

Многослойные нейронные сети  
прямого распространения  
Сеть Кохонена

# Сеть Кохонена

Если для решения задач классификации применять нейронные сети, то необходимо формализовать задачу.

Выберем в качестве входных данных вектор параметров единственного объекта. Результатом работы сети будет код класса, к которому принадлежит предъявленный на входе объект. В нейросетях принято кодирование номером канала. Поэтому сеть будет иметь  $M$  выходов, по числу классов, и чем большее значение принимает выход номер  $m_0$ , тем больше "уверенность" сети в том, что входной объект принадлежит к классу  $m_0$ .

# Сеть Кохонена

Каждый выход можно будет трактовать как вероятность того, что объект принадлежит данному классу. Все выходы образуют полную группу, т.к. сумма выходов равна единице, и объект заведомо относится к одному из классов. Выберем евклидову меру близости. В этом случае ядро класса, минимизирующее сумму мер близости для объектов этого класса, совпадает с центром тяжести объектов:

$$c^{m_0} = \frac{1}{N(m_0)} \sum_{p:m(p)=m_0} X^p$$

где  $N(m_0)$  - число объектов  $X^p$  в классе  $m_0$ .

При разбиении на классы должна быть минимизирована суммарная мера близости для всего множества входных объектов:

$$D = \sum_p \sum_i (X_i^p - c_i^{m(p)})^2 = \sum_p [(X^p, X^p) - 2(X^p, c^{m(p)}) + (c^{m(p)}, c^{m(p)})]$$

В этой сумме два слагаемых не зависят от способа разбиения и постоянны:

$$\sum_p (c^{m(p)}, c^{m(p)}) = const$$

$$\sum_p (X^p, X^p) = const$$

Поэтому задача поиска минимума  $D$  эквивалентна поиску максимума выражения:

$$\min D \rightarrow \max \sum_p \sum_i x_i^p c_i^{m(p)}$$

Запишем вариант алгоритма классификации для поиска максимума этой функции:

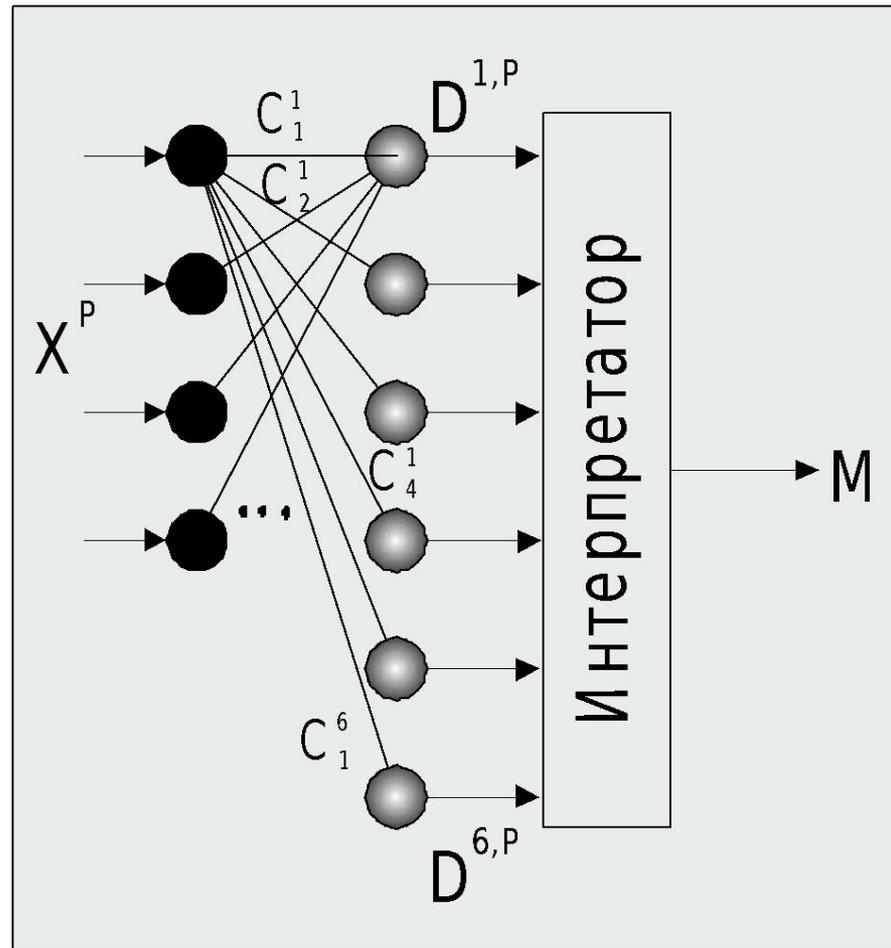
1. Цикл: для каждого вектора  $x^p$  {
  2. Цикл: для каждого  $m$  {
  3. Рассчитать  $\sum_i x_i^p c_i^m = D^{m,p}$   
} // конец цикла  $i$
  4. Находим  $m_0$ , для которого  $m_0 = \max \{D^{m,p}\}$
  5. Относим объект к классу  $m_0$ .
- } //конец цикла

Данный алгоритм легко реализуется в виде нейронной сети. Для этого требуется  $M$  сумматоров, находящихся все, и интерпретатора, находящего сумматор с максимальным выходом.

Выберем  $x_i^p$  в качестве входных сигналов и компоненты ядер  $c_i^m$  в качестве весовых коэффициентов. Тогда каждый формальный нейрон с числом входов, равным числу компонент во входном векторе, будет давать на выходе одну из сумм  $D^{m,p}$ .

Чтобы определить класс, к которому относится объект, нужно выбрать среди всех нейронов данного слоя один с максимальным выходом (данную операцию осуществляет интерпретатор. Интерпретатор — или программа, выбирающая нейрон с максимальным выходом, или слой нейронов с латеральным торможением, состоящий из нейронов с обратными связями.

Рассмотренная сеть нейронов, использующая евклидову меру близости для классификации объектов, называется *сетью Кохонена*



Нейроны слоя Кохонена генерируют сигналы  $D^{m,p}$ . Интерпретатор выбирает максимальный сигнал слоя Кохонена и выдает номер класса  $m$ , соответствующий номеру входа, по которому интерпретатором получен максимальный сигнал. Это соответствует номеру класса объекта, который был на входе, в виде вектора  $X^p$ .

Ядра  $c^m$  являются весовыми коэффициентами нейронов. Каждый нейрон Кохонена запоминает одно ядро класса, и отвечает за определение объектов в своем классе, т.е. величина выхода нейрона тем больше, чем ближе объект к данному ядру класса.

Общее количество классов совпадает с количеством нейронов Кохонена. Меняя количество нейронов, можно динамически менять количество классов.

Входные вектора сети Кохонена чаще всего нормируются:

$$\frac{X^p}{|X^p|} \rightarrow X^p \quad \text{или} \quad \frac{X^p}{\sum_p |X^p|^2} \rightarrow X^p$$

## Обучение слоя Кохонена

Задача обучения — научить сеть активировать один и тот же нейрон для похожих векторов на входе.

### *1. Присвоение начальных значений*

Обычно начальные значения в нейронных сетях выбираются малыми случайными числами. Для слоя Кохонена такой выбор возможен, но имеет недостатки. Если веса инициализируются случайными значениями с равномерным распределением ядер, то в областях пространства  $X$ , где мало входных векторов, ядра будут использоваться редко, т.к. мало будет похожих векторов.

В тех областях, где входных векторов много, плотность ядер окажется недостаточной, и непохожие объекты будут активировать один и тот же нейрон, т.к. более похожего ядра не найдется. Для устранения данной проблемы можно выделять ядра в соответствии с плотностью входных векторов. Но распределение входных векторов часто бывает заранее неизвестно. В этом случае помогает метод выпуклой комбинации.

## *2. Обучение сети*

Если число входных векторов равно числу ядер (т.е. нейронов), то обучение не нужно. Достаточно присвоить ядрам значения входных векторов и каждый вектор будет активировать свой нейрон Кохонена. Но чаще всего количество классов меньше числа входных векторов.

В этом случае веса сети настраиваются итеративным алгоритмом. Алгоритм аналогичен исходному алгоритму классификации, но коррекции весов проводятся после предъявления каждого входного вектора, а не после предъявления всех, как требует исходный алгоритм. Сходимость при этом сохраняется.

а) Присваиваем начальные значения весовым коэффициентам.

б) Подаем на вход один из векторов .

в) Рассчитываем выход слоя Кохонена , , и определяем номер выигравшего нейрона  $m_0$ , выход которого максимален:

$$m_0 = \max_m \{ D^{m,p} \}$$

г) Корректируем веса только выигравшего нейрона  $m_0$  :

$$w_{m_0} := w_{m_0} + \alpha (X^P - w_{m_0})$$

коррекция записана в виде векторного выражения (вектор весов  $w_{m_0}$  нейрона  $m_0$  имеет столько компонент, сколько их у входного вектора  $X^P$ ).  $\alpha$ -скорость обучения, малая положительная величина. Часто используют расписание с обучением, когда  $\alpha = \alpha(t)$  монотонно убывает. Требования к  $\alpha(t)$  те же, что и в случае многослойного перцептрона.

Веса корректируются так, что вектор весов приближается к текущему входному вектору. Скорость обучения управляет быстротой приближения ядра класса (вектора весов) ко входному вектору  $X^P$ .

Алгоритм выполняется до тех пор, пока веса не перестанут меняться.

# Метод выпуклой комбинации

Этот метод полезен при обучении, чтобы правильно распределить плотность ядер классов (векторов весов) в соответствии с плотностью входных векторов в пространстве  $X$ .

1. Присваиваем всем весам одно и то же начальное значение:

$$w_i = \frac{\dim X}{\sqrt{n}}$$

Вектора весов получают длину, равную единице, как требует нормировка. Все вектора весов одинаковы.

2. Задаем обучающее множество  $\{X^p\}$  и проводим обучение, но не с векторами  $X^p$ , а с векторами:

$\beta(t)X^p + \frac{1-\beta(t)}{\sqrt{n}}$  — время обучения,  $\beta(t)$  — монотонно возрастающая функция, меняющаяся при обучении от 0 до 1.

В начале обучения  $\beta(t) = 0$  и все обучающие вектора одинаковы и равны начальному значению весов. По мере обучения  $\beta(t)$  растет и обучающие вектора расходятся из точки с координатами  $\frac{1}{n}$  и приближаются к своим конечным значениям  $\beta$ , которые достигаются при  $\beta = 1$ . Каждый вектор весов "захватывает" группу или один обучающий вектор и отслеживает его по мере роста  $\beta$ .

Метод выпуклой комбинации дает правильное распределение плотности ядер. При этом в сети не остается "ненужных" необученных нейронов, которые бывают при обычном обучении. Когда вектор весов нейрона находится далеко от всех обучающих векторов, этот нейрон никогда не будет "выигрывать", и его веса не будут корректироваться при обучении. Выпуклая комбинация не оставляет в сети таких нейронов.

# Модификации алгоритма обучения сети Кохонена

1. *Чувство справедливости*: чтобы не допустить отсутствие обучения по любому из нейронов, вводится "чувство справедливости". Если нейрон чаще других выигрывает "состязание", т.е. получает максимальный выход чаще, чем в 1 из  $M$  случаев, то его значение выхода искусственно уменьшается, чтобы дать возможность выиграть другим нейронам. Это включает все нейроны сети в процесс обучения.

2. *Коррекция весов пропорционально выходу*: в этой модификации корректируются не только веса выигравшего нейрона, но и всех остальных, пропорционально их нормированному выходу. Нормировка выполняется по максимальному значению выхода слоя или по его среднему значению. Этот метод также исключает "мертвые" нейроны и улучшает распределение плотности весов.

# Режимы работы сети Кохонена

Обычная сеть Кохонена работает в режиме *аккредитации*. Это означает, что активируется единственный нейрон Кохонена с максимальным значением выхода. Можно не затормаживать остальные нейроны слоя Кохонена, а пронормировать выходные сигналы, например, функцией активации softmax:

$$OUT_j = \frac{e^{NET_j}}{\sum_i e^{NET_i}}$$

Тогда сумма всех выходов слоя будет равна единице и можно трактовать выходы, как вероятность отнесения объекта к каждому из классов.

Данный режим работы сети, когда активируется несколько нейронов одновременно, называется *режимом интерполяции*. Название режима объясняется тем, что если входной вектор  $X^P$  плавно меняется от одного вектора весов  $X^P = w^{m_1}$  к другому вектору весов  $X^P = w^{m_2}$ , то выход сети в режиме интерполяции (если применена функция softmax) будет плавно меняться от  $m_1$  к  $m_2$ , т.е. классификация оказывается непрерывной. Если же сеть работает в режиме аккредитации, выход изменится от  $m_1$  к  $m_2$  скачкообразно.