

## Bayesian Classification, Naive Bayes

Submit a PDF of your answers to Canvas.

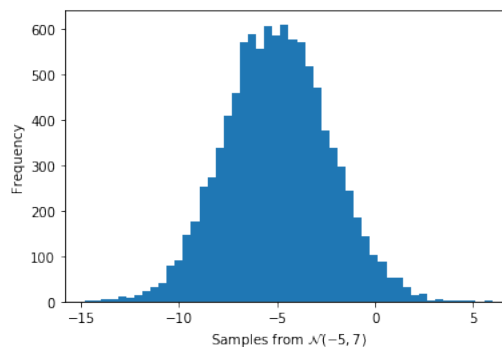
1. *Univariate Gaussian.* Let  $X \sim \mathcal{N}(\mu, \sigma^2)$ .

- 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.

### SOLUTION:

- a)  $E[aX + b] = aE[X] + b$ , so  $E[Y] = a\mu + b$ . Also,  $\text{var}(aX + b) = a^2\text{var}(X) = a^2\sigma^2$ .
- b) Setting  $a = \frac{1}{\sigma}$  and  $b = \frac{\mu}{\sigma}$  gives the desired result.
- c) See code below:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.random.randn(10000)
y = x*np.sqrt(7) - 5
plt.hist(y,50)
plt.ylabel('Frequency')
plt.xlabel('Samples from  $\mathcal{N}(-5,7)$ ')
plt.show()
```



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

- a) Consider the random vector  $\mathbf{y} = \mathbf{A}(\mathbf{x} + \mathbf{c})$ . Find an expression for  $E[\mathbf{y}]$ .
- b) Find an expression for  $\Sigma_{\mathbf{y}}$  (the covariance matrix of  $\mathbf{y}$ ) in terms of  $\Sigma_{\mathbf{x}}$ .

**SOLUTION:**

- a)  $E[\mathbf{y}] = E[\mathbf{A}(\mathbf{x} + \mathbf{c})] = \mathbf{A}E[\mathbf{x}] + \mathbf{A}\mathbf{c} = \mathbf{A}\mu_{\mathbf{x}} + \mathbf{A}\mathbf{c} = \mathbf{A}(\mu_{\mathbf{x}} + \mathbf{c})$
- b) Since the variance does not depend on the mean, we could drop the additive  $\mathbf{c}$  and save some work in our computation. Without this trick, we see:

$$\begin{aligned}
 \Sigma_{\mathbf{y}} &= E[\mathbf{y}\mathbf{y}^T] - \mu_{\mathbf{y}}\mu_{\mathbf{y}}^T \\
 &= E[\mathbf{A}(\mathbf{x} + \mathbf{c})(\mathbf{x} + \mathbf{c})^T \mathbf{A}^T] - \mathbf{A}(\mu_{\mathbf{x}} + \mathbf{c})(\mu_{\mathbf{x}} + \mathbf{c})^T \mathbf{A}^T \\
 &= \mathbf{A}E[\mathbf{x}\mathbf{x}^T + \mathbf{x}\mathbf{c}^T + \mathbf{c}\mathbf{x}^T + \mathbf{c}\mathbf{c}^T] \mathbf{A}^T - \mathbf{A}(\mu_{\mathbf{x}} + \mathbf{c})(\mu_{\mathbf{x}} + \mathbf{c})^T \mathbf{A}^T \\
 &= \mathbf{A}(\Sigma_{\mathbf{x}} + \mu_{\mathbf{x}}\mu_{\mathbf{x}}^T + \mu_{\mathbf{x}}\mathbf{c}^T + \mathbf{c}\mu_{\mathbf{x}}^T + \mathbf{c}\mathbf{c}^T - \mu_{\mathbf{x}}\mu_{\mathbf{x}}^T - \mu_{\mathbf{x}}\mathbf{c}^T - \mathbf{c}\mu_{\mathbf{x}}^T - \mathbf{c}\mathbf{c}^T) \mathbf{A}^T \\
 &= \mathbf{A}\Sigma_{\mathbf{x}}\mathbf{A}^T
 \end{aligned}$$

which does not depend on  $\mathbf{c}$  as expected.

3. *Naïve Bayes on MNIST.* In this problem we will classify MNIST handwritten digits using MAP and Naïve Bayes. The class  $y \in \{0, 1, \dots, 9\}$  represents the true digit while  $\mathbf{x} \in \{0, 1\}^{784}$  is the  $28 \times 28$  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,  $p(y|\mathbf{x})$ ), we can use the Maximum A Posteriori (MAP) estimate as a classification rule:

$$\hat{y} = \arg \max_y p(y|\mathbf{x})$$

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

$$\hat{y} = \arg \max_y p(\mathbf{x}|y)p(y)$$

which follows from Bayes Rule, since  $p(y|\mathbf{x}) = p(\mathbf{x}|y)p(y)/p(\mathbf{x})$ , and then dropping  $p(\mathbf{x})$  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  $\mathbf{x}$ .

Before we can use classifier, we need to (approximately) learn  $p(\mathbf{x}|y)$  and  $p(y)$ . Learning  $p(y)$  is easy, since we simply count the occurrences of the classes in the training data. Learning  $p(\mathbf{x}|y)$  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  $\mathbf{x}$  are independent. More precisely Naïve Bayes assumes  $p(\mathbf{x}|y) \approx \prod_{i=1}^n p(x_i|y)$ .

- a) How many different parameters are needed to describe the distribution  $p(\mathbf{x}|y)$  if we assume  $p(\mathbf{x}|y) \approx \prod_{i=1}^n p(x_i|y)$ ? Assume each pixel  $x_i$  only takes on values in  $\{0, 1\}$ .
- b) Is the independence assumption  $p(\mathbf{x}|y) \approx p(x_1|y) \times p(x_2|y) \dots$  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  $y = 9$ :

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

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  $y = 0, 1, \dots, 9$  and for each pixel  $i$ . Store your estimates of  $p(x_i = 0|y)$  and  $p(x_i = 1|y)$ , each in ten  $28 \times 28$  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  $\mathbf{x}$ , write code to output an estimate:

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

- 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,  $p(y = 0) \approx p(y = 1) \dots$ ), 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 *probabilistic model* given by  $p(\mathbf{x}|y)$  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?

**SOLUTION:**

- a)** We need to estimate  $28 \times 28 = 784$  Bernoulli parameters for each of the 10 classes.
- b)** No. There is clearly dependence between the pixels. When one pixel is dark, we expect it's neighbors to be dark as well. So the Naïve assumption is a bad assumption for MNIST, but still (maybe surprisingly) works.
- c)** See the associated notebook.