# «Полигон» — распределённая система для эмпирического анализа задач и алгоритмов классификации*

***Воронцов К. В., Ивахненко А. А, Инякин А. С., Лисица А. В., Минаев П. Ю.***
vokov@forecsys.ru, lisitsa@forecsys.ru
Москва, Вычислительный Центр РАН, ЗАО «Форексис»

Система «Полигон» предназначена для массового выполнения типовых экспериментов по тестированию алгоритмов классификации на модельных и реальных данных. В отличие от существующих систем такого типа, «Полигон» является Интернет-ресурсом, имеет централизованное хранилище задач и результатов тестирования, распределённую наращиваемую пользователями сеть вычислительных серверов и расширенную визуальную методику тестирования, основанную на $t\times q$-кратном скользящем контроле. Ресурс ориентирован на специалистов по анализу данных, прикладных экспертов, разработчиков алгоритмов, научных работников, учащихся и преподавателей вузов.

На конференции ММРО-13 было анонсировано начало работ по созданию распределённой системы тестирования алгоритмов классификации на задачах со стандартным представлением исходных данных в виде матрицы «объекты–признаки» [2]. В настоящее время прототип системы доступен по адресу `http://poligon.MachineLearning.ru`.

«Полигон» автоматизирует типовое эмпирическое исследование по сравнению качества заданного набора алгоритмов на заданном наборе задач. Проведение такого рода сравнительных экспериментов считается обязательным как при разработке новых методов классификации, так и при решении практических задач классификации.

В настоящее время широко известные системы анализа данных Matlab, R, STATISTICA, SAS, SPSS и др. имеют мощные библиотеки алгоритмов классификации. Свободно доступные системы для решения задач машинного обучения (RapidMiner, WEKA) укомплектованы наборами реальных задач и процедурами скользящего контроля для проведения упомянутых выше типовых экспериментов. Тем не менее, создание новой системы имеет смысл в силу ряда причин.

1. Перечисленные системы не гарантируют воспроизводимость и верифицируемость результатов тестирования. Все они устанавливаются локально на компьютере пользователя, оставляя ему возможность по-своему реализовать методику тестирования, модифицировать как исходные данные, так и сами алгоритмы. Как следствие, эмпирические результаты, представленные в разных публикациях, оказываются несопоставимыми, даже если тестирование проводилось на одних и тех задачах, как правило, из репозитория UCI [3].

2. В системах с открытым кодом R, RapidMiner, WEKA невозможно широкое использование коммерческих алгоритмов. В то же время, создатели алгоритмов могут быть заинтересованы в предоставлении их для пробного использования, не предполагающего извлечения коммерческой выгоды, и при этом максимально простого для пользователя. «Полигон» предоставляет удобную площадку для такого использования, при этом алгоритмы остаются на серверах правообладателя, и он имеет возможность ограничивать доступ к ним.

3. В «Полигоне» нет ограничений на язык программирования, на котором реализуются алгоритмы. RapidMiner и WEKA допускают только Java, что снижает скорость выполнения алгоритмов и затрудняет перенос в систему готовых алгоритмов, написанных на других языках.

4. В настоящее время стандартной методикой тестирования считается $t\times q$-кратный скользящий контроль [7]. В минимальном варианте для каждой пары ⟨алгоритм, задача⟩ вычисляется один скалярный критерий — средняя частота ошибок на контроле, как правило, с доверительным интервалом. В «Полигоне» реализована расширенная методика тестирования, включающая ряд широко известных методов, дающих более детальное и наглядное представление о качестве классификации. Эти методы обобщены и унифицированы таким образом, чтобы каждый из них, по возможности, позволял анализировать качество отдельно по классам, сопоставлять обучение и контроль и вычислять доверительные интервалы для всех оцениваемых величин.

Цель данного сообщения — показать возможности реализованной в «Полигоне» методики тестирования и привлечь научное сообщество к активному использованию и пополнению системы алгоритмами и задачами.

## Класс решаемых задач

С точки зрения «Полигона» алгоритм классификации — это функция, принимающая на входе:

- обучающую выборку в виде матрицы «объекты–признаки» $(x_{ij})_{\ell\times n}$, где $x_{ij}$ — значение $j$-го признака на $i$-м обучающем объекте $x_i$;
- вектор ответов $(y_i)_{i=1}^{\ell}$, соответствующих объектам $x_i$, где $y_i \in Y$, $Y$ — множество классов;

*Работа поддержана РФФИ (проекты №07-07-00372, №08-07-00422, №07-07-00181), программой ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики и информационные системы нового поколения».

— матрицу потерь $(C_{yy'})_{|Y|\times|Y|}$, где $C_{yy'}$ — штраф за отнесение объекта класса $y$ к классу $y'$;
— вектор информации $(I_j)_{j=1}^n$ о типах признаков;
— тестовую выборку в виде матрицы «объекты–признаки» $(x'_{ij})_{k\times n}$, где $x'_{ij}$ — значение $j$-го признака на $i$-м тестовом объекте $x'_i$;

и выдающая на выходе:

— вектор ответов $(y'_i)_{i=1}^k$ на тестовой выборке;
— матрицу оценок $(p'_{iy})_{k\times|Y|}$ принадлежности каждого тестового объекта каждому из классов.

Оценки принадлежности могут быть апостериорными вероятностями (в байесовских классификаторах), или просто значениями дискриминантных функций классов. Если алгоритм не вычисляет вещественных оценок принадлежности, то полагается $p'_{iy} = [y'_i = y]$.

## Методика тестирования

**Процедура скользящего контроля.** Производится $N$ разбиений выборки $X = \{x_1, \ldots, x_L\} = X_n^\ell \sqcup X_n^k$ на обучающую подвыборку длины $\ell$ и контрольную длины $k = L - \ell$, где $n = 1, \ldots, N$ — номер разбиения. Обозначим через $a_n \colon X \to Y$ *функцию классификации*, получаемую при $n$-м разбиении в результате обучения по выборке $X_n^\ell$.

*Оценка скользящего контроля* для произвольной функции от разбиения $\xi(n)$ определяется как среднее $\hat{\mathsf{E}}\xi = \frac{1}{N}\sum_{n=1}^{N}\xi(n)$. Разбиения строятся по стандартной методике $t\times q$-fold cross-validation [7, 8]: генерируется $t$ случайных разбиений выборки $X^L$ на $q$ блоков примерно равной длины и пропорциональными долями классов, и каждый блок поочерёдно становится контрольной выборкой. Таким образом, $N = tq$ и $k = \frac{L}{q}$, с точностью до округления. Каждый объект $x_i \in X^L$ выступает $t$ раз в роли контрольного и $(t-1)q$ раз в роли обучающего. При достаточно больших $t$ это позволяет строить доверительные интервалы для случайной величины $\xi(n)$ с помощью порядковых статистик, не делая никаких дополнительных предположений о виде распределения $\xi$.

Далее рассматриваются методы оценивания качества, реализованные в системе «Полигон», и приводятся примеры интерпретации результатов. Для иллюстрации взят алгоритм SVM и медицинская задача `Liver_Disorders` из репозитория UCI — разделение пациентов с нарушением работы печени (145 объектов класса 1) и здоровых людей (200 объектов класса 2). Число признаков равно 6.

**Средняя частота ошибок** на обучении $\nu^\ell = \hat{\mathsf{E}}\nu(a_n, X_n^\ell)$ и на контроле $\nu^k = \hat{\mathsf{E}}\nu(a_n, X_n^k)$ используется в эмпирических исследованиях чаще всего. В данной задаче $\nu^\ell = 22\pm5\%$, $\nu^k = 27\pm11\%$. Поверхностный анализ на таких оценках, как правило, и завершается. Однако уже простое разделение частоты по классам

класс 1, частота ошибок (обуч./конт.): 36%/43%;
класс 2, частота ошибок (обуч./конт.): 12%/18%;

выявляет важную особенность задачи: распознать больного намного труднее, чем здорового.

В «Полигоне» наряду с доверительными интервалами строятся графики эмпирических распределений частот ошибок на обучении и на контроле, а также *переобученности* $\delta_n = \nu(a_n, X_n^k) - \nu(a_n, X_n^\ell)$. Примеры таких графиков можно найти в [1].

**Анализ вариации и смещения** (bias-variance) [8]. Вводится функция *среднего предсказания* $\tilde{y}(x)$ — это класс, к которому объект $x_i \in X$ относится большинством функций $a_n$, $n = 1, \ldots, N$. *Смещение* на объекте $x_i$ определяется как $B(x_i) = [\tilde{y}(x_i) \neq y_i]$. Соответственно, объекты выборки $X$ разделяются на *смещённые* ($B(x_i) = 1$) и *несмещённые* ($B(x_i) = 0$). Смещённость означает, что объект плохо описывается данной моделью классификации (под моделью понимается параметрическое семейство алгоритмов).

*Вариация* $V(x_i)$ на объекте $x_i$ определяется как доля разбиений $n$, при которых $a_n(x_i) \neq \tilde{y}(x_i)$. Эта величина характеризует изменчивость ответов на данном объекте по отношению к составу обучающей выборки. Неустойчивы, как правило, классификации объектов, находящихся вблизи границы классов. Поэтому число объектов с наибольшими вариациями (близкими к $\frac{1}{2}$) характеризует толщину пограничного слоя между классами. Чем тоньше этот слой, тем «правильней» модель классификации подобрана под задачу.

Заметим, что выявление пограничных объектов в многомерных пространствах признаков является нетривиальной задачей. «Полигон» решает эту задачу универсально, независимо от природы задачи и конструкции алгоритма; при этом список «пограничных» объектов может быть выдан пользователю в явном виде.

Суммирование смещений и вариаций по объектам даёт характеристики, называемые *смещением* и *вариацией выборки*. В сумме они составляют среднюю частоту ошибок классификации. Большая величина смещения говорит о том, что модель классификации, возможно, выбрана неудачно, и для решения данной задачи следует искать другой алгоритм. Большая величина вариации говорит о том, что результат обучения слишком сильно зависит от состава выборки; в таких случаях качество классификации можно улучшить, оставаясь в рамках той же модели классификации, путём регуляризации или композиции алгоритмов.

Объекты, которые оказываются смещёнными относительно большого числа различных алгоритмов, можно считать шумовыми выбросами. Таким

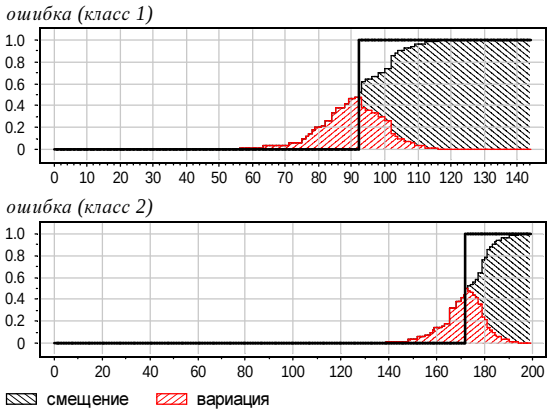


**Рис. 1.** Анализ вариации и смещения, по классам.

образом, «Полигон», при одновременном использовании большого числа разнообразных алгоритмов, позволяет идентифицировать выбросы более объективно, не привязываясь к какой-либо конкретной модели классификации.

«Полигон» позволяет анализировать разложение ошибки на смещение и вариацию отдельно по классам, рис. 1. По оси абсцисс отложены объекты, отсортированные по возрастанию ошибки на них. По оси ординат отложены значения вариации и ошибки на объектах. На несмещённых объектах ошибка состоит только из вариации, на смещённых — из разности смещения и вариации [4]. Видно, что у класса 1 (больные) пограничная зона больши́х вариаций намного шире, чем у второго класса. Опять-таки, это означает, что первый класс отделить труднее.

**Кривая ошибок** (ROC-кривая, receiver operating characteristic) используется для представления результатов классификации в тех случаях, когда соотношение цены ошибок I и II рода заранее неизвестно [6]. Предполагается, что имеется два класса: «положительный» и «отрицательный». По оси X откладывается доля *ошибочных положительных* классификаций, по оси Y — доля *правильных положительных* классификаций. Собственно, кривая получается в результате варьирования порога $\theta$ в функции классификации вида $a(x) = \text{sign}(f(x) - \theta)$, где $f(x)$ — вещественная дискриминантная функция.

Для многоклассовой задачи в роли положительного выступает каждый из классов по очереди, все остальные объединяются в один отрицательный класс; в результате строится серия из $|Y|$ ROC-кривых. Сопоставление ROC-кривых разных классов позволяет судить о том, какие алгоритмы, и при каких соотношениях цены ошибок, более целесообразно применять.

Чем больше *площадь под кривой* (AUC, area under curve), тем выше качество классификации. Критерий AUC является оценкой качества классификации, не зависящей от выбора соотношения цены ошибок. Различие между ROC-кривыми на обучении и контроле (рис. 2) позволяет судить о величине переобучения. Возможны ситуации, когда переобучение существенно различается в левой-нижней и в правой-верхей ветвях кривой; в таких случаях можно давать рекомендации об использовании данного алгоритма классификации только при определённых соотношениях цены ошибок.

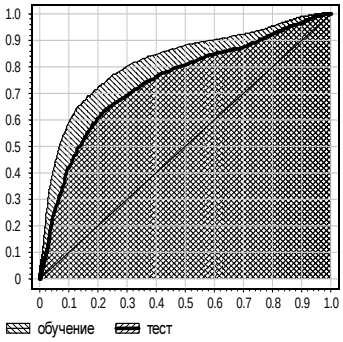


**Рис. 2.** Кривая ошибок.

**Распределение отступов.** Понятие *отступа* (margin) $m(x_i)$ объекта $x_i$ определено для алгоритмов, формирующих оценки принадлежности классам, $m(x_i) = p_{iy_i} - \max\limits_{y \neq y_i} p_{iy}$. В зависимости от значения $m(x_i)$ объекты разделяются на четыре типа: *шумовые* ($m \ll 0$), *пограничные* ($m \approx 0$), *неинформативные* ($m > 0$), *эталонные* ($m \gg 0$). В прикладных задачах такая типизация объектов имеет, как правило, самостоятельную ценность.

«Полигон» строит распределения отступов [5], усредняя их по разбиениям $n$. Отдельно строятся распределения отступов только по обучающим и только по контрольным объектам, что позволяет оценивать величину переобучения и число пограничных объектов в «зоне неуверенной классификации», рис. 3. Также строятся распределения отдельно по классам. По оси абсцисс откладываются объекты, упорядоченные по возрастанию среднего отступа; по оси ординат — значения отступов.

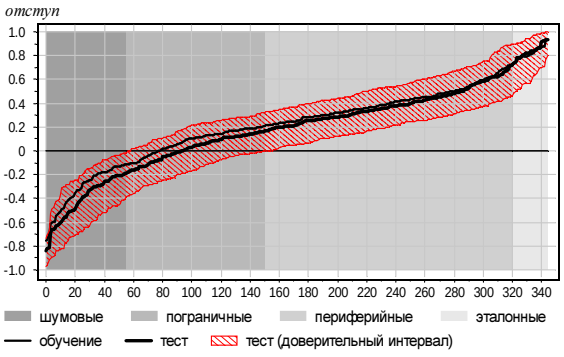


**Рис. 3.** Распределение отступов.

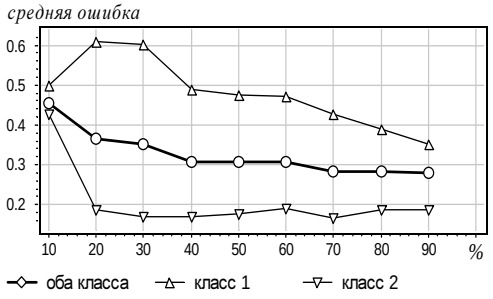


**Рис. 4.** Кривая обучения.

**Кривая обучения** (learning curve) в стандартном варианте — это зависимость средней частоты ошибок на контроле от длины обучающей выборки. «Полигон» дополняет стандартную методику построением кривых обучения отдельно по каждому классу с наложением их на одном графике, рис. 4. Здесь для класса 2 приемлемое качество классификации достигается, начиная с длины обучения 20% и далее сохраняется на постоянном уровне. Для класса 1 средняя ошибка снижается медленно, не успевая «выйти на насыщение». Отсюда можно сделать вывод, что для повышения качества классификации на объектах класса 1 (больные) необходимо существенно увеличить число обучающих объектов, причём только класса 1.

## Архитектура системы

Распределённая система «Полигон» состоит из *Центрального Сервера* (ЦС), который хранит репозиторий задач, результаты тестирования, индивидуальные настройки пользователей и отчетов, и *Вычислительных Серверов* (ВС), обеспечивающих работу алгоритмов. ВС принимают от центрального сервера задания на решение задач классификации и возвращают результаты работы алгоритмов. Функции ВС может выполнять любой компьютер в сети Интернет, на котором установлена специальная программа — *Менеджер ВС*, осуществляющая запуск алгоритмов и обмен данными с ЦС. Эта программа предоставляется разработчиками «Полигона». На одном вычислительном сервере может работать несколько алгоритмов. Любой зарегистрированный пользователь может установить Менеджер ВС на свой компьютер, реализовать один или несколько алгоритмов и объявить свой ВС в системе «Полигон».

Начальный набор задач формируется из репозитория UCI [3] и других общедоступных источников данных. Пользователи имеют возможность загружать в систему свои задачи и устанавливать на них права доступа.

Один раз вычисленные результаты тестирования сохраняются в центральной базе данных «Полигона», и при повторном запросе выдаются без обращения к алгоритмам. Во большинстве случаев это позволяет получать отчёты очень быстро. Когда алгоритм обновляется, его сохранённые результаты стираются.

## Добавление алгоритмов

Для разработчиков новых алгоритмов под платформой *.NET* предоставляется библиотека классов, содержащая основные структуры данных и базовый класс алгоритма, требующий написания двух функций — обучение и контроль. На сайте проекта имеются подробно документированные примеры реализации алгоритмов. Разработчикам алгоритмов на других языках программирования предоставляются специальные «обёртки» для реализации алгоритмов в виде *dll* (как *.NET*, так и *native*), исполняемых *exe*-файлов, web-сервисов, *matlab*-функций, и т. п. В частности, адаптация уже существующего алгоритма заключается в настройке или доработке одной из возможных «обёрток», исходный код которых находится в открытом доступе и подробно документирован.

## Выводы

Методика тестирования, реализованная в системе «Полигон», существенно расширяет и унифицирует известные методы анализа качества классификации, основанные на скользящем контроле. Она позволяет не только констатировать тот или иной уровень ошибок, но и выявлять их причины, идентифицировать шумовые и пограничные объекты (независимо от типа алгоритма), целенаправленно подбирать алгоритм под конкретную задачу.

## Литература

[1] *Ботов П. В.* Точные оценки вероятности переобучения для монотонных и унимодальных семейств алгоритмов // Всеросс. конф. ММРО-14 — М.: МАКС Пресс, 2009 — С. **??**–**??** (в настоящем сборнике).

[2] *Воронцов К. В., Инякин А. С., Лисица А. В.* Система эмпирического измерения качества алгоритмов классификации // Всеросс. конф. ММРО-13. — М.: МАКС Пресс, 2007. — С. 577–581.

[3] *Asuncion A., Newman D. J.* UCI Machine Learning Repository // University of California, Irvine. — 2007. `www.ics.uci.edu/~mlearn/MLRepository.html`.

[4] *Domingos P.* A unified bias-variance decomposition and its applications: ICML'17. — 2000. — Pp. 231–238.

[5] *Garg A., Roth D.* Margin distribution and learning algorithms: ICML'03. Washington, DC USA. — 2003. — Pp. 210–217.

[6] *Hand D., Till R.* A simple generalization of area under the ROC curve for multiple class classification problems // Machine Learning, 45. — 2001. — Pp. 171–186.

[7] *Langley P.* Crafting papers on machine learning // In Proceedings of ICML'2000. Pp.1207–1216.

[8] *Webb G. I.* MultiBoosting: A technique for combining boosting and wagging // Machine Learning. — 2000. — Vol. 40, No. 2. — Pp. 159–196.