Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 1 (22). 2017

УДК 004.93

ОБУЧЕНИЕ КАСКАДОВ ХААРА

Е. А. Белых

Данная статья посвящена каскадам Хаара и базируется на статье Viola P., Jones M. «Rapid Object Detection using a Boosted Cascade of Simple Features». Здесь описаны некоторые тонкости обучения каскадов, которые не были описаны в оригинальной статье. В частности, это метод перебора порогов слабых классификаторов, а также оптимизированный метод построения каскада классификаторов.
Ключевые слова: распознавание образов, машинное обучение, классификация, обработка изображений.

1. Введение

Обученный каскад Хаара [1], принимая на вход изображение, определяет, есть ли на нем искомый объект, т. е. выполняет задачу классификации, разделяя входные данные на два класса (есть искомый объект, нет искомого объекта).

Правильно обученный каскад Хаара имеет хорошую скорость выполнения классификации, а также неплохую устойчивость к разного рода отклонениям. Изначально данный способ был предназначен для обнаружения лиц, однако его можно использовать и для обнаружения других объектов.

2. Признаки Хаара

Признак Хаара является набором прямоугольных областей изображения, примыкающих друг к другу и разделенных на две группы. Возможных признаков Хаара огромное множество (разнообразные комбинации областей разной ширины и высоты с разными позициями на изображении). Первоначальный набор признаков зависит от реализации и конкретной задачи.

Чтобы вычислить значение конкретного признака Хаара для какого-либо изображения, надо сложить яркости пикселей изображения в первой и второй группах прямоугольных областей по отдельности, а затем вычесть из первой полученной суммы вторую. Полученная разность и есть значение конкретного признака Хаара для данного изображения:

$$a_i = \sum_{j=y_{a_i}}^{y_{a_i}+h_{a_i}-1} \sum_{k=x_{a_i}}^{x_{a_i}+w_{a_i}-1} v_{jk}, \qquad b_i = \sum_{j=y_{b_i}}^{y_{b_i}+h_{b_i}-1} \sum_{k=x_{b_i}}^{x_{b_i}+w_{b_i}-1} v_{jk}, \tag{1}$$

$$h(u) = \sum_{i=1}^{N_a} a_i - \sum_{i=1}^{N_b} b_i,$$

где v_{jk} — яркость пикселя с координатами $[j, k]$, a_i — сумма яркостей пикселей в i-й области первой группы, b_i — сумма яркостей пикселей в i-й области второй группы, h — значение признака Хаара для этого изображения, h_{a_i}, w_{a_i}, h_{b_i} и w_{b_i} — высота и ширина i-х областей первой и второй групп соответственно, y_{a_i} и x_{a_i}, y_{b_i} и x_{b_i} — смещения по оси y и x i-х областей первой и второй групп, а N_a и N_b — количество областей в первой и второй группах.

3. Алгоритм AdaBoost

Для выбора признаков, лучше всего классифицирующих изображения, используется алгоритм *AdaBoost* (adaptive boosting). Этот алгоритм предложили Йовав Фройнд (Yoav Freund) и Роберт Шапир (Robert E. Schapire) [2]. Данный алгоритм построен на идее, что из большого числа простых способов классификации (называемых *слабыми классификаторами*) можно составить новый способ, выполняющий эту задачу намного эффективнее.

Слабый классификатор — это функция, которая принимает на вход изображение, вычисляет значение соответствующего ей признака Хаара для этого изображения и сравнивает это значение с порогом, возвращая либо 0, либо 1.

$$h_i(x) = \begin{cases} 1, & p_i f_i(x) < p_i \theta_i, \\ 0, & \text{иначе}, \end{cases} \tag{2}$$

где θ_i — порог, x — входное изображение, $f_i(x)$ — значение соответствующего признака Хаара для изображения x, p_i — направление знака неравенства, а $h_i(x)$ — слабый классификатор.

Данный алгоритм перебирает все возможные слабые классификаторы и выбирает те, которые допускают меньше всего ошибок. После выбора очередного слабого классификатора веса перераспределяются так, что неверно классифицированные изображения начинают сильнее влиять на значение ошибки. Ниже приведем полный алгоритм.

Входные данные:

h_i — i-й признак Хаара (из всех возможных);

E_i — i-й обучающий пример;

y_i — 0, если i-й обучающий пример отрицательный, и 1, если i-й обучающий пример положительный;

n — количество обучающих примеров.

Переменные:

w_i — вес, соответствующий i-му обучающему примеру;

θ_i и p_i — порог и направление знака неравенства для i-го признака Хаара, дающие наименьшую ошибку;

$\epsilon(h_i)$ — ошибка i-го слабого классификатора;

$\dot{h}_i$ — слабый классификатор, выбранный на i-й итерации;

β_i — минимальное значение ошибки слабого классификатора на i-й итерации, представленное в другой форме (для оптимизации вычислений и экономии места на бумаге).

Выходные данные (результат):

1. Инициализировать веса обучающих примеров: w_i, h_i, β_i.

$$\begin{cases} w_i = \frac{1}{2m}, & \text{если } y_i = 0, \\ w_i = \frac{1}{2l}, & \text{если } y_i = 1, \end{cases} \tag{3}$$

   где m — число отрицательных примеров, а l — число положительных примеров.

2. Для $t = 1, \ldots, C$:

(a) Нормализовать веса обучающих примеров:

$$w_i = \frac{w_i}{\sum_{j=1}^{n} w_j}. \tag{4}$$

(b) Найти для каждого признака Хаара такие параметры, чтобы слабый классификатор дал наименьшую ошибку:

$$\begin{gathered} \dot{h}(x) = \begin{cases} 1, & p_j h_j(x) < p_j \theta_j, \\ 0, & \text{иначе;} \end{cases} \\ \epsilon(\dot{h}) = \sum_{k=0}^{n} w_k \left| \dot{h}(E_k) - y_k \right| \text{— минимальный.} \end{gathered} \tag{5}$$

(c) Найти классификатор с наименьшей ошибкой:

$$\dot{h_t} = \begin{cases} 1, & p_n h_n(x) < p_n \theta_n, \\ 0, & \text{иначе,} \end{cases} ; \ \epsilon(\dot{h_t}) = \min \epsilon(h_j); \tag{6}$$

где n — номер слабого классификатора, дающего наименьшую ошибку.

(d) Обновить веса обучающих примеров:

$$\begin{gathered} \beta_t = \frac{\epsilon(\dot{h_t})}{1 - \epsilon(\dot{h_t})}; \\ w_i = w_i \beta_t^{\,1 - |\dot{h_t}(E_i) - y_i|}. \end{gathered} \tag{7}$$

3. Итоговый сильный классификатор:

$$H(x) = \begin{cases} 1, & \sum_{t=1}^{C} \log\left(\frac{1}{\beta_t}\right) \dot{h_t}(E_i) \geq \frac{1}{2} \sum_{t=1}^{T} \log\left(\frac{1}{\beta_t}\right), \\ 0, & \text{иначе.} \end{cases} \tag{8}$$

Полученный сильный классификатор уже способен выполнять задачу классификации, хоть и очень медленно. Значение $a_i = \log \frac{1}{\beta_i}$ далее будет рассматриваться как «коэффициент i-го слабого классификатора».

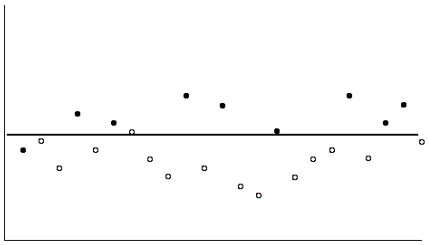

Рис. 1. Значения признака Хаара

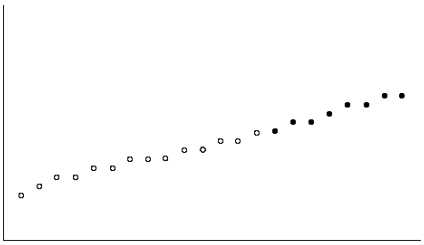

Рис. 2. Отсортированные значения признака Хаара

4. Вычисление порога слабого классификатора

Перебор всех возможных признаков Хаара выполняется за приемлемое время. Однако нахождение наилучшего порога и направления знака неравенства таким же способом требует такого количества времени, что применение на практике каскадов Хаара становится затруднительным (так как порог — число вещественное, количество его возможных значений ограничено лишь особенностями представления вещественных чисел на конкретной машине).

Если предположить, что веса всех обучающих примеров равны, то порог должен разделять плохие и хорошие примеры по значению признака Хаара таким образом, чтобы с одной стороны было как можно больше хороших, а с другой — как можно больше плохих (с какой стороны должны быть какие примеры, определяет направление знака неравенства). Но так как веса обычно разные, то задачу надо изменить так: сумма весов примеров, находящихся не по ту сторону порога, по которую им следует находиться, должна быть минимальной.

На рис. 1 изображены значения одного признака Хаара для разных обучающих примеров: по горизонтальной оси расположены номера обучающих примеров, по вертикальной — значения признака Хаара для соответствующих обучающих примеров. Закрашенные кружки — значение признака Хаара для хороших примеров, незакрашенные кружки — для плохих. Горизонтальная линия на графике — один из возможных порогов.

Отсортировав обучающие примеры по их значению признака Хаара, можно получить картинку, как на рис. 2. Теперь можно заметить, что все пороги, находящиеся между двумя соседними точками на отсортированном графике, дают одинаковые результаты. Следовательно, чтобы получить наименьшую возможную ошибку, достаточно проверить только $n - 1$ порогов, где n — количество обучающих примеров,

беря как значение порога, например, среднее между двумя соседними отсортированными значениями признака Хаара.

Однако даже после этого перебор признаков Хаара занимает очень много времени. Одной из причин является вычисление ошибки слабого классификатора, требующее большое количество операций суммирования и выполняющееся огромное количество раз.

К счастью, есть способ вычислять ошибку слабого классификатора, не выполняя столько операций суммирования. Для этого можно использовать трюк, очень похожий на интегральную форму представления изображения:

h — признак Хаара, для которого ищется порог;

E_i — i-й обучающий пример;

y_i — 0, если i-й обучающий пример отрицательный, и 1, если i-й обучающий пример положительный;

n — количество обучающих примеров;

w_i — вес, соответствующий i-му обучающему примеру;

w^+, w^- — дополнительные массивы;

$\dot{\theta}$ — порог, проверяемый на текущей итерации;

$\dot{\epsilon}$ — ошибка на текущей итерации;

ϵ — наименьшая полученная ошибка, изначально равна 1;

θ и p — значение порога и направление знака неравенства, при которых была получена наименьшая ошибка.

1. Отсортировать обучающие примеры. Веса и элементы массива y расположить в таком же порядке, как и отсортированные примеры (т. е. чтобы каждому примеру после сортировки соответствовал тот же вес и элемент из y, что и до сортировки).

2. Сделать два дополнительных массива, в первом массиве i-й элемент содержит сумму весов хороших примеров до i-го значения, а во втором — плохих:

$$\begin{aligned} w_i^+ &= \sum_{j=1}^{i} \begin{cases} w_j, \text{ если} & y_j = 1, \\ 0, \text{ если} & y_j = 0; \end{cases} \\ w_i^- &= \sum_{j=1}^{i} \begin{cases} w_j, \text{ если} & y_j = 0, \\ 0, \text{ если} & y_j = 1. \end{cases} \end{aligned} \tag{9}$$

3. Для всех i от 2 до n:

   (a) Вычислить значение проверяемого порога:

   $$\dot{\theta} = \frac{h(E_{i-1}) + h(E_i)}{2}. \tag{10}$$

   (b) Посчитать ошибку для $p = 1$:

   $$\dot{\epsilon} = w_{i-1}^{-} + w_n^{+} - w_{i-1}^{+}. \tag{11}$$

   (c) Если ошибка проверяемого порога меньше текущей минимальной ошибки, запомнить этот порог, ошибку и направление знака неравенства:

   $$\text{если } \dot{\epsilon} < \epsilon, \text{тогда } \epsilon = \dot{\epsilon}, \theta = \dot{\theta}, p = 1. \tag{12}$$

   (d) Посчитать ошибку для $p = -1$:

   $$\dot{\epsilon} = w_{i-1}^{+} + w_n^{-} - w_{i-1}^{-}. \tag{13}$$

   (e) Если ошибка проверяемого порога меньше текущей минимальной ошибки, запомнить этот порог, ошибку и направление знака неравенства:

   $$\text{если } \dot{\epsilon} < \epsilon, \text{тогда } \epsilon = \dot{\epsilon},\ \theta = \dot{\theta},\ p = -1. \tag{14}$$

5. Каскад классификаторов

Если уменьшать значение порога *сильного* классификатора, уменьшается количество ложных негативных срабатываний (неверно классифицированных хороших примеров) и увеличивается количество ложных позитивных (неверно классифицированных плохих примеров). Таким образом, даже если сильный классификатор состоит из малого количества слабых классификаторов, понижая порог ценой большего числа ложных позитивных срабатываний, можно свести к минимуму количество ложных негативных.

Используя это свойство, из найденных слабых классификаторов можно составить *каскад*, являющийся набором сильных классификаторов (называемых стадиями), через которые последовательно проходит проверяемое изображение.

Ниже приведен алгоритм, описанный в статье Пола Виолы и Майкла Джонса.

C — число доступных слабых классификаторов;

C_i — число слабых классификаторов на i-й стадии;

Θ_i — порог на i-й стадии;

s — номер текущей стадии;

ϕ — заданное наперед значение;

P_i и N_i — доля ложных позитивных и ложных негативных срабатываний i-й стадии соответственно;

a_i — коэффициент i-го слабого классификатора.

1. Установить текущей стадией первую, сделав число слабых классификаторов на ней равным 1:

$$s = 1,\ C_s = 1. \tag{15}$$

2. Установить для текущей стадии такой порог, чтобы все положительные примеры классифицировались верно:

$$\text{установить } \Theta_s \text{ такой, что } N_s = 0. \tag{16}$$

3. Если доля ложных позитивных срабатываний текущей стадии больше ϕ, увеличить число слабых классификаторов на ней, в противном случае сделать текущей следующую стадию, установив на ней число слабых классификаторов равным числу слабых классификаторов предыдущей, увеличенному на 1:

$$\begin{gathered}\text{если } P_s > \phi, \text{тогда } C_s = C_s + 1,\\ \text{иначе } s = s + 1,\ C_s = C_{s-1} + 1.\end{gathered} \tag{17}$$

4. Если число слабых классификаторов на текущей стадии меньше числа доступных слабых классификаторов или доля ложных положительных срабатываний равна 0, перейти к пункту 2:

$$\text{если } C_s < C \text{ или } P_s = 0,\ \text{перейти к пункту 2.} \tag{18}$$

5. Вернуть стандартный порог последней стадии, если выход из цикла произошел из-за того, что закончились доступные слабые классификаторы:

$$\text{если } P_s > 0, \text{тогда } \Theta_s = \frac{1}{2}\sum_{i=1}^{C_s} a_i. \tag{19}$$

6. Быстрое построение каскада

Во втором пункте указанного выше алгоритма подразумевается, что требуемый порог ищется перебором. На практике это не очень эффективно.

Однако с помощью упрощенного варианта оптимизации, использовавшейся для нахождения порогов слабых классификаторов, можно свести затраты времени на построение каскада к минимуму.

Для того чтобы применить этот метод, надо изменить условия, при которых порог считается наилучшим. Также следует учесть, что у порога сильного классификатора фиксированное направление знака неравенства ($>$).

Для слабого классификатора порог считался наилучшим, если сумма весов неверно классифицированных примеров была минимальной. В сильных же классификаторах наилучший порог тот, при котором верно классифицируются все хорошие примеры, а также максимальное число отрицательных.

Значение, сравниваемое с порогом сильного классификатора, является суммой коэффициентов тех классификаторов, которые верно классифицировали входное изображение:

$$\sum_{i=1}^{C} a_i \dot{h}_i(E) \ , \tag{20}$$

где C — число доступных слабых классификаторов, a_i — коэффициент i-го слабого классификатора, $\dot{h}_i$ — i-й слабый классификатор, а E — входное изображение.

Следовательно, чтобы все хорошие примеры были классифицированы верно, порог должен быть меньше наименьшей среди всех таких сумм для хороших примеров.

Учитывая, что чем порог ниже, тем больше плохих примеров будет классифицировано неверно, условие стоит изменить так: порог должен быть меньше наименьшей среди всех таких сумм для хороших примеров на наименьшее возможное значение.

При таком условии, в отличие от порогов слабых классификаторов, не требуется ни сортировки, ни проверки выбранного порога. Теперь можно оптимизировать вычислениие сумм для хороших примеров. Для этого следует обратить внимание на тот факт, что при каждой новой итерации число слабых классификаторов на текущей стадии на 1 больше, чем на предыдущей. Тогда вместо того, чтобы каждый раз вычислять сумму заново, можно сделать дополнительный массив (для каждо-

го хорошего примера — соответствующий элемент массива) и при каждой итерации добавлять туда необходимые слагаемые. Ниже приведен алгоритм с учетом всех описанных оптимизаций:

C — число доступных слабых классификаторов;

$\dot{h}_i$ — i-й слабый классификатор;

C_i — число слабых классификаторов на i-й стадии;

Θ_i — порог на i-й стадии;

s — номер текущей стадии;

ϕ — заданное наперед значение;

P_i и N_i — доля ложных позитивных и ложных негативных срабатываний i-й стадии;

g_i — i-й хороший пример;

a_i — коэффициент i-го слабого классификатора;

S_i — i-я ячейка дополнительного массива, соответствующая i-му хорошему примеру;

δ — минимальное возможное число, которое можно вычесть из уменьшаемого.

1. Задать каждому элементу дополнительного массива значение 0:

$$S_i = 0. \tag{21}$$

2. Установить текущей стадией первую, сделав число слабых классификаторов на ней равным 1:

$$\begin{aligned} s &= 1, \\ C_s &= 1. \end{aligned} \tag{22}$$

3. Для всех хороших примеров: если последний классификатор текущей стадии классифицировал пример верно, прибавить коэффициент этого классификатора к значению в соответствующей примеру ячейке дополнительного массива:

$$S_i = S_i + \dot{h}_{C_s}(g_i). \tag{23}$$

4. Задать порогу текущей стадии значение, меньшее наименьшего элемента дополнительного массива на минимально возможное число:

$$\Theta_s = \min(S_i) - \delta. \tag{24}$$

5. Если доля ложных позитивных срабатываний текущей стадии больше ϕ, увеличить число слабых классификаторов на ней, в противном случае сделать текущей следующую стадию, установив в ней число слабых классификаторов равным числу слабых классификаторов предыдущей, увеличенному на 1:

$$\begin{aligned} &\text{если } P_s > \phi, \text{тогда } C_s = C_s + 1, \\ &\text{иначе } s = s + 1,\ C_s = C_{s-1} + 1. \end{aligned} \tag{25}$$

6. Если число слабых классификаторов на текущей стадии меньше числа доступных слабых классификаторов или доля ложных положительных срабатываний равна 0, перейти к пункту 2.

7. Вернуть стандартный порог последней стадии, если выход из цикла произошел из-за того, что закончились доступные слабые классификаторы:

$$\text{если } P_s > 0, \text{тогда } \Theta_s = \frac{1}{2}\sum_{i=1}^{C_s} a_i. \tag{26}$$

7. Результаты

Алгоритм обучения, похожий на описанный выше, уже реализован в библиотеке *OpenCV* и неплохо справляется со своей задачей. Но, к сожалению, он имеет ряд недостатков.

Одним из недостатков каскадов Хаара в *OpenCV* является отсутствие прозрачности. Несмотря на то что исходный код библиотеки открытый, реализованный там алгоритм слишком сложен и при возникновении ошибок часто не дает информации, достаточной, чтобы их локализовать. Также в нем есть некоторые проблемы со стабильностью. К примеру, иногда возникает ошибка, при которой он попадает в бесконечный цикл.

Алгоритм обучения, описанный в данной статье, был реализован на компьютере в виде набора утилит для работы с каскадами Хаара.

Для обучения использовалось 500 хороших примеров, 3226 плохих примеров и 500 тестовых примеров. Итоговый каскад состоит из 2000 слабых классификаторов и 12 стадий. Размер окна был задан равным 56×12 пикселей. Обучение данного каскада заняло примерно сутки.

Данный каскад способен обнаруживать номера при ощутимых перспективных искажениях и повороте до 25 градусов. Проблемой для кас-

када являются полностью белые области изображения, которые вызывают большое количество ложных позитивных срабатываний, однако очень легко определяются при последующей обработке.

Более серьезной проблемой являются ложные позитивные срабатывания на разные надписи, дорожные знаки и прочие объекты. Такие срабатывания происходят в $12,8\%$ случаев, однако значительную их часть также можно выявить при последующей обработке.

Список литературы

1. **Viola P., Jones M.** Rapid Object Detection using a Boosted Cascade of Simple Features // *2013 IEEE Conference on Computer Vision and Pattern Recognition. 2001. Vol. 01. 511 p.*

2. **Freund Y., Schapire R. E.** Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting // *Journal of computer and system sciences 55. 1997. №SS971504. Pp. 119–139.*

СГУ им. Питирима Сорокина *Поступила 02.02.2017*

Summary

Belykh E. A. Teaching Haar cascade

This article describes Haar cascades and based on article by Paul Viola and Michael Jones. Here is described some features, that weren't decsribed in the original article. In particular, this is a weak classifier's threshold choosing and also optimized method of building the cascade of classifiers.
Keywords: pattern recognition, machine learning, classification, image processing.

References

1. **Viola P., Jones M.** Rapid Object Detection using a Boosted Cascade of Simple Features, *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2001, vol. 01, 511 p.

2. **Freund Y., Schapire R. E.** Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of computer and system sciences 55*, 1997, №SS971504, pp. 119–139.

Для цитирования: Белых Е. А. Обучение каскадов Хаара // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2017. Вып. 1 (22). С. 41–53.*

For citation: Belykh E. A. Teaching Haar cascade, *Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Informatics,* 2017, №1 (22), pp. 41–53.