The Boosting Approach to Machine Learning

Maria-Florina Balcan

03/16/2015

Boosting

- General method for improving the accuracy of any given learning algorithm.
- Works by creating a series of challenge datasets s.t. even modest performance on these can be used to produce an overall high-accuracy predictor.
    - Works amazingly well in practice --- Adaboost and its variations one of the top 10 algorithms.
    - Backed up by solid foundations.

Readings:

- The Boosting Approach to Machine Learning: An Overview. Rob Schapire, 2001
- Theory and Applications of Boosting. NIPS tutorial.

  http://www.cs.princeton.edu/~schapire/talks/nips-tutorial.pdf

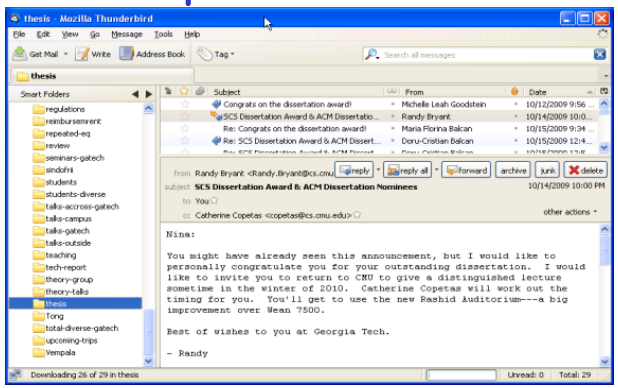
Plan for today:

- Motivation.
- A bit of history.
- Adaboost: algo, guarantees, discussion.
- Focus on supervised classification.

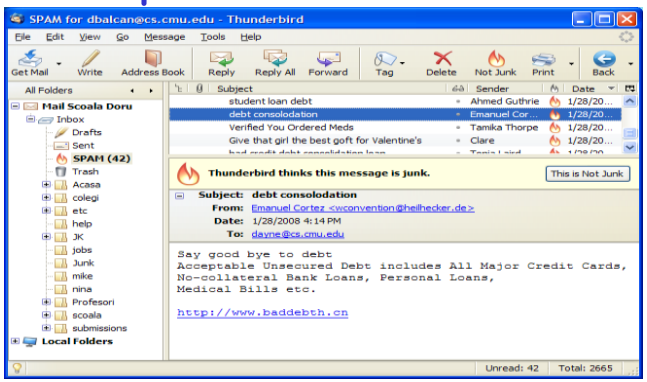
An Example: Spam Detection

- E.g., classify which emails are spam and which are important.

Not spam



spam

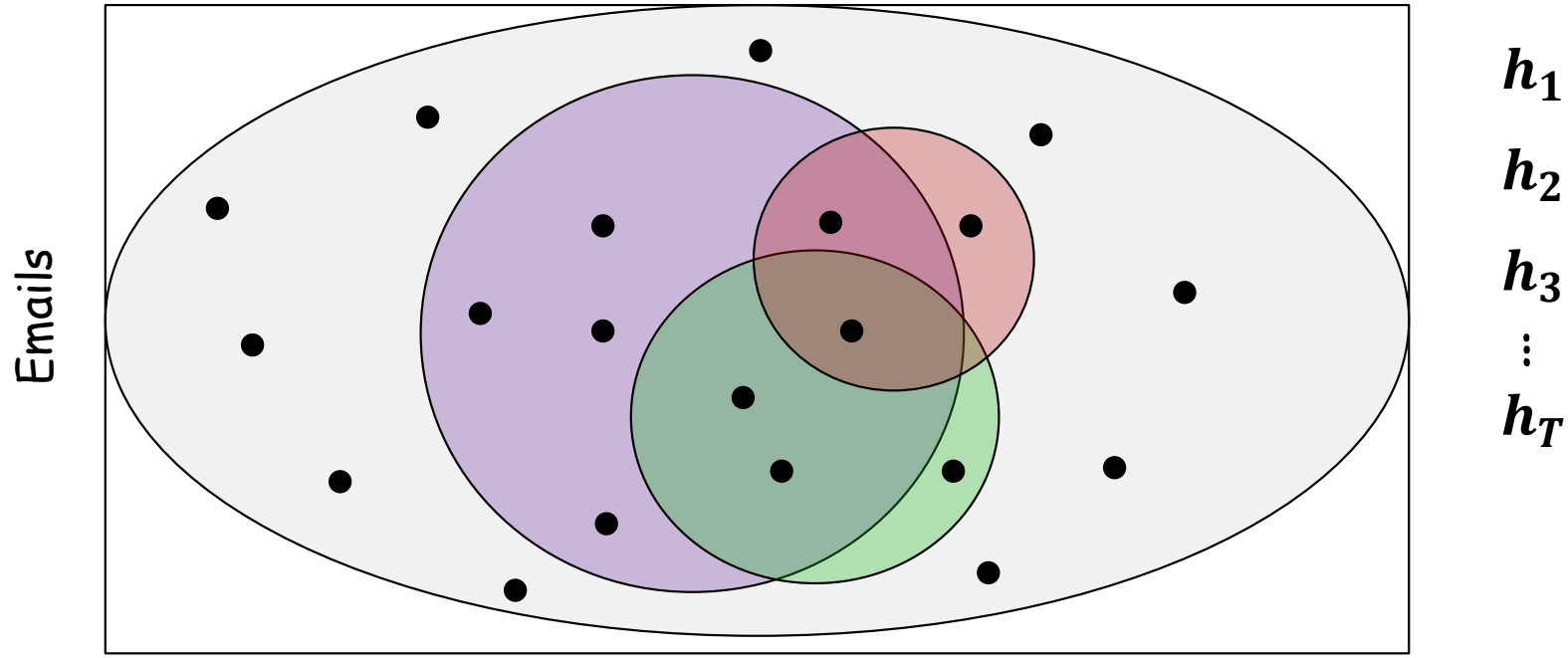


Key observation/motivation:

- Easy to find rules of thumb that are often correct.
  - E.g., "If buy now in the message, then predict spam."
  - E.g., "If say good-bye to debt in the message, then predict spam."
- Harder to find single rule that is very highly accurate.

An Example: Spam Detection

- Boosting: meta-procedure that takes in an algo for finding rules of thumb (weak learner). Produces a highly accurate rule, by calling the weak learner repeatedly on cleverly chosen datasets.



- apply weak learner to a subset of emails, obtain rule of thumb
- apply to 2nd subset of emails, obtain 2nd rule of thumb
- apply to 3rd subset of emails, obtain 3rd rule of thumb
- repeat T times; combine weak rules into a single highly accurate rule.

改变分类的权重.

Boosting: Important Aspects

How to choose examples on each round?

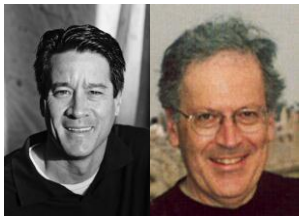
- Typically, concentrate on "hardest" examples (those most often misclassified by previous rules of thumb)

How to combine rules of thumb into single prediction rule?

- take (weighted) majority vote of rules of thumb

Historically….

Weak Learning vs Strong/PAC Learning



- [Kearns & Valiant '88]: defined weak learning: being able to predict better than random guessing (error $\leq \frac{1}{2} - \gamma$), consistently. 弱分类器.

- Posed an open pb: "Does there exist a boosting algo that turns a weak learner into a strong PAC learner (that can produce arbitrarily accurate hypotheses)?"

- Informally, given "weak" learning algo that can consistently find classifiers of error $\leq \frac{1}{2} - \gamma$, a boosting algo would provably construct a single classifier with error $\leq \epsilon$.

Weak Learning vs Strong/PAC Learning

Strong (PAC) Learning

- $\exists$ algo A
- $\forall\, c \in H$
- $\forall D$
- $\forall\, \epsilon > 0$
- $\forall\, \delta > 0$
- A produces h s.t.:

$$\Pr[err(h) \geq \epsilon] \leq \delta$$

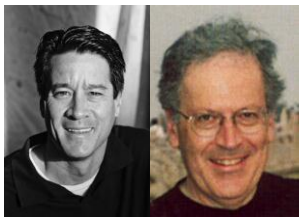
$P(err(h) < \delta) > 1 - \delta$

Weak Learning

- $\exists$ algo A
- $\exists \gamma > 0$
- $\forall\, c \in H$
- $\forall D$
- $\forall\, \epsilon > \frac{1}{2} - \gamma$
- $\forall\, \delta > 0$
- A produces h s.t.

$$\Pr[err(h) \geq \epsilon] \leq \delta$$

- [Kearns & Valiant '88]: defined weak learning & posed an open pb of finding a boosting algo.



Surprisingly….

Weak Learning =Strong (PAC) Learning

Original Construction [Schapire '89]:

- poly-time boosting algo, exploits that we can learn a little on every distribution.
  - A modest booster obtained via calling the weak learning algorithm on 3 distributions.

$$\text{Error} = \beta < \frac{1}{2} - \gamma \rightarrow \text{error } 3\beta^2 - 2\beta^3$$

多数投票 → 合到一起

  - Then amplifies the modest boost of accuracy by running this somehow recursively.

- Cool conceptually and technically, not very practical.

An explosion of subsequent work

Background (cont.)

- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms
- experiments and applications using AdaBoost:

[Drucker & Cortes '96]
[Jackson & Craven '96]
[Freund & Schapire '96]
[Quinlan '96]
[Breiman '96]
[Maclin & Opitz '97]
[Bauer & Kohavi '97]
[Schwenk & Bengio '98]
[Schapire, Singer & Singhal '98]
[Abney, Schapire & Singer '99]
[Haruno, Shirai & Ooyama '99]
[Cohen & Singer' 99]
[Dietterich '00]
[Schapire & Singer '00]
[Collins '00]
[Escudero, Màrquez & Rigau '00]
[Iyer, Lewis, Schapire, Singer & Singhal '00]
[Onoda, Rätsch & Müller '00]
[Tieu & Viola '00]
[Walker, Rambow & Rogati '01]
[Rochery, Schapire, Rahim & Gupta '01]
[Merler, Furlanello, Larcher & Sboner '01]
⋮

- continuing development of theory and algorithms:

[Breiman '98, '99]
[Schapire, Freund, Bartlett & Lee '98]
[Grove & Schuurmans '98]
[Mason, Bartlett & Baxter '98]
[Schapire & Singer '99]
[Cohen & Singer '99]
[Freund & Mason '99]
[Domingo & Watanabe '99]
[Mason, Baxter, Bartlett & Frean '99, '00]
[Duffy & Helmbold '99, '02]
[Freund & Mason '99]
[Ridgeway, Madigan & Richardson '99]
[Kivinen & Warmuth '99]
[Friedman, Hastie & Tibshirani '00]
[Rätsch, Onoda & Müller '00]
[Rätsch, Warmuth, Mika, Onoda, Lemm & Müller '00]
[Allwein, Schapire & Singer '00]
[Friedman '01]
[Koltchinskii, Panchenko & Lozano '01]
[Collins, Schapire & Singer '02]
[Demiriz, Bennett & Shawe-Taylor '02]
[Lebanon & Lafferty '02]
⋮

Adaboost (Adaptive Boosting)

"A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting"

[Freund-Schapire, JCSS'97]

Godel Prize winner 2003

Informal Description Adaboost

- Boosting: turns a weak algo into a strong (PAC) learner.

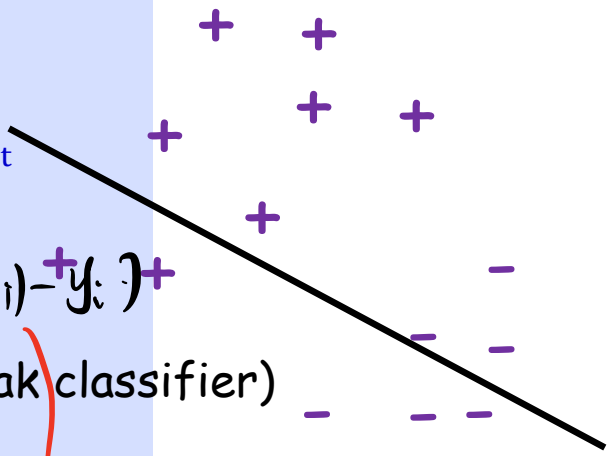
Input: $S=\{(x_1, y_1), \ldots, (x_m, y_m)\}$; $x_i \in X, y_i \in Y = \{-1,1\}$

weak learning algo A (e.g., Naïve Bayes, decision stumps)

- For $t=1,2, \ldots, T$
  - Construct D_t on $\{x_1, \ldots, x_m\}$
  - Run A on D_t producing $h_t: X \to \{-1,1\}$ (weak classifier)

$\epsilon_t = P_{x_i \sim D_t}(h_t(x_i) \neq y_i)$ error of h_t over D_t

- Output $H_{final}(x) = \mathrm{sign}(\sum_{t=1} \alpha_t h_t(x))$



Roughly speaking D_{t+1} increases weight on x_i if h_t incorrect on x_i ; decreases it on x_i if h_t correct.

Adaboost (Adaptive Boosting)

- Weak learning algorithm A.
- For t=1,2, … ,T
  - **Construct $\mathbf{D_t}$ on $\{x_1, \ldots, x_m\}$**
  - Run *A* on D_t producing h_t

Constructing D_t

- D_1 uniform on $\{x_1, \ldots, x_m\}$ [i.e., $D_1(i) = \frac{1}{m}$]
- Given D_t and h_t set

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

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

权重减少 (因为已分对)

分类错误 => 提高权重

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

规化因子 使 $\sum P(x) = 1$

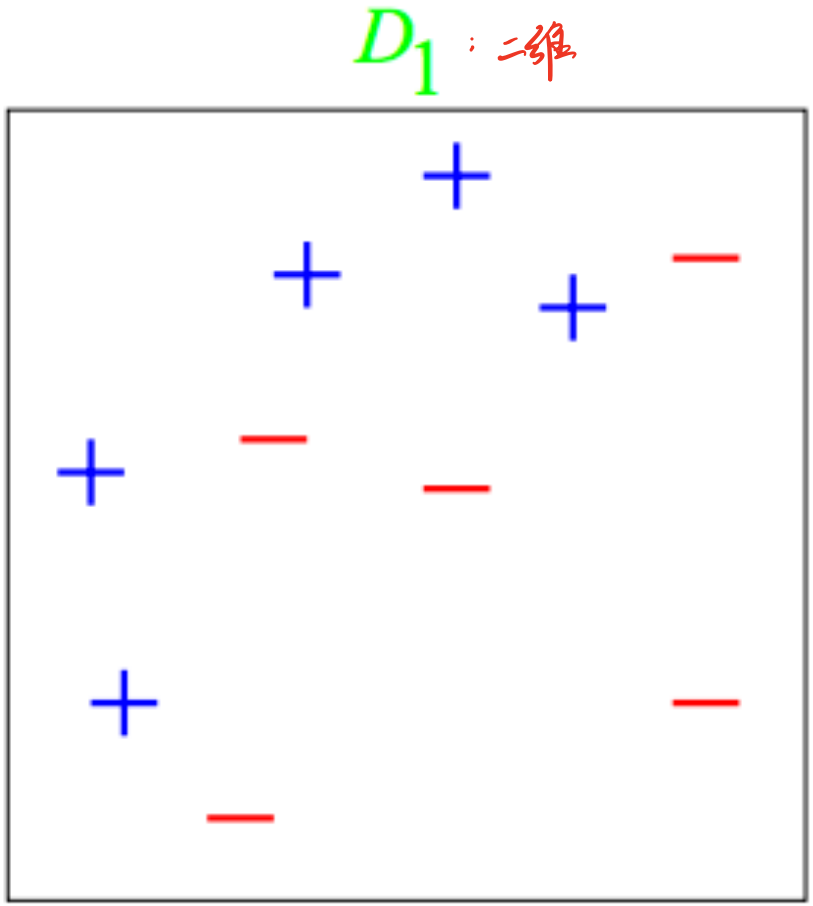
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

D_{t+1} puts half of weight on examples x_i where h_t is incorrect & half on examples where h_t is correct

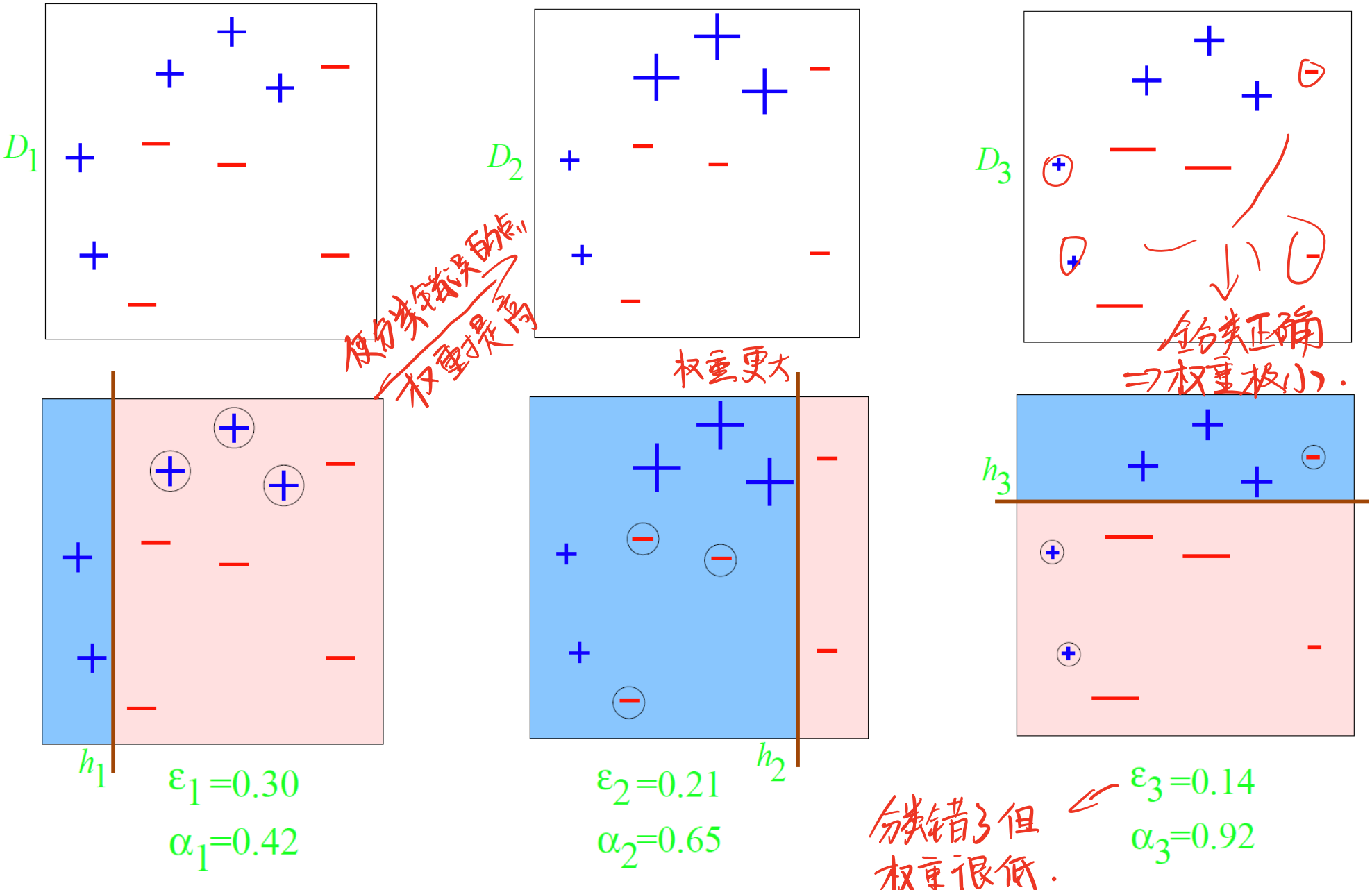
Final hyp: $H_{final}(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

Adaboost: A toy example

Weak classifiers: vertical or horizontal half-planes (a.k.a. decision stumps)

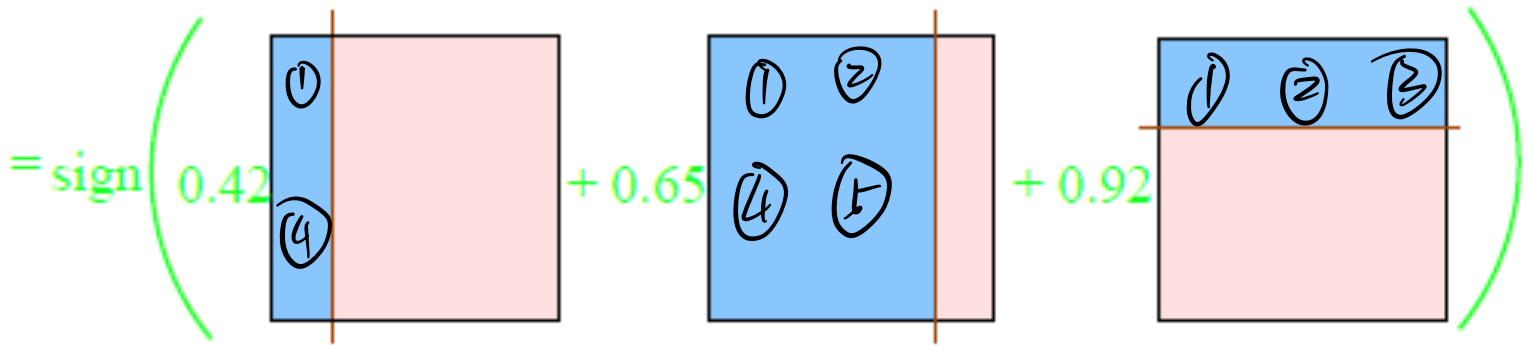


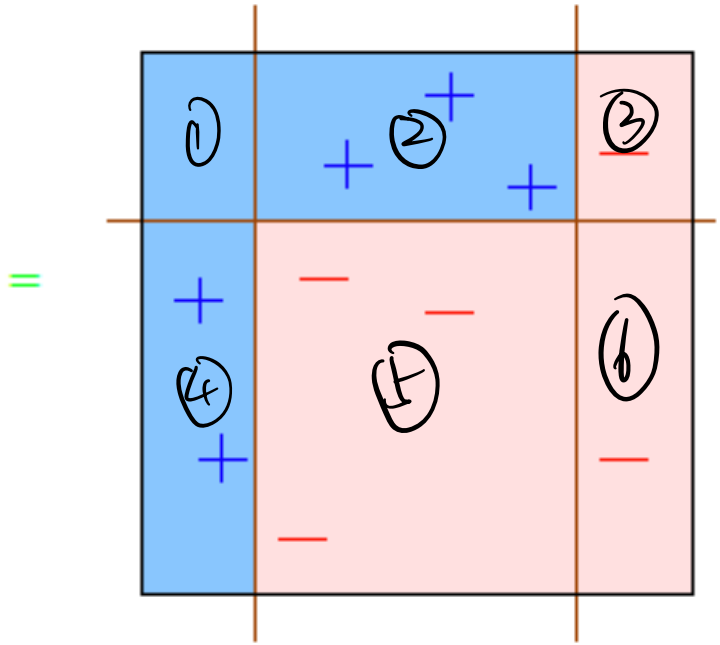
Adaboost: A toy example



Adaboost: A toy example

H_{final} 假设有下=3个分类器，那么对这=者进行多数投票.





把每一块的权重相加去看.

Adaboost (Adaptive Boosting)

- Weak learning algorithm A.
- For t=1,2, … ,T
  - **Construct $\mathbf{D_t}$ on $\{x_1, \ldots, x_m\}$**
  - Run *A* on D_t producing h_t

Constructing D_t

- D_1 uniform on $\{x_1, \ldots, x_m\}$ [i.e., $D_1(i) = \frac{1}{m}$]
- Given D_t and h_t set

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

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

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$$

D_{t+1} puts half of weight on examples x_i where h_t is incorrect & half on examples where h_t is correct

Final hyp: $H_{final}(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

缺点：①对异常点，敏感 ②对样本数量要求高.

Nice Features of Adaboost

- Very general: a meta-procedure, it can use any weak learning algorithm!!! (e.g., Naïve Bayes, decision stumps)

  用强分类器也可以但是提升不大

- Very fast (single pass through data each round) & simple to code, no parameters to tune.
- Shift in mindset: goal is now just to find classifiers a bit better than random guessing.
- Grounded in rich theory.
- Relevant for big data age: quickly focuses on "core difficulties", well-suited to distributed settings, where data must be communicated efficiently [Balcan-Blum-Fine-Mansour COLT'12].

Analyzing Training Error

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$$

训练误差. γ_t: 第t次的γ

So, if $\forall t, \gamma_t \geq \gamma > 0$, then $err_S(H_{final}) \leq \exp[-2\gamma^2 T]$

有一个下确界. 指数级 => Loss下降快但对异常值敏感

The training error drops exponentially in T!!!

假设 $err_S(H_{final}) \leq e^{-2\gamma^2 T} \leq \epsilon$

则 $T \geq \frac{1}{2\gamma^2}\ln\frac{1}{\epsilon}$

To get $err_S(H_{final}) \leq \epsilon$, need only $T = O\left(\frac{1}{\gamma^2}\log\left(\frac{1}{\epsilon}\right)\right)$ rounds

Adaboost is adaptive

- Does not need to know γ or T a priori
- Can exploit $\gamma_t \gg \gamma$

Understanding the Updates & Normalization

Claim: D_{t+1} puts half of the weight on x_i where h_t was incorrect and half of the weight on x_i where h_t was correct.

每次分类，将一半权重分给错误分类的，另一半给分类正确的

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

Probabilities are equal!

$\epsilon_t = \mathbb{E}_{D_t}[I[h_t(x_i) \neq y_i]] = P_{D_t}(h_t(x_i) \neq y_i) = \sum_{i=1}^{n} D_t(x_i) I[h_t(x_i) \neq y_i]$

$$\Pr_{D_{t+1}}[y_i \neq h_t(x_i)] = \sum_{i:y_i \neq h_t(x_i)} \frac{D_t(i)}{Z_t} e^{\alpha_t} = \epsilon_t \frac{1}{Z_t} e^{\alpha_t} = \frac{\epsilon_t}{Z_t} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{Z_t}$$

$$\Pr_{D_{t+1}}[y_i = h_t(x_i)] = \sum_{i:y_i = h_t(x_i)} \frac{D_t(i)}{Z_t} e^{-\alpha_t} = \frac{1-\epsilon_t}{Z_t} e^{-\alpha_t} = \frac{1-\epsilon_t}{Z_t} \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} = \frac{\sqrt{(1-\epsilon_t)\epsilon_t}}{Z_t}$$

$P(\text{correct}) = P(\text{error})$ 且有 Z_t $\Rightarrow$ $P(\text{correct}) + P(\text{error}) = 1$ 因为分类正确的点更多

故每个点，相乘也会偏小。

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

规范因子

$$= (1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Analyzing Training Error: Proof Intuition

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$$

- On round t, we increase weight of x_i for which h_t is wrong.
- If H_{final} incorrectly classifies x_i,
  - Then x_i incorrectly classified by (wtd) majority of h_t's.
  - Which implies final prob. weight of x_i is large.

    Can show probability $\geq \frac{1}{m}\left(\frac{1}{\prod_t Z_t}\right)$
- Since sum of prob. $= 1$, can't have too many of high weight.

  Can show # incorrectly classified $\leq m\,(\prod_t Z_t)$.

  And $(\prod_t Z_t) \to 0$.

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$

where $f(x_i) = \sum_t \alpha_t h_t(x_i)$. [Unthresholded weighted vote of h_i on x_i]

Step 2: $\mathrm{err}_S(H_{final}) \leq \prod_t Z_t$.

Step 3: $\prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}$

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$ where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

初始 递归关系

Recall $D_1(i) = \frac{1}{m}$ and $D_{t+1}(i) = D_t(i)\frac{\exp(-y_i \alpha_t h_t(x_i))}{Z_t}$

$$D_{T+1}(i) = \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times D_T(i)$$

$$= \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times \frac{\exp(-y_i \alpha_{T-1} h_{T-1}(x_i))}{Z_{T-1}} \times D_{T-1}(i)$$

……

$$= \frac{\exp(-y_i \alpha_T h_T(x_i))}{Z_T} \times \cdots \times \frac{\exp(-y_i \alpha_1 h_1(x_i))}{Z_1}\frac{1}{m}$$

$$= \frac{1}{m}\frac{\exp(-y_i(\alpha_1 h_1(x_i) + \cdots + \alpha_T h_T(x_T)))}{Z_1 \cdots Z_T} = \frac{1}{m}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$

where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i)$

$H(x_i) = \text{sign}(f(x_i))$

Step 2: $\text{err}_S(H_{final}) \leq \prod_t Z_t$.

$$\text{err}_S(H_{final}) = \frac{1}{m}\sum_i 1_{y_i \neq H_{final}(x_i)}$$

$$= \frac{1}{m}\sum_i 1_{y_i f(x_i) \leq 0}$$

预测结果与观测结果不一致

$$\leq \frac{1}{m}\sum_i \exp(-y_i f(x_i))$$

是上界.

代入

$$= \sum_i D_{T+1}(i) \prod_t Z_t = \prod_t Z_t.$$

用指数损失逼近0-1损失.

Analyzing Training Error: Proof Math

Step 1: unwrapping recurrence: $D_{T+1}(i) = \frac{1}{m}\left(\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}\right)$ where $f(x_i) = \sum_t \alpha_t h_t(x_i)$.

Step 2: $\mathrm{err}_S(H_{final}) \leq \prod_t Z_t$.

$1-x \leq e^{-x} \Rightarrow \sqrt{1-x} \leq e^{-\frac{x}{2}}$

$\epsilon_t = \frac{1}{2} - \gamma_t$

Step 3: $\prod_t Z_t = \prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}$

Note: recall $Z_t = (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

α_t minimizer of $\alpha \to (1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha}$

Analyzing Training Error: Proof Intuition

Theorem $\epsilon_t = 1/2 - \gamma_t$ (error of h_t over D_t)

$$err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$$

- On round t, we increase weight of x_i for which h_t is wrong.
- If H_{final} incorrectly classifies x_i,
  - Then x_i incorrectly classified by (wtd) majority of h_t's.
  - Which implies final prob. weight of x_i is large.

Can show probability $\geq \frac{1}{m}\left(\frac{1}{\prod_t Z_t}\right)$ $D_{t+1}(i) = \frac{1}{m}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$

- Since sum of prob. $= 1$, can't have too many of high weight.

Can show # incorrectly classified $\leq m\,(\prod_t Z_t)$.

And $(\prod_t Z_t) \to 0$.

↳否则概率之和超过1

Generalization Guarantees

Theorem $err_S(H_{final}) \leq \exp\left[-2\sum_t \gamma_t^2\right]$ where $\epsilon_t = 1/2 - \gamma_t$

How about generalization guarantees?

Original analysis [Freund&Schapire'97]

- H space of weak hypotheses; d=VCdim(H)

  H_{final} is a weighted vote, so the hypothesis class is:

  G={all fns of the form $\text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$ }

Theorem [Freund&Schapire'97]

$\forall\, g \in G, err(g) \leq err_S(g) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$ T= # of rounds

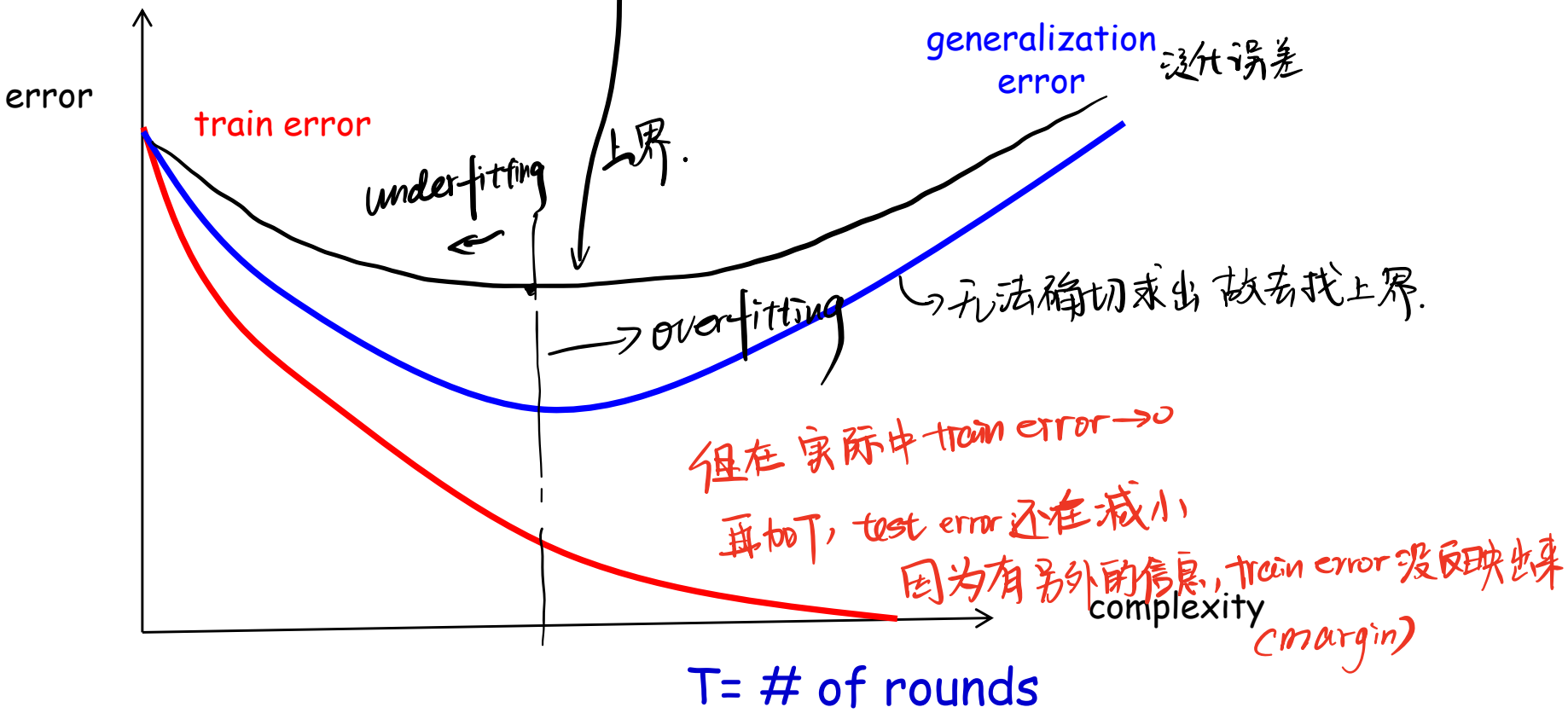
迭代次数 · d 描述算法复杂度 · 样本个数

Key reason: $\text{VCdim}(G) = \tilde{O}(dT)$ plus typical VC bounds.

Generalization Guarantees

Theorem [Freund&Schapire'97]

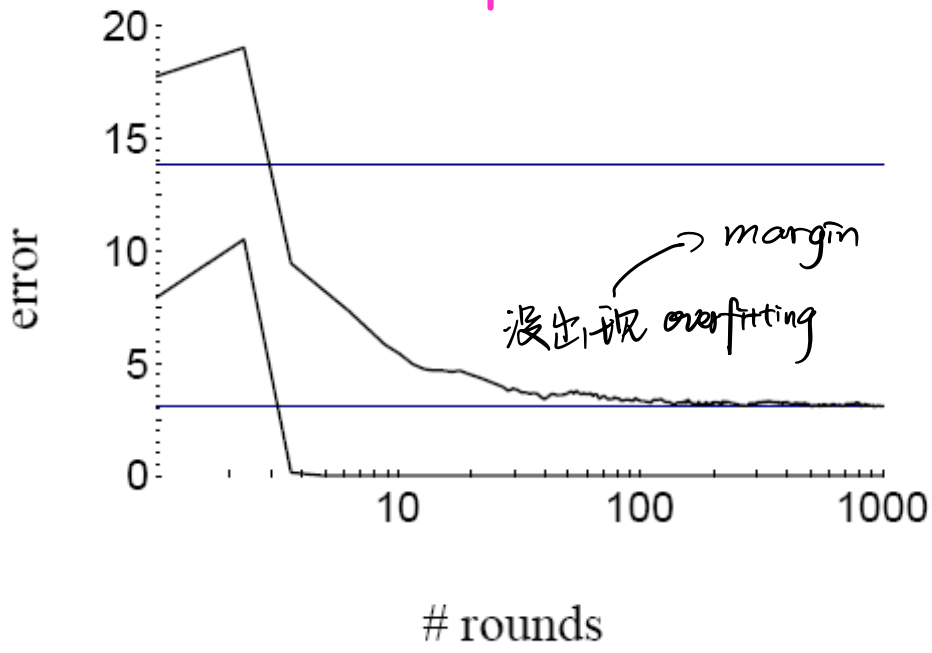
$$\forall\, g \in co(H), err(g) \le \underbrace{err_S(g)}_{\text{train error}} + \underbrace{\tilde{O}\left(\sqrt{\frac{Td}{m}}\right)}_{\text{模型复杂度}} \quad \text{where d=VCdim(H)}$$



Generalization Guarantees

- Experiments with boosting showed that the test error of the generated classifier usually does not increase as its size becomes very large.

- Experiments showed that continuing to add new weak learners after correct classification of the training set had been achieved could further improve test set performance!!!



Generalization Guarantees

- Experiments with boosting showed that the test error of the generated classifier usually does not increase as its size becomes very large.
- Experiments showed that continuing to add weak learners after correct classification of the training set had been achieved could further improve test set performance!!!
- These results seem to contradict FS'87 bound and Occam's razor (in order to achieve good test error the classifier should be as simple as




How can we explain the experiments?

R. Schapire, Y. Freund, P. Bartlett, W. S. Lee. present in "*Boosting the margin: A new explanation for the effectiveness of voting methods*" a nice theoretical explanation.

Key Idea:

Training error does not tell the whole story.

We need also to consider the classification confidence!!

$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

但 $\sum_{t=1}^{T} \alpha_t h_t(x)$ 可以是 1, 0.1, 0.001.

即 0.001 有极大可能变成负的.

$|f(x)|$ 越大，置信度越高.

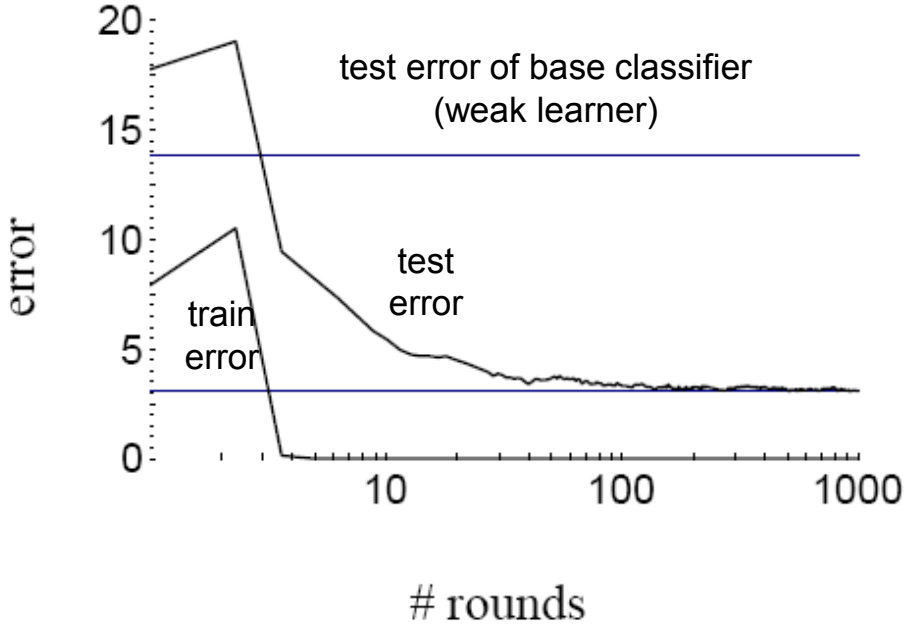
训练误差概率: $err_S = Pr_S(y \neq H(x))$

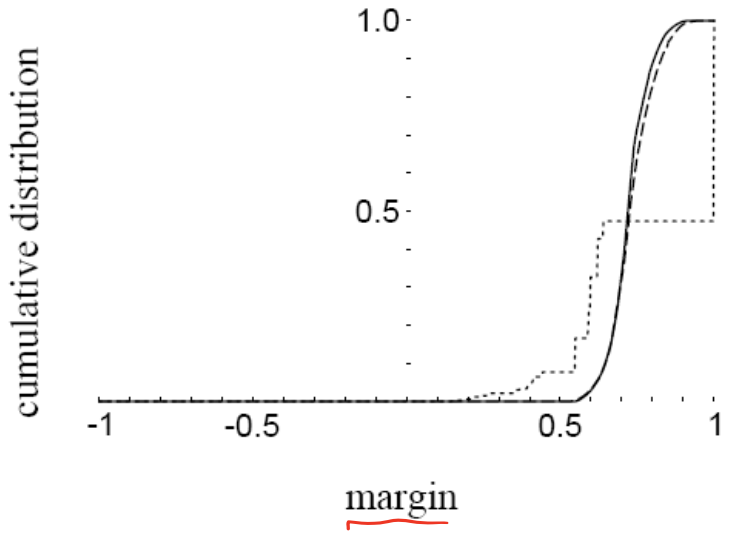
$= Pr_S(yH(x) < 0) = Pr_S(yf(x) < 0)$

margin : $yf(x)$ 函数间隔.

Boosting didn't seem to overfit…(!)

…because it turned out to be increasing the *margin* of the classifier





Error Curve, Margin Distr. Graph - Plots from [SFBL98]

Classification Margin

对 α_t 归一化：$\alpha_t = \frac{\alpha_t}{\sum_{t=1} \alpha_t}$ 使 α_t 变成类似概率的形式.

- H space of weak hypotheses. The convex hull of H:

$$co(H) = \{f = \sum_{t=1}^{T} \alpha_t h_t, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t = 1, h_t \in H\}$$

- Let $f \in co(H), f = \sum_{t=1}^{T} \alpha_t h_t, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t = 1.$

  The majority vote rule H_f given by f (given by $H_f = sign(f(x))$) predicts wrongly on example (x, y) iff $yf(x) \leq 0$.

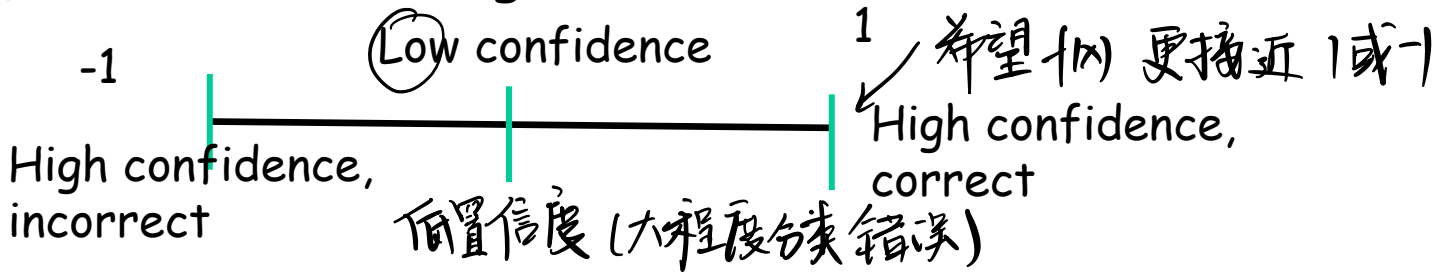
Definition: margin of H_f (or of f) on example (x, y) to be $yf(x)$.

$$yf(x) = y\sum_{t=1}^{T}[\alpha_t h_t(x)] = \sum_{t=1}^{T}[y\alpha_t h_t(x)] = \sum_{t:y=h_t(x)} \alpha_t - \sum_{t:y\neq h_t(x)} \alpha_t$$

分类正确 分类错误

The margin is positive iff $y = H_f(x)$.

See $|yf(x)| = |f(x)|$ as the strength or the confidence of the vote.



Boosting and Margins

之前与T相关
现在是与θ(margin)相关

Theorem: $\text{VCdim}(H) = d$, then with prob. $\geq 1 - \delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{d\ln^2\frac{m}{d}}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

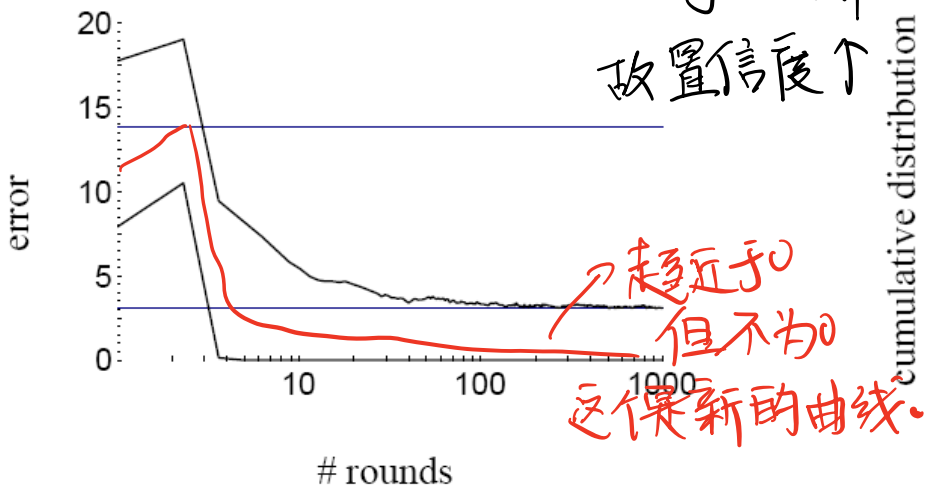
$err_S = \Pr_S(yf(x) \leq 0)$

S: 数据集

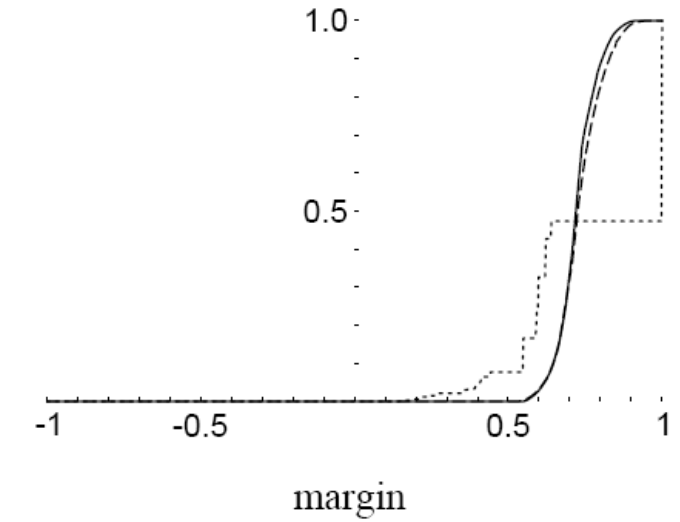
=> 只保留线性部分(去非线性部分) => $O\left(\sqrt{\frac{1}{m}}\right)$

Note: bound does **not** depend on T (the # of rounds of boosting), depends only on the complex. of the weak hyp space and the margin!

随训练次数↑
margin 更靠右
故置信度↑



趋近于0
但不为0
这个是新的曲线.



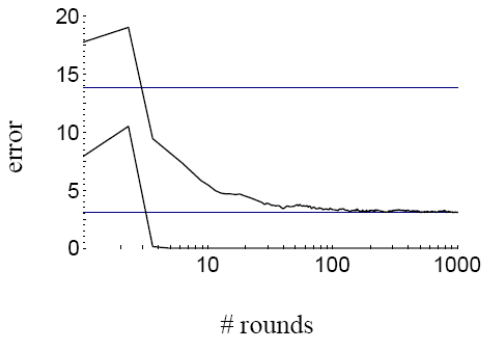
Quiz: according to this slide, explain why adaboost keeps decreasing testing error, even if training error equals to zero.

Boosting and Margins

Theorem: VCdim$(H) = d$, then with prob. $\geq 1 - \delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}} \sqrt{\frac{\mathrm{d} \ln^2 \frac{m}{d}}{\theta^2} + \ln \frac{1}{\delta}}\right)$$

- If all training examples have large margins, then we can approximate the final classifier by a much smaller classifier.
- Can use this to prove that better margin ➔ smaller test error, regardless of the number of weak classifiers.
- Can also prove that boosting tends to increase the margin of training examples by concentrating on those of smallest margin.
- Although final classifier is getting larger, margins are likely to be increasing, so the final classifier is actually getting closer to a simpler classifier, driving down test error.

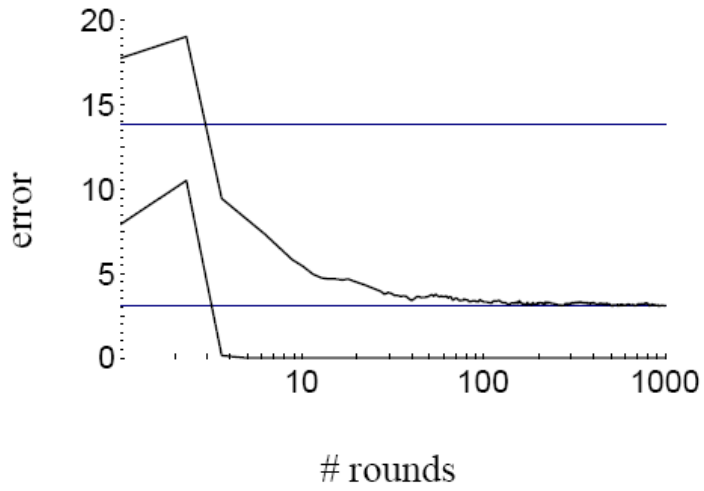


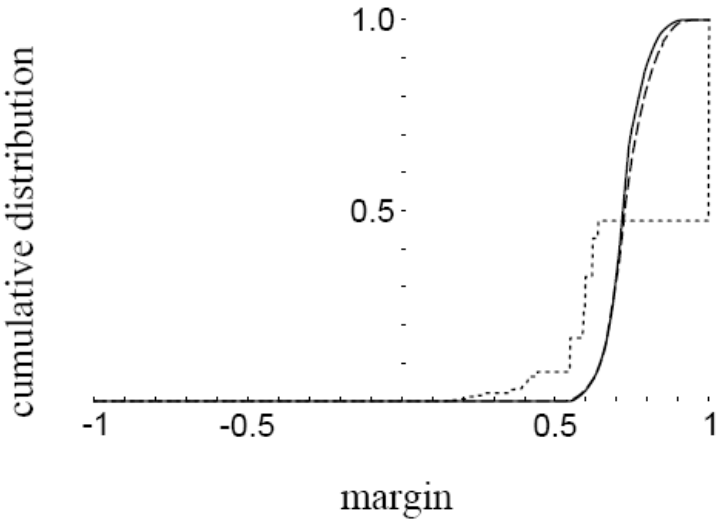
Boosting and Margins

Theorem: VCdim$(H) = d$, then with prob. $\geq 1 - \delta$, $\forall f \in co(H)$, $\forall \theta > 0$,

$$\Pr_D[yf(x) \leq 0] \leq \Pr_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\sqrt{\frac{\mathrm{d}\ln^2\frac{m}{d}}{\theta^2} + \ln\frac{1}{\delta}}\right)$$

Note: bound does **not** depend on T (the # of rounds of boosting), depends only on the complex. of the weak hyp space and the margin!





Boosting, Adaboost Summary

- Shift in mindset: goal is now just to find classifiers a bit better than random guessing.
- Backed up by solid foundations.
- Adaboost work and its variations well in practice with many kinds of data (one of the top 10 ML algos).
- More about classic applications in Recitation.

- Relevant for big data age: quickly focuses on "core difficulties", so well-suited to distributed settings, where data must be communicated efficiently [Balcan-Blum-Fine-Mansour COLT'12].

Interestingly, the usefulness of margin recognized in Machine Learning since late 50's.

Perceptron [Rosenblatt'57] analyzed via geometric (aka L_2, L_2) margin.

根据协方差优化.

在线学习. 之前是Batch Learning.

Original guarantee in the online learning scenario.

The Perceptron Algorithm

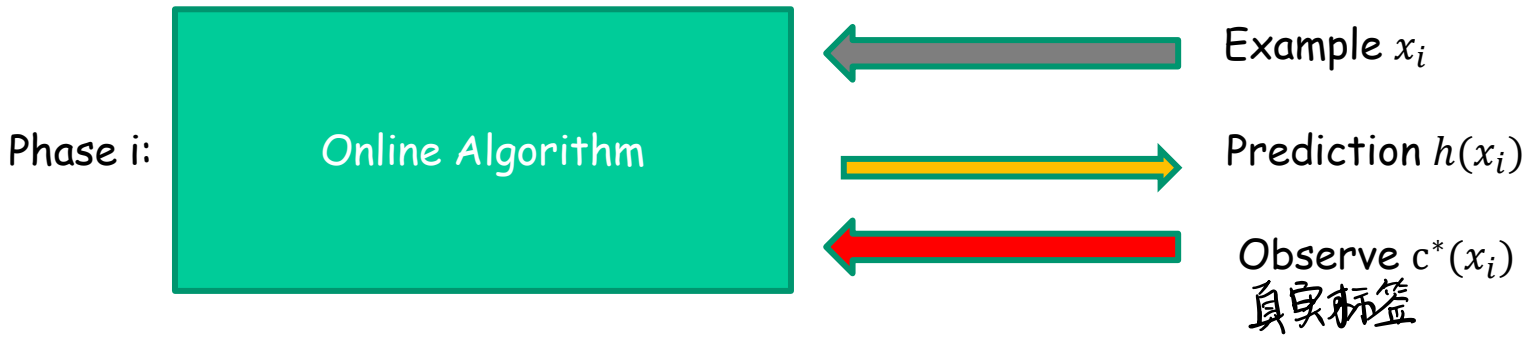
- Online Learning Model
- Margin Analysis
- Kernels

The Online Learning Model

- Example arrive sequentially.
- We need to make a prediction.

  Afterwards observe the outcome.

For i=1, 2, …, :



Mistake bound model

不对数据分布做假设 ⇒ 不能统计概率.

- Analysis wise, make no distributional assumptions.
- **Goal**: Minimize the number of mistakes.

最小化分类出错的次数

The Online Learning Model. Motivation

- Email classification (distribution of both spam and regular mail changes over time, but the target function stays fixed - last year's spam still looks like spam).

- Recommendation systems. Recommending movies, etc.

- Predicting whether a user will be interested in a new news article or not.

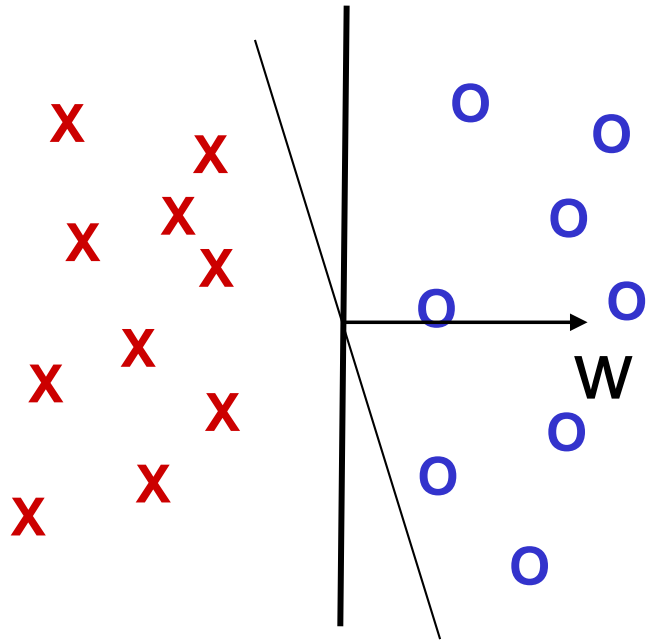
- Add placement in a new market.

Linear Separators

- Instance space $X = R^d$
- Hypothesis class of linear decision surfaces in R^d.

$w^T x$ w, x：d维向量.

- $h(x) = w \cdot x + w_0$, if $h(x) \geq 0$, then label x as +, otherwise label it as -



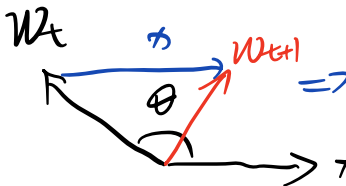
Claim: WLOG $w_0 = 0$.

Proof: Can simulate a non-zero threshold with a dummy input feature x_0 that is always set up to 1.

- $x = (x_1, \ldots, x_d) \rightarrow \tilde{x} = (x_1, \ldots, x_d, 1)$
- $w \cdot x + w_0 \geq 0$ iff $(w_1, \ldots, w_d, w_0) \cdot \tilde{x} \geq 0$

where $w = (w_1, \ldots, w_d)$

假设：二维空间：



$\Rightarrow w_t \cdot x < 0$

真实值：>0 $\Rightarrow$ mistake on positive $w_{t+1} = w_t + x \Rightarrow w_{t+1} \cdot x > 0$

但经常是一次拉不回来，需多次迭代拉回来.

Linear Separators: Perceptron Algorithm

真实值<0 => mistake on negative $w_{t+1} = w_t - x$

- Set t=1, start with the all zero vector w_1.
- Given example x, predict positive iff $w_t \cdot x \geq 0$
- On a mistake, update as follows: 如果没出差错：$w_{t+1} \leftarrow w_t$.
  - Mistake on positive, then update $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, then update $w_{t+1} \leftarrow w_t - x$

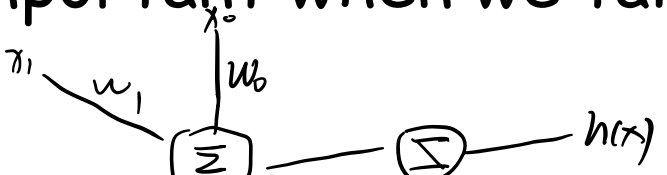
或用 $w_{t+1} = w_t + y_t \cdot y \in \{+1, -1\}$ 是统合后的形式.

Note: w_t is weighted sum of incorrectly classified examples

$$w_t = a_{i_1} x_{i_1} + \cdots + a_{i_k} x_{i_k}$$

$$w_t \cdot x = a_{i_1} x_{i_1} \cdot x + \cdots + a_{i_k} x_{i_k} \cdot x$$

Important when we talk about kernels.

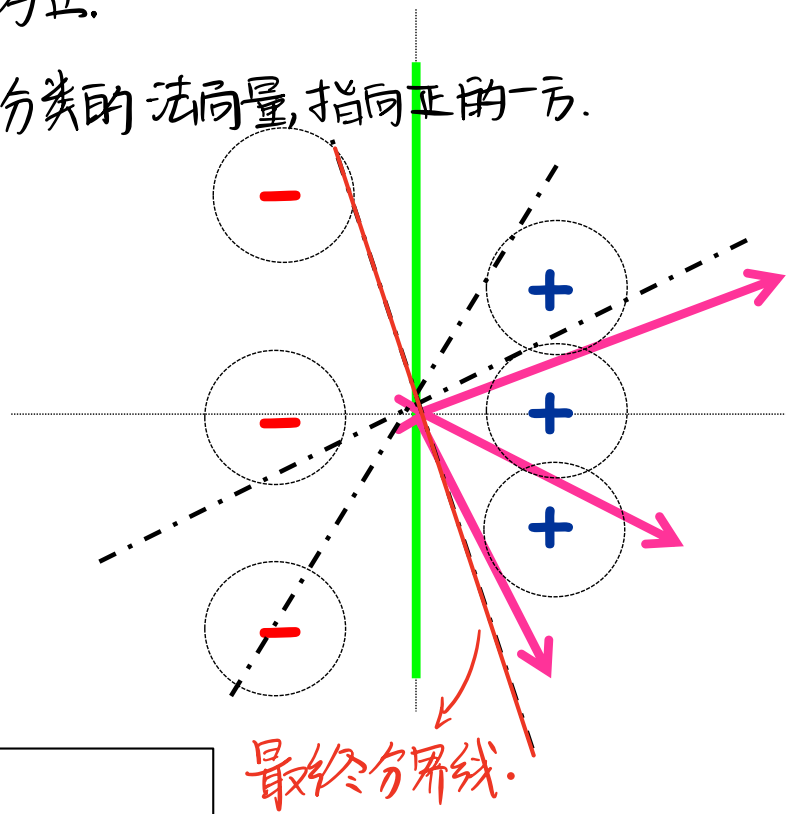


神经元

Perceptron Algorithm: Example

initialize $w_1 = (0,0)$ 认为0的符号为正.

	x	y	$\text{sign}(w^T x)$	
Example:	$(-1,2)$	$-$	$+$	X
	$(1,0)$	$+$	$+$	✓
	$(1,1)$	$+$	$-$	X
	$(-1,0)$	$-$	$-$	✓
	$(-1,-2)$	$-$	$+$	X
	$(1,-1)$	$+$	$+$	✓



Algorithm:

- Set t=1, start with all-zeroes weight vector w_1.
- Given example x, predict positive iff $w_t \cdot x \geq 0$.
  - On a mistake, update as follows:
    - Mistake on positive, update $w_{t+1} \leftarrow w_t + x$
    - Mistake on negative, update $w_{t+1} \leftarrow w_t - x$

$w_1 = (0,0)$

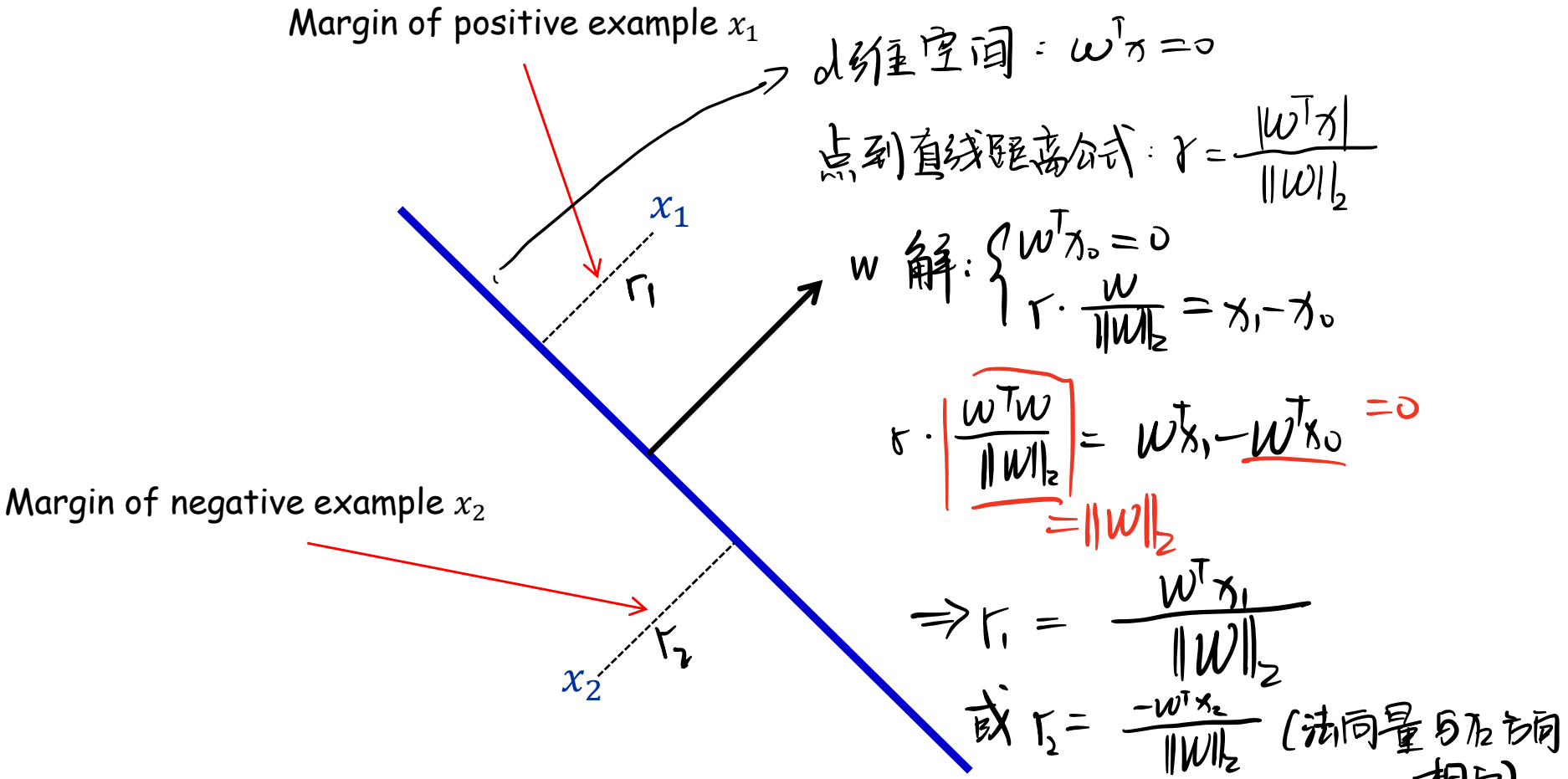
$w_2 = w_1 - (-1,2) = (1,-2)$

$w_3 = w_2 + (1,1) = (2,-1)$

$w_4 = w_3 - (-1,-2) = (3,1)$

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)



d维空间：$w^T x = 0$

点到直线距离公式：$\gamma = \frac{|w^T x|}{\|w\|_2}$

w 解：$\begin{cases} w^T x_0 = 0 \\ r \cdot \frac{w}{\|w\|_2} = x_1 - x_0 \end{cases}$

$r \cdot \left| \frac{w^T w}{\|w\|_2} \right| = w^T x_1 - w^T x_0 = 0$

$= \|w\|_2$

$\Rightarrow r_1 = \frac{w^T x_1}{\|w\|_2}$

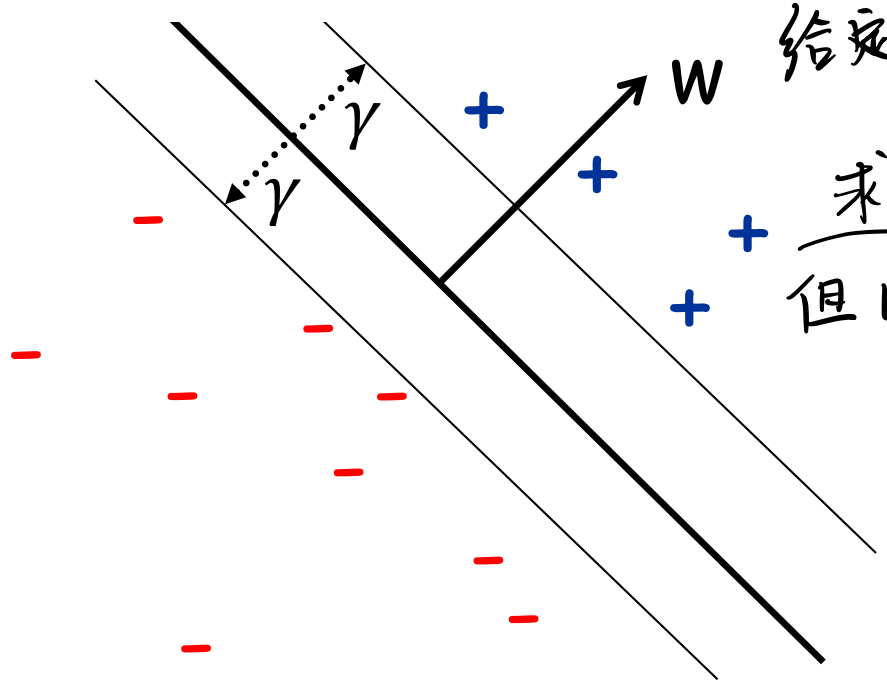
或 $r_2 = \frac{-w^T x_2}{\|w\|_2}$ (法向量与方向相反)

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.

Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.



给定数据集合，$\gamma_w = \min_{x_i \in S} \frac{|w^T x_i|}{\|w\|_2}$

↓

求最小间隔

但 w 可以改变 $\Rightarrow \gamma = \max_w \gamma_w$

$\Rightarrow \gamma = \max_{w \in R^d} \min_{x_i \in S} \frac{y_i w^T x_i}{\|w\|_2}$

起到取绝对值的作用

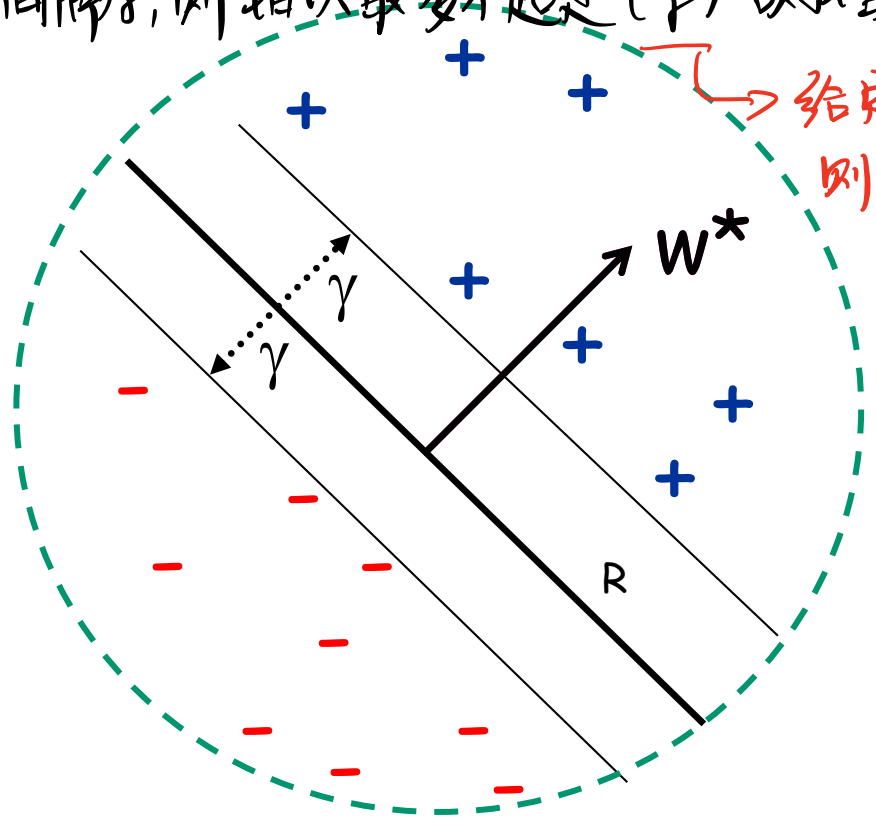
$y_i \in \{+1, -1\}$

Perceptron: Mistake Bound

Theorem: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)

所有点在R半径球内 间隔γ，则错误最多不超过$(\frac{R}{\gamma})^2$ 故找最大γ



→ 给定最大间隔的w是w^*

则有 $\gamma = \min_{x_i \in S} \frac{y_i w^{*T} x_i}{\|w^*\|_2}$

Perceptron Algorithm: Analysis

Theorem: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

Update rule:

- Mistake on positive: $w_{t+1} \leftarrow w_t + x$
- Mistake on negative: $w_{t+1} \leftarrow w_t - x$

$w_{t+1} = w_t + yx \Rightarrow w_{t+1} \cdot w^* = w_t \cdot w^* + yxw^*$

由于定义了 $\gamma = \min \frac{yw^*x}{\|w^*\|}$ 故 $yxw^* \geq \gamma$

Proof:

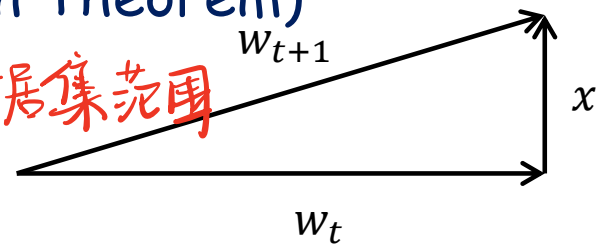
Idea: analyze $w_t \cdot w^*$ and $\|w_t\|$, where w^* is the max-margin sep, $\|w^*\| = 1$.

Claim 1: $w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma$. (because $l(x)x \cdot w^* \geq \gamma$)

Claim 2: $\|w_{t+1}\|^2 \leq \|w_t\|^2 + R^2$. (by Pythagorean Theorem)

$\|w_t + yx\|_2^2 = \|w_t\|_2^2 + 2\|w_t yx\|_2 + \|yx\|_2^2 \leq \|w_t\|_2^2 + R^2$ 数据集范围

$2\|w_t yx\|_2 < 0$ (预测错误) $\|yx\|_2^2 \leq R^2$

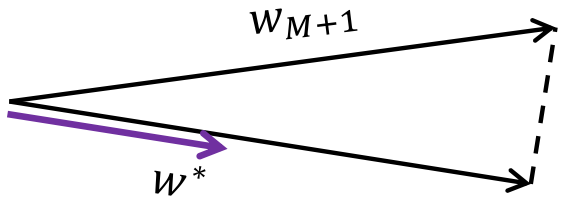


After M mistakes:

$w_{M+1} \cdot w^* \geq \gamma M$ (by Claim 1)

$\|w_{M+1}\| \leq R\sqrt{M}$ (by Claim 2)

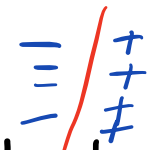
$w_{M+1} \cdot w^* \leq \|w_{M+1}\|$ (since w^* is unit length)



So, $\gamma M \leq R\sqrt{M}$, so $M \leq \left(\frac{R}{\gamma}\right)^2$.

Perceptron Extensions

- Can use it to find a consistent separator (by cycling through the data).



- One can convert the mistake bound guarantee into a distributional guarantee too (for the case where the x_is come from a fixed distribution).

假如是线性可分 ⇒ 一定有一个一致的分类器. 且循环不超过 $(\frac{R}{\gamma})^2$ 次 → 总错误数

- Can be adapted to the case where there is no perfect separator as long as the so called hinge loss (i.e., the total distance needed to move the points to classify them correctly large margin) is small. 若有少量误差, 则若 $\|w^T x\| < \varepsilon$ 则认为是一个正确的分类器. (soft margin)

- Can be kernelized to handle non-linear decision boundaries!

核化.

若线性不可分: 使用内核化的方式拓展到高维空间使线性可分

后映射回低维空间.

Perceptron Discussion

- Simple online algorithm for learning linear separators with a nice guarantee that depends only on the geometric (aka L_2, L_2) margin.

- It can be kernelized to handle non-linear decision boundaries --- see next class!

- Simple, but very useful in applications like Branch prediction; it also has interesting extensions to structured prediction.