Protein engineering

Jiří Kléma

Department of Computer Science,
Czech Technical University in Prague

IDA
Intelligent Data Analysis
RESEARCH GROUP

http://cw.felk.cvut.cz/wiki/courses/b4m36bin/start

Overview

- Protein engineering
  - success stories,
  - basic strategies
    - rational design,
    - directed evolution,
    - machine learning driven directed evolution (MLDE),
- MLDE as Bayesian optimization
  - exploration vs exploitation dilemma,
  - key components
    - the surrogate model to capture protein fitness,
    - acquisition function to choose the next protein for screening,
    - sequential kernels and protein embeddings to assess protein similarity,
- summary
  - the reached results, the overall goals.

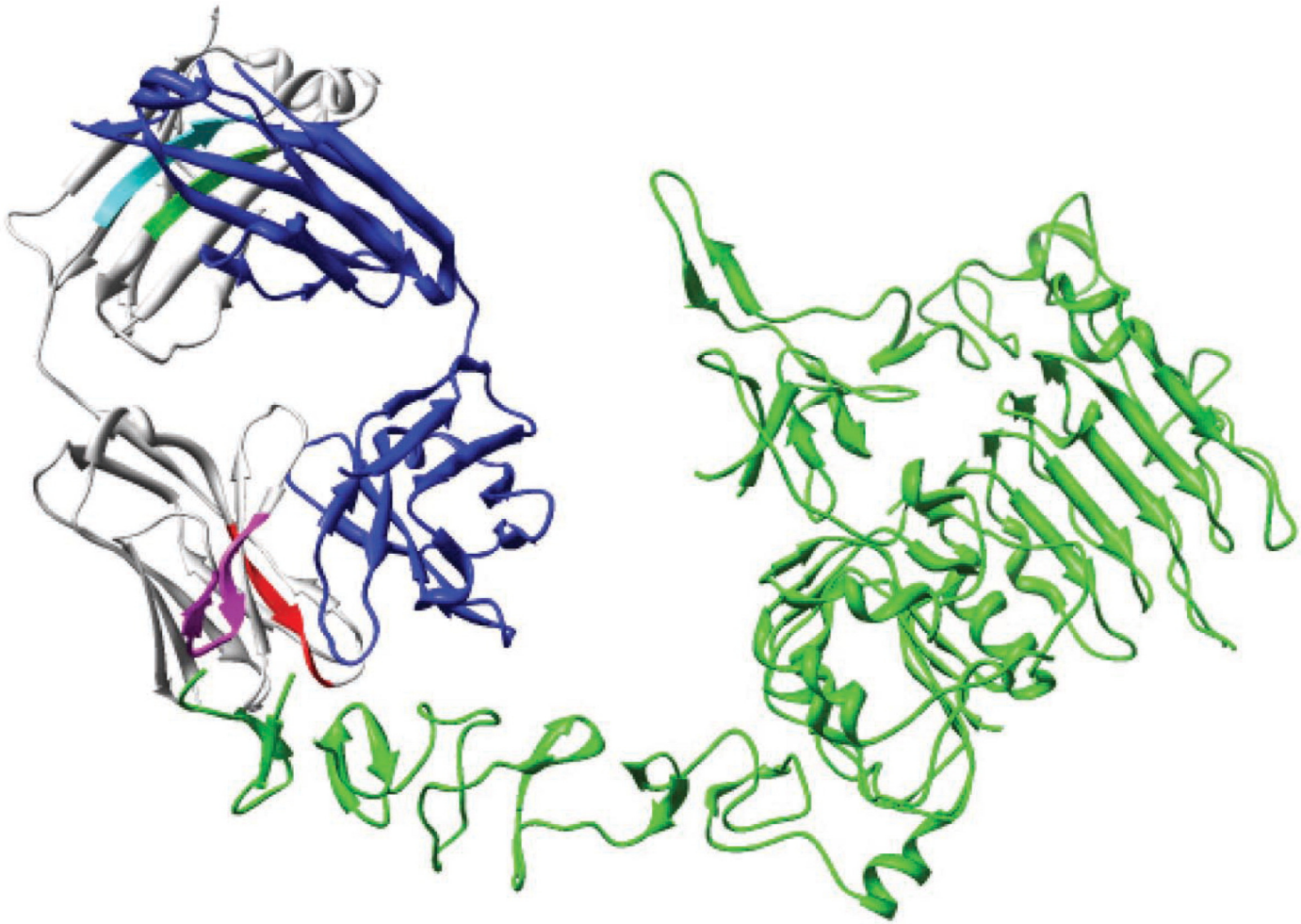
Protein engineering

- **Protein engineering**
  - serves to design, modify and create proteins in order to obtain molecules with specific properties,
  - traditionally solved through **rational design** using functional knowledge
    * identify the desired property,
    * analyze the protein structure and function,
    * predict which mutations will affect the desired property,
    * introduce changes to the protein e.g. through site-directed mutagenesis,
    * perform functional and stability testing,
    * iterate and optimize,
  - disadvantage: requires detailed structural and mechanistic insights
    * difficult to be automatized,
    * explores only a small part of sequence space,
    * difficulty handling complex proteins.

Protein engineering success stories

- Therapeutic antibodies
  - Herceptin, an antibody engineered to target the HER2 receptor, a groundbreaking treatment for breast cancer,
- industrial applications
  - cellulases, better enzymes for biofuel production, used to break down plant biomass into fermentable sugars,
- protein therapeutics
  - Humalog (insulin lispro), a synthetic insulin analog, engineered to have a faster onset and shorter duration of action compared to regular insulin,
- CRISPR/Cas systems
  - improving the specificity and efficiency of the CRISPR/Cas9 gene-editing technology.

Protein engineering – Herceptin and HER2 receptor

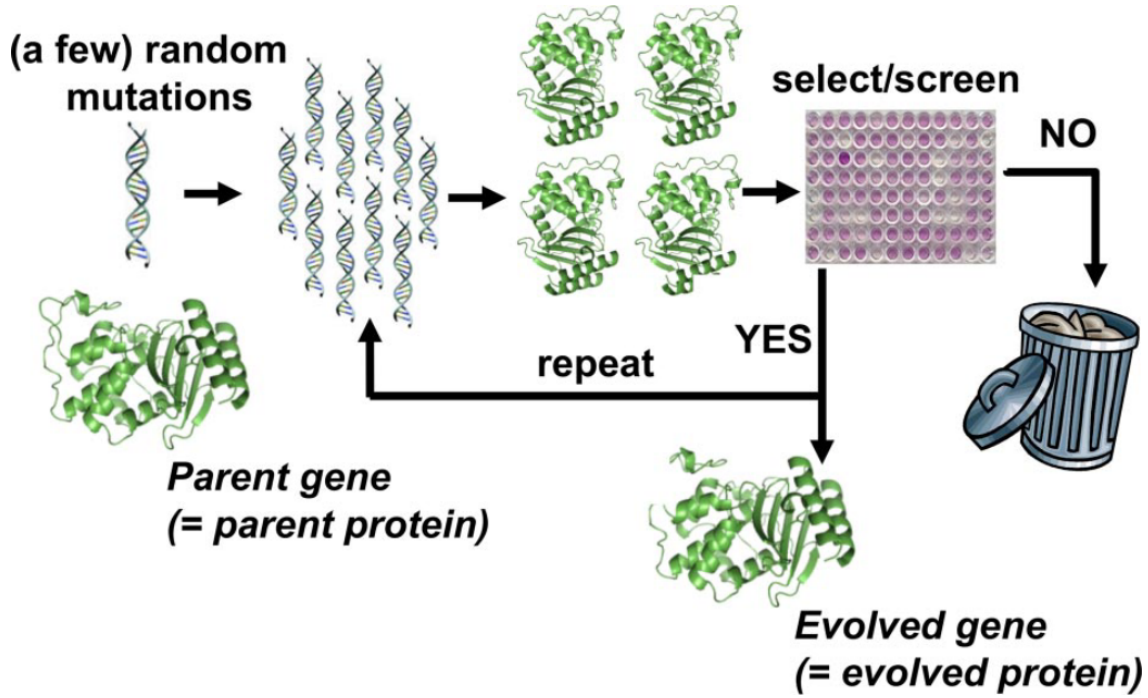


Wang et al.: Potential aggregation prone regions in biotherapeutics, 2009.

Structure of the antigen-binding fragment of Herceptin (the synthetic protein, drug) in complex with extracellular domain of HER2 (a receptor involved in breast cancer). The green chain is HER2. The light and heavy chains of Herceptin are shown in gray and blue, respectively. The predicted aggregation-prone motifs are highlighted with different colors. The magenta segment is LLIYSASFLY in CDR L2 and the red is YCQQHY in CDR L3. The two motifs SVFIFP and VVCLL in LC are in cyan and green, respectively.

Protein engineering and directed evolution

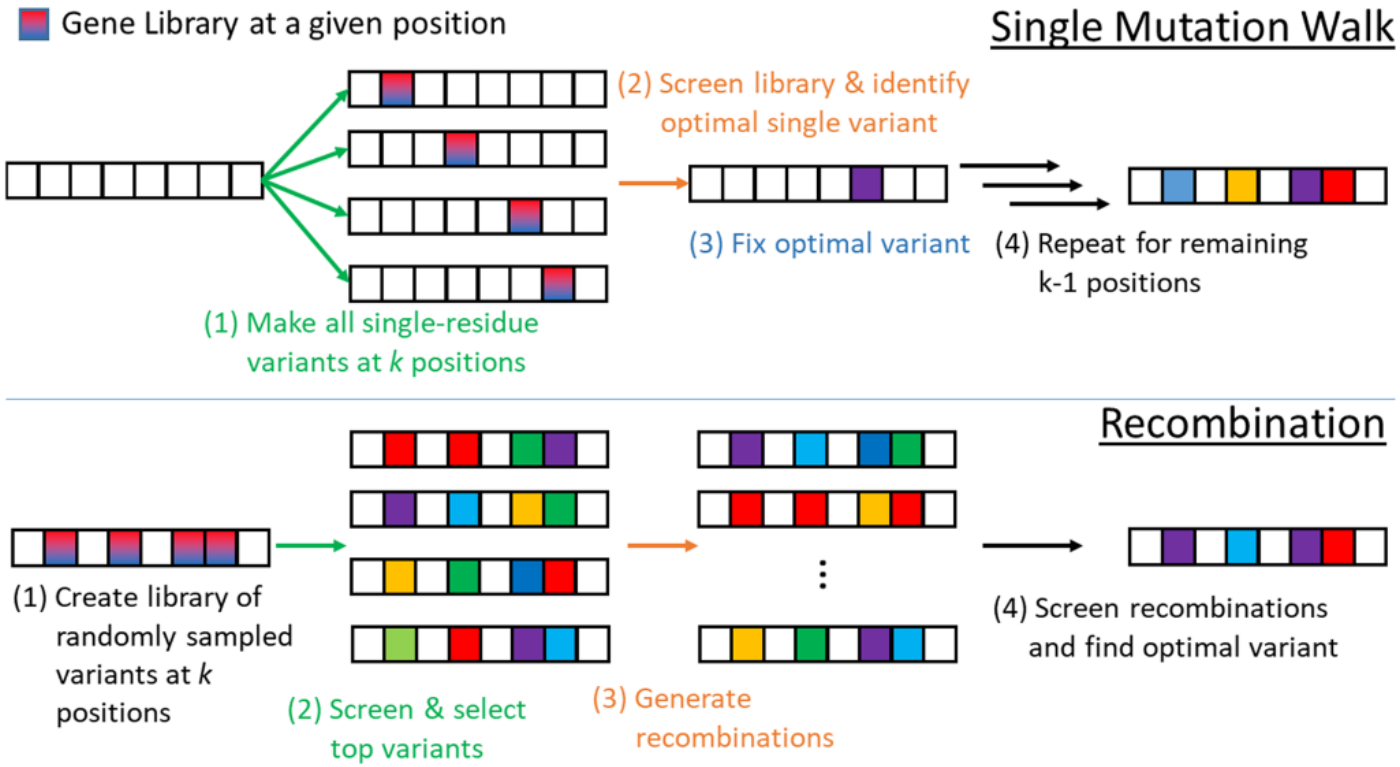
- **Directed evolution**
  - is a more recent protein engineering strategy,
  - it mimics natural evolution,
  - Frances Arnold won the 2018 Nobel Prize in Chemistry for it.



Bloom & Arnold: In the light of directed evolution: Pathways of adaptive protein evolution, 2009.

Model-free improvements in directed evolution

- Directed evolution is challenging
  - especially screening (a large number of mutations, their production and testing),
  - natural evolution can be accelerated by simple model-free approaches.



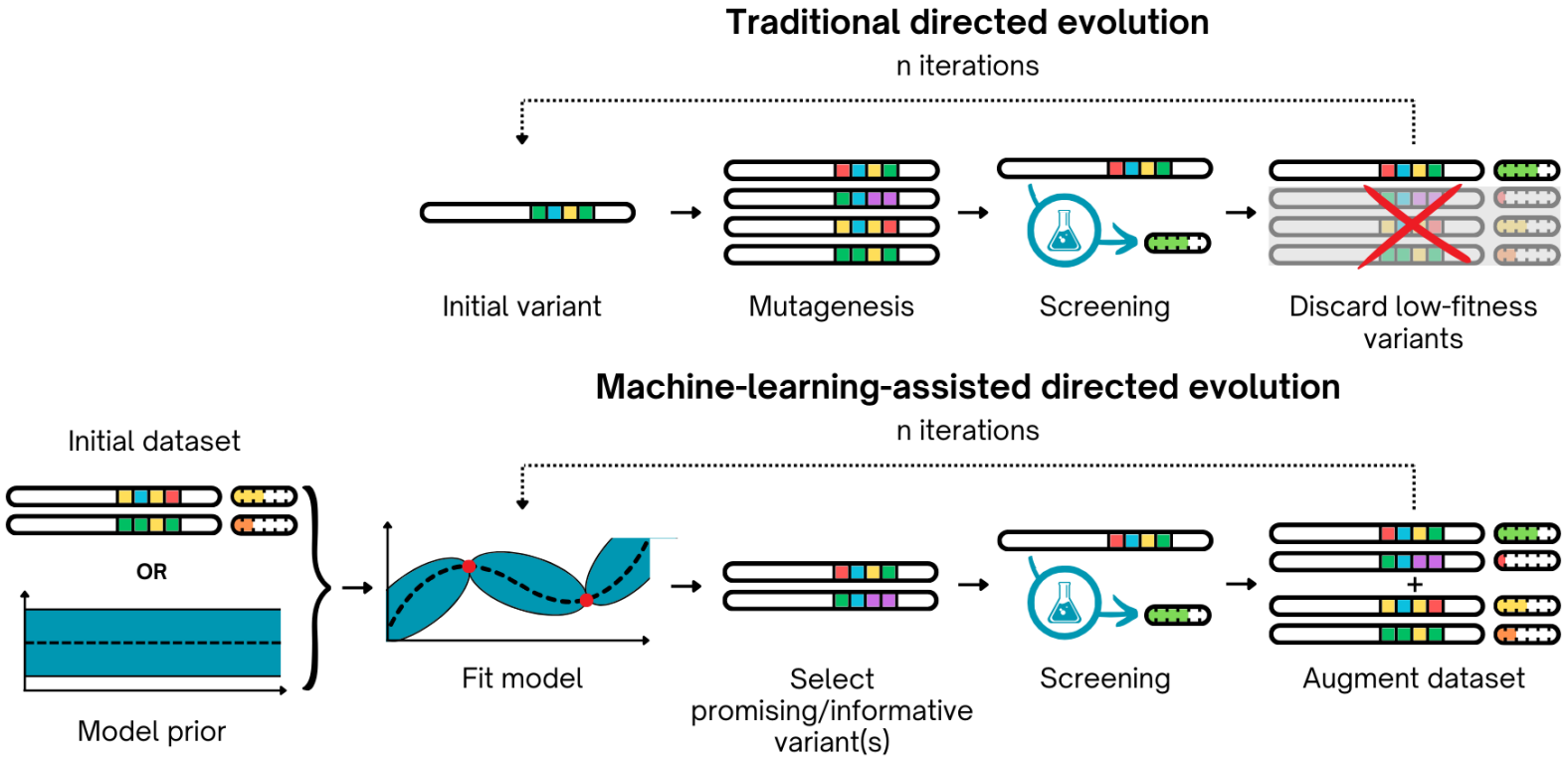
Frisby & Langmead: Bayesian optimization with evolutionary and structure-based regularization for directed protein evolution, 2021.

Model-free directed evolution

- Has got a couple of advantages
  - does not need any computational model
    - ∗ . . . and necessary training data,
  - unbiased discovery with no prior assumptions
    - ∗ . . . useful for de novo discoveries,
- however, the **model** may be used to
  - learn a mapping from protein sequence to functional properties
    - ∗ e.g., stability, activity, binding affinity,
  - increase evolution efficiency and decrease screening burden
    - ∗ the model learns patterns and reduces the number of experiments,
  - identify epistatic (= context dependent) interactions,
    - ∗ the effect of one mutation depends on the presence of another mutation,
  - avoid evolutionary traps
    - ∗ model-free methods may miss a better solution.

Machine learning in directed evolution

- **Machine learning** can identify useful protein variants even faster
  - uses a model predicting powerful mutations and their combinations.



Soldát & Kléma: Directed evolution of proteins via Bayesian optimization in embedding space, 2024.

Machine learning in directed evolution

- Active learning
  - minimizes the amount of labeled data needed,
  - while maximizing model performance,
  - focuses the learning process on most informative variants (= mutants),
- how it works
  - take a (small) initial set of screened variants,
  - build a model and use it to assign the scores to all the variants,
  - select the variants with the highest uncertainty (where the model is unsure)
    - or resolve exploration vs exploitation dilemma,
  - experimentally test the selected variants = verify their predicted scores,
  - feed the results to the model to improve it,
  - repeat from step 2 until a stopping condition is met,
  - deliver the best scoring variant.

Bayesian optimization (BO) for ML-assisted evolution

- BO is an efficient way to guide the search for high-performing protein variants
  - the problem is to optimize an expensive, unknown function $f(x)$,
  - i.e., to find $x^* = \arg\max_{x \in \mathcal{X}} f(x)$, when evaluating $f(x)$ is costly,
  - its model is probabilistic, most often Gaussian process (GP),
- GP is its key part
  - a surrogate model to predict $f(x)$ and guide the search,
  - a non-parametric model without assuming a specific function form,
  - it provides both mean $f(x)$ (prediction) and its variance (uncertainty),
  - it can incorporate domain knowledge via the kernel function,
  - it guides the acquisition function to select the most informative next point.

Gaussian process as a surrogate model

- A GP models an unknown function $f(x)$ as a distribution over functions

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

  - $m(x) = \mathbf{E}[f(x)]$ is the mean function (often assumed to be 0),
  - $k(x, x') = \mathbf{E}[(f(x) - m(x))(f(x') - m(x'))]$ is the covariance function,
  - e.g., RBF kernel: $k(x, x') = \sigma^2 \exp\left(-\frac{||x-x'||^2}{2l^2}\right)$,

- given training inputs $X = \{x_1, \ldots, x_n\}$, the prior distribution over function values is

$$f \sim \mathcal{N}(m(x), K(X, X))$$

  - $K(X, X)$ is the covariance matrix computed using the kernel function,
  - the prior represents our belief about the function before observing any data,
  - kernel controls the smoothness and variability of functions sampled from the GP prior.

Gaussian process as a surrogate model

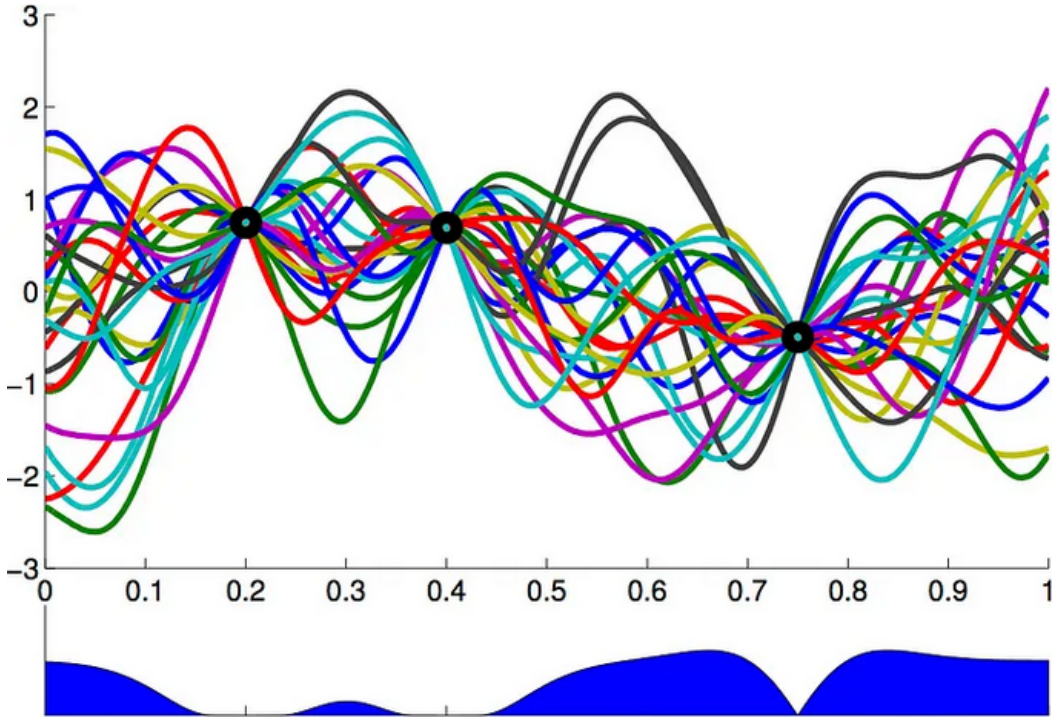
- The posterior distribution over function values gives predictions at new points,
- given observed data $D_n = \{(x_i, f(x_i))\}_{i=1}^n$, the GP posterior at a new point x_{n+1} is:

$$\mu(x_{n+1}) = k_{n+1}^T (K + \sigma_n^2 I)^{-1} y$$
$$\sigma^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k_{n+1}^T (K + \sigma_n^2 I)^{-1} k_{n+1}$$

  - K is the kernel matrix of observed points,
  - k_{n+1} is the kernel vector between x_{n+1} and observed points,
  - $y = (f(x_1), \ldots, f(x_n))$ is the vector of observed values,
  - σ_n^2 is the variance of the observation noise,
  - $\mu(x_{n+1})$ is the predicted mean,
  - $\sigma^2(x_{n+1})$ is the predicted variance.

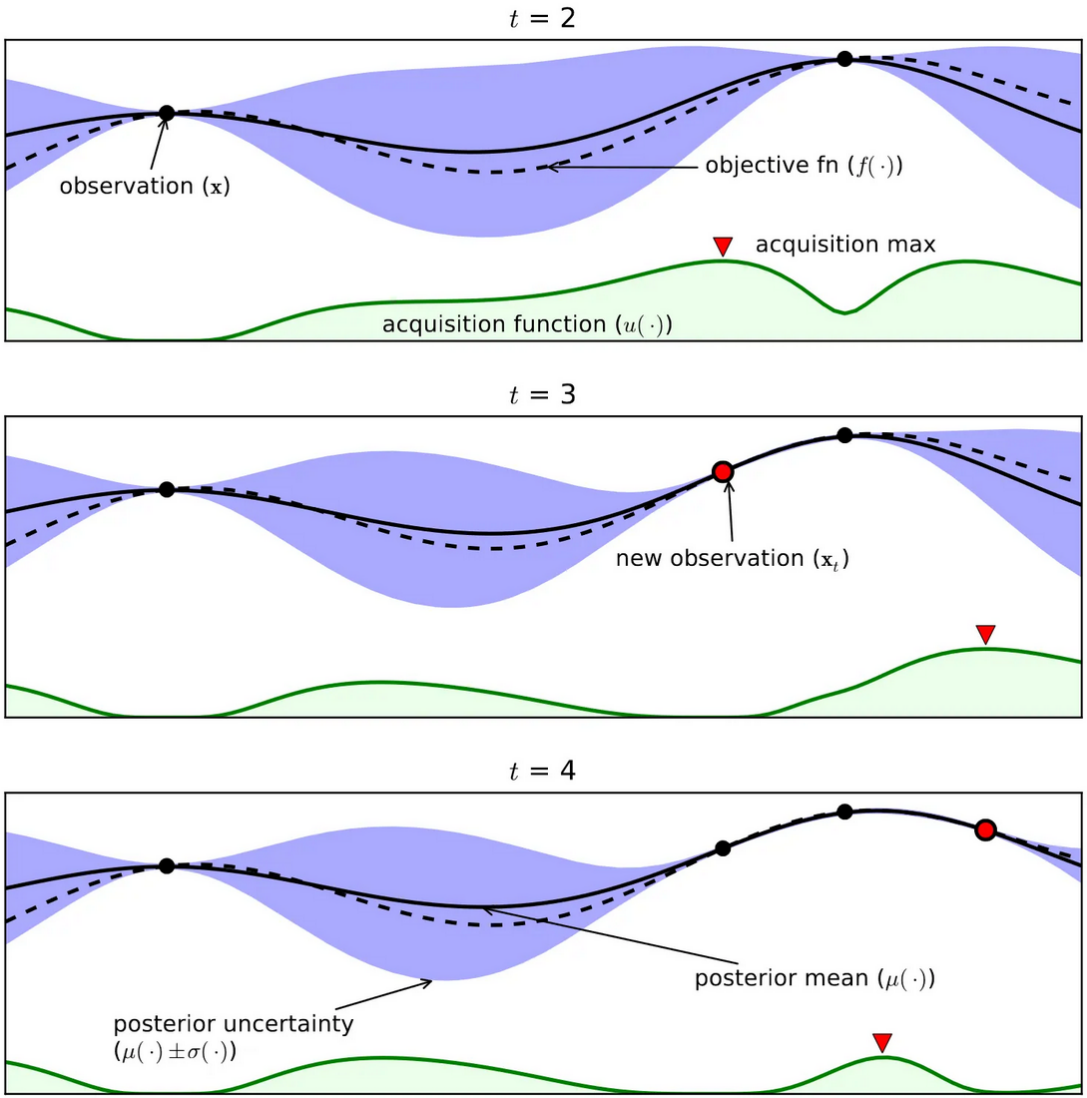
Gaussian process as a surrogate model

- An illustration of GP with three training observations available
  - each curve is one sample from a multivariate normal distribution,
  - the kernel function determines the curve shapes,
  - the uncertainty grows with the distance from training observations.



Chandradevan: Shallow understanding on Bayesian optimization, Towards Data Science, 2017.

Bayesian optimization and Gaussian processes



Chandradevan: Shallow understanding on Bayesian optimization, Towards Data Science, 2017.

Bayesian optimization loop

- Bayesian optimization loop
  1. fit a GP to observed data D_n,
  2. compute the acquisition function over the search space (two options shown)

$$UCB(x) = \mu(x) + \lambda\sigma(x)$$

  - $\mu(x)$ and $\sigma(x)$ represent the mean prediction and its variance,
  - λ is an exploration parameter,

$$EI(x) = (\mu(x) - f_{max} - \xi)\Phi\left(\frac{\mu(x) - f_{max} - \xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - f_{max} - \xi}{\sigma(x)}\right)$$

  - f_{max} is the best observed function value,
  - ξ is an exploration parameter,
  - $\Phi()$ and $\phi()$ are the CDF and PDF of the standard normal distribution,
  3. select the next point x_{n+1} that maximizes the acquisition function,
  4. evaluate $f(x_{n+1})$, $n = n + 1$ and repeat from step 1.

Kernel options for protein sequences

- BO typically represents function inputs as vectors in a continuous space
  - $f : \mathbf{R}^d \to \mathbf{R}$ and $k : \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}$,
- however, proteins p are sequences, formally
  - $p \in \mathcal{A}^L$, where $\mathcal{A}$ stands for amino acids and L for sequence length,
  - the kernels such as the RBF kernel shown before cannot be applied directly,
- we can close this gap in a couple of ways
  - introduce dedicated **sequence-based kernels**
    - $k_s : \mathcal{A}^L \times \mathcal{A}^L \to \mathbf{R}$,
  - **embed protein sequences** into a numerical vector space
    - $e : \mathcal{A}^L \to \mathbf{R}^d$,
    - ... and use the original kernels $k : e(\mathcal{A}^L) \times e(\mathcal{A}^L) \to \mathbf{R}$.

Sequence-based kernels

- Sequence-based kernels calculate protein similarity directly
  - edit distance (Levenshtein kernel)
    - measures the minimum number of insertions, deletions, or substitutions to transform one sequence into another,
    - pros: simple and interpretable,
    - cons: may not capture functional relationships well,
  - spectrum kernel
    - converts sequences into k-mer frequency vectors $\phi_s(p)$
    - uses a dot-product kernel

$$k_s(p, p') = \sum_{s \in \mathcal{S}_k} \phi_s(p)\phi_s(p')$$

    - $\phi_s(p)$ is the count of k-mer s in protein p,
    - S_k is the set of all possible k-mers (substrings of length k),
    - pros: works well for evolutionary-related sequences,
    - cons: ignores order and structural context.

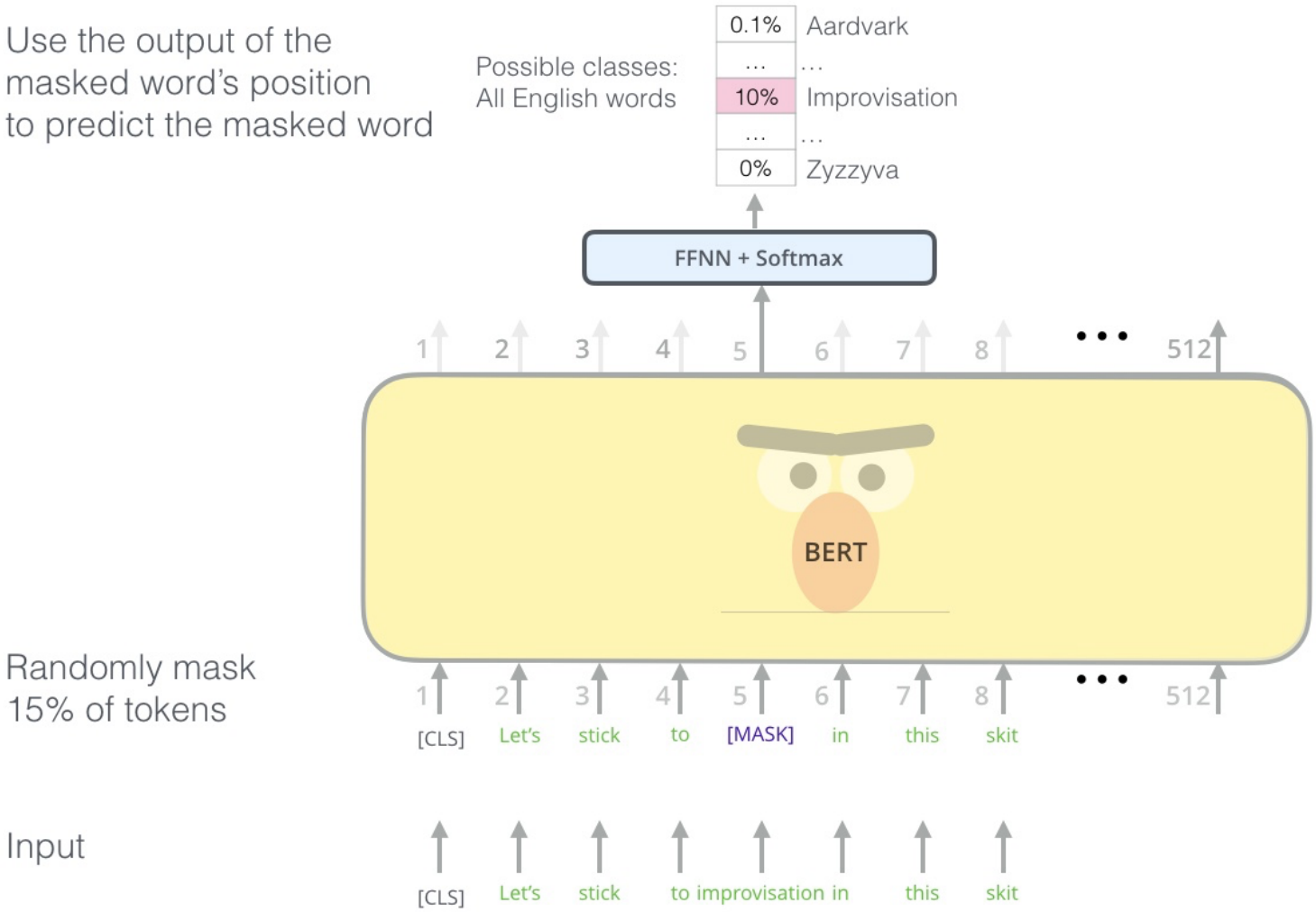
Embedding-based kernels

- Embedding-based representations transform proteins into numerical vectors
  - **one-hot encoding** is one of the most simple options
    - each amino acid mapped to a unique unit vector,
    - $\mathcal{A} = \{A, C, \ldots, Y\} \rightarrow A = (1, 0, 0, \ldots, 0)$, $C = (0, 1, 0, \ldots, 0)$, ...
    - a protein of length L is a vector of length $|\mathcal{A}|^L$ (or matrix $|\mathcal{A}| \times L$),
    - pros: simple and interpretable,
    - cons: a sparse representation, does not learn any relationships,
  - **protein large language model embeddings**
    - use pretrained protein language models such as ESM-2 or ProGen,
    - calculate embeddings and compute kernel values in the embedding space,
    - pros: captures functional and evolutionary similarity among proteins,
    - cons: requires a pretrained model.

ESM-2 pretraining to obtain protein embeddings

- ESM-2 (Evolutionary Scale Modeling) is a protein BERT trained using masking
  1. take a large database of raw protein sequences (no annotations needed),
  2. initialize a transformer model with multi-head self-attention,
  3. tokenize the sequences (most often 1 token = 1 amino acid (AA)),
  4. randomly pick a sequence,
  5. randomly mask about 15% amino acids in the sequence,
     - original: MKTLLLTLW,
     - masked input: MK[MASK]LLLT[MASK]W,
  6. model predicts the masked amino acids, error used to modify the model
     - most often the cross-entropy loss that is backpropagated through model,
  7. repeat from step 4 until the model converges,
  8. extract embeddings that capture protein properties
     - the final layer of the model represents each AA as a dense vector,
     - the protein embedding can be achieved as their average embedding.

General BERT pretraining by masking



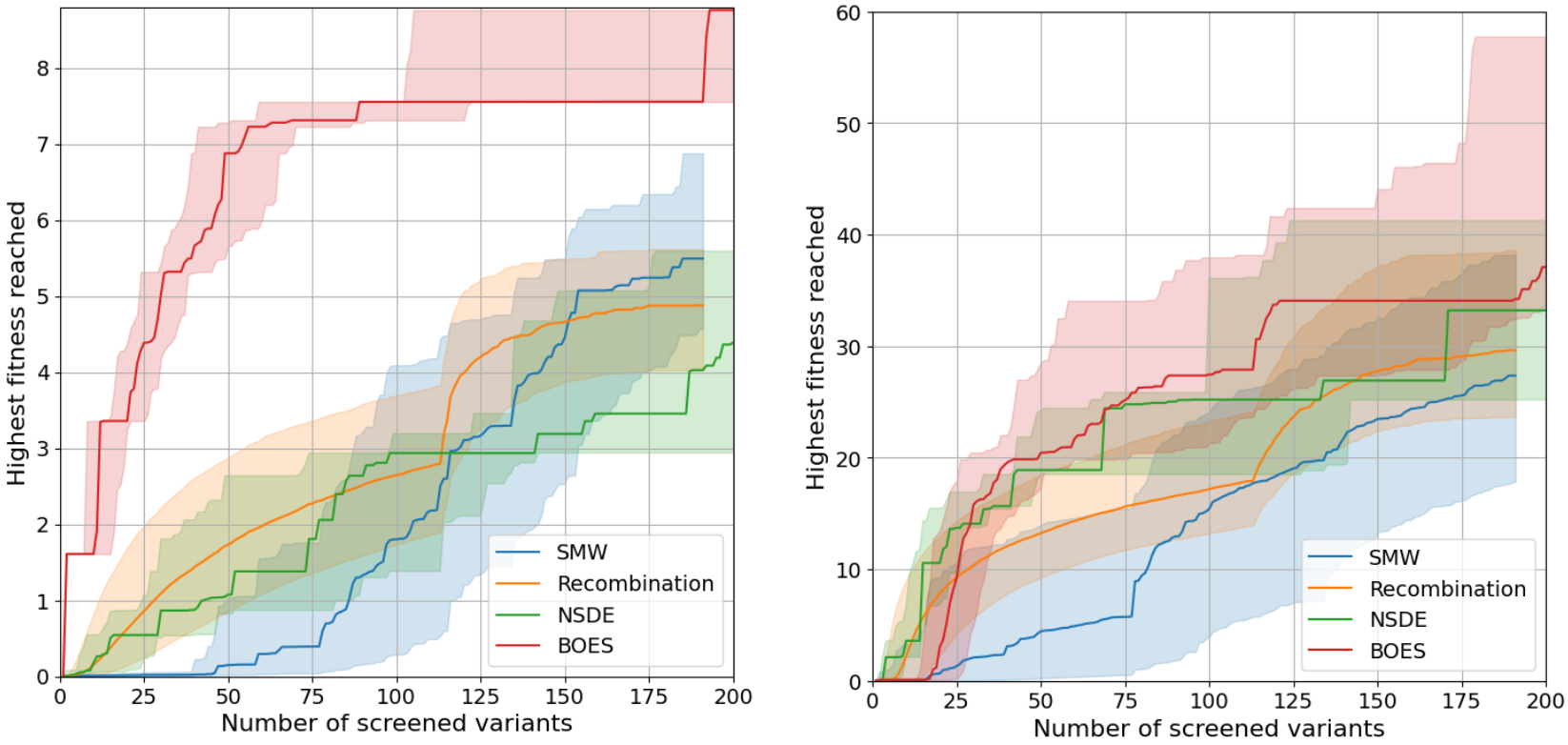
https://jalammar.github.io/illustrated-bert/

Machine learning in directed evolution

- Bayesian optimization in embedded protein space
  - procedure proposed within the diploma thesis of the LEB student in the IDA group
    * use **large protein language model**
      · maps protein sequences to number vectors,
      · facilitates the subsequent learning of fitness = protein quality,
    * over these number vectors learn **Gaussian process**
      · predicts the fitness of proteins, simultaneously indicating the uncertainty in the predictions,
    * we start with a small number of randomly selected proteins
      · initial undirected screening is limited (1 protein),
    * **Bayesian optimization** selects additional useful mutations to screen
      · screening reveals the fitness value and we can refine our model.

Bayesian optimization in the embedded protein space

- The proposed procedure surpasses the approaches known so far
  - both classical DE procedures without the use of machine learning,
  - and an identical optimization procedure without the protein language model.



Results reached in two tasks: GB1 (left), PhoQ (right).
Soldát & Kléma: Directed evolution of proteins via Bayesian optimization in embedding space, 2024.

Summary

- Protein engineering is a very popular way of synthetic protein creation,
- especially, ML-based directed evolution becomes more and more popular
  - bayesian optimization in embedded protein space is a good example,
  - however, there are also many other approaches including
    * reinforcement learning, transfer learning, generative models,
- the role of ML-based methods will even grow with
  - availability of representative protein training datasets,
  - ability to deal with imbalanced/sparse/skewed data
    * well-performing proteins are rather rare,
  - incorporating protein dynamics in ML pipelines,
  - explainable (powerful deep) ML models,
  - availability of ML pipelines to clinicians/biologists.