IKM 1st Practice

1 Introduction and Overview

1.1 Bayesian Approach to Classification

Some classifiers rely on probability theory. Essentially they try to predict a probability of class (y) given features (x) – $p(y|x)$. But as you can imagine, there is hardly any application where we could sample every possible x so many times to have a reasonable statistic about y. Moreover, if x is continuous, the problem becomes intractable. Bayes rule is a straightforward application of one of the axioms of conditional probability. It states that the unreachable a posteriori probability ($p(y|x)$) can be expressed as a fraction:

$$p(y,x) = p(x,y) \tag{1}$$

$$p(y|x)p(x) = p(x|y)p(y) \tag{2}$$

$$p(y|x) = \frac{\overbrace{p(x|y)}^{\text{likelihood}}\overbrace{p(y)}^{\text{prior}}}{\underbrace{p(x)}_{\text{marginal likelihood}}} \tag{3}$$

The divisor – marginal likelihood can be omitted in the comparison of two (or more) a posteriori probabilities because it acts as a multiplicative constant. For this reason, a bayesian approach to binary classification problems can be simplified into

$$\text{prediction} = \arg\max_i p(x|y_i)p(y_i). \tag{4}$$

One question remains: How to model the likelihood ($p(x|y)$). We could try complex statistical models like a Gaussian mixture model, but it is usually computationally expensive and may create additional problems like overtraining. A good rule of thumb is to use simpler models first and then improve based on results. The central limit theorem states that if a random variable (in our case $\sim p(x|y)$) is the outcome of a summation of a number of independent random variables, its pdf approaches a normal distribution. It is quite sensible to model likelihood as a normal distribution - it has only a few parameters and theoretically nice properties. The multivariate normal distribution is defined as

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{5}$$

Where Σ is a (co)variance matrix ($D \times D$), μ is a D-dimensional mean vector. Combining equation 4 and 5, we obtain

$$\text{prediction} = \arg\max_i p(y_i|x) = \arg\max_i p(x|y_i)p(y_i) = \arg\max_i \mathcal{N}(x|\mu_i,\Sigma_i)p(y_i). \tag{6}$$

1.2 Quadratic Discriminant Analysis

Let's investigate the properties of decision hyperplanes of such a classifier. In the case of binary classification, the decision hyperplane can be obtained by the comparison of two probabilities:

$$\mathcal{N}(\mu_0, \Sigma_0)p(y_0) \overset{?}{<} \mathcal{N}(\mu_1, \Sigma_1)p(y_1) \tag{7}$$

$$\frac{\mathcal{N}(\mu_0, \Sigma_0)p(y_0)}{\mathcal{N}(\mu_1, \Sigma_1)p(y_1)} \overset{?}{<} 1 \tag{8}$$

$$\underbrace{x^T\left(-\Sigma_0^{-1} + \Sigma_1^{-1}\right)x}_{\text{quadratic term}} + \underbrace{2\left(\mu_0^T\Sigma_0^{-1} - \mu_1^T\Sigma_1^{-1}\right)x}_{\text{linear term}} - \underbrace{\mu_0^T\Sigma_0^{-1}\mu_0 + \mu_1^T\Sigma_1^{-1}\mu_1 - c}_{\text{constant term}} \overset{?}{<} 0 \tag{9}$$

Where

$$c = \log\left(|\Sigma_0|\right) - \log\left(|\Sigma_1|\right) - 2\log\left(p(y_0)\right) + 2\log\left(p(y_1)\right).$$

So the resulting method – quadratic discriminant analysis (QDA) produce quadratic classifier, and the decision boundaries of such a classifier would be hyperquadrics (i.e., ellipsoids, parabolas, hyperbolas, etc.); see wiki for pictures. In practice, we keep only a discriminative function ($q_i(x)$) defined as

$$q_i^{\text{QDF}}(x) = -x^T\Sigma_i^{-1}x + 2\mu_i^T\Sigma_i^{-1}x - \mu_i^T\Sigma_i^{-1}\mu_i - \log\left(|\Sigma_i|\right) + 2\log\left(p(y_i)\right) \tag{10}$$

The final classification is as simple as

$$\text{prediction} = \arg\max_i q_i(x) \tag{11}$$

But there is a catch: It makes sense to use QDA only if we have enough data. Most of the time, it is sensible to use at least one sample for each model's free variable. In this case, the number of free parameters[1] is

$$C\left(\underbrace{\frac{D\left(D+1\right)}{2}}_{\Sigma} + \underbrace{D}_{\mu}\right) + \underbrace{C-1}_{p(y)}$$

C is the number of classes.

Let's put it into perspective: The popular task of recognizing handwritten digits (see figure 1) has $28 \times 28 = 784$ pixels per sample and ten classes. So we would need at least 315,569 examples (1 example per parameter) while only 60,000 are provided for training. Without any dimensionality reduction technique, the classifier will make an arbitrary decision about boundaries which probably hinders our efforts.

1.3 Linear Discriminant Analysis

We can modify QDA to reduce the number of free variables. Let's assume that $\forall_i(\Sigma_i = \Sigma)$. Then the discriminative function in eq. 10 become linear because the quadratic term in eq. 22 will be canceled out. The linear discriminative function is defined as

$$q_i^{\text{LDF}}(x) = 2\mu_i^T\Sigma^{-1}x - \mu_i^T\Sigma^{-1}\mu_i + 2\log\left(p(y_i)\right). \tag{12}$$

This method is called Linear discriminative analysis[2] (LDA). For LDA, the required number of samples is significantly reduced to

$$\underbrace{DC}_{\mu} + \underbrace{C-1}_{p(y)}$$

The covariance matrix Σ is not an effective parameter because it is not directly responsible for the prediction. Our MNIST example then needs 7,849 training examples which is seven times lower than what is being provided by the dataset.

The question is whether risking with a model that can make a random choice about decision surface is better than assuming something about data that is not true and hoping it will not make a difference.

[1] One additional class can be spared by subtracting a particular discriminative function from all the others.

[2] Fisher's linear discriminant analysis is something different.

1.4 Gaussian Naive Bayes

Let's assume that the covariance is diagonal, i.e., features do not affect each other. The classifier is still quadratic and equation 22 can be split by dimension. In other words, the classifier creates likelihoods for each dimension separately, assuming conditional independence given class. Then the Gaussian Naive Bayes uses the product rule – a product of probabilities of independent events is a probability of the joint event (eq. 13). If all dataset features are independent, the QDA produces the same result as Gaussian Naive Bayes. Unfortunately, this rarely happens. For example, let's consider dataset classifying people, then weight and height are expected to be dependent features because taller individuals are generally heavier. On the other hand, violating the assumption does not mean that the classifier won't work entirely, and many techniques make the data at least linearly independent, e.g. Singular Value Decomposition and its generalized variants.

$$\hat{y} = \arg\min_i p(y_i) \prod_{j=1}^{D} p(x_j|y_i) = \arg\min_i p(y_i) \prod_{j=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_{i,j}} \exp -\frac{1}{2} \left(\frac{x_j - \mu_{i,j}}{\sigma_{i,j}} \right)^2 \tag{13}$$

1.5 General Naive Bayes

The Naive Bayes is a more general classifier then the Gaussian Naive Bayes. The difference is in the statistical modelling – the Naive Bayes allows any distribution to be the likelihood, whereas Gaussian Naive Bayes allows only the normal distribution. Other aspects are the same – the Naive Bayes assumes conditional independence for all features given class. Hence, there is no consideration for other features when fitting the distributions of the classifier. The Naive Bayes is not restricted to continuous feature spaces; it is a popular classifier whenever binary or categorical variables hold the most information.

$$\hat{y} = \arg\min_i p(y_i) \prod_{j=1}^{D} p(x_j|y_i) \tag{14}$$

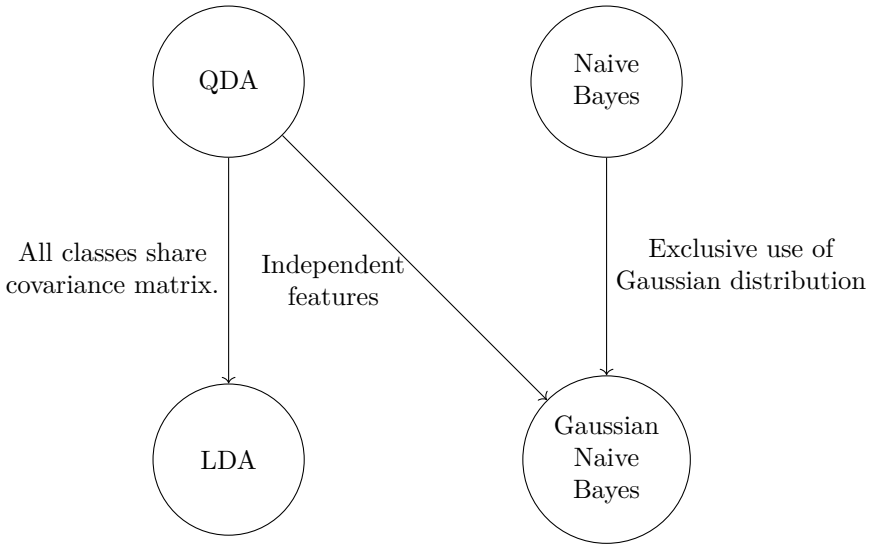


Figure 1: The relationships between all four types of probabilistic classifiers mentioned up until now.

2 Iterative Classifiers

Commonly, the distribution of parameters is unknown. In this case, it may be safer not to rely on the central limit theorem and instead focus on the important part of classification – the decision line.

2.1 Logistic Regression

The naive approach to the search for a reasonable decision border is to try every possible one on our data. As you can imagine, another approach must be found for continuous spaces for which this method is intractable.

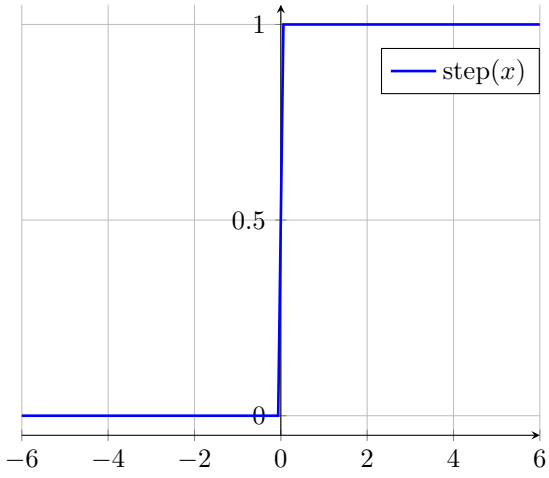
We want a function that produces a class prediction given point in space. An example of such a function is the step function (see figure 2a), which restricts optimization efforts because of the flat, non-informative regions and sharp slope. A gradiet-descent optimization cannot be used because it needs informative derivatives. On the other hand, the sigmoid function (see fig. 2b and equation 15) makes such an optimization possible. The slope of the sigmoid function can be shaped to create something that acts as a probability of class instead of the label itself. It needs to be pointed out that this value is not a probability. The value converges to the probability only if the assumptions of a classifier are not violated and the gradient descent succeeds.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

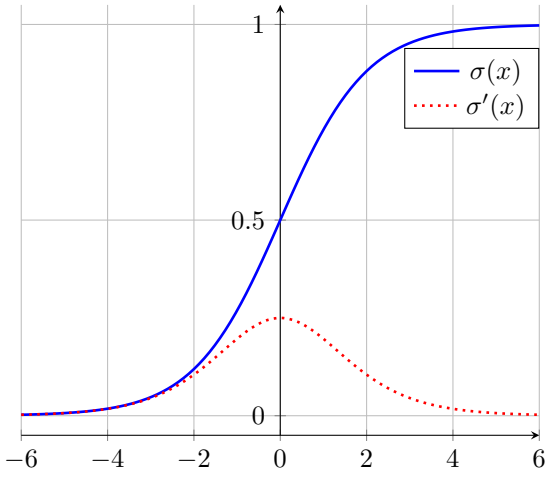
Logistic Regression is a linear classifier. Therefore samples from feature space must be linearly transformed into a scalar to use the sigmoid function. The most intuitive transformation is a dot product which can be geometrically interpreted as an orthogonal projection of a sample from feature space into a normal vector of the decision hyperplane. In one-dimensional case, the final sigmoid function may look like

$$f(x) = \frac{1}{1 + \exp\left(-(b_0 + b_1 x_1)\right)},$$

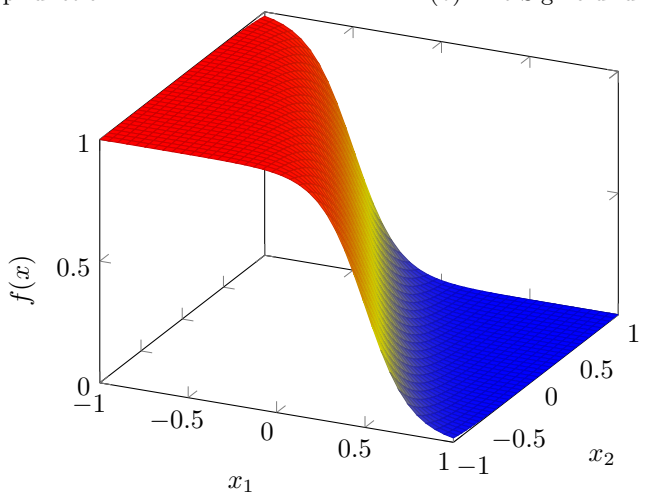
where b_0 and b_1 are trainable parameters of the classifier and x_1 is the 1D feature. In practice, it is implemented as a dot product, an example of 2d decision boundary is depicted in figure 2c. The nature of the selected optimization algorithm used on the non-convex function does not guarantee finding the best decision boundary every time. It may be helpful to fit the model multiple times with different random seeds (initial placement of the sigmoid) to improve the performance.



(a) The Step function



(b) The Sigmoid function



(c) An example of the Sigmoid function on 2d dataset

Figure 2: Forms of the decision boundary functions.

Appendix A Derivation of Decision Boudaries for Quadratic Discriminant Analysis

$$\frac{\mathcal{N}(\mu_0, \Sigma_0)p(y_0)}{\mathcal{N}(\mu_1, \Sigma_1)p(y_1)} \overset{?}{<} 1 \tag{16}$$

$$\frac{\frac{1}{(2\pi)^{D/2}|\Sigma_0|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\right) p(y_0)}{\frac{1}{(2\pi)^{D/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right) p(y_1)} \overset{?}{<} 1 \tag{17}$$

$$\left[\log\left(\frac{1}{(2\pi)^{D/2}|\Sigma_0|^{1/2}}\right) - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \log\left(p(y_0)\right)\right] \tag{18}$$

$$-\left[\log\left(\frac{1}{(2\pi)^{D/2}|\Sigma_1|^{1/2}}\right) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \log\left(p(y_1)\right)\right] \overset{?}{<} 0$$

$$\left[-\frac{1}{2}\log\left(|\Sigma_0|\right) - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \log\left(p(y_0)\right)\right] \tag{19}$$

$$-\left[-\frac{1}{2}\log\left(|\Sigma_1|\right) - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \log\left(p(y_1)\right)\right] \overset{?}{<} 0$$

$$\left[-(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\right] - \left[-(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] \overset{?}{<} c \tag{20}$$

$$-\left[x^T\Sigma_0^{-1}x - 2\mu_0^T\Sigma_0^{-1}x + \mu_0^T\Sigma_0^{-1}\mu_0\right] + \left[x^T\Sigma_1^{-1}x - 2\mu_1^T\Sigma_1^{-1}x + \mu_1^T\Sigma_1^{-1}\mu_1\right] \overset{?}{<} c \tag{21}$$

$$\underbrace{x^T\left(-\Sigma_0^{-1} + \Sigma_1^{-1}\right)x}_{\text{quadratic term}} + \underbrace{2\left(\mu_0^T\Sigma_0^{-1} - \mu_1^T\Sigma_1^{-1}\right)x}_{\text{linear term}} - \underbrace{\mu_0^T\Sigma_0^{-1}\mu_0 + \mu_1^T\Sigma_1^{-1}\mu_1 - c}_{\text{constant term}} \overset{?}{<} 0 \tag{22}$$