

# Bayes and Naïve Bayes

## **Concepts:**

Discriminative and Generative model

Bayes Classifier

Modeling joint distribution of discrete variables

Naïve Bayes assumption

Naïve Bayes classifier

Bernoulli and Multi-nomial Naïve Bayes

MLE and MAP estimation, smoothing

# Two main approaches for learning probabilistic classifiers

- **Discriminative:**

- Learn  $P(y|\mathbf{x})$  directly (don't care about  $P(\mathbf{x})$ )
- Logistic regression is one of such techniques

- **Generative:**

- Learn  $P(y)$  and  $P(\mathbf{x}|y)$
- Compute  $P(y|\mathbf{x})$  using Bayes rule

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

# Generative models

- A generative model specifies a generative “story” regarding how the data  $(\mathbf{x}, y)$  was generated --- modeling both  $\mathbf{x}$  and  $y$ 
  - In contrast, discriminative model can have a “generative story” regarding how  $y$  is produced based on  $\mathbf{x}$
- Given parameters for the model, new data (both  $\mathbf{x}$  and  $y$ ) can be generated
- Maximum likelihood estimation is often used for estimating the model parameters
- Bayes Rule is often used to make predictions

- Let's try to define a generative model for emails of two classes (spam vs. non-spam)
- How to represent an email of  $M$  words as a feature vector? --- popular option: bag of words

**Option 1: binary vector** (indicating presence/absence)

With a vocabulary of size  $V$ ,  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  where  $x_i \in \{0,1\}$

- $x_i = 1$  if email contains the  $i$ th vocabulary word
- $x_i = 0$  otherwise

**Option 2: integer vector** (indicating count\*)

With a vocabulary of size  $V$ ,  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  where  $x_i \in N_0$

- $x_i$ : nonnegative integer count of the  $i$ th vocabulary word

\* It is also common to use normalized counts, e.g, tf-idf

# Generative model for emails

1. Flip a weighted coin ( $y$ ) to decide if spam or not
2. If head ( $y=1$ , aka spam), generate an email  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  from the spam distribution  
 $P(\mathbf{x}|y = 1)$
3. If tail ( $y=0$ , non-spam), generate an email  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  from the non-spam distribution  
 $P(\mathbf{x}|y = 0)$

This is equivalent to factorize the joint distribution of  $\mathbf{x}$  and  $y$  as:

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

# Classification with a generative model

- 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

# Estimating $P(y)$ and $P(\mathbf{x}|y)$

- $P(y)$ : prior distribution of  $y$ 
  - $P(y = 1)$ : portion of spams
  - $P(y = 0)$ : portion of non-spams
- $P(\mathbf{x}|y)$ : the distribution of  $\mathbf{x}$  given  $y$ 
  - $P(\mathbf{x}|y = 1)$ : distribution of  $\mathbf{x}$  given  $y = 1$  (spam)
  - $P(\mathbf{x}|y = 0)$ : distribution of  $\mathbf{x}$  given  $y = 0$  (non-spam)
- Learning  $P(\mathbf{x}|y = 1)$  or  $P(\mathbf{x}|y = 0)$  is a (joint) density estimation problem

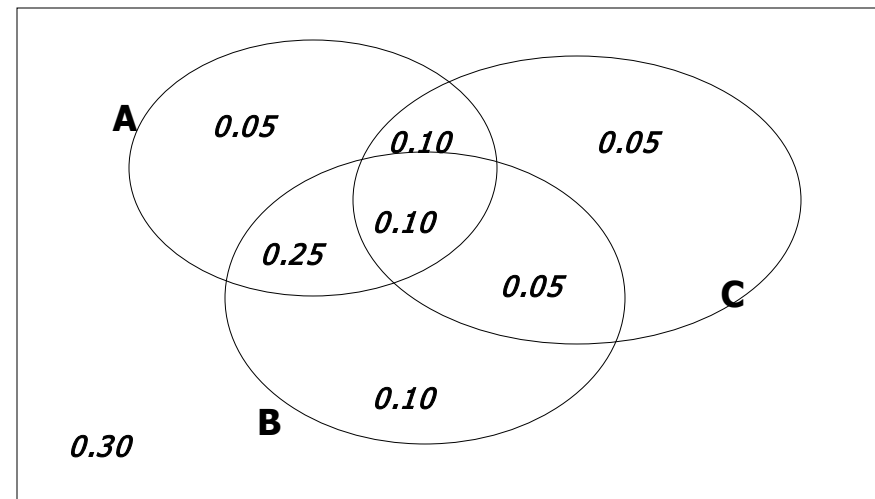
# Joint Distribution of Boolean Variables

Recipe for making a joint distribution of  $D$  variables:

1. Make a table listing all value combinations of the variables ( $D$  binary variables  $\Rightarrow 2^M$  rows).
2. For each value combination, say how probable it is.
3. Due to the axioms of probability, those numbers must sum to 1.

*Example: Binary variables  $A, B, C$*

<b>A</b>	<b>B</b>	<b>C</b>	<b>Prob</b>
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 Joint Dist. 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	?

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






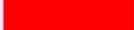
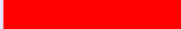
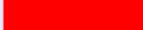
The fill in each row with

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

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	<b>0.25</b>
1	1	1	0.10

# 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>

# A basic classifier

Given training data:  $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N)$ ,  
where  $\mathbf{x}_i \in \{0,1\}^d, y_i \in \{1,2, \dots, K\}$

- Learn  $P(y)$
- Learn  $P(\mathbf{x}|y = 1), P(\mathbf{x}|y = 2), \dots, P(\mathbf{x}|y = K)$  each as a large joint distribution table with  $2^d$  entries
- Compute

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

- Predict with decision theory or use:  
$$\operatorname{argmax}_y P(y|\mathbf{x}) = \operatorname{argmax}_y P(\mathbf{x}|y)P(y)$$

# Joint distribution overfits

- Let  $\mathbf{x}$  be a  $d$ -dim binary vector, and  $y \in \{1, 2, \dots, K\}$
- Learning the joint distribution  $P(\mathbf{x}|y = i)$  for  $i = 1, \dots, K$  involves estimating  $K \times (2^d - 1)$  parameters
- For large  $d$ , this number is prohibitively large and we don't have enough data to estimate them accurately
- A common situation: no training examples have the exact  $\mathbf{x} = [u_1, \dots, u_d]^T$  value combination
  - $P(\mathbf{x} = [u_1, \dots, u_d]^T | y = i) = 0$  for all values of  $i$
  - Overfitting

# Naïve Bayes Assumption

- Assume  $p(x_1, x_2, \dots, x_d|y) = p(x_1|y)p(x_2|y) \dots p(x_d|y)$   
i.e., the features are independent from one another given  
the class label

- **Conditional independence:**  $x$  is **(conditionally) independent** of  $y$  given  $z$ , if

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

Or equivalently

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

Often denoted as

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

$$\text{or } p(x, y|z) = p(x|z)p(y|z)$$

# Example

$$p(\text{thunder}|\text{raining}, \text{lightening}) = p(\text{thunder}|\text{lightening})$$

- Note that the events thunder and raining are not independent from one another
  - Much high probability of thunder when it is raining
- However, once we observe lightening, they become independent
  - With lightening you will get thunder, whether there is rain or not
- They are **conditionally independent** given lightening

# Conditional independence vs. independence

- Conditional independence ( $x$  and  $y$  are conditionally independent given  $z$ ):

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

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

- Independence ( $x$  and  $y$  are independent):

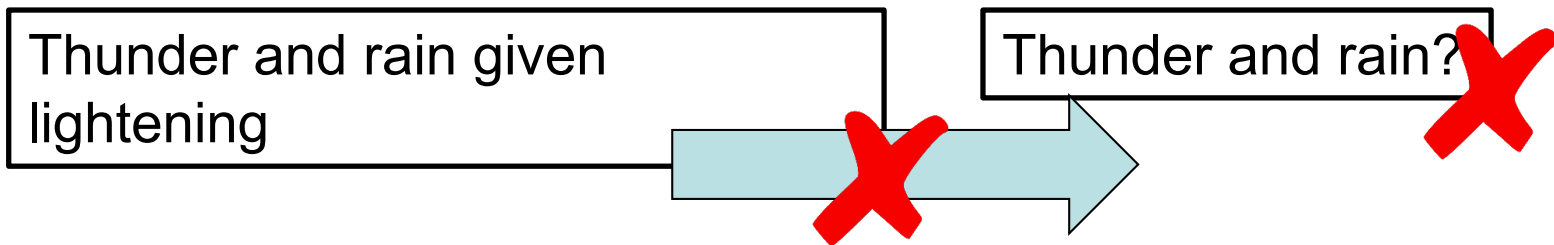
$$p(x, y) = p(x)p(y)$$

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

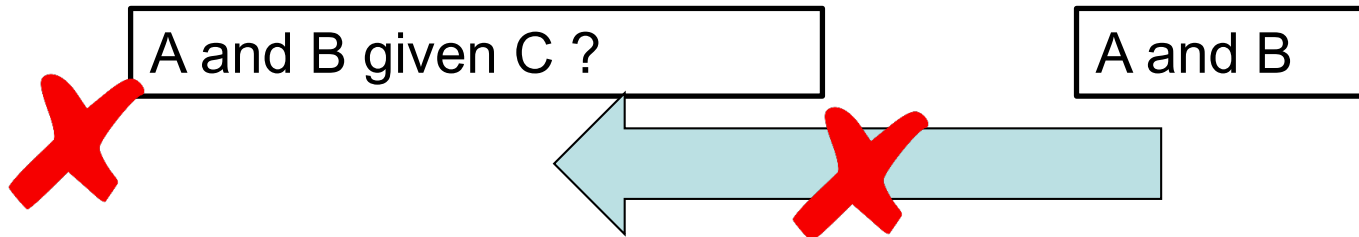
# Conditional independence $\neq$ Independence

Conditional  
Independence

Independence



Let A and B be two random variables representing rock paper scissor for two opponents, and C represents the outcome of the game

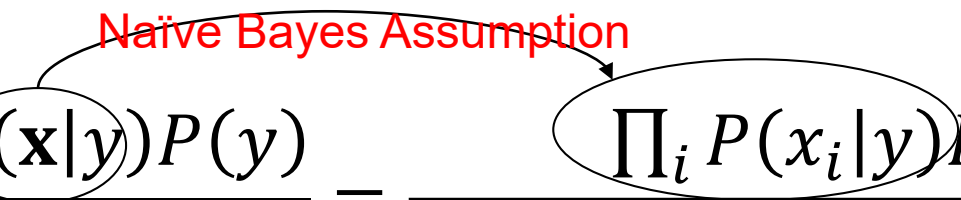




# Naive Bayes Classifier

Given training data

- Learn  $P(y = j)$  for  $j = 1, \dots, K$
- Learn  $P(x_i|y = j)$  for  $i = 1, \dots, d$  and  $j = 1, \dots, K$
- Compute


$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \frac{\prod_i P(x_i|y)P(y)}{\sum_j (\prod_i P(x_i|y = j)P(y = j))}$$

- Predict with decision theory or use:  
$$\arg \max_y P(y|\mathbf{x})$$

# of parameters? see concept warehouse question

# Example

$X_1$	$X_2$	$X_3$	$Y$
1	1	1	0
1	1	0	0
0	0	0	0
0	1	0	1
1	0	1	1
0	1	1	1

Apply Naïve Bayes to this data and use the learned model to compute:

$$P(y = 1|(1,0,0))$$

$$\frac{0.5 * \frac{1}{3} * \frac{1}{3} * \frac{1}{3}}{0.5 * \frac{1}{3} * \frac{1}{3} * \frac{1}{3} + 0.5 * \frac{2}{3} * \frac{1}{3} * \frac{2}{3}}$$

- Represent an email of  $M$  words as a feature vector using bag of words

**Option 1: binary vector** (indicating presence/absence)

With a vocabulary of size  $V$ ,  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  where  $x_i \in \{0, 1\}$

- $x_i = 1$  if email contains the  $i$ th vocabulary word
- $x_i = 0$  otherwise

**Bernoulli Model**

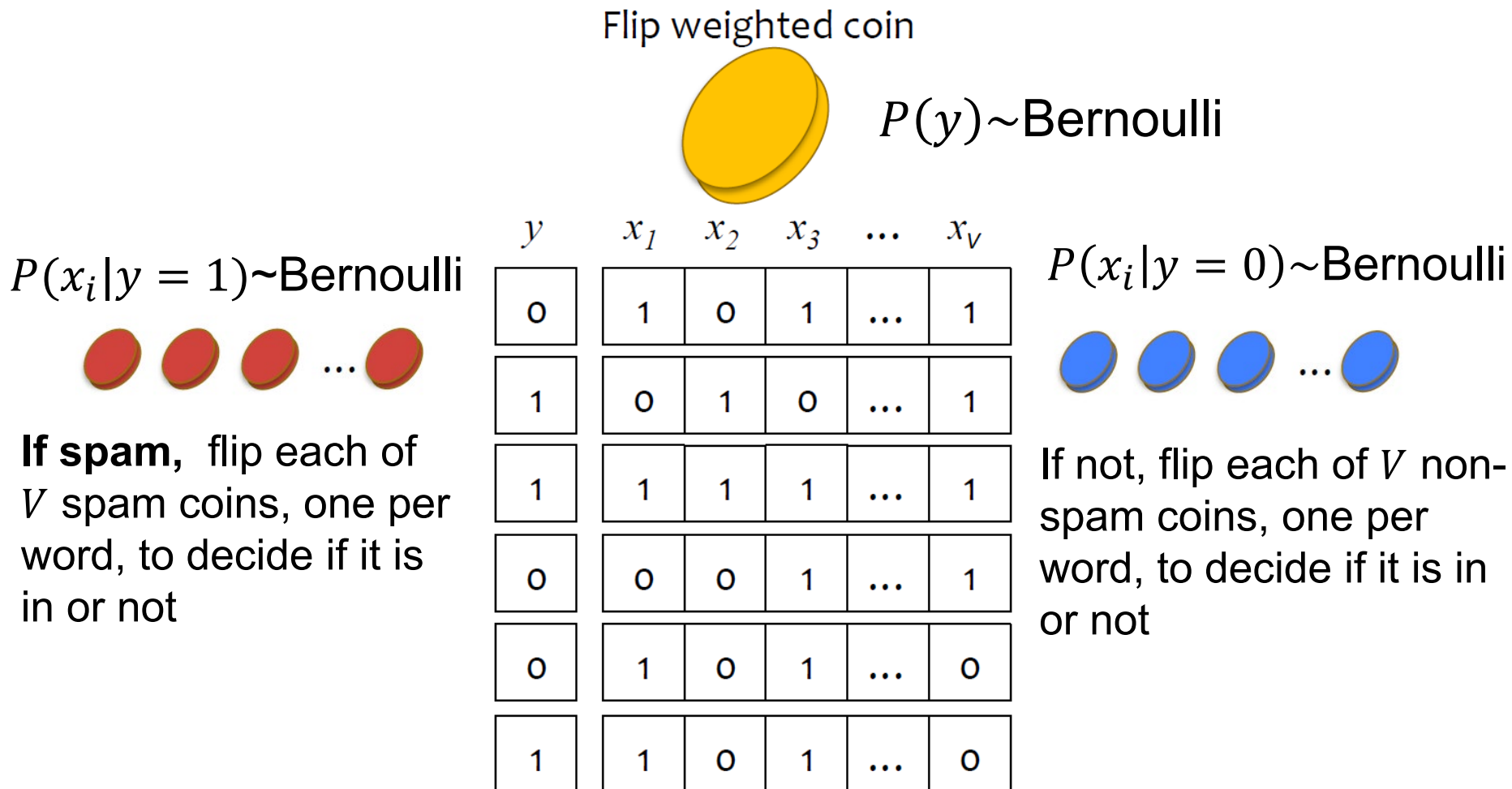
**Option 2: integer vector** (indicating count)

With a vocabulary of size  $V$ ,  $\mathbf{x} = [x_1, x_2, \dots, x_V]$  where  $x_i \in N_0$

- $x_i$ : nonnegative integer count of the  $i$ th vocabulary word

**Multinomial Model**

# Generative story for Bernoulli Naïve Bayes



We do not really generate emails this way --- but this is what we assume to be the process how emails are generated

# Generative story for Multi-nomial Naïve Bayes

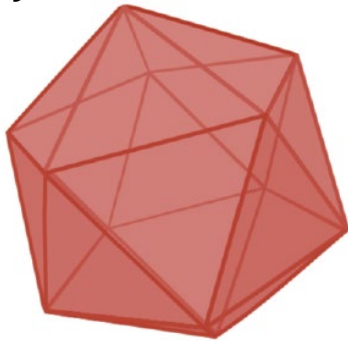
Generating an email of length  $M$

Flip weighted coin



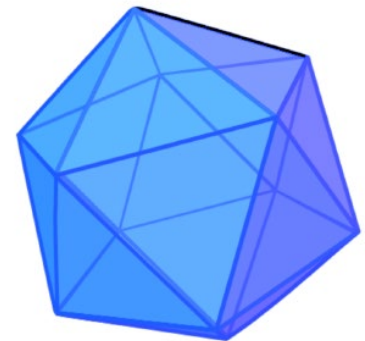
$$P(y) \sim \text{Bernoulli}$$

$$P(x_i | y = 1) \sim \text{Multinomial}$$



If head, roll the spam die  $M$  times and record the count for each of the  $V$  sides

$$P(x_i | y = 0) \sim \text{Multinomial}$$



If tail, roll the non-spam die  $M$  times and record the count for each of the  $V$  sides

$y$	$x_1$	$x_2$	$x_3$	...	$x_V$
0	2	0	7	...	1
1	0	1	0	...	5
1	10	1	0	...	1
0	0	0	15	...	3
0	11	0	0	...	0
1	7	0	1	...	0

We do not really generate emails this way --- but this is what we assume to be the process how emails are generated

# Multinomial model vs Bernoulli model

- The likelihood of observing one email  $E$  :

Bernoulli model:

Each feature  $x_i \in \{0,1\}$

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

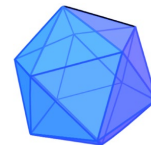


Single occurrence is no different to multiple occurrences

Multinomial model:

Each feature  $x_i \in N_0$   
(i.e., nonnegative integer)

$$P(\mathbf{x}|y) = \prod_{i=1}^V P(w_i|y)^{x_i}$$



Multiple occurrences make a difference. Allow us to take counts into consideration

# MLE for Naïve Bayes with Bernoulli Model

## For Spam Filter

Given a set of  $N$  training emails, MLE of the parameters are:

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

- For each feature  $i$ , learn a Bernoulli model for each class:

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

# of spam emails the  $i$ -th word appeared

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

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

# of non-spam emails the  $i$ -th word appeared

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

Total number of parameters for  $K$  classes?  $(K - 1) + K * V$

# MLE for Naïve Bayes with Multinomial model

- MLE estimate for the  $i$ -th word in the dictionary:

$$p(w_i|y) = \frac{\text{total \# of word } i \text{ in class } y \text{ emails}}{\text{total \# of words in class } y \text{ emails}}$$

- Total number of parameters?
  - $K(V - 1) + (K - 1)$



# 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:  $x_i \in R$ 
  - Learn a continuous distribution for each  $p(x_i|y)$ , e.g., Gaussian
  - Alternatively, we can 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” (the  $n + 1$  th word in the vocabulary) has never appeared in the training emails
$$P(x_{n+1} = 1|y = 1) = P(x_{n+1} = 1|y = 0) = 0$$
- 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 results in prob. of 0 or 1 which are “too strong” and cause problems
- To solve this problem, we can use **Maximum A Posterior** (MAP) estimate

# Bayesian Parameter Estimation

- Parameters are random variables
- Prior distribution captures our prior belief about the parameter before seeing any data
- When the observed data is sparse, we fall back to the prior and avoid the issues faced by MLE

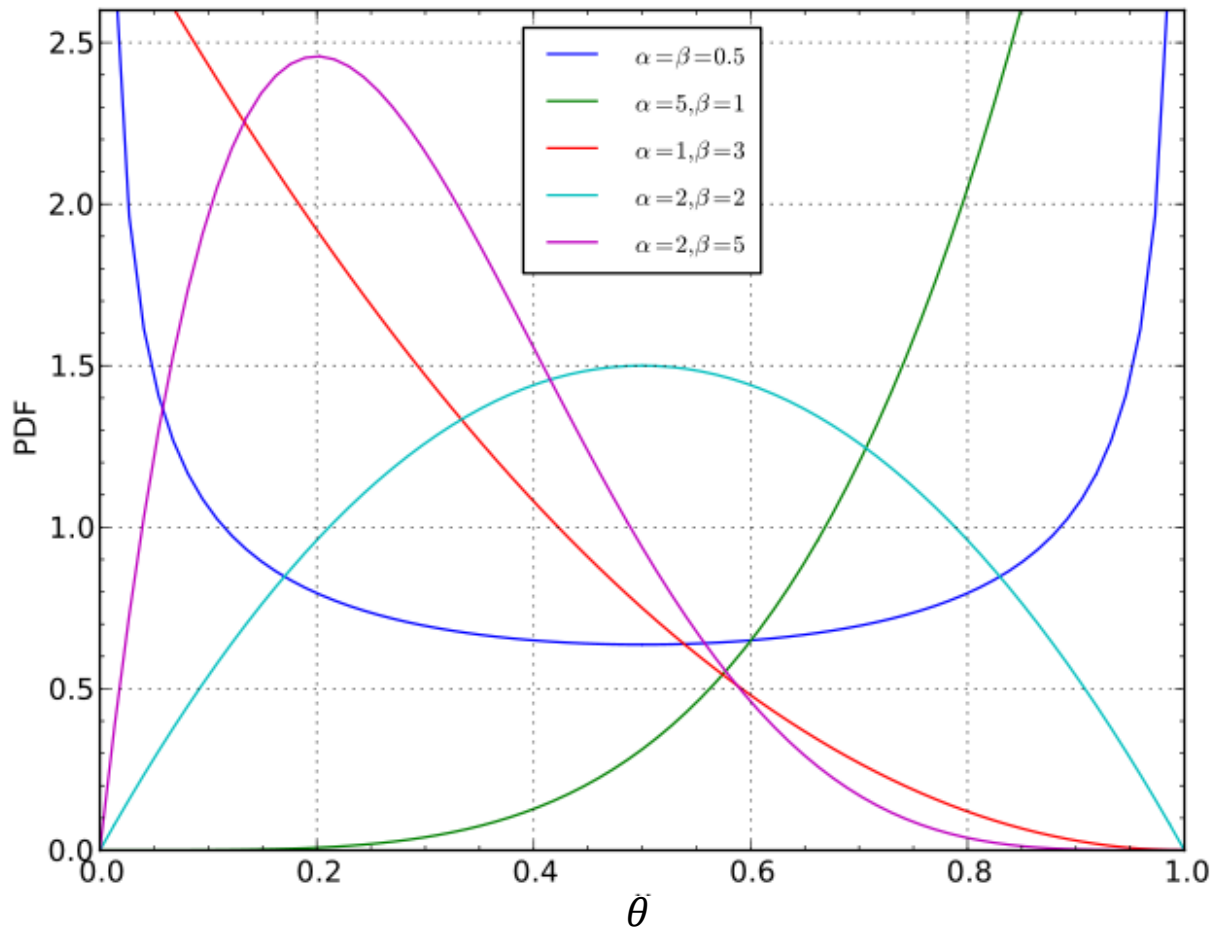
# Example: Bernoulli

- Given a unfair coin, we want to estimate  $\theta$  - the probability of head
- We toss the coin  $n$  times, and observe  $n_1$  heads
- MLE estimate:  $\theta = \frac{n_1}{n}$
- Now assume that  $\theta$  is a random variable and has a prior
- For reasons that will become clear later, we assume the following prior for  $\theta$ :

$$\theta \sim \text{Beta}(\alpha, \beta)$$
$$p(\theta; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1 - \theta)^{\beta-1}$$

# Beta distribution

$$p(\theta; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1 - \theta)^{\beta-1}$$



- $\alpha = \beta = 1$ : uniform
- $\alpha, \beta < 1$ : U-shape
- $\alpha, \beta > 1$ : uni-model
- $\alpha = \beta$ : symmetric
- Mode ( $\alpha, \beta > 1$ ):  
$$\frac{\alpha - 1}{\alpha + \beta - 2}$$

What parameter should you choose if you strongly believe your parameter is approximately 2/3?

# Posterior distribution of $\theta$

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

$$p(\theta|D) \propto \theta^{n_1}(1-\theta)^{n_0} \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

$$p(\theta|D) \propto \frac{1}{B(\alpha, \beta)} \theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1}$$

$$p(\theta|D) = \frac{1}{B(n_1 + \alpha, n_0 + \beta)} \theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1}$$

Noting that the posterior has exactly the same form as the prior. This is not a coincidence, it is due to careful selection of the prior distribution – **conjugate prior**

# Maximum *a-Posterior* (MAP)

- A full Bayesian treatment will not care about estimating the parameters
- Instead, it will use the posterior distribution to make predictions for the target variable by marginalizing  $\theta$

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x}, \theta) p(\theta|D) d\theta$$

- This can be computationally expensive or intractable
- Frequently, we use Maximum A Posterior estimation:

$$\theta_{map} = \arg \max_{\theta} p(\theta|D)$$

# MAP for Bernoulli

$$p(\theta|D) = \frac{1}{B(n_1 + \alpha, n_0 + \beta)} \theta^{n_1 + \alpha - 1} (1 - \theta)^{n_0 + \beta - 1}$$

- Mode: the maximum point for  $Beta(a, b)$  (for  $a, b > 1$ ) is

$$\frac{a - 1}{a + b - 2}$$

- In this case  $a = n_1 + \alpha, b = n_0 + \beta$ , we have:

$$\theta_{MAP} = \frac{n_1 + \alpha - 1}{n + \alpha + \beta - 2}$$

- Let  $\alpha = 2, \beta = 2$ , we have  $\theta_{MAP} = \frac{n_1 + 1}{n + 2}$ 
  - Comparing to MLE, it is like adding two fake coin tosses (one head, one tail) into the observed data – this is also called Laplace smoothing



# MAP for Naïve Bayes Spam Filter

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

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

$$P(x_i = 1 | y = 0) = \frac{N_{i|0} + 1}{N_0 + 2} \quad \text{MAP } (\alpha = \beta = 2), \text{ or Laplace smoothing}$$

- When encounter a new word not appeared in training set, now the probabilities do not go to zero

# MAP estimation for Multi-nomial

- The conjugate prior for multinomial is called Dirichlet distribution
- For  $K$  outcomes, a Dirichlet distribution has  $K$  parameters, each serves a similar purpose as Beta distribution's parameters
  - Acting as fake observation(s) for the corresponding output, the count depends on the value of the parameter
- Laplace smoothing for this case:

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

# Laplace Smoothing for Multi-nomial case

MLE:  $p(w_i|y = 1) = \frac{\text{total \# of } i\text{-th word in spam emails}}{\text{total \# of words in spam emails}}$

With Laplace smoothing:

$$p(w_i|y = 1) = \frac{\text{total \# of } i\text{-th word in spam emails} + 1}{\text{total \# of words in spam emails} + V}$$

where  $V$  is the size of the vocabulary

# Summary

- Generative classifier
  - learn  $P(\mathbf{x}|y)$  and  $P(y)$
  - Use Bayes rule to compute  $P(y|\mathbf{x})$  for classification
  - Predict with  $\arg \max_y P(y|\mathbf{x})$  or use decision theory
- Naïve Bayes assumes conditional independence between features given class labels
  - Greatly reduces the numbers of parameters to learn
  - Assumptions are not always satisfied in reality but often works reasonably well
  - Extremely efficient and can scale to large data and high dimensionality
- Bernoulli and Multi-nomial Naïve Bayes for text classification
  - It can be show that both lead to linear classifiers
- MAP estimation (or smoothing) can be helpful in avoiding overfitting and extreme probability values

# Comparing Naïve Bayes with Logistic regression

## Logistic Regression

- Discriminative:
  - Learns  $p(y|\mathbf{x})$
  - Do not explicitly model  $p(\mathbf{x})$
  - Learns a linear decision boundary
- Iterative learning
  - Concave objective: L2 regularized log-likelihood
  - Global optimal solution

## Naïve Bayes

- Generative
  - Learns  $p(\mathbf{x}|y)$  and  $p(y)$
  - Use Bayes rule to compute  $p(y|\mathbf{x})$
  - For discrete features, or continuous feature under certain Gaussian assumption, also learns a linear decision boundary
- Learning involves MLE (or MAP) for  $p(y)$  and  $p(\mathbf{x}|y)$  --- highly efficient and scalable both in terms of dimension and training set size

# Connection between LG and NB

- We have:

$$P(y = 1|\mathbf{x}) = \frac{P(y = 1)P(\mathbf{x}|y = 1)}{P(y = 1)P(\mathbf{x}|y = 1) + P(y = 0)P(\mathbf{x}|y = 0)}$$

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \frac{P(y = 0)P(\mathbf{x}|y = 0)}{P(y = 1)P(\mathbf{x}|y = 1)}}$$

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp^{-\log \frac{P(y=1)P(\mathbf{x}|y = 1)}{P(y=0)P(\mathbf{x}|y = 0)}}}$$

- In your homework assignment, you will show that when  $\mathbf{x}$  is binary, with the naïve bayes assumption,  $\log \frac{P(y=1)P(\mathbf{x}|y=1)}{P(y=0)P(\mathbf{x}|y=0)}$  is a linear function of  $\mathbf{x}$
- Similarly if  $\mathbf{x}$  is continuous and Gaussian, with the naïve bayes assumption we have:

$$\log \frac{P(y=1)P(\mathbf{x}|y=1)}{P(y=0)P(\mathbf{x}|y=0)} = \log \frac{P(y=1)}{P(y=0)} + \log \frac{\prod_i P(x_i|y=1)}{\prod_i P(x_i|y=0)}$$

If we assume

$$\begin{aligned} P(x_i|y=1) &\sim \text{Gaussian}(\mu_{i1}, \sigma_i^2) \\ P(x_i|y=0) &\sim \text{Gaussian}(\mu_{i0}, \sigma_i^2) \end{aligned}$$

**Shared variance,  
different mean**

$$\begin{aligned} \log \frac{\prod_i P(x_i|y=1)}{\prod_i P(x_i|y=0)} &= \sum_i \log \frac{P(x_i|y=1)}{P(x_i|y=0)} = \sum_i \log P(x_i|y=1) - \log P(x_i|y=0) \\ &= \sum_i \log \exp \frac{-(x_i - \mu_{i1})^2}{2\sigma_i^2} - \log \exp \frac{-(x_i - \mu_{i0})^2}{2\sigma_i^2} = \sum_i -\frac{(x_i - \mu_{i1})^2}{2\sigma_i^2} + \frac{(x_i - \mu_{i0})^2}{2\sigma_i^2} \\ &= \sum_i \frac{2\mu_{i1}x_i - 2\mu_{i0}x_i}{2\sigma^2} + \frac{\mu_{i0}^2 - \mu_{i1}^2}{2\sigma_i^2} \end{aligned}$$

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\log \frac{P(y = 1)P(\mathbf{x}|y = 1)}{P(y = 0)P(\mathbf{x}|y = 0)})}$$

$$\log \frac{P(y = 1)P(\mathbf{x}|y = 1)}{P(y = 0)P(\mathbf{x}|y = 0)} = \log \frac{p(y = 1)}{p(y = 0)} + \sum_i \frac{\mu_{i0}^2 - \mu_{i1}^2}{2\sigma_i^2} + \sum_i \frac{\mu_{i1} - \mu_{i0}}{\sigma_i^2} x_i$$

$\Rightarrow$

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp -(w_0 + w_1x_1 + \cdots + w_dx_d)} = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

Where

$$w_0 = \log \frac{p(y=1)}{p(y=0)} + \sum_i \frac{\mu_{i0}^2 - \mu_{i1}^2}{2\sigma_i^2} \quad \text{and} \quad w_i = \frac{\mu_{i1} - \mu_{i0}}{\sigma_i^2}$$



# Implications

- Naïve Bayes leads to the same functional form for  $P(y = 1|\mathbf{x})$  as logistic regression
- The  $w$ 's of Naïve bayes' are learned by learning the generative probabilities, the values of  $w$ 's then follows
  - Hence, they are learned in a more constrained way, constrained by the distributional assumptions (Gaussian with shared variance, Naïve bayes)
- Logistic regression learns the weights directly
  - Less assumptions are made
  - Less constrained
- Which one is better?
  - When the distributional assumptions made by NB is correct, it can learn with substantially fewer training examples because the distribution assumptions are rich knowledge that are directly infused into the model
  - When the assumptions of NB are incorrect --- Logistic regression will lead to better performance given sufficient data