Group Meeting - November 6, 2020

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Confucius (Kung Fu Tzu)

1. Wheresoever you go, go with all your heart.
2. Choose a job you love, and you will never have to work a day in your life.

Papers

1. **Autofocused oracles for model-based design** (NeurIPS 2020) `https://arxiv.org/abs/2006.08052`
2. **Self-supervised Learning on Graphs: Deep Insights and New Directions** `https://arxiv.org/abs/2006.10141`

Paper 1

Autofocused oracles for model-based design (NeurIPS 2020)
Clara Fannjiang and Jennifer Listgarten
https://arxiv.org/abs/2006.08052

Design problems (1)

Design problems

Design problems can be cast as seeking points in the design space, $\boldsymbol{x} \in \mathcal{X}$, that with high probability satisfy desired conditions on a property random variable, $y \in \mathbb{R}$. Solve:

$$\arg\max_{\boldsymbol{x}} P(y \in S|\boldsymbol{x})$$

where S is a constrant set.

Design problems (1)

Design problems

Design problems can be cast as seeking points in the design space, $\boldsymbol{x} \in \mathcal{X}$, that with high probability satisfy desired conditions on a property random variable, $y \in \mathbb{R}$. Solve:

$$\arg\max_{\boldsymbol{x}} P(y \in S|\boldsymbol{x})$$

where S is a constrant set.

Model-based optimization (MBO)

MBO seeks the parameters θ of a **search model** $p_\theta(\boldsymbol{x})$ that maximizes an objective that bounds the original objective:

$$\max_{\boldsymbol{x}} P(y \in S|\boldsymbol{x}) \geq \max_{\theta\in\Theta} \mathbb{E}_{p_\theta(\boldsymbol{x})}[P(y \in S|\boldsymbol{x})] = \max_{\theta\in\Theta} \mathbb{E}_{p_\theta(\boldsymbol{x})}\left[\int_S p(y|\boldsymbol{x})dy\right]$$

Design problems (2)

Oracle-based model-based design (MBD)

- Oracle-based MBD replaces costly and time-consuming queries of the ground truth $p(y|\mathbf{x})$, with calls to a trained regression model (i.e., **oracle**) $p_\beta(y|\mathbf{x})$ with parameters $\beta \in B$.
- Given access to a fixed dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the oracle is typically trained **once** using standard techniques and thereafter considered **fixed**.
- Optimize the lower bound:

$$\max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\mathbf{x})}\left[\int_S p_\beta(y|\mathbf{x})dy\right]$$

Design problems (3)

MBO problems are often tackled with an **Estimation of Distribution Algorithm** (EDA):

- Belongs to a class of iterative optimization algorithms
- Monte Carlo Expectation-Maximization

Given an oracle $p_\beta(y|\boldsymbol{x})$ and an inital search model $p_{\theta^{(t=0)}}$:

1. **E-step:**
   - Sample from the current search model $\bar{\boldsymbol{x}}_i \sim p_{\theta^{(t-1)}}(\boldsymbol{x})$ for all $i \in \{1, .., m\}$.
   - Compute a weight for each sample $v_i = V(P_\beta(y \in S|\bar{\boldsymbol{x}}_i))$ where $V(.)$ is a method-specific, monotonic transformation.
2. **M-step:** Perform weighted MLE to yield an updated search model $p_{\theta^{(t)}}(\boldsymbol{x})$ which tends to have more mass where $P_\beta(y \in S|\boldsymbol{x})$ is high.

Model-based design as a game (1)

Problem

Substituting the oracle $p_\beta(y|\mathbf{x})$ for the ground-truth $p(y|\mathbf{x})$ has a problem: **the oracle is only likely to be reliable over the distribution from which its training data were drawn**.

Solution

- An algorithmic strategy for iteratively updating the oracle within any MBO algorithm.
- Reformulate the MBD problem as a **non-zero-sum game**.

Model-based design as a game (2)

The oracle-based optimization

$$\arg\max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\boldsymbol{x})}[P_\beta(y \in S|\boldsymbol{x})]$$

has the solution to be **sub-optimal** with respect to the original objective that uses the ground-truth $P(y \in S|\boldsymbol{x})$. But the ground-truth is not accessible. We introduce the **oracle gap**:

$$\mathbb{E}_{p_\theta(\boldsymbol{x})}[|P(y \in S|\boldsymbol{x}) - P_\beta(y \in S|\boldsymbol{x})|]$$

Model-based design as a game (3)

A non-zero-sum game with the coupled objectives of two players:

$$\arg\max_{\theta \in \Theta} \mathbb{E}_{p_\theta(\boldsymbol{x})}[P_\beta(y \in S|\boldsymbol{x})]$$

$$\arg\min_{\beta \in B} \text{ORACLEGAP}(\theta, \beta) = \arg\min_{\beta \in B} \mathbb{E}_{p_\theta(\boldsymbol{x})}[|P(y \in S|\boldsymbol{x}) - P_\beta(y \in S|\boldsymbol{x})|]$$

$\rightarrow$ Search for a **Nash** equilibrium: a pair of values (θ^*, β^*) such that **neither** can improve its objective given the other.
$\rightarrow$ An alternating ascent-descent algorithm.

Model-based design as a game (4)

An alternating ascent-descent algorithm:

1. **The Ascent step:** Fixing the oracle parameters and updating the search model parameters to increase the objective.
2. **The Descent step:** Fixing the search model parameters and updating the oracle parameters to decrease the objective.

Note

Isn't it just a variant of EM? I think it doesn't need a whole bunch of game theory (Nash equilibrium) machinery here.

Model-based design as a game (5)

Some bound

For any search model $p_\theta(\boldsymbol{x})$, if the oracle parameters β satisfy:

$$\mathbb{E}_{p_\theta(\boldsymbol{x})}[\mathcal{D}_{\mathrm{KL}}(p(y|\boldsymbol{x})||p_\beta(y|\boldsymbol{x}))] = \int_{\mathcal{X}} \mathcal{D}_{\mathrm{KL}}(p(y|\boldsymbol{x})||p_\beta(y|\boldsymbol{x}))p_\theta(\boldsymbol{x})d\boldsymbol{x} \leq \epsilon$$

then the following bound holds:

$$\mathbb{E}_{p_\theta(\boldsymbol{x})}[|P(y \in S|\boldsymbol{x}) - P_\beta(y \in S|\boldsymbol{x})|] \leq \sqrt{\frac{\epsilon}{2}}$$

Auto-focusing

Generally, we don't have access to the ground-truth $p(y|\boldsymbol{x})$, we do have labeled training data, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, whose labels come from the ground - truth distribution $y_i \sim p(y|\boldsymbol{x} = \boldsymbol{x}_i))$. Practical oracle parameter update:

$$\beta^{(t)} = \arg\max_{\beta \in B} \frac{1}{n} \sum_{i=1}^{n} \frac{p_{\theta^{(t)}}(\boldsymbol{x}_i)}{p_0(\boldsymbol{x}_i)} \log p_\beta(y_i|\boldsymbol{x}_i)$$

Auto-focusing strategy: The oracle is retrained on re-weighted training data according to the importance weights:

$$w_i = p_\theta(\boldsymbol{x}_i)/p_0(\boldsymbol{x}_i)$$

Experiment & Our research

Discussion

I find the experiments in this paper **not** related to our current work. But the open question is how to apply it into our work of graph/molecule generation?

Some ideas

We want to generate molecules with certain properties:

- In our case, the **oracle model** $p_\beta(y|\boldsymbol{x})$ is a molecular-properties-predicting model, $\boldsymbol{x}$ is the molecular graph, y is the property, and β is the learnable parameters of a GNN.
- The **search model** $p_\theta(\boldsymbol{x})$ **cannot** be directly applied to VAE. But I think of a way around as follows. Let $p_\theta(\boldsymbol{z})$ be the learnable/adaptive prior of a VAE (or GAN), where $\boldsymbol{z}$ is the latent. From $\boldsymbol{z}$, we reconstruct the molecule $\boldsymbol{x}$ by the decoder.
- The game theory machinery can be applied similarly.

Self-supervised Learning on Graphs: Deep Insights and New Directions

Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, Jiliang Tang

https://arxiv.org/abs/2006.10141

Proposals

Proposals

Self-supervised learning (SSL): creates domain specific pretext tasks on unlabled data – SelfTask.

Note

- In my opinion, the authors haven't applied the famous **label propagation** (neighborhood aggregation) that propagates labels from labeled nodes into un-labeled nodes.
- However, that label propagation carries **uncertainty** $\rightarrow$ I have an idea of a **probabilistic model** addressing this uncertainty.

Local structure information (1)

- **Node Property:** The aim is to predict the property for each node in the graph such as vertex degree, local node importance, and local clustering coefficient.

$$\mathcal{L}_{\text{self}}(\theta, \boldsymbol{A}, \boldsymbol{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{D}_U|} \sum_{v_i \in \mathcal{D}_U} (f_\theta(\mathcal{G})_{v_i} - d_i)^2$$

where $\mathcal{D}_U$ represents the set of unlabeled nodes.

Local structure information (2)

- **Edge Mask:** Randomly mask some edges and then the model is asked to reconstruct the masked edges:

$$\mathcal{L}_{\text{self}}(\theta, \boldsymbol{A}, \boldsymbol{X}, \mathcal{D}_U) = \frac{1}{|\mathcal{M}_e|} \sum_{(v_i, v_j) \in \mathcal{M}_e} \ell(f_w(|f_\theta(\mathcal{G})_{v_i} - f_\theta(\mathcal{G})_{v_j}|), 1)$$

$$+ \frac{1}{|\bar{\mathcal{M}}_e|} \sum_{(v_i, v_j) \in \bar{\mathcal{M}}_e} \ell(f_w(|f_\theta(\mathcal{G})_{v_i} - f_\theta(\mathcal{G})_{v_j}|), 0)$$

where $\mathcal{M}_e$ is the edge set while $\bar{\mathcal{M}}_e$ is the set of non-edges.

Global structure information

- **Pairwise Distance:** Predict the shortest path pairwise distance. Discretize into 4 categories: $p_{ij} = 1$, $p_{ij} = 2$, $p_{ij} = 3$, and $p_{ij} \geq 4$.
- **Distance to clusters**: First, partitioning the graph to get k clusters. For each cluster, assign the node with the highest degree to the center of the corresponding cluster. Create a cluster distance $\boldsymbol{d}_i \in \mathbb{R}^k$ for node v_i where the j-th element is the distance from v_i to the center of C_j.

Label propagation

Note

- The authors missed the **label propagation** (message passing) that propagates the labels from labeld nodes into un-labeled nodes.
- However, the propagated labels carry **uncertainty**. We need to address this too.

Ideas:

- Suppose we have discrete labels of N types. Initialize each node with a one-hot vector of size N indicating its labels. Un-labeled nodes just have a zero vector.
- Run message passing iterations.
- For each un-labeled nodes, we normalize its vector to sum up to 1. This vector **might** indicate the probability the node belongs to each type $\rightarrow$ addresses **uncertainty**.
- Train the pretext model to predict each node's distribution. We can use KL or Jensen-Shannon distance.