math_2020_082

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

Використання обмежених машин Больцмана при розпізнаванні зображень

Дипломна робота

Рівень вищої освіти - другий (магістерський)

Виконав (ла):
студент (ка) 6 курсу, групи 607
спеціальності 124 – Системний аналіз
(назва спеціальності)
Пустильник Наталія Олександрівна
(прізвище, ім'я та по-батькові)
Керівник к.ф.-м.н. Лукашів Т.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:

Протокол засідання кафедри № 7

від „ 15 ” грудня 2020 р.

зав. кафедри _____ проф. Черевко І.М.

Чернівці – 2020

Анотація

В роботі прореферовано теоретичний матеріал, що стосується алгоритма обмеженої машини Больцмана.

Використовуючи наведений теоретичний матеріал, засобами середовища R побудовано нейронну мережу, призначену для розв’язання задачі розпізнавання образів.

Зміст

Вступ

Обмежена машина Больцмана (Restricted Boltzmann Machine, RBM) – це штучна нейронна мережа, яка має лише 2 шари: один видимий і один прихований.

Перша обмежена машина Больцмана була побудована в 1986 році Полом Смоленськом під назвою Harmonium, але здобула популярність тільки після винаходу Хінтона – швидких алгоритмів навчання в середині 2000-х років.

Такої назви алгоритм набув через те, що він є модифікацією звичайної машини Больцмана, в якій нейрони розділили на видимі і приховані, а зв'язки допустимі тільки між нейронами різного типу, таким способом обмеживши зв'язки. Значно пізніше, в 2000-х роках, обмежені машини Больцмана набули неабиякої популярності і стали розглядатися вже не як варіації машини Больцмана, а як особливі компоненти в архітектурі мереж глибокого навчання. Об'єднання декількох каскадів обмежених машин Больцмана формує глибинну мережу переконань (Deep Belief Network, DBN), особливий вид багатошарових нейронних мереж, які можуть самонавчатися без вчителя за допомогою алгоритму зворотного поширення помилки.

Обмежені машини Больцмана мають широкий спектр застосувань – це задачі зниження розмірності даних, задачі класифікації, колаборативна фільтрація, виділення ознак і тематичне моделювання тощо.

В обмеженій машині Больцмана нейрони утворюють двочастковий граф (біграф), з одного боку графа знаходяться видимі нейрони (вхід), а з іншого боку – приховані, причому перехресні зв'язки встановлюються між кожним видимим і кожним прихованим нейроном. Така система зв’язків дозволяє застосувати при навчанні мережі метод градієнтного спуску з контрастивною дивергенцією (алгоритм порівняльної розбіжності).

Видимий шар – це входи (Рис. 1). Прихований шар зрештою стане інформацією про корисні функції, якщо навчання буде успішним.

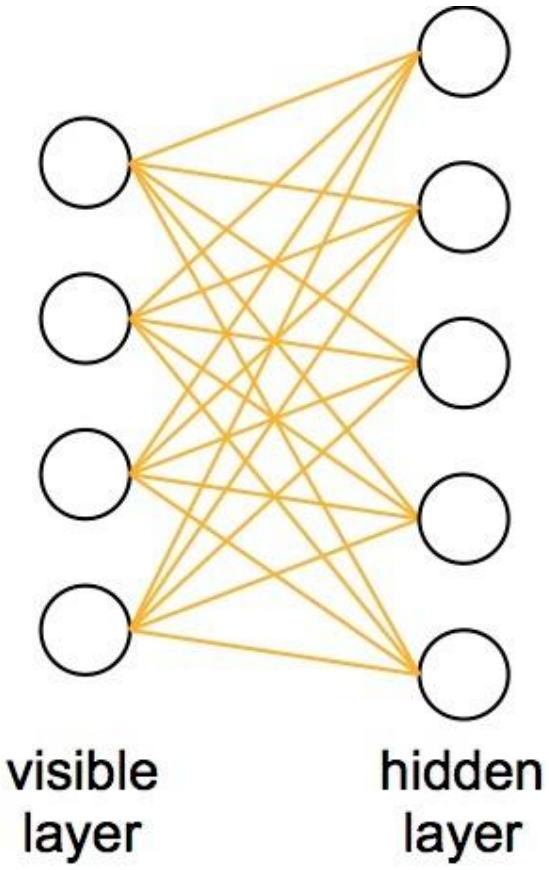


Рис. 1 Архітектура RBM.

Обмежена машина Больцмана, розроблена Смоленським у 1986 [1], є розширеною версією машини Больцмана, обмеженою одним принципом: не існує асоціацій ні між видимими вузлами, ні між прихованими вузлами. Знову ж таки DBN розглядається як імовірнісна генеративна модель, що складається з численних шарів стохастичних прихованих вузлів, що укладаються, елементами яких є RBM. Кожна RBM має два шари (Рис. 2).

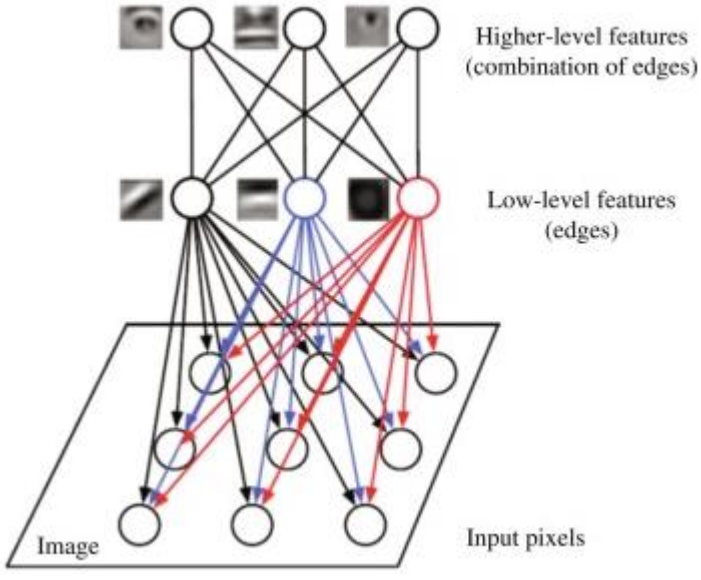


Рис. 2 Аналіз зображення в DBN.

Основний рівень складається з видимих вузлів, таких як спостережувані змінні, а наступний шар містить приховані концентратори – приховані змінні. DBN – це не просто укладання RBM. Тільки верхній шар має двонаправлені асоціації, тоді як центральні шари не мають.

При розгляді проблеми обробки зображень видимий шар може спочатку переглянути зображення та передати його на приховані шари для вивчення і відтворення низькорівневих функцій на прихованому шарі, а ненормальний стан включається при другому прихованому шарі в укладену RBM (Рис. 3).

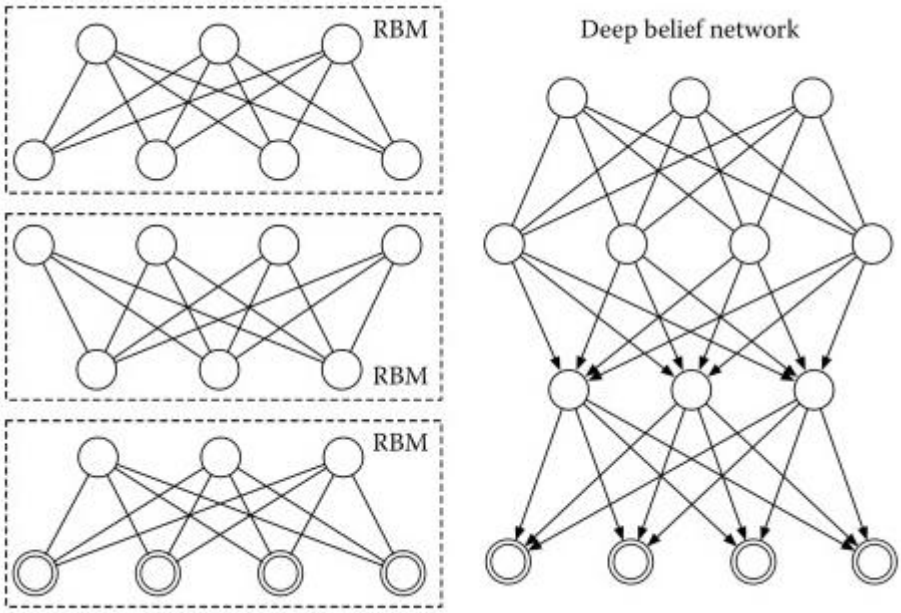


Рис. 3 Типова структура DBN з одним видимим та двома прихованими шарами RBM.

1 Основні означення і позначення

Машини Больцмана – це розподіли ймовірностей на двійкових векторах великих розмірів, які є аналогами випадкових полів Гаусса-Маркова, у тому сенсі, що вони повністю визначаються моментами першого та другого порядку. Однак, ключова відмінність полягає в тому, що розширення машин Больцмана додаванням прихованих змінних збільшує клас розподілів, які можна моделювати, так що в принципі можна моделювати розподіли довільної складності. З іншого боку, маргіналізація за прихованими змінними в нормальному розподілі дає ще один нормальний [2].

Навчання машин Больцмана все ще здається більше мистецтвом, ніж наукою, однак ЕМ-алгоритм розв'язує цю проблему досить ефективно для класу малозв'язаних машин Больцмана, що включає глибокі машини Больцмана [3]. Цей підхід забезпечує принципову основу для побудови складних машин Больцмана поступово і його можна прямо розширити для роботи з машинами Больцмана вищого порядку (наприклад, машини Больцмана, визначені моментами третього порядку).

Бінарна / гауссова відмінність не є взаємовиключною дихотомією: можуть бути побудовані гібридні моделі, що містять обидва типи змінних. Термін машина Гауса-Бернуллі-Больцмана, зазвичай, використовується для позначення моделі, в якій вектор видимих змінних v є неперервним, а вектор прихованих змінних h є двійковим. Для кожної двійкової конфігурації h умовний розподіл v при заданому h є гауссовим. Тому зображення

$$P(v) = \sum_h P(h)P(v|h)$$

показує, що цю модель можна інтерпретувати як гауссову суміш з великою кількістю компонентів. Цей тип моделі часто використовується для бінаризації неперервних даних (наприклад, можна було б пов'язати з кожним вектором даних v, максимальну апостеріорну оцінку h), щоб двійкові машини Больцмана можна було використовувати в подальшій обробці. У додатках для обробки мови така модель, як ця, може бути використана як універсальна фонова модель, і вона може вмістити вектори даних набагато вищого розміру, ніж традиційні гауссові сумішеві моделі. Зокрема, цепстральні коефіцієнти, отримані з 10-15 фреймів, можуть бути об'єднані та оброблені як простий вектор даних

Машини Больцмана можуть бути використані для реалізації як генеративного, так і дискримінаційного підходів розпізнавання зразків. Загалом, генеративні підходи найкраще працюють при розпізнаванні диктора, а дискримінаційні підходи найкраще при розпізнаванні мови. Сучасні системи розпізнавання ораторів використовують декілька типів генеративної моделі як екстрактори характеристик (універсальна фонова модель, аналіз спільного

фактора та екстрактор i-вектора) та як класифікатори (Ймовірнісний лінійний дискримінантний аналіз та метрика косинусної відстані).

У літературі по машинному навчанні, машини Больцмана в основному використовуються для неконтрольованого навчання іншого типу генеративної моделі, відомої як сигмоподібна мережа переконань або мережа глибоких переконань (Deep Belief Network, DBN). Приховані змінні в DBN можна розглядати як такі, що дають причинно-наслідкові пояснення даних, так що обчислення апостеріорних значень прихованих змінних у DBN може розглядатися як своєрідне отримання ознак високого рівня. Стандартне наближення, що використовується для апостеріорних обчислень в DBN, реалізовується нейронною мережею прямого поширення (де відсутній softmax верхнього рівня, який зазвичай використовується для прийняття рішень щодо розпізнавання). Припустимо, тоді, що DBN можна навчати без вчителя на певній сукупності даних таким чином, щоб оцінки ознак високого рівня були корисними для розпізнавання шаблонів даних. За таких обставин, було б доцільно використовувати апостеріорні обчислення в DBN як ініціалізацію для навчання оберненого розширення нейронної мережі з прямим зв'язком, призначеним для розрізнення цих шаблонів.

Ця стратегія успішно застосовується у розпізнаванні мови, особливо у Microsoft. Цікаво, що Microsoft недавно повідомили про дуже хороші результати розпізнавання мови, отримані за допомогою оберненого розширення без сторонньої допомоги в глибоких нейронних мережах.[4]

Ми використовуватимемо символ $P(\cdot)$ для позначення ймовірностей, розрахованих за даною моделлю, незалежно від того, чи можна ці ймовірності точно оцінити, чи ні. Як правило, ми використовуватимемо символ $Q(\cdot)$ для позначення апостеріорних ймовірностей прихованих змінних даних. Ці апостеріори можуть бути приблизними або точними, залежно від контексту.

h буде використовуватись для позначення прихованих векторних змінних, v для позначення видимих змінних, та x для позначення загальних змінних (у ситуаціях, коли немає необхідності розрізняти приховані та видимі змінні).

Відстань Кульбака-Лейблера між двома розподілами $P_1(x)$ і $P_2(x)$ позначається $D(P_1(x)||P_2(x))$ і задається як

$$D(P_1(x)||P_2(x)) = \sum_x P_1(x)\, ln \frac{P_1(x)}{P_2(x)}.$$

Вона володіє властивістю, що $D(P_1(x)||P_2(x)) > 0$ і $D(P_1(x)||P_2(x)) = 0$, коли P_1 і P_2 ідентичні. Дивергенція є асиметричною мірою різниці між двома розподілами.

Ентропія $P(x)$ позначається $H(P(x))$ і записується як

$$H\big(P(x)\big) = -\sum_{x} P(x) lnP(x).$$

І володіє властивістю $H(P(x)) > 0$. Крім того $H\big(P(x)\big) = 0$ тоді і тільки тоді, коли $P(x)$ концентрується на одному значенні x.

1.1 Доведення нерівності

Припустимо, у нас є модель із прихованими та видимими змінними. Частинний розподіл ймовірностей видимих змінних, задана у вигляді

$$P(v) = \sum_{h} P(v, h),$$

відома.

Проблема оцінки або апроксимації $P(v)$ виявляється еквівалентною задачі оцінки або апроксимації апостеріора $Q(h|v)$.

Визначено

$$L = E\left[ln\frac{P(v,h)}{Q(h|v)}\right],$$

де математичне сподівання обчислюється відносно $Q(h|v)$, яке може бути точним або приблизним апостеріором. Це можна записати у декількох різних формах, усі вони корисні:

$E\ [ln\ P(v,h)] + H(Q(h|v))$;

$$E\ [ln\ P(v|h)] - D\ (Q(h|v)\ ||\ P(h));$$

$$ln\ P(v) - D\ (Q(h|v)\ ||\ P(h|v)).$$

Перше випливає з визначення L, а друге – з першого (запишіть $P(v,h) = P(v|h)P(h)$). У третьому виразі ми використовуємо позначення $P(h|v)$, щоб вказати точний апостеріор h, задану через v, так що

$$L = E\left[ln\frac{P(v,h)}{Q(h|v)}\right] =$$

$$= E\left[ln\frac{P(h|v)P(v)}{Q(h|v)}\right] =$$

$$= lnP(v) + E\left[ln\frac{P(h|v)}{Q(h|v)}\right] =$$

$$= lnP(v) - D(Q(h|v)\big|\big|P(h|v)\big).$$

Властивість невід'ємності відстані Кульбака-Лейблера передбачає, що $L < lnP(v)$ і $L = lnP(v)$ тоді і тільки тоді, коли апостеріор є точним. Нерівність доведена.

Нижня межа іноді записується як

$$\langle lnP(v,h)\rangle_{data} + H$$

де "$data$" позначає (відповідає) апостеріорний розподіл $Q(h|v)$. (Аналогічно чином, $\langle \dots \rangle_{model}$ використовується для позначення математичного сподівання відповідно до апріорного розподілу $P(v,h)$.)

Величина $E[lnP(v,h)]$ відома як допоміжна функція ЕМ.

1.2 ЕМ-алгоритм

Якщо задано навчальний набір, що складається з набору видимих векторів $\{v^n\}$, нижня межа задається

$$L = \sum_n E_{Q(h|v^n)}\left[ln\frac{P(h,v^n)}{Q(h|v^n)}\right].$$

Оцінка цих математичного сподівання – це Е-крок алгоритму ЕМ.

Для того, щоб підігнати модель до даних, ми хочемо зробити L якомога більшим. L можна збільшити двома способами:

1) Зафіксуємо апостеріори $Q(h|v^n)$ і скоригуємо параметри моделі $P(v,h)$ так, щоб збільшити допоміжну функцію ЕМ

$$\sum_n \sum_h Q(h|v^n) lnP(h,v^n).$$

2) Зафіксуємо модель $P(v,h)$ і відкоригуємо апостеріор $Q(h|v^n)$ так, щоб зменшити відстань

$$\sum_n D(Q(h|v^n)||P(h,v^n)).$$

Записавши

$$lnP(v,h) = lnP(h) + lnP(v|h),$$

маємо 2 особливі випадки для 1), а саме: корегування $P(v|h)$ так, щоб максимізувач $\sum_n \sum_h Q(h|v^n) lnP(h,v^n)$ та корегування $P(h)$ так, щоб максимізувати $\sum_n \sum_h Q(h|v^n) lnP(h)$. Останнє еквівалентно мінімізації відстані

$$\sum_n D(Q(h|v^n)||P(h)),$$

яку називають оцінкою мінімальної дивергенції $P(h)$.

У випадку, коли $Q(h|v) = P(h|v)$, нижня межа є щільною ($lnP(v) = L$), тому чергування між 1) і 2) гарантовано збільшує оцінку максимальної правдоподібності навчального набору. Це класичний алгоритм ЕМ.

У загальному випадку, коли використовується приблизний варіаційний апостеріор, значення нижньої межі гарантовано збільшується, але це не гарантує збільшення оцінки максимальної правдоподібності. Це варіаційний ЕМ-алгоритм Байєса

1.3 ЕМ-алгоритм Байєса

У ситуаціях, коли апостеріорну оцінку важко знайти, ми можемо шукати наближення вигляду

$$Q(h|v) = \prod_i Q(h_i|v),$$

де фактори з правого боку вибираються таким чином, щоб максимізувати L. Зафіксуємо i і припустимо, ми хочемо оцінити $Q(h_i|v)$, припускаючи, що $Q(h_j v)$ зафіксовано для $j \neq i$. Позначивши $h = (h_i, h_{\setminus i})$,

$$L = E_{Q(h|v)}\left[ln\frac{P(v,h)}{Q(h|v)}\right] =$$

$$= E_{Q(h_i)}\left[E_{Q(h_{\setminus i})}\left[ln\frac{P(h_i, h_{\setminus i}, v)}{Q(h_i)Q(h_{\setminus i})}\right]\right] =$$

$$= E_{Q(h_i)}\left[E_{Q(h_{\setminus i})}\left[ln\frac{P(h_i, h_{\setminus i}, v)}{Q(h_i)}\right]\right] + const =$$

$$= E_{Q(h_i)}\left[ln\frac{\tilde{Q}(h_i)}{Q(h_i)}\right] + const =$$

$$= -D(Q(h_i)|\left|\tilde{Q}(h_i)\right),$$

де $\tilde{Q}(h_i)$ – розподіл, визначений

$$ln\tilde{Q}(h_i) = E_{Q(h_{\setminus i})}\left[lnP(h_i, h_{\setminus i}, v)\right] + const$$

Константа $const$ визначається з умови, що $\sum_{h_i}\tilde{Q}(h_i) = 1$. (Надалі використовуватимемо символ $\equiv$ для позначення рівності з точністю до константи). Таким чином, щоб максимізувати L, ми мінімізуємо відстань і покладемо $Q(h_i) = \tilde{Q}(h_i)$. Необхідно прокручувати фактори до збіжності. (Збіжність гарантована).

Може бути так, що важко зробити вибірку з апостеріорного розподілу $P(x)$, але легко і ефективно отримати вибірку з апостеріорного розподілу

$Q\ (x_i|x_{\setminus i})$. Наприклад, припустимо, що x має лише 2 компоненти. Ми генеруємо ланцюг Маркова

$$x^1 \to x^2 \to \cdots \to x^n \ ...$$

шляхом вибору випадкового значення x_2^0 та вибірки

$$x_1^1 \sim Q(x_1^1|x_2^0), x_2^1 \sim Q(x_2^1|x_1^1), x_1^2 \sim Q(x_1^2|x_2^1), x_2^2 \sim Q(x_2^2|x_1^2), ...$$

(Для кожного повного кроку $x^1 \to x^2, x^2 \to x^3, ...$ необхідні дві вибірки). Для достатньо великих n, x^n буде розподілено відповідно до $P(x)$. (Модельний розподіл – це рівноважний розподіл ланцюга Маркова).

Принцип прямо поширюється на апостеріорний розподіл. Таким чином вибірки можна проводити з апостеріорного розподілу $Q(h|v)$ за умови, що для кожного i легко проводити вибірки з $Q(h_i|\ h_{\setminus i}, v)$. Взагалі вибірка Гіббса набагато дорожча обчислювально, ніж варіаційний байєсівський підхід. Однак, вибірка Гіббса є точним методом, а варіаційний байєсівський підхід – ні.

2 Машина Больцмана

Машина Больцмана – це розподіл ймовірностей на двійкових векторах x виду

$$P(x) = \frac{1}{Z} e^{-E(x)},$$

де "енергетична функція" $E(x)$ має вигляд

$$E(x) = -\sum_{i<j} x_i w_{ij} x_j$$

сума, що поширюється на всі пари (i, j) така, що $i < j$. Нормалізуюча константа Z називається функцією розділення.

Як альтернативний варіант можна розглянути

$$E(x) = -\frac{1}{2}\sum_{i,j} x_i w_{ij} x_j.$$

Одиниці іноді називають нейронами, а ваги w_{ij} – синаптичними вагами. Матрицю ваг ми позначаємо W. Деякі з одиниць можна виділити як видимі, а інші – як приховані. Коли необхідно зробити це розмежування, ми будемо використовувати символ v_i для двійкової змінної, пов'язаної з видимою одиницею i, і символ h_j для двійкової змінної, пов'язаної з прихованою одиницею j. Матриця W вважається симетричною з нульовою діагоналлю (діагональні елементи повинні дорівнювати нулю, щоб запобігти взаємодії нейронів із самими собою). Такі доданки, як $\sum_j w_j x_j$, також можуть бути

включені, але загального виграшу немає, оскільки це еквівалентно прирівнюванню деяких компонентів x до 1. Ми виключимо ці доданки «упередженості», щоб зробити виведення простими.

Моделі, які ґрунтуються на енергії, відрізняються від розподілу ймовірностей, які визначаються спрямованими ациклічними графами (байєсівські мережі).

Зокрема, зазвичай буває непросто обчислити стохастичну суму і ми не будемо розглядати це питання для машин Больцмана [3] [5].

Енергетичну функцію можна розглядати як визначення «м'яких» обмежень на дані (області з низькою енергією, які іноді називають ярами в енергетичній поверхні та мають велику ймовірність), а додаткові обмеження можуть бути накладені введенням додаткових доданків до енергетичної функції. Але енергетична функція не передбачає простого рецепту вибірки з розподілу ймовірностей. (Потрібен певний тип методу Монте-Карло для ланцюга Маркова (MCMC), алгоритм вибірка Гіббса або алгоритм Метрополіса).

2.1 Вибірка Гіббса

Формули вибірки Гіббса показують, що машину Больцмана можна розглядати як стохастичну нейронну мережу.

$$Q\left(x_i = 1 \middle| x_{\setminus i}\right) = \frac{P(x_i = 1, x_{\setminus i})}{P\left(x_i = 0, x_{\setminus i}\right) + P(x_i = 1, x_{\setminus i})} =$$

$$= \frac{exp(\sum_{j \neq i} w_{ij} x_j)}{1 + exp(\sum_{j \neq i} w_{ij} x_j)} =$$

$$= \sigma\left(\sum_{j \neq i} w_{ij} x_j\right),$$

де σ - сигмоїдна (тобто S-подібна) нелінійність, визначена як

$$\sigma(u) = (1 + e^{-u})^{-1}.$$

Легко перевірити, що

$$\sigma(-u) + \sigma(u) = 1,$$

$$\sigma'(u) = \sigma(u)(1 - \sigma(u)).$$

Аргумент σ (а не значення, яке він повертає), а саме $\sum_{j \neq i} w_{ij} x_j$, згадується в [5] як активація блоку i, а σ називається функцією активації.

2.2 Наближення середнього поля

Для того, щоб перетворити машину Больцмана в детерміновану нейронну мережу, ми можемо застосувати варіаційний баєсівський алгоритм до апріорного розподілу $P(x)$, тобто обчислити варіаційне наближення

$$P(x) \approx \prod_i Q(x_i) = \prod_i \mu_i^{x_i,}$$

де $\mu_i = Q(x_i = 1)$. (У контексті машин Больцмана варіаційний байєсівський алгоритм, зазвичай, називають наближенням середнього поля.)

Варіаційні рівняння оновлення мають вигляд

$$lnQ(xi) \equiv E_{Q(x_{\setminus i})}\left[lnP\left(x_i, x_{\setminus i}\right)\right] =$$

$$= E_{Q(x_{\setminus i})}\left[\sum_{j \neq i} x_i w_{ij} x_j\right] =$$

$$= \begin{cases} \sum_{j \neq i} w_{ij}\mu_{ij} & if \quad x_i{=}1, \\ 0 & if \quad x_i{=}0. \end{cases}$$

Таким чином,

$$Q(x_i = 1) = \frac{\exp(\sum_{j \neq i} w_{ij}\, \mu_j)}{1 + \exp(\sum_{j \neq i} w_{ij}\, \mu_j)}$$

$$= \sigma\left(\sum\nolimits_{j \neq i} w_{ij}\, \mu_j\right).$$

Оскільки $\mu_i = Q(x_i = 1)$, то рівняння з фіксованою точкою є

$$\mu_i = \sum_{j \neq i} w_{ij}\, \mu_j.$$

Це те саме, що рівняння для вибірки Гіббса, за винятком того, що x замінюються їх середніми значеннями μ. Це пояснює термін "середнє поле" і показує, як розподіл Больцмана може бути апроксимований детермінованою нейронною мережею.

Зауважимо, що наближення середнього поля не є добрим наближенням до $P(x)$, оскільки воно розглядає одиниці як статистично незалежні, і апроксимований розподіл є унімодальним. Унімодальні наближення, як правило, мають сенс лише для апроксимації апостеріорів прихованих змінних, що отримується під час спостереження. Приховані змінні, зазвичай, розглядаються як пояснення спостереження; сказати, що задній розподіл прихованих змінних є унімодальним, просто означає, що існує лише одне пояснення спостережень. На

практиці варіаційний баєсівський алгоритм слід застосовувати лише до апостеріорного розподілу, а не до апріорного розподілу.

2.3 Марковська властивість

Найцікавіша ситуація виникає тоді, коли матриця W розріджена. Нехай $x_{\setminus\{i,j\}}$ позначає набір змінних, відмінних від x_i та x_j. Якщо задано $x_{\setminus\{i,j\}}$ і $w_{ij} = 0$, можна записати енергетичну функцію як

$$E(x) = ax_i + bx_j + c$$

де a, b і c залежать від $x_{\setminus\{i,j\}}$, але не від x_i або x_j. Таким чином, x_i та x_j умовно незалежні, якщо задано $x_{\setminus\{i,j\}}$, за умови, що $w_{ij} = 0$.

Машина Больцмана представлена графічною моделлю з неорієнтованими ребрами, що з'єднують усі пари вершин i, j, для яких w_{ij} не рівно нулю. Властивість умовної незалежності можна узагальнити наступним чином. Припустимо, нам дано розподіл одиниць на три підмножини A, B і C із тією властивістю, що між вершинами A і C немає ребра. Тоді x_A і x_C умовно незалежні, якщо x_B задано, де $x_A = \{x_i : i \in A\}$ і т.д. Це називається марковською властивістю: «майбутнє» (C) не залежить від «минулого» (A), якщо «теперішнє» (B) дано.

Отже, граф надає інформацію про те, як умовні розподіли, такі як $P(\cdot \mid x_B)$, розкладаються на фактори це відрізняється від випадку направлених графічних моделей, де топологія графа говорить вам про те, як факторизується апостеріорний розподіл.) На практиці, аналізуючи, неорієнтовану графічну модель, можна сказати чи вибірку Гіббса та варіаційний байєсівський алгоритм є реалізовні.

3 Обмежені машини Больцмана

У цьому випадку існує прихований шар і видимий шар без прихованих зв'язків між прихованими і прихованими, чи видимими та видимими вузлами. Вектори h, v мають розмірність $J \times 1$ і $I \times 1$, а W – розмірність $I \times J$. Енергетична функція має вигляд

$$E(v, h) = -v\,TWh.$$

За властивістю Маркова $Q(h|v)$ і $P(v|h)$ розкладаються на множники

$$Q(h|v) = \prod_j Q(h_j|v),$$

$$P(v|h) = \prod_i Q(v_i|h),$$

і з цього випливає, що кожен із факторів легко можна точно оцінити. Таким чином, немає необхідності у використанні варіаційного баєсівського алгоритма чи алгоритма Гіббса для побудови вибірки. Вибірку можна ефективно реалізувати чергування прихованого та видимого рівнів. Такий підхід відомий як блочна вибірка Гіббса.

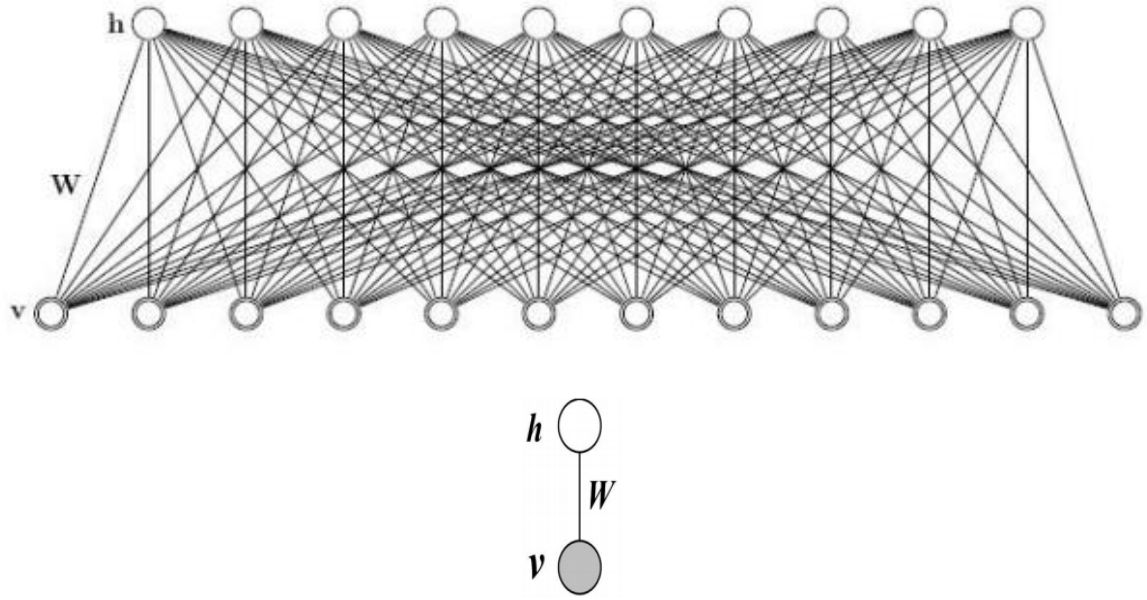


Рис. 4 Обмежена машина Больцмана

Маргінальний розподіл $P(v)$ не факторизується (якби це було так, то модель була б тривіальною), і за симетрією, те саме вірно, і для апостеріорного розподілу $P(h)$. Таким чином, приховані змінні не є незалежними в попередньому, на відміну від того, що може запропонувати досвід роботи з направленими графічними моделями. Пізніше ми побачимо, як оцінювати $P(v)$ та $P(h)$ точністю до функції розділення Z.

3.1 Гауссові випадкові поля Маркова

Неперервним аналогом розподілу Больцмана є багатовимірний розподіл Гауса з матрицею розрідженої точності (тобто оберненою коваріаційною матрицею) W, інакше відомою як гауссове марковське випадкове поле. Щоб розподіл можна було нормалізувати, матриця точності повинна бути додатно визначеною, тому в цьому випадку діагональні елементи W повинні бути строго додатними. З іншого боку, у бінарному випадку ми припускали, що діагональні елементи дорівнюють нулю. Як і машини Больцмана, гауссові марковські випадкові поля нелегко реалізувати у дуже великих розмірах. Для того, щоб оцінити статистичну суму, слід визначити визначник матриці точності, і для вибірки з розподілу (не вдаючись до вибірки Гіббса) матрицю точності потрібно діагоналізувати. Що стосується оцінки максимальної правдоподібності, якщо обмеження розрідженості не накладаються, правильною процедурою буде

оцінка матриці коваріації, прирівнюючи матрицю коваріації моделі та матрицю коваріації навчальних даних

$$\langle x_i x_j \rangle_{data} = \langle x_i x_j \rangle_{model},$$

а потім інвертувати матрицю коваріації, щоб отримати матрицю точності (для простоти ми припустили, що середній вектор дорівнює нулю). Однак накладання обмежень розрідженості на матрицю точності не є простим (хоча накладання обмежень розрідженості на матрицю коваріації не представляє труднощів).

3.2 Роль прихованих одиниць

Як машини Больцмана, так і гауссові марковські випадкові поля можуть моделювати лише статистику даних другого порядку. Мабуть, найцікавіша різниця між ними полягає в тому, що включення прихованих змінних у машини Больцмана розширює клас розподілів, які можна моделювати, але це не так для гауссово-марковських випадкових полів. Якщо $P(v,h)$ є гауссовим марковським випадковим полем, то маргінальний розподіл $P(v)$ є просто гауссовим. З іншого боку, якщо $P(v,h)$ – розподіл Больцмана з енергетичною функцією $E(v,h)$, ми можемо представити маргінальний розподіл $P(v)$ як енергетичну модель із енергетичною функцією $F(v)$, тобто

$$P(v) = \frac{1}{Z} e^{-F(v)},$$

визначаючи $F(v)$ як

$$F(v) = -ln \sum_h e^{-E(v,h)}$$

Зрозуміло, що вільна енергія не може бути представлена як квадратична форма за v, тому введення прихованих змінних розширює клас розподілів, які можуть моделювати машини Больцмана. Здається, додаючи достатньо багато прихованих змінних, можна апроксимувати будь-який розподіл за допомогою дискретних двійкових векторів [2]. Конкретний приклад розподілу, який можна змоделювати за допомогою машини Больцмана з прихованими змінними, але не за допомогою машини Больцмана, див. [6].

3.3 Навчання контрастної дивергенції

Машини Больцмана та суміжні моделі традиційно навчаються за допомогою стохастичного градієнтного підйому, а не в пакетному режимі, як це зазвичай відбувається при обробці мови.

Спочатку обчислимо градієнт логарифму функції максимальної правдоподібності відносно параметрів моделі, враховуючи маркер навчання x^1.

$$\frac{\partial ln P(x)}{\partial w_{ij}} \Big|_{x=x^1} = -\frac{\partial F(x)}{\partial w_{ij}} \Big|_{x=x^1} - \frac{\partial ln Z}{\partial w_{ij}} =$$

$$= x_i^1 x_j^1 - \frac{\partial}{\partial w_{ij}} ln \sum_x e^{-F(x)\frac{\partial F(x)}{\partial w_{ij}}} =$$

$$= x_i^1 x_j^1 - \frac{1}{Z} \sum_x e^{-F(x)}\, x_i x_j =$$

$$= x_i^1 x_j^1 - \sum_x P(x)\, x_i x_j =$$

$$= x_i^1 x_j^1 - \langle x_i x_j \rangle_{model}$$

Якщо є N навчальних маркерів $x^1 \dots x^N$, то градієнт логарифму функції максимальної правдоподібності становить

$$\frac{\partial ln P(x^1 \dots x^N)}{\partial w_{ij}} = x_i^1 x_j^1 + \cdots + x_i^N x_j^N - N\langle x_i x_j \rangle_{model},$$

І отже,

$$\frac{1}{N} \frac{\partial ln P(x^1 \dots x^N)}{\partial w_{ij}} = \langle x_i x_j \rangle_{data} - \langle x_i x_j \rangle_{model,}$$

що дорівнює нулю в критичній точці функції максимальної правдоподібності, точно так само, як у випадку гауссово марковського випадкового поля. Ми будемо позначати на кореляції $\langle x_i x_j \rangle_{data}$ та $\langle x_i x_j \rangle_{model}$ як статистику даних та статистику моделі.

На практиці навчання здійснюється послідовно, а модель оновлюється після представлення кожного маркера навчання або міні-партії. Зокрема, якщо представлений навчальний маркер x^1, модель оновлюється відповідно до

$$w_{ij} \leftarrow w_{ij} + \alpha \frac{\partial ln P(x)}{\partial w_{ij}} \Big|_{x=x^1},$$

де α - рівень навчання. Статистика моделі $\langle x_i x_j \rangle_{data}$, як правило, оцінюється за допомогою контрастної дивергенції (CD) [7], яка в загальній формулюванні Бенджо та Деллале (Y. Bengio and O. Delalleau) може бути застосована за допомогою будь-якого алгоритму МСМС для моделювання моделі (а не лише вибірки Гіббса). Починаючи із заданого навчального маркера x^1, запустимо ланцюг Маркова для n кроків: $x^1 \to \cdots \to x^{n+1}$. Якщо n достатньо велике, то x^{n+1} буде приблизно розподілено відповідно до розподілу моделі $P(x)$, тобто

$$E\left[x_i^{n+1} x_j^{n+1}\right] \approx \langle x_i x_j \rangle_{model},$$

і ми можемо наблизити

$$\frac{\partial lnP(x)}{\partial w_{ij}} \Big|_{x=x^1} \approx x_i^1 x_j^1 - x_i^{n+1} x_j^{n+1}.$$

Це наближення відоме як $CD-n$. На диво $CD-1$ добре працює на практиці. $CD-1$ діє так, що пригнічує енергетичну поверхню у реальній точці x^1 і збільшує енергію біля сусідньої фіктивної точки x^2. Можна стверджувати, що чистий ефект полягає у тісній підгонці енергетичної поверхні до реальних точок даних.

3.4 Постійна контрастна дивергенція.

Набір вибірок (або "частинок") x^1 ... x^N, отриманий із розподілу моделі, підтримується та оновлюється щоразу, коли модель оновлюється: N ланцюгів Маркова виконуються паралельно, і при кожному оновленні в кожному ланцюгу виконується кілька кроків вибірки Гіббса. Невелика швидкість навчання гарантує, що вибірки завжди беруться з розподілу моделі, хоча модель постійно оновлюється. Статистика моделі отримується шляхом усереднення по частинках:

$$\langle x_i x_j \rangle_{model} = \frac{1}{N} \sum_x x_i^n x_j^n.$$

Постійна CD, як правило, працює краще, ніж $CD-1$.

3.5 Тренування МБ із прихованими одиницями

Для навчання машин Больцмана із прихованими блоками ми використовуємо EM-алгоритм. Враховуючи вектор даних v, ми прагнемо максимізувати

$$\langle lnP(x) \rangle_{data}$$

відносно параметрів моделі, де $x = (v, h)$ та $\langle . \rangle_{data}$ відноситься до математичного моделювання, обчисленого з постеріором h, за заданим v. Ми можемо ігнорувати член ентропії в нижній межі, оскільки він не залежить від параметрів моделі. Диференціюючи за w_{ij}, одержимо

$$\frac{\partial}{\partial w_{ij}} \langle lnP(x) \rangle_{data} = \langle x_i x_j \rangle_{data} - \langle x_i x_j \rangle_{model}$$

Той факт, що не всі одиниці видно, не має різниці в способі обробки статистики моделі. І єдина відмінність полягає в обробці статистики даних. У випадку обмеженої машини Больцмана постеріор $Q(h|v)$ може бути точно оцінений, і обчислення статистичних даних є простим. У загальному випадку можна використовувати або вибірку Гіббса, або варіаційний байєсівський алгоритм для обчислення апостеріорного матсподівання необхідного для оцінки статистичних даних. На практиці використовується варіативний байєсівський

алгоритм, оскільки вибірка Гіббса є занадто повільною. Це алгоритм Салаххутдінова для навчання машин Больцмана [3].

Зауваження. Варіаційні байєсівські алгоритми можуть бути обчислювально дорогими, оскільки для досягнення збіжності може знадобитися багато разів перебирати всі змінні. Стандартна хитрість, яка полегшує цю проблему, полягає в ініціалізації варіаційних оновлень Байєса для даного маркера навчання за допомогою варіаційного наближення, обчисленого останнього разу, коли маркер був досягнутий під час навчання.

3.6 Розрахунки апостеріорної та вільної енергії для обмежених машин Больцмана

Повертаємось до обмеженої машини Больцмана, зображеної на Рис. 1. Припускаємо, що є I видимих одиниць, J прихованих одиниць і жодних прихованих або прихованих або видимих зв'язків. Таким чином, h має розмірність $J \times 1$, v має розмірність $I \times 1$, і ми можемо записати функцію енергії у вигляді

$$E(v,h) = -v^T W h,$$

де W має розмірність $I \times J$. І-й рядок W позначимо через W_i. ·, а j-й стовпець – через $W_{\cdot j}$.

Нагадаємо, що вільна енергія визначається як

$$F(v) = -ln \sum_h \exp\big(-E(v,h)\big).$$

Щоб оцінити це

$$\sum_h \exp(-E(v,h)) = \sum_h exp(v^T W\ h) =$$

$$= \sum_{h_1 \dots h_J} \exp\left(\sum_{j=1}^{J} v^T W_{\cdot j}\ h_j\right) =$$

$$= \sum_{h_1 \dots h_J} \prod_{j=1}^{J} \exp(\ v^T W_{\cdot j}\ h_j) =$$

$$= \prod_{j=1}^{J} \sum_{h_j \in \{0,1\}} \exp(\ v^T W_{\cdot j}\ h_j) =$$

$$\prod_{j=1}^{J}(1+\exp(\,v^T W_{\cdot j})) =$$

Що стосується апостеріора , то

$$Q(h|v) = \frac{\prod_{j=1}^{J}\exp(\,v^T W_{\cdot j}\,h_j)}{\prod_{j=1}^{J}(1+\exp(\,v^T W_{\cdot j}))} =$$

$$= \prod_{j=1}^{J}\frac{\exp(\,v^T W_{\cdot j}\,h_j)}{(1+\exp(\,v^T W_{\cdot j}))} =$$

$$= \prod_{j=1}^{J} Q\left(h_j|v\right) =,$$

де

$$Q\left(h_j = 1|v\right) = \sigma\left(v^T W_{\cdot j}\right)$$

Аналогічно

$$P(v_i = 1|h) = \sigma\left(W_{\cdot j}h\right).$$

3.7 Малозв’язані машини Больцмана

Ми використовуємо термін шар для позначення набору одиниць, жоден з яких не зв’язаний іншим, тобто відповідна підматриця вагової матриці дорівнює нулю. Ми говоримо, що машина Больцмана є слабо зв'язаною (або просто розрідженою), якщо блоки можна розділити на невелику кількість шарів. Позначимо змінні, що відповідають шарам, через h^0 ... h^L, де h^0 відповідає видимому

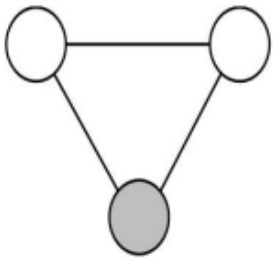


Рис. 5 Розріджена машина Больцмана з одним видимим та двома прихованими шарами

Матриця ваг має виглядає

$$\begin{pmatrix} \mathbf{0} & & & \\ & \mathbf{0} & & \\ & & \ddots & \\ & & & \mathbf{0} \end{pmatrix}$$

Існує нульова матриця для кожного шару, і немає обмежень щодо недіагональних блоків. У випадку обмеженої машини Больцмана є лише 2 блоки. У випадку глибокої машини Больцмана [2], шари укладаються так, що кожен шар спілкується лише зі своїми сусідами зверху і знизу. У цій ситуації провідні супердіагональні та субдіагональні блоки не дорівнюють нулю, а все інше дорівнює 0:

$$\begin{pmatrix} \mathbf{0} & * & & & \\ * & \mathbf{0} & & & \\ & & \ddots & & \\ & & & \mathbf{0} & * \\ & & & * & \mathbf{0} \end{pmatrix}.$$

Якщо W^{kl} позначає матрицю ваг на ребрах, що приєднують одиниці в шарі l до тих, що знаходяться в шарі k, то функція енергії визначається як

$$E(h) = -\sum_{k<l} h^{kT} W^{kl} h^{l},$$

де k та l змінюються від 0 до L.

Для будь-якої пари шарів k, l спільний умовний розподіл $P(h^k, h^l, h^{\setminus k,l})$ є обмеженою машиною Больцмана (матриця ваги, яка є модифікованою версією W^{kl}); спільного безумовного розподілу немає. Фактично,

$$E(h) = \sum_{h^{\setminus k,l}} P(h^k, h^l, h^{\setminus k,l})$$

показує, що спільний безумовний розподіл можна розглядати як машину Больцмана із прихованими змінними, і ми знаємо, що клас машин Больцмана з прихованими змінними є ширшим, ніж клас машин Больцмана. Зокрема, з цього випливає, що у загальному випадку

$$\langle h_i^k h_j^l \rangle_{model} \neq \langle h_i^k h_j^l \rangle_{RBM},$$

де RBM позначає обмежену машини Больцмана, визначену матрицею W^{kl}. Під час тренування розрідженої машини Больцмана може виникнути спокуса оцінити вагові матриці W^{kl} індивідуально, використовуючи тренування RBM, але це не буде коректною процедурою.

Однак розріджені машини Больцмана легше тренувати, ніж загальні машини Больцмана, оскільки вибірки Гіббса та EM-алгоритм Байєса можуть бути реалізовані дуже ефективно, рухаючись між шарами (замість того, щоб рухатись між окремими одиницями), точно так само, як вибірки Гіббса можна проводити в RBM, чергуючи прихований і видимий шар. Зауважимо, що для кожного шару l апостеріор $Q\ (h^l|\ h^{\setminus l})$ є факторіальним, оскільки

$$lnQ(h^l|\ h^{\setminus l}) = lnP(h) - lnP(\ h^{\setminus l}) =$$

$$= \sum_{h^{\setminus l}} E(h^l|\ h^{\setminus l}) =$$

$$= a^T h^l,$$

де a залежить від $h^{\setminus l}$, але не від h^l. Таким чином, ми можемо записати

$$Q(h^l|\ h^{\setminus l}) = \prod_i Q(h^l|\ h^{\setminus l})$$

і реалізувати вибірку Гіббса на шарі l шляхом вибірки різних одиниць незалежно одна від одної. Аналогічно для EM-алгоритм Байєса, припускаючи, що варіаційні апостеріори розкладаються на шари, тобто

$$Q(h|\ h^0) = \prod_i Q(h^l|h^0),$$

достатньо для забезпечення повну факторизацію. Це пояснюється тим, що формула варіаційного оновлення для шару l є

$$Q(h^l|\ h^0) \equiv E_{h^{\setminus l}}[lnP(h)] \equiv$$

$$\equiv E_{h^{\setminus l}}[-E(h)] \equiv$$

$$= \sum_{k<l} h^{kT} W^{kl} h^l =$$

$$= a^T h^l,$$

де а залежить від $h^{\setminus l}$, але не від h^l. Таким чином, ми можемо реалізувати EM-алгоритм Байєса на рівні l, оновлюючи варіаційні апостеріори для різних одиниць самостійно.

Варіаційну нижню межу також легко оцінити, якщо відома функція розділення.

Покладемо

$$\mu_1^l = Q(h_1^l = 1|h^0)$$

так, що сподівання h^l, розраховане за варіаційним апостеріором дорівнює

$$\langle h^l \rangle = \mu^l.$$

Зауважимо, що якщо $k \neq l$, то h^k і h^l незалежні у варіаційному апостеріорному розподілі і

$$\langle h^l h^{kT} \rangle = \langle h^l \rangle \langle h^k \rangle^T =$$
$$= \mu^l \mu^{kT}.$$

Таким чином,

$$\langle h^{kT} W^{kl} h^l \rangle = \langle tr(W^{kl} h^l h^{kT}) \rangle =$$
$$= Sp(W^{kl} \langle h^l h^{kT} \rangle) =$$
$$= Sp(W^{kl} \mu^l \mu^{kT}) =$$
$$= \mu^{kT} W^{kl} \mu^l,$$

і

$$\mathcal{L} = \langle lnP(h) \rangle + H =$$
$$= -\langle E(h) \rangle - lnZ + H =$$
$$= \sum_{k<l} \mu^{kT} W^{kl} \mu^l - lnZ - \sum_l \sum_i \left(\mu_1^l ln \mu_1^l + \left(1 - \mu_1^l\right) \ln\left(1 - \mu_1^l\right) \right).$$

Якщо функція розділення невідома, цей розрахунок дає нижню межу вільної енергії і цього достатньо для багатьох цілей.

3.8 RBM Гаусса-Бернуллі

Видимі вектори вважаються дійсними, приховані вектори двійковими, а функція енергії задана у вигляді

$$E(v, h) = \frac{1}{2}(v - b)^T (v - b) - c^T h - v^T W h \quad (v \in \mathbb{R}^I, h \in \{0,1\}^j)$$

Якщо h фіксована, то це є квадратичною формулою за v і

$$lnP(v|\, h) \equiv -\frac{1}{2} v^T v + v^T (b + Wh)$$

Порівнюючи це з квадратичною формою, яка визначає гауссіан із середнім значенням μ та коваріацією Σ, а саме:

$$-\frac{1}{2}(v - \mu)^T \Sigma^{-1} (v - \mu) = -\frac{1}{2} v^T \Sigma^{-1} v + v^T \Sigma^{-1} \mu - \frac{1}{2} \mu^T \Sigma^{-1} \mu,$$

то отримаємо розподіл $P(v|h)$ є гауссовим із середнім значенням $b + Wh$ та одиничною коваріаційною матрицею. Що стосується $P(h|v)$, то

$$lnP(h|\, v) \equiv (c^T + v^T W)h =$$

$$= \sum_j a_j h_j,$$

де $a_j = c_j + v^T W_{.j}$ так, що

$$Q(h|\, v) = \prod_j Q\big(h_j \big| v\big),$$

$$Q\big(h_j = 1 \big| v\big) = \frac{e^{a_j}}{1 + e^{a_j}} =$$

$$= \sigma\big(a_j\big) =$$

$$= \sigma(c_j + v^T W_{.j})$$

Маргінальний розподіл P(h), тобто

$$\int P(v|\, h)dv$$

неможливо оцінити явно, але якщо ми запишимо

$$P(v) = \sum_h P(h)P(v|h),$$

то ми зможемо інтерпретувати $P(v)$ як гауссову суміш з великою кількістю компонентів, де ваги суміші обчислюються складно, і кожен компонент має одиничну матрицю як матрицю коваріації. Звичайно, припущення про загальну одиничну коваріаційну матрицю є нерозумним, якщо дані не були належним чином попередньо оброблені (наприклад, очищуючи їх так, щоб середнє значення даних дорівнювало нулю, а загальна матриця коваріації була одиничною. Моделювання діагональної загальної коваріаційної матриці для всіх компонентів суміші, відмінної від одиничної не є надто складним [8].

Однак загальний випадок, коли існує повна матриця коваріації або матриця повної точності для кожного компонента суміші h, є складнішим. Підхід у роботі [9] передбачає, що матриця точності, пов'язана із компонентом суміші h, лінійно залежить від h (аналогічно середньому вектору). Всі матриці точності моделюються за допомогою набору матриць точності першого рангу. Для даної складової суміші h, відповідна матриця точності є зваженою лінійною комбінацією матриць точності рангу 1, де ваги лінійні за h.

Універсальна фонова модель для мовлення. При обробці мовлення, а особливо при розпізнаванні дикторів, термін «універсальна фонова модель» (Universal Background Model, UBM) використовується для позначення великої моделі Гаусса (із сотнями або тисячами компонентів суміші), яка зазвичай

навчається на сотнях годин даних, і яка призначена охоплювати можливі джерела мінливості мовного сигналу. Використовуючи машини Больцмана для розпізнавання мови або розпізнавання дикторів, першим кроком є підготовка RBM Гаусса-Бернуллі щодо векторів акустичних характеристик (тобто цепстральних коефіцієнтів), які, як ми вище пояснили, можна розглядати як модель гауссової суміші з астрономічною кількістю компонентів суміші.

Роль UBM полягає у поданні вектора акустичних особливостей v як "індексу компонентної суміші" h. У найпростішій реалізації ідея тут полягає в тому, що вектор даних v може бути зіставлений з двійковим вектором h, який максимізує $Q(h|v)$. На практиці, як правило, використовується наближення середнього поля, так що замість двійкового вектора отримується вектор ймовірностей Бернуллі (тобто ймовірностей $Q(h_j = 1|v)$), а не двійковий вектор. Це подання підходить для подальшої обробки, що включає варіаційні обчислення Байєса за допомогою двійкових машин Больцмана – шляхом обчислення апостеріора $Q(h|v)$. Таким чином, акустичні спостереження представляються у вигляді двійкових векторів для всієї подальшої обробки. Ключовим моментом є те, що це подання здатне працювати з векторами ознак відносно великих розмірів (кілька сотень), завдяки чому інформація може бути вилучена з довгих вікон (наприклад, цілих фонем).

У роботі [10] вектори акустичних особливостей виділяються наступним чином. Кожні 10 мс звичайно отримують вектор кепстральних коефіцієнтів і обчислюють першу та другу похідні. Вектори, виділені через вікно з 11 кадрами, об'єднуються, утворюючи вектор спостереження. Ці вектори нормуються так, що дисперсія кожного компонента дорівнює 1. Цікаво, що вектори не є попередньо очищеними. Оскільки похідні лінійно залежать від цепстральних коефіцієнтів, ефект попереднього очищення полягав би у видаленні першої та другої похідних, чого слід уникати, оскільки ці похідні компенсують припущення про діагональну коваріацію в суміші. Ефект припущення про діагональ полягає в тому, щоб розглядати послідовні кадри як статистично незалежні. При обробці мови добре відомо, що якщо немає механізму для моделювання цих залежностей, то включення похідних є корисним.

Таким чином, дизайн в [10] та багатьох наступних роботах є досить штучним, оскільки він значною мірою призначений для компенсації неадекватності RBM Гаусса-Бернуллі. Ці недоліки усуваються при побудові середньої коваріації RBM [9]. При роботі з цим типом RBM похідні не включаються в інтерфейс, а дані попередньо очищуються (для зменшення їх розмірності).

4 Практична частина

4.1 Тестовий набір даних MNIST

База даних MNIST (Mixed National Institute of Standards and Technology) – об'ємна база даних зразків рукописного написання цифр. Є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрування і зіставлення методів розпізнавання зображень за допомогою машинного навчання, в першу чергу на основі штучних нейронних мереж [11]. База даних містить 60 000 зображень для навчання і 10 000 зображень для тестування.

Дані складаються з заздалегідь підготовлених прикладів зображень, отриманих з рукописних документів шляхом обробки чорно-білих зразків символів NIST розміром 20×20 пікселів. Творці бази даних NIST, в свою чергу, використовували набір зразків з Бюро перепису населення США, до якого були додані ще тестові зразки, написані студентами американських університетів. Зразки з набору з NIST були нормалізовані, пройшли згладжування та приведення до напівтонового зображенню розміром 28×28 пікселів.

Рис. 6 Зразки зображень з тестового набору даних MNIST

4.2 Алгоритм на мові R

Набір даних MNIST представляє собою рукописний цифровий набір даних, який завантажимо з Kaggle [12]. Дані представлені у форматі списку з навчальним набором, ярликами навчального набором, тестовим набором та ярликами тестового набору.

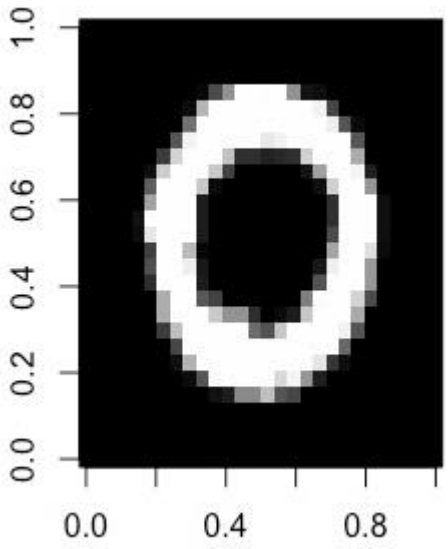
Завантажимо дані:

```
# Завантажуємо дані MNIST
load("MNIST.RData")
```

Перевіримо, чи виглядають дані такими, як вони повинні виглядати, виведемо їх на екран:

```
# Перевіряємо дані з набору train set
image(matrix(MNIST$trainX[2, ], nrow = 28), col =
grey(seq(0, 1, length = 256)))
```

Якщо дані завантажені правильно, це повинно виглядати приблизно так:



Спочатку почнемо з навчання RBM і бережемо її як модель:

```
# Спочатку отримуємо дані з train data від MNIST
train <- MNIST$trainX
# Підганяємо модель
modelRBM <- RBM(x = train, n.iter = 1000, n.hidden = 100,
size.minibatch = 10)
```

Для навчання RBM нам необхідні функції train data, які повинні бути матрицею форми (samples * features), інші параметри за замовчуванням.

Кількість ітерацій визначає кількість епох навчання, в кожну епоху RBM() буде вимагати вибір нового «міні-пакета». Коли у нас буде достатньо даних, встановимо велику кількість ітерацій, так як це поліпшить нашу модель; зворотною стороною є те, що навчання цієї функції займе більше часу. Аргумент n.hidden визначає, скільки прихованих вузлів матиме RBM, а size.minibatch – це кількість навчальних вибірок, які будуть в кожній епоху.

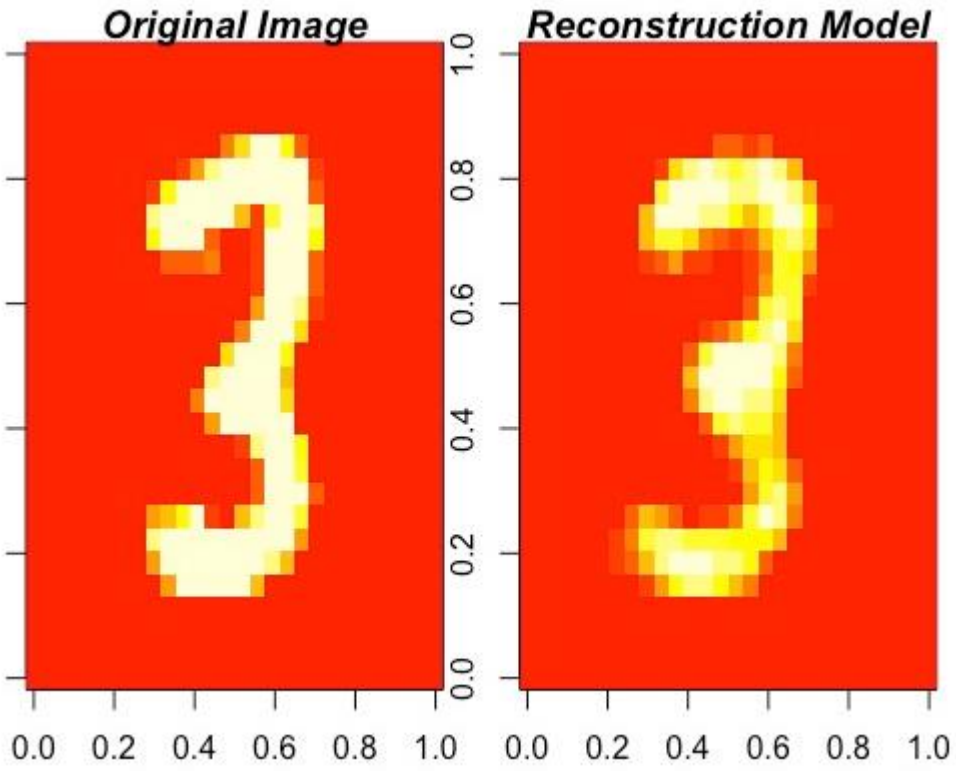
Використаємо plot, щоб побачити, що вивчають приховані вузли:

```
# Використовуємо plot, щоб побачити, як навчаються приховані вузли
RBM(x = train, plot = TRUE, n.iter = 1000, n.hidden = 30, size.minibatch = 10)
```

Після навчання моделі RBM ми можемо перевірити, наскільки добре вона відновлює дані за допомогою функції ReconstructRBM() :

```
# Отримуємо дані test data від MNIST
test <- MNIST$testX
# Відновлення зображення за допомогою modelRBM
ReconstructRBM(test = test[6, ], model = modelRBM)
```

Потім функція виведе вихідне зображення поруч з відновленим зображенням. Якщо модель хороша, відтворене зображення повинно бути схоже або навіть краще за оригінал:



Висновок

В даній роботі було опрацьовано теоретичний матеріал, що стосується алгоритма обмеженної машини Больцмана.

Дане дослідження також демонструє використання та застосування на практиці алгоритма обмеженої машини Больцмана при аналізі зображень, за допомогою засобів середовища R.

Винайдена Джеффрі Хінтоном, обмежена машина Больцмана – це алгоритм, який можна використовувати для зменшення розмірності, класифікації, регресії, спільної фільтрації, вивчення особливостей та моделювання тем.

Вони складаються лише з двох шарів вузлів, а саме – прихованого шару із прихованими вузлами та видимого шару, що складається з вузлів, які представляють дані. У більшості додатків видимий шар представлений двійковими одиницями; на прикладі нашого алгоритму ми використовуємо значення піксельних зображень нормованих даних MNIST.

Хоча RBM належать до сім'ї машин Больцмана, вони не такі, як машини Больцмана. Вони є обмеженою версією машини Больцмана, оскільки вузли в шарі можуть мати з'єднання лише з вузлами в іншому шарі. Ця, здавалося б, невеликий нюанс насправді відіграє велику роль. Через відсутність зв'язків між вузлами всередині прихованого та видимого шарів, RBM є повним дводольним графом. Повні дводольні графи володіють однією важливою властивістю: вузли всередині шару є умовно незалежними з огляду на інший шар. Це робить RBM набагато простішим та ефективнішим у навчанні, ніж оригінальна машина Больцмана.

Використовуючи наведений теоретичний матеріал, засобами середовища R побудовано нейронну мережу, призначену для розв'язання задачі розпізнавання образів.

Література

1. Smolensky P "Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory" 1986. pp. 194–281.
2. Bengio Y. “Learning deep architectures for AI,” Foundations and Trends in Machine Learning, vol. 2, no. 1, 2009. pp. 1–127.
3. Salakhutdinov R., Hinton G. “An efficient learning procedure for deep boltzmann machines,” in CSAIL Technical report, MIT-CSAIL-TR-2010-37, Aug. 2010.
4. Seide F., Li G., Yu D. “Conversational speech transcription using context-dependent deep neural networks,” in Proc. Eurospeech 2011, 2011.
5. Bishop C. Pattern Recognition and Machine Learning. New York, NY: Springer Science+Business Media, LLC, 2006.
6. MacKay D., Information theory, inference and learning algorithms. New York, NY: Cambridge University Press, 2003.
7. Bengio Y., Delalleau O., “Justifying and generalizing contrastive divergence,” Neural Computation, vol. 21, no. 6, 2009. pp. 1601–1621.
8. Cho K., Ilin A., Raiko T. “Improved learning of Gaussian-Bernoulli restricted Boltzmann machines,” in Master’s Thesis, Aalto University, 2011.
9. Dahl G., Hinton G. “Phone recognition with the mean-covariance restricted boltzmann machine,” in Advances in Neural Information Processing 23, 2010.
10. Mohammed A., Dahl G., Hinton G. “Acoustic modeling using deep belief networks,” IEEE Trans. on Audio, Speech, and Language Processing, 2010.
11. LeCun Y., Cortes C., Burges. C. MNIST handwritten digit database, Офіційний сайт URL: http://yann.lecun.com/exdb/mnist/ (дата останнього звернення: 10.10.2020).
12. Kaggle – Набір даних MNIST URL: https://www.kaggle.com/c/mnist-digits/data (дата останнього звернення: 21.10.2020).

Додаток

```
setwd("D:/Documents/Університет/МАГІСТЕРСЬКА/RBM")

# Завантажуємо дані MNIST

load("MNIST.RData")

# Перше встановлення devtools

#install.packages("devtools", dependencies=TRUE)

# Завантажуємо devtools

#library(devtools)

library(keras)


# Перевіряємо дані з набору train set
image(matrix(MNIST$trainX[2, ], nrow = 28), col = grey(seq(0, 1, length =
256)))


VisToHid <- function(vis, weights, y, y.weights) {

  # Функція для обчислення прихованого шару.

   V0 <- vis

  if ( is.null(dim(V0))) {

    # Якщо visual - це вектор, створюємо матрицю

    V0 <- matrix(V0, nrow= length(V0))

  }

  if(missing(y) & missing(y.weights)) {

    # Обчислюємо прихований шар за допомогою trained weights та bias

    H <- 1/(1 + exp(-( V0 %*% weights)))

  } else {

    Y0 <- y

    H <- 1/(1 + exp(- ( V0 %*% weights + Y0 %*% y.weights)))

  }

  return(H)

}


# Функція відновлення даних із прихованого шару

HidToVis <- function(inv, weights, y.weights) {
```

```
  # Функція відновлення видимого шару
  if(missing(y.weights)) {
    # Реконструюємо лише видимий шар, коли відсутні y.weights
    V <- 1/(1 + exp(-(   inv %*% t(weights)) ))
    return(V)
  } else {
    # Реконструюємо видимий шар та мітки
    Y <- 1/(1 + exp(-( inv %*% t(y.weights))))
    return(Y)
  }
}

# Логістична функція
logistic <- function(x) {
  1/(1+exp(-x))
}

# Функція для обчислення енергії RBM
Energy <- Energy <- function(vis, inv, weights, y, y.weights) {
  # Функція для розрахунку енергії тренованої RBM
  if(!missing(y) & !missing(y.weights)){
    E <- -(vis %*% weights %*% t(inv)) - (y %*% y.weights %*% t(inv))
  } else {
    # Обчислення ф. енергії якщо вона не контролюється
    E <- -(vis %*% weights %*% t(inv))
  }
    return(E)

}

# Функція для створення контрастної дивергенції CD
CD <- function(vis, weights, y, y.weights) {
```

```
  # Функція, коли k = 1 CD
    # Початок позитивної фази
  if (missing(y) & missing(y.weights)) {
    # Обчислення прихованого шару
    H0 <- VisToHid(vis, weights)
    H0[,1] <- 1
  } else {
    H0 <- VisToHid(vis, weights, y, y.weights)
    H0[,1] <- 1
  }
  # Бінарізуємо прихований шар:
  unif  <- runif(nrow(H0) * (ncol(H0)))
  H0.states <- H0 > matrix(unif, nrow=nrow(H0), ncol= ncol(H0))

  # Обчислюючи позитивну фазу, використовуємо для цього ймовірності
  pos.phase <- t(vis) %*% H0
  if (!missing(y)) {
    pos.phase.y <- t(y) %*% H0
  }
  # Початок негативної фази
  # Реконструюємо видимий шар
  V1 <- HidToVis(H0.states, weights)
  # Встановіть одиницю зміщення на 1 (bias)
  V1[,1] <- 1

  if (missing(y) & missing(y.weights) ) {
    #  Реконструюмо  прихований  шар  без  нагляду,  більше  не  потрібно
виправляти зміщення
H1 <- VisToHid(V1, weights)
  } else {
     Y1 <- HidToVis(H0, weights,  y.weights )
    # Встановлюмо одиницю зміщення на 1
```

```
    Y1[,1] <- 1
    # Реконструюємо прихований шар під контролем, більше не потрібно виправляти упередження
H1 <- VisToHid(V1, weights, Y1, y.weights)
  }
  # Обчислюючи негативні асоціації, використовуємо для цього ймовірності:
  neg.phase <- t(V1) %*% H1
  if (!missing(y) & !missing(y.weights)) {
    # Calculate negative phase y
    neg.phase.y <- t(Y1) %*% H1
  }
  ## Обчислюємо градієнти (для вагів)
    grad.weights <- pos.phase - neg.phase

  if (!missing(y) & !missing(y.weights)) {
    # Обчислюємо градієнти для y.weigths
    grad.y.weights <- pos.phase.y - neg.phase.y

    return(list('grad.weights' = grad.weights,'grad.y.weights' = grad.y.weights))
  } else {
     return(list('grad.weights' = grad.weights  ))
  }
}

# Функція для бінаризації даних міток
LabelBinarizer <- function(labels) {

  # Ініціалізуємо матрицю, щоб зберегти вектори міток:
  y <- matrix(0, length(labels), length(unique(labels)))
  for (i in 1:length(labels)) {
        y[i, labels[i] + 1] <- 1
```

```
  }
  return(y)
}

# Ініціалізуємо функцію RBM
RBM <- function (x, y, n.iter = 100, n.hidden = 30, learning.rate = 0.1,
                 plot = FALSE, size.minibatch = 10, momentum = 0.5, lambda
= 0.001) {

  if (plot == TRUE) {
    if ((n.iter %% 10) == 0) {

      plot.epoch <- n.iter/10
    } else {

      print ('Number of iterations was not devicable by ten: plots are
turned off')
      plot <- FALSE
      plot.epoch <- 0
    }
  } else {

    plot.epoch <- FALSE
  }

  if (!is.matrix(x)) {
    print('Data was not in a matrix, converted data to a matrix')
    x <- as.matrix(x)
  }
  if (any(!is.numeric(x))) {
    stop('Sorry the data has non-numeric values, the function is
terminated')
  }
```

```
  if (n.iter > 10000) {
    print("Number of epochs for > 10000, could take a while to fit")
  }
  if (!missing(y)) {
    if (any(!is.numeric(y))) {
      stop('Sorry the labels have non-numeric values, the function is terminated')
    }
    if (any(!is.finite(y))) {
      stop('Sorry this function cannot handle NAs or non-finite label values')
    }
    if (length(y) != nrow(x)) {
      stop('Labels and data should be equal for supervised RBM: try training an unsupervised RBM')
    }
  }
  if (any(!is.finite(x))) {
    stop('Sorry this function cannot handle NAs or non-finite data')
  }
  if (size.minibatch > 100) {
    print('Sorry the size of the minibatch is too long: resetting to 10')
    size.minibatch <- 10
  }
  if (size.minibatch > 20) {
    print('Large minibatch size, could take a long time to fit model')
  }
  if (min(x) < 0 | max(x) > 1) {
    stop('Sorry the data is out of bounds, should be between 0 and 1')
  }
  if( length(dim(x)) < 2 ) {
    stop("Dimensions of the data were not right, should be of shape n.features * n.samples")
```

```
  }
  if(ncol(x) > nrow(x)) {
    print('Less data than features, this will probably result in a bad model fit')
  }

  # Ініціалізуємо ваги, n.features * n.hidden зі значеннями з гауссового розподілу
  weights <- matrix(rnorm(ncol(x) * n.hidden, 0, .01), nrow = ncol(x), ncol = n.hidden)
  # Ініціалізуємо матрицю momentum_speed
  momentum_speed_x <- matrix(0, nrow = ncol(x) + 1, ncol = n.hidden + 1)

  # Додаємо зміщення вагам
  weights <- cbind(0, weights)
  weights <- rbind(0, weights)

  # Додайте 1 для зміщення до x
  x <- cbind(1, x)

    if (!missing(y)) {
    labels <- unique(y)
     idx <- vector('list', length = length(labels))
    # Зберегаємо індекси
    for (i in 1:length(labels)) {
      idx[[i]]<- which(y == labels[i])
    }
    # Складаємо бінаризовані вектори міток
    y <- LabelBinarizer(y)
    # Додаємо один термін для упередженості
    y <- cbind(1, y)

    # Створюємо матрицю у ваг
```

```
    y.weights <- matrix(rnorm(length(labels) * n.hidden, 0, 01), nrow = length(labels), ncol = n.hidden)
    # Додаємо матрицю швидкості імпульсу
    momentum_speed_y <- matrix(0, nrow = length(labels) + 1, ncol = n.hidden + 1)

    # додаємо зміщення вагам
    y.weights <- cbind(0, y.weights)
    y.weights <- rbind(0, y.weights)

  }

  if(plot == TRUE){

    par(mfrow = c(3,10), mar = c(3,1,1,1))
    plot.weights <- weights[-1, -1]
    if (n.hidden > 30) {

      print('n.hidden > 30, only plotting a sample of the invisible nodes')

      samp.plot <- sample(1:n.hidden, 30)
            for(i in samp.plot) {
        # Виводимо ваги
        image(matrix(plot.weights[, i], nrow = sqrt(ncol(x)-1)), col=grey.colors(255))
        title(main = paste0('Hidden node ', i), font.main = 4)
        # Ініціалізуємо лічильник для побудови графіку
        plot.counter <- 0
      }
    } else {
      for(i in 1:n.hidden) {
        # Виводимо ваги
```

```
        image(matrix(plot.weights[,   i],   nrow   =   sqrt(ncol(x)-1)),
col=grey.colors(255))
        title(main = paste0('Hidden node ', i), font.main = 4)
        # Ініціалізуємо лічильник для побудови графіку
        plot.counter <- 0
      }
    }
  }
  plot.counter <- 0
  # Починаємо CD, k = 1
  for (i in 1:n.iter){
    if (missing(y)) {
      # Зразок minibatch від x, без нагляду
      samp <- sample(1:nrow(x), size.minibatch, replace = TRUE)
    } else {
      # Підбираємо збалансовані ярлики
      samp <- rep(0,size.minibatch)
      p <- 1
      for (i in 1 : size.minibatch){
        samp[p]<- sample(idx[[p]], 1)
        p <- p + 1
        if (p == length(labels) +1) {
          # Скидаємо лічильник
          p <- 1
        }
      }
    }
    plot.counter <- plot.counter + 1
    # На ітерації встановлюємо видимий шар до випадкової вибірки train:
    V0 <- x[samp, ,drop = FALSE]
    if (missing(y)) {
      # Обчислюємо градієнт
```

```
      grads <- CD(V0, weights)
    } else {
      # Обчислюємо градієнт
      grads <- CD(V0, weights, y[samp,,drop = FALSE], y.weights)
    }

    momentum_speed_x <- momentum * momentum_speed_x + ((grads$grad.weights
- (lambda * weights))/ size.minibatch)

    # Оновлюємо ваги та зміщення
    weights <- weights + (learning.rate * momentum_speed_x)

    if (!missing(y)) {
      momentum_speed_y     <-     momentum     *     momentum_speed_y     +
((grads$grad.y.weights - (lambda * y.weights))/ size.minibatch)


      # Оновлюємо ваги та зміщення
      y.weights <- y.weights + (learning.rate * momentum_speed_y)
    }
    # Графічне вивчення прихованих вузлів plot.epoch:
    if(plot.counter == plot.epoch & plot == TRUE) {
      par(mfrow = c(3,10), mar = c(3,1,1,1))
      plot.weights <- weights[-1, -1]
      if (n.hidden > 30) {
        for(i in samp.plot) {
          image(matrix(plot.weights[,   i],   nrow   =   sqrt(ncol(x)-1)),
col=grey.colors(255))
          title(main = paste0('Hidden node ', i), font.main = 4)
        }
      } else {
        for(i in 1:n.hidden) {
```

```
          image(matrix(plot.weights[,   i],   nrow   =   sqrt(ncol(x)-1)),
col=grey.colors(255))
          title(main = paste0('Hidden node ', i), font.main = 4)
        }
      }
      # Скиньте лічильник:
      plot.counter <- 0
    }
  }
  # повертаємо список із матрицями тренованих ваг та умовами зміщення
  if (!missing(y)) {
    return(list('trained.weights'    =    weights,'trained.y.weights'    =
y.weights))
  } else {
    return(list('trained.weights' = weights))
  }
}

# Відновлюємо зображення за допомогою modelRBM
ReconstructRBM <- function(test, model, layers = 1) {
  if (!missing(layers)) {
    if ( layers == 1) {
      print('Layers is 1, will treat the model as a regular RBM.
            If the model is a stacked RBM only the first layer weights
will be used')
    }
    }
  if (!is.null(dim(test))) {
    stop('It is only possible te reconstruct one training image at a time')
  }
  if (any(!is.numeric(test))) {
    stop('Sorry the data has non-numeric values, the function is executed')
  }
```

```
  if (any(!is.finite(test))) {
    stop('Sorry this function cannot handle NAs or non-finite data')
  }

  if (length(model)  != layers) {
    stop("Number of layers is unequal to the number of weight matrices in
the model")
  }

    test <- matrix(test, nrow = 1)



  V <- cbind(1, test[1,, drop = FALSE])

  # Реконструюємо зображення
  if (missing(layers)) {
    # Обчислюємо прихований шар
    H <- VisToHid(V, model$trained.weights)
    # Реконструюємо видимий шар
    V.rec <- HidToVis(H, model$trained.weights)
  } else {

    for (i in 1:layers) {
      V <- VisToHid(V, model[[i]]$trained.weights)
    }

    H <- V

    for (i in layers:1) {
      H <- HidToVis(H, model[[i]]$trained.weights)
    }
```

```
    V.rec <- H

    V <- cbind(1, test[1,, drop = FALSE])
  }



  par(mfrow = c(1,2))
  image(matrix(V[, -1], nrow = sqrt(ncol(test))))
  title(main = 'Original Image', font.main = 4)
  image(matrix(V.rec[, -1], nrow = sqrt(ncol(test))))
  title(main = 'Reconstruction Model', font.main = 4)
  }

# Використовуємо функцію RBM()
# Спочатку отримайте дані train data від MNIST
train <- MNIST$trainX
# Підганяємо модель
modelRBM <- RBM(x = train, n.iter = 1000, n.hidden = 100, size.minibatch
= 10)

# Використовуємо plot, щоб побачити, як навчаються приховані вузли
RBM(x = train, plot = TRUE, n.iter = 1000, n.hidden = 30, size.minibatch
= 10)

# Отримуємо дані test data від MNIST
test <- MNIST$testX
# Відновлення зображення за допомогою modelRBM
ReconstructRBM(test = test[6, ], model = modelRBM)
```