修士論文

高効率かつロバストな
多入力アナログ演算器の設計と評価

上竹　規之

2019年1月31日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

上竹　規之

審査委員：

中島 康彦 教授	（主指導教員）
井上 美智子 教授	（副指導教員）
中田 尚 准教授	（副指導教員）
Tran Thi Hong 助教	（副指導教員）
張 任遠 助教	（副指導教員）

高効率かつロバストな
多入力アナログ演算器の設計と評価*

上竹　規之

内容梗概

IoT や機械学習アルゴリズムが飛躍的発展を遂げ、組み込み機器の実アプリケーションとして実社会に使われるようになった。それに伴い組み込み機器に要求される演算性能も高くなっている。一方で、デナード・スケーリング則の恩恵が減りつつある今、集積度による性能向上が難しく、今までにはない形の計算方式の模索を強いられている。この課題に対して、本研究では、アナログ回路を用いた近似計算による改善を試みる。近似計算とは、厳密解の代わりに近似解を算出することで、演算効率を向上させる手法である。アナログ回路は、デジタル演算器と比べて、クロックレスのため演算速度が速く、小規模実装できるが、温度や製造ばらつきなどのノイズによる性能劣化やノンプログラマブルである点が課題である。しかし、我々の提案する演算器は回帰アルゴリズム SVR をアナログ回路に実装することにより、プログラマブルなアナログ演算器である。近似対象の関数は、SVR の教師あり学習によって決定される座標 SV を用いたカーネル関数の総和で表現される。本研究では、カーネル関数にガウシアン関数を用いた近似を行っており、動的に調整可能な近似ガウシアン関数を出力するカーネル回路を実装することで、アナログ回路による SVR の実装を可能とする。また、このカーネル回路の出力が電流であるため、キルヒホッフの法則によって SVR の総和を算出する。この演算器の回路規模は SV の数に依存しており、複数回に渡る学習と入力サンプルの厳選を行い、小規模実装かつ高精度なアナログ演算器の実装を試みる。本稿は、SVR を使うことができるオープンライブラリ LIBSVM と回路シミュレーションツールで

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, 2019 年 1 月 31 日.

ある HSPICE を用いたアナログ演算器の設計及び評価を行う。ノイズによる性能劣化を考慮しない 1,2,9 変数の関数近似例として、$f(x) = x^2, f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$, $f(\mathbb{X}) = \sqrt{x_1^2 + x_2^2 + \cdots + x_8^2 + x_9^2}$ を対象とした近似計算を行い、それぞれの誤差が $2.64\%, 4, 20\%, 10.85\%$ となり、演算時間は 330n 秒であった。また、この演算に用いられた SV の数はそれぞれ 10 個、20 個、90 個となり、回路規模は 1,2 入力計算がトランジスタ 600 個、9 入力計算が 9000 個となった。しかし、この回路の温度と製造ばらつきを考慮した場合、カーネル回路の誤差が大きかったため、MOSFET のサイズ調整による改善を試みた。$f(x) = x^2, f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ の近似計算による誤差がそれぞれ $7, 52\%, 5.74\%$ となり、ノイズの影響の低減に成功した。また、2 入力近似計算の最適化を行い、SV が 10 から 25 個の場合が誤差も許容範囲かつ小規模実装可能でであることを確認した。

キーワード

エッジコンピューティング, 近似計算, サポートベクトル回帰, 最適化

Design and Evaluation of Efficient and Robust Analog Calculation Unit with Multiple Inputs*

Noriyuki Uetake

Abstract

Along with the fast development of IoT andmachine learning algorithms, the implementations of embedded devices havewidely appeared in real-world applications, which are hungry for the high computing capacity. On the other hand, the computing performance is hardly improved only by the benefits of higher integration since the road-map of scaling-down will reach the end soon. Thus, innovative computing technologies should be developed. For this purpose, the approximate computing technology implemented by analog circuits is explored. The approximate computing aims at improving hardware efficiency by fault-tolerant processing instead of exact computations. The analog calculation appears some merits on high speed (due to calculations in continuous time) and compact scale, but suffers from the poor programmability and sensitivity against temperature and process variations. In this work, a programmable analog calculation unit is developed by implementing support vector regression (SVR) algorithm with analog circuitries. A set of dynamically tunable Gaussian function generation circuits are designed for performing the kernels of SVR in terms of analog current. By collecting thecurrent of all the Gaussian kernels, the regression is carried out from the SVR theory. Obviously, the circuit scale and cost strongly depend on the number of support vectors (SVs). To shrink the hardware resource, a novel scheme of SVR is proposed by repeating the SVR learning process iteratively. By using this

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, January 31, 2019.

scheme, the number of SVs is greatly reduced with a high regression accuracy, which leads to the reduction of circuit scale. The machine learning platform LIBSVM is employed for SVR training; and HSPICE is used for the circuit simulation of the proposed analog calculation units. Several example functions are verified for proof-of-concept as: one-operand function $f(x) = x^2$ two-operand function $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ and nine-operand function $f(\mathbb{X}) = \sqrt{x_1^2 + x_2^2 + \cdots + x_8^2 + x_9^2}$. From the simulation results without the consideration of variations, these three functions are executed by the proposed ACU in 330ns with the inaccuracy of 2.64%, 4.2%, and 10.85%, respectively. The ACUs for implementing two- and nine-operand functions are designed with the scales of 600 and 9000 transistors, respectively (corresponding to 20 and 90 SVs). Moreover, the robustness against temperature and process variations is investigated and improved. By optimizing the sizes of MOS transistors, the computing errors with the considerations of temperature and process variations are reduced to 7.5% and 5.74% for the example of $f(x) = x^2$, $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$. The optimum number of SVs is found as the range from 10 to 25 for two-operand calculation, which allows to build ACUs in a small circuit size with acceptable accuracy.

Keywords:

Edge computing, Approximate computing, Support Vector Regression, Optimisation

目 次

図目次

表 目 次

1. はじめに

1.1 背景

近年、人が身に着けても気にならないほどの小型かつ軽量な組み込み機器の実現に成功したことにより、スマートフォンやウェアラブルデバイスなどのセンシングデバイスの需要が高まりつつある。これに追従するように、IoT や機械学習、ビッグデータといった技術も進展し、社会に浸透しつつある [3][4]。これらの技術は膨大な計算量を必要とすることがあり、サーバのような潤沢な計算資源が用意された場所で集中処理することが一般的である。しかし、図 1 に示すように、これらのデバイス数は今後も増加することが予測されており、エッジデバイスからサーバへのデータ転送の一極集中によるネットワーク帯域の圧迫が問題視となることが予想され、エッジデバイスで処理の一部を担うエッジコンピューティングなどによる改善が行われている [5]。それにより、低消費電力・小規模といった厳しい制約を必要とする組み込み機器にも高効率かつ強力な演算性能をもつ演算器が求めれらている。一方で、現在まで続く集積回路技術の立役者ともいえる存在であるデナード・スケーリング則 [6][7] の崩壊が囁かれている。この法則は 1974 年に R.H.Dennard 氏らによって提案された指標で、トランジスタの微細化が集積度、速度、消費電力の性能向上をもたらすというものである。そのため、この法則は、集積回路技術には必要不可欠であった半導体性能の向上を支えてきた。しかし、2000 年代初頭から微細化による恩恵が少なくなっていることが、文献 [8] から理解できる。集積度による性能向上の限界が見えてきたことで、今まで通りの手法による組み込み機器の演算性能の向上は困難である。つまり、今までにはない計算方式を用いて、組み込み機器の制約を満たす高効率演算器が求められている。

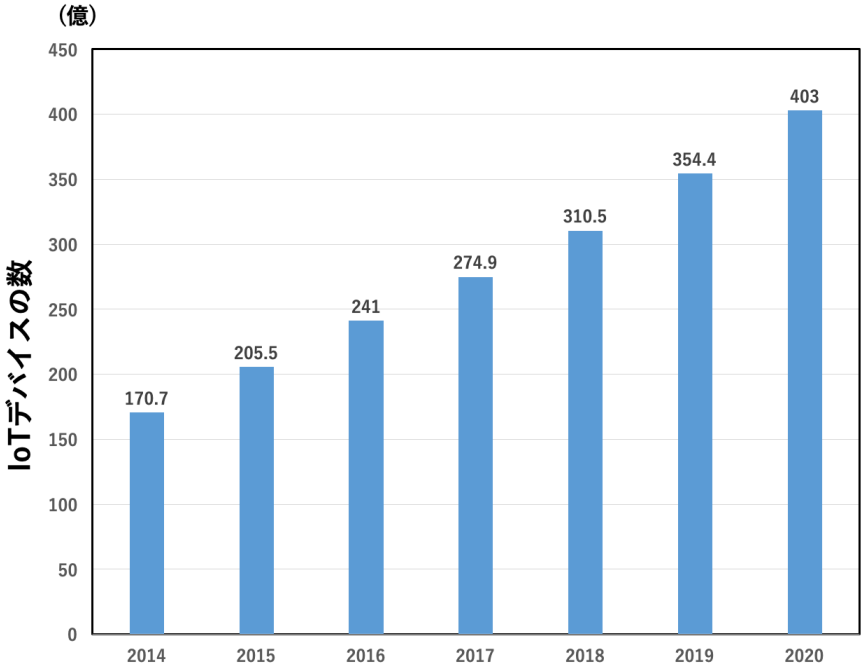


図 1 世界の IoT デバイス推移と予測 [1]

1.2 研究目的

本稿では、上述の課題に対して、アナログ演算器による近似計算を行うことで解決を試みる。アナログ演算器はデジタル演算回路とは違い、ノンプログラマビリティであり、温度や製造ばらつきによるノイズの影響を受けやすく、出力に誤差が存在するため、精密演算ができない短所を持っている。一方で、デジタル演算回路と比べて、高速かつ低消費電力・小規模実装可能という組み込み機器に適切といえる長所を持っている。また、アナログ演算器の精度の悪さも厳密解を必要としない近似計算として用いることで、演算による誤差が許容範囲内を超えない限り、実用可能である。

アナログ演算器のもう一つの欠点であるプログラマビリティは、Support Vector Regression(SVR)[9][10][11] と呼ばれる回帰アルゴリズムによる近似計算を、アナログ回路に実装することによって解決可能である。これにより、トランジスタの微細化に代わる高効率な演算器の実現を目的とする。

1.3 構成

本稿では、本研究背景及び目的を述べた。2 章では、近似計算についてより深く見ていくとともに、先行研究として他の近似計算手法について述べる。それらのことを踏まえた上で、本稿で提案する手法の近似手法及び、アナログ回路実装について述べる。このアナログ演算器の回路シミュレーションを行い、近似精度を算出する。考察では、他のハードと精度や規模などの比較、温度と製造ばらつきといった外的要因が誤差に与える影響の大きさの調査及び、この演算器の最適化について述べる構成である。

2. 近似計算と関連研究

2.1 近似計算とは

前章で述べたように、IoT やビッグデータの発展により、効率的で強力な演算器の要求が高まっている。演算効率を上げる手法の一つに近似計算と呼ばれる手法がある。この手法は、厳密解の代わりに近似解を算出することで、計算量を減らし、小規模、低消費電力・高速な演算を行う手法である [12]。機械学習やビッグデータ解析への応用が期待されており [4][13]、ヒューマンインターフェースや認識など様々なアプリケーションやデータセンタの膨大なデータ処理の影響で大容量化しているメモリの省電力化手法などにも研究されている [14][15][16]。機械学習やビッグデータなどの現在主流とされている集中処理システムは、エッジコンピューティングとは違い、消費電力、回路規模などの制約が低く、ビッグデータ分析や機械学習といった膨大な計算量を要求する処理を行うことができ。パフォーマンス向上のために計算品質を下げるために、対象としている計算が許容できる誤差の範囲や計算結果が本当に信頼できるかを把握する必要があり、近似計算への敷居をさげるために、近似計算可能なプログラムを見つけたり、出力品質を管理するするような機能も模索されている [17]。デジタル回路はビット幅によって精度が決定されるため、ビット幅を調整することで精度を変えることができ、精度とパフォーマンスのトレードオフの調整が容易である [18][19][20]。しかし、アナログ演算器はデジタル演算器よりも高速実行可能であり、低消費電力を実現できる可能性があるため、現在よりもより効率的な演算器となる可能性を持っている [21][22][2]。

2.2 ビット幅の削減

近似計算のシンプルな例として、ビット幅を小さくすることで、回路規模や消費電力を削減する手法がある。文献 [20] は、小さいビット幅の ALU の例を示しており、全体像を図 2 に示す。ALU はデジタルシステム設計において重要なシステムであり、四則演算や論理演算などの算術的処理を行う。本例の ALU は 4 つの選択入力 (S) とモード制御入力 (M) により、16 種類の算術及び論理演算を行うことが可能である。また、全ての演算が並列に行われ、同時利用可能であるため、クロックリソースを無駄にしないようになっている。MUX では必要な出力を選択するために用いられる。ALU はデジタル演算器であるため、ビット幅と同じ数の配線を必要とし、高精度な演算を行うには、回路規模や消費電力が大きくなる。そのため、ビット幅を小さくすることによる恩恵をアナログ演算器よりも大きく受けられる。

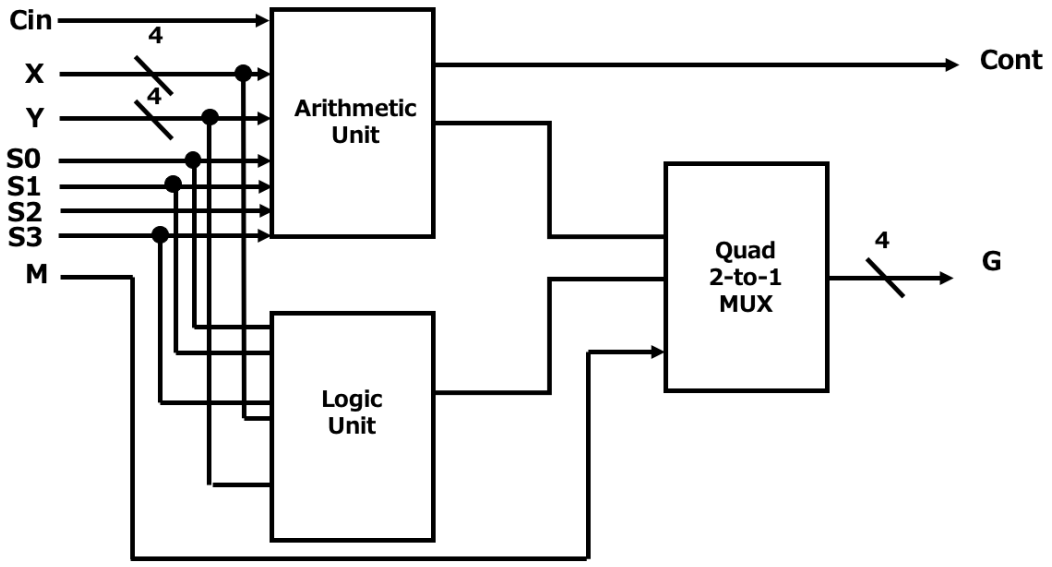


図 2 4bit ALU

2.3 アナログ LUT

アナログとデジタルを組み合わせ手法 [23] は、多値論理を用いた近似手法であり、全体像を図 3 に示す。アナログ入力をデジタル値に変換し、Look up tables(LUT) により出力値を決めた後、DAC によりアナログ値に戻している。この演算器は非線形微分方程式を解くことができ、LUT を用いることで、任意の計算を可能にしている。また、アナログ入力とアナログ出力にすることで、演算速度の向上や連続値を出力できるというメリットがある。一方で、LUT を用いているために、回路規模が大きくなるデメリットもある。

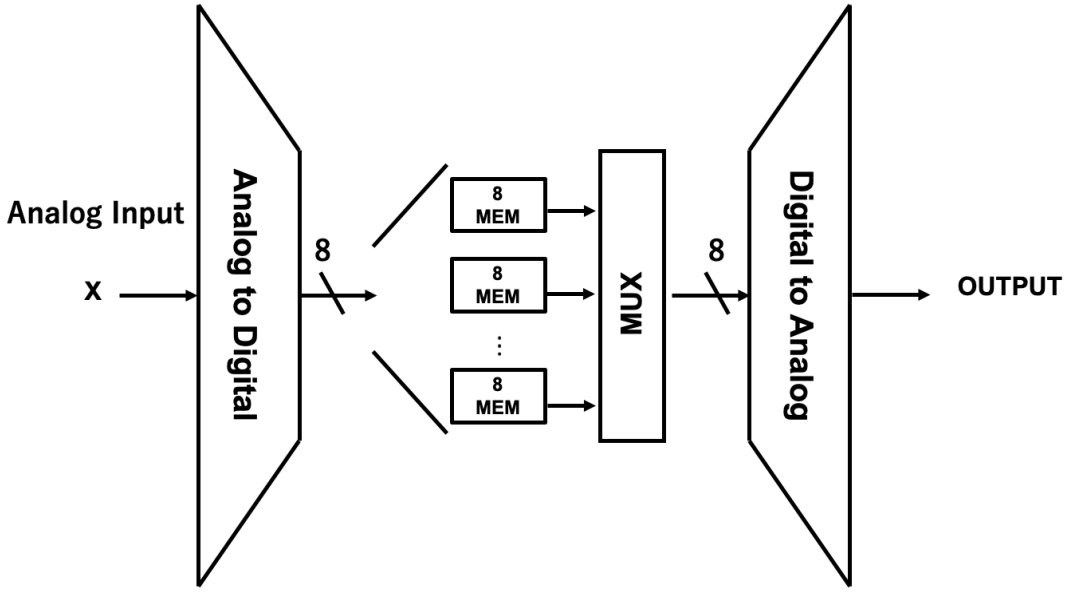


図 3 Analog LUT

2.4 ニューラルネットワークによる近似手法

文献 [24] は、ニューラルネットワーク (NN) を用いた近似計算手法である。NN は cos, sin, exp, log と言った基本的な関数を生成することができ、テイラー展開などを用いて近似対象の関数を NN で再現できる関数で表現し、近似式を算出する。単一の NN によって、関数を表現するため、小規模実装が可能である。入力された関数をマクローリン展開などを用いて複数の関数の和へと変換する。そして、NN で関数再現を行い、CPU で再現された関数の合算を行うことで、関数近似を可能とする。

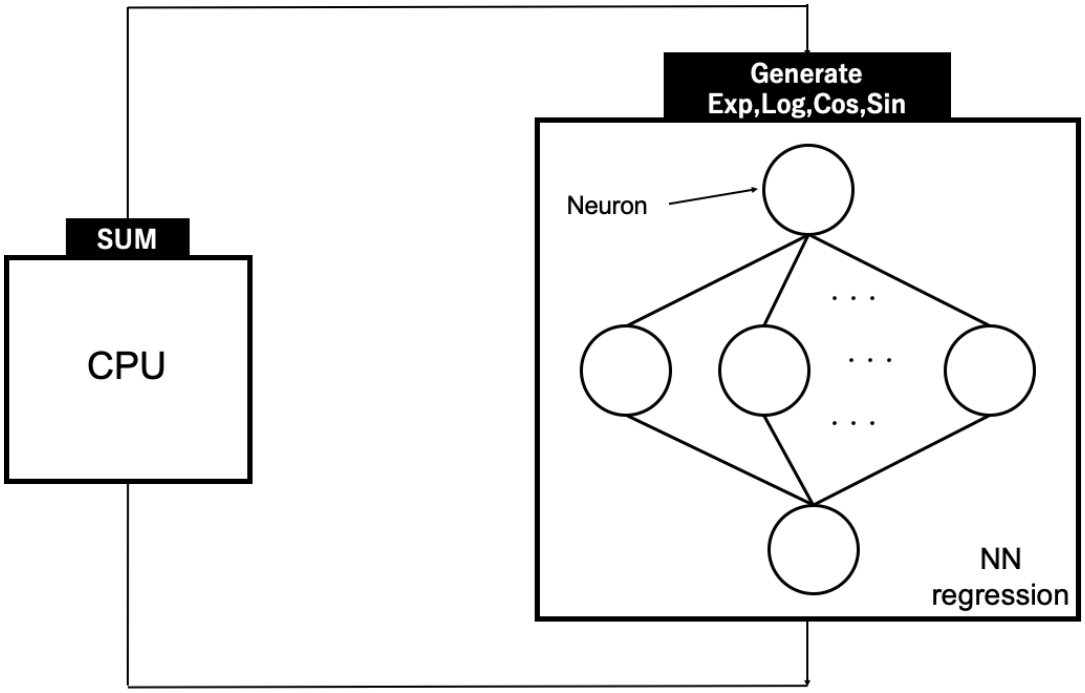


図 4 ニューラルネットワークによる近似例

3. 提案手法

本章では、近似手法である SVR について述べた後、アナログ演算器の構造及び設計手法について述べる。

3.1 Support Vector Regression

SVR とは、パターン認識手法である Support Vector Machine(SVM) を回帰分析に応用したものである [10][11]。SVM と同様に、カーネルトリックと呼ばれる手法を用いる。高次元の特徴空間への非線形写像をすることで、本来線形分離不可能なデータセットも非線形回帰分析を行うことができる。また、単回帰や多項式回帰とは違い、事前に回帰曲線にあたりをつける必要がなく、モデルを事前仮定しないノンパラメトリックな手法である。

回帰分析は、説明変数 $X = [x_1, ..., x_n]$ と目的関数 y との間の因果関係を求める式 f を算出することであり、n は説明変数の個数を示している。SVR の基本的な部分は線形回帰分析手法と同じであり、回帰ベクトル b を用いて、i 番目 $(1 \leq i \leq n)$ のサンプルの目的関数の値 $f(x^{(i)})$ は以下のように記述される。

$$f(x^{(i)}) = x^{(i)}b \tag{1}$$

線形分離不可能な空間であった場合であっても、カーネル関数 ϕ を用いた非線形写像:$x \rightarrow \phi(x)$ を行い、次元数 k の高次元空間での非線形回帰モデル式の算出を行うことができる。非線形写像された $f(x^{(i)})$ は次のように表される。

$$f(x^{(i)}) = \phi(x^{(i)})w + c \tag{2}$$

図5に示すように、回帰曲線 $f(x^{(i)})$ は重み $w = [w_1, ..., w_k]$ の値と定数 C によって決定される線形しきい素子のような構造をしている。

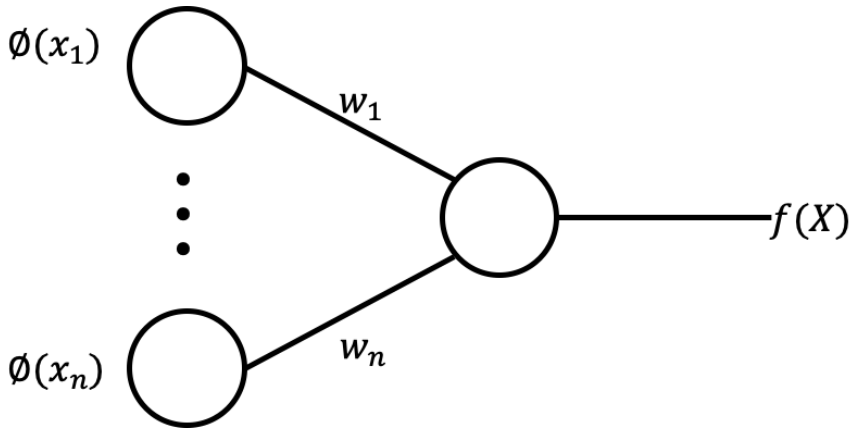


図5 SVRの構造

SVRの学習とは、重み w と誤差 $|y_{(i)} - f(x^{(i)})|$ の最小化である。以下の様に記述され、誤差だけでなく重み w を同時に最小化することで過学習を防ぐことができる。

$$minimize \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} h(y^{(i)} - f(x^{(i)})) \tag{3}$$

誤差関数 h を図6に示す。

$-\epsilon \leq$ 誤差 $\leq \epsilon$の場合において、誤差を0と見なす不感帯を導入することで、ノイズの影響を抑えることができる。

$$h(y^{(i)} - f(x^{(i)})) = max(0, |y^{(i)} - f(x^{(i)})| - \epsilon) \tag{4}$$

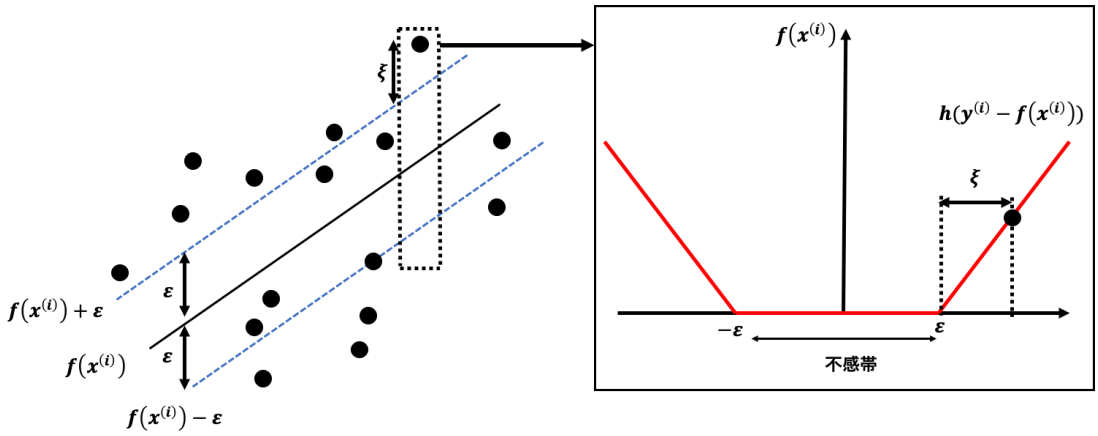


図 6 誤差の不感帯

多くのサンプルが不感帯の中に入っている回帰曲線は誤差が小さいといえるため、回帰面 $f(x^{(i)})$ を中心にデータが正しく分けられているかを評価するための スラック変数ξ,ξ^*を導入する。スラック変数 ξ,ξ^* は、図 7 のように、それぞれのサンプルに対して不感帯との距離を意味しており、スラック変数と回帰曲線の関係は以下の様に表される。

$$y^{(i)} \leq f(\boldsymbol{x}^{(i)}) + \epsilon + \xi \tag{5}$$

$$y^{(i)} \geq f(\boldsymbol{x}^{(i)}) - \epsilon - \xi^* \tag{6}$$

$$subject\ \ to \quad \xi,\xi^* \geq 0 \tag{7}$$

上記の式から、誤差の最小化は以下の様に表される。

$$minimize \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi+\xi^*) \tag{8}$$

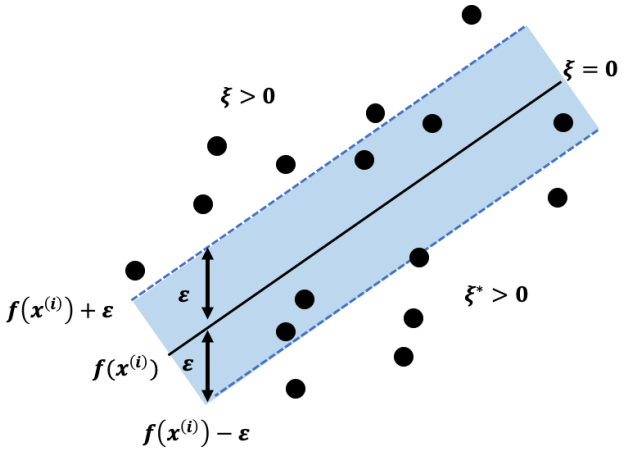


図 7 スラック変数と不感帯の関係

ラグランジュ乗数 $\alpha_i, \alpha_i^*, \beta_i, \beta_i^* (i = 1, 2, ..., n)$ を導入し、

$$\begin{aligned} L = \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi + \xi^*) - C\sum_{i=1}^{n}(\beta\xi + \beta^*\xi^*) \\ & - \sum_{i=1}^{n}\alpha_i(\epsilon + \xi_i + f(\boldsymbol{x^{(i)}}) - y_i) \\ & - \sum_{i=1}^{n}\alpha_i^*(\epsilon + \xi_i^* - f(\boldsymbol{x^{(i)}}) + y_i) \end{aligned} \tag{9}$$

この最適化問題を解くことで、SVR の回帰式が算出される。

$$f(x^{(j)}) = \sum_{i=0}^{n}(\alpha_i - \alpha_i{}^*)K(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)}) + c \tag{10}$$

SVR の学習サンプルの内、$(\alpha_i - \alpha_i{}^*) \neq 0$ であるサンプルを Support Vector(SV) と呼ぶ。本稿の SVR は、カーネル関数として、ガウシアンカーネルを用いた非線形回帰モデルである。

$$K(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)}) = exp(-\gamma\|\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}\|^2) \tag{11}$$

この非線形回帰モデルの $(\alpha_i - \alpha_i{}^*)$ はガウシアン関数の高さ、γはガウシアン関数の幅に相当する。

3.2 SVR による関数近似

SVR はデータセットを使った教師あり学習によって、関数生成するアルゴリズムであり、近似対象の関数の入出力関係をデータセットとして与えることで、関数近似を可能とする。関数近似を行うには、データのサンプリング、学習、回帰曲線の生成という 3 つの手順が必要である。それぞれの手順について簡単に述べる。

まず最初に、図 8 に示すように、SVR の教師データとなるサンプルを近似対象の関数から抽出する。近似対象の関数からデータセットを生成するので、近似対象の関数は既知である。一般的に、サンプルの数が多いほど SV の細かい調整ができ、高精度な近似式が生成されるが、学習にかかる時間も長くなる傾向にある。また、関数の変数の数が大きいほど、学習に必要なサンプルの数も多く必要になり、高い精度を出すことが困難になる。

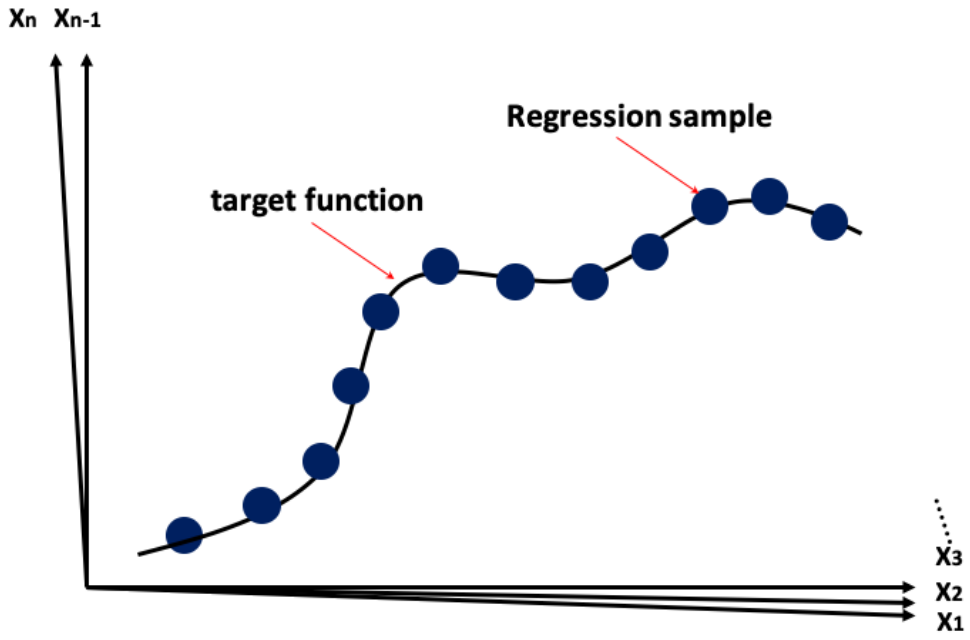


図 8 サンプルの抽出

次に学習について述べる。学習は、前章で述べたように、近似曲線の構成要素となるサンプル SV を決め、それぞれの SV はガウシアン関数の高さとなるパラメータが決められる。

最後に近似曲線の生成について述べる。近似曲線がガウシアン関数の総和にバイアスを加えた式で表現される。一般的に、ガウシアン関数の数が多いほど、高精度な近似曲線を生成することができる。

図 9 は、SV を選ぶ学習と近似曲線の生成を図示している。

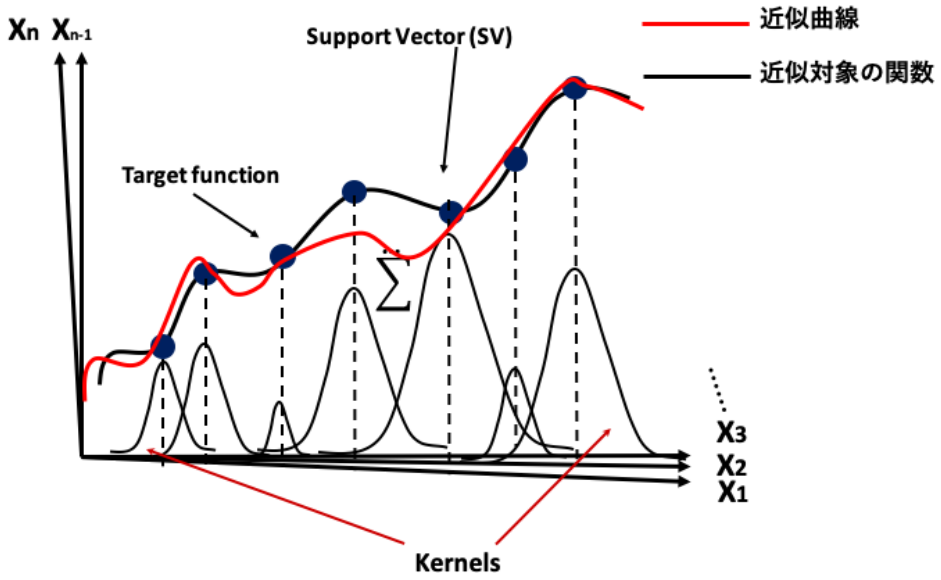


図 9 学習と近似曲線の生成

3.3 Analog calculation unit の全体像

SVR による近似式は、ガウシアン関数の総和とバイアスで表されるため、ガウシアン関数をアナログ回路で実装することで、アナログ演算器の欠点であるノンプログラマビリティを補完することができる。また、デジタル演算器は離散値にすることで、熱やトランジスタの製造ばらつきなどのノイズによって生じる誤差の発生を防いでいる。しかし、アナログ演算器は電流・電圧の値が演算結果となるため、ノイズの影響を受けやすく、必ず誤差が出る。

図 10 は、我々が提案する ACU の全体像を示す。

この演算器は SVR の学習を行うソフトウェア部分と算出されたパラメータと入力 $[x_1.x_2]$ から近似曲線を生成するハードウェア部分に分かれている。図 10 から、SV と同じ数だけのカーネル回路を実装する必要があり、SV の数が回路規模に直接関係していることがわかる。

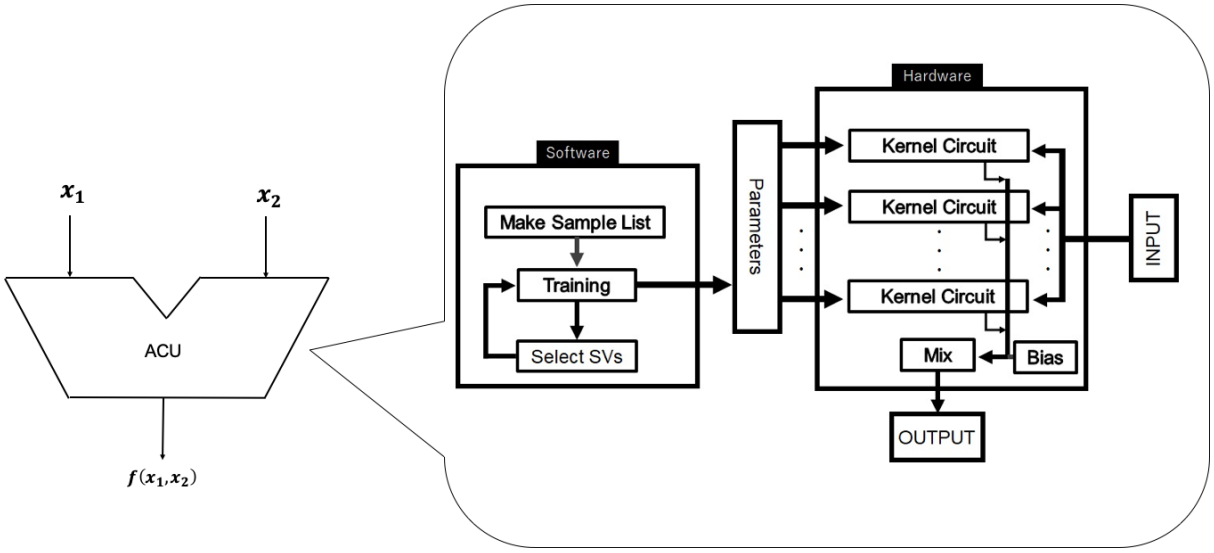


図 10 2 入力 ACU の全体像

3.3.1 ソフトウェア部分

ソフトウェア部分では、サンプリングと学習、そして回路規模を小さくするための SV の厳選が行われる。これを繰り返すことで、SV の数をある程度指定することができ、誤差が許容範囲かつ回路規模も小さい演算器の設計を可能としている。2 変数関数の SV を減らす手法を図 11 に示す。近似対象の関数からサンプリングすることで、121 個の教師データが生成される。その後、1 回目の学習を行い、約 100 個の SV が選ばれ、次に SV の厳選が行われる。SVR によって生成される近似曲線は、ガウシアン関数の総和で表現されるため、SV の厳選基準は、ガウシアン関数の高さ $(\alpha_i - \alpha_i^*)$ が高い SV ほど近似曲線に大きく影響しており、高さが小さい SV ほど近似曲線生成に与える影響も小さい。SV の高さに閾値を設け、$(\alpha_i - \alpha_i^*) < 0.01$ の SV を削除した。1 回目の厳選により、SV の数は約 50 個まで減らすことができた。再度学習を行った後、再び同じ閾値で SV の厳選を行い、約 30 個まで SV を減らした。最後に、SV の数を指定の数になるように調整をし、再学習を行うことで誤差の増加を抑えつつ、SV の数を減らすことを可能としている。

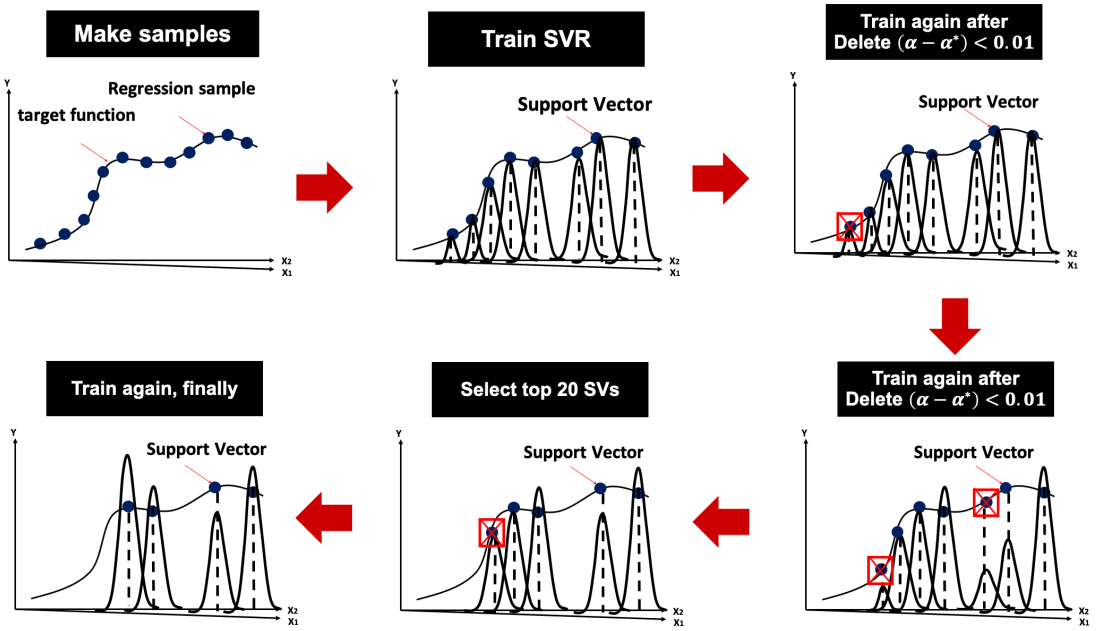


図 11 SV の数の調整方法

表 1 SV の数と誤差の変移

# of learning	# of SV	Error
1st learning	112	0%
2st learning	53	0.000061%
3st learning	28	0.0024%
4st learning	20	0.0036%

3.3.2 ハードウェア部分

ハードウェア部分では、ソフトウェア部分で算出したパラメータを基にして、近似曲線を生成している。回路の全体像を図 12 及び、出力 I_{out} を示す [2]。

$$I_{out} \approx \frac{I_c}{2} e^{-\gamma(\boldsymbol{V}_i - \boldsymbol{V}_j)^2} \tag{12}$$

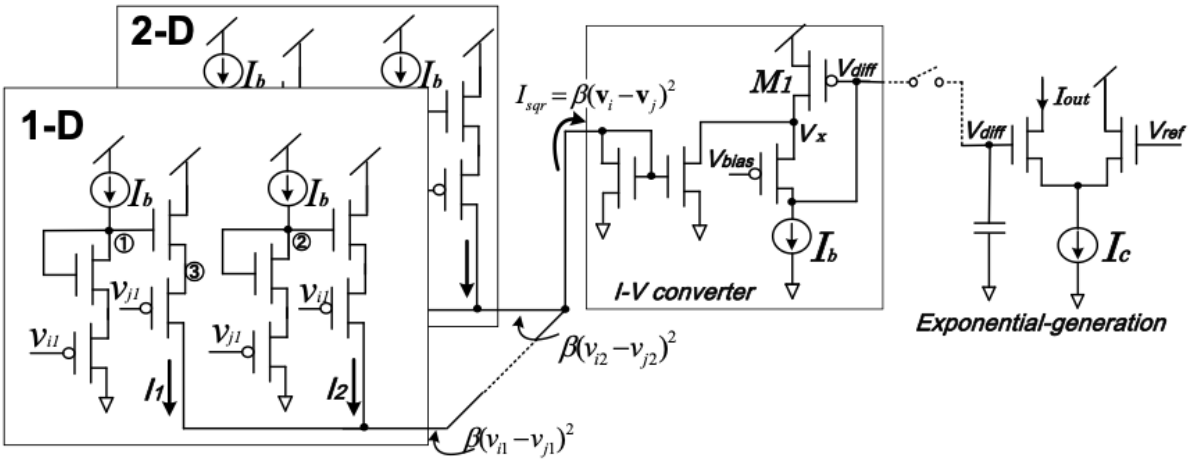


図 12 カーネル回路の回路図 [2]

この回路は、ガウシアン関数の乗数部分に相当する $\beta(\boldsymbol{V}_i - \boldsymbol{V}_j)^2$、ガウシアン関数の幅を決定する γ を決める I-V コンバータ、最後に指数関数の生成と高さを決める 3 つの部分から構成され、I_c で高さを変えることができ、$\boldsymbol{V}_{bias}$ を変化させることで、幅を調整することができる。

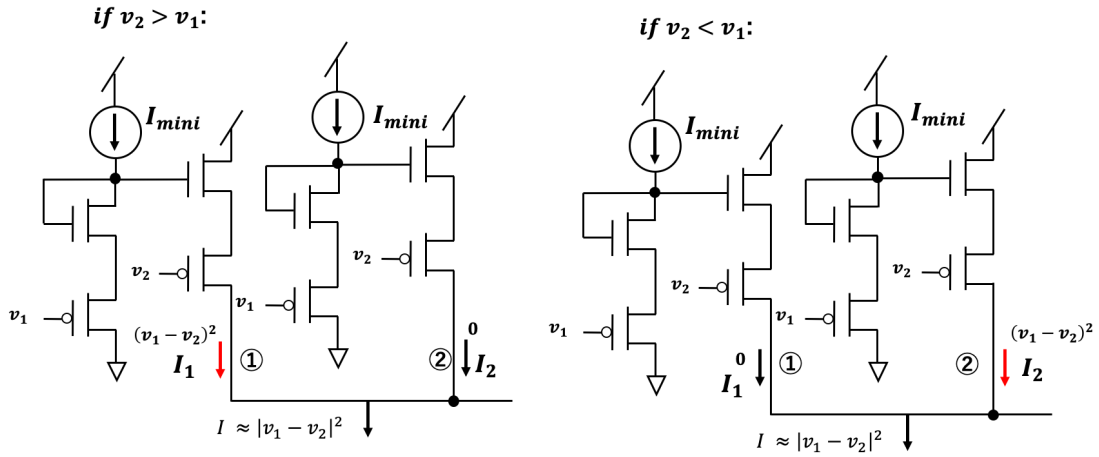
乗数を生成する回路を図 13 に示す。I_b には非常に小さい電流が流れており、$v_2 > v_1$ の時、1 を流れる電流は $I_1 = (v_2 - v_1)^2$ で表現され、2 を流れる電流 I_2 は I_1 比べて非常に小さく、$I_2 = 0$ と見なすことができ、出力 I は下の様に表記される。

$$I \approx (v_1 - v_2)^2 \tag{13}$$

$v_1 > v_2$ の場合は、$I_1 = 0$ となり、$I_2 = (v_2 - v_1)^2$ と表現することができ、出力 I は下記の通りである。

$$I \approx (v_1 - v_2)^2 \tag{14}$$

したがって、v_1 と v_2 の大小関係に関わらず、$I \approx |v_1 - v_2|^2$ が出力される。



$$\boldsymbol{I \approx |v_1 - v_2|^2}$$

図 13 ガウシアン関数乗数部分の回路構造

図 14 にはガウシアン関数の幅 γ を決める回路構造が図示されており、カレントミラー回路と I-V コンバータによって構成されている。トランジスタ M1 のゲート電圧 V_{out} よりもドレイン電圧 V_x のほうが値が大きく、$V_{out} > V_out$ が成り立つ。そのため、M1 は線形領域にあることがわかり、二乗部分の出力 $I_{sqr} = \beta \sum_k (v_{ik} - v_jk)^2$、PMOS の閾値を V_{THP}、電源電圧を V_{dd}、ドレインーソース間の電圧を V_{ds} とすると、V_{out} は以下の様に記述される。

$$V_{ds} = V_{dd} - V_x \tag{15}$$

$$V_{diff} = V_0 - \frac{I_{sqr}}{Kp(V_{dd} - V_{bias} - |V_{THP}|)} \tag{16}$$

$$= V_0 - \gamma \sum_k (v_{ik} - v_{jk})^2 \tag{17}$$

$$V_0 = \frac{V_{dd} + V_{bias} - |V_{THP}|}{2} \tag{18}$$

$$\gamma = \frac{Kn}{(V_{dd} - V_{bias})(\sqrt{Kp} + \sqrt{Kn})} \tag{19}$$

上記の式から、γ が V_{bias} によって決まることがわかる。
Kp、Kn はトランスコンダクタンスを表すパラメータである。

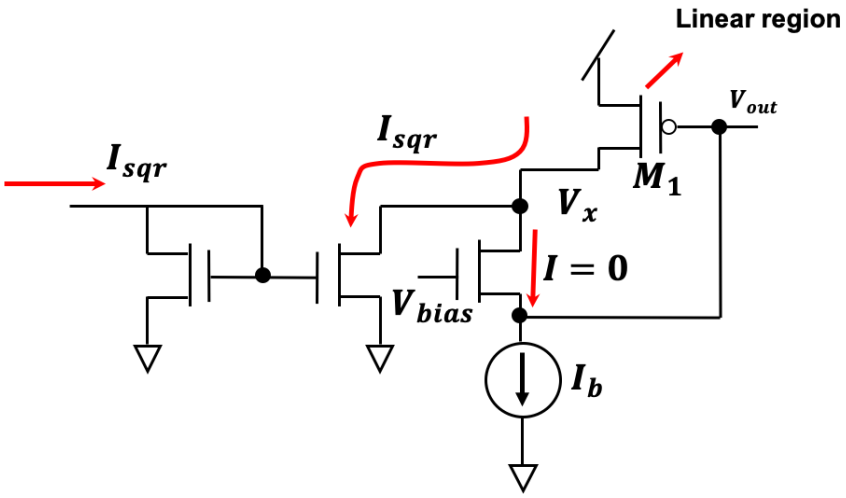


図 14 ガウシアン関数の幅を決定する回路構造

図 15 に、指数関数を生成する回路を示す。I_c を中心に対称構造している。v_{ref} 側の入力には、SV の値が入力されるため、$v_{jk} = v_{ik}$ となる。
V_{diff} は式 (17) のように記述されるのに対し、V_{ref} は

$$V_{ref} = V_0 - \gamma \sum_k (0)^2 = V_0 \tag{20}$$

また、定電流 I_c を用いて、I_{out} は以下の様に記述される。

$$I_{out} = \frac{I_c}{1 + e^{\Delta V}} \tag{21}$$

$$\approx \frac{I_c}{2} e^{-\Delta V} \tag{22}$$

$$\Delta V = |V_{diff} - V_{ref}| \tag{23}$$

式 (22) は式 (17)(20)(23) を代入することで以下の様に記述される。

$$I_{out} = \frac{I_c}{2} e^{-|V_{diff} - V_{ref}|} \tag{24}$$

$$= \frac{I_c}{2} e^{-\gamma \sum_k (|v_{ik} - v_{jk}|)^2} \tag{25}$$

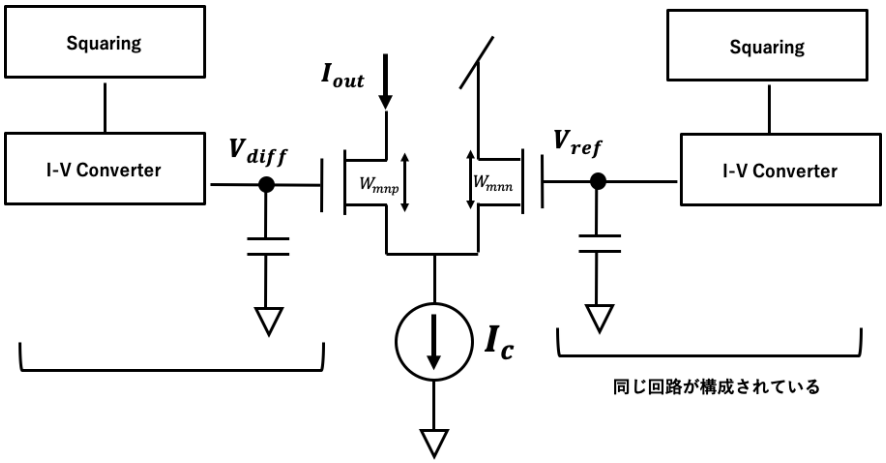


図 15 指数関数を生成する回路構造

SVR によって生成されるガウシアン関数の高さは正の値だけでなく、負の値も取るため、SVR の総和には、和算と減算を行う回路が必要である。図 1617 はカーネル回路の出力を加算・減算する回路の構造をそれぞれ示している。カレントミラー回路は、カーネル回路の構造に関係なく、流れてきた電流 I_{out} と同じ値をもう片方の MOSFET に流す回路である。出力が電流であるため、キルヒホッフの法則によって加算を表現できることから、加算回路は一つの 2 つの MOSFET を組み合わせたカレントミラー回路による表現が可能である。減算も同様にキルヒホッフの法則により、電流の向きを逆にするだけで表現できる。そのため、pMOS で構成されたカレントミラー回路の出力を nMOS で構成されたカレントミラー回路の入力として与えることで、電流の向きを反対方向に変える。

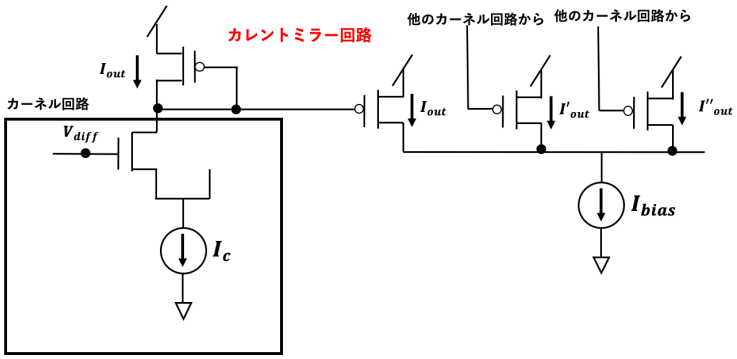


図 16 SVR の加算を行う回路構造

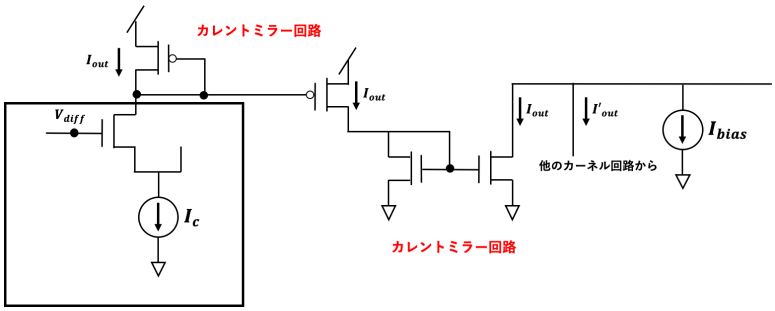


図 17 SVR の減算を行う回路構造

図 18 は、カーネル回路は電流・電圧を調整することで様々なガウシアン関数に近似することができることを示す。カーネル回路の出力は近似ガウシアン関数である。

図 19 は、近似ガウシアン関数と理論的なガウシアン関数の比較を示しており、

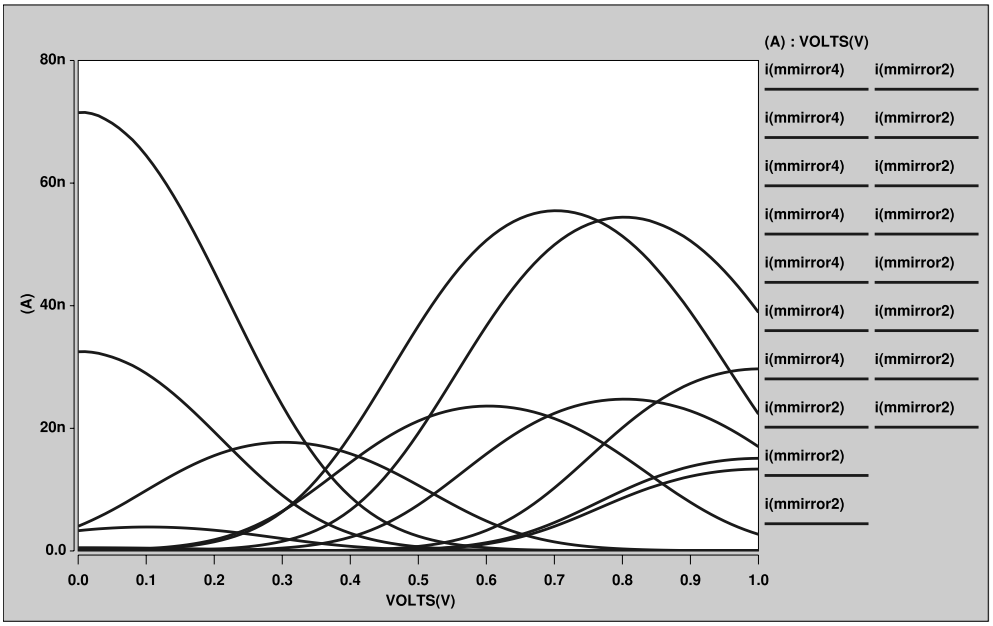


図 18 カーネル回路の実行例

明らかな誤差がある。そのため、カーネル回路の数が多く必要になるほど、カーネル回路による誤差も大きくなると考えられる。この演算器には、近似ガウシアン関数によって生じる誤差と温度や製造ばらつきノイズから生じる誤差の二種類が存在する。

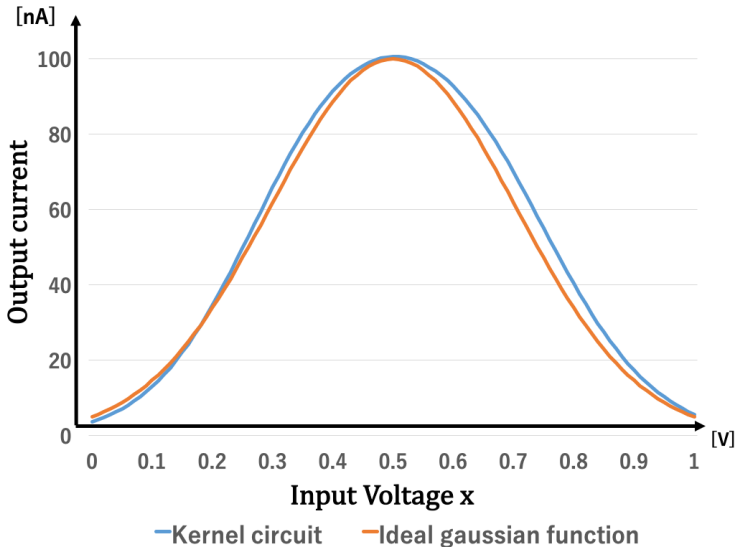


図 19 近似ガウシアン関数と理論値の差

3.3.3 ノイズを抑える調整

温度や製造ばらつきによる誤差を測定した結果、カーネル回路の出力変動の幅が大きく、誤差の許容が難しいことが判明したため、MOSFET の幅や高さを調整することによる改善を試みた。MOSFET の幅や高さを変えることによる誤差の変化量を確認することで、温度や製造ばらつきに大きく影響を与えている MOSFET を見つけ、その MOSFET の高さや幅を調整することで、影響を抑えた。

図 20,21 は、MOSFET を調整する前後の出力結果を比較している。調整前の出力は温度と製造ばらつきともに大きく影響を受けている。一方で、調整後のガウシアン関数は製造ばらつきの影響を小さくなっていることが確認できる。

温度の場合、カーネル回路の頂点にあたる V=0.5 の時が最も誤差が小さく、頂点から離れるにつれて誤差が大きくなる傾向にある。一方、製造ばらつきは、頂点の座標が最も誤差が大きい。

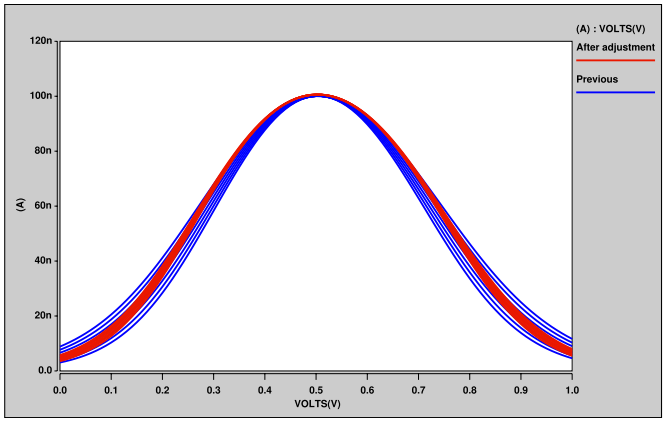


図 20 温度がカーネル回路に与える影響の差

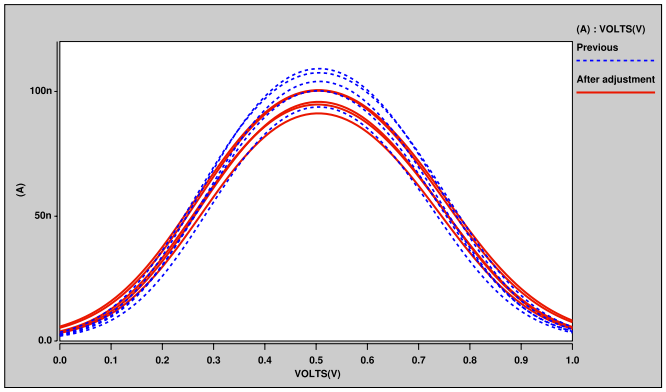


図 21 製造ばらつきがカーネル回路に与える影響の差

4. 実験

4.1 実験概要

SVR の学習及び理論的な誤差の測定には LIBSVM[25][26] を、回路シミュレーションには HSIPCE を使った実験を行った。LIBSVM はガウシアン関数の幅と最大の高さを指定した学習を行う。本実験では、最後の学習時のガウシアン関数の幅 $\gamma = 11$、$(\alpha_i - \alpha_i^*) \leq 2.0$ とした。

回路シミュレーションでは、ノイズによる誤差を考慮しない場合と考慮した場合の両方を行った。温度によるノイズは 20 から 80 度に 10 度ごとに変化させたものとした。製造ばらつきによるノイズは HSPICE のモンテカルロ解析によってトランジスタの製造ばらつきを再現した。温度と製造ばらつき両方を含んだ場合の誤差を測定し、それぞれの影響の大きさについて考察を行う。

この回路シミュレーションでの演算結果は電流で出力されており、出力電流は 100[nA] を演算結果 1.0 とする。

4.2 誤差の定義

本実験では、平均絶対値誤差を用いる。誤差を調べる用いたサンプルサイズを n とし、1,2,9 変数関数において、11 個、121 個、19683 個とする。

4.2.1 ノイズを考慮しない誤差の算出方法

ACU が出力した値 $\tilde{f}_i$ と近似対象の関数の理論値 f_i として、誤差 $Error_{total}$ は以下の様に表す。

$$Error_{total} = \frac{1}{n}\sum_{i=0}^{n}|\tilde{f}_i - f_i| \tag{26}$$

4.2.2 ノイズを考慮した誤差の算出方法

ACU の出力と理論値の誤差 $Error_{total}$ と温度や製造ばらつきによって生じる誤差 $Error_{diff}$ は、最も誤差の大きくなるように変化した出力 $\tilde{f}^{(i)}_{worst}$ と最も誤差が小さくなるように変化した $\tilde{f}^{(i)}_{best}$ を用いて以下の様に表す。

$$Error_{total} = \frac{1}{n}\sum_{i=0}^{n}|\tilde{f}^{(i)}_{worst} - f_i| \tag{27}$$

$$Error_{diff} = \frac{1}{n}\sum_{i=0}^{n}|\tilde{f}^{(i)}_{worst} - \tilde{f}^{(i)}_{best}| \tag{28}$$

5. 実験結果

5.1 ノイズを考慮しない近似計算結果 (調整前)

5.1.1 1 変数関数

1 変数関数の実験結果について述べる。

1 入力 ACU は、2 入力 ACU の片方の入力を 0 にすることで、表現する。近似対象の関数は以下の式であり、初期サンプルが 11 個であり、1 回目の学習で 10 個の SV が得られる。1 回目の学習終了時の SV の数が既に十分に少ないため、この SV を用いた近似を行い、結果を図 22、表 2 に示す。

$$f(x) = x^2 \tag{29}$$

表 2 1 変数関数近似結果 ($f(x) = x^2$)

	After optimisation
# of SVs	10
error(theoretical)	0.00%
error(ACU)	2.64%

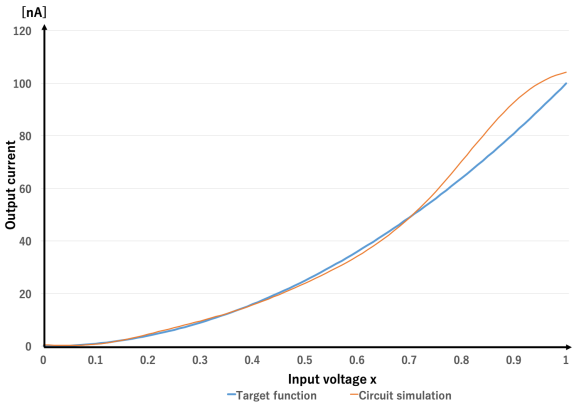


図 22 $f(x) = x^2$ 近似結果

5.1.2 2 変数関数

2 変数関数の実験結果について述べる。
近似対象の関数は以下の式である。

$$f(x_1, x_2) = \sqrt{x_1^2 + x_2^2} \tag{30}$$

近似結果を図 23、表 3 に示す。誤差の増加を抑えながら、SV の数が $\frac{1}{5}$ であることが確認できる。また、誤差も 4.2% であり、回路規模はトランジスタ 600 個である。

表 3 2 変数関数近似結果 ($f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$)

	Original SVR	After optimisation
# of SVs	98	20
error(theoretical)	0.03%	0.25%
error(ACU)	4.10%	4.2%

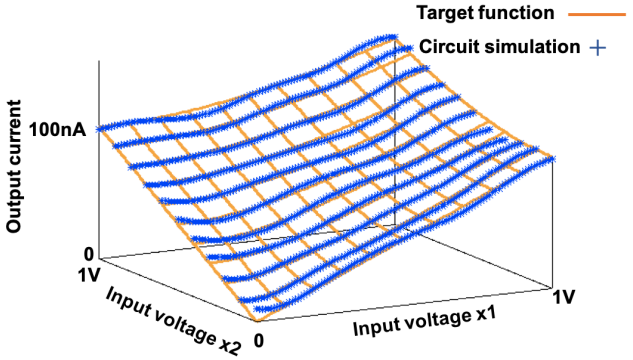


図 23 $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ 近似結果

5.1.3 9 変数関数

9 変数関数の実験結果について述べる。近似対象の関数は以下の式と水平フィルタ、ガウシアンフィルタであり、それぞれの結果を表に示す。

$$f(\mathbb{X}) = \sqrt{x_1^2 + x_2^2 + \cdots + x_8^2 + x_9^2} \tag{31}$$

表 4 9 変数関数近似結果 ($f(\mathbb{X}) = \sqrt{x_1^2 + x_2^2 + \cdots + x_8^2 + x_9^2}$)

	Original SVR	After optimisation
# of SVs	15876	90
error(theoretical)	0.74%	3.13%
error(ACU)	-	10.85%

表 5 9 変数関数近似結果 (水平フィルタ)

	Original SVR	After optimisation
# of SVs	11116	90
error(theoretical)	0.20%	2.33%
error(ACU)	-	6.65%

表 6 9 変数関数近似結果 (ガウシアンフィルタ)

	Original SVR	After optimisation
# of SVs	11254	90
error(theoretical)	0.84%	3.57%
error(ACU)	-	5.36%

5.2 ノイズを考慮した近似計算結果 (調整後)

5.2.1 1 変数関数

1 変数関数の実験結果について述べる。近似対象の関数は以下の式である。

$$f(x) = x^2 \tag{32}$$

結果を表 7、図 24 に示す。ノイズの影響により、誤差が大きくなっている。

表 7 調整後の 1 変数関数近似結果 ($f(x) = x^2$)

	After optimisation
# of SVs	10
error(theoretical)	0.00%
error(ACU)	≤ 7.52%

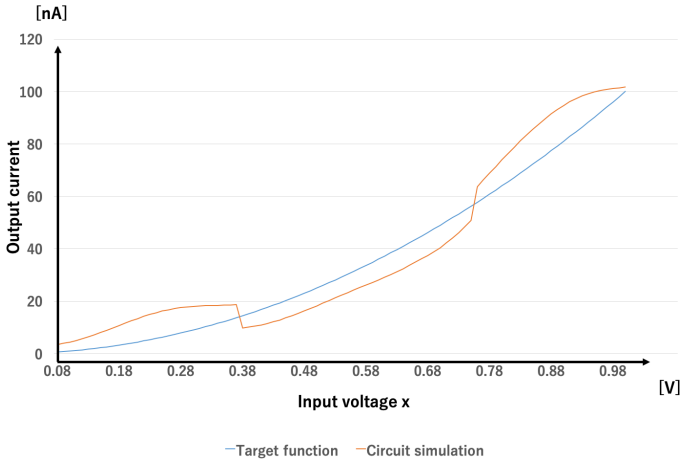


図 24 $f(x) = x^2$ 近似結果

5.2.2 2 変数関数

2 変数関数の実験結果について述べる。近似対象の関数は以下の式である。

$$f(x_1, x_2) = \sqrt{x_1^2 + x_2^2} \tag{33}$$

近似対象の関数から 121 個のサンプルを選び、最初の学習のみの結果と SV の数を減らす処理を終えた後の結果を表 8 に示す。SV の数を 112 個 から 15 個まで減らしたにもかかわらず、誤差は 6.31% から 5.74% であり、誤差の増加を 1% 未満であった。消費電力が 810[nW] であり、演算時間が 330[ns] であることから、消費エネルギーは 267.3[pJ] と概算される。

表 8 調整後の 2 変数関数近似結果 ($f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$)

	Original SVR	after optimisation
# of SVs	112	15
error(theoretical)	0.00%	0.049%
error(ACU)	6.31%	5.74%

6. 考察

本章は、ACU の考察を行う。考察項目は以下の 5 つである。

- 先行研究との比較
- SV の数と精度の関係
- 温度による誤差の影響
- 製造ばらつきによる誤差の影響
- 温度と製造ばらつきを考慮した最適化

最初に、先行研究と比較を行う。比較項目は数値の表現方法、演算器の誤差、演算可能な関数、演算器の入力数、回路規模、演算速度である。次に、結果に用いた二変数関数を用いた考察を行う。カーネル回路による誤差やノイズによる誤差と SV の数の関係を図示し、それぞれの誤差の変化量について考察を行い、最後に、上記の考察を踏まえて、効率の良い演算を行うための最適な SV を決定する。

6.1 他のハードとの比較

表 9 他のハードウェアとの比較

	4-bit ALU	LUT-based analog Calc.	NN regression	ACU(THis work)
Radix	Binary	Analog	Binary	Analog
Bits/Error	4	Error~0.76%	Error ~4.32%	Error~4.2%(~10.85%)
Function	Simple Arithetic	Arbitrary	Transcendental	Arbitrary
Operands	2	1	2	2(9)
# of Tr.s	> 1000	> 30000	> 3000 + CPU	>600(>9000)
Speed	Multiple	Single	Multiple	330[ns]

表 9 に、他の近似演算手法との比較を示す。この比較には、温度と製造ばらつきによる影響を考慮しないものとする。比較対象は、先行研究で述べたハードウェアであり、デジタル演算器による近似手法、アナログ回路を用いた近似手法、NN に関数近似を行う 3 つとの比較である。誤差に関しては、LUT を用いたアナログ演算器が低い値を出しているが、回路規模も大きい。我々の提案手法は、誤差も小さい上に、トランジスタの数も 600 個と非常に少ない結果である。また、入力数を増やすことができ、9 入力に増やした場合の誤差は 10.85%、9000 個のトランジスタによる実装が可能である。

6.2 SV の数と精度の関係

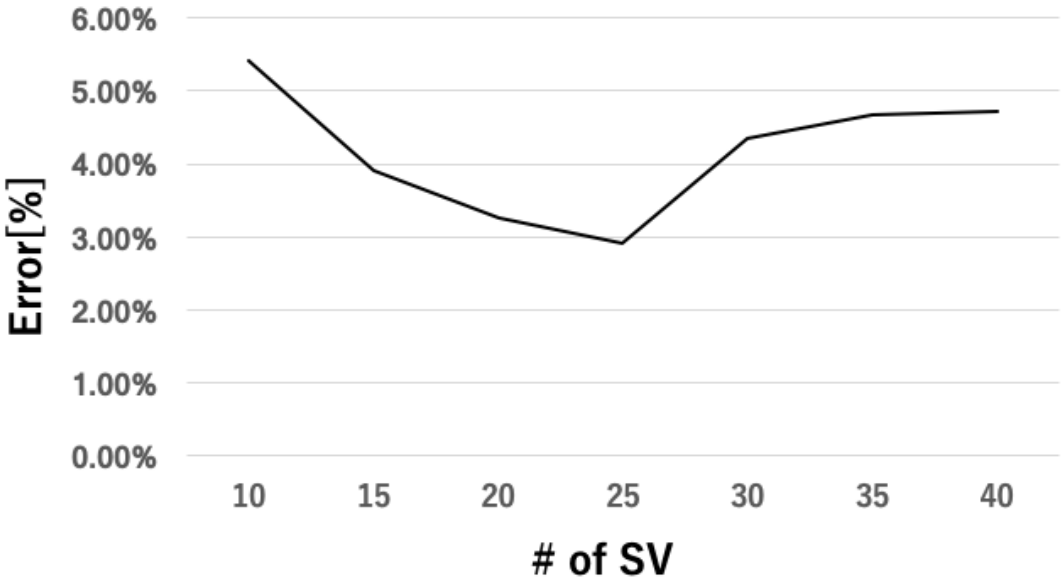


図 25 SV の数と誤差の変化

SV の数と誤差の関係を図 25 に示す。一般的な SVR は、SV の数が多いほど誤差も小さくなるが、ACU は近似ガウシアン関数による近似式であるため、図 25 に示すように、SV の数が 25 個をを超えたあたりから誤差が増えている。そのため、SV の数と精度のトレードオフを考慮すると、SV の数は 15 から 25 の場合が最適である。SV の数によって精度が大きく変わるため、必要となる SV の数とカーネル回路による誤差がどの程度影響を与えているか考慮した上で、近似を行うことが求められる。

6.3 温度による出力変動

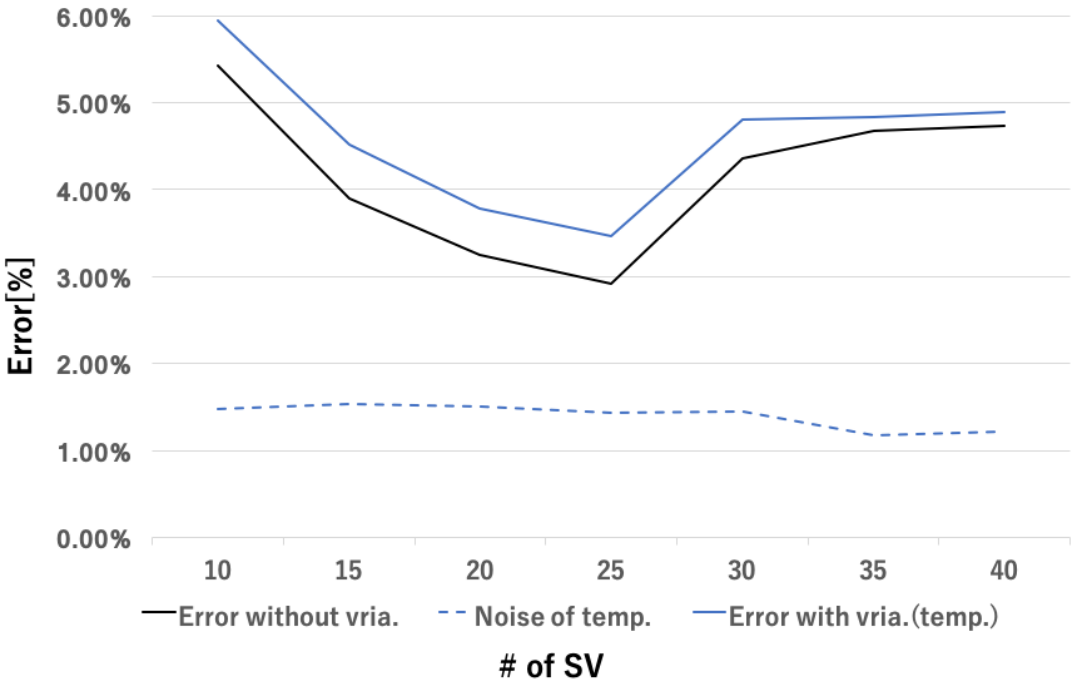


図 26 SV の数と温度の関係

図 26 は、SV の数と温度変化の影響を図示しており、破線が温度変化による出力変動幅である。温度による影響は 1 から 2%と小さく、カーネル回路の MOSFET の高さや幅を調整したことで、影響を小さく抑えることができた。SV の数が変化しても、温度を考慮した誤差と考慮しない誤差の変化は同じような変動をすることから、カーネル回路による誤差は温度による誤差よりも影響が大きいと言える。また、SV < 30 の場合、温度による誤差が小さい。この理由として、二つ考えられる。ガウシアン関数の高さは負の値も取るため、正の高さを持つガウシアン関数と負の値を持つガウシアン関数が互いの誤差を打ち消しあうようなパラメータになっていたことが考えられる。SV の数が多いほど、誤差の打ち消し起こる確率も高くなることから、SV の数が 30 を超えたあたりから、変動幅が小さくなった。

6.4 製造ばらつきによる出力変動

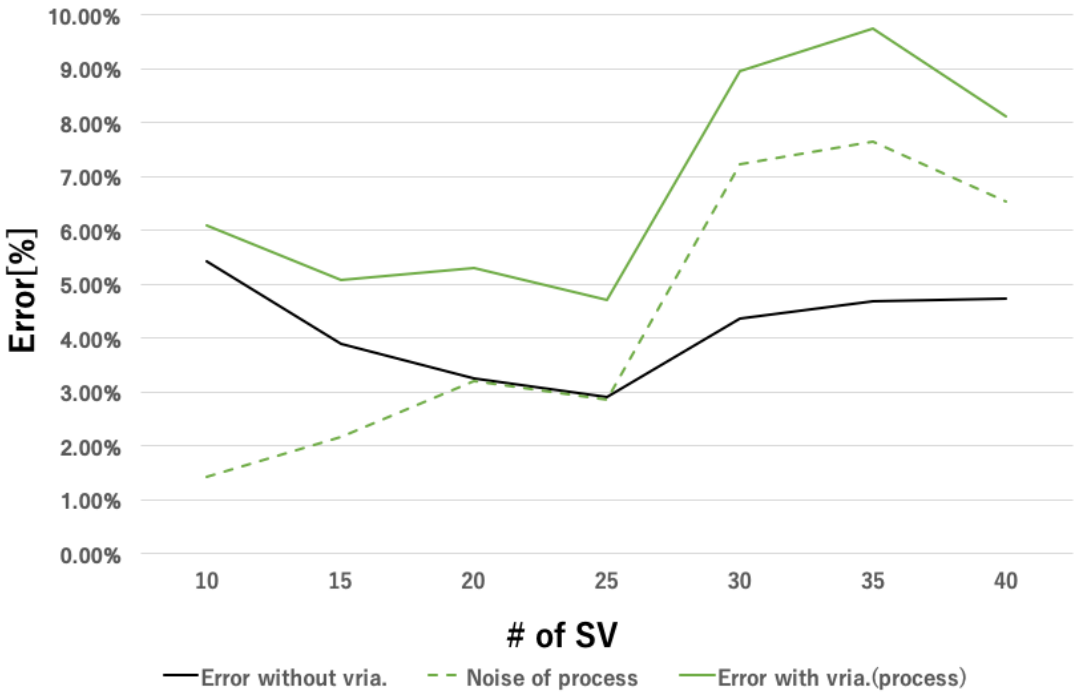


図 27 SV の数と製造ばらつきの関係

図 27 は、SV の数と製造ばらつきによる誤差について図示している。SV の数が増えるのに応じて、破線で図示されている製造ばらつきによる誤差も増加傾向にある。カーネル回路による誤差と製造ばらつきを考慮した誤差を比較すると、製造ばらつきによる誤差がが SV の数が増えるほどに大きくなっているため、最も大きい誤差は約 10%になった。製造ばらつきによる誤差は、ガウシアン関数の高さが大きいほど、製造ばらつきによる誤差も大きくなる傾向にあり、近似式に対する貢献度が高い SV ほど、製造ばらつきによる誤差も大きくなる。

6.5 最適化

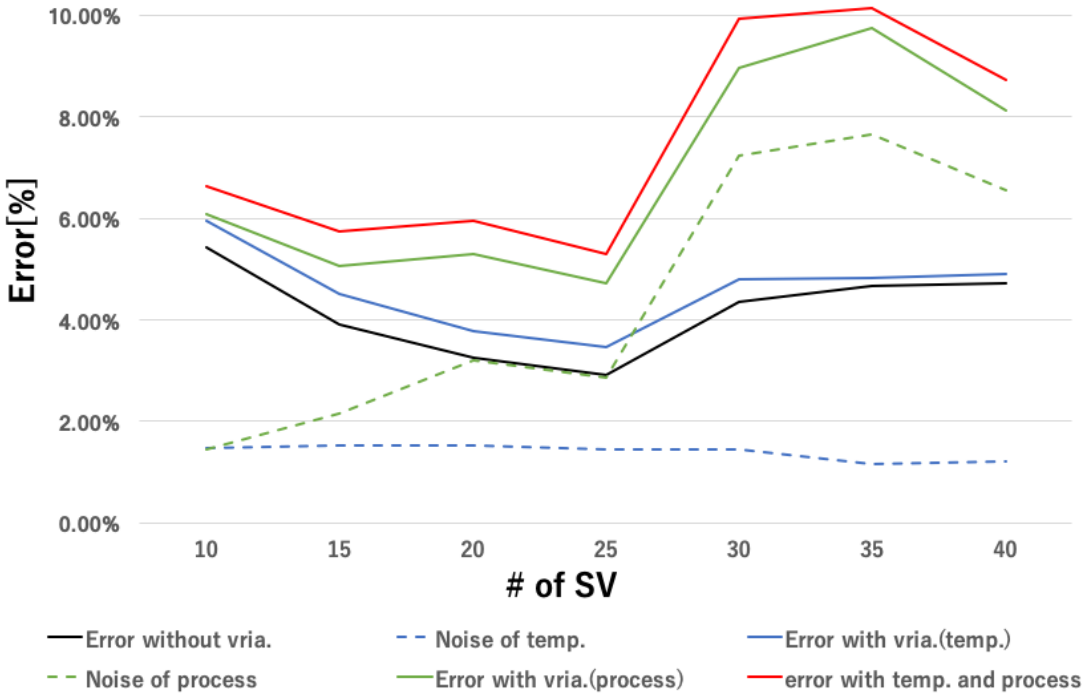


図 28 SV の数と温度と製造ばらつきによるノイズの関係

図 28 は、温度変化と製造ばらつき両方を考慮した誤差が図示されている。製造ばらつきによる誤差の影響が非常に大きく、$SV > 25$ から急激に誤差が大きくなっていることが確認できる。SV の数が増えることで、誤差が大きくなっていることから、この関数においては、$10 < SV < 25$ あたりが最適である。

7. おわりに

本稿では、回帰アルゴリズム SVR を用いたアナログ演算器 ACU の設計及びその評価を行った。ノイズによる性能劣化を考慮しない場合として、1,2,9 変数の関数 $f(x) = x^2$,$f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$, $f(\mathbb{X}) = \sqrt{x_1^2 + x_2^2 + \cdots + x_8^2 + x_9^2}$ を対象とした近似計算を行い、それぞれの誤差が 2.64%, 4, 20%, 10.85% となり、演算時間は 330n 秒であった。小規模実装のために、SV の厳選を行い、演算に用いられた SV の数はそれぞれ 10 個、20 個、90 個となり、回路規模は 1,2 入力計算がトランジスタ 600 個、9 入力計算が 9000 個での演算を可能とした。一方で、この回路の温度と製造ばらつきを考慮した場合、、$f(x) = x^2$, $f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ の近似計算による誤差がそれぞれ 7, 52%, 5.74% となり、ノイズの影響の低減に成功したが、考慮しない結果と比べて、誤差が増えている。評価には、アナログ演算器での課題になると思われる温度と製造ばらつきといった外的要因によって生じる誤差とカーネル回路によって生じる誤差を考慮した上での評価を行い、最適と考えられる SV の数について考察を行った。その結果、$f(x_1, x_2) = \sqrt{x_1^2 + x_2^2}$ において、SV の数が 10 から 25 個が最適であり、誤差は 5.7% と高精度な結果を得ることができた。

謝辞

研究活動から日常生活まで、多大なる御指導をいただいた本学の中島康彦教授に深く感謝の意を表します。本稿をご精読いただき、発表の場においても貴重なご意見をいただいた本学の井上美智子教授に深く感謝いたします。また、日ごろの研究活動におけるご指導はもちろん、本稿執筆をはじめとする多くのご助言を頂いた本学の中田尚准教授に心より御礼申し上げます。研究者としての姿勢や回路の面白さを御教授頂いた張任遠助教に深く感謝いたします。講義において丁寧なご指導ご鞭撻を頂いた TRAN THi Hong 助教に深く感謝いたします。2 年間の研究生活で多くを与えてくださった木村睦客員教授に深く感謝致します。そして、研究活動から日常生活に至るまで 2 年間を共に過ごし、また支えてくださった同輩の菊谷雄真氏、平賀由利亜氏、山根弘樹氏、吉田怜矢氏に深く感謝いたします。就職活動や研究活動をはじめ様々なご助言を頂いた山野龍佑氏、福岡久和氏を始めとするコンピューティング・アーキテクチャ研究室の OB の方々に深く感謝いたします。そして、普段より研究室を賑やかに支えてくださった一倉孝宏氏、池田裕哉氏、岩本淳氏、西本宏樹氏、新谷隆太氏をはじめとするコンピューティング・アーキテクチャ研究室の皆様にはただただ感謝の気持ちでいっぱいです。最後に、この大学院への入学を喜んで認めてくださり、如何なる時も暖かく見守ってくださった家族に感謝いたします。

参考文献

[1] 総務省｜平成 30 年版　情報通信白書｜iot デバイスの急速な普及. http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nd111200.html.

[2] Renyuan Zhang and Tadashi Shibata. Fully parallel self-learning analog support vector machine employing compact gaussian generation circuits. *Japanese Journal of Applied Physics*, 51(4S):04DE10, 2012.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[4] H. Hu, Y. Wen, T. Chua, and X. Li. Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, 2:652–687, 2014.

[5] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, May 2016.

[6] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[7] Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[8] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, June 2011.

[9] Steve R Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[10] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[11] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[12] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. Enerj: Approximate data types for safe and general low-power computation. In *ACM SIGPLAN Notices*, volume 46, pages 164–174. ACM, 2011.

[13] Ravi Nair. Big data needs approximate computing: Technical perspective. *Commun. ACM*, 58(1):104–104, December 2014.

[14] Y. Wu, W. AdbAlmageed, S. Rawls, and P. Natarajan. Computationally efficient template-based face recognition. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1424–1429, Dec 2016.

[15] F. Li, A. Wong, and J. Zelek. Hierarchical grouping approach for fast approximate rgb-d scene flow. In *2016 13th Conference on Computer and Robot Vision (CRV)*, pages 140–147, June 2016.

[16] 空道 穐山 and 崇宏 広渕. 性能カウンタを用いた近似実行の高速なエミュレーション. Technical Report 12, 国立研究開発法人産業技術総合研究所, 国立研究開発法人産業技術総合研究所, jul 2017.

[17] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Comput. Surv.*, 48(4):62:1–62:33, March 2016.

[18] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Design Automation Conference*, pages 555–560, June 2010.

[19] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan. Approximate computing: An integrated hardware approach. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 111–117, Nov 2013.

[20] Priyanka Yadav, Gaurav Kumar, and Sumita Gupta. Design and implementation of 4-bit arithmetic and logic unit chip with the constraint of power consumption. 2014.

[21] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan. Analog computing in a modern context: A linear algebra accelerator case study. *IEEE Micro*, 37(3):30–38, 2017.

[22] J. Hasler. Opportunities in physical computing driven by analog realization. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, Oct 2016.

[23] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M. Seok, S. Sethumadhavan, and Y. Tsividis. Energy-efficient hybrid analog/digital approximate computation in continuous time. *IEEE Journal of Solid-State Circuits*, 51(7):1514–1524, July 2016.

[24] Schuyler Eldridge, Florian Raudies, David Zou, and Ajay Joshi. Neural network-based accelerators for transcendental function approximation. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '14, pages 169–174, New York, NY, USA, 2014. ACM.

[25] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[26] Chih-Chung Chang and Chih-Jen Lin. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9):2119–2147, 2001.

業績

1. Noriyuki UETAKE, Renyuan ZHANG, Takashi NAKADA, Yasuhiko NAKASHIMA: ”A programmable analog calculation unit for vector computations” ,COOL CHIPS , April. (2018)

2. R. Zhang and N. Uetake and T. Nakada and Y. Nakashima: ”Design of Programmable Analog Calculation Unit by Implementing Support Vector Regression for Approximate Computing”, IEEE Micro,pp.73-82, Nov.(2018)