Дипломна работа

Диана Генева <dageneva@qtrp.org>

2019

Съдържание

Глава 1

Нулева зона

Бла, бла, бла, аз съм толкова емоционална. Не знам

Приложение A

Приложение за AdaBoost

Ще разгледаме алгоритъма AdaBoost в дискретния случай, както оригинално е представен в фдсдф. Задачата, която имаме е следната: Имаме тренировъчни данни $M := (x_1, y_1), (x_2, y_2), \cdots (x_n, y_n), x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}, i = 1 \cdots n$. Имаме множество $\mathcal{H}$ от функции от вида $h_i : \mathbb{R}^n \to \{-1, 1\}$ и е дадена константа T. Търсим T функции от $\mathcal{H}$, които да съчетаем до получаването на нова функция, която е линейна комбинация на избраните T:

$$H(x) = \sum_{i=1}^{T} \alpha_i h_i(x)$$

AdaBoost алгоритъмът избира последователно функции h_i от $\mathcal{H}$ и избира за всяка тегло α_i. С $H_t(x)$ ще означаваме линейната комбинация, получена от първи t избрани класификатора и имаме, че:

$$\begin{aligned} H_t(x) &= \sum_{i=1}^{t} \alpha_i h_i(x) \\ &= H_{t-1}(x) + \alpha_t h_t(x) \end{aligned}$$

$$H_0(x) = 0 \quad \forall x \in \mathbb{R}^n \tag{A.0.1}$$

Идеята е t-тата избрана хипотеза да поправи грешките, които предните $t-1$ хипотези правят върху тренировъчното множество.

На всяка итерация t на лагоритъма, дефинираме разпределение върху тренировъчните данни, което ще означаваме с $\mathcal{D}_t$, където $\mathcal{D}_t(i)$ дава вероятност на i-тия пример. Идеята е да може да се даде по-голяма вероятност на тези примери, върху които предните $t-1$ избрани функции бъркат и да се избере функция, която се представя добре върху така претеглените данни. В началото на алгоритъма за $\mathcal{D}_1$ избираме равномерно разпределение, тоест:

$$\mathcal{D}_1(i) = \frac{1}{n}, i = 1 \cdots n$$

Алгоритъмът е следният:

1. $D_1(i) = \frac{1}{n}, i = 1 \cdots n$
2. $\mathcal{H} = \emptyset$
3. За всяко t от 1 до T се прави следното:
   - 3.1. $h_t = argmin_{h \in \mathcal{H}} P_{i \sim D_t}(h(x_i) \neq y_i)$
   - 3.2. $\varepsilon_t = P_{i \sim D_t}(h_t(x_i) \neq y_i)$
   - 3.3. $\alpha_t = \frac{1}{2} ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$
   - 3.4. $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{(h_t, \alpha_t)\}$
   - 3.5. За всяко i от 1 до n:
     - 3.5.1. $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)}}{\sum\limits_{j=1}^{n} \mathcal{D}_t(j) e^{-\alpha_t y_j h_t(x_j)}}$
4. $\mathcal{H} = \mathcal{H}_T$
5. Връщаме $\mathcal{H}$

Вижда се, че изборът на разпределението $\mathcal{D}(t+1)$ е такова, че дава висока вероятност на примерите, върху които предният класификатор h_t греши и обратното. Тоест, ако $h_t(x_i) = y_i, e^{-\alpha_t y_i h_i(x_i)} = e^{-\alpha_t}$, тъй като y_i и $h_t(x_i)$ са с еднакъв знак, а при $h_t(x_i) \neq y_i, e^{-\alpha_t y_i h_i(x_i)} = e^{\alpha_i}$. Тъй като $\alpha_t \geq 0$, това означава, че грешката на класификаторите, от които избираме, не трябва да надхвърля $\frac{1}{2}$ за бинарни класификатори. При модификацията за няколко класа, трябва да е изпълнено, че грешката е не повече от $\frac{1}{K}$, където K е броят на класовете.

Целта на AdaBoost алгоритъмът е да се намери такова $\mathcal{H}$, което минимизира експоненциалната грешка, която се пресмята по формулата:

$$l(h, x, y) = e^{-yh(x)}$$

За да видим, че горният алгоритъм намира такова $\mathcal{H}$, ще разгледаме следните твърдения.

Нека $w_{t,i} = e^{-y_i \mathcal{H}_{t-1}(x_i)}$

Твърдение 1. $\mathcal{D}_t = \frac{w_{t,i}}{\sum\limits_{j=1}^{n} w_{t,j}}$

Доказателство: Ще го докажем по индукция.

- База $t = 1$

$$\frac{w_{1,i}}{\sum\limits_{j=1}^{n} w_{1,j}} = \frac{e^{-y_i \mathcal{H}_0(x_i)}}{\sum\limits_{j=1}^{n} e^{-y_j \mathcal{H}_0(x_j)}}$$

$$\overset{\text{A.0.1}}{=\!=} \frac{1}{n}$$

$$\overset{\text{peg 1.}}{=\!=} \mathcal{D}_1(i) \text{ за всяко i}$$

- $\mathcal{D}_t = \dfrac{w_{t,i}}{\sum\limits_{j=1}^{n} w_{t,j}} \Rightarrow \mathcal{D}_{t+1} = \dfrac{w_{t+1,i}}{\sum\limits_{j=1}^{n} w_{t+1,j}}$

$$\mathcal{D}_{t+1} = \frac{w_{t+1,i}}{\sum\limits_{j=1}^{n} w_{t+1,j}}$$

$$\overset{\text{peg 3.5.1.}}{=\!=} \frac{\mathcal{D}_t e^{-\alpha_t y_i h_t(x_i)}}{\sum\limits_{j=1}^{n} \mathcal{D}_t e^{-\alpha_t y_j h_t(x_j)}}$$

$$\overset{\text{ИХ}}{=\!=} \frac{\frac{1}{1} e^{-\alpha_t y_i h_t(x_i)}}{\sum\limits_{j=1}^{n} \frac{1}{1} e^{-\alpha_t y_j h_t(x_j)}}$$

Твърдение 2. $P_{i \sim D_t}(h(x_i) \neq y_i) = \sum\limits_{i: h(x_i) \neq y_i} \dfrac{e^{y_i \mathcal{H}_{t-1}(x_i)}}{\sum\limits_{j=1}^{n} e^{y_j \mathcal{H}_{t-1}(x_j)}}$

Твърдение 3. Изборът на $h_t = argmin_{h \in \mathcal{H}} P_{i \sim D_t}(h(x_i) \neq y_i)$ от точка 3.2 на алгоритъма минимизира експоненциалната грешка на $\mathcal{H}_t$ върху тренировъчното множество, тоест:

$$h_t = argmin_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} l(\mathcal{H}_{t-1} + Ch, x_i, y_i) \right),$$

където C е произволна константа.

Доказателство:

$$\begin{aligned}
h_t &= argmin_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} l(\mathcal{H}_{t-1} + Ch, x_i, y_i) \right) \\
&= argmin_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} e^{-y_i (\mathcal{H}_{t-1}(x_i) + Ch(x_i))} \right) \\
&= argmin_{h \in \mathcal{H}} \left(\frac{1}{n} \sum_{i=1}^{n} e^{-y_i \mathcal{H}_{t-1}(x_i)} e^{-y_i Ch(x_i)} \right)
\end{aligned}$$

Полагаме $w_{t,i} = e^{-y_i \mathcal{H}_{t-1}(x_i)}$ и махаме константата $\dfrac{1}{n}$, тъй като тя не влияе на минимизацията.

$$= argmin_{h\in\mathcal{H}} \left(\sum_{i=1}^{n} w_{t,i}e^{-y_iCh(x_i)}\right)$$

Сега можем да разделим сумата на две, в зависимост дали $h(x_i) = y_i$. Ако това е изпълнено, то $y_i h(x_i) = 1$ и е равно на -1 в противен случай, тоест:

$$\begin{aligned}
&= argmin_{h\in\mathcal{H}} \left(\sum_{i:h(x_i)=y_i}^{n} w_{t,i}e^{-C} + \sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}e^{C}\right) \\
&= argmin_{h\in\mathcal{H}} \left(\left[\sum_{i=1}^{n} w_{t,i}e^{-C} - \sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}e^{-C}\right] + \sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}e^{C}\right) \\
&= argmin_{h\in\mathcal{H}} \left(\sum_{i=1}^{n} w_{t,i}e^{-C} + \sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}(e^{C} - e^{-C})\right)
\end{aligned}$$

$\sum\limits_{i=1}^{n} w_{t,i}e^{-C}$ е константа спрямо h, затова също не участва в минимизацията и тогава:

$$\begin{aligned}
&= argmin_{h\in\mathcal{H}} \left(\sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}(e^{C} - e^{-C})\right) \\
&= argmin_{h\in\mathcal{H}} \left((e^{C} - e^{-C}) \sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}\right) \\
&= argmin_{h\in\mathcal{H}} \left(\sum_{i:h(x_i)\neq y_i}^{n} w_{t,i}\right),
\end{aligned}$$

тъй като и $(e^{C} - e^{-C})$ не зависи от h. Можем да умножим по константата $\frac{1}{\sum\limits_{j=1}^{n} w_{t,j}}$ и да получим:

$$\begin{aligned}
&= argmin_{h\in\mathcal{H}} \left(\frac{\sum\limits_{i:h(x_i)\neq y_i}^{n} w_{t,i}}{\sum\limits_{j=1}^{n} w_{t,j}}\right), \\
&= argmin_{h\in\mathcal{H}} \left(\sum_{i:h(x_i)\neq y_i}^{n} \frac{w_{t,i}}{\sum\limits_{j=1}^{n} w_{t,j}}\right), \\
&= P_{i\sim D_t}(h(x_i) \neq y_i)
\end{aligned}$$