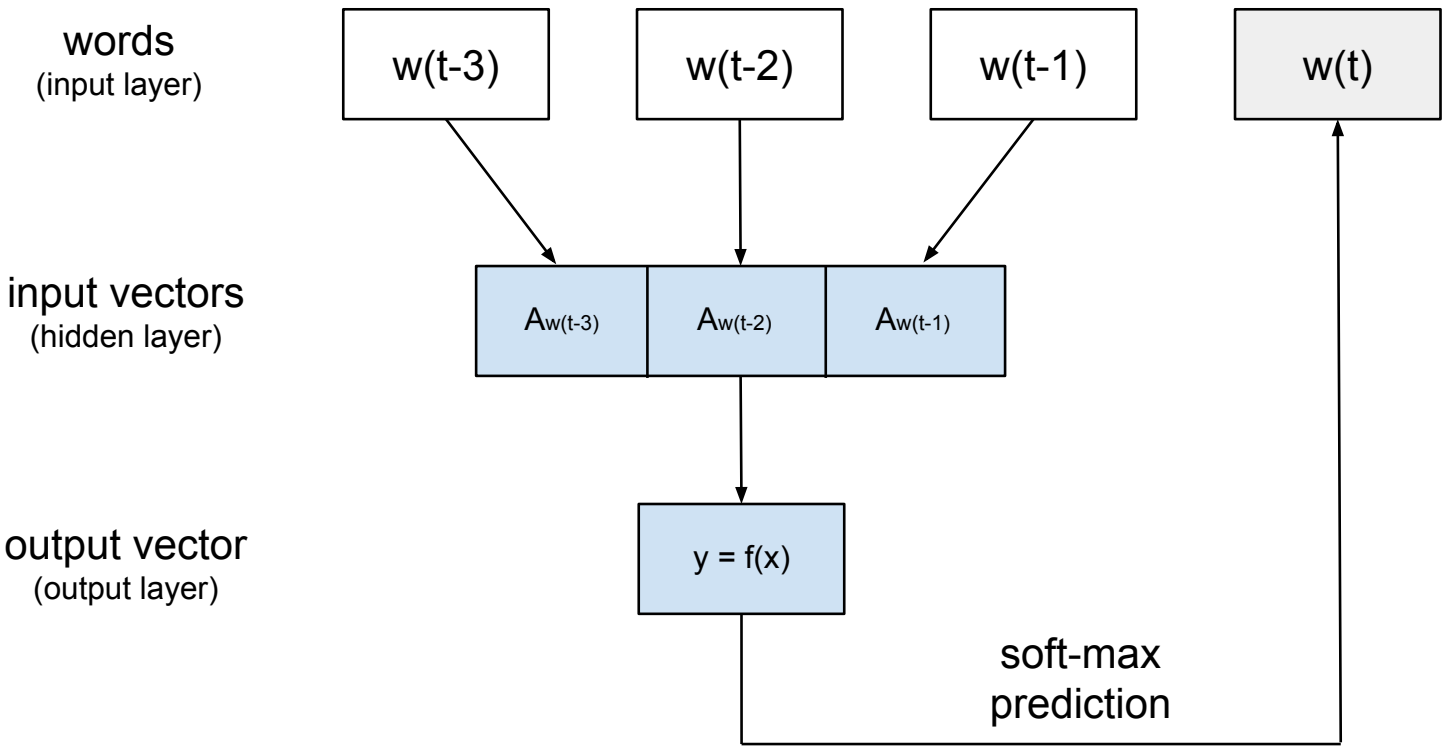
Построение векторных представлений для многозначных слов

Сергей Бартунов

группа Байесовских Методов

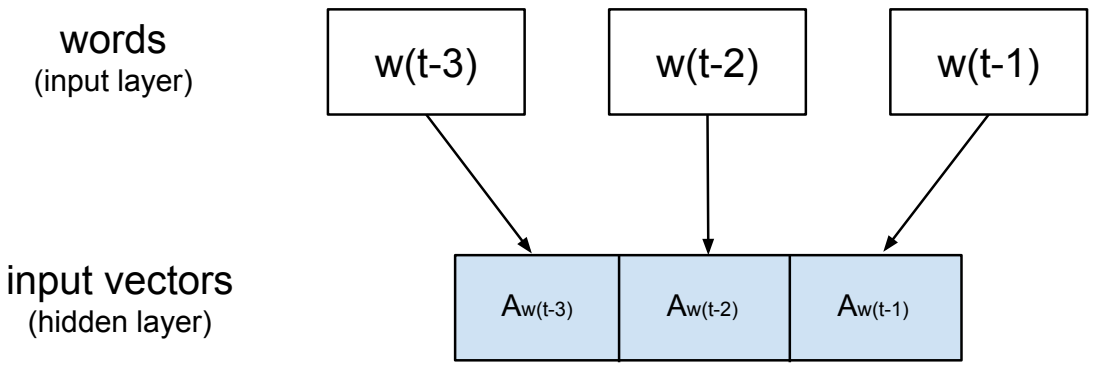
24 декабря 2014 г.

Neural language model



$$p(\mathbf{w}) = \prod_t p(w_t | w_{t-1}, w_{t-2}, w_{t-3})$$

Input → hidden

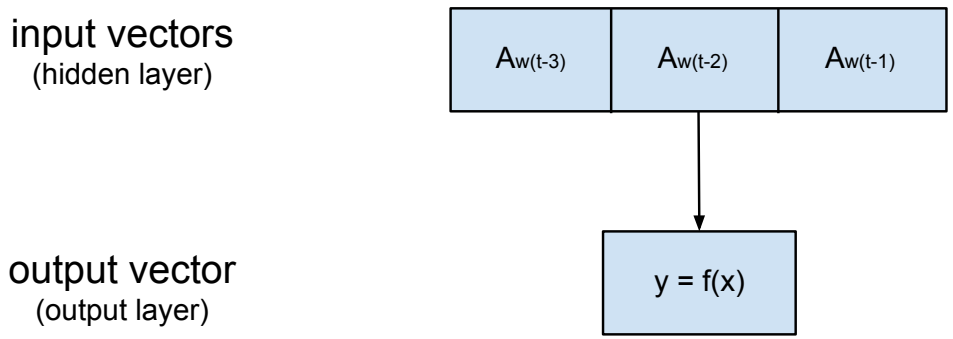


$A \in \mathbb{R}^{D \times |V|}$ - матрица скрытых представлений слов

Проекция на скрытый слой:

$$\mathbf{x} = \begin{pmatrix} Aw_{t-3} \\ Aw_{t-2} \\ Aw_{t-1} \end{pmatrix}$$

Hidden → output

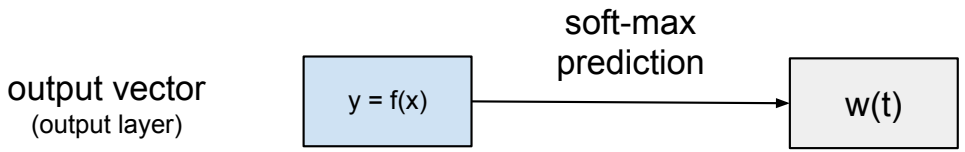


- $\mathbf{y} = f(\mathbf{x})$ - что угодно, например, перцептрон:

$$f(\mathbf{x}) = \sigma(W\mathbf{x} + b)$$

- Самая вычислительно дорогая операция в модели
  - Умножение матрицы на вектор
  - Нелинейные зависимости

Output → prediction

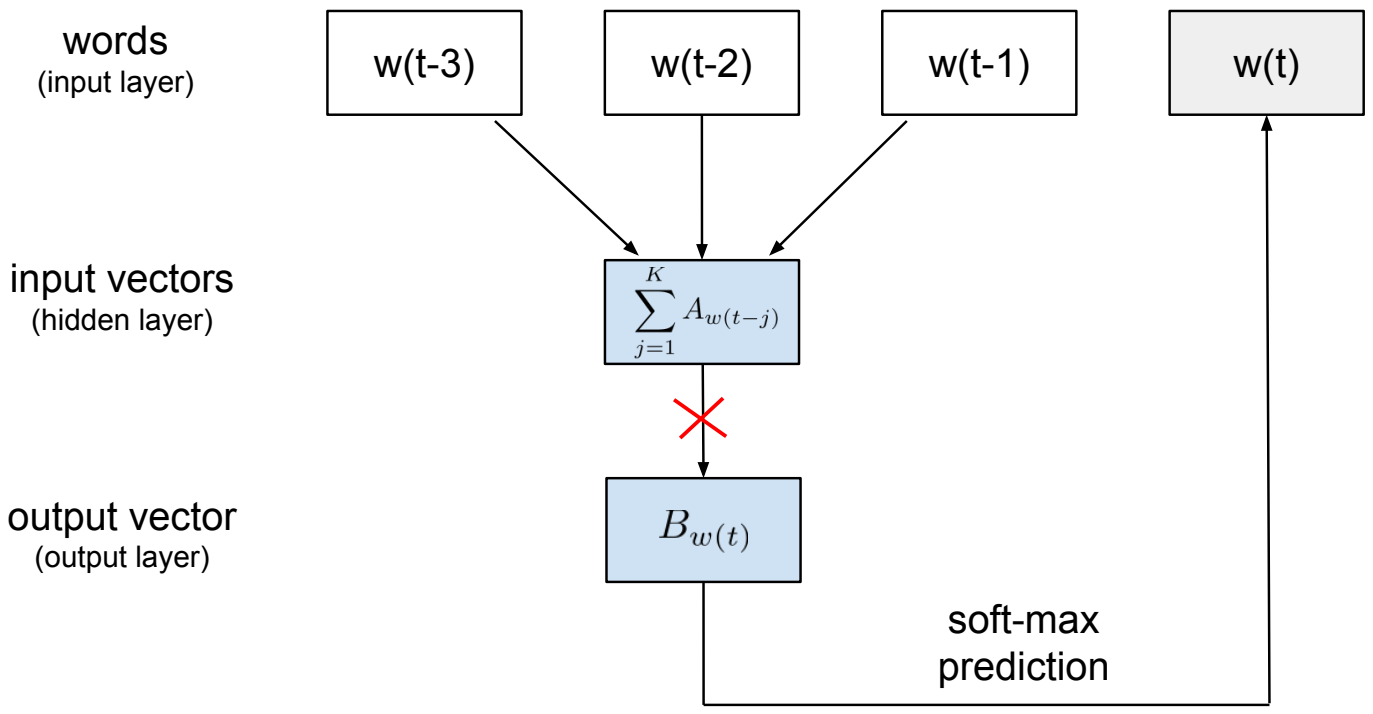


$$p(w_t = v|\mathbf{y}) = \frac{\exp(y_v)}{\sum_{v'=1}^{|V|} \exp(y_{v'})}$$

- Линейная сложность от размера словаря

[Bengio et al.(2003)Bengio, Ducharme, Vincent, and Janvin]

Continuous bag of words model



Continuous bag of words (CBOW)

- Больше нет зависимости между входным и выходным слоями!
  - вместо этого есть выходные представления B_w для каждого слова
- Слово предсказывается по контексту

$$p(v|\mathbf{w}) = \frac{\exp(B_v^T(\sum_j A_{w_j}))}{\sum_s \exp(B_s^T(\sum_j A_{w_j}))}$$

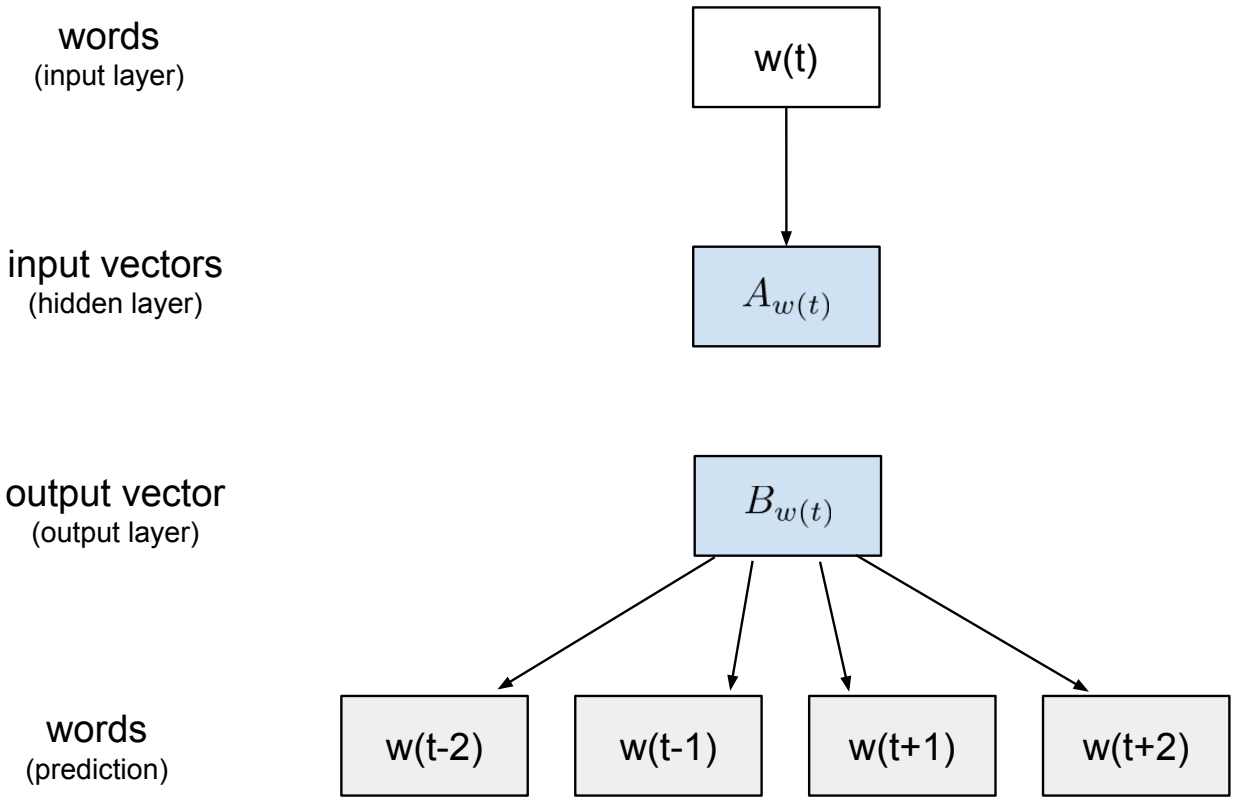
- Входные и выходные представления обучаются в ходе оптимизации целевой функции:

$$\mathcal{F}(A, B) = \sum_t \log p(w_t|w_{t-1}, \cdots, w_{t-K})$$

  - Мы больше не обязаны строить языковую модель и можем учитывать также "будущие" слова $w_{t+1}, w_{t+2}, \ldots$

[Mikolov et al.(2013a)Mikolov, Chen, Corrado, and Dean]

Skip-gram (SG)


words
(input layer)
w(t)
input vectors
(hidden layer)
$A_{w(t)}$
output vector
(output layer)
$B_{w(t)}$
words
(prediction)
w(t-2)
w(t-1)
w(t+1)
w(t+2)

Skip-gram model

- Вывернутая “наизнанку” модель CBOW
  - из слова предсказывается его контекст
  - предсказания слов выполняются независимо

$$p(v|w) = \frac{\exp(B_v^T A_w)}{\sum_s \exp(B_s^T A_w)}$$

- Предложена вместе с CBOW, позднее была придумана сверхбыстрая процедура обучения
- Входные и выходные представления обучаются в ходе оптимизации целевой функции:

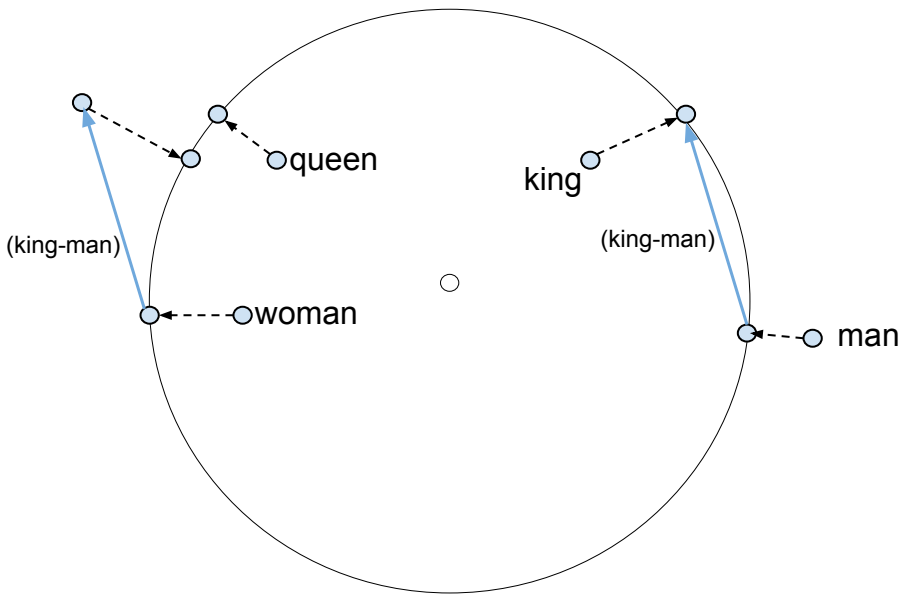
$$\mathcal{F}(A, B) = \sum_t \sum_{j \in c(t)} \log p(w_j|w_t)$$

[Mikolov et al.(2013b)Mikolov, Sutskever, Chen, Corrado, and Dean]

Latent semantic space

- Learned word vectors form the rich latent semantic space
- Super-feature: "linguistic regularities"

$$\arg\max_{w} cos(w, \text{king} - \text{man} + \text{woman}) = \text{queen}$$



Hierarchical soft-max

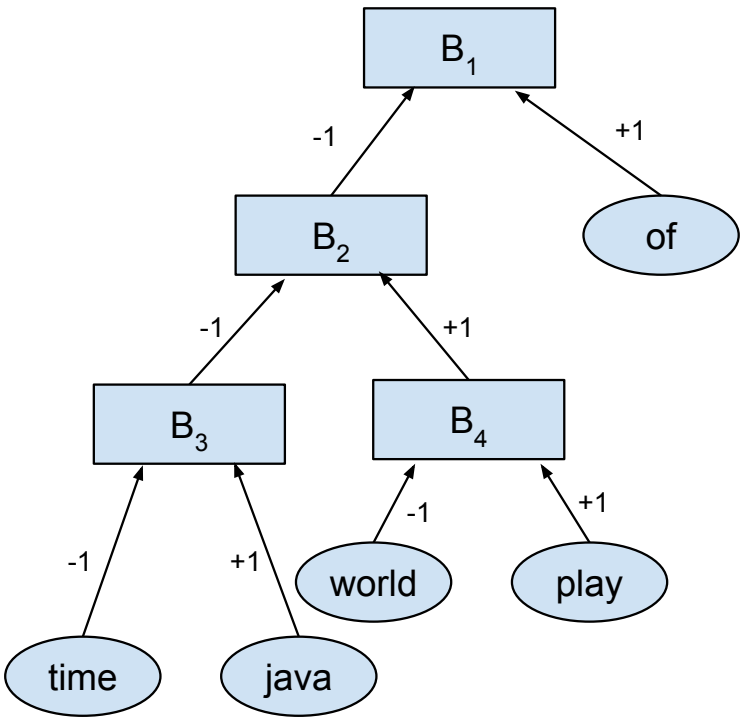
$$p(v|w) = \frac{\exp(A_w^T B_v)}{\sum_{v'=1}^{|V|} \exp(A_w^T B_{v'})}$$

Линейная сложность по размеру словаря (сотни тысяч) это дорого

Нужно придумать такую функцию $p(v|w)$, чтобы

- Считалась быстрее
- $p(v|w) > 0$ и $\sum_{v=1}^{|V|} p(v|w) = 1$

Hierarchical soft-max



$$p(v|w) = \prod_{n \in path(v)} \sigma(ch(n) In_w^T Out_n)$$

- $path(v)$ - пусть из листа v до корня
- $ch(n) = 1$ или -1 в зависимости от того, правый или левый сын узел n
- NB $\sigma(x) + \sigma(-x) = 1$

Стохастическая оптимизация

Типичная задача оптимизации в машинном обучении:

$$L(\theta) = \sum_{i=1}^{N} l(x_i, \theta) + \lambda f(\theta) \to \min_{\theta}$$

На каждой итерации

- $\theta^{(t+1)} = \theta^{(t)} - \rho_t \nabla L(\theta)$
- $\nabla L(\theta) \approx g(\theta)$

Стохастическая оптимизация

Типичная задача оптимизации в машинном обучении:

$$L(\theta) = \sum_{i=1}^{N} l(x_i, \theta) + \lambda f(\theta) \to \min_{\theta}$$

На каждой итерации

- $\theta^{(t+1)} = \theta^{(t)} - \rho_t \nabla L(\theta)$
- $\nabla L(\theta) \approx g(\theta)$

Условия сходимости:

- $\mathbb{E} g(\theta) = \nabla L(\theta)$
- $\sum_t \rho_t = \infty$, но $\sum_t \rho_t^2 < \infty$

Стохастическая оптимизация

Типичная задача оптимизации в машинном обучении:

$$L(\theta) = \sum_{i=1}^{N} l(x_i, \theta) + \lambda f(\theta) \to \min_{\theta}$$

На каждой итерации

- $\theta^{(t+1)} = \theta^{(t)} - \rho_t \nabla L(\theta)$
- $\nabla L(\theta) \approx g(\theta)$

Условия сходимости:

- $\mathbb{E} g(\theta) = \nabla L(\theta)$
- $\sum_t \rho_t = \infty$, но $\sum_t \rho_t^2 < \infty$

Как вариант, $g(\theta) = N \nabla l(x_j, \theta) + \lambda \nabla f(\theta)$

Стохастическая оптимизация

$$\mathcal{F}(\theta) = \sum_i \sum_j \underbrace{\log p(w_j|w_i,\theta)}_{l(x_t,\theta)} \to \max_\theta$$

На каждой итерации

- $\theta^{(t+1)} = \theta^{(t)} + \rho_t \nabla \mathcal{F}(\theta)$
- $\nabla \mathcal{F}(\theta) \approx N \nabla \sum_j \log p(w_j|w_i,\theta)$

Стохастическая оптимизация

$$\mathcal{F}(\theta) = \sum_i \sum_j \underbrace{\log p(w_j|w_i,\theta)}_{l(x_t,\theta)} \to \max_\theta$$

На каждой итерации

- $\theta^{(t+1)} = \theta^{(t)} + \rho_t \nabla \mathcal{F}(\theta)$
- $\nabla \mathcal{F}(\theta) \approx N \nabla \sum_j \log p(w_j|w_i,\theta)$

Learning rate и инициализация очень важны!

Секреты скорости

- Операции только вида умножения вектора на вектор
- Все помещается в кеш процессора
- Дерево Хаффмана
- Параллельная стохастическая оптимизация в общей памяти
- Никакой синхронизации между потоками!

Что происходит с многозначными словами словами

- “Смешивание” смыслов
- “доминирование” наиболее частотного смысла

Ближайшие слова к слову platform

- platforms
- four-car-long
- 10-car-long
- six-car-long
- eight-car-long
- accessed
- 12-car-long
- cross-platform
- ...
- rails

- Наблюдаемые данные: $\{(x_i, \mathbf{y}_i)\}_{i=1}^N$, где x_i это i-ое в тексте, а $\mathbf{y}_i$ - его контекст
- Для каждого слова w число векторов T фиксировано
- Введем скрытые переменные z_i - номер смысла для слова x_i

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \prod_{i=1}^{N} p(z_i) \prod_{j \in c(i)} p(y_{ij}|z_i, x_i)$$

- $p(v|z = k, w) = \frac{\exp(B_v^T A_{wk})}{\sum_s \exp(B_s^T A_{wk})}$

Обучение векторов

Стохастический EM-алгоритм:

$$\log p(\mathbf{y}|\mathbf{x}, A, B) \geq \mathcal{L}(q(\mathbf{z}), A, B) =$$

$$\mathbb{E}_q \Big[\sum_i \big(\log p(z_i) + \sum_j \log p(y_{ij}|z_i, x_i) \big) - \sum_i \log q(z_i) \Big] \to \max_{q(\mathbf{z}), A, B}$$

Обучение векторов

Стохастический EM-алгоритм:

$$\log p(\mathbf{y}|\mathbf{x}, A, B) \geq \mathcal{L}(q(\mathbf{z}), A, B) =$$

$$\mathbb{E}_q \Big[\sum_i \big(\log p(z_i) + \sum_j \log p(y_{ij}|z_i, x_i) \big) - \sum_i \log q(z_i) \Big] \to \max_{q(\mathbf{z}),A,B}$$

- Для каждого слова i:
  - E-шаг: infer
    $q(z_i = k) = p(z_i = k|x_i, \mathbf{y}_i) \propto \exp(\sum_j \log p(y_{ij}|k, x_i))$
    for $k = 1, \ldots, T$
  - M-шаг: сделать шан по градиенту функции

$$\mathcal{F}_i(A, B) = \mathbb{E} = \sum_j \sum_k q(z_i = k) \log p(y_{ij}|k, x_i)$$

- взвешенный функционал Skip-gram

Результаты

Ближайшие слова к слову platform

fwd	stabling	software
sedan	turnback	ios
fastback	pebblemix	freeware
chrysler	citybound	netfront
hatchback	metcard	linux
notchback	underpass	microsoft
rivieraoldsmobile	sidings	browser
liftback	tram	desktop
superoldsmobile	cityrail	interface
sheetmetal	trams	newlib

Проблемы наивной многосмысловой модели

- Непонятно, как выбирать число смыслов T
  - При больших T происходит расщепление смыслов
  - При малых T происходит смешение либо потеря важных смыслов

Проблемы наивной многосмысловой модели

- Непонятно, как выбирать число смыслов T
  - При больших T происходит расщепление смыслов
  - При малых T происходит смешение либо потеря важных смыслов
- Вывод: необходимо определять число смыслов автоматически

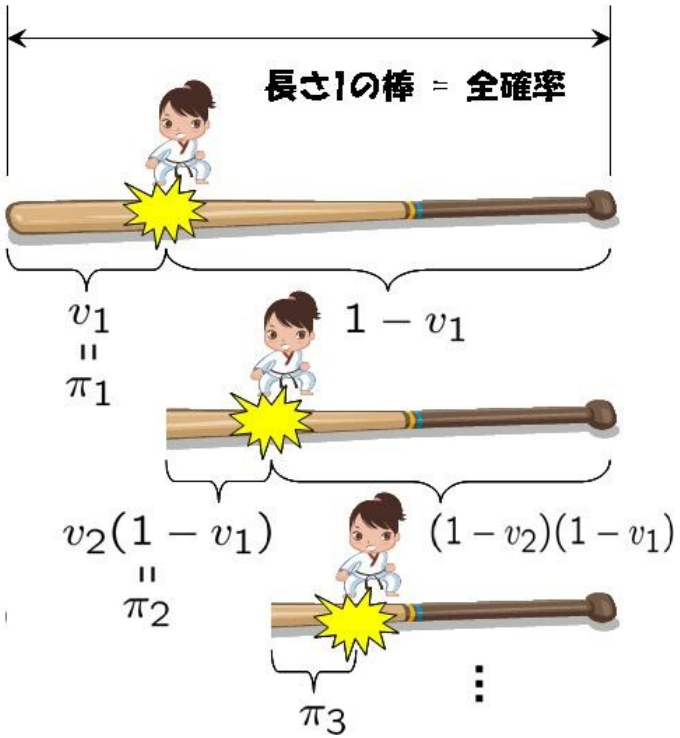
Процесс Дирихле для автоматического определения числа смыслов

Пусть каждое слово w ассоциироровано с собственным процессом Дирихле.
Рассмотрим его представление через процесс разлома палки (англ. stick-breaking process):

Процесс Дирихле для автоматического определения числа смыслов

Пусть каждое слово w ассоциировано с собственным процессом Дирихле.

Рассмотрим его представление через процесс разлома палки (англ. stick-breaking process):



$$\beta_{wk} \sim \mathtt{Beta}(1, \alpha), \quad k = 1, 2, \ldots$$

$$\pi_{wk} = \beta_{wk} \prod_{t<k} (1 - \beta_{vt}) = \beta_{wk}(1 - \sum_{t<k} \pi_{wt})$$

Процесс Дирихле для автоматического определения числа смыслов

Процесс Дирихле (потенциально неограниченное число смыслов):

$$\beta_{wk} \sim \texttt{Beta}(1, \alpha), \quad k = 1, 2, \ldots$$
$$\pi_{wk} = \beta_{wk} \prod_{t<k} (1 - \beta_{vt}) = \beta_{wk}(1 - \sum_{t<k} \pi_{wt})$$

Выбор смысла:

$$p(z = k|\pi, w) = \pi_{wk}$$

Модель предсказания слова остается прежней:

$$p(v|z = k, w) = \frac{\exp(B_v^T A_{wk})}{\sum_s \exp(B_s^T A_{wk})}$$

Вариационное приближение

Целевая функция:

$$\log p(\mathbf{x}|\mathbf{y},\theta) \geq \mathcal{L}(q,\theta) = \mathbb{E}_q \Big[\sum_{v=1}^{V} \sum_{k=1}^{T} \log p(\beta_{vk}|\alpha) - \log q(\beta_{vk}) + \sum_{i=1}^{N} \big(\log p(z_i|x_i,\beta) - \log q(z_i) + \sum_{j=1}^{C} \log p(y_{ij}|z_i,x_i,\theta) \big) \Big]$$

Вариационное приближение

Целевая функция:

$$\log p(\mathbf{x}|\mathbf{y},\theta) \geq \mathcal{L}(q,\theta) = \mathbb{E}_q \Big[\sum_{v=1}^{V} \sum_{k=1}^{T} \log p(\beta_{vk}|\alpha) - \log q(\beta_{vk}) + \sum_{i=1}^{N} \big(\log p(z_i|x_i,\beta) - \log q(z_i) + \sum_{j=1}^{C} \log p(y_{ij}|z_i,x_i,\theta) \big) \Big]$$

Вариационное приближение апостериорного распределения:

$$p(\beta,\mathbf{z},\mathbf{x}|\mathbf{y},\theta) \approx \prod_{v=1}^{V} \prod_{k=1}^{T} q(\beta_{vk}) \prod_{i=1}^{N} q(z_i)$$

Число возможных смыслов ограничено сверху числом T.

Новая нижняя оценка

Заметим, что $q(\mathbf{z})$ однозначно пересчитывается по $q(\beta)$:

$$q^*(\mathbf{z}) = \exp(\mathbb{E}_{q(\beta)} \log p(\mathbf{y}, \mathbf{z}, \beta | \mathbf{x}, \theta)) + \mathtt{const}$$

Новая нижняя оценка

Заметим, что $q(\mathbf{z})$ однозначно пересчитывается по $q(\beta)$:

$$q^*(\mathbf{z}) = \exp(\mathbb{E}_{q(\beta)} \log p(\mathbf{y}, \mathbf{z}, \beta | \mathbf{x}, \theta)) + \mathtt{const}$$

Рассмотрим $\mathcal{L}^*(q(\beta), \theta) = \mathcal{L}(q(\beta), q^*(\mathbf{z}), \theta)$

Новая нижняя оценка

Заметим, что $q(\mathbf{z})$ однозначно пересчитывается по $q(\beta)$:

$$q^*(\mathbf{z}) = \exp(\mathbb{E}_{q(\beta)} \log p(\mathbf{y}, \mathbf{z}, \beta | \mathbf{x}, \theta)) + \mathtt{const}$$

Рассмотрим $\mathcal{L}^*(q(\beta), \theta) = \mathcal{L}(q(\beta), q^*(\mathbf{z}), \theta)$

Кстати, $\mathcal{L}^*(q(\beta), \theta) \geq \mathcal{L}(q(\beta), q(\mathbf{z}), \theta)$

Новая нижняя оценка

Заметим, что $q(\mathbf{z})$ однозначно пересчитывается по $q(\beta)$:

$$q^*(\mathbf{z}) = \exp(\mathbb{E}_{q(\beta)} \log p(\mathbf{y}, \mathbf{z}, \beta | \mathbf{x}, \theta)) + \mathtt{const}$$

Рассмотрим $\mathcal{L}^*(q(\beta), \theta) = \mathcal{L}(q(\beta), q^*(\mathbf{z}), \theta)$

Кстати, $\mathcal{L}^*(q(\beta), \theta) \geq \mathcal{L}(q(\beta), q(\mathbf{z}), \theta)$

Теперь мы можем оптимизировать $\mathcal{L}^*$, которая зависит только от **глобальных параметров**

Общая схема алгоритма

Будем выполнять покоординатную стохастическую оптимизацию:

- Для каждого слова i:
- E-шаг:

$$q(z_i = k) = \exp(\mathbb{E}_{q(\beta_{x_i})} \log \pi_{x_i k} + \sum_j \log p(y_{ij}|x_i, k, \theta)) + \text{const}$$

- M-шаг:
  - Оптимизация векторов:

$$\theta^{t+1} = \theta^t + \rho_t \nabla_\theta \sum_{k=1}^{T} q(z_i = k) \sum_j \log p(y_{ij}|x_i, k, \theta)$$

  - Оптимизация глобального вар. распределения:

$$q^{t+1}(\beta) = q^t(\beta) + \gamma_t \nabla_{q(\beta)} \ldots ?$$

Оптимизация глобальных распределений

Вариационные распределения $q(\beta_w) = \prod_{k=1}^{T} q(\beta_{wk})$ зависят только от числа слов, отнесенных к тому или иному смыслу:

$$q(\beta_{wk}) = \texttt{Beta}(a_{nk}, b_{nk})$$

$$a_{nk} = 1 + n_{wk} \quad b_{nk} = \alpha + \sum_{r=k+1}^{T} n_{wr}$$

Оптимизация глобальных распределений

Вариационные распределения $q(\beta_w) = \prod_{k=1}^{T} q(\beta_{wk})$ зависят только от числа слов, отнесенных к тому или иному смыслу:

$$q(\beta_{wk}) = \texttt{Beta}(a_{nk}, b_{nk})$$

$$a_{nk} = 1 + n_{wk} \quad b_{nk} = \alpha + \sum_{r=k+1}^{T} n_{wr}$$

Следовательно, можно считать $\mathcal{L}^*$ параметрически зависимой от a и b

Шаг по градиенту $q(\beta)$

$$q^{t+1}(\beta) = q^t(\beta) + \gamma_t \widetilde{\nabla}_{q(\beta)} \mathcal{L}^* \Leftrightarrow \begin{array}{ll} a^{t+1} & = (1-\gamma_t)a^t + \gamma_t \hat{a} \\ b^{t+1} & = (1-\gamma_t)b^t + \gamma_t \hat{b} \end{array}$$

Шаг по градиенту $q(\beta)$

$$q^{t+1}(\beta) = q^t(\beta) + \gamma_t \widetilde{\nabla}_{q(\beta)} \mathcal{L}^* \Leftrightarrow \begin{array}{ll} a^{t+1} & = (1-\gamma_t)a^t + \gamma_t \hat{a} \\ b^{t+1} & = (1-\gamma_t)b^t + \gamma_t \hat{b} \end{array}$$

Параметры $\hat{a}$ и $\hat{b}$ получены, исходя из пересчета $q(\beta|\hat{a},\hat{b}) = \exp(\mathbb{E}_{q(\mathbf{z})} \log p(\mathbf{y},\mathbf{z},\beta|\mathbf{x},\theta)) + \mathtt{const}$, где мат. ожидание оценивается только по i-ому слову

[Hoffman et al.(2013)Hoffman, Blei, Wang, and Paisley]

Разрешение многозначности

Задача выбора смысла слова по контексту:

$$p(z = k|x, y, \theta, \mathcal{D}) = \int p(z = k|x, y, \beta, \theta)p(\beta|\theta, \mathcal{D})d\beta =$$

$$\mathbb{E}_{q(\beta)}p(z = k|x, y, \beta, \theta) = \mathbb{E}\beta_{xk}\prod_{r<k}\mathbb{E}(1 - \beta_{xr})$$

Примеры

- $p(z|\text{apple}, \text{iphone was announced today by}) = [0.9946, 0.0054]$
- $p(z|\text{apple}, \text{very tasty red}) = [0.0832, 0.9168]$

Вопросы?

Bibliography I


Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin.
A neural probabilistic language model.
Journal of Machine Learning Research, 3:1137–1155, 2003.


Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley.
Stochastic variational inference.
J. Mach. Learn. Res., 14(1):1303–1347, May 2013.
ISSN 1532-4435.


Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
CoRR, abs/1301.3781, 2013a.

Bibliography II

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean.
Distributed representations of words and phrases and their compositionality.
In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, NIPS, pages 3111–3119, 2013b.