Why the Normal Distribution?

Raul Rojas
Freie Universität Berlin

Februar 2010

Abstract

This short note explains in simple terms why the normal distribution is so ubiquitous in pattern recognition applications. As we will see, the answer comes from information theory: when we compress a data set by keeping only the mean and variance of the problem's classes, the distribution which allows us to keep on working probabilistically, while making minimal assumptions about the data, is, precisely, the normal distribution.

Why the normal distribution?

The geek store offers a t-shirt with the legend "let $\epsilon < 0 \ldots$". That is the complete punch line. Similarly, in pattern recognition applications people would raise their heads if someone said "let us modell the data clusters, about which we know almost nothing, by using an xy distribution", where "xy" is not the word "normal". In other words: you better have good reasons to propose modelling data clusters with anything else than a Gaussian, or at least a mixture of Gaussians. But then, why is the normal distribution so prevalent in pattern recognition applications? The classical book by Duda & Hart, for example, starts right with the Bayes rule and the multivariate normal distribution [1].

The Gaussian (normal) distribution, is used in many pattern recognition problems as an easy way of modelling the probability density of experimental data.

The one-dimensional Gaussian distribution is determined by just two parameters: the mean μ and the standard deviation σ of the data. The normal distribution is defined by the expression

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

Consider Fig. 1 which represents a hypotetical histogram of one class in a classi-

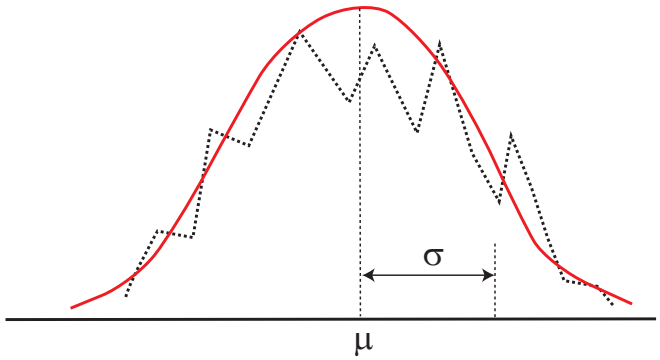

Figure 1: Histogram of data and a fitted Gaussian

fication problem. The histogram of the experimental data has a complex shape. However, for modelling purposes, we abstract from that specific shape, with its several peaks (modes) and keep only the mean μ and the variance σ^2 of the data. We then fit a Gaussian to this experimental data. Instead of keeping the whole one-dimensional data set for later comparison with new data, we compress it into just two numbers. This is an economical way of extracting useful information from a data set without having to carry it around for applications (as is done with the k-nearest neighbor method, for example)

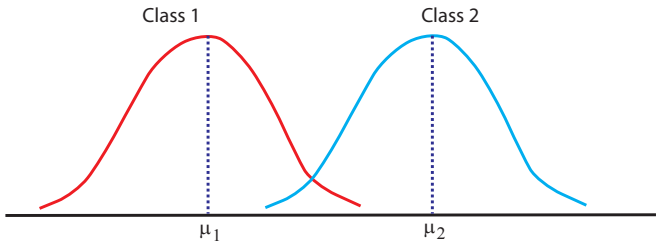

Figure 2: Two classes with Gaussian probability distributions

Given two classes of one-dimensional data-points (as shown in Fig. 2), we can use fitted Gaussians to compare the probability densities for different values of x. Points to the left of μ_1, for example, have a much higher probability of belonging to Class 1 than to Class 2. If the distribution fits the data well, the comparison of probability densities provides a useful classifier.

Normality

The Gaussian distribution enjoys a privileged role in statistics because it is so ubiquitous (so "normal"), appearing in many different experimental settings, and because many other distributions approach the normal distribution as soon as they become "messy". The binomial distribution, for example, converges to the normal distribution for a large numbers of Bernoulli trials. The Central Limit Theorem of probability theory tells us that a sum of identically distributed independent variables has, in the limit, a normal distribution. And so on [2]. As Jaynes has pointed out, the normal distribution seems to be the center of the galaxy of distributions towards which all other distributions gravitate [3].

There is an explanation for this: The Gaussian distribution is the distribution of maximum disorder (or maximum entropy) among all distributions with a given mean μ and variance σ^2. Therefore, when we apply the Gaussian distribution to pattern recognition problems, we do because we are trying to avoid special cases. We try to remain as general as possible without jumping to premature conclusions about the shape of the data cloud. To understand this we have to talk about the entropy concept in information theory.

Enter Entropy

Given a discrete set of data points (consider them messages) $x_1, x_2, \ldots, x_n$, each one of them with probability $p_1, p_2, \ldots, p_n$ of being pulled out from a data set (or of being transmitted, if they are messages), the entropy E of the distribution is given by

$$E(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log(p_i)$$

The entropy of the distribution is therefore just the expected value of negative $\log(p_i)$ for $i = 1, \ldots, n$. The negative logarithm of p_i is a measure of the *information content* of the data point (or message) x_i. If the data point has low probability, then negative $\log p_i$ is high, and we assign a high information content to the message (if something happens only occasionally, we are more surprised). If $p_i = 1$, then the point (or message) x_i does not convey new information (since we knew what was coming).

Why the log of the probability p_i? For the following reason: We can think of the information content of a piece of data as the number of questions we would have to ask (in a binary tree) in order to pinpoint the message being received. Asking those questions is equivalent to decoding the message. Assume, for example, that a message can be any of the numbers $1, 2, 3 \ldots, 8$, and all eight numbers are equally probable. In a guessing game, where someone picks one of these numbers and the questioner has to guess the number, the binary tree in Figure 3 would reveal the label of the digit using a minimum number of questions, that is, just three in average.

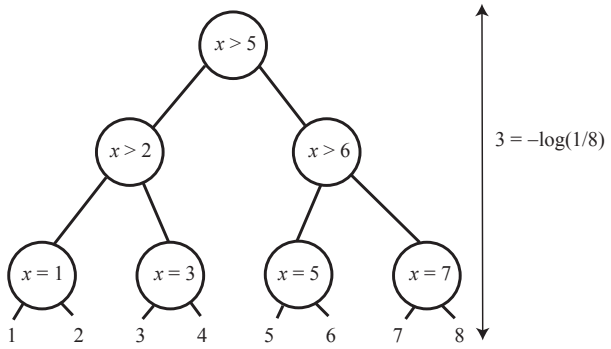

Figure 3: A guessing tree for eight possible messages

In this example, any of the messages has the same information content, that is, 3 bits, which is also the *expected information content* of all 8 messages, and also the number of bits we need in order to encode all of them.

If the probability of each digit being picked is skewed, there could be a better tree for decoding the data points. If, for example, digits 1 and 2 occur half of the time, a better decoding tree would be the one shown in Fig. 4.

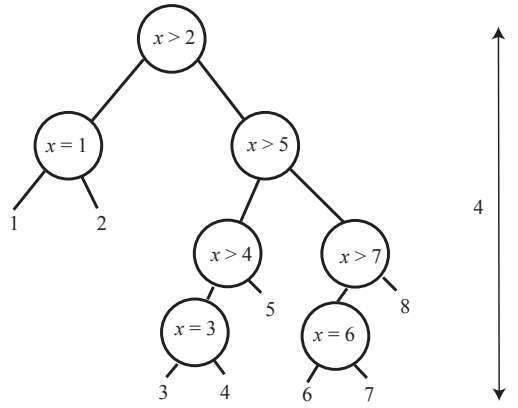

Figure 4: A better guessing tree when messages 1 and 2 occur half of the time

As the tree shows, we only need two questions to detect the message 1 or 2, which occur half of the time. We can compute the expected value of the number of questions for the new guessing tree, and we will obtain something lower than 3. The Huffman encoding algorithm builds this kind of optimal guessing trees for any discrete probability distribution of symbol appearance.

Why this measure of entropy?

It is easy to see why negative $\log(p_i)$ is a good measure of information content. First of all, notice that it is an additive measure, that is, if we receive two independent messages with respective probabilities p_1 and p_2, the total information obtained is $-log(p_1 p_2) = -(log(p_1) + log(p_2))$. This means that the information content of both messages can be added.

Now, assume that we want to decode messages using binary trees (as explained above). What would be the average depth of the binary tree for a given distribution of messages (or data points)?

To see this, suppose that we have an experiment producing a result 1 with probability p and a result 0 with probability $q = 1 - p$. Suppose that we repeat this experiment m times and transmit the results. If there is some regularity in the data, we can use less than one bit for each result. If, for example, the result is $1111\ldots$ we can encode the whole m bits with a single bit. If the result is of the type $11001100\ldots$, that is, each bit appears twice consecutively, we can encode

every two experiments with a single bit, and so on. Therefore, depending on the probability of occurrence of the result being "1" or "0", we need a lower or higher number of bits to optimally (as economically as possible)transmit the result of the m experiments.

Now, the number X of expected different outcomes of the m experiment is given by

$$X = \binom{m}{pm} = \frac{m!}{(pm)!((1-p)m))!}$$

Here we use the fact that $pm + (1-p)m = m$. Let our binary guessing tree have depth n. We need at least a number of leaves equal to X in order to be able to uniquely decode all expected histories of the m experiments. That means that in the limit of large m we can always encode what happened at the source in such a way as to be able to decode it using the tree.

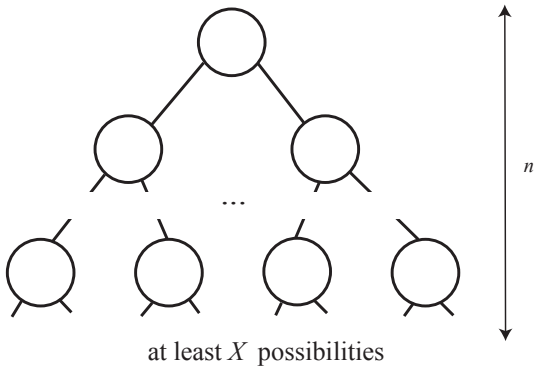

Figure 5: A guessing tree for decoding X possible experiment histories

Therefore, since we are using a binary tree with a potential number of leaves equal to 2^n, we require

$$2^n \geq X \quad \text{i.e.} \quad n \geq \log X$$

Using the approximation $n! = n\log n - n$, we can write

$$\begin{aligned} n &\geq m\log(m) - m - (pm)\log(pm) + pm - ((1-p)m)\log((1-p)m) + (1-p)m \\ &\geq (pm + (1-p)m)\log(m) - (pm)\log(pm) - ((1-p)m)\log((1-p)m) \\ &\geq pm\log\left(\frac{m}{pm}\right) + (1-p)m\log\left(\frac{m}{(1-p)m}\right) \end{aligned}$$

and therefore

$$\frac{n}{m} \geq -p \log p - (1-p) \log(1-p)$$

The result tells us, that if we can ask n questions for the m experiments, we need an average of n/m questions for each single experiment, and n/m must be larger than the entropy of a single experiment. The expression $-p \log p - (1-p) \log(1-p)$ is the average number of bits that we need to encode the outcome of a single experiment. If $p = 1$ or $p = 0$, in both cases the outcome is certain, and we need 0 bits to encode the result. We need 1 bit when $p = 1/2$, and less than one bit when $p \neq 1/2$. This is a wonderful result: it tells us that there must be a coding of the experiment outcomes that needs less than one-bit for experiment, whenever $p \neq 1/2$, without having us to provide the exact coding.

Working from first principles, Claude Shannon proposed this entropy formula in 1948 as a measure of information content. It is very useful in the field of pattern recognition, because there we handle *information* about different classes of data. It can be proved that starting from a simple sets of requirements for an information measure the entropy formula give above is the only one which fulfills all requirements [3], [4].

Maximum Entropy

A data set where each point is equally probable has maximum entropy (or disorder). If we are given 16 letters of the alphabet and each one of them appears equally often, and independently of each other, we need exactly 4 bits to encode the 16 letters. If we do not put any constraints on the data, the uniform distribution has maximum entropy. Now, assume that we are given data points from a specific problem (binarization of documents, for example) and we decide to keep only the mean μ and the variance σ^2 as descriptors of the data. The question would be, which distribution, among the many possible probability distributions, should we use later on to describe the data? Which is the distribution which makes no additional spurious assumptions? Which distribution most effectively *models our ignorance* by maximizing disorder? Not surprisingly, the answer is that the Gaussian distribution is the one with maximum entropy.

Gaussians Maximize Entropy

The proof that the Gaussian distribution maximizes entropy is rather simple. Here I slightly simplify the proof in [3]. If the sets $\{p_i\}_n$ $\{u_i\}_n$ represent two discrete probability distributions, with $\Sigma p_i = \Sigma u_i = 1$, and from the inequality $\log x \leq (x-1)$, we deduce

$$\Sigma p_i \log \frac{u_i}{p_i} \leq \Sigma p_i \left(\frac{u_i}{p_i} - 1 \right) = \Sigma u_i - \Sigma p_i = 0$$

but then

$$\Sigma p_i \log \frac{u_i}{p_i} = \Sigma p_i \log \frac{1}{p_i} + \Sigma p_i \log u_i \leq 0$$

and therefore

$$E(p_i, \ldots, p_n) \leq -\Sigma p_i \log u_i$$

Equality is attained when $p_i = u_i$. Now, with the benefit of hindsight, let us take a discrete distribution given by

$$u_i = \frac{1}{\sqrt{2\pi}\sigma} e^{-\lambda_1 x_i - \lambda_2 (x_i - \mu)^2} \tag{1}$$

From the previous inequality we deduce

$$\begin{aligned} E(p_1, \ldots, p_n) &\leq \Sigma p_i \left(\lambda_1 x_i + \lambda_2 (x_i - \mu)^2 + \log \sqrt{2\pi}\sigma \right) \\ &\leq \log \sqrt{2\pi}\sigma + \lambda_i < x_i > + \lambda_2 < (x_i - \mu)^2 > \end{aligned}$$

Therefore, the entropy function is bounded by an expression involving the expected values of x_i and their quadratic deviation from the mean. We need to find λ_1 and λ_2 so that the function we picked fulfills the constraints (remember: μ and σ are given). Since the canonical Gaussian

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \tag{2}$$

has mean μ and standard deviation σ, it is easy to see that we need $\lambda_1 = 0$ and $\lambda_2 = \frac{1}{2\sigma^2}$ to equate Eq. 1 with Eq. 2.

Let us stand back and see what we have achieved. The entropy $E(p_1, \ldots, p_n)$ of any discrete distribution fulfilling the given constraints is bounded by the expression

$$\log \sqrt{2\pi}\sigma + \frac{1}{2\sigma^2}\sigma^2 = \log \sqrt{2\pi}\sigma + \frac{1}{2}$$

The maximum is achieved when we take the Gaussian distribution with the given mean and variance. Also, the form of the upper bound expression implies that the mean of the distribution is irrelevant for the computation of the entropy bound! We can always displace a Gaussian without changing the amount of disorder (think of the continuous case where the Gaussian is defined over all the reals). The spread of the Gaussian is the important point. The more variance the distribution has, the larger the entropy. In the limit of very large σ, the normal distribution converges uniformly to the uniform distribution.

All of these coments apply also to the continuous case, which is proved using the same techniques (and some variational calculus).

Conclusions

When in a pattern recognition problem we refer to a data class by its mean and variance, we are in fact compressing the original data. We delete all additional information possibly hidden in the data set. But we gain simplicity. If we keep only the mean and variance of the data, the distribution which does not "jump to conclusions", that is, the most general distribution given such constraints, is the Gaussian distribution. When we model data using the normal distribution we are trying to be as general as possible and we are trying to avoid introducing spurious expectations in the distribution. We are in fact recognizing our ignorance about the real data distribution.

In applications, when dealing with real data, we expect a normal distribution (for example for the height of people or for the darkness of ink pixels in an OCR task). Most of the time, however, we have no real clue as to the specific probability distribution. In such cases, we play it safe, assuming a Gaussian distribution.

References

[1] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification* (2nd Edition), Wiley-Interscience,2000.

[2] Jagdish K. Patel, Campbell B. Read, *Handbook of the Normal Distribution*, Statistics: Textbooks and monographs series, Vol. 40, Marcel Dekker Inc, 1996.

[3] E. T. Jaynes, *Probability Theory: the Logic of Science*, Cambridge University Press, 2003.

[4] David J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.