Machine Learning Applications: Classification

Alessio Benavoli

CSIS
University of Limerick

Classification

In machine learning,

> *classification is the problem of identifying to which of a set of categories (classes) a new observation belongs, on the basis of a training set of data containing observations (or instances or inputs) whose category membership is known.*

Examples are assigning a given email to the "spam" or "non-spam" class:

- Input: presence of words in the email
- Output: "spam" or "non-spam" (binary variable).

Example: assigning a diagnosis to a given patient based on observed characteristics of the patient (sex, blood pressure, presence or absence of certain symptoms, etc.).

Supervised Learning

Classification is a

Supervised Learning Technique: it means that we learn from examples

Each example is a pair consisting of

- an input object (typically a vector);
- a desired output value (class).

Example: classifying emails as “spam” or “non-spam”:

	text	class
0	Did you hear about the new "Divorce Barbie"? I...	1
1	Will u meet ur dream partner soon? Is ur caree...	1
2	Maybe i could get book out tomo then return it...	0
3	Boltblue tones for 150p Reply POLY# or MONO# e...	1
4	Your credits have been topped up for http://ww...	1
5	22 days to kick off! For Euro2004 U will be ke...	1
6	Hi I'm sue. I am 20 years old and work as a la...	1
7	08714712388 between 10am-7pm Cost 10p	1

This is called **training dataset**.

Prediction

A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

The goal is to correctly determine the class labels for unseen instances.
This operation is called a **prediction**.

This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Guessing gender from names

Consider the names “Alice” and “Bob” – most people would instantly mark Alice as a female name and Bob as a male one.

Is this the case primarily because we have seen so many examples of male Bobs and female Alices that our brains have built up a latent association between the specific name and the corresponding gender?

But some component of the name itself (its spelling / combination of letters) contributes to the gender. Is there a rule?

Is it just a toy example?

Accurate prediction of an unknown individual's gender is desirable for use in marketing.

In internet, people often tend to use "name_number" ("John167") as alias.

Guessing gender from names

First of all, we start to build a database of male and female names:

$$D = \begin{bmatrix} 'Powell' & 'male' \\ 'Pearla' & 'female' \\ 'Maude' & 'female' \\ \dots & \\ 'Renaldo' & 'male' \\ 'Tess' & 'female' \\ 'Ina' & 'female' \end{bmatrix}$$

we have collected a set of examples and let's say that the name Alice is not in our dataset.

What do we do next?

Our goal is to predict the gender of any name, even names that are not in our training dataset.

What's gender for the name "Alice"?

This is called a prediction.

Look-up table

Our classifier could be a look-up table: it is a block of data that is held in the program memory and which can be accessed by the program and used within it.

```
def predict(name):
    #search in the table
    ind = np.where(data[:,0]==name)[0]
    #return the gender
    return data[ind,1]

print(predict('Powell'))
>>'male'
```

Why is that bad?

```
print(predict('Alice'))
```

Overfitting

This is an (extreme) example of **overfitting**.

The previous algorithm performs very well in the training dataset ("print(predict('Powell'))"): it always returns the correct gender.

It performs badly on unseen data: it cannot predict unseen data.

We will see other (less extreme) examples of **overfitting**.

Using features

We know that male and female names have distinctive characteristics. In English, names ending in a, e and i are likely to be female:

Alice, Lisa, Fidelity

while names ending in k, o, r, s and t are likely to be male.

Mark, Jacques

We aim to build a classifier to model these differences and predict the gender of any name.

Training dataset

Before

$$D = \begin{bmatrix} 'Powell' & 'male' \\ 'Pearla' & 'female' \\ 'Maude' & 'female' \\ \dots & \\ 'Renaldo' & 'male' \\ 'Tess' & 'female' \\ 'Ina' & 'female' \end{bmatrix}$$

now it becomes

$$D = \begin{bmatrix} 'l' & 'male' \\ 'a' & 'female' \\ 'e' & 'female' \\ \dots & \\ 'o' & 'male' \\ 's' & 'female' \\ 'a' & 'female' \end{bmatrix}$$

Extracting last letter

```
def gender_features(word):
    return word[-1:].lower() #last_letter
gender_features('Alice')
>>'e'
```

Inputs

$$X = \begin{bmatrix} 'l' \\ 'a' \\ 'e' \\ \dots \\ 'o' \\ 's' \\ 'a' \end{bmatrix}, \qquad y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \dots \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

We denoted female as 1 and male as 0.

Encoding

Many ML algorithms use numerical inputs, therefore, we need to encode those letters into numerical value.
We will use the so called "LabelBinarizer"

$$\begin{aligned} a \rightarrow & \quad [1,0,0,0,\ldots,0] \\ b \rightarrow & \quad [0,1,0,0,\ldots,0] \\ c \rightarrow & \quad [0,0,1,0,\ldots,0] \\ \ldots\ldots & \qquad\qquad\qquad .... \end{aligned}$$

```
from sklearn import preprocessing
lb = preprocessing.LabelBinarizer()
lb.fit(X)
X2 =  lb.transform(X)
```

$$X = \begin{bmatrix} 'l' \\ 'a' \\ 'e' \\ ... \\ 'o' \\ 's' \\ 'a' \end{bmatrix}, \qquad X2 = \begin{bmatrix} 0 & 0 & 0 & ... & 0 & 0 & 0 \\ 1 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 & 0 \\ ... & & & & & & \\ 0 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 & 0 \\ 1 & 0 & 0 & ... & 0 & 0 & 0 \end{bmatrix}$$

Classifier: Multinomial Naive-Bayes

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()

#training/fitting
clf.fit(X2, y)

#prediction
xpred = lb.transform( gender_features('Alice') )
clf.predict(xpred)
>>1
```

Evaluation metrics

How do we evaluate the performance of a classifier

Accuracy

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Look-up table

Let's go back to the "look-up table" algorithm:

```
def predict(name):
    #search in the table
    ind = np.where(data[:,0]==name)[0]
    #return the gender
    return data[ind,1]

print(predict('Powell'))
>>'male'
```

What is its accuracy?

We can evaluate its accuracy on the training dataset.

```
ypred = []
for i in range(data.shape[0]):
    ypred.append(predict(data[i,0]))

Accuracy= len(np.where(ypred==data[:,1])[0])/len(ypred)
```

The result is

$$Accuracy = 1$$

Look-up table

We can now predict the gender of unseen names, if we do that for the "Look-up table" algorithm:

$$Accuracy = 0$$

This algorithm has a huge generalisation error, it is not able to predict unseen data.

This is the way we "measure" if an algorithm is really learning: applying it to unseen data (**testing dataset**).

Multinomial Naive-Bayes

What about this algorithm?

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
#training/fitting
clf.fit(X2, y)
#prediction
clf.predict(Xtest)
```

We consider a list of 7943 common names and use

Training dataset: 4765 labeled names (we know the gender)

Testing dataset 3178 unlabeled names

Note that, we have randomly split the names in those above groups.

In other words, we pretend we do not know the gender of these 3178 names and we ask the algorithm to predict it and then check its accuracy.

Accuracy for the Multinomial Naive-Bayes

Training dataset: 4765 labeled names (we know the gender)

Testing dataset 3178 unlabeled names

$$\text{accuracy training set} = 0.76$$

$$\textbf{accuracy test set} = 0.76$$

It means that the algorithm correctly guessed the gender in 76% of the instances.

Remark: we are only using the last letter of the name.

Accuracy for the Multinomial Naive-Bayes

Is 0.76 good?

What can we say about that?

We can compare this performance against "dummy classifiers"

random guesser it tosses a coin and predicts "spam" if the coin lands Head. Its accuracy is ~ 0.5.

Majority class classifier it returns the class that has more instances in the training dataset.

What is the accuracy of the majority class classifier?

Majority class classifier

In the training dataset (4765 instances), there are 3019

- 3019 female names
- 1746 male names

Why? This is true in general (it is not just our dataset), because there are more female names than male names.

The proportion is

$$\frac{3019}{4765} = 0.63$$

Therefore, assuming that is also true for the *testing dataset*, we expect that **Majority class classifier** will have an accuracy of 0.63 in the testing set.

This is in fact the case: accuracy is 0.63 in the testing set.

Multinomial Naive-Bayes

$$0.76 > 0.63$$

therefore, the Multinomial Naive-Bayes is able to extract useful information about the gender from the last letter of the name.

The Majority class classifier is always a good baseline for comparison.

Evaluation metric: Confusion matrix

Another interesting metric is the **confusion matrix**.

By definition a confusion matrix C is such C_{ij} that is equal to the number of observations known to be in class i but predicted to be in group j.

Confusion matrix test set:

$$\begin{bmatrix} 702 & 495 \\ 267 & 1714 \end{bmatrix}$$

Here 702 represents the number of male names in the testing set that were correctly predicted as male.
267 represents the number of male names in the testing set that were wrongly predicted as female.
495 represents the number of female names in the testing set that were wrongly predicted as male.
1714 represents the number of female names in the testing set that were correctly predicted as female.

Mathematical model

Let us define the probabilistic model.
Variables and domains:

$$G \in \{M, F\}$$

$$LL \in \{a, b, c, d, \ldots, z\}$$

where LL stands for last letter in the name.
We define

$$p(G = 1) = \theta$$

$$p(G = 0) = 1 - \theta$$

with 1 means Female and 0 Male. This is a Bernoulli distribution.

Mathematical model

Let's now denote with θ_{lg} the probability that LL is equal to the letter l given that the gender G is g, that is

$$p(LL = l|G = g) = \theta_{lg}$$

we have 26×2 parameters.
We are interesting in computing (Bayes' rule)

$$p(G = g|LL = l) = \frac{p(LL = l|G = g)p(G = g)}{p(LL = l)}$$

that is the posterior probability that the gender is $g \in \{0, 1\}$ when the last letter in the name is $l \in \{a, b, c, ..., z\}$.
For instance

$$P(G = 1|LL = a) = \frac{P(LL = a|G = 1)p(G = 1)}{p(LL = a)} = \frac{\theta_{a1}\theta_1}{\theta_{a0}\theta_0 + \theta_{a1}\theta_1}$$

Mathematical model

$$P(G=1|LL=a) = \frac{P(LL=a|G=1)p(G=1)}{p(LL=a)} = \frac{\theta_{a1}\theta_1}{\theta_{a0}\theta_0 + \theta_{a1}\theta_1}$$

We can estimate them from a Dataset of all Male and Female English givennames:

$$\mathcal{D} = \{(a,1),(n,0),(o,0),(a,1),\dots\}$$

we do not know the thetas. What do we do?

Maximum likelihood estimation (MLE)

MLE:

$$\arg\max_{\theta_{lg},\theta_g} \prod_{i=1}^{N} p(LL = l(i)|G = g(i))P(G = g(i)) = \arg\max_{\theta_{lg},\theta_g} \prod_{i=1}^{N} \theta_{l(i)g(i)}\theta_{g(i)}$$

The resulting classifier is called **Multinomial Naive-Bayes estimator**.

Example assume that $\mathcal{D} = \{(a,1),(n,0),(o,0),(a,1)\}$, that is N=4 observations then

$$\prod_{i=1}^{4} p(LL = l(i)|G = g(i))P(G = g(i)) = \theta_{a1}\theta_1\,\theta_{n0}\theta_0\,\theta_{o0}\theta_0\,\theta_{a1}\theta_1 = \theta_{a1}^2\theta_1^2\,\theta_{n0}\theta_0\,\theta_{o0}\theta_0$$

Note that, by summing the exponent for the same base, we make the computation of this likelihood **much faster**.

Maximum likelihood estimation (MLE) solution

The MLE estimate is

$$\theta_g = \frac{n_g}{N} \quad \textit{for } g = 0, 1$$

where $n_{g=1}$ is the number of instances (rows) in the dataset where the class variable is equal to one and N is the total number of instances.
Similarly,

$$\theta_{lg} = \frac{n_{lg}}{n_g}$$

where n_{lg} is the number of instances where the letter is l and the gender is g.

Regularisation

It may happen that $n_{lg} = 0$ and, therefore, $\theta_{lg} = 0$.

Whys is that a problem?

To avoid this problem, it is common to add a regularisation term.

$$\theta_{lg} = \frac{n_{lg} + \alpha}{n_g + \alpha\, m}$$

where m is the number of features (in our example, the number of letters). $\alpha = 1$ is called Laplace smoothing.

Regularisation

This is the way we access

$$P(G = g|LL = a)$$

```
clf.predict_proba(lb.transform(np.array([[gender_features("Anna")]])
                                        ))
>> array([[0.0162742, 0.9837258]])
```

so $P(G = 1|LL =' a') \approx 0.984$.

Exercise: verify that this probability has been computed as described in the previous slides.

Spam filter

It is the same algorithm: only the dataset changes

	text	class
0	Did you hear about the new "Divorce Barbie"? I...	1
1	Will u meet ur dream partner soon? Is ur caree...	1
2	Maybe i could get book out tomo then return it...	0
3	Boltblue tones for 150p Reply POLY# or MONO# e...	1
4	Your credits have been topped up for http://ww...	1
5	22 days to kick off! For Euro2004 U will be ke...	1
6	Hi I'm sue. I am 20 years old and work as a la...	1
7	08714712388 between 10am-7pm Cost 10p	1

We need to a bit of data-cleaning.