Protein Structure Prediction Using RL

Krishanu, Nirjhar, Hardik

April 22, 2024

Outline

1. Protein folding problem
   - Problem Relevance
   - The Hydrophobic-Polar Model
2. Mechanism
   - Organizational Stages of Protein Formation
   - Hydrophobic collapse
3. Conceptual Representation
4. Reinforcement Learning
   - Reward Function
   - Learning Policy
   - ϵ-Greedy Policy
   - Q-Learning Approach
5. Conclusion and Future Work
6. Bibliography

Problem Relevance

- Proteins are essential biological molecules that carry genetic messages and perform various functions in organisms, such as structural support and catalysis of metabolic reactions.

Problem Relevance

- Proteins are essential biological molecules that carry genetic messages and perform various functions in organisms, such as structural support and catalysis of metabolic reactions.
- They are made of amino acids that fold into a three-dimensional structure, known as the native state, which allows them to interact with other proteins and perform their roles.

Problem Relevance

- Proteins are essential biological molecules that carry genetic messages and perform various functions in organisms, such as structural support and catalysis of metabolic reactions.
- They are made of amino acids that fold into a three-dimensional structure, known as the native state, which allows them to interact with other proteins and perform their roles.
- The protein folding problem, which involves predicting a protein's structure from its amino acid sequence, is a major challenge in bioinformatics with significant implications for medicine and genetic engineering.

Problem Relevance

- Proteins are essential biological molecules that carry genetic messages and perform various functions in organisms, such as structural support and catalysis of metabolic reactions.
- They are made of amino acids that fold into a three-dimensional structure, known as the native state, which allows them to interact with other proteins and perform their roles.
- The protein folding problem, which involves predicting a protein's structure from its amino acid sequence, is a major challenge in bioinformatics with significant implications for medicine and genetic engineering.
- In the following we are addressing the Bi-dimensional Protein Folding Problem (BPFB), but this model can be easily extended to the three-dimensional protein folding problem.

The Hydrophobic-Polar Model

- The Hydrophobic-Polar (HP) model introduced by Dill [1] is a molecular energy-stabilizer model which categorizes amino acids in two categories as per their behaviour in aqueous solutions.

The Hydrophobic-Polar Model

- The Hydrophobic-Polar (HP) model introduced by Dill [1] is a molecular energy-stabilizer model which categorizes amino acids in two categories as per their behaviour in aqueous solutions.
- The amino acids repelled by water belong to the hydrophobic (H) or non-polar category whereas those which have an affinity for water belong to the hydrophilic or polar (P) category.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
    - The amino acid sequence of polypeptide chain forms the primary structure.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
    - The amino acid sequence of polypeptide chain forms the primary structure.
    - It contains all the information necessary for the higher order of structure.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
  - The amino acid sequence of polypeptide chain forms the primary structure.
  - It contains all the information necessary for the higher order of structure.
- Secondary

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
    - The amino acid sequence of polypeptide chain forms the primary structure.
    - It contains all the information necessary for the higher order of structure.
- Secondary
    - Formed by hydrogen bond interactions of adjacent amino acids.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
  - The amino acid sequence of polypeptide chain forms the primary structure.
  - It contains all the information necessary for the higher order of structure.
- Secondary
  - Formed by hydrogen bond interactions of adjacent amino acids.
  - Large number of such local interactions form α-helices and β-pleated sheets.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
    - The amino acid sequence of polypeptide chain forms the primary structure.
    - It contains all the information necessary for the higher order of structure.
- Secondary
    - Formed by hydrogen bond interactions of adjacent amino acids.
    - Large number of such local interactions form α-helices and β-pleated sheets.
- Tertiary

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
  - The amino acid sequence of polypeptide chain forms the primary structure.
  - It contains all the information necessary for the higher order of structure.
- Secondary
  - Formed by hydrogen bond interactions of adjacent amino acids.
  - Large number of such local interactions form α-helices and β-pleated sheets.
- Tertiary
  - More compact 3D structure.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
  - The amino acid sequence of polypeptide chain forms the primary structure.
  - It contains all the information necessary for the higher order of structure.
- Secondary
  - Formed by hydrogen bond interactions of adjacent amino acids.
  - Large number of such local interactions form α-helices and β-pleated sheets.
- Tertiary
  - More compact 3D structure.
  - Large proteins often consist of several protein domains, which are distinct structural units that fold somewhat independently from one another.

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
  - The amino acid sequence of polypeptide chain forms the primary structure.
  - It contains all the information necessary for the higher order of structure.
- Secondary
  - Formed by hydrogen bond interactions of adjacent amino acids.
  - Large number of such local interactions form α-helices and β-pleated sheets.
- Tertiary
  - More compact 3D structure.
  - Large proteins often consist of several protein domains, which are distinct structural units that fold somewhat independently from one another.
- Quaternary

Organizational Stages of Protein Formation

The building blocks of every protein consists of 20 different types of amino acids.

- Primary
    - The amino acid sequence of polypeptide chain forms the primary structure.
    - It contains all the information necessary for the higher order of structure.
- Secondary
    - Formed by hydrogen bond interactions of adjacent amino acids.
    - Large number of such local interactions form α-helices and β-pleated sheets.
- Tertiary
    - More compact 3D structure.
    - Large proteins often consist of several protein domains, which are distinct structural units that fold somewhat independently from one another.
- Quaternary
    - Found in those proteins that have two or more interacting polypeptide chain, termed subunits.

Organizational Stages of Protein Formation

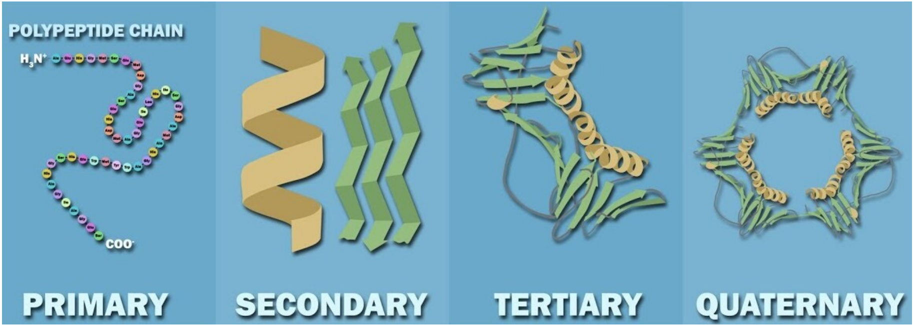

Figure 1: Organizational Stages of Protein Formation

Hydrophobic collapse

The biological foundation of the HP model is that the first-order driving force of protein folding is due to a “hydrophobic collapse”, in which those residues which prefer to be shielded from water (hydrophobic residues) are driven to the core of the protein, while those which interact more favorably with water (polar residues) remain on the outside of the protein.

Conceptual Representation

The primary structure of a protein is seen as a sequence of n amino acids and each amino acid is classified in one of the two categories: hydrophobic (H) or polar (P):

$$\mathcal{P} = p_1 p_2 \dots p_n,$$

where $p_i \in \{H, P\}, \forall\, 1 \leq i \leq n$.
A conformation of the protein $\mathcal{P}$ is a function C, that maps the protein sequence $\mathcal{P}$ to the ordered set of positions of the amino-acids on the 2D Cartesian lattice.
Formally, if

$$\mathcal{B} = \{\mathcal{P} = p_1 p_2 \dots p_n \mid p_i \in \{H, P\}, \forall\, 1 \leq i \leq n\},$$

$$\mathcal{G} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \mid x_i, y_i \in \mathbb{Z},\, 1 \leq i \leq n\},$$

then a conformation of C is defined as $C : \mathcal{B} \to \mathcal{G}$ such that

$$\mathcal{P} = p_1 p_2 \dots p_n \mapsto \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\},$$

where (x_i, y_i) represents the position on the lattice to which the amino acid p_i is mapped by C.

Conceptual Representation

- Obviously, the sequence of amino acids in the 2D space needs to form a path since amino acids are chained together, but not all conformations constitute a path.

Conceptual Representation

- Obviously, the sequence of amino acids in the 2D space needs to form a path since amino acids are chained together, but not all conformations constitute a path.
- For a conformation to be a *path*, the points on the grid need to form a chain where each two consecutive amino-acids (in the primary sequence) are placed immediately next to one another (horizontally or vertically) on the given lattice.

Conceptual Representation

- Obviously, the sequence of amino acids in the 2D space needs to form a path since amino acids are chained together, but not all conformations constitute a path.
- For a conformation to be a *path*, the points on the grid need to form a chain where each two consecutive amino-acids (in the primary sequence) are placed immediately next to one another (horizontally or vertically) on the given lattice.
- Also, the mapped positions of two different amino acids must not be superposed in the lattice, i.e., the path should be *self-avoiding*.

Conceptual Representation

Formally, the mapping C is called a *path* if:

$$\forall\, 1 \le i, j \le n,\ |i - j| = 1 \implies |x_i - x_j| + |y_i - y_j| = 1.$$

A path C is *self-avoiding* if C is injective, i.e.,

$$\forall\, 1 \le i, j \le n,\ i \ne j \implies (x_i, y_i) \ne (x_j, y_j).$$

We call a conformation *valid* if it is a *self-avoiding path*.

Conceptual Representation

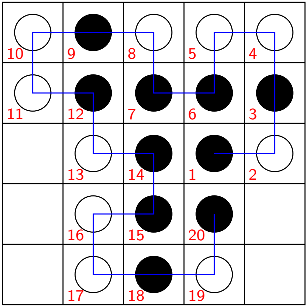

Figure 2: A protein configuration for the sequence $\mathcal{P} = HPHPPHHPHPPHPHHPPHPH$, of length 20. Black circles represent hydrophobic (H) amino acids, while white circles represent hydrophilic (P) ones.

Free Energy of a Fold

- We need to distinguish the quality of a given fold of a protein from any other fold.

Free Energy of a Fold

- We need to distinguish the quality of a given fold of a protein from any other fold.
- We assess the stability of a protein structure through the Gibbs free energy associated with the particular folding configuration.

Free Energy of a Fold

- We need to distinguish the quality of a given fold of a protein from any other fold.
- We assess the stability of a protein structure through the Gibbs free energy associated with the particular folding configuration.
- We say a contact between two residues (amino acids) is a *topological contact* if they are not covalently linked and there is an edge connecting the lattice points of the two residues.

Free Energy of a Fold

- We need to distinguish the quality of a given fold of a protein from any other fold.
- We assess the stability of a protein structure through the Gibbs free energy associated with the particular folding configuration.
- We say a contact between two residues (amino acids) is a *topological contact* if they are not covalently linked and there is an edge connecting the lattice points of the two residues.
- The energy function in the HP model reflects the fact that hydrophobic amino acids have a propensity to form a hydrophobic core.

Free Energy of a Fold

- We need to distinguish the quality of a given fold of a protein from any other fold.
- We assess the stability of a protein structure through the Gibbs free energy associated with the particular folding configuration.
- We say a contact between two residues (amino acids) is a *topological contact* if they are not covalently linked and there is an edge connecting the lattice points of the two residues.
- The energy function in the HP model reflects the fact that hydrophobic amino acids have a propensity to form a hydrophobic core.
- Consequently, the energy function adds a value of -1 for each two hydrophobic amino acids that are mapped by C on neighboring positions in the lattice, but that are not consecutive in the primary structure $\mathcal{P}$. This gives a rough estimate of the Gibbs free energy of the fold.

Free Energy of a Fold

- It is clear that any hydrophobic amino acid in a valid conformation C can have at most 2 such neighbors (except for the first and last amino acids, that can have at most 3 topological neighbors).

Free Energy of a Fold

- It is clear that any hydrophobic amino acid in a valid conformation C can have at most 2 such neighbors (except for the first and last amino acids, that can have at most 3 topological neighbors).
- If we define $I : [n] \times [n] \rightarrow \{-1, 0\}$ by

$$I(i,j) = \begin{cases} -1, & \text{if } p_i = p_j = H \text{ and } |x_i - x_j| + |y_i - y_j| = 1 \\ 0, & \text{otherwise} \end{cases}$$

  $\forall\ 1 \leq i, j \leq n$ with $|i - j| \geq 2$, then the energy function for a given conformation C is defined as follows:

$$E(C) \overset{\text{def}}{=} \sum_{1 \leq i \leq j-2 \leq n} I(i,j)$$

Free Energy of a Fold

- It is clear that any hydrophobic amino acid in a valid conformation C can have at most 2 such neighbors (except for the first and last amino acids, that can have at most 3 topological neighbors).
- If we define $I : [n] \times [n] \to \{-1, 0\}$ by

$$I(i,j) = \begin{cases} -1, & \text{if } p_i = p_j = H \text{ and } |x_i - x_j| + |y_i - y_j| = 1 \\ 0, & \text{otherwise} \end{cases}$$

  $\forall\ 1 \leq i, j \leq n$ with $|i - j| \geq 2$, then the energy function for a given conformation C is defined as follows:

$$E(C) \overset{\text{def}}{=} \sum_{1 \leq i \leq j-2 \leq n} I(i,j)$$

- Note that this is also equal to $-1 \times (\#$ topological contacts between pairs of hydrophobic residues).

Free Energy of a Fold

- It is clear that any hydrophobic amino acid in a valid conformation C can have at most 2 such neighbors (except for the first and last amino acids, that can have at most 3 topological neighbors).
- If we define $I : [n] \times [n] \to \{-1, 0\}$ by

$$I(i,j) = \begin{cases} -1, & \text{if } p_i = p_j = H \text{ and } |x_i - x_j| + |y_i - y_j| = 1 \\ 0, & \text{otherwise} \end{cases}$$

  $\forall\ 1 \leq i, j \leq n$ with $|i - j| \geq 2$, then the energy function for a given conformation C is defined as follows:

$$E(C) \overset{\text{def}}{=} \sum_{1 \leq i \leq j-2 \leq n} I(i,j)$$

- Note that this is also equal to $-1 \times (\#$ topological contacts between pairs of hydrophobic residues).
- The protein folding problem in the HP model is to find the conformation C whose energy function $E(C)$ is minimum.

Free Energy of a Fold

- In Figure 1, the value of the energy function of the configuration is -9, as there are 9 pairs in topological contact: (1,6), (1,14), (1,20), (3,6), (7,12), (7,14), (9,12), (15,18), (15,20).

Free Energy of a Fold

- In Figure 1, the value of the energy function of the configuration is -9, as there are 9 pairs in topological contact: (1,6), (1,14), (1,20), (3,6), (7,12), (7,14), (9,12), (15,18), (15,20).
- A solution for the bi-dimensional HP protein folding problem, corresponding to an n-length sequence $\mathcal{P} \in \mathcal{B}$ could be represented by an $(n-1)$-length sequence $\pi = \pi_1\pi_2 \ldots \pi_{n-1}$, $\pi_i \in \{L, U, R, D\}$, $\forall 1 \leq i \leq n-1$, where each position encodes the direction of the current amino acid relative to the previous one (L-left, R-right, U-up, D-down).

Free Energy of a Fold

- In Figure 1, the value of the energy function of the configuration is -9, as there are 9 pairs in topological contact: (1,6), (1,14), (1,20), (3,6), (7,12), (7,14), (9,12), (15,18), (15,20).
- A solution for the bi-dimensional HP protein folding problem, corresponding to an n-length sequence $\mathcal{P} \in \mathcal{B}$ could be represented by an $(n-1)$-length sequence $\pi = \pi_1\pi_2 \dots \pi_{n-1}$, $\pi_i \in \{L, U, R, D\}$, $\forall 1 \leq i \leq n-1$, where each position encodes the direction of the current amino acid relative to the previous one (L-left, R-right, U-up, D-down).
- The solution configuration corresponding to the sequence presented in Figure 1 is $\pi = RUULDLULLDRDRDLDRRU$.

Reinforcement Learning

The RL task associated to the BPFP is defined as follows:

- For a protein of amino-acids, a State is defined as a path of length p, where $p \leq n$, representing the positions on the lattice of the first elements in the sequence. The state space $\mathcal{S}$ will consist of $\frac{4^n-1}{3}$ states, i.e., $\mathcal{S} = \{s_1, s_2, \ldots s_{\frac{4^n-1}{3}}\}$, where s_1 is the *initial state* of the agent in the environment. A state $s_{i_k} \in \mathcal{S}$ $\left(i_k \in [1, \frac{4^n-1}{3}]\right)$ reached by the reached by the agent at a given moment after it has visited states $s_1, s_{i_1}, \ldots, s_{i_{k-1}}$ is a *final state* (or *terminal state*) the number of states visited by the agent in the current sequence is $n-1$, i.e. $k = n-2$. A path from the initial to a final state will represent a possible bi-dimensional structure of the protein sequence $\mathcal{P}$.

Reinforcement Learning

The RL task associated to the BPFP is defined as follows:

- The action space $\mathcal{A}$ consists of 4 actions available to the problem solving agent and corresponding to the 4 possible directions: left (L), up (U), right (R) and down (D) used to encode a solution. We can write $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, where $a_1 = L$, $a_2 = U$, $a_3 = R$ and $a_4 = D$.

Reinforcement Learning

The RL task associated to the BPFP is defined as follows:

- The action space $\mathcal{A}$ consists of 4 actions available to the problem solving agent and corresponding to the 4 possible directions: left (L), up (U), right (R) and down (D) used to encode a solution. We can write $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, where $a_1 = L$, $a_2 = U$, $a_3 = R$ and $a_4 = D$.
- The transition function $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is defined as (see Figure 3):

$$\delta(s_j, a_k) \overset{\text{def}}{=} s_{4j-3+k} \,\forall\, k \in \{1, 2, 3, 4\}, \forall\, j \in \left[1, \frac{4^{n-1} - 1}{3}\right].$$

  This means that, at a given moment, from a state $s \in \mathcal{S}$ the agent can move in 4 successor states, by executing one of the 4 possible actions. We say that a state s' that is accessible from a state s via some action, is the *neighboring state* of s.

Reinforcement Learning

The RL task associated to the BPFP is defined as follows:

- The action space $\mathcal{A}$ consists of 4 actions available to the problem solving agent and corresponding to the 4 possible directions: left (L), up (U), right (R) and down (D) used to encode a solution. We can write $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, where $a_1 = L$, $a_2 = U$, $a_3 = R$ and $a_4 = D$.
- The transition function $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is defined as (see Figure 3):

  $$\delta(s_j, a_k) \overset{\text{def}}{=} s_{4j-3+k} \, \forall \, k \in \{1, 2, 3, 4\}, \, \forall \, j \in \left[1, \frac{4^{n-1} - 1}{3}\right].$$

  This means that, at a given moment, from a state $s \in \mathcal{S}$ the agent can move in 4 successor states, by executing one of the 4 possible actions. We say that a state s' that is accessible from a state s via some action, is the *neighboring state* of s.
- The reward function will be defined later.

Reinforcement Learning

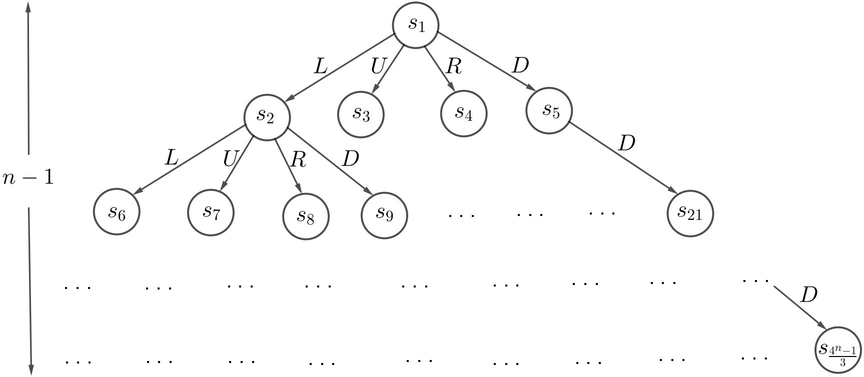

Figure 3: The States space

Reinforcement Learning

Consider a path π in the above defined environment from the initial to a final state: $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$, where $\pi_0 = s_1$ and π_{k+1} is a neighboring state of π_k $\forall$ $0 \leq k \leq n-2$. The sequence of actions obtained following the transitions between the successive states from path π will be denoted by $a_\pi = (a_{\pi_0}, a_{\pi_1}, \ldots, a_{\pi_{n-2}})$, where $\pi_{k+1} = \delta(\pi_k, a_{\pi_k})$ $\forall$ $0 \leq k \leq n-2$.

The sequence a_π will be referred as the configuration associated to the path π and it can be viewed as a possible bi-dimensional structure of the protein sequence $\mathcal{P}$. Consequently we can associate to a path π a value denoted by E_π representing the energy of the bi-dimensional configuration a_π of protein $\mathcal{P}$. (Check the definition of free energy given above).

The BPFP formulated as a RL problem will consist in training the agent to find a path π from the initial to a final state that will correspond to the bidimensional structure of protein $\mathcal{P}$ given by the corresponding configuration a_π and having the minimum associated energy.

Reward Function

Next, we define our reward function as the following:

- if the transition results in a *valid* path, then the reward received by the agent is 0.1.

Reward Function

Next, we define our reward function as the following:

- if the transition results in a *valid* path, then the reward received by the agent is 0.1.
- if the transition generates a configuration that is not *valid*, the received reward is 0.

Reward Function

Next, we define our reward function as the following:

- if the transition results in a *valid* path, then the reward received by the agent is 0.1.
- if the transition generates a configuration that is not *valid*, the received reward is 0.
- the reward received after a transition to a final state π_{n-1} after states $s_1 = \pi_0, \pi_1, \pi_2, \ldots,$ were visited is the energy of the bi-dimensional structure of protein $\mathcal{P}$ corresponding to the configuration a_π.

Reward Function

Thus, $r(\pi_k \mid s_1, \pi_1, \ldots, \pi_{k-1}) = \begin{cases} 0, \text{ if } a_\pi \text{ is not valid} \\ -E_\pi, \text{ if } k = n-1 \\ 0.1, \text{ otherwise} \end{cases}$ where
$r(\pi_k \mid s_1, \pi_1, \ldots, \pi_{k-1})$ denotes the reward received by the agent in state π_k, after it has visited states $s_1 = \pi_0, \pi_1, \pi_2, \ldots, \pi_{k-1}$.
Considering the reward function defined above, it is understood that the agent should be trained to find a self avoiding path π that minimizes the associated energy E_π.

Learning Policy

- In this work, we opted for an ϵ-greedy policy as the learning policy of our agent.

Learning Policy

- In this work, we opted for an ϵ-greedy policy as the learning policy of our agent.
- In this setting, the agent chooses the optimal action (as currently estimated from Q) with a probability $1 - \epsilon$, and a random action with probability ϵ.

Learning Policy

- In this work, we opted for an ϵ-greedy policy as the learning policy of our agent.
- In this setting, the agent chooses the optimal action (as currently estimated from Q) with a probability $1 - \epsilon$, and a random action with probability ϵ.
- This is desirable as the element of randomness encourages the exploration of the state space while the optimal choice of action ensures exploitation of what was previously learned.

Learning Policy

- Since our agent initially starts with no prior knowledge of action values, a good approach is to start with a high value of ϵ (1 for example) such that the agent mostly explores during the first episodes, and slowly decrease its value until some minimum threshold, denoted ϵ_{min}, to ensure the agent is mainly exploiting what it has learned by the end of the episode.

Learning Policy

- Since our agent initially starts with no prior knowledge of action values, a good approach is to start with a high value of ϵ (1 for example) such that the agent mostly explores during the first episodes, and slowly decrease its value until some minimum threshold, denoted ϵ_{min}, to ensure the agent is mainly exploiting what it has learned by the end of the episode.
- The non-null minimum threshold, as opposed to a 0 value for ϵ (which would mean 100% exploitation), is helpful in the sense that it prevents the agent for being permanently stuck in a local minimum.

Learning Policy

- Since our agent initially starts with no prior knowledge of action values, a good approach is to start with a high value of ϵ (1 for example) such that the agent mostly explores during the first episodes, and slowly decrease its value until some minimum threshold, denoted ϵ_{min}, to ensure the agent is mainly exploiting what it has learned by the end of the episode.
- The non-null minimum threshold, as opposed to a 0 value for ϵ (which would mean 100% exploitation), is helpful in the sense that it prevents the agent for being permanently stuck in a local minimum.
- For this reason, we also define a new variable λ called the *decay rate*, which controls the rate by which we decrease the value of after each episode. Below is the pseudo-code implementing the ϵ-greedy policy.

ϵ-Greedy Policy

- At the start of learning, set $\epsilon = 1$.

ϵ-Greedy Policy

- At the start of learning, set $\epsilon = 1$.
- During each episode, generate a number from a uniform distribution $\beta \sim U(0, 1)$.

ϵ-Greedy Policy

- At the start of learning, set $\epsilon = 1$.
- During each episode, generate a number from a uniform distribution $\beta \sim U(0, 1)$.
- If $\beta \leq \epsilon$, then pick a random action from $\mathcal{A}$. Else pick the best action from the current estimation of Q:

$$a_{\max} = \arg\max_{a'} Q(s, a').$$

ϵ-Greedy Policy

- At the start of learning, set $\epsilon = 1$.
- During each episode, generate a number from a uniform distribution $\beta \sim U(0, 1)$.
- If $\beta \leq \epsilon$, then pick a random action from $\mathcal{A}$. Else pick the best action from the current estimation of Q:

$$a_{\text{max}} = \arg\max_{a'} Q(s, a').$$

- After each episode, if $\epsilon > \epsilon_{\text{min}}$, set $\epsilon \leftarrow \epsilon \cdot \lambda$.

Q-Learning Approach

If $Q(s, a)$ denotes the value of doing the action a in state s, $r(s, a)$ denotes the reward received in state s after performing action a and s' represents the state of the environment reached by the agent after performing action a in state s, the Bellman equation for Q-learning (which represents the constraint equation that must hold at equilibrium when the Q-values are correct) is the following [2]:

$$Q(s, a) = r(s, a) + \gamma \cdot \max_{a'} Q(s', a'),$$

where γ is the discount factor for the future rewards.

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
    - Select the initial state s.

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
  - Select the initial state s.
  - Choose a from s using policy derived from Q (ϵ-greedy)

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
  - Select the initial state s.
  - Choose a from s using policy derived from Q (ϵ-greedy)
  - Loop for each step of episode:

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
  - Select the initial state s.
  - Choose a from s using policy derived from Q (ϵ-greedy)
  - Loop for each step of episode:
    - Take action a, observe r, s'.

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
    - Select the initial state s.
    - Choose a from s using policy derived from Q (ϵ-greedy)
    - Loop for each step of episode:
        - Take action a, observe r, s'.
        - Update the table entry $Q(s, a)$ as follows:

$$Q(s, a) \leftarrow r(s, a) + \gamma \cdot \max_{a'} Q(s', a').$$

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
    - Select the initial state s.
    - Choose a from s using policy derived from Q (ϵ-greedy)
    - Loop for each step of episode:
        - Take action a, observe r, s'.
        - Update the table entry $Q(s, a)$ as follows:

$$Q(s,a) \leftarrow r(s,a) + \gamma \cdot \max_{a'} Q(s', a').$$

        - $s \leftarrow s'$

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
    - Select the initial state s.
    - Choose a from s using policy derived from Q (ϵ-greedy)
    - Loop for each step of episode:
        - Take action a, observe r, s'.
        - Update the table entry $Q(s, a)$ as follows:

$$Q(s, a) \leftarrow r(s, a) + \gamma \cdot \max_{a'} Q(s', a').$$

        - $s \leftarrow s'$
    - Until s is terminal

Q-Learning Approach

The general form of the Q-learning algorithm is the following:

- Initialize $Q(s, a)$ to 0 for each pair (s, a).
- Loop for each episode:
    - Select the initial state s.
    - Choose a from s using policy derived from Q (ϵ-greedy)
    - Loop for each step of episode:
        - Take action a, observe r, s'.
        - Update the table entry $Q(s, a)$ as follows:

          $$Q(s, a) \leftarrow r(s, a) + \gamma \cdot \max_{a'} Q(s', a').$$

        - $s \leftarrow s'$
    - Until s is terminal
- Until the maximum number of episodes reached or the Q-values do not change.

Conclusion and Future Work

It has been proved by Czibula et al. [3] that the Q-values learned by the agent converge to their optimal values (i.e the values that lead to the policy corresponding to the bi-dimensional structure of protein $\mathcal{P}$ having the minimum associated energy) as long as all state-action pairs are visited an infinite number of times, thereby giving a mathematical validation of this approach.

We shall extend and implement our Q-Learning algorithm using a neural network to approximate Q values, a technique known in the literature as Deep Q-Learning.

Bibliography

[1] (1989) K. Dill and K. Lau. A lattice statistical mechanics model on the conformational sequence spaces of proteins. *Macromolecules*, 22:3986–3997.

[2] (2003) S. Russell and P. Norvig. Artificial Intelligence - A Modern Approach. *Prentice Hall International Series in Artificial Intelligence*. Prentice Hall.

[3] (2011) Czibula, Gabriela, Maria-Iuliana Bocicor, and Istvan-Gergely Czibula. A reinforcement learning model for solving the folding problem. *International Journal of Computer Technology and Applications*, 2:171-182.