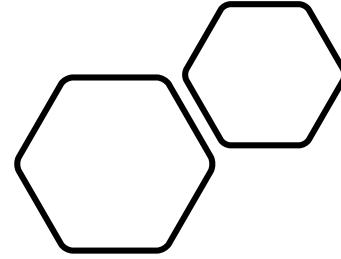




Probabilistic ML algorithms

MLE, MAP and Naïve Bayes Model

Basic probability notions you need [link](#)



Axioms of Probability Theory

- All probabilities are between 0 and 1

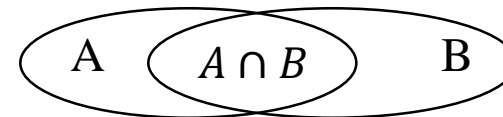
$$0 \leq P(A) \leq 1$$

- The true proposition has probability 1, false has probability 0.

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

$$P(A \text{ OR } B) = P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

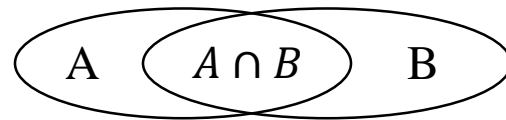
- Disjunctive probabilities:



Conditional Probability

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

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$



Statistical Independence

- *Random variables A and B are **independent** if and only if:*

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

- Therefore, if A and B are **independent**:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = P(A)$$

$$P(A \cap B) = P(A, B) = P(A)P(B)$$

Bayes Formula

- $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, **53:370-418**

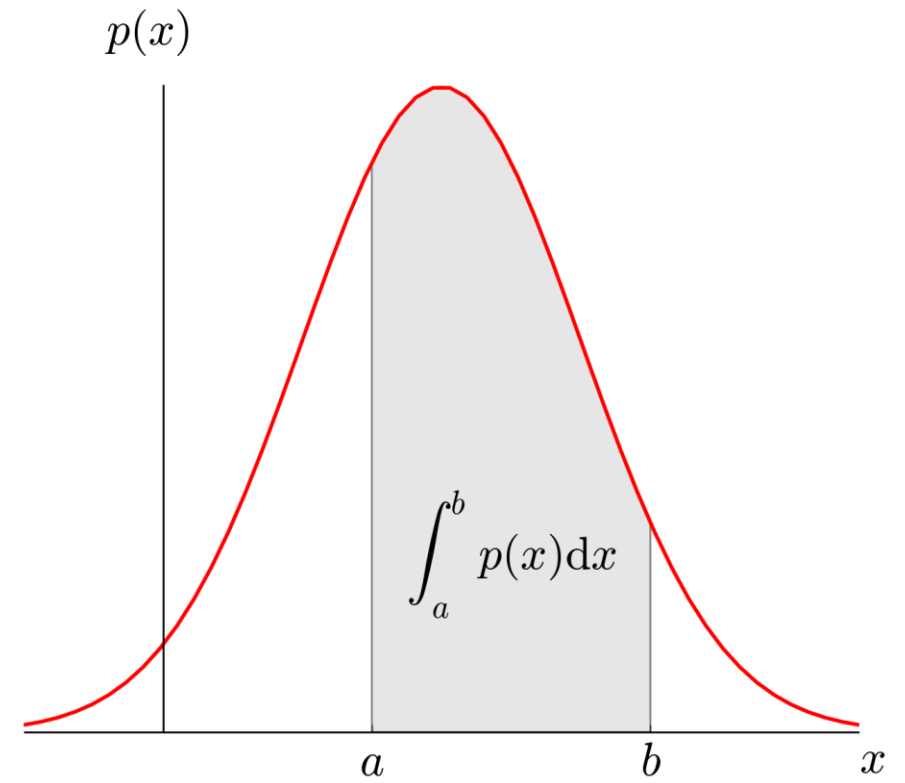
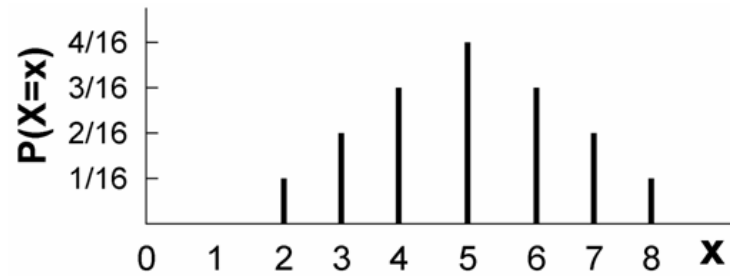


Univariate and Multivariate distributions

- **Univariate** distribution is when there is only **one random variable** (e.g. if instance vectors \mathbf{x} in X are described by just one feature, or when there is one classification function $C(\mathbf{x}) = Y$)
 - The feature or classification space is one-dimensional (for example real numbers, list of labels, ordered labels or binary)
- **Multivariate** if many random variables are involved (e.g. $\mathbf{x}: (X_1, \dots, X_d)$ or $C(\mathbf{x}): (Y_1, Y_2, \dots, Y_N)$ - multiple features and multiple classes. Now, any feature j (or any class i) can be described by a random variable X_j (or Y_i).
- If a random variable X_j is discrete, we can estimate $P(X_j = x_{jk})$, the probability mass function PMF, where x_{jk} with $k = 1, \dots, d_j$ are the d_j possible values for feature j . (e.g., binomial multinomial)
- If X_j is continuous, then we can estimate $p(X_j)$, the Probability Density Function PDF of its values (e.g., a Gaussian).

Probability mass and probability density

x	$P(x)$
2	1/16
3	2/16
4	3/16
5	4/16
6	3/16
7	2/16
8	1/16



Probabilistic ML

- We reformulate the learning/predictions problems in probabilistic terms
- Input instances x are treated as (univariate or multivariate) random variables, output y is a (univariate or multivariate) random variable, the model to be learned can be described in probabilistic terms, e.g., $P(Y=y|X=x)$ the conditional probability of y given the observation of x .

Mapping the terminology

Symbolic (table)	Geometric (vectors)	Probabilistic interpretation	Variables or values?
X : feature space	feature space	(multivariate) random variables	Set of variables
\mathbf{x} : record, instance	feature vector	Observation, trial, sample, random vector	Set of values
D : dataset	Set of feature vectors	A sample of X	Set of valued random vectors
X_i : i-th feature	i-th dimension of feature space	i-th random variable of the multivariate distribution	variable
x_i : a value for feature X_i	value of i-th coordinate of a feature vector	A value that can be assumed by the random variable X_i (a <i>sample/observation</i> of X_i)	value

In general, in statistics, lowercase indicate observations (values) and uppercase random variables

Probabilistic formulation of ML

Many machine learning problems can be formulated in probabilistic terms.

- The target of a ML classifier is to learn a classification function (a model M) from data D
 $f(\mathbf{x}): \mathbf{x} \rightarrow y$ (or \mathbf{y} if output is also a vector)
 - Given an unseen instance \mathbf{x} , assign a category label y to this instance using the learned function $f(\mathbf{x})$.
 - In a probabilistic formulation, $f(\mathbf{x})$ is a probability function, e.g., $P(y|\mathbf{x})$. What is the probability to observe $Y=y$ for the output random variable, given that $X=\mathbf{x}$?

Example:
Handwriting Recognition

Is this “a” or “9”??

The classifier may return non-zero probabilities for all options



Probabilistic formulation

Two alternative probabilistic formulations of a ML classifier are:

GENERATIVE models: define a **function** $L(D; \theta)$, where θ are the parameters of a model M_θ and «;» stands for a joint probability.

- $L(D; \theta)$ is the *likelihood* of θ given the observation of D . The likelihood is used to estimate the parameters θ
- Use θ to specify the model M_θ that explains our data (and predict new ones)

DISCRIMINATIVE models: Use the dataset D to estimate the parameters θ of a **probability function** $P(Y = y | x)$ which is the *conditional probability* of class label y given the observation of x .

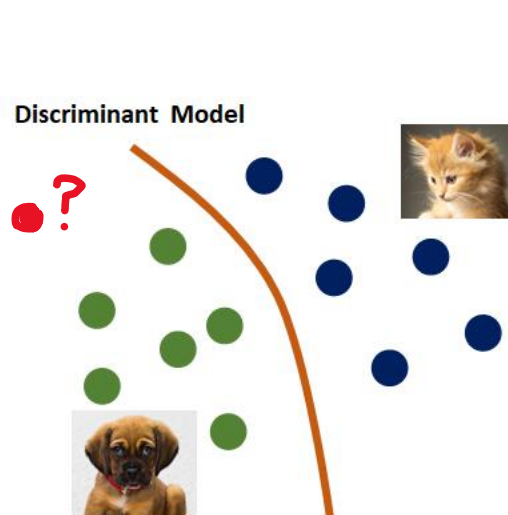


Understanding the difference

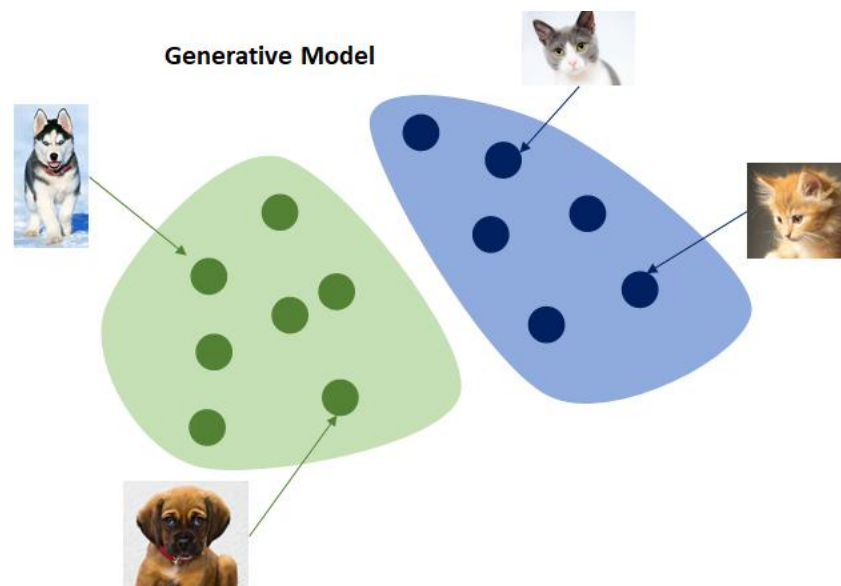
- A generative model implies learning **the distribution of the data itself** and tells you **how likely a given example (instance) is**. For example, we can postulate that our data are generated by a Gaussian distribution, G_{θ} with unknown parameters σ and μ . Learning the parameters allows us to predict the probability of observing a particular instance $X=x$.
- A discriminative model **ignores the question of whether a given instance x is likely**, and just tells you how likely a **label** (category) is, GIVEN the observation of the instance ($P(Y=y_i | X=x)$).
- [Here](#) is a nice paper that explains in detail the difference

Discriminative/generative

$$\text{Argmax}(P(Y=\text{dog}/X=x), P(Y=\text{cat}/X=x))$$



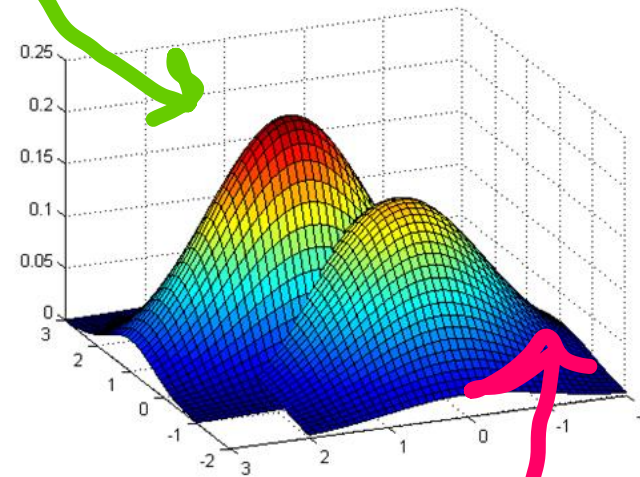
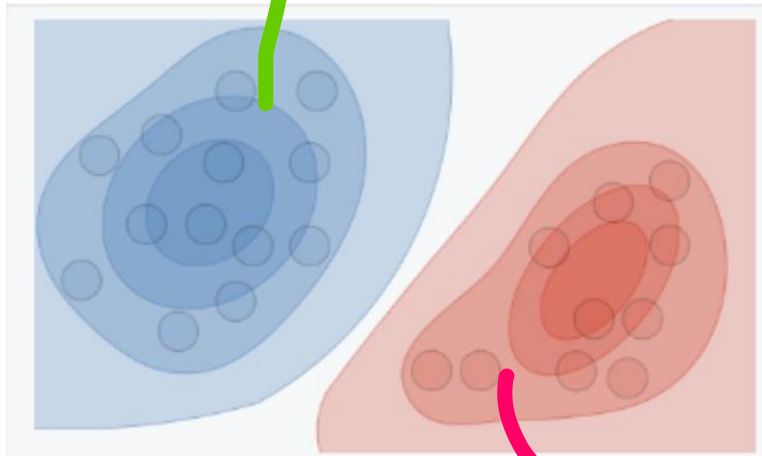
The model just predicts the probability that a new instance is a cat or dog, only based on whether it is placed wrt the decision boundary



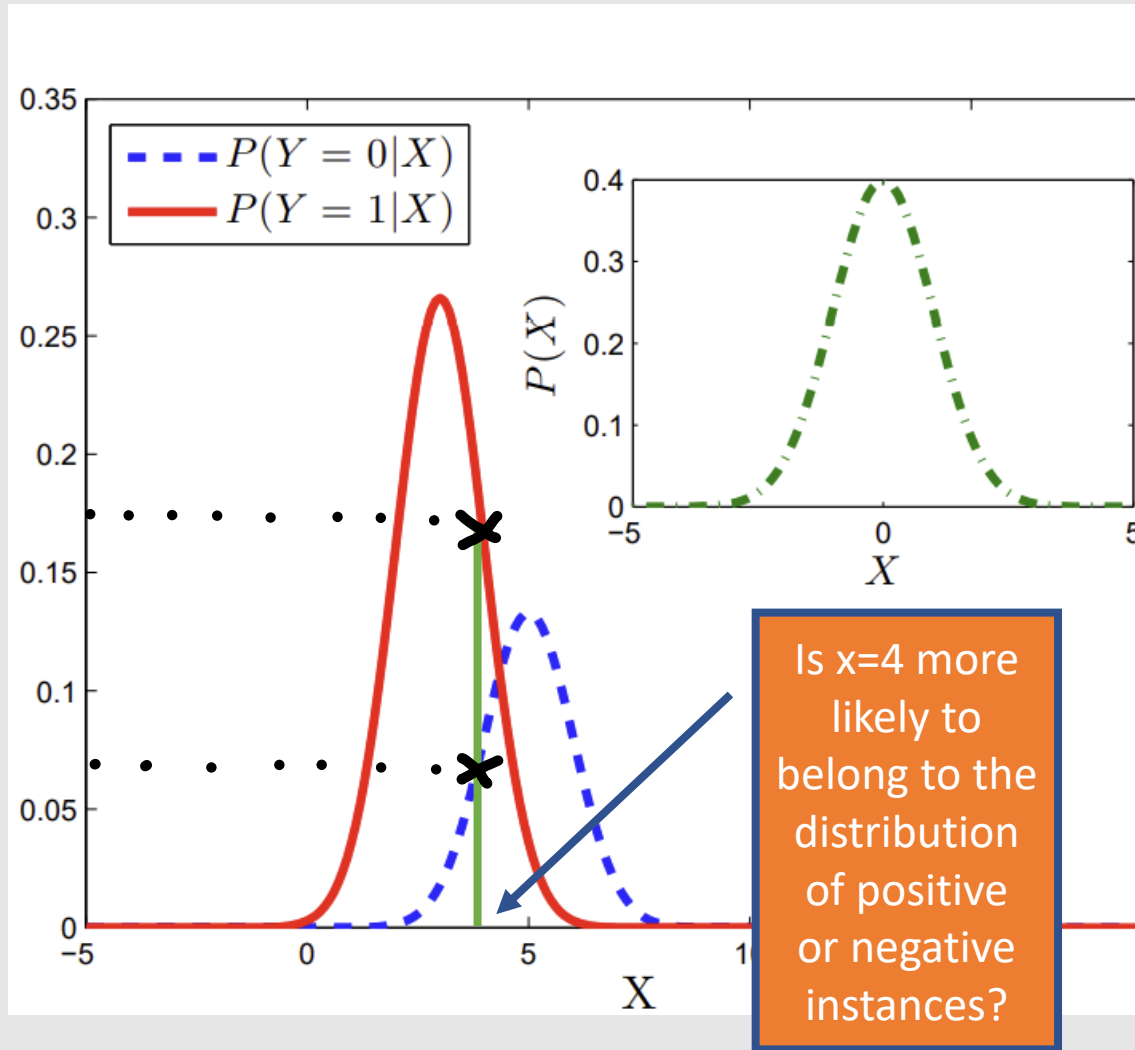
The model «interprets» the input data (**provides a probabilistic model that may have generated the observed data**) and it is able to say how likely a new instance is to belong to the probability distribution – e.g., of «cats» or of «dogs» -

What is this «model» of the data?

An additional dimension is added, a PMF (or PDF) – the «Model» – that «interprets» the data in probabilistic terms (e.g, in this two-dimensional space: $P(X_1, X_2; \text{Positive})$ and $P(X_1, X_2; \text{negative})$)

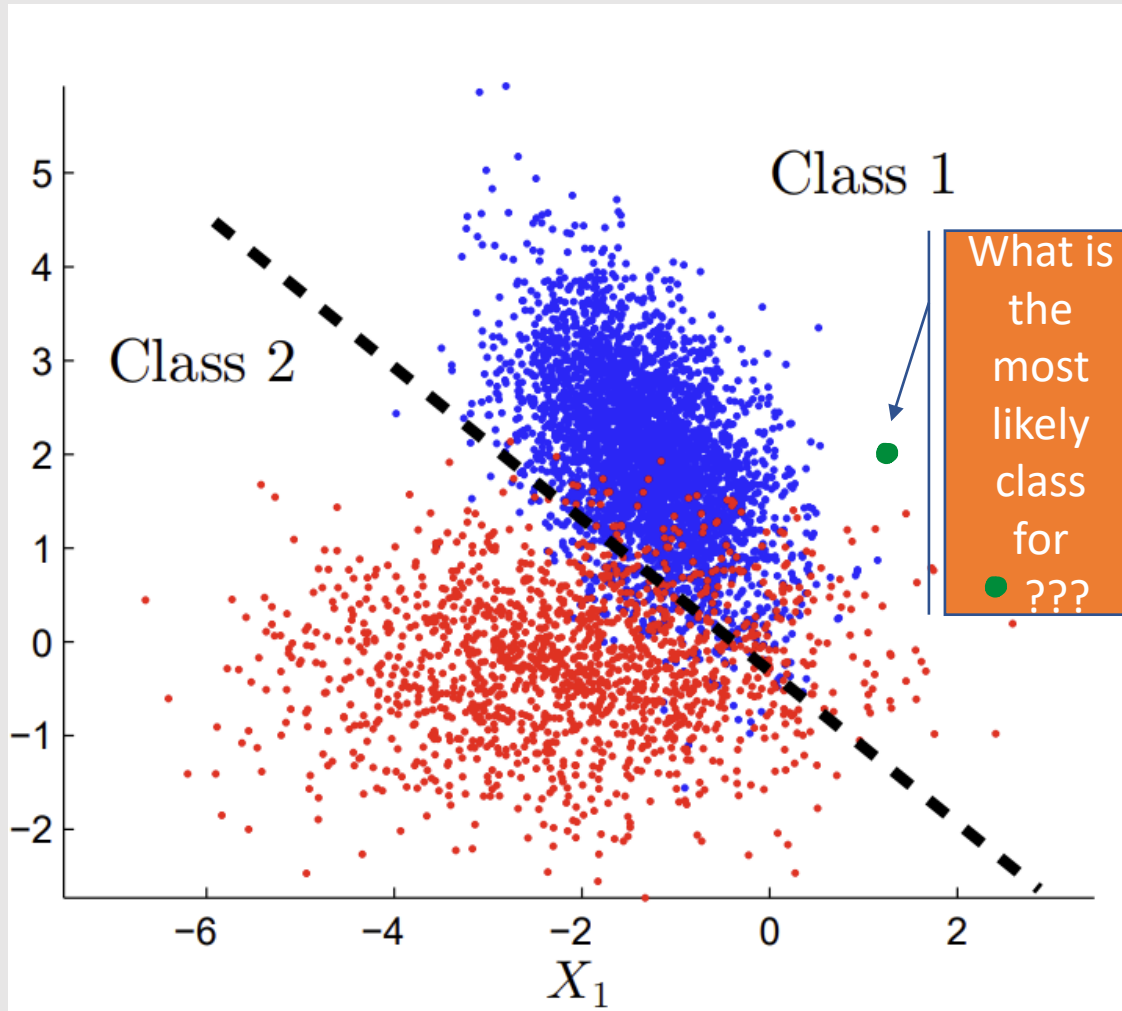


For example: x and y are the body-mass-index and cholesterol value of a sample population D of diabetic (positive, $D+$) and non-diabetic (negative, $D-$) patients. We are assuming a gaussian distribution of these values, with unknown σ, μ



Generative: find parameters that “explain” all data

For example, if we assume that the values of a continuous random variable X follow a Gaussian distribution, we can estimate the parameters of this distribution for the positive and negative examples $(\sigma_+, \mu_+, \sigma_-, \mu_-)$ in D



Discriminative:
finds parameters that help to predict relevant data.

Note the analogy between the difference between classifiers and regressors, and discriminative and generative models.

Generative models formulation

- Let M be a probabilistic formulation (**model**) of a classification task , our training task is to “fit” the model with respect to our dataset D , like for algebraic models. In probabilistic terms, our task is to model the **joint** probability $P(D; M_{\theta})$ where θ are the model parameters («*what is the most likely set of parameters that may have generated D ??*» The training set D now is seen as an «evidence» generated by some unknown distribution)
- Suppose that we know exactly the “structure” of M (e.g. a softmax function or a gaussian), this means that we can express our model in some precise probabilistic form, but the values of its **probabilistic parameters θ** (e.g. the weights of softmax or the σ of a Gaussian) are **unknown**. The problem involves finding

M_{θ} that “best explains” the training data D

Intuitively: To learn the parameters of the model we have to maximize a probability function of observing the **evidence** provided by the training set D .

Goal: After observing several examples $x_1.. x_N$ (a so-called “sufficient statistics”) estimate the model parameters, θ , that may have generated the observed data.

Maximum Likelihood Estimation

In statistics, **Maximum Likelihood Estimation (MLE)** is a method for estimating the parameters of a statistical model M given observations D , by finding the parameter values that maximize the likelihood of observing D . So, again, it is an optimization problem.

What is the «likelihood»?



Likelihood Function for Random Variables

- The **likelihood function** expresses the **joint** probability (or probability density) of a sample data D , given a set of model parameters values.
- $L_D(\theta) = f(D; \theta) = f(x_1, x_2, \dots, x_{|D|}; \theta)$
- f is a function of the parameters, that express:
 - **Discrete Case:** probability **mass** function (PMF) (e.g. Bernoulli, Geometric, Binomial distributions)
 - **Continuous Case:** a probability **density** function (PDF) (Gaussian, Poisson, Exponential, Softmax..)
- What does *likelihood* mean and how is “likelihood” different than “probability”?
 - **Discrete distributions:** likelihood is a synonym for the **joint probability** of your data D
 - **Continuous Distribution:** likelihood refers to the **joint probability density** of your data D

Likelihood Function What are these “parameters”?

- The definition of θ is quite general.
- A set of parameters $\{\Theta_1, \Theta_2, \dots, \Theta_m\}$
 - **Discrete Case:** the parameters are probabilities $P(X_i=x_i)$, e.g. $P(\text{Color}=\text{red})$ or *conditioned probabilities* $P(Y=y|X=\mathbf{x})$ (e.g. $P(Y=\text{yes}|Color=\text{red})$)
 - **Continuous Case:** the parameters are the coefficients in a probabilistic formulation (e.g., if M is the softmax function (a.k.o. exponential) : $y_i = \text{softmax}_i(\mathbf{x}, W) = \frac{e^{x_i w_i}}{\sum_j e^{x_j w_j}}$) parameters θ are the w_j ; if M is a gaussian, θ are σ and μ)

Statistical Parameter Fitting

(general definition for multivariate case)

- Consider instances in dataset $D: \langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|D|} \rangle$
 - such that:
 - The set of values that $y = Y(\mathbf{x})$ can take is known (to simplify, let's Y be univariate and either binomial or multinomial)
 - Each \mathbf{x}_i is sampled from the same distribution
 - Each \mathbf{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 Principle:

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

$$\Theta_{optimal} = \operatorname{argmax}_{\Theta} (L(D; \Theta)) = \operatorname{argmax}_{\Theta} (P(D; \Theta))$$

- A “sufficient statistics” is a function whose value contains all the information needed to compute any estimate of the parameters
- In MLE we seek the model parameters Θ that **maximize the likelihood** ($\operatorname{argmax}_{\theta} (L)$), and the likelihood is expressed as a conditional probability. It is thus an **optimization** problem (as for all ML algorithms!)

Maximum Likelihood Estimation

Finding MLE's involves techniques of differential calculus (as usual). To maximize $L_D(\boldsymbol{\theta})$ with respect to each θ_i in $\boldsymbol{\theta}$:

1. First calculate the (partial) derivative of $L_D(\boldsymbol{\theta})$ with respect to all θ_i ,
2. Set the derivative equal to zero, and
3. Solve the resulting equation for θ_i .

- These computations can often be simplified by maximizing the *log-likelihood function*:

$$l_D(\boldsymbol{\theta}) = \ln(L_D(\boldsymbol{\theta}))$$

$$l_D(\boldsymbol{\theta}) = \sum_{i=1}^{|D|} \ln(P(\mathbf{x}_i; \boldsymbol{\theta})) \quad (\text{in the i. i. d. case})$$

- The natural log is an increasing function, maximizing the log-likelihood **is the same as maximizing the likelihood**. The loglikelihood often has a much simpler form than the likelihood and is usually easier to differentiate.

Likelihood Function for **discrete** variables

- First, we deal with **discrete distributions of random variables** (so we can directly use the classical probability notation P instead of f).

$$L_D(\theta) = P(D; \theta) = P(x_1, x_2, \dots, x_{|D|}; \theta)$$

- **Note: Categorical distributions like Bernoulli, Binomial and Multinomial are discrete distributions!!!**
- For different values of the parameters, the likelihood of our data will be different. For that reason we write likelihood as a function of our parameters (θ)
- Furthermore, if the random variables (instances in X) are **identical and independently distributed (i.i.d.)**, we can apply the independent and identically distributed assumption to the likelihood:

$$L_D(\theta) = P(D; \theta) = \prod_{i=1}^{|D|} P(x_i; \theta)$$

*The joint distribution $P(x_1, x_2, \dots, x_{|D|}; \theta)$
can be expressed as the product of independent joint probabilities of observing each instance!*

MLE for Binomial Distribution (univariate)

- Suppose that $(x_1, x_2, \dots, x_{|D|})$ represents $n=|D|$ i.i.d. samples of a binary random variable Y , that follows a **binomial** distribution. Let θ be the probability that a sample x_j is equal to 1, and $(1-\theta)$ the probability of being 0. (Note: Y is here univariate, therefore is not in bold)
- Note: The binomial distribution here could model **both the distribution of class labels y_j AND the distribution of feature values x_j in case of discrete features**. The purpose is just to see how MLE applies to these types of distributions.
- Consider the PMF (probability mass function) of a Binomial distribution where r is the number of times we observe $x=1$ in n trials (e.g., the PMF modeling the probability of observing r “1s” in the n samples in D):

$$P(D; \theta) = \frac{n!}{r! (n-r)!} \theta^r (1-\theta)^{n-r}$$

So, if the MODEL is a Binomial distribution, the only parameter is θ the probability $P(x=1)$

MLE for Binomial Distribution

- The likelihood for θ based on *the observation of n samples* is defined as the **joint probability distribution** of observing a sequence of n binary values (y_1, y_2, \dots, y_n) such that every y_i is either 1 or 0. Since (y_1, y_2, \dots, y_n) are i.i.d. observations of the random variable X, the joint distribution is:

$$L_D(\boldsymbol{\theta}) = P(D; \boldsymbol{\theta}) = \prod_{i=1}^n P(y_i; \boldsymbol{\theta}) = \theta^r (1 - \theta)^{n-r}$$

such that: $\theta + (1 - \theta) = 1$ and $\theta \geq 0$

- Note that the constant $\frac{n!}{r!(n-r)!}$ of the binomial distribution is not necessary in the likelihood formula. Since the $L_D(\boldsymbol{\theta})$ is a function on $\boldsymbol{\theta}$, and $\frac{n!}{r!(n-r)!}$ is a fixed constant, it does not affect the MLE (maximization).
- The constant values in the expression of $P(x_i; \boldsymbol{\theta})$ are statistically irrelevant.

MLE for Binomial Distribution

- Note that the values of r and $n-r$ in previous formula are, respectively, N_1 (the number of observed samples where $Y=1$) and N_0 (the number of samples with $Y=0$) in D ($|D|=n=N_1+N_0$). They represent a **sufficient statistics** to estimate the parameter θ of the binomial distribution
- We can employ MLE to **estimate** θ using the log-likelihood:

$$l_D(\theta) = \log(L_D(\theta)) = N_1 \log \theta + N_0 \log(1 - \theta)$$

- With $\theta+(1-\theta)=1$
- Taking derivative $\frac{\partial(l_D(\theta))}{\partial(\theta)}$ and equating it to 0 we obtain (log is the natural log):

Note: $\hat{\theta}$ is our ESTIMATE of the parameter, given the evidence

$$\frac{N_1}{\theta} = \frac{N_0}{1-\theta} \rightarrow \hat{\theta} = \frac{N_1}{N_0+N_1} = \frac{N_1}{|D|} = \frac{N_1}{n}$$

Remember: to maximize or minimize a function you need to take the derivative

MLE for Multinomial Distribution

- Suppose that $D = (x_1, x_2, \dots, x_n)$ represents n i.i.d. samples ($n=|D|$) of a random variable X . The random variable is now **multinomial**, i.e., it can assume one out of k different values each with probability θ_j (e.g., assigning a category label to a document in a predictive task)

- Consider the PMF of a Multinomial distribution:

$$P(D; \theta) = \frac{n!}{r_1! r_2! \dots r_k!} \theta_1^{r_1} \theta_2^{r_2} \dots \theta_k^{r_k} = \frac{n!}{r_1! r_2! \dots r_k!} \prod_{j=1}^k \theta_j^{r_j}$$

Where r_j is the number of occurrences of the outcome j . **Each r_j can be estimated by the dataset D .**

- Notice that in the multinomial distribution, the Maximum likelihood parameter vector θ is composed by a set of k probabilities $\theta_1 \theta_2 \dots \theta_k$ of observing each possible outcome $X=x_j$.

MLE for Multinomial Distribution

- Since (x_1, x_2, \dots, x_n) are i.i.d., the joint distribution is:

$$(1) L_D(\boldsymbol{\theta}) = P(D; \boldsymbol{\theta}) = \prod_{i=1}^n P(x_i; \boldsymbol{\theta}) = \prod_{j=1}^k \theta_j^{r_j} \text{ *such that: } \sum_i \theta_i = 1 \text{ and } \theta_i \geq 0 \forall i*$$

- Since the $L_D(\boldsymbol{\theta})$ is a function on $\boldsymbol{\theta}$, $\frac{n!}{r_1!r_2!\dots r_k!}$ is a fixed constant, it does not affect the

MLE (maximization)! **As before, the constants are statistically irrelevant.**

- In this case the «sufficient statistics» are $r_1, r_2 \dots r_k$, e.g. the observations of the of $|D|$ values x_i of the multinomial random variable X (*we observe* r_1 times value x_1 , ecc.)
- To calculate the log-likelihood and incorporating constraints (the «such that» above) we can use the **Lagrangian method**

Lagrangian optimization

Given a function $f(x)$ and a set of constraints c_1, \dots, c_n , a *Lagrangian* is a function $L(f, c_1, \dots, c_n, \alpha_1, \dots, \alpha_n)$ that “incorporates” the constraints in the optimization problem

$$L(x, \alpha) = f(x) - \sum \alpha_i c_i(x)$$

Karush-Kuhn-Tucker (KKT) conditions: The optimum is at a point where

$$1) \nabla f(x) - \sum \alpha_i \nabla c_i(x) = 0$$

The derivative is 0

$$2) \alpha_i \geq 0$$

$$3) \alpha_i c_i(x) = 0 \quad \forall i$$

This condition is known as **complementarity condition** (it means that at least one of $\alpha_i, c_i(x)$ must be zero).

More at this [link](#)

MLE optimization with Lagrangian

- MLE optimization then implies first, incorporating the constraints with lagrangians
- From previous equation (1) by applying the log and incorporating constraints (α is the lagrangian coefficient), we obtain:

$$l_D(\theta) = \sum_j r_j \log \theta_j + \alpha \left(1 - \sum_j \theta_j \right)$$

- Next, we apply the partial derivatives vrs. each θ_j and set them to zero, and solve equations considering that $\sum_j \theta_j = 1$

$$\theta_j = \frac{r_j}{\sum_i r_i} = \frac{r_j}{|D|} = \frac{r_j}{n}$$

MLE for supervised learning

- Previous formulas estimate the probability of observing a given value x_j in an «unconditional» setting, that is: we estimate the likelihood of observing a given «outcome» in the data, **regardless of the function $y=f(x)$ that generates these outcomes.**
- The MLE principle is:

$$\theta_{MLE} = \operatorname{argmax}_{\theta} (P(D; \theta))$$

- Where θ are the parameters of M , a model that may represent the distribution of data, e.g. a binomial.
- We now need to apply MLE to a **supervised** context where we consider both input and output data (x and y)

$$\theta_{MLE} = \operatorname{argmax}_{\theta} (P(D; (\mathbf{x}, \theta)))$$

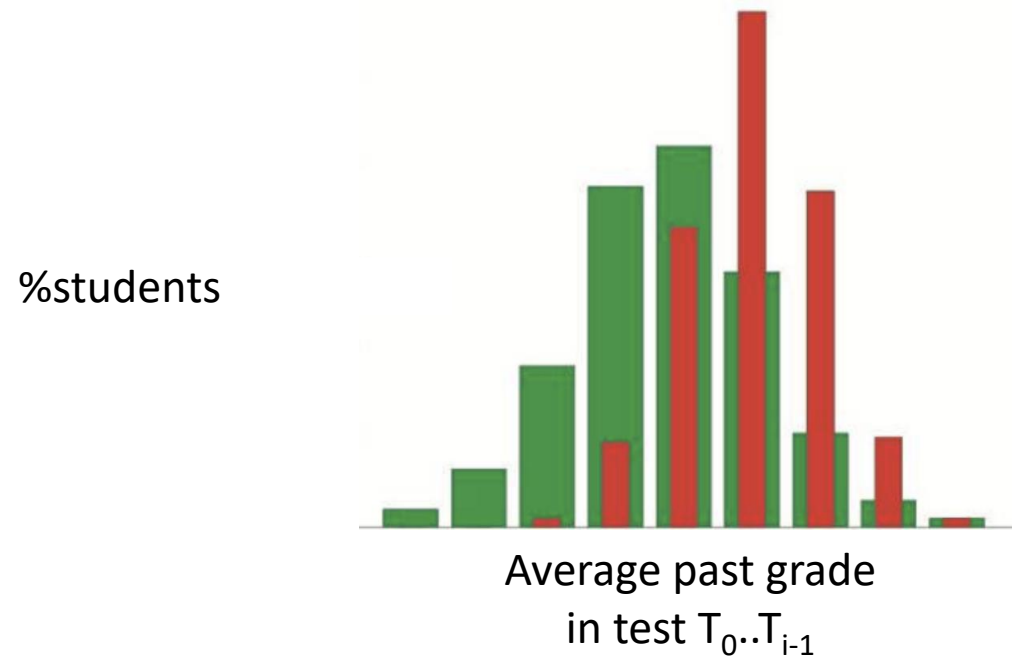
Example of prediction with MLE

- Suppose we have univariate multinomial instances x describing a unique feature: the age of citizens (values are, e.g.: A: [16-20) B:[20-23) C:[23-28) D [28, 100]), the class label is binary and represents their average cellular phone bill, where $Y=0 \rightarrow$ less than €100, $Y=1 \rightarrow \geq \text{€}100$
- Step 1: we partition our data (patients) in two subsets $D_{Y=0}$ and $D_{Y=1}$;
- Step 2: we assume that x follows a multinomial distribution, and we estimate, for each subset $D_{Y=i}$ **separately**, the model parameters $\Theta_{D_{Y=i}}$ using the log-likelihood maximization seen above
- Step 3: given a new unseen instance x' , we predict the class Y as

$$Y = \operatorname{argmax}_j (P(x'; Y = j))$$

- in practice this will be the most likely bill range for citizens aged in the range A, B, C or D, to which x belongs. **We select the class value j that maximizes the probability that x has been generated by the model $M(D_{Y=j})$ with parameters $\Theta_{D_{Y=i}}$**

MLE for continuous variables



Suppose we have data points representing the average of past grades of students in a class (so we have only one feature for each instance, a univariate problem, but now the variable is continuous), and we want to predict if they will pass a new test. (in our train set D , red passed test T_i , green did not). So here x is continuous, y is binary. In the histogram, x is the average, y is the % of the population with that average value, the color distinguishes the distribution (in our sample) of those who passed and those who not passed.

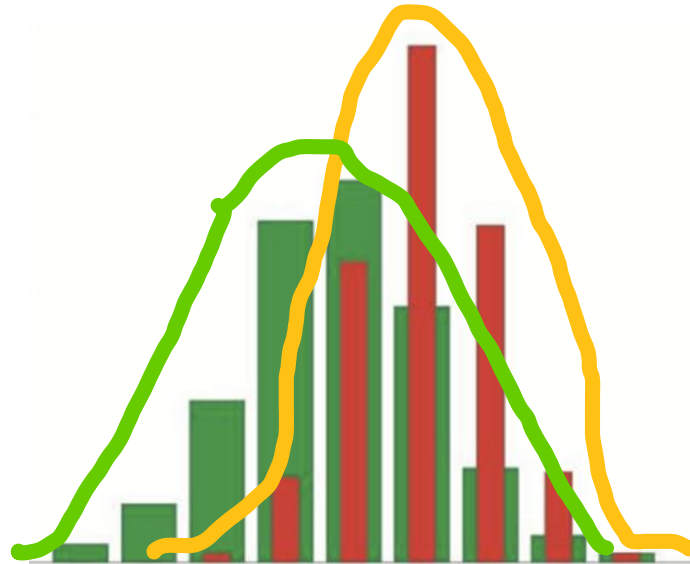
MLE for Gaussian distribution

- Given our data, let's suppose that the model for input variable X (average grade) follows approximately a Gaussian distribution
- So the model M is (probability density of X in C):

$$p(x|C) = \frac{1}{\sqrt{2\pi}\sigma_C} \exp\left(-\frac{(x - \mu_C)^2}{2\sigma_C^2}\right)$$

- Note, as for the discrete case before, that θ parameters ($\sigma_{C=1}\mu_{C=1}; \sigma_{C=0}\mu_{C=0}$) **are DIFFERENT for each class C** (those who passed and those who didn't).
- In the example we have just two classes (the output C is binomial) and one feature (univariate, continuous) but this applies in general to multiple classes and multiple features – each would follow a Gaussian)

Example (3): fitting the model with data



We need to find, **for each subset of samples** (in our example, students who passed test T_i and those who didn't), the values of θ parameters σ, μ of the Gaussian, such that $P(D_C; \theta)$ best «fits» the observed data.

Example: MLE for Gaussian distributions

- Let D_{C_i} be the subset of D of instances with class $C=C_i$ (0 or 1) and let's use a superscript to denote different instances $x^{(i)}$ in D_{C_i} (so that for multi-variate observations we can denote with subscripts their features).
- We then have for the log-likelihood (we omit the C_i subscript to avoid overloading the notation and furthermore $|D_{C_i}|=N$):

$$l_D(\theta) = -\ln \left(\prod_{n=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp \left(-\frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right) \right)$$
$$= \sum_{n=1}^N \ln(\sqrt{2\pi}\sigma) + \sum_{n=1}^N \frac{(x^{(n)} - \mu)^2}{2\sigma^2} = \frac{N}{2} \ln(2\pi\sigma^2) + \sum_{n=1}^N \frac{(x^{(n)} - \mu)^2}{2\sigma^2}$$

$x^{(n)}$ is the n -th
observation in D

We have two parameters, μ and σ

Note the «-» sign is like multiplying by a constant, does not change the ranking and simplifies the expression

Computing partial derivatives

- We then apply the derivative $\frac{\partial(l)}{\partial\mu}$ to the log likelihood and set to zero

$$\begin{aligned} &= \frac{\partial \left(\frac{N}{2} \ln(2\pi\sigma^2) + \sum_{n=1}^N \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right)}{\partial\mu} = \frac{d \left(\sum_{n=1}^N \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right)}{d\mu} \\ &= \frac{-\sum_{n=1}^N 2(x^{(n)} - \mu)}{2\sigma^2} = -\sum_{n=1}^N \frac{(x^{(n)} - \mu)}{\sigma^2} = \frac{N\mu - \sum_{n=1}^N x^{(n)}}{\sigma^2} \end{aligned}$$

- Setting to zero the derivative (\Rightarrow numerator must be =0), we obtain:

$$\mu = \frac{1}{N} \sum_{n=1}^N x^{(n)}$$

Note that for multivariate instances \mathbf{x} , also $\boldsymbol{\mu}$ is a vector

Computing partial derivatives

- Next, we compute derivative $\frac{\partial l}{\partial \sigma}$ and set it to 0

$$\begin{aligned} & , \\ & = \frac{\partial \left(\frac{N}{2} \ln(2\pi\sigma^2) + \sum_{n=1}^N \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right)}{\partial \sigma^2} \\ & = \frac{N}{2} \frac{1}{2\pi\sigma^2} 2\pi + \frac{\sum_{n=1}^N (x^{(n)} - \mu)^2}{2} \left(\frac{-1}{\sigma^4} \right) \\ & = \frac{N}{2\sigma^2} - \frac{\sum_{n=1}^N (x^{(n)} - \mu)^2}{2\sigma^4} \end{aligned}$$

$$0 = \frac{N}{2\sigma^2} - \frac{\sum_{n=1}^N (x^{(n)} - \mu)^2}{2\sigma^4} = \frac{N\sigma^2 - \sum_{n=1}^N (x^{(n)} - \mu)^2}{2\sigma^4}$$

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x^{(n)} - \mu)^2$$

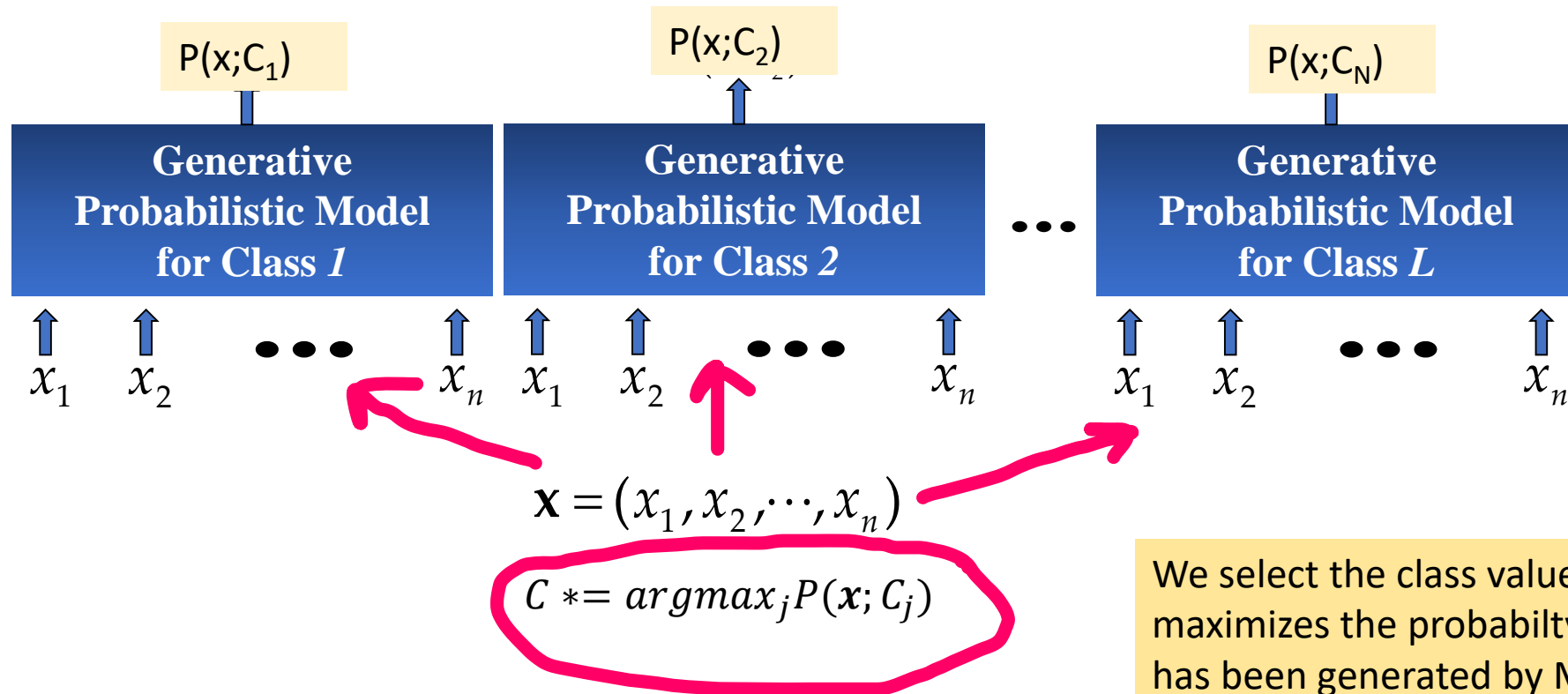
(which is a rather obvious result..)

This generalizes in a straightforward way to the multivariate case (multiple features): now \mathbf{x}^n , $\boldsymbol{\sigma}$, $\boldsymbol{\mu}$ are vectors, and we compute partial derivatives

$$\frac{\partial l}{\partial \mu_i}, \frac{\partial l}{\partial \sigma_i}$$

When the model parameters have been estimated, how is a new prediction made with MLE?

Given a new instance \mathbf{x} (we now suppose \mathbf{x} is a vector), and having estimated the parameters of the generative models for each possible class, we compute $P(\mathbf{x} | C_i)$ for each model, and compute the argmax



Maximum A Posteriori (MAP) another generative model

- MAP is an alternative probabilistic formulation of ML problems, still belonging to the **generative** class of models
- In short, the essential difference wrt MLE is in **what** is modeled using probability theory. MLE first, learns a joint probability $L_D(\theta) = f(D; \theta)$ and then, to predict a class, estimates the argmax of $P(X=x; Y=y_i)$ for every output value or class
- MAP uses a **Bayesian** approach: The formulation is a **conditional** probability A/B rather than a joint probability A;B.
- MAP estimates conditional probabilities rather than the parameters of a distribution

Maximum A Posteriori estimate

- In MAP, the probabilistic model M is: $P(Y = y | X = \mathbf{x})$ ($P(y|\mathbf{x})$ for short) , the conditional probability of class label y , given the observation of an instance \mathbf{x}
- The prediction is computed as $y^* = \operatorname{argmax}_{y_j} (P(Y = y_j | X = \mathbf{x}))$ (remember, in MLE we computed the joint probability $P(\mathbf{x}; Y(\mathbf{x}))!!$)
- As for MLE, we need to estimate the parameters θ of M

Learn the values of model parameters that **maximize** the the conditional probability of a class label y given the observation of x . Model parameters are probabilities.

- *The solution of the problem is based on the **Bayes theorem***

Maximum a posteriori estimate learning

- Similarly to MLE, we estimate y_i by maximizing a probability function distribution, which is for MAP:

$$y^* = \operatorname{argmax}_{y_i} (P(Y = y_i | X = \mathbf{x}))$$

- In probability calculus, often estimating $P(a|b)$ is easier than estimating $P(b|a)$ so MAP applies the **Bayes theorem** to **invert** conditional probabilities

$$y^* = \operatorname{argmax}_{y_i} (P(Y = y_i | X = \mathbf{x})) = \operatorname{argmax}_{y_i} \left(\frac{P(Y = y_i) P(X = \mathbf{x} | Y = y_i)}{P(X = \mathbf{x})} \right)$$

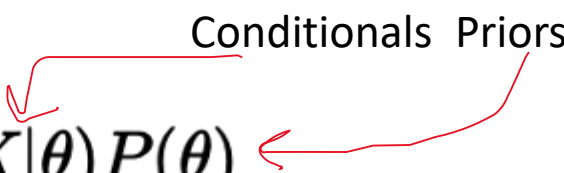
$$\approx \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = \mathbf{x} | Y = y_i))$$

- Note that since the denominator $P(X = \mathbf{x})$ is **common to all the probabilities, it does not affect the ranking in the argmax computation. No need to compute** it!!!
- Also note that the problem is now formulated both in terms of prior probabilities (the $P(Y = y_i)$) of the random variable to be predicted, **and** conditional probabilities $P(X | Y)$

Given previous formula, how does MAP works?

$$\theta_{MAP} = \arg \max_{\theta} P(X|\theta)P(\theta)$$

Conditionals Priors



$$= \arg \max_{\theta} \log P(X|\theta) + \log P(\theta)$$

Apply log

$$= \arg \max_{\theta} \log \prod_i P(x_i|\theta) + \log P(\theta)$$

Use property of independence of observations

$$= \arg \max_{\theta} \sum_i \log P(x_i|\theta) + \log P(\theta)$$

Change product with sum (property of logs)

Probabilistic Classification with MAP

(multinomial, multivariate X and Y)

- Let Y be a univariate *multinomial* random variable for the **output class** Y which takes values $\{y_1, y_2, \dots, y_n\}$ (n possible classifications for our instances).
- Let X be a *multivariate multinomial random variable* describing input instances consisting of d features
- $\mathbf{x}: \langle X_1, X_2, \dots, X_d \rangle$, let x_j be a possible **value** for the feature X_j ($j = 1, \dots, d$)
- Remember: we now use **uppercase** X_i for features, **to suggest that features are random variables, lowercase** for instances **xi (random vectors)** of X in D .
- Since features X_i are multinomial (multiple values), for our classification task, we need to compute the conditional probabilities:

$$P(Y = y_i | X = \mathbf{x}: \langle X_1 = x_1, \dots, X_d = x_d \rangle) \text{ for } i = 1, \dots, n$$

(e.g. $P(Y = \text{positive} | X = \mathbf{x}: \langle \text{color} = \text{blue}, \text{shape} = \text{circle} \rangle)$)

Remember:

- Multinomial is a random variable that can take a finite number of values
- Multivariate is a random vector made of multiple random variables

Probabilistic Classification

$$P(Y = y_i | X = \mathbf{x} < X_1 = x_1, \dots, X_d = x_d >) \text{ for } i = 1, \dots, n$$

(e.g. $P(Y = \text{positive} | X = \mathbf{x} < \text{color} = \text{blue}, \text{shape} = \text{circle} >)$)

- the objective is to classify a new unseen instance \mathbf{x} by first estimating the probability of each possible classification y_i , **given** the observation of feature values of the instance to be classified
- To estimate $P(Y = y_i | X = \mathbf{x})$ we use a **learning set D** of pairs $(\mathbf{x}, Y(\mathbf{x}))$
- **Summary notation:**
 - i is index of *class values* (Y is univariate and multinomial)
 - j is index of *features* (X is multivariate, X_j are multinomial)
 - uppercase X_j is a feature, lowercase x_j is a value
 - Bold lowercase \mathbf{x} is a specific instance in D ($\mathbf{x} \in X$) – here the idea is that uppercase represents a random variable, lowercase an observation.
 - We omit the superscript h of instances \mathbf{x}^h in D (unless strictly necessary) to avoid overloading the notation

How can we compute

$$P(Y = y_i | X = \mathbf{x})??$$

- Example: we observe the instance \mathbf{x} <color = **red**, shape = **circle**> (two symbolic features) and there are two possible classes \mathbf{Y} < y_1 = positive, y_2 = negative>
- Possible values for features are: color: {*red*, *blue*} ; shape: {*circle*, *square*}
- We need to compute:

$$P(\text{positive} | \text{red} \dot{\cup} \text{circle}) = \frac{P(\text{positive} \dot{\cup} \text{red} \dot{\cup} \text{circle})}{P(\text{red} \dot{\cup} \text{circle})}$$

$$P(\text{negative} | \text{red} \dot{\cup} \text{circle}) = \frac{P(\text{negative} \dot{\cup} \text{red} \dot{\cup} \text{circle})}{P(\text{red} \dot{\cup} \text{circle})}$$

- So we need to estimate the **joint probabilities** (e. g. $P(\text{positive} \wedge \text{red} \wedge \text{circle})$)
- The **joint probability distribution** for a set of **random independent variables** gives the probability of **every combination of values**

Joint probability tables

$P(\text{shape} = \text{circle}, \text{color} = \text{blue}, C = \text{positive})$

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 training set, by summing the appropriate subset of values from the joint distribution.

$$P(\text{red} \cup \text{circle}) = P(\text{red} \cup \text{circle} \cup \text{positive}) + P(\text{red} \cup \text{circle} \cup \text{negative}) = 0.20 + 0.05 = 0.25$$

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

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

Example

Consider this learning set D of 5 annotated instances:

- x^1 (red, circle), positive
- x^2 (red,square),negative
- x^3 (blue,circle),positive
- x^4 (red, circle),negative
- x^5 (red, circle), positive

$$P(\text{positive} \mid \text{red} \wedge \text{circle}) = \frac{P(\text{positive} \wedge \text{red} \wedge \text{circle})}{P(\text{red} \wedge \text{circle})} = \frac{2/5}{3/5} = \frac{2}{3}$$

Probabilistic Classification

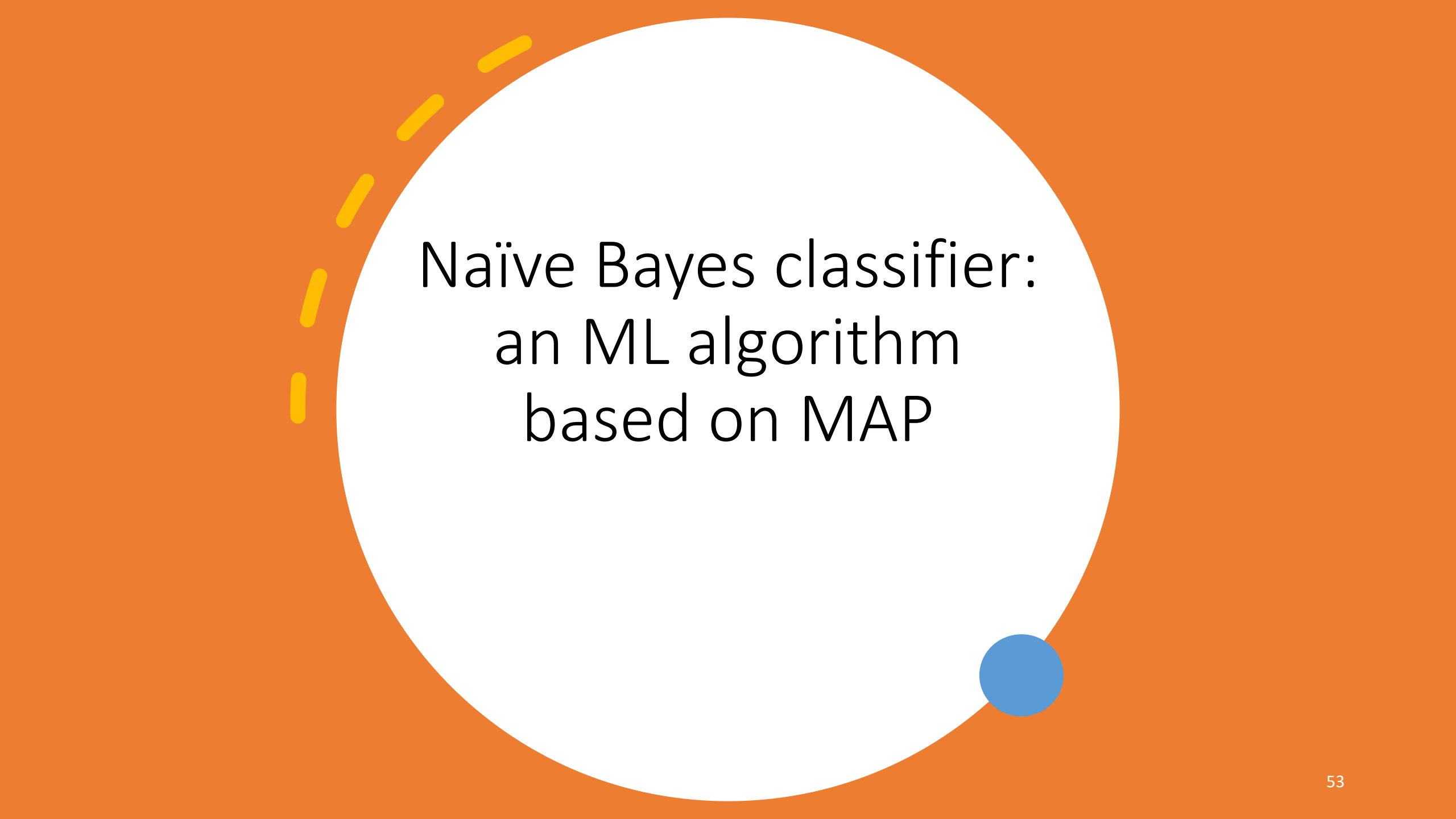
- However, given no other assumptions, this requires tables **assigning a probability to each category label for each possible combination of feature values in the instance space \mathbf{X}** , which is **impossible to accurately estimate** from a reasonably-sized training set \mathbf{D} .
- E.g $P(Y = y_i | X_1 = x_1, X_2 = x_2, \dots, X_d = x_d)$ for all y_i and x_j
Assuming that Y and all \mathbf{X}_j are binomial, and we have d features, we need 2^d entries to estimate $P(Y = pos | X = \mathbf{x}^k)$ for each of the 2^d possible *combinations of feature values* since:
 - $P(Y = neg | X = \mathbf{x}) = 1 - P(Y = pos | X = \mathbf{x})$



Summary MAP so far

- Given:
 - An unclassified random vector \mathbf{x} , which is represented by d discrete features (e.g. binomial or multinomial)
 - A **multinomial classification function** $Y(\mathbf{x})$ with possible values $\{y_1, \dots, y_m\}$
- The **target** of the probabilistic classifier is to classify \mathbf{x} based on:
$$y^* = \operatorname{argmax}_{y_i} (P(y_i | \mathbf{x}))$$

(the most likely classification given the specific combination of feature values in \mathbf{x})
- The **parameters** of the model are the $P(y_i | \mathbf{x})$
- We need to estimate $P(y_i | \mathbf{x})$ for all y_i , **using the evidence provided by the previously seen instances** $\langle \mathbf{x}_i, y_i \rangle$ in D
- Quite likely, the specific combination of feature values of \mathbf{x} is **not in the learning set**: even if values are boolean, there are **2^d possible combinations!** So how do we go about it?



Naïve Bayes classifier:
an ML algorithm
based on MAP

Naïve Bayes Model

Let's define the model:

- We are given an evidence represented by an annotated learning set D of classified instances (\mathbf{x}, y) .
- Consider an instance \mathbf{x} as a random vector of d random variables

$$\langle X_1, X_2, \dots, X_d \rangle$$

- Each X_j feature can assume a discrete value $x_k \in \{1, \dots, d_j\}$ (i.e. binary or multinomial)
- The **class** variable is a multinomial random variable Y that can take a set of values y_i with $i = 1, \dots, n$, usually categorical (e.g., red, blue,..)

Naïve Bayes assumption: all the X_i are **statistically independent**

Naïve Bayes Model

- Our goal is to make a prediction, based on a conditional probability model $P(Y|X)$, whose parameters can be estimated considering the dataset D (Training)

- For MAP and Bayes Theorem, this requires to compute (see previous slides):

$$y^* = \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = \mathbf{x} | Y = y_i))$$

Where \mathbf{x} is a vector of feature values $\langle X_1 = x_1, X_2 = x_2, \dots, X_d = x_d \rangle$

Naïve Bayes Model

How can we calculate $P(X = \mathbf{x} < x_1, \dots, x_d > | Y = y_i)$?

$$P(\mathbf{x} = \langle X_1 = x_1, X_2 = x_2, \dots, X_d = x_d \rangle | Y = y_i) =$$

By Independence assumption (e.g. Naïve Bayes Assumption):

$$= \prod_{j=1}^d P(X_j = x_j | Y = y_i)$$

We assume that features' random variables are also statistically independent!

To classify a new instance we apply the MAP formulation + NB assumption:

$$y^* \approx \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = \mathbf{x} | Y = y_i)) = \\ \operatorname{argmax}_{y_i} (P(Y = y_i) \prod_{j=1}^d P(X_j = x_j | Y = y_i))$$

Naïve Bayes Model

Now, we have to estimate the priors $P(Y = y_i)$ and the conditionals $P(X_j = x_j | Y = y_i)$ (our parameters)

- $P(Y = y_i)$ can be estimated directly from the learning set D, since categories are **complete** and **disjoint**
- Next, **we use again likelihood** to estimate two sets of parameters θ^1 and θ^2 :
 - $\theta^1 : P(Y = y_i)$ (the priors)
 - $\theta^2 : P(X_j = x_j | Y = y_i) = \frac{P(X_j = x_j, Y = y_i)}{P(Y = y_i)}$ (the conditionals)

Naïve Bayes Model

- The **estimate** of parameters depend on the random variable distribution:
 - Categorical Distributions (e.g. Binomial, Multinomial, Geometric)
 - Gaussian, exponential..
 - Etc.
- Let first suppose that both Y and the X_j follow a **multinomial** distribution, then (for what we have seen in previous slides), using the likelihood maximization:

$$\theta_i^1 = \frac{N(y_i)}{|D|} \qquad \theta_{ij}^2 = \frac{N(x_j, y_i)}{N(y_i)}$$

- Where $N(y_i)$ is the number of instances classified y_i in the training set (so θ_i^1 are the priors);
- Where $N(x_j, y_i)$ is the number of times the label y_i is seen in conjunction with feature value x_j in D : number of instances in D in which $X_j=x_j$ and $Y=y_i$

Note that these formulas are obtained exactly as for the MLE case (see previous slides): we consider the log of the multinomial for each random variable, then take the derivative and equate to 0 (ignoring the constant values).

Naïve Bayes Model

Make Predictions

Finally to make a **prediction** for a new instance $\mathbf{x} = \langle X_1 = x_1, X_2 = x_2, \dots, X_d = x_d \rangle$ we must calculate:

$$y^* \approx \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = \mathbf{x} | Y = y_i)) = \\ \operatorname{argmax}_{y_i} \left(P(Y = y_i) \prod_{j=1}^d P(X_j = x_j | Y = y_i) \right) =$$

$$\operatorname{argmax}_{y_i} \left(\frac{N(y_i)}{|D|} \prod_{j=1}^d \frac{N(x_j, y_i)}{N(y_i)} \right)$$

- Where $N(y_i)$ is the number of instances classified with label y_i in the training set;
- Where $N(x_j, y_i)$ is the number of times the label y_i is seen in conjunction with the feature $X_j = x_j$

Naïve Bayes Model Example

x	X1	X2	X3	Y
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

$$|D|=16, d=3$$

X1= size X2=color X3=shape

X1={*small, medium, large*}

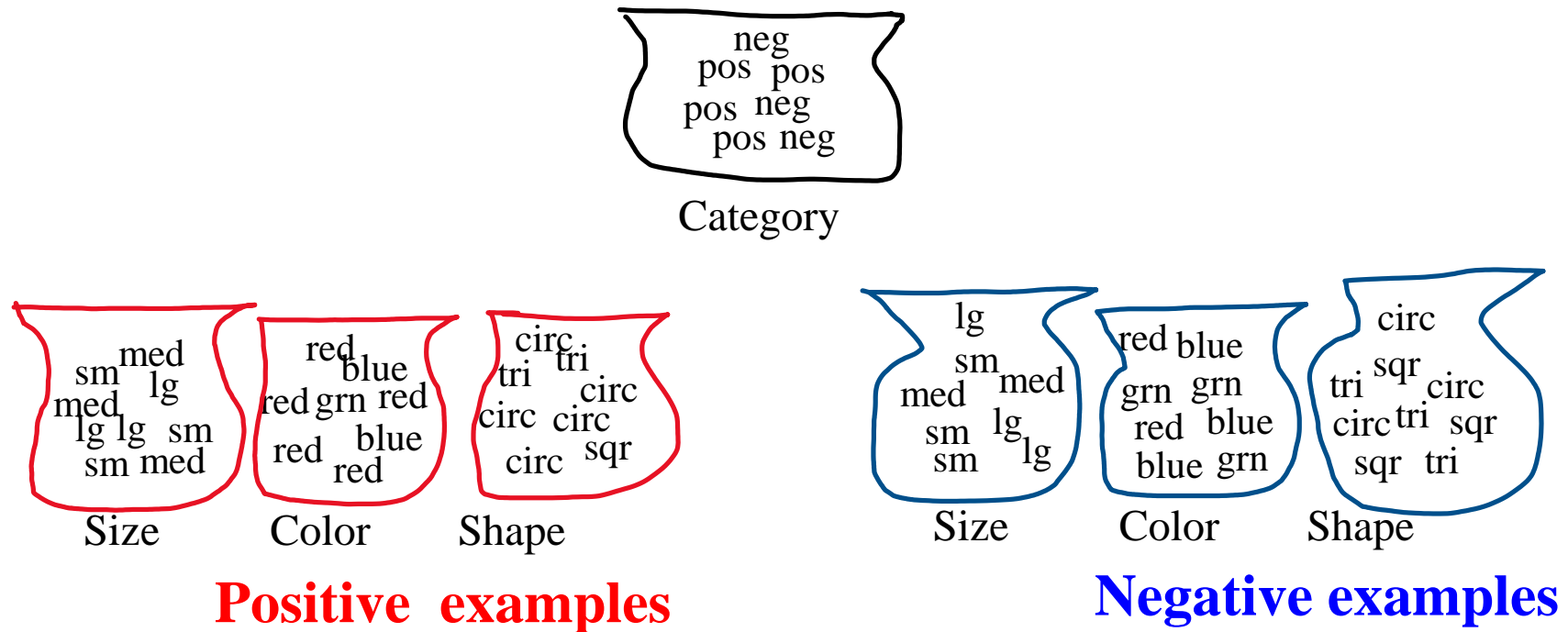
X2={*red, green, blue*}

X3={*triangle, square, circle*}

Y = {*pos, neg*}

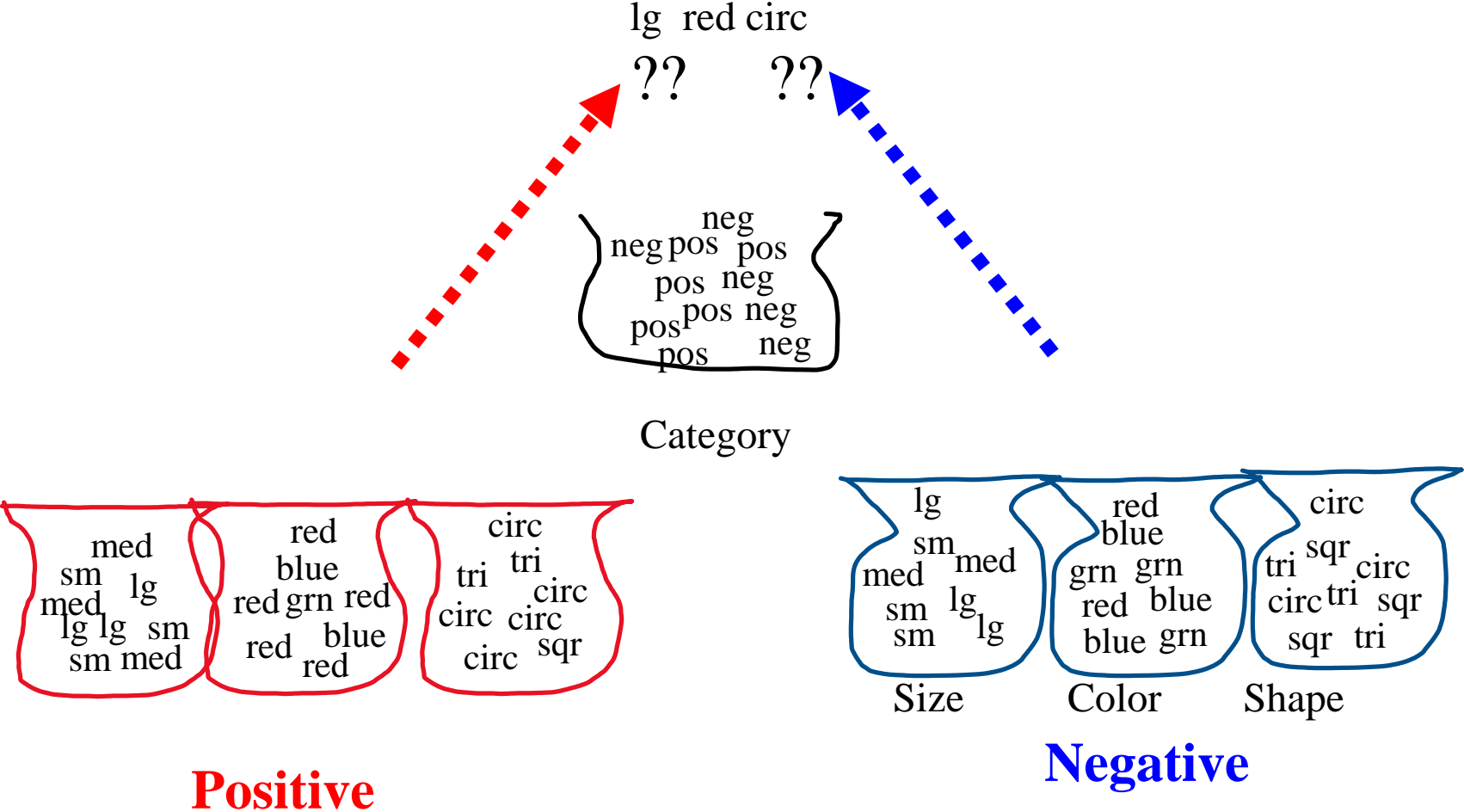
Naïve Bayes Model Example

- Urns represent occurrences of feature **values** in training set D. Red urns are positive, blue negative. Separate urns indicate **statistical independence** of features (sampling of values from each urn does not affect the others).



Naïve Bayes Inference Problem

Let's say we have a new unclassified instance $x: \langle large, red, circle \rangle$. We need to estimate, using the learning set D , the class label y_i that maximizes the probability of observing $x: \langle large, red, circle \rangle$. Is it more likely to extract this combination from red or from blue urns?

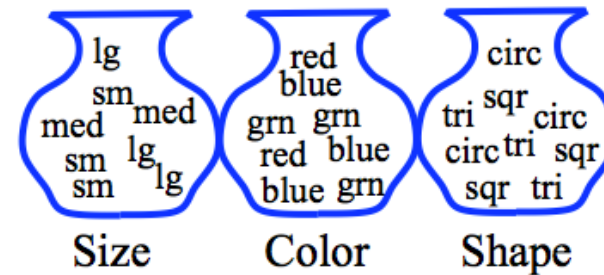
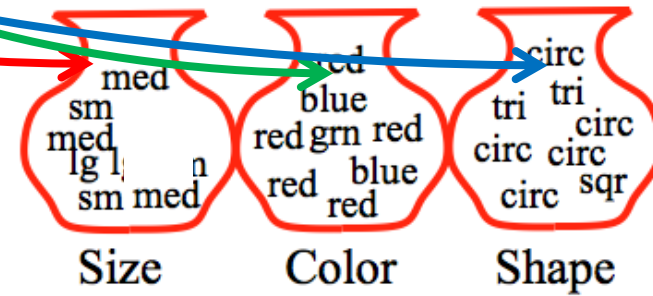


How?

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

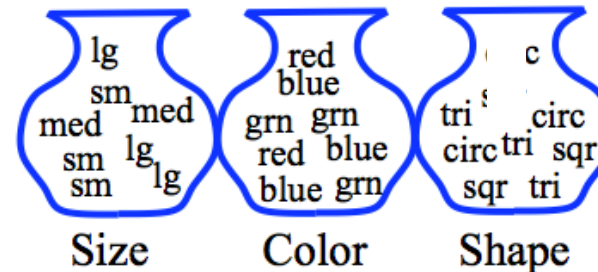
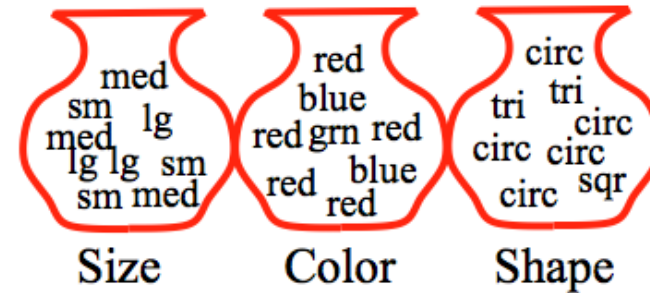
Note that because of independence assumption, the specific combination of feature values does not matter!



Now we can compute parameters θ_i^1 and θ_{ij}^2

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

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



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

For example, \mathbf{x} :<medium ,red, circle> \mathbf{y} =?

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

$$y^* \approx \operatorname{argmax}_{y_i} \left(\theta_i^1 \prod_{j=1}^d \theta_{ij}^2 \right)$$

$$P(\text{positive} | X) = \frac{P(\text{positive}) \cdot P(\text{medium} | \text{positive}) \cdot P(\text{red} | \text{positive}) \cdot P(\text{circle} | \text{positive})}{P(X)}$$

$$= \frac{0.5 \cdot \frac{3}{8} \cdot \frac{5}{8} \cdot \frac{5}{8}}{0.5 \cdot \frac{3}{8} + 0.5 \cdot \frac{2}{8} + 0.5 \cdot \frac{2}{8} + 0.5 \cdot \frac{2}{8}}$$

$$= 0,073$$

$$y^* = \operatorname{argmax}_{y_i} (P(Y = y_i | X = x)) = \operatorname{argmax}_{y_i} \left(\frac{P(Y = y_i) P(X = x | Y = y_i)}{P(X = x)} \right)$$

$$\approx \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = x | Y = y_i))$$

Note: sum is not 1 since we ignore the denominator of original formulation $P(X = x)$

Naive Bayes Summary

Classify any new unseen instance $\mathbf{x} = (X_1, \dots, X_d)$ as:

$$y^* \approx \operatorname{argmax}_{y_i} (P(Y = y_i) P(X = \mathbf{x} | Y = y_i)) = \operatorname{argmax}_{y_i} \left(P(Y = y_i) \prod_{j=1}^d P(X_j = x_j | Y = y_i) \right)$$

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

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

- For each attribute value of x_j of each datum instance
$$P(X_j = x_j | Y = y_i) := \mathbf{estimate} P(x_j | y_i)$$

Estimating Probabilities

- Usually, as in previous example, probabilities are estimated based on observed frequencies in the training data.
- If D contains $N(y_i)$ examples in category y_i , and $N(x_j, y_i)$ of these $N(y_i)$ examples have value x_j for feature X_j , then:

$$P(X_j = x_j | Y = y_i) = \frac{N(x_j, y_i)}{N(y_i)}$$

- However, estimating such probabilities from small training sets is **error-prone** (**bias** in the estimate, as we have seen)
- If - due only to low chance - a rare feature value, $X_j = x_j$ is never observed in the training data, then our estimate

$$P(X_j = x_j | Y = y_i) = 0.$$

- If $X_j = x_j$ actually **occurs** in a test instance, \mathbf{x} , the result is that $\forall y_i: P(Y = y_i | \mathbf{x}) = 0$ (since individual probability estimates **are multiplied**)

Probability Estimation Example

Test Instance x :
 <medium, red, circle>

No instances in D with Size=medium!

$$P(\text{positive} \mid x) = 0.5 * 0.0 * 1.0 * 1.0 / P(x) = 0$$

$$P(\text{negative} \mid x) = 0.5 * 0.0 * 0.5 * 0.5 / P(x) = 0$$

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

Probability	positive	negative
$P(Y)$	0.5	0.5
$P(\text{small} \mid Y)$	0.5	0.5
$P(\text{medium} \mid Y)$	0.0	0.0
$P(\text{large} \mid Y)$	0.5	0.5
$P(\text{red} \mid Y)$	1.0	0.5
$P(\text{blue} \mid Y)$	0.0	0.5
$P(\text{green} \mid Y)$	0.0	0.0
$P(\text{square} \mid Y)$	0.0	0.0
$P(\text{triangle} \mid Y)$	0.0	0.5
$P(\text{circle} \mid Y)$	1.0	0.5

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 .

$$P(X_j = x_j | Y = y_i) = \frac{N(x_j, y_i) + mp}{N(y_i) + m}$$

- For binary features, p is simply assumed to be 0.5, while it can be set to $\frac{1}{k}$ where k is the number of values that x_j can assume.

Laplace Smoothing Example

- Assume training set contains 10 **positive** examples, and the feature “Size” has 3 values, but 1 value (size=medium) is not observed in D :
 - 4: small
 - 0: medium
 - 6: large
- Estimate parameters as follows (if we set $m=1$, $p=1/3$)
 - $P(\text{small} \mid \text{positive}) = (4 + 1/3) / (10 + 1) = 0.394$
 - $P(\text{medium} \mid \text{positive}) = (0 + 1/3) / (10 + 1) = 0.03$
 - $P(\text{large} \mid \text{positive}) = (6 + 1/3) / (10 + 1) = 0.576$
 - $P(\text{small or medium or large} \mid \text{positive}) = 1.0$

Naïve Bayes (MAP) with Continuous Distributions

- If X is a **continuous** (univariate) random variable rather than a discrete one, need another way to calculate $P(X = x_j | Y = y_i)$.
- Assume that X has a **Gaussian** distribution whose mean and variance depend on Y .
- During training, **for each combination of the continuous values of X and a class value y_i for Y ,** estimate a **mean μ_i** , and standard deviation **σ_i** based on the observed values of feature X in class y_i in the training data. E.g., **μ_i is the mean value of X observed in instances for which $Y = y_i$ in D**
- **During testing**, estimate $P(X = x_j | Y = y_i)$ for a given observation $X = x_j$, using the Gaussian distribution defined by μ_i and σ_i .

$$P(X = x_j \in R | Y = y_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(\frac{-(X_j - \mu_i)^2}{2\sigma_i^2}\right)$$

Probability of observing the value x_j (our current observation) given that the class is y_i

Similarly if we assume that X follows other types of distributions, e.g., softmax, logit.. This method easily extends to multivariate X : $X_1..X_n$, and we need to estimate σ_{ji} and μ_{ji}

Comments on Naïve Bayes

- May work well despite **strong assumption** of conditional independence.
- 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.
- Does not perform any search of the hypothesis space. Directly constructs a hypothesis from parameter estimates that are easily calculated from the training data.
- Not guaranteed consistency with training data.
- Typically handles noise well since it does not even focus on completely fitting the training data.

More

- Likelihood Function and MLE: [LINK](#), [LINK](#), [LINK](#), [LINK](#), [LINK](#), [LINK](#), [LINK](#), [LINK](#)
- MLE VS MAP: [LINK](#),[LINK](#), [LINK](#),[LINK](#), [LINK](#),[LINK](#)
- Naïve Bayes [LINK](#), [LINK](#), [LINK](#), [LINK](#)
- Note that what we said also applies to DEEP models. For example, \mathbf{x} can be the “deep” representation of an instance (e.g., an image), and we want to infer the model M that underlies the generation of our images (inferring M can be easier if we capture the “invariants” of the input).
- See here an [introduction to deep generative models](#)