Лекция 11

1. Композиционные алгоритмы распознавания образов. Баггинг и бустинг

Композиционные (комитетные) алгоритмы (КА) основаны на объединении ансамбля из нескольких классификаторов (распознающих алгоритмов), каждый из которых дает свою оценку ситуации, пригодную для принятия решения. На основе совокупности результатов, полученных от всех классификаторов, выдается итоговый общий результат в виде окончательного решения.

Композиционные алгоритмы настраиваются на основе обучающих данных (прецедентов). Основная идея – ансамбль классификаторов может обеспечить существенно более высокую достоверность распознавания, чем каждый из них в отдельности.

1. Общая структура композиционных алгоритмов

Пусть есть классы образов $\omega_1,...\omega_M$ и связанная с ними смешанная проиндексированная обучающая выборка

$$X^N=\{x^{(1)},...,x^{(N)}\},\ D^N=\{d^{(1)},...,d^{(N)}\},\ d^{(*)}\in D=\{1,2,...M\},$$

$$x^{(*)}\in\omega_j,\ d^{(*)}=j,\quad \{x^{(i)},d^{(i)},i=\overline{1,N}\}.$$

Вводится вспомогательное множество оценок R, называемое пространством оценок. Рассматриваются алгоритмы, имеющие вид суперпозиции

$$g'(x)=g'[g(x)],$$

где $g: X\to R$ называется базовым алгоритмом (БА), а функция $g': R\to D$ – решающим правилом.

Композицией БА $g_k(x),k=\overline{1,L}$ называется решающее правило вида, основанное на использовании суперпозиции

$$G'(x)=G'[G(g_1(x),...g_L(x))],G: R^L\to R,\ G': R\to D,$$

где $G: R^L\to R$ называется корректирующей операцией.

Замечание: введение корректирующей операции для получения сначала комбинированной оценки на множестве R вместо того, чтобы использовать совокупность собственных решений БА (взять в качестве суперпозиции $G: D^L \to D$) дает возможность расширить перечень рассматриваемых вариантов объединения базовых алгоритмов.

Рассмотрим два класса ω_1, ω_2 образов. Пусть имеется ансамбль БА $\{g_k(x), k = \overline{1, L}\}$, каждый из которых обучен формировать значения оценки. Эти значения далее могут быть использованы для принятия решений вида

$$g'_k(x) = sign[g_k(x)] = \begin{cases} +1 \to x \in \omega_1, \\ -1 \to x \in \omega_2. \end{cases}$$

Для случая двух классов с использованием понятия вероятности ошибки p_g вводится следующие определения для базовых классификаторов:

классификатор называется плохим, если $p_g \geq 1/2$;

классификатор называется слабым, если $p_g = 1/2 - \varepsilon$;

классификатор называется сильным, если $p_g = \varepsilon$.

Различают линейные композиции (комбинации) БА и смеси БА нелинейного вида. Стандартный вариант **линейного** ансамбля БА:

$$G(x) = G\left[\sum_{k=1}^{L} \alpha_k g_k(x)\right], \quad G'(x) = sign[G(x)], \quad \sum_{k=1}^{L} \alpha_k = 1,$$

где $\alpha_k \geq 0, k = \overline{1, L}$ – весовые коэффициенты, учитывающие вклад каждого базового алгоритма. Если $\alpha_k = 1/L, k = \overline{1, L}$, то такое объединение называется простым голосованием.

Простейший вариант: использование суммы бинарных пороговых классификаторов

$$G'(x) = sign[G(x)] = sign\left[\sum_{k=1}^{L} sign(g_k(x))\right].$$

Такой классификатор обеспечивает принятие решений путем «голосования» по большинству решений, принимаемых членами комитета в пользу того или иного класса.

Наиболее часто встречающиеся подходы к реализации КА: **баггинг и бустинг**.

Баггинг (bagging) – сокращение от bootstrap aggregating – агрегированный бутстреп. Bootstrap – подход к увеличению репрезентативности обучающей выборки, основанный на случайном извлечении (с последующим возвращением) из исходной обучающей выборки нескольких подмножеств примеров с целью получения устойчивых оценок. При выполнении баггинга композиция составляется на основе обучения нескольких классификаторов по бутстреп - множествам, полученным из общей обучающей выборки, и агрегирования результатов их работы путем простого голосования. **Общая идея баггинга –** снижение зависимости «экспертов» – базовых классификаторов ансамбля друг от друга.

Бустинг (boosting – улучшение) – процедура итеративного последовательного построения композиций базовых алгоритмов. Каждая следующая композиция включает большее количество БА, стремясь при этом компенсировать недостатки предыдущей композиции.

На каждой итерации осуществляется перевзвешивание наблюдений, добавление и коррекция весов базовых алгоритмов. В итоге происходит «усиление» слабых классификаторов и повышение результирующей эффективности распознавания. **Общая идея бустинга –** «эксперты» (базовые классификаторы) учатся на ранее допущенных ошибках других классификаторов.

2. Баггинг: деревья решений и композиции «случайный лес»

Для иллюстрации идей баггинга первоначально рассмотрим алгоритмы принятия решений, известные под названием «деревья решений». Идея, лежащая в основе деревьев решений (ДР), состоит в разбиении множества возможных значений вектора признаков на непересекающиеся множества и настройке простой модели решений для каждого такого множества.

ДР представляет собой ориентированный (заданы направления соединения вершин ребрами графа) связный граф без циклов (без обратных связей).

Под корневым деревом понимается дерево, в котором одна вершина выделена и называется корнем. В качестве ДР рассматриваются только ориентированные корневые деревья, в которых дуги (ориентированные ребра) направлены от корня к вершинам. Они удовлетворяют следующим условиям:

существует только одна вершина, называемая корнем, в которую не ведет ни одна дуга;

в каждую вершину (исключая корень) ведет только одна дуга; существует единственный путь от корня к любой вершине.

Если (v,w) – некоторая дуга, то вершина v называется родителем w, а вершина w – потомком вершины v. Вершина, не имеющая потомков, называется терминальной вершиной или листом. Дерево называется бинарным, если каждая его вершина (за исключением терминальных вершин) имеет ровно двух потомков.

Алгоритм формирования дерева решений для распознавания образов реализуется на основе смешанной проиндексированной обучающей выборки $XD^N = \{(x^{(i)}, d^{(i)}), i = \overline{1,N}\}$. Пусть $X \subseteq \mathbf{R}^n$ пространство образов. Деревом решений называется граф, в котором для каждой вершины t заданы:

- подмножество в признаковом пространстве $X_t \subset X$, при этом с корневой вершиной связывается все пространство X;
- подвыборка обучающей выборки $XD_t \subset XD^N$, такая, что $XD_t = \{(x,d) \in XD^N, x \in X_t\}$, при этом с корневой вершиной связывается вся XD^N;
- правило (функция) $h_t : X_t \to \{0,1,\ldots,k_{t-1}\}$, где $k_t \geq 2$ – количество потомков вершины t, определяющее разбиение множества X_t на k_t непересекающиеся подмножества, при этом для терминальных вершин правило не вводится.

Обозначим $t_{i(t)}$, $i = 0,1,\ldots,k_{t-1}$ вершину, являющуюся i - м потомком вершины t. Подмножество X_t и правило h_t определяют подмножества $X_{t_{i(t)}}$, возникающие после разбиения, следующим образом:

$$X_{t_{i(t)}} = X_t \cap \{x \in X : h_t(x) = i\}.$$

Цель построения дерева решений состоит в распознавании нового вектора x. Процесс принятия решений начинается с корневой вершины и состоит в последовательном применении правил, связанных с вершинами. Результатом этого процесса является определение терминальной вершины t такой, что $x \in X_t$. Вектор x относят к классу, которые наиболее часто встречается в подвыборке, связанной с этой терминальной вершиной.

Классический алгоритм CART (Classification And Regression Trees), реализующий данный подход, основан на идее рекурсивного разбиения обучающей выборки на две более однородные подвыборки с помощью одного из признаков. Для реализации этой идеи необходимо определить понятие

однородности, которая обычно рассчитывается на основе показателей, характеризующей степень загрязненности подвыборок при выполнении разбиения.

Показатель загрязненности (impurity). Общая идея состоит в следующем. Пусть t – вершина дерева решений, $XD_t \subset XD^N$ – подвыборка, связанная с этой вершиной, и $J(t)$ – загрязненность вершины. Загрязненность вершины равна 0, если XD_t содержит прецеденты только одного класса и максимальна в случае, если XD_t содержит одинаковое число прецедентов каждого класса.

Используются различные показатели загрязненности. Индекс Джини представляет собой частоту ошибочной классификации при случайном назначении меток классов образам подвыборки XD_t.

$$J(t) = 1 - \sum_{i=1}^{M} N_t^2(\omega_i)$$

Расщепление деревьев (split). Правило разбиения множества X, связанное с каждой вершиной дерева решений, называется расщеплением.

Бинарное расщепление вершины t можно рассматривать как функцию $h_t : X_t \to \{0,1\}$, $x \in X_t$, где в случае $h_t(x) = 0$ вектор относится к первому (левому) потомку, а в случае $h_t(x) = 1$ – ко второму (правому).

Расщепление состоит в выборе порогового значения x_{*0}, которое минимизирует используемый показатель загрязненности. В этом случае

$$h_t(x) = 0,\ x_* < x_{*0},\ h_t(x) = 1,\ x_* \geq x_{*0}.$$

Области решений будут представлять собой многомерные параллелепипеды, при этом любая часть границы области решений, соответствующая данному расщеплению, будет представлять собой часть гиперплоскости, параллельной соответствующей координатной оси в пространстве признаков.

Критерии остановки расщепления. Процесс расщепления вершин имеет естественный предел, когда становится невозможно уменьшить загрязненность очередной вершины. Такие вершины объявляются терминальными, а соответствующее дерево называется полным.

В полном ДР каждая терминальная вершина содержит примеры только одного класса, а само дерево содержит большое количество вершин.

Полное дерево, как правило, обладает низкой достоверностью классификации. Это связано с уже упомянутой проблемой переобучения, заключающейся в том, что построенная деревом слишком точная модель обучающих данных фактически описывает только эту выборку и непригодна в качестве модели для другой выборки.

Использование критериев остановки расщепления ДР позволяет решить проблему построения слишком больших и детальных деревьев играют. Наиболее простой способ состоит в задании минимального числа для количества наблюдений в подвыборках, соответствующих терминальным вершинам (или минимальной доли наблюдений обучающей выборки).

Усечение деревьев (pruning). Недостатком подхода, основанного на способах остановки расщепления вершин для предотвращения построения больших деревьев, является то, что решение об остановке принимается без учета ситуации, которая могла бы сложиться при продолжении расщепления.

Альтернативный подход – построение полных деревьев и их дальнейшее усечение.

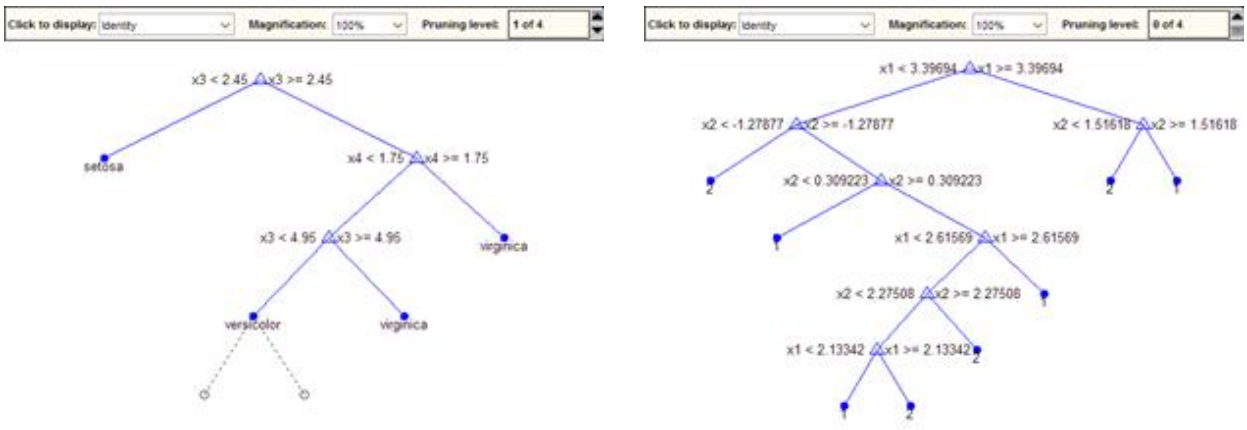
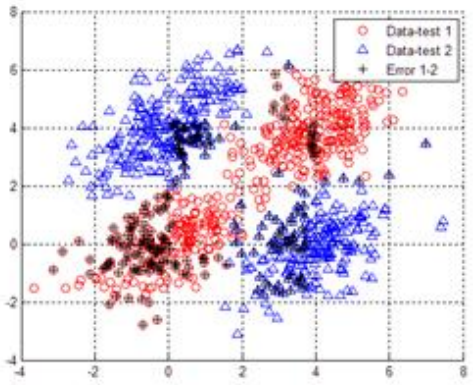
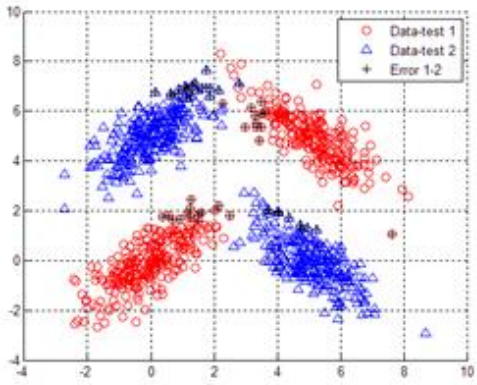


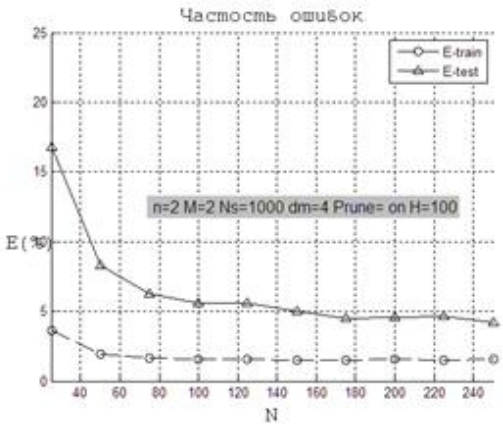
Рис.1. Исходное и усеченное на один уровень деревья решений в примере fisheriris (а), дерево решений в примере «заплетенные восьмерки (б)



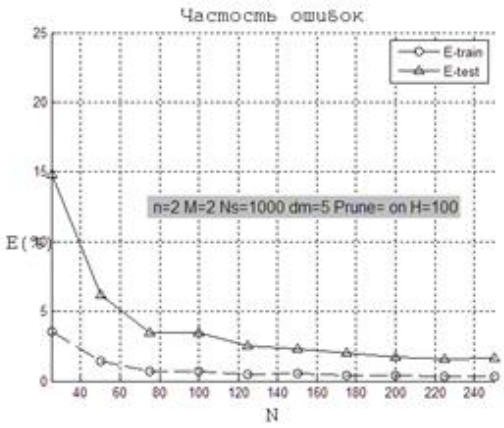
а)



б)



в)



г)

Рис.2. Результаты тестирования алгоритма ДР на основе дерева решений для примера «заплетенные восьмерки»

Усечение означает процедуру замены в построенном полном дереве некоторой вершины и связанного с ней поддерева терминальной вершиной.

Большинство методов усечения основано на оценке чувствительности поддеревьев по отношению к некоторой мере и удалению поддеревьев, которые оказывают минимальное влияние на эту меру.

Обобщающая способность ДР не так хороша, как у алгоритма SVM: наблюдается существенное отличие числа ошибок, допущенных на обучающей и на тестирующей выборках (рис.1,2). В примере «заплетенные восьмерки» степень «заплетенности» классов определяется параметром dm, задающим относительный сдвиг центров областей локализации данных в каждом классе (рис.2а,б). Если dm=0, то классы сливаются, если dm становится достаточно большим, то каждый класс состоит из двух не пересекающихся областей эллипсоидальной формы.

Случайный лес (Random Forest) на основе баггинга. Случайные леса реализуют подходы к построению ансамбля классификаторов в виде ДР, основанные на манипулировании с обучающими данными и манипулировании с признаками с включением в эти процессы элементов случайности.

«Случайный лес», основан на формировании бутстреп - обучающей выборки для каждого классификатора ансамбля путем случайной выборки с возвращением из исходной обучающей выборки. При этом каждый раз получается подвыборка того же объема, что и исходная обучающая выборка.

Каждая бутстреп - выборка содержит в среднем примерно 63% наблюдений исходной обучающей выборки: поскольку выборка с возвращением, то некоторые наблюдения в нее не попадают, а некоторые попадают несколько раз).

В качестве ансамбля БА в случайном лесе рассматривается ансамбль ДР, каждое из которых строится на основе бутстреп-подвыборки из исходной обучающей, причем для расщепления вершин используется только доля случайно отбираемых признаков. При объединении, или агрегирования решений отдельных классификаторов используется метод голосования.

Информация, касающаяся деталей алгоритма, обзор и примеры могут быть найдена по адресу:

http://www.stat.berkeley.edu/users/breiman /RandomForests

Стандартный алгоритм построения случайного леса. На этапе обучения реализуется индукция леса. В цикле для $k=\overline{1,L}$ деревьев ансамбля выполнить следующие действия:

- по исходной обучающей выборке $XD^N=\{(x^{(i)},d^{(i)}),i=\overline{1,N}\}$ сформировать бутстреп выборку XD_k^N (случайную выборку того же объема с возвращением элементов);
- по бутстреп выборке XD_k^N индуцировать не усеченное дерево решений T_k с минимальным количеством наблюдений в терминальных вершинах, равным $n_{\min}$, рекурсивно следуя следующей процедуре:

  a) из исходного набора n признаков случайно выбрать p признаков;

  b) из p признаков выбрать признак, который обеспечивает наилучшее расщепление;

c) расщепить выборку, соответствующую обрабатываемой вершине, на две подвыборки;

В результате получаем ансамбль деревьев решений $\{T_k, k = \overline{1, L}\}$.

Распознавание новых наблюдений: выбирается класс образов $x \in \omega_j$, который максимальное число раз выбран базовыми классификаторами:

$$\tilde{d}(x) = i = \arg\max_{j=\overline{1,J}} \left\{ \sum_{k=1}^{L} I_k(x \in \omega_j) \right\}, \quad I_k(x \in \omega_j) = \begin{cases} 1, x \in \omega_j, \\ 0, x \notin \omega_j. \end{cases}$$

Одним из достоинств случайных лесов является то, что для оценки вероятности ошибочной классификации нет необходимости использовать тестовую выборку. Оценка вероятности ошибочной классификации случайного леса может осуществляться внутренним образом на основе метода **"Out-Of-Bag" (OOB).**

При OOB выполняется распознавание каждого вектора $x^{(*)} \in X^N$ с использованием при голосовании только тех деревьев, которые строились по бутстреп - выборкам, не содержащим $x^{(*)}$. Частота ошибочно классифицированных векторов обучающей выборки при таком способе представляет собой оценку вероятности ошибочной классификации случайного леса методом OOB.

Случайные леса обладают рядом привлекательных свойств:

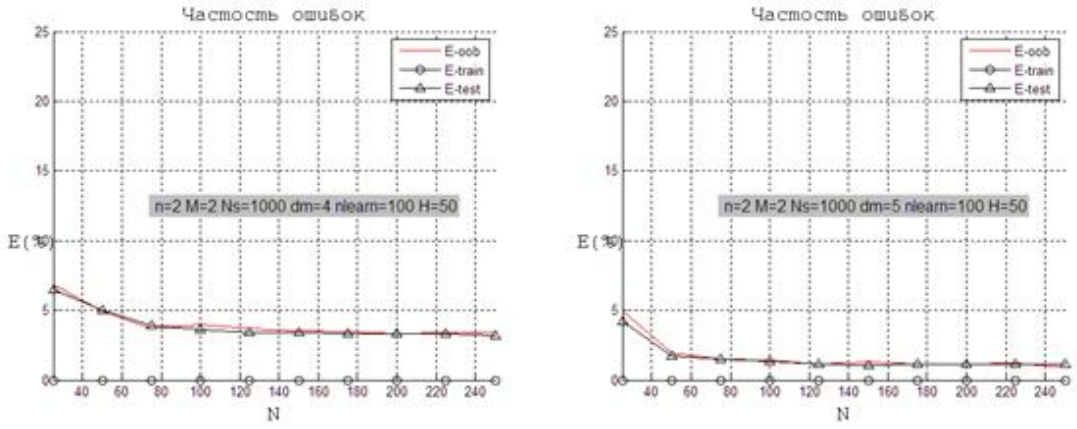
- повышение достоверности распознавания за счет слабой зависимости деревьев вследствие двойной инъекции случайности – посредством баггинга и использования случайного набора признаков при расщеплении каждой вершины;

- сложная задача усечения полного дерева решений снимается, поскольку деревья в случайном лесу не усекаются (это также приводит к высокой вычислительной эффективности).

- проблема переобучения не стоит так остро, поскольку ансамбль ДР уже не является единственным детально настроенным алгоритмом;

- простота настройки алгоритма: единственными параметрами являются количество деревьев и количество признаков, случайно отбираемых для расщепления в каждой вершине.

Алгоритм «случайный лес» показывает в целом лучшие результаты по сравнению алгоритмом, основанным на использовании одного дерева решений (рис.3).



а) б)

Рис.3. Результаты тестирования алгоритма «случайный лес» для примера «заплетенные восьмерки»

3. Композиции базовых алгоритмов на основе бустинга

Композиции, формируемые на основе бустинга, обладают следующими особенностями:

реализация последовательного процесса обучения за несколько итераций;

учет предыдущих результатов классификации образов из обучающей выборки при выполнении каждой итераций в плане допущенных на них ошибках;

учет результатов работы ранее используемых базовых классификаторов в плане допущенных ими ошибок.

Цель подобной обработки состоит в том, чтобы усилить слабые классификаторы.

Описание стандартного алгоритма AdaBoost (adaptive boosting). Пусть на входе имеется $XD^N = \{(x^{(i)}, d^{(i)}), i = \overline{1,N}\}$ обучающая выборка для двух классов образов ω_1, ω_2 $d^{(i)} \in \{-1, +1\}$. При формировании композиции выполняется L итераций. В качестве БА рассматриваются алгоритмы

$$g_k(x) = \begin{cases} +1 \to x \in \omega_1, \\ -1 \to x \in \omega_2. \end{cases}$$

В качестве итоговой композиции рассматривается алгоритм

$$G(x) = sign\left[\sum_{k=1}^{L} \alpha_k g_k(x)\right].$$

Для произвольного БА вводится взвешенное число допущенных ошибок

$$E(g_k, w) = \sum_{i=1}^{N} w_i I(g_k(x^{(i)}) \neq d^{(i)}), \quad \sum_{i=1}^{N} w_i = 1$$

Общий функционал качества распознавания, который следует минимизировать, определяется как

$$Q(g, \alpha) = \sum_{i=1}^{N}\left[M(x^{(i)}) < 0\right] = \sum_{i=1}^{N}\left[d^{(i)} \sum_{k=1}^{L} \alpha_k g_k(x^{(i)}) < 0\right] \to \min .$$

1. Назначить одинаковые начальные веса для элементов обучающей выборки $w_i = 1/N, \ i = \overline{1, N}$ (имеют смысл априорных вероятностей).

2. Для каждого шага $k = 1,2,\ldots,L$

   - обучить базовый алгоритм $g_k(x)$ на XD^N, который минимизирует взвешенную ошибку классификации $E_k = \min E(g_k, w)$;
   - если $E_k = 0$, то $G(x) = g_k(x)$ и переход к п. 3.
   - если $E_k > 1/2$, то переход к п. 4.
   - вычислить коэффициент $\alpha_k = \frac{1}{2}\ln\left((1 - E_k)/E_k\right)$ и зафиксировать его как вес соответствующего БА;
   - пересчитать веса обучающих примеров следующим образом $w_i' = w_i \exp(-\alpha_k d^{(i)} g_k(x^{(i)})), \ i = \overline{1, N}$
   - выполнить нормировку весовых коэффициентов образов $\sum_{i=1}^{N} w_i' = 1$.

3. Зафиксировать композицию $G(x) = sign\left[\sum_{k=1}^{M} \alpha_k g_k(x)\right]$.

4. Окончание.

В алгоритме при перевзвешивании наблюдений усиливается роль тех из них, на которых допущены ошибки. Большие веса получают те объекты, которые «плохо» классифицировались на предыдущих шагах.

Весовые коэффициенты БА учитывают допущенные ими ошибки как напрямую, так и косвенно, через веса обучающих образов. Таким образом, на каждом последующем шаге новый классификатор приспосабливается исправить ранее допущенные ошибки предыдущих классификаторов.

При выполнении алгоритма по схеме AdaBoost на каждом шаг добавляется новый классификатор, при этом значения коэффициентов БА, полученные на предыдущих шагах не изменяются.

Использование приведенных соотношений для перевзвешивания весов образов эквиваленты замене отступов на экпоненциальную функцию

$$\left[M(x^{(i)}) < 0\right] \le \exp\left[-M(x^{(i)})\right]$$

и минимизации функционала вида

$$\overline{Q}(g, X) = \sum_{i=1}^{N} \exp\left[-M(x^{(i)})\right] \ge Q(g, X), \; H(M) = \exp(-M)$$

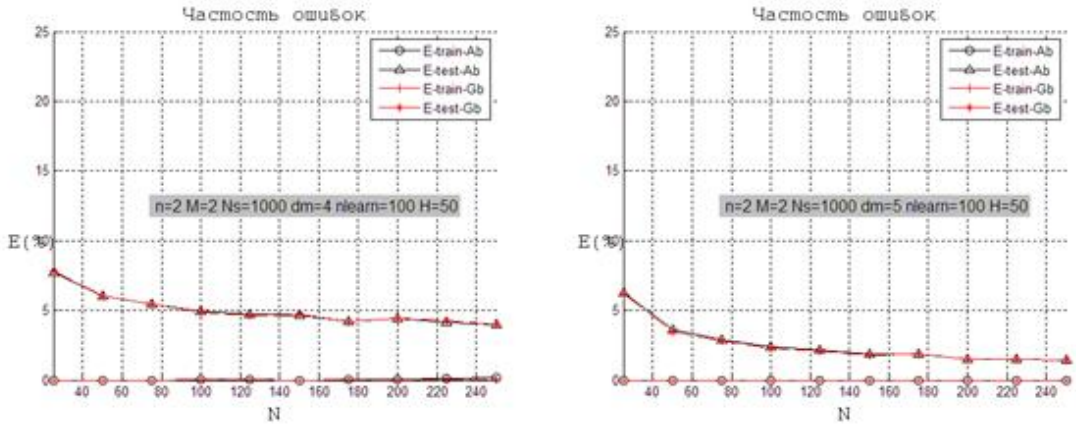
В качестве БА в алгоритмах типа AdaBoost чаще всего используются **деревья решений малой высоты** и простые правила порогового типа.

Среда MATLAB предоставляет широкие возможности по применению следующих вариантов организации бустинга и, прежде всего, следующие методы:

- 'AdaBoostM1' – классический AdaBoost для распознавания двух классов;
- 'AdaBoostM2' – классический AdaBoost для распознавания многих классов;
- 'LogitBoost' – бустинг с использованием в качестве $H(M)$ логарифмической функции;
- 'GentleBoost' – бустинг с использованием в качестве $H(M)$ квадратичной функции;
- 'RobustBoost' – бустинг с использованием приемов, направленных на исключение чрезмерного повышения весов отдельных наблюдений и последующего игнорирования большей частью обучающих данных;

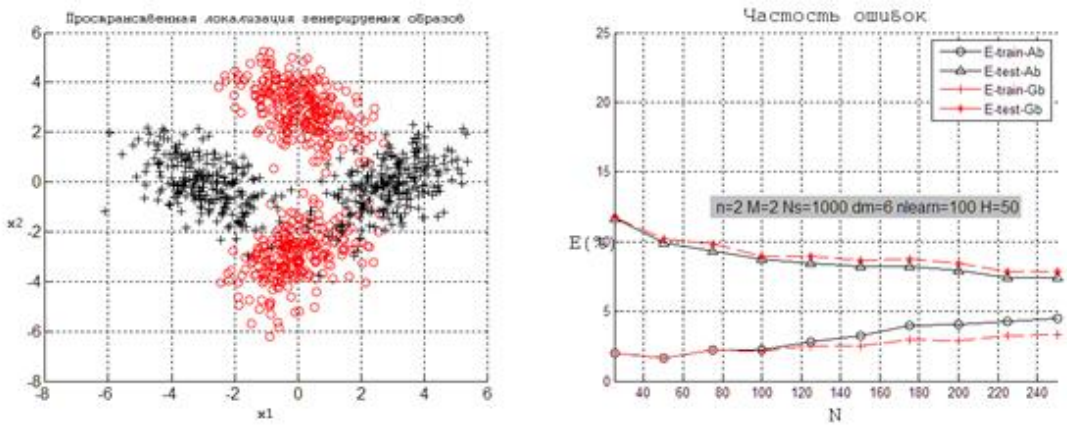
- 'LPBoost' – бустинг с использованием для оптимизации коэффициентов методов линейного программирования.

Результаты тестирования различных вариантов алгоритмов, основанных на бустинге (AdaBoostM1 и GentleBoost) для примера «заплетенные восьмерки» при различных конфигурациях областей локализации классов представлены на рис.4 (стандартная, рис.2а) и рис. 5 (отличающаяся от стандартной, рис.5а). Графики отражают зависимости частости ошибок от объема обучающих данных.



а) б)

Рис.4. Результаты тестирования алгоритмов AdaBoostM1 и GentleBoost для примера «заплетенные восьмерки» при стандартной конфигурации рис.2а



а) б)

Рис.5. Результаты тестирования алгоритмов AdaBoostM1 и GentleBoost для примера «заплетенные восьмерки» для конфигурации рис.5а

◄ Лекция 10
Перейти на...
Перейти на...

Лекция 12 ►