Naïve Bayes

Discrete feature values

- Consider building an email spam filter using machine learning
- We wish to classify messages according to whether they are unsolicited commercial (spam) email, or non-spam email ➔ Automatically filter out
- Classifying emails is one example of a broader set of problems called classification
- Training set
  - ✓ A set of emails labeled as spam or non-spam
  - ✓ We will begin our construction of our spam filter by specifying the features x_i used to represent an email

Training set

- We will represent an email via a feature vector whose length is equal to the number of words in the dictionary

- If an email contains the *i*-th word of the dictionary, then we will set $x_i = 1$; otherwise, we let $x_i = 0$
  - ✓ For instance, the vector is used to represent an email that contains the words 'a' and 'buy', but not 'aardvark', 'aardwolf', or 'zygmurgy'

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{array}{c} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{array}$$

Training set

- Having chosen our feature vector, we now want to build a generative model $p(x \mid y)$
- To model $p(x \mid y)$, we will make a very strong assumption
  - ✓ We will assume that the x_i's are conditionally independent given *y*
  - ✓ This assumption is called the Naïve Bayes assumption
  - ✓ The resulting algorithm is called the Naïve Bayes classifier
- If y = 1 means spam email; 'buy' is word 2087 and 'price' is word 39831;
  - ✓ Then we are assuming that if I tell you y = 1 (that a particular piece of email is spam),
  - ✓ Then knowledge of x_{2087} (knowledge of whether 'buy' appears in the message) will have no effect on your beliefs about the value of x_{39831} (whether 'price' appears)

$$p(x_{2087} \mid y) = p(x_{2087} \mid y, x_{39831})$$

  - ✓ We are only assuming that x_{2087} and x_{39831} are conditionally independent *given y*

Training set

$$
\begin{aligned}
p(x_1, \ldots, x_{50000} \mid y) \\
&= p(x_1 \mid y)\, p(x_2 \mid y, x_1)\, p(x_3 \mid y, x_1, x_2) \cdots p(x_{50000} \mid y, x_1, \cdots, x_{49999}) \\
&= p(x_1 \mid y)\, p(x_2 \mid y)\, p(x_3 \mid y) \cdots p(x_{50000} \mid y) \\
&= \prod_{i=1}^{n} p(x_i \mid y)
\end{aligned}
$$

- The first equality simply follows from the usual properties of probabilities
- The second equality used the Naïve Bayes assumption

- Even though the Naïve Bayes assumption is an extremely strong assumptions, the resulting algorithm works well on many problems

Training model

- Our model is parameterized by

$$\phi_{i|y=1} = p(x_i = 1 \mid y = 1),\ \phi_{i|y=0} = p(x_i = 1 \mid y = 0) \text{ and } \phi_y = p(y = 1)$$

- Given a training set $\left\{\left(x^{(i)}, y^{(i)}\right); i = 1, \cdots, m\right\}$

- Write down the joint likelihood of the data

$$L\left(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}\right) = \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}\right)$$

- Maximizing this with respect to $\phi_y, \phi_{j|y=0}, \phi_{j|y=1}$ gives the maximum likelihood estimates

Training model

- The maximum likelihood estimates:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\}}$$

$$\phi_{y} = \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}{m}$$

- In the equations above, the ‘$\wedge$ ‘ symbol means ‘and’

Training model

- Having fit all these parameters, to make a prediction on a new example with features x, we then simply calculate

$$p\left(y=1 \mid x\right)=\frac{p\left(x \mid y=1\right) p(y=1)}{p(x)}$$

$$=\frac{\prod_{j=1}^{n} p\left(x_j \mid y=1\right) p(y=1)}{\prod_{j=1}^{n} p\left(x_j \mid y=1\right) p(y=1)+\prod_{j=1}^{n} p\left(x_j \mid y=0\right) p(y=0)}$$

Training model

- The Naïve Bayes classifier finds the most possible state with the largest probability as the posterior probability for y

$$
\begin{aligned}
v &= \arg\max_y p\left(y=1 \mid X\right) \\
&= \arg\max_y p\left(X \mid y=1\right) p(y=1) \\
&= \arg\max_y \prod_{j=1}^{n} p\left(x_j \mid y=1\right) p(y=1)
\end{aligned}
$$

Laplace smoothing

- The Naïve Bayes algorithm as we have described it will work fairly well for many problems, but there is a simple change that makes it work much better

- In case of work “nips”
  - ✓ We had not previously seen any emails containing the word “nips”
  - ✓ “nips” did not ever appear in your training set of spam/non-spam emails
  - ✓ Assuming that “nips” was the 35000th word in the dictionary

$$\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}} = 0$$

$$\phi_{35000|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\}} = 0$$

Laplace smoothing

- Because it has never seen "nips" before in either spam or non-spam training examples, it thinks the probability of seeing it in either type of email is zero

- Hence, when trying to decide if one of these messages containing "nips" is spam

$$p\left(y=1 \mid x\right)=\frac{\prod_{j=1}^{n} p\left(x_j \mid y=1\right) p(y=1)}{\prod_{j=1}^{n} p\left(x_j \mid y=1\right) p(y=1)+\prod_{j=1}^{n} p\left(x_j \mid y=0\right) p(y=0)}$$
$$=0$$

- Our algorithm obtains 0/0, and doesn't know how to make a prediction

Laplace smoothing

- To avoid this problem, we can use *Laplace smoothing*

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} + k}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} + k}$$

Example

- Input features: color, type, origin

No.	Color	Type	Origin	Stolen?
1	RED	Sports	Domestic	YES
2	RED	Sports	Domestic	NO
3	RED	Sports	Domestic	YES
4	YELLOW	Sports	Domestic	NO
5	YELLOW	Sports	Imported	YES
6	YELLOW	SUV	Imported	NO
7	YELLOW	SUV	Imported	YES
8	YELLOW	SUV	Domestic	NO
9	RED	SUV	Imported	NO
10	RED	Sports	Imported	YES

- We want to classify a Red Domestic SUV. Note there is no example of a Red Domestic SUV in our data set

Example

- Find the probabilities:

$$p(x_j \mid y=1) = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} + k} \quad p(x_j \mid y=0) = \frac{\sum_{i=1}^{m} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\} + 1}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} + k}$$

$$p(y=1) = \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\} + 1}{m+k} \qquad p(y=0) = \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} + 1}{m+k}$$

$$p(RED \mid Yes) = 0.57, \quad p(RED \mid No) = 0.43$$

$$p(SUV \mid Yes) = 0.29, \quad p(SUV \mid No) = 0.57$$

$$p(Domestic \mid Yes) = 0.43, \quad p(Domestic \mid No) = 0.57$$

Example

- Since 0.070 > 0.036, the example gets classified as "No"

$$v_1 = p(Yes) \cdot p(RED \mid Yes) \cdot p(SUV \mid Yes) \cdot p(Domestic \mid Yes) = 0.036$$
$$v_2 = p(No) \cdot p(RED \mid No) \cdot p(SUV \mid No) \cdot p(Domestic \mid No) = 0.070$$

END