13/11/24 Update

13 November 2024 15:04

Some derivation for finding orthogonal "points" maintaining K values:

Say some point: $P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{bmatrix}$, $P \in \mathbb{R}$

$$V(K) = K_1 \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \\ \vdots \\ s_{1n} \end{bmatrix} + K_2 \begin{bmatrix} s_{21} \\ s_{22} \\ s_{23} \\ \vdots \\ s_{2n} \end{bmatrix} + \cdots K_r \begin{bmatrix} s_{r1} \\ s_{r2} \\ s_{r3} \\ \vdots \\ s_{rn} \end{bmatrix}, \quad V \in \mathbb{R}.$$

(for V's not starting at 0,0, transpose first by P).

minimise objective function:

$$f(K) = (P - V(K))^2 = \left(p_1 - (K_1 s_{11} + K_2 s_{21} + \cdots K_r s_{r1})\right)^2 + \left(p_2 - (K_1 s_{12} + K_2 s_{22} \cdots + K_r s_{r2})\right)^2$$
$$\vdots$$
$$+ \left(p_n - (K_1 s_{1n} + K_2 s_{2n} + \cdots K_r s_{rn})\right)^2$$

take partial derivative:

$$\frac{\partial f}{\partial K} = \left[\frac{\partial f}{\partial K_1} \; \frac{\partial f}{\partial K_2} \cdots \frac{\partial f}{\partial K_r}\right]$$

$$0 = \frac{\partial f}{\partial K_1} = 2(-s_{11})\left(p_1 - (K_1 s_{11} + K_2 s_{21} \cdots)\right) + 2(-s_{12})\left(p_2 - (K_1 s_{12} + K_2 s_{22} \cdots)\right)$$

etc.

r unknowns, r equations

formulates a set of r simultaneous equations in form $Ax = b$, where

$$A = \begin{bmatrix} \sum_{i=1}^{r} s_{1i}^2 & \sum_{i=1}^{r} s_{1i} s_{2i} & \sum_{i=1}^{r} s_{1i} s_{3i} & \cdots & \sum_{i=1}^{r} s_{1i} s_{ri} \\ & \sum s_{2i}^2 & \sum s_{2i} s_{3i} & \cdots & \\ & & \ddots & \vdots & \\ & & & & \sum s_{ri}^2 \end{bmatrix}$$

(Note: symmetric)

$$b = \begin{bmatrix} \sum_{i=1}^{r} p_i s_{1i} \\ \sum^{r} p_i s_{2i} \end{bmatrix} \qquad x = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \end{bmatrix}$$

$$b = \begin{bmatrix} \sum_{i=1} p_i s_{1i} \\ \sum_{i=1}^{r} p_i s_{2i} \\ \vdots \\ \sum_{i=1}^{r} p_i s_{ri} \end{bmatrix} \qquad x = \begin{bmatrix} \vdots \\ K_2 \\ \vdots \\ K_r \end{bmatrix}$$

Using pythons np.linalg.solve function (could exist a faster way for symmetric matrices?) solves for K

Seems to work well in the context of the algorithm discussed last week.
(solves for 2 1,000 x 1,000 vectors in 100,000 Dimensional space reasonably quickly.) → only performs 3-4 iterations. Can't tell if this is good.
Time complexity: $O(r^3)$ → in theory, doesn't really scale with n?

Iteration count seems to go DOWN with scaling n and r, which is strange but could be a property of the system. Made sure to normalise RMS checking metric.

Significantly faster than algorithm from last time, often requiring 1000+ x less iterations to converge.

```
No size array provided, setting to default.
(array([   1. , -166.3,  312.6,  143.1,  -15.5, -163.4,  -17.9, -237.1,
         12.3,   95.8]), array([  17. , -279.1, -274.4, -166.3,   23.8, -224.1,  109.9,  -29.4,
        -84.6,  -21.8]), 12583)
Made it here!
(array([   1.0155751 , -166.29187912,  312.64925603,  143.12072228,
        -15.54382458, -163.378709  ,  -17.90770774, -237.06126441,
         12.26956715,   95.77559908]), array([  16.98606249, -279.07444188, -274.35807456, -166.27144871,
         23.75668287, -224.06748873,  109.93861973,  -29.41294294,
        -84.61245244,  -21.78368758]), 7)
PS C:\Users\oweng\Desktop\University Work\Y4\Project\testbench>
```

method 1

MUCH higher

Good accuracy.

new method.

Much lower.

Potential problems / questions:

1. Bottlenecked test bench:
   Funnily enough, the current bottleneck in computation is the generation method. Can be optimised, probably, but generating the aformentioned 1000 D vectors in 100,000 D space requires the random generation of 100,000,000 random numbers, each. Could need some parallelism.

Could need some parallelism.

2. Accuracy.

Method 1 is too slow to reasonably run on higher spaces. Hard to verify how accurate the algorithm is at higher dimensional spaces, especially since the iteration counter is so small.

3. Is solving the set of simultaneous equations really the best way? Can't tell.

4. Would a similar objective function exist directly between two vectors?

i.e. minimise:

$$f(k_1, k_2) = (k_1 V_1 - k_2 V_2)^2.$$

+ more but cannot remember off the top of my head.