特集
Special Feature

［知能コンピューティング― AI とハードウェアの出会い―］

2 確率的コンピューティングの再開拓
―その場学習が可能な極低電力エッジ AI に向けて―

浅井哲也　北海道大学大学院情報科学研究院

確率的コンピューティングとは

確率的コンピューティングは，行いたい演算を確率事象の演算に対応させる相似計算の 1 つであり，その源流はなんと 1960 年代に提唱されている[1]．通常のデジタル演算回路と比較して，きわめて小規模の論理ゲート等により演算が可能であるが，できる演算が限られている（多くはアルゴリズム上の工夫が必要である）．また，高精度演算にも不向きである．そのため，比較的単純な演算の繰り返しかつ高い演算精度が求められない「人工知能の演算」に適用することで，その低電力化や低リソース化に貢献する可能性がありそうだ．本稿では，その具体例とともに，問題点と可能性を洗い出してみたい．

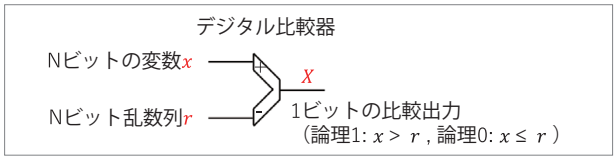
確率的コンピューティングを行うためには，演算したい数値をバイナリビットストリームに変換（エンコード）し，必要な演算（人工知能の演算では，乗算，積和演算，微分，非線形変換など）を行い，その演算が完了したら演算結果（これもまたバイナリビットストリーム）をもとの数値表現に戻す（デコードする）必要がある．本稿ではまずその手順から解説する．

数値をバイナリビットストリームに変換する「エンコード」

図 -1 に示すようなデジタル回路により，変換したい変数x（N ビット，正の値）と，それと同じビット数の一様乱数列r（線形帰還シフトレジスタ（LFSR）などでつくった疑似乱数）を用意し，それらの大小をデジタル比較器で比較する．ここで，変換したい N ビット変数xと乱数列rを 1 に規格化した値変数をそれぞれ$\bar{x}$ ($\equiv x/2^N$)，$\bar{r}$ ($\equiv r/2^N$) で表すと，この比較器の出力 X は確率 $\bar{x}$ で論理 1 となる（確率$\bar{r}$ $=1-\bar{x}$で出力が論理 0 となる）バイナリビットストリームとなる．たとえば，$\bar{x}$が 0.1 であれば，$\bar{r}$は確率 0.9 で 0.1 よりも大きい値となることから，確率 0.1 で X は論理 1 となる．この変換（変数xと乱数列rの比較）を，ここでは「エンコード」と呼ぶ．数値xを確率的コンピューティングの世界で扱うバイナリビットストリーム X（確率$\bar{x}$で論理 1 となる時系列）にエンコードすることが，確率的コンピューティングを行うための入口（最初の準備）となる．

バイナリビットストリームを数値に変換する「デコード」

図 -1 の比較器の出力 X は一定値ではなく，疑似乱数の値を更新するたびに変化するバイナリビットストリームである（ただし，$\bar{x}=0, 1$ の場合を除く）．この乱数の更新を 2^N 回行うとし，その都度 X の値



■図 -1　変数のエンコード方法

を観測する．この観測値のうち，X の値が論理 1 であった回数を $N_{X=1}$ とすれば，$N_{X=1}$ は x の値（$N_{X=1}/2^N$ は $\bar{x}$ の値）を表す．N の値が大きければ $N_{X=1}/2^N$ は $\bar{x}$ に漸近する．この観測行為（$N_{X=1}$ のカウントおよび $N_{X=1}/2^N$ の計算）をここでは「デコード」と呼ぶ．通常は，カウンタ回路を用いてバイナリビットストリーム X 中の論理 1 のビット数をカウントして，もとの（通常の計算を行う）世界の変数値 $\bar{x}$ にデコードする．確率的コンピューティングで何らかの演算が行われ，その結果がバイナリビットストリーム X で表されているとき，それを取り出すには上記のデコード（通常の世界に演算結果を取り出すための出口）が必要である．

確率的コンピューティングのオーバヘッド

確率的コンピューティングを行っている途中で，確率的コンピューティングではどうしても実行できない演算があったとする．この場合，その直前の演算結果を一度デコードした後，通常のデジタル回路を用いて演算し，それをまたエンコードすれば確率的コンピューティングの演算を続けることができる．しかし，エンコード，デコードにはそれぞれ比較器，カウンタが必要であり，これらが確率的コンピューティングのオーバヘッドとなるため，エンコード，デコード（を行う回路）の数を極力減らす必要がある．そのためには，なるべく多くの演算を確率的コンピューティングで行うよう工夫しなければならない．さらに，メモリやレジスタが必要な演算では，それらの値自体もバイナリビットストリームで表現されなければならない．

確率的コンピューティングの基礎

二変数の確率的コンピューティングの例をいくつか説明する．なお，確率的コンピューティングにおいては，各変数のエンコードで使う乱数列は互いに独立である必要があることに注意されたい[1)]．

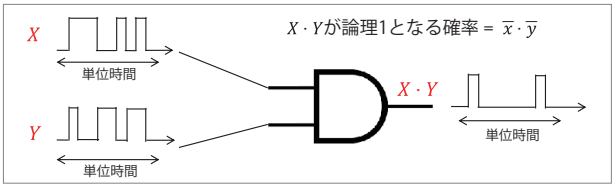
乗算

x と y がエンコードされたバイナリビットストリーム X と Y は，それぞれ確率 $\bar{x}$ と $\bar{y}$ で論理 1 を出力している．ここで，X と Y の論理積を考える．X と Y が同時に論理 1 を出力したときにのみ X と Y の論理積が論理 1 となる（図 -2）．そうなる確率は $\bar{x}$ と $\bar{y}$ の積で表される（確率論で言うところの積事象）．よって，X と Y の論理積をデコードすると，その値は $\bar{x}\cdot\bar{y}$（乗算）を表すことになる．これが確率的コンピューティングの乗算の基本的な仕組みであり，行いたい乗算を確率の積事象の演算に対応（相似）させたものである．

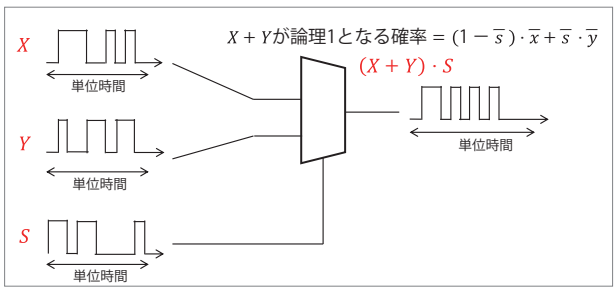
重み付き加算

確率は 0 から 1 の連続値で表されるため，確率的コンピューティングでは扱う値の最小値，最大値がそれぞれ 0，1 となるような規格化が行われる．このような規格化を伴う加算は，バイナリビットストリーム X または Y を確率的に選択して出力する行為により実現する．

まず，確率 $\bar{s}$ で論理 1 となる信号 S を用意する（図 -3）．この S も x，y のエンコードで使う乱数列



■図 -2　確率的コンピューティングによる乗算



■図 -3　確率的コンピューティングによる重み付き加算

とは別の乱数列で生成しなければならない．選択信号 S が論理 0 で X，論理 1 で Y が出力されるようなマルチプレクサを考えると，出力は $(1-\bar{s})\cdot\bar{x}+\bar{s}\cdot\bar{y}$ となる．このような演算を重み付き加算と呼ぶ．$\bar{s}=0.5$ となるような選択信号を用いると出力は $(\bar{x}+\bar{y})\cdot 0.5$ となり，これは x と y の和が 1 に規格化される．

不完全加算

図 -2（乗算）で導入した論理積を論理和で置き換えると，「不完全加算」と呼ばれる非線形演算が行われる（$\bar{x}+\bar{y}-\bar{x}\cdot\bar{y}$）（図 -4）．これは，$\bar{x}$, $\bar{y}$ の値が小さい（$\bar{x}, \bar{y}\ll 1$）ときはそれらの加算値で近似される．

負の値の表現

確率的コンピューティングでは，負の値を直接的に表現できない．そのため，正負の演算が必要な場合は差動表現が用いられる．たとえば，正と負の値を取り得るニューラルネットワークの 1 つの重み w を，w^+ と w^- $(0\le w^+, w^-\le 1)$ の 2 つの変数の差 $(w^+-w\_)$ で表す．この方法は，古くはアナログ電子回路（現在は不揮発メモリ素子のクロスバー構造）によるニューラルネットワークの積和演算で用いられており，確率的コンピューティングにおいてもしばしば用いられる．

その他の演算

$\bar{x}$ の論理反転を行うと，$1-\bar{x}$ といった演算ができる．さらに，排他的論理和やフリップフロップを組み合わせたより複雑な演算（除算や積分など）も提案されている．詳細説明は文献 1）に譲る．

確率的コンピューティングとニューラルネット

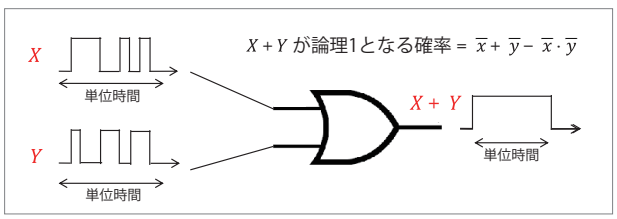
上述した確率的コンピューティングの基本演算を少し拡張すると，人工知能（脳の神経回路の一部を模した数理モデルであるニューラルネットワーク）の演算ができるようになる．以下にその要素を説明する．

活性化関数

ニューラルネットワークの活性化関数 $(\equiv g(x))$ には，シグモイド関数 $(g(x)=(1+e^{-x})^{-1})$ や ReLU $(g(x)=\max(0, x))$ などの非線形関数が用いられる．文献 2）では，$g(x)=\tanh(x)$ の活性化関数が確率的コンピューティングにより実現されているが，その必要リソースはきわめて大きい．ここでは，より簡便な方法でシグモイド関数に似た非線形関数を確率的コンピューティングに導入する方法を紹介する [3), 4)]．図 -4 において，不完全加算（$\bar{x}+\bar{y}-\bar{x}\cdot\bar{y}$）の仕組みを説明した．ここでもし，$\bar{x}=\bar{y}$ であれば，不完全加算の結果は $2\bar{x}-\bar{x}^2$ となる．先に説明した乗算を用いて，ある乱数列で作られた $2\bar{x}-\bar{x}^2$ と，別の乱数列で作られた $2\bar{x}-\bar{x}^2$ の積を計算すると，$(2\bar{x}-\bar{x}^2)^2$ が得られる．この演算を n 回繰り返せば演算結果は $(2\bar{x}-\bar{x}^2)^{2n}$ となり，強い非線形性を持つ関数が得られる．図 -5 に n=1 の例を示す．シグモイド関数と似た非線形関数となる．なお，図中の D-FF は，バイナリビットストリームを 1 クロック遅延させて独立した系列にするためのものである．

短絡抑制による減算相当演算

ニューラルネットワークの演算において必要となる「減算」をより簡便な方法で置き換える手法を紹介する [3), 4)]．ある正の入力 X を与えたことによりニューロン（神経細胞）が電位を持ち，その電位が



■図 -4　確率的コンピューティングによる不完全加算

維持されているとする．この電位を減らすためには，負の入力 Y をニューロンに与える方法（減算（$X-Y$）に相当），または別の入力 Y により電位の基となる電荷をニューロンの外へ逃がす方法（「短絡」により電気的な逃げ道を作ることに相当），の２通りがある．後者の方法でニューロンの電位を減らすことを「短絡抑制」と呼ぶ[5]．そこでは，$X(1-Y)$ のような乗算と反転の演算により電位の抑制が行われる．Y の値が小さければ X の値はほぼ X であり，Y の値が大きくなれば（1 に近づけば）$X(1-Y)$ の値は 0 に近づくため，定性的には $X(1-Y)$ と $X-Y$ は同じ性質を持つ．確率的コンピューティングでは，$X(1-Y)$ を Y の反転（$1-Y$），およびその値と X の論理積により計算できるため都合がよい．この方法でニューラルネットの順伝搬演算におけるすべての減算を表現し，逆伝搬演算についても（デコードと同時に行われる重みの更新方向を決定するための減算を除いた）すべての減算がこの演算で置き換え可能である[3], [4]．

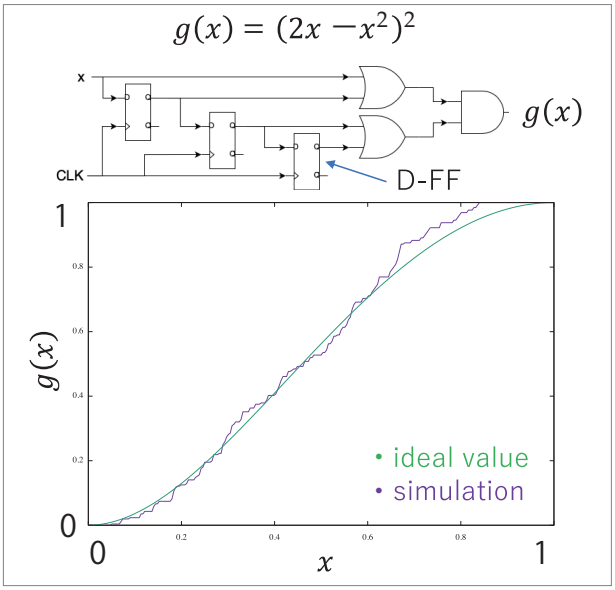
積和演算と活性化

上述した重み付き加算を用いれば，ニューラルネットの積和演算における和の計算（総和演算）が可能である．その総和値を活性化関数に与えればニューロンの出力が得られるが，重み付き加算では，その演算結果が総入力数で規格化されてしまうため，入力数に応じて活性化関数のしきい値を変更しなければならない．そもそも，傾きがシグモイド関数で活性化されることを前提とした総和演算の結果は，その値がシグモイド関数のしきい値よりも大きい場合は，正確な値でなくても良いと考えられる（シグモイド関数の入力値が大きい場合は，出力が 1 に貼りつく傾向になるため）．そのため，総和演算は，不完全加算（小さな値の加算は正確，大きな値の加算は不正確）でも十分かもしれない，という考えに至る．

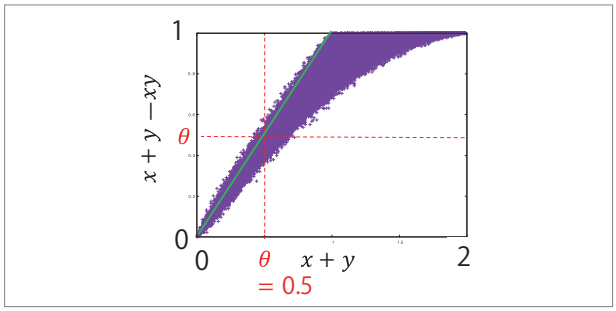
不完全加算による演算結果を，同じ入力を与えた際の単純加算と比較した結果を図 -6 に示す．不完全加算では全体として精度が落ちるが，入力値 x，y がともに小さい場合や，活性化関数のしきい値（約 0.5）付近では，演算精度はさほど低くない．よって，活性化関数への入力を前提とした演算では重み付き加算よりも不完全加算が適していると考えられる[3], [4]．

多層パーセプトロン の実装

多層パーセプトロンとは，順伝搬型ニューラルネットワークの基本形である単純パーセプトロン（複数の入力データを重み付け加算した値を活性化した，１つの値を出力する関数）を多出力化・多層化したものであり，人工知能の標準形とも言われる．ここでは，上述した手法を用いて，図 -7 に示す構



■図 -5　活性化関数の例



■図 -6　不完全加算と通常の加算の誤差

造の三層パーセプトロンの順伝搬および逆伝搬演算を実装する．活性化関数を $g(x)$，不完全加算による総和演算を $OR\sum$ と定義すると，入力総から中間層への順伝搬の演算は，

$$h^{+}_{V_j} = OR\sum_{k=0}^{N_X-1} w^{+}_{j,k}X_k \quad \text{（積の不完全加算）}$$
$$h^{-}_{V_j} = OR\sum_{k=0}^{N_X-1} w^{-}_{j,k}X_k \quad \text{（積の不完全加算）}$$
$$V_j = g\left(h^{+}_{V_j}\left(1-h^{-}_{V_j}\right)\right) \quad \text{（短絡抑制と中間層の活性化）}$$

で表される．ここで，$h^{+}_{V_j}$ と $h^{-}_{V_j}$ が h_{V_j} の差動表現となっており，それらが短絡抑制されて（$h^{+}_{V_j} - h^{-}_{V_j}$ と定性的に同じものとして）活性化関数に与えられる．中間層から出力層への順伝搬は，同様に

$$h^{+}_{Y_i} = OR\sum_{j=0}^{N_V-1} W^{+}_{i,j}V_j \quad \text{（積の不完全加算）}$$
$$h^{-}_{Y_i} = OR\sum_{j=0}^{N_V-1} W^{-}_{i,j}V_j \quad \text{（積の不完全加算）}$$
$$Y_i = g\left(h^{+}_{Y_i}(1-h^{-}_{Y_i})\right) \quad \text{（短絡抑制と出力総の活性化）}$$

となる．次に，誤差関数を

$$E_i = \frac{1}{2}\sum_{i=0}^{N_Y-1}(Y_i - Z_i)^2$$

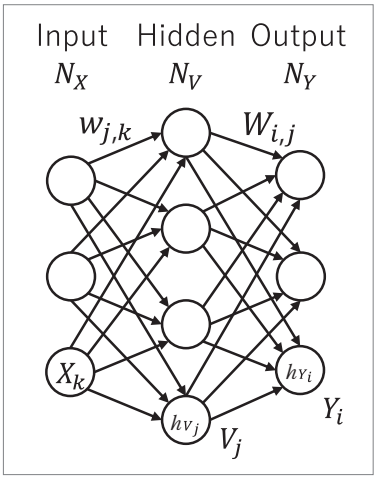
と定義する．ここで，Z_i は教師ラベルである．すると，通常の勾配降下法による各重みの更新量は，

$$\Delta W^{+}_{i,j} = -\eta\frac{\partial E_i}{\partial W^{+}_{i,j}}, \quad \Delta W^{-}_{i,j} = -\eta\frac{\partial E_i}{\partial W^{-}_{i,j}}$$
$$\Delta w^{+}_{j,k} = -\eta\sum_{i=0}^{N_Y-1}\frac{\partial E_i}{\partial w^{+}_{j,k}}, \quad \Delta w^{-}_{j,k} = -\eta\sum_{i=0}^{N_Y-1}\frac{\partial E_i}{\partial w^{-}_{j,k}}$$

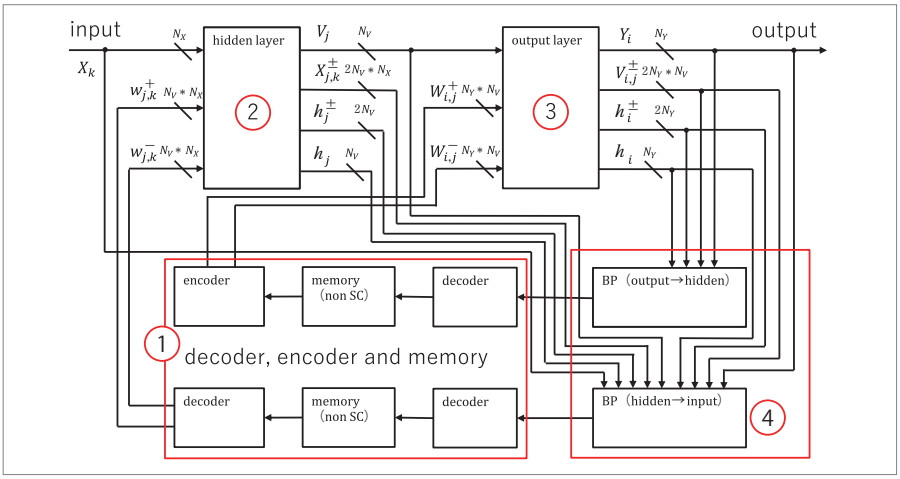
となる．これらを展開して確率的コンピューティングでその値を計算することにより，各重みの更新量を得る[3), 4)]．

図 -7 の構造の順伝搬と逆伝搬の演算を行うブロック図を図 -8 に示す．①のブロックは，確率的コンピューティングで扱うことができないメモリブロックであり，重み（$w^{\pm}_{j,k}$，$W^{\pm}_{i,j}$）を保持する通常のメモリ回路を使うために，エンコード・デコードを行うための回路ブロック（encoder, decoder）が挟まれている．②と③のブロックは，それぞれ中間層，出力層における積和演算と活性化の演算を行うブロック，④は重み更新量（$\Delta w^{\pm}_{j,k}$，$\Delta W^{\pm}_{i,j}$）を算出するブロックである．①のブロックを「確率的メモリ☆1」で丸ごと置き換えることができれば，すべての演算が確率的コンピューティングで完結するため，省リソース・低電力かつ推論とその場学習（オンライン学習）が可能なエッジ向け人工知能ハードウェアが構築できる可能性がある．現在のエッジ向け AI チップは推論専用であり（学習はその場（エッジ）ではなくクラウドで行われる），その場（エッジ）で学習ができ

☆1 インクリメンタルな更新が可能（自身の記憶値を少し増やす，または減らすことができる），かつその出力が自身の記憶値に相当する確率で論理 1 となるビットストリームを出力するメモリ．真正乱数発生回路とアナログ不揮発メモリ素子の組合せにより実現できる可能があるが，筆者が調査した範囲では現時点でそのような「確率的メモリ」は存在しないようである．確率的コンピューティングではすべてのメモリ要素に同時にアクセスする必要があるため，現在のメモリアレイのような構造は好ましくない．確率コンピューティングのための各論理回路のすぐそばに必要な確率的メモリが置かれる「インメモリコンピューティング」アーキテクチャとなる．



■図 -7　三層パーセプトロンとパラメータ



■図 -8　順伝搬と逆伝搬演算のブロック図

るようになれば，ユーザや環境などへの適応や，ユーザ自身による AI の教育が可能となることから，多くの新しいアプリケーションが生まれるだろう．

性能評価

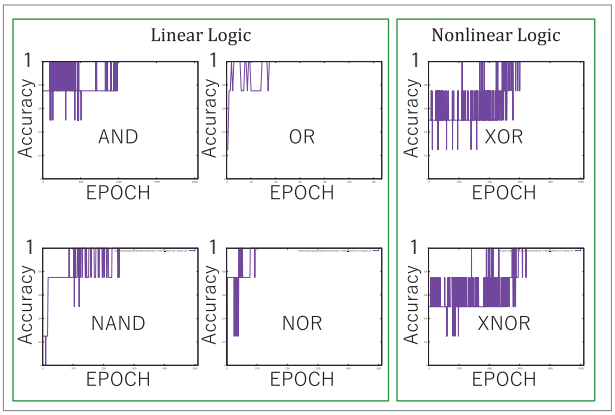
上述した手法を用いて図 -7 に示す三層パーセプトロンを実装し，評価した例をいくつか紹介する．

論理関数の学習

最も簡単な例として，二入力の論理関数を学習させた例を紹介する（N_X=3,N_V=4,N_Y=1,η=0.3）．図 -9（横軸は学習回数，縦軸は精度（× 100 %））に示すように，AND や OR といった線形分離可能な論理演算だけでなく，XOR や XNOR のような線形分離不可能な論理演算についても数百回程度の更新で正しく学習できている．

線形回帰問題

次に 8 ビットの線形回帰問題の例を紹介する（N_X=9,N_V=32,N_Y=8,η=0.3）．100 通りの数値に対して出力 Y= 教師 X，および出力 Y= 教師 $X/2$ として 4,069 回のオンライン学習を行い，256 通りの入力に対して推論を行った結果を図 -10（横軸は入力値，縦軸は出力層の出力値（バイナリ）を数値に変換したもの）に示す．この例でも学習は問題なく完了している．



■図 -9　基本的な論理関数の学習結果

非線形回帰問題

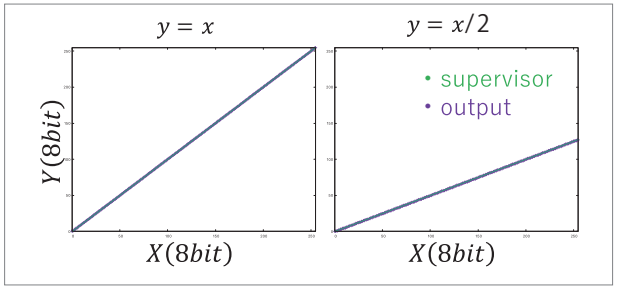
次に 8 ビットの非線形回帰問題の例を紹介する（N_X=9,N_V=128,N_Y=8,η=0.3）．256 通りの数値に対して出力 Y= cos(X)，および出力 Y= sin(X) として 4,069 回のオンライン学習を行い，25 通りの入力に対して推論を行った結果を図 -11（横軸は入力値，縦軸は出力層の出力値（バイナリ）を数値に変換したもの）に示す．ここでも学習は問題なく完了している．

縮小 MNIST の評価

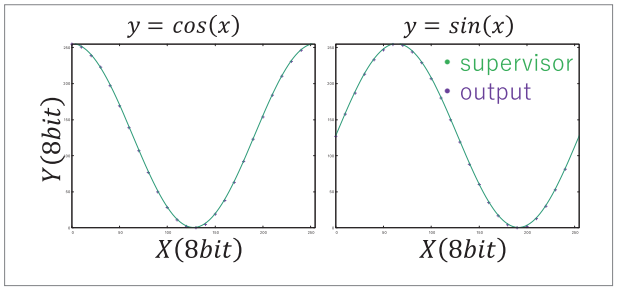
手書き文字のデータセットを小規模化した縮小 MNIST[6] を用いた評価の一例を紹介する（N_X=197,N_V=128,N_Y=10,η=0.3）．58,078 個の手書き文字データとラベルを用いてオンライン学習を行い，9,534 個のテストデータを用いて精度を評価したところ，およそ 80% 程度の正解率を示している（図 -12：横軸は学習回数，縦軸は精度（× 100 %））．オンライン学習であるため，正解率は低めとなる．

消費電力の見積り

このアーキテクチャの消費電力について紹介する．前節の縮小 MNIST の評価を行った場合と同じ



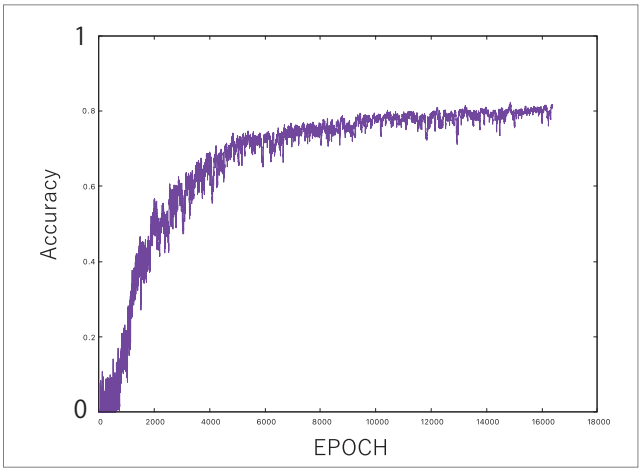
■図 -10　線形回帰問題の推論例



■図 -11　非線形回帰問題の推論例

ネットワーク構造 (N_X=197,N_Y=128,N_Y=10) を想定した．不完全加算を行う回路は超多段の OR ゲートとなるため，多数決ゲートにより構成し，その電力が NAND2 相当であると仮定すると，消費電力はおおむね表 -1 のように計算できる．

少々古いプロセス（0.18um プロセス）のスタンダードセルライブラリを用いて論理合成した結果より算出された値ではあるが，この値をもとに，先端プロセスを用いた場合の消費電力を（スケーリング則により）ラフに見積ることも可能である（例：28nm プロセス，250MHz 動作，1.1V コア電源における推論動作の消費電力は約 3.3 mW，学習動作の消費電力は約 24.6mW，等）．これから確率的コンピューティングの集積 AI に取り組んでみたいという方々の参考値となれば幸いである．



■図 -12　縮小 MNIST の学習（精度変化）の様子

■表 -1　推論時および学習時の消費電力（推定）

Software	Software (without decoder, encoder and memory)	
Technology	UMC 0.18-μm 1P6M CMOS	
Function	training	inference
Network Configuration	197-64-10	
Supply Voltage	1.8 V	
Frequency	100 MHz	
TOPS	2.0*	0.02*
Dynamic power	57.4 mW	4.45 mW
Static power	32.9 mW	7.66 mW
Total power	90.3 mW	12.1 mW

* Stream length = 256 (10P = 256 stochastic operation)

残る課題

確率的コンピューティング技術の概要，およびその技術を用いたニューラルネットワークの実装例を紹介した．ニューラルネットワークの一部の演算を確率的コンピューティングで置換する先駆的アプローチ[2]に続き，本稿で紹介した技術は，メモリ（重みの記憶）機能以外のすべての演算（推論・学習）を確率的コンピューティングで置換できるものであり，確率的コンピューティングに基づく「集積可能な」人工知能チップの実現へあと一歩というところまで近づいたものと思われる．本稿でも述べたが，自身の記憶値に相当する確率でビットストリームを出力する「確率的メモリ」の実現が確率的コンピューティングに基づく人工知能集積回路の最後の砦であり，その攻略に今後も注力したい．

参考文献

1）Gaines, B. R. : Stochastic Computing Systems, Advances in Information Systems Science, pp.37-172, Springer (1969).
2）Sato, S., Nemoto, K., Akimoto, S., Kinjo, M. and Nakajima, K. : Implementation of A New Neurochip Using Stochastic Logic, IEEE Trans. Neural Networks, Vol.14, No.5, pp,1122-1127 (2003).
3）浅井哲也，西田浩平，佐々木義明：学習装置，減算回路および活性化関数回路，特願 2021-118326（2021 年 7 月 16 日）．
4）Sasaki, Y., Muramatsu, S., Nishida, K., Akai-Kasaya, M. and Asai, T. : Digital Implementation of A Multilayer Perceptron Based on Stochastic Computing with Online Learning Function, NOLTA, IEICE, Vol.13, in press.
5）Asai, T., Kanazawa, Y. and Amemiya, Y. : A Subthreshold MOS Neuron Circuit Based on The Volterra system, IEEE Trans., Neural Networks, Vol.14, No.5, pp.1308-1312 (2003).
6）Mochida, R., and Kohno, K., et al. : A 4M Synapses Integrated Analog ReRAM Based 66.5 TOPS/W Neural-network Processor with Cell Current Controlled Writing and Flexible Network Architecture, in Proc. 2018 IEEE Symposium on VLSI Technology, pp.175-176 (2018).

（2021 年 12 月 13 日受付）

■浅井哲也　asai@ist.hokudai.ac.jp

1999 年豊橋技術科学大学博士課程修了．博士（工学）．1999 ～ 2001 年同大助手，2001 ～ 2015 年北海道大学准教授．2016 年から同大教授．