COMP30027 Machine Learning
The Naïve Bayes Learner

Semester 1, 2019

Jeremy Nicholson & Tim Baldwin & Karin Verspoor

Lecture Outline

1. Theory of Naive Bayes

2. Implementation of NB
   - Training Phase
   - Testing Phase
   - Evaluation Phase

A little thought experiment... I

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
overcast	cool	high	true	no

What (do you think) is the class of `sunny`, `cool`, `normal`, `false`?

A little thought experiment... II

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
sunny	cool	normal	false	yes
sunny	mild	high	false	no
overcast	cool	high	true	no

What (do you think) is the class of `rainy`, `hot`, `normal`, `true`?

A little thought experiment... III

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
`overcast`	`mild`	`normal`	`true`	yes
`sunny`	`mild`	`normal`	`false`	yes
`overcast`	`hot`	`high`	`true`	yes
`sunny`	`cool`	`high`	`false`	yes
`rainy`	`cool`	`normal`	`true`	no
`overcast`	`hot`	`normal`	`true`	no
`sunny`	`hot`	`normal`	`false`	no
`sunny`	`mild`	`normal`	`true`	no
`rainy`	`cool`	`high`	`true`	no

What (do you think) is the class of `overcast`, `mild`, `high`, `false`?

A Probabilistic Learner I

- Let's come up with a **supervised machine learning** method
- We build a probabilistic model of the training data, and then use that to predict the class labels of the test data
- So, *given* a test instance T, which class c is most likely?
- $\hat{c} = \arg\max_{c \in C} P(c|T)$

A Probabilistic Learner II

The obvious way of doing this:

- For each class c:
  - Find the instances in the training data labelled as c
  - Count the number of times T has been observed
- Choose $\hat{c}$ with the greatest frequency of observed T

A Probabilistic Learner III

The obvious way of doing this:

- Would require an *enormous* amount of data
- A test instance T is a bundle of attribute values: to classify an (as-yet) unseen instance would require that *every possible* combination of attribute values has been attested in the training data a non-trivial number of times
- For m attributes, each taking k different values, and $|C|$ classes, this means $\mathcal{O}(|C| \cdot k^m)$ instances
    - Weather example: perhaps 100s of instances
    - 2-class problem, 20 binary attributes: at least 2M instances
    - 4 classes, 60 ternary attributes: at least 10^{28} instances
- Would only be meaningful for the instances that we've actually seen

Bayes' Rule

$$P(C, X) = P(C|X)P(X) = P(X|C)P(C)$$

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

The Naive Bayes (NB) Learner I

Task: classify an instance T according to one of the classes $c_j \in C$

$$
\begin{aligned}
\hat{c} &= \arg\max_{c_j \in C} P(c_j|T) \\
&= \arg\max_{c_j \in C} \frac{P(T|c_j)P(c_j)}{P(T)} \\
&= \arg\max_{c_j \in C} P(T|c_j)P(c_j)
\end{aligned}
$$

Intuition:

- a class generates instances
- a class will generate this particular instance with some likelihood
- we will choose more probable classes more often than less probable classes

The Naive Bayes (NB) Learner II

Task: classify an instance $T = \langle x_1, x_2, ..., x_n \rangle$ according to one of the classes $c_j \in C$

$$
\begin{aligned}
\hat{c} &= \arg\max_{c_j \in C} P(c_j | x_1, x_2, ..., x_n) \\
&= \arg\max_{c_j \in C} \frac{P(x_1, x_2, ..., x_n | c_j) P(c_j)}{P(x_1, x_2, ..., x_n)} \\
&= \arg\max_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j)
\end{aligned}
$$

The "Naive" Part

To get around this problem, we do something stupid:

$$\begin{aligned} P(x_1, x_2, ..., x_n | c_j) &\approx P(x_1|c_j)P(x_2|c_j)...P(x_n|c_j) \\ &= \prod_i P(x_i|c_j) \end{aligned}$$

This is a **conditional independence assumption**, and makes Naive Bayes a tractable method.
It is also demonstrably untrue in almost every dataset. But Naive Bayes (kinda) works anyway!

The complete Naive Bayes Learner

Task: classify an instance $T = \langle x_1, x_2, ..., x_n \rangle$ according to one of the classes $c_j \in C$

$$
\begin{aligned}
\hat{c} &= \arg\max_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \\
&= \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)
\end{aligned}
$$

Intuition:

- a class generates attribute values, according to some likelihood
- we will choose the class that is most likely to generate this particular set of values
- we will choose more probable classes more often than less probable classes

Other notable assumptions

- $P(c_j)$
  - can be estimated from the frequency of classes in the training examples **[maximum likelihood estimate]**
- the distribution of data in the training instances is (roughly) the same as the distribution in the test instances (more on this in later weeks)

Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

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

We need $P(c_j)$, $P(x_i|c_j)$: for every x_i, c_j

Naive Bayes Example 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

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

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

Naive Bayes Example III

Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccl} P(C) & \times & P(H=m|C)P(S=s|C)P(T=n|C)P(C=n|C) \\ \frac{2}{5} & \times & (\frac{1}{2})(\frac{1}{2})(\frac{2}{2})(\frac{1}{2}) = 0.05 \end{array}$$

Flu:

$$\begin{array}{ccl} P(F) & \times & P(H=m|F)P(S=s|F)P(T=n|F)P(C=n|F) \\ \frac{3}{5} & \times & (\frac{1}{3})(\frac{1}{3})(\frac{2}{3})(\frac{0}{3}) = 0 \end{array}$$

Naive Bayes Example IV

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccl} P(C) & \times & P(H=s|C)P(S=m|C)P(T=h|C)P(C=n|C) \\ \frac{2}{5} & \times & (\frac{0}{2})(\frac{0}{2})(\frac{0}{2})(\frac{1}{2}) = 0 \end{array}$$

Flu:

$$\begin{array}{ccl} P(F) & \times & P(H=s|F)P(S=m|F)P(T=h|F)P(C=n|F) \\ \frac{3}{5} & \times & (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\frac{0}{3}) = 0 \end{array}$$

Probabilistic Smoothing I

- Notice that this is a product, so if any $P(x_i|c_j) = 0$, then the final value is 0
- This is bad for two reasons:
  - To make plausible predictions, we still need to see every possible attribute value–class pair ... and so, we still require lots and lots of data
  - Unseen events mean that we discard a whole lot of otherwise useful information
- Solution: no event is impossible (every probability > 0)
- To maintain a **probability distribution**, we need to reduce the probability of seen events

Probabilistic Smoothing II

The (conceptually) simplest approach:

- If we calculate $P(x_i|c_j) = 0$, we replace zero with a trivially small non-zero number, typically called ϵ
- ϵ is a constant, which needs to be less (and preferably substantially less) than $\frac{1}{n}$, for n instances
- Effectively reduces most comparisons to the cardinality of ϵ (fewest ϵs wins)
- Assume that $(1 + \epsilon) \approx 1$, so that we don't need to do anything extra with the non-zero probabilities

Probabilistic Smoothing III

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccc} P(C) & \times & P(H=s|C)P(S=m|C)P(T=h|C)P(C=n|C) \\ \frac{2}{5} & \times & (\epsilon)(\epsilon)(\epsilon)(\frac{1}{2}) = \frac{\epsilon^3}{5} \end{array}$$

Flu:

$$\begin{array}{ccc} P(F) & \times & P(H=s|F)P(S=m|F)P(T=h|F)P(C=n|F) \\ \frac{3}{5} & \times & (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\epsilon) = \frac{4\epsilon}{45} \end{array}$$

Probabilistic Smoothing IV

Slightly more complicated:

- Unseen events get a **count** of 1
- All **counts** are increased to ensure that monotonicity is maintained (1 becomes 2, 2 becomes 3, etc.)
- Formally, for $|V|$ different attribute values for attribute X:

$$\hat{P}(x_i|c_j) = \frac{1 + \mathit{freq}(x_i, c_j)}{|V| + \mathit{freq}(c_j)}$$

Probabilistic Smoothing 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

$$P(Headache = severe|Flu) = \frac{1+2}{3+3} = 3/6$$
$$P(Headache = mild|Flu) = \frac{1+1}{3+3} = 2/6$$
$$P(Headache = no|Flu) = \frac{1+0}{3+3} = 1/6$$

$$P(Headache = severe|Cold) = \frac{1+0}{3+2} = 1/5$$
$$P(Headache = mild|Cold) = \frac{1+1}{3+2} = 2/5$$
$$P(Headache = no|Cold) = \frac{1+1}{3+2} = 2/5$$

...

Probabilistic Smoothing VI

- This method is known as **Laplace smoothing** or **add-one smoothing**
- Probabilities are changed drastically when there are few instances; with a large number of instances, the changes are small (mimics **confidence**)
- Laplace is the most common smoothing mechanism for serious NB implementations; it is easy to implement, and tends to be reasonably accurate
- It is known to systematically over–estimate the likelihood of unseen events, creating **bias** in certain circumstances

Probabilistic Smoothing VII

- Other methods include:
  - **add-k smoothing**: like Laplace, but instead of adding 1 to all counts, add $k < 1$
  - **Good-Turing estimation**: uses number of singletons to estimate number of unseen events; counts are progressively adjusted
  - **Regression**: leads to a more complicated learner...

Naive Bayes and Missing Values

- If a value is missing in a test instance, it is possible to simply ignore that feature for the purposes of classification
- If a value is missing in a training instance, it is possible to simply have it not contribute to the attribute–value counts/probability estimates for that feature

Naive Bayes, analysis I

So ... why does Naive Bayes work, given that we are making a blatantly untrue assumption?

- We don't need a correct estimate of $P(c|T)$: we only need to know which c_j is the greatest (this idea will lead us to a better learner later!)
- Relatively robust to two common types of errors:
  - We have over-estimated some $P(x_i|c_j)$, but we have under-estimated others ($\rightarrow$ tends to under-estimate overall probability)
  - Some marginally-relevant attributes are correlated ($\rightarrow$ we over-estimate our confidence, but the predicted class tends to be the same)

Naive Bayes, analysis II

Naive Bayes (NB) Classifier is very simple to build, extremely fast to make decisions, and relatively straightforward to change the probabilities when new data becomes available.

- Works well in many application areas.
- Scales easily for large number of dimensions (1000s) and data sizes.
- Moderately easy to explain the reason for the decision made.
- One should apply NB first before launching into more sophisticated classification techniques.

Lecture Outline

1. Theory of Naive Bayes

2. Implementation of NB
   - Training Phase
   - Testing Phase
   - Evaluation Phase

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 each attributes might have a different number of possible 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

Cold:	{}
Flu:	{severe:1}

Temperature

Cold:	{}
Flu:	{}

Sore

Cold:	{}
Flu:	{mild:1}

Cough

Cold:	{}
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, so as we progress through the classes, we can keep track of the greatest value so far.
- (However, sometimes it's valuable to store the entire list of calculated class scores.)

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 NB classifier

Evaluation in a supervised ML context (for NB and other methods):

- fundamentally based around comparing predicted labels with the actual labels

We'll talk about this in much more detail in the next lecture (and attribute correlation in the following one).

Places to get help with Project 1

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'll talk more in the lectures next week

Summary

- What is the Naïve Bayes algorithm?
  - What is Bayes' Rule and how is NB related?
  - What are the simplifying assumptions in the NB method?
- Probabilities in NB
  - How and why do we use smoothing in NB?
  - How can we deal with missing values in NB?
- How does one implement a NB classifier?

Further reading:

- Tan et al. (2006), Introduction to Data Mining. p. 227–238