Ограниченная машина Больцмана (Restricted Boltzmann machine)

Введение

Ограниченная машина Больцмана (RBM) — это генеративная стохастическая искусственная нейронная сеть, которая моделирует распределение вероятности входных данных.

Как следует из названия, RBM являются разновидностью машин Больцмана с ограничением. Ограничение состоит в том, что нейроны данной сети образуют неориентированный двудольный граф без петель. Данный факт означает, что множество нейронов разбивается на две группы (видимые и скрытые нейроны), причем связана любая пара узлов, принадлежащих разным группам, а узлы одной группы связей не имеют. Такая нейронная сеть содержит симметричные синаптические соединения, т.е. видимые и скрытые нейроны взаимодействуют в обоих направлениях. Указанное ограничение позволяет использовать более эффективные алгоритмы обучения по сравнению с теми, которые доступны для общего класса машин Больцмана. Кроме того, особенностью ограниченных машин Больцмана является возможность их обучения без учителя, т.е. не требуется разметка тренировочных данных, на которых настраиваются веса нейронной сети.

Обозначения и вывод формул

Введем следующие обозначения.

- Состояния нейронов видимого слоя. Видимые нейроны являются **стохастическими**, т.е. могут находиться в двух состояниях "включен" или "выключен". Принимают значения 0 или 1. 1 означает, что нейрон активирован ("включен"), 0 - не активирован ("выключен").

$$v_1, v_2, \ldots, v_{N_v}; v_i \in \{0, 1\}, i = \overline{1, N_v}, v \in R^{N_v}$$

- Состояния нейронов скрытого слоя. Скрытые нейроны по аналогии с видимыми являются стохастическими и принимают значения 0 или 1.

$$h_1, h_2, \ldots, h_{N_h}; h_i \in \{0, 1\}, i = \overline{1, N_h}, h \in R^{N_h}$$

- Матрица связей между нейронами видимого и скрытого слоя:

$$W \in R^{N_v \times N_h}$$

- Вес связи между i-ым нейроном видимого слоя и j-ым нейроном скрытого слоя:

$$w_{ij}, i = \overline{1, N_v}, j = \overline{1, N_h}$$

- Смещение видимого нейрона:

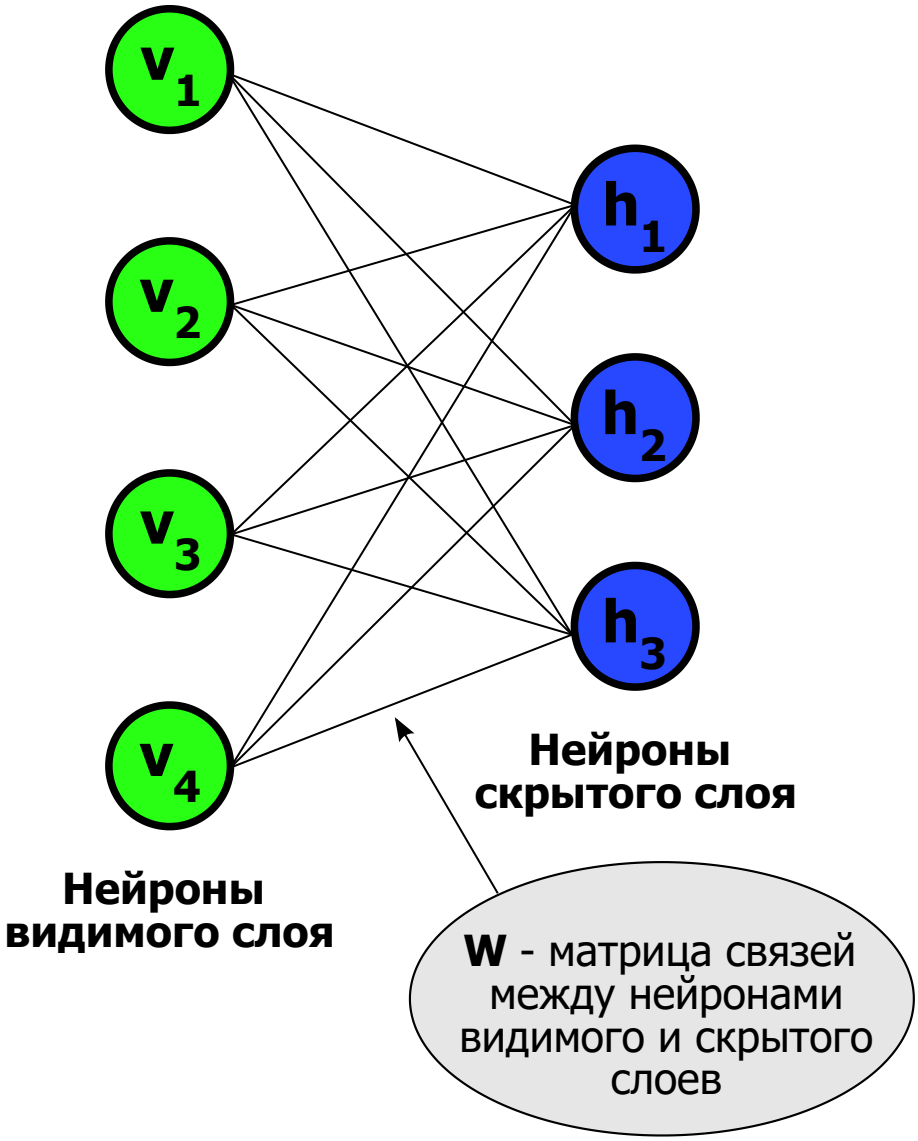
$$a_i, i = \overline{1, N_v}$$

- Смещение скрытого нейрона:

$$b_j, j = \overline{1, N_h}$$

Примечания:

1. Состояние нейрона может принимать значения 1 и -1, 1 означает, что нейрон активирован ("включен"), -1 - не активирован ("выключен").
2. Вектора смещений можно внести в матрицу связей между видимыми и скрытыми нейронами, если ввести фиктивный нейрон, который всегда находится в активированном состоянии. В результате получается равносильная модель RBM.
3. В общем случае состояния нейронов могут принимать вещественные значения.



Энергия и вероятности

- Введем понятие энергии для ограниченной машины Больцмана:

$$E(v,h) = -\sum_{i=1}^{N_v} a_i v_i - \sum_{j=1}^{N_h} b_j h_j - \sum_{i=1}^{N_v}\sum_{j=1}^{N_h} w_{ij} v_i h_j = -a^\top v - b^\top h - v^\top W h$$

Примечания:

1. Если вектора смещений внесены в матрицу связей между видимыми и скрытыми нейронами, то функция энергии имеет упрощенный вид.

$$E(v,h) = -\sum_{i=0}^{N_v}\sum_{j=0}^{N_h} w_{ij}v_i h_j = -v^\top W h$$

2. Если вектора смещений внесены в матрицу связей между видимыми и скрытыми нейронами, и состояния нейронов принимают значения 1 и -1 (1 соответствует состоянию "включен", -1 - состоянию "выключен"), то функция энергии отличается от предыдущей наличием дополнительного коэффициента.

$$E(v,h) = -\frac{1}{2}\sum_{i=0}^{N_v}\sum_{j=0}^{N_h} w_{ij}v_i h_j = -\frac{1}{2}v^\top W h$$

- Моделируемая совместная плотность вероятности представляет собой распределение Гиббса.

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

где Z — это статсумма следующего вида (суммирование по всевозможным векторам v и h, L и S - количество образов видимых и скрытых векторов соответственно):

$$Z = \sum_{l=0}^{L-1}\sum_{s=0}^{S-1} e^{-E(v^{(l)},h^{(s)})}$$

- Полная вероятность вектора v (суммирование по всевозможным векторам h):

$$p(v) = \frac{1}{Z}\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})}$$

Вычисление условных вероятностей

- Вероятность того, что при данном входном векторе v активируется скрытый нейрон $h_k = 1$:

$$p(h_k = 1|v) = \frac{p(v,h_k=1)}{p(v)} = \frac{p(v,h_k=1)}{p(v,h_k=0)+p(v,h_k=1)} = \frac{e^{-E(v,h_k=1)}}{e^{-E(v,h_k=1)}+e^{-E(v,h_k=0)}}$$
$$= \frac{1}{1+e^{E(v,h_k=1)-E(v,h_k=0)}} = \frac{1}{1+e^{-b_k-\sum_{i=1}^{N_v} w_{ik}v_i}} = \sigma(b_k+\sum_{i=1}^{N_v} w_{ik}v_i),$$

где $\sigma(z) = \frac{1}{1+e^{-z}}$ - сигмоидальная функция.

- Вероятность того, что при данном скрытом векторе h активируется видимый нейрон $v_k = 1$:

$$p(v_k = 1|h) = \frac{p(v_k=1,h)}{p(h)} = \frac{e^{-E(v_k=1,h)}}{e^{-E(v_k=1,h)}+e^{-E(v_k=0,h)}} = \frac{1}{1+e^{E(v_k=1,h)-E(v_k=0,h)}}$$
$$= \frac{1}{1+e^{-a_k-\sum_{j=1}^{N_h} w_{kj}h_j}} = \sigma(a_k+\sum_{j=1}^{N_h} w_{kj}h_j)$$

- Поскольку при фиксированном входном векторе v состояния скрытых нейронов h_j независимы друг от друга (как и при фиксированном h состояния видимых нейронов v_i), то условные вероятности для векторов имеют вид:

$$p(h|v) = \prod_{j=1}^{N_h} p(h_j|v)$$

$$p(v|h) = \prod_{i=1}^{N_v} p(h|v_i)$$

Цель обучения

Состояния видимых нейронов можно наблюдать, поскольку видимые нейроны соответствуют входным данным, но состояния скрытых нейронов напрямую увидеть нельзя. Несмотря на это, опираясь на наблюдаемые состояния, можно сделать вероятностный вывод относительно скрытых состояний. После обучения модели можно делать вероятностные выводы (при помощи теоремы Байеса) о том, какими являются видимые нейроны, опираясь на скрытые. Таким образом, **цель** - обучить модель таким образом (настроить ее параметры), чтобы полученный в результате работы RBM вектор состояний видимых нейронов был близок к входному состоянию, из которого он был восстановлен. Множество весов RBM реализует совершенную модель, если оно приводит к такому же распределению вероятности состояния видимых элементов, что и подача входных векторов.

Данную цель можно достигнуть путем максимизации вероятности $p(v)$. Применяя метод максимального правдоподобия, вместо максимизации вероятности $p(v)$, можно максимизировать натуральный логарифм от нее $p(v)$. Максимум $p(v)$ соответствует максимуму $\ln p(v)$, поскольку при переходе от функции x к $\ln x$ монотонность не изменяется.

Максимизация вероятности $p(v)$

Найдем производную от натурального логарифма $p(v)$ по весу w_{ij}:

$$\frac{\partial \log(p(v))}{\partial w_{ij}} = \frac{1}{p(v)} \frac{\partial p(v)}{\partial w_{ij}} = \frac{1}{p(v)} \frac{\partial}{\partial w_{ij}} \left(\frac{\sum_{s=0}^{S-1} e^{-E\left(v,h^{(s)}\right)}}{Z} \right)$$
$$= \frac{1}{p(v)} \frac{\frac{\partial}{\partial w_{ij}} \left(\sum_{s=0}^{S-1} e^{-E\left(v,h^{(s)}\right)} \right) Z - \sum_{s=0}^{S-1} e^{-E\left(v,h^{(s)}\right)} \frac{\partial Z}{\partial w_{ij}}}{Z^2},$$

где

$$p(v) = \frac{1}{Z} \sum_{s=0}^{S-1} e^{-E\left(v,h^{(s)}\right)},$$

$$Z = \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} e^{-E\left(v^{(l)},h^{(s)}\right)},$$

$$E(v,h) = -\sum_{i=1}^{N_v} a_i v_i - \sum_{j=1}^{N_h} b_j h_j - \sum_{i=1}^{N_v} \sum_{j=1}^{N_h} w_{ij} v_i h_j.$$

Вычислим производные:

$$\frac{\partial}{\partial w_{ij}} \left(\sum_{s=0}^{S-1} e^{-E\left(v,h^{(s)}\right)} \right) = \sum_{s=0}^{S-1} v_i h_j^{(s)} e^{-E\left(v,h^{(s)}\right)}$$

$$\frac{\partial Z}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{l=0}^{L-1} \sum_{s=0}^{S-1} e^{-E\left(v^{(l)}, h^{(s)}\right)} \right) = \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} e^{-E\left(v^{(l)}, h^{(s)}\right)}$$

Подставим полученные производные в формулу производной от логарифма плотности распределения вероятности входных данных.

$$\frac{\partial \ln(p(v))}{\partial w_{ij}} = \frac{1}{p(v)} \frac{\frac{\partial}{\partial w_{ij}} (\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})}) Z - \sum_{s=0}^{S-1} e^{-E(v,h^{(s)})} \frac{\partial Z}{\partial w_{ij}}}{Z^2} =$$

$$= \frac{Z}{\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})}} \frac{Z(\sum_{s=0}^{S-1} v_i h_j^{(s)} e^{-E(v,h^{(s)})}) - (\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})})(\sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} e^{-E(v^{(l)},h^{(s)})})}{Z^2} =$$

$$= \frac{\sum_{s=0}^{S-1} v_i h_j^{(s)} e^{-E(v,h^{(s)})}}{\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})}} - \frac{\sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} e^{-E(v^{(l)},h^{(s)})}}{Z} =$$

$$= (\frac{Z}{\sum_{s=0}^{S-1} e^{-E(v,h^{(s)})}})(\sum_{s=0}^{S-1} v_i h_j^{(s)} \frac{e^{-E(v,h^{(s)})}}{Z}) - \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} \frac{e^{-E(v^{(l)},h^{(s)})}}{Z} =$$

$$= \sum_{s=0}^{S-1} v_i h_j^{(s)} \frac{p(v, h^{(s)})}{p(v)} - \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} p(v^{(l)}, h^{(s)}) =$$

$$= \sum_{s=0}^{S-1} v_i h_j^{(s)} p(h^{(s)}|v) - \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} p(v^{(l)}, h^{(s)})$$

Таким образом, получаем формулу для вычисления производных функции ошибки по весам сети.

$$\frac{\partial \ln(p(v))}{\partial w_{ij}} = \sum_{s=0}^{S-1} v_i h_j^{(s)} p\left(h^{(s)}|v\right) - \sum_{l=0}^{L-1} \sum_{s=0}^{S-1} v_i^{(l)} h_j^{(s)} p\left(v^{(l)}, h^{(s)}\right) = \mathsf{E}_{data}[v_i h_j] - \mathsf{E}_{model}[v_i h_j],$$

где $\mathrm{E}(.)$ - математическое ожидание.

Аналогично, можно получить формулы для производных по смещениям:

$$\frac{\partial \ln(p(v))}{\partial a_i} = \mathsf{E}_{data}[v_i] - \mathsf{E}_{model}[v_i], \frac{\partial \ln(p(v))}{\partial b_j} = \mathsf{E}_{data}[h_j] - \mathsf{E}_{model}[h_j]$$

Правила обновления параметров модели

Введем параметр η (learning rate), отвечающий за скорость обучения модели (он влияет на то, насколько сильно изменяются веса при обучении). Тогда получаем следующие правила обновления параметров модели (весов и смещений):

$$\Delta w_{ij} = \eta(\mathsf{E}_{data}[v_i h_j] - \mathsf{E}_{model}[v_i h_j])$$

$$\Delta a_i = \eta(\mathsf{E}_{data}[v_i] - \mathsf{E}_{model}[v_i])$$

$$\Delta b_j = \eta(\mathsf{E}_{data}[h_j] - \mathsf{E}_{model}[h_j])$$

Алгоритм обучения Contrastive Divergence (CD-k)

Алгоритм обучения имеет название **алгоритма контрастной дивергенции** (Contrastive Divergence, CD-k). Алгоритм основывается на процедуре **сэмплирования Гиббса** (Gibbs sampling) - процедура получения

значения случайной величины на основании ее плотности распределения. Общая схема алгоритма имеет вид:

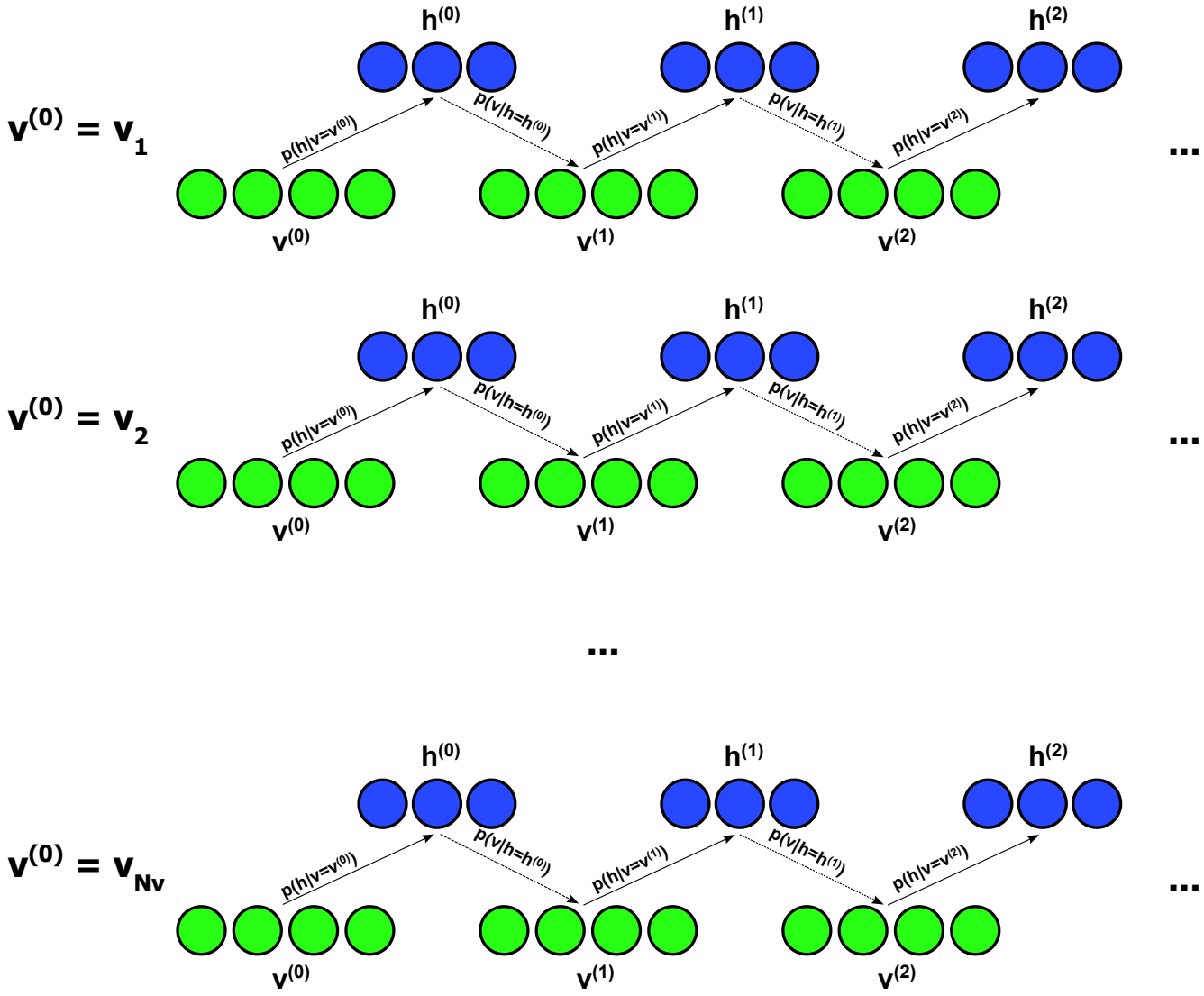
1. Состояние видимых нейронов x приравнивается к входному образу сети $v^0 = v^{(0)}$ ($v^{(0)}$ - элемент тренировочных данных).
2. Реализуется цикл по индексу t, состоящий из k итераций (k - параметр алгоритма). При этом осуществляется сбор статистики для вычисления математических ожиданий в формулах коррекции весов.

    A. Выводятся вероятности состояний скрытого слоя из распределения $p(h|v = v^t)$, из полученного распределения сэмплируется вектор h^t.
    B. Выводятся вероятности состояний видимого слоя из распределения $p(v|h = h^{t-1})$, из полученного распределения сэмплируется вектор v^t.

1. Выполняется коррекция весов сети и переход к следующему входному образу (элементу тренировочной выборки).

Иллюстрация алгоритма:

$$S = (v_1, v_2, \ldots, v_{Nv})$$



Псевдокод алгоритма приведен ниже (ссылка на источник).

Algorithm 1. k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S

Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$

forall the $\boldsymbol{v} \in S$ **do**

  $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$

  for $t = 0, \ldots, k-1$ **do**

    for $i = 1, \ldots, n$ **do** sample $h_i^{(t)} \sim p(h_i \mid \boldsymbol{v}^{(t)})$

    for $j = 1, \ldots, m$ **do** sample $v_j^{(t+1)} \sim p(v_j \mid \boldsymbol{h}^{(t)})$

  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**

    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \mid \boldsymbol{v}^{(0)}) \cdot v_j^{(0)} - p(H_i = 1 \mid \boldsymbol{v}^{(k)}) \cdot v_j^{(k)}$

    $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$

    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \mid \boldsymbol{v}^{(0)}) - p(H_i = 1 \mid \boldsymbol{v}^{(k)})$

Работа данного алгоритма далее будет рассмотрена на простом примере. Прежде чем к нему перейти, рассмотрим одну из возможных реализаций процедуры сэмплирования.

Сэмплирование значений случайной величины из распределения

Пусть в результате вычисления сигмоидальной функции была получена вероятность 0.6723. Округлим её до двух знаков дробной части и получим число 0.67. То есть:

$$p(h|v_0) = \sigma(b + W^\top v)_0 = 0.6723 \approx 0.67$$

Теперь можно поступить следующим образом:

> 1. Заведем массив A из 100 элементов, в котором первые 0.67 * 100 = 67 элементов будут единицами, а остальные 100 - 67 = 33 элемента — нулями.
> 2. Сгенерируем псевдослучайное число следующего вида:
>
> $$\xi \sim U[1, 100],$$
>
> где U[1, 100] — непрерывное равномерное распределение на отрезке [1, 100].
> 3. Будем считать, что (с учетом того, что индексы в массиве начинаются с нуля):
>
> $$h_0 = A[\xi - 1]$$

Можно, например, округлять вероятность до трех знаков дробной части, и заводить массив на 1000 элементов, но это более затратно.

Пошаговая иллюстрация процедуры обучения

Рассмотрим следующий пример. Пусть имеются данные об оценках трех университетских курсов — математического анализа, линейной алгебры и программирования. Каждому из них студенты поставили оценку 0 или 1, где 1 означает, что предмет студенту нравится, и 0 в противном случае.

Студент	Математический анализ	Линейная алгебра	Программирование
Андрей	0	0	1
Павел	1	0	1
Вадим	0	1	0

Рассмотрим следующую матрицу:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

По столбцам в ней записаны значения для нейронов видимого слоя. На первой итерации алгоритма рассмотрим первый столбец матрицы S:

$$v^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Будем считать, что изначально вектора сдвигов являются нулевыми, а матрицу весов заполним случайным образом:

$$a = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, W = \begin{pmatrix} 1 & -1 \\ -2 & 2 \\ 1 & -1 \end{pmatrix}$$

Посчитаем вероятности для нейронов скрытого слоя:

$$p(h^{(0)}|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \sigma(\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & -2 & 1 \\ -1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}) = \sigma(\begin{pmatrix} 1 \\ -1 \end{pmatrix}) = \begin{pmatrix} \sigma(1) \\ \sigma(-1) \end{pmatrix}$$
$$= \begin{pmatrix} 0.7311 \\ 0.2689 \end{pmatrix}$$

В соответствии с полученными вероятностями проведем сэмплирование: каждый скрытый нейрон с полученной вероятностью принимает состояние 1. Будем считать, что скрытый нейрон h_1 принимает значение 1 (вероятность больше 0.7), а скрытый нейрон h_2 — значение 0 (вероятность около 0.27). Но в процессе сэмплирования значения могли получиться иными (в данном примере будем считать, что если вероятность больше 0.5, то нейрон принимает значение 1). Таким образом:

$$h^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Теперь попробуем восстановить исходный вектор, используя $h^{(0)}$:

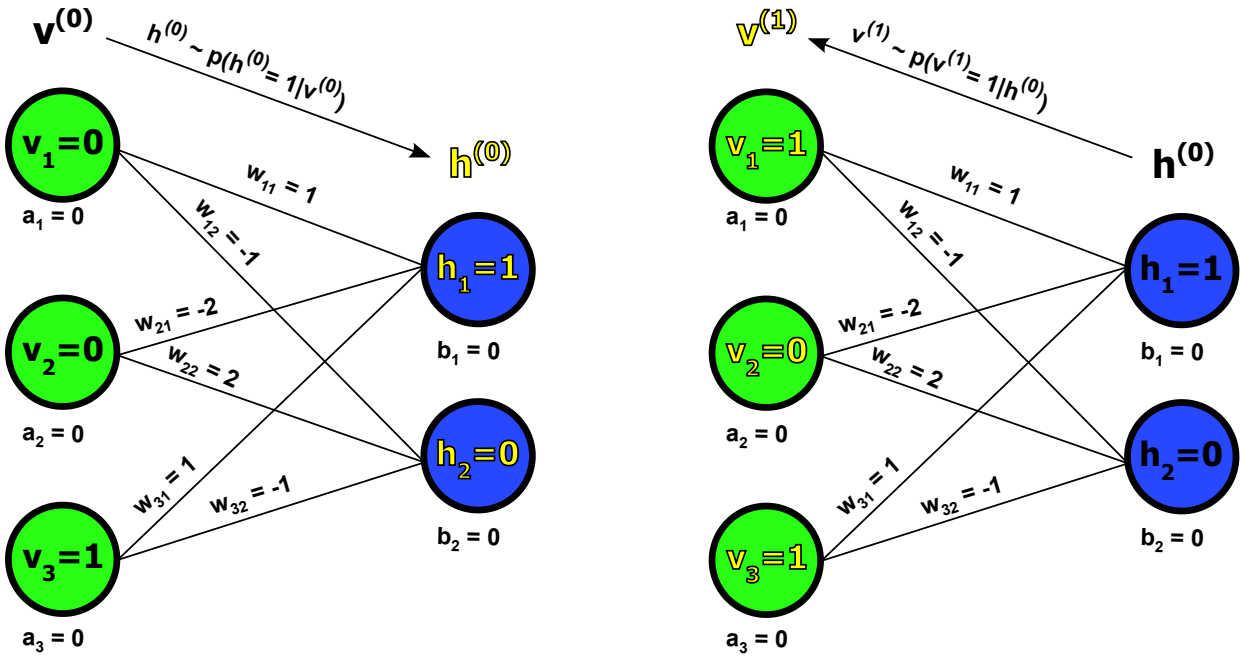
$$p(v^{(1)}|h^{(0)}) = \sigma(a + Wh^{(0)}) = \sigma(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & -1 \\ -2 & 2 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \sigma(\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}) = \begin{pmatrix} \sigma(1) \\ \sigma(-2) \\ \sigma(1) \end{pmatrix}$$
$$= \begin{pmatrix} 0.7311 \\ 0.1192 \\ 0.7311 \end{pmatrix}$$

Предположим, что в процессе сэмплирования видимые нейроны v_1 и v_3 приняли значение 1 (вероятность больше 0.7), а видимый нейрон v_2 — значение 0 (вероятность около 0.12). Получаем:

$$v^{(1)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Ниже наглядно показана первая итерация алгоритма:

Первая итерация

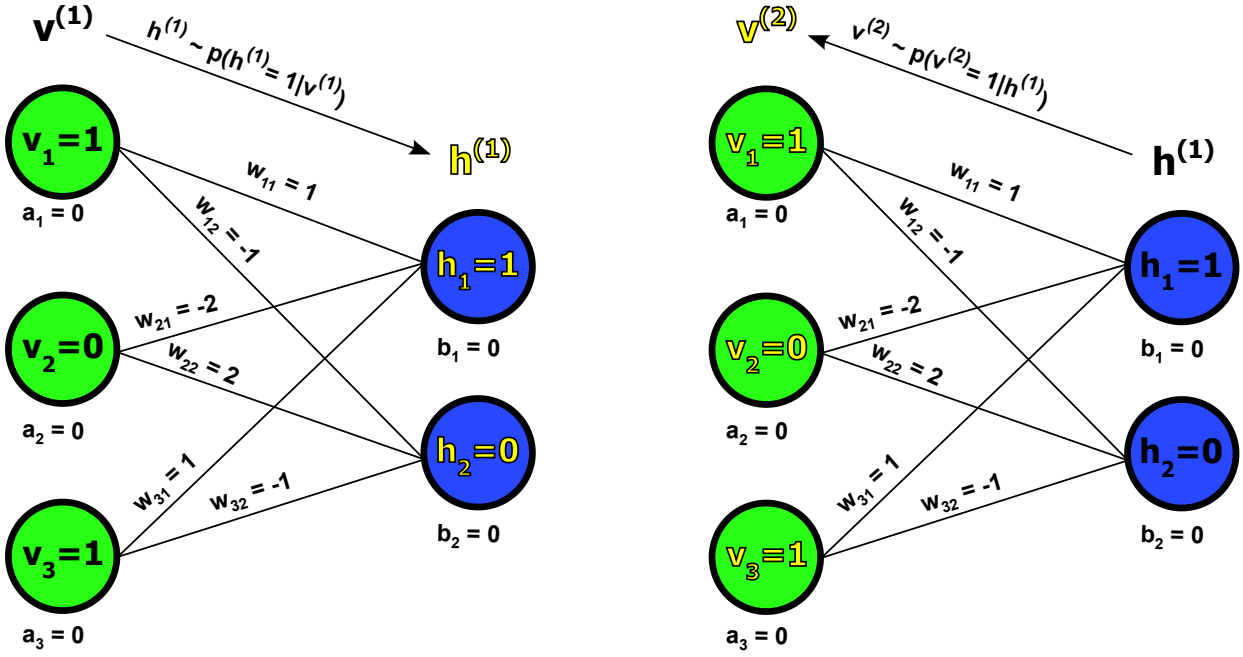


Будем считать, что мы используем двухшаговый алгоритм контрастной дивергенции (CD-2), поэтому необходимо выполнить еще одну аналогичную итерацию:

$$p(h^{(1)}|v^{(1)}) = \sigma(b + W^\top v^{(1)}) = \begin{pmatrix} 0.8808 \\ 0.1192 \end{pmatrix}, h^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$p(v^{(2)}|h^{(1)}) = \sigma(a + Wh^{(1)}) = \begin{pmatrix} 0.7311 \\ 0.1192 \\ 0.7311 \end{pmatrix}, v^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Вторая итерация



Последний шаг алгоритма завершен, и теперь необходимо обновить веса по следующим формулам (в псевдокоде не учитывался гиперпараметр η, отвечающий за скорость обучения, а точнее за то, насколько сильно меняются веса, но мы его теперь учтем; кроме того, вектора в нем обозначались другими буквами):

$$w_{ij} = w_{ij} + \eta(p(h_j = 1|v^{(0)})v_i^{(0)} - p(h_j = 1|v^{(k)})v_i^{(k)})$$

$$a_i = a_i + \eta(v_i^{(0)} - v_i^{(k)})$$

$$b_j = b_j + \eta(p(h_j = 1|v^{(0)}) - p(h_j = 1|v^{(k)})),$$

где k = 2, поскольку используется CD-2, а индексы меняются в следующих диапазонах (N_v = 3, N_h = 2):

$$i = \overline{1, N_v}, j = \overline{1, N_h}$$

Для обновления весов нам понадобятся следующие вектора:

$$v^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, v^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$p(h = 1|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \begin{pmatrix} 0.7311 \\ 0.2689 \end{pmatrix}$$

$$p(h = 1|v^{(2)}) = \sigma(b + W^\top v^{(2)}) = \begin{pmatrix} 0.8808 \\ 0.1192 \end{pmatrix}$$

$$a = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, W = \begin{pmatrix} 1 & -1 \\ -2 & 2 \\ 1 & -1 \end{pmatrix}$$

Кроме того, будем считать, что learing rate принимает следующее значение:

$$\eta = 0.5$$

Теперь обновим значения весов:

$$a_1 = 0 + \frac{1}{2}(0 - 1) = -0.5$$

$$a_2 = 0 + \frac{1}{2}(0 - 0) = 0$$

$$a_3 = 0 + \frac{1}{2}(1 - 1) = 0$$

$$b_1 = 0 + \frac{1}{2}(0.7311 - 0.8808) = -0.07485$$

$$b_2 = 0 + \frac{1}{2}(0.2689 - 0.1192) = 0.07485$$

$$w_{11} = 1 + \frac{1}{2}(0.7311 \cdot 0 - 0.8808 \cdot 1) = 0.5596$$

$$w_{12} = -1 + \frac{1}{2}(0.2689 \cdot 0 - 0.1192 \cdot 1) = -1.0596$$

$$w_{21} = -2 + \frac{1}{2}(0.7311 \cdot 0 - 0.8808 \cdot 0) = -2$$

$$w_{22} = 2 + \frac{1}{2}(0.2689 \cdot 0 - 0.1192 \cdot 0) = 2$$

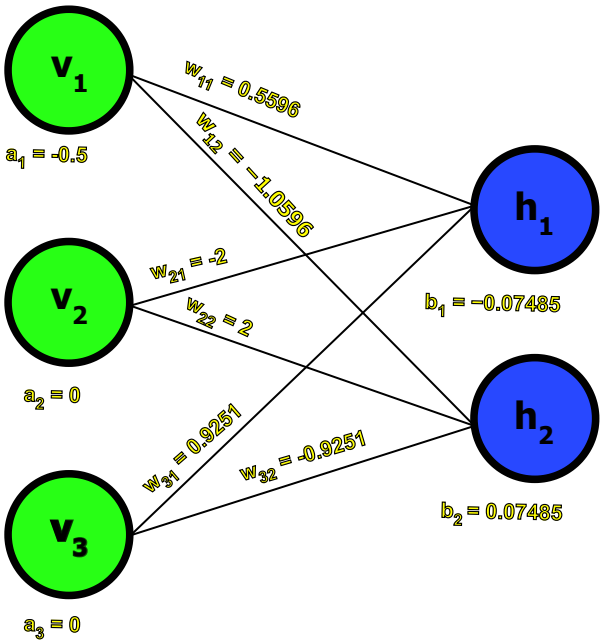
$$w_{31} = 1 + \frac{1}{2}(0.7311 \cdot 1 - 0.8808 \cdot 1) = 1$$

$$w_{32} = -1 + \frac{1}{2}(0.2689 \cdot 1 - 0.1192 \cdot 1) = -1$$

Таким образом, получаем:

$$a = \begin{pmatrix} -0.5 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} -0.07485 \\ 0.07485 \end{pmatrix}, W = \begin{pmatrix} 0.5596 & -1.0596 \\ -2 & 2 \\ 0.9251 & -0.9251 \end{pmatrix}$$

Обновление весов



Для удобства все вычисления выполнялись в среде MatLab. Ограниченная машина Больцмана обучилась на первом векторе входных данных, проведем обучение на остальных:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Рассмотрим второй столбец матрицы S:

$$v^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, a = \begin{pmatrix} -0.5 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} -0.07485 \\ 0.07485 \end{pmatrix}, W = \begin{pmatrix} 0.5596 & -1.0596 \\ -2 & 2 \\ 0.9251 & -0.9251 \end{pmatrix}$$

$$p(h^{(0)}|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \begin{pmatrix} 0.8037 \\ 0.1289 \end{pmatrix}, h^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$p(v^{(1)}|h^{(0)}) = \sigma(a + Wh^{(0)}) = \begin{pmatrix} 0.5149 \\ 0.1192 \\ 0.7161 \end{pmatrix}, v^{(1)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$p(h^{(1)}|v^{(1)}) = \sigma(b + W^\top v^{(1)}) = \begin{pmatrix} 0.8037 \\ 0.1289 \end{pmatrix}, h^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$p(v^{(2)}|h^{(1)}) = \sigma(a + Wh^{(1)}) = \begin{pmatrix} 0.5149 \\ 0.1192 \\ 0.7161 \end{pmatrix}, v^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Для обновления весов нам понадобятся следующие вектора:

$$p(h = 1|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \begin{pmatrix} 0.8037 \\ 0.1289 \end{pmatrix}$$

$$p(h = 1|v^{(2)}) = \sigma(b + W^\top v^{(2)}) = \begin{pmatrix} 0.8037 \\ 0.1289 \end{pmatrix}$$

Значения матрицы весов и векторов сдвигов после пересчета не изменились:

$$a = \begin{pmatrix} -0.5 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} -0.07485 \\ 0.07485 \end{pmatrix}, W = \begin{pmatrix} 0.5596 & -1.0596 \\ -2 & 2 \\ 0.9251 & -0.9251 \end{pmatrix}$$

Теперь рассмотрим последний столбец матрицы S:

$$v^{(0)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$p(h^{(0)}|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \begin{pmatrix} 0.1115 \\ 0.8884 \end{pmatrix}, h^{(0)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$p(v^{(1)}|h^{(0)}) = \sigma(a + Wh^{(0)}) = \begin{pmatrix} 0.1737 \\ 0.8808 \\ 0.2839 \end{pmatrix}, v^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$p(h^{(1)}|v^{(1)}) = \sigma(b + W^\top v^{(1)}) = \begin{pmatrix} 0.1116 \\ 0.8884 \end{pmatrix}, h^{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$p(v^{(2)}|h^{(1)}) = \sigma(a + Wh^{(1)}) = \begin{pmatrix} 0.1737 \\ 0.8808 \\ 0.2839 \end{pmatrix}, v^{(2)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Для обновления весов нам понадобятся следующие вектора:

$$p(h = 1|v^{(0)}) = \sigma(b + W^\top v^{(0)}) = \begin{pmatrix} 0.1115 \\ 0.8884 \end{pmatrix}$$

$$p(h = 1|v^{(2)}) = \sigma(b + W^\top v^{(2)}) = \begin{pmatrix} 0.1115 \\ 0.8884 \end{pmatrix}$$

Значения матрицы весов и векторов сдвигов после пересчета снова не изменились:

$$a = \begin{pmatrix} -0.5 \\ 0 \\ 0 \end{pmatrix}, b = \begin{pmatrix} -0.07485 \\ 0.07485 \end{pmatrix}, W = \begin{pmatrix} 0.5596 & -1.0596 \\ -2 & 2 \\ 0.9251 & -0.9251 \end{pmatrix}$$

Запуск обученной RBM на тестовых данных

Обучение RBM проходило на следующих данных:

Студент	Математический анализ	Линейная алгебра	Программирование
Андрей	0	0	1
Павел	1	0	1
Вадим	0	1	0

Пусть имеются тестовые данные следующего вида:

Студент	Математический анализ	Линейная алгебра	Программирование
Дарья	1	0	1
Анатолий	0	1	0
Алексей	1	1	0
Николай	0	0	1

Заметим, что почти все строки тестовой выборки совпадают с строками обучающей (кроме имени). Интересно посмотреть, как программа отработает, если ей подать данные, на которых она обучалась.

Теперь запустим обученную ограниченную машину Больцмана на предложенных тестовых данных (алгоритм остался тем же, только значения матрицы весов и векторов сдвигов больше не обновляются):

$$Test_{data} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, Predict_{data} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Из полученных предсказаний видно, что алгоритм ошибся в двух случаях из четырех. Но это не удивительно, поскольку:

> - Алгоритм обучался на малом количестве входных данных и всего лишь единственный раз (количество эпох равно единице).
> - Можно было бы поэксперементировать и поменять (увеличить) количество шагов метода CD-k и learning rate (попробовать взять его меньше).
> - Сэмплирование происходило не так, как должно быть на самом деле (в данном случае, если вероятность была больше 0.5, то состояние считалось равным 1; вместо этого необходимо использовать алгоритм сэмплирования вероятностей, рассмотренный ранее в одной из глав).

Скрипт на MATLAB

Все вычисления проводились в MATLAB, и был написан скрипт, который иллюстрирует работу ограниченной машины Больцмана на рассмотренном выше примере.

Он написан очень подробно, с множеством комментариев, чтобы по нему довольно легко можно было понять, как работает RBM. Его можно посмотреть здесь (или найти его в репозитории по следующему пути: **RBM/MATLAB/ExampleRBM.m**).

Кроме того, можно посмотреть на результат его работы для вышеописанного примера (если не хочется запускать сам скрипт). Его можно найти тут (в репозитории он находится по следующему пути: **RBM/MATLAB/ExampleRBM_output.txt**).

Спасибо за внимание!

Обратная связь: **frixinglife@gmail.com**