Monte Carlo Techniques

Gibbs Sampling

The Markov chain Monte Carlo algorithm applied to a Gaussian Mixture Model

Author:
Eric Bataller

Examiner:
prof. dr. R.H.P. Kleiss

April, 2020

I. INTRODUCTION

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. Monte Carlo methods are mainly used in two problem types: numerical integration, and generating draws from a probability distribution.

Monte Carlo methods can be theoretically used to solve any problem having probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean of independent samples of the variable. As a classical example of Monte Carlo sampling methods we find rejection sampling. In a nutshell, rejection sampling is based on the observation that to sample a random variable x that follows a probability density $P(x) = P(x)^*/Z$, one can perform a random sampling of a simpler proposal density Q(x) which we can evaluate (within a multiplicative factor Z_Q). And assuming that we know the value of a constant c such that $cQ(x)^* > P(x)^*$ for all x, we can turn those x samples that follow Q(x) into samples from the P(x) distribution by assigning a generated uniformly distributed random variable u from the interval [0,$cQ(x)^*$] to every x sample and accepting it or rejecting it as part of the set of samples $\{x^{(r)}\}$ that follow the P(x) distribution depending on whether $u < P(x)^*$ or $u > P(x)^*$, respectively.

Markov chain Monte Carlo (MCMC) are computer-driven sampling methods. The idea in MCMC sampling is to sample, not directly from P(x), but from a different distribution such that, in the limit of a large number of samples, effectively the samples will be from P(x).It allows one to characterize a distribution without knowing all of the distribution's mathematical properties by randomly sampling values out of the distribution.To achieve this we forward sample from a Markov transition whose stationary distribution is equal to P(x). Therefore we combine Monte Carlo techniques with a Markov chain scheme. The Markov chain property of MCMC is the idea that the random samples are generated by special sequential process. Each random sample is used as a stepping stone to generate the next random sample (hence, the *chain*).

In contrast to rejection sampling, where the accepted points $\{x^{(r)}\}$ are independent samples from the desired distribution, Markov chain Monte Carlo methods involve Markov process in which a sequence of states $\{x^{(t}\}$ is generated, each sample $x^{(t)}$ having a probability distribution that depends on the previous value $x^{(t-1)}$. Since successive samples are dependent, the Markov chain may have to be run for a considerable time in order to generate samples that are effectively independent samples from P.

II. MARKOV CHAINS

Consider the conditional distribution $Q(x^{t+1}; x^t)$ from which for a given initial sample x^1, we can recursively generate samples x^2,x^3,...,x^T. After a long time T >> 1, and provided the Markov chain is 'irreducible' (meaning that we can eventually get from any state to any other state in the state space) and 'aperiodic' (meaning we don't periodically revisit any state), those samples are from the unique stationary distribution $Q_\infty(x)$ which is defined as (for a continuous variable)

$$Q_\infty(x') = \int_x Q(x'; x) Q_\infty(x), \tag{1}$$

The condition for a discrete variable is analogous on replacing integration with summation. The idea in MCMC is, for a given distribution $P(x)$, to find a transition $Q(x'; x)$ which has $P(x)$ as its stationary distribution. If we can do so, then we can draw samples from the Markov chain by forward sampling and take these as samples from $P(x)$ as the chain converges towards its stationary distribution.

Note that for every distribution $P(x)$ there will be more than one transition $Q(x'; x)$ with $P(x)$ as its stationary distribution. This is why there are many different MCMC sampling methods, each with different characteristics and varying suitability for the particular distribution at hand. A special property of the chain is that, while each new sample depends on the one before it, new samples do not depend on any samples before the previous one (this is the "Markov" property).

III. METROPOLIS-HAISTING ALGORITHM

Rejection sampling works well only if the proposal density $Q(x)$ is similar to $P(x)$. In large and complex problems it is difficult to create a single density $Q(x)$ that has this property. Specially in high-dimensional problems. In contrast

to rejection sampling, here it is not necessary that $Q(x';x)$ look at all similar to $P(x)$ in order for the algorithm to be practically useful. The idea behind Metropolis-Hasting (MH) method is the following: Suppose we are at a state point x^t. Then, using some prescription, we generate a candidate new point y sampling from a proposal probability density $\tilde{Q}(y;x)$. Next, we compute the so called *acceptance ratio*:

$$a = \frac{P(y)\tilde{Q}(y;x)}{P(x)\tilde{Q}(x;y)} \tag{2}$$

If $a > 1$ then we accept the candidate point $y = x^{t+1}$. Otherwise, if $a < 1$, the candidate y is accepted as x^{t+1} with probability a. In case it is not accepted, then we stick to the old point, and have $x^{t+1} = x^t$.

More often than not, the user will choose a symmetrical proposal density $\tilde{Q}(y;x)$ such as a Gaussian centred on the current state x. The transitions from x to y and from y to x will be, therefore, equally probable:

$$\tilde{Q}(y;x) = \tilde{Q}(x;y) \tag{3}$$

This property is called *detailed balance*, and the Markov chains that satisfy it are also called *reversible* Markov chains. The reason why detailed-balance property is of interest is that it implies invariance of the distribution P(x) under the Markov chain with transition Q(x';x), which is a necessary condition for the key property that we want from our MCMC simulation -that the probability distribution of the chain should converge to P(x). Thus, detailed balance is a sufficient but not necessary condition for the existance of a stationary distribution of the chain.

Assuming that the detailed balance condition is fulfilled, the acceptance ratio can be now written as:

$$a = \frac{P(y)}{P(x)} \tag{4}$$

Just as before, if $a > 1$ then we accept the candidate point $y = x^{t+1}$. Otherwise, if $a < 1$, the candidate y is accepted as x^{t+1} with probability a. The fact that we allow for a reduction in probability is what allows us to wander all over the space; otherwise we would simply be working our way towards a point of (locally) maximal probability.

The transition function $Q(x';x)$ that we will use therefore in order to have $P(x)$ as the stationary distribution of the Markov chain could be written as:

$$Q(x';x) = \int_y \tilde{Q}(y;x) \times \left[\theta\left(\frac{P(y)}{P(x)} > 1\right)\delta(x'-y) + \theta\left(\frac{P(y)}{P(x)} < 1\right)\left\{\frac{P(y)}{P(x)}\delta(x'-y) + \left(1 - \frac{P(y)}{P(x)}\right)\delta(x'-x)\right\}\right] \tag{5}$$

where $\theta(condition)$ is equal to 1 when the condition between brackets is fulfilled and 0 otherwise, and δ is the Dirac function. This transition function fully comprehend the algorithm process. And by integrating it in the limit of $t \to \infty$ steps where $Q_\infty(x) = P(x)$ we can get to the convergence of the Markov chain:

$$P(x') = \int_x Q(x';x)P(x), \tag{6}$$

Notice that solving this integral is not always trivial if detailed balance is not used. Proving that detailed balance holds is often a key step when proving that a Markov chain Monte Carlo simulation will converge to the desired distribution. For the rest, we are completely free in the choice of $\tilde{Q}$: we may change it at will at any moment. This is called *adaptive MCMC*. The choice of $\tilde{Q}$ does influence the performance of the algorithm, though. If we take 'small' steps the probability of accepting a candidate point is probably high, but it will take a longer time to cover the space. If the steps are 'large' then we move over the space quickly, but we may expect that not many of such proposed steps are accepted.

For this kind of methods with local proposal distributions (local in the sense that are unlikely to propose a candidate far from the current sample), if the target distribution has isolated islands of high density, then the chance that we would move from one island to the other is very small. Conversely, if we attempt to make the proposal less local by using one with a high variance the chance then of landing at random on a high density island is remote. Auxiliary variable methods use additional dimensions to aid exploration (suppressing random walks behaviors) and in certain cases to provide a bridge between isolated high density islands. An example of such a method is the so called *Hamiltonian Monte Carlo algorithm*, but since it falls outside of the purpose of this work, we are not gonna cover it.

IV. GIBBS SAMPLING

A. The idea behind

In this section, we introduce Gibbs sampling, one of the most popular MCMC algorithm. This is the MCMC analog of coordinate descent.

The idea behind Gibbs sampling is simple: In order to sample from a joint distribution $P(x)$ where x is a vector of D elements $(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_D)$, we sample each variable at a time, conditioned on the values of all other variables in the distribution. That is, given a joint sample x^t of all the variables, we generate a new sample x^{t+1} by sampling each component at a time, based on the most recent values of the other variables that we have. For example, for a joint distribution where the input vector x has $D = 3$ variables, we use:

$$\begin{aligned} \mathrm{x}_1^{t+1} &\sim P(\mathrm{x}_1 \mid \mathrm{x}_2^t, \mathrm{x}_3^t) \\ \mathrm{x}_2^{t+1} &\sim P(\mathrm{x}_2 \mid \mathrm{x}_1^{t+1}, \mathrm{x}_3^t) \\ \mathrm{x}_3^{t+1} &\sim P(\mathrm{x}_3 \mid \mathrm{x}_1^{t+1}, \mathrm{x}_2^{t+1}) \end{aligned} \tag{7}$$

This can easily be generalized to D variables. The expression $P(\mathrm{x}_i \mid x_{-i})$ is called the full conditional for variable i, where x_{-i} is the set of all the variables except the i component. In general, x_i may only depend on some of the other variables. If we represent $P(x)$ as a graphical model, we can infer the dependencies by looking at i's Markov blanket. That is, the parent nodes of the i node, its "children" nodes, and the parent nodes of those children. In other words, in order to sample x_i, we only need to know the values of the node's neighbours in the graph. It is usually assumed in Gibbs sampling that, whilst $P(x)$ is too complex to draw samples from directly, its conditional distributions $P(\mathrm{x}_i \mid x_{-i})$ are tractable to work with. For many graphical models (but not all) these one-dimensional conditional distributions are straightforward to sample from. Conditional distributions that are not of standard form may still be sampled from by adaptive rejection sampling if the conditional distribution satisfies certain convexity properties (Gilks and Wild, 1992).

An attractive advantage of Gibbs sampling over its competitors is that involves no adjustable parameters. The sampling is done directly from the conditional distributions as opposed to other methods where designing a proposal distribution might be required. Gibbs sampling is therefore a good strategy when one wants to get a model running quickly.

B. Gibbs sampling as special case of MH

It turns out that Gibbs sampling is a special case of MH. Specifically, it is equivalent to using MH with a sequence of proposals of the form:

$$\tilde{Q}(x' \mid x) = P(\mathrm{x'}_i \mid x_{-i}) \mathbb{I}(x'_{-i} = x_{-i}) \tag{8}$$

That is, we move to a new state where x_i is sampled from its full conditional, but x_{-i} is left unchanged. The acceptance rate of each such proposal is 1, so the overall algorithm also has an acceptance rate of 100%. We have

$$\begin{aligned} a &= \frac{P(x')\tilde{Q}(x \mid x')}{P(x)\tilde{Q}(x' \mid x)} \\ &= \frac{P(\mathrm{x'}_i, x_{-i})P(\mathrm{x}_i \mid x_{-i})}{P(\mathrm{x}_i, x_{-i})P(\mathrm{x'}_i \mid x_{-i})} \\ &= \frac{P(\mathrm{x'}_i \mid x_{-i})P(x_{-i})P(\mathrm{x}_i \mid x_{-i})}{P(\mathrm{x}_i \mid x_{-i})P(x_{-i})P(\mathrm{x'}_i \mid x_{-i})} \\ &= 1 \end{aligned} \tag{9}$$

where we used the fact that $x'_{-i} = x_{-i}$, and that $\tilde{Q}(x' \mid x) = P(\mathrm{x'}_i \mid x_{-i})$. Because Gibbs sampling is a Metropolis method, the probability distribution of x^t tends to $P(x)$ as $t \to \infty$, as long as P(x) does not have pathological properties. Note that, in order to show that Gibbs sampling is a MH method, single variable-update Gibbs sampling has been used. That is, in every iteration we just sample one of the conditionals and we take it as a new state

$x^{(t+1)}$. The samples generated by the most common implementation of the Gibbs sampling algorithm (the one described earlier) are a subset of the samples generated by the single variable-update Gibbs sampling. Therefore, the convergence of one algorithm leads to the convergence of the other one.

C. Gibbs sampling in high dimensions

Gibbs sampling suffers from the same defect as simple Metropolis algorithms: the state space is explored by a slow random walk, unless a good parameterization has been chosen that makes the probability distribution $P(x)$ separable. For example, if two variables x_1 and x_2 are strongly correlated, having marginal densities of width L and conditional densities of width α, if $L << \alpha$, it will take lots of iteration to cover the state space properly and get to the point in which the Markov Chain converges. Thus, making difficult to generate an independent sample from the target density. The difference between sampling with a good and bad parametrization can be seen in Fig.(1).

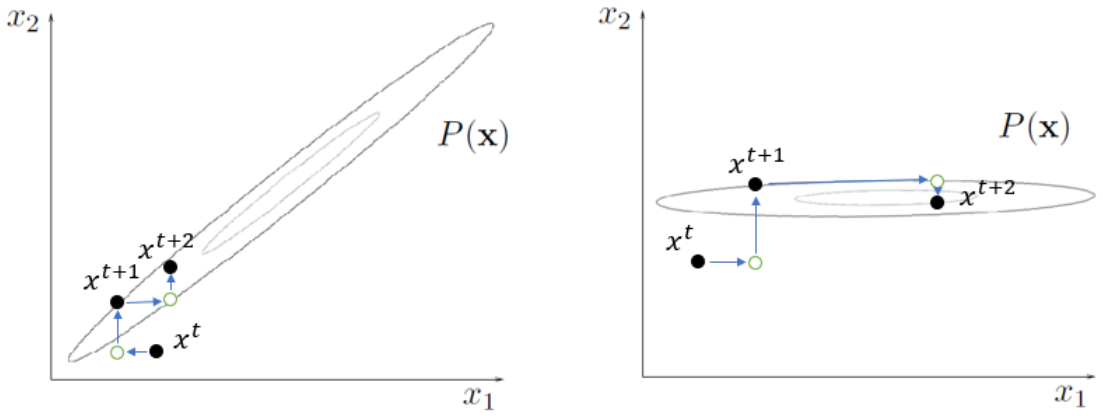


FIG. 1: On the left, a bad parametrization has been chosen. Since the path of sampling it's drawn sequentially using the conditional distributions, slow random walk behaviour occurs. On the right, a good parametrization is used. The variables are not highly correlated. Long steps are more likely to happen. The conditional distributions and the marginal distribution have the same width given this parametrization. That leads to a much more efficient exploration, making faster the process of generating independent samples from the target density.

If the variables are highly correlated, the algorithm will move very slowly through the state space. In some cases we can efficiently sample groups of variables at a time. This is called *blocking Gibbs sampling* or *blocked Gibbs sampling*, and can make much bigger moves through the state space.

D. Collapsed Gibbs sampling

Another variant of Gibbs sampling is the so called *Collapsed Gibbs sampling*. In this algorithm, we analytically integrate out (or marginalize over) as many variables as possible before sampling from the conditional distribution of some variable. This tends to make the sampling much more efficient, since it is sampling in a lower dimensional space. More precisely, suppose the target probability distribution we want to sample from is $p(x, y \mid z)$. Gibbs sampling, we would alternately sample $x^{t+1} \sim P(x \mid y^t, z)$ and $y^{t+1} \sim P(y \mid x^{t+1}, z)$. In collapsed Gibbs sampling, suppose we sample x and integrate out y: $x^{t+1} \sim P(x \mid z)$. Thus the y parameters do not participate in the Markov chain; consequently we can draw conditionally independent samples $y^{t+1} \sim P(y \mid x^{t+1}, z)$. Note that in this case, we are drawing samples from the exact distribution $P(x, y \mid z) = P(y \mid x, z)P(x \mid z)$. The ordering of the variables in the sampling procedure is very important for collapsed Gibbs sampling (to ensure that the resulting Markov chain has the right stationary distribution) since the right ordering might depend on which variables we marginalize out. Drawing samples in this way will make them have much lower variance than samples drawn from the joint state space. This process is called Rao-Blackwellisation, named after the following theorem:

Theorem 1. (Rao-Blackwell). *Let x and y be dependent random variables, and f(x,y) be some scalar function. Then*

$$var_{x,y}[f(x,y)] \geq var_x[\mathbb{E}_y[f(x,y) \mid x]] \tag{10}$$

This theorem guarantees that the variance of the estimate created by analytically integrating out y will always be lower (or rather, will never be higher) than the variance of a direct MC estimate. In collapsed Gibbs, we sample x with y integrated out; the above Rao-Blackwell theorem still applies in this case.

Although it can reduce statistical variance, it is only worth doing if the integrating out can be done quickly, otherwise we will not be able to produce as many samples per second as the naive method.

V. BAYESIAN INFERENCE WITH MCMC

MCMC is particularly useful in Bayesian inference because of the focus on posterior distributions which are often difficult to work with via analytic examination. In these cases, MCMC allows the user to approximate aspects of posterior distributions that cannot be directly calculated (e.g., random samples from the posterior, posterior means, etc.). Bayesian inference uses the information provided by observed data about a (set of) parameter(s), formally the likelihood, to update a prior state of beliefs about a (set of) parameter(s) to become a posterior state of beliefs about a (set of) parameter(s). Formally, Bayes' rule is defined as

$$P(w \mid D) \propto P(D \mid w)P(w) \tag{11}$$

where w indicates a (set of) parameter(s) of interest and D indicates the data, $P(w \mid D)$ indicates the posterior or the probability of w given the data, $P(D \mid w)$ indicates the likelihood or the probability of the data given w, and $P(w)$ indicates the prior or the a–priori probability of w. The important point for this exposition is that the way the data are used to update the prior belief is by examining the likelihood of the data given a certain (set of) value(s) of the parameter(s) of interest. Ideally, one would like to assess this likelihood for every single combination of parameter values. When an analytical expression for this likelihood is available, it can be combined with the prior to derive the posterior analytically. Often times in practice, one does not have access to such an analytical expression. In Bayesian inference, this problem is most often solved via MCMC: drawing a sequence of samples from the posterior, and examining their mean, range, and so on.

VI. CLUSTERING AS PROBABILISTIC INFERENCE FOR A GAUSSIAN MIXTURE MODEL

Gaussian mixture model is the most commonly used mixture model given its simplicity and its broad application.

A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1, \ldots, K\}$, where K is the number of clusters of our dataset. Each observation of the model $\{x_1, ..., x_N\}$ will have a probability π_k (also called mixing coefficient) of belonging to a Gaussian k which is comprised of the parameters $\phi_k = [\mu_k, \Sigma_k]$. Notice that the mixing coefficients are themselves probabilities and must meet this condition $\sum_k^K \pi_k = 1$. The model will be, therefore, constituted by the parameters $\{\pi, \phi\}$.

Now how do we determine the optimal values for these parameters? To achieve this we must ensure that each Gaussian fits the data points belonging to each cluster. This is exactly what maximum likelihood does.

In general, the Gaussian density function is given by:

$$N(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} \mid \Sigma \mid^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{12}$$

Where x represents our data points, D is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively. If we have a dataset comprised of $N = 1000$ three-dimensional points ($D = 3$), then x will be a 1000×3 matrix. μ will be a 1×3 vector, and Σ will be a 3×3 matrix.

Given the fact that we are dealing with more than one Gaussian, we are now going to introduce some additional notation. First, let's suppose we want to know what is the probability that a data point x_i comes from Gaussian k. We can express this as:

$$P(z_i = k \mid x_i) \tag{13}$$

Which reads "given a data point x_i, what is the probability it came from Gaussian k?" In this case, z_i is a latent variable that takes only K possible values $z_i \in \{1, ..., K\}$. It is equal to k when x_i came from Gaussian k. Now let z be the set of the latent variables z of the dataset, hence:

$$\mathrm{z} = \{z_1, ..., z_N\} \tag{14}$$

Every point of the dataset x_i will have a z_i associated that indicates us which Gaussian it comes from. For example, if x_i comes from Gaussian k, then $z_i = k$. Actually, we don't get to see these z_i variables in reality, but knowing its probability of occurrence will be useful in helping us determine the Gaussian mixture parameters, as we discuss later.
Likewise, we can state the following:

$$\pi_k = P(z = k) \tag{15}$$

Which means that the overall probability of observing a point that comes from Gaussian k is actually equivalent to the mixing coefficient for that Gaussian. We know beforehand that each point (and therefore each z) occurs independently of others and that z can be equal to k when k is equal to the cluster the point comes from. Therefore:

$$P(\mathrm{z} \mid \pi) = P(z = 1)^{n_1} P(z = 2)^{n_2} ... P(z = K)^{n_K} = \prod_{k=1}^{K} \pi_k^{n_k} \tag{16}$$

Where n_k is the count of occurrences of $z_i = k$ in z.
The Bayesian network of the model can be represented as the graph shown in Fig.(2).

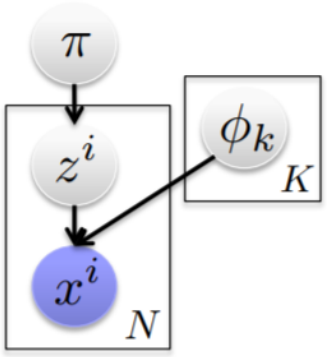


FIG. 2: Representation of model as a Bayesian network.

In a clustering problem, we wanna know the parameters $\{\pi, \phi\}$ of K Gaussians based on a set of observations $\{x_1, ..., x_N\}$ that come from those same K Gaussians . We can compute the likelihood of such a model as follows:

$$P(x_i \mid \pi, \phi) = \sum_{k=1}^{K} \pi_k P(x_i \mid \phi_k) \tag{17}$$

That is, the probability of obtaining the data point x^i given the parameters of the model $\{\pi, \phi\}$.
The clustering problem of such a model is most commonly solved by the EM algorithm. Such an algorithm looks for the $\{\pi, \phi\}$ that maximizes the posterior $P(\pi, \phi \mid x)$. That is, $\{\pi, \phi\}^{MAP}$. The way EM algorithm estimates $\{\pi, \phi\}^{MAP}$ is by performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. By doing this, we eventually will get to a maximum (global or local) in the posterior. But what if we want a full characterization of the posterior? In order to have a measure of uncertainty over the estimation of our parameters, for example. As alternative to solve this problem we have MCMC Gibbs sampling.

VII. GIBBS SAMPLING, A SOLUTION TO THE GMM CLUSTERING PROBLEM

The full joint distribution (from the graphical model) for a Gaussian mixture model is $P(x, \mathrm{z}, \mu, \Sigma, \pi)$. However, since we are looking to solve a clustering problem based on a dataset x, we are actually looking to obtain samples from the conditional distribution $P(\mathrm{z}, \mu, \Sigma, \pi \mid x)$. As we will see in a moment, we can't sample directly from the joint distribution so we will have to construct a Markov chain whose steady state distribution is the one we want to sample from.

The full joint distribution can be expressed, according to our model, as:

$$\begin{aligned} P(x, \mathrm{z}, \mu, \Sigma, \pi) &= P(x \mid \mathrm{z}, \mu, \Sigma) P(\mathrm{z} \mid \pi) P(\pi) \prod_{k=1}^{K} P(\mu_k, \Sigma_k) \\ &= P(x \mid \mathrm{z}, \mu, \Sigma) P(\pi \mid \mathrm{z}) \prod_{k=1}^{K} P(\mu_k, \Sigma_k) \end{aligned} \tag{18}$$

Inferring π of a multinomial model. What is $p(\pi \mid \boldsymbol{z})$?

We have a set of a latent variables z from which each point z_i can come from K Gaussians. Suppose we observe N points, $D = z_1, ..., z_N$, where $z_i \in \{1, ..., K\}$. If we assume the data is iid, the likelihood (as we have seen in section VI when we introduced the latent variables) has the form:

$$P(\mathrm{z} \mid \pi) = \prod_{k=1}^{K} \pi_k^{n_k} \tag{19}$$

For the prior, on the other hand, since the parameter vector π lives in the K-dimensional probability simplex, we need a prior that has support over this simplex. Ideally it would also be conjugate (choosing a conjugate prior for the likelihood function means that the prior and the posterior will be in the same probability distribution family).

Fortunately, the Dirichlet distribution (Appendix A) turns out to be the conjugate prior for the multinomial distribution (our current likelihood distribution). So we will use it as prior:

$$Dir(\pi \mid \alpha) = \frac{1}{\beta(\alpha)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \delta(\sum_k \pi_k - 1) \tag{20}$$

Multiplying the likelihood by the prior, we find that the posterior is also Dirichlet:

$$\begin{aligned} P(\pi \mid \mathrm{z}) &\propto P(\mathrm{z} \mid \pi) P(\pi) \\ &\propto \prod_{k=1}^{K} \pi_k^{\alpha_k + n_k - 1} \propto Dir(\pi \mid \alpha_1 + n_1, ..., \alpha_K + n_K) \end{aligned} \tag{21}$$

We see that the posterior is obtained by adding the prior hyper-parameters (pseudo-counts) α_k to the empirical counts n_k.

Inferring the parameters of a multivariate normal distribution. What is $P(\mu_k, \Sigma_K)$?

Inference techniques in a Gaussian distribution are widely known assuming that the parameters $\phi = (\mu, \Sigma)$ are known. We now discuss how to infer the parameters themselves. We will assume the data has the form $x_i \sim N(\mu, \Sigma)$ for $i = 1, ..., n$. To simplify the presentation, we derive the posterior in three parts: first we compute $P(\mu \mid D, \Sigma)$; then we compute $P(\Sigma \mid D, \mu)$; finally we will discuss about the joint $P(\mu, \Sigma \mid D)$.

- Posterior distribution of μ:

The likelihood has the form $P(D \mid \mu) = N(\bar{x} \mid \mu, \frac{1}{n}\Sigma)$. For simplicity, we will use a conjugate prior. In particular, the Gaussian family is conjugate to itself. And therefore, if $P(\mu) = N(\mu \mid m_0, V_0)$ then we can derive a Gaussian posterior for μ:

$$P(\mu \mid D, \Sigma) = N(\mu \mid m_n, V_n) \tag{22}$$

where $V_n^{-1} = V_0^{-1} + n\Sigma^{-1}$ and $m_n = V_n(\Sigma^{-1}(n\bar{x}) + V_0^{-1}m_0)$.

We can model an uninformative prior by setting $V_0 = \inf I$. In this case, we will have $P(\mu \mid D, \Sigma) = N(\bar{x}, \frac{1}{n}\Sigma)$, so the posterior mean is equal to the maximum likelihood estimator.

- *Posterior distribution of* Σ*:*

The likelihood has the form $P(D \mid \mu, \Sigma) \propto \mid \Sigma \mid^{-\frac{n}{2}} exp\left(-\frac{1}{2}tr(S_\mu \Sigma^{-1})\right)$. The corresponding conjugate prior is known as the inverse Wishart distribution (Appendix B): $IW(\Sigma \mid S_0^{-1}, \nu_0) \propto \mid \Sigma \mid^{-(\nu_0+D+1)/2} exp\left(-\frac{1}{2}tr(S_0\Sigma^{-1})\right)$

Here $\nu_0 > D-1$ is the degrees of freedom, and S_0 is a symmetric positive defined matrix. We see that S_0^{-1} , and $n_0 \widehat{=} \nu_0 + D + 1$ control the strength of the prior, and hence plays a role analogous to the sample size n. These priors are chosen for mathematical convenience and interpretable expressiveness. Multiplying the likelkihood and the prior we find that the posterior is also inverse Wishart:

$$\begin{aligned} P(\Sigma \mid D, \mu) &\propto \mid \Sigma \mid^{-\frac{n}{2}} exp\left(-\frac{1}{2}tr(S_\mu \Sigma^{-1})\right) \mid \Sigma \mid^{-(\nu_0+D+1)/2} exp\left(-\frac{1}{2}tr(S_0\Sigma^{-1})\right) \\ &= \mid \Sigma \mid^{-\frac{n+(\nu_0+D+1)}{2}} exp\left(-\frac{1}{2}tr[\Sigma^{-1}(S_\mu + S_0)]\right) \\ &= IW(\Sigma \mid S_n, \nu_n) \end{aligned} \tag{23}$$

Where $\nu_n = \nu_0 + n$ and $S_n^{-1} = S_0 + S_\nu$. In words, this says that the posterior strength ν_n is the prior strength ν_0 plus the number of observations n, and the posterior scatter matrix S_n is the prior scatter matrix S_0 plus the data scatter matrix S_ν.

- *Posterior distribution of* μ *and* Σ*:*

We now discuss how to compute $P(\mu, \Sigma \mid D)$. The results are a bit complex, so we will skip some of the parts of the derivations in order to be concise. The likelihood has the form:

$$\begin{aligned} P(D \mid \mu, \Sigma) &= (2\pi)^{-nD/2} \mid \Sigma \mid^{-n/2} exp\left(-\frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^T\Sigma^{-1}(x_i - \mu)\right) \\ &= (2\pi)^{-nD/2} \mid \Sigma \mid^{-n/2} exp\left(-\frac{n}{2}(\mu - \bar{x})^T\Sigma^{-1}(\mu - \bar{x})\right) exp\left(-\frac{n}{2}tr(\Sigma^{-1}S_{\bar{x}})\right) \end{aligned} \tag{24}$$

The obvious prior to use is the following: $P(\mu, \Sigma) = N(\mu \mid m_0, V_0)IW(\Sigma \mid S_0, \nu_0)$. The parameters can be interpreted as follows: m_0 is our prior mean for μ, and V_0^{-1} establishes how strongly we believe this prior; and S_0 is (proportional to) our prior mean for Σ, and ν_0 is how strongly we believe this prior.

Unfortunately, this is not conjugate to the likelihood. To see why, note that μ and Σ appear together in a non-factorized way in the likelihood; hence they will also be coupled together in the posterior. This implies that there is no closed form expression for this posterior from which we can sample from. However, the above prior is what is known as a semi-conjugate or conditionally conjugate. A prior distribution is semi-conjugate for a family of likelihood distributions if the prior and (full) conditional posterior distributions belong to the same family of distributions. For semiconjugacy to hold, parameters must typically be independent a priori; that is, their joint prior distribution must be the product of the individual marginal prior distributions. Semiconjugacy may provide an efficient way of sampling from posterior distributions and is used in Gibbs sampling. In this particular case, we could sample $P(\mu, \Sigma \mid D)$ by sampling first from $P(\mu \mid D, \Sigma)$ and then from $P(\Sigma \mid D, \mu,)$.

One can show that the (improper) uninformative prior has the form:

$$\lim_{k \to 0} N(\mu \mid m_0, \Sigma/k)IW(\Sigma \mid S_0, k) \propto \mid \Sigma \mid^{-(\frac{D}{2}+1)} \tag{25}$$

In practice, it is often better to use a weakly informative data-dependent prior. A common choice is to use $S_0 = diag(S_{\bar{x}})/n$, and $\nu_0 = D + 2$, to ensure $\mathbb{E}[\Sigma] = S_0$, and to set $\mu_0 = \bar{x}$ and choose a small V_0^{-1}.

The Gibbs sampler:

From the beginning of the approach, if it happened that we don't want to solve the clustering problem and we just want to infer the parameters of the Gaussians, we could have been tempted to sample from this model without having in consideration any latent variable z. In order to draw a posterior sample of the model we might then try to use Gibbs sampling as it follows:

$$\begin{aligned}\pi^{(t)} &\sim P(\pi \mid \phi^{(t-1)}, x_1, ..., x_N) \\ \phi^{(t)} &\sim P(\phi \mid \pi^{(t)}, x_1, ..., x_N)\end{aligned} \tag{26}$$

Generating therefore a Markov chain $[\{\pi^1, \phi^1\}, \{\pi^2, \phi^2\}, ..., \{\pi^t, \phi^t\}, ...]$. Unfortunately, if we try to build this samplers, we will realize that they don't have a closed form from which we can sample from. However, if we augment the variables of interest $\{\pi, \phi\}$ with variables z we will allow a closed-form from which we are gonna be able to sample from. We therefore, introduce the cluster indicators z into the sampler to generate two "subchains" $[\{\pi^1, \phi^1\}, \{\pi^2, \phi^2\}, ..., \{\pi^t, \phi^t\}, ...]$ and $[\mathrm{z}^1, \mathrm{z}^2, ..., \mathrm{z}^t, ...]$ that will form our Markov chain (even though we might be just interested on the first one).

Suppose we use a semi-conjugate prior for $P(\mu_k, \Sigma_k)$, the full joint distribution of the model would look like this:

$$\begin{aligned}P(\mathrm{z}, \mu, \Sigma, \pi, x) &= P(x \mid \mathrm{z}, \mu, \Sigma) P(\pi \mid \mathrm{z}) \prod_{k=1}^{K} P(\mu_k) P(\Sigma_k) \\ &= \left(\prod_{i=1}^{N} \prod_{k=1}^{K} (\pi_k N(x_i \mid \mu_k, \Sigma_k))^{\theta(z_i = k)} \right) Dir(\pi \mid \alpha) \prod_{k=1}^{K} N(\mu_k \mid m_0, V_0) \times IW(\Sigma_k \mid S_0, \nu_0)\end{aligned} \tag{27}$$

The full conditionals are as follows:

$$\begin{aligned}&P(z_i = k \mid x_i, \mu, \Sigma, \pi) \propto \pi_k N(x_i \mid \mu_k, \Sigma_k) \\ &P(\pi \mid \mathrm{z}) = Dir(\pi \mid \alpha_1 + n_1, ..., \alpha_K + n_K) \\ &P(\mu_k \mid \Sigma_k, \mathrm{z}, x) = N(\mu_k \mid m_k, V_k) \\ &\quad \text{Where:} \quad V_k^{-1} = V_0^{-1} + n_k \Sigma_k^{-1} \\ &\qquad\qquad\ \ m_k = V_k(\Sigma_k^{-1} n_k \bar{x}_k + V_0 - 1 m_0) \\ &\qquad\qquad\ \ n_k = \sum_{i=1}^{N} \theta(z_i = k) \\ &\qquad\qquad\ \ \bar{x}_k = \frac{\sum_{i=1}^{N} \theta(z_i = k) x_i}{n_k} \\ &P(\Sigma_k \mid \mu_k, \mathrm{z}, x) = IW(\Sigma_k \mid S_k, \nu_k) \\ &\quad \text{Where:} \quad S_k = S_0 + \sum_{i=1}^{N} \theta(z_i = k)(x_i - \mu_k)(x_i - \mu_k)^T \\ &\qquad\qquad\ \ \nu_k = \nu_0 + n_k\end{aligned} \tag{28}$$

Where, again, we are using the function $\theta(condition)$ that is equal to 1 when the condition between brackets is fulfilled and 0 otherwise.

We organize our sampler in the following the pseudocode:

$$\begin{aligned}&\pi^{(t)} \sim P(\pi \mid \mathrm{z}^{(t-1)}, x_1, ..., x_N, \alpha) \\ &\text{For k=1,...,K:} \\ &\quad \mu_k^{(t)} \sim P(\mu_k \mid \Sigma_k^{(t-1)}, \mathrm{z}^{(t-1)}, x_1, ..., x_N) \\ &\quad \Sigma_k^{(t)} \sim P(\Sigma_k \mid \mu_k^{(t)}, \mathrm{z}^{(t-1)}, x_1, ..., x_N) \\ &\text{For i=1,...,N:} \\ &\quad z_i^{(t)} \sim P(z_i \mid \pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}, x_i)\end{aligned} \tag{29}$$

VIII. MCMC BURN-IN

In Markov Chain Monte Carlo (MCMC), it's common to throw out the first few states of a Markov chain, this is colloquially known as "Burn-in".

Since we are the ones choosing the parameters from which the Markov chain will start running, that initial states of the chain might not be a good representation of our distribution (a typical sample, as it were). We therefore wait some iterations until the current state of the Markov chain enters a high probability region, a place where the states of the Markov chain are more representative of the distribution you're sampling. All the previous samples that brought us to that "representative" state are thrown out, and we just take in consideration the samples of the chain drawn after the burn-in period.

IX. SAMPLING THE PARAMETERS OF THE IRIS DATASET.

In order to put in practice what we have explained above, we implemented the Gibbs Sampling method on the *Iris dataset* (Appendix C) so as to sample from the distribution $P(z, \mu, \Sigma, \pi \mid x)$. After a *burn-in* of 50 iterations, a typical random sample from the steady state distribution of the MCMC is drawn. The clustering result of such a sample can be seen in Fig.(3).

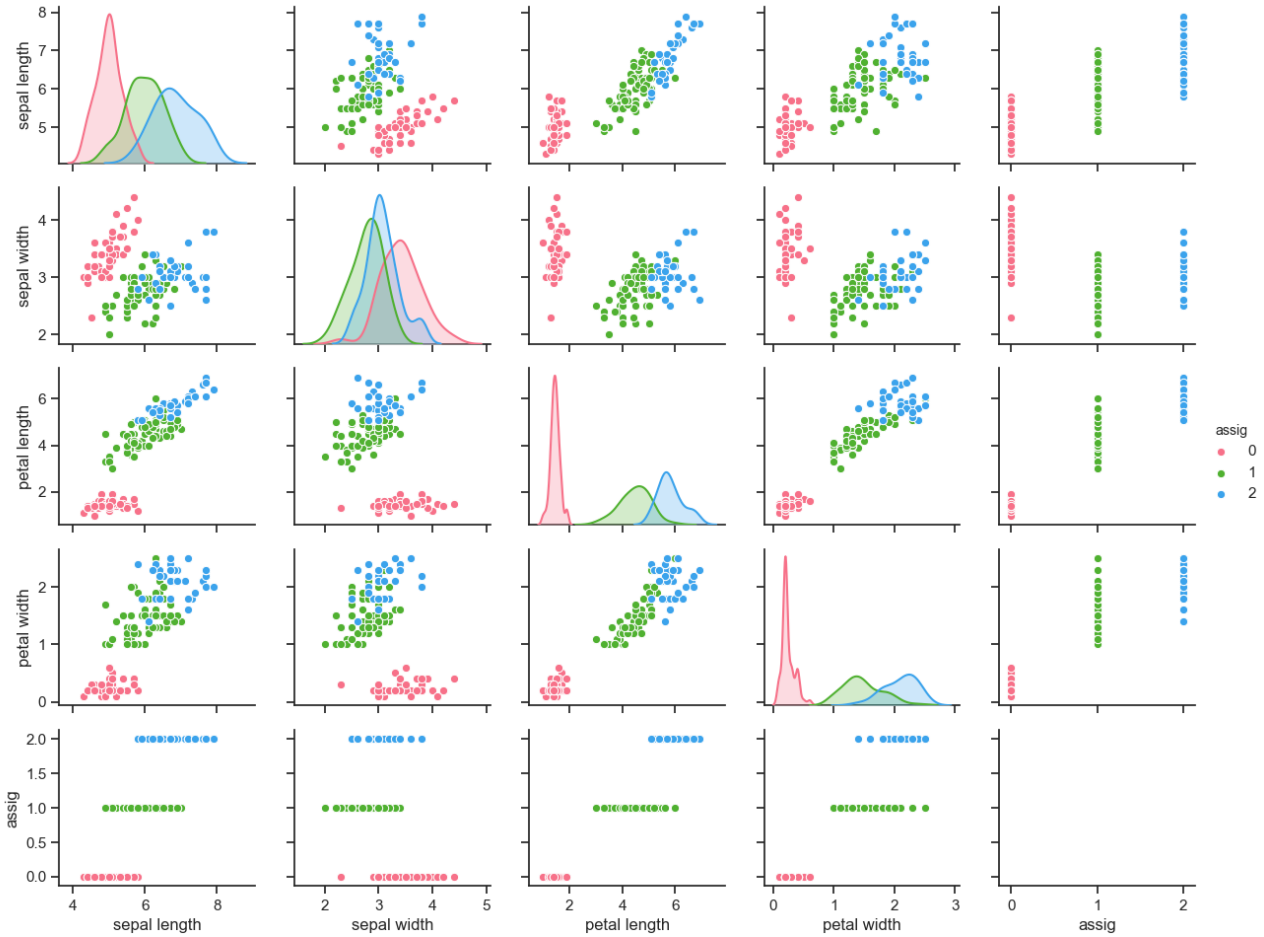


FIG. 3: Representation of one of the possible z state drawn from the Gibbs Sampler. *Assig* makes reference to the assignation vector z.

Notice that this is just a typical sample drawn from the distribution, and therefore it's not the most probable state. The parameters of the Gaussians will change in every iteration of the sampler, and so do the assignations of the points. However, the amount of z assigned to each Gaussian usually doesn't change much from one state to the next one since the other assignations are not as probable as this one (specially for the red Gaussian).

By drawing several samples from the joint distribution, and collecting the parameters of those samples, we could compute the probability of different states and estimate the uncertainty of the most probable combination of parameters. However, the parameters of the model $\{\pi, \phi\}$, and the indicator functions z, are unidentifiable, since we can arbitrarily permute the hidden labels without affecting the likelihood. Consequently, we cannot just take a Monte Carlo average of the samples to compute posterior means, since what one sample considers the parameters for cluster 1 may be what another sample considers the parameters for cluster 2. This can be seen in the sampler by looking the colors of the assignations from one iteration to the other. What we consider the red Gaussian on the current state might switch to blue in the next one. Indeed, if we could average over all modes, we would find $\mathbb{E}[\mu_k \mid D]$ is the same for all k (assuming a symmetric prior). This is called the label switching problem.

This problem does not arise in EM , which just "lock on" to a single mode. However, it arises in any method that visits multiple modes. In 1d problems, one can try to prevent this problem by introducing constraints on the parameters to ensure identifiability, e.g., $\mu_1 < \mu_2 < \mu_3$. However, this does not always work, since the likelihood might overwhelm the prior and cause label switching anyway. Furthermore, this technique does not scale to higher dimensions.

Perhaps the best solution is simply to "not ask" questions that cannot be uniquely identified. For example, instead of asking for the probability that data point i belongs to cluster k, ask for the probability that data points i and j belong to the same cluster. The latter question is invariant to the labeling.

X. APPENDIX

A. Beta and Dirichlet distributions. A probability density over a probability.

In this section we will see some of the basic properties of the Beta distribution and its big brother, the Dirichlet distribution.

The beta distribution is a probability density over a variable π that is a probability, $\pi \in (0, 1)$:

$$P(\pi \mid u_1, u_2) = \frac{1}{\beta(u_1, u_2)} \pi^{u_1 - 1}(1 - \pi)^{u_2 - 1} \tag{30}$$

The parameters u_1, u_2 may take any positive value. The normalizing constant is the beta function,

$$\beta(u_1, u_2) = \frac{\Gamma(u_1)\Gamma(u_2)}{\Gamma(u_1 + u_2)} \tag{31}$$

Special cases include the uniform distribution - $u_1 = 1, u_2 = 1$; the Jeffreys prior - $u_1 = 0.5, u_2 = 0.5$; and the improper Laplace prior $u_1 = 0, u_2 = 0$. If we transform the beta distribution to the corresponding density over the logit $l \equiv ln\frac{\pi}{(1-\pi)}$, we find it is always a pleasant bell-shaped density over l, while the density over π may have singularities at $\pi = 0$ and $\pi = 1$.

More dimensions

The Dirichlet distribution is a density over an K-dimensional vector π whose K components are positive and sum to 1. The beta distribution is a special case of a Dirichlet distribution with $K = 2$. The Dirichlet distribution is parameterized by a measure α (a vector with all coefficients $\alpha_k > 0$):

$$P(\pi \mid \alpha) = \frac{1}{\beta(\alpha)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \delta(\sum_k \pi_k - 1) \equiv Dirichlet^{(K)}(\pi \mid \alpha) \tag{32}$$

The function $\delta(x)$ is the Dirac delta function, which restricts the distribution to the simplex such that π is normalized, i.e., $\sum_k \pi_k = 1$. The normalizing constant of the Dirichlet distribution is:

$$\beta(\alpha) = \prod_k \Gamma(\alpha_k)/\Gamma(\alpha_0) \tag{33}$$

Where $\alpha_0 = \sum_k^K \alpha_k$. The vector α/α_0 is the mean of the probability distribution:

$$\int Dirichlet^{(K)}(\pi \mid \alpha)\pi d^K\pi = \alpha/\alpha_0 \tag{34}$$

B. The inverse Wishart distribution.

The Wishart distribution is the generalization of the Gamma distribution to positive definite matrices. It's mostly used to model our uncertainty in covariance matrices, Σ, or their inverses, $\Lambda = \Sigma^{-1}$.

The pdf of the Wishart is defined as follows:

$$Wi(\Lambda \mid S, \nu) = \frac{1}{Z_{Wi}} \mid \Lambda \mid^{(\nu-D-1)/2} exp\left(-\frac{1}{2}tr(\Lambda S^{-1})\right) \tag{35}$$

Here ν is called the "degrees of freedom" and S is the "scale matrix". The normalization constant for this distribution is the following expression:

$$Z_{Wi} = 2^{\nu D/2}\Gamma_D(\nu/2) \mid S \mid^{\nu/2} \tag{36}$$

where $\Gamma_D(a)$ is the multivariate gamma function:

$$\Gamma_D(x) = \pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma(x + (1-i)/2) \tag{37}$$

Hence,

$$\Gamma_D(\nu_0/2) = \prod_{i=1}^{D}\Gamma\left(\frac{\nu + 1 - i)}{2}\right) \tag{38}$$

The normalization constant only exists (and hence the pdf is only well defined) if $\nu > D - 1$.

There is a connection between the Wishart distribution and the Gaussian. In particular, let $x_i \sim N(0, \Sigma)$. Then the scatter matrix $S = \sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T = \sum_{i=1}^{N} x_i x_i^T$ has a Wishart distribution: $S \sim Wi(\Sigma, 1)$. Hence $\mathbb{E}[S] = N\Sigma$. More generally, one can show that the mean and mode of $Wi(S, \nu)$ are given by

$$mean = \nu S, mode = (\nu - D - 1)S \tag{39}$$

where the mode only exists if $\nu > D + 1$.

Inverse Wishart distribution

If $\Sigma^{-1} \sim Wi(S, \nu)$ then $\Sigma \sim IW(S^{-1}, \nu + D + 1)$, where IW is the inverse Wishart, the multidimensional generalization of the inverse Gamma. It is defined as follows, for $\nu > D - 1$ and $S > 0$:

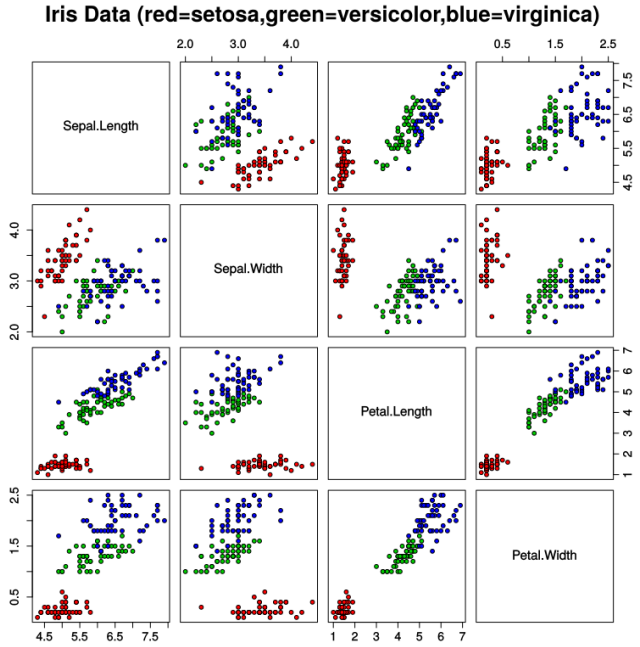
$$IW(\Sigma \mid S, \nu) = \frac{1}{Z_{IW}} \mid \Sigma \mid^{-(\nu+D+1)/2} exp\left(-\frac{1}{2}tr(S^{-1}\Sigma^{-1})\right) \tag{40}$$

$$Z_{IW} = 2^{\nu D/2}\Gamma_D(\nu/2) \mid S \mid^{-\nu/2} \tag{41}$$

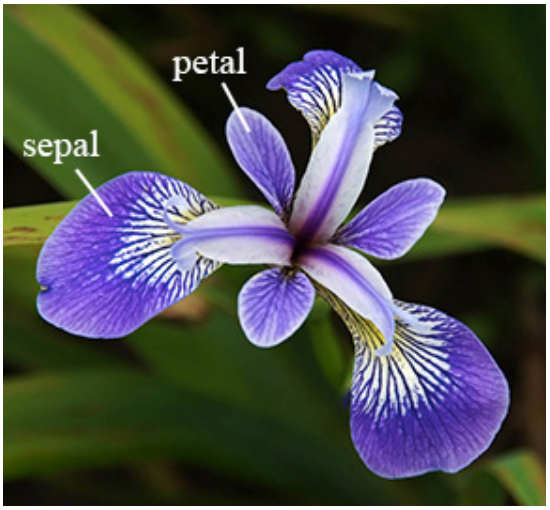
C. The Iris data set.

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper *"The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis"*.

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.



(a) Scatter plot of the data set



(b) Iris flower.

FIG. 4: Iris data set.

[1] Ronald Kleiss "Monte Carlo: Techniques and Theory" IMAPP, Radboud University of Nijmegen (2019).
[2] David J.C. MacKay "Information Theory, Inference, and Learning Algorithms" Cambridge University Press (1995).
[3] Kevin P. Murphy "Machine Learning: A Probabilistic Perspective" The MIT Press (2012)
[4] Trevor Hastie Robert Tibshirani Jerome Friedman "The Elements of Statistical Learning" Springer (2009)
[5] Roger B. Grosse, David K. Duvenaud. "Testing MCMC code" Cornell University arXiv:1412.5218 (2014)
[6] David M. Blei "Bayesian Mixture Models and the Gibbs Sampler" Columbia University (2015).
[7] Radford M. Neal, "Bayesian learning for neural networks" University of Toronto (1995).
[8] Jun S. Li, "The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem" Probabilistic Graphical Models 10-708 (2014).
[9] Eric P. Xing, "Advanced topics in MCMC" Carnegie Mellon University (2013).
[10] Philip Resnik, Eric Hardisty, "Gibbs sampling for the uninitiated" University of Maryland (2010).
[11] Ilker Yildirim, "Bayesian Inference: Gibbs Sampling" University of Rochester (2012).