The AlphaFold algorithm

Francisco Simoes

August 23, 2020

Abstract

AlphaFold is an AI system created by DeepMind which tries to predict the spatial structure of a protein from its chemical composition. This text is a summary of the main ideas behind AlphaFold and explains in some detail the AlphaFold algorithm.

It was written in the first weeks of my internship at UMC Utrecht Brain Center, under the supervision of Dr. Kevin Kenna. Although we ended up abandoning AlphaFold when it became clear the open sourced part of the code is not enough to actually use it for the purposes we were interested in, I leave this *informal summary* here since it may be useful for someone trying to decode what the AlphaFold documentation and papers are saying. It is not self-contained and serves mainly as a pointer to important concepts that the reader should research.

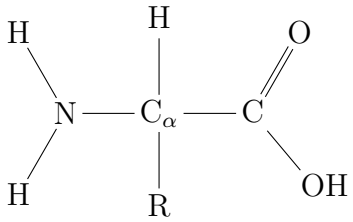
1 Protein structure

To understand what AlphaFold does, we first need to familiarize ourselves with the problem it is trying to solve: getting the structure of a protein from its composition.

With this in mind, we briefly discuss the chemical composition of aminoacids and proteins, and how one may describe the spatial structure of a protein.

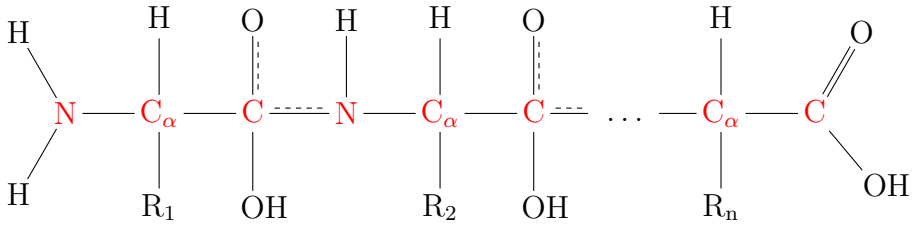
1.1 Chemical structure

Chemical structure of an aminoacid:



where R is the so-called *side-chain*, and distinguishes different aminoacids. The NH_2 and the COOH parts of the aminoacid are called the *amino group* and *carboxyl group* respectively. Note that C_α is just a normal carbon atom; we only use the subscript α to distinguish it from the carboxyl carbon, which is sometime denoted by C_β.

The bonds between the aminoacids in a protein are called *peptide bonds*.



It turns out that, in each mid-chain aminoacid, the double bond between the carboxyl C and O delocalize, making the peptide bonds also have partial double bonds (half-dotted bonds). This ends up preventing the peptide bond from rotating, reducing the degrees of freedom.

The *backbone* of the protein is the sequence of atoms drawn in red.

1.2 Physical/spatial structure

We distinguish between 4 types of structure:

Primary: Order of the aminoacids.

Secondary: Physical structure due to interactions between the backbone atoms.

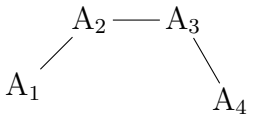
Tertiary: Physical structure due side chain interactions (*e.g.* does the protein curl up upon itself?).

Quaternary: (When > 1 polypeptide chain) arrangement of multiple peptide chains.

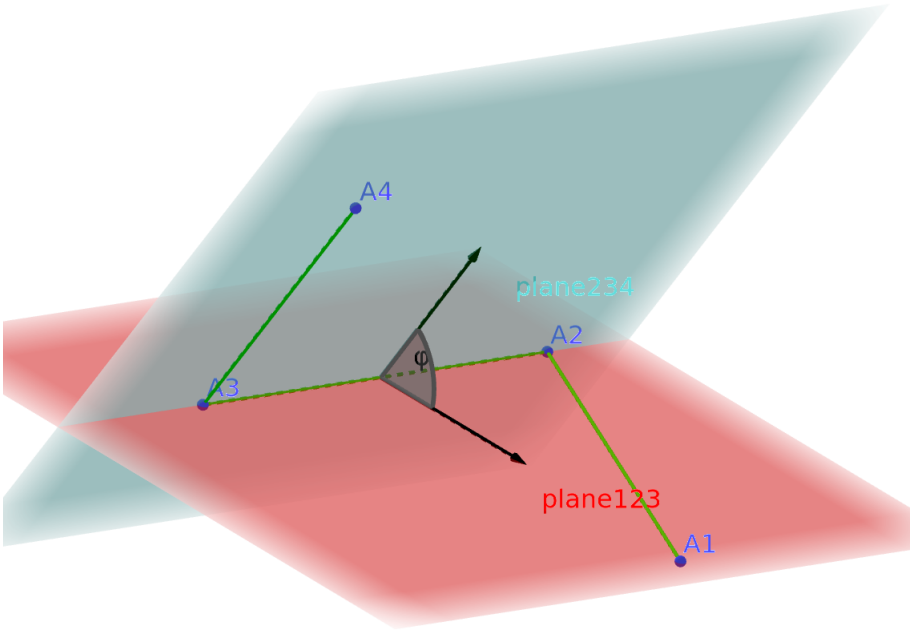
1.3 Torsion angles

We can describe the spatial structure of a protein using the so-called *torsion angles*. Let us define torsion angles:

Consider 4 atoms $A_1, ..., A_4$ connected by bonds $A_i - A_{i+1}$.



We can define their torsion angle ϕ (with respect to the middle bond $A_2 - A_3$) by taking the planes $A_1A_2A_3$ and $A_2A_3A_4$ and setting $\phi :=$ {angle between the planes} := {angle between the normal to the intersection line lying in the first plane and the normal to the intersection line lying in the second plane}.



In the case of a protein, one defines, for each residue (aminoacid) in the protein, two torsion angles ϕ, ψ, corresponding to the following backbone atoms:

$$\phi : \mathrm{N} - \mathrm{C}_\alpha - \mathrm{C} - \mathrm{N}$$

$$\psi : \mathrm{C} - \mathrm{N} - \mathrm{C}_\alpha - \mathrm{C}$$

For mid-chain aminoacids, the positions of the carboxyl carbons (the C_βs) + the torsion angles give the entire structure of the protein.

2 AlphaFold

The DeepMind team published two papers on AlphaFold. I will focus on the paper published in Nature. The two papers contain very different descriptions of the AlphaFold, and do not even seem to agree. For instance, the second paper says that AlphaFold uses fragmentation + simmulated annealing, while the Nature paper says exactly the opposite.

2.1 Distograms

Before I describe the algorithm/structure of AlphaFold, I must introduce the concept of "distogram":

Usually neural networks have numbers as their output. However one can instead obtain probability distributions. In the AlphaFold case, the main NN does not predict distances $d_{i,j}$ between residues. Instead, it predicts *one discrete probability distribution* $P(d_{ij}|inputs)$ *for each distance* d_{ij}. These are discrete because AlphaFold divides the range 2-20 $\mathring{A}$ of sensible distances in 64 bins, and predicts a probability for each one (using the softmax function after the last layer, converting its logits into probabilities).

A *distogram* is the set of all these probability distributions for a given sequence of aminoacids. Symbolically, it can be written $\{P(d_{ij}|\text{inputs})\}_{i,j}$. It can be graphically represented by a stack of 64 contact matrix style graphics with each matrix corresponding to a bin (of a distance d), and each matrix entry acquiring a color according to the probability $P(d_{ij})$.

Distograms contain much more information about the classical contact maps.

2.2 AlphaFold's main structure

1. A deep convolutional neural network predicts the distogram of a protein with primary structure (aminoacid sequence) S.

    - Inputs: S and MSA(S).

- Output: Distrogram $\{P(d_{ij})\}_{i,j}$, where i, j denote C_β atoms of the residues. The distogram is constructed using 64 bins.
- Ground truth used when training: known PDB structures.

2. Define a potential V_{distance} using the distogram. This will be the most important part of the total potential V.

   (a) This potential has two parts. The most relevant one (by far) is obtained by smoothing the negative log[1] of the probability distributions, and summing over all residues:

   $$V^{(1)}_{\text{distance}}(\mathbf{r}) = \sum_{i,j} \text{spline}[-\log(P(d_{i,j}|S, \text{MSA}(S)))]$$

   where $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_L)$, with $\mathbf{r}_i$ the coordinate vector of the carboxyl carbon atom of the residue i and L the length of the protein.

   (b) The second part is an auxiliary reference potential which again is defined using a distogram, but this distogram comes from a CNN similar to the one described above, except that its only input is the length of the protein. Hence:

   $$V^{(2)}_{\text{distance}}(\mathbf{r}) = \sum_{i,j} \text{spline}[-\log(P(d_{i,j}|\text{length}))]$$

3. Allow an output head of the original CNN to predict torsion angles $(\phi, \psi) := \{\phi_i, \psi_j\}_{i,j}$, resulting in a probability distribution $P(\phi_i, \psi_i|S, MSA(S))$ for each residue i.

4. Define a torsion potential $V_{torsion}$ using the torsion probability distributions.

   (a) We compose[2] the torsion probability distributions with the von Mises distribution (circular analogue of the normal distribution, with perion 2π and centered at 0).

[1] As usual, cost functions use log since it turns multiplications into additions, thus speeding up the computation of derivatives for the gradient descent.

[2] In the paper they say "fit" instead of "compose". I guess this is what they mean, since it makes sense (under the assumption that usually torsion angles follow a normal distribution around 0).

(b) Define the torsion potential by:

$$V_{\text{torsion}}(\phi,\psi) = -\sum_i \log[P_{\text{von Moses}}\left(P\left(\phi_i,\psi_i|S,\text{MSA}(S)\right)\right)]$$

5. A small improvement is obtained by adding a third potential $V_{\text{score2 smooth}}(\phi,\psi)$ used in another software (Rosetta). This potential includes the van der Waals potential, thus avoiding clashes between electron clouds.

6. The total potential is the sum of the above three potentials.

   (a) Notice that V_dist is not, as it stands, a function of the torsion angles, while the other two potentials are. But we can parametrize the residues' coordinates by the torsion angles: $\mathbf{r} = G(\phi,\psi)$. Note that the parametrization function G must contain the bond lengths already.

   (b) We can then use G to write $d_{ij} = \parallel \mathbf{r_i} - \mathbf{r_j} \parallel$ as a function of the torsion angles, and thus we can write $V_dist(\phi,\psi)$.

   (c) The total potential is then:

$$V(\phi,\psi) = V_{\text{dist}}(\phi,\psi) + V_{\text{torsion}}(\phi,\psi) + V_{\text{score2 smooth}}(\phi,\psi)$$

7. We can now use repeated gradient descent on V, which is smooth on the torsion angles, to find the correct torsion angles.

   (a) Initialize by sampling ϕ,ψ from $P(\phi,\psi|S,\text{MSA}(S))$.

   (b) Repeat gradient descent for different initializations, all samples as described. This results in a pool of predicted (ϕ,ψ).

   (c) Multiple gradient descents again, this time initializing using the (ϕ,ψ) from the pool of torsion angles just created, but summing a backbone torsion noise (these are called "noisy restarts"), thus ending up with a new pool of torsion angles.

   (d) Repeat the previous step a few times, and then choose the lowest potential (ϕ,ψ) from all the torsion angles in all the pools.

8. Finally, one can slightly improve the results by allowing "relaxation" of the structure.

3 Why AlphaFold cannot easily be used by us?

Long story short: we do not have access to the entire code.

Long story long: the code for the feature generation part of the pipeline is not completely available, and it is not even clear what many of the features are (see the AlphaFold GitHub repository). The gradient descent part of the pipeline is also not available. The only part of the pipeline that is open sourced is the CNN that takes the features as inputs and outputs the distograms + contacts.

In practice this means that, as of now, AlphaFold can only be used by DeepMind, unless one makes her own feature generation and gradient descent scripts. I contacted one of the authors of the AlphaFold paper and he confirmed that unfortunately this is indeed the case.