**Метод k-ближайших соседей**

**Wiki**

Метод k ближайших соседей ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *k-nearest neighbor algorithm*) - метод автоматической классификации объектов. Основным принципом **метода ближайших соседей** является то, что объект присваивается тому классу, который является наиболее распространённым среди соседей данного элемента.

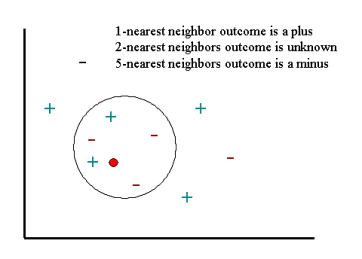
Соседи берутся исходя из множества объектов, классы которых уже известны, и, исходя из ключевого для данного метода значения k высчитывается, какой класс наиболее многочислен среди них.

**http://www.spc-consulting.ru/dms/machine%20learning/machinelearning/Overviews/KNearestNeighborsIntroductoryOverview%20.htm**

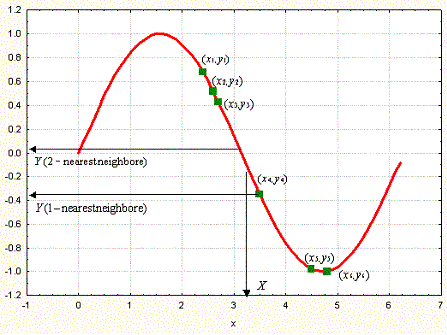
**Метод K - Ближайших соседей: Вводный Обзор**

**Классификация**

Чтобы наглядно описать принципы работы метода *K - Ближайших соседей,* рассмотрим задачу классификации новых объектов (точек запроса) среди некоторого количества уже известных примеров. Эта задача проиллюстрирована ниже; примеры (известные экземпляры) отмечены знаком "+" или "-", а точки запроса - красным кружочком. Наша цель заключается в оценке (классификации) отклика точек запроса с использованием специально выбранного числа их ближайших соседей. Другими словами, мы хотим узнать, как классифицировать точки запроса: как знак "+" или как знак "-".



Для начала рассмотрим результат работы процедуры анализа *К - БС* с использованием одного ближайшего соседа. Ясно, что в этом случае отклик точки запроса будет предсказан как знак плюс (т.к. ближайшая соседняя точка имеет знак плюс). Теперь увеличим число используемых ближайших соседей до двух. На этот раз процедура К - БС не сможет классифицировать отклик точки запроса по причине того, что вторая ближайшая точка имеет знак минус и оба знака равноценны (т.е. победа с одинаковым количеством голосов). На следующем шаге увеличим число используемых ближайших соседей до 5. Таким образом, будет определена целая окрестность точки запроса (на графике ее граница отмечена красной окружностью). Так как в области содержится 2 точки со знаком "+" и 3 точки со знаком "-" , алгоритм К - БС присвоит знак "-" отклику точки запроса.



**Регрессия**

В этом разделе мы обобщим принцип *К - БС* для задач регрессии. Регрессионные задачи связаны с предсказанием отклика зависимой переменной по данному набору независимых переменных. Для начала рассмотрим приведенную выше схему. Изображенный на ней набор точек (зеленые прямоугольники) получен по связи между независимой переменной *x* и зависимой переменной *y* (кривая красного цвета). Задан набор зеленых объектов (т.е набор примеров); мы используем метод *К - ближайших соседей* для предсказания выхода точки запроса *X* по данному набору примеров (зеленые прямокгольники).

Сначала рассмотрим в качестве примера метод *К - БС* с использованием одного ближайшего соседа. Мы ищем набор примеров (зеленые прямоугольники) и выделяем из их числа ближайший к точке запроса *X*. Для нашего случая пусть это будет *x*4. Выход *x*4 (т.е. *y*4), таким образом, принимается в качестве результата предсказания выхода *X* (т.е. *Y*). Следовательно, для 1 - ближайшего соседа можем записать:

*Y* = *y*4

http://www.spc-consulting.ru/dms/machine%20learning/Images/eq1.gif

Далее рассмотрим метод *2* - ближайших соседей. В этом случае мы выделяем уже 2 ближайшие к *X* точки. Пусть, например, они будут *y*3 и *y*4 соответсвенно. Вычислив среднее их выходов, записываем решение для Y в виде:

http://www.spc-consulting.ru/dms/machine%20learning/Images/eq1.gif

Описанные выше действия с легкостью переносятся на случай использования произвольного числа ближайших соседей. Подводя итог, отметим, что в методе *К - ближайших соседей* выход Y точки запроса X кладется равным среднему значению выходов К ближайших соседей точки запроса.

**Технические подробности**

Метод *STATISTICA* К *- Ближайщих соседей (К - БС)* существенно использует память. Модель определяется набором объектов, называемых примераим (или экземплярами), для которых известны выходы (т.е каждый экземпляр помечен). Каждый пример состоит из набора независимых значений, помеченных набором зависимых выходов. Независсимые и зависимые переменные могут быть как непрерывными, так и категориальными. Для непрерывных зависимых переменных задача рассматривается как регрессионная, иначе считается классификационной. Таким образом, метод *STATISTICA К - БС* применим для решения задач как регрессионного, так и классификационного типа.

Имея новый экземпляр зависимых значений, мы хотим оценить выход, используя примеры для алгоритма *К - БС*. *STATISTICA К - БС* разрешает этот вопрос, отыскивая *К* наболее близко расположенных к точке запроса примеров (поэтому метод и получил такое название). Предсказание в регрессионной задаче получается усреднением выходов *К* ближайших соседей, а решение классификационной задачи основано на принципе "принято большинством голосов".

Выбор значения параметра *К* - ключевое место в построении модели *К - БС*. Действительно, параметр *К* является одним из наиболее значимых факторов модели, влияющих на качество прогноза. Один из содержательных подходов к оценке необходимого числа ближайших соседей - воспринимать *К* как параметр сглаживания. Для любой задачи выбор малого значения параметра *К* приведет к сильному разбросу значений прогноза. Напротив, большое значение параметра *К* может повлечь сильную смещенность модели. Следовательно, по величине *К* должен быть достаточно большим, чтобы минимизировать вероятность ошибочной классификации, и достаточно малым (в соответсвии с объемом образцовой выборки), чтобы *К* соседей располагались достаточно близко к точке запроса. Таким образом, для параметра *К*, как и для любого сглаживающего параметра, необходимо найти оптимальное значение, при котором бы достигался компромисс между смещенностью и силой размаха модели. *STATISTICA К - БС* оценивает параметр *К,* используя алгоритм кросс - проверки (Bishop, 1995).

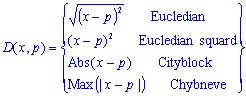
**Кросс - проверка**

Кросс - проверка - широко признанный метод получения оценок неизвестных параметров модели. В данном разделе обсуждается использование этой техники для оценки параметра *К.*

Основная идея метода заключается в разделении выборки данных на *v* "складок" (случайным образом выделенные изолированные подвыборки или сегменты). По фиксированному значению *К* строится *К - БС* модель для получения предсказаний на *v - ом* сегменте (при этом остальные сегменты используются как примеры) и оценки ошибки. Для регрессионных задач наиболее часто в качестве оценки ошибки выступает сумма квадратов, а для классификационных задач удобней рассматривать точность (процент корректно классифицированных наблюдений). Далее процесс последовательно повторяется для всех возможных вариантов выбора *v*. По исчерпании *v* "складок" (циклов), вычисленные ошибки усредняются и используются в качестве меры устойчивости модели (т.е. меры качества предсказания в точках запроса). Вышеописанные действия повторяются для различных *К* , и значение соответсвующее наименьшей ошибке (или наибольшей классификационной точности) принимается как оптимальное (оптимальное в смысле метода кросс - проверки). Отметим, что кросс - проверка вычислительно емкая процедура и следует быть готовым предоставить время для работы алгоритма особенно, если объем образцовой выборки велик. Альтернативный путь - самостоятельно задать значение параметра *К*. Этот способ приемлем, если вы располагаете обоснованными предположениями относительно возможного значения параметра (например, предыдущий *К - БС* анализ проводился над сходными данными и для них было подобрано оптимальное значение).

**Метрика расстояния**

Как было описано выше, имея точку запроса, процедура *К - БС* строит прогноз на основе выходов *K* ближайших к этой точке соседов. Следовательно, для работы метода *K* *- БС* необходимо определить метрику для вычисления расстояния между точкой запроса и наблюдениями образцовой выборки. Наиболее часто используется Евклидова метрика. Применяются и другие меры: Квадрат евклидового расстояния, Манхеттенское расстояние, расстояние Чебышева:



где *x* и *p* - точка запроса и наблюдение из образцовой выборки соответственно.

**Предсказания методом *К - Ближайщих соседей (К - БС)***

Определив значение параметра *К,* вы можете построить прогноз, используя наблюдения - примеры для метода *К - БС*. Для задачи регрессии прогноз - это усреднение выходов ближайших соседов.

http://www.spc-consulting.ru/dms/machine%20learning/Images/KNNOverViewEquationB.gif

где *yi* - *i - ое* наблюдение образцовой выборки и *y* - предсказанное значение (выход) точки запроса. В отличие от регрессии, пердсказание для задач классификации строится на основе схемы голосования, в соответсвии с которой в качестве результата берется метка победителя.

**Замечание**

Для двузначных классификационных задач во избежание ситуаций, при которых метки обоих классов получают равное число голосов, используются нечетные значения *y = 1,3,5.*

До настоящего момента мы рассматривали механизм работы анализа *К - БС*, не придавая значения относительному расстоянию между К ближайшими наблюдениями - примерами и точкой запроса. Другими словами, мы допускали, что К соседей оказывают одинаковое влияние на прогноз независимо от их относительного расстояния до точки запроса. Альтернативный подход (Shepard 1968) состоит в использовании сколь угодно больших значений параметра *К,* однако*,* при этом наибольшую значимость приобретают наблюдения, наиболее близко расположенные от точки запроса. Этот подход реализуется с ипользованием так называемого дистанционного взвешивания.

**Дистанционное взвешивание**

Так как метод предсказания *К - БС* основывается на интуитивном предположении о том, что близко расположенные друг к другу объекты потенциально похожи, имеет смысл при построении прогнозов различать *K* ближайших соседов по их расстоянию до точки запроса (т.е. допустить, что чем ближе сосед к точке запроса, тем существенней ее влияние на прогноз). Это можно реализовать, используя набор весов *W* для каждого ближайшего соседа, который формируется на основе относительной близости каждого соседа к точке запроса. Таким образом:

http://www.spc-consulting.ru/dms/machine%20learning/Images/KNNOverViewEquationC.gif

где *D(x, pi )* - расстояние между точкой запроса *x* и *pi* *(i* -*тым* наблюдением образцовой выборки). Ясно, что определенные таким образом веса будут удовлетворять:

http://www.spc-consulting.ru/dms/machine%20learning/Images/KNNOverViewEquationD.gif

Следовательно, для задачи регрессии имеем:

http://www.spc-consulting.ru/dms/machine%20learning/Images/KNNOverViewEquationE.gif

Для задач классификации максимум вышеприведенного уравнения берется для каждой переменной класса.

Из приведенного здесь описания понятно, что *K>1* естественно определяет стандартное отклонение предсказания регрессионной задачи:

http://www.spc-consulting.ru/dms/machine%20learning/Images/KNNOverViewEquationF.gif

**http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_k\_%D0%B2%D0%B7%D0%B2%D0%B5%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D1%85\_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85\_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9\_%28%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%29**

Kвзвешенных ближайших соседей - это [метрический алгоритм классификации](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80), основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты [обучающей выборки](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B0).

## Постановка задачи

Пусть X \in \mathbb{R}^n\- множество объектов; Y- множество допустимых ответов. Задана обучающая выборка \{(x_i,y_i)\}_{i=1}^\ell. Задано множество объектов \ X^m =\{x_i\}_{i=1}^m.

Требуется найти множество ответов \{y_i\}_{i=1}^mдля объектов \{x_i\}_{i=1}^m.

## Алгоритм Kвзвешенных ближайших соседей

На множестве объектов задается евклидова функция расстояния \rho(x,x'):

\rho(x,x') = \sum_{i=1}^n (x_i-x'_i)^2.

Для произвольного объекта x\in \{x_i\}_{i=1}^mрасположим объекты обучающей выборки x_iв порядке возрастания расстояний до x:

\rho(x,x_{1; x}) \leq  \rho(x,x_{2; x}) \leq \cdots \leq \rho(x,x_{m; x}),

где через x_{i; x}обозначается тот объект обучающей выборки, который является i-м соседом объекта x. Аналогичное обозначение введём и для ответа на i-м соседе: y_{i; x}.

Таким образом, произвольный объект xпорождает свою перенумерацию выборки. В наиболее общем виде [алгоритм](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) ближайших соседей есть

a(x) = \mathrm{arg}\max_{y\in Y} \sum_{i=1}^m \bigl[ x_{i; x}=y \bigr] w(i,x),

где w(i,x)— заданная *весовая функция*, которая оценивает степень важности i-го соседа для классификации объекта u. Так, при w(i,x)=1при i<kалгоритм соответствует медоду kближайших соседей. Но в задаче с несколькими возможными ответами максимальная сумма голосов может достигаться на нескольких классах одновременно. Неоднозначность можно устранить, если в качестве весовой функции взять нелинейную последовательность, например геометрическую прогрессию: в рассматриваемом примере w(i,x) = [i\leq k] q^i,что соответствует методу kэкспоненциально взвешенных ближайших соседей, причем предполагается 0.5 \leq q \leq 1.

### Алгоритм отыскания оптимальных параметров

Оптимальные значения параметров Kи qопределяют по критерию скользящего контроля с исключением объектов по одному: (k^{*};q^{*}) = \arg{ } \min_{k,q}\ LOO(k;q;X^\ell\),где LOO(k;q;X^\ell\)= \sum_{i=1}^l\[y_i = a(x_i;X^l/x_i;k;q)]   в случае равной значимости ошибок и LOO(k;q;X^\ell\)= \sum_{i=1}^l err(y_i, a(x_i;X^l/x_i;k;q))   , если ошибки не равнозначны. Здесь err(i,j)- матрица, (i,j)элементом которой является величина ошибки при классификации объекта iого класса как объект jого класса.

### Алгоритм отбора признаков

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 работе использован алгоритм жадного добавления признаков Add.

В алгоритме Add строится набор информативных призаков. Работа начинается с пустого множества. На каждом шаге iдобавяется новый признак, который при использовании с ранее выбранными минимизирует внешний критерий при числе признаков равном i(в данной работе внешний критерий - это LOO(k;q;X^\ell\)). Если в течении dшагов значение внешнего критерия не доставляло минимум внешнему критерию или добавлены все признаки, то алгоритм прекращает работу.

## Некоторые достоинства и недостатки алгоритма Kвзвешенных ближайших соседей как метрического алгоритма

### Достоинства

* Простота реализации.
* Классификацию, проведенную данным алгоритмом, легко интерпретировать путём предъявления пользователю нескольких ближайших объектов.

### Недостатки

* Неэффективный расходу памяти и чрезмерное усложнение решающего правила вследствии необходимости хранения обучающей выборки целиком.
* Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки, что требует линейного по длине выборки числа операций.

## Вычислительный эксперимент

Показана работа алгоритма в серии задач, основанных как на реальных, так и на модельных данных.

### Пример 1

Рассмотрим пример на модельных данных. Выборка состоит из четырех классов, которые являются гауссовскими рапределением с диагональными матрицами ковариации. Классы легко разделимы.

%% Example 1

% Example, which is simply classifed by WeightedKNN. It seems like XOR

% problem.

%% generate training sample

% generate 2 groups of normal classes. Each group consistes of 2 simple

% normal classes

XL1 = [GetNormClass(50,[0,0],[1,1]); GetNormClass(50,[6,6],[1,1])];

XL2 = [GetNormClass(50,[6,0],[1,1]); GetNormClass(50,[0,6],[1,1])];

XL = [XL1; XL2];

YL = [repmat(1,100,1);repmat(2,100,1)];

%% generate control data with the same distribution

X1 = [GetNormClass(50,[0,0],[1,1]); GetNormClass(50,[6,6],[1,1])];

X2 = [GetNormClass(50,[6,0],[1,1]); GetNormClass(50,[0,6],[1,1])];

X = [X1; X2];

Y = [repmat(1,100,1);repmat(2,100,1)];

%% get classification

%% choosing parametrs

PP = ParAdjust(XL, YL);

PP.XL = XL;

PP.YL = YL;

%% classification

Y = WeightKNN(X, PP);

%% results visuaisation

%% plotting real classes of objects

plot(X1(:,1),X1(:,2),'\*r');

hold on

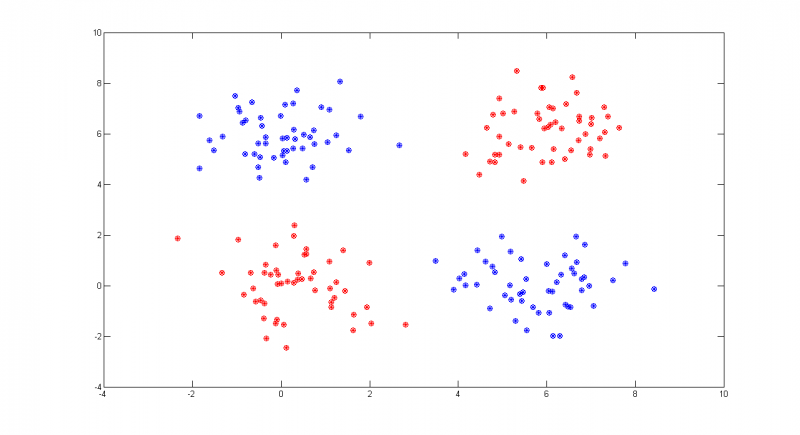
plot(X2(:,1),X2(:,2),'\*b');

%% plotting classification results

plot(X(Y == 1,1),X(Y == 1,2),'or');

plot(X(Y == 2,1),X(Y == 2,2),'ob');

hold off

[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:WKNN_ex1.png)

На графике по осям отложены величины признаков объектов, различные классы показаны крестиками различных цветов, а результат классификации показан кружочками соотвествующих цветов. Алгоритм не допустил при классификации ни одной ошибки.

### Пример 2

В качестве второго примера возьмем четыре плохо разделимых класса. Классы обладают большой дисперсией. Можно наблюдать области, в которых элементы различны классов проникают в чужие классы.

%% Example 2

% Classes cross each other in this example. Algorithm makes misstakes in

% classification.

%% generate training sample

% generating 4 sample normal classes

XL1 = GetNormClass(100,[5,0],[10,2]);

XL2 = GetNormClass(100,[5,10],[10,2]);

XL3 = GetNormClass(100,[0,5],[4,4]);

XL4 = GetNormClass(100,[10,5],[4,4]);

XL = [XL1; XL2; XL3; XL4];

YL = [repmat(1,100,1);repmat(2,100,1);repmat(3,100,1);repmat(4,100,1)];

%% generate control data with the same distribution

X1 = GetNormClass(100,[5,0],[10,2]);

X2 = GetNormClass(100,[5,10],[10,2]);

X3 = GetNormClass(100,[0,5],[4,4]);

X4 = GetNormClass(100,[10,5],[4,4]);

X = [X1; X2; X3; X4];

% X is going to be changed in getting classification. X0 is needed to plot data.

X0 = X;

%% getting classification

%% features standardization

[p, m] = size(X);

[n, m] = size(XL);

Z = [XL; X];

Z =FeaturesStand(Z);

XL = Z(1:n, :);

X = Z(n+1:n+p, :);

%% choosing parametrs

PP = ParAdjust(XL, YL);

PP.XL = XL;

PP.YL = YL;

%% classification

Y = WeightKNN(X, PP);

%% results visuaisation

%% plotting real classes of objects

plot(X1(:,1),X1(:,2),'\*r');

hold on

plot(X2(:,1),X2(:,2),'\*b');

plot(X3(:,1),X3(:,2),'\*g');

plot(X4(:,1),X4(:,2),'\*y');

%% plotting classification results

plot(X0(Y == 1,1),X0(Y == 1,2),'or');

plot(X0(Y == 2,1),X0(Y == 2,2),'ob');

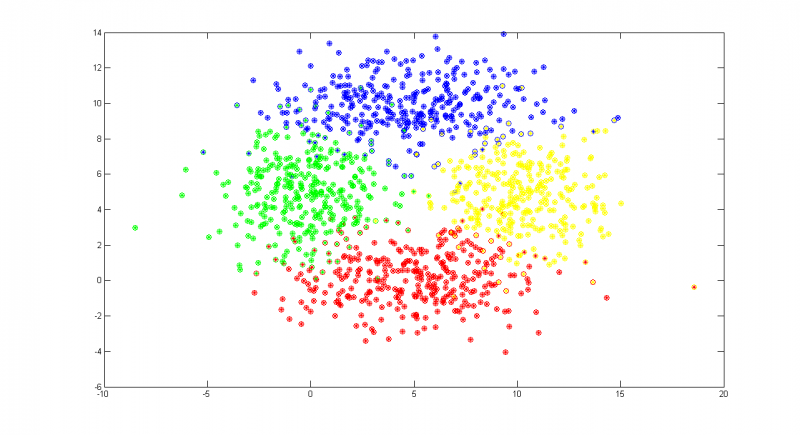
plot(X0(Y == 3,1),X0(Y == 3,2),'og');

plot(X0(Y == 4,1),X0(Y == 4,2),'oy');

%% count errors

errors = sum([Y(1:100) == 1; Y(101:200) == 2; Y(201:300) == 3; Y(301:400) == 4])

hold off

[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:WKNN_ex2.png)

На графике по осям отложены величины признаков объектов, различные классы показаны крестиками различных цветов, а результат классификации показан кружочками соотвествующих цветов. Алгоритм допустил 10% ошибок при обучении и 9% ошибок на контрольной выборке.

### Пример на реальных данных 1: ирисы

Проведем проверку алгоритма на классической задаче: [Ирисы Фишера](http://archive.ics.uci.edu/ml/datasets/Iris). Объектами являются три типа ирисов: [setosa](http://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B8%D1%81_%D1%89%D0%B5%D1%82%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%8B%D0%B9), [versicolor](http://en.wikipedia.org/wiki/Iris_versicolor), virginica

[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:Setosa.jpg)[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:Versicolor.jpg)[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:Virginica.jpg)

У каждого объекта есть четыре признака: длина лепестка, ширина лепестка, длина чашелистика, ширина чашелистика. Для удобства визуализации результатов будем использовать первые два признака. В качестве обучающей и контрольной выборок выбрано по 25 представителей каждого из типов ирисов.

%% Example 'iris'

% Example with real data

%% data preparation

%% load data

load 'iris.txt';

S = iris;

%% eliminating first two attributes

S(:,1:2) = [];

%% divide data into training part and pert to classify

XL = S([1:25,51:75,101:125],:);

X = S([26:50,76:100,126:150],:);

%% creating class labels

YL = [ones([25,1]);2\*ones([25,1]);3\*ones([25,1])];

Y = [ones([25,1]);2\*ones([25,1]);3\*ones([25,1])];

%% plotting real classes of objects

plot(X(Y == 1,1),X(Y == 1,2),'\*r');

hold on

plot(X(Y == 2,1),X(Y == 2,2),'\*b');

plot(X(Y == 3,1),X(Y == 3,2),'\*g');

%% getting classification

%% features standardization

[p, m] = size(X);

[n, m] = size(XL);

Z = [XL; X];

Z =FeaturesStand(Z);

XL = Z(1:n, :);

Xtemp = Z(n+1:n+p, :);

%% choosing parametrs

PP = ParAdjust(XL, YL);

PP.XL = XL;

PP.YL = YL;

%% classification

Y = WeightKNN(Xtemp, PP);

%% plotting resulting classification

plot(X(Y == 1,1),X(Y == 1,2),'or');

plot(X(Y == 2,1),X(Y == 2,2),'ob');

plot(X(Y == 3,1),X(Y == 3,2),'og');

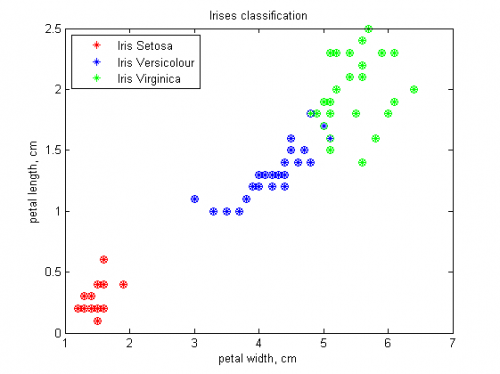
title('Irises classification')

xlabel('petal width, cm');

ylabel('petal length, cm');

legend('Iris Setosa','Iris Versicolour','Iris Virginica','Location','NorthWest');

hold off;

[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:Ireses_sorted_by_WeightedKNN.png)

На графике различные классы показаны крестиками различных цветов, а результат классификации показан кружочками соотвествующих цветов. Алгоритм хорошо классифицировал ирисы, допустив 4% ошибок.

### Пример на реальных данных 2: данные по кредитованию одним из немецких банков

Проведем проверку алгоритма на задаче из репозитория UCI: [Statlog (German Credit Data) Data Set](http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29) . Объектами являются анкеты людей, желавших получить кредит в банке. Изначально анкеты содержали 20 пунктов, таких как состояние банковсого счета заемщика, информация о его кредитной истории, цель займа, величина займа, возраст заемщика, время с момента трудоустройства на текущее место работы и другие. Но так как некоторые из них не были числовыми, а многие алгоритмы (в том числе рассматриваемый в данной статье) работают только с числовыми признаками, то из 20 разнородных признаков было составлено 24 числовых. Ответы - «хорошим» или «плохим» заемщиком оказался человек, получивший кредит. По условию задачи, распознать «плохого» заемщика как «хорошего» в 5 раз хуже, чем «хорошего» как «плохого». При обучении использован алгоритм отбора признаков Add. В выборке представлена 1000 объектов. В качестве обучающей выборки выбраны первые 690 объектов, в качестве контроля 310 оставшихся.

%% Example 'german credits'

% Example with real data

%% data preparation

%% load data

load 'germanDataNumeric.txt';

S = germanDataNumeric;

PP.XL = S(1:690,1:24);

% if votes are equal, it is better to classify as 2nd class. Make new marks

%of classes

PP.YL = 3-S(1:690,25);

% and change cost matrix

PP.errCost = [0 5; 1 0];

% initialization of d parametr

PP.d = 24;

%% run add algorithm

[features, jBest] = Add(PP);

%% run ParAdjust with selected features

% initialization of PP, X

PP = [];

PP.XL = zeros(690,jBest);

PP.XL= S(1:690,features(1:jBest));

X= S(691:1000,features(1:jBest));

PP.YL = 3-S(1:690,25);

PP.errCost = [0 5; 1 0];

% run FeaturesStand

[p, m] = size(X);

[n, m] = size(PP.XL);

Z = [PP.XL; X];

Z =FeaturesStand(Z);

PP.XL = Z(1:n, :);

X = Z(n+1:n+p, :);

% run ParAdjust

PP = ParAdjustMod(PP.XL, PP.YL, PP)

%% run WeightKNN

Y = WeightKNN(X, PP);

YTrue = 3- S(691:1000,25);

%% count weighted errors and errors

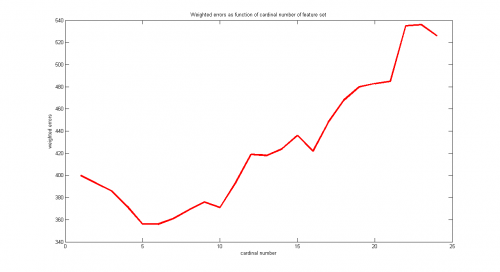
ErrW =0;

for i = [1:310]

ErrW = ErrW+ PP.errCost(YTrue(i), Y(i));

end

Err = sum(Y~=YTrue);

[](http://www.machinelearning.ru/wiki/index.php?title=%D0%98%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5:WKNN_German.png)

На графике построена зависимость величины взвешенных ошибок при обучении от числа используемых признаков. Алгоритмом Add было отобрано 5 признаков(номера 1, 15, 14, 3, 19). Названия признаков: 1 - текущее состояние банковского счета, 15 - жилье (собственное, снимаете или бесплатно предоставляется), 14 - other installment plans (bank, stores or none), 3 - сведения о ранее выдававшихся кредитах и 19 - один из девяти признаков, полученных в резульате обаботки семи пунктов анкеты (пп 4, 8, 10, 15, 17).

Средней ценой ошибки назовем величину  AE= (\sum_{i=1}^m err(y_i, a(x_i)) )/m  , где, как и ранее, err(i,j)- матрица, (i,j)элементом которой является величина ошибки при классификации объекта iого класса как объект jого класса.

При обучении средняя цена ошибок алгоритма составила 0.52 и 0.83 на контроле. Так как алгоритм допустил значительно больше ошибок на обучении, чем на контроле, то уместно говорить о [переобучении](http://www.machinelearning.ru/wiki/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Отметим также то, что средняя цена ошибок алгоритма k взвешенных ближайших соседей, запущенный без отбора признаков, составила 0.77 при обучении и 0.68 на контроле.

Таким образом, алгоритм отбора признаков существенно снизил число с 24 до 5, понизив качество классификации.

Приведем результаты работы разных алгоритмов на данной выборке:

|  |  |  |
| --- | --- | --- |
| **Название алгоритма** | **Стоимость ошибкок при обучении** | **Стоимость ошибкок на контроле** |
| Discrim | 0.509 | 0.535 |
| Quadisc | 0.431 | 0.619 |
| Logdisc | 0.499 | 0.538 |
| SMART | 0.389 | 0.601 |
| ALLOC80 | 0.597 | 0.584 |
| k-NN | 0.000 | 0.694 |
| k-NN,k=17, eucl,std | - | 0.411 |
| CASTLE | 0.582 | 0.583 |
| CART | 0.581 | 0.613 |
| IndCART | 0.069 | 0.761 |
| NewID | 0.000 | 0.925 |
| AC2 | 0.000 | 0.878 |
| Baytree | 0.126 | 0.778 |
| NaiveBay | 0.600 | 0.703 |
| CN2 | 0.000 | 0.856 |
| C4.5 | 0.640 | 0.985 |
| ITrule | \* | 0.879 |
| Cal5 | 0.600 | 0.603 |
| Kohonen | 0.689 | 1.160 |
| DIPOL92 | 0.574 | 0.599 |
| Backprop | 0.446 | 0.772 |
| RBF | 0.848 | 0.971 |
| LVQ | 0.229 | 0.963 |
| WKNN, std | 0.770 | 0.732 |
| WKNN, std, Add | 0.516 | 0.839 |

Последние две строки таблицы - результат работы алгоритма, рассматриваемого в данной статье( WKNN, std - без отбора признаков, WKNN, std, Add - с отбором). Данные для построения таблицы взяты с <http://www.is.umk.pl/projects/datasets-stat.html> .

## http://www.basegroup.ru/library/analysis/regression/knn/

## Алгоритм KNN

Рассмотрим задачу [классификации](http://www.basegroup.ru/glossary_ajax/definitions/classification).

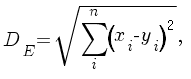
Пусть имеется n наблюдений, каждому из которых соответствует запись в таблице. Все записи принадлежат какому-либо классу. Необходимо определить класс для новой записи.

На первом шаге алгоритма следует задать число k – количество ближайших соседей. Если принять k = 1, то алгоритм потеряет [обобщающую способность](http://www.basegroup.ru/glossary_ajax/definitions/generalization_abili) (то есть способность выдавать правильный результат для данных, не встречавшихся ранее в алгоритме) так как новой записи будет присвоен класс самой близкой к ней. Если установить слишком большое значение, то многие локальные особенности не будут выявлены.

На втором шаге находятся k записей с минимальным расстоянием до [вектора признаков](http://www.basegroup.ru/glossary_ajax/definitions/attribute_vect) нового объекта (поиск соседей). Функция для расчета расстояния должна отвечать следующим правилам:

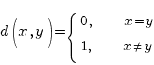
1. d(x,y) ≥ 0, d(x,y) = 0 тогда и только тогда, когда x = y;
2. d(x,y) = d(y,x);
3. d(x,z) ≤ d(x,y) + d(y,z), при условии, что точки x, y, z не лежат на одной прямой.

Для упорядоченных значений атрибутов находится Евклидово расстояние:



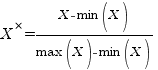
где n – количество атрибутов.

Для строковых переменных, которые не могут быть упорядочены, может быть применена функция отличия, которая задается следующим образом:

.

Часто перед расчетом расстояния необходима нормализация. Приведем некоторые полезные формулы.

Минимаксная нормализация:

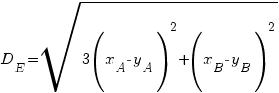
.

Нормализация с помощью стандартного отклонения:

X^{*}={X-X_cp}/sigma_x,

где sigma_x– [стандартное отклонение](http://www.basegroup.ru/glossary_ajax/definitions/rmsd), Xср – среднее значение.

При нахождении расстояния иногда учитывают значимость атрибутов. Она определяется экспертом или аналитиком субъективно, полагаясь на собственный опыт. В таком случае при нахождении расстояния каждый i-ый квадрат разности в сумме умножается на коэффициент Zi. Например, если атрибут A в три раза важнее атрибута B (ZA = 3, ZB = 1), то расстояние будет находиться следующим образом:

.

Подобный прием называют растяжением осей (stretching the axes), что позволяет снизить ошибку классификации.

Следует отметить, что если для записи B ближайшем соседом является А, то это не означает, что В – ближайший сосед А. Данная ситуация представлена на рисунке 1. При k = 1 ближайшей для точки B будет точка A, а для A – X. При увеличении коэффициента до k = 7, точка B так же не будет входить в число соседей.

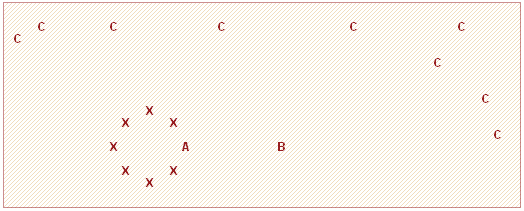


Рисунок 1 – Ближайшие соседи А и B

На следующем шаге, когда найдены записи, наиболее похожие на новую, необходимо решить, как они влияют на класс новой записи. Для этого используется функция сочетания (combination function). Одним из основных вариантов такой функции является простое невзвешенное голосование (simple un[weighted](http://www.basegroup.ru/glossary_ajax/definitions/synapse) voting).

### Простое невзвешенное голосование

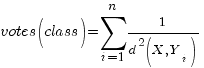
Вначале, найдя число k, мы узнали, сколько записей будет иметь право голоса при определении класса. Затем мы выявили эти записи, расстояние от которых до новой оказалось минимальным. Теперь можно приступить к простому невзвешенному голосованию.

Расстояние от каждой записи при голосовании здесь больше не играет роли. Все имеют равные права в определении класса. Запись отдает предпочтение тому классу, к которому она принадлежит, а тот, который наберет наибольшее количество голосов, присваивается новой записи.

Но что делать в случае, если несколько классов набрали равное количество голосов? Эту проблему снимает взвешенное голосование (weighted voting).

### Взвешенное голосование

В такой ситуации учитывается также и расстояние до новой записи. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по следующей формуле:

,

где d2(X, Yi) – квадрат расстояния от известной записи Yi до новой X, n – количество известных записей класса, для которого рассчитываются голоса, class - наименование класса.

Класс, набравший наибольшее количество голосов, присуждается новой записи. При этом вероятность того, что несколько классов наберут одинаковые голоса, гораздо ниже. Совершенно очевидно, что при k = 1 новой записи присваивается класс самого ближайшего соседа.

Но что делать, если стоит задача [прогнозирования](http://www.basegroup.ru/glossary_ajax/definitions/forecast) непрерывной переменной? В этом случае вместо функции сочетания используют следующие формулы:

,

w_i=1/{d^2(X,X_i)},

где – квадрат расстояния от записи Xi до новой записи X, y^*– значение выходной непрерывной переменной для новой записи.

## Примеры работы алгоритма KNN

Решим задачу классификации для ирисов (используя данные об ирисах Фишера, которые доступны по [ссылке](http://archive.ics.uci.edu/ml/datasets/Iris)). Имеется набор данных, собранных Р. Фишером, о 150 цветках трех классов: Iris Setosa, Iris Versicolour, Iris Virginica, по 50 записей для каждого. Известна длина и ширина чашелистика и лепестка всех этих ирисов.

Для простоты рассмотрим только два входных поля: длина чашелистика и длина лепестка, а также выходное – класс цветка.

Необходимо определить класс двух цветков со следующими значениями длины чашелистика и лепестка в сантиметрах: 5,3 и 1,6 (первый), 6,1 и 4,8 (второй). На рисунке 2 показано размещение новых (светло-зеленая точка – цветок 1, красная – цветок 2) записей среди уже известных.

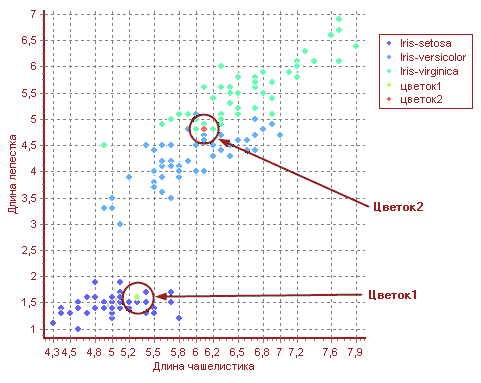


Рисунок 2 – Диаграмма размещения классов

Рассмотрим первый цветок. Установим значение k = 3. Таким образом, необходимо найти трех ближайших соседей. Для первого цветка это будут цветки со следующими значениями длины чашелистика и лепестка: A (5,3; 1,5), В(5,2; 1,5) и С(5,2; 1,5). Покажем, чему равны расстояния до первого цветка (оба атрибута равнозначны):

d(цветок1, A) = *sqrt{(5,3-5,3)^2 + (1,6-1,5)^2}=sqrt{0+0,01}=0,1*

d(цветок1, В) = *sqrt{(5,3-5,2)^2 + (1,6-1,5)^2}=sqrt{0,01+0,01}=0,141421356*

d(цветок1, C) = *sqrt{(5,3-5,2)^2 + (1,6-1,5)^2}=sqrt{0,01+0,01}=0,141421356*

Полученные сведения объединим в таблицу 1.

Таблица 1 – 3 ближайших соседа для цветка 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Запись** | **Длина чашелистика** | **Длина лепестка** | **Расстояние** | **Класс** |
| Цветок 1 | 5,3 | 1,6 | – | – |
| A | 5,3 | 1,5 | 0,1 | Iris-Setosa |
| B | 5,2 | 1,5 | 0,14 | Iris-Setosa |
| C | 5,2 | 1,5 | 0,14 | Iris-Setosa |

Теперь определим класс нового цветка. Воспользуемся простым невзвешенным голосованием. Так как все три цветка A, B и C принадлежат к классу Iris-Setosa, то он получает 100% голосов, и первому цветку присваиваем этот класс. Если увеличить (рисунок 3) на диаграмме окрестность около первого цветка, то можно увидеть, что он окружен точками синего цвета, соответствующие цветам класса Iris-Setosa.



Рисунок 3 – Увеличенный фрагмент диаграммы возле первого цветка

Теперь рассмотрим более сложный случай cо вторым цветком.

Зададим k = 3 и предположим, что длина лепестка вдвое важнее длины чашелистика. Ближайшими тремя соседями будут следующие цветки:

A(6,1; 4,7), B(6; 4,8), C(6,2 4,8)

Покажем, чему для них равны расстояния:

d(цветок2, A) = *sqrt{(6,1-6,1)^2 + 2(4,8-4,7)^2}=sqrt{0,02}=0,141421356*

d(цветок2, B) = *sqrt{(6,1-6)^2 + 2(4,8-4,8)^2}=sqrt{0,01}=0,1*

d(цветок2, C) = *sqrt{(6,1-6,2)^2 + 2(4,8-4,8)^2}=sqrt{0,01}=0,1*

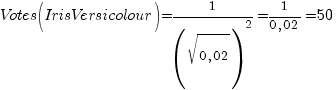
В и С – это Iris Virginica, а A – Iris Versicolour.

Объединим полученные сведения в таблицу 2.

Таблица 2 – Три ближайших соседа для цветка 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Запись** | **Длина чашелистика** | **Длина лепестка** | **Расстояние** | **Класс** |
| Цветок 2 | 6,1 | 4,8 | – | – |
| A | 6,1 | 4,7 | 0,14 | Iris Versicolour |
| B | 6 | 4,8 | 0,1 | Iris Virginica |
| C | 6,2 | 4,8 | 0,1 | Iris Virginica |

Рассмотрим взвешенное голосование.



Votes(Iris Virginica) = 1/{0,1^2}+1/{0,1^2}=2/{0,01}=200

Так как 200>50, то второй цветок классифицируется как Iris Virginica.

## Области применения алгоритма KNN

Алгоритм k-ближайших соседей имеет широкое применение. Например:

1. Обнаружение [мошенничества](http://www.basegroup.ru/glossary_ajax/definitions/fraud). Новые случаи мошенничества могут быть похожи на те, которые происходили когда-то в прошлом. Алгоритм KNN может распознать их для дальнейшего рассмотрения.
2. Предсказание отклика клиентов. Можно определить отклик новых клиентов по данным из прошлого.
3. Медицина. Алгоритм может классифицировать пациентов по разным показателям, основываясь на данных прошедших периодов.
4. Прочие задачи, требующие классификацию.

В заключение отметим достоинства и недостатки алгоритма KNN.

Перечислим положительные особенности.

1. Алгоритм устойчив к аномальным выбросам, так как вероятность попадания такой записи в число k-ближайших соседей мала. Если же это произошло, то влияние на голосование (особенно взвешенное) (при k>2) также, скорее всего, будет незначительным, и, следовательно, малым будет и влияние на итог классификации.
2. Программная реализация алгоритма относительно проста.
3. Результат работы алгоритма легко поддаётся интерпретации. Экспертам в различных областях вполне понятна логика работы алгоритма, основанная на нахождении схожих объектов.
4. Возможность модификации алгоритма, путём использования наиболее подходящих функций сочетания и [метрик](http://www.basegroup.ru/glossary_ajax/definitions/metric) позволяет подстроить алгоритм под конкретную задачу.

Алгоритм KNN обладает и рядом недостатков. Во-первых, набор данных, используемый для алгоритма, должен быть репрезентативным. Во-вторых, модель нельзя "отделить" от данных: для классификации нового примера нужно использовать все примеры. Эта особенность сильно ограничивает использование алгоритма

*Сергей Царьков*  
  
**Литература**

1. Berry, Michael J. A. “Data mining techniques: for [marketing](http://www.basegroup.ru/glossary_ajax/definitions/marketing), sales, and customer relationship management “/ Michael J.A. Berry, Gordon Linoff. – 2nd ed.
2. Larose, Daniel T. “Discovering knowledge in data: an introduction to data mining” / Daniel T. Larose

**Прочитать в книжке Загорулько стр 167**

**Журавлев**

**Метод-ближайших соседей**. Простым, но достаточно эффективным подходом здесь является метод -ближайших соседей. Оценка условных вероятностей  в методе -ближайших соседей ведется по ближайшей окрестности  точки в признаковом пространстве, содержащей по крайней мере  признаковых описаний объектов обучающей выборки. В качестве оценки  выступает отношение , где - число объектов из класса  в окрестности . Естественно, что поиск ближайшей окрестности должен основываться на использовании некоторой функции расстояний, заданной на множестве пар точек признакового пространства. В качестве такой функции расстояний в частности может выступать евклидова метрика. Точность распознавания методом -ближайших соседей существенно зависит от числа , оптимизация которого может производится по обучающей выборке. При этом в качестве оптимального берется то число ближайших соседей, при котором оценка точности распознавания с использованием режима скользящего контроля максимальна. Основным недостатком метода -ближайших соседей является снижение его эффективности при малых объемах выборки и высокой размерности признакового пространства.

**Метод голосований (основанный на частичной прецедентности)**

**Журавлев**

**Алгоритмы распознавания, основанные на принципе частичной прецедентности**

Настоящий класс алгоритмов выделен в отдельный раздел по нескольким причинам. Представляя исторически логические подходы в теории распознавания, в рамках данного подхода разработана также теория алгоритмов вычисления оценок, объединяющая все существующие методы распознавания и положенная в основу алгебраического подхода в теории распознавания. Теоретические основы алгоритмов частичной прецедентности (вычисления оценок, голосования, или комбинаторно-логических алгоритмов) описаны в многочисленных научных публикациях /25-27/, и другие. В настоящем разделе описаны некоторые алгоритмы данного класса, широко используемые в практическом распознавании.

Принципиальная идея данных алгоритмов основана на отнесении распознаваемого объекта *S* в тот класс, в котором имеется большее число «информативных» фрагментов эталонных объектов («частичных прецедентов»), приблизительно равных соответствующим фрагментам объекта *S*. Вычисляются близости – «голоса» (равные 1 или 0) распознаваемого объекта к эталонам некоторого класса по различным информативным фрагментам объектов класса. Данные близости («голоса») суммируются и нормируются на число эталонов класса. В результате вычисляется нормированное число голосов, или оценка объекта *S* за класс  – эвристическая степень близости объекта *S* к классу . После вычисления оценок объекта за каждый из классов, осуществляется отнесение объекта к одному из классов с помощью порогового решающего правила.

**1.3.1. Тестовый алгоритм.**

Тестовый алгоритм распознавания был опубликован в /15/ и базируется на понятии теста, введенного в 1956 году С.В.Яблонским. В классическом изложении тестовый алгоритм был предназначен для бинарных и к-значных признаков. Пусть 

**Определение 1.** Тестом таблицы  называется совокупность столбцов  таких, что после удаления из  всех столбцов, за исключением имеющих номера , в полученной таблице  все пары строк, принадлежащих разным классам, различны. Тест  называется тупиковым, если никакая его истинная часть не является тестом /59/.

Пусть найдено множество  тупиковых тестов таблицы  и . Выделим в описании распознаваемого объекта *S* фрагмент описания , соответствующий признакам с номерами . Сравним  со всеми фрагментами  объектов таблицы .

Число совпадений  с , , для каждого , , обозначим через .

Величина  (1.12)

называется оценкой *S* по классу *.* Имея оценки объект *S* легко классифицируется с помощью решающих правил. Например, он относится в тот класс, за который получена его максимальная оценка. В случае наличия нескольких классов с максимальными оценками, происходит отказ от его классификации.

Если отдельное совпадение в (1.12) назвать «голосом» в пользу класса **, то выражение (1.12) будет нормированным числом голосов за данный класс. В связи с данной интерпретацией, алгоритмы с подобной схемой вычисления оценок называют также алгоритмами голосования.

Тупиковые тесты широко используются для оценки информативности (важности) признаков. Пусть  - число тупиковых тестов таблицы , содержащих *i-*й столбец , а  - общее число тупиковых тестов таблицы. Тогда вес *i-*го признака определяется как . Чем больше вес, тем важнее признак для описания объектов. Это утверждение основывается на следующем рассуждении. Таблица  является исходным описанием классов и признаков. Тупиковый тест есть несжимаемое далее по признакам описание классов, сохраняющее разделение классов. Чем в большее число таких неизбыточных описаний входит *i-*й признак, тем он важнее.

Задача нахождения множества тупиковых тестов таблицы  сводится к вычислению всех тупиковых покрытий строк некоторой бинарной матрицы ее столбцами. Пусть , где . Таблице  ставится в соответствие матрица сравнения по признакам всевозможных пар объектов из различных классов , где , , .

Столбцы  образуют покрытие строк матрицы , если  : . Покрытие называется тупиковым, если произвольное его собственное подмножество не является покрытием. Каждому тупиковому тесту соответствует тупиковое покрытие строк матрицы сравнения и наоборот. Нахождение всех тупиковых тестов является сложной вычислительной задачей. Тем не менее здесь существуют подходы, эффективные для таблиц средней размерности /21/. При решении практических задач достаточно нахождение лишь части тупиковых тестов для вычисления оценок согласно (1.12). В случае наличия вещественнозначных признаков используют обычно один из следующих двух подходов. Область определения каждого вещественнозначного признака разбивается на *k* интервалов. Значение признака из *i-* го интервала полагается равным *i-1.* Далее задача распознавания решается относительно полученной *k-* значной таблицы обучения и *k-* значных описаний распознаваемых объектов. Другой подход связан с введением числовых пороговых параметров для каждого признака. Для вещественнозначных таблиц вводится аналог тупиковых тестов, в котором требование несовпадения значений признаков заменяется требованием их различия не менее чем на соответствующий порог. В обоих подходах при выборе значности новой таблицы или значений пороговых параметров необходимо учитывать, что соответствующие матрицы сравнения не должны содержать нулевых строк, поэтому значность (или пороговые параметры) следует выбирать из требования отделимости *k-* значных «образов» эталонных объектов различных классов (или отделимости с точностью до пороговых значений).

[**http://azfor.ucoz.ru/publ/3-1-0-4**](http://azfor.ucoz.ru/publ/3-1-0-4)

**Современное состояние в области практических методов распознавания,**

**классификации и анализа данных.**

Применение современных практических методов анализа данных и распознавания востребовано в технических и гуманитарных областях, в науке и производстве, бизнесе и финансах. В данном описании представлена основная алгоритмическая суть, понимание которой является полезным для более эффективного использования методов распознавания и классификации при анализе данных.

1. Задача распознавания (классификации с учителем) и современное состояние в области практических методов для ее решения. Основные этапы в развитии теории и практики распознавания: создание эвристических алгоритмов, модели распознавания и оптимизация моделей, алгебраический подход к коррекции моделей. Основные подходы - основанные на построении разделяющих поверхностей, потенциальные функции, статистические и нейросетевые модели, решающие деревья, и другие.

Более подробно описаны основные подходы и алгоритмы комбинаторно-логических методов распознавания (модели вычисления оценок или алгоритмы, основанные на принципе частичной прецедентности), разработанные в ВЦ РАН им. А.А. Дородницына. В основе данных моделей лежит идея поиска важных частичных прецедентов в признаковых описаниях исходных данных (информативных фрагментов значений признаков, или представительных наборов). Для вещественных признаков находятся оптимальные окрестности информативных фрагментов. В другой терминологии, данные частичные прецеденты называют знаниями или логическими закономерностями, связывающими значения исходных признаков с распознаваемой или прогнозируемой величиной. Найденные знания являются важной информацией об исследуемых классах (образах) объектов. Они непосредственно используются при решении задач распознавания или прогноза, дают наглядное представление о существующих в данных взаимозависимостях, что имеет самостоятельную ценность для исследователей и может служить основой при последующем создании точных моделей исследуемых объектов, ситуаций, явлений или процессов. По найденной совокупности знаний вычисляются также значения таких полезных величин, как степень важности (информативности) признаков и объектов, логические корреляции признаков и логические описания классов объектов, и решается задача минимизации признакового пространства.

2. Методы решения основной задачи кластерного анализа (классификации без учителя) – нахождение группировок объектов (кластеров) в заданной выборке многомерных данных. Приведен краткий обзор основных подходов для решения задачи кластерного анализа и описание комитетного метода синтеза коллективных решений.

3. Программная система интеллектуального анализа данных, распознавания и прогноза РАСПОЗНАВАНИЕ. В основу требований к системе положены идеи универсальности и интеллектуальности. Под универсальностью системы понимается возможность ее применения к максимально широкому кругу задач (по размерностям, по типу, качеству и структуре данных, по вычисляемым величинам). Под интеллектуальностью понимается наличие элементов самонастройки и способности успешного автоматического решения задач неквалифицированным пользователем. В рамках Системы РАСПОЗНАВАНИЕ разработана библиотека программ, реализующих линейные, комбинаторно-логические, статистические, нейросетевые, гибридные методы прогноза, классификации и извлечения знаний из прецедентов, а также коллективные методы прогноза и классификации.

1. **Алгоритмы распознавания, основанные на вычислении оценок.** Распознавание осуществляется на основе сравнения распознаваемого объекта с эталонными по различным наборам признаков, и использования процедур голосования. Оптимальные параметры решающего правила и процедуры голосования находятся из решения задачи оптимизации модели распознавания - определяются такие значения параметров, при которых точность распознавания (число правильных ответов на обучающей выборке) является максимальной.

2. **Алгоритмы голосования по тупиковым тестам.** Сравнение распознаваемого объекта с эталонными осуществляется по различным «информативным» подмножествам признаков. В качестве подобных подсистем признаков используются тупиковые тесты (или аналоги тупиковых тестов для вещественнозначных признаков) различных случайных подтаблиц исходной таблицы эталонов.

3. **Алгоритмы голосования по логическим закономерностям.**

По обучающей выборке вычисляются множества логических закономерностей каждого класса – наборы признаков и интервалы их значений, свойственные каждому классу. При распознавании нового объекта вычисляется число логических закономерностей каждого класса, выполняющихся на распознаваемом объекте. Каждое отдельное «выполнение» считается «голосом» в пользу соответствующего класса. Объект относится в тот класс, нормированная сумма «голосов» за который является максимальной. Настоящий метод позволяет оценивать веса признаков, логические корреляции признаков, строить логические описания классов, находить минимальные признаковые подпространства.

4. **Алгоритмы статистического взвешенного голосования.**

По данным обучающей выборки находятся статистически обоснованные логические закономерности классов. При распознавании новых объектов вычисляется оценка вероятности принадлежности объекта к каждому из классов, которая является взвешенной суммой «голосов».

5. **Линейная машина.**

Для каждого класса объектов находится некоторая линейная функция. Распознаваемый объект относится в тот класс, функция которого принимает максимальное значение на данном объекте. Оптимальные линейные функции классов находятся в результате решения задачи поиска максимальной совместной подсистемы системы линейных неравенств, которая формируется по обучающей выборке. В результате находится специальная кусочно-линейная поверхность, правильно разделяющая максимальное число элементов обучающей выборки.

6. **Линейный дискриминант Фишера.**

Классический статистический метод построения кусочно-линейных поверхностей, разделяющих классы. Благоприятными условиями применимости линейного дискриминанта Фишера являются выполнение следующих факторов: линейная отделимость классов, дихотомия, «простая структура» классов, невырожденность матриц ковариаций, отсутствие выбросов. Созданная модификация линейного дискриминанта Фишера позволяет успешно использовать его и в «неблагоприятных» случаях.

7. **Метод к-ближайших соседей.**

Классический статистический метод. Распознаваемый объектотносится в тот класс, из которого он имеет максимальное число соседей. Оптимальное число соседей и априорные вероятности классов оцениваются по обучающей выборке.

8. **Нейросетевая модель распознавания с обратным распространением**

Создана модификация известного метода обучения нейронной сети распознаванию образов (метод обратного распространения ошибки). В качестве критерия качества текущих параметров нейронной сети используется гибридный критерий, учитывающий как сумму квадратов отклонений значений выходных сигналов от требуемых, так и количество ошибочных классификаций на обучающей выборке.

9. **Метод опорных векторов.**

Метод построения нелинейной разделяющей поверхности с помощью опорных векторов. В новом признаковом пространстве (спрямляющем пространстве) строится разделяющая поверхность, близкая к линейной. Построение данной поверхности сводится к решению задачи квадратичного программирования.

10. **Алгоритмы решения задач распознавания коллективами различных распознающих алгоритмов.**

Задача распознавания решается в два этапа. Сначала применяются независимо различные алгоритмы Системы. Далее находится автоматически оптимальное коллективное решение с помощью специальных методов-«корректоров». В качестве корректирующих методов используются различные подходы.

11. **Методы кластерного анализа (автоматической классификации или обучения без учителя).**

Используются следующие известные подходы:

- алгоритмы иерархической группировки;

- кластеризация c критерием минимизации суммы квадратов отклонений;

- метод к-средних.

Возможно решение задачи классификации как при заданном, так и неизвестном числе классов.

12. **Алгоритм построения коллективных решений задачи классификации.**

Задача классификации решается в два этапа. Сначала находится набор различных решений (в виде покрытий или разбиений) при фиксированном числе классов с помощью различных алгоритмов Системы. Далее находится оптимальная коллективная классификация в результате решения специальной дискретной оптимизационной задачи.

[**http://www.disser.h10.ru/artical/ychplan.html**](http://www.disser.h10.ru/artical/ychplan.html)

**AdaBoost**

Материал из Википедии — свободной энциклопедии

**AdaBoost** (сокращение от Adaptive [Boosting](http://ru.wikipedia.org/w/index.php?title=Boosting&action=edit&redlink=1)) — алгоритм [машинного обучения](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предложенный Йоавом Фройндом ([en:Yoav Freund](http://en.wikipedia.org/wiki/Yoav_Freund)) и Робертом Шапиром ([en:Robert Schapire](http://en.wikipedia.org/wiki/Robert_Schapire)). Этот алгоритм может использоваться в сочетании со многими другими алгоритмами обучения для улучшения их эффективности. AdaBoost является адаптивным в том смысле, что каждый следующий классификатор строится по объектам, неверно классифицированным предыдущими классификаторами. AdaBoost чувствителен к шуму в данных и выбросам. Однако он менее подвержен [переобучению](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), чем многие другие алгоритмы обучения.

AdaBoost вызывает слабый классификатор в цикле  t = 1,\ldots,T. После каждого вызова обновляется распределение весов *Dt*, которые отвечают важности каждого из объектов обучающего множества для классификации. На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом новый классификатор «фокусирует своё внимание» на этих объектах.

## [[править](http://ru.wikipedia.org/w/index.php?title=AdaBoost&action=edit&section=1)] Алгоритм для задачи построения бинарного классификатора

Дано: (x_{1},y_{1}),\ldots,(x_{m},y_{m})где x_{i} \in X,\, y_{i} \in Y = \{-1, +1\}

Инициализируем D_{1}(i) = \frac{1}{m}, i=1,\ldots,m.

Для каждого t = 1,\ldots,T:

* Находим классификатор h_{t} : X \to \{-1,+1\}который минимизирует взвешенную ошибку классификации:  
  h_{t} = \arg \min_{h_{j} \in \mathcal{H}} \epsilon_{j}, где  \epsilon_{j} = \sum_{i=1}^{m} D_{t}(i)[y_i \ne h_{j}(x_{i})]
* Если величина \epsilon_{t} \geqslant 0.5, то останавливаемся.
* Выбираем \alpha_{t} \in \mathbf{R}, обычно \alpha_{t}=\frac{1}{2}\textrm{ln}\frac{1-\epsilon_{t}}{\epsilon_{t}}где ε*t* взвешенная ошибка классификатора *ht*.
* Обновляем:

D_{t+1}(i) = \frac{ D_{t}(i) \, e^{- \alpha_{t} y_{i} h_{t}(x_{i})} }{ Z_{t} }  
где *Zt* является нормализующим параметром (выбранным так, чтобы *Dt* + 1 являлось [распределением вероятностей](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9), то есть \sum_{i=1}^{m} D_{t+1}(i) = 1).

Строим результирующий классификатор:

H(x) = \textrm{sign}\left( \sum_{t=1}^{T} \alpha_{t}h_{t}(x)\right)

Выражение для обновления распределения *Dt* должно быть сконструировано таким образом, чтобы выполнялось условие:

e^{- \alpha_{t} y_{i} h_{t}(x_{i})} \begin{cases} <1, & y(i)=h_{t}(x_{i}) \\ >1, & y(i) \ne h_{t}(x_{i}) \end{cases}

Таким образом, после выбора оптимального классификатора h_{t} \,для распределения D_{t} \,, объекты x_{i} \,, которые классификатор h_{t} \,идентифицирует корректно, имеют веса меньшие, чем те, которые идентифицируются некорректно. Следовательно, когда алгоритм тестирует классификаторы на распределении D_{t+1} \,, он будет выбирать классификатор, который лучше идентифицирует объекты неверно распознаваемые предыдущим классификатором.

**http://cmp.felk.cvut.cz/~sochmj1/adaboost\_talk.pdf**

**http://www.site.uottawa.ca/~stan/csi5387/boost-tut-ppr.pdf**

<http://users.rowan.edu/~polikar/RESEARCH/PUBLICATIONS/csm06.pdf>

## http://cgm.computergraphics.ru/content/view/112

## Введение

В основе метода усиления простых классификаторов лежит простая предпосылка: скомбинировать некоторое количество элементарных (простых) признаков, таким образом, чтобы получить один, но более мощный. Приведём классический пример: пускай человек, играющий на скачках, решил создать программу, которая бы предсказывала, придёт ли интересующая его лошадь первой к финишу. Опросив некоторое количество играющих людей, он смог определить несколько эмпирических правил: ставь на лошадь, которая победила в трёх предыдущих заездах, ставь на лошадь, ставки на которую максимальны и т.д. Ясно, что каждое из таких правил по отдельности недостаточно надёжно и встает вопрос можно ли их оптимально скомбинировать для получения надёжных результатов.

Ответ на этот вопрос даёт семейство алгоритмов работающих на принципе усиления простых классификаторов. Это семейство использует простые правила классификации подобно деталям конструктора, комбинируя их неким образом, чтобы в итоге получить более сильное правило.

## AdaBoost

Мы рассмотрим один из самых ранних алгоритмов из данного семейства - AdaBoost (от <адаптивность> и <усиление>). Этот алгоритм был опубликован в 1996 и послужил основой для всех последующих исследований в данной области. На его основе была построена на данный момент пожалуй самая эффективная (как по уровню распознавания, так и по скорости работы) система поиска объектов на изображении (Viola-Jones Object Detector [5]). На данный момент наиболее распространёнными вариантами базового алгоритма являются Gentle AdaBoost и Real AdaBoost, превосходящие базовый алгоритм по своим характеристикам, но сохраняют все основные принципы. К основным достоинствам AdaBoost и его вариантов можно отнести высокую скорость работы, высокую эффективность распознавания, простоту реализации, общность.

### Описание алгоритма

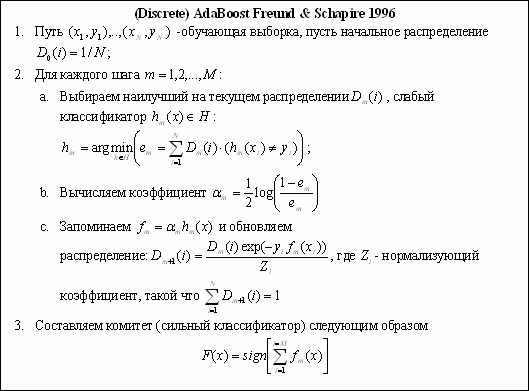
Требуется построить классифицирующую функцию http://cgm.computergraphics.ru/files/images/old_stories/112/image002.gif, где *X -* пространство векторов признаков, *Y -* пространство меток классов. Пусть в нашем распоряжении имеется обучающая выборка *(x1, y1), ..., (xN, yN)*. Где http://cgm.computergraphics.ru/files/images/old_stories/112/image006.gifвектор признаков, а http://cgm.computergraphics.ru/files/images/old_stories/112/image008.gifметка класса, к которому принадлежит http://cgm.computergraphics.ru/files/images/old_stories/112/image009.gif. Далее в статье мы будем рассматривать задачу с двумя классами, то есть *Y = {-*1*; +*1*}*. Также у нас есть семейство простых классифицирующий функций http://cgm.computergraphics.ru/files/images/old_stories/112/image013.gif. Мы будем строить финальный классификатор в следующей форме:

http://cgm.computergraphics.ru/files/images/old_stories/112/image015.gif,

где http://cgm.computergraphics.ru/files/images/old_stories/112/image017.gif. Построим итеративный процесс, где на каждом шаге будем добавлять новое слагаемое

http://cgm.computergraphics.ru/files/images/old_stories/112/image019.gif,

вычисляя его с учётом работы уже построенной части классификатора.



|  |
| --- |
|  |

На каждом шаге будем для каждого примера *(xi, yi)* из обучающей выборки вычислять его "вес": положим *D0*(*i*) *=* 1 */ N*, тогда

http://cgm.computergraphics.ru/files/images/old_stories/112/image036.gif,

где *Zi* - нормализующий коэффициент, такой что

http://cgm.computergraphics.ru/files/images/old_stories/112/image038.gif

Вес каждого элемента обучающей выборки на текущем шаге задает <важность> этого примера для очередного шага обучения алгоритма. Чем больше вес, тем больше алгоритм будет <стараться> на данном шаге классифицировать этот пример правильно. Как видно из формулы, чем уверенней пример распознаётся предыдущими шагами, тем его вес меньше - таким образом, самые большие веса получают примеры, которые предыдущими шагами были классифицированы неверно. Проще говоря, мы варьируем веса таким образом, чтобы классификатор, включенный в комитет на текущем шаге, <концентрировался> на примерах, с которыми предыдущие шагами <не справились>. Таким образом на каждом шаге мы работаем с какой-то частью данных, плохо классифицируемой предыдущими шагами, а в итоге комбинируем все промежуточные результаты.

Очередной простой классификатор мы будем выбирать исходя из взвешенной с распределением *Dm* ошибки. Мы выбираем (тренируем) http://cgm.computergraphics.ru/files/images/old_stories/112/image042.gifминимизирующий взвешенную ошибку классификации

http://cgm.computergraphics.ru/files/images/old_stories/112/image044.gif

Заметим, что если рассмотреть *Dm* как распределение вероятности над *X*, что правомерно т.к.

http://cgm.computergraphics.ru/files/images/old_stories/112/image048.gif, то

http://cgm.computergraphics.ru/files/images/old_stories/112/image050.gif

Далее вычисляется вклад текущего слагаемого классифицирующей функции

http://cgm.computergraphics.ru/files/images/old_stories/112/image051.gif

Мы продолжаем процесс до некоторого шага *M* , номер которого определяется вручную.

### Роль простого классификатора

В этом разделе мы уделим внимание фундаменту всех методов усиления простых классификаторов - семейству простых классификаторов   
http://cgm.computergraphics.ru/files/images/old_stories/112/image052.gif  
Что это такое? Для ясности приведём пример: пусть входные данные это n-мерные вектора *X = Rn*, пусть тогда

http://cgm.computergraphics.ru/files/images/old_stories/112/image056.gif

,  
то есть это порог по k-той координате. Такой классификатор в англоязычной литературе носит имя "пень" (stump) - основа дерева.

Как при таком множестве *H* происходит выбор наилучшего классификатора *hΘ, k* на каждой итерации (шаг алгоритма 2.a)? В данном случае делается следующее - для каждого *k =* 1*..n*, вычисляется порог *Θ'k*, реализующий минимум взвешенной ошибки *em*, затем из полученных классификаторов *hΘ, k*, *k =* 1*..n* выбирается соответствующий минимальной *em*.

Несмотря на свою простоту, этот классификатор, усиленный алгоритмом AdaBoost, дает весьма впечатляющие результаты. Система поиска объектов на изображении Viola-Jones находит 95% всех искомых объектов и с 0.0001% ложных срабатываний.

Какими свойствами должен обладать простой классификатор? В первую очередь, вероятность его ошибки должна быть хотя бы немного меньше 1/2, то есть он должен работать лучше чем "орел/решка":

http://cgm.computergraphics.ru/files/images/old_stories/112/image069.gif

Так же, простой классификатор должен быть максимально простой структуры (обладать малой VC-размерностью [11]) - это связано с оценкой ошибки обобщения сильного классификатора; более подробную информация можно найти здесь [1][3][4].

Самыми часто используемыми на практике простыми классификаторами являются пороги (stumps) и CART решающие деревья [12], [13].

### Внутренняя механика AdaBoost

В этой части мы попытаемся пролить немного света на внутреннюю механику алгоритма. Фактически, AdaBoost осуществляет два действия:

* Отбор простых классификаторов (простых признаков)
* Комбинирование отобранных классификаторов

Первое действие является своеобразным отображением пространства входных векторов в пространство значений простых классификаторов:

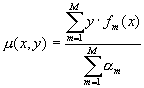
http://cgm.computergraphics.ru/files/images/old_stories/112/image071.gif

Комбинирование простых классификаторов происходит линейно (составляется линейная комбинация), а решение принимается в зависимости от знака полученной комбинации. Это фактически эквивалентно разделению пространства значений простых классификаторов гиперплоскостью и принятие решения в зависимости от того, по какую сторону гиперплоскости лежит отображение вектора признаков.

Таким образом, готовый классификатор производит вначале отображение в некое пространство, обычно намного более высокой размерности, чем исходное, в котором производит линейную классификацию. На этапе тренировки алгоритм последовательно строит и это отображение, и саму гиперплоскость.

Стоит заметить, что работа AdaBoost в значительной мере напоминает работу алгоритма ядерной машины опорных векторов (Kernel Support Vector Machine - kernel SVM). Исследования последних лет показывает глубокую связь этих двух алгоритмов, что является серьёзным теоретическим результатом [9].

Одна из интерпретаций работы алгоритмов на основе AdaBoost основана на понятие <грани> (margin). В случае AdaBoost грань определяется как:



Эту величину можно интерпретировать как меру <уверенности> классификатора в примере *(x, y)*. Если классификация правильная, то грань больше нуля, иначе грань отрицательна. Чем больше простых классификаторов правильно классифицируют пример, тем больше его грань.

Если учесть то, как на каждом шаге вычисляются веса примеров, то легко видеть, что на каждом шаге AdaBoost пытается максимизировать минимальную грань тренировочной выборки. Утверждается, что данное действие положительно сказывается на обобщающих способностях алгоритма. Больше про данную интерпретацию семейства алгоритмов на основе AdaBoost можно прочитать в [9].

## Заключение

В данный момент подход усиления простых классификаторов является одним из наиболее популярных и, вероятно, наиболее эффективным методом классификации. За счёт высокой скорости, простоты реализации и высокой эффективности распознавания это семейство алгоритмов нашло свое применение во множестве областей так или иначе связанных с классификацией (медицина, анализ изображений, анализ текстов и т.д.).

Мы кратко описали самый базовый алгоритм, основывающийся на идеи усиления простых классификаторов. Все последующие его модификации сохраняют основные свойства своего предка. Для более детального знакомства предлагается обратиться к указанной в библиографии литературе, а также посетить интернет-сайт [www.boosting.org](http://www.boosting.org/).