Интеллектуальный анализ данных и объектно-ориентированное программирование

К. В. Воронцов
(vokov@forecsys.ru, http://www.ccas.ru/voron)

Кафедра «Интеллектуальные системы» ФУПМ МФТИ,
ЗАО «Фóрексис»,
Вычислительный Центр РАН,
Школа анализа данных Яндекс

Зимняя компьютерная школа МФТИ,
Долгопрудный, 12 января 2011

Задача обучения по прецедентам (определения и обозначения)

X — объекты; Y — ответы (классы);
$y^*: X \to Y$ — неизвестная зависимость.

Дано: $x_i = (x_i^1, \ldots, x_i^n)$ — *обучающие объекты* с известными ответами $y_i = y^*(x)$.

$$\begin{pmatrix} x_1^1 & \ldots & x_1^n \\ \ldots & \ldots & \ldots \\ x_\ell^1 & \ldots & x_\ell^n \end{pmatrix} \begin{matrix} \to \\ \to \\ \to \end{matrix} \begin{pmatrix} y_1 \\ \ldots \\ y_\ell \end{pmatrix}$$

Найти: алгоритм $a: X \to Y$, предсказывающий правильные ответы на новых объектах $z_i = (z_i^1, \ldots, z_i^n)$.

$$\begin{pmatrix} z_1^1 & \ldots & z_1^n \\ \ldots & \ldots & \ldots \\ z_k^1 & \ldots & z_k^n \end{pmatrix} \begin{matrix} \to \\ \to \\ \to \end{matrix} \begin{pmatrix} ? \\ \ldots \\ ? \end{pmatrix}$$

Примеры прикладных задач обучения по прецедентам

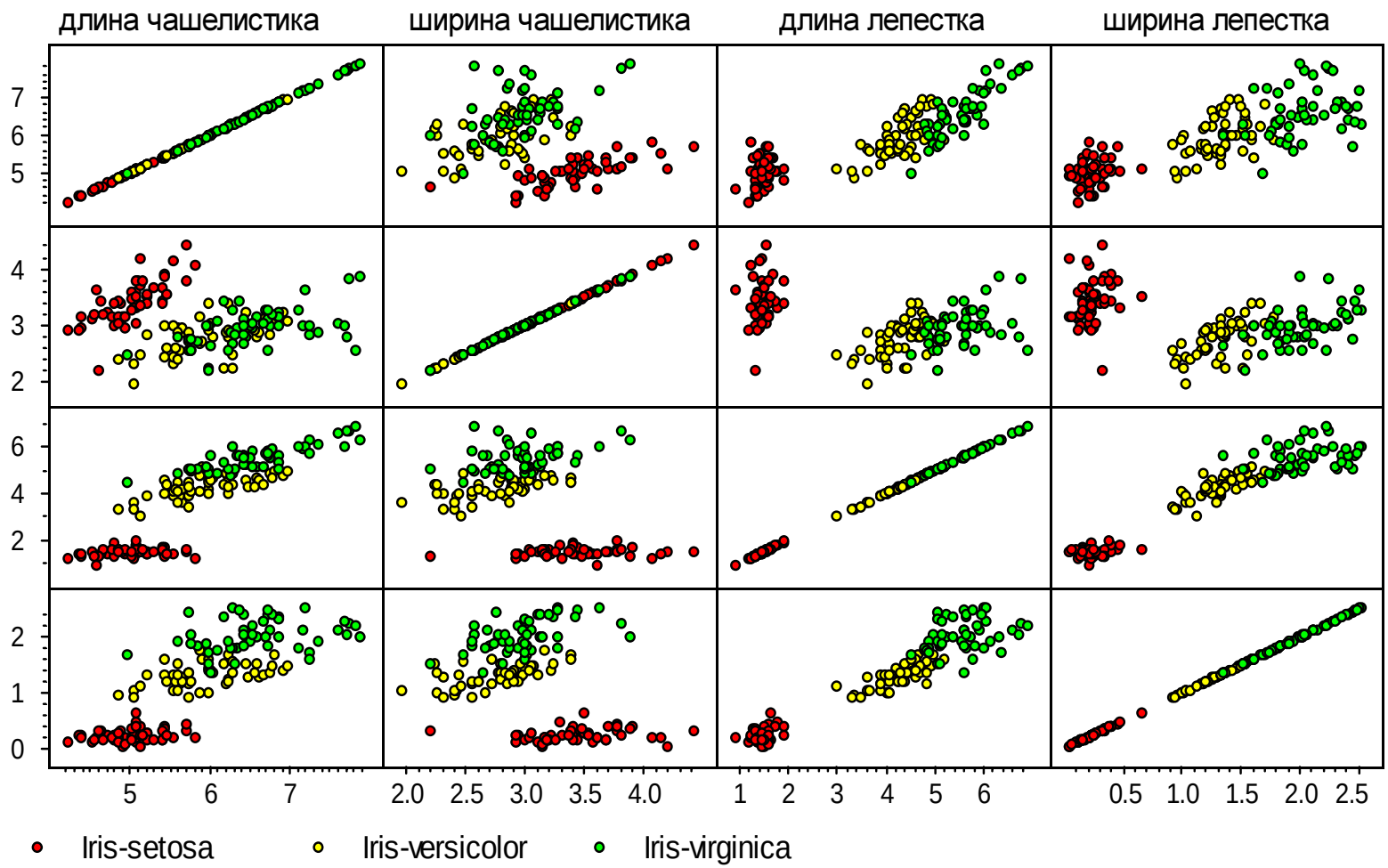
- Распознавание, классификация, принятие решений ($|Y| < \infty$):
  - x — пациент; y — долгосрочный исход операции;
  - x — заёмщик; y — кредит выдать / не выдать;
  - x — курсы акций; y — купить / продать.
  - x — абонент; y — уйдёт / не уйдёт к другому оператору;
  - x — фотопортрет; y — идентификатор личности;
  - x — фрагмент ДНК; y — функция: промотор / ген;
  - x — фрагмент белка; y — тип вторичной структуры;
  - x — текстовое сообщение; y — спам / не спам;
- Регрессия и прогнозирование ($Y = \mathbb{R}$ или $\mathbb{R}^m$):
  - x — структура хим. соединения; y — его свойство;
  - x — параметры технолог. процесса; y — свойство продукции;
  - x — история продаж; y — прогноз потребительского спроса;
  - x — данные о недвижимости; y — продажная стоимость;
  - x — пара $\langle$клиент,товар$\rangle$; y — рейтинг товара.

Некоторые методы классификации

- Нейронная сеть;
- Метод опорных векторов;
- Бустинг (TreeNet, MatrixNet);
- Метод ближайших соседей;
- Решающее дерево;
- Логистическая регрессия;
- Взвешенное голосование (комитет большинства);
- Решающий список (комитет старшинства);
- Персептрон;
- Метод потенциальных функций;
- Наивный байесовский классификатор;
- ... ...

Пример: задача классификации цветков ириса [Фишер, 1936]

$n = 4$ признака, $|Y| = 3$ класса, длина выборки $\ell = 150$.



Метод ближайшего соседа

Гипотеза компактности:

Схожие объекты, как правило, лежат в одном классе.

Расстояние между объектами $x_i = (x_i^1, \ldots, x_i^n)$ и $z = (z^1, \ldots, z^n)$:

$$\rho(z, x_i) = \left(\sum_{j=1}^{n} \left| z^j - x_i^j \right|^p \right)^{\frac{1}{p}}$$

Упорядочим выборку по расстояниям до объекта z:

$$\rho(z, x^{(1)}) \leqslant \rho(z, x^{(2)}) \leqslant \cdots \leqslant \rho(z, x^{(\ell)}),$$

где $x^{(i)}$ — i-й сосед объекта x.

Алгоритм классификации: $a(z) = y(x^{(i)})$.

Проблемы и обобщения метода ближайшего соседа

- **Проблема:** объекты-выбросы
  - брать не одного, а нескольких соседей
  - оптимизировать число соседей
  - отбирать эталонные объекты из обучающей выборки
- **Проблема:** неинформативные признаки
  - отбирать полезные признаки по обучающей выборке
- **Проблема:** функция расстояния
  - настраивать веса признаков по обучающей выборке

Пожелания к программной реализации

Хотелось бы по-разному дорабатывать этот метод,
затем тестировать и сравнивать разные реализации,
как будто это один и тот же метод.

Объектно-ориентированное программирование

Три мудрёных слова про ООП

- **Инкапсуляция:**
  внутренние данные объекта и его функции описаны в одном месте и доступны по имени объекта.
- **Наследование:**
  можно заимствовать полезные данные и функции у другого объекта (родителя).
- **Полиморфизм:**
  разные объекты могут иметь разные реализации одной и той же функции; можно обращаться к функции, даже «не зная», какого типа объект её выполнит.

CLearner — основной объект

CLearner — это абстрактный обучаемый алгоритм.
Все обучаемые алгоритмы должны иметь такие же функции.

```
// Метод обучения по прецедентам (абстрактный класс)
class CLearner {
public:
    CLearner(void) {}
    virtual ~CLearner(void) {}
    // Алгоритм обучения
    virtual int Train (
        CMatrix &Features, // матрица Объекты * Признаки
        CMatrix &Target    // вектор-столбец Объекты * (ЦелевойПризнак)
    ) = 0;
    // Функция применения
    virtual int Test (
        CMatrix &Features, // матрица Объекты * Признаки
        CMatrix &Result    // вектор-столбец Объекты * (ЦелевойПризнак)
    ) = 0;
};
```

CMatrix — числовая матрица

```
// Двумерная числовая матрица
class CMatrix {
public:
    // Создание матрицы заданного размера: rows строк, cols столбцов
    CMatrix (int rows, int cols=1);
    CMatrix ();   // Создание пустой матрицы
    virtual ~CMatrix ();   // Уничтожение матрицы
    CMatrix& operator= (CMatrix& right);   // Копирование матрицы
    int Rows();   // Число строк в матрице
    int Cols();   // Число столбцов в матрице
    void Zeroize ();   // Обнуление всех элементов матрицы
    // Изменение размера матрицы, возможно, с сохранением данных
    void SetSize (int rows, int cols=1, bool preserve_data=false);
    void Load (const char* filename);   // Загрузка матрицы из файла
    void Save (const char* filename, const char* mode="w");   // Запись в файл
    // Доступ к елементу матрицы [row,col]
    double& Elem (int row, int col=0);
    // Доступ к строке row, чтобы запись M[r][c] была эквивалентна M.Elem(r,c)
    double* operator[] (int row);
private:
    int n_rows;   // число строк
    int n_cols;   // число столбцов
    double *data;   // все строки матрицы одна за другой
};
```

Как это используется

```
// выполнение контеста на Зимней Компьютерной Школе МФТИ - 2011
void contest_MIPT_2011 () {
    // загрузка матриц исходных данных
    CMatrix data, test_data;
    data.Load ("../contest/german-MIPT-contest-train.txt");
    test_data.Load ("../contest/german-MIPT-contest-test-noy.txt");
    // переписывание обучающих данных
    int L = data.Rows();  // длина обучающей выборки
    int n = data.Cols()-1;  // число признаков
    CMatrix train_data (L,n);
    CMatrix train_target (L);
    for (int i=0; i<L; i++) {
        // классификации содержатся в последней колонке
        train_target[i][0] = data[i][data.Cols()-1];
        for (int j=0; j<n; j++)
            train_data[i][j] = data[i][j];
    }
    // создание конкретного метода обучения по прецедентам
    CLearner_kNN learner;
    learner.NearestNeighbors = 1;
    // обучение классификатора на обучающей выборке
    learner.Train (train_data, train_target);
    // классификация тестовой выборки
    CMatrix test_result (test_data.Rows());
    learner.Test (test_data, test_result);
    // запись результатов классификации в файл
    test_result.Save ("test_result.txt");
    check_result (test_result);
}
```

Как определяется собственный метод

```
// Классификатор по k ближайшим соседям
class CLearner_kNN : public CClassificationLearner {
public:
    int NearestNeighbors;   // число ближайших соседей
    double Power;   // показатель степени в метрике
    CLearner_kNN();
    virtual ~CLearner_kNN();
    // Метод обучения
    virtual int Train (CMatrix &Features, CMatrix &Target);
    // Метод применения
    virtual int Test (CMatrix &Features, CMatrix &Result);
protected:
    CMatrix train_features, train_target;   // обучающая выборка
    CMatrix feature_weight;   // вектор весов признаков
    // расстояние между двумя объектами
    virtual double distance (double* row1, double* row2);
};

double CLearner_kNN::distance (double* row1, double* row2) {
    double sum = 0.0;
    for (int j=0; j<train_features.Cols(); j++)
        sum += feature_weight[j][0] * pow (fabs(row1[j]-row2[j]), Power);
    return pow (sum, 1.0/Power);
}
```

Что это за класс — CClassificationLearner

```
// Метод обучения классификации (абстрактный класс)
class CClassificationLearner : public CLearner {
public:
    CClassificationLearner(void) {}
    virtual ~CClassificationLearner(void) {}
protected:
    // вектор меток классов, classes.Rows() равно числу классов в выборке
    CMatrix classes;
    // сформировать вектор меток классов
    void get_classes (CMatrix &Target);
    // найти номер класса ctarget в массиве classes
    // (классы могут нумероваться не подряд!)
    int find_class (double ctarget);
};
```

Как реализуется «ленивое обучение»

```
// Метод обучения
int CLearner_kNN::Train (CMatrix &Features, CMatrix &Target) {
    // запомнить обучающую выборку
    train_features = Features;
    train_target = Target;
    // все признаки имеют одинаковый вес
    feature_weight.SetSize (Features.Cols());
    for (int j=0; j<feature_weight.Rows(); j++)
        feature_weight[j][0] = 1.0;
    // подсчёт числа классов в выборке путём сортировки вектора Target
    get_classes (Target);
    return 0;
}
```

Учёт произвольного числа соседей k

Итак, метод первого попавшегося соседа не очень хорош...
Среднее расстояние от объекта z до k его ближайших соседей класса y:

$$r(z,y) = \frac{1}{k}\sum_{i=1}^{k} \rho(z, x^{(i,y)}).$$

где $x^{(i,y)}$ — i-й сосед объекта z среди всех обучающих объектов класса y.

Относим объект к классу с наименьшим средним расстоянием:

$$a(z) = y : r(z,y) \leqslant r(z,\bar{y}), \quad \forall \bar{y} \neq y.$$

Появился параметр k — число соседей. Как его задавать?

Выбор числа соседей k — скользящий контроль

При каждом k посчитаем число ошибок классификации:

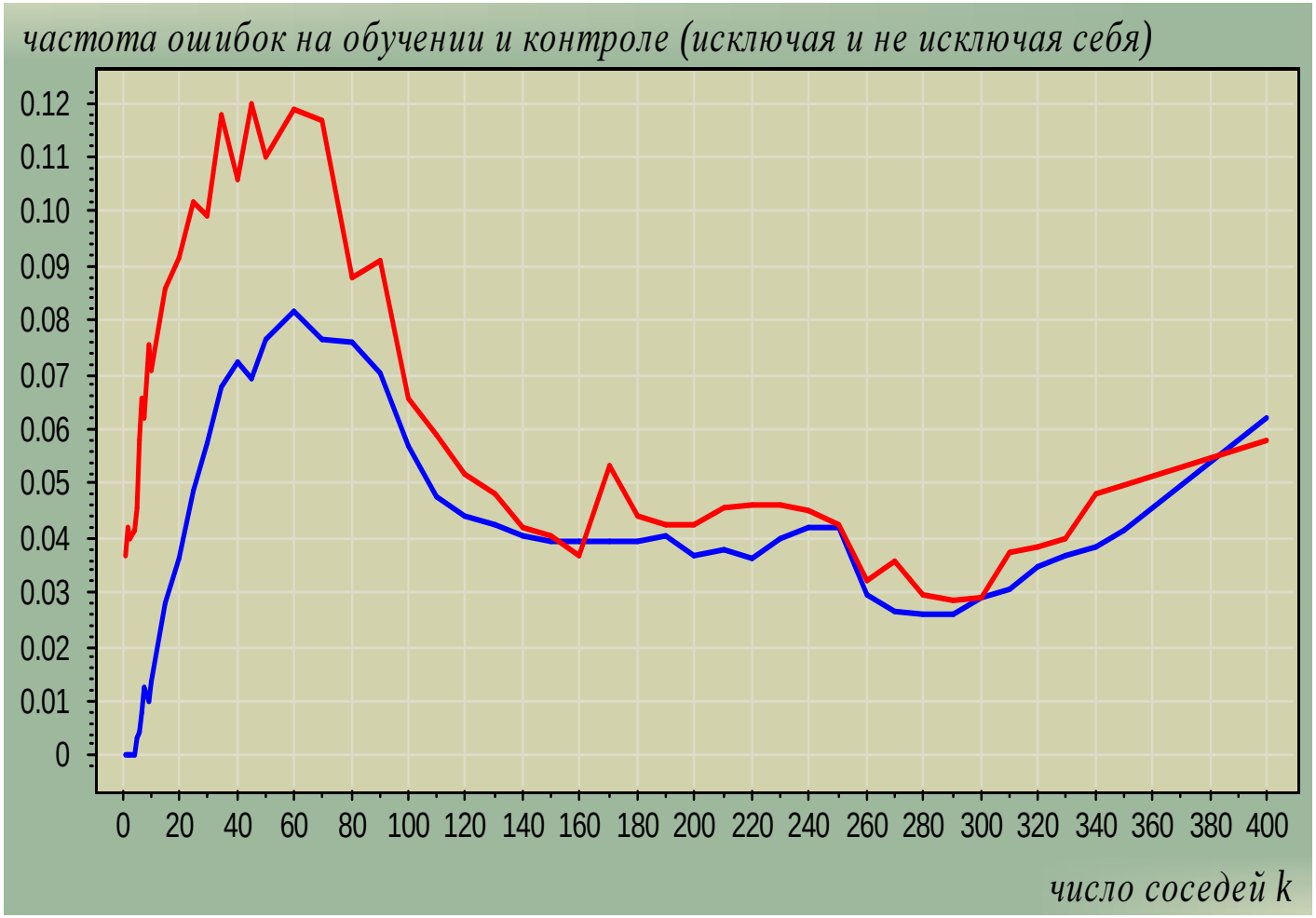
$$Q(k) = \sum_{i=1}^{\ell} \big[a(x_i) \neq y_i\big] \to \min_k .$$

Тонкость:
при вычислении $a(x_i)$ объект x_i не должен быть соседом самого себя. Иначе получим вырожденное решение $Q(1) = 0$.

Исправленное среднее расстояние от объекта x_i до класса y_i:

$$r(x_i, y_i) = \frac{1}{k} \sum_{i=2}^{k+1} \rho(z, x^{(i,y_i)}).$$

Пример зависимости $Q(k)$



(в реальных задачах минимум редко бывает при $k = 1$)

Оценка близости i-го объекта к своему классу

$r_i = r(x_i, y_i)$ — среднее расстояние до объектов своего класса;
$\bar{r}_i = r(x_i, \bar{y}_i)$ — среднее расстояние до объектов чужого класса;

Всё познаётся в сравнении!

$$d_i = \frac{\bar{r}_i - r_i}{\bar{r}_i + r_i} \approx \begin{cases} +1, & \text{объект близок к своим;} \\ 0, & \text{объект пограничный;} \\ -1, & \text{объект близок к чужим;} \end{cases}$$

Назовём d_i *благонадёжностью* объекта x_i.

Удаление шума из обучающей выборки

Отсортировать все объекты по возрастанию d_i;
Убрать ℓ_0 наименее благонадёжных объектов.

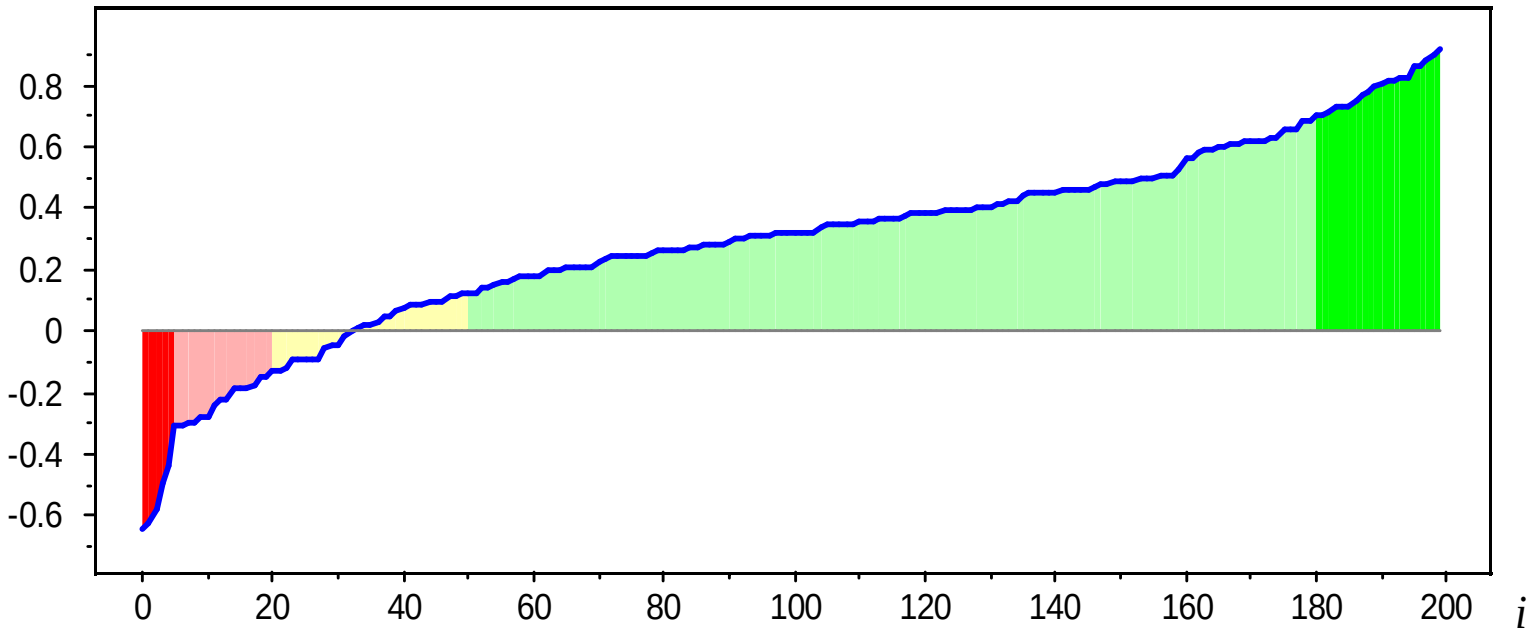
Увы, новый параметр ℓ_0 также придётся подбирать.

Благонадёжность выборки

Суммарная благонадёжность выборки характеризует то, насколько функция расстояния ρ подходит для данной задачи

$$D(\rho) = \sum_{i=1}^{\ell} d_i = \sum_{i=1}^{\ell} \frac{\bar{r}_i - r_i}{\bar{r}_i + r_i}$$

распределение объектов по благонадёжности di



Проблема отбора признаков

Суть проблемы:
обычно имеется много неинформативных признаков.
Чем их больше, тем выше шансы найти закономерность там, где её на самом деле нет (эффект *переобучения*).

Методы отбора признаков:

- перебор нескольких разумных вариантов вручную;
- полный перебор 2^n вариантов $J \subset \{1, \ldots, n\}$;
- жадное добавление (см. далее);
- случайный поиск;
- поиск в глубину;
- поиск в ширину;
- генетический алгоритм;
- ... ...

Жадное добавление признаков

1. А вдруг одного признака уже достаточно?
Расстояние по j-му признаку: $\rho_j(z, x_i) = \left|z^j - x_i^j\right|$.
Выберем наиболее благонадёжное расстояние: $D(\rho_j) \to \max\limits_j$.

2. Пусть уже есть расстояние ρ.
Попробуем добавить к нему ещё один признак j.

$$\rho_{jt}(z, x_i) = (1 - t) \cdot \rho(z, x_i) + t \cdot \rho_j(z, x_i).$$

Найдём $t \in [0, 1]$ и признак j, при которых благонадёжность $D(\rho_{jt})$ максимальна (два вложенных цикла перебора).

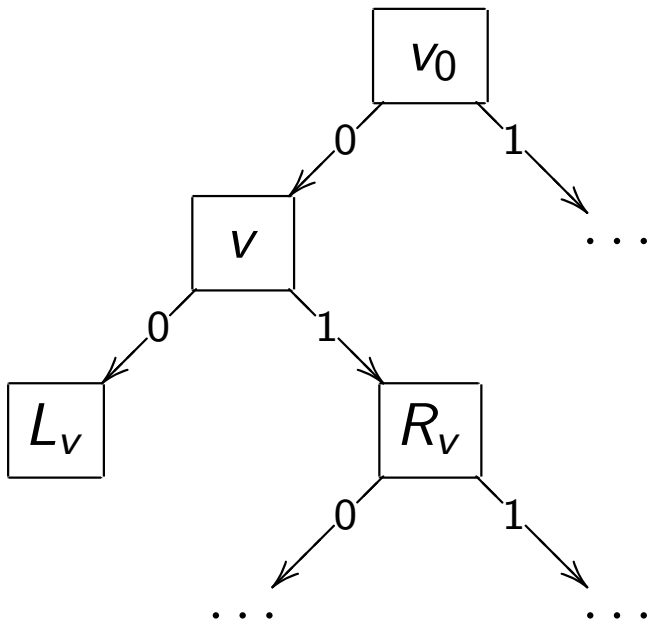
3. Будем добавлять признаки до тех пор,
пока благонадёжность $D(\rho_{jt})$ увеличивается.

Определение бинарного решающего дерева

Бинарное решающее дерево — алгоритм классификации $a(x)$, задающийся бинарным деревом:

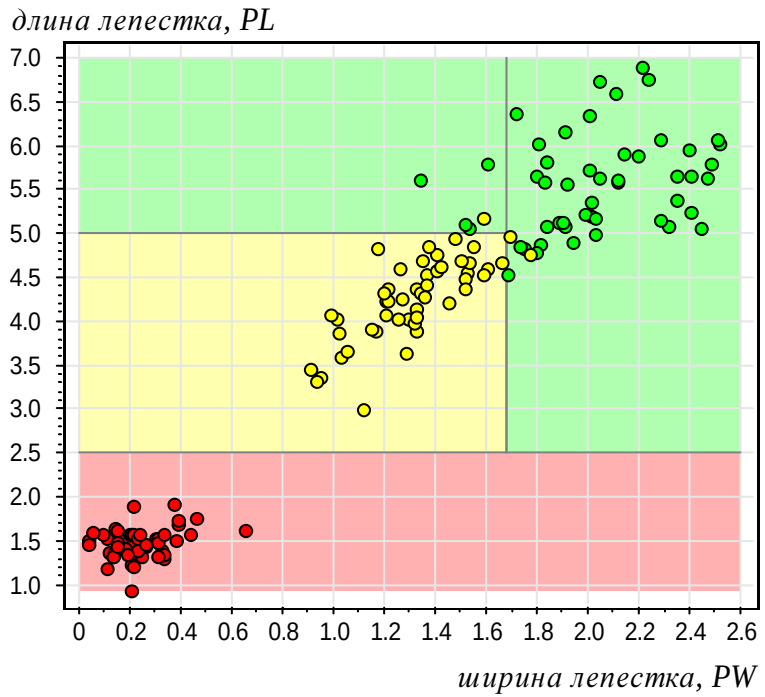
1) $\forall v \in V_{\text{внутр}} \rightarrow$ условие ветвления $b_v(x) \in \{0, 1\}$;
2) $\forall v \in V_{\text{лист}} \rightarrow$ имя класса $c_v \in Y$.

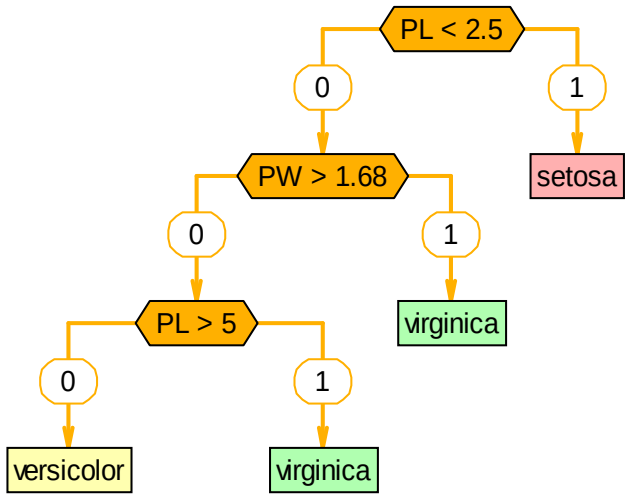
1: $v := v_0$;
2: **пока** $v \in V_{\text{внутр}}$
3: **если** $b_v(x) = 1$ **то**
4: переход вправо:
$v := R_v$;
5: **иначе**
6: переход влево:
$v := L_v$;
7: **вернуть** c_v.



Пример решающего дерева

Задача Фишера о классификации цветков ириса на 3 класса, в выборке по 50 объектов каждого класса, 4 признака.





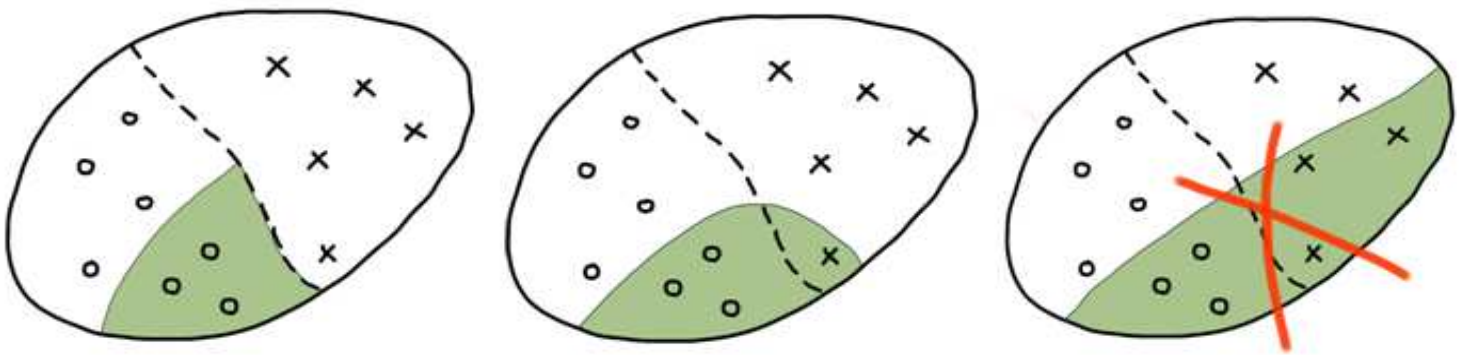
На графике: в осях двух самых информативных признаков (из 4) два класса разделились без ошибок, на третьем 3 ошибки.

Понятие закономерности

Логическая закономерность (правило, rule) класса y
— это функция $r(x)$, $r(x) \in \{0, 1\}$:

- $r(x)$ определяется простой понятной формулой, например, $r(x) = [x^j < \alpha]$;
- $r(x) = 1$ преимущественно на объектах класса y:

$$p(r, U) = \#\{x_i \in U\colon r(x_i)=1 \text{ и } y_i=y\} \to \max;$$
$$n(r, U) = \#\{x_i \in U\colon r(x_i)=1 \text{ и } y_i\neq y\} \to \min;$$



Критерии информативности

Два критерия $p(r,U) \to \max$, $n(r,U) \to \min$ — это не удобно!

Варианты критериев информативности:

- $I(r,U) = (N-n)p + (P-p)n \to \max$;
- $I(r,U) = (N-n)^2 + (P-p)^2 + p^2 + n^2 \to \max$;
- $I(r,U) = \sqrt{p} - \sqrt{n} \to \max$;
- $I(r,U) = \sqrt{\frac{p}{P}} - \sqrt{\frac{n}{N}} \to \max$;
- $I(r,U) = -\log \frac{C_{P+N}^{p+n}}{C_P^p C_N^n} \to \max$; // степень неслучайности;

где $P = \#\{x_i \in U : y_i = y\}$, $N = \#\{x_i \in U : y_i \neq y\}$

Вообще, критериев информативности придумано более 20...

Жадный алгоритм построения дерева ID3

1: **ПРОЦЕДУРА** LearnID3 (U);
2: **если** все объекты из U лежат в одном классе y **то**
3: **вернуть** новый лист v, $c_v := y$;
4: найти условие ветвления с максимальной информативностью:
$b := \arg\max_b I(b, U)$;
5: разбить выборку на две части $U = U_0 \cup U_1$ по условию b:
$U_0 := \{x \in U : b(x) = 0\}$;
$U_1 := \{x \in U : b(x) = 1\}$;
6: **если** $I(b, U) \leqslant I_0$ **то**
7: **вернуть** новый лист v, $c_v :=$ Большинство(U);
8: создать новую внутреннюю вершину v: $b_v := b$;
построить левое поддерево: $L_v :=$ LearnID3 (U_0);
построить правое поддерево: $R_v :=$ LearnID3 (U_1);
9: **вернуть** v;

Что посмотреть в Интернете?

- Регулярные соревнования по анализу данных:
  http://www.kaggle.com
- Репозиторий реальных задач UCI (Университет Ирвайн):
  archive.ics.uci.edu/ml
- Системы с алгоритмами машинного обучения:
  WEKA — www.cs.waikato.ac.nz/ml/weka
  RapidMiner — rapidminer.com
- Полигон алгоритмов классификации:
  Poligon.MachineLearning.ru
- Вики-ресурс по машинному обучению на русском языке:
  www.MachineLearning.ru
  я там: «Участник:Vokov»