Bayesian Classification, Naive Bayes

Submit a PDF of your answers to Canvas.

1. *Univariate Gaussian.* Let $X \sim \mathcal{N}(\mu, \sigma^2)$.

a) For any $a, b \in \mathbb{R}$, $Y = aX + b$ is also Gaussian. If $X \sim \mathcal{N}(\mu, \sigma^2)$, what is the mean and variance of Y?

b) Given $X \sim \mathcal{N}(\mu, \sigma^2)$, find $a, b \in \mathbb{R}^n$ so that $aX + b \sim \mathcal{N}(0, 1)$.

c) Use the result from above to generate 10,000 independent samples of a random variable $\sim \mathcal{N}(-5, 7)$. You may want to use the NumPy function `np.random.randn()`. Plot a histogram of the results.

2. Consider a random vector $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ with mean $\mu_{\boldsymbol{x}} \in \mathbb{R}^{n\times 1}$ and covariance $\Sigma_{\boldsymbol{x}}$, a deterministic matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and a deterministic vector $\boldsymbol{c} \in \mathbb{R}^{n\times 1}$.

a) Consider the random vector $\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x} + \boldsymbol{c})$. Find an expression for $E[\boldsymbol{y}]$.

b) Find an expression for $\Sigma_{\boldsymbol{y}}$ (the covariance matrix of $\boldsymbol{y}$) in terms of $\Sigma_{\boldsymbol{x}}$.

3. *Naïve Bayes on MNIST.* In this problem we will classify MNIST handwritten digits using MAP and Naïve Bayes. The class $y \in \{0, 1, \ldots, 9\}$ represents the true digit while $\boldsymbol{x} \in \{0, 1\}^{784}$ is the 28×28 black and white image that we'd like to classify.

Recall that if we know the *posterior* (i.e, the distribution of the class given the data, $p(y|\boldsymbol{x})$), we can use the Maximum A Posteriori (MAP) estimate as a classification rule:

$$\widehat{y} = \arg\max_y p(y|\boldsymbol{x})$$

where $\widehat{y}$ is our estimate of the class. Equivalently,

$$\widehat{y} = \arg\max_y p(\boldsymbol{x}|y)p(y)$$

which follows from Bayes Rule, since $p(y|\boldsymbol{x}) = p(\boldsymbol{x}|y)p(y)/p(\boldsymbol{x})$, and then dropping $p(\boldsymbol{x})$ from the denominator, since it doesn't depend on y and is inside the arg max.

The MAP classifier is very intuitive – as it aims to find the most likely class y, given the data $\boldsymbol{x}$.

Before we can use classifier, we need to (approximately) learn $p(\boldsymbol{x}|y)$ and $p(y)$. Learning $p(y)$ is easy, since we simply count the occurrences of the classes in the training data.

Learning $p(\boldsymbol{x}|y)$ in general is hard, and even with $60k$ training examples, we don't seem to have enough data. Notice that when the images have binary values, there are 2^{784} possible images. It's difficult to estimate the probability of one out of 2^{784} possible images when we only have $60k$ training examples. There are in essence 2^{784} different parameters that we must estimate.

Naïve Bayes makes a very naïve assumption: all elements of $\boldsymbol{x}$ are independent. More precisely Naïve Bayes assumes $p(\boldsymbol{x}|y) \approx \prod_{i=1}^{n} p(x_i|y)$.

a) How many different parameters are needed to describe the distribution $p(\boldsymbol{x}|y)$ if we assume $p(\boldsymbol{x}|y) \approx \prod_{i=1}^{n} p(x_i|y)$? Assume each pixel x_i only takes on values in $\{0, 1\}$.

b) Is the independence assumption $p(\boldsymbol{x}|y) \approx p(x_1|y) \times p(x_2|y) \ldots$ a reasonable assumption for the MNIST dataset? Why or why not?

c) Note that $p(x_i|y)$ is the (marginal) probability that pixel x_i is 0 or 1 given the class y. We can estimate $p(x_i = 0|y = 9)$, for example, by counting the number of times pixel i is equal to 0, vs. the number of times it is equal to 1 among the images that are labeled $y = 9$:

$$p(x_i = 0|y = 9) = \frac{\text{count of examples of class 9 with pixel } i \text{ equal to 0}}{\text{count of examples of class 9}}. \qquad (1)$$

Write code that estimates $p(x_i = 0|y)$ and $p(x_i = 1|y)$ by counting the occurrences $x_i = 0$ and $x_i = 1$ for $y = 0, 1, \ldots, 9$ and for reach pixel i. Store your estimates of $p(x_i = 0|y)$ and $p(x_i = 1|y)$, each in ten 28×28 matrices. Two tricks to help:

- sometimes the numerator in (1) will be zero. Add a small number (for example, 0.7) to both the numerator and denominator. This is called additive or *Laplace* smoothing.
- store the log of the quantity above (to avoid multiplying small numbers when we multiply the marginal probabilities of each pixel in the next step).

d) For numerical stability, we can compute $\sum_i \log(p(x_i|y))$ instead of $\prod_i p(x_i|y))$, without changing the problem. For a new test image $\boldsymbol{x}$, write code to output an estimate:

$$\widehat{y} = \arg\max_y \sum_{i=1}^{n} \log(p(x_i|y)).$$

e) What is the classification error rate of your classifier on the $10k$ test images?

f) **(optional)** The digits are close to equally likely (i.e, $p(y = 0) \approx p(y = 1) \ldots$), so we didn't incorporate the prior probability. Incorporate the prior probabilities, as estimated from the training data, into your classifier. How does the classification error rate change?

g) **(optional)** We now have a *probablistic model* given by $p(\boldsymbol{x}|y)$ that we can use to generate random MNIST images. Display a few random images using the Naïve Bayes model you derived above. Do the images look like real MNIST images?

h) **(optional)** Randomly permute the pixels of a few images. Are the permuted pixel images recognizable by a human (display a few)? If you randomly permute the pixels and retrain your Naïve Bayes classifier, will the performance change? What does this say about our model?