Genesis.EA

Recall that the **Generator** creates a corresponding symbol table for every entry in the training database. Now, to generate the population and chromosomes — and additionally to **generalize** across proteins — we define a function:

$$F_i :< \text{Sequence} > \rightarrow < \text{Vector of fixed length P} >$$

F is essentially a **weighted sum** of all the composition functions ($f_1, f_2, \ldots, f_N$) applied on some sequence X:

$$F(X) = a \cdot f_1(X) + b \cdot f_2(X) + \cdots + z \cdot f_N(X)$$

Each **chromosome** in the population encodes one possible instantiation of this F — i.e., it holds the weights $a, b, \ldots, z$ applied to the respective functions. So a chromosome is simply:

Chromosome for F:

a	b	c	...	z

Where each weight corresponds to a learned contribution of the respective feature extractor function f_i.

Fitness Computation

Fitness is calculated by evaluating F over all sequences in the training set. For a known protein X, we compute the table $[f_1(X), f_2(X), \ldots, f_N(X)]$, apply the weights from a given chromosome to get $F(X)$, and then compare $F(X)$ against the **true 3D structure or known representation**.

This comparison can use various metrics like **RMSD**, **phi/psi angle distance**, or any domain-specific structural distance. The **closer** the output of F is to the actual known structure, the **higher** the fitness score.

This process is repeated for every protein in the training set, and the **average (or aggregate)** error determines the **fitness of the chromosome**.

Optimization and Generalization

Over successive generations (if using EA) or iterations (if using PSO), the population of chromosomes is optimized to minimize error (or maximize fitness). The weights $a, b, \ldots, z$ evolve accordingly.

Once the model achieves high enough accuracy (e.g., ≥90%) on the training set, the **best-performing chromosome** is selected. The corresponding function F then becomes a **generalized estimator**.

Inference on Novel Proteins

For a novel protein sequence S_new:

1. Compute the table $[f_1(S_{new}), f_2(S_{new}), \ldots, f_N(S_{new})]$.
2. Apply the learned F (i.e., use the evolved weights) to compute $F(S_{new})$.
3. The result is an estimated **3D representation** of S_{new}.

Since the composition functions f_i rely purely on sequence properties, and F is trained to generalize across many sequences, the model should be able to infer accurate approximations for **previously unseen proteins**.