Частичное Обучение с Учителем на Небольших Исходных Выборках

Сергей Завалишин, Илья Сафонов
Национальный Исследовательский Ядерный Университет «МИФИ», Москва, Россия
{Etti.Nekov, ilia.safonov} at gmail dot com

Аннотация

В данной работе предлагается алгоритм частичного обучения с учителем, эффективно работающий в условиях использования экстремально малой исходной обучающей выборки, состоящей из небольшого количества заранее классифицированных объектов, и большого, но заранее неизвестного количества неклассифицированных объектов. Рассматривается проблема пошагового обучения, когда число объектов в обучающей выборке не является постоянным и увеличивается по ходу обучения. Тестирование метода производится с помощью изображений, скачанных с сервиса Flickr, которые не отбирались вручную, что ведёт к появлению ошибок в исходной обучающей выборке. Для устранения этих ошибок и повышения эффективности обучения предлагается использовать ряд специальных методик: перемаркировку голосованием, фиксацию результатов и фильтрацию данных.

Ключевые слова: *частичное обучение с учителем, самообучение, совместное обучение, активное обучение.*

1. ВВЕДЕНИЕ

В настоящее время персонализированный поиск всё глубже входит в нашу жизнь. Множество веб-сервисов запускают механизмы анализа пользовательских предпочтений для предоставления своим пользователям наиболее интересных материалов, в том числе и рекламного характера, что обуславливает необходимость внедрения систем автоматического обучения на основе собранных данных. В случае, когда объём этих данных достаточно велик, обучение не представляет каких-либо проблем, однако в реальности, как правило, количество доступной информации о каждом конкретном пользователе ограничено, особенно это касается новых пользователей, которые только недавно начали посещать какой-либо веб-сервис. В подобных случаях [15] эффективным является использование методов частичного обучения с учителем, когда для успешного обучения требуется наличие лишь небольшого числа заранее разбитых на классы объектов (соответствующих предпочтениям пользователя) и большое число объектов, относящихся к неизвестным классам. Очевидно, что база данных любого веб-сервиса, полностью удовлетворяет данным требованиям, что заметно упрощает процесс интеграции системы обучения подобного типа. Тем не менее, несмотря на кажущуюся простоту данного подхода, имеется и ряд проблем, которые требуют решения.

Современные алгоритмы частичного обучения с учителем рассчитаны на использование обучающих выборок заранее известного размера, однако базы данных веб-сервисов, как правило, содержат миллионы объектов, что значительно увеличивает время обучения. Можно было бы ограничиться некоторым заранее определённым числом объектов в обучающей выборке, однако исследования показывают, что с ростом числа объектов улучшается и качество работы обученного классификатора. В данной работе предлагается пошаговый метод частичного обучения с учителем, при котором объём данных растёт постепенно, а процесс обучения заканчивается в тот момент, когда дальнейшее увеличение объёма выборки перестаёт иметь смысл.

Вторая проблема заключается в том, что исходная выборка, состоящая из промаркированных объектов и отражающая предпочтения пользователя, может иметь ошибки – промаркированные объекты, которые не относятся ни к одному из имеющихся в системе классов. Известно, что ошибки подобного рода негативно влияют на качество обучения, поэтому устранение их влияния является одной из важных задач.

Наконец, третья проблема касается объёмов исходных (промаркированных вручную человеком) обучающих выборок. Как правило, в исследовательских работах рассматривается ситуация, когда исходная выборка состоит из нескольких сотен объектов, однако в реальности она может содержать всего лишь несколько десятков объектов, что значительно затрудняет использование алгоритмов частичного обучения с учителем.

Для решения этих проблем нами предлагается ряд специальных методов: перемаркировка голосованием, фиксация результатов и фильтрация данных.

2. ОБЗОР СУЩЕСТВУЮЩИХ ПУБЛИКАЦИЙ

Существует множество алгоритмов частичного обучения с учителем [15]. В рамках данной работы было принято решение сфокусироваться на наиболее популярных из них: самообучении (self-trainnig) [9] и совместном обучении (co-training) [1].

Основная идея первого из них состоит в следующем: первоначально классификатор обучается на некоторой исходной выборке, состоящей из небольшого числа промаркированных (классифицированных) человеком объектов. Затем обученный классификатор уже без помощи человека маркирует другой – больший – набор объектов, классы которых неизвестны. Два полученных набора объектов, каждый из которых теперь стал промаркированным, объединяются вместе, формируя тем самым новую обучающую выборку, на которой классификатор обучается повторно. После этого классификатор заново маркирует все объекты в объединённой обучающей выборке и снова переобучается на ней. Процесс проходит до тех пор, пока маркировка отдельных объектов не перестанет меняться.

Метод совместного обучения работает, в целом, аналогичным образом, однако вместо одного классификатора на одном и том же наборе данных параллельно обучаются два, а перемаркировка отдельных объектов осуществляется либо за счёт голосования между классификаторами, цель которого – выбор наиболее верной метки, либо через использование двух независимых обучающих выборок с разными метками, когда один классификатор маркирует обучающую выборку другого.

Одним из хорошо известных алгоритмов частичного обучения с учителем является алгоритм Яровского [5]. Существует ряд модификаций данного алгоритма, позволяющих увеличить качество обучения.

Ряд работ [4, 7] указывает на то, что увеличение количества немаркированных объектов в обучающей выборке позволяет достичь лучшего качества работы классификатора при использовании алгоритмов частичного обучения с учителем, однако данный подход эффективен только в том случае, если модель данных, полученная за счёт обучения, оказывается близка к реальным данным, чего бывает трудно достичь при использовании малой исходной выборки. Для решения данной проблемы рекомендуется использовать фильтрацию шумов.

Так же для увеличения результативности обучения часто используется метод активного обучения [12, 13], который производит перемаркировку объектов не в полностью автоматическом режиме, а с участием человека: в процессе перемаркировки для некоторых объектов, в которых классификатор не уверен, требуется ручной ввод меток классов. Данный подход хорошо согласуется с идеей пошагового обучения, поэтому он, наряду с другими методами, используется для улучшения качества классификатора.

3. СРЕДА ОБУЧЕНИЯ

3.1. Модификация LIBSVM

Существует множество подходов к машинному обучению, таких как нейронные сети, наивный Байес и прочих, однако в рамках данной работы используется машина опорных векторов (далее SVM – Support Vector Machine), так как она одновременно обеспечивает достаточное быстродействие и точность [6] классификации. В данной работе SVM реализована с помощью открытой библиотеки LIBSVM [3].

Основная задача машины опорных векторов – это построение оптимальной разделяющей гиперплоскости с наибольшим зазором (margin) между классами в пространстве векторов с размерностью равной количеству признаков.

Для нахождения максимальной разделяющей гиперплоскости производится решение задачи оптимизации следующего вида:

$$\begin{cases} \mathcal{L}(w, w_0; \lambda) = \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{l} \lambda_i (y_i(\langle w, x_i\rangle - w_0) - 1) \longrightarrow \min_{w,w_0} \max_{\lambda}; \\ \lambda_i \geq 0, i = 1, \ldots, l; \\ \lambda_i = 0, \text{либо } \langle w, x_i\rangle - w_0 = y_i,\ i = 1, \ldots, l \end{cases} \quad (1)$$

Где $x_i=(x_i^1,\ldots,x_i^n)$ – признаковое описание объекта x_i;

$w=(w^1,\ldots,w^n) \in \mathbb{R}^n$ и $w_0 \in \mathbb{R}$ – параметры алгоритма; $y_i=y^*(x_i)$ – целевая зависимость y^*: $X \to Y$, где X – пространство объектов, а Y – множество ответов; $\lambda=(\lambda_1,\ldots,\lambda_l)$ – вектор двойственных переменных.

Очевидно, что разделяющая гиперплоскость может быть найдена далеко не всегда, поэтому в таких случаях применяется так называемое ядерное преобразование, позволяющее строить нелинейные разделители. Основная идея состоит в замене всех скалярных произведений, использующихся при нахождении разделяющей гиперплоскости, функциями ядра – то есть скалярными произведениями в пространстве большей размерности, где уже может существовать оптимальная разделяющая гиперплоскость.

Нами используется машина опорных векторов, которая в качестве параметра использует значение веса C, влияющего на решение задачи оптимизации:

$$\min_{w,b,\xi} \ \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \quad (2)$$

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad (3)$$

$$\xi_i \geq 0,$$

где ϕ – функция, переводящая вектор x_i в пространство с большей размерностью (при этом $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ называется функцией ядра); $w=(w^1,\ldots,w^n) \in \mathbb{R}^n$ и $w_0 \in \mathbb{R}$ – параметры алгоритма; $y_i=y^*(x_i)$ – целевая зависимость y^*: $X \to Y$, где X – пространство объектов, а Y – множество ответов; $\xi=(\xi_1,\ldots,\xi_l)$ – вектор двойственных переменных.

Параметр $C > 0$ определяет уровень ошибки и используется для отсечения заведомо некорректно классифицированных объектов. В рамках данной работы используется значение C = 1, так как отсечение производится другими методами.

Современные исследования показывают, что использование ядра χ^2 позволяет увеличить качество классификатора:

$$k(x, y) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)} \quad (5)$$

При этом экспоненциальная форма данного ядра χ^2_{exp}; оказывается ещё лучше:

$$k(x, y) = \exp\left(-\gamma \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)} \right) \quad (6)$$

По умолчанию LIBSVM поддерживает несколько различных ядер, но поддержки ядра χ^2 в этой библиотеке нет. Существуют различные решения от сторонних разработчиков, добавляющие χ^2, однако они все нацелены на использование в среде MATLAB, что требует наличия соответствующей программной среды или её компонентов. Для решения данной проблемы в библиотеку LIBSVM была включена поддержка ядер χ^2 и χ^2_{exp}.

Чтобы определить наиболее подходящее решаемой задаче ядро был проведён ряд тестов на выборке следующего объёма – 700 изображений для обучения и 100 для проверки, соответствующих двум классам сцен: внутри и вне помещений, скачанных с сервиса Flickr. При таком соотношении разница в результатах, показываемых различными ядрами, оказалась наиболее заметна. В качестве численных метрик качества классификации использовались следующие метрики:

$$Precision = \frac{TP}{TP+FP} \cdot 100\%;\ Recall = \frac{TP}{TP+FN} \cdot 100\%;$$

$$Accuracy = \frac{TP+TN}{N} \cdot 100\%; \quad (7)$$

Где N – общее число изображений, TP – число верных срабатываний, а FP и FN – число ложных срабатываний и пропусков, соответственно. При этом здесь и далее целевым классом являлся класс, соответствующий изображениям, содержащим сцены внутри помещений.

Как и ожидалось, ядро χ^2_{exp} показало наилучшие результаты (таблица 1).

Таблица 1: Сравнение различных ядер

Ядро	Accuracy, %	Precision, %	Recall, %
RBF	75	75	76
χ^2	79	76	84
χ^2_{exp}	**82**	**80**	**86**

3.2. Распознавание сцен внутри и вне помещений

Для распознавания сцен внутри и вне помещений мы использовали метод [8], который основывается на применении низкоуровневых цветовых и текстурных признаков изображений.

Обычно в качестве гистограммных признаков используются значения гистограммы различных каналов изображения с уменьшенным числом оттенков (до 16 градаций на канал). Наиболее очевидным вариантом является использование каналов RGB, однако такой подход не всегда эффективен: в RGB яркость не отделёна от цветового тона и насыщенности, что приводит к дублированию информации, так как для большого количества фотографий в этом цветовом пространстве вид каналов отличается незначительно

Мы использовали цветовое пространство LST [11]. Преобразование пиксела из пространства RGB в LST выглядит следующим образом:

$$L = \frac{k}{\sqrt{3}}(R+G+B), \qquad S = \frac{k}{\sqrt{2}}(R-B),$$
$$T = \frac{k}{\sqrt{6}}(R-2G+B) \quad (8)$$

Где L – канал освещённости, S и T отражают хроматическую информацию, а $k = 255/\max(R,G,B)$. Особенность S и T компонент состоит в том, что их значения не меняются с изменением интенсивности источника света. При этом канал S определяет вариации цветовой температуры от дневного света до искусственного, создаваемого лампой накаливания, что очень удобно для распознавания сцен внутри и вне помещений.

Для определения текстурных признаков используется двухуровневое wavelet-преобразование канала L при помощи фильтров Добеши, определяющихся следующими коэффициентами:

$$h \leftrightarrow [-0{,}129,\ 0{,}224,\ 0{,}837,\ 0{,}483],$$
$$g \leftrightarrow [-0{,}483,\ 0{,}837,\ -0{,}224,\ -0{,}129]$$

Где h – высокочастотный фильтр, а g – низкочастотный.

Текстурные признаки $e_1, e_2, \ldots, e_8$ определяются по формуле (9):

$$e_k = \frac{1}{M_k N_k}\sum_{i=1}^{M_k}\sum_{j=1}^{N_k} c_k^2(i,j),\ k = 1, 2, \ldots, 8, \quad (9)$$

Где $c_k(i, j)$ представляют собой коэффициенты двухуровневого wavelet-преобразования изображения (рис.1).

Таким образом, общее количество низкоуровневых признаков изображения составляет 32: 24 цветовых признака (по 8 градаций на канал) и 8 текстурных.

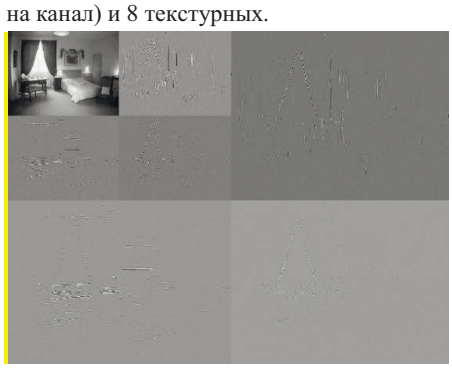

Рис 1: Пример двухуровневого wavelet-преобразования изображения.

4. ПРЕДЛАГАЕМЫЙ ПОДХОД К ЧАСТИЧНОМУ ОБУЧЕНИЮ С УЧИТЕЛЕМ

4.1. Общий принцип

При использовании сетевых хранилищ данных число объектов в обучающей выборке заранее неизвестно, что приводит к невозможности использовать итерационное обучение на одних и тех же данных в надежде на то, что каждый шаг будет улучшать качество распознавания классификатора. Так как данные добавляются небольшими порциями, то нельзя гарантировать того, что новая порция не ухудшит результаты. Соответственно, требуется применять различные дополнительные методы для предотвращения ухудшения характеристик классификатора.

Общий алгоритм обучения при использовании сетевых хранилищ можно представить следующим образом:

Алгоритм 1: Предлагаемый алгоритм пошагового обучения

1: Дано: $X^{(0)}$, $Y^{(0)}$
2: Для $t \in \{0, 1, \ldots\}$:
3: $\quad X^{(t+1)} := X^{(t)} \cup Z^{(t)}$
4: $\quad \Lambda^{(t)} = \{x \in X \mid Y^{(t)} \neq \perp\}$
5: $\quad$ Тренируем классификатор π^{t+1} на $(\Lambda^{(t)}, Y^{(t)})$
6: $\quad$ Для каждого вектора $x \in X$:
7: $\quad\quad \hat{y} := \arg max_j\ \pi_x^{(t+1)}(j)$
8: $\quad\quad Y_x^{(t+1)} := \begin{cases} Y_x^{(0)}, & \text{если } x \in \Lambda^{(0)} \\ \hat{y}, & \text{если } x \in \Lambda^{(t)} \vee \pi_x^{(t+1)}(\hat{y}) > \frac{1}{L} \\ \perp, & \text{иначе} \end{cases}$
9: $\quad$ Если $Y^{(t+1)} = Y^{(t)}$, то конец, иначе шаг 2

Где L – число возможных меток (иными словами – количество классов); $X^{(t)}$ – множество маркированных и немаркированных объектов на шаге t; $Z^{(t)}$ – порция немаркированных объектов, скачиваемых на шаге t; $\Lambda^{(t)}$ – множество маркированных объектов на шаге t; $Y^{(t)}$ – функция, производящая маркировку каждого объекта; $\pi_x^{(t+1)}(j)$– уверенность в принадлежности объекта x к классу j на шаге t+1.

Можно заметить, что вероятность выполнения условия $Y^{(t+1)} = Y^{(t)}$ достаточно мала, так как вряд ли наступит момент, когда ни одна из меток объектов на очередном шаге не изменится, поэтому данное условие имеет смысл заменить более мягким, например, когда процесс обучения заканчивается при достижении некоторого заданного процента неизменившихся меток. Можно расширить данный алгоритм для случая активного обучения, изменив шаг 8 следующим образом:

Алгоритм 2: Шаг, реализующий активное обучение

$$Y_x^{(t+1)} := \begin{cases} Y_x^{(0)}, & \text{if } x \in \Lambda^{(0)} \\ y, & \text{if } \pi_x^{(t+1)}(j) < \pi_{0x}, \\ \hat{y}, & \text{иначе} \end{cases}$$

где y – метка, выставляемая вручную человеком, и π_{0x} – минимальный порог для отступа, определяющий достаточную уверенность в метке.

4.2. Перемаркировка голосованием

Изначально предполагается, что на каждом новом шаге метки объектов могут изменяться произвольным образом, в зависимости от полученной на предыдущем шаге модели данных. Однако если в случае алгоритмов обучения, использующих заранее известные выборки, данный подход

обеспечивает определённую сходимость, то при порционном скачивании сходимость не гарантируется. К тому же может наблюдаться некоторый отрицательный эффект: нередко с каждым новым шагом качество работы классификатора при таком обучении только снижается.

Для предотвращения подобных проблем целесообразно использовать своего рода «память»: при маркировке объекта запоминается не только метка, но и её отступ (margin). В данном случае под термином «отступ» понимается «степень погруженности» объекта в свой класс, соответственно, чем меньше его значение, тем выше вероятность ошибки. В англоязычной литературе так же часто используется термин confidence (степень доверия). В случае если на последующем шаге отступ будет ниже сохранённого, то перемаркировка не производится. Это позволяет, с одной стороны, защитить классификатор от неверной пермаркировки, но с другой – может приводить к накоплению ошибок, когда неверно принятое решение в дальнейшем закрепляется под видом верного. Тем не менее, данной ситуации можно избежать при применении более интеллектуального способа выбора меток, использующего принцип голосования.

$$y_i = \frac{\sum_{j=1}^{n} p_{i,j} y_{i,j} + p_{i-1} y_{i-1}}{\sum_{j=1}^{n} p_{ij} + p_{i-1}}, \quad (10)$$

где y_i - новая метка объекта на i-м шаге; $y_{i,j}$ - метка объекта на i-м шаге, выставленная j-м классификатором; $p_{i,j}$ - отступ метки $y_{i,j}$; y_{i-1} - метка объекта на i - 1 шаге; p_{i-1} - отступ метки y_{i-1}.

Использование подобной схемы позволяет предотвратить накопление ошибок, реализуя при этом защиту от принятия неправильных решений в ходе маркировки.

4.3. Фиксация результатов

К сожалению, перемаркировка голосованием в чистом виде эффективна далеко не всегда, поэтому может применяться и более «жёсткий» способ сохранения достигнутого качества распознавания: в случае, если на очередной итерации работа классификатора улучшилась, производится фиксация данных: все метки, полученные на этом шаге, жёстко фиксируются без возможности дальнейшего изменения. Алгоритмически данный процесс выглядит следующим образом:

Алгоритм 3: Алгоритм пошагового обучения с фиксацией

1: Дано: $X^{(0)}$, $Y^{(0)}$,Λ_0
2: **Для** $t \in \{0, 1, \ldots\}$:
3: $X^{(t+1)} := X^{(t)} \cup Z^{(t)}$
4: $\Lambda^{(t)} = \{x \in X \mid Y^{(t)} \neq \perp\}$
5: Тренируем классификатор π^{t+1} на $(\Lambda_0 \cup \Lambda^{(t)}, Y^{(t)})$
6: **Для** каждого вектора $x \in X$:
7: $\hat{y} := \arg max_j \ \pi_x^{(t+1)}(j)$
8: $$Y_x^{(t+1)} := \begin{cases} Y_x^{(0)}, & \text{если } x \in \Lambda^{(0)} \\ \hat{y}, & \text{если } x \in \Lambda^{(t)} \vee \pi_x^{(t+1)}(\hat{y}) > \frac{1}{L} \\ \perp, & \text{иначе} \end{cases}$$
9: **Если** классификатор π^{t+1} лучше, чем π_0, **то** $\pi_0 := \pi^{t+1}$,
10: $\Lambda_0 := \Lambda_0 \cup \Lambda^{(t)}$
11: **Если** $Y^{(t+1)} = Y^{(t)}$, **то** конец, **иначе** шаг 2

где Λ_0 – множество объектов с фиксированными метками; π_0 – зафиксированный классификатор.

Как правило, не имеет смысла фиксировать сам классификатор, достаточно лишь сохранить параметры его работы, чтобы их можно было сравнивать с результатами других классификаторов. Подобными параметрами могут быть, например, accuracy, precision и recall.

4.4. Фильтрация Данных

Известно, что SVM весьма чувствительны к шумовым выбросам (неверно промаркированных объектам), поэтому фильтрация шумов является достаточно важным процессом, позволяющим заметно повысить эффективность работы машины опорных векторов. В задачах автоматического обучения ей часто пренебрегают, так как шум не оказывает заметного влияния на результаты. Однако при использовании итерационной модели обучения шумы могут негативно влиять на обучаемый классификатор, так как присутствует эффект накопления ошибок, описанный ранее (рис. 2).

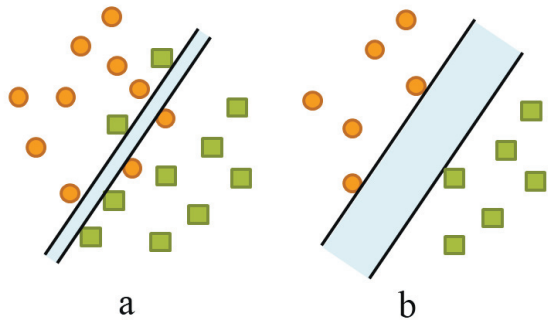

Рис 2: Фильтрация данных: a – без фильтрации, b – с фильтрацией.

Можно видеть, что при использовании фильтрации оптимальная разделяющая гиперплоскость, во-первых, будет шире, а во-вторых, число ошибочно классифицированных объектов уменьшится. Так же плюсом фильтрации является то, что увеличивается скорость обучения за счёт уменьшения числа объектов в обучающей выборке.

Наиболее простым, но при этом эффективным, методом отсечения шумов является отсечение по порогу: если отступ метки оказывается ниже некоторого минимума, то объект, соответствующий данной метке, удаляется из обучающей выборки. Однако при этом отсекается и часть полезной информации, так как низкое значение отступа лишь свидетельствует о высокой вероятности ошибки, но не гарантирует её наличия. Тем не менее, экспериментальные результаты показывают, что даже такой грубый подход оказывается лучше, чем полное отсутствие фильтрации.

5. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Для тестирования предложенного алгоритма пошагового обучения нами с сервиса Flickr были скачаны фотографии, соответствующие двум классам: со сценами внутри и вне помещений. Поиск фотографий производился по текстовым тегам ‘indoor’ и ‘architecture’ для изображений внутри помещений и по тегу ‘landscape’ вне помещений. Никакого ручного деления на классы в обучающей выборке не производилось. Для унификации процесса вычисления низкоуровневых признаков изображений все фотографии были масштабированы до размера 500x500 пикселов, при этом важным моментом является нормализация этих признаков: как было показано в [10], она позволяет улучшить качество распознавания. Однако так как в случае заранее неизвестного числа объектов в обучающей выборке

нормализовать все признаки достаточно затруднительно, было принято решение произвести лишь приблизительную нормализацию, когда часть значений может выходить за границы заданного диапазона [0, 1]. Для обеспечения корректности результатов тестирования нами был сформирован набор из 30,000 изображений двух классов. Исходная обучающая выборка (состоящая из фотографий, классы которых известны) состояла из 20 изображений (рисунок 4).

Рис 3: Изображения исходной обучающей выборки. Как можно видеть, некоторые из них (2-4) нельзя отнести ни к фотографиям, содержащим сцены внутри помещений, ни к фотографиям, содержащим сцены вне помещений.

При этом в данных наборах количество изображений обоих классов было одинаковым. Размер одной порции скачанных данных на каждом шаге алгоритма составлял 3500 изображений. Для оценки качества распознавания с помощью метрик accuracy, precision и recall использовался набор из 1000 фотографий, несвязанных с основной выборкой. Параметр веса C у SVM был равен C=1. При этом использовалось ядро χ^2_{exp} со значением параметра $\gamma = 0.5$.

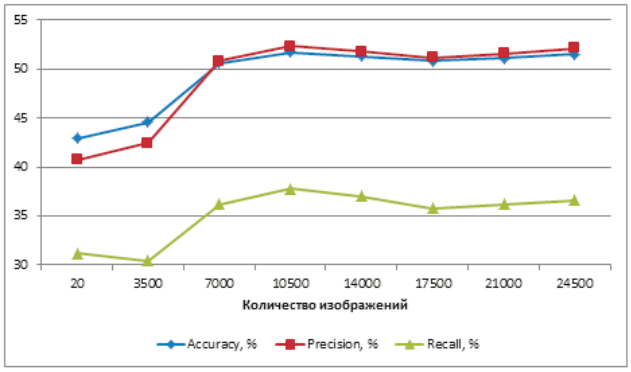

Рис 4: Пошаговое самообучение. Перемаркировка голосованием, фиксация результатов и фильтрация включены. Максимальные значения: accuracy = 51.7%, precision = 52.4%, recall = 37.8%.

На первом этапе мы протестировали наш алгоритм пошагового обучения с использованием метода самообучения. Порог фильтрации при этом был равен 0.6 (фильтрация производилась по параметру probability, использующемуся в LIBSVM как мера оценки уверенности принадлежности метки какому-либо классу). На первой итерации было отфильтровано порядка 20% изображений, на последующих – около 1%. Как можно видеть на рисунке 5, пошаговое самообучение оказалось эффективно на нескольких начальных итерациях обучения, однако в дальнейшем его эффективность снизилась.

Затем мы провели сравнение результатов, показываемых предлагаемым алгоритмом пошагового самообучения и обычным алгоритмом самообучения, реализованным на основе модифицированного алгоритма Яровского. Тестирование последнего проводилось следующим образом: исходная обучающая выборка, как и в случае пошагового самообучения, состояла из 20 фотографий, к которым затем добавлялось некоторое число неклассифицированных изображений, после чего проводилось обучение. Как можно видеть на рисунке 6, обычное самообучение, в отличие от пошагового, при данных условиях оказалось полностью неэффективно.

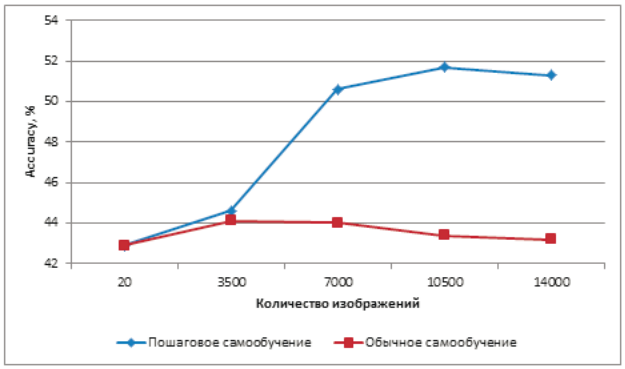

Рис 5: Сравнение обычного самообучения и пошагового самообучения. Максимальные значения:
Пошаговое самообучение: accuracy = 51.7%,
Обычное самообучение: accuracy = 44.1%.

Ситуация с совместным обучением была полностью иная: результаты пошагового и обычного совместного обучения оказались достаточно близки, при этом использование

дополнительных методов улучшения качества обучения, таких как перемаркировка голосованием, фиксация результатов и фильтрация, не несла практической пользы: работа классификатора при их использовании не улучшилась. Результаты пошагового совместного обучения показаны на рисунке 7.

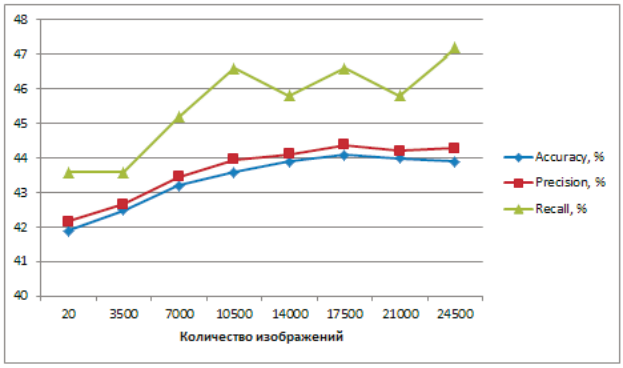

Рис 6: Пошаговое совместное обучение. Перемаркировка голосованием, фиксация результатов и фильтрация выключены. Максимальные значения: accuracy = 44.1%, precision = 44.4%, recall = 47.2%.

Наконец, мы протестировали пошаговое само- и совместное обучение, совместив его с методом активного обучения. При этом вручную маркировались только изображения, которые в противном случае были бы отфильтрованы по порогу 0.6, указанному ранее. Соответственно, фильтрация не использовалась. Наибольшая эффективность при использовании активного обучения была достигнута в случае пошагового совместного активного обучения. Можно заметить, что оно, в целом, оказалось значительно более результативным, чем во всех описанных ранее случаях (рисунок 8).

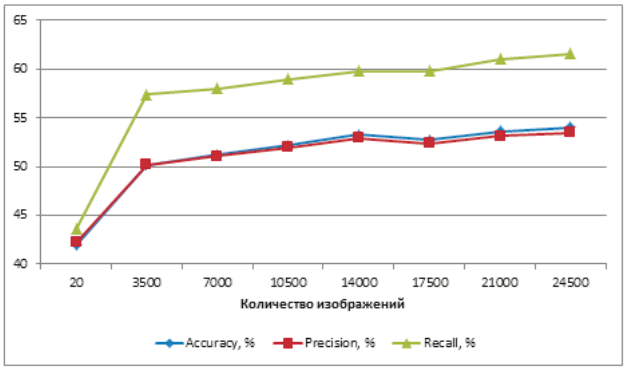

Рис 7: Пошаговое активное совместное обучение. Перемаркировка голосованием и фиксация результатов включены, фильтрация выключена. Максимальные значения: accuracy = 54.0%, precision = 53.5%, recall = 61.6%

Как можно видеть, в случае использования экстремально малых исходных выборок предложенные модификации алгоритмов пошагового обучения с учителем позволяют достичь лучших результатов, чем хорошо известные существующие алгоритмы.

6. ССЫЛКИ

[1] Avrim Blum, Tom Mitchell. “Combining labeled and unlabeled data with co-training”, COLT' 98 Proceedings of the eleventh annual conference on Computational learning theory, pp. 92 – 100, 1998.

[2] Ulf Brefeld, Thomas Gärtner, Tobias Scheffer, Stefan Wrobel. “Efficient co-regularised least squares regression”, ICML '06 Proceedings of the 23rd international conference on Machine learning, pp. 137 – 144, 2006.

[3] Chih-Chung Chang, Chih-Jen Lin. “LIBSVM – A Library for Support Vector Machines”, ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[4] George Forman, Ira Cohen. “Learning from Little: Comparison of Classifiers Given Little Training”, HP Laboratories Palo Alto, 2004.

[5] Gholamreza Haffari, Anoop Sarkar. “Analysis of semi-supervised learning with the Yarowsky algorithm”, 2007.

[6] Maji, S., Berg, A.C., Malik, J. “Classification using intersection kernel support vector machines is efficient”, Computer Vision and Pattern Recognition, pp. 1–8, 2008.

[7] Perronnin F., Sanchez J. and Yan Liu. “Large-scale image categorization with explicit data embedding”, Computer Vision and Pattern Recognition (CVPR). pp. 2297 – 2304, 2010.

[8] Ioannis Pratikakis, Basilios Gatos, Stelios C. A. Thomopoulos. “Scene categorization using low-level visual features”.

[9] Chuck Rosenberg, Martial Hebert, Henry Schneiderman. “Semi-supervised self-training of object detection models”, Seventh IEEE Workshop on Applications of Computer Vision, January, 2005.

[10] W. S. Sarle. “Neural Network FAQ”, 1997.

[11] Navid Serrano, Andreas E. Savakis, Jiebo Luo. “Improved scene classification using efficient low-level features and semantic cues”, Pattern Recognition, Volume 37, Issue 9, pp. 1773–1784, 2004.

[12] Burr Settles. “Active learning literature survey”, Computer Sciences Technical Report 1648, University of Wisconsin-Madison, Vol. 1648. 2010.

[13] Gokhan Tura, Dilek Hakkani-Türa, Robert E. Schapire. “Combining active and semi-supervised learning for spoken language understanding”, Speech Communication, Volume 45, Issue 2, pp 171–186, 2005.

[14] Vapnik V., Chapelle O. “Bounds on error expectation for support vector machines”, Neural Computation, Vol.12, no.9. P. 2013 – 2036, 2000.

[15] X. Zhu. “Semi-supervised learning literature survey”, Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005/9.

Об авторах

Сергей Завалишин – студент-дипломник НИЯУ «МИФИ».
Илья Сафонов – к.т.н., доцент НИЯУ «МИФИ».