2022年度機械学習ハンズオン講習会「材料DX Boot camp」
15:00 – 16:00 セミナー

ベイズ最適化を使った固体電解質の効率的な探索

名工大:中山将伸

目次

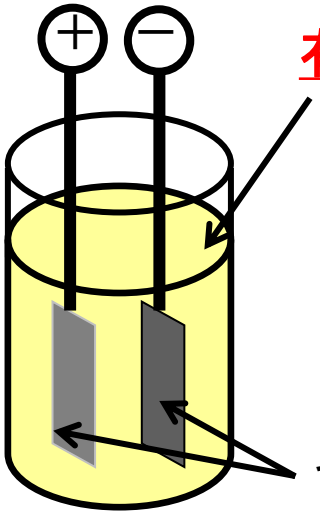
1）ベイズ最適化による固体電解質材料組成の最適化

2）ベイズ最適化 python codingについて

これからの電池：無機全固体電池

〈現在〉

リチウムイオン電池



有機電解液

× 可燃性

× 揮発性

○高イオン伝導性

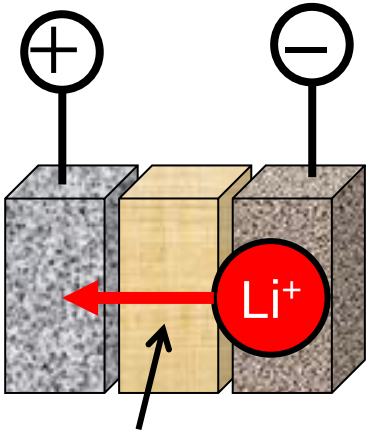
セラミックス

用途：ポータブル機器用電源

安全性に問題

〈将来〉

全セラミックスLi電池



セラミックス固体電解質

○不燃性

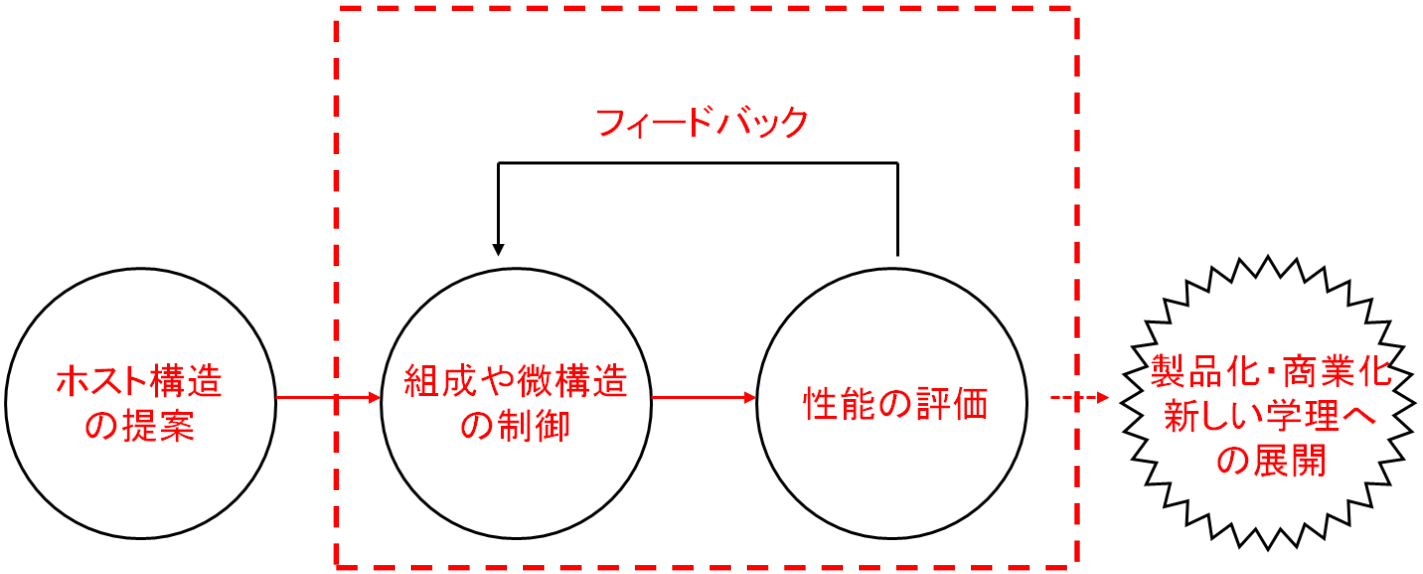
○不揮発

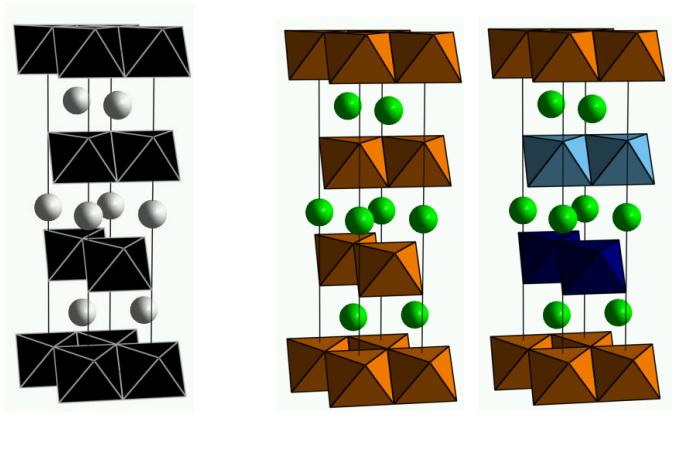
× 低イオン導電性

用途：電気自動車

家庭用発電システム

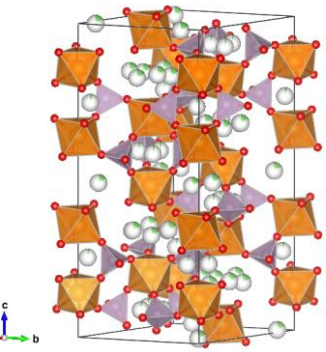
新しい材料と製品化の流れ



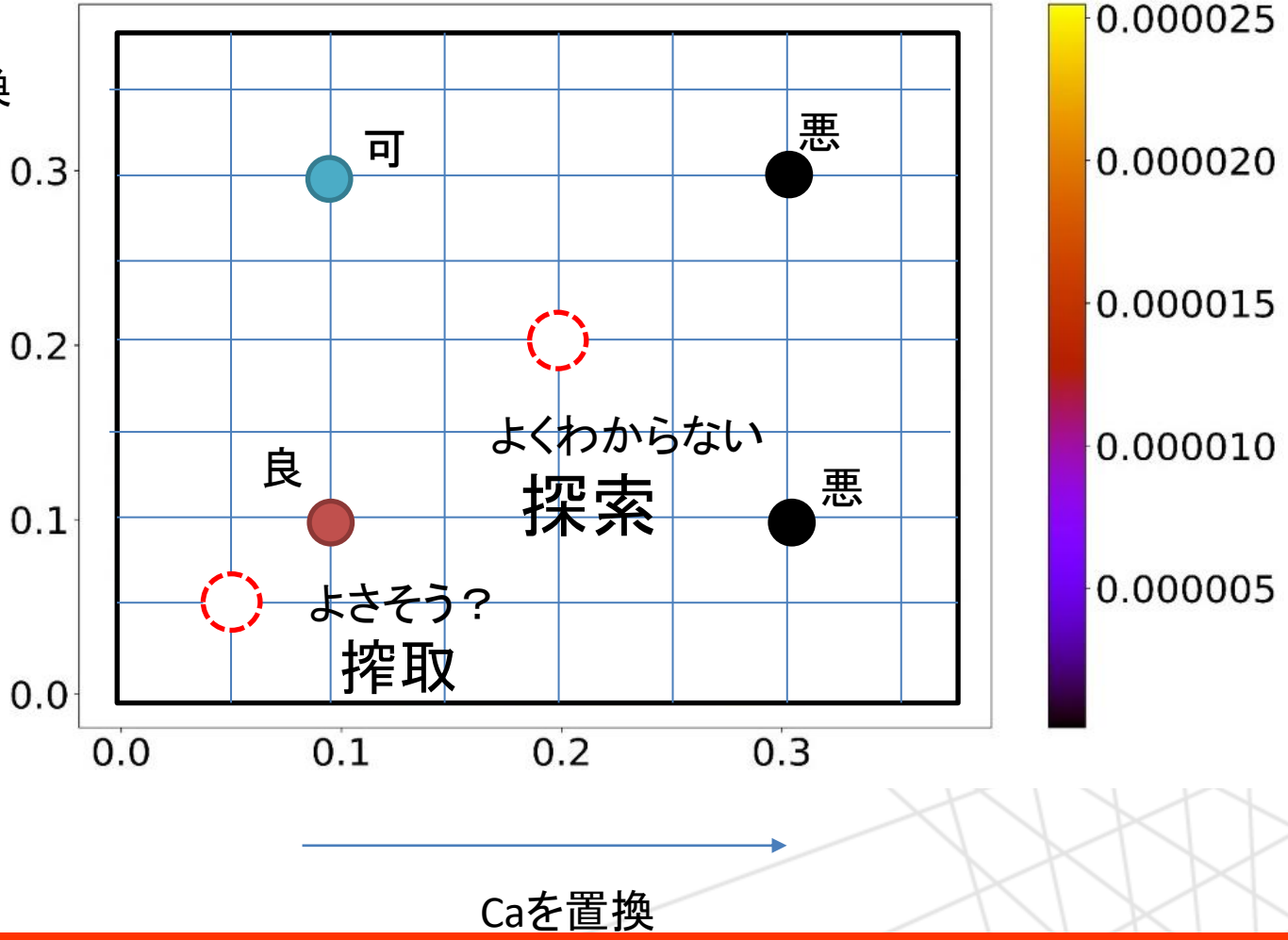


ベイズ最適化：実験との連携：組成最適化を効率よく

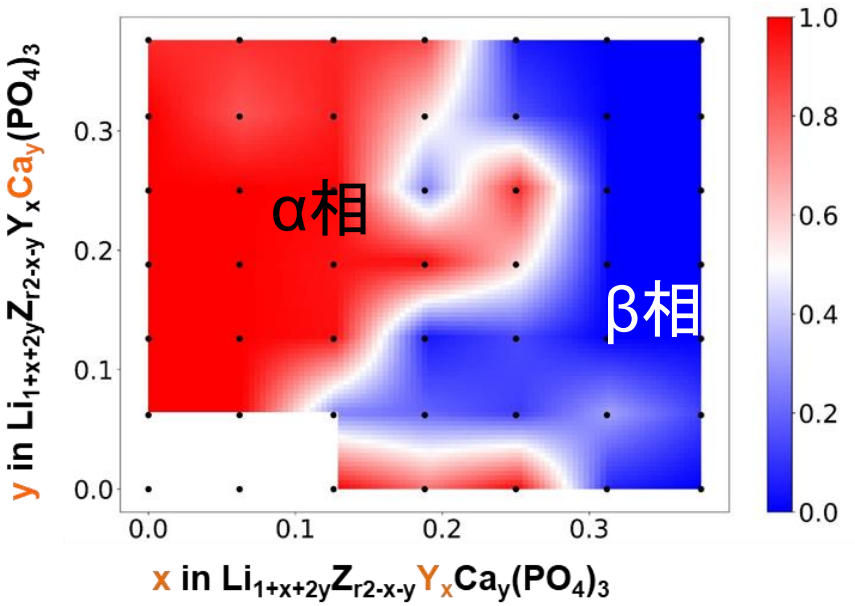
Step1　最初はランダムサンプリング
Step2　空気を読む（予測）
Step3　よくなりそうなところと、不明瞭なところのどちらかを次の候補に

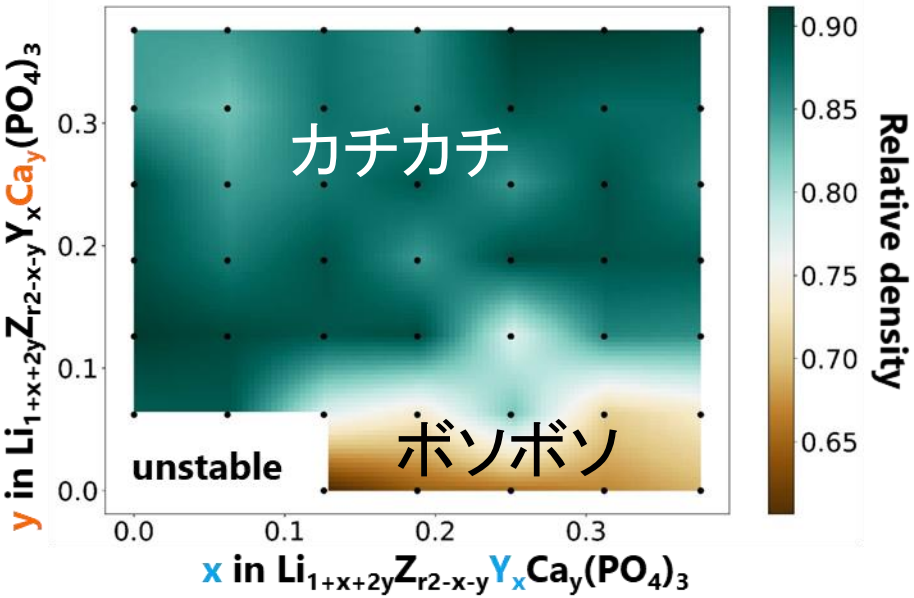


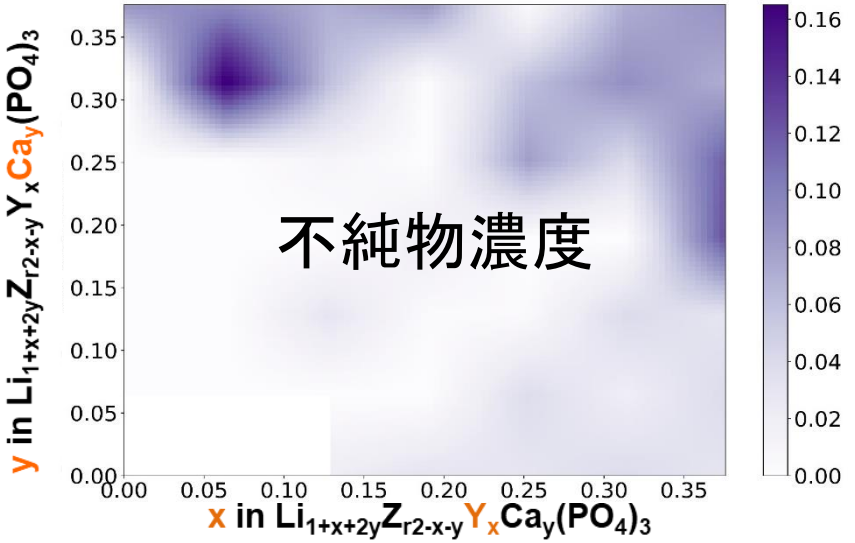
母材料

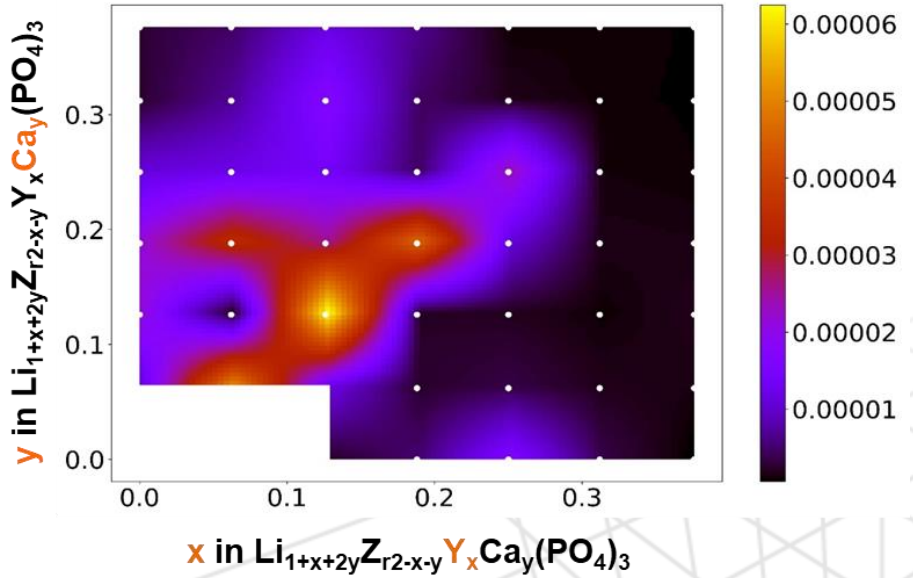


事例2：組成最適化を効率よく

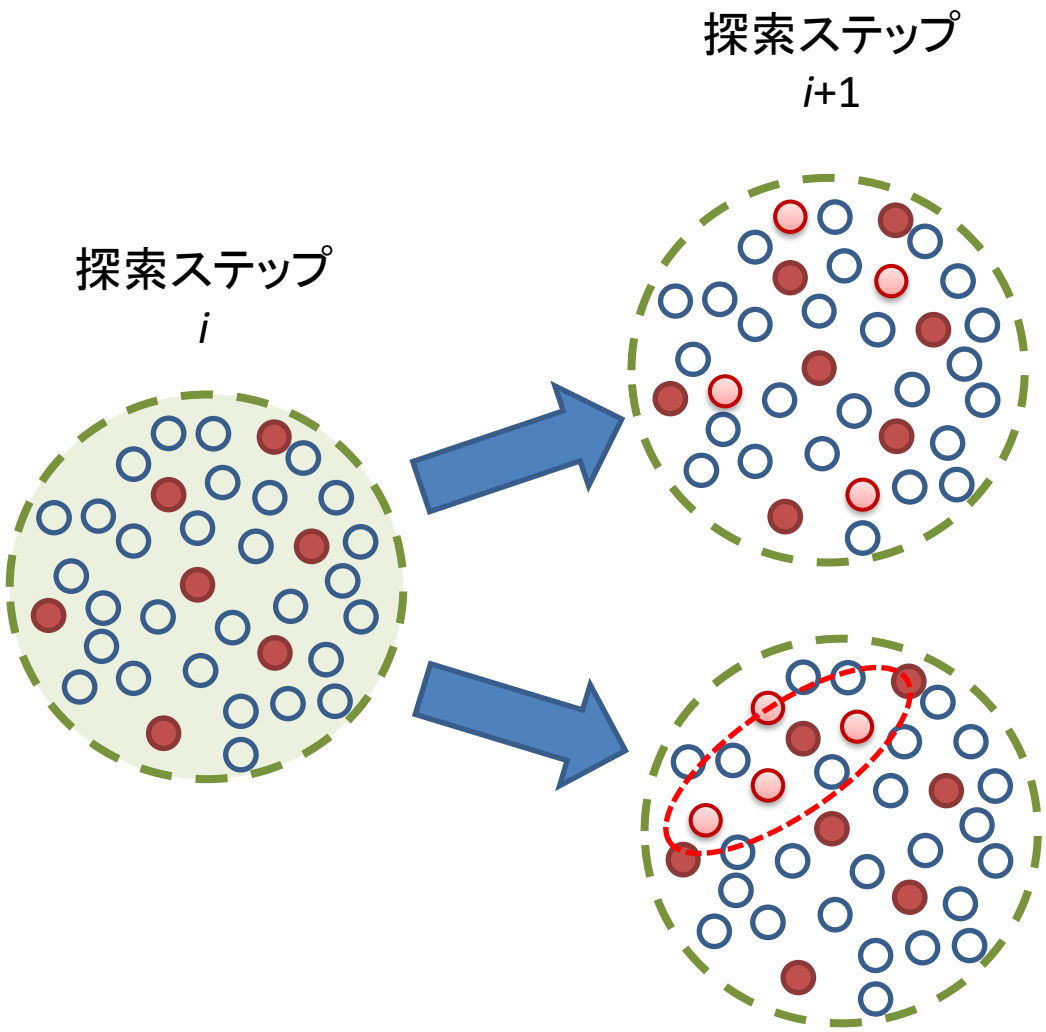








頻度主義とベイズ主義

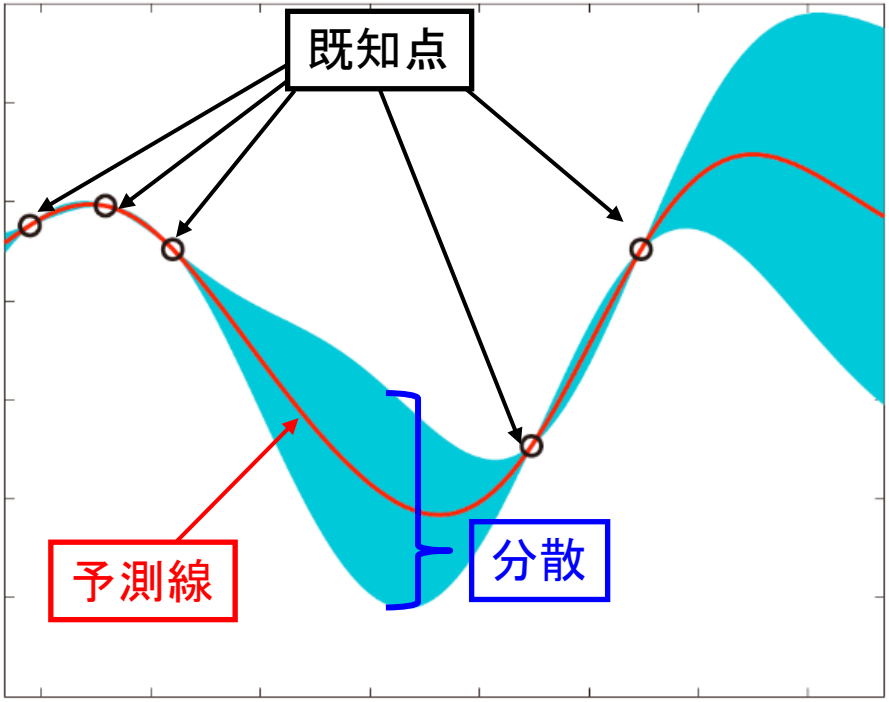


頻度主義:
ランダムサンプリングのみが許される

ベイズ主義:
経験に基づいて確度を考えながら
よりクレバーに次のサンプリングを実施

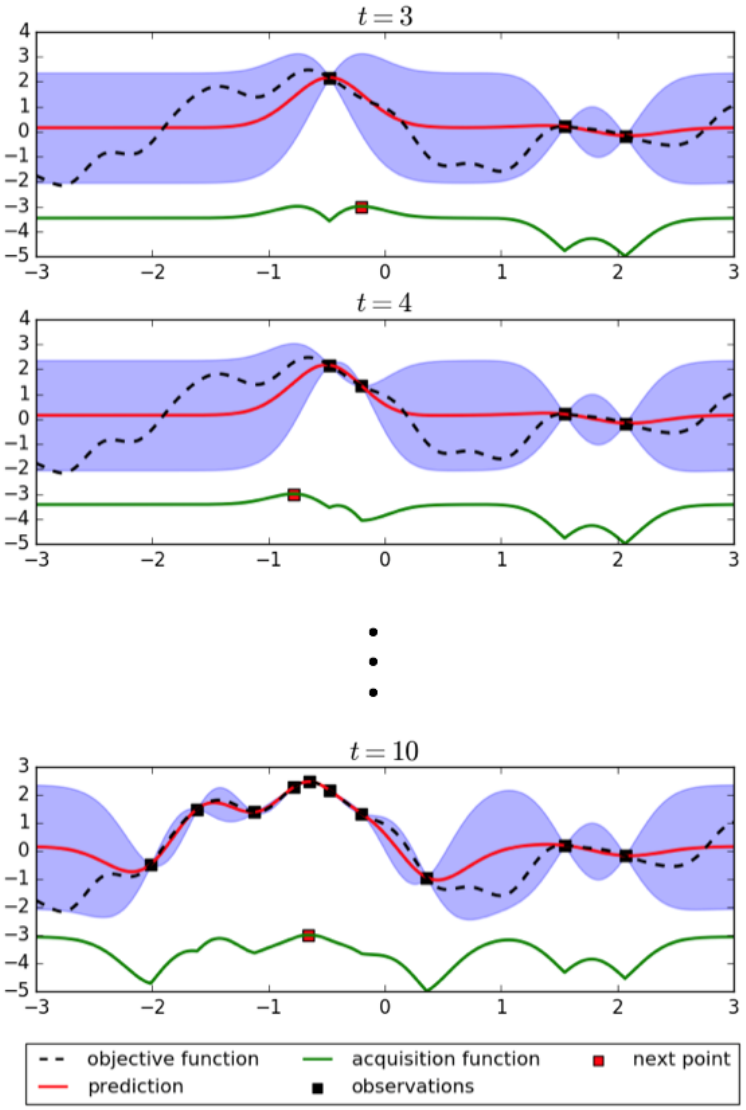
ベイズ最適化（bayesian kriging）

実際には、「もっとも性能がよさそうな場所」を次に調べるのではなく、「UNKOWNな領域（分散が大きい領域）」を優先して調べる。



ホッピング障壁

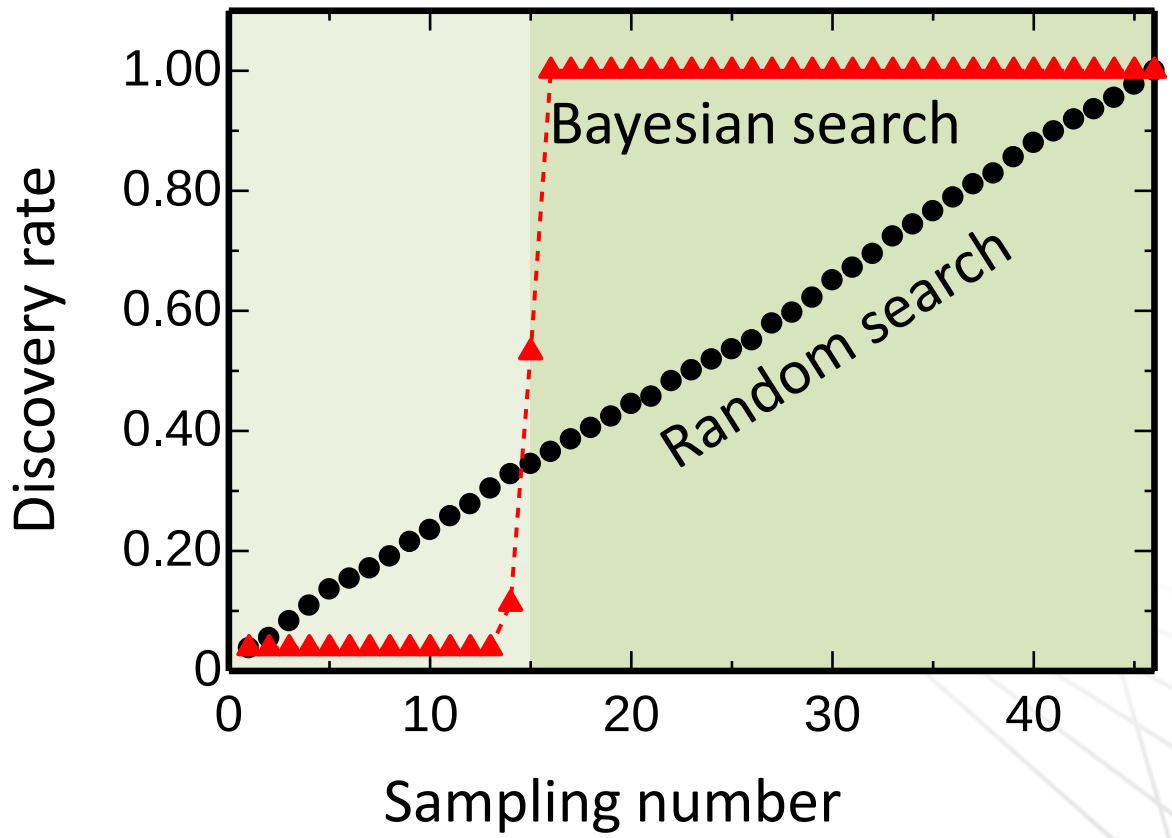
探索材料空間



「ベイズ最適化」による材料探索

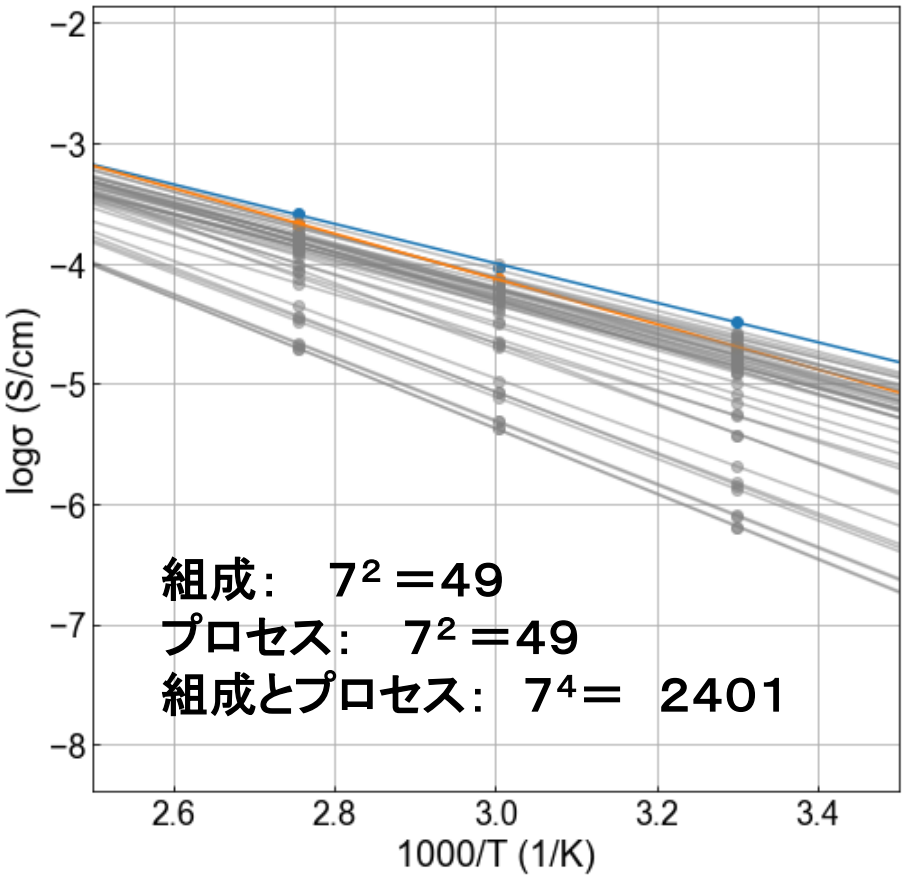
探索範囲が広ければ広いほど効率はアップする。

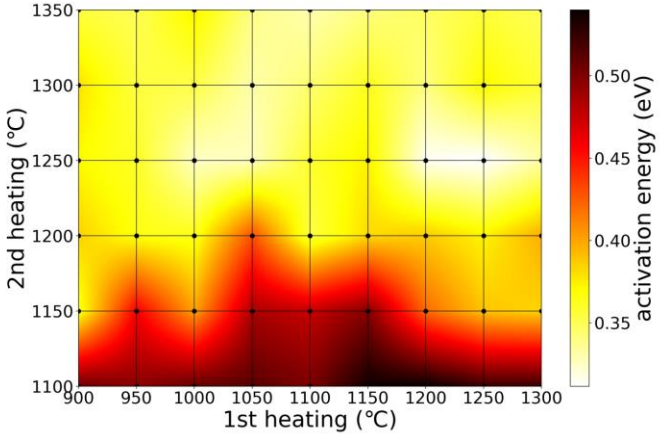
発見確率（1000回平均）

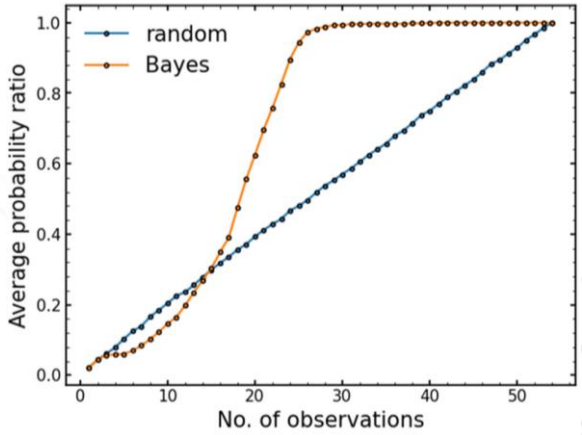


プロセス依存性（焼成温度　仮焼・本焼）

同一組成にあっても、焼成条件が異なれば抵抗は2桁変化することもある。

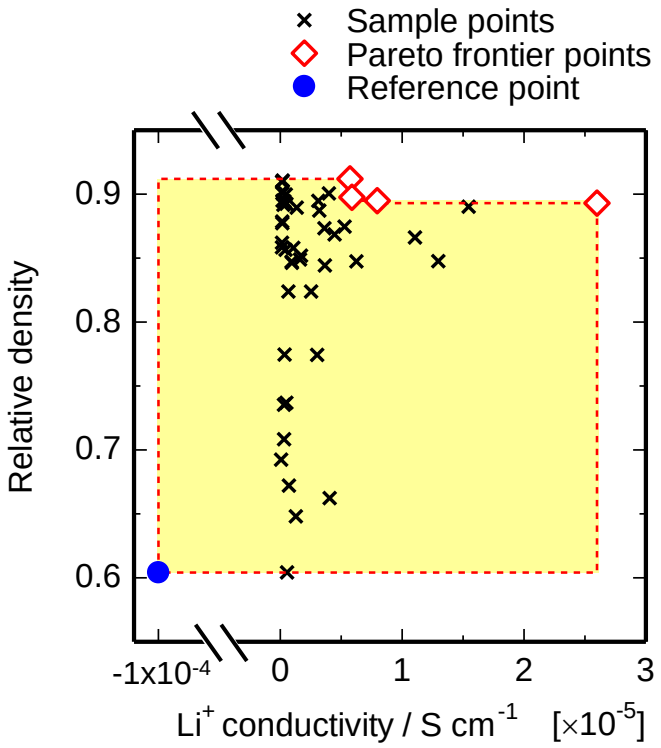


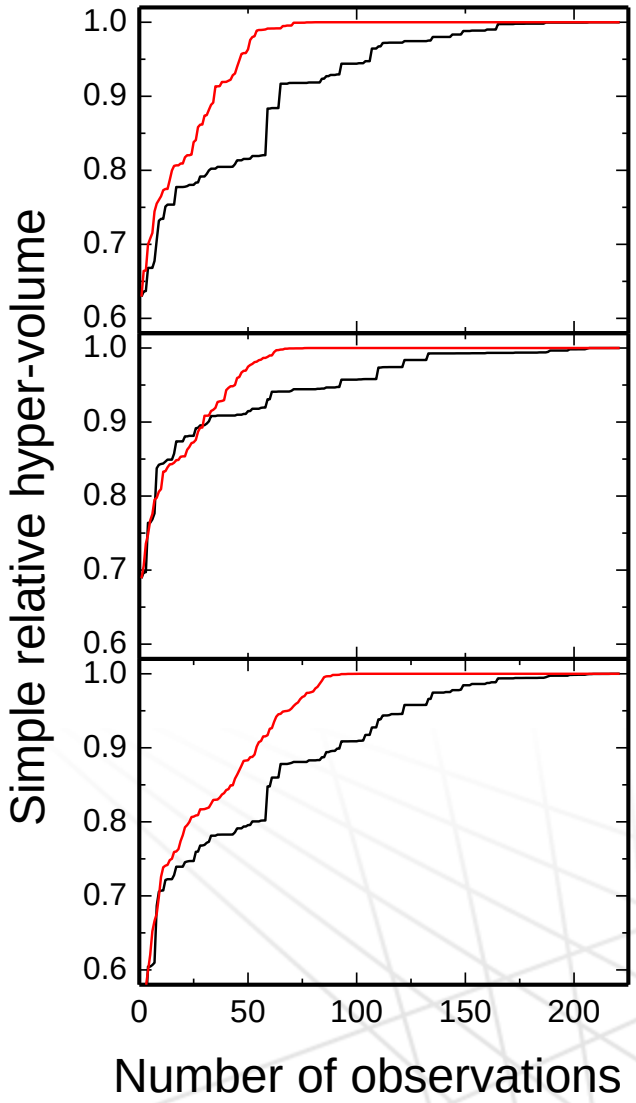




H. Takeda et al., Mater. Adv. (2022)

多目的最適化


Sample points
Pareto frontier points
Reference point
Relative density
0.9
0.8
0.7
0.6
-1x10-4
0
1
2
3
Li+ conductivity / S cm-1 [×10-5]


Simple relative hyper-volume
1.0
0.9
0.8
0.7
0.6
0
50
100
150
200
Number of observations

Transfer learning: reuse accumulated knowledge

Replacement of Zr by Ca, Y

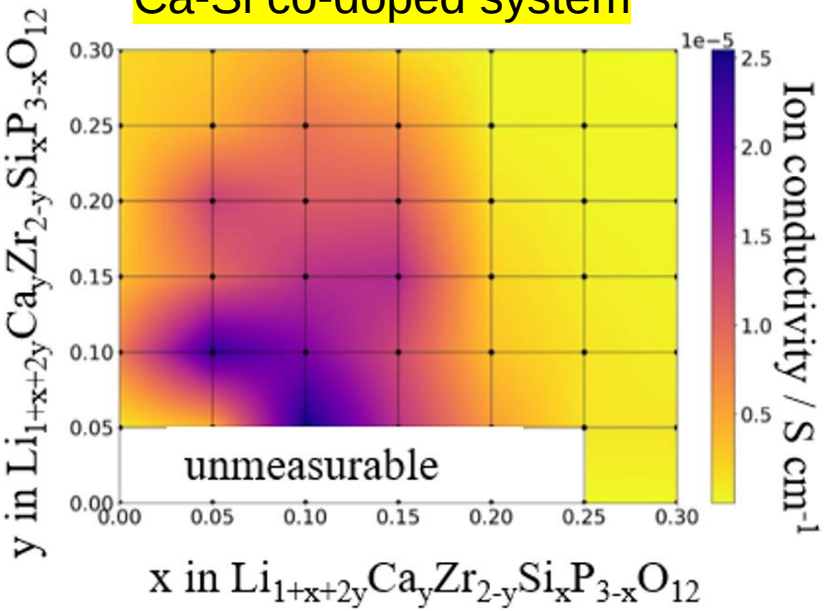
47 compositions

Sharing optimization knowledge

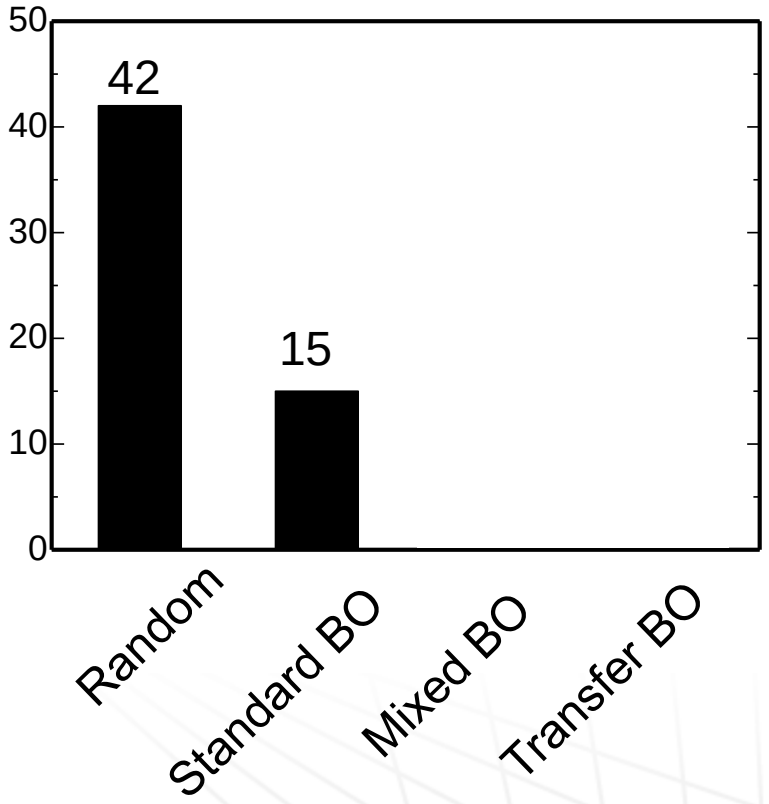
Replacement of Zr by Ca and P by Si

44 compositions

Ca-Si co-doped system

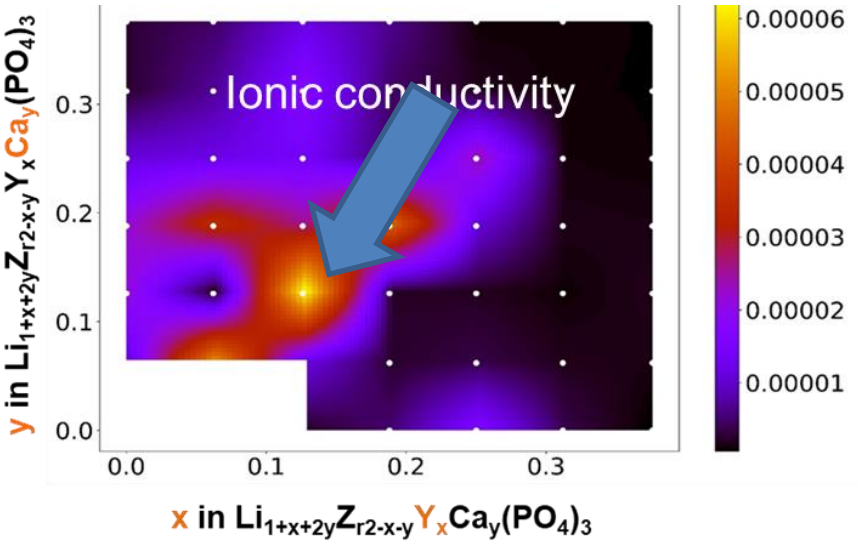


BO steps to discover the best (95%)

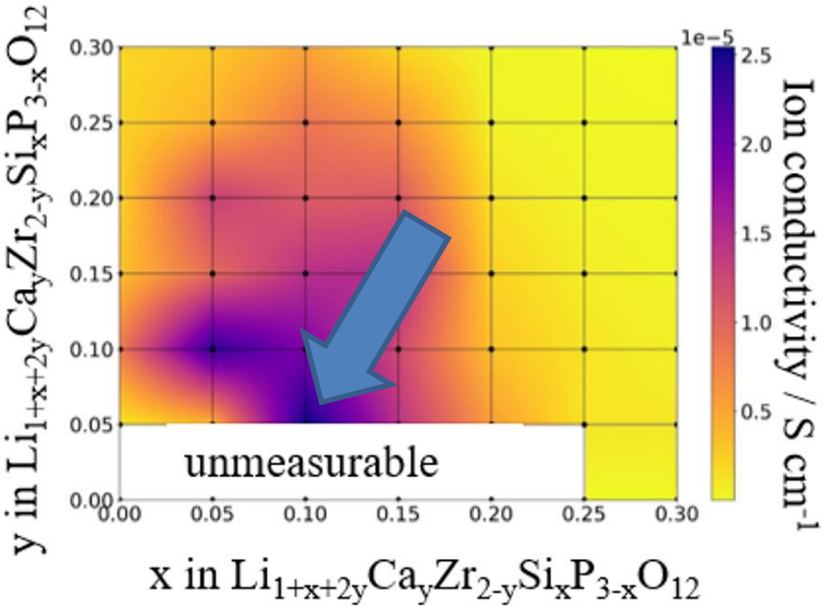


Transfer learning: reuse accumulated knowledge

Ca-Y co-doped system



Ca-Si co-doped system



Knowledge transfer

Reduce trial steps to find optimized composition

転移学習（マルチタスクガウス過程回帰＋ベイズ最適化）

マルチタスクガウス過程回帰

$$k([i,x],[j,x']) = k_{\text{task}}(i,j) \cdot k_{\text{input}}(x,x')$$

タスク間の類似度　入力の類似度

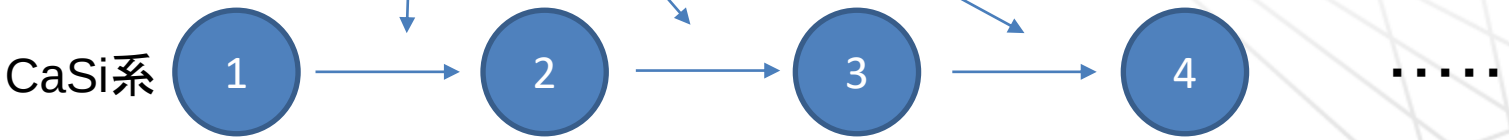
・タスク間の共分散行列を用いることで複数のタスク間の情報の転移を行うモデル
・本研究では intrinsic coregionalization model*と呼ばれるタスク間共分散構造を使用
・実装には Botorchを利用

過去の成果

$Li_{1+2x+y}Ca_xY_yZr_{2-x-y}(PO_4)_3$

実験で調べたイオン伝導度 47点

1. 過去のデータと現データを並列に扱う(mixed-bayes)
2. 類似の記述子のみを使う(transfer-bayes)



Random選択　Bayes最適化法による選択

Transfer learning: reuse accumulated knowledge

Replacement of Zr by Ca, Y

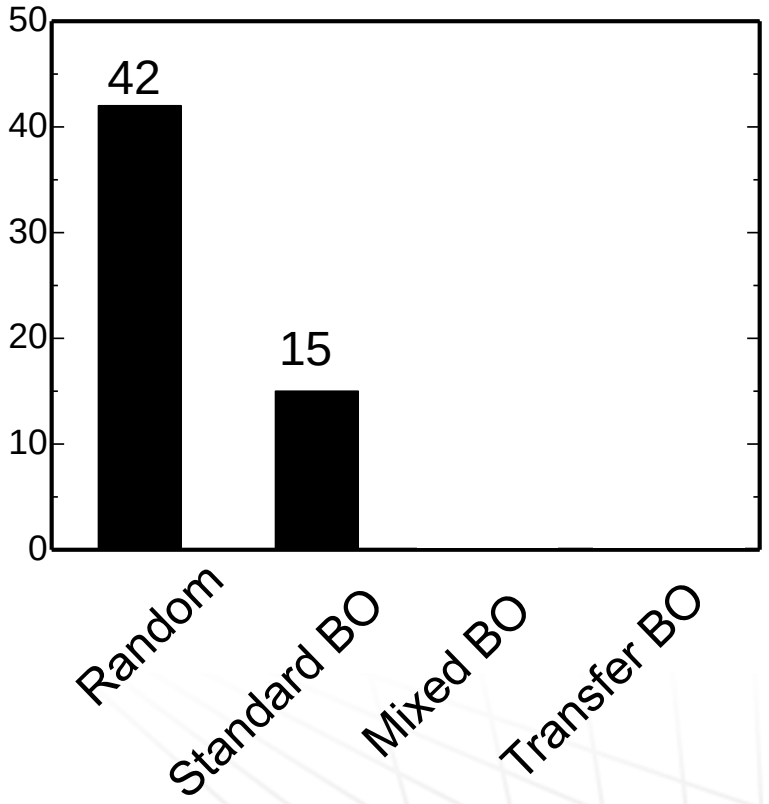
47 compositions

Sharing optimization knowledge

Replacement of Zr by Ca and P by Si

44 compositions

BO steps to discover the best (95%)



Mixed BO:

Just provide 47 Ca/Y data to Ca/Si system

Transfer BO:

Consider similarity of machine learning process and use common part between Ca/Y and Ca/Si systems by using multi-task gaussian process regression.

H. Fukuda et al., RSC advances (2022)

Transfer learning: reuse accumulated knowledge

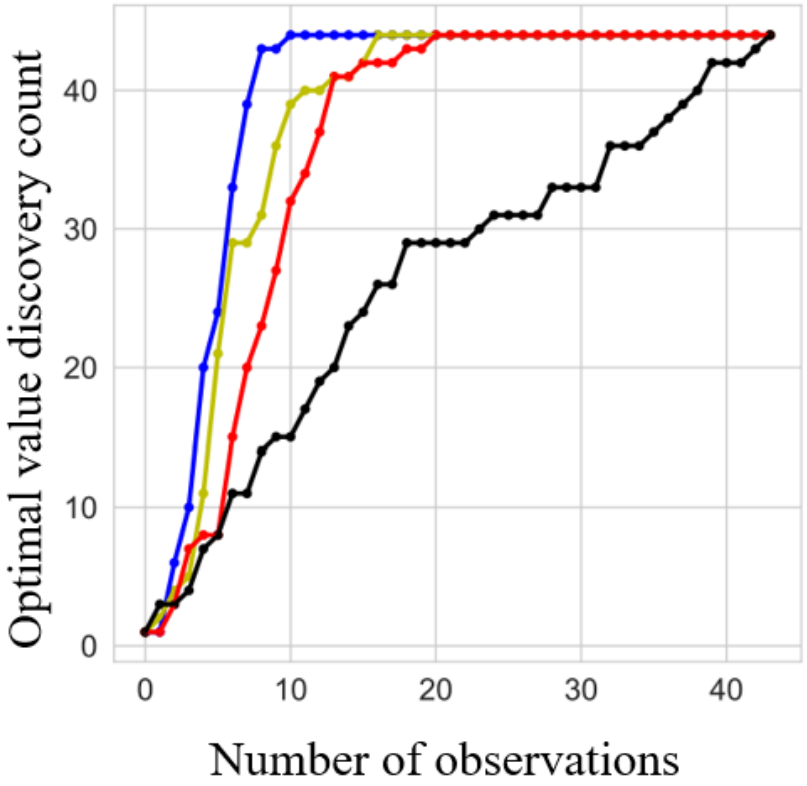
Replacement of Zr by Ca, Y

47 compositions

Sharing optimization knowledge

Replacement of Zr by Ca and P by Si

44 compositions



Mixed BO:

Just provide 47 Ca/Y data to Ca/Si system

Transfer BO:

Consider similarity of machine learning process and use common part between Ca/Y and Ca/Si systems by using multi-task gaussian process regression.

H. Fukuda et al., RSC advances (2022)

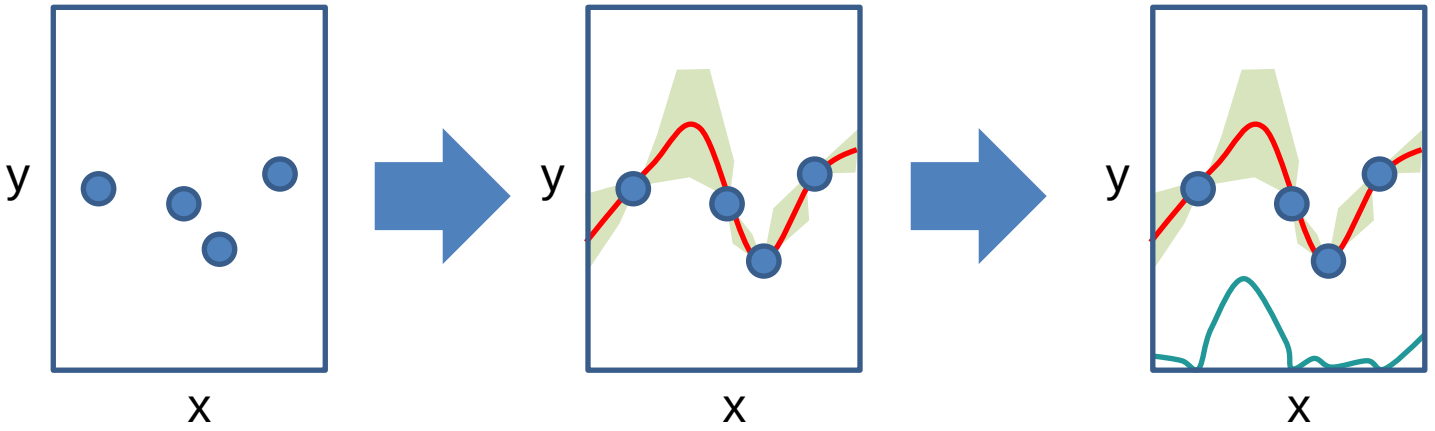
目次

1）ベイズ最適化による固体電解質材料組成の最適化

2）ベイズ最適化 python codingについて

https://github.com/NakayamaLab-NITech/20221124_BayesOptimization-Example-.git

ベイズ最適化（bayesian kriging）の手順



入力（訓練データ）
記述子＋目的変数

ガウス過程回帰分析

予測線と分散領域

獲得関数

PI, EI 戦略など

データの入力1

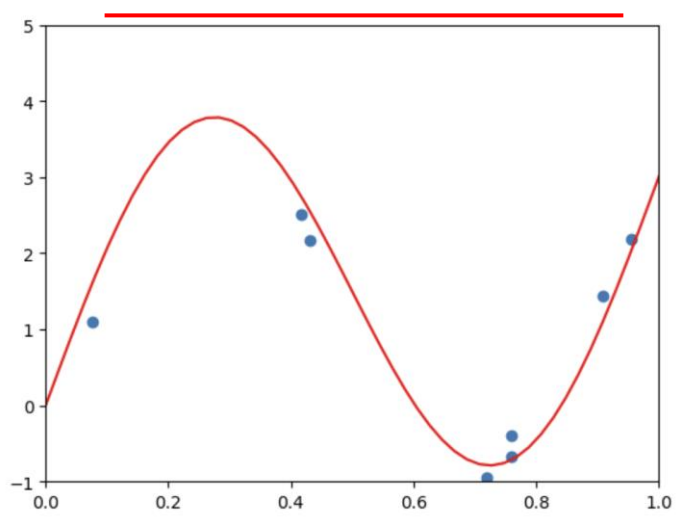
```
import math
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

Numpy 形式

```
# Training data is 100 points in [0,1] inclusive regularly spaced
train_x = np.random.rand(8)
rdm_y=np.random.rand(8)-0.5
# True function is sin(2*pi*x) with Gaussian noise
train_y = 3*np.sin(train_x * (2 * math.pi)) + 3*train_x + rdm_y

true_x=np.linspace(0,2,100)
true_y = 3*np.sin(true_x * (2 * math.pi)) +3*true_x

plt.xlim([0,1])
plt.ylim([-1,5])
plt.plot(true_x,true_y,linestyle='solid',color='red')
plt.scatter(train_x,train_y)

print (train_x,"¥n",train_y)
```

乱数 [0,1]区間に8 ケ生成

$y = 3 \sin(2\pi x) + 3x + \varepsilon$




```
[0.76055624 0.43125453 0.76014872 0.90859528 0.71849764 0.07520469
 0.41750078 0.95449045]
 [-0.66321946  2.16607334 -0.3922612   1.43674083 -0.94807702  1.10158279
  2.50920786  2.18435285]
```

データの入力2

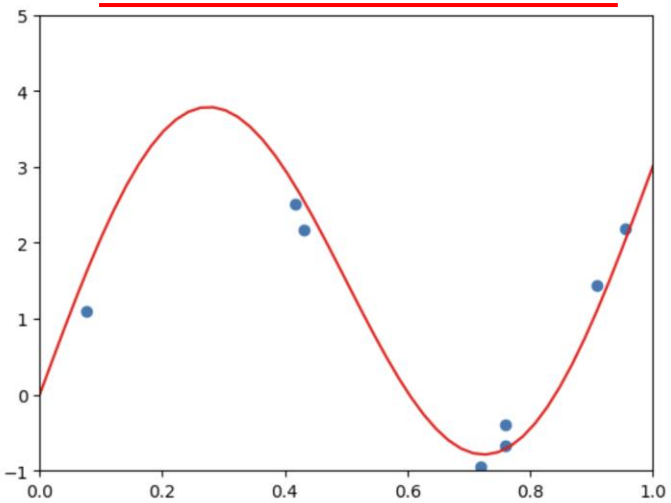
train_x
```
[0.76055624 0.43125453 0.76014872 0.90859528 0.71849764 0.07520469
 0.41750078 0.95449045]
```
train_y
```
[-0.66321946  2.16607334 -0.3922612   1.43674083 -0.94807702  1.10158279
  2.50920786  2.18435285]
```

```
In [11]: train_nx=train_x.reshape(-1,1)
         train_ny=train_y.reshape(-1,1)

         print(train_nx)
         print(train_ny)
         type(train_nx)
```

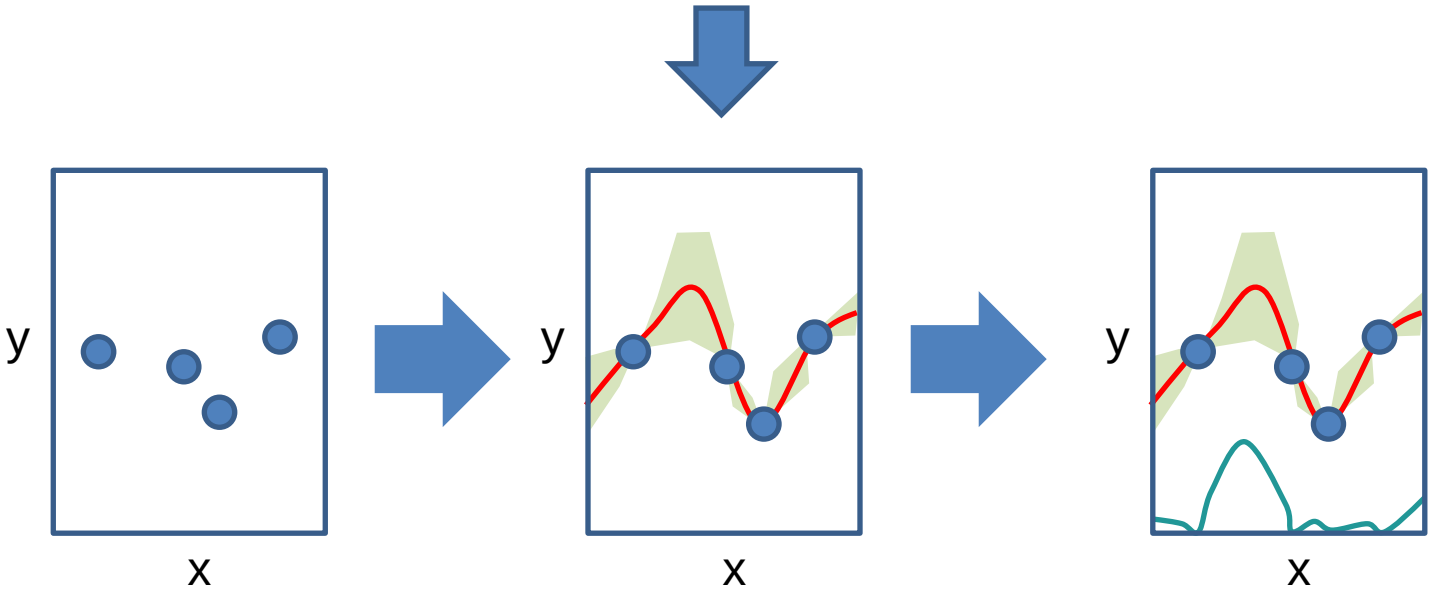
train_nx
```
[[0.35243878]
 [0.6968924 ]
 [0.04598922]
 [0.81188661]
 [0.59208324]
 [0.80183309]
 [0.70652208]
 [0.88731074]]
```
train_ny
```
[[ 3.91037735]
 [-0.72101721]
 [ 1.36775299]
 [-0.15092817]
 [-0.32060918]
 [-0.92732187]
 [-0.63886297]
 [ 0.52296188]]
```

```
Out[11]: numpy.ndarray
```

$y = 3 \sin(2\pi x) + 3x + \varepsilon$



ベイズ最適化（bayesian kriging）の手順



入力（訓練データ）
記述子＋目的変数

ガウス過程回帰分析

予測線と分散領域

獲得関数

PI, EI 戦略など

ガウス過程：sklearn

```
import sklearn.gaussian_process as gp
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process import kernels
from sklearn.preprocessing import StandardScaler

%matplotlib inline
```

```
kk=gp.kernels.ConstantKernel()*gp.kernels.RBF() + gp.kernels.WhiteKernel()
#kk=gp.kernels.RBF(length_scale=2)
#kk=gp.kernels.Matern(nu=5)
#kk=gp.kernels.Matern(nu=5) + gp.kernels.WhiteKernel()

gpm = gp.GaussianProcessRegressor(kernel=kk,alpha=0.1, n_restarts_optimizer=30, normalize_y=True)
gpm.fit(train_nx, train_ny)
print(gpm.log_marginal_likelihood())

#print ("theta0",gpm.kernel.theta)
#print ("theta1",gpm.kernel_.theta)
#print(gpm.kernel.bounds)
```

RBFカーネルの設定

回帰関数の設定→回帰分析→尤度出力

```
-8.336693315581899
```

ガウス過程：sklearn

RBFカーネル

ハイパーパラメーター

$$K(x, x') = \exp(-\gamma||x - x'||^2)$$

y = a * exp (-γ (x-0.35243878)^2)
+ b * exp (-γ (x-0.69689240)^2)
+ c * exp (-γ (x-0.04598922)^2)
+ .....

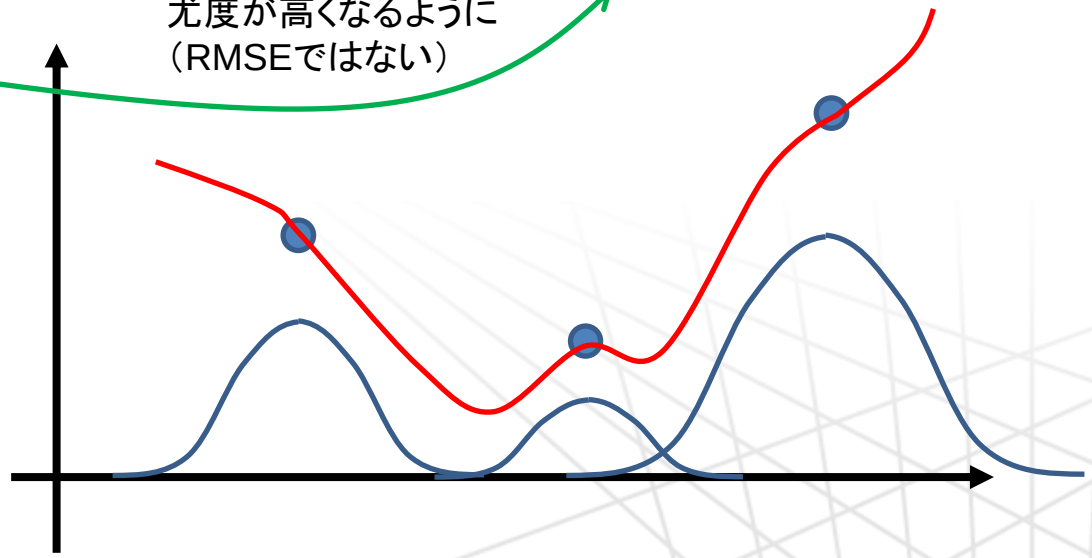
回帰係数

既知のx

尤度が高くなるように
（RMSEではない）

```
In [11]: train_nx=train_x.reshape(-1,1)
         train_ny=train_y.reshape(-1,1)

         print(train_nx)
         print(train_ny)
         type(train_nx)
```

```
[[0.35243878]
 [0.6968924 ]
 [0.04598922]
 [0.81188661]
 [0.59208324]
 [0.80183309]
 [0.70652208]
 [0.88731074]]
[[ 3.91037735]
 [-0.72101721]
 [ 1.36775299]
 [-0.15092817]
 [-0.32060918]
 [-0.92732187]
 [-0.63886297]
 [ 0.52296188]]
Out[11]: numpy.ndarray
```



ガウス過程：sklearn

RBFカーネル

ハイパーパラメーター

$$K(x, x') = \exp(-\gamma||x - x'||^2) \times \theta + \varepsilon$$

```
kk=gp.kernels.ConstantKernel()*gp.kernels.RBF() + gp.kernels.WhiteKernel()
#kk=gp.kernels.RBF(length_scale=2)
#kk=gp.kernels.Matern(nu=5)
```

```
gpm = gp.GaussianProcessRegressor(kernel=kk,alpha=0.1, n_restarts_optimizer=30, normalize_y=True)
gpm.fit(train_nx, train_ny)
```

ハイパーパラメーターを調整する

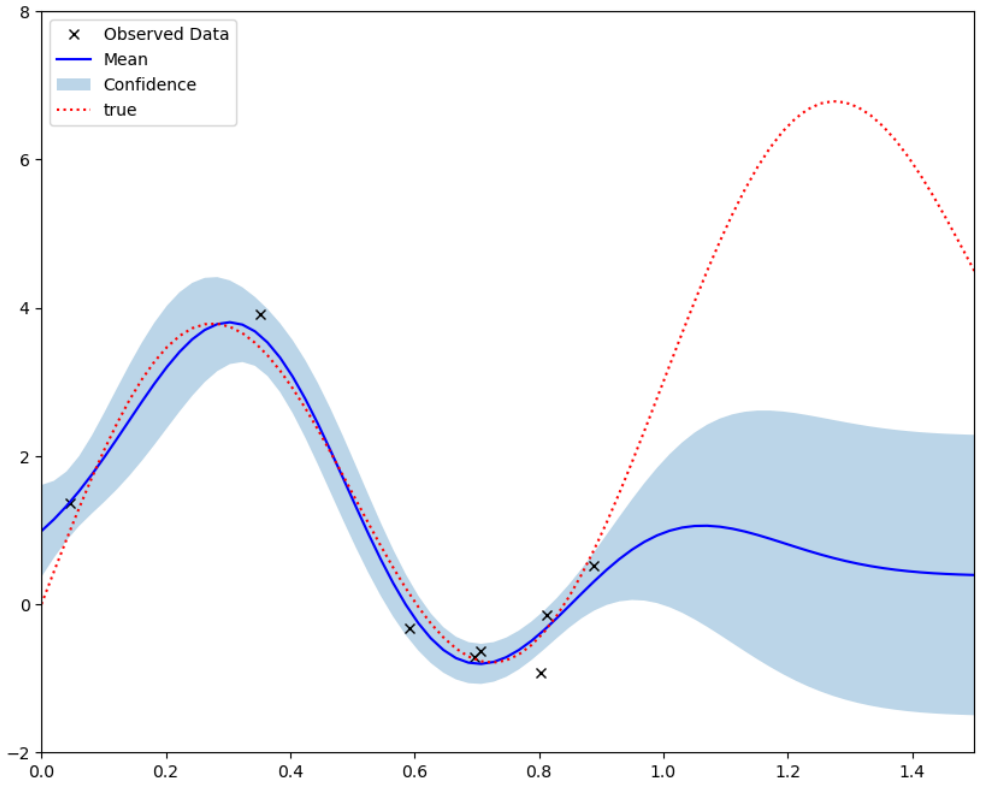
ハイパーパラメーターの確認

```
#print ("theta0",gpm.kernel.theta)
#print ("theta1",gpm.kernel_.theta)
```

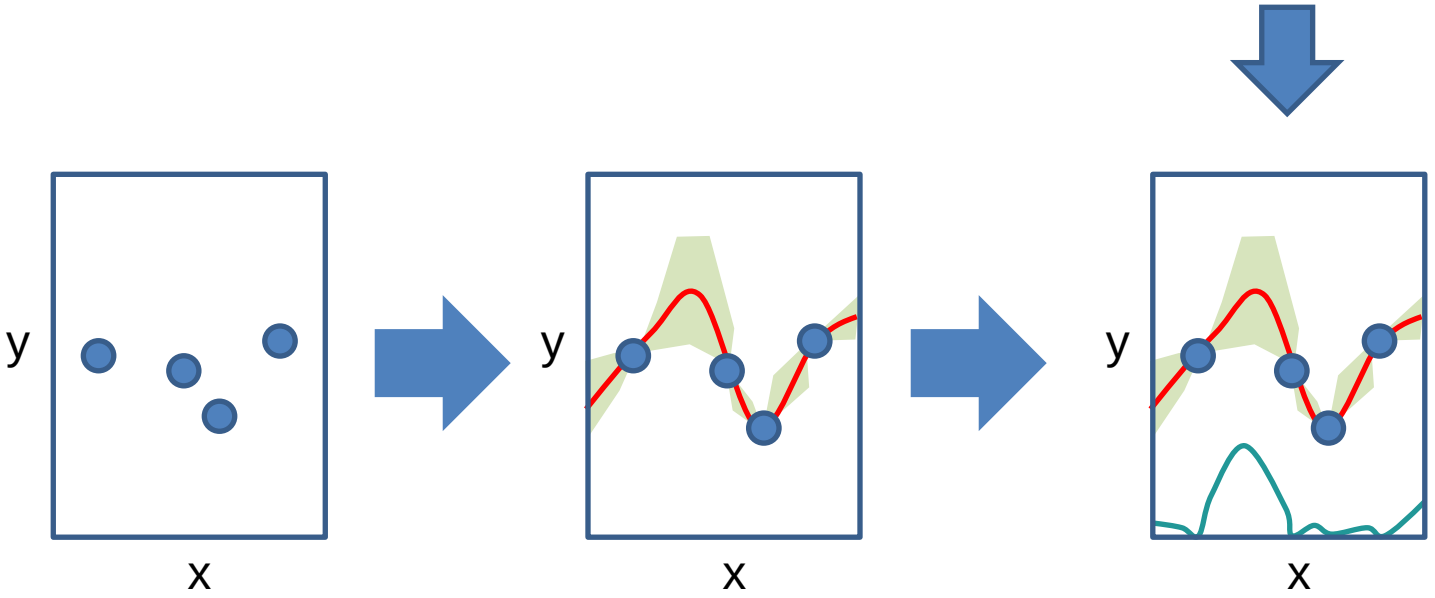
チューニング前
チューニング後

ガウス過程：sklearn　予測式と分散の計算

```
newx = np.linspace(0, 2, 100).reshape(100, -1)
GPf, GPv = gpm.predict(newx, return_cov=True)   # 予測値と分散共分散行列
GPsd = np.sqrt(np.diag(GPv)).reshape(-1, 1)                    # 予測値の標準偏差
#GPsd = np.sqrt(GPv)                          # 予測値の標準偏差
```



ベイズ最適化（bayesian kriging）の手順



入力（訓練データ）
記述子＋目的変数

ガウス過程回帰分析

予測線と分散領域

獲得関数

PI, EI 戦略など

獲得関数:EI戦略

EI戦略　Expected Improvement (期待改善量)

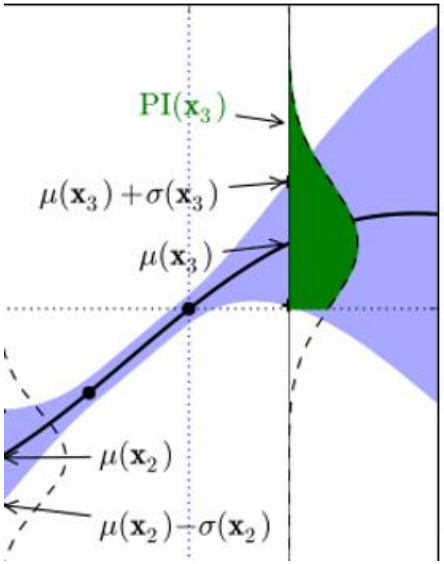
現在まで に観測したデータの最大値 $f(x_{best})$の更新幅の期待値を 関数にとる。

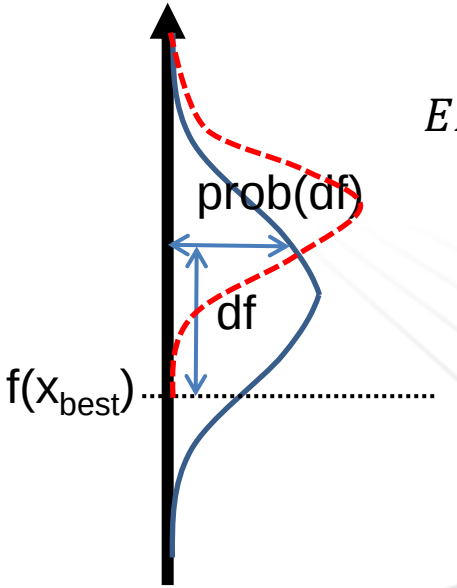
PI戦略:これまでの最大値を(わずかでもいいから)超える確率が最大

↓

EI戦略:更新値の量も考える （予測値は小さいが、分散が大きい箇所も対象となる）

探索と搾取の両立ができるアルゴリズム

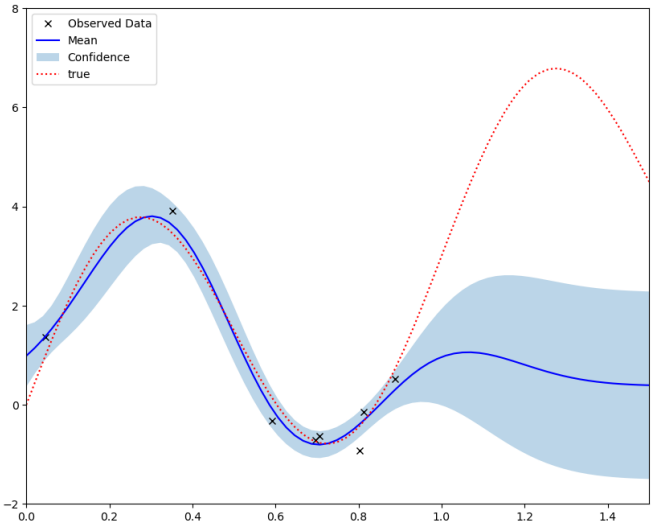


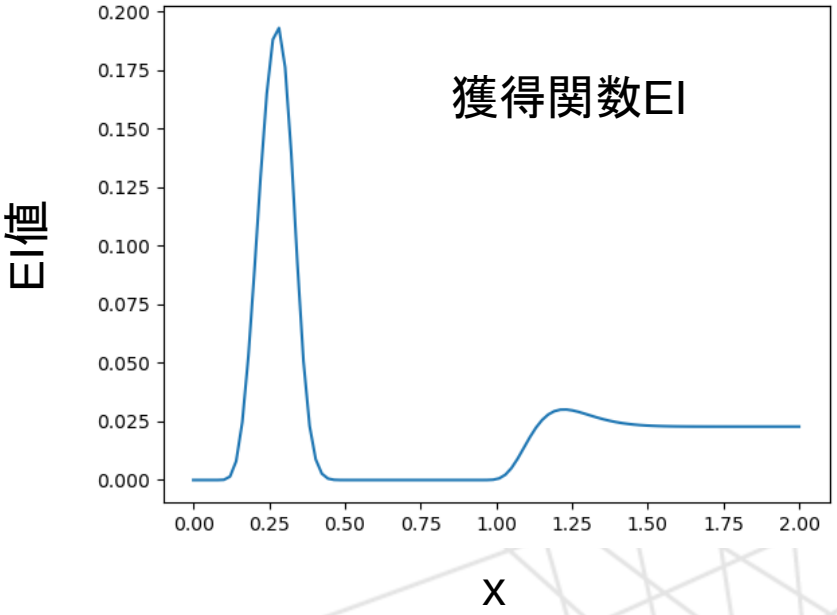


$$EI = \frac{\sum_{df=0}^{\infty}(df \times prob(df))}{\sum_{df=0}^{\infty} prob(df)}$$

獲得関数:コーディング

In [26]:
```
from scipy.stats import norm

current_ymax=np.max(train_ny)
Z = (GPf-current_ymax)/GPsd
ei = ((GPf-current_ymax)*norm.cdf(Z))+(GPsd*norm.pdf(Z))

plt.plot(newx,ei)
```





まとめ

ベイズ最適化

ガウス過程回帰による予測と不確かさ評価
搾取と探索（EI戦略）による効率的な正解探し
コーディングの勉強

謝辞

竹内先生、烏山先生、小林先生
研究室：　スタッフ・学生一同
国プロ・共同研究企業　関係者様

名工大・中山研