Generative models

September 15, 2020

Abstract

Topics: generative models; Gaussian discriminative analysis; naive Bayes classifier; Laplacian smoothing.

1 Generative models

We have used probability distributions, such as Gaussian, Bernoulli, and multinomial, to model the output variable y, with the motivation of deriving a probabilistic interpretation of a classifier. Now we want to do the same for the input variable $\mathbf{x}$, which can be continous or discrete and be modeled by different probability distributions.

Using a probability distribution to describe $\mathbf{x}$ is equivalent to describing how $\mathbf{x}$ is generated from the probability distribution, thus the name "generative models". Generative models can be used in other problems such as clustering and neural networks. Here we focus on classification problems.

2 Gaussian discriminative analysis (GDA)

Let the I.I.D. training data be $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ and $y^{(i)} \in \{0,1\}$. For an specific training example $(\mathbf{x}^{(i)}, y^{(i)})$, we want to calculate the joint probability distribution

$$\Pr(\mathbf{x}^{(i)}, y^{(i)}) = \Pr(y^{(i)})\Pr(\mathbf{x}^{(i)}|y^{(i)}). \tag{1}$$

$y^{(i)}$ follows the Bernoulli distribution with parameter ϕ so that $\Pr(y^{(i)}=1)=\phi$.

Let's assume $\mathbf{x}^{(i)}$ is a n-dimensional continuous feature vector and follow a n-dimensional Gaussian distribution for the class $y^{(i)} \in \{0,1\}$:

$$\Pr(\mathbf{x}^{(i)}|y^{(i)}=c) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})^\top \Sigma^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})\right\}. \tag{2}$$

Figure 6 shows the variaous 2-dimensional Gaussian density functions and contour plots.

Note that the mean vector $\boldsymbol{\mu}_{y^{(i)}}$ of the Gaussian is tied to the class label $y^{(i)}$, indicating that there are two Gaussians located in two places if $\boldsymbol{\mu}_0 \neq \boldsymbol{\mu}_1$. The covariance matrix Σ is the same for the two classes for simplicity but can be generalized to different covariance matrices for different classes (see PRML Section 4.2).

Therefore, the probability (or likelihood) to generate the I.I.D. m training examples is

$$\begin{aligned}
& L(\phi, \mu_0, \mu_1, \Sigma; \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) && (3)\\
= & \prod_{i=1}^m \Pr(y^{(i)})\Pr(\mathbf{x}^{(i)}|y^{(i)}) && (4)\\
= & \prod_{i=1}^m \phi^{y^{(i)}}(1-\phi)^{y^{(i)}} \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})^\top \Sigma^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})\right\}. && (5)
\end{aligned}$$

Compare this likelihood function to that of *discriminative models*, such as linear regression or logistic regression, the above *generative model* consider not just the probability $\Pr(y^{(i)})$ of the output $y^{(i)}$ but also the conditional probability $\Pr(\mathbf{x}^{(i)}|y^{(i)})$ of the input $\mathbf{x}^{(i)}$. In contrast,

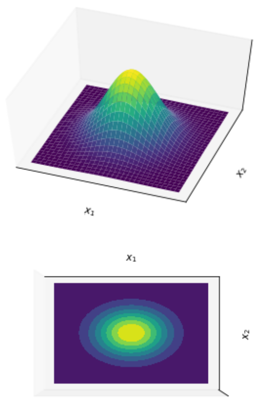

Figure 1: $\boldsymbol{\mu} = [0, 0]$, Σ is a 2×2 identity matrix.

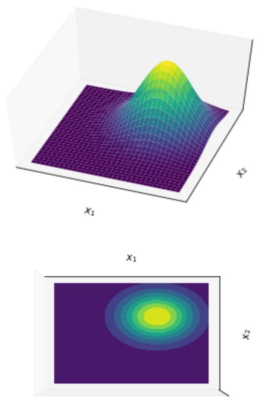

Figure 2: $\boldsymbol{\mu} = [1, 1]$, Σ is a 2×2 identity matrix.

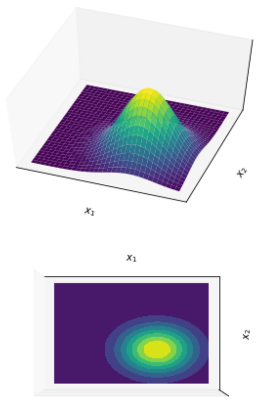

Figure 3: $\boldsymbol{\mu} = [1, -1]$, Σ is a 2×2 identity matrix.

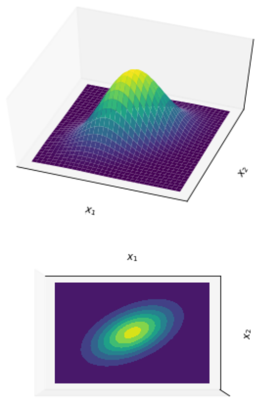

Figure 4: $\boldsymbol{\mu} = [0, 0]$, Σ is a 2×2 identity matrix plus 0.5 on the off-diagonal.

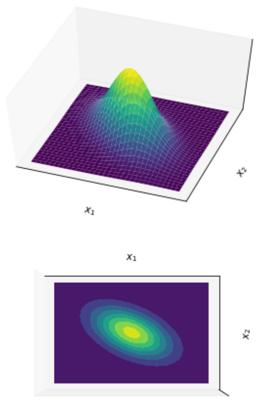

Figure 5: $\boldsymbol{\mu} = [0, 0]$, Σ is a 2×2 identity matrix plus -0.5 on the off-diagonal.

Figure 6: Gaussians with different mean vectors and covariance matrices.

the discriminative models consider the probability $\Pr(y^{(i)}|\mathbf{x}^{(i)})$, that is, given a input $\mathbf{x}^{(i)}$, the probability of see class $y^{(i)}$.

To classify a new test example $\mathbf{x}$, the generative model uses the Bayes rule to compute the *posterior*

$$\Pr(y|\mathbf{x}) = \frac{\Pr(y)\Pr(\mathbf{x}|y)}{\Pr(\mathbf{x})} \tag{6}$$

and select the class y^* that maximizes the posterior

$$y^* = \arg\max_y \Pr(y|\mathbf{x}) = \arg\max_y \frac{\Pr(y)\Pr(\mathbf{x}|y)}{\Pr(\mathbf{x})} = \arg\max_y \Pr(y)\Pr(\mathbf{x}|y) \tag{7}$$

In the above, $\Pr(y)$ is called the prior distribution of the label y and $\Pr(\mathbf{x}|y)$ is called the likelihood of $\mathbf{x}$ given y.

2.1 MLE for GDA

To train the generative model, we can estimate the parameters ϕ, $\boldsymbol{\mu}_0$, $\boldsymbol{\mu}_1$, and Σ. We work with the log-likelihood of the generative model

$$\begin{aligned} & \ell(\phi, \mu_0, \mu_1, \Sigma; \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) & (8) \\ = & \sum_{i=1}^m \left[y^{(i)} \log \phi + (1 - y^{(i)}) \log(1-\phi) - \frac{1}{2}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}})^\top \Sigma^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_{y^{(i)}}) \right] & (9) \\ & -\frac{m}{2} \log |\Sigma| + \text{constant}. & (10) \end{aligned}$$

The constant is not a function of any parameters.

2.1.1 Estimating the prior distribution

Collecting the terms in ℓ related to ϕ and recognize that the resulting problem is the MLE of Bernoulli distribution

$$\phi^{\text{MLE}} = \arg\max_\phi \sum_{i=1}^m y^{(i)} \log \phi + (1 - y^{(i)}) \log(1-\phi). \tag{11}$$

By taking the derivative of the above summation with respect to ϕ and setting the derivative to 0, you can find

$$\phi^{\text{MLE}} = \frac{\sum_{i=1}^m \mathbb{1}[y^{(i)} = 1]}{m}. \tag{12}$$

2.1.2 Estimating the Gaussian means

For estimating the mean vector of the Gaussian for class 0, collect the terms that have $\boldsymbol{\mu}_0$,

$$\sum_{i=1}^m \mathbb{1}[y^{(i)} = 0] \left[-\frac{1}{2} (\mathbf{x}^{(i)})^\top \Sigma^{-1} \boldsymbol{\mu}_0 + \boldsymbol{\mu}_0^\top \Sigma^{-1} \boldsymbol{\mu}_0 \right]. \tag{13}$$

Taking the derivative of the above summation with respect to $\boldsymbol{\mu}_0$ and setting that to 0, you can find

$$\boldsymbol{\mu}_0^{\text{MLE}} = \frac{\sum_{i=1}^m \mathbb{1}[y^{(i)} = 0] \mathbf{x}^{(i)}}{\sum_{i=1}^m \mathbb{1}[y^{(i)} = 0]}, \tag{14}$$

which is just the mean of the feature vectors that are in class 0. The mean vector for class 1 can be estimated similarly.

2.1.3 Estimating the covariance of the Gaussians

The derivation of the estimation of Σ shared by the two Gaussians is more involved. See PRML Section 4.2 for more details. The resulting estimation is quite intuitive:

$$\Sigma^{\text{MLE}} = \frac{m_1}{m} S_1 + \frac{m_0}{m} S_0 \tag{15}$$

$$S_1 = \frac{1}{m_1} \sum_{i=1}^{m} \mathbb{1}[y^{(i)} = 1] \mathbf{x}^{(i)} (\mathbf{x}^{(i)})^\top \tag{16}$$

$$S_0 = \frac{1}{m_0} \sum_{i=1}^{m} \mathbb{1}[y^{(i)} = 0] \mathbf{x}^{(i)} (\mathbf{x}^{(i)})^\top \tag{17}$$

As $\mathbf{x}$ is a column vector of length n, $\mathbf{x}\mathbf{x}^\top$ is the outer product of $\mathbf{x}$ with itself and produces a square matrix of $n \times n$. The matrices S_0 and S_1 are the MLE of covariance matrix of the Gaussian for class 0 and 1, respectively. m_1 and m_0 are the number of training examples belonging to class 1 and 0, respectively. You can interprete Eq.(15) as the weighted sum of two covariance matrices from the two classes.

3 Naive Bayes classifier

We have modeled continuous input feature vectors using Gaussian distributions. How about discrete feature vectors. Let's assume that the input feature vector $\mathbf{x}$ are binary, that is, each entry x_i of $\mathbf{x}$, $i = 1, \ldots, n$, takes value 0 or 1. The binary features motivate us to use Bernoulli distribution to model each feature.

In general, there can be dependencies among the features and the joint probability in the generative model for binary input is

$$\begin{aligned} \Pr(\mathbf{x}, y) &= \Pr(y)\Pr(x_1|y)\Pr(x_2|x_1, y)\ldots\Pr(x_n|x_1, \ldots, x_{n-1}, y) && (18) \\ &= \Pr(y)\prod_{j=1}^{n}\Pr(x_j|x_1, \ldots, x_{j-1}, y). && (19) \end{aligned}$$

To parametrize this joint distribution, here are the parameters needed: one parameter to model the prior Bernoulli distribution $\Pr(y)$, two parameters for the Bernoulli $\Pr(x_1|y)$, four parameters for the Bernoulli $\Pr(x_2|x_1, y)$, ... and 2^n parameters for $\Pr(x_n|x_1, \ldots, x_{n-1}, y)$. This is not scalable to problems that have more than a few tens of features. Besides the infeasible computation complexity, the data needed to estimate the parameters also increase with the number of parameters so that the cost of obtaining labeled data is also exponential (at least one data point is needed for each specific binary vector $\mathbf{x}$). We will see that this is a bad behavior of a machine learning model when we talk about learning theory.

Naive Bayes is a generative model that makes the simplifying assumption that the n features are independent to avoid the exponential growth of the number of parameters,

$$\Pr(\mathbf{x}, y) = \Pr(y)\Pr(x_1|y)\Pr(x_2|y)\ldots\Pr(x_n|y). \tag{20}$$

You can see that we just remove the dependencies between the features but let each feature depend only on the class label y. Each $\Pr(x_j|y)$ is a Bernoulli parametrized by ϕ_{jy}. The log-likelihood is

$$\begin{aligned} & \sum_{i=1}^{m}\left\{\sum_{j=1}^{n}\log\Pr(x_j^{(i)}|y^{(i)}) + \log\Pr(y^{(i)})\right\} && (21) \\ = & \sum_{i=1}^{m}\sum_{y=1}^{1}\left\{\sum_{j=1}^{n}\mathbb{1}[x_j^{(i)} = 1, y^{(i)} = y]\log\phi_{jy} + \log\Pr(y^{(i)} = y)\right\}. && (22) \end{aligned}$$

The MLE of $\phi_y = \Pr(y = 1)$ is $\frac{\sum_i^m \mathbb{1}[y^{(i)}=1]}{m}$, and the MLE of ϕ_{jy} is $\frac{\sum_i^m \mathbb{1}[x_j^{(i)}=1, y^{(i)}=y]}{\sum_i^m \mathbb{1}[y^{(i)}=y]}$.

To predict the class of a test example $\mathbf{x}$, naive Bayes selects the label that maximizes the posterior

$$\arg\max_y \Pr(y = 1)\prod_{j=1}^{n}\Pr(x_j|y), \tag{23}$$

where x_j is the j-th feature of $\mathbf{x}$.

3.1 Laplacian smoothing

Even with the reduction in the number of parameters to be estimated, there is a chance that the estimated Bernoulli $\Pr(x_j = 1|y) = 0$ for some j and y. This happens when no training example in class y takes value 1 on the j-th feature. If $\Pr(x_j = 1|y) = 0$, the class y will be predicted to have probability 0 for *any* test example that has value 1 on its j-th feature. Similarly, if $\Pr(x_j = 1|y) = 1$ then the opposite class $y = 0$ will not be predicted for any test example with $_j = 1$. However, such 0 probability may not be because the real world has zero probability seeing such a data point, but is because the construction of the naive Bayes model and the MLE. It is likely that the limited training can't cover all data points which have some non-zero probability to be observed.

To address this anomaly, Laplacian smoothing estimate the MLE by adding a constant λ, typically 1, to the quantities $\sum_i^m \mathbb{1}[x_j^{(i)} = 1, y^{(i)} = y]$ so that there is a non-zero chance that the even $(x_j^{(i)} = 1, y^{(i)} = y)$ will happen. This is equivalent to adding a trainig example that takes value 1 on feature x_j for each of the two classes. The addition will also increase the number of training examples from m to $m + 2$. The Laplacian estimation of ϕ_{jy} is $\frac{\sum_i^m \mathbb{1}[x_j^{(i)}=1, y^{(i)}=y]+1}{\sum_i^m \mathbb{1}[y^{(i)}=y]+1}$. You can verify that, in the event that no training data in class y has value 1 on the j-th feature, then the Laplacian estimation will be $\frac{1}{\sum_i^m \mathbb{1}[y^{(i)}=y]+1}$, which is non-zero.