MNIST Digit Classification

Derek Wayne

08 April 2019

Abstract

In this note, we explore the differences in capabilities and performance of generative and discriminative models fit to the MNIST dataset of handwritten digits. There are some readily apparent advantages to using a probabilistic classifier such as naive Bayes; generating new data and being able to handle missing data, for example. However, the naive Bayes independence assumption is often considered too strong and results in a less than optimal misclassification rate. Using a predictive model can improve performance without needing to account for correlated features.

Set up

Each datapoint is regarded as a 28 by 28 pixel greyscale image of handwritten digits ($\{0, 1, \dots, 9\}$), and a classification label indicating the true digit. Thus we can state the problem as follows: Given the image of a written digit, what is the true digit meant to be? That is, given datapoint $\mathbf{x}$, what is its true label? Our first approach will be Bayesian. We will also binzarize the pixels values, using > 0.5 as the cut-off.

Given input space $\chi = \{0,1\}^{784}$ and target space $\mathcal{Y} = \{0, 1, \dots, 9\}$ let $c \in \mathcal{Y}$ where the distribution of target c is given by $p(c|\pi) = \pi_c$. Hence for a given image $\mathbf{x}$, and class c, the joint density is given by

$$p(\mathbf{x}, c \,|\, \theta, \pi) = \pi_c \prod_{d=1}^{784} \theta_{cd}^{x_d}(1-\theta_{cd})^{1-x_d}$$

For a given class/label c and a pixel d. We model the pixel state (white or black) with the Bernoulli likelihood:

$$p(x_d \,|\, c, \theta_{cd}) = \theta_{cd}^{x_d}(1-\theta_{cd})^{1-x_d}$$

it follows that $\theta_{cd} = P\{x_d = 1 \,|\, c, \theta_{cd}\}$ for a single example. Let $\mathbf{y}$ be the $N = 60{,}000$-dimensional vector of predictions for the dataset. Then we may estimate the conditional pixel means θ_{cd} by maximum likelihood estimation as follows,

$$\widehat{\theta}_{cd} := arg \max_{\theta_{cd}} \prod_{n=1}^{N} \pi_c \prod_{d=1}^{784} \theta_{cd}^{x_d^{(n)}}(1-\theta_{cd})^{1-x_d^{(n)}}$$

$$= arg \max_{\theta_{cd}} \sum_{n=1}^{N} \left(log\, \pi_c + \sum_{d=1}^{784} x_d^{(n)} log\, \theta_{cd}^{(n)} + (1 - x_d^{(n)}) log\,(1 - \theta_{cd}^{(n)}) \right)$$

which is the solution to the likelihood equation:

$$\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\} \left(x_d^{(n)}(1-\theta_{cd}) - (1 - x_d^{(n)})\theta_{cd} \right) = 0$$

and so,

$$\widehat{\theta}_{cd,MLE} = \frac{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\} x_d^{(n)}}{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\}}$$

which is to be interpreted as the ratio of the number of examples with the dth pixel activated in class c to the total number of examples of class c, which is a natural result for the MLE. However, this estimator can be problematic if there are no examples of a particular class within the dataset (thus we have 0 in the denominator of our estimator), or the pixel value is 1 in every example in dataset (then the estimator is equal to 1).

Consider instead the maximum aposteriori estimator with a $Beta(\alpha, \beta)$ prior, given by,

$$\begin{aligned}
\widehat{\theta}_{cd,MAP} &:= arg \max_{\theta_{cd}} P\{\theta_{cd} \,|\, x_d, c\} \\
&= arg \max_{\theta_{cd}} \prod_{n=1}^{N} \pi_c \prod_{d=1}^{784} \theta_{cd}^{x_d^{(n)}} (1-\theta_{cd})^{1-x_d^{(n)}} \times Beta(\alpha,\beta) \\
&= arg \max_{\theta_{cd}} \left(\theta_{cd}^{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)}=c\} x_d^{(n)} + (\alpha-1)} (1-\theta_{cd})^{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)}=c\}(1-x_d^{(n)}) + (\beta-1)} \right)
\end{aligned}$$

where the expression inside parenthesis on the last line may be recognized as $Beta(\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\}x_d^{(n)} + \alpha\,, \sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\}(1 - x_d^{(n)}) + \beta)$ which for $\alpha, \beta > 1$ (hence unimodal), has mode:

$$\frac{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\}x_d^{(n)} + \alpha - 1}{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\} + \alpha + \beta - 2}$$

In particular if we let $\alpha = \beta = 2$ (symmetric about 1/2) we have,

$$\widehat{\theta}_{cd,MAP} = \frac{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\}x_d^{(n)} + 1}{\sum_{n=1}^{N} \mathbb{1}\{y^{(n)} = c\} + 2}$$

This is known as "Laplacian smoothing". Then the 784×10 matrix $\widehat{\theta}_{MAP}$ will have entry c, d given as derived above.

Next we will load in the MNIST dataset from the loader provided in [http://yann.lecun.com/exdb/mnist/].

```r
source('loadMNIST.R')
load_mnist()
train$x <- ifelse(train$x > 0.5, 1, 0) # binarize pixels
test$x <- ifelse(test$x > 0.5, 1, 0)
```

Utilizing the derived MAP estimates for the pixel means we fit θ to the training set:

```r
pixel_means <- function(data) {
  m <- matrix(0, nrow=784, ncol=10)
  for (i in 0:9) {
    class_set <- data$x[data$y == i, ]
    theta.c <- (colSums(class_set) + 1) / (nrow(class_set) + 2)
    m[ ,i+1] <- theta.c
  }
  m
}
theta.hat <- pixel_means(train)
```

The learned parameters are plotted in Figure 1.

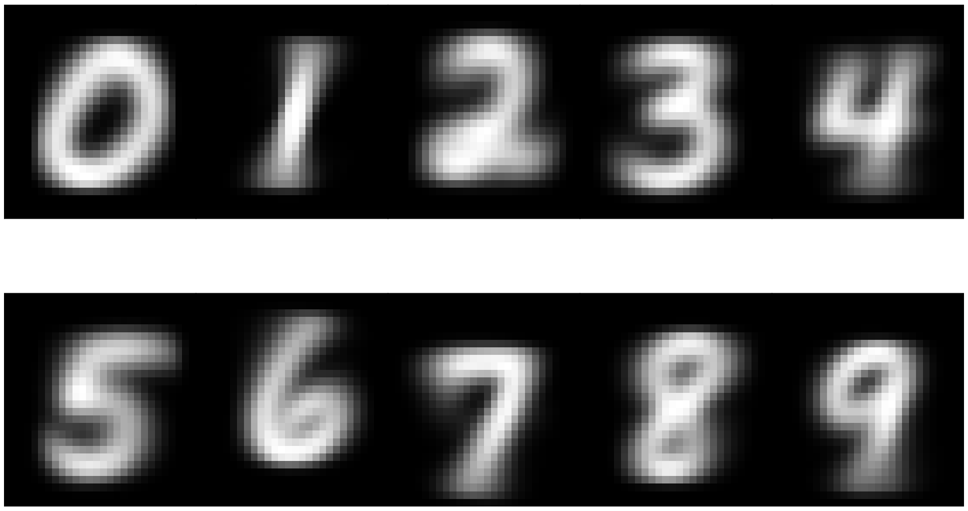
Figure 1: Condition Pixel Means