Глава III. Самоорганизующиеся многообразия малой размерности

Введение

В первом параграфе вводится понятие главных кривых. Также описывается идея итерационного алгоритма построения главной кривой.

Во втором параграфе вводится понятие самоорганизующихся главных кривых на основе самоорганизующихся карт Кохонена [33, 74].

Третий параграф посвящен самоорганизующимся кривым – одномерным нелинейным многообразиям. Приводится алгоритм их построения. В четвертом же описываются расширения самоорганизующихся кривых – соответствующие самоорганизующиеся карты. Приводятся две основные методики представления карт – с квадратной и гексагональной сетками.

В отличие от линейных и квазилинейных моделей, построение самоорганизующихся кривых и карт связано с проблемой попадания в область локального минимума, возможные решения которой приводятся в пятом параграфе: метод "отжига" и многосеточные методы.

Еще одна проблема, с которой приходится сталкиваться при построении самоорганизующихся многообразий на основе ломаных, заключается в кусочной гладкости полученных многообразий. Поэтому они нуждаются в сглаживании, варианты которого описываются в шестом параграфе.

Последний, седьмой, параграф посвящен механической интерпретации принципа построения самоорганизующихся кривых и карт.

III.1. Определение главных кривых

Главная кривая P_X (Principal Curve – PC) [73] есть обобщение главной компоненты на большой класс нелинейных вектор-функций $f(\lambda)$, $\lambda \in R^1$, предоставляемых для представления данных. Проекция $\lambda(v)$ точки данных v на кривую опять же (как и в случае главных компонент) определяется таким значением λ, для которого $f(\lambda)$ ближайшая к v:

$$\lambda(v) = \arg\min_{\lambda}(v - f(\lambda))^2 .$$

В общем, каждая точка (сегмент в дискретном случае) кривой является проекцией подмножества точек данных, которые называются ее проекторами. В линейном случае проекция совпадает с центром масс проекторов. В этом смысле можно сказать, что главная кривая проходит через середину своих точек данных. Это определение может быть распространено и на нелинейный случай. Следовательно, главная кривая определяется как проходящая через центр масс точек, проецируемых на нее (рис 3.1).

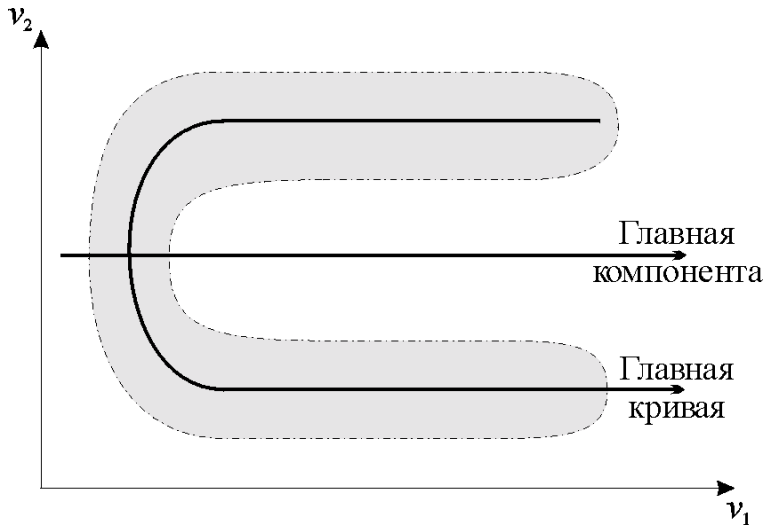

Рис. 3.1.

Также можно сказать, что главной кривой требуется быть центром ее проекторов. Это может быть сформулировано в терминах вариационного принципа, т.е. мы определим среднеквадратическое отклонение между точками данных и их проекциями (ошибка восстановления)

$$E = \int_X (v - f(\lambda))^2 P(v) dv ,$$

и потребуем, чтобы ее изменение относительно $f(\lambda)$ было равно нолю

$$\frac{\partial}{\partial f} E = 0 . \qquad (3.1)$$

Заметим, что мы не должны требовать, чтобы ошибка восстановления была минимальна, как в линейном случае. Вместо этого более слабое свойство стационарности возникает из уравнения (3.1). Причину этого хорошо видно из патологического случая кривой, которая проходит через каждую точку шумящего множества данных X так, что ошибка восстановления равна нолю. Впрочем эта кривая не удовлетворяет назначению главных кривых, поскольку она не исключает шум. Для того, чтобы предотвращать такие патологии, требуется добавочное условие для завершения определения главной кривой. Обычно это условие гладкости, которое ограничивает кривизну PC. Другое условие – это требование, чтобы отображение $v \rightarrow \lambda$ было топографическим лучшим возможным способом. Это точное формулировка тех критериев, которые делают проблему сильно нетривиальной.

Алгоритм Hastie-Stuetzle

Hastie и Stuetzle [68] (здесь и далее HS) описали алгоритм для построения главных кривых, который работает итерационно, начиная с главной компоненты множества данных. На каждой итерации получается новая оценка PC, которая получается вычислением центров масс точек данных относительно текущей оценки PC.

Определение 1: Путь X обозначает случайный вектор в пространстве $\Re^d$ и имеется n векторов данных. Главная кривая $f \subset \Re^d$ – это гладкая (C^∞) кривая в

$\Re^d$, параметризованная параметром $t \in A \subset R^1$, которая проходит через середину d-мерных данных, описываемых X,

$$f(t) = \begin{bmatrix} f_1(t) \\ \vdots \\ f_d(t) \end{bmatrix} = E\left\{X \middle| t_f(X) = t\right\}, \tag{3.2}$$

где:

$t_f(x) = \sup_t \left\{t : \|x - f(t)\| = \inf_\mu \|x - f(\mu)\|\right\}$ – функция проектирования на кривую.

Эта функция определяет такое значение t, для которого дистанция между x и $f(t)$ минимальна.

Таким образом, $f(t)$ есть условное математическое ожидание X среди тех X, к которым $f(t)$ ближе среди любых других точек f (то есть среднее среди всех точек данных, которые проецируются в нее).

Среднеквадратичная ошибка (Mean Square Error – MSE) между точками данных и их ближайшими проекциями на главную кривую на j-ой итерации получается как следующее:

$$MSE^{(j)} = E\left\{\left\|X - f^{(j)}(t_{f^{(j)}}(X))\right\|^2\right\}.$$

Алгоритм построения главной кривой – итерационный, включающий шаг вычисления математического ожидания (3.2) и шаг проектирования на каждой итерации.

Толерантность (TOL) на j-ой итерации определяется как относительное изменение MSE следующим образом:

$$TOL^{(j)} = \frac{\left|MSE^{(j)} - MSE^{(j-1)}\right|}{\left|MSE^{(j-1)}\right|}.$$

Построение прерывается, когда TOL становится ниже заданного порога.

Главная кривая может быть замкнутой или открытой. Для замкнутой кривой во время вычислений добавляется сегмент, включающий конечные точки.

Таким образом, главные кривые являются гладкими self-consistent кривыми, которые проходят через "середину" распределения и обеспечивают хорошее одномерное представление данных.

III.2. Идея самоорганизующихся кривых

Квазилинейные факторы хорошо работают для "средне нелинейных" задач, в которых вклад линейной части относительно велик (порядка 50% или более). Для существенно нелинейных задач естественно использовать вместо линейных главных компонент какие-либо приближения главных кривых.

Предлагается вместо линейных многообразий малой размерности использовать соответствующие самоорганизующиеся карты (Self-Organizing map – SOM) или, другими словами, самоорганизующиеся кривые – SOC (Self-Organizing Curve). Используемый нами алгоритм несколько отличается от

метода карт Кохонена ([74], доступное изложение см. также в [33]) более прозрачной физической интерпретацией и явной формой вариационного принципа.

Предлагается использовать принципиально нелинейный способ. В R^n размещается либо одномерная ломаная, либо двумерная серка, расположение узлов которой удовлетворяет определенным требованиям. И для ломаной, и для сетки этими требованиями являются: а) близость узлов ломаной/сетки к данным; б) не слишком сильная «растянутость» ломаной/сетки; в) не слишком сильная изогнутость ломаной/сетки.

Новшества метода заключается в использовании дополнительных "модулей упругости", а также в использовании кусочно-линейного непрерывного проектора данных на полученную ломаную с последующим сглаживанием по формуле Карлемана.

III.3. SOC

Пусть SOC определяется набором точек (ядер) $Y=\{y_j\}$ ($j=1..m$), последовательно расположенных на кривой (в первом приближении пусть SOC – просто ломаная Y) и требуется отобразить на ней набор точек данных $X=\{x_i\}$. Введем преобразование $П$, которое каждому вектору $x\in X$ сопоставляет ближайшую к нему точку из Y:

$$x \xrightarrow{П} y_j, \left\|y_j - x\right\|^2 \to \min, \tag{3.3}$$

каждому ядру y_j сопоставляется его таксон

$$K_j = \left\{ x \in X \middle| x \xrightarrow{П} y_j \right\}. \tag{3.4}$$

Метод построения SOC напоминает метод динамических ядер, за исключением добавления дополнительных ограничений на связность и нелинейность. Минимизируемая величина строится из следующих слагаемых:

1) мера приближения данных:

$$D_1 = \sum_j \sum_{x\in K_j} \left\|x - y_j\right\|^2; \tag{3.5}$$

Эта мера служит для того, чтобы SOC наилучшим (в некотором определенном смысле) образом приближала исходные данные. Здесь K_j – множество точек из X, для которых узел ломаной y_j является ближайшим.

2) мера связности (близкие точки на кривой должны переходить в близкие в пространстве данных):

$$D_2 = \sum_j \left\|y_j - y_{j+1}\right\|^2; \tag{3.6}$$

Необходимость этой меры вызвана тем, чтобы ломаная не выходила за границы данных – чтобы свободные узлы притягивались к ее "центру". Также эта мера, как будет более подробно описано ниже, используется в различных алгоритмах построения.

3) мера нелинейности:

$$D_3 = \sum_j \left\| 2y_j - y_{j-1} - y_{j+1} \right\|^2 . \qquad (3.7)$$

Эта мера также является мерой равномерности – чтобы расстояния между соседними узлами были примерно одинаковыми.

Таким образом, для построения SOC требуется минимизировать функционал:

$$D = \frac{D_1}{|X|} + \lambda \frac{D_2}{m} + \mu \frac{D_3}{m} \to \min , \qquad (3.8)$$

где λ, μ – параметры связности и нелинейности – "модули упругости" (деление на число точек $|X|$ и число ядер m означает нормировку "на одно слагаемое" и позволяет для выборок разной мощности использовать одинаковые способы изменения λ и μ.)

При фиксированном разбиении множества данных на таксоны SOC строится однозначно – решается простая линейная задача. При фиксированном положении ядер таксоны также легко строятся по формулам (3.3, 3.4). Расщепляя задачу на последовательный поиск ядер – таксонов – ядер … , получаем итерационный алгоритм, сходимость которого гарантируется тем, что на каждом его шаге уменьшается критерий D (3.8).

Алгоритм построения SOC

Пусть задана таблица (a_{ij}) с пробелами (некоторые a_{ij}=@) и набор ядер $Y=\{y^k\}$ (k=0..m-1). Формулы (3.3) – (3.8) перепишутся следующим образом (a_i – i-ая строка таблицы (a_{ij})):

$$a_i \overset{\Pi}{\to} y^k , \sum_i (a_{ij} - y_j^k)^2 \to \min , \qquad (3.9)$$

$$K_k = \left\{ a_i \in (a_{ij}) \middle| a_i \overset{\Pi}{\to} y^k \right\} . \qquad (3.10)$$

$$D_1 = \sum_{k=0}^{m-1} \sum_{i \in K_k} \sum_{\substack{j \\ a_{ij} \neq @}} (a_{ij} - y_j^k)^2 , \qquad (3.11)$$

$$D_2 = \sum_{k=0}^{m-2} \sum_j (y_j^k - y_j^{k+1})^2 , \qquad (3.12)$$

$$D_3 = \sum_{k=1}^{m-2} \sum_j (2y_j^k - y_j^{k-1} - y_j^{k+1})^2 . \qquad (3.13)$$

Требуется решить следующую задачу:

$$D = \frac{D_1}{n} + \lambda \frac{D_2}{m} + \mu \frac{D_3}{m} \to \min , \qquad (3.14)$$

где n – число строк матрицы (a_{ij}).

Значения y_j^k, доставляющие минимум форме (3.14) при заданном разбиении K_k, определяются из системы равенств $\partial D / \partial y_j^k = 0$:

$$\frac{\partial D}{\partial y_j^k} = \frac{1}{n} \cdot \frac{\partial D_1}{\partial y_j^k} + \frac{\lambda}{m} \cdot \frac{\partial D_2}{\partial y_j^k} + \frac{\mu}{m} \cdot \frac{\partial D_3}{\partial y_j^k}, \tag{3.15}$$

$$\frac{\partial D_1}{\partial y_j^k} = -2 \sum_{\substack{i \in K_k \\ a_{ij} \neq @}} (a_{ij} - y_j^k), \tag{3.16}$$

$$\frac{\partial D_2}{\partial y_j^k} = -2(y_j^{k-1} - y_j^k) + 2(y_j^k - y_j^{k+1}), \tag{3.17}$$

$$\frac{\partial D_3}{\partial y_j^k} = -2(2y_j^{k-1} - y_j^{k-2} - y_j^k) + \\ + 4(2y_j^k - y_j^{k-1} - y_j^{k+1}) - 2(2y_j^{k+1} - y_j^k - y_j^{k+2}). \tag{3.18}$$

Подставляя (3.16 – 3.18) в (3.15), для каждого j имеем систему из k линейных уравнений относительно y_j^k:

$$A_{kj}^{-2} y_j^{k-2} + A_{kj}^{-1} y_j^{k-1} + A_{kj}^0 y_j^k + A_{kj}^1 y_j^{k+1} + A_{kj}^2 y_j^{k+2} = B_j^k, \; k=1..m,$$

где A_{kj}^l ($l = -2 \ldots 2$) – пятидиагональная матрица, коэффициенты которой определяются из уравнений (3.15 – 3.18).

Решение может быть получено каким-нибудь численным методом для диагональной матрицы [51].

III.4. SOM

Пусть SOM определяется набором точек (ядер) $Y=\{y_{ij}\}$ ($i,j=1..m$ – будем считать, что сетка квадратная, но для прямоугольной все описанные формулы аналогичны), последовательно расположенных на квадратной сетке и требуется отобразить на ней набор точек данных $X=\{x_i\}$. Введем преобразование $П$, которое каждому вектору $x \in X$ сопоставляет ближайшую к нему точку из Y:

$$x \xrightarrow{П} y_{ij}, \; \|y_{ij} - x\|^2 \to \min, \tag{3.19}$$

каждому ядру y_{ij} сопоставляется его таксон

$$K_{ij} = \left\{ x \in X \middle| x \xrightarrow{П} y_{ij} \right\}. \tag{3.20}$$

Метод построения SOM также, как и SOC, напоминает метод динамических ядер, за исключением добавления дополнительных ограничений на связность и нелинейность.

А так как может быть построено две разновидности SOM – квадратная и гексагональная, то остановимся подробно на каждой из них.

Квадратная SOM

Как и в случае с SOC, минимизируемая величина строится из следующих слагаемых:

1) мера приближения данных:

$$D_1 = \sum_{ij} \sum_{x \in K_{ij}} \left\| x - y_{ij} \right\|^2 ; \tag{3.21}$$

2) мера связности (близкие точки на карте должны переходить в близкие в пространстве данных):

$$D_2 = \sum_{ij} \left[\left\| y_{ij} - y_{i+1,j} \right\|^2 + \left\| y_{ij} - y_{i,j+1} \right\|^2 \right]; \tag{3.22}$$

3) мера нелинейности (или равномерности):

$$D_3 = \sum_{j} \left\| 2y_{ij} - y_{i-1,j} - y_{i+1,j} \right\|^2 + \left\| 2y_{ij} - y_{i,j-1} - y_{i,j+1} \right\|^2 . \tag{3.23}$$

Таким образом, для построения SOC требуется минимизировать функционал:

$$D = \frac{D_1}{|X|} + \lambda \frac{D_2}{m^2} + \mu \frac{D_3}{m^2} \to \min , \tag{3.24}$$

где λ, μ – параметры связности и нелинейности – "модули упругости" (деление на число точек $|X|$ и число ядер m означает нормировку "на одно слагаемое" и позволяет для выборок разной мощности использовать одинаковые способы изменения λ и μ.)

Как и для SOC, процедура построения SOM также является итерационной – то есть задача расщепляется на последовательны поиск ядер – таксонов – ядер … и т.д. И эта процедура также гарантировано сходится.

Гексагональная SOM

С учетом того, что каждый узел SOM в данном случае имеет не 4, а 6 соседей, то минимизируемые функционалы перепишутся в следующем виде:

1) мера приближения данных:

$$D_1 = \sum_{ij} \sum_{x \in K_{ij}} \left\| x - y_{ij} \right\|^2 ; \tag{3.25}$$

2) мера связности (близкие точки на карте должны переходить в близкие в пространстве данных):

$$\begin{aligned} D_2 = & \sum_{ij} \left\| y_{ij} - y_{i,j+1} \right\|^2 + \\ & + \sum_{\substack{ij \\ i-\text{четное}}} \left[\left\| y_{ij} - y_{i+1,j} \right\|^2 + \left\| y_{ij} - y_{i+1,j-1} \right\|^2 \right] + \sum_{\substack{ij \\ i-\text{нечетное}}} \left[\left\| y_{ij} - y_{i+1,j+1} \right\|^2 + \left\| y_{ij} - y_{i+1,j} \right\|^2 \right] \end{aligned} \tag{3.26}$$

3) мера нелинейности (или равномерности):

$$\begin{aligned} D_3 = & \sum_{ij} \left\| 2y_{ij} - y_{i,j-1} - y_{i,j+1} \right\|^2 + \\ & + \sum_{\substack{ij \\ i-\text{четное}}} \left[\left\| 2y_{ij} - y_{i-1,j-1} - y_{i+1,j} \right\|^2 + \left\| 2y_{ij} - y_{i-1,j} - y_{i+1,j-1} \right\|^2 \right] + \\ & + \sum_{\substack{ij \\ i-\text{нечетное}}} \left[\left\| 2y_{ij} - y_{i-1,j} - y_{i+1,j+1} \right\|^2 + \left\| 2y_{ij} - y_{i-1,j+1} - y_{i+1,j} \right\|^2 \right] \end{aligned} \tag{3.27}$$

Алгоритм построения квадратной SOM

Пусть задана таблица (a_{ij}) с пробелами (некоторые a_{ij}=@) и набор ядер $Y=\{y^{kl}\}$ (k,l=0..m–1). Формулы (3.19 – 3.24) перепишутся следующим образом (a_i – i-ая строка таблицы (a_{ij})):

$$a_i \overset{\Pi}{\to} y^{kl},\ \sum_i (a_{ij} - y_j^{kl})^2 \to \min, \tag{3.28}$$

$$K_{kl} = \left\{ a_i \in (a_{ij}) \middle| a_i \overset{\Pi}{\to} y^{kl} \right\}. \tag{3.29}$$

$$D_{1j} = \sum_{k,l=0}^{m-1} \sum_{i \in K_{kl}} \sum_{\substack{j \\ a_{ij} \neq @}} (a_{ij} - y_j^{kl})^2, \tag{3.30}$$

$$D_{2j} = \sum_{k,l=0}^{m-2} \sum_j \left[(y_j^{kl} - y_j^{k+1,l})^2 + (y_j^{kl} - y_j^{k,l+1})^2 \right], \tag{3.31}$$

$$D_{3j} = \sum_{k,l=1}^{m-2} \sum_j \left[(2y_j^{kl} - y_j^{k-1,l} - y_j^{k+1,l})^2 + (2y_j^{kl} - y_j^{k,l-1} - y_j^{k,l+1})^2 \right]. \tag{3.32}$$

Требуется решить следующую задачу:

$$D_j = \frac{D_{1j}}{n} + \lambda \frac{D_{2j}}{m^2} + \mu \frac{D_{3j}}{m^2} \to \min, \tag{3.33}$$

где n – число строк матрицы (a_{ij}).

Значения y_j^{kl}, доставляющие минимум форме (3.33) при заданном разбиении K_{kl}, определяются из системы равенств $\partial D_j / \partial y_j^{kl} = 0$:

$$\frac{\partial D_j}{\partial y_j^{kl}} = \frac{1}{n} \cdot \frac{\partial D_{1j}}{\partial y_j^{kl}} + \frac{\lambda}{m^2} \cdot \frac{\partial D_{2j}}{\partial y_j^{kl}} + \frac{\mu}{m^2} \cdot \frac{\partial D_{3j}}{\partial y_j^{kl}}, \tag{3.34}$$

$$\frac{\partial D_{1j}}{\partial y_j^{kl}} = -2 \sum_{\substack{i \in K_{kl} \\ a_{ij} \neq @}} (a_{ij} - y_j^{kl}), \tag{3.35}$$

$$\frac{\partial D_{2j}}{\partial y_j^{kl}} = -2(y_j^{k-1,l} - y_j^{kl}) + 2(y_j^{kl} - y_j^{k+1,l}) - 2(y_j^{k,l-1} - y_j^{kl}) + 2(y_j^{kl} - y_j^{k,l+1}), \tag{3.36}$$

$$\begin{aligned} \frac{\partial D_{3j}}{\partial y_j^{kl}} = & -2(2y_j^{k-1,l} - y_j^{k-2,l} - y_j^{kl}) + 4(2y_j^{kl} - y_j^{k-1,l} - y_j^{k+1,l}) - \\ & - 2(2y_j^{k+1,l} - y_j^{kl} - y_j^{k+2,l}) - 2(2y_j^{k,l-1} - y_j^{k,l-2} - y_j^{kl}) + \\ & + 4(2y_j^{kl} - y_j^{k,l-1} - y_j^{k,l+1}) - 2(2y_j^{k,l+1} - y_j^{kl} - y_j^{k,l+2}). \end{aligned} \tag{3.37}$$

Для каждого j имеем систему из m^2 линейных уравнений относительно y_j^{kl}:

$$\begin{aligned} & A_j^{k-2,l} y_j^{k-2,l} + A_j^{k-1,l} y_j^{k-1,l} + A_j^{kl} y_j^{kl} + A_j^{k+1,l} y_j^{k+1,l} + A_j^{k+2,l} y_j^{k+2,l} + \\ & + A_j^{k,l-2} y_j^{k,l-2} + A_j^{k,l-1} y_j^{k,l-1} + A_j^{k,l+1} y_j^{k,l+1} + A_j^{k,l+2} y_j^{k,l+2} = B_j^{kl}, \end{aligned}$$

где A_j^{kl} ($k,l = 0 \ldots m$) – коэффициенты при соответствующих y_j^{kl}, которые определяются из уравнений (3.34 – 3.37).

Можно заметить, что матрица имеет диагональный вид, поэтому она с достаточной точностью может быть решена каким-нибудь численным методом, например, оптимизированным для диагональных матриц методом Гаусса, либо методом Гаусса-Зейделя [51].

Алгоритм построения гексагональной SOM

При построении гексагональной SOM претерпевают изменения только функционалы D_2 и D_3:

$$D_{2j} = \sum_{kl} (y_j^{kl} - y_j^{k,l+1})^2 +$$

$$+ \sum_{\substack{kl \\ k-\text{четное}}} \left[(y_j^{kl} - y_j^{k+1,l})^2 + (y_j^{kl} - y_j^{k+1,l-1})^2 \right] + \sum_{\substack{kl \\ k-\text{нечетное}}} \left[(y_j^{kl} - y_j^{k+1,l+1})^2 + (y_j^{kl} - y_j^{k+1,l})^2 \right],$$

$$D_{3j} = \sum_{kl} (2y_j^{kl} - y_j^{k,l-1} - y_j^{k,l+1})^2 +$$

$$+ \sum_{\substack{kl \\ k-\text{четное}}} \left[(2y_j^{kl} - y_j^{k-1,l-1} - y_j^{k+1,l})^2 + (2y_j^{kl} - y_j^{k-1,l} - y_j^{k-1,l-1})^2 \right] +$$

$$+ \sum_{\substack{kl \\ k-\text{нечетное}}} \left[(2y_j^{kl} - y_j^{k-1,l} - y_j^{k+1,l+1})^2 + (2y_j^{kl} - y_j^{k-1,l+1} - y_j^{k+1,l})^2 \right]$$

$$\frac{\partial D_{2j}}{\partial y_j^{kl}} = -2(y_j^{k,l-1} - y_j^{kl}) + 2(y_j^{kl} - y_j^{k,l+1}) +$$

$$+ 2\left[(y_j^{kl} - y_j^{k+1,l}) - (y_j^{k-1,l-1} - y_j^{kl}) + (y_j^{kl} - y_j^{k+1,l-1}) - (y_j^{k-1,l} - y_j^{kl}) \right]_{k-\text{четное}} +$$

$$+ 2\left[(y_j^{kl} - y_j^{k+1,l+1}) - (y_j^{k-1,l} - y_j^{kl}) + (y_j^{kl} - y_j^{k+1,l}) - (y_j^{k-1,l+1} - y_j^{kl}) \right]_{k-\text{нечетное}},$$

$$\frac{\partial D_3}{\partial y_j^{kl}} = -2(2y_j^{k,l-1} - y_j^{k,l-2} - y_j^{kl}) + 4(2y_j^{kl} - y_j^{k,l-1} - y_j^{k,l+1}) - 2(2y_j^{k,l+1} - y_j^{kl} - y_j^{k,l+2}) +$$

$$+ 2\left[2(2y_j^{kl} - y_j^{k-1,l-1} - y_j^{k+1,l}) - (2y_j^{k-1,l-1} - y_j^{k-2,l-1} - y_j^{kl}) - (2y_j^{k+1,l} - y_j^{kl} - y_j^{k+2,l+1}) \right]_{k-\text{четное}} +$$

$$+ 2\left[2(2y_j^{kl} - y_j^{k-1,l} - y_j^{k+1,l-1}) - (2y_j^{k-1,l} - y_j^{k-2,l+1} - y_j^{kl}) - (2y_j^{k+1,l-1} - y_j^{kl} - y_j^{k+2,l-1}) \right]_{k-\text{четное}} +$$

$$+ 2\left[2(2y_j^{kl} - y_j^{k-1,l} - y_j^{k+1,l+1}) - (2y_j^{k-1,l} - y_j^{k-2,l-1} - y_j^{kl}) - (2y_j^{k+1,l+1} - y_j^{kl} - y_j^{k+2,l+1}) \right]_{k-\text{нечетное}} +$$

$$+ 2\left[2(2y_j^{kl} - y_j^{k-1,l+1} - y_j^{k+1,l}) - (2y_j^{k-1,l+1} - y_j^{k-2,l+1} - y_j^{kl}) - (2y_j^{k+1,l} - y_j^{kl} - y_j^{k+2,l-1}) \right]_{k-\text{нечетное}}.$$

Подставляя полученные частные производные в общий функционал и группируя коэффициенты при одинаковых y_j^{kl}, получаем систему из m^2 линейных уравнений относительно y_j^{kl}. Аналогично предыдущему пункту, матрица имеет диагональный вид, поэтому она с достаточной точностью решается численным методом Гаусса, либо методом Гаусса-Зейделя.

III.5. ПРОБЛЕМА ЛОКАЛЬНОГО МИНИМУМА

В отличии от линейного и квазилинейного случаев данная задача минимизации функционала (3.8) не является выпуклой, и поэтому возникают трудности, связанные с попаданием в области локального минимума. Это может привести к неудовлетворительному решению задачи.

Хотя для решения данной проблемы существует множество методов, мы же остановимся на многосеточном методе и методе "отжига".

Метод отжига

Так называемый метод "отжига" состоит в том, что к исходной минимизируемой функции произвольным образом добавляется некоторая функция, в результате чего можно добиться сглаживания локальных минимумов, благодаря чему происходит выход процесса минимизации из локального минимума. Затем добавочная функция устремляется к нулю.

Для нашей задачи предполагается использовать следующий метод "отжига":

За счет увеличений в (3.8) коэффициентов λ, μ система ставится в очень жесткие рамки. Далее они постепенно ослабляются (происходит уменьшение соответствующих коэффициентов). В частности, при больших μ получим отрезки, близкие к первой главной компоненте.

Здесь существует два способа:

1) сначала система стягивается в точку (λ – очень велико), а затем за счет ослабления λ растягивается до нужного уровня;

2) сначала система делается очень большой (λ – очень мало), а затем за счет увеличения λ сжимается до нужного уровня.

Аналогичная процедура может проводиться и с коэффициентом μ.

Многосеточный метод

Опишем обычные многосеточные алгоритмы в абстрактной алгебраической форме [11].

Предположим, что есть последовательность пространств ограниченной размерности

$$M_0, M_1,\ldots,M_k$$

с внутренними произведениями, описываемыми как $(\cdot,\cdot)_i$ для каждого M_i. Предположим также, что имеются операторы "интерполяции"

$$I_i: M_i \to M_{i+1},\ i=0,\ldots,k-1;$$

операторы "сужения"

$$R_i: M_i \to M_{i-1},\ i=1,\ldots,k;$$

и операторы обращения

$$L_i: M_i \to M_i,\ i=0,\ldots,k.$$

Основная цель этих алгоритмов – решить следующую задачу в M_k: дано $f_k \in M_k$, найти $v_k \in M_k$ такое, что

$$L_k v_k = f_k. \qquad (3.38)$$

Особенность многосеточных алгоритмов состоит в необходимости решения вспомогательных задач на нижних уровнях. Поэтому, сформулируем MG_i-алгоритм для приближенного решения задачи:

$$L_i v_i = f_i, \text{ где } f_i \in M_i \qquad (3.39)$$

для любого уровня $i \in [0,k]$. Начнем с некоторого начального приближения w_0 и получим в результате другое приближение (как ожидается, более близкое к v_i), описываемое как $w_1 = MG_i(w_0, f_i)$.

***MG_i*-алгоритм.** Если $i=0$, то

$$w_1 = MG_0(w_0, f_0) = L_0^{-1} f_0,$$

начальное приближение w_0 больше не имеет значения и MG_0 – закончен. Для $i>0$ алгоритм определен как следующее:

A_1 (пре-сглаживание). Пусть $u_0 = w_0$ и определим u_{m_1} следующим образом

$$u_{l+1} = u_l - \Im_{i,l+1}(L_i u_l - f_i),\ l=0,1,\ldots,m_1-1;$$

A_2 (сужение). Пусть

$$g_{i-1} = R_i(L_i u_{m_1} - f_i);$$

A_3 (крупно-сеточное решение). Пусть $\tilde{w}_0 = 0 \in M_{i-1}$ и повторим MG_{i-1}-алгоритм γ раз:

$$\tilde{w}_s = MG_{i-1}(\tilde{w}_{s-1}, g_{i-1}),\ s=1,\ldots,\gamma;$$

A_4 (коррекция). Пусть

$$y_0 = u_{m_1} - I_{i-1}\tilde{w}_\gamma;$$

A_1 (пост-сглаживание). Определим y_{m_2} следующим образом

$$y_{l+1} = y_l - \Im_{i,l+m_1+1}(L_i y_l - f_i),\ l=0,1,\ldots,m_2-1.$$

Окончательно получим

$$w_1 = MG_i(w_0, f_i) = y_{m_2}$$

как результат MG_i-алгоритма.

$\Im_{i,l}$ ($l=1,\ldots,m_1+m_2$) – некоторый линейный оператор. Его структуру можно посмотреть в [11].

Три шага A_2–A_4 вместе взятые обычно называются крупно-сеточной коррекцией.

Полный многосеточный *FMG*-алгоритм для решения задачи (3.39), включающий верхний уровень (3.38) на заключительном этапе:

***FMG*-алгоритм.**

1. Пусть $\tilde{u}_0 = L_0^{-1} f_0$.
2. Для $i=1,2,\ldots,k$ делается следующее:
   2.1. Установить $\tilde{u}_i = I_{i-1}\tilde{u}_{i-1}$;
   2.2. Повторить MG_i-алгоритм t раз:
   $$\tilde{u}_i := MG_i(\tilde{u}_i, f_i).$$

На k-ом шаге алгоритма получается окончательный результат $\tilde{u}_k$, который является некоторым приближением решения v_k задачи (3.38) для уровня k.

Многосеточные или, другими словами, иерархические методы основаны на том, что в процессе вычисления шаг сетки каким-либо образом изменяется. Например, в каскадном методе изначально берется достаточно большой шаг, а затем он постепенно уменьшается. Методы же V-цикла и W-цикла имеют более сложные законы изменения шага сетки.

В данной задаче построения SOC и SOM предлагается использовать каскадный метод, при котором, как уже написано выше, изначально берется сетка с очень большим шагом. На этой сетке строится интерполирующая функция. Получается начальное и весьма грубое приближение. Далее сетка подвергается постепенному дроблению, то есть размер сетки шаг за шагом уменьшается. При этом происходит постепенное исправление начальной аппроксимирующей функции. Сетку можно дробить не равномерно, а в зависимости от требуемой на разных участках точности – то есть для всей области данных можно использовать сетку с достаточно крупным шагом, а в некоторых интересующих местах сетка дробится до достижения требуемой точности.

В результате для имеющейся задачи построения SOC многосеточный метод принимает следующую форму:

Изначально ломаная состоит из двух точек. После минимизации функционала (3.8) получим главную прямую (аналог описанных ранее линейных моделей). Затем этот отрезок делится на две части – т.е. добавляется новый узел. Опять минимизируется функционал (3.8). Далее каждый отрезок ломаной дробится на две части, минимизируется функционал и т.д. до достижения требуемой точности.

Вариант метода с делением только тех отрезков ломаной, которые превышают заданную для данного шага длину, оказывается более экономичным.

III.6. Сглаживание

Проблема сглаживания

Полученная ломаная $\{y_j\}$ (j=1..m) может быть сглажена различными способами, например, с использованием кубических сплайнов. Здесь, однако, возникают определенные трудности, связанные с нахождением проекций данных на сглаженную кривую, т.к. для этого требуется решать алгебраические уравнения 5-ой степени.

Для сглаживания возможно использовать также сглаживающий фильтр, но он также не дает возможности для нахождения проекций данных на ломаную.

Поэтому возникает необходимость в создании специального проектора данных на SOC или SOM.

Пусть M – самоорганизующееся многообразие (SOC или SOM). Требуется построить отображение P: $\boldsymbol{x}\in X \rightarrow \boldsymbol{r}\in M$, которое будем называть правилом проектирования или *проектором*. Для рассматриваемых задач желательно, чтобы проектор обладал следующими качествами:

а) проектор должен сохранять отношения соседства, то есть желательно, чтобы близким точкам в R^n соответствовали близкие точки на M (по крайней мере, в некоторой окрестности M);

б) проектор, по крайней мере, в некоторой окрестности M должен быть однозначным;

в) проектор должен быть по возможности непрерывным, чтобы плавным изменениям состояния системы в X соответствовали непрерывные изменения положения образа в M;

г) проекция должна не слишком сильно отличаться от ближайшей вершины ломаной или сетки.

Создателем SOM Кохоненом был применен самый очевидный и в некотором смысле естественный вариант проектирования – кусочно-постоянный. При этом каждой точке из X сопоставляется та вершина сетки, которая является ближайшей к этой точке.

Достоинством такого проектирования являются его логическая прозрачность и простота, очевидный его недостаток – разрывность, что не позволяет наиболее полно смоделировать многообразием M данные X.

Для предложенной технологии построения самоорганизующихся многообразий такой кусочно-постоянный способ проектирования малопригоден, поскольку, в отличие от Кохоненовских SOM, каждый из узлов M, вообще говоря, не располагается в центре локального сгущения точек данных. Напротив, M представляет собой более-менее равномерно натянутую на данные ломаную/сетку, и поэтому существенная часть данных может быть расположена в промежутках между узлами. В этой работе предлагается вариант кусочно-линейного непрерывного проектирования многомерных данных как на SOC, так и на SOM.

Рассмотрим для начала вариант кусочно-линейного непрерывного проектора на SOC. Концам ломаной при этом соответствуют значения (–1) и 1 соответственно, а проекции узлов на отрезок [–1,1] определяются узлами равномерной (учитывается только число узлов ломаной) или неравномерной (учитываются также расстояния между узлами ломаной) сетки.

Окончательное сглаживание производится с использованием формул Карлемана. Ломаная $\{y_j\}$ (j=1..m) при этом заменяет главную компоненту классического метода, а процесс сглаживания аналогичен построению квазилинейной модели.

Кусочно-линейная проекция на ломаную

Требуется построить отображение линейных многообразий $a=a_t \in (a_{ij})$ на ломаную, определяемую набором вершин $\{y^k\}$ (k=0…m–1), то есть каждому x сопоставить некоторое t, определяющее его проекцию на ломаную.

Пусть y_k – ближайшая к a вершина ломаной. Тогда возможны два варианта – либо эта вершина является крайней вершиной ломаной, либо она является внутренней. Процедуры проектирования для каждого из этих случаев различаются, поэтому рассмотрим на каждом из них отдельно. Но перед этим введем дополнительные обозначения:

Полученную ближайшую вершину y_k будем обозначать как y_0, а соседние с ней две вершины – y_1 и y_2 соответственно (если y_k – крайняя, то берем только одну соседнюю – y_1). И в дальнейшем координаты всех точек y_0, y_1, y_2 и a будем считать относительно точки y_0, т.е. их координатами будут:

$$y_1 = y_1 - y_0,\ y_2 = y_2 - y_0,\ a = a - y_0,\ y_0 = 0. \qquad (3.40)$$

А теперь вернемся к двум вариантам расположения ближайшей вершины.

1. Пусть y_k – крайняя вершина ломаной, т.е. k=0 или k=m–1 (рис. 3.2).

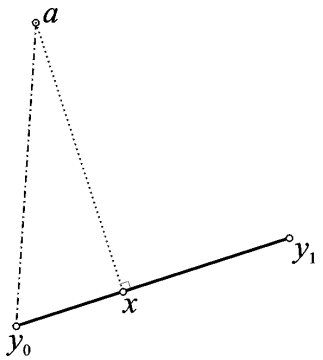

Рис. 3.2.

В этом случае проводится обычная ортогональная проекция точки a на вектор y_1 (см. вторую главу). Т.е. требуется найти такую точку x, лежащую на y_1, что вектор, соединяющий a и x, будет ортогонален y_1.

Так как x лежит на y_1, то ее координата будет равна x=x_1y_1, а условие ортогональности запишется в виде скалярного произведения. Таким образом имеем следующую систему:

$$\begin{cases} x = x_1 y_1, \\ (x - a, y_1)_a = 0. \end{cases}$$

Подставив первую строчку системы во вторую, получим:

$$(x_1 y_1 - a, y_1)_a = 0, \Rightarrow x_1 (y_1, y_1)_a = (a, y_1)_a, \Rightarrow$$

$$x_1 = \frac{(a, y_1)_a}{\|y_1\|_a^2}. \qquad (3.41)$$

Полученное значение x_1 определяет проекцию на одно ребро ломаной, но чтобы получить окончательное значение проекции на ломаную, его надо подвергнуть обработке. Поэтому, для определенности, если k=0, то значение x_1 берется "как есть", а для k=m–1 оно берется с противоположным знаком.

2. Пусть y_k – внутренняя вершина ломаной, т.е. 0<k<m–1 (рис. 3.3).

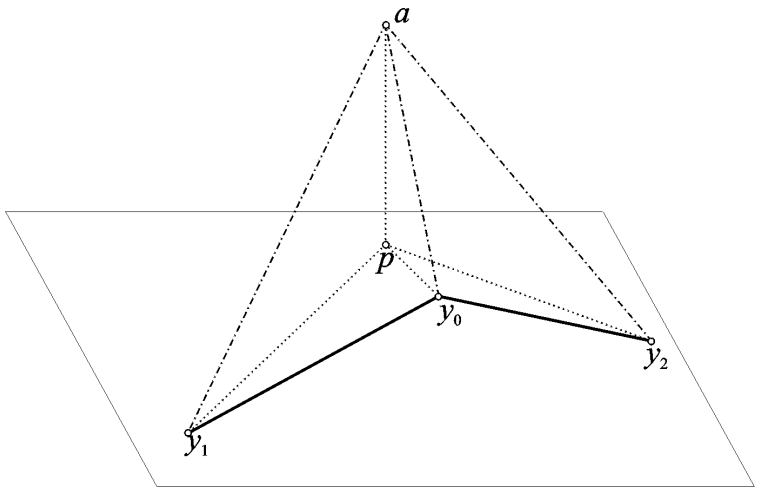

Рис. 3.3.

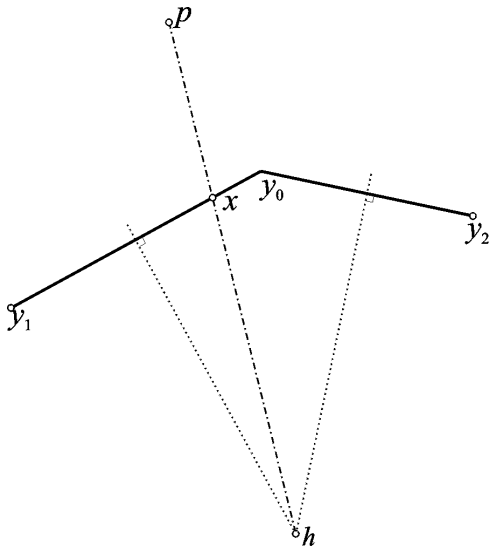

Рис. 3.4.

Из *n*-мерной задача легко сводится к двумерной за счет проецирования точки *a* на плоскость, образованную двумя ближайшими ребрами ломаной (рис. 3.3). При этом легко доказать, что отношения близости между полученной точкой *p* и вершинами ломаной y_0, y_1 и y_2 не изменятся.

Таким образом, первая задача заключается в нахождении такой точки *p*, на плоскости, образованной векторами y_1 и y_2, то вектор, соединяющий *p* и *a*, будет ортогонален обоим векторам y_1 и y_2.

Так как *p* лежит на плоскости, образованной y_1 и y_2, то ее координата будет равна $p=p_1y_1+p_2y_2$, а условие ортогональности запишется в виде скалярного произведения. Таким образом для нахождения координат точки *p* требуется решить следующую систему уравнений:

$$\begin{cases} p = p_1 y_1 + p_2 y_2; \\ (a-p, y_1)_a = 0; \\ (a-p, y_2)_a = 0, \end{cases} \Rightarrow \begin{cases} p_1 (y_1, y_1)_a + p_2 (y_1, y_2)_a = (a, y_1)_a; \\ p_1 (y_1, y_2)_a + p_2 (y_2, y_2)_a = (a, y_2)_a. \end{cases} \quad (3.42)$$

Решение может быть получено любым численным способом. Заметим, что система совместна, если векторы y_1 и y_2 не лежат на одной прямой. В противном случае проводится описанная выше ортогональная проекция на одно из ребер ломаной.

Далее требуется спроецировать полученную точку *p* на ребро ломаной. Для определенности сначала найдем значение проекции на ребро y_1, а затем, по аналогии, на y_2.

Определим проекцию точки *p* на ребро ломаной (рис. 3.4) как точку пересечения ребра ломаной и прямой, соединяющей *p* и точку пересечения серединных перпендикуляров к ребрам ломаной *h*.

Так как искомая точка *h* лежит в плоскости векторов y_1 и y_2, а векторы, соединяющие ее с серединами векторов y_1 и y_2, соответственно ортогональны этим векторам, то точку *h* легко находим из системы уравнений:

$$\begin{cases} h = h_1 y_1 + h_2 y_2, \\ (h - 0.5 y_1, y_1) = 0, \\ (h - 0.5 y_2, y_2) = 0. \end{cases}$$

Подставив первое уравнение системы в остальные два, получим следующую систему линейных уравнений с неизвестными h_1 и h_2:

$$\begin{cases} h_1(y_1, y_1) + h_2(y_1, y_2) = 0.5(y_1, y_1); \\ h_1(y_1, y_2) + h_2(y_2, y_2) = 0.5(y_2, y_2). \end{cases}$$

Она также легко решается численными методами. Будем считать, что векторы y_1 и y_2 не коллинеарные, поэтому система имеет единственное решение.

Координаты проекции x точки p находится из условий, что искомая точка одновременно лежит на прямой, образованной вектором h–p, и на прямой, образованной вектором y_1. Поэтому координаты проекции x на ребро y_1 определяются из системы уравнений:

$$\begin{cases} x - p = \alpha(h - p), \\ x = x_1 y_1. \end{cases}$$

Поставляя первое уравнение во второе, получим:

$$x_1 y_1 = \alpha h_1 y_1 + \alpha h_2 y_2 + p_1 y_1 (1 - \alpha) + p_2 y_2 (1 - \alpha) = 0,$$

и после группировки слагаемых при y_1 и y_2 получим следующее уравнение:

$$y_1(\alpha h_1 + p_1(1 - \alpha) - x_1) + y_2(\alpha h_2 + p_2(1 - \alpha)) = 0,$$

А так как векторы y_1 и y_2 линейно независимы, то последнее равенство выполняется только в том случае, когда коэффициенты при y_1 и y_2 равны нолю:

$$\begin{cases} (h_1 - p_1)\alpha - x_1 = -p_1, \\ (h_2 - p_2)\alpha = -p_2. \end{cases}$$

В результате, проекция на ребро y_1 равна:

$$x_1 = (h_1 - p_1)\frac{p_2}{(p_2 - h_2)} + p_1.$$

Аналогично, проекция на ребро y_2 равна:

$$x_2 = (h_2 - p_2)\frac{p_1}{(p_1 - h_1)} + p_2.$$

Здесь возможна неоднозначность, когда точка пересечения серединных перпендикуляров h и проецируемая точка p совпадают. В этом случае в качестве значения проекции берем середину того ребра ломаной, которое ближе к точке p. Если же расстояния равны (что возможно лишь в том случае, если длины ребер совпадают), то проекция выбирается случайным образом.

Учитывая, что проекция должна лежать на ребре, окончательно берем то из значений x_1 или x_2, которое лежит в отрезке [0, 1]. Для определенности, значение x_1 берется с противоположным знаком, а x_2 без изменений, т.е.:

$$x_1 = \begin{cases} -x_1, \text{ если } x_1 \in [0,1], \\ x_2, \text{ если } x_2 \in [0,1]. \end{cases}$$

В итоге проекция на ломаную в зависимости от типа сетки может определяться двумя способами.

1. Для равномерной сетки.

$$x = -1 + 2\frac{k + x_1}{m - 1}.$$

2. Для неравномерной сетки (с учетом расстояния между узлами).

Пусть l_i – длина i-го ребра ломаной, т.е.:

$$l_i = \|y_{i+1} - y_i\|,\ i=0..m-2.$$

Тогда узлу y_k ломаной будет соответствовать проекция:

$$x(y_k) = -1 + 2\sum_{i=0}^{k-1} l_i \Big/ \sum_{i=0}^{m-2} l_i .$$

И окончательным значением проекции будет (для определенности будем считать, что $l_{-1}=l_0$ и $l_{m-1}=l_{m-2}$):

$$x = -1 + 2 \cdot \begin{cases} \left(\sum_{i=0}^{k-1} l_i + \frac{l_k}{2} x_1 \right) \Big/ \sum_{i=0}^{m-2} l_i , \text{ если } x_1 \ge 0, \\ \left(\sum_{i=0}^{k-1} l_i - \frac{l_{k-1}}{2} x_1 \right) \Big/ \sum_{i=0}^{m-2} l_i , \text{ если } x_1 < 0. \end{cases}$$

Напомним, что значение $x = \pm 1$ соответствуют концам ломаной.

Остается уточнить, что в формулах (3.41) и (3.42) используется обобщенное на случай данных с пробелами скалярное произведение и норма, которые были определены в предыдущей главе. Таким образом, построенный кусочно-линейный проектор данных на ломаную работает как с полными данными, так и с данными, которые содержат пробелы. Действительно, в (3.41) производится проектирование на прямую, а в (3.42) – на плоскость. А это есть ни что иное, как проектирование данных на одномерные и двумерные линейные многообразия, которое и было рассмотрено во второй главе.

В принципе, возможны и другие варианты проецирования данных с пробелами. К примеру, отсутствующие компоненты вектора данных могут быть восстановлены за счет ближайшей вершины, а уж потом уже "комплектный" вектор проецируется на ломаную.

Кусочно-линейная проекция на квадратную и гексагональную сетки

Для квадратной сетки предварительно проводится триангуляция, что переводит ее в гексагональную; таким образом, процесс проецирования для квадратной и гексагональной сетки один и тот же (за исключением произвола в выборе способа триангуляции квадратной сетки).

Проецирование производится либо в ближайшую вершину, либо на ближайшее ребро, либо на ближайшую грань – аналогично одномерному случаю.

III.7. Механическая интерпретация

В описанных линейных и квазилинейных моделях содержится очень сильное ограничение – балка была либо жесткой, либо она гибкая, но все равно проложена вдоль прямой, причем это оказывается существенным, например, в том случае, когда данные расположены не вдоль какой-либо прямой, а вдоль окружности (или хотя бы вдоль сильно изогнутой дуги).

Чтобы избежать этого, балку следует сделать упругой (определяется не прямой, а какой-либо кривой). Но тут возникают сложности в виде определения расстояния от точки до кривой в пространстве (тем более, что вместо точки может выступать линейное многообразие).

При использовании описанного выше метода, близкого методу самоорганизующихся карт Кохонена, искомая упругая балка (кривая – SOC) представляется в виде ломаной, узлы которой свободно соединены с данными (рис. 3.5).

Аналогично жесткому случаю, система через несколько итераций придет в равновесие. Причем их число будет конечно, т.к. на каждом шаге суммарная энергия растяжения пружинок уменьшается, а число возможных состояний (способов крепления узлов ломаной пружинками к данным) конечно.

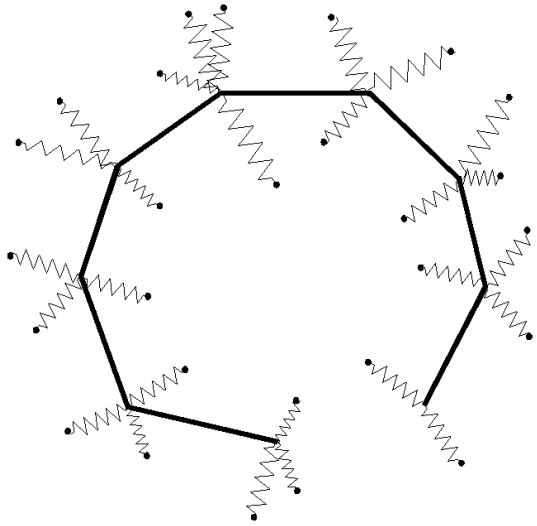

Рис. 3.5.

Введенные модули упругости представляют собой соответственно степень притяжения узлов ломаной друг к другу и степень сопротивления изгибу в узлах.