Analyzer B.O. Engine

Objective function (Christos)

1. **Maximizing performance over cost:**

$$J_1(\mathbf{x}) = \arg\max_{\mathbf{x}} \frac{Performance(\mathbf{x})}{Cost(\mathbf{x})}$$

2. **Minizing cost:**

$$J_2(\mathbf{x}) = \arg\min_{\mathbf{x}} Cost(\mathbf{x})$$

$$s.t. Performance(\mathbf{x}) \text{ satisfies } SLO$$

3. **Maxizing performance:**

$$J_3(\mathbf{x}) = \arg\max_{\mathbf{x}} Performance(\mathbf{x})$$

$$s.t. Cost(\mathbf{x}) < something$$

, where performance could be either $QosValue_{throughput}$ or $QosValue_{latency}$

TODO:

1. Implement the objective functions. Note: Use log version of objective function (8) with epsilon term. At some point we need to model the epsilon term as a variable.

Instance type encoder/decoder (Xiaoyun)

TODO:

1. Encoder function that maps instance type(string) to numpy 1d array

```
encode(metricdb.nodetype) -> feature_vector
```

2. Decoder function that maps numpy 1d array back to instance type

```
decode(feature_vector) -> metricdb.nodetype
```

Note: might need to quantize the feature_vector

Acquisition Function (Che-Yuan)

$$EI_{constraint} = P(Performance(\mathbf{x}) \text{ statisfies } SLO) * EI$$

TODO:

1. Implement constraind expected improvement acquisition function:

   Compute $P(Performance(\mathbf{x}) \text{ statisfies } SLO)$

Starting points

```
do
    randomly pick of instance_type
while (distance_function(x1,x2) < threshold)
```

TODO:

1. Naïve distance function (same family or not)

Termination condition

1. difference of improvement 10% && and max run N = 6 TODO: add that into our workflow

Kernel

- Matern5/2