Занятие 6: Подготовка текстовых отчётов в системе TEX

Практикум на ЭВМ 2017/2018

Д.А. Кропотов

10 октября 2017 г.

Элементы хорошего отчёта по заданию

- Отчёт подготовлен в системе TEX;
- Объём отчёта: 5–20 страниц;
- Текст отчёта не повторяет полной формулировки задания;
- Структура отчёта соответствует пунктам задания;
- Используются векторные шрифты;
- Графики оформлены надлежащим образом;
- Шкала для графиков выбрана правильно;
- На разных графиках результаты для одинаковых методов отображаются одним и тем же цветом;

Элементы хорошего отчёта по заданию

- Между расположением графиков и местами их упоминания в тексте относительно небольшое расстояние (на той же или на соседней странице);
- На страницах не должно быть много пустого места;
- В большинстве случаев графики/таблицы/псевдокоды алгоритмов не должны занимать большей части одной страницы отчёта;
- Все числа в тексте/таблицах указаны с необходимым числом значащих цифр;
- В большинстве случае в отчёте не должно быть никакого кода;
- Для всех экспериментов описан выбранный дизайн экспериментов, а также сделаны выводы из полученных результатов;

Растровые и векторные шрифты

Растровые шрифты:

Для каждой задачи написать н
да различной эффективности, и
ванный вариант и один вариант
работы реализаций на нескольк
Проанализировать полученные
реализаций и сделать выводы.

Векторные шрифты:

Для выполнения задания было предлож
дой задачи представлены в отдельных pyth
задачи).

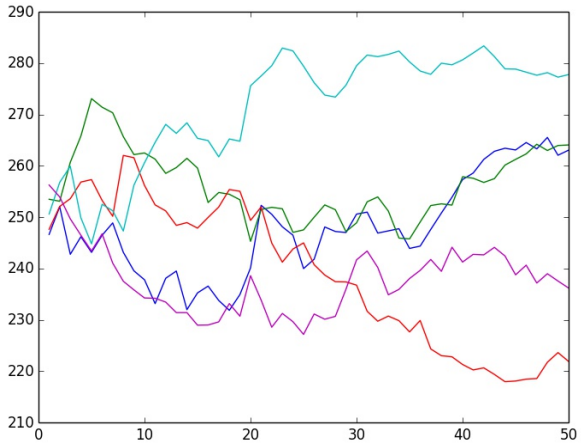
Реализация тестов представлена в модул
образом: первая часть тестов проверяет кор
решения задачи, вторая часть проверяет сов
решений.

Проведение экспериментов реализовано
римент, направленный на сравнение времени
проводился 100 раз, после чего выбиралось

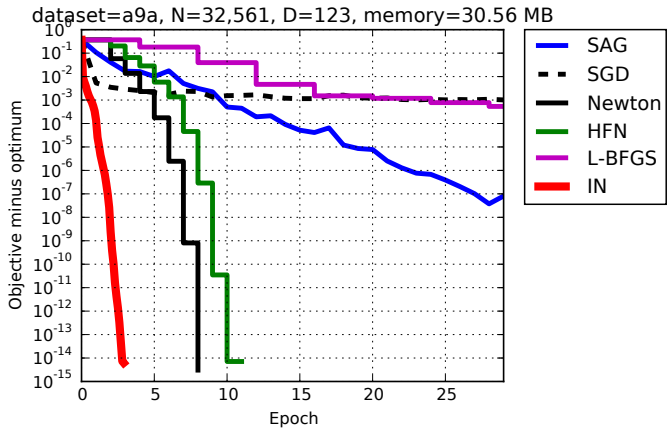
Для генерации текстов с векторными шрифтами достаточно установить пакет TEX `cm-super`.

Оформление графиков

Плохой график:



Хороший график:



Элементы хорошего графика:

- Все линии жирные;
- Есть легенда;
- По осям указаны значения, сами оси подписаны;
- По осям выбрана правильная шкала.

Примеры плохо организованных страниц

Таблица 9: Результаты экспериментов задачи №8. Время работы измерено в микросекундах

	vectorized_8	non_vectorized_8	my_method_8	multivariate_normal
shape = (50, 100)	940.2	630531.5	42079.1	426.7
shape = (100, 200)	3985.8	4968530.7	386522.0	2636.1
shape = (150, 300)	10492.1	16607857.2	1408267.6	8813.9

Таблица 10: Результаты точности вычисления

	vectorized_8	non_vectorized_8	my_method_8
shape = (50, 100)	2.29e-13	1.29e-12	3.23e-12
shape = (100, 200)	2.44e-13	1.33e-12	3.29e-12
shape = (150, 300)	2.32e-13	1.29e-12	3.25e-12

15

Заметим, что стандартная реализация scipy.stats.multivariate_normal отстает от vector_method:

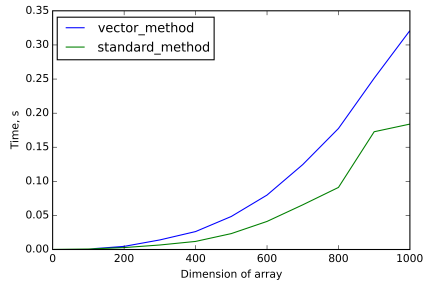


Рис. 6

Погрешность была вычислена как евклидова норма от разности двух функций:

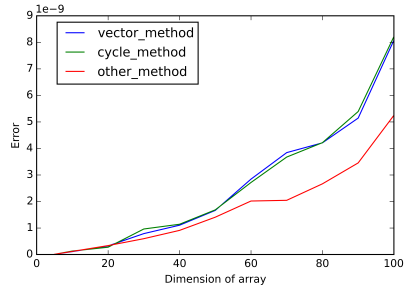


Рис. 7: Погрешность

Есть большие пустые пространства на странице, графики занимают много места.

Примеры плохо организованных страниц

Измерение времени

Время выполнения функций проверялось на квадратных матрицах X размера $n \times n$ сгенерированных из равномерного распределения и векторах i, j размера n сгенерированных из дискретного равномерного распределения. Результаты (Рис. 2), как и в задаче 1, показывают, что стандартные циклы в Python работают медленнее чем функции и методы, реализованные в библиотеке numpy. Индексация в numpy массивах работает медленнее чем метод take, но разница небольшая. В отличии от первой задачи, разница во времени выполнения между реализациями с ростом размерности данных не изменяется.

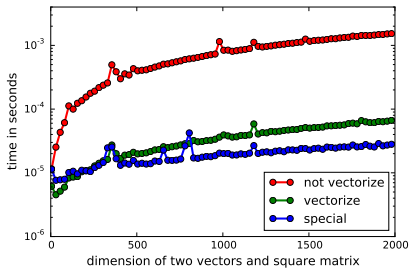


Рис. 2: Зависимость времени выполнения задачи 2 от размерности данных

Есть большие пустые пространства на странице.

Примеры плохо организованных страниц

10 Задача 8

Условие

Реализовать функцию вычисления логарифма плотности многомерного нормального распределения
Входные параметры: точки X, размер (N, D), мат. ожидание m, вектор длины D, матрица ковариаций C, размер (D, D). Сравнить с $scipy.stats.multivariate_normal(m, C).logpdf(X)$ как по скорости работы, так и по точности вычислений.

Решение 1. Векторизованное

Формула плотности невырожденного нормального распределения выполнена для всей матрицы X.

```
def v1_vector(X, m, C):
    n = m.shape[0]
    ans = -(n/2.0)*np.log(2*np.pi) - 0.5*np.linalg.slogdet(C)[1]
    ans -= 0.5*np.dot(np.dot((X-m), np.linalg.inv(C)), (X-m).T)
    return np.diag(ans)
```

Решение 2. Менее векторизованное

Формула плотности невырожденного нормального распределения выполнена для каждого вектора матрицы X.

```
def v2_non_vector(X, m, C):
    ans = []
    n = m.shape[0]
    for i in range(X.shape[0]):
        x = X[i]
        ans.append(-(n/2.0)*np.log(2*np.pi) - 0.5*np.linalg.slogdet(C)[1] -
                   0.5*np.dot(np.dot((x-m), np.linalg.inv(C)), (x-m).T))
    return np.array(ans)
```

Решение 3. С использованием SciPy

```
def v3_part_vector(X, m, C):
    return scipy.stats.multivariate_normal(m, C).logpdf(X)
```


Вычисление скоростей

В модуле $task2.py$ описана функция $gen(size)$, которая в зависимости от значения параметра $size$ генерирует случайные тестовые данные разных объёмов.
Время измеряется в IPython Notebook функцией %timeit -o -q.

Значение $size$	0	1	2	3	4
Размеры матрицы X	70×50	100×120	520×500	1000×1020	5020×5000
Размеры векторов i, j	60	110	510	1010	5010
Время работы векторного решения	18 μs	21 μs	40 μs	73 μs	0.44 ms
Время работы невекторного решения	134 μs	227 μs	1.1 ms	2.2 ms	12.1 ms
Время работы решения 3	48 μs	72 μs	0.45 ms	5 ms	125 ms

Вывод

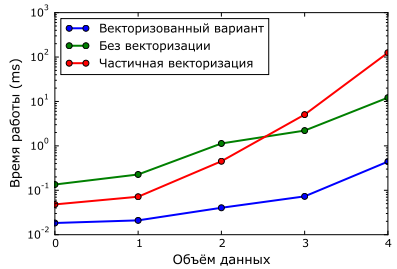


Рис. 2: Задача 2. График зависимости скоростей разных реализаций от объёма входных данных

Как видно на рисунке 2, векторная реализация работает быстрее остальных. Частично-векторная реализация работает быстрее невекторной на маленьких объемах данных. Но при росте объема данных замедляется, так как при выборе всех нужных строк сохраняется много лишней информации.

Текст/таблицы не выровнены по правому краю.

Описание экспериментов

Эксперименты не описаны, нет выводов:

Задание 2

Векторизированный вариант:

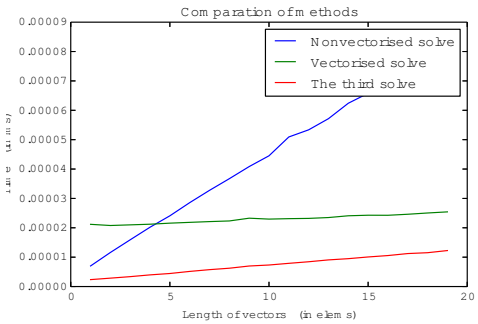
```
def vect_2(x, i, j):
    return np.vectorize(lambda z, y: x[z, y])(i, j)
```

Не векторизированный вариант:

```
def vect_1(x, i, j):
    r = np.array([], x.dtype)
    for k in range(np.size(i)):
        r = np.append(r, x[i[k], j[k]])
    return r
```

Третий вариант:

```
def vect_3(x, i, j):
    return np.array([x[t[0]][t[1]] for t in zip(i, j)])
```

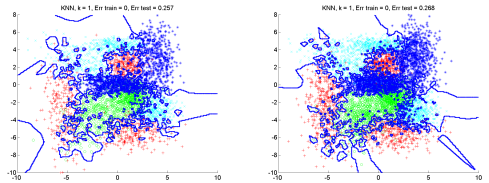


Самое оптимальное третье решение.

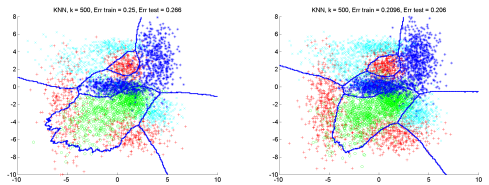
4

Рассуждения проведены:

однако совершенно не приближается к оптимальному классификатору (уровень ошибки на тесте фиксирован на уровне 0.25). Увеличение объёма данных не приведёт к успеху в этой слишком простой модели, уровень ошибки существенно не уменьшится. Вот результат для 3000 и 5000 обучающих объектов:



Метод 500 ближайших соседей сильно недообучен, границы, выдаваемые им, слишком просты. Он обладает высоким *bias* (близкие кривые обучения на высоком уровне ошибки) при низком *variance*. При размере данных меньше 500 (50%) на тестовой выбоке наблюдается плато на уровне 75%: классификатор выдаёт в качестве ответа максимальный класс. На тестовой выборке при этом ошибка всё же ниже: сказывается случайный порядок объектов в обучающей выборке, соотношение классов не по 25%, и доминирующий класс составляет чуть более 25%. Однако видна тенденция к падению уровня ошибки. Это происходит потому, что с увеличением объёма данных ослабляется эффект «доминирующего класса», и в этом случае добавление новых данных приведёт к значительному улучшению. Результат для 3000 обучающих объектов будет выглядеть гораздо более приемлемо, а для 5000 он даже приближается к оптимальному:



Налицо усложнение вида границ чисто за счёт большего объёма обучающей выборки. Это ещё раз наглядно иллюстрирует, насколько разный результат может давать одно и то же значение структурного параметра в зависимости от размера выборки.

3.2 SVM

Вот как выглядит зависимость ошибки SVM на обучении, валидации и тесте от каждого из параметров C и γ (другой параметр при этом фиксирован и равен 1; для параметров используется логарифмическая шкала):

8

Система TEX

TEX — система компьютерной вёрстки, построенная по принципу компиляции документа, записанного с помощью специального языка разметки.

Изобретена Д. Кнутом в конце 70х годов.

Является де-факто стандартом для написания научных статей.

Язык разметки TEX используется для набора формул во многих других системах: в вики-разметке, в matplotlib, Microsoft Office и др.

Дистрибутивы и редакторы TEX

Дистрибутивы:

- Windows: MiKTeX, TeX Live
- Linux: TeX Live
- Mac OS: MacTeX, TeX Live

Редакторы: WinEdt, TeXnicCenter, Kile и др.

Структура файла .tex

```
\documentclass{article}

%Преамбула документа

% задаём кодировку файла
\usepackage[utf8]{inputenc}
% задаём правила переносов для русского языка
\usepackage[russian]{babel}

%Текст документа
\begin{document}
    Некоторый текст в первом абзаце.
    Несколько      пробелов подряд считаются
    одним пробелом. Конец абзаца можно
    задать командой \par

    Также конец абзаца задаётся пустой строкой.
\end{document}
```

Разделы и подразделы

```
\section{Раздел 1}
\subsection{Подраздел}
\paragraph{Подподраздел}

\section{Раздел 2}
```

Формулы

```
Формула в тексте: $\sin^2x + \cos^2x = 1$.
```

Формула в тексте: $\sin^2 x + \cos^2 x = 1$.

```
Выносная формула:
$$
    A_{ij} = b_i^2 + c_j^3\quad \forall i,j=1,\dots,n.
$$
```

Выносная формула:

$$A_{ij} = b_i^2 + c_j^3 \quad \forall i, j = 1, \dots, n.$$

Формулы

```
Выравнивание формул:
\begin{align}
    \notag E &= mc^2 \\
    \label{eq:1}&E = mc^2
\end{align}
```

Выравнивание формул:

$$E = mc^2$$

$$E = mc^2 \tag{1}$$

Ссылки

```
\begin{equation}
\label{eq::1}
    E = mc^2.
\end{equation}

Ссылка на формулу~\eqref{eq::1}
```

$$E = mc^2. \tag{2}$$

Ссылка на формулу (2)

```
\section{Ссылки}\label{sec:1}
Ссылка на раздел~\ref{sec:1} в документе.
```

Картинки и таблицы

```
Картинка в тексте:
\includegraphics[width=5cm]{a9a_epoch.pdf}
Таблица:
\begin{tabular}{ccc}
    a & b & c \\
    d & e & f \\
\end{tabular}
```

Картинка в тексте:

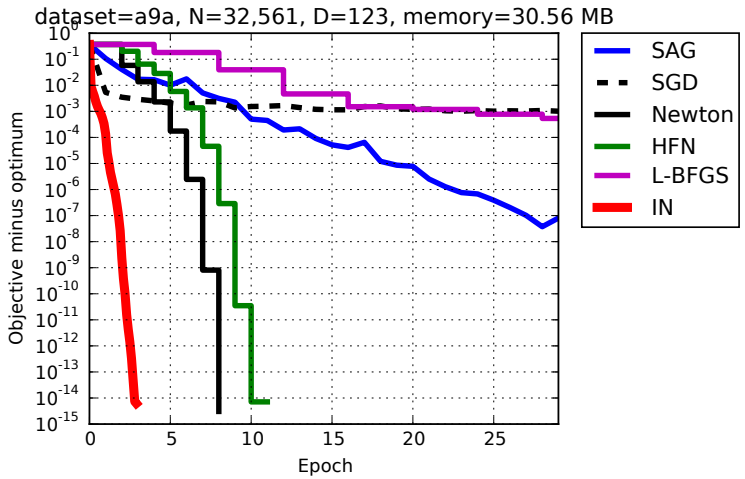


Таблица:

a	b	c
d	e	f

Расположение картинок на странице

```
\begin{figure}[h]  %Разместить таблицу здесь
    \begin{center}
        \includegraphics[width=5cm]{a9a_epoch}
    \end{center}
    \caption{Картинка \label{fig::1}}
\end{figure}

Ссылка на картинку: \ref{fig::1}
```

Список литературы

```
Ссылка в тексте на публикацию~\cite{vorontsovLatex}.

% В конце документа
\section{Список литературы}
\begin{thebibliography}{99}
\bibitem{vorontsovUrl}
Воронцов К. В., Полезная информация для
пользователей \LaTeX,
\url{http://www.ccas.ru/voron/latex.html}

\bibitem{vorontsovLatex}
Воронцов К. В., \LaTeX в примерах, 2005,
\url{http://www.ccas.ru/voron/download/voron05latex.
\end{thebibliography}
```

Список литературы

```
Ссылка в тексте на публикацию~\cite{blei06variational

% В конце документа
\section{Список литературы}
\bibliographystyle{gost780s}
\bibliography{references}
```

В файле references.bib:

```
@ARTICLE{blei06variational,
  author =        {D. Blei and M. Jordan},
  title =         {Variational inference for {D}irichlet
  journal =       {Journal of Bayesian Analysis},
  year =          {2006},
  volume =        {1},
  number =        {1},
  pages =         {121--144},
}
```