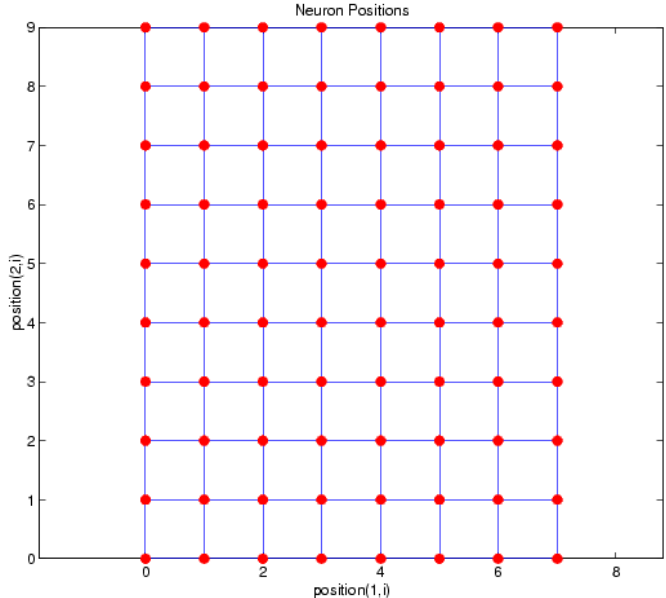
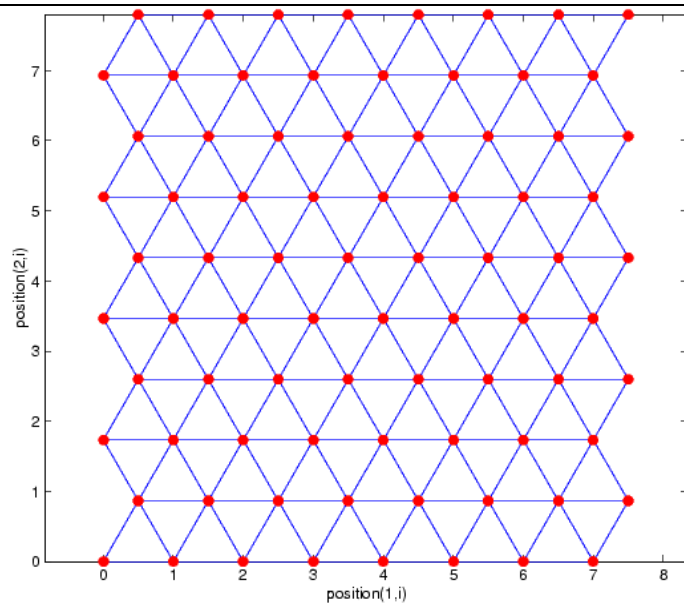


ЛАБОРАТОРНАЯ РАБОТА №6 "НЕЙРОННЫЕ СЕТИ. СЕТИ КОХОНЕНА"

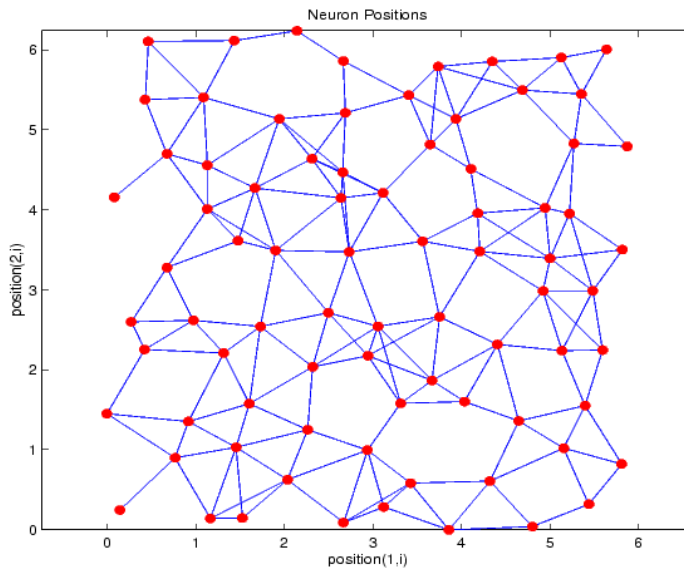
Приложение

Функция	Описание функции
newc	<p>Функция создания конкурирующего слоя Кохонена.</p> <p><code>net = newc(pr,m,klr,clr)</code> Входные параметры: <code>pr</code> – матрица размера $R \times 2$ минимальных и максимальных значений для количества наблюдений (объектов) R. <code>m</code> – число нейронов. <code>klr</code> – параметр функции настройки весов, значение по умолчанию 0.01. <code>clr</code> – параметр функции настройки смещений, значение по умолчанию 0.001. Архитектура конкурирующего слоя:</p>  <p>Конкурирующая передаточная функция <code>compet</code> для слоя возвращает выходы равные 0 для всех нейронов, за исключением нейрона-победителя, для которого выход равен 1. Если все смещения равны 0, то нейрон, чей весовой вектор наиболее близок к вектору входа имеет наименьший негативный вклад и становится победителем (его выход равен 1). Инициализация весов входов <code>net.IW{1,1}</code> производится автоматически с помощью вычисления функции средних значений <code>midpoint</code>. Инициализация весов смещений нейронов <code>net.b{1}</code> производится автоматически с помощью функции равных смещений <code>initcon</code>. Пример: <code>>>P = [.1 .8 .1 .9; .2 .9 .1 .8];</code> <code>>>net = newc([0 1; 0 1],2);</code> <code>>>net = train(net,P);</code></p>
train	<p>Функция обучения нейронной сети.</p> <p>Для задач адаптивной классификации вызов функции имеет следующий вид: <code>net = train(net,P)</code> Входные параметры: <code>net</code> – нейронная сеть. <code>P</code> – массив векторов входных сигналов. Функция <code>train</code> обучает нейронную сеть <code>net</code> в соответствии с функцией указанной в параметре сети <code>net.trainFcn</code> и с набором параметров обучения <code>net.trainParam</code>. Можно использовать параметры по умолчанию <code>trainFcn</code>:</p>

	<p>'trainr' и net.trainParam: .epochs, .goal, .show, .time. В результате выполнения функции оцениваются веса net.IW{1,1} и смещения net.b{1} (для слоя Кохонена).</p> <p>Пример:</p> <pre>net.trainParam.epochs = 500 net = newc([0 1; 0 1],2); net = train(net,P)</pre>
sim	<p>Функция классификации векторов входов P по разработанной нейронной сети net.</p> <p>Пример:</p> <pre>Y = sim(net,P)</pre>
vec2ind	<p>Функция конвертации классифицированных объектов, результатов вычисления функции sim, в индексы нейронов (классов или кластеров).</p> <p>Пример:</p> <pre>Yc = vec2ind(Y)</pre>
newsom	<p>Функция для создания самоорганизующейся карты Кохонена.</p> <pre>net = newsom(PR,[d1,d2,...],tfcn,dfcn,olr,osteps,tlr,tns)</pre> <p>Входные параметры:</p> <p>PR – матрица размера Rх2 минимальных и максимальных значений для R количества наблюдений (объектов).</p> <p>di – размер карты, значение по умолчанию = [5 8].</p> <p>tfcn - функция топологии карты, значение по умолчанию = 'hextop' ('gridtop', 'randtop').</p> <p>Топология сети типа gridtop:</p>  <p>Топология сети типа hextop:</p>



Топология сети типа randtop:



dfcn – функция расстояния, значение по умолчанию = 'linkdist' ('dist', 'mandist', 'boxdist').

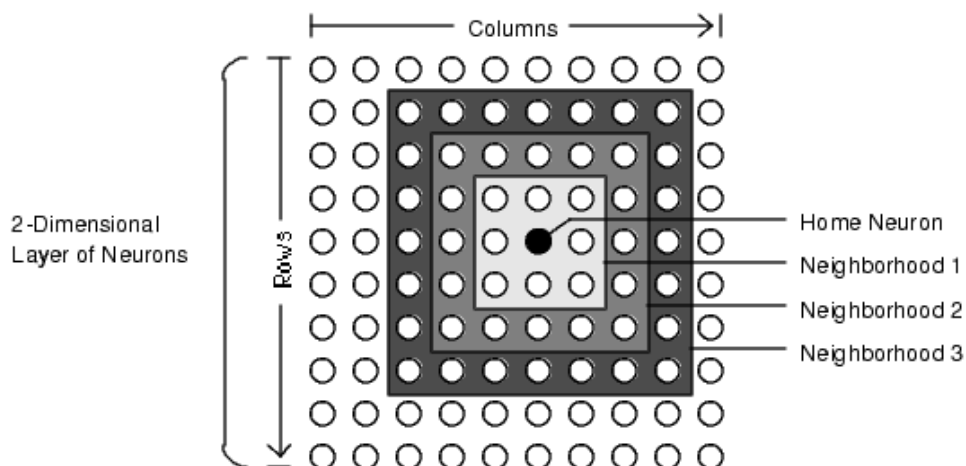
olr – параметр скорости обучения на этапе упорядочения, значение по умолчанию = 0.9.

osteps – число циклов обучения на этапе упорядочения, значение по умолчанию = 1000.

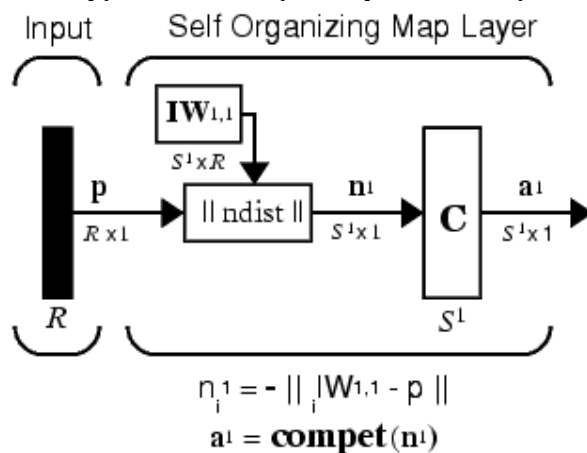
tlr – параметр скорости на этапе подстройки, значение по умолчанию = 0.02;

tnd – размер окрестности на этапе подстройки, значение по умолчанию = 1.

Примеры окрестностей нейрона-победителя, равные 1, 2, 3.



Архитектура слоя самоорганизующей карты Кохонена:



Весь процесс обучения карты Кохонена делится на два этапа:

- А) этап упорядоченности векторов весовых коэффициентов в пространстве признаков;
- Б) этап подстройки весов нейронов по отношению к набору векторов входа.

На этапе упорядочения используется фиксированное количество шагов. Начальный размер окрестности назначается равным максимальному расстоянию между нейронами для выбранной топологии. Затем уменьшается до величины, используемой на следующем этапе, и вычисляется по формуле: $nd = 1.00001 + (\max(d) - 1)(1 - s / \text{Osteps})$, где $\max(d)$ – максимальное расстояние между нейронами; s – номер текущего шага.

Параметр скорости обучения изменяется по правилу $lr = \text{tlr} + (\text{olr} - \text{tlr})(1 - s / \text{Osteps})$.

На этапе подстройки, который продолжается в течение оставшейся части процедуры обучения, размер окрестности остается постоянным и равным $nd = \text{tnd} + 0.00001$,

а параметр скорости обучения изменяется по следующему правилу $lr = \text{tlr} * \text{Osteps} / s$.

Параметр скорости обучения продолжает уменьшаться, но очень медленно. Малое значение окрестности и медленное уменьшение параметра скорости обучения хорошо настраивают сеть при сохранении размещения, найденного на предыдущем этапе. Число шагов на этапе подстройки должно

	<p>значительно превышать число шагов на этапе размещения (10^4 Osteps). На этом этапе происходит тонкая настройка весов нейронов по отношению к набору векторов входов.</p> <p>Нейроны карты Кохонена будут упорядочиваться так, чтобы при равномерной плотности векторов входа нейроны также были распределены равномерно. Если векторы входа распределены неравномерно, то и нейроны будут иметь тенденцию распределяться в соответствии с плотностью размещения векторов входа.</p>
--	---