Contents

1 Introduction

1.1 Proteins

1.1.1 Prediction of Structure

Proteins are essential macromolecules of living organisms [1]. While having diverse essential biological functions ranging from DNA replication, forming cytoskeletal structures, transporting oxygen within multicellular organisms, functioning as enzymes and converting one molecule into another [2] analysing their threedimensional structure and knowledge of their functions is also a crucial information for development of new drugs, better crops and even synthetic biofuels. Since large scale sequencing-technologies were developed, the availability of protein sequence information has been grown exponentially. To determine the threedimensional structure of proteins experimental approaches are X-ray crystallography or nuclear magnetic resonance. The drawback of these methods is that they are expensive in terms of time and experimental requirements and not feasible for all proteins [1, 3]. This is where researchers increasingly rely on computational methods. Fully successfull computational methods for protein structure prediction are still lacking in order to provide information for the large fraction of sequences whose structures are not determined experimentally [4].

1.1.2 Structure of Proteins

Proteins are built of a chain of amino acids with varying lengths which are joined through peptide bonds. The peptide bond is planar. There are 20 natural amino acids, that have a common basic structure, with a central C_α-atom, a carboxyl group, an amino group and a side chain R. The peptide bond is formed between the carboxyl group of the first amino acid and the amino group of the second amino acid, see figure 1. The chain of N-C_α-C-N-C_α-C-... atoms is called the backbone. The amino acids differ from each other through their side chain R. The side chains can have variable properties like ability to form hydrogen bonds, charge or hydrophobicity [2].

As can be seen in figure 2 the protein folds into a defined threedimensional structure driven by interactions between side chains, backbone and solvent. This particular 3D structure is the basis on which protein binding sides, functions, biochemical properties and interaction with other molecules can be derived from [5].

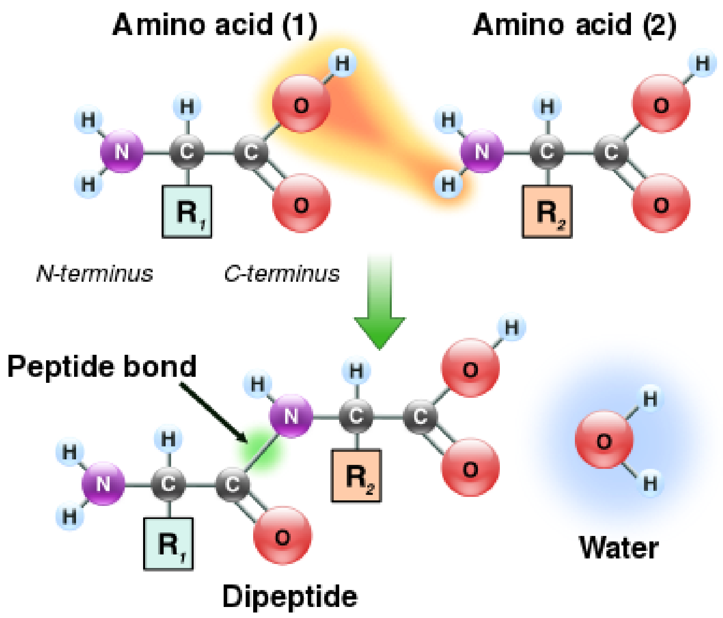


Figure 1: Two amino acids with with a central C-atom, a carboxyl group, an amino group and a side chain R each forming the peptide bond and releasing one water molecule. Retrieved from: https://en.wikipedia.org/wiki/Amino_acid. Date of access: 13.05.2019.

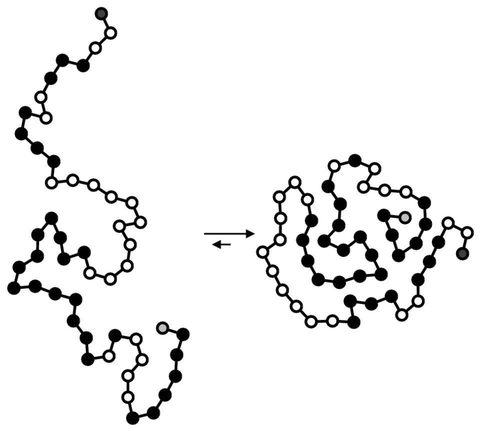


Figure 2: Example of protein folding driven by hydrophobic forces. In the schematic representation of the folding can be seen that the protein is folded so that hydrophobic amino acids (black dots) are shielded from the aqueous environment. Retrieved from: https://en.wikipedia.org/wiki/Protein_folding. Date of access: 13.05.2019.

A protein has three dihedral angles, namely ω, ϕ and ψ between the backbone atomes C-N, N-C_α and C_α-C respectively, as can be depicted in figure 3. The ω adapts to one of two configuratoins, a value of 0 or 180°. The configuration of ϕ and ψ are limitied by unfavourable close contacts with neighboring atoms and pose steric constraints in the conformational space. The allowed values for ϕ and ψ were first determined by G. N. Ramachandran. The allowed values are indicated in a two-dimensionl plot of ϕ against ψ that is called Ramachandran plot after G. N. Ramachandran who first determinated the allowed values for ϕ and ψ [6], see figure 4. The characteristics of a protein can be further described by differing bond lengths between the backbone atoms [7][8].

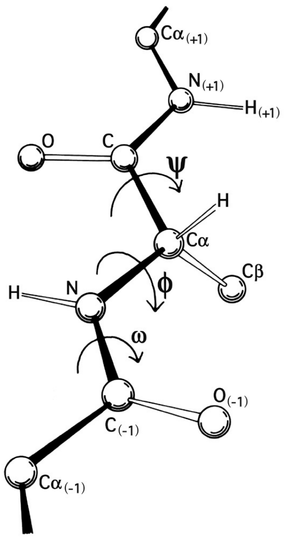


Figure 3: Model of the backbone dihedral angles ϕ and ψ (and ω). Retrieved from: https://en.wikibooks.org/wiki/Structural_Biochemistry/Proteins/Ramachandran_Plot. Date of access: 14.05.2019.

1.1.3 Ramachandran Plot

A Ramachandran Plot showed in figure 4 shows the sterically inhibited regions in white, the regions without any steric clashes in red which are namely the alpha-helical and beta-sheet conformations. The yellow regions show allowed regions of limited stability, the let handed alpha helix [8].

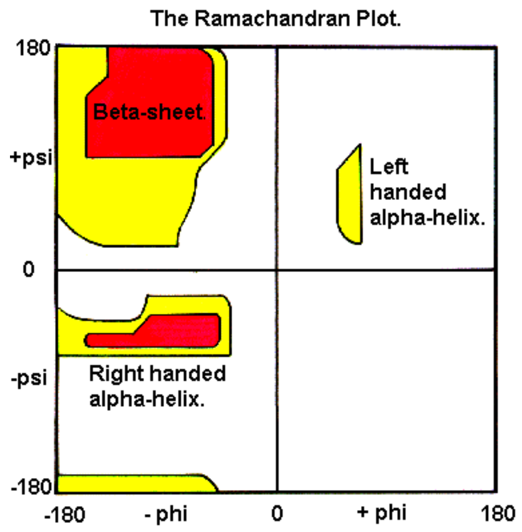


Figure 4: Example of a Ramachandran Plot of a Protein visualiszing the dihedral angles ϕ and ψ of a Protein. Retrieved from: http://www.cryst.bbk.ac.uk/PPS95/course/3_geometry/rama.html. Date of access: 14.05.2019.

1.2 Prediction methods and CASP

Protein structure prediction is still an unsolved problem which is approached by many research groups. To evaluate and assess different modelling approaches a scientific biennial competition called Critical Assessment of Protein Structures (CASP). In every CASP participants attempt to computationally model the 3D structures of proteins of which the structure has been solved experimentally already but it has not been released publically [9].

The protein prediction techniques where no template comparison or evolutionary information is used and the fold is predicted on the amino acid sequence only is called *ab initio*. The first principle-based *ab initio* approaches start with an unfolded conformation and simulate physical forces to fold the protein into its 3D structure.

1.3 Machine Learning

2 Material and Methods

2.1 Bayes' Theorem

Bayesian Statistics is a probabilistic approach to statistical inference. The fundamental difference to the more commonly known Frequentism is that instead of obtaining a singular value one is provided with a distribution, called the posterior (distribution) [10]. The posterior contains a range of values with corresponding probabilities for each value, from which a single sample can be drawn. It is based on Bayes' theorem,

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

where z represents the latent random variables and x represents the observation. The posterior distribution $p(z|x)$ is given by the likelihood $p(x|z)$ times the prior $p(z)$ divided by the normalization constant $p(x)$. The normalization constant $p(x)$ is often untracable. Which leads to the assumption that [11]:

$$p(z|x) \propto p(x|z)p(z)$$

$$posterior \propto likelihood \times prior$$

2.2 Hidden Markov Model

A hidden Markov model (HMM) is a markov chain of latent variables from which observed variables can be obtained. A sequence of events is generated. The latent variable z_i is conditioned on the previous z_{i-1}, so it is governed by the transition probability $p(z_i|z_{i-1})$. The observed variables x_i are conditioned on the respective z_i and they are governed by the emission probability $p(x_i|z_i)$ as can be seen in figure 5. This way the model can represent complex time dependencies. The latent states are not directly observable thus they are regarded as hidden and it is called a *hidden* markov model [12].

Hidden Markov models quickly became popular as a tool for supervised machine learning and they are now commonly used for sequence analysis [13].

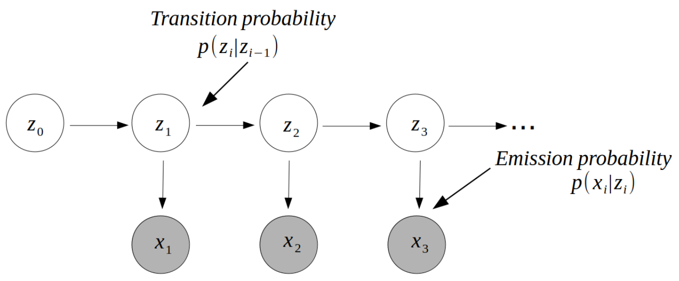


Figure 5: Hidden Markov Model with latent states z and observed states x. Transition probabilites govern the latentn variables z and emission probabilites govern x.

2.3 Neural Networks

[14] Neural networks (NNs) were initially developed in analogy to neurons as a model for information processing and learning in the brain. NNs are paramterized graphical models with interconnected units. The connection between two units j and i is denoted by a weight w_{ij}. A NN can be seen as a weight directed graph.

Architectures of neural networks are feedforward, recurrent and layered. In the context of this project layered feedforward and recurrent NNs are used. As can be seen in figure 6 mulitlayer feedforward NNs consist of multiple layers with visible input and output layers and multiple hidden layers.

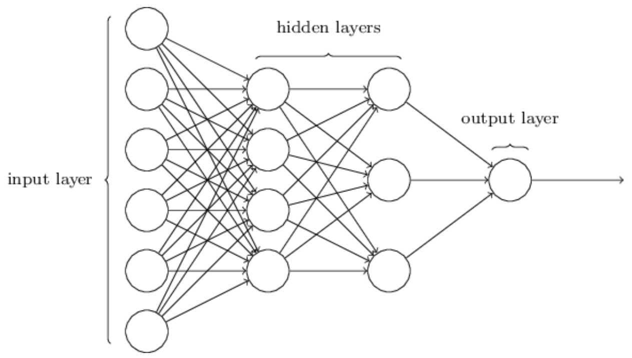


Figure 6: Neural Network with input layer, two hidden layers and output layer. Retrieved from https://chatbotslife.com/how-neural-networks-work-ff4c7ad371f7. Date of access: 22.03.2019

A recurrent neural network (RNN) has at least one feedback loop, that means neurons can feed its output signal back to the inputs of other neurons. It has a profound impact on the learning capability and the performance of the network [15]. In the scope of this project a RNN with one hidden layer is used.

2.3.1 Activation Functions

Activation functions are the nonlinear units in neural networks and determine the output behaviour of the nodes. In order to predict posterior probabilities that lie in range of 0 and 1 the weight w_{ij} which is a linear function is transformed using a nonlinear function that is called activation function. There are different forms of activation functions [16].

Sigmoid Function

The sigmoid function also called logistic sigmoid function is defined by:

$$Sigmoid(x) = \frac{1}{1 + \exp(-x)}$$

It maps the whole real axis into a finite interval.

Softmax Function

The softmax function is known as the normalized exponential and can be regarded as a multiclass generalisation of the logistic sigmoid. It is defined by:

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

It is used in classification cases with more than two classes. The softmax function is indivually applied to evey input element. After application all elements sum to 1 [16].

Rectified Linear Unit Function

The Rectified Linear Unit (ReLU) is described by:

$$ReLU(x) = \max(0, x)$$

Applied to x negative values become 0 wheareas positive numbers it returns back that value [17].

2.4 Distributions

In the DMM two distributions are used to sample amino acids and angles. For the amino acids a von Mises distribution is used, for the amino acids an one hot categorical distribution.

Von Mises Distribution

The von Mises distribution is a continuous normal distribution on a circle and the most prominent among the univariate circular distributions. The probability density function for the angle x is given by

$$f(x|\mu, \kappa) = \frac{e^{\kappa \cos(x-\mu)}}{2\pi I_0(\kappa)}$$

where $I_0(\kappa)$ is the modified Bessel function of order p. The parameters μ and $\frac{1}{\kappa}$ are equivalents of the mean μ and the variance σ^2 in the normal distribution. κ is a measure of concentration. As can be seen in figure 7 it means that for κ equal to zero or being very small the distribution is uniform or close to uniform. For large κ the distribution becomes very concentrated at the angle μ with κ being a measure of concentration [18, 19].

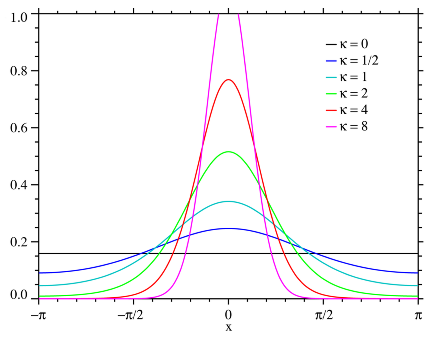


Figure 7: Plot of the von Mises distribution with different k. Retrieved from https://en.wikipedia.org/wiki/Von_Mises_distribution. Date of access: 22.03.2019.

One hot Categorical

The amino acids where sampled using a one hot categorical distribution, in order to avoid any ranking of the aminoacids.

2.5 Pyro and probabilistic programming

The model was implemented in Pyro, a deep probabilistic programming language written in Python and based on the framework PyTorch [20]. A deep probabilistic programming language combines deep neural networks and probabilistic models. Deep learning is automatic hierarhical and supervised representation learning. They enable including Bayesian statistics and making probabilistic assumptions in powerful machine-learning applications and in this way also representing uncertainty. Output data is generated by sampling from a latent probability distribution. The model is trained upon an inference procedure which uses observed output data to fit the latent distribution [21].

In figure 8 a graphical model represents the general structure. N is the number of observed datapoints $\{x_i\}$. As it is a generative model, every datapoint is generated by a local random variable z_i. θ is a global parameter because all datapoints depend on it [this paragraph is not finished yet. I m still writing on it]

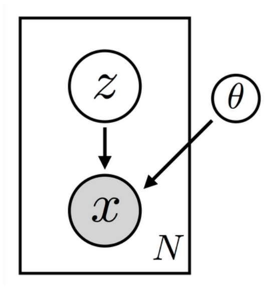


Figure 8: Graphical Representation

2.6 Deep Markov Model

The model used for predicting the protein structure is a Deep (hidden) Markov Model (DMM) retrieved from the deep probabilistic programming framework Pyro [20]. A DMM is a fusion of a HMM with NNs in between the nodes. The structure can be seen in figure 9. The nodes are represented by cirlces, z_0 is the initial state and the black squares represent neural networks. An observation of length three is shown $\{x_1, x_2, x_3\}$ corresponding to a sequence of latent variables $\{z_1, z_2, z_3\}$. The joint distribution is:

$$p(x_{123}, z_{123}) = p(x_1|z_1)p(x_2|z_2)p(x_3|z_3)p(z_1)p(z_2|z_1)p(z_3|z_2)$$

The observations x are independet fromfigure each other and only depend on the latent variable z_t at the current timestep. The markov property of the model can also be seen: each latent $z_t$is conditioned on its previous z_{t-1}but independent of all previous latent states $\{z_{t-2}, z_{t-3}, ...\}$. The latent variables z are called the latent space. The first z_1 is conditioned on z_0 is a trainable parameter.

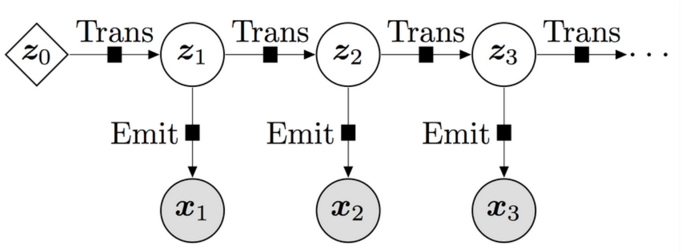


Figure 9: Deep Markov Model for three time steps with latent variables z, observed variables x and Neural Networks (black squares) in between the nodes.Sigmoid Function

In our specific case the observed variables x are the mean of the the of the observed angles μ_ϕ and μ_ψ of the dihedral angles ϕ and φ of the protein and the κ_ϕ and κ_ψ as well as the amino acids aa at every specific timestep t, see figure

10. For simplicity the bond lengths between the atoms are considered constant. The third angle ω was not considered in the model.

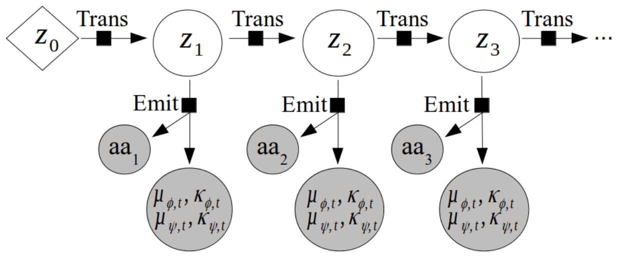


Figure 10: Deep Markov Model for three timesteps with latent variabls z and observed variables for amino acids aa, for the means μ_ϕ and μ_ψ and for the variation κ_ϕ and κ_ψ.

The observation likelihoods, the probability distributions which control the observations of the mean of the angles $p(\mu_{\phi,t}|z_t)$ and $p(\mu_{\psi,t}|z_t)$ and the variances $p(\kappa_{\phi,t}|z_t)$ and $p(\kappa_{\psi,t}|z_t)$ are von Mises distributions, which as explained in 2.4 is used to describe angles. For the amino acids $p(aa_t|z_t)$ one hot categorical distributions for which the PyTorch implementation is used [22].

2.6.1 Neural Networks

The black solid squares represent the non-linearities in the model. There are two different kind of NNs. Transitional NNs ('Trans' in Fig. 10) controls the dynamic of the latent variables and Emitter NNs ('Emit' in Fig. 10) control the how the observations depend on the latent dynamics.

The NNs make our model a 'deep' markov model and allow to capture complex dynamics. It can be destinguished between the part of linear transformations and the part of nonlinear activation function.

Emitter NN

The emitter NN is a forward NN with four hidden layers and a dimension of 100. The latent state z_t is the input. ReLU is used as an activation function except for the last nonlinear transformation of the probability of the mean μ where sigmoid is used and the amino acids where softmax is used.

Transition NN

The transition NN is a gated transition function. The architecture is different to the emitter NN because the output are the mean and the variance which are required to define a gaussian distribution. Also, they both need to have the same dimension as the latent space. The mean is a sum of two terms of which only one depends non-linearily on the input. This makes it possible to support both linear and nonlinear dynamics, so part of the dynamic of the latent space can be linear while the remainder can stay nonlinear.

Gradient Descent

[here I still have to write something]

2.6.2 Model

In the model we first sample z_1 that is conditioned on z_0 which is a trainable parameter. Once we have sampled z_1 we can sample $z_2 \sim p(z_2|z_1)$ and so on. For sampling all z a for-loop is implemented. The probability distribution $p(z_t|z_{t-1})$ is defined by the parameters of the mean and variance which are called 'z_loc' and 'z_scale' in pyro. They are computed at a specific timestep t using the transition NN and are conditioned on the previous timestep $t-1$.

Once z_t is sampled at a specific timestep, we need to observe the datapoint x_t. Therfor, z_t is passed through the emitter NN and the probability for the aa and the mean μ and the variance κ are output. Using the von Mises distribution and μ and κ the angles are sampled for ϕ and ψ and a one-hot categorical distribution is used for sampling the amino acids.

The posterior of the model, the probability of the parameters of the NNs θ and the probability of the latent variable z conditioned on the data x is defined by equation 1, the likelihood of the data x conditioned on the parameters θ and z times the prior of θ times the prior of z. z is estimated by approximating the posterior, using Bayesian Statistics, whereas the estimation of θ is a point estimation and not bayesian in this model. Ideally the estimation of θ would also be defined by a posterior distribution. However having several NNs with multiple layers, being bayesian about θ would not be computationally feasible at this point.

Following this, the posterior is approximated by equation 2.

$$p(\theta, z|x) \propto p(x|\theta, z)p(\theta)p(z) \tag{1}$$

$$p(\theta, z|x) \approx p_\theta(x|z)p(z) \tag{2}$$

2.6.3 Inference

Stochastic Variational Inference (SVI) was used as an inference method. A variational lower bound is optimised to approximate the data log-likelihood because the exact posterior is intractable.

The model has the probability density function [23]:

$$p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$$

It is assumed that $p_\theta(x, z)$ consists of various probability distributions p_i. θ describes the parameter space of the model. To find θ that describes the model best, the log evidence is maximised:

$$\theta_{\max} = \arg\max_{\theta} \log p_\theta(x)$$

The log evidence is given by the integral over the joint probability function:

$$\log p_\theta(x) = \log \int p_{\theta_{\max}}(x, z)\, dz$$

The integral is unavailable in closed form or requires exponential time to compute, but it is the denominator of the posterior which we also want to compute:

$$p_{\theta_{\max}}(z|x) = \frac{p_{\theta_{\max}}(x, z)}{\int p_{\theta_{\max}}(x, z)\, dz}$$

That is why the posterior $p_{\theta_{\max}}(z|x)$ is approximated by a guide $q_\phi(z)$ which is a paramterized distribution with variational parameters ϕ. Since the guide is an approximation to the posterior $p_{\theta_{\max}}(z|x)$ it needs to provide a valid joint probability density over the latent space. As described by Blei, Kucukelbir and McAuliffe [24] an optimization problem is set up by finding the guide which is closest to the posterior. An appropriate objective function has to be defined: the *ELBO*: $\underline{\text{e}}$vidence $\underline{\text{l}}$ower $\underline{\text{bo}}$und:

$$ELBO \equiv \mathbb{E}_{q_\phi(z)}[\log p_\theta(x, z) - \log q_\phi(z)]$$

By assumption the log probabilites indside the expectation function can be computed. The *ELBO* is a lower bound to the log evidence and can never become bigger than the log evidence. In an ideal case it would be equal to the log evidence:

$$\log p_\theta(x) \geq ELBO$$

The gap between the *ELBO* and the log evidence is given by the Kullback-Leibler (KL) divergence [23, 20], a measure of the difference between two probability distributions [25], between the guide and the posterior:

$$\log p_\theta(x) - ELBO = KL(p_\theta(z|x) \| q_\phi(z))$$

It can be seen, that the bigger the *ELBO* becomes the closer it will be to the log evidence and the better the approximation. Hence we want to maximize the *ELBO* and an optimization problem is created. The smaller the KL divergence betweent the posterior $p_\theta(z|x)$ and the guide $q_\phi(z)$ the better the approximation is [23, 20].

Using variational inference, the sampling problem is transferred into an optimization problem [24].

2.6.4 Guide

As explained in chapter 2.6.3, the purpose of the guide which is given by the variational distribution is to provide a parameterised approximation to the exact posterior $p(z_{1:T}|x_{1:T})$. The size of the parameters θ is very high even for small

N. It has to describe a very high-dimensional distribution to approximate the posterior. To make this enormous growth of the dataset possible amortisation is used.

Amortization

The space which is optimised grows with the number of sequences N which makes calculating local variational parameters for every z_i problematic. Instead amortization is used which learns a single parametric function $f(\cdot)$. It is sharing one global parameter θ across all datapoints $q(z_i)$ instead of multiple local parameters so that the guide is given by:

$$q(z, \beta) = q(\beta) \prod_{n=1}^{N} q(z_i | f(x_i))$$

In order to construct a parametric function $f(\cdot)$ that supports different sequence lenghts a recurrent neural network (RNN) was implemented as part of the guide. In figure 11 a graphical representation of the guide is shown. $\{x_1, x_2, x_3\}$ at the bottom of the figure are the observations which are the inputs from right to the left to the RNN. The RNN outputs hidden states $\{h_1, h_2, h_3\}$ which are together with the previous latent state z_{t-1} the input to the combiner. The combiner outputs the mean and the variance of the conditional distribution $q(z_t|z_{t-1}, x_{t:T})$ which have to be in the form of a gaussion distribution. Only for timestep $t = 1$ z_1 is conditioned on a trainable variational parameter z_0^q.

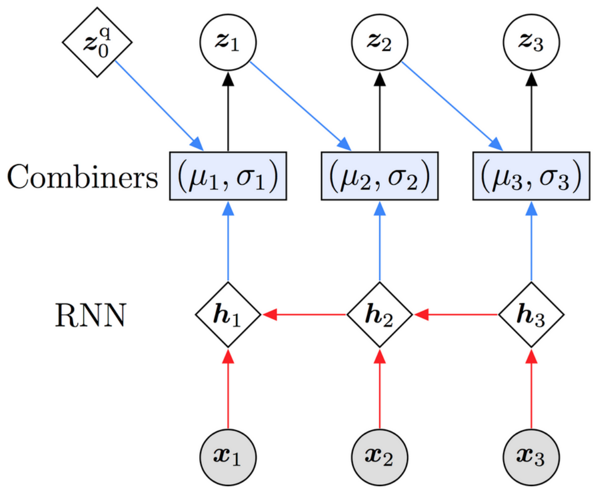


Figure 11: The guide rolled out for T=3 time steps with the observation $\{x_1, x_2, x_3\}$, the hidden states $\{h_1, h_2, h_3\}$, the latent variables $\{z_1, z_2, z_3\}$.

2.6.5 Training

The model was trained on a GeForce GTX 1080 Graphics Card. In order to run on GPU the parameter of the model had to be made GPU compatible and had to be loaded onto the GPU server. For training the dataset is partitioned into mini-batches. The order of the sequences is shuffled during training to keep the learning general.

As an optimizer adam optimizer is used which includes gradient clipping in order to avoid problems like vanishing or explodeing gradients that can typically occur during training.

The stochastic variational inference uses a stochastic gradient estimator to take gradient steps on an objective function. The objective function which is the ELBO cannot be computed analytically so the parameters will be updated following Monte Carlo gradient estimates. The ELBO is the lower bound to the log evidence $\log p(D)$. Every step being taken that maximizes the ELBO moves the guide $q(\cdot)$ closer to the exact posterior. The ELBO has probability involved, therefore it is hard to compute the gradient estimator. Automatic differentiation and full reparameterization for all latent variables throughout the model is used in pyro.

KL annealing

As another optimization strategy KL annealing was used. As explained in section 2.6.3 the ELBO is part of the variational inference and it consists of the following two terms:

$$ELBO = \mathbb{E}_{q(z_{1:T})}[\log p(x_{1:T}|z_{1:T})] - \mathbb{E}_{q(z_{1:T})}[\log q(z_{1:T}) - \log p(z_{1:T})]$$

The expectation of the log likelihood which measures the model fit in the first term and the second term being the KL divergence term which serves to regularize the approximate posterior.

The later term can be a quite strong regularizer and especially in the early stages of training it tends to favor regions of the loss surface that contain lots of bad local optima. To avoid these bad local optima the KL divergence terms are annealed by multiplying them with the annealing factor which is a scalar that ranges between zero and one, as it was also applied in reference [26]:

$$ELBO = \mathbb{E}_{q(z_{1:T})}[\log p(x_{1:T}|z_{1:T})] - annealingfactor \times \mathbb{E}_{q(z_{1:T})}[\log q(z_{1:T}) - \log p(z_{1:T})]$$

The annealing factor is set to a very low value near to zero at the beginning of the training and then rises to its final value 1 during the course of training. The number of epochs which were influenced by the annealing factor was set to 1000 epochs. That means that starting with zero over the range of 1000 epochs the annealing factor rose in a linear way to 1.

2.6.6 Normalization Flows

The approximations of the posterior distribution are constructed using normalization flows which transforms a simple initial density into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. The performance of the posterior approximation is improved by deep auto-regressive networks that use an auto-regressive dependency structure. Performing inference with normalization flows provides a tighter, modified variational lower bound (ELBO) [27].

2.6.7 Data and Loading

The dataset was retrieved from MolProbity which is a protein database of high quality structures (source: http://kinemage.biochem.duke.edu/databases/top500.php, date of access: 29.03.2019). The dataset called "top500" consists of 500 protein sequences and was reduced to a dataset of 402 proteins not including sequences with chain breaks or that caused errors upon parsing. From this dataset a textfile including the phi and psi angles and the amino acid was created which was used as input to the model.

The protein data loader is a script that is preprocessing the data for the DMM. The amino acids were encoded using a one hot categorical distribution. The dataset was divided into minibatches of varying sizes.

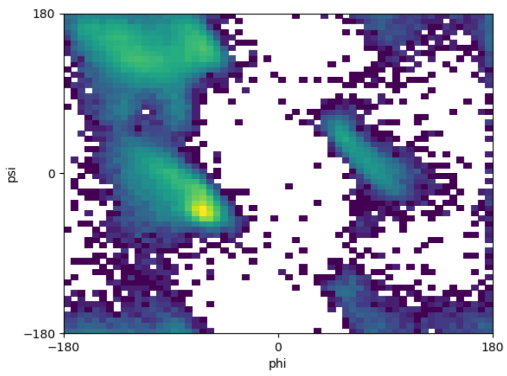
3 Results

3.1 Modelling with Categorical encoding for amino acids (DMM One Hot)

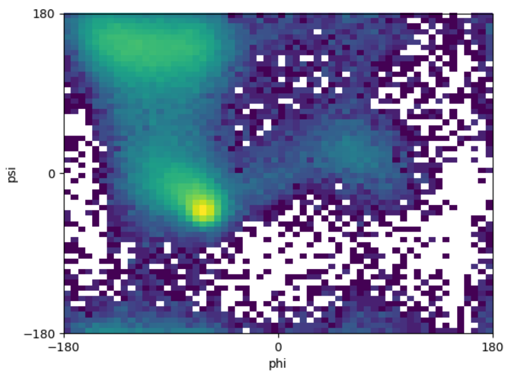
Figure 12a shows the Ramachandran Plot created by the protein dataset. The ϕ angles are plotted on the x-axis and the ψ angles are plotted on the y-axis. An accumulation of angles can be seen at the top left corner, in the lower left quadrant and on the right-hand side which represents according to the sterically possible configurations beta-sheets, right-handed alpha helixes and left-handed alpha helixes, compare with Figure 4. In Figure 12b results can be seen from modelling the ϕ and ψ angles with the HMM. It can be seen that results are approximating expected results when compared to Figure 12a. The accumulations of the beta-sheet in the top left corner and the accumulation for the right-handed alpha helix in the bottom left corner are approximately predicted as expected when comparing Figure 12a and Figure 12b. The accumulation of angles in the right handside of the Ramachandran plot is only vaguely approximated by the model.

In Figure 12c the negative ELBO can be seen converging. It can be seen that the model was stopped and restarted twice after 1000 iterations and a second time after 5000 iterations. The impact of the annealing factor on the ELBO for 1000 iterations after starting the model can be seen.

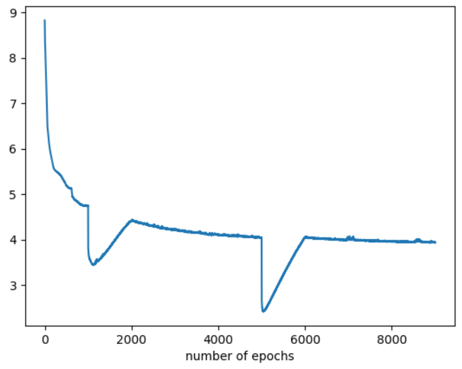
In Figure 12d the sampled means of the protein angls are shown. It can be seen that the means correlate with the sampled angles in Figure 12b.



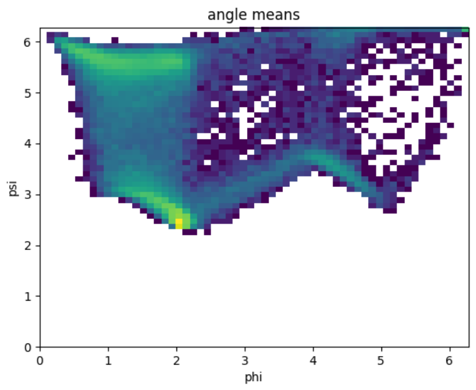
(a) Ramachandran plot created by the data set



(b) Ramachandran plot created by HMM



(c) The negative ELBO of HMM



(d) rtzrzrtzrztrtrzrttzrtzrtzt

Figure 12: Ramachandran plots created by the the ϕ and ψ angles of the 402 amino acid sequences and results

Limiting κ

Limiting the variance of the von Mises distribution of the angles κ resulted in

3.2 Modelling with One Hot encoding for amino acids

Model	time per iteration	number of iterations	total training time
DMM_best	$\sim$ 48 sec.	9000	119 h
DMM 4	$\sim$ 57 sec.	6368	101 h
DMM 5	$\sim$ 374 sec.	1695	176 h
DMM_1k	$\sim$ 516 sec.	1461	209 h
DMM_k12	$\sim$ 545 sec.	660	100 h
DMM_one_hot	$\sim$ 358 sec.	2149	214 h

Table 1

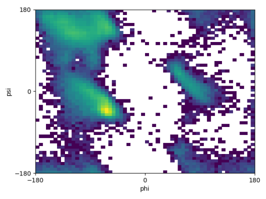
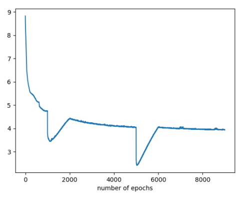
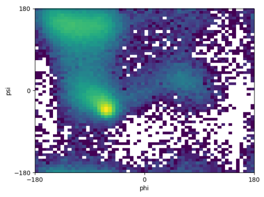


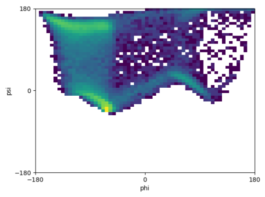
Figure 13



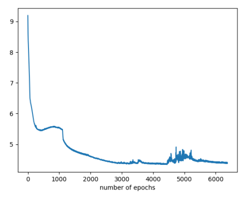
(a)



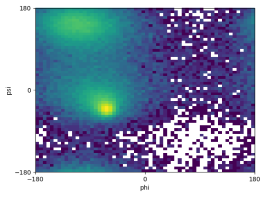
(b)



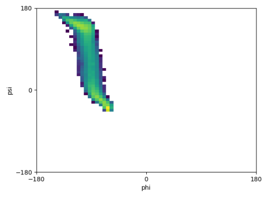
(c)



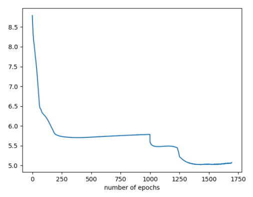
(d)



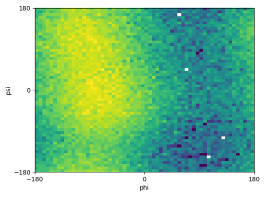
(e)



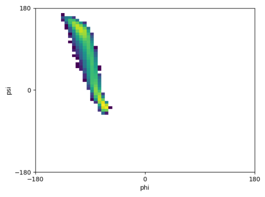
(f)



(g)



(h)



(i)

Figure 14

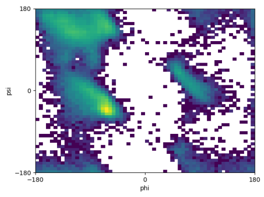
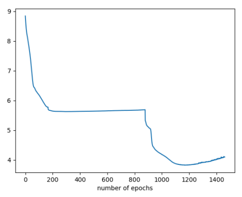
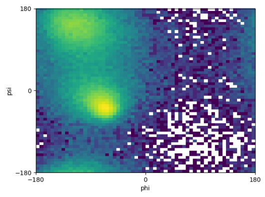


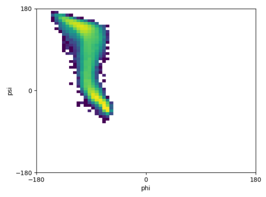
Figure 15



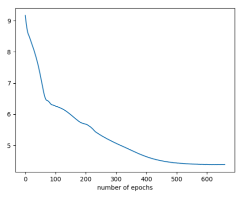
(a)



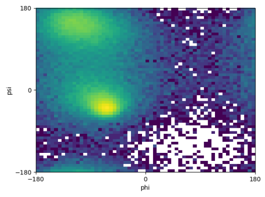
(b)



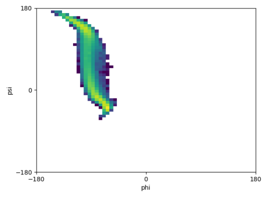
(c)



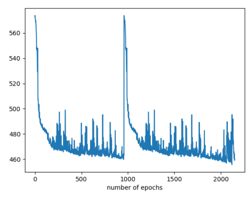
(d)



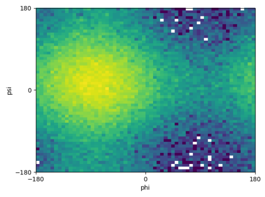
(e)



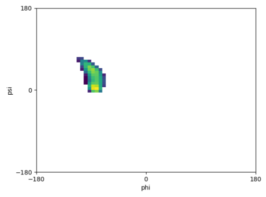
(f)



(g)



(h)



(i)

Figure 16

4 Discussion

References

[1] Gaurav Pandey, Vipin Kumar, and Michael Steinbach. Computational approaches for protein function prediction: A survey. *Twin Cities: Department of Computer Science and Engineering, University of Minnesota*, 2006.

[2] Theodore Peters. Proteins: Structure and function. *Clinical Chemistry*, 51(11):2220–2221, 2005.

[3] David Lee, Oliver Redfern, and Christine Orengo. Predicting protein function from sequence and structure. *Nature reviews molecular cell biology*, 8(12):995, 2007.

[4] David Baker and Andrej Sali. Protein structure prediction and structural genomics. *Science*, 294(5540):93–96, 2001.

[5] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. 2018.

[6] GN T Ramachandran and V Sasisekharan. Conformation of polypeptides and proteins. In *Advances in protein chemistry*, volume 23, pages 283–437. Elsevier, 1968.

[7] Alan D McNaught and Andrew Wilkinson. *IUPAC Compendium of chemical terminology.* IUPAC, S.l., 2003.

[8] David Whitford. *Proteins, structure and function.* John Wiley and Sons, Chichester, 2005.

[9] John Moult, Jan T Pedersen, Richard Judson, and Krzysztof Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 23(3):ii–iv, 1995.

[10] Jake VanderPlas. Frequentism and bayesianism: A python-driven primer. *arXiv preprint arXiv:1411.5018*, 2014.

[11] Thomas Hamelryck, Kanti Mardia, and Jesper Ferkinghoff-Borg. *Bayesian methods in structural bioinformatics.* Springer, 2012.

[12] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge university press, 1998.

[13] Benjamin Schuster-Böckler and Alex Bateman. An introduction to hidden markov models. *Current protocols in bioinformatics*, 18(1):A–3A, 2007.

[14] Pierre Baldi, Søren Brunak, and Francis Bach. *Bioinformatics: the machine learning approach.* MIT press, 2001.

[15] Simon Haykin. *Neural networks*, volume 2. Prentice hall New York, 1994.

[16] Christopher M Bishop. *Pattern recognition and machine learning.* Information science and statistics. Springer, London, 2006.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[18] DJ Best and Nicholas I Fisher. Efficient simulation of the von mises distribution. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(2):152–157, 1979.

[19] Kanti V Mardia, Gareth Hughes, Charles C Taylor, and Harshinder Singh. A multivariate von mises distribution with applications to bioinformatics. *Canadian Journal of Statistics*, 36(1):99–109, 2008.

[20] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[21] Guillaume Baudart, Martin Hirzel, and Louis Mandel. Deep probabilistic programming languages: A qualitative study. *arXiv preprint arXiv:1804.06458*, 2018.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[23] Uber Technologies, Inc. Pyro documentation: SVI part I: An introduction to stochastic variational inference in pyro. `http://pyro.ai/examples/svi_part_i.html`, 2017. [Online; accessed 14-February-2019].

[24] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[25] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[26] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.