Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Уральский федеральный университет

имени первого Президента России Б. Н. Ельцина»

Институт математики и компьютерных наук

Кафедра алгебры и дискретной математики

Динамический анализ потенциально опасных файлов сложных форматов на предмет их относимости к вредоносным программам

|  |  |
| --- | --- |
| Допустить к защите:  Зав. кафедрой  доктор физико-математических наук,  профессор,  Волков М. В.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2015 г.  Нормоконтролер | Дипломная работа  студента 6 курса  Мельник Б. В.  Научный руководитель:  ученая степень,  ученое звание,  Белявко В. А. |

Екатеринбург

2015

# РЕФЕРАТ

Мельник Б. В. ДИНАМИЧЕСКИЙ АНАЛИЗ ПОТЕНЦИАЛЬНО ОПАСНЫХ ФАЙЛОВ СЛОЖНЫХ ФОРМАТОВ НА ПРЕДМЕТ ИХ ОТНОСИМОСТИ К ВРЕДОНОСНЫМ ПРОГРАММАМ, дипломная работа: стр. XX, рис. XX, табл. XX, библ. XX назв.

Ключевые слова: ВРЕДОНОСНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, PROCMON, MICROSOFT WORD, ДИНАМИЧЕСКИЙ АНАЛИЗ, МАШИННОЕ ОБУЧЕНИЕ, АЛГОРИТМ DYNAMIC TIME WARPING.

Объект исследования - файлы сложного формата “doc”, используемые для работы в программном продукте Microsoft Word.

Цель работы - исследование процесса работы программного пакета Microsoft Word, описание работы вредоносных программ, распространяющихся через файлы сложных форматов, создание методов, позволяющих определять степень вредоносности файлов формата “doc”.

В результате анализа работы была изучена работа вредоносных программ, распространяющихся через уязвимости файловых форматов. Также предложено несколько алгоритмов, использующих техники машинного обучения и позволяющих с приемлемым качеством классифицировать вредоносные объекты.

# ОГЛАВЛЕНИЕ

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

ВВЕДЕНИЕ

# УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

# ВВЕДЕНИЕ

Современный мир невозможно представить без компьютера. Компьютерные технологии проникли во множество направлений деятельности человека. Если раньше персональные компьютеры были связаны напрямую с потребностями в работе, то сейчас они есть практически у каждого человека. Также с развитием содержания меняется и форма. Такое направление как “носимая электроника” почти стирает грань между техникой и человеком. Всё быстрее компании, которые сталкиваются с проблемами, касающихся использования протокола IPv4 во время обмена сообщениями, переходят на обновлённый IPv6, позволяющий буквально присвоить интернет адрес каждой песчинке на Земле.

Но и как в большинстве дел, связанных с получением дохода или информации, в компьютерной индустрии присутствуют субъекты, пытающиеся извлечь для себя максимальную прибыль. Часто не совсем законными способами. Прибыль не обязательно выражается в материальном, денежном виде. В нашем случае часто большую ценность может иметь информация. И растущая распространённость техники, контролируемой автоматически, может пугать своими потенциальными последствиями её захвата. Часто в качестве средства захвата выступает вредоносное ПО. Существует множество типов

В первой главе мы опишем, какие файлы были включены в исследование.

Во второй главе мы рассмотрим один способов сбора информации о запущенных процессах в операционной системе “Windows”. По сформированной выборке из файлов формата .doc мы будем собирать информацию с помощью утилиты Procmon. Также будет описан формат файла, содержащего различные данные о процессе.

Третья глава посвящена введению в теорию машинного обучения, которая нам понадобится для дальнейшей работы с созданием метода классификации вредоносных программ. В начале главы мы сформулируем главную проблему классификации и дадим необходимые определения. Так как мы хотим классифицировать файлы произвольной структуры, далее, мы рассмотрим принципы, позволяющие работать с произвольно сложными объектами в качестве субъектов машинного обучения. Также будут описаны несколько алгоритмов классификации, пользуясь которыми, мы в дальнейшем будем решать задачу распознавания вредоносных объектов. В качестве таких алгоритмов мы рассмотрим метод K ближайших соседей и логистическую регрессию. В завершении главы мы опишем способы оценки качества полученных моделей классификации и сравним несколько алгоритмов между собой на примере задачи распознавания цветков ириса.

В четвёртой главе мы воспользуемся информацией, рассмотренной в ранних главах, и создадим набор из двух алгоритмов, позволяющих классифицировать объекты из собранной выборки.

# ГЛАВА 1. ФОРМИРОВАНИЕ ВЫБОРКИ ДЛЯ ИССЛЕДОВАНИЯ

Базовым действием для любого статистического анализа является создание выборки исследуемых объектов. В этой главе мы рассмотрим каким образом мы выбирали

# ГЛАВА 2. СБОР ДАННЫХ ПО ВЫБОРКЕ

# ГЛАВА 3. ПОДХОД МАШИННОГО ОБУЧЕНИЯ

В этой главе будут даны основные определения, а также сформулирована общая задача классификации.

Машинного обучение — это подраздел искусственного интелекта, находящийся на стыке статистики и компьютерных наук. Главная цель машинного обучения — создание методов, позволяющих строить математические модели и способных обучаться по прецедентам. Под обучением стоит понимать процесс аппроксимации заранее неизвестной функции по набору точек из её области определения и известных значений, полученных для каждой такой точки.

В нашем случае точки — это файлы формата .doc, а неизвестная функция отображает каждый такой файл во множество, состоящее из двух элементов {“вредоносный файл”, “безопасный файл”}.

Далее, мы более формально опишем, каким образом объекты из реальной жизни можно рассматривать как точки, принадлежащие области определения неизвестной нам функции, и опишем подходы для извлечения зависимостей, c помощью которых в следующей главе будем решать задачи распознавания вредоносных программ.

# §3.1 Обобщённая задача классификации

Пусть X — это множество объектов, а Y — это множество классов.

Существует неизвестное нам отображение F : X -> Y, ставящее каждому объекту из X в соответствие метку класса из Y. Несмотря на то, что само отображение нам неизвестно, для некоторого набора элементов {x\_1, …, x\_n} из X мы можем получить значения F в этих точках y\_i = F(x\_i), i = 1…n .

В такой постановке задачи набор пар (x\_i, y\_i) называется обучающей выборкой и, имея такую выборку, нам необходимо построить функцию F\*, которая наилучшим образом приближает функцию F. Конечно, мы хотим чтобы функция F\* не только возвращала метку y = F\*(x) для точки x из обучающей выборки, но и могла возвращать метку для новых, ранее неизвестных нам объектов.

Поиск F\* можно свести к минимизации некоторого функционала M(F\*, F), отражающего близость F\* к F. В качестве M можно взять, например, разность квадратов. Важно понимать, что уменьшение данной метрики на обучающей выборке не всегда приводит к уменьшению количества ошибок, совершаемых на новых объектах. Вопросы измерения качества классификации будут рассмотрены в следующих параграфах.

В некоторых случаях при использовании методов классификации нам важно знать не только конечную метку класса, но и степень уверенности в ней. В таком случае задачу классификации можно поставить следующим образом: существует неизвестное нам отображение F : X -> Y, ставящее каждому объекту из X в соответствие метку класса из Y, необходимо по обучающей выборке (x\_i, y\_i) получить функцию F\* : X -> [0…1]^|Y|, которая переводит объект в вектор вероятностей принадлежности к каждому из классов, и для любого x из X сумма F\*(x) от 1 до |Y| должна быть равна 1.

Получение значений приближенной функции F\* предполагается производить на компьютере, поэтому алгоритм, который стоит за вычислением F\*, должен быть достаточно эффективным.

Почти всегда функцию F\* мы можем выбирать из некоторого параметризированного семейства функций, тогда задача поиска наилучшего приближения функции F сводится к минимизации функционала M(F\*, F) по вектору параметров, задающих F\*. Одним из подходов поиска параметров является градиентный спуск.

# §3.2 Признаки объектов, матрица объект-признак

Зачастую рассматриваемые нами объекты имеют довольно сложную структуру. Из-за этого могут появляться проблемы в формализации функций F\* и F, введённых в предыдущем параграфе. Один из способов борьбы с такой проблемой — это перевод объекта в вектор формальных признаков.

Итак, пусть x — элемент множества X. Введём набор функций D\_i отображающие x в признаки d\_i. Признаки могут быть различных типов, например:

* бинарный, d\_i принадлежит множеству {0, 1}
* категориальный, d\_i принадлежит конечному множеству несравнимых между собой элементов
* количественный, d\_i принадлежит R

Имея {D\_1,…,D\_m}, мы можем построить для каждого из объектов {x\_1,…,x\_n} их признаковое описание. Таким образом, вместо множества объектов произвольной структуры мы будем оперировать матрицей D размера n на m.

Один из примеров такого подхода к решению задачи — это проблема классификации экземпляров цветка ириса. В этой проблеме для каждого объекта выделяются 4 количественных признака:

* длина чашелистика
* ширина чашелистика
* длина лепестка
* ширина лепестка

В качестве классов представлены три вида ирисов:

* Iris setosa
* Iris versicolor
* Iris virginica

| Длина чашелистика | Ширина чашелистика | Длина лепестка | Ширина лепестка | Вид ириса |
| --- | --- | --- | --- | --- |
| 5.1 | 3 | 1.4 | 0.2 | setosa |
| 7 | 3.2 | 4.7 | 1.4 | versicolor |
| 6.3 | 3.3 | 6 | 2.5 | virginica |

Табл 1. Фрагмент матрицы объект-признак для задачи ирисы Фишера.

В дальнейшем, при решении задачи классификации вредоносных объектов мы также перейдём от файлов к их признаковым описаниям и впоследствии к матрице объект-признак.

Имея на руках такую матрицу, мы можем применять большинство методов машинного обучения.

# §3.3 Линейные модели классификации

Одним из самых простых и распространенных методов построения классификаторов является логистическая регрессия. Данный метод позволяет создавать модели предназначенные только для бинарной классификации. Без ограничения общности будем считать, что множество классов равно {0, 1}.

Для того, чтобы построить классификатор, нам нужна только матрица объект-признак, введенная ранее. Мы подаём её на вход, а на выход получаем линейную модель, задаваемую набором чисел {a\_0, a\_1,…, a\_i}, где i это количество признаков у объектов.

После этапа обучения мы можем предсказывать класс новых объектов следующим образом. Пусть x — это новый объект, используя ранее введёные функции D\_i, создаём признаковое описание {d\_1,…,d\_i} элемента x. На этом шаге важно использовать те же самые функции D\_i, с помощью которых была создана матрица объект-признак на этапе обучения.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Метка класса | |
| 0 | 1 |
| Ответ классификатора | 0 | TN | FN |
| 1 | FP | TP |

Далее, мы вычисляем следующее выражение f(сумма от 1 до i ( d\_i \* a\_i ) + a\_0), где f(z) = 1 / (1 + e^(-z)). Значение этого выражения определяет вероятность принадлежности элемента к первому классу.

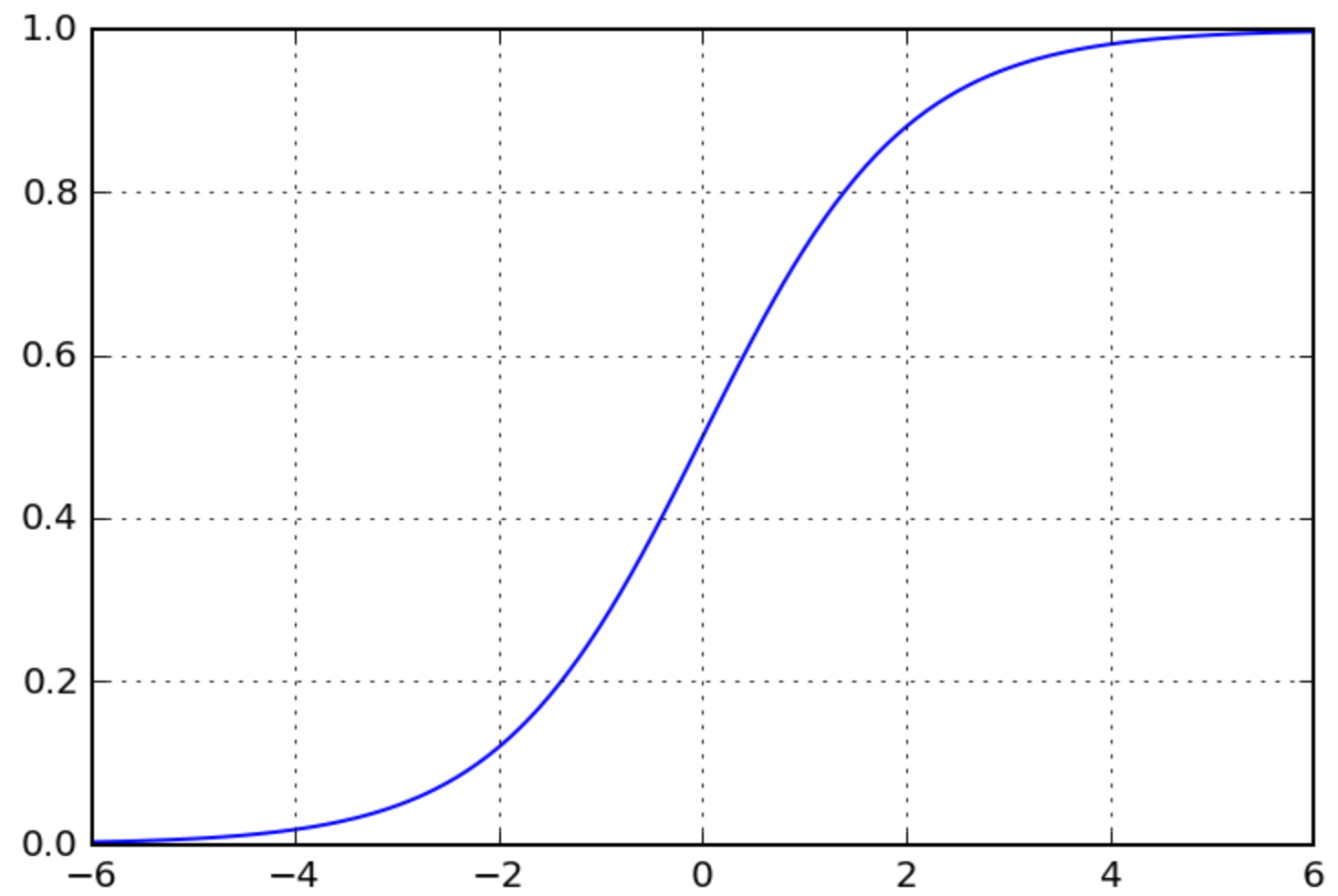
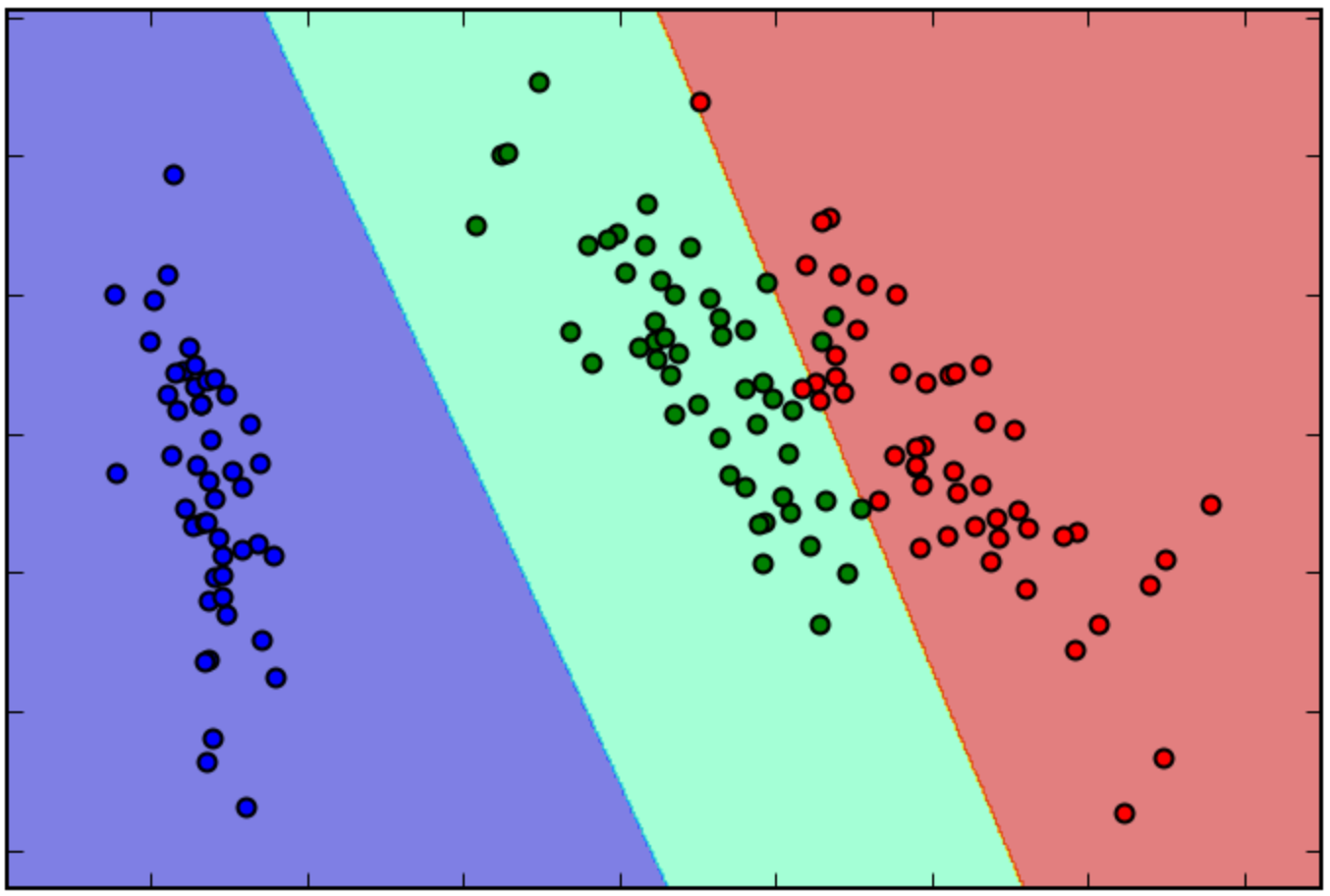


График функции f(z) = 1 / (1 + e^(-z))

Для поиска коэффициентов {a\_0,…,a\_i}, способствующих наилучшему приближению на обучающей выборке, существует множество эффективных методов. В дальнейшем мы будем использовать этот алгоритм для финального объединения результатов, полученных от большого набора других классификаторов.

Данную модель можно интерпретировать следующим образом: модуль коэффициента при признаке под номером i передаёт вес данного признака для итоговой оценки: чем он больше, тем важнее признак. Знак коэффициента отвечает за то, к какому классу мы будем отдавать предпочтение, если коэффициент отрицательный, то к классу “0”, если положительный, то, соответственно, классу “1”. Вычислив произведение вектора коэффициентов на вектор признаков, мы получаем число, которое необходимо сравнить с границей принятия решения -a\_0. Если мы перешли эту границу, то метод будет отдавать предпочтение классу “1”, и, чем дальше данное число будет уходить за эту границу, тем больше вероятность принадлежности объекта к классу “1”.

Конечно, логистическую регрессию можно использовать не только как способ объеденения других классификаторов. Ниже я привожу пример решения задачи классификации цветков ириса. Для возможности визуалиации я предварительно понизил размерность вектора признаков каждого объекта с четырёх признаков до двух. Для понижения размерности я воспользовался методом главных компонент.



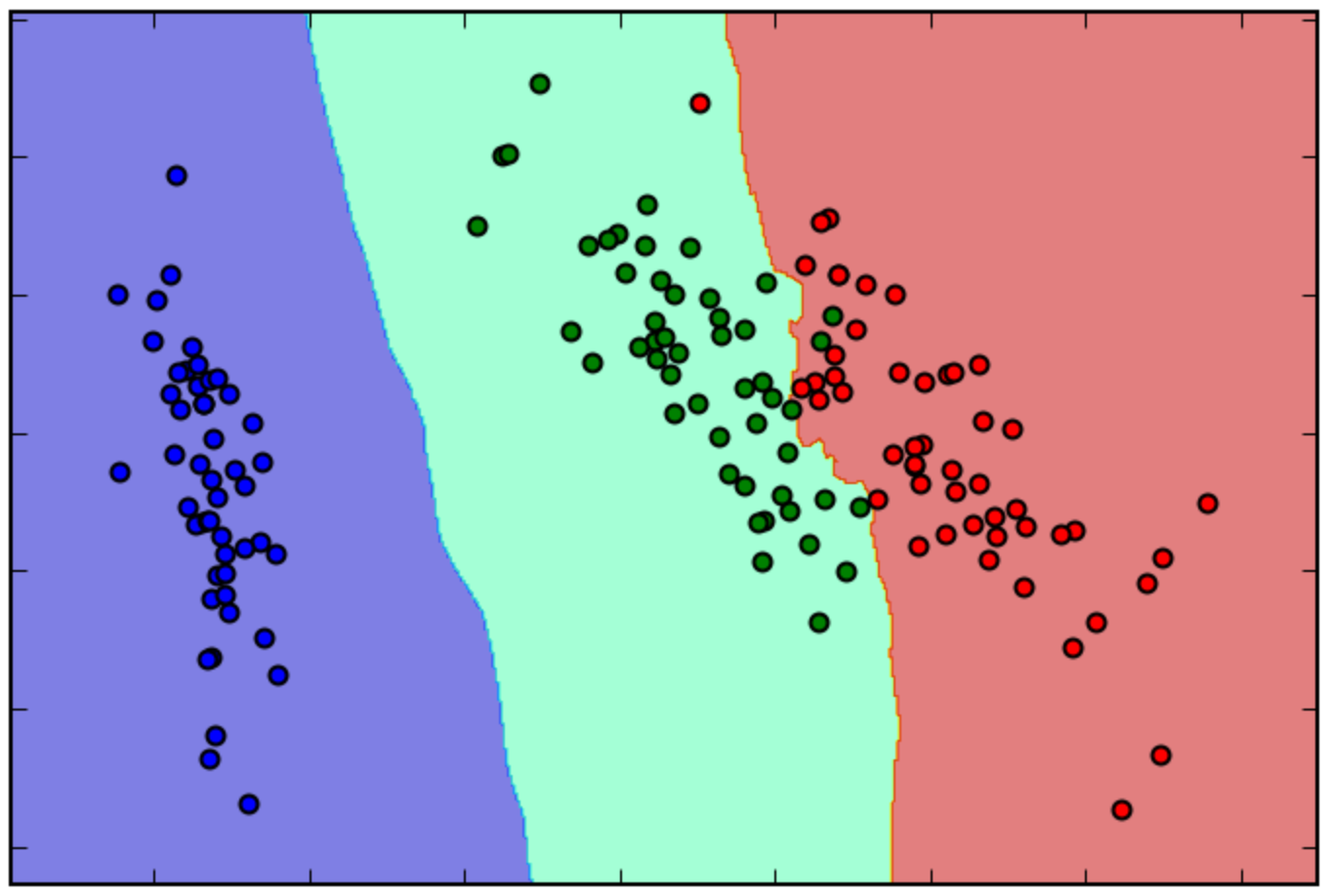
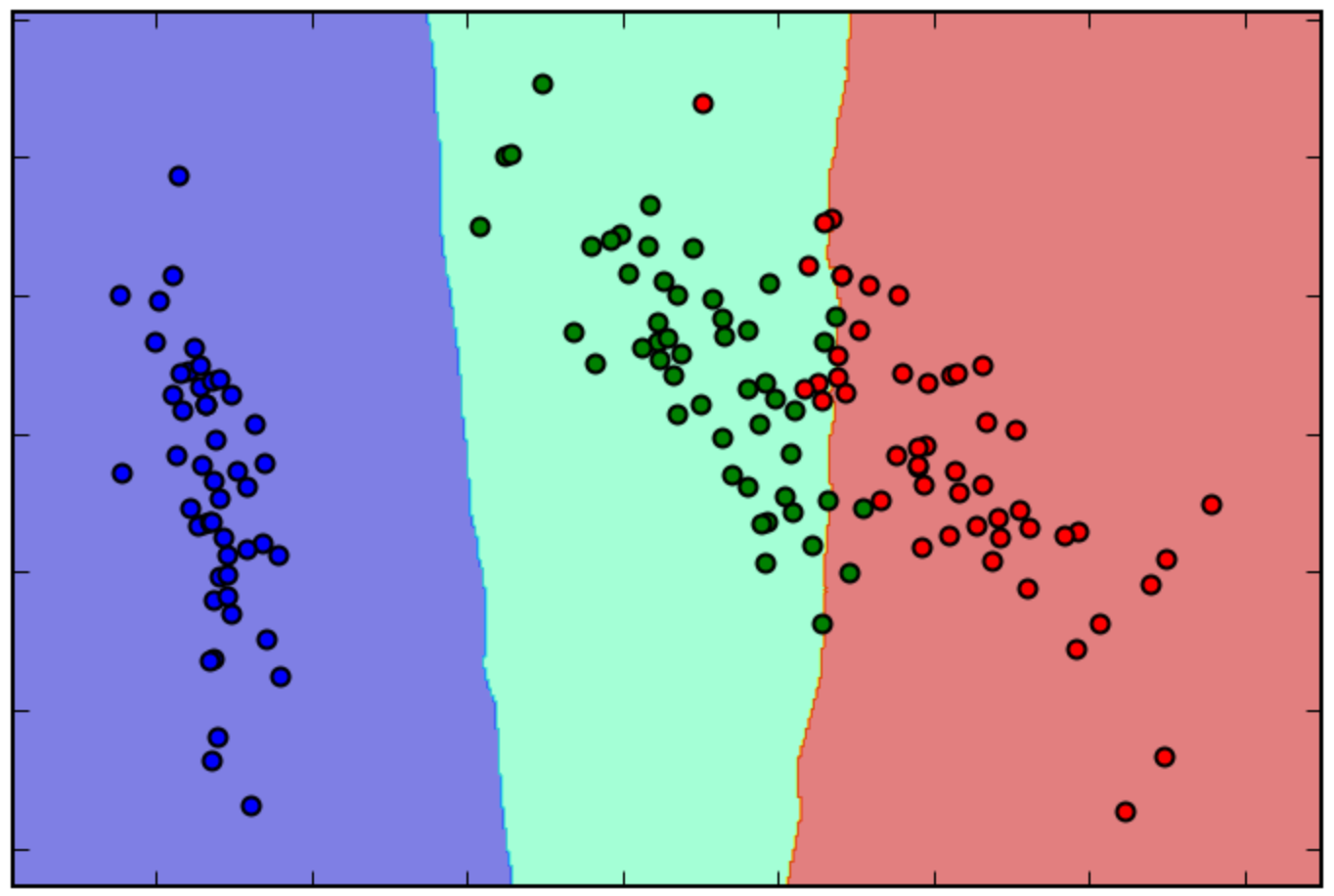
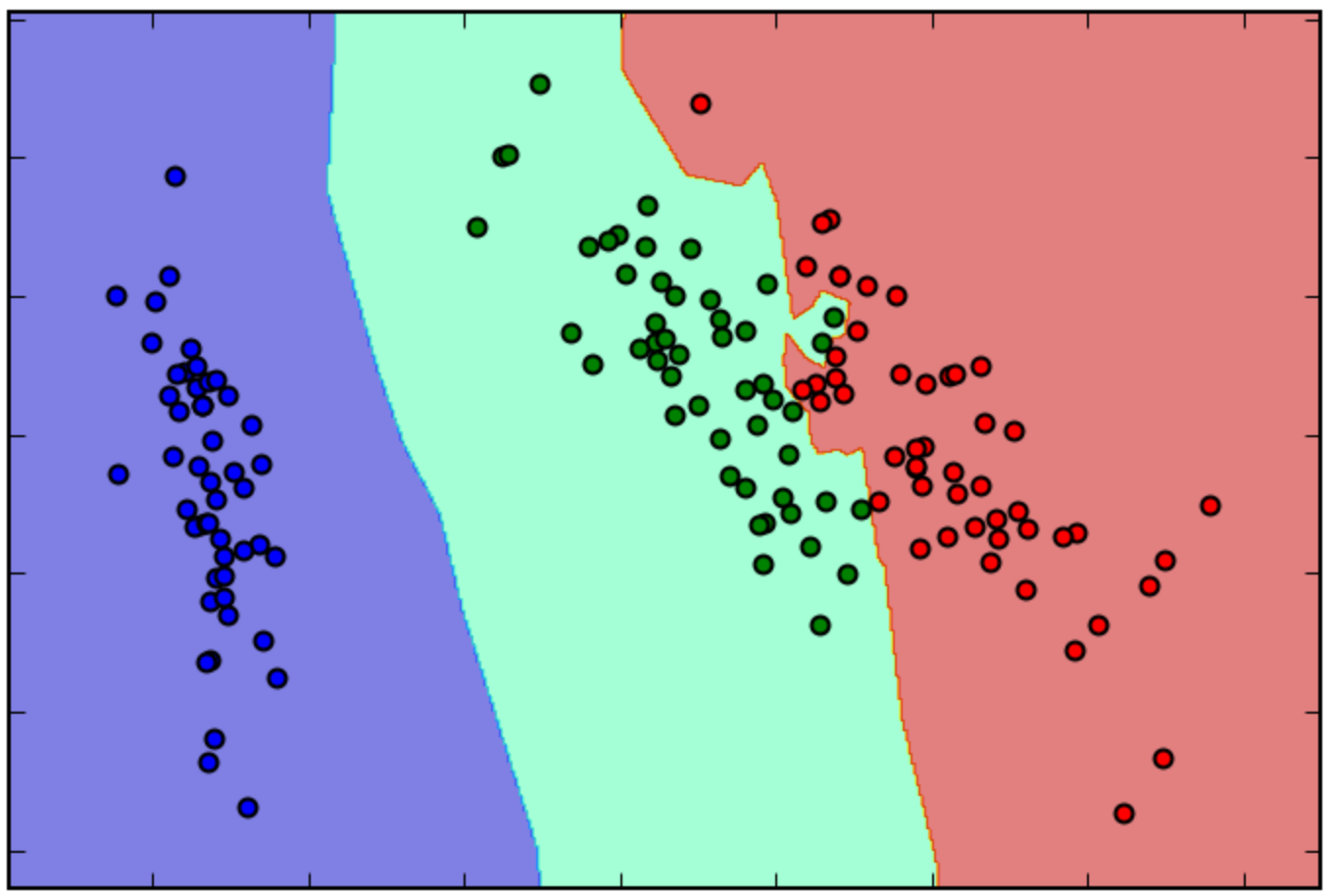
Ирисы Фишера. Границы классов, полученные методом логистической регрессии.

# §3.4 Метрические алгоритмы классификации

Существует другой подход к задаче классификации, базирующийся на следующем предположении: объекты из одного класса должны располагаться недалеко друг от друга (относительно некоторой функции близости или расстояния //). Такой метод полезен в случаях, когда какой-то объект не описывается однозначно с помощью конечного числа признаков. Исходя из данного предположения, легко построить эвристический метод классификации, который называется метод K ближайших соседей.

Всё, что требуется от нас, это функция расстояния F : X x X -> R, определённая для каждой пары объектов. Если рассмотреть одну из простых реализаций данного метода, то алгоритм классифицирующий новые объекты сводится к следующей последовательности действий: во-первых, мы сортируем элементы обучающей выборки в порядке возрастания расстояния до нового объекта. Далее, мы выбираем K первых элементов отсортированной последовательности и подсчитываем, какой класс сколько раз встретился. В итоге, новому объекту мы назначаем метку того класса, который встретился больше всего.

Кроме описаного варианта реализации также существуют более сложные, например, один из способов вводит такое понятие как вес объекта и позволяет оперировать им в процессе обучения, уменьшая вес элемента с увеличением его номера позиции в отсортированной выборке.



Ирисы Фишера. Результат работы алгоритма K ближайших соседей

для K равного 1, 5 и 50 соседей соответственно.

Существует много готовых функций расстояний, например, если наш объект в обучающей выборке принадлежит пространству R^n, то в качестве функции расстояния можно взять Евклидову метрику. Далее, когда нам будет необходимо сравнивать информацию о двух запущенных процессах, мы введём более сложную метрику.

# §3.5 Оценка качества классификации

После того, как мы подобрали выборку, состоящую из рассматриваемых объектов, и построили по ним модель для классификации, наступает этап внедрения созданного метода в рабочий процесс. Во время работы для нас важно, как часто модель ошибается на новых объектах. Если ошибки будут возникать слишком часто, то польза от такого алгоритма будет поставлена под сомнение.

Часто при оценке полученных моделей в машинном обучении прибегают к следующему способу: мы делим обучающую выборку на две части. Одну из частей мы по-прежнему будем использовать для обучения метода классификации, а вторую часть назовём тестовой выборкой и будем на ней проверять качество. Для каждого объекта из тестовой выборки мы с помощью нашего метода предсказываем метку класса и, далее, сравниваем полученный ответ с истинным. Истинный ответ нам известен, так как тестовая выборка является подмножеством изначальной обучающей выборки.

Имея на руках две метки класса, одну из которых мы получили от применения классификатора, мы можем применять огромное множество способов для оценки качества классификации. Один из способов, которым мы будем пользоваться в дальнейшем, основан на анализе матрицы ответов.

Матрица возможных ответов классификатора.

В данной матрице мы обозначили число в каждой ячейке заглавными латинскими буквами, рассмотрим что они означают:

TN — количество объектов правильно классифицируемых как элементы класса “0”

FP — количество объектов, которое метод ошибочно отнёс к классу “0” (данный тип ошибок называется ошибками первого рода),

FN — количество объектов, которое метод ошибочно отнёс к классу “1” (данный тип ошибок называется ошибками второго рода),

TP — количество объектов, которые были правильно классифицированы и являются элементами класса “1”.

Через величины TN, FP, FN, TP вводятся промежуточные оценки классификации: полнота и точность.

R = TP / (TP + FN)

Формула для вычисления полноты.

P = TP / (TP + FP)

Формула для вычисления точности.

Значения полноты и точности принадлежат отрезку [0..1]. Значение полноты можно интерпретировать как вероятность того, что случайно выбранный элемент во время классификации будет отнесён к классу “1”, при условии, что он действительно принадлежит классу “1”. А значение точности — как вероятность того, что случайно выбранный элемент, отнесённый классификатором к классу “1”, действительно принадлежит классу “1”.

В случае задачи классификации вредоносных программ величина полноты предсказывает то, как часто мы будем пропускать вредоносные программы, а величина точности — насколько часто мы будем принимать хорошие программы за плохие.

В некоторых случая удобнее иметь одну метрику для сравнения классификаторов вместо двух. Для этого существует показатель, называющийся F-мерой, совмещающий в себе одновременно точность и полноту.

F = 2 \* P \* R / (P + R)

Формула F- меры.

В дальнейшем мы будем сравнивать качество классификаторов именно по F-мере.

В момент отделения тестовой выборки возникает несколько проблем. Во-первых, мы можем поделить выборки таким образом, что одна из них получится смещённой по отношению к другой. В таком случае наша оценка классификатора окажется неверной. Во-вторых, мы теряем для использования в процессе обучения элементы, попавшие в тестовую выборку, и тем самым упускаем потенциальную возможность улучшить наш классификатор новыми данными. Для борьбы с этим можно использовать подход, имеющий название KFold. Мы делим обучающую выборку на N частей и последовательно используем в качестве тестовой выборки каждую из частей. После этого у нас будет N оценок классификатора, которые можно усреднить для получения финальной оценки.

Также стоит упомянуть отдельный случай, когда N принимает значение равное длине выборки. То есть мы будем обучаться на всей выборке кроме одного объекта, а потом пытаться его предсказать. Данный метод имеет название Leave One Out или сокращённо — LOO. С помощью этого метода мы не сможем правильно посчитать точность, полноту и F-меру, но зато сможем понять по количеству ошибок насколько далёк наш классификатор от идеального. Этот метод проверки качества особо актуален в случае недостатка рассматриваемых объектов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод классификации | Точность | Полнота | F-мера | Количество ошибок при LOO |
| Логистическая регрессия | 0.96 | 1.0 | 0.97 | 4 |
| 1 ближайший сосед | 0.96 | 0.94 | 0.93 | 6 |
| 5 ближайших соседа | 0.96 | 0.96 | 0.94 | 5 |
| 50 ближайших соседей | 0.90 | 0.88 | 0.87 | 9 |

Табл 1. Сравнение метрик качеств для задачи ирисы Фишеры.

Взяты только два пересекающихся класса цветков.

# ГЛАВА 4. РЕШЕНИЕ ЗАДАЧИ КЛАССИФИКАЦИИ

# ПЛОХОЙ-ХОРОШИЙ

В данной главе будут предложены несколько алгоритмов классификации вредоносных файлов, позволяющих с приемлемым качеством определять метку объектов, подготовленных нами ранее.

Действия в этой главе являются заключительными для создания классификатора. Мы будем использовать результат работы предыдущих глав, где мы создали необходимую выборку файлов, включающую в себя как вредоносные файлы, так и файлы, не причиняющие никакого вреда, а также собрали информацию о работе процесса “Microsoft Word”, специальным образом подготавливая каждый файл из выборки для запуска.

На практике очень часто методы машинного обучения тесно связаны с рассматриваемыми объектами и признаками, которые могут быть извлечены из них. Поэтому каждый предложенный нами алгоритм окажется отдельным подходом извлечения формального набора признаков для каждого исследуемого объекта. В зависимости от типа получившихся факторов мы будем использовать подходящий метод машинного обучения.

# §4.1 Процесс как последовательность действий

Для построения первой модели, мы рассмотрим процесс исполнения программы Microsoft Word, во время открытия каждого файла, как последовательность простых операций. В качестве операции будут выступать вызовы функций WinAPI, регистрируемые утилитой PROCMON, например такие как открытие файла, запись в файл или загрузка динамической библиотеки.

Во время создания того или иного набора признаков, принято обосновывать почему именно такой подход имеет право на жизнь. В нашем случае мы будем опираться на следующее предположение: последовательность действий, совершаемых вредосными файлами отличаются от нормального исполнения обычных пользовательских файлов.

Имея на руках последовательность действий мы будем использовать технику, часто применяемую для построения различных языковых моделей натуральных языков. Только в отличии от традиционного использования таких методов, отдельным элементом языка у нас будет не слово на английском или русском языках, а операция, совершаемая запущенной программой. Каждый процесс в таком случае можно представить как документ, содержащий одно длинное предложение со словами из заданого множества.

Первый шаг необходимый для построения данной модели это перевод сырых данных, собранных с помощью утилиты PROCMON для каждого файла, в вектор слов. В нашем случае слово будет принадлежать конечному множеству операций, которые поддерживает PROCMON для перехвата, при этом мы будем рассматривать только подмножество операций непосредственно совершаемых Microsoft Word’ом и перехваченных во время исполнения.

|  |  |
| --- | --- |
| Process Start | 0 |
| Thread Create | 1 |
| QueryDeviceInformationVolume | 2 |
| Load Image | 3 |
| CreateFile | 4 |

Табл 1. Пример операций и их отображений.

Для дальнейшей работы с этими операциями нам будет удобно отобразить их в короткие численные представления. Используя всё выше сказанное мы можем рассматривать элементы выборки как большие предложения, состоящие из чисел.

|  |  |
| --- | --- |
| Process Start | 0 |
| Thread Create | 1 |
| QueryNameInformationFile | 2 |
| Load Image | 3 |
| Load Image | 3 |
| CreateFile | 4 |
| QueryDeviceInformationVolume | 5 |
| QueryOpen | 6 |
| Load Image | 3 |
| CreateFile | 4 |

Табл 1. Первые 10 операций и их отображений.

Например, если допустить что таблица 1 содержит результат работы программы Microsoft Word от начала до конца, то объект, сгенерировавший данные операции, будет иметь вид: “0 1 2 3 3 4 5 6 3 4”.

Один из случаев, которые мы сможем определить такой моделью, является вызов некоторой аномальной функции, совершающей потенциально зловредные действия. Тогда слово, соответствующее отображению этой функции в число, не будет принадлежать никакому предложению, составленному по безвредным объектам. В таком сценарии мы можем использовать распределение частот по каждому слову, состоящему из цифр, для принятия решения. Однако, в некоторых случаях для нас может представлять опасность не только отдельный вызов функций, а некоторая последовательность вызовов безобидных подпрограм. Например, последовательность функций “CreateFile”, “ReadFile”, “WriteFile” может служить сигналом о создании копии вируса в системе. Чтобы учесть такие ситуации, мы будем использовать распределение N-грамм как обобщение частот отдельных слов.

N-граммы для N равного 1 представляют из себя частоты каждого слова документа в отдельности. Для N равного 2 это распределение пар слов и т.д. Например, рассмотрим такое распределение для объекта из таблицы 1, где N принимает значения из множества {1,2}.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N-грамма | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 0 1 | 1 2 | 2 3 | 3 3 | 3 4 | 4 5 | 5 6 | 6 3 |
| Частота | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |

Табл 1. Разложение объекта X на N-граммы для N из {1, 2}

Количество различных N-грамм, конечно, зависит от самого значения числа N, поэтому его нужно выбирать из соображений баланса между длиной потенциально вредной последовательности и размером конечной матрицы объект-признак. В нашем случае для создания рабочего метода мы будем использовать N равное 3 ( Рассмотреть набор экспериментов для N от 1 до 3 и указать что разницы большой нет. Обосновать почему именно 3). Таким способом мы переведём каждый объект в вектор частот, получив в итоге матрицу объект-признак. Как и в примере с объектом из таблицы 1, в качестве признака будет частота определённой N-граммы встретившейся в каждом объекте из обучающей выборки.

Это и дальнейшие преобразования будут производиться с помощью библиотеки scikit-learn для языка Python. ( Подробней про эти библиотеки и про язык ).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G0 | 38 | 5 | 91 | 17 | 10 | 2 | 4 | 1 | 1 | 1 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G1 | 49 | 5 | 16 | 17 | 10 | 2 | 5 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 4 | 0 | 0 | 0 | 0 | 0 |
| B0 | 285 | 9 | 78 | 85 | 79 | 3 | 15 | 5 | 1 | 1 | 3 | 3 | 0 | 7 | 12 | 1 | 3 | 1 | 5 | 1 |
| B1 | 288 | 8 | 81 | 85 | 79 | 2 | 8 | 1 | 1 | 1 | 3 | 3 | 0 | 4 | 12 | 1 | 0 | 0 | 2 | 0 |

Табл 1. Фрагмент матрицы объект-признак для двух хороших (G0, G1) и двух плохих (B0, B1) файлов.

Имея на руках данную матрицу, мы можем воспользоваться ранее описаной линейной моделью. После обучения метода и проверки качества классификации с помощью подсчёта ошибок техникой LOO, мы заметим что метод со 100 процентным качеством определяет вредоносные файлы. То есть составленная нами выборка линейно разделима по какому-то набору признаков. Мы можем найти примеры таких признаков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | QueryAllInformationFile  FileSystemControl | QueryAllInformationFile  FileSystemControl  QueryDirectory | QueryAllInformationFile  FileSystemControl  WriteFile | Плохой или хороший |
| G0 | 0 | 0 | 0 | 0 |
| G1 | 0 | 0 | 0 | 0 |
| B0 | 2 | 1 | 1 | 1 |
| B1 | 2 | 1 | 1 | 1 |

Табл 1. Признаки позволяющие линейно разделить выборку.

Нашу выборку можно разделить по последовательностям действий представленным в таблице 1. Для нас это означает что существует последовательность операций, совершаемых только вредоносными объектами. Не смотря на идеальную точность, данный результат считается недостаточно полным без дальнешего анализа происхождения данных операций, но подробный разбор работы Microsoft Word в случаях эксплуатаций различных уязвимостей является достаточно крупной задачей, достойной рассмотрения в отдельной работе. Мы же ограничимся получившимся результатом, как первичной грубой оценкой при первичном анализе зловредных объектов. ( Показать в исходных данных найденный паттерн. Подробней написать как модель работает. )

Следующий метод основан на более сложном подходе, требующий объединения нескольких алгоритмов машинного обучения.

# §4.2 Процесс как набор графиков

В предыдущем параграфе был рассмотрен подход, основанный на разбиении выполнения процесса на простые операции с последующим анализом сгенерированных последовательностей операций.

Ранее нами было описан общий план проникновения и исполнения зловредного кода через эксплуатацию уязвимостей в сложных форматах файлов. Одним из этапов заражения является подготовка атакуемой программы для запуска шелл-кода. Часто в этот момент программа начинает вести себя необычно: зависать, работать медленней или аварийно завершаться. Чтобы отлавливать подобные ситуации автоматически, нам необходимо ввести набор формальных признаков, способных различать такие ситуации. Однако мы не можем подстраиваться под определённый вид поведения, например, завершение программы, потому что новый вирус, способный избегать появление такого демаскирующего признака, ускользнёт из нашего исследования.

В качестве обобщённого набора признаков, не привязанных к определённому виду визуального проявления, мы будем рассматривать время исполнения каждой операции. Время как признак является довольно чувствительными показателем работы процесса, если процесс будет вести себя необычно, то это прямо скажется на численных показателях времени. Исходя из таких рассуждений, был разработан новый метод извлечения формальных признаков работы из запущенной программы, основанный на представлении работы процесса в виде двумерного графика, где по оси X мы будем отображать номер операции, а по оси Y затраченное время на выполнение этой операции.

То есть теперь в качестве признака выступает график, а не одно число, как в случае с частотой N-грамм. Это более богатое семейство, в отличии от N-грамм составленных по последовательностям из простых операций.

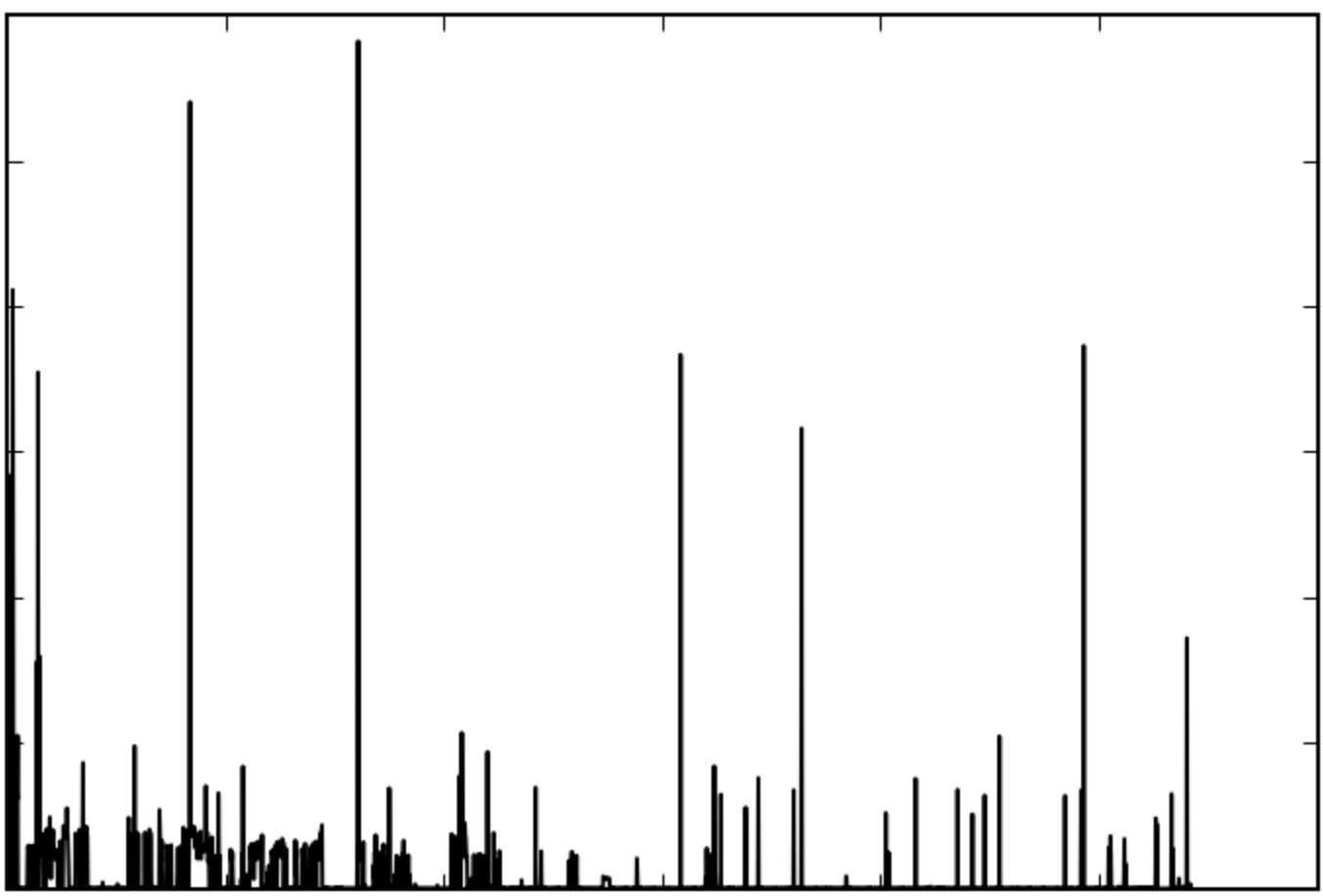


Рис. 1 График длительностей выполнения операций.

В графике, изображённом на рисунке 1, по оси X были отображены операции всех типов в порядке их перехвата утилитой PROCMON. При работе больших программ вроде Microsoft Word многие операции производятся асинхронно, это означает что порядок некоторых групп операций неопределён. Как способ борьбы с этой проблемой мы можем разделить большой график, включающий все операций, на множество графиков, отображающих каждый тип операции в отдельности. Работа с множеством графиков также позволяет использовать некоторое подмножество операции, дающих наибольший прирост к качеству классификации. Подробности реализации такого отбора, будут рассмотрены в дальнейших параграфах.

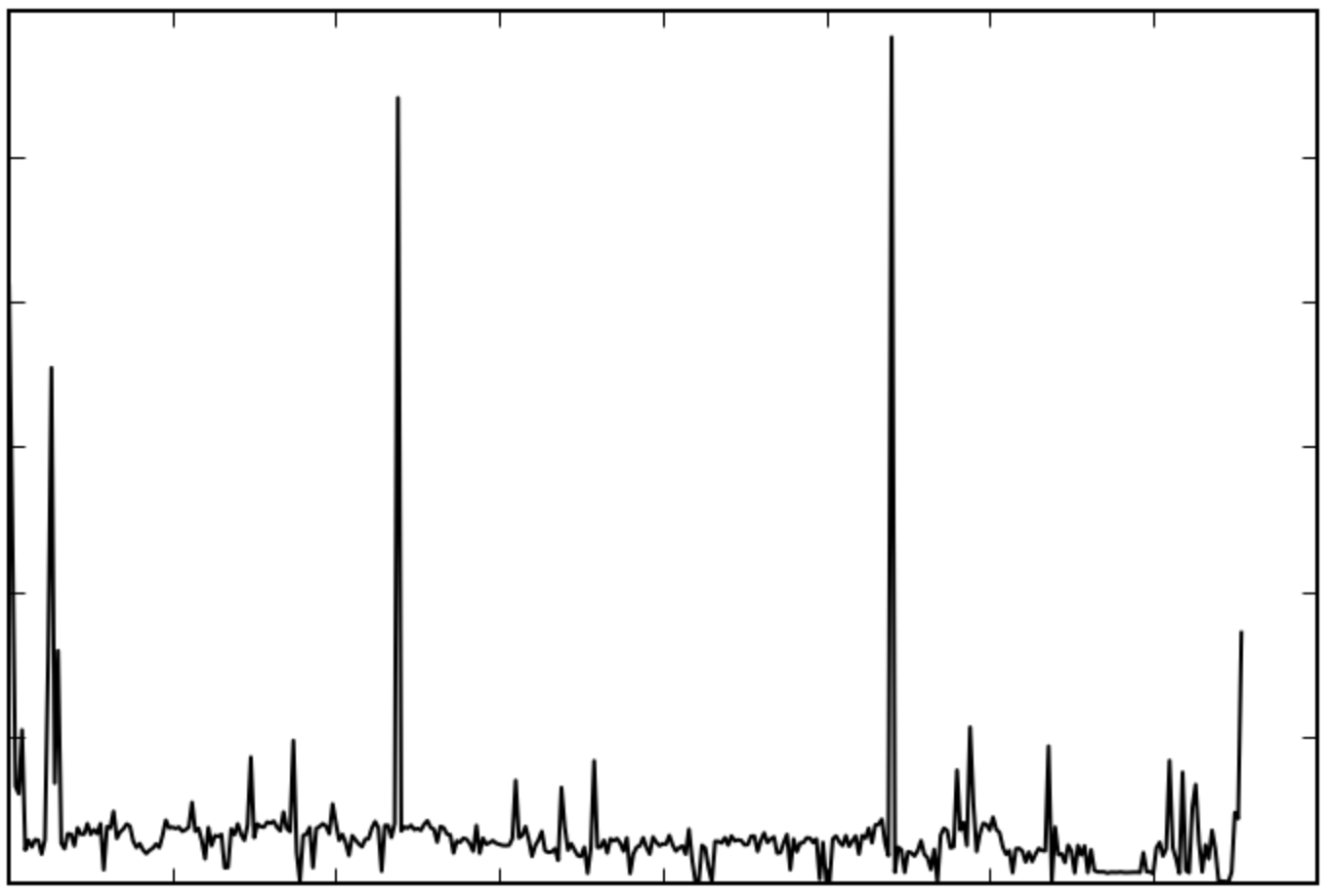
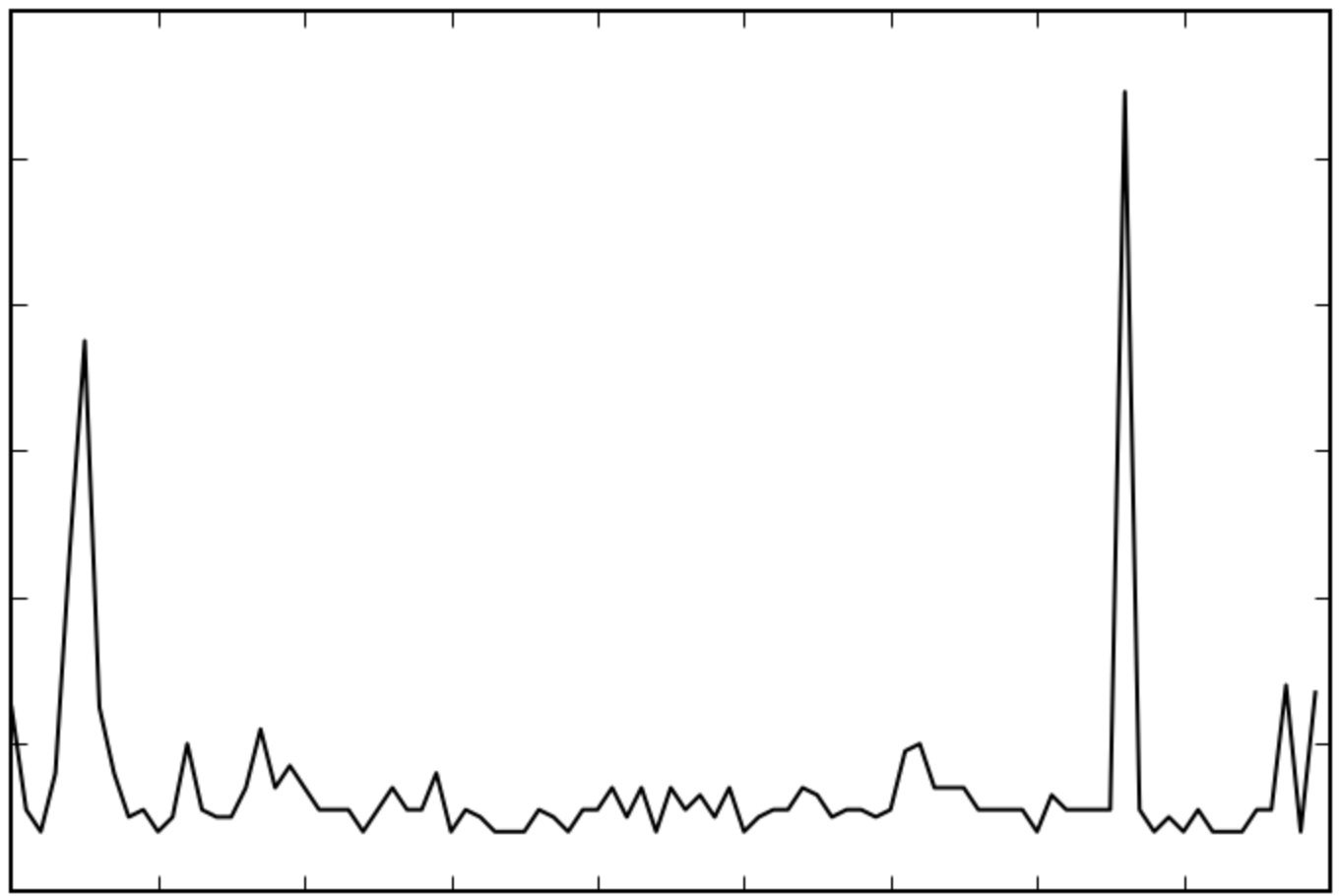
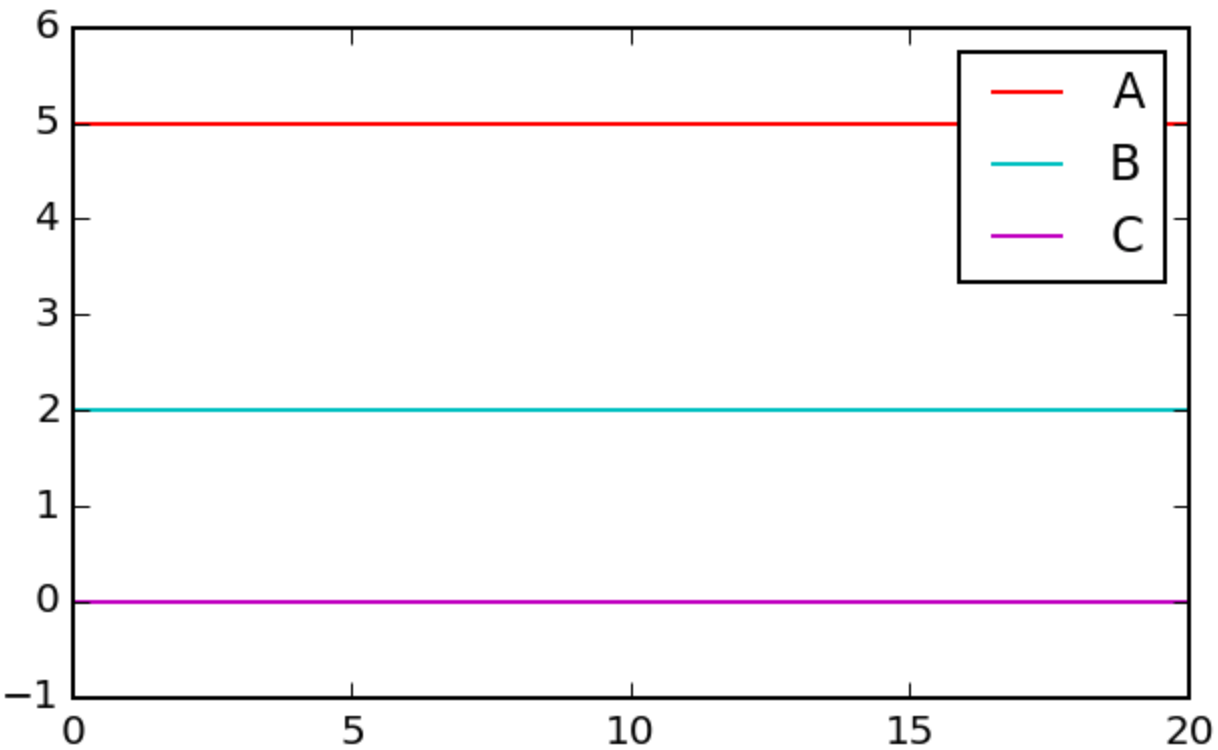


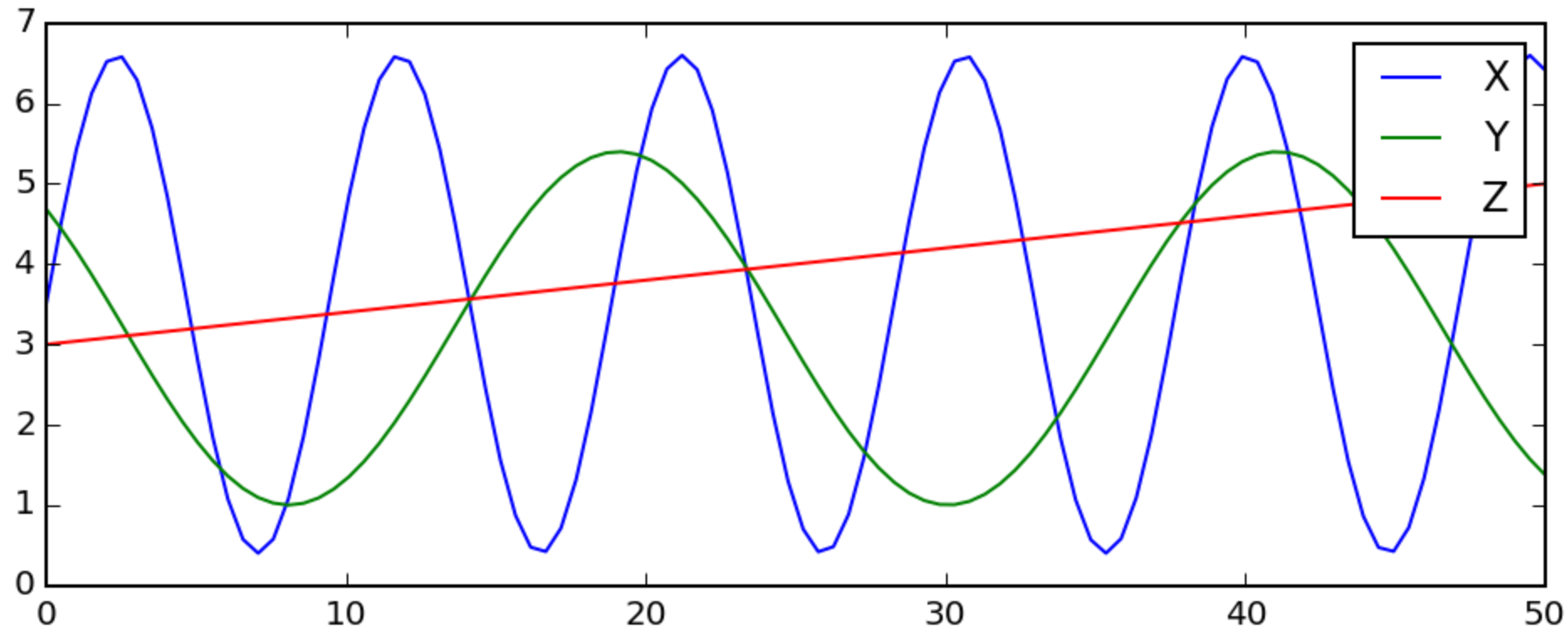
Рис. 1 Графики операций QueryBasicInformationFile и ReadFile соответственно

Итак мы разделили общий поток операций на несколько и, если в предыдущем параграфе в качестве признака объекта использовалась частота определённой N-граммы, то сейчас мы имеем набор графиков в качестве множества признаков. В отличии от действительных чисел, новые признаки в виде графиков не подходят для прямого использования в линейной модели и нам необходимо использовать другие методы машинного обучения. В данном случае, для решения задачи классификации, мы введём функцию расстояния между двумя сигналами и далее воспользуемся метрическим алгоритмом K ближайших соседей. Данную функцию мы будем использовать для каждого типа графиков в отдельности, поэтому при таком решении у нас возникнет вектор расстояний. Как мы увидим в дальнейшем, используя данный вектор, мы создадим целый набор классификаторов, после чего решим проблему их объединения.

# §4.3 Функция расстояния между сигналами. Алгоритм DTW.

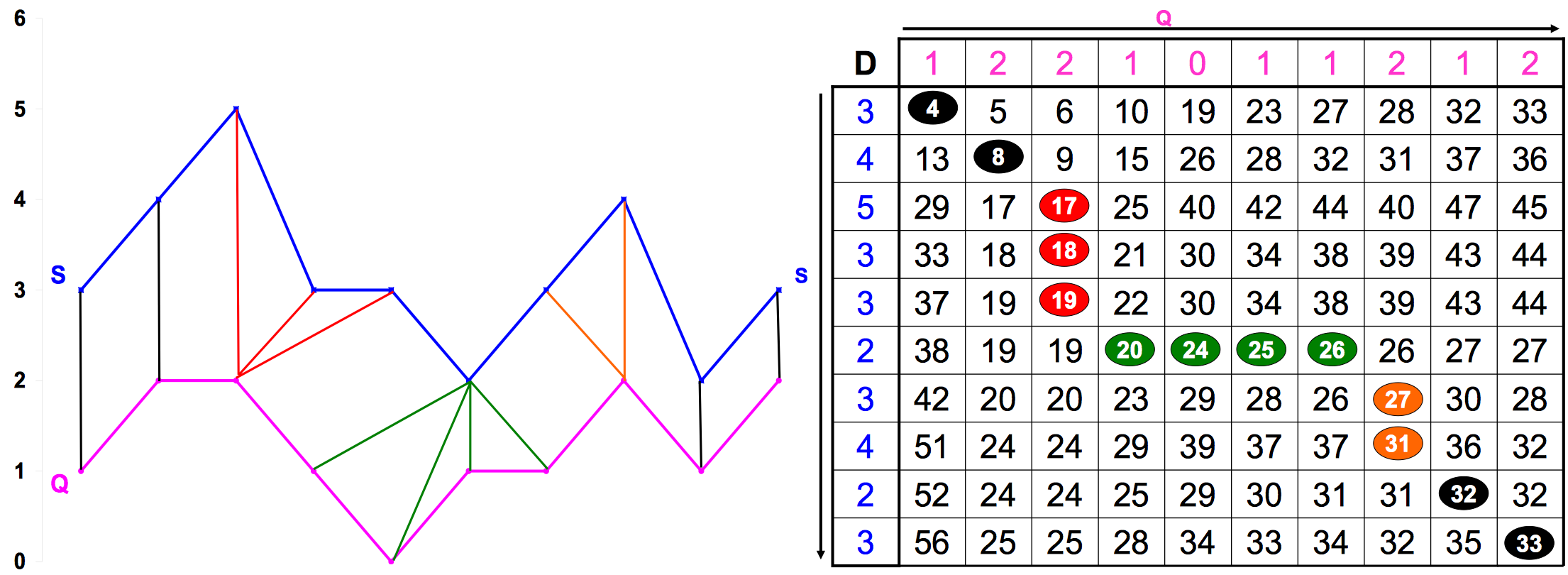
Ранее получив набор графиков в качестве признаков объекта, нам понадобился алгоритм сравнения двух функций на близость. Человек, глядя на изображение нескольких функций, почти всегда сможет сказать похожи они или нет. В качестве простого примера можно рассмотреть набор из трёх графиков на рисунке 1. Довольно легко увидеть что графики B и C более схожи по значениям, чем графики A и C.

Для таких простых объектов мы можем использовать поточечную разность в качестве функции расстояния, но, если мы сталкиваемся с более сложными графиками, то довольно трудно оценить качество работы такой функции. Рассмотрим другой пример. На нём также представлены графики трёх функции. Но в отличии от предыдущего случая, при использовании функции поточечной разницы для нахождения ближайшего графика к объекту X, мы получим объект Z. Мы можем заметить, что у графиков X и Y присутствует период и для нас они более похожи между собой, чем Z в качестве прямой.



Поточнечной разность как функция расстояния никак не учитывает смещение графиков по оси X, если бы мы перед расчётом разницы выравнивали графики между собой наилучшим способом, то мы бы решили эту проблему. Одним из известных алгоритмов пытающихся решить проблему выравнивания (и ещё кроме выравнивание написать про поиск расстояния) двух графиков является метод под названием Dynamic Time Warping или DTW. Его интерфейс довольно прост — он принимает на вход значения двух функций и возвращает расстояние между ними. Данный алгоритм впервые был применён в задачах, связанных с распознаванием речи, сейчас же он используется в различных сферах.

(Подробней написать что имеется ввиду под выравниванием) Алгоритм, реализующий DTW, является классической задачей динамического программирования. Пусть Q = q\_1, q\_2, …, q\_n и S = s\_1, s\_2, …, s\_m значения двух функций, переданные нам в качестве входных данных. Определим вспомогательную матрицу D размером n на m. Зададим начальное значение D[1, 1] = d(q\_1, s\_1), где d является функцией Евклидового расстояния или любой другой функции расстояния. Далее остальные ячейки заполняются с помощью правила D[i, j] = min(D[i, j - 1], D[i - 1, j - 1], D[i - 1, j]) + d(q\_i, s\_j)



Конечное расстояние между двумя объектами будет находится в нижней правой ячейке D[n, m]. Стоит заметить что существует модификация данного алгоритма использующее дополнительное численное ограничение W на матрицу. Данное ограничение позволяет не заполнять пути отстающие от диагонали матрицы D более чем на W ячеек, что существенно ускоряет работу алгоритма. Также нетрудно увидеть что сложность алгоритма DTW составляет O(nm), где n и m количество значений каждой функции. Такая сложность связана с необходимостью заполнения матрицы размера n на m.

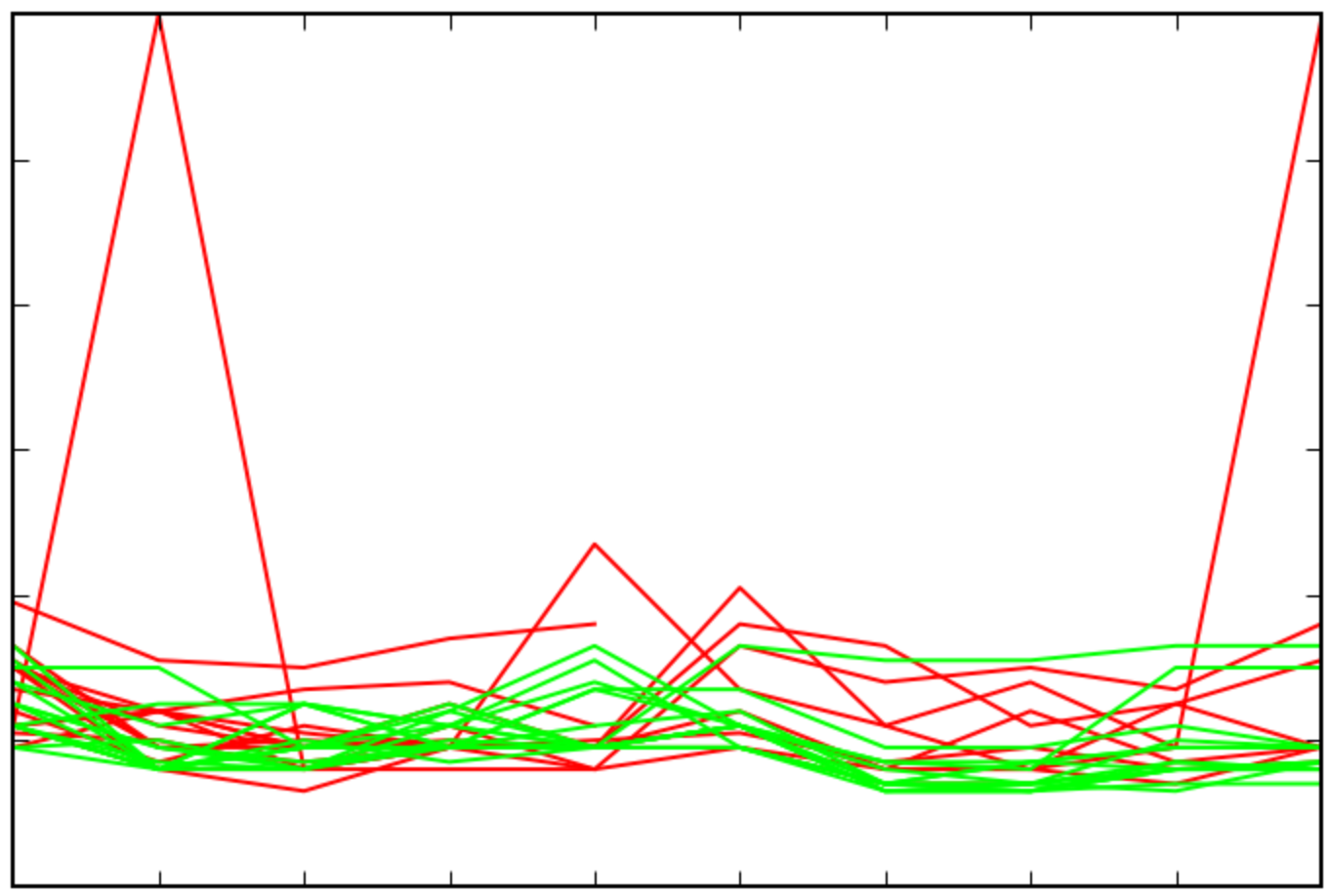
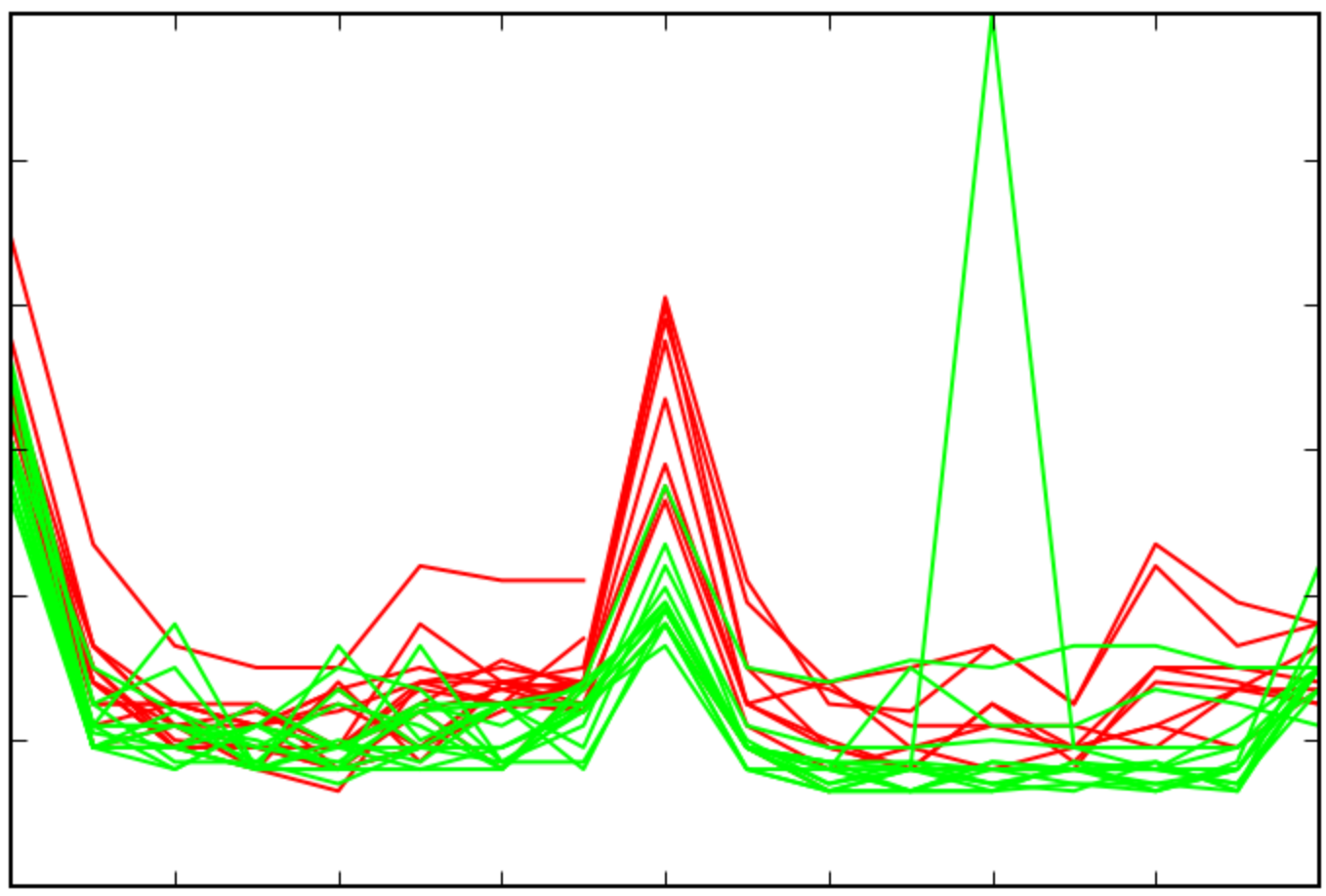
Итак, мы смогли подобрать функцию расстояния между несколькими сигналами, далее мы ей воспользуемся для создания первого слоя классификаторов реализующих метод K ближайших соседей.

# §4.4 Метрическая классификация. Первичный вектор оценок.

Каждый объект, являющийся файлом формата .doc, мы представили набором функций отображающих зависимость между номером определённое операции и временем выполнения этой операции. После перевода сырых данных, представленные утилитой PROCMON, удалось извлеч 24 различных операции: {‘QueryNameInformationFile', 'SetAllocationInformationFile', 'QueryOpen', 'QueryStandardInformationFile', 'LockFile', 'QueryEaInformationFile', 'SetBasicInformationFile', 'QueryStreamInformationFile', 'QueryBasicInformationFile', 'Thread Create', 'SetEndOfFileInformationFile', 'QueryDirectory', 'Process Profiling', 'FileSystemControl', 'Thread Exit', 'CreateFileMapping', 'Load Image', 'UnlockFileSingle', 'ReadFile', 'WriteFile', 'CloseFile', 'QueryAttributeTagFile', 'CreateFile', 'Process Start’}.

Перед тем как начинать работать с графиками, построенными по длительностям собранных операций, нам нужно произвести отбор графиков, точно неподходящих нам. Неподходящими графиками мы будем считать те, которые априори не могут повлиять на конечный результат. Это может быть, например, график с операциями, всегда выполняющихся за время равное 0. Также мы хотим выровнять по длине графики, построенные по одному типу операций, между всеми объектами. Это нужно для избежания искусственного преимущества одного класса перед другим. Так как явным признаком заражения может являться количество точек в области определения функции, потому что процесс, в котором эксплуатируется какая-либо уязвимость, довольно часто завершается аварийно.

После такого этапа фильтрации и выравнивания у нас остаётся 14 операций: {‘QueryNameInformationFile', 'SetAllocationInformationFile', 'QueryOpen', 'QueryStandardInformationFile', 'LockFile', 'QueryBasicInformationFile', 'SetEndOfFileInformationFile', 'QueryDirectory', 'CreateFileMapping', 'UnlockFileSingle', 'ReadFile', 'WriteFile', 'CloseFile', ‘CreateFile’}. ( Описать для каждого признака почему мы его отфильтровали, разделить на группы по причине. Описать процесс отсечения операций с конца графика. )

Для создания и проверки качества модели, нам необходимо разделить исходную выборку на две части: обучающую и тестовую. Далее будем обучать наш метод на одной выборке и получать результаты для оценки на второй. Как и говорилось ранее, обучать мы будем метод одного ближайщего соседа. Весь алгоритм создания классификатора можно свести к следующим действиям: на этапе обучения мы просто запоминаем выборку, переданную нам, а во время предсказания меток для новых объектов мы с помощью ранее описаного алгоритма DTW находим ближайший график, возвращая его метку класса в качестве ответа.

Так как размер отфильтрованного набора графиков операций равен 14, то в итоге на каждый объект из тестовой выборки мы получим вектор оценок, состоящий из 14 элементов, каждый элемент которого будет принадлежать множеству {0, 1}. То есть мы независимо предсказали метку класса по каждому типу операций. По каждому такому вектору нам необходимо вернуть окончательное решение, представляющее итоговую метку исследуемого объекта. Один из самых простых способов отображения вектора меток класса в итоговую метку это взятие большинства, но данный метод никак не учитывает качество работы каждого классификатора. Часто на практике для таких целей используют иной подход: мы можем рассматривать каждый вектор из 14 элементов как входные данные для другого классификатора. В таком случае первый слой классификаторов мы будем называть базовыми классификаторами, а алгоритм, формирующий итоговую оценку, мета-классификатором.

# §4.4 Объединение оценок и качество итоговой классификации.

Если посмотреть на вектора оценок, составленных работой базовых алгоритмов, то мы можем увидеть уже знакомый вид матрицы объект-признак, где каждый элемент строки принадлежит R.

В итоге наш итоговый классификатор приобретёт многослойный вид. На первом слое находится набор из 14 моделей одного ближайшего соседа с метрикой DTW, пытающихся передать сигнал непосредственно из формальных признаков объекта. В качестве мета-алгоритма мы будем использовать линейную модель для конечного объединения оценок в итоговую метку класса. На вход линейной моделе будут переданы набор векторо оценок, полученных базовыми алгоритмами. Такой набор можно аналогично представить как матрицу объект-признак. Данный многослойный способ построения классификаторов много раз показывал себя с лучшей стороны в мировой практике.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| G0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| G1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

(Описать точнее классификацию и данные.)

Если вспомнить, что каждый базовый алгоритм, возвращающий значение в каждой строчке матрицы, строится по отдельной операции, то итоговая линейная модель примет вид взвешенного голосованиями между исходным набором операций. Если по какой-то операции был построен плохой классификатор, часто возвращающий неправильную метку, то во время обучения объединяющей модели такой классификатор получит маленький или вовсе нулевой вес.

Для оценки качества классификации данной конструкции мы разделим исходную выборку объектов на три множества: A, B и C. Используя выборку A, мы будем обучать базовые алгоритмы одного ближайшего соседа, затем предсказав для выборок B и C метки классов по каждому из 14 классификаторов. Таким образом мы получим набор из промежуточных векторов, представляющих матрицу объект-признак. Как последний этап создания классификатора, мы обучим линейный мета-алгоритм на выборке B и предскажем метки классов для объектов из выборки C. Получив качество предсказания выборки C, мы можем поменять между собой выборки B и C и проделать все шаги повторно, получив новый вариант оценок качествах. Таким образом, меняя выборки для обучения каждого из слоёв классификаторов, мы сможем получить 6 варинтов оценок, которые стоит усреднить для борьбы с шумом. На каждом этапе построения моделей важно делать обучение на непересекающихся выборках, иначе измеренное качество может получится смещённым.

|  |  |  |  |
| --- | --- | --- | --- |
| Варианты разбиений | Точность | Полнота | F-мера |
| 1 | 1 | 0.59 | 0.74 |
| 2 | 1 | 1 | 1 |
| 3 | 1 | 0.80 | 0.88 |
| 4 | 0.71 | 1 | 0.83 |
| 5 | 1 | 1 | 1 |
| 6 | 1 | 0.8 | 0.88 |
| Итоговая оценка | 0.95 | 0.86 | 0.89 |

Получив усреднённые оценки, мы увидим, что точность нашего метода близка к единице. На практике это означает, что почти все вредоносные объекты будут верно классифицированы. Чуть меньше оказалась величина полноты равна 0.86. Это говорит нам о том, что, во время использования алгоритма, примерно 14% безвредных объектов будут распознаны как вредоносные.