Maths and MATLAB review
Supervised Learning 2004

Iain Murray and **Edward Snelson**

University College London, UK

http://www.gatsby.ucl.ac.uk/~fernando/SupLearn.html

Introduction

- Attached to your notes is a **"crib-sheet"**. Please go through it in your own time.
  - Some of you will know all or most of this
  - Experience shows some of you will not

- We will review some important material through simple examples coded in MATLAB
  - The point is to help you get started: all code is available on the course website.
  - With all demos think "what is wrong with this method?", "how could I make it fail?" and "how could it be improved?"

- **Do interrupt**

- **We will also go around and discuss anything you want in the break**

Introduction to MATLAB

Let's **not** go through lots of simple material (see right)

Type `demo` **within** MATLAB to be walked through lots of introductory material.

If you want a command to do something try `lookfor` to find it and then `help` for more information.

Advanced tips and tricks are (eg) here:

`http://www.ee.columbia.edu/~marios/matlab/matlab_tricks.html`

```
>> a=0.1

a =

    0.1000

>> x=[1.1; 2.3]

x =

    1.1000
    2.3000

>> x'

ans =

    1.1000    2.3000

>> A=[11 12 13 14; 21 22 23 24; 31 32 33 34]+a

A =

   11.1000   12.1000   13.1000   14.1000
   21.1000   22.1000   23.1000   24.1000
   31.1000   32.1000   33.1000   34.1000

>> B=A'*A;
>>
```

Let me emphasise one thing

Check the sizes of everything in your maths and your code

As you (will) know very well inner dimensions of quantities must match when you multiply.

If A is $l \times m$ and B is $m \times n$, where l, m and n are different. Then:

Maths	MATLAB	Dimension
AB	`A*B`	$l \times n$
$A^\top A$	`A'*A`	$m \times m$
$A^\top AB$	`A'*A*B`	$m \times n$
$A^\top B$	`A'*B`	Meaningless
BB	`B*B`	Meaningless
$A^\top AB^\top$	`A'*A*B'`	Meaningless

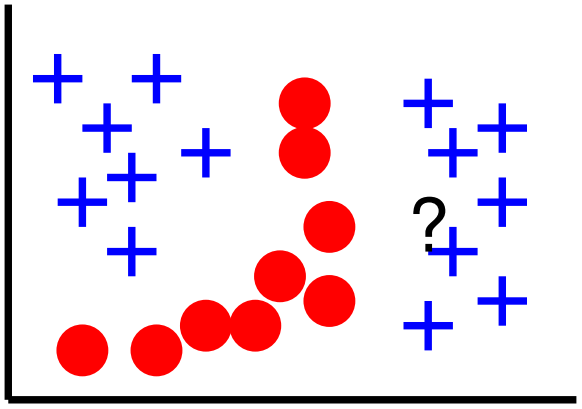
MATLAB gives errors when dimensions don't match.

Type `whos` to give the dimensions of all your variables.

Use `size` to get the dimensions within your code.

First example — Nearest Neighbour Classifier

Classify the question mark



There are many ways you could do this.

One approach is to **pick the nearest neighbour** and **steal its label**.

Distance to neighbour $i = \sqrt{\sum_{d=1}^{D} \left(x_d^{(i)} - x_d^{(?)} \right)^2}$ (simple Pythagoras)

```
% Assume we have loaded NxD data set Y, Nx1 labels L
% and 1xD test point X
[N,D]=size(Y);
Ydisp=Y-repmat(X,N,1);
Ydist2=sum(Ydisp.^2,2);
[mindist2,j]=min(Ydist2);
% Now Y(j,:) is our nearest neighbour; we choose label L(j)
```

Probabilities — Text classification example

Make sure you are familiar with all of the basic rules on the crib sheet

Assume we have models for generating text t for languages English $L = E$, and Welsh $L = W$. That is, $P(t|e)$ and $P(t|w)$ are known.

We want to know if documents are English or Welsh: $P(e|t)$ and $P(w|t)$. We will assume they are one of the two.

Bayes rule twice:

$$P(e|t) = \frac{P(t|e)P(e)}{P(t)} \propto P(t|e)P(e)$$

$$P(w|t) = \frac{P(t|w)P(w)}{P(t)} \propto P(t|w)P(w)$$

Assumption:

$$P(e|t)+P(w|t) = 1 \quad \Rightarrow \quad P(e|t) = \frac{1}{1 + \frac{P(w|t)}{P(e|t)}}$$

Plug in Bayes results:

$$\therefore \quad P(e|t) = \frac{1}{1 + \frac{P(t|w)P(w)}{P(t|e)P(e)}}$$

Probabilities — Text classification example

In class we will go through a specific example with code online.

Integration

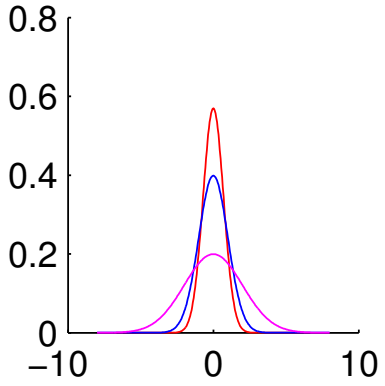
Probabilistic manipulations often use integrals. These are sometimes performed in high-dimensional spaces.

Be comfortable with what the following mean:

$$\int_{\text{Line}} \mathrm{d}x \; f(x) \quad \text{as well as} \quad \int_{\text{Volume}} \mathrm{d}\mathbf{x} \; f(\mathbf{x})$$

We will tackle how to approximate hard integrals for machine learning later in the course.

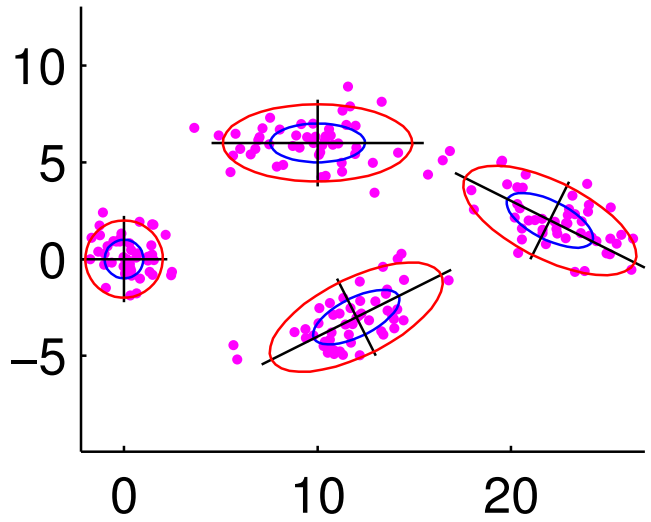
Multivariate Gaussians



$$p\left(x|\sigma^2,\mu\right)=\mathcal{N}(\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}\ \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$p\left(\mathbf{x}|\Sigma,\mu\right)=\mathcal{N}(\mu,\Sigma)=|2\pi\Sigma|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

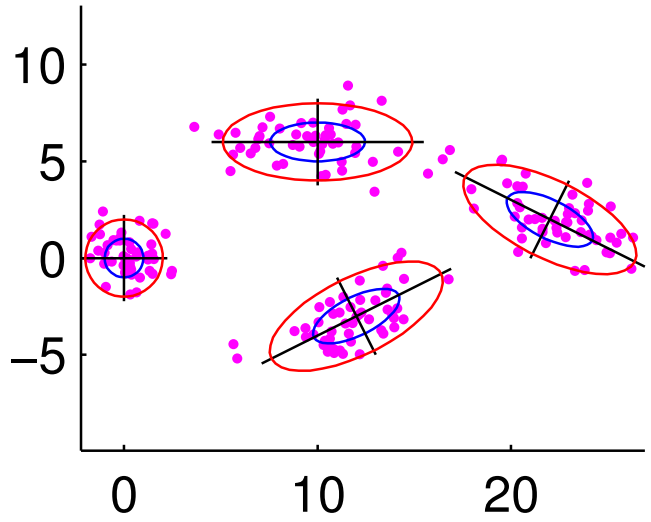
```
% plotgauss.m is not part of Matlab
% Available on the course website
mu=zeros(2,1); % Mean
S=eye(2);      % Covariance matrix
plotgauss(mu,S);

plotgauss([10;6],[6 0;0 1]);
plotgauss([12;-3],[5 2;2 2]);
plotgauss([22;2],[5 -2;-2 2]);
```



Eigenvectors and Eigenvalues

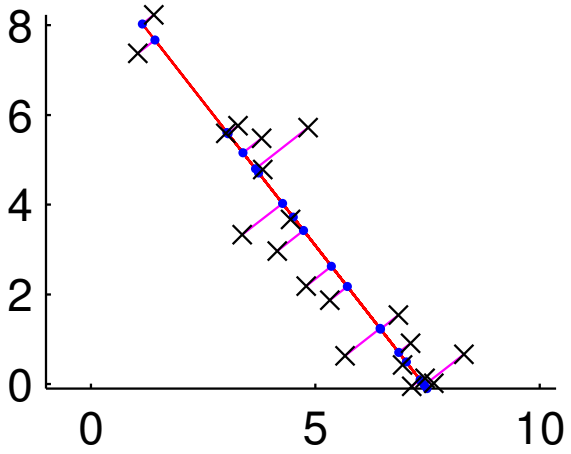
$$B\mathbf{e}^{(i)} = \lambda^{(i)}\mathbf{e}^{(i)} \quad \Leftrightarrow \quad \text{“}\lambda^{(i)} \text{ is an eigenvalue of } B \text{ with eigenvector } \mathbf{e}^{(i)}\text{”}$$

- eigenvectors of covariances are perpendicular
- They lie along the axes of the ellipse characterising the covariance
- The larger the eigenvalues the more pointy the ellipse in that direction



PCA — Principal Components Analysis

Really an *un*supervised learning task, dimensionality reduction using PCA can be a useful preprocessing step.



K = 1

× = X

· = Xproj

— = V(:,1)

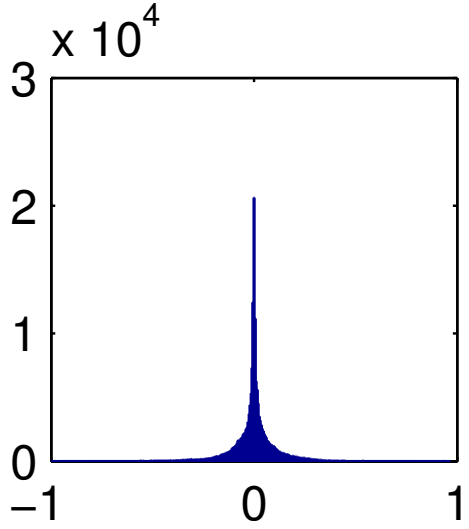
```
function [Xproj,meanX,V,S] = PCA(X,K)

N = size(X,1);
% Calculate mean and covariance matrix
meanX = mean(X);
covX = cov(X,1);
% SVD to calculate K principal eigenvectors
[V,S] = svds(covX,K);
% Project X onto K principal eigenvectors
Xproj = (X - repmat(meanX,N,1))*V*V' + ...
        repmat(meanX,N,1);
```

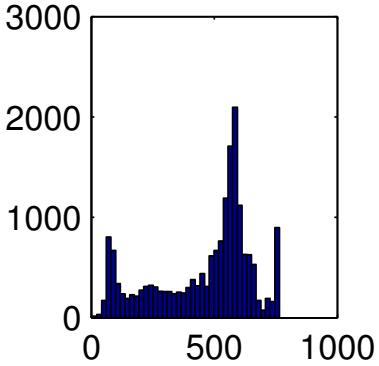
Fun with PCA — eigenfaces

$D = 32 \times 32 = 1024$, `K=1:30`

Gaussians are not the only fruit

```
x=importdata('KDE_Startup.wav');
hist(x.data,400);
```






```
% Note must convert to double
% for some operations
x=double(importdata('hpucl.jpg'));
x=sum(x,3); % add up colors
hist(reshape(x,1,prod(size(x))),40);
```

Gaussians **are** very useful, but do be careful about your assumptions.

Differentiation

- During the course we will often need to maximize or minimize some cost function, in order to find 'optimal' parameter values

- You will need to be comfortable with **multi-variate differentiation**. In particular, knowledge of how take derivatives with respect to vectors and matrices, will be required

- **Key point** - if in doubt, always write out indices explicitly before taking derivatives

- With experience some short-cuts can be taken

Case sudy: Linear regression

- N input/target pairs $\{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$; $\mathbf{x}^{(n)}$ and $\mathbf{y}^{(n)}$ are J and K dimensional vectors
- Aim: minimize $\mathcal{L} = \sum_n \left(\mathbf{y}^{(n)} - W\mathbf{x}^{(n)} - \mathbf{b}\right)^2$ w.r.t. matrix W and vector $\mathbf{b}$.
- In MATLAB it is often best to construct data matrices: $X_{jn} = \mathbf{x}_j^{(n)}$ and $Y_{kn} = \mathbf{y}_k^{(n)}$
- First we consider the long-winded, but safe, approach of writing out all indices explicitly ...

Linear regression

$$\mathcal{L} = \sum_{nk} \left(Y_{kn} - \sum_j W_{kj} X_{jn} - b_k \right)^2$$

Take partial derivative w.r.t. component of vector $\mathbf{b}$, and set to zero:

$$\frac{\partial \mathcal{L}}{\partial b_k} = -2 \sum_n \left(Y_{kn} - \sum_j W_{kj} X_{jn} - b_k \right) = 0$$

$$\Rightarrow \qquad b_k = \tfrac{1}{N} \sum_n Y_{kn} - \sum_j W_{kj} \, \tfrac{1}{N} \sum_n X_{jn}$$

```
b = mean(Y,2) - W*mean(X,2);
```

Linear regression

Taking derivative w.r.t. entry of matrix W leads to the following equations, which are also shown in MATLAB below:

$$\langle X, X \rangle_{ij} = \frac{1}{N} \sum_n X_{in} X_{jn} - \frac{1}{N} \sum_n X_{in} \frac{1}{N} \sum_n X_{jn}$$

$$\langle Y, X \rangle_{kj} = \frac{1}{N} \sum_n Y_{kn} X_{jn} - \frac{1}{N} \sum_n Y_{kn} \frac{1}{N} \sum_n X_{jn}$$

$$W = \langle Y, X \rangle \langle X, X \rangle^{-1}$$

```
covX = X*X'/N - mean(X,2)*mean(X,2)';
covYX = Y*X'/N - mean(Y,2)*mean(X,2)';
W = covYX/covX;
```

Linear regression

- With experience, it is possible to take derivatives w.r.t. vectors and matrices without writing out the indices. For example:

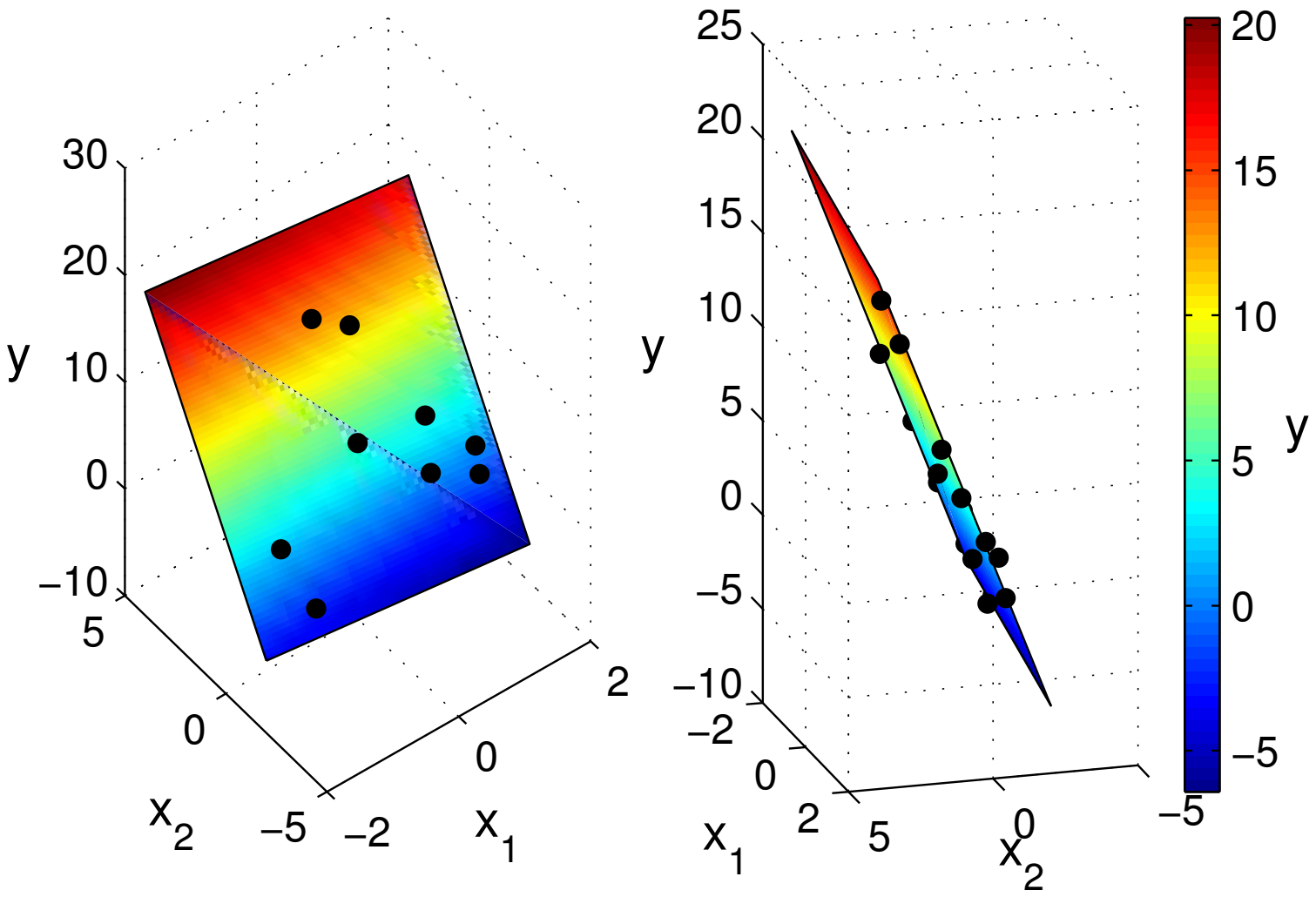
$$\mathcal{L} = \sum_n \left(\mathbf{y}^{(n)} - W\mathbf{x}^{(n)} - \mathbf{b} \right)^2$$

$$\frac{\partial \mathcal{L}}{\partial W} = -2 \sum_n \left(\mathbf{y}^{(n)} - W\mathbf{x}^{(n)} - \mathbf{b} \right) {\mathbf{x}^{(n)}}^\top$$

- We arrive at the correct answer by making sure the matrix dimensionality of each side of the equation matches

- Sam Roweis has a cribsheet which lists a set of rules for performing this type of derivative: `http://www.cs.toronto.edu/~roweis/notes/matrixid.pdf`

Linear regression demo



- Code to produce this figure available online

End of Part I

- That's it from us. Fernando will review optimization next
- **Ask us anything you like during the break**