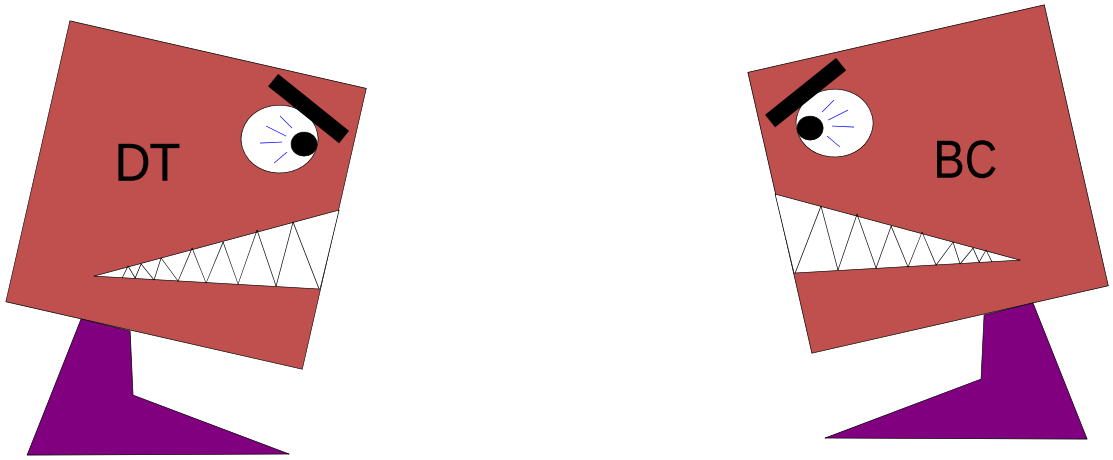
Bayes Classifier and Naïve Bayes

CS434

Bayes Classifiers

- A formidable and sworn enemy of decision trees



Probabilistic Classification

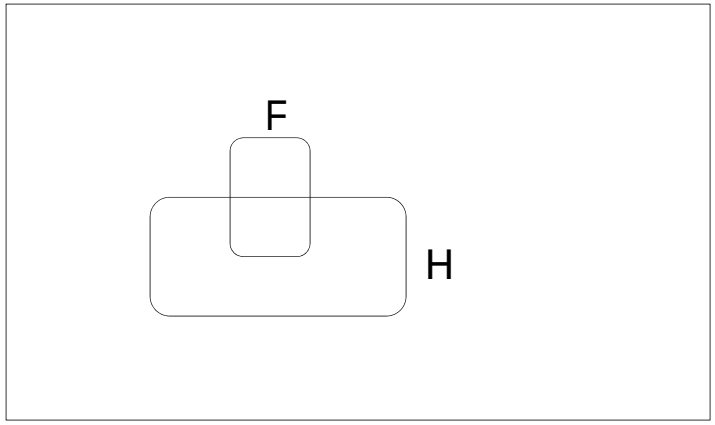
- Credit scoring:
  - Inputs are income and savings
  - Output is low-risk vs high-risk
- Input: $\boldsymbol{x} = [x_1, x_2]^T$,Output: $C \in \{0,1\}$
- Prediction:

$$\text{choose} \begin{cases} C = 1 \text{ if } P(C = 1 \mid x_1, x_2) > 0.5 \\ C = 0 \text{ otherwise} \end{cases}$$

or equivalently

$$\text{choose} \begin{cases} C = 1 \text{ if } P(C = 1 \mid x_1, x_2) > P(C = 0 \mid x_1, x_2) \\ C = 0 \text{ otherwise} \end{cases}$$

A side note: probabilistic inference



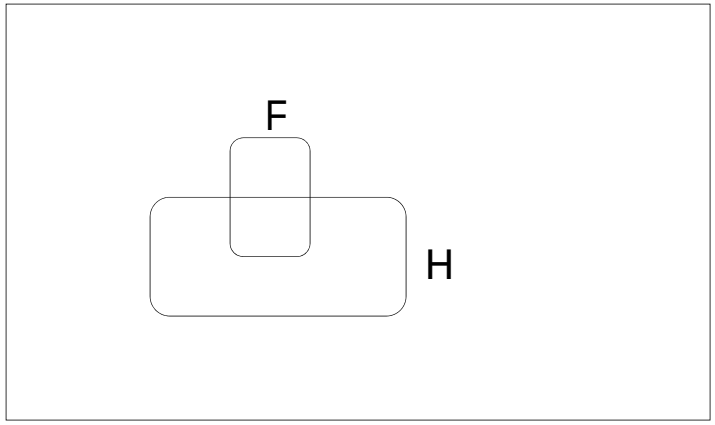
H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

One day you wake up with a headache. You think: “Drat! 50% of flus are associated with headaches so I must have a 50-50 chance of coming down with flu”

Is this reasoning good?

Probabilistic Inference



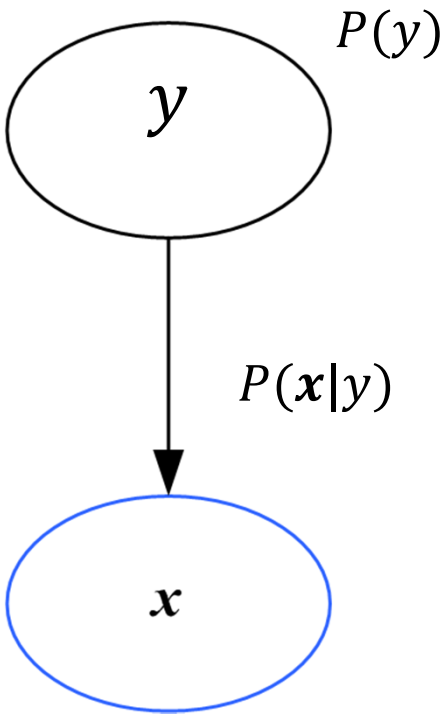
H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

$$P(F \wedge H) = P(F)P(H \mid F) = \frac{1}{40} * \frac{1}{2} = \frac{1}{80}$$

$$P(F \mid H) = \frac{P(F \wedge H)}{P(H)} = \frac{1}{8}$$

Bayes classifier



A simple bayes net

$$P(y \mid \mathbf{x}) = \frac{P(y)\,p(\mathbf{x} \mid y)}{p(\mathbf{x})}$$

prior

posterior

Given a set of training examples, to build a Bayes classifier, we need to

1. Estimate P(y) from data
2. Estimate P(x|y) from data

Given a test data point x, to make prediction

1. Apply bayes rule: $P(y \mid \mathbf{x}) \propto P(y)P(\mathbf{x} \mid y)$
2. Predict $\arg\max_y P(y \mid \mathbf{x})$

Maximum Likelihood Estimation (MLE)

- Let y be the outcome of the credit scoring of a random loan applicant, yϵ{0,1}
  - P_0=P(y=0), and P_1=P(y=1)=1-P_0
- This can be compactly represented as

$$P(y) = P_0^{(1-y)}(1-P_0)^y$$

- If you observe n samples of y: y_1, y_2,…,y_n
- we can write down *the likelihood function* (i.e. the probability of the observed data given the parameters*)*:

$$L(p_0) = \prod_{i=1}^{n} p_0^{1-y_i}(1-p_0)^{y_i}$$

- The log-likelihood function:

$$l(p_0) = \log L(p_0) = \log \prod_{i=1}^{n} p_0^{1-y_i}(1-p_0)^{y_i}$$

$$= \sum_{i}^{n} [(1-y_i)\log p_0 + y_i \log(1-p_0)]$$

MLE cont.

- MLE maximizes the likelihood, or the log likelihood

$$p_0^{MLE} = \arg\max_{p_0} l(p_0)$$

- For this case:

$$p_0^{MLE} = \frac{\sum_{i=1}^{n} (1 - y_i)}{n}$$

i.e., the frequency that one observes y=0 in the training data

Bayes Classifiers in a nutshell

1. Estimate P(x_1, x_2, … x_m | $y=v_i$) for each value v_i
3. Estimate P($y=v_i$) as fraction of records with $y=v_i$.

} *learning*

4. For a new prediction:

$$y^{\text{predict}} = \operatorname*{argmax}_v P(y = v \mid x_1 = u_1 \cdots x_m = u_m)$$
$$= \operatorname*{argmax}_v P(x_1 = u_1 \cdots x_m = u_m \mid y = v) P(y = v)$$

Estimating the joint distribution of x_1, x_2, … x_m given y can be problematic!

Joint Density Estimator Overfits

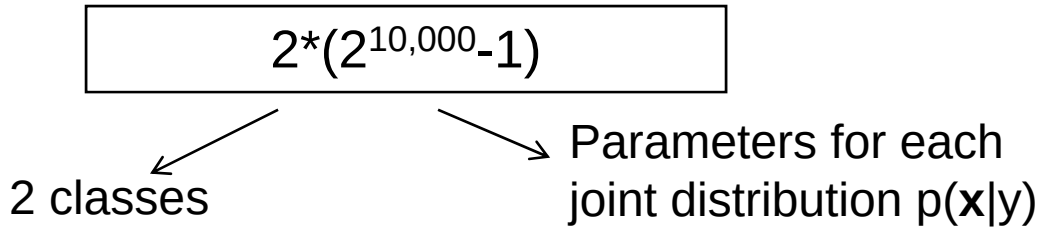
- Typically we don't have enough data to estimate the joint distribution accurately
- It is common to encounter the following situation:
  - If no training examples have the exact **x**=(u_1, u_2, …. u_m), then P(x|y=v_i) = 0 for all values of Y.

- In that case, what can we do?
  - we might as well guess a random y based on the prior, i.e., p(y)

$$P(y \mid \mathbf{x}) = \frac{P(y)\, p(\mathbf{x} \mid y)}{p(\mathbf{x})}$$

Example: Spam Filtering

- Assume that our vocabulary contains 10k commonly used words & tokens--- we have 10,000 attributes
- Let's assume these attributes are binary
- How many parameters that we need to learn?

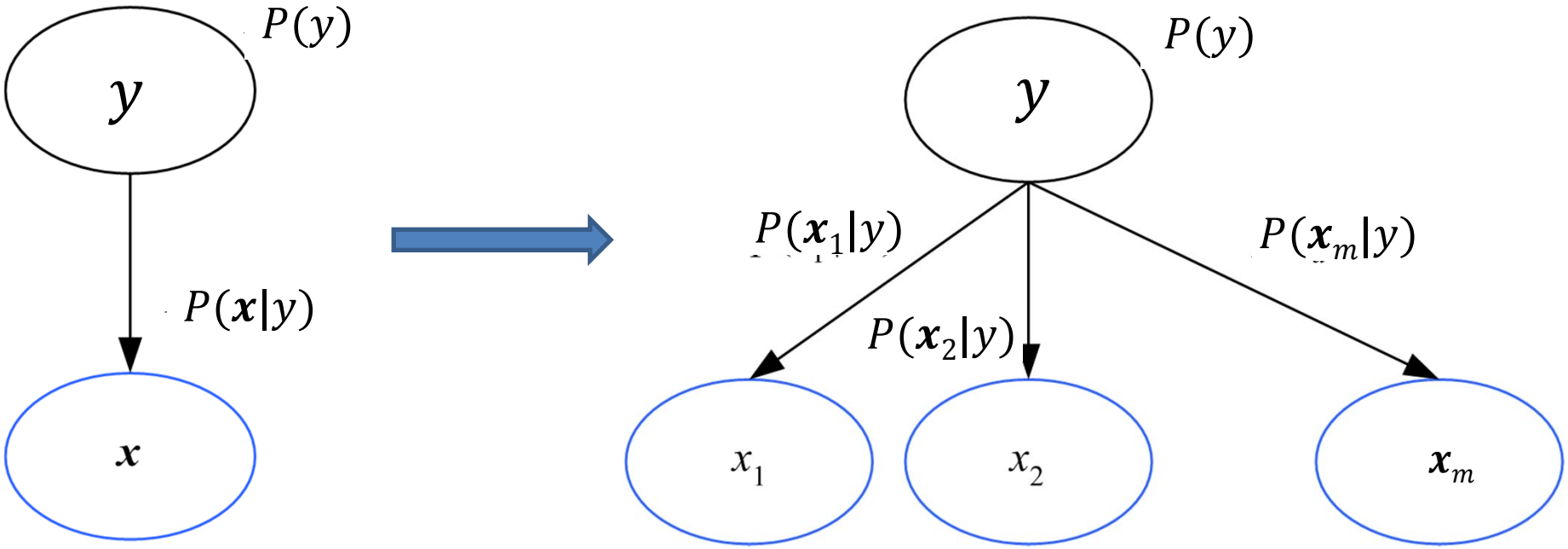


Clearly we don't have enough data to estimate that many parameters

The Naïve Bayes Assumption

- Assume that each attribute is independent of any other attributes given the class label

$$P(x_1 = u_1 \cdots x_m = u_m \mid y = v_i)$$
$$= P(x_1 = u_1 \mid y = v_i) \cdots P(x_m = u_m \mid y = v_i)$$



A note about independence

- Assume A and B are two Random Variables. Then

"A and B are independent"

if and only if

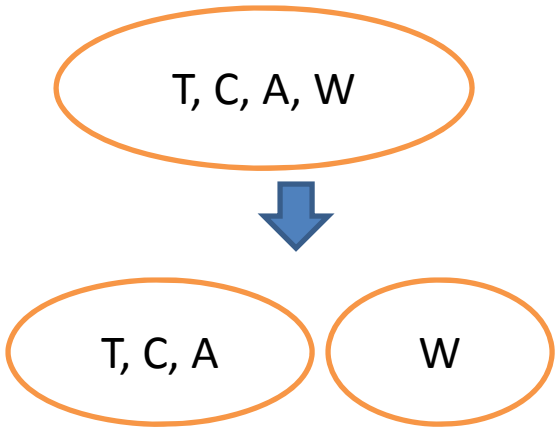
$$P(A|B) = P(A)$$

- "A and B are independent" is often notated as

$$A \perp B$$

Examples of independent events

- Two separate coin tosses
- Consider the following four variables:
  - T: Toothache (I have a toothache)
  - C: Catch (dentist's steel probe catches in my tooth)
  - A: Cavity
  - W: Weather
  - P(T, C, A, W) =?



Conditional Independence

- $P(x_1 | x_2, y) = P(x_1 | y)$
  - X_1 is independent of x_2 given y
  - x_1 and x_2 are conditionally independent given y
- If X_1 and X_2 are conditionally independent given y, then we have
  - $P(X_1, X_2 | y) = P(X_1 | y)\, P(X_2 | y)$

Example of conditional independence

- T: Toothache (I have a toothache)
- C: Catch (dentist's steel probe catches in my tooth)
- A: Cavity

T and C are conditionally independent given A: $P(T|C,A) = P(T|A)$

$$P(T, C|A) = P(T|A)*P(C|A)$$

Events **that are not independent from each other might be conditionally independent given some fact**

It can also happen the other way around. **Events that are independent might become conditionally dependent given some fact.**

B=Burglar in your house; A = Alarm **(**Burglar**)** rang in your house
E = Earthquake happened
B is independent of E (ignoring some minor possible connections between them)
However, if we know A is true, then B and E are no longer independent. Why?
$P(B|A) >> P(B|A, E)$ Knowing E is true makes it much less likely for B to be true

Naïve Bayes Classifier

- By assuming that each attribute is independent of any other attributes given the class label, we now have a *Naïve* Bayes Classifier
- Instead of learning a joint distribution of all features, we learn $p(x_i|y)$ separately for each feature x_i
- Everything else remains the same

Naïve Bayes Classifier

- Assume you want to predict output *y* which has n_Y values v_1, v_2, … v_{ny}.
- Assume there are *m* input attributes called **x**=(x_1, x_2, … x_m)
- Learn a conditional distribution of p(**x**|y) for each possible y value, y = v_1, v_2, … v_{ny}, we do this by:
  - Break training set into n_Y subsets called S_1, S_2, … S_{ny} based on the y values, i.e., S_i contains examples in which $y=v_i$
  - For each S_i, learn $p(y=v_i)= |S_i| / |S|$
  - For each S_i , learn the conditional distribution each input features, e.g.:

$$\boxed{P(x_1 = u_1 \mid y = v_i), \cdots, P(x_m = u_m \mid y = v_i)}$$

$$y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(x_1 = u_1 \mid y = v) \cdots P(x_m = u_m \mid y = v) P(y = v)$$

Example

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

Apply Naïve Bayes, and make prediction for (1,0,1)?

1. Learn the prior distribution of y.
   P(y=0)=1/2, **P**(y=1)=1/2
2. Learn the conditional distribution of x_i given y for each possible y values
   p(X_1|y=0), **p**(X_1|y=1)
   p(X_2|y=0), **p**(X_2|y=1)
   p(X_3|y=0), **p**(X_3|y=1)

For example, **p**(X_1|y=0):
P(X_1=1|y=0)=2/3, **P**(X_1=1|y=1)=0
…

To predict for (1,0,1):

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

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

Laplace Smoothing

- With the Naïve Bayes Assumption, we can still end up with zero probabilities
- E.g., if we receive an email that contains a word that has never appeared in the training emails
  - P(**x**|y) will be 0 for all y values
  - We can only make prediction based on p(y)
- This is bad because we ignored all the other words in the email because of this single rare word
- Laplace smoothing can help

  $P(X_1=1|y=0)$

  = (**1+** # of examples with y=0, X_1=1) /(**k+** # of examples with y=0)

  k = the total number of possible values of x

- For a binary feature like above, p(x|y) will not be 0

Final Notes about (Naïve) Bayes Classifier

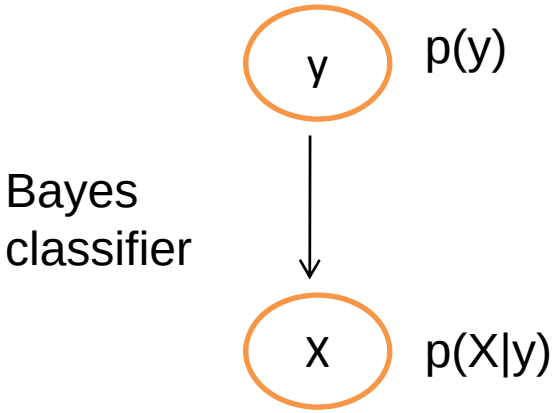
- Any density estimator can be plugged in to estimate $P(x_1,x_2, \ldots, x_m \mid y)$, or $P(x_i \mid y)$ for Naïve bayes

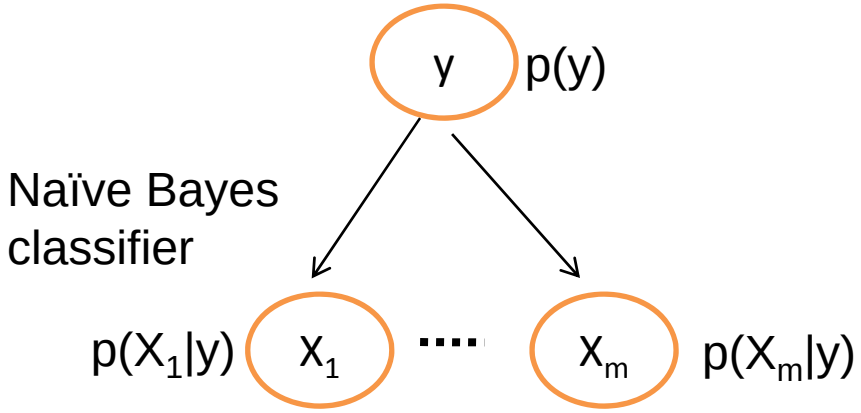
- Real valued attributes can be modeled using simple distributions such as Gaussian (Normal) distribution

- Naïve Bayes is wonderfully cheap and survives tens of thousands of attributes easily

Bayes Classifier is a ***Generative Approach***

- Generative approach:
  - Learn p(y), p(**x**|y), and then apply bayes rule to compute p(y|**x**) for making predictions
  - This is equivalent to assuming that each data point is generated following a ***generative process*** governed by p(y) and p(X|y)





- Generative approach is just one type of learning approaches used in machine learning
  - Learning a correct generative model is difficult
  - And sometimes unnecessary
- KNN and DT are both what we call discriminative methods
  - They are not concerned about any generative models
  - They only care about finding a good discriminative function
  - For KNN and DT, these functions are deterministic, not probabilistic
- One can also take a probabilistic approach to learning discriminative functions
  - i.e., Learn $p(y|X)$ directly without assuming X is generated based on some particular distribution given y (i.e., $p(X|y)$)
  - Logistic regression is one such approach