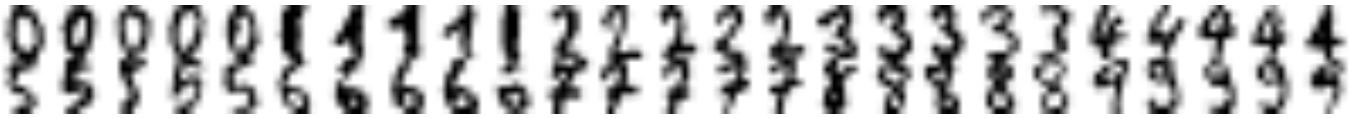
Школа анализа данных

Задание 3. Процессы Дирихле для кластеризации изображений цифр

Курс: Байесовские методы в машинном обучении, осень 2015



Начало выполнения задания: 30 ноября
Срок сдачи: **13 декабря (воскресенье), 23:59.**
Среда для выполнения задания – PYTHON 3.x.

Описание модели

Рассмотрим вероятностную модель смеси распределений с априорным процессом Дирихле:

$$\begin{aligned} &G \sim \mathrm{DP}(\alpha, H), \\ &\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_N \sim G, \\ &\boldsymbol{x}_n \sim p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_n),\ n = \overline{1, N}. \end{aligned} \tag{1}$$

Здесь $\alpha > 0$ – параметр концентрации, H – базовая вероятностная мера, G – вероятностная атомическая мера, $\boldsymbol{x}_n$ – наблюдаемые данные, $\hat{\boldsymbol{\theta}}_n$ – параметры компоненты смеси для объекта $\boldsymbol{x}_n$. В силу атомичности меры G некоторые компоненты $\hat{\boldsymbol{\theta}}_n$ совпадают между собой, формируя таким образом кластеры данных. Для удобства байесовского вывода модель (1) можно представить в эквивалентном виде с помощью процесса stick-breaking:

$$\begin{aligned} &p_G(\boldsymbol{\theta}) = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}), \\ &\boldsymbol{\theta}_k \sim p_H(\boldsymbol{\theta}),\ \pi_k = v_k \prod_{i=1}^{k-1}(1 - v_i),\ v_k \sim \mathrm{Beta}(v|1, \alpha), \\ &z_1, \ldots, z_N \sim \mathrm{Discrete}(\boldsymbol{\pi}), \\ &\boldsymbol{x}_n \sim p(\boldsymbol{x}|\boldsymbol{\theta}_{z_n}). \end{aligned} \tag{2}$$

Здесь $\delta(\cdot)$ – дельта-функция, $\boldsymbol{\theta}_k$ – атомы меры G, z_n – номер компоненты смеси для объекта $\boldsymbol{x}_n$. Совместное распределение всех переменных в модели (2) можно записать как

$$p(X, Z, v, \theta|\alpha, H) = \left[\prod_{k=1}^{\infty} p_H(\boldsymbol{\theta}_k)\,\mathrm{Beta}(v_k|1, \alpha)\right] \prod_{n=1}^{N}\prod_{k=1}^{\infty}\left(p(\boldsymbol{x}_n|\boldsymbol{\theta}_k) v_k \prod_{i=1}^{k-1}(1 - v_i)\right)^{[z_n = k]}. \tag{3}$$

Здесь через $[z_n = k]$ обозначен индикатор, равный 1, если $z_n = k$, и 0 иначе. Можно показать, что в модели (2) величины π_k с ростом k стремятся к нулю, а среднее количество значимо отличных от нуля π_k определяется выражением $\alpha \log(1 + N/\alpha)$. Таким образом, не ограничивая общности, в совместном распределении (3) максимальное число кластеров можно ограничить величиной $T = \mathrm{Const} \cdot \alpha \log(1 + N/\alpha)$, где Const – некоторая константа (например, 10).

Рассмотрим в качестве объектов $\boldsymbol{x}_n$ изображения рукописных цифр из выборки `Digits` (`sklearn.datasets.load_digits()`). В ней каждая цифра представлена черно-белым изображением размера 8×8 с градациями серого. Преобразуем все изображения в бинарные путём отсечения по порогу 8 (максимальное значение яркости пиксела в выборке равно 16) и вытянем каждое изображение в вектор $\boldsymbol{x}_n$ длины 64. В качестве одной компоненты смеси рассмотрим независимые распределения Бернулли:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{D} \theta_i^{x_i}(1 - \theta_i)^{1 - x_i}. \tag{4}$$

Здесь $x_i \in \{0, 1\}$ – i-ый пиксел изображения, $\theta_i \in (0, 1)$ – параметры компоненты. Из соображений сопряжённости в качестве априорного распределения для $\boldsymbol{\theta}$ возьмём независимое Бета-распределение с общими параметрами $a, b > 0$:

$$p_H(\boldsymbol{\theta}) = \prod_{i=1}^{D} \mathrm{Beta}(\theta_i|a, b). \tag{5}$$

Формулировка задания

Для модели (3) с компонентами смеси (4) и априорными распределениями (5) с помощью алгоритма вариационного вывода требуется найти факторизованное приближение для апостериорного распределения:

$$q(Z)q(\theta)q(v) \approx p(Z, \theta, v | X, \alpha, a, b).$$

Для выполнения задания необходимо:

1. Выписать формулы пересчёта компонент вариационного приближения $q(Z)$, $q(\theta)$, $q(v)$, формулу для вариационной нижней оценки $\mathcal{L}(q)$ и необходимые формулы для статистик компонент q.
2. Реализовать алгоритм вариационного вывода со стартом из нескольких случайных начальных приближений и выбором лучшего приближения q по максимальному значению $\mathcal{L}$. Реализовать отображение центров кластеров, найденных алгоритмом.
3. Протестировать полученный алгоритм на небольшой подвыборке `Digits`. Качественно охарактеризовать, как влияют параметры модели α, a, b на вид и количество образующихся кластеров. Выбрать конкретные значения α, a, b и использовать их во всех дальнейших экспериментах.
4. Запустить алгоритм на полной выборке `Digits`. Сколько и каких получилось кластеров? Рассмотреть величины $q(z_n = k)$ в качестве признаков n-го объекта выборки, обучить любой классификатор на образованных данных, построить матрицу точности на контрольной выборке. Проинтерпретировать результат.
5. Протестировать алгоритм в условиях, когда обучающие данные приходят порциями. Как меняются кластеры по мере добавления данных с новыми видами цифр?
6. Написать отчёт в формате PDF с описанием всех проведённых исследований. В отчёте также должен содержаться вывод необходимых формул.

Рекомендации по выполнению задания

1. Функционал $\mathcal{L}(q)$ *должен* монотонно возрастать с течением итераций. Если это не так, то в реализации или в выводе формул ошибка. Рекомендуется также на этапе отладки следить за тем, чтобы функционал $\mathcal{L}(q)$ монотонно возрастал после каждого пересчёта отдельной компоненты вариационного приближения q.
2. Для того, чтобы избежать проблем с точностью вычислений, следует везде, где это возможно, переходить от произведений к суммированию логарифмов.
3. Для подсчета дигамма, гамма, и логарифма гамма функции можно использовать библиотеку scipy (модуль `scipy.special`, функции `digamma`, `gamma`, `gammaln` соответственно).
4. На этапе дообучения алгоритма при поступлении новой порции данных рекомендуется оставлять только компоненты $q(\theta)$, соответствующие найденным ненулевым кластерам, а все остальные величины инициализировать заново.

Спецификация

Необходимо предоставить py-файл, в котором реализован класс DPMixture для работы с описанной моделью кластеризации.

Конструктор класса:

```
def __init__(self, X, alpha, a, b)
```

- `X` — переменная типа `numpy.array`, матрица размера $N \times D$, наблюдаемые бинарные данные X,
- `alpha` — параметр концентрации априорного процесса Дирихле,
- `a`, `b` — параметры априорного Бета-распределения.

Алгоритм вариационного вывода:

```
def var_inference(self, num_start=1, display=True, max_iter=100, tol_L=1e-4)
```

- `num_start` — количество запусков из случайных начальных приближений,

- `display` — параметр отображения, если равен `True`, то показывается промежуточная информация об оптимизации вида номер текущей итерации, значение функционала $\mathcal{L}(q)$, кол-во найденных кластеров и проч.,
- `max_iter` — максимальное количество итераций для поиска вариационного приближения,
- `tol_L` — относительная точность оптимизации по значению функционала $\mathcal{L}(q)$.

Функция возвращает объект класса DPMixture.

Добавление обучающих данных:

```
def add_sample(self, X)
```

- `X` — новые наблюдаемые данные, которые добавляются к уже сохранённым внутри объекта класса.

Функция возвращает объект класса DPMixture.

Отображение найденных кластеров:

```
def show_clusters(self)
```

Оформление задания

Выполненное задание следует отправить письмом по адресу *bayesml@gmail.com* с заголовком письма

«[ШАД БММО15] Задание 3, Фамилия Имя».

Убедительная просьба присылать выполненное задание только один раз с окончательным вариантом. Также убедительная просьба придерживаться заданной выше спецификации.

Присланный вариант задания должен содержать в себе:

- Текстовый файл в формате PDF с указанием ФИО, содержащий описание всех проведённых исследований.
- Все исходные коды с необходимыми комментариями.