Программа анализа направления развития ядра Linux на основе сетей Кохонена

Постановка задачи

- Классифицировать компании принимающие участие в разработке ядра Linux по следующим критериям:
  - *разрабатывают патчи для arm;*
  - *разрабатывают патчи для x86;*
  - *разрабатывают драйвера*.
- В качестве цифровой оценки - берем количество патчей.
- Для группировки по компании берем e-mail того, кто разработал патч.
- В качестве инструмента - карты Кохонена.
- Классифицируем последние 1000 изменений.

СЕТИ КОХОНЕНА

- Сети Кохонена относятся к *самоорганизующимся* нейронным сетям. Самоорганизующаяся сеть позволяет выявлять группы входных векторов, обладающих некоторыми общими свойствами. При этом выделяют :
  - *Сети с неупорядоченными нейронами* (часто называемые *слоями Кохонена*)
  - *Сети с упорядочением нейронов* (часто называемые *самоорганизующимися картами*, или *SOM – self-organizing map*).
- Сеть Кохонена, в отличие от многослойной нейронной сети, очень проста; она представляет собой два слоя: входной и выходной.

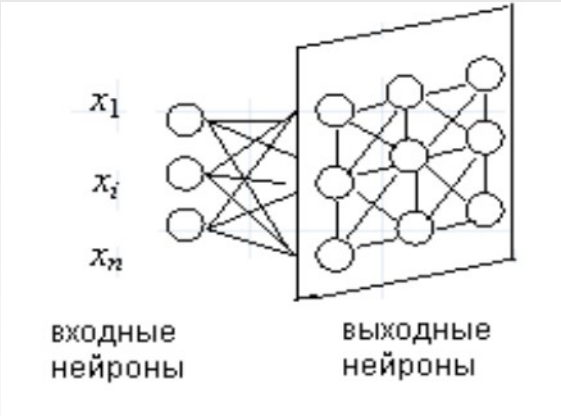

Рис. 1. Сеть Кохонена

СЕТИ КОХОНЕНА

- В качестве нейронов сети Кохонена применяются линейные взвешенные сумматоры. Каждый j- ый нейрон описывается вектором весов $w_j = (w_{1j},\ w_{2j},\ \ldots,\ w_{mj})$, где m- число компонентов входных векторов.
- Входной вектор имеет вид $x_i = (x_{i1},\ x_{i2},\ \ldots,\ x_{im})$.

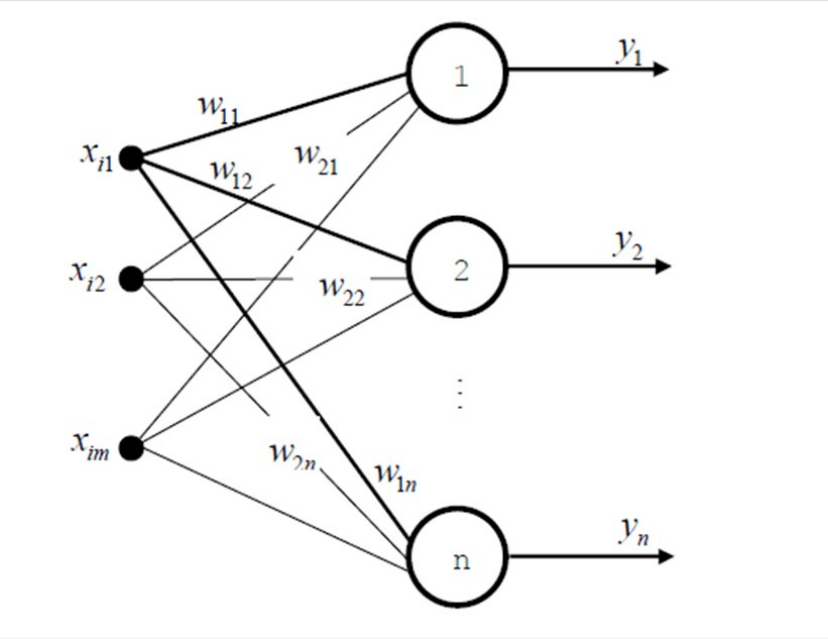

Рис. 2. Структура сети Кохонена

ОБУЧЕНИЕ СЕТИ КОХОНЕНА

- В процессе последовательной подачи на вход сети обучающих примеров определяется наиболее схожий нейрон . Этот нейрон объявляется победителем и является центром при подстройке весов у соседних нейронов.
- Обучение при этом заключается не в минимизации ошибки, а в постройке весов для наибольшего совпадения с входными данными и продолжается до тех пор, пока не пройдет заданное число итераций.
- В результате обучения сети необходимо определить меру соседства нейронов, т.е. окрестность нейрона-победителя. Для этого должно быть выполнено соотношение:

$$d\left(x,\ w_j\right)=\min_{1\le i\le n} * d\left(x,\quad w_i\right),$$

где n – количество нейронов, j – номер нейрона победителя, d(x, w) – расстояние (в смысле выбранной метрики) между векторами x и w.

- Чаще всего в качестве меры расстояния используется *евклидова мера:*

$$d\left(x,\quad w_i\right)=\left(x-w_i\right)=\sqrt{\sum_{j=1}^{n}\left(x_j-w_{ij}\right)^2}.$$

ОБУЧЕНИЕ СЕТИ КОХОНЕНА

- Веса нейрона-победителя и всех нейронов, лежащих в пределах его радиуса обучения, подвергаются обучению (адаптации) по *правилу Кохонена:*

$$w_i^{(k+1)} = w_i^{(k)} + \eta_i^{(k)}\left(x - w_i^{(k)}\right),$$

где x – входной вектор, k – номер цикла обучения, $\eta_i^{(k)}$ - коэффициент скорости обучения i-го нейрона из радиуса обучения в k-ом цикле обучения.

- Веса нейронов, находящийся за пределами радиуса обучения не изменяются.
- Коэффициент скорости обучения разбивается на две части: функцию соседства $\eta_i(d,\ k)$ и функцию скорости обучения $a(k)$:

$$\eta_i^{(k)} = \eta_i(d,\ k)a(k).$$

- В качестве функции соседства применяется Гауссовская функция:

$$\eta_i(d,\ k) = e^{\frac{-d_i}{2\sigma(k)}},$$

где $\sigma(k)$- убывающая функция от номера цикла обучения.

- Функция скорости обучения: $a(k) = \frac{A}{k + B}$, где A и B — константы.

КАРТЫ КОХОНЕНА

- Карта Кохонена состоит из ячеек прямоугольной или шестиугольной (рис. 2) формы.

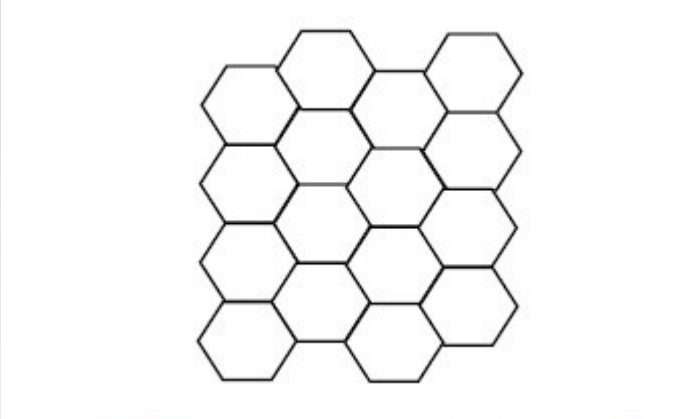

Рис. 3. Шестиугольные ячейки сети Кохонена

- Каждой ячейке соответствует нейрон сети Кохонена.
- Двумерная карта Кохонена отражает на плоскости близость *многомерных* векторов признаков, то есть сходство объектов.

КАРТЫ КОХОНЕНА

- Шестиугольные ячейки более корректно отображают расстояние между объектами на карте, т. к. для этих ячеек расстояние между центрами смежных ячеек одинаковы (рис. 3).

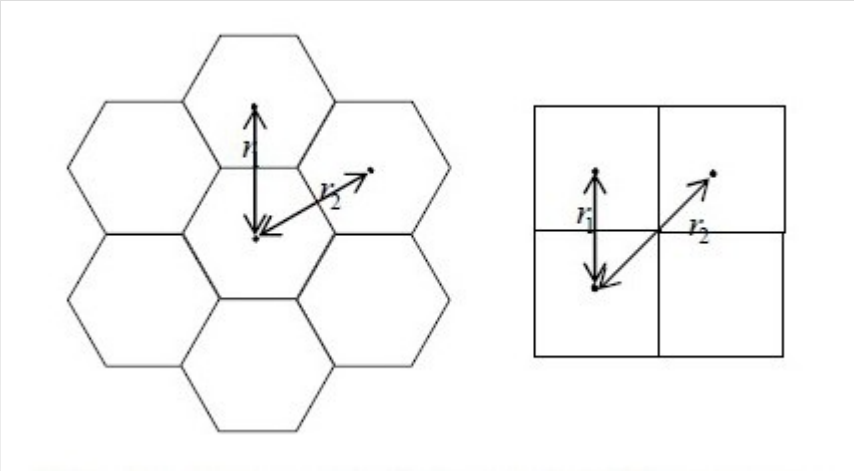

Рис. 4. Шестиугольные и прямоугольные ячейки

- Для того, чтобы проанализировать по каким параметрам проявляется сходство объектов, используется раскраска карт Кохонена.
- Для этого строится столько карт, сколько параметров анализируется.
- Ячейки карты раскрашиваются в разные цвета (или оттенки серого цвета) в зависимости от значения параметров, соответствующих каждой ячейке.

Data Mining

- Data Mining - это процесс выделения из данных неявной и неструктурированной информации и представления ее в виде, пригодном для использования.
- Data Mining является мультидисциплинарной областью.

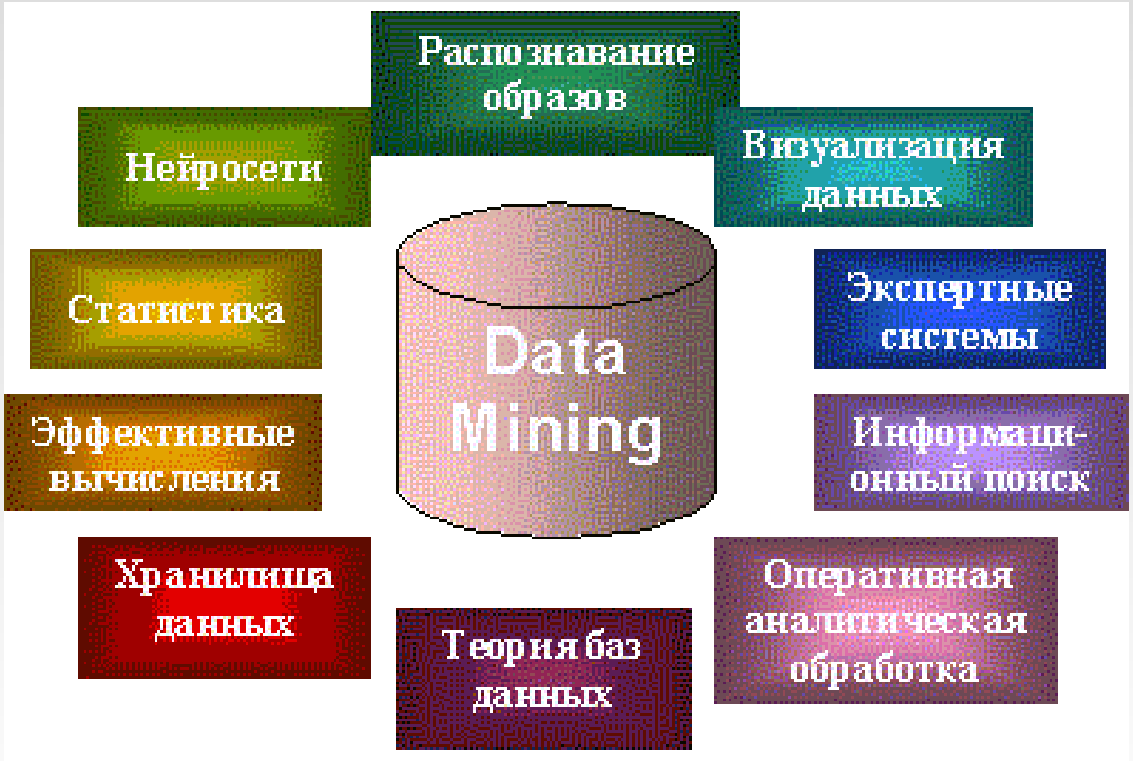

Рис. 5. База развития Data Mining

Задачи, решаемые методами Data Mining:

- *Классификация – это отнесение объектов к одному из заранее известных классов.*
- *Регрессия – установление зависимости непрерывных выходных переменных от входных значений.*
- *Кластеризация – объекты внутри кластера должны быть 'похожими' друг на друга и отличаться от объектов, вошедших в другие кластеры.*
- *Ассоциация – нахождение зависимости, что из события X следует событие Y.*
- *Последовательные шаблоны – установление закономерностей между связанными во времени событиями.*
- *Анализ отклонений – выявления наиболее нехарактерных шаблонов.*

Наиболее популярные алгоритмы:

- *Деревья решений – алгоритм C4.5.*
- *Искусственные нейронные сети – многослойный персептрон, обучение при помощи алгоритма обратного распространения ошибки.*
- *Линейная регрессия – классическая линейная модель.*
- *Самоорганизующиеся карты Кохонена.*
- *Ассоциативные правила – алгоритм Apriori.*

Пример №1 Классификация должников

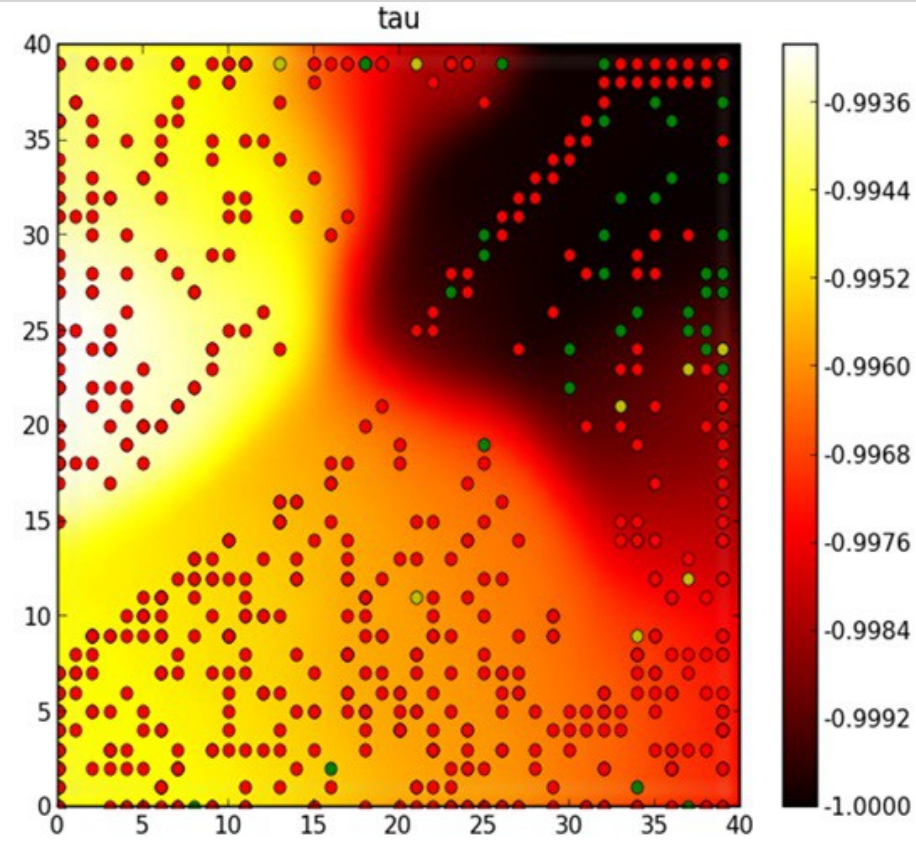

Рис. 6. Результаты погашения просроченной задолженности на карте Кохонена

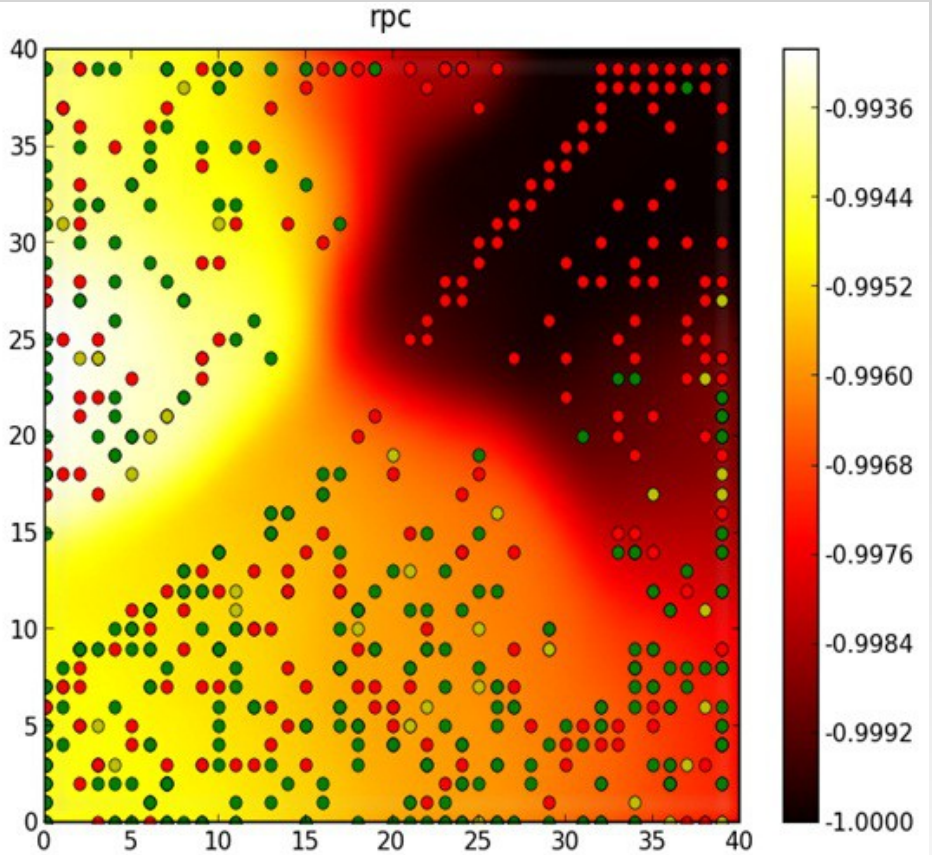

Рис. 7. Right Person Contacts (RPC) - успешные контакты с должниками на карте Кохонена.

Пример №2 Кластеризация продуктов

Таблица 1. Содержание белков, углеводов и жиров в продуктах (в 100 гр.)

	Продукт	Белки	Углеводы	Жиры
1	Яблоки (Apples)	0.4	11.8	0.1
2	Авокадо (Avocado)	1.9	1.9	19.5
3	Бананы (Bananas)	1.2	23.2	0.3
4	Говяжий стейк (Beef Steak)	20.9	0	7.9
5	Биг Мак (Big Mac)	13	19	11
6	Бразильские орехи (Brazil Nuts)	15.5	2.9	68.3
7	Хлеб (Bread)	10.5	37	3.2
8	Масло (Butter)	1	0	81
9	Сыр (Cheese)	25	0.1	34.4
10	Ватрушка (Cheesecake)	6.4	28.2	22.7
11	Печенье (Cookies)	5.7	58.7	29.3
12	Кукурузные хлопья (Cornflakes)	7	84	0.9
13	Яйца (Eggs)	12.5	0	10.8
14	Жареный цыпленок (Fried Chicken)	17	7	20
15	Жаркое (Fries)	3	36	13
16	Горячий шоколад (Hot Chocolate)	3.8	19.4	10.2
17	Пепперони (Pepperoni)	20.9	5.1	38.3
18	Пицца (Pizza)	12.5	30	11
19	Пирог со свининой (Pork Pie)	10.1	27.3	24.2
20	Картофель (Potatoes)	1.7	16.1	0.3
21	Рис (Rice)	6.9	74	2.8
22	Жаркое из курицы (Roast Chicken)	26.1	0.3	5.8
23	Сахар (Sugar)	0	95.1	0
24	Стейк из тунца (Tuna Steak)	25.6	0	0.5
25	Вода (Water)	0	0	0

Пример №2 Кластеризация продуктов

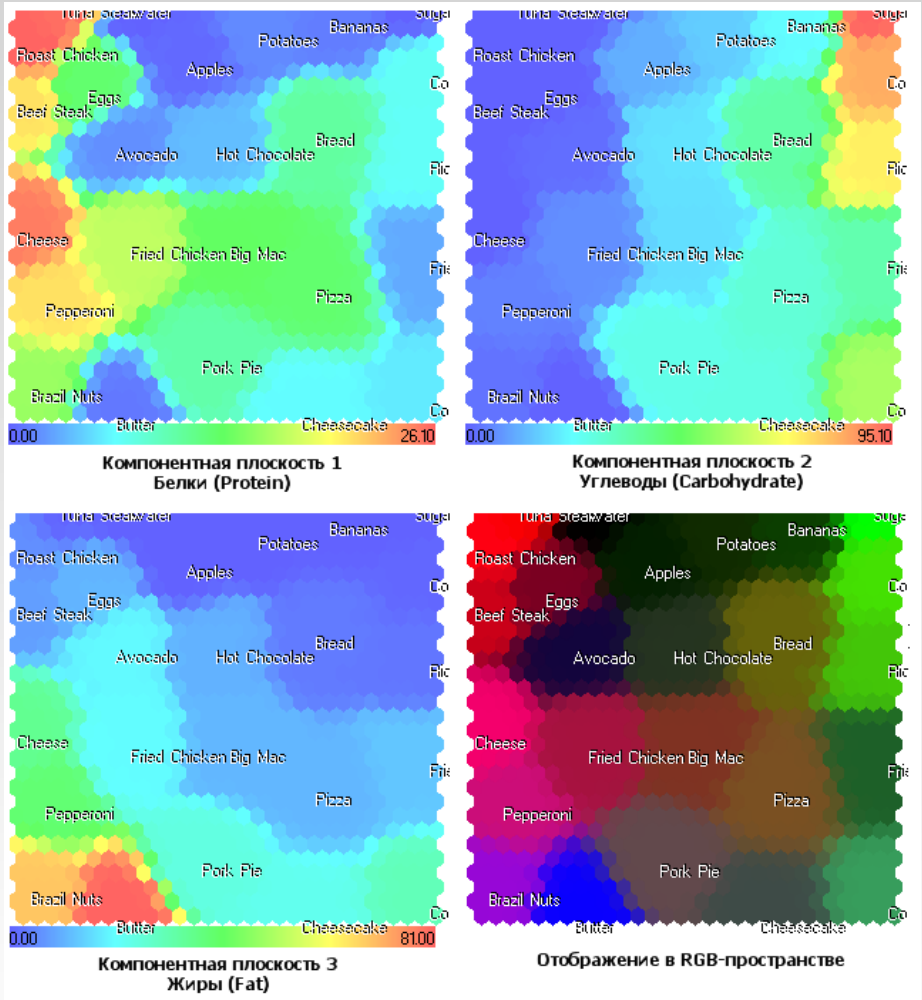

Рис. 8. Продукты. Компонентные плоскости и отображение в RGB-пространстве

Пример №3 Четырехмерный случай. Ирисы Фишера.

- Ирисы Фишера представляют собой данные о 150 экземплярах ириса, по 50 экземпляров каждого из трех сортов: **iris setosa**, **iris viginica** и **iris versicolor**:

Рис. 9. Ирисы Фишера

- Для каждого экземпляра есть 4 характеристики:
  - *Длина чашелистика (sepal length);*
  - *Ширина чашелистика (sepal width);*
  - *Длина лепестка (petal length);*
  - *Ширина лепестка (petal width).*

Пример №3 Четырехмерный случай. Ирисы Фишера.

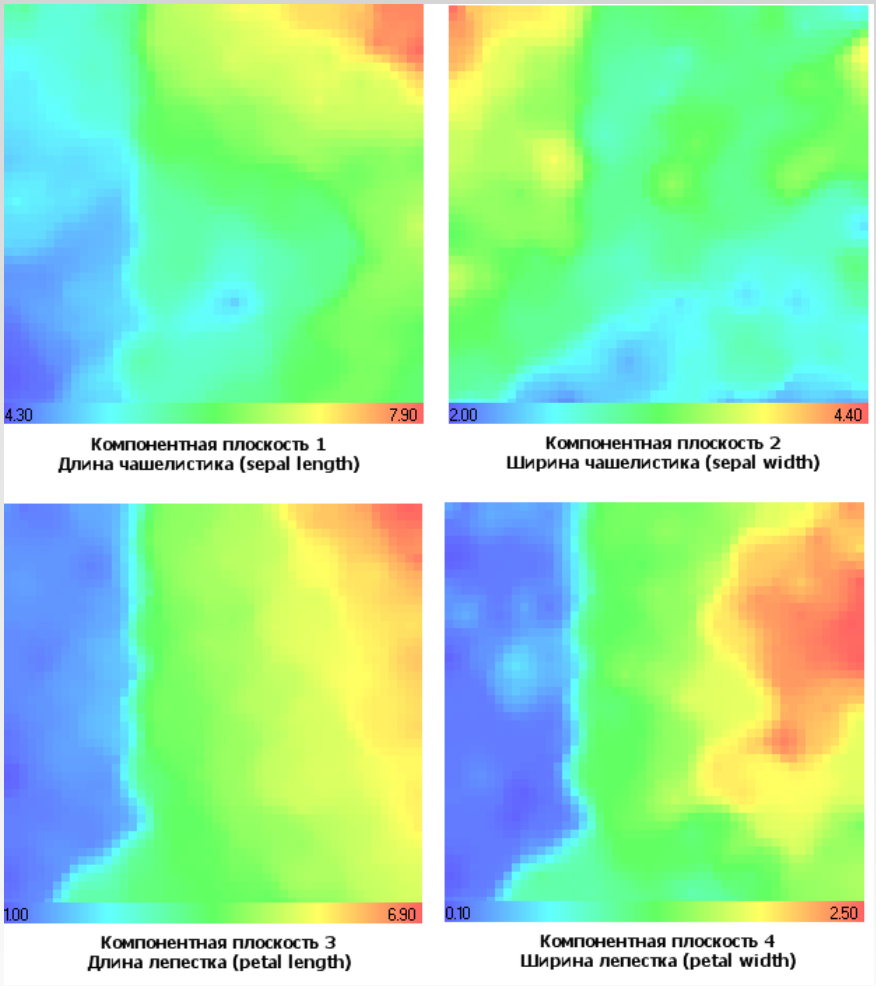

Рис. 10. Ирисы Фишера. Представление в виде компонентных плоскостей

Рис. 11. Карта Кохонена для ирисов Фишера, построенная в виде проекций в цветовой базис CMYK

Структура программы

С помощью скрипта git-parser.py из исходников ядра Linux выбираем последние 1000 изменений. Для каждого разработчика выбираем три параметра: количество изменений, внесенных для ARM, X86 и драйверов. Полученный список сохраняем в файле d_list.xml

Загружаем данные из xml-файла в вектор m_training_sets_array.

На вход метода Train подается вектор из 3-х элементов, и, в соответствии с алгоритмом обучения сети Кохонена, выбирается нейрон-победитель – наиболее близкий к вектору обучения – и рассчитывается радиус окрестности обучения. Производится корректировка весов в направлении выбранного вектора. Далее для каждого набора параметров алгоритм повторяется.

Программа содержит следующие классы

- CSOMNode – задет ячейку карты или нейрон, и выполняет:
  - Инициализация параметров ячейки сети;
  - Инициализацию весов случайными значениями;
  - Возвращает значение веса;
  - Возвращает координаты узла;
  - Расчет квадрата расстояния между весами и заданным вектором.
- CSOM выполняет:
  - Загрузку данных из xml-файла;
  - Выбор узла победителя;
  - Обучение нейронной сети;
  - Сохранение полученной карты.

Диаграмма классов

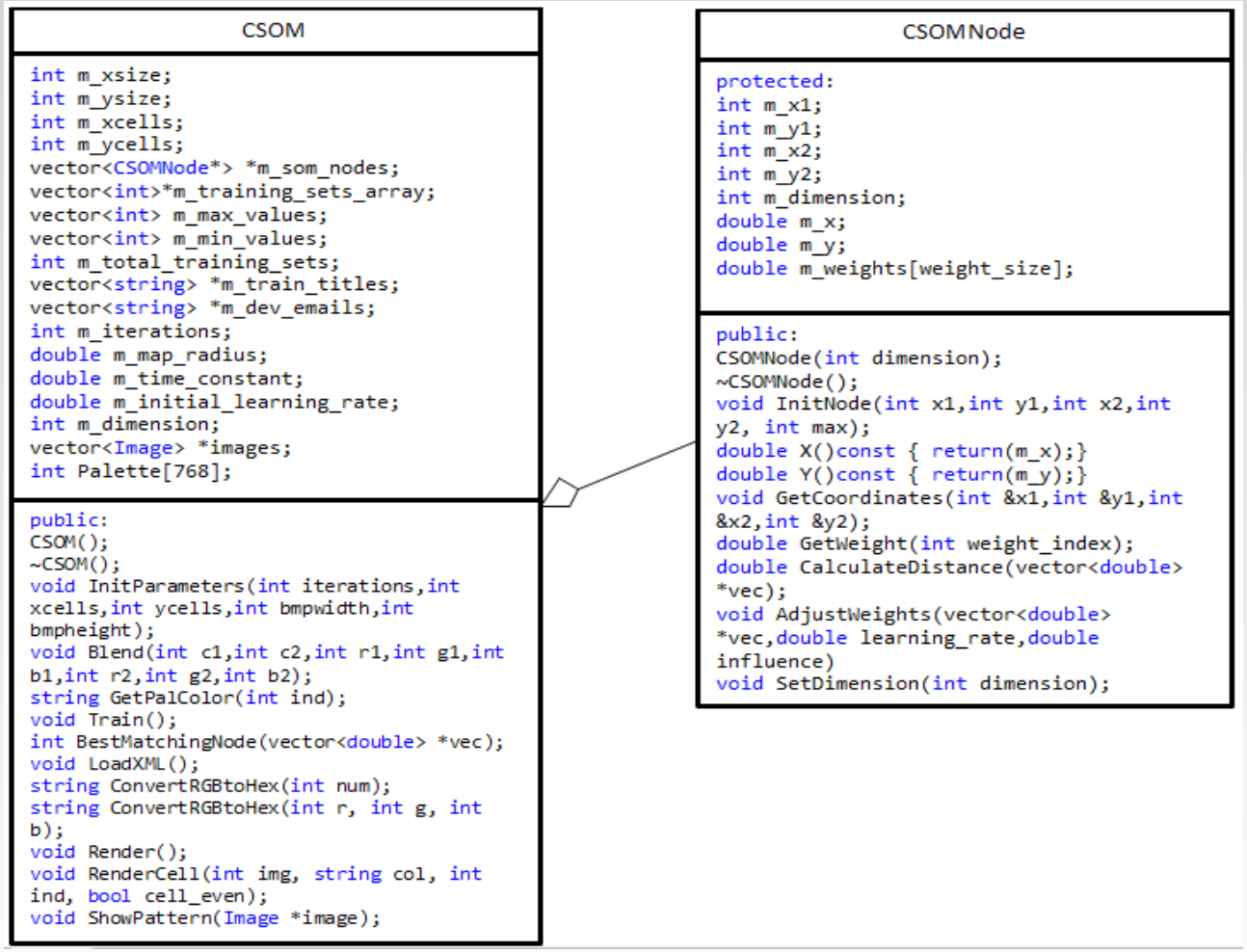

Рис. 12. Диаграмма классов

Содержание файла d_list.xml

```
<?xml version="1.0" ?>
<GitLinux>
        <Parameter par="arm"/>
        <Parameter par="x86"/>
        <Parameter par="drivers"/>
        <Developer Email=" benh@kernel.crashing.org" arm=" 0" x86=" 0" drivers=" 1" />
        <Developer Email=" linus.walleij@linaro.org" arm=" 3" x86=" 0" drivers=" 0" />
        <Developer Email=" nicolas.ferre@atmel.com" arm=" 8" x86=" 0" drivers=" 3" />
        <Developer Email=" JBottomley@Parallels.com" arm=" 0" x86=" 0" drivers=" 30" />
        <Developer Email=" avi@redhat.com" arm=" 0" x86=" 2" drivers=" 0" />
        <Developer Email=" jason@lakedaemon.net" arm=" 2" x86=" 0" drivers=" 0" />
        <Developer Email=" s.hauer@pengutronix.de" arm=" 2" x86=" 0" drivers=" 0" />
        <Developer Email=" linville@tuxdriver.com" arm=" 0" x86=" 0" drivers=" 33" />
        <Developer Email=" inki.dae@samsung.com" arm=" 0" x86=" 0" drivers=" 2" />
        <Developer Email=" vinod.koul@linux.intel.com" arm=" 0" x86=" 0" drivers=" 12" />
        <Developer Email=" gregkh@linuxfoundation.org" arm=" 0" x86=" 0" drivers=" 32" />
        <Developer Email=" tj@kernel.org" arm=" 0" x86=" 1" drivers=" 0" />
        <Developer Email=" hpa@linux.intel.com" arm=" 0" x86=" 22" drivers=" 1" />
        <Developer Email=" torvalds@linux-foundation.org" arm=" 1" x86=" 12" drivers=" 30" />
        <Developer Email=" mingo@kernel.org" arm=" 0" x86=" 12" drivers=" 0" />
        <Developer Email=" agk@redhat.com" arm=" 0" x86=" 0" drivers=" 4" />
        <Developer Email=" bskeggs@redhat.com" arm=" 0" x86=" 0" drivers=" 3" />
        <Developer Email=" airlied@redhat.com" arm=" 0" x86=" 0" drivers=" 20" />
        <Developer Email=" cjb@laptop.org" arm=" 0" x86=" 0" drivers=" 13" />
        <Developer Email=" haojian.zhuang@gmail.com" arm=" 4" x86=" 0" drivers=" 1" />
        <Developer Email=" rjw@sisk.pl" arm=" 9" x86=" 0" drivers=" 0" />
        <Developer Email=" broonie@opensource.wolfsonmicro.com" arm=" 0" x86=" 0" drivers=" 3" />
        <Developer Email=" grant.likely@secretlab.ca" arm=" 0" x86=" 0" drivers=" 15" />
        <Developer Email=" paul.gortmaker@windriver.com" arm=" 0" x86=" 0" drivers=" 2" />
```

Алгоритм метода Train:

- 1. На вход метода Train подается вектор из 3-х элементов.
- 2. Находятся минимальное и максимальное значения компонент обучающего набора «m_training_sets_array».
- 3. Из обучающего множества случайным образом находится индекс вектора и вектор обучения «datavector» устанавливается его значениями.

- 4. В соответствии с алгоритмом обучения сети Кохонена, выбирается нейрон-победитель, т.е. находится индекс узла сети наиболее близкий по значению к datavector, при этом рассчитывается радиус окрестности обучения.
- 5. Циклично для всех узлов сети рассчитывается квадрат расстояния до узла-победителя. Если узел оказывает внутри квадрата радиуса окрестности, то его вес корректируется в направлении выбранного вектора обучения.
- 6. Для каждого набора параметров алгоритм повторяется. Коэффициент обучения экспоненциально уменьшается с каждой итерацией.

Таблица исходных данных

№	Developer Email	Parameter		
		arm	x86	drivers
1	benh@kernel.crashing.org	0	0	1
2	linus.walleij@linaro.org	3	0	0
3	nicolas.ferre@atmel.com	8	0	3
4	JBottomley@Parallels.com	0	0	30
5	avi@redhat.com	0	2	0
6	jason@lakedaemon.net	2	0	0
7	s.hauer@pengutronix.de	2	0	0
8	linville@tuxdriver.com	0	0	33
9	inki.dae@samsung.com	0	0	2
10	vinod.koul@linux.intel.com	0	0	12
11	gregkh@linuxfoundation.org	0	0	32
12	tj@kernel.org	0	1	0
13	hpa@linux.intel.com	0	22	1
14	torvalds@linux-foundation.org	1	12	30
15	mingo@kernel.org	0	12	0
16	agk@redhat.com	0	0	4
17	bskeggs@redhat.com	0	0	3
18	airlied@redhat.com0	0	0	20
19	cjb@laptop.org	0	0	13
20	haojian.zhuang@gmail.com	4	0	1
21	rjw@sisk.pl	9	0	0
22	broonie@opensource.wolfsonmicro.com	0	0	3
23	grant.likely@secretlab.ca	0	0	15
24	paul.gortmaker@windriver.com	0	0	2

Далее

Результат работы программы

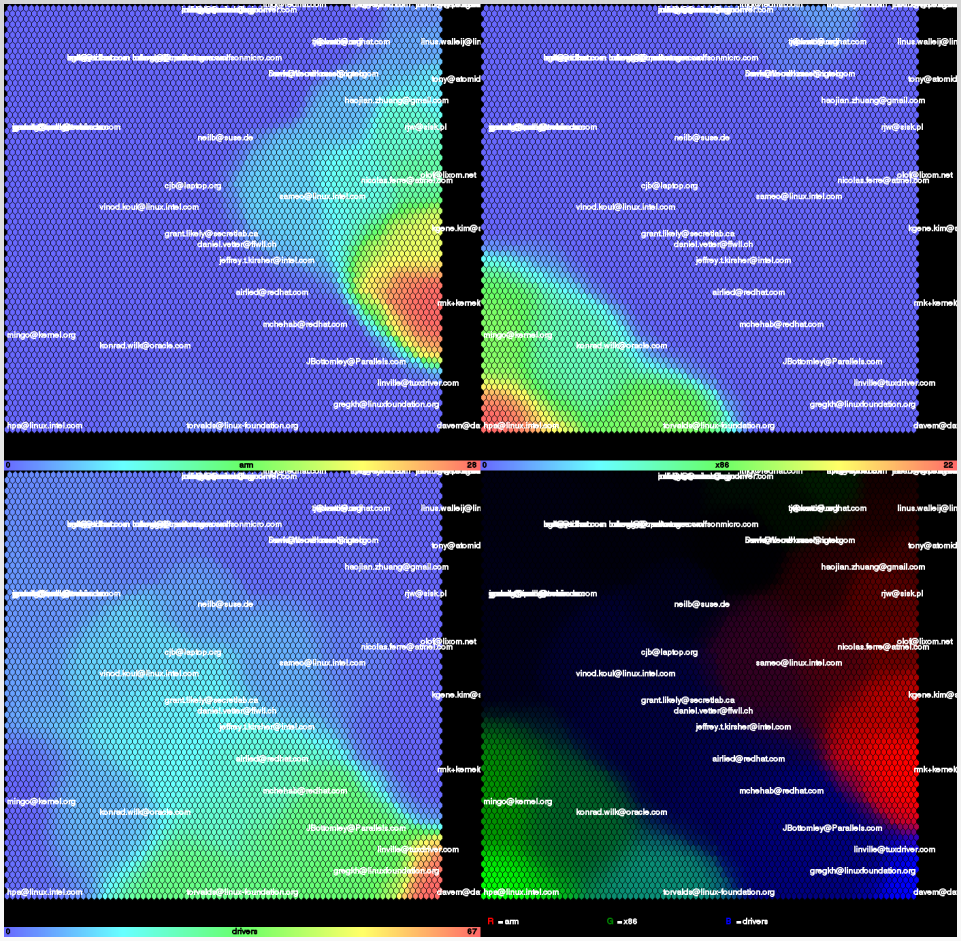

Рис. 13. Компонентные плоскости и отображение карты Кохонена в RGB-пространстве для последних изменений ядра

Построение карты при помощи программы Deductor (сравнение)

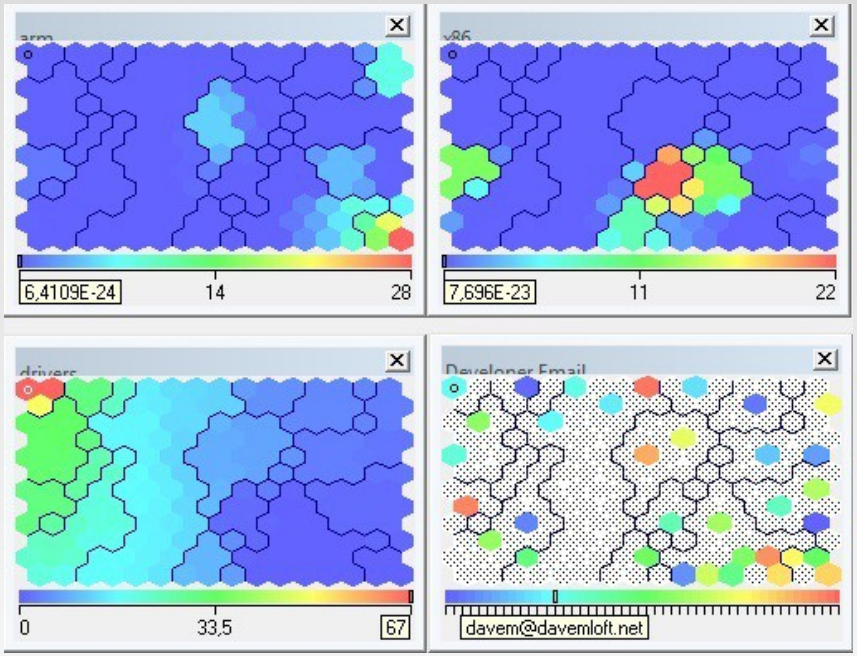

Рис. 14. Изображение, построенное при помощи аналитического приложения Deductor Studio

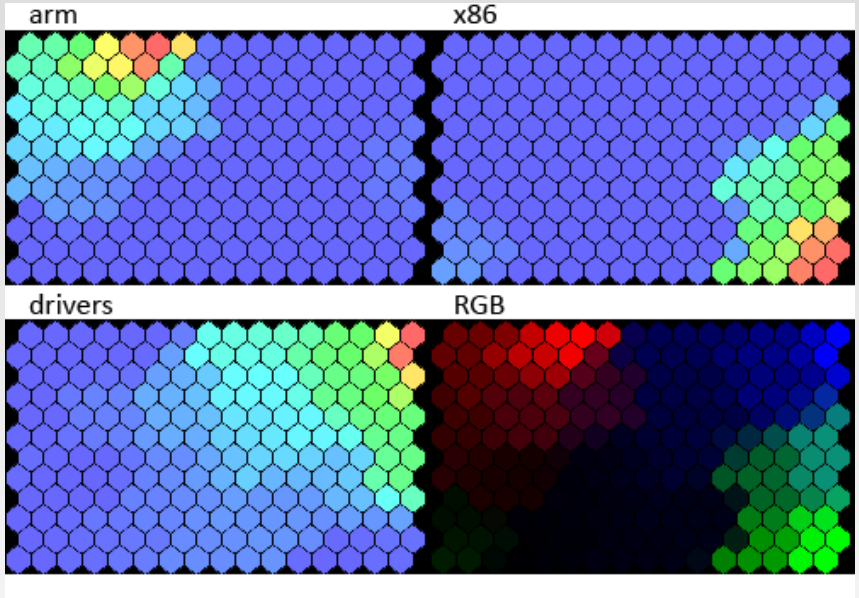

Рис. 15. Изображение, построенное при помощи программы проекта

Описание работы программы на примере

Таблица 2. Пять записей из таблицы исходных данных

№	Developer Email	Parameter		
		arm	x86	Drivers
1	benh@kernel.crashing.org	0	0	1
2	linus.walleij@linaro.org	3	0	0
3	nicolas.ferre@atmel.com	8	0	3
4	JBottomley@Parallels.com	0	0	15
5	avi@redhat.com	0	2	0

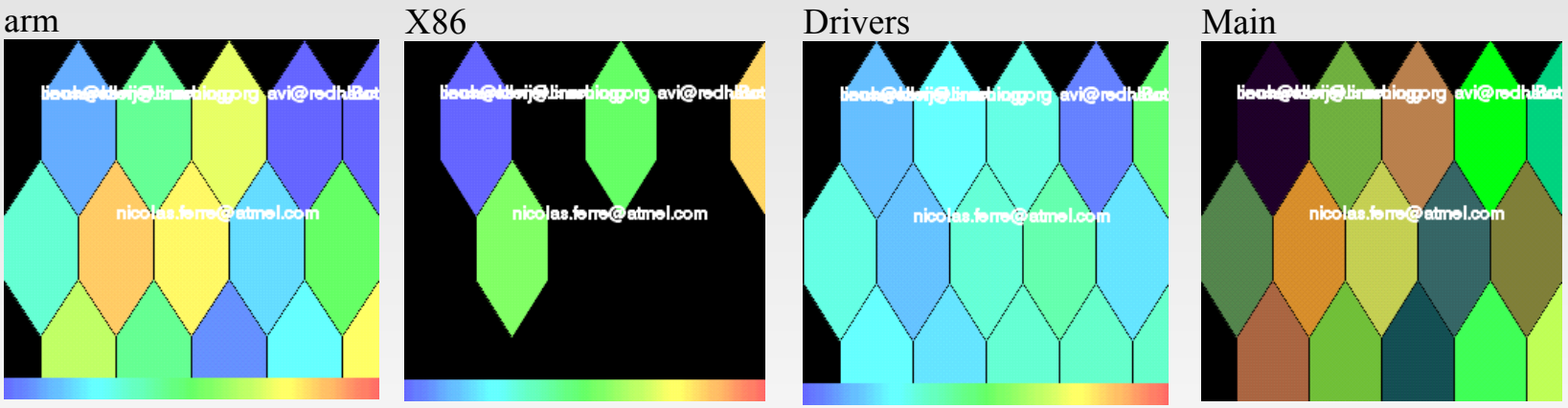

Рисунок 2. Карты Кохонена (количество узлов 5x3)

Раскраска карты Кохонена

Формулы, по которым считаем:

int r = (int)(255*(m_som_nodes->at(i)->GetWeight(0)-m_min_values[0])/(m_max_values[0]-m_min_values[0]));

int g = (int)(255*(m_som_nodes->at(i)->GetWeight(1)-m_min_values[1])/(m_max_values[1]-m_min_values[1]));

int b = (int)(255*(m_som_nodes->at(i)->GetWeight(2)-m_min_values[2])/(m_max_values[2]-m_min_values[2]));

Используем значения весов, которые выдаёт программа, минимальные и максимальные значения берём из таблицы.

m_min_values[0]=0;

m_max_values[0]=8;

m_min_values[1]=0;

m_max_values[1]=2;

m_min_values[2]=0;

m_max_values[2]=15.

Расчёты:

1. r = (255*0)/8=0;
   g = (255*0)/2=0;
   b = (255*3)/15=51.

Получившийся цвет 

Раскраска карты Кохонена

2. r = (255*2)/8=40;
   g = (255*6)/2=2FD;
   b = (255*5)/15=55.
   Цвет 

3. r = (255*5)/8=9F;
   g = (255*3)/2=17F;
   b = (255*4)/15=44.
   Цвет 

4. r = (255*3)/8=60;
   g = (255*4)/2=1FE;
   b = (255*4)/15=44.
   Цвет 

5. r = (255*6)/8=BF;
   g = (255*2)/2=FF;
   b = (255*2)/15=22.
   Цвет

6. r = (255*3)/8=60;
   g = (255*6)/2=2FD;
   b = (255*3)/15=33.
   Цвет 

7. r = (255*6)/8=BF;
   g = (255*1)/2=80;
   b = (255*5)/15=55.
   Цвет 

8. r = (255*6)/8=BF;
   g = (255*4)/2=1FE;
   b = (255*5)/15=55.
   Цвет 

9. r = (255*0)/8=0;
   g = (255*5)/2=27E;
   b = (255*5)/15=55.
   Цвет 

10. r = (255*0)/8=0;
    g = (255*5)/2=27E;
    B = (255*0)/15=0.
    Цвет 

11. r = (255*1)/8=20;
    g = (255*3)/2=182;
    b = (255*6)/15=66.
    Цвет 

12. r = (255*2)/8=40;
    g = (255*4)/2=1FE;
    b = (255*5)/15=55.
    Цвет 

13. r = (255*0)/8=0;
    g = (255*2)/2=FF;
    b = (255*6)/15=66.
    Цвет 

14. r = (255*4)/8=80;
    g = (255*5)/2=27E;
    b = (255*3)/15=33.
    Цвет 

15. r = (255*6)/8=BF;
    g = (255*4)/2=1FE;
    b = (255*5)/15=55.
    Цвет 

Анализ результата

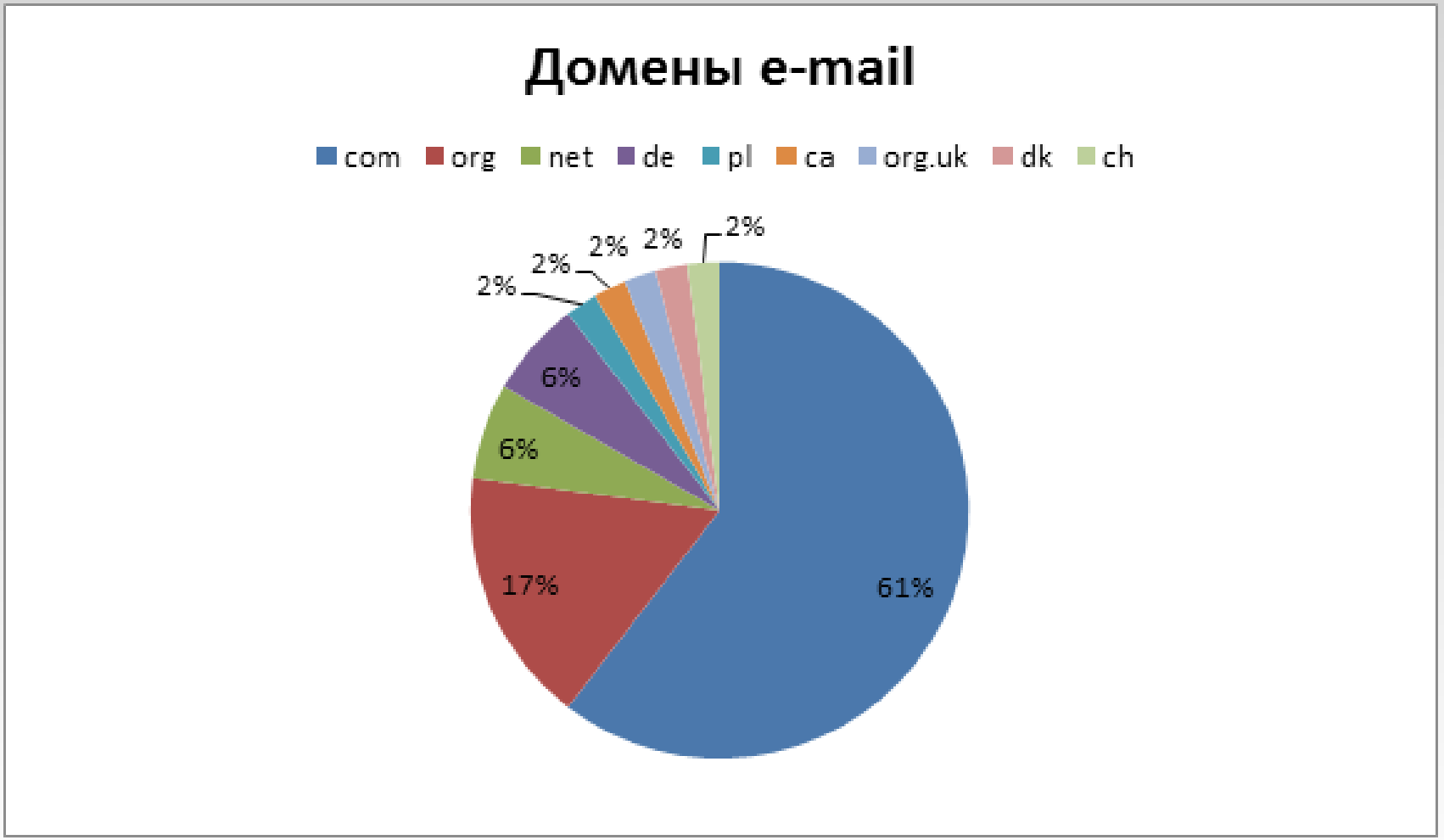

Рис. 15. Локализация разработчиков по доменам

Анализ результата

Таблица 2.

Категория домена	Количество
Корпорации	26
Проекты открытого ПО	8
Другие	7
Персональные	3
Поставщики доменов и хостов	2
Поставщики интернет-услуг	1
Поставщик услуг электронной почты	1

Анализ результата

Таблица 3. Влияние на ядро компаний

Компания	Количество упоминаний	arm	x86	drivers
Red Hat	9	0	4	60
Intel	5	5	22	42
Pengutronix	2	2	0	5
The Linux Foundation	2	1	12	62
Samsung	2	20	0	2
Parallels	1	0	0	30
Atmel	1	8	0	3
Linaro	1	3	0	0
Wolfson Microelectronics	1	0	0	3
Secret Lab	1	0	0	15
Wind River Systems	1	0	0	2
Ericsson	1	0	0	5
Pure Storage	1	0	0	3
Texas Instruments	1	0	0	4
Suse	1	0	0	7
Nokia	1	0	0	5
Oracle	1	0	8	9

Дополнительная информация о проекте

Текущая версия программы:
https://github.com/PaGrom/kohonen.git

Источники

1. Осовский. С. Нейронные сети для обработки информации / *С. Осовский*. – М.: Финансы и статистика, 2002. – 244 с.

2. Хайкин. С. Нейронные сети: полный курс / *С. Хайкин*. – М.: Вильямс, 2006. – 1104 с.

3. Кохонен. Т. Самоорганизующиеся карты / *Т. Кохонен*. – М.: БИНОМ. Лаборатория знаний, 2008. – 655 с.

4. Паклин Н. Б., Орешков В. И. Бизнес-аналитика: от данных к знаниям. – СПб.: Питер, 2009. – 624 с.

5. И.А. Чубукова «Data Mining»

6. http://www.mql5.com/ru/articles/283

7. http://ami.nstu.ru/~vms/lecture/data_mining/rules.htm

8. http://www.basegroup.ru/glossary/definitions/linear_regression/

9. http://www.statsoft.ru/HOME/TEXTBOOK/modules/stneunet.html

10. http://www.frigat.ru/131/

11. http://www.iteam.ru/publications/it/section_92/article_1448/