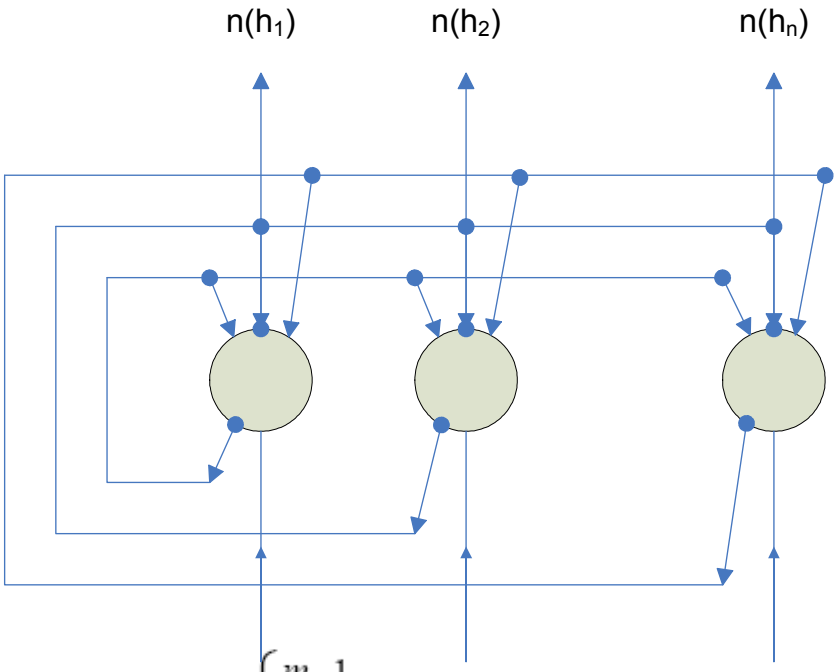
Сеть Хопфилда



$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad i = j \end{cases}$$

i, j - значения предсинаптического и постсинаптического сигналов

Сеть хопфилда состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах. характеристики процессов), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов.

Алгоритм функционирования сети следующий (p – номер итерации):

1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i,\ i = 0...n-1, \quad (2)$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от y_i означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p),\ j=0...n-1 \quad (3)$$

и новые значения аксонов

$$y_j(p+1) = f\left[s_j(p+1)\right] \quad (4)$$

3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы застабилизировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Энергия:

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i,$$

- w_{ij} - сила связи между i-ым и j-ым нейронами,
- θ_i - индивидуальные порог для каждого нейрона
- s_i - состояние нейрона.

E - мера близости к стабильному состоянию

Стохастический нейрон

Переключение нейрона происходит с вероятностью, зависящей от индуцированного локального поля, то есть передаточная функция определена как, таким образом нейрон активируется с вероятностью P(u) $f(u) = \begin{cases} 1,...P(u) \\ 0,...1 - P(u) \end{cases}$

где распределение вероятности P(u) обычно имеет вид сигмоида: $\sigma(u) = \frac{A(T)}{1 + \exp(-u/T)}$

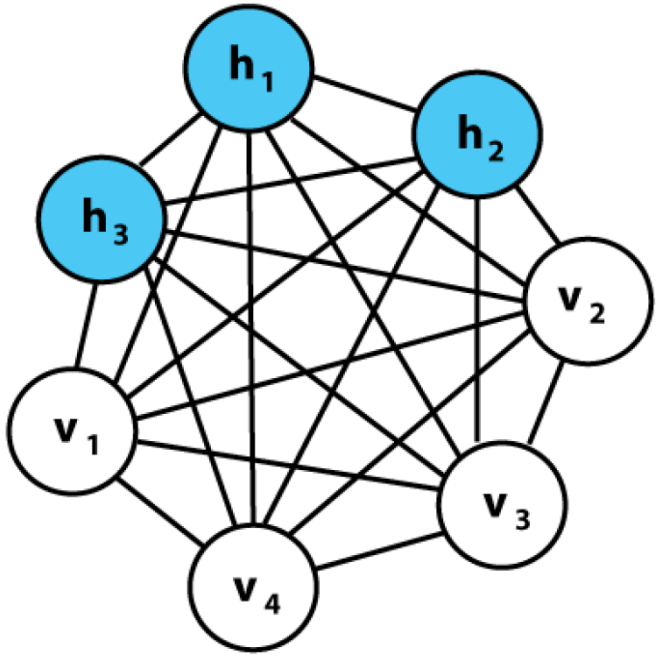
нормировочная константа A(T) вводится для условия нормализации распределения вероятности $\int_0^1 \sigma(u)du = 1$.

Параметр T — аналог температуры (но не температуры нейрона) и определяет беспорядок в нейронной сети. Если T устремить к 0, стохастический нейрон перейдет в обычный нейрон с пороговой функцией.

Стохастическая машина Больцмана:

Стохастическая рекуррентная нейронная сеть, изобретенная Джеффри Хинтоном и Терри Сейновски в 1985 году. Машина Больцмана может рассматриваться как стохастический генеративный вариант сети Хопфилда. Специалисты по статистике называют такие сети случайными марковскими полями. Эта сеть использует для обучения алгоритм имитации отжига и оказалась первой нейронной сетью, способной обучаться внутренним представлениям, решать сложные комбинаторные задачи. Несмотря на это, из-за ряда проблем, машины Больцмана с неограниченной связностью не могут использоваться для решения практических проблем. Если же связность ограничена, то обучение может быть достаточно эффективным для использования на практике.

Машина Больцмана является сетью нейронов с определенной для неё понятием "энергии". Расчет глобальной энергии производится идентичным по форме с сетью Хопфилда образом:



$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i,$$

Где $w_{i,j}$ сила связи между нейронами j и i.
s_i состояние, s_i {0,1}, нейрона i, θ_i порог для нейрона i.
Связи имеют следующие ограничения:

$w_{ii}=0$. (нейрон не может иметь связь с самим собой);

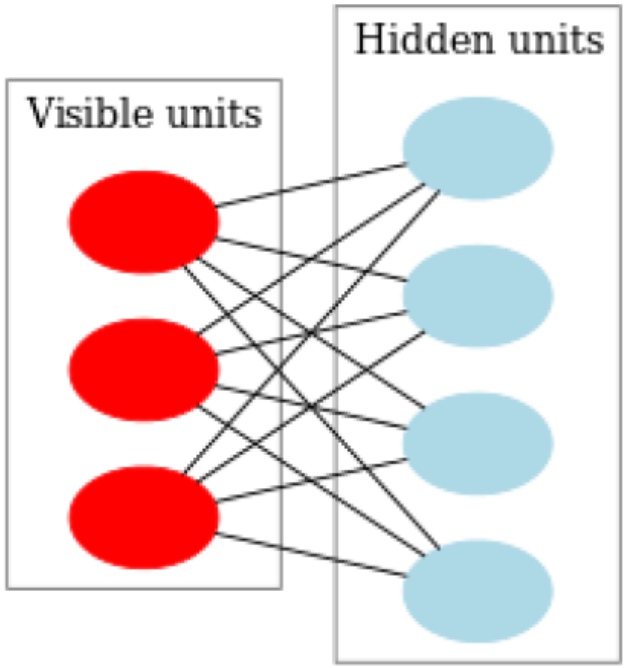
$w_{ij}=w_{ji}$(все связи являются симметричными).

Одним из недостатков сети Хопфилда является «стабилизация» состояния сети в локальном минимуме. Желательно, чтобы сеть переходила в глубокие минимумы энергии чаще, чем неглубокие, и чтобы относительная вероятность перехода сети в один из двух минимумов с разной энергией зависела только от соотношения их глубин. Это позволило бы управлять вероятностями получения конкретных выходных векторов путем изменения профиля энергетической поверхности за счет модификации весов связей. На основе этих соображений и построена машина Больцмана. Вероятность активности некоторого нейрона определяется на основе вероятностной функции Больцмана:

$P_k = \frac{1}{1 + \exp(-E_k / t)}$ где t — уровень теплового шума в сети; E_k — сумма весов связей к-го нейрона со всеми активными в данный момент нейронами.

Ограниченная машина Больцмана

Но если убрать связи внутри группы, чтобы получился двудольный граф, мы получим структуру модели RBM.



Особенность этой модели в том, что при данном состоянии нейронов одной группы, состояния нейронов другой группы будут независимы друг от друга.

RBM интерпретируются аналогично скрытым моделям Маркова.

У нас есть ряд состояний, которые мы можем наблюдать (видимые нейроны, которые предоставляют интерфейс для общения с внешней средой) и ряд состояний, которые скрыты, и мы не можем напрямую увидеть их состояние (скрытые нейроны). Но мы можем сделать вероятностный вывод относительно скрытых состояний, опираясь на состояния, которые мы можем наблюдать.

Введем следующие обозначения:

- w_{ij} - вес между i-ым нейроном
- a_i - смещение видимого нейрона
- b_j - смещение скрытого нейрона
- v_i - состояние видимого нейрона
- h_j - состояние скрытого нейрона

Мы будем рассматривать обучающее множество, состоящее из бинарных векторов. Предположим, что у нас n видимых нейронов и m скрытых. Введем понятие энергии для RBM:

$$E(v,h) = -\sum_{i}^{n} a_i v_i - \sum_{j}^{m} b_j h_j - \sum_{i}^{n}\sum_{j}^{m} w_{ij} v_i h_j$$

Нейросеть будет вычислять совместную вероятность всевозможных пар v и h следующим образом:

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)},$$

где Z - это статсумма следующего вида:

$$Z = \sum_{r}^{2^n} \sum_{t}^{2^m} e^{-E(v^r, h^t)}$$

Очевидно, что полная вероятность вектора v будет вычисляться суммированием по всем h:

$$p(v) = \sum_{t}^{M} P(v, h^t) = \frac{1}{Z} \sum_{t}^{M} e^{-E(v,h^t)}$$

Рассмотрим вероятность того, что при данном v одно из скрытых состояний $h_k = 1$. Для этого представим один нейрон, тогда энергия системы при 1 будет E_1, а при 0 будет E_0.

$$p(h_k = 1|v) = \frac{e^{-E_1}}{e^{-E_1} + e^{-E_0}} = \frac{1}{1 + e^{E_1 - E_0}} =$$

$$= \frac{1}{1 + e^{-b - \sum_i^n v_i w_{ik}}} = \sigma\Big(- b - \sum_i^n v_i w_{ik} \Big)$$

А так как при данном v все h_k не зависят друг от друга, то:

$$P(h|v) = \prod_j^m P(h_j|v)$$

Аналогичный вывод делается и для вероятности v при данном h.

Этот алгоритм придуман профессором Хинтоном в 2002 году, и он отличается своей простотой.

1. состояние видимых нейронов приравнивается к входному образу
2. выводятся вероятности состояний скрытого слоя
3. каждому нейрону скрытого слоя ставится в соответствие состояние «1» с вероятностью, равной его текущему состоянию
4. выводятся вероятности видимого слоя на основании скрытого
5. если текущая итерация меньше k, то возврат к шагу 2
6. выводятся вероятности состояний скрытого слоя

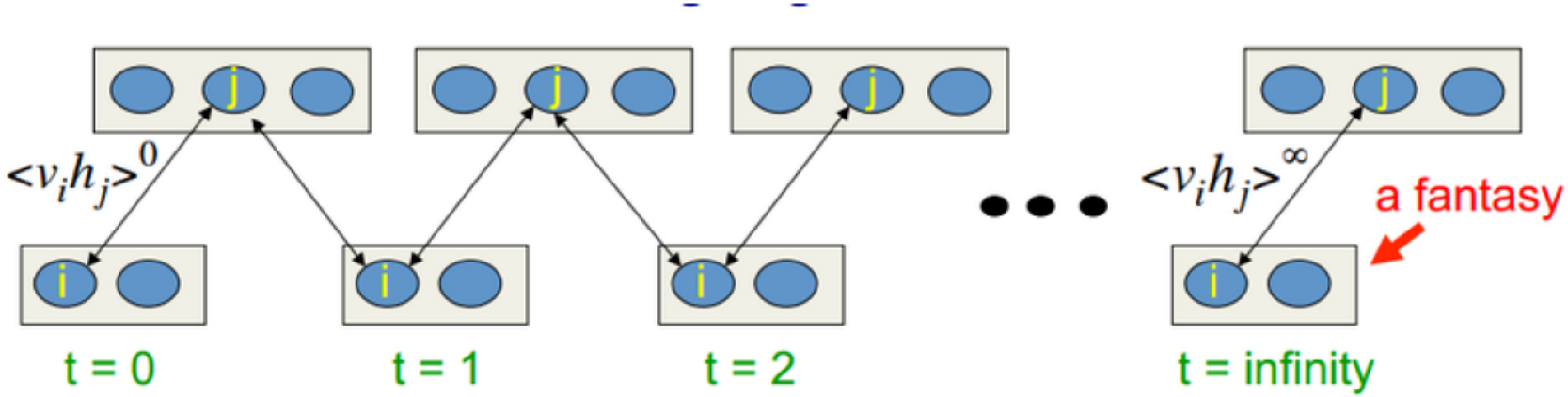
$$\frac{\partial \ln P(v^{(k)})}{\partial w_{ij}} = \sum_{t}^{M} v_i^{(k)} h_j^{(t)} P(h^{(t)}|v^{(k)}) - \sum_{r}^{N} \sum_{t}^{M} v_i^{(r)} h_j^{(t)} P(h^{(t)}, v^{(k)})$$

$$\frac{\partial \ln P(v^{(k)})}{\partial w_{ij}} = M \left[v_i^{(k)} h_j \right]_{\text{data}} - M \left[v_i h_j \right]_{\text{model}}$$

$$\frac{\partial \ln P(v^{(k)})}{\partial b_j} = \sum_{t} h_j^{(t)} P(h^{(t)}|v^{(k)}) - \sum_{r} \sum_{t} h^{(t)} P(v^{(r)}, h^{(t)}) =$$

$$M \left[h_j \right]_{\text{data}} - M \left[h_j \right]_{\text{model}}$$

$$\frac{\partial \ln P(v^{(k)})}{\partial a_i} = v^{(k)} - \sum_{r} v_i^{(k)} P(v^{(r)}|h^{(t)}) = v_i^k - M \left[v_i \right]_{\text{model}}$$



- $\Delta w_{ij} = \eta\left(M[v_i h_j]^{(0)} - M[v_i h_j]^{(\infty)}\right)$
- $\Delta a_i = \eta\left(v_i - M[v_i]^{(\infty)}\right)$
- $\Delta b_j = \eta\left(M[h_j]^{(0)} - M[h_j]^{(\infty)}\right)$

1. Подать вектор из обучающей выборки на видимый слой $v^0 = x^i$
2. Вычислить состояние h_j для всех скрытых нейронов по формуле (5)
3. Вычислить состояния v_j для всех видимых нейронов по формуле (4)
4. Повторять пункты 2,3… k раз, где k - параметр сети.
5. Вычислить

$$\Delta W = \Delta W + \mu(\sum h_j^0 v_j^0 - \sum h_j^k v_j^k)$$

$$\Delta b = \Delta b + \mu(v^0 - v^k)$$

$$\Delta c = \Delta c + \mu(h^0 - h^k)$$

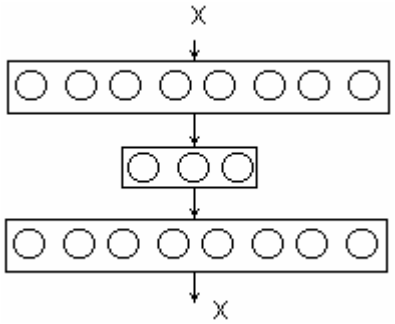
6. $h_j^0, v_j^0, h_j^k, v_j^k$ – состояния нейронов полученные на итерации с номерами 0 и k соответственно, μ- скорость обучения
7. Перейти к пункту 1, взяв следующий вектор x^{i+1} из обучающей выборки

$$v_i \sim P(v_i = 1, h) = \frac{1}{1 + e^{a_i + \sum_{j \in h} h_j w_{ij}}} \quad (4)$$

$$h_j \sim P(h_j = 1, v) = \frac{1}{1 + e^{b_j + \sum_{i \in v} v_i w_{ij}}} \quad (5)$$

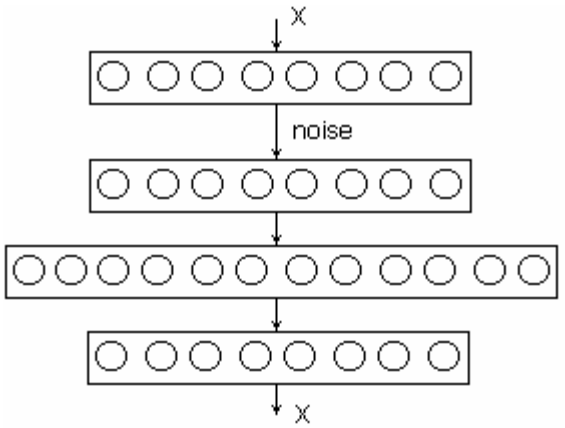
Модель autoencoder.

Обычный autoencoder это трёхслойная нейронная сеть прямого распространения, в которой вторая половина сети зеркально повторяет первую. Как следует из названия, autoencoder отображает входной образ в себя, т.е. эта сеть обучается таким образом, что бы воспроизводить учебное множество.



Autoencoder можно использовать для решения разных задач, например как ассоциативную память или метод сжатия данных (понижения размерности) с возможностью восстанавливать исходный образ с некоторой точностью, для этого размер скрытого слоя задают меньшим (например в 2 раза) чем размер входного (и соответственно выходного) слоя.

Autoencoder также используют как систему шумоподавления (denoising autoencoder). Для этого при обучении на входной образ накладывают шум и требуют от сети восстановить исходный образ, в этом случае размер скрытого слоя задают большим чем размер входного (и соответственно выходного) слоя.



Обучать autoencoder можно как обычный перцептрон с одним скрытым слоем используя градиентные методы.
Для сетей с большим количеством слоев существует проблема исчезающего градиента.

Предобучение многослойной нейронной сети с помощью autoencoder.

Предобучение представляет собой следующую процедуру -- мы берём пары соседних слоёв глубокой сети начиная с входного слоя и конструируем из этой пары autoencoder, путём добавления выходного слоя идентичного входному. Процедура последовательно повторяется для всех слоёв сети.

Алгоритм

1. загрузка учебного набора данных X0
2. определить параметры сети - количество (N) и размеры слоёв, установить номер текущего слоя i=0
3. построить автоенкодер для слоёв i, i+1
4. обучить автоенкодер на наборе Xi
5. убрать вспомогательный (выходной) слой автоенкодера
6. сохранить веса Wi связей слоёв i,i+1

7. если есть ещё пары слоёв для обработки (i<N-2), то переход на следующий пункт, иначе переход на п.10
8. сгенерировать набор данных Xi+1 для следующего автоенкодера, для этого пропустить через пару слоёв i,i+1 набор данных Xi,
9. переход п.3

После этой процедуры сеть целиком дообучается одним из градиентных методов.