能、智能制造、机器人、云计算等，也包括传统工科专业的升级改造。

统计数据显示，到2020年，新一代信息技术产业、电力装备、高档数控机床和机器人、新材料将成为人才缺口最大的几个专业，其中新一代信息技术产业人才缺口将会达到750万人。到2025年，新一代信息技术产业人才缺口将达到950万人，电力装备的人才缺口也将达到909万人。

这门课程对新工科人才的培养是十分重要的。

主要教学内容

背景

预备知识的回顾

机器学习主流算法

算法的实现

算法的应用

课程特色

（1）注重基础理论的讲授，提高研究生基础理论的水平，为深度学习打好基础

注重

算法的意义
如何公式化
如何数值求解

（2）注重算法的实现

标准PCA聚类：

```
%% standard PCA
disp('Performing standard PCA...');
Y1=PCA(data,d);
figure;hold on;
plot(Y1(1:2:end,1),Y1(1:2:end,2),'b*');
plot(Y1(2:2:end,1),Y1(2:2:end,2),'ro');
legend('class 1','class 2');
title('standard PCA');
drawnow;
```

eig
Eigenvalues and eigenvectors

```
function [Y, eigVector, eigValue]=PCA(X,d)

%% eigenvalue analysis
Sx=cov(X);
[V,D]=eig(Sx);
eigValue=diag(D);
[eigValue,IX]=sort(eigValue,'descend');
eigVector=V(:,IX);

%% normailization
norm_eigVector=sqrt(sum(eigVector.^2));
eigVector=eigVector./repmat(norm_eigVector,size(eigVector,1),1);

%% dimensionality reduction
eigVector=eigVector(:,1:d);
Y=X*eigVector;
```

课程特色

（3）注重应用的介绍

中国人口数和大学生毕业5年后工资的线性回归

PCA算法在某农业生态经济系统的应用

PCA算法在男性身材分析中的应用

KPCA在人脸识别中的应用

KPCA在数据可视化中的应用

LDA算法在图像处理中的应用

KLDA在数据分类中的应用

SVM的在遥感图像识别中的应用

课程特色

（4）注重机器学习主流算法的升级

强调基于Kernel的算法，并及时更新算法。

机器学习先进算法和应用

课程特色

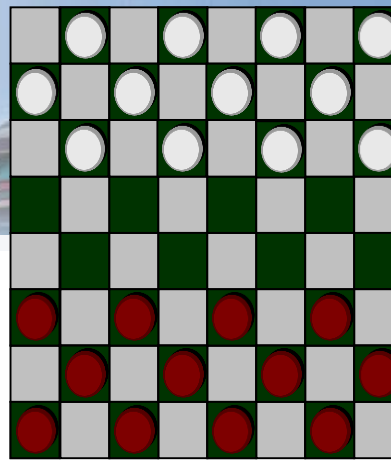
（5）考试方式强调基础理论和综合应用与分析能力方面的考核

1.大作业 （40%）

2.完成与课程相关的研究项目：60%

科技报告（文献综述、方法描述、实验测试、量化评估、结果讨论和总结）

编写程序



Machine Learning definition

Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

机器学习是研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。它是人工智能的核心，是使计算机具有智能的根本途径，其应用遍及人工智能的各个领域.

Machine Learning

-New capability for computers

Examples:

- Database mining

  Large datasets from growth of web.

  E.g., Web click data, medical records, biology, engineering

- Applications can't program by hand.

  E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.

Medical, Image classification, The prediction of an event.

Emotion Calssification on Face Images

Figure 1: Examples of face images. From left to right: Disgust (label 3), Happy (label 5) and Sadness (label 6).

SIFT (Scale-Invariant Feature Transform)

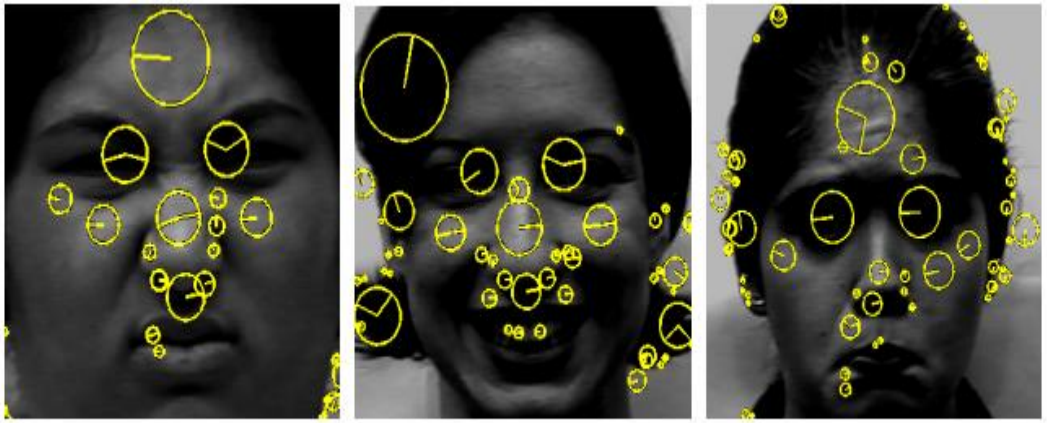


Figure 2: Examples of SIFT descriptors of our images.

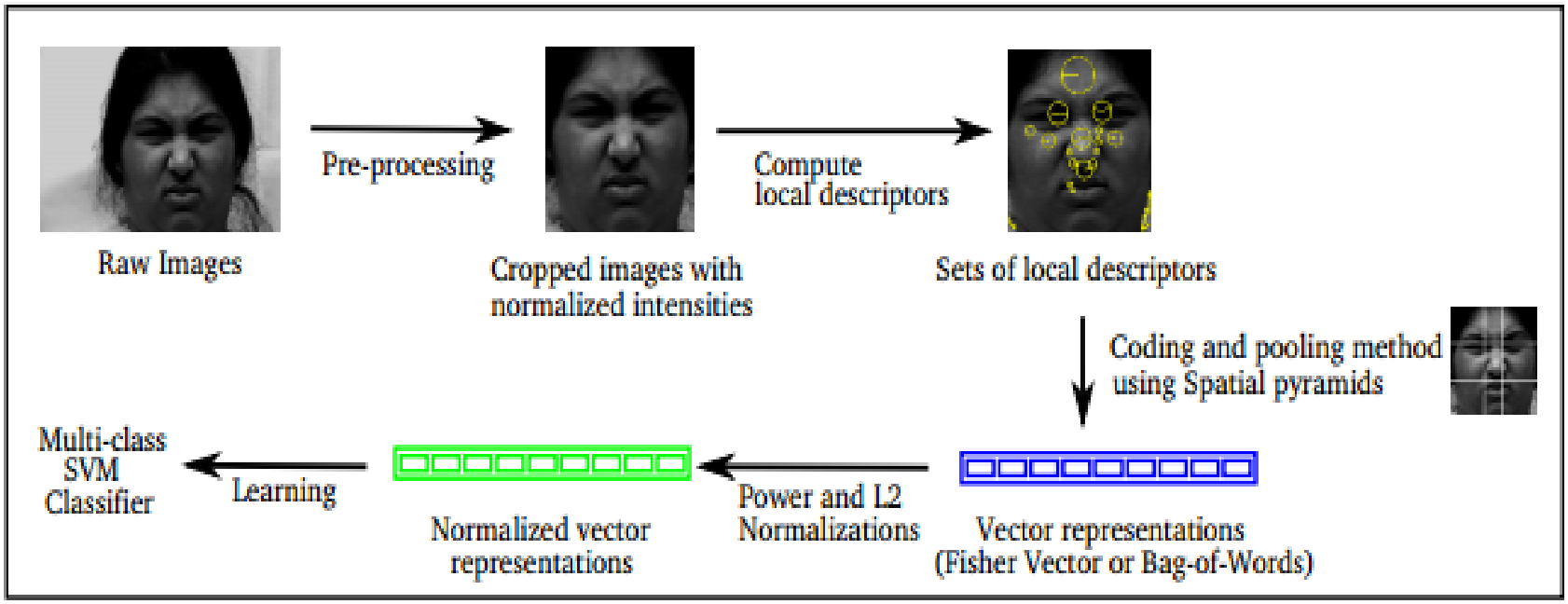


Figure 4: Pipeline of our learning algorithm.

Calorie Estimation from Food Images

Dynamic Eye Gaze Tracking and Prediction

Predicting Short-term Market Response to Liquidity Shocks

Predicting Student Earnings After College

Predicting Stock Prices through Textual Analysis of Web News

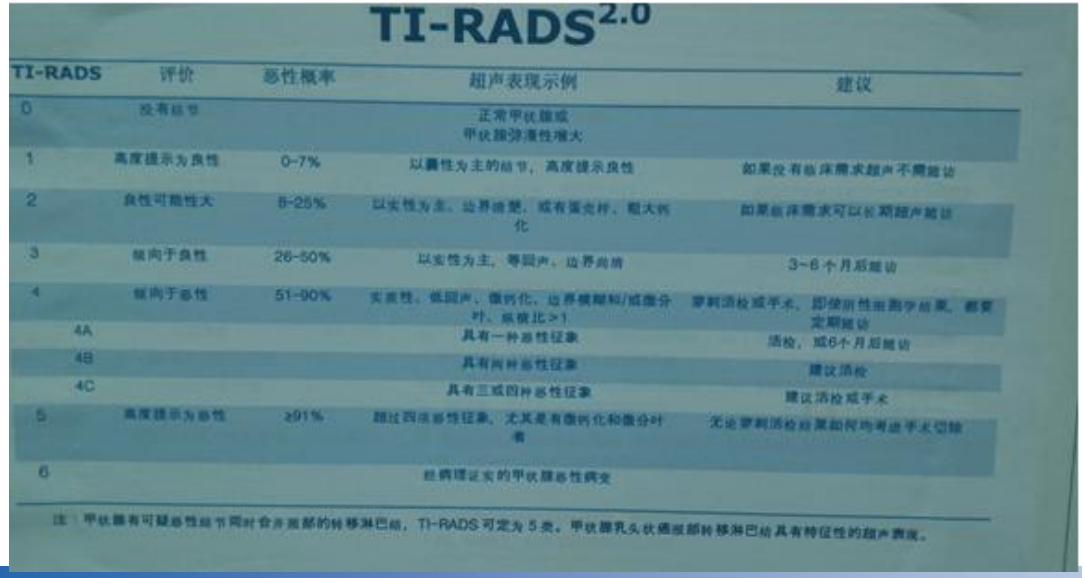
Human Activity Recognition Using Cell Phones

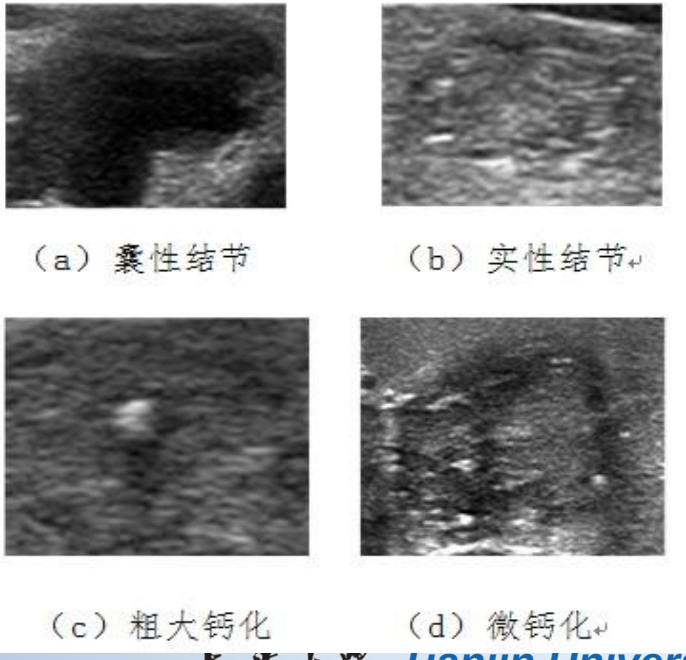
Human Activity Recognition Using Wearable Devices Sensor Data

3D model classification using convolutional neural network

基于机器学习的甲状腺结节分级方法

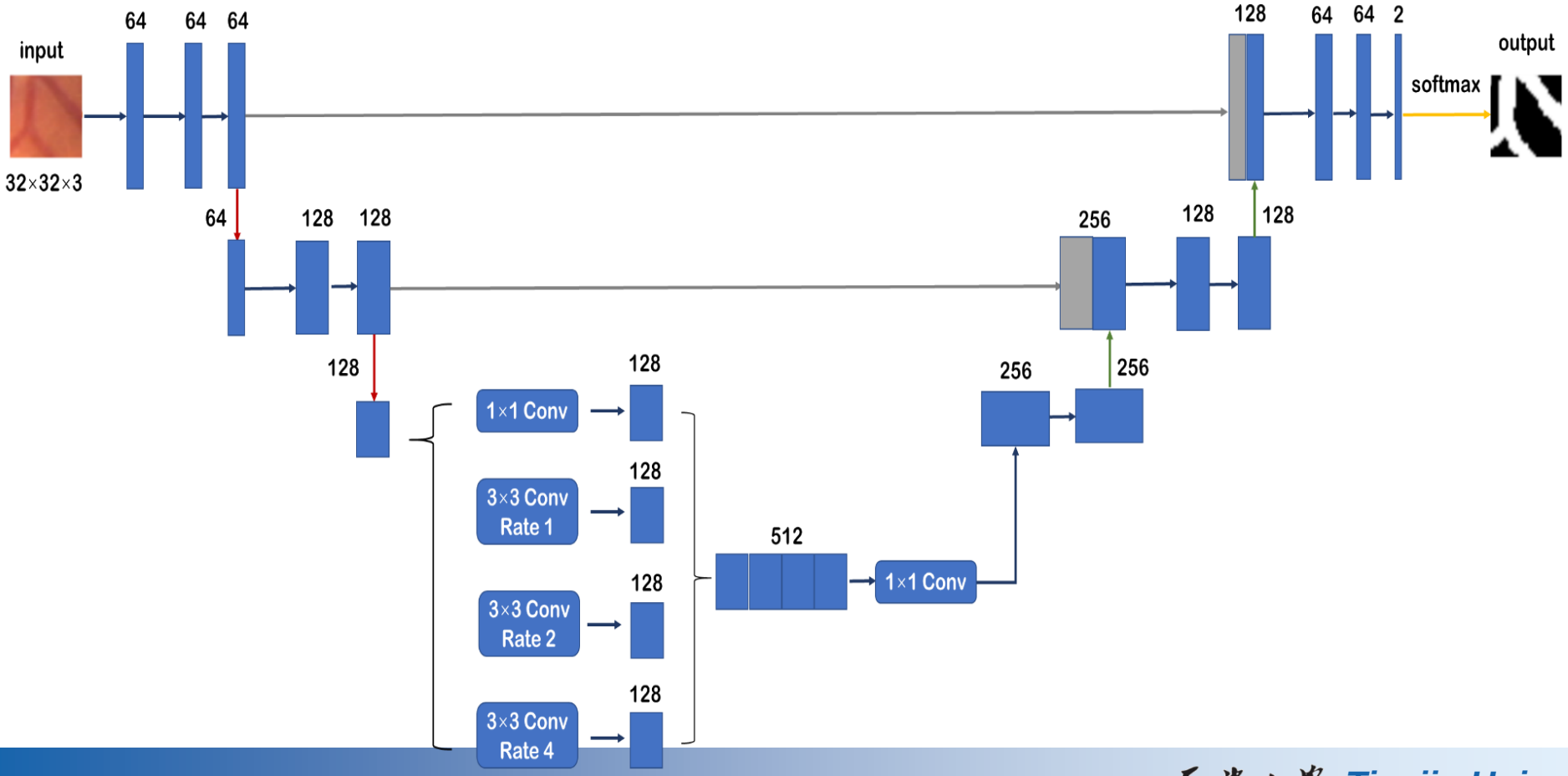
课题组采用多尺度融合方法对甲状腺结节超声图像特征进行提取，包括：实性结节内部组织，边界，回声，钙化程度。然后基于机器学习方法对甲状腺结节实现分级，目前该工作正在完善中。左图为甲状腺疾病分级图，右图为甲状腺结节超声图像。





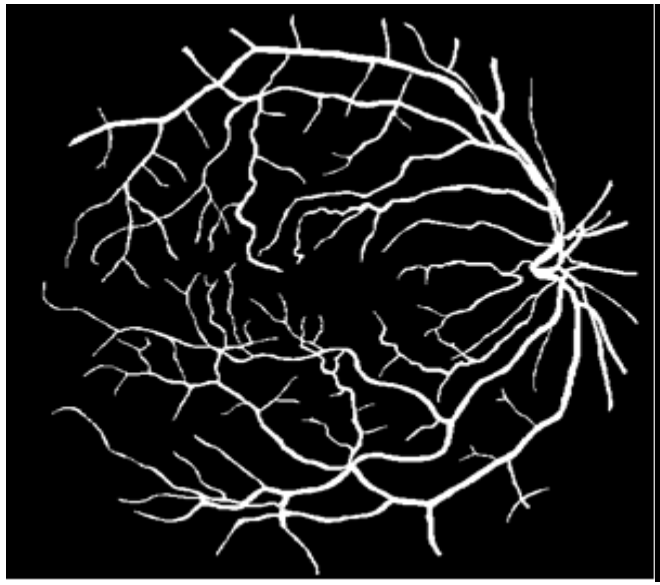
基于结合多尺度特征的卷积神经网络的视网膜血管分割方法

- 网络整体结构图

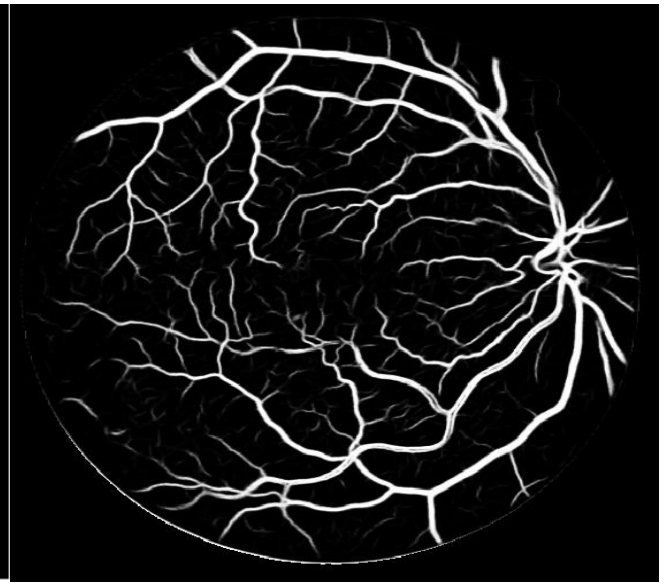


- 血管分割图

原始图像



第一专家手动分割图



视网膜血管分割图

Intelligent transportation;

Intelligent home furnishing;

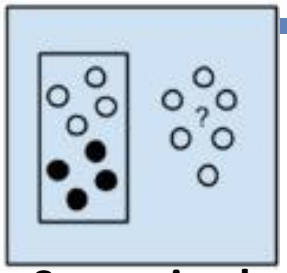
Intelligent home appliances;

Intelligent buildings;

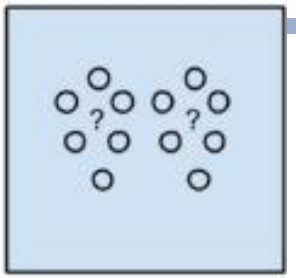
Intelligent internet.

Intelligence: Simulate and extend human behavior.

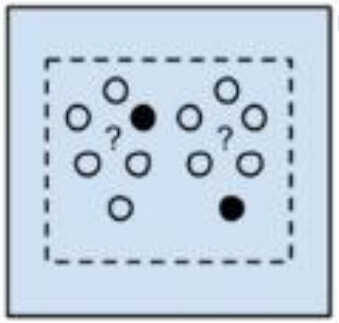
Types of machine learning:



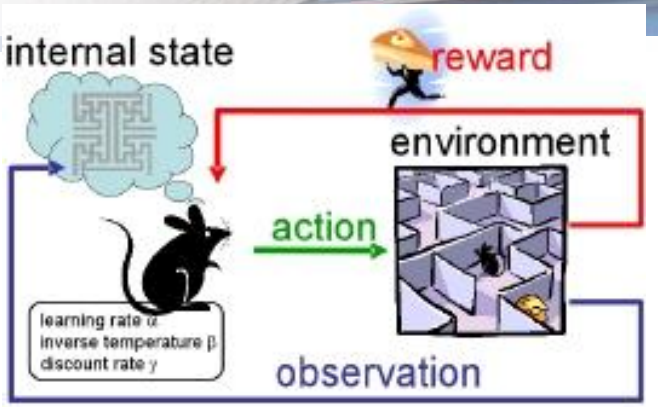
Supervised Learning



Unsupervised Learning

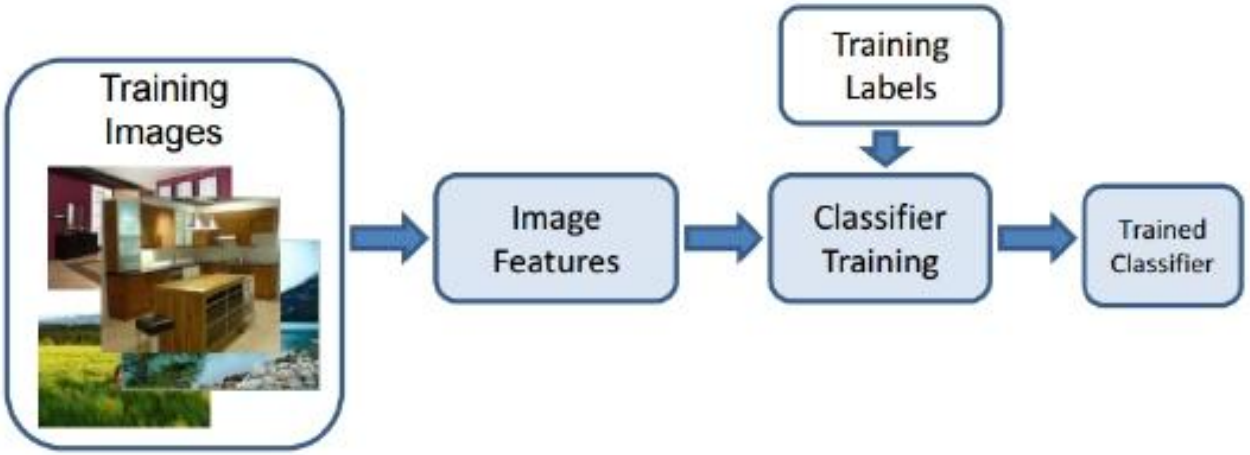


Semi-Supervised Learning



Reinforcement Learning

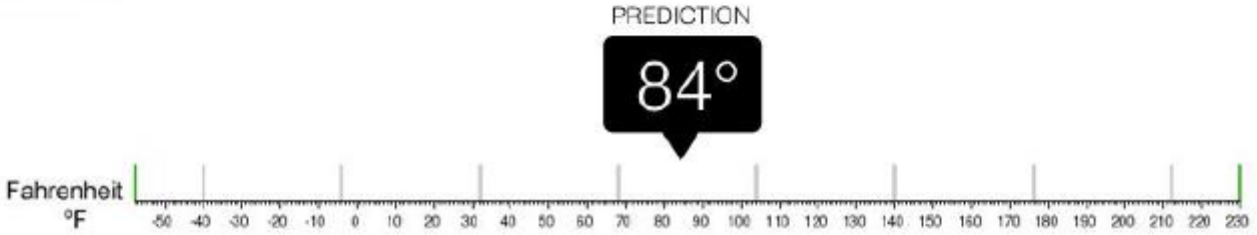
Generalized supervised learning pipeline:



Regression vs Classification

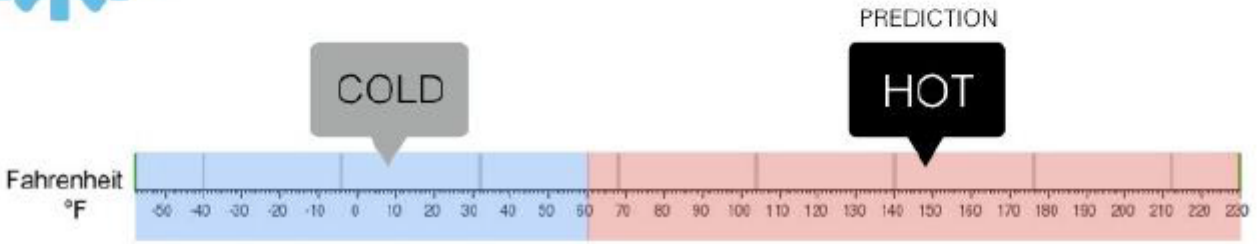
Regression

What is the temperature going to be tomorrow?

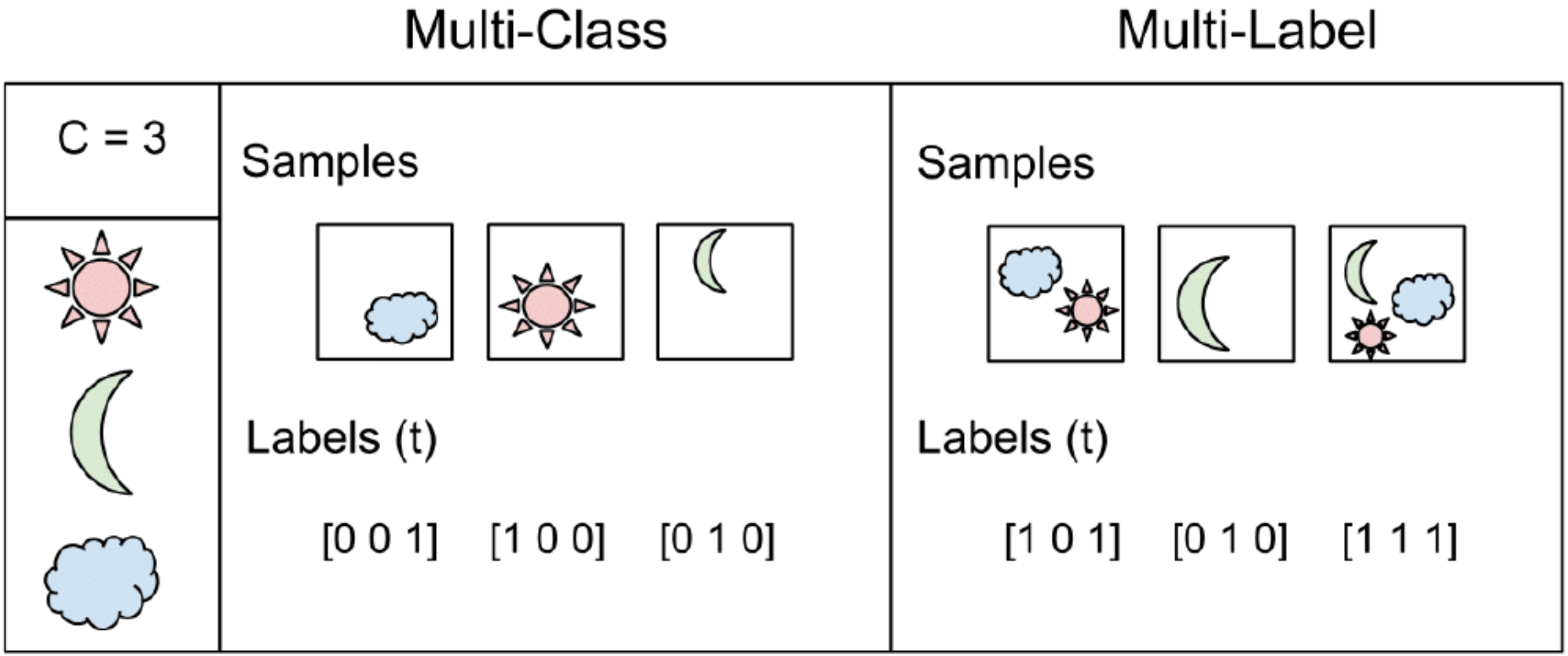


Classification

Will it be Cold or Hot tomorrow?

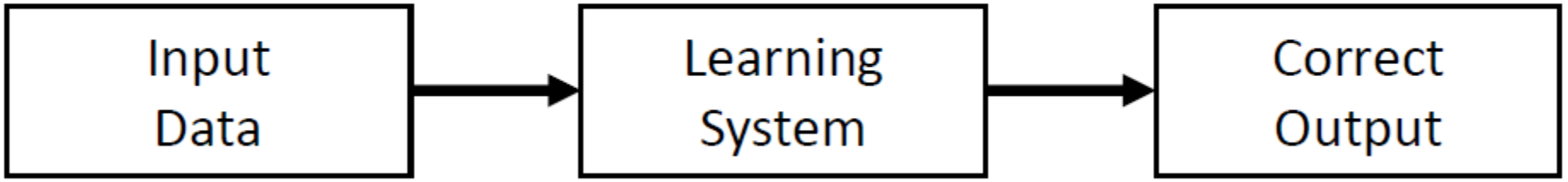


Multi-Class vs Multi-Label



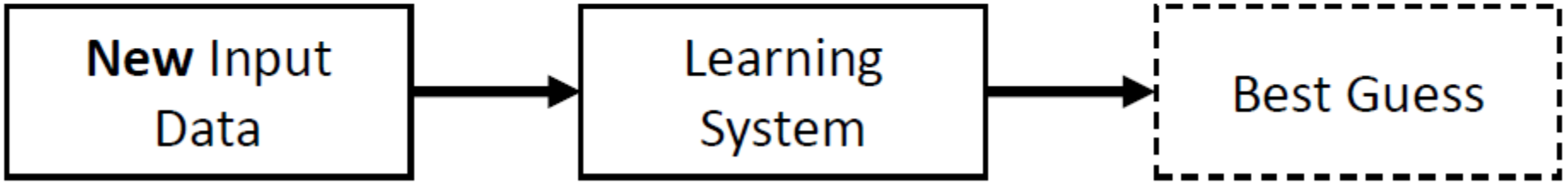
machine Learning: Training and Testing

Training Stage:

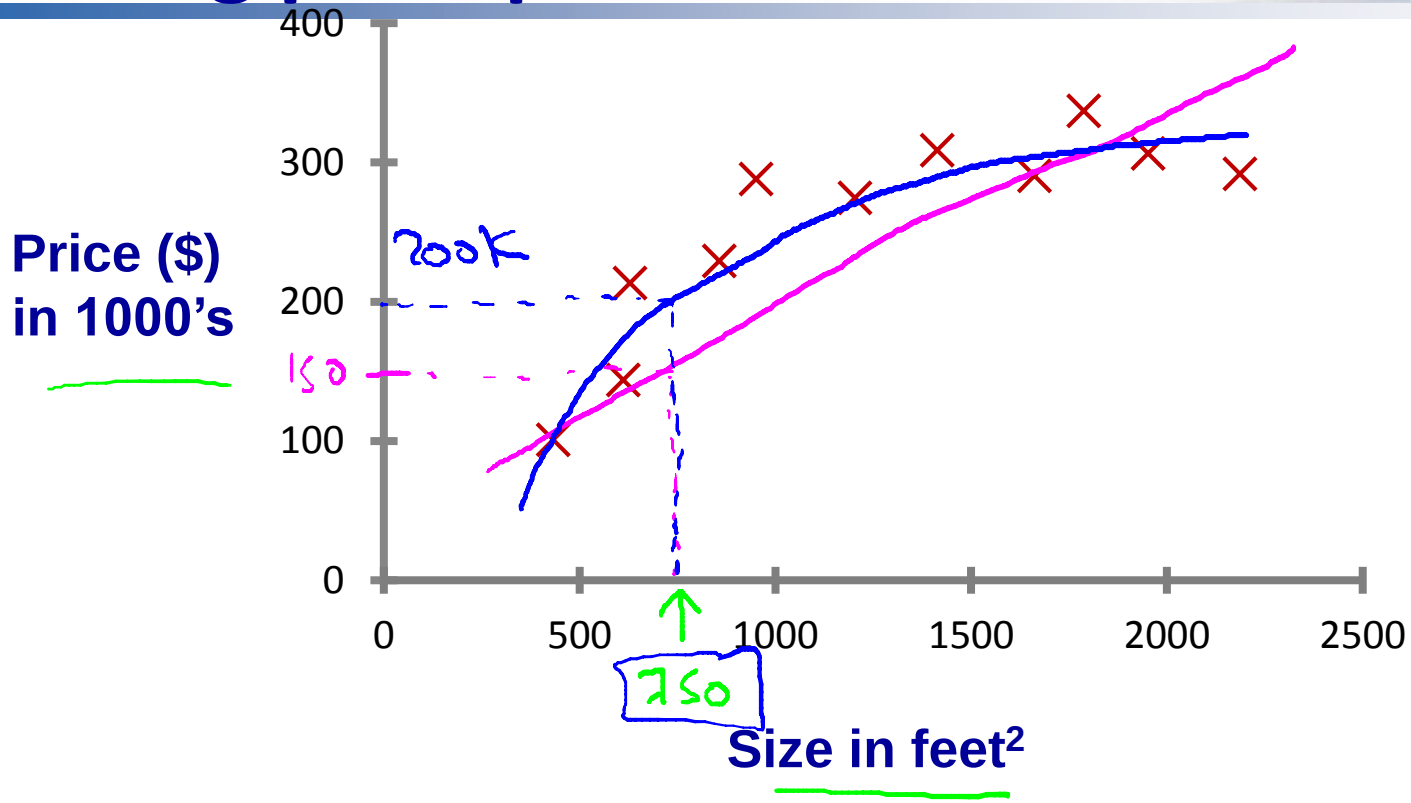


(aka "Ground Truth")

Testing Stage:



Supervised learning
Housing price prediction.

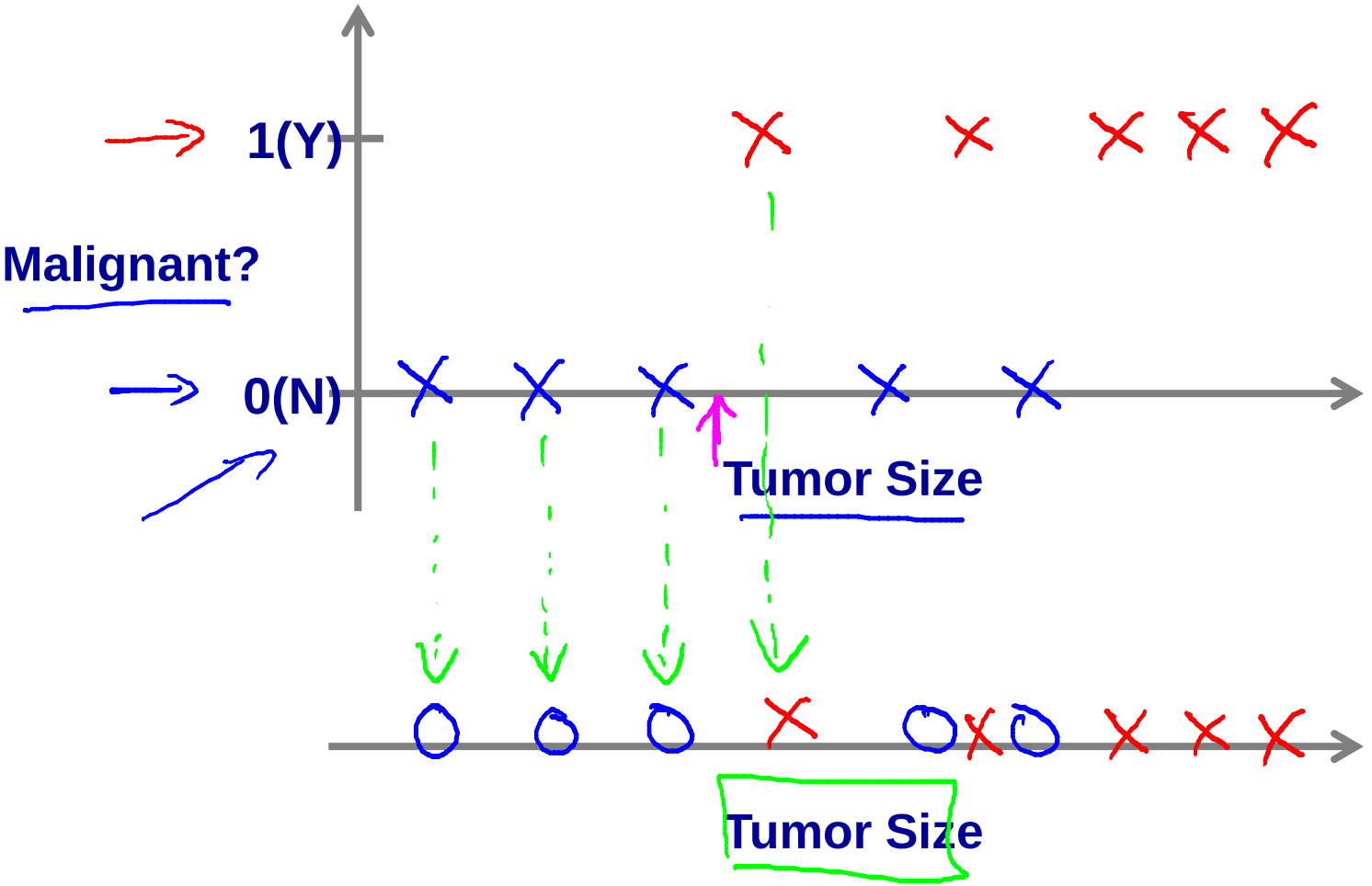


Supervised Learning

"right answers" given

Regression: Predict continuous valued output (price)

Breast cancer (malignant, benign)

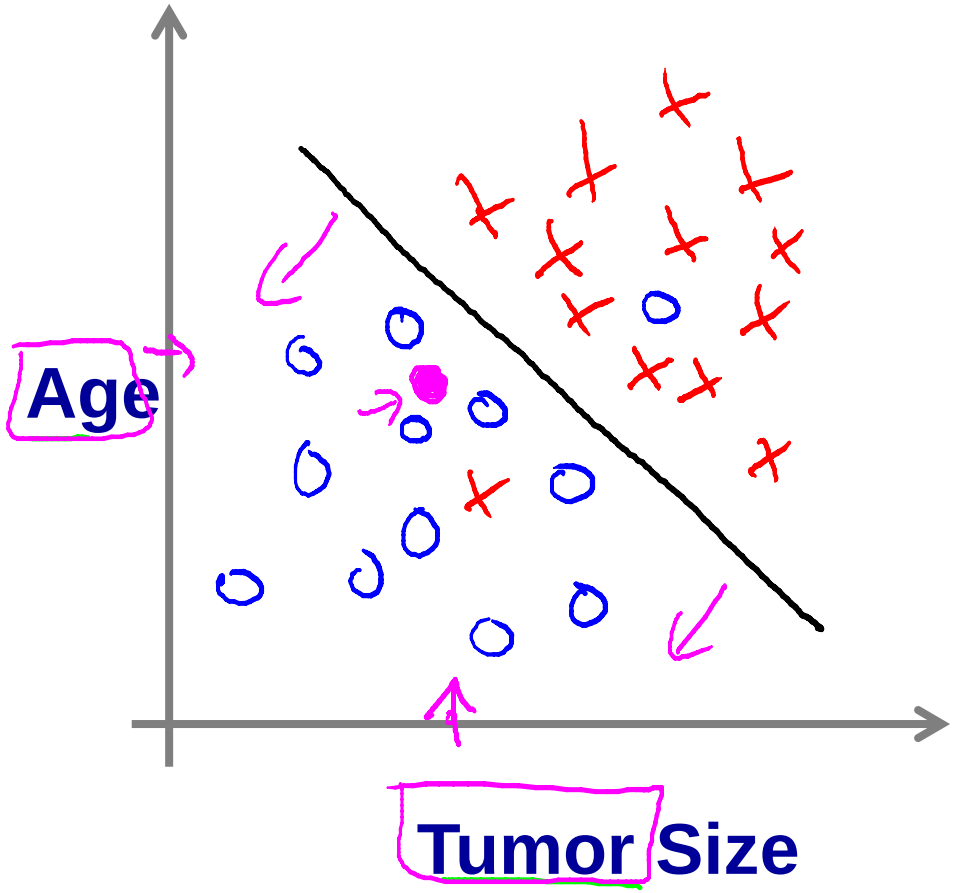


Classification

Discrete valued output (0 or 1)

0, 1, 2, 3

benign type 1 Cancer



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

...

You are running a company, and you want to develop learning algorithms to address each of two problems.

Problem1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.

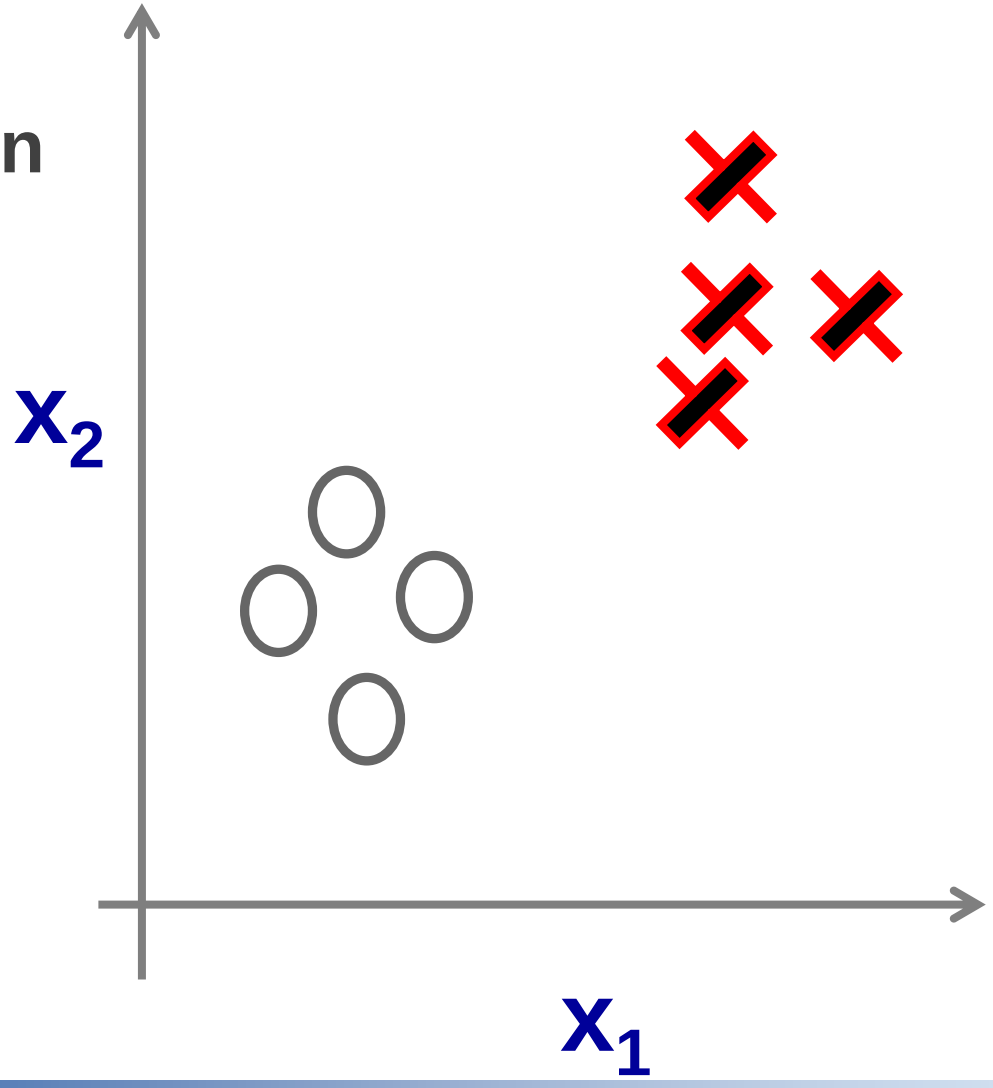
Problem2: You would like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

Should you treat these as classification or as regression problems?

A. Treat both as classification problems.
B. Treat problem 1 as a classification problem, problem 2 as a regression problem.
C. Treat problem 1 as a regression problem , problem 2 as a classification problem.
D. Treat both as regression problems.

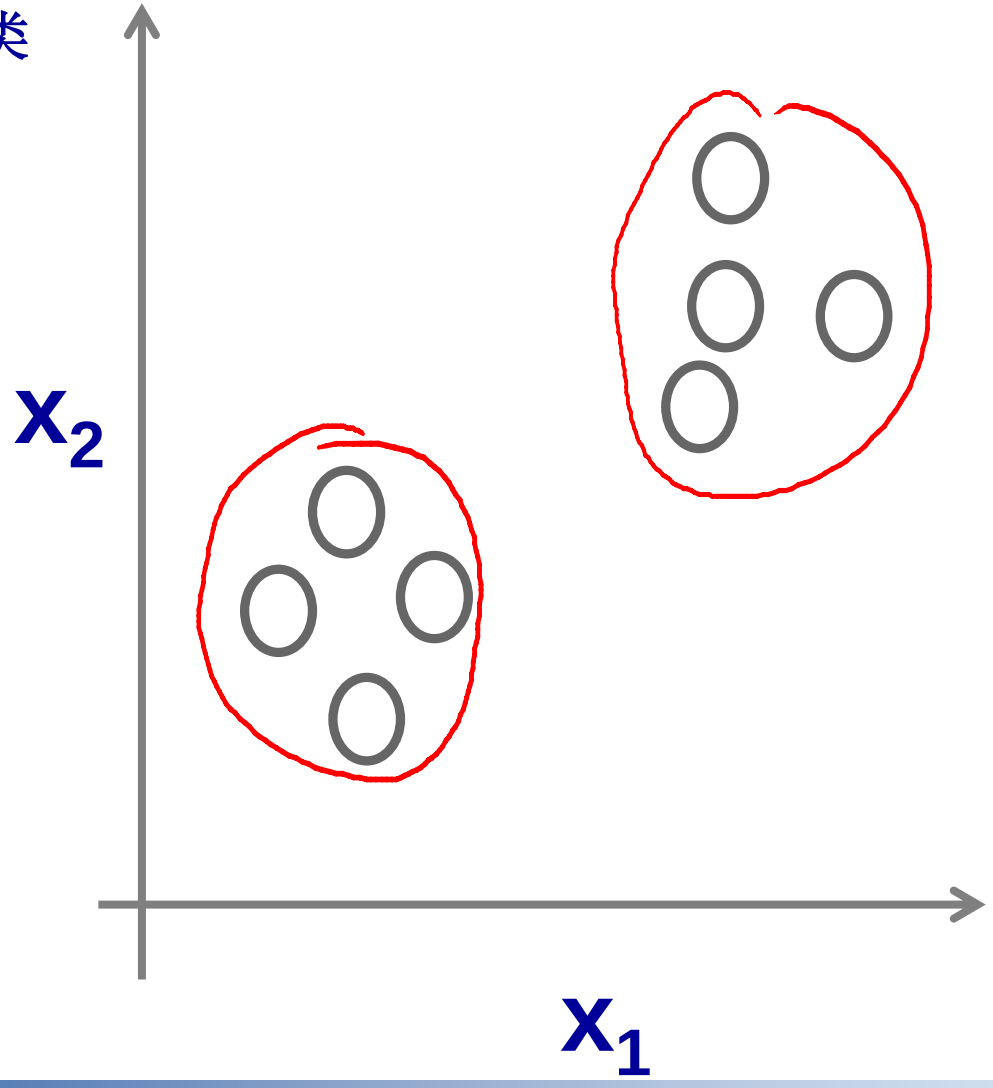
Supervised Learning

Classification
分类



Unsupervised Learning

Clustering : 聚类



news.google.com

Of the following examples, which would you address using an unsupervised learning algorithm? (Check all that apply.)

Given email labeled as spam/not spam, learn a spam filter.

Given a set of news articles found on the web, group them into set of articles about the same story.

Given a database of customer data, group customers into different market segments.

Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

第一次作业：基于WIFI信号强度的室内定位

数据集: **Collected in indoor space by observing signal strengths of seven WiFi signals visible on a smartphone. The decision variable is one of the four rooms.**

Data Set Characteristics:	**Multivariate**	**Number of Instances:**	**2000**	**Area:**	**Computer**
Attribute Characteristics:	Real	Number of Attributes:	7	Date Donated	2017-12-04
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	7850

选择合适算法做实验，按作业模板完成报告

报告与实验代码打成一个压缩包，提交至MachineLearningTJU@126.com

压缩包命名：作业1-姓名-学号

提交日期：第八周周日（）

Lecture_2

Matlab Introduction

(1) Numerical calculation(Optimization algorithm);

(2) 2D and 3D plot;

(3) Image processing

roots

Solving a equation

Example

$$2x^5 - 3x^3 + 71x^2 - 9x + 1 = 0$$

$$p = [2, 0, -3, 71, -9, 1];$$

$$x = roots(p)$$

```
x =

  -3.4774 + 0.0000i
   1.6753 + 2.7105i
   1.6753 - 2.7105i
   0.0634 + 0.1007i
   0.0634 - 0.1007i
```

【Functional demonstration-2】
Solving a linear equation

$$\begin{cases} 2x+3y-z=2 \\ 8x+2y+3z=4 \\ 45x+3y+9z=23 \end{cases}$$

```
a = [2,3,-1;8,2,3;45,3,9];%建立系数矩阵a
b = [2;4;23];%建立列向量b
x = inv(a)*b
```

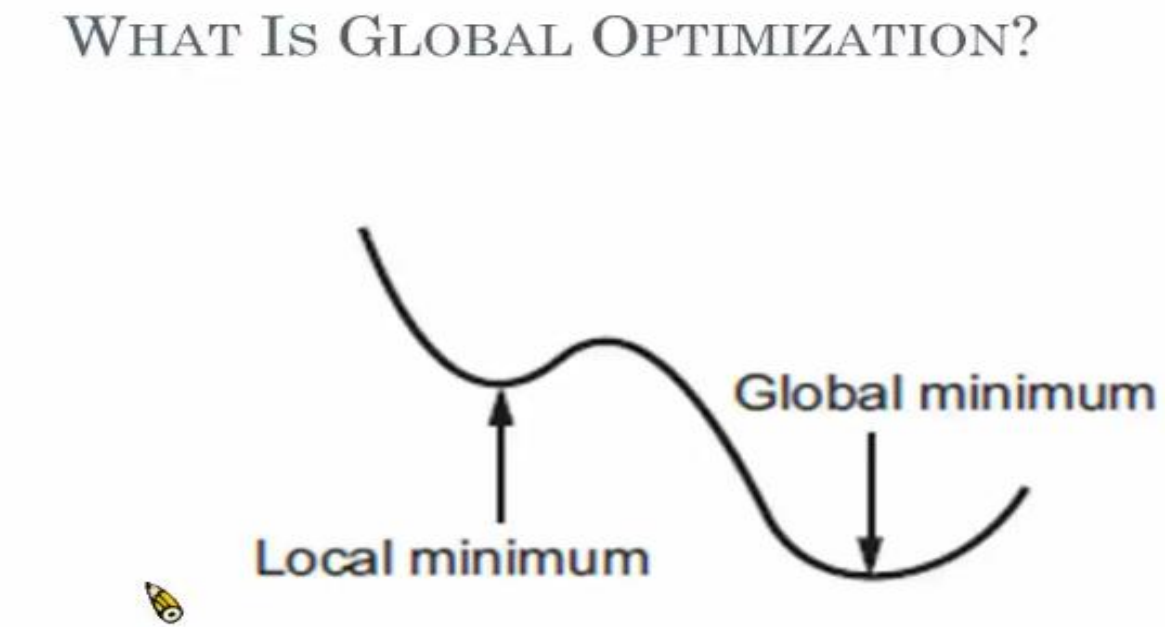
```
x =
    0.5531
    0.2051
   -0.2784
```

Symbolic calculation

```
syms x y z %建立符号变量
[x,y,z]=solve(2*x+3*y-z-2,8*x+2*y+3*z-4,45*x+3*y+9*z-23)
```

```
x =
151/273
y =
8/39
z =
-76/273
```

Optimization algorithm



Optimization Workflow

- Local or global solution
- Write objective function (constraint functions)
- Set appropriate options
- Run the solver
- Examine the result
- If the result is unsatisfactory, change options or start points or otherwise update your optimization and rerun it.

fmincon

Find minimum of constrained nonlinear multivariable function

Finds the minimum of a problem specified by

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub, \end{cases}$$

b and beq are vectors, A and Aeq are matrices, c(x) and ceq(x) are functions that return vectors, and f(x) is a function that returns a scalar. f(x), c(x), and ceq(x) can be nonlinear functions.

1. x = fmincon(fun,x0,A,b) startsat x0 and attempts to find a minimizer x ofthe function described in fun subject to the linear inequalities A*x ≤ b. x0 canbe a scalar, vector, or matrix.

2. x = fmincon(fun,x0,A,b,Aeq,beq) minimizes fun subjectto the linear equalities Aeq*x = beq and A*x ≤ b. If no inequalitiesexist, set A = [] and b = [].

3. x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub) definesa set of lower and upper bounds on the design variables in x,so that the solution is always in the range lb ≤ x ≤ ub.If no equalities exist, set Aeq = [] and beq= []. If x(i) is unbounded below, set lb(i)= -Inf, and if x(i) is unbounded above,set ub(i) = Inf.

Example

Find values of x that minimize $f(x) = -x_1x_2x_3$, starting at the point x = [10;10;10], subject to the constraints:

$0 \le x_1 + 2x_2 + 2x_3 \le 72$.

```
%1  Write a file that returns a scalar value f ofthe objective function evaluated at x:

function f = myfun(x)
f = -x(1) * x(2) * x(3);
```

```
clear;



%Rewrite the constraints
-x(1)-2*x(2)-2*x(3) <= 0;
 x(1) + 2*x(2) +2*x(3) <= 72;
% Since both constraints are linear, formulate themas the matrix inequality A·x ≤ b,where
A = [-1 -2 -2; ...
      1  2  2];
b = [0;72];
% Supply a starting point and invoke an optimizationroutine:
x0 = [10;10;10];    % Starting guess at the solution
[x,fval] = fmincon(@myfun,x0,A,b);
```

fminunc

Find minimum of unconstrained multivariable function

Finds the minimum of a problem specified by

min f (x)

where f(x) is a function that returns a scalar. x is a vector or a matrix

Examples

minimize the function $f(x) = 3X_1^2+2X_1X_2+X_2^2$

```
function f = myfun(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;     % Cost function
```

```
%Then call fminunc to find a minimum of myfun near [1,1]:
clear
x0 = [1,1];
[x,fval] = fminunc(@myfun,x0);

x
fval
```

minimize the function $f(x) = 3X_1^2+2X_1X_2+X_2^2$

minimize the function f(x) = sin(x) + 3 by using matlab software

Constructing function

```
function f = myfun(x)
f = sin(x)+3;
```

Calling function

[x,fval] = fminunc(@myfun,0)

Plot: 2-D line plot;

Mesh: 3-D line plot

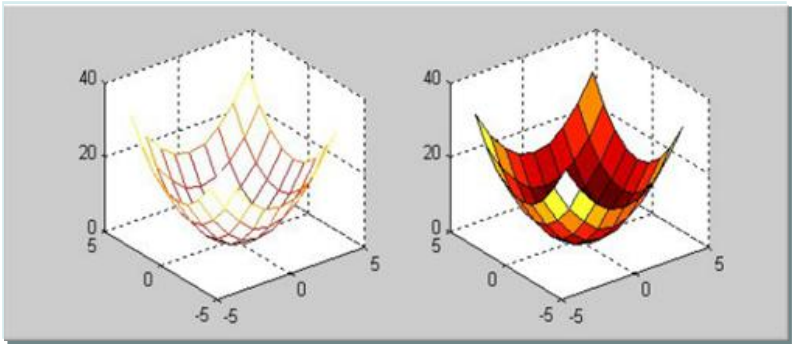
Example

Drawing the surface of function, $z=x^2+y^2$

```
x=-4:4;y=x;
[x,y]=meshgrid(x,y);          %生成 x-y 坐标“格点”矩阵
z=x.^2+y.^2;                  %计算格点上的函数值
subplot(1,2,1), mesh(x,y,z);  %三维网格图

subplot(1,2,2), surf(x,y,z);  %三维曲面图
colormap(hot);
```

The obtained surface of function, $z=x^2+y^2$



Matlab for image processing

imread
Read image from graphics file

```
A = imread('ngc6543a.jpg');
```

```
imshow(A);
```

imwrite
Write image to graphics file

adapthisteq (adapt hist eq)

Contrast-limited adaptive histogram equalization (CLAHE)

J = adapthisteq(I)
Performs CLAHE on the intensity image I.

bwarea(BW)

total = bwarea(BW) estimates the area of the objects in binary image BW. total is a scalar whose value corresponds roughly to the total number of on pixels in the image.

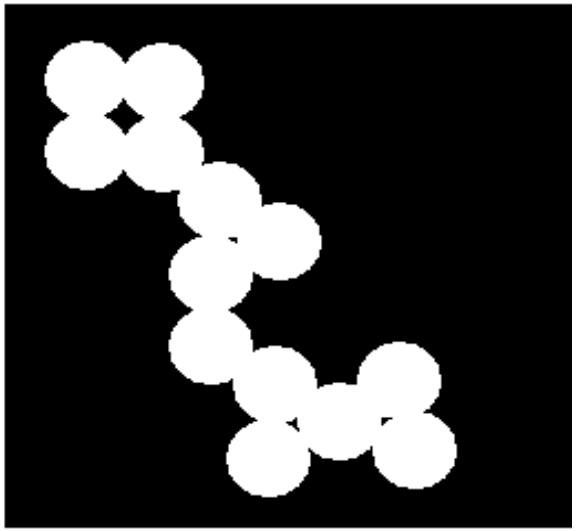
Example

```
BW = imread('circles.png');

imshow(BW);

    bwarea(BW)


ans =
  1.4187e+004
```

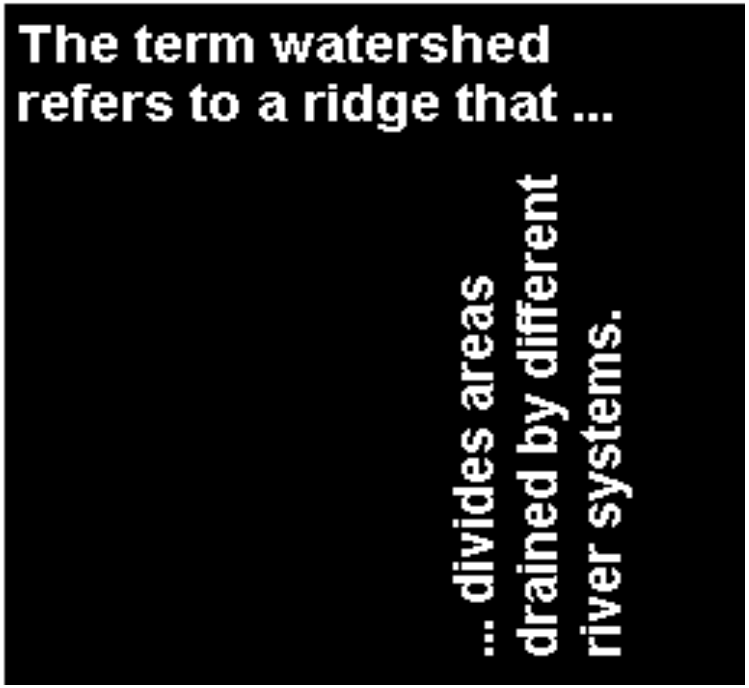


bwareaopen Remove small objects from binary image

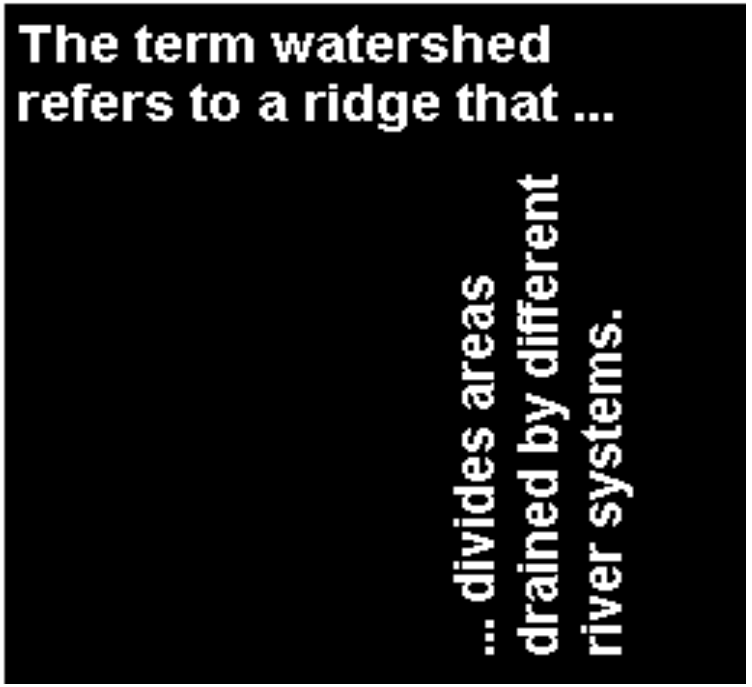
Remove all objects smaller than 50 pixels

Example

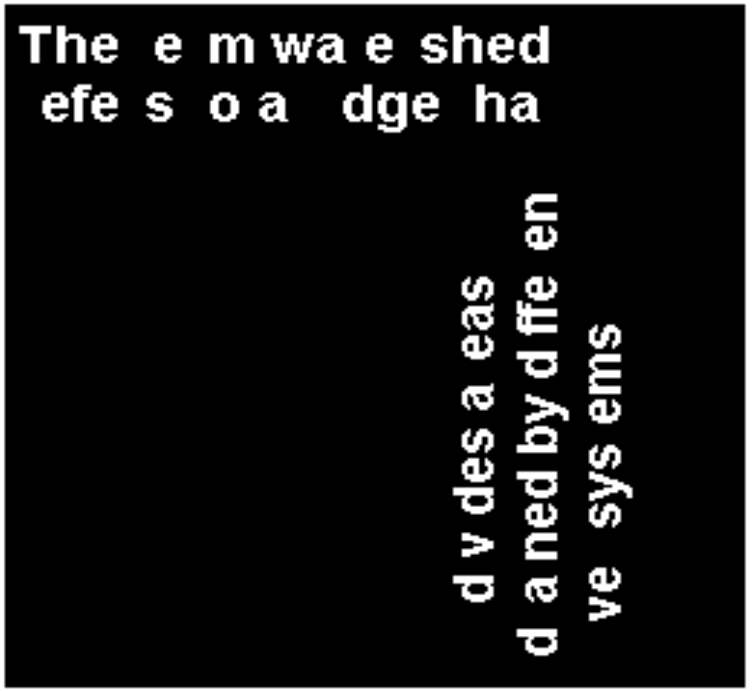
```
originalBW = imread ('text.png');
imshow(originalBW)
bwAreaOpenBW =
bwareaopen(originalBW,50);
figure, imshow(bwAreaOpenBW)
```



original BW



original BW



bwareaopen BW

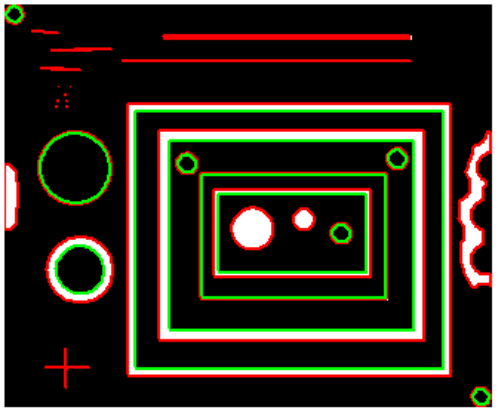
bwboundaries

Trace region boundaries in binary image

Example

Display object boundaries in red and hole boundaries in green.

```
BW = imread('blobs.png');
[B,L,N] = bwboundaries(BW);
figure; imshow(BW); hold on;
for k=1:length(B),
   boundary = B{k};
   if(k > N)
      plot(boundary(:,2),...
         boundary(:,1),'g','LineWidth',2);
   else
      plot(boundary(:,2),...
         boundary(:,1),'r','LineWidth',2);
   end
end
```



bwmorph

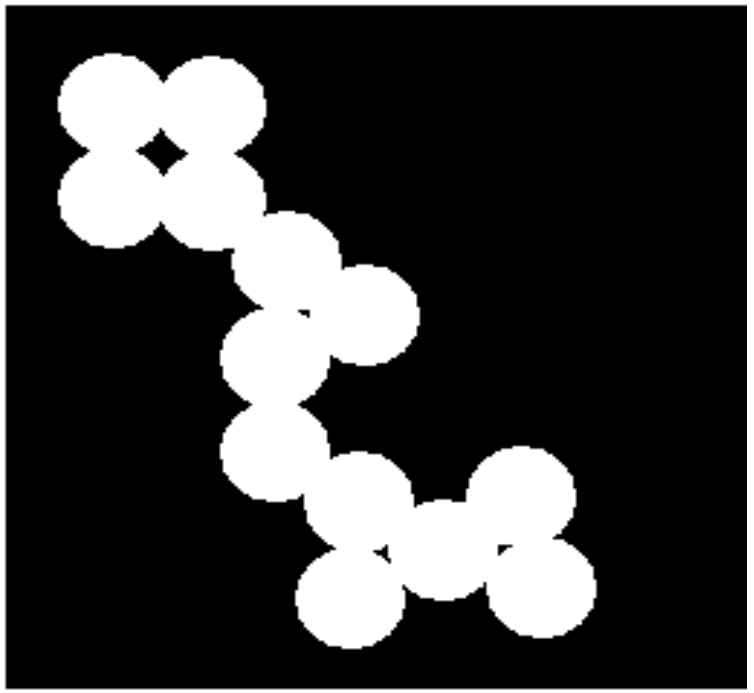
Morphological operations on binary images

'clean'	Removes isolated pixels
'fill'	Fills isolated interior pixels
'remove'	Removes interior pixels.
'skel'	allow removes pixels on the boundaries of objects without allowing objects to break apart.

Example

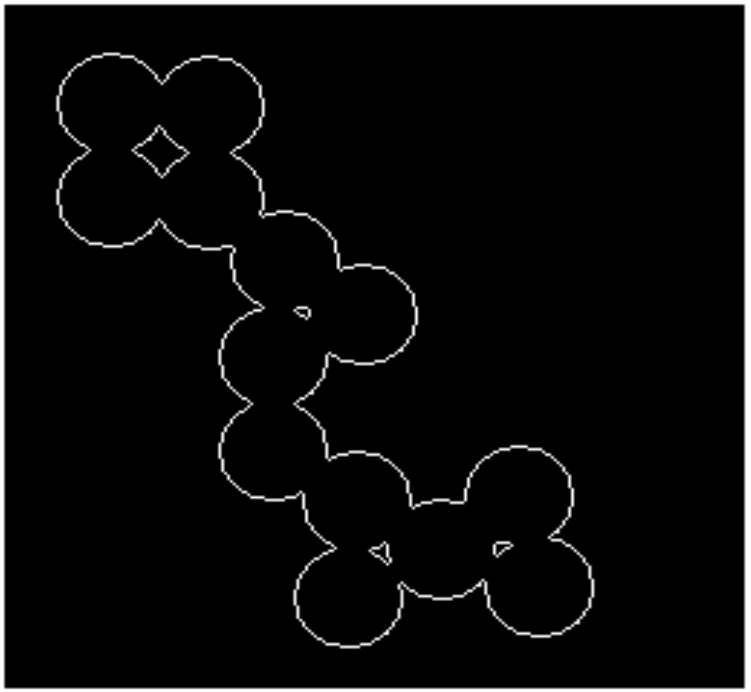
```
BW = imread('circles.png');

imshow(BW);
```



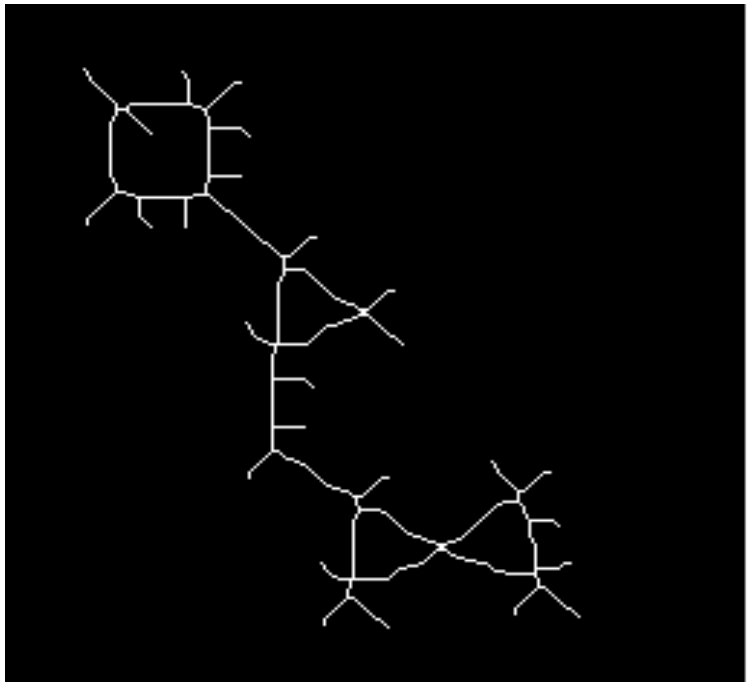
BW2 = bwmorph(BW,'remove');

figure, imshow(BW2)



```
BW3 = bwmorph(BW,'skel',Inf);

figure, imshow(BW3)
```

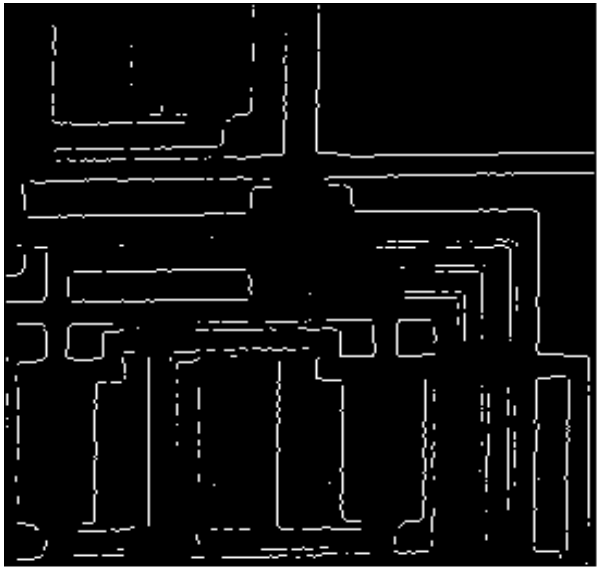


edge

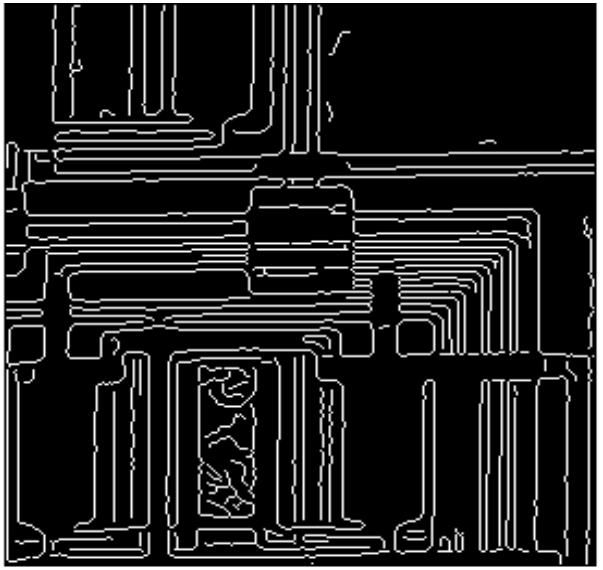
Find edges in grayscale image

Examples: **Find the edges of an image using the Prewitt and Canny methods.**

```
I = imread('circuit.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
imshow(BW1);
figure, imshow(BW2)
```



Prewitt Filter



Canny Filter

gray2ind

Convert grayscale or binary image to indexed image

gray2ind scales, then rounds, an intensity image to produce an equivalent indexed image.

Example

```
I = imread('cameraman.tif');

[X, map] = gray2ind(I, 16);

imshow(X, map);
```

imadjust

Adjust image intensity values

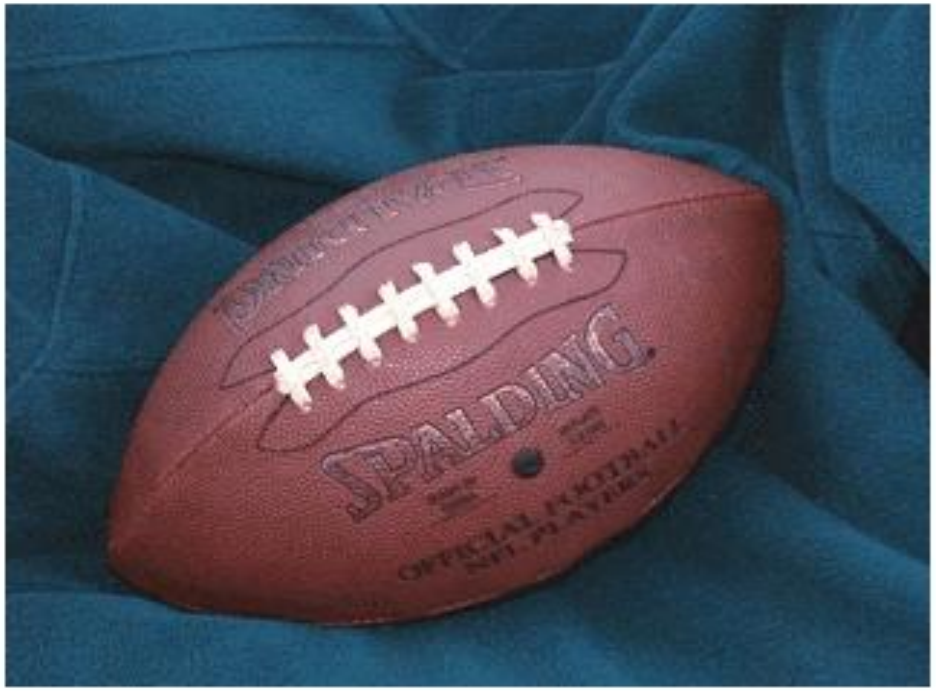
Examples

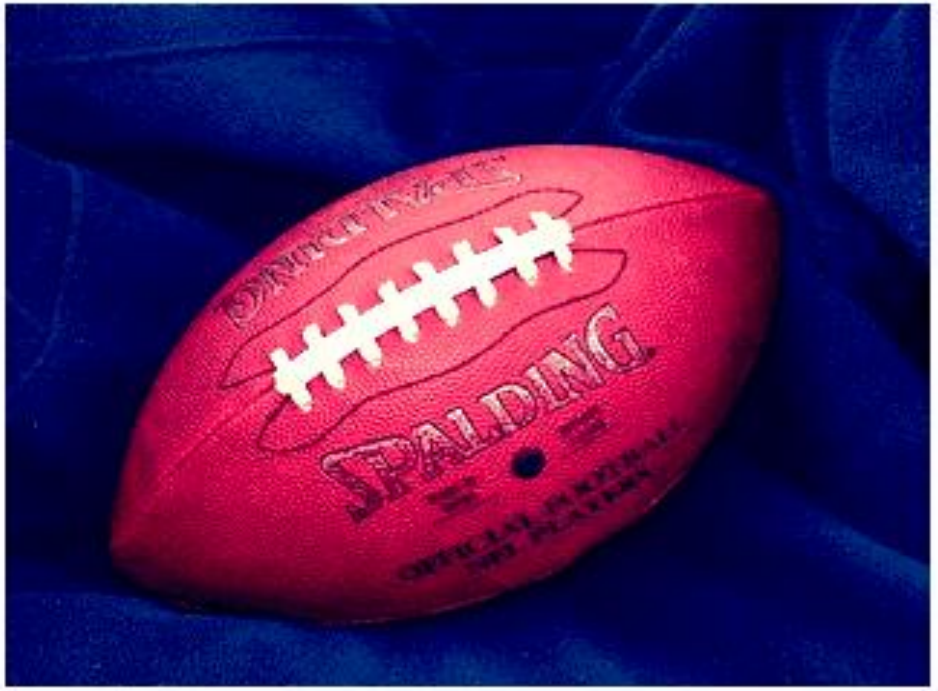
Adjust a low-contrast grayscale image.

```
I = imread('pout.tif');
J = imadjust(I);
imshow(I), figure, imshow(J)
```

Adjust an RGB image.

```
RGB1 = imread('football.jpg');
RGB2 = imadjust(RGB1,[.2 .3 0; .6 .7 1],[]);
imshow(RGB1), figure, imshow(RGB2)
```





imfilter

N-D filtering of multidimensional images

imhist

Display histogram of image data

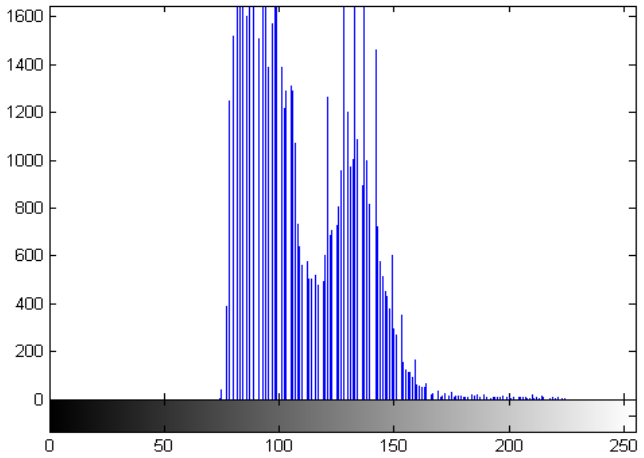
Example

```
I = imread('pout.tif');

imshow(I);

figure(2)

imhist(I)
```



imnoise

Add noise to image

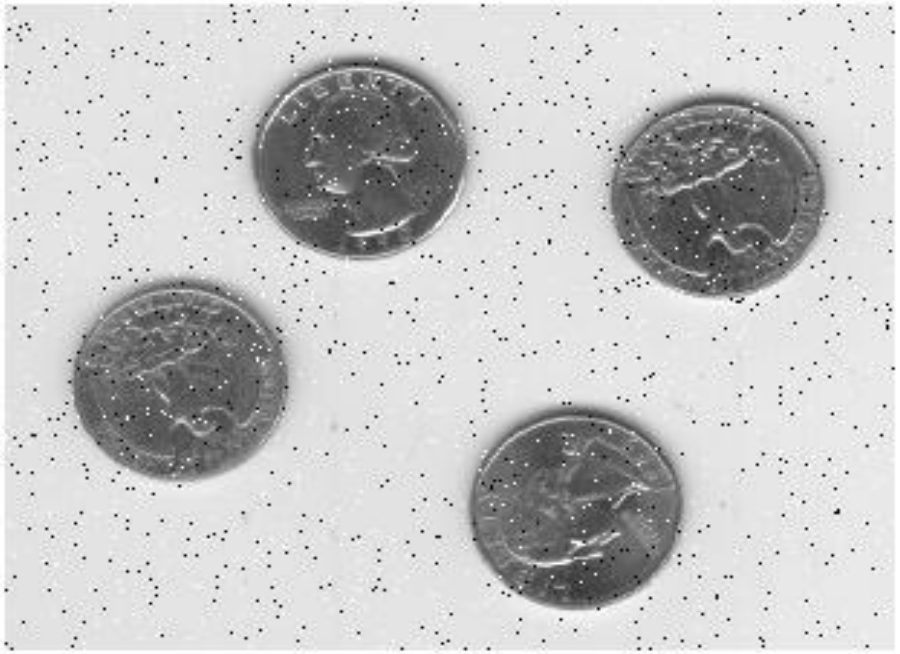
Description

J = imnoise(I, type) adds noise of a given type to the intensity image I. type is a string that specifies any of the following types of noise. Note that certain types of noise support additional parameters. See the related syntax.

Value	Description
`'gaussian'`	Gaussian white noise with constant mean and variance
`'localvar'`	Zero-mean Gaussian white noise with an intensity-dependent variance
`'poisson'`	Poisson noise
`'salt & pepper'`	On and off pixels
`'speckle'`	Multiplicative noise

Examples

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(I)
figure, imshow(J)
```

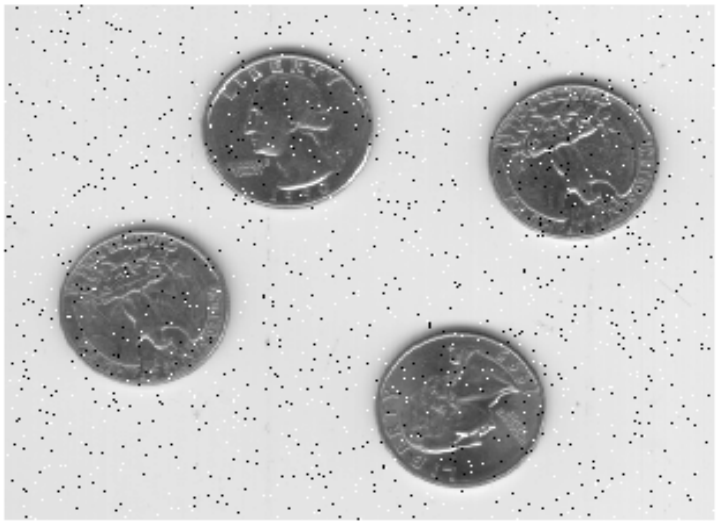


medfilt2
2-D median filtering

Examples

Add salt and pepper noise to an image and then restore the image using `medfilt2`.

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
imshow(J), figure, imshow(K)
```



mat2gray

Convert matrix to grayscale image

mean2

Average or mean of matrix elements

rgb2gray

Convert RGB image or color map to grayscale

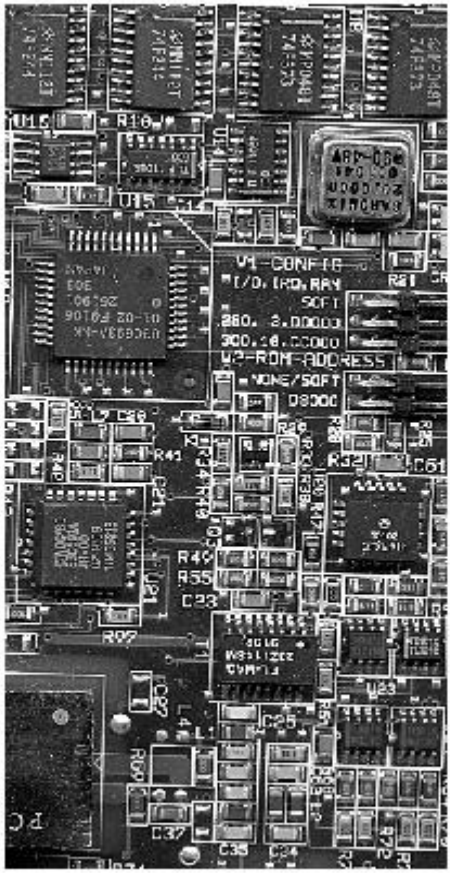
Example

```
I = imread('board.tif');

J = rgb2gray(I);

figure, imshow(I), figure, imshow(J);
```


V1-CONFIG
I/O.IRD.RAM
SOFT
260. 8.00000
300.10.CC000
V2-ROM-ADDRESS
NONE/SOFT
D8000


V1-CONFIG
I/O.IRD.RAM
SOFT
260. 8.00000
300.10.CC000
V2-ROM-ADDRESS
NONE/SOFT
D8000

Some basic concepts for matrices

Identity Matrix

Denoted I (or $I_{n\times n}$).

Examples of identity matrices:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2 x 2

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3 x 3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 x 4

For any matrix A,

$$A \cdot I = I \cdot A = A$$

$m\times n$ $n\times n$ $m\times m$ $m\times n$ $m\times n$

$I_{n\times n}$

Note:

$$AB \neq BA$$ **In general**

$$AI = IA$$

The transpose matrix A^T has elements $(A^T)_{ij} = A_{ji}$.
From the definition of transpose, we have

$$(AB)^T = B^T A^T \qquad \text{(C.1)}$$

The inverse of A, denoted A^{-1},
Satisfies

$$AA^{-1} = A^{-1}A = I. \qquad \text{(C.2)}$$

Because $ABB^{-1}A^{-1} = I$, we have:

$$(AB)^{-1} = B^{-1}A^{-1}. \qquad \text{(C.3)}$$

Because $AA^{-1} = I$

$$(AA^{-1})^T = (A^{-1})^T A^T = I$$

So have:

$$(A^T)^{-1} = (A^{-1})^T \qquad \text{(C.4)}$$

The transpose matrix of A

A’

The inverse of A

inv(A)

Traces and Determinants

Trace and determinant apply to square matrices. The trace Tr(A) of a matrix A is defined as the sum of the elements on the leading diagonal.

$$\mathrm{Tr}(AB) = \mathrm{Tr}(BA).$$

By applying this formula multiple times to the product of three matrices

$$\mathrm{Tr}(ABC) = \mathrm{Tr}(CAB) = \mathrm{Tr}(BCA)$$

The determinant of a product of two matrices is given by

$$|AB| = |A||B|$$

$$\left| A^{-1} \right| = 1 / \left| A \right|$$

trace and det

Matrix Derivatives

Sometimes we need to consider derivatives of vectors and matrices with respect to scalars.

The derivative of a vector with respect to a scalar x is itself a vector . Whose components are given by:

$$\left(\frac{\partial \mathbf{a}}{\partial x}\right)_i = \frac{\partial \mathbf{a}_i}{\partial x}$$

with an analogous definition for the derivative of a vector. Derivatives with respect to vectors and matrices can also be defined, for instance:

$$\left(\frac{\partial x}{\partial \mathbf{a}}\right)_i = \frac{\partial x}{\partial \mathbf{a}_i}$$

$$\frac{\partial}{\partial x}(\mathbf{AB}) = \frac{\partial \mathbf{A}}{\partial x}\mathbf{B} + \mathbf{A}\frac{\partial \mathbf{B}}{\partial x}$$

The derivative of the inverse of a matrix can be expressed as

$$\frac{\partial}{\partial x}\left(A^{-1}\right) = -A^{-1}\frac{\partial A}{\partial x}A^{-1}$$

$$\frac{\partial}{\partial x}\left(A^{-1}\right) = \frac{\partial}{\partial x}\left(A^{-1}\right)\left(AA^{-1}\right)$$

$$= \frac{\partial A^{-1}}{\partial x}\left(AA^{-1}\right) + A^{-1}\frac{\partial AA^{-1}}{\partial x}$$

$$= \frac{\partial A^{-1}}{\partial x} + A^{-1}\frac{\partial A}{\partial x}A^{-1} + A^{-1}A\frac{\partial A^{-1}}{\partial x}$$

$$= \frac{\partial A^{-1}}{\partial x} + A^{-1}\frac{\partial A}{\partial x}A^{-1} + \frac{\partial A^{-1}}{\partial x}$$

$$\Rightarrow \frac{\partial}{\partial x}\left(A^{-1}\right) = -A^{-1}\frac{\partial A}{\partial x}A^{-1}$$

```
>> clear
>> syms a t x;   %定义符号变量
>> f=[a,t^5;t^2*cos(x), log(x)];   %创建符号函数数组
>> f

f =

[          a,      t^5]
[ t^2*cos(x), log(x)]
```

```
>> dfdx=diff(f)      %求矩阵f对x的一阶导数

dfdx =

[             0,    0]
[ -t^2*sin(x), 1/x]
```

```
>> dfdt2=diff(f,t,2)      %求矩阵f对t的二阶导数

dfdt2 =

[         0, 20*t^3]
[ 2*cos(x),       0]

>>
```

Linear Regression & Model Representation

Outline:

1、 Linear regression

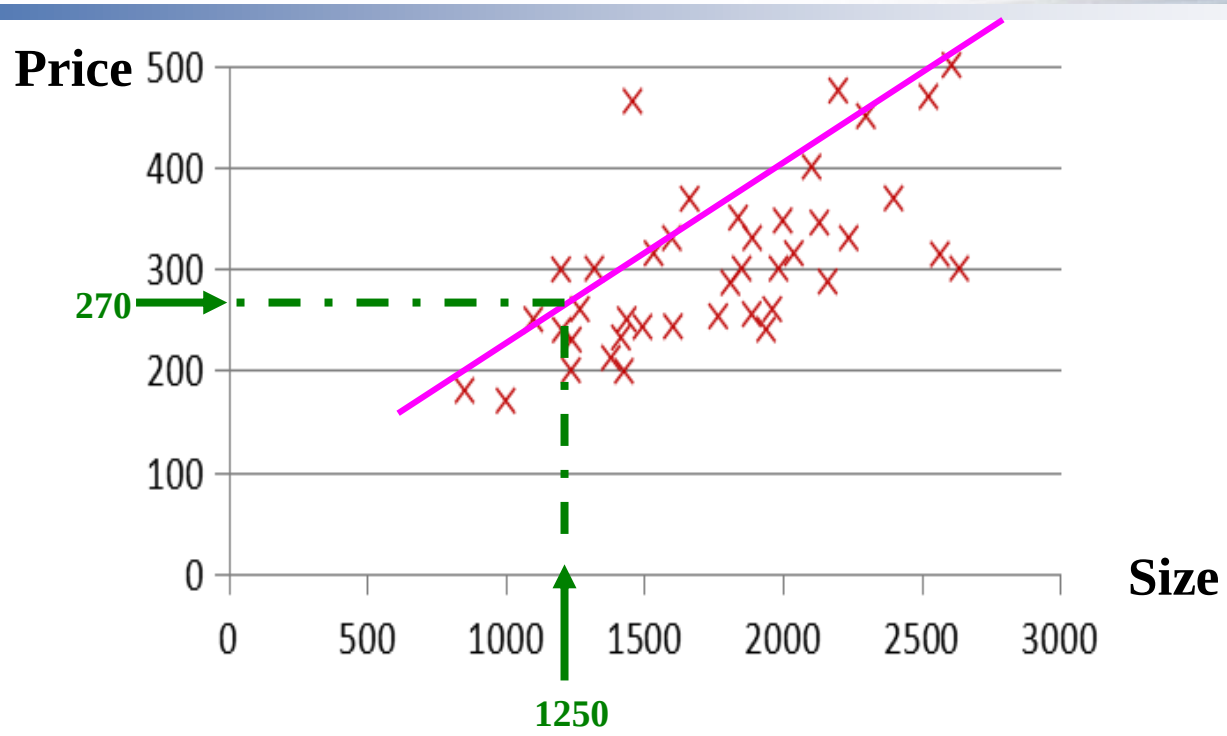
2、 Gradient descent

3、 Normal equations

Training set of housing prices

Size in feet^2 (x)	Price ($) in 1000's ($y$)
2104	460
1416	232
1534	315
852	178
⋮	⋮

N=47

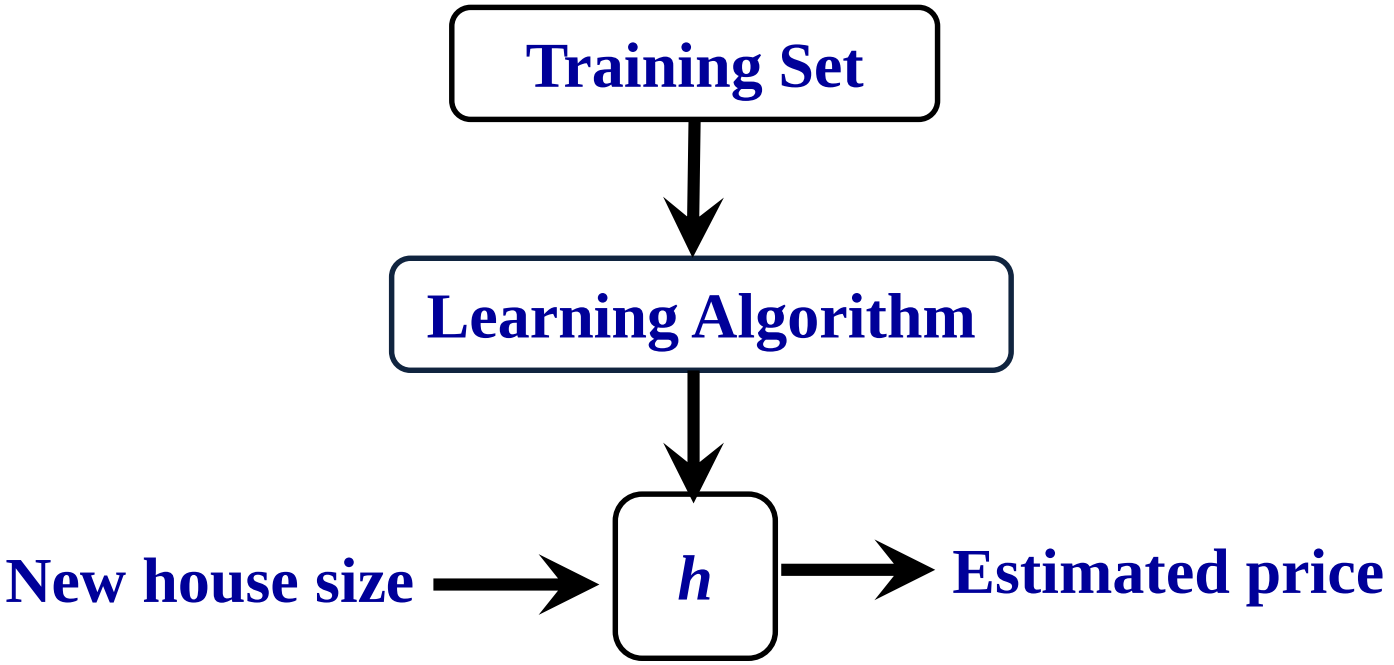


Supervised Learning:

Given the "right answer" for each example in the data.

Regression Problem:

Predict real-valued output



hypothesis

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Here, θ_i (i=0,1) are the parameters

Notation

N=the number of training examples

x=’input’ variables / features

y=’output’ variables / ’target’ variables

(x,y)→training example

i^{th} training example=(x_i, y_i)

Multiple features (variables)

Size (feet^2) $x^{(1)}$	Number of bedrooms $x^{(2)}$	Number of floors $x^{(3)}$	Age of home (years) $x^{(4)}$	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

N=47

Notation:

n: number of features, here n=4

x_i: input (features) of i^{th} training example

$x_i^{(j)}$: the value of feature j in i^{th} training example.

$$x_2 = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_2^{(3)} = 2$$

Hypothesis

Previously,

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$$h_\theta(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_n x^n$$

Here, *n* is the number of features

Hypothesis:

$$h_\theta(X)=\theta^T X= \theta_0 x^0+\theta_1 x^1+\theta_2 x^2+\ldots+\theta_n x^n$$

Parameters:

$X=(x^0,x^1,\ldots,x^n)$,*and* $\theta=(\theta_0,\theta_1,\ldots,\theta_n)$ is $n+1$ dimensional vector

Cost function:

$$J(\theta_0,\theta_1,\ldots,\theta_n)=\frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x_i)-y_i)^2$$

$$J(\theta)=J(\theta_0,\theta_1,\ldots,\theta_n)=\frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x_i)-y_i)^2$$

$$\min_{\theta} imize \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=\min_{\theta} imize\, J(\theta)$$

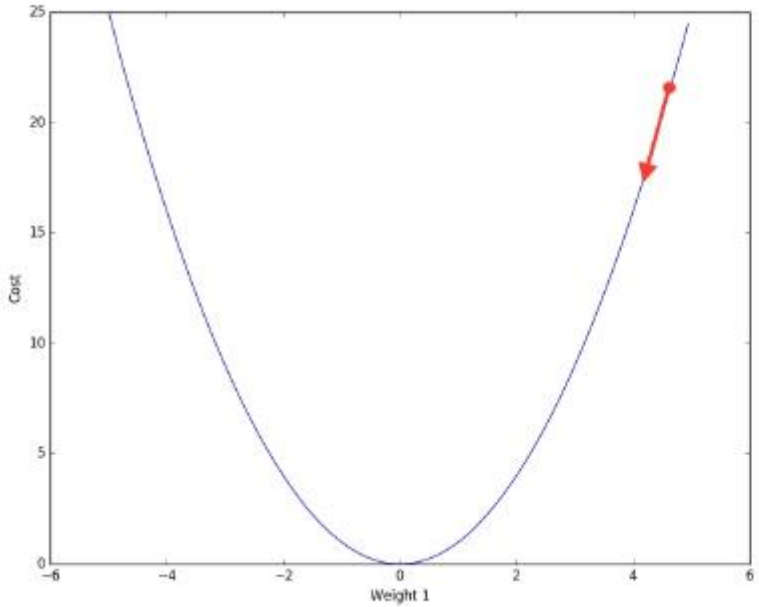
Gradient descent

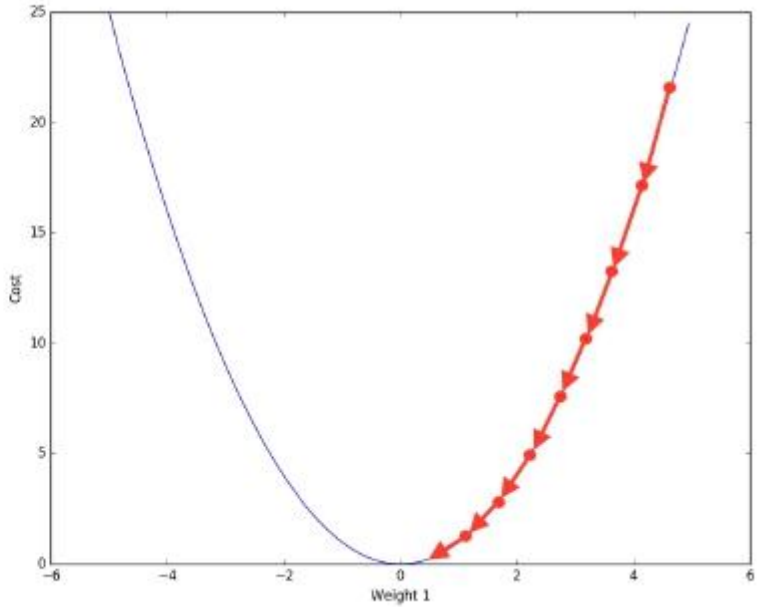
start with some θ, *eg.* $\theta=0$;

keep changing θ reduce $J(\theta)$;

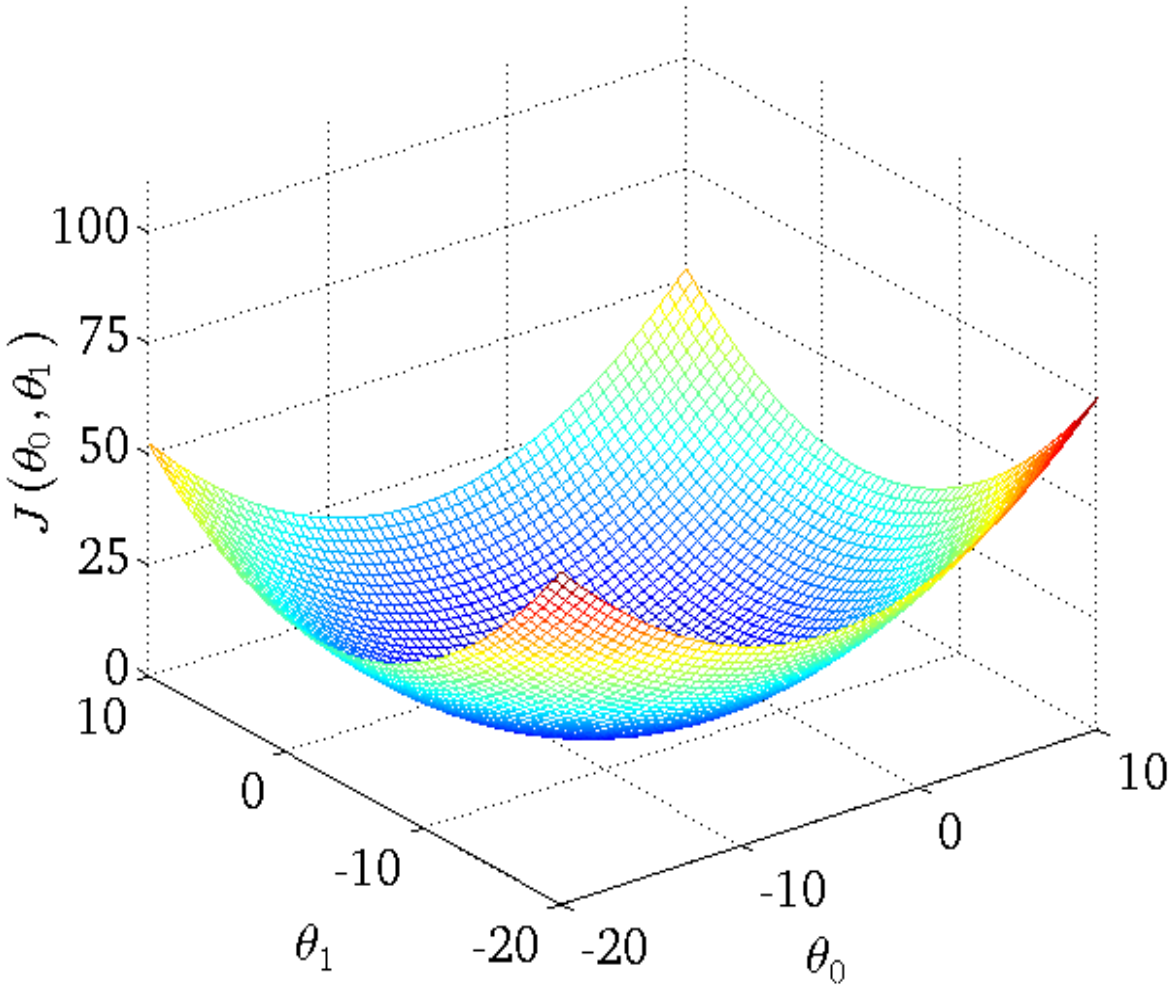
gradient descent

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta) \qquad i = 0, \ldots, n$$



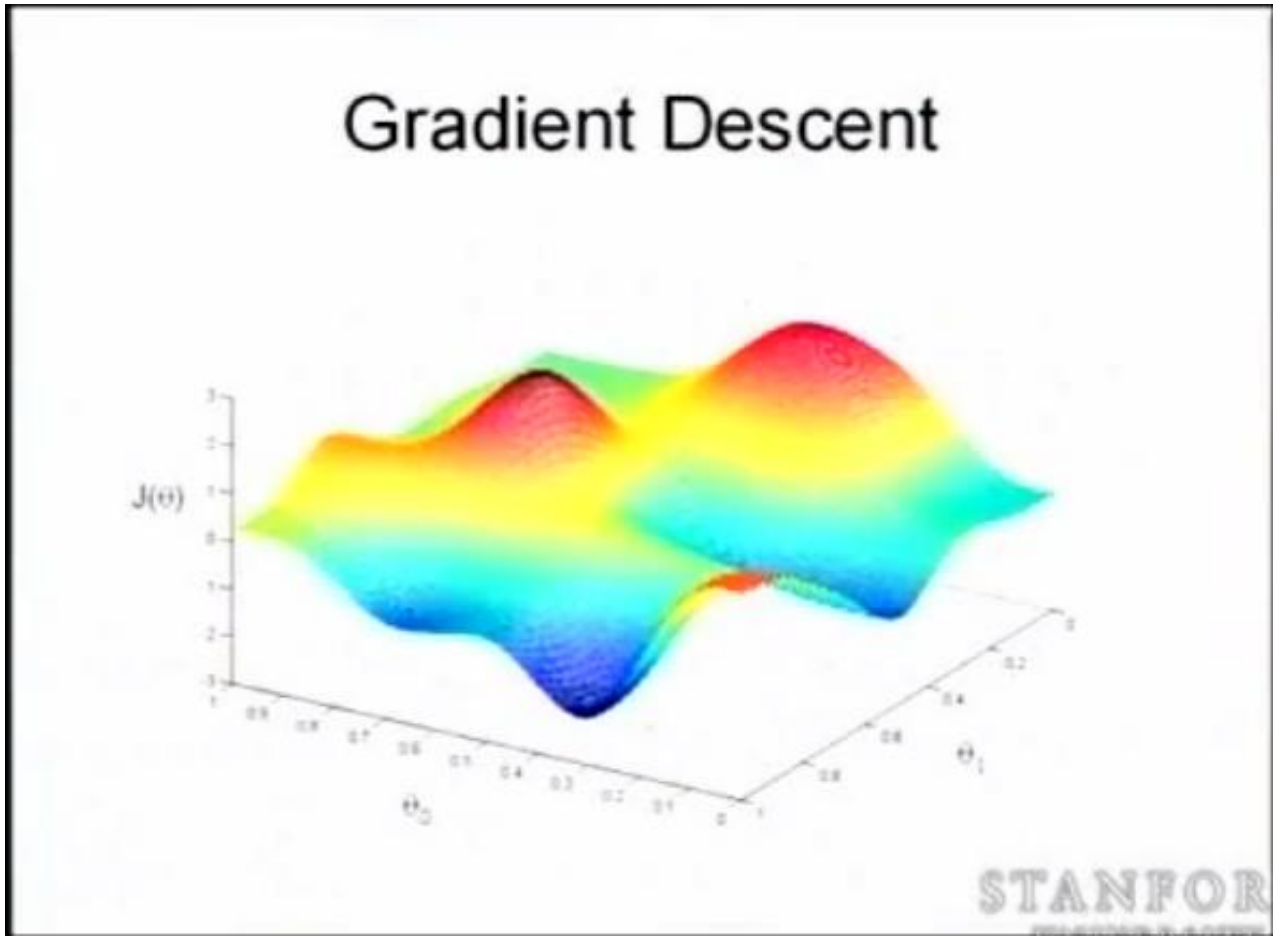


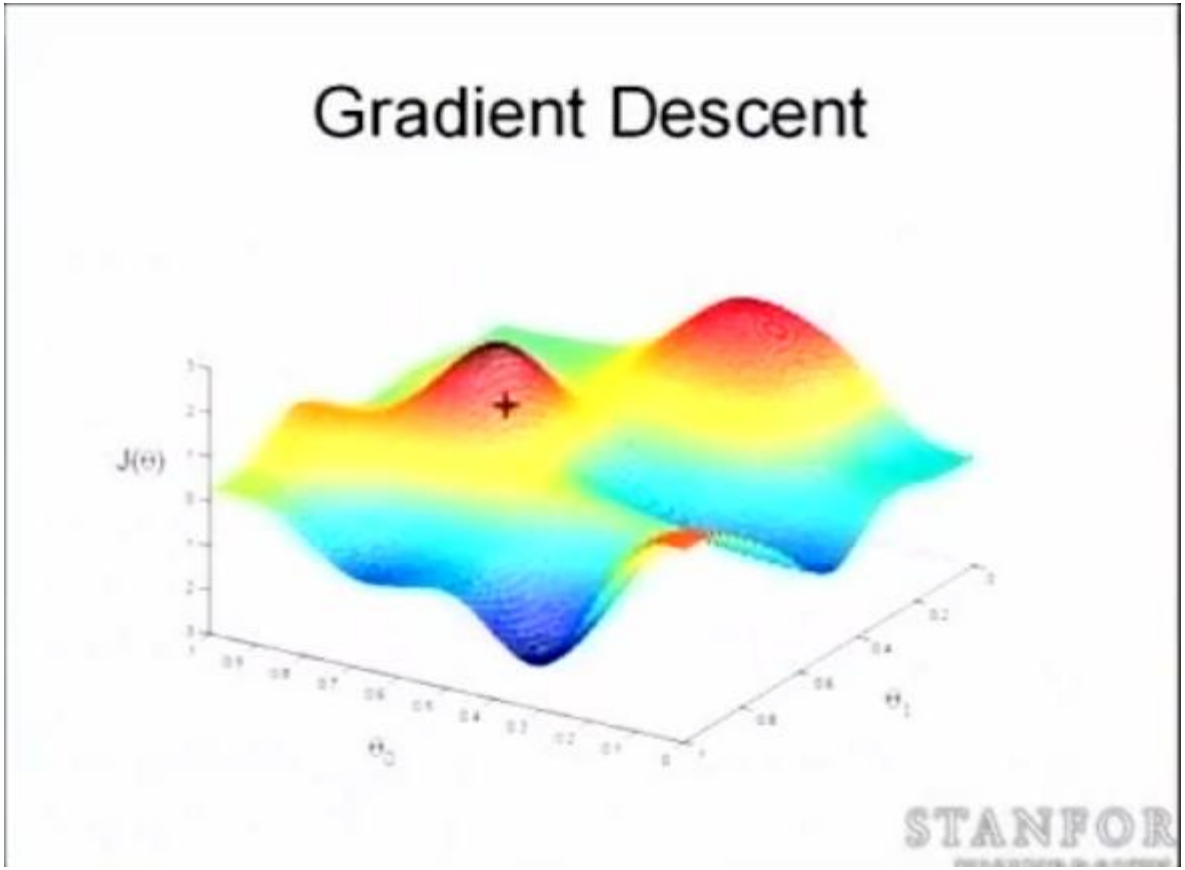
Gradient descent

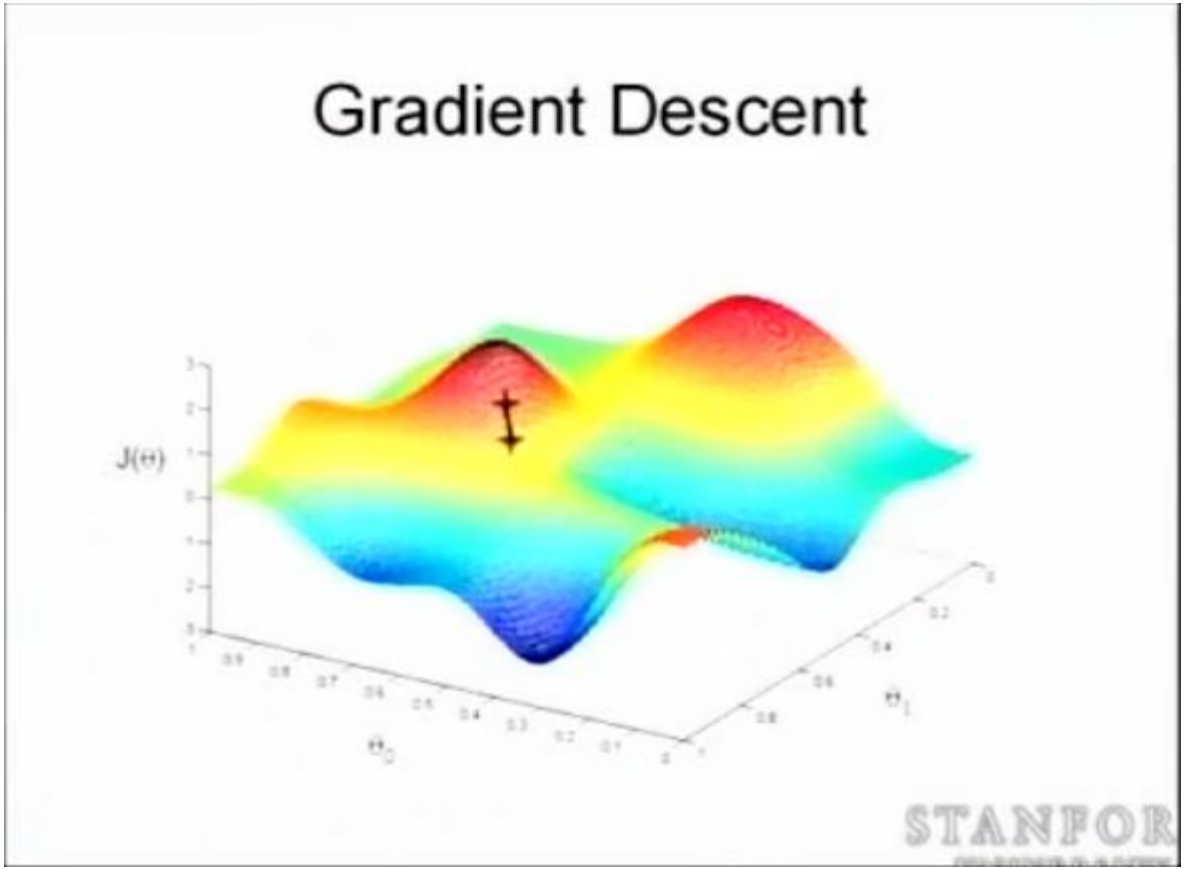


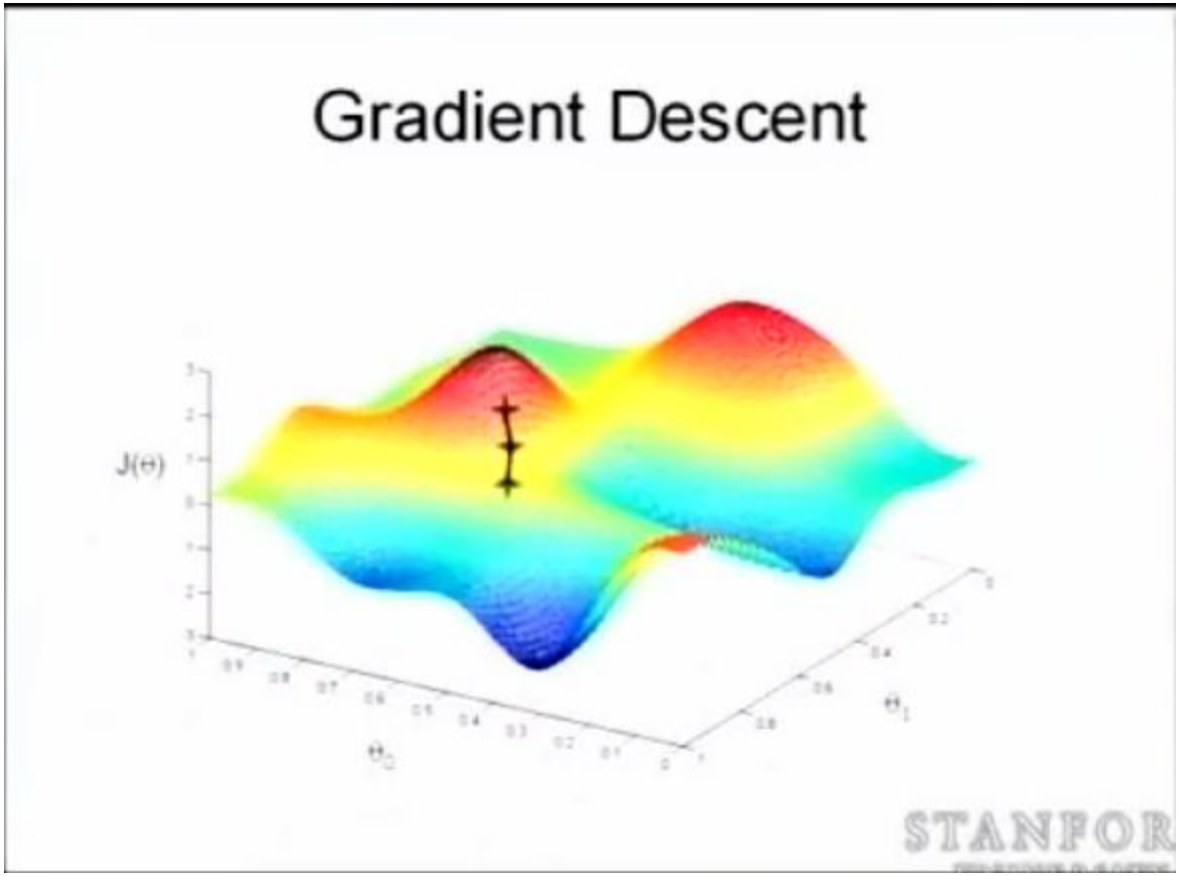
“convex function”

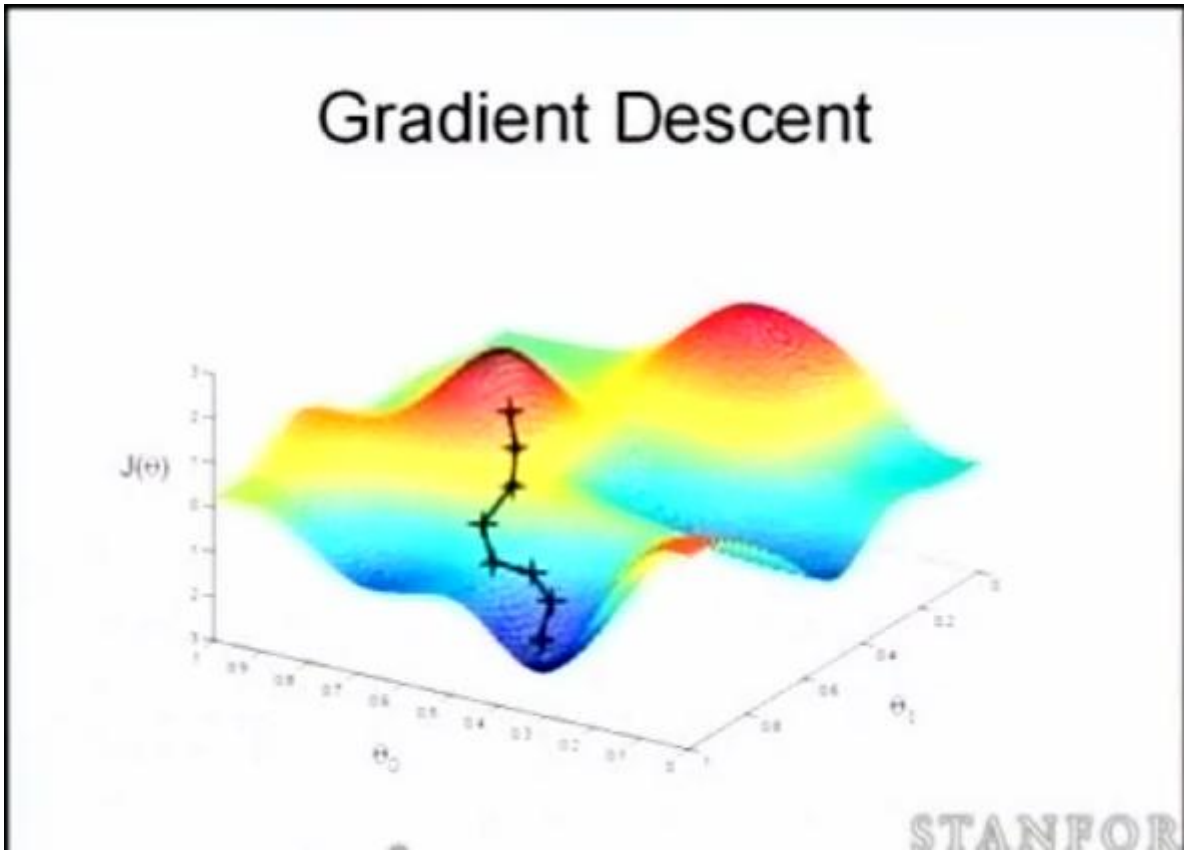
Bowl - shaped

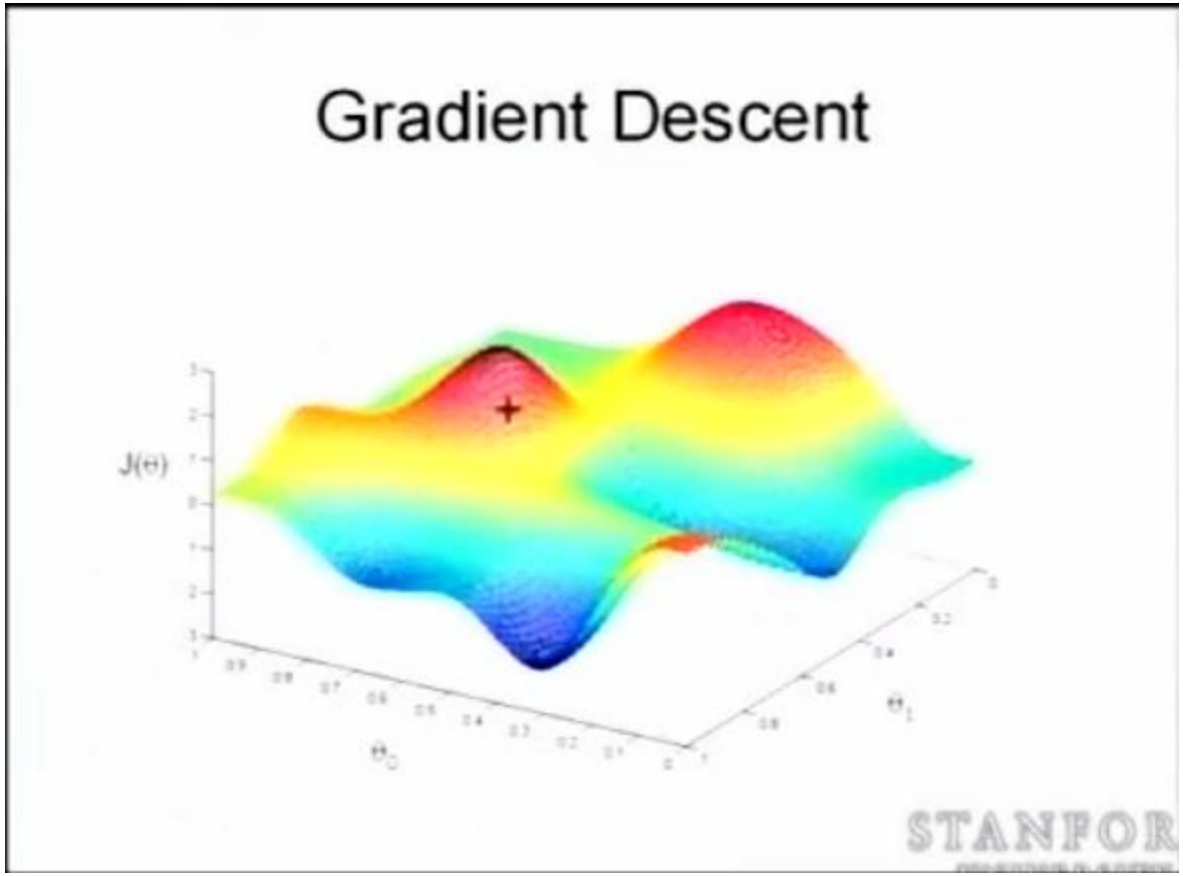

Gradient Descent
J(θ)
θ0
θ1
STANFOR

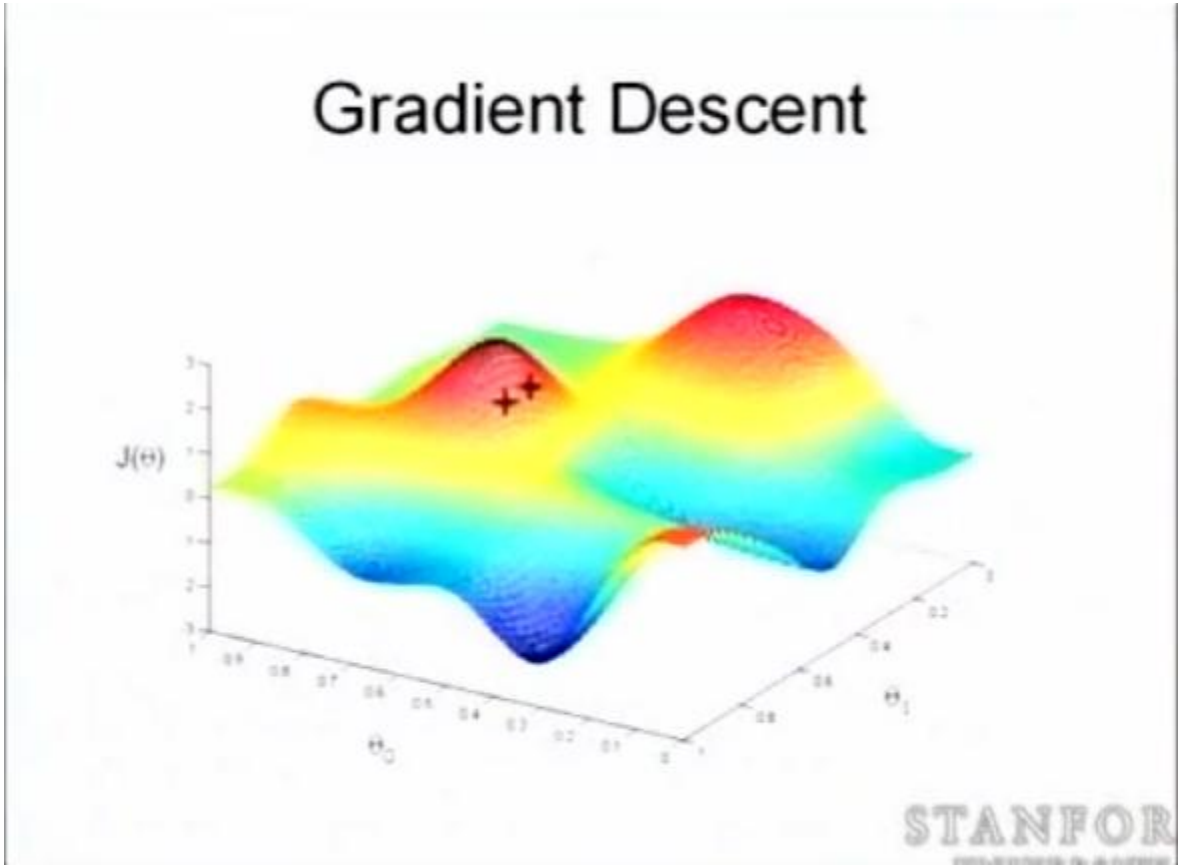

Gradient Descent
J(θ)
θ0
θ1
STANFOR

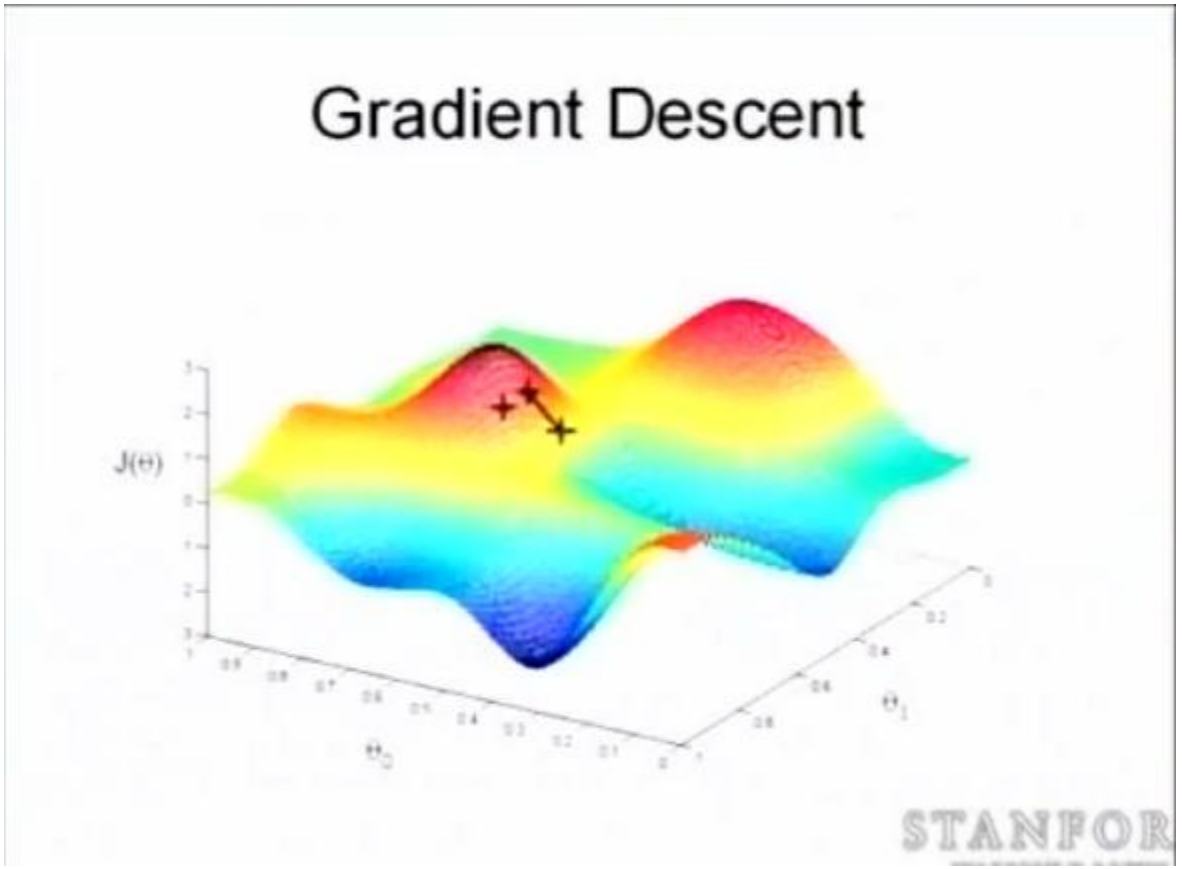

Gradient Descent
J(θ)
θ0
θ1
STANFOR

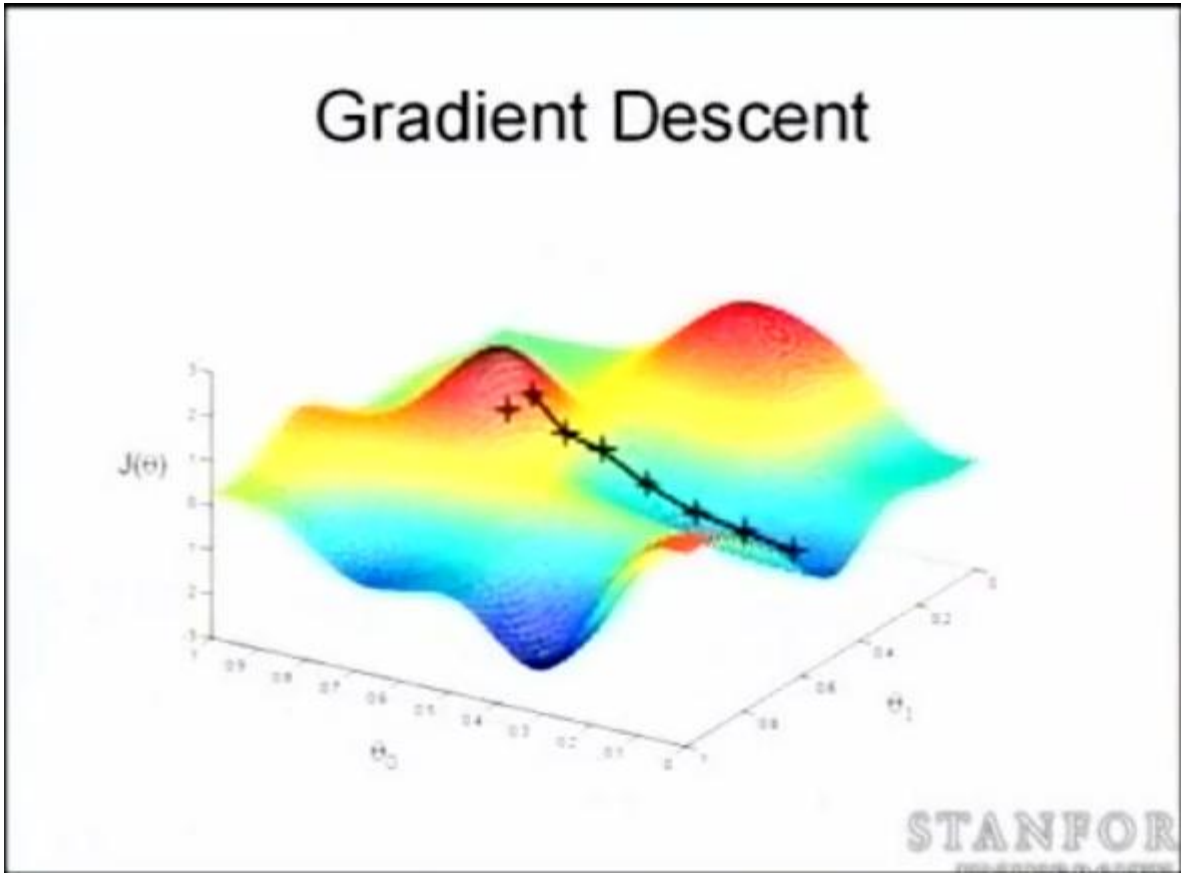

Gradient Descent
J(θ)
θ0
θ1
STANFOR


Gradient Descent
J(θ)
θ0
θ1
STANFOR


Gradient Descent
J(θ)
θ0
θ1
STANFOR


Gradient Descent
$J(\theta)$
θ_0
θ_1
STANFOR

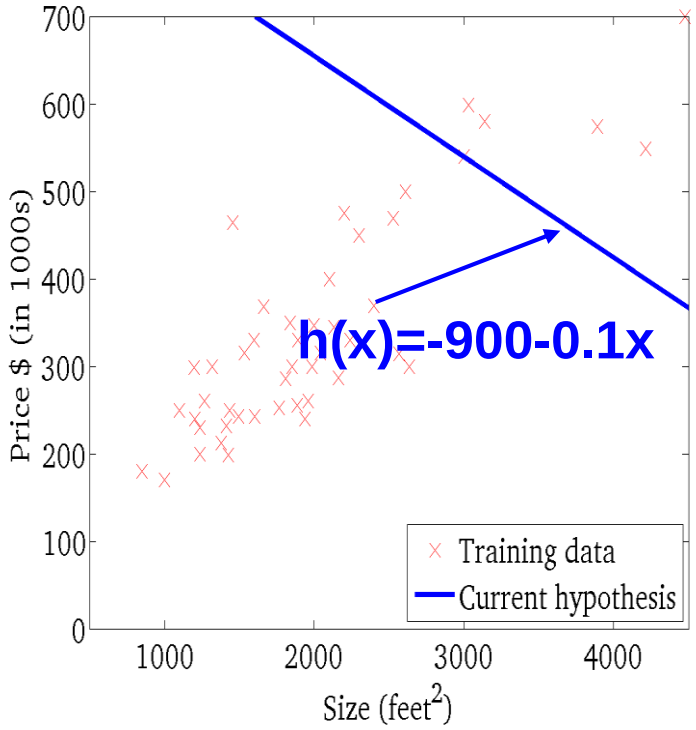

Gradient Descent
J(θ)
θ0
θ1
STANFOR


Gradient Descent
J(θ)
θ0
θ1
STANFOR

Gradient descent

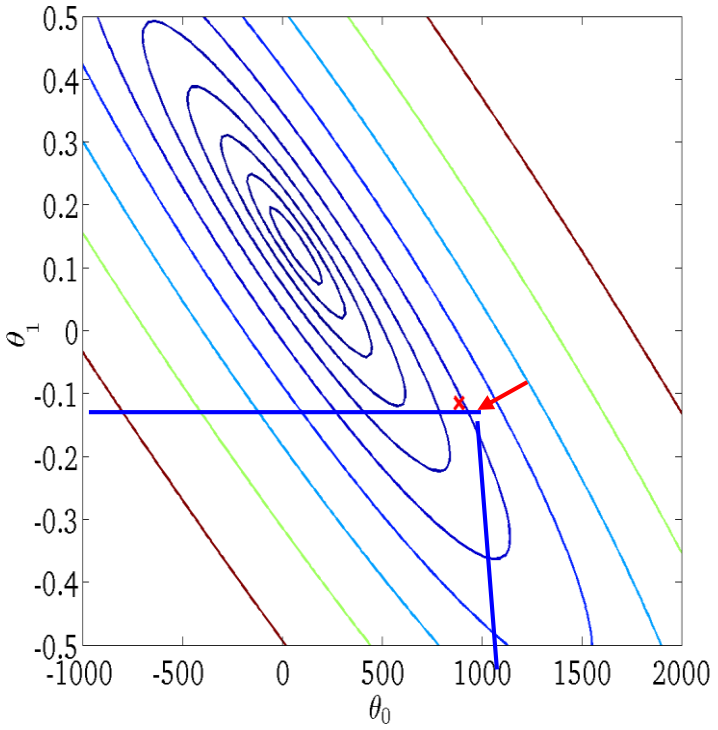
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

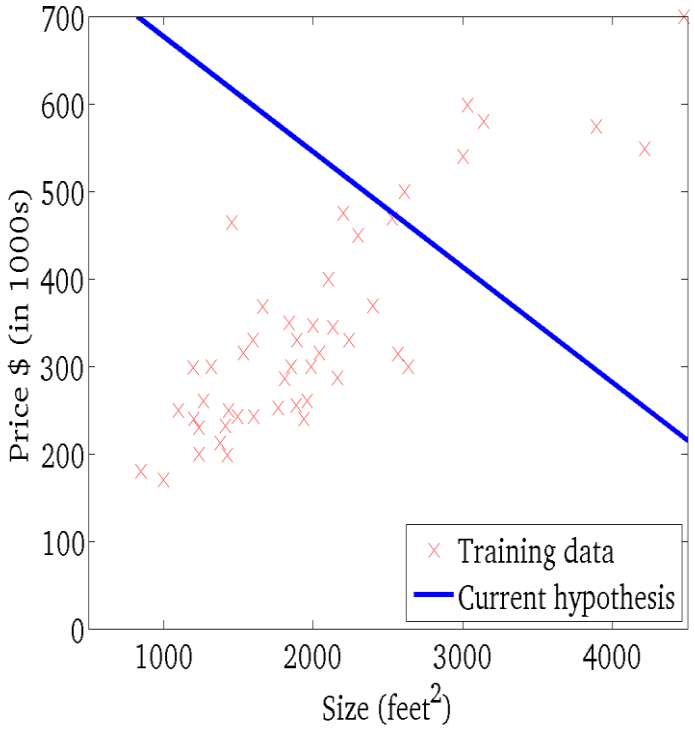
This is a function of the parameters θ_0, θ_1



Gradient descent

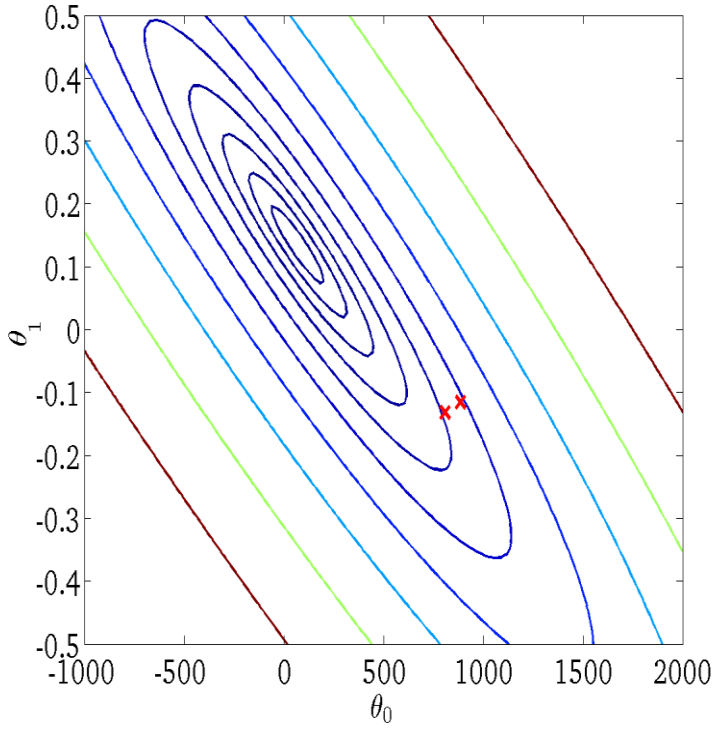
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

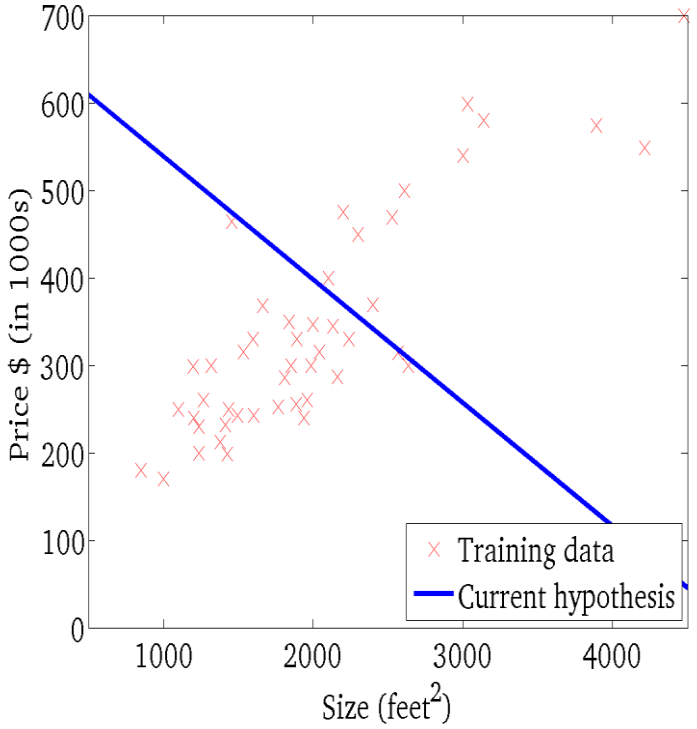
This is a function of the parameters θ_0, θ_1



Gradient descent

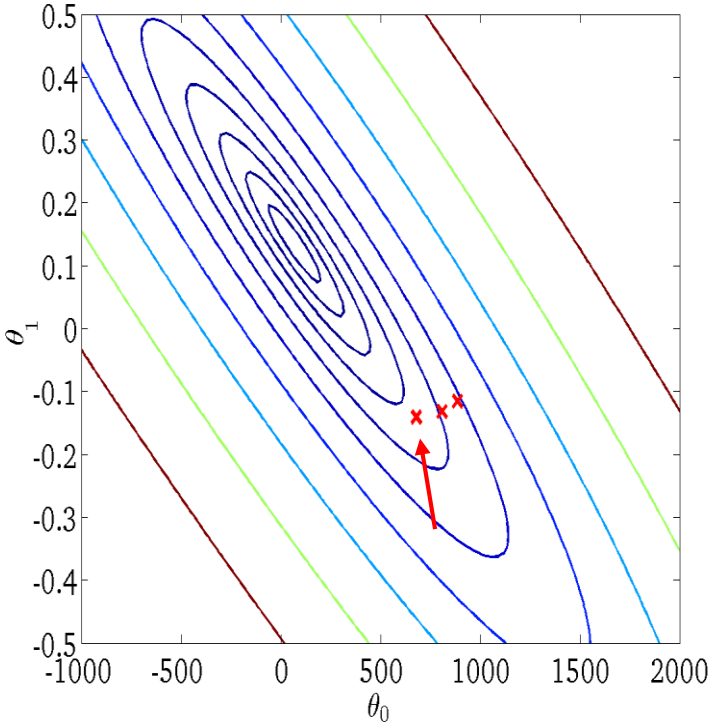
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

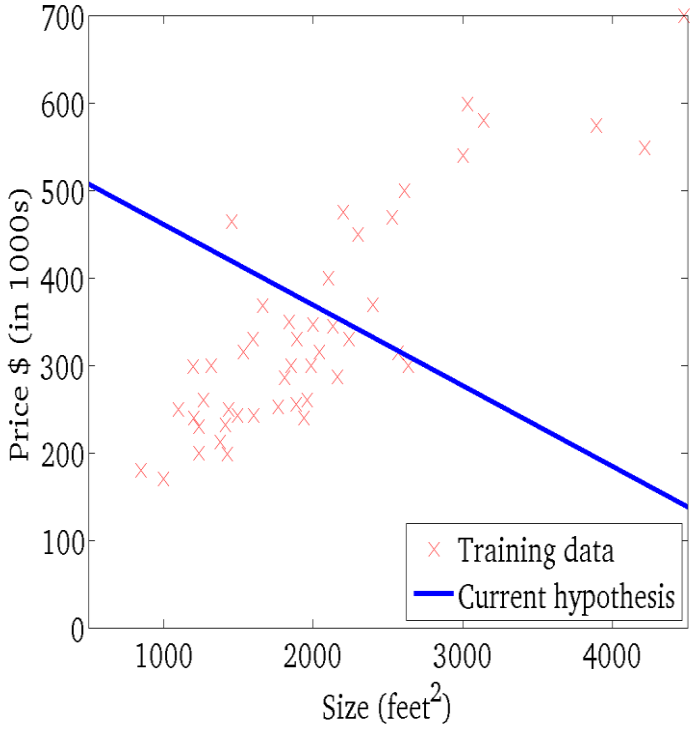
This is a function of the parameters θ_0, θ_1



Gradient descent

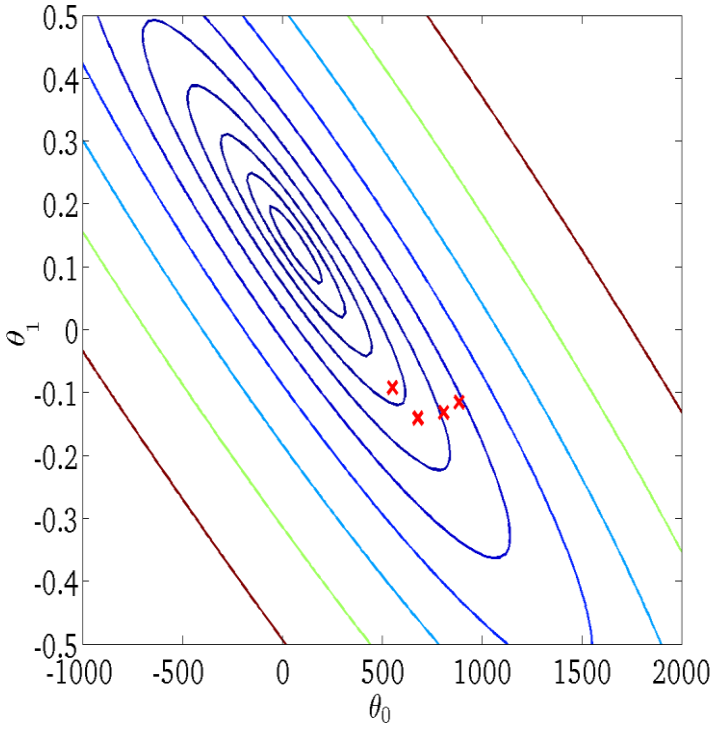
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

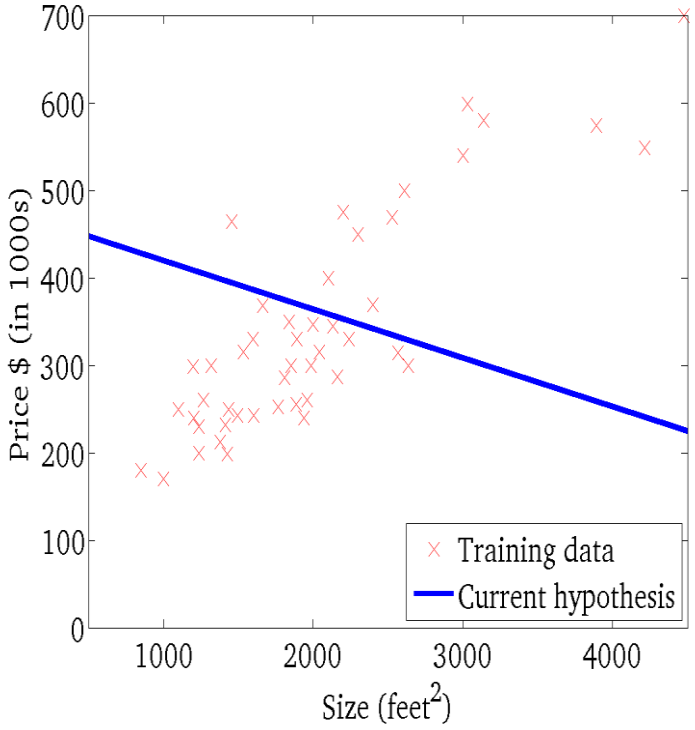
This is a function of the parameters θ_0, θ_1



Gradient descent

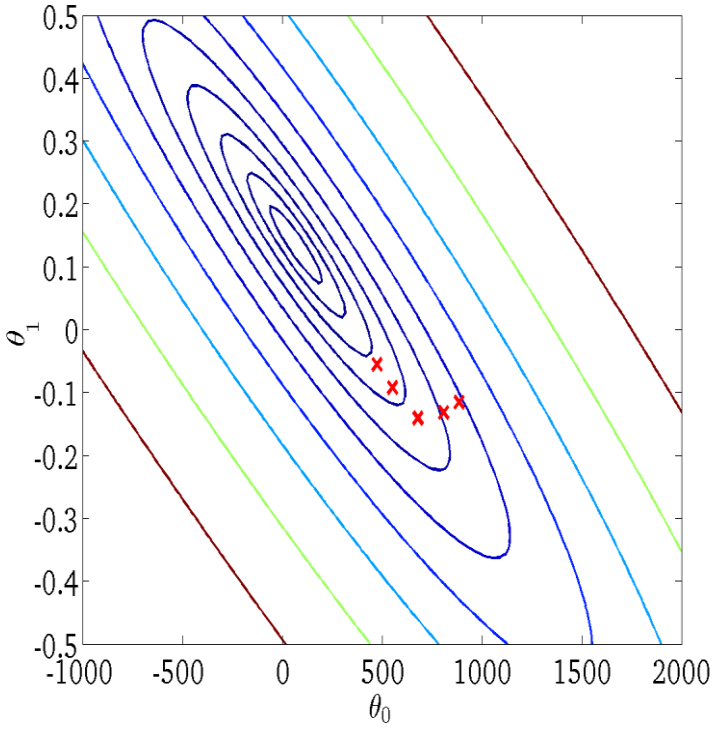
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

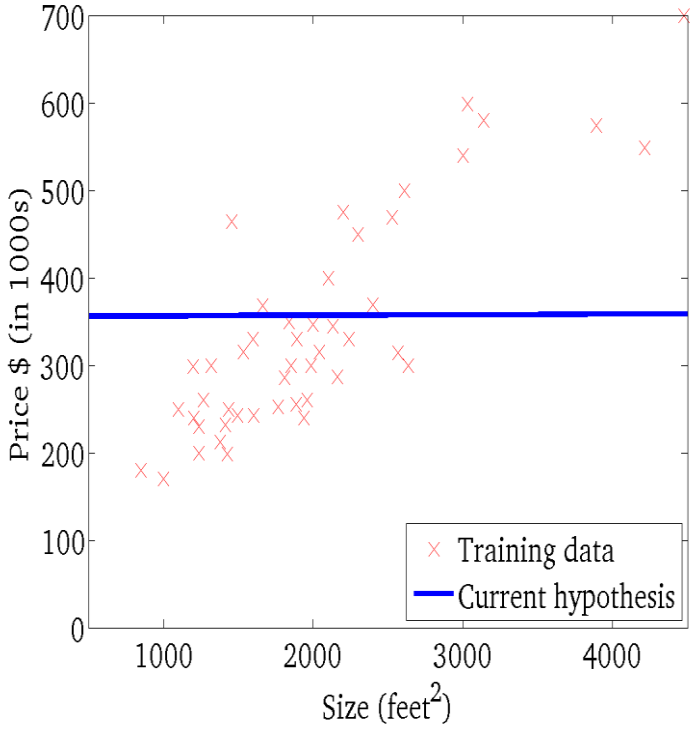
This is a function of the parameters θ_0, θ_1



Gradient descent

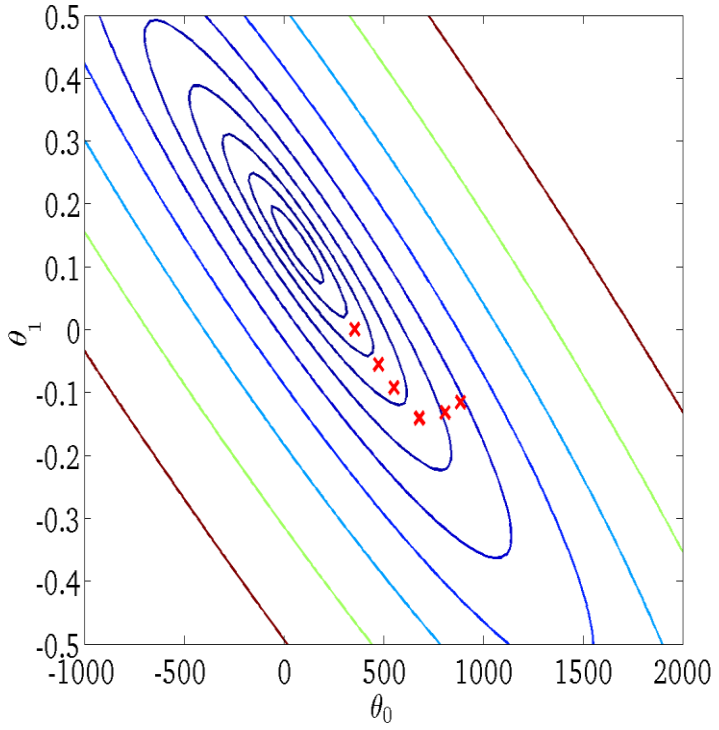
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

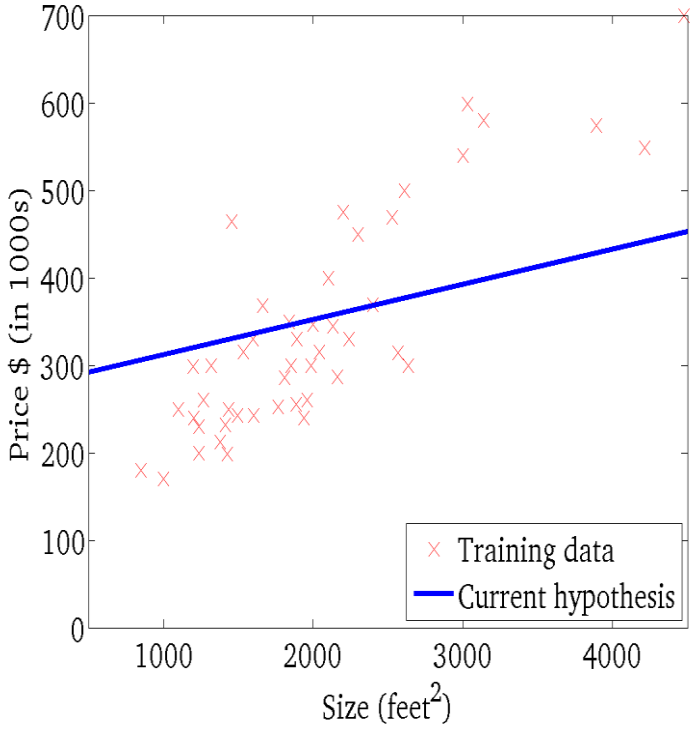
This is a function of the parameters θ_0, θ_1



Gradient descent

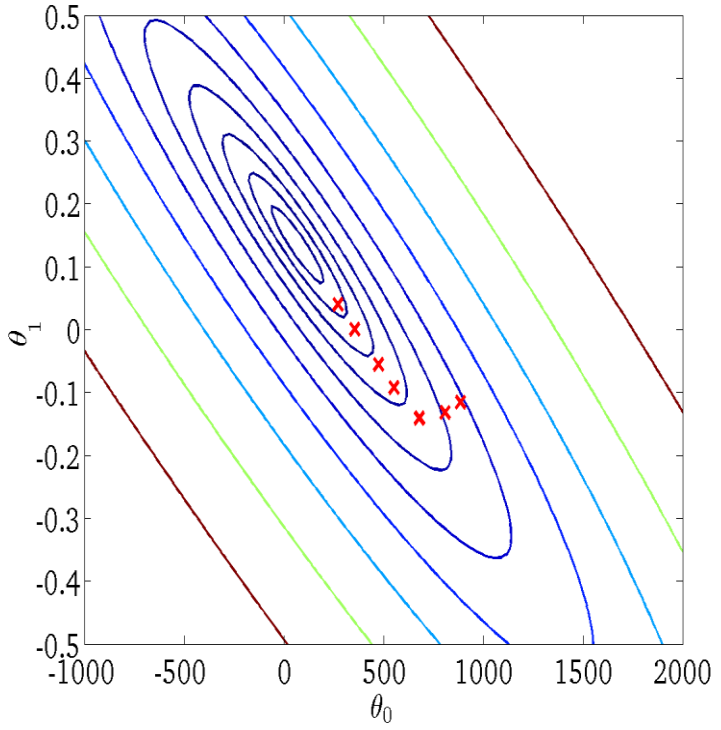
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

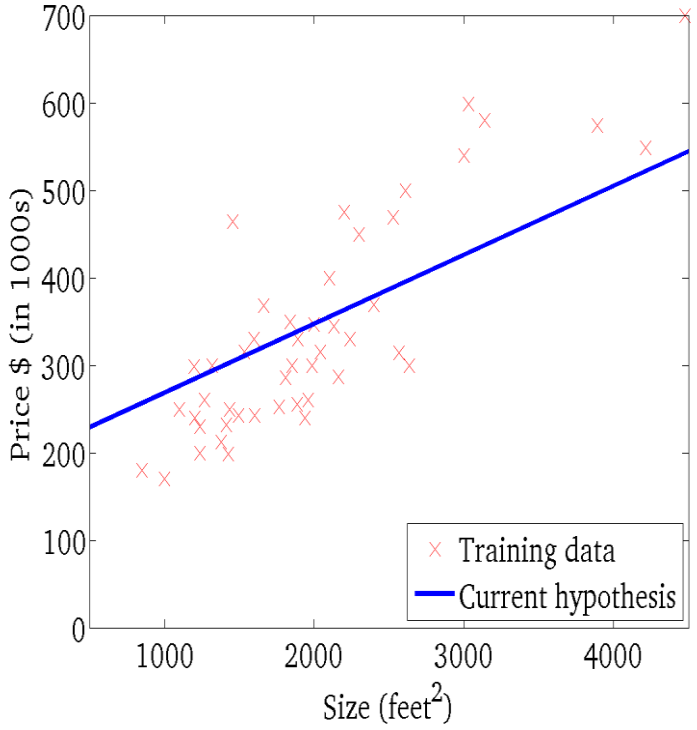
This is a function of the parameters θ_0, θ_1



Gradient descent

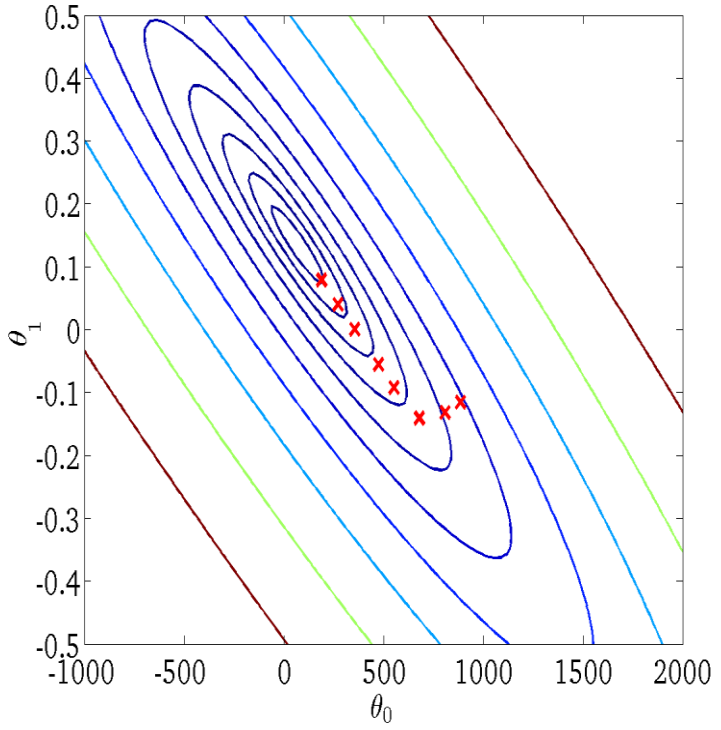
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

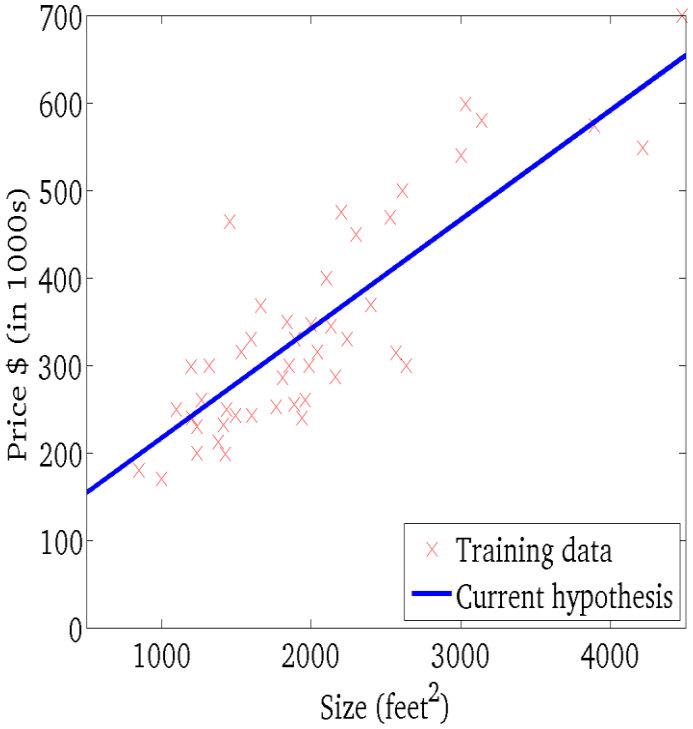
This is a function of the parameters θ_0, θ_1



Gradient descent

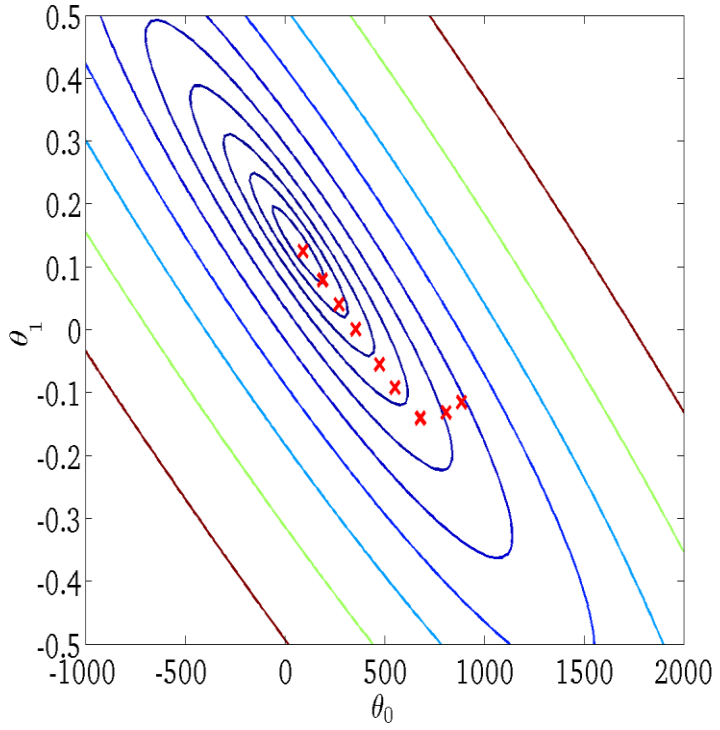
$h_\theta(x)$

For fixed θ_0, θ_1, this is a function of x



$J(\theta_0, \theta_1)$

This is a function of the parameters θ_0, θ_1



Gradient descent

Gradient of Cost function

$$\frac{\partial}{\partial\theta_i}J(\theta)=\frac{\partial}{\partial\theta_i}\frac{1}{2}\left(h_\theta(x)-y\right)^2 \qquad N=1$$

$$=2\times\frac{1}{2}\left(h_\theta(x)-y\right)\frac{\partial}{\partial\theta_i}\left(h_\theta(x)-y\right)$$

$$=\left(h_\theta(x)-y\right)\frac{\partial}{\partial\theta_i}\left(\theta_0x^0+\ldots+\theta_ix^i\ldots+\theta_nx^n-y\right)$$

$$=\left(h_\theta(x)-y\right)x^i$$

Updated:

$$\theta_i:=\theta_i-\alpha\left(h_\theta(x)-y\right)x^i$$

Repeat until converge

Gradient descent

$$\theta_i := \theta_i - \alpha \underbrace{\sum_{j=1}^{N} \left(h_\theta \left(x_j \right) - y_j \right) x_j^{(i)}}_{\frac{\partial}{\partial \theta_i} J(\theta)}$$

$$\nabla_\theta J = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix} \in R^{n+1}$$

gradient descent:

$$\theta := \theta - \alpha \nabla_\theta J$$

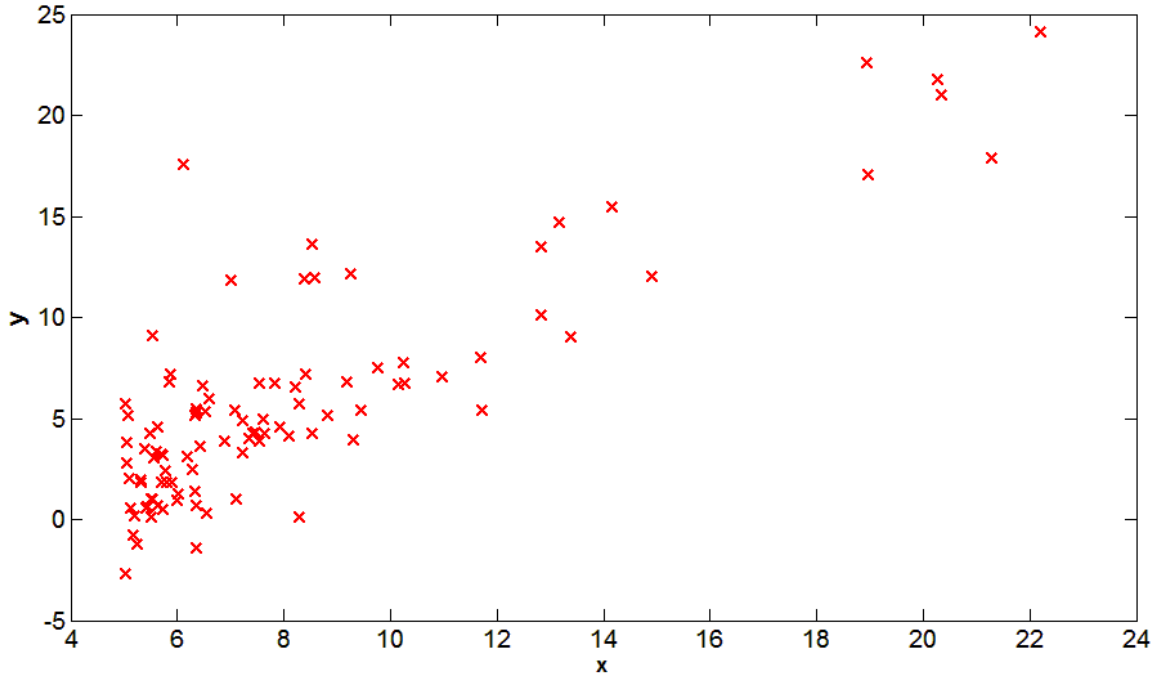
θ: R^{n+1} ; $\nabla_\theta J$: R^{n+1}

Example1_ grad_desent.m

```
%梯度下降
% 加载数据和数据预处理
clear;
 data = load ('data1.txt');
% data = load('data3.txt');
%  data = data';
fprintf('Running Gradient Descent ...\n');
%
 X = data(:, 1);    %取X集合
 y = data(:, 2);        %取Y集合
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% data = load('data3.txt');
% data = data';
% data1=load('data4.txt');
% X=data;
% X = [ones(length(data),1),data];
% y=data1;
```

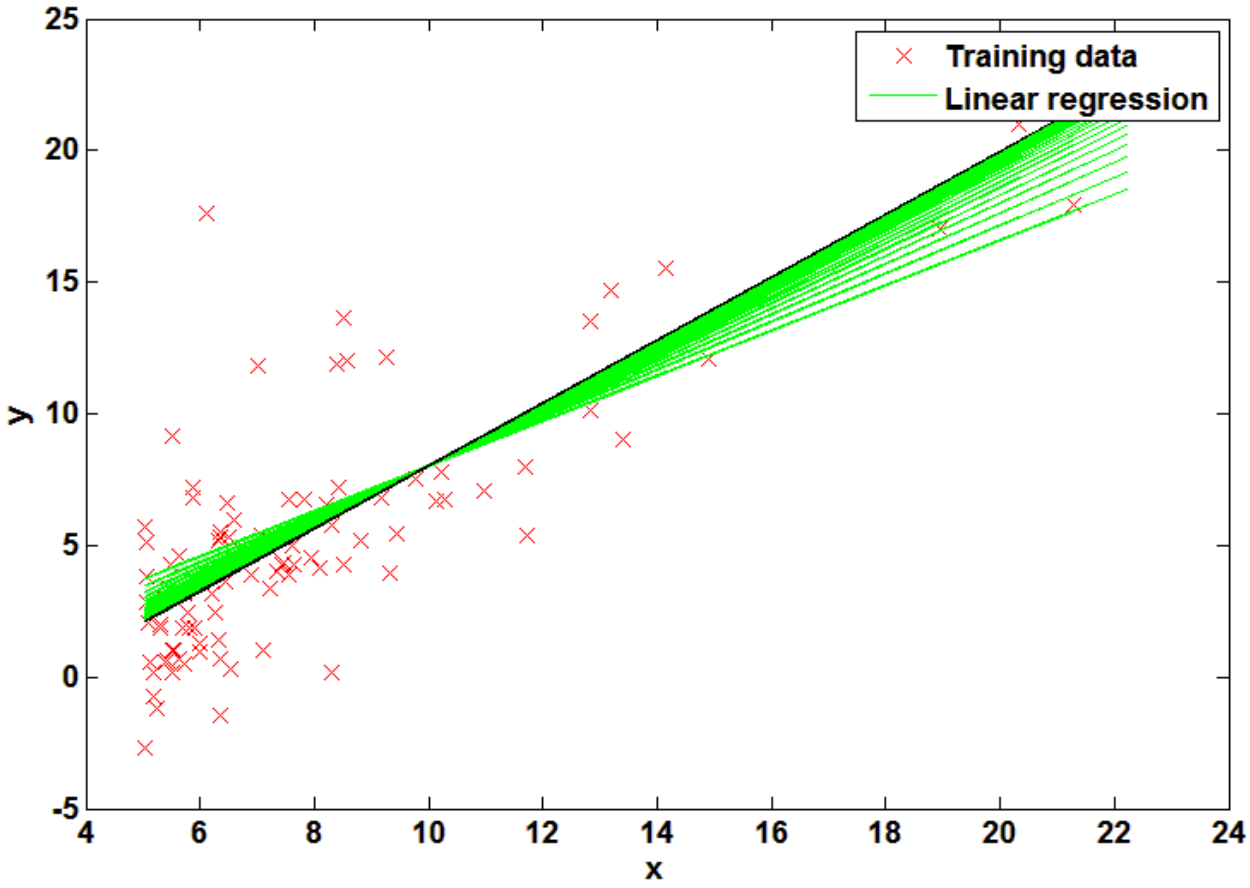
data1

```
6.1101,17.592
5.5277,9.1302
8.5186,13.662
7.0032,11.854
5.8598,6.8233
8.3829,11.886
7.4764,4.3483
8.5781,12
6.4862,6.5987
5.0546,3.8166
5.7107,3.2522
14.164,15.505
5.734,3.1551
8.4084,7.2258
5.6407,0.71618
5.3794,3.5129
6.3654,5.3048
5.1301,0.56077
```


25
20
15
10
5
0
-5
y
4
6
8
10
12
14
16
18
20
22
24
x

```
32 -     m = length(y);  %%数据组数
33 -     J_history = zeros(max_iter,1);  %%初始化迭代误差变量?
34       % iter = 1;
35 -     for iter = 1:max_iter
36           %每迭代100次画一条曲线
37 -         if  mod(iter,100) == 0
38 -             plot(X(:,2), X*theta, 'g')
39 -         end
40
41 -         theta = theta - alpha / m * X' * (X * theta - y);   %梯度下降
42 -         J_history(iter) = sum((X * theta-y) .^ 2) / (2*m);  %记录每次迭代后的全局误差
43 -         fprintf('iter:%d ------ Error:%f\n',iter,J_history(iter));
44 -     end
46       %输出最终的theta值和最终的拟合曲线?disp('Theta found by gradient descent:');
47 -     disp(theta);
48 -     plot(X(:,2), X*theta, 'k')
49 -     legend('Training data', 'Linear regression')
50       % hold off
```

```
>> grad_desent
Running Gradient Descent ...
iter:1 ------ Error:6.737190
iter:2 ------ Error:5.931594
iter:3 ------ Error:5.901155
iter:4 ------ Error:5.895229
   ........
iter:4998 ------ Error:4.476971
iter:4999 ------ Error:4.476971
iter:5000 ------ Error:4.476971
   -3.8953
    1.1930
```


Training data
Linear regression
25
20
15
10
5
0
-5
y
4
6
8
10
12
14
16
18
20
22
24
x

Polyfit
polynomial curve fitting

```
clear all;
close all;
data = load ('data1.txt');
 X = data(:, 1);          %取X集合
 y = data(:, 2);                  %取y集合
%  X = [ones(length(data),1),data(:,1)];
 p = polyfit(X,y,1)%调用matlab自带多项式拟合，1代表a+bx型
```

Normal equations

For the function of matrix A, $f(A)$ and $A \in R^{m \times n}$,

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

If $A \in R^{n \times n}$, the trace of A is $\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$

Normal equations

In fact,

$tr\ (AB) = tr\ (BA)$

$tr\ (ABC) = tr\ (CAB) = tr\ (BCA)$

$\nabla_A\, tr\ (AB) = B^T$

$tr\ (A) = tr\ (A^T)$

If $a \in \mathrm{R}$, $tr\ (a) = a$

$\nabla_A\, tr\ (ABA^TC) = CAB + C^TAB^T$

Normal equations

$$tr(AB) = tr\left(\begin{bmatrix} \leftarrow & a_{1i} & \rightarrow \\ \leftarrow & a_{2i} & \rightarrow \\ \leftarrow & \vdots & \rightarrow \\ \leftarrow & a_{ni} & \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & \uparrow & \uparrow \\ b_{i1} & b_{i2} & \cdots & b_{in} \\ \downarrow & \downarrow & \downarrow & \downarrow \end{bmatrix}\right)$$

$$= \sum_{i=1}^{n} a_{1i} b_{i1} + \sum_{i=1}^{n} a_{2i} b_{i2} + \cdots + \sum_{i=1}^{n} a_{ni} b_{in}$$

$$\Rightarrow \frac{\partial tr(AB)}{\partial a_{ij}} = b_{ji}$$

$$\Rightarrow \nabla_A tr(AB) = B^T$$

$\nabla_A tr(ABA^TC) = CAB+C^TAB^T$

tr（ABA^TC）对A求导，A出现过两次，相当于是函数乘积求导。先固定BA^TC，然后对第一个A求导；固定AB和C，对第二个A求导（注意，在固定AB和C时，先利用tr（ABC）=tr（BCA）=tr（CAB）这一结论，将tr（ABA^TC）等价变形为tr（$CABA^T$）;最后将两部分结果相加。具体过程如下：

$$\nabla_A tr(ABA^TC) = \nabla_A tr[A(BA^TC)] + \nabla_A tr[(CAB)A^T]$$

$$= (BA^TC)^T + \nabla_A tr\left[(CAB)A^T\right]^T$$（注：利用$\nabla_A tr(AB) = B^T$和$tr(A) = tr(A^T)$）

$$= C^TAB^T + \nabla_A tr[A(CAB)^T]$$

$$= C^TAB^T + \left((CAB)^T\right)^T$$

$$= C^TAB^T + CAB$$

$(AB)^T = B^TA^T$

Normal equations

$$X = \begin{bmatrix} 1 & (x_1)^T & - \\ 1 & (x_2)^T & - \\ \vdots & \vdots & - \\ 1 & (x_N)^T & - \end{bmatrix} \qquad X\theta = \begin{bmatrix} 1 & (x_1)^T & - \\ 1 & (x_2)^T & - \\ \vdots & \vdots & - \\ 1 & (x_N)^T & - \end{bmatrix} \theta$$

$$X \in N \times (n+1)$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \qquad X\theta = \begin{bmatrix} x_1^T \theta \\ \vdots \\ x_N^T \theta \end{bmatrix} = \begin{bmatrix} h_\theta(x_1) \\ \vdots \\ h_\theta(x_N) \end{bmatrix}$$

Normal equations

$$X\theta - y = \begin{bmatrix} h_\theta(x_1) - y_1 \\ \vdots \\ h_\theta(x_N) - y_N \end{bmatrix}$$

$$\frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$= \frac{1}{2}\sum_{i=1}^{N}(h_\theta(x_i) - y_i)^2$$

$$= J(\theta)$$

Normal equations

$$\nabla_\theta J(\theta) \overset{set}{=} 0$$

$$Z^T Z = \sum_i z_i^2 = tr(Z^T Z)$$

$$\nabla_\theta \frac{1}{2}(X\theta - y)^T (X\theta - y)$$

$$tr(ABC) = tr(CAB) = tr(BCA)$$

$$= \frac{1}{2}\nabla_\theta tr[(X\theta - y)^T (X\theta - y)]$$

$$-\nabla_\theta tr\left(\theta^T X_1^T y\right) = -\nabla_\theta tr\left(\theta^T X_1^T y\right)^T$$

$$= \frac{1}{2}\nabla_\theta tr[(\theta^T X^T - y^T)(X\theta - y)]$$

$$tr(A) = tr(A^T)$$

$$= \frac{1}{2}\nabla_\theta tr\left(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y\right)$$

$$= \frac{1}{2}\left[\nabla_\theta tr(\theta\theta^T X^T X) - \nabla_\theta tr(y^T X\theta) - \nabla_\theta tr(y^T X\theta)\right]$$

Normal equations

$\nabla_\theta tr(\theta\theta^T X^T X)$ $\qquad$ $\nabla_A tr\ (ABA^TC) = CAB+C^TAB^T$

$$\nabla_\theta tr\underbrace{\theta I}_{A\ B}\underbrace{\theta^T}_{A^T}\underbrace{X^T X}_{C} = \underbrace{X^T X}_{C}\underbrace{\theta I}_{A\ B} + \underbrace{X^T X}_{C^T}\underbrace{\theta I}_{A\ B^T}$$

$$= X^T X\theta + X^T X\theta$$

$$trAB = trBA \longrightarrow \nabla_\theta tr(y^T X\theta) = \nabla_\theta tr(\theta y^T X)$$

$$\nabla_A trAB = B^T \longrightarrow = \left(y^T X\right)^T = X^T y$$

Normal Equation

$$\nabla_\theta J(\theta) = \frac{1}{2}\left[X^T X\theta + X^T X\theta - X^T y - X^T y\right]$$

$$= X^T X\theta - X^T y \overset{set}{=} 0$$

$$X^T X\theta = X^T y$$

$$\theta = (X^T X)^{-1} X^T y$$

Normal Equation

```
% normal equation

clear all;
close all;
data = load ('data1.txt');
 X = data(:, 1);          %取X集合
 y = data(:, 2);                  %取y集合
 X = [ones(length(data),1),data(:,1)];
theta=zeros(2,1);
x_t=X';%矩阵转置
x1=x_t*X;
x2=inv(x1);
theta=x2*x_t*y
```

Linear Regression Extension

Example3: One-variable/multivariable linear regression

(注：该示例为血压与年龄、体重指数、吸烟习惯等的线性回归模型)

Number	Blood pressure (y)	Age (x^1)	body weight index (x^2)	Smoking habit (x^3)	**Number**	Blood pressure (y)	Age (x^1)	body weight index (x^2)	Smoking habit (x^3)
1	144	39	24.2	0	**16**	130	48	22.2	1
2	215	47	31.1	1	**17**	135	45	27.4	0
3	138	45	22.6	0	**18**	114	18	18.8	0
…	…	…	…	…	**…**	…	…	…	…
15	128	42	21.7	0	**30**	175	69	27.4	1

(1) One-variable linear regression:(注：该一元线性回归分析为血压与体重指数的模型，下面公式中的X^2指的是体重指数)

$$h_\theta(x) = \theta_0 + \theta_2 x^2$$

1) gradient descent:(梯度下降法实现一元线性回归及程序)

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

```
1     function J = computeCost(X, y, theta)
2
3 -       J = 0;
4 -       predictions = X * theta;
5 -       J = 1/(2*m)*(predictions - y)'*(predictions - y);
6 -   end
```

```
function [theta, J_history]= gradientDescent(X,y,theta,alpha, num_iters)
m = length(y);        %number of training examples
J_history = zeros(num_iters,1);
for iter = 1:num_iters
    temp(1) = theta(1) - (alpha / m )* sum((X * theta - y).*X(:,1));
    temp(2) = theta(2) - (alpha / m )* sum((X * theta - y).*X(:,2));
    theta(1)=temp(1);
    theta(2)=temp(2);
    J_history(iter)=computerCost(X, y, theta);

end
```

$$\theta_i := \theta_i - \alpha \sum_{j=1}^{m} \left(h_\theta \left(x_j \right) - y_j \right) \cdot x_j^{(i)}$$

```
clc
clear all
close all
data = load ('data1.txt');
 X = data(:, 1);
 y = data(:, 2);
 X = [ones(size(X,1),1),X];
 alpha=0.01;
 num_iters=400;
 theta=zeros(2,1);
 [theta, J_history]= gradientDescent(X,y,theta,alpha, num_iters);
 fprintf('Theta compute from gradient descent:\n');
 fprintf('%f\n',theta);
  fprintf('\n');
```

$$y = -3.895301 + 1.192985x^2$$

```
Theta compute from gradient descent:
-1.963957
0.998961
```

num_iters=400

```
Theta compute from gradient descent:
-3.895301
1.192985
```

num_iters=5000

2) normal equation:(注：梯度下降法实现一元线性回归及程序)

$$\theta = \left(X^T X\right)^{-1} X^T y$$

```
1    function [theta] = normalEqn(X,y)
2 -    theta = zeros(size(X,2),1);
3
4 -    theta = pinv((X'*X))*X'*y;
5 -  end
```

```
clc
clear all
close all
data = csvread('data1.txt');
X = data(:, 1);
y = data(:, 2);
m=length(y);
X=[ones(m,1) X];
[theta]=normalEqn(X,y);
fprintf('Iheta compute from the normal equations:\n');
 fprintf('%f\n',theta);
 fprintf('\n');
```

```
Iheta compute from the normal equations:
-3.895781
1.193034
```

$$y = -3.895781 + 1.193034x^2$$

regress

Multiple linear regression

(2)Multiple Linear Regression:(注：使用梯度法和正则方程法实现多元线性回归分析时，程序代码与一元线性回归相同，只需调主程序第五、六行为X = data(:,1:3); y = data(:,4)即可。)

$$y = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \theta_3 x^3$$

```
clc
clear all
close all
x1=[39 47 45 47 65 46 67 42 67 56 64 56 59 34 42 48 45 18 20 19 36 50 39 21 44 53 63 29 25 69];
x2=[24.2 31.1 22.6 24.0 25.9 25.1 29.5 19.7 27.2 19.3 28.0 25.8 27.3 20.1 21.7 22.2 27.4 18.8 22.6...
21.5 25.0 26.2 23.5 20.3 27.1 28.6  28.3  22.0  25.3  27.4];
x3=[0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1];
y=[144 215 138 145 162 142 170 124 158 154 162 150 140 110 128 130 135 114 116 124 136 142 120 120 160 15
X=[ones(30,1),x1',x2',x3'];
[b,bint,r,rint,stats]=regress(y',X);
b,bint,stats
```

```
b =

   45.3636
    0.3604
    3.0906
   11.8246


bint =

    3.5537   87.1736
   -0.0758    0.7965
    1.0530    5.1281
   -0.1482   23.7973
```

regression coefficient	Estimates of regression coefficient	confidence interval of regression coefficient
θ_0	45.3636	[3.5537 87.1736]
θ_1	0.3604	[-0.0758 0.7965]
θ_2	3.0906	[1.0530 5.1281]
θ_3	11.8246	[-0.1482 23.7973]

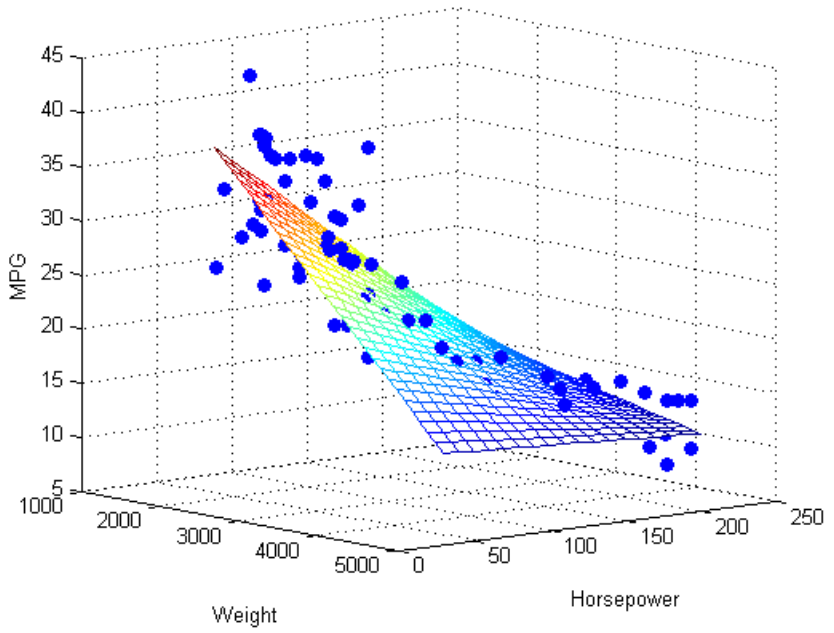
$$y = 45.3636 + 0.3604x^1 + 3.0906x^2 + 11.8246x^3$$

Example4: Load the sample data. Identify weight and horsepower as predictors, and mileage as the response.

```
load carsmall
x1 = Weight;
x2 = Horsepower;     % Contains NaN data
y = MPG;
```

```
X = [ones(size(x1)) x1 x2 x1.*x2];
b = regress(y,X)     % Removes NaN data
```

```
scatter3(x1,x2,y,'filled')
hold on
x1fit = min(x1):100:max(x1);
x2fit = min(x2):10:max(x2);
[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);
YFIT = b(1) + b(2)*X1FIT + b(3)*X2FIT + b(4)*X1FIT.*X2FIT;
mesh(X1FIT,X2FIT,YFIT)
xlabel('Weight')
ylabel('Horsepower')
zlabel('MPG')
view(50,10)
```



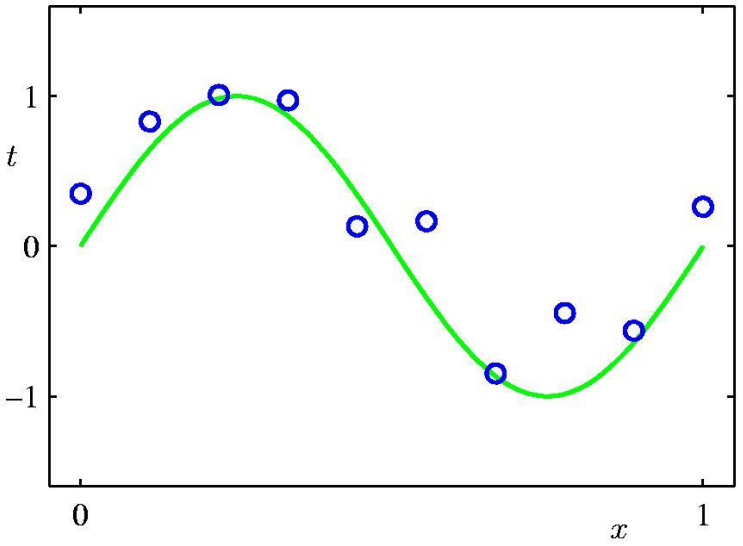
Improved gradient descend:

$$
\begin{cases}
\theta^{(0)} \\
\theta^{(m+1)} = \theta^{(m)} - \eta^{(m)} \left(\dfrac{\partial J^{(m)}}{\partial \theta_i^m} \right) + \alpha^{(m)} \left(\theta^{(m)} - \theta^{(m-1)} \right) \\
\eta^{(m+1)} = \begin{cases} \eta^{(m)} \phi & J^{(m+1)} - J^{(m)} \le 0 \\ \eta^{(m)} \beta & J^{(m+1)} - J^{(m)} > 0 \end{cases} \\
\alpha^{(m+1)} = \begin{cases} \alpha_0 & J^{(m+1)} - J^{(m)} \le 0 \\ 0 & J^{(m+1)} - J^{(m)} > 0 \end{cases} \\
i, m = 0, 1, \ldots.
\end{cases}
$$

In this equation, the parameter η and α are the step length and the inertia coefficient of the iterative process respectively, which are changing with the changing number of iteration.

Besides, the parameter Φ and β are the weight coefficient for η, used in the iterative process. And $\Phi > 1$, $0 < \beta < 1$.

Overfitting underfitting



- **Training set: x, t**
- **Goal: exploit this training set in order to make predictions of the value $\hat{t}$ of the target variable for some new value $\hat{x}$**
- **Method: find a function $t = f(x)$**

fit the data using a polynomial function

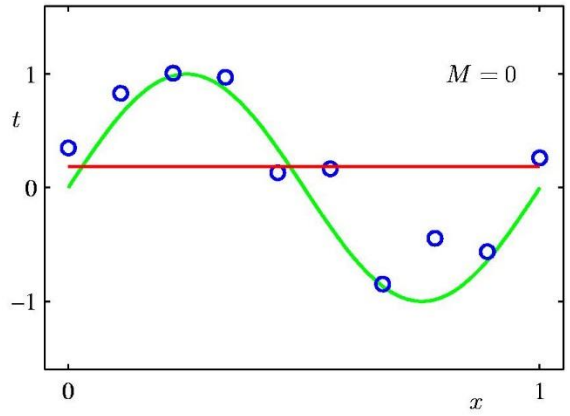
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Linear model: although the polynomial function y(x,w) is a nonlinear function of x, it is a linear function of the coefficients w.

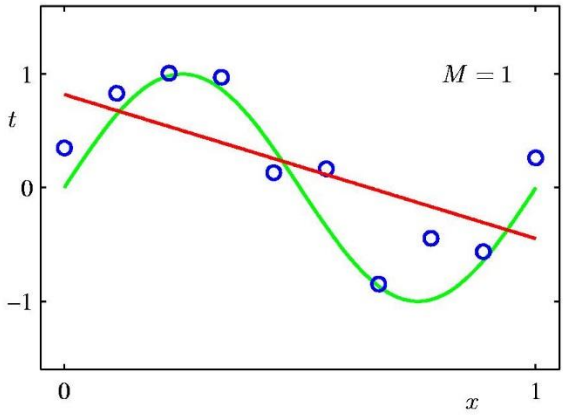
Overfitting underfitting

- Problem: choosing the order M of the polynomial

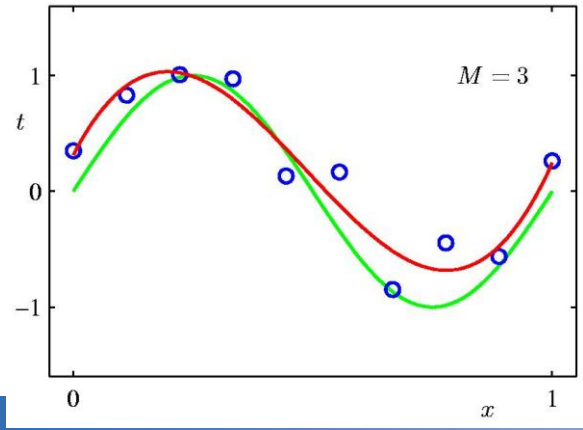
0th Order Polynomial



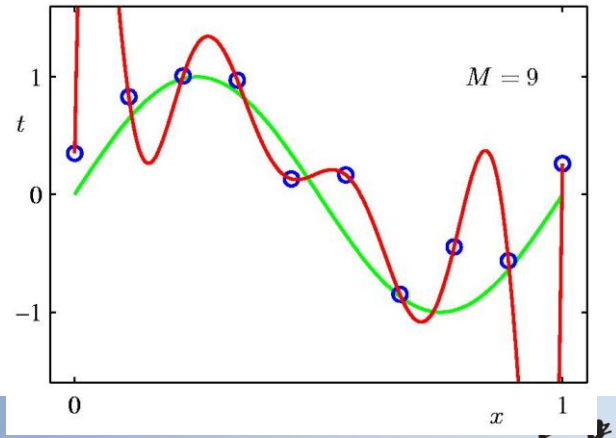
1st Order Polynomial



3rd Order Polynomial

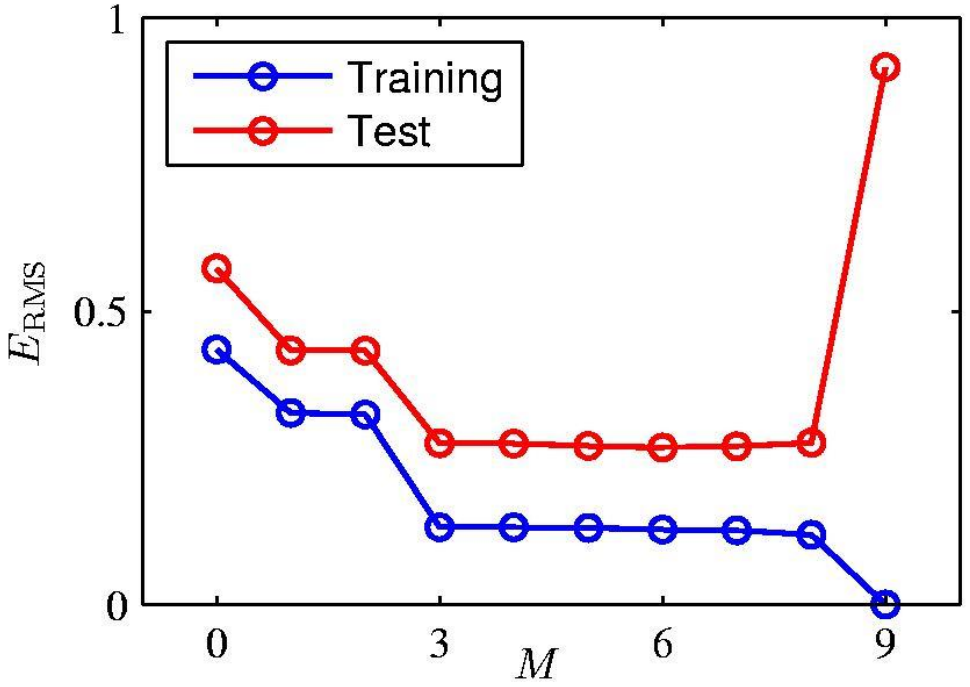


9th Order Polynomial



Over-fitting

Goal: is to achieve good generalization by making accurate predictions for new data.



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

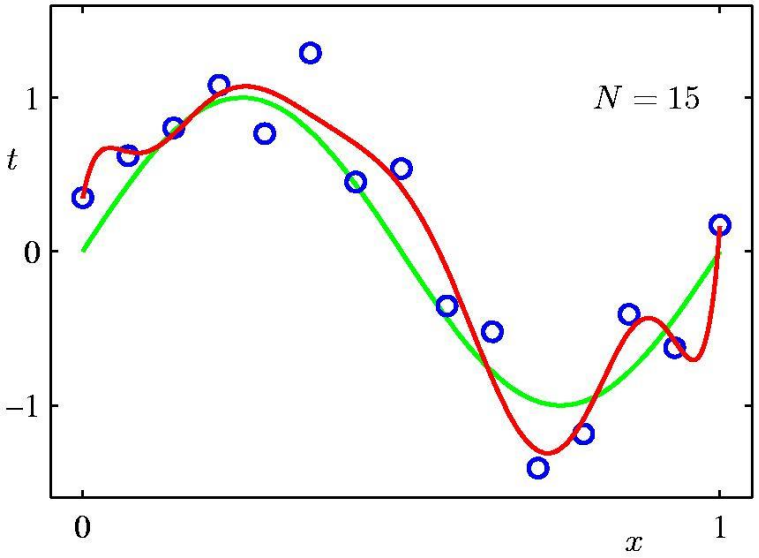
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Data Set Size:

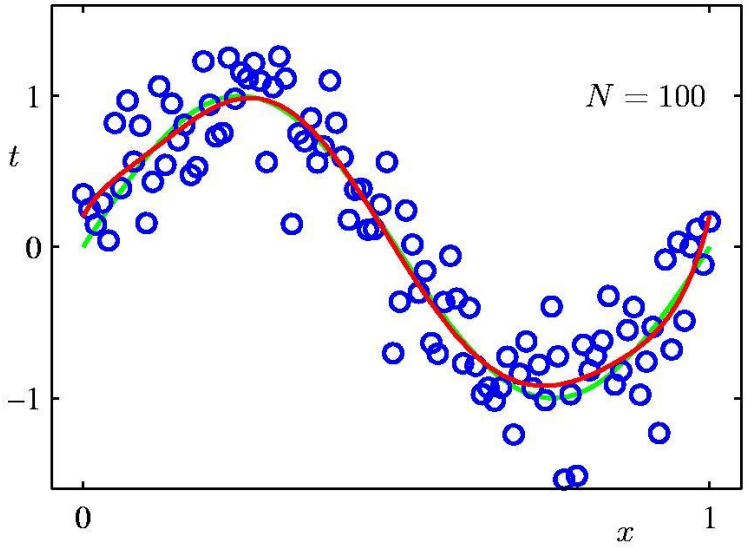
$N = 15$

9th Order Polynomial



$N = 100$

9th Order Polynomial



the larger the data set, the more complex (in other words more flexible) the model that we can afford to fit to the data.

3 solutions to over-fitting problem

- Rough heuristic: data-set-size >= some multiple (say 5 or 10) of the number of adaptive parameters in the model.
- Regularization: adding a penalty term to the error function in order to discourage the coefficients from reaching large values.
- Local weighted regression(LWR)

Regularization

Penalize large coefficient values

The simplest such penalty term: sum of squares of all of the coefficients

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where $\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} = w_1^2 + \cdots + w_M^2$ (w_0 is omitted, or with its own regularization coefficient)

coefficient *λ*: governs the relative importance of the regularization term compared with the sum-of-squares error term.

Gradient Descent with Regulirization

- Repeat {

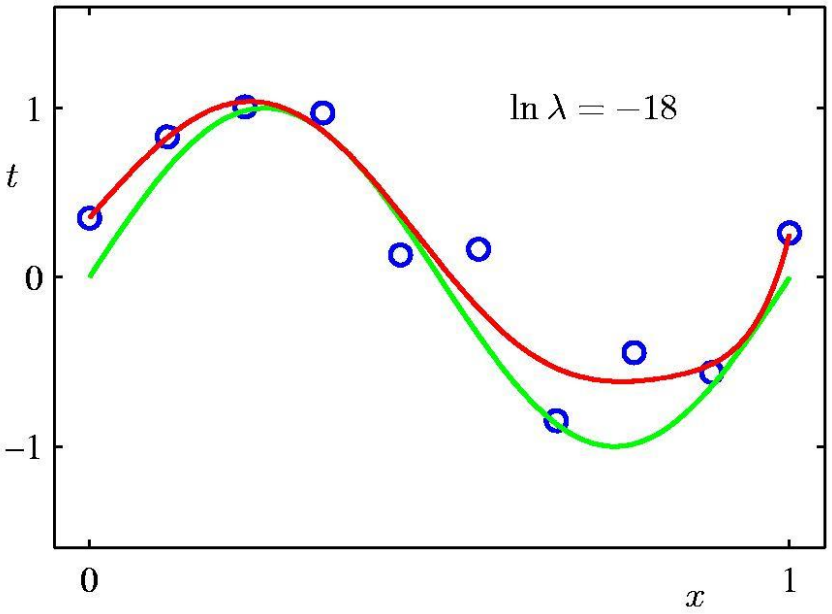
$$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_i^0$$

$$\theta_j := \theta_j - \alpha \left[\sum_{i=1}^{N} (h_\theta(x_i) - y_i) x_i^j + \lambda \theta_j \right]$$

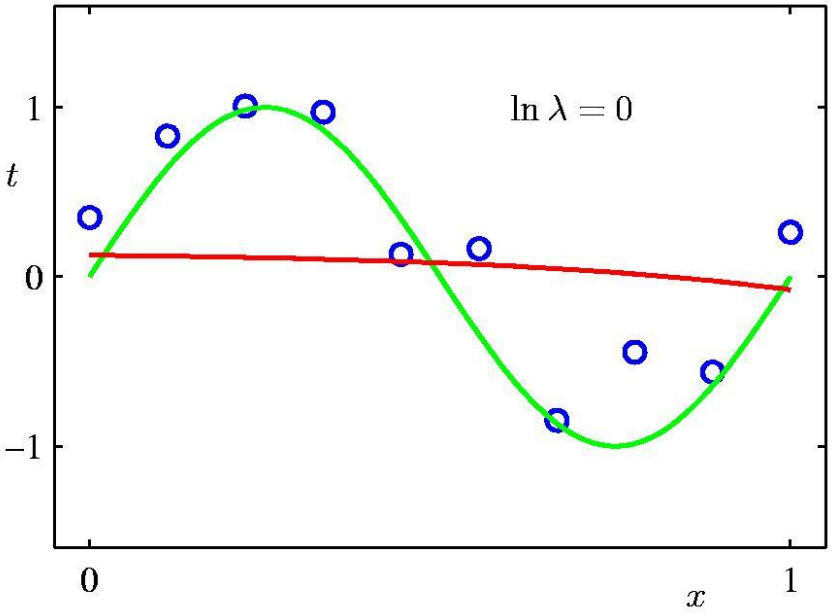
- }

Regularization:

$\ln \lambda = -18$



$\ln \lambda = 0$

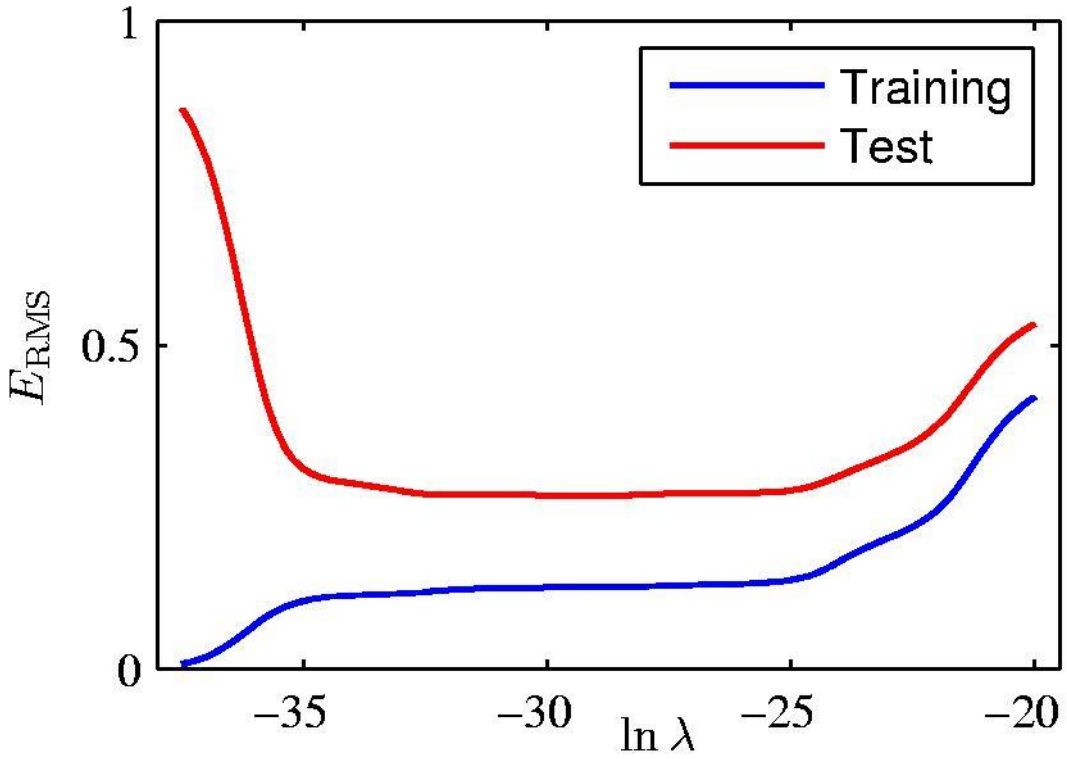


Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

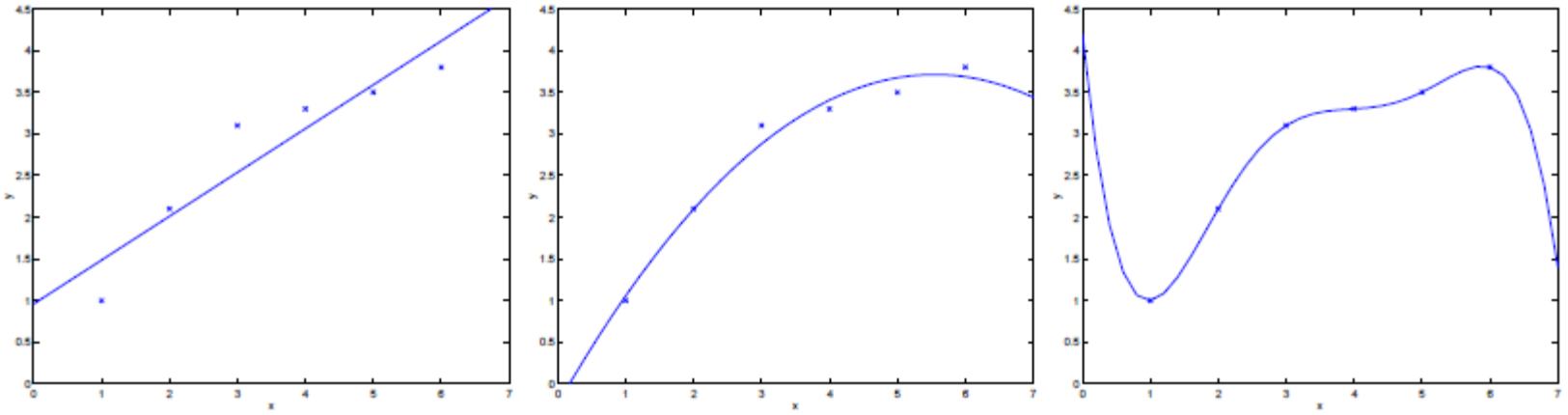
Regularization

E_{RMS}



$\ln \lambda$

Locally weighted linear regression



- Left: linear, underfitting
- Middle: quadra $y = \theta_0 + \theta_1 x + \theta_2 x^2,$
- Right: fitting a 5-th order polynomial $y = \sum_{j=0}^{5} \theta_j x^j.$
  - overfitting
- Naively, it might seem that the more features we add, the better. However, there is also a danger in adding too many features
- LWR: makes the choice of features less critical.

In the original linear regression algorithm, to make a prediction at a query point x (i.e., to evaluate $h(x)$), we would:

1. Fit θ to minimize $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$.

2. Output $\theta^T x$.

In contrast, the locally weighted linear regression algorithm does the following:

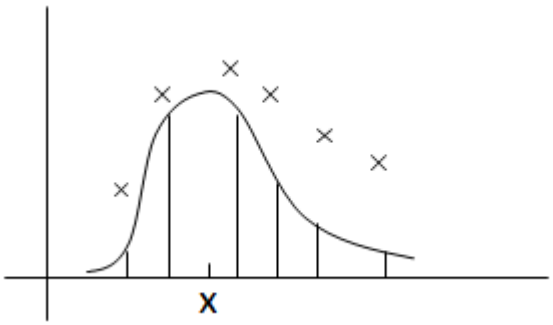
1. Fit θ to minimize $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$.

2. Output $\theta^T x$.

$w^{(i)}$'s Non-negative valued weights

- If small, the error term will be pretty much ignored in the fit
- Standard choice

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$



- Depends on the particular point x at which we are trying to evaluate.
- If $\left|x^{(i)} - x\right|$ is small, $w^{(i)}$ is close to 1
- If $\left|x^{(i)} - x\right|$ is large, $w^{(i)}$ is small
- θ is chosen giving a much higher "weight" to the (errors on) training examples close to the query point x.
- τ : bandwidth parameter, controls how quickly the weight of a training example falls off with distance of its $x^{(i)}$ from the query point x

- Locally weighted linear regression: non-parameteric algorithm
  - Keep the entire training set around
- (unweighted) linear regression : parametric learning algorithm
  - Has a fixed, finite number of parameters, which fit to the data
  - Once fit the θ, stored them away, no longer need to keep the training data around to make future predictions

Logistic Regression

Classification

- • The goal in classification is to take an input vector **x** and to assign it to one of *K* discrete classes C_k where $k = 1, \ldots, K$.
- • The input space is divided into K decision regions whose boundaries are called decision boundaries or decision surfaces.
- Each region represents one class and each input is classified into one of regions (classes).

Linear Classification

- • Linear classification means that the decision surfaces are linear functions of the input vector x and hence are defined by (D − 1)- dimensional hyperplanes within the D-dimensional input space.
- • Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable.

Linear Regression vs.Linear Classification

- Linear regression

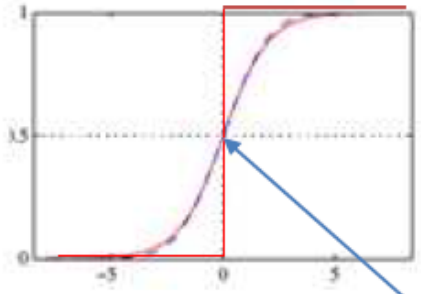
$$y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x)$$

- Linear classification
  - Regression $y(x, w) = \sum_{j=0}^{M-1} w_j x_j$, $\phi_j(x)=x_j$, where x_j is the jth element of x
  - Classification c=f(y(x)), where f is a function (activation function) that maps x into different classes C.

Activation Functions

- The simplest activity function is just y itself, i.e., f(y)=y.
- The most commonly used activity function is the sigmoid (S-shape) function

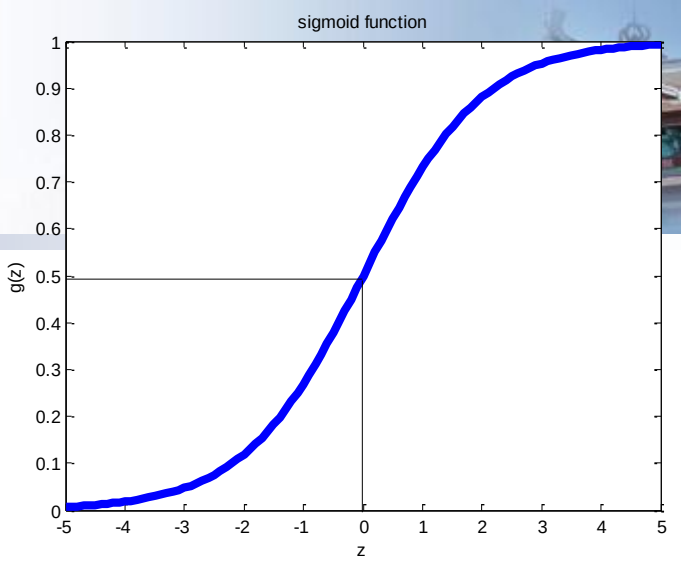
$$f(y) = \frac{1}{1+e^{-y}}$$



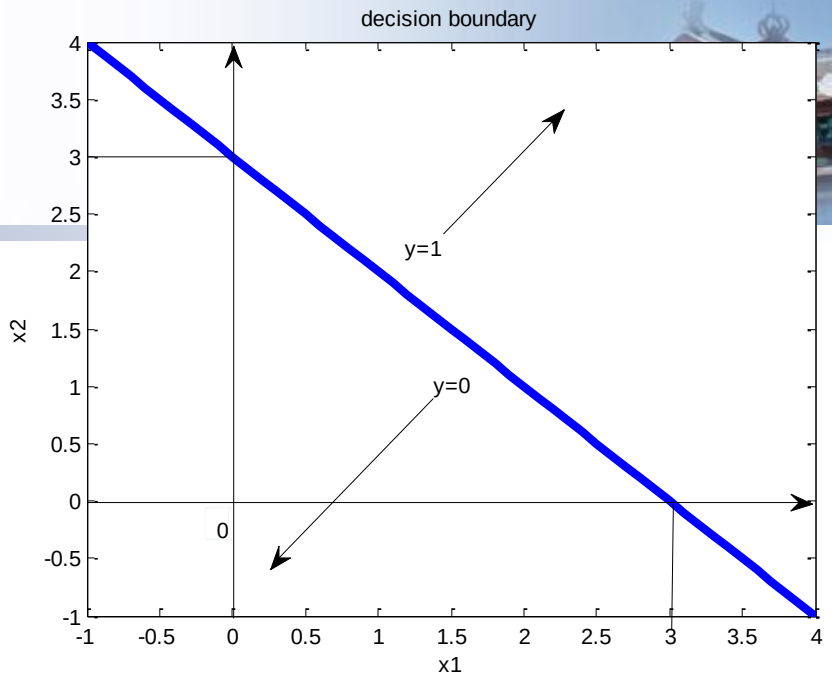
which maps (squashes) the real value y into a finite intervals (classes).

Step threshold function

- Classification with sigmoid activation function is often called logistic regression.



- Sigmoid function:
  - Y > 0, f(y) > 0.5; y->∞, f(y)->1
  - y < 0, f(y) < 0.5; y->- ∞, f(y)->0
- $y = w^T x$, $f(y) = f(w^T x) = P(y=1|x; w)$
  - $w^T x > 0$, $P(y=1|x; w) > 0.5$, predict "y=1"
  - $w^T x < 0$, $P(y=1|x; w) < 0.5$, predict "y=0"
  - $w^T x = 0$, decision boundary



decision boundary:

In logistic regression: input = 0; $w^T x = 0$

Linear decision boundary: $w^T x = \sum w_i x_i$

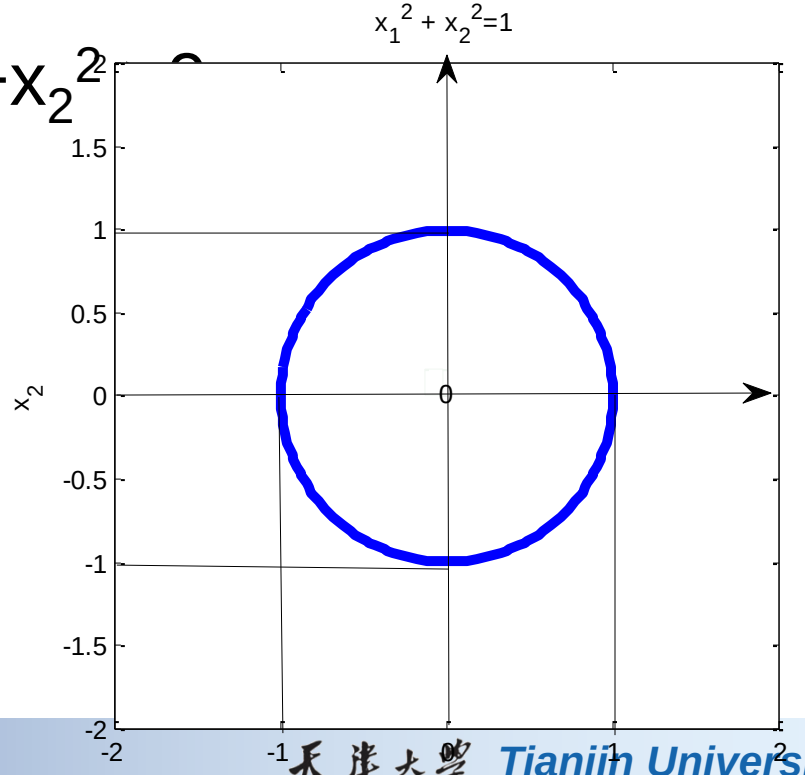
non-linear decision boundary

Exampel1: linear decision boundary

X = [1 x_1 x_2], w=[-3 1 1]^T ; -3 + x_1 + x_2 =0

Predict "y=1" if -3 + x_1 + x_2 >0

- Example 2: nonlinear decision boundary
  - $X = [\ 1\ x_1\ x_2\ x_1^2\ x_2^2]$, $w = [\ -1\ 0\ 0\ 1\ 1]$, $-1 + x_1^2 + x_2^2 = 0$
  - Predict "y=1" if $-1 + x_1^2 + x_2^2$



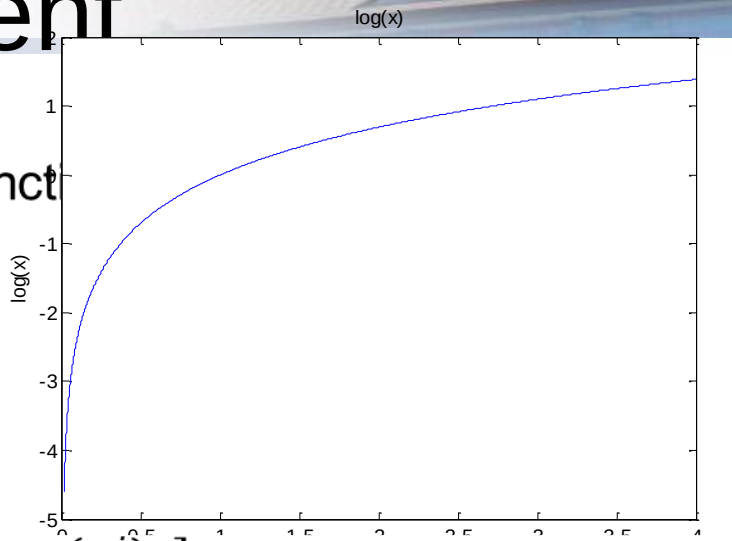
Update rule 1: cost function & gradient descent

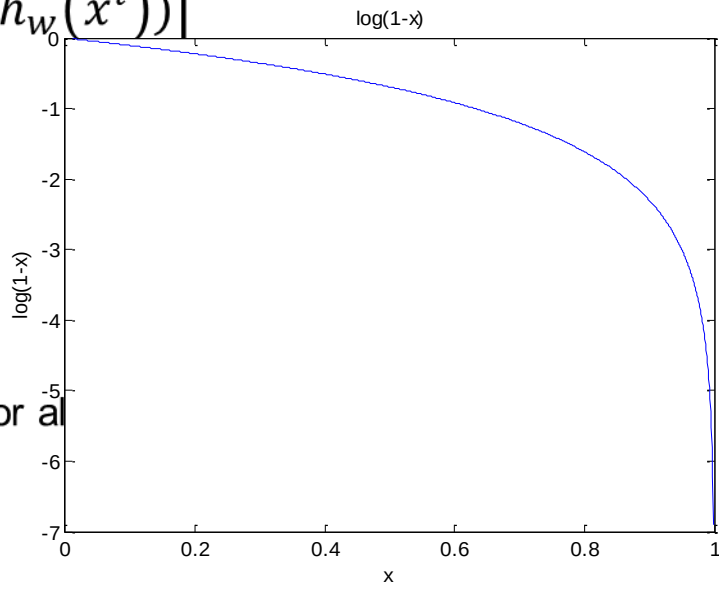
- Cost function $J(w) = \frac{1}{N}\sum_{i=1}^{N} Cost(h_w(x^i), y^i)$
- $Cost(h_w(x^i), y^i)$: can't use LMS(not a convex funct minima)
- $$Cost(h_w(x^i), y^i) = \begin{cases} -\log(h_w(x^i)) & y^i = 1 \\ -\log(1 - h_w(x^i)) & y^i = 0 \end{cases}$$
- Simplified cost function:
- $J(w) = \frac{1}{N}\sum_{i=1}^{N}\left[-y^i \log(h_w(x^i)) - (1 - y^i)\log(1 - h_w(x^i))\right]$
- Gradient descent
- $w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$
- $\frac{\partial}{\partial w_j} J(w) = \frac{1}{N}\sum_{i=1}^{N}(h_w(x^i) - y^i)x_j$

– Repeat until convergence{

– $w_j := w_j - \alpha \frac{1}{N}\sum_{i=1}^{N}(h_w(x^i) - y^i)x_j$(simultaneously update for al

– }





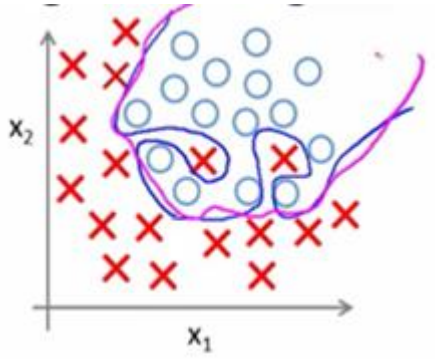
- Feature scaling also applys to gradient descent in logistic regression.
- Differences with implementing linear regression:
- $h_w(\mathrm{x}^i) = \frac{1}{1+e^{-w^T \mathrm{x}^i}}$

Update rule 2: log-likelihood gradient ascent

- Assume that: $P(y=1|x; w)=h_w(\mathbf{x}^i)$
- $P(y=0|x; w)=1-h_w(\mathbf{x}^i)$
- can be written more compactly as
- $P(y|x; w)=h_w(\mathbf{x}^i)^{y^i}(1-h_w(\mathbf{x}^i))^{1-y^i}$
- likelihood function of the parameters
- $L_w=\prod_{i=1}^{N} p(y^i|\mathbf{x}^i,w)=\prod_{i=1}^{N} h_w(\mathbf{x}^i)^{y^i}(1-h_w(\mathbf{x}^i))^{1-y^i}$
- maximize the log likelihood:
- $l_w=\log L_w=\sum_{i=1}^{N}\left[y^i\log\left(h_w(\mathbf{x}^i)\right)+(1-y^i)\log(1-h_w(\mathbf{x}^i))\right]$
- Update rule: $w_j := w_j+\alpha\frac{\partial}{\partial w_j}l_w$ $\frac{\partial}{\partial w_j}l_w=\sum_{i=1}^{N}(y^i-h_w(\mathbf{x}^i))x_j$

Regularized logistic regression

- $h_w(\mathbf{x}^i) = \frac{1}{1+e^{-w^T\phi}}$
- Cost function:



$$J(w) = \frac{1}{N}\sum_{i=1}^{N}\left[-y^i\log(h_w(\mathbf{x}^i)) - (1-y^i)\log(1-h_w(\mathbf{x}^i))\right] + \frac{\lambda}{2M}\sum_{j=1}^{M} w_j^2$$

- Gradient descent:

Repeat until convergence{

$$w_0 := w_0 - \alpha\frac{1}{N}\sum_{i=1}^{N}(h_w(\mathbf{x}^i) - y^i)x_0$$

$$w_j := w_j - \alpha\frac{1}{N}\sum_{i=1}^{N}(h_w(\mathbf{x}^i) - y^i)x_j + \frac{\lambda}{M}w_j$$

}

Multiple Class Discriminant

- A single *K*-class discriminant consists of *K* linear functions of the form

$$y_k(x) = w_k^T x + w_{k0}$$

for k=1,2, …,K

- assigning **x** to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$.

$$k^* = \arg\max_k \; y_k(x)$$

Discriminant Parameter Learning

- Learn the weights **w** , given input a set of input **x** and output **t**.
- Learning methods
  - Least-squares method
  - Fisher discriminant learning

Least-squares method (ML method)

- Estimate w such that the model predictions as close as possible to a set of target values

- Each class C_k is described by its own linear model so that

$$y_k(x) = w_k^T x + w_{k0} = [w_{k0}\ w_k^T] \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$k = 1,2,\ldots,K$$

Least-squares method

- • In vector format, we have

$$Y^{Kx\,1}(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ . \\ . \\ y_K(x) \end{bmatrix}$$

$$W^{(D+1)xK} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{K0} \\ w_1 & w_2 & \cdots & w_K \end{bmatrix}$$

$$Y(x) = W^T \begin{bmatrix} 1 \\ x \end{bmatrix} = W^T x'$$

Least-squares method (cont'd)

- Given N samples $x_1, x_2, \ldots, x_N$ and their corresponding class labels $t_1, t_2, \ldots, t_N$,where t_n is a Kx1 vector that follows 1-of-K coding, i.e., the elements t_n are all zeros, except for the class that x_n belongs to. Least-squares solution is to find W that minimizes the sum-of-squares errors E, i.e.,

$$E = \sum_{n=1}^{N} [t_n - Y(x_n)]^T [t_n - Y(x_n)]$$

$$= \sum_{n=1}^{N} [t_n - W^T x_n^{'}]^T [t_n - W^T x_n^{'}]$$

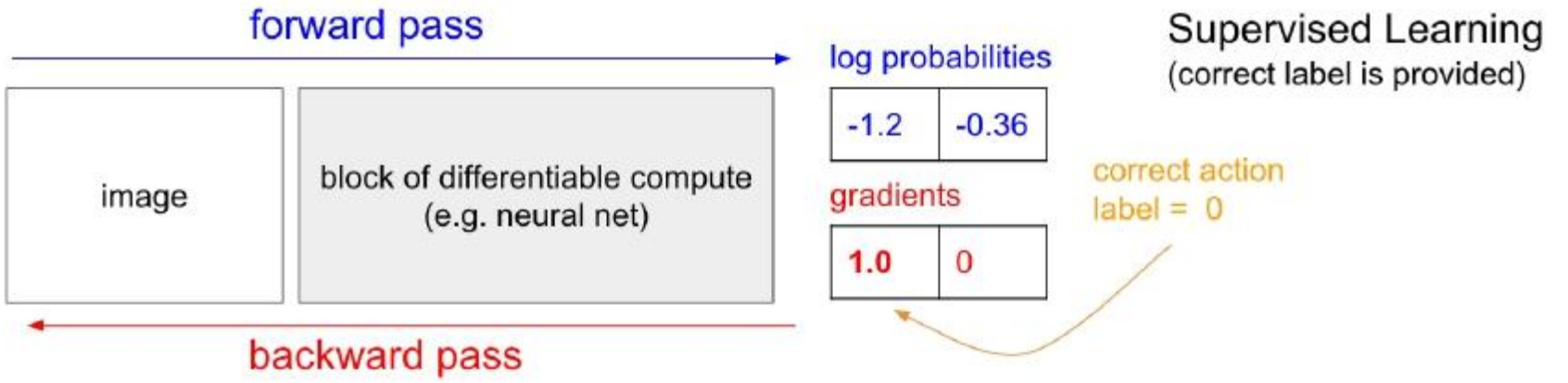
1-of-K Coding

- discrete variables that can take on one of K possible mutually exclusive states.
- convenient representation: 1-of-K scheme
  - variable is represented by a K-dimensional vector **y** in which one of the elements y_k equals 1, and all remaining elements equal 0.
  - E.g.

g	y_1	y_2	y_3	y_4
3	0	0	1	0
1	1	0	0	0
2	0	1	0	0
4	0	0	0	1
1	1	0	0	0

$$y^1 = [0, 0, 1, 0]^T$$

Task: Classify an Image of a Number

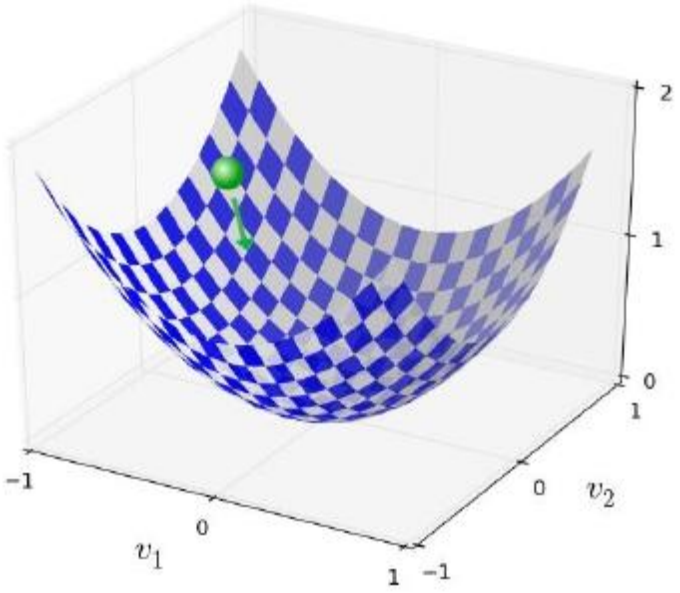


Ground truth for "6":

$$y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0)^T$$

"Loss" function:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$



Least-squares method (cont'd)

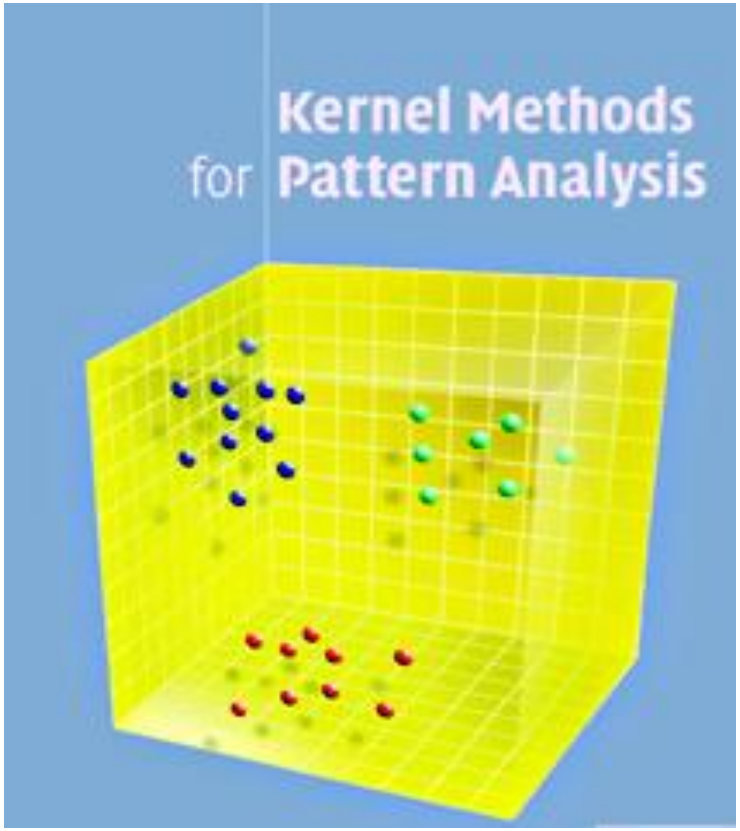
Take the derivatives of E w.r.t to W and setting it to zero, we can solve W as follows

$$W = (X^T X)^{-1} X^T T$$

where

$$X^{Nx(D+1)} = \begin{bmatrix} x'^T_1 \\ x'^T_2 \\ . \\ . \\ x'^T_N \end{bmatrix} \qquad T^{Nxk} = \begin{bmatrix} t^T_1 \\ t^T_2 \\ . \\ . \\ t^T_N \end{bmatrix}$$

Kernel Method

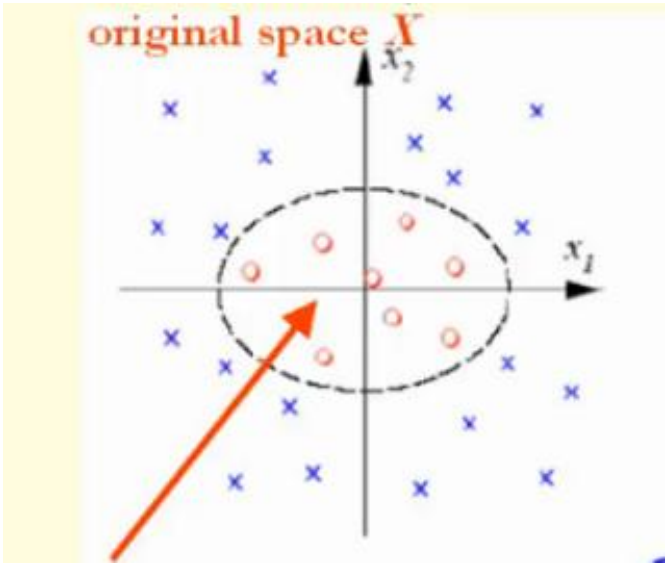

Kernel Methods
for Pattern Analysis

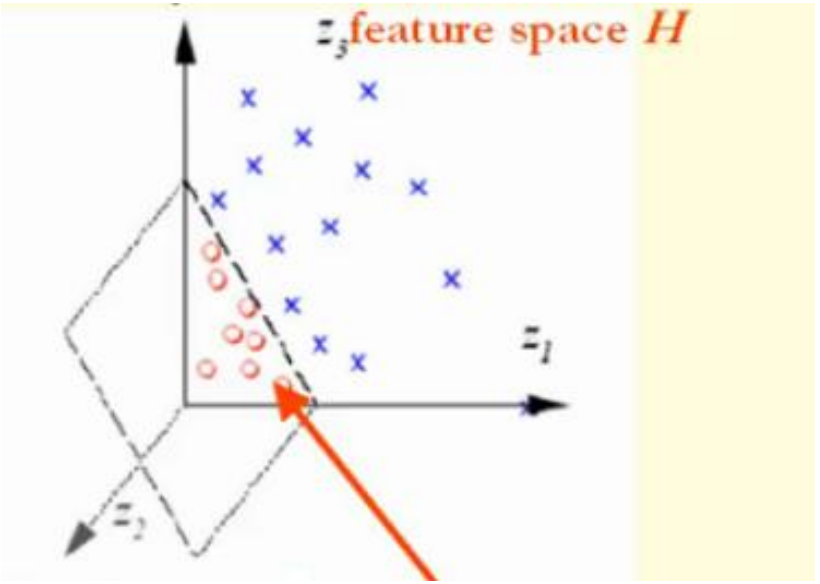
The Basic Idea

feature mapping

$$\phi: R^2 \to R^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$





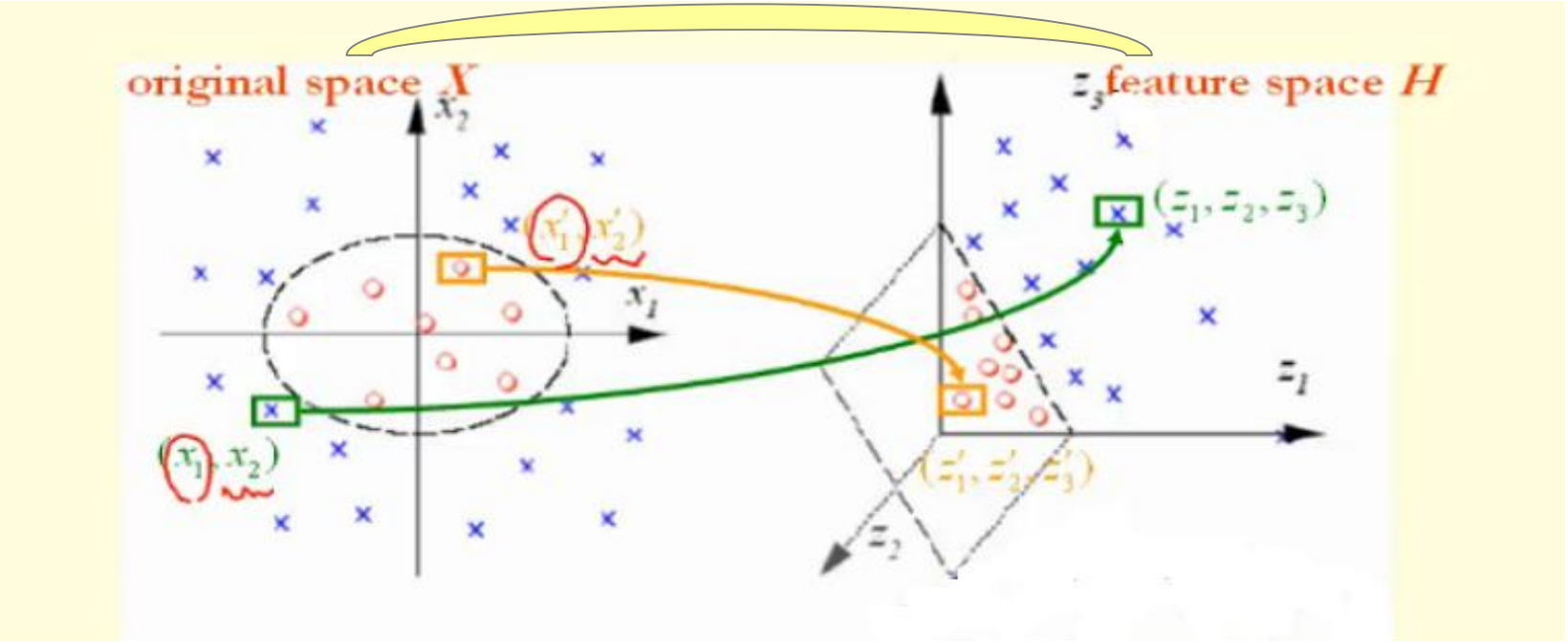
$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1$$

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \Rightarrow \frac{1}{a^2} z_1 + 0 \cdot z_2 + \frac{1}{a^2} z_3 = 1$$

The Basic Idea

feature mapping $\phi: R^2 \to R^3$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^{\,2}, \sqrt{2}x_1x_2, x_2^{\,2}\right)$$



$$<\phi(x_1, x_2), \phi(x_1', x_2')> = <(z_1, z_2, z_3), (z_1', z_2', z_3')> = <\left(x_1^{\,2}, \sqrt{2}x_1x_2, x_2^{\,2}\right), \left(x_1'^{\,2}, \sqrt{2}x_1'x_2', x_2'^{\,2}\right)>$$

$$= x_1^{\,2}x_1'^{\,2} + 2x_1x_2x_1'x_2' + x_2^{\,2}x_2'^{\,2} = \left(x_1x_1' + x_2x_2'\right)^2 = \left(<x, x'>\right)^2 = K(x, x')$$

Kernel function

The Basic Idea

Kernel Methods work by

- **embedding data in a vector space(with more high dimensionality)**
- **looking for(linear) relations in such space**

If map is chosen suitably, complex relations can be simplified, and easilly detected.

The Basic Idea

Distance in the Feature Space

$$\left\|\phi(x)-\phi(x')\right\|^2=\left(\phi(x)-\phi(x')\right)^T\left(\phi(x)-\phi(x')\right)$$

$$=\phi(x)^T\phi(x)-2\phi(x)^T\phi(x')+\phi(x')^T\phi(x')$$

$$=<\phi(x),\phi(x)>-2<\phi(x),\phi(x')>+<\phi(x'),\phi(x')>$$

$$=K(x,x)-2K(x,x')+K(x',x')$$

The Basic Idea

Angle in the Feature Space

$$<\phi(x),\phi(x')>=\|\phi(x)\|\|\phi(x')\|\cos\theta$$

$$\Rightarrow \cos\theta=\frac{<\phi(x),\phi(x')>}{\|\phi(x)\|\|\phi(x')\|}=\frac{<\phi(x),\phi(x')>}{\sqrt{<\phi(x),\phi(x)>}\sqrt{<\phi(x'),\phi(x')>}}$$

$$=\frac{K(x,x')}{\sqrt{K(x,x)}\sqrt{K(x',x')}}$$

The Basic Idea

Recall that

$$<\phi(x_1,x_2),\phi(x_1',x_2')>=<(z_1,z_2,z_3),(z_1',z_2',z_3')>=<\left(x_1^{\,2},\sqrt{2}x_1x_2,x_2^{\,2}\right),\left(x_1'^{\,2},\sqrt{2}x_1'x_2',x_2'^{\,2}\right)>$$

$$=x_1^{\,2}x_1'^{\,2}+2x_1x_2x_1'x_2'+x_2^{\,2}x_2'^{\,2}=\left(x_1x_1'+x_2x_2'\right)^2=\left(<x,x'>\right)^2:=K(x,x')$$

Distance in the Feature Space

$$\left\|\phi(x)-\phi(x')\right\|^2=\left(\phi(x)-\phi(x')\right)^T\left(\phi(x)-\phi(x')\right)$$

$$=\phi(x)^T\phi(x)-2\phi(x)^T\phi(x')+\phi(x')^T\phi(x')$$

$$=<\phi(x),\phi(x)>-2<\phi(x),\phi(x')>+<\phi(x'),\phi(x')>$$

$$=K(x,x)-2K(x,x')+K(x',x')$$

Angle in the Feature Space

$$<\phi(x),\phi(x')>=\left\|\phi(x)\right\|\left\|\phi(x')\right\|\cos\theta$$

$$\Rightarrow\cos\theta=\frac{<\phi(x),\phi(x')>}{\left\|\phi(x)\right\|\left\|\phi(x')\right\|}=\frac{<\phi(x),\phi(x')>}{\sqrt{<\phi(x),\phi(x)>}\sqrt{<\phi(x'),\phi(x')>}}=\frac{K(x,x')}{\sqrt{K(x,x)}\sqrt{K(x',x')}}$$

The Basic Idea

Inner Product Matrix/Gram Matrix/Kernel Matrix

$$K = \begin{bmatrix} <\phi(x_1),\phi(x_1)> & \cdots & <\phi(x_1),\phi(x_N)> \\ \vdots & \ddots & \vdots \\ <\phi(x_N),\phi(x_1)> & \cdots & <\phi(x_1),\phi(x_N)> \end{bmatrix}$$

$$= \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1 \cdots x_N) \\ \vdots & \ddots & \vdots \\ K(x_N,x_1) & \cdots & K(x_N,x_N) \end{bmatrix}$$

Kernel Function and Feature Mapping

One definition:

Finitely positive semi-definite functions

One theorem:

Characterization of kernels

Finitely Positive Semi-definite Functions

A function

$$K : X \times X \to R$$

satisfies the finitely positive semi-definite property if it is a symmetric function for which the matrices formed by restriction to any finite subset of the space are positive semi-definite.

Given any

$$\{x_1, \cdots, x_n\} \longrightarrow K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

any y

$$y^T K y \geq 0$$

Finitely Positive Semi-definite Functions

Example: Show that

$$K(x,z)=\langle x,z\rangle$$

is a finitely positive semi-definite function.

Given any $\{x_1,\cdots,x_n\}$

$$\Rightarrow K=\begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix}=\begin{bmatrix} <x_1,x_1> & \cdots & <x_1,x_n> \\ \vdots & & \vdots \\ <x_n,x_1> & \cdots & <x_n,x_n> \end{bmatrix}$$

$$=\begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_n \\ \vdots & & \vdots \\ x_n^T x_1 & \cdots & x_n^T x_n \end{bmatrix}=\begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$$

$$(AB)^T=B^T A^T$$

$$\Rightarrow y^T K y=y^T\begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}y=\left(\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}y\right)^T\left(\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}y\right)$$

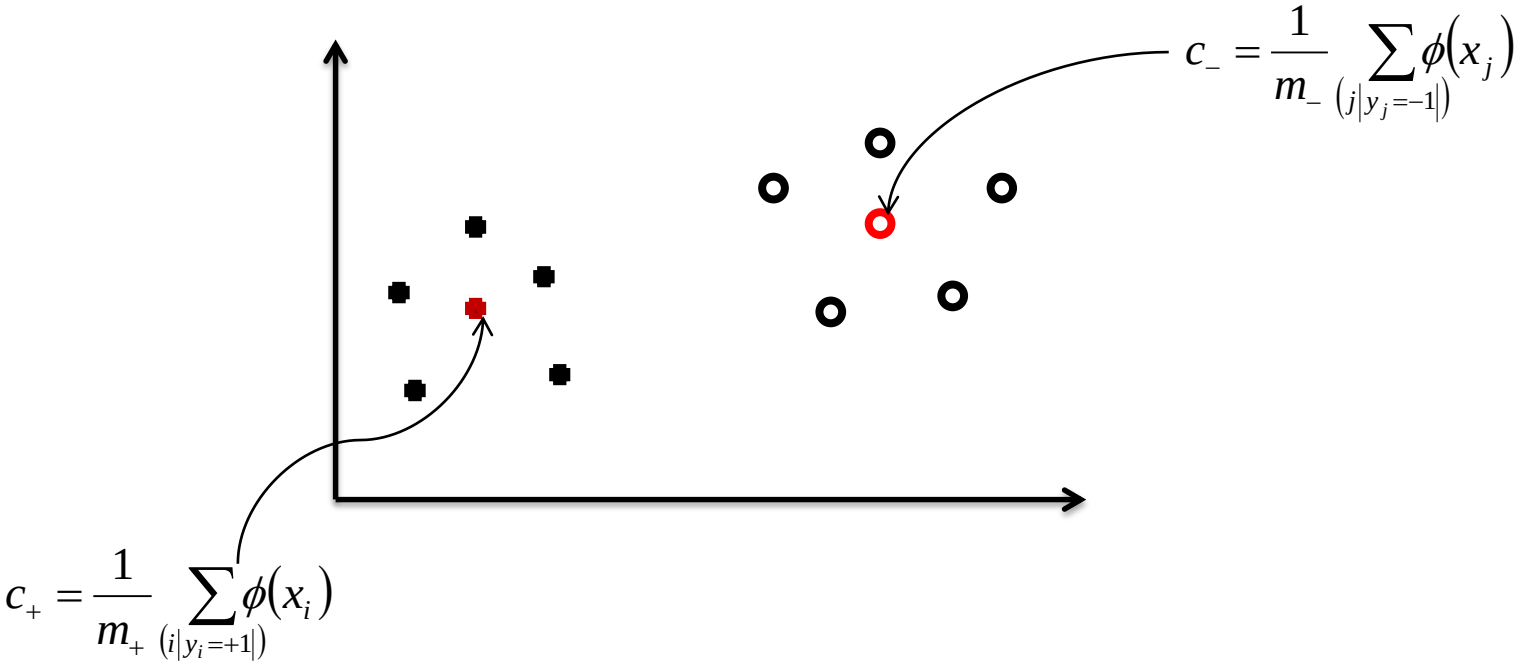
$$=\left\|\begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}y\right\|^2\geq 0$$

A Simple Recognition Algorithm

Training Samples:

feature mapping

$$\{(x_1, y_1), \cdots (x_n, y_n)\} \subset R^d \times \{1, -1\} \xrightarrow{\phi} \{(\phi(x_1), y_1), \cdots, (\phi(x_n), y_n)\} \subset H \times \{1, -1\}$$



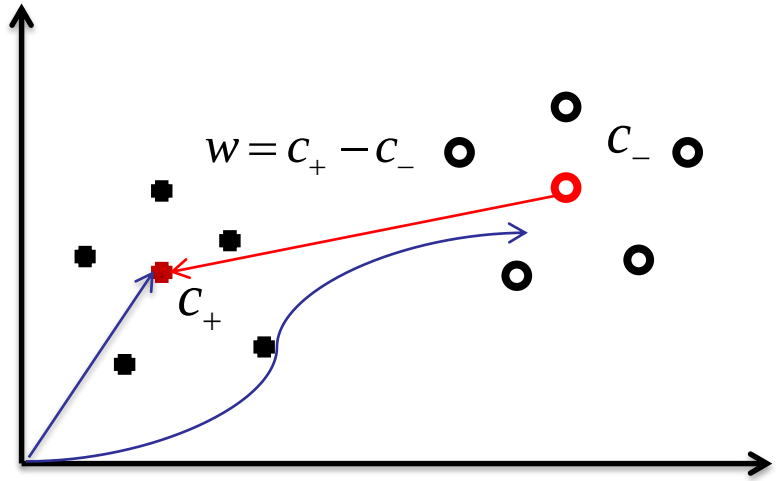
A Simple Recognition Algorithm

Training Samples:

feature mapping

$$\phi$$

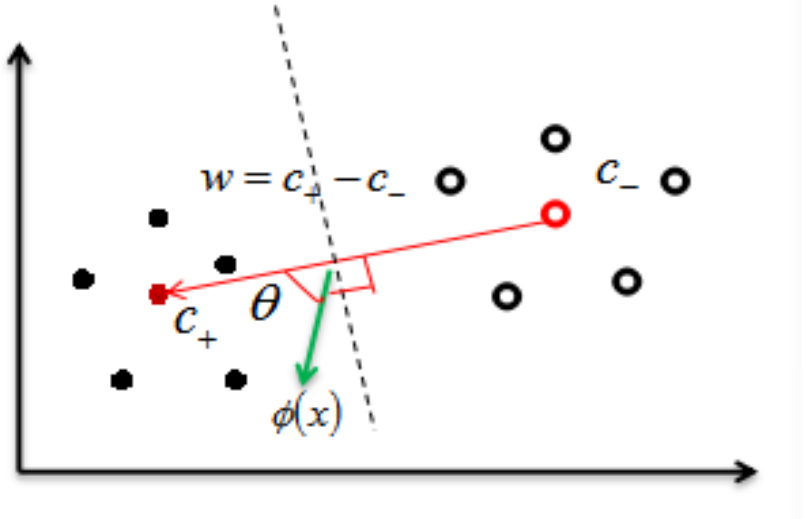
$$\{(x_1, y_1), \cdots (x_n, y_n)\} \subset R^d \times \{1, -1\} \to \{(\phi(x_1), y_1), \cdots, (\phi(x_n), y_n)\} \subset H \times \{1, -1\}$$



A Simple Recognition Algorithm

Testing Sample:

$$(x, y) \xrightarrow{\phi} (\phi(x), y) \rightarrow y = ?$$

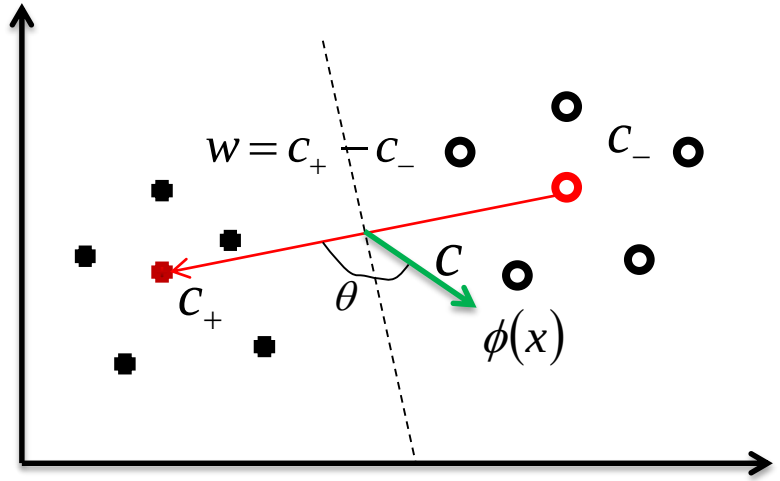


$$y = 1 \Leftrightarrow 0 \le \theta < \frac{\pi}{2} \Leftrightarrow 0 < \cos\theta \le 1 \Leftrightarrow 0 < \frac{< \phi(x) - c, w >}{\sqrt{\|\phi(x) - c\|}\sqrt{\|w\|}} \le 1 \Leftrightarrow < \phi(x) - c, w >> 0$$

A Simple Recognition Algorithm

Testing Sample:

$$(x, y) \xrightarrow{\phi} (\phi(x), y) \rightarrow y = ?$$

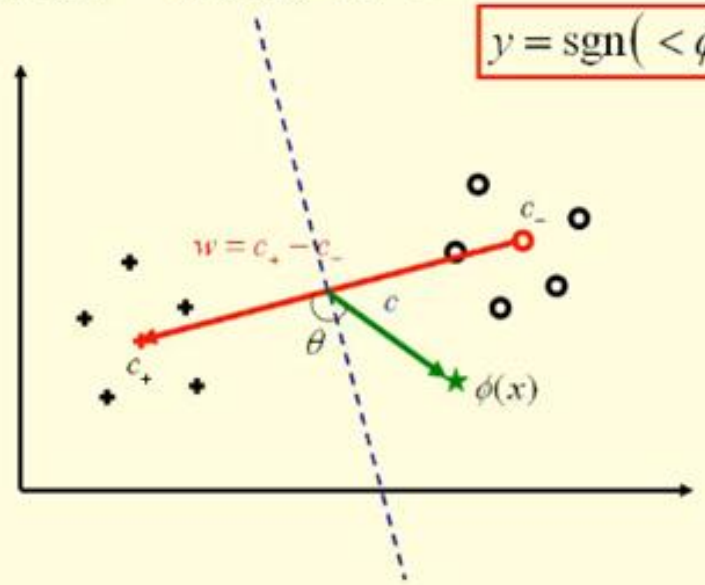


$$y = -1 \Leftrightarrow \frac{\pi}{2} < \theta \le \pi \Leftrightarrow -1 \le \cos\theta < 0 \Leftrightarrow -1 \le \frac{<\phi(x) - c, w>}{\sqrt{\|\phi(x) - c\|}\sqrt{\|w\|}} < 0 \Leftrightarrow <\phi(x) - c, w> \quad < 0$$

A Simple Recognition Algorithm

Testing Sample: $(x,y) \xrightarrow{\phi} (\phi(x),y) \rightarrow y = ?$

$$y = \mathrm{sgn}\left(< \phi(x) - c, w > \right)$$



$$y = 1 \Leftrightarrow 0 \le \theta < \frac{\pi}{2} \Leftrightarrow 0 < \cos\theta \le 1 \Leftrightarrow 0 < \frac{< \phi(x) - c, w >}{\sqrt{\|\phi(x) - c\|}\sqrt{\|w\|}} \le 1 \Leftrightarrow < \phi(x) - c, w > \quad > 0$$

$$y = -1 \Leftrightarrow \frac{\pi}{2} < \theta \le \pi \Leftrightarrow -1 \le \cos\theta < 0 \Leftrightarrow -1 \le \frac{< \phi(x) - c, w >}{\sqrt{\|\phi(x) - c\|}\sqrt{\|w\|}} < 0 \Leftrightarrow < \phi(x) - c, w > \quad < 0$$

A Simple Recognition Algorithm

- How to compute $y = \mathrm{sgn}(< \phi(x) - c, w >)$?
- One way:
  - $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset R^d \times \{1, -1\} \xrightarrow{\phi} \{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\} \subset H \times \{1, -1\}$ (feature mapping)
  - $(x, y) \xrightarrow{\phi} (\phi(x), y) \rightarrow y = ?$

$$y = \mathrm{sgn}(< \phi(x) - c, w >)$$

A Simple Recognition Algorithm

How to compute $y = \mathrm{sgn}\left(< \phi(x) - c, w >\right)?$

The other way

$$< \phi(x) - c, w >= w^T \left(\phi(x) - c\right) = w^T \phi(x) - w^T c = \left(c_+ - c_-\right)^T \phi(x) - \frac{1}{2}\left(c_+ - c_-\right)^T \left(c_+ + c_-\right)$$

$$= \left(\frac{1}{m_+} \sum_{(i|y_i=1)} \phi(x_i) - \frac{1}{m_-} \sum_{(i|y_i=-1)} \phi(x_i) \right)^T \phi(x) - \left(\frac{1}{2} {c_+}^T c_+ - {c_+}^T c_- + \frac{1}{2} {c_-}^T c_- \right)$$

$$= \left(\frac{1}{m_+} \sum_{(i|y_i=1)} \phi(x_i)^T \phi(x) - \frac{1}{m_-} \sum_{(i|y_i=-1)} \phi(x_i)^T \phi(x) \right) - b$$

$$= \left(\frac{1}{m_+} \sum_{(i|y_i=1)} K(x_i, x) - \frac{1}{m_-} \sum_{(i|y_i=-1)} K(x_i, x) \right) - b$$

What is b? Exercise!

Note that if we only have the Kernel Function K, we still can find y.

Some Widely Used Kernel Functions

Linear Kernel

$$k(x,z) = < x, z >$$

Polynomial Kernel

$$k(x,z) = (< x, z > +1)^r, r \in Z^+$$

RBF(Gaussian) Kernel

$$k(x,z) = \exp(\frac{-\| x - z \|^2}{2\sigma^2}), \sigma \in R - \{0\}$$

Kernel Function and Feature Mapping

- Is ϕ necessary? Not necessary!
- Can we only employ κ ? Yes!
- What kind of κ can be used? Finitely positive semi-definite!
- Given a feature mapping ϕ, can we find a corresponding kernel function κ computing the dot product in feature space? Yes!
- Given a kernel function κ, can we construct a feature space H (i.e., a feature mapping ϕ) such that κ computes the dot product in H? Yes!

The review of the Linear regression

Given m observation $x_1, \ldots, x_N$ **and the corresponding responses** $y_1, \ldots, y_N$ **, respectively.**

$$X_1 = \begin{bmatrix} 1 & (x_1)^T \\ \vdots & \vdots \\ 1 & (x_N)^T \end{bmatrix}$$

The review of the Linear regression

Normal Equation

$$J(\theta) = \frac{1}{2}(X_1\theta - y)^T(X_1\theta - y)$$

$$\nabla_\theta J(\theta) = \frac{1}{2}\left[X_1^T X_1\theta + X_1^T X_1\theta - X_1^T y - X_1^T y\right]$$

$$= X_1^T X_1\theta - X_1^T y \overset{set}{=} 0$$

$$X_1^T X_1\theta = X_1^T y \qquad \theta = (X_1^T X_1)^{-1} X_1^T y$$

Kernel-based Linear Regression

In addition, a feature mapping ϕ and the corresponding kernel function κ are also given

$$X_1 = \begin{bmatrix} 1 & \phi(x_1)^T \\ \vdots & \vdots \\ 1 & \phi(x_N)^T \end{bmatrix} = \left[\mathbf{1}_{N\times 1} \, , \mathrm{X} \right]$$

$$X = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Kernel-based Linear Regression

Again,

$$X_1^T X_1 \theta = X_1^T y \Rightarrow \theta = \left(X_1^T X_1\right)^{-1} X_1^T y$$

Note that

$$\theta = \left(X_1^T X_1\right)\left(X_1^T X_1\right)^{-1}\left(X_1^T X_1\right)^{-1} X_1^T y = \left(X_1^T X_1\right)\left(X_1^T X_1\right)^{-2} X_1^T y = X_1^T \alpha,$$

where $\alpha = X_1\left(X_1^T X_1\right)^{-2} X_1^T y$

$$X_1 = \begin{bmatrix} 1 & \phi(x_1)^T \\ \vdots & \vdots \\ 1 & \phi(x_N)^T \end{bmatrix} = \left[\mathbf{1}_{N\times 1}, \mathrm{X}\right] \qquad X = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

Kernel-based Linear Regression

Now we want to find α

$$X_1^T X_1 \theta = X_1^T y$$

Substitute θ **by** $X_1^T \alpha$

$$X_1^T X_1 X_1^T \alpha = X_1^T y$$

and multiply both sides by X

$$X_1 X_1^T X_1 X_1^T \alpha = X_1 X_1^T y$$

$$\Rightarrow K_1^2 \alpha = K_1 y, \qquad where \;\; K_1 = X_1 X_1^T$$

$$\Rightarrow K_1 \alpha = y, \qquad if \;\; \det(K) \neq 0$$

$$\Rightarrow \alpha = K_1^{-1} y$$

Kernel-based Linear Regression

$$K_1 = X_1 X_1^T = [\mathbf{1}_{N\times 1}, X]\begin{bmatrix}\mathbf{1}_{1\times N}\\ X^T\end{bmatrix} = \mathbf{1}_{N\times N} + XX^T$$

$$= \mathbf{1}_{N\times N} + K$$

$$X = \begin{bmatrix}\phi(x_1)^T\\ \vdots\\ \phi(x_N)^T\end{bmatrix}$$

$$XX^T = \begin{bmatrix}\phi\left(x_1^{(1)}\right) & \cdots & \phi\left(x_1^{(n)}\right)\\ \phi\left(x_2^{(1)}\right) & \cdots & \phi\left(x_2^{(n)}\right)\\ \vdots & \vdots & \vdots\\ \phi\left(x_N^{(1)}\right) & \cdots & \phi\left(x_N^{(n)}\right)\end{bmatrix}\begin{bmatrix}\phi\left(x_1^{(1)}\right) & \cdots & \phi\left(x_N^{(1)}\right)\\ \vdots & \cdots & \vdots\\ \vdots & \vdots & \vdots\\ \phi\left(x_1^{(n)}\right) & \cdots & \phi\left(x_N^{(n)}\right)\end{bmatrix}$$

Kernel-based Linear Regression

$$
= \begin{bmatrix}
\langle \phi(x_1), \phi(x_1) \rangle & \cdots & \langle \phi(x_1), \phi(x_N) \rangle \\
\vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots \\
\langle \phi(x_N), \phi(x_1) \rangle & \cdots & \langle \phi(x_N), \phi(x_N) \rangle
\end{bmatrix}
$$

$$
= \begin{bmatrix}
K(x_1, x_1) & \cdots & K(x_1, x_N) \\
\vdots & \cdots & \vdots \\
\vdots & \vdots & \vdots \\
K(x_N, x_1) & \cdots & K(x_N, x_N)
\end{bmatrix} = K
$$

Kernel-based Linear Regression

Hence,

$$\alpha = \left(\mathbf{1}_{N\times N} + K\right)^{-1} y$$

$$\theta = X^T \alpha = X^T \left(\mathbf{1}_{N\times N} + K\right)^{-1} y$$

$$X = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix} \qquad K = XX^T \qquad K_1 = \mathbf{1}_{N\times N} + K$$

$$X_1 = \left[\mathbf{1}_{N\times 1}, \mathrm{X}\right]$$

X is $N \times n$

$\bar{X}^T$ is $n \times N$

K is $N \times N$

K_1 is $N \times N$

X^T is $(n+1) \times N$

$\mathbf{1}_{N\times N} + K$ is $N \times N$

y is $N \times 1$

θ is $(n+1) \times 1$

α is $N \times 1$

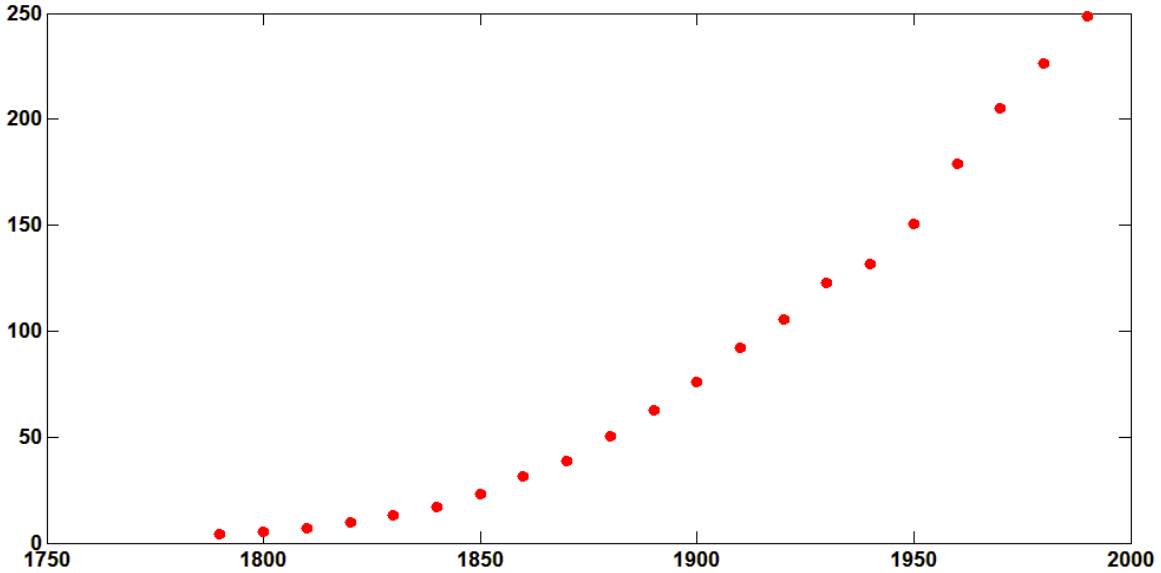
Kernel-based Linear Regression

For a input x, we have

$$
\begin{aligned}
f(x) &= \left[1, \phi(x)^T\right]\theta = \left[1, \phi(x)^T\right] X_1{}^T \alpha \\
&= \left[1, \phi(x)^T\right]\begin{bmatrix} \mathbf{1}_{1\times N} \\ X^T \end{bmatrix}\alpha \\
&= \left(\mathbf{1}_{1\times N} + \left(X\phi(x)\right)^T\right)\alpha \\
&= \left[1+\kappa(x_1, x), \ldots, 1+\kappa(x_N, x)\right]\alpha \\
&= \left(\alpha_1 + \ldots + \alpha_N\right) + \alpha_1\kappa(x_1, x) + \ldots + \alpha_N\kappa(x_N, x)
\end{aligned}
$$

Example: Predicting the US Population

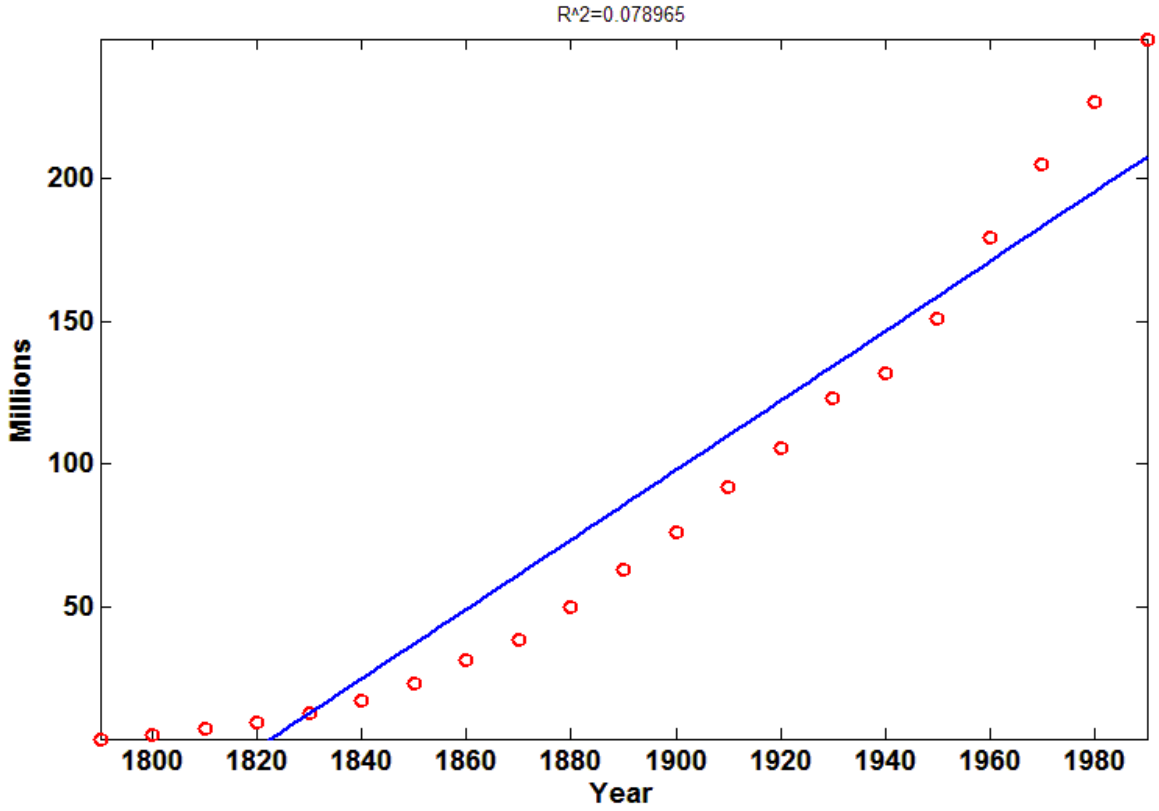
```
1 –     clear
2 –     clc
3 –     close all
4
5 –     load census;
6
7 –     x = cdate;
8 –     y = pop;
9 –     plot(x,y,'ro');
10 –    hold on
```



```
%% Linear regression

[N,d]=size(x);
X=[ones(N,1),x];
hw=pinv(X'*X)*X'*y;
yhat=X*hw;
R2=norm(y-yhat)^2/norm(y-mean(y))^2;

xfit=linspace(min(x),max(x),100)';
yfit=[ones(size(xfit,1),1),xfit]*hw;
plot(xfit,yfit,'b-');
xlabel('Year');
ylabel('Millions');
axis([min(x),max(x),min(y),max(y)]);
title(['R^2=' num2str(R2)]);
pause
```


R^2=0.078965
Millions
200
150
100
50
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
Year

```
% %%Kernel-based linear regression
sigma=[1:1:10,20:10:100,200:100:1000];
figure
for i=1:length(sigma)
    K=rbf_kernel(x,x,sigma(i));
    ha=pinv(ones(size(K))+K)*y;
    yhat=(ones(size(K))+K)*ha;
    R2=norm(y-yhat)^2/norm(y-mean(y))^2;
```

$$\alpha = \left(\mathbf{1}_{N\times N} + K\right)^{-1} y$$

$$K_1 \alpha = y$$

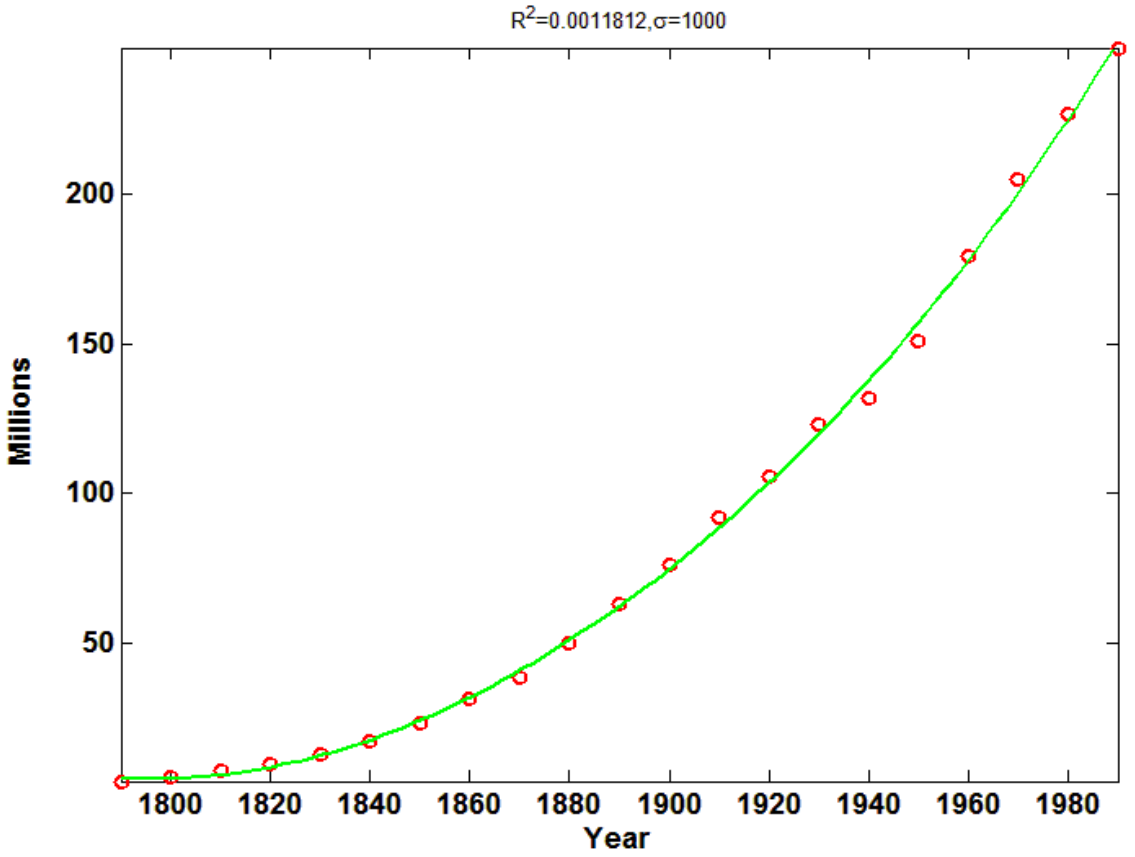
$$K_1 = X_1 X_1^{\ T} = \left[\mathbf{1}_{N\times 1}, X\right]\begin{bmatrix}\mathbf{1}_{1\times N}\\ X^T\end{bmatrix} = \mathbf{1}_{N\times N} + XX^T$$

$$= \mathbf{1}_{N\times N} + K$$

```
38 -        Ktest=rbf_kernel(xfit,x,sigma(i));
39 -        yfit=(ones(size(Ktest))+Ktest)*ha;
40 -        plot(x,y,'ro');
41 -        hold on
42 -        plot(xfit,yfit,'g-');
43 -         xlabel('Year');
44 -         ylabel('Millions');
45 -         axis([min(x),max(x),min(y),max(y)]);
46 -         title(['R^2=' num2str(R2) ',\sigma=', num2str(sigma(i))]);
47 -         hold off
48 -         pause(0.5)
49 -    end
```

$$f(x)=\left[1,\phi(x)^T\right]\theta=\left[1,\phi(x)^T\right]X_1{}^T\alpha$$

$$=\left[1+\kappa(x_1,x),\ldots,1+\kappa(x_N,x)\right]\alpha$$


R^2=0.0011812,σ=1000
Millions
200
150
100
50
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
Year

Principal Component Analysis (PCA)

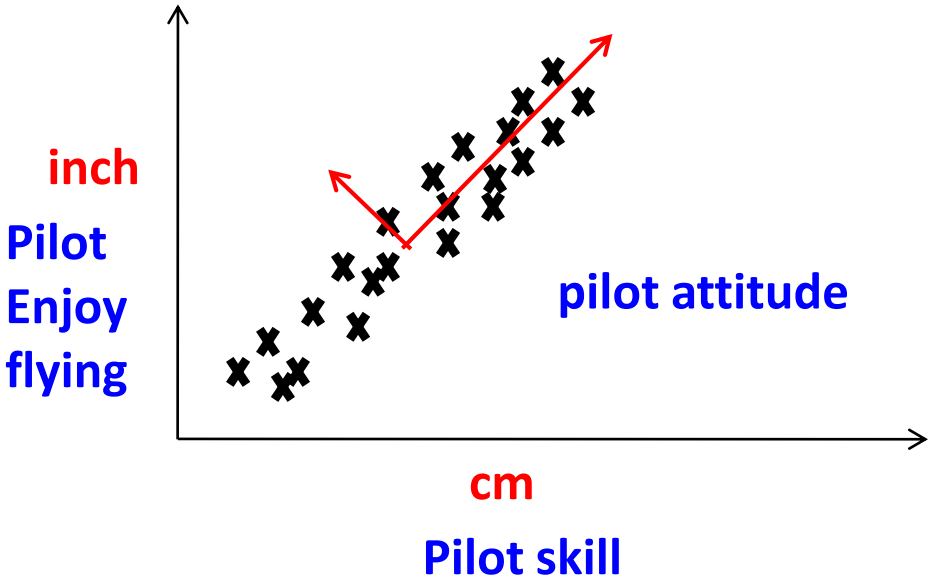
Kernel –Based PCA (KPCA)

Principal Component Analysis

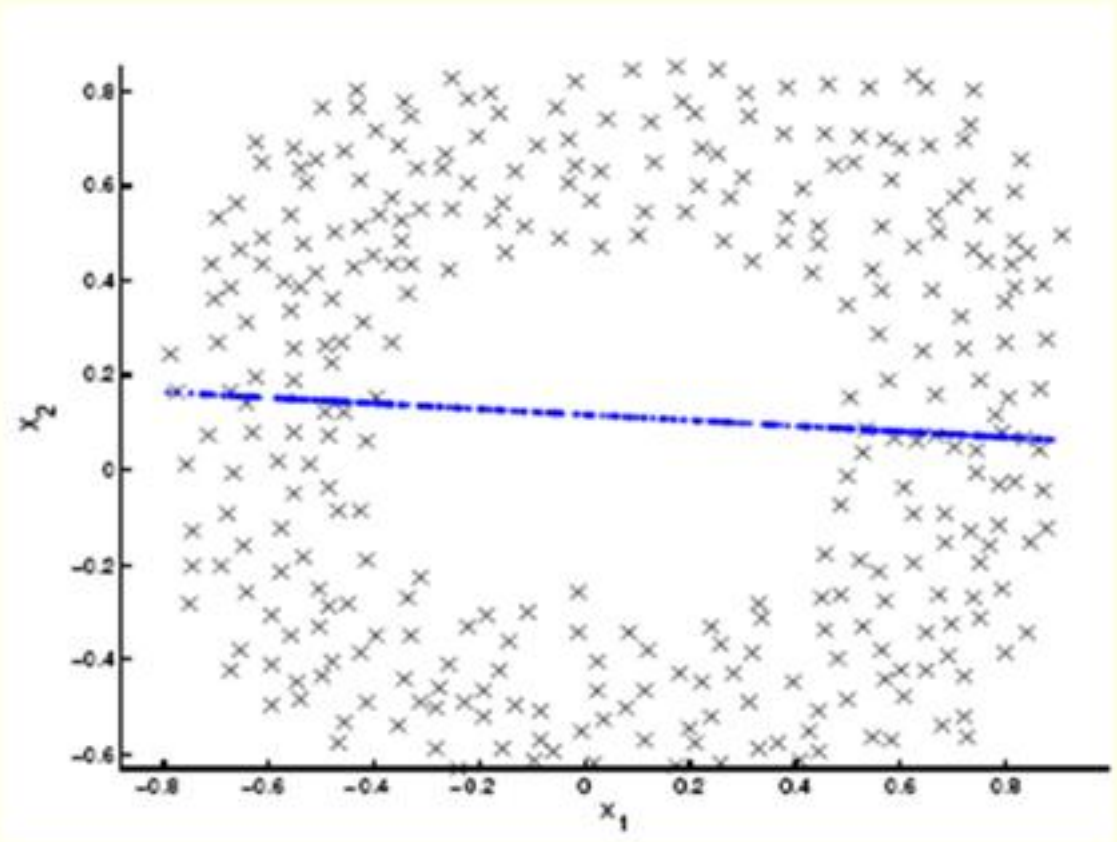
Given $\left\{x^{(1)},\cdots,x^{(m)}\right\}$ $\left\{x_1,\cdots,x_N\right\}$

$x^{(i)} \in R^n$ $x_i \in R^n$

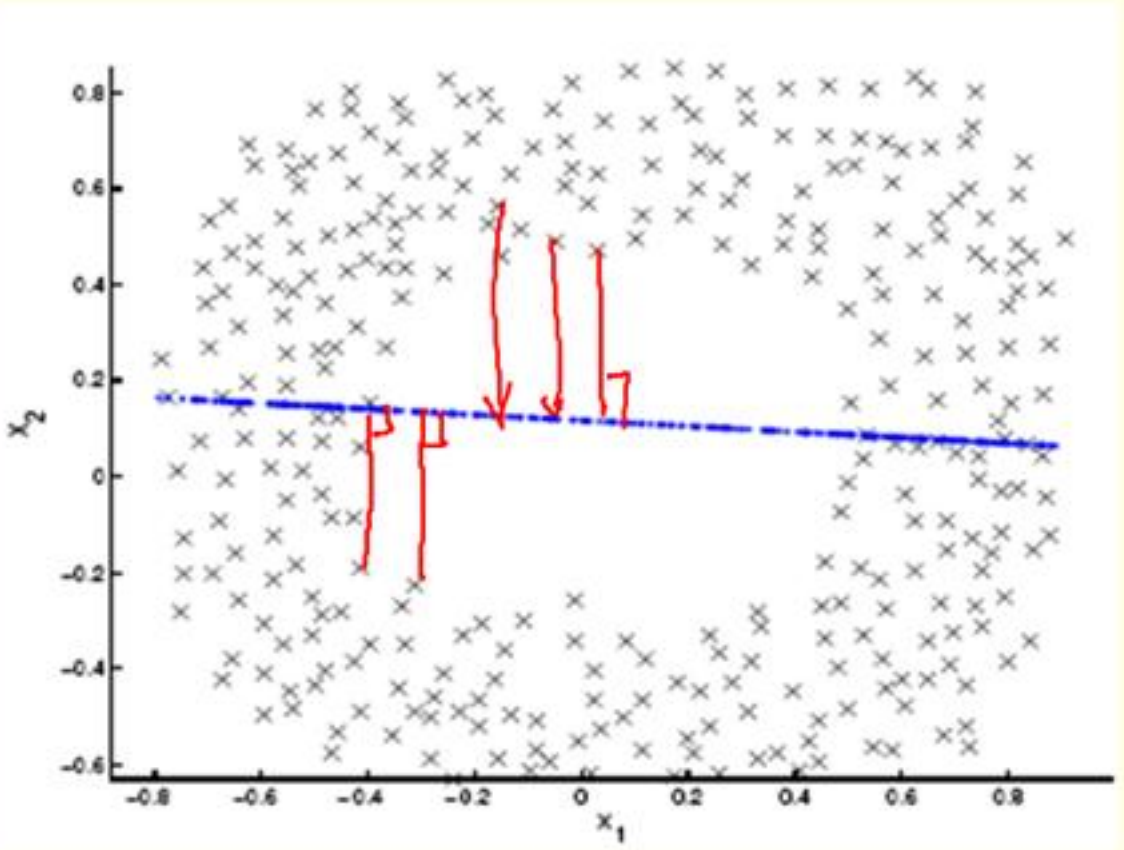
Reduce it to *p-dimensional*, *maxmum variance* $(p<n)$



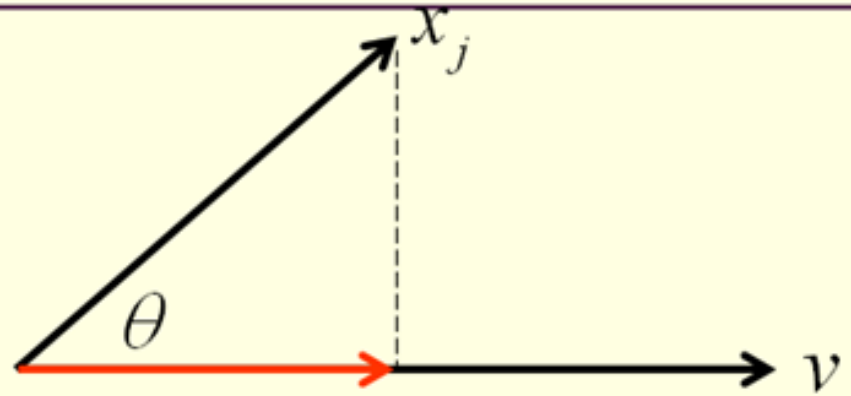
Principal Component Analysis



Principal Component Analysis

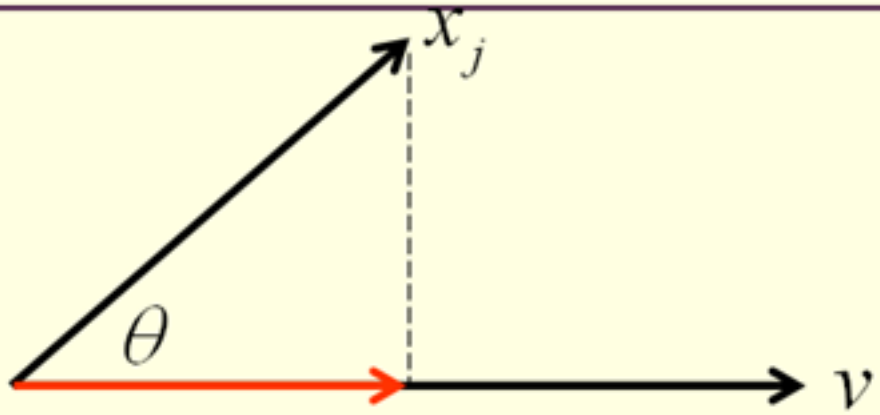


Principal Component Analysis



$$< x_j, v >= \| x_j \| \cdot \| v \| \cos\theta \Rightarrow \begin{cases} < x_j, v > \geq 0, & \text{if } 0 \leq \theta \leq \dfrac{\pi}{2} \\ < x_j, v > < 0, & \text{if } \dfrac{\pi}{2} < \theta \leq \pi \end{cases}$$

Principal Component Analysis

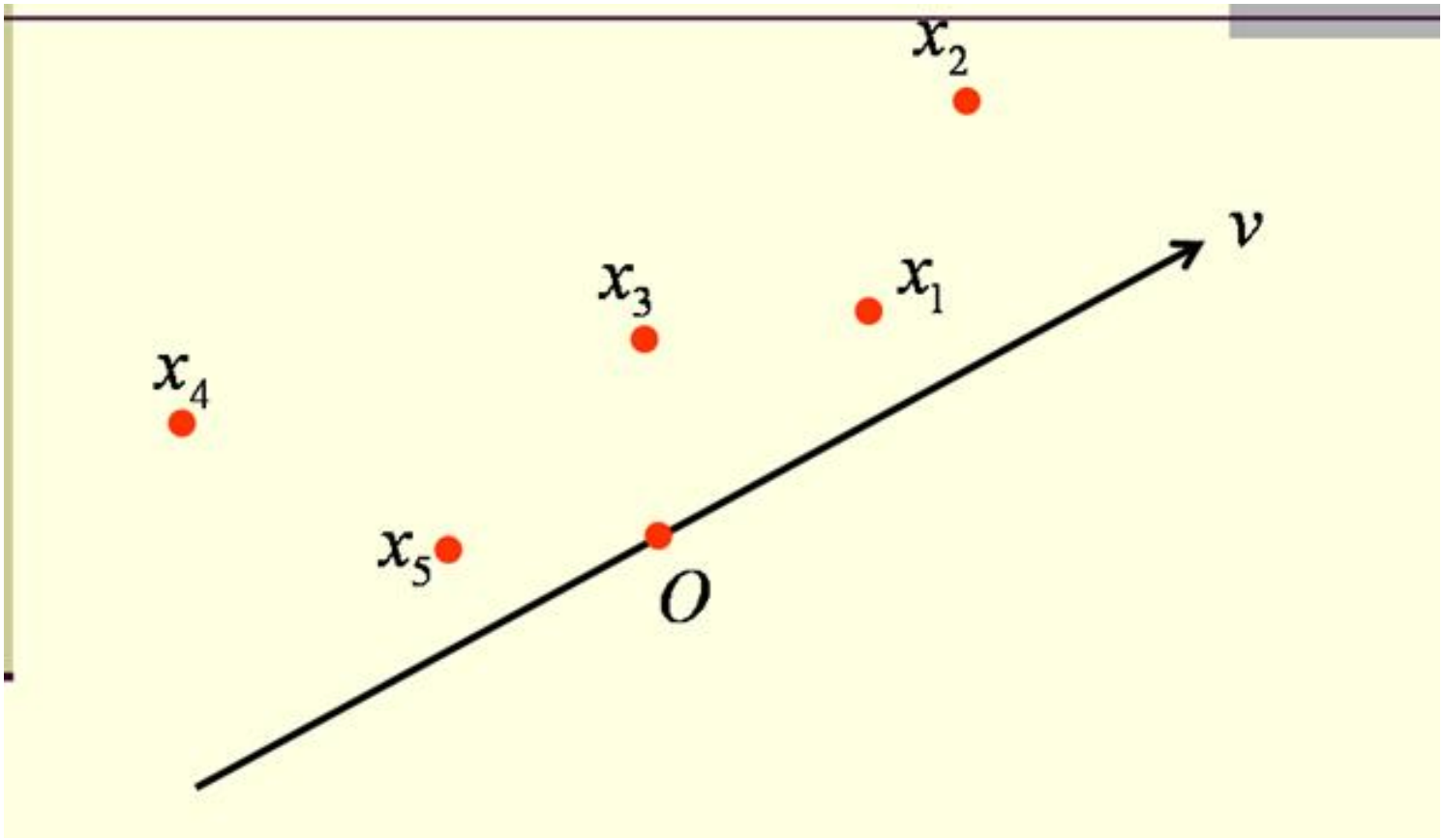


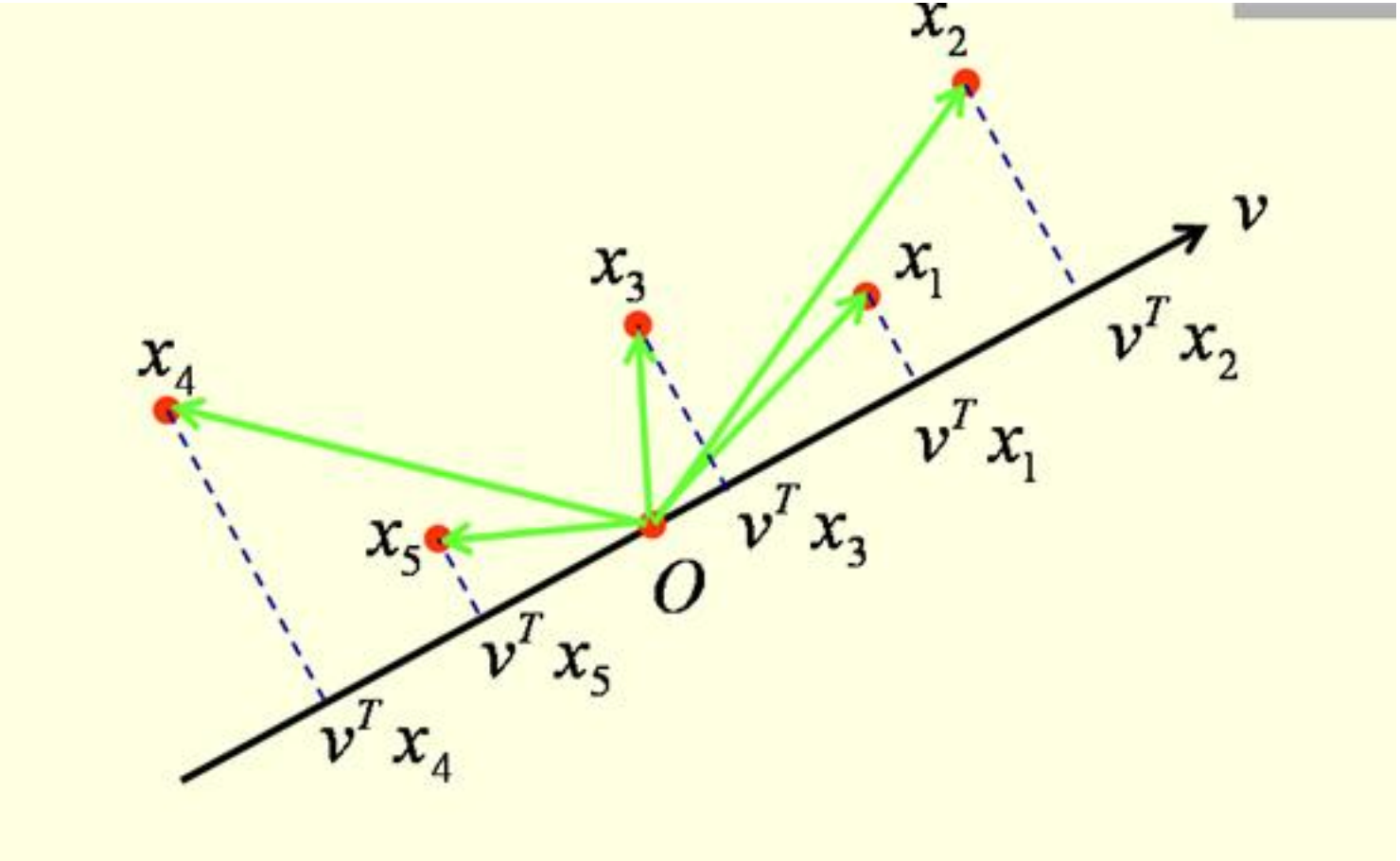
$$(\| x_j \| \cos\theta)\frac{v}{\| v \|} = \| x_j \| \frac{< x_j, v >}{\| x_j \| \cdot \| v \|}\frac{v}{\| v \|} = \frac{< x_j, v >}{\| v \|^2} v$$

If $\| v \| = 1$, then $(\| x_j \| \cos\theta)\frac{v}{\| v \|} = < x_j, v > v,$

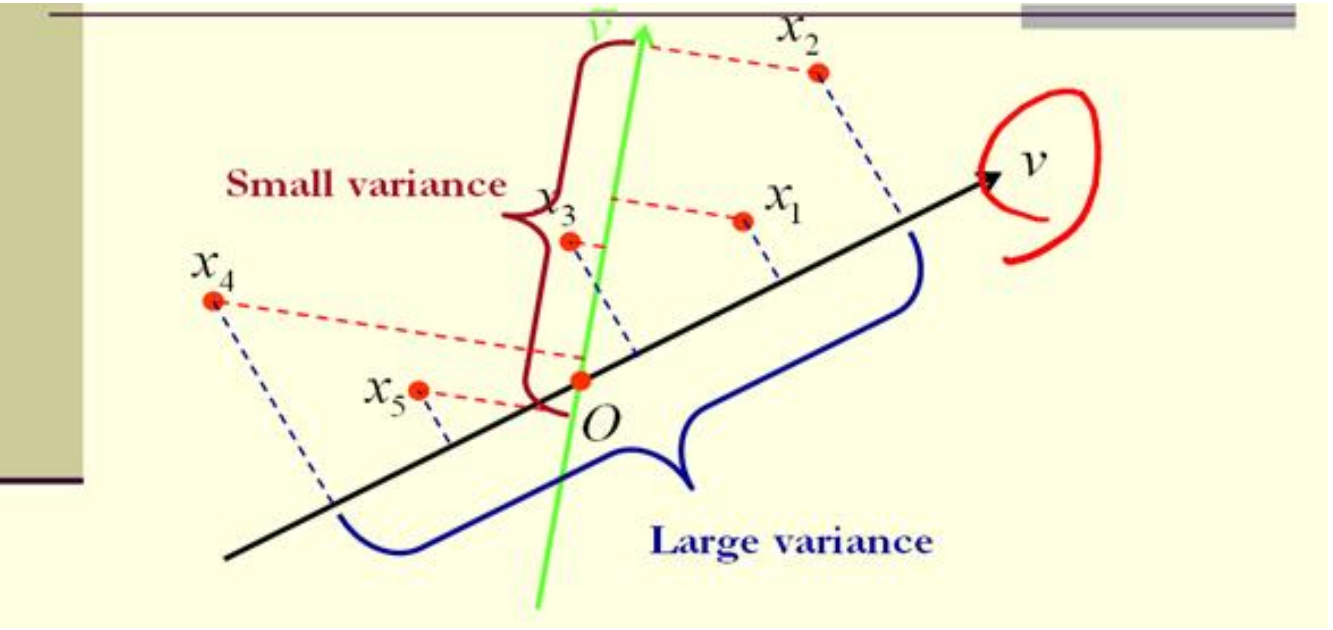
i.e., the coefficient is $< x_j, v >= x_j^T v = v^T x_j.$

Principal Component Analysis




x_2
v
x_3
x_1
x_4
$v^T x_2$
$v^T x_1$
x_5
$v^T x_3$
O
$v^T x_5$
$v^T x_4$

Principal Component Analysis



Pre-processing:

1. set $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

2. Replace x_i **with** $x_i - \mu$

Zero out mean

Principal Component Analysis

- The projections of the all pixels x_j onto this normalized direction v are

$$v^T x_1, \ldots, v^T x_N.$$

- The variance of the projections is

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(v^T x_i - 0)^2 = \frac{1}{N}\sum_{i=1}^{N}(v^T x_i)(v^T x_i) = \frac{1}{N}\sum_{i=1}^{N}(v^T x_i)(v^T x_i)^T$$

$$= \frac{1}{N}\sum_{i=1}^{N} v^T x_i x_i^T v = v^T\left(\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T\right)v = v^T C v$$

where

$$C = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T.$$

Principal Component Analysis

- Note that

$$C = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T = \frac{1}{N}[x_1, \ldots, x_N]\begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}.$$

- If $X^T = [x_1, \ldots, x_N]$, then

$$C = \frac{1}{N} X^T X.$$

Principal Component Analysis

- The first principal vector can be found by the following equation:

$$v = \arg\max_{v \in R^d,\ \|v\|=1} v^T C v.$$

Principal Component Analysis

- The first principal vector can be found by the following equation:

$$v = \arg\max_{v \in R^d,\ \|v\|=1} v^T C v.$$

Lagrangian: $f(v,\lambda) = v^T C v - \lambda(v^T v - 1)$

$$\frac{\partial f}{\partial v} = 2Cv - 2\lambda v = 0 \Rightarrow Cv = \lambda v$$

$$\frac{\partial f}{\partial \lambda} = v^T v - 1 = 0 \Rightarrow v^T v = 1$$

$$v^T C v = v^T \lambda v$$

$$= \lambda v^T v = \lambda = \sigma^2$$

Principal Component Analysis

- The first principal vector can be found by the following equation:

$$v = \arg\max_{v \in R^d,\ \|v\|=1} v^T C v.$$

- This is equivalent to find the largest eigenvalue of the following eigenvalue problem:

$$\begin{cases} Cv = \lambda v \\ \|v\| = 1 \end{cases}.$$

Steps

Step.1 **Pre-processing:**

1. set $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

2. Replace x_i **with** $x_i - \mu$

} **Zero out mean**

Step.2 Calculate matrix (correlation coefficient matrix)

$$C = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$$

Step.3

Calculate eigenvalues and eigenvectors：

$$Cv = \lambda v$$

$$\left|\lambda I - C\right| = 0$$

$$\lambda_1 \geq \lambda_2 \geq \cdots, \geq \lambda_n \geq 0$$

Step.4

计算主成分贡献率及累计贡献率

▲贡献率：

$$\frac{\lambda_i}{\sum_{i=1}^{n}\lambda_i} \qquad (i=1,2,\cdots,n)$$

▲累计贡献率：

$$\frac{\sum_{j=1}^{i}\lambda_j}{\sum_{j=1}^{n}\lambda_j} \qquad (i=1,2,\cdots,k)$$

一般取累计贡献率达85—95%的特征值 $\lambda_1,\lambda_2,\cdots,\lambda_k$ 所对应的第一、第二、…、第k（k≤n）个主成分。

Now x_i can be represented as

$$x_i = \sum_{j=1}^{n} \left(x_i^T v_j \right) v_j$$

x_i can be also approximated by

$$x_i \approx \sum_{j=1}^{k} \left(x_i^T v_j \right) v_j$$

Principal Component Analysis

-Visualization

-Compression

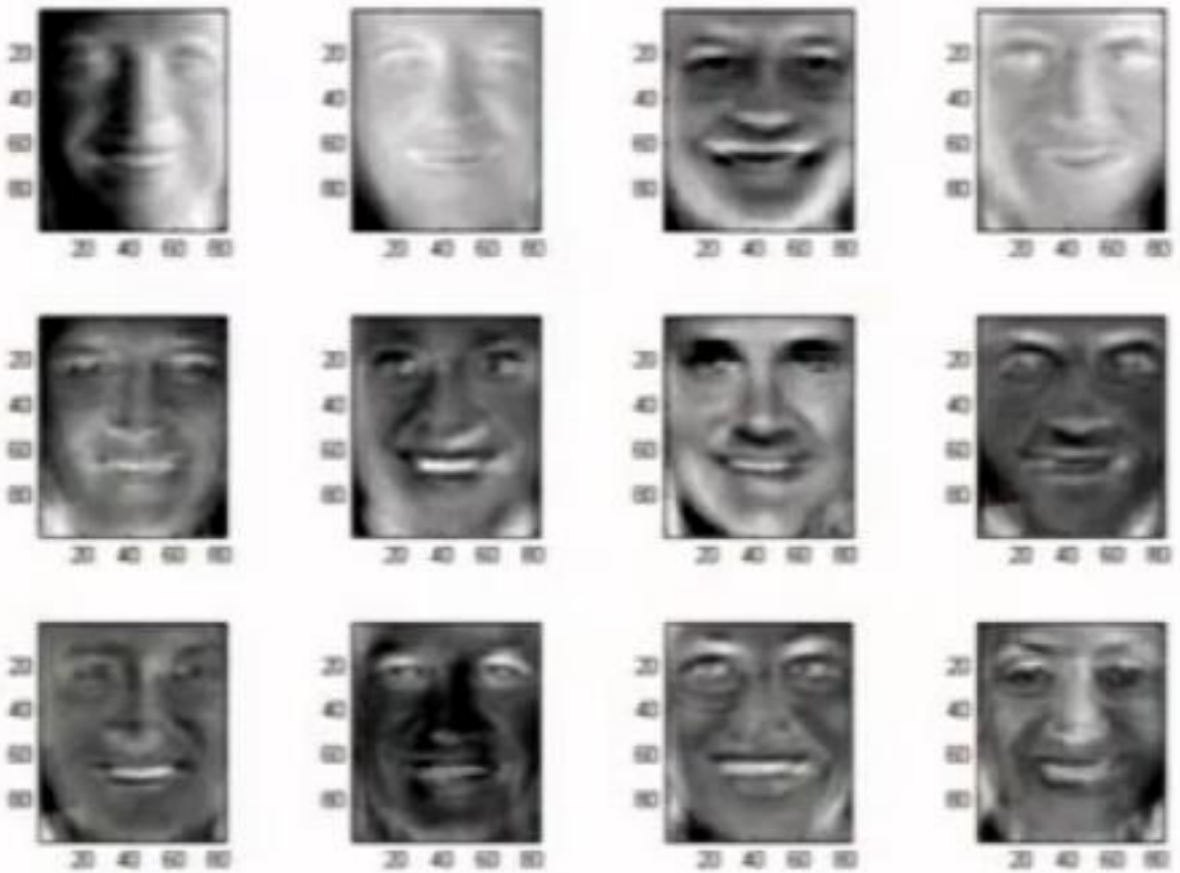
-Anomaly detection

-Matching/Distance calculations

Principal Component Analysis

Training set

Principal Component Analysis



Principal components learned (eigenfaces)

三、主成分分析方法应用实例（一）

下面，我们根据表1给出的数据，对某农业生态经济系统做主成分分析：

表1　某农业生态经济系统各区域单元的有关数据

样本序号	x_1：人口密度(人/km^2)	x_2：人均耕地面积(ha)	x_3：森林覆盖率(%)	x_4：农民人均纯收入(元/人)	x_5：人均粮食产量(kg/人)	x_6：经济作物占农作物播面比例(%)	x_7：耕地占土地面积比率(%)	x_8：果园与林地面积之比(%)	x_9：灌溉田占耕地面积之比(%)
1	363.912	0.352	16.101	192.11	295.34	26.724	18.492	2.231	26.262
2	141.503	1.684	24.301	1752.35	452.26	32.314	14.464	1.455	27.066
3	100.695	1.067	65.601	1181.54	270.12	18.266	0.162	7.474	12.489
4	143.739	1.336	33.205	1436.12	354.26	17.486	11.805	1.892	17.534
5	131.412	1.623	16.607	1405.09	586.59	40.683	14.401	0.303	22.932

6	68.337	2.032	76.204	1540.29	216.39	8.128	4.065	0.011	4.861
7	95.416	0.801	71.106	926.35	291.52	8.135	4.063	0.012	4.862
8	62.901	1.652	73.307	1501.24	225.25	18.352	2.645	0.034	3.201
9	86.624	0.841	68.904	897.36	196.37	16.861	5.176	0.055	6.167
10	91.394	0.812	66.502	911.24	226.51	18.279	5.643	0.076	4.477
11	76.912	0.858	50.302	103.52	217.09	19.793	4.881	0.001	6.165
12	51.274	1.041	64.609	968.33	181.38	4.005	4.066	0.015	5.402
13	68.831	0.836	62.804	957.14	194.04	9.11	4.484	0.002	5.79
14	77.301	0.623	60.102	824.37	188.09	19.409	5.721	5.055	8.413
15	76.948	1.022	68.001	1255.42	211.55	11.102	3.133	0.01	3.425
16	99.265	0.654	60.702	1251.03	220.91	4.383	4.615	0.011	5.593
17	118.505	0.661	63.304	1246.47	242.16	10.706	6.053	0.154	8.701
18	141.473	0.737	54.206	814.21	193.46	11.419	6.442	0.012	12.945
19	137.761	0.598	55.901	1124.05	228.44	9.521	7.881	0.069	12.654
20	117.612	1.245	54.503	805.67	175.23	18.106	5.789	0.048	8.461
21	122.781	0.731	49.102	1313.11	236.29	26.724	7.162	0.092	10.078

步骤如下：（1）将表1中的数据作标准差标准化处理，然后将它们代入公式计算相关系数矩阵（见表2）。

2 相关系数矩阵

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_1	1	-0.327	-0.714	-0.336	0.309	0.408	0.79	0.156	0.744
x_2	-0.33	1	-0.035	0.644	0.42	0.255	0.009	-0.078	0.094
x_3	-0.71	-0.035	1	0.07	-0.74	-0.755	-0.93	-0.109	-0.924
x_4	-0.34	0.644	0.07	1	0.383	0.069	-0.05	-0.031	0.073
x_5	0.309	0.42	-0.74	0.383	1	0.734	0.672	0.098	0.747
x_6	0.408	0.255	-0.755	0.069	0.734	1	0.658	0.222	0.707
x_7	0.79	0.009	-0.93	-0.046	0.672	0.658	1	-0.03	0.89
x_8	0.156	-0.078	-0.109	-0.031	0.098	0.222	-0.03	1	0.29
x_9	0.744	0.094	-0.924	0.073	0.747	0.707	0.89	0.29	1

（2）由相关系数矩阵计算特征值，以及各个主成分的贡献率与累计贡献率（见表3）。

表3. 特征值及主成分贡献率

主成分	特征值	贡献率(%)	累积贡献率(%)
z_1	4.661	51.791	51.791
z_2	2.089	23.216	75.007
z_3	1.043	11.589	86.596
z_4	0.507	5.638	92.234
z_5	0.315	3.502	95.736
z_6	0.193	2.14	97.876
z_7	0.114	1.271	99.147
z_8	0.0453	0.504	99.65
z_9	0.0315	0.35	100

由表3可知，第一，第二，第三主成分的累计贡献率已高达86.596%（大于85%），故只需要求出第一、第二、第三主成分z_1，z_2，z_3即可。

（3）对于特征值=4.6610，=2.0890，=1.0430分别求出其特征向量z_1，z_2，z_3，计算各变量x_1，x_2，…，x_9在主成分z_1，z_2，z_3上的投影。

Principal Component Analysis

	1	2	3	4
1	'变量'	'Z3'	'Z2'	'Z1'
2	'x1:人口密度'	-0.0599	0.3679	0.3421
3	'x2:人均耕地面积'	-0.0276	-0.6135	0.0572
4	'x3:森林覆盖率'	0.0929	-0.0661	-0.4464
5	'x4:人均纯收入'	0.0362	-0.6006	0.0193
6	'x5:人均粮食产量'	-0.0107	-0.3068	0.3765
7	'x6:经济作物所占比例'	0.1222	-0.1241	0.3793
8	'x7:耕地占土地面积比率'	-0.2461	0.0920	0.4322
9	'x8:果园与林地面积之比'	0.9497	0.0695	0.0914
10	'x9:灌溉田所占比率'	0.0898	0.0173	0.4464
11				

分析：

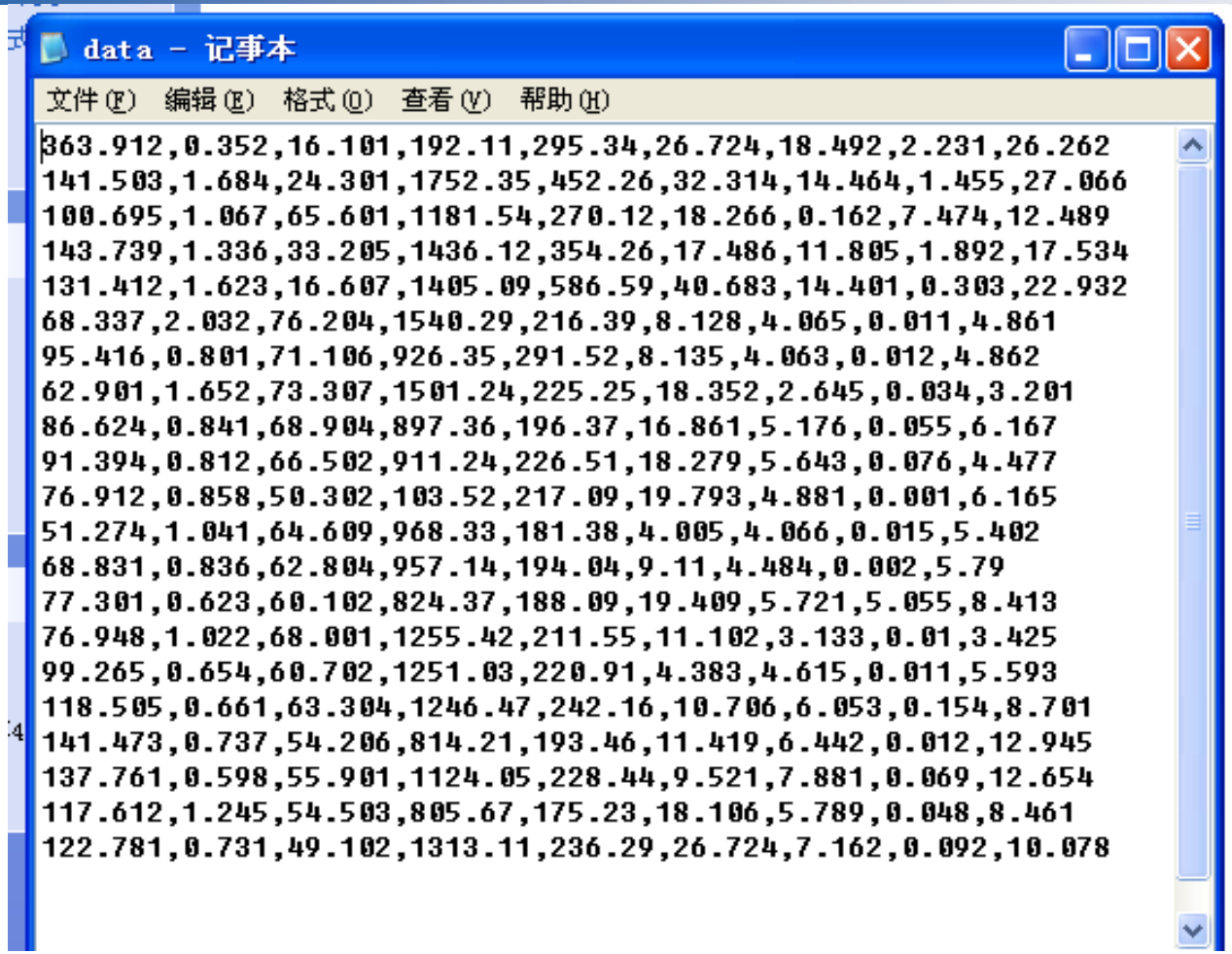
①第一主成分z_1与x_1，x_5，x_6，x_7，x_9呈显出较强的正相关，与x_3呈显出较强的负相关，而这几个变量则综合反映了生态经济结构状况，因此可以认为第一主成分z_1是生态经济结构的代表。

②第二主成分z_2与x_2，x_4，x_5呈显出较强的负相关，与x_1呈显出较强的正相关，其中，除了x_1为人口总数外，x_2，x_4，x_5都反映了人均占有资源量的情况，因此可以认为第二主成分z_2代表了人均资源量。

③第三主成分z_3，与x_8呈显出的正相关程度最高，其次是x_6，而与x_7呈负相关，因此可以认为第三主成分在一定程度上代表了经济作物和果园对农业经济的影响。

显然，用三个主成分Z_1、Z_2、Z_3代替原来9个变量（X_1，X_2，…，X_9），描述农业生态经济系统，可以使问题更进一步简化、明了。

Principal Component Analysis



Principal Component Analysis

```
clc
clear all
close all
data=load('data.txt');%读入数据，即表3.4.5中的数据
[Z,MU,SIGMA] = zscore(data);%将读入的数据做标准化处理
covMat = cov(Z);%计算相关系数矩阵
[eigVect,eigVal] = eig(covMat);%计算相关系数矩阵的特征值和特征向量
d=diag(eigVal);%提取特征值矩阵的对角线元素，即特征值
D=sort(d,'descend');%将各个特征值进行降序排列
rat1=D./sum(D);%计算贡献率
rat2=cumsum(D)./sum(D);%计算累积贡献率
```

Principal Component Analysis

```
%%导出特征值，贡献率及累积贡献率%%
result1(1,:)={'特征值','贡献率','累积贡献率'};
result1(2:10,1)=num2cell(D);
result1(2:10,2)=num2cell(rat1);
result1(2:10,3)=num2cell(rat2);
%%主成分载荷%%
result2(:,1)={'变量';'x1:人口密度';'x2:人均耕地面积';'x3:森林覆盖率';
    'x4:人均纯收入';'x5:人均粮食产量';'x6:经济作物所占比例';
    'x7:耕地占土地面积比率';'x8:果园与林地面积之比';'x9:灌溉田所占比率'};
result2(1,2:4)={'Z3','Z2','Z1'};%由累积贡献率可知，只需分析三个主成分，z1,z2,z3
result2(2:10,2:4)=num2cell(eigVect(:,7:9));%z1,z2,z3分别为特征值4.6611,2.0895,1.0430对应的特征向量
```

三、主成分分析方法应用实例（二）

128名男子身材的六项指标的样本相关系数矩阵

变量	身高（x_1）	座高（x_2）	胸围（x_3）	手臂长（x_4）	肋围（x_5）	腰围（x_6）
身高（x_1）	1	0.79	0.36	0.76	0.25	0.51
座高（x_2）	0.79	1	0.31	0.55	0.17	0.35
胸围（x_3）	0.36	0.31	1	0.35	0.64	0.58
手臂长（x_4）	0.76	0.55	0.35	1	0.64	0.38
肋围（x_5）	0.25	0.17	0.64	0.16	1	0.63
腰围（x_6）	0.51	0.35	0.58	0.38	0.63	1

相关系数矩阵：

pho =

1.0000	0.7900	0.3600	0.7600	0.2500	0.5100
0.7900	1.0000	0.3100	0.5500	0.1700	0.3500
0.3600	0.3100	1.0000	0.3500	0.6400	0.5800
0.7600	0.5500	0.3500	1.0000	0.1600	0.3800
0.2500	0.1700	0.6400	0.1600	1.0000	0.6300
0.5100	0.3500	0.5800	0.3800	0.6300	1.0000

调用pcacov函数作主成分分析:

```
>> [COEFF,latent,explained]=pcacov(pho)
```

```
COEFF =
   -0.4689   -0.3648    0.0922   -0.1224   -0.0797    0.7856
   -0.4037   -0.3966    0.6130    0.3264    0.0270   -0.4434
   -0.3936    0.3968   -0.2789    0.6557    0.4052    0.1253
   -0.4076   -0.3648   -0.7048   -0.1078   -0.2346   -0.3706
   -0.3375    0.5692    0.1643   -0.0193   -0.7305   -0.0335
   -0.4268    0.3084    0.1193   -0.6607    0.4899   -0.1788
```

特征值对应的特征向量

```
latent =
    3.2872
    1.4062
    0.4591
    0.4263
    0.2948
    0.1263
```

特征值

```
explained =
   54.7867
   23.4373
    7.6516
    7.1057
    4.9133
    2.1054
```

贡献率

```
>> cumsum(explained)
ans =
   54.7867
   78.2240
   85.8756
   92.9813
   97.8946
  100.0000
```

累积贡献率

从上述结果来看，前3个主成分的累积贡献率达到了85.8756%，因此可以用前3个主成分进行后续的分析；这样做虽然会有一定的信息损失，但损失不大，不影响大局。

	z1	z2	z3
x1	-0.4689	-0.3648	0.0922
x2	-0.4037	-0.3966	0.6130
x3	-0.3936	0.3968	-0.2789
x4	-0.4076	-0.3648	-0.7048
x5	-0.3375	0.5692	0.1643
x6	-0.4268	0.3084	0.1193

结果分析：

1.从第一主成分z1来看，它在每个变量上有近似的负载荷，说明每个变量对z1的重要性都差不多。当一个人的身材“五大三粗”，也就是说又高又胖时，x1，x2，…，x6都比较大，此时z1的值就比较小；反之，当一个人又矮又瘦时x1，x2，…，x6　都比较小，此时z1的值就比较大，所以可以认为第一主成分是身材的综合成分（或魁梧成分）。

2.从第二主成分Z2来看，它在变量x1,x2和X4上有近似的负载荷，在x3,x5和x6上有近似的正载荷，说明当x1,x2和X4增大时，Z2的值减小，当x3,x5和x6增大时，z2的值增大。当一个人的身材瘦高时，z2的值比较小；当一个人的身材矮胖时，z2的值比较大，所以可以认为第二主成分是身材的高矮和胖瘦的协调成分。

3.从第三主成分z3的表达式来看，它在变量x2上有比较大的正载荷，在x4上有比较大的负载荷，在其它变量上的载荷比较小，说明x2（坐高）和x4（手臂长）对z3的影响比较大，也就是说z3反映了坐高（即上半身）与手臂长之间的协调关系，这对做长袖上衣时制定衣服和袖子的长短提供了参考；所以可认为第三主成分z3是臂长成分。

featureNormalize

```
%FeatureNormalize.m Normalizes the features in X

mu = mean(X);
X_norm = bsxfun(@minus, X, mu);

sigma = std(X_norm);
X_norm = bsxfun(@rdivide, X_norm, sigma);
```

Standard deviation

corr(X,Y)

Principal Component Analysis

```
A = magic(5)

A =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

mean(A)

ans =

    13    13    13    13    13

X_norm = bsxfun(@rdivide, X_norm, sigma)'
```

```
X_norm = bsxfun(@minus, A, mean(A))
X_norm =

     4    11   -12    -5     2
    10    -8    -6     1     3
    -9    -7     0     7     9
    -3    -1     6     8   -10
    -2     5    12   -11    -4

sigma = std(X_norm)'

sigma =

   7.245688373094719e+000
   8.062257748298549e+000
   9.486832980505138e+000
   8.062257748298549e+000
   7.245688373094719e+000
```

Principal Component Analysis (PCA) algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\sum_{ij} = \frac{1}{M-1} X_i^T X_j \qquad \sum = \frac{1}{M-1} X^T X \qquad \mathrm{cov}(X)$$

Compute "eigenvectors" of matrix Σ:

```
[U,S,V] = svd(Sigma);
```

Covariance matrix always Satisfy "symmetric semi-positive Definite.

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

The first k columns $u^{(1)}$ to $u^{(k)}$ are the k dimensional vectors that we want

$$U_{reduced} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$Z \in R^k$$

$$Z^{(i)} = X^{(i)} U_{reduced}$$

Choosing k (number of principal components)

Average squared projection error:

Total variation in the data:

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \qquad (1\%)$$

"99% of variance is retained"

↓

90%~99%

Choosing k (number of principal components)

```
[U,S,V] = svd(Sigma)
```

Pick smallest value of k for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

(99% of variance retained)

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2}$$

0

0

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$$

Kernel-based PCA

Note that

$$C = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T = \frac{1}{N}\left[\phi(x_i),\cdots,\ \phi(x_N)\right]\begin{bmatrix}\phi(x_i)^T\\ \vdots \\ \phi(x_N)^T\end{bmatrix}$$

If $X^T = \left[\phi(x_i),\cdots,\ \phi(x_N)\right]$,

then

$$C = \frac{1}{N}X^T X$$

Kernel-based PCA

Note that the kernel matrix

$$K = \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_N) \\ \vdots & \ddots & \vdots \\ \kappa(x_N, x_1) & \cdots & \kappa(x_N, x_N) \end{bmatrix} =$$

$$\begin{bmatrix} \phi(x_1)^T \phi(x_1) & \cdots & \phi(x_1)^T \phi(x_N) \\ \vdots & \ddots & \vdots \\ \phi(x_N)^T \phi(x_1) & \cdots & \phi(x_N)^T \phi(x_N) \end{bmatrix}$$

$$= \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix} \left[\phi(x_1), \cdots, \phi(x_N) \right] = XX^T$$

Note that $X^T X \neq XX^T = K$

Kernel-based PCA

The eigenvalue problem of $K = XX^T$ **is**

$$\left(XX^T\right)u = \beta u.$$

Multiply both sides by X^T **:**

$$X^T\left(XX^T\right)u = \beta X^T u \Rightarrow C\left(X^T u\right) = \beta\left(X^T u\right).$$

This means that

$$X^T u$$

is a eigenvectors of C

However its norm is not sure 1.

Kernel-based PCA

$$\nu = \frac{1}{\|X^T u\|} X^T u = \frac{1}{\sqrt{u^T X X^T u}} X^T u$$

$$= \frac{1}{\sqrt{u^T (\beta u)}} X^T u = \frac{1}{\sqrt{\beta}} X^T u,$$

Where β is the corresponding eigenvalue of K .

Kernel-based PCA

The projection of the testing sample $\phi(x')$ on the i-th eigenvector can be computed by

$$v_i^T \phi(x') = \left(\frac{1}{\sqrt{\beta_i}} X^T u_i \right)^T \phi(x') = \frac{1}{\sqrt{\beta_i}} u_i^T X \phi(x')$$

$$= \frac{1}{\sqrt{\beta_i}} u_i^T \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix} \phi(x') = \frac{1}{\sqrt{\beta_i}} u_i^T \begin{bmatrix} \kappa(x_1, x') \\ \vdots \\ \kappa(x_N, x') \end{bmatrix}.$$

Summarization of Kernel-based PCA

☐Solve the following eigenvalue problem:

$$Ku_i = \beta_i u_i, \quad \beta_1 \geq \beta_2 \geq \cdots \geq \beta_k$$

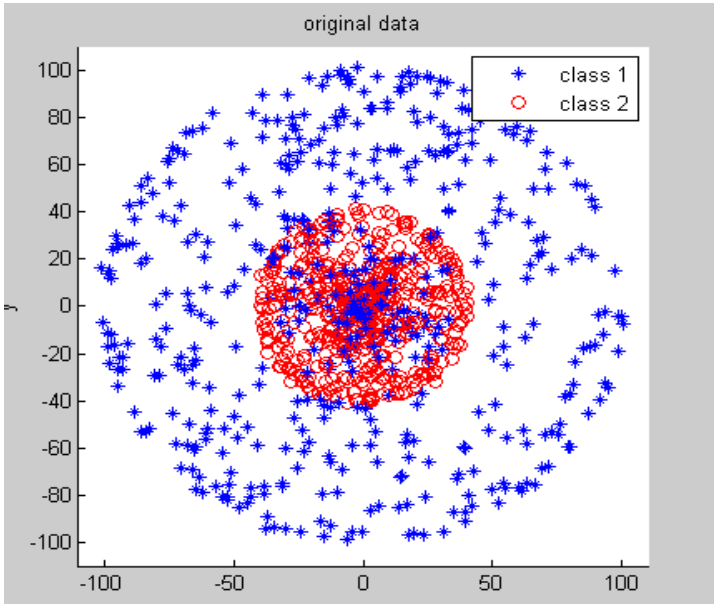
☐The projection of the testing sample $\phi(x')$ on the i-th eigenvector can be computed by

$$v_i^T \phi(x') = \frac{1}{\sqrt{\beta_i}} u_i^T \begin{bmatrix} k(x_1, x') \\ \vdots \\ k(x_k, x') \end{bmatrix}.$$

Pattern Classication for Synthetic Data

3.1.1. Data Description

We assume that we have equal numbers or data points distributed on two concentric sphere surfaces. If N is the total number of all data points, then we have $N/2$ class 1 points on a sphere of radius r_1, and $N/2$ class 2 points on a sphere of radius r_2. In the spherical coordinate system, the inclination (polar angle) θ is uniformly distributed in $[0, \pi]$, and the azimuth (azimuthal angle) ϕ is uniformly distributed in $[0, 2\pi)$ for both classes. Our observations of the data points are the (x, y, z) coordinates in the Cartesian coordinate system, and all the three coordinates are perturbed by a Gaussian noise of standard deviation σ_{noise}. We set $N = 1000$, $r_1 = 40$, $r_2 = 100$, $\sigma_{\text{noise}} = 1$, and give a 3D plot of the data in Figure 1.



原始数据为1000*3的矩阵，表示两类不同的点。
为节约篇幅，从中截取部分表示如下：

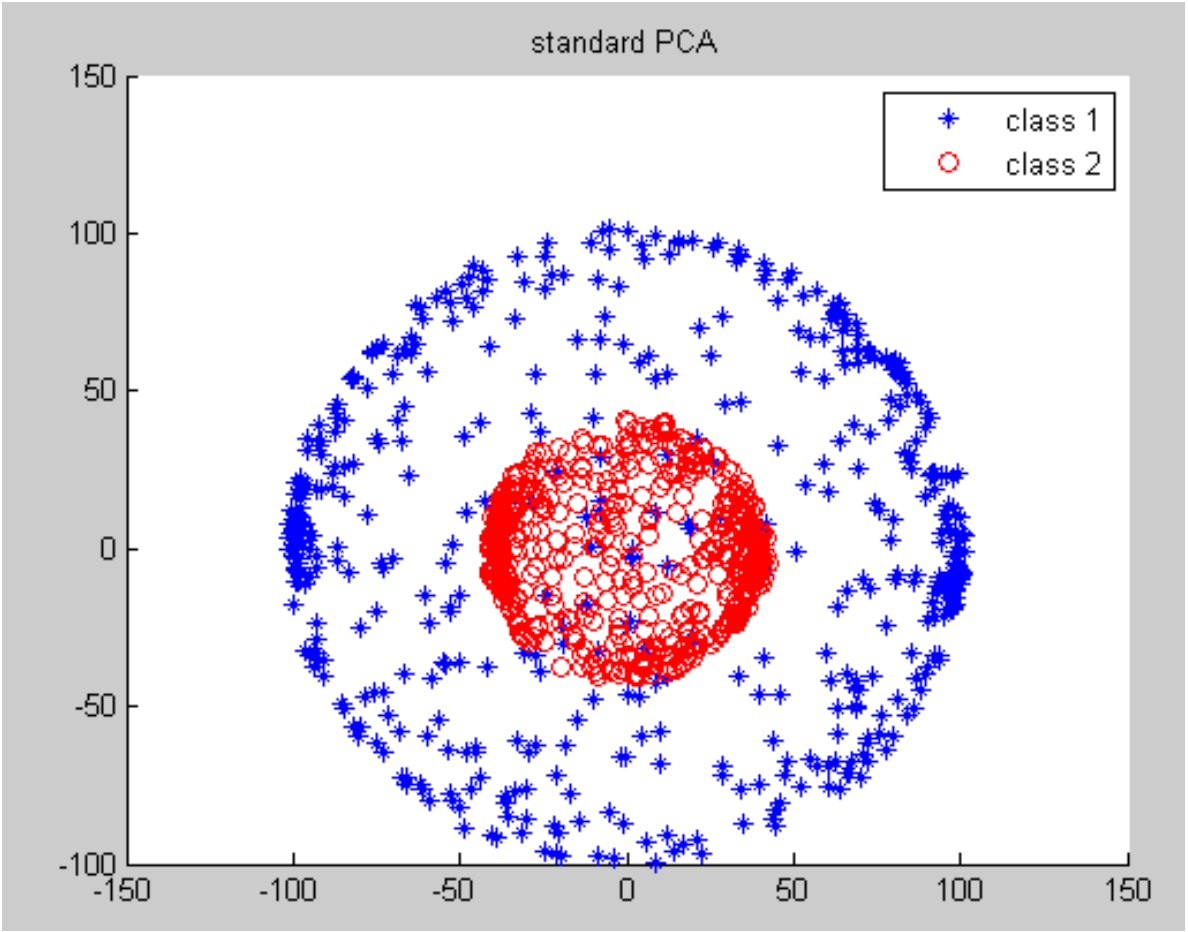
	1	2	3
1	-88.2783	-44.9607	7.0568
2	18.7660	12.1832	33.9861
3	85.6706	51.3761	-3.8543
4	2.8330	-1.9369	38.9479
5	24.5787	-5.8881	-95.7886
6	16.4186	34.3755	13.1516
7	2.1680	-2.9920	-99.4955
8	-16.8696	-11.3769	-34.3947
9	-84.0856	20.0495	-52.8139
10	-0.2452	0.8573	-39.3020
11	-2.7936	46.4088	-87.0521
12	-1.3023	38.2229	-1.4955
13	-4.2918	-0.4123	-98.7998
14	-16.8444	8.2598	-36.8473
15	85.4805	-17.6086	-48.4263
16	20.1288	-27.8644	21.7843
17	1.0819	-0.5867	-100.5959
18	15.7314	20.3911	29.5500
19	-8.1825	83.1212	54.0508
20	2.8475	-30.0695	24.6174
21	4.1690	-12.7177	97.3090
22	4.0096	-11.5281	-39.9645
23	39.0154	91.4269	-4.3496
24	27.8044	27.4301	-9.7474
25	-64.7173	-59.3353	-49.3511
26	-2.2166	-39.8347	1.6756
27	65.6895	-7.2869	-75.4613
28	-38.5712	-9.8804	3.5479
29	54.4305	-8.6069	83.8852
30	-3.9201	-10.1438	-38.8079
31	28.0910	63.9337	72.2697
32	16.3860	-36.3944	5.0192

	1	2	3
32	16.3860	-36.3944	5.0192
33	-100.6940	-6.4452	-1.0960
34	-28.5941	-26.7088	7.9973
35	-29.2244	89.7650	-32.8902
36	27.9354	27.9749	-2.8863
37	-11.5172	65.3030	75.4915
38	-31.6205	-25.1086	0.8250
39	-66.5808	-1.4838	-75.4468
40	5.8847	31.1994	23.7942
41	40.9660	-80.3497	43.7461
42	25.4686	-29.5762	9.2897
43	-5.3068	-23.7835	96.8680
44	-10.0345	38.9043	3.7991
45	-3.7578	10.0166	98.6781
46	1.9958	-5.7128	40.2225
47	16.5785	-57.6323	79.1247
48	28.1717	28.1596	-1.3051
49	12.3425	19.8227	-97.7523
50	38.8445	5.2272	-3.6825
51	-97.5163	-11.7493	-20.4343
52	-34.2409	18.8793	11.8706
53	89.7897	-16.2756	40.6405
54	-32.3305	10.0440	19.9357
55	42.4553	-84.7449	-34.1944
56	0.3350	-18.3883	35.6729
57	-5.8878	-7.3094	99.8485
58	-2.8967	-13.0541	-38.9086
59	-51.2399	68.8863	-50.2910
60	-16.8810	16.1630	-32.0438
61	-25.0192	-5.8718	-95.5164
62	7.9160	-3.4730	39.9748
63	-82.7399	-51.8764	-24.1630

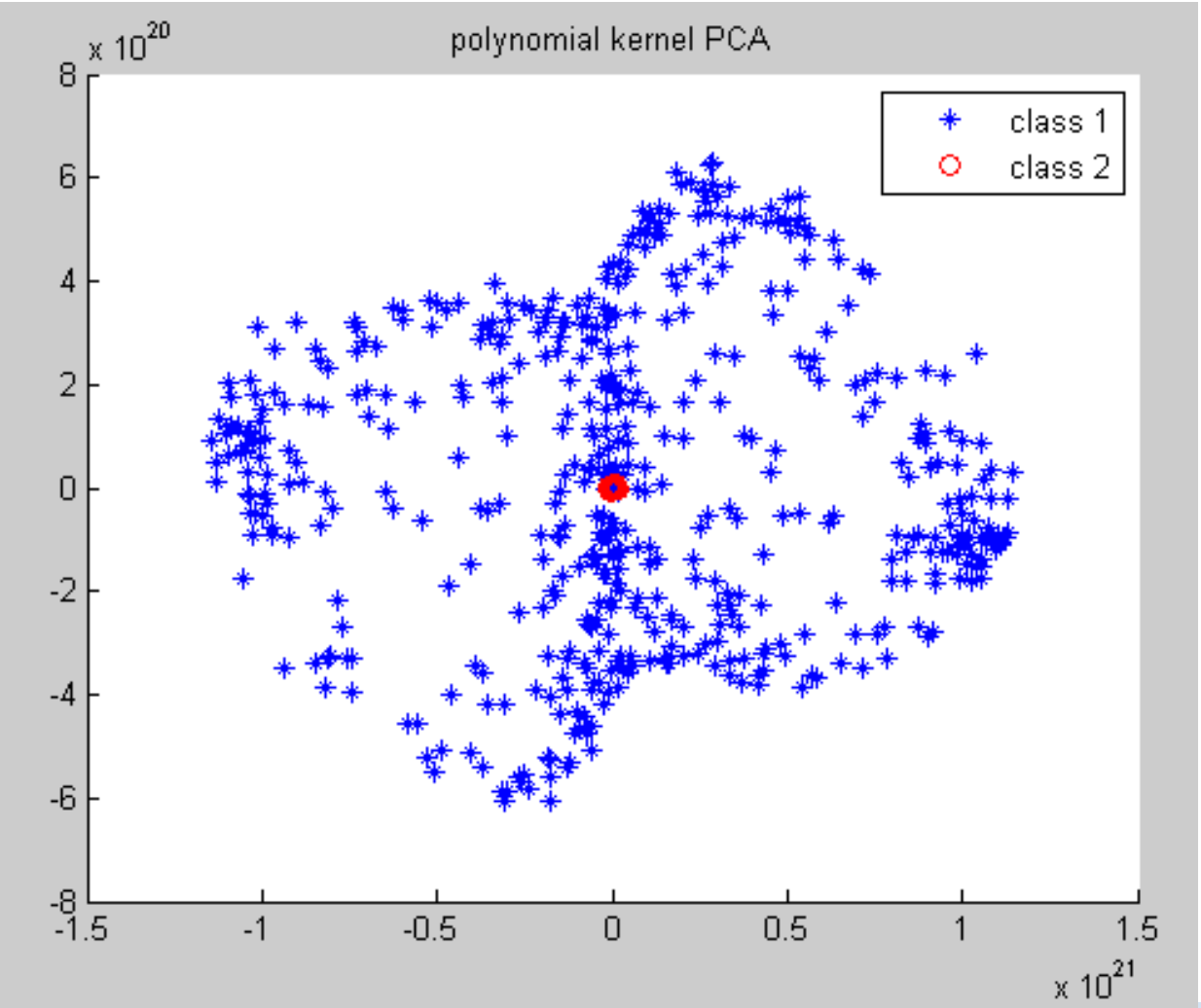
	1	2	3
64	33.7954	4.3451	-24.3365
65	-9.6023	-40.9501	-93.1028
66	-0.5545	33.6727	19.4823
67	-88.1515	37.2810	-27.5918
68	32.1920	6.7318	22.6564
69	50.1551	-77.5785	-36.1045
70	-13.1424	-14.0452	-34.9507
71	-20.3627	68.3021	72.3880
72	-11.3068	-34.1125	15.2281
73	27.6767	-12.1305	94.0013
74	8.1707	-20.6474	33.3189
75	-61.8277	-78.5864	-7.2388
76	11.6272	15.5509	36.9328
77	78.0188	-38.4314	-47.0786
78	-33.1136	22.4965	1.1917
79	56.7623	39.1851	-72.9892
80	1.8105	5.9147	-41.2094
81	-14.8765	-47.3224	-87.1255
82	-14.2931	14.6900	-33.5905
83	-22.3948	-1.5863	-96.3321
84	-0.1045	-0.6660	-40.3545
85	94.6064	-34.4596	-6.2075
86	-20.0563	32.1719	16.4020
87	-35.3545	20.5063	91.4720
88	4.4381	7.8252	39.9853
89	-39.1146	90.0463	-13.3213
90	32.8661	-2.4591	-24.8136
91	-4.0100	-58.9698	-78.8484
92	16.8515	9.3305	33.6024
93	-10.9411	35.8924	95.7406
94	-18.0028	-25.2695	-24.6476
95	-40.0947	-33.2454	83.8713

To visualize our results, we project the original 3-imensional data onto a 2-dimensional feature space by using both standard PCA and kernel PCA.

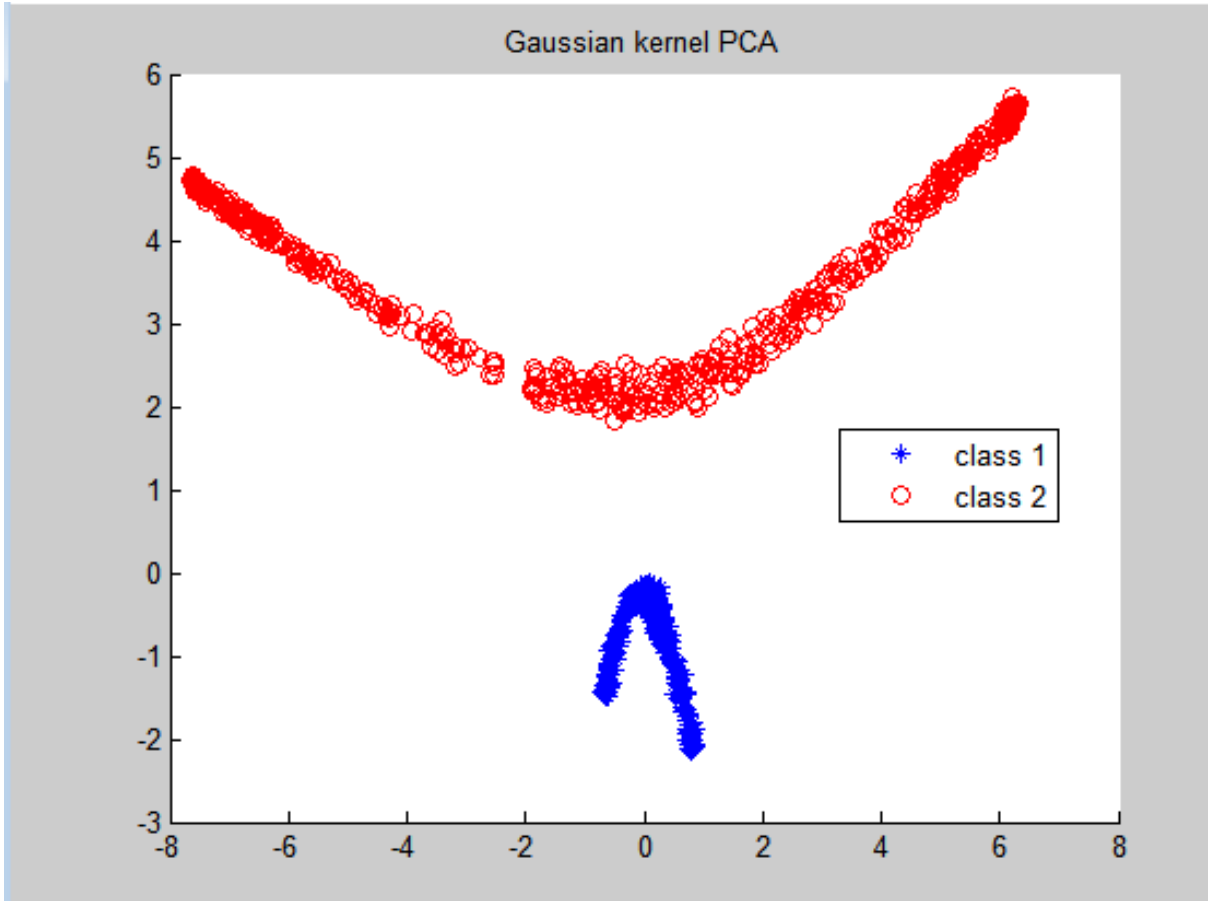
使用标准PCA将上述数据聚类的结果：



使用**polynomial kernel PCA** 将上述数据聚类的结果：



使用**Gaussian kernel PCA** 将上述数据聚类的结果：



In the resulting figures we can see that, standard PCA does not reveal any structural information of the original data.

For polynomial kernel PCA, in the new feature space, class 2 data points are clustered while class 1 data points are scattered. But they are still not linearly separable.

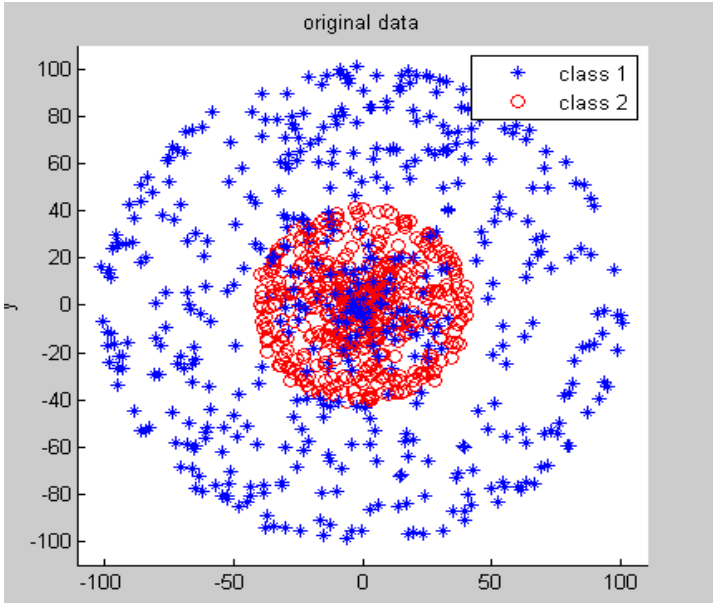
For Gaussian kernel PCA, the two classes are completely linearly separable.

实现上述可视化效果的matlab代码

原始数据可视化：

Data里1,3，…行是蓝色数据；2,4，…行是红色数据

```
%% original data
figure;hold on;
plot3(data(1:2:end,1),data(1:2:end,2),data(1:2:end,3),'b*');
plot3(data(2:2:end,1),data(2:2:end,2),data(2:2:end,3),'ro');
legend('class 1','class 2');
axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
axis([-110 110 -110 110 -110 110]);
title('original data');
drawnow;
```



标准PCA聚类：

eig
Eigenvalues and eigenvectors

```
%% standard PCA
disp('Performing standard PCA...');
Y1=PCA(data,d);
figure;hold on;
plot(Y1(1:2:end,1),Y1(1:2:end,2),'b*');
plot(Y1(2:2:end,1),Y1(2:2:end,2),'ro');
legend('class 1','class 2');
title('standard PCA');
drawnow;
```

```
function [Y, eigVector, eigValue]=PCA(X,d)

%% eigenvalue analysis
Sx=cov(X);
[V,D]=eig(Sx);
eigValue=diag(D);
[eigValue,IX]=sort(eigValue,'descend');
eigVector=V(:,IX);

%% normailization
norm_eigVector=sqrt(sum(eigVector.^2));
eigVector=eigVector./repmat(norm_eigVector,size(eigVector,1),1);

%% dimensionality reduction
eigVector=eigVector(:,1:d);
Y=X*eigVector;
```

polynomial kernel PCA聚类：

```
%% polynomial kernel PCA
para=5;
disp('Performing polynomial kernel PCA...');
[Y2]=kPCA(data,d,'poly',para);
figure;hold on;
plot(Y2(1:2:end,1),Y2(1:2:end,2),'b*');
plot(Y2(2:2:end,1),Y2(2:2:end,2),'ro');
legend('class 1','class 2');
title('polynomial kernel PCA');
drawnow;
```

Gaussian kernel PCA聚类：

```
%% Gaussian kernel PCA
DIST=distanceMatrix(data);
DIST(DIST==0)=inf;
DIST=min(DIST);
para=5*mean(DIST);
disp('Performing Gaussian kernel PCA...');
[Y3, eigVector]=kPCA(data,d,'gaussian',para);
figure;hold on;
plot(Y3(1:2:end,1),Y3(1:2:end,2),'b*');
plot(Y3(2:2:end,1),Y3(2:2:end,2),'ro');
legend('class 1','class 2');
title('Gaussian kernel PCA');
drawnow;
```

```
function [Y, eigVector, eigValue]=kPCA(X,d,type,para)

%% check input
if ( strcmp(type,'simple') || strcmp(type,'poly') || ...
        strcmp(type,'gaussian') ) == 0
    Y=[];
    eigVector=[];
    fprintf(['\nError: Kernel type ' type ' is not supported. \n']);
    return;
end

N=size(X,1);

%% kernel PCA
K0=kernel(X,type,para);
oneN=ones(N,N)/N;
K=K0-oneN*K0-K0*oneN+oneN*K0*oneN;

%% eigenvalue analysis
[V,D]=eig(K/N);
eigValue=diag(D);
[~,IX]=sort(eigValue,'descend');
eigVector=V(:,IX);
eigValue=eigValue(IX);

%% normailization
norm_eigVector=sqrt(sum(eigVector.^2));
eigVector=eigVector./repmat(norm_eigVector,size(eigVector,1),1);

%% dimensionality reduction
eigVector=eigVector(:,1:d);
Y=K0*eigVector;
```

```
X: data matrix, each row is one observation, each column is one feature
d: reduced dimension
type: type of kernel, can be 'simple', 'poly', or 'gaussian'
para: parameter for computing the 'poly' and 'gaussian' kernel,
    for 'simple' it will be ignored
Y: dimensionanlity-reduced data
eigVector: eigen-vector, will later be used for pre-image
    reconstruction
```

If the projected dataset $\{\phi(\mathbf{x}_i)\}$ does not have zero mean, we can use the Gram matrix $\widetilde{\mathbf{K}}$ to substitute the kernel matrix $\mathbf{K}$. The Gram matrix is given by

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N,$$

where $\mathbf{1}_N$ is the $N \times N$ matrix with all elements equal to $1/N$