# Probabilistic Graphical Models

Beltrán Castro

January 2023

## 1 Random Variables

A random variable is a variable whose value is determined by a random process. There are two types of random variables: hidden and observable.

A hidden random variable is one that is not directly observable, but can be inferred from other observations. For example, in a medical study, the hidden random variable might be a patient's genetic predisposition to a certain disease. This variable is not directly observable, but it can be inferred from other observations such as family history or symptoms.

An observable random variable is one that can be directly measured or observed. For example, in the same medical study, the observable random variable might be a patient's blood pressure. This variable can be directly measured, and it is not inferred from other observations.

In summary, a hidden random variable is not directly observable, but it can be inferred from other observations, while an observable random variable can be directly measured or observed.

### 1.1 Priors

A prior is a probability distribution that represents our initial belief about the value of a parameter or a set of parameters before we have any data. It is based on previous knowledge or experience.

A prior is considered a hidden random variable because it represents our initial belief about the value of a parameter or set of parameters before we have any data. It is not directly observable but inferred from previous knowledge or experience.

A prior can be considered a form of a random variable in Bayesian statistics, where the prior represents our uncertainty about the true value of a parameter before we have any data. The prior can be a continuous or discrete probability distribution and it is used to update our belief about the parameter after we have observed data.

For example, in Bayesian linear regression, the prior on the coefficients would be a random variable. In a Bayesian model, we start by specifying a prior on the parameters, and then update it as we observe data. It's important to mention that the prior can be a subjective or objective, depending on the scenario. A subjective prior is based on expert knowledge, intuition or personal belief, while an objective prior is based on the available data and it aims to be as non-informative as possible.

In summary, a prior is a probability distribution that represents our initial belief about the value of a parameter or set of parameters before we have any data, and it can be considered as a form of a random variable in Bayesian statistics.

## 2 Markov Chain Monte Carlo algorithms

Markov Chain Monte Carlo (MCMC) is a class of algorithms for sampling from a probability distribution. The idea is to construct a Markov Chain, whose stationary distribution is the distribution of interest. By running the chain for a long time and collecting samples from the chain, we can obtain a sample from the target distribution.

MCMC algorithms are particularly useful for sampling from high-dimensional and complex distributions that are difficult or computationally expensive to sample from directly.

MCMC algorithms are based on a Markov Chain, which is a sequence of random variables that satisfies the Markov property, meaning that the probability of the next state depends only on the current state, and not on any previous states. The sequence of states generated by an MCMC algorithm is called a Markov Chain.

There are several different types of MCMC algorithms, such as Metropolis-Hastings, Gibbs sampling and Hamiltonian Monte Carlo. Each algorithm has its own set of assumptions and requirements and is suited to different types of problems.

All MCMC algorithms have a few elements in common, however. They all start with an initial state and use a transition kernel to generate new states. The transition kernel is defined by a probability distribution that determines the probability of moving from the current state to a new state. The transition kernel should be chosen in a way that the chain converges to the target distribution.

It's important to note that MCMC algorithms require a large number of iterations to converge to the target distribution.

### 2.1 Proposal distribution

In Markov Chain Monte Carlo (MCMC) methods, a proposal distribution is a probability distribution used to generate new samples. The proposal distribution is used to generate a candidate sample, and the candidate sample is then accepted or rejected based on the acceptance probability. The acceptance probability is defined as the ratio of the target density at the candidate sample and the current sample.

A proposal distribution is typically chosen to be easy to sample from and to have a high acceptance rate. The choice of proposal distribution can have a big impact on the performance of the MCMC algorithm. Commonly used proposal distributions include Gaussian, uniform, and Cauchy distributions.

In the Metropolis-Hastings algorithm, the proposal distribution is also known as the transition kernel. The transition kernel should be chosen in a way that the chain converges to the target distribution. In other words, the proposal distribution should be chosen so that it generates samples that are likely to be accepted, and it should be able to explore the entire target distribution.

In the Gibbs sampler, the proposal distribution is the conditional distribution of each variable given the current values of the other variables. This is why it is important to have closed form solutions for the conditional distributions in order to use the Gibbs sampler.

It's important to note that the choice of proposal distribution can affect the performance of the MCMC algorithm and how long it takes to converge to the target distribution. Therefore, it's often useful to try different proposal distributions.

## 2.2 Gibbs Sampling

### 2.2.1 How does a Gibbs sampler work?

A Gibbs sampler is a Markov Chain Monte Carlo (MCMC) algorithm that is used to generate samples from a multivariate distribution. It works by iteratively sampling from the marginal distributions of each variable, given the current values of the other variables.
The general idea is as follows:

1. Start with an initial state, which is a set of values for all the variables in the distribution.
2. At each iteration, select one variable at random, and sample a new value for that variable from its marginal distribution, given the current values of the other variables.
3. Repeat step 2 for a certain number of iterations, or until the samples converge to the desired distribution.

The Gibbs sampler is a useful method for generating samples from complex distributions that are difficult or computationally expensive to sample from directly.

One key feature of the Gibbs sampler is that it can be used to estimate the posterior distributions of Bayesian models, which often have complex dependencies between variables.

It's also important to note that in some cases the marginal distributions may not have closed form, in this case an approximation method like Metropolis-Hastings algorithm can be used.

Gibbs sampling can be computationally intensive, especially when the dimension of the variables is high. Additionally, the choice of initial values for the variables can affect the performance of the sampler, so it is often important to carefully choose good initial values or use techniques like burn-in and thinning to ensure that the samples are representative of the true distribution.

### 2.2.2 How do we implement a Gibbs sampler and what equations are neccessary for doing it?

To implement a Gibbs sampler, you need to do the following steps:

1. Define the joint probability distribution of all the variables of interest. This is typically done as part of a Bayesian model, where the joint distribution is given by the product of the likelihood function and the prior distributions of the variables.
2. Define the marginal distributions of each variable, given the current values of the other variables. These marginal distributions can be derived from the joint distribution by integrating out the other variables.
3. Choose an initial state, which is a set of values for all the variables. This can be done randomly or using some other method such as a maximum likelihood estimate.
4. Iterate over the variables and sample new values for each variable from its marginal distribution, given the current values of the other variables.
5. Repeat step 4 for a certain number of iterations, or until the samples converge to the desired distribution.

The equations necessary for implementing the Gibbs sampler depend on the specific problem you are trying to solve. In general, you will need the following:

- The joint probability distribution of the variables.
- The marginal distributions of each variable, given the current values of the other variables.

- The transition probabilities for the Markov Chain (if the variables are discrete) or the probability density function of the distribution (if the variables are continuous).
- The transition kernel which defines the conditional probability of the next state given the current one.

It's important to note that the choice of the transition kernel is important and it should be chosen in a way that the chain converge to the target distribution.

It's also important to keep in mind that the implementation of the Gibbs sampler may vary depending on the complexity of the problem and the availability of closed form solutions for the marginal and conditional distributions.

#### 2.2.3 How can we implement a Gibbs sampler for estimating the mean and variance of a normal distribution?

To implement a Gibbs sampler for estimating the mean and variance of a normal distribution, you can follow these steps:

1. Define the likelihood function of the data, assuming it follows a normal distribution with unknown mean and variance. The likelihood function is given by:
$$p(x|\mu,\sigma^2) = \left(\frac{1}{(2\pi\sigma^2)^{(n/2)}}\right) * exp\left(-\left(\frac{1}{2\sigma^2}\right)\sum(x_i-\mu)^2\right) \tag{1}$$
2. Define prior distributions for the mean and variance. For example, you can assume a normal prior for the mean, with a known mean and variance, and an inverse-gamma prior for the variance.
3. Choose an initial state for the mean and variance, for example, you can set the mean and variance to the sample mean and variance of the data.
4. At each iteration of the Gibbs sampler:
   - Sample a new value for the mean from its conditional distribution given the current value of the variance and the data. The conditional distribution is also a normal distribution, with a mean and variance that can be computed using the data and the current value of the variance.
   - Sample a new value for the variance from its conditional distribution given the current value of the mean and the data. The conditional distribution is an inverse-gamma distribution, with parameters that can be computed using the data, the current value of the mean, and the prior distribution of the variance.
5. Repeat steps 4 for a certain number of iterations, or until the samples converge to the desired distribution.

It's important to note that the choice of the prior distributions can affect the performance of the sampler. It's also important to check the convergence of the sampler, and in case of non-convergence, use techniques like burn-in and thinning to ensure that the samples are representative of the true distribution.

Also, you may consider using some diagnostic tools to ensure that the chain has reached the equilibrium and the samples are indeed coming from the target distribution.

## 2.3 Metropolis-Hastings Sampling

A Metropolis-Hastings (MH) sampler is a Markov Chain Monte Carlo (MCMC) algorithm that is used to generate samples from a target probability distribution. The algorithm works as follows:

1. Start with an initial state, $x_0$, chosen from an arbitrary distribution.

2. At each iteration $t$, generate a new candidate state, $x_{candidate}$, from a proposal distribution, which is a symmetric distribution (pdf) centered at $x_t$, the current state.
3. Calculate the acceptance probability, $\alpha$, as the ratio of the target density at $x_{candidate}$ and the current state, $\alpha = min\left(1, \frac{p(x_{candidate})}{p(x_t)}\right)$
4. Generate a random variable, $u$, from a uniform distribution between 0 and 1.
5. Compare $u$ to $\alpha$.
   - If $u < \alpha$, accept the candidate state as the new state, $x_{t+1} = x_{candidate}$.
   - If $u >= \alpha$, reject the candidate state, and $x_{t+1} = x_t$.
6. Repeat steps 2 through 5 for a certain number of iterations or until the samples converge to the desired distribution.

The Metropolis-Hastings algorithm is an instance of a more general class of algorithm called Metropolis algorithm. The idea behind the Metropolis algorithm is to generate a sequence of states that can be used to approximate the target distribution.

It's important to note that the choice of proposal distribution can affect the performance of the MH sampler and the rate of convergence. A good proposal distribution should have a high acceptance rate.

### 2.3.1 What equations are needed to implement a Metropolis-Hastings sampler?

To implement a Metropolis-Hastings (MH) sampler, you need the following equations:

- The target distribution: This is the distribution that you want to sample from. It is typically a probability density function (pdf) that you need to evaluate at any given point.
- The proposal distribution: This is the distribution that you use to generate candidate states. It is typically a symmetric probability density function (pdf) centered at the current state. The proposal distribution should be easy to sample from and have a high acceptance rate.
- The acceptance probability: This is the probability of accepting a candidate state as the new state. It is given by the equation $\alpha = min\left(1, \frac{p(x_{candidate})}{p(x_t)}\right)$, where $p(x)$ is the target density at $x$.
- The transition kernel: The transition kernel is the probability of moving from the current state $x_t$ to the candidate state $x_{candidate}$ given the proposal distribution.
- The acceptance-rejection step: A random number $u$ is generated from a uniform distribution between 0 and 1, and compare it to the acceptance probability.

## 2.4 What are the differences between a Gibbs sampler and a Metropolis-Hastings sampler?

A Gibbs sampler and a Metropolis-Hastings sampler are both Markov Chain Monte Carlo (MCMC) algorithms that are used to generate samples from a multivariate distribution. However, there are some key differences between the two methods.

- The main difference between a Gibbs sampler and a Metropolis-Hastings sampler is the way they generate new samples. In a Gibbs sampler, new samples are generated by iteratively sampling from the marginal distributions of each variable, given the current values of the other variables. In a Metropolis-Hastings sampler, new samples are generated by proposing new values for all the variables and accepting or rejecting these proposed values based on the acceptance probability.
- The acceptance probability in Metropolis-Hastings is defined as the ratio of the target density at the proposed state and the current state, if the proposed state is accepted, otherwise it is 1. This means that the Metropolis-Hastings algorithm may generate some samples from regions of low probability.

- A Gibbs sampler is generally easier to implement when the conditional distributions of the variables have closed form solutions. A Metropolis-Hastings sampler is more flexible and can be used in cases where the conditional distributions do not have closed form solutions.
- A Gibbs sampler is guaranteed to converge to the target distribution as long as the Markov Chain is ergodic, while the convergence of a Metropolis-Hastings sampler is not guaranteed.
- A Gibbs sampler can be more efficient than a Metropolis-Hastings sampler when the variables are highly correlated.

## 3 Importance Sampling

Importance sampling is a technique used to estimate expectations of a function with respect to a target distribution, while Markov Chain Monte Carlo (MCMC) methods are a class of algorithms used to generate samples from a target distribution.

Importance sampling is a technique used in statistics and machine learning to estimate the expectation of a function with respect to a probability distribution, when the function is difficult or impossible to evaluate directly. The basic idea behind importance sampling is to sample from a different probability distribution, known as the importance distribution, and to use the samples to estimate the expectation of the function with respect to the target distribution.

The key to importance sampling is to find an importance distribution that is easy to sample from, but that is similar to the target distribution in the areas where the function of interest has high values. The samples drawn from the importance distribution are then used to estimate the expectation of the function with respect to the target distribution using the following formula:

$$E[f(x)] = \frac{1}{N} * \sum \frac{f(x_i) * p(x_i)}{q(x_i)} \tag{2}$$

where $E[f(x)]$ is the expectation of the function $f(x)$ with respect to the target distribution $p(x)$, $x_i$ are samples drawn from the importance distribution $q(x)$, $N$ is the number of samples, and $p(x_i)$ and $q(x_i)$ are the probabilities of $x_i$ under the target and importance distributions, respectively.

Importance sampling is useful in many applications such as simulation-based estimation, Bayesian inference, and reinforcement learning. It allows to estimate the expectation of a function with respect to a target distribution, without the need to evaluate the function directly or to compute the normalizing constant of the target distribution.

In summary, Importance sampling is a technique used to estimate the expectation of a function with respect to a probability distribution, when the function is difficult or impossible to evaluate directly. It samples from a different probability distribution, known as the importance distribution, and use the samples to estimate the expectation of the function with respect to the target distribution.

### 3.1 Steps of an Importance Sampler

An importance sampler consists of the following steps:

1. Define the target distribution: The target distribution is the probability distribution that we want to estimate the expectation of a function with respect to. It is typically difficult or impossible to sample directly from this distribution.
2. Define the importance distribution: The importance distribution is a probability distribution that is easy to sample from, but that is similar to the target distribution in the areas where the function of interest has high values.

3. Sample from the importance distribution: Using a suitable sampling method, draw a set of samples from the importance distribution.

4. Evaluate the function of interest: For each sample, evaluate the function of interest.

5. Compute the importance weights: The importance weights are computed as the ratio of the target distribution probability to the importance distribution probability for each sample.

6. Estimate the expectation: The expectation of the function of interest with respect to the target distribution can be estimated as the weighted average of the function evaluations using the importance weights.

7. Estimate the variance: The variance of the importance sampling estimator is also an important aspect to consider, and can be computed using the sample variance of the importance weights.

8. Repeat steps 3 to 7 multiple times to reduce the variance of the estimator

In summary, Importance Sampling is a technique that uses a set of samples from an importance distribution that is easy to sample from but similar to the target distribution in regions where the function of interest has high values, to estimate the expectation of that function with respect to the target distribution. The importance weight is used to weigh each sample and then the expectation is estimated as the weighted average of the function evaluations.

## 4 Expectation-Maximization algorithm

An expectation–maximization (EM) algorithm is an iterative method to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables.
The EM iteration alternates between 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. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

### 4.1 Gaussian Mixture Models (GMM)

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes that the data is generated from a mixture of several Gaussian distributions with unknown parameters. In other words, it models the data as a combination of multiple Gaussian distributions, rather than a single Gaussian distribution.

The generative process of a GMM is as follows:

1. Select one of the Gaussian components with a probability proportional to the weight of the component.

2. Generate a data point from the selected Gaussian component.

3. Repeat the process to generate multiple data points.

In more formal terms, a GMM is defined by a set of parameters, including the mean and covariance matrix of each Gaussian component, and the weight of each component. The probability density function (PDF) of a GMM is given by a weighted sum of the PDFs of the individual Gaussian components:

$$p(x) = \sum w_i * N(x; \mu_i, \Sigma_i) \tag{3}$$

where $x$ is the data point, $w_i$ is the weight of the i-th component, $\mu_i$ is the mean of the i-th component and $\Sigma_i$ is the covariance matrix of the i-th component. $N(x; \mu_i, \Sigma_i)$ is the PDF of a multivariate Gaussian distribution with mean $\mu_i$ and covariance $\Sigma_i$.

A GMM can be used for a variety of tasks such as density estimation, clustering, and classification. The most common method for estimation of GMM parameters is the Expectation-Maximization (EM) algorithm.

## 4.2 Estimating the parameters of a GMM with EM

The Expectation-Maximization (EM) algorithm is a widely used method for estimating the parameters of a Gaussian Mixture Model (GMM). The EM algorithm consists of two steps: the Expectation (E) step and the Maximization (M) step.

- E-Step
  - In the E-step, the algorithm estimates the probability that each data point belongs to each of the Gaussian components, given the current estimates of the parameters. This is done by computing the posterior probability of the i-th data point belonging to the j-th component, denoted by $P(j|i)$, using the current estimates of the parameters and the likelihood of the data given the parameters.
- M-Step
  - In the M-step, the algorithm uses the estimated posterior probabilities to update the parameters of the GMM. Specifically, the new estimates of the parameters are obtained by maximizing the expected complete-data log-likelihood with respect to the parameters.
  - The new estimates of the mean of the j-th component is calculated as the weighted average of the data points, where the weight is the estimated posterior probability of the j-th component for each data point.
  - The new estimates of the covariance matrix of the j-th component is calculated as the weighted average of the outer product of the data points and their deviation from the j-th component mean, where the weight is the estimated posterior probability of the j-th component for each data point.
  - The new estimates of the weight of the j-th component is calculated as the proportion of data points that are assigned to that component.

These two steps (E-step and M-step) are repeated until the change in the log-likelihood function between consecutive iterations is less than a predefined threshold or a maximum number of iterations is reached.

# 5 Gibbs Sampling to estimate the parameters of a GMM

Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method that can be used to generate samples from a multivariate distribution by iteratively sampling from the marginal distributions of each variable. It can also be used to estimate the parameters of a Gaussian Mixture Model (GMM), although it is not as widely used as the Expectation-Maximization (EM) algorithm for this purpose.

The basic idea behind applying Gibbs sampling to a GMM is to iteratively sample the latent variables (which indicate the component assignment of each data point) and the parameters of the GMM (means, variances, and mixture weights) given the current values of the other variables.

The main steps of the Gibbs sampling algorithm are:

1. Initialize the parameters of the GMM and the latent variables.
2. For each iteration
   - For each data point, sample the mixture component indicator variable for that data point, given the current estimates of the parameters and the indicator variables for the other data points.
     This can be done by computing the conditional probability of each component given the data point and the current estimates of the parameters, and then drawing a sample from the categorical distribution defined by these probabilities.
   - For each mixture component, sample the parameters of the GMM (mean, variance, and mixture weight) given the current latent variables.
3. Repeat step 2 until convergence.

It's worth noting that the Gibbs sampling for GMM is computationally more expensive than the EM algorithm, and the convergence rate is also slower. However, the Gibbs sampling can be useful in cases where the EM algorithm may have difficulty in converging to the correct solution

## 6 Variational Inference

Variational Inference (VI) is a method for approximating a complex probability distribution by a simpler one, called the variational distribution. The goal of VI is to find the best approximation of the target distribution, given a set of observations.

The basic idea behind VI is to define a family of distributions, called the variational family, that approximates the target distribution. Then, the goal is to find the best member of this family that approximates the target distribution. This is done by minimizing the Kullback-Leibler (KL) divergence between the target distribution and the variational distribution. The KL divergence is a measure of the difference between two probability distributions.

VI can be seen as a form of approximate Bayesian inference, where instead of computing the exact posterior distribution, we approximate it using a simpler distribution.

## 7 Kullback-Leibler divergence

Kullback-Leibler divergence, KL divergence or relative entropy is a measure of the difference between two probability distributions.
It is a non-symmetric measure, meaning that $D_{KL}(P||Q)$ is not equal to $D_{KL}(Q||P)$. It is defined as:

$$D_{KL}(P||Q) = \sum P(x) * log\left(\frac{P(x)}{Q(x)}\right) \tag{4}$$

which is equivalent to:

$$D_{KL}(P||Q) = -\sum P(x) * log\left(\frac{Q(x)}{P(x)}\right) \tag{5}$$

where $P$ and $Q$ are two probability distributions over the same sample space, $x$.

KL divergence is always non-negative, with equality if and only if $P$ and $Q$ are the same distribution. It's worth noting that KL divergence is not a distance metric, it doesn't satisfy the triangular inequality property. It is also not symmetric.

The KL divergence measures the average amount of information lost when approximating $P$ with $Q$.
The KL divergence is also commonly used as a measure of the difference between the true distribution of data and a model's distribution, which is why it is used to optimize the variational parameters in Variational Inference.

KL divergence is a popular measure of the dissimilarity between two probability distributions, and it has many applications in machine learning and information theory.

## 8 Variational Bayes

Variational Bayes (VB) is a specific application of VI to Bayesian models, where the goal is to approximate the posterior distribution of the model parameters given the observations.

In Variational Bayes the model parameters are treated as random variables and the goal is to approximate the posterior distribution of these variables using a simpler distribution. This is done by defining a family of distributions, called the variational family, that approximates the posterior distribution. Then, the goal is to find the best member of this family that approximates the posterior distribution by minimizing the KL divergence between the posterior and the variational distribution.

## 9 Variational Autoencoders

A variational autoencoder (VAE) is a type of generative model that is optimized to learn a compact, continuous representation of the data, known as the latent variable. The optimization process is performed via a variant of stochastic gradient descent called variational inference. Variational inference is a method of approximating a complex probability distribution using a simpler, more tractable distribution.

In a VAE, the encoder network learns to map the input data to a set of parameters of the latent variable distribution, while the decoder network learns to map the latent variable back to the original data space. The VAE objective function is a combination of a reconstruction loss, which measures the difference between the input data and the decoder output, and a regularization term, known as the KL divergence, which encourages the latent variable to approximate a predefined prior distribution.

The traditional auto-encoder approach is a non-probabilistic model. Traditional autoencoders only learn to reconstruct the input data and don't learn the probability distribution of the data.

VAE can be seen as an extension of the traditional autoencoder, in that it learns not just to reconstruct the input data, but also to learn the underlying probability distribution of the data. It also provides a way to sample new data that is similar to the training data.

### 9.1 Amortization

In a variational autoencoder (VAE), amortization refers to the process of approximating the intractable posterior distribution of the latent variable using the encoder network. The encoder network learns to map the input data to a set of parameters of the latent variable distribution, such as the mean and variance. In other words, the encoder network learns a mapping from the input data space to the space of the latent variables' parameters, this mapping is known as the *amortized* posterior distribution.

In traditional variational inference, the posterior distribution is approximated using a separate variational distribution for each input. This is computationally expensive and not scalable to large datasets. Amortization allows for a more efficient inference by using a single neural network to approximate the posterior for all inputs, thus reducing the number of parameters to be learned.

In summary, amortization is a technique that allows the VAE to efficiently approximate the intractable posterior distribution of the latent variable using the encoder network, rather than using a separate variational distribution for each input. This makes the model more computationally efficient and scalable to large datasets.

### 9.2 The reparametrization trick

The reparametrization trick is a technique used in variational autoencoders (VAEs) to make the optimization of the VAE objective function more stable and efficient. The reparametrization trick is used to make the sampling process of the VAE differentiable with respect to the model parameters, which is necessary for the optimization process to be performed using backpropagation and gradient-based optimization methods.

The basic idea behind the reparametrization trick is to introduce a set of auxiliary random variables that are independent of the model parameters. These auxiliary variables are used to sample from the latent variable distribution. The encoder network learns to map the input data to the parameters of the latent variable distribution, such as the mean and variance, and the auxiliary variables are used to sample from this distribution.

The reparametrization trick works by reparametrizing the sampling process of the latent variable in terms of the mean and variance of the latent variable distribution, and a set of independent random variables. For example, if the latent variable follows a normal distribution, the reparametrization trick consists of sampling from this distribution using the mean and variance learned by the encoder network, and an independent random variable sampled from a standard normal distribution.

In summary, the reparametrization trick is a technique used in VAEs to make the sampling process of the latent variable differentiable with respect to the model parameters. It allows for the optimization of the VAE objective function to be performed using gradient-based optimization methods, which makes the training process more stable and efficient.

### 9.3 Loss of a VAE

The loss of a variational autoencoder (VAE) is typically calculated by combining two terms: the reconstruction loss and the KL divergence.

The reconstruction loss measures the difference between the input data and the decoder output. It is used to ensure that the VAE can accurately reconstruct the input data. The reconstruction loss is calculated as the mean squared error (MSE) (Equation 6) or the mean absolute error (MAE) (Equation 7) between the input data and the decoder output.

$$L_{rec} = \frac{1}{N}\sum (x_i - f(z_i))^2 \tag{6}$$

or

$$L_{rec} = \frac{1}{N}\sum |x_i - f(z_i)| \tag{7}$$

The KL divergence is a regularization term that encourages the latent variable to approximate a predefined prior distribution. The KL divergence measures the difference between the learned latent variable distribution and the prior distribution. The KL divergence is calculated as the divergence between the encoded latent variable distribution and the prior.

$$L_{KL} = \frac{1}{N}\sum D_{KL}(Q(z|x_i)||P(z)) \tag{8}$$

The goal of VAE is to find the optimal set of parameters for the encoder and decoder networks that minimize the reconstruction loss while maintaining the KL divergence as low as possible. The overall loss is the sum of the reconstruction loss and the KL divergence, and it is the quantity that is minimized during the training process.

$$L = L_{rec} + L_{KL} \tag{9}$$

## 10 Exam Example

1. What is the difference between a directed and an undirected graphical model?
    - In a directed graphical model (also known as a Bayesian network), the edges in the graph represent causal relationships between variables, while in an undirected graphical model (also known as a Markov network), the edges represent only dependencies between variables.
2. What are the two main components of a probabilistic graphical model?
    - The two main components of a probabilistic graphical model are the graph structure and the set of local probability distributions associated with each variable.
3. How do you use a probabilistic graphical model to represent a joint probability distribution?
    - In a probabilistic graphical model, the joint probability distribution is represented by the product of the local probability distributions associated with each variable, with the dependencies between variables encoded in the graph structure.

4. What is the sum-product algorithm and how is it used in probabilistic graphical models?

    - The sum-product algorithm is a message-passing algorithm used to perform inference in probabilistic graphical models, particularly in Bayesian networks. The algorithm uses the graph structure to propagate information about the variables throughout the network.

5. What is the difference between a Bayesian network and a Markov network?

    - A Bayesian network is a directed graphical model with a specific type of graph structure, where each node represents a random variable and each directed edge represents a causal relationship between variables. A Markov network is an undirected graphical model with no explicit causal structure.

6. How do you use variable elimination to perform inference in a probabilistic graphical model?

    - Variable elimination is a technique for performing inference in probabilistic graphical models by iteratively integrating out (eliminating) variables from the joint distribution. By eliminating variables that are not of interest or not easily observable, the remaining variables can be more efficiently inferred.

7. What are some applications of probabilistic graphical models?

    - Probabilistic graphical models have wide range of application in various domains such as bioinformatics, natural language processing, computer vision, speech recognition, recommender systems and many more.