МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ОТЧЕТ

о прохождении производственной практики

«Средства анализа данных и машинного обучения в Python»

Студент: гр.353502 Фридман А.Я.

Руководитель практики от кафедры:

Михневич М.В. _____

Руководитель практики от организации:

Толстиков А.А. _____

Минск

2016

Оглавление

Введение

Производственная практика является неотъемлемой частью учебного процесса, в ходе которого закрепляется теоретические знания на производстве, направленной на получение практических знаний и навыков профессиональной деятельности. Производственная практика специализирована для расширения представлений о специальности, полученных при теоретическом обучении, а так же для приобретения производственного опыта и конкретных производственных навыков по специальности.

Практика позволяет изучить функциональные обязанности инженера-программиста, получить опыт командной работы, разработки программного обеспечения промышленного уровня.

1 Jupyter Notebook

1.1 Общие сведения

Jupyter Notebook — клиент-серверное приложение позволяющее редактировать и запускать @notebook documents@ в веб-браузере. Приложение может выполняться на локальном десктопном компьютере не имеющего доступа к сети интернет, либо на удаленном сервере. В дополнение к отображению, редактированию, запуску @notebook documents@ позволяет манипулировать файлами.

1.2 Развертывание Jupyter Notebook на удаленном сервере

Для выполнения развертывания необходимо выполнить следующие шаги:

1. Установить python библиотеку jupyter

```
$ pip install jupyter
```

2. Вычислить хэш от желаемого пароля по алгоритму sha1. Для этого в IPython необходимо выполнить следующие команды:

```
In [1]: from IPython.lib import passwd
In [2]: passwd()
Enter password:
Verify password:
Out[1]: 'sha1:2ac16fba64bd:01a052108dfca7f73cd277179ae58235f0698173'
```

3. Сгенерировать стандартный конфигурационный файл

```
$ jupyter-notebook —generate-config
```

4. Сгенерировать пару ssh ключей

```
$ openssl req -x509 -nodes -newkey rsa:1024 -keyout cert.pem -out cert.pem
```

5. Дописать следующие строки в конфигурационный файл:

```
c = get_config()
c.IPKernelApp.pylab = 'inline'
c.NotebookApp.certfile = u'/abs/path/to/certificate/cert.pem'
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.password = u
'sha1:2ac16fba64bd:01a052108dfca7f73cd277179ae58235f0698173'
c.NotebookApp.port = 8888
```

Для запуска приложения нужно выполнить команду

```
$ jupyter-notebook
```

1.3 «Магические функции» в Jupyter Notebook

В Jupyter Notebook присутствует множество команд влияние которых может распространяться как на одну ячейку, так и на весь документ.

1. %alias — объявляет псевдоним для системной команды.

```
In [2]: alias bracket echo "Input in brackets: <%l>"
In [3]: bracket hello world
Input in brackets: <hello world>
```

2. %cd — сменить текущую рабочую директорию

```
In [10]: cd parent/child
/home/tsuser/parent/child
```

%debug — активировать интерактивный отладчик

```
%debug [--breakpoint FILE:LINE] [statement [statement …]]
```

%dhist — отобразить историю песенных директорий

%doctest_mode — включить режим doctest

%env — получить, установить список переменных среды.

%history — вывести список недавно использованных переменных

```
%history [-n] [-o] [-p] [-t] [-f FILENAME] [-g [PATTERN [PATTERN ...]]]
             [-l [LIMIT]] [-u]
             [range [range …]]
```

%pastebin — загрузить код на pastebin

%pdoc — вывести док-строку объекта
%precision — установить точность вывода для вещественных чисел

```
In [1]: from math import pi

In [2]: %precision 3
Out[2]: u'%.3f'

In [3]: pi
Out[3]: 3.142
```

%psource — вывести исходный код объекта
%reset — очистить пространство имён
%run — выполнить скрипт

```
%run [-n -i -e -G]
     [( -t [-N<N>] | -d [-b<N>] | -p [profile options] )]
     ( -m mod | file ) [args]
```

%time — вывести время выполнения блока кода

```
In [1]: %time 2**128
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 0.00
Out[1]: 340282366920938463463374607431768211456L

In [2]: n = 1000000

In [3]: %time sum(range(n))
CPU times: user 1.20 s, sys: 0.05 s, total: 1.25 s
Wall time: 1.37
Out[3]: 499999500000L
```

2 Машинное обучение

2.1 Введение

Машинное обучение (англ. Machine Learning) — обширный подраздел искусственного интеллекта, математическая дисциплина, использующая разделы математической статистики, численных методов оптимизации, теории вероятностей, дискретного анализа, и извлекающая знания из данных. Различают два типа обучения. Обучение по прецедентам, или индуктивное обучение, основано на выявлении закономерностей в эмпирических данных. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний. Дедуктивное обучение принято относить к области экспертных систем, поэтому термины машинное обучение и обучение по прецедентам можно считать синонимами.

Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации (Information Extraction), интеллектуальным анализом данных (Data mining).

2.2 Классификация

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

В математической статистике задачи классификации называются также задачами дискриминантного анализа.

В машинном обучении задача классификации относится к разделу обучения с учителем. Существует также обучение без учителя, когда разделение объектов обучающей выборки на классы не задаётся, и требуется классифицировать объекты только на основе их сходства друг с другом. В этом случае принято говорить о задачах кластеризации или таксономии, и классы называть, соответственно, кластерами или таксонами.

Типы входных данных

- Признаковое описание — наиболее распространённый случай. Каждый объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми.
- Матрица расстояний между объектами. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки. С этим типом входных данных работают немногие методы, в частности, метод ближайших соседей, метод парзеновского окна, метод потенциальных функций.
- Временной ряд или сигнал представляет собой последовательность измерений во времени. Каждое измерение может представляться числом, вектором, а в общем случае — признаковым описанием исследуемого объекта в данный момент времени.
- Изображение или видеоряд.
- Встречаются и более сложные случаи, когда входные данные представляются в виде графов, текстов, результатов запросов к базе данных, и т. д. Как правило, они приводятся к первому или второму случаю путём предварительной обработки данных и извлечения признаков.

Классификацию сигналов и изображений называют также распознаванием образов.

Типы классов

- Двухклассовая классификация. Наиболее простой в техническом отношении случай, который служит основой для решения более сложных задач.
- Многоклассовая классификация. Когда число классов достигает многих тысяч (например, при распознавании иероглифов или слитной речи), задача классификации становится существенно более трудной.
- Непересекающиеся классы.
- Пересекающиеся классы. Объект может относиться одновременно к нескольким классам.
- Нечёткие классы. Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

Классификация: формальная постановка

Пусть X — множество описаний объектов, Y — конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость — отображение $y^*\colon X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1,y_1),\dots,(x_m,y_m)\}$. Требуется построить алгоритм $a\colon X \to Y$, способный классифицировать произвольный объект $x \in X$.

Вероятностная постановка задачи

Более общей считается вероятностная постановка задачи. Предполагается, что множество пар «объект, класс» $X \times Y$ является вероятностным пространством с неизвестной вероятностной мерой P. Имеется конечная обучающая выборка наблюдений $X^m = \{(x_1,y_1),\dots,(x_m,y_m)\}$, сгенерированная согласно вероятностной мере P. Требуется построить алгоритм $a\colon X \to Y$, способный классифицировать произвольный объект $x \in X$.

Признаковое пространство

Признаком называется отображение $f\colon X \to D_f$, где D_f — множество допустимых значений признака. Если заданы признаки $f_1,\dots,f_n$, то вектор $x = (f_1(x),\dots,f_n(x))$ называется признаковым описанием объекта $x \in X$. Признаковые описания допустимо отождествлять с самими объектами. При этом множество $X = D_{f_1} \times \dots \times D_{f_n}$ называют признаковым пространством.

В зависимости от множества D_f признаки делятся на следующие типы:

- бинарный признак: $D_f = \{0,1\}$;
- номинальный признак: D_f — конечное множество;
- порядковый признак: D_f — конечное упорядоченное множество;
- количественный признак: D_f — множество действительных чисел.

Часто встречаются прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.

2.3 Регрессия

Регрессионный анализ — метод моделирования измеряемых данных и исследования их свойств. Данные состоят из пар значений зависимой переменной (переменной отклика) и независимой переменной (объясняющей переменной). Регрессионная модель есть функция независимой переменной и параметров с добавленной случайной переменной. Параметры модели настраиваются таким образом, что модель наилучшим образом приближает данные. Критерием качества приближения (целевой функцией) обычно является среднеквадратичная ошибка: сумма квадратов разности значений модели и зависимой переменной для всех значений независимой переменной в качестве аргумента. Регрессионный анализ — раздел математической статистики и машинного обучения. Предполагается, что зависимая переменная есть сумма значений некоторой модели и случайной величины. Относительно характера распределения этой величины делаются предположения, называемые гипотезой порождения данных. Для подтверждения или опровержения этой гипотезы выполняются статистические тесты, называемые анализом остатков. При этом предполагается, что независимая переменная не содержит ошибок. Регрессионный анализ используется для прогноза, анализа временных рядов, тестирования гипотез и выявления скрытых взаимосвязей в данных.

Формальная постановка задачи регрессии

Регрессия — зависимость математического ожидания (например, среднего значения) случайной величины от одной или нескольких других случайных величин (свободных переменных), то есть $E(y|x) = f(x)$. Регрессионным анализом называется поиск такой функции f, которая описывает эту зависимость. Регрессия может быть представлена в виде суммы неслучайной и случайной составляющих.

$$y = f(x) + \nu,$$

где f — функция регрессионной зависимости, а ν — аддитивная случайная величина с нулевым матожиданием. Предположение о характере распределения этой величины называется гипотезой порождения данных. Обычно предполагается, что величина ν имеет гауссово распределение с нулевым средним и дисперсией σ_ν^2.

Задача нахождения регрессионной модели нескольких свободных переменных ставится следующим образом. Задана выборка — множество $\{x_1,\dots,x_N | x \in \mathbb{R}^M\}$ значений свободных переменных и множество $\{y_1,\dots,y_N | y \in \mathbb{R}\}$ соответствующих им значений зависимой переменной. Эти множества обозначаются как D, множество исходных данных $\{(x,y)_i\}$. Задана регрессионная модель — параметрическое семейство функций $f(w,x)$ зависящая от параметров $w \in \mathbb{R}$ и свободных переменных x. Требуется найти наиболее вероятные параметры $\overline{w}$:

$$\overline{w} = \arg\max_{w \in \mathbb{R}^W} p(y|x,w,f) = p(D|w,f).$$

Функция вероятности p зависит от гипотезы порождения данных и задается Байесовским выводом или методом наибольшего правдоподобия.

2.4 Кластеризация

Кластерный анализ (Data clustering) — задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Задача кластеризации относится к широкому классу задач обучения без учителя.

Типология задач кластеризации

Типы входных данных

- Признаковое описание объектов. Каждый объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми.
- Матрица расстояний между объектами. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки.

Матрица расстояний может быть вычислена по матрице признаковых описаний объектов бесконечным числом способов, в зависимости от того, как ввести функцию расстояния (метрику) между признаковыми описаниями. Часто используется евклидова метрика, однако этот выбор в большинстве случаев является эвристикой и обусловлен лишь соображениями удобства.

Обратная задача — восстановление признаковых описаний по матрице попарных расстояний между объектами — в общем случае не имеет решения, а приближённое решение не единственно и может иметь существенную погрешность. Эта задача решается методами многомерного шкалирования.

Таким образом, постановка задачи кластеризации по матрице расстояний является более общей. С другой стороны, при наличии признаковых описаний часто удаётся строить более эффективные методы кластеризации.

Цели кластеризации

- Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»).
- Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера.
- Обнаружение новизны (novelty detection). Выделяются нетипичные объекты, которые не удаётся присоединить ни к одному из кластеров.

В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую (или фиксированную) степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются задачами таксономии.

Результатом таксономии является древообразная иерархическая структура. При этом каждый объект характеризуется перечислением всех кластеров, которым он принадлежит, обычно от крупного к мелкому. Визуально таксономия представляется в виде графика, называемого дендрограммой.

Классическим примером таксономии на основе сходства является биноминальная номенклатура живых существ, предложенная Карлом Линнеем в середине XVIII века. Аналогичные систематизации строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов.

Формальная постановка задачи кластеризации

Пусть X — множество объектов, Y — множество номеров (имён, меток) кластеров. Задана функция расстояния между объектами $\rho(x,x')$. Имеется конечная обучающая выборка

объектов $X^m = \{x_1, \ldots, x_m\} \subset X$. Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^m$ приписывается номер кластера y_i.
Алгоритм кластеризации — это функция $a: X \to Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$. Множество Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.
Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки исходных объектов y_i изначально не заданы, и даже может быть неизвестно само множество Y.
Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин:

- Не существует однозначно наилучшего критерия качества кластеризации. Известен целый ряд эвристических критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты.
- Число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием.
- Результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

2.5 Линейные модели

2.5.1 Линейная регрессия

Линейная регрессия — метод восстановления зависимости между двумя переменными. Ниже приведен пример программы, которая строит линейную модель зависимости по заданной выборке и показывает результат на графике.

Для заданного множества из m пар $(x_i,y_i), i=1,\ldots,m$, значений свободной и зависимой переменной требуется построить зависимость. Назначена линейная модель

$$y_i = f(w,x_i)+\varepsilon_i$$

с аддитивной случайной величиной ε. Переменные x,y принимают значения на числовой прямой $\mathbb{R}$. Предполагается, что случайная величина распределена нормально с нулевым матожиданием и фиксированной дисперсией σ_ε^2, которая не зависит от переменных x,y. При таких предположениях параметры w регрессионной модели вычисляются с помощью метода наименьших квадратов.

Одномерная регрессия

Определим модель зависимости как

$$y_i = w_1+w_2x_i+\varepsilon_i.$$

Согласно методу наименьших квадратов, искомый вектор параметров $w=(w_1,w_2)^T$ есть решение нормального уравнения

$$w=(A^TA)^{-1}A^Ty,$$

где y — вектор, состоящий из значений зависимой переменной, $y=(y_1,\ldots,y_m)$.

Столбцы матрицы A есть подстановки значений свободной переменной $x_i^0 \to a_{i1}$ и $x_i^1 \to a_{i2}, i=1,\ldots,m$. Матрица имеет вид

$$A=\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \ldots & \ldots \\ 1 & x_m \end{pmatrix}.$$

Зависимая переменная восстанавливается по полученным весам и заданным значениям свободной переменной

$$y_i^* = w_1+w_2x_i,$$

иначе

$$y^*=Aw.$$

% Для оценки качества модели используется критерий суммы квадратов регрессионных остатков, SSE — Sum of Squared Errors.

$$SSE=\sum_{i=1}^{m}(y_i-y_i^*)^2=(y-y^*)^T(y-y^*).$$

2.5.2 Многомерная линейная регрессия

Многомерная линейная регрессия — это линейная регрессия в n-мерном пространстве (объекты и признаки являются n-мерными векторами).

Имеется множество объектов $X=\mathbb{R}^n$ и множество ответов $Y=\mathbb{R}$. Также имеется набор n вещественнозначных признаков $f_j(x),\ j=1,\ \ldots,\ n$. Введём матричные

обозначения: матрицу информации F, целевой вектор y, вектор параметров α и диагональную матрицу весов:

$$F=\begin{pmatrix} f_1 \cdots f_n \end{pmatrix},\ f_i=\begin{pmatrix} f_i(x_1) \\ \vdots \\ f_i(x_l) \end{pmatrix},\ y=\begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix},\ \alpha=\begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix},\ W=diag(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_l})$$

Алгоритм:

$$a(x) = \sum_{j=1}^{n} \alpha_j f_j(x) = F\alpha$$

Оценим качество его работы на выборке $X^l=(x_i, y_i)_{i=1}^l \in X^*Y$ методом наименьших квадратов:

$$Q(\alpha,X^l) = \sum_{i=1}^{l} w_i(a(x_i)-y_i)^2 \to \min_{\alpha\in\mathbb{R}^n}$$, или, в матричных обозначениях,

$$Q(\alpha) = \|W(F\alpha - y)\|^2 \to \min_{\alpha\in\mathbb{R}^n}.$$

Задача с произвольной матрицей весов легко приводится к единичной матрице весов заменой $F'=WF$, $y'=Wy$.

$$Q(\alpha) = \|F'\alpha - y'\|^2 = (F'\alpha - y')^\top(F'\alpha - y').$$

Таким образом, в дальнейшем будем рассматривать только задачу с единичными весами.

Найдём минимум $Q(\alpha)$ по α:

$$\frac{\partial Q(\alpha)}{\partial \alpha}=2F^T(F\alpha-y)=0 \Rightarrow (F^TF)\alpha=F^Ty.$$

Если $rank(F^TF)=n$, то можно обращать матрицу

$F^TF : \alpha^*=(F^TF)^{-1}F^Ty=F^+y$, где введено обозначение

$F^+=(F^TF)^{-1}F^T$.

В таком случае функционал качества записывается в более удобной форме:

$Q(\alpha^*)=\|F(F^TF)^{-1}F^Ty-y\|^2=\|P_Fy-y\|^2$, где P_F — проекционная матрица:

P_Fy — вектор, являющийся проекцией y на $\mathcal{L}(f_1, \cdots, f_n)$.

Теперь рассмотрим сингулярное разложение матрицы F:

$$F = VDU^T.$$

В таких обозначениях:

$F^+ = (F^TF)^{-1}F^T = (UDV^TVDU^T)^{-1}UDV^T = (UDDU^T)^{-1}UDV^T = U^{-T}D^{-2}U^{-1}UDV^T = U^{-T}D^{-2}DV^T$

$$F^+ = UD^{-1}V^T = \sum_{j=1}^{n}\frac{1}{\sqrt{\lambda_j}}u_jv_j^T$$

, а так как $U^{-1} = U^T$, то в силу диагональности матрицы *D*.

А решение метода наименьших квадратов запишется в следующем виде:

$$\alpha^* = F^+ y = \sum_{j=1}^{n} \frac{1}{\sqrt{\alpha_j}} u_j (v_j^T, y);$$

А так как $\|\alpha\|^2 = \alpha^T \alpha$, то

$$\|\alpha^*\|^2 = \|UD^{-1}V^T y\|^2 = y^T V D^{-T} U^T U D^{-1} V^T y = y^T V D^{-2} V^T y = \|D^{-1}V^T y\|^2 = \sum_{j=1}^{n} \frac{1}{\alpha_j} (v_j^T, y)^2.$$

2.5.3 Логистическая регрессия

Логистическая регрессия (Logistic regression) — метод построения линейного классификатора, позволяющий оценивать апостериорные вероятности принадлежности объектов классам.

Пусть объекты описываются n числовыми признаками $f_j: X \to \mathbb{R},\ j = 1,\ldots,n$.

Тогда пространство признаковых описаний объектов есть $X = \mathbb{R}^n$. Пусть Y — конечное множество номеров (имён, меток) классов.

Пусть задана обучающая выборка пар «объект, ответ»

$$X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}.$$

Случай двух классов

Положим $Y = \{-1, +1\}$. В логистической регрессии строится линейный алгоритм классификации $a: X \to Y$ вида

$$a(x, w) = \operatorname{sign}\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right) = \operatorname{sign}\langle x, w \rangle,$$

где w_j — вес j-го признака, w_0 — порог принятия решения,

$w = (w_0, w_1, \ldots, w_n)$ — вектор весов, $\langle x, w \rangle$ — скалярное произведение признакового описания объекта на вектор весов. Предполагается, что искусственно введён «константный» нулевой признак: $f_0(x) = -1$.

Задача обучения линейного классификатора заключается в том, чтобы по выборке X^m настроить вектор весов w. В логистической регрессии для этого решается задача минимизации эмпирического риска с функцией потерь специального вида:

(1)

$$Q(w) = \sum_{i=1}^{m} \ln\left(1 + \exp(-y_i \langle x_i, w \rangle)\right) \to \min_w.$$

После того, как решение w найдено, становится возможным не только вычислять классификацию $a(x) = \operatorname{sign}\langle x, w \rangle$ для произвольного объекта x, но и оценивать апостериорные вероятности его принадлежности классам:

(2)

$$\mathbb{P}\{y|x\} = \sigma\big(y\langle x, w \rangle\big), \quad y \in Y,$$

где $\sigma(z) = \frac{1}{1 + e^{-z}}$ — сигмоидная функция. Во многих приложениях апостериорные вероятности необходимы для оценивания рисков, связанных с возможными ошибками классификации.

Обоснования

С точки зрения минимизации эмпирического риска

Введём понятие отступа (margin) объекта

$$M(x_i) = y_i \langle x_i, w \rangle.$$

Отступ можно понимать, как «степень погруженности» объекта в свой класс. Чем меньше значение отступа $M(x_i)$, тем ближе объект подходит к границе классов. Отступ $M(x_i)$ отрицателен тогда и только тогда, когда алгоритм $a(x,w)$ допускает ошибку на объекте x_i. Число ошибок классификации можно записать через отступы:

$$Q_0(w) = \sum_{i=1}^{m} [M(x_i) < 0].$$

Под знаком суммы стоит пороговая функция потерь, поэтому данный функционал не является ни выпуклым, ни даже непрерывным, и минимизировать его неудобно. Идея заключается в том, чтобы заменить пороговую функцию потерь непрерывной оценкой сверху:

$$[M < 0] \leq \log_2(1 + e^{-M}).$$

В результате такой замены и получается функционал (1).

С точки зрения байесовской классификации

Наиболее строгое обоснование логистической регрессии опирается на следующую теорему.

Теорема. Пусть:

- функции правдоподобия (плотности распределения) классов $p_y(x)$ принадлежат экспонентному семейству плотностей $p_y(x) = \exp\big(\langle \theta, x \rangle \cdot a(\delta) + b(\delta, \theta) + d(x, \delta)\big)$, где a, b d — произвольные функции;
- функции правдоподобия имеют равные значения параметра разброса δ и отличаются только значениями параметра сдвига θ_y;
- среди признаков есть константа, скажем, $f_0(x) = -1$.

Тогда

- линейный классификатор является оптимальным байесовским классификатором;
- апостериорные вероятности классов оценивается по формуле (2);
- минимизация функционала (1) эквивалентна максимизации правдоподобия выборки.

Таким образом, оценки апостериорных вероятностей (2) являются точными только при довольно сильных теоретико-вероятностных предположениях. На практике гарантировать выполнение этих условий вряд ли возможно. Поэтому трактовать выходы сигмоидных функций как вероятности следует с большой осторожностью. На самом деле они дают лишь оценку удалённости объекта от границы классов, нормированную так, чтобы она принимала значения из отрезка $[0,1]$.

2.6 Древовидные модели

2.6.1 Решающее дерево

Решающее дерево — средство поддержки принятия решений, использующееся в статистике и анализе данных для прогнозных моделей. Структура дерева представляет собой «листья» и «ветки». На ребрах («ветках») дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах — атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. Подобные деревья решений широко используются в интеллектуальном анализе данных. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной на основе нескольких переменных на входе.

Каждый лист представляет собой значение целевой переменной, измененной в ходе движения от корня по листу. Каждый внутренний узел соответствует одной из входных переменных. Дерево может быть также «изучено» разделением исходных наборов переменных на подмножества, основанные на тестировании значений атрибутов. Это процесс, который повторяется на каждом из полученных подмножеств. Рекурсия завершается тогда, когда подмножество в узле имеет те же значения целевой переменной, таким образом, оно не добавляет ценности для предсказаний. Процесс, идущий «сверху вниз», индукция деревьев решений (TDIDT)[1], является примером поглощающего «жадного» алгоритма, и на сегодняшний день является наиболее распространенной стратегией деревьев решений для данных, но это не единственная возможная стратегия. В интеллектуальном анализе данных, деревья решений могут быть использованы в качестве математических и вычислительных методов, чтобы помочь описать, классифицировать и обобщить набор данных, которые могут быть записаны следующим образом:

$$(x, Y) = (x_1, x_2, x_3 \ldots x_k, Y)$$

Зависимая переменная Y является целевой переменной, которую необходимо проанализировать, классифицировать и обобщить. Вектор x состоит из входных переменных x_1, x_2, x_3 и т. д., которые используются для выполнения этой задачи.

2.6.2 Типология деревьев

Деревья решений, используемые в Data Mining, бывают двух основных типов:

- Анализ дерева классификации, когда предсказываемый результат является классом, к которому принадлежат данные;
- Регрессионный анализ дерева, когда предсказанный результат можно рассматривать как вещественное число (например, цена на дом, или продолжительность пребывания пациента в больнице).

Упомянутые выше термины впервые были введены Брейманом и др. Перечисленные типы имеют некоторые сходства, а также некоторые различия, такие, как процедура используется для определения, где разбивать.

Некоторые методы позволяют построить более одного дерева решений:

1. Деревья решений «мешок», наиболее раннее дерево решений, строит несколько деревьев решений, неоднократно интерполируя данные с заменой, и деревья голосований для прогноза консенсуса;
2. Случайный классификатор «лесной» использует ряд деревьев решений, с целью улучшения ставки классификации;
3. «Повышенные» деревья могут быть использованы для регрессионного типа и классификации типа проблем.
4. «Вращение леса» — деревья, в которых каждое дерево решений анализируется первым применением метода главных компонент (PCA) на случайные подмножества входных функций.

2.6.3 Алгоритмы построения решающего дерева

Общая схема построения дерева принятия решений по тестовым примерам выглядит следующим образом:

- Выбираем очередной атрибут Q, помещаем его в корень.
- Для всех его значений i:
    - Оставляем из тестовых примеров только те, у которых значение атрибута Q равно i
    - Рекурсивно строим дерево в этом потомке
- Основной вопрос состоит в том, как выбирать очередной атрибут.
- Есть различные способы выбирать очередной атрибут:
- Алгоритм ID3, где выбор атрибута происходит на основании прироста информации (англ. Gain), либо на основании индекса Гини.
- Алгоритм C4.5 (улучшенная версия ID3), где выбор атрибута происходит на основании нормализованного прироста информации (англ. Gain Ratio).
- Алгоритм CART и его модификации — IndCART, DB-CART.
- Автоматический детектор взаимодействия Хи-квадрат (CHAID). Выполняет многоуровневое разделение при расчете классификации деревьев;
- MARS: расширяет деревья решений для улучшения обработки цифровых данных.

На практике в результате работы этих алгоритмов часто получаются слишком детализированные деревья, которые при их дальнейшем применении дают много ошибок. Это связано с явлением переобучения. Для сокращения деревьев используется отсечение ветвей (англ. pruning).

2.6.4 Регулирование глубины дерева

Регулирование глубины дерева — это техника, которая позволяет уменьшать размер дерева решений, удаляя участки дерева, которые имеют маленький вес. Один из вопросов, который возникает в алгоритме дерева решений — это оптимальный размер конечного дерева. Так, небольшое дерево может не охватить ту или иную важную информацию о выборочном пространстве. Тем не менее, трудно сказать, когда алгоритм должен остановиться, потому что невозможно спрогнозировать, добавление какого узла позволит значительно уменьшить ошибку. Эта проблема известна как «эффект горизонта». Тем не менее, общая стратегия ограничения дерева сохраняется, то есть удаление узлов реализуется в случае, если они не дают дополнительной информации. Необходимо отметить, что регулирование глубины дерева должно уменьшить размер обучающей модели дерева без уменьшения точности её прогноза или с помощью перекрестной проверки. Есть много методов регулирования глубины дерева, которые отличаются измерением оптимизации производительности.

2.7 Метод ближайших соседей

Метод ближайших соседей — простейший метрический классификатор, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

2.7.1 Введение

Метод ближайшего соседа является, пожалуй, самым простым алгоритмом классификации. Классифицируемый объект x относится к тому классу y_i, которому принадлежит ближайший объект обучающей выборки x_i.

Метод k ближайших соседей. Для повышения надёжности классификации объект относится к тому классу, которому принадлежит большинство из его соседей — k ближайших к нему объектов обучающей выборки x_i. В задачах с двумя классами число соседей берут нечётным, чтобы не возникало ситуаций неоднозначности, когда одинаковое число соседей принадлежат разным классам.

Метод взвешенных ближайших соседей. В задачах с числом классов 3 и более нечётность уже не помогает, и ситуации неоднозначности всё равно могут возникать. Тогда i-му соседу приписывается вес w_i, как правило, убывающий с ростом ранга соседа i. Объект относится к тому классу, который набирает больший суммарный вес среди k ближайших соседей.

2.7.2 Гипотеза компактности

Все перечисленные методы неявно опираются на одно важное предположение, называемое гипотезой компактности: если мера сходства объектов введена достаточно удачно, то схожие объекты гораздо чаще лежат в одном классе, чем в разных. В этом случае граница между классами имеет достаточно простую форму, а классы образуют компактно локализованные области в пространстве объектов. (Заметим, что в математическом анализе компактными называются ограниченные замкнутые множества. Гипотеза компактности не имеет ничего общего с этим понятием, и должна пониматься скорее в «бытовом» смысле этого слова.)

2.7.3 Основная формула

Пусть задана обучающая выборка пар «объект-ответ»

$$X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}.$$

Пусть на множестве объектов задана функция расстояния $\rho(x, x')$. Эта функция должна быть достаточно адекватной моделью сходства объектов. Чем больше значение этой функции, тем менее схожими являются два объекта x, x'.

Для произвольного объекта u расположим объекты обучающей выборки x_i в порядке возрастания расстояний до u:

$$\rho(u, x_{1;u}) \leq \rho(u, x_{2;u}) \leq \dots \leq \rho(u, x_{m;u}),$$

где через $x_{i;u}$ обозначается тот объект обучающей выборки, который является i-м соседом объекта u. Аналогичное обозначение введём и для ответа на i-м соседе: $y_{i;u}$. Таким образом, произвольный объект u порождает свою перенумерацию выборки. В наиболее общем виде алгоритм ближайших соседей есть

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{m} [y(x_{i;u}) = y] w(i,u),$$ где $w(i,u)$ — заданная *весовая функция*, которая оценивает степень важности i-го соседа для классификации объекта u. Естественно полагать, что эта функция неотрицательна и не возрастает по i. По-разному задавая весовую функцию, можно получать различные варианты метода ближайших соседей.

- $w(i,u) = [i = 1]$ — простейший метод ближайшего соседа;
- $w(i,u) = [i \le k]$ — метод k ближайших соседей;
- $w(i,u) = [i \le k] q^i$ — метод k экспоненциально взвешенных ближайших соседей, где предполагается $q < 1$;
- $w(i,u) = K\left(\frac{\rho(u,x_{i;u})}{h}\right)$ — метод парзеновского окна фиксированной ширины h;
- $w(i,u) = K\left(\frac{\rho(u,x_{i;u})}{\rho(u,x_{k+1;u})}\right)$ — метод парзеновского окна переменной ширины;
- $w(i,u) = K\left(\frac{\rho(u,x_{i;u})}{h(x_{i;u})}\right)$ — метод потенциальных функций, в котором ширина окна $h(x_i)$ зависит не от классифицируемого объекта, а от обучающего объекта x_i.

Здесь $K(r)$ — заданная неотрицательная монотонно невозрастающая функция на $[0,+\infty)$, ядро сглаживания.

2.7.4 Основные проблемы и способы их решения

Выбор числа соседей k

При $k = 1$ алгоритм ближайшего соседа неустойчив к шумовым выбросам: он даёт ошибочные классификации не только на самих объектах-выбросах, но и на ближайших к ним объектах других классов. При $k = m$, наоборот, алгоритм чрезмерно устойчив и вырождается в константу. Таким образом, крайние значения k нежелательны. На практике оптимальное значение параметра k определяют по критерию скользящего контроля, чаще всего — методом исключения объектов по одному (leave-one-out cross-validation).

Отсев шума (выбросов)

Обычно объекты обучения не являются равноценными. Среди них могут находиться типичные представители классов — *эталоны*. Если классифицируемый объект близок к эталону, то, скорее всего, он принадлежит тому же классу. Ещё одна категория объектов — *неинформативные* или *периферийные*. Они плотно окружены другими объектами того же класса. Если их удалить из выборки, это практически не отразится на качестве классификации. Наконец, в выборку может попасть некоторое количество *шумовых выбросов* — объектов, находящихся «в гуще» чужого класса. Как правило, их удаление только улучшает качество классификации.

Исключение из выборки шумовых и неинформативных объектов даёт несколько преимуществ одновременно: повышается качество классификации, сокращается объём хранимых данных и уменьшается время классификации, затрачиваемое на поиск ближайших эталонов.
Идея отбора эталонов реализована в алгоритме STOLP, описанном в книге Н. Г. Загоруйко.

Сверхбольшие выборки

Метод ближайших соседей основан на явном хранении всех обучающих объектов.

Сверхбольшие выборки ($m \gg 10^3$) создают несколько чисто технических проблем: необходимо не только хранить большой объём данных, но и уметь быстро находить среди них k ближайших соседей произвольного объекта u.
Проблема решается двумя способами:

- выборка *прореживается* путём выбрасывания неинформативных объектов (см. выше);
- применяются специальные индексы и эффективные структуры данных для быстрого поиска ближайших соседей (например, kd-деревья).

2.7.5 Проблема выбора метрики

Это наиболее сложная из всех проблем. В практических задачах классификации редко встречаются такие «идеальные случаи», когда заранее известна хорошая функция расстояния $\rho(x, x')$. Если объекты описываются числовыми векторами, часто берут евклидову метрику. Этот выбор, как правило, ничем не обоснован — просто это первое, что приходит в голову. При этом необходимо помнить, что все признаки должны быть измерены «в одном масштабе», а лучше всего — отнормированы. В противном случае признак с наибольшими числовыми значениями будет доминировать в метрике, остальные признаки, фактически, учитываться не будут.
Однако и нормировка является весьма сомнительной эвристикой, так как остаётся вопрос: «неужели все признаки одинаково значимы и должны учитываться примерно с одинаковым весом?»
Если признаков слишком много, а расстояние вычисляется как сумма отклонений по отдельным признакам, то возникает проблема проклятия размерности. Суммы большого числа отклонений с большой вероятностью имеют очень близкие значения (согласно закону больших чисел). Получается, что в пространстве высокой размерности все объекты примерно одинаково далеки друг от друга; выбор k ближайших соседей становится практически произвольным.
Проблема решается путём отбора относительно небольшого числа информативных признаков (features selection). В алгоритмах вычисления оценок строится множество различных наборов признаков (т.н. опорных множеств), для каждого строится своя функция близости, затем по всем функциям близости производится голосование.

2.8 Методы валидации модели

2.8.1 Скользящий контроль

Скользящий контроль или **кросс-проверка,** или **кросс-валидация** (cross-validation, CV) — процедура эмпирического оценивания обобщающей способности алгоритмов, обучаемых по прецедентам.

Фиксируется некоторое множество разбиений исходной выборки на две подвыборки: обучающую и контрольную. Для каждого разбиения выполняется настройка алгоритма по обучающей подвыборке, затем оценивается его средняя ошибка на объектах контрольной подвыборки. Оценкой скользящего контроля называется средняя по всем разбиениям величина ошибки на контрольных подвыборках.

Если выборка независима, то средняя ошибка скользящего контроля даёт несмещённую оценку вероятности ошибки. Это выгодно отличает её от средней ошибки на обучающей выборке, которая может оказаться смещённой (оптимистически заниженной) оценкой вероятности ошибки, что связано с явлением переобучения.

Скользящий контроль является стандартной методикой тестирования и сравнения алгоритмов классификации, регрессии и прогнозирования.

2.8.2 Определения и обозначения

Рассматривается задача обучения с учителем.

Пусть X — множество описаний объектов, Y — множество допустимых ответов.

Задана конечная выборка прецедентов $X^L = (x_i, y_i)_{i=1}^L \subset X \times Y$.

Задан алгоритм обучения — отображение μ, которое произвольной конечной выборке прецедентов X^m ставит в соответствие функцию (алгоритм) $a\colon X \to Y$.

Качество алгоритма a оценивается по произвольной выборке прецедентов X^m с помощью функционала качества $Q(a, X^m)$. Для процедуры скользящего контроля не важно, как именно вычисляется этот функционал. Как правило, он аддитивен по объектам выборки:

$$Q(a, X^m) = \frac{1}{m} \sum_{x_i \in X^m} \mathcal{L}(a(x_i), y_i),$$

где $\mathcal{L}(a(x_i), y_i)$ — неотрицательная функция потерь, возвращающая величину ошибки ответа алгоритма $a(x_i)$ при правильном ответе y_i.

2.8.3 Процедура скользящего контроля

Выборка X^L разбивается N различными способами на две непересекающиеся подвыборки: $X^L = X_n^m \cup X_n^k$, где X_n^m — обучающая подвыборка длины m, X_n^k — контрольная подвыборка длины $k = L - m$, $n = 1, \ldots, N$ — номер разбиения.

Для каждого разбиения n строится алгоритм $a_n = \mu(X_n^m)$ и вычисляется значение функционала качества $Q_n = Q(a_n, X_n^k)$. Среднее арифметическое значений Q_n по всем разбиениям называется оценкой скользящего контроля:

$$CV(\mu, X^L) = \frac{1}{N} \sum_{n=1}^{N} Q(\mu(X_n^m), X_n^k).$$

Различные варианты скользящего контроля отличаются видами функционала качества и способами разбиения выборки.

2.8.4 Доверительное оценивание

Кроме среднего значения качества на контроле строят также доверительные интервалы.

Непараметрическая оценка доверительного интервала. Строится вариационный ряд значений $Q_n = Q(a_n, X^k_n)$, $n = 1, \ldots, N$:

$Q^{(1)} \leq Q^{(2)} \leq \ldots \leq Q^{(N)}$.

Утверждение 1. Если разбиения осуществлялись случайно, независимо и равновероятно, то с вероятностью $\eta = \frac{t}{N+1}$ значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N-t+1)}$.

Следствие 1. Значение случайной величины $Q(a(X^m), X^k)$ не превосходит $Q^{(N)}$ с вероятностью $\eta = \frac{1}{N+1}$.

В частности, для получения верхней оценки с надёжностью 95% достаточно взять $N = 20$ разбиений.

Утверждение 2. Если разбиения осуществлялись случайно, независимо и равновероятно, то с вероятностью $\eta = \frac{2t}{N+1}$ значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы доверительного интервала $\left[Q^{(t)}, Q^{(N-t+1)}\right]$.

Следствие 2. Значение случайной величины $Q(a(X^m), X^k)$ не выходит за границы вариационного ряда $\left[Q^{(1)}, Q^{(N)}\right]$ с вероятностью $\eta = \frac{2}{N+1}$.

В частности, для получения двусторонней оценки с надёжностью 95% достаточно взять $N = 40$ разбиений.

Параметрические оценки доверительного интервала основаны на априорном предположении о виде распределения случайной величины $Q(a(X^m), X^k)$. Если априорные предположения не выполняются, доверительный интервал может оказаться сильно смещённым. В частности, если предположения о нормальности распределения не выполнены, то нельзя пользоваться стандартным «правилом двух сигм» или «трёх сигм». Джон Лангфорд в своей диссертации (2002) указывает на распространённую ошибку, когда правило двух сигм применяется к функционалу частоты ошибок, имеющему на самом деле биномиальное распределение. Однако биномиальным распределением в общем случае тоже пользоваться нельзя, поскольку в результате обучения по случайным подвыборкам X^m вероятность ошибки алгоритма $a(X^m)$ оказывается случайной величиной.

Следовательно, случайная величина $Q(a(X^m), X^k)$ описывается не биномиальным распределением, а (неизвестной) смесью биномиальных распределений. Аппроксимация смеси биномиальным распределением может приводить к ошибочному сужению доверительного интервала. Приведённые выше непараметрические оценки лишены этого недостатка.

2.8.5 Стратификация

Стратификация выборки — это способ уменьшить разброс (дисперсию) оценок скользящего контроля, в результате чего получаются более узкие доверительные интервалы и более точные (tight) верхние оценки.

Стратификация заключается в том, чтобы заранее поделить выборку на части (страты), и при разбиении на обучение длины m и контроль длины k гарантировать, что каждая страта будет поделена между обучением и контролем в той же пропорции $m:k$.
Стратификация классов в задачах классификации означает, что каждый класс делится между обучением и контролем в пропорции $m:k$.
Стратификация по вещественному признаку. Объекты выборки сортируются согласно некоторому критерию, например, по возрастанию одного из признаков. Затем выборка разбивается на k последовательных страт одинаковой (с точностью до 1) длины. При формировании контрольных выборок из каждой страты выбирается по одному объекту, либо с заданным порядковым номером внутри страты, либо случайным образом.

2.8.6 Полный скользящий контроль (complete CV)

Оценка скользящего контроля строится по всем $N = C_L^k$ разбиениям. В зависимости от k (длины обучающей выборки) различают:

▪ Частный случай при $k = 1$ — контроль по отдельным объектам (leave-one-out CV);
Было показано (Li,1987), что контроль по отдельным объектам является асимптотически оптимальным при некоторых условиях, то есть:

$$\frac{L_n(\widehat{m})}{\inf\limits_{m \in M_n} L_n(m)} \to 1$$ по вероятности,

где:

1. M_n - класс сравниваемых моделей;
2. $L_n(m)$ - среднеквадратичная ошибка при выборе m-ой модели;
3. $\widehat{m} = \arg \min\limits_{m \in M_n} CV(m)$.

▪ Общий случай при $k>2$. Здесь число разбиений $N = C_L^k$ становится слишком большим даже при сравнительно малых значениях k, что затрудняет практическое применение данного метода. Для этого случая полный скользящий контроль используется либо в теоретических исследованиях (Воронцов,2004), либо в тех редких ситуациях, когда для него удаётся вывести эффективную вычислительную формулу. Например, такая формула известна для метода k ближайших соседей [1], что позволяет эффективно выбирать параметр k. На практике чаще применяются другие разновидности скользящего контроля.

2.8.7 Случайные разбиения

Разбиения $n = 1,\ldots,N$ выбираются случайно, независимо и равновероятно из множества всех C_L^k разбиений. Именно для этого случая справедливы приведённые выше оценки доверительных интервалов. На практике эти оценки, как правило, без изменений переносится и на другие способы разбиения выборки.

2.8.8 Контроль на отложенных данных (hold-out CV)

Оценка скользящего контроля строится по одному случайному разбиению, $N = 1$.
Этот способ имеет существенные недостатки:

1. Приходится слишком много объектов оставлять в контрольной подвыборке. Уменьшение длины, обучающей подвыборки приводит к смещённой (пессимистически завышенной) оценке вероятности ошибки.

2. Оценка существенно зависит от разбиения, тогда как желательно, чтобы она характеризовала только алгоритм обучения.
3. Оценка имеет высокую дисперсию, которая может быть уменьшена путём усреднения по разбиениям.

Следует различать скользящий контроль по отложенным данным и контроль по тестовой выборке. Если во втором случае оценивается вероятность ошибки для классификатора, построенного по обучающей подвыборке, то в первом случае - для классификатора, построенного по полной выборке (то есть доля ошибок вычисляется не для того классификатора, который выдается в качестве результата решения задачи).

2.8.9 Контроль по отдельным объектам (leave-one-out CV)

Является частным случаем полного скользящего контроля при $k = 1$, соответственно, $N = L$. Это, пожалуй, самый распространённый вариант скользящего контроля.
Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.
Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится L раз.
Некоторые методы обучения позволяют достаточно быстро перенастраивать внутренние параметры алгоритма при замене одного обучающего объекта другим. В этих случаях вычисление LOO удаётся заметно ускорить.

2.8.10 Контроль по q блокам (q-fold CV)

Выборка случайным образом разбивается на q непересекающихся блоков одинаковой (или почти одинаковой) длины $k_1,\ldots,k_q$:
$X^L = X_1^{k_1} \cup \ldots \cup X_q^{k_q}$,
$k_1 + \ldots + k_q = L$. Каждый блок по очереди становится контрольной подвыборкой, при этом обучение производится по остальным $q-1$ блокам. Критерий определяется как средняя ошибка на контрольной подвыборке:

$$CV(\mu, X^L) = \frac{1}{q} \sum_{n=1}^{q} Q\left(\mu\left(X^L \setminus X_n^{k_n}\right), X_n^{k_n}\right).$$

Это компромисс между LOO, hold-out и случайными разбиениями. С одной стороны, обучение производится только q раз вместо L. С другой стороны, длина обучающих подвыборок, равная $L\frac{q-1}{q}$ с точностью до округления, не сильно отличается от длины полной выборки L. Обычно выборку разбивают случайным образом на 10 или 20 блоков.

2.8.11 Контроль по r×q блокам (r×q-fold CV)

Контроль по *q* блокам (*q*-fold CV) повторяется *r* раз. Каждый раз выборка случайным образом разбивается на *q* непересекающихся блоков. Этот способ наследует все преимущества *q*-fold CV, при этом появляется дополнительная возможность увеличивать число разбиений.
Данный вариант скользящего контроля, со стратификацией классов, является стандартной методикой тестирования и сравнения алгоритмов классификации. В частности, он применяется в системах WEKA и «Полигон алгоритмов».

2.8.12 Недостатки скользящего контроля

1. Задачу обучения приходится решать *N* раз, что сопряжено со значительными вычислительными затратами.

2. Оценка скользящего контроля предполагает, что алгоритм обучения μ уже задан. Она ничего не говорит о том, какими свойствами должны обладать «хорошие» алгоритмы обучения, и как их строить. Такого рода подсказки дают, например, теоретические оценки обобщающей способности.
3. Попытка использовать скользящий контроль для обучения, в роли оптимизируемого критерия, приводит к тому, что он утрачивает свойство несмещённости, и снова возникает риск переобучения.
4. Скользящий контроль дает несмещенную точечную, но не интервальную оценку риска. В настоящее время не существует методов построения на основе скользящего контроля точных доверительных интервалов для риска, то есть математического ожидания потерь (в частности, вероятности ошибочной классификации).

3 Matplotlib

Библиотека matplotlib - это бибилиотека двумерной графики для языка программирования python с помощью которой можно создавать высококачественные рисунки различных форматов. Matplotlib представляет собой модуль-пакет для python.
Matplotlib состоит из множества модулей. Модули наполнены различными классами и функциями, которые иерархически связаны между собой.
Так как matplotlib организована иерархически, а наиболее простыми для человека являются самые высокоуровневые функции, то описание matplotlib следует начать с самого высокоуровневого модуля matplotlib.pyplot. Так, чтобы нарисовать гистограмму с помощью этого модуля, нужно вызывать всего одну команду: plt.hist().
Пользователю не нужно думать, как именно библиотека нарисовала эту диаграмму. Если бы мы рисовали гистрограмму самостоятельно, то заметили бы, что она состоит из повоторяющихся по форме фигур - прямоугольников. А чтобы нарисовать прямоугольник, нужно знать хотя бы координату одного угла и ширину/длину. Рисовали же бы мы прямоугольник линиями, соединяя угловые точки прямоугольника. Этот пример отображает иерархичность рисунков, когда итоговая диаграмма (высокий уровень) состоит из простых геометрических фигур (более низкий, средний уровень), созданных несколькими универсальными методами рисования (низкий уровень). Если бы каждый рисунок нужно было бы создавать вот так, с нуля, это было бы очень долго и утомительно.
Matplotlib.pyplot является набором команд и функций, которые делают синтаксис графических matploltib команд похожим на команды, используемые в среде MATLAB(c). Изначально matplotlib планировался как свободная альтернатива MATLAB(c), где в одной среде имелись бы средства как для рисования, так и для численного анализа. Именно так в Matplotlib и появился pylab, который объединяет модули pyplot и numpy в одно пространство имён.
В тоже время для более серьёзных задач (внедрение matplotlib в пользовательскую GUI) требуется больше контроля над процессом и больше гибкости, чем могут предоставить эти два модуля. Необходим доступ к более низкоуровневым возможностям библиотеки, которая реализована в объектно-ориентированном стиле. ООС заметно сложнее для новичков и требует знаний о работе конкретных классов и их методах, но предоставляет самые большие возможности по взаимодействию с библиотекой matplotlib.
Иерархическая структура рисунка в matplotlib
Главной единицей (объектом самого высокого уровня) при работе с matplotlib является рисунок (Figure). Любой рисунок в matplotlib имеет вложенную структуру.

- **Рисунок (Figure)**

Рисунок является объектом самого верхнего уровня, на котором располагаются одна или несколько областей рисования (Axes), элементы рисунка Artisits (заголовки, легенда и т.д.) и основа-холст (Canvas). На рисунке может быть несколько областей рисования Axes, но данная область рисования Axes может принадлежать только одному рисунку Figure.

- **Область рисования (Axes)**

Область рисования является объектом среднего уровня, который во является, наверное, главным объектом работы с графикой matplotlib в объектно-ориентированом стиле. Это то, что ассоциируется со словом "plot", это часть изображения с пространством данных. Каждая область рисования Axes содержит две (или три в случае трёхмерных данных) координатных оси (Axis объектов), которые упорядочивают отображаение данных.

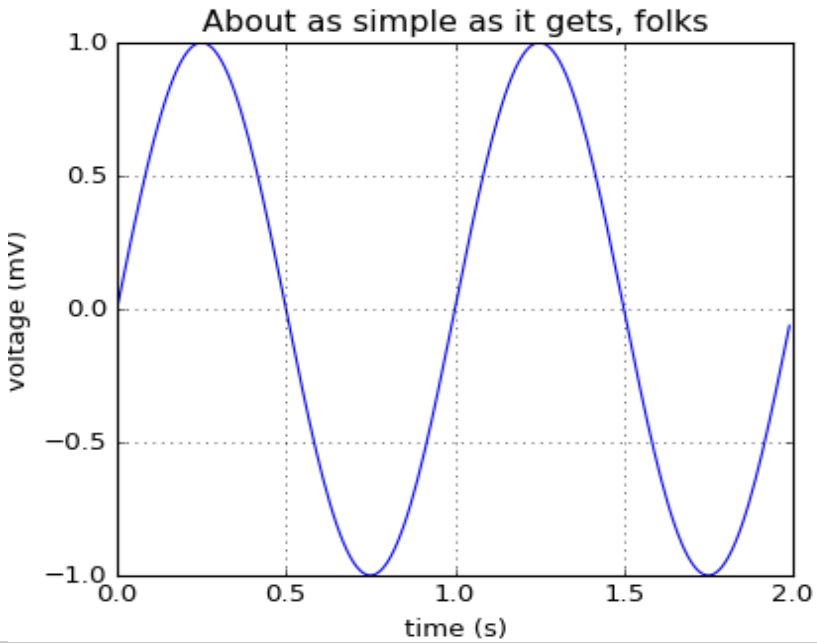
- **Координатная ось (Axis)**

Координатная ось являются объектом среднего уровня, которые определяют область изменения данных, на них наносятся деления ticks и подписи к делениям ticklabels. Расположение делений определяется объектом Locator, а подписи делений обрабатывает объект Formatter. Конфигурация координатных осей заключается в комбинировании различных свойств объектов Locator и Formatter.

- **Элементы рисунка (Artists)**

Элементы рисунка Artists является как бы красной линией для всех иерархических уровней. Практически всё, что отображается на рисунке является элементом рисунка (Artist), даже объекты Figure, Axes и Axis. Элементы рисунка Artists включают в себя такие простые объекты как текст (Text), плоская линия (Line2D), фигура (Patch) и другие. Когда происходит отображение рисунка (figure rendering), все элементы рисунка Artists наносятся на основу-холст (Canvas). Большая часть из них связывается с областью рисования Axes. Также элемент рисунка не может совместно использоваться несколькими областями Axes или быть перемещён с одной на другую.
Для того, чтобы убедиться в лёгкости работы с библиотекой достаточно взглянуть на несколько примеров.

Построение изображения синусоиды

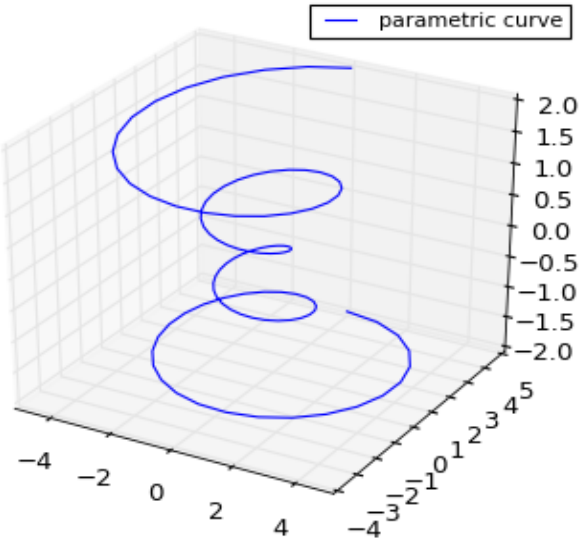



```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s = np.sin(2*np.pi*t)
plt.plot(t, s)

plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('About as simple as it gets, folks')
plt.grid(True)
plt.savefig("test.png")
plt.show()
```

Построение изображения параметрической кривой

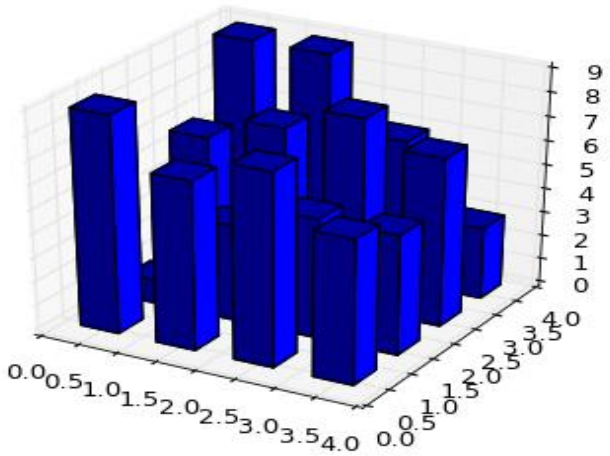



```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure()
ax = fig.gca(projection='3d')
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)
ax.plot(x, y, z, label='parametric curve')
ax.legend()
plt.show()
```

Построение трехмерной гистограммы




```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x, y = np.random.rand(2, 100) * 4
hist, xedges, yedges = np.histogram2d(x, y, bins=4)

elements = (len(xedges) - 1) * (len(yedges) - 1)
xpos, ypos = np.meshgrid(xedges[:-1] + 0.25, yedges[:-1] + 0.25)

xpos = xpos.flatten()
ypos = ypos.flatten()
zpos = np.zeros(elements)
dx = 0.5 * np.ones_like(zpos)
dy = dx.copy()
dz = hist.flatten()

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color='b', zsort='average')

plt.show()
```

4 Сравнение некоторых моделей на наборе данных «Ирисы Фишера»

In [1]:
```
import pandas as pd
import numpy as np
from pandas.tools.plotting import parallel_coordinates, scatter_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.grid_search import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import Normalizer
from sklearn.cross_validation import StratifiedKFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt

%matplotlib inline
```

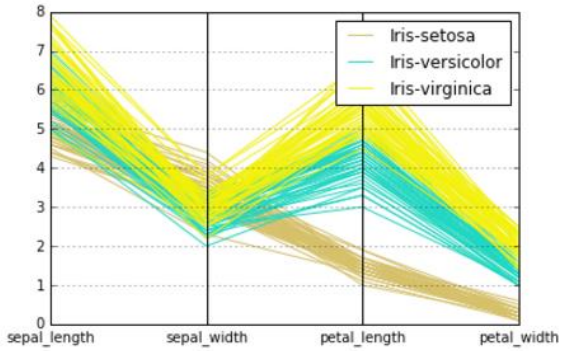
In [2]:
```
dataset_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
```

In [3]:
```
data = pd.read_csv(dataset_url, names=['sepal_length', 'sepal_width', 'petal_length',
                                       'petal_width', 'name'])
```

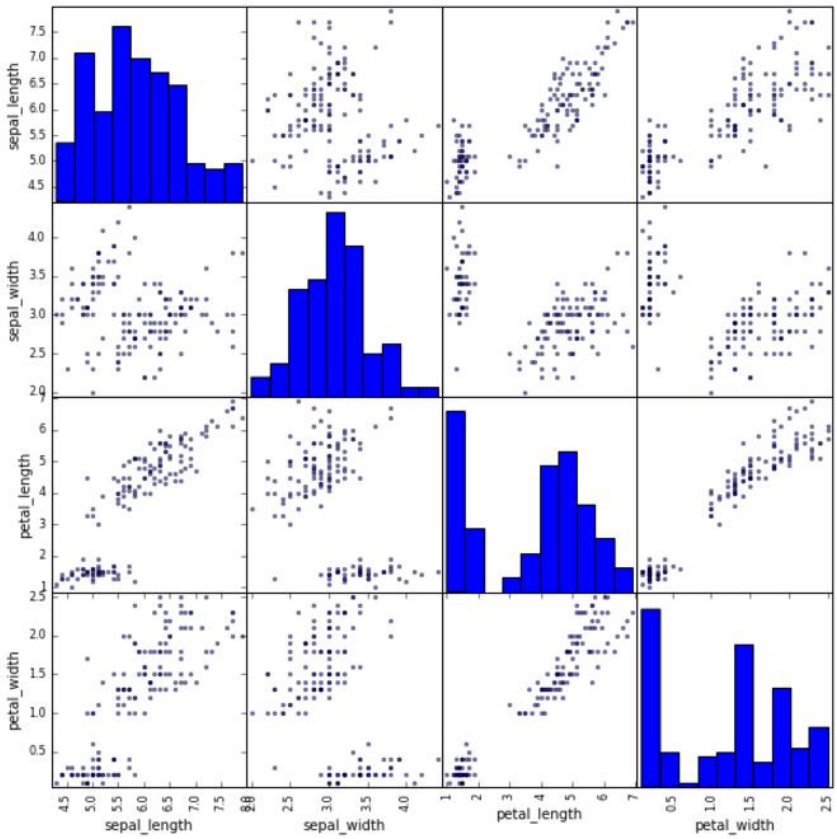
Визуализация набора данных

In [4]:
```
parallel_coordinates(data, 'name')
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x55f0684c50>
```



In [5]:
```
_ = scatter_matrix(data[['sepal_length', 'sepal_width',\
                         'petal_length', 'petal_width']], grid=True, figsize=(10, 10))
plt.show()
```



Кодирование категориальной целевой переменной

In [4]:
```
data['name_encoded'] = LabelEncoder().fit_transform(data['name'])

data.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	name	name_encoded
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0

In [5]:
```
X, y = data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].as_matrix(),\
       data['name_encoded'].as_matrix().ravel()
```

Грид-поиск оптимальной комбинации параметров для логистической регрессии

In [6]:
```
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': np.linspace(1e-4, 1)
}

scores = GridSearchCV(LogisticRegression(), param_grid, scoring='accuracy').fit(X, y)

scores.best_score_, scores.best_params_
```

Out[6]:
```
(0.94666666666666666, {'C': 0.55106530612244897, 'penalty': 'l2'})
```

Выполнение нормализации вещественнозначных признаков

In [7]:
```
X_norm = Normalizer().fit_transform(X)
```

Грид-поиск оптимальной комбинации параметров для логистической регрессии при нормализованной матрице предикторов

In [8]:
```
scores = GridSearchCV(LogisticRegression(), param_grid, scoring='accuracy').fit(X_norm, y)

scores.best_score_, scores.best_params_
```

Out[8]:
```
(0.9733333333333338, {'C': 0.0001, 'penalty': 'l2'})
```

Грид-поиск оптимальной комбинации параметров для дерева решений

In [9]:
```
param_grid = {
    'criterion': ['gini'],
    'splitter': ['best'],
    'max_features': ['sqrt'],
    'max_depth': [2, 5],
    'min_samples_split': [5, 7],
    'min_samples_leaf': [1, 2],
    'min_weight_fraction_leaf': [0, 0.05, 0.01],
    'max_leaf_nodes': [10, 15],
}

scores = GridSearchCV(DecisionTreeClassifier(), param_grid, scoring='accuracy', n_jobs=-1,
                      cv=StratifiedKFold(y)).fit(X, y)

scores.best_score_, scores.best_params_
```

Out[9]:
```
(0.9733333333333338,
 {'criterion': 'gini',
  'max_depth': 5,
  'max_features': 'sqrt',
  'max_leaf_nodes': 15,
  'min_samples_leaf': 1,
  'min_samples_split': 5,
  'min_weight_fraction_leaf': 0,
  'splitter': 'best'})
```

Грид-поиск оптимальной комбинации параметров для метода ближайших соседей

In [10]:
```
param_grid = {
    'n_neighbors': np.arange(1, 30),
    'weights': ['uniform', 'distance'],
    'p': [1, 2, 3]
}

scores = GridSearchCV(KNeighborsClassifier(), param_grid, scoring='accuracy', n_jobs=-1,
                      cv=StratifiedKFold(y)).fit(X, y)

scores.best_score_, scores.best_params_
```

Out[10]:
```
(0.9866666666666669, {'n_neighbors': 5, 'p': 2, 'weights': 'uniform'})
```

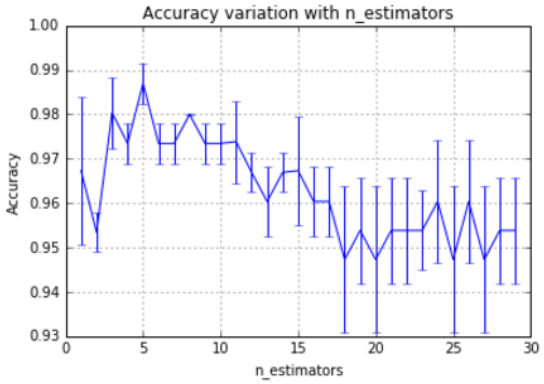
Графическое отображение зависимотси между количеством соседей и точностью классификации

In [11]:
```
acc_mean = []
acc_error = []

for n_neighbors in np.arange(1, 30):
    score = cross_val_score(KNeighborsClassifier(n_neighbors=5, weights='uniform', p=2),
                            X, y=y, scoring='accuracy', cv=StratifiedKFold(y))
    acc_mean.append(score.mean())
    acc_error.append(score.std() / 2)

plt.ylabel("Accuracy")
plt.xlabel("n_estimators")
plt.grid(which='both')
plt.title("Accuracy variation with n_estimators")
plt.errorbar(np.arange(1, 30), acc_mean, yerr=acc_error)
```

Out[11]:
```
<Container object of 3 artists>
```



На графике можно видеть, что после некоторого порога (5), точность классификации начинает убывать, что подтверждает действие гипотезы компактности

Заключение

В ходе практики были усвоены знания, приобретенные навыки теоретического обучения были закреплены на реальном производстве, знакомство со структурой информационно-вычислительного центра предприятия, освоение и получение навыков самостоятельной работы, приобретение знаний, таких как работа в системах семейства Unix.
Кроме вышеперечисленного, перед началом прохождения практики на предприятии состоялся инструктаж по технике безопасности.
В ходе прохождения практики были выполнены следующие задачи:

- изучение математических моделей и алгоритмов машинного обучения;
- изучение инструментов предобработки выборочных данных;
- изучение средств визуализации данных;
- настройка рабочего окружения;
- проектирование программного обеспечения;
- разработка программного обеспечения;

Список использованных материалов

1. http://jupyter.org/
2. http://scikit-learn.org/
3. http://pandas.pydata.org/
4. http://matplotlib.org/
5. Червоненкис А.Я. "Компьютерный анализ данных"
6. http://www.machinelearning.ru/
7. https://www.coursera.org/specializations/machine-learning-data-analysis
8. https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie/home/welcome