航空宇宙情報システム学第三 レポート 1

工学部航空宇宙工学科三年 野本陽平 (03-180332)

2019 年 1 月 31 日

はじめに

問題はバランスよく全ての章から, それぞれなるべく難しい問題を恣意的に選択しました. 練習問題 2-8 は易しいですが, 正直 2 章の問題はどれも満遍なく易しいので仕方なく選びました. 練習問題 5-2 は多分学科の人たちは殆ど回答しないだろうなと予測しトライしてみましたが, あまりに自由な話をしても面白くないと思ったので, 最近僕が注目しているサービスについて書いてみました. 矢入先生の意図した小話ではない可能性があるのでその場合はすみません... 問題 8.1 は, 多分みんな 8 章は 8-2 や 8-3 をやるだろうなと思ったので敢えて 8-1 をチョイスしました.(推測が外れていたら残念です.) 全てのコードは Python3.7 で記述し, 後ろに Appendix として添付しています.

練習問題 1-2 (バネ-マス-ダンパ系の離散時間システム表現)

次式の微分方程式で表されるバネ・マス・ダンパ系について答えよ.

$$M\ddot{x} + D\dot{x} + Kx = u$$

1. 線形連続時間システムとしての状態方程式 ($\dot{x} = A_c x + b_c u$ の形) に直せ.

   Solution.

$$\underline{\frac{d\mathbf{x}}{dt} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{D}{M} \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} \quad \left(\text{ただし } \mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}\right)}$$

   ■

2. $M = 1.0[kg], K = 1.0[N/m], D = 0.3[Nsec/m]$, サンプリング間隔 $\Delta t = 1.0[sec]$ として, 離散時間システムの状態方程式を求めよ. また, $u_k = 0$ の場合のシステムの漸近安定性を確かめよ.

   Proof. 与えられた値 $M = 1.0[kg], K = 1.0[N/m], D = 0.3[Nsec/m]$ を代入すると,

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{D}{M} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -0.3 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

   故に A の固有値 λ は,

$$\lambda = \frac{-3 \pm \sqrt{391}i}{20} := \alpha \pm \beta i$$

   である. (計算を簡略化するために α と β を与えた.) 正則行列およびその逆行列 P, P^{-1}, 固有値 λ が並ぶ対角行列 D は $A = PDP^{-1}$ で結びついており, それぞれ

$$P = \begin{bmatrix} 1 & 1 \\ \alpha + \beta i & \alpha - \beta i \end{bmatrix}, \quad P^{-1} = -\frac{1}{2\beta i}\begin{bmatrix} \alpha - \beta i & -1 \\ -\alpha - \beta i & 1 \end{bmatrix}, \quad D = \begin{bmatrix} \alpha + \beta i & 0 \\ 0 & \alpha - \beta i \end{bmatrix}$$

である. この状態方程式を線形離散化すると行列指数関数を用いて以下のように変形できる.

$$
\begin{aligned}
A' &= \mathrm{e}^{A\Delta t} = P\mathrm{e}^{D\Delta t}P^{-1} \\
&= -\begin{bmatrix} 1 & 1 \\ \alpha+\beta i & \alpha-\beta i \end{bmatrix}\begin{bmatrix} \exp((\alpha+\beta i)\Delta t) & 0 \\ 0 & \alpha-\beta i \end{bmatrix}\frac{1}{2\beta i}\begin{bmatrix} \alpha-\beta i & -1 \\ -\alpha-\beta i & 1 \end{bmatrix} \\
&= \frac{\mathrm{e}^{\alpha\Delta t}}{2\beta i}\begin{bmatrix} \alpha(\mathrm{e}^{\beta i}-\mathrm{e}^{-\beta i})-\beta i(\mathrm{e}^{\beta i}+\mathrm{e}^{-\beta i}) & -(\mathrm{e}^{\beta i}-\mathrm{e}^{-\beta i}) \\ (\alpha^2+\beta^2)(\mathrm{e}^{\beta i}-\mathrm{e}^{-\beta i}) & -\alpha(\mathrm{e}^{\beta i}-\mathrm{e}^{-\beta i})-\beta i(\mathrm{e}^{\beta i}+\mathrm{e}^{-\beta i}) \end{bmatrix} \\
&= -\frac{\mathrm{e}^{\alpha\Delta t}}{\beta}\begin{bmatrix} \alpha\sin\beta\Delta t-\beta\cos\beta\Delta t & -\sin\beta\Delta t \\ \sin\beta\Delta t & -\alpha\sin\beta\Delta t-\beta\cos\beta\Delta t \end{bmatrix} \\
b' &= A^{-1}(\mathrm{e}^{A\Delta t}-I)b = A^{-1}(A'-I)b \\
&= \begin{bmatrix} -0.3 & -1 \\ 1 & 0 \end{bmatrix}\left(-\frac{\mathrm{e}^{\alpha\Delta t}}{\beta}\begin{bmatrix} \alpha\sin\beta\Delta t-\beta\cos\beta\Delta t & -\sin\beta\Delta t \\ \sin\beta\Delta t & -\alpha\sin\beta\Delta t-\beta\cos\beta\Delta t \end{bmatrix}-\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)\begin{bmatrix} 0 \\ 1 \end{bmatrix}
\end{aligned}
$$

ここで得た A', b' によって $\mathbf{x}_{k+1} = A'\mathbf{x}_k + b'u$ の関係が作られる. さらに $\alpha = -\frac{3}{20}, \beta = \frac{\sqrt{391}}{20}$ を代入すれば,

$$
\underline{\mathbf{x}_{k+1} = \begin{bmatrix} 0.582 & 0.727 \\ -0.727 & 0.364 \end{bmatrix}\mathbf{x}_k + \begin{bmatrix} 0.418 \\ 0.727 \end{bmatrix}u}
$$

なる状態推移の式を得る. A' の固有方程式は

$$
\lambda^2 - 0.946\lambda + 0.740 = 0
$$

より判別式が負なので虚数解を持つ. この時固有値の絶対値は

$$
|\lambda| = \sqrt{\lambda_1\lambda_2} = \sqrt{0.740} < 0
$$

故に漸近安定である. □

3. $u(t) = \sin(2t)[N], x(0) = 1, dx/dt(0) = 0$ として, 解析解を求めよ.

Solution. 状態方程式は

$$
\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -1 & -0.3 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\sin 2t
$$

両辺ラプラス変換すると,

$$
\begin{aligned}
s\mathcal{L}[\mathbf{x}] - x(0) &= \begin{bmatrix} 0 & 1 \\ -1 & -0.3 \end{bmatrix}\mathcal{L}[\mathbf{x}] + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\frac{2}{s^2+4} \\
\therefore \mathcal{L}[\mathbf{x}] &= \begin{bmatrix} s & -1 \\ 1 & s+0.3 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ \dfrac{2}{s^2+4} \end{bmatrix} \\
&= \frac{1}{(s-\alpha)^2+\beta^2}\begin{bmatrix} s+\dfrac{3}{10}+\dfrac{2}{s^2+4} \\ -1+\dfrac{2s}{s^2+4} \end{bmatrix}
\end{aligned}
$$

これをラプラス逆変換することによって以下の解を得る.

$$
\begin{aligned}
x(t) &= \mathcal{L}^{-1}\left[\frac{s+\frac{3}{10}}{(s-\alpha)^2+\beta^2}+\frac{\frac{5}{78}s+\frac{103}{156}}{(s-\alpha)^2+\beta^2}+\frac{-\frac{5}{78}s-\frac{25}{39}}{s^2+4}\right] \\
&= \mathcal{L}^{-1}\left[\frac{\frac{83}{78}s+\frac{749}{780}}{(s-\alpha)^2+\beta^2}+\frac{-\frac{5}{78}s-\frac{25}{39}}{s^2+4}\right] \\
&= \mathcal{L}^{-1}\left[\frac{83}{78}\frac{\left(s-\left(-\frac{3}{20}\right)\right)+\frac{1249/1560}{sqrt391/20}\frac{\sqrt{391}}{20}}{\left(s-\left(-\frac{3}{20}\right)\right)^2+\left(\frac{\sqrt{391}}{20}\right)^2}-\frac{5}{78}\frac{s+\frac{10}{2}2}{s^2+2^2}\right] \\
&= \underline{\frac{83}{78}e^{\alpha t}\left(\cos\beta t+\frac{1249}{1560\beta}\sin\beta t\right)-\frac{5}{78}(\cos 2t+5\sin 2t)}
\end{aligned}
$$

■

4. さらにこのとき, 離散時間システムの状態方程式の数値解を計算機によって求め, 解析解と比較せよ.

Solution. 解析解と数値解は以下のように反映される. なお, コードは末尾の Appendix にまとめて記載してある. かなり粗い近似であったが, それなりの精度で数値解は解析解に追従している.

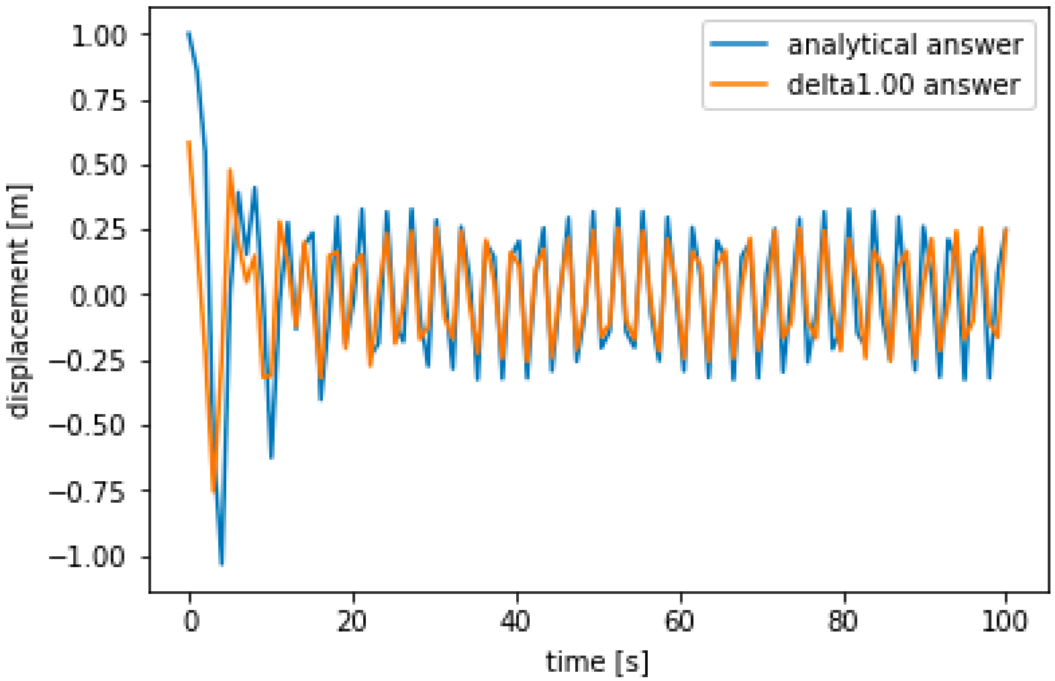

図 1　バネ・マス・ダンパ系の解析解と数値解の比較

ここで周波数はほぼ一致しているが, 振幅はやや小さく出ている. これは特に 20[sec] 以上の範囲で顕著であり, 故に原因は入力信号にあると推定できる. 近似が粗いため, ピーク部分が反映されていないのではないかと思われる (多くのタイミングでピークは小さく出ているが, 時に一致している箇所もあるのはこの関係である.) ■

練習問題 2-8 (線形和の期待値・分散)

2 つの確率変数 X と Y があり, その期待値 (平均) がともに μ であるとする. すなわち, $E[X] = E[Y] = \mu$ である. また, それぞれの分散は, $V(X) = 1, V(Y) = 2$ であり, 両者の共分散は, $Cov(X, Y) = 0$ であるとする. このとき, X と Y の重み付き線形和 $Z = aX + bY$(a, b は定数) について以下の問いに答えよ.

1. $E[Z]$ および $V(Z)$ を求めよ.

   Solution.

$$E(Z) = E(aX + bY) = aE(X) + b(Y) = \underline{(a + b)\mu}$$
$$V(Z) = V(aX + bY) = V(aX) + V(bY) = a^2V(X) + b^2V(X) = \underline{a^2 + 2b^2}$$

■

2. $E[Z] = \mu$ となる条件を求めよ.

   Solution.

$$E(Z) = (a + b)\mu = \mu$$
$$\therefore a + b = \underline{1}\text{のとき.}$$

■

3. 上の条件を満たすとき, $V[Z]$ が最小となるような a, b の値を求めよ.

   Solution. $b = 1 - a$ により未知数を減らす.

$$V(Z) = a^2 + 2(1 - a)^2 = 3a^2 - 4a + 2 = 3\left(a - \frac{2}{3}\right)^2 + \frac{2}{3}$$
$$\therefore (a, b) = \underline{\left(\frac{2}{3}, \frac{1}{3}\right)} \quad \text{のとき}, V(Z) \text{は最小となり} \frac{2}{3}.$$

■

練習問題 3-1

前のスライドで述べた [性質 1][性質 2] が成り立つことを示せ. また, 二乗誤差の期待値 $E[(\hat{X} - X)^2]$ が, 観測数 n に対してどのように変化するかを述べよ.

[性質 1] 普遍性 (unbiasness)

$E[\hat{X} - X] = 0$「推定したい変数 X との差の期待値が 0」(注意：誤差が 0 というわけではない)

[性質 2] 二乗誤差の期待値最小

$E[(\hat{X} - X)^2] \Rightarrow min.$「二乗誤差の期待値が最小」

Proof. 以下の条件を仮定し, これらの条件のもとで性質 1,2 が成立していることを証明する.
条件 1:各観測量 Y_i は, 真の量 X に誤差 W_i が加わったものである. すなわち, $Y_i = X + W_i$
条件 2:全ての誤差 W_i について, 平均 $E[W_i]$ が 0 であり, 分散が等しい. すなわち, $V(W_i) = \sigma^2$
条件 3:各観測誤差同士が無相関である. すなわち, この場合, $Cov(X_i, X_j) = E[W_i W_j] = 0$(ただし $i \neq j$)

$$\hat{X} = \Sigma_{i=1}^{n} a_i Y_i \quad (\text{ただし} \Sigma_{i=1}^{n} a_i = 1)$$

とおく. 条件 1, 2 より直ちに以下の関係式

$$E[Y_i] = E[X + W_i] = E[X] + E[W_i] = X$$
$$V[X] = V[X + W_o] = V[W_i] = \sigma^2$$

を得るので, 直ちに性質 1 が証明できる;

$$E[\hat{X} - X] = E[\Sigma_{i=1}^{n} a_i Y_i - X] = \Sigma_{i=1}^{n} a_i E[Y_i] - E[X] = X\Sigma_{i=1}^{n} a_i - X = 0$$

さて, 性質 1 と期待値の線形性より

$$E[(\hat{X} - X)^2] = E[(\hat{X} - E[\hat{X}])^2]$$

であるので, 性質 2 は推定量 $\hat{X}$ の分散が最小になることを考える問題である.

$$E[(\hat{X} - X)^2] = V[\hat{X}] = V[\Sigma_{i=1}^{n} a_i Y_i] = \Sigma_{i=1}^{n} a_i^2 V[Y_i] = \Sigma_{i=1}^{n} a_i^2 \sigma^2$$

今 a_i を最小化するため

$$f(a_1, a_2, ..., a_n) = \Sigma_{i=1}^{n} a_i^2 \sigma^2$$
$$g(a_1, a_2, ..., a_n) = \Sigma_{i=1}^{n} a_i - 1 = 0$$

とおいてラグランジュの未定乗数法を用いる. (f は最小にしたい関数で, g は拘束条件である.)

$$\frac{\partial f}{\partial a_n} = \lambda \frac{\partial g}{\partial a_n} \Leftrightarrow 2\sigma^2 a_n = \lambda \quad for \quad \forall n$$

ここで λ はある実数である. これを解くと性質 2 が示される;

$$\lambda = \frac{2\sigma^2}{n}, \quad a_i = \frac{1}{n}$$

確かに算術平均 $\hat{X} = \Sigma_{i=1}^{n} \frac{1}{n} Y_i$ が二乗誤差の期待値を最小にすることがわかる. なお

$$E[(\hat{X} - X)^2] = \Sigma_{i=1}^{n} \frac{1}{n^2} \sigma^2 = \frac{\sigma^2}{n}$$

より, 標本数を増やせば分散は 0 へと収束していく (直感とも一致している). □

練習問題 4-6 (非線形モデルの最尤推定)

下の表は, ある人工衛星の表面温度の変化を記録したものである. この温度変化のモデルを,

$$T(t) = T_0 + a\sin(\omega t + \theta) + \varepsilon$$
$$\varepsilon(t) \approx N(0, \sigma^2)$$

としたとき, パラメータ $T_0, a, \omega, \theta, \sigma^2$ を最尤推定せよ.

Solution. ある時刻 t_i において, T_i が生じる確率を $\varepsilon = T_i - T_0 - a\sin(\omega t_i + \theta)$. 正規分布に従うので, 対数尤度 LL は

$$LL(T_0, a, \omega, \theta, \sigma^2) = -\frac{n}{2}(\log 2\pi + \log\sigma^2) - \frac{1}{2\sigma^2}\Sigma_{i=1}^{n}\varepsilon^2$$

となる. 今回の最尤推定の問題は, 以下の 5 つの偏微分が 0 になる条件を探す連立方程式に帰着する.

$$\begin{aligned}
\frac{\partial}{\partial T_0}\Sigma_{i=1}^{n}\varepsilon^2 &= 0 = -\Sigma_{i=1}^{n}2(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
\frac{\partial}{\partial a}\Sigma_{i=1}^{n}\varepsilon^2 &= 0 = -\Sigma_{i=1}^{n}2\sin(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
\frac{\partial}{\partial \omega}\Sigma_{i=1}^{n}\varepsilon^2 &= 0 = -\Sigma_{i=1}^{n}2at_i\cos(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
\frac{\partial}{\partial \theta}\Sigma_{i=1}^{n}\varepsilon^2 &= 0 = -\Sigma_{i=1}^{n}2a\cos(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
\frac{\partial}{\partial \sigma^2}LL &= 0 = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\Sigma_{i=1}^{n}(T_i - T_0 - a\sin(\omega t_i + \theta))^2
\end{aligned}$$

数値計算にはニュートン法を用いる. 非線形多変数関数に対するニュートン法は

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k)\right)\mathbf{f}(\mathbf{x}_k)$$

であり,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_k) = \begin{bmatrix}
\frac{\partial^2}{\partial T_0^2}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial T_0 \partial a}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial T_0 \partial \omega}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial T_0 \partial \theta}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial T_0 \partial \sigma^2}\Sigma_{i=1}^{n}\varepsilon^2 \\
\frac{\partial^2}{\partial a \partial T_0}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial a^2}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial a \partial \omega}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial a \partial \theta}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial a \partial \sigma^2}\Sigma_{i=1}^{n}\varepsilon^2 \\
\frac{\partial^2}{\partial \omega \partial T_0}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \omega \partial a}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \omega^2}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \omega \partial \theta}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \omega \partial \sigma^2}\Sigma_{i=1}^{n}\varepsilon^2 \\
\frac{\partial^2}{\partial \theta \partial T_0}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \theta \partial a}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \theta \partial \omega}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \theta^2}\Sigma_{i=1}^{n}\varepsilon^2 & \frac{\partial^2}{\partial \theta \partial \sigma^2}\Sigma_{i=1}^{n}\varepsilon^2 \\
\frac{\partial^2}{\partial \sigma^2 \partial T_0}LL & \frac{\partial^2}{\partial \sigma^2 \partial a}LL & \frac{\partial^2}{\partial \sigma^2 \partial \omega}LL & \frac{\partial^2}{\partial \sigma^2 \partial \theta}LL & \frac{\partial^2}{\partial (\sigma^2)^2}LL
\end{bmatrix} := [a_{ij}]$$

ただし

$$
\begin{aligned}
a_{11} &= 2n \\
a_{12} &= a_{21} = \Sigma_{i=1}^{n} 2\sin(\omega t_i + \theta) \\
a_{13} &= a_{31} = \Sigma_{i=1}^{n} 2at_i\cos(\omega t_i + \theta) \\
a_{14} &= a_{41} = \Sigma_{i=1}^{n} 2a\cos(\omega t_i + \theta) \\
a_{15} &= a_{25} = a_{35} = a_{45} = 0 \\
a_{22} &= \Sigma_{i=1}^{n} 2\sin^2(\omega t_i + \theta) \\
a_{23} &= a_{32} = \Sigma_{i=1}^{n} (2t_i(T_0 - t_i) + 4at_i\sin(\omega t_i + \theta))\cos(\omega t_i + \theta) \\
a_{24} &= a_{42} = \Sigma_{i=1}^{n} (2(T_0 - t_i) + 4a\sin(\omega t_i + \theta))\cos(\omega t_i + \theta) \\
a_{33} &= \Sigma_{i=1}^{n} 2at_i^2((t_i - T_0)\sin(\omega t_i + \theta) + a\cos(2(\omega t_i + \theta))) \\
a_{34} &= a_{43} = \Sigma_{i=1}^{n} at_i(2(t_i - T_0)\sin(\omega t_i + \theta) + a\cos(2(\omega t_i + \theta))) \\
a_{44} &= \Sigma_{i=1}^{n} 2a((t_i - T_0)\sin(\omega t_i + \theta) + a\cos(2(\omega t_i + \theta))) \\
a_{51} &= -\frac{1}{2\sigma^4}\Sigma_{i=1}^{n} 2(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
a_{52} &= -\frac{1}{2\sigma^4}\Sigma_{i=1}^{n} 2\sin(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
a_{53} &= -\frac{1}{2\sigma^4}\Sigma_{i=1}^{n} 2at_i\cos(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
a_{54} &= -\frac{1}{2\sigma^4}\Sigma_{i=1}^{n} 2a\cos(\omega t_i + \theta)(T_i - T_0 - a\sin(\omega t_i + \theta)) \\
a_{55} &= \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}\Sigma_{i=1}^{n} (T_i - T_0 - a\sin(\omega t_i + \theta))^2
\end{aligned}
$$

これを計算して得られた結果を下に記す. これは良さそうである. 局所解に引っかかってしまったら修正

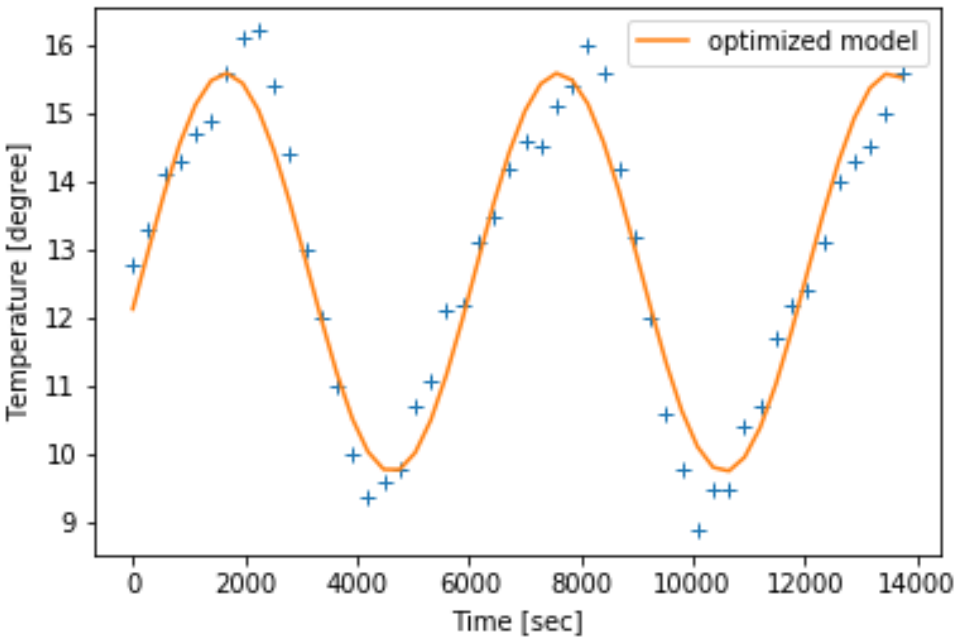

図 2　人工衛星の温度分布の最尤推定

オイラー法などを用いるつもりでいたが, 標本がフィッティング曲線の上にくる回数と下にくる回数が大体等しかったためこれで良いとした. ■

練習問題 4-7 (不完全データの最尤推定)

前ページの対数尤度関数 $LL(\theta)$ を数値的に最大化することによって, [例題 (発展)] の答えを求めよ.

Solution. ここでは, 男女の記録が混合されたソフトボール投げのデータを混合ガウス分布モデルによってそれぞれの人数の割合, 記録の平均および分散を推定する. 添字が F の場合は女子のものを指し, M のものは男子のものを指すとする. まず, 以下のように文字を定める.

$$\alpha = (\text{その性別の割合}) \quad \mu = (\text{その性別のの平均記録}) \quad \sigma^2 = (\text{その性別の分散})$$
$$\text{パラメータ}\theta = (\alpha_F, \mu_F, \sigma_F^2, \alpha_M, \mu_M, \sigma_M^2)$$

ただし, $\alpha_F + \alpha_M = 1, \alpha_F > 0, \alpha_M > 0$ であることに注意する. i 番目の生徒の記録 x_i の確率密度関数, ならびに対数尤度は次の通りになる. 今回は $LL(\theta)$ を最大化すなわち $-LL(\theta)$ を最小化することを考える ;

$$p(x_i|\theta) = \alpha_F \frac{1}{\sqrt{2\pi\sigma_F^2}} \exp\left(-\frac{(x_i-\mu_F)^2}{2\sigma_F^2}\right) + \alpha_M \frac{1}{\sqrt{2\pi\sigma_M^2}} \exp\left(-\frac{(x_i-\mu_M)^2}{2\sigma_M^2}\right)$$

$$\therefore LL(\theta) = \Sigma_{i=1}^n \log\left\{\alpha_F \frac{1}{\sqrt{2\pi\sigma_F^2}} \exp\left(-\frac{(x_i-\mu_F)^2}{2\sigma_F^2}\right) + \alpha_M \frac{1}{\sqrt{2\pi\sigma_M^2}} \exp\left(-\frac{(x_i-\mu_M)^2}{2\sigma_M^2}\right)\right\}$$

なお今回は代表パラメータとして α_F を用い, α_M は $1-\alpha_F$ で求める. 問題のヒントにある通り, scipy のメソッド optimize.differential evolution でも求めることができるが, これは局所解な上確率的なアルゴリズムの都合上考察できることに乏しい. そこで EM アルゴリズムを使って解く.
”パターン認識と機械学習”に証明の詳細が記載されているのでここではざっくりとした EM アルゴリズムの説明をする. まず $\mu_\mathbf{k}, \sigma_\mathbf{k}^2, \alpha_\mathbf{k}$ に適当な初期値を与える. 次に負担率 $\gamma = \dfrac{\alpha_k \mathcal{N}(x_n|\mu_k, \sigma_k^2)}{\Sigma_i \alpha_k \mathcal{N}(x_n|\mu_i, \sigma_i^2)}$ を求める. これによりパラメータを更新し, 収束するまで繰り返す. ■

表 1　EM アルゴリズムによるソフトボール投げの男女の最尤推定パラメータ

α_F	0.24474394463217622
α_M	0.75525605536
μ_F	15.729451928654447
μ_M	29.275252718511176
σ_F^2	19.517065414030384
σ_M^2	158.63396693878727

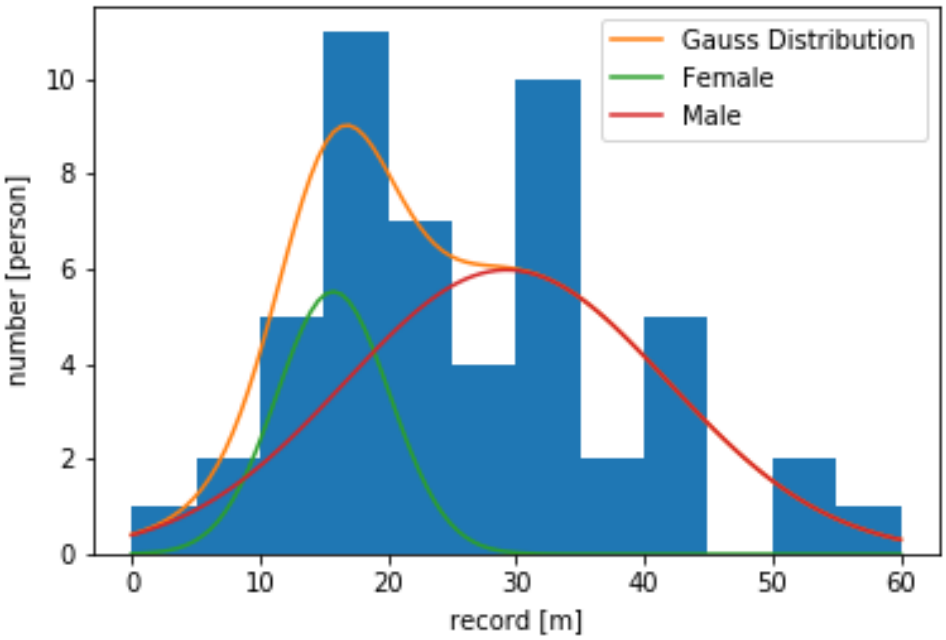

図 3　EM アルゴリズムによるソフトボール投げの男女の最尤推定グラフ

練習問題 5-2

「オッカムの剃刀」を題材にして, オリジナルの小話を作りなさい.

Solution. 何故 **Tiktok** は流行し **Vine** はクローズしたのか？

現在若者の間で Tiktok が流行っている. Tiktok は 15 秒以内の動画を音楽に合わせて撮影し投稿するプラットフォームである.(最近もっと長いコンテンツも出るようになってきた.) 一方 Vine は 6 秒以内の動画を (大体の場合音楽に合わせて) 撮影し投稿するプラットフォームである. 両者ともにターゲットは若者向けでコンセプトも非常に近いものだと思うが, Vine は数年前採算がとれないとしてサービスをクローズした. Tiktok は中 Bytedance 社が日本に進出してまだ一年ほどであるが, 国内の音楽サービス AWA や音楽レーベル大手エイベックスなどと組み一大コミュニティを形成した. ここでは両者の明暗がなぜ分かれたのか考察し, 21 世紀の小話としたい.

そもそも一般に SNS は利用者数以外で差がつきにくい業界である. 例えば米 Snap 社が提供する Snapchat の持っていた, 相手にメッセージが届くと既読ののち自動で消滅する機能は一世を風靡したもののストーリー機能として Facebook や Instagram に組み込まれたことでコモディティ化し価値を失った.(こうなるとより多くの利用者数を抱えている Facebook の方が有利になる.) これと同じことがかつて Vine にも起きた. Twitter が動画投稿機能をつけた後に Vine はサービスのクローズを決めたのだ.

Tiktok のユニークな人気の本質はオススメ機能にある. 自分がどの動画にいいねをつけたか, どれだけの時間視聴したかによってサービスの裏にあるアルゴリズムがリアルタイムで流行っている別の動画をリコメンドする. これ自体は珍しいことではなく, それこそ Facebook もニュースを薦めてくれる. だが Tiktok がユニークなのは, いいね数が伸びる前の動画は人が目を通し (!) どれが伸びそうかを判断して

いる点なのだそうだ. これは機械だと初期の段階で流行る動画を予想するのが難しいから導入された仕組みらしいが, 初めて聞いた時は AI を売りにしてるテクノロジー企業がマンパワーで流行りを判断していることを面白いと感じた. ところで Facebook はフェイクニュースを配信していることが発覚し昨年世の中を騒がせた. 原因は, AI が記事の信憑性を判断できないことだった. フェイクニュースの問題が出た時, 社内のエンジニアは最適配信アルゴリズムをどう改良すればこの問題を解決できるか考えたに違いない. それは多分, エンジニアの傲慢なのだと思う. Uber は自動運転を目指しているが, 現段階では自動運転の実現が困難であるとして人を雇って働かせている. どのようなアルゴリズムを組めばそれが可能になるのかは甚だ難しい課題の前では, おそらく一番シンプルな「人間を使う」という答えが正しい (少なくとも今の時点では.). 実際機械を用いた一番確実な方法は, ワシントンポストは ok で怪しい掲示板の類は全部ダメというように決められた情報ソースへのアクセスのみを許すことだ. そうしなければ, フェイクニュースは誇張された内容であることが多いので, 機械には人気のコンテンツにうつってしまう; これは極めて危険なことだと多くの書籍で警告されている.
機械による究極の最適化はいつか可能になるのかもしれないが, 統計というツールをベースに使っている限り過学習の可能性は避けられない. 一方で, システムに人間を組み込めば過学習することはない (裏を返せば将棋の名人でも機械に勝つことはできない.) だから今の時代を生きるエンジニアがオッカムの剃刀から学ぶことは, アルゴリズムを磨き続けることより, 全てを技術で解決できるというプライドを捨て人間を使うことが時に意味があるということだ. 僕たちは謙虚になり「人と機械が共存する社会」というものを考えることは, 実は機械が過学習した先にある不自由な世界を防ぐためにあるのだと思う. ■

練習問題 5-3 (ハッブルの法則)

下表は, エドウィン・ハッブルが「ハッブルの法則」(遠い銀河ほど高速で遠ざかっている) を発見するものになった観測データである. このとき, 定数項無しの単回帰モデル ($y = a_1 x + \varepsilon$) と, 定数項有りのモデル ($y = a_0 + a_1 x + \varepsilon$) それぞれを仮定したときのパラメータ値を推定せよ. また, AIC の値を比較して議論せよ.

Solution. 定数項なしでは

$$\hat{a_1} = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

定数項ありでは

$$\begin{bmatrix} \hat{a_0} \\ \hat{a_1} \end{bmatrix} = \begin{bmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{bmatrix}$$

また, AIC と AICc は以下の定義に従った. (今回サンプル数 n が小さいため念の為 AICc も求めた.) k はそれぞれ 1 と 2 である.

$$AIC = n \log \hat{\sigma^2} + 2k$$
$$AICc = n \log \hat{\sigma^2} + \frac{2kn}{n - 2k - 1}$$

グラフは以下のようになった.

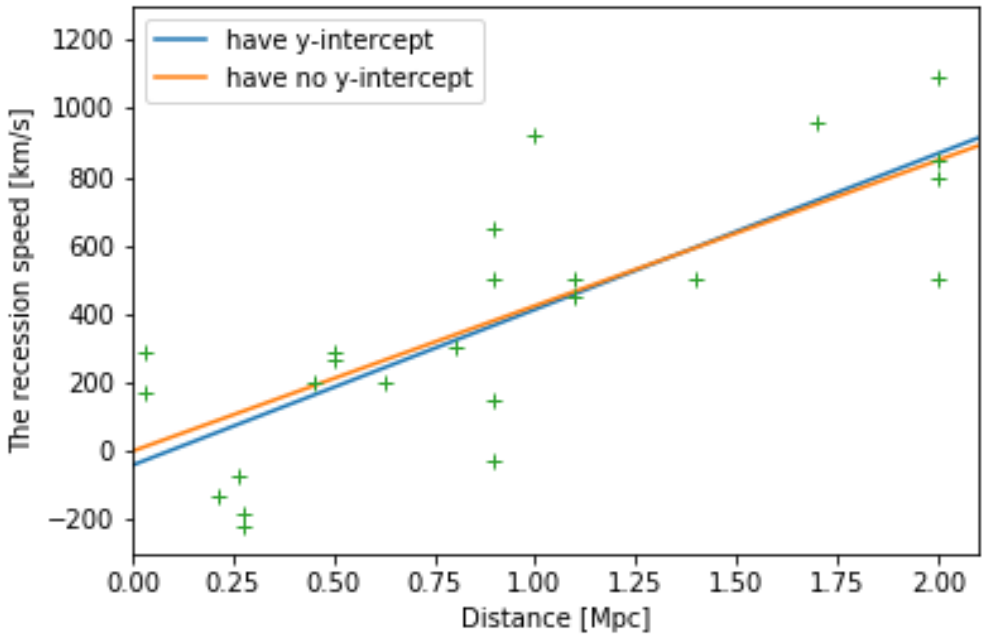

図 4　ハッブルの法則の単回帰モデルの定数項あり・なしの比較

また, AIC や AICc も含めた諸々の値は次の表のようになった.

表 2　ハッブルの観測データの単回帰のそれぞれのパラメータ

	$y = a_1 x + \epsilon$	$y = a_0 + a_1 x + \epsilon$
$\hat{a_0}$	-	-40.7836491
$\hat{a_1}$	423.9373232316303	454.15844092
$\hat{\sigma^2}$	50266.77658179018	49726.76526134
AIC	261.80239118062116	263.5431664857152
AICc	262.08810546633543	264.5957980646625

これはオッカムの剃刀の格好の題材である. 分散は定数項ありのモデルの方が小さい (合わせにいってるので当然である) が, モデル選択基準 AIC および n が小さい時の修正された基準 AICc はともに定数項なしの方が小さい. すなわち, 定数項なしの方がモデルとして優れているということになる. この結論は, 我々のよく知るところのハッブルの法則 $v = H_0 r$ を導いただけに留まらず, 高次の過度なフィッティングだけでなく定数項による近似でさえも時にモデルに過学習させてしまうことがあるということを示唆している. ■

練習問題 6-3 迷惑メール判定

ある人が受け取るメールについて, 以下のことが分かっているとする

- この人が受け取る全メールのうち, 25% が迷惑メールである.

- ”money”という単語を含むメールの割合は, 迷惑メールでは 10% であり, 非迷惑メールでは 3% である.
- ”click”という単語を含むメールの割合は, 迷惑メールでは 60% であり, 非迷惑メールでは 10% である.
- 迷惑メールであっても, 非迷惑メールであっても, 2 つの単語”money”と”click”が含まれる確率は独立であると仮定してよい.

このとき, 以下の問いに答えよ.

1. あるメールが, ”money”を含んでいるとき, このメールが迷惑メールである確率はいくらか?

   Solution. 迷惑メールである確率を $P(X)$, ”money”を含んでいる確率を $P(money)$ とする. ベイズの定理より,

   $$\begin{aligned} P(X|money) &= \frac{P(money|X)P(X)}{P(money|X)P(X) + P(money|\overline{X})P(\overline{X})} \\ &= \frac{0.1 \times 0.25}{0.1 \times 0.25 + 0.03 \times (1 - 0.25)} = \underline{0.53} \end{aligned}$$

   ■

2. あるメールが, ”money”と”click”を含んでいるとき, このメールが迷惑メールである確率はいくらか?

   Solution. 上の問題文で定義した内容に加え, ”click”を含んでいる確率を $P(click)$ とする. ベイズの定理より,

   $$P(X|money \wedge click) = \frac{P(money \wedge click|X)P(X)}{P(money \wedge click|X)P(X) + P(money \wedge click|\overline{X})P(\overline{X})}$$

   ”click”と”money”は独立変数であることから, 確率積の法則より,

   $$\begin{aligned} P(X|money \wedge click) &= \frac{P(money|X)P(click|X)P(X)}{P(money|X)P(click|X)P(X) + P(money|\overline{X})P(click|\overline{X})P(\overline{X})} \\ &= \frac{0.1 \times 0.6 \times 0.25}{0.1 \times 0.6 \times 0.25 + 0.03 \times 0.10 \times (1 - 0.25)} = \underline{0.87} \end{aligned}$$

   今, 2 では”click”と”money”が独立変数であると仮定した. だが迷惑メールの場合, 直感的に考えて”money”を貰うためにはこちらを”click”!のように明らかに相関があると思う. そこで自分の迷惑フォルダの中で, ”money”も”click”も含まれている確率を考えた時, これが 0.08 とし, 逆に迷惑メールでもないのに”money”も”click”も入っているメールはもう迷惑メールとカウントされ Gmail に自動で迷惑ボックスフォルダにクラスタリングされても文句は言えないと思うので 0.002 と置き直し計算してみる.

   $$\begin{aligned} P(X|money \wedge click) &= \frac{P(money \wedge click|X)P(X)}{P(money \wedge click|X)P(X) + P(money \wedge click|\overline{X})P(\overline{X})} \\ &= \frac{0.08 \times 0.25}{0.08 \times 0.25 + 0.002 \times (1 - 0.25)} = 0.93 \end{aligned}$$

こうなると 90% を超えてくるようだ. ”sign” (外国ではどうか分からないが, 日本では「今すぐ登録！」といった表記は散見される) など, 後 2, 3 ほど怪しい単語を足せば 99% 以上の精度にできそうなので, 案外スパムフィルタは簡単なアルゴリズムで出来ているのかもしれないことが推測できる. ■

練習問題 7-4 (ランダムウォーク位置推定)

ある変数 X_t が次式で表されるランダムウォークに従って変化するとする.

$$X_t = X_{t-1} + v_t$$

ここで, v_t は外乱 (プロセスノイズ) を表し, 平均 0, 分散 1 の正規分布 $N(0,1)$ に従い, かつ, 時刻間で無相関であるとする. また, 観測者は X_t の値そのものではなく, 次式で表される観測 Y_t の値を得ることができる.

$$Y_t = X_t + w_t$$

ここで w_t は観測ノイズを表し, やはり, 平均 0, 分散 1 の正規分布 $N(0,1)$ に従い, かつ, 時刻間で無相関であるとする. また, v_t と w_t も無相関であるとする. 今, 時刻 $t = 1$ において $p(X_1) = N(0,4)$ であることが分かっているとする. 以下の問いに答えよ.

1. Y_1 の分散 $\sigma^2_{Y_1}$ および, X_1 と Y_1 との共分散 $\sigma_{X_1Y_1}$ を求めよ.

   Solution.

   $$\begin{aligned}\sigma^2_{Y_1} &= V[Y_1] = V[X_1 + w_1] = V[X_1] + V[w_1] + 2Cov(X_1, w_1) = 4 + 1 + 0 = \underline{5} \\ \sigma_{X_1Y_1} &= E[X_1Y_1] - E[X_1Y_1] = E[X_1(X_1 + w_1)] - E[X_1]E[X_1 + w_1] \\ &= E[X_1^2] + E[X_1w_1] - (E[X_1])^2 - E[X_1]E[w_1] = E[X_1^2] - (E[X_1])^2 = V[X_1] = \underline{4}\end{aligned}$$

   ■

2. 時刻 $t = 1$ における観測値が $Y_1 = 2$ であったとき, $p(X_1|Y_1 = 2)$ が正規分布になることを示し, その平均と分散を求めよ.

   Proof.

   $$p(X_1|Y_1 = y) = \frac{p(X_1, Y_1 = y)}{p(Y_1 = y)}$$

   であり, $p(Y_1 = y)$ は Y_1 の周辺確率分布である.

   $$|\Sigma| = \sigma^2_{X_1}\sigma^2_{Y_1}\left(1 - \frac{\sigma^2_{X_1Y_1}}{\sigma^2_{X_1}\sigma^2_{Y_1}}\right) = \sigma^2_{X_1}\sigma^2_{Y_1}(1 - \rho)$$

となる (ただし ρ は X_1, Y_1 の相関係数.). よって二次元正規分布同時確率密度関数は,

$$
\begin{aligned}
p(X_1, Y_1) &= N\left(\begin{bmatrix} \mu_{X_1} \\ \mu_{X_2} \end{bmatrix}, \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1Y_1} \\ \sigma_{X_1Y_1} & \sigma_{Y_1}^2 \end{bmatrix}\right) \\
&= \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left[-\frac{\sigma_{Y_1}^2(X_1-\mu_{X_1})^2 - 2\rho\sigma_{X_1}\sigma_{Y_1}(X_1-\mu_{Y_1})(X_1-\mu_{Y_1}) + \sigma_{X_1}^2(Y_1-\mu_{Y_1})^2}{2|\Sigma|}\right] \\
\Rightarrow p(X_1, Y_1 = y) &= \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left[-\frac{\sigma_{Y_1}^2\left(X_1 - \left(\mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{Y_1}}\right)\right)^2 + (1-\rho)\sigma_{X_1}^2(y-\sigma_{Y_1}^2)}{2|\Sigma|}\right]
\end{aligned}
$$

これを x について広義積分する.

$$
\begin{aligned}
p(Y_1 = y) &= \int_{-\infty}^{\infty} p(X_1, Y_1 = y)dx = \frac{1}{\sqrt{2\pi\sigma_{Y_1}^2}} \exp\left[-\frac{(y-\mu_{Y_1})^2}{\sigma_{Y_1}^2}\right] \\
&= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_{X_1}^2(1-\rho^2)}} \exp\left[-\frac{(X_1 - (\mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{Y_1}}(y-\mu_{Y_1})))^2}{2\sigma_{X_1}^2(1-\rho^2)}\right] dX_1 \\
&= \frac{1}{\sqrt{2\pi\sigma_{Y_1}^2}} \exp\left[-\frac{(y-\mu_{Y_1}^2)}{\sigma_{Y_1}^2}\right] \cdot \frac{1}{\sqrt{2\pi\sigma_{X_1}^2(1-\rho^2)}} \sqrt{2\sigma_{X_1}^2(1-\rho^2)}\,\sqrt{\pi} \\
&= \frac{1}{\sqrt{2\pi\sigma_{Y_1}^2}} \exp\left[-\frac{(y-\mu_{Y_1}^2)}{\sigma_{Y_1}^2}\right]
\end{aligned}
$$

よって,

$$
\begin{aligned}
p(X_1|Y_1 = y) &= \frac{p(X_1, Y_1 = y)}{p(Y_1 = y)} = \frac{1}{\sqrt{2\pi\sigma_{X_1}^2(1-\rho^2)}} \left[-\frac{\sigma_{Y_1}^2(X_1 - (\mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{Y_1}}(y-\mu_{Y_1})))^2}{2|\Sigma|}\right] \\
&= \frac{1}{\sqrt{2\pi\sigma_{X_1}^2(1-\rho^2)}} \exp\left[-\frac{(X_1 - (\mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{Y_1}}(y-\mu_{Y_1})))^2}{2\sigma_{X_1}^2(1-\rho^2)}\right] \\
&= \frac{1}{\sqrt{2\pi\hat{\sigma_{X_1}}}} \exp\left[-\frac{(X_1 - \hat{\mu_{X_1}})^2}{2\hat{\sigma}_{X_1}^2}\right]
\end{aligned}
$$

を得る. したがって条件つき確率密度関数 $p(X_1|Y_1 = y)$ は正規分布である. □

Solution. 以上より, 求める期待値および分散は,

$$
\begin{aligned}
E[X_1] &= \mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{Y_1}}(2-\mu_{Y_1}) = \mu_{X_1} + \frac{\sigma_{X_1Y_1}}{\sigma_{Y_1}^2}(2-\mu_{Y_1}) = 0 + \frac{4}{5}(2-0) = \underline{1.6} \\
V[X_1] &= \sigma_{X_1}^2(1-\rho^2) = \sigma_{X_1}^2 - \frac{\sigma_{X_1Y_1}^2}{\sigma_{Y_1}^2} = 4 - \frac{4^2}{5} = \underline{0.8}
\end{aligned}
$$

■

3. 上記のとき, 時刻 $t = 2$ における X_2 の確率密度 (予測確率密度)$p(X_2|Y_1 = 2)$ を求めよ.

Solution. 正規分布の再生性より,

$$E[X_2] = E[X_1] + E[v_2] = 1.6 + 0 = 1.6$$
$$V[X_2] = V[X_1] + V[v_2] = 0.8 + 1 = 1.8$$

■

4. 時刻 $t = 2$ における観測値が $Y_2 = 1$ であったとする. このとき, 確率密度分布 $p(X_2 Y_1 = 2, Y_2 = 1)$ が正規分布になることを示し, その平均と分散を求めよ.

Proof.

$$\begin{aligned} p(X_2|Y_1 = y_1, Y_2 = y_2) &= \frac{p(X_2, Y_1 = y_1, Y_2 = y_2)}{p(Y_1 = y_1, Y_2 = y_2)} \\ &= \frac{p(X_2, Y_2 = y_2|Y_1 = y_1) \cdot p(Y_1 = y_1)}{p(Y_2 = y_2|Y_1 = y_1) \cdot p(Y_1 = y_1)} \\ &= \frac{p(X_2, Y_2 = y_2, Y_1 = y_1)}{p(Y_2 = y_2, Y_1 = y_1)} \end{aligned}$$

より, $p(X_1) \to p(X_2|Y_1 = 2), X_1 \to X_2, Y_1 \to Y_2, p(X_1|Y_1 = y) \to p(X_2|Y_1 = y_1, Y_2 = y_2)$ と置き換え先ほどと同様の積分と入れ替えを行うことで $p(X_2|Y_1 = y_1, Y_2 = y_2)$ は正規分布に従うことが示される. □

Solution.

$$\begin{aligned} \sigma^2_{Y_2} &= V[Y_2] = V[X_2 + w_2] = V[p(X_2|Y_1 = 2)] + V[w_2] + 2Cov[X_2.w_1] = 1.8 + 1 + 0 = 2.8 \\ \sigma_{X_2 Y_2} &= E[X_2 Y_2] - E[X_2]E[Y_2] = E[X_2(X_2 + w_2)] - E[X_2]E[X_2 + w_2] = E[X_2^2] - (E[X_2])^2 \\ &= V[p(X_2|Y_1 = 2)] = 1.8 \\ \sigma_{X_2} &= E[p(X_2|Y_1 = 2)] = 1.6 \\ \sigma_{Y_2} &= E[X_2 + w_2] = E[p(X_2|Y_1 = 2)] + E[w_2] = 1.6 + 0 = 1.6 \end{aligned}$$

$$\therefore E[X_2] = \mu_{X_2} + \rho\frac{\sigma_{X_2}}{\sigma_{Y_2}}(1 - \mu_{Y_2}) = \mu_{X_2} + \frac{\sigma_{X_2 Y_2}}{\sigma^2_{Y_2}}(1 - \mu_{Y_2}) = 1.6 + \frac{1.8}{2.8}(1 - 1.6) = \underline{1.2}$$

$$\therefore V[X_2] = \sigma^2_{X_2}(1 - \rho^2) = \sigma^2_{X_2} - \frac{\sigma^2_{X_2 Y_2}}{\sigma^2_{Y_2}} = 1.8 - \frac{1.8^2}{2.8} = \underline{0.643}$$

■

練習問題 8-1

標準正規分布 $N(0, 1)$ からランダムサンプリングが可能であるとする. このとき, 任意の 2 変量ガウス分布,

$$N(\mathbf{x}|\mu, \mathbf{\Sigma}) = p\left(\begin{bmatrix} x \\ y \end{bmatrix} | \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \sigma^2_x & \sigma_{xy} \\ \sigma_{xy} & \sigma^2_y \end{bmatrix}\right)$$

のサンプル (x, y) を得る方法を考えよ. また, 実際にプログラミングして結果を確かめよ.

Solution. 一次元標準正規分布から, n 次元標準正規分布に従う確率分布 $\mathrm{Z} \sim \mathcal{N}(0, I)$ を作ることができる. 以下の変換により多次元正規分布を得ることができる.
多次元標準正規分布に従う $\mathrm{Z} \sim \mathcal{N}(0, I)$ に対角行列 D をかける;

$$\mathbf{X} := D\mathbf{Z} \sim \mathcal{N}(0, D^2)$$

$\mathbf{X}$ に直交行列 Q を書けて $\mathbf{Y} = Q\mathbf{X}$ とすると,

$$E[\mathbf{Y}] = QE[\mathbf{X}] = \mathbf{0}$$
$$V[\mathbf{Y}] = QV[\mathbf{X}]Q^T = QD^2Q^T$$

$\mathbf{Y} \sim \mathrm{N}(0, V), \hat{\mathbf{Y}} := \mathbf{Y} + \mu \Rightarrow \hat{\mathbf{Y}} \sim \mathrm{N}(\mu, V)$. また, $V = QD^2Q^T$ の両辺に左から Q^T, 右から Q をかけると

$$Q^T V Q = Q^T(QD^2Q^T)Q = ID^2I = D^2$$

よって対称行列 V の直交行列 Q による対角化で D と Q を求めることができる. このようにして求めた散布図は以下の通り. ■

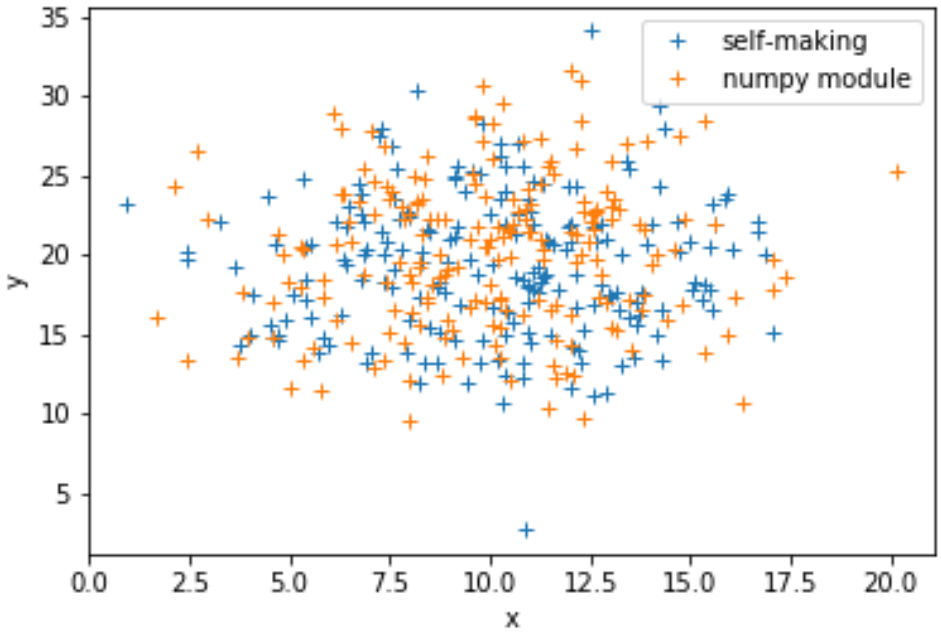

図 5　自作関数によるランダムプロット (n=200)

Appendix

プログラム 1　Q.1-2

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
```

```
Created on Wed Jan 30 21:23:12 2019

@author: yohei
"""

import numpy as np
import matplotlib.pyplot as plt

A = np.array([[0, 1], [-1, -0.3]])
B = np.array([[0], [1]])
eigenvalue, eigenvector = np.linalg.eig(A)
a, b = np.real(eigenvalue[0]), np.imag(eigenvalue[0])
P = np.array([[1, 1],[eigenvalue[0], eigenvalue[1]]])
P_inv = np.linalg.inv(P)
temp = np.dot(P_inv, A)
D = np.dot(temp, P)

def ideal(t):
    func = 83/78*np.exp(a*t)*(np.cos(b*t)+1249/(1560*b)*np.sin(b*t)) - 5/78*(np
        .cos(2*t)+5*np.sin(2*t))
    return func

def A_p(delta):
    coef = -np.exp(a*delta)/b
    A_prime = np.array([[coef*(a*np.sin(b*delta)-b*np.cos(b*delta)), coef*(-np.
        sin(b*delta))],
                        [coef*(np.sin(b*delta)), coef*(-a*np.sin(b*delta)-b*np.cos(
                            b*delta))]])
    return A_prime

def B_p(delta):
    A_inv = np.linalg.inv(A)
    middle_factor = A_p(delta) - np.array([[1,0],[0,1]])
    temp = np.dot(A_inv, middle_factor)
    B_prime = np.dot(temp, B)
    return B_prime

def vectolizer(delta):
    first_vec = np.array([[1],
                          [0]])
    vec_possess = []
    for i in range(int(100/delta)):
        u = np.sin(2*i*1.0/delta)
        next_vec = np.dot(A_p(delta), first_vec) + np.dot(B_p(delta), u)
        vec_possess.append(next_vec)
```

```
        first_vec = next_vec
    yn = []
    for i in range(int(100/delta)):
        yn.append(vec_possess[i][0])
    return yn

t = np.linspace(0, 100, 100)
y1 = ideal(t)
y2 = vectolizer(1.0)
plt.plot(t, y1, label="analytical answer")
plt.plot(t, y2, label="delta1.00 answer")
plt.xlabel("time [s]")
plt.ylabel("displacement [m]")
plt.legend()
plt.show()
```

プログラム 2　Q.4-6

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 31 06:22:29 2019

@author: yohei
"""

import numpy as np
import csv
import matplotlib.pyplot as plt

t = np.arange(0,14000,280)
T = np.zeros(50)
with open('satellite-temp.csv', 'r') as f:
    reader = csv.reader(f)
    i = 0
    for row in reader:
        T[i] += float(row[1])
        i += 1

x0 = np.array([[np.mean(T)],[2.0],[2.0*np.pi/6000],[0],[0.1]])
x1 = np.array([[0],[0],[0],[0],[0]])
n = T.size

while (np.linalg.norm(x1-x0) > 0.00001):
    A = np.zeros([5,5])
    f_x = np.zeros([5,1])
```

```
    A[0][0] = 2 * n
    for i in range(n):
        A[0][1] += 2*np.sin(x0[2]*t[i]+x0[3])
        A[0][2] += 2*x0[1]*t[i]*np.cos(x0[2]*t[i]+x0[3])
        A[0][3] += 2*x0[1]*np.cos(x0[2]*t[i]+x0[3])
        A[1][1] += 2*np.sin(x0[2]*t[i]+x0[3])**2
        A[1][2] += 2*t[i]*(x0[0]-T[i])*np.cos(x0[2]*t[i]+x0[3])+2*x0[1]*t[i]*np
            .sin(2*(x0[2]*t[i]+x0[3]))
        A[1][3] += 2*(x0[0]-T[i])*np.cos(x0[2]*t[i]+x0[3])+2*x0[1]*np.sin(2*(x0
            [2]*t[i]+x0[3]))
        A[2][2] += 2*x0[1]*t[i]**2*(T[i]-x0[0])*np.sin(x0[2]*t[i]+x0[3])+2*x0
            [1]**2*t[i]**2*np.cos(2*(x0[2]*t[i]+x0[3]))
        A[2][3] += 2*x0[1]*t[i]*(T[i]-x0[0])*np.sin(x0[2]*t[i]+x0[3])+x0[1]**2*
            t[i]*np.cos(2*(x0[2]*t[i]+x0[3]))
        A[3][3] += 2*x0[1]*(T[i]-x0[0])*np.sin(x0[2]*t[i]+x0[3])+2*x0[1]**2*np.
            cos(2*(x0[2]*t[i]+x0[3]))
        A[4][0] += 1.0/2/x0[4]**2*(-2)*(T[i]-x0[0]-x0[1]*np.sin(x0[2]*t[i]+x0
            [3]))
        A[4][1] += 1.0/2/x0[4]**2*(-2)*np.sin(x0[2]*t[i]+x0[3])*(T[i]- x0[0]-x0
            [1]*np.sin(x0[2]*t[i]+x0[3]))
        A[4][2] += 1.0/2/x0[4]**2*(-2)*x0[1]*t[i]*np.cos(x0[2]*t[i]+x0[3])*(T[i
            ]-x0[0]-x0[1]*np.sin(x0[2]*t[i]+x0[3]))
        A[4][3] += 1.0/2/x0[4]**2*(-2)*x0[1]*np.cos(x0[2]*t[i]+ x0[3])*(T[i]-x0
            [0]-x0[1]*np.sin(x0[2]*t[i]+x0[3]))
        A[4][4] += 1.0/2/x0[4]**2-1.0/x0[4]**3*(T[i]-x0[0]-x0[1]*np.sin(x0[2]*t
            [i]+x0[3]))**2
        f_x[0] += -2*(T[i]-x0[0]-x0[1]*np.sin(x0[2]*t[i]+x0[3]))
        f_x[1] += -2*np.sin(x0[2]*t[i]+x0[3])*(T[i]-x0[0]-x0[1]*np.sin(x0[2]*t[i
            ]+x0[3]))
        f_x[2] += -2*x0[1]*t[i]*np.cos(x0[2]*t[i]+x0[3])*(T[i]-x0[0]-x0[1]*np.
            sin(x0[2]*t[i]+x0[3]))
        f_x[3] += -2*x0[1]*np.cos(x0[2]*t[i]+x0[3])*(T[i]-x0[0]-x0[1]*np.sin(x0
            [2]*t[i]+x0[3]))
        f_x[4] += -1.0/2/x0[4]+1.0/2/x0[4]**2*(T[i]-x0[0]-x0[1]*np.sin(x0[2]*t[
            i]+x0[3]))**2
    A[1][0] = A[0][1]
    A[2][0] = A[0][2]
    A[3][0] = A[0][3]
    A[2][1] = A[1][2]
    A[3][1] = A[1][3]
    A[3][2] = A[2][3]
    A = np.linalg.inv(A)
    x1 = x0
    x0 = x1 - np.dot(A,f_x)

```

```
T1 = x0[0] + x0[1]*np.sin(x0[2]*t+x0[3])
print (x0)

plt.plot(t,T,"+")
plt.plot(t,T1, label="optimized model")
plt.xlabel("Time [sec]")
plt.ylabel("Temperature [degree]")
plt.legend()
#plt.show()
plt.savefig('p3.png')
```

プログラム 3　Q.4-7

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 29 10:07:02 2019

@author: yohei
"""

import numpy as np
import csv
from scipy.stats import norm
from scipy.optimize import differential_evolution
import matplotlib.pyplot as plt

T=np.zeros(50)
with open('ballthrow.csv', 'r') as f:
    reader = csv.reader(f)
    i = 0
    for row in reader:
        T[i] += float(row[0])
        i += 1

def LL(x):
    a = -sum([np.log(x[0]*1/np.sqrt(2*np.pi*x[3])*np.exp(-(T[i]-x[1])**2/(2*x
        [3]))+(1-x[0])*1/np.sqrt(2*np.pi*x[4])*np.exp(-(T[i]-x[2])**2/(2*x[4])
        ))for i in range(50)])
    return a

def dif_ev():
    limit = [(0, 1), (0, 50), (0, 50), (0, 200), (0, 200)]
    result = differential_evolution(LL, limit)
    print(result.fun)
    print(result.x)
```

```
32      return result.x
33
34  def gauss_dist(meter, x):
35      return [x[0]*norm.pdf(x=meter, loc=x[1], scale=np.sqrt(x[3])) \
36              +(1-x[0])*norm.pdf(x=meter, loc=x[2], scale=np.sqrt(x[4])),
37              x[0]*norm.pdf(x=meter, loc=x[1], scale=np.sqrt(x[3])),
38              (1-x[0])*norm.pdf(x=meter, loc=x[2], scale=np.sqrt(x[4]))]
39
40  def plot(x):
41      fig = plt.figure()
42      ax = fig.add_subplot(1, 1, 1)
43      ax.set_xlabel("record [m]")
44      ax.set_ylabel("number [person]")
45      ax.hist(T, bins=[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60])
46      meter_list = np.linspace(0, 60, 100)
47      ax.plot(meter_list, 250 * gauss_dist(meter_list, x)[0], label="Gauss
            Distribution")
48      ax.plot(meter_list, 250 * gauss_dist(meter_list, x)[1], label="Female")
49      ax.plot(meter_list, 250 * gauss_dist(meter_list, x)[2], label="Male")
50      ax.legend()
51      plt.savefig("p4.png")
52
53  lista = [0]
54
55  def step1(x):
56      temp = np.array([[(lambda y: x[0] if y == 0 else 1 - x[0])(j)
57      * norm.pdf(x=T[i], loc=x[j + 1], scale=np.sqrt(x[j + 3])) /
58      (x[0] * norm.pdf(x=T[i], loc=x[1], scale=np.sqrt(x[3])) +
59       (1 - x[0]) * norm.pdf(x=T[i], loc=x[2], scale=np.sqrt(x[4])))
60      for i in range(len(T))] for j in range(2)]).T
61      return temp
62
63  def step2(x, temp):
64      N_F = sum(temp[:, 0])
65      N_M = sum(temp[:, 1])
66      muf_next, mum_next = 1 / N_F * sum(temp[:, 0] * T), 1 / N_M * sum(temp[:, 1]
            * T)
67      sigmaf_next,sigmam_next = 1 / N_F * sum(temp[:, 0] * ((T - muf_next) ** 2)),
            1 / N_M * sum(temp[:, 1] * ((T - mum_next) ** 2))
68      piF = 1 / len(T) * sum(temp[:, 0])
69      x_new = [piF, muf_next, mum_next, sigmaf_next, sigmam_next]
70      return x_new
71
72  def iterate(x):
73      temp = step1(x)
```

```
    x = step2(x, temp)
    likeli_val = LL(x)
    lista.append(likeli_val)
    return x, likeli_val

x = [0.5, 15, 30, 20, 165]
for i in range(1000):
    x, likeli_val = iterate(x)
print(x)
print(dif_ev())
plot(x)
```

プログラム 4　Q.5-3

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 30 19:11:58 2019

@author: yohei
"""

import numpy as np
import csv
import matplotlib.pyplot as plt

own1 = np.zeros((2,2))
own2 = np.zeros((2,1))

X, Y = [], []
data1, data2 = [], []
with open('Hubbles_constant.csv', 'r') as f:
    reader = csv.reader(f)
    header = next(reader)
    for row in reader:
        data1.append(float(row[0]))
        data2.append(float(row[1]))
        own1[0][0] += 1
        own1[0][1] += float(row[0])
        own1[1][0] += float(row[0])
        own1[1][1] += float(row[0])**2
        own2[0][0] += float(row[1])
        own2[1][0] += float(row[0])*float(row[1])
        X.append(float(row[0]))
        Y.append(float(row[1]))
    own1_inv = np.linalg.inv(own1)
```

```
    answer = np.dot(own1_inv, own2)
    nown = own2[1][0]/own1[1][1]

def own(x):
    temp = answer[0]+answer[1]*x
    return temp

def noown(x):
    return nown*x

def var(datax, datay):
    temp1, temp2 = 0, 0
    for i in range(len(datax)):
        temp1 += (own(datax[i]) - datay[i])**2
        temp2 += (noown(datax[i]) - datay[i])**2
    return temp1/len(datax), temp2/len(datax)

var1, var2 = var(data1, data2)
AIC1 = len(data1)*np.log(var1[0]) + 2*2
AIC2 = len(data1)*np.log(var2) + 2*1
AICc1 = len(data1)*np.log(var1[0]) + 2*2*len(data1)/(len(data1)-2*2-1)
AICc2 = len(data1)*np.log(var2) + 2*1*len(data1)/(len(data1)-2*1-1)

x = np.linspace(0, 2.1, 100)
y1 = own(x)
y2 = noown(x)
plt.plot(x, y1, label="have y-intercept")
plt.plot(x, y2, label="have no y-intercept")
plt.plot(X, Y, '+')
plt.xlim(0, 2.1)
plt.ylim(-300, 1300)
plt.xlabel("Distance [Mpc]")
plt.ylabel("The recession speed [km/s]")
plt.legend()
plt.savefig('p2.png')
```

プログラム 5　Q.8-1

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 31 14:26:26 2019

@author: yohei
"""

```

```
import numpy as np
import matplotlib.pyplot as plt

def multi(mean, sigma):
    z = np.array([np.random.randn() for i in range(2)])
    u, d, v = np.linalg.svd(sigma)
    D = np.diag(np.sqrt(d))
    x = np.dot(D, z)
    y = np.dot(np.linalg.inv(u), x)
    y += mean
    return y

mean = [10, 20]
sigma = np.array([[10, 1],[1, 20]])
fig = plt.figure()
x = np.array([multi(mean, sigma) for i in range(200)])
y = np.array([np.random.multivariate_normal(mean, sigma) for i in range(200)])
plt.plot(x[:, 0], x[:, 1], "+", label="self-making")
plt.plot(y[:, 0], y[:, 1], "+", label="numpy module")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.savefig('p5.png')
```

Reference

パターン認識と機械学習 (上)(下)

人工知能はどのようにして「名人」を超えたのか？

The four GAFA 四騎士が塗り替えた世界

そのほかたくさんの Qiita