Probabilistic ML algorithm

Naïve Bayes and Maximum Likelyhood

There is uncertainty in the world

- **Many machine learning problems can be formulated in probabilistic terms**. A probabilistic formulation is the following:
  - the target of a ML system is to learn a classification function

    f(x): x➔y
  - Given an unseen instance **x**, assign a category label *y* to this instance using the learned function f().
  - The corresponding probability formulation is: learn a **probability density function** p(y/**x**) which is the *conditioned probability of class label y given the observation of* ***x***.
  - A label can be selected based on $argmax_y p(y/\boldsymbol{x})$

Example: handwriting recognition

Is this “a” or “9”??

The classifier may return non-zero probabilities for all options

BASIC PROBABILITY NOTIONS YOU NEED

Axioms of Probability Theory

- All probabilities between 0 and 1

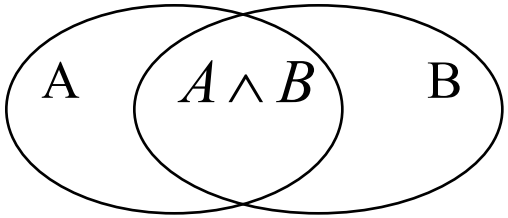
$$0 \le P(A) \le 1$$

- True proposition has probability 1, false has probability 0.

$$\text{P(true)} = 1 \qquad \text{P(false)} = 0.$$

- The probability of disjunction is:

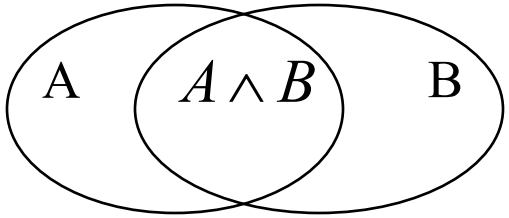
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



Conditional Probability

- P($A \mid B$) is the probability of A given B
- Assumes that B is all and only information known.
- Defined by:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$



Statistical Independence

- A and B are *independent* iff:

$$P(A \mid B) = P(A)$$
$$P(B \mid A) = P(B)$$

These two constraints are logically equivalent

- Therefore, if A and B are independent:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = P(A)$$

$$P(A \wedge B) = P(A)P(B)$$

Univariate and Multivariate distributions

- **Univariate** distribution is when there is only one random variable, e.g. if instance vectors **x in X** are described by just one feature, or when there is one classification function C(**x**)=Y
- **Multivariate** if many random variables are involved: e.g. **x**:($x_1 \ldots x_d$). Now any feature of **x** can be described by a random variable and we can estimate $P(x_j=v_{jk})$ where v_{jk} $k=1..m_j$ are the possible values for feature j

Probabilistic Classification

- Let Y be the random variable for the class C which takes values $\{y_1, y_2, \ldots y_m\}$ ($|C|=m$ possible classifications for our instances).
- Let X be a random variable describing an instance consisting of a vector of values for d features $<x_1, x_2 \ldots x_d>$, let v_{jk} be a possible value for feature x_j ($j=1 \ldots d$)
- For our classification task, we need to compute the (multivariate) conditional probabilities:

  $P(Y=y_i \mid X=\mathbf{x}(x_1, \ldots x_d))$ for $i=1 \ldots m$

  *(e.g. P(Y=positive/**x**=<color=blue,shape=circle>))*
- E.g. the objective is to classify a new unseen **x** by estimating the probability of each possible classification y_i, **given** the observation of feature values of the instance to be classified
- To estimate $P(Y=y_i \mid X=\mathbf{x})$ we use a learning set D of pairs (**x**,C(**x**))
- **Remember: *i* is index of class values, *j* is index of features, *k* is index of feature values**

How can we compute P(Y=y_i/X=**x**)??

- Example: **x**:(color=red,shape=circle) **C**: y_1=positive, y_2=negative
- We need to compute:

$$P(positive \mid red \wedge circle) = \frac{P(positive \wedge red \wedge circle)}{P(red \wedge circle)}$$

$$P(negative \mid red \wedge circle) = \frac{P(positive \wedge red \wedge circle)}{P(red \wedge circle)}$$

- So we need to compute **joint probabilities**
- The **joint probability distribution** for a set of random INDEPENDENT variables, $x_1,\ldots,x_d$ gives the probability of every combination of values (a *d*-dimensional array with *k* values if all *d* features are discrete with *k* values, and furthermore $\Sigma_{j=1..d} P(x_j=v_{jk})=1$)

Joint probability tables

Pr(shape=circle, color=blue, C=+)

Class=positive

	circle	square
red	0.20	0.02
blue	0.02	0.01

Class=negative

	circle	square
red	0.05	0.30
blue	0.20	0.20

- The probability of all possible conjunctions (= assignments of values to some subset of features) can be estimated from the learning set, by summing the appropriate subset of values from the joint distribution.

$$P(red \wedge circle) = P(red \wedge circle \wedge positive) + P(red \wedge circle \wedge negative) = 0.20 + 0.05 = 0.25$$

- **If all joint probabilities can be estimated, all conditional probabilities can be calculated**.

$$P(positive \mid red \wedge circle) = \frac{P(positive \wedge red \wedge circle)}{P(red \wedge circle)} = \frac{0.20}{0.25} = 0.80$$

Example

- Consider this learning set D of annotated instances:
- $\mathbf{x}^1$ (red, circle), positive
- $\mathbf{x}^2$(red,square),negative
- $\mathbf{x}^3$(blue,circle),positive
- $\mathbf{x}^4$(red, circle),negative
- $\mathbf{x}^5$(red, circle), positive

Pr(positive/red&circle)=Pr(positive&red&circle)/Pr(red&circle) = (2/5)/(3/5)=2/3

Probabilistic Classification (2)

- However, given no other assumptions, this requires tables **giving the probability of each category for each possible instance (combination of feature values) in the instance space,** which is **impossible to accurately estimate** from a reasonably-sized training set D.
- E.g. $P(Y=y_i/X_1=v_{1j},X_2=v_{2j}\ldots X_d=v_{dj})$ for all y_i and v_{kj} Assuming that Y and all X_i are binary valued, and we have d features, we need 2^d entries to specify $P(Y=\text{pos} \mid X=x_k)$ for each of the 2^d possible x_k in D since:
  - $P(Y=\text{neg} \mid X=x) = 1 - P(Y=1 \mid X=x)$
  - Compared to $2^{d+1} - 1$ entries for the joint distribution $P(Y,X_1,X_2\ldots X_d)$

Example: dimension of joint probability tables

- x:(x_1,x_2..x_4), X{ 0,{ 0,1 }Y:{ 0,1 } }(all binary values)
- # of features: d=4, # of values for the class Y: m=2)
- Consider instance $\mathbf{x}^k$:(0,1,0,0)
- Need to estimate Pr(Y=0/(x_1=0, x_2=1, x_3=0, x_4=0))
- If P(Y=0/(0,1,0,0))>P(Y=1/(0,1,0,0))=(1-P(Y=0/(0,1,0,0)))
  then class is 0, else class is 1
- Overall, **2^4 estimates (2^d)** are needed for our probabilistic classifier (the number of possible combinations for feature values)
- *For large d and n this is not feasible, even with large learning sets. Simply, we do not have all the necessary evidence!!*

Summary

- We are given an unclassified instance **x**, which is represented by *m* features (boolean, multi-valued or continuous)
- We have a multi-valued classification function C(**x**) with possible values $y_1 \cdot\cdot y_m$
- Target of the probabilistic classifier is to classify **x** based on: **$y^* = \text{argmax}_{y_i} \Pr(y_i/x)$** (the most likely classification, given the specific combination of feature values in **x**)
- To do so, we need to estimate $\Pr(y_i/\mathbf{x})$ for all y_i, **using evidence provided by previously seen examples** $\langle \mathbf{x}_i, y_i \rangle$ in D
- Quite likely, the specific combination of feature values of **x** has not seen in the learning set: even if values are boolean, there are **2^d possible combinations**! So how do we go about?

Solution: Maximum Likelihood Learning

- Let *M* be a probabilistic formulation (MODEL) of our classification task $p(y_i/\mathbf{x})$
- Suppose that we know exactly the structure of *M:*

  (e.g., M is the softmax function:

$$y_i = softmax_i(\boldsymbol{x}, \boldsymbol{w}) = \frac{e^{w_i x_i}}{\sum e^{w_j x_j}} \quad)$$

  means that we are able to express our conditional probabilities p(y/x) in some precise mathematical form, (e.g. a softmax), but the values of its probabilistic parameters, Θ (e.g. in the softmax example, **the coefficients w_j**), are unknown.
- We can consider our observations, (x,y_i), as "generated" by the (unknown) distribution induced by *M*.

Stated more intuitively: learn the values of model parameters that MAXIMIZE the likelihood of observing the **evidence** provided by the learning set

Maximum Likelihood Estimation (MLE)

- The likelihood of the observed data, given the model parameters Θ, is the conditional probability that the probabilistic model, *M*, with parameters Θ, "generates" the observations x_1 ,…, $x_{|D|}$ of our dataset D

$$L(\Theta)=P(x_1..x_{|D|}|\ \Theta, M)=P(D|\Theta, M)$$

- In MLE we seek the model parameters, Θ, that maximize the likelihood of observing our EVIDENCE E, represented by the learning set D.

Reversing the problem..

- Rather than finding the class value y* that maximizes the conditional probability of any class value given an observed instance **x** (the $\text{argmax}_y P(y/\mathbf{x})$) we try to find **the model parameters Θ that maximize the probability of having observed a particular dataset D** (=our evidence):

$$\text{argmax}_{\Theta}(P(D/\Theta))$$

- $P(D/\Theta)$ is called **likelyhood function** and is also denoted as $L_D(D/\Theta)$

What are these "parameters"?

- The definition of Θ is quite general.
- It can be a set of coefficients in a probabilistic formulation (the coefficients of a softmax function, or the σ of a gaussian function):

  $p(y/\mathbf{x})=f(w_1, w_2,.. w_m,\mathbf{x})$ $\Theta =(w_1, w_2,.. w_m)$
- Or a set of «simpler» probabilities in which I can decompose the original model M:

  $P(y/\mathbf{x})=P(y) \wedge P(x_1=v_1) \wedge P(x_2=v_2) \wedge .. P(x_m=v_m)$

  $\Theta =(P(x_1=v_1), P(x_2=v_2),.. P(x_m=v_m))$

Notation: **p** is a **probability distribution** and is a general formulation since it applies also to continuous random variables. **P** is the **probability** of observing specific **values**.

Statistical Parameter Fitting (general definition for multivariate case)

- Consider instances in dataset D: $x_1, x_2, \ldots, x_{|D|}$ such that:
  - The set of values that C(**x**) can take is known
  - Each **x**i is sampled from the same distribution
  - Each **x**i is sampled independently of the rest

  } i.i.d. Samples

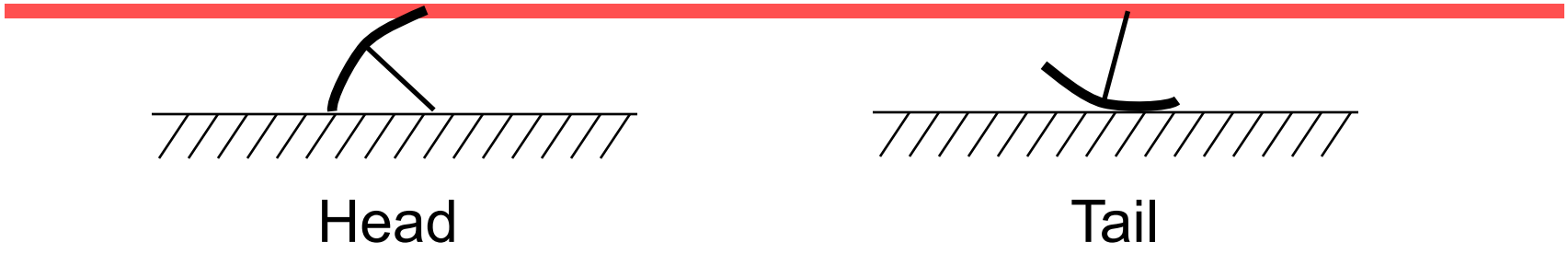
The task is to find a vector of parameters Θ that have generated the given data D. This vector parameter Θ can be used to (probabilistically) predict the class of future data.

Maximum Likelihood Estimation (MLE)

- In MLE we seek the model parameters, Θ, that maximize the likelihood: it is thus an OPTIMIZATION problem (as for all ML algorithms!)
- The MLE principle is applicable in a wide variety of ML problems, from speech recognition, to natural language processing, computational biology, etc.
- We will start with the simplest example:

Estimating the bias of a thumbtack (we got boared with coins!).

Example: Binomial Experiment



- When tossing the thumbtack, it can land in one of two positions: *Head* ***(H)*** or *Tail* ***(T)***
  - We denote by θ the (unknown) probability **P(H)**.

Estimation task:

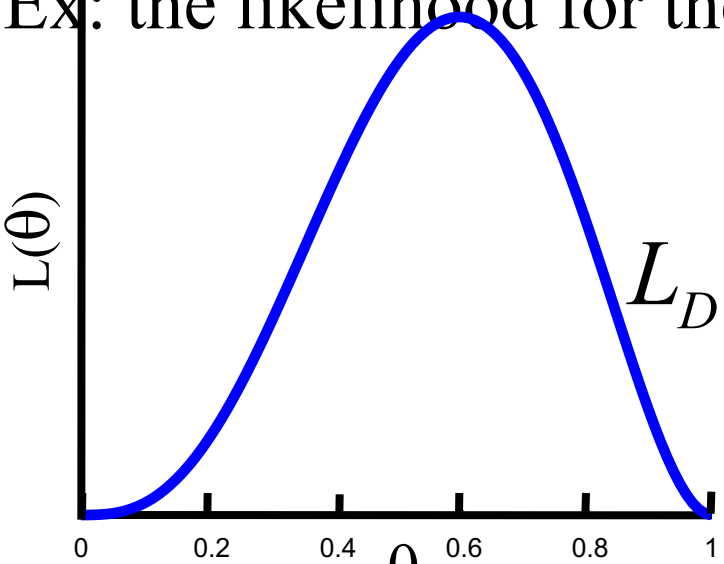
- Given a sequence of **m** toss samples **x1..xm**(our evidence dataset) we want to estimate the probabilities **P(H)=θ** and **P(T) = 1 - θ**
- **θ is the model parameter**
- **In this case the problem is univariate (only one stocastic variable)**

The Likelihood Function

- How good is a particular θ? It depends on **how likely the related model is to generate the observed sequence of outcomes (our data)**

$$L_D(\theta) = P(D \mid \theta) = \prod_{j=1..m} P((xj, yi) \mid \theta)$$

- Where are the observed data? In the thumbtack example, we can toss the thumbtack several times and observe a sequence of results
- Ex: the likelihood for the sequence H, T, T, H, H is:



$$L_D(\theta) = \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta$$

Sufficient Statistics

- To compute the likelihood in the thumbtack example we only require N_H and N_T
  (the number of heads and the number of tails in a sequence of tosses)

$$L_D(\theta) = \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta = \theta^{N_H} \cdot (1-\theta)^{N_T}$$

- N_H and N_T are **sufficient statistics** for the binomial distribution
- A sufficient statistic is a function whose value contains all the information needed to compute any estimate of the parameter

Maximum Likelihood Estimation

MLE Principle:

Given sufficient statistics, choose parameters that maximize the likelihood function (the likelihood of observing data)

- **Θ=α1…αn**
- $\Theta_{optimal}=argmax_{\Theta}(P(D/\Theta))$
- *In our example, we maximize* $L_D(\theta) = P(D \mid \theta) = \theta^{N_H} \cdot (1-\theta)^{N_T}$
- MLE is one of the **most commonly used estimators** in statistics
- One usually maximizes the **log-likelihood function**, defined as $l_D(\theta) = \ln L_D(\theta)$

Example: MLE in Binomial Data (i.e., Y is boolean so we only need P(Y=1/X) since P(Y=0/X)=1-P (Y=1/X))

$$l_D(\theta) = N_H \log\theta + N_T \log(1-\theta)$$

Taking derivative and equating it to 0 we get

$$\frac{N_H}{\theta} = \frac{N_T}{1-\theta} \Rightarrow \hat{\theta} = \frac{N_H}{N_H + N_T}$$

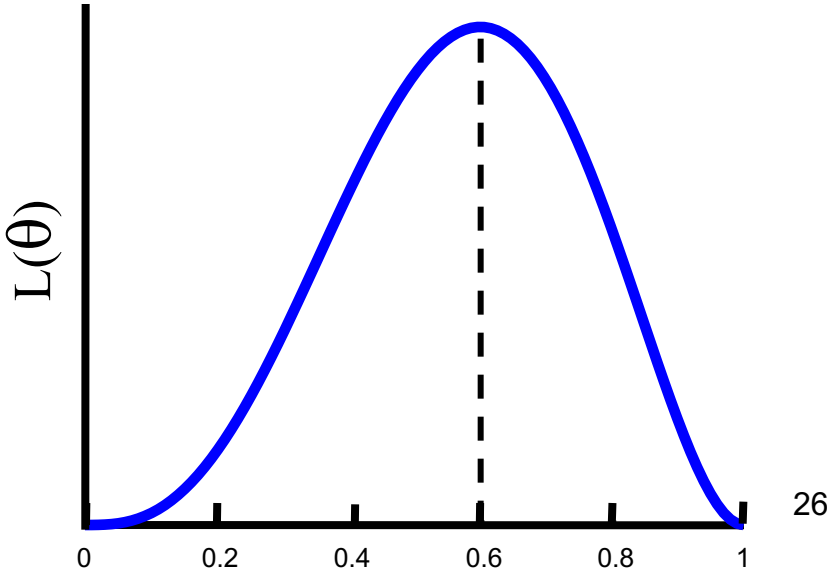
Remember, to maximize minimize a function you need to take the derivative

(which coincides with what one would expect)

Example:

$(N_H, N_T) = (3,2)$

MLE estimate is $3/5 = 0.6$



From Binomial to Multinomial

- Now suppose Y can take the values: $1,2,\ldots,k$
  (For example tossing a die has 6 values, 1..6)
- We want to learn the parameters $\theta_1, \theta_2, \ldots, \theta_n$ *(the vector* Θ of probabilities $\theta_i = P(Y=y_i)$*)*

Sufficient statistics:

- $N_1, N_2, \ldots, N_K$ - the number of times each outcome is observed
- **The optimization pr… log of:**

Remember that θ_i are probabilities, so they must sum 1

$$L_D(\theta) = \prod_{i=1}^{K} \theta_i^{N_k} \quad such\ that: \sum_i \theta_i = 1 \ \ and \ \ \theta_i \geq 0 \ \ \forall i$$

Maximizing log-likelihood with constraints

We consider the log-likelihood (so we get sums rather than products) and we represent the optimization problem with a Lagrangian (again!! Remember SVM)

$$l(\alpha, \Theta) = \sum N^i log\theta_i - \alpha(\sum \theta_i - 1)$$

Remember: $\frac{\partial(K \cdot log(x))}{\partial x} = \frac{K}{x}$

$$\frac{dl(\alpha, \Theta)}{d\theta_i} = 0$$

$$\frac{N^i}{\theta_i} - \alpha = 0 \implies \theta_i = \frac{N^i}{\alpha}$$

$$\sum \frac{N^i}{\alpha} = 1 \Rightarrow \alpha = \sum N^i$$

$$\hat{\theta}_i = \frac{Ni}{\sum_j Nj}$$

Another example: proteine sequences

- Let $x_1 x_2 \ldots . x_n$ be a protein sequence
- We want to learn the parameters $\theta_1, \theta_2, \ldots, \theta_{20}$ corresponding to the probabilities of the 20 amino acids
- $N_1, N_2, \ldots, N_{20}$ - the number of times each amino acid is observed in the sequence

Likelihood function: $$L_D(q) = \prod_{i=1}^{20} \theta_i^{N_i}$$

MLE: $$\theta_i = \frac{N_i}{n} \quad n = \sum_{i=1}^{20} N_i$$

NAIVE BAYES CLASSIFIER: A ML ALGORITHM BASED ON MLE PRINCIPLE

Naive bayes

- We are given an evidence represented by an annotated learning set D of classified instances $(\mathbf{x}_i, y_i)$.
- We would like to estimate the probabilities $p(Y=y_i/\mathbf{x})$ for unobserved instances, given the evidence D
- In probability calculus, often estimating P(a/b) is easier that estimating P(b/a) so we need the Bayes theorem to **invert** conditional probabilities

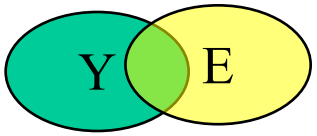
Bayes Theorem

- Y=classification
- E= evidence of data

$$P(Y \mid E) = \frac{P(E \mid Y)P(Y)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(Y \mid E) = \frac{P(Y \wedge E)}{P(E)} \qquad \text{(Def. cond. prob.)}$$



$$P(E \mid Y) = \frac{P(Y \wedge E)}{P(Y)} \qquad \text{(Def. cond. prob.)}$$

$$P(Y \wedge E) = P(E \mid Y)P(Y)$$

QED: $$P(Y \mid E) = \frac{P(E \mid Y)P(Y)}{P(E)}$$

Naïve Bayes Model (1)

For each classification value y_i we first apply Bayes:

$$P(Y = y_i \mid X = x\ ) = \frac{P(Y = y_i)P(X = x \mid Y = y_i)}{P(X = x\ )}$$

- P(Y=yi) and P(X=$\boldsymbol{x}$) are called **priors** and can be estimated from the learning set D since categories are **complete** and **disjoint**

Example (estimation of priors)

- Given the following dataset:

  X1 (red, circle), positive

  X2(red,square),negative

  X3(blue,circle),positive

  X4(red, circle),negative

  X5(red, circle), positive

- We have:
  - P(positive)=3/5 ; P(negative)=1-P(positive)
  - P(red, circle)=3/5
  - P(red,square)=1/5
  - P(blue,circle)=1/5

Naive Bayes Model (2)

$$P(Y = y_i \mid X = x) = \frac{P(Y = y_i)P(X_1 = v_1, X_2 = v_2, \cdots X_d = v_d \mid Y = y_i)}{P(X = x)} =$$

$$P(Y = y_i)\frac{\prod_{j=1}^{d} P(X_j = v_{jk} \mid Y = y_i)}{P(X = x)}$$

The basic assumption of NB is that feature values v_{jk} of different features X_j are **statistically independent**.

v_{jk} is the k-th value of feature j where j=1,2..d

and k=1…K_j (if binary features, k=0 or 1)

e.g. P(x(color=blue, shape=circle, dimension=big))= P(color=blue)P(shape=circle)P(dimension=big)

and furthermore

$$\sum P(X_j = v_{jk}/Y = y_i) = 1$$

Estimating model parameters Θ

- We have two sets of parameters $\boldsymbol{\theta^1}$ and $\boldsymbol{\theta^2}$:
- $\boldsymbol{\theta^1}$: $P(Y=y_i)$ for all i, and
- $\boldsymbol{\theta^2}$: $P(X_j = v_{jk} \mid Y = y_i)$ **for all i,j and k**
- The log-likelihood function is then:

$$L(\Theta) = \sum N^i log\theta_i^1 + \sum \prod N^{jki} log\theta_{jki}^2$$

- Where N_i is the number of times we see class y_i in dataset, and $\mathbf{N^{jki}}$ is the number of times we see an instance with $x_k=v_{jk}$ and $Y=y_i$ (k-th feature is equal to v_{ik} and class is y_i)

Remember: *i* is index of class values, *j* is index of features, k is index of values

Estimating model parameters (2)

The MLE problem is therefore: **maximize** l(θ) subject to:

$$\sum \theta_i^1 = 1 \quad \theta_i^1 \geq 0; \quad \sum_k \theta_{jki}^2 = 1 \quad \theta_{jki}^2 \geq 0$$

Parameters are probabilities, so they must obey probability rules

$$l(\Theta) = \sum N^i log\theta_i^1 + \sum\prod N^{jki} log\theta_{jki}^2 - \alpha \sum_{i=1..|C|} (\theta_i^1 - 1) - \beta \sum_{k=1...K} (\theta_{jki}^2 - 1)$$

Since we have logarithms:

$$\sum\prod N^{jki} log\theta_{jki}^2 = \sum\sum N^{jki} log\theta_{jki}^2$$

Estimating model parameters (3)

To maximize the likelyhood we need to find values that take to zero the derivative

$$\frac{\partial L(\alpha,\beta,\Theta)}{\partial \theta^2_{jki}} = \frac{N^{jki}}{\theta^2_{jki}} - \beta = 0 \rightarrow \theta^2_{jki} = \frac{N^{jki}}{\beta}$$

For all j,k,i

$$and\ since \sum_{k=1..K} \theta^2_{jki} = 1$$

$$we\ obtain\ \theta^2_{jki} = \frac{N^{jki}}{\sum_k \theta^2_{jki}} = \frac{N^{jki}}{N^i}$$

I.e. the θ^2 can be estimated as the ratio between the number of times feature X_j takes value v_{jk} when class is $Y=y_i$, and the total number of examples in D for which $Y=y_i$

How do we predict the category of an unseen instance with Naive Bayes?

$$P(Y = y_i | X = x_k) = P(Y = y_i) \prod_{j=1..d} P(v_{jk} | Y = y_i) / \prod_{j=1..d} P(v_{jk}) =$$

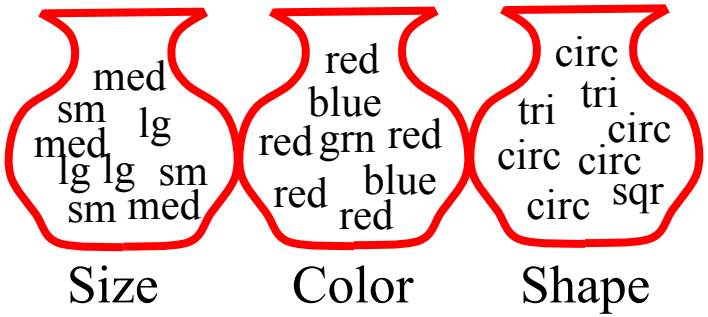
$$\theta_i^1 \prod_{j=1..d} \theta_{ikj}^2 / \prod_{j=1..d} P(v_{jk})$$

$$y_i = \underset{k}{\text{argmax}} (\theta_i^1 \prod_{j=1..d} \theta_{ikj}^2)$$

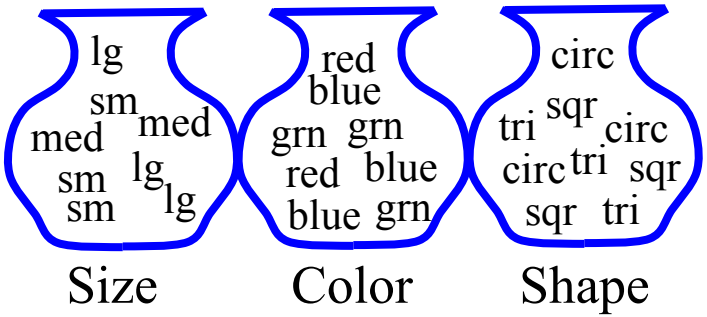
Note that since the denominator is common to all conditional probabilities, it does not affect the ranking.
No need to compute it!

Naïve Bayes Generative Model Example

K=3,|C|=2,d=3



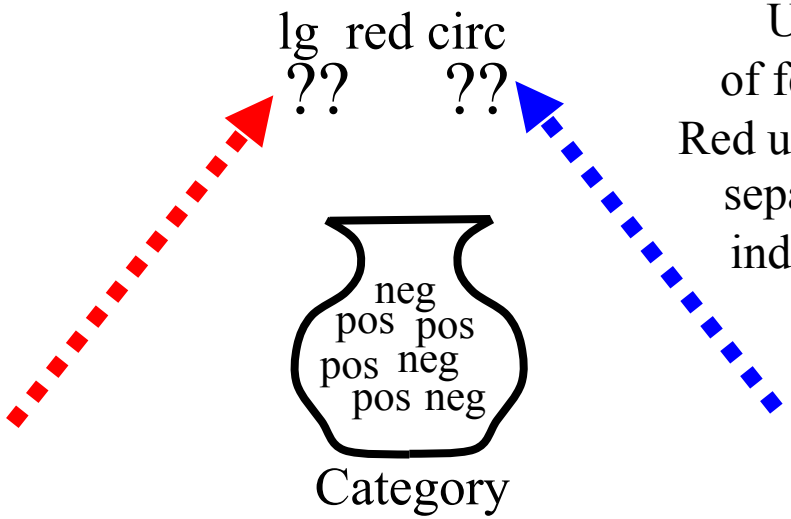
Positive



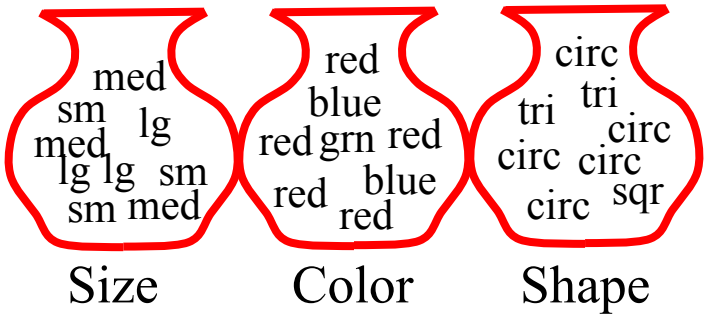
Negative

Naïve Bayes Inference Problem

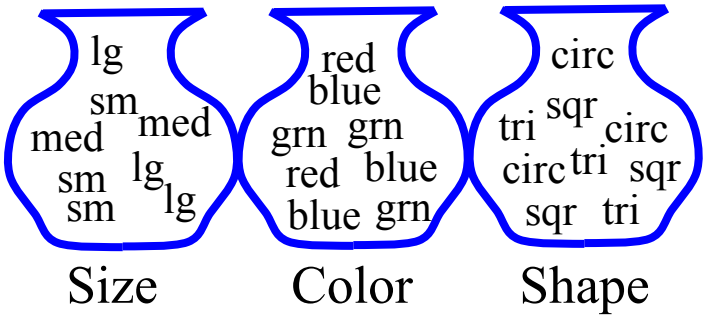
Let's say we have a new unclassified instance x:<large, red, circle>.
We need to estimate, on the learning set, the probability of extracting lg, red, circ from the red or blue urns. Whichone is higher?



Urns represent occurrences of feature values in learning set. Red urns are positive, blue negative separate urns indicate statistical independence of feature values



Positive



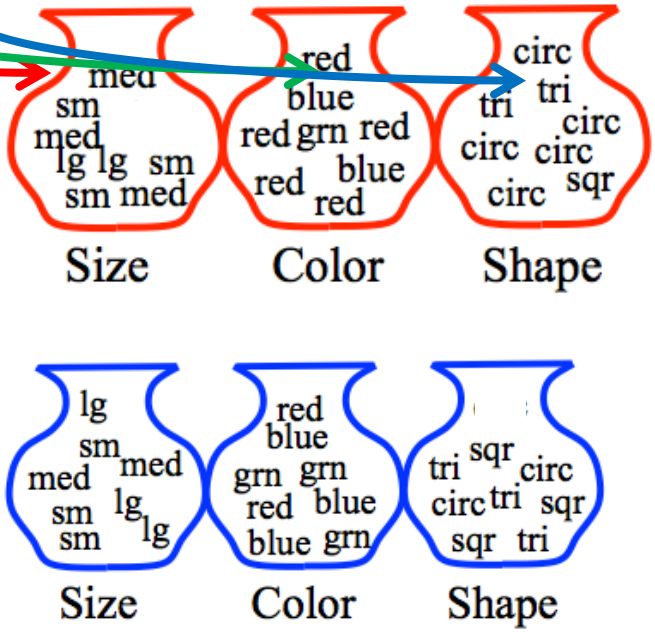
Negative

HOW?

We fill urns using the available dataset D

1	med	red	circ	pos
2	sm	blue	tri	pos
3	med	red	tri	pos
4	lg	grn	circ	pos
5	lg	red	circ	pos
6	sm	blue	circ	pos
7	sm	red	sqr	pos
8	med	red	circ	pos
9	lg	red	sqr	neg
10	sm	blue	tri	neg
11	med	grn	circ	neg
12	med	grn	tri	neg
13	lg	red	circ	neg
14	sm	blue	sqr	neg
15	sm	blue	tri	neg
16	lg	grn	sqr	neg

Note that because of independence hypothesis, the specific combination of feature values does not matter!

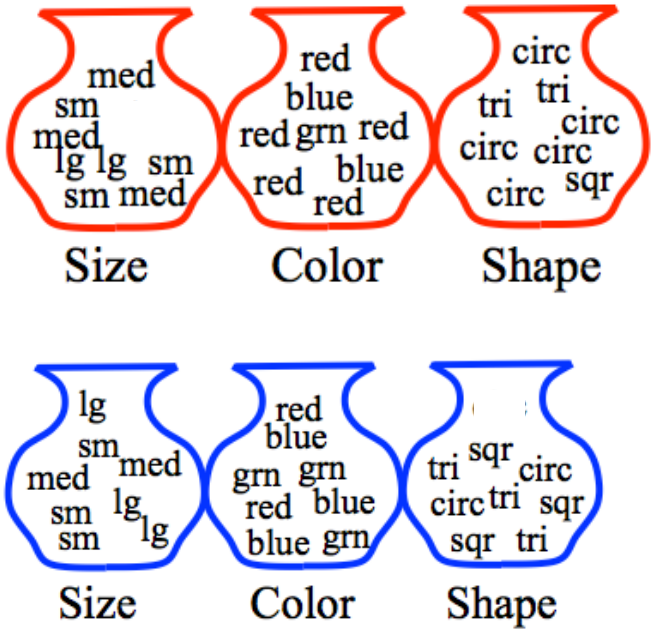


Now we can compute parameters

$$\theta_{size,small,positive} = P(size = small/C = positive)$$

Probability	Y=positive	Y=negative
P(*Y*)	0.5	0.5
P(small | *Y*)	3/8	3/8
P(medium | *Y*)	3/8	2/8
P(large | *Y*)	2/8	3/8
P(red | *Y*)	5/8	2/8
P(blue | *Y*)	2/8	3/8
P(green | *Y*)	1/8	3/8
P(square | *Y*)	1/8	3/8
P(triangle | *Y*)	2/8	3/8
P(circle | *Y*)	5/8	2/8

Shape Color Size



We have 3 small out of 8 instances in red "size" urn
then P(size=small/pos)=3/8=0,375 (round 4)

Training set D (evidence)

Using parameters, we can classify unseen instances

Probability	Y=positive	Y=negative
P(*Y*)	0.5	0.5
P(medium | *Y*)	3/8	2/8
P(red | *Y*)	5/8	2/8
P(circle | *Y*)	5/8	2/8

$$y_i = \underset{k}{\operatorname{argmax}}(\theta_i^1 \prod_{j=1..d} \theta_{ikj}^2)$$

Test Instance:
X:<medium ,red, circle>

P(positive | *X*) = P(positive)*P(medium | positive)*P(red | positive)*P(circle | positive)/P(X)

0.5 * 3/8 * 5/8 * 5/8

= 0,073

P(negative | *X*) = P(negative)*P(medium | negative)*P(red | negative)*P(circle | negative) / P(*X*)

0.5 * 2/8 * 2/8 * 2/8

=0.0078

P(positive/X)>P(negative/X) ➔ positive

Note: sum is not 1 since we ignore the denominator of original formulation $\prod P(v_{jk})$

Naive summary

Classify any new datum instance $\mathbf{x_k}=(x_1,\ldots x_n)$ *as:*

$$y_{Naive\ Bayes} = \underset{i}{\text{argmax}}\, P(y_i)P(\mathbf{x}\mid y_i) \rightarrow \underset{i}{\text{argmax}}\, P(y_i)\prod_{j=1..d} P(v_{jk}\mid y_i)$$

- To do this based on training examples, estimate the parameters from the training examples in D:
  - For each target value of the classification variable (hypothesis) y_i

$$\hat{P}(Y = y_j) := \textbf{estimate}\ P(y_i)$$

  - For each attribute value v_{jk} of each datum instance

$$\hat{\boldsymbol{P}}(\boldsymbol{X_j = v_{jk}} \mid \boldsymbol{Y = yi}) := \textbf{estimate}\ \boldsymbol{P(v_{jk}\mid y_i)}$$

Estimating Probabilities

- Normally, as in previous example, probabilities are estimated based on observed frequencies in the training data.
- If D contains N_i examples in category y_i, and N_{jki} of these N_i examples have the k-th value for feature X_j, v_{jk}, then:

$$P(X_j = v_{jk} \mid Y = y_i) = \frac{N_{jki}}{N_i}$$

- However, estimating such probabilities from small training sets is **error-prone**. (**bias** in the estimate, as we have seen)
- If -due only to chance- a rare feature value, X_j=v_{jk} is never observed in the training data, then $P(X_j=v_{jk} \mid Y=y_i) = 0$.
- If X_j=v_{jk} then occurs in a test instance, X, the result is that $\forall y_k$: $P(X \mid Y=y_i) = 0$ and $\forall y_i$: $P(Y=y_i \mid X) = 0$ (since individual probability estimates are multiplied)

Probability Estimation Example

Ex	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negitive
4	large	blue	circle	negitive

Probability	positive	negative
P(*Y*)	0.5	0.5
P(small | *Y*)	0.5	0.5
P(medium | *Y*)	0.0	0.0
P(large | *Y*)	0.5	0.5
P(red | *Y*)	1.0	0.5
P(blue | *Y*)	0.0	0.5
P(green | *Y*)	0.0	0.0
P(square | *Y*)	0.0	0.0
P(triangle | *Y*)	0.0	0.5
P(circle | *Y*)	1.0	0.5

Test Instance *X*:
<medium, red, circle>

P(positive | *X*) = 0.5 * 0.0 * 1.0 * 1.0 / P(X) = 0

P(negative | *X*) = 0.5 * 0.0 * 0.5 * 0.5 / P(X) = 0

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
- *Laplace smoothing* using an *m*-estimate assumes that each feature is given a prior probability, *p*, that is assumed to have been previously observed in a "virtual" sample of size *m*.

$$P(Xj = v_{jk} \mid Y = y_i) = \frac{N_{jki} + mp}{N_i + m}$$

- For binary features, *p* is simply assumed to be 0.5, while it can be set to 1/k for k-valued features.

Laplace Smothing Example

- Assume training set contains 10 positive examples, and feature "size" has 3 values, but 1 (medium) is not observed in D:
  - 4: small
  - 0: medium
  - 6: large
- Estimate parameters as follows (if we set m=1, p=1/3)
  - P(small | positive) = (4 + 1/3) / (10 + 1) = 0.394
  - P(medium | positive) = (0 + 1/3) / (10 + 1) = 0.03
  - P(large | positive) = (6 + 1/3) / (10 + 1) = 0.576
  - P(small or medium or large | positive) = 1.0

Continuous Attributes

- If X_j is a **continuous** feature rather than a discrete one, need another way to calculate $P(X_j = vij | Y = y_i)$.
- Assume that X_j has a Gaussian distribution whose mean and variance depends on Y.
- During training, for each combination of a continuous feature X_j and a class value for Y, y_i, estimate a **mean**, μ_{ji}, and standard deviation $\boldsymbol{\sigma_{ji}}$ based on the observed values of feature X_j in class y_i in the training data. μ_{ji} is the mean value of Xj observed in instances for which $Y = y_i$ in D
- **During testing**, estimate $P(X_j = v \mid Y = y_i)$ for a given example, using the Gaussian distribution defined by μ_{ji} and σ_{ji}.

$$\boldsymbol{P(Xj = v \in \Re \mid Y = y_i) = \frac{1}{\sigma_{ji}\sqrt{2\pi}} \exp\left(\frac{-(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)}$$

Comments on Naïve Bayes

- Tends to work well despite strong assumption of conditional independence.
- Experiments show it to be quite competitive with other classification methods on standard UCI datasets.
- Although it does not produce accurate probability estimates when its independence assumptions are violated, it may still pick the correct maximum-probability class in many cases.
  - Able to learn conjunctive concepts in any case
- Does not perform any search of the hypothesis space. Directly constructs a hypothesis from parameter estimates that are easily calculated from the training data.
  - Strong bias
- Not guaranteed consistency with training data.
- Typically handles noise well since it does not even focus on completely fitting the training data.