Machine Learning

Dr. Muhammad Adeel Nisar

Assistant Professor – Department of IT,
Faculty of Computing and Information Technology,
University of the Punjab, Lahore

Probabilities (recap)

Slides Courtesy: Dr. Kamran Malik

Probabilities (Recap)

- $P(X = 0) =$
- $P(Y = 3) =$
- $P(X = 1, Y = 2) =$
- $P(Y = 2, X = 1) =$
- $P(X = 1 \mid Y = 2) =$
- $P(Y = 2 \mid X = 1) =$

X	Y
0	0
0	1
1	0
1	2
2	3
2	0
2	3
1	3
1	2
0	3
0	2
0	0

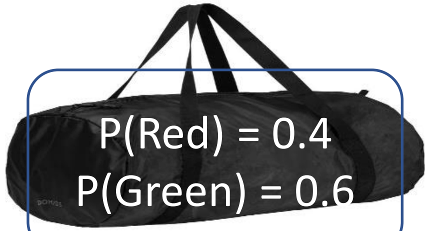
Probabilities (Recap)

- $P(X = 0) =$ **5/12**
- $P(Y = 3) =$ **4/12**
- $P(X = 1, Y = 2) =$ **2/12**
- $P(Y = 2, X = 1) =$ **2/12**
- $P(X = 1 \mid Y = 2) =$ **2/3**
- $P(Y = 2 \mid X = 1) =$ **2/4**

X	Y
0	0
0	1
1	0
1	2
2	3
2	0
2	3
1	3
1	2
0	3
0	2
0	0

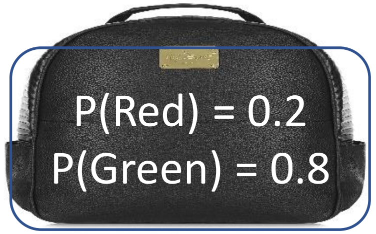
Probabilities (Recap)

P(bag1) = 0.3



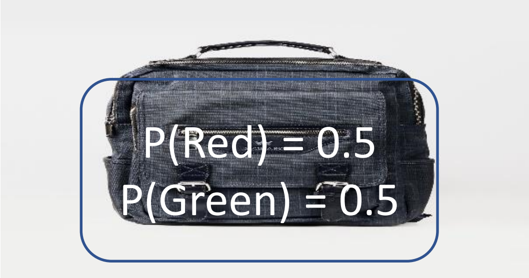
P(Red) = 0.4
P(Green) = 0.6

P(bag2) = 0.5



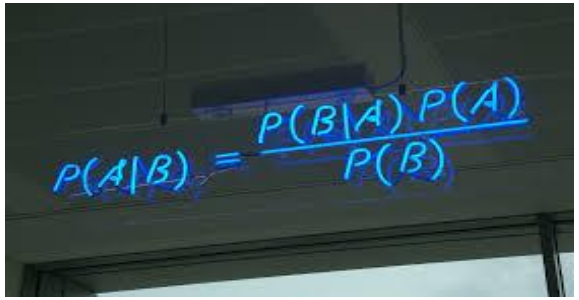
P(Red) = 0.2
P(Green) = 0.8

P(bag3)=0.2



P(Red) = 0.5
P(Green) = 0.5

- P(bag2) =
- P(Red|bag1) =
- P (Red) =
- P(bag1|Red) =



Probabilities (Recap)

- $P(A, B)$ or $P(A \cap B) = ?$
- $P(A, B) = P(A \mid B) * P(B) = P(B \mid A) * P(A)$
- $P(A, B \mid C) = P(A \mid B, C) * P(B \mid C) = P(B \mid A, C) * P(A \mid C)$
- $P(A \cap B \cap C) = P(A, B, C)$
- $P(A \cap B \cap C \cap D) = ?$

Probabilities (Recap)

if all events are independent

- $P(A, B) = P(A) * P(B)$
- $P(A, B, C) =$
- $P(A, B, C, D) =$
- $P(A1,A2, . . ., An) =$

Conditionally independent

- $P(A, B \mid C) = P(A \mid C) * P(B \mid C)$
- $P(A, B, C \mid D) =$
- $P(A1,A2, . . ., An \mid Z) =$

Naïve Bayes Classifier

(Classification Algorithms)

Naïve Bayes Classifier

$$P(x \mid c)$$

$$P(x_1, x_2, \ldots, x_n \mid c)$$

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Naïve Bayes Classifier

- Assumption

Conditional Independence: Assume the feature probabilities x are independent given the class *c.*

$$P(x \mid c)$$

$$P(x_1, x_2, \ldots, x_n \mid c)$$

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Naïve Bayes Classifier

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naïve Bayes Classifier

Bayes Rule: $$P(A\,|\,B) = \frac{P(B\,|\,A)P(A)}{P(B)}$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c\,|\,x)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(x\,|\,c)P(c)}{P(x)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}}\, P(x\,|\,c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier

- Training Data

	x1	x2	x3	x4	x5	x6	Class
$x^{(1)}$	1	0	0	1	1	1	y
$x^{(2)}$	1	1	0	1	1	1	y
$x^{(3)}$	0	1	0	0	0	0	y
$x^{(4)}$	0	0	1	0	0	1	n
$x^{(5)}$	1	1	0	1	1	0	n
$x^{(6)}$	0	1	0	1	1	0	n

P(y) = 3/6 = 1/2 P(n) = 3/6 = 1/2

P(x1=1|y) = 2/3 P(x1=0|y)= 1/3

P(x1=1|n) = 1/3 P(x1=0|n) = 2/3

P(x2=1|y) = 2/3 P(x2=0|y) = 1/3

P(x2=1|n) = 2/3 P(x2=0|n) = 1/3

P(x3=1|y) = 0/3 P(x3=0|y) = 3/3

P(x3=1|n) = P(x3=0|n) =

P(x4=1|y) = P(x4=0|y) =

P(x4=1|n) = P(x4=0|n) =

P(x5=1|y) = P(x5=0|y) =

P(x5=1|n) = P(x5=0|n) =

P(x6=1|y) = P(x6=0|y) =

P(x6=1|n) = P(x6=0|n) =

- Testing Data

$P(y|t^{(1)})$, $P(n|t^{(1)})$

x1 x2 x3 x4 x5 x6

$t^{(1)}$ 1, 1, 0, 0, 0, 1 $\boxed{y}$

$t^{(2)}$ 1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x \mid c) P(c)$$

Naïve Bayes Classifier

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

$2/3 \times \frac{2}{3} \times 1 \times \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{1}{2} = P(y|t^{(1)})$

$P(n|t^{(1)})$

P(y) = 1/2	P(n) = 1/2
P(x1=1|y) =2/3	P(x1=0|y) =1/3
P(x1=1|n) = 1/3	P(x1=0|n) = 2/3
P(x2=1|y) = 2/3	P(x2=0|y) = 1/3
P(x2=1|n) = 2/3	P(x2=0|n) = 1/3
P(x3=1|y) = 0	P(x3=0|y) = 1
P(x3=1|n) = 1/3	P(x3=0|n) = 2/3
P(x4=1|y) = 2/3	P(x4=0|y) = 1/3
P(x4=1|n) = 2/3	P(x4=0|n) = 1/3
P(x5=1|y) = 2/3	P(x5=0|y) = 1/3
P(x5=1|n) = 2/3	P(x5=0|n) = 1/3
P(x6=1|y) = 2/3	P(x6=0|y) = 1/3
P(x6=1|n) = 1/3	P(x6=0|n) = 2/3

$P(c=y|t^{(1)}) = P(x_1|y) \cdot P(x_2|y) \cdot P(x_3|y) \cdot P(x_4|y) \cdot P(x_5|y) \cdot P(x_6|y) \times P(y)$

- Testing Data

$t^{(1)} =$ 1, 1, 0, 0, 0, 1 y

1, 0, 1, 1, 1, 1 y

$P(x_1|n)$

$$\underset{c \in C}{\operatorname{argmax}} P(x|c)P(c)$$

Naïve Bayes Classifier

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

P(y) = 1/2 P(n) = 1/2

P(x1=1|y) =2/3 P(x1=0|y) =1/3

P(x1=1|n) = 1/3 P(x1=0|n) = 2/3

P(x2=1|y) = 2/3 P(x2=0|y) = 1/3

P(x2=1|n) = 2/3 P(x2=0|n) = 1/3

P(x3=1|y) = 0 P(x3=0|y) = 1

P(x3=1|n) = 1/3 P(x3=0|n) = 2/3

P(x4=1|y) = 2/3 P(x4=0|y) = 1/3

P(x4=1|n) = 2/3 P(x4=0|n) = 1/3

P(x5=1|y) = 2/3 P(x5=0|y) = 1/3

P(x5=1|n) = 2/3 P(x5=0|n) = 1/3

P(x6=1|y) = 2/3 P(x6=0|y) = 1/3

P(x6=1|n) = 1/3 P(x6=0|n) = 2/3

- Testing Data

$t^{(1)}$: 1, 1, 0, 0, 0, 1 y

$t^{(2)}$: 1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x \mid c) P(c)$$

Naïve Bayes Classifier

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

- Testing Data

$t^{(1)}$: 1, 1, 0, 0, 0, 1 y

$t^{(2)}$: 1, 0, 1, 1, 1, 1 y

P(y) = 1/2 P(n) = 1/2

P(x1=1|y) =2/3 P(x1=0|y) =1/3

P(x1=1|n) = 1/3 P(x1=0|n) = 2/3

P(x2=1|y) = 2/3 P(x2=0|y) = 1/3

P(x2=1|n) = 2/3 P(x2=0|n) = 1/3

P(x3=1|y) = 0 P(x3=0|y) = 1

P(x3=1|n) = 1/3 P(x3=0|n) = 2/3

P(x4=1|y) = 2/3 P(x4=0|y) = 1/3

P(x4=1|n) = 2/3 P(x4=0|n) = 1/3

P(x5=1|y) = 2/3 P(x5=0|y) = 1/3

P(x5=1|n) = 2/3 P(x5=0|n) = 1/3

P(x6=1|y) = 2/3 P(x6=0|y) = 1/3

P(x6=1|n) = 1/3 P(x6=0|n) = 2/3

- $P(t^{(2)}|c=y) =$

- $P(t^{(2)}|c=n) =$

$$\underset{c \in C}{\operatorname{argmax}} P(x|c)P(c)$$

Naïve Bayes Classifier (smoothing)

Naïve Bayes Classifier (Smoothing)

- A solution would be **Laplace smoothing** , which is a technique for smoothing categorical data.
- A small-sample correction, or **pseudo-count**, will be incorporated in every probability estimate.
- Consequently, no probability will be zero.
- This is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing.
- While in the general case it is often called **Lidstone smoothing**.

Naïve Bayes Classifier (after Smoothing)

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

P(y) = P(n) =

P(x1=1|y) = P(x1=0|y)=

P(x1=1|n) = P(x1=0|n) =

P(x2=1|y) = P(x2=0|y) =

P(x2=1|n) = P(x2=0|n) =

P(x3=1|y) = P(x3=0|y) =

P(x3=1|n) = P(x3=0|n) =

P(x4=1|y) = P(x4=0|y) =

P(x4=1|n) = P(x4=0|n) =

P(x5=1|y) = P(x5=0|y) =

P(x5=1|n) = P(x5=0|n) =

P(x6=1|y) = P(x6=0|y) =

P(x6=1|n) = P(x6=0|n) =

- Testing Data

1, 1, 0, 0, 0, 1 y

1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x \mid c) P(c)$$

Naïve Bayes Classifier (Smoothing)

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

P(y) = 1/2 P(n) = 1/2

P(x1=1|y) =3/5 P(x1=0|y) =2/3

P(x1=1|n) = 2/5 P(x1=0|n) = 3/5

P(x2=1|y) = 3/5 P(x2=0|y) = 2/5

P(x2=1|n) = 3/5 P(x2=0|n) = 2/5

P(x3=1|y) = 1/5 P(x3=0|y) = 4/5

P(x3=1|n) = 2/5 P(x3=0|n) = 3/5

P(x4=1|y) = 3/5 P(x4=0|y) = 2/5

P(x4=1|n) = 3/5 P(x4=0|n) = 2/5

P(x5=1|y) = 3/5 P(x5=0|y) = 2/5

P(x5=1|n) = 3/5 P(x5=0|n) = 2/5

P(x6=1|y) = 3/5 P(x6=0|y) = 2/5

P(x6=1|n) = 2/5 P(x6=0|n) = 3/5

- Testing Data

1, 1, 0, 0, 0, 1 y

1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x \mid c) P(c)$$

Programming Assignment (example)

Homework on Jupyter Notebook

```
In [25]: yes_zero_prob = 1 - yes_one_prob

In [26]: print(yes_zero_prob)
[0.33333333 0.33333333 1.         0.33333333 0.33333333 0.33333333]

In [27]: no_one_prob = no_train.sum(axis = 0) / no_train.shape[0]

In [28]: print(no_one_prob)
[0.33333333 0.66666667 0.33333333 0.66666667 0.66666667 0.33333333]

In [29]: no_zero_prob = 1 - no_one_prob
         print(no_zero_prob)
[0.66666667 0.33333333 0.66666667 0.33333333 0.33333333 0.66666667]
```

Testing Data

```
In [30]: X_test = np.array([[1,1,0,0,0,1],
                            [1,0,1,1,1,1]])
```

Homework-1:

Perform Testing

jupyter Naïve Bayes Classifier - Example Last Checkpoint

File Edit View Insert Cell Kernel Widgets Help

Run Code

Naïve Bayes Classifier

```
In [1]: import numpy as np
```

Training Data

```
In [19]: X_train = np.array([[1,0,0,1,1,1],
                             [1,1,0,1,1,1],
                             [0,1,0,0,0,0],
                             [0,0,1,0,0,1],
                             [1,1,0,1,1,0],
                             [0,1,0,1,1,0]])
         y_train = np.array([1,1,1,0,0,0])

In [20]: print(X_train)
[[1 0 0 1 1 1]
 [1 1 0 1 1 1]
 [0 1 0 0 0 0]
 [0 0 1 0 0 1]
 [1 1 0 1 1 0]
 [0 1 0 1 1 0]]

In [21]: yes_train = X_train[:3,:]
         no_train = X_train[3:,:]

In [22]: print(yes_train)
         print(no_train)
[[1 0 0 1 1 1]
 [1 1 0 1 1 1]
 [0 1 0 0 0 0]]
[[0 0 1 0 0 1]
 [1 1 0 1 1 0]
```

Using scikit-learn ¶

https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes https://scikit-lea

Naïve Bayes

- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Categorical Naive Bayes
- Complement Naive Bayes
- Bernoulli Naive Bayes

```
In [41]: from sklearn.naive_bayes import GaussianNB

In [42]: gnb = GaussianNB()
         gnb.fit(X_train, y_train)
         y_pred = gnb.predict(X_test)

In [43]: print(y_pred)
[1 0]
```

```
In [44]: from sklearn.naive_bayes import MultinomialNB

In [45]: mnb = MultinomialNB()
         mnb.fit(X_train, y_train)
         y_pred = mnb.predict(X_test)

In [46]: print(y_pred)
[1 0]

In [47]: from sklearn.naive_bayes import CategoricalNB

In [48]: cnb = CategoricalNB()
         cnb.fit(X_train, y_train)
         y_pred = cnb.predict(X_test)

In [49]: print(y_pred)
[1 1]
```

Homework-2

Read, understand and compare different types of Naïve Bayes classifiers

https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes

Naïve Bayes Classifier

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

P(y) = P(n) =

P(x1=1|y) = P(x1=0|y)=

P(x1=1|n) = P(x1=0|n) =

P(x2=1|y) = P(x2=0|y) =

P(x2=1|n) = P(x2=0|n) =

P(x3=1|y) = P(x3=0|y) =

P(x3=1|n) = P(x3=0|n) =

P(x4=1|y) = P(x4=0|y) =

P(x4=1|n) = P(x4=0|n) =

P(x5=1|y) = P(x5=0|y) =

P(x5=1|n) = P(x5=0|n) =

P(x6=1|y) = P(x6=0|y) =

P(x6=1|n) = P(x6=0|n) =

- Testing Data

1, 1, 0, 0, 0, 1 y

1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x \mid c) P(c)$$

Naïve Bayes Classifier

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

```
X_Train = np.array([[1,0,0,1,1,1],
                    [1,1,0,1,1,1],
                    [0,1,0,0,0,0],
                    [0,0,1,0,0,1],
                    [1,1,0,1,1,0],
                    [0,1,0,1,1,0]])
print(X_Train)
```

```
yes_train = X_Train[:3,:]
print(yes_train)
```

```
[[1 0 0 1 1 1]
 [1 1 0 1 1 1]
 [0 1 0 0 0 0]]
```

```
yes_train.shape
```

```
yes_one_Prob = yes_train.sum(axis=0)/yes_train.shape[0]
print(yes_one_Prob)
```

```
[0.66666667 0.66666667 0.         0.66666667 0.66666667 0.66666667]
```

```
yes_zero_Prob = 1 - yes_one_Prob
print(yes_zero_Prob)
```

```
[0.33333333 0.33333333 1.         0.33333333 0.33333333 0.33333333]
```

Naïve Bayes Classifier

- Training Data

x1	x2	x3	x4	x5	x6	Class
1	0	0	1	1	1	y
1	1	0	1	1	1	y
0	1	0	0	0	0	y
0	0	1	0	0	1	n
1	1	0	1	1	0	n
0	1	0	1	1	0	n

P(y) = 1/2　　P(n) = 1/2

P(x1=1|y) =2/3　　P(x1=0|y) =1/3

P(x1=1|n) = 1/3　　P(x1=0|n) = 2/3

P(x2=1|y) = 2/3　　P(x2=0|y) = 1/3

P(x2=1|n) = 2/3　　P(x2=0|n) = 1/3

P(x3=1|y) = 0　　P(x3=0|y) = 1

P(x3=1|n) = 1/3　　P(x3=0|n) = 2/3

P(x4=1|y) = 2/3　　P(x4=0|y) = 1/3

P(x4=1|n) = 2/3　　P(x4=0|n) = 1/3

P(x5=1|y) = 2/3　　P(x5=0|y) = 1/3

P(x5=1|n) = 2/3　　P(x5=0|n) = 1/3

P(x6=1|y) = 2/3　　P(x6=0|y) = 1/3

P(x6=1|n) = 1/3　　P(x6=0|n) = 2/3

- Testing Data

1, 1, 0, 0, 0, 1 y

1, 0, 1, 1, 1, 1 y

$$\underset{c \in C}{\operatorname{argmax}} P(x|c)P(c)$$

Programming Assignment (OCR)

Programming Assignment

- Implement an OCR system that distinguishes between the images of digits
- Four files (trainX, trainY, testX and testY)

```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train=x_train.reshape(60000,784)
x_test=x_test.reshape(10000,784)
```

Programming Assignment

```python
%matplotlib inline
import numpy as np
from matplotlib import pyplot as plt
```

```python
img2=np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
               0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,
               0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,0,
               0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,
               1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,
               0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,
               0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,
               0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
               0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
               0,0,0,0],dtype=np.uint8)
```

Programming Assignment

```
1  img2.shape
```

(256,)

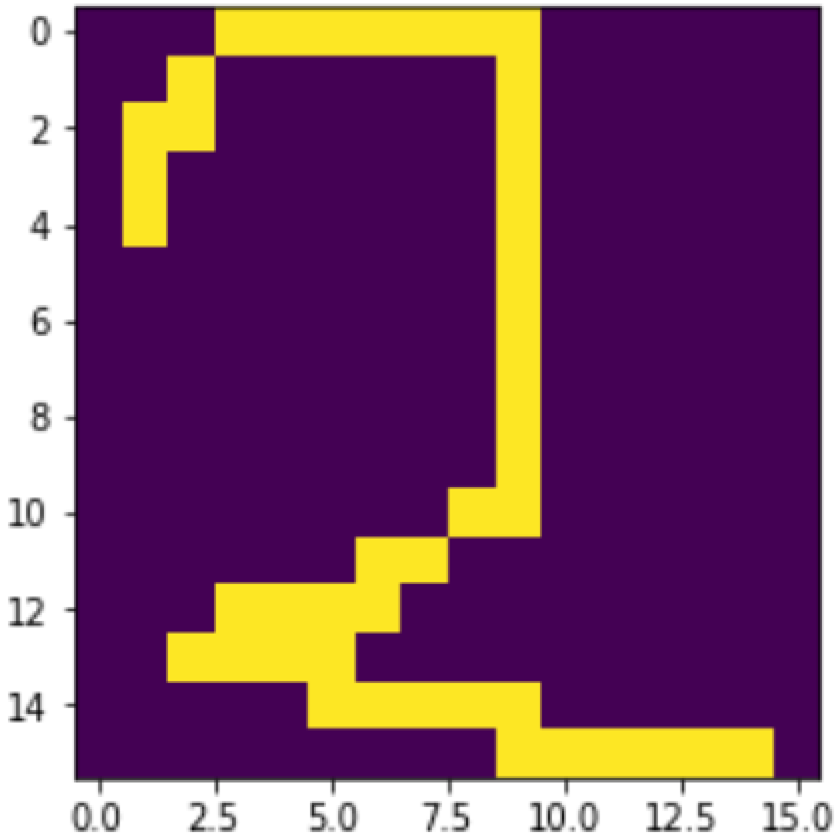
```
img = np.reshape(img2,(16,16),order='F')
```

```
1  img.shape
```

(16, 16)

```
1  plt.imshow(img)
```

<matplotlib.image.AxesImage at 0x5cc1780>



Homework

- Book 1: 2.4, 2.5, 5.1, 5.2, 5.3
- Book 2: Chapter 3
- Book 3: Chapter 3 (3.1, 3.2)