DOI https://doi.org/10.15407/usim.2018.02.031
УДК 519.245.

Е.В. ВОДОЛАЗСКИЙ, ст. научн. сотр.,
waterlaz@gmail.com

С.А. ЛАТЮК, инж.-програм. I кат.,
serhiylatyuk55@gmail.com

Международный научно-учебный центр информационных технологий
и систем НАН Украины и МОН Украины, просп. Глушкова, 40, Киев 03187, Украина

БЛОЧНАЯ МОДИФИКАЦИЯ СЭМПЛИРОВАНИЯ ПО ГИББСУ ДЛЯ РАСПОЗНАВАНИЯ СКРЫТЫХ МАРКОВСКИХ ПОЛЕЙ

Исследовано применение сэмплирования по Гиббсу и его модификаций для распознавания скрытых марковских полей, а также конструктивный метод реализации его блочной модификации распознавания изображений для случая, когда блоками служат строки изображения. Предложено использование нового метода оценки математического ожидания для задач распознавания на структурах, разметки которых обладают марковским свойством.

***Ключевые слова**: сэмплирование по Гиббсу, гиббсовский сэмплер, задачи разметки, распознавание скрытых марковских полей.*

Введение

Статья посвящена демонстрации блочной модификации сэмплирования по Гиббсу [1] и может быть конструктивно реализована и применена при решении задач распознавания изображений, в частности для решения задач разметки [2], к которым сводятся некоторые задачи компьютерного зрения. Известно, что задачи разметки заключаются в оптимизации функций от большого количества дискретных аргументов. Специфика их заключается в том, что функция, которую необходимо оптимизировать, есть суммой большого количества слагаемых, каждое из которых зависит от небольшого количества переменных.

Множество подобных задач образует *NP*-полный класс, а известные полиномиально разрешимые подклассы [2—4] включают в себя далеко не все ситуации, возникающие на практике. Поэтому появляется необходимость исследовать также неточные методы решения подобного рода задач, один из которых — сэмплирование по Гиббсу [1].

Сэмплирование уместно применять не во всех задачах распознавания, но при некоторых функциях штрафа оно может быть использовано для статистического оценивания определенных величин.

Определение основных понятий

Пусть T — множество *объектов*; например, множество пикселов поля зрения. Отдельный объект из T обозначим t или t'. На множестве T задано подмножество $\Im \subset T \times T$ пар объектов, которое назовем *отношением соседства*. Элементы множества $\Im$ обозначим (t, t'). Вы-

ражение $(t, t') \in \Im$ указывает, что объекты t и t' *соседние*. В дальнейшем для краткости (t, t') будем записывать как tt'.

Пусть K — конечное множество, элементы которого назовем *метками*, а отдельную метку обозначим k. Для каждого объекта t и каждой метки k задано число $q_t(k)$, а для каждой пары объектов tt' и пары меток k и k' — число $g_{tt'}(k, k')$. Эти числа назовем *весами*. Объект вместе с соответствующей ему меткой назовем *вершиной* и обозначим (k, t). Пару вершин, в которую входят соседние объекты назовем *дужкой* и обозначим $((k, t), (k', t'))$.

Разметкой назовем функцию k_T: $T \to K$, которая для каждого объекта $t \in T$ указывает метку $k_T(t) \in K$. Иногда для краткости метку в объекте t_i вместо $k_T(t_i)$ будем обозначать k_i. Набор меток, соответствующий некоторой группе объектов $\tau \subset T$, обозначим $k_T(\tau)$. Вес разметки определяется как сумма весов вершин и дужек, которые в нее входят:

$$G(k_T) = \sum_{tt' \in \mathrm{T}} g_{tt'}\left(k_T(t), k_T(t')\right) + \sum_{t \in T} q_t\left(k_T(t)\right).$$

Задача (max, +)-разметки — это поиск разметки с максимальным весом:

$$k_T^* = \arg\max_{k_T \in K_T} G(k_T).$$

Марковская модель изображения

Такая модель изображения может быть легко выражена в терминах понятий, введенных ранее. Пусть дано изображение размером m на n. *Множеством пикселов* назовем множество объектов (i, j) таких, что $1 \le i \le m$, $1 \le j \le n$. Для каждого пиксела (i, j) определим множество $N(i, j)$ соседних пикселов, $N(i, j) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$. Определим также два типа весовых функций: $g^B(k_T(t), k_T'(t')) = g_{tt'}(k_T(t), k_T'(t'))$, которые на каждой конкретной паре меток k, k' для всех пар объектов tt' вида (i, j), $(i+1, j)$ принимают одни и те же значения и $g^\Gamma(k_T(t), k_T'(t')) = g_{tt'}(k_T(t), k_T'(t'))$, обладающие аналогичным свойством для всех пар объектов вида (i, j), $(i, j+1)$.

Вероятность разметки для некоторого множества пикселов определяется так:

$$p(k_T) = \frac{1}{Z} \prod_{j=1}^{n} \left[\prod_{i=2}^{m} g^B\left(k_T(i-1, j), k_T(i, j)\right) \right] \times \prod_{i=1}^{m} \left[\prod_{j=2}^{n} g^\Gamma\left(k_T(i, j-1), k_T(i, j)\right) \right],$$

где Z — некоторый нормирующий множитель.

Пусть помимо метки, в каждом пикселе задан *сигнал* $x(i, j) \in X$. Совокупность сигналов во всех пикселах назовем *изображением* и обозначим x_T. Условную вероятность сигнала $x_T(i, j)$ в пикселе (i, j) при заданном значении $k_T(i, j)$ метки обозначим $q(x_T(i, j) | k_T(i, j))$. Вероятность изображения x_T при условии заданной разметки k_T может быть вычислена как

$$p(x_T | k_T) = \prod_{(i,j) \in T} q\left(x_T(i, j) | k_T(i, j)\right).$$

Распознавание скрытых марковских полей

Пусть каждый объект $t \in T$, имея метку $k_T(t) \in K$, с вероятностью $q(x_T(i, j) | k_T(i, j))$ излучает сигнал $x_T(t) \in X$. Условные вероятности $q(x|k)$ сигнала x при фиксированном значении метки k одинаковы для всех объектов. Задачу распознавания можно сформулировать так: дана группа объектов T, которые излучают сигналы $x_T(t) \in X$. Требуется, имея совокупность сигналов x_T, функции $g_{tt'}(k_T(t), k_T'(t'))$ и условные вероятности $q(x_T(t) | k_T(t))$, принять целесообразное, в определенном смысле, решение о метках $k_T(t) \in K$ этих объектов.

Одной из наиболее распространенных формализаций этого требования есть поиск апостериори наиболее вероятной разметки (*MAP*-оценка):

$$k_T^* = \arg\max_{k_T} \left(k_T | x_T\right).$$

Но даже при такой постановке задача в общем случае — *NP*-сложная. Полиномиальный алгоритм существует только для некоторых подклассов марковских сетей, например, когда структура имеет древовидную форму [2] либо при метках, удовлетворяющих условию супермодулярности [3, 4].

Однако, как известно [5], поиск апостериори наиболее вероятной разметки — это лишь

частный случай решения задачи минимизации байесовского риска (1):

$$k_T^* = \arg\min_{k_{T'}} \sum_{k_T \in K^T} p^*(k_T) \cdot W(k_T, k_T') \qquad (1)$$

при функции потерь, которая одинаково штрафует решение, вне зависимости от количества неправильно распознанных пикселов:

$$\begin{aligned} W(k_T, k_T') &= 0, \text{ если } k_T = k_T'. \\ W(k_T, k_T') &= 1, \text{ если } k_T \neq k_T. \end{aligned} \qquad (2)$$

Для некоторых задач (например сегментации или восстановления изображения в условиях шума) более удобной штрафной функцией есть не вероятность решения в целом (2), а математическое ожидание количества неправильно распознанных пикселов (так называемая *аддитивная* функция штрафа). В этом случае она имеет следующий вид:

$$W(k_T, k_T') = \sum_{t \in T} w\left(k_T(t), k_T'(t)\right),$$

где

$$w\left(k_T(t), k_T'(t)\right) = 0, \text{ если } k_T(t) = k_T'(t),$$
$$w\left(k_T(t), k_T'(t)\right) = 1, \text{ иначе.}$$

Известно, что в этом случае решение

$$k_T' = \arg\min_{k_T'} \sum_{k_T} p(k_T \mid x_T) W(k_T, k_T')$$

распадается на $|T|$ независимых оптимизаций вида

$$k' = \arg\min_{k'} \sum_{k} p(x_T, k) w(k, k') = \arg\max_{k} p(k \mid x_T).$$

Теперь проблема поиска оптимума как таковая отпадает, но вместо нее появляется другая — вычисления необходимых маргинальных вероятностей. Точное вычисление этих величин в общем случае представляет собой *NP*-сложную задачу. Поэтому используются различные приближенные методы, об одном из которых пойдет речь далее.

Сэмплирование по Гиббсу

Этот метод, в отличие от других, хорошо масштабируется и, следовательно, может эффективно функционировать на пространствах с

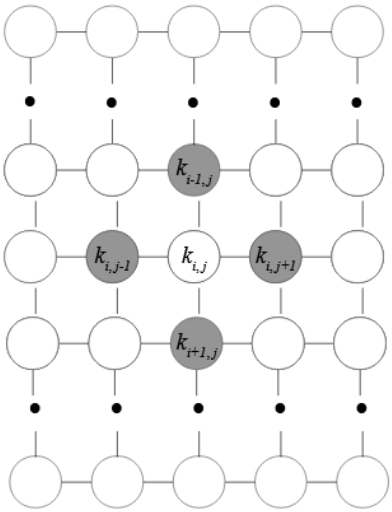


Рис. 1. Попиксельное сэмплирование

очень высокой размерностью. Идея, положенная в его основу — разбить сэмплирование из совместного распределения, — трудно выполнима:

$$p(k_T) = p\left(k_T(t_1), k_T(t_2), \ldots, k_T\left(t_{|T|}\right)\right),$$

на некоторое количество сэмплирований из условных распределений, которые выполнить легко

$$p\left(k_T(t_i) \mid k_T(t_1), \ldots, k_T(t_{i-1}), k_T(t_{i+1}), \ldots, k_T\left(t_{|T|}\right)\right).$$

Алгоритм выглядит так:

- установить начальные значения для всех меток (например на основе некоторого априорного распределения);
- заданное количество раз выполнить следующее:

– сгенерировать новое значение для $k(t_1)$ при условии текущих значений всех остальных меток:

$$k_T(t_1)^{new} \sim p(k_T(t_1) \mid k_T(t_2) \ldots k_T(t_{|T|}));$$

– сгенерировать новое значение для $k(t_2)$ при условии текущих значений всех остальных меток (с учетом нового значения $k(t_1)$):

$$k_T(t_2)^{new} \sim p(k_T(t_2) \mid k_T(t_1)^{new}, k_T(t_3) \ldots k_T(t_{|T|}));$$
$$\ldots$$
$$- k_T(t_i)^{new} \sim p(k_T(t_i) \mid k_T(t_1)^{new} \ldots k_T(t_{i-1})^{new}, k_T(t_{i+1}) \ldots k_T(t_{|T|}));$$
$$\ldots$$
$$- k_T(t_{|T|})^{new} \sim p(k_T(t_{|T|}) \mid k_T(t_1)^{new} \ldots k_T(t_{|T|-1})^{new}).$$

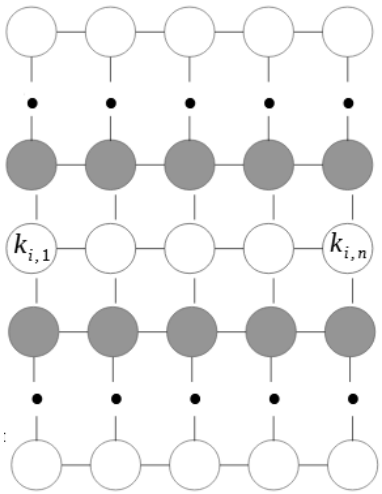


Рис. 2. Построчное сэмплирование

В силу марковского свойства гиббсовых случайных полей условное распределение одной метки при фиксированных значениях всех остальных меток равно ее условному распределению при фиксированных *соседних* метках (для изображения *m* на *n* пикселов), в чем и состоит эффективность сэмплирования (рис. 1):

$$p\left(k_T(i,j) \mid k_T(1,1),\ldots,k_T(i,j-1),k_T(i,j+1),\ldots\right.$$
$$\left.\ldots,k_T(m,n)\right) = p\left(k_T(i,j) \mid k_T(i-1,j),\right.$$
$$\left.k_T(i+1,j),k_T(i,j-1),k_T(i,j+1)\right).$$

В частности при распознавании изображений каждый пиксел имеет всего лишь четыре соседа.

Другим известным вариантом сэмплирования по Гиббсу есть так называемое блочное сэмплирование. Его отличие от предыдущего метода состоит в том, что на каждом шаге генерируется значение меток сразу для группы или *блока* объектов. Алгоритм приобретает следующий вид:

- установить начальные значения для меток объектов во всех блоках (предположим, что совокупность всех объектов разбита на *n* блоков);
- заданное количество раз выполнить следующее:

– сгенерировать новые значения $k_T(\tau_1)$ для блока τ_1 при условии текущих значений меток во всех остальных блоках:

$$k_T(\tau_1)^{new} \sim p(k_T(\tau_1) \mid k_T(\tau_2) \ldots k_T(\tau_n));$$

– сгенерировать новые значения для $k_T(\tau_2)$ при условии текущих значений меток во всех остальных блоках (с учетом новых значений $k_T(\tau_1)$):

$$k_T(\tau_2)^{new} \sim p(k_T(\tau_2) \mid k_T(\tau_1)^{new}, k_T(\tau_3) \ldots k_T(\tau_n));$$

$$\ldots$$

$$- k_T(\tau_i)^{new} \sim p(k_T(\tau_i) \mid k_T(\tau_1)^{new} \ldots k_T(\tau_{i-1})^{new}, k_T(\tau_{i+1}) \ldots k_T(\tau_n));$$

$$\ldots$$

$$- k_T(\tau_n)^{new} \sim p(k_T(\tau_n) \mid k_T(\tau_1)^{new} \ldots k_T(\tau_{n-1})^{new}).$$

Блочная модификация сэмплирования по Гиббсу для распознавания изображений

Такая модификация — воплощение попытки ускорить сходимость уже имеющегося метода. Поскольку две последовательно полученные разметки отличаются только значениями метки в одном объекте, то исследование всего пространства разметок осуществляется с довольно низкой скоростью. Решением этой проблемы могла бы быть возможность изменять на каждом шаге метку не в одном пикселе, а в группе пикселов. Это позволит алгоритму на каждом шаге *перепрыгивать* на менее похожую разметку и тем самым совершить блуждание по более широкой области.

Вариант блочной модификации, который предлагают авторы, модифицирует изображение построчно. Он возник из идеи, что если бы изображение состояло всего из одной строки, то никакое сэмплирование не понадобилось бы. В силу ацикличности имеющейся структуры можно было бы вычислить все интересующие вероятности за полиномиальное время [2]. Хотя в общем случае имеется гораздо больше строк, чем одна, которые зависимы одна от другой, генерация новых меток в целой строке, а не в одном пикселе все же могла бы оказаться эффективнее.

Пусть $k_T(i, j)$ обозначает метку в пикселе, который находится в i-й строке и j-м столбце. И пусть изображение, предоставленное для распознавания, имеет размер m на n пикселов. Одна итерация семплирования в этом случае выглядит так:

- $k_T(1, 1)^{new} \dots k_T(1, n)^{new} \sim p(k_T(1, 1) \dots \dots k_T(1, n) \mid k_T(2, 1) \dots k_T(2, n))$;
- $k_T(2, 1)^{new} \dots k_T(2, n)^{new} \sim p(k_T(2, 1) \dots \dots k_T(2, n) \mid k_T(1, 1)^{new} \dots k_T(1, n)^{new}, k_T(3, 1) \dots k_T(3, n))$;

...

- $k_T(\mathrm{i}, 1)^{new} \dots k_T(i, n)^{new} \sim p(k_T(i, 1) \dots \dots k_T(i, n) \mid k_T(i-1, 1)^{new} \dots k_T(i-1, n)^{new}, k_T(i+1, 1) \dots \dots k_T(i+1, n))$;

...

- $k_T(m, 1)^{new} \dots k_T(m, n)^{new} \sim p(k_T(m, 1) \dots \dots k_T(m, n) \mid k_T(m-1, 1)^{new} \dots k_T(m-1, n)^{new})$.

Отметим, что самое существенное при выборе блоков такой формы, блочное сэмплирование по Гиббсу имеет конструктивную реализацию, одна итерация которой, может быть выполнена за полиномиальное время. Это можно увидеть, выполнив следующее:

выразим совместное распределение полученного изображения и предполагаемой разметки в виде произведения сомножителей, зависящих от конкретной строки i^* этого изображения и тех, которые от нее не зависят:

$$p(k_T, x_T) = \frac{1}{Z}\prod_{j=1}^{n}\left[\prod_{i=2}^{m} g^B\left(k_T(i-1,j), k_T(i,j)\right)\right]\times$$

$$\times\prod_{i=1}^{m}\left[\prod_{j=2}^{n} g^\Gamma\left(k_T(i,j-1), k_T(i,j)\right)\right]\times$$

$$\times\prod_{i=1}^{m}\prod_{j=1}^{n} q\left(x_T(i,j) \mid k_T(i,j)\right) =$$

$$= \frac{1}{Z}\prod_{j=1}^{n}\left[\prod_{\substack{i=2 \\ i\neq i^*, i^*+1}}^{n} g^B\left(k_T(i-1,j), k_T(i,j)\right)\right]\times$$

$$\times\prod_{\substack{i=1 \\ i\neq i^*}}^{m}\left[\prod_{j=2}^{n} g^\Gamma\left(k_T(i,j-1), k_T(i,j)\right)\right]\times$$

$$\times\prod_{\substack{i=1 \\ i\neq i^*}}^{m}\prod_{j=1}^{n} q\left(x_T(i,j) \mid k_T(i,j)\right)\times$$

$$\times\prod_{j=1}^{n}\Big[g^B\left(k_T(i^*-1,j), k_T(i^*,j)\right)\times$$

$$\times g^B\left(k_T(i^*,j), k_T(i^*+1,j)\right)\Big]\times$$

$$\times\prod_{j=2}^{n} g^\Gamma\left(k_T(i^*,j-1), k_T(i^*,j)\right)\times$$

$$\times\prod_{j=1}^{n} q\left(x_T(i^*,j) \mid k_T(i^*,j)\right).$$

Введем обозначение:

$$q_j^1\left(k_T(i^*,j)\right) = q\left(x_T(i^*,j) \mid k_T(i^*,j)\right)\times$$

$$\times g^B\left(k_T(i^*-1,j), k_T(i^*,j)\right)\times$$

$$\times g^B\left(k_T(i^*,j), k_T(i^*+1,j)\right).$$

Теперь условное распределение меток и сигналов, соответствующих пикселам строки изображения, может быть выражено так:

$$p\left(k_T(i^*), x_T \mid k_T^*(\neg i^*)\right) =$$

$$= \frac{1}{Z}\prod_{j=2}^{n} g^\Gamma\left(k_T(i^*,j-1), k_T(i^*,j)\right)\times\prod_{j=1}^{n} q_j^1\left(k_T(i^*,j)\right),$$

где $q_j^1\left(k_T(i^*,j)\right)$ можно записать как $q_j^1\left(k_T(j)\right)$, поскольку значение i^* фиксировано.

Введем две вспомогательные функции:

$$\vec{f}_{j^*}\left(k_{j^*}\right) =$$

$$= \sum_{k_1}\sum_{k_2}\dots\sum_{k_{j^*-1}}\left[\prod_{j=2}^{j^*} g^\Gamma\left(k_{j-1}, k_j\right)\times\prod_{j=1}^{j^*-1} q_j^1\left(k_j\right)\right],$$

$$\overleftarrow{f}_{j^*}\left(k_{j^*}\right) = \sum_{k_{j^*+1}}\sum_{k_{j^*+2}}\dots\sum_{k_{j^*+n}}\left[\prod_{j=j^*+1}^{n} g^\Gamma\left(k_{j-1}, k_j\right)\times\right.$$

$$\left.\times\prod_{j=j^*+1}^{n} q_j^1\left(k_j\right)\right],$$

значения которых пропорциональны маргинальным вероятностям метки $k_T(j^*)$ в объекте (i^*, j^*). Легко показать, что

$$\vec{f}_{j+1}\left(k_{j+1}\right) = \sum_{k_j} g^\Gamma\left(k_j, k_{j+1}\right) * q_j^1\left(k_j\right) * \vec{f}_j\left(k_j\right),$$

$$\overleftarrow{f}_j\left(k_j\right)=\sum_{k_{j+1}} g^T\left(k_j,k_{j+1}\right)*q^1_{j+1}\left(k_{j+1}\right)*\overleftarrow{f}_{j+1}\left(k_{j+1}\right),$$

а также, что

$$p\left(k_{j-1},k_j\right)=\frac{1}{Z}\overrightarrow{f}_{j-1}\left(k_{j-1}\right)*q^1_{j-1}\left(k_{j-1}\right)\times$$
$$\times g^T\left(k_{j-1},k_j\right)*q^1_j\left(k_j\right)*\overleftarrow{f}_j\left(k_j\right).$$

Таким образом, новые значения меток для строки изображения могут быть сгенерированы за время порядка $O(n|K|^2)$.

Оценка апостериорных вероятностей меток

Как отмечено ранее, в случае минимизации количества неправильно распознанных объектов, задача разметки распадается на совокупность независимых оптимизаций по каждому объекту. Каждая из них заключается в поиске метки с максимальной апостериорной вероятностью. Посредством сэмплирования по Гиббсу можно приближенно оценить эти вероятности, рассматривая их как математические ожидания. Пусть дана функция $\varphi : K^T \to \theta$, которая каждой разметке ставит в соответствие некоторое число. Математическое ожидание значения функции φ обозначим θ^*:

$$\theta^* = \sum_{k_T \in K^T} p^*\left(k_T\right)\cdot\varphi\left(k_T\right).$$

Из $\lim\limits_{t\to\infty} p^t = p^*$ следует, что

$$\lim_{t\to\infty}\sum_{k_T\in K^n} p^t\left(k_T\right)\cdot\varphi\left(k_T\right)=\sum_{k_T\in K^n} p^*\left(k_T\right)\cdot\varphi\left(k_T\right)=\theta^*.$$

Еще один, менее распространенный способ вычисления математического ожидания — это предел среднего значения функции φ на первых τ разметках:

$$\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{t=0}^{\tau-1}\varphi\left(k_T{}^t\right) = \theta^*.$$

Проблема заключается в том, что приведенное равенство выполняется только в случае независимых разметок. О его выполнении или невыполнении в других случаях ничего не известно. Но, тем не менее, в случае марковской зависимости (как в данной ситуации), разметки можно считать независимыми, если брать каждую n-ю (где n — достаточно большое число), отбрасывая при этом все остальные.

Маргинальные вероятности метки k в пикселе t_i (далее $p_i(k)$), вычисление которых необходимо для решения задачи распознавания, можно рассматривать как математические ожидания значения некоторой функции, равной единице в случае метки a в пикселе i и равна нулю во всех остальных случаях:

$$p_i\left(a\right)=\sum_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_n} p^*\left(k_1,\ldots,a,\ldots,k_n\right).$$

Точное вычисление математического ожидания в данном случае невозможно вследствие экспоненциально большого количества возможных разметок. Но можно попытаться оценить его приближенно, генерируя достаточно большую выборку из распределения вероятностей разметок.

Первый метод назовем *подсчет*. Он заключается в том, чтобы подсчитать все разметки из сгенерированной выборки, в которых пиксел имеет какую-то определенную метку, например a. Затем разделить это количество на размер выборки и полученное число считать вероятностью того, что данный пиксел имеет метку a:

$$p_i\left(a\right)\approx\frac{1}{m}\sum_{t=1}^{m}\left[k_T\left(t_i\right)=a\right].$$

Второй метод назовем *усреднением*. Главная его идея в том, чтобы рассматривать вероятность присутствия в пикселе некоторой метки a, как математического ожидания апостериорной вероятности того, что данный пиксел имеет метку a, при условии излучения сигнала x. Математическое ожидание в свою очередь, как известно, можно аппроксимировать средним значением:

$$p_i\left(a\right)=\sum_{x\in X} p\left(x\right)\cdot p\left(a\,|\,x\right) \approx \frac{1}{m}\sum_{t=1}^{m} p\left(a\,|\,x_t\right).$$

Экспериментальное сравнение четырех вариантов реализации

Итак, имеем четыре возможных варианта реализации сэмплирования по Гиббсу:

- попиксельный с подсчетом;
- попиксельный с усреднением;
- блочная модификация с подсчетом;
- блочная модификация с усреднением.

В качестве задачи распознавания, которая решалась перечисленными алгоритмами, для простоты была выбрана задача восстановления зашумленного изображения. Выполнена следующая процедура:

- сгенерировано изображение с некоторыми текстурными характеристиками, на котором показано некоторое количество цветов (в данном случае цвет — это метка);
- каждый пиксел с некоторой фиксированной вероятностью меняет свой цвет на другой, выбранный случайно цвет;
- полученное изображение подается на вход для распознавания каждому из четырех алгоритмов.

Теоретически, все четыре метода должны сходиться к одному и тому же результату (но не обязательно с одинаковой скоростью).

Тестирование проведено для разных уровней шума (0,4—0,9) и разного количества меток (3—8) на двух типах текстур: так называемых *диагональных* и *монотонных* текстурах.

Диагональные текстуры характерны тем, что для соседних пикселов (как по горизонтали, так и по вертикали) вероятность появления в них одинаковых меток намного больше, чем разных (рис. 3). Жирным на рисунке обозначены пары меток, которые возникают с высокой вероятностью, тонким — с низкой. Типичная текстура с такой матрицей весов выглядит как совокупность округлых пятен (рис. 4).

На текстурах этого типа улучшений в работе блочной модификации в сравнении с попиксельным алгоритмом не наблюдалось.

Монотонные текстуры характерны тем, что метки всегда могут быть пронумерованы так, чтобы в каждой строке и каждом столбце их номера слева направо и сверху вниз не умень-

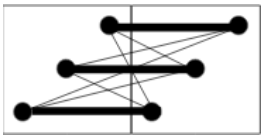


Рис. 3. Схематическое изображение отношений между вероятностями различных пар меток на двух соседних пикселах для диагональных текстур

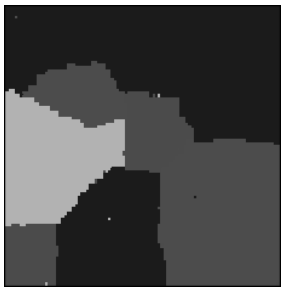


Рис. 4. Пример диагональной текстуры

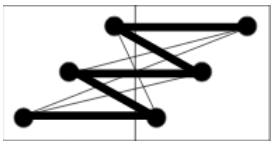


Рис. 5. Схематическое изображение отношений между вероятностями различных пар меток на двух соседних пикселах для монотонных текстур

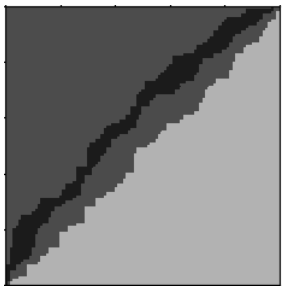


Рис. 6. Монотонная текстура

шались (т. е. были расположены по неубыванию) (рис. 5).

Типичная текстура с такими свойствами выглядит как два больших треугольных пятна в левом верхнем и правом нижнем углах и два тонких продолговатых диагональных пятна по центру (рис. 6).

На этом типе текстур удалось достичь некоторых интересных результатов. Хотя при малом уровне шума все четыре метода ведут себя

почти одинаково, при его повышении блочные модификации начинают сходиться быстрее. Отметим, что именно для блочной модификации метод усреднения для оценки маргинальных вероятностей работает лучше, в то время как для попиксельной реализации такого не наблюдается.

Результаты сравнения работы алгоритмов

Очевидно, что время выполнения одной итерации сэмплирования классическим алгоритмом меньше, чем посредством блочной модификации. Это обусловлено тем, что ассимптотическая сложность попиксельного сэмплирования — $O(n|K|)$ в то время как блочной модификации — $O(n|K|^2)$. Но, тем не менее, для малого количества меток (восемь и меньше) на монотонных текстурах блочная модификация сходится быстрее, что скорее всего происходит из-за возможности учитывать зависимости между метками пикселов одной строки, находящиеся далеко одна от другой. Насколько сильно эти зависимости влияют на скорость распознавания в общем случае, пока сказать трудно, что говорит о необходимости дальнейших исследований в этой области.

Ниже представлены графики зависимости качества распознавания, демонстрируемого алгоритмами, от времени их работы. По горизонтальной оси — время в секундах, по вертикали — количество неправильно распознанных пикселов.

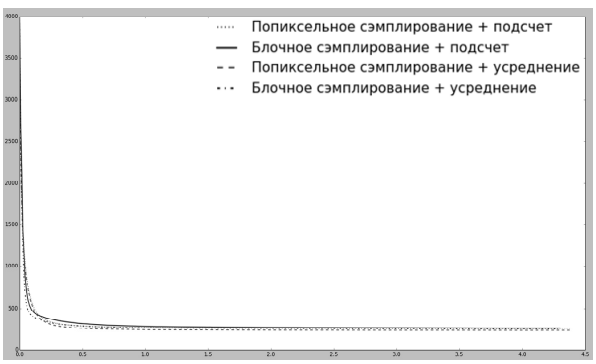
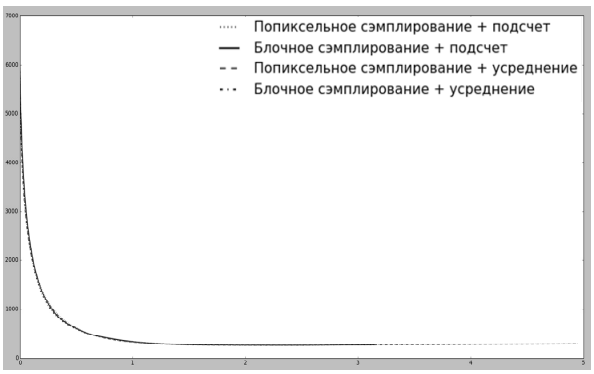


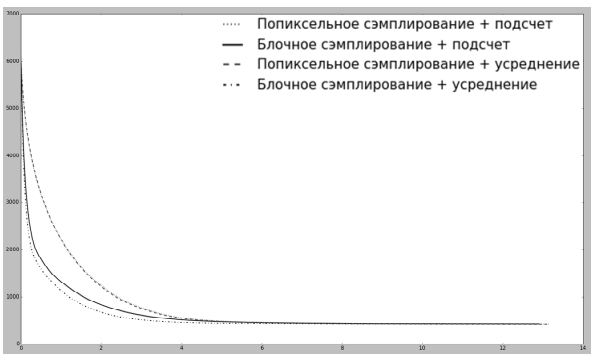
Рис. 7. Распознавание на диагональных текстурах

На рис. 7 показано распознавание на диагональных текстурах при различном уровне шума и количестве меток, которое проводится одинаково как классическим алгоритмом, так и его модификациями.

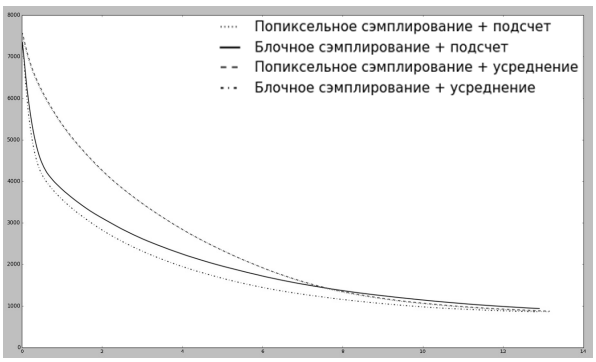
На рис. 8 показан результат распознавания на монотонных текстурах при количестве меток — четыре и различном уровне шума.



а — уровень шума — 0,4



б — уровень шума — 0,6



в — уровень шума — 0,7

Рис. 8. Распознавание на монотонных текстурах

Работа распознающей программы

На рис. 9 показаны результаты распознавания на диагональных и монотонных текстурах.

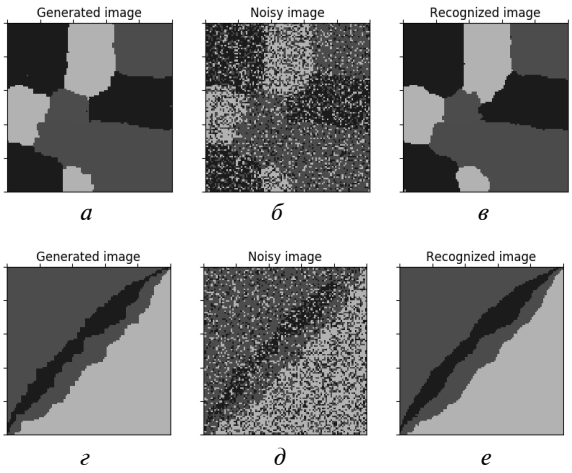


Рис. 9. Распознавание на диагональных (*а-в*) и монотонных текстурах (*г-е*).

Результаты: *а* и *г* — для сгенерированной разметки, *б* и *д* — для входного зашумленного изображения, *в* и *е* — результат распознавания.

Заключение

В случае распознавания изображений блочная модификация сэмплирования по Гиббсу может быть конструктивно реализована, если в качестве блоков выступают строки пикселов изображения. Предложенный альтернативный метод оценки математического ожидания может быть использован в задачах распознавания скрытых марковских полей.

Сравнение работы общеизвестного алгоритма сэмплирования по Гиббсу и его блочной модификации проведено на изображениях с двумя типами текстур на количестве меток от трех до восьми, из чего можно сделать следующие выводы:

- при невысоком уровне шума блочная модификация на обеих текстурах работает не хуже общеизвестной попиксельной реализации;
- при увеличении уровня шума на монотонных текстурах блочная модификация работает лучше, что выражается в меньшем количестве неправильно распознанных пикселов в сравнении с попиксельной на начале работы алгоритма;
- предложен ный альтернативный метод для оценки математического ожидания заметных улучшений в работе алгоритма не дал, но исходя из полученных графиков, можно предположить, что применительно к блочной модификации, на высоком уровне шума он более эффективен, чем стандартный.

СПИСОК ЛИТЕРАТУРЫ

1. *Geman S.*, *Geman D.* Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984, **6**, N 6, P. 721–741.
2. *Шлезингер М.И.* Синтаксический анализ двумерных зрительных сигналов в условиях помех, Кибернетика, К., 1976, Т. 4, С. 113–130.
3. *Schlesinger M.I.*, *Flach B.* Some solvable subclass of structural recognition problems, Czech Pattern Recognition Workshop 2000, Ed. by T. Svoboda, Praha: Czech Pattern Recognition Society, Feb. 2000, P 55–62.
4. *Schlesinger M.*, *Flach B.* Analyzis of optimal labelling problems and their applications to image segmentation and binocular stereovision, Proc. East-West-Vision 2002 (EWV'02), Ed. by F. Leberl, A. Ferko, International Workshop and Project Festival on Computer Vision, Computer Graphics, New Media, 2002, P. 55–60.
5. *Schlesinger M.*, *Hlavác V.* Ten Lectures on Statistical and Structural Pattern Recognition, Dordrecht: Kluwer Acad. Publ., 2002, P. 355–361.

Поступила 06.07.2018

Ye.V. Vodolazskiy, Senior researcher, waterlaz@gmail.com

S.A. Latiuk, Engineer-programmer of 1-st category, serhiylatyuk55@gmail.com

International Research and Training Center for Information Technologies and Systems of the National Academy of Sciences (NAS) of Ukraine and Ministry of Education and Science (MES) of Ukraine

BLOCKING MODIFICATION OF GIBBS SAMPLING FOR RECOGNITION OF HIDDEN MARKOV FIELDS

Introduction. This paper is dedicated to demonstration of the fact that blocking Gibbs sampling can be constructively implemented and applied to solve the image recognition tasks, in particular — for solving the labelling problems, to which some Computer Vision problems can be reduced. It's known, that these problems are optimization problems with the big number of the discrete arguments. The specificity of these tasks lies in the fact that function, which we want to optimize is the sum of a large number of terms, each of which depends on a few arguments. Tasks of such type are *NP*-hard, and polynomially solvable subclasses include very small number of situations that can happen in practice. So, there is a need to explore approximate methods for solving such problems. One of these methods is Gibbs sampling. There is no sense to apply Gibbs sampling in all recognition tasks, but it can be used in situations with some special loss functions for the statistical estimation of some quantities.

Purpose of this paper is to demonstrate that blocking Gibbs sampling can be constructively implemented for solving image recognition problems. Secondly, we want to compare blocking modification with standard algorithm and find types of the textures, on which blocking modification will give better results. And, finally, we want to test validity of the proposed method for estimation of expectation.

Methods. Blocking modification of Gibbs sampling for image recognition is built on the principles of dynamic programming. Software that allows to generate images of the given texture, choose proper modification of Gibbs sampling for solving the problem and show graphs using *Python* and *Cython*.

Results. Blocking Gibbs sampling works better then standard realization, when images for recognition has so-called «monotonous» texture. In case of recognition the images with so-called «diagonal» texture (that very often arises in practice) blocking modification works not worse than standard method. It was also noticed that proposed method for estimation of expectation works a little better than the standard one.

Conclusion. Blocking Gibbs sampling can be applied for the solving image recognition tasks. It has constructive implementation which works in polynomial time. Asymptotically, one iteration of blocking Gibbs sampling is performing slower than one iteration of the standard Gibbs sampling algorithm. But in our experiments recognition at all (on «monotonous» textures) was performing faster with blocking algorithm. Due to this fact, we can conclude that blocking modification is able to take into consideration some additional information that allows to perform the recognition faster. So, it makes sense to do further research in this fild.

***Keywords**: Gibbs sampling, Gibbs sampler, labelling problems, hidden Markov fields recognition.*

Є.В. Водолазський, ст. наук. співр., Міжнародний науково-навчальний центр інформаційних технологій та систем НАН України та МОН України, просп. Глушкова, 40, Київ 03187, Україна, waterlaz@gmail.com

С.О. Латюк, інж.-програм. І кат., serhiylatyuk55@gmail.com

Міжнародний науково-навчальний центр інформаційних технологій та систем НАН України та МОН України, просп. Глушкова, 40, Київ 03187, Україна

БЛОЧНА МОДИФІКАЦІЯ СЕМПЛУВАННЯ ЗА ГІБСОМ ДЛЯ РОЗПІЗНАВАННЯ ПРИХОВАНИХ МАРКІВСЬКИХ ПОЛІВ

Вступ. Статтю присвячено демонстрації того, що блокова модифікація семплування за Гібсом може бути конструктивно реалізована та з успіхом застосована у вирішенні задач розпізнавання зображень, зокрема — у вирішенні задач розмітки, до яких зводяться деякі задачі комп'ютерного зору. Відомо, що задачі розмітки полягають в оптимізації функцій від великої кількості дискретних аргументів. Специфіка цих задач полягає в тому, що функція, яку необхідно оптимізувати, є сумою великої кількості доданків, кожен з яких залежить від невеликої кількості змінних. Множина таких задач утворює *NP*-повний клас, а відомі поліноміально розв'язні підкласи містять далеко не всі ситуації, які можуть виникнути на практиці. Тому є потреба досліджувати також неточні методи розв'язання задач такого типу, одним з яких є семплування за Гібсом. Сенс застосовувати семплування не у всіх задачах розпізнавання, проте при деяких функціях штрафу воно може бути використане для статистичного оцінювання певних величин.

Мета статті — демонстрація існування конструктивної реалізації блочної модифікації семплування за Гібсом для розпізнавання зображень. Порівняння її роботи зі стандартною реалізацією та пошук текстур, на яких блочна модифікація працює краще. Перевірка придатності запропонованого методу оцінки математичного сподівання для використання при розпізнаванні прихованих марківських полів.

Методи. Блочну модифікацію семплування за Гібсом для розпізнавання зображень побудовано на принципах динамічного програмування. Додаток, що дозволяє генерувати зображення заданої текстури, обирати потрібну модифікацію семплування за Гібсом розв'язку задачі розпізнавання та отримати графіки залежності швидкості розпізнавання кожного алгоритму від часу реалізовано на мові Python з використанням бібліотеки Cython для написання розширень на мові *C*.

Результат. Вявлено, що блочна модифікація семплування за Гібсом на так званих «монотонних» текстурах при високому рівні шуму працює краще, ніж стандартний алгоритм. На «діагональних» текстурах, які дуже часто зустрічаються, блочна модифікація працює не гірше, ніж загальновідомий метод. Також виявлено, що на «монотонних» текстурах запропонований метод для оцінки математичного сподівання демонструє кращі результати, ніж загальноприйнятий.

Висновок. Блочну модифікацію семплування за Гібсом можна використати для розпізнавання зображень, завдяки наявності конструктивної реалізації, яка працює за поліноміальний час. Хоча асимптотично одна ітерація блочної модифікації виконується повільніше, ніж одна ітерація стандартного алгоритму семплування за Гібсом, проте на малій кількості міток, на «монотонних» текстурах розпізнавання в цілому відбувалося швидше завдяки блочному семплуванню, що свідчить про спроможність даного алгоритму враховувати певну додаткову інформацію, яка в свою чергу пришвидшує процес розв'язку. А отже, є сенс проводити подальші дослідження в даній галузі.

Ключові слова: *семплування за Гібсом, Гібсівський семплер, задачі розмітки, розпізнавання прихованих марківських полів.*