Семинар 1.
ММП, весна 2013
19 февраля

Илья Толстихин
iliya.tolstikhin@gmail.com

Темы семинара:

- Бустинг;
- AdaBoost и другие функции потерь;
- Многоклассовый бустинг.

1 Алгоритм Adaboost

Напомним, что AdaBoost — одна из самых первых реализаций бустинга — применяется для решения задачи классификации в случае, когда у вас есть процедура обучения, дающая «слабые» классификаторы $h\colon \mathbb{X} \to \mathbb{Y}$ с ошибкой немного меньше $\frac{1}{2}$, а вы хотите с ее помощью построить точный классификатор с малым уровнем ошибок. AdaBoost описывает итерационную процедуру построения взвешенной композиции $a(x) = \operatorname{sgn}\big(F_T(x)\big) = \operatorname{sgn}\Big(\sum_{i=t}^{T} \alpha_t h_t(x)\Big)$ слабых *базовых* классификаторов h_t, которая на каждом t-ом шаге добавляет в композицию $F_{t-1}(x)$ один базовый классификатор $h_t(x)$ с некоторым весом α_t, не меняя при этом классификаторы $h_1, \ldots, h_{t-1}$ и веса $\alpha_1, \ldots, \alpha_{t-1}$, добавленные за предыдущие шаги.

Рассмотрим задачу с $\mathbb{Y} = \{-1, +1\}$ и обучающей выборкой $\{(x_i, y_i)\}_{i=1}^{\ell}$. Алгоритм AdaBoost использует в качестве верхней гладкой оценки пороговой функции потерь $[yF(x) \leqslant 0]$ экспоненциальную функцию потерь $\exp\big(-yF(x)\big)$. На каждом шаге очередной базовый классификатор с его весом подбираются так, чтобы достичь наибольшего уменьшения следующего функционала ошибки на обучении

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L_{\exp}\big(y_i F(x_i)\big) = \frac{1}{\ell}\sum_{i=1}^{\ell} \exp\big(-y_i F(x_i)\big),$$

который, очевидно, является верхней оценкой доли ошибок на обучающей выборке.

Итак, пусть мы находимся на t-м шаге и зафиксировали первые $t-1$ базовых классификаторов с их весами. Мы хотим решить следующую задачу:

$$\mathcal{L}(h_t, \alpha_t) = \frac{1}{\ell}\sum_{i=1}^{\ell} L_{\exp}\Big(y_i\big(F_{t-1}(x_i) + \alpha_t h_t(x_i)\big)\Big) = \frac{1}{\ell}\sum_{i=1}^{\ell} \exp\big(-y_i F_{t-1}(x_i) - y_i \alpha_t h_t(x_i)\big) \to \min_{\alpha_t, h_t}.$$

Запишем оптимизируемый функционал в следующем виде:

$$\mathcal{L}(h_t, \alpha_t) = \sum_{i=1}^{\ell} \underbrace{\frac{1}{\ell} \exp\bigl(-y_i F_{t-1}(x_i)\bigr)}_{w_i^{(t)}} \exp\bigl(-y_i \alpha_t h_t(x_i)\bigr) = \tag{1}$$

$$= \sum_{i=1}^{\ell} w_i^{(t)} \exp\bigl(-y_i \alpha_t h_t(x_i)\bigr) =$$

$$= \left(\sum_{i=1}^{\ell} \tilde{w}_i^{(t)} \exp\bigl(-y_i \alpha_t h_t(x_i)\bigr)\right) \left(\sum_{i=1}^{\ell} w_i^{(t)}\right) =$$

$$= \left(\sum_{i=1}^{\ell} \tilde{w}_i^{(t)} \exp\bigl(-y_i \alpha_t h_t(x_i)\bigr)\right) \mathcal{L}(h_{t-1}, \alpha_{t-1}) \to \min_{\alpha_t, h_t}.$$

Введенные (нормированные) веса $\tilde{w}_i^{(t)}$ не зависят ни от h_t ни от α_t, поэтому они выступают в роли *распределения* на объектах обучающей выборки на t-м шаге. Заметим, что $\tilde{w}_i^{(t)}$ зависят от F_{t-1}, поэтому на каждом шаге они (а значит и распределение на объектах) меняются.

Выберем сначала базовый классификатор h_t, воспользовавшись следующей записью:

$$\frac{\mathcal{L}(h_t, \alpha_t)}{\mathcal{L}(h_{t-1}, \alpha_{t-1})} = e^{-\alpha_t} \sum_{i: h_t(x_i) = y_i} \tilde{w}_i^{(t)} + e^{\alpha_t} \sum_{i: h_t(x_i) \neq y_i} \tilde{w}_i^{(t)} =$$

$$= (e^{\alpha_t} - e^{-\alpha_t}) \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} [h_t(x_i) \neq y_i] + e^{-\alpha_t} \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} \to \min_{\alpha_t, h_t}.$$

Таким образом, на t-ом шаге мы должны выбрать алгоритм h_t, имеющий наименьшее значение взвешенной ошибки на обучающей выборке относительно весов $\tilde{\boldsymbol{w}}^{(t)}$. При этом на первом шаге все веса совпадают, то есть мы решаем обычную задачу минимизации эмпирического риска. Обозначим взвешенную ошибку с помощью ϵ_t. Тогда мы получаем

$$\mathcal{L}(h_t, \alpha_t) = \bigl(e^{-\alpha_t}(1 - \epsilon^t) + e^{\alpha_t} \epsilon^t\bigr) \mathcal{L}(h_{t-1}, \alpha_{t-1}) \to \min_{\alpha_t}, \tag{2}$$

и становится понятно, как выбирать α_t:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right). \tag{3}$$

Замечание 1. В данном случае, благодаря виду экспоненциальной функции потерь, на следующей итерации весовые коэффициенты в (1) $w_i^{(t+1)}$ получаются из весов на t-м шаге $w_i^{(t)}$ простым домножением на $\exp\bigl(-y_i \alpha_t h_t(x_i)\bigr)$ — то есть веса объектов, правильно классифицированных базовым классификатором h_t уменьшаются, а веса остальных объектов увеличиваются. Это приведет к тому, что на следующем шаге слабый алгоритм обучения будет «больше внимания» уделять тем объектам, которые на прошлых шагах оказались наиболее сложными для классификации. В этом заключается *адаптация* алгоритма AdaBoost (*adaptive* boosting).

В общем случае при использовании других гладких верхних оценок пороговой функции потерь правила обновления весов могут иметь более сложный вид, и описанный выше шаг итерации может оказаться не таким простым.

Замечание 2. Обратим внимание на (2). Если наш слабый алгоритм обучения для любого распределения на объектах обучающей выборки $\boldsymbol{w}$ найдет базовый классификатор h с взвешенной относительно этого распределения ошибкой строго меньшей $\frac{1}{2}$:

$$\epsilon = \sum_{i=1}^{\ell} w_i[h(x_i) \neq y_i] \leqslant \frac{1}{2} - \gamma,\ \gamma > 0,$$

то с каждым шагом итераций функционал ошибки будет уменьшаться в геометрической прогрессии с множителем $2\sqrt{\epsilon(1-\epsilon)} \leqslant \sqrt{1-4\gamma^2}$.

Задача 1. *Покажите, что взвешенная ошибка выбранного на шаге t базового классификатора h_t относительно распределения $\tilde{\boldsymbol{w}}^{(t+1)}$ на объектах обучающей выборки после обновления весов равна* 0.5.

Решение:

$$\sum_{i=1}^{\ell} w_i^{(t+1)}[h_t(x_i) \neq y_i] = \sum_{i=1}^{\ell} \frac{w_i^{(t)} e^{-y_i\alpha_t h_t(x_i)}}{\sum_{j=1}^{\ell} w_j^{(t)} e^{-y_j\alpha_t h_t(x_j)}}[h_t(x_i) \neq y_i] =$$

$$= \frac{e^{\alpha_t}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) \neq y_i]}{e^{\alpha_t}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) \neq y_i] + e^{-\alpha_t}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) = y_i]} =$$

$$= \frac{\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) \neq y_i]}{\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) \neq y_i] + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}\sum_{i=1}^{\ell} w_i^{(t)}[h_t(x_i) = y_i]} =$$

$$= \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{\sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{\epsilon_t(1-\epsilon_t)}} = \frac{1}{2}.$$

Таким образом, на каждом шаге новое распределение на объектах подбирается таким образом, чтобы слабому алгоритму обучения было сложнее всего справиться с выборкой.

2 AnyBoost: другие функции потерь

Попробуем обобщить описанные выше шаги минимизации верхней оценки функционала риска для других функций потерь. Зачем рассматривать другие функции потерь? Дело в том, что экспоненциальная функция потерь, рассмотренная в прошлом разделе, чрезвычайно чувствительна к выбросам: объект с очень большим по модулю отрицательным отступом получает на каждом шаге очень большой вес, что ведет к «концентрации» слабого алгоритма обучения на работе с этим объектом. Подобное поведение AdaBoost может приводить к переобучению при наличии в выборке шумовых объектов — происходит настраивание на шум.

Есть несколько путей борьбы с подобным эффектом. Первый, как уже было рассказано на лекциях, — можно иногда выкидывать из обучающей выборки объекты с очень большими по модулю отрицательными отступами после t-й итерации

AdaBoost и запускать итерации заново. Другой — использование функций потерь, не столь сильно штрафующих большие отрицательные отступы. Одной из таких функций потерь является логарифмическая, использовавшаяся в логистической регрессии.

Задача 2. [4] *Выведите шаги бустинга для задачи классификации с* $\mathbb{Y} = \{-1, +1\}$ *и семейством базовых классификаторов* $\mathcal{H} = \{h\colon \mathbb{X} \to \mathbb{Y}\}$*, используя вместо экспоненциальной функции потерь логарифмическую:*

$$L_{\log}\big(yF(x)\big) = \log\big(1 + \exp\big(-yF(x)\big)\big)\,.$$

Решение. Во-первых, обратим внимание на то, что логарифмическая функция потерь не является строгой верхней оценкой пороговой функции: на объектах с малым отступом $yF(x) \approx 0$ мы получаем $L_{\log}\big(y, F(x)\big) \approx \log(2) < 1$. Ситуацию можно исправить, перейдя к логарифму по основанию 2.

Запишем задачу минимизации на t-ой итерации:

$$\mathcal{L}(\alpha_t, h_t) = \frac{1}{\ell}\sum_{i=1}^{\ell} L_{\log}\Big(y_i\big(F_{t-1}(x_i) + \alpha_t h_t(x_i)\big)\Big) =$$

$$= \frac{1}{\ell}\sum_{i=1}^{\ell} \log\big(1 + \exp\big(-y_i F_{t-1}(x_i) - y_i\alpha_t h_t(x_i)\big)\big) \to \min_{\alpha_t, h_t}.$$

На этот раз нам не удастся так же легко как в AdaBoost расписать этот функционал через веса объектов на этой итерации и потери t-го базового классификатора. Но мы постараемся повторить те же шаги: **сначала выбрать базовый классификатор** h_t, а затем решить одномерную задачу оптимизации

$$\mathcal{L}(\alpha_t) = \frac{1}{\ell}\sum_{i=1}^{\ell} L_{\log}\Big(y_i\big(F_{t-1}(x_i) + \alpha_t h_t(x_i)\big)\Big) \to \min_{\alpha_t}. \tag{4}$$

Если при фиксированном h_t рассматривать $\mathcal{L}$ как функцию α_t, то ее производная в точке $\alpha_t = 0$ характеризует скорость убывания функции (угол наклона графика) в этой точке. Чем меньше значение производной — тем быстрее функция $\mathcal{L}$ начнет убывать (поскольку логарифмические потери — строго убывающая функция) при увеличении α_t от нуля. Воспользуемся этим наблюдением и выберем тот базовый классификатор, который минимизирует производную функции $\mathcal{L}$ в точке $\alpha_t = 0$:

$$\frac{\partial\mathcal{L}}{\partial\alpha_t}\bigg|_{\alpha_t=0} = \frac{1}{\ell}\sum_{i=1}^{\ell}\big(-y_i h_t(x_i)\big)\underbrace{\frac{\exp\big(-y_i F_{t-1}(x_i)\big)}{1 + \exp\big(-y_i F_{t-1}(x_i)\big)}}_{w_i^{(t)}} \to \min_{h_t}.$$

Мы можем снова нормировать выделенные веса и получить следующую задачу:

$$-\frac{1}{\ell}\sum_{i=1}^{\ell}\tilde{w}_i^{(t)} y_i h_t(x_i) \to \min_{h_t};$$

$$\tilde{w}_i^{(t)} = \frac{w_i^{(t)}}{\sum_{j=1}^{\ell} w_j^{(t)}}, \quad w_i^{(t)} = -\left.\frac{\partial L_{\log}(z)}{\partial z}\right|_{z=y_i F_{t-1}(x_i)}.$$

Если заметить полезное тождество

$$[y_i \neq h_t(x_i)] = \frac{1 - h_t(x_i)y_i}{2},$$

то задача поиска базового классификатора h_t сводится к

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \tilde{w}_i^{(t)}[y_i \neq h_t(x_i)] \to \min_{h_t}.$$

Таким образом, мы снова приходим к задаче минимизации взвешенной доли ошибок на обучающей выборке с весами, меняющимися от итерации к итерации.

Нам остается решить задачу одномерной оптимизации (4).

Задача 3. *Докажите, что взвешенная доля ошибок выбранного на шаге t базового классификтора h_t относительно распределения $\tilde{\boldsymbol{w}}^{(t+1)}$ на объектах обучающей выборки после обновления равна* 0.5.

Решение:

$$\sum_{i=1}^{\ell} \tilde{w}_i^{(t+1)}[h_t(x_i) \neq y_i] = \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{\ell} \tilde{w}_i^{(t+1)} h_t(x_i)y_i = \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{\ell} \frac{e^{-y_i F_t(x_i)}}{1 + e^{-y_i F_t(x_i)}} h_t(x_i)y_i =$$

$$= \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{\ell} \left(- \frac{\partial L_{\log}(z)}{\partial z}\bigg|_{z=y_i F_t(x_i)} \right) h_t(x_i)y_i = \frac{1}{2} + \frac{1}{2}\frac{\partial \mathcal{L}(\alpha_t)}{\partial \alpha_t} = \frac{1}{2},$$

поскольку оптимальное значение α_t минимизирует функционал (4).

3 Случай многих классов

Возможно ли обобщить AdaBoost на случай K классов $\mathbb{Y} = \{1, \ldots, K\}$? В чем будет разница? Во-первых, базовые классификаторы теперь возвращают одну из K меток классов: $h_t \colon \mathcal{X} \to \{1, \ldots, K\}$. Во-вторых, нам придется поменять вид итоговой композиции: теперь вместо функции знака $\operatorname{sgn}(\cdot)$ будет использоваться $\arg\max$:

$$a(x) = \arg\max_{k=1,\ldots,K} F_T^k(x) = \arg\max_{k=1,\ldots,K} \sum_{t=1}^{T} \alpha_t [h_t(x) = k].$$

Задача 4. *Выведите шаги алгоритма AdaBoost для случая K классов.*

Решение: Итак, пусть мы на t-й итерации и базовые классификаторы $h_1, \ldots, h_{t-1}$ с их весами $\alpha_1, \ldots, \alpha_{t-1}$ уже зафиксированы. Наша ближайшая цель — выбрать очередной слабый классификатор h_t и его вес. Выбирать мы их будем также, как и раньше — чтобы они минимизировали выражение

$$\mathcal{L}(h_t, \alpha_t) = \frac{1}{\ell}\sum_{i=1}^{\ell} [\arg\max_{k=1,\ldots,K} F_T^k(x_i) \neq y_i] \to \min_{\alpha_t, h_t}. \tag{5}$$

Однако, мы пока что не можем использовать экспоненциальную функцию потерь: в AdaBoost мы использовали ее, пользуясь следующей цепочкой соотношений:

$$[h(x) \neq y] = [h(x)y \leqslant 0] \leqslant \exp\bigl(-yh(x)\bigr),$$

которая теперь в условиях многих классов не применима.

Попробуем преобразовать вид функционала (5). Нам пригодится следующее простое наблюдение. Пусть у нас есть вектор $(c_1, \ldots, c_K)$ с положительными координатами $c_i \geqslant 0$, $i = 1, \ldots, K$. Тогда справедливо следующее неравенство:

$$[\arg\max_{k=1,\ldots,K} c_k \neq j] \leqslant [c_j \leqslant \textstyle\sum_{i \neq j} c_i],$$

которое очень легко проверить.

Пользуясь этим соотношением (тем самым налагая ограничения $\alpha_i \geqslant 0$), мы можем далее оценить сверху оптимизируемый функционал следующим выражением:

$$\mathcal{L}(h_t, \alpha_t) \leqslant \frac{1}{\ell} \sum_{i=1}^{\ell} \Big[F_t^{y_i}(x_i) \leqslant \textstyle\sum_{k \neq y_i} F_t^k(x_i) \Big] =$$

$$= \frac{1}{\ell} \sum_{i=1}^{\ell} \Big[\textstyle\sum_{j=1}^{t} \alpha_j [h_j(x_i) = y_i] \leqslant \sum_{j=1}^{T} \alpha_j [h_j(x_i) \neq y_i] \Big] =$$

$$= \frac{1}{\ell} \sum_{i=1}^{\ell} \Big[\textstyle\sum_{j=1}^{t} \alpha_j \big([h_j(x_i) = y_i] - [h_j(x_i) \neq y_i] \big) \leqslant 0 \Big] =$$

$$= \frac{1}{\ell} \sum_{i=1}^{\ell} \Big[\textstyle\sum_{j=1}^{t} \alpha_j \big(2[h_j(x_i) = y_i] - 1 \big) \leqslant 0 \Big].$$

Заметим, что главной нашей целью было именно избавиться от индикаторов вида $[h_t(x) = k]$ и перейти к индикаторам вида $[\cdot \leqslant 0]$, которые далее можно сверху оценивать экспонентой. Этой цели мы достигли. Теперь мы можем воспользоваться верхней оценкой $[x \leqslant 0] \leqslant e^{-x}$, как AdaBoost:

$$\mathcal{L}(h_t, \alpha_t) \leqslant \frac{1}{\ell} \sum_{i=1}^{\ell} \exp\Big(\textstyle\sum_{j=1}^{t} \alpha_j \big(1 - 2[h_j(x_i) = y_i] \big) \Big) =$$

$$= \sum_{i=1}^{\ell} \underbrace{\frac{1}{\ell} \exp\Big(\textstyle\sum_{j=1}^{t-1} \alpha_j \big(1 - 2[h_j(x_i) = y_i] \big) \Big)}_{w_i^{(t)}} e^{\alpha_t \left(1 - 2[h_t(x_i) = y_i] \right)} =$$

$$= \left(e^{\alpha_t} \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} [h_t(x_i) \neq y_i] + e^{-\alpha_t} \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} [h_t(x_i) = y_i] \right) \sum_{i=1}^{\ell} w_i^{(t)} =$$

$$= \left((e^{\alpha_t} - e^{-\alpha_t}) \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} [h_t(x_i) \neq y_i] + e^{-\alpha_t} \sum_{i=1}^{\ell} \tilde{w}_i^{(t)} \right) \sum_{i=1}^{\ell} w_i^{(t)} \to \min_{h_t, \alpha_t}.$$

Учитывая положительность коэффициента α_t, эта задача эквивалентна поиску базового классификатора h_t с минимальным значением взвешенной относительно распределения $\tilde{\boldsymbol{w}}$ ошибки на обучающей выборке. Обозначим эту ошибку ϵ_t. Тогда значение α_t, минимизирующее наш функционал, выражается следующим образом:

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right),$$

то есть мы повторяем все шаги AdaBoost. Заметим, что при этом условие $\epsilon_t \leqslant \frac{1}{2}$ помимо сходимости ошибки композиции на обучении к нулю ведет к неотрицательности коэффициента α_t, которая была необходимом условием справедливости наших выкладок. По этой причине, если слабый алгоритм обучения на очередной итерации производит базовый классификатор со взвешенной ошибкой $\epsilon_t \geqslant \frac{1}{2}$, мы выходим из итераций без его добавления в композицию (случай $\epsilon_t = \frac{1}{2}$ означает, что ничего не изменилось). В случае двух классов $\mathbb{Y} = \{-1, +1\}$ мы бы могла просто добавить найденный классификатор h_t в композицию с отрицательным весом $\alpha_t < 0$, но, как уже отмечалось, в случае многих классов такого трюка не получится.

Замеание. В отличие от случая классификации с двумя классами, когда вероятность ошибки случайного ответа равнялась $\frac{1}{2}$, теперь случайный ответ имеет уровень ошибки $\frac{K-1}{K}$. В этом случае наши требования к слабому алгоритму обучения (производить для каждого распределения базовый классификатор со взвешенной ошибкой меньше $\frac{1}{2}$) уже не являются такими слабыми. С этой проблемой пытаются бороться ряд подходов [3, 4].

Список литературы

[1] *Hastie, T., R. Tibshirani, and J. H. Friedman.* The Elements of Statistical Learning: Data Mining, Inference and Prediction. — Springer, 2001.

[2] *Robert Shapire* Theory and Applications of Boosting. — tutorial at the Machine Learning Summer School 2009
http://videolectures.net/mlss09us_schapire_tab/.

[3] *Ji Zhu, Hui Zou, Saharon Rosset and Trevor Hastie* (2009) Multi-class AdaBoost. *Statistics and its interface.* — V. 2. — Pp. 349–360.
http://www.stanford.edu/~hastie/Papers/samme.pdf.

[4] *J. H. Friedman, Hastie, T., and R. Tibshirani.* (2000) Additive logistic regression: a statistical view of boosting. *The Annals of Statistics.* — V. 28, N. 2. — Pp. 337–407.
http://www.stanford.edu/~hastie/Papers/AdditiveLogisticRegression/alr.pdf.