Machine Learning

Ben Herbst, Johan du Preez and Steve Kroon

Contents

Chapter 1. BASIC PROBABILITY 7
1.1. Introduction. 7
1.2. Discrete Probability. 7
1.3. Probability Densities. 21
1.4. Expectation and Covariances. 24
1.5. Decision Theory. 27

Chapter 2. PROBABILITY DENSITY FUNCTIONS 32
2.1. Introduction. 32
2.2. Binary Variables. 32
2.3. Multinomial Variables. 37
2.4. Model comparison. 39
2.5. Gaussian Distribution. 49
2.6. Linear Transformations of Gaussians and the central limit theorem. 64

Chapter 3. DIMENSIONALITY REDUCTION 67
3.1. Introduction. 67
3.2. Principal Components . 68
3.3. Numerical Calculation. 76
3.4. Linear Discriminant Analysis (LDA) 78

Chapter 4. CLASSIFICATION 87
4.1. Introduction. 87
4.2. Probabilistic Generative Models. 88
4.3. Probabilistic Discriminative Models. 94

Chapter 5. PARTIALLY OBSERVED DATA AND THE EM ALGORITHM 102
5.1. Introduction. 102

5.2. K-Means Clustering. 102
5.3. Gaussian Mixture Models. 105
5.4. The Expectation Maximization (EM) Algorithm for Gaussian Mixture Models. 109

Chapter 6. KALMAN FILTERS 112
6.1. Introduction. 112
6.2. Kalman Filter Equations. 112

Chapter 7. DYNAMIC PROGRAMMING. 118

Chapter 8. HIDDEN MARKOV MODELS 128
8.1. Introduction. 128
8.2. Basic concepts and notation 128
8.3. Calculating the likelihood $p(\mathbf{x}_1^T|M)$ 132
8.4. Calculating the most likely state sequence: The Viterbi algorithm 136
8.5. Training/estimating HMM parameters 136

Bibliography 141

Index 142

Uncertainties are part of life. Observations can never be perfect, and the processes we need to investigate are often so complex that it is impossible to model it with any accuracy. In this part we therefore change emphasis. Instead of modeling physical processes using the fundamental laws of physics, we now build probabilistic models directly from data obtained from observations or measurements. Fundamentally we assume that the data is generated by a specific process, reflected by patterns and regularities that we are able to learn from the data. In order to illustrate this point, let me share with you my personal discomfort in searching for patterns in stock market trends (bearing in mind that many, if not most researchers do not share this discomfort with me). The process generating the stock market data is truely complex and there is no hope to do any detailed modeling, therefore a prime candidate for the type of modeling we'll be studying. So what is the problem? Let us suppose that we have been able to extract the patterns of market movements from the data, even if there is still significant uncertainty, and start to exploit this knowledge. It is inevitable that our exploitation has an effect on the mechanism that will eventually break the process, in which case it is necessary to start learning the patterns all over again. If we don't break the system, there is the potential of gaining umlimited wealth, and that does not make sense to me. Said in a different way, what we really need to do is to be able to predict the break-downs of the mechanism, i.e. when the patterns we have learned are no longer of any use to us. In which case the same objection applies.

Be it as it may, it is important that the data we study is generated by some orderly process, producing patterns and regularities that we can identify.

In this manuscript the emphasis is on parametric modeling, where the data is used to estimate the model parameters. The model itself depends on the questions one wants to ask, i.e. on the problem one has to solve. One may, for example, be interested in knowing whether a given handwritten signature is a forgery or not, or one may want to identify the person speaking on the other side of a telephone. In yet another application you may want to identify the corresponding points in a stereo vision pair. It is also possible to repair badly damaged images by estimating the most likely pixel values, based on the information conveyed by the undamaged part of the image.

In all examples like these we proceed in the same basic way. First the probabilistic model is constructed based on all available data. This is the learning phase, and it often boils down to estimating the parameters of a model using something called maximum likelihood. Note that finding the parameters is probably the easy part. The actual modeling process is much harder, and determines the efficacy of the system. In this part we'll introduce powerful techniques, as well as a number of real-life examples, always with the understanding that proficiency in the modeling process only comes with experience.

The second part of the process is to address queries to the model. This is where we actually find the answers to the questions we would like ask, such as those mentioned above. If the modeling is done carefully, the main issue that remains is probably computational speed. If it is our task to estimate the position and velocity of an aircraft during take-off and landing as part of an automated control system, we had better get the results in real time.

As it stands the reader will not fail to recognize our debt to the book by Chris Bishop [**3**], we have freely borrowed from his images that he has generously made available on-line. Other useful references include [**1, 2, 4, 5**]

CHAPTER 1

BASIC PROBABILITY

1.1. Introduction.

One needs surprisingly little background in probability in order to follow the discussion in this part of the manuscript. Everything we do is based on just two basic rules: the sum rule, and the product rule. In this chapter we develop the essential ideas.

1.2. Discrete Probability.

Everything we do derives from a few basic rules of probability. Assuming that you are at least familiar with the basic ideas we dive right in.

We need to specify a a triple $(X, \mathcal{A}_X, \mathcal{P}_X)$, the discrete random variable, X, the set of possible values, or outcomes, or realizations of X, $\mathcal{A}_X = \{a_1, \ldots, a_I\}$, and the relative frequencies with which we observe the different values of X, $\mathcal{P}_X = \{p_1, \ldots, p_I\}$. Here p_i is the frequency or probability with which a_i observed, and we write $P(X = a_i) = p_i,\ p_i \geq 0$. In words, this says that the probability that X takes on the value a_i, $P(X = a_i)$ is equal to p_i. Since X has to take one of the values in $\mathcal{A}_X$ we need,

$$\sum_{a_i \in \mathcal{A}_\mathcal{X}} P(X = a_i) = 1. \tag{1.1}$$

The two conditions, $p_i \geq 0$, and $\sum_{i=1}^{I} p_i = 1$, ensure that that the p_i are proper probabilities.

Instead of $P(X = a_i)$ the shorthand $P(a_i)$ will often be used but make sure you understand the distinction between a random variable X, written in upper case, and its value or realization, x or a_i, written in lower case. (Often we don't make the distinction, hoping it will be clear from the context.)

If the sum in (1.1) is only over a subset T of $\mathcal{A}_X$, i.e. $T \subseteq \mathcal{A}_X$, then, noting that the values are mutually exclusive,

$$P(T) = P(X \in T) = \sum_{a \in T} P(a). \tag{1.2}$$

This is the first form of the *sum rule.*

Example 1. If one throws two six-sided, balanced dice, what is the probability that at least one 3 will show? Noting that there are 36 possible combinations, with each combination being equally likely, one has to add the probabilities of all the combinations that contain a 3. Since there are 11 possible combinations, each with a probability of 1/36, the total probability is 11/36

A *joint ensemble* XY is an ensemble in which each outcome is an ordered pair, $(x,\ y)$ with $x \in \mathcal{A}_X$ and $y \in \mathcal{A}_Y$. We call $P(X, Y)$ the joint probability of X and Y. More intuitively, $P(X = x, Y = y)$ is the probability of observing the values x and y simultaneously. For $P(X, Y)$ to be a proper probability it has to be normalized so that

$$\sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} P(x, y) = 1, \tag{1.3}$$

where the reader should note the simplification in notation.

Let us now consider the new function obtained from the joint distribution,

$$P(X) = \sum_{y \in \mathcal{A}_Y} P(X, y). \tag{1.4}$$

Because of the normalization (1.3) this function $P(X)$ is a probability distribution over X, since

$$\sum_{x \in \mathcal{A}_X} P(x) = \sum_{x \in \mathcal{A}_X} \sum_{y \in \mathcal{A}_Y} P(x, y) = 1.$$

We are therefore justified in calling $P(X)$ the probability distribution of X. This process of finding the probability distribution of a single variable from the joint probability distribution is called *marginalization.* Moreover, the marginalization equation (1.4), or the *sum rule*, is the first of the two important operations in probability theory. The second one is the product rule which we arrive at by defining the *conditional probability*.

Let us return to the joint probability $P(X,Y)$ and imagine that somehow we know, perhaps through an observation, the value of Y, say $Y = b$ to be precise. What can we say about $P(X, Y = b)$? First of all it is not a *probability* distribution since $\sum_{x\in\mathcal{A}_X} P(x, Y = b)$ does not sum to 1. In fact, the sum rule tells us that

$$\sum_{x\in\mathcal{A}_X} P(x, Y = b) = P(Y = b).$$

This means that we can normalize $P(X, Y = b)$ by dividing by $P(Y = b)$ which we obtain by a marginalization of the joint distribution, as above. The result is again a probability distribution, the *conditional* probability distribution, given by

$$P(X|Y = b) = \frac{P(X, Y = b)}{P(Y = b)}.$$

The conditional probability $P(X = a|Y = b)$ gives the probability that $X = a$ if we know that $Y = b$. We'll gradually develop a stronger intuition about conditional probabilities. Let it suffice for now to point out that the conditional probability tells us what we can learn about the probability of X, given information about Y. Also note that the conditional probability is not defined if $P(Y = b_j) = 0$. This makes sense since if the probability that $Y = b$ is zero, then it is senseless to think of a probability distribution over X given an impossible realization of Y.

We have inadvertently arrived at the product rule,

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X), \tag{1.5}$$

where the second equality derives from the fact that the joint probability is symmetric, i.e. we make no distinction between the order of events when we observe X and Y simultaneously. In In mathematical terms this means that $P(X,Y) = P(Y,X)$. Together the two factorizations lead to the important Bayes' theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \tag{1.6}$$

It is important to realize that the joint probability $P(X,Y)$ is the most fundamental of the quantities mentioned above. Knowing the joint probability one derives the marginals as well as the conditional probabilities. In fact, all our probabilistic models are in the form of a joint distribution.

We alluded to the denominator as a normalization constant (as far as Y is concerned), ensuring that the posterior probability is properly normalized. Using the sum and product rules one can also write the denominator as

$$P(X) = \sum_{y \in \mathcal{A}_Y} P(X|y)P(y). \tag{1.7}$$

For reasons that will become clear later, this is also called the *evidence*.

If $P(X|Y) = P(X)$, i.e. if the conditional probability has the same value, no matter what the values of Y, the two variables are *statistically independent*. In particular this implies that for statistically independent variables

$$\begin{aligned} P(X, Y) &= P(X|Y)P(Y) \\ &= P(X)P(Y). \end{aligned}$$

Statistical independence means that we cannot infer anything about X from knowledge about Y, and vice versa.

We noticed how the factorization simplifies if the random variables are statistically independent. This simplification has important consequences in practice – it often means the difference between being to calculate the joint distribution, or not. For this reason it is sometimes assumed that the variables are statistically independent, even when it is not the case. Although often unavoidable, statistical independence assumptions should always be made with care.

It is also true that we often rely on statistical dependence between variables in our models. In many cases, perhaps *most* cases, it is not possible to observe the state variables directly. Here 'state variables' is an abstract term that simply means those quantities that we are interested in knowing. If there is a statistical dependence between the unknown state variable a known, observed variable, then it is possible to infer something about the unknown variable from the observed variable. We'll encounter many examples in the course of our studies, but for a quick example, let us return to our crowing cock.

Imagine that we are isolated in a windowless room with no windows and no watch to tell the time of day. The only communication we have with the outside world is through sounds that are relayed through a speaker system. If we hear a cock crowing, or any other typical sounds, we can deduce something about the time of day from what we hear, since there is a statistical dependence between the time of day, and the sounds we hear, e.g. the crowing of a cock. On the other hand if we have a cock that is a little bit crazy and starts crowing at all times of the night and day, we would not be able to infer anything about the time of day from what we hear. In this case the time of day and the crowing of the cock are statistically independent.

EXAMPLE 2. Two discrete random variables x and y have a joint probability distribution $P(X,Y)$, given by the following table,

$Y=0$	0.03	0.12	0.15
$Y=1$	0.01	0.04	0.05
$Y=2$	0.06	0.24	0.30
	$X=0$	$X=1$	$X=2$

.

We first calculate the marginal probabilities $P(X)$ and $P(Y)$. Since

$$P(X) = \sum_{b\in\mathcal{A}_Y} P(X, Y=b),$$

we find $P(X)$ by adding the *columns* of the table. This gives,

0.1	0.4	0.5
$X=0$	$X=1$	$X=2$

.

Similarly, $P(Y)$ is found by adding the rows of the table, i.e. $P(Y)$ is given by,

$Y=0$	0.3
$Y=1$	0.1
$Y=2$	0.6

.

We can now calculate the conditional probability $P(X|Y)$, given in terms of the joint- and marginal distributions. This is also conveniently expressed in a probabilistic table,

$Y = 0$	0.1	0.4	0.5
$Y = 1$	0.1	0.4	0.5
$Y = 2$	0.1	0.4	0.5
	$X = 0$	$X = 1$	$X = 2$

Two observations are important. First note that the rows sum to 1, but not the columns. This clearly indicates that $P(X|Y)$ is a probability distribution as a function of X, but it is *not* a probability distribution as a function of Y. The second observation is that $P(X|Y)$ as a function of X, is independent of the value of Y, i.e. $P(X|Y) = P(X)$, and is the same as $P(X)$ calculated above. This means that X and Y are statistically independent.

It is important to note that a statistical dependence does not imply any causal relationship between the variables. Statistical dependence means that information about one variable changes the state of our knowledge of the other. For example, if we hear a cock crow, there is a distinct probability that the dawn is neigh. It does not however, imply that the dawn is *caused* by the cock crowing[1]. Although we do not study causal relationships, it is obviously of considerable interest. It is for example, very important to know whether a desired effect is actually caused by the medicine administered to the patient, or due to some other serendipitous effect. Establishing causal effects is beyond the scope of this work, see [**6**].

In the next example we use Bayes' theorem to investigate the practical efficacy of a signature verification system.

EXAMPLE 3. Suppose you have developed a signature verification system with error curves shown in Figure 1.2.1[2].

It is clear from the error curves that your system is not perfect—there does not exist a perfect system. Any system on occasion accepts a forgery, or reject a genuine signature. Suppose you have installed your system in a bank and your

[1]Let us not blame the cock; politicians also believe they make the world go round.

[2]We thank Hanno Coetzer who provided these results.

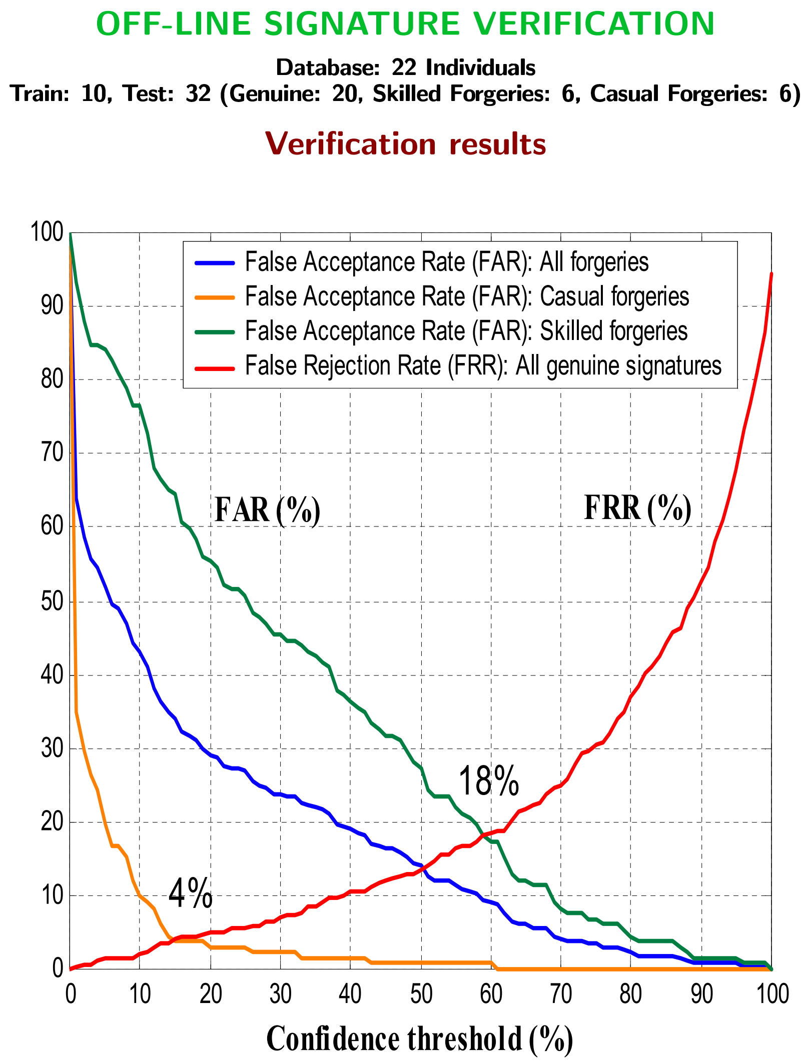


FIGURE 1.2.1. Error rates of a static signature verification system.

system rejects a signature. What is the probability that it is actually a forgery? This is an important question. If your system rejects a large percentage of genuine signatures, it is basically useless. From the Figure is is clear that you cannot expect any better than a 4% equal error rate (when the false acceptance rate equals the false rejection rate), and that is only for casual forgeries. Crime statistics are normally given as a number per one hundred thousand of the population. We don't have exact statistics of the number of forgers in a society but in countries with high crime rates, a crime rate of forty per hundred thousand of the population is typical. Use these numbers to calculate the probability that a signature is a forgery if rejected by the system. What should the error rate be before your system becomes useful, if by 'useful' you mean that every second signature that is rejected is actually a forgery?

First of all, let us introduce our random variables. Let S denote the signatures, i.e. S can take on two values, $S = t$ indicating a genuine signature, and $S = f$ a forgery. Similarly, let M indicates the measurement (the output of the system) so that $M = t$ and $M = f$ indicate a genuine signature and a forgery respectively, according to the system.

Let us start by considering what we can say about a signature without consulting your system, i.e. if we are presented with a signature how much confidence do we have that it is genuine, before we present it to your system for verification? The important thing to realize is that we do have prior knowledge about the status of a signature, knowledge that is modified by the output of your system. Using the crime statistics quoted above, it is reasonable to expect that the prior probability that it is a forgery, is $\frac{40}{100\,000}$, i.e. $P(S = f) = \frac{40}{100\,000} = 0.0004$. Without the benefit of the system we conclude that it is quite unlikely to encounter a forgery. Let us now consider how an actual measurement of your system modifies this prior belief. Suppose the system indicates a forgery, we are interested in the posterior $P(S = f|M = f)$. Using Bayes' rule we write

$$P(S = f|M = f) = \frac{P(M = f|S = f)P(S = f)}{P(M = f)}.$$

The likelihood, $P(M = f|S = f)$ is an indication of how well the system describes the signatures, and is used as a correction of our prior assumption. According to the error curves we have $P(M = f|S = f) = 0.96$, and the normalization constant

$$\begin{aligned} P(M = f) &= P(M = f|S = f)P(S = f) + P(M = f|S = t)P(S = t) \\ &= 0.96 \times 0.0004 + 0.04 \times 0.9996 \\ &= 0.0404. \end{aligned}$$

The posterior is therefore given by $P(S = f|M = f) = \frac{0.96 \times 0.0004}{0.0404} = 0.0095$, or about 0.95%. Although the likelihood has changed our prior belief, your system performs very poorly indeed—we cannot trust the system even if it does indicate a forgery, there are just too many false alarms. Can you think of the reason for this?

If one really does not have any prior idea of the prevalence of forgeries, one can use an uninformative prior $P(S = f) = 0.5$. In this case the posterior becomes

$$P(S = f|M = f) = \frac{0.96 \times 0.5}{0.96 \times 0.5 + 0.04 \times 0.5} = 0.96$$

which is just the output from the system. Using this prior really means that in the absence of any information from the system one believes that half the population forge signatures. Hopefully this is not the case.

If you find the next exercise somewhat vague, it is on purpose. We cannot do any inference without making assumptions. In this exercise we want you to make what you think are reasonable assumptions, state your assumptions, and explore the consequences of those assumptions. It will be even better if you explore the consequences of different sets of assumptions. There is always some subjective element in the assumptions we make. Once we agree on the assumptions however, we should always come to the same conclusion. In that sense, conclusions should always be objective.

Example 4. You have lost the last digit of the phone number of your friend and you decide to try a number at random and see whether you can reach your friend. Since you have a random choice between 10 different digits, the chances that you get connected is $\frac{1}{10}$. What are your chances if you allow yourself a second chance?

If you ever encounter a problem where you are not sure how to proceed it is a good idea to list all the possibilities. In this case you may want to first do it for fewer digits, say four. We go straight for ten digits.

Since you are not going to repeat the number already dialed, you may dial any of the following combinations,

$$(0,1),\ (0,2),\ldots,\ (0,9),\ (1,2),\ldots,\ (1,9),\ldots,\ (7,8),\ (7,9),\ (8,9),$$

for a total of $9+8+\cdots+1=45$ combinations. Looking at this list we see that each number appears in nine different combinations. Thus the probability of hitting the right number is $\frac{9}{45}=\frac{1}{5}$.

Let us now reason in a different way. Choosing the first number out of ten possibilities gives us a probability of $\frac{1}{10}$ of being right. For the second number we choose a different one, that is one out of nine possibilities. This has to be multiplied with the probability $(\frac{9}{10})$ that the correct number is among those nine numbers. The total probability of dialing the correct number given two tries, is therefore $\frac{1}{10}+\frac{1}{9}\frac{9}{10}=\frac{1}{5}$.

What is the probability of hitting the correct number given three tries? Note that you need to dial all ten numbers to be sure that you get the right one.

Exercise 5. We now want you to build a probabilistic model of the previous problem by writing down a joint probability. This will force you to think carefully about your modeling assumptions. Introduce three binary random variables, S and X_1, X_2. $S=1$ describes a successful connection with your friend. X_1 and X_2 describe the outcomes of two trials. For example, $P(S=1|X_1=1)=1$. From the joint probability $P(S,X_1,X_2)$ the total probability of success is given by,

$$\begin{aligned} P(S) &= \sum_{X_1,X_2\in\{0,1\}} P(S,X_1,X_2) \\ &= \sum_{X_1,X_2\in\{0,1\}} P(S|X_1,X_2)P(X_2|X_1)P(X_1). \end{aligned}$$

You are interested in $P(S=1)$. Assign values to the different terms on the right hand side, and note in particular how they derive from your modeling assumptions.

Let us think about specific assumptions. A very reasonable assumption is that you are a rational person, and having discovered that the first number is not the correct one, choose another one among the remaining ones. A less reasonable assumption

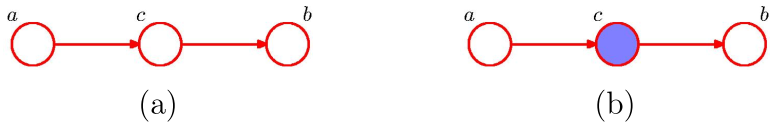


FIGURE 1.2.2. (a) a and b dependent (b) a and b independent, given c.

but perhaps not impossible, is that it simply does not occur to you to discard the wrong number, and that you select the second number again from all ten possibilities. Finally, it might just be possible that your friend has a heavy cold and that you are unable to tell, even if you have reached the right number.

Maybe you can think of more scenarios. But once you have decided on a specific scenario, you should be able to come to a definite conclusion.

1.2.1. Conditional independence.

It is interesting, and important, to note that additional knowledge may change dependencies into independencies and vice versa. The main idea is that of *conditional* independence . Variables X and Y are conditionally independent given Z, if

$$P(X, Y|Z) = P(X|Z)P(Y|Z), \tag{1.8}$$

or equivalently,

$$P(X|Y, Z) = P(X|Z) \tag{1.9}$$

also written as

$$X \amalg Y|Z.$$

EXERCISE 6. Show that $P(X, Y|Z) = P(X|Z)P(Y|Z) \iff P(X|Y, Z) = P(X|Z)$.

Let us look at few examples illustrating how knowledge about Z can influence our knowledge about the dependencies between X and Y.

EXAMPLE 7. Suppose we have a model where the joint distribution factorizes as

$$P(A, B, C) = P(B|C)P(C|A)P(A). \tag{1.10}$$

One can illustrate these dependencies as in Figure 1.2.2.

Assuming that $C = c$ is observed, it follows from the product rule and (1.10) that,

$$\begin{aligned} P(B|A, C = c) &= \frac{P(A, B, C = c)}{P(A, C = c)} \\ &= \frac{P(B|C = c)P(C = c|A)P(A)}{P(C = c|A)P(A)} \\ &= P(B|C = c). \end{aligned}$$

Thus if $C = c$ is observed in this particular model, A and B become conditionally independent.

A practical example of this scenario is, where your gnome depends on the gnomes of your grand parents. Once you know the gnome of your parents, you have all the available knowledge of your own gnome and it becomes independent of that of your grand parents.

Example 8. For this model the joint distribution factorizes as follows (see Figure 1.2.3),

$$P(A, B, C) = P(C|A, B)P(A)P(B).$$

If C is unobserved we can marginalize and write

$$\begin{aligned} P(A, B) &= \sum_c P(A, B, c) \\ &= \sum_c P(c|A, B)P(A)P(B) \\ &= P(A)P(B). \end{aligned}$$

This means that as long as C is unobserved, A and B are statistically independent. The moment C is observed we can no longer marginalize (C is locked to its observed value, $C = c$), and A and B have become conditionally dependent.

Again a practical situation is where the genomes of you and your husband are independent. The moment you know the genome of your children, you can infer something about your own genome from the genome of your husband—your genome has become linked to your husband's through knowledge of your children's.

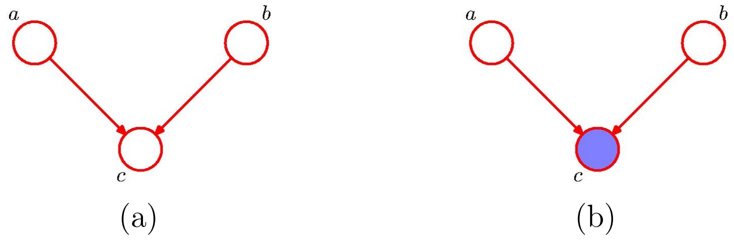


FIGURE 1.2.3. (a) a and b independent. (b) a and b dependent, given c.

The previous example is a particularly vivid illustration of the fact that there need not be any causal relationship between statistically dependent variables. The fact that you know your children's genome, does not alter the relationship between your or your wife's genome in any sense; physically they remain independent. In probability we are talking about a *logical* relationship. You can infer something about your own genome via the link through your children, by knowing your wife's and your children's genome.

EXAMPLE 9. Let us return to the signature verification example, Example 3, above. The question arises of what happens when your systems rejects a signature as a forgery? Do you immediately call security? What if it is the system that erroneously rejects the signature? In that case you have just lost what might have been a valued customer. Another approach is to verify your system by asking the customer to sign again. Let us analyze this situation.

In general if we have two signatures, we are interested in

$$P(S|M_1, M_2) = \frac{P(M_1, M_2|S)P(S)}{P(M_1, M_2)}.$$

So far we have not made any assumptions; in order to make further progress we now need to make an assumption. We assume that the two measurements, M_1 and M_2 are *conditionally* independent, given S. This means that

$$P(M_1, M_2|S) = P(M_1|S)P(M_2|S)$$

in which case we have

$$P(S|M_1, M_2) = \frac{P(M_1|S)P(M_2|S)P(S)}{P(M_1, M_2)}.$$

Using the same numerical values as in Example 3, we can now calculate the probability of the signature being a forgery, if the system indicates a forgery both time,

$$P(S = f|M_1 = f, M_2 = f) = \frac{P(M_1 = f|S = f)P(M_2 = f|S = f)P(S = f)}{P(M_1 = f, M_2 = f)}.$$

Since

$$\begin{aligned} P(M_1, M_2) &= \sum_s P(M_1, M_2, s) \\ &= \sum_s P(M_1, M_2|s)P(s) \\ &= \sum_s P(M_1|s)P(M_2|s)P(s), \end{aligned}$$

we find that

$$\begin{aligned} P(S = f|M_1 = f, M_2 = f) &= \frac{0.96 \times 0.96 \times 0.0004}{0.96 \times 0.96 \times 0.0004 + 0.04 \times 0.04 \times 0.9996} \\ &= 0.1873 \end{aligned}$$

which is about 19%. Although we still cannot be too sure, our confidence that it is a forgery has increased after the signature has been rejected twice.

This example illustrates the difference between independence and *conditional* independence. The fact that we assumed *conditional* independence

$$P(M_1, M_2|S) = P(M_1|S)P(M_2|S)$$

does *not* imply independence, i.e.

$$P(M_1, M_2) = P(M_1)P(M_2)$$

is *not* true. They have, after all, been produced by the same signatory.

Let us first give the intuitive reason. If we know S, i.e. we already know whether the signature is a forgery or not, a second measurement does not change this knowledge. It means the two measurements are conditionally independent. On the other hand if we don't know whether the signature is a forgery, then our only information comes from the measurements. Thus if the first measurement indicates a forgery, we have a reasonable expectation that the second measurement will also indicate

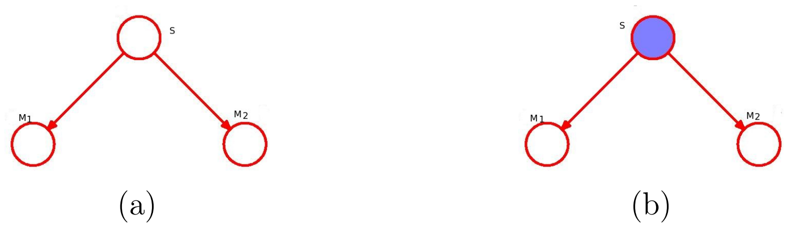


FIGURE 1.2.4. Conditional probability for Example 9.(a) S is not observed. (b) S is observed.

a forgery, because, as already pointed out, they have been produced by the same person.

Let us now discuss the formal explanation, illustrated by the appropriate graphical model. Assuming conditional independence as above, the joint distribution factorizes as follows,

$$P(M_1, M_1, S) = P(M_1|S)P(M_2|S)P(S)$$

with the graphical model given by Figure 1.2.4.

EXERCISE 10. Show that, given this graphical model, M_1 and M_2 are conditionally independent given S, and that M_1 and M_2 are not independent.

1.3. Probability Densities.

In the previous section we looked at discrete random variables. In this section we turn to continuous random variables. With this also comes a change in notation: continuous random variables are denoted by lower case; leaving it to the context in to determine whether we are working with a random variable or its realization.

Since the state space is continuous, i.e. the random variable x can assume values drawn from the real numbers, we need to replace the idea of a probability distribution. After all, the probability of drawing a specific number from a continuum is zero. Thus instead of a probability distribution over a discrete random variable, $P(X)$, we now introduce a continuous random variable x with a probability *density* function $p(x)$.

If the probability of a real-valued variable x falling in the interval $(x, x + \delta x)$ is given by $p(x)\delta x$ for $\delta x \to 0$,then $p(x)$ is called the probability density over x. The probability that x will lie in the interval (a, b) is therefore given by the sum of the probabilities (do you see any relationship with (1.2)?),

$$P(x \in (a,b)) = \int_a^b p(x)dx.$$

Note: $p(x)$ satisfies

- $p(x) \geq 0$
- $\int_{-\infty}^{\infty} p(x)dx = 1.$

The sum (marginal)– and product rules become

$$\begin{aligned} p(x) &= \int p(x,y)dy \\ p(x,y) &= p(x|y)p(y). \end{aligned}$$

Change of variables: If $x = g(y)$ we want to figure out how the density functions transform, i.e. how $p_x(x)$ is transformed to $p_y(y)$. For simplicity we only consider the case where $g(y)$ is monotonic, either increasing or decreasing. First consider the case where $g(y)$ monotonic increasing. The probability that x falls in the range $(x, x+\delta x)$ has to be the same as the probability that y falls in the range $(y, y+\delta y)$, where $g(y)$ maps $(y, y+\delta y)$ to $(x, x+\delta x)$. Accordingly we require that $p_x(x)\delta x = p_y(y)\delta y$, or

$$\begin{aligned} p_y(y) &= p_x(x)\frac{dx}{dy} \\ &= p_x(g(y))g'(y). \end{aligned}$$

If $g(y)$ is monotonic decreasing then $(y, y+\delta y)$ is again mapped to $(x, x+\delta x)$ but in this case $x + \delta x < x$ if $\delta y > 0$. We therefore require that $-p_x(x)\delta x = p_y(y)\delta y$, or

$$\begin{aligned} p_y(y) &= -p_x(x)\frac{dx}{dy} \\ &= -p_x(g(y))g'(y). \end{aligned}$$

Since $g(y)$ is monotonic decreasing, $\frac{dx}{dy} < 0$. Combining the two results, it follows that for monotonic functions,

$$\begin{aligned} p_y(y) &= p_x(x)\left|\frac{dx}{dy}\right| \\ &= p_x(g(y))\left|g'(y)\right|. \end{aligned}$$

Example 11. A transformation function of particular importance is given by

$$x = g(y) = \int_0^y p_y(r)dr, \tag{1.1}$$

with

$$\begin{aligned} \frac{dx}{dy} &= \frac{dg}{dy} \\ &= p_y(y). \end{aligned}$$

The transformed density function therefore becomes,

$$\begin{aligned} p_x(x) &= p_y(y)\left|\frac{dy}{dx}\right| \\ &= p_y(y)\left|\frac{1}{p_y(y)}\right| \\ &= 1. \end{aligned}$$

Let us say you have available to you a method that draws unbiased random samples from a uniform distribution, this transformation then allows you to draw unbiased random samples from any other density function.

In image processing this can be used to enhance the contrast in an image, through histogram equilization. Given a histogram $p_y(y)$ (normalized so that $\sum_y p_y(y) = 1$) the idea is to transform it so that $p_x(x) = 1$. This is precisely the transformation given above. If the image has L gray levels, i.e. y is the discrete variable with values $y_k,\ k = 0, \ldots, L-1$, and there are n pixels in the image, then $p_y(y_k) = \frac{n_k}{n}$ where n_k is the number of pixels that have gray level y_k. The discrete version of the transform

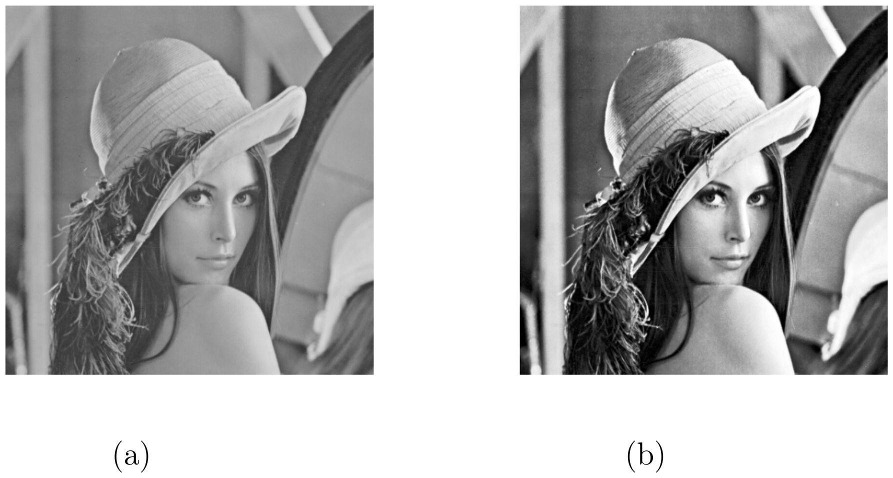


FIGURE 1.3.1. (a) Original image. (b) After histogram equalization.

(1.1) is then given by

$$\begin{aligned} x_k &= g(y_k) \\ &= \sum_{r=0}^{k} p_y(r) \\ &= \sum_{r=0}^{k} \frac{n_r}{n}, \; k = 0, \ldots, L-1. \end{aligned}$$

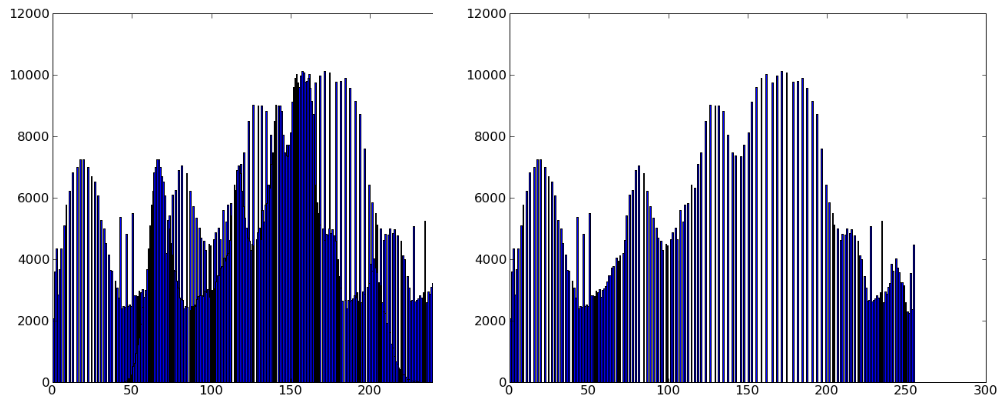
The transformed image is obtained by mapping each pixel with gray level y_k to a pixel with gray level x_k. The result is shown in Figure 1.3.1. Note how the contrast is enhanced by the histogram equalization.

The histograms are given by the following Figure. Do you have any explanation why the histogram after equalization does not show a flat curve?

1.4. Expectation and Covariances.

Discrete expectation of a function $f(X)$ of a random variable is defined as,

$$\mathbb{E}[f] = \sum_x P(x) f(x).$$



(a) (b)

FIGURE 1.3.2. (a) Histogram of original image. (b) Histogram of equalized image.

Similarly the *Continuous expectation* is defined as,

$$\mathbb{E}[f] = \int p(x)f(x)dx.$$

The expectation that is most often used, is the expectation of the random variable itself, i.e. $f(X) = X$, for the discrete case, and similarly for continuous variables. In this case we have,

$$\mathbb{E}[x] = \sum_x P(x)x,$$

or

$$\mathbb{E}[x] = \int p(x)xdx.$$

These are of course, the means of the random variables.

EXAMPLE 12. Let us revisit the probability table for the joint variables X and Y, given by

$Y = 0$	0.03	0.12	0.15
$Y = 1$	0.01	0.04	0.05
$Y = 2$	0.06	0.24	0.30
	$X = 0$	$X = 1$	$X = 2$

.

The mean of (X, Y) is given by,

$$\begin{aligned}\mathbb{E}[X,Y] &= \sum_{x,y} P(X=x,Y=y)(X=x,Y=y)\\ &= 0.03\times(0,0)+0.12\times(1,0)+0.15\times(2,0)+\\ &\quad 0.01\times(0,1)+0.04\times(1,1)+0.05\times(2,1)+\\ &\quad 0.06\times(0,2)+0.24\times(1,2)+0.3\times(2,2)\\ &= (1.4,1.3).\end{aligned}$$

If we have N points $x_n,\ n=1,\ldots,N$ drawn from the probability density $p(x)$ the expectation can be approximated by

$$E[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n). \tag{1.1}$$

One way to think about this is to assume that all the samples are equally likely and that these are the outcomes of the random variable, i.e. $P(X=x_n)=\frac{1}{N}$. If we plug this into the expression for the discrete expectation, we immediately arrive at (1.1). This approximation becomes exact for $N\to\infty$. What we are actually doing is to approximate $p(x)$ with samples drawn from it. There are different strategies for doing this, see for example Chapter 11 of Bishop's book [**3**].

Conditional expectation:

$$\mathbb{E}[f|y] = \sum_{x} P(x|y)f(x).$$

Variance:

$$\mathrm{var}[f] = \mathbb{E}[(f(x)-\mathbb{E}[f(x)])^2].$$

Covariance:

$$\begin{aligned}cov[x,y] &= \mathbb{E}_{x,y}\left[\{x-\mathbb{E}[x]\}\{y-\mathbb{E}[y]\}\right]\\ &= \mathbb{E}_{x,y}[xy]-\mathbb{E}[x]\mathbb{E}[y].\end{aligned}$$

For vector variables this becomes

$$\text{cov}[\mathbf{x},\mathbf{y}] = \mathbb{E}_{x,y}\left[\{\mathbf{x}-\mathbb{E}[\mathbf{x}]\}\{\mathbf{y}-\mathbb{E}[\mathbf{y}]\}^T\right] = \mathbb{E}_{x,y}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T].$$

A particularly important case of the covariance matrix is $\text{cov}\,[\mathbf{x},\mathbf{x}]$. If $\mathbf{x}$ is a d-dimensional vector then $\text{cov}\,[\mathbf{x},\mathbf{x}]$ is a $d\times d$, symmetric, positive semi-definite matrix, given by

$$\text{cov}[\mathbf{x},\mathbf{x}] = \mathbb{E}\left[(\mathbf{x}-\mathbb{E}[\mathbf{x}])(\mathbf{x}-\mathbb{E}[\mathbf{x}])^T\right] = \mathbb{E}_{x,y}[\mathbf{x}\mathbf{x}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}^T].$$

Thus, its eigenvalues are real and non-negative.

In practice we seldom know the probability density function $p(\mathbf{x})$. In fact the learning part of machine learning is to estimate or approximate the pdf from data. In general we observe samples that we assume are generated by the pdf, from these samples we have to estimate, ideally the pdf, or perhaps the mean and covariance. We have already given the expression for the sample estimate for the mean. The (biased) sample estimate for the covariance matrix, given N samples $\mathbf{x}_n, n = 1,\ldots,N$ is

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)(\mathbf{x}_n-\mu)^T,$$

where μ is the mean.

Exercise 13. Show that the sample covariance matrix Σ is symmetric and positive semi-definite, i.e. that it is symmetric and all its eigenvalues are non-negative.

1.5. Decision Theory.

In the signature verification example we referred to a decision boundary that allows one to accept a signature, or reject it as a forgery. The question is how does one find the optimal decision boundary. We classify a given signature x into one of two classes $C_k,\ k = 0,1$ and we are interested in the probability of class C_k given a signature x, i.e. we are interested in $P(C_k|x)$. Using Bayes theorem we can write this as

$$P(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)}.$$

The prior probability $P(C_k)$ tells us the probability of class C_k in the absence of any further information. It simply tells us how likely it is that any given signature is a forgery. This represents our (subjective) belief in the honesty of the population and may for example be based on crime statistics. $p(C_k|x)$ is the revised, posterior estimate, based on the actual measurement x. The class-based likelihood $p(x|C_k)$ requires knowledge about our system.

We want to minimize the chances of assigning x to the wrong class. Intuitively we assign x to the class with the highest posterior probability. Let us make this precise.

1.5.1. Minimizing misclassification. Two types of errors are possible, accepting a forgery, or rejecting a genuine signature. In general an input vector belonging to C_0 can be wrongly assigned to class C_1, or vice versa. The probability of a mistake is therefore given by

$$P(\text{mistake}|x) = \begin{cases} P(C_0|x) & \text{if we choose } C_1 \\ P(C_1|x) & \text{if we choose } C_0 \end{cases}. \tag{1.1}$$

It may be even better to minimize the probability of a mistake, averaged over all possible input values, i.e. minimize the marginal probability,

$$\begin{aligned} P(\text{mistake}) &= \int P(\text{mistake}, x)dx \\ &= \int_{R_0} P(\text{mistake}, x)dx + \int_{R_1} P(\text{mistake}, x)dx \\ &= \int_{R_0} P(\text{mistake}|x)p(x)dx + \int_{R_1} P(\text{mistake}|x)p(x)dx \end{aligned}$$

where R_k is the region where all input vectors x are assigned to C_k, see Figure 1.5.1. Making use of (1.1) this can be written as

$$\begin{aligned} P(\text{mistake}) &= \int_{R_0} P(C_1|x)p(x)dx + \int_{R_1} P(C_0|x)p(x)dx \\ &= \int_{R_0} p(x|C_1)P(C_1)dx + \int_{R_1} p(x|C_0)P(C_0)dx \end{aligned}$$

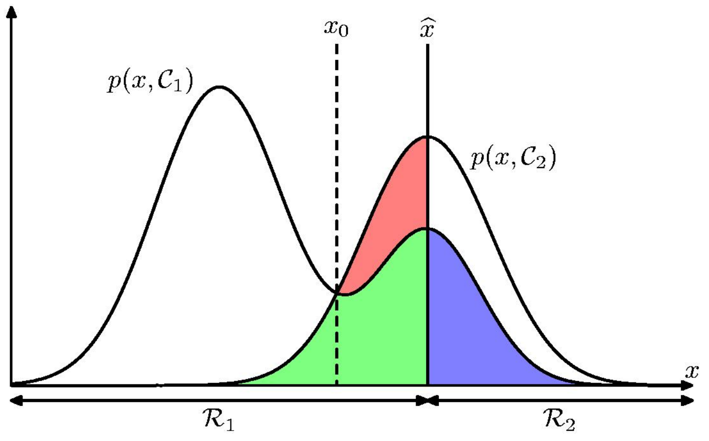


FIGURE 1.5.1. Optimal decision boundary.

where the last line follows from Bayes' theorem. This is minimized if x is assigned to the region for which the integrand is the smallest, as illustrated in Figure 1.5.1.

Note that it is not possible to make an informed decision in the absence of knowledge of our system. In fact, following this approach we need to know quite a lot: essentially we need to know the joint densities $p(x, C_k)$. The 'learning' part of machine learning is about inferring joint densities or class conditional densities from the data. Signature verification is hard because we do not have a model for forgeries—a forgery is everything that is not a genuine signature. The reason why it is at all possible to design efficient systems is because we are able to design a good models for the genuine signatures.

1.5.2. Minimizing expected loss.

If there is a penalty involved in a particular misclassification it can be modeled by a loss function. Commercial banks for instance would rather accept more forgeries than alienating their customers by rejecting genuine signatures—there is a heavy penalty for rejecting a genuine signature.

Let us generalize a little and assume that there are N different classes (or states of nature), and a different actions $\alpha_1, \ldots, \alpha_N$ every time we observe x. Action α_j for example, may indicate: put x in class C_j. The posterior is given as usual by

$$P(C_j|x) \propto p(x|C_j)P(C_j).$$

Let λ_{ij} expresses the penalty for taking action α_i when the true state is C_j. The loss or penalty for taking action α_i having observed x, is the average penalty

$$R(\alpha_i|x) = \sum_{j=1}^{N} P(C_j|x)\lambda_{ij}.$$

Our decision rule is a function $\alpha(x)$ that maps each x to an action α_i. The average penalty is minimized if, for each x choose

$$\alpha(x) = \arg\ \min_{\alpha_i} R(\alpha_i|x).$$

Note that this also minimizes the overall risk, defined as the expected loss for a given decision rule,

$$R = E\left[R(\alpha(x)|x)\right] = \int R(\alpha(x)|x)p(x)dx.$$

1.5.3. Reject option. If it is essential that a correct classification be made, one might consider a reject option for samples falling in the ambiguous region close to the decision boundary.

1.5.4. Inference and decision. Instead of two different stages, inference and decision, one might opt for a *discriminant* function. Three approaches

(1) Calculate the posterior density $p(C_K|x)$ from the class conditional densities $p(x|C_k)$. Then use decision theory, i.e. incorporate the possible risk. This is a so-called *generative* model. The name stems from the fact that, since one builds a probabilistic model of each class, it is possible to use the model to generate data by drawing samples from the probability distributions. The main drawback of this approach is the danger of overkill. We may not be interested in a detailed model of each class, in which case it might be better to use the available data to directly achieve your aim.
(2) Find $p(C_k|x)$ directly, then use decision theory. This is a so-called *discriminative* model. Instead of using the data to build a model of each class, the available data is used to estimate parameters that directly gives the probability of class membership. Note that any penalty funtions are built into the model.

(3) Find a function $f(x)$ that maps x directly onto a class label. This is perhaps the simplest approach. The main drawback is that one does not get a confidence value with the estimate and therefore has no indication to what extent the class assignment can be trusted.

Combining models. Suppose we have two independent test of a quantity, for example using both a signature and fingerprint for personal identification. If the two tests are (conditionally) independent we write

$$p(x_1, x_2|C_k) = p(x_1|C_k)p(x_2|C_k).$$

the posterior density is then given by

$$\begin{aligned} p(C_k|x_1, x_2) &\propto p(x_1, x_2|C_k)p(C_k) \\ &= p(x_1|C_k)p(x_2|C_k)p(C_k) \\ &\propto \frac{p(C_k|x_1)p(C_k|x_2)}{p(C_k)}. \end{aligned}$$

It is always a good idea to use different, independent models. Note that this also works if one obtains a second conditionally independent measurement using the same model.

CHAPTER 2

PROBABILITY DENSITY FUNCTIONS

2.1. Introduction.

We introduce various probability density functions, the most important one is the Gaussian or normal density function. It does show up regularly in practical situations but its main attraction is its analytical properties. Operations that are otherwise intractable can be evaluated analytically in case of Gaussian.

2.2. Binary Variables.

2.2.1. The Bernoulli distribution. If $X \in \{0, 1\}$ is a discrete random variable such that $P(X = 1) = \mu$, implying $P(X = 0) = 1 - \mu$, then the probability distribution $P(X|\mu)$ is given by the Bernoulli distribution,

$$P(X|\mu) = \mathrm{Bern}(X|\mu) = \mu^X(1-\mu)^{1-X}. \tag{2.1}$$

Exercise 14. Show that $\mathbb{E}[X] = \mu$, and $\mathrm{var}[X] = \mu(1-\mu)$.

The goal is to estimate μ from data, $\mathcal{D} = \{x_1, \dots, x_N\}$. If we draw the samples independently, the likelihood function of μ is given by

$$P(\mathcal{D}|\mu) = \prod_{n=1}^{N} P(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}. \tag{2.2}$$

If we observe $X = 1$, m times, then the likelihood becomes

$$P(\mathcal{D}|\mu) = \mu^m(1-\mu)^{N-m}.$$

Maximizing the log-likelihood,

$$\ln P(\mathcal{D}|\mu) = m \ln \mu + (N - m) \ln(1 - \mu),$$

gives

$$\mu_{ML} = \frac{m}{N}.$$

Note: If $N = 3$ and we observe $x = 1$ for all three samples, then the maximum likelihood estimate gives $\mu_{ML} = 1$, which is nonsense. The problem is that we are working with a point estimate of μ, and not a probability distribution over μ. We return to this problem later when we introduce a prior distribution over μ.

2.2.2. The binomial distribution. If we observe $x = 1$, m times out of N observations, it follows from (2.2) that the probability of that sequence is $\mu^m(1-\mu)^{N-m}$. Since there are $\begin{pmatrix} N \\ m \end{pmatrix} = \frac{N!}{(N-m)!m!}$ ways of getting m ones out of a sequence of N trials, the total total probability of getting m ones out of a sequence of N trials, is given by the *binomial* distribution

$$\text{Bin}(m|N,\mu) = \begin{pmatrix} N \\ m \end{pmatrix} \mu^m (1-\mu)^{N-m}. \tag{2.3}$$

Note that $\sum_{m=0}^{N} \text{Bin}(m|N,\mu) = 1$.

It can be shown that

$$\begin{aligned} \mathbb{E}[m] &= N\mu \\ \text{var}[m] &= N\mu(1-\mu). \end{aligned}$$

The following example asks questions based directly on the data that is observed. The answers are therefore obtained directly from the data.

Example 15. An urn contains K balls, of which B are black and $W = K - B$ are white. You draw a ball at random, and replace it, N times. Let us calculate the probability distribution of the number of times a black ball is drawn n_B, as well as the expectation and variance of n_B,

$$P(n_B|\mathcal{D}) = \begin{pmatrix} N \\ n_B \end{pmatrix} \mu^{n_B}(1-\mu)^{N-n_B}$$

where $\mu = \frac{B}{K}$,

$$\mathbb{E}[n_B] = N\mu \text{ and } \text{var}[n_B] = N\mu(1-\mu).$$

You might also, for example, be interested in the quantity

$$z = \frac{(n_B - N\mu)^2}{N\mu\,(1-\mu)}.$$

Since n_B is a random variable, z is also a random variable and we can calculate its expected value,

$$\mathbb{E}\,[z] = \frac{\mathbb{E}\left[(n_B - N\mu)^2\right]}{N\mu\,(1-\mu)} = 1$$

since the numerator is the variance of n_B.

In the next example we are still using a generative model, i.e. we again use a model that describes a process that generates data, but this time we ask questions about latent quantities, i.e. quantities not directly observed. For this Bayes' theorem is invariably required.

Example 16. We now have eleven urns $\mathcal{C}_j, \;\; j = 0, \ldots, 10$, each one containing ten balls, either white or black. Suppose the jth urn contains j black balls, and $10 - j$ white balls. One urn is now selected at random and N balls are drawn from it, with replacement. You don't know which urn is selected but you know that n_B black balls, and $N - n_B$ white balls are drawn. What is the probability that the balls are drawn from urn $\mathcal{C}_u$?

Each urn can be modeled as,

$$P(n_B|\mathcal{C}_j) = \left(\begin{array}{c} N \\ n_B \end{array}\right) \mu_j^{n_B}\,(1-\mu_j)^{N-n_B}\,, \;\; j = 0, \ldots, 10,$$

where $\mu_j = \frac{j}{10}$. In this case however, the information we need is not directly accessible from the data. The question is, given n_B, what is the probability of $\mathcal{C}_u$, i.e. what is $P(\mathcal{C}_u|n_B)$? Using Bayes' theorem we can invert the posterior to get

$$P(\mathcal{C}_u|n_B) = \frac{P(n_B|\mathcal{C}_u)P(\mathcal{C}_u)}{P(n_B)}.$$

The prior probability $P(\mathcal{C}_u) = \frac{1}{11}$, and the marginal,

$$P(n_B) = \sum_j P(n_B|\mathcal{C}_j)P(\mathcal{C}_j).$$

EXERCISE 17. Assuming that in the previous problem $N = 10$ and $n_B = 3$, calculate $P(\mathcal{C}_u|n_B)$ for $u = 0, \ldots, 10$. Now you draw another ball from the same urn. What is the probability that the next ball is black? Hint: You need to calculate $P(\text{ball}_{N+1} \text{ is black}|n_B, N)$. Write this as the marginal

$$P(\text{ball}_{N+1} \text{ is black}|n_B, N) = \sum_u P(\text{ball}_{N+1} \text{ is black}, \mathcal{C}_u|n_B, N).$$

Factorize this in the usual way and note that

$$P(\text{ball}_{N+1} \text{ is black}|\mathcal{C}_u, n_B, N) = P(\text{ball}_{N+1} \text{ is black}|\mathcal{C}_u).$$

This means that ball_{N+1} is black is conditionally independent of n_B and N, given that we know it is drawn from urn $\mathcal{C}_u$.

Please note carefully the reasoning in this problem—by marginalizing you are taking the average over all urns. Let us see how your answer differs if instead, you simply use the fact that $n_B = 3$, to determine the most likely urn, in this case urn $u = 3$. Drawing the next ball from urn $u = 3$, gives a probability of 0.3 of drawing a black ball. This answer is not as good as the one you calculated above, because it does not take the uncertainty of the urn into account.

2.2.3. The beta distribution. Given the data $\mathcal{D}$ in (2.2) we calculated the maximum likelihood estimate of μ using the likelihood $p(\mathcal{D}|\mu)$. We noted that this can run into problems for a small number of samples. Let us therefore examine the possibility of maximizing the posterior probability $p(\mu|\mathcal{D})$ instead. This also has a more philosophical interpretation. Instead of the point estimate we get from maximizing the likelihood, we now end up with a probability distribution over μ, given the data. In order to do so a 'subjective' element is introduced in the form of a prior over μ. To wit, according to Bayes' theorem the posterior is proportional to $p(\mathcal{D}|\mu)p(\mu)$, where $p(\mu)$is the prior. The question is how to choose a suitable prior—the subjective element. In the absence of other considerations, it is *convenient* to choose it in such a way so that the posterior distribution has the same functional form as the prior, referred to as a *conjugate* prior. A suitable conjugate prior is the beta distribution

$$\text{Beta}(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{2.4}$$

where the gamma function is defined as

$$\Gamma(x) = \int_0^\infty u^{x-1}e^{-u}du,$$

and

$$\begin{aligned}\mathbb{E}[\mu] &= \frac{a}{a+b} \\ \text{var}[\mu] &= \frac{ab}{(a+b)^2(a+b+1)}.\end{aligned}$$

Given that $x = 1$ is observed m times in a trial of N samples, and setting $l = N - m$ (the number of times $x = 0$ is observed) the posterior is proportional to

$$p(\mu|m, l, a, b) \propto \mu^{m+a-1}(1-\mu)^{l+b-1}.$$

This is another (unormalized) beta distribution and properly normalized (using (2.4)) the posterior becomes,

$$p(\mu|m, l, a, b) = \frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)}\mu^{m+a-1}(1-\mu)^{l+b-1}.$$

Starting with the prior note how it is modified by the subsequent observations: a is increased by the number of times $X = 1$ is observed, and b is increased by the number of times $X = 0$ is observed. The effect of the prior is as if we had a certain number of prior observations of $X = 0$ and $X = 1$. This prior 'belief' is then modified in view of the actual observations received.

Note (important): The posterior can be used as a prior if subsequent observations are made. This allows for sequential estimates.

Let us be a little more precise and compute the probability of observing $X = 1$ in view of the data $\mathcal{D}$, by marginalizing over the joint distribution $P(X = 1, \mu|\mathcal{D})$,

$$P(X = 1|\mathcal{D}) = \int_0^1 P(X = 1|\mu)p(\mu|\mathcal{D})d\mu = \int_0^1 \mu p(\mu|\mathcal{D})d\mu = \mathbb{E}[\mu|\mathcal{D}].$$

Using the mean of the beta distribution, we get

$$P(X = 1|\mathcal{D}) = \frac{m+a}{m+a+l+b}.$$

The interpretation of the prior is now clear: one can think of a and b as fictitious observations prior to any actual observations (of $X = 1$, and $X = 0$ respectively).

The probability of observing observing $X = 1$ is the fraction of the total number of times $m + a$, including the prior 'observations', out of a total of $m + a + l + b$ observations, again including the prior 'observations'.

It is important to note the shift in point of view: when we wrote down the Bernoulli distribution, all the way through to the maximum likelihood estimate of μ from the binomial distribution, μ was just a parameter with $P(X = 1|\mu) = \mu$. Having introduced the prior, we get a probability distribution over μ. μ has become a random variable itself, and in order to find $P(X = 1|\mathcal{D})$ it has become necessary to marginalize over μ, as we did above.

Finally note that our estimate of $P(X = 1|\mathcal{D})$ (using a prior) agrees with the maximum likelihood estimate as $m,\ l \to \infty$.

EXERCISE 18. Suppose that you observe $X = 1$ three times out of three trials. Using maximum likelihood, we find that $\mu = 1$. Introducing a prior, investigate how the estimate $P(X = 1|\mathcal{D})$ is modified by the presence of data. Suppose you have reason to believe that your coin is balanced, how do you assign values to the parameters of the prior? Suppose this prior belief turns out to be wrong, how does subsequent observations correct this mistaken prior belief? Any idea how many observations are required in order to correct a mistaken prior belief?

2.3. Multinomial Variables.

Let us rewrite the Bernoullim distribution in a form that makes obvious how to generalize to more dimensions. Recall that X is a binary variable that can take on the binary values $\{0, 1\}$. If $P(X = 1) = \mu$ the Bernoulli distribution is given by,

$$P(X) = \mu^X (1 - \mu)^{1-X}.$$

Suppose that we introduce a vector variable $\mathbf{X} = [X_1, X_2]$ where each component X_1 and X_2 take on the binary values $\{0, 1\}$, with $P(X_1 = 1) = \mu_1$ and $P(X_2 = 1) = \mu_2$ with $\mu_1 + \mu_2 = 1$ so that the distribution is properly normalized. Thus $\mathbf{X}$ can take on one of two mutually exlusive states, $[x_1, x_2] = [1, 0]$ or $[x_1, x_2] = [0, 1]$, and the

Bernoulli distribution is rewritten as,

$$P(\mathbf{X}|\mu) = \prod_{k=1}^{2} \mu_k^{x_k}$$

Variables that can take on one of K mutually exclusive states can be represented by a K-dimensional vector of the form,

$$\mathbf{x} = [0\ \cdots\ 0\ 1\ 0\ \cdots\ 0]^T$$

where the position of the 1 indicates the specific state. If $P(x_k = 1) = \mu_k$, then the multinoulli distribution is given by,

$$P(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]^T,\ \mu_k \geq 0$ and $\sum_k \mu_k = 1$.

Note:

$$\sum_{\mathbf{x}} P(\mathbf{x}|\boldsymbol{\mu}) = \sum_{k=1}^{K} \mu_k = 1$$

and

$$\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu})\mathbf{x} = [\mu_1, \ldots, \mu_K]^T.$$

For a data set $\mathcal{D}$ consisting of N independent observations, the likelihood is,

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mu_k^{x_{nk}} = \prod_{k=1}^{K} \mu_k^{(\sum_n x_{nk})} = \prod_{k=1}^{K} \mu_k^{m_k},$$

where

$$m_k = \sum_n x_{nk},$$

the number of observations of $x_k = 1$. Maximizing the log-likelihood $\ln p(\mathcal{D}|\boldsymbol{\mu})$ subject to the constraint $\sum_k \mu_k = 1$, is achieved through a Lagrange multiplier, i.e. by maximizing

$$\sum_{k=1}^{K} m_k \ln \mu_k + \lambda \left(\sum_{k=1}^{K} \mu_k - 1 \right).$$

Setting the partial derivative with respect to μ_k equal to zero, gives

$$\mu_k = -\frac{m_k}{\lambda},$$

and the constraint gives

$$\lambda = -N.$$

Thus

$$\mu_k^{ML} = \frac{m_k}{N},$$

and the multinomial distribution is given by

$$\mathrm{Mult}(m_1, \ldots, m_K|\boldsymbol{\mu}, N) = \left(\begin{array}{c} N \\ m_1 \cdots m_K \end{array} \right) \prod_{k=1}^{K} \mu_k^{m_k},$$

where

$$\sum_{k=1}^{K} m_k = N.$$

2.3.1. Dirichlet distribution. The Dirichlet distribution is the conjugate prior for the multinomial distribution

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1}$$

where

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k,$$

and

$$\sum_{k=1}^{K} \mu_k = 1.$$

2.4. Model comparison.

It is not always clear what underlying process generates the observed data. There might be competing hypotheses and one would like a systematic procedures to make an informed decision.

Suppose we observe N trials of a binary process, i.e. we are given a sequence $\mathbf{s}$ of N zero's and ones. One hypothesis, let us call it $\mathcal{H}_1$, is that the sequence is generated by a biased coin. We are not given the bias and need to estimate it from the data. We also want to predict the outcome of the next trial. This is of course the problem already encountered in Section 2.2.

The probability of observing m ones and $n = N - m$ zero's is given by

$$P(\mathbf{s}|\mu, N, \mathcal{H}_1) = \mu^m(1-\mu)^n,$$

where the dependence on our hypothesis is explicitly stated. Let us also assume a uniform prior, $p(\mu|\mathcal{H}_1) = 1$. There is nothing too special about this choice—the choice of prior is part of our assumptions and we cannot do inference without assumptions. Here we make this choice more as a matter of convenience; we might have chosen a conjugate prior, the beta distribution. We want to infer μ where $P(X = 1|\mu, \mathcal{H}_1) = \mu$, and predict the outcome of the next trial. Again it might be useful to note that, since predictions are never certainties, they are always expressed in terms of probabilities. Assuming that $\mathcal{H}_1$ is true, the posterior probability is given by

$$\begin{aligned} p(\mu|\mathbf{s}, N, \mathcal{H}_1) &= \frac{P(\mathbf{s}|\mu, N, \mathcal{H}_1)p(\mu|\mathcal{H}_1)}{P(\mathbf{s}|N, \mathcal{H}_1)} \\ &= \frac{P(\mathbf{s}|\mu, N, \mathcal{H}_1)}{P(\mathbf{s}|N, \mathcal{H}_1)}, \end{aligned} \tag{2.1}$$

using the uniform prior. Since the likelihood of the parameter μ is known, the posterior is given by

$$p(\mu|\mathbf{s}, N, \mathcal{H}_1) = \frac{\mu^m(1-\mu)^n}{P(\mathbf{s}|N, \mathcal{H}_1)},$$

where the normalization constant (also called the *evidence*, for reasons that will become clear shortly) is given by

$$\begin{aligned} P(\mathbf{s}|N, \mathcal{H}_1) &= \int_0^1 \mu^m(1-\mu)^n d\mu \\ &= \frac{\Gamma(m+1)\Gamma(n+1)}{\Gamma(m+n+2)} \\ &= \frac{m!n!}{(m+n+1)!}. \end{aligned}$$

Given our hypothesis and the observed data, the posterior probability distribution of μ is therefore given by

$$p(\mu|\mathbf{s}, N, \mathcal{H}_1) = \frac{(m+n+1)!}{m!n!}\mu^m(1-\mu)^n.$$

EXERCISE 19. Recall that the maximum likelihood estimate gives $\mu = \frac{m}{N}$. Calculate the most probable value of μ from the posterior, i.e. calculate the value of μ that maximizes the posterior (this is also called the maximum a posteriori (MAP) estimate). What is the mean of μ under this distribution? How does it compare with the maximum likelihood estimate?

Turning to prediction, we now calculate the probability $P(X=1|\mathbf{s}, N)$. The sum rule gives,

$$\begin{aligned} P(X=1|\mathbf{s}, N, \mathcal{H}_1) &= \int_0^1 p(X=1, \mu|\mathbf{s}, N, \mathcal{H}_1)d\mu \\ &= \int_0^1 P(X=1|\mu, \mathbf{s}, N, \mathcal{H}_1)p(\mu|\mathbf{s}, N, \mathcal{H}_1)d\mu. \end{aligned}$$

Since the probability of observing $X=1$ is μ, the prediction for observing $X=1$ is given by

$$\begin{aligned} P(X=1|\mathbf{s}, N, \mathcal{H}_1) &= \int_0^1 \mu\frac{(m+n+1)!}{m!n!}\mu^m(1-\mu)^n d\mu \\ &= \frac{(m+n+1)!}{m!n!}\int_0^1 \mu^{m+1}(1-\mu)^n d\mu \\ &= \frac{(m+n+1)!}{m!n!}\frac{(m+1)!n!}{(m+n+2)!} \\ &= \frac{m+1}{m+n+2}. \end{aligned}$$

Now suppose we have a second hypothesis $\mathcal{H}_0$ of how the sequence $\mathbf{s}$ is generated. Suppose this second hypothesis states that the sequence is generated by casting a normal, six-sided dice, with five sides painted 'zero' and one side painted 'one'. How do we decide between these two alternative hypotheses, is $\mathcal{H}_0$ more probable that $\mathcal{H}_1$?

Also note that $\mathcal{H}_0$ does not have any free parameters whereas $\mathcal{H}_1$ has one, namely $\mu \in [0,1]$.

Using Bayes' theorem again,

$$P(\mathcal{H}_1|\mathbf{s},N) = \frac{P(\mathbf{s}|N,\mathcal{H}_1)P(\mathcal{H}_1)}{P(\mathbf{s}|N)}$$

and

$$P(\mathcal{H}_0|\mathbf{s},N) = \frac{P(\mathbf{s}|N,\mathcal{H}_0)P(\mathcal{H}_0)}{P(\mathbf{s}|N)}.$$

The normalization constant is the same for both models, and if we only consider these two models it is given by

$$P(\mathbf{s}|N) = P(\mathbf{s}|N,\mathcal{H}_0)P(\mathcal{H}_0) + P(\mathbf{s}|N,\mathcal{H}_1)P(\mathcal{H}_1).$$

In order to evaluate the posterior probabilities we need to assign the prior probabilities $P(\mathcal{H}_0)$ and $P(\mathcal{H}_1)$. Without any additional information we might set both to $\frac{1}{2}$. The two data dependent terms indicate how much the data favor the two hypotheses; this is also referred to as the *evidence* for the model. Note that it appeared as the normalization constant in (2.1)when we inferred μ from the data.

The evidence for model $\mathcal{H}_0$ is very simple because it has no free parameters

$$P(\mathbf{s}|N,\mathcal{H}_0) = \mu_0^m(1-\mu_0)^n$$

where $\mu_0 = \frac{1}{6}$. The ratio of the two models becomes,

$$\begin{aligned} \frac{P(\mathcal{H}_1|\mathbf{s},N)}{P(\mathcal{H}_0|\mathbf{s},N)} &= \frac{P(\mathbf{s}|N,\mathcal{H}_1)}{P(\mathbf{s}|N,\mathcal{H}_0)} \\ &= \frac{m!n!}{(m+n+1)!}/\mu_0^m(1-\mu_0)^n. \end{aligned}$$

EXERCISE 20. Suppose you observe the following sequence, which one of the two models is the more probable,

01110001010011100101011010010010010100110101101100010?

It should be no surprise that the model $\mathcal{H}_1$with the one free parameter is favored over the model $\mathcal{H}_0$ with no free parameters. The reason is that free parameters can be adjusted to favor the data at least to some extent. Such models are therefore seldom totally unlikely.

EXAMPLE 21. A tale of three prisoners.

In a cruel, far-away country three prisoners find themselves in a cell awaiting their lot. Two of them will be set free the next morning but the third one will go free. The warden draws the name of the prisoner to be executed from a uniform distribution, i.e. each prisoner has a $\frac{1}{3}$ chance of being selected for execution. The warden draws the name but it will only be announced the next morning.

The first prisoner quite agitated, reasons that at least one of the other two prisoners will be released, and he begs the warden to tell him the name of one of the others that will be released. The warden gives him a name. Then the first prisoner realizes that he might have made a mistake because, according to his reasoning reasoning, there are only two of them left, with equal probabilities of being executed. His chances of being executed has just gone up from $\frac{1}{3}$ to $\frac{1}{2}$. Or is he wrong?

Let us introduce three different hypotheses, $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ where $\mathcal{H}_i$ is the hypothesis that prisoner $i, \quad i = 1,\ 2\ ,3$ is executed. Let $\mathcal{D}$ denotes the available data, i.e. the answer provided by the warden. The three hypotheses have the same prior probabilities, $P(\mathcal{H}_i) = \frac{1}{3}, \ \ i = 1, \ldots, 3$. Since prisoner one asked the question, we are interested in $P(\mathcal{H}_1|\mathcal{D})$. Again using Bayes' theorem we find that

$$P(\mathcal{H}_1|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H}_1)P(\mathcal{H}_1)}{P(\mathcal{D})}.$$

The response $\mathcal{D}$ from the warden is either $\mathcal{D} = 2$ or $\mathcal{D} = 3$. For hypothesis $\mathcal{H}_1$ (both prisoners 2 and 3 are being released) the warden has a choice. Let us assume that he chooses randomly between them (no bias). This means that $P(\mathcal{D} = 2|\mathcal{H}_1) = \frac{1}{2} = P(\mathcal{D} = 3|\mathcal{H}_1)$. In the case of $\mathcal{H}_2$ or $\mathcal{H}_3$, the warden does not have a choice, and it follows that

$$\begin{aligned} P(\mathcal{D} = 2|\mathcal{H}_2) = 0 = P(\mathcal{D} = 3|\mathcal{H}_3), \\ P(\mathcal{D} = 3|\mathcal{H}_2) = 1 = P(\mathcal{D} = 2|\mathcal{H}_3). \end{aligned}$$

This allows us to compute

$$\begin{aligned} P(\mathcal{D}) &= \sum_{i=1}^{3} P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i) \\ &= \frac{1}{3}\sum_{i=1}^{3} P(\mathcal{D}|\mathcal{H}_i). \end{aligned}$$

Thus

$$\begin{aligned} P(\mathcal{D}=2) &= \frac{1}{3}\left[P(\mathcal{D}=2|\mathcal{H}_1)+P(\mathcal{D}=2|\mathcal{H}_2)+P(\mathcal{D}=2|\mathcal{H}_3)\right] \\ &= \frac{1}{3}\left[\frac{1}{2}+0+1\right] \\ &= \frac{1}{2}, \end{aligned}$$

similarly

$$P(\mathcal{D}=3)=\frac{1}{2}.$$

Thus we find that

$$\begin{aligned} P(\mathcal{H}_1|\mathcal{D}) &= \frac{\frac{1}{2}\times\frac{1}{3}}{\frac{1}{2}} \\ &= \frac{1}{3}, \end{aligned}$$

for either $\mathcal{D}=2$ or $\mathcal{D}=3$. Thus, perhaps surprisingly, the additional information does not affect the chances of prisoner 1 to be released at all. This does not mean that he slept any easier.

Exercise 22. Let us say the prisoners learn that prisoner 3 is being released. How does this affect the chances of prisoner 2?

Example 23. At a crime scene blood samples from two individuals are found. After testing, the samples turn out to be of type O, a relative common blood type (about 60% of the population), and type AB, a rather rare blood type (about 1% of the population). A suspect is tested and his blood type is found to be of type O. What is the probability, based on blood types, that he was at the crime scene?

We are dealing with two different hypotheses. Let us denote the hypothesis, 'the suspect and one other unknown person was at the crime scene', by S, and the alternative, 'two unknown people were present at the crime scene', by $\overline{S}$. In this case the priors are the prior probabilities of the two scenarios, of which we might not know much. Let us therefore concentrate on what can be learned from the data or, equivalently, assign the same prior probabilities to the two scenarios. Given S, i.e.

the suspect was present at the crime scene, the probability of the data is

$$P(\mathcal{D}|S) = p_{AB},$$

the probability of the blood type AB. (We are given that the suspect was present, thus the presence of blood type O is a given.) The probability of the data given $\overline{S}$, is

$$P(\mathcal{D}|\overline{S}) = 2p_O p_{AB}.$$

The presence of the factor 2 might cause some confusion. We need to take into account all possible scenarios that explain the two different blood types, subject to the hypothesis. Here the hypothesis is that two unknown individuals are present, one with blood type O and one with blood type AB. There are two ways in which that can happen, given two individuals.

The likelihood ration is therefore

$$\frac{P(\mathcal{D}|S)}{P(\mathcal{D}|\overline{S})} = \frac{1}{2p_O} = 0.83.$$

This analysis provides weak evidence, *against* the suspect being present at the crime scene.

This result looks reasonable. If, for instance, the analysis provided evidence *for* the suspect being present at the crime scene, the same analysis would then indicate that *all* persons with blood type are suspect, 60% of the population.

Exercise 24. Suppose the suspect has blood type AB, what is the probability that he was present at the crime scene.

Exercise 25. Consider the following (actual) statement:

> When spun on edge 250 times, a Belgian one-euro coin came up heads 140 times and tails 110. ‘It looks very suspicious to me’, said Barry Blight, a statistics lecturer at the London School of Economics. ‘If the coin were unbiased the chance of getting a result as extreme as that would be less than 7%’.

What do you think, does this data provide evidence that the coin is biased rather than fair?

Exercise 26. You visit a family with three children, but you don't know their sexes. Each child has his/her own bedroom, and you stumble by chance into one of the bedrooms. It is clear that it is the bedroom of a girl. Then you observe a letter from the school addressed to parents of the boys in the school.

Now you know that the family has at least one boy and one girl. What is more likely, that the family has one boy and two girls, or one girl and two boys? State your answer in terms of probabilities.

Exercise 27. The Monte Hall problem. (For this problem it is a good idea to study Example 21 first.)

The rules of a game show are as follows. The contestant is shown three door, behind one door is a prize, behind the other two doors, nothing. The contestant chooses a door and indicate it to the game show host. At lease one of the two remaining doors, is empty. The host, who knows where the prize is, chooses an empty door between the other two, and open it. The contestant is now faced with two closed doors, his original choice and one other, and one open, empty door. The contestant now gets the opportunity of changing his/her original choice.

Calculate the probabilities of winning the prize when (a) the contestant changes his/her original choice, (b) sticks with his/her original choice.

If you know this problem, chances are that you know an easy argument to find the probabilities. If you don't know the argument, try and find it. What I want you to do, is to argue systematically, based on conditional probabilities. In what way does the opening of an empty door, adds information to the system that you can exploit?

Exercise 28. This problem is exactly like the previous one. Except that after the contestant made his/her initial choice, there is a small earthquake, causing one of the remaining empty doors to fly open. Assuming that earthquakes know nothing about the game, how does this change the situation? What is the probabilities of winning the prize, using both strategies?

Exercise 29. You play a game with your friend consisting of three identical cards, except that both sides of one card are painted black, both sides of the second card are painted red, and the two sides of the third card are painted red and black respectively. Your friend picks a card at random, then show you, also randomly, one of the sides. You therefore see either a red or black side. The game consists of you guessing the color of the other, unseen, side of the card. What is your best strategy to guess right as often as possible? How often do you expect to guess right?

You can again approach the problem by listing all possibilities. Alternatively, you can use Bayes' theorem. Try comparing three hypotheses, you see the black-black card, you see the red-red card, you see the red-black card. If C is the random variable describing your observation, i.e. $C \in \{\text{red, black}\}$, you need to calculate the posterior probabilities, $P(\mathcal{H}|C)$. Answer: $P(\mathcal{H} = \text{red-red}|C = \text{red}) = \frac{2}{3}$, $P(\mathcal{H} = \text{red-black}|C = \text{red}) = \frac{1}{3}$.

Now that we have seen the examples, and you have worked through a few exercises, let us look at the general situation. Suppose have are observing a generative process providing us with observations $\mathcal{D}$, assumed to be statistically independent. We now *model* the process, resulting in a generative model. That is we hypothesize that the observed data is generated by a pdf $p(\mathbf{x}|\theta_a, \mathcal{H}_a)$, where we now explicitly indicate that the model is conditioned on our hypothesis $\mathcal{H}_a$, and the model depends on parameters θ_a that can be adjusted to the data. Given the observations, we form the likelihood $p(\mathcal{D}|\theta_a, \mathcal{H}_a)$. Because the observations are statistically independent, this joint distribution factorizes over the individual observations.

The posterior distribution over the parameters requires a prior $p(\theta_a|\mathcal{H}_a)$ over the parameters, and is given by

$$p(\theta_a|\mathcal{D}, \mathcal{H}_a) = \frac{p(\mathcal{D}|\theta_a, \mathcal{H}_a)p(\theta_a|\mathcal{H}_a)}{p(\mathcal{D}_a|\mathcal{H}_a)}, \tag{2.2}$$

where we recall that the denominator is also referred to as the *evidence.* Given the posterior of the parameters, we make predictions. For example, the probability of making a specific observation $\mathbf{x}$, is given by

$$p(\mathbf{x}|\mathcal{D}, \mathcal{H}_a) = \int p(\mathbf{x}|\theta, \mathcal{H}_a)p(\theta|\mathcal{D}, \mathcal{H}_a)d\theta,$$

where we have made use of the conditional independence of $\mathbf{x}$ on the data $\mathcal{D}$, given the parameters θ.

We can also ask how well the data supports the hypothesis. Introducing the prior $P(\mathcal{H}_a)$, it follows that

$$P(\mathcal{H}_a|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{H}_a)P(\mathcal{H}_a)}{p(\mathcal{D})},$$

where $p(\mathcal{D}|\mathcal{H}_a)$ is given by the *evidence* in (2.2). The normalization term $p(\mathcal{D})$ poses a problem. In order to do the normalization we need an exhaustive set of hypotheses, which may not be available. In practice however, a *comparison* of different hypotheses is often all that is required. Accordingly, suppose we have an alternative hypothesis $\mathcal{H}_b$ with model parameters θ_b. Following the same reasoning as above, we find that

$$\frac{P(\mathcal{H}_a|\mathcal{D})}{P(\mathcal{H}_b|\mathcal{D})} = \frac{p(\mathcal{D}|\mathcal{H}_a)P(\mathcal{H}_a)}{p(\mathcal{D}|\mathcal{H}_b)P(\mathcal{H}_b)}.$$

Taking the log, gives

$$\log\frac{P(\mathcal{H}_a|\mathcal{D})}{P(\mathcal{H}_b|\mathcal{D})} = \log\frac{p(\mathcal{D}|\mathcal{H}_a)}{p(\mathcal{D}|\mathcal{H}_b)} + \log\frac{P(\mathcal{H}_a)}{P(\mathcal{H}_b)}.$$

This nicely separates the contributions of two quantities. The *odds* in favor of hypothesis $\mathcal{H}_a$ against hypothesis $\mathcal{H}_b$ is just the ratio $\frac{P(\mathcal{H}_a)}{P(\mathcal{H}_b)}$. Thus, the posterior log-odds equals the prior log-odds plus the log-likelihood ratio.

If you are wondering about the meaning of the term 'odds', it derives from betting where the odds in favor of an event E is given by

$$\text{odds}(E) = \frac{P(E)}{1 - P(E)}.$$

For example, if $P(E) = 0.2$, then the odds in favor of event E, is given by $\text{odds}(E) = 0.25$. In betting the bookmakers give their odds *against* the event taking place. Thus, if $P(E) = 0.2$, the odds against it taking place is $1/\text{odds}(E) = 4$. This means that betting R1, you win R4 if the event realizes, against the bookie who wins your R1 if the event does not take place.

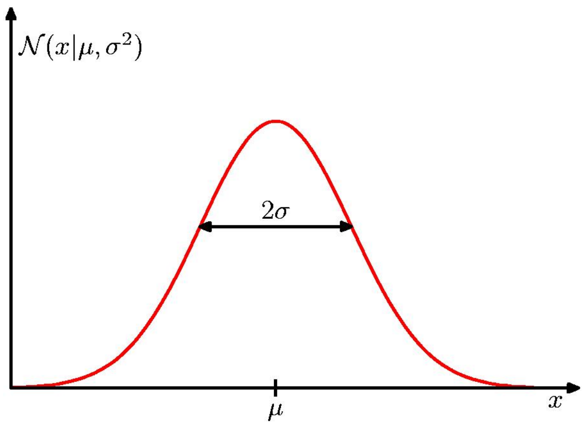


FIGURE 2.5.1. Gaussian density function.

2.5. Gaussian Distribution.

2.5.1. Univariate Gaussian Distributions. One of the most important distributions is the Gaussian– or normal distribution

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}. \tag{2.1}$$

The Gaussian distribution is governed by exactly two parameters, the mean μ and the variance σ^2. The square root of the variance σ is called the standard deviation and the reciprocal of the variance $1/\sigma^2$ is called the precision—the smaller the variance, the greater the precision, see Figure 2.5.1.

Note that

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} N(x|\mu,\sigma^2)x dx = \mu$$

and

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} N(x|\mu,\sigma^2)x^2 dx = \mu^2 + \sigma^2.$$

Thus the variance is given by

$$\operatorname{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2.$$

Suppose we have N independent observations $\mathbf{x} = [x_1 \; \cdots \; x_N]^T$ all drawn from the same Gaussian distribution $N(x|\mu,\sigma^2)$. Such data points are said to be *independent* and i*dentically* distributed, or i.i.d. Furthermore, suppose that we do not know

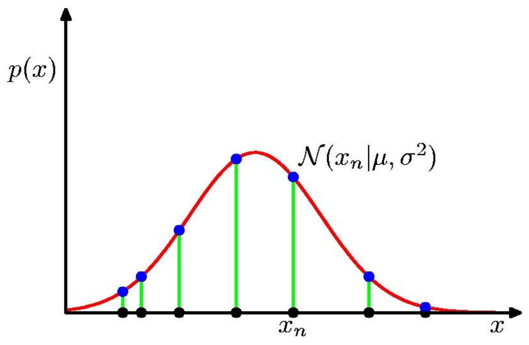


FIGURE 2.5.2. Likelihood function.

the values of μ and σ^2 and want to infer them from the observations. Because the data set $\mathbf{x}$ is i.i.d. we can factorize the joint distribution as

$$p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} N(x_n|\mu,\sigma^2).$$

Viewed as a function of μ and σ^2 this is the likelihood function for the Gaussian parameters, see Figure 2.5.2.

In order to infer the values of μ and σ^2 from the data, one might maximize the likelihood function, or rather, the log likelihood,

$$\ln p(x|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi).$$

Maximizing with respect to μ gives the *maximum likelihood* solution,

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

which is the *sample* mean. The maximum likelihood solution of the variance is similarly obtained as

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{ML})^2.$$

EXERCISE 30. The sample mean and sample variance are also random variables and one can calculate their mean. Show that

- $\mathbb{E}[\mu_{ML}] = \mu$

- $\mathbb{E}[(\mu_{ML}-\mu)^2]=\frac{\sigma^2}{N}$. Hint: It is the simplest to write $\delta_n=\mu-x_n$, so that $\mathbb{E}[\delta]=0$ and $\mathbb{E}[\delta^2]=\sigma^2$.
- $\mathbb{E}[\sigma^2_{ML}]=\frac{N-1}{N}\sigma^2$. Hint: Write

$$\begin{aligned}\sum_{n=1}^{N}(x_n-\mu_{ML})^2 &= \sum_{n=1}^{N}\left[(x_n-\mu)-(\mu_{ML}-\mu)\right]^2\\ &= \sum_{n=1}^{N}(x_n-\mu)^2-2(\mu_{ML}-\mu)\sum_{n=1}^{N}(x_n-\mu)+N(\mu_{ML}-\mu)^2\\ &= \sum_{n=1}^{N}(x_n-\mu)^2-N(\mu_{ML}-\mu)^2.\end{aligned}$$

From Exercise 30 it is clear that a bias is introduced into the sample variance. This is due to the fact that the variance is calculated with respect to the sample mean and not the real mean. An unbiased estimate is easily obtained (show it yourself!) by

$$\sigma^2_{UB}=\frac{1}{N-1}\sum_{n=1}^{N}(x_n-\mu_{ML})^2.$$

The bias can also be explained in the following way. First note that the data points are distributed around the true mean μ with mean squared error σ^2. The sample mean μ_{ML} minimizes the mean squared error of the data. Thus, unless μ and μ_{ML} coincide, the data has a larger sum-squared deviation from the the true mean μ than from the sample mean μ_{ML}. The expected mean squared deviation from the sample mean is therefore smaller than the mean squared deviation from the true mean. This is illustrated in Figure 2.5.3.

Exercise 31. Assume that you are observing a generative process that generates observations from a univariate Gaussian distribution, $\mathcal{N}(x|0,\sigma^2=1)$. You are asked to write a computer program to generate samples from this distribution. You may assume that you have access to an algorithm that draws samples from a uniform distribution. These are readily available and you may use whatever is available in your favorite software library. Explain, based on the discussion of Section~1.3 how you can use samples drawn from a uniform distribution to generate samples from a Gaussian distribution. Use this to write a program that will generate an arbitrary

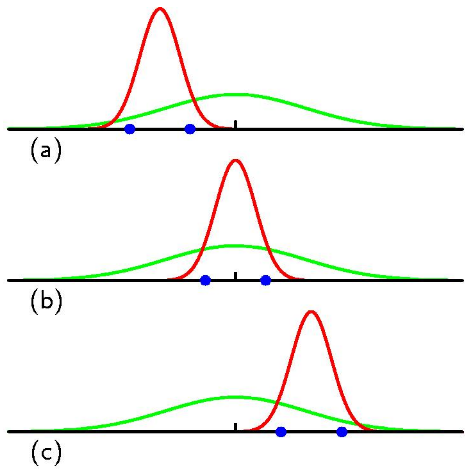


FIGURE 2.5.3. Illustration of how a bias arises. Averaged over three data sets the mean is correct but the variance is under-estimated.

number of data points from the Gaussian distribution. Plot the data together with the Gaussian distribution. How can you easily adapt your code in order to generate data from a Gaussian distribution with arbitrary mean and covariance?

2.5.2. Multivariate Gaussian Distributions. In d dimensions the Gaussian distribution is given by,

$$N(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{|2\pi\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{2.2}$$

where $|\cdot|$ denotes the determinant, and the covariance matrix Σ is an $d \times d$ symmetric matrix.

In order to qualify as a covariance matrix, a matrix has to satisfy a few properties. We have already mentioned that it needs to be symmetric. This implies that all its eigenvalues are real, and the eigenvectors orthogonal. We'll alway ensure that the eigenvectors are also normalized, i.e. they all have unit length. Real eigenvalues are not sufficient, we also require that they are *non-negative.*

In practice, just as in the one dimensional case, the mean and covariance are estimated from data. Using maximum likelihood, as in the one dimensional case, the mean and (biased) covariance are estimated as,

$$\mu = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n,$$

and

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^T.$$

EXERCISE 32. Show that there is no loss in generality to assume that the covariance matrix is symmetric. Hint: Assume that Σ^{-1} is not symmetric and write it as $\Sigma^{-1} = \Sigma_s^{-1} + \Sigma_{as}^{-1}$, i.e. write it as the sum of a symmetric and an anti-symmetric matrix. Now show that the anti-symmetric part cancels and draw the appropriate conclusion.

The Mahalanobis distance is defined as,

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}). \tag{2.3}$$

Note:

- $\Delta =$ const defines the curves of constant probability.

There is a simple but very useful transformation that allows one (i) to visualize the curves of constant probability, and (ii) to better understand what the covariance matrix, and the Gaussian distribution, is about. The basic idea is to make a transformation that diagonalizes the covariance matrix.

If we write $U = [\mathbf{u}_1 \cdots \mathbf{u}_d]^T$ and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$ where the $\mathbf{u}_j$ are the eigenvectors, and the λ_j the corresponding eigenvalues of the covariance matrix, i.e.,

$$\Sigma\mathbf{u}_j = \lambda_j\mathbf{u}_j,\ j = 1, \ldots, d,$$

then we can write

$$\begin{bmatrix} \Sigma\mathbf{u}_1 & \ldots & \Sigma\mathbf{u}_d \end{bmatrix} = \begin{bmatrix} \lambda_1\mathbf{u}_1 & \ldots & \lambda_d\mathbf{u}_d \end{bmatrix}$$

or

$$\Sigma U = U\Lambda.$$

This means that

$$\Sigma = U\Lambda U^T$$

or

$$\Sigma^{-1} = U\Lambda^{-1}U^T$$

assuming, of course, that the eigenvectors are normalized. Multiplying out this gives,

$$\Sigma = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{u}_i^T,$$

and

$$\Sigma^{-1} = \sum_{i=1}^{d} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T.$$

If we now change variables

$$\mathbf{y} = U^T(\mathbf{x} - \boldsymbol{\mu}) \tag{2.4}$$

in (2.3) we get

$$\begin{aligned} \Delta^2 &= \mathbf{y}^T U^T \Sigma^{-1} U \mathbf{y} \\ &= \mathbf{y}^T \Lambda^{-1} \mathbf{y} \\ &= \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i} \end{aligned}$$

Let us again look at the curves of constant probability, $\Delta^2 = \text{const}$. In the y-coordinates, this is given by,

$$\sum_{i=1}^{d} \frac{y_i^2}{\lambda_i} = \text{const}.$$

This should be familiar, in the y-coordinates the curves are ellipses, with the major axes aligned along the coordinate axes. If this is not clear, choose $d = 2$ and draw the curves. Moreover, the ellipse $\sum_{i=1}^{d} \frac{y_i^2}{\lambda_i} = 1$, intersects the coordinate axes at $\pm\sqrt{\lambda_i}, \quad i = 1, \ldots, d$. This means that the eigenvalues of the covariance matrix measure the 'spread' of the data (transformed to y-variables) along the coordinate axes. Thus the λ_i are the variances in the coordinates directions.

How about the original coordinates? The transformation (2.4) is simply a rotation around the mean of the data so that the principle axes (the eigenvectors of the

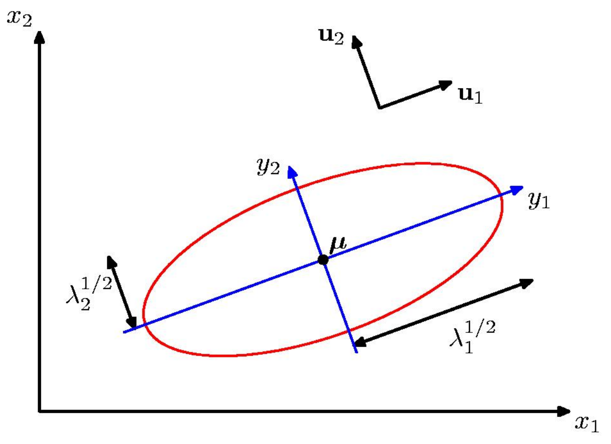


FIGURE 2.5.4. Rotating the principle axes.

covariance matrix) coincide with the coordinate axes, see Figure 2.5.4. Thus in the original coordinates, the eigenvalues λ_i are the variances along the principal directions.

There is one more important conclusion we can draw from this transformation. In the y-coordinates the Gaussian distribution becomes

$$\begin{aligned} p(\mathbf{y}) &= N(\mathbf{y}|\mathbf{0}, \Lambda). \\ &= \prod_{i=1}^{d} N(y_i|0, \lambda_i). \end{aligned}$$

This is magical: The $\mathbf{y}$ coordinates, obtained by a rotation of the $\mathbf{x}$ coordinates, are statistically independent since the covariance matrix is diagonal. Thus for normally distributed variables one easily removes any statistical dependence through a simple rotation around the mean. Moreover, it should now be clear that the variances along the different directions are given by the eigenvalues of the covariance matrix.

Note: For the normal distribution (2.2) the mean and covariance are given by

- $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$.
- $\mathrm{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T] = \Sigma$.

The following exercise prepares the way for what is to follow.

EXERCISE 33. Jointly Gaussian. Assume that two Gaussian variables x and y are jointly Gaussian in the sense that

$$p(x, y) = N(\mathbf{x}|\boldsymbol{\mu}, \Sigma),$$

where $\mathbf{x} = [x\ y]^T$ and $\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

- Show that the marginal distributions $p(x)$ and $p(y)$ are Gaussian and calculate their mean and covariance.
- Show that $p(x|y)$ is Gaussian and calculate its mean and covariance.
- Now show that x and y are independent if and only if $\sigma_{xy} = 0$.

Although the next section might seem a little technical, the results discussed below is one of the reasons why we use Gaussian distributions extensively.

Conditional Gaussian distribution. The following identity will be used extensively,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} \tag{2.5}$$

where

$$M = \left(A - BD^{-1}C\right)^{-1}.$$

EXERCISE 34. Verify the identity by showing that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} = I.$$

Suppose that $\mathbf{x} \in \mathbb{R}^d$ is a Gaussian random variable $\mathbf{x} \sim N(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$. Partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}$$

where $\mathbf{x}_a$ is the first M components of $\mathbf{x}$. We also partition the mean and covariance matrix,

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix},$$

and

$$\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

where $\Sigma_{aa} = \Sigma_{aa}^T$, $\Sigma_{bb} = \Sigma_{bb}^T$, $\Sigma_{ab} = \Sigma_{ba}^T$. Also partition the precision matrix,

$$\Lambda = \Sigma^{-1},$$

as

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}.$$

We are looking for $p(\mathbf{x}_a|\mathbf{x}_b)$. It turns out that it is again a Gaussian distribution in $\mathbf{x}_a$. Therefore we only need to find the mean and covariance of $p(\mathbf{x}_a|\mathbf{x}_b)$. In principle this is easy to calculate. All we need to do is to fix $\mathbf{x}_b$ in the joint distribution, and to extract the quadratic form of the remaining $\mathbf{x}_a$. A tedious but essentially straightforward calculation gives,

$$\Sigma_{a|b} = \Lambda_{aa}^{-1},$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \Lambda_{aa}^{-1}\Lambda_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b).$$

Making use of the matrix identity this can also be expressed in terms of the partitioned covariance matrix

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \tag{2.6}$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b). \tag{2.7}$$

Note that the conditional mean takes the form

$$\boldsymbol{\mu}_{a|b} = A\mathbf{x}_b + \mathbf{b}$$

and that the conditional covariance is independent of $\mathbf{x}_a$. This is an example of a linear Gaussian model.

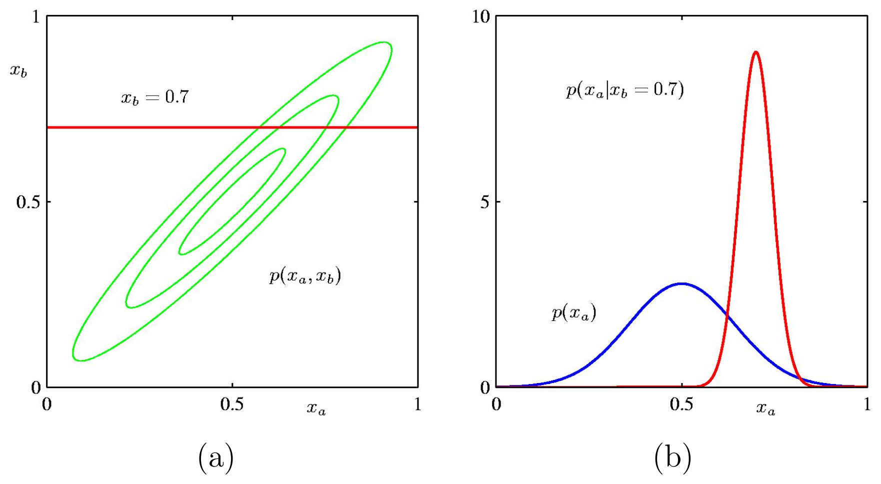


FIGURE 2.5.5. (a) $2D$ Gaussian distribution. (b) Marginal and conditional distributions.

2.5.3. Marginal Gaussian distribution. We want to compute the marginal distribution

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b.$$

Again it turns out to be a Gaussian distribution,

$$p(\mathbf{x}_a) = N(\mathbf{x}_a | \boldsymbol{\mu}_a, \Sigma_{aa}).$$

This means that the marginal distribution can be simply read off the partitioned mean and covariance. This illustrated in Figure 2.5.5

2.5.4. Bayes' theorem for Gaussian variables. Assume we are given a Gaussian marginal distribution $p(\mathbf{x})$ and a Gaussian conditional distribution $p(\mathbf{y}|\mathbf{x})$ that has a mean that is linear in $\mathbf{x}$ and a covariance independent of $\mathbf{x}$. We wish to find the marginal distribution $p(\mathbf{y})$ and the conditional distribution $p(\mathbf{x}|\mathbf{y})$. Let

$$\begin{aligned} p(\mathbf{x}) &= N(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}) \\ p(\mathbf{y}|\mathbf{x}) &= N(\mathbf{y}|A\mathbf{x} + \mathbf{b}, L^{-1}). \end{aligned}$$

If $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^d$ then A is an $d \times m$ matrix. Note that we are essentially given the joint distribution

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}).$$

Since it is the product of Gaussian distributions, it is also a Gaussian distribution. Therefore the marginal $p(\mathbf{y})$ can be obtained through marginalization. Again an essentially straightforward but tedious calculation gives the mean and covariance of the marginal,

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= A\boldsymbol{\mu} + \mathbf{b} \\ \operatorname{cov}[\mathbf{y}] &= L^{-1} + A\Lambda^{-1}A^T. \end{aligned}$$

The posterior distribution is now obtained using Bayes' theorem,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}.$$

Not surprisingly this is a Gaussian distribution with mean and covariance given by,

$$\begin{aligned} \mathbb{E}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1}\left(A^T L(\mathbf{y} - \mathbf{b}) + \Lambda\boldsymbol{\mu}\right) \\ \operatorname{cov}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1}. \end{aligned}$$

Summary:

Given a marginal distribution $p(\mathbf{x})$ and a conditional distribution $p(\mathbf{y}|\mathbf{x})$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}) \tag{2.8}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|A\mathbf{x} + \mathbf{b}, L^{-1}) \tag{2.9}$$

the marginal $p(\mathbf{y})$ and conditional $p(\mathbf{x}|\mathbf{y})$ distributions are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|A\boldsymbol{\mu} + \mathbf{b}, L^{-1} + A\Lambda^{-1}A^T) \tag{2.10}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\Sigma\left\{A^T L(\mathbf{y} - \mathbf{b}) + \Lambda\boldsymbol{\mu}\right\}, \Sigma) \tag{2.11}$$

where

$$\Sigma = (\Lambda + A^T L A)^{-1}.$$

2.5.5. Maximum likelihood. Given an i.i.d. data set, $X = [\mathbf{x}_1 \dots \mathbf{x}_N]$ drawn from a multivariate Gaussian distribution, the log likelihood is given by

$$\ln p(X|\boldsymbol{\mu}, \Sigma) = -\frac{Nd}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}).$$

The sufficient statistics are given by $\sum_{n=1}^N \mathbf{x}_n$ and $\sum_{n=1}^N \mathbf{x}_n\mathbf{x}_n^T$. Maximizing gives,

$$\boldsymbol{\mu}_{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

and

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

The unbiased estimate is given by

$$\Sigma_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

2.5.6. Bayesian inference for the univariate Gaussian. Suppose that the variance σ^2 is known. We need to infer the mean μ given data $\mathcal{D} = \{x_1, \dots, x_N\}$. The likelihood is given by

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}.$$

Maximizing the likelihood gives a point estimate of μ. A systematic Bayesian approach treats μ like a random variable and looks for a pdf over μ, given the data. This means that we we are after the posterior

$$p(\mu|\mathcal{D}) = \frac{p(\mathcal{D}|\mu)p(\mu)}{p(\mathcal{D})}.$$

Here one has to provide a suitable prior $p(\mu)$. Note that the likelihood as a function of μ, assumes the form of a Gaussian distribution albeit not properly normalized. This however, allows us to choose a conjugate prior $p(\mu)$ in the form of a Gaussian,

$$p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$$

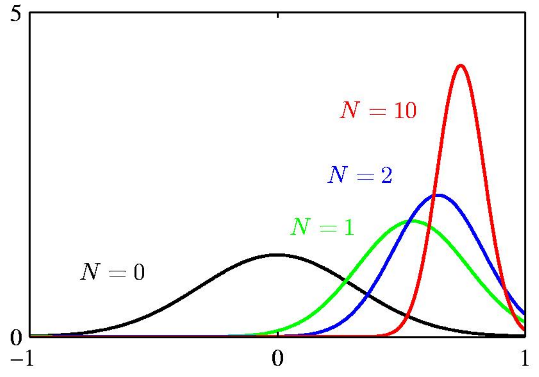


FIGURE 2.5.6. Bayesian inference for the mean μ.

so that the posterior distribution becomes

$$p(\mu|\mathcal{D}) \propto p(\mathcal{D}|\mu)p(\mu).$$

Completing the square it follows that

$$p(\mu|\mathcal{D}) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$$

where

$$\begin{aligned} \mu_N &= \frac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\mu_{ML} \\ \frac{1}{\sigma_N^2} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \end{aligned}$$

where μ_{ML} is the sample mean.

Note that the prior plays the role of a fictitious observation followed by N 'real' observations. The observations serve to 'correct' or modify the prior assumption, see Figure 2.5.6. Also note that one can manipulate the relative importance of the prior by adjusting the prior variance σ_0^2. A small value of σ_0^2 indicates high confidence in the prior. In fact $\sigma_0^2 = 0$ shows absolute confidence with the result that the measurements have no effect. Small values means that its effect is persistent and only disappears as the number of observations $N \longrightarrow \infty$.

Note that this leads naturally to a sequential view where is observation is used to correct our current best estimate.

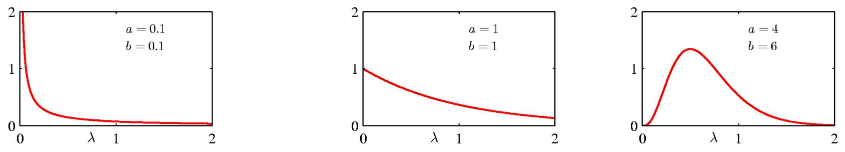


FIGURE 2.5.7. The gamma distribution.

Now suppose that the mean is known and we want to infer the variance, or more conveniently the precision $\lambda = \frac{1}{\sigma^2}$. The likelihood is given by

$$p(\mathbf{x}|\lambda) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \lambda^{-1}) \propto \lambda^{N/2} \exp\left\{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}.$$

Following the same reasoning as above, the conjugate prior is a gamma distribution

$$\text{Gam}(\lambda|a,b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda),$$

with mean and variance given by,

$$\begin{aligned}\mathbb{E}[\lambda] &= \frac{a}{b}\\ \text{var}[\lambda] &= \frac{a}{b^2}.\end{aligned}$$

The gamma distribution is illustrated in Figure 2.5.7

Using the prior $p(\lambda) = \text{Gam}(\lambda|a_0, b_0)$ the posterior becomes

$$p(\lambda|\mathcal{D}) \propto \lambda^{a_0-1}\lambda^{N/2} \exp\left\{-b_0\lambda - \frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$

which is a Gamma distribution of the form $\text{Gam}(\lambda|a_N, b_N)$ where

$$\begin{aligned}a_N &= a_0 + \frac{N}{2}\\ b_N &= b_0 + \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2 = b_0 + \frac{N}{2}\sigma_{ML}^2.\end{aligned}$$

Note that N observations increase a_0 by $\frac{N}{2}$. One therefore interprets a_0 as being the equivalent of $2a_0$ prior observations. The N observations also increase the value of b_0 by $N\sigma_{ML}^2/2$. If we want to interpret b_0 as being the result of the $2a_0$ prior observations, we have to write $b_0 = a_0 \frac{b_0}{a_0}$ so that we conclude that the prior is the equivalent of $2a_0$observations with variance $\frac{b_0}{a_0}$.

If both the mean and precision are unknown, we are looking for an expression for the posterior $p(\mu, \lambda|\mathcal{D}) \propto p(\mathcal{D}|\mu, \lambda)p(\mu, \lambda)$. It is worth writing the likelihood again,

$$\begin{aligned} p(\mathcal{D}|\mu, \lambda) &= \prod_{n=1}^{N} \left(\frac{\lambda}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{\lambda}{2}(x_n - \mu)^2\right\} \\ &\propto \left[\lambda^{\frac{1}{2}} \exp\left(-\frac{\lambda\mu^2}{2}\right)\right]^N \exp\left\{\lambda\mu \sum_{n=1}^{N} x_n - \frac{\lambda}{2}\sum_{n=1}^{N} x_n^2\right\}. \end{aligned}$$

A conjugate prior takes the same functional form as the likelihood and should therefore be of the form

$$\begin{aligned} p(\mu, \lambda) &\propto \left[\lambda^{\frac{1}{2}} \exp\left(-\frac{\lambda\mu^2}{2}\right)\right]^{\beta} \exp\{c\lambda\mu - d\lambda\} \\ &= \exp\left(-\frac{\beta\lambda\mu^2}{2} + c\lambda\mu\right)\left[\lambda^{\frac{\beta}{2}} \exp(-d\lambda)\right] \\ &= \exp\left\{-\frac{\beta\lambda}{2}\left(\mu - \frac{c}{\beta}\right)^2\right\}\left[\lambda^{\frac{\beta}{2}} \exp\left\{-\left(d - \frac{c^2}{2\beta}\right)\lambda\right\}\right]. \end{aligned}$$

Factorized the prior as $p(\mu, \lambda) = p(\mu|\lambda)p(\lambda)$, and compare with the expression above to find that

$$\begin{aligned} p(\mu, \lambda) &= p(\mu|\lambda)p(\lambda) \\ &= \mathcal{N}\left(\mu|\mu_0, (\beta\lambda)^{-1}\right) \mathrm{Gam}(\lambda|a, b) \end{aligned}$$

with

$$\mu_0 = \frac{c}{\beta}, \quad a = 1 + \frac{\beta}{2}, \quad b = d - \frac{c^2}{2\beta}.$$

This means that the posterior has a normal-gamma distribution.

Exercise 35. In Exercise 31 you developed a program that samples from a univariate Gaussian pdf. Generalize your program to generate samples from a multivariate Gaussian distribution. Plot your results for a 2D Gaussian distribution. Plot the data points together with the contours of the pdf (this might require some thought). Choose covariance matrices so that the two components are (i) independent, (ii) dependent. Also use a covariance matrix that is a multiple of the identity matrix. Note how the shape of the contours change. And of course your data should be consistent with these curves.

2.6. Linear Transformations of Gaussians and the central limit theorem.

There is another important and useful property of Gaussian pdf's namely that a linear transformation results in another Gaussian. Let

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

where A is an invertible matrix. Substituting into (2.2) gives,

$$\begin{aligned}\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_y, \Sigma_y) &\propto \exp(-\frac{1}{2}(A\mathbf{y}-\boldsymbol{\mu})^T\Sigma^{-1}(A\mathbf{y}-\boldsymbol{\mu})) \\ &= \exp(-\frac{1}{2}(\mathbf{y}-A^{-1}\boldsymbol{\mu})^T A^T\Sigma^{-1}A(\mathbf{y}-A^{-1}\boldsymbol{\mu})).\end{aligned}$$

It now easily follows that $\boldsymbol{\mu}_y = A^{-1}\boldsymbol{\mu}$ and $\Sigma_y = A^{-1}\Sigma A^{-T}$. Although it is a linear transformation, it is a very special kind of linear transformation. Consider for instance the following transformation

$$z = x_1 + x_2$$

where both x_1 and x_2 are Gaussian random variables, let us say with means μ_1 and μ_2 and variances σ_1^2 and σ_2^2. This is also a linear transformation but in this case the transformation matrix is not invertible. The previous derivation does not cover this result, and the question remains, is a linear transformation of a Gaussian again a Gaussian? The next exercise guides you through a simple construction.

Exercise 36. Let $z_1 = x_1 + x_2$ and define $z_2 = x_2$. Write down the linear transformation $\mathbf{z} = A\mathbf{x}$ where $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Show that A is invertible. Assume that x_1 and x_2 are jointly Gaussian and write down the joint distribution for $\mathbf{x}$. From this you should be able to calculate the Gaussian for $\mathbf{z}$ from which you can obtain the distribution of z_1 by marginalization. Consider the separate cases when x_1 and x_2 are statistically independent, and when they are dependent.

What about random variables that are not Gaussian? The next exercise guides you through an example.

EXERCISE 37. Let $z = x_1 + x_2$ and assume that x_1 and x_2 are statistically independent random variables with, in this case, the pdf's given by the general functions $p_1(x_1)$ and $p_2(x_2)$ respectively. Try the same trick as in the previous exercise, i.e. write $z_1 = x_1 + x_2$ and $z_2 = x_2$. If $p(x_1, x_2)$ is the joint pdf we can now transform to the z-coordinates, $p(z_1, z_2) = p(x_1(z_1, z_2), x_2(z_1, z_2)) = p_1(x_1(z_1, z_2))p_2(x_2(z_1, z_2)) = p_1(z_1 - z_2)p_2(z_2)$ (noting that the determinant of the transformation matrix equals 1). If we do the transformation, and marginalize, $p(z_1) = \int p(z_1, z_2)dz_2$, we can write the pdf of z in the form of a convolution.

EXERCISE 38. Given two statistically independent random variables x and y with means $\overline{x}$ and $\overline{y}$ respectively, calculate the mean of $z = x + y$. Show that the variance of z is the sum of the variances of x and y.

EXERCISE 39. This exercise, together with the next one should help you understand the idea behind the central limit theorem.

Assume that you throw two fair six-sided dice, i.e. assume that the six sides are drawn from a uniform distribution every time you throw. Write down the probability distribution of the sum of the two dice. How does this relate to Exercise 37? What is the probability distribution of the sum of the numbers if you throw three dice?

Suppose that you throw 100 dice, what is the probability distribution of the sum of the numbers? Write a computer program that simulates this. Also describe how one can calculate the distribution theoretically, at least in principle (you may find it tedious to do it in practice).

EXERCISE 40. The central limit theorem states that if independent random variables $x_1, x_2, \ldots, x_N$ have means μ_n and variances σ_n^2, then, in the limit of large N, the sum $\sum_n x_n$ has a distribution that tends to a Gaussian distribution with mean $\sum_n \mu_n$ and variance $\sum_n \sigma_n^2$.

Calculate the mean and variance of one die (assume a uniform distribution). Use the results of the previous problems and calculate the mean and variance of the sum one hundred dice. In the previous question you simulated the probability distribution of the sum of 100 dice. Compare your simulation with a Gaussian distribution with mean and variance you have just calculated. Do you find confirmation for the central limit theorem in the result?

Since you have shown that the probability distribution of the sum of two independent variables is given by a convolution, another way of approaching this problem is to calculate the convolution of two uniform distributions (giving the distribution of two dice). Show that by taking convolutions of the result with a uniform distribution recursively, generates the probability distributions of more and more dice (you need to ensure that the probability distributions are properly normalized of course).

CHAPTER 3

DIMENSIONALITY REDUCTION

3.1. Introduction.

In this chapter we study two basic dimensionality reduction techniques, namely Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA). In each case the goal is to project the data onto a lower dimensional space while retaining some of the essential characteristics of the data. Each of the two methods focus on very different but specific characteristics of the data. Let us consider the two approaches separately.

In very general terms, PCA can be characterized as identifying a lower dimensional subspace of a given linear vector space. In order to explain the general idea, we consider a practical application.

In facial recognition we are interested in facial images, and only facial images. Any image is represented as an $m \times n$ matrix where every entry in the matrix (every pixel) represents a shade of gray (the dimension increases three-fold for color images). It should be clear that an $m \times n$ gray-scale images is a specific point in an $m \times n$-dimensional vector space. This can be very high even for low resolution images. Since we are interested in only facial images it might just be possible that the facial images occupy lower dimensional subspace of the full $m \times n$-dimensional images space.

In summary, PCA finds lower dimensional subspaces that describe the essential properties of our data. This allows one to project the data onto these lower dimensional subspaces, sometimes leading to significant dimensionality reductions.

Since principle components are closely related to the SVD the reader may find it useful to read this chapter together with Section **??** on the SVD.

LDA is particularly useful in classification problems. We'll discuss different techniques for classification later, here we are interested in preparing the data in order to build more efficient classifiers. In particular, we want to reduce the dimensionality of the data in such a way that we have maximum class separation in the lower dimensional space.

3.2. Principal Components .

Given N observations $\mathbf{x}_n \in \mathbb{R}^d,\ n = 1, \ldots, N$ of dimension d, we want to find the directions of maximum variation in the data. First we project the data onto a one dimensional subspace defined by $\mathbf{u}_1$, where $\mathbf{u}_1^T\mathbf{u}_1 = 1$. Thus $\mathbf{x}_n$ is projected onto the vector $\left(\mathbf{u}_1^T\mathbf{x}_n\right)\mathbf{u}_1$, and in this coordinate system, the projected value is just given by $\mathbf{u}_1^T\mathbf{x}_n$. The idea is to choose $\mathbf{u}_1$ in such a way that the variance of the projected values is a large as possible. If the sample mean is given by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n,$$

then the mean of the projected data is $\mathbf{u}_1^T\overline{\mathbf{x}}$ and the variance of the projected data is

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}}\right)^2 &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_1^T(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T\mathbf{u}_1 \\ &= \mathbf{u}_1^T S\mathbf{u}_1\end{aligned}$$

where S is the sample covariance,

$$S = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T.$$

Since we are only interested in the direction of $\mathbf{u}_1$, we minimize $\mathbf{u}_1^T S\mathbf{u}_1$ subject to the constraint $\mathbf{u}_1^T\mathbf{u}_1 = 1$. Introducing a Lagrange multiplier λ, we therefore maximize

$$\mathbf{u}_1^T S\mathbf{u}_1 + \lambda(1 - \mathbf{u}_1^T\mathbf{u}_1),$$

subject to $\mathbf{u}_1^T\mathbf{u}_1 = 1$. Differentiation with respect to $\mathbf{u}_1$ leads to the eigenvalue problem

$$S\mathbf{u}_1 = \lambda\mathbf{u}_1,$$

and using the constraint gives

$$\lambda = \mathbf{u}_1^T S\mathbf{u}_1.$$

The projected variance therefore equals the eigenvalue λ of the sample covariance S, and attains its maximum value if we choose the largest eigenvalue λ_1, with $\mathbf{u}_1$ its corresponding eigenvector. This is our first principal vector. Also note that the standard deviation—the mean deviation—is given by $\sqrt{\lambda}$.

The rest of the principal vectors are obtained by each time projecting onto a vector orthogonal all the previous principal directions. This means that the principal directions are the eigenvectors of the sample covariance matrix, and the eigenvalues are the variances in those directions.

EXERCISE 41. Show that the eigenvalues of the sample covariance matrix are all non-negative; a fact that is important for our discussion above.

3.2.1. Dimensionality reduction. In the previous section we identified the principal directions—the eigenvectors of the sample covariance matrix—as well as the variance in these directions—the corresponding eigenvalues, written as

$$SQ = Q\Lambda, \tag{3.1}$$

where Q is the eigenvector matrix of the covariance matrix S, and Λ a diagonal matrix with the eigenvalues along its diagonal. For convenience we arrange that the eigenvalues are ordered in non-increasing order of magnitude, i.e. $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_d$. Also recall that the eigenvector matrix is *orthogonal*, i.e. $Q^T Q = I = QQ^T$.

With these formalities taken care of, consider the transformation,

$$\mathbf{y}_n = Q^T \left(\mathbf{x}_n - \overline{\mathbf{x}}\right), \quad n = 1, \ldots, N. \tag{3.2}$$

Since Q is an orthogonal matrix this means that we have shifted the origin of the coordinate axes to the mean of the data, and we have rotated the coordinate axes around the mean so that the coordinate axes coincide with the principal directions.

Thus the sample covariance of the transformed data is given by

$$\begin{aligned} S_y &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^T \\ &= \frac{1}{N}\sum_{n=1}^{N} Q^T\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)\left(\mathbf{x}_n - \overline{\mathbf{x}}\right)^T Q \\ &= Q^T S Q \\ &= \Lambda. \end{aligned}$$

Since the transformed covariance matrix S_y is *diagonal*—with the eigenvalues of S along its diagonal—it also means that the different components of the transformed data vectors $\mathbf{y}$ are uncorrelated.

At this point $\mathbf{y} \in \mathbb{R}^d$ and we have not yet achieved any dimensionality reduction. In fact we have not changed the data at all—we have simply shifted and rotated the coordinate axes. Let us now form a new matrix Q_r consisting of the first r columns of the eigenvector matrix Q, i.e. the columns of Q_r consist of the r eigenvectors of S belonging to the r largest eigenvalues. Note that,

$$Q_r^T S Q_r = \Lambda_r, \tag{3.3}$$

where Λ_r is an $r \times r$ diagonal matrix consisting of the r largest eigenvalues of S. It is now straightforward to transform our original data as

$$\mathbf{y}_n = Q_r^T(\mathbf{x}_n - \overline{\mathbf{x}}), \quad n = 1, \ldots, N, \tag{3.4}$$

where $\mathbf{y} \in \mathbb{R}^r$, i.e. the data dimensionality is reduced from d to r.

How much do we loose by reducing the dimensionality in this way? It turns out that the magnitude of the first neglected eigenvalue, i.e. λ_{r+1} is a natural measure of the amount of information we loose. One can therefore define the relative amount of information that we loose as $\frac{\lambda_{r+1}}{\lambda_1}$. Of course if $\lambda_{r+1} = 0$ we don't loose anything at all.

3.2.2. The whitening transformation. We need not stop with the dimensionality reduction given by (3.3), indeed it is often convenient to transform the data so that it is spherically distributed. Since this amounts to the identity covariance matrix, the corresponding transformation follows immediately from (3.3),

$$\Lambda_r^{-1/2} Q_r^T S Q_r \Lambda_r^{-1/2} = I_r, \tag{3.5}$$

where I_r is the $r \times r$ identity matrix. Thus the whitening transformation of the data is given by

$$\mathbf{y} = \left(Q_r \Lambda_r^{-1/2}\right)^T (\mathbf{x} - \overline{\mathbf{x}}) . \tag{3.6}$$

Geometrically, what we do is to rotate the data around the mean so that the coordinate axes are aligned with the principal directions. Then we scale each axis so that the variances along all the coordinate axes are unity.

It might be useful to point out exactly what we have done. First we use the covariance matrix S to find the transformation (3.5) that transforms it to the identity matrix. Having identified this transformation, it can then be used to transform the data (3.6) so that the covariance matrix of the transformed data is the identity.

3.2.3. Using the Singular Value Decomposition. Although the procedure described above is acceptable, a more elegant alternative is available. Let us therefore describe the use of the Singular Value Decomposition (SVD). Again we assume that we have N, d-dimensional observations, $\mathbf{x}_j, \;\; j = 1, \ldots, N$. We also assume that we have already removed the mean from the data so that $\sum_{j=1}^N \mathbf{x}_j = \mathbf{0}$. We now collect this data into a single $d \times N$ matrix (array),

$$D = [\mathbf{x}_1 \cdots \mathbf{x}_N].$$

It is important that $N > d$. Can you think why? Let us now write down the SVD of D,

$$D = U \Sigma V^T.$$

Since D is an $d \times N$ matrix, U and V are $d \times d$ and $N \times N$ *orthogonal* matrices, respectively. Σ is an $d \times N$ *diagonal* matrix with the nonzero *singular values* arranged from large to small, on its diagonal,

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots \cdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & \sigma_d & 0 & \cdots & 0 \end{bmatrix}.$$

Remember that the singular values are *ordered*, from large to small, and any good SVS software automatically orders the singular values in this way. In fact, this ordering of the singular values is one of the defining properties of the SVD — no ordering from large to small, no SVD.

Taking into account all the zero's in Σ, the SVD can be written in reduced form,

$$D = U\Sigma_+ V_+^T, \tag{3.7}$$

where U is unchanged (assuming $N > d$), Σ_+ is now the $d \times d$ diagonal matrix obtained from Σ as,

$$\Sigma_+ = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_d \end{bmatrix},$$

and V_+ is an $N \times d$ dimensional matrix, consisting of the first d columns of V. This is interesting: The dimension of V_+^T appearing in the reduced SVD is $d \times N$, exactly the same as the data D.

It will become clear that the three factors in the SVD provides us with important information about the data. Let us now investigate exactly what information about the data we can extract from the SVD. Accordingly, let us form the products, DD^T and D^TD. Keeping in mind that $UU^T = I = U^TU$, and $VV^T = I = V^TV$, it follows that

$$DD^T = U\Sigma\Sigma^T U^T \text{ and } D^TD = V\Sigma^T\Sigma V^T$$

so that

$$\left(DD^T\right)U = U\left(\Sigma\Sigma^T\right) \text{ and } \left(D^TD\right)\text{V=V}\left(\Sigma^T\Sigma\right).$$

Both $\Sigma^T\Sigma$ and $\Sigma\Sigma^T$ are square, *diagonal* matrices with the *squares* of the singular values on its diagonal. We now draw the important conclusion that U is the eigenvector matrix of the symmetric matrix DD^T, with eigenvalues the squares of the singular values. V is the eigenvector matrix of the symmetric matrix D^TD, and again its eigenvalues are the squares of the singular values. In order to understand the meaning of all of this, note that DD^T is almost the data covariance matrix that appeared in (3.1). All that is missing is a factor $\frac{1}{N}$ or $\frac{1}{N-1}$, depending on whether we use a biased or unbiased estimate. Specifically, the biased covariance matrix is given by $\frac{1}{N}DD^T$. If we compare with (3.1), the only difference is in the scaling of the eigenvalues. This difference in scaling is quite a nuisance since it makes the interpretation of the singular values not quite obvious. Fortunately, U in (3.7) and Q in (3.1) are exactly the same.

The first important thing we learn from the SVD is that the principal directions of the data is given by U in (3.7).

We demonstrated that the eigenvalues of the covariance matrix (3.1) are the variances along the principal directions. These eigenvalues differ by the scaling factor $\frac{1}{N}$ from the squares of the singular values.

The second important fact we learn from the SVD is that $\frac{1}{\sqrt{N}}\Sigma_+$ are the standard deviations along the principal directions.

But we are not done yet with the magic of the SVD. Let us return to the reduced SVD given by (3.7), but now rewritten as

$$U^T D = I\Sigma_+ V_+^T.$$

Let us concentrate on the left-hand-side $U^T D$. Since we multiply the data from the left, it means that we have now transformed the original data D to a new set of data D'. So what exactly is this new data D'? Let us calculate its SVD. But wait a minute, we already have it,

$$D' = I\Sigma_+ V_+^T.$$

The principal directions of the transformed data are just the columns of I, the identity matrix. Therefore the principal directions of the transformed data D' are aligned with the coordinate axes. The new data

$$D' = \Sigma_+ V_+^T$$

is therefore just the old data rotated so that its principal directions are aligned with the coordinate axes. And the standard deviations along the principal directions are again given by $\frac{1}{\sqrt{N}}\Sigma_+$. We get this almost for free from the SVD.

Let us continue and transform D' by dividing with the largest singular value σ_1,

$$\frac{1}{\sigma_1}D' = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\sigma_d}{\sigma_1} \end{bmatrix} V_+^T.$$

This is just a scaled version of D' so that the largest singular value equals one. If we need the largest standard deviation to be one, we still need to *multiply* the right-hand-side by $\sqrt{N}$, since we actually need to divide by $\frac{\sigma_1}{\sqrt{N}}$.

This is still not the end of the magic. Let us assume that all the singular values are non-zero so that we can multiply with the inverse of Σ_+ to get

$$\Sigma_+^{-1} D' = V_+^T.$$

Since we can also rewrite this as,

$$\Sigma_+^{-1} U^T D = V_+^T,$$

it is clearly another transformation of the original data. In order to see what it means let us write down its SVD explicitly,

$$\Sigma_+^{-1} U^T D = IIV_+^T.$$

For this data both the U and Σ matrices are the identities. This means two things:

(1) The principal directions are aligned with the coordinate axes, i.e. the data is rotated.
(2) All the singular values are equal to one. This means that the data has been reshaped to be spherical. And if we need to standard deviations to be unity, we need to multiply the right-hand-side with $\sqrt{N}$.

This is worth repeating: The factor V^T returned by the SVD, is just the *whitened* data!

Even this is not quite everything that the SVD reveals about the data. Let us now assume that only $r \leq d$ of the singular values are non-zero. Since the singular values measure the spread in specific directions, a zero singular value tells us that the spread in that direction is zero. This means that the data is not truly d-dimensional, but rather r-dimensional. r is therefore called the *rank* of the data, in which case the reduced SVD becomes

$$D = U_r \Sigma_r V_r^T,$$

where U_r is and $d \times r$ matrix consisting of the first r columns of U, V_r is an $r \times N$ dimensional matrix consisting of the first r columns of V, and Σ_r is the $r \times r$ diagonal matrix consisting of the *nonzero* singular values.

You now need to answer the following important questions:

(1) What does it mean of we transform the original data as $D' = U_r^T D$?
(2) What does it mean if we transform the original data as $\frac{1}{\sigma_1} D' = \frac{1}{\sigma_1} U_r^T D$?
(3) What does it mean if we transform the original data as $\Sigma_r^{-1} D' = \Sigma_r^{-1} U_r^T D$?

Note that there is nothing to prevent us from keeping less singular values than the rank. Say we keep ν singular values and form a new data set

$$\mathcal{D}_\nu = U_\nu \Sigma_\nu V_\nu^T,$$

where U_ν is and $d \times \nu$ matrix consisting of the first ν columns of U, V_ν is an $\nu \times N$ dimensional matrix consisting of the first ν columns of V, and Σ_ν is the $\nu \times \nu$ diagonal matrix consisting of the ν singular values. In this case $\mathcal{D}_\nu$ is an approximation of the original matrix, and the quality of the approximation depends only on $\sigma_{\nu+1}$, the first discarded singular value. If $\nu = r$, then $\sigma_{\nu+1} = 0$. More precisely, the difference between D and $\mathcal{D}_\nu$ is given in the Frobenious norm by

$$\|D - \mathcal{D}_\nu\|_F = \sigma_{\nu+1}.$$

We are therefore justified to think of the fraction of the original data described by the first ν principal components as

$$\frac{\sigma_{\nu+1}}{\sigma_1}.$$

Finally, note that the eigenvalue problem of a *covariance* matrix (3.1), written as,

$$S = Q \Lambda Q^T$$

is just the SVD of S.

Question: What is special about a covariance matrix, why is this not true for any symmetric matrix S?

3.3. Numerical Calculation.

In practice it is usually a good idea to scale the data in order to ensure that one does not run into numerical instabilites. For example, the data is often scaled so that the variance along the first principal axis is one. This is achieved by transforming the data according to

$$\mathbf{y}_n = \frac{1}{\sqrt{\lambda_1}} Q_r^T (\mathbf{x}_n - \overline{\mathbf{x}}) \;\; n = 1, \ldots, N,$$

where λ_1 is the largest eigenvalue of the sample covariance matrix.

It is in general not a good idea to calculate the eigenvalues of the sample covariance matrix numerically since this procedure is unstable. More precisely, by calculating the covariance matrix we loose roughly half the available significant digits, and this can cause problems. An alternative and more stable way is to form the data matrix

$$X = \frac{1}{\sqrt{N}} \left[\begin{array}{ccc} \mathbf{x}_1 - \overline{\mathbf{x}} & \cdots & \mathbf{x}_N - \overline{\mathbf{x}} \end{array} \right]$$

and calculate its SVD, $X = U\Sigma V^T$. Let us remind the reader that the singular values of X are the square roots of the eigenvalues of the sample covariance matrix S, i.e. $\sigma_n^2 = \lambda_n$, and the columns of U are the eigenvectors of S. Thus the singular values σ_n measure the standard deviation along the principal directions, and the eigenvalues λ_n the variances.

Let us now explain in little more detailed how dimensionality reduction is achieved. If we choose $r = \text{ rank}(X)$, i.e. r equals the number of nonzero singualr values, then the first r columns of U form an orthogonal basis for the column space of X. Since the singular values are the standard deviations along these principal directions it measures to what extent a particular principal direction is 'occupied' by the data. A zero singular value implies an empty dimension without data representation. Projecting our data onto the first r columns of U therefore entails no loss in information. It squeezes out the 'empty' dimensions. In practice a system often benefits from also squeezing the directions with little data representation, i.e. the directions associated with small singular values. A simple example may be helpful. Suppose we know that our 2D data is supposed to fall on a straight line. If we calculate the principal components of this data, we certainly expect that the first principal direction should point in the direction of the line (can you see why it is important to remove the mean?). If life were perfect the second singular value should be zero. But of course there are small measurement and possibly other errors. This means that the data does not fall exactly on the line, with the result that the second singular value measures the extent of the noise—the deviation from the straight line. The fact that the second singular value is small, is probably an indication that we are dealing with noise in which case it is perfectly reasonable to project the data back onto the space (line) defined by the first principal component. Note that this amounts to calculating the linear least squares approximation.

Let us now consider the higher dimensional case. The singular values tell us which of the directions we can safely squeeze out. Let us say for our application only the first r singular values are significant. We then project the data onto the r-dimensional subspace defined by the orthogonal basis consisting of the first r columns of U. Let us call this matrix U_r. Given a data value $\mathbf{x}$ its projection onto U_r is given by $\mathbf{z}$ where $\mathbf{x} - \overline{\mathbf{x}} = U_r\mathbf{z}$ in a least squares sense. Since U_r has orthogonal columns, the least squares solution gives us the projection,

$$\mathbf{z} = U_r^T(\mathbf{x} - \overline{\mathbf{x}}).$$

It sometimes happens that we have so much data that it is just not possible to form the data matrix explicitly as required by the SVD approach. This is often the case in speech processing, for example. In that case one has no choice but to form the covariance matrix and calculate its eigenvalues and eigenvectors. This can be done recursively. Suppose we have a set of $n-1$ data values, $\mathbf{x}_j, \quad j = 1, \ldots, n-1$, with mean μ_{n-1} and sample covariance S_{n-1}. We now add another value $\mathbf{x}_n$ to the set. We want to construct the new mean and covariance that include the new data value, from the known values μ_{n-1} and S_{n-1}. It is left as an exercise to derive the following formulas,

$$\begin{aligned}
\mu_n &= \mu_{n-1} + \frac{1}{n}\left(\mathbf{x}_n - \mu_{n-1}\right), \\
S_n &= \frac{n-1}{n}S_{n-1} + \frac{n-1}{n}\left(\mu_n - \mu_{n-1}\right)\left(\mu_n - \mu_{n-1}\right)^T + \frac{1}{n}\left(\mathbf{x}_n - \mu_n\right)\left(\mathbf{x}_n - \mu_n\right)^T.
\end{aligned}$$

It is interesting to briefly look at the structure of these equations. If it so happens that the new data value equals the previous mean $\mathbf{x}_n = \mu_{n-1}$ then one might be tempted to think that this does not add any information about the data set; the mean for example, remains unchanged and the final two terms on the right hand side of the expression for the updates sample covariance become zero. Therefore the new sample covariance is slightly smaller, $S_n = \frac{n-1}{n} S_{n-1}$, than the previous one. This means that the data is slightly better clustered. One can also see that for large n, the updates become negligible, unless the new data value differs substantially from the previous mean.

3.4. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is closely associated with the classification problem. In this problem each data value or observation is assigned to one of say, k *classes*. For example, the observation can be one of the ten digits, $0, 1, \ldots, 9$, and we need to decide which one. Taken literally, this statement is often too harsh—we do not necessarily want to assign an observation to a specific class, certainly not if there is good evidence that it can also belong to a different class. We therefore prefer to do a soft assignment, where we assign a *probability* to each class. Let us assume that we are dealing with k different classes and we are presented with an observation $\mathbf{x}$, then we want to calculate the k different probilities

$$P(\mathcal{C}_j|\mathbf{x}),$$

where $\mathcal{C}_j, \quad j = 1, \ldots, k$, indicate k different classes. The simplest way of constructing the model $P(\mathcal{C}_j|\mathbf{x})$ is to learn it from labeled data $\mathcal{D}$ consisting of N observations, where each observation consists of a data value $\mathbf{x}_n$ and a class label t_n. In a later chapter we describe how to construct models such as these. In this section we are interested in manipulating the data so that the classification task becomes easier. In particular, we want to reduce the *dimension* of the data values $\mathbf{x}_n$ in such a way that maximum class separation is obtained in the lower dimensional space.

3.4.1. Two-class LDA. Given labeled data, i.e. each observation $\mathbf{x}_n \in \mathbb{R}^d$ is provided with a corresponding class label t_n, we first consider the case where we have only two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Given a d-dimensional observation $\mathbf{x}_n$, it is projected down to one dimension using,

$$y_n = \mathbf{w}^T \mathbf{x}_n,$$

where $\mathbf{w}$ has to be chosen in such a way as to preserve maximum separation between the classes in the projected one-dimensional space. One possibility is to maximize the separation between the projected class means. This idea is illustrated in Figure 3.4.1. Calculating the means of the two classes,

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n,$$

we want to maximally separate the projected class means, i.e. we want to choose $\mathbf{w}$ so as to maximize

$$m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1),$$

where $m_k = \mathbf{w}^T \mathbf{m}_k$. Thus we want to maximize $\mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$ subject to the constraint. Introducing a Lagrange multiplier λ, we therefore maximize

$$L(\mathbf{w}) = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(\mathbf{w}^T\mathbf{w} - 1),$$

subject to $\mathbf{w}^T\mathbf{w} = 1$. From $\frac{\partial L}{\partial w_n} = 0$ follows

$$(m_{2n} - m_{1n}) + 2\lambda w_n = 0, \; n = 1, \ldots, D.$$

subject to $\mathbf{w}^T\mathbf{w} = 1$. In vector notation this looks even better,

$$\mathbf{m}_2 - \mathbf{m}_1 + 2\lambda\mathbf{w} = \mathbf{0},$$

or

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1.$$

This means that $\mathbf{w}$ points in the direction of the vector connecting the two class means, implying that the decision boundary is orthogonal to this direction, see Figure 3.4.1(a).

This is not the best one can do, as seen in Figure 3.4.1(b). Not only do we want to maximize the separation of the projected class means but, simultaneously, also minimize class overlap by minimizing the total projected within-class scatter. The

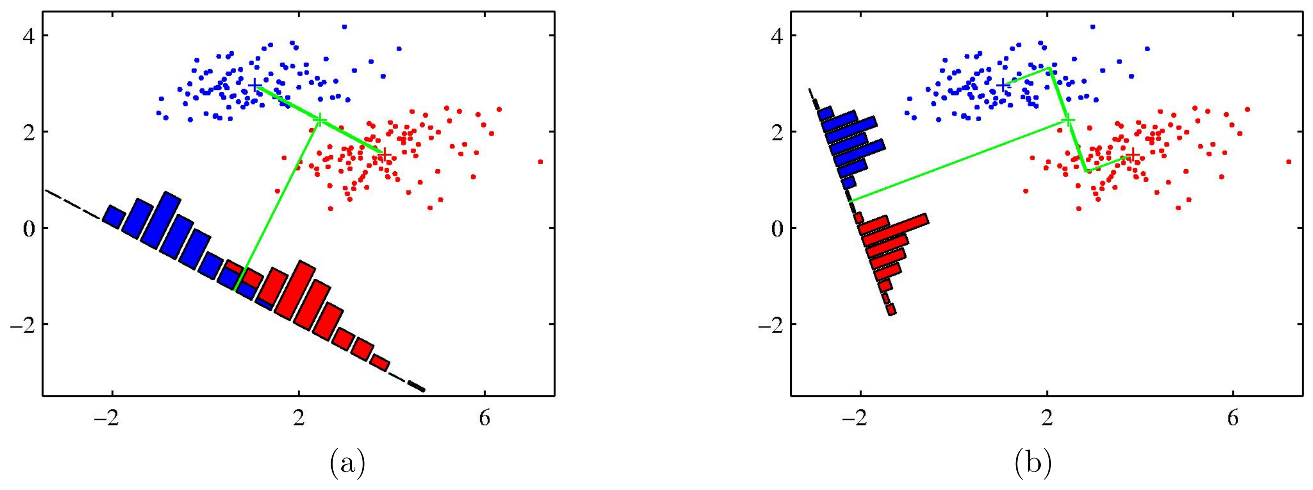


FIGURE 3.4.1. Projection onto the line joining the class means.

within-class scatter for each of the two classes is given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2,$$

where $y_n = \mathbf{w}^T \mathbf{x}_n$, and the total within-class scatter by $s_1^2 + s_2^2$. The Fisher criterion is to maximize $J(\mathbf{w})$ where

$$\begin{aligned} J(\mathbf{w}) &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}. \\ &= \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}, \end{aligned}$$

where

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

and

$$S_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T.$$

Calculating the gradient $\nabla J(\mathbf{w}) = \mathbf{0}$, it follows that $J(\mathbf{w})$ is maximized when

$$(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}.$$

Note that since we are only interested in the direction of $\mathbf{w}$ not its magnitude, we can drop the scalar factors. Also $S_B \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$ points in the direction of $\mathbf{m}_2 - \mathbf{m}_1$ and we get that

$$\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1).$$

The attentive reader would have noted that the within-class scatter for each class is not normalized by the number of points in that class. This means that less weight is attached to classes with fewer members.

3.4.2. LDA for multiple classes.

Given a training set $\mathbf{x}_n \in \mathbb{R}^d, \ \ n = 1, \ldots, N$ consisting of N observations, each of dimension d, and each one with class labels t_n indicating one of k different classes, the idea is to project the data onto a d'-dimensional subspace, through,

$$\mathbf{y}_n = W^T \mathbf{x}_n, \ \ n = 1, \ldots, N, \tag{3.1}$$

where W is an $d \times d'$ matrix, written in terms of its columns as,

$$W = \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_{d'} \end{bmatrix}.$$

Our task is to find the W that ensures maximum class separation in the projected space. Formally one finds a measure $J(W)$ of the class separation. Once $J(W)$ is known it becomes a matter of finding the optimal W^*,

$$W^* = \arg \max_W J(W).$$

Instead of constructing $J(W)$ explicitly however, we proceed intuitively relying on geometric intuition. Similarly as in the two-class problem we first define the total within— and between-class scatters. The total within-class scatter is defined by

$$S_w = \sum_{n=1}^{k} P_n \Sigma_n \tag{3.2}$$

where Σ_n is the sample class covariance for class n and P_n is the prior class probability, $P_n = \frac{N_n}{N}$, with N_n the number of data values belonging to class n.

If $\mathbf{m}$ is the mean of the total data set,

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \sum_{n=1}^{k} P_n \mathbf{m}_n, \tag{3.3}$$

where $\mathbf{m}_n$ is the mean of class n, the between-class scatter is defined as

$$S_b = \sum_{n=1}^{k} P_n(\mathbf{m}_n - \mathbf{m})(\mathbf{m}_n - \mathbf{m})^T. \tag{3.4}$$

As in the case of the two-class problem, the idea is to minimize the within-class scatter while simultaneously maximizing the between-class scatter. Geometrically, this is achieved by whitening the within-class covariance, i.e. find a transformation W such that $W^T S_w W = I$, while simultaneously diagonalizing the between-class covariance, $W^T S_b W = D_b$, where D_b is a diagonal matrix. (Note: The notation is no accident—as demontrated below, it turns out that the transformation W that simultaneously diagonalizes both S_w and S_b is exactly what is needed in (3.1).) This proceeds in two steps. First the within-class scatter is minimized by whitening S_w, i.e. S_w is transformed to the identity matrix. Thus the within-class scatter is minimized by transforming the data in such a way that the variances along all the coordinate axes become unity. Then we calculate the between-class scatter S_b of the transformed data by applying the same transformation to S_b. This has the effect that the between-class scatter is exagerated in the direction of minimum within-class scatter. The final step is to rotate the coordinate axes of the transformed data so that they align with the principle directions defined by the transformed between-class scatter. Since the total within-class scatter of the transformed data is already spherical, it is not changed by a rotation of the axes. This final rotation however, identifies the directions of maximum variance of the between-class scatter of the transformed data. These are therefore the desired axes onto which the data is projected.

Exercise 42. Is it in general possible to simultaneously diagonalize more than two covariance matrices?

Let us now make an algorithm out of this procedure:

(1) Whiten the within-class scatter S_w by using the spectral decomposition[1] of S_w,

$$S_w = Q\Lambda Q^T,$$

where Q is the eigenvector matrix of S_w and Λ the corresponding diagonal eigenvalue matrix. We also want to remove the 'empty' dimensions, i.e. the directions in which the variances are zero. For this purpose we conveniently arrange things[2] so that the eigenvalues are ordered from large to small, $\lambda_{n+1} \geq \lambda_n$. Assuming that the rank of S_w is r (defined as the number of nonzero eigenvalues), define Q_r as the $d \times r$ matrix consisting of the first r columns of Q, and Λ_r as the diagonal $r \times r$ matrix with the first r eigenvalues on its diagonal. One can then write

$$S_w = Q_r \Lambda_r Q_r^T.$$

Thus, if S_w is singular, it is whitened by

$$\Lambda_r^{-\frac{1}{2}} Q_r^T S_w Q_r \Lambda_r^{-\frac{1}{2}} = I_r.$$

It might be informative to confirm that what we have obtained the correct transformation by applying it directly to the data,

$$\mathbf{z}_j = \Lambda_r^{-\frac{1}{2}} Q_r^T \mathbf{x}_j, \quad j = 1, \ldots, N. \tag{3.5}$$

The within-class covariance of the transformed data $\mathbf{z}_j$, is given by

$$S_w^z = \sum_{n=1}^{k} P_n \Sigma_n^z,$$

[1]If S_w is non-singular, it is computationally more efficient to whiten S_w by factorizing $S_w = LL^T$ by using for example, the Cholesky factorization, i.e.

$$L^{-1} S_w L^{-T} = I_d,$$

where I_d is the $d \times d$ identity matrix.

[2]Although computationally not particularly efficient, you may want to consider using the SVD of your favorite software library. In this case the SVD is the same as the spectral decomposition. The main advantage is that you don't need to order the eigenvalues, the SVD does it automatically.

with the transformed class covariance,

$$\Sigma_n^z = \frac{1}{N_n}\sum_{j=1}^{N_n}(\mathbf{z}_j - \overline{\mathbf{z}}_n)(\mathbf{z}_j - \overline{\mathbf{z}}_n)^T, \quad n = 1, \ldots, k,$$

and the transformed class mean,

$$\overline{\mathbf{z}}_n = \frac{1}{N_n}\sum_{j=1}^{N_n}\mathbf{z}_j,$$

or

$$\overline{\mathbf{z}}_n = \Lambda_r^{-\frac{1}{2}} Q_r^T \overline{\mathbf{x}}_n.$$

Since $\mathbf{z}_j$ and $\overline{\mathbf{z}}_n$ are given in terms of the original variables by (3.5) for $j = 1, \ldots, N_n$ and $n = 1, \ldots, k$, respectively, it follows that

$$\Sigma_n^z = \Lambda_r^{-\frac{1}{2}} Q_r^T \Sigma_n Q_r \Lambda_r^{-\frac{1}{2}},$$

and

$$\begin{aligned} S_w^z &= \Lambda_r^{-\frac{1}{2}} Q_r^T S_w Q_r \Lambda_r^{-\frac{1}{2}} \\ &= I_r. \end{aligned}$$

Thus we have shown that the transformation (3.5) derived from the total with-class scatter, transforms the data in a such a way that the total within-class scatter of the transformed data is whitened.

(2) We now calculate the between-class scatter S_b^z of the transformed data, using (3.5). Of course there is no need to actually calculate the transformed data since we can directly calculate the transformed between-class scatter,

$$S_b^z = \Lambda_r^{-\frac{1}{2}} Q_r^T S_b Q_r \Lambda_r^{-\frac{1}{2}}.$$

Note that S_b^z is an $r \times r$ matrix with $r = d$ if S_w is nonsingular.

(3) Since S_b^z is also a covariance matrix, we 'rotate' it so that the principal directions coincide with the coordinate axes. Accordingly, we diagonalize S_b^z using its orthogonal eigenvector matrix U_b, i.e. $S_b^z = U_b D_b U_b^T$ where

$U_bU_b^T = I_r = U_b^TU_b$. This means that $U_b^TS_b^zU_b = D_b$, where D_b is the diagonal eigenvalue matrix.

(4) Substituting the expression for S_b^z, it follows that

$$U_b^T\Lambda_r^{-\frac{1}{2}}Q_r^TS_bQ_r\Lambda_r^{-\frac{1}{2}}U_b = D_b.$$

It is easy to verify that, rotating the transformed data by U_b, leaves the transformed within-class covariance S_w invariant,

$$U_b^T\Lambda_r^{-\frac{1}{2}}Q_r^TS_wQ_r\Lambda_r^{-\frac{1}{2}}U_b = U_b^TI_rU_b = I_r.$$

(5) For $W = Q_r\Lambda_r^{-\frac{1}{2}}U_b$ we therefore find that $W^TS_wW = I_r$ and $W^TS_bW = D_b$ as required.

The most important observation is that W *is exactly the transformation matrix needed for dimensionality reduction*, as alluded to above. We have therefore arrived at an expression for W in (3.1),

$$W = Q_r\Lambda_r^{-\frac{1}{2}}U_b, \tag{3.6}$$

where we use the expression appropriate for squeezing out the 'empty' dimensions corresponding to zero singular values. Of course if S_w is not singular, Q_r and Λ_r are simply the full eigenvector— and eigenvalue matrices respectively, of S_w.

In summary, the projection is given by $\mathbf{y}_n = W^T\mathbf{x}_n$ where $\mathbf{x}_n \in \mathbb{R}^d$, where

$$W^T = U_b^T\Lambda_r^{-\frac{1}{2}}Q_r^T.$$

Since Q_r is an $d \times r$ matrix where r is the rank of S_w, W^T is an $r \times d$ matrix. Thus apart from squeezing out the 'empty' dimensions of S_w we have not yet achieved any dimensionality reduction. For that we need to investigate the eigenvalues of U_b. Since we want to project down to d' dimensions, we select the eigenvectors in U_b belonging to the d' largest eigenvalues to form an $r \times d'$ matrix $U_{d'}$. The transformation matrix therefore becomes

$$W^T = U_{d'}^T\Lambda_r^{-\frac{1}{2}}Q_r^T. \tag{3.7}$$

It is interesting to note that there is an upper limit on the choice of d', namely $d' \leq k-1$, where k is the number of classes. This is because of the fact that the maximum rank of S_b is $k-1$. To see this, note that S_b is the sum of k rank-one matrices that satisfy a single constraint (3.3). More specifically, (3.4) tells us that S_b is a linear combination of rank one matrices based on the vectors $\mathbf{m}_n - \mathbf{m}, \ \ n = 1, \ldots, k$ satisfying the constraint (3.3). The constraint implies that the vectors $\mathbf{m}_n - \mathbf{m}, \ \ n = 1 \ldots, k$ are linearly dependent (by noting that $\sum_{n=1}^{k} P_n (\mathbf{m}_n - \mathbf{m}) = 0$). Therefore the rank of S_b is at most $k-1$, and S_b therefore has at most $k-1$ nonzero eigenvalues. Also note that the only way in which one can attain $d' = k - 1$ is if the orginal dimension of the problem satisfies $d \geq k$.

CHAPTER 4

CLASSIFICATION

4.1. Introduction.

The basic problem addressed in this chapter is, given an observation $\mathbf{x}$ assign it to one of k classes $\mathcal{C}_j, \quad j = 1, \ldots, k$. We are looking for an answer in terms of a probability $P(\mathcal{C}_j|\mathbf{x})$. We might for example, want to assign $\mathbf{x}$ to $\mathcal{C}^\star$ where

$$\mathcal{C}^\star = \arg\max_{\mathcal{C}_j} P(\mathcal{C}_j|\mathbf{x}).$$

More generally, we might not want to pick the class with highest probability, since they often is a cost involved for a wrong classification. Think for example of spam classification where the consequences of classifying an important message as spam, can be serious. The same is true if is medical test fails to diagnose a serious disease. Dealing with the cost of classification errors belongs to decision theory, which makes use of a so-called loss function. The key insight is that the actual probabilities of each class are useful beyond just providing the maximum class probability.

Thus the problem is reduced to constructing the model $P(\mathcal{C}_j|\mathbf{x})$, given training data $\mathcal{D}$. In this chapter it is assumed that the training data $(\mathbf{x}_j, y_j), \quad j = 1, \ldots, N$, is fully observed, i.e. each observation $\mathbf{x}_j$ comes with a class label y_j where $y_j = \mathcal{C}_n$ if $\mathbf{x}_j$ belongs to $\mathcal{C}_n$. Given fully observed data one can follow one of two important routes in constructing $P(\mathcal{C}_j|\mathbf{x})$—one can follow a *generative*, or a *discriminative* approach.

For the generative approach a model is developed for each class, $p(\mathbf{x}|\mathcal{C}_j)$ from the observations known to belong to that class[1]. Once this model is known it is in principle possible to generate observations for each class, hence the name, *generative*

[1]Note that a more precise way of defining generative models, is to develop a model for the *joint* distribution $p(\mathbf{x}, y)$. From the joint distribution one can form the class-conditional distribution $p(\mathbf{x}|y)$, i.e. one can generate data for each class.

approach. Introducing a class prior $P(\mathcal{C}_j)$, the posterior is then given by

$$P(\mathcal{C}_j|\mathbf{x}) \propto p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j). \tag{4.1}$$

The alternative, *discriminative* approach dispenses with the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_j)$, and directly estimates the posterior $P(\mathcal{C}_j|\mathbf{x})$ from the data. This has the advantage that the data is used to best effect to discriminate between the classes, and not 'waste' data to estimate class-conditional models that may not be needed anyway. Thus the available training data is used more effectively in distinguishing between the classes. It will become clear however that it is more complicated to train than the generative models.

One major difference between generative and dicriminative models is that in the case of generative models, a model is trained for each class, *totally ignoring the properties of the other classes.* Thus one only uses the training data of each class to learn that specific class. This is what makes the training quite straightforward. In the case of discriminative models, *all the training data* is used simultaneously to build the model. Since the information of all the classes is utilized during the training stage, it can be used to good effect to determine the *differences* between the classes. This however, makes the training harder, as alluded to above.

4.2. Probabilistic Generative Models.

Recall from the Introduction that we are studying a system where each observation in the training set is provided with a class label $\mathcal{C}_j$. Thus the training set is given by $(\mathbf{x}_j, y_j), \quad j = 1, \ldots, N$, where $y_j = \mathcal{C}_n$ if $\mathbf{x}_j$ belongs to class $\mathcal{C}_n$.

In order to use the posterior (4.1), a class-conditional model $p(\mathbf{x}|\mathcal{C}_j)$ is required. We will look into more general model later, for now we fix the ideas assuming a normal distribution for each class. Using the training data specific for each class, one estimates the class normal distribution by estimating the class mean and – covariance. In general, once an appropriate model is selected, the training data is used to estimate all the class-conditional densities $p(\mathbf{x}|\mathcal{C}_j), \quad j = 1, \ldots, k$. Given the priors $P(\mathcal{C}_j)$, Bayes' theorem then gives the posterior probabilities $P(\mathcal{C}_j|\mathbf{x})$. In the case of only two classes, for example, with $P(\mathcal{C}_2|\mathbf{x}) + P(\mathcal{C}_1|\mathbf{x}) = 1$,

$$\begin{aligned} P(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a(\mathbf{x}))} \\ &= \sigma(a(\mathbf{x})) \qquad (4.1) \end{aligned}$$

where $a(\mathbf{x})$ is the log posterior odds,

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)},$$

and the *logistic sigmoid* function is given by,

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

Note that the logistic function maps $a(\mathbf{x})$ to a value between 0 and 1. It therefore assigns a probability—the posterior probability—to each input value $\mathbf{x}$, see Figure~4.2.1. Note that the posterior odds provide us with a classification scheme. If we define

$$\text{odds} = \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)}, \qquad (4.2)$$

then $\mathbf{x}$ is assigned to $\mathcal{C}_1$ if odds > 1, otherwise it is assigned to $\mathcal{C}_2$.

For k classes we have

$$\begin{aligned} P(\mathcal{C}_n|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_n)P(\mathcal{C}_n)}{\sum_{j=1}^k p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} \\ &= \frac{\exp a_n(\mathbf{x})}{\sum_{j=1}^k \exp a_j(\mathbf{x})} \end{aligned}$$

where

$$a_j(\mathbf{x}) = \ln p(\mathbf{x}|\mathcal{C}_j) + \ln P(\mathcal{C}_j).$$

It is straightforward to verify that $\sum_{j=1}^k P(\mathcal{C}_j|\mathbf{x}) = 1$. This normalizing sum of exponentials is also known as the softmax function because it is a smoothed version of the 'max' function. If $a_n >> a_j, \;\; j \neq n$, then $P(\mathcal{C}_n|\mathbf{x}) \approx 1$ and $P(\mathcal{C}_j|\mathbf{x}) \approx 0$ for $j \neq n$.

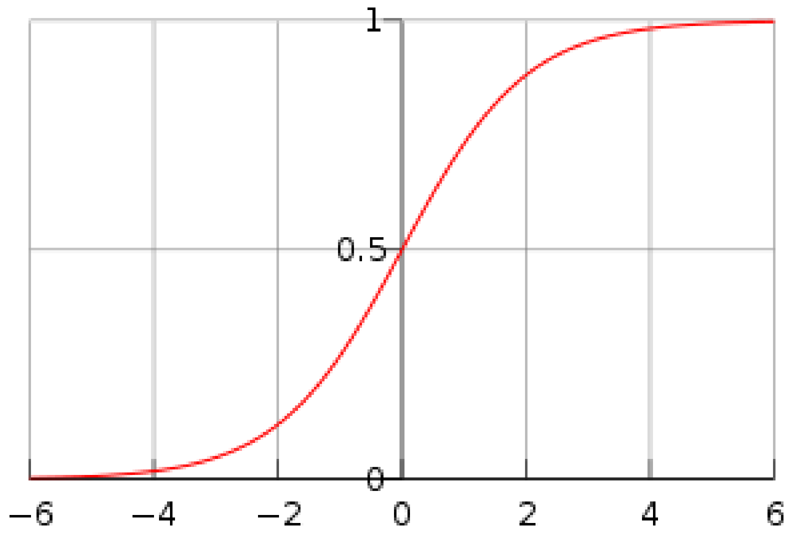


FIGURE 4.2.1. The logistic sigmoid function $\sigma(a)$.

It should be clear that in order to calculate the posterior probability, two things are needed, the prior class probabilities $P(\mathcal{C}_j)$ and the class-conditional deinsities, or models, $p(\mathbf{x}|\mathcal{C}_j)$. In the next paragraph we investigate the use of Gaussian class-conditional densities.

4.2.1. Gaussian class-conditional pdf's. Assume that all the class-conditional densities are Gaussian, sharing the *same* covariance matrix (but not the same means) so that the class-conditional densities are given by,

$$p(\mathbf{x}|\mathcal{C}_j) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)^T\right\}.$$

For the two-clas situation it follows that

$$P(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0) \tag{4.3}$$

where

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{4.4}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{P(\mathcal{C}_1)}{P(\mathcal{C}_2)}. \tag{4.5}$$

Note:

(1) The prior probabilities enter only through the bias term w_0.
(2) The quadratic terms in the Gaussians cancel because of the shared covariance. This makes it a *linear* classifier. To be more precise, we classify $\mathbf{x}$ as belonging to $\mathcal{C}_1$ if $P(\mathcal{C}_1|\mathbf{x}) > P(\mathcal{C}_2|\mathbf{x})$ otherwise to $\mathcal{C}_2$. The decision boundary is therefore given by $P(\mathcal{C}_1|\mathbf{x}) = P(\mathcal{C}_2|\mathbf{x}) = 1 - P(\mathcal{C}_1|\mathbf{x})$. This can be rewritten as $\sigma(\mathbf{w}^T\mathbf{x} + w_0) = 1 - \sigma(\mathbf{w}^T\mathbf{x} + w_0)$, or $\sigma(\mathbf{w}^T\mathbf{x} + w_0) = \frac{1}{2}$ so that $\mathbf{w}^T\mathbf{x} + w_0 = 0$.
(3) We have just achieved a remarkable result, one that will be fully exploited in the next section on logistic regression. For the time being, simply note the parameter reduction in (4.3). If we have d-dimensional data, then the two classes with different means but a shared covariance, require a total of $2d + \frac{1}{2}d(d+1)$ parameters. In addition one has to assign prior class probabilities. In (4.3) we combine the parameters in such a way that we are effectively left with only $d+1$ parameters, a significant savings. One can therefore ask whether one can side-step the class-conditional pdf's and directly estimate the $d+1$ parameters in (4.3). This is the idea behind logistic regression discussed in the next section, and in general, the idea behind *discriminative* models.

For k classes sharing the same covariance, it follows that

$$P(\mathcal{C}_n|\mathbf{x}) = \frac{\exp a_n(\mathbf{x})}{\sum_{j=1}^{k} \exp a_j(\mathbf{x})},$$

where

$$a_j(\mathbf{x}) = \mathbf{w}_j^T\mathbf{x} + w_{j0},$$

with

$$\begin{aligned} \mathbf{w}_j &= \Sigma^{-1}\boldsymbol{\mu}_j \\ w_{j0} &= \frac{1}{2}\boldsymbol{\mu}_j^T\Sigma^{-1}\boldsymbol{\mu}_j + \ln P(\mathcal{C}_j). \end{aligned}$$

Because of the shared covariance, the decision boundaries are linear. For non-shared covariance matrices the decision boundaries become quadratic, see Figure 4.2.2.

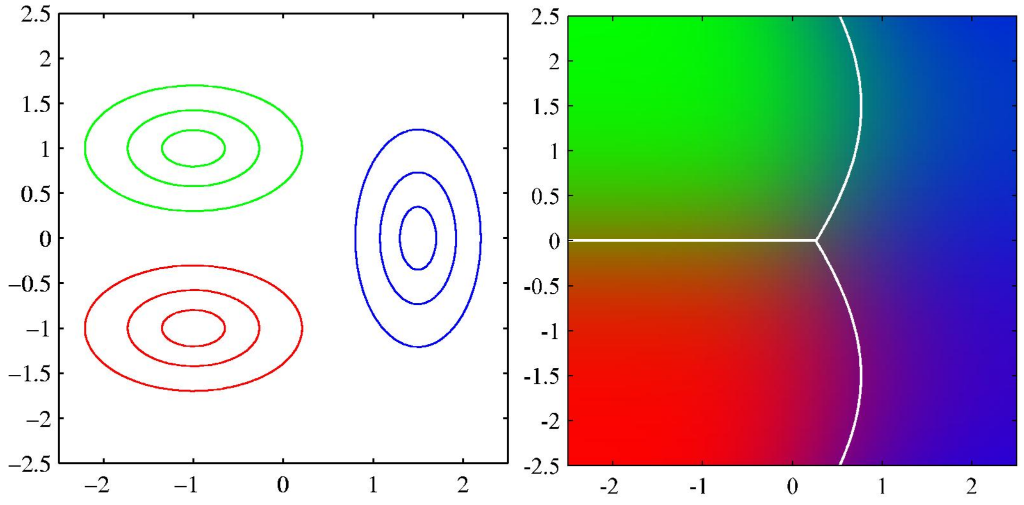


FIGURE 4.2.2. Quadratic discriminant functions.

4.2.2. Maximum likelihood solution. In order to find the parameters of the class conditional densities as well as the prior probabilities we need a fully observed set of observations, meaning observations as well as their class labels. Assuming two classes and Gaussian class-conditional density functions, we can estimate the mean and covariance of each class in the usual manner. If we also let $P(\mathcal{C}_1) = \pi$ and $P(\mathcal{C}_2) = 1 - \pi$, a maximum likelihood estimate gives

$$\pi = \frac{N_1}{N},$$

i.e. the prior probability of belonging to class $\mathcal{C}_1$ equals the fraction of the data points belonging to $\mathcal{C}_1$. The mean of the two classes are estimated by

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} y_n \mathbf{x}_n \text{ and } \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-y_n)\mathbf{x}_n,$$

where $y_n = 1$ if $\mathbf{x}_n \in \mathcal{C}_1$ and $y_n = 0$ if $\mathbf{x}_n \in \mathcal{C}_2$. The respective covariances are given by

$$\Sigma_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \text{ and } \Sigma_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T.$$

This simply means that the mean and covariance of each class are estimated using the data known to belong to that class.

The extension to k classes is straightforward.

4.2.3. Naive Bayes classifier. There is a computational difficulty with the previous approach—it is expensive. Assuming a k-class problem in d dimensions and a different, full covariance matrix for each class (leading to quadratic, not linear decision surfaces), one has to estimate $d + \frac{1}{2}d(d+1)$ parameters for each class, i.e. a total of $\frac{1}{2}kd(d+3)$ parameters for k classes. This becomes unwieldy for large d. One possibility is to share a full covariance between all the classes, as we did above. In this case the number of parameters is reduced to $kd + \frac{1}{2}d(d+1)$. This however, may not be 'rich' enough to provide a good descriptions of each class. Naive Bayes is somewhere in-between, essentially assuming a *diagonal* covariance for each class. This reduces the number of parameters to $2kd$, a substantial saving especially if d is large. More generally the naive Bayes assumption is that the attributes (the components of the random vector $\mathbf{x}$) are conditionally independent, conditioned on the class, i.e. if $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}^T$ then

$$p(\mathbf{x}|\mathcal{C}) = \prod_{n=1}^{d} p(x_n|\mathcal{C}).$$

Exercise 43. Show that the conditional independence assumption leads to a diagonal covariance matrix if one assumes a Gaussian class-conditional pdf.

Using Bayes' theorem we write

$$\begin{aligned} P(\mathcal{C}_j|\mathbf{x}) &= \frac{P(\mathcal{C}_j)p(\mathbf{x}|\mathcal{C}_j)}{p(\mathbf{x})} \\ &= \frac{P(\mathcal{C}_j)\prod_n p(x_n|\mathcal{C}_j)}{\sum_i P(\mathcal{C}_i)\prod_n p(x_n|\mathcal{C}_i)}. \end{aligned}$$

Once the posterior probabilities are known one then proceeds to build an appropriate classifier, i.e. a rule that assigns a given observation $\mathbf{x}$ to a specific class. The most naive classifier is given by

$$\mathcal{C}^\star = \arg\max_{\mathcal{C}_j} \frac{P(\mathcal{C}_j)\prod_n p(x_n|\mathcal{C}_j)}{\sum_i P(\mathcal{C}_i)\prod_n p(x_n|\mathcal{C}_i)}$$

which simplifies to (since the denominator does not depend on the class),

$$\mathcal{C}^{\star} = \arg\max_{\mathcal{C}_j} P(\mathcal{C}_j) \prod_n p(x_n|\mathcal{C}_j).$$

EXERCISE 44. Show, using maximum likelihood, and assuming Gaussian pdf's that

$$\begin{aligned} P(\mathcal{C}_j) &= \frac{N_j}{N} \\ \mu_{nj} &= \frac{1}{N_j}\sum x_{nj} \\ \sigma_{nj}^2 &= \frac{1}{N_j}\sum (x_{nj}-\mu_{nj})^2\,,\ n=1,\ldots,d;\ j=1,\ldots,k \end{aligned}$$

where N_j is the number of samples in class $\mathcal{C}_j$ and the sums go over all the samples x_{nj} that belong to class $\mathcal{C}_j$.

4.3. Probabilistic Discriminative Models.

4.3.1. Logistic Regression. Limiting ourselves for the moment to a discussion of the two-class problem, recall from (4.1) that the posterior probability is given by a logistic sigmoid function,

$$P(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1+\exp(-a(\mathbf{x}))} =: \sigma(a(\mathbf{x})) \tag{4.1}$$

where $a(\mathbf{x})$ is the log posterior odds,

$$a(\mathbf{x}) = \ln\frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)}.$$

Also recall that, assuming Gaussian densities with a shared covariance, the posterior probability is given by,

$$P(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}+w_0), \tag{4.2}$$

where the parameters are given by (4.4) and (4.5). In this case $a(\mathbf{x})$ is a linear function in $\mathbf{x}$, with the parameters derived from using Gaussian class conditionals. Upon closer inspection it should be clear that given $\mathbf{w}$, (4.1) and (4.2) directly map the input variable $\mathbf{x}$ to the posterior class probability $P(\mathcal{C}|\mathbf{x})$. There is no compelling reason for Gaussian class conditionals, or any class conditional for that matter. We could equally well assume a linear expression $a(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ and infer the parameters directly from the training data. In fact there is not even any particular reason to assume a linear relationship for $a(\mathbf{x})$ and we may assume the more general form,

$$P(\mathcal{C}_1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})), \tag{4.3}$$

where $\boldsymbol{\phi} : \mathbb{R}^d \longrightarrow \mathbb{R}^{d'}$, in which case $\mathbf{w} \in \mathbb{R}^{d'}$. For example, if $a(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ as above, then

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}.$$

Also note that $P(\mathcal{C}_2|\mathbf{x}, \mathbf{w}) = 1-P(\mathcal{C}_1|\mathbf{x}, \mathbf{w})$, where the conditioning on the parameters $\mathbf{w}$ is stated explicitly[2]. Since the decision boundary is at $P(\mathcal{C}_1|\mathbf{x}, \mathbf{w}) = P(\mathcal{C}_2|\mathbf{x}, \mathbf{w})$ it is given by

$$\mathbf{w}^T\phi(\mathbf{x}) = 0. \tag{4.4}$$

Our task is, given training data, determine the parameters $\mathbf{w}$ describing the decision boundary. Postponing a full Bayesean approach until late, we maximuize the likelihood. Accordingly, assume that we are given an training data set $\mathcal{D} : \{\mathbf{x}_n, y_n\}$, where the data is sampled independently from the same distribution and y_n indicates the class that $\mathbf{x}_n$ belongs to. To be more specific, let $y_n = 1$ and $y_n = 0$ indicate classes $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. It is also convenient to combine the observations and class labels by introducing the notation $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{y} = \{y_1, \ldots, y_N\}$. The training data set is then written as $\mathcal{D} = [X, \mathbf{y}]$, and the joint distribution given by,

$$\begin{aligned} p(\mathcal{D}|\mathbf{w}) = p(X, \mathbf{y}|\mathbf{w}) &= P(\mathbf{y}|X, \mathbf{w})p(X) \\ &= p(X)\prod_{n=1}^{N} P(y_n|\mathbf{x}_n, \mathbf{w}), \end{aligned} \tag{4.5}$$

[2]In fact, in a Bayes approach, we do think of the parameters as random variables in which case we even specify a prior distribution over $\mathbf{w}$.

where we assume that y_n is conditionally independent of the rest of the data, given $\mathbf{x}_n$, and X of course does not depend on $\mathbf{w}$. Since y_n satisfies a Bernoulli distribution we write,

$$P(y_n|\mathbf{x}_n, \mathbf{w}) = P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{y_n} (1 - P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{1-y_n},$$

where $P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})$ is given by (4.3). The likelihood therefore becomes,

$$p(\mathcal{D}|\mathbf{w}) = p(X) \prod_{n=1}^{N} P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{y_n} (1 - P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w}))^{1-y_n}, \tag{4.6}$$

or, using (4.3) and the notation $\phi_n := \phi(\mathbf{x}_n)$,

$$p(\mathcal{D}|\mathbf{w}) \;=\; p(X) \prod_{n=1}^{N} \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)^{y_n} \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right)^{1-y_n}. \tag{4.7}$$

We can now maximize the likelihood but even better, from a numerical stability point of view, minimixe the negative log-likelihood,

$$E(\mathbf{w}) := -\ln p(\mathcal{D}|\mathbf{w}) = -\sum_{n=1}^{N} \left\{ y_n \ln \sigma(\mathbf{w}^T \boldsymbol{\phi}_n) + (1 - y_n) \ln \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right) \right\} - \ln p(X). \tag{4.8}$$

One can simply ask your favorite optimizer to do the optimizations, but it is worth looking at the optimization in more detail. Since the optimal parameters satisfy $\boldsymbol{\nabla} E(\mathbf{w}) = \mathbf{0}$, we calculate the gradient,

$$\boldsymbol{\nabla} E(\mathbf{w}) = \sum_{n=1}^{N} \left(\sigma(\mathbf{w}^T \boldsymbol{\phi}_n) - y_n\right) \boldsymbol{\phi}_n,$$

where we have used the fact that

$$\frac{d\sigma}{da} = \sigma(1 - \sigma).$$

Thus $\mathbf{w}$ is given by the system of equations,

$$\sum_{n=1}^{N} \left(\sigma(\mathbf{w}^T \boldsymbol{\phi}_n) - y_n\right) \boldsymbol{\phi}_n = \mathbf{0}. \tag{4.9}$$

This is a nonlinear system of equations in $\mathbf{w}$ that has to be solved numerically. Since the gradient is readily available, a gradient-based optimization method can be applied directly to (4.8). Below we explain how the Newton-Raphson method can be used for this purpose.

It is important to realize that the system (4.9) 'sees' samples from both classes. This is necessary in order to find maximal class separation within the parameter space.

In order to develop some intuition of the mechanism that achieves maximal class separation, recall that the decision boundary separating the two classes is given by (4.4). Now imagine a data value $\mathbf{x}_n$ that is far from the decision boundary. If it belongs to class $y_n = 1$, the logistic function should assign a confidence $P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^T\phi(\mathbf{x}_n) \approx 1$ in All values $\mathbf{x}_n$ far from the decision boundary should be confidently assigned to their respective classes since there can be little doubt about their class labels. To be more specific, consider a data value $\mathbf{x}_n \in \mathcal{C}_1$, far from the decision boundary. For this value $P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)) \approx 1$. Since for $\mathbf{x}_n \in \mathcal{C}_1$ it is given that $y_n = 1$, it follows that $\sigma(\mathbf{w}^T\phi_n) - y_n \approx 0$ in (4.9). Thus this particular data value makes no contribution towards the system (4.9). Thus it is only the observations close to the decision boundary that contributes, and this is exactly how it should be—only those observations that are in doubt, contribute towards the exact placement of the decision boundary.

It should also be noted that the shape of the logistic function $\sigma(a)$ in Figure 4.2.1 and (4.3) shows that for $P(\mathcal{C}_1, |\mathbf{x}, \mathbf{w}) \longrightarrow 1$, $\mathbf{w}^T\phi(\mathbf{x}_n) \longrightarrow \infty$. The latter is easily achieved by simply choosing large values of $\mathbf{w}$, without changing the position of the decision boundary. It is therefore necessary to restrict its value. One possibility is to do a constrained optimization, by for example imposing $\mathbf{w}^T\mathbf{w} = 1$.

Another subtlety that is worth pointing out is that of specialization. Above we argued that it is mainly the observations close to the decision boundary that determine the decision boundary, i.e. the value of $\mathbf{w}$. Suppose we increase the complexity hence the flexibility of the model, by increasing the number of parameters d' for example. It is still only the observations close to the decision boundary that affects the values of the parameters and there is therefore a distinct danger that the more complex model specializes on the peculiarities of the training data in the vicinity of the decision boundary, to the detriment of generalization.

In our experience some kind of 'regularization' of discriminative schemes is essential in practice to avoid specialization or overfitting. In the next subsection we introduce a principled way to achieve this.

4.3.2. Full Bayesian Approach. In a full Bayesian approach we consider $\mathbf{w}$ as a random variable and calculate the posterior,

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathbf{w})p(\mathcal{D}|\mathbf{w}),$$

where $P(\mathcal{D}|\mathbf{w})$ is given by (4.5), and we might assume the following simple form for the prior,

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda I). \tag{4.10}$$

One can now follow a full Bayesian approach and marginalize over the parameters to find the predictive distribution directly

$$\begin{aligned}
P(\mathcal{C}_1|\mathbf{x},\mathcal{D}) &= \int P(\mathcal{C}_1, \mathbf{w}|\mathbf{x},\mathcal{D})d\mathbf{w} \\
&= \int P(\mathcal{C}_1|\mathbf{w},\mathbf{x},\mathcal{D})p(\mathbf{w}|\mathbf{x},\mathcal{D})d\mathbf{w} \\
&= \int P(\mathcal{C}_1|\mathbf{w},\mathbf{x})p(\mathbf{w}|\mathcal{D})d\mathbf{w},
\end{aligned}$$

assuming that the probability of class $\mathcal{C}_1$ becomes independent of the data $\mathcal{D}$ once the parameters $\mathbf{w}$ are known, and that the parameters are independent of the present measurement $\mathbf{x}$, given the training data $\mathcal{D}$. This has to be evaluated numerically, and although powerful methods such as MCMC are available for this purpose, it is in our opinion often reasonable to use the Maximum A Posteriori (MAP) estimate of the parameters by directly minimizing the negative log-posterior (**??**). Comparison with (4.8) shows that the prior over $\mathbf{w}$ introduces a regularization term, essential for preventing overfitting.

EXERCISE 45. Consider two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ with class conditional densities given by $p(x|\mathcal{C}_1) = \mathcal{N}(x|\mu_1 = -0.3, \sigma_1^2 = 0.8)$ and $p(x|\mathcal{C}_2) = \mathcal{N}(x|\mu_2 = 1.1, \sigma_2^2 = 0.8)$. The prior probabilities are the same, $P(\mathcal{C}_1) = P(\mathcal{C}_2) = \frac{1}{2}$. Plot both class-conditional densities on the same axes.

Suppose you want to classify two observations, $x_1 = 2.0$ and $x_2 = 0.2$. Calculate the likelihood $p(x|\mathcal{C}_2)$ for both data points. You should observe that both data points have the same likelihood. Can you conclude from this that it is equally likely that the two points belong to class $\mathcal{C}_2$? Plot the two data points on the same axes as the class-conditional densities. What does it tell you?

Calculate the likelihood ratios $\frac{p(x|\mathcal{C}_2)}{p(x|\mathcal{C}_1)}$[3] for both observed values. Compare this with the posterior probabilities $P(\mathcal{C}_1|x)$ and $P(\mathcal{C}_2|x)$ for the two observations. Is there any doubt about the classification of x_1?

Finally plot the two posterior probabilities $P(\mathcal{C}_1|x)$ and $P(\mathcal{C}_2|x)$ as functions of x, on the same axis as the class-conditional densities. What does it tell you about the classification of observations?

4.3.3. Newton's method for minimizing the negative log-likelihood.

Making use of (4.7) and (4.10) , the negative log-likelihood $\ell(\mathbf{w})$ is given by,

$$\begin{aligned}
\ell(\mathbf{w}) &:= -\ln\left[p(\mathbf{w})p(\mathcal{D}|\mathbf{w})\right] \\
&= -\ln p(\mathcal{D}|\mathbf{w}) - \ln p(\mathbf{w}) \\
&= -\sum_{n=1}^{N}\left\{y_n \ln\sigma(\mathbf{w}^T\boldsymbol{\phi}_n) + (1-y_n)\ln(1-\sigma(\mathbf{w}^T\boldsymbol{\phi}_n))\right\} + \frac{1}{2}\lambda\mathbf{w}^T\mathbf{w} + \text{const.}
\end{aligned}$$

Minimizing this quantity gives us the MAP estimate of $\mathbf{w}$, i.e.

$$\mathbf{w}_{\text{map}} = \arg\min \ell(\mathbf{w}).$$

Since the MAP estimate is obtained by setting the gradient of the negative log-likelihood equal to zero, $\boldsymbol{\nabla}\ell(\mathbf{w}) = \mathbf{0}$, one has to solve the nonlinear system of equations in $\mathbf{w}$,

$$\boldsymbol{\nabla}\ell(\mathbf{w}) = -\sum_{n=1}^{N}\left(y_n - \sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\right)\boldsymbol{\phi}(\mathbf{x}_n) + \lambda\mathbf{w} = \mathbf{0}.$$

Newton's method solves this nonlinear system of equations in $\mathbf{w}$ by iterating,

$$H(\mathbf{w}_k)\left(\mathbf{w}_{k+1} - \mathbf{w}_k\right) = -\boldsymbol{\nabla}\ell(\mathbf{w}_k), \quad k = 0, 1, \ldots,$$

[3]Bayesians call this quantity the Bayes factor.

where $H(\mathbf{w}_k)$ is the Hessian of $\ell(\mathbf{w})$. The ij component of the Hessian is given by h_{ij} where

$$h_{ij}(\mathbf{w}) = \sum_{n=1}^{N} \sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\left(1-\sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\right)\phi_i(\mathbf{x}_n)\phi_j(\mathbf{x}_n) + \lambda\delta_{ij}.$$

Thus the Hessian is given by

$$H(\mathbf{w}) = \sum_{n=1}^{N} \sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\left(1-\sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\right)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T + \lambda I,$$

where I is the identity matrix.

The only remaining question is whether Newton's method actually produces a minimum of the negative log-likelihood, instead of an extremum. For this we have to show that the Hessian is positive definite. All that is required is to show that $\mathbf{z}^T H(\mathbf{w})\mathbf{z} > 0$, for all $\mathbf{z} \neq \mathbf{0}$. Using the notation, $\sigma_n := \sigma\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)$, it follows that

$$\begin{aligned}\mathbf{z}^T H(\mathbf{w})\mathbf{z} &= \sum_{n=1}^{N} \sigma_n\left(1-\sigma_n\right)\mathbf{z}^T\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T\mathbf{z} + \lambda\left\|\mathbf{z}\right\|^2 \\ &= \sum_{n=1}^{N} \sigma_n\left(1-\sigma_n\right)\left\|\boldsymbol{\phi}_n^T\mathbf{z}\right\|^2 + \lambda\left\|\mathbf{z}\right\|^2.\end{aligned}$$

Since $0 < \sigma_n < 1$, it follows that $\mathbf{z}^T H(\mathbf{w})\mathbf{z} > 0$, i.e. $H(\mathbf{w})$ is positive definite. The fact that $H(\mathbf{w})$ is everywhere positive definite (not only in the vicinity of the minimum), has another important implication—it tells us that there is only one global minimum. This is one of the fortunate situations where there is no concern about multiple optima. It also implies that Newton's method will converge, and converge fast, for any choice of initial values.

4.3.4. Multiclass logistic regression. For k classes we need k models

$$\begin{aligned}P(\mathcal{C}_i|\mathbf{x},\mathbf{w}_i) &= \frac{\exp\left(\mathbf{w}_i^T\boldsymbol{\phi}(\mathbf{x})\right)}{\sum_{j=1}^{k}\exp\left(\mathbf{w}_j^T\boldsymbol{\phi}(\mathbf{x})\right)} \\ &= \frac{\exp\left(a_i(\mathbf{x})\right)}{\sum_{j=1}^{k}\exp\left(a_j(\mathbf{x})\right)},\ n = 1,\ldots,k,\end{aligned}$$

where $a_i(\mathbf{x}) = \mathbf{w}_i^T \boldsymbol{\phi}(\mathbf{x})$. Given data $\mathbf{x}_n$, $\mathbf{t}_n$, $\; n = 1, \ldots, N$, we use a 1-of-k coding scheme, i.e. the j-th element $(j = 1, \ldots, k)$ of $\mathbf{t}_n$, $t_{jn} = 1$ if $\mathbf{x}_n$ belongs to $\mathcal{C}_j$ with the rest of the elements, $t_{in} = 0, \;\; i \neq j$. We derive a suitable likelihood function from the joint distribution

$$p(X, T|\mathbf{w}) \;\; = \;\; p(X) \prod_{n=1}^{N} P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}),$$

again assuming that $\mathbf{t}_n$ is conditionally independent of the rest of the data points, given $\mathbf{x}_n$. We also need an expression for the distribution $P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w})$, given by the generalized Bernoulli distribution,

$$P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) = \prod_{j=1}^{k} P(\mathcal{C}_j|\mathbf{x}_n, \mathbf{w})^{t_{nj}}.$$

It therefore follows that

$$p(X, T|\mathbf{w}) = p(X) \prod_{n=1}^{N} \prod_{j=1}^{k} P(\mathcal{C}_j|\mathbf{x}_n, \mathbf{w})^{t_{nj}}.$$

The log-likelihood is given by

$$\begin{aligned}
\ell(\mathbf{w}) \;\; &= \;\; \ln p(X) + \sum_{n=1}^{N} \sum_{j=1}^{k} t_{jn} \ln P(\mathcal{C}_j|\mathbf{x}_n, \mathbf{w}_j) \\
&= \;\; \ln p(X) + \sum_{n=1}^{N} \sum_{j=1}^{k} t_{jn} \left[a_j(\mathbf{x}_n) - \ln \left(\sum_{i=1}^{k} \exp a_i(\mathbf{x}_n) \right) \right],
\end{aligned}$$

and we need to find $\mathbf{w}^* = \arg\max_{\mathbf{w}} \ell(\mathbf{w})$, or $\nabla_{\mathbf{w}} \ell(\mathbf{w}^*) = 0$. Since

$$\nabla_{\mathbf{w}_j} \ell(\mathbf{w}) = \sum_{n=1}^{N} \left[t_{nj} - \frac{\exp \mathbf{w}_j^T \boldsymbol{\phi}(\mathbf{x}_n)}{\sum_{i=1}^{k} \exp \mathbf{w}_i^T \boldsymbol{\phi}(\mathbf{x}_n)} \right] \boldsymbol{\phi}(\mathbf{x}_n), \; j = 1, \ldots, k,$$

there are no analytical solutions of $\nabla_w \ell(\mathbf{w}) = 0$, and again one has to resort to numerical optimization techniques.

CHAPTER 5

PARTIALLY OBSERVED DATA AND THE EM ALGORITHM

5.1. Introduction.

In our discussion of classification in Chapter~4 we assumed fully observed data, i.e. each observation came with a class label. In many situations class labels are not available and need to be inferred from the data itself. This problem is often referred to as 'clustering'. The more general problem is to infer *any* missing information from the observed data. Throughout our discussion the fundamental assumption is that, should the missing data somehow become available, the training can be done with ease. A general tool for dealing with partially observed data is the Expectation Maximization (EM) algorithm. The basic idea is very simple. Since we can proceed with the training (estimation of the parameter values) for the fully observed data, we estimate the missing values by calculating an expectation based on the current estimate of the parameters. Once we have values for the missing data, we proceed to maximize a likelihood to get an updated estimate for the parameters. These are then used to re-estimate the missing data, etc.

The simplest example of the EM algorithm is K-means clustering, the starting point of our discussion.

5.2. K-Means Clustering.

We have N observations, each observation belonging to one of K classes, but we are not given any class information. Our task is to somehow assign each observation to an appropriate class, more commonly referred to in this setting as *clusters*. It is important to note that we know, or guess, the value of K, i.e. we assume that we know the number of clusters.

The intuition behind the K-means algorithm is straightforward, and is based on the fact that if we know the cluster labels of wach data value, then it is easy to calculate the means of each cluster. If the means of each cluster is known, it is easy to assign each each data value to a cluster. We therefore proceed in the following manner: First we somehow assign (we'll come back to this point), exactly K cluster means. Then we assign each data value to the cluster mean closest to it[1]. Once all the data values have been assigned to a specific cluster, we use the data values belonging to each cluster to *update* the cluster means. With updated cluster means available, one can re-assign the data values, until convergence. Let us do this formally.

Using the same notation as in Chapter 4, a cluster indicator $\mathbf{t}_n = \begin{bmatrix} t_{n1} & \cdots & t_{nK} \end{bmatrix}^T$ is associated with each D-dimensional observation $\mathbf{x}_n$, where $t_{nk} = 1$ if $\mathbf{x}_n$ belongs to cluster k, otherwise $t_{nk} = 0$. Let us emphasize that in the classification problem of Chapter 4 the $\mathbf{t}_n$ are known. Now they are unknown and our task is to infer the labels from the data. Formally our aproach is to do the cluster assignments by minimizing an appropriate objective function. Suppose that we are, somehow, given K, d-dimensional vectors $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$, which we think of as the cluster means. We then define an objective function as,

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 ,$$

and our goal becomes to find the t_{nk} *and* $\boldsymbol{\mu}_k$ so as to minimize J. This proceeds in two steps, as described above. First we choose initial values for the $\boldsymbol{\mu}_k$. Keeping these fixed we minimize J with respect to t_{nk}, i.e. we assign each data value to cluster. In the second step we keep the t_{nk} fixed and minimize J with respect to the $\boldsymbol{\mu}_k$, i.e. we re-estimate the cluster means. These steps are repeated until convergence.

The t_{nk} are easily determined. Each data value $\mathbf{x}_n$ is assigned to the cluster for which $\left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$ is the smallest. This means that each data point is assigned to its nearest cluster, as defined by its estimated cluster mean $\boldsymbol{\mu}_k$.

Setting the derivative of J with respect to $\boldsymbol{\mu}_k$ equal to zero, we get

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} t_{nk} \mathbf{x}_n}{\sum_{n=1}^{N} t_{nk}}.$$

[1]It is possible to do a more sophisticated assignment.

This means that we set $\boldsymbol{\mu}_k$ equal to the sample mean of all the points assigned to cluster k.

Note:

(1) The description above is based on the minimization of an objective function. One can also follow a probabilistic approach in which case the two steps described above correspond to the two steps of the Expectation Maximization (EM) algorithm. In a probabilistic setting, the clusters labels are *hidden*, i.e. the $\mathbf{t}_n$ are *latent* variables. This means that we are unable to calculate the complete likelihood as in the previous chapter, where it was a simple matter of calculating the the paramters (e.g. the cluster means) by maximizing the likelihood. The main difference here is that, given the K different cluster means, we calculate the *expected* values of the latent variables $\mathbf{t}_n$ (the E step of the EM algorithm). This allows us to calculate an approximate likelihood, which we can use to estimate the parameters by *maximizing* the approximate likelihood (the M step of the EM algorithm). We describe these ideas in detail in the next section when we discuss Gaussian Mixture Models.

(2) It is common practice to rescale the data before the clustering algorithm is applied, by, for example, whitening the data. This ensures that the data is normalized so that the distance measures have some objective meaning. Note however the warning given by Duda and Hart that by doing this one runs the risk of rescaling distances when they are actually indicative of between-cluster separation.

(3) The K-means algorithm uses a *hard* assignment of each data point to a specific class. This does not take into account that at least for some points there may be considerable ambiguity as to which class they belong. In such cases it may be better to use a *soft* assignment where data points are assigned to clusters in such a way as to reflect this uncertainty.

(4) The K-means algorithm is an optimization procedure, and the objective function allows multiple *local* minima. The initialization of the iterative procedure selects a specific local optimum and in general there is no guarantee that this local optimum is also a global optimum.

(5) The K-means algorithm is initialized by assigning initial cluster means. As pointed out above, this choice is important because it determines the final clustering. One possibility is to select K random samples from the data and use these as the initial cluster means. This does not work well in practice since the choice of initial clusters may be heavily biased. In our experience a binary split procedure works better. It starts by selecting two initial cluster means at random. The data points are then assigned to these clusters in the normal way. The scatter of each cluster is then computed and the cluster with the largest scatter is split by choosing two samples from it at random. The algorithm is now iterated a few times, starting with these three initial cluster means. Then the cluster with the largest scatter is split again, and the procedure is repeated until the desired number of clusters is obtained. The algorithm is then iterated until convergence.

5.3. Gaussian Mixture Models.

We have seen that Gaussian pdf's have particularly useful analytical properties. In practice however, one often encounters situations where the data cannot be accurately represented by a single Gaussian pdf which is certainly the case for all multi-modal data sets. One possibility is to represent the data by a mixture of K Gaussians,

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k), \tag{5.1}$$

where where $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$ is a Gaussian distribution with means and covariance, μ_k and Σ_k, respectively, and $\pi_k \geq 0$ with $\sum_{k=1}^{K} \pi_k = 1$. The parameters θ that need to be estimated are the class means and covariances, $\boldsymbol{\mu}_k$ and Σ_k respectively, as well as the 'class probabilities' or mixture coefficients, π_k. These should be estimated from the data, $\mathbf{x}_n$, $n = 1, \ldots, N$, but there is a complication—we don't know how the data values $\mathbf{x}_n$ contribute towards the different mixture components. In fact, this is exactly the same as the situation encountered with the K-means algorithm of the previous section. In this case the mixture components can be viewed as the different clusters of the K-means algorithm.

If we know the mixture components to which each each data point belongs, the learning problem is easy, we simply calculate sample estimates of the means and covariances from the data belonging to each mixture component. The fraction of the data points assigned to each mixture component estimates the mixture coefficients. Since we don't know it, we proceed in a more principled probabilistic manner.

The familiar latent random variable $\mathbf{z} = \begin{bmatrix} z_1 & \cdots & z_K \end{bmatrix}^T$ where $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$, assigns $\mathbf{z}$ to one of K mixture components, according to which element is non-zero. Making use of the sum rule, we find that

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) \\ &= \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z}) P(\mathbf{z}) \\ &= \sum_{k=1}^{K} p(\mathbf{x}|z_k = 1) P(z_k = 1). \end{aligned}$$

This is simply a general mixture model, where $P(z_k = 1)$ is the prior mixture component probability, and $p(\mathbf{x}|z_k = 1)$ the mixture density function. For the specific case of a *Gaussian* mixture model, the density functions become Gaussians. Comparison with (5.1) shows that

$$P(z_k = 1) = \pi_k,$$

and

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k).$$

Next we want to assign an observation to the different mixture components, i.e. we want to calculate the mixture component probability $P(z_k = 1|\mathbf{x})$. Defining $\gamma(z_k) = P(z_k = 1|\mathbf{x})$, it follows from Bayes' theorem that

$$\begin{aligned} \gamma(z_k) &= \frac{P(z_k = 1) p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} P(z_j = 1) p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}. \end{aligned} \tag{5.2}$$

Note that this is a *soft* assignment where $\mathbf{x}$ is not assigned to a specific class, but its probability of belonging to a class is distributed over all the classes. Here π_k as the

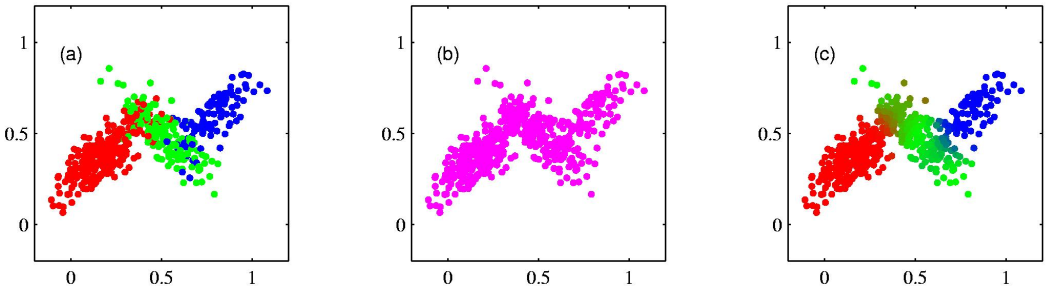


FIGURE 5.3.1. Generating samples from a mixture of 3 Gaussians.

prior probability of the k-th mixture component, $z_k = 1$ i.e. $P(z_k = 1) = \pi_k$, *before* any observations, and $\gamma(z_k)$ is the posterior probability once we have observed $\mathbf{x}$. Alternatively, $\gamma(z_k)$ can be viewed as the *responsibility* that the k-th component takes for 'explaining' the observation $\mathbf{x}$. Said in a different way, $\gamma(z_k)$ is the probability of $\mathbf{x}$ belonging to the k-th mixture component.

In summary, provided we know the mixture parameters, π_k, μ_k and Σ_k the responsibilities allow one to make soft assignments of data to the different components. In the next section we'll see, given the responsibilities, how to estimate the mixture parameters.

EXAMPLE 46. *Ancestral sampling* to generate samples from the joint probability using three mixture components gives Figure 5.3.1. In (a) the samples as drawn from the different components are shown. (This corresponds to fully observed data where each data point is assigned a 'cluster'.) In (b) the data is shown without its mixture components; this is the way it is presented to us. In (c) the responsibility that each component takes for every data point is indicated with different color shades.

Suppose we are given N, d-dimensional data points $\mathbf{x}_n$ collected into an $d \times N$ matrix X. The likelihood is defined as the joint distribution $p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$,

$$\begin{aligned} p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) &= p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \end{aligned}$$

where we have used (5.1) in the last step. The log likelihood is therefore given by

$$\ln p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}.$$

If we do not want to provide prior probabilities $p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ for the parameters (or maximizing over this more complex expression), we rather consider,

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}. \tag{5.3}$$

Remark: It is important to note that for mixture models the likelihood can actually become infinite, i.e. it has a singularity, a situation that should be avoided. To see how this can happen, consider a mixture model where all the components have covariances of the form $\sigma_k^2 I$. Suppose one of the mixture components has its mean, say $\boldsymbol{\mu}_j$, exactly equal to one of the data points, i.e. $\boldsymbol{\mu}_j = \mathbf{x}_n$. The contribution of this data point to the likelihood is of the form

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi\sigma_j^2}}.$$

This goes to infinity as $\sigma_j \to 0$, i.e. this particular component collapses onto a specific data point, sending the likelihood to infinity—we are in the process of maximizing the likelihood after all. This is always a possibility and one has to take steps in order to avoid this situation. (Maybe it was not such a good idea after all to get rid of the priors above.) Also note that this danger does not exist for single Gaussians.

5.4. The Expectation Maximization (EM) Algorithm for Gaussian Mixture Models.

We want to estimate the parameters by maximizing the log-likelihood (5.3). Setting the partial derivative of the log-likelihood (5.3) with respect to $\boldsymbol{\mu}_k$ equal to zero gives

$$\begin{aligned} 0 &= -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} \Sigma_k \left(\mathbf{x}_n - \boldsymbol{\mu}_k\right) \\ &= -\sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k \left(\mathbf{x}_n - \boldsymbol{\mu}_k\right), \end{aligned}$$

or

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n, \tag{5.1}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

Note that all the data points contribute towards the mean of the k-th mixture component, weighted with their responsibilities to the k-th component. The expression for the covariance follows similar lines,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T.$$

Again all the data points contribute towards the k-th mixture component, weighted with their responsibilities.

In order to find the mixture coefficients we need to impose the constraint $\sum_{k=1}^{K} \pi_k = 1$. This is done by introducing a Lagrange multiplier into the log-likelihood,

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right).$$

Maximizing with respect to π_k, yields,

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda.$$

Multiply both sides with π_k and summing over k gives $\lambda = -N$. Another multiplication of both sides, this time withwith π_k, gives

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) - \pi_k N,$$

or

$$\pi_k = \frac{N_k}{N},$$

which is the average responsibility for the k-th mixture component.

Note these equations are not solvable in closed form since the responsibility depends in a complicated way on the parameters. Their form however, does suggest the following algorithm:

Expectation Maximization Algorithm for GMMs

1. Initialize the the different coefficients, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and Σ. One possibility is to use the K-means algorithm.

2. **E step**. Evaluate the responsibilities using the current estimates,

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}.$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \\
\Sigma_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

4. Evaluate the log likelihood

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\},$$

check for convergence of either the parameters, or the log likelihood.

5. Keep iterating from step 2 until convergence.

CHAPTER 6

KALMAN FILTERS

6.1. Introduction.

Up to now we have only used i.i.d. data. The next major step is to drop this assumption. The first example that we'll encounter where the observations are no longer independent is the Kalman filter.

6.2. Kalman Filter Equations.

Suppose we know that the dynamics of the system under consideration is given by

$$\mathbf{x}_{n+1} = A_n \mathbf{x}_n + \boldsymbol{\omega}_n \tag{6.1}$$

where $\mathbf{x}_n$ is the feature vector describing the system at time n, A_n is a matrix describing the transition from $\mathbf{x}_n$to $\mathbf{x}_{n+1}$. This means that the process is linear which is of course a strong assumption. It is possible to generalize; in fact, the derivation of the Kalman filter given in this Chapter is eminently suitable for generalization. The final term, $\boldsymbol{\omega}_n$, is a noise term. It may represent inadequate knowledge of the system, and/or noise introduced during the evolution of the system. We assume that the noise is Gaussian with zero mean,

$$p(\boldsymbol{\omega}_n) \;\; = \;\; \mathcal{N}(\boldsymbol{\omega}_n | \mathbf{0}, Q_n)$$

and that the $\boldsymbol{\omega}_n$ are independent for different n.

Apart from the dynamics, we also observe the system and the observation equations are given by

$$\mathbf{z}_{n+1} = H_n \mathbf{x}_{n+1} + \boldsymbol{\nu}_{n+1}. \tag{6.2}$$

The system is only indirectly observed and the observations $\mathbf{z}_{n+1}$ are linked to the features $\mathbf{x}_n$ by the matrix H_n. Again a linear relationship is assumed, something that can and should be generalized. The observation noise is given by $\boldsymbol{\nu}_{n+1}$and is again assumed to be zero mean and Gaussian,

$$\boldsymbol{\nu}_{n+1} = \mathcal{N}(\boldsymbol{\nu}_{n+1}|\mathbf{0}, R_n).$$

We also assume that the $\boldsymbol{\nu}_{n+1}$ are independent for different n. Since we assume linear processes and Gaussian noise, it should be clear that the state variables are also Gaussian, assuming Gaussian initial values. This makes life much easier, in fact, it allows an analytical solution of the Kalman equations.

Our task is to obtain the best possible estimate of the system $\mathbf{x}_n$, given the observations $Z_n = \{\mathbf{z}_j\}$, $j = 1, \dots, n$, as well as our knowledge of the noise terms, i.e. given the covariances Q_n and R_n. For this purpose we rewrite the equations in probabilistic form. The dynamic equation becomes,

$$p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n) = p(\mathbf{x}_{n+1}|\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\mathbf{x}_n, Q_n) \tag{6.3}$$

and the observation equation becomes

$$p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n) = p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}) \tag{6.4}$$

$$= \mathcal{N}(\mathbf{z}_{n+1}|H_{n+1}\mathbf{x}_{n+1}, R_n). \tag{6.5}$$

It should be pointed out that it is assumed that $\mathbf{x}_{n+1}$ is conditionally independent of the observations Z_n, given that $\mathbf{z}_n$ is known. Similarly, $\mathbf{z}_{n+1}$ is conditionally independent of Z_n, given that $\mathbf{x}_{n+1}$ is known.

Assuming that we know $p(\mathbf{x}_n|Z_n)$, our goal is to calculate $p(\mathbf{x}_{n+1}|Z_{n+1})$, given the latest observation $\mathbf{z}_{n+1}$. Since we know that $\mathbf{x}_n$ is a Gaussian variable, it is useful to write

$$p(\mathbf{x}_n|Z_n) = \mathcal{N}(\mathbf{x}_n|\widehat{\mathbf{x}}_n, P_n).$$

Note:

- This formulation leads naturally to a recursive implementation. Given $p(\mathbf{x}_0|Z_0)$ initially, we calculate $p(\mathbf{x}_n|Z_n)$ every time we receive a new observation $\mathbf{z}_n$.

The marginal distribution is given by

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_n) &= \int p(\mathbf{x}_{n+1}, \mathbf{x}_n|Z_n)d\mathbf{x}_n \\ &= \int p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n)p(\mathbf{x}_n|Z_n)d\mathbf{x}_n \\ &= \int p(\mathbf{x}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|Z_n)d\mathbf{x}_n. \end{aligned}$$

Since we know both Gaussian distributions under the integration, it follows immediately that

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\widehat{\mathbf{x}}_n, Q_n + A_nP_nA_n^T). \tag{6.6}$$

Suppose we need the best estimate of the state variables $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$. Usually one has to be careful in specifying exactly what is meant with 'best' estimate. Since we are dealing with Gaussian densities however, 'best' estimate in just about every meaningful sense of the word amounts to the same quantity namely the mean. Rewriting (6.6) as

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|\mathbf{x}_{n+1}^-, P_{n+1}^-)$$

with

$$\mathbf{x}_{n+1}^- = A_n\widehat{\mathbf{x}}_n \tag{6.7}$$

and

$$P_{n+1}^- = Q_n + A_nP_nA_n^T$$

the best estimate of $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$ is given by (6.7).

The marginal distribution of the observation $\mathbf{z}_{n+1}$ is given by

$$\begin{aligned} p(\mathbf{z}_{n+1}|Z_n) &= \int p(\mathbf{z}_{n+1}, \mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n)p(\mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1})p(\mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1}. \end{aligned}$$

Since both terms in the integrand are Gaussian, it follows in a straightforward way that the observation marginal is given by

$$p(\mathbf{z}_{n+1}|Z_n) = \mathcal{N}(\mathbf{z}_{n+1}|\mathbf{z}^-_{n+1}, S_{n+1})$$

where

$$\mathbf{z}^-_{n+1} = H_{n+1}\mathbf{x}^-_{n+1} \tag{6.8}$$

and

$$S_{n+1} = R_{n+1} + H_{n+1}P^-_{n+1}H^T_{n+1}. \tag{6.9}$$

Note:

- Since $\mathbf{x}^-_{n+1}$ is the best estimate of $\mathbf{x}_{n+1}$, $\mathbf{z}^-_{n+1}$ given by (6.8) is our best estimate of the next measurement, prior to actually receiving the next measurement.

Since we expect to next measure $\mathbf{z}^-_{n+1}$, the extent to which the actual measurement deviates from this expectation, is a measure of the how much we learn from the measurement at $n+1$. Using Bayes' theorem we now calculate

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_{n+1}) &= p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}, Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n)p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1})p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \end{aligned}$$

where we made use of the various conditionally independencies. It again follows in s straightforward manner that

$$p(\mathbf{x}_{n+1}|Z_{n+1}) = \mathcal{N}(\mathbf{x}_{n+1}|\widehat{\mathbf{x}}_{n+1}, P_{n+1})$$

where

$$\widehat{\mathbf{x}}_{n+1} = P_{n+1}\left(P^-_{n+1}\right)^{-1}\mathbf{x}^-_{n+1} + P_{n+1}H^T_{n+1}R^{-1}_{n+1}\mathbf{z}_{n+1} \tag{6.10}$$

and

$$P_{n+1} = \left[\left(P^-_{n+1}\right)^{-1} + H^T_{n+1}R_{n+1}H_{n+1}\right]^{-1}. \tag{6.11}$$

We are almost done. These equations give us exactly what we want. The best estimate of $\mathbf{x}_{n+1}$, given the observation sequence Z_{n+1}, is given by $\widehat{\mathbf{x}}_{n+1}$with associated covariance P_{n+1}. All that remains to be done is simplify these equation, rewriting them in a more convenient form. For this purpose we define the so-called Kalman gain

$$W_{n+1} = P^-_{n+1} H^T_{n+1} S^{-1}_{n+1}. \tag{6.12}$$

It takes a bit of manipulation but it is not hard to show that this allows us to write

$$P_{n+1} = P^-_{n+1} - W_{n+1} S_{n+1} W^T_{n+1} \tag{6.13}$$

and

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}^-_{n+1} + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}^-_{n+1}\right). \tag{6.14}$$

In summary, given $\widehat{\mathbf{x}}_n$ and P_n:

1. $\mathbf{x}^-_{n+1} = A_n \widehat{\mathbf{x}}_n$, and $P^-_{n+1} = Q_n + A_n P_n A^T_n$ (using the dynamic information)
2. $\mathbf{z}^-_{n+1} = H_{n+1}\mathbf{x}^-_{n+1}$ and $S_{n+1} = R_{n+1} + H_{n+1} P^-_{n+1} H^T_{n+1}$ (using the measurement equation)
3. $W_{n+1} = P^-_{n+1} H^T_{n+1} S^{-1}_{n+1}$ (Kalman gain)
4. Given the new measurement $\mathbf{z}_{n+1}$, calculate

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}^-_{n+1} + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}^-_{n+1}\right) \text{ and } P_{n+1} = P^-_{n+1} - W_{n+1} S_{n+1} W^T_{n+1}.$$

6.2.1. Simplifying Using the Kalman Gain. We give the details of deriving (6.13) and (6.14) from (6.10) and (6.11) respectively.

It is the easiest to verify that

$$\begin{aligned} P_{n+1} &= \left[\left(P^-_{n+1}\right)^{-1} + H^T_{n+1} R_{n+1} H_{n+1}\right]^{-1} \\ &= P^-_{n+1} - W_{n+1} S_{n+1} W^T_{n+1} \end{aligned}$$

by showing that

$$\left[\left(P^-_{n+1}\right)^{-1} + H^T_{n+1} R_{n+1} H_{n+1}\right]\left[P^-_{n+1} - W_{n+1} S_{n+1} W^T_{n+1}\right] = I.$$

Multiplying out we find that

$$\begin{aligned}
\left[\left(P_{n+1}^-\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]\left[P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right] &= \\
I - H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T + H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- - \\
H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T &= \\
I - H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T + H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- - \\
H_{n+1}^T R_{n+1}^{-1}\left(S_{n+1} - R_{n+1}\right) S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T &= I,
\end{aligned}$$

where we have used (6.9) and the fact that covariance matrices are symmetric.

Starting from (6.10) we find that

$$\begin{aligned}
\widehat{\mathbf{x}}_{n+1} &= P_{n+1}\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- + P_{n+1}H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\
&= \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\
&= \mathbf{x}_{n+1}^- - P_{n+1}^- H_{n+1}^T S_{n+1}^{-1} S_{n+1} S_{n+1}^{-T} H_{n+1} P_{n+1}^-\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- \\
&\quad + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\
&= \mathbf{x}_{n+1}^- - P_{n+1}^- H_{n+1}^T S_{n+1}^{-1}\mathbf{z}_{n+1}^- + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\
&= \mathbf{x}_{n+1}^- - W_{n+1}\mathbf{z}_{n+1}^- + W_{n+1}S_{n+1}\left(I - W_{n+1}^T H_{n+1}^T\right)R_{n+1}^{-1}\mathbf{z}_{n+1}.
\end{aligned}$$

If now use (6.9) it follows that

$$\begin{aligned}
S_{n+1}\left(I - W_{n+1}^T H_{n+1}^T\right)R_{n+1}^{-1} &= I + H_{n+1}P_{n+1}^- H_{n+1}^T R_{n+1}^{-1} - \\
&\quad S_{n+1}S_{n+1}^{-1}H_{n+1}P_{n+1}^- H_{n+1}^T R_{n+1}^{-1} \\
&= I,
\end{aligned}$$

and we are done.

CHAPTER 7

DYNAMIC PROGRAMMING.

In practice we are often faced with a situation where it is necessary to compare different *signals* of different length. For instance, if one is interested in identifying a word or a phrase in a speech recognition application, one realizes that different speakers utter the same phrase in different ways, and with different durations. Comparing these signals is exactly the same problem we face in a signature verification problem. Again we have two similar signals, with differences in details and of different durations to compare. The algorithm we are about to describe was therefore extensive used in speech processing applications until it was largely replaced by even more powerful Hidden Markov Models. It should become clear that there is a straightforward generalization to vector-valued signals; for the moment we keep things as simple as possible and concentrate on single signals. The basic idea is to first define a suitable 'cost function' that gives an indication of the difference (total cost) between the signals. One then proceeds to find a map that maps one signal onto the other in such a way that the cost function is minimized. In this sense one therefore finds the best possible match between the two signals, taking into account that they are usually very different. One can then relate the total 'cost' (difference between the two signals) to a confidence value—the lower the cost, the higher the confidence that the two signals are actually different renderings of the same entity.

It might be interesting to note that the algorithms is much more general than just for comparing signals. It is a powerful algorithm for calculating maps minimizing an appropriate cost function. It is particularly useful in situations where one has only one example against which to compare, i.e. when it is not possible to build a model from a number of examples (as an additional advantage we can use it as a pro-type for the Viterbi algorithms that we will encounter a little later). For example, the line-break, paragraph-blocking algorithm used by TEX is based on this algorithm.

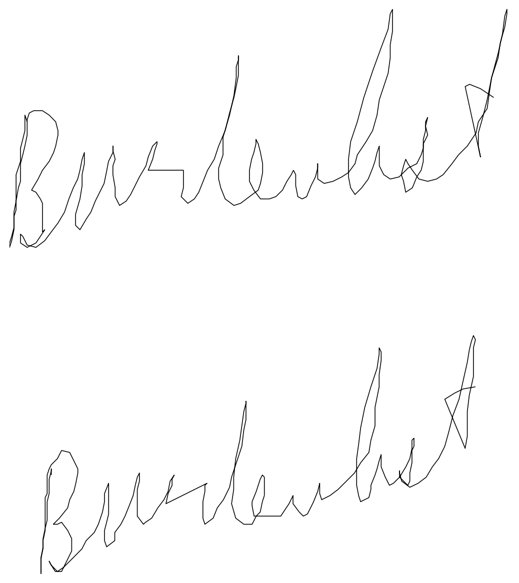


FIGURE 7.0.1. Two different signatures by the same signatory for comparison.

The naïve approach is to estimate the number number of words that will fit into a line, given the page width, and then calculate the inter-word distances so that the line is of exactly the specified length. The problem with this algorithm is that the result does not look good. Sometimes it is better to move a word to the next line for improved overall appearance, etc. Thus the best appearance is achieved by analyzing a whole paragraph and not simply each line separately. The way Knuth [**?**] does this is to define a suitable cost function that is constructed with overall appearance in mind. A mapping is then calculated that fills a whole paragraph in block form.

Let us have a closer look at the problem. In Figure 7.0.1 we show two different signatures by the same person with their y coordinates shown in Figure 7.0.2. It should be clear that the natural way of comparing these signals is to find matching points on the two signatures, such as the peaks indicated in Figure 7.0.2. This clearly requires a nonlinear stretching or 'warping' of the signatures to best fit each other. Mathematically this amounts to a *re-parametrization* of the two signatures.

Given the two discrete signals, $\mathbf{y}_1 := \{y1(t), \quad t = 1, \ldots, L_1\}$ and $\mathbf{y}_2 := \{y_2(s), \quad s = 1, \ldots, L_2\}$, the problem is to find re-parametrizations $p(w)$ and $q(w)$ such that one can identify $y1(p(w))$ with $y_2(q(w)), \quad w = 1, \ldots, L$ in such a way that the difference

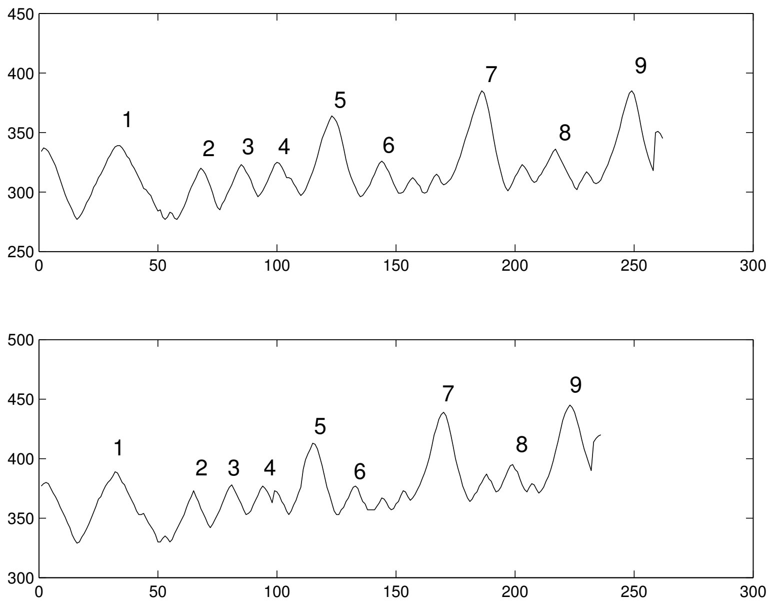


FIGURE 7.0.2. The y coordinates of the two signatures.

between the two resulting function is minimized. In order for $p(w)$ and $q(w)$ to be proper re-normalizations they need to be *non-decreasing* functions of w. Since the value of L depends on y_1 and y_2 it is determined as part of the algorithm.

Let us now become very specific and assume that the two signals we want to compare are given by,

$$\begin{aligned}\mathbf{y}_1 &= [1\ \ 0\ \ 1\ \ 2\ \ 1]\\ \mathbf{y}_2 &= [1\ \ 0\ \ 2\ \ 1].\end{aligned}$$

Since we want to compare each value of $\mathbf{y}_1$ with each value of $\mathbf{y}_2$ it is convenient to draw a grid as shown in Figure 7.0.3. If we now draw a curve consisting of straight lines connecting the lower left-hand corner with the upper right-hand corner (the reader may wish to have a peak at Figure 7.0.7) below, any such curve defines a mapping between the two signals. It is important to note however that the monotonicity constraint on $p(w)$ and $q(w)$ implies that grid-point (i, j) can only be reached from either $(i-1, j)$, $(i, j-1)$ or $(i-1, j-1)$, as indicated (if there is a tie, we give preference to the diagonal). It is this curve that we are after, constructed in such a way that the two signals are aligned in the best possible way (in a sense that has to be made precise). Also note that our assumption that the curve begins and ends at the two corners implies that we match the start– and endpoints of the two signals. In some applications it is useful to relax this constraint—it is absolutely straightforward to relax the endpoint matching constraint, as will become clear.

The brute-force way of solving the problem is to investigate all possible paths connecting the lower left-hand corner with the upper right-hand corner. This is a use number and the reader may find it a fun exercise to derive the following formula for the total number of possible paths

$$\mathcal{N} = \sum_{s=0}^{\min(L_1,L_2)} \frac{(L_1 + L_2 - s)!}{(L_1 - s)!(L_2 - s)!s!},$$

constrained only by our monotonicity requirement. This is an enormous number, dominated by the first term which counts only the number of paths not containing a diagonal,

$$(L_1 + L_2)!/L_1!L_2!.$$

Even this is too large to investigate exhaustively by computer for all but the smallest number of samples. We clearly need to do better.

All the most efficient algorithms are based on a very simple observation (which the reader can prove for herself): Each sub-path of an optimal path, is also optimal. This means that the global optimal path is pieced together from local optimal paths—exactly the defining strategy of Dynamic Programming (DP). The key is to define a local distance measure (local cost function) that measures the difference between the two functions. An obvious choice is

$$C_{i,j} = d(y1(i), y2(j)) := |y_1(i) - y_2(j)|, \quad i = 1, \dots, L_1; \quad j = 1, \dots, L_2. \tag{7.1}$$

For our example, this is written as

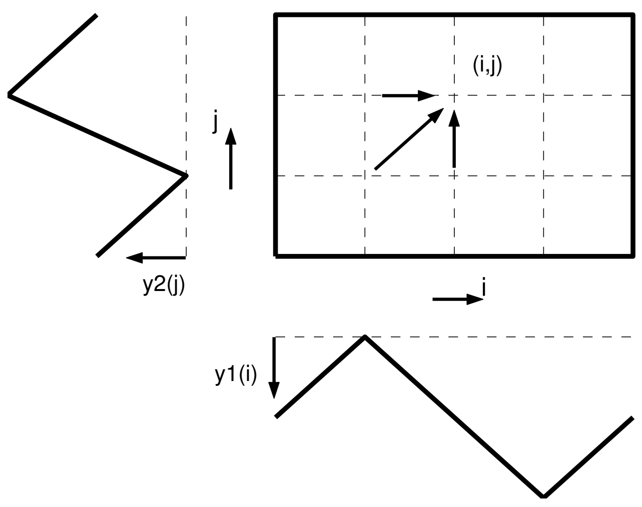


FIGURE 7.0.3. The two signals to be compared.

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

where one should note that the ordering corresponds to the ordering of the grid of Figure 7.0.4. Also note that in Figure 7.0.4 that it is convenient for us to number the grid points consecutively from 1 to 20. For example, grid point 9 has coordinates $(4, 2)$ and its local distance value is $C_{4,2} = 2$. Based on this local distance measure, the total cost function for the two signals $y_1(t)$ becomes

$$C = \sum_{w=1}^{L} |y_1(p(w)) - y_2(q(w))| . \tag{7.2}$$

Once the the local costs have been calculated, the rest of the algorithm proceeds without any reference to the original signals—it allows us to calculate the optimal path from the lower left-hand corner to every other point on the grid. This may sound wasteful but with the slight modification explained below, it is still the most efficient algorithm known. The result is shown in Figure 7.0.5. The Figure shows the optimal path to each grid point with the total cost of each path. For example, the cost of reaching point 19 with coordinates $(4, 4)$, is 2. In order find its optimal path, we start at point and then backtrack until we reach point 1. Note that all we need is to know from where a specific point is reached, i.e. point 19 is reached from point 13, is reached from point 7, etc. Accordingly, we keep track of all the different paths, by defining an array $I(i), i = 1, \ldots, L_1L_2$ where $I(k)$ denotes the point from which point k is reached. For our example, Figure 7.0.5, we find that $I(20) = 14, I(19) = 13$, etc. Thus the optimal path is obtained from $I(20) = 14,\ I(14) = 8,\ I(8) = 7,\ I(7) = 1$.

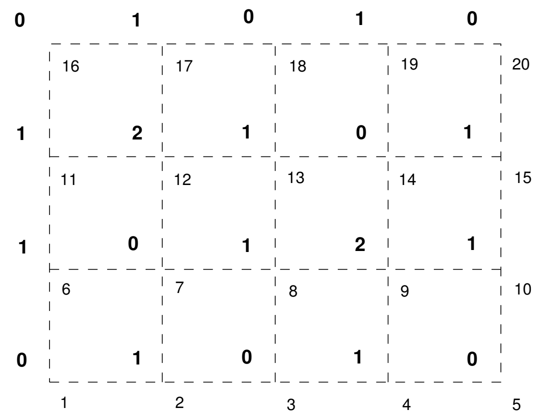


FIGURE 7.0.4. The local cost grid.

Let us proceed with the description of the algorithm. Since the local cost for point 1 is zero (see Figure 7.0.4), the total cost to reach grid point 1 is $TC(1) = C_11 = 0$. Noting that point 2 can only be reached from point 1, there is no decision to make and the total cost of reaching point 2 is: $TC(2) = TC(1) + C_{21} = 1$. We also record that it is reached via the optimal (and only) path from point 1, i.e. we set $I(2) = 1$. For points 3, 4, 5 and 6 we do the same, $TC(3) = TC(2) + C_{31} = 1, \quad I(3) = 2, \quad TC(4 = TC(3 + C_{41} = 2, \quad I(4) = 3, \quad TC(5 = TC(4 + C_{51} = 2, \quad I(5) = 4, \quad TC(6 = TC(1 + C_{12} = 1, \quad I(6) = 1$. However, point 7 may be reached from either points 1, 2 or 6. We already know the optimal paths to points 1, 2 and 6. Since the cost of the path to point 1, is less or equal to the cost to points 2 or 6, we reach point 7 via point 1 and set $TC(7) = TC(1) + C_{22} = 0, \quad I(7) = 1$. Now we do the same for the rest of the points on the grid:

$$\begin{aligned}
TC(8) &= TC(7) + C_{32} = 1, \quad I(8) = 7 \\
TC(9) &= TC(3) + C_{42} = 3, \quad I(9) = 3 \\
TC(10) &= TC(4) + C_{52} = 3, \quad I(10) = 4 \\
TC(11) &= TC(6) + C_{13} = 2, \quad I(11) = 6 \\
TC(12) &= TC(6) + C_{23} = 2, \quad I(12) = 6 \\
TC(13) &= TC(7) + C_{33} = 1, \quad I(13) = 7 \\
TC(14) &= TC(8) + C_{43} = 1, \quad I(14) = 8 \\
TC(15) &= TC(14) + C_{53} = 2, \quad I(15) = 14 \\
TC(16) &= TC(11) + C_{14} = 2, \quad I(16) = 11 \\
TC(17) &= TC(11) + C_{24} = 3, \quad I(17) = 11 \\
TC(18) &= TC(13) + C_{34} = 1, \quad I(18) = 13 \\
TC(19) &= TC(13) + C_{44} = 2, \quad I(19) = 13 \\
TC(20) &= TC(14) + C_{54} = 1, \quad I(20) = 14.
\end{aligned}$$

With the help of I it is now a simple matter of finding the optimal path as pointed out above, and shown in Figure 7.0.6, with its total cost $TC(20) = 1$.

The computational cost of this algorithm amounts to three tests at each grid point, i.e. it is of $O(L_1 L_2)$. This can be further reduced by realizing that the optimal path should not stray too far from the diagonal, at least not for similar signals. One can therefore restrict the search to a band around the diagonal, further reducing the computational cost. If we restrict the search to a band of width d, the computational cost is of $O(L_1)$ with a possibly large constant, depending on d (assuming $L_1 \geq L_2$).

Figure 7.0.7(a) shows the matching of the y coordinates of different signatures belonging to the same person. It is interesting to note how close the warping function remains to the diagonal, an indication that there is a good correspondence between the two functions. If one now plots y_1 and y_2 against their re-parametrizes indexes, Figure 7.0.7(b) shows how the different peaks are now perfectly aligned. Incidentally, the total cost in this case is 374.1.

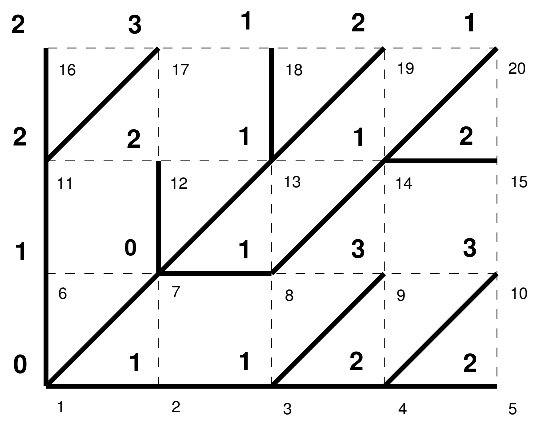


FIGURE 7.0.5. The optimal paths to all the grid points.

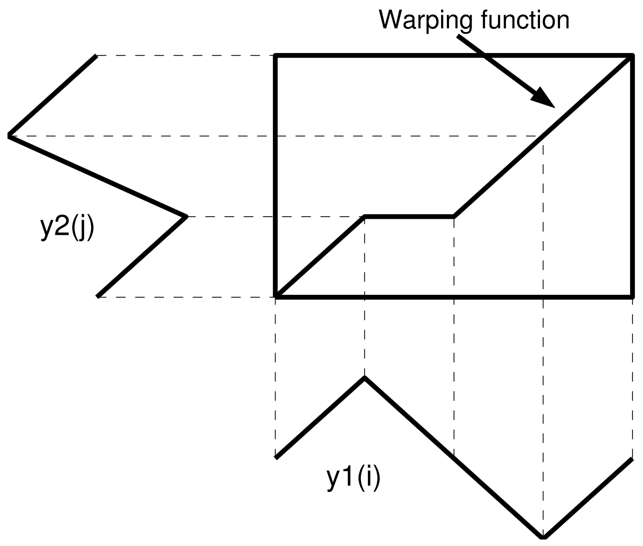


FIGURE 7.0.6. The optimal path.

Let us now do the same thing for two completely different signatures as shown in Figure 7.0.8. One may think of the second signature as a *casual* forgery since the forger clearly had no idea what the original looked like. The point is that the algorithm still finds the best possible match, which in this case is not good at all.

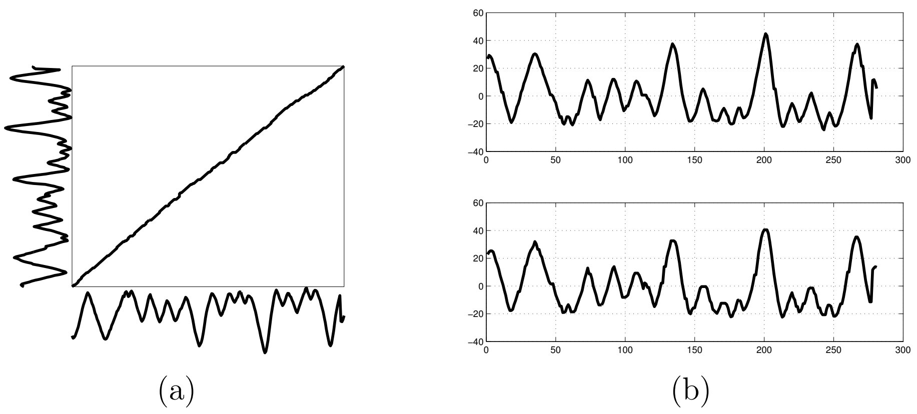


FIGURE 7.0.7. Matching two similar signatures.(a) The warping path. (b) The two signals aligned.

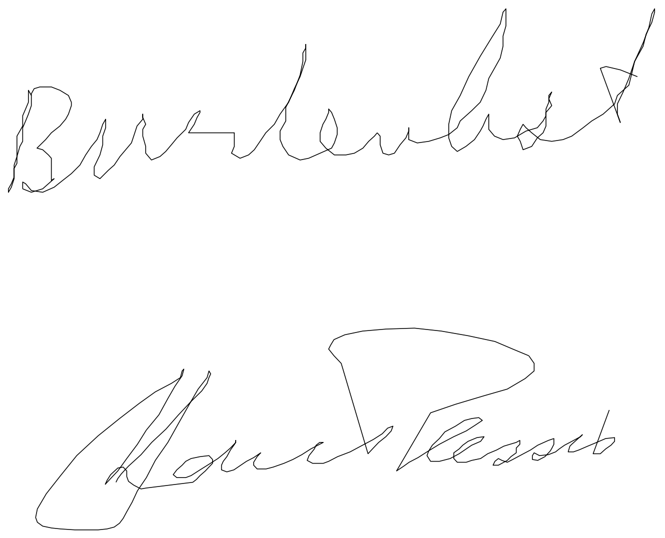


FIGURE 7.0.8. Two unrelated signatures.

The y coordinates, their warping function as well as their best possible alignment are shown in Figure 7.0.9. Although the algorithm still finds the best match, the warping function deviates considerably from the diagonal. The big difference between the two signatures is reflected by the large value of its total cost function, 2 885.

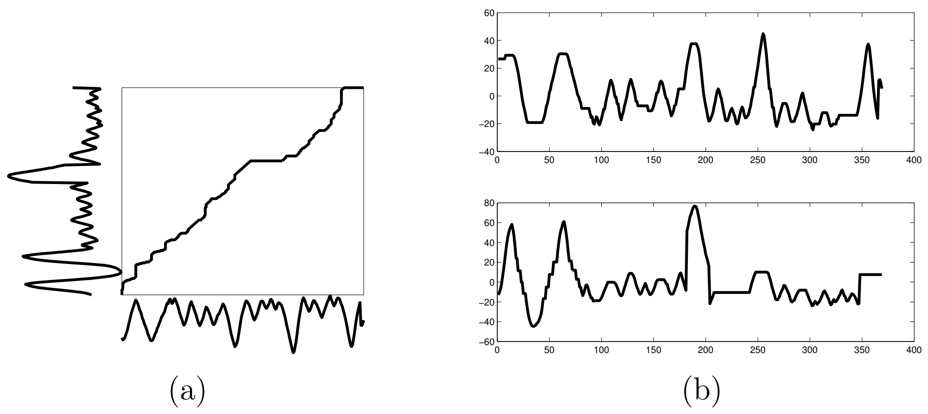


FIGURE 7.0.9. Matching two unrelated signatures.(a) The warping path. (b) The two signals aligned.

CHAPTER 8

HIDDEN MARKOV MODELS

8.1. Introduction.

Hidden Markov models (HMMs) and Kalman filters are related in the sense that both are models where the usually unobserved internal state governs the external observations that we make. Furthermore the internal state at the next time step is only determined by the current internal state (the so-called 1st-order Markov property). However, they are also very different. Firstly, while the Kalman filter has a continuous state space (i.e. its internal state is described by a set of real-valued numbers), the HMM has a discrete state space (i.e. its state can be depicted by a limited set of integers). The Kalman filter changes its state in a linear manner (i.e. a matrix multiplication will take you from its current state to the next). The HMM changes its state in a probabilistic manner, a transition matrix specifies which states can follow on the current one and with what probability this will happen. This makes its behaviour to be very non-linear. Although these links are intriguing, we will develop the our understanding of the HMM by taking a further step back to pick up the thread from the Naive Bayes approach we previously encountered.

8.2. Basic concepts and notation

Hidden Markov models (HMMs) and Kalman filters are related in the sense that both are models where the usually unobserved internal state governs the external observations that we make. Furthermore the internal state at the next time step is only determined by the current internal state (the so-called 1st-order Markov property). However, they are also very different. Firstly, while the Kalman filter has a continuous state space (i.e. its internal state is described by a set of real-valued numbers), the HMM has a discrete state space (i.e. its state can be depicted by a limited set of integers). The Kalman filter changes its state in a linear manner (i.e.

a matrix multiplication will take it from its current state to the next). The HMM changes its state in a probabilistic manner, a transition matrix specifies which states can follow on the current one and with what probability this will happen. This makes its behavior very non-linear. Although these links are intriguing, we will develop the our understanding of the HMM by taking a further step back to pick up the thread from the Naive Bayes approach we previously encountered.

Before we continue, it might be useful to remind ourselves exactly what we hope to achieve. Assume that somehow we are given a number of models M. Late on we describe how to obtain or learn each models from data known to belong to that specific model. Thus we follow a generative approach. If we are given a sequence of T observations $\mathbf{x}_1^T$ the problem is to assign it to one of the models. As before we therefore interested in,

$$P(M|\mathbf{x}_1^T) = \frac{p(\mathbf{x}_1^T|M)P(M)}{\sum_M p(\mathbf{x}_1^T|M)P(M)}, \tag{8.1}$$

In order to calculate the posterior, we need the likelihood $p(\mathbf{x}_1^T|M)$ as well as the prior model probabilities $P(M)$. Superficially there does not appear to be any difference from the generative classification problems we studied before. However, there is one crucial difference. Before we assumed the observations $\mathbf{x}_1^T$ to be *independent.* In the naive Bayes approach for example, we could write the likelihood as $p(\mathbf{x}_1^T|M) = \prod_{t=1}^T p(\mathbf{x}_t|M)$[1]. Statistical independence is a very strong assumption, destroying any hope that this model will be able to capture some time-dependent behavior between the various $\mathbf{x}_t$'s, in fact jumbling them into any arbitrary time order will not affect the resulting pdf value at all.

The crucial generalization that is introduced in this chapter, is that of statistical *dependence.*

8.2.1. Emitting states. We now want to make less drastic assumptions that will allow us to model time-dependent behavior whilst at the same time keeping computations tractable. HMMs do this by introducing the concept that the model is in 1 of N states at any given time. However, we typically do *not* know in which

[1]Here we use the '$|M$' to indicate that the calculation is being done using a specific set of model parameters. When it is clear from the context that we are referring to a specific model we may omit this for the sake of simplicity.

state we are at such a given time - hence the name *hidden* Markov model. Each state $s = i,\ i = 1, \ldots, N$ on its own behaves very much like the naive Bayes / iid model we previously encountered. It has a pdf describing the feature vectors associated with it, ie $p(\mathbf{x}|s)$. In this way one can think of the naive Bayes model as a 1 state HMM, or alternatively of an HMM as a series of naive Bayes models with probabilistic jumps between them. We will need to refer generically to the state that is active at time t, we will use the notation s_t for this.

8.2.2. Transitions. These states are coupled to each other with transition probabilities $a_{i,j} = P(s_{t+1} = j|s_t = i)$. In other words, it indicates the probability that we will transit to state j at the next time step *if* we knew that we are in state i at the current time step (which we of course unfortunately do not know). Of course, once one has made a transition to a new state, a new pdf apply, thereby giving the HMM the ability to model temporal patterns. This basically allows us to change the model behavior as a function of time. These probabilities are collected in a transition matrix A, its row indexes indicating the source state and the columns indicating the destination state of the transition. Since all the probabilities departing from a given state must sum to one, each row in this matrix will sum to one.

8.2.3. Non-emitting/null states. In addition to the above we also need to be able to specify the states in which our model can start (the initial states) and those in which it can terminate. We do this by adding two special states without any densities, namely state 0 and state $N + 1$[2]. These states are called null or non-emitting states to distinguish them from the normal/emitting states we encountered above [3]. State 0 will have no links entering it, and state $N + 1$ will have no links leaving from it. We require that the process always start in state 0 and terminate in state $N + 1$. As can be seen from figure 8.2.3 this allows for arbitrary initial and terminal states. We extend the s_t notation to allow for these states by also including an s_0 and s_{T+1}. Note, however, that the transition to the final state does not occupy

[2]It is also possible and useful to use such null states in other places in the model. Since it complicates algorithms we refrain from using this here.

[3]Our notation deviates here from the common convention to use a vector π to indicate initial states, and possibly another vector $\mathbf{F}$ to indicate final/terminating states.

an extra time step since it has no pdf. Therefore s_{T+1} actually is active at the same time as s_T.

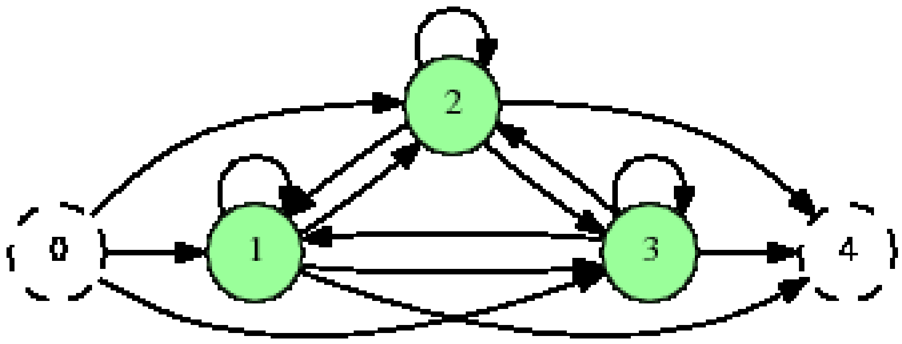


FIGURE 8.2.1. A simple fully connected HMM

8.2.4. Fundamental HMM assumptions. The above model makes two important assumptions about the relationship between the feature vectors:

(1) The observation independence assumption states that:

$$p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_0^t) = p(\mathbf{x}_t|s_t). \tag{8.2}$$

This means that the likelihood of the tth feature vector depends only on the current state and is therefore otherwise unaffected by previous states and feature vectors. This assumption is not affected by the order of the HMM.

(2) The first-order[4] Markov assumption:

$$P(s_t|s_0^{t-1}, \mathbf{x}_1^{t-1}) = P(s_t|s_{t-1}). \tag{8.3}$$

This implies that, apart from the immediately preceding state, no other previously observed states or features affect the probability of occurrence of the next state.

In addition to the above the reader will note that our notation $a_{i,j}$ for transition probabilities does not allow them to be time dependent. We assume the transition probability between two states to be constant irrespective of the time when the transition actually takes place.

[4]This can also be generalized to (more powerful) higher Markov orders.

8.2.5. A few basic HMM topologies. Topology refers to which states are connected to each other. Many configurations are popular, it usually makes sense to carefully match the topology to the characteristics of the observations that are being modeled. The most generic version is the fully connected topology shown in Figure 8.2.3. It is often used in applications where observations repeats over time (for example a text-independent speaker recognition system).

When the observations has a well-defined sequential nature, such as encountered in for instance word recognition, a left-to-right form such as shown in Figure 8.2.5 may be more appropriate.

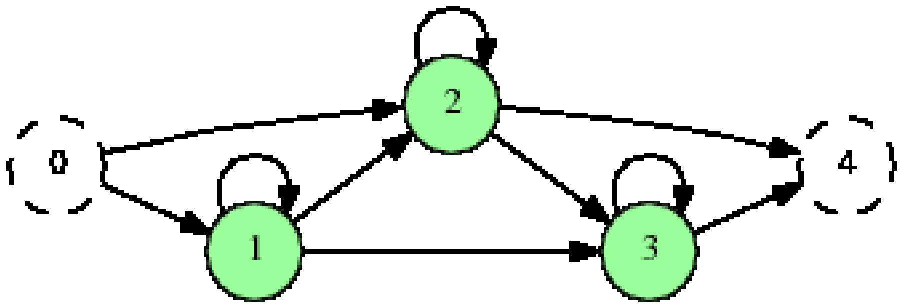


FIGURE 8.2.2. A simple left-to-right HMM

As mentioned above, many other topologies are useful, this will be discussed at a later stage.

8.3. Calculating the likelihood $p(\mathbf{x}_1^T|M)$

The crucial quantity we need to calculate in the expression for the posterior (8.1) is the likelihood $p(\mathbf{x}_1^T|M)$. This not only allows us to calculate the posterior probabilities, but also provide a way of estimating the model parameters by maximazing the likelihood for a given set of training data.

In the following we repeatedly use two results from basic probability theory namely conditional probability (the product rule)

$$p(a,b|c) = p(a|b,c)p(b|c) \tag{8.1}$$

and total probability (the marginal)

$$p(a) = \sum_i p(a, b_i) \tag{8.2}$$

where b_i forms a partition. For simplicity we will (mostly) also omit the reference to the model M.

8.3.1. A direct approach. Marginalizing, we can write the likelihood as,

$$p(\mathbf{x}_1^T) = \sum_{\forall s_0^{T+1}} p(\mathbf{x}_1^T, s_0^{T+1}), \tag{8.3}$$

where we have suppressed the model dependency, i.e. we write $p(\mathbf{x}_1^T)$ instead of $p(\mathbf{x}_1^T|M)$. Using the product rule, the Markov assumption, and the observation independence, we write,

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^{T+1}) &= P(s_{T+1}|s_0^T, \mathbf{x}_1^T)p(\mathbf{x}_1^T, s_0^T) \\ &= a_{s_T,s_{T+1}}p(\mathbf{x}_T|\mathbf{x}_1^{T-1}, s_0^T)p(\mathbf{x}_1^{T-1}, s_0^T) \\ &= a_{s_T,s_{T+1}}p(\mathbf{x}_T|s_T)p(\mathbf{x}_1^{T-1}, s_0^T). \end{aligned}$$

The last term on the right hand side is exactly of the same form as the left hand side, we can therefore recursively complete the evaluation to yield,

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^{T+1}) &= a_{s_T,s_{T+1}}p(\mathbf{x}_T|s_T)a_{s_{T-1},s_T}p(\mathbf{x}_{T-1}|s_{T-1})\ldots \\ &\quad \ldots a_{s_1,s_2}p(\mathbf{x}_1|s_1)P(s_1|s_0)P(s_0) \\ &= a_{s_T,s_{T+1}}p(\mathbf{x}_T|s_T)a_{s_{T-1},s_T}p(\mathbf{x}_{T-1}|s_{T-1})\ldots \\ &\quad \ldots a_{s_1,s_2}p(\mathbf{x}_1|s_1)a_{0,s_1} \end{aligned} \tag{8.4}$$

Note that the final state s_{T+1} is a non-emitting, terminating state previously indicated as $s_{T+1} = N+1$. Since all the values on the right hand side are either known, or can be readily calculated, it would seem that we have succeeded in providing an approach towards calculating (8.3). The snag lies in the $\forall s_0^{T+1}$. In a fully connected model with N states and T time steps the number of possible state sequences is N^T—which very rapidly becomes prohibitively large[5]. We need to find another method that can do this more efficiently.

8.3.2. The forward algorithm. Let us consider the calculation of the so-called forward likelihoods,

$$\begin{aligned}
\alpha_t(j) &= p(\mathbf{x}_1^t, s_t = j), \quad t = 2, \ldots, T\text{and } j = 1, \ldots, N \\
&= p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_t = j)p(\mathbf{x}_1^{t-1}, s_t = j) \\
&= p(\mathbf{x}_t|s_t = j)\sum_{i=1}^{N} p(\mathbf{x}_1^{t-1}, s_{t-1} = i, s_t = j) \\
&= p(\mathbf{x}_t|s_t = j)\sum_{i=1}^{N} P(s_t = j|s_{t-1} = i, \mathbf{x}_1^{t-1})p(\mathbf{x}_1^{t-1}, s_{t-1} = i).
\end{aligned}$$

Recognizing that the last term on the right hand side is $\alpha_{t-1}(i)$, and using the Markov assumption this reduces to a very usable recursive form,

$$\alpha_t(j) = p(\mathbf{x}_t|s_t = j)\sum_{i=1}^{N} a_{i,j}\alpha_{t-1}(i),$$

or if we want to state it more generally to include state 0,

(8.5)
$$\alpha_t(j) = \begin{cases} p(\mathbf{x}_t|s_t = j)\sum_{i=1}^{N} \alpha_{t-1}(i)a_{i,j} & \text{with } t = 1, \ldots, T, \;\; j = 1, \ldots, N \\ 0 & \text{with } t = 1, \ldots, T, \;\; j = 0. \end{cases}$$

To start the recursion at $t = 0$ we need $\alpha_0(j) = P(s_0 = j)$,

$$\alpha_0(j) = \begin{cases} 1 & \text{with } j = 0 \\ 0 & \text{with } j > 0 \end{cases} \tag{8.6}$$

The total likelihood of the data given the model fits very nicely into this framework. Starting by marginalizing out all posible states that lead to termination, we get,

[5]with $N = 500$ and $T = 300$ we already have approximately 10^{800} paths!

$$\begin{aligned} p(\mathbf{x}_1^T) &= \sum_{j=0}^{N} p(\mathbf{x}_1^T, s_T = j, s_{T+1} = N+1) \\ &= \sum_{j=0}^{N} P(s_{T+1} = N+1|s_T = j, \mathbf{x}_1^T) p(s_T = j, \mathbf{x}_1^T) \\ &= \sum_{j=0}^{N} a_{j,N+1}\alpha_T(j). \end{aligned} \tag{8.7}$$

In a fully connected model with N emitting states and T time steps the number of calculations is now $O(N^2T)$, which is very do-able indeed.

8.3.3. Preventing numerical over-/underflow. Inspecting equation 8.5 reveals repeated multiplication of quantities that are either probabilities and therefore in the [0:1] range, or high-dimensional pdf values that, although they could have any positive value, are likely to be very small. In practical terms the iterative calculation of the above α's *will* underflow and we need to take precautions against this. Two approaches are popular:

- Rescale the α's: After all the $\alpha_t(j)$'s at a specific time t have been calculated, divide them by their sum $\sum_j \alpha_t(j)$ while also recording this sum in log format in a separate variable. Accumulating these log-sums over time will yield the exact factor which should be added to the calculated/scaled $\log p(\mathbf{x}_1^T)$ to yield its correct value.
- Work with $\log \alpha_t(j)$ throughout: Working in log format should prevent any underflow problems. However, now we need to be careful with the summation in equation 8.5 which still needs to be done in the linear domain. Directly converting these log values to linear before summation will immediately re-introduce the underflow problems we are trying to prevent. We come up against expressions such as $L = \log(e^{L_1} + e^{L_2} + \ldots + e^{L_M} + \ldots + e^{L_N})$ where none of the individual terms e^{L_n} are expressible in linear form. This is not as daunting as it might seem at first and can be calculated as $L = L_M + \log(e^{L_1 - L_M} + e^{L_2 - L_M} + \ldots + 1 + \ldots + e^{L_N - L_M})$ where $L_M = \max(L_1, L_2, \ldots L_N)$.

8.4. Calculating the most likely state sequence: The Viterbi algorithm

We see in (8.3) that the total likelihood of the data involves a sum of all possible state sequences s_0^{T+1}. Due to the 'hiddenness' of the HMM, we never know with certainty which of those many state sequences actually give rise to our observed feature vectors $\mathbf{x}_1^T$. But at least one of them will provide the biggest contribution to this sum. This is the most likely state sequence. We can find it easily with a small modification to (8.5) and (8.7). If we replace the summations there with maximizations, while also recording which specific previous state resulted in each such a maximum, we have calculated $p(\mathbf{x}_1^T, S^*)$ where S^* denotes the most likely state sequence. This most likely state sequence can be recovered by recursively backtracking from state $N+1$ to its most likely predecessor etc, until we reach state 0. It is left as an exercise for the reader to determine why this procedure results in the optimal state sequence (hint: the above is a dynamic programming algorithm). Once again one has to give consideration to possible underflow problems. However, by using these Viterbi maximizations, the total calculation has reverted to a single long sequence of multiplications, exactly as we found in equation 8.4. It now becomes very simple to work in the log domain, this sequence of multiplications simply changes to additions.

8.5. Training/estimating HMM parameters

The GMM we encountered in a previous chapter, weighed and combined a collection of observation pdfs. Because we did not know which of the basis-pdfs we should associate a specific feature vector (i.e. there was missing/latent information), we had no closed form solution for estimating its parameters. We had to resort to the EM algorithm to supply an interactively improving estimate. This estimate converged only to a locally optimal set of parameters, there is no guarantee that the solution is the best possible one.

The HMM also combines a collection of observation pdfs. This combining is even more complex now, since the time order of observations also plays an important role. Once again there is missing information, in this case we do not know for sure which state we should associate each feature vector with. Similar to the GMM case we will find that we can estimate the model parameters via the EM algorithm, and once again the solution will only have the guarantee of being locally optimal.

8.5.1. If the (hidden) state sequence is known. Suppose for the moment that for each sequence of training observations/feature vectors $\mathbf{x}_1^T$ we somehow knew the corresponding state sequence s_1^T. From this it is fairly easy to estimate the model parameters.

- **Transition probabilities:** For the transition probabilities we simply have to count how often various states follow on each specific source state.

$$\hat{a}_{i,j} = \frac{\#(s_t = i, s_{t+1} = j)}{\#(s_t = i)}, \tag{8.1}$$

  where we use ˆ to indicate an estimate, and # to indicate the count of the number of occurrences of its argument.
- **Observation pdfs:** Similarly we would calculate the parameters of the state pdfs $p(\mathbf{x}|s = i)$ by simply collecting all the feature vectors associated with each state i and from them estimate the pdf parameters using techniques we have encountered before. This will depend on the specific pdf being used, it could be as simple as calculating the mean and variance vectors of a diagonal Gaussian[6].

Since the transition probabilities and state pdfs comprise our full HMM, it therefore is quite simple to estimate the model parameters *if* we know the (hidden/latent) state sequences. But of course we, unfortunately, do not. However, *if* we somehow knew the HMM parameters, we can use the Viterbi algorithm to estimate the optimal state sequence S^* for each $\mathbf{x}_1^T$. Using this estimated state sequence should be a good stand-in for the the true state sequence. So we have a chicken-egg situation here. With known state sequences we can calculate the model parameters and with known model parameters we can calculate optimal state sequences. But unfortunately both of these 'knowns' are actually unknown. This is where the EM algorithm, here in the form of the Viterbi Re-estimation algorithm, comes in.

8.5.2. The EM algorithm — Viterbi Re-estimation.

[6]It is quite common for the pdf to be a GMM.

(1) **Initialization:**
  (a) For every training observation sequence $\mathbf{x}_1^T$, assign in a sensible manner a corresponding state sequence s_1^T and extend it to s_0^{T+1} by adding the initial and termination state. This 'sensible' manner is dependent on the desired topology. For a fully connected model clustering might provide initial labels, for left-to-right models each training sequence can be subdivided in N equal portions[7].
  (b) From this initial state sequence, generate an initial model as discussed in section 8.5.1.

(2) **EM Re-estimation:**
  (a) **Expectation step:** For the current model estimate, apply the Viterbi algorithm on every training sequence $\mathbf{x}_1^T$, to calculate $\log p(\mathbf{x}_1^T, S^*)$. S^* is the expected state sequence for this observation sequence. Accumulate the 'scores' i.e. $f = \sum_{\forall \text{ training } \mathbf{x}_1^T} \log p(\mathbf{x}_1^T, S^*)$, to be used later to test for convergence.
  (b) **Maximization step:** Use *all* the S^*'s obtained in the E-step to update the parameters of the HMM as discussed in section 8.5.1.

(3) **Termination:** Compare the total score f obtained in the E-step to that obtained from the previous E-step. If it is within an acceptable tolerance, terminate, otherwise continue with the re-estimation (i.e. step 2).

8.5.3. The EM algorithm — Baum Welch Re-estimation. The reader might have noticed that in the re-estimation of GMM's, each feature vector $\mathbf{x}$ was partially/probabilistically associated with multiple basis pdfs, where-as in the Viterbi algorithm we associated it fully with only one specific state. This is more similar to what we encountered with K-means clustering where a feature vector was also only associated with one specific cluster.

For all three the above training scenarios we actually have a choice whether we want to use the 'hard' approach which associates each feature vector with only one missing label namely the most probable one, or the 'soft' approach which generalizes this by probabilistically/partially associating the feature vector with multiple labels

[7]A random state assignment typically ultimately results in an inferior local optimum and is not recommended.

in accordance to our probabilistic knowledge of the situation. Below we briefly outline this algorithm.

The 'soft' version of the Viterbi re-estimation is called Baum Welch re-estimation. To do this we need to use the so-called 'backwards algorithm' to calculate another set of likelihoods:

$$\beta_t(j) = p(\mathbf{x}_{t+1}^T | s_t = j).$$

Note that the time sequence now runs in the opposite direction, x_t is *not* included, and s_t is now a given. Similarly to the α's there is a recursive algorithm to calculate them efficiently,

$$\beta_{T+1}(j) = \begin{cases} 1 & \text{with } j = N+1 \\ 0 & \text{with } j < N+1 \end{cases} \tag{8.2}$$

and

$$\beta_t(j) = \begin{cases} \sum_{k=1}^{N+1} a_{j,k}\beta_{t+1}(k)p(\mathbf{x}_{t+1}|s_{t+1} = k) & \text{with } t = 1\ldots T, j = 1\ldots N \\ 0 & \text{with } t = 1\ldots T, j = N+1 \end{cases} \tag{8.3}$$

From these and (8.7) we can then calculate the probability that a specific state $s_t = j$ is active,

$$\gamma_t(j) = p(s_t = j|\mathbf{x}_1^T) = \frac{\alpha_t(j)\beta_t(j)}{p(\mathbf{x}_1^T)}. \tag{8.4}$$

We can also calculate the probability that a specific transition is active,

$$\xi_t(j,k) = p(s_t = j, s_{t+1} = k|\mathbf{x}_1^T) = \frac{\alpha_t(j)a_{j,k}p(\mathbf{x}_{t+1}|s_{t+1} = k)\beta_{t+1}(k)}{p(\mathbf{x}_1^T)}. \tag{8.5}$$

These then are the 'soft'/probabilistic counterpart to the 'hard'/optimal state and transition labellings resulting from the Viterbi algorithm. The EM algorithm we encountered in section 8.5.2 still applies, but now we no longer use a hard allocation of feature vectors x_t to states s_t. Instead by using equation 8.4 and 8.5 we are allocating each x_t to all states, but proportionally to the probability that it matched the state. For instance the transition probabilities equation 8.1 now become:

$$\hat{a}_{i,j} = \frac{\sum_{t=0}^{T} \xi_t(i,j)}{\sum_{t=0}^{T} \gamma_t(i)}. \tag{8.6}$$

Similar extensions apply to the estimation of state densities, but is dependent on the specific density being used. The reader is encouraged to for instance find the maximum likelihood estimate applicable if the state densities were multi-dimensional Gaussian. For establishing convergence/termination we can monitor $p(\mathbf{x}_1^T)$.

In terms of modeling accuracy this is the more general approach. In practical terms, however, the accuracy of these algorithms are mostly very similar. We are not going to investigate this further here, consult the literature for more details.

Bibliography

[1] Etham Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2010.

[2] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[4] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[5] Stephen Marsland. *Machine Learning: An Algorithmic Perspective (Chapman & Hall/CRC Machine Learning & Pattern Recognition)*. Chapman and Hall/CRC, 2009.

[6] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

Index

A

ancestral sampling, 107

B

Bayes factor, 99
Bayes' theorem, 9, 34
Bernoulli distribution, 32, 96
beta distribution, 35
bias, 51
binary split, 105
binary variable, 32
binomial distribution, 33

C

causality, 12
central limit theorem, 65
classification, 67, 78
cluster, 102
clustering, 102
Clustering, K-means, 102
combining models, 31
conditional expectation, 26
conditional independence, 17, 48
conditional probability, 8, 9
conjugate prior, 35, 62
Covariance, 26
covariance, 55

D

decision boundary, 27
dimensionality reduction, 69
Dirichlet distribution, 39
discriminative, 87
discriminative model, 30
discriminative models, 91
dynamic programming, 118

E

evidence, 10, 40, 42, 47, 48
expectation, 24, 25
Expectation Maximization, 104
expectation maximization, 109
Expectation Maximization Algorithm, 110

F

Fisher criterion, 80
forward likelihood, 134

G

gamma distribution, 62
gamma function, 36
Gaussian distribution, 49
Gaussian mixture model, 105
Gaussian, Bayes' theorem, 58
Gaussian, conditional distribution, 56
Gaussian, marginal distribution, 58

generative, 87
generative model, 30
gnome, 18

H
histogram equilization, 23
hypothesis, 39, 48

I
independent and identically distributed, 49
independent, statistically, 10

J
joint ensemble, 8

K
Kalman filter, 112

L
Lagrange multiplier, 38, 109
Linear Discriminant Analysis (LDA), 78
Logistic Regression, 94
logistic regression, multiclass, 100
logistic sigmoid, 89
loss function, 29

M
Mahalanobis distance, 53
MAP, 98, 99
marginalization, 8
Markov assumption, 133
maximum a posteriori (MAP) estimate, 41
maximum likelihood, 50
multinomial distribution, 39
multinomial variables, 37
multinoulli, 38

N
Newton's method, 99
normal distribution, 49
normal-gamma distribution, 63

O
odds, 48, 94

P
posterior odds, 89
precision, 49
principal components, 68
probability density function, 21
product rule, 9

R
random variable, 7
realization, 7
regularization, 98
reject option, 30
responsibility, 107
risk, 30

S
sample mean, 50
sample variance, 51
sampling, 23
signature verification, 12
singularity (GMM), 108
soft assignment, 78
standard deviation, 49
sufficient statistics, 60
sum rule, 8

U
unbiased estimate, 51, 60
uniform prior, 40
uninformative prior, 15

V
Variance, 26

W

whitening transformation, 70