

Lecture 15: Bayesian learning

CSI 5v93: Introduction to machine learning

Baylor University
Computer Science Department

Dr. Greg Hamerly
<http://cs.baylor.edu/~hamerly/>

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 1/21

Announcements

- Homework 4 assigned – start now!

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 2/21

Questions?

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 3/21

Bayesian learning and naive Bayes

- Bayes' rule and modelling
- naive Bayes
- smoothing
- binning continuous variables
- applications to text
- shaping probabilities

See handouts from Mitchell as primary source (also see section 6.6.3 in your book)

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 4/21

Bayes' rule

$$\Pr(M|X) = \frac{\Pr(X|M) \Pr(M)}{\Pr(X)}$$

Breaking it apart:

- M – model
- X – data
- $\Pr(M|X)$ – probability of the model given the data
- $\Pr(X|M)$ – probability of the data given the model
- $\Pr(M)$ – prior probability of the model
- $\Pr(X)$ – prior probability of the data

Naive Bayes classifier

A naive Bayes classifier is based on Bayes' rule.

Naive assumption: input variable i is independent of input variable j , *given the class*. This is called *conditional independence*.

In probability terms, if input x has features x_i and x_j :

$$\Pr(x_i, x_j|c) = \Pr(x_i|c) \Pr(x_j|c)$$

Naive Bayes probabilities

Starting again with Bayes' rule:

$$\Pr(M|X) = \frac{\Pr(X|M) \Pr(M)}{\Pr(X)}$$

and substituting our new definition:

$$\Pr(X|M) = \prod_{i=1}^d \Pr(X_i|M)$$

we get the new probability:

$$\Pr(M|X) = \frac{\Pr(M) \prod_{i=1}^d \Pr(X_i|M)}{\Pr(X)}$$

Typical naive Bayes application: discrete data

Suppose that our data is a *text document*. How would you model this?

In other words, we need a probability model for a document:

$$\Pr(X = \text{"John is a computer scientist..."}|M)$$

If we have this model $\Pr(X|M)$, then we can use Bayes' rule to do classification.

A bag-of-words model

How do we model $\Pr(X = \text{"John is a computer scientist..."}|M)$?

Imagine that we limit the vocabulary – to say, 10 words. Only these words are permitted: at computer scientist is c++ john baylor a he where.

Then using the bag-of-words model, this is a 10-dimensional problem. Each dimension can have a 1 (the word is present in the document) or 0 (the word is not present in the document).

The document “John is a computer scientist” would have the feature vector:

at	computer	scientist	is	c++	john	baylor	a	he	where
0	1	1	1	0	1	0	1	0	0

We ignore duplicate words in a document (for now).

So now we have a discrete representation, but not yet a probability model.

The bag-of-words model

For discrete data, we represent it using a non-parametric model, which means counting.

Counting probabilities:

$$\Pr(X = x) = \frac{\text{count}(X = x)}{n}$$

where $\text{count}(\alpha)$ means to count all records where α is true, and n is the total number of records considered in the training data.

So to model a document, we need to learn a probability such as above – the probability of the individual words appearing together in that document.

Modelling the joint bag-of-words probability

Document a = "John is a computer scientist"

Classes: EMAIL or SPAM

$$\Pr(X = a | \text{EMAIL}) = \frac{\text{count}(X = a \cap \text{EMAIL})}{\text{count}(\text{EMAIL})}$$

To calculate this, we would have to count all documents in the class EMAIL, and look for documents that have the words "john, is, a, computer, scientist".

Likewise for SPAM:

$$\Pr(X = a | \text{SPAM}) = \frac{\text{count}(X = a \cap \text{SPAM})}{\text{count}(\text{SPAM})}$$

Joint probabilities and lack of data

If we are looking for a document that has the words "john is a computer scientist", then we need a lot of training data.

Roughly, we need an *exponential* amount of training data for the number of words in the vocabulary.

For 10 words, we need roughly $2^{10} = 1024$ documents; for 100 words, we need roughly $2^{100} = 1,267,650,600,228,229,401,496,703,205,376$ documents.

We need *at least* this many documents *per class*!

A reasonable dictionary has thousands if not 10,000's of words.

So it becomes impossible for us to accurately model the joint probability.

Another example: joint probabilities and COD

COD = Curse Of Dimensionality

Imagine you have a discrete variable that can take k values. You want to model the variable; how many probabilities do you need to learn?

To learn each probability, how many examples do you need?

What if you have 2 discrete variables that each take k values?

What if you have d discrete variables?

How much data can you store on a computer? In 512MB memory, you can store 134,217,728 `float` or `int` numbers, or 67,108,864 `double` numbers.

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 13/21

Solution: independence assumption

If we make the naive Bayes assumption that each dimension is independent (given the class), then we reduce the amount of training data needed.

$$\begin{aligned}\Pr(X = a|\text{SPAM}) &= \frac{\text{count}(X_1 = a_1 \cap \text{SPAM})}{\text{count}(\text{SPAM})} \times \dots \times \frac{\text{count}(X_d = a_d \cap \text{SPAM})}{\text{count}(\text{SPAM})} \\ &= \prod_{i=1}^d \frac{\text{count}(X_i = a_i \cap \text{SPAM})}{\text{count}(\text{SPAM})}\end{aligned}$$

For this independence assumption, we need $O(|V|)$ documents for a vocabulary of size $|V|$; for the joint probability we need $O(2^{|V|})$

The idea is that we can model the distribution of individual words well (with little training data). However, modelling the joint probability of words is impossible – it requires too much data.

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 14/21

Using log-probabilities

If we multiply many small numbers (between 0 and 1), we will get a very small number. Using logarithms helps us fix this problem.

$$\begin{aligned}\Pr(X = a|M) &= \prod_{i=1}^d \frac{\text{count}(X_i = a_i \cap M)}{\text{count}(M)} \\ \log \Pr(X = a|M) &= \log \prod_{i=1}^d \frac{\text{count}(X_i = a_i \cap M)}{\text{count}(M)} \\ &= \sum_{i=1}^d \log \frac{\text{count}(X_i = a_i \cap M)}{\text{count}(M)} \\ &= \sum_{i=1}^d \left[\log \text{count}(X_i = a_i \cap M) - \log \text{count}(M) \right] \\ &= \left[\sum_{i=1}^d \log \text{count}(X_i = a_i \cap M) \right] - d \log \text{count}(M)\end{aligned}$$

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 15/21

Using log-probabilities

We can apply the log probabilities to the entire Bayes' rule:

$$\begin{aligned}\log \Pr(M|X) &= \log \frac{\Pr(X|M) \Pr(M)}{\Pr(X)} \\ &= \log \Pr(X|M) + \log \Pr(M) - \log \Pr(X) \\ &= \left[\sum_{i=1}^d \log \text{count}(X_i = a_i \cap M) \right] - d \log \text{count}(M) \\ &\quad + \log \Pr(M) - \log \Pr(X)\end{aligned}$$

Then the MAP classification is still the class M that gives the largest $\log \Pr(M|X)$ (similar for ML).

CSI 5v93: Introduction to machine learning, Lecture 15 – p. 16/21

Prior probabilities – $\Pr(M)$

For the tasks we will use, we will model $\Pr(M)$ as the frequency of class M .

Therefore:

$$\begin{aligned}\Pr(M) &= \frac{\text{count}(M)}{n} \\ \log \Pr(M) &= \log \text{count}(M) - \log(n)\end{aligned}$$

where n is the total number of records in the training set, and $\text{count}(M)$ is the number of times that class M occurs in the training set.

Zero probabilities

Problem: if the probability of any feature is zero, then the product of all feature probabilities is also zero.

In other words, if you see a feature value you've never seen in a class, then that will make the probability of the record being from that class be zero.

Example: modelling text documents for SPAM and EMAIL:

- both classes have training data with many examples, but only EMAIL has example with the word "water"
- if you get a new message with the word "water", it will have zero probability of being in SPAM

We don't want any feature to have zero probability; it makes the classifier brittle.

Note that $\log(0) = -\infty$ is difficult to deal with, also.

Smoothing zero probabilities

A standard way to deal with zero probabilities is to eliminate them by smoothing.

The simplest way to smooth probabilities is by adding a smoothing constant λ . For example,

$$\Pr(X = x|M) = \frac{\text{count}(X = x \cap M) + \lambda}{\text{count}(M) + k\lambda}$$

Where $\lambda > 0$ and k is the number of different values of X that have been observed.

Example of smoothing zero probabilities

We observe the following frequencies in the training set:

$$\begin{aligned}\text{count}(\text{TEMP} = \text{HIGH} \cap \text{SUNNY}) &= 4 \\ \text{count}(\text{TEMP} = \text{MED} \cap \text{SUNNY}) &= 2 \\ \text{count}(\text{TEMP} = \text{LOW} \cap \text{SUNNY}) &= 0\end{aligned}$$

Then if $\lambda = 1$

	before smoothing	after smoothing
$\Pr(\text{TEMP} = \text{HIGH} \text{SUNNY})$	$= 4/6$	$\frac{4+1}{6+3} = 5/9$
$\Pr(\text{TEMP} = \text{MED} \text{SUNNY})$	$= 2/6$	$\frac{2+1}{6+3} = 3/9$
$\Pr(\text{TEMP} = \text{LOW} \text{SUNNY})$	$= 0/6$	$\frac{0+1}{6+3} = 1/9$

2-minute journal

Please write a response to the following on a piece of paper and hand it in immediately. Please make it anonymous (no names). Write about:

- major points you learned today
- areas not understood or requiring clarification