uOttawa

FACULTY OF SCIENCE - 2020/2019

DEPARTMENT OF MATHEMATICS AND STATISTICS

Statistical machine learning Analysis :

Chemistry Application

Aaron Smith

Made by:
Soufiane Fadel

Abstract

This report provides an overview of some machine learning techniques that have recently appeared in the computational chemistry literature. We demonstrate that machine learning can be used to predict the performance of a synthetic reaction in multidimensional chemical space using data obtained via high-throughput experimentation. The natural fit between machine learning and chemistry research leads to the common utilization of learning algorithms to construct a smart strategy to drive Chemical Reactions to desired outcomes with a minimum cost. We created scripts to explore the Chemical space of a some defined reaction and we make it available in this GitHub repository [9].

Contents

1 Introduction

Synthetic chemists often have the following basic problem: they want to synthesize some new chemicals and have an enormous list of experiments that might achieve this goal, but they don't have the time or money to run all the experiments in the list. Since they can't run all of their experiments, they are forced to do the next-best thing: run a sequence of experiments one-at-a-time, with the hope that the information from the first few experiments in the sequence will give them the information they need (typically, they want to know which experiments have a large yield).

Naturally, we'd like to use a sequence that is as short as possible. It would be nice to have an algorithm that can predict reaction yield and determine which is the best (Reagents/Ligand/Base/Substrate/Catalyst) react towards maximized product yields. An algorithm that told us what sequence of experiments we should run to do this, but such an algorithm doesn't exist; the goal of this thesis is to design one.

This problem remains largely irreproducible/non-deterministic task in terms of efficiency. So to assist future discovery chemistry we need such a machine learning framework to facilitate the identification of optimal reaction conditions.

Nowadays, many machine learning implementations rely on providing large datasets joined to complicated algorithms and difficult to interpret benchmark statistics since the problem of data limitation is the most challenging one, in fact, if you feed a model poorly, then it will only give you poor results. Also, given the need for expert chemistry knowledge to adequately tackle non-routine tasks, these obstacles have kept the optimization of chemical reaction conditions a challenge with no well defined statistics-automated solution available and only a few applications reported [11],[7],[13],[12],[18],[19].

1.1 Process overview

We are going to try a batch of reaction conditions, see what happens, suggest another batch, see what happens, and keep on going until either we get a good result or decide the experiment is not promising. The (Fig 1) illustrate the process of our experiments and provides readers with general plan and key elements of our work.

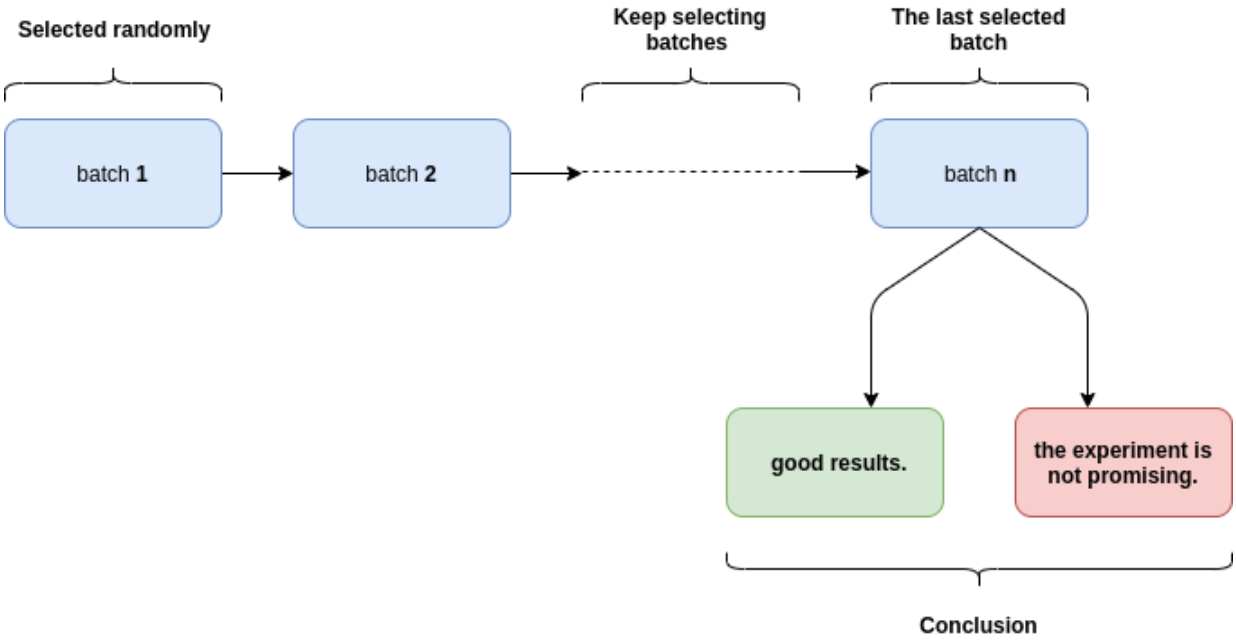


Figure 1: The process overview of our experiments.

1.2 Chemists main interest

Through our collaboration with chemists we notice that their main interest remains on the following points:

- Since the optimal results of a reaction is always not guaranteed. The chemists' goal is to ensure that the yield reaches a target goal without plateauing at an undesired level. So they want to know how to determine the ceiling of the reaction yield? How many iterations do we need to run to get conclusive results?

- Designing an efficient strategy to select the next experiments. The chemists want to design an algorithm that gives many experiments that are closely related and would give similar results. We want to maximize the diversity of the subsequent experiments and don't want to run redundant experiments. What is the proper number of experiments to run in every iteration? the goal is to run the fewest number of experiments due to cost and time but obtain good results.

- Selecting a minimum number of experiments to make a first-generation training set (batch **1**/initial data). This will reasonably be related to the number of our features. Our goal is to have a training set that is comprehensive but remains workable without consuming too much time or materials. We want to have the initial training set to

be diverse. So we may figure out How to ensure that our training set is as diverse as possible? We want to screen as much chemical space[1] as possible.

- Determining how much/fast the model is learning is very important for the chemists, it helps them to measure the confidence of the output. So how the data should be fed to model to determine if the model is learning?

In this paper we will try to answer all these questions, we have come up with solutions based on machine leaning algorithms and Bayesian optimization framework. The goal of our Bayesian optimization approach is to ensure that the experiments we select are diverse. As mentioned earlier, the strategy should consider conditions that are as distant from the training set as possible. Out framework allows the chemist to optimize the components of the reaction using only the minimal cost possible. Moreover, the algorithm doesn't need specific computational or chemical hardware.

2 Datasets description

2.1 Suzuki-Miyaura reaction

The used dataset in our analysis is about a famous organic reaction called "Suzuki-Miyaura cross-coupling". The goal is to model the reaction yield based on the components of the reaction [Reagents/Ligand/base/solvant] (see [Damith Perera al., 2018][5]).

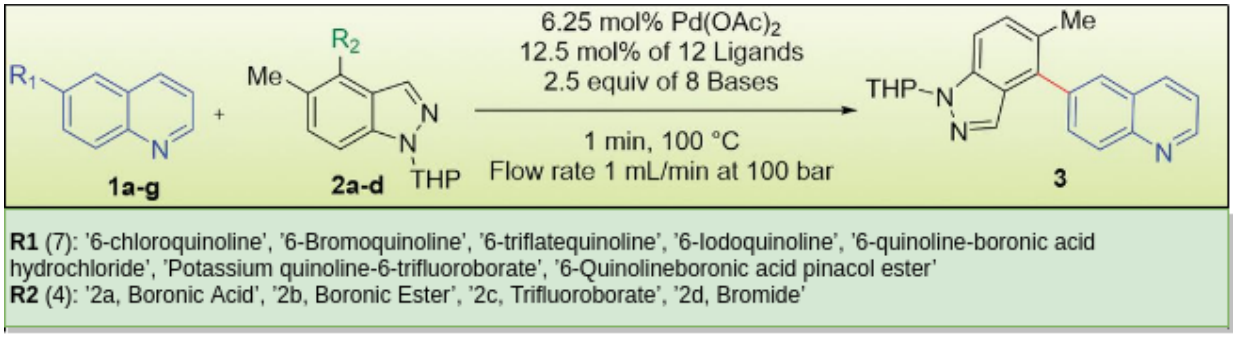


Figure 2: Suzuki-Miyaura cross-coupling reaction [5]

2.1.1 Dataset information

The dataset we have is mainly suitable for regression analysis, since the yield is a continuous variable that took values between 0 and 1 (Fig 3). We are not sure if all input variables are relevant. So it could be interesting to find new features that encode our datasets (see encoding with descriptors 2.2.2 subsection).

[1]Chemical space is a concept in cheminformatics referring to the property space spanned by all possible molecules and chemical compounds adhering to a given set of construction principles and boundary conditions.

Reactant_1_Name	Reactant_2_Name	Ligand_Short_Hand	Base_Short_Hand	Solvent_1_Short_Hand	yield
6-chloroquinoline	2a, Boronic Acid	P(tBu)3	NaOH	MeCN	0.047641092184596195
6-chloroquinoline	2a, Boronic Acid	P(Ph)3	NaOH	MeCN	0.0412096203597114
6-chloroquinoline	2a, Boronic Acid	AmPhos	NaOH	MeCN	0.0258383665286105
6-chloroquinoline	2a, Boronic Acid	P(Cy)3	NaOH	MeCN	0.0444317065372505
6-chloroquinoline	2a, Boronic Acid	P(o-Tol)3	NaOH	MeCN	0.0194987400815928

Figure 3: The fisrt 6 reaction in our dataset

Data Set Characteristics:	Multivariate
Number of Instances:	3304
Area:	Chemistry
Attribute Characteristics:	Real
Number of Attributes:	6
Associated Tasks:	Regression

Table 1: Dataset Information

Attribute Information

- **Input variables**
  - ♣ Reactant 1 (7) :
    - ∗ '6-chloroquinoline'
    - ∗ '6-Bromoquinoline'
    - ∗ '6-triflatequinoline'
    - ∗ '6-Iodoquinoline'
    - ∗ '6-quinoline-boronic acid hydrochloride'
    - ∗ 'Potassium quinoline-6-trifluoroborate'
    - ∗ '6-Quinolineboronic acid pinacol ester'
  - ♣ Reactant 2 (4)
    - ∗ '2a, Boronic Acid'
    - ∗ '2b, Boronic Ester'
    - ∗ '2c, Trifluoroborate'
    - ∗ '2d, Bromide'
  - ♣ Ligand (9)
    - ∗ 'P(tBu)3'
    - ∗ 'P(Ph)3'

- ∗ 'P(Cy)3'
- ∗ 'P(o-Tol)3'
- ∗ 'SPhos'
- ∗ 'XPhos'
- ∗ 'dppf'
- ∗ 'Xantphos'
- ∗ 'P(Ph)3 '

♣ Base (7)

- ∗ 'NaOH'
- ∗ 'NaHCO3'
- ∗ 'CsF'
- ∗ 'K3PO4'
- ∗ 'KOH'
- ∗ 'LiOtBu'
- ∗ 'Et3N'

♣ Solvent (5)

- ∗ 'MeCN'
- ∗ 'THF'
- ∗ 'DMF'
- ∗ 'MeOH'
- ∗ 'MeOH/H2OV2 9:1'

- **Output variable**

♣ The Product Yield (yield between 0 and 1)

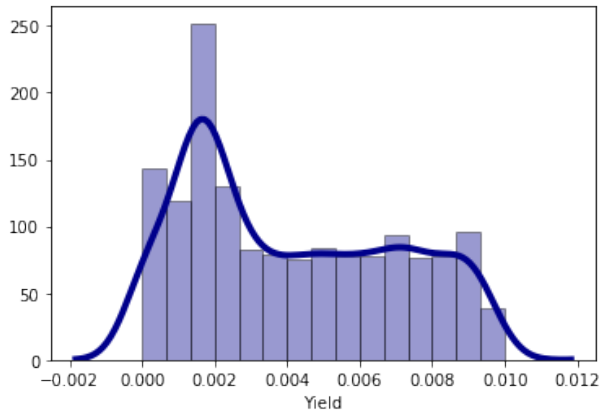


Figure 4: The yield distribution

Note:

We notice that our output variable is well distributed, we have values from all the ranges $[0, 1]$ (a little more closer to 0.2 value). Therefore we can suppose that our experiences are perfectly diversified.

2.2 Encoding

2.2.1 One hot encoding

For categorical variables there is no ordinal relationship, so the encoding by categories might be not enough to have an accurate model. We will one-hot encode each feature as its own category, eliminating any notion of chemical information. Each one-hot vector simply provides a reference to the entity. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected result.

2.2.2 Descriptors

With the help of our Chemists collaborators we created scripts to convert our integer encoding to molecular descriptors encoding for some components of the Suzuki-Miyaura cross-coupling reaction.

Using these descriptors as inputs and reaction yield as output, we want to investigate if our algorithm provides significant improvement for the predictive/exploration performance.

Base	**pKa in Water**	**Organic/Inorganic**	**# of Deprotonations**
NEt3	10,8	1	1
LiOtBu	17	1	1
CsF	3,2	0	1
K3PO4	2,1	0	3
KOH	15,7	0	1
NaHCO3	3,6	1	2
NaOH	15,7	0	1

Solvent	**PCA1**	**PCA2**	**PCA3**	**PCA4**
MeOH	-4.10648	-4.89092	0.651199	-0.305101
THF	0.424405	-1.34983	-1.44861	-1.35114
MeCN	-2.18069	-1.48531	-0.247267	-2.24492
DMF	-2.67998	0.527336	-1.58183	-0.775239
H2O	-9.22003	-3.67193	4.60824	-1.27075

Where:

- PCA1 : Polarity of Co-Solvent
- PCA2 : Polarisability
- PCA3 : Hydrogen Bonding
- PCA4 : Potentially a solubility parameter

Ligand	Angle	Denticity	Hybridisation	E (H)	E (L)	Steric	Proton Affinity
P(tBu)3	182	1	0	-0.173	0.028	23.38	250.8
P(Ph)3	145	1	0	-0.187	-0.051	8	241.3
P(Cy)3	170	1	0	-0.177	0.026	16.65	250.4
P(o-Tol)3	194	1	2	-0.182	-0.049	30.11	244.1
SPhos	208	1	1	-0.17	-0.038	24.06	255.96
XPhos	221	1	1	-0.181	-0.05	22.15	255.65
dppf	99.6	2	2	-0.1869	-0.0506	23.34	241.3
Xantphos	102.7	2	2	-0.1869	-0.0506	25.45	241.3

3 Modelling

In this section we compared the performance of the widely-used feed-forward back-propagation artificial neural network (ANN) with random forest (RF) on the error of predicting the yield of the reactions for both encoding of our dataset (Suzuki-Miyaura).

The literature for both models (random forest and neural networks) is suggested in many sources, here we are going to follow the one from "Trevor Hastie : The Elements of Statistical Learning Data Mining, Inference, and Prediction" [20]

3.1 Random Forest

Random Forest is a common machine learning algorithm within the supervised learning technique. It can be used for both classification and regression problems. It is based on the principle of the ensemble learning, which is a process of combining multiple classifiers in order to solve a complex problem and to enhance model efficiency [2].

In order to understand deeply the random first model we need to introduce first the concepts of *bootstrapping* and *bagging*.

3.1.1 Bootstrapping

We clarify how the normal uses of the bootstrap method work. Throughout this section, let Z be our training dataset of size N, where $Z = \{z_1, z_2, \cdots, z_N\}$ and $z_i = (x_i, y_i)$ are data points. Let $T(Z)$ be a statistic of interest that we can compute from a dataset Z.

Bootstrapping is used to have additional estimation of a specific statistic on randomly sampled M subsets Z_i of the dataset with replacement, where $|Z_i| = |Z| = N$. The pseudo-code of the algorithm is listed in 1

Algorithm 1 Bootstrap

Require: Train dataset Z, Statistic T

1: Choose a number of bootstrap samples to perform : M.

2: **for** `i = 1 to M` **do**

3: (1) Calculate the bootstrap sample Z_i by drawing randomly with replacement of size N from the training data Z.

4: (2) Calculate $T(Z_i)$ the quantity of interest on the each bootstrap training set.

5: **Return:** The values $T(Z_i), \cdots, T(Z_M)$

The next section makes use of the Bootstrap method.

3.1.2 Bagging (Bootstrap Aggregation)

Let $\hat{f}$ be a trained model fit on Z, giving prediction $\hat{f}(x)$ for a data point x.

The *Bagging* method uses bootstrapping (Algorithm 1) on Z where T is considered to be $\hat{f}$ trained on a bootstrap subset of Z, and simply takes the average of $T(Z_i)(x)$ for $i = 1, \ldots, m$ to produce a prediction for data point x. The pseudo-code for bagging is is defined with the following algorithm 2

Algorithm 2 Bagging

Require: Training sample $Z = (z_1, z_2, \cdots, z_N) = ((x_1, y_1), (x_2, y_2), \ldots (x_N, y_N))$

1: Produce the bootstrap subsets Z_i as in Algorithm 1.

2: **for** `m = 1 to M` **do**

3: Fit our model with the sample Z_i to produce the estimator $\hat{f}_i$.

4: Then the bagging (or, bagged) estimate is:

$$\hat{f}_{\text{bag}} \coloneqq \frac{1}{M} \sum_{m=1}^{M} \hat{f}_i \tag{1}$$

5: **Return:** Estimator $\hat{f}_{\text{bag}}$

3.1.3 Random Forest algorithm

As we have seen in (Section 3.1.2), the basic principle in bagging is to combine many models. Bagging tends to work especially well for *low-bias*, *high-variance* procedures, like trees that are considered ideal candidates for bagging, as they can catch complex connection structures in the data, and have generally less bias when grown deep enough. The pseudocode is illustrated in Algorithm 3.

Algorithm 3 Random Forest for Regression [20]

1: **for** `i = 1 to M` **do**

2: (a) Draw a bootstrap sample $\mathbf{Z_i}$ of size N from the training data.

3: (b) Grow a random-forest tree T_i to the bootstrapped data.

4: Output the ensemble of trees $\{T_i\}_1^M$

5: To make a prediction at a new point x for a Regression problem :

$$\hat{f}_{\text{rf}}^M(x) = \frac{1}{M} \sum_{i=1}^{M} T_i(x)$$

3.2 Neural Network

An Artificial Neural Network is a computational model inspired by the way biological neural networks in the human brain process information.(Fig 5). Similar to a human brain, neurons are linked, artificial neural networks often have neurons connected in separate layers of the networks. Those neurons are called nodes.

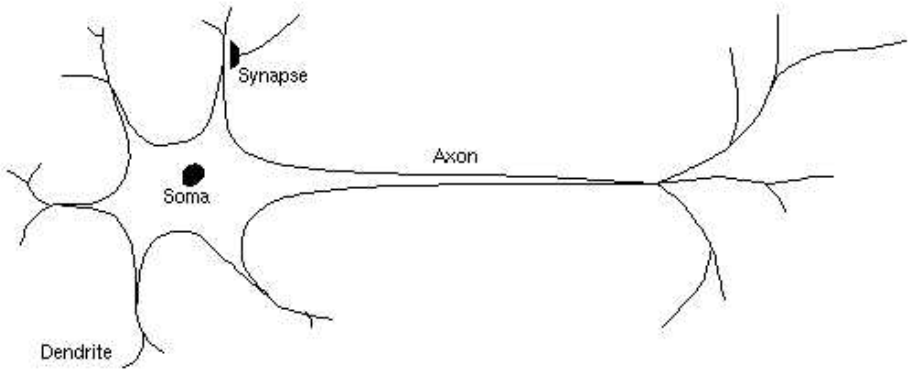


Figure 5: A biological neuron [1]

In this section we describe the class of neural network methods. The central idea is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features. The result is a powerful learning method, with widespread applications in many fields.

3.2.1 Neural Networks description

Without loss of generality, we define neural networks with a single layer. Neural networks of multiple layers (referred to as multi-layer perceptron (MLP)) are stacks of multiple layers one on top of another and follow the same principles and computations.

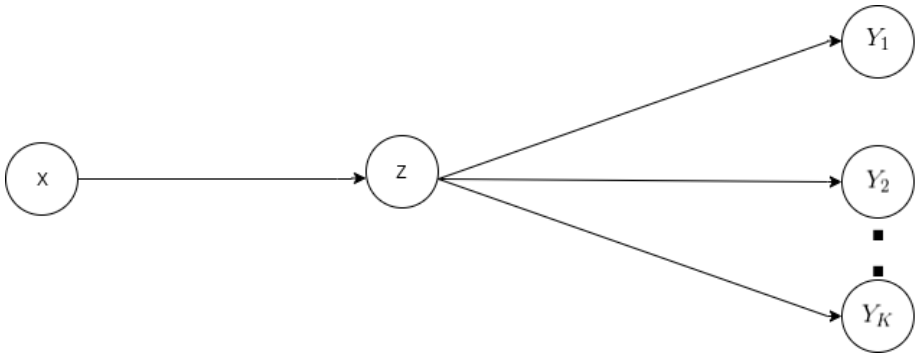


Figure 6: Schematic of single hidden layer, feed-forward neural network [20].

Neural networks are a mathematical function with a huge number of parameters allowing the approximation of a complex function f specified by a dataset exhibiting the relationship between inputs and their desired outputs. The following algorithm 4 illustrates the how neural nets are mathematically modeled.

Algorithm 4 Single hidden layer neural network

Require: sample $X = (x_1, x_2, \cdots, x_p)$, Parameters $\alpha_{0m}, \alpha_m, \beta_{0k}, \beta_k$

1: **for** `m = 1 to M` **do**

2: Calculate the hidden layer by computing the linear combination of the inputs and feed it with a non-linear function a.

$$Z_m = a\left(\alpha_{0m} + \alpha_m^T X\right), \quad m = 1, \ldots, M$$

3: Put the calculated single hidden nodes in one vector:

$$Z = (Z_1, Z_2, \cdots, Z_M)$$

4: **for** `m = 1 to k` **do**

5: Calculate the output layer by computing the linear combination of the inputs of the hidden layer.

$$\hat{f}_k(X) = \beta_{0k} + \beta_k^T Z, \quad k = 1, \ldots, K$$

6: **Return** : $\hat{f}(X) = (\hat{f}_1(X), \hat{f}_2(X), \cdots, \hat{f}_K(X))$ called the logits of X.

Note that if a is a linear, then the neural network becomes a linear model which lacks the expressiveness and capacity of non-linear models. It is proven in [16] that neural networks satisfying mild conditions can sufficiently approximate any continuous function given enough data and appropriate activation functions (non-polynomial in the cited reference).

The goal is to learn the optimal parameters that minimize (or maximize) an objective function. We can achieve this for example using gradient-descent base methods (e.g Stochastic gradient decent) with the back-propagation algorithm [15].

3.3 Evaluation metrics

To assess both models' performance, we used different metrics, we denoted our final output that we are trying to predict as y_i and $\hat{y}_i$ as there prediction.

1. **Root mean squared errors** (RMSE) : is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(\mathrm{y}_i - \hat{\mathbf{y}}_i\right)^2}{N}} \tag{2}$$

2. **Mean absolute deviation** (MAD) :

$$\mathrm{MAD} = \frac{1}{N} \sum_{i=1}^{N} |\hat{\mathbf{y}}_i - \mathbf{y}_i| \tag{3}$$

3. **Coefficient of variation** (CV): measures the variation in error with respects to the actual consumption average $\hat{\mathbf{y}}$ and is defined by:

$$CV = \frac{\sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}}}{\hat{\mathbf{y}}} \times 100 \tag{4}$$

and

$$\hat{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^{N} \hat{\mathbf{y}}_i \tag{5}$$

4. **Coefficient of determination** (R^2) : It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{6}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual values, $\bar{y}$ is the mean of the observed values and N is the total number of samples.

We used the implementation of random forests and neural nets included in the scikit-learn module of python programming language. All development and experimental work was carried out on a personal computer (1.8 GHz Intel Core i5, with 8 GB of RAM).

3.4 Results

3.4.1 Calibration plots

Calibration plots of "Actual vs Predicted" are one of the richest form of data visualization. We can tell pretty much everything from it. Theoretically, we want all the points to be on the diagonal line. So, if the Actual is 0, your predicted should be reasonably close to 0 to. If the Actual is 0.8, your predicted should also be reasonably close to 0.8. If your model had a high $\mathbf{R^2}$, all the points would be close to this diagonal line. The lower the $\mathbf{R^2}$, The lower the $\mathbf{R^2}$, the poorer the quality of your estimation and the more cloudy your points are.

The following scatter plots are performed over 20% of the data which is about 661 reactions.

- **For One hot encoding dataset**

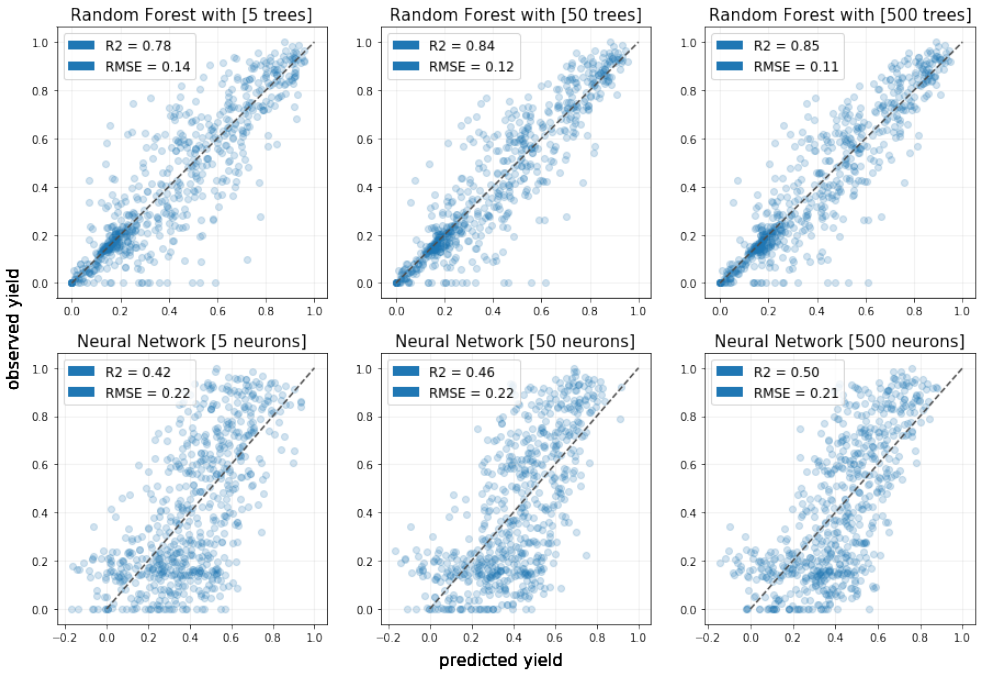


Figure 7: Calibration plots for one hot encoding dataset

- **For Descriptors dataset**

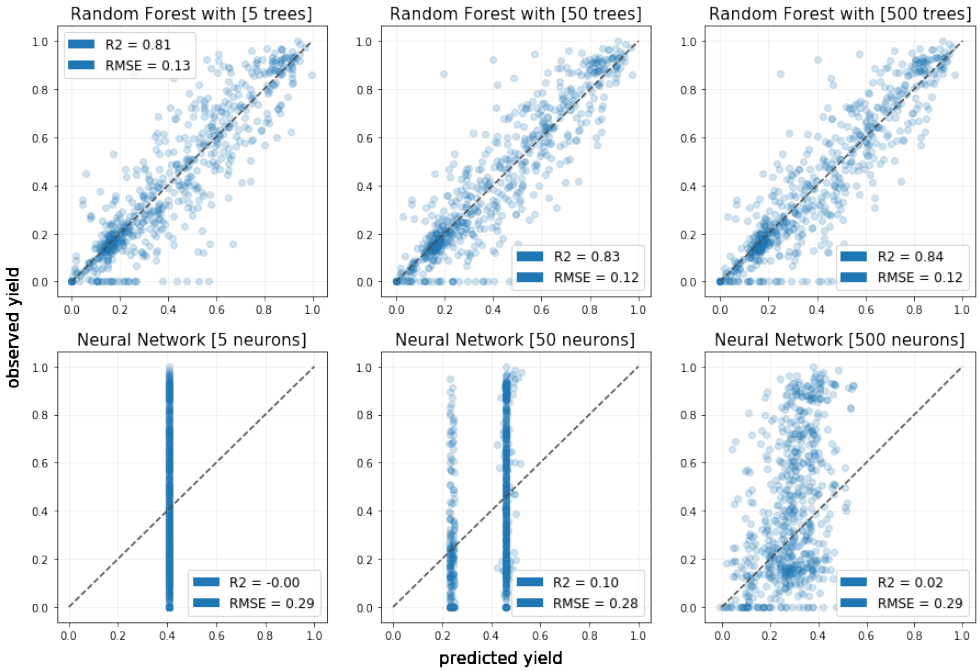


Figure 8: Calibration plots for descriptors dataset

We can see that our models (the neural nets and the random forest) seem to have different performances depending on the encoding and hyperparameters (number of trees in the random forest and number of neurons in the neural network).

Random Forest:

For all the plots we can see that the models seem to have three subsections of performance.

- The first one is where actuals have values between 0 and 0.3 Within this zone, the model does good predictions for both encoding.
- The second one is where actuals are between 0.3 and 0.8, within this zone, the model is a bit more random, but obviously we can see that there is a relationship between the model's predicted values and actuals.
- The third zone is for actuals > 0.8. Within this zone, the Random forest model steadily greatly estimates the actual values.

Neural nets:

After trying different meta parameters for:

- **Optimization** like solvers {'lbfgs', 'sgd', 'adam'}
- **Architecture** like activation functions {'identity', 'logistic', 'tanh', 'relu'} and hidden layer sizes {5, 50, 500}

We can confirm that the neural network performance under those parameters and the encoding with descriptors is very bad and the neural network doesn't fit well the descriptors encoding. However they give descent results with the one hot encoding but still underestimates the actual values.

3.4.2 Learning curves

The learning curve provides a measurement of predictive performance on various amounts of data [3]. In our work, the learning curves determine cross-validated training and test of the negative root mean square error score for different training set sizes. To do so we are cross-validating by splitting randomly the whole dataset 10 times in training and test data. The training subsets will be used to train the model and then compute the scores for each one. Afterward, we will average and compute the variance of scores over all 10 runs for each training subset size (the semi-transparent area represent the variance bounds).

- **For One hot encoding dataset**

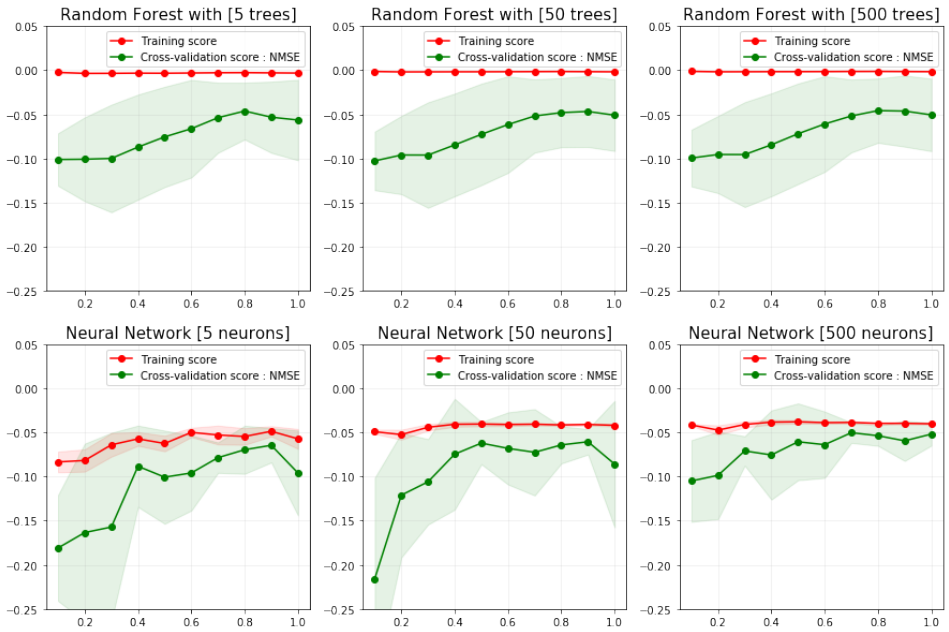


Figure 9: Learning curve with negative mean square error score for one hot encoding dataset

- **For Descriptors dataset**

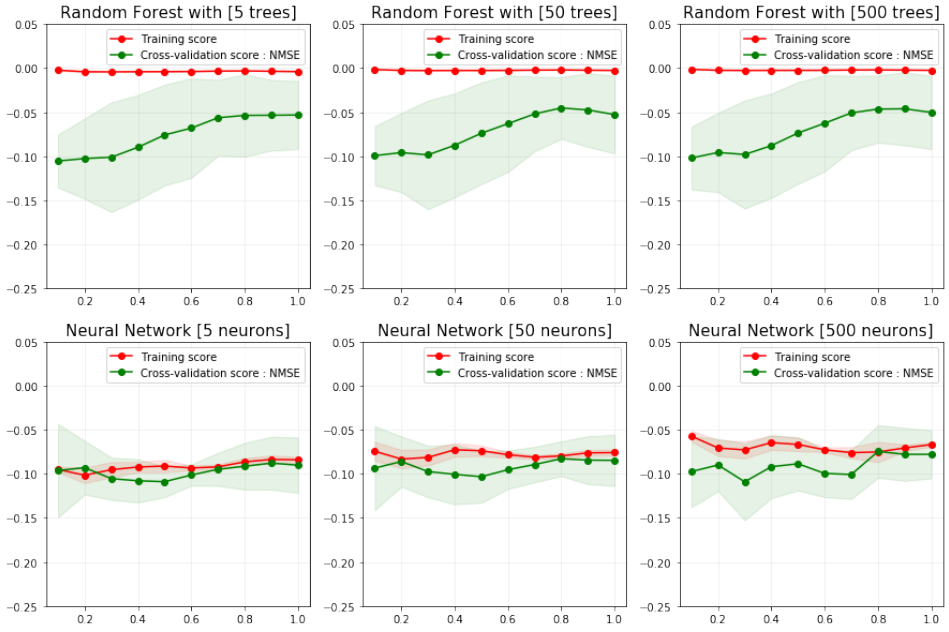


Figure 10: Learning curve with negative mean square error score for descriptors dataset

Comments:

The goal is to keep errors as small as possible. Two primary sources of error are bias and variance. In order to create more accurate models, we should reduce these two factors.

We notice that our learning algorithm (random forests) suffers from high variance and quite a low bias with overfitting the training data (since the red curve is constant). On the other hand, the learning curve for the neural network confirms the overfitting that we sow with the calibration plots before (the negative root mean square error for test data is not increasing).

3.4.3 What's the best model?

Basing on the previous plots (calibration curves and learning curves) we can easily choose the random forest over the neural networks for our exploration strategy, and we can confirm our choice with the following comparisons based on the evaluation metrics.

		R^2	RMSE	MAD	CV
OHE	RF with 500 trees	**0.845**	**0.114**	**0.076**	**28.08**
	NN with 500 neurons	0.502	0.206	0.171	49.83
Descriptors	RF with 500 trees	0.83	0.117	0.079	28.79
	NN with 500 neurons	0.01	0.28	0.22	101.7

4 Exploration with Bayesian Optimization

4.1 Bayesian Optimization framework

The goal of this section is to present the Bayesian optimization mechanism, which is a strategy to find the highest of pricey cost functions. Bayesian optimization uses the Bayesian methodology of setting a prior over the observed function and integrating it with data to achieve a posterior result. This section relies heavily on formulas and observations from [17][8].

The next evaluation to be made on the objective function, must take into account :

- **Exploration:** by sampling from areas of high uncertainty.
- **Exploitation:** by sampling from areas that are likely to provide improvement over the current best.

4.1.1 Concepts of Bayesian optimization

Bayesian optimization is an optimization method with the goal solving:

$$\max_{x \in \mathcal{X}} f(\mathbf{x})$$

where $f(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ is an objective function.
This optimization approach is based on modeling values of f by probability distributions, and to infer the distribution of f at an unseen point $x \in \mathcal{X}$ from previous points using conditional probability on seen data points.

To perform Bayesian optimization, we need two main components:

1. Define a *Gaussian process* as the underlying probabilistic model of the objective function at $\mathcal{X}$
2. Define an *acquisition function* that we use to acquire the next data point.

The essence of Bayesian optimization is that the distributions involved and the acquisition function are cheaper to compute.

4.1.2 General Bayesian optimization algorithm

The goal is to get around the question of optimizing an objective function that is expensive and probably not even analytically accessible to a series of acquisition function optimizations that are easier to test.

In the next sections, we will review the details of the necessary components to formally give a general Bayesian optimization algorithm (inspired by [17]).

Algorithm 5 Bayesian optimization algorithm

Require: Objective function evaluator f, Input space $\mathcal{X}$, acquisition function **a**

1: Generate an initial sample of points X_0 and associated objective function evaluations f_0.
2: **while** not criterion **do**
3: Maximize the acquisition function **a** using the posterior mean and variance under the Gaussian process assumption:

$$\mathbf{x}_{i+1} = \arg\max_{x \in \mathcal{X}} \; a(\mathbf{x})$$

4: Evaluate the objective function at this point and put $f_{i+1} = f(\mathbf{x}_{i+1})$
5: Update the posterior probability distribution on f using all available data $\{x_1, \ldots, x_{i+1}\}$.
6: **return** : The maximum f_{max} to be the maximum value in the final f.

4.1.3 Surrogate model

Gaussian process (GP) is the most popular used surrogate model for Bayesian optimization since it gives closed analytical formulas for the acquisition function (see section 4.1.4). GPs incorporate prior beliefs like "smoothness" over the objective function. However the GP posterior is our case is not adequate for our data since the data is categorical. So to use this Bayesian optimization framework we are going to build our proper surrogate model based on the Random forest regressor.

Algorithm 6 The Surrogate model

Require: Training data $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, Number of trees : B

1: Use the Algorithm [3] to fit our data $\mathcal{D}$ to the random forest model and get the ensemble of trees $\{\mathbf{T}_b\}_b^B$.

2: Compute the mean function for our GP:

$$\begin{aligned} m(x) &= \mathbb{E}(f(x)) \\ &\approx \frac{1}{B}\sum_{b=1}^{B} T_b(x) \qquad \text{(for } B \text{ large enough)} \end{aligned}$$

3: Compute the covariance function for our GP:

$$\begin{aligned} k(x, x^{'}) &= \mathbb{E}[(f(x) - m(x))(f(x^{'}) - m(x^{'}))] \\ &\approx \frac{1}{B}\sum_{b=1}^{B}(T_b(x) - m(x))(T_b(x^{'}) - m(x^{'})) \qquad \text{(for } B \text{ large enough)} \end{aligned}$$

4: **return** : Gaussian Process with mean function $\mu(x)$ and the covariance function $k(x, x^{'})$

4.1.4 Acquisition function

The acquisition function $\mathbf{a}$ and the goal is to find which point in $\mathcal{X}$ is the next one to examine the true objective function at.

$$\mathbf{a} : \mathcal{X} \to \mathbb{R}$$

We hope that the acquisition function values correspond to the high true objective values. Formally:

$$\mathbf{x}_{\text{next}} = \arg\max_{x \in \mathcal{X}} a(\mathbf{x}) \tag{7}$$

We make use of the fact that $a(\mathbf{x})$ is relatively straightforward to optimize since its assessment at any point relies just on the probabilistic model.

In the rest of this section, $f(x)$ for a given $x \in \mathcal{X}$ is a normal random variable with mean $\mu(x)$ and variance $\sigma^2(x)$. We will denote by $f\left(\mathbf{x}^+\right)$ the maximum value of the true objective function observed at the data points defined in Formula 7.

An acquisition function preferably helps finding the rights points that yield either a high posterior mean or posterior variance (or both) which basically taking both exploration and exploitation into account as mentioned in section 4.1.

In our experiments, we try the following acquisition functions:

♣ Probability of improvement (PI)

The first acquisition function was the probability of improvement (PI) in [14]. The core concept is to maximize the probability of improving the current best value of $f\left(\mathbf{x}^+\right)$. Since we suppose that distribution of $f(\mathbf{x})$ is normal with mean μ and variance σ^2 then acquisition function is described as the following:

$$\begin{aligned}
\mathrm{PI}(\mathbf{x}) &= \mathbb{P}\left[f(\mathbf{x}) > f\left(\mathbf{x}^+\right)\right] \\
&= 1 - \mathbb{P}\left[f(\mathbf{x}) < f\left(\mathbf{x}^+\right)\right] \\
&= 1 - \mathbb{P}\left[\frac{f(\mathbf{x}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})} < \frac{f\left(\mathbf{x}^+\right) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right] \\
&= 1 - \Phi\left(\frac{f\left(\mathbf{x}^+\right) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) \\
&= \Phi\left(\frac{\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)}{\sigma(\mathbf{x})}\right).
\end{aligned}$$

This strategy is susceptible to preferring points with high variance, which leads to over-exploitation of points near the mean.

♣ Expected improvement (EI)

The Expected Improvement (EI) was introduced in [10] as a way to mitigate the issue with PI by using the expected value of PI instead of merely taking points maximizing it. This results in an acquisition function also dependent on the magnitude of $f - \mu$.

Let's first defined some quantities:

- Improvement function $I(x)$:

$$\mathrm{I}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) - f\left(\mathbf{x}^{+}\right), & f(\mathbf{x}) > f\left(\mathbf{x}^{+}\right). \\ 0, & \text{otherwise.} \end{cases}$$

- Change of variable 1:

$$Z = \frac{f - \mu}{\sigma}$$

- Change of variable 2:

$$f'_{+} = \frac{f_{+} - \mu}{\sigma}$$

The goal is to compute the expectation of I which is simply a function of a random variable which will then be the wanted acquisition function $\mathbb{E}[\mathrm{I}]$.

By using the variables change Z and f we notice that the improvement function can then be rewritten as :

$$\mathrm{I} = \begin{cases} \sigma\left(Z - f'_{+}\right), & Z > f'_{+}, \sigma > 0. \\ 0, & \text{otherwise.} \end{cases}$$

So have two cases:

♠ When $\sigma = 0$ or $Z \leq f'_{+}$:

$$\mathbb{E}[\mathrm{I}] = 0$$

♠ When $\sigma \neq 0$ and $Z > f'_{+}$:

$$
\begin{aligned}
\mathbb{E}[\mathrm{I}] &= \sigma \int_{f'_+}^{\infty} \left(z - f'_+\right) \phi(z) dz. \\
&= -\sigma \int_{-f'_+}^{-\infty} \left(-t - f'_+\right) \phi(-t) dt \qquad\qquad (z = -t) \\
&= \sigma \left(\int_{-\infty}^{-f'_+} -t\phi(t) dt - f'_+ \int_{-\infty}^{-f'_+} \phi(t) dt \right) \\
&= \sigma \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-f'_+} -t \exp\left(-\frac{t^2}{2}\right) dt - f'_+ \Phi\left(-f'_+\right) \right) \\
&= \sigma \left(\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\left(f'_+\right)^2}{2}} \exp(u) du - f'_+ \Phi\left(-f'_+\right) \right) \qquad (u = -\frac{t^2}{2}) \\
&= \sigma \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(f'_+\right)^2}{2}\right) - f'_+ \Phi\left(-f'_+\right) \right) \\
&= \sigma \left(\phi\left(f'_+\right) - f'_+ \Phi\left(-f'_+\right) \right) \\
&= \sigma\phi\left(\frac{\mu - f_+}{\sigma}\right) + (\mu - f_+) \Phi\left(\frac{\mu - f_+}{\sigma}\right).
\end{aligned}
$$

♣ **Upper confidence bound (UCB)**

The *upper confidence bound* [4] combines exploitation (μ) and exploration (σ) with the specification of a hyper-parameter β to control the importance of exploration in the optimization problem. Formally:

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \beta\sigma(\mathbf{x}).$$

In practice, we fine-tune the value of β to find the best trade-off between exploration and exploitation that yield the optimal results.

5 Results

5.1 Workflow description

As we mention before the strategy we developed is an adaptive Gaussian process based on a random forest model that navigate smartly the search space with only a few data points [k = 12] as a starting train data to build a simple and basic model from which a new experiment is suggested. Our algorithm doesn't suppose any prior assumptions on the

model. Considering that each selected reaction is informative, the ideal surrogate model parameters change dynamically.

As we described in the previous section the Bayesian Optimization adopts an exploitative approach, so we are selecting the least confident of the perceived top-k high-yielding reactions with k is called "the screen size" in our framework (in our experiment we have chosen k to be 6)

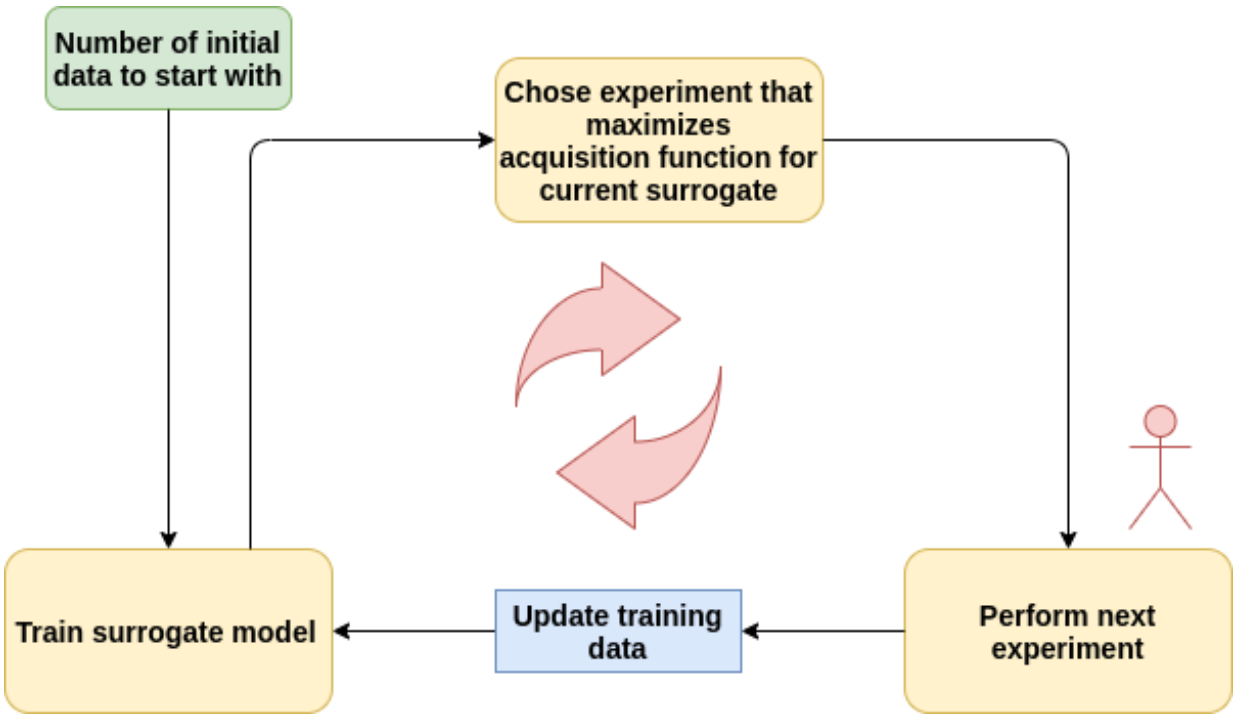


Figure 11: Workflow for our reaction optimization procedure.

The figure (11) illustrate workflow for our reaction optimization procedure. Reactions are performed with computationally suggested conditions and subsequently assessed at the chemistry lab. The reaction conditions are used as features for the training surrogate model.

5.2 Validation of our strategy

To Validate our algorithm, we are going to test it on the Suzuki-Miyaura reaction dataset described in section 2. The following analysis is simulated 50 times to remove the noise from the predictions and have more stable and consistent results.

- **For One hot encoding dataset**

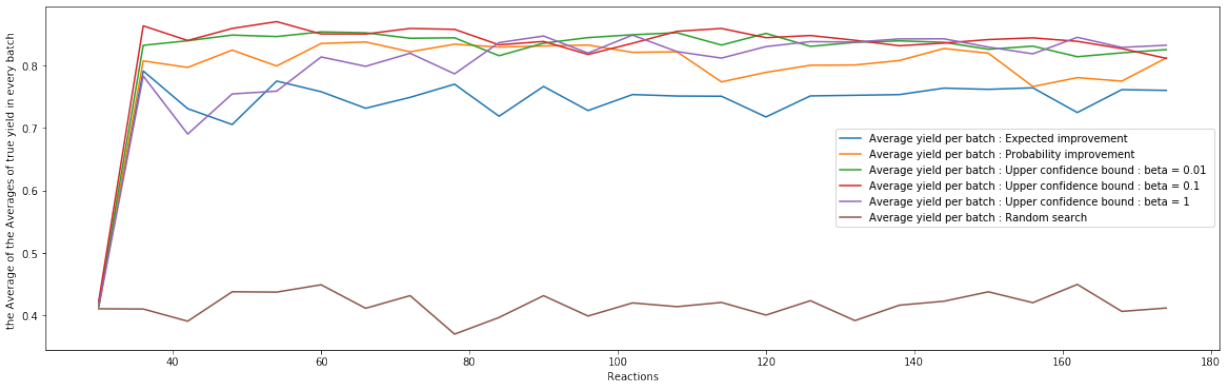


Figure 12: The average of the yield improvement with batch size = 6 for OHE dataset

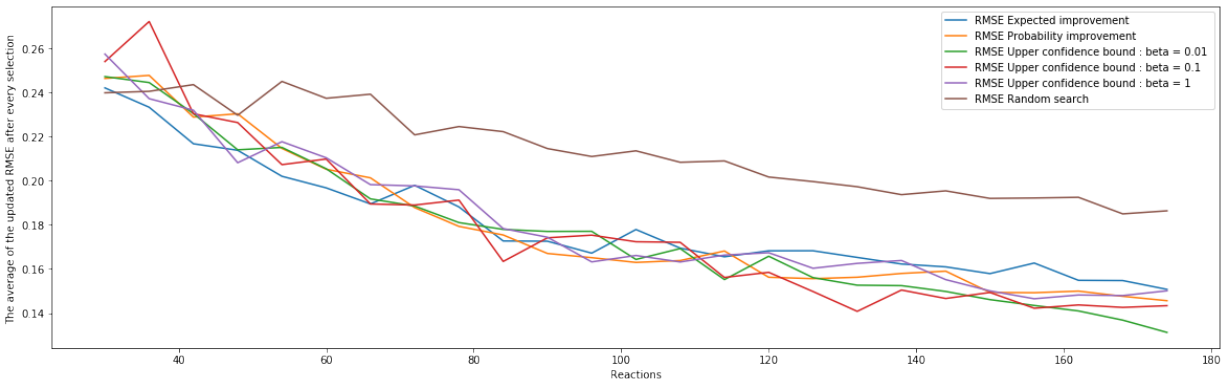


Figure 13: The average of the RMSE improvement with batch size = 6 for OHE dataset

- **For Descriptors dataset**

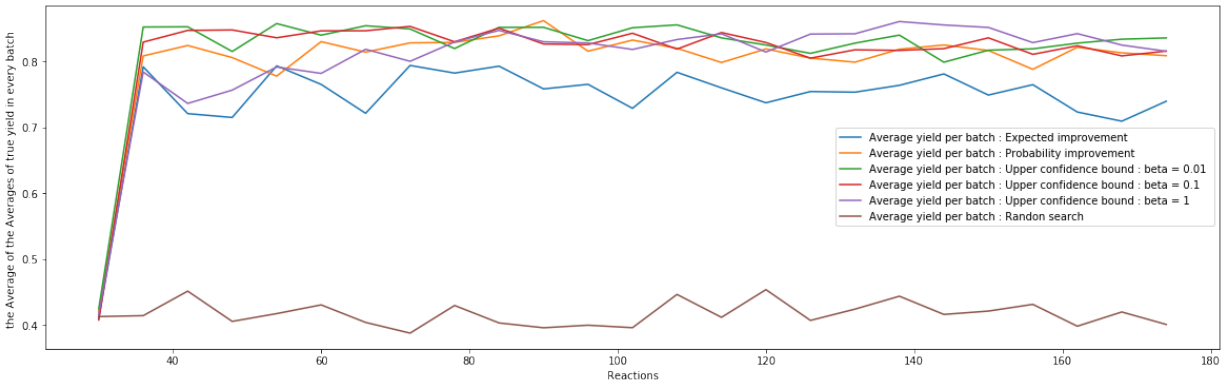


Figure 14: The average of the yield improvement with batch size = 6 for descriptors dataset

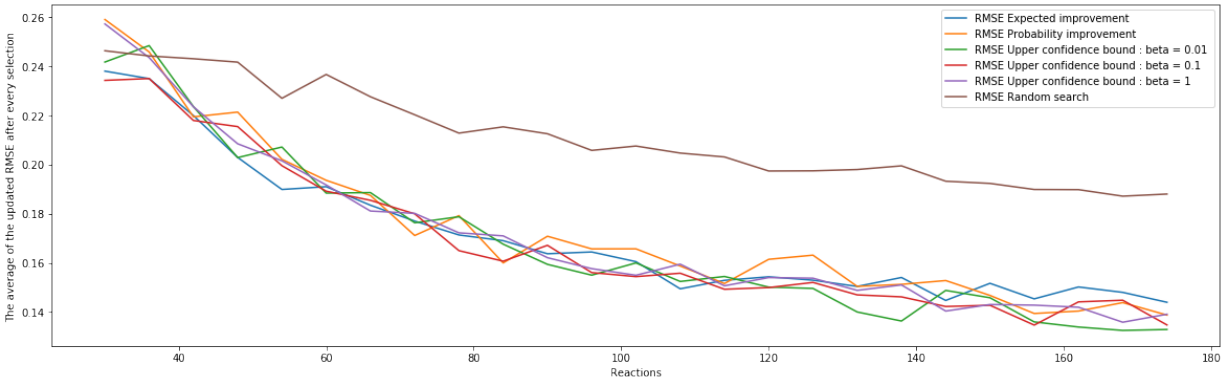


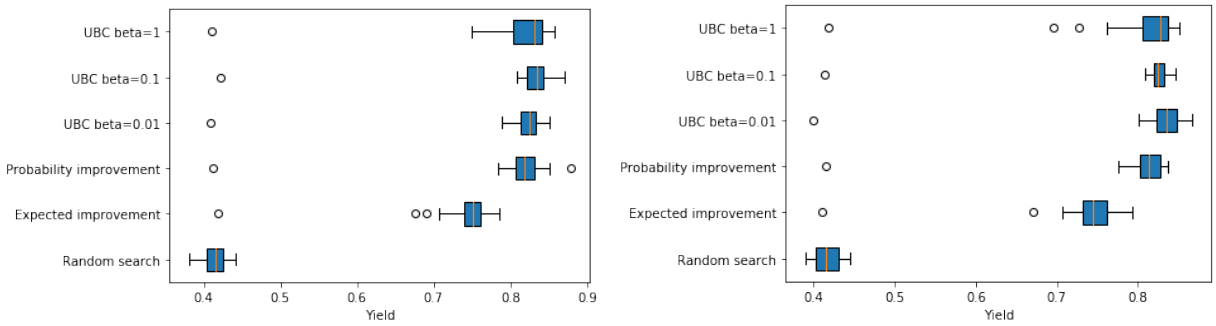
Figure 15: The average of the RMSE improvement with batch size = 6 for descriptors dataset

As an initial training set, we provided our strategies with only 12 random reactions, representing a minute amount (12/3304 or 0.3%) of the vast multidimensional search space (here compressed to 3304 discrete combinations). Those 12 random reactions provided a range of 40% yield which is the mean of our yield distribution (Fig. 4).

The first thing to notice is that any strategy of the Bayesian optimization (expected improvement, probability improvement, upper confidence bound) beat easily the random search strategy. On average, in a smaller number of steps, Bayesian optimization finds a better optimum than random search and beats the baseline in almost every run. For larger search spaces, the effect becomes even more effective. Here, the search space is 5-dimensional (Reactant 1, Reactant 2, Ligand, Base, Solvent) which is rather low to substantially profit from Bayesian optimization.

Between iterations 12–18 (after the first 6 selected reactions) (Fig 12 & 14) we can see the huge jump of the yields picked reactions (from 40% to 80%). So we can obviously confirm that with only our 12 initial reactions, the optimal reaction conditions were achieved.

Another interesting fact about our Bayesian optimization strategies is that the slope of the average RMSE improvement (the learning) is much faster when selection is by smart policies instead of the random search (Fig 13 & 15).



(a) One hot encoding dataset

(b) descriptors dataset

Figure 16: This is box plots for yield using Bayesian optimization strategies and random search

The last plot is about the distribution of the average yield calculated between a given reaction and all previous iterations (for both encodings) by using all the Bayesian optimization strategies described before. We conclude that the strategy with the "upper confidence bound" with a well-tuned β parameter that interpreted as the amount of preferred exploration ($\beta \in [0.01, 0.1]$) perform extremely better than the random search and slightly on the others strategies.

6 Conclusion

This report introduces a framework for accelerating chemistry tasks in the presence of small data sets. The idea is to use machine learning with Bayesian optimization. We expect that machine learning will find broad applicability in accelerating discovery chemistry, democratizing chemical syntheses, eliminating non-informative experiments, and freeing chemists for non-routine tasks. We have shown that our method employing a simple machine learning simple algorithm enables synthetic chemistry research/industry to achieve great results and use chemical knowledge to navigate inside reaction condition spaces, identify optimized reaction conditions, predict conversions for relevant chemistry and provide reactivity insights. Our adaptive approach to learning can be extended to any chemical transformation since the algorithm can't identify the chemical reaction.

References

[1] The perceptron. https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Neuron/index.html.

[2] Random forest algorithm. https://www.javatpoint.com/machine-learning-random-forest-algorithm. Accessed: 2018.

[3] P. C. Learning curves in machine learning. *Springer*, 2011.

[4] D. D. Cox and S. John. Sdo: A statistical method for global optimization. *SIAM, Philadelphia*, 1997.

[5] S. B. C. J. H. A. C. W. F. P. R. Damith Perera1, Joseph W. Tucker. A platform for automated nanomole-scale reaction screening and micromole-scale synthesis in flow. *Science*, 2018.

[6] M. Denninger. An efficient probabilistic online classification approach for object recognition with random forest. Technical report, Technical University of Munich, 2017.

[7] S. L. S. D. A. G. Derek T.Ahneman, Jesus G.Estrada1. Predicting reaction performance in c–n cross-coupling using machine learning. *Science*, 2018.

[8] V. M. C. Eric Brochu and N. de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. 2010.

[9] S. Fadel. Synthetic reaction optimization. https://github.com/sfade070/Chemistry_machine_learning_project, 2020.

[10] V. T. J. Mockus and A. Zilinskas. The application of bayesian methods for seeking the extremum. *In Towards Global Optimization*, 1978.

[11] V. D. D. L. L. C. Jarosław M. Granda, Liva Donina. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature*, 2018.

[12] R. P. S. S. D. D. A. G. D. Jesús G. Estrada1, Derek T. Ahneman1. Response to comment on "predicting reaction performance in c–n cross-coupling using machine learning". *Science*, 2018.

[13] M. J. K. Kangway V. Chuang. Comment on "predicting reaction performance in c–n cross-coupling using machine learning". *Science*, 2018.

[14] H. J. Kushner. A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[17] A. Matosevic. On bayesian optimization and its application to hyperparameter tuning. Technical report, Linnaeus University, 2018.

[18] D. R. G. B. T. Rodrigues. Evolving and nano data enabled machine intelligence for chemical reaction optimization. *ChemRxiv*, 2018.

[19] M. Shevlin. Practical high-throughput experimentation for chemists. *ACSPublications*, 2017.

[20] J. F. Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2008.