COMP30027 Machine Learning
The Naïve Bayes Learner

Semester 1, 2018

Jeremy Nicholson & Tim Baldwin & Karin Verspoor

THE UNIVERSITY OF MELBOURNE

Lecture Outline

1. Project 1 Intro
   - Sup. Training Phase
   - Sup. Testing Phase
   - Sup. Evaluation
   - Unsup. Train/Test
   - Unsup. Evaluation

Implementing a Naive Bayes Classifier

Naive Bayes is a supervised machine learning method:

- We need to build a model ("training phase")
- We need to make predictions using that model ("testing phase")
- We need to evaluate

Training a NB Classifier I

Our model consists of two kinds of probabilities:

- **priors** $P(c_j)$ (one per class)
- **posteriors** $P(x_i|c_j)$ (one per attribute value, per class)

Training a NB Classifier II

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

$$P(Flu) = 3/5$$
$$P(Headache = severe|Flu) = 2/3$$
$$P(Headache = mild|Flu) = 1/3$$
$$P(Headache = no|Flu) = 0/3$$
$$P(Sore = severe|Flu) = 1/3$$
$$P(Sore = mild|Flu) = 2/3$$
$$P(Sore = no|Flu) = 0/3$$
$$P(Temp = high|Flu) = 1/3$$
$$P(Temp = normal|Flu) = 2/3$$
$$P(Cough = yes|Flu) = 3/3$$
$$P(Cough = no|Flu) = 0/3$$

$$P(Cold) = 2/5$$
$$P(Headache = severe|Cold) = 0/2$$
$$P(Headache = mild|Cold) = 1/2$$
$$P(Headache = no|Cold) = 1/2$$
$$P(Sore = severe|Cold) = 1/2$$
$$P(Sore = mild|Cold) = 0/2$$
$$P(Sore = no|Cold) = 1/2$$
$$P(Temp = high|Cold) = 0/2$$
$$P(Temp = normal|Cold) = 2/2$$
$$P(Cough = yes|Cold) = 1/2$$
$$P(Cough = no|Cold) = 1/2$$

Calculating priors by counting I

There is one prior $P(c_j)$ per class: 1D array (Python list)

Cold	Flu
0	0

Calculating priors by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
0	1

Calculating priors by counting III

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	**Cold**
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
1	1

Calculating priors by counting IV

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	**Flu**
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
1	2

Calculating priors by counting V

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	**Cold**
severe	severe	normal	yes	Flu

Cold	Flu
2	2

Calculating priors by counting VI

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	**Flu**

Cold	Flu
2	3

When we need to use this, we can divide through by the sum of the entries in the list (or keep a separate counter for the total number of instances N, which is often useful).

Calculating posteriors by counting I

There is one posterior $P(x_i|c_j)$ per attribute value, per class: 2D array?

Calculating posteriors by counting II

There is one posterior $P(x_i|c_j)$ per attribute value, per class, **for each attribute** X: ~~2D array?~~ 3D array?

- But some attributes have different numbers of attribute values.
- And we might not know all of the various attribute values before we start counting.
- So...
  - 2D array of dictionaries?
  - 1D array of dictionaries of dictionaries?
  - Dictionary of dictionaries of dictionaries?

Calculating posteriors by counting III

Assuming number of classes and number of attributes is known:

Headache		Temperature
Cold: {}		Cold: {}
Flu: {}		Flu: {}

Sore		Cough
Cold: {}		Cold: {}
Flu: {}		Flu: {}

Calculating posteriors by counting IV

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache		Temperature	
Cold:	{}	Cold:	{}
Flu:	{severe:1}	Flu:	{}

Sore		Cough	
Cold:	{}	Cold:	{}
Flu:	{}	Flu:	{}

Calculating posteriors by counting V

Headache	Sore	Temperature	Cough	Diagnosis
severe	**mild**	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache		Temperature	
Cold:	{}	Cold:	{}
Flu:	{severe:1}	Flu:	{}

Sore		Cough	
Cold:	{}	Cold:	{}
Flu:	{mild:1}	Flu:	{}

Calculating posteriors by counting VI

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	**yes**	**Flu**

Headache

Cold:	{no:1, mild:1}
Flu:	{severe:2, mild:1}

Temperature

Cold:	{normal:2}
Flu:	{high:1, normal:2}

Sore

Cold:	{severe:1, no:1}
Flu:	{mild:2, severe:1}

Cough

Cold:	{yes:1, no:1}
Flu:	{yes:3}

Calculating posteriors by counting VII

We need to know the number of instances of class c_j to turn these counts into probabilities:

- The slow way: sum the entries in the corresponding dictionary
- The fast way: read off the class array

Smoothing can be done:

- When accessing values, e.g. if value is 0, replace with ϵ
- Using a `defaultdict`, e.g. default for Laplace is 1

Making predictions using a NB Classifier I

$$\hat{c} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i|c_j)$$

- These values can be read off the data structures from the training phase.
- We only care about the class corresponding to the maximal value
- However, it is often valuable to keep the whole distribution over classes (especially when we get to the unsupervised version later)

Making predictions using a NB Classifier II

We're multiplying a bunch of numbers $[0, 1]$ together — because of our floating-point number representation, we tend to get **underflow**.
One common solution is a **log-transformation**:

$$\begin{aligned} \hat{c} &= \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i|c_j) \\ &= \arg\max_{c_j \in C} [\log(P(c_j)) + \sum_i \log(P(x_i|c_j))] \end{aligned}$$

Evaluating a Supervised Classifier

- The basic idea is to compare our prediction $\hat{c}$, with the true class, of some instances.
- We can then count the number of instances where our predictions match the true class ("correct") and the number of instances where they mismatch ("incorrect").
- Various metrics or visualisations can then be determined, for example:

$$\text{Accuracy} = \frac{\text{number of correct instances}}{\text{total number of instances}}$$

An Unsupervised NB Classifier I

An **unsupervised** machine learning method does not have access to labelled training data. What does that mean in this context?

- We can't estimate $P(c_j)$, because we don't have any instances labelled with classes
- We can't estimate $P(x_i|c_j)$, because we (still) don't have any instances labelled with classes
- We can't make predictions $\hat{c}$, because we don't have the probabilities that we need to determine which class is most likely
- ... So, what will we do?

An Unsupervised NB Classifier II

Let's pretend that we had labelled instances!

- Interestingly, deterministically labelling the instances doesn't work very well. (You might like to confirm this for yourself!)
- Instead, let's label instances with random (non-uniform) **class distributions**:

Headache	Sore	Temperature	Cough	Cold	Flu
severe	mild	high	yes	0.4	0.6
no	severe	normal	yes	0.7	0.3
mild	mild	normal	yes	0.9	0.1
mild	no	normal	no	0.2	0.8
severe	severe	normal	yes	0.6	0.4

An Unsupervised NB Classifier III

Now, instead of adding 1 when we count instances, we add the corresponding (fractional) value for the class:

Headache	Sore	Temperature	Cough	Cold	Flu
severe	mild	high	yes	**0.4**	**0.6**
no	severe	normal	yes	0.7	0.3
mild	mild	normal	yes	0.9	0.1
mild	no	normal	no	0.2	0.8
severe	severe	normal	yes	0.6	0.4

Cold	Flu
0.4	0.6

An Unsupervised NB Classifier IV

Headache	Sore	Temperature	Cough	Cold	Flu
severe	mild	high	yes	**0.4**	**0.6**
no	severe	normal	yes	0.7	0.3
mild	mild	normal	yes	0.9	0.1
mild	no	normal	no	0.2	0.8
severe	severe	normal	yes	0.6	0.4

Headache

Cold:	{severe:0.4}
Flu:	{severe:0.6}

Temperature

Cold:	{high:0.4}
Flu:	{high:0.6}

Sore

Cold:	{mild:0.4}
Flu:	{mild:0.6}

Cough

Cold:	{yes:0.4}
Flu:	{yes:0.6}

An Unsupervised NB Classifier V

- Once we have our (now fractional) counts, predictions work the same way as the supervised version.
- However, now the class distributions in those predictions are (hopefully) more reliable than the random distributions that we had to begin with.
  - So, let's **iterate**, hopefully improving our estimates each time
  - Except that we will need to normalise (to ensure that they add to 1):

$$\tilde{c} = \frac{\hat{c}}{\sum_j \hat{c}_j}$$

  - We also need some way of deciding when to stop iterating. At this point, it isn't clear how to do that.

Evaluating the Unsupervised NB Classifier

- The general idea for unsupervised evaluation (in this context) can be the same as the supervised evaluation:
  - Find the most probable class for each instance
  - Compare with the actual class
  - Count the number of correct predictions
- One major difference in this case is that the classes can get "swapped":
  - For example, if every Cold instance is labelled as Flu, and every Flu instance is labelled as Cold
  - This classifier is (probably) correct; we just need to swap the labels when calculating
- On datasets of our size (esp. 2 classes), it might be easier to print the **confusion matrix** and calculate the relevant metrics by hand

Other places to get help

What if I have more questions?

- Chat with other students (about what the questions are asking, not what the answers are!)
- Post to the Discussion Forum
- My office hours
- We might talk more in the lectures next week...