Generative VS. Discriminative

Compare LDA and Logistic Regression

- Generative method vs Discriminative method
  - Discriminative methods model *P*(*y* | **x**) directly
  - Generative methods model *P*(**x** | *y*) (and P(*y*))
- Under LDA model we can show

$$P(y=1 \mid \mathbf{x}; p, \Sigma, \mu_0, \mu_1) = \frac{1}{1+\exp(-\theta^T \mathbf{x})},$$

where θ is some function of $p, \Sigma, \mu_0,$ and μ_1

  the same form used by logistic regression

- This indicates
  - If *P*(**x** | y) is a multivariate Gaussian distribution, *P*(y | **x**) follows a logistic function
  - But the converse is not true
- LDA makes stronger modeling assumptions

Comparing Perceptron, Logistic Regression, and LDA

- They all learn linear decision boundaries
- How should we choose among these three algorithms?
- There is a big debate within the machine learning community!
  - Computational efficiency
  - Statistical efficiency
  - Robustness to model assumptions
  - Robustness to missing features and noise/outliers

Issues in the Debate

- Statistical Efficiency.
  - If the generative model P($\mathbf{x}$,y) is correct, LDA usually performs the best, particularly when the amount of training data is small.
  - In theory, if the model is correct, LDA requires 30% less data than Logistic Regression.
- Computational Efficiency.
  - Generative models typically are the easiest to learn.
  - LDA can be computed directly from the data without using search algorithm.

Issues in the Debate

- Robustness to model assumptions
  - LDA makes the strongest assumptions --- tend to perform poorly when violated, e.g., if $P(\mathbf{x} \mid y)$ is non-gaussian
  - Logistic Regression and Perceptron are more robust
- Robustness to missing values and noise
  - In many applications, some of the features may be missing or corrupted in some training examples.
  - Generative models typically provide better ways of handling missing values than discriminative models.
  - Noise can mislead generative models
  - Discriminative models are less sensitive to noise as long as they are not close to decision boundary

Generative Model for Discrete Inputs: Naïve Bayes

- LDA: generative model for continuous inputs
- How about discrete inputs?
  - The Naïve Bayes Classifier

Example: Spam Filter

- The naïve Bayes classifier is widely used for text data (hence this example)
- We want to classify email messages into the spam and non-spam categories
- Our training set is a set of emails that has been classified manually into the two categories
- First question: how do we represent an email using a feature vector **x** – what features should we use?

Bag-of-Words Representation for Text Classification

- First we decide a vocabulary
  - The dictionary? Too big, not necessary
  - All words and tokens used in the training set
- Represent an email by a vector whose dimension is the number of words in our vocabulary

- x_i=1 if the *i*th word is present
- x_i=0 if the *i*th word is not present

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

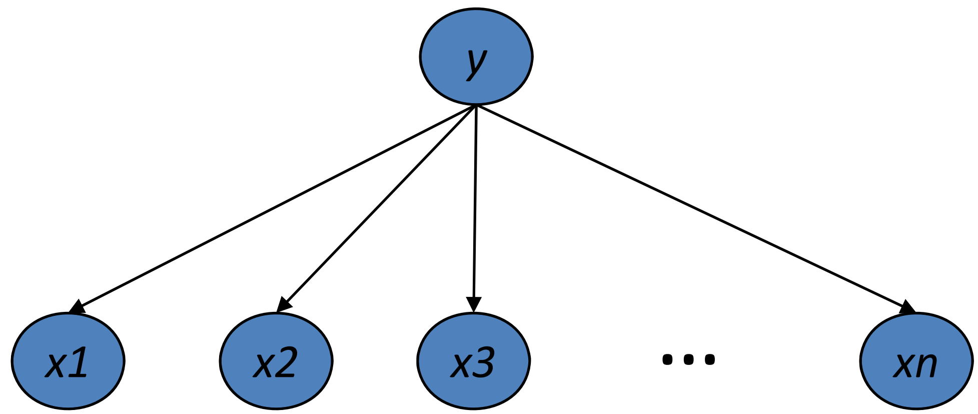
A Bayes Classifier

- To learn a bayes classifier, we need to model P(**x**|*y*) and P(*y*)
- If our vocabulary has n words, there are 2^n possible values for **x**
- If we model P(**x**|*y*) explicitly as a multinomial distribution over all possible values of **x**, we need to learn $2^*(2^n – 1)$ parameters
- To avoid such problem, we can assume that x_i*'s* are conditionally independent given y, i.e.,

$$P(x_1, x_2, ..., x_n \mid y) = \prod_{i=1}^{n} P(x_i \mid y)$$

- This is called the **Naïve Bayes assumption**
- The number of parameters for P(x|y) is now 2*n (Why?)

Naïve Bayes Classifier



- A generative model – an email is generated as follows:
  - Determine if it is a spam or not according to P(y) (Bernoulli)
  - Determine if each word x_i in the vocabulary is contained in the message *independently* according to P(x_i | y) (Bernoulli)
- For this model, we need to learn:
  - For y: P($y=1$)
  - For x_i: P($x_i =1 | y = 1$) and P($x_i =1 | y = 0$) “class conditional probability” for i=1,…,n

MLE for Naïve Bayes

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

$$P(y=1)=\frac{N_1}{N}$$, where N_1 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 the nonspam emails where x_i appeared

What if x_i is Multinomial?

- If x_i is discrete with more than two possible values $\{v_1, \ldots, v_m\}$, $P(x_i|y)$ can be described by a conditional probability table

	$y = 0$	$y=1$
$x_i=v_1$	$P(x_i=v_1 \mid y = 0)$	$P(x_i=v_1 \mid y = 1)$
$x_i=v_2$	$P(x_i=v_2 \mid y = 0)$	$P(x_i=v_2 \mid y = 1)$
…	…	…
$x_i=v_m$	$P(x_i=v_m \mid y = 0)$	$P(x_i=v_m \mid y = 1)$

- Really only needs m-1 rows since rows sum to 1
- In multi-class cases, we just need to add more columns to the above table.

$$P(x_i = v_j \mid y = k) = \frac{N_{ij|k}}{N_k}$$

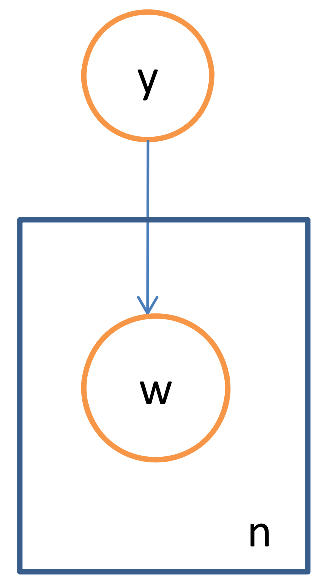
i.e., the fraction of class k examples where x_i took value v_j

Multinomial model for bag-of-words

- An alternative to the binary formulation of the bag-of-words
- Generate each word in the document as an independent categorical variable
- # category = size of the dictionary

e.g., $P_{mle}(w=\text{“apple”}|y=1) = \dfrac{\text{\# of times “apple” appears in positive documents}}{\text{\# of total words in positive documents}}$

- Considers the word counts rather than just present/absent
- Typically performs better than binomial model



The numbered plate indicates that the random variable w is sample n times, independently according to p(w|y)

Problem with MLE

- Many words are rare, particularly when considering a particular class
  - The probability estimates for such words can be poor for such words, even with a reasonably large dataset
- Consider the spam example:
  - Suppose in our training set "Mahalanobis" appears in a non-spam mail and never appears in a spam mail
  - Suppose also that "XXX" appears in a spam message but no non-spam messages
  - Now suppose we get a new message **x** that contains both words
- We will have that P(**x**|*y*) = $\prod_i P(x_i \mid y) = 0$ for both *y=0* and *y=1*
  - Because P("Mahalanobis" | **y**=1) = 0 and P("XXX" | **y**=0) = 0
- Given limited training data, MLE can result in probabilities of 0 or 1. Such extreme probabilities are "too strong" and cause problems.
  - Use Laplace smoothing to help correct this

Laplace Smoothing

- Suppose we estimate a probability P(z) and we have n_0 examples where z is false and n_1 examples where z is true. Our MLE estimate is $$P(z=1) = \frac{n_1}{n_0 + n_1}$$
- Laplace Estimate. 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).

  As n_0 and n_1 get large converges to MLE
- If z has K different outcomes, then we estimate it as

$$P(z=k) = \frac{n_k + 1}{n + K}$$

Learning and Predicting with Naïve Bayes Classifiers

- Learning
  - Need to estimate the following probability distributions (via counting)

$p(y)$ Prior distribution of y

$p(x_i \mid y)$ Class conditional distribution of x_i

- Predicting
  - Given **x** =(x_1, x_2, …, x_d), compute p(y|**x**)

$$p(y \mid \mathbf{x}) = \frac{p(y)\, p(\mathbf{x} \mid y)}{p(\mathbf{x})} \propto p(y) \prod_i p(x_i \mid y)$$

  - Apply decision theory to make final prediction of *y*

Discrete Naïve Bayes learns a Linear Decision Boundary

- For binary feature spaces Naïve Bayes gives a linear decision boundary

$$P(\mathrm{x}|Y=y) = P(x_1 = v_1|Y=y) \cdot P(x_2 = v_2|Y=y) \cdots P(x_n = v_n|Y=y)$$

- Define a discriminant function for class 1 versus class 0

$$h(\mathrm{x}) = \frac{P(Y=1|\mathrm{X})}{P(Y=0|\mathrm{X})} = \frac{P(x_1 = v_1|Y=1)}{P(x_1 = v_1|Y=0)} \cdots \frac{P(x_n = v_n|Y=1)}{P(x_n = v_n|Y=0)} \cdot \frac{P(Y=1)}{P(Y=0)}$$

Log of Odds Ratio

$$\frac{P(y=1|\mathrm{x})}{P(y=0|\mathrm{x})} = \frac{P(x_1=v_1|y=1)}{P(x_1=v_1|y=0)} \cdots \frac{P(x_n=v_n|y=1)}{P(x_n=v_n|y=0)} \cdot \frac{P(y=1)}{P(y=0)}$$

$$\log\frac{P(y=1|\mathrm{x})}{P(y=0|\mathrm{x})} = \log\frac{P(x_1=v_1|y=1)}{P(x_1=v_1|y=0)} + \ldots \log\frac{P(x_n=v_n|y=1)}{P(x_n=v_n|y=0)} + \log\frac{P(y=1)}{P(y=0)}$$

Suppose each x_j is binary and define

$$\alpha_{j,0} = \log\frac{P(x_j=0|y=1)}{P(x_j=0|y=0)}$$

$$\alpha_{j,1} = \log\frac{P(x_j=1|y=1)}{P(x_j=1|y=0)}$$

Log Odds (2)

- Now rewrite as

$$\log \frac{P(y=1|\mathrm{x})}{P(y=0|\mathrm{x})} \;=\; \sum_j \alpha_{j,1} \cdot x_j + \alpha_{j,0} \cdot (1 - x_j) + \log \frac{P(y=1)}{P(y=0)}$$

$$\log \frac{P(y=1|\mathrm{x})}{P(y=0|\mathrm{x})} \;=\; \sum_j (\alpha_{j,1} - \alpha_{j,0}) x_j + \left(\sum_j \alpha_{j,0} + \log \frac{P(y=1)}{P(y=0)} \right)$$

- We classify into class 1 if this is ≥ 0 and into class 0 otherwise

- For arbitrary multinomial features the boundary is linear in a binary one-vs-all encoding of the features
- For numeric features the Gaussian naïve Bayes classifier does not give a linear boundary

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
  - Referred to as the ***Naïve assumption***
- Batch learning but can be easily turned into online learning
  - Just incrementally update the various probability estimates
- Often works surprisingly well and a good “first thing” to try