Маргинализация в общем виде

Сергей Николенко

Machine Learning — CS Club, весна 2008

Outline

Введение

- Мы уже много лекций подряд рассматривали задачи маргинализации, потому что они являются основными для байесовского вывода.
- Рассматривали много частных случаев.
- Сегодня мы наконец-то обобщим всё то, чем занимались, до практически самой общей из применимых на практике ситуаций.

Функция в общем виде

- Чтобы поставить задачу в общем виде, рассмотрим функцию
$$p^*(X) = \prod_{j=1}^{m} f_j(X_j),$$
где $X = \{x_i\}_{i=1}^n$, $X_j \subseteq X$.
- Т.е. мы рассматриваем функцию, которая раскладывается в произведение нескольких других функций.

Нормализованная функция в общем виде

- Мы написали p^*, потому что обычно, чтобы получилась вероятность, нужно ещё нормализовать:

$$p(X) = \frac{1}{Z} \prod_{j=1}^{m} f_j(X_j),$$

где $Z = \sum_X \prod_{j=1}^{m} f_j(X_j)$.

Пример

$$p^*(X) = f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1,x_2)f_5(x_2,x_3),$$

где

$$f_1(x_1) = \begin{cases} 0.05, & x_1 = 0 \\ 0.95, & x_1 = 1 \end{cases} \qquad f_4(x_1x_2) = \begin{cases} 1, & x_1 = x_2 \\ 0, & x_1 \neq x_2 \end{cases}$$

$$f_2(x_2) = \begin{cases} 0.05, & x_2 = 0 \\ 0.95, & x_2 = 1 \end{cases} \qquad f_5(x_2x_3) = \begin{cases} 1, & x_2 = x_3 \\ 0, & x_2 \neq x_3 \end{cases}$$

$$f_3(x_3) = \begin{cases} 0.95, & x_3 = 0 \\ 0.05, & x_3 = 1 \end{cases}$$

- Что это за функция? Вспомните прошлую лекцию.

Пример

- Это функция апостериорной вероятности повторяющего кода, если вероятность ошибки равна 0.05.
- f_4 и f_5 утверждают, что изначально все биты были одинаковые, а f_1, f_2 и f_3 указывают на вероятность ошибки.
- Задачи мы тоже формулировали: это были задачи маргинализации в общем и маргинализации побитовой.
- Давайте их обобщим.

Задачи

- Задача нормализации: найти $Z = \sum_X \prod_{j=1}^m f_j(X_j)$.
- Задача маргинализации: найти

$$p_i^*(x_i) = \sum_{k \neq i} p^*(X).$$

- Задача нормализованной маргинализации: найти

$$p(x_i) = \sum_{k \neq i} p(X).$$

- Также может понадобиться, например, $p_{i_1 i_2}$, но реже.

Задачи

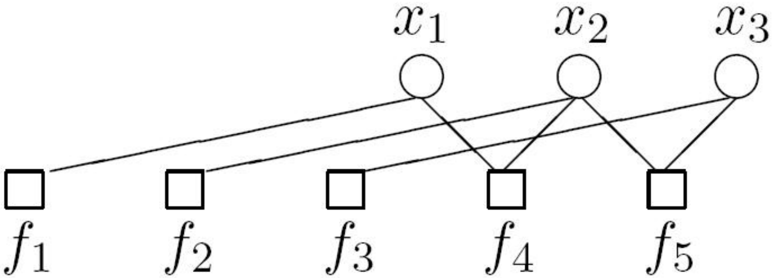
- Все эти задачи NP-трудные.
- То есть, если мир не рухнет, сложность их решения в худшем случае возрастает экспоненциально.
- Но можно решить некоторые частные случаи.
- Этим мы и займёмся.

Структура графа

- Граф состоит из вершин, соответствующих переменным, и вершин, соответствующих функциям.
- Функции соединены с переменными, которых они описывают.

Структура графа

$$p^*(X) = f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_1, x_2) f_5(x_2, x_3).$$



Идея алгоритма

- Как и прежде, идея в том, чтобы пересылать сообщения.
- Сообщения двух видов: от функций к переменным и от переменных к функциям.

Сообщения

- От переменной x_i к функции f_j:

$$q_{i\to j}(x_i) = \prod_{k\in \mathtt{nei}(i)\setminus j} r_{k\to i}(x_i).$$

- От функции f_j к переменной x_i:

$$r_{j\to i}(x_i) = \sum_{X_j\setminus x_i}\left(f_j(X_j)\prod_{k\in \mathtt{nei}(j)\setminus i} q_{k\to j}(x_k)\right).$$

Сообщения в листьях

- Отметим, что если у вершины только один сосед, то её сообщение можно вычислить, не зная входящих сообщений.
- Пустое произведение равно единице.
- Поэтому функция f_j, зависящая только от одной переменной x_i, передаёт своей переменной $r_{j\to i} = f_j(x_i)$, а переменная x_i, участвующая только в одной функции, передаёт ей 1.

Работа алгоритма I

- Предположим, что граф — дерево.
- Тогда можно начать с листьев и, постепенно вычисляя сообщения, обойти все вершины.
- При этом применяется стандартное правило передачи сообщений: сообщение можно передавать, только если его можно полностью построить.
- За количество шагов, равное диаметру графа, работа алгоритма закончится.

Работа алгоритма II

- Что делать, если граф — не дерево, и не ясно, с чего начинать?
- Вариант — начать с того, что все переменные передают сообщение 1, а потом уже его модифицируют, когда до них доходят сообщения от функций.
- Такой алгоритм не всегда работает правильно и делает много лишнего, но всё же полезен на практике.

Упражнение. Привести пример, когда алгоритм sum–product на графе с циклами работает некорректно.

Результат работы

- Когда рассылка сообщений закончится, можно будет вычислить маргиналы:

$$p_i^*(x_i) = \prod_{j \in \mathtt{nei}(i)} r_{j \to i}(x_i).$$

- Ясно, что $Z = \sum_i p_i^*(x_i)$, и $p(x_i) = \frac{1}{Z} p_i^*(x_i)$.

Упражнение. Доказать, что действительно получится $p_i^*(x_i)$.

Нормализация on-the-fly

- Возможно, мы интересуемся только нормализованными маргиналами (настоящими вероятностями).
- Тогда можно просто на каждом шаге нормализовать сообщения от переменных к функциям (от функций к переменным сообщения те же):

$$q_{i\to j}(x_i) = \alpha_{ij} \prod_{k\in \mathrm{nei}(i)\setminus j} r_{k\to i}(x_i),$$

где α_{ij} подобраны так, чтобы

$$\sum_i q_{i\to j}(x_i) = 1.$$

Упражнение. Пошагово применить вышеописанный алгоритм к функции из примера (функции правдоподобия повторяющего кода).

Разложение функции p^*

- Что такое sum–product математически?
- Изначально было разложение

$$p^*(X) = \prod_{j=1}^{m} f_j(X_j).$$

- А мы перераскладываем функцию в произведение

$$p^*(X) = \prod_{j=1}^{m} \phi_j(X_j) \prod_{i=1}^{n} \psi_i(x_i),$$

  где ϕ_j соответствуют узлам–функциям, а ψ_i — узлам–переменным.
- Изначально, до передачи сообщений, $\phi_j(X_j) = f_j(X_j)$ и $\psi_i(x_i) = 1$.

Разложение функции p^*

- Каждый раз, когда приходит сообщение $r_{j\to i}$ из функции в переменную, ϕ и ψ пересчитываются:

$$\psi_i(x_i) = \prod_{j\in \mathrm{nei}(i)} r_{j\to i}(x_i),$$

$$\phi_j(X_j) = \frac{f_j(X_j)}{\prod_{i\in \mathrm{nei}(j)} r_{j\to i}(x_i)}.$$

- Очевидно, общее произведение от этого не меняется.

Упражнение. Доказать, что в результате этого процесса ψ_i действительно станет маргиналом $p^*(x_i)$. Это докажет корректность алгоритма.

Постановка задачи

- Маргинализация — не единственная возможная постановка задачи.
- Вот, например, задача максимизации: найти значения переменных из X, которые максимизируют функцию p^*.
- Мы уже решали её и видели, какие нужны для этого инструменты.

От sum–product к max–product

- Задача максимизации p^* решится, если в узлах операцию суммирования $\sum$ заменить на max.
- Тогда нормализационная константа $Z = \sum_X p^*(x)$ превратится в $\max_X p^*(x)$, а каждый маргинал $p^*(x_i)$ станет показывать максимальное значение p^*, которого можно достичь в зависимости от значения x_i; отсюда уже тривиально можно будет найти максимизирующий набор.

От max–product к min–sum

- Умножать дороже, чем складывать. Как бы нам начать складывать вместо того чтобы умножать?

От max–product к min–sum

- Умножать дороже, чем складывать. Как бы нам начать складывать вместо того чтобы умножать?
- Естественно, надо логарифмировать (а потом добавить знак «минус»); тогда из max–product получится min–sum. Мы это уже много раз делали.
- Такой алгоритм называется алгоритмом Витерби (Viterbi).

Суть

- Sum–product работает корректно, только если граф — дерево (ну, разве что скрестить пальцы и помолиться...).
- Что делать, когда граф содержит циклы?
- Нужно использовать деревья сочленений.

Деревья сочленений — неформально

- Если цикл не сдаётся, его уничтожают, то есть заменяют весь цикл на одну вершину.
- Получается дерево, в котором уже можно работать обычным sum–product'ом; но при этом, конечно, замена нескольких вершин одной приводит к её экспоненциальному раздуванию (множество значений соответствующей переменной должно содержать все комбинации значений исходных переменных).
- О том, как это сделать формально и правильно, мы говорили на лекции по байесовским сетям.

Другие методы

- Есть приближённые методы, мы сейчас приведём пару примеров.
- Но, как мы уже видели, есть очень хороший метод работать тогда, когда нельзя применять sum–product.
- Метод заключается в том, чтобы применять sum–product. :)
- Он работает довольно часто даже тогда, когда в принципе работать не обязан (когда есть циклы).

Outline

Аппроксимационные методы

- Кроме точной маргинализации, можно рассматривать и приближённую. Особенно это важно, когда распределения непрерывные — тогда точно может просто вообще не получиться.
- Для начала рассмотрим простейший метод Лапласа.
- Этот метод прямого отношения к маргинализации не имеет — мы просто заменим сложное распределение на простое.

Суть метода Лапласа

- Рассмотрим ненормализованную плотность распределения p^* с нормализационной константой $Z = \int p^*(x)dx$.
- Предположим, что у p^* имеется максимум в точке x_0; разложим в ряд Тейлора логарифм $p^*(x)$:

$$\ln p^*(x) = \ln p^*(x_0) - \frac{c}{2}(x - x_0)^2 + \ldots,$$

где

$$c = -\left.\frac{\partial^2}{\partial x^2} \ln p^*(x)\right|_{x=x_0}.$$

Суть метода Лапласа

- А теперь приблизим p^* ненормированным гауссианом:

  $$q^*(x) = p^*(x_0)e^{-\frac{c}{2}(x-x_0)^2},$$

  а нормализационную константу приблизим константой гауссиана:

  $$Z_q = p^*(x_0)\sqrt{\frac{2\pi}{c}}.$$

- Получится гауссиан, логарифм которого совпадает с первыми двумя членами ряда Тейлора $\ln p^*$.

Обобщение метода Лапласа

- Пусть теперь $p^*(x_1, \ldots, x_k)$ работает в многомерном пространстве.
- Тогда появляется целая матрица вторых производных A:

$$A_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} \ln p^*(x) \Big|_{x=x_0}.$$

- А ряд Тейлора выглядит как

$$\ln p^*(x) = \ln p^*(x_0) - \frac{1}{2}(x - x_0)^t A(x - x_0) + \ldots,$$

где x и x_0 — k–мерные векторы.

Обобщение метода Лапласа

- Наконец, нормализационная константа гауссиана будет равна
$$Z_q = p^*(x_0)\sqrt{\frac{(2\pi)^k}{\det A}}.$$

Упражнение. Докажите это.

- То есть суть метода Лапласа — просто в том, чтобы заменить распределение похожим на него гауссианом.
- Важное замечание: чтобы гауссиан был действительно похож, надо найти максимум. Как?

Постановка

- Напомним постановку задачи (чуть сменив обозначения). Рассмотрим переменные $\mathbf{x} = (x_0, \ldots, x_{N-1})$.
- Множество переменных разбивается на *регионы* $r \subseteq [N]$; множество регионов — R.
- У каждого региона есть *ядро* $\alpha_r(\mathbf{x}_r)$, зависящее только от переменных $x \in r$.

Постановка

- Рассмотрим общее совместное распределение

$$B(\mathbf{x}) = \frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r).$$

- Обозначим $B_s(\mathbf{x}_s) = \sum_{\mathbf{x}_{[N]\setminus s}} B(\mathbf{x})$ маргиналы $B(\mathbf{x})$.
- Задача: найти один или несколько маргиналов $B_s(\mathbf{x}_s)$ и/или нормировочную константу Z.

Постановка

- Рассмотрим множество R и введём на нём структуру частично упорядоченного множества по включению.
- Обозначим:
  - $\mathcal{A}(r) = \{s \in R | r \subset s\}$ (ancestors);
  - $\mathcal{D}(r) = \{s \in R | s \subset r\}$ (descendants);
  - $\mathcal{F}(r) = \{s \in R | s \subseteq r\}$ (forebears);
  - $\mathcal{P}(r) = \{s \in R | r \succ s\}$ (parents);
  - $\mathcal{C}(r) = \{s \in R | s \succ r\}$ (children).
- Аргументом может быть не обязательно регион из R, но в результате получается набор регионов из R.

Статистическая физика

- И снова вспомним статистическую физику.
- x — это типа спин i-й частицы в системе из N частиц.
- Обозначим $b(\mathbf{x})$ распределение вероятностей на конфигурациях спинов.
- Рассмотрим некоторую R-раскладываемую функцию энергии:

$$E(\mathbf{x}) = \sum_{r \in R} E_r(\mathbf{x}_r).$$

Статистическая физика

- Можно определить *вариационную свободную энергию* (по Гельмгольцу):

$$F(b(\mathbf{x})) = U(b(\mathbf{x})) - H(b(\mathbf{x})),$$

где $U(b(\mathbf{x})) = \sum_{\mathbf{x}} b(\mathbf{x}) E(x)$ — средняя энергия, $H(b(\mathbf{x})) = -\sum_{\mathbf{x}} b(\mathbf{x}) \log b(\mathbf{x})$ — энтропия системы.

- Как связать эту постановку с нашей, где нужно маргинализовать переменные из произведения?

Статистическая физика

- Положим $E_r(\mathbf{x}_r) = -\log \alpha_r(\mathbf{x}_r)$. Тогда

$$E(\mathbf{x}) = \sum_{r \in R} E_r(\mathbf{x}_r) = -\sum_{r \in R} \log \alpha_r(\mathbf{x}_r) =$$

$$= -\log(\frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r)) - \log(Z) = -\log(B(\mathbf{x})) - \log(Z).$$

Статистическая физика

- Тогда вариационная свободная энергия будет равна

$$F(b) = \sum_{\mathbf{x}} b(\mathbf{x}) \left(-\log(B(\mathbf{x})) - \log(Z)\right) + \sum_{\mathbf{x}} b(\mathbf{x}) \log b(\mathbf{x}) =$$

$$= \sum_{\mathbf{x}} b(\mathbf{x}) \log \frac{b(\mathbf{x})}{B(\mathbf{x})} - \log(Z) = D_{\mathrm{KL}}(b, B) - \log(Z),$$

где $D_{\mathrm{KL}}(b, B)$ — расстояние Кульбака–Ляйблера (относительная энтропия).

Статистическая физика

- Kullback–Leibler distance (divergence) — это информационно-теоретическая мера того, насколько далеки распределения друг от друга.

$$D_{KL}(p, q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = H(p) - H(p, q).$$

- Его ещё называют *относительной энтропией*.
- Известно, что это расстояние всегда неотрицательно, равно нулю iff $p \equiv q$.

Статистическая физика

- Зачем это всё нужно? Теперь оказалось, что у $F(b)$ один глобальный минимум, когда $b = B$, и при этом $F_0 = -\log Z$.
- Если мы решим эту задачу оптимизации, мы узнаем Z из значения функции.
- А если мы сможем выразить F через $b_r(\mathbf{x}_r)$, то argmin в этой задаче оптимизации даст нам маргинальные распределения.
- Осталось выразить.

Приближение энтропии

- Мы хотим выразить $F(b(\mathbf{x})) = U(b(\mathbf{x})) - H(b(\mathbf{x}))$ через $b_r(\mathbf{x}_r)$.
- Со средней энергией дело простое, ведь энергия и так уже разложена:

$$E(\mathbf{x}) = \sum_{r \in R} E_r(\mathbf{x}_r), \text{ и}$$

$$U(b(\mathbf{x})) = \sum_{r \in R} \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) E_r(\mathbf{x}_r).$$

- А вот с энтропией всё не так просто.

Приближение энтропии

- Вообще говоря, $H(b(\mathbf{x}))$ не раскладывается по $b_r(\mathbf{x}_r)$.
- Идея приближения Кикучи:

$$H(b(\mathbf{x})) \approx \sum_{r \in R} k_r H_r(b_r(\mathbf{x}_r)), \text{ где}$$

$$H_r(b_r(\mathbf{x}_r)) = -\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \log b_r(\mathbf{x}_r).$$

- Вопрос: какими должны быть константы k_r?

Приближение энтропии

- Разумно потребовать, чтобы там, где можно ожидать точного ответа, приближение могло быть точным.
- То есть чтобы эта взвешенная сумма могла давать точное значение энтропии тогда, когда $b(\mathbf{x}) = \frac{1}{Z}\prod_{r\in R}\alpha_r(\mathbf{x}_r)$, для всяких α_r, $r \in R$.
- “Могла давать” — потому что всё равно не всегда, а только если граф нашего чума будет без ненаправленных циклов.
- Но этим мы сейчас не будем заниматься; и доказывать правильный ответ не будем.

Приближение энтропии

- Ответ: коэффициенты k_r нужно брать равными функции Мёбиуса на чуме $R \cup \{[N]\}$:

$$k_r = -\mu(r, [N]).$$

- Они тогда будут однозначно определяться формулой обращения Мёбиуса:

$$k_r = 1 - \sum_{s \in \mathcal{A}(r)} k_s.$$

- Что вы знаете о функции Мёбиуса?

Функция Мёбиуса

- Возможно, вы знаете функцию Мёбиуса на натуральных числах:

$$\mu(n) = \begin{cases} 1, & n = p_1 \dots p_{2l}, \\ -1, & n = p_1 \dots p_{2l+1}, \\ 0, & p^2 \mid n. \end{cases}$$

- Функция сама по себе очень интересная, со многими интересными свойствами и применениями.

Функция Мёбиуса

- Главное свойство — формула обращения Мёбиуса: для арифметических функций f и g

$$g(n) = \sum_{d|n} f(d) \quad \Leftrightarrow \quad f(n) = \sum_{d|n} \mu(\frac{n}{d}) g(d).$$

- Функция Мёбиуса позволяет обратить сумму по делителям. Можно и не обязательно в целых числах:

$$g(n) = \sum_{n \le x} f(\frac{x}{n}) \quad \Leftrightarrow \quad f(x) = \sum_{n \le x} \mu(n) g(\frac{x}{n}).$$

Функция Мёбиуса

- Применение в физике: рассмотрим модель свободного риманова газа (где частицы не взаимодействуют).
- В ней фундаментальные частицы (примоны) имеют энергии $\log p$ для простых чисел p.
- А общий гамильтониан системы определяется целым числом:

$$|n\rangle = |p_1, p_2, p_3, \ldots\rangle = |p_1\rangle|p_2\rangle|p_3\rangle \ldots$$

т.к. частицы не взаимодействуют. Суммарная энергия равна сумме логарифмов.

Функция Мёбиуса

- Так вот, это всё разумно, если частицы являются бозонами (частицами с целым спином: фотон, мезоны).
- А если мы хотим рассмотреть газ из фермионов (частиц с половинным спином: протоны, электроны), то возникает *принцип Паули*: два тождественных фермиона не могут находиться в одном квантовом состоянии.
- О чём это говорит? Правильно, о том, что в разложении целого числа запрещаются квадраты простых.
- Оператор, который разделяет бозоны и фермионы — как раз функция Мёбиуса.

Функция Мёбиуса

- Кстати, эта модель риманова газа вообще очень интересна математически.
- Помните нормировочную константу Z? Она в физике называется *статистической суммой* (partition function) и играет важную роль, через неё выражаются термодинамические параметры всякие.
- А в модели риманова газа Z будет в точности дзета-функцией Римана...
- Но вернёмся к функции Мёбиуса.

Функция Мёбиуса

- Обобщим функцию Мёбиуса с целых чисел.
- Для всякого локально конечного чума и коммутативного кольца с единицей R можно определить *алгебру инцидентности*.
- Её элементы — функции f, которые действуют на множестве интервалов: $f([a,b]) \in R$.
- Сложение и умножение на скаляр будут покомпонентными, умножение функций — свёртка

$$(f * g)(a,b) = \sum_{a \leq x \leq b} f(a,x)g(x,b).$$

- Какая будет единица в этой алгебре?

Функция Мёбиуса

- Единица — это дельта-функция:

$$\delta(a, b) = \begin{cases} 1, & a = b, \\ 0, & a < b. \end{cases}$$

- Ещё одна интересная функция — дзета-функция $\zeta(a, b) = 1$ (т.е. константа).
- Умножение на ζ — это вроде интегрирования.

Функция Мёбиуса

- Так вот, ζ, оказывается, можно обратить. Обратная к функции ζ — это и есть функция Мёбиуса μ.
- Классическая функция Мёбиуса на числах получается, если рассмотреть чум натуральных чисел, упорядоченных по делимости.

Упражнение. Проверьте!

Функция Мёбиуса

- А если взять натуральные числа с нормальным порядком, получится разностный оператор Δ:

$$\mu(x,y) = \begin{cases} 1, & y - x = 0, \\ -1, & y - x = 1, \\ 0, y - x > 1. \end{cases}$$

- Как видите, функция Мёбиуса — это действительно вроде дифференцирования.

Функция Мёбиуса

- А если у функций приравнять значения $f(a,b) = f(a+c, b+c)$, останутся последовательности элементов R.
- И получится кольцо формальных степенных рядов с коэффициентами из R (свёртка двух функций — в точности произведение рядов).
- При таком подходе δ — это формальный ряд $(1, 0, 0, \ldots)$, т.е. 1 ($\delta(a,b) = 1$ iff $a = b$).
- ζ — это формальный ряд $(1, 1, 1, \ldots)$, т.е. $1 + x + x^2 + x^3 + \ldots$.

Функция Мёбиуса

- Но мы же все умеем обращать такой ряд: $1 + x + x^2 + \ldots = \frac{1}{1-x}$.
- Т.е. μ — это ряд вида $(1, -1, 0, 0, \ldots)$ — ровно так, как получалось два слайда назад.
- Вот такая чудесная функция Мёбиуса.

Функция Мёбиуса

- Но нам нужен немножко другой пример — булева алгебра подмножеств некоторого множества X.
- Тогда (доказывать не будем) функция Мёбиуса $\mu(T, S) = (-1)^{S \setminus T}$.
- Это называется *принцип включения–исключения*:

$$g(S) = \sum_{T \subseteq S} f(T) \quad \Leftrightarrow \quad f(S) = \sum_{T \subseteq S} (-1)^{S \setminus T} g(T).$$

- Примерно такой будет и функция Мёбиуса на нашем чуме из регионов.
- Всё, лирическое отступление закончилось.

Аппроксимация по Кикучи

- Вернёмся к нашим маргинализациям.
- Итак, мы приблизили энтропию:

$$H(b(\mathbf{x})) \approx \sum_{r \in R} k_r H_r(b_r(\mathbf{x}_r)).$$

- Что дальше?

Аппроксимация по Кикучи

- Определим Δ_R — семейство R-маргиналов всех вероятностных распределений.
- Т.е. $\{b_r(\mathbf{x}_r)\} \in \Delta_R$ iff $\{b_r(\mathbf{x}_r)\}$ являются маргинальными распределениями некоторого $b(\mathbf{x})$.
- Вспомним задачу минимизации:

$$F_0 = \min_{b(\mathbf{x})} F(b(\mathbf{x})) = F(B(\mathbf{x})) = -\log Z.$$

Аппроксимация по Кикучи

- Теперь её можно переписать как

$$F_0 = \min_{\{b_r(\mathbf{x}_r)\} \in \Delta_R} F_R(\{b_r(\mathbf{x}_r)\}),$$

$$\{b_r^*(\mathbf{x}_r)\} = \mathrm{argmin}_{\{b_r(\mathbf{x}_r)\} \in \Delta_R} F_R(\{b_r(\mathbf{x}_r)\}), \text{ где}$$

$$F_R(\{b_r(\mathbf{x}_r)\}) = \min_{b(\mathbf{x}):\{b_r\} \text{ -- маргиналы } b} F(b(\mathbf{x})).$$

- Теперь давайте всё сведём воедино.

Аппроксимация по Кикучи

- Получилась задача минимизации

$$\{B_r(\mathbf{x}_r)\} \approx \{b_r^*(\mathbf{x}_r)\} = \mathrm{argmin}_{\{b_r(\mathbf{x}_r)\}\in\Delta_R^K} F_R^K(\{b_r(\mathbf{x}_r)\}), \text{ где}$$

$$F_R^K(\{b_r(\mathbf{x}_r)\}) = \sum_{r\in R}\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)E_r(\mathbf{x}_r) + \sum_{r\in R}\sum_{\mathbf{x}_r} k_r b_r(\mathbf{x}_r)\log(b_r(\mathbf{x}_r)),$$

$$\Delta_R^K = \{\{b_r(\mathbf{x}_r), r\in R\} :$$

$$\forall t, u \in R, t \subset u, \quad \sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) = b_t(\mathbf{x}_t),$$

$$\forall u \in R, \sum_{\mathbf{x}_u} b_u(\mathbf{x}_u) = 1\}.$$

Аппроксимация по Кикучи

- То есть задача минимизации F_R^K при ограничениях Δ_R^K (которые задают совместимость b_r).
- F_R^K называется *свободной энергией Кикучи*.
- Задач осталось две: как-нибудь постараться сделать так, чтобы F_R^K было как можно больше похоже на F, и чтобы Δ_R^K было похоже на Δ_R.
- За счёт чего? Вроде бы не осталось степеней свободы.

Аппроксимация по Кикучи

- На самом деле немножко ещё осталось. Давайте рассмотрим множество регионов R_0 (исходное) и ядер $\alpha_r^0(\mathbf{x}_r)$, $r \in R^0$.
- Мы бы хотели решить задачу маргинализации для R_0 и $\{\alpha_r^0\}$.
- Введём новое множество регионов R так, чтобы

$$\forall r' \in R^0 \exists r \in R \quad r' \supseteq r.$$

Аппроксимация по Кикучи

- Мы тогда сможем построить новое множество ядер $\{\alpha_r(\mathbf{x}_r), r \in R\}$ так, чтобы

$$-\sum_{r \in R} \log(\alpha_r(\mathbf{x}_r)) = -\sum_{r \in R^0} \log(\alpha_r^0(\mathbf{x}_r)) = E(\mathbf{x}).$$

- И если мы теперь определим $\beta_r = \prod_{s \subset r} \alpha_s(\mathbf{x}_s)$, то

$$B(\mathbf{x}) = \frac{\prod_{r \in R^0} \alpha_r^0(\mathbf{x}_r)}{Z} = \frac{\prod_{r \in R} \alpha_r(\mathbf{x}_r)}{Z} = \frac{\prod_{r \in R} \beta_r(\mathbf{x}_r)^{k_r}}{Z},$$

потому что $\sum_{r \subset s} k_r = 1$ для всех $s \in R$.

Аппроксимация по Кикучи

- То есть мы можем модифицировать набор регионов, на котором проводим оптимизацию.
- От этого может измениться качество приближения, хотелось бы оптимизировать.
- Но эти задачи мы решать сейчас не будем.

Спасибо за внимание!

- Lecture notes и слайды будут появляться на моей homepage:
  http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching
- Присылайте любые замечания, решения упражнений, новые численные примеры и прочее по адресам: sergey@logic.pdmi.ras.ru, snikolenko@gmail.com
- Заходите в ЖЖ smartnik.