Learning from Data 1
Preliminaries : Data and Mathematics

David Barber
dbarber@anc.ed.ac.uk
course page : http://anc.ed.ac.uk/∼dbarber/lfd1/lfd1.html

1 What kinds of Data, and how to represent it?

There are generally three types of data entries that we can encounter; these are *categorical*, *ordinal* and *numerical* types. Since we will ultimately wish to perform computations with the data, we need to transform any entries in the database which are non-numerical into numerical values. Some care needs to be taken at this point, since biases can be inadvertently introduced at this stage, as described below.

1.1 Categorical

For categorical (or nominal) data, the observed value belongs to one of a number of classes, and there is no intrinsic ordering of the classes. An example of a categorical variable would be the description of the type of job that someone does, e.g. healthcare, education, financial services, transport, homeworker, unemployed, engineering etc. One way to transform this data into numerical values would be to use 1-of-m encoding. Here's an example: There are 4 kinds of jobs: soldier,sailor,tinker,spy. A person who is a soldier is represented as (1,0,0,0), a sailer as (0,1,0,0), a tinker as (0,0,1,0) and a spy as (0,0,0,1). So, here's how to transform the following records :

1-of-m encoding

age	32	63
profession	sailor	spy

$\rightarrow$

age	32	63
profession1	0	0
profession2	0	0
profession3	1	0
profession4	0	1

This kind of coding is sensible since the distance between two vectors representing two different professions (see mathematics notes) is constant (equal to 1).

Attributes and values

Any database is described by a list of attributes. In the above example on the left, there are two attributes, 'age' and 'profession'. Each record in the database corresponds to a setting of the list of attributes to a particular value. (Sometimes data entries might be missing, so that we would talk of 'missing data' or 'missing values'). In the above example on the right, there are five attributes, 'age', 'profession1','profession2','profession3' and 'profession4'.

Warning!

It is clear that 1-of-m encoding induces dependencies in the profession attributes. Clearly, if one of the profession attributes is 1, the others must be zero. Be very careful (especially with Naive Bayes) that you do not fall into the trap of assuming that these attributes are independent.

1.2 Ordinal

An ordinal variable again consists of categories, but there there is an ordering or ranking of the categories, e.g. cold, cool, warm, hot or 3, 2(ii), 2(i), 1 (for university degrees).

In this case, we may wish to encode the fact that there is an explicit ordering in these data. Thus, we could perhaps use -1 for cold, 0 for cool and +1 for warm and +2 for hot. Clearly, this choice is somewhat arbitrary, and one should keep in mind that any results will be dependent on the choice of coding.

1.3 Numerical

Numerical data takes on values that are real numbers, e.g. a temperature measured by a thermometer, or the salary that someone earns.

2 What's this course all about?

The course is about fitting models to data , and using them to answer questions about the data. Technically, these are called learning and inference. It's interesting to look at the dictionary definition of these words :

To *learn*:

- To gain knowledge, comprehension, or mastery of through experience or study.
- To fix in the mind or memory; memorize: learned the speech in a few hours.
- To acquire experience of or an ability or a skill in: learn tolerance; learned how to whistle.
- To become aware: learned that it was best not to argue.

To *infer*:

- To conclude from evidence or premises
- To reason from circumstance
- surmise: We can infer that his motive in publishing the diary was less than honorable
- To lead to as a consequence or conclusion: Socrates argued that a statue inferred the existence of a sculptor.
- To hint; imply.

It's clear that "learning" is always contextual – what do you learn; in what area have you gained a skill? We will attempt to make 'machines' that can be skilled in particular areas in which we provide data and possibly additional hints.

2.1 Learning

Statistics in disguise?

Learning is the process of fitting a model to data. For example, in a supervised learning scenario, we might be interested in learning a mapping from inputs to outputs. One way to do this is to postulate the existence of some kind of underlying data generating mechanism for which we do not know the exact parameter settings. In many ways, this is related to statistics. Whatever you decide to call it, ultimately, fitting models to data is the domain of statistics. What changes is the domain expertise that can be brought to bear. For example, a physicist may be rather indignant to be called a statistician just because s/he fits a postulated physical model to data from sub-nuclear experiments. Similarly, a researcher in Natural Language Processing may be rather annoyed to hear that "s/he's just a statistician". Indeed, science is (largely) about fitting models to data – it's just that domain knowledge can be very deep and specific. (The difficult and interesting thing about science is suggesting – from amongst the infinite sea of possibilities – a useful restricted class of models in the first place.) In this course, our domain of interest is applications in some rather non-specific areas. Is it possible to come up with methods and models that work in the absence of specific domain knowledge? Well, not really. We shall provide various general hints that always bias our solutions in a certain way that usually intuitively corresponds to the way we think (hope!) the world works.

Mathematics is the language of science. Probability is the logic of science.

2.2 Inference

A doctor spends a lifetime learning both from textbooks but from patients. A new patient comes into the surgery exhibiting a list of symptoms. The doctor (based on her own internal model of how symptoms and diseases are

related) infers that the patient has glandular fever.

Inference is using the learned model to answer specific questions. For example, we may have fitted a classification model from a database. We might be then interested in using the model to infer the class of a novel input. For example, will this person default on their loan if they are married, ask for a loan of 100000 pounds, and are a teacher?

3 What kind of Learning?

Machine learning is traditionally (and not always helpfully!) split into two main areas: supervised, and unsupervised learning. The difference between the two depends on what kind of question you wish the data to try to answer (and possibly on the data available). (Reinforcement learning –covered in LFD2 – is a kind of supervised learning in which the supervisor provides rewards for actions which improve a situation and penalties for deleterious actions).

3.1 Unsupervised Learning

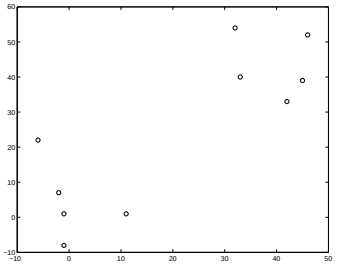
A baby processes a mass of initially confusing sensory data. After a while the baby begins to understand her environment in the sense that novel sensory data from the same environment is familiar or expected. When a strange face presents itself, the baby recognises that this is not familiar and may be upset. The baby has learned a representation of the familiar and can distinguish the expected from the unexpected; this is an example of unsupervised learning.

Descriptive modelling

In a mathematical sense, here we just wish to fit a model which describes succinctly and accurately the data in the database. That is, there is no supervisor telling us what is right or wrong – we simply observe some data and try to describe it in an efficient way with our model. For example, here are some points:

x_1	-2	-6	-1	11	-1	46	33	42	32	45
x_2	7	22	1	1	-8	52	40	33	54	39

This is an example of *unlabelled* data. In a sense, there are no outputs, only inputs. We can visualise this data by plotting it in 2 dimensions:



By simply eye-balling the data, we can see that there are two apparent clusters here, one centred around (0,0) and the other around (35,35). A reasonable model to describe this data might therefore be to describe it as two clusters, centred at (0,0) and (35,35), each with a variance (spread) of around 1.

3.2 Supervised Learning

I'm fond of the following story :

" A father decides to teach his young son what a sports car is. Finding it difficult to explain in words, he decides to try to explain by examples.

They stand on a motorway bridge and, as each car passes underneath, the father cries out 'that's a sports car!' when a sports car passes by. After ten minutes, the father asks his son if he's got it. The son says, 'sure, it's easy'. An old red VW Beetle passes by, and the son shouts – 'that's a sports car!'. Dejected, the father asks – 'why do you say that?'. 'Because all sports cars are red!', replies the son. "

This story is an example of supervised learning. Here the father is the supervisor, and his son is the 'learner', or 'machine learner' or 'predictor'. The nice point about this story is that you can't expect miracles – unless you explicitly give extra information, learning from examples may not always give you what you might hope for. On the other hand, if they had been there the whole week, probably the son would have learned a reasonably good model of a sports car, and helpful hints by the father would be less important. It's also indicative of the kinds of problems typically encountered in machine learning in that it is not really clear anyway what a sports car is – if we knew that, then we wouldn't need to go through the process of learning!

Predictive modelling

We typically have a training set of *labelled* data, for example, here are some data

nationality	British	Dutch	Taiwanese	British
height(cm)	175	195	155	165
sex	m	m	f	f

We might have a large database of such entries. A supervised learning problem might be: given a new, previously unseen (nationality,height) pair, predict the sex of the person. For example, given that a person is Taiwanese and 167cm tall, are they going to be male or female? In this case we see the training data as a collection of (input,output) pairs, where the output or label has been given by a 'supervisor'. Ultimately, we wish to form a mapping from the inputs to the output (possibly more than one output) that accurately describes the label/output given the inputs. Ideally, we would like our model to *generalise* well (predict accurately) novel test data not seen previously during the model building process.

Uncertainty

Note that this is a good example to motivate our later ideas about probability/uncertainty – there is clearly not going to be absolute certainty about our predictions in this case since there are always going to be tall females and shorter males that will make classifying a novel person an inexact science. However, we may be able to infer what is the probability that a novel person is male, given our trained model. In practice, uncertainty often plays a major role in machine learning, and we need to use a framework that can handle this. Uncertainty is not just an issue in supervised learning. Also we may be uncertain as to the exact values in an unsupervised set of data, and we may wish to take this into account in building a model. In my humble opinion, any method that does not take uncertainty into account (such as decision trees) are not really worth their salt. However, many traditional methods from machine learning and computer science are non-probabilistic and some are included in this course since their use is (unfortunately!) rather widespread.

Supervised learning problems traditionally come in two flavours, classification and regression.

Classification

Given a set of inputs, predict the class (one of a finite number of discrete labels). Normally, the class is ordinal (there is no intrinsic information in the class label). For example, given an image of a handwritten digit, predict whether it is 0,1,2,3,4,5,6,7,8 or 9. This would be a 10-class classification problem. Many problems involve binary classes (you can always convert a multi-class problem into a set of binary class problems – though this is not always natural or desirable). For binary classes, there is usually no

information as to whether we say the data are labelled as class 0 or 1, or alternatively as class 1 or 2. For example, the sports-car classification problem would have been the same if the father said '1' or '0' when the car passing by was a sports car or not. A great deal of problems in the machine learning arena are classification problems. Uncertainty will ultimately play a key role in any real world application. Can we really say that Mr Smith will definitely default on his loan? This may seem a very strong statement if there is little obvious difference between the attributes of Mr Smith and Mr Brown.

Regression Given a set of inputs, predict the output value (one of a potentially infinite set of real-valued points). For example, given historical stock market data, predict the course of the FTSE for tomorrow.

4 Mathematical Representation

In order to conveniently describe the data and algorithms in a mathematical way, we will typically use vectors. Thus, the dataset could be represented as

coffee	1	0	0
tea	0	0	1
milk	1	0	1
beer	0	0	0
diapers	0	0	1
aspirin	0	1	0

as $\mathbf{x}^1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{x}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \mathbf{x}^3 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

where each vector $\mathbf{x} = (x_1, \ldots, x_6)^T$ is a 6 dimensional vector whose components represent the values of each attribute. Note that the upper-index, *e.g.*,$\mathbf{x}^3$ refers simply to the third datapoint (vector) in the dataset. (Some care is needed since it is common in mathematics to use $\mathbf{a}^2$ to refer to $\mathbf{a}^T\mathbf{a}$. I will also use this notation in the text – hopefully the context should dispel any confusion.)

In general, we can represent any dataset as a set of vectors $X = \{\mathbf{x}^1, \ldots \mathbf{x}^P\}$, which would represent a dataset of P items. Normally I will use an index (upper) to denote which datapoint we are referring to, and a suffix (lower) to denote the attribute (or component) of the data vector. Thus x_6^4 would represent the sixth attribute of the fourth datapoint. A dataset, being a collection of datapoints can then be represented as a matrix X in which the element $X_{ij} = x_i^j$.

4.1 A Matlab Interlude

```
% A first look at matlab

% Here are a set of datapoints (done in a longhanded way!)
x(:,1)=[-2 7]'; x(:,2)=[-6 22]'; x(:,3)=[-1 1]'; x(:,4)=[11 1]';
x(:,5)=[-1 -8]'; x(:,6)=[46 52]';x(:,7)=[33 40]'; x(:,8)=[42 33]';
x(:,9)=[32 54]'; x(:,10)=[45 39]';

x  % each *column* represents a 2 dimensional datapoint, of which there are 10

mean(x')' % need transposes since matlab assumes that data are arranged
          % in rows. To be consistent with our mathematics though, we
          % will use the column representation (at the cost of having to
          % use a few more transposes)

plot(x(1,:),x(2,:),'o')
```

5 What kind of Approaches?

This course focusses on a modelling approach, framed within probability theory. Some of the approaches we talk about (such as decision trees, and KNN and K-means clustering) are non-probabilistic. However, probabilistic versions of these methods have been developed but are beyond the scope of this course.

5.1 Why do we expect to be able to learn anything?

Arguably all machine learning approaches are based on some notion of smoothness or regularity underlying the mechanism that generated the observed data. Roughly speaking : if two people (datapoints) are close neighbours, they are likely to behave similarly. We will usually frame this intuition mathematically, in which we will need to be precise by what we mean by 'close' and 'behave'. This naturally places machine learning in the arena of measuring distances between datapoints, a natural place for vectors and vector algebra.

5.2 How are we going to learn?

The general procedure will be to postulate some model and then adjust it's parameters to best fit the data. For example in a regression problem, we may think that the data $\{(x^\mu, y^\mu), \mu = 1, \ldots, P\}$, where x is an input and y an output, is well modelled by the function $y = wx$, and our task is to find an appropriate setting of the parameter w. An obvious way to do this is to see how well the current model predicts the training data that we have, and then to adjust the parameter w to minimise the errors that our model makes on predicting the data. This general procedure will usually involve therefore optimisation methods, usually in high dimensional spaces (although the above is a one-dimensional example). Sometimes, the parameter space that we will search for a solution in is discrete (for example, in evolution, the parameters are the discrete sequence of genes, and the optimisation function is survival). Optimisation in discrete parameter spaces can be very difficult – however, there are many approaches developed in mathematics to do this. Recently, some computer scientists have been excited by 'genetic algorithms' – it is worth bearing in mind that these are simply discrete optimisation methods (and in my opinion, rather poor ones). Why should optimisation mechanisms based on sexual selection have any relevance to finding the best construction of a bridge? My point is that you shouldn't be swayed by applying 'sexy' methods to areas which are not obviously related.

Genetic Algorithms

Noise, overfitting and Generalisation

In the case that there is noise on the data (sometimes, the father might be inconsistent in his labelling of sports cars, or there might be essentially random perturbations on the FTSE index), we don't want to model this noise. That is, we have to be careful to make sure that our models only capture the underlying process that we are truly interested in, and not necessarily the exact details of the training data. If we have an extremely flexible model, it may overfit noisy training data be a very poor predictor of future novel inputs (that is, it will generalise poorly). This is very important topic and central to machine learning. We shall return to this in a later chapter.

6 Mathematics Required

The material here is presented to give you an idea of the level of mathematics required. Don't worry if you don't understand too well all of the following. However, it would be very useful for you to foster enthusiasm for learning this material since I may ask exam questions using this level of mathematics.

6.1 Vectors

The course assumes that you are familiar with the basics of vectors and vector calculations. Let $\mathbf{x}$ denote the n-dimensional vector with components

$$(x_1, x_2, \cdots, x_n)$$

Then $|\mathbf{x}|$ denotes the length of this vector, using the usual Euclidian definition:

$$|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

The inner product $\mathbf{w} \cdot \mathbf{x}$ is defined as:

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

and has a natural geometric interpretation as:

$$\mathbf{w} \cdot \mathbf{x} = |\mathbf{w}|\,|\mathbf{x}| \cos(\theta)$$

where θ is the angle between the two vectors. Thus if the lengths of two vectors are fixed their inner product is largest when $\theta = 0$, whereupon one is just some constant multiple of the other.

6.2 Matrices

The course assumes some familiarity with matrices, which are shown as upper-case bold letters such as $\mathbf{A}$. If the element of the i-th row and j-th column is a_{ij}, then $\mathbf{A}^T$ denotes the matrix that has a_{ji} there instead - the *transpose* of $\mathbf{A}$. So, for example if $\mathbf{A}$ is a 3×3 matrix:

$$\mathbf{A} = \begin{pmatrix} 2 & 3 & 4 \\ 4 & 5 & 9 \\ 6 & 7 & 1 \end{pmatrix}$$

then the transpose (written $\mathbf{A}^T$) is:

$$\mathbf{A}^T = \begin{pmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 1 \end{pmatrix}$$

The product of two matrices $\mathbf{A}$ and $\mathbf{B}$ has $\sum_k a_{ik} b_{kj}$ in the i-th row and j-th column.

The matrix $\mathbf{I}$ is the identity or unit matrix, necessarily square, with 1s on the diagonal and 0s everywhere else. If $\det(\mathbf{A})$ denotes the determinant of a square matrix $\mathbf{A}$ then the equation

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0$$

is called the *characteristic polynomial* of $\mathbf{A}$. Using the example above, the characteristic polynomial would be:

$$\begin{vmatrix} 2-\lambda & 3 & 4 \\ 4 & 5-\lambda & 9 \\ 6 & 7 & 1-\lambda \end{vmatrix} = 0$$

which is

$$(2-\lambda)((5-\lambda)(1-\lambda)-63)-3(4(1-\lambda)-54)+4(28-6(5-\lambda))=0$$

which simplifies to:

$$-\lambda^3+8\lambda^2+82\lambda+26=0$$

Note that a square matrix must satisfy its own characteristic polynomial, by definition of the polynomial, so (pre- or post-multiplying through by $\mathbf{A}^{-1}$) it provides a way to calculate the inverse of a matrix using only matrix multiplication, if that inverse exists. Clearly the inverse exists if and only if the matrix is square and $\det(\mathbf{A}) \neq 0$ (note that $\det(\mathbf{A})$ is the constant term in the characteristic polynomial).

The roots of the characteristic polynomial are called the *eigenvalues* of the matrix. Note that if $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{x}$ is an n-dimensional (column) vector, then

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

represents a linear map into an m-dimensional space. If $\mathbf{A}$ happens to be a square matrix then any vector which is transformed by the linear map into a scalar multiple of itself is called an *eigenvector* of that matrix. Obviously, in that case $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ for some λ. The eigenvectors can be found by finding the eigenvalues and then solving the linear equation set:

$$(\mathbf{A}-\lambda\mathbf{I})\mathbf{x}=0$$

An *orthogonal matrix* is a square matrix $\mathbf{A}$ such that $\mathbf{A}^T = \mathbf{A}^{-1}$. Such matrices represent a mapping from one rectangular co-ordinate system to another. For such a matrix,

$$\mathbf{A}\mathbf{A}^T = \mathbf{I}$$

- the inner product of any two different rows is 0 and the inner product of any row with itself is 1.

6.3 Basic combinatorics

The number of ways of selecting k items from a collection of n items is

$$\left(\begin{array}{c} n \\ k \end{array}\right) = \frac{n!}{k!(n-k)!}$$

if the ordering of the selection doesn't matter. This quantity is also the coefficient of x^k in the expansion of $(1+x)^n$. Stirling's formula provides a useful approximation for dealing with large factorials:

$$n! \approx n^n e^{-n}\sqrt{2\pi n}$$

There are a huge number of formulae involving combinations. For example, since $(1+x)^{n+1} = (1+x)^n(1+x)$ it is clear that

$$\left(\begin{array}{c} n \\ k \end{array}\right) + \left(\begin{array}{c} n \\ k+1 \end{array}\right) = \left(\begin{array}{c} n+1 \\ k+1 \end{array}\right)$$

and so on.

6.4 Basic probability and distributions

A random variable X is a variable which, in different experiments carried out under the same conditions, assumes different values x_i, each of which then represents a random event. A discrete random variable can take one of only a finite, or perhaps a countably infinite, set of values. A continuous random variable can take any value in a finite or infinite interval. Random variables are completely characterised by their probability density and distribution functions.

For a discrete random variable, if $p(X = x)$ is the probability that it takes the value x then

$$F(x) = p(X < x)$$

is the distribution function of X. For a continuous random variable, there is a probability density function $f(x)$ such that

$$\int_{-\infty}^{\infty} f(x)\, dx = 1$$

and the distribution function is then:

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

For a discrete random variable, the mean value μ is

$$\mu = \sum x_i p(X = x_i)$$

and for a continuous variable it is

$$\mu = \int_{-\infty}^{\infty} t f(t)\, dt$$

The variance σ^2 is, for a discrete variable:

$$\sigma^2 = \sum (x_i - \mu)^2 p(X = x_i)$$

and for a continuous variable:

$$\sigma^2 = \int_{-\infty}^{\infty} (t - \mu)^2 f(t)\, dt$$

There are several widely-occurring distributions that are worth knowing about. Suppose that some event will happen with fixed probability p. Then the probability that it will happen exactly k times in n trials is

$$\left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k}$$

and this is the binomial distribution. It has mean np and variance $np(1-p)$. If one lets $n \to \infty$ one gets the Gaussian or normal distribution, typically parameterised by two constants a and b; it has density function

$$\frac{1}{a\sqrt{2\pi}} e^{-(x-b)^2/(2a^2)}$$

with mean b and variance a^2. If one starts with the binomial distribution and lets $n \to \infty$ and $p \to 0$ with the extra assumption that $np = a$, where a is some constant, then one gets the Poisson distribution with density function

We will not use the Poisson distribution in this course

$$\frac{a^k e^{-a}}{k!}$$

with mean and variance both a.

6.5 Partial differentiation

If $z = f(x_1, x_2, \cdots, x_n)$ is a function of n independent variables then one can form the partial derivative of the function with respect to one variable (say x_i),

$$\frac{\partial f}{\partial x_i}$$

by treating all other variables as constant. For example, if

$$f = xy + y^3$$

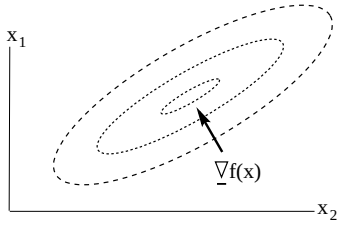


Figure 1: Interpreting the gradient. The ellipses are contours of constant function value, $f = const$. At any point $\mathbf{x}$, the gradient vector $\nabla f(x)$ points along the direction of maximal increase of the function.

then

$$\frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x + 3y^2$$

The geometric significance of a quantity such as $\frac{\partial f}{\partial x}$ is as follows. If the function f is plotted and represents some suitably well-behaved surface, then this partial derivative represents the slope of the surface in a direction parallel to the x-axis at any given point (x, y). The total derivative dz is given by

$$dz = \sum_i \frac{\partial z}{\partial x_i} dx_i$$

and clearly, if all the x_i are functions of one variable t then

$$\frac{dz}{dt} = \sum_i \frac{\partial z}{\partial x_i} \frac{dx_i}{dt}$$

There is a directly analogous version of this 'chain rule' for the case where the x_i are each functions of several variables and you wish to find the partial derivative of z with respect to one of those variables.

Exercise: Find the partial derivatives of the function

$$f(x, y, z) = (x + 2y)^2 \sin(xy)$$

6.6 The gradient vector operator

Consider a function $\phi(\mathbf{x})$ that depends on a vector $\mathbf{x}$. We are interested in how the function changes when the vector $\mathbf{x}$ changes by a small amount : $\mathbf{x} \to \mathbf{x} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is a vector whose length is very small. According to a Taylor expansion, the function ϕ will change to

$$\phi(\mathbf{x} + \boldsymbol{\delta}) = \phi(\mathbf{x}) + \sum_i \delta_i \frac{\partial \phi}{\partial x_i} + O\left(\boldsymbol{\delta}^2\right) \tag{6.1}$$

We can interpret the summation above as the scalar product between the vector $\nabla\phi$ with components $[\nabla\phi]_i = \frac{\partial \phi}{\partial x_i}$ and $\boldsymbol{\delta}$.

$$\phi(\mathbf{x} + \boldsymbol{\delta}) = \phi(\mathbf{x}) + (\nabla\phi)^T \boldsymbol{\delta} + O\left(\boldsymbol{\delta}^2\right) \tag{6.2}$$

6.7 Interpreting the Gradient $\nabla f(x)$

The gradient points along the direction in which the function increases most rapidly. Why?

Consider a direction $\hat{\mathbf{p}}$ (a unit legnth vector). Then a displacement, δ units along this direction changes the function value to

$$f(\mathbf{x} + \delta\hat{\mathbf{p}}) \approx f(\mathbf{x}) + \delta \nabla f(x) \cdot \hat{\mathbf{p}}$$

The direction $\hat{\mathbf{p}}$ for which the function has the largest change is that which maximises the overlap

$$\nabla f(x) \cdot \hat{\mathbf{p}} = |\nabla f(x)||\hat{\mathbf{p}}| \cos\theta = |\nabla f(x)| \cos\theta$$

where θ is the angle between $\hat{\mathbf{p}}$ and $\nabla f(x)$. The overlap is maximised when $\theta = 0$, giving $\hat{\mathbf{p}} = \nabla f(x)/|\nabla f(x)|$. Hence, the direction along which the function changes the most rapidly is along $\nabla f(x)$.

6.8 Optimization: Lagrange multipliers

Suppose that you wish to find the stationary points (maxima or minima) of some n-argument function $f(\mathbf{x}) = f(x_1, \cdots, x_n)$, subject to the m constraints $g_1(\mathbf{x}) = 0, \cdots, g_m(\mathbf{x}) = 0$. Lagrange showed that they could be found as the solution of the $(n + m)$ equations in the $(n + m)$ variables $x_1, \cdots, x_n, \lambda_1, \cdots, \lambda_m$:

$$\begin{aligned}
\frac{\partial f}{\partial x_1} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_1} &= 0 \\
&\cdots \\
\frac{\partial f}{\partial x_n} - \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_n} &= 0 \\
g_1(\mathbf{x}) &= 0 \\
&\cdots \\
g_m(\mathbf{x}) &= 0
\end{aligned}$$

where the λ_j are m specially-introduced variables called Lagrange multipliers. This theorem provides a handy way to tackle a range of optimization problems. Notice that the above equations are the $(n+m)$ partial derivatives of the function

$$f - \sum_{j=1}^{m} \lambda_j g_j$$

each set to zero.

For example, to find the maximum of $f(x, y) = x+y$ subject to the constraint $x^2 + y^2 = 1$, solve:

$$\begin{aligned}
1 - 2\lambda x &= 0 \\
1 - 2\lambda y &= 0 \\
x^2 + y^2 - 1 &= 0
\end{aligned}$$

to get $x = y = \lambda = \pm 1/\sqrt{2}$, after which you should then check to determine which of these two solutions is the true maximum.

Exercise: Find the maximum of $y - x$ subject to the constraint $y + x^2 = 4$.

You can find the answer to the same problem experimentally as follows. Plot the graph of $y = 4 - x^2$ and the graph of $y = x + m$, and find the largest value of m such that the two graphs still intersect.

7 Optimization methods

We are given a function $E(\mathbf{w})$ which depends on the vector of variables $\mathbf{w}$. We can also calculate the vector of partial derivatives (or gradient) $\mathbf{g}(\mathbf{w})$ where $g_i = \partial E/\partial w_i$. How should be use this information to optimize E? There are two methods that we consider

1. Gradient descent with fixed stepsize

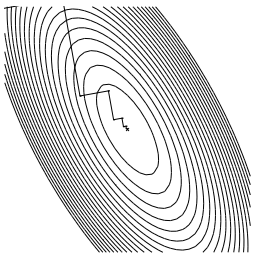


Figure 2: Optimisation using line search along steepest descent directions. Rushing off following the steepest way downhill from a point (and continuing for a finite time in that direction) doesn't always result in the fastest way to get to the bottom!

2. Gradient descent with line searches

One of the most powerful general purpose optimisation methods is conjugate-gradients (this is implemented in the NETLAB package for MATLAB), and I would recommend that you use this in any (continuous) optimisation problem. It is based on line search techniques.

7.1 Gradient descent with fixed stepsize

Locally, if we are at point $\mathbf{w}$, the quickest way to decrease E is to take a step in the direction $-\mathbf{g}(\mathbf{w})$. If we make the update equation

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \mathbf{g}(t)$$

then we are doing gradient descent with fixed stepsize η. If η is non-infinitesimal, it is always possible that we will step over the true minimum. Making η very small guards against this, but means that the optimization process will take a very long time to reach a minimum.

7.2 Gradient descent with line searches

An obvious extension to the idea of gradient descent is to choose the direction of steepest descent, as indicated by the gradient $\mathbf{g}$, but to calculate the value of the step to take which most reduces the value of E when moving in that direction. This involves solving the one-dimensional problem of minimizing $E(\mathbf{w}(t) - \lambda \mathbf{g}(t))$ with respect to λ, and is known as a *line search*. That step is then taken and the process repeated again.

Finding the size of the step takes a little work; for example, you might find three points along the line such that the error at the intermediate point is less than at the other two, so that there is some minimum along the line lies between the first and second or between the second and third, and some kind of interval-halving approach can then be used to find it. (The minimum found in this way, just as with any sort of gradient-descent algorithm, may not be a global minimum of course.) There are several variants of this theme. Notice that if the step size is chosen to reduce E as much as it can in that direction, then no further improvement in E can be made by moving in that direction for the moment. Thus the next step will have no component in that direction; that is, the next step will be at right angles to the one just taken. This can lead to zig-zag type behaviour in the optimisation, see fig(2).

8 Just for Interest ....

Sections at the end of each chapter contain material of interest related to the course.

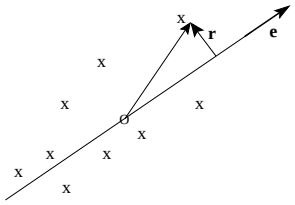


Figure 3: Learning the best line fit through a set of data. For zero mean data, we want to find the best unit length vector $\mathbf{e}$ such that when data points are projected onto the direction $\mathbf{e}$, the residual $\mathbf{r}$ is minimised.

8.1 Whence optimisation?

This section is rather more technical and not directly examinable. It is provided for the interest of the more motivated student and demonstrates that vector algebra, eigenvectors and Lagrange multipliers are central to even the simplest data modelling problems.

Optimisation plays a central role in learning from data. For motivation, consider the following simple problem of trying to explain data by a straight line.

Let $\mathbf{x}^\mu, \mu = 1, \ldots P$ be a set of P data points. We wish to "learn" the best straight line approximation to the data.

Each datapoint $\mathbf{x}$ can be expressed in terms of component parallel to the direction $\mathbf{e}$, and one orthogonal to $\mathbf{e}$:

$$\mathbf{x} = (\mathbf{x} \cdot \mathbf{e})\mathbf{e} + \mathbf{r} \tag{8.1}$$

(remember that $\mathbf{e} \cdot \mathbf{e} = 1$. Hence the squared length of the residual vector $\mathbf{r}$ is

$$\mathbf{r}^2 = (\mathbf{x} - (\mathbf{x} \cdot \mathbf{e})\mathbf{e})^2 = \mathbf{x} \cdot \mathbf{x} - 2(\mathbf{x} \cdot \mathbf{e})(\mathbf{x} \cdot \mathbf{e}) + (\mathbf{x} \cdot \mathbf{e})(\mathbf{x} \cdot \mathbf{e}) = \mathbf{x} \cdot \mathbf{x} - (\mathbf{x} \cdot \mathbf{e})^2 \tag{8.2}$$

Thus, finding $\mathbf{e}$ that minimises the residual is equivalent to finding $\mathbf{e}$ that maximises $(\mathbf{x} \cdot \mathbf{e})^2$. If we wish to find the best direction for the whole dataset, the problem can be stated as

$$\text{Find } \mathbf{e} \text{ to maximise } \sum_{\mu=1}^{P} (\mathbf{x}_\mu \cdot \mathbf{e})^2 \text{ such that } \mathbf{e} \cdot \mathbf{e} = 1 \tag{8.3}$$

Thus, this very simple problem of learning a straight line requires us to optimise a quadratic function subject to constraints.

We can restate the problem as

$$\text{Find } \mathbf{e} \text{ to maximise } \mathbf{e}^T A \mathbf{e} \text{ such that } \mathbf{e} \cdot \mathbf{e} = 1 \tag{8.4}$$

where A is the correlation matrix of the data, $A = (1/P) \sum_\mu \mathbf{x}_\mu \mathbf{x}_\mu^T$. One way to solve this is to use the Lagrangian method:

$$L = \mathbf{e}^T A \mathbf{e} + \lambda(1 - \mathbf{e}^T \mathbf{e}) \tag{8.5}$$

The gradient $\nabla_{\mathbf{e}}$ of L gives the condition $A\mathbf{e} = \lambda \mathbf{e}$. That is, $\mathbf{e}$ is an eigenvector of the correlation matrix. The eigenvector must be normalised to satisfy the constraint. For a 2 dimensional problem, there will, in general, be two eigenvectors – that corresponding to the largest eigenvalue is the solution that we require (since the error $\mathbf{e}^T A \mathbf{e} = \lambda$ at the optimum).