ОБРАБОТКА ЕСТЕСТВЕННЫХ ЯЗЫКОВ

Сергей Николенко

СПбГУ — Санкт-Петербург

4 марта 2022 г.

Random facts:

- 04 марта 1733 г. был издан указ Анны Иоанновны «Об учреждении полиции в городах», согласно которому в 23 городах России были созданы полицмейстерские конторы
- 04 марта 1762 г., ровно через 29 лет после этого, император Пётр III подписал указ «О свободной для всех торговле»
- 04 марта 1803 г., ещё через 39 лет, император Александр I издал «Указ о вольных хлебопашцах», по которому землевладельцы получили право освобождать крестьян с обязательным наделением их землёй
- 04 марта 1837 г., ещё через 34 года и через 103 года после первого факта, Михаил Лермонтов был арестован за стихотворение «Смерть поэта»
- 04 марта 2012 г. прошли выборы президента Российской Федерации; Владимир Путин был избран президентом России на третий срок

Вариационное приближение для смеси гауссианов

Смесь гауссианов

- Смесь гауссианов: $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$,

$$p(\mathbf{Z} \mid \pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}},$$

$$p(\mathbf{X} \mid \mathbf{Z}, \mu, \Lambda) = \prod_{n=1}^{N} \prod_{k=1}^{K} N\left(\mathbf{x}_n \mid \mu_k, \Lambda_k^{-1}\right).$$

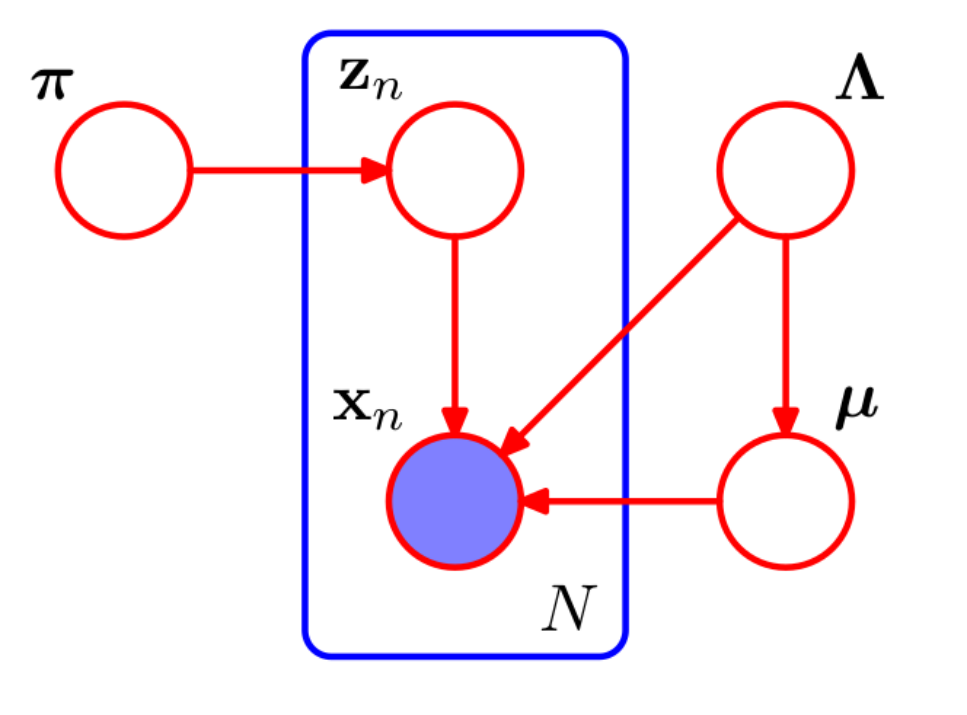
- Выберем сопряжённые априорные распределения:

$$p(\pi) = \mathrm{Dir}(\pi \mid \alpha_0) = C(\alpha_0) \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1},$$

$$\begin{aligned} p(\mu, \Lambda) &= p(\mu \mid \Lambda) p(\Lambda) \\ &= \prod_{k=1}^{K} N\left(\mu_k \mid \mathbf{m}_0, (\beta_0 \Lambda_k)^{-1}\right) W(\Lambda_k \mid \mathbf{W}_0, \nu_0). \end{aligned}$$

Смесь гауссианов

- Вот такая графическая модель:



- Распределение Дирихле пусть будет симметричное для простоты; часто ещё $\mathbf{m}_0 = 0$.
- Заметьте разницу между латентными переменными и параметрами модели.

Вариационное приближение

- Теперь вариационное приближение. Сначала сама факторизация:

$$p(\mathbf{X}, \mathbf{Z}, \pi, \mu, \Lambda) = p(\mathbf{X} \mid \mathbf{Z}, \mu, \Lambda)p(\mathbf{Z} \mid \pi)p(\pi)p(\mu \mid \Lambda)p(\Lambda).$$

- Мы наблюдаем только $\mathbf{X}$, остальное всё надо как-то оценить.
- Интересно, что единственное предположение про наше вариационное приближение выглядит так:

$$q(\mathbf{Z}, \pi, \mu, \Lambda) = q(\mathbf{Z})q(\pi, \mu, \Lambda).$$

- И всё! Дальше всё само собой получится. Но не сразу...

Вариационное приближение

- Сначала $q^*(\mathbf{Z})$:

$$\begin{aligned}\ln q^*(\mathbf{Z}) &= \mathrm{E}_{\pi,\mu,\Lambda}\left[\ln p(\mathbf{X},\mathbf{Z},\pi,\mu,\Lambda)\right] + \text{const} \\ &= \mathrm{E}_{\pi,\mu,\Lambda}\left[\ln p(\mathbf{Z}\mid\pi)\right] + \mathrm{E}_{\mu,\Lambda}\left[\ln p(\mathbf{X}\mid\mathbf{Z},\mu,\Lambda)\right] + \text{const} \\ &= \ldots = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\ln\rho_{nk} + \text{const},\end{aligned}$$

$$\text{где } \ln\rho_{nk} = \mathrm{E}[\ln\pi_k] + \frac{1}{2}\mathrm{E}[\ln|\Lambda_k|] - \frac{D}{2}\ln(2\pi) - \\ - \frac{1}{2}\mathrm{E}_{\mu_k,\Lambda_k}\left[(\mathbf{x}_n-\mu_k)^\top\Lambda_k(\mathbf{x}_n-\mu_k)\right].$$

- Нормируем:

$$q^*(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}}, \text{ где } r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K}\rho_{nj}}.$$

Вариационное приближение

- Теперь $\mathrm{E}[z_{nk}] = r_{nk}$, т.е. r_{nk} – то, насколько точка $\mathbf{x}_n$ принадлежит кластеру k.
- Можно определить статистики с их учётом, как обычно:

$$N_k = \sum_{n=1}^{N} r_{nk},$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n,$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^\top.$$

- То же самое происходило и в EM-алгоритме.

Вариационное приближение

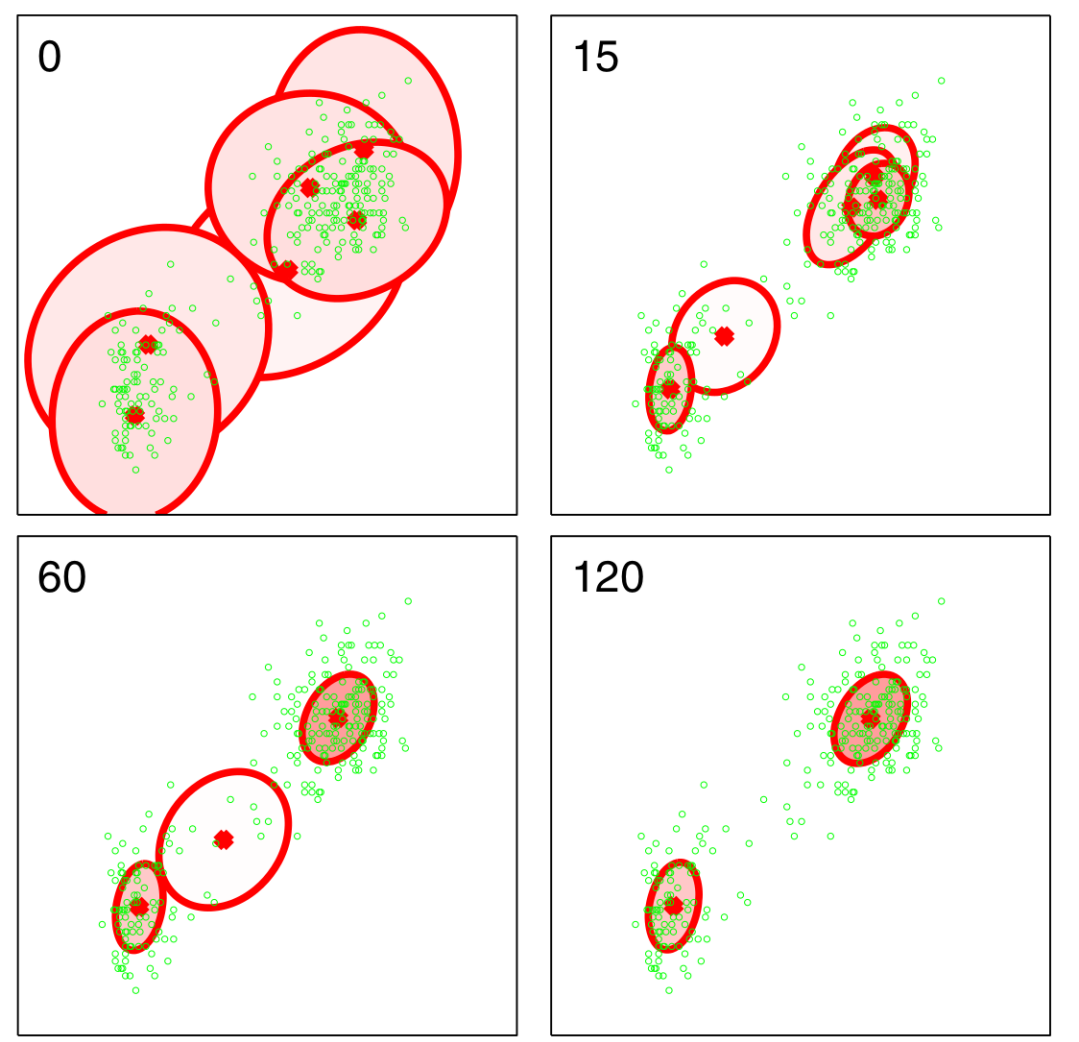
- Теперь $q^*(\pi, \mu, \Lambda)$:

$$\ln q^*(\pi, \mu, \Lambda) = \ln p(\pi) + \sum_{k=1}^{K} \ln p(\mu_k, \Lambda_k) + \mathrm{E}_{\mathbf{Z}}[\ln p(\mathbf{Z} \mid \pi)]$$
$$+ \sum_{k=1}^{K} \sum_{n=1}^{N} \mathrm{E}[z_{nk}] \ln N(\mathbf{x}_n \mid \mu_k \Lambda_k^{-1}) + \mathrm{const}.$$

- Вот уже получилось, что $q^*(\pi, \mu, \Lambda)$ раскладывается в $q^*(\pi)q^*(\mu, \Lambda)$, опять же без предположений.
- Более того, $q^*(\mu, \Lambda) = \prod_{k=1}^{K} q(\mu_k, \Lambda_k)$.
- И теперь можно по отдельности посчитать (упражнение), получится типичный M-шаг.
- Причём распределения останутся той же формы (т.к. были сопряжённые).

Вариационное приближение

- Теперь даже model selection автоматически получается, просто у некоторых компонент $N_k \approx 0$:



- Никакого оверфиттинга или коллапса компонент.

Другие примеры

- Есть другие примеры вариационных приближений.
- Обращение матриц; например, для линейной регрессии надо посчитать $\beta^* = C^{-1}\mathbf{b}$:

$$J(\beta) = \frac{1}{2}(\beta^* - \beta)^\top C(\beta^* - \beta) = \ldots = \text{Const} - \beta^\top \mathbf{b} + \frac{1}{2}\beta^\top C\beta,$$

  и теперь можно решать такую задачу выпуклой оптимизации.
- Метод конечных элементов – для уравнения Пуассона $-u''(x) = f(x)$, $x \in (a, b)$:

$$J(u) = \frac{1}{2}\int_a^b \left(u'(x) - u^{*\prime}(x)\right)^2 \mathrm{d}x = \ldots = \text{Const} - \int_a^b u(x)f(x)\mathrm{d}x + \frac{1}{2}\int_a^b u'(x)^2\mathrm{d}x,$$

  и если ищем в подпространстве $\tilde{u}(x) = \sum_{i=1}^k \alpha_i\phi_i(x)$, то опять

$$\tilde{J}(\alpha) = \alpha^\top \mathbf{b} + \frac{1}{2}\alpha^\top C\alpha.$$

В графических моделях

- В графических моделях – теория среднего поля (mean field theory). Пусть дано $p(\mathbf{x})$, $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_h)$, и надо найти

$$\log p(\mathbf{x}_v) = \log \sum_{\mathbf{x}_h} p(\mathbf{x}_v, \mathbf{x}_h), \quad p(\mathbf{x}_h \mid \mathbf{x}_v) = p(\mathbf{x}_h, \mathbf{x}_v)/p(\mathbf{x}_v).$$

- Опять делаем тот же трюк:

$$J(q) = \log p(\mathbf{x}_v) - \mathrm{KL}(q_{\mathbf{x}_h} \| p_{\mathbf{x}_h|\mathbf{x}_v}) = \log p(\mathbf{x}_v) - \sum_{\mathbf{x}_h} q(\mathbf{x}_h) \log \frac{q(\mathbf{x}_h)}{p(\mathbf{x}_h \mid \mathbf{x}_v)}$$

$$\ldots = H(q) + \mathbb{E}_q \left[\log p(\mathbf{x}_h, \mathbf{x}_v)\right] = H(q) + \sum_{C \in C} \sum_{\mathbf{x}_{C \cap h}} q(\mathbf{x}_{C \cap h}) \log \Psi_C(\mathbf{x}_C),$$

где $q(\mathbf{x}_{C \cap h})$ – маргинальная вероятность по скрытым переменным из клики C.

- Теория среднего поля – это когда $q(\mathbf{x}_h) = \prod_{i \in h} q_i(x_i)$.

Задачи обработки естественного языка

Я вгрызаюсь в тело текста

- Работы, связанные с естественным языком, — это одна из ключевых задач для создания искусственного интеллекта.
- А.А. Марков-старший, 1913: численные эксперименты с текстом «Евгения Онегина», языковые модели через марковские цепи
- Ранний оптимизм: 1950-е, Ноам Хомский, Дартмутский семинар.
- Georgetown-IBM experiment, 1954: «через 3-5 лет машинный перевод будет решён».
- Но оказалось, что всё сложно; первая зима нейронных сетей — это отчасти fail проекта по машинному переводу.
- И до сих пор мы, хотя умеем решать связанные с языком задачи всё лучше и лучше, очень далеки от истинного понимания.

Я вгрызаюсь в тело текста

- Почему сложно? Во многом из-за модели окружающего мира, commonsense reasoning.
- Пример — разрешение *анафоры*:
  - мама вымыла раму, и теперь она блестит;
  - мама вымыла раму, и теперь она устала.
- Это пример хорошо определённой задачи, фактически задачи классификации, для которой легко набрать датасет, но она в нетривиальных случаях всё равно очень, очень сложная.
- Какие ещё бывают задачи в обработке естественного языка?

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging): разметить в заданном тексте слова по частям речи (существительное, глагол, прилагательное...) и, возможно, по морфологическим признакам (род, падеж...);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation): разделить слова в заданном тексте на *морфемы*, т.е. синтаксические единицы вроде приставок, суффиксов и окончаний; для некоторых языков (например, английского) это не очень актуально, но в русском языке морфологии очень много;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- другой вариант задачи о морфологии отдельных слов — *стемминг* (stemming), в котором требуется выделить основы слов, или *лемматизация* (lemmatization), в которой слово нужно привести к базовой форме (например, форме единственного числа мужского рода);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ предложения* (sentence boundary disambiguation): разбить заданный текст на предложения; задача непростая даже в русском и английском, а в языках вроде китайского весьма нетривиальной становится даже задача *пословной сегментации* (word segmentation), потому что поток иероглифов без пробелов может делиться на слова по-разному;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition): найти в тексте собственные имена людей, географических и прочих объектов, разметив их по типам сущностей (люди, места, названия компаний и т.п.);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation): выбрать, какой из омонимов, какой из разных смыслов одного и того же слова используется в данном отрывке текста;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation);
- *синтаксический парсинг* (syntactic parsing): по заданному предложению (и, возможно, его контексту) построить его синтаксическое дерево, прямо по Хомскому;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation);
- *синтаксический парсинг* (syntactic parsing);
- *разрешение кореференций* (coreference resolution): определить, к каким объектам или другим частям текста относятся те или иные слова и обороты; частный случай этой задачи — то самое разрешение анафоры, которое мы обсуждали выше.

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models): по заданному отрывку текста предсказать следующее слово или следующий символ; эта задача очень важна, например, для распознавания речи;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis): определить по тексту его тональность, т.е. позитивное ли отношение несёт этот текст или негативное;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis);
- *выделение отношений* или *фактов* (relationship extraction, fact extraction): выделить из текста хорошо определённые отношения или факты об упоминающихся там сущностях; например, кто с кем находится в родственных отношениях, в каком году основана упоминающаяся в тексте компания и т.д.;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis);
- *выделение отношений* или *фактов* (relationship extraction, fact extraction);
- *ответы на вопросы* (question answering): дать ответ на заданный вопрос; в зависимости от постановки это может быть или чистая классификация (выбор из вариантов ответа, как в тесте), или классификация с очень большим числом классов (ответы на фактологические вопросы вроде «кто» или «в каком году»), или даже порождение текста (если отвечать на вопросы нужно в рамках естественного диалога).

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization): по тексту породить его краткое содержание, abstract, так сказать; это можно рассмотреть как задачу классификации, если просить модель выбрать из текста готовые предложения, лучше всего отражающие смысл всего текста, а можно как задачу порождения, если краткое содержание нужно написать с нуля;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization);
- *машинный перевод* (machine translation): по тексту на одном языке породить соответствующий текст на другом языке;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization);
- *машинный перевод* (machine translation);
- *диалоговые модели* (dialog and conversational models): поддержать разговор с человеком.

Поиск дубликатов

- Есть и другие задачи, менее специфичные именно для NLP.
- Например, *поиск дубликатов*: как найти похожие тексты? Классический ответ довольно простой:
  - сначала представим текст как множество (слов? или чего?)
  - выберем для них представление (хешем)
  - найдём похожесть между множествами чисел (как это?)

Кстати, об n-граммах

- n-граммы (шинглы) попадаются в NLP часто. Например, для языковых моделей:
  - предсказываем следующее слово по нескольким предыдущим
  - оцениваем через *перплексию*:

$$PP(S) = p(w_1, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i \mid w_1, \dots, w_{i-1})}} = 2^{H(S)}$$

  - надо сглаживать:
    - лапласовское сглаживание;
    - откат (backoff) и интерполяция;
    - Kneser-Ney smoothing:

$$p_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - d)}{c(w_{i-1})} + \lambda(w_{i-1})p_{CONT}(w_i),$$

      где d – константная (!) разница между числом биграмм в training set и validation set, $\lambda(w_{i-1}) = \frac{d|\{w : c(w_{i-1}w) > 0\}|}{\sum_w c(w_{i-1}w)}$ – вес интерполяции, $p_{CONT}(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}w_j) > 0\}|}$.

Кстати, об n-граммах

- А ещё n-граммами разумно расширять набор токенов: Криштиану_Роналду, Евгений_Онегин...
- Но добавлять все пары слов было бы странно и контрпродуктивно.
- Какие нужно добавлять биграммы?..

Кстати, об n-граммах

- А ещё n-граммами разумно расширять набор токенов: Криштиану_Роналду, Евгений_Онегин...
- Но добавлять все пары слов было бы странно и контрпродуктивно.
- Какие нужно добавлять биграммы?..
- ...неожиданные! То есть сильно выбивающиеся из предположения независимости:

$$p(\text{Евгений\_Онегин}) \gg p(\text{Евгений})p(\text{Онегин})$$

Общие замечания

- Часто задачу можно выразить как задачу классификации (*категоризации текстов*) или регрессии.
- В таких случаях можно использовать обычные классификаторы: логистическую регрессию, SVM и т.п.
- Вопрос: что давать им на вход?
- Можно просто считать каждое слово своей размерностью и давать векторы документов как счётчики слов (bag of words).
- Почему это может быть плохо?

Общие замечания

- Одна причина – всё будет сильно зависеть от слов, которые как раз ничего не определяют, и стоп-листами это не решить.
- Вариант – *tf-idf веса*:

$$\mathrm{tf}(t, d) = \frac{n_t}{|d|}, \quad \mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D \mid t \in d\}|}.$$

- Обычно результат улучшается, если заменить просто число вхождений на tf-idf веса.
- А сейчас чаще всего используют word embeddings, о них позже...

Наивный байесовский классификатор

Категоризация текстов

- Классическая задача машинного обучения и information retrieval – категоризация текстов.
- Дан набор текстов, разделённый на категории. Нужно обучить модель и потом уметь категоризовать новые тексты.
- Атрибуты $a_1, a_2, \ldots, a_n$ – это слова, v – тема текста (или атрибут вроде «спам / не спам»).
- Bag-of-words model: забываем про порядок слов, составляем словарь. Теперь документ – это вектор, показывающий, сколько раз каждое слово из словаря в нём встречается.

Naive Bayes

- Заметим, что даже это – сильно упрощённый взгляд: для слов ещё довольно-таки важен порядок, в котором они идут...
- Но и это ещё не всё: получается, что $p(a_1, a_2, \dots, a_n | x = v)$ – это вероятность *в точности такого набора слов* в сообщениях на разные темы. Очевидно, такой статистики взять неоткуда.
- Значит, надо дальше делать упрощающие предположения.
- Наивный байесовский классификатор – самая простая такая модель: давайте предположим, что все слова в словаре условно независимы при условии данной категории.

Naive Bayes

- Иначе говоря:

$$p(a_1, a_2, \dots, a_n | x = v) = p(a_1 | x = v) p(a_2 | x = v) \dots p(a_n | x = v).$$

- Итак, наивный байесовский классификатор выбирает v как

$$v_{NB}(a_1, a_2, \dots, a_n) = \arg\max_{v \in V} p(x = v) \prod_{i=1}^{n} p(a_i | x = v).$$

- В парадигме классификации текстов мы предполагаем, что разные слова в тексте на одну и ту же тему появляются независимо друг от друга. Однако, несмотря на такие бредовые предположения, naive Bayes на практике работает очень даже неплохо (и этому есть разумные объяснения).

Многомерная модель

- Но в деталях реализации наивного байесовского классификатора есть тонкости.
- Сейчас мы рассмотрим два разных подхода к naive Bayes, которые дают разные результаты: мультиномиальный (multinomial) и многомерный (multivariate).

Многомерная модель

- В многомерной модели документ – это вектор бинарных атрибутов, показывающих, встретилось ли в документе то или иное слово.
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что встретилось каждое слово из документа и вероятности того, что не встретилось каждое (словарное) слово, которое не встретилось.
- Получается модель многомерных испытаний Бернулли. Наивное предположение в том, что события «встретилось ли слово» предполагаются независимыми.

Многомерная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|V|$, состоящий из битов B_{it}; $B_{it} = 1$ iff слово w_t встречается в документе d_i.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = \prod_{t=1}^{|V|} \left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right).$$

- Для обучения такого классификатора нужно обучить вероятности $p(w_t \mid c_j)$.

Многомерная модель

- Обучение – дело нехитрое: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем биты B_{it} (знаем документы).
- Тогда можно подсчитать оптимальные оценки вероятностей того, что то или иное слово встречается в том или ином классе (при помощи лапласовой оценки):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} B_{it} p(c_j \mid d_i)}{2 + \sum_{i=1}^{|D|} p(c_j \mid d_i)}.$$

Многомерная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$c = \arg\max_j p(c_j)p(d_i \mid c_j) =$$

$$= \arg\max_j \left(\frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)\right) \prod_{t=1}^{|V|} \left(B_{it}p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right) =$$

$$= \arg\max_j \left(\log(\sum_{i=1}^{|D|} p(c_j \mid d_i)) + \sum_{t=1}^{|V|} \log\left(B_{it}p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right)\right).$$

Мультиномиальная модель

- В мультиномиальной модели документ – это последовательность событий. Каждое событие – это случайный выбор одного слова из того самого «bag of words».
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что мы достали из мешка те самые слова, которые встретились в документе. Наивное предположение в том, что мы достаём из мешка разные слова независимо друг от друга.
- Получается мультиномиальная генеративная модель, которая учитывает количество повторений каждого слова, но не учитывает, каких слов *нет* в документе.

Мультиномиальная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|d_i|$, состоящий из слов, каждое из которых «вынуто» из словаря с вероятностью $p(w_t \mid c_j)$.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}},$$

  где N_{it} – количество вхождений w_t в d_i.
- Для обучения такого классификатора тоже нужно обучить вероятности $p(w_t \mid c_j)$.

Мультиномиальная модель

- Обучение: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем вхождения N_{it}.
- Тогда можно подсчитать апостериорные оценки вероятностей того, что то или иное слово встречается в том или ином классе (не забываем сглаживание – правило Лапласа):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} p(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} p(c_j \mid d_i)}.$$

Мультиномиальная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$
\begin{aligned}
c &= \arg\max_j p(c_j)p(d_i \mid c_j) = \\
&= \arg\max_j \left(\frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}} = \\
&= \arg\max_j \left(\log \left(\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) + \sum_{t=1}^{|V|} N_{it} \log p(w_t \mid c_j) \right).
\end{aligned}
$$

Как можно обобщить наивный байес

- В наивном байесе есть два сильно упрощающих дело предположения:
  - мы знаем метки тем всех документов;
  - у каждого документа только одна тема.
- Мы сейчас уберём оба эти ограничения.
- Во-первых, что можно сделать, если мы не знаем метки тем, т.е. если датасет неразмеченный?

Кластеризация

- Тогда это превращается в задачу кластеризации.
- Её можно решать EM-алгоритмом (Expectation-Maximization, используется в ситуациях, когда есть много скрытых переменных, причём если бы мы их знали, модель стала бы простой):
  - на E-шаге считаем ожидания того, какой документ какой теме принадлежит;
  - на M-шаге пересчитываем наивным байесом вероятности $p(w \mid t)$ при фиксированных метках.
- Это простое обобщение.

Спасибо!

Спасибо за внимание!