Chapter 4

Statistics

In this course, we will be examining large data sets coming from different sources. There are two basic types of data:

- Deterministic Data: Data coming from a specific function- each point has a unique image. If we know the function, we can predict exactly the data terms that we see.

- Probabilistic Data: This data cannot be predicted exactly; with similar inputs, we might find very different outputs. The data is *random* in some sense. In this case, we can only hope to model some characteristics of the data, and not the exact data points themselves. We saw this earlier in the $N-$armed bandit problem, where we saw, with each pull of the arm, we would get different outputs. Thus, we were interested in extracting general information about the processes- the expected values, or mean payoff of each machine.

In data analysis, most data falls in between the two classes- that is, data from most sources involves a deterministic part, and a random part (perhaps from measurement errors).

We will not do a comprehensive listing of statistical procedures here. Instead, we introduce this material so that we have some statistical language to work with later on.

4.1 Measuring Variability

Some data sets naturally have variation- For example, suppose I measure the temperature at the airport in Walla Walla on January 29th of each year. Here is the data for the maximum temperature recorded on January 29th for the years

shown:

2008	39	2003	51	1998	46	1993	41	1988	54
2007	30	2002	34	1997	30	1992	59	1987	53
2006	55	2001	45	1996	18	1991	39	1986	36
2005	50	2000	28	1995	61	1990	48	1985	32
2004	57	1999	55	1994	43	1989	55	1984	52

Given the historical record, we might try to predict the temperature for next January 29th. What might our answer be?

- The maximum is 61, the minimum is 18 and the average is 44.44 degrees, so we might guess that the temperature should be approximately 44.44 degrees.
- We could make bins out of the data and look at the frequency of temperatures in each bin. For example, suppose we take 5 equally spaced intervals between 18 and 61. In each of these intervals, we count how many temperatures are present (you should verify a couple of them!):

Interval	Count (out of 25)	Percentage
$18 - 26.6$	2	8
$26.6 - 35.2$	4	16
$35.2 - 43.8$	5	20
$43.8 - 52.4$	6	24
$52.4 - 61$	8	32

  We might now say that 44.4 degrees was probably too cold- It looks like 32 percent of the time, the temperature on January 29th was between 52.4 and 61 degrees, and this "bin" is the most likely.

We could continue the process of "binning" if we had a lot of data- In Figure 4.1 we use a computer simulation using random data to see what happens as we add more and more bins (or intervals). On the vertical axis, we are measuring the frequency rather than the percentage. By adding more data and more subintervals, it looks like the probabilities are approaching a continuous function. This function is what we define next:

4.2 Functions that Define Data

The basic way of defining non-deterministic data is through the use of a *probability density function*[1], or p.d.f. Rather than defining a function in terms of inputs and outputs, a p.d.f. defines the probability of certain events occurring.

To be more specific, a function $f(x)$ is said to be a probability density function if it satisfies the following conditions:

[1]Some texts use different vocabulary for discrete v. continuous data- We will not, as it should be clear from the context

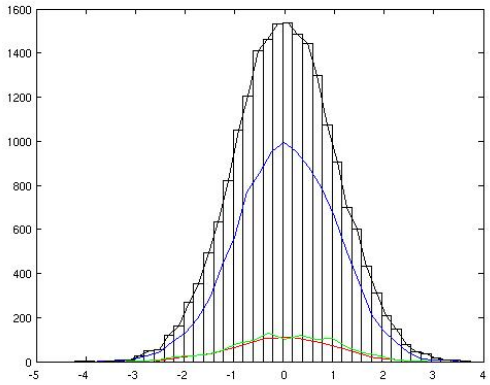

Figure 4.1: The result of adding more and more data from a computer simulation. The vertical axis is the frequency count for each bin. We see that the histogram is beginning to approximate a continuous function.

1. $f(x)$ is always non-negative.
2. $\int_{-\infty}^{\infty} f(x)\,dx = 1$
3. The probability of an event between values $x = a$ and $x = b$ is given by:

$$\Pr(a \leq X \leq b) = \int_a^b f(x)\,dx$$

From the definition, we see that the probability of any specific number is zero. Furthermore, in practice we won't be dealing with continuous functions (although we might try modeling them); rather, we will be looking at discrete intervals.

Therefore, suppose that our function f is non-zero on a finite interval (another way to say this is that f has bounded support). Then we can break the interval (or intervals) up into subintervals, and consider the probability of data occurring in each of these. In this case, the p.d.f. is discrete, and our definition changes somewhat:

A discrete p.d.f. will be a finite set of numbers, $\{P_1, P_2, \ldots, P_k\}$, so that:

1. P_i is non-negative, for all i.
2. $\sum_{i=1}^{k} P_k = 1$
3. The probability of a data value occurring in subinterval i (or bin i) is P_i.

Note that in this case, each of our "events" (data in subinterval i) are disjoint, as the probability of landing on an endpoint is zero. In the $N-$armed bandit

problem, we had some experience with these- in that case, $P_i = \pi(i)$, which was the probability of choosing machine i.

In Matlab, we visualize a PDF via the `hist` command, which produces a histogram of the input. Let's take a look at some template probability distributions:

- **Example 1: The Uniform Distribution**
  - The Continuous Version:
$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } a \le x \le b \\ 0, & \text{otherwise} \end{cases}$$
  - The Discrete Version (using N bins over the same interval):
$$\Pr\left(a + (i-1)\frac{b-a}{N} \le x \le a + i\frac{b-a}{N}\right) = \frac{1}{N} = P_i, i = 1, 2, \ldots, N.$$
  - In Matlab, to obtain an $m \times n$ array of numbers from a uniform distribution over $[0, 1]$, we type $x =$ `rand(m,n)` or, for a single number: $x =$ `rand`. Here is an example you should try. In this case, we use 5000 data points and 20 intervals (try changing them around):

```
x=rand(5000,1);
hist(x,20);
```

- **Example 2: The Normal (or Gaussian) Distribution**
  - The Continuous Version: The Normal distribution with mean μ and variance σ^2 (to be defined shortly) is defined as:
$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(\frac{-(x-\mu)^2}{\sigma^2}\right)$$
    This is the common "bell-shaped curve"; the constant in the front is needed to make the integral evaluate to 1. Note that the normal distribution with zero mean and variance 1 simplifies to:
$$f(x) = \mathcal{N}(0,1) = \frac{1}{\sqrt{2\pi}}\,\mathrm{e}^{-x^2}$$
  - In Matlab, we can obtain values from a normal distribution with zero mean and unit variance by $x =$ `randn(m,n)` or $x =$ `randn`. Try the same commands as before:

```
x=randn(5000,1);
hist(x,20);
```

    To change the mean, just add a constant. To change the standard deviation, just multiply:

```
x=randn(5000,1);
y=5*x+8;
hist(y,20);
mean(y)
std(y)
```

- **Example 3:** The p.d.f. commonly used to model the human voice is called the double Laplacian distribution:

$$f(x) = \begin{cases} K\mathrm{e}^{-|x|}, & -1 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

In the exercises, you'll be asked to determine the value of K, and we'll also see how the shape of the Laplacian compares to a normal distribution. In the meantime, you can visualize this in Matlab- The file `laughter` is built into Matlab:

```
load laughter
sound(y,Fs);  %Plays the file over the speaker, if available
hist(y,50)
```

4.2.1 The cumulative distribution function (cdf)

The cumulative distribution function (a.k.a. distribution function) is defined via the pdf:

$$F(X) = \Pr(-\infty < X < x) = \int_{-\infty}^{x} f(t)\,dt$$

We note that:

- By the Fundamental Theorem of Calculus, part I, $F(x)$ is the antiderivative of $f(x)$.
- $F(x)$ is strictly increasing, going from a minimum of zero to a maximum of 1.
- We saw the discrete version of this when we were working with the $N-$armed bandit; we used the Matlab function `cumsum` to create the cumulative distribution.
- To minimize confusion between terms, we'll refer to $F(x)$ as the cdf.

4.3 Measuring the Central Tendency of Data

The most basic way to characterize a data set is through one number- the average, which can be defined a number of ways. Let us assume we have some data: $\{x_1, x_2, \ldots, x_m\}$:

- The *Sample Mean* is:

$$\overline{x} = \frac{1}{m}\sum_{k=1}^{m} x_k$$

We're all familiar with this definition- it's just the average of the data points.

For those who have had a course in statistics, you will recall that this is different than the theoretical *population mean*, which requires knowledge of the underlying p.d.f., which we seldom have. In that case, the definition (a.k.a. the Expected value) is given by:

$$\mu = E(X) = \sum_{\text{all x}} xf(x)$$

You might remember this formula by seeing that this is really a weighted average, with those events most probable getting a higher weight than events less probable.

Also, if your data is being drawn independently from a fixed p.d.f., then the sample mean will converge to the population mean, as the number of samples gets very large.

Suppose we have m vectors in $\mathbb{R}^n$. we can similarly define the mean, just replace the scalar x_k with the k^{th} vector:

$$\overline{\boldsymbol{x}} = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{x}^{(k)}$$

In Matlab, the mean is a built-in function. The command is `mean`, and the output depends on whether you input a vector or a matrix of data:

  - For vectors, `mean(x)` outputs a scalar.
  - For an $m \times n$ matrix X,
    - `mean(X)`, the default, is the same as `mean(X,1)`
    - `mean(X,1);` returns a row of n elements
    - `mean(X,2);` returns a column of m elements
    - **The Matrix Grand Mean** is the mean of all the data in the matrix (so the output is one number).
      Exercise: Show, algebraically, that the result of `mean(mean(X))` will be the same as taking the average of all the matrix elements.

We will look more closely at these commands below, but first we finish a few definitions:

- The *Sample Median* is a number so that exactly half the data is above that number, and half the data is below that number. Although the median

does not have to be unique, we follow the definitions below if we are given a finite sample:

If there are an odd number of data points, the median is the middle point. If there is an even number of data points, then there are two numbers in the middle- the median is the average of these.

Although not used terribly often, Matlab will perform the median as well as the mean: `median(X);`

- The *Mode* rarely if ever comes into use as a measure of central tendency, but it is the value taken the most number of times. In the case of ties, the data is said to be multi-modal.

4.3.1 Some Matlab:

There are times when we want to define an array as a block of numbers, then build a new array out of it. For example a matrix A is given and two different ways of building it in Matlab follow:

$$b = \begin{bmatrix} 1.3 \\ 0.1 \\ 2.1 \\ -1.0 \end{bmatrix} \qquad A = \begin{bmatrix} 1.3 & 1.3 & 1.3 & 1.3 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 2.1 & 2.1 & 2.1 & 2.1 \\ -1.0 & -1.0 & -1.0 & -1.0 \end{bmatrix}$$

$\texttt{A} = \texttt{repmat}(\texttt{b}, 1, 4)$ $\qquad$ $\texttt{A} = \texttt{b} * \texttt{ones}(1, 4);$

Example: Build an array A by repeating the sub-array given in B three times down and two times across (so that the array A is 6×6).

$$B = \begin{bmatrix} 1 & 0 & 2 \\ 0 & -1 & 1 \end{bmatrix}$$

SOLUTION (Check in Matlab):

```
B=[1 0 2; 0 -1 1];
A=repmat(B,3,2)
```

4.3.2 Centering and Double Centering Data

Let matrix A be $n \times m$, which may be considered n points in R^m or m points in $\mathbb{R}^n$. If we wish to look at A both ways, a double-centering may be appropriate.

The result of the double-centering will be that (in Matlab), we determine $\hat{A}$ so that

$$\texttt{mean}(\hat{A}, 1) = 0, \qquad \texttt{mean}(\hat{A}, 2) = 0$$

The algorithm is (in Matlab):

```
%Let A be n times m
[n,m]=size(A);
rowmean=mean(A);
A1=A-repmat(rowm,n,1);
colmean=mean(A1,2);
Ahat=A1-repmat(colmean,1,m);
```

or, equivalently:

```
%Let A be n times m
[n,m]=size(A);
colmean=mean(A,2);
A1=A-repmat(colmean,1,m);
rowmean=mean(A1,1);
Ahat=A1-repmat(rowmean,n,1);
```

Proof: For the first version (row mean first):

Let A_1 be the matrix A with the row mean **b** subtracted:

$$A_1 = \begin{bmatrix} a_{11}-b_1 & a_{12}-b_2 & \cdots & a_{1m}-b_m \\ a_{21}-b_1 & a_{22}-b_2 & \cdots & a_{2m}-b_m \\ \vdots & & & \vdots \\ a_{n1}-b_1 & a_{n2}-b_2 & \cdots & a_{nm}-b_m \end{bmatrix}$$

with

$$b_j = \frac{1}{n}\sum_{i=1}^{n} a_{ij}$$

Now define **c** as the column mean of A_1. Mean subtraction of this column results in the $\hat{A}$, written explicitly as:

$$\hat{A} = \begin{bmatrix} a_{11}-b_1-c_1 & a_{12}-b_2-c_1 & \cdots & a_{1m}-b_m-c_1 \\ a_{21}-b_1-c_2 & a_{22}-b_2-c_2 & \cdots & a_{2m}-b_m-c_2 \\ \vdots & & & \vdots \\ a_{n1}-b_1-c_n & a_{n2}-b_2-c_n & \cdots & a_{nm}-b_m-c_n \end{bmatrix}$$

By definition, the column mean of $\hat{A}$ is zero. Is the new row mean zero? It is clear that the new row mean is zero iff $\sum_k c_k = 0$, which we now show:

Proof that $\sum_{k=1}^{n} c_k = 0$

We explicitly write down what c_k is:

$$c_k = \frac{1}{m}\sum_{j=1}^{m}(a_{kj} - b_j)$$

and substitute the expression for b_j,

$$c_k = \frac{1}{m}\sum_{j=1}^{m}\left(a_{kj} - \frac{1}{n}\sum_{i=1}^{n} a_{ij}\right) = \frac{1}{m}\sum_{j=1}^{m} a_{kj} - \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} a_{ij}$$

Now sum over k:

$$\sum_{k=1}^{n} c_k = \sum_{k=1}^{n} \left(\frac{1}{m} \sum_{j=1}^{m} a_{kj} - \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} a_{ij} \right) =$$

$$\frac{1}{m} \sum_{k=1}^{n} \sum_{j=1}^{m} a_{kj} - \frac{n}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} a_{ij} = 0$$

It may be clear that these two methods produce the same result (e.g., row subtract first, then column subtract or vice-versa). If we examine the (i, j)th entry of $\hat{A}$,

$$\hat{A}_{ij} = a_{ij} - b_j - c_i = a_{ij} - \frac{1}{n} \sum_{k=1}^{n} a_{kj} - \frac{1}{m} \sum_{k=1}^{m} a_{ik} + \sum_{r=1}^{m} \sum_{s=1}^{n} a_{rs}$$

Therefore, to double center a matrix of data, each element has subtracted from it its corresponding row mean and column mean, and we add back the average of all the elements.

As a final note, this technique is only suitable if it is reasonable that the $m \times n$ matrix may be data in either $\mathbb{R}^n$ or $\mathbb{R}^m$. For example, you probably would not double center a data matrix that is 5000×2 unless there is a specific reason to do so.

Exercise: Experimentally verify the results of this section by performing (in Matlab) three ways of double-centering the data on a random 6×8 matrix A (it should not already have mean zero in either columns or rows- you should check that first).

- Subtract the row mean of A, then compute and subtract the column mean.
- Subtract the column mean of A, then compute and subtract the row mean.
- Compute the row mean and column mean and overall mean of the matrix A. Subtract row means, then column means, then add in the overall mean.

4.4 Measuring Spread- The Variance

The number that is used to describe the spread of the data about its mean is the *variance*:

Let $\{x_1, \ldots, x_m\}$ be as defined above. Then the *Sample Variance* is:

$$s^2 = \frac{1}{m-1} \sum_{k=1}^{m} (x_k - \overline{x})^2$$

where $\overline{x}$ is the mean of the data. If we think of the data as having zero mean, and placing each data point in a vector of length m, then this formula becomes:

$$s^2 = \frac{1}{m-1} \|\boldsymbol{x}\|^2$$

Why does $m - 1$ appear in the fraction?

- People who have had a course in statistics know that this makes s^2 an "unbiased estimator" of the population variance.

- In these notes, the actual variance will not be important- rather, we will be interested in comparing variances across data sets in which case the constant used for normalizing will not matter.

While we're defining the terms, we should distinguish between the sample variance and the actual population variance (as we did in the mean). The population variance is defined as the expected value of $(x-\mu)^2$,

$$E((x-\mu)^2) = \sum_{\text{all x}} (x-\mu)^2 f(x)$$

However, we seldom know the population variance in practice, so we will only use the sample variance.

The *Standard Deviation* is the square root of the variance.

Let's take some template data to look at what the variance (and standard deviation) measure: Consider the data:

$$-\frac{2}{n}, -\frac{1}{n}, 0, \frac{1}{n}, \frac{2}{n}$$

If n is large, our data is tightly packed together about the mean, 0. If n is small, the data are spread out. The sample variance is:

$$s^2 = \frac{1}{4}\left(\frac{4+1+0+1+4}{n^2}\right) = \frac{5}{2n^2}$$

and this is in agreement with our heuristic: If n is large, our data is tightly packed about the mean, and the standard deviation is small. If n is small, our data is loosely distributed about the mean, and the standard deviation is large. Another way to look at the standard deviation is in linear algebra terms: If the data is put into a vector of length m (call it $\boldsymbol{x}$), then the standard deviation is:

$$s = \frac{\|\boldsymbol{x}\|}{\sqrt{m-1}}$$

4.4.1 Covariance and Correlation Coefficients

If we have two data sets, sometimes we would like to compare them to see how they relate to each other.

Definition: Let $X = \{x_1, \ldots, x_m\}, Y = \{y_1, \ldots, y_m\}$ be two data sets with means μ_x, μ_y respectively. Then the *covariance* of the data sets is given by:

$$\text{Cov}(X,Y) = s_{xy}^2 = \frac{1}{m}\sum_{k=1}^{m}(x_k-\mu_x)(y_k-\mu_y)$$

There are exercises at the end of the chapter that will reinforce the notation and give you some methods for manipulating the covariance. In the meantime, it is easy to remember this formula if you think of the following:

If X and Y have mean zero, and we think of X and Y as vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, then the covariance is just the dot product between the vectors, divided by m:

$$\text{Cov}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{m}\boldsymbol{x}^T\boldsymbol{y}$$

We can then interpret what it means for X, Y to have a covariance of zero: $\boldsymbol{x}$ is orthogonal to $\boldsymbol{y}$. Continuing with this analogy, if we normalized by the size of $\boldsymbol{x}$ and the size of $\boldsymbol{y}$, we'd get the cosine of the angle between them. This is the definition of the correlation coefficient, and gives the relationship between the covariance and correlation coefficient:

Definition: The *correlation coefficient* between x and y is given by:

$$\rho_{xy} = \frac{\sigma_{xy}^2}{\sigma_x\sigma_y} = \frac{\sum_{k=1}^m (x_k - \mu_x)(y_k - \mu_y)}{\sqrt{\sum_{k=1}^m (x_k - \mu_x)^2 \cdot \sum_{k=1}^m (y_k - \mu_y)^2}}$$

Again, thinking of X, Y as having zero mean and placing the data in vectors $\boldsymbol{x}, \boldsymbol{y}$, then this formula becomes:

$$\rho_{xy} = \frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|} = \cos(\theta)$$

This works out so nicely because we have a $\frac{1}{m}$ in both the numerator and denominator, so they cancel each other out.

We also see immediately that ρ_{xy} can only take on the real numbers between -1 and 1. Some interesting values of ρ_{xy}:

If ρ_{xy} is:	Then the data is:
1	Perfectly correlated ($\theta = 0$)
0	Uncorrelated ($\theta = \frac{\pi}{2}$)
-1	Perfectly (negatively) correlated ($\theta = \pi$)

One last comment before we leave this section: The Covariance and Correlation Coefficient only look for *linear* relationships between data sets!

For example, we know that $\sin(x)$ and $\cos(x)$ (as functions, or as data points sampled at equally spaced intervals) will be uncorrelated, but, because $\sin^2(x) + \cos^2(x) = 1$, we see that $\sin^2(x)$ and $\cos^2(x)$ are perfectly correlated.

This difference is the difference between the words "correlated" and "statistically independent". Statistical independence (not defined here) and correlations are not the same thing! We will look at this difference closely in a later section.

4.5 The Covariance Matrix

If we have p data points in $\mathbb{R}^n$, we can think of the data as a $p \times n$ matrix. Let X denote the *mean-subtracted* data matrix (as we defined previously). A natural question to ask is then how the i^{th} and j^{th} dimensions (columns) covary- so we'll compute the covariance between the i, j columns to define:

$$\sigma_{ij}^2 = \frac{1}{p} \sum_{k=1}^{p} X(k,i) \cdot X(k,j)$$

Computing this for all i, j will result in an $n \times n$ symmetric matrix, C, for which:

$$C_{ij} = \sigma_{ij}^2$$

In the exercises, we have you show that we can conclude that C can be computed using the definition below:

Definition: Let X denote a matrix of data, so that, if X is $p \times n$, then we have p data points in $\mathbb{R}^n$. Furthermore, we assume that the data in X has been mean subtracted. Then the *covariance matrix* associated with X is given by:

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

In Matlab, it is easy to compute the covariance matrix. For your convenience, we repeat the mean-subtraction routine here:

```
%X is a pxn matrix of data:
[p,n]=size(X);
m = mean(X);
Xm = X-repmat(m,p,1);
C=(1/p)*X'*X;
```

Matlab also has a built-in covariance function. It will automatically do the mean-subtraction (which is a lot of extra work if you've already done it!).

```
C=cov(X);
```

If you forget which sizes Matlab uses, you might want to just compute the covariance yourself. It assumes, as we did, that the matrix is $p \times n$, and returns an $n \times n$ covariance- HOWEVER, it will divide by $p - 1$ rather than by p. This is not a big issue for the applications we will be considering- you may use either method, but be aware that there are differences in the actual algorithms.

4.6 Exercises

1. By hand, compute the mean and variance of the following set of data:

$$1, 2, 9, 6, 3, 4, 3, 8, 4, 2$$

2. Obtain a sampling of 1000 points using the uniform distribution: and 1000 points using the normal distribution:

```
x=rand(1000,1);
y=randn(1000,1);
```

Compare the distributions using Matlab's *hist* command: `hist([x y],100)` and print the results. You'll note that the histograms have not been scaled so that the areas sum to 1, but we do get an indication of the nature of the data.

3. Compute the value of K in the double Laplacian function so that f is a p.d.f.

4. Next, load a sample of human voice: `load laughter` If you type `whos`, you'll see that you have a vector y with the sound data. The computers in the lab do have sound cards, but they don't work very well with Matlab, so we won't listen to the sample. Before continuing, you might be curious about what the data in y looks like, so feel free to plot it. We want to look at the distribution of the data in the vector y, and compare it to the normal distribution. The mean of y is already approximately zero, but to get a good comparison, we'll take a normal distribution with the same variance:

```
clear
load laughter
whos
sound(y,Fs);  %This only works if there's a good sound card
s=std(y);
x=s*randn(size(y));
hist([x y],100);  %Blue is "normal", Red is Voice
```

Print the result. Note that the normal distribution is much flatter than the distribution of the voice signal.

5. Compute the covariance between the following data sets:

x	-1.0	-0.7	-0.4	-0.1	0.2	0.5	0.8
y	-1.3	-0.7	-0.1	0.5	1.1	1.7	2.3

(4.1)

6. Let $\mathbf{x}$ be a vector of data with mean μ, and let a, b be scalars. What is the mean of $a\mathbf{x}$? What is the mean of $\mathbf{x} + b$? What is the mean of $a\mathbf{x} + b$? (NOTE: Formally, the addition of a vector and a scalar is not defined. Here, we are utilizing Matlab notation: The result of a vector plus a scalar is addition done componentwise. This is only done with scalars- for example, a matrix added to a vector is still not defined, while it is valid to add a matrix and a scalar).

7. Let $\mathbf{x}$ be a vector of data with variance σ^2, and let a, b be scalars. What is the variance of $a\mathbf{x}$? What is the variance of $\mathbf{x}+b$? What is the variance of $a\mathbf{x}+b$?

8. Show that, for data in vectors $\mathbf{x}$, $\mathbf{y}$ and a real scalar a,

$$\text{Cov}(ax, y) = a\text{Cov}(x, y) \quad \text{Cov}(x, by) = b\text{Cov}(x, y)$$

9. Show that, for data in $\mathbf{x}$ and a vector consisting only of the scalar a,

$$\text{Cov}(x, a) = 0$$

10. Show that, for a and b fixed scalars, and data in vectors $\mathbf{x}, \mathbf{y}$,

$$\text{Cov}(x + a, y + b) = \text{Cov}(x, y)$$

11. If the data sets X and Y are the same, what is the covariance? What is the correlation coefficient? What if $Y = mX$? What if $Y = mX + b$?

12. Let X be a $p \times n$ matrix of data, where we n columns of p data points (you may assume each column has zero mean). Show that the $(i, j)^{\text{th}}$ entry of $\frac{1}{p}X^T X$ is the covariance between the i^{th} and j^{th} columns of X. HINT: It might be convenient to write X in terms of its columns,

$$X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$$

Also show that $\frac{1}{p}X^T X$ is a symmetric matrix.

13. This exercise shows us that our geometric insight might not extend to high dimensional space. We examine how points are distributed in high dimensional hypercubes and unit balls. Before we begin, let us agree that a hypercube of dimension n has the edges:

$$(\pm 1, \pm 1, \pm 1, \ldots, \pm 1)^T$$

so, for example, a 2-d hypercube (a square) has edges:

$$(1, 1)^T, (-1, 1)^T, (1, -1)^T, (-1, -1)^T$$

(a) Show that the distance (standard Euclidean) from the origin to a corner of a hypercube of dimension d is $\sqrt{d}$. What does this imply about the shape of the "cube", as $d \to \infty$?

(b) The volume of a $d-$dimensional hypersphere of radius a can be written as:

$$V_d = \frac{S_d a^d}{d}$$

where S_d is the $d-$dimensional surface area of the unit sphere.

First, compute the volume between hyperspheres of radius a and radius $a - \epsilon$.
Next, show that the ratio of this volume to the full volume is given by:

$$1 - \left(1 - \frac{\epsilon}{a}\right)^d$$

What happens as $d \to \infty$?
If we have 100,000 data points "uniformly distributed" in a hypersphere of dimension 10,000, where are "most" of the points?

4.7 Linear Regression

In this section, we examine the simplest case of fitting data to a function. We are given two sets of data-

$$X = \{x_1, x_2, \ldots, x_n\} \quad Y = \{y_1, y_2, \ldots, y_n\}$$

We wish to find the best linear relationship between X and Y. But what is "best"? It depends on how you look at the data, as described in the next three exercises.

1. **Exercise:** Let y be a function of x. Then we are trying to find m and b so that

   $$y = mx + b$$

   best describes the data. If the data were perfectly linear, then this would mean that:

   $$\begin{aligned} y_1 &= mx_1 + b \\ y_2 &= mx_2 + b \\ &\vdots \\ y_n &= mx_n + b \end{aligned}$$

   However, most of the time the data is not actually, *exactly* linear, so that the values of y don't match the line: $mx + b$. Thus we have an error:

   $$E_1 = \sum_{k=1}^{n} |y_k - (mx_k + b)|$$

   (a) Show graphically what this error would represent for one of the data points.

   (b) E_1 is a function of two variables, m and b. What is the difficulty in determining the minimum[2] error using this error function?

[2]We could solve this problem by using Linear Programming, but that is outside the scope of this text.

(c) Define the squared error as:

$$E_{\text{se}} = \sum_{k=1}^{n} (y_k - (mx_k + b))^2$$

Why is it appropriate to use this error instead of the other error?

(d) E_{se} is a function of m and b, so the minimum value occurs where

$$\frac{\partial E_{\text{se}}}{\partial m} = 0 \quad \frac{\partial E_{\text{se}}}{\partial b} = 0$$

Show that this leads to the system of equations: (the summation index is 1 to n)

$$\begin{aligned} m \sum x_k^2 + b \sum x_k &= \sum x_k y_k \\ m \sum x_k + bn &= \sum y_k \end{aligned}$$

(e) **Exercise:** Write a Matlab routine that will take a $2 \times n$ matrix of data, and output the values of m and b found above. The first line of code should be:

```
function [m,b]=Line1(X)
```

and save as `Line1.m`.

2. **Exercise:** If we treat x as a function of y (i.e., $x = f(y)$), how does that change the equations above? Draw a picture of what the error would represent in this case. Write a new Matlab routine `[m,b]=Line2(X)` to reflect these changes.

3. **Exercise:** The last case is where we treat x and y independently, so that we don't assume that one is a function of the other.

(a) Show that, if $ax + by + c = 0$ is the equation of the line, then the distance from (x_1, y_1) to the line is

$$\frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$

which is the size of the orthogonal projection of the point to the line. This is actually problem 53, section 11.3 of Stewart's Calculus text, if you'd like more information.
(HINT: The vector $[a, b]^T$ is orthogonal to the line $ax + by + c = 0$. Take an arbitrary point P on the line, and project an appropriate vector to $[a, b]^T$.)
Conclude that the error function is:

$$E = \sum_{k=1}^{n} \frac{(ax_k + by_k + c)^2}{a^2 + b^2}$$

(b) Draw a picture of the error in this case, and compare it graphically to the error in the previous 2 exercises.

(c) The optimimum value of E occurs where $\frac{\partial E}{\partial c} = 0$. Show that if we mean subtract X and Y, then we can take $c = 0$. This leaves only two variables.

(d) Now our error function is:

$$E = \sum_{k=1}^{n} \frac{(ax_k + by_k)^2}{a^2 + b^2}$$

Show that we can transform this function (with appropriate assumptions) to:

$$E = \sum_{k=1}^{n} \frac{(x_k + \mu y_k)^2}{1 + \mu^2}$$

(for some μ), and conclude that E is a function of one variable.

(e) Now the minimum occurs where $\frac{dE}{d\mu} = 0$. Compute this quantity to get:

$$\mu^2 A + \mu B + C = 0$$

where A, B, C are expressions in $\sum x_k y_k$, $\sum x_k^2$, $\sum y_k^2$. This is a quadratic expression in μ, which we can solve. Why are there (possibly) 2 real solutions?

(f) Write a Matlab routine `[a,b,c]=Line3(X)` that will input a $2 \times n$ matrix, and output the right values of a, b, and c.

4. **Exercise:** Try the 3 different approaches on the following data set, which represents heights (in inches) and weight (in lbs.) of 10 teenage boys. (Available in `HgtWgt.mat`)

X	69	65	71	73	68	63	70	67	69	70
Y	138	127	178	185	141	122	158	135	145	162

Plot the data with the 3 lines. What do the 3 approaches predict for the weight of someone that is 72 inches tall?

5. **Exercise:** Do the same as the last exercise, but now add the data point $(62, 250)$. Compare the new lines with the old. Did things change much?

6. **Matlab Note:** Consider using the command `subplot` to plot multiple graphs on the same figure. For example, try the following sequence of commands:

```
x=linspace(-8,8);
y1=sin(x);
y2=sin(2*x);
```

```
y3=sin(x.*x);
y4=sin(exp(-x));
subplot(2,2,1);
plot(x,y1);
subplot(2,2,2);
plot(x,y2);
subplot(2,2,3);
plot(x,y3);
subplot(2,2,4);
plot(x,y4);
```

4.8 The Median-Median Line:

The median of data is sometimes preferable to the mean, especially if there exists a few data points that are far different than "most" data.

1. **Definition:** The *median* of a data set is the value so that exactly half of the data is above that point, and half is below. If you have an odd number of points, the median is the "middle" point. If you have an even number, the median is the average of the two "middle" points. Matlab uses the `median` command.

2. **Exercise:** Compute (by hand, then check with Matlab) the medians of the following data:

   - $$1, 3, 5, 7, 9$$
   - $$1, 2, 4, 9, 8, 1$$

   The motivation for the median-median line is to have a procedure for line fitting that is not as sensitive to "outliers" as the 3 methods in the previous section.

Median-Median Line Algorithm

- Separate the data into 3 equal groups (or as equal as possible). Use the $x-$axis to sort the data.
- Compute the median of each group (first, middle, last).
- Compute the equation of the line through the first and last median points.
- Find the vertical distance between the middle median point and the line.
- Slide the line 1/3 of the distance to the middle median point.

3. **Exercise:** Understand the commands in the program `mmline`. There are several new matlab functions used. Below are some hints as to how we'll divide the data into three groups.

   - If we divide a number by three, we have three possible remainders: $0, 1, 2$. What is the most natural way of seperating data in these three cases (i.e., if we had 27, 28 or 29 data points)?
   - Look at the Matlab command `rem`. Notice that:

     `rem(27,3)=0  rem(28,3)=1  rem(29,3)=2`

   - Look at the Matlab command `sort`. The full command looks like: `[s,index]=sort(x)` The output vector s will be x sorted. The vector `index` will contain which order the original indices were in. That is,

     `x(index)=s`

   - We can therefore sort x first, then sort y according to the index for x.

4. **Exercise:** Try this algorithm on the last data set, then add the new data point. Did your lines change as much?

5. **Exercise:** Consider the following data set [**?**] which relates the index of exposure to radioactive contamination from Hanford to the number of cancer deaths per $100,000$ residents. We would like to get a relationship between these data. Use the four techniques above, and compare your answers. Compute the actual errors for the first three types of fits and compare the numbers.

County/City	Index	Deaths
Umatilla	2.5	147
Morrow	2.6	130
Gilliam	3.4	130
Sherman	1.3	114
Wasco	1.6	138
Hood River	3.8	162
Portland	11.6	208
Columbia	6.4	178
Clatsop	8.3	210