Last time:
- discuss Winnow 1 → $F(\delta)$ "$\delta$-sep. mon LTFs" LTF's
- Winnow 2 alg: OLMB alg. for certain LTF's (with positive weights, $X=\{0,1\}^n$)
- Perceptron alg. for LTFs "with a margin" over $\mathbb{R}^n$

Today:
- finish PCT proof; example of using PCT
- dual Perceptron, kernelization — End of our "concrete examples"
- start generic bounds: "halving algorithm"

Questions?

---

Recall PCT setup:

$v$ = target vector $\|v\|=1$

$w$ = hyp. vector
↳ initially $0^n$

**Thm:** (Perc. Convergence Theorem):
Suppose run Perc. on seq of ex labeled by $\text{sign}(v\cdot x)$ (a2), where $\|v\|$ (a3), each $\|x\|$ (a2) are $=1$.
Let $\delta := \min_{\text{all ex. } x \text{ we get}} |v\cdot x|$. (= min Euclidean dist. from $x$ to $v$'s hyperplane)
Then # mist Perc. makes is $\leq \frac{1}{\delta^2}$.

L1: After $M$ mistakes, $v\cdot w \geq \delta\cdot M$

L2: After $M$ mist., $w\cdot w \leq M$.

$(\|w\| = \sqrt{w\cdot w} \leq \sqrt{M})$

## Proof of PCT:

(length of proj of $w$ in dir. $v$) $\leq$ (length of $w$)

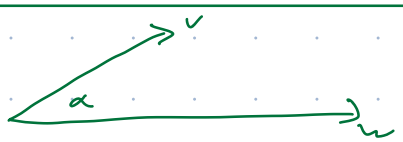
$\delta M \overset{L1}{\leq} v \cdot w \leq \|w\| = \sqrt{w \cdot w} \overset{L2}{\leq} \sqrt{M}$, so

$\delta \sqrt{M} \leq 1$, so $M \leq 1/\delta^2$. ∎

---

$v \cdot w = \|w\| \cdot \cos(\alpha)$

b/c $v$ = unit vector




---

## Discussion:

- \+ simple!
- \+ noise-tolerant
- \+ fast
- \+ "Kernelizable" (more soon...)
- $-\ 1/\delta^2$ : not great.

(better MB) Faster LTF learning known, with m.b. $\approx \log \frac{1}{\delta}$ but these lack nice properties.

---

## Ex: Class of "reoriented n-var MAJ's"

(n odd) $\mathcal{C} = \{MAJ(\ell_1, \ell_2, \ldots, \ell_n) : \ell_i \text{ either } x_i \text{ or } \overline{x_i}\}$

$|\mathcal{C}| = 2^n$ $\quad X = \{0,1\}^n$

ex: $n=3$ $\quad MAJ(x_1, \bar{x}_2, \bar{x}_3)$

To use PCT: want each ex $\|x\|=1$,
target $\|v\|=1$, $\quad sign(v\cdot x)$

Remap: $x \in \{0,1\}^n \quad 0 \to -\frac{1}{\sqrt{n}}$

$\downarrow \qquad 1 \to \frac{1}{\sqrt{n}}$.

$z \in \{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\}^n$.

every $z \in \{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\}^n$ has $\|z\|=1$. ☺

Have $\underline{sign(v\cdot z)} \equiv MAJ(\ell_1, \ldots, \ell_n)$ ($\ell_i$: $x_i$ or $\bar{x}_i$)

by $x \to z$ as above,

$$v_i = \begin{cases} 1 & \text{if } \ell_i = x_i \\ -1 & \text{if } \ell_i = \bar{x}_i \end{cases}$$

$v\cdot z$ adds up $\pm z_1 \pm z_2 \cdots \pm z_n$ $\quad +$ iff $MAJ(\ell_1, \ldots, \ell_n)=1$

$v \leftarrow \frac{1}{\sqrt{n}}$ this $v$:

$v_i = \pm\frac{1}{\sqrt{n}}$ so $\|v\|=1$

What's $\delta$? $\quad \delta = \min |v\cdot z|$

$\underline{v\cdot z} = \left(\frac{\pm 1}{\sqrt{n}} \cdot \frac{\pm 1}{\sqrt{n}}\right) + \ldots$

$n$ of these

$|v \cdot z| \geq \frac{1}{n}$ so $\delta = \frac{1}{n}$ + Perc. will make $\leq n^2$ mistakes.

---

## ① Dual Perceptron + ② Kernelization

↳ Different, ≡ way to run Perceptron

Clunkier... but enables "☺" - way to run Perc. over very high-dim feature space super-efficiently!

---

① Key insight: all it takes to run Perc. is being able to compute $x \cdot x'$ where $x, x'$ are example vectors.

Recall how Perc works: $w_{init} = (0, \ldots, 0)$

After 1st update, on example $x^1$, $w$ is either $x^1$ or $-x^1$.

To keep running Perceptron: on next ex $x$, $w \cdot x$ is $x^1 \cdot x$ or $-x^1 \cdot x$ ☺, by

$2^{nd}$ mist is on, say, $x^2$, + new $w$ is (say) $x^1 - x^2$

So $w \cdot x = x^1 \cdot x - x^2 \cdot x$ ☺ Can keep going! ☺

---

More formal descr. of dual Perc:

Hyp. maintained as list of pairs

$\bigstar$ $\{(x^1, y^1), \ldots, (x^k, y^k)\}$ $(x^i, y^i)$ = $i^{th}$ example Perc. got wrong.

(example; label $\in \{-1, 1\}$)

Given new example $x$, compute

$$w \cdot x = \sum_{i=1}^{k} y^i (x^i \cdot x).$$

Perfectly simulates Perceptron.

---

Why? • More memory-intense, slower :(

• :) :) :) We can replace "$\cdot$" (inner prod. over original ex) with any "Kernel function".

---

② Kernel functions & "feature expansions"

Let $X$ = original feature space ($\{0,1\}^n$ or $\mathbb{R}^n$).

$X'$ = new "high dim" feature space ($\{0,1\}^N$ or $\mathbb{R}^N$)

→ both spaces have inner products.

---

Def: A feature expansion $\Phi$ is a mapping $X \to X'$.

Ex: $X = \{0,1\}^n$, $X' = \{0,1\}^{3^n}$,

$\Phi$ = "all-conjunctions" feature expansion

$\Phi(x)$ has a coord for each of the $3^n$ conj. over $x_1, \ldots, x_n$.

$n=2$: $x = (x_1, x_2) \in \{0,1\}^2$

$\Phi(x)$ = 9-bit string

$= (1, x_1, \bar{x}_1, x_2, x_1 \wedge x_2, \bar{x}_1 \wedge x_2, \bar{x}_2, x_1 \wedge \bar{x}_2, \bar{x}_1 \wedge \bar{x}_2)$

(1 = empty conj)

$\Phi(x)$ = "expanded" version of $x$.

---

A kernel function $K$ corresp. to $\Phi$ is a fn

$K: X \times X \to \mathbb{R}$, $K(a,b) = \Phi(a) \cdot \Phi(b)$

($a \in X$, $b \in X$; $\Phi(a) \in X'$, $\Phi(b) \in X'$; $K(a,b) \in \mathbb{R}$)

---

Back to learning:

Suppose we get ex. in $X$ ($= \mathbb{R}^n$) but we want to run Perc. over $X'$ ($= \mathbb{R}^N$) using $\Phi(x)$ for each ex. $x$.

- Obvious approach: do it directly: for each $x$ we get, write down $\Phi(x)$, maintain $w \in \mathbb{R}^N$ as hyp. vector, run Perc. on $\Phi(x)$.
  Drawback: takes time $\geq N$ per example - slow!

- Better way: use dual Perc., replacing each $a \cdot b$ with $K(a,b)$.

This exactly simulates running Perc. on $\Phi(\cdot)$-versions of examples!

---

If we can evaluate $K(a,b)$ "fast", we can run Perc. over $X'$ fast!

---

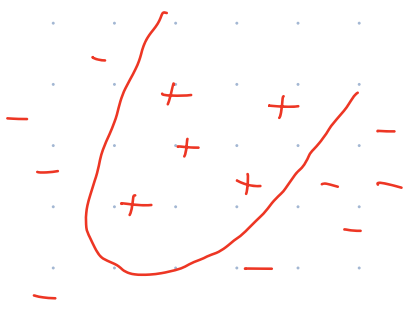
Nice to run Perc over feature-expanded space:

- over $\mathbb{R}^n$, Perc. hyp. is like $\text{sign}(3x_1+4x_2-5x_3)$

- If we did a feature exp. to

$$\Phi(x) = (1, x_1, \dots, x_n, x_1^2, x_1x_2, \dots x_1x_n, x_2^2, x_2x_3, \dots x_n^2)$$

Perc hyp. can be $\text{sign}(5x_1^2 - 4x_1x_4 + 3x_2^2)$



---

Good news: for some interesting $\Phi$'s, can eval. the associated $K(a,b)$ very fast!

Ex: "all-conj feature exp" $\Phi$ from above.

$X = \{0,1\}^n \qquad X' = \{0,1\}^{3^n}$

$\Phi$ = "all-conjunctions" feature expansion

$\Phi(x)$ has a coord for each of the $3^n$

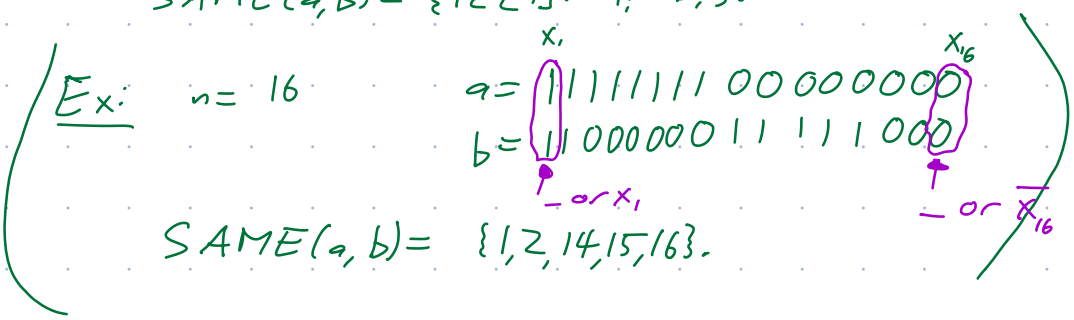
conj. over $x_1, \ldots, x_n$.

$K(a,b) = \Phi(a) \cdot \Phi(b)$

Given $a, b \in \{0,1\}^n$, define $SAME(a,b) \subseteq [n]$

$SAME(a,b) = \{i \in [n] : a_i = b_i\}$.

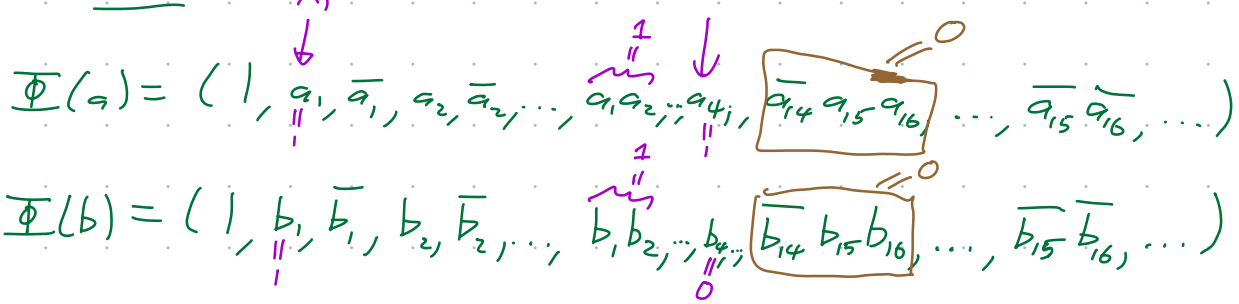
Ex: $n = 16$




$SAME(a,b) = \{1, 2, 14, 15, 16\}$.

Claim: $K(a,b) = 2^{|SAME(a,b)|}$.

Pf:




$\Phi(a) \cdot \Phi(b)$ = # posit. where $\Phi(a)$ & $\Phi(b)$ both are 1 in that pos.

Any conj containing a var outside $SAME(a,b)$:

gives $1 \cdot 0$ or $0 \cdot 1$ or $0 \cdot 0$ $\longrightarrow 0$

What abt conj. only over vars in SAME?

For each $i \in SAME$

Must either not include $x_i$ or include

$$\begin{cases} x_i & \text{if } a_i = b_i = 1 \\ \bar{x}_i & \text{if } a_i = b_i = 0. \end{cases}$$

So $2^{|SAME(a,b)|}$ conj. that are sat. both by $a$ & by $b$;

so $K(a,b) = 2^{|SAME(a,b)|}$. → $O(n)$ time

---

So... can run Perc. over space of all conj., & after $k$ mistakes, it takes $O(k \cdot n)$ time per trial.

---

End of "specific" algs

---

Next time: generic algorithms