混沌系工学特論 配布資料 #9 (最終回)

担当：井上 純一 (情報科学研究科棟 8-13)
URL：http://chaosweb.complex.eng.hokudai.ac.jp/˜j _inoue/

平成 17 年 1 月 31 日

目 次

3.10 学習レートの最適化と学習曲線

3.10.1 教師機械の性能に関する補足

我々がここで考えてきた教師機械 $K = 3$ パリティ・マシン (非単調パーセプトロン) の性能として, 全ての可能な例題数 2^N のうちのいくつを実現できるのか, という記憶容量を基準にとり, この基準で生徒機械である単純パーセプトロンとの比較をする場合, 教師機械の表現能力が生徒機械のそれよりも高く, その結果として我々が考えてきたオンライン学習は「実現不可能な規則の学習」となった. ここでは, その容量がパラメータ a の関数としてどのように振舞うのか, を具体的に見ておきたい. 以下の結果を導くためにはレプリカ法を用いなければならず, ここでその詳細まで立ち入って述べる余裕はないので, 結果のみを示しておく. 我々の教師機械の実現できる最大例題数 $\alpha_c = P/N$ は a の関数として次式で与えられる.

$$\alpha_c(a) \quad = \quad \left[\int_0^{a/2} Dz\, z^2 + \int_{a/2}^{\infty} Dz\, (a-z)^2 \right]^{-1} \tag{81}$$

ここに, $Dz = dz\, \mathrm{e}^{-z^2/2}/\sqrt{2\pi}$ である. 従って, この (81) 式を a の値を変えてプロットしていけばよい. それを図 16 に載せよう. この図で, $a \to \infty$ の極限が単純パーセプトロンの記憶容量に相当することに注意する. この図より, 我々が教師として用いていた学習機械 $a < \infty$ の実現できる記憶容量は既に述べていた通り, 生徒機械である単純パーセプトロンのそれよりもかなり大きい. よって, ここでまで調べていた学習は実現不可能な規則の学習ということになるのである.

3.10.2 学習レートの導入

我々は前回, 学習則の一つとしてパーセプトロン学習：

$$\boldsymbol{J}^{m+1} \quad = \quad \boldsymbol{J}^m - \Theta(-T_a(v)S(u))S(u)\boldsymbol{x} \tag{82}$$

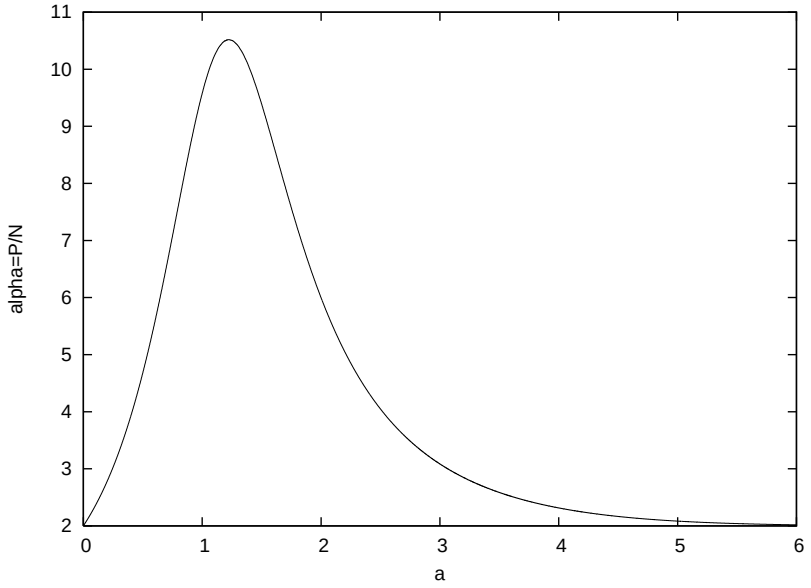

図 16: 我々が調べていた教師機械の実現できる最大パターン数 $\alpha_c = P/N$ のパラメータ a 依存性. 生徒機械のそれは $a \to \infty$ での $\alpha_c = 2$ に相当する.

を取り上げ, その学習過程のダイナミックスを調べてきた. ここで, 上記で与えられる結合 $\boldsymbol{J}$ についての更新則において, 入力 $\boldsymbol{x}$ を足しあげるときの重みは常に 1 であった. しかし, 例えば, 学習の初期の段階ではある程度大きな重みを用いて入力ベクトル $\boldsymbol{x}$ を結合ベクトル $\boldsymbol{J}$ に足しこみ, 解が見えてきた段階で結合の更新を微調整する, つまり, 比較的小さな重みで入力を加えていく, というように, 学習のステップ数 α に応じて, 適応的に上記学習式 (82) の更新重みを変更していけば, 場合によっては学習が加速することが予想される. そこで, この学習ステップに依存する更新重み, つまり, **学習レート**を $g(\alpha)$ と置いてみることにする. すると, パーセプトロン学習の更新式 (82) は次のように書ける.

$$\boldsymbol{J}^{m+1} \quad = \quad \boldsymbol{J}^m - g(\alpha)\,\Theta(-T_a(v)S(u))S(u)\boldsymbol{x} \tag{83}$$

ここで, $g(\alpha)$ にどのような更新ステップ α 依存性を持たせるか, が学習の加速についての決め手になるのではあるが, まずは前回, パーセプトロン学習のところで行った同様の手続きにより, このミクロな方程式 (83) からマクロな量である, R, l に関する状態更新式を導出しよう. すると直ちに, 学習レート $g(\alpha)$ を用いて発展方程式が

$$\frac{dR}{d\alpha} \quad = \quad \frac{1}{l^2}\left[-\frac{R}{2}g(\alpha)^2 E_a(R) + g(\alpha)[F_a(R)R - G_a(R)]l\right] \equiv L(g(\alpha)) \tag{84}$$

$$\frac{dl}{d\alpha} \quad = \quad \frac{1}{l}\left[\frac{g(\alpha)^2 E_a(R)}{2} - g(\alpha)F_a(R)l\right] \tag{85}$$

のように書ける. ここで, $E_a(R)$ 等は, 前回の講義ノート ([**講義ノート** #8]) で既に定義されたものであることに注意されたい. 従って, この微分方程式の中に現れる学習レート $g(\alpha)$ の学習ステップ依存性を具体的に与えれば, そのレートによって学習過程がどの程度まで加速されたのか, あるいは加速されないのか, を調べることができる. しかし, このように ad hoc に学習レートを決めるのではなく, 次に示すように, 「汎化誤差の減少率を最大化する」という観点でこの学習レートを最適化することができる. それを次に見ていくことにしょう.

3.10.3　学習レートの最適化

前回見たように, パラメータ a が $a > a_{c2} = 0.80$ を満たす領域では, 教師, 生徒の結合の重なり R を $R = 1$ に導くように学習することにより, 理論的に到達できる最小の残留汎化誤差 ϵ_{min} が得られる. 従って, この観点からは, R の例題数 α の増加に対する変化率：$dR/d\alpha$ を最大化することが望ましい. 従って, ここでは R の従う発展方程式 (84) の右辺である $L(g(\alpha))$ を学習の各ステップで g に関して最大化しよう. そこで, 実際に $\partial L/\partial g = 0$ を行ってみると, 具体的に最適な $g(\alpha)$ は次の形をしているべきであることがわかる.

$$g_{\rm opt}(\alpha) \quad = \quad \frac{[F_a(R) - G_a(R)]l}{RE_a(R)} \tag{86}$$

ここで l 自体が (85) 式に従い, α 依存性を持つこと, また, $E_a(R), F_a(R), G_a(R)$ も, R の発展式 (84) を介して α 依存性を持つことから, ここで求めた最適な学習レート $g_{\rm opt}(\alpha)$ はステップ数 α 依存性を持つことに注意しよう. この (86) 式を状態更新式 (84)(85) 式に代入したものがここでの学習過程を記述することになる. しかし, この最適な学習レートの場合, (84) 式を (85) 式で辺々割ることにより

$$\frac{dR}{dl} \quad = \quad -\frac{[F_a(R)R - G_a(R)]R}{[F_a(R)R + G_a(R)]l} \tag{87}$$

が得られ, これは直ちに積分することができて, R-l 空間でのフローが, 数値的に微分方程式を解くまでもなく

$$(1+R)^{-(1+A)/A}(1-R)^{(1-A)/A}R \quad = \quad c\,l \tag{88}$$

のように陽な形で求めることができる. ここに, c は初期条件から決まる定数であり, $A = 1-2\Delta = 1-2\,{\rm e}^{-\alpha^2/2}$ である.

さて, 具体的に最適な学習レート $g_{\rm opt}(\alpha)$ に対する汎化誤差の例題数依存性を見ていく前に, この最適な学習レート $g_{\rm opt}(\alpha)$ が実際に最適か否かを簡単にチェックしておくことにする. そこで, (86) 式から学習レートの例題数依存性を

$$g(\alpha) \quad = \quad \gamma(\alpha)l \tag{89}$$

のように推測し, $\gamma(\alpha)$ の形を α^{-k} と置き, いくつかの k の値に対して作られる $g(\alpha)$ に対する学習過程を汎化誤差の例題数依存性として調べてみる. その結果を図 17 に載せる. この図からわかるように, この例題数が $\mathcal{O}(1)$ の領域では $k = 0.5$ と選ぶのが最も汎化誤差の減少が速く, $k = 0.1$ では学習レートの減衰が遅すぎるために学習が停滞し, 逆に $k = 1.2$ でも学習レートを速く落とし過ぎているため, うまくいっていない. そこで, $k = 0.5$ が最適のように見えるが, 我々の指針で求めた最適な学習レートのスケジューリング：$g_{opt}(\alpha)$ と $g(\alpha) = l\,\alpha^{-0.5}$ を比べてみることは意味があるであろう. そこで, 図 18 に, まずは学習可能な場合 ($a = \infty$) について両者を比較するプロットを載せる. この図より, 我々の選んだ学習レートのスケジューリングによる結果は $g(\alpha) = l\,\alpha^{-0.5}$ を凌いでいることがわかる.

ところで, $a = \infty$ の場合の漸近解析によれば, $\alpha \to \infty$ の漸近領域で学習曲線は

$$\epsilon_g \quad = \quad \frac{4}{\pi}\,\alpha^{-1} \tag{90}$$

のように振る舞い, α^{-1} 則に従うことがわかる. 学習レートを 1 に固定した場合のパーセプトロン学習アルゴリズムの漸近領域での学習曲線は前回みたように $\alpha^{-1/3}$ 則に従ったわけであるから, 確かに我々がここで行った学習レートの最適化によって, 学習が加速されていることがわかる. 一方, このときの学習レートは漸近領域で

$$g_{\rm opt} \quad = \quad 2c\sqrt{2\pi}\,\frac{{\rm e}^{-16/\alpha^2}}{\alpha} \tag{91}$$

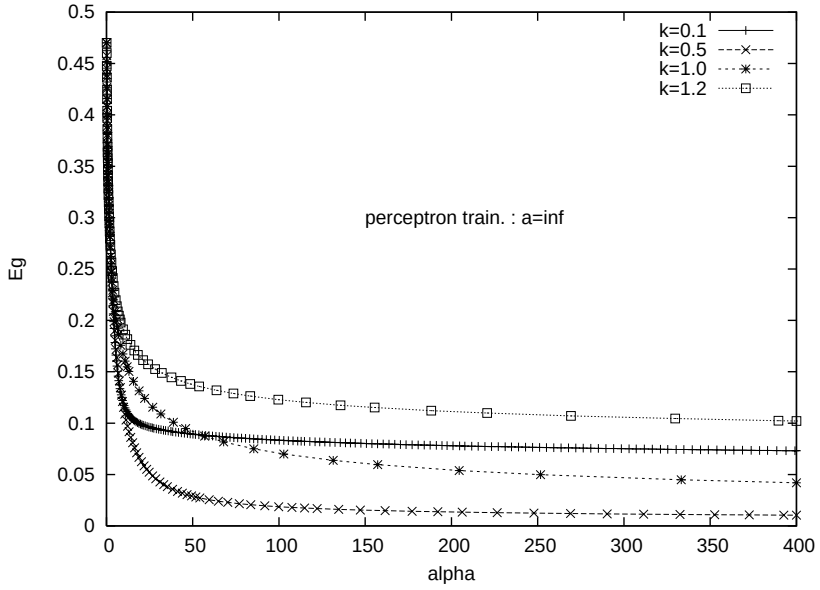

図 17: パーセプトロン学習アルゴリズムを学習レート $g(\alpha) = l\,\alpha^{-k}$ で動作させた場合の汎化誤差の例題数依存性. $k = 0.1, 0.5, 1, 1.2$ と選んである.

のように振舞う. ここで, c は初期条件により定まる定数である. ところで, 図 18 の結果を見る限りにおいて, $g(\alpha) = l\,\alpha^{-0.5}$ と $g_{\rm opt}(\alpha)$ にはほとんど差がないように見える. しかし, これは $\mathcal{O}(1)$ の例題数領域では $g_{\rm opt}$ は $l\,\alpha^{-0.5}$ のような例題数依存性を持つけれども, 例題数が増加するにつれて**学習レートの例題数依存性が $l\,\alpha^{-0.5}$ から (91) 式に切り替わる**ことを意味している. 従って, 両者の差は図 18 からも多少読み取れるように, α の増加とともに広がっていく. いずれにしても, 我々が選んだ学習レートの最適なスケジューリング $g_{\rm opt}$ は学習可能な場合には全例題数の領域で学習を加速し, 結果的に学習曲線を最適化することがわかった.

学習不可能な場合にはどうであろうか？ $a = 1$ の場合について, その結果を図 19 に載せる. この図より, $g = l\,\alpha^{-k}$ と仮定して, $k = 0.5, 1$ のように選んだ場合には, 学習曲線が最適化されるどころか, 理論的な最小値 ϵ_{min} に収束せず, $k = 1$ の場合には過学習が生じてしまっている. 一方, g を最適化することにより得られる $g_{\rm opt}$ を用いると, 例題数が小さな領域で汎化誤差が一時的に増加に転じる以外は順調に理論上の最小残留汎化誤差 ϵ_{min} に向かうことがわかる. 例題数が無限大での漸近解析によれば学習不可能な場合に対する漸近領域での汎化誤差の振る舞いは

$$\epsilon_g \;=\; \frac{\sqrt{2}}{\pi}\frac{\sqrt{2\pi H(a)}}{1-2\Delta}\frac{1}{\sqrt{\alpha}} + 2H(a) \tag{92}$$

となり (ここに, $\Delta = {\rm e}^{-a^2/2}, H(a) = \int_a^\infty dz\,{\rm e}^{-z^2/2}/\sqrt{2\pi}$ であることに注意), $\alpha^{-1/2}$ 則で理論上の最小残留汎化誤差 ϵ_{min} 向かう.

このときの学習レート $g_{\rm opt}$, 生徒機械の結合ベクトルの長さ l, 及び, 重なり R は漸近的に

$$g_{\rm opt} \;=\; c\frac{\sqrt{2\pi}}{1-2\Delta}\frac{\alpha^{-2\Delta/(1-2\Delta)}}{\alpha} \tag{93}$$

$$l \;=\; c\,\alpha^{-2\Delta/(1-2\Delta)} \tag{94}$$

$$R \;=\; 1 - \frac{2\pi H(a)}{(1-2\Delta)^2}\frac{1}{\alpha} \tag{95}$$

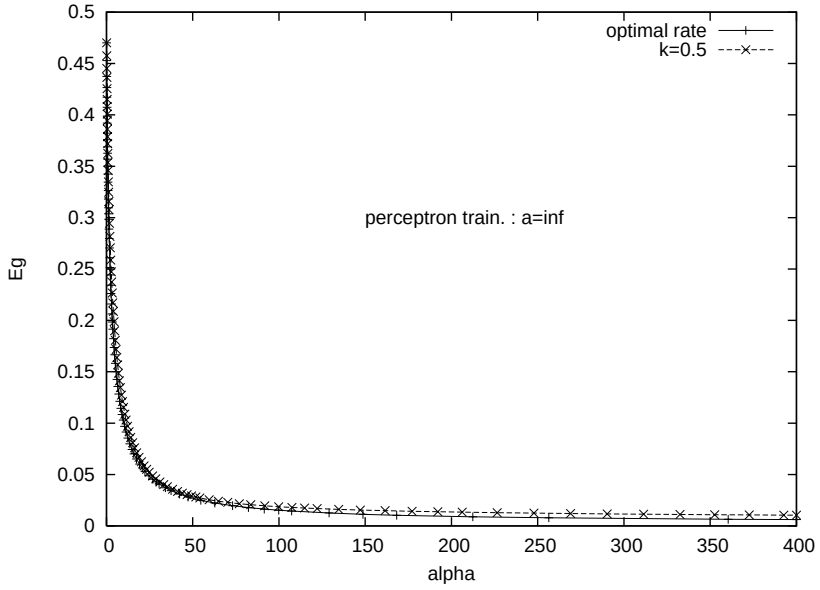

図 18: 最適な学習レート $g_{\rm opt}$ による結果と $g = l\,\alpha^{-0.5}$ の比較.

のように振る舞い, 結合の長さ l は $a < a_{c1} = \sqrt{2\log 2}$ で R が 1 に向かうにつれて発散することも見て取れる (図 19 (右) 参照).

3.10.4　質問付きオンライン Hebb 学習の最適化

ここまでの議論で, 各学習ステップで重なりの変化率 $dR/d\alpha = L(g(\alpha))$ を最大化することにより, 学習レートが最適化され, 結果的に学習を加速させることがわかった. しかし, 前回の講義で見たように, $a > a_{c1} = 0.80$ の場合にはこの戦略で良いが, $a < a_{c1}$ の場合には, 理論上の最小残留汎化誤差が

$$\epsilon_{min} = E_a(R_*) \tag{96}$$

$$R_* = -\sqrt{\frac{2\log 2 - a^2}{2\log 2}} \tag{97}$$

で与えられた. 従って, ここでの $dR/d\alpha$ を各学習ステップで最大化するという戦略は使えない.

そこで, 前回見た Hebb 学習に質問の効果を導入した場合の学習曲線が $a < a_{c1} = \sqrt{2\log 2}$ の領域で理論上の最小値 ϵ_{min} に向かったことに注目する. [**講義ノート** #8] では明記しなかったが, この場合の汎化誤差は漸近的に

$$\epsilon_g = \epsilon_{min} - \frac{16\log 2\sqrt{2\log 2 - a^2}}{a^2}\left[1 - Q\left(2, \frac{1}{2}\log 2\right)\right]\exp\left[-\frac{8\log 2}{\sqrt{\pi}a}\sqrt{2\log 2 - a^2}\sqrt{\alpha}\right] \tag{98}$$

のように振舞う. ここに, $Q(z,x)$ は

$$Q(z,x) = \frac{1}{\Gamma(z)}\int_x^\infty {\rm e}^{-t}\,t^{z-1}dt \tag{99}$$

で定義される不完全ガンマ関数である. 従って, $a < a_{c1}$ では質問付きのオンライン Hebb 学習は $g = 1$ であっても非常に良好な振る舞いを見せるのである.

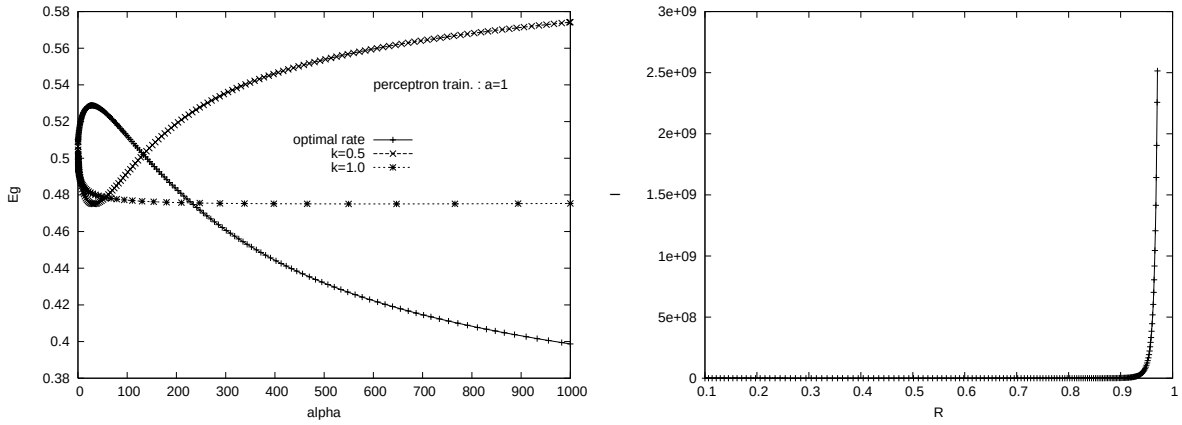

図 19: 学習不可能は場合 $a = 1$ の学習レートを導入した学習曲線. $g = l\,\alpha^{-k}$ と仮定して, $k = 0.5, 1$ と選んだ場合と, $g_{\rm opt}$ による結果 (左). 右側の図は学習レートを最適化した場合の R-l 空間でのフロー.

一方, $a_{c2} = 0.80 < a < a_{c1} = \sqrt{2\log 2}$ の場合には最小残留汎化誤差を与える重なりの値が $R = 1$ であるから, 我々が前節で用いた学習レートの最適化法が適用できる. 従って, この場合の質問付きオンライン Hebb 学習にも学習レートを導入し, それを最適化することにより, 学習を加速できないものか, と考えるのは自然であろう.

そこで, ここではこうして最適化された学習レートが $g = 1$ の場合の結果と比べてどの程度改善されるのかを見るために, 漸近領域での振舞いのみを結果として書いておこう. まず, 汎化誤差は漸近領域において

$$\epsilon_g \quad = \quad 2H(a) + \frac{\sqrt{c}}{\pi}\exp\left(-\frac{\alpha}{\pi}\right) \tag{100}$$

のように振舞う. 学習レートを $g = 1$ で固定した場合の結果 (98) からわかるように, 固定レート $g = 1$ ではパラメータ a の値が $a_{c1} = \sqrt{2\log 2}$ に近づくにつれ, 収束性が落ち, $a = a_{c1}$ で「緩和時間」が発散し, 収束性は $\mathrm{e}^{-\sqrt{\alpha}}$ から冪則に落ちてしまう. 一方で, 学習レートを上記の意味で最適化した場合には, (100) 式からわかるように, a の値に依らずに $\exp(-\alpha/\pi)$ で最小残留汎化誤差へと漸近していくことがわかる.

さて, この節では「ステップ数依存性」「例題数依存性」という言い方をしてきたが, オンライン学習の場合に両者は同じものを指していることに注意されたい. 前回はオンライン・パーセプトロン学習の学習曲線の漸近形が $\alpha^{-1/3}$ 則に従い, ギブス・バッチ学習のそれは α^{-1} であったために, 精度的にはオンライン学習が劣る, と結論付けた. しかし, 皆さんの中には「α をステップ数と考えれば, この結果はギブス・バッチ学習の方が計算時間が少ないことを意味していないか ?」「精度も処理時間もバッチ学習の方が良好ではないか ?」と思われた方がいるかもしれない. しかし, バッチ学習の場合の α は学習ステップ数ではなく, [**学習ステップ数**]≠[**例題数**] であることに注意しなければならない. 実は, この講義で我々は両者の計算速度/ 処理時間の比較に関しては何も議論してこなかったのである. もし, バッチ学習に関し, その計算時間を議論するのであれば, ある例題数を与えた場合に訓練誤差を「コスト関数」とし, それをゼロ (あるいは最小) にするまでに要する「ステップ数」を評価しなければならない. このコスト関数の最小化としては既に述べたシミュレーテッド・アニーリングや最急降下法等, 様々に考えられるが, **その計算時間の評価はそこで用いる最小化手法を具体的に与えて初めて可能となる**ことに注意すべきである. また, これとは別の問題としてアニーリングのようなギブス・サンプラーを用いる場合には, 単位モンテカルロ・ステップがオンライン学習における微分方程式の単位ステップとどのような関係があるのか, をもはっきりとさせる必要があり, このあたりの微妙な注意が必要となってくる. 最も簡単には実際にシステムのサイズを与えて両者のシミュレーションを行い, 解を得るまでに要した CPU 時間で比較するのが良いと思われる.

問 13：
この講義ではパーセプトロン学習に学習レート $g(\alpha)$ を導入し，その例題数依存性をうまく制御することにより，学習を加速することができた．そこで，同じ方法をオンライン Hebb 学習で試してみたい．つまり，オンライン Hebb 学習に学習レート $g(\alpha)$ を導入し

$$\begin{aligned} \boldsymbol{J}^{m+1} &= \boldsymbol{J}^m + g(\alpha)\, T_a(v)\boldsymbol{x} \\ g(\alpha) &= \frac{l}{\alpha^k} \end{aligned}$$

として，k の値を様々変えて調べてみよう．そこで，前回出題した [問い 12] の計算機シミュレーションのプログラムを改良し，上記を調べよ．ここで参考のため，下記にオンライン・パーセプトロン学習の計算機シミュレーションのサンプルプログラム，及び，その出力結果例 (図 20) を載せるので参考にしてもよい．

(注)：下記プログラム (online.c) は混沌系工学特論のホームページ：

`http://chaosweb.complex.eng.hokudai.ac.jp/~j_inoue/KONTON2004/konton2004.html`

にアップロードしてあるので，必要とあれば各自がダウンロードし，適時修正を加えて使ってみてください．

3.11　付録：オンライン学習の計算機シミュレーション

下記にオンライン・パーセプトロン学習のサンプルプログラムとその実行例を図 20 に載せる．各自が修正，改良を加えて様々楽しんでみると良いと思う．

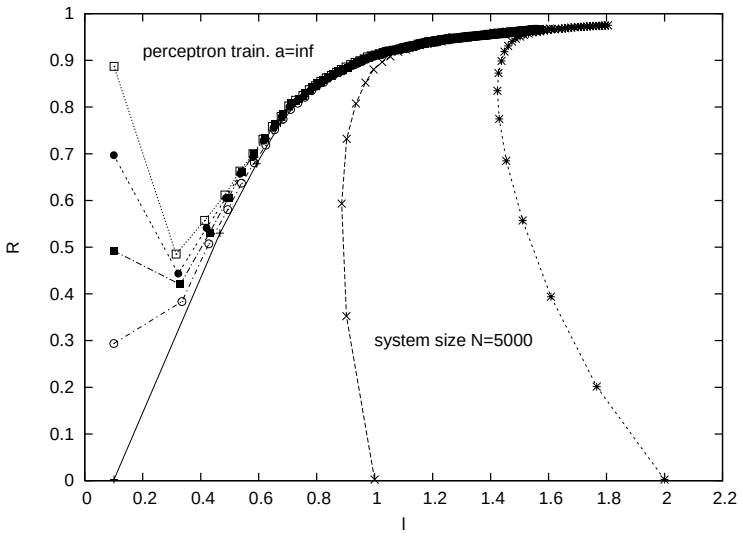

図 20: パーセプトロン・オンライン学習の計算機シミュレーション例．学習可能な場合の R-l 空間でのフロー．入力次元は $N = 5000$ に選んである．各自の計算機環境に応じてサイズ等を変えること．

```
/*************************************************************/
/*         Computer simulation of on-line learninig          */
/*                    (perceptron traininig)                 */
/*                                    J. Inoue               */
/*************************************************************/
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
#define size 5000           /* 入力の次元 */
#define a 6.0              /* 教師機械の内部パラメータ */
#define SEEDINITIAL 341
#define SEEDINITIAL_S 953
float J0[size];            /* 教師機械の結合 */
float J[size];             /* 生徒機械の結合*/
float x[size];             /* 例題 */
int OutTeach;              /* 教師機械の出力 */
int OutStudent;            /* 生徒機械の出力 */
int Delta;              /* 生徒がアップデートするか否かを決めるファクター */
float hT;               /* 教師機械の内部ポテンシャル */
float hS;               /* 生徒機械の内部ポテンシャル */
float Norm;             /* 生徒機械の結合の長さの規格化因子 */
float L;                /* 秩序変数（結合の長さ）*/
float Q;                /* 秩序変数（重なり）*/
float ll;
/*********************************************************************/
/*    Generating randomnumbers by Numerical Recepies in C (ran3)     */
/*********************************************************************/
#define MBIG 1000000000
#define MSEED 161803398
#define MZ 0
#define FAC (1.0/MBIG)
float ran3(long *idum)
{
static int inext,inextp;
static long ma[56];
static int iff=0;
long mj,mk;
int i,ii,k;

if (*idum < 0 || iff == 0) {
iff=1;
mj=MSEED-(*idum < 0 ? -*idum : *idum);
mj %= MBIG;
ma[55]=mj;
```

```
mk=1;
for (i=1;i<=54;i++) {
ii=(21*i) % 55;
ma[ii]=mk;
mk=mj-mk;
if (mk < MZ) mk += MBIG;
mj=ma[ii];
}
for (k=1;k<=4;k++)
for (i=1;i<=55;i++) {
ma[i] -= ma[1+(i+30) % 55];
if (ma[i] < MZ) ma[i] += MBIG;
}
inext=0;
inextp=31;
*idum=1;
}
if (++inext == 56) inext=1;
if (++inextp == 56) inextp=1;
mj=ma[inext]-ma[inextp];
if (mj < MZ) mj += MBIG;
ma[inext]=mj;
return mj*FAC;
}
#undef MBIG
#undef MSEED
#undef MZ
#undef FAC
/* (C) Copr. 1986-92 Numerical Recipes Software 1+5-5i. */
/*************************************************************/
/*              Generating random examples                   */
/*************************************************************/
void GenerateRandomPattern(int k)
{
  long *idum,m;
  int i;
  idum = &m;
  *idum = k;
    for(i=0; i < size; i++){
      x[i] =  2.0*ran3(&m)-1.0;
    }
    for(i=0, Norm=0; i < size; i++){
      Norm = Norm + x[i]*x[i];
    }
```

```
    for(i=0; i < size; i++){
      x[i] = x[i]/sqrt(Norm);
    }
}
/*********************************************************/
/*       Set initial Teacher's weight vector             */
/*********************************************************/
void SetInitial()
{
  long *idum,m;
  int i,j,k;
  float J0norm;
  idum = &m;
  *idum = -SEEDINITIAL;
  for(i=0; i < size; i++){
    J0[i] = 2.0*ran3(&m) -1.0;
  }
  for(j=0, J0norm=0; j < size; j++){
    J0norm = J0norm + J0[j]*J0[j];
  }
  for(k=0; k < size; k++){
    J0[k] = J0[k]*sqrt(size)/sqrt(J0norm);
  }
}
/*********************************************************/
/*               Set initial Student                     */
/*********************************************************/
void SetInitialStudent(float d)
{
  int i,j,k;
  float Jnorm;
  long *idum,m;
  idum = &m;
  *idum = -SEEDINITIAL_S;
  for(i=0; i < size; i++){
    J[i] = 2.0*ran3(&m) -1.0;
  }
  for(j=0, Jnorm=0; j < size; j++){
    Jnorm = Jnorm + J[j]*J[j];
  }
  for(k=0; k < size; k++){
    J[k] = J[k]*(d)*sqrt(size)/sqrt(Jnorm);
  }
}
```

```
/*******************************************************/
/*           Calculating local fields                  */
/*******************************************************/
/*------ 教師の内部ポテンシャルを計算    -------------*/
void LocalFieldT()
{
  int i;
    for(i=0,hT=0; i < size; i++){
      hT = hT+ J0[i]*x[i];
    }
}
/*----- 生徒の内部ポテンシャルを計算 ----------*/
void LocalFieldS()
{
  int i;
    for(i=0,hS=0; i < size; i++){
      hS = hS + J[i]*x[i]*sqrt(ll)/sqrt(size);
    }
}
/*********************************************************/
/*         Calculating Teacher's OutPut                  */
/*********************************************************/
void TeacherOut()
{
       if((hT > 0 && hT  < a) || (hT < -a)){
      OutTeach = 1;
    }else{
      OutTeach= -1;
    }
}
/*******************************************************/
/*      Calculating Student's OutPut                   */
/*******************************************************/
void StudentOut()
{
    if(hS > 0){
      OutStudent = 1;
    }else{
      OutStudent = -1;
    }
}
/*************************************************************/
/*    Condition for updating the weight of the Student       */
/*************************************************************/
```

```
void CalDelta()
{
    if(OutStudent*OutTeach < 0){
      Delta = 1;
    }else{
      Delta = 0;
    }
}
/*********************************************************/
/*     Update rule of the Student's weight           */
/*********************************************************/
void Update()
{
  int i;
    for(i=0; i < size; i++){
      J[i] = J[i] - Delta*OutStudent*x[i];
    }
}
/*****************************************************************/
/*    Calculating the length of the Student's weight vector   */
/*****************************************************************/
void CalLength()
{
  int i;
  for(i=0,ll=0; i < size; i++){
    ll = ll + J[i]*J[i];
  }
  L = sqrt(ll)/sqrt(size);
}
/*********************************************************************/
/*  Calculating the overlap bewteen the Teacher and the Student  */
/*********************************************************************/
void CalOverlap()
{
  int i;
  for(i=0,Q=0; i < size; i++){
    Q = Q + (J0[i]*J[i]/sqrt(ll))/sqrt(size);
  }
}
/*********************************************************/
/*                    Main program                       */
/*********************************************************/
main()
{
```

```
  FILE *pf;
  int i,imax,c;
  float d;
  imax = 200000;
  d = 2.00;  /* 生徒機械の長さの初期値  */
  SetInitial();
  SetInitialStudent(d);
  CalLength();
  CalOverlap();
  if((pf = fopen("on-b.dat","wt")) != NULL){
    fprintf(pf,"\n %f %f", L,Q);
  }
  fclose(pf);
  if((pf = fopen("on-b.dat","wt")) != NULL){
    for(i=0; i < imax; i++){
      GenerateRandomPattern(-i);
      LocalFieldT();
      LocalFieldS();
      TeacherOut();
      StudentOut();
      CalDelta();
      Update();
      CalLength();
      CalOverlap();
      /* 出力ファイルが大きくなりすぎるので，5000 ステップ毎にモニタリング */
      /* 全ての点をプロットしたければ，fmod(i,5000) を fmod(i,1) に修正する */
      if(fmod(i,5000)==0){fprintf(pf,"\n %f %f",L,Q);}
    }
  }
  fclose(pf);
}
```