一般化と関数近似

今までやってきたこと

- テーブル方式（表方式）

　　　状態と行動の組1つ1つに対し、その価値を求める

→状態と行動の組が多くなるとメモリ量的にも計算量的にも不可能

　　　一般化しよう！

	a1	a2	a3
s1	2	5	1
s2	3	-10	4
s3	1	8	2

関数近似

- 状態価値関数V_tを表形式ではなく、$\overrightarrow{\theta_t}$をパラメータベクトルとするパラメータ関数として表す。

Ex) $\overrightarrow{\theta_t}$を結合比重として持つニューラルネットワーク

特徴ベクトル$\vec{x}$との内積　　$V(\vec{x}) = \overrightarrow{\theta_t}^T \overrightarrow{x}$

関数の形は変えずに$\overrightarrow{\theta_t}$を改善することによって、近似性能を上げる

$\overrightarrow{\theta_t}$の要素数は状態数よりはるかに小さく、パラメータ1つの変更で全体の推定値が大きく変わる。

平均二乗誤差(MSE) ～近似関数の評価～

- $\overrightarrow{\theta_t}$によって定まる状態価値関数$V_t(s)$の性能を以下で評価する

$$MSE\left(\overrightarrow{\theta_t}\right) = \sum_{s \in S} P(s)[V^{\pi}(s) - V_t(s)]^2$$

P(s)は重みであり、より誤差を小さくしたいsに対しては、この重みを大きく設定する。一般にP(s)は試行中に状態sが現れる確率とすることが多い。

方策オン型学習では、その状態sが出てくるたびに$\overrightarrow{\theta_t}$を更新していけばMSEをP(s)に従って小さくすることができる。

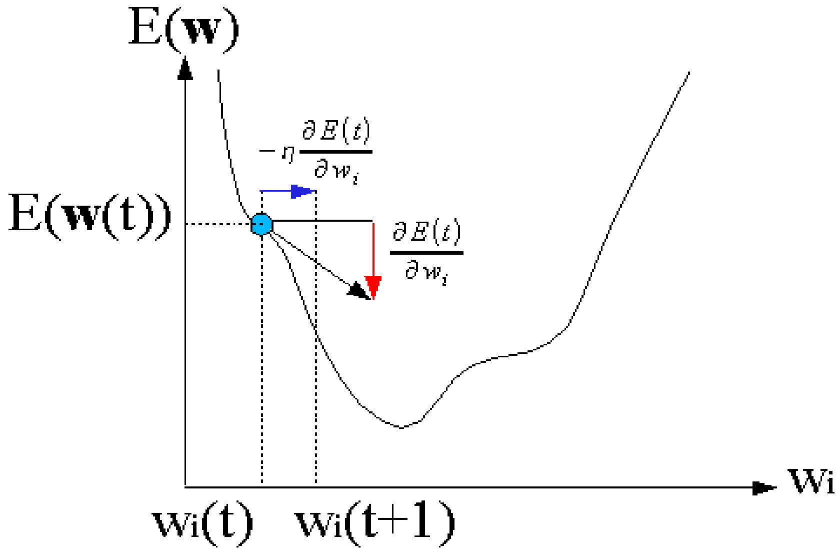
大域最適解には収束しないが、局所最適解には収束することが保証されている

最急降下法

- 実際の試行の中で以下で$\overrightarrow{\theta_t}$を更新していく方法

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \frac{1}{2}\alpha\nabla_{\vec{\theta}_t}[V^{\pi}(s) - V_t(s)]^2$$
$$= \vec{\theta}_t + \alpha[V^{\pi}(s) - V_t(s)]\nabla_{\vec{\theta}_t}V_t(s)$$

α: 学習率(0<α<1)徐々に小さくしていく



最急降下法(前方観測的見方)

- 最適状態価値関数$V^{\pi}(s)$は学習過程では分からない

→今まで使ってきた更新式(推定値)を代わりに用いる

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha[v_t - V_t(s)]\nabla_{\vec{\theta}_t} V_t(s)$$

推定値を用いた時の収束は推定値に偏りがない時のみ保証される

つまり、$E\{v_t\} = V^{\pi}(s)$のとき収束する

たとえばTD(λ)を用いるなら$v_t = R_t^{\lambda}$

λ<1のとき、これは偏りがあるので収束の保証はない

しかし、これは効果的で異なる種類の性能保証がある

最急降下法（後方観測的見方）

- TD(λ)の例を用いる

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha\delta_t\vec{e}_t$$

$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$ …TD誤差

$\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \nabla_{\vec{\theta}_t} V_t(s)$ …適格度トレース

最急降下法

$\vec{\theta}$を任意に初期化し$\vec{e} = \vec{0}$とする
各エピソードに対して繰り返し:
　　s←エピソードの初期状態
　　エピソードの各ステップに対して繰り返し:
　　　　a←sに対してπで与えられる行動
　　　　行動aをとり、報酬rと結果の状態s′を観測する
　　　　δ←r+γV(s’)-V(s)
　　　　$\vec{e} \leftarrow \gamma\lambda\vec{e} + \nabla_{\vec{\theta}} V_t(s)$
　　　　$\vec{\theta} \leftarrow \vec{\theta}$+αδ $\vec{e}$
　　　　s←s’
　　sが終端状態ならば繰り返しを終了

線形手法

- 勾配法を用いるプログラムには大きく二種類ある

①逆誤差伝播法(いわゆるニューラルネット)

②線形手法…特徴ベクトル$\vec{\phi}_s$とパラメータベクトル$\vec{\theta}_t$の内積を用いる

$$V(s) = \overrightarrow{\theta_t}^T \vec{\phi}_s$$

$\nabla_{\vec{\theta}_t} V_t(s) = \vec{\phi}_s$と、偏微分がとても簡単な式で表わされる。

また、最適解がただ一つしかないので、局所最適解⇒大域最適解

TD(λ)のMSE

- TD(λ)は一般に収束保証がないと紹介したが、以下の式が成り立つ

TD(λ)によって得られる$\vec{\theta}_\infty$と、大域最適解$\vec{\theta}^*$について、

$$MSE(\vec{\theta}_\infty) \leq \frac{1-\lambda\gamma}{1-\gamma} MSE(\vec{\theta}^*)$$

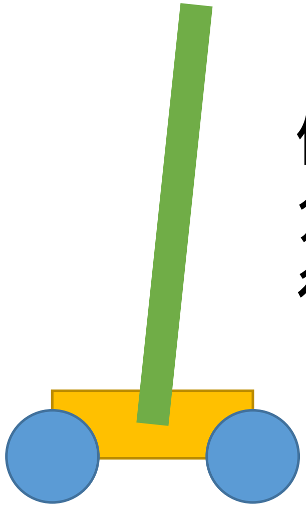
よってλ→1とすれば大域最適解に限りなく近づく

特徴ベクトルの選び方

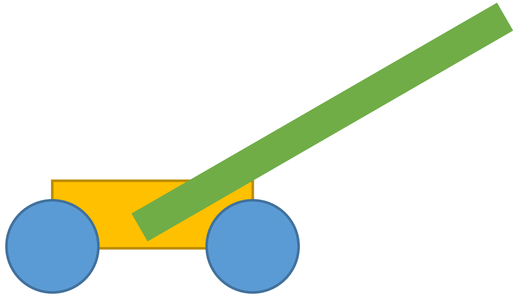
- 特徴ベクトルは当然、状態をよく表したものとならねばならない。
- また、各特徴が互いと無関係でなければならない。

NG例

ボールバランシング問題にて、角速度と棒の傾き



傾き　小
角速度　大
復元力の働くいい状態



傾き　大
角速度　大
今にも倒れそうな悪い状態

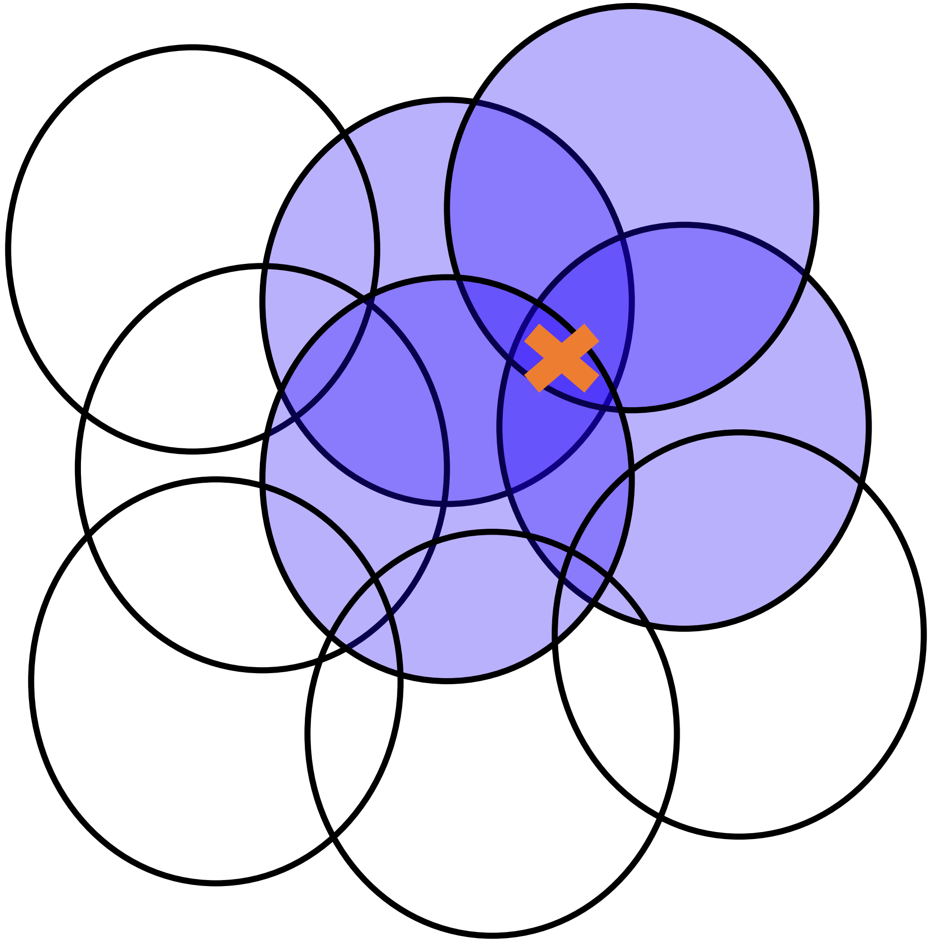
粗いコード化

二次元の連続量に対し、その二次元を覆うような円盤を考える。

円盤の内部に点があれば1、なければ0とすると、円盤の個数分の01のバイナリデータが得られる。

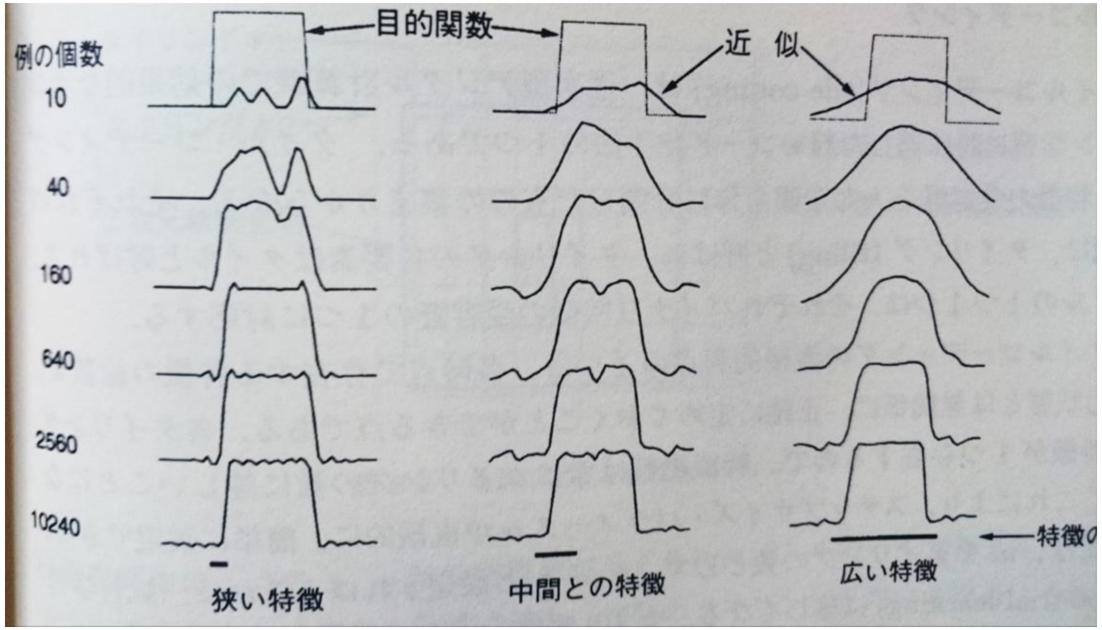
このバイナリデータを特徴として、その重みを学習させていく。(線形学習)

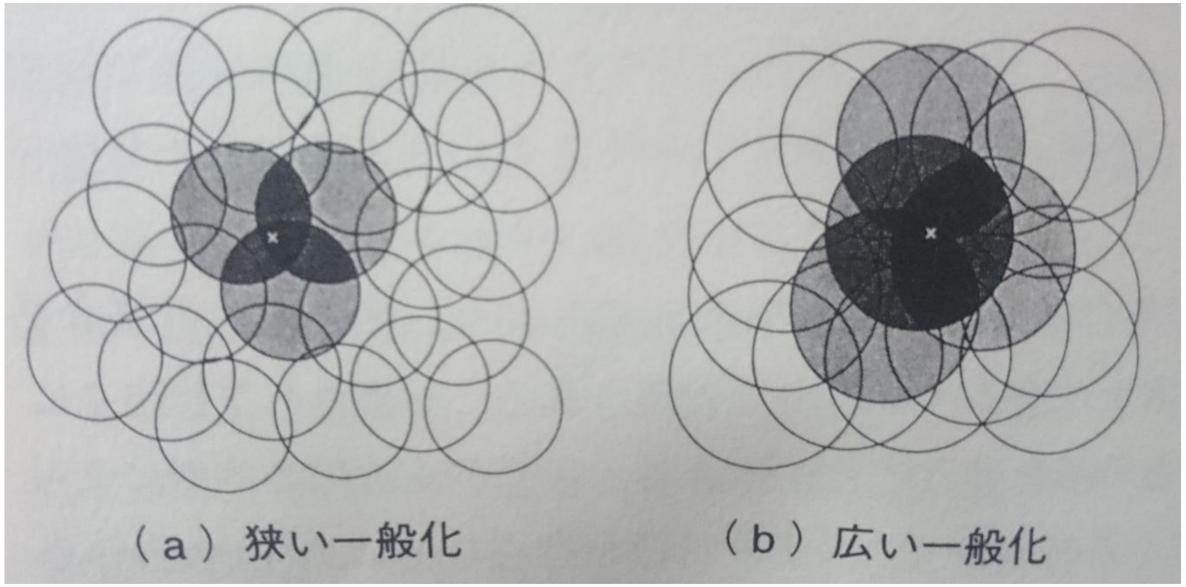
コード化の形は円盤でなくてもよい。



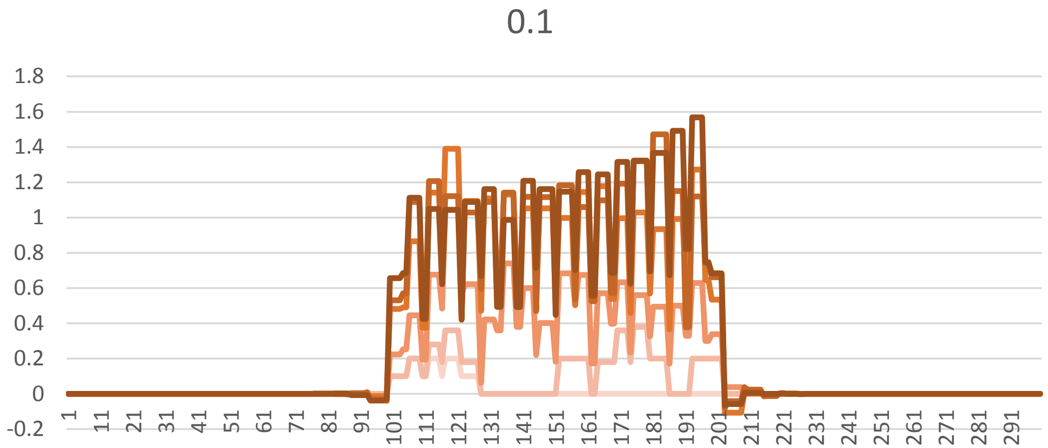
コード化の粗さ

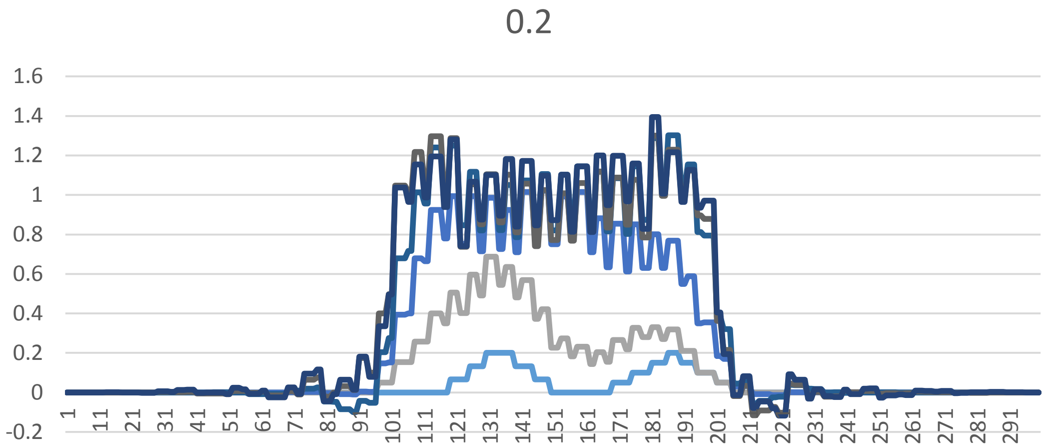
- 円盤の大きさと学習にはどのような関係性があるのか？可能な識別の細かさは実は円盤の大きさには寄らず、円盤の数で決まる。
- 円盤の大きさが大きいほど一回の更新で変更されるパラメーターが多くなるため、初期段階において近似関数がなだらかになる。

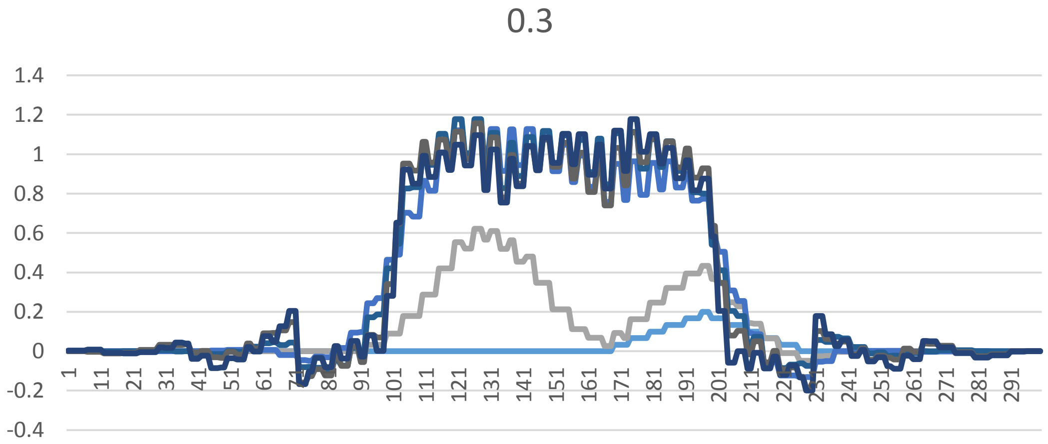


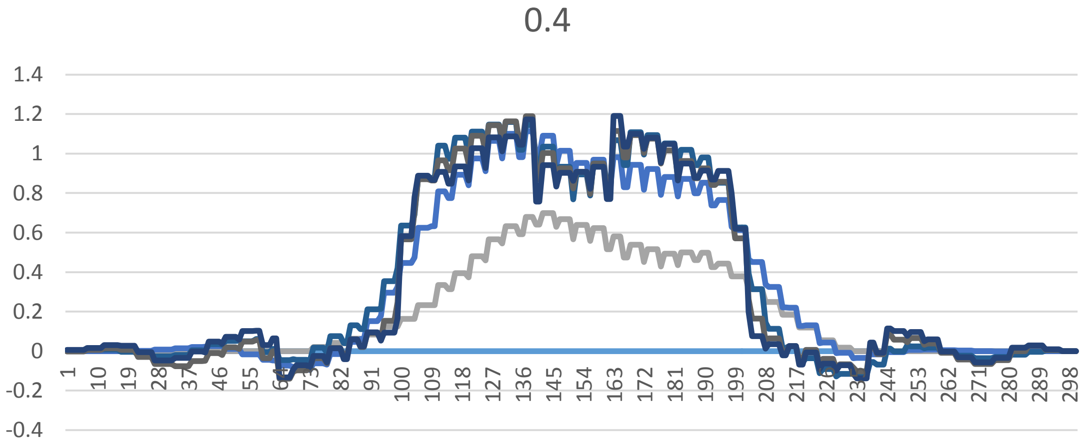


コードの粗さ



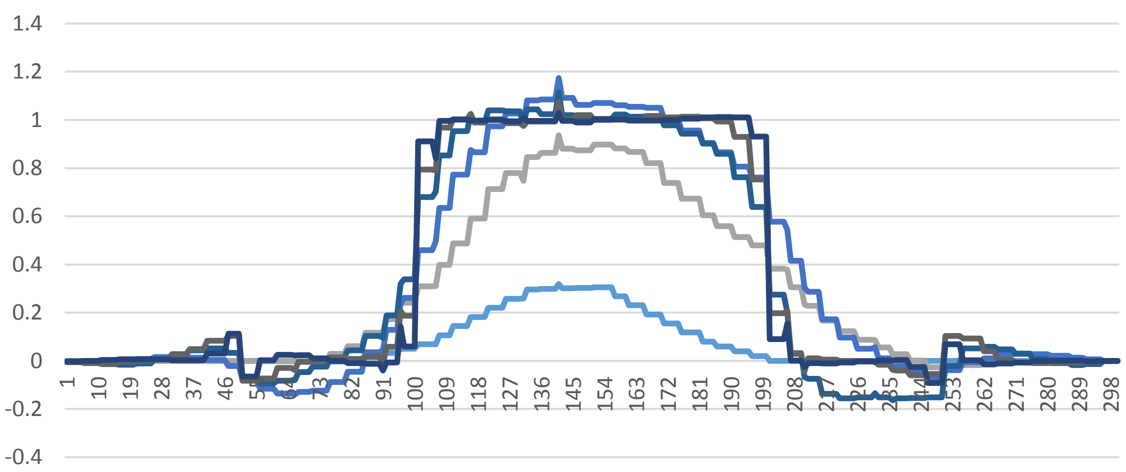




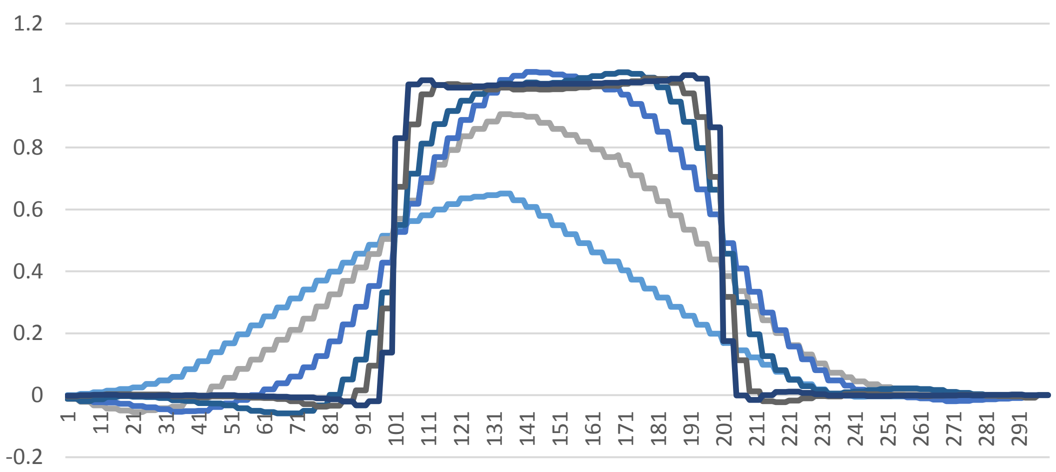


コード化

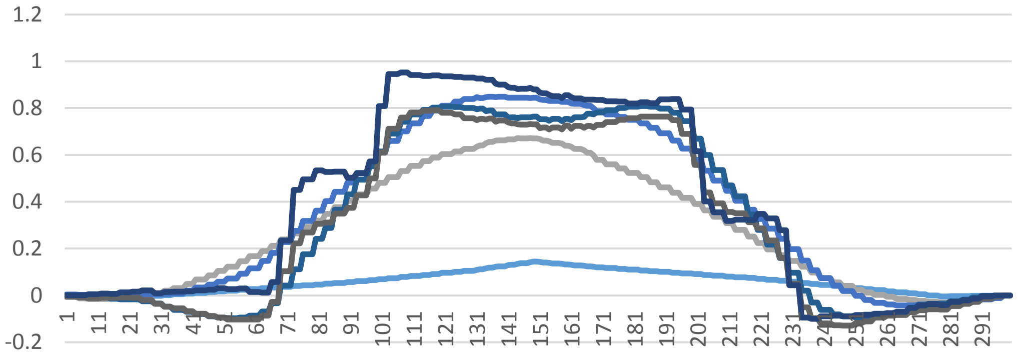
0.5



1.0



1.3



1.5