Lecture 9: A Probabilistic and Generative Approach to Classification

Iain Styles

8 November 2019

Linear and Quadratic Discriminant Analysis

k-nearest-neighbours is a classification technique that is *discriminative*: it is able to discriminate between different classes of data on the basis of their nearest neighbours in the training set. We will now look at a different approach to classification based on a *generative* model. The basic idea is that we build probabilistic models of the data classes and use these to derive explicit boundaries between the classes (Figure 1). Thus, training learns where the boundaries lie; inference determines where new data lies in relation to the boundaries. We will develop the mathematics for a linear binary classifier for which the decision boundary will be a linear combination of variables that discriminates between two classes of point (i.e. binary).

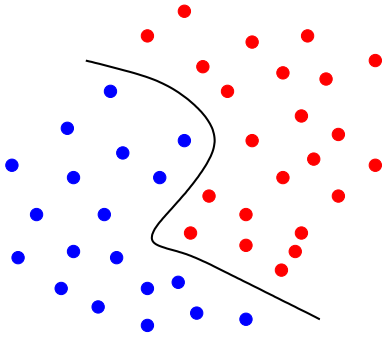

Figure 1: An example of an explicit classification boundary between two classes.

Our approach will make use of Bayes' theorem and we will therefore need to define the probability density functions that represent our prior knowledge and the likelihood. First we define our **prior knowledge** of the data: we define prior probabilities

$$P(\mathbf{x} \in \Pi_i) = \pi_i \tag{1}$$

that express the probability that a randomly selected observation $\mathbf{x}$ is in class Π_i. This requires us to understand something of the prior distribution of the classes. For MNIST, the prior is uniform and the same for all classes because the data is balanced, and so $\pi_i = 0.1$ for all classes i. This will, of course, not be the case for other datasets.

We now define the likelihood of an observation, given that it is coming from a specified class, as the conditional probability

$$P(\mathbf{x}|\Pi_i) = f_i(\mathbf{x}) \tag{2}$$

Note that we do not yet define what this probability density function looks like – we will do this later.

Given these quantities, a simple application of Bayes' theorem allows us to calculate the *posterior* probability $P(\Pi_i|\mathbf{x})$, that is, the probability that a given point $\mathbf{x}$ belongs to class Π_i. For a two-class model we have

$$P(\Pi_i|\mathbf{x}) = \frac{P(\mathbf{x}|\Pi_i)P(\Pi_i)}{P(\mathbf{x})} \tag{3}$$

$$= \frac{f_i(\mathbf{x})\pi_i}{f_1(\mathbf{x})\pi_1 + f_2(\mathbf{x})\pi_2} \tag{4}$$

So, if we know the prior distributions of the groups, and the distribution of points within each of the groups, then we can compute the probability of a specific point being in each of the groups, and

assign the point to the class to which it belongs with the highest probability. That is,

$$\frac{P(\Pi_1|\mathbf{x})}{P(\Pi_2|\mathbf{x})} > 1 \mapsto \mathbf{x} \in \Pi_1 \tag{5}$$

otherwise $\mathbf{x} \in \Pi_2$. Re-expressing in terms of f and π we find that $\mathbf{x}$ belongs to Π_1 if

$$\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} > \frac{\pi_2}{\pi_1} \tag{6}$$

otherwise to Π_2. If the ratios are equal, then randomly (and with equal probability) assign the point to either class.

In order to implement this, we either need to know or to assume something about the properties of our classes. We will assume the following for a simple two-class binary classifier:

- Each data point belongs to exactly one of exactly two distinct and identifiable groups, Π_1 and Π_2
- The two groups are normally distributed with different means $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ but identical *covariances* $\mathbf{\Sigma}$.

Covariance is a measure of the extent to which two variables change together. It is defined for two variables X and Y as $\Sigma_{X,Y} = \mathbb{E}\left[(X - \bar{X})(Y - \bar{Y})\right]$ (where $\bar{X}$ is the mean value of X). and is related to the correlation $\rho_{x,y}$ by $\rho_{x,y} = \Sigma_{x,y}/\sigma_x\sigma_y$. As such, positive covariance implies that two variables increase/decrease together, whilst a negative covariance implies that as one increases, the other tends to decrease. For a multivariate problem, we can compute a *covariance matrix* $\mathbf{\Sigma}$ with components

$$\Sigma_{ij} = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^{(n)} - \bar{x}_i\right)\left(x_j^{(n)} - \bar{x}_j\right) \tag{7}$$

which describe the covariance between the variables x_i and x_j, computed over N datapoints. It should be noted that when we compute the mean and covariance from data, they are known as the *sample mean* and *sample covariance* and are not necessariy the same as the true mean and covariance. This is especially problematic when we only have a small number of data points. **Caution: care is needed when computing the sample covariance from data.**

There is a simple way to compute $\mathbf{\Sigma}$ if we organise our dataset in a sensible way. First, we construct the matrix $\mathbf{X}$ with components $X_{ij} = x_i^{(j)}$; that is, each *row* of $\mathbf{X}$ corresponds to a variable, and each *column* corresponds to a sample. Then, compute the mean of every row (variable) and tile the resulting column vector side-by-side N times such that every column is identical and every row contains the mean value of the corresponding variable in every column. We will call the new matrix $\bar{\mathbf{X}}$. Then, we form the covariance matrix as

$$\mathbf{\Sigma} = \frac{1}{N-1}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\left(\mathbf{X} - \bar{\mathbf{X}}\right)^{\mathrm{T}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}. \tag{8}$$

Given the covariance matrix, we can then build a probabilistic model of the class-conditional likelihood, which is a multivariate distribution where the variables are *not* independent. This can be modelled, for classes $n = \{1, 2\}$, as:

$$P(\mathbf{x}|\Pi_n) = f_n(\mathbf{x}) = \frac{1}{(2\pi)^{r/2}\,|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}}_n)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}}_n)\right) \tag{9}$$

We have made two strong assumptions in writing down this distribution. First, we assume that each class obeys a normal distribution. Secondly, we assume that the covariance of each class is the same. **Beware: this does not mean that the covariance of each class is the same as the covariance of their union: the classes have different means.** There are no guarantees that this will hold for any particular example. The method can be modified to permit different covariance structure for each class and we will do this shortly. Notice that if the variables are independent, the covariance matrix is diagonal and contains only the variances of each variable. This allows the distribution to be factorised into product form, which can simplify the calculations. We will not consider this further here.

These probability density functions model the distributions of the points in our two classes, and permit us to formulate the classification rule. We will now derive a rule that maximally separates the two groups. First, consider the logarithm of the ratio of the two probability distributions. The normalising prefactor is the same in both cases and so we need only consider the exponentials which are "cancelled" by the logarithm:

$$\ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = (\mathbf{x} - \bar{\mathbf{x}}_1)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}}_1) - (\mathbf{x} - \bar{\mathbf{x}}_2)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}}_2) \tag{10}$$

$$= (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) \tag{11}$$

This is *linear* in $\mathbf{x}$: the second term on the RHS of this equation is independent of $\mathbf{x}$ and is a scalar quantity. We write

$$L(\mathbf{x}) = \ln \frac{f_1(\mathbf{x})\pi_1}{f_2(\mathbf{x})\pi} = \ln \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} + \ln \frac{\pi_1}{\pi_2} = \mathbf{M}^{\mathrm{T}}\mathbf{x} + c, \tag{12}$$

which is linear in $\mathbf{x}$ and has "gradient"

$$\mathbf{M} = \mathbf{\Sigma}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \tag{13}$$

and "intercept"

$$c = -\frac{1}{2}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2) + \log_e \frac{\pi_1}{\pi_2} \tag{14}$$

With $L(\mathbf{x})$ written in this way, and by reference to Eq. (6) we have the classification rule

$$\text{if } L(\mathbf{x}) > 0 \text{ assign } \mathbf{x} \text{ to } \Pi_1 \tag{15}$$

else assign $\mathbf{x}$ to Π_2. The "straight line" defined by Eq. (13) defines a boundary between the classes.

The particular form of LDA we have introduced here is *Gaussian LDA*, and the the function $\mathbf{M}^{\mathrm{T}}\mathbf{x}$ is referred to as *Fisher's linear discriminant function*.

The application of LDA is straightforward: the class means and covariance are estimated, and this permits equations (13) and (14) to be evaluated. In practice it is common to make these estimations from the training data in the absence of better information, subject to the caveats noted earlier.

LDA makes two strong assumptions about our data. First, we have imposed a Gaussian distribution on the data without any justification for doing so. This will hold for some problems, but is not guaranteed to hold in the general case. Caution is also necessary when working with small sample numbers because the estimates of the parameters of the distribution (mean, covariance) will be subject to significant uncertainty. Secondly, we have assumed that the classes differ only by their means, and that they have the same covariance structure. This is a very strong assumption that is very unlikely to be true in most situations, although this may not stop LDA from being an effective classifier.

Fortunately, it is not too difficult to extend the analysis to the case where the covariances of the classes are not equal. A significant quantity of algebra is required and we do not reproduce the derivation here[1] If the covariances of the two groups are given by $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ then the discriminant function (Eq. (12)) becomes

[1] See Chapter 8 of Izenman, Multivariate Statistical Analysis, Springer (New York) 2013

$$Q(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c \tag{16}$$

with

$$\mathbf{A} = -\frac{1}{2}\left(\mathbf{\Sigma}_1^{-1} - \mathbf{\Sigma}_2^{-1}\right) \tag{17}$$

$$\mathbf{b} = \mathbf{\Sigma}_1^{-1}\bar{\mathbf{x}}_1 - \mathbf{\Sigma}_2^{-1}\bar{\mathbf{x}}_2 \tag{18}$$

$$c = -\frac{1}{2}\left(\log_e \frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_2|} + \bar{\mathbf{x}}_1^{\mathrm{T}}\mathbf{\Sigma}_1^{-1}\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2^{\mathrm{T}}\mathbf{\Sigma}_2^{-1}\bar{\mathbf{x}}_2\right) - \log_e \frac{\pi_1}{\pi_2} \tag{19}$$

and the classification rule being

$$\text{if } Q(\mathbf{x}) > 0 \text{ assign } \mathbf{x} \text{ to } \Pi_1 \tag{20}$$

else assign $\mathbf{x}$ to Π_2. This is known as the *Quadratic discriminant*.

Multiclass LDA

So far, our analysis has been restricted to binary classification. Many problems involve more than two classes so an extension to multiple classes is useful, although the algebra is significantly more complex. The general argument is that we compute the pairwise relative probabilities as before and form the discriminant between classes i and j as

$$L_{ij}(\mathbf{x}) = \log_e\left(\frac{P(\Pi_i|\mathbf{x})}{P(\Pi_j|\mathbf{x}}\right) = \log_e\left(\frac{f_i(\mathbf{x})\pi_i}{f_j(\mathbf{x})\pi_j}\right) \tag{21}$$

and then $\mathbf{x}$ is assigned to Π_i if $L_{ij} > 0$ for all $j \neq i$. From this it follows from the earlier argument that the discriminant function is

$$L_{ij}(\mathbf{x}) = \mathbf{m}_{ij}^{\mathrm{T}}\mathbf{x} + c_{ij} \tag{22}$$

with

$$\mathbf{m}_{ij} = \left(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\right)^{\mathrm{T}} \mathbf{\Sigma}^{-1} \text{ and} \tag{23}$$

$$c_{ij} = -\frac{1}{2}\left(\bar{\mathbf{x}}_i^{\mathrm{T}}\mathbf{\Sigma}^{-1}\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j^{\mathrm{T}}\mathbf{\Sigma}^{-1}\bar{\mathbf{x}}_j\right) + \log_e \frac{\pi_i}{\pi_j}. \tag{24}$$

One key point to note about this is we are drawing boundaries in space and it is not obvious that these are necessarily consistent with each other. The obviously problematic situation is shown in Figure 2, where in the shaded gray region in the middle, $P1 < P3$, $P3 < P2$, and $P3 < P1$ and we cannot resolve the class (1, 2, 3) in that region. However, a moment's thought explains why this situation is impossible. The class boundaries have been constructed from the lines $P1 = P2$, $P1 = P3$, and $P2 = P3$ and therefore they *must* meet at a single point where $P1 = P2 = P3$. The division of space is self-consistent *by design*.

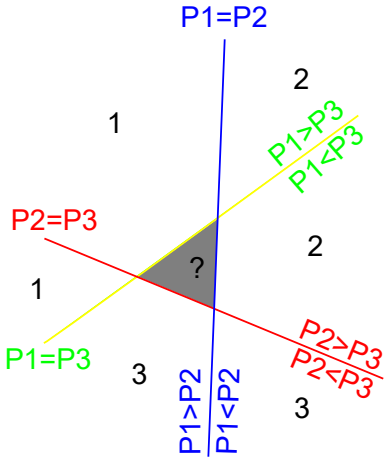

Figure 2: Decision boundaries in multiclass LDA. The shaded gray region in the middle is impossible by construction.

LDA on MNIST

Let us see how LDA performs on MNIST digit classification. This is a ten-class problem and we will therefore need to use multiclass LDA. LDA requires that we model the distribution of data within each class as a Gaussian, and that the covariance of each class distribution is the same. We will not attempt to determine whether this is true for this problem, because this is, in general, a very difficult thing to do. Instead, we assume that this is the case and see how well we can do. Note that this is extremely common practice, although perhaps should not be encourage. As we will see, it can work very well even if the assumptions are not met.

Our model for the distributions of the data in each of the classes is the multivariate normal distribution (Equation (9)). First, we note that the prefactor is the same in all cases if we assume the same covariance for each class, so we can ignore this because we will be taking ratios of the class distributions (Equation (21)). Secondly, we need to compute the mean of each class $\{\bar{\mathbf{x}}_i\}_{i=0}^{9}$. This is a straighforward calculation that we do by taking the mean of each pixel for the images in that class. Thirdly, we need to compute the covariance that we assume to be the same for each class. With this, we have to be careful. It is tempting to simply compute the covariance of the whole dataset, but this is not the correct thing to do: the covariance of the dataset is not the same as the class covariance because the classes are in different locations in the image space. Since we take the covariances to be the same for all class, we estimate this by aligning the means of the classes and then computing the covariance over the whole dataset. Note that we must align the means because otherwise the covariance matrix will be modelling

inter-class rather than intra-class variations. In practice, this means that we substract the class mean from every point in the dataset, and then compute the covariance, before moving the points back to their original position. This approach has many problems from a statistical perspective, but seems to work well in practice.

A further issue arises in the MNIST problem because of the non-varying pixels in the data. These lead to entire rows/columns of the covariance matrix being all-zero and are therefore not linearly independent. This means that the necessary matrix inversion cannot be performed. We therefore have to remove these from the data before computing the covariance.

Finally, we compute the class priors $\{\pi_i\}_{i=0}^{9}$ by computing the proportion of the training set that belongs to each class. This gives us all of the information we need.

We train the model by computing the class means, the covariance, and the class priors, from the training data. From this, we can compute the class separation boundaries. In practice, though, it is not necessary to compute these explicitly. Instead, we can classify a new data point $\mathbf{x}$ by simply computing $\arg\max_i f_i(\mathbf{x})\pi_i$.

The results of applying this to the MNIST test set as shown in Figure 3. The overall classification accuracy is 83%. This might be regarded as impressive given tha naïvety of the approach we have taken here. There is no guarantee that the data obeys LDAs assumptions and we have sought no assurances that LDA will perform well on this data. Furthermore, by comparison with k-nn, LDA is computationally very efficient. Whilst there is no training in k-nn, inference has time complexity $\mathcal{O}(N^2)$, where N is the number of samples (pairwise distance have to be computed between all pairs of points). In LDA, training and inference both have time complexity $\mathcal{O}(NM)$ (where M is the number of classes) – a huge saving in this case where $M \ll N$.

Generative Properties of LDA

The basis of LDA is that we "fit" a normal distribution to each class of data and use the lines of equality between the classes to define the boundaries between them for classification purposes. Because we have learned the distribution of the data in the form of the class conditional distribution, we are able to resample from the distributions and *generate new samples*. This can potentially be useful in situations where data is limited, but is increasingly being used as an end in itself to generate realistic synthetic examples. The state of the art in this area uses generative adversarial networks (GANs), a technique in which a generator and discriminator (real/not real) are trained against each (adversarially) such the the generator tried to fool the discriminator that its samples are real, and the discriminator tries to guess whether the sample it sees is real or not. By co-training the discriminator and generator, both can simulataneously improve which leads to vastly improved quality of generated sam-

T \ P	0	1	2	3	4	5	6	7	8	9
0	0.96	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.01	0.00
1	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
2	0.09	0.03	0.73	0.03	0.02	0.00	0.03	0.01	0.05	0.00
3	0.07	0.00	0.02	0.82	0.00	0.02	0.00	0.01	0.03	0.01
4	0.03	0.01	0.01	0.00	0.88	0.00	0.01	0.00	0.01	0.05
5	0.05	0.01	0.00	0.04	0.01	0.79	0.02	0.01	0.04	0.02
6	0.12	0.01	0.01	0.00	0.02	0.03	0.79	0.00	0.02	0.00
7	0.09	0.03	0.01	0.01	0.02	0.00	0.00	0.77	0.00	0.07
8	0.04	0.03	0.01	0.03	0.02	0.04	0.01	0.00	0.80	0.03
9	0.04	0.01	0.00	0.01	0.06	0.00	0.00	0.04	0.01	0.83

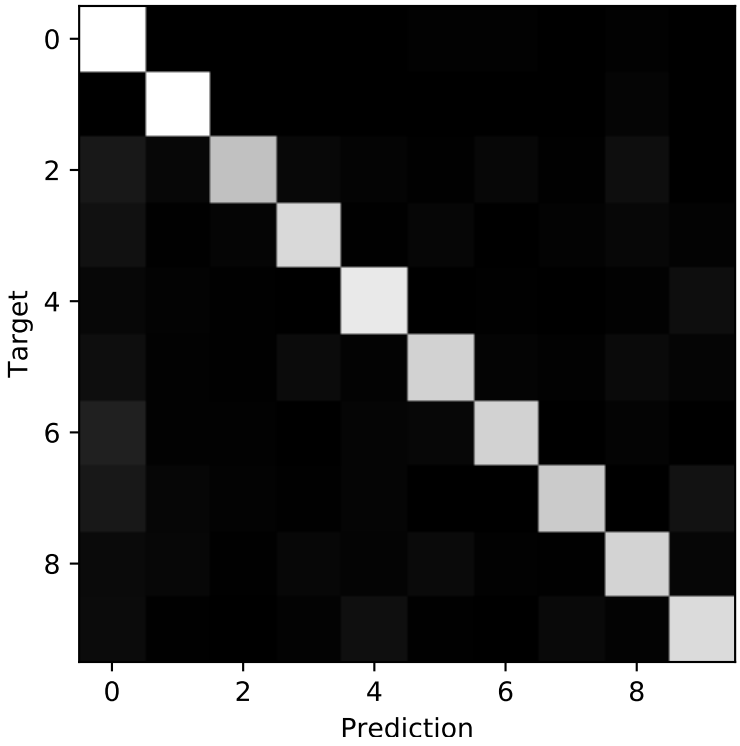

Figure 3: Confusion table and matrix for multiclass LDA on the MNIST dataset.

ples. The best current approach to this is "BigGAN" from Google Brain, described at `https://arxiv.org/abs/1809.11096`.

Our model is nowhere near as sophisticated as this: we have assumed that all classes are normally distributed and have the same covariance, and will simply resample from the estimated distributions. Ten generated samples from each class are shown in Figure 4. Some of these samples are identifiable, others are less so. This is far from the state-of-the-art, but the distributions are clearly capturing the essence of the data, although certainly not the details.

Figure 4: Ten examples of generated samples from MNIST, using the distributions learned by multiclass LDA.

Reading

The derivations here follow Chapter 8 of Izenman, Multivariate Statistical Analysis, Springer (New York) 2013. In Bishop's Pattern Recognition and Machine Learning, Chapter 4 covers this topic and more in detail.