1 Model-Based Offline Optimization

Model-Based Offline Optimization (MBO) is a powerful approach to find the "best" design (projektu), represented as a parameter vector $\mathbf{w}$ that maximizes (or minimizes) a costly scalar objective function $f(\mathbf{w})$, using solely (wyłącznie) a fixed (ustalony), pre-collected (wcześniej wybrany) dataset. The function $f(\mathbf{w})$ is considered a "black-box" because we only observe its inputs and outputs without access to its internal workings, analytical form, or derivatives. Unlike online methods that iteratively query (iteracyjnie zapytują) f, offline MBO prohibits (zabraniają) additional evaluations during optimization, relying entirely (opiera się w pełni) on existing data to propose improved designs. This makes it uniquely suited (szczególnie dopasowany) for real-world problems where further evaluations are expensive, risky, or impossible, such as drug discovery (odkrywanie leków) or materials science.

1.1 Definition

Given a pre-collected (wcześniej zebrany) dataset of parameter-score pairs $D = \{(\mathbf{w}_i, f(\mathbf{w}_i))\}_{i=1}^N$, identify (wyznaczyć) a new design (set of parameters) $\mathbf{w}^*$ such that $f(\mathbf{w}^*)$ is as large as possible (for maximization) or as small as possible (for minimization), without ever evaluating f on new inputs during the optimization process. This constraint (ograniczenie) distinguishes (odróżnia) offline optimization from traditional approaches that iteratively query the objective function.

Key Components

1. **Designs:** $\{\mathbf{w}_i\}_{i=1}^N$, where each $\mathbf{w}_i$ is a parameter vector (e.g., a molecule configuration (konfiguracja cząstki), neural network architecture, or material property set (zestaw właściwości materiału)).

2. **Scores:** $\{f(\mathbf{w}_i)\}_{i=1}^N$, where $f(\mathbf{w}_i)$ is the observed, expensive to compute output of the black-box function f.

3. **No New Queries:** Optimization must proceed using only D, with no opportunity (bez możliwości) to evaluate f at new points.

Core Idea

1. Train a surrogate model (e.g. neural net) $\hat{f}(\mathbf{w})$ to approximate $f(\mathbf{w})$ based on D.

2. Optimize $\hat{f}(\mathbf{w})$ to propose (aby zaproponować) $\mathbf{w}^* = \arg\min_{\mathbf{w}} \hat{f}(\mathbf{w})$ using gradient-based methods. Since $\hat{f}(\mathbf{w})$ is a neural network, we can leverage automatic (wykorzystać automatycznie) differentiation frameworks (e.g., PyTorch) to obtain (uzyskać) gradients and apply optimization algorithms like Adam. The challenge lies in ensuring $\mathbf{w}^*$ performs well under

(sprawdza się względem) the true f, despite limited data and no further feedback (dalszych informacji zwrotnych).

1.2 The Offline MBO Process

The workflow of offline MBO is straightforward:

1. **Start with a Fixed Dataset:**
   - Use $D = \{(\mathbf{w}_i, f(\mathbf{w}_i))\}_{i=1}^N$, collected prior to optimization (e.g., from past experiments or simulations).
   - This dataset is the **only** source of information about f.
2. **Train a Surrogate Model:**
   - Build $\hat{f}(\mathbf{w})$ (a neural network) to predict $f(\mathbf{w})$ for any $\mathbf{w}$.
   - Ensure $\hat{f}$ is accurate and computationally cheap to evaluate.
3. **Optimize the Surrogate:**
   - Apply an optimization algorithm to find $\mathbf{w}^* = \arg\min_{\mathbf{w}} \hat{f}(\mathbf{w})$. Since $\hat{f}(\mathbf{w})$ is a neural network, we can leverage automatic differentiation frameworks (e.g., PyTorch) to obtain gradients and apply optimization algorithms like Adam.
   - Leverage $\hat{f}$'s low cost for extensive searches (szerokich poszukiwań).
4. **Propose the Design:**
   - Output $\mathbf{w}^*$ as the recommended design, with its true score $f(\mathbf{w}^*)$ unknown unless post-hoc (po fakcie, późniejsza) evaluation is feasible (wykonywalny).

Key Constraint: No additional evaluations of f are allowed during training or optimization, distinguishing (odróżniający) offline MBO from iterative methods like Bayesian optimization.

2 Motivation and Real-World Relevance (znaczenie)

Why Offline MBO Matters:

- **No Additional Queries:** In domains like drug design, robotics hardware, or materials science, evaluating f (e.g., synthesizing a compound (synteza związku chemicznego), building a prototype) is costly or risky. Offline MBO uses existing data, sometimes years of prior (lat wcześniejszych) experiments, to propose new designs without further expense (ponownych kosztów).

- **Leveraging Existing Data:** Organizations often have vast (obszerne) databases of past results. Offline MBO turns this static data into actionable insights (użyteczne spostrzeżenia), recommending designs (użyteczne projekty) that improve on what's already known.

- **Broad Applicability:(szeroka zastosowalność)** From optimizing neural network hyperparameters to designing novel proteins (nowych białek), offline MBO tackles (rozwiązuje) problems where live experimentation is impractical.

3 Challenges

- **Limited View of Design Space: (przestrzeni projektowej)** A small or unrepresentative $\mathcal{D}$ restricts (ogranicza) $\hat{f}$'s ability to model f accurately (precyzyjnie) across all (w odniesieniu do) $\mathbf{w}$.

- **Out-of-Distribution (OOD) Issues:** (problem z danymi spoza rozkładu) $\hat{f}$ may overestimate (zawyżać) scores for designs unlike those in $\mathcal{D}$, leading the optimizer to propose suboptimal or invalid $\mathbf{w}^*$.

- **Surrogate Accuracy:** Simple regression (e.g., minimizing mean squared error) can falter (może zawodzić) in OOD regions, risking (co niesie ryzykow) poor generalization.

4 Recommended Reading

1. Tan, Rong-Xi, et al. *Offline Model-Based Optimization by Learning to Rank.*

2. Trabucco, Brandon, et al. *Conservative objective models for effective offline model-based optimization.*

3. Momeni, Ali, et al. *Locality-aware Surrogates for Gradient-based Black-box Optimization.*

4. Trabucco, Brandon, et al. *Design-bench: Benchmarks for data-driven offline model-based optimization.*

5. Karpathy, Andrej *A Recipe for Training Neural Networks*