論文紹介

192x123x 呉雨楓

2020 年 11 月 10 日

出典

"High Dimensional Bayesian Optimization Using Dropout"
Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, Alistair Shilton
Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

Abstract

ベイズ最適化を高次元にスケーリングすることは、高次元の獲得関数の大域的最適化に費用がかかり、実行不可能な場合が多いため、難易度が高い。既存の方法は、限定された「アクティブ」変数または目的関数の加法形式のいずれかに依存します。ドロップアウト戦略を使用して、各反復で変数のサブセットのみを最適化して、高次元ベイズ最適化の新しい方法を提案します。後悔の理論的限界を導き出し、それがアルゴリズムの導出にどのように情報を与えることができるかを示します。2 つのベンチマーク関数と 2 つの実際のアプリケーション（カスケード分類器のトレーニングと合金組成の最適化）での最適化のためのアルゴリズムの有効性を示します。

1 Introdution

混合製品（シャンプー、合金など）の混合プロセス（合金の熱処理など）は、目標製品を達成するための制御変数の最適値を見つける必要性がほとんどの工業プロセスの中心にあります。制御変数とターゲットの間の数学的関係がわからないかつこの関数はブラックボックス関数なので、処理するのは複雑です。こういう最適化実験のプロセスは複雑的、コストも高いです。

実験的最適化のプロセスは、制御変数の数が増えるとすぐに限界に達します。たとえば、青銅器時代以来、合金を作るために組み合わされた元素は 12 個未満です。しかし、周期表には 97 の自然元素が含まれています。複雑度が大きいため、ターゲットスペースのごく一部しか調査されていません。元素あたり 3 つの混合レベルを持つ高張力合金を見つけるために元素の数をわずか 15 に増やすことにより、検索スペースは 1,400 万を超える選択肢にエスカレートします [Xue et al.,2016]。別の実例となる問題は、高速民間輸送（HSCT）航空機の翼構成設計に、目標の翼構成に到達するために最大 26 個の変数が含まれる可能性があることです [Koch et al.,1999]。

ベイズ的最適化（BO）[Snoek et al.,2012; Nguyen et al.,2016] は、高価なブラックボックス関数を最適化するための強力な手法です。古典的な BO は、ガウス過程（GP）[Rasmussen and Williams,2005] を使用して、ターゲット関数の平均と分散をモデル化します。この関

数の評価コストが高いので、代理関数（または獲得関数）は GP から構築され、活用（平均が高い場合）と探索（不確実性が高い場合）のトレードオフを行います。次のサンプルは、獲得関数を最大化することによって決定されます。高次元での BO メソッドのスケーリングには、2 つの主要な課題があります。まず、GP が必要とする観測値の数は、入力次元が増加するによって指数関数的に増加します。これは、より高いコストがあって、実際のアプリケーションでは多くの場合実行不可能であることを意味します。二つ、実際に高次元獲得関数の大域的最適化は難しい問題なので、それを実行可能にするために費用がかかる可能性があります [Kandasamy et al.,2015; Rana et al.,2017]。

高次元でのベイズ的最適化のソリューションが提案されています。[Wang et al.,2013] は、高次元空間を低次元の部分空間に投影し、次に低次元部分空間（REMBO）で獲得関数を最適化する方法が提案しました。この仮定は一部の次元（$d_e \ll D$）のみが有効である、多くの場合は制限的です。[Qian et al.,2016] の研究ではすべての次元が有効である場合を研究しました。しかし、それらの多くは、埋め込みギャップを減らすために順次ランダム埋め込みという手法を使用することによって、効果がよくありません。高次元関数がすべての次元で効果が同じである場合、これらの方法は使えない可能性があります。加法分解の仮定は、高次元関数分析のもう 1 つのソリューションです。[Kandasamy et al.,2015] は、目的関数が低次元関数（接点なし）のセットの和であると想定され、BO を低次元空間で実行できる Add-GP-UCB モデルを提案しました。Add-GP-UCB を使用すると、目的関数を特徴空間全体に沿って変化することができます。[Li et al.,2016] では軸に沿った表現を削除することにより、Add-GP-UCB を正規化しました。ただし、実際には、特に分離不可能な関数の場合、関数の分解を事前に知ることが困難です。私たちの研究と最も関連するものは DSA(Dimension Scheduling Application)[Ulmasov et al.,2016] であり、これは PCA による各反復での変数の数を減らします。DSA を使用する場合、2 つの問題があります。（1）変数の選択に PCA を使用することは、データポイントが多数ある場合にのみ有効です。これは特にベイズ的最適化が最初にポイント数が少ない場合には当てはまりません。少数のデータポイントを使用した固有ベクトル推定は、多くの場合不正確であり、誤解を招く可能性があります。（2）DSA は、他の座標を現在の最良値にクランプするだけなので、局所最適値で落ちる可能性があります。

この論文は、目的関数が限定された「アクティブな」特徴に依存するという仮定に依存しないアプローチを提案します。つまり、投影は低次元のサブ空間に行うことができます。（[Djolonga et al.,2013; Wang et al.,2013][Ulmasovet al.,2016]）または加法形式に分解された目的関数 [Kandasamy et al.,2015; Li et al.,2016]。ニューラルネットワークのドロップアウトアルゴリズム [Srivastavaet al.,2014] に動機付けられて、我々は高次元ベイズ的最適化における次元ドロップアウトを調査します。各反復で D 次元（$d_e < D$）から d をランダムに選択し、ベイズ的最適化を通じて選択した次元から変数のみを最適化します。省略された次元から変数を「埋める」ために、代替戦略を検討します。つまり、ランダム値、もしくはこれまでに見つかった関数値からの変数の値、およびこの 2 つの方法の混合です。ドロップアウトアルゴリズムを定式化し、ベンチマーク関数と 2 つの実際のアプリケーションに適用します（トレーニングカスケード分類器の [Viola and Jones,2001] およびアルミニウム合金設計）。それらをベースライン（ランダム検索、標準 BO、REMBO [Wang et al.,2013] および Add-GP-UCB [Kandasamy et al.,2015]）と比較します。実験

結果は、私たちのアルゴリズムの有効性を示しています。理論的に後悔の限界を導き出します。予想どおり、ドロップアウトアルゴリズムのコストは残りの「regret のギャップ」であり、ドロップアウトされた変数を埋めるために策定した戦略を通じて、このギャップをどのように削減できるかについての記録を提供します。主な貢献は次のとおりです：

1. 高次元ベイズ最適化のための新しい可変ドロップアウト法の定式化；
2. ドロップアウトアルゴリズムに regret bound の理論的分析、およびドロップアウトされた変数を入力する方法をガイドするために後悔バウンドの使用；
3. 2 つの合成関数と 2 つの実際のアプリケーション（カスケード分類器のトレーニングと改善された（フェーズ）ユーティリティによるアルミニウム合金の設計）のベースラインとアルゴリズムのデモンストレーションと比較。

2 Formulation

Preliminaries ベイズ的最適化は、入力領域 $\mathcal{X} \subseteq \mathbb{R}^D$ の関数 f を最大化または最小化するために使用されます。これには 2 つの重要なコンポーネントが含まれています：事前分布と獲得関数。ガウス過程（GP）は、事後分布と予測分布の扱いやすさから、事前分布を計算するための一般的な選択肢であり、平均 $m(.)$ と共分散カーネル関数 $k(.,.)$ によって指定されます。観測値のセット $x_{1:t}$ と対応する値 $f(x_{1:t})$ を与えると、$\mathbf{f}$ の有限セットの確率はガウス分布になります。

$$\mathbf{f}(\boldsymbol{x}) \sim \mathcal{N}\left(\mathbf{m}(\boldsymbol{x}), \mathbf{K}\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right) \tag{1}$$

ここで、$\mathbf{K}\left(\boldsymbol{x}, \boldsymbol{x}'\right)_{i,j} = k\left(\boldsymbol{x}_i, \boldsymbol{x}'_j\right)$ は共分散行列です。カーネル k に関して二つ人気な選択としては、二乗指数（SE）カーネルと Mate´ rn カーネル。新し点にたいして、予測分布は式（２）になります。

$$f_{t+1} \mid \mathbf{f}_{1:t} \sim \mathcal{N}\left(\mu_{t+1}\left(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_{1:t}, \mathbf{f}_{1:t}\right), \sigma^2_{t+1}\left(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_{1:t}, \mathbf{f}_{1:t}\right)\right) \tag{2}$$

ここで、$\mathbf{f}_{1:t} = \mathbf{f}(\boldsymbol{x}_{\mathbf{1:t}})$、$\mu_{t+1}(.) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:t}$,$\sigma^2_{t+1}(.) = k\left(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}\right) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}$ そして、$k = \left[k\left(\boldsymbol{x}_{t+1}, \boldsymbol{x}_1\right), \cdots, k\left(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t\right)\right]$。

獲得関数は、予測平均と分散から導出された代理関数であり、次のサンプルポイントを決定します。獲得関数 $a\left(\boldsymbol{x} \mid \{\boldsymbol{x}_{1:t}, f_{1:t}\}\right)$ と次のサンプルポイント $\boldsymbol{x}_{t+1} = \operatorname{argmax}_{x \in \mathcal{X}} a(\boldsymbol{x} \mid \{\boldsymbol{x}_{1:t}, f_{1:t}\})$ を示します。 獲得関数の例としては、期待される改善（EI）や GP-UCB などがあります [Srinivas et al.,2010]。EI ベースの獲得関数は、現在の最大値 $f(\boldsymbol{x}^+)$、または $\mathbf{EI}(\boldsymbol{x}) = \mathbb{E}\left(\max\left\{0, f_{t+1}(\boldsymbol{x}) - f\left(\boldsymbol{x}^+\right)\right\} \mid \boldsymbol{x}_{1:t}, \boldsymbol{f}_{1:t}\right.$ に関して期待される改善を計算することです。閉じた形は [Mockuset al., 1978; Jones et al., 1993] の研究で導出されました。GP-UCB[Srinivas et al.,2010] は、$\mathrm{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sqrt{\beta}\sigma(\boldsymbol{x})$ として定義されます。ここで、β は正のトレードオフです。第 1 項は活用に関して、第 2 項は探索に関しての係数です。DIRECT[Jones et al.,1993] は、獲得関数のグローバル最大値を見つけるためによく使用されます。

制限されたドメイン $\mathcal{X} = [0,1]^D$ で関数 f を最大化しようとします（これは通常スケーリングによって達成できます）。最大関数値はクエリポイント $\boldsymbol{x}^?$ で達成できると仮定します。つまり、$\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ です。反復 t および対応するクエリポイント $\boldsymbol{x} \in \mathcal{X}$ で、瞬間的な後悔 r_t は $r_t = f\left(x^?\right)?f\left(x_t\right)$ と定義され、累積的な後悔 R_T は R_T と定義されます。アルゴリズムの望ましい特性は、後悔しないことです：$\lim_{T\to\infty} \frac{1}{T} R_T = 0$。

Dropout AlgorithmI_d を D 次元以外の d のインデクスとします。I_{D-d} を d を除いた D 次元内のインデクス。つまり、$I_d \cup I_{D-d} = \{1, \cdots, D\}$。$\boldsymbol{x}^{I_d}$ と $\boldsymbol{x}^{I_{D-d}}$ を I_d と I_{D-d} の対応変数にしてます。便宜上、$\boldsymbol{x}^d = \boldsymbol{x}^{I_d}$、$\boldsymbol{x}^{D-d} = \boldsymbol{x}^{I_{D-d}}$ として、さらに $\boldsymbol{x} = [\boldsymbol{x}^d, \boldsymbol{x}^{D-d}]$。

ニューラルネットワークのドロップアウトアルゴリズム [Srivastavaet al.,2014] から得られた経験を基づいて、高次元ベイズ的最適化のための次元ドロップアウトを研究します。各反復で D 次元以外のところで d をランダムに選択し（$d < D$）、ベイズ的最適化によって d 次元変数のみを最適化します。具体的には、d 次元空間で、すべての $\boldsymbol{x}^{D-d}$ について観測値 $y = f\left(\left[\boldsymbol{x}^d, \boldsymbol{x}^{D-d}\right]\right) + \varepsilon$ を仮定します。ここで、$\varepsilon \sim \mathcal{N}(0, \sigma^2)$ とします。次に、ガウス過程を使用して、関数値 $f\left(\left[x^d, x^{D-d}\right]\right), \forall x^{D-d}$ をモデル化します。予測平均 $\mu(\boldsymbol{x}^d)$ と分散 $\sigma(\boldsymbol{x}^d)$ を計算できます。GP-UCB [Srinivas et al.,2010] と同様に、d 次元空間で獲得関数を構築します。

$$a\left(\boldsymbol{x}^d\right) = \mu_{t-1}\left(\boldsymbol{x}^d\right) + \sqrt{\beta_t^d}\sigma_{t-1}\left(\boldsymbol{x}^d\right) \tag{3}$$

ここで、β_t^d は探索 $\sigma_{t-1}(\boldsymbol{x}^d)$ と $\mu_{t-1}(\boldsymbol{x}^d)$ 活用を調整するトレードオフです。各反復にたいして、式（3）を最大化して新しい $\boldsymbol{x}^d$ を計算します。

$\boldsymbol{x}_t^d$ が既知として、残っている $D-d$ 次元で変数 $\boldsymbol{x}_t^{D-d}$ を埋める必要があります。我々は $\boldsymbol{x}_t^{D-d}$ にたいして三つの埋める戦略を導出します：

1. ランダムドロップアウト：ドメイン内のランダム値を使用：

$$\boldsymbol{x}_t^{D-d} \sim \mu(boldsymbolx^{D-d}) \tag{4}$$

   ここで、$\mu(.)$ は一様分布。

2. ドロップアウトコピー：一番効果がいい関数から変数の値をコピーする：

$$\boldsymbol{x}_t^+ = \text{argmax}_{t'\le t} f\left(\boldsymbol{x}_{t'}\right) \tag{5}$$

$$\boldsymbol{x}_t^{D-d} = \left(\boldsymbol{x}_t^+\right)^{D-d} \tag{6}$$

   ここで、

3. ドロップアウトミックス：上記の 2 つの方法を組み合わせて使用します。 確率 p のランダム値を使用するか、確率 $1?p$ の条件でこれまでに見つかった関数値の変数から値をコピーします。

次元数が膨大な場合、ドロップアウトランダムの効果が僅か。ただし、この方法は反復ごとに変数 d を最適化するため、最適化を改善することができます。これまでの最良の関数値から変数の値をコピーすることは、以前の最良の regret を一貫して改善するための効率

的な戦略のようです。ただし、この方法は局所的最適でスタックする可能性があります。3 番目の戦略ではこの問題を解決することができます。

私たちのアプローチは、d 次元空間でベイズ最適化を実行するため、DIRECT では、ζ 精度を達成するために取得関数への O（ζ -d）呼び出しが必要です [Jones et al。、1998]。これは、DIRECT が O（ζ -D）目的関数呼び出しを必要とする全次元 BO よりも大幅に優れています。 私たちのアプローチと全次元 BO の両方で、共分散行列の逆行列を計算するために時間計算量 O（n3）が必要です。ここで n は観測数です。 アルゴリズムを以下に要約します。

Algorithm 1 Dropout Algorithm for High-dimensional Bayesian Optimization

Input: $\mathcal{D}_1 = \{\boldsymbol{x}_0, y_0\}$

1: **for** $t = 1, 2, \ldots$ **do**
2: randomly select d dimensions
3: $\boldsymbol{x}_t^d \leftarrow \text{argmax}_{\boldsymbol{x}_t^d \in \mathcal{X}^d} a\left(\boldsymbol{x}^d \mid \mathcal{D}_t\right)$
4: $\boldsymbol{x}_t^{D-d} \leftarrow$ one of three "fill-in" strategies
5: $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t^d \cup \boldsymbol{x}_t^{D-d}$
6: $\boldsymbol{y}_t \leftarrow$ query $\boldsymbol{y}_t$ at $\boldsymbol{x}_t$
7: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \boldsymbol{x}_t, \boldsymbol{y}_t$
8: **end for**

3 Theoretical Analysis

私たちの主な貢献は、アルゴリズムに対する regret bound を導き出し、ヒューリスティック戦略について議論することです。$\boldsymbol{x}_d$ が与えられた場合の最悪の関数値として $f(\boldsymbol{x}_d)$ を示します。例として、$f\left(\boldsymbol{x}^d\right) = f\left(\left[\boldsymbol{x}^d, \boldsymbol{x}_w^{D-d}\right]\right)$。ここで、$\boldsymbol{x}_w^{D-d} = \text{argmin}_{\boldsymbol{x}^{D-d}} f\left(\left[\boldsymbol{x}^d, \boldsymbol{x}^{D-d}\right]\right)$ です。

Assumption 1 あるカーネル $k(\boldsymbol{x}, \boldsymbol{x}^{'})$ を使用して GP から f をサンプリングする。つまり、これはすべての x の L-リプシッツです。次に、f の偏導関数は、定数 a、$b > 0$ の条件で高い確率限界を満たします。

$$\mathcal{P}\left(\forall j, \frac{\partial f}{\partial x_j} < L\right) \geq 1 - ae^{-(L/b)^2}, \forall t \geq 1 \tag{7}$$

この仮定は、次の方程式がすべての x について 1?$\sigma/2$ より大きい確率で成り立つことを意味します。

$$\begin{aligned} \left|f(\boldsymbol{x}) - f\left(\boldsymbol{x}^d\right)\right| &= \left|f\left(\left[\boldsymbol{x}^d, \boldsymbol{x}^{D-d}\right]\right) - f\left(\left[\boldsymbol{x}^d, \boldsymbol{x}_w^{D \cdot d}\right]\right)\right| \\ &\leq L\left\|\boldsymbol{x}^{D-d} - \boldsymbol{x}_w^{D-d}\right\|_1 \leq L|D-d| \end{aligned} \tag{8}$$

ここで、$L = b\sqrt{\log(2(D-d)a/\delta)}$ です。

Lemma 1 $\delta \in (0,1)$ から取って、$\beta_t^d = 2\log\left(4\pi_t/\delta\right) + 2d\log\left(dt^2br\sqrt{\log(4da/\delta)}\right)$ を設置します。ここで、$\Sigma_{t\geqslant 1}\pi_t^{-1} = 1, \pi_t > 0$。そして、$d$ 次元空間で確率を $\geqslant 1-\delta/2$ として、

$$\left|f\left(\boldsymbol{x}^d\right) - \mu_{t-1}\left(\boldsymbol{x}^d\right)\right| \leq \sqrt{\beta_t^d}\sigma_{t-1}\left(\boldsymbol{x}^d\right), \forall t \geq 1 \tag{9}$$

補題 *2* は、d 次元空間でのベイズ的最適化から導出されます。証明は *[Srinivaset al.,2010]* の定理 *2* と同じです。

Lemma 2 β_t^d を補題 2 のような定義にします。さらに、$\sigma'_{t-1}\left(\boldsymbol{x}^d\right) = \sigma_{t-1}\left(\boldsymbol{x}^d\right) + \frac{L(D-d)}{\sqrt{\beta_t^d}}$ を設定します。そして、次の式を得ることができます。

$$\left|f(\boldsymbol{x}) - \mu_{t-1}\left(\boldsymbol{x}^d\right)\right| \leq \sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}^d\right) \tag{10}$$

確率が $\geqslant 1-\delta$ です。

証明。次の式はすべての $t \geqslant 1$ と $\boldsymbol{x} \in \mathcal{X}$ の確率が $\geqslant 1-\delta$ の条件で当てはまります。

$$\begin{aligned}
& \left|f(\boldsymbol{x}) - \mu_{t-1}\left(\boldsymbol{x}^d\right)\right| \\
\leq\ & \left|f(\boldsymbol{x}) - f\left(\boldsymbol{x}^d\right)\right| + \left|f\left(\boldsymbol{x}^d\right) - \mu_{t-1}\left(\boldsymbol{x}^d\right)\right| \\
\leq\ & L\left\|\boldsymbol{x} - \left[\boldsymbol{x}^d, \boldsymbol{x}_w^{D-d}\right]\right\|_1 + \sqrt{\beta_t^d}\sigma_{t-1}\left(\boldsymbol{x}^d\right) \qquad (11) \\
\leq\ & L(D-d) + \sqrt{\beta_t^d}\sigma_{t-1}\left(\boldsymbol{x}^d\right) \\
\leq\ & \sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}^d\right) \qquad (12)
\end{aligned}$$

ここで、$\sigma'_{t-1}\left(\boldsymbol{x}^d\right) = \sigma_{t-1}\left(\boldsymbol{x}^d\right) + \frac{L(D-d)}{\sqrt{\beta_t^d}}$。式（10）の 3 行目から 4 行目は、式（7）を利用しています。β_t^d が増加しているため、分散差 $\frac{L(D-d)}{\sqrt{\beta_t^d}}$ は反復 t とともに減少します。

Lemma 3 $\delta \in (0,1)$ を選択して、β_t^d を補題 2 のように定義します。そして、$regretr_t$ の上限は $\sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}^d\right) + \frac{1}{t^2}$ です。

証明。先ずは $\boldsymbol{x}_t^d$ の定義によって、$\mu_{t-1}\left(\boldsymbol{x}_t^d\right) + \sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}^d\right) \geqslant \mu_{t-1}\left([\boldsymbol{x}^*]_t^d\right)$ $+ \sqrt{\beta_t^d}\sigma'_{t-1}\left([\boldsymbol{x}^*]^d\right)$ が得ることができます。[Srinivas et al., 2010] の補題 5.7 によって、$\mu_{t-1}\left([\boldsymbol{x}^*]_t^d\right) + \sqrt{\beta_t^d}\sigma'_{t-1}\left([\boldsymbol{x}^*]^d\right) + \frac{1}{t^2} \geqslant f(x^*)$ が得ることができます。ここの $[x^*]_t$ では、離散化 $D_t \subset \mathcal{X}$ から $\boldsymbol{x^*}$ への最も近い点を示します。そして、

$$\begin{aligned}
r_t =\ & f(\boldsymbol{x^*}) - f(\boldsymbol{x}_t) \\
\leq\ & \sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}_t^d\right) + \mu_{t-1}\left(\boldsymbol{x}_t^d\right) + \frac{1}{t^2} - f\left(\left[\boldsymbol{x}_t^d, \boldsymbol{x}_t^{D-d}\right]\right) \\
\leq\ & 2\sqrt{\beta_t^d}\sigma'_{t-1}\left(\boldsymbol{x}^d\right) + \frac{1}{t^2}
\end{aligned}$$

Lemma 4 $\delta \in (0,1)$ を選択して、β_t^d を補題 2 のように定義します。その後、累積的な *regret* は確率 $\geqslant 1-\delta$ があります。そして $C_1 = 8/log(1+\sigma^{-2})$、

$$R_T \leq \sqrt{C_1 \beta_T^d \gamma_T T} + 2TL(D-d) + 2 \tag{13}$$

証明。以下のように補題 5 を証明することができます。

$$R_T = \sum_{t \leq T} r_t \leq \sum_{t \leq T} \left(2\sqrt{\beta_t^d} \sigma'_{t-1}\left(\boldsymbol{x}^d\right) + \frac{1}{t^2} \right)$$

$$= \sum_{t \leq T} \left(2\sqrt{\beta_t^d} \sigma_{t-1}\left(\boldsymbol{x}^d\right) \right) + 2TL(D-d) + \frac{\pi^2}{6} \tag{14}$$

$$\leq \sqrt{C_1 \beta_T^d \gamma_T T} + 2TL(D-d) + 2 \tag{15}$$

ここで、式（１２）の二行目の $\sum_{t \leq T} \left(2\sqrt{\beta_t^d} \sigma'_{t-1}\left(\boldsymbol{x}^d\right) \right)$ の上限は [Srinivas et al., 2010] の定理 1 で計算することができます。γ_T では違うカーネルで上限を計算することができます。SE カーネルの場合、$\gamma_T = \mathcal{O}\left((\log T)^{d+1}\right)$。

3.1 Discussion

補題 5 は、$\lim_{T\to\infty} \frac{1}{T} R_T \leq 2L(D-d)$ を示します。つまり、regret のギャップが限界にとどまります。これは、$\left\| \boldsymbol{x} - \left[\boldsymbol{x}^d, \boldsymbol{x}^{D-d}\right] \right\|_1, \forall \boldsymbol{x}^{D-d}$（式（7））の最悪の場合の限界を通して導入されます。実際には、$D?d$ 次元の正しい選択は、この差を減らすことによってこの限界を改善します。したがって、ドロップアウトされたディメンションの変数を入力するための 3 つのオプションを作成しました。ランダム、最良の値、および 2 つの混合です。直感的に、現在の最適値がグローバルな最適値から遠く離れている場合、他の情報がないため、ランダムな値が「フィルイン」の適切な推測になります。現在の最適値がグローバル最適値に近い場合、ドロップアウトされた変数の値を、見つかった最良の関数値からコピーすると、前の反復で得られた最良の regret が改善される可能性があります。これは、他の座標を固定したまま、選択した座標ブロックを最適化するブロック座標降下法 [Nesterov,2012] のように動作します。違いところは、ブロック座標降下は、前の反復が最良の値に到達したことを前提としているのに対し、Dropout-Copy は前の反復の最良から開始することです。現在の最適値が局所最適値に近い場合、Dropout-Copy がスタックする可能性があります。この局所最適を回避するために、Dropout-Mix を使用して、事前に指定された確率でランダムなフィルインを Dropout-Copy に導入します。

4 Experiments

ベンチマーク関数と 2 つの実際のアプリケーションで提案手法を評価します。私たちの方法を 4 つのベースラインと比較します : 単純なランダムサンプリング方法であるランダム検

索、標準の BO、高次元を低次元に投影する REMBO[Wang et al.,2013]、および ADDGP-UCB（変数の互いに素のグループを最適化し、それらを結合します。）[Kandasamy et al.,2015]。標準の BO の場合、30 秒のバジェット（アルゴリズムによって返される時間よりも長い）を割り当てて、各反復で獲得関数を最適化します。初期観測数は $d+1$ に設定されています。獲得関数を最適化するために、長さスケール 0.1 および DIRECT [Jones et al。、1993] の SE カーネルを使用します。異なる初期化で各アルゴリズムを 20 回実行し、標準誤差で平均値を報告します。

4.1 Optimization of Benchmark Functions

Dropout-Mix が局所収束を処理できることを示すために、テスト関数としてバイモーダルガウス混合関数を選択します。ガウス混合関数は、$y = \mathcal{NP}\left(\boldsymbol{x}; \mu_1, \Sigma_1\right) + \frac{1}{2}\mathcal{NP}\left(\boldsymbol{x}; \mu_2, \Sigma_2\right)$ として定義されます。ここで、$\mathcal{NP}$ はガウス確率関数、$\mu_1 = [2, 2, \ldots, 2]$、$\mu_2 = [3, 3, \ldots, 3]$ および $\Sigma_1 = \Sigma_2 = diag\left([1, 1, \ldots, 1]\right)$。 定義のドメイン $\mathcal{X} = [1, 4]^D$ およびそのグローバル最大値は $\boldsymbol{x}^? = [2, 2, \ldots, 2]$ にあります。この関数には極大値があり、相互作用する変数はありません。アルゴリズムが相互作用する変数を持つ関数に対して効果的に機能することを示すために、2 番目のテスト関数としてユニモーダル Schwefel の 1.2 関数 $f(x) = \sum_{j=1}^{D}\left(\sum_{i=1}^{j}\boldsymbol{x}_i\right)^2$ を使用します。これは定義域 $\mathcal{X} = [-1, 1]^D$ で定義され、$\boldsymbol{x}^? = [0, 0, \ldots, 0]$ でグローバル最小値を持ちます。任意の反復までに到達した最良の関数値に関してアルゴリズムを比較します。**On the number of chosen dimensions** d 選択した次元数 d がドロップアウトアルゴリズムにどのように影響するかを調査しました。ドロップアウトコピーで $D = 20$ に対して $d = 1, 2, 5, 10$ を実験します。1 つの実験では、すべての反復で同じ数の次元を維持します。2 つのテスト関数の結果を図 1 に示します。最適化されたガウス混合関数には相互作用しない変数があるため、変数を個別に最適化して、$d = 1$ の Dropout-Copy 法で到達する最大値を改善できます。Dropout-Copy では $d = 5$ の時が最適です。ただし、Schwefel の 1.2 関数を使った場合、変数個別に最適化することは効率的な方法ではありません。図 1（b）は、d が大きいほど収束速度が速いを示しています。これは、d が大きいほど、規定された反復内で相互作用する変数を最適化する確率が比較的高いことで説明できます。これらのグラフは、以下のことを示しています : d が大きい場合、d 次元空間での大域的最適化にはコストがかかる可能性があり、d が小さい場合、相互作用する変数を持つ関数の収束速度は遅くなります。**On the probability** p Dropout-Mix の確率 p の影響を調べるために、$p = 0$、0.1、0.5、0.8、1 を設定します。Dropout-Copy と Dropout-Random は、Dropout-Mix の特殊なケースです（$p = 0$ と $p = 1$）。この実験では、低次元のガウス混合関数（$D = 2$ および $D = 5$）に対して、$\mu_1 = [2, 2, \ldots, 2]$、$\mu_2 = [5, 5, \ldots, 5]$、$X = [0, 7]^D$ であるため、2 つのモードは遠いです。$D = 2$ の場合は $d = 1$ を使用し、$D = 5$ の場合は $d = 2$ を使用します。高次元のガウス混合関数と Schwefel の関数両方は以前と同じ設定を維持し、$D = 20$ の場合は $d = 5$ を使用します。 結果を図 2（a）、（b）、（c）に示します。

Dropout-Mix と Dropout-Random は低次元で機能することがわかります。 ドロップアウト-コピーはうまく機能せず、低次元関数の局所最適にスタックする可能性があるためです。 これは高次元では低い確率で発生するため、Drop-Copy の平均パフォーマンスは図 2

（c）の Dropout-Mix よりもわずかに優れています。 DropoutCopy は、図 2（d）に示すように、常にユニモーダル関数に対して最高のパフォーマンスを発揮します。したがって、高次元では、p が小さい値を設定して（0.1、0.2 など）DropCopy および Drop-Mix を使用することをお勧めします。**Comparison with existing approaches** 上記の実験に基づいて、$d = 2$ 場合は $D = 5$、$d = 5$ の場合は $D = 10$、20、30 でアルゴリズムをテストします。ドロップアウトミックスは $p = 0.1$ で適用されます。ガウス混合関数の場合、すべての D に対して $\mu_1 = [2, 2, \ldots, 2]$ および $\mu_2 = [3, 3, \ldots, 3]$ を設定します。関数の構造がわからないため、各グループで d 次元の射影を持つ REMBO 法と d 次元の ADD-GP-UCB 法を使用します。ここで、値 d はドロップアウトアルゴリズムと同じです。この 2 つの関数に対して 500 個の関数評価を実行します。結果をそれぞれ図 3 と図 4 に示します。低次元（D = 5 および D = 10）では、標準 BO が最高のパフォーマンスを発揮します。高次元（D = 20 および D = 30）では、Dropout-Mix および Dropout-Copy は他のベースラインを大幅に上回っています。当然、REMBO 法と ADD-GP-UCB 法では、関数固有の構造が以前の仮定に適合しないため、うまく機能しません。

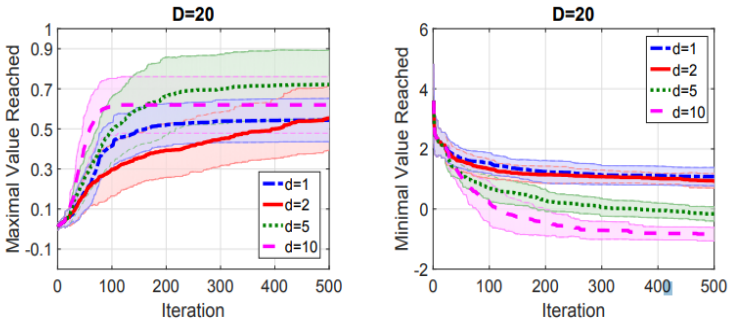
4.2 Training Cascade Classifier

UCI リポジトリから 3 つのデータセット（IJCNN1、ドイツ語、電離層データセット）でカスケード分類器 [Viola and Jones,2001] をトレーニングすることにより、ドロップアウトアルゴリズムを評価します。カスケード分類器は、一連の弱い分類器で構成されます。それぞれの弱い分類器は単純決定木です。インスタンスの重みは、前の弱い分類器からのエラー率に基づいて更新されます。したがって、決定木でのしきい値は非常に重要です。一般的に、しきい値を個別に計算することは最適な戦略ではありません。トレーニングの精度を最大化することにより、最適なしきい値を見つけようとします。すべての日付セットの機能は、$[0, 1]$ の間でスケーリングされます。ステージの数は、データセット内の特徴の数と同じに設定されます。すべてのデータセットに $d = 5$ および $p = 0.1$ を使用します。DIRECT が機能できるため、Add-GP-UCB の各グループのディメンションが 10 未満であることを確認しします。実験結果を図 5 に示します。Dropout-Copy と Dropout-Mix は同様に機能しますが、他の方法よりも大幅に優れています。

4.3 Alloy Design

AA-2050 は、低密度で耐食性の高い合金であり、航空宇宙用途に使用されます。現在の合金は数十年前に設計されており、冶金学者の協力者によってさらなる改善の第一候補と見なされています。合金組成の有用性を測定するために、ソフトウェアベースの熱力学シミュレーター（THEMOCALC）を使用します。合金は 9 つの元素（Al、Cu、Mg、Zn、Cr、Mn、Ti、Zr、Sc）で構成されています。効用は、生成される 4 つのフェーズの加重組み合わせによって定義されます。相は内部構造に関連し、合金の特性に影響を与えます。全部で 13 次元の最適化問題（9 つの要素と 4 つの操作パラメーター）があります。この実用性を最大化する構成を求めています。結果を図 6 に示します。約 4.5 の効用から始めました。 500 回の最適化の反復後、Dropout-Mix は 5.1 の効用を達成できますが、標準 BO

は 100 回の反復後も約 4.8 を維持します。結果は、合金設計の実際のアプリケーションに対する私たちの方法の有効性を示しています。

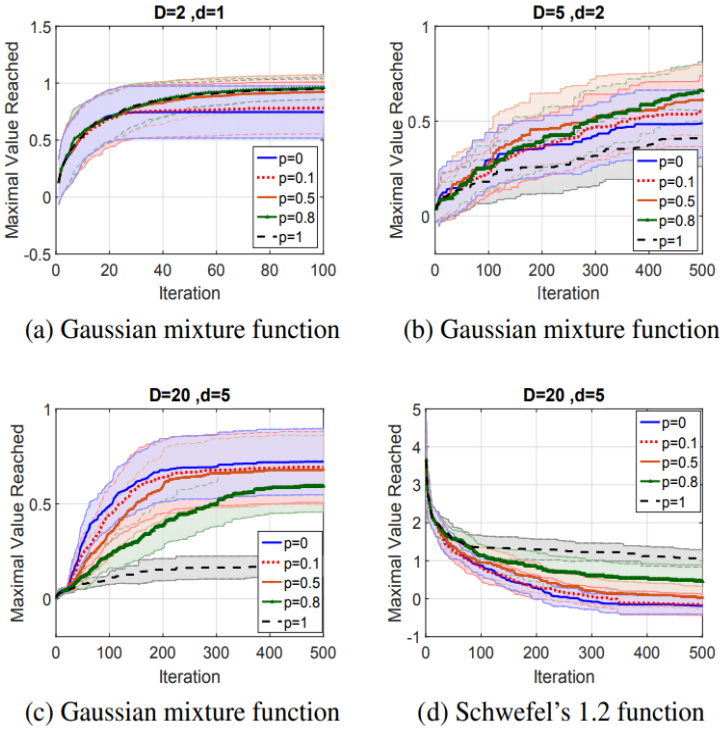


(a) Gaussian mixture function (b) Schwefel's 1.2 function

図 1: The effect of the number of dimensions d in DropoutCopy. The y-axis in Gaussian mixture function presents the real function value (Higher value is better). The y-axis in Schwefel's 1.2 function presents the logarithm of function value (Lower value is better).

5 Conclusion and Future Work

ドロップアウト戦略を用いて、高次元ベイズ的最適化の新しい方法を提案しました。ドロップアウトされたディメンションから変数を入れるため 3 つの戦略を開発します。「ランダム値」、「これまでに見つかった最良のサンプルからの値」、およびこれらの戦略を組み合わせた方法が含まれます。私たちのメソッドの regret 限界が導き出され、議論されました。合成および実際のアプリケーションに関する実験結果は、私たちの方法が高次元の最適化に効果的に機能することを示しています。 ドロップアウトされた次元から構築された獲得関数に局所的最適化のみを適用する場合、それは有望かもしれません。将来的には、より効率的な次元サブセットの選択方法を検討していきます。



(a) Gaussian mixture function (b) Gaussian mixture function

(c) Gaussian mixture function (d) Schwefel's 1.2 function

図 2: The effect of the probability p in Dropout-Mix

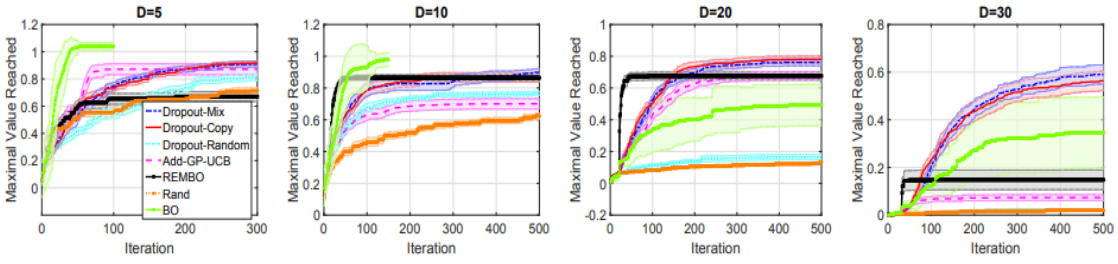


図 3: The optimization for the Gaussian mixture function. Higher value is better. Four different dimensions are tested from left to right (a) D = 5 (b) D = 10 (c) D = 20 (d) D = 30. The BO for D = 5 and D = 10 is terminated once it converges. The graphs are best seen in color

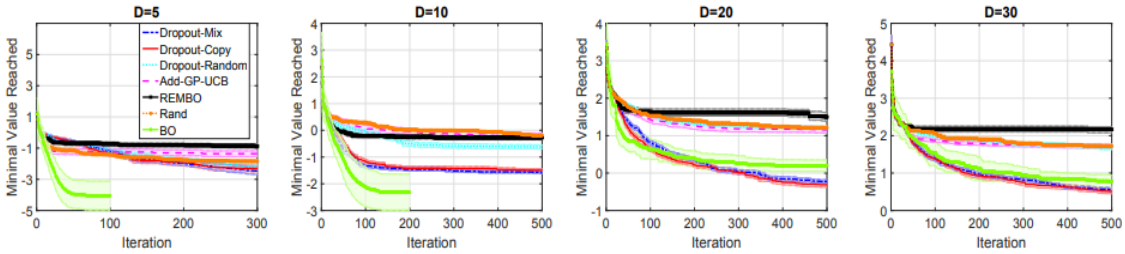


図 4: The optimization for Schwefel's 1.2 function. Lower value is better. Four different dimensions are tested from left to right (a) D = 5 (b) D = 10 (c) D = 20 (d) D = 30. The graphs are best seen in color.

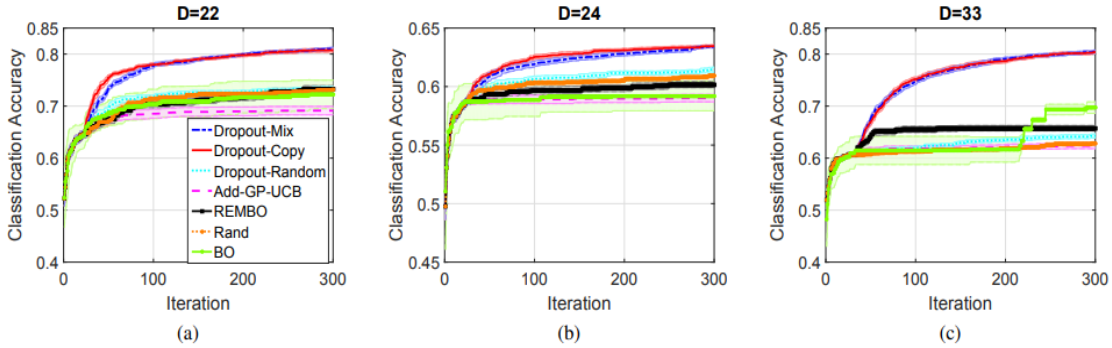


図 5: Maximum classification accuracy for training data as a function of Bayesian optimization iteration. The number of stages in a cascade classifier is equal to the number of features in three datasets (a) IJCNN1 D = 22 (b) German D = 24, (c) Ionosphere D = 33.

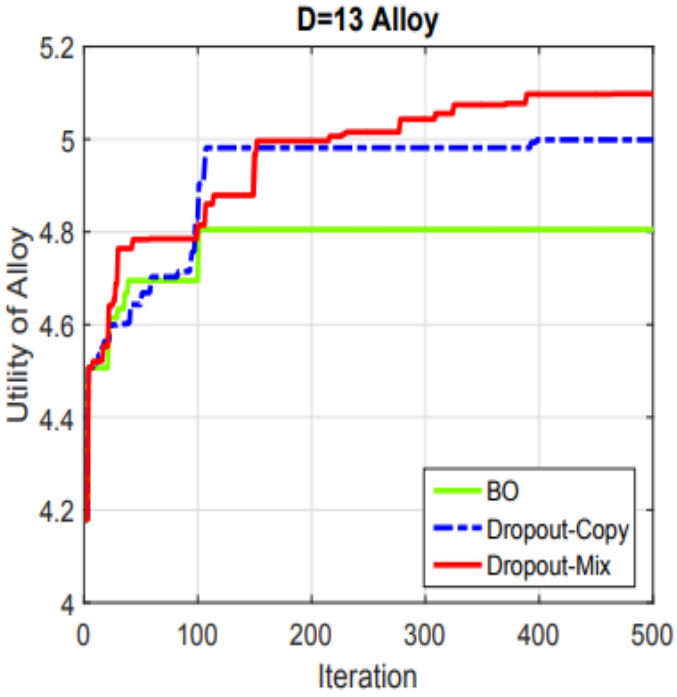


図 6: The utility of alloy vs the iterations of Bayesian optimization. The number of optimal parameters is 13. We use d = 5 in this experiment.