Boosting.
Metoda wzmocnienia klasyfikatorów

Hung Son Nguyen

Maj 2016

Outline

Boosting jest ogólną metodą służącą zwiększeniu skuteczności dowolnego algorytmu uczenia.

Boosting jest ogólną metodą służącą zwiększeniu skuteczności dowolnego algorytmu uczenia.

Boosting jest ogólną metodą służącą zwiększeniu skuteczności dowolnego algorytmu uczenia.

Idea

Budowanie "mocnego i złożonego klasyfikatora" ze "słabych i prostych klasyfikatorów".

Rys historyczny

Leslie Valiant i Michael Kearns byli pierwszymi, którzy pokazali, że "słabe" algorytmy uczące, których skuteczność jest nawet niewiele lepsza niż losowe zgadywanie, mogą być wykorzystane do stworzenia dowolnie skutecznego "silnego" klasyfikatora.

Rys historyczny

Leslie Valiant i Michael Kearns byli pierwszymi, którzy pokazali, że "słabe" algorytmy uczące, których skuteczność jest nawet niewiele lepsza niż losowe zgadywanie, mogą być wykorzystane do stworzenia dowolnie skutecznego "silnego" klasyfikatora.

Robert Schapire jako pierwszy przedstawił algorytm wzmacniania działający w czasie wielomianowym.

Rys historyczny

Yoav Freund zaproponował znacznie bardziej efektywny algorytm wzmacniania, który mimo tego, że był w pewnym sensie optymalny, miał pewne istotne w praktyce wady.

Rys historyczny

Yoav Freund zaproponował znacznie bardziej efektywny algorytm wzmacniania, który mimo tego, że był w pewnym sensie optymalny, miał pewne istotne w praktyce wady.

Pierwsze eksperymenty z tymi wczesnymi algorytmami wzmacniania zostały przeprowadzone przez zespół Drucker, Schapire, Simard i dotyczyły zadania OCR - optical character recognition (sieci neuronowe jako "proste" klasyfikatory).

Rys historyczny

Yoav Freund zaproponował znacznie bardziej efektywny algorytm wzmacniania, który mimo tego, że był w pewnym sensie optymalny, miał pewne istotne w praktyce wady.

Pierwsze eksperymenty z tymi wczesnymi algorytmami wzmacniania zostały przeprowadzone przez zespół Drucker, Schapire, Simard i dotyczyły zadania OCR - optical character recognition (sieci neuronowe jako "proste" klasyfikatory).

Freund i Schapire przedstawili algorytm **AdaBoost**, który rozwiązał wiele praktycznych trudności wcześniejszych algorytmów wzmacniania.

Outline

Algorytm na wejściu otrzymuje zbiór treningowy $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, gdzie każdy x_i należy do pewnej dziedziny problemu X, natomiast każda etykieta (decyzja) y_i należy do pewnego zbioru Y. Dla ułatwienia będziemy na razie zakładać, że $Y = \{-1, +1\}$.

Algorytm na wejściu otrzymuje zbiór treningowy $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, gdzie każdy x_i należy do pewnej dziedziny problemu X, natomiast każda etykieta (decyzja) y_i należy do pewnego zbioru Y. Dla ułatwienia będziemy na razie zakładać, że $Y = \{-1, +1\}$.

AdaBoost (Adaptive Boosting) wywołuje wybrany "słaby" algorytm uczący w serii T iteracji. Zakładamy, że błąd uzyskiwanych klasyfikatorów na zbiorze treningowym jest mniejszy niż $\frac{1}{2}$.

Algorytm na wejściu otrzymuje zbiór treningowy $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, gdzie każdy x_i należy do pewnej dziedziny problemu X, natomiast każda etykieta (decyzja) y_i należy do pewnego zbioru Y. Dla ułatwienia będziemy na razie zakładać, że $Y = \{-1, +1\}$.

AdaBoost (Adaptive Boosting) wywołuje wybrany "słaby" algorytm uczący w serii T iteracji. Zakładamy, że błąd uzyskiwanych klasyfikatorów na zbiorze treningowym jest mniejszy niż $\frac{1}{2}$.

Jedną z głównych idei algorytmu jest utrzymywanie rozkładu (lub wag elementów) dla zbioru treningowego. Wagę i-tego elementu ze zbioru treningowego w iteracji t będziemy oznaczali przez $D_t(i)$.

Początkowo wszystkie wagi są ustawione na równe wartości, ale po każdej iteracji wagi elementów źle klasyfikowanych są zwiększane. Dzięki temu mamy możliwość skierowania uwagi "słabego" klasyfikatora na pewne elementy (trudne do wyuczenia) ze zbioru treningowego.

Początkowo wszystkie wagi są ustawione na równe wartości, ale po każdej iteracji wagi elementów źle klasyfikowanych są zwiększane. Dzięki temu mamy możliwość skierowania uwagi "słabego" klasyfikatora na pewne elementy (trudne do wyuczenia) ze zbioru treningowego.

Zadaniem "słabego" algorytmu uczącego jest zbudowanie klasyfikatora (ang. hypothesis) $h_t : X \to Y$ odpowiedniego dla aktualnego rozkładu D_t.

Początkowo wszystkie wagi są ustawione na równe wartości, ale po każdej iteracji wagi elementów źle klasyfikowanych są zwiększane. Dzięki temu mamy możliwość skierowania uwagi "słabego" klasyfikatora na pewne elementy (trudne do wyuczenia) ze zbioru treningowego.

Zadaniem "słabego" algorytmu uczącego jest zbudowanie klasyfikatora (ang. hypothesis) $h_t : X \to Y$ odpowiedniego dla aktualnego rozkładu D_t.

Skuteczność takiego klasyfikatora jest mierzona przez jego błąd (z uwzględnieniem rozkładu D_t):

$$\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i)\neq y_i} D_t(i)$$

W praktyce "słaby" algorytm uczący może być algorytmem, który uwzględnia rozkład D_t. Nie jest to jednak konieczne. Kiedy algorytm nie pozwala na bezpośrednie uwzględnienie rozkładu D_t, losuje się (względem D_t) podzbiór zbioru treningowego, na którym następnie wywołuje się algorytm uczący.

W praktyce "słaby" algorytm uczący może być algorytmem, który uwzględnia rozkład D_t. Nie jest to jednak konieczne. Kiedy algorytm nie pozwala na bezpośrednie uwzględnienie rozkładu D_t, losuje się (względem D_t) podzbiór zbioru treningowego, na którym następnie wywołuje się algorytm uczący.

Kiedy AdaBoost dostaje klasyfikator h_t dobierany jest parametr α_t. Intuicyjnie α_t odpowiada za wagę jaką przykładamy do klasyfikatora h_t. Zauważmy, że $\alpha_t \geq 0$ gdy $\varepsilon_t \leq \frac{1}{2}$. Ponadto α_t rośnie kiedy ε_t maleje.

W praktyce "słaby" algorytm uczący może być algorytmem, który uwzględnia rozkład D_t. Nie jest to jednak konieczne. Kiedy algorytm nie pozwala na bezpośrednie uwzględnienie rozkładu D_t, losuje się (względem D_t) podzbiór zbioru treningowego, na którym następnie wywołuje się algorytm uczący.

Kiedy AdaBoost dostaje klasyfikator h_t dobierany jest parametr α_t. Intuicyjnie α_t odpowiada za wagę jaką przykładamy do klasyfikatora h_t. Zauważmy, że $\alpha_t \geq 0$ gdy $\varepsilon_t \leq \frac{1}{2}$. Ponadto α_t rośnie kiedy ε_t maleje.

Rozkład D_t jest następnie zmieniany tak, aby zwiększyć (zmniejszyć) wagi elementów zbioru treningowego, które są źle (dobrze) klasyfikowane przez h_t. Stąd wagi mają tendencję do skupiania się na "trudnych" przykładach.

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
- $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i) \neq y_i} D_t(i)$

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
- $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i) \neq y_i} D_t(i)$
- $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
- $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i) \neq y_i} D_t(i)$
- $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
- Uaktualnij wagi elementów zbioru treningowego:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{jeśli } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{jeśli } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} \times D_t(i) \times e^{-\alpha_t y_i h_t(x_i)} \end{aligned}$$

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
- $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i) \neq y_i} D_t(i)$
- $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
- Uaktualnij wagi elementów zbioru treningowego:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{jeśli } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{jeśli } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} \times D_t(i) \times e^{-\alpha_t y_i h_t(x_i)} \end{aligned}$$

Pseudokod algorytmu

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Inicjalizacja: $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$

for $t = 1, \ldots, T$ **do**:

- Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
- $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i) \neq y_i} D_t(i)$
- $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
- Uaktualnij wagi elementów zbioru treningowego:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{jeśli } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{jeśli } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} \times D_t(i) \times e^{-\alpha_t y_i h_t(x_i)} \end{aligned}$$

Z_t jest faktorem normalizującym (wybranym tak, aby D_{t+1} było rozkładem).

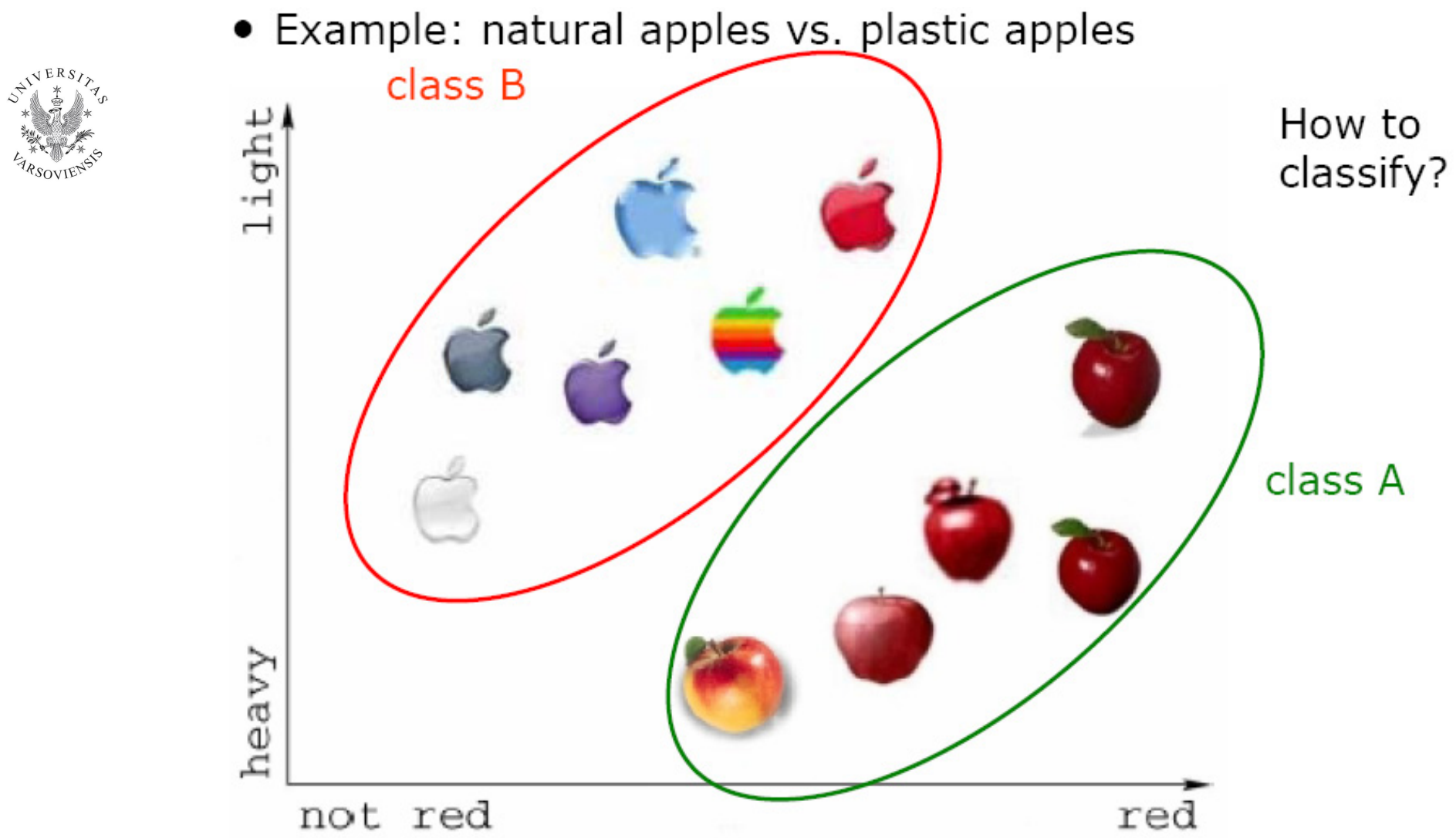
Wynikowy klasyfikator:

Wynikowy klasyfikator powstaje przez ważone głosowanie klasyfikatorów h_t, gdzie α_t jest wagą przypisaną klasyfikatorowi h_t.

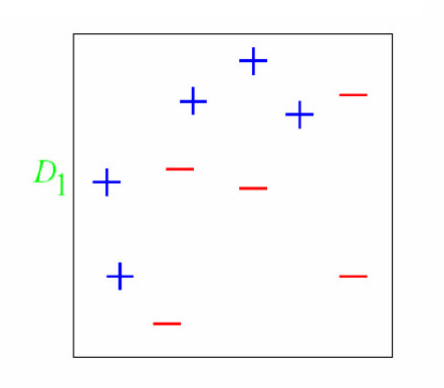
$t =$	1	2	...	T
słabe klasyfikatory	h_1	h_2	...	h_T
wagi	α_1	α_2	...	α_T

$$H(x) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) = \operatorname{argmax}_{i \in \{-1,+1\}} \left(\sum_{\substack{t \in \{1,\ldots,T\} \\ h(x)=i}} \alpha_t \right)$$

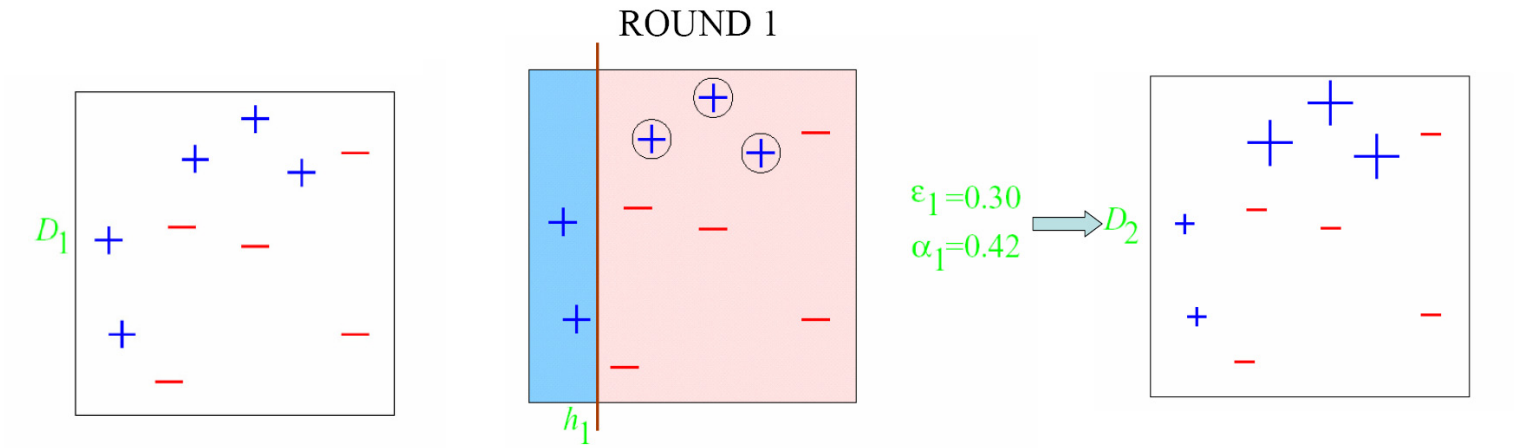
Przykład


Example: natural apples vs. plastic apples
class B
How to classify?
light
heavy
not red
red
class A

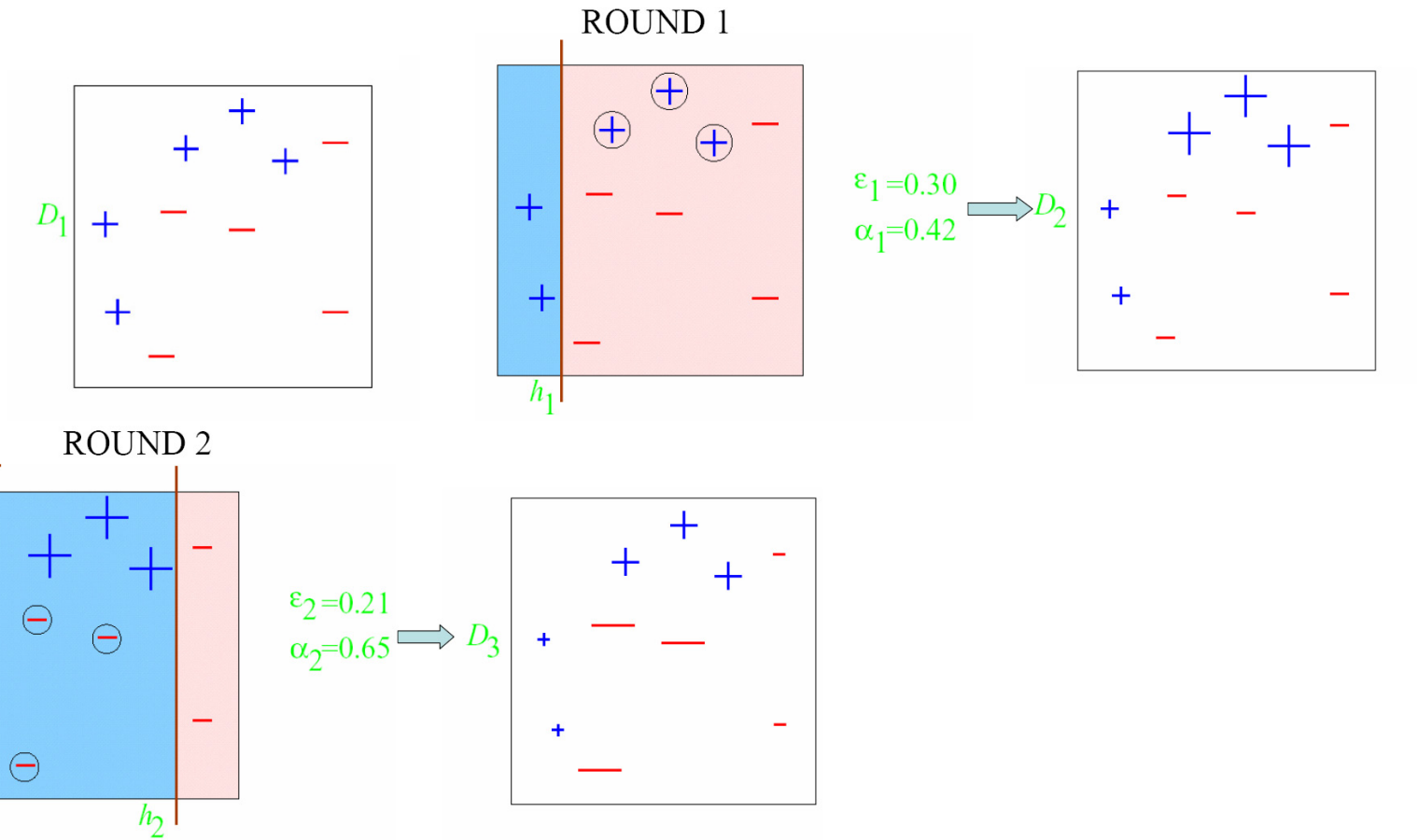
Przykład



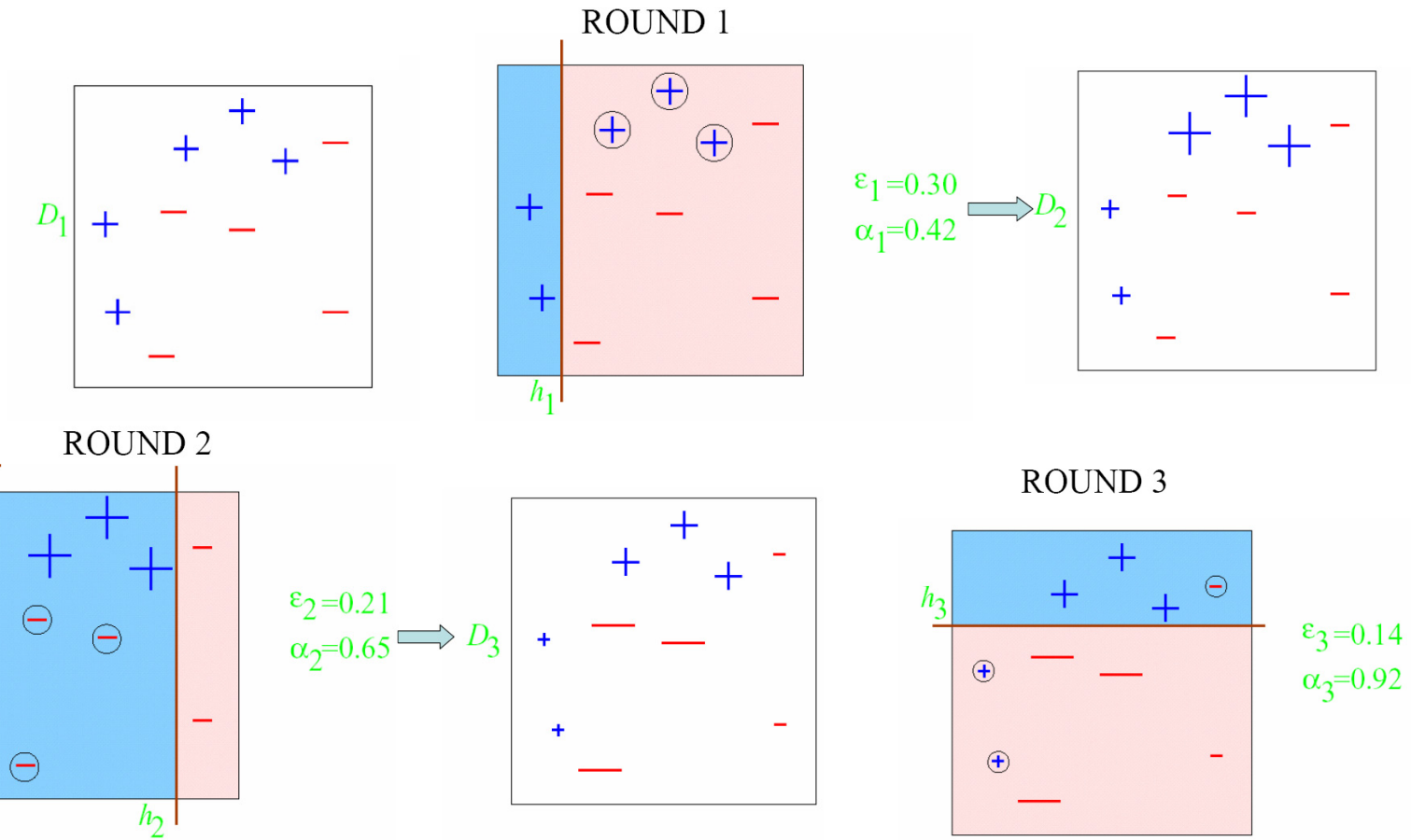
Przykład



Przykład

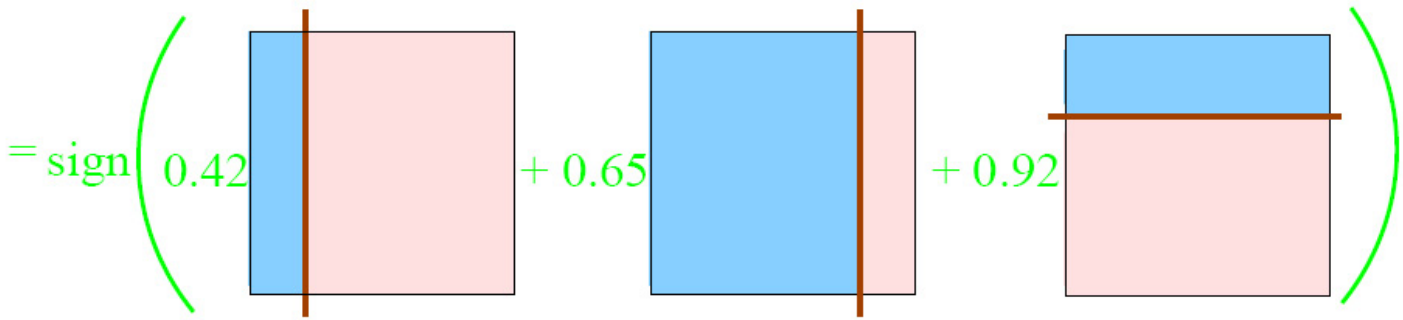

ROUND 1
D_1
h_1
$\varepsilon_1=0.30$
$\alpha_1=0.42$
D_2
ROUND 2
h_2
$\varepsilon_2=0.21$
$\alpha_2=0.65$
D_3

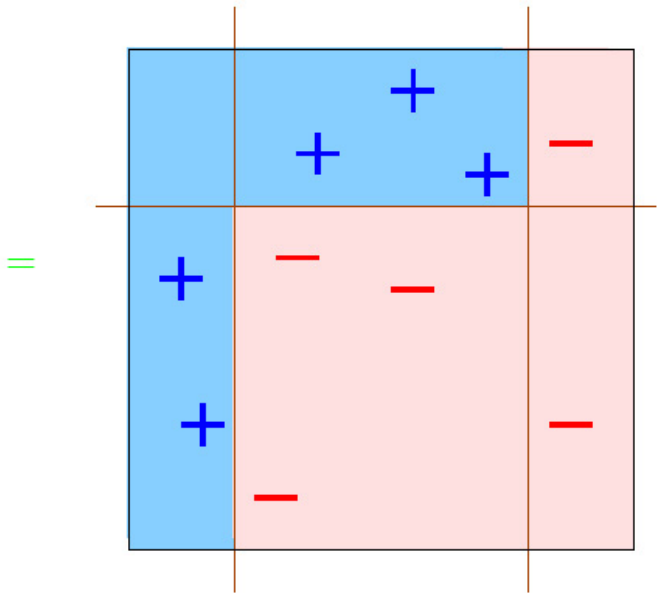
Przykład


ROUND 1
D_1
h_1
$\varepsilon_1=0.30$
$\alpha_1=0.42$
D_2
ROUND 2
h_2
$\varepsilon_2=0.21$
$\alpha_2=0.65$
D_3
ROUND 3
h_3
$\varepsilon_3=0.14$
$\alpha_3=0.92$

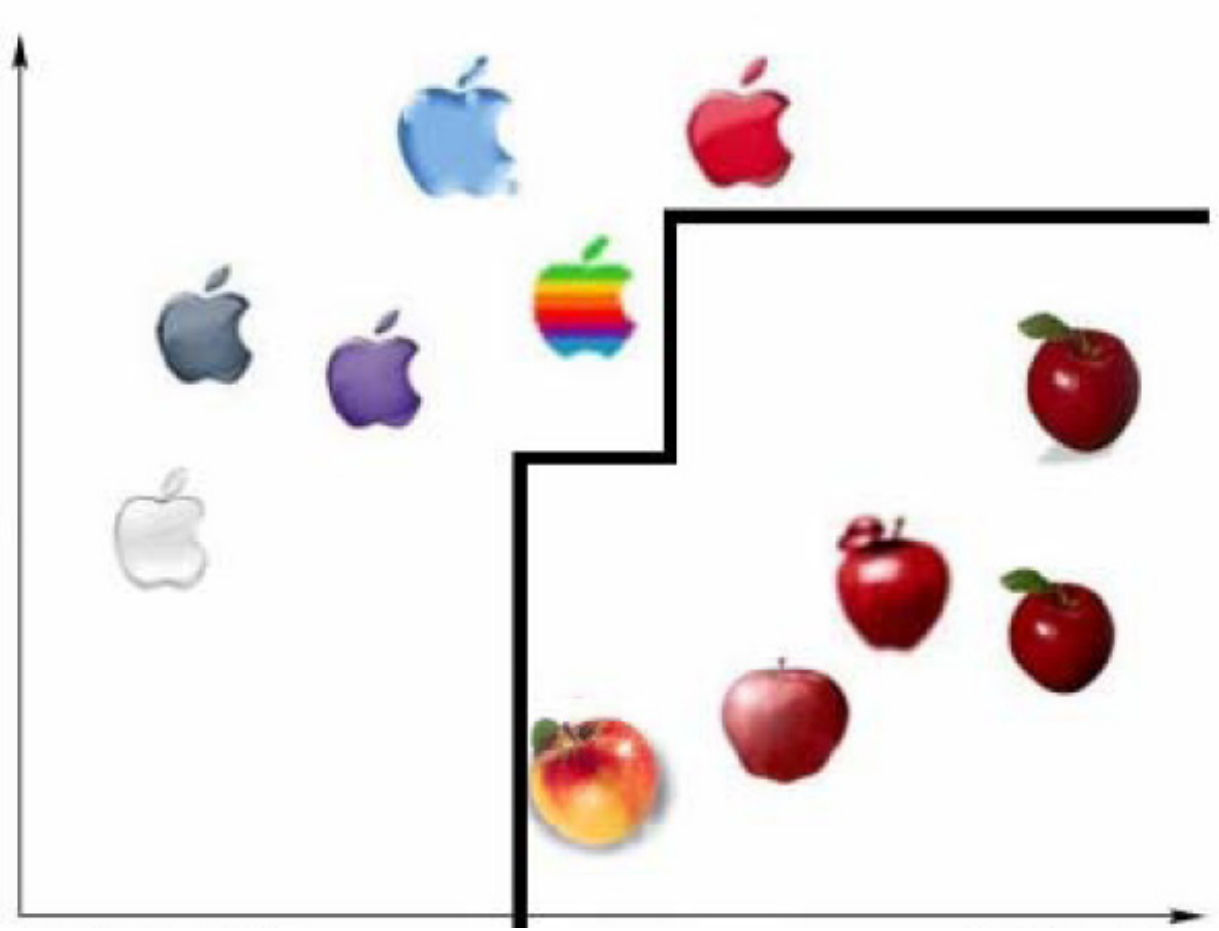
Przykład - ostateczny klasyfikator

$$H_{final} =$$





Przykład



Outline

Najbardziej podstawowa własność algorytmu AdaBoost to jego zdolność do zredukowania błędu na zbiorze treningowym.

Najbardziej podstawowa własność algorytmu AdaBoost to jego zdolność do zredukowania błędu na zbiorze treningowym.

Schapire i Singer pokazali, że błąd wynikowego klasyfikatora na zbiorze treningowym spełnia nierówność:

$$\frac{1}{m}|i : H(x_i) \neq y_i| \leq \frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t, \qquad (1)$$

gdzie $f(x) = \sum_t \alpha_t h_t(x)$, czyli $H(x) = sign(f(x))$.
Nierówność bierze się stąd, że

$$\text{jeżeli } (H(x_i) \neq y_i) \text{ to } (e^{-y_i f(x_i)} \geq 1).$$

Udowodnimy teraz równość:

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t.$$

Udowodnimy teraz równość:

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t.$$

Pamiętamy definicję:

$$D_{t+1} = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}.$$

Udowodnimy teraz równość:

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t.$$

Pamiętamy definicję:

$$D_{t+1} = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}.$$

Stąd

$$\frac{Z_t D_{t+1}(i)}{D_t(i)} = e^{-\alpha_t y_i h_t(x_i)}.$$

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} \quad = \quad \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)}$$

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} = \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)}$$

$$= \frac{1}{m}\sum_i \prod_t \exp(-y_i \alpha_t h_t(x_i))$$

$$
\begin{aligned}
\frac{1}{m}\sum_i e^{-y_i f(x_i)} &= \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)} \\
&= \frac{1}{m}\sum_i \prod_t \exp(-y_i \alpha_t h_t(x_i)) \\
&= \frac{1}{m}\sum_i \prod_t \frac{Z_t D_{t+1}(i)}{D_t(i)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{m}\sum_i e^{-y_i f(x_i)} &= \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)} \\
&= \frac{1}{m}\sum_i \prod_t \exp(-y_i \alpha_t h_t(x_i)) \\
&= \frac{1}{m}\sum_i \prod_t \frac{Z_t D_{t+1}(i)}{D_t(i)} \\
&= \frac{1}{m}\sum_i \frac{Z_1 D_2(i)}{D_1(i)} \frac{Z_2 D_3(i)}{D_2(i)} \cdots \frac{Z_T D_{T+1}(i)}{D_T(i)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{m}\sum_i e^{-y_i f(x_i)} &= \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)} \\
&= \frac{1}{m}\sum_i \prod_t \exp(-y_i \alpha_t h_t(x_i)) \\
&= \frac{1}{m}\sum_i \prod_t \frac{Z_t D_{t+1}(i)}{D_t(i)} \\
&= \frac{1}{m}\sum_i \frac{Z_1 D_2(i)}{D_1(i)} \frac{Z_2 D_3(i)}{D_2(i)} \cdots \frac{Z_T D_{T+1}(i)}{D_T(i)} \\
&= \frac{1}{m}\prod_t Z_t \sum_i \frac{D_{T+1}(i)}{D_1(i)} = \prod_t Z_t
\end{aligned}
$$

Oszacowanie (1) sugeruje (strategię zachłanną), że błąd na zbiorze treningowym może być redukowany poprzez wybieranie takich α_t i h_t, aby w każdej iteracji minimalizować:

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}.$$

Oszacowanie (1) sugeruje (strategię zachłanną), że błąd na zbiorze treningowym może być redukowany poprzez wybieranie takich α_t i h_t, aby w każdej iteracji minimalizować:

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}.$$

Wybierając

$$\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t}),$$

dostajemy:

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}$$

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t}
\end{aligned}
$$

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} \\
&= (1 - \varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}
\end{aligned}
$$

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} \\
&= (1-\varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \\
&= (1-\varepsilon_t) \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}
\end{aligned}
$$

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} \\
&= (1-\varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \\
&= (1-\varepsilon_t) \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \\
&= 2\sqrt{\varepsilon_t (1-\varepsilon_t)}
\end{aligned}
$$

Mamy zatem:

$$\prod_t Z_t \quad = \quad \prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}]$$

Mamy zatem:

$$\begin{aligned}\prod_t Z_t &= \prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] \\ &= \prod_t [1-4\gamma_t^2]\end{aligned}$$

Mamy zatem:

$$\begin{aligned}\prod_t Z_t &= \prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] \\ &= \prod_t [1-4\gamma_t^2] \\ &\leq e^{-2\sum_t \gamma_t^2},\end{aligned}$$

gdzie $\varepsilon_t = \frac{1}{2} - \gamma_t$.

Mamy zatem:

$$\begin{aligned}\prod_t Z_t &= \prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] \\ &= \prod_t [1-4\gamma_t^2] \\ &\leq e^{-2\sum_t \gamma_t^2},\end{aligned}$$

gdzie $\varepsilon_t = \frac{1}{2} - \gamma_t$.

Ograniczenie to jako pierwsi pokazali Freund i Schapire. Stąd, jeśli "słaby" algorytm produkuje klasyfikatory, dla których γ_t jest większe od zera, błąd na zbiorze treningowym spada wykładniczo szybko do zera.

Jakkolwiek jest to miła własność nie zawsze będziemy nią zainteresowani. Zbieżność błędu na zbiorze treningowym do zera może powodować zbytnie dopasowanie wynikowego klasyfikatora do danych treningowych (ang. overfitting).

Outline

Freund i Schapire pokazali jak można oszacować ogólny błąd wynikowego klasyfikatora w terminach błędu na zbiorze treningowym, mocy zbioru treningowego m, wymiaru Vapnika-Chervonenkisa d dla klasy "słabych" klasyfikatorów oraz ilości iteracji algorytmu wzmacniania T.

Freund i Schapire pokazali jak można oszacować ogólny błąd wynikowego klasyfikatora w terminach błędu na zbiorze treningowym, mocy zbioru treningowego m, wymiaru Vapnika-Chervonenkisa d dla klasy "słabych" klasyfikatorów oraz ilości iteracji algorytmu wzmacniania T.

Pokazali oni, że z dużym prawdopodobieństwem ogólny błąd wynikowego klasyfikatora jest nie równy co najwyżej:

$$\hat{Pr}[H(x) \neq y] + O(\sqrt{\frac{Td}{m}}),$$

gdzie $\hat{Pr}[\cdot]$ oznacza empiryczne prawdopodobieństwo na zbiorze treningowym.

Wymiar Vapnika-Chervonenkisa

Wymiar Vapnika-Chervonenkisa mierzy "umiejętności" algorytmów klasyfikacji. Jest on zdefiniowany jako największa liczba punktów jakie algorytm może "poszatkować", "rozbić" w dowolny sposób.

Wymiar Vapnika-Chervonenkisa

Wymiar Vapnika-Chervonenkisa mierzy "umiejętności" algorytmów klasyfikacji. Jest on zdefiniowany jako największa liczba punktów jakie algorytm może "poszatkować", "rozbić" w dowolny sposób.

Przykłady...

Ograniczenie to sugeruje, że wynikowy klasyfikator (przy zbyt dużej liczbie iteracji) może być zbytnio dopasowany do danych treningowych.

Ograniczenie to sugeruje, że wynikowy klasyfikator (przy zbyt dużej liczbie iteracji) może być zbytnio dopasowany do danych treningowych.

W praktyce czasami się to zdarza. Jednakże przeprowadzone przez kilku autorów eksperymenty pokazały empirycznie, że zbytnie dopasowanie nie występuje zbyt często, nawet przy dużej liczbach iteracji ustawianych na duże wartości.

Ograniczenie to sugeruje, że wynikowy klasyfikator (przy zbyt dużej liczbie iteracji) może być zbytnio dopasowany do danych treningowych.

W praktyce czasami się to zdarza. Jednakże przeprowadzone przez kilku autorów eksperymenty pokazały empirycznie, że zbytnie dopasowanie nie występuje zbyt często, nawet przy dużej liczbach iteracji ustawianych na duże wartości.

Co więcej, obserwowano przypadki, w których AdaBoost zmniejszał ogólny błąd długo po tym jak błąd na zbiorze treningowym osiągnął zero.

Outline

W praktyce algorytm AdaBoost ma sporo zalet:

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.
- Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.
- Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.
- Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.
- Nie potrzebuje wcześniejszej wiedzy na temat "słabego" algorytmu uczenia, który wykorzystuje. Co za tym idzie, może być elastycznie połączony z dowolną metodą szukającą "słabych" klasyfikatorów.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.
- Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.
- Nie potrzebuje wcześniejszej wiedzy na temat "słabego" algorytmu uczenia, który wykorzystuje. Co za tym idzie, może być elastycznie połączony z dowolną metodą szukającą "słabych" klasyfikatorów.

W praktyce algorytm AdaBoost ma sporo zalet:

- Jest szybki, prosty i łatwy w implementacji.
- Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.
- Nie potrzebuje wcześniejszej wiedzy na temat "słabego" algorytmu uczenia, który wykorzystuje. Co za tym idzie, może być elastycznie połączony z dowolną metodą szukającą "słabych" klasyfikatorów.
- Miłą własnością algorytmu AdaBoost jest jego zdolność do wykrywania przypadków nietypowych. Ponieważ AdaBoost zwiększa wagi elementów źle klasyfikowanych. Elementy, na których skupiony został rozkład, te z największymi wagami, możemy uważać za przypadki nietypowe.

Z drugiej strony można jednak dostrzec kilka wad:

- Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.

Z drugiej strony można jednak dostrzec kilka wad:

- Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.

Z drugiej strony można jednak dostrzec kilka wad:

- Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.
- Wzmacnianie może nie przynieść oczekiwanego skutku jeśli algorytm nie dostanie dostatecznej ilości danych, zastosujemy zbyt skomplikowany lub zbyt prosty "słaby" algorytm uczenia.

Z drugiej strony można jednak dostrzec kilka wad:

- Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.
- Wzmacnianie może nie przynieść oczekiwanego skutku jeśli algorytm nie dostanie dostatecznej ilości danych, zastosujemy zbyt skomplikowany lub zbyt prosty "słaby" algorytm uczenia.

Z drugiej strony można jednak dostrzec kilka wad:

- Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.
- Wzmacnianie może nie przynieść oczekiwanego skutku jeśli algorytm nie dostanie dostatecznej ilości danych, zastosujemy zbyt skomplikowany lub zbyt prosty "słaby" algorytm uczenia.
- Wydaje się być szczególnie wrażliwy na szumy w danych.

Outline

Bibliografia I

R.E. Schapire

A brief introduction to boosting.

In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999.

Bibliografia II

R. Meir and G. Rätsch
An introduction to boosting and leveraging.
In S. Mendelson and A. Smola, editors, Advanced Lectures on Machine Learning, LNCS, pages 119-184. Springer, 2003. In press. Copyright by Springer Verlag.