CS480/680: Introduction to Machine Learning

Lecture 2: Perceptron

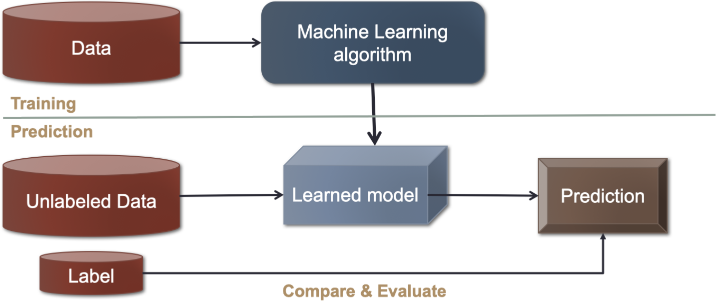
Hongyang Zhang

UNIVERSITY OF WATERLOO

Jan 11&16, 2024

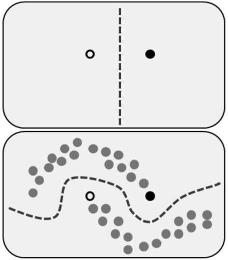
Machine Learning Phases

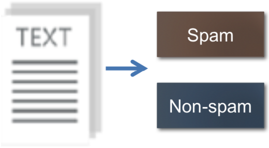
- Artificial Intelligence is a scientific field concerned with the development of algorithms that allow computers to learn without being explicitly programmed
- Machine Learning is a branch of Artificial Intelligence, which focuses on methods that learn from data and make predictions on unseen data
- Three phases: 1) training; 2) prediction (a.k.a. inference or test); 3) evaluation



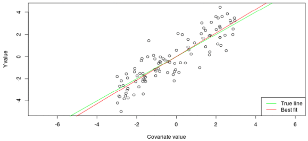
Paradigms of ML Algorithms (training phase)

- Supervised: learning with labeled data $(\mathbf{x}, y)$
  - Example: email classification, image classification
  - Example: predicting house price
- Unsupervised: discover patterns in unlabeled data $\mathbf{x}$
  - Example: cluster similar data points
  - Example: reduce the data dimension
  - Example: learn representation for downstream tasks
- Semi-supervised: using both labeled and unlabeled data

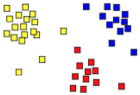




Classification



Regression



Clustering

What a Dataset Looks Like

	Training samples						Test samples	
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}'_1$	$\mathbf{x}'_2$
$\mathbb{R}^d \ni$ Feature	0	1	0	1	$\cdots$	1	1	0.9
	0	0	1	1	$\cdots$	0	1	1.1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
	1	0	1	0	$\cdots$	1	1	-0.1
Label y	$+$	$+$	$-$	$+$	$\cdots$	$-$	?	?

- each column is a data point: n in total; each has d features
- bottom y is the label vector; binary in this case
- $\mathbf{x}'_1$ and $\mathbf{x}'_2$ are test samples whose labels need to be predicted (may not appear in the training set; we will use $\mathbf{x}'$ to refer to test samples throughout the course)

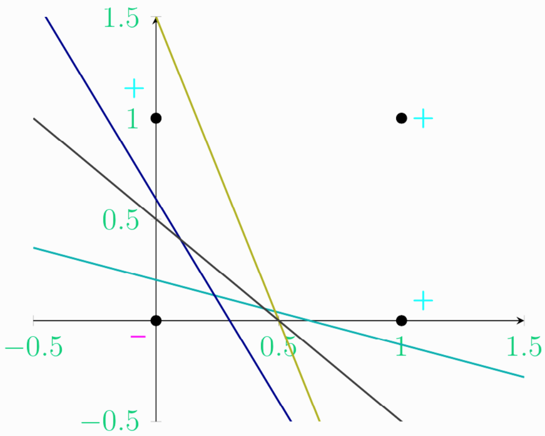
Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}'$
and	1	0	0	1	1	1	1
viagra	1	0	1	0	0	0	1
the	0	1	1	0	1	1	0
of	1	1	0	1	0	1	0
nigeria	1	0	0	0	1	0	0
y	$+$	$-$	$+$	$-$	$+$	$-$	?

- Bag-of-words representation of text; if a word appears, the feature is 1
- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathrm{y} = [\mathrm{y}_1, \ldots, \mathrm{y}_n] \in \{\pm 1\}^n$
  - each column of X is an email $\mathbf{x}_i \in \mathbb{R}^d$, each with d (binary) features
  - each entry in y is a label $\mathrm{y}_i \in \{\pm 1\}$, indicating spam or not
- Given a new email $\mathbf{x}'$ (which might not be seen before), predict spam or not

OR Dataset Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
y	−	+	+	+



Notations and Linear Separator

- Inner product: define inner product $\langle \mathbf{a}, \mathbf{b} \rangle := \sum_j a_j b_j$, where a_j and b_j are the j-th elements of vectors $\mathbf{a}$ and $\mathbf{b}$
- Linear function: $\forall \alpha, \beta \in \mathbb{R}, \forall \mathbf{x}, \mathbf{z} \in \mathbb{R}^d$,

$$f(\alpha \mathbf{x} + \beta \mathbf{z}) = \alpha \cdot f(\mathbf{x}) + \beta \cdot f(\mathbf{z})$$

  - Equivalently, $\exists \mathbf{w} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle := \sum_j x_j w_j$
- Proof: $(\Rightarrow)$ Let $\mathbf{w} := [f(\mathbf{e}_1), ..., f(\mathbf{e}_d)]^T$, where $\mathbf{e}_i$ is the i-th coordinate vector.

$$\begin{aligned} f(\mathbf{x}) &= f(x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + ... + x_d \mathbf{e}_d) \\ &= x_1 f(\mathbf{e}_1) + x_2 f(\mathbf{e}_2) + ... + x_d f(\mathbf{e}_d) = \langle \mathbf{x}, \mathbf{w} \rangle \end{aligned}$$

  $(\Leftarrow)$ We have

$$\begin{aligned} f(\alpha \mathbf{x} + \beta \mathbf{z}) &= \langle \alpha \mathbf{x} + \beta \mathbf{z}, \mathbf{w} \rangle \\ &= \alpha \langle \mathbf{x}, \mathbf{w} \rangle + \beta \langle \mathbf{z}, \mathbf{w} \rangle \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{z}) \end{aligned}$$

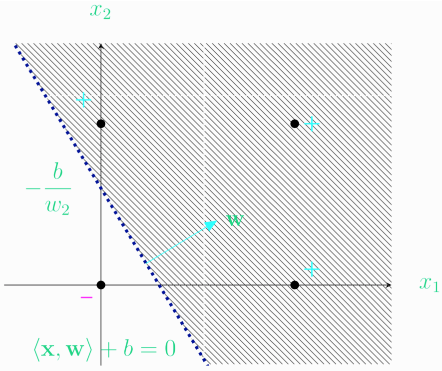
Notations and Linear Separator — Cont'

- Inner product: define inner product $\langle \mathbf{a}, \mathbf{b} \rangle := \sum_j a_j b_j$, where a_j and b_j are the j-th elements of vectors $\mathbf{a}$ and $\mathbf{b}$
- Linear function: $\forall \alpha, \beta \in \mathbb{R}, \forall \mathbf{x}, \mathbf{z} \in \mathbb{R}^d$,

$$f(\alpha \mathbf{x} + \beta \mathbf{z}) = \alpha \cdot f(\mathbf{x}) + \beta \cdot f(\mathbf{z})$$

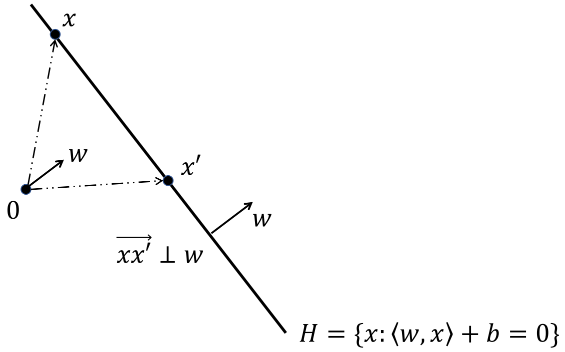
  - Equivalently, $\exists \mathbf{w} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle := \sum_j x_j w_j$
- Affine function: $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\mathrm{sign}(t) = \begin{cases} +1, & t > 0 \\ -1, & t \leq 0 \end{cases}$
  - It doesn't matter where to put the edge case $t = 0$.
- Combined together: $\hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b) = \begin{cases} +1, & \langle \mathbf{x}, \mathbf{w} \rangle + b > 0 \\ -1, & \langle \mathbf{x}, \mathbf{w} \rangle + b \leq 0 \end{cases}$

Geometrically



- $\mathbf{w}$ and b will uniquely determine the linear separator.
- **Shadow area:** $\langle \mathbf{x}, \mathbf{w} \rangle + b > 0$; **White area:** $\langle \mathbf{x}, \mathbf{w} \rangle + b < 0$
  - Therefore, a mistake happens iff $\mathrm{y}(\langle \mathbf{x}, \mathbf{w} \rangle + b) \leq 0$, where y is the true label.

Why is $\mathbf{w}$ orthogonal to decision boundary H?



- Any vector with both head and tail in H can be written as $\overrightarrow{\mathbf{x}\mathbf{x}'} = \mathbf{x}' - \mathbf{x}$ for $\mathbf{x}, \mathbf{x}' \in H$
- $\langle \mathbf{w}, \mathbf{x}' - \mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{x}' \rangle - \langle \mathbf{w}, \mathbf{x} \rangle = -b - (-b) = 0$
- b does not matter for the orthogonality. Holds for any $H = \{x : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$.
- The length of $\mathbf{w}$ does not matter in determining the decision boundary.

The Early Hype in AI...

Origins of AI hype?

Frank Rosenblatt (1928-1971)

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

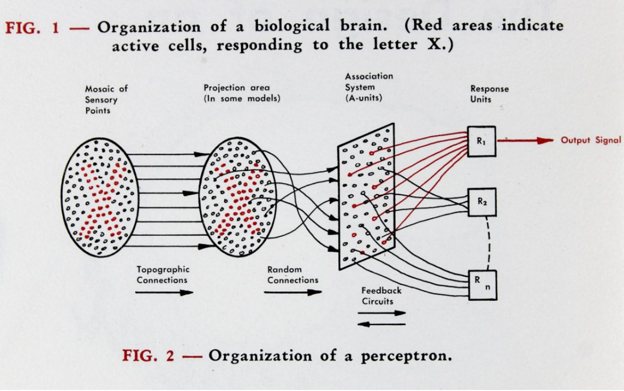
Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

...due to Perceptron to learn a linear separator



by Frank Rosenblatt
(1928 – 1971)

- Frank Rosenblatt optimistically predicted that the perceptron "may eventually be able to learn, make decisions, and translate languages".
- ...of course, which is not true.

F. Rosenblatt (1958). "The perceptron: A probabilistic model for information storage and organization in the brain". Psychological Review, vol. 65, no. 6, pp. 386–408.

Algorithm 1 Training Perceptron

Input: Dataset $= (\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n$, initialization $\mathbf{w}_0 \in \mathbb{R}^d$ and $b_0 \in \mathbb{R}$

Output: $\mathbf{w}$ and b (so a linear classifier $\mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$)

```
for t = 1, 2, ... do
    receive index I_t ∈ {1, ..., n}                    // I_t can be random
    if y_{I_t}(⟨x_{I_t}, w⟩ + b) ≤ 0 then              // a "mistake" happens
        w ← w + y_{I_t} x_{I_t}                        // update after a "mistake"
        b ← b + y_{I_t}
    end
end
```

- Typically setting $\mathbf{w}_0 = \mathbf{0}$ and $b_0 = 0$
  - $\mathsf{y}\hat{y} > 0$ (correct) $\;$ vs. $\mathsf{y}\hat{y} < 0$ (wrong), where $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle + b$ (a.k.a. $\mathsf{score}_{\mathbf{w},b}(\mathbf{x})$)
- Lazy update: "update only when a mistake happens"

Perceptron as a Feasibility Problem

$$\text{find } \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \;\; \text{such that} \;\; \forall i, \;\; \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) > 0.$$

- Perceptron solves the above feasibility problem!
  - it is in iteration: going through the data one by one
  - it converges faster if the problem is “easier”
- Key insight whenever a mistake happens on $(\mathbf{x}, \mathsf{y})$:

$$\mathsf{y}[\langle \mathbf{x}, \mathbf{w}_{k+1}\rangle + b_{k+1}] = \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k + \mathsf{y}\mathbf{x}\rangle + b_k + \mathsf{y}] = \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k\rangle + b_k] + \underbrace{\|\mathbf{x}\|_2^2 + 1}_{\text{always positive}}$$

  - Always increase the confidence $\mathsf{y}\hat{y}$ after the update

Spam Filtering Revisited

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
and	1	0	0	1	1	1
viagra	1	0	1	0	0	0
the	0	1	1	0	1	1
of	1	1	0	1	0	1
nigeria	1	0	0	0	1	0
y	$+$	$-$	$+$	$-$	$+$	$-$

- Recall the update: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$ (when a mistake happens on $(\mathbf{x}, \mathsf{y})$)
  - $\mathbf{w}_0 = [0,0,0,0,0], \quad b_0 = 0 \implies \text{score}_{\mathbf{w}_0,b_0}(\mathbf{x}_1) = 0 \implies \hat{\mathsf{y}}_1 = -$ ×
  - $\mathbf{w}_1 = [1,1,0,1,1], \quad b_1 = 1 \implies \text{score}_{\mathbf{w}_1,b_1}(\mathbf{x}_2) = 2 \implies \hat{\mathsf{y}}_2 = +$ ×
  - $\mathbf{w}_2 = [1,1,-1,0,1], b_2 = 0 \implies \text{score}_{\mathbf{w}_2,b_2}(\mathbf{x}_3) = 0 \implies \hat{\mathsf{y}}_3 = -$ ×
  - $\mathbf{w}_3 = [1,2,0,0,1], \quad b_3 = 1 \implies \text{score}_{\mathbf{w}_3,b_3}(\mathbf{x}_4) = 2 \implies \hat{\mathsf{y}}_4 = +$ ×
  - $\mathbf{w}_4 = [0,2,0,-1,1], b_4 = 0 \implies \text{score}_{\mathbf{w}_4,b_4}(\mathbf{x}_5) = 1 \implies \hat{\mathsf{y}}_5 = +$ ✓
  - $\mathbf{w}_4 = [0,2,0,-1,1], b_4 = 0 \implies \text{score}_{\mathbf{w}_4,b_4}(\mathbf{x}_6) = -1 \implies \hat{\mathsf{y}}_6 = -$ ✓

Spam Filtering Revisited — Cont'

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
and	1	0	0	1	1	1
viagra	1	0	1	0	0	0
the	0	1	1	0	1	1
of	1	1	0	1	0	1
nigeria	1	0	0	0	1	0
y	$+$	$-$	$+$	$-$	$+$	$-$

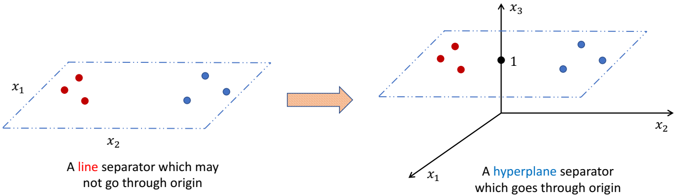
- Let's check the correctness of $\mathbf{w}_4 = [0, 2, 0, -1, 1]$ and $b_4 = 0$:
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_1) = 2 \implies \hat{y}_1 = +$ ✓
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_2) = -1 \implies \hat{y}_2 = -$ ✓
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_3) = 2 \implies \hat{y}_3 = +$ ✓
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_4) = -1 \implies \hat{y}_4 = -$ ✓
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_5) = 1 \implies \hat{y}_5 = +$ ✓
  - $\text{score}_{\mathbf{w}_4, b_4}(\mathbf{x}_6) = -1 \implies \hat{y}_6 = -$ ✓

A Trick for Hiding the Bias Term

- Previously, we talked about affine function $\langle \mathbf{x}, \mathbf{w} \rangle + b$
- Padding constant 1 to the end of each $\mathbf{x}$:

$$\langle \mathbf{x}, \mathbf{w} \rangle + b = \left\langle \underbrace{\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{x}_{\text{pad}}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}_{\text{pad}}} \right\rangle$$

(We only need to analyze a linear function)



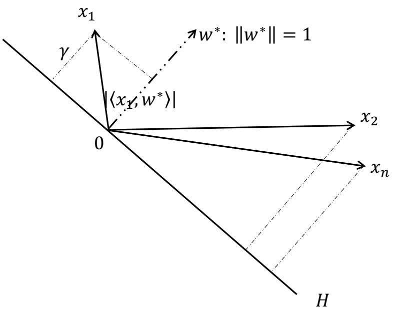
- Update rule when a mistake happens on $(\mathbf{x}, \mathrm{y})$:

$$\begin{cases} \mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}\mathbf{x} \\ b \leftarrow b + \mathrm{y} \end{cases} \Leftrightarrow \quad \mathbf{w}_{\text{pad}} \leftarrow \mathbf{w}_{\text{pad}} + \mathrm{y}\mathbf{x}_{\text{pad}}$$

Convergence Theorem (Linearly Separable Case)

Theorem: Block (1962); Novikoff (1962)

Suppose $\exists \mathbf{w}^*$ such that $\mathrm{y}_i \langle \mathbf{x}_i, \mathbf{w}^* \rangle > 0$ for $\forall i$. Assume that $\|\mathbf{x}_i\|_2 \leq C$ for $\forall i$ and we normalize the $\mathbf{w}^*$ such that $\|\mathbf{w}^*\|_2 = 1$. Let us define the margin $\gamma := \min_i |\langle \mathbf{x}_i, \mathbf{w}^* \rangle|$. Then the Perceptron algorithm converges after C^2/γ^2 mistakes.



The Proof

- Recall that the update is $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}$ (when a mistake happens on $(\mathbf{x}, \mathsf{y})$)
- Consider the effect of an update on $\langle \mathbf{w}, \mathbf{w}^* \rangle$:

  $$\langle \mathbf{w} + \mathsf{y}\mathbf{x}, \mathbf{w}^* \rangle = \langle \mathbf{w}, \mathbf{w}^* \rangle + \mathsf{y}\langle \mathbf{x}, \mathbf{w}^* \rangle \overset{\mathbf{w}^* \text{ is perfect}}{=} \langle \mathbf{w}, \mathbf{w}^* \rangle + |\langle \mathbf{x}, \mathbf{w}^* \rangle| \geq \langle \mathbf{w}, \mathbf{w}^* \rangle + \gamma$$

  This means that for each update, $\langle \mathbf{w}, \mathbf{w}^* \rangle$ grows by at least $\gamma > 0$.
- Consider the effect of an update on $\langle \mathbf{w}, \mathbf{w} \rangle$:

  $$\langle \mathbf{w} + y\mathbf{x}, \mathbf{w} + y\mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{w} \rangle + \underbrace{2\mathsf{y}\langle \mathbf{w}, \mathbf{x} \rangle}_{<0} + \underbrace{\mathsf{y}^2\langle \mathbf{x}, \mathbf{x} \rangle}_{\in[0, C^2]} \leq \langle \mathbf{w}, \mathbf{w} \rangle + C^2$$

  This means that for each update, $\langle \mathbf{w}, \mathbf{w} \rangle$ grows by at most C^2.

The Proof — Cont'

- Let $\mathbf{w}_0 = \mathbf{0}$. Now we know that after M updates:
  - $\langle \mathbf{w}, \mathbf{w}^* \rangle \geq M\gamma$;
  - $\langle \mathbf{w}, \mathbf{w} \rangle \leq MC^2$.
- We can then complete the proof:

$$
\begin{aligned}
1 \geq \cos(\mathbf{w}, \mathbf{w}^*) &= \frac{\langle \mathbf{w}, \mathbf{w}^* \rangle}{\|\mathbf{w}\| \|\mathbf{w}^*\|} \\
&\geq \frac{M\gamma}{\sqrt{MC^2} \times 1} \\
&= \sqrt{M} \frac{\gamma}{C}.
\end{aligned}
$$

  This implies $M \leq C^2/\gamma^2$.
- The larger the margin γ is, the more (linearly) separable the data will be, and hence the faster the Perceptron algorithm will converge!

Optimization Perspective on Perceptron

- Linear classifier: $\hat{y} = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$
- Minimize Perceptron loss:

$$l(\mathbf{w}, \mathbf{x}_t, \mathsf{y}_t) = -\mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle \mathbb{I}[\text{mistake on } \mathbf{x}_t] = -\min\{\mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle, 0\}$$

$$L(\mathbf{w}) = -\frac{1}{n} \sum_{t=1}^{n} \mathsf{y}_t \langle \mathbf{w}, \mathbf{x}_t \rangle \mathbb{I}[\text{mistake on } \mathbf{x}_t]$$

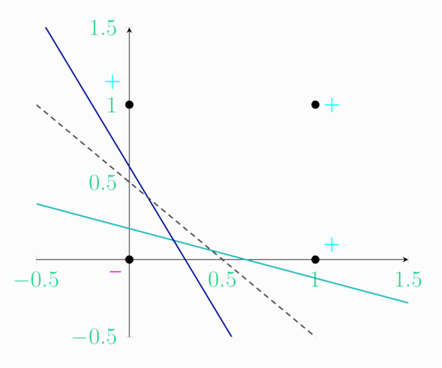
- (Stochastic) gradient descent update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} l(\mathbf{w}_t, \mathbf{x}_t, \mathsf{y}_t) = \mathbf{w}_t + \eta_t \mathsf{y}_t \mathbf{x}_t \mathbb{I}[\text{mistake on } \mathbf{x}_t]$$

- Set step size $\eta_t = 1$. If a mistake on $(\mathbf{x}_t, \mathsf{y}_t)$, then

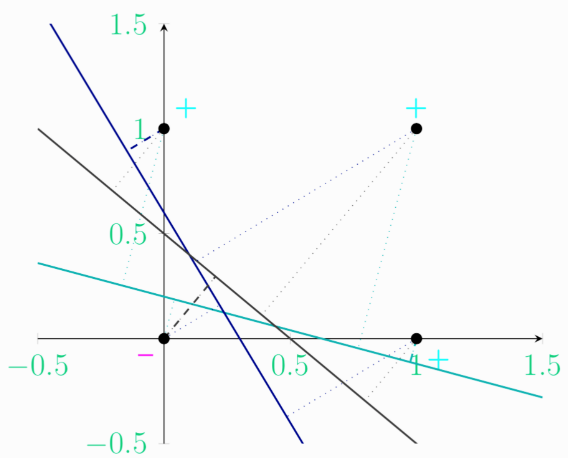
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathsf{y}_t \mathbf{x}_t \quad (\text{Perceptron update rule!})$$

But...Is Perceptron Unique?



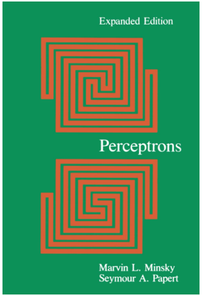
- Not unique, because the algorithm stops as long as there is no mistake.
  - Depend on initialization
  - Depend on the sampling rule of the updated data index I_t
- Then which one should we choose?

Maximize Margin: Support Vector Machines

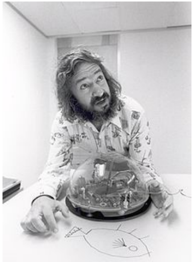


$$\max_{\mathbf{w}:\forall i, \hat{y}_i \mathsf{y}_i > 0} \quad \min_{i=1,\dots,n} \frac{\hat{y}_i \mathsf{y}_i}{\|\mathbf{w}\|}, \hat{y}_i := \langle \mathbf{x}_i, \mathbf{w} \rangle + b$$

Perceptron and the 1st AI Winter



Marvin Minsky
(1927 – 2016)

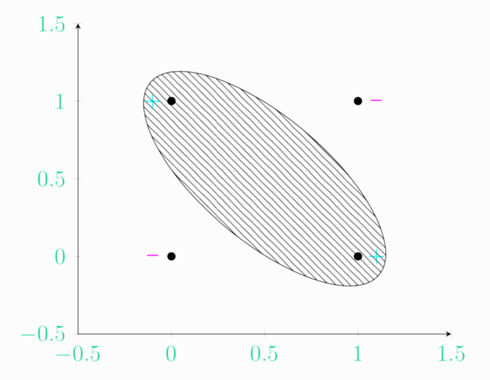


Seymour Papert
(1928 – 2016)

- When Minsky and Papert published the book Perceptrons in 1969, which outlined the limits of what perceptrons could do.

XOR Dataset

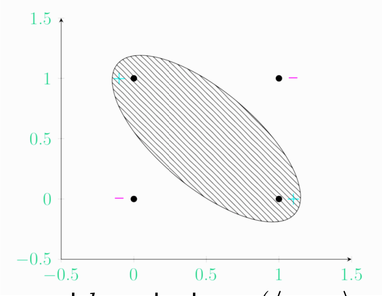
$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
0	1	0	1
0	0	1	1
$-$	$+$	$+$	$-$



- No line can separate $+$ from $-$

Proof: No Separating Hyperplane

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
	$-$	$+$	$+$	$-$



Suppose there exist $\mathbf{w}$ and b such that $\mathsf{y}(\langle \mathbf{x}, \mathbf{w} \rangle + b) > 0$

- $\mathbf{x}_1 = (0, 0), \mathsf{y}_1 = - \implies b < 0$
- $\mathbf{x}_2 = (1, 0), \mathsf{y}_2 = + \implies w_1 + b > 0$
- $\mathbf{x}_3 = (0, 1), \mathsf{y}_3 = + \implies w_2 + b > 0 \implies w_1 + w_2 + 2b > 0$
- $\mathbf{x}_4 = (1, 1), \mathsf{y}_4 = - \implies w_1 + w_2 + b < 0 \implies b > 0$

$\boxed{\text{Contradiction!}}$ What happens if we run Perceptron regardless?

Hardness Result (Non-linearly Separable Case)

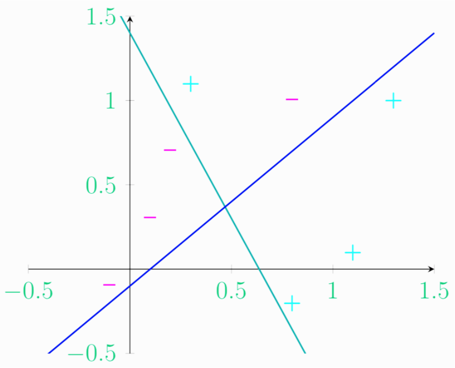
Theorem: Minsky and Papert (1969); Block and Levin (1970)

If there is no perfect separating hyperplane, then the Perceptron algorithm cycles.

- "...proof of this theorem is complicated ..." (Minsky and Papert, 1987); see Amaldi and Hauser (2005)

M. L. Minsky and S. A. Papert (1969). "Perceptron". MIT press; H. D. Block and S. A. Levin (1970). "On the boundedness of an iterative procedure for solving a system of linear inequalities". Proceedings of the American Mathematical Society, vol. 26, pp. 229–235; E. Amaldi and R. Hauser (2005). "Boundedness Theorems for the Relaxation Method". Mathematics of Operations Research, vol. 30, no. 4, pp. 939–955.

Beyond Separability



- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

$$\min_{\mathbf{w}} \ \hat{\mathbb{E}}\ell(\mathrm{y}\hat{y}) + \mathrm{reg}(\mathbf{w}), \quad \text{s.t.} \quad \hat{y} := \langle \mathbf{x}, \mathbf{w} \rangle + b$$

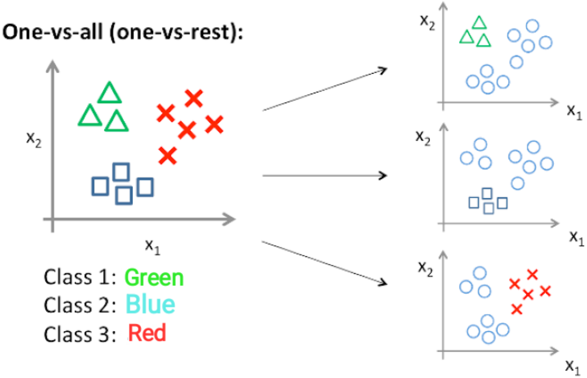
- Penalizing a mistake by the loss ℓ, but not infinitely large (allow error)

When to Stop Perceptron?

- Maximum number of iterations is reached, iter $==$ maxiter
- Maximum allowed runtime is reached
- Training error stops changing
- Validation error stops decreasing

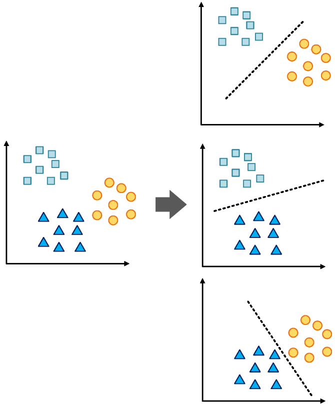
Multiclass Perceptron

- Let c be the total number of classes
- One vs. all
  - let class k be positive, and all other classes as negative
  - train Perceptron $\mathbf{w}_k$; in total c imbalanced Perceptrons
  - predict according to the highest score: $\hat{y} := \mathrm{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$



Multiclass Perceptron — Cont'

- Let c be the total number of classes
- One $\;\mathrm{vs.}\;$ one
  - let class k be positive, class l be negative, and discard all other classes
  - train Perceptron $\mathbf{w}_{k,l}$; in total $\binom{c}{2}$ balanced Perceptrons
  - predict by majority vote: $\hat{y} := \mathrm{argmax}_k \sum_{l:l \neq k} \langle \mathbf{x}, \mathbf{w}_{k,l} \rangle$




Questions
?
?
Answers
?