Маргинализация грубой силой. Кодирование

Сергей Николенко

Machine Learning — CS Club, весна 2008

Outline

Суть

- Пусть нам, как обычно, нужно понять, какая гипотеза лучше других описывает имеющиеся данные.
- Предлагается алгоритм: перечислить все гипотезы и сравнить их правдоподобия (likelihoods).
- Мы сначала рассмотрим, как этот метод работает в дискретном булевском случае, а затем в непрерывном случае нормального распределения.
- Тем самым мы ещё раз поймём, в чём наша основная цель.

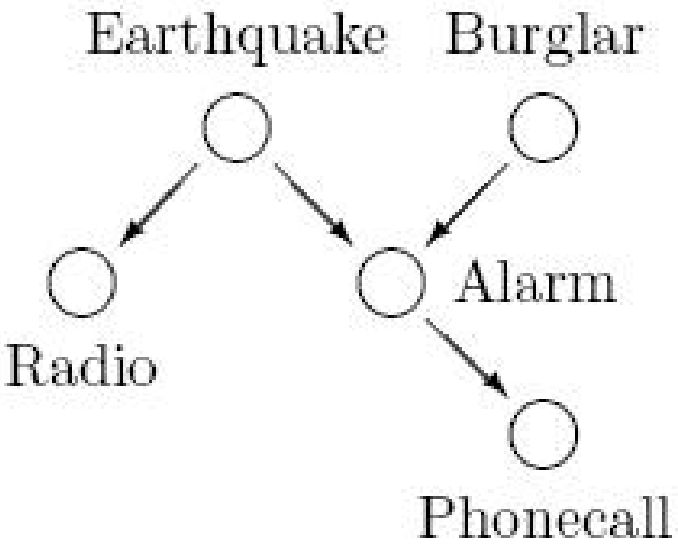
Байесовские сети

- Байесовская сеть — направленный граф, в котором стрелки показывают причинно-следственную связь.
- У нас были разработаны алгоритмы вывода на байесовских сетях, но сейчас мы будем действовать грубой силой.
- Чтобы решить байесовскую сеть полным перебором, нужно представить её в виде большого произведения всех вероятностей, которые в ней участвуют, а затем *маргинализовать* по вероятностям, которые нам известны.

Пример: разработаем сеть

- Ситуация: Вася поехал на работу, и тут вдруг ему звонит сосед и говорит, что у его дома звонит сигнализация против грабителей.
- Вася уже было возвращается, но тут слышит по радио, что рядом с его домом было микроземлетрясение.
- Вася знает, что вполне возможно, что микроземлетрясение само собой вызвало срабатывание сигнализации.
- Какая должна быть модель такой ситуации?

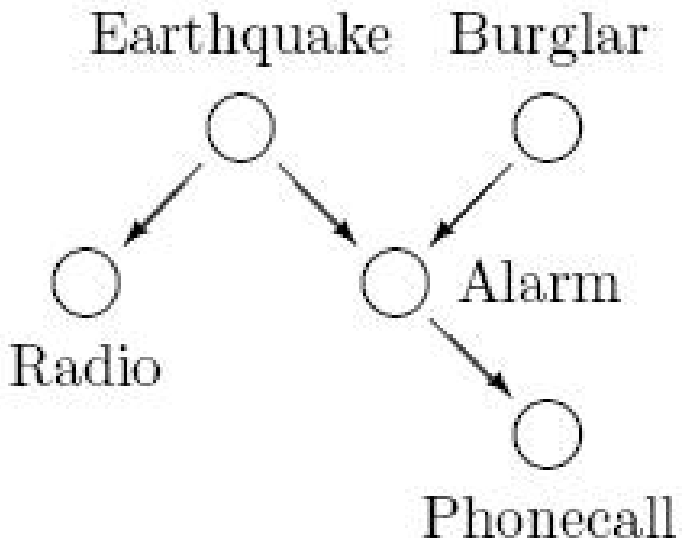
Пример



- Вот несложная сеть (даже без циклов), описывающая затруднительное положение Васи.
- Совместная вероятность всей сети:

$$p(b, e, a, p, r) = p(b)p(e)p(a|b, e)p(p|a)p(r|e).$$

Вероятности



- $p(b = 1) = \beta = 0.001$;
- $p(e = 1) = \epsilon = 0.001$;
- $p(p = 1|a = 0) = 0$,
  $p(p = 1|a = 1) = 1$;
- $p(r = 1|e = 0) = 0$,
  $p(r = 1|e = 1) = 1$.

Вероятности: Noisy-OR

- Осталось специфицировать распределение $p(a|b,e)$. Давайте предположим, что существует некая вероятность α_b того, что сигнализация сработает на грабителя, вероятность α_e того, что сигнализация сработает на землетрясение, и вероятность α_f просто ложного срабатывания. То есть на самом деле

$$\mathrm{alarm} = \mathrm{burglar} \vee \mathrm{earthquake} \vee \mathrm{false\_alarm},$$

  но не точно логически, а с некоторыми вероятностями.
- Такая ситуация называется Noisy-OR (есть ещё аналогичный Noisy-AND).
- Какие тогда будут условные вероятности?

Вероятности: Noisy-OR

$$
\begin{aligned}
p(a=1|b=0,e=0) &= \alpha_f, \\
p(a=1|b=1,e=0) &= 1-(1-\alpha_f)(1-\alpha_b), \\
p(a=1|b=0,e=1) &= 1-(1-\alpha_f)(1-\alpha_e), \\
p(a=1|b=1,e=1) &= 1-(1-\alpha_f)(1-\alpha_b)(1-\alpha_e).
\end{aligned}
$$

Например, при $\alpha_f = 0.001$, $\alpha_e = 0.01$ и $\alpha_b = 0.99$

$$
\begin{aligned}
p(a=1|b=0,e=0) &= 0.001, \\
p(a=1|b=1,e=0) &= 0.99001, \\
p(a=1|b=0,e=1) &= 0.01099, \\
p(a=1|b=1,e=1) &= 0.9901099.
\end{aligned}
$$

Собственно вывод

- Теперь давайте проводить маргинализацию.
- В первой ситуации мы знаем, что нам позвонили, и хотим выяснить распределение вероятностей визита грабителя и землетрясения, т.е. найти $p(b, e|p = 1)$.
- Используем теорему Байеса:

$$p(b, e|p = 1) = \frac{p(p = 1|b, e)p(b)p(e)}{p(p = 1)}$$

и маргинализуем $p(p = 1|b, e)$ и $p(p = 1)$ из нашей сети:

$$p(b, e|p = 1) = \frac{\sum_a p(p = 1|a)p(a|b, e)p(b)p(e)}{\sum_{a,b,e} p(p = 1|a)p(a|b, e)p(b)p(e)}.$$

Вывод cont'd

- В итоге получается

$$\begin{aligned} p(b=0, e=0|p=1) &= 0.4993, \\ p(b=1, e=0|p=1) &= 0.4947, \\ p(b=0, e=1|p=1) &= 0.0055, \\ p(b=1, e=1|p=1) &= 0.0005. \end{aligned}$$

- То есть вероятность реального грабителя — около 50%.
- А если пересчитать с учётом события $r=1$, то получится

$$\begin{aligned} p(b=0|p=1, r=1) &= 0.92, \\ p(b=1|p=1, r=1) &= 0.08. \end{aligned}$$

- Вот поэтому Вася и может успокоиться.

Домашнее задание

Упражнение. Разработать и обсчитать ещё один аналогичный пример, но такой, чтобы в нём фигурировал Noisy-AND.

Суть

- Пусть мы хотим найти скрытые параметры, например, нормального распределения методом полного перебора.
- Как это сделать, ведь параметры непрерывные, всех не переберёшь?
- Можно просто сделать пространство дискретным, перебрать параметры с каким-то шагом.
- Мы уже выясняли на лекции по сэмплингу, что это не выход, но в качестве упражнения можно попробовать.

Важное замечание

- Когда мы проводим байесовский вывод, у нас, кроме правдоподобия, должно быть ещё *априорное распределение* (prior distribution) по всем возможным значениям параметров.
- Мы будем подсчитывать только правдоподобия, т.е. предполагать, что априорное распределение равномерное на интервале, который мы дискретизуем.
- Позже мы рассмотрим более разумные априорные распределения.

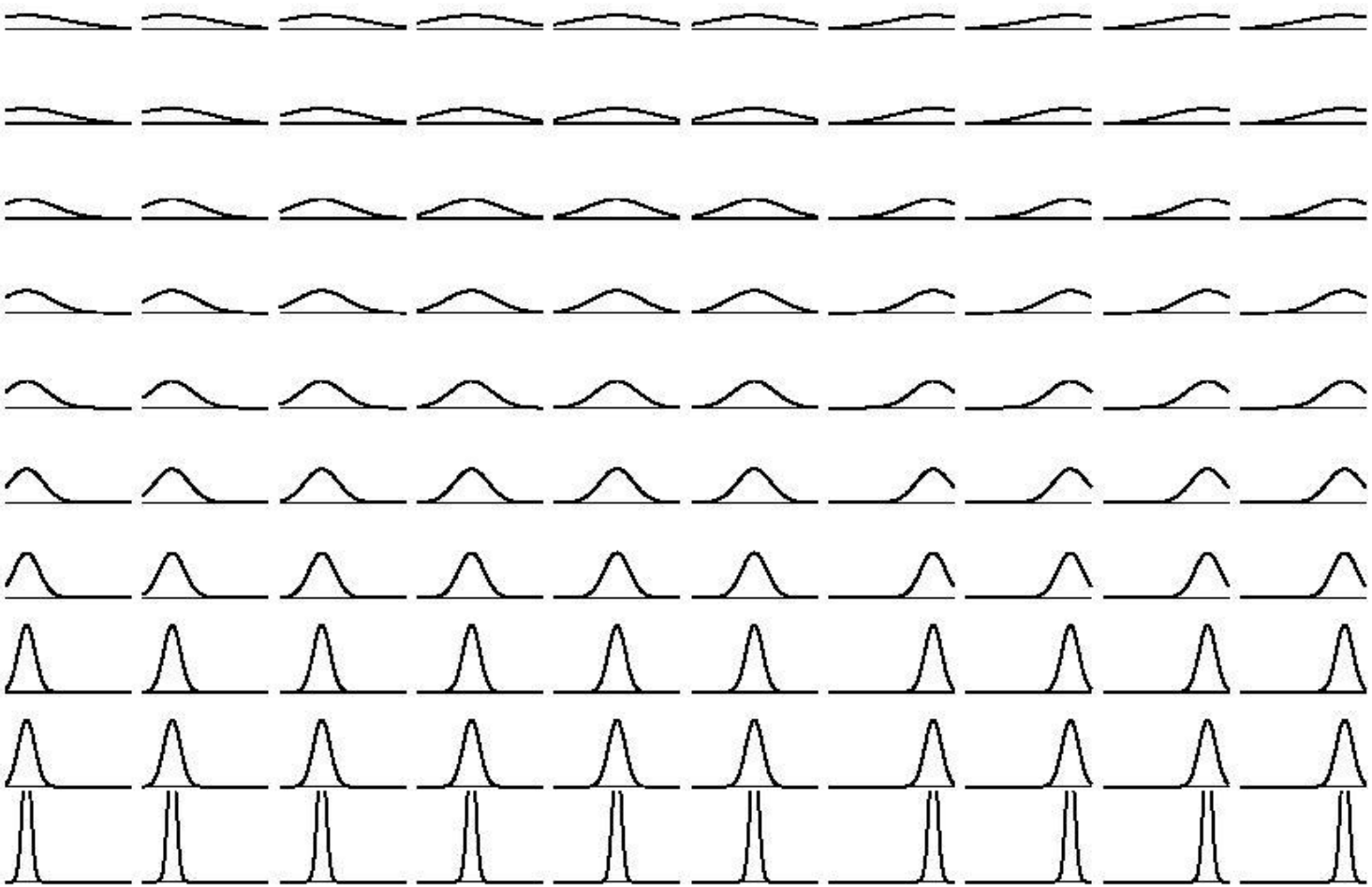
Нормальное распределение

- Возьмём нормальное распределение:

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

- У него два параметра, по которым можно перебирать.
- То есть алгоритм будет просто перебирать параметры μ и σ и подсчитывать функцию правдоподобия $p(\{x_i\}|\mu,\sigma)$.

Нормальное распределение



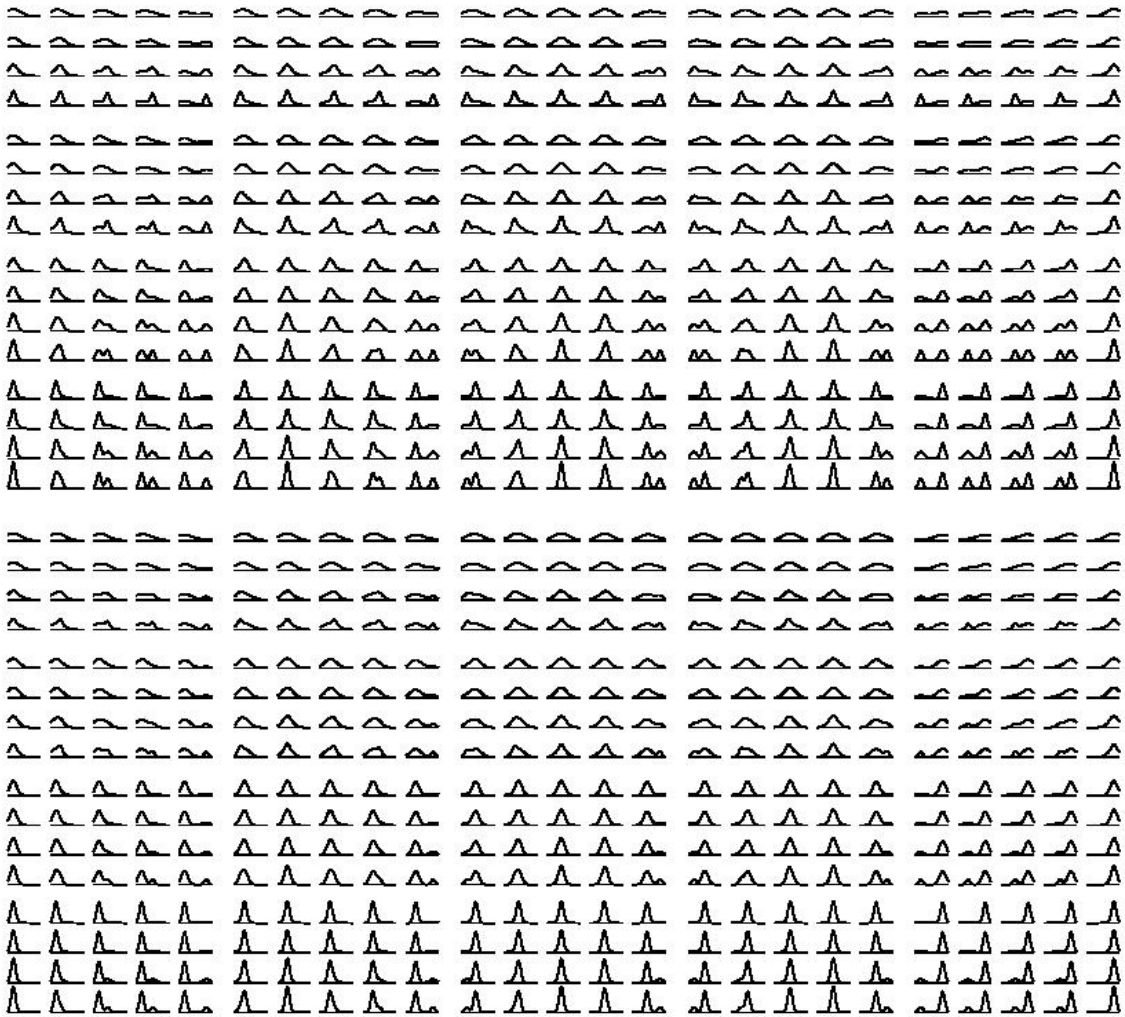
Смесь нормальных распределений

- Более сложный случай — когда распределение представляет собой смесь гауссианов, которые берутся с весами α_1 и α_2, $\alpha_1 + \alpha_2 = 1$:

$$p(x|\mu_1, \sigma_1, \alpha_1, \mu_2, \sigma_2, \alpha_2) = \\ = \frac{\alpha_1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{\alpha_2}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}.$$

- Тут уже... сколько параметров?

Смесь нормальных распределений



Сопряжённые априорные распределения

- Мы занимаемся байесовским выводом, т.е. в непрерывном случае решаем задачу поиска скрытых параметров:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}.$$

- Чтобы вести вывод, нужно выбирать априорные распределения $p(\theta)$. Как это делать? Какова может быть цель?

Сопряжённые априорные распределения

- Разумная цель: давайте будем выбирать распределения так, чтобы они оставались такими же и *a posteriori*.
- До начала вывода есть априорное распределение $p(\theta)$.
- После него есть какое-то новое апостериорное распределение $p(\theta|x)$.
- Я хочу, чтобы $p(\theta|x)$ тоже имело тот же вид, что и $p(\theta)$, просто с другими параметрами — $p(\theta|x) = p(\theta')$.

Сопряжённые априорные распределения

- Разумеется, вид такого хорошего априорного распределения зависит от вида распределения, которым накидываются собственно данные.
- Такое p называется *сопряжённым априорным распределением* (conjugate prior).
- Conjugate priors подсчитаны для многих распределений.
- Вот, например, каким будет сопряжённое априорное распределение для бросания нечестной монетки (испытаний Бернулли)?

Сопряжённые априорные распределения

- Ответ: это будет бета-распределение; плотность распределения нечестности монетки q

$$p(q = x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}.$$

- Тогда, если мы посэмплируем монетку, получив s орлов и f решек, получится

$$p(s, f \mid q = x) = \binom{s+f}{s} x^s (1-x)^f, \text{ и}$$

$$p(q = x|s, f) = \frac{\binom{s+f}{s} x^{s+\alpha-1}(1-x)^{f+\beta-1}/B(\alpha, \beta)}{\int_0^1 \binom{s+f}{s} y^{s+\alpha-1}(1-y)^{f+\beta-1}/B(\alpha, \beta)\,dy} =$$

$$= \frac{x^{s+\alpha-1}(1-x)^{f+\beta-1}}{B(s+\alpha, f+\beta)}.$$

Сопряжённые априорные распределения

- Для параметра μ нормального распределения с фиксированной дисперсией сопряжённое априорное распределение — это тоже нормальное распределение.
- При получении сэмплов параметры этого распределения будут меняться вот как:

$$(\mu_0, \sigma_0) \to \left(\frac{\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}, \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} \right).$$

Сопряжённые априорные распределения

- А при фиксированном μ для σ, точнее, для $\beta = 1/\sigma^2$, естественным априорным распределением будет гамма-распределение:

$$p(\beta|k_\beta, \theta_\beta) = \beta^{\theta_\beta - 1} \frac{e^{-\beta/k_\beta}}{k_\beta^{\theta_\beta} \Gamma(\theta_\beta)}, \quad \beta > 0.$$

- Но мы увлеклись.

Маргинализация

- Вспомним, что маргинализация — это суммирование по некоторым переменным так, чтобы их из произведения изгнать.
- Маргинализация — основа байесовского вывода, главный (и самый вычислительно сложный) инструмент.
- Когда мы делали вывод грубой силой, мы проводили маргинализацию, суммируя по всем возможным значениям, а для непрерывных переменных рассматривали все возможные значения с некоторым шагом.
- Но ведь можно и просто взять определённый интеграл. Этим мы сейчас и займёмся.

Неправильные априорные распределения

- Мы будем пользоваться не сопряжёнными распределениями, а их предельными случаями.
- Для μ будем рассматривать $p(\mu) = const$; это совершенно неправильное распределение.
- Для σ будем рассматривать предел при $k_\beta \theta_\beta = 1$, $\theta_\beta \to 0$, т.е. плоское распределение $\ln \sigma$ (он же $\ln \beta$).
- Это и не распределения вовсе (не интегрируются); так и называются — improper priors. Но для простоты их часто используют.

Оценки параметров

- Пусть есть данные $D = \{x_i\}_{i=1}^n$. Тогда оценка среднего

$$\bar{x} = \sum_{i=1}^{n} x_i / n,$$

а оценка дисперсии —

$$\sigma_{n-1} = \sqrt{\frac{\sum_{i=1}^{n} (\bar{x} - x_n)^2}{n-1}}.$$

Оценки параметров

- Мы уже доказывали, что $(\bar{x}, \sigma_n)$ — гипотеза максимального правдоподобия.
- Теперь давайте попробуем найти апостериорное распределение μ при данном σ:

$$p(\mu|\{x_i\}_{i=1}^n, \sigma) = \frac{p(\{x_i\}_{i=1}^n|\mu, \sigma)p(\mu)}{p(\{x_i\}_{i=1}^n|\sigma)} = Ce^{-n(\mu-\bar{x})^2/(2\sigma^2)}.$$

- То есть получили нормальное распределение на μ с параметрами $(\bar{x}, \sigma^2/n)$.

Маргинализация

- Теперь — собственно задача маргинализации.
- Пусть мы хотим найти наиболее вероятную σ при имеющихся данных.
- Это значит, что нам придётся маргинализировать μ из данных, когда мы будем подсчитывать

$$p(\sigma|\{x_i\}_{i=1}^n) = \frac{p(\{x_i\}_{i=1}^n|\sigma)p(\sigma)}{p(\{x_i\}_{i=1}^n)}.$$

Маргинализация

- Здесь мы, как и раньше, предполагаем, что $p(\mu) = 1/\sigma_\mu = const$. Тогда

$$p(\{x_i\}_{i=1}^n|\sigma) = \int p(\{x_i\}_{i=1}^n|\sigma,\mu)p(\mu)d\mu =$$

$$= \frac{1}{\sigma_\mu}\int \frac{1}{\sigma\sqrt{2\pi}} e^{-\sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2}} d\mu,$$

и

$$\ln p(\{x_i\}_{i=1}^n|\sigma) = -n\ln(\sqrt{2\pi}\sigma) - \frac{\sum_{i=1}^n (\bar{x}-x_n)^2}{2\sigma^2} + \ln\frac{\sqrt{2\pi}\sigma/\sqrt{n}}{\sigma_\mu}.$$

- За счёт последнего члена (так называемого *фактора Оккама* — мы эти факторы ещё обсудим) максимум и сдвигается с σ_n на σ_{n-1}.

Краткий итог

- Мы вычислили один простой интеграл, которым маргинализовали одну из переменных.
- В этом суть точной маргинализации с непрерывными переменными: нужно проводить точное интегрирование (а как иначе...).
- Интеграл был такой простой, потому что мы предполагали очень простые (неправильные) априорные распределения.

Упражнение. Провести такую же маргинализацию для настоящих сопряжённых априорных распределений.

Что будет дальше

- А сейчас мы перейдём к реально применяемым алгоритмы.
- Мы рассмотрим точную маргинализацию на решётках (trellises), причём не просто так, а в применении к задачам декодирования коррекционных кодов (кодов, исправляющих ошибки). Заодно и про сами коды поговорим.

Outline

Суть

- Что такое коды, исправляющие ошибки (error–correcting codes)?
- Это коды, которые умеют даже по неправильному кодовому слову достаточно часто выдавать правильное сообщение.
- Формально — какая задача стоит перед декодером?

Задача декодирования

- *Задача полного декодирования*: по сигналу понять, какое кодовое слово передавалось.
- *Задача побитового декодирования*: для каждого передаваемого бита t_n понять, насколько вероятно, что это был 0 или 1.

Задача декодирования

- Обозначим: t — кодовое слово, y — полученный сигнал.

$$p(t|y) = \frac{p(y|t)p(t)}{p(y)}.$$

- Если канал не имеет памяти, то $p(y|t) = \prod_{i=1}^{n} p(y_i|t_i)$.
- Априорное распределение $p(t)$ считаем равномерным (почему?).
- Знаменатель $p(y) = \sum_t p(y|t)p(t)$.

Решение задачи декодирования

- Полное решение — список всех кодовых слов и их вероятностей при условии данного сигнала.
- Но на самом деле нам столько не нужно, нам нужно просто найти самое вероятное кодовое слово.
- *Задача декодирования с максимальным правдоподобием* (MAP codeword decoding problem) — задача поиска наиболее вероятного кодового слова t при условии данного сигнала y. При равномерном априорном распределении эта задача сводится к максимизации правдоподобия $p(y|t)$.

Решение задачи побитового декодирования

- Задачу побитового декодирования можно решить маргинализацией:

$$p(t_i = 1|y) = \sum_{t:t_i=1} p(t|y) = \sum_t [t_i = 1] p(t|y),$$

$$p(t_i = 0|y) = \sum_{t:t_i=0} p(t|y) = \sum_t [t_i = 0] p(t|y).$$

- Мы только что научились решать эту задачу полным перебором. Но в байесовских сетях мы уже умели делать это поумнее.
- Так будем и сейчас, но сначала придётся вспомнить о том, что же такое кодирование.

Идея кодов, исправляющих ошибку

- Есть канал, который может портить передаваемые сигналы.
- Нужно научиться передавать сообщения так, чтобы получатель мог расшифровать их, даже если случится где-то ошибка.
- Как это сделать?

Повторяющий код

- Простейший код — повторять каждый бит три раза; повторяющий код (repetition code).
- Тогда, даже если одна ошибка в тройке идущих подряд битов попадётся, мы всё равно раскодируем правильно, взяв большинство ответов.
- Защищает от однократной ошибки, не защищает от двух ошибок в одной тройке битов.

Упражнение. Докажите, что решение большинством голосов в этом случае реализует максимальную апостериорную гипотезу, если вероятность ошибки в одном бите меньше $1/2$.

Недостатки повторяющего кода

- Главный недостаток — слишком большой overhead; неэффективно.
- При n повторениях вероятность ошибки равна

$$\sum_{i=(n+1)/2}^{n} \binom{n}{i} \alpha^i (1-\alpha)^{n-i},$$

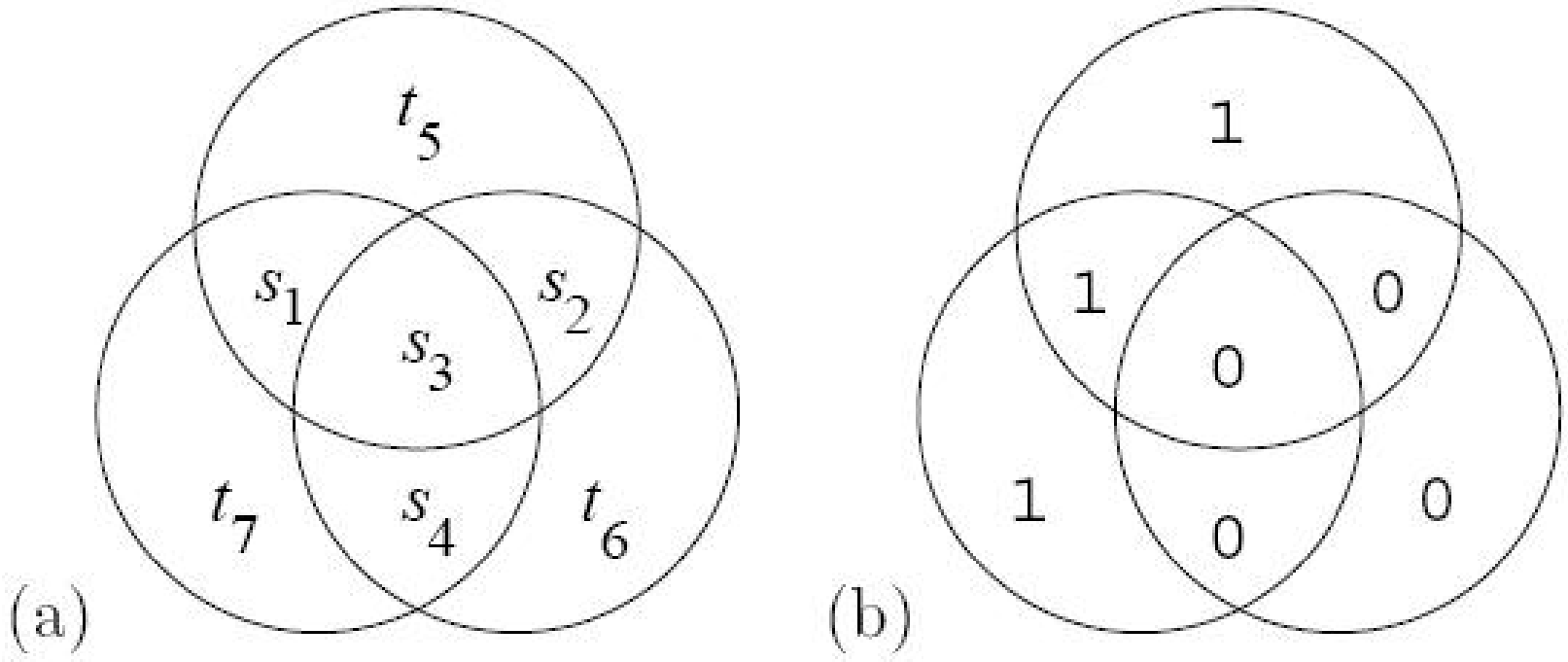
где α — вероятность ошибки в одном бите.

Линейные коды

- Обобщим. Пусть у нас блок размера k переходит в блок размера n при кодировании ($n > k$, разумеется).
- Предположим, что первые k бит кодового слова — это просто само сообщение, а оставшиеся $n - k$ — линейные функции от битов сообщения (parity checks).
- Такие коды называются *линейными*.

Пример

- Широко известен код Хэмминга $(7,4)$ (на 4 бита сообщения 7 битов сигнала).
- Линейные функции — сумма битов сообщения, попадающих в соответствующий круг.



Пример

- Главное свойство этого кода — то, что кодовые слова отличаются друг от друга как минимум в трёх битах. То есть мы достигли того же эффекта, что и с троекратными повторениями, но битов стало больше не в 3, а в $\frac{7}{4}$ раза.
- На самом деле не вполне того же самого; в чём разница?
- Как декодировать?

Линейный код в общем виде

- Кодовое слово получается в виде $t = G^t s$, где G — матрица, называющаяся *генератором* кода.
- Например, для $(7, 4)$–кода Хэмминга

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}.$$

- В нашем определении у G всегда будет сверху единичная подматрица.

Синдромы

- Для декодирования используются так называемые *синдромы*. Синдром — это разница между реальным сигналом и сигналом, вычисленным на основании полученных битов сообщения.
- Если $t = G^t s$, и $G = \begin{bmatrix} I_k \\ P \end{bmatrix}$, то синдром $z = Hr$, где $H = [-P\ I_{n-k}]$. Например, для $(7,4)$–кода Хэмминга

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

- Для всякого валидного кодового слова t $Ht = 0$. Докажите!

Постановка задачи декодирования

- Получаемый вектор r — это сумма кодового слова и шума:

$$r = G^t s + n.$$

- Задача декодирования синдрома — это задача поиска наиболее вероятного вектора шума n, удовлетворяющего уравнению

$$Hn = z.$$

- Её-то мы и будем решать.

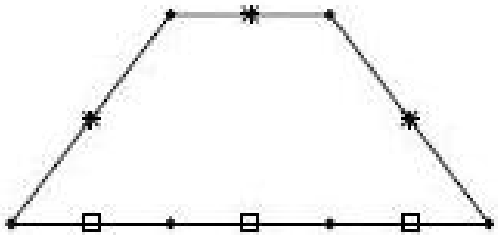
Outline

Решётки

- Поставим коду в соответствие его решётку (trellis). Решётка — это такой граф, вершины которого разделены на несколько групп (*времён*, times), причём каждое ребро соединяет вершину из времени i с вершиной времени $i-1$ или $i+1$. Крайнее левое и крайнее правое времена имеют по одной вершине.
- Такая решётка определяет код: кодовое слово соответствует пути из левого в правое время.
- Если код линейный, решётка тоже линейная; отныне у нас все решётки линейные.

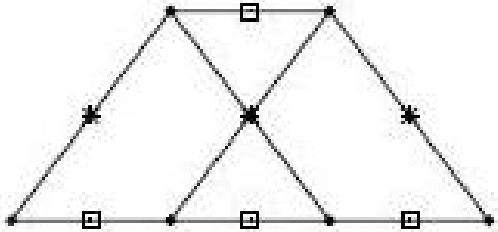
Примеры

(a)



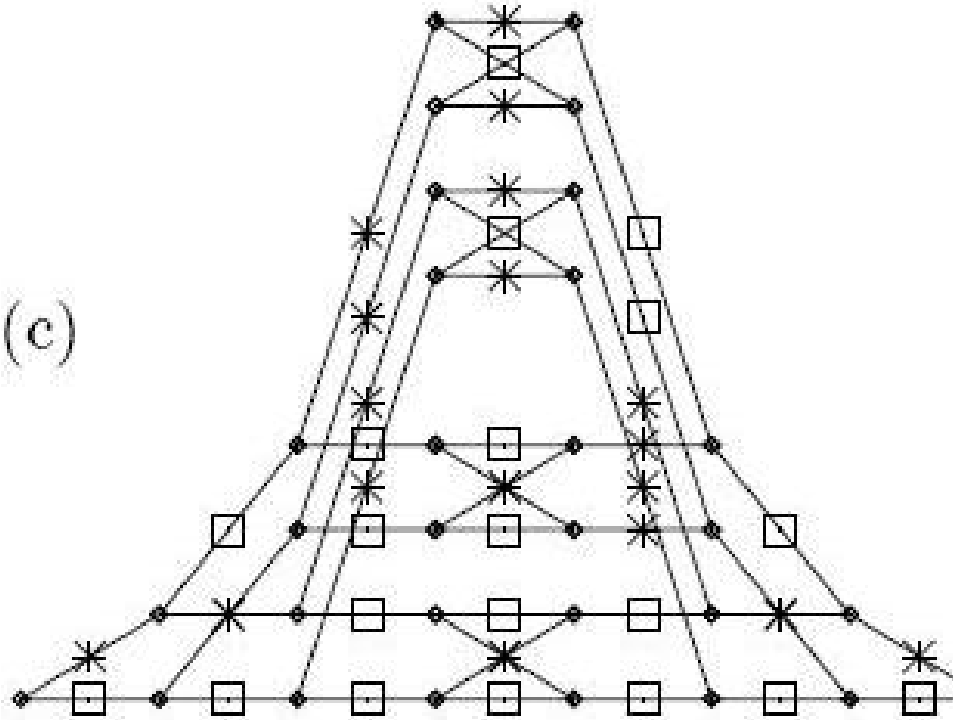
Repetition code R_3

(b)



Simple parity code P_3

(c)



(7, 4) Hamming code

Решётки и кодовые слова

- Решётку можно рассматривать как модель вероятностного процесса, которым получается кодовое слово.
- Каждое разветвление решается случайным образом (по текущему биту сообщения).
- Задача теперь превращается в такую: дана последовательность меток на рёбрах с шумом, а найти нужно наиболее вероятную исходную последовательность или наиболее вероятное значение того или иного бита.

Декодирование как поиск MAP

- Обозначим y — то, что мы получили, t — то, что хотели передать. Тогда

$$p(t|y) = \frac{p(y|t)p(t)}{p(y)}.$$

- Будем предполагать, что априорное распределение $p(t)$ равномерно (кодовые слова появляются с равной вероятностью).
- Этого, кстати, всегда можно достичь при помощи разумного кодирования.

Веса и правдоподобие

- Так как ошибки независимы, $p(y|t) = \prod_{i=1}^{n} p(y_i|t_i)$. Значит, $\log p(y|t) = \sum_{i=1}^{n} \log p(y_i|t_i)$.
- И нам нужно максимизировать эту сумму. Поменяв знак, будем минимизировать сумму величин $-\log p(y_i|t_i)$.
- Для этого просто снабдим рёбра весами $-\log p(y_i|t_i)$ и будем искать путь с минимальным весом. Как?

Min-sum algorithm

- Будем использовать передачу сообщений между узлами.
- Каждый узел, как только он получит сообщение от всех своих предков, может передать сообщение о своей цене всем своим потомкам, с учётом нового задействованного ребра.
- Когда этот процесс закончится, все узлы будут знать свою минимальную стоимость, в том числе и целевой узел.
- Это и есть min-sum algorithm, или алгоритм Витерби (Viterbi).

Побитовое декодирование

- Это мы решали задачу декодирования вообще, т.е. поиска кодового слова.
- Теперь давайте решим задачу декодирования побитового, т.е. поиска наиболее вероятного данного бита.
- Для этого придётся от min-sum перейти к min-product; т.е. считать правдоподобия $p(y_n|t_n)$, а не их логарифмы. И вместо поиска минимума будем суммировать.
- Но теперь понадобится два прохода. Рассмотрим их подробнее.

Прямой проход

- Сообщение на первом шаге $\alpha_0 = 1$, сообщение на очередном шаге
$$\alpha_i = \sum_{j \in \mathrm{pa}(i)} w_{ij} \alpha_j.$$
- В итоге первого прохода сообщение, полученное каждым из узлов i с временной координатой l — это совместная вероятность того, что путь кодового слова прошёл через этот узел, а первые l символов кодового слова были $y_1, \ldots, y_l$ (с точностью до константы).
- Это — *прямой проход* алгоритма (forward pass).

Обратный проход

- Другой набор сообщений в это время отправляется справа налево.
- Сообщение на первом шаге $\beta_0 = 1$, сообщение на очередном шаге
$$\beta_j = \sum_{i \in \mathrm{ch}(j)} w_{ij} \beta_i .$$
- А эти сообщения пропорциональны *условным* вероятностям того, что, при условии что путь кодового слова прошёл через вершину i, последующие символы были равны $y_{l+1}, \ldots, y_n$.

Последний шаг

- Итого, чтобы найти вероятность того, что n–й бит был равен 0 или 1, нужно подсчитать две суммы. Пусть i пробегает узлы времени n, j — узлы времени $n-1$, t_{ij} — значение t_n, стоящее на ребре решётки от узла j к узлу i. Тогда

$$r_n^{(0)} = \sum_{i,j:j\in \mathrm{pa}(i),t_{ij}=0} \alpha_j w_{ij} \beta_i, \quad r_n^{(1)} = \sum_{i,j:j\in \mathrm{pa}(i),t_{ij}=1} \alpha_j w_{ij} \beta_i.$$

- Эти суммы и дадут искомые вероятности; нужно только их нормализировать (разделить на $r_n^{(0)} + r_n^{(1)}$).

Замечания

- Отметим, что оба эти алгоритма распределённые — в узлах могут стоять независимые машины, причём довольно слабые.
- А ещё отметим, что min-product — это в точности тот алгоритм, который мы использовали для вывода на байесовских сетях.

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**.