データサイエンスと数学: 数学ソフトウエアとその活用

高山信毅 (神戸大学)

このスライドは講義で触れなかった発展的話題も含みます.

このスライドおよび参考資料はこちらにあります.

`http://www.math.kobe-u.ac.jp/HOME/taka/2019/data-b`

ソフトウエアは (2 進数 32 桁の) 整数達の四則演算 * と大小比較による分岐ですべてを処理.

*正確には高校数学 A の発展コラムによく出てる mod 計算

文字認識ソフトウエアのデモ

画像も 1byte 数の列. bmp の例. 2 進数と 16 進数.

0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f;
10,11,12, ..., 19, 1a,1b,..., 1f;
20,21,22, ..., 29, 2a,2b,..., 2f;

tensorflow on python.

```
train_step
=tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

アルゴリズムの核は最小値問題.
“微分して極値を求める” 以外にもいろんな方法がある.
目標: gradient descent.

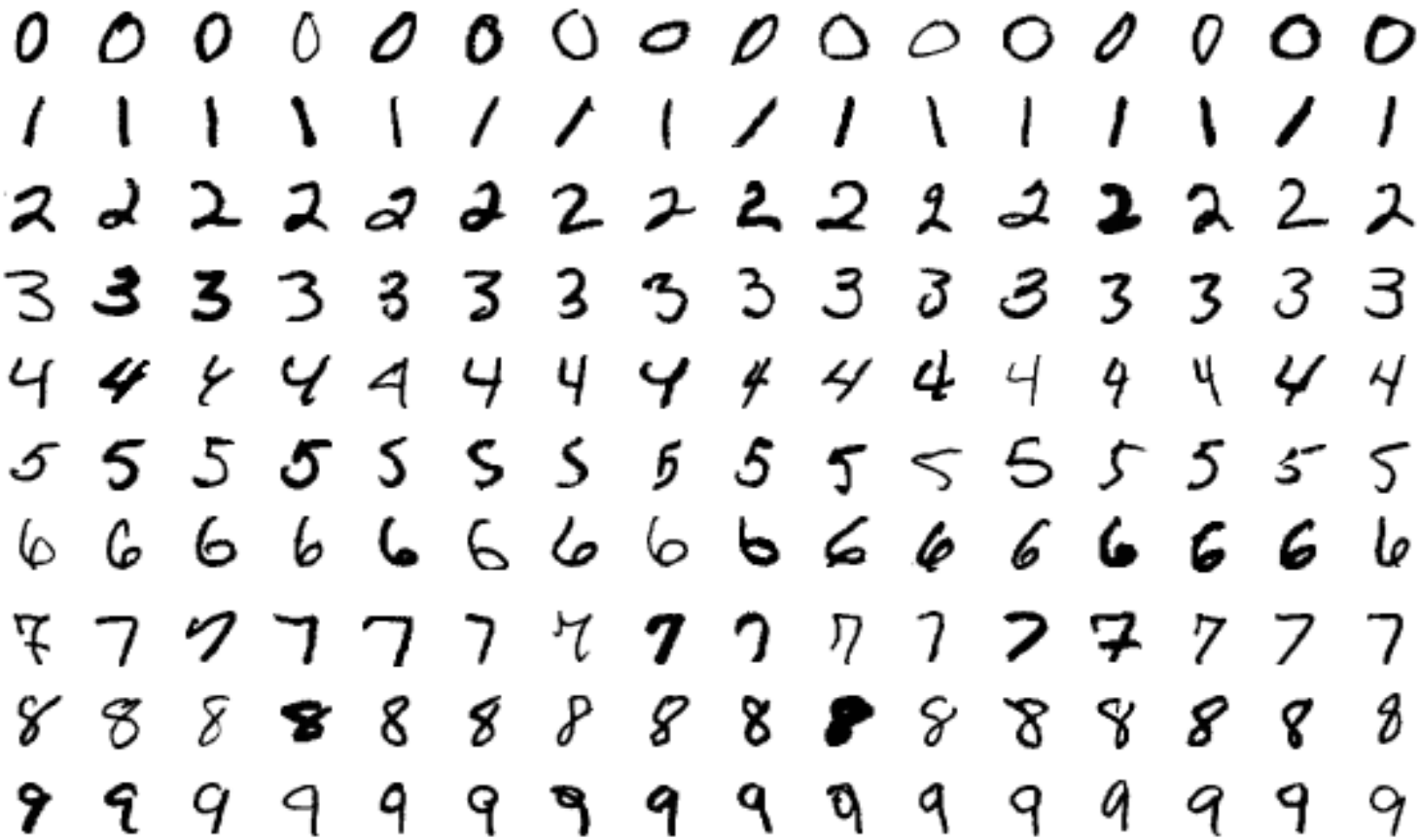


Figure: mnist の手書き数字データの例. By Josef Steppan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64810040

発展. 入力 $28 \times 28 = 784$ pixel data.

$$S(xW + b),\ x \in \mathbf{R}^{1\times 784}, W \in \mathbf{R}^{784\times 10}$$

値は (0 の確率, ..., 9 の確率). $S(x)_i = \exp(x_i)/\sum_{j=1}^{10} \exp(x_j)$.

```
# coding: utf-8
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)
import tensorflow as tf   #注意: 上は offline では実行できません.
x = tf.placeholder(tf.float32,[None,784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x,W)+b)
y_=tf.placeholder(tf.float32,[None,10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_*tf.log(y),reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
for i in range(1000):
    batch_xs,batch_ys = mnist.train.next_batch(100)
    sess.run(train_step,feed_dict={x:batch_xs,y_:batch_ys})
# feed_dict で入力ベクトルと正解を環境にする.
sess.run(tf.argmax(y,1),feed_dict={x: mnist.test.images})
#     array([7, 2, 1, ..., 4, 5, 6], dtype=int64)
sess.run(tf.argmax(y_,1),feed_dict={y_:mnist.test.labels})
#     array([7, 2, 1, ..., 4, 5, 6], dtype=int64)
```

http://www.math.kobe-u.ac.jp/HOME/taka/2018/c1/2018-05-31-c1-optim-screen/ : デモ movie

発展: Kullback divergence　x に対する正解を $p(x)$, 推測値を $q(x;\theta)$ とする. ここで θ は学習パラメータ (前のページの例では W, b. S が q).

最小にする目的関数の例 1.

$$F(\theta) = \sum_{x\in\Omega} |p(x) - q(x;\theta)|^2$$

最小にする目的関数の例 2 (Kullback divergence).

$$\begin{aligned} D(p|q) &= \sum_{x\in\Omega} p(x) \log \frac{p(x)}{q(x;\theta)} \\ &= \sum_{x\in\Omega} p(x) \log p(x) - \sum_{x\in\Omega} p(x) \log q(x;\theta) \end{aligned}$$

$D(p|q) \geq 0$, $D(p|p) = 0$.

Ω = bitmap data の集合 $\times$ I, $I = \{0, 1, \ldots, 9\}$. $p(x), q(x;\theta)$ は Ω の上の確率分布. $p(x)$ はプログラムの y_, $q(x)$ はプログラムの y. $p(x)$ は手書き文字の bitmap data x_1 に対する正解の x_2 のみで $1/N$ となる確率分布. たとえば x_1 の正解が 2 なら $p((x_1, 2)) = 1/N$, $p((x_1, k)) = 0$, $k \neq 2$. ここで N は bitmap data の数.

Gradient descent (勾配降下法) とは?

一変数の滑らかな関数 $f(x)$ の極小値を求めたい. Taylor 展開より

$$f(x+h) = f(x) + f'(x)h + O(h^2)$$

h として $-f'(x)\varepsilon$, $\varepsilon > 0$ をとると,

$$f(x+h) = f(x) - f'(x)^2\varepsilon + O(h^2)$$

なので, $f(x+h) < f(x)$ だろう †.

1. $f'(x)$ を計算. **if** $f'(x) = 0$ かつ $f''(x) > 0$, **then** x で $f(x)$ は極小値を持つ. 終了.
2. $x \leftarrow x - f'(x)\varepsilon$. Goto 1.

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3)$$

$h = 0.1$, $h^2 = 0.01$, $h^3 = 0.001$, $\cdots$.

問. ε はどのくらいにとればいいのか?

† $g(h) = O(h^2)$ とは, $M, H > 0$ が存在して, $|h| < H$ なら $|g(h)| \leq M|h^2|$ となること.

素手での計算実験

$f(x) = x^2$ の時, $x = 1$ の近くでは,
$f(1+h) \sim f(1) + f'(x)h = 1 + 2h$ のはず. $h = 0.1$, $h = 0.05$, ...
で調べてみる.

python での計算実験

$f(x) = x^4$ の時,

$$f(x+h) = (x+h)^4 = x^4 + 4x^3h + O(h^2)$$

のはず.

```
x = 3; h = 0.1; (x+h)**4-x**4-4*x**3*h
```

```
python
>>> x=3; h=0.1; (x+h)**4-x**4-4*x**3*h
0.552100000000065
>>> x=3; h=0.001; (x+h)**4-x**4-4*x**3*h
5.401200099140746e-05
>>> x=3; h=0.0001; (x+h)**4-x**4-4*x**3*h
5.40012027681197e-07
```

python での計算実験　$f(x) = x^4$ の時,

$$f(x+h) = (x+h)^4 = x^4 + 4x^3h + \frac{12}{2}x^2h^2 + O(h^3)$$

のはず.

```
x = 3; h = 0.1; (x+h)**4-x**4-4*x**3*h-6*x**2*h**2
```

```
python
>>> x = 3; h = 0.1; (x+h)**4-x**4-4*x**3*h-6*x**2*h**2
0.012100000000006328
>>> x = 3; h = 10**(-4); (x+h)**4-x**4-4*x**3*h-6*x**2*h**2
1.2027681197030863e-11
>>> x = 3; h = 10**(-5); (x+h)**4-x**4-4*x**3*h-6*x**2*h**2
1.285115505759688 4e-14
```

参考:微分を使わない方法

Th $f(x)$ は区間 $[a, c]$ で連続. $a < b < c$.

$f(b) < f(a), f(b) < f(c) \Rightarrow f(x)$ は (a, c) に極小値を持つ

区間 $[a, c]$ を如何に小さくして極小値を網にいれていくか?
Input: $a < b < c$
Output: 新しい 3 つの数.

1. **if** $c - b > b - a$,
   - 1.1 $x \in (b, c)$ を 適当に 選ぶ.
   - 1.2 **if** $f(x) < f(b)$, **then return** $b < x < c$ **else if** $f(b) < f(x)$, **then return** $a < b < x$.
2. **if** $c - b \leq b - a$,
   - 2.1 $x \in (a, b)$ を 適当に 選ぶ.
   - 2.2 **if** $f(x) < f(b)$, **then return** $a < x < b$ **else if** $f(b) < f(x)$, **then return** $x < b < c$.

x をどのように選ぶか?

参考:微分を使わない方法—黄金分割

問. x をどのように選ぶか?

案. $[a,b]$, $[b,c]$ の大きい方を 0.38 : 0.62 に分割する点を x とする.

$\frac{b-a}{c-a} = w$, つまり b は $[a,c]$ を $w:(1-w)$ に内分してるとする.

$x-b = z(c-a)$ で z をきめる. $\frac{x-b}{c-b} = \frac{z}{1-w}$ なので, $w < 1/2$ なら x は $[b,c]$ を $\frac{z}{1-w} : 1-\frac{z}{1-w}$ に内分している[‡].

これらの内分比が等しいとすると,

$$w = \frac{z}{1-w} \tag{1}$$

$w < 1/2$ と仮定しよう. $f(x)$ の値ににより, 新しい三組は $a < b < x$ か $b < x < c$ になる. それぞれの区間の幅は $x-a$, $c-b$ である. この幅を w, z で書くと, $w(c-a)+z(c-a)$, $(1-w)(c-a)$. ここで, 新しい三組どちらが発生しても新しい区間の幅が等しいと仮定すると[§],

$$w + z = 1 - w \tag{2}$$

(1), (2) より z を消去すると $w^2 - 3w + 1 = 0$ をよる. $0 < w < 1/2$ より $w = \frac{3-\sqrt{5}}{2} \sim 0.38$ を得る. 導出の方法より $\frac{x-b}{c-b} = w$ に注意しておく. よって, x は, 長い方の区間を $w : 1-w$ に内分した点に選ぶのがよいのではないかと思われる.

---

‡ $\frac{z(c-a)}{c-b} = \frac{z}{1-w}$

§ w(c-a)+z(c-a) = (1-w)(c-a) より

Gradient descent (勾配降下法) とは? 2 変数

$$f(x+p, y+q) = f(x,y) + f_x(x,y)p + f_y(x,y)q + O(p^2+q^2) \quad (3)$$

ここで $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$. $(f_x(x,y), f_y(x,y))$ を f の gradient と呼ぶ.
たとえば,

$$(p,q) = -(f_x(x,y), f_y(x,y))\varepsilon, \quad 1 \gg \varepsilon > 0 \quad (4)$$

とすれば,

$$f(x+p, y+q) \sim f(x,y) - (f_x(x,y)^2 + f_y(x,y)^2)\varepsilon \quad (5)$$

なので

$$f(x+p, y+q) < f(x,y)$$

となると期待できる. (4) の方向へ $f(x,y)$ を制限する. この一変数関数の極小値を求める. 極小値でまた同じことを繰り返す. このように $f(x,y)$ が極小値をもつ (x,y) を見つけるのが gradient descent. (4) よりうまい方向を提案するのが conjugate gradient method.

また python での計算実験　$f(x,y) = x^4 + y^4$ の時,

$$f(x+p, y+q) = (x+p)^4+(y+q)^4 = x^4+y^4+4x^3p+4y^3q+O(p^2+q^2)$$

のはず.

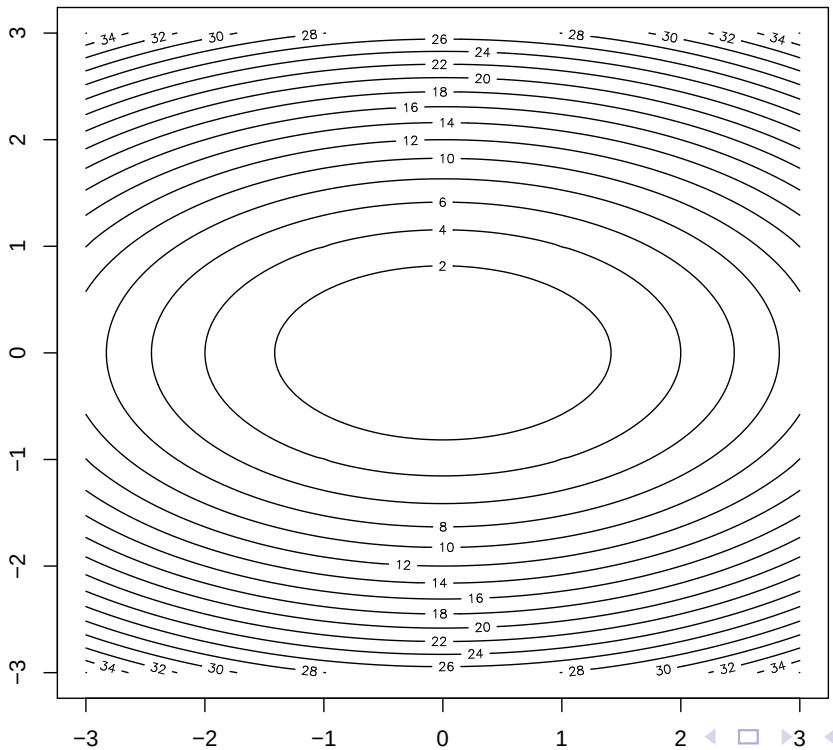
```
x = 3; y=4; p = 0.1; q=0.1;
(x+p)**4+(y+q)**4-x**4-y**4-4*x**3*p-4*y**3*q
```

```
python
>>> x = 3; y=4; p = 0.1; q=0.1;
... (x+p)**4+(y+q)**4-x**4-y**4-4*x**3*p-4*y**3*q
1.52819999999888
>>> x = 3; y=4; p = 10**(-3); q=10**(-4);
... (x+p)**4+(y+q)**4-x**4-y**4-4*x**3*p-4*y**3*q
5.497201692733142e-05
>>> x = 3; y=4; p = 10**(-10); q=10**(-10);
... (x+p)**4+(y+q)**4-x**4-y**4-4*x**3*p-4*y**3*q
2.4328031575481872e-14
```

例　$f(x,y) = x^4 + y^4 - xy$, $(f_x, f_y) = (4x^3 - y, 4y^3 - x)$.
$(x,y) = (1,1)$ で動くべき方向は, $-(3,3)\varepsilon$.

発展　共役勾配法では次の方向は Hessian について直交する方向を選ぶ. Hessian を計算せずにうまく直交に近い方向をみつけるアルゴリズムがいろいろ提案されている.

例　$f(x,y) = x^2 + 3y^2$ の最小値. Hessian は $\begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}$.



R の optim 関数 (極小値を探す)

R, (https://www.r-project.org) と Grapher.

例 $f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$.

```
rb<-function(x) {
 (1-x[1])^2+100*(x[2]-x[1]^2)^2;
}
rb(c(1,2))    # 値を計算してるか確かめてる.
optim(par=c(-1.2,1),fn=rb)  # default, Nelder-Mead 法
optim(par=c(-1.2,1),fn=rb, method="CG") #CG 法
```

```
$par
[1] 1.000260 1.000506
$value
[1] 8.825241e-08
$counts
function gradient
     195       NA

$par
[1] -0.7648079  0.5927148
$value
[1] 3.106475
$counts
function gradient
     402      101
```

微分積分で習う方法はダメ?　そんなことはない. 数学ソフトウエアのアルゴリズムには微分積分で習う方法をさらに精緻にしたものがある. 解ける問題のサイズは小さいが正確な答えをだせる.

1. 一変数代数方程式, Schönhage algorithm. pari/gp の roots.
2. 多変数, Gröbner basis, たとえば
   `http://www.math.kobe-u.ac.jp/Asir/index-ja.html`
3. Numerical homotopy method, たとえば
   `http://homepages.math.uic.edu/~jan/download.html`

```
[1982] F;
5*x^10-5*x^8+x-5
[1983] G=diff(F,x);
50*x^9-40*x^7+1
[1984] pari(roots,G);
[ -0.9150100387698543982058464793395548142 1
  0.6608736761041111872598746977391170880 4
  0.8623159802759930519653427612509520687 2
  虚部のある解は省略.]
```

微分積分で習う方法, 例

Rosenbrock の banana 関数.

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{6}$$

グラフを見ると極小値は3つあるかも.
R, (`https://www.r-project.org`) と Grapher.

```
curve((1-x)^2+100*(2-x^2)^2, xlim=c(-2,2),ylim=c(-1,200),
      ylab="",col="green")
```

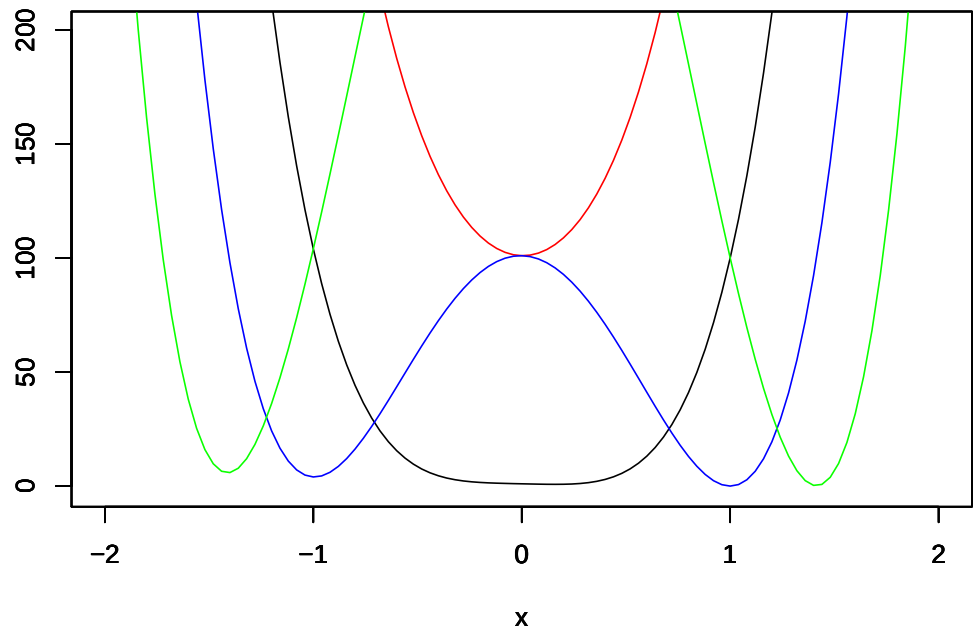


Figure: red: $y = -1$, black: $y = 0$, blue: $y = 1$, green: $y = 2$

グレブナー基底で $f_x = 0, f_y = 0$ を満たす (x, y) をすべて求めると?

グレブナー基底を用いるソフトウエア
`http://asir2.math.kobe-u.ac.jp/cgi-bin/cgi-data-b.sh`
で $f_x = 0, f_y = 0$ を満たす (x, y) をすべて探して調べる.
$\Rightarrow$
$(x, y) = (0, 0)$ のみに極小値.

一体何個 gradient $= \mathbf{0}$ となる点があるか?

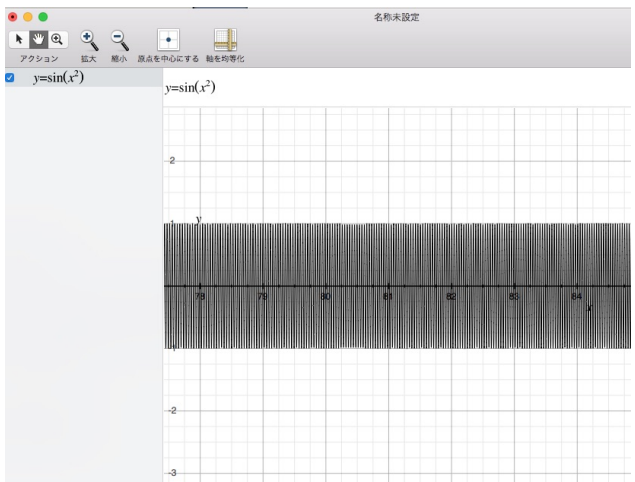
選択課題 3 [¶] を matrix factorization で同じように解く. この場合 phc pack に実装されてる numerical homotopy 法によると, gradient が 0 となる点は (複素トーラス $(\mathbf{C} \setminus \{0\})^{14}$ の中で) 687961 個あり. Mac mini (2.3GHz Intel Core i7) 約一日の計算でその中の 5865 個の近似値を求めた.
並列計算可能なので, 約 118 台の Mac mini を動かせば一日で全部の点が求まる.

[¶] `http://www.math.kobe-u.ac.jp/HOME/taka/2018/data-b` の quiz にあり.

数学ソフトウエアの落とし穴　有限桁の２進数を基礎.

```
curve(sin(x),from=0,to=10);  #正しい
curve(sin(x),from=0,to=20);  #正しい
curve(sin(x^2),from=0,to=10);  # 間違い
curve(sin(x^2),from=0,to=10,n=1000);  # いい感じ.
curve(sin(x),from=0,to=100); # 間違い
```

Grapher, 手のマークを選択して, 右左にグラフを動かすと範囲外が見える. なかなか優秀. しかし欠けているところがすこしあり.



参考書, その他のソフトウエア

1. Numerical recipes, http://numerical.recipes, 日本語訳もあり.
2. Numerical Optimization, http://users.iems.northwestern.edu/~nocedal/book/
3. パターン認識と機械学習, [MS], https://www.microsoft.com/en-us/research/people/cmbishop/
4. グレブナー道場, http://www.math.kobe-u.ac.jp/OpenXM/Math/dojo/dojo-mult/dojo-toc.html
5. mathlibre(sage, Risa/Asir, R, ...), http://www.mathlibre.org/index-ja.html, VMware 仮想マシン http://www.math.kobe-u.ac.jp/vmkm
6. Sage, http://www.sagemath.org
7. Maple, https://www.maplesoft.com
8. Mathematica, http://www.wolfram.com/mathematica/
9. Matlab, https://jp.mathworks.com/products/matlab.html

エキスパート向けソフトウエア　GSL

https://www.gnu.org/software/gsl/,

tensorflow https://www.tensorflow.org

```
source tensorflow/bin/activate
...
deactivate
```

```
# coding: utf-8
# 出典: https://ai-coordinator.jp/mnist
#  ## が後ろについてる行のみを実行.
#  ###  image の png file を作成する場合.
# pip install --upgrade tensorflow
import tensorflow as tf   ##
# pip install --upgrade keras が必要.
from keras.datasets import mnist  ##   ###
from keras.utils import np_utils
# pip install --upgrade pillow が必要
from PIL import Image  ###
(X_train, y_train), (X_test, y_test) = mnist.load_data() ##  ###
# 実行すると download が発生.
# たとえば X_train[0] ## で data の array を見れる.
outImg=Image.fromarray(
  X_train[train_no].reshape((28,28))).convert("RGB") ###
outImg.save("train.png") ### 手書き文字を png 形式に
```