Lecture #7: Naïve Bayes

Probabilistic Classifier Learning Approaches

- To learn a probabilistic classifier, there are two types of approaches
- **Generative:**
  - Learn $P(y)$ and $P(\mathbf{x}|y)$
  - Compute $P(y|\mathbf{x})$ using Bayes rule

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y)P(y)}{\sum_y P(\mathbf{x}, y)}$$

- **Discriminative:**
  - Learn $P(y|\mathbf{x})$ directly
  - Logistic regression is one of such techniques

Bayes Classifier

- Generative model learns $p(y)$ and $p(\mathbf{x}|y)$

- Prediction is made by
$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{\sum_y p(\mathbf{x}, y)}$$

- Often referred to as the Bayes Classifier due to using the Bayes rule

Joint Density Estimation

- More generally, learning $p(\mathbf{x}|y)$ is a density estimation problem, which is a challenging task
- Now we will consider the case where $\mathbf{x}$ is a d-dimensional binary vector
- How to learn $P(\mathbf{x}|y)$ in this case?

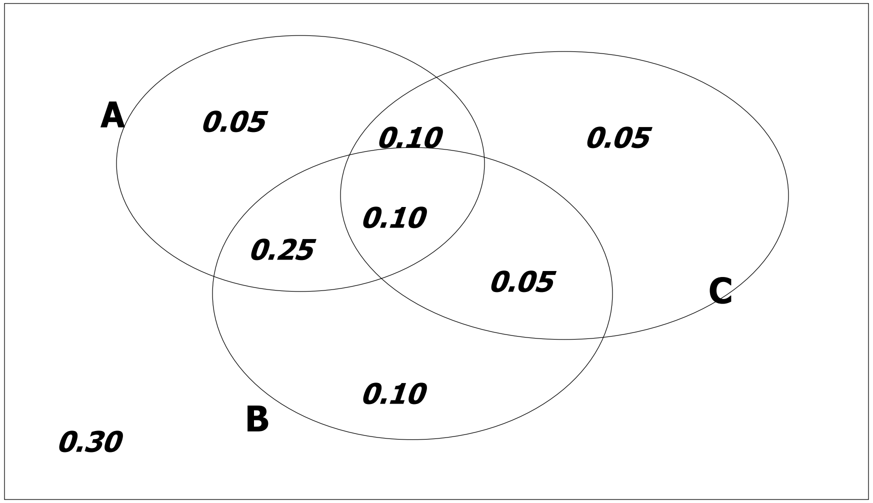
The Joint Distribution

Example: Boolean variables A, B, C

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values of your variables (M Boolean variables $\Rightarrow 2^M$ rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



Learning a joint distribution

Build a JD table in which the probabilities are unspecified

A	**B**	**C**	**Prob**
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Then fill in each row with

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	**B**	**C**	**Prob**
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	**0.25**
1	1	1	0.10

Fraction of all records in which A and B are True but C is False

Example of Learning a Joint

- This Joint was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

UCI machine learning repository: http://www.ics.uci.edu/~mlearn/MLRepository.html

Learning Joint Distribution and Overfitting

- Let $\mathbf{x}$ be a d-dimensional binary vector, and $y \in \{1,2,\ldots,k\}$

- Learning the joint distribution $\mathrm{P}(\mathbf{x}|y=i)$ for $i = 1,\ldots,k$ involves estimating $k \times \left(2^d - 1\right)$ parameters
  - For large d, this number is prohibitively large
  - Not enough data to estimate the joint distribution accurately
  - Common to encounter the situation where no training examples have the exact $\mathbf{x} = [u_1,\ldots,u_d]^T$ value combination
  - Then $P(\mathbf{x} = [u_1,\ldots,u_d]^T | y = i) = 0$ for all values of i
  - This will lead to severe overfitting

Naïve Bayes Assumption

- **Assumption:** each feature is independent from one another given the class label

- **Definition:** x is **conditionally independent** of y given z, if the probability distribution governing x is independent of the value of y, given the value of z

$$\forall i, j, k \; P(x = i | y = j, z = k) = P(x = i | z = k)$$

Often denoted as $p(x|y, z) = p(x|z)$

- **Example:**

$$p(thunder|raining, lightening) \\ = p(thunder|lightening)$$

Conditional Independence vs. Independence

- **Conditional Independence:**

$$p(x, y|z) = p(x|z)p(y|z)$$

or equivalently: $p(x|y, z) = p(x|z)$

- **Independence:**

$$p(x, y) = p(x)(y)$$

or equivalently: $p(x|y) = p(x)$

- Conditional independence $\neq$ independence

Naïve Bayes Classifier

- Under Naïve Bayes assumption, we have:

$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y)$$

- No need to estimate the joint distribution

- We only need to estimate $p(x_i|y)$ for each feature i

- **Example:** with d binary features and k classes, we reduce the number of parameters from $k\left(2^d - 1\right)$ to kd
  - Significantly reduces overfitting

Example: Spam Filtering

- Bag-of-words representation to describe emails
- Represent an email by a vector whose dimension = the number of words in our "dictionary"
- Example: Bernoulli feature
  - $x_i = 1$ if the i-th word is present
  - $x_i = 0$ if the i-th word is not present

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

- The ordering/position of the words does not matter
- "Dictionary" can be formed by looking through the training set and identifying all the words & tokens that have appeared at least once (with stop-words like "the", "and" removed)

MLE for Naïve Bayes with Bernoulli Model

- Suppose our training set contains N emails, maximum likelihood estimate of the parameters are:

$$P(y=1)=\frac{N_1}{N}, \text{ where } N_1 \text{ is the number of spam emails}$$

$$P(x_i=1\,|\,y=1)=\frac{N_{i|1}}{N_1},$$

i.e., the fraction of spam emails where x_i appeared

$$P(x_i=1\,|\,y=0)=\frac{N_{i|0}}{N_0}$$

i.e., the fraction of nonspam emails where x_i appeared

Naïve Bayes Prediction

- To make a prediction for a new example with feature $\mathbf{x} = [u_1, \ldots, u_d]^T$

$$P(y = 1|\mathbf{x})$$
$$= \frac{P(y = 1) \prod_{i=1}^{d} P(x_i = u_i | y = 1)}{\sum_{y' \in \{0,1\}} P(y = y') \prod_{i=1}^{d} P(x_i = u_i | y = y')}$$
$$\propto P(y = 1) \prod_{i=1}^{d} P(x_i = u_i | y = 1)$$

Discrete and Continuous Features

- Naïve Bayes can be easily extended to handle features that are not binary-valued
- **Discrete:** $x_i \in \{1,2,\dots,k_i\}$
  - $P(x_i = j|y)$ for $j \in \{1,2,\dots,k_i\}$ - categorical distribution in place of Bernoulli
- **Continuous:** $x_i \in R$
  - Discretize the feature, then build categorical distribution for each feature
  - When the feature does not follow Gaussian, this can result in a better classifier

Problem with MLE

- Suppose you picked up a new word “Mahalanobis” in your class and started using it in your email $\mathbf{x}$
- Because “Mahalanobis” (say it’s the $n+1$ th word in the vocabulary) has never appeared in any of the training emails, the probability estimate for this word will be $P(x_{n+1}=1|y=1)=P(x_{n+1}=1|y=0)=0$
- Now $P(\mathbf{x}|y)=\prod_i P(x_i|y)=0$ for both $y=0$ and $y=1$
- Given limited training data, MLE can often result in probabilities of 0 or 1. Such extreme probabilities are “too strong” and cause problems

Laplace Smoothing

- Suppose we estimate a probability P(z) and we have n_0 examples where $z = 0$ and n_1 examples where $z = 1$ MLE estimate is

$$P(z = 1) = \frac{n_1}{n_0 + n_1}$$

- **Laplace Smoothing:** Add 1 to the numerator and 2 to the denominator

$$P(z=1) = \frac{n_1 + 1}{n_0 + n_1 + 2}$$

If we don't observe any examples, we expect $P(z=1) = 0.5$, but our belief is weak (equivalent to seeing one example of each outcome).

MAP for Naïve Bayes Spam Filter

- When estimating $p(x_i|y=1)$ and $p(x_i|y=0)$
  - Bernoulli case:

$$\underbrace{P(x_i = 1 \mid y = 0)}_{\textbf{MLE}} = \frac{N_{i|0}}{N_0} \Rightarrow \underbrace{P(x_i = 1 \mid y = 0)}_{\textbf{MAP}} = \frac{N_{i|0} + 1}{N_0 + 2}$$

- When encounter a new word that has not appeared in training set, now the probabilities do not go to zero
- This is called Laplace Smoothing

Naïve Bayes Summary

- **Generative classifier**
  - learn $P(\mathbf{x}|y)$ and $P(y)$
  - Use Bayes rule to compute $P(y|\mathbf{x})$ for classification
- Assumes conditional independence between features given class labels
  - Greatly reduces the numbers of parameters to learn
- MAP estimation (or Laplace smoothing) is necessary to avoid overfitting and extreme probability values
- In practice, a fast and solid baseline for text classification