МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ
Институт системной и программной инженерии
и информационных технологий (Институт СПИНТех)

Лабораторный практикум по курсу "Интеллектуальные системы" (09/22 – 01/23)

Лабораторная работа 3
Статистические методы обучения. Метод опорных векторов [1].

На этом занятии компьютерного практикума Вы изучите *метод опорных векторов* (англ. SVM, *Support Vector Machine*) и примените данный метод для решения задачи классификации различных двумерных наборов данных. В последних публикациях на русском языке метод называется также *машинами поддерживающих векторов*, или, в более общем смысле, ядерными машинами (англ. *Kernel Machine*). В методах, основанных на их использовании, предусмотрен эффективный механизм обучения, а сами они позволяют представить сложные, нелинейные функции.

Ядерные машины превосходят все другие способы распознавания рукописных символов, в частности цифр; кроме того, они быстро находят применение и в других приложениях, особенно в тех, которые отличаются большим количеством входных характеристик.

Прежде чем приступить, собственно, к программированию, настоятельно рекомендуется ознакомиться с материалом лекций, а также с дополнительными материалами, имеющими отношение к задачам классификации.

Файлы, включенные в задание:

ex3.m – скрипт, реализующий пошаговое выполнение первой части задания по разделу «Статистические методы обучения. Метод опорных векторов»;
ex3data1.mat – 1-й набор данных;
ex3data2.mat – 2-й набор данных;
ex3data3.mat – 3-й набор данных;
svmTrain.m – функция обучения для метода опорных векторов;
svmPredict.m – функция для прогноза метода опорных векторов;
plotData.m – функция графического отображения данных;
visualizeBoundaryLinear.m – отображение линейной границы раздела данных;
visualizeBoundary.m – отображение нелинейной границы раздела данных;
linearKernel.m – ядро линейного отображения;
* *gaussianKernel.m* – ядро Гаусса для метода опорных векторов;
* *dataset3Params.m* – Параметры для 3-го набора данных

[1] Материал лабораторной работы составлен на основании аналогичного задания по курсу «Машинное обучение» на портале онлайн обучения Coursera.org (профессор Эндрю Ын, Стэнфордский университет - https://ru.wikipedia.org/wiki/Ын,_Эндрю)

Указание: Вторая часть задания является факультативной и не требуется для зачёта по лабораторной работе.

ex3spam.m – скрипт, реализующий пошаговое выполнение второй части задания по разделу «Статистические методы обучения. Метод опорных векторов»
spamTrain.mat – набор данных для обучения классификатора спама;
spamTest.mat – набор данных для проверки классификатора спама;
emailSample1.txt – 1-е электронное письмо;
emailSample2.txt – 2-е электронное письмо;
spamSample1.txt – 1-е спам-письмо;
spamSample2.txt – 2-е спам-письмо;
vocab.txt – словарь (список слов);
getVocabList.m – программа, выполняющая загрузку списка слов;
porterStemmer.m – функция для определения корня слова;
readFile.m – считывает файл и преобразует его в строковый тип;
* *processEmail.m* – функция для предварительной обработки писем;
* *emailFeatures.m* – функция преобразования текста письма в вектор свойств.

Функции, отмеченные знаком *, следует написать самостоятельно.

В этой Лабораторной работе следует использовать скрипты *ex3.m* и *ex3spam.m*, не изменяя их (за исключением тестирования работы алгоритма для различных значений параметра *C*). В указанных скриптах (коротких программах) подготовлены обращения к исходным данным. Далее производится вызов функций, написанных Вами, и отображаются результаты вычислений. Необходимо дописать функции в файлах по инструкциям упражнения.

1 Метод опорных векторов

Как было сказано выше, Вы будете использовать метод опорных векторов для классификации 2-мерных наборов данных. В ходе работы над упражнением вы изучите собственно метод, а также научитесь использовать с SVM ядро Гаусса. Уже написанный код программы *ex3.m*, поможет Вам при решении первой части упражнения.

Указание: Вы располагаете также написанными ранее программами, которые здесь можно, в случае необходимости, использовать.

1.1 Набор данных 1

Начнем работу с обработки первого набора данных, который может быть разделен с помощью линейной границы. Скрипт программы *ex3.m* автоматически построит график, отображающий эти данные (рис. 1). На визуализированной картине данных видно естественное разделение между «положительными» примерами (обозначенными как «+») и «отрицательными» (обозначенными как «о»). Обратите внимание на присутствие «положительного» примера (*x = 0.1; y = 4.1*), расположенного вдали от области расположения основного числа «положительных» примеров. Выполняя задание в этой части упражнения, можно понять, как такое расположение влияет на проведение прямолинейной границы раздела 2-х областей в процессе применения метода опорных векторов.

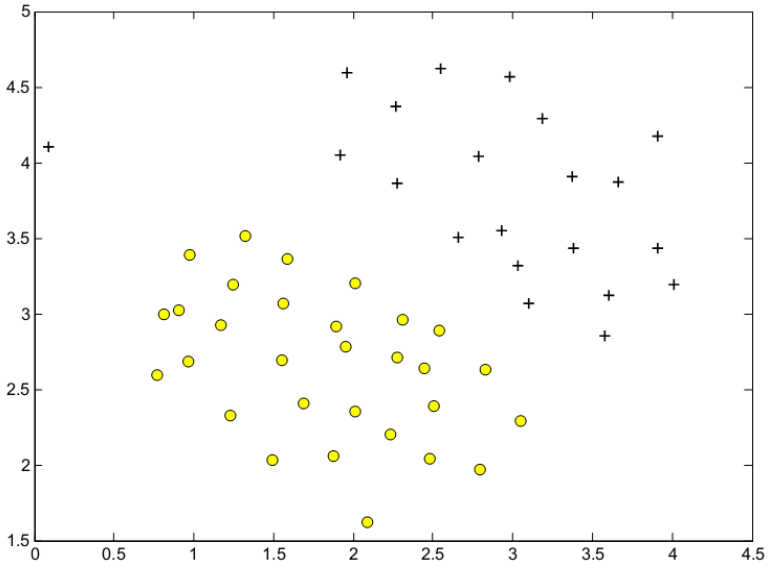

Рис. 1. Первый набор данных

Следует использовать различные значения параметра *C*, который принимает положительные значения, влияющие на значение штрафной функции при ошибочной классификации обучающего примера. Более высокие значения параметра *C* соответствуют требованию наиболее точной классификации *всех* обучающих примеров. В этом смысле этот параметр аналогичен обратным значением параметра регуляризации λ, который применялся в логистической регрессии.

Следующая часть скрипта в программе *ex3.m* производит обучение алгоритма SVM (с параметром *C* = 1), при этом запускается код программы *svmTrain.m*. Если параметр *C* равен 1, исполнение кода метода опорных векторов определит границу раздела 2-х областей в промежутке между ними, а также не классифицирует «+» значение, расположенное в самой левой части графика (рис. 2).

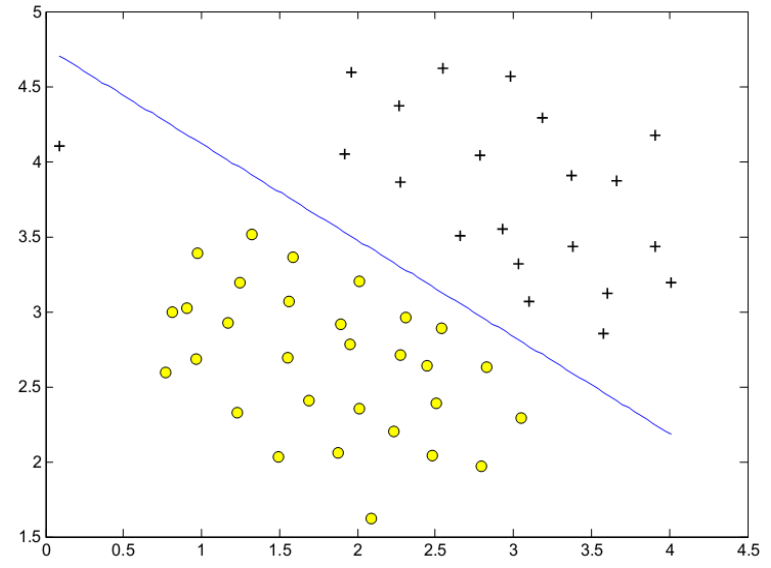

Рис. 2. Граница раздела при *C=1* (первый набор данных)

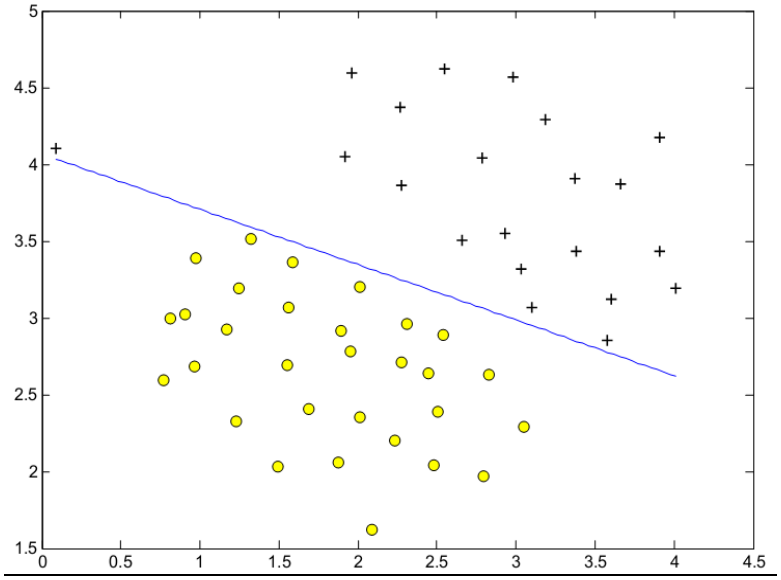

Рис. 3. Граница раздела при *C=1000* (первый набор данных)

Указание: Большинство программных пакетов для метода опорных векторов (включая *svmTrain.m),* автоматически добавляют значение $x_0 = 1$ и параметр θ_0. Поэтому нет необходимости добавлять это свойство самостоятельно. В пакете Matlab код должен работать с обучающими примерами $x \in R^n$ (вместо $x \in R^{n+1}$); например, в первый наборе данных $x \in R^2$.

Задание: Следует протестировать различные значения параметра C для представленного набора данных и оценить его влияние на качество классификации. В частности, если поменять значение параметра C в коде на 100 и запустить обучение SVM заново, то можно обнаружить, что метод опорных векторов классифицирует каждый пример корректно, но граница раздела не будет соответствовать «естественному» ожиданию для текущего набора данных (рис. 3).

1.2 Метод опорных векторов с ядром Гаусса

В этой части упражнения демонстрируется применение метода опорных векторов для нелинейной классификации данных. В частности, предстоит применить SVM с ядром Гаусса в ситуации, когда линейное разделение невозможно.

1.2.1 Ядро Гаусса

Для нахождения нелинейных границ с помощью метода опорных векторов, необходимо запрограммировать функцию, реализующую применение ядра Гаусса. Под ядром Гаусса подразумевается функция, определяющая сходство пары образцов на основании оценки расстояния между ними $(x^{(i)}, x^{(j)})$. Ядро Гаусса регулируется параметром σ, который определяет, насколько быстро уменьшается «схожесть» двух примеров при увеличении расстояния между ними.

Теперь Вы можете завершить код программы *gaussianKernel.m,* требующийся для расчета ядра Гаусса (расстояния) между 2-мя примерами $(x^{(i)}, x^{(j)})$. Функция ядра Гаусса определена ниже:

$$K_{gaussian}(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{k=1}^{n}(x_k^{(i)} - x_k^{(j)})^2}{2\sigma^2}\right)$$

Как только вы закончите написание программы *gaussianKernel.m*, скрипт программы *ex3.m* проверит Вашу функцию нахождения ядра на 2-х представленных примерах, в ответе Вы должны будете увидеть следующее значение: 0.324652.

1.2.2 Набор данных 2

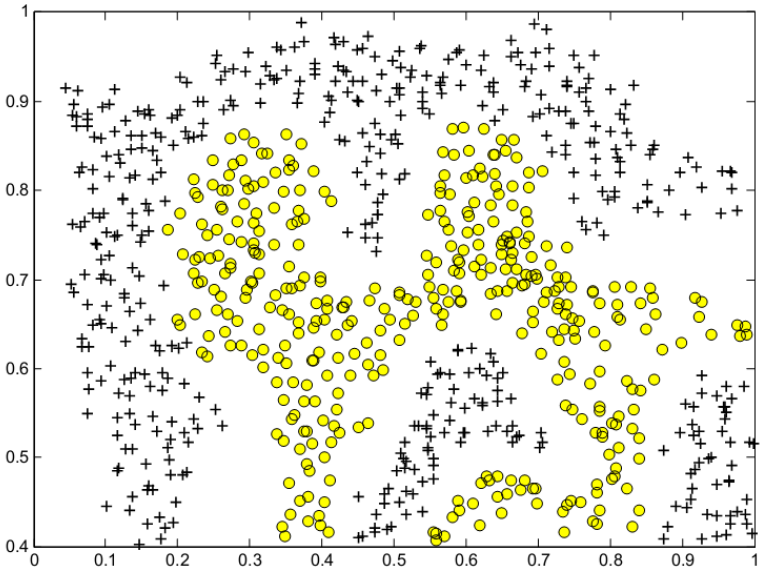

Рис. 4. Второй набор данных

Следующая часть скрипта в программе *ex3.m* загрузит и отобразит набор данных для 2-й части упражнения. Нетрудно видеть, что невозможно провести линейную границу раздела, которая бы отделила «положительные» примеры от «отрицательные».

Задание: Используя SVM с ядром Гаусса, построить нелинейную границу раздела, которая наиболее точно подойдет для классификации предоставленного набора данных.

Если Вы правильно написали программу расчета ядра Гаусса, программа *ex3.m* продолжит обучение алгоритма, используя 2-й набор данных. На графике (рис. 5) изображена граница раздела 2-х областей, найденная с помощью метода опорных векторов с ядром Гаусса.

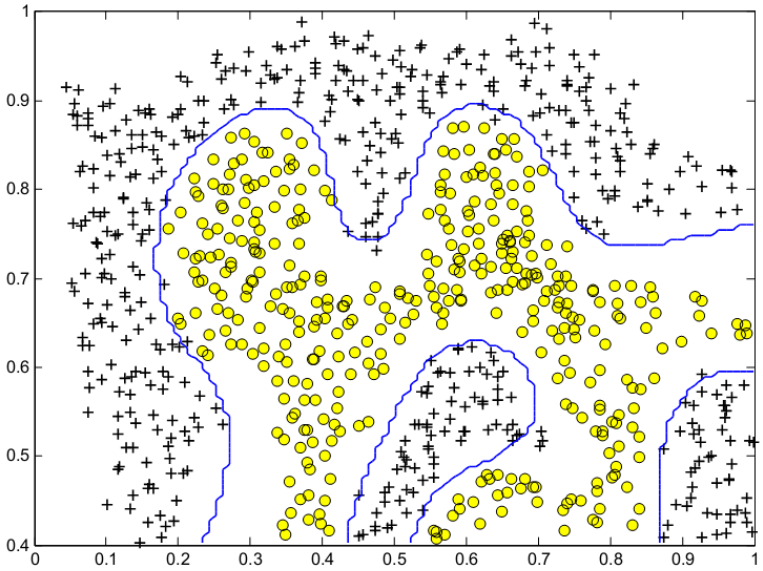

Рис. 5. Метод опорных векторов с ядром Гаусса,
граница раздела для второй выборки

1.2.3 Набор данных 3

В этой части упражнения, Вы усовершенствуете свои навыки по использованию метода опорных векторов с ядром Гаусса для проведения нелинейной классификации данных. Следующая часть скрипта программы *ex3.m* загрузит и отобразит график с набором данных для этой части упражнения (рис. 6).

Исходные данные в файле *ex3data3.m* описываются переменными *X, y, Xval, yval*. Код программы *ex3.m* обучает классификатор *SVM*, используя обучающий набор данных (X, y), а также параметры, подгружаемые из программы *dataset3Params.m*.

Задание: Определить оптимальные параметры C и σ, используя метод перекрестной проверки с помощью множества *Xval, yval*.

Как для C, так и для σ, рекомендуется брать значения с увеличивающимся шагом (например, 0.01; 0.03; 0.1; 0.3; 1; 3; 10; 30). Заметьте, что следует перебрать все возможные пары значений C и σ (например, C = *0.3* и σ = *0.1*). Если, например, Вы попробуете перебрать все перечисленные выше значения для параметра C и σ^2, то при обучении и пересчете программы классификации данных, Вы получите 64 разных модели границы раздела 2 областей. После того, как Вы определите наилучшие параметры C и σ, Вам будет необходимо изменить их в начальном коде программы *dataset3Params.m*. Используя новые параметры C и σ, программа расчета *SVM* возвращает оптимальную границу раздела 2-х областей, представленную на рис. 7.

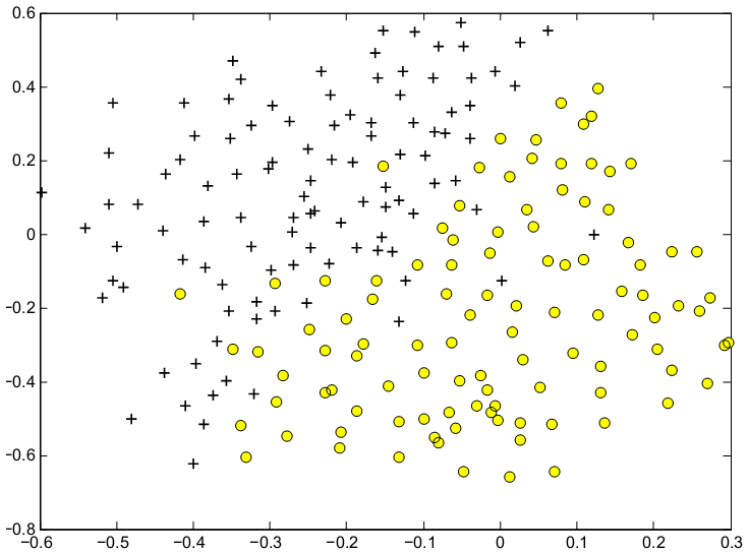

Рис. 6. Третий набор данных

Указание: При выполнении перекрестной проверки данных, для определения оптимальных параметров C и σ Вам необходимо рассчитать ошибку для набора данных, выбранных для проверки. Ошибка определяет долю примеров для перекрестной проверки, классифицированных неправильно. В Matlab данную ошибку можно вычислить, используя функцию нахождения среднего `mean(double(predictions~=yval))`, где

```
predictions = svmPredict(model, Xval)
```

- вектор всех предсказаний, полученных с помощью метода опорных векторов, а значения *yval* – значения из набора данных для проведения перекрестной проверки. Программа *svmPredict.m* в методических целях содержится в каталоге файлов, образующих данное лабораторное задание.

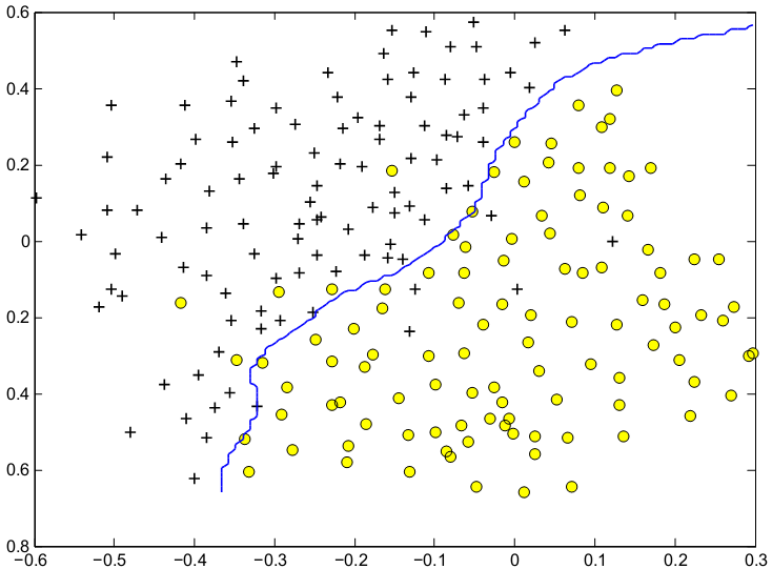

Рис. 7. Метод опорных векторов с ядром Гаусса,
граница раздела для третьего набора данных

2 Классификатор спама (**факультативно, не требует выполнения!**)

Большинство mail-сервисов имеют «*спам*-фильтры», т.е. программы которые с достаточно высокой точностью классифицируют электронные письма на *спам* и *не спам*. В этой части упражнения, Вы будете использовать SVM для создания своего собственного *спам*-фильтра. Ваша задача будет заключаться в обучении классификатора электронных писем *x*, для определения: *спам* это ($y = 1$) или же *не спам* ($y = 0$). В частности, Вам будет необходимо преобразовать каждое письмо в вектор свойств, где $x \in R^n$. В этой части упражнения, Вы поймете, как такой вектор свойств может быть сформирован из e-mail письма.

В оставшейся части упражнения, Вы будете использовать программу *ex6_spam.m*. Данные для этой части упражнения взяты с сайта компании *The SpamAssassin Public Corpus2*. Для выполнения поставленной задачи, Вам потребуется использование лишь самого текста электронного письма (не принимая во внимание его заголовок).

1.2 Предварительная обработка электронных писем

```
> Anyone knows how much it costs to host a web portal ?
>
Well, it depends on how many visitors youre expecting.  This can be
anywhere from less than 10 bucks a month to a couple of $100.  You
should checkout http://www.rackspace.com/ or perhaps Amazon EC2 if
youre running something big..

To unsubscribe yourself from this mailing list, send an email to:
groupname-unsubscribe@egroups.com
```

Рис. 8. Пример письма

Перед началом написания кода программы, полезно проанализировать исходные данные, подготовленные для этой части упражнения. На рис. 8 показано обычное письмо (на английском языке), содержащее адрес *Единого указателя ресурсов* URL (англ. *Uniform Resource Locator),* e-mail адрес (в конце письма), числовые данные и суммы в долларах. Несмотря на то, что большое

количество электронных писем может иметь однотипное содержание (например, какие-либо числовые данные, URL–адреса или же адреса электронной почты), специфическая информация в письме (например, адрес URL или же количественные данные) будут скорее всего отличаться. Таким образом, метод решения данной проблемы, применяющийся для обработки писем, заключается в «нормализации» числительных, цен и адресов таким образом, чтобы URL-данные, различные числа и т.д. - обрабатывались одинаково, вне зависимости от их содержания или значений. Например, мы можем заменить каждую URL-страницу в электронном письме универсальной строчкой «httpaddr», указывающей на наличие данного атрибута в письме. Этот прием позволяет классификатору делать вывод исходя из наличия любого адреса URL в письме. Это улучшает работу классификатора в целом, поскольку «спаммеры» очень часто меняют URL–адреса. Таким образом, вероятность того, что в письме снова попадётся уже знакомый адрес – очень мала.

В программе *processEmail.m* выполнены следующие шаги предварительной обработки и нормализации mail-писем:

- понижение регистра: например, IndIcaTE после обработки имеет вид – Indicate;
- удаление всех HTML тэгов: после этого остается лишь текст письма;
- «нормализация» URL – адресов: все URL-адреса заменены текстом «httpaddr»;
- «нормализация» e-mail адресов: все e-mail адреса заменены текстом «emailaddr»;
- «нормализация» чисел: все числа заменены на текст «number»;
- «нормализация» денежного эквивалента ($): т.е. обозначения «$» заменены на текст «dollar»;
- определение корня слова: морфологический поиск - выделение корня слова позволяет поисковой машине вести поиск слова не в строго заданном виде, но и во всех его морфологических формах. Например, слова «discount», «discounts», «discounted» and «discounting» были заменены словом «discount»;
- удаление символов (знаков препинания и символов пунктуации) и замена их единичным пробелом.

Результат такого предварительного преобразования исходных данных изображен на рис. 9. С такой формой письма легче работать и можно быстро выделить любое свойство.

```
anyon know how much it cost to host a web portal well it depend on how
mani visitor your expect thi can be anywher from less than number buck
a month to a coupl of dollarnumb you should checkout httpaddr or perhap
amazon ecnumb if your run someth big to unsubscrib yourself from thi
mail list send an email to emailaddr
```

Рис. 9. Обработанное письмо

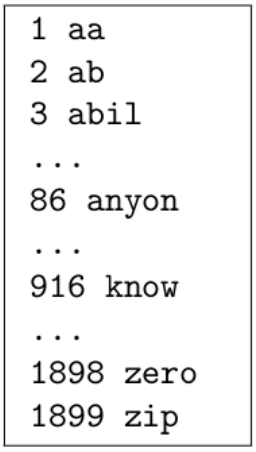

Рис. 10. Словарь

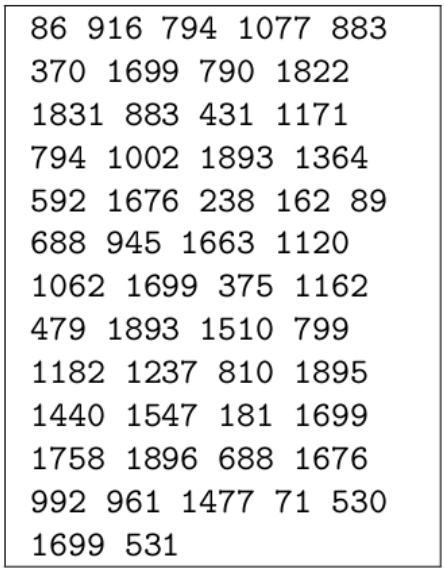

Рис. 11. Индексы слов для данного письма

2.1.1 Список слов

После предварительной обработки электронных писем, у нас есть список слов (рис. 9) для каждого письма. Следующий шаг – выбрать, какие слова мы бы хотели использовать для нашего классификатора, а какие нет. Для этого упражнения, выбраны наиболее часто встречающиеся слова (см. файл «*vocabulary list*»). Слова, которые редко встречаются в исходных данных, присутствуют лишь в нескольких письмах, могут вызвать переобучение модели наших исходных данных. Полный список слов (рис. 10) представлен в файле данных *vocab.txt*. Список слов в данном задании составлен таким образом, что каждое подобранное слово встречается в блоке спам – писем, по меньшей мере, 100 раз. В результате получен список из 1899 слов. На практике, такой список обычно содержит от 10 до 50 тыс. слов. Теперь мы можем отобразить каждое слово в виде индекса, характеризующего положение этого слова в общем списке слов, таким образом, получим список индексов. На рис. 11 изображен список индексов (обозначающих различные слова) для одного из письма. А именно, в нашем примере, слово «anyone» было изначально преобразовано в слово «anyon» (пример морфологического преобразования) и затем, этому слову присвоили индекс 86 в списке всех слов.

Ваша задача теперь завершить код программы *processEmail.m* для создания такой индексации каждого слова. В коде присутствует строка *str*, в которой сохранено отдельное слово, полученное после обработки письма. Вам необходимо найти это слово в общем списке (*vocabList.txt*) и также определить – присутствует оно в нем или нет. Ели слово в списке представлено, то Вам необходимо присвоить индекс этого слова переменной *word_indices*. Если его там нет, Вы можете его пропустить. Как только Вы закончите работу над программой *processEmail.m*, скрипт программы *ex3_spam.m* применит его к исходным данным, и Вы должны будете увидеть результат, схожий с результатом, представленным на рис. 9 и 11.

Указание: В программе Matlab имеется возможность производить сравнение 2 строчек с помощью специальной функции *strcmp*. Например, функция *strcmp (str1, str2)* вернет «1» только в том случае, если эти строки одинаковые. В исходном коде список слов представлен в виде матрицы элементов, содержащей слова из списка. В программе Matlab матрица элементов представлена в виде обычного массива данных (например, вектора), за исключением того, что элементы массива могут быть символьными переменными (каковыми они не могут быть в обычной

матрице/векторе), поэтому Вам необходимо использовать фигурные скобки, при обращении к индексам этих слов, вместо обычных квадратных. Например, для того, чтобы вывести слово под индексом *i*, Вы можете использовать функцию *vocabList{i}*. Кроме того, можно использовать функцию *length(vocabList)* для получения количества слов в общем списке.

2.2 Выделение свойств (признаков) из электронных писем

Теперь предстоит выполнить программу создания вектора свойств, которая преобразует каждое электронное письмо в вектор R^n. В этой части упражнения Вы будете использовать переменную n равную количеству слов в общем списке (словаре). А именно, свойство $x_i \epsilon \{0,1\}$ для произвольного письма показывает, присутствует ли i-ое слово из общего списка (словаря) в самом письме или нет. Таким образом, $x_i = 1$, если i-ое слово присутствует в письме и $x_i = 0$ – если оно отсутствует.

Соответственно, для типичного письма, это свойство будет выглядеть следующим образом:

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in Rn.$$

Теперь Вам необходимо завершить написание кода программы *emailFeatures.m* для создания вектора свойств письма из полученных индексов соответствующих слов в файле *word_indices.txt*. После выполнения программы *emailFeatures.m,* следующая часть скрипта *ex6_spam.m* применит Ваш код для обработки файла с индексами слов из электронного письма. Вы увидите, что вектор свойств состоит из 1899 элементов и 45 из них не равны нулю.

2.3 Обучение SVM для создания спам-классификатора

После того, как Вы завершили редактирование функции *emailFeature.m*, следующим шагом программа *ex3_spam.m* загрузит предварительно подготовленные исходные данные для обучения классификатора на основе SVM. Файл *spamTrain.mat* содержит 4000 обучающих примеров, а файл *spamTest.mat* содержит 1000 примеров для тестирования нашего классификатора. Каждое письмо было обработано с помощью программ *processEmail.m* и emailFeatures.m и преобразовано в вектор свойств, где $x^{(i)} \epsilon R^{1899}$.

После загрузки данных, программа *ex3_spam.m* начнет обучение классификатора SVM по определению спам ($y = 1$) и не спам ($y = 0$). После завершения данной части упражнения, Вы должны будет получить точность выполнения обучения классификатора, равную 99.8%, а также точность тестирования нашего классификатора, равную 98.5%.

2.3 Определение наиболее часто встречающихся слов

```
our click remov guarante visit basenumb dollar will price pleas nbsp
most lo ga dollarnumb
```

Рис. 12. Наиболее вероятные спам-слова

Для лучшего понимания работы спам-классификатора можно рассмотреть параметры с целью понимания, какие слова, по решению классификатора, используются наиболее часто в спам-

письмах. Для этого программа *ex3_spam.m* находит наибольшие положительные значения параметров и отображает соответствующие слова (рис. 12). Таким образом, если электронное письмо содержит такие слова, как «guarantee», «remove», «dollar» и «price» (наиболее часто встречающиеся, см. рис. 12), то такое письмо будет с большой вероятностью классифицировано как спам.