Deep Mining:
Copula-Based Hyperparameter Optimization for Machine Learning Pipelines

Algorithmic details

1 GCP

Let X be the matrix of inputs (each row is a unique hyperparameter) and Y the outputs (performance estimations).

1. Normalize the data :
   $\tilde{X} := [\tilde{X}_1|...|\tilde{X}_m]$ where $\tilde{X}_i = \frac{1}{\sigma_i}(X_i - m_i[1...1]^T)$ and m_i, σ_i are respectively the mean and standard deviation of $\{X_{j,i}|j = 1,..,N\}$. $\tilde{Y} := \frac{1}{\sigma}(Y - m[1...1]^T)$.

2. Compute clusters' attributes :
   Perform K-Means on $Z := [\tilde{X}_1|...|\tilde{X}_m|\tilde{Y}]$ to compute the k clusters $C_1, ..., C_k$. For each cluster C_l, let $d_{est,l}$ be the Kernel Density Estimation of $\tilde{Y}_{C_l}$, the submatrix of $\tilde{Y}$ made from the rows j such that the rows Z_j are in cluster C_l.

3. Compute the mapping function :
   Define the mapping function $\Psi(x, y) = \sum_{l=1}^{k} \alpha_l(x) * \Psi_l(y)$ where $\alpha_l(x)$ is a coefficient that decreases with the distance between x and the cluster C_l; $\Psi_l(y) = \Phi^{-1}(\int_{-\infty}^{y} d_{est,l}(u).du)$ and Φ is the cumulative density function of the standard univariate Gaussian.

4. Transform the original data :
   Consider now the vector $W := [w_1, .., w_N]^T$ where $w_j = \Psi(\bar{x}_j, y_j)$ and $\bar{x}_j, y_j$ are respectively the jth row f $\tilde{X}$ and $\tilde{Y}$.

5. Fit a Gaussian Process:
   Fit a GP on the data ($\tilde{X}$,W) via Maximum Likelihood Estimation. Choose either the *squared exponential* or *exponential periodic* covariance function.

6. Predict :
   To predict the value y at a given point $x = [x_1, .., x_m]$, first normalize x, $\tilde{x} := [\tilde{x}_1, .., \tilde{x}_m]$ where $\tilde{x}_i = \frac{x_i - m_i}{\sigma_i}$. Then use the fitted GP to predict the values of $\mu(\tilde{x})$ and $\sigma(\tilde{x})$ representing the mean and standard deviation of the Gaussian posterior at point $\tilde{x}$. Finally numerically inverse $\Psi(\tilde{x}, .)$ and compute the expectation $\tilde{y}$ as:

$$\tilde{y} = \int u.\Psi'(u).\phi_{\mu_*,\sigma_*} \circ \Psi(u).du \tag{1}$$

7. Return $y := m + \sigma * \tilde{y}$.

2 Handling cross-validation results and multiple evaluations

The algorithm above needs to be slightly updated when we consider the hyperparameter optimization process. Indeed when dealing with cross-validation results, we have a vector of observations $\bar{y}$ instead of a real one y. Moreover, it is possible that the Smart Sampling process select more than once a given hyperparameter, and in that case we would have more than one CV results. So actually Y is a list for which each entry may have a different length. The first solution is to simply replace $\bar{y}$ by its mean and use the algorithm presented above. However we have proposed some methods to leverage such information, and here is the description of how it modifies the GCP algorithm:

- Use all CV results to build the mapping function:
  The aim is to compute the KDE estimation of the density function by taking all pairs (hyperparameter,one of the CV results) into account. To do that, we duplicate the rows in $\tilde{X}$ as many times as we have observation for the corresponding hyperparameter (a hyperparameter is a row of X) to obtain the matrix X', and consider the vector Y' of all the CV results that appear in Y. We then use X' and Y' as $\tilde{X}$ and Y in point *ii.* in the algorithm above.

- Model the noise through the EGN method :
  We compute the values of $\sigma_1, .., \sigma_N$ as the standard deviation of each list entry in Y, *ie.* the standard deviation of all CV results for a given hyperparameter. Then we modify accordingly the covariance matrix when fitting the GP, as :

$$K_{t,n} = K_t + \frac{1}{\sigma_f^2} \begin{pmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_N^2 \end{pmatrix}$$

- Fit the GP with all CV results :
  In that case we completely replace $\tilde{X}$ and Y by X' and Y'. We only compute the means and standard deviations in step *i.* with X and Y_μ where Y_μ is the mean of each list entry in Y. Note that this method is not compatible with the EGN.

3 Smart Sampling

Process overview :

1. Random initialization :
   Sample uniformly inside the bounded hyperparameter space N_{init} hyperparameters and compute a performance evaluation for each of them.

2. Perform N_{smart} *smart* iterations :
   Model the performance function with an (L)GCP given all the hyperparameters already tested and their performance evaluations. Use the fitted (L)GCP to compute the value of the acquisition at $n_{param-sampling}$ random candidates uniformly sampled in the hyperparameter space. Select the candidate that maximizes the acquisition function and compute its performance evalution.

3. End with N_{final} *smart* iterations for which the acquisition function is simply the predicted performance.