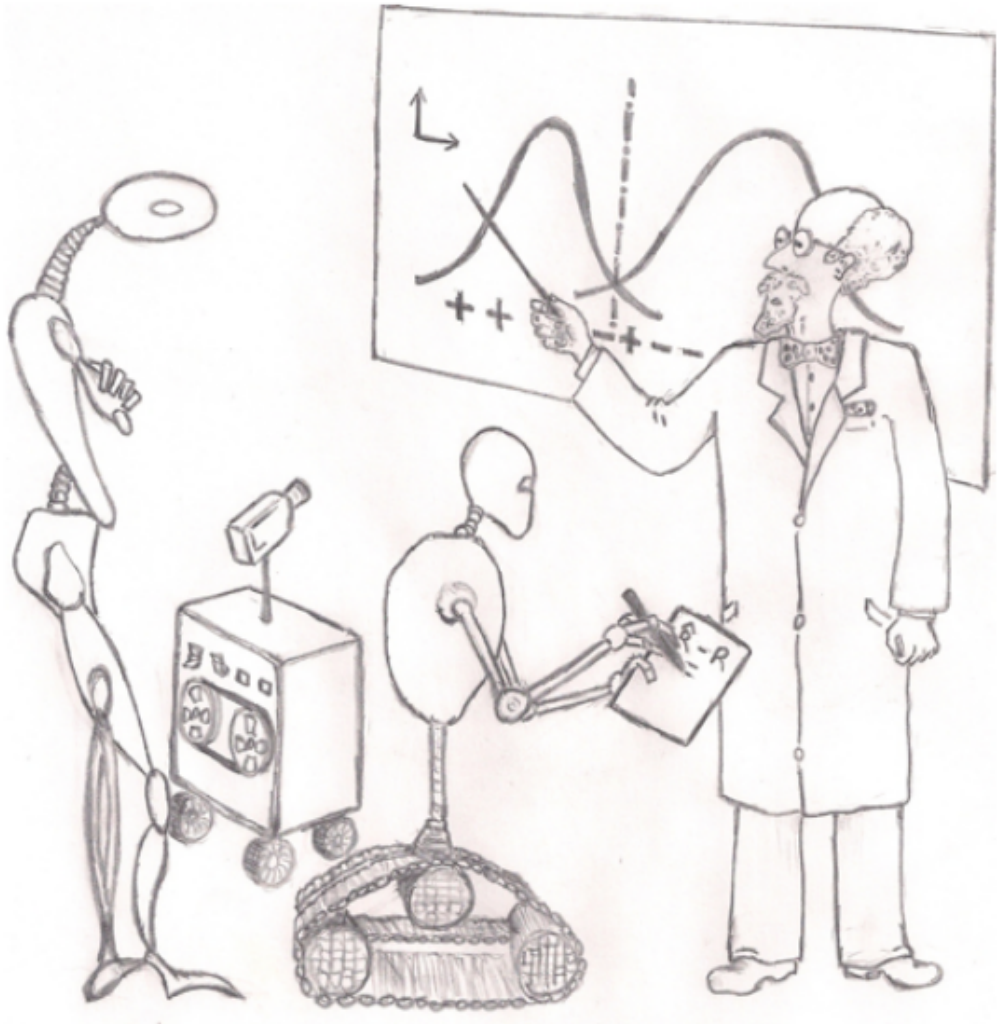
یادگیری برخط

چه چیزی را می‌توان یاد گرفت؟

بهراد منیری

دانشجوی دکترای دانشگاه پنسیلوانیا

bemoniri@seas.upenn.edu



عکس از شای شالِو شوارتز

هدف

- یادگیری
- یادگیری ماشین
- یادگیری برخط

یادگیری چیست؟

یادگیری چیست؟

کسب **دانش** با استفاده از **تجربه**.

چرا ماشین باید یاد بگیرد؟

چرا ماشین باید یاد بگیرد؟

- برنامه‌نویسی مستقیم برخی کارها سخت است!

چرا ماشین باید یاد بگیرد؟

- برنامه‌نویسی مستقیم برخی کارها سخت است!
- **بینایی ماشین**: سگ و گربه را از هم تمییز می‌دهیم، اما چگونه؟

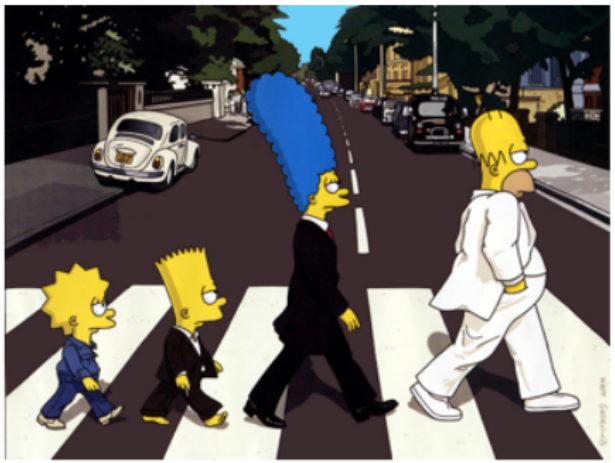
چرا ماشین باید یاد بگیرد؟

- برنامه‌نویسی مستقیم برخی کارها سخت است!
- **بینایی ماشین**: سگ و گربه را از هم تمییز می‌دهیم، اما چگونه؟
- **موتورهای جست و جو**: داده‌ها زیاد است! چگونه یاد بگیریم؟!

مثال – تشخیص عابر پیاده

مثال - تشخیص عابر پیاده

مثال - تشخیص عابر پیاده



حفظ کردن چیزهای خوب!

- **تعمیم پذیری**: باید در شرایط نادیده، خوب عمل کنیم.

حفظ کردن چیزهای خوب!

- **تعمیم پذیری**: باید در شرایط نادیده، خوب عمل کنیم.

- **سوال‌های اساسی**: چگونه یاد بگیریم؟ چه چیزهای قابل یادگرفتن هستند؟

تیغ اُکام

به نظر می‌رسد که توضیحی ساده برای یک پدیده، از یک توضیح پیچیده بهتر است.

– ویلیامِ اُکام، منطق‌دان. قرن ۱۴ میلادی، انگلستان.

ناهار مجانی نداریم!

بدون دانش پیشین، یادگیری ناممکن است.

یادگیری برخط!

یادگیری برخط: یادگیری ترتیبی. جواب دادن به یک سوال، بر مبنای پاسخ صحیح سوال‌های زمان قبل.

یادگیری برخط!

یادگیری برخط: یادگیری ترتیبی. جواب دادن به یک سوال، بر مبنای پاسخ صحیح سوال‌های زمان قبل.

یک مدل ریاضی

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{۰, ۱\}$ را حدس بزنیم.

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{۰, ۱\}$ را حدس بزنیم.
- **زمان t:**

۱. اطلاعات زمان t، یعنی $x_t \in \mathbb{R}$ به یادگیرنده داده می‌شود.

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{۰, ۱\}$ را حدس بزنیم.
- **زمان t:**

۱. اطلاعات زمان t، یعنی $x_t \in \mathbb{R}$ به یادگیرنده داده می‌شود.

۲. بر اساس x_t و اطلاعات و برچسب‌های درست زمان‌های قبل، یادگیرنده A، حدس زیر را برای برچسب زمان t می‌زند:

$$\hat{y}_t = A\big(x_t; (x_۱, y_۱), \ldots, (x_{t-۱}, y_{t-۱})\big)$$

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{0, 1\}$ را حدس بزنیم.
- **زمان t:**

۱. اطلاعات زمان t، یعنی $x_t \in \mathbb{R}$ به یادگیرنده داده می‌شود.

۲. بر اساس x_t و اطلاعات و برچسب‌های درست زمان‌های قبل، یادگیرنده A، حدس زیر را برای برچسب زمان t می‌زند:

$$\hat{y}_t = A\big(x_t; (x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\big)$$

۳. برچسب واقعی y_t به یادگیرنده داده می‌شود.

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{۰, ۱\}$ را حدس بزنیم.
- **زمان t:**

۱. اطلاعات زمان t، یعنی $x_t \in \mathbb{R}$ به یادگیرنده داده می‌شود.

۲. بر اساس x_t و اطلاعات و برچسب‌های درست زمان‌های قبل، یادگیرنده A، حدس زیر را برای برچسب زمان t می‌زند:

$$\hat{y}_t = A\big(x_t; (x_۱, y_۱), \ldots, (x_{t-۱}, y_{t-۱})\big)$$

۳. برچسب واقعی y_t به یادگیرنده داده می‌شود.

اینجا هم ناهار مجانی نداریم!

بدون هیچ فرضی، یادگیری ممکن نیست.

یک مدل ریاضی

- در زمان t، قصد داریم یک برچسب، $y_t \in \{0, 1\}$ را حدس بزنیم.
- **زمان t:**

۱. اطلاعات زمان t، یعنی $x_t \in \mathbb{R}$ به یادگیرنده داده می‌شود.

۲. بر اساس x_t و اطلاعات و برچسب‌های درست زمان‌های قبل، یادگیرنده A، حدس زیر را برای برچسب زمان t می‌زند:

$$\hat{y}_t = A\big(x_t; (x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\big)$$

۳. برچسب واقعی y_t به یادگیرنده داده می‌شود.

اینجا هم ناهار مجانی نداریم!

بدون هیچ فرضی، یادگیری ممکن نیست.

یک فرض ساده: در هر زمان t داریم $y_t = h^*(x_t)$، برای یک $h^* \in \mathcal{H}$.

یک مدل ریاضی

- **ورودی:** $\mathbf{X} = (x_1, \ldots, x_T)$
- **دانش پیشین:** مجموعه‌ی $\mathcal{H}$ از توابع $\mathbb{R}$ به $\{0, 1\}$.
- **خروجی:** $h^*(x_1), \ldots, h^*(x_T)$ برای یک تابع $h^* \in \mathcal{H}$ که یادگیرنده از آن اطلاع ندارد.

یک مدل ریاضی

- **ورودی:** $\mathbf{X} = (x_1, \ldots, x_T)$
- **دانش پیشین:** مجموعه‌ی $\mathcal{H}$ از توابع $\mathbb{R}$ به $\{0, 1\}$.
- **خروجی:** $h^*(x_1), \ldots, h^*(x_T)$ برای یک تابع $h^* \in \mathcal{H}$ که یادگیرنده از آن اطلاع ندارد.

یک مدل ریاضی

- **ورودی:** $\mathbf{X} = (x_1, \ldots, x_T)$
- **دانش پیشین:** مجموعه‌ی $\mathcal{H}$ از توابع $\mathbb{R}$ به $\{0, 1\}$.
- **خروجی:** $h^*(x_1), \ldots, h^*(x_T)$ برای یک تابع $h^* \in \mathcal{H}$ که یادگیرنده از آن اطلاع ندارد.

سوال: یک الگوریتم چند خطا مرتکب می‌شود؟

$$\mathbb{1}\{\hat{y}_1 \neq y_1\} + \mathbb{1}\{\hat{y}_2 \neq y_2\} + \cdots + \mathbb{1}\{\hat{y}_T \neq y_T\}$$

یک مدل ریاضی

- **ورودی:** $\mathbf{X} = (x_1, \ldots, x_T)$
- **دانش پیشین:** مجموعه‌ی $\mathcal{H}$ از توابع $\mathbb{R}$ به $\{0, 1\}$.
- **خروجی:** $h^*(x_1), \ldots, h^*(x_T)$ برای یک تابع $h^* \in \mathcal{H}$ که یادگیرنده از آن اطلاع ندارد.

سوال: یک الگوریتم چند خطا مرتکب می‌شود؟

$$\mathbb{1}\{\hat{y}_1 \neq y_1\} + \mathbb{1}\{\hat{y}_2 \neq y_2\} + \cdots + \mathbb{1}\{\hat{y}_T \neq y_T\}$$

هدف: الگوریتم A را به نحوی بیابیم که کمیت زیر را کمینه کند:

$$\max_{\mathbf{X},\, h^*} \left[\mathbb{1}\{\hat{y}_1 \neq y_1\} + \cdots + \mathbb{1}\{\hat{y}_T \neq y_T\} \right]$$

حالت ساده: فرض کنید $|\mathcal{H}| < \infty$.

حالت ساده: فرض کنید $|\mathcal{H}| < \infty$.

Consistent Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: choose *any* $h \in V_1$
5: predict $\hat{y}_t = h(x_t)$
6: receive true label y_t
7: update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$
8: **end for**

حالت ساده: فرض کنید $|\mathcal{H}| < \infty$.

Consistent Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: choose *any* $h \in V_1$
5: predict $\hat{y}_t = h(x_t)$
6: receive true label y_t
7: update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$
8: **end for**

تحلیل: این الگوریتم حداکثر $|\mathcal{H}| - 1$ خطا مرتکب می‌شود.

حالت ساده: فرض کنید $|\mathcal{H}| < \infty$.

Consistent Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: choose *any* $h \in V_1$
5: predict $\hat{y}_t = h(x_t)$
6: receive true label y_t
7: update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$
8: **end for**

تحلیل: این الگوریتم حداکثر $|\mathcal{H}| - 1$ خطا مرتکب می‌شود.

سوال: آیا می‌توان بهتر عمل کرد؟

Halving Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: set

$$\begin{cases} V_t^{(0)} = \{h \in V_t : h(x_t) = 0\} \\ V_t^{(1)} = \{h \in V_t : h(x_t) = 1\} \end{cases}$$

5: predict

$$\hat{y}_t = \begin{cases} 1 & |V_t^{(0)}| \leq |V_t^{(1)}| \\ 0 & |V_t^{(0)}| > |V_t^{(1)}| \end{cases}$$

6: receive true label y_t
7: update $V_{t+1} = V_t^{(y_t)}$
8: **end for**

Halving Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: set

$$\begin{cases} V_t^{(0)} = \{h \in V_t : h(x_t) = 0\} \\ V_t^{(1)} = \{h \in V_t : h(x_t) = 1\} \end{cases}$$

5: predict

$$\hat{y}_t = \begin{cases} 1 & |V_t^{(0)}| \leq |V_t^{(1)}| \\ 0 & |V_t^{(0)}| > |V_t^{(1)}| \end{cases}$$

6: receive true label y_t
7: update $V_{t+1} = V_t^{(y_t)}$
8: **end for**

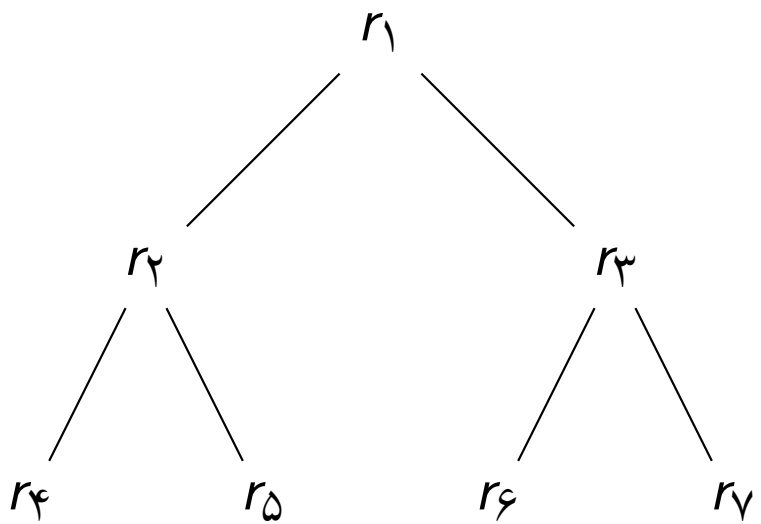
تحلیل: این الگوریتم حداکثر $\log_2(|\mathcal{H}|)$ خطا مرتکب می‌شود.

یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یک استراتژی برای شیطان:

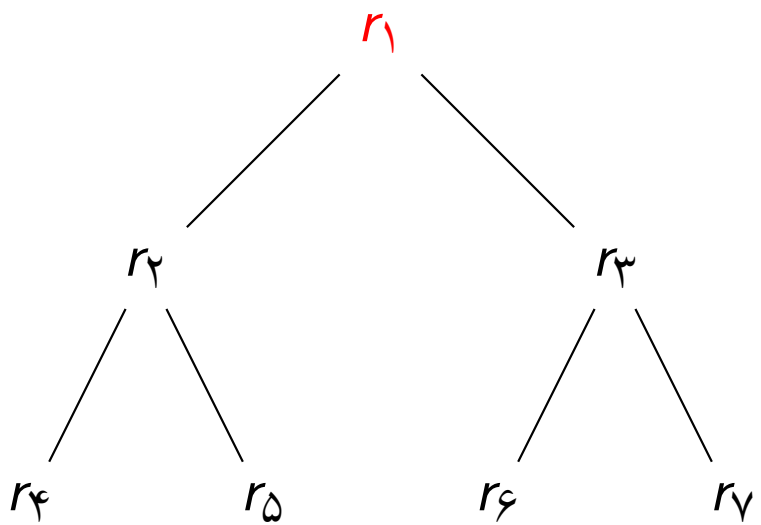
$t = ۰$



یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یک استراتژی برای شیطان:

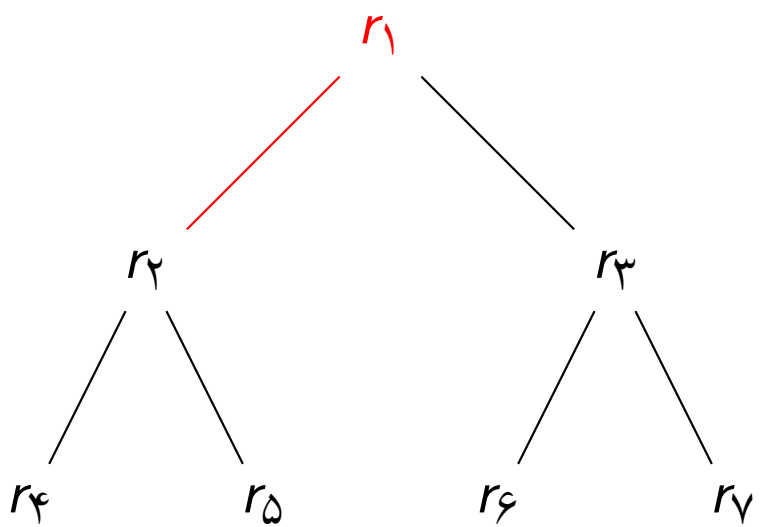
$t = ۱; \quad x_۱ = r_۱$



یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یک استراتژی برای شیطان:

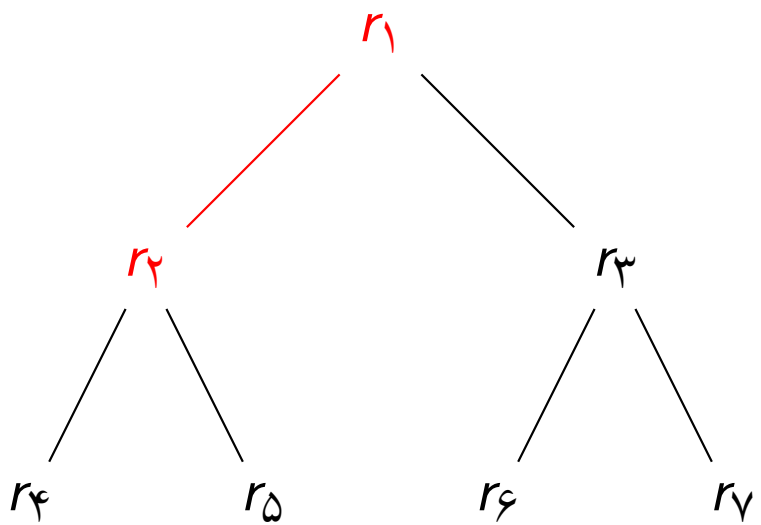
$$t = ۱; \quad x_۱ = r_۱; \quad \hat{y}_۱ = ۱; \quad y_۱ = ۰$$



یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یک استراتژی برای شیطان:

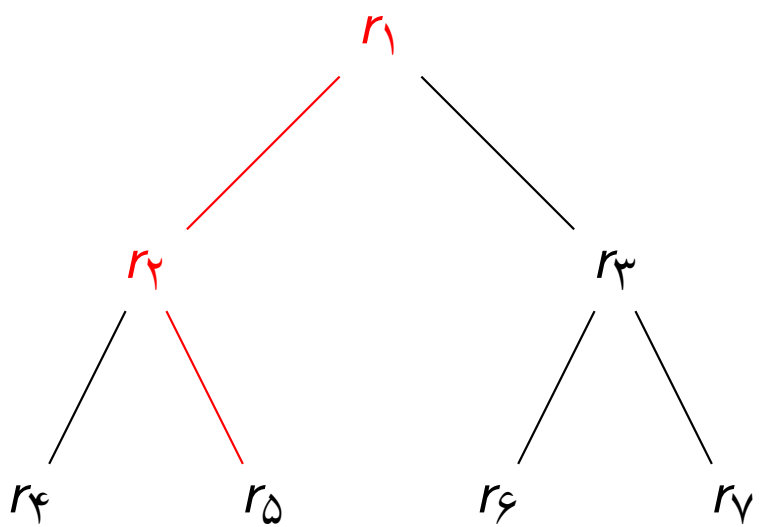
$t = ۲; \quad x_۲ = r_۲$



یادگیری برخط یک بازی دو نفره است بین شیطان و یادگیرنده!

یک استراتژی برای شیطان:

$t = ۲;\quad x_۲ = r_۲;\quad \hat{y}_۲ = ۰;\quad y_۲ = ۱$



- هر مسیر روی درخت از ریشه به برگ‌ها، متناظر است با یک $(y_1, \ldots, y_T)$.

- هر مسیر روی درخت از ریشه به برگ‌ها، متناظر است با یک $(y_1, \ldots, y_T)$.
- این استراتژی برای شیطان موفقیت‌آمیز است، اگر برای هر $(y_1, \ldots, y_T)$، وجود داشته باشد $h \in \mathcal{H}$ به نحوی که در هر $t \leq T$ داشته باشیم $y_t = h(x_t)$.

- هر مسیر روی درخت از ریشه به برگ‌ها، متناظر است با یک $(y_1, \ldots, y_T)$.
- این استراتژی برای شیطان موفقیت‌آمیز است، اگر برای هر $(y_1, \ldots, y_T)$، وجود داشته باشد $h \in \mathcal{H}$ به نحوی که در هر $t \leq T$ داشته باشیم $y_t = h(x_t)$.
- **درخت شقّه‌شده**: یک درخت به عمق d توسط $\mathcal{H}$ شقّه می‌شود اگر ...

- هر مسیر روی درخت از ریشه به برگ‌ها، متناظر است با یک $(y_1, \ldots, y_T)$.
- این استراتژی برای شیطان موفقیت‌آمیز است، اگر برای هر $(y_1, \ldots, y_T)$، وجود داشته باشد $h \in \mathcal{H}$ به نحوی که در هر $t \leq T$ داشته باشیم $y_t = h(x_t)$.
- **درخت شقّه‌شده**: یک درخت به عمق d توسط $\mathcal{H}$ شقّه می‌شود اگر ...

تعریف: بُعد لیتل‌استون

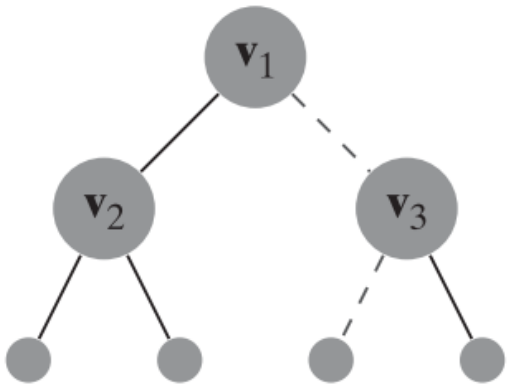
$Ldim(\mathcal{H})$ بزرگترین عدد صحیح T است، به نحوی که یک درخت شقّه‌شده به عمق T برای $\mathcal{H}$ وجود داشته باشد.

- هر مسیر روی درخت از ریشه به برگ‌ها، متناظر است با یک $(y_1, \ldots, y_T)$.
- این استراتژی برای شیطان موفقیت‌آمیز است، اگر برای هر $(y_1, \ldots, y_T)$، وجود داشته باشد $h \in \mathcal{H}$ به نحوی که در هر $t \leq T$ داشته باشیم $y_t = h(x_t)$.
- **درخت شقّه‌شده**: یک درخت به عمق d توسط $\mathcal{H}$ شقّه می‌شود اگر ...

تعریف: بُعد لیتل‌استون

$Ldim(\mathcal{H})$ بزرگترین عدد صحیح T است، به نحوی که یک درخت شقّه‌شده به عمق T برای $\mathcal{H}$ وجود داشته باشد.

- **مثال از محاسبه‌ی بعد لیتل‌استون**



	h_1	h_2	h_3	h_4
$\mathbf{v}_1$	0	0	1	1
$\mathbf{v}_2$	0	1	*	*
$\mathbf{v}_3$	*	*	0	1

قضیه‌ی معکوس

هر یادگیرنده‌ای را می‌توان متحمل تعداد $Ldim(\mathcal{H})$ خطا نمود.

قضیه‌ی معکوس

هر یادگیرنده‌ای را می‌توان متحمل تعداد $Ldim(\mathcal{H})$ خطا نمود.

- اثبات: شیطان با پیگیری استراتژی مطرح شده می‌تواند به این میزان خطا تحمیل کند.

قضیه‌ی معکوس

هر یادگیرنده‌ای را می‌توان متحمل تعداد $Ldim(\mathcal{H})$ خطا نمود.

- اثبات: شیطان با پیگیری استراتژی مطرح شده می‌تواند به این میزان خطا تحمیل کند.
- نیکولاس لیتل‌استون. رساله‌ی دکترا، دانشگاه کالیفرنیا در سانتا کروز، سال ۱۹۹۰.

قضیه‌ی معکوس

هر یادگیرنده‌ای را می‌توان متحمل تعداد $Ldim(\mathcal{H})$ خطا نمود.

- اثبات: شیطان با پیگیری استراتژی مطرح شده می‌تواند به این میزان خطا تحمیل کند.
- نیکولاس لیتل‌استون. رساله‌ی دکترا، دانشگاه کالیفرنیا در سانتا کروز، سال ۱۹۹۰.
- الان کجاست؟! از سال ۲۰۰۱ گم شده :)

خواص بعد لیتل‌استون

$$\mathcal{H}_1 \subseteq \mathcal{H}_2 \implies Ldim(\mathcal{H}_1) \leq Ldim(\mathcal{H}_2) \blacktriangleleft$$

خواص بعد لیتل‌استون

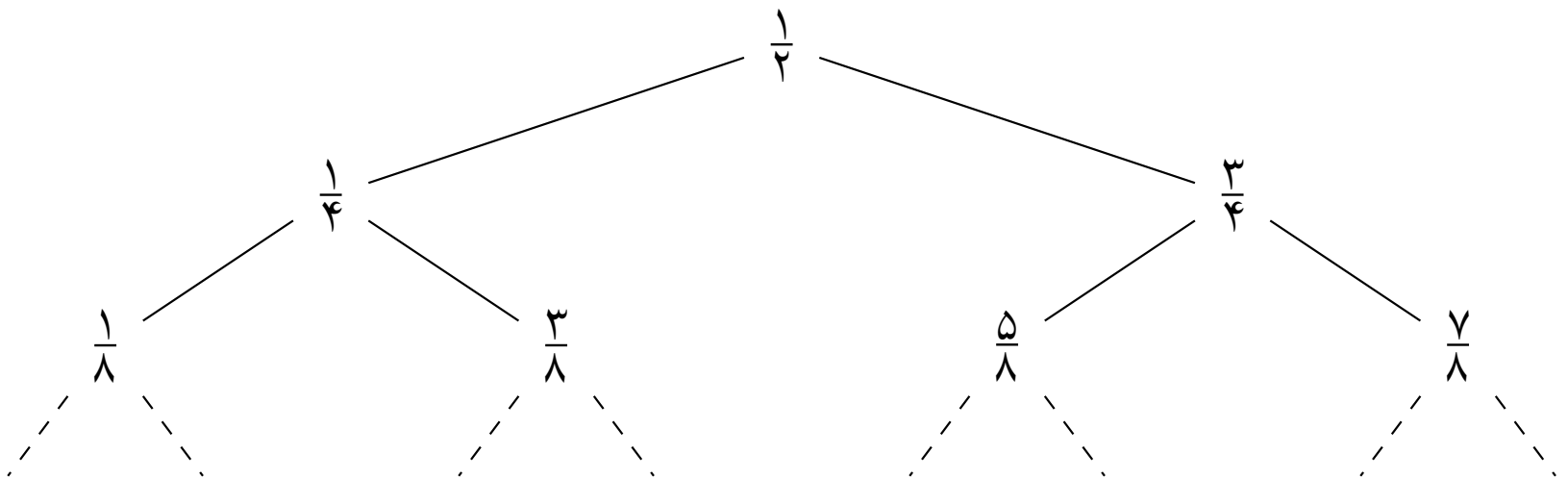
- $\mathcal{H}_1 \subseteq \mathcal{H}_2 \implies Ldim(\mathcal{H}_1) \leq Ldim(\mathcal{H}_2)$
- $2^{Ldim(\mathcal{H})} \leq |\mathcal{H}|$

خواص بعد لیتل‌استون

- $\mathcal{H}_1 \subseteq \mathcal{H}_2 \implies Ldim(\mathcal{H}_1) \leq Ldim(\mathcal{H}_2)$
- $2^{Ldim(\mathcal{H})} \leq |\mathcal{H}|$
- توابع ترشهلود: $\mathcal{H}_t = \{1_{[x<a]}(x) \mid a \in \mathbb{R}\}$
  - $Ldim(\mathcal{H}_t) = \infty$

خواص بعد لیتل‌استون

- $\mathcal{H}_۱ \subseteq \mathcal{H}_۲ \implies Ldim(\mathcal{H}_۱) \leq Ldim(\mathcal{H}_۲)$
- $۲^{Ldim(\mathcal{H})} \leq |\mathcal{H}|$
- توابع ترشهلود: $\mathcal{H}_t = \{ ۱_{[x<a]}(x) \mid a \in \mathbb{R} \}$
  - $Ldim(\mathcal{H}_t) = \infty$



◀ **سوال:** آیا الگوریتمی وجود دارد که متحمل تعداد خطایی کمتر یا مساوی بعد لیتل استون شود؟

Standard Optimal Algorithm

1: $V_1 = \mathcal{H}$
2: **for** $t = 1, 2, \ldots$ **do**
3: receive x_t
4: For $r = 1, 0$, set $V_t^{(r)} = \{h \in V_t : h(x_t) = r\}$
5: predict

$$\hat{y}_t = \arg \max_{r \in \{0,1\}} Ldim(V_t^{(r)})$$

6: receive true label y_t
7: update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$
8: **end for**

قضیه

الگوریتم فوق، حداکثر مرتکب $Ldim(\mathcal{H})$ خطا می‌شود.

اثبات بهینه بودن الگوریتم بهینه‌ی استاندارد

- کافی است نشان دهیم که هر بار که الگوریتم مرتکب خطا می‌شود، داریم

$$Ldim(V_{t+۱}) \leq Ldim(V_t) - ۱.$$

اثبات بهینه بودن الگوریتم بهینه‌ی استاندارد

- کافی است نشان دهیم که هر بار که الگوریتم مرتکب خطا می‌شود، داریم

$$Ldim(V_{t+۱}) \leq Ldim(V_t) - ۱.$$

- فرض می‌کنیم $Ldim(V_{t+۱}) = Ldim(V_t)$ وبه تناقض می‌رسیم (برهان خلف).

اثبات بهینه بودن الگوریتم بهینه‌ی استاندارد

- کافی است نشان دهیم که هر بار که الگوریتم مرتکب خطا می‌شود، داریم

$$Ldim(V_{t+1}) \leq Ldim(V_t) - 1.$$

- فرض می‌کنیم $Ldim(V_{t+1}) = Ldim(V_t)$ وبه تناقض می‌رسیم (برهان خلف).

$$\begin{aligned} Ldim(V_t) = Ldim(V_{t+1}) = Ldim(V_t^{(y_t)}) = Ldim(V_t^{(1-\hat{y}_t)}) \\ \leq Ldim(V_t^{(\hat{y}_t)}) \leq Ldim(V_t) \end{aligned}$$

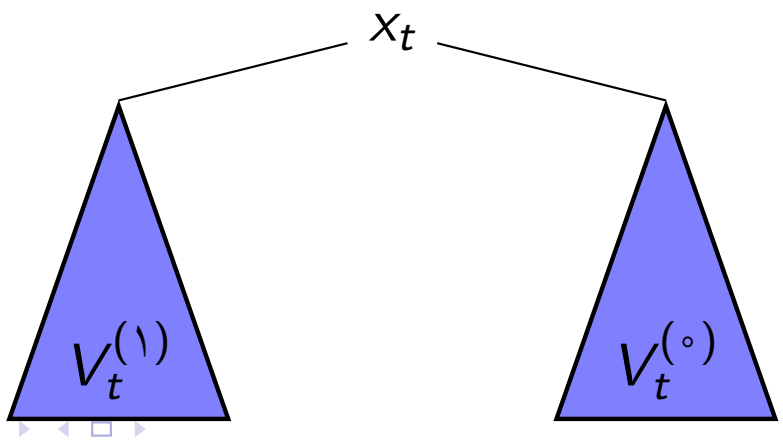
اثبات بهینه بودن الگوریتم بهینه‌ی استاندارد

- کافی است نشان دهیم که هر بار که الگوریتم مرتکب خطا می‌شود، داریم

$$Ldim(V_{t+1}) \leq Ldim(V_t) - 1.$$

- فرض می‌کنیم $Ldim(V_{t+1}) = Ldim(V_t)$ وبه تناقض می‌رسیم (برهان خلف).

$$\begin{aligned} Ldim(V_t) = Ldim(V_{t+1}) = Ldim(V_t^{(y_t)}) = Ldim(V_t^{(1-\hat{y}_t)}) \\ \leq Ldim(V_t^{(\hat{y}_t)}) \leq Ldim(V_t) \end{aligned}$$



چه چیزی یاد گرفتیم؟

- در تمام الگوریتم‌ها، در هر مرحله تعداد زیادی تابع وجود دارند که با نمونه‌های قبلی هم‌خوان هستند! اما فقط بعضی از آن‌ها آینده را درست پیش‌بینی می‌کنند.

چه چیزی یاد گرفتیم؟

- در تمام الگوریتم‌ها، در هر مرحله تعداد زیادی تابع وجود دارند که با نمونه‌های قبلی هم‌خوان هستند! اما فقط بعضی از آن‌ها آینده را درست پیش‌بینی می‌کنند.
- ناهار مجانی نداریم! مجبور شدیم $\mathcal{H}$ را در نظر بگیریم.

چه چیزی یاد گرفتیم؟

- در تمام الگوریتم‌ها، در هر مرحله تعداد زیادی تابع وجود دارند که با نمونه‌های قبلی هم‌خوان هستند! اما فقط بعضی از آن‌ها آینده را درست پیش‌بینی می‌کنند.
- ناهار مجانی نداریم! مجبور شدیم $\mathcal{H}$ را در نظر بگیریم.
- بعد لیتل‌استون معیاری است از پیچیدگی (غنی بودن) مجموعه‌ی توابع $\mathcal{H}$.

چه چیزی یاد گرفتیم؟

- در تمام الگوریتم‌ها، در هر مرحله تعداد زیادی تابع وجود دارند که با نمونه‌های قبلی هم‌خوان هستند! اما فقط بعضی از آن‌ها آینده را درست پیش‌بینی می‌کنند.
- ناهار مجانی نداریم! مجبور شدیم $\mathcal{H}$ را در نظر بگیریم.
- بعد لیتل‌استون معیاری است از پیچیدگی (غنی بودن) مجموعه‌ی توابع $\mathcal{H}$.
- هر چه مسئله پیچیده‌تر، یادگیری سخت‌تر (تیغ اُکام توجیه می‌شود)

کجا هنوز جای کار دارد؟

- فرض $h^* \in \mathcal{H}$ خیلی قوی است! شاید کمی نویز داشتیم!!

کجا هنوز جای کار دارد؟

- فرض $h^* \in \mathcal{H}$ خیلی قوی است! شاید کمی نویز داشتیم!!
- آیا الگوریتم بهینه‌ی استاندارد، از نظر محاسباتی هم بهینه است؟

کجا هنوز جای کار دارد؟

- فرض $h^* \in \mathcal{H}$ خیلی قوی است! شاید کمی نویز داشتیم!!
- آیا الگوریتم بهینه‌ی استاندارد، از نظر محاسباتی هم بهینه است؟
- شیطان در این مدل خیلی قوی است! دنیا انقدر هم سخت‌گیر نیست.

سوال؟!

سوال؟!

عصر به خیر!