CS6375: Machine Learning

Gautam Kunapuli

Naïve Bayes

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Generative vs. Discriminative Methods

Discriminative methods model $y = f(x)$ or $P(y|x)$ directly; e.g.,*SVMs, Decision Trees, Logistic Regression.*
Generative methods model the distributions $P(x|y)$ and $P(y)$, or the joint distribution $P(x, y)$; e.g., *Naïve Bayes.*

Generative Methods

- create distributions that can **generate examples**
  - can create **complete** input feature vectors
  - describes probability distributions for <u>all</u> features
  - Stochastically create a plausible feature vector
  - Example: Bayes net

Example: describe the class of birds

  - Probably has feathers, lays eggs, flies, etc
- Make a model that *generates* positive examples
- Make a model that *generates* negative examples
- Classify a test example based on which model is **more likely to generate it**

Discriminative Methods

- create functions or distributions that can differentiate between examples
  - don’t try to model all the features, instead focus on the task of categorizing
  - captures differences between categories

Example: what differentiates birds and mammals?

  - birds lay eggs, but mammals don’t (let’s ignore monotremes for now)
- Make a model that discriminates between positive and negative examples
- Classify a test example based on **features that successfully aid in discriminating it**

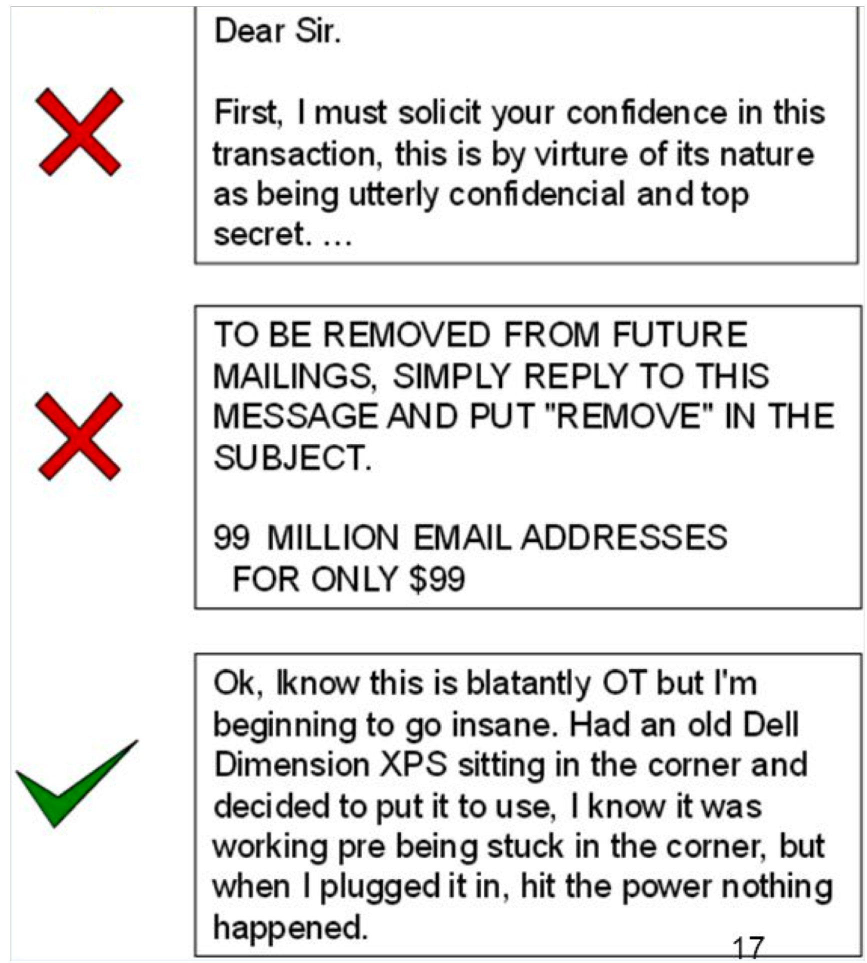
Example: Spam Filtering

Example: Develop a model to **classify if a new e-mail is spam or not.** This is a supervised classification problem where the features ($\boldsymbol{x}$) are e-mail **bag-of-words representation** of spam keywords.

To learn a **generative classifier**, we need to model $P(\boldsymbol{x}|y)$ and $P(y)$

- if our vocabulary has d words (or **generally, binary features**), there are 2^d possible inputs (*in the spam example,* 2^d *types of email bag-of-words combinations*)
- if we model 2^d explicitly as a multinomial distribution over all possible values of $\boldsymbol{x}$, we will need to learn $2 \cdot (2^d - 1)$ parameters!

This is clearly not feasible.



Example: Spam Filtering

Example: Develop a model to **classify if a new e-mail is spam or not.** This is a supervised classification problem where the features (x) are e-mail **bag-of-words representation** of spam keywords.

To learn a **generative classifier**, we need to model $P(\boldsymbol{x}|y)$ and $P(y)$

- To avoid combinatorial complexity, we can **assume that the features are conditionally independent given the labels** y that is,

$$P(x_1, x_2, \dots, x_d|y) = P(x_1|y) \cdot P(x_2|y) \cdot \dots \cdot P(x_d|y)$$
$$= \prod_{j=1}^{d} P(x_j|y)$$

This is called the **Naïve Bayes assumption**.

For the spam filtering example, this is equivalent to assuming that spam words are all completely independent of each other

- The number of parameters is now $2d$ (why?)
- *This assumption avoids* estimating probability of **compound features** (e.g., $x_1 \wedge x_2$)

The **naïve Bayes assumption is often violated**, yet it **performs surprisingly well** in many practical situations!
- *Plausible reason: only need the probability of the correct class to be the largest*!
- *Example*: in binary classification; just need to figure out the correct side of 0.5 and not the actual probability (0.51 is the same as 0.99).

The Bayes Rule

The **Bayes Rule** is a fundamental result in probability and widely used in generative models:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

probability of features given the label

prior probability of the label

probability of the data

Consider the simple problem of predicting if an email is spam or not based on whether the phrase (a single feature) “Nigerian prince” (denoted np for short) occurs in the text of the e-mail

$$P(spam = 1|np = 1) = \frac{P(np = 1|spam = 1) \cdot P(spam = 1)}{P(np = 1)}$$

$$= \frac{P(np = 1|spam = 1) \cdot P(spam = 1)}{P(np = 1|spam = 1) \cdot P(spam = 1) + P(np = 1|spam = 0) \cdot P(spam = 0)}$$

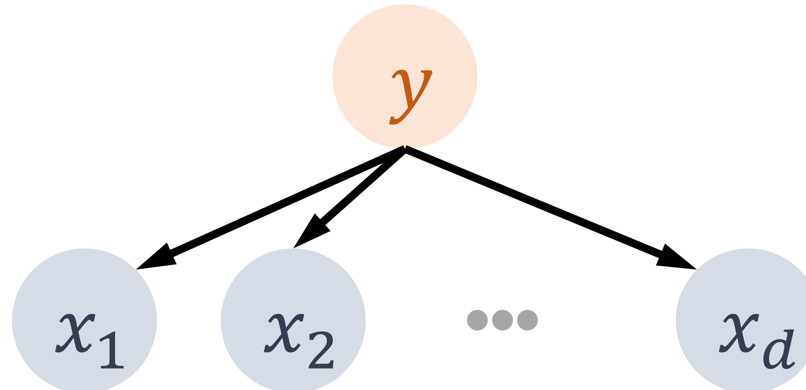
the denominator is the total probability of seeing the phrase “Nigerian prince” irrespective of the context

probability of seeing the phrase “Nigerian prince” in ***spam***

probability of seeing the phrase “Nigerian prince” in ***non-spam***

The Naïve Bayes Classifier

Example: Develop a model to classify if a new e-mail is spam or not. This is a supervised classification problem where the features (x) are e-mail **bag-of-words representation** of spam keywords.



Naïve Bayes assumption: features are conditionally independent given y that is,

$$P(x_1, x_2, \dots, x_d | y) = \prod_{j=1}^{d} P(x_j | y)$$

A **generative model**: an e-mail is **generated** as follows

- y: determine if an e-mail is spam or not according to $\mathrm{P}(y)$ (Bernoulli distribution)
- x_i: determine if each word x_i in the vocabular is contained in the message independently of all other words according to $\mathrm{P}(x_i|y)$ (another Bernoulli distribution)

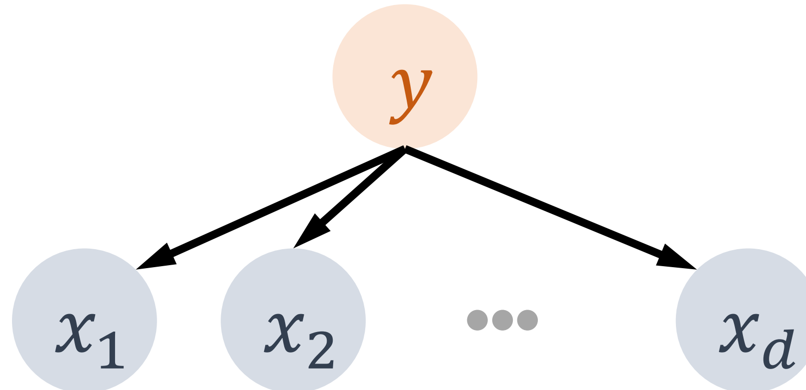
For this model we need to learn:

- for the labels y, $\mathrm{P}(y = 1)$ *(probability that the e-mail **is spam**)*
- for the features x_j, $\mathrm{P}(x_j = 1|y = 1)$ *(probability of seeing word x_j when the e-mail **is spam**)* $\mathrm{P}(x_j = 1|y = 0)$ *(probability of seeing word x_j when the the e-mail **is not spam**)*

Maximum Likelihood Estimation: Learning

For this model we need to learn:

- for the labels y, $\mathrm{P}(y = 1)$ *(probability that the e-mail* ***is spam****)*
- for the features x_j, $\mathrm{P}(x_j = 1|y = 1)$ *(probability of seeing word x_j when the e-mail* ***is spam****)* $\mathrm{P}(x_j = 1|y = 0)$ *(probability of seeing word x_j when the the e-mail* ***is not spam****)*



number of ***spam*** *e-mails*

$\#(y=1)$	$\#(y=0)$
113	679

number of ***non-spam*** *emails*

We can estimate the probabilities $\mathrm{P}(y = 1)$ by simply counting the number of spam e-mails (positive examples); we also have $\mathrm{P}(y = 0) = 1 - \mathrm{P}(y = 1)$.

number of times the feature "please read" occurred in ***spam*** *emails*

each feature takes only two values, so we only estimate $\mathrm{P}(x_i|y)$ for $x_j = 1$.

	$\#(x_j = 1\|y = 1)$	$\#(x_j = 1\|y = 0)$
please read	12	17
credit debt	22	2
order now	21	0
spam phrase d	3	36

number of times the feature "please read" occurred in ***non-spam*** *emails*

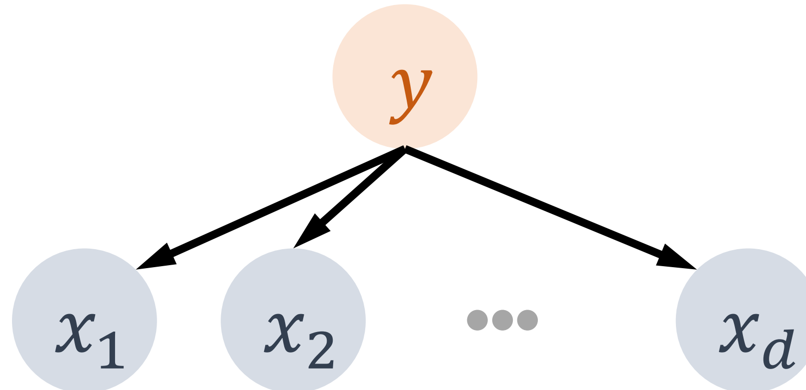
one row for each feature

We can estimate the probabilities $\mathrm{P}(x_j = 1|y = 1)$ and $\mathrm{P}(x_j = 1|y = 0)$ by simply counting all the occurrences of each spam keyword (feature) in the data set for spam and non-spam emails (for all label values).

Maximum Likelihood Estimation: Classification

For this model we need to learn:

- for the labels y, $\mathrm{P}(y = 1)$ *(probability that the e-mail **is spam**)*
- for the features x_j, $\mathrm{P}(x_j = 1|y = 1)$ *(probability of seeing word x_j when the e-mail **is spam**)* $\mathrm{P}(x_j = 1|y = 0)$ *(probability of seeing word x_j when the the e-mail **is not spam**)*



$\mathrm{P}(y=1)$	$\mathrm{P}(y=0)$
.14	.86

$$y = \underset{y}{\operatorname{argmax}}\, P(y|\boldsymbol{x}) = \underset{y}{\operatorname{argmax}}\, P(y) \cdot \prod_{j=1}^{d} P(x_j|y)$$

Predict the label probability of new examples using Bayes rule; we don't need to consider $P(\boldsymbol{x})$ here (why?)

for each, each feature = 1, we need probabilities corresponding to each label (in this case, $y = 0$ and $y = 1$)

	$\mathrm{P}(x_i = 1|y = 1)$	$\mathrm{P}(x_i = 1|y = 0)$
$x_i = 1$	$\frac{22}{144}$	$\frac{36}{41}$

	$y = 1$	$y = 0$
$x_i = 1$	22	36
$x_i = 0$	122	5

convert all conditional counts to probabilities*;* *note that **for each feature** there are **four cases** and we are interested in two quantities:* $\mathrm{P}(x_i = 1|y = 1)$ *and* $\mathrm{P}(x_i = 1|y = 0)$ *from the count table.*

Maximum Likelihood Estimation: Extensions

Binary features, binary classification: Until now, we only considered binary features: $x_i = 0$ or $x_i = 1$, and binary classification problems: $x_i = 0$ or $x_i = 1$.

	$P(x_i = 1 \mid y = 1)$	$P(x_i = 1 \mid y = 0)$
$x_i = 1$	$\frac{22}{144}$	$\frac{36}{41}$

Multivariate features: however each feature x_i can take k values (compare with decision trees), that is, we can have $x_j = v_1, x_j = v_2, \ldots, x_j = v_k$ for different training examples.

Easily handled by adding more rows to account for new feature values.

	$P(x_1 \mid y = 1)$	$P(x_1 \mid y = 0)$
$x_1 = v_1$	.2	.3
$x_1 = v_2$	.1	.1
$x_1 = v_k$	.3	.6

each feature takes k values, so we now need k rows per feature.

actually, we can get away with $k-1$ rows per feature (why?); in the binary feature case, we used only 1 row per feature

all columns must sum to 1

Multiple labels: labels can take C values in **multi-class** classification problems.

Easily handled by adding more columns to account for more labels.

	$P(x_j \mid y = 0)$	$P(x_j \mid y = 1)$	$P(x_j \mid y = 2)$	$P(x_j \mid y = C)$
$x_1 = 1$	.2	.1	.1	.5
$x_2 = 1$	.4	.0	.2	.0
$x_d = 1$	.1	.7	.0	.6

Naïve Bayes: Algorithm

Train Naïve Bayes (training examples)

for each possible class label $c = 1, \ldots, C$

- count the number of training examples with label c : $\#(y = c)$
- **for each** feature $j = 1, \ldots, d$
  - **for each** possible value of feature x_j, $k = 1, \ldots, K$
    - count the number of training ex. with j-th feature v_k and label c: $\#(x_j = v_k | y = c)$

convert the counts to **multinomial distributions** (whose rows sum to 1)

$$P(y = c) = \frac{\#(y = c)}{\sum_{\ell=1}^{C} \#(y = \ell)}$$

$$P\left(x_j = v_k | y = c\right) = \frac{\#(x_j = v_k | y = c)}{\sum_{m=1}^{K} \#(x_j = v_m | y = c)}$$

Classify Naïve Bayes (test example)

use Bayes rule

$$y_{test} = \underset{c=1,\ldots,C}{\operatorname{argmax}} P(c) \cdot \prod_{j=1}^{d} P\left(x_j^{test} | c\right)$$

MLE: Limitations

Consider the case where:

- the word “Mahalanobis” was seen in a **non-spam** e-mail exactly once in the training data
- the word “Viagra” was seen in a **spam** e-mail exactly once in the training data

	$P(“Maha” \mid spam = 1)$	$P(“Maha” \mid spam = 0)$
Mahalanobis	*.0*	*1.*
Viagra	*1.*	*.0*
	$P(“Viagra” \mid spam = 1)$	$P(“Viagra” \mid spam = 0)$

Given limited training data for certain feature-label combinations, their probabilities become either 0.0 or 1.0, that is, they take **extreme values**; such probabilities are too strong and cause problems:

$$P(”Viagra”, “Maha” | \mathbf{spam = 1}) = P(“Viagra” | \mathbf{spam = 1}) P(“Maha” | \mathbf{spam = 1})$$
$$P(”Viagra”, “Maha” | \mathbf{spam = 0}) = P(“Viagra” | \mathbf{spam = 0}) P(“Maha” | \mathbf{spam = 0})$$

This is because the **probability estimates from MLE** can be very **poor** even in large data sets, as we use feature-label combinations.

MLE: Laplacian Smoothing

For a binary feature, MLE estimate is given by:

$$P(z=1) = \frac{\#(z=1)}{\#(z=0) + \#(z=1)}$$

To avoid extreme values, perform Laplacian smoothing by adding 1 to the numerator and 2 to the denominator:

$$P(z=1) = \frac{\#(z=1) + \mathbf{1}}{[\#(z=0) + \mathbf{1}] + [\#(z=1) + \mathbf{1}]}$$

	$P(\text{"Maha"} \mid \mathbf{spam=1})$	$P(\text{"Maha"} \mid \mathbf{spam=0})$
Mahalanobis	.3	.6
Viagra	.6	.3
	$P(\text{"Viagra"} \mid \mathbf{spam=1})$	$P(\text{"Viagra"} \mid \mathbf{spam=0})$

Now, we can use smoothed probabilities exactly as before:

$$P(\text{"Viagra"}, \text{"Maha"} \mid \mathbf{spam=1}) = P(\text{"Viagra"} \mid \mathbf{spam=1}) P(\text{"Maha"} \mid \mathbf{spam=1})$$
$$P(\text{"Viagra"}, \text{"Maha"} \mid \mathbf{spam=0}) = P(\text{"Viagra"} \mid \mathbf{spam=0}) P(\text{"Maha"} \mid \mathbf{spam=0})$$

As more and **more counts** are available, the **smoothed estimate will converge to the MLE**.

If z has K different outcomes (values)

$$P(z=v_k) = \frac{\#(z=v_k) + \mathbf{1}}{\sum_{m=1}^{K} [\#(z=v_m) + \mathbf{1}]} = \frac{\#(z=v_k) + \mathbf{1}}{[\sum_{m=1}^{K} \#(z=v_m)] + \boldsymbol{K}}$$