CS480/680: Introduction to Machine Learning

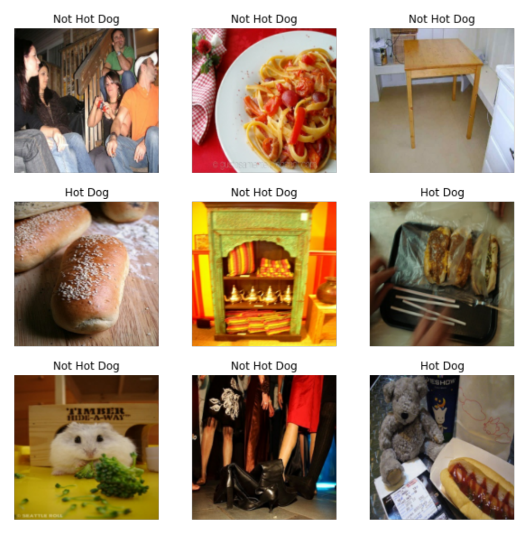
Lec 01: Perceptron

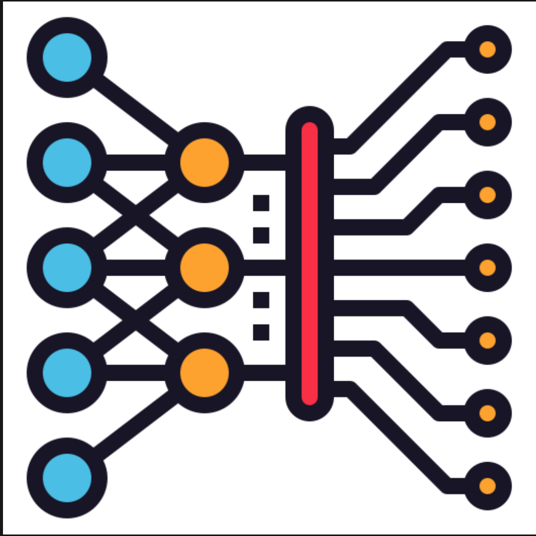
Yaoliang Yu

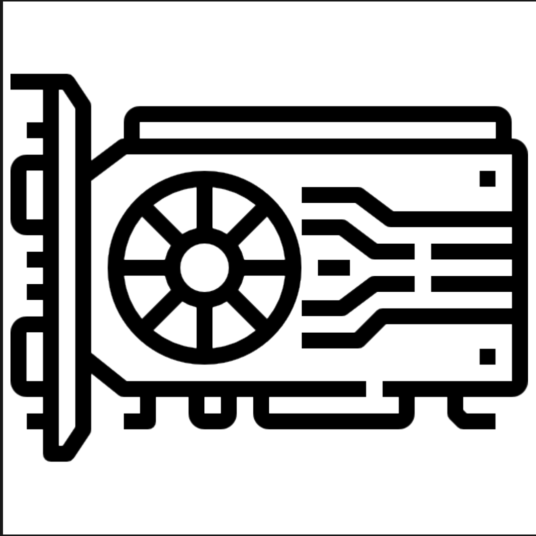
UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

May 08, 2024

dataset

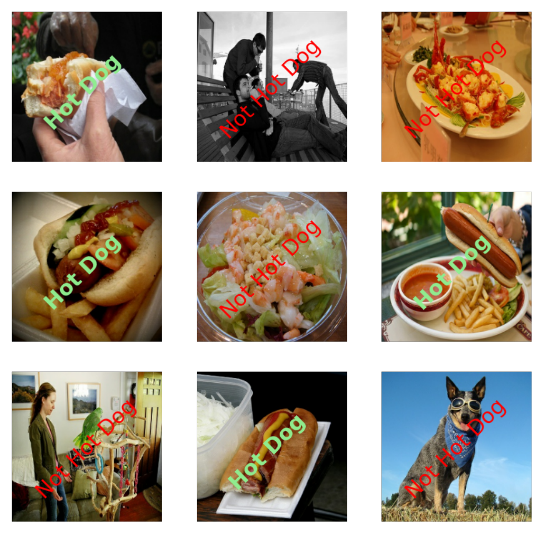

Not Hot Dog
Not Hot Dog
Not Hot Dog
Hot Dog
Not Hot Dog
Hot Dog
Not Hot Dog
Not Hot Dog
Hot Dog





Supervised Learning ☞ hotdog app

example results



What a Dataset Looks Like

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}$	$\mathbf{x}'$
$\mathbb{R}^d \ni$	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
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
- $\mathbf{x}$ and $\mathbf{x}'$ are test samples whose labels need to be predicted

What a Dataset Looks Like

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}$	$\mathbf{x}'$
$\mathbb{R}^d \ni$	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
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
- $\mathbf{x}$ and $\mathbf{x}'$ are test samples whose labels need to be predicted

What a Dataset Looks Like

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}$	$\mathbf{x}'$
$\mathbb{R}^d \ni$	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
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
- $\mathbf{x}$ and $\mathbf{x}'$ are test samples whose labels need to be predicted

What a Dataset Looks Like

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\cdots$	$\mathbf{x}_n$	$\mathbf{x}$	$\mathbf{x}'$
$\mathbb{R}^d \ni$	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
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
- $\mathbf{x}$ and $\mathbf{x}'$ are test samples whose labels need to be predicted

Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
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	+	−	+	−	+	−

- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_n] \in \{\pm 1\}^n$
  - 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 $\mathsf{y}_i \in \{\pm 1\}$, indicating spam or not
- Bag-of-words representation of text (email)

Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
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	+	−	+	−	+	−

- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_n] \in \{\pm 1\}^n$
  - 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 $\mathsf{y}_i \in \{\pm 1\}$, indicating spam or not
- Bag-of-words representation of text (email)

Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
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	+	−	+	−	+	−

- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}, \quad \mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_n] \in \{\pm 1\}^n$
  - 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 $\mathsf{y}_i \in \{\pm 1\}$, indicating spam or not
- Bag-of-words representation of text (email)

Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
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	+	−	+	−	+	−

- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}, \quad \mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_n] \in \{\pm 1\}^n$
  - 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 $\mathsf{y}_i \in \{\pm 1\}$, indicating spam or not
- Bag-of-words representation of text (email)

Spam Filtering Example

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
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	+	−	+	−	+	−

- Training set: $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}, \quad \mathsf{y} = [\mathsf{y}_1, \ldots, \mathsf{y}_n] \in \{\pm 1\}^n$
  - 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 $\mathsf{y}_i \in \{\pm 1\}$, indicating spam or not
- Bag-of-words representation of text (email)

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Batch vs. Online

- Batch learning
  - interested in performance on test set X'
  - training set (X, y) is just a means
  - statistical assumption on X and X'
- Online learning
  - data comes one by one (streaming)
  - need to predict y before knowing its true value
  - interested in making as few mistakes as possible
  - compare against some baseline

Thought Experiment

- Repeat the following game:
  - observe instance x_i
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{\mathsf{y}}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{\mathsf{y}}_i \neq \mathsf{y}_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Thought Experiment

- Repeat the following game:
  - observe instance $\mathbf{x}_i$
  - predict its label $\hat{y}_i$ (in whatever way you like)
  - reveal the true label y_i
  - suffer a mistake if $\hat{y}_i \neq y_i$
- How many mistakes in the worst-case?
- Predict first, reveal next: no peeking into the future!

Linear Threshold Function

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\text{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{\text{y}} = \text{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

Linear Threshold Function

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\text{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{\mathrm{y}} = \text{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

Linear Threshold Function

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\mathrm{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{y} = \mathrm{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

Linear Threshold Function

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\operatorname{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{y} = \operatorname{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

Linear Threshold Function

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\mathrm{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{\mathrm{y}} = \mathrm{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

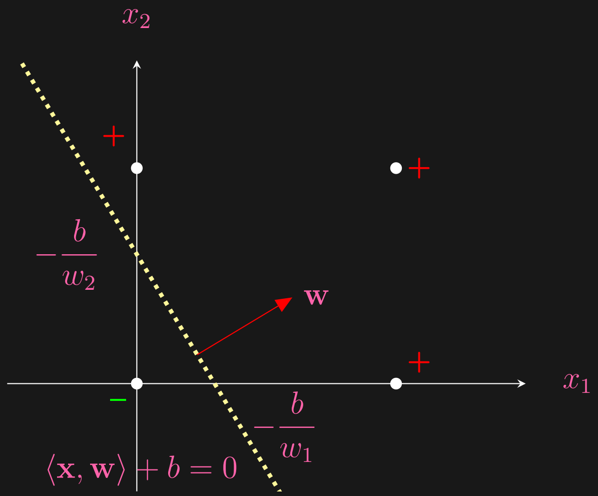
Linear Threshold Function

- Linear function: $\forall \alpha, \beta \in \mathbb{R}, \forall \mathbf{x}, \mathbf{z} \in \mathbb{R}^d$,

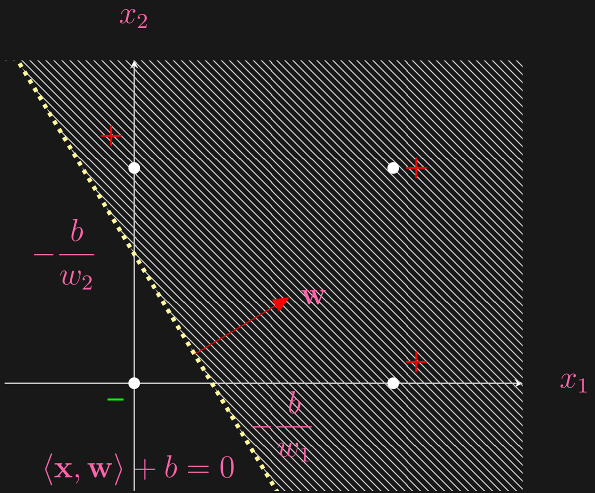
$$f(\alpha \mathbf{x} + \beta \mathbf{z}) = \alpha \cdot f(\mathbf{x}) + \beta \cdot f(\mathbf{z})$$

- 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: $\beta = 1 - \alpha$, or equivalently $\exists \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Thresholding: $\mathrm{sign}(t) = \begin{cases} 1, & t > 0 \\ -1, & t < 0 \\ ?, & t = 0 \end{cases}$
- Combined together: $\hat{\mathrm{y}} = \mathrm{sign}(\underbrace{\langle \mathbf{x}, \mathbf{w} \rangle + b}_{\hat{y}}) = \begin{cases} 1, & \hat{y} > 0 \\ -1, & \hat{y} < 0 \\ ?, & \hat{y} = 0 \end{cases}$

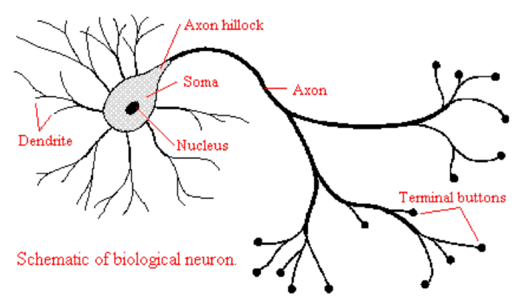
Geometrically

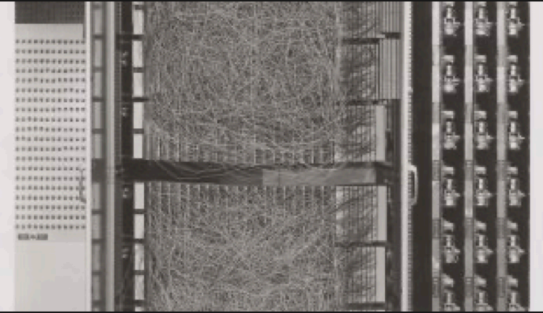


Geometrically

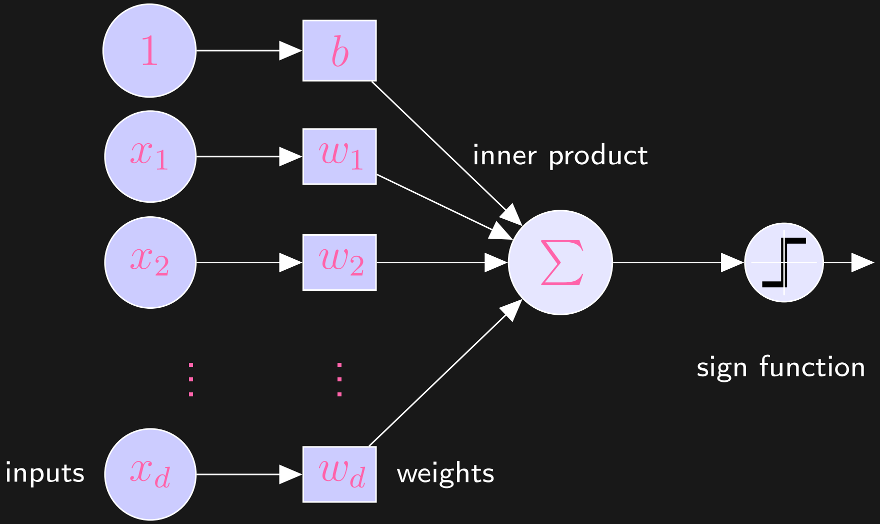


Biological Inspiration

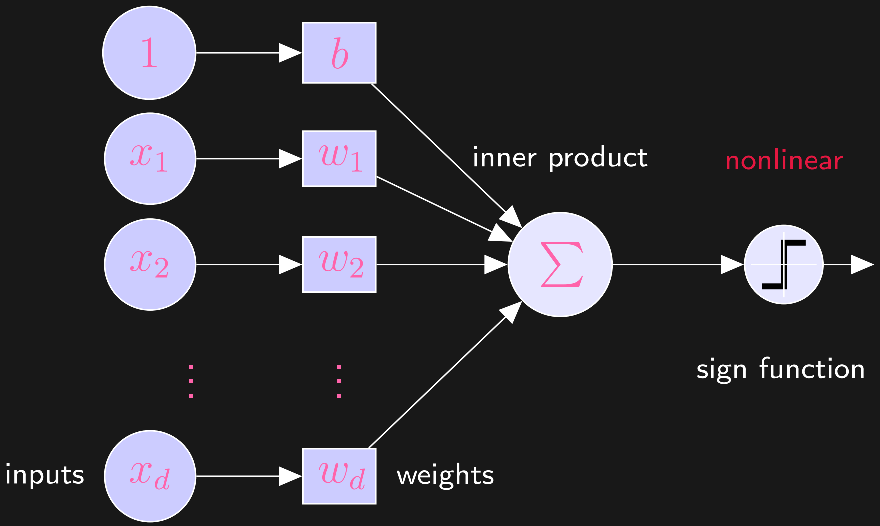




W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". *The bulletin of mathematical biophysics*, vol. 5, no. 4 (1943), pp. 115–133.



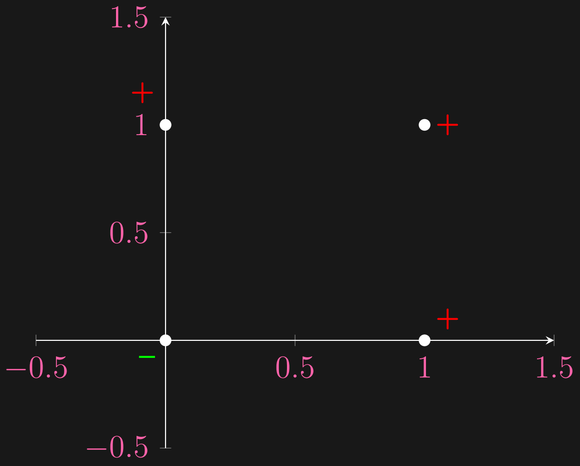
1
b
x_1
w_1
inner product
x_2
w_2
Σ
sign function
inputs
x_d
w_d
weights



1
b
x_1
w_1
x_2
w_2
x_d
w_d
inputs
weights
inner product
Σ
nonlinear
sign function

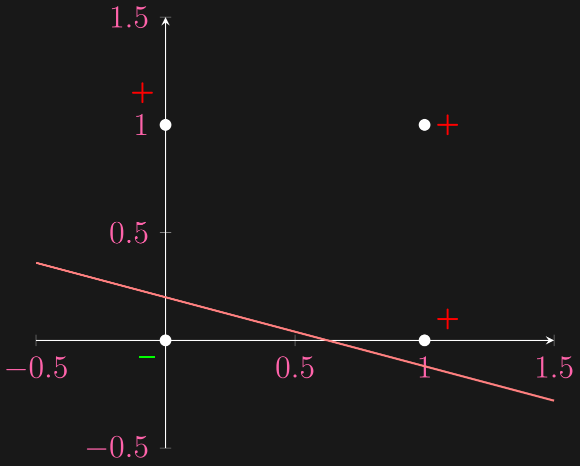
OR Dataset

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



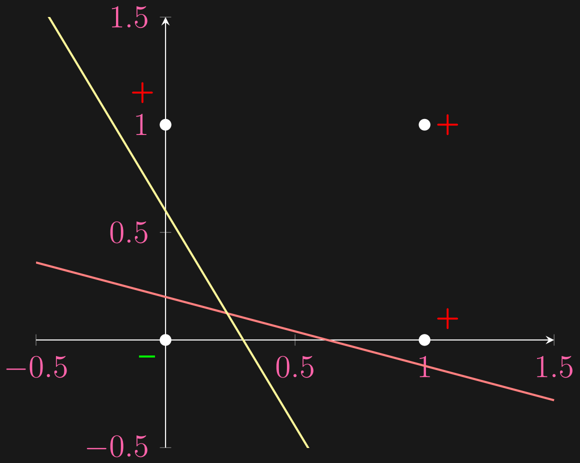
OR Dataset

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



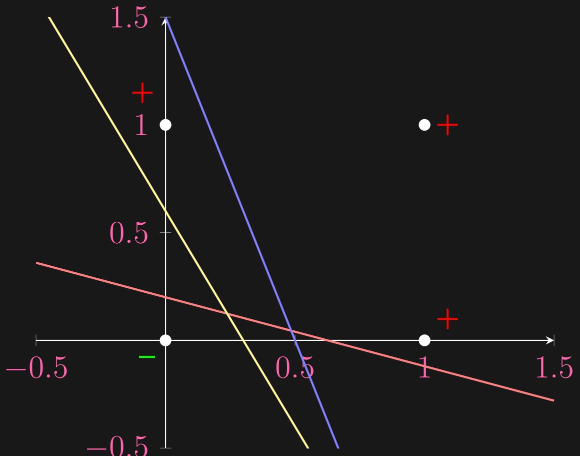
OR Dataset

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



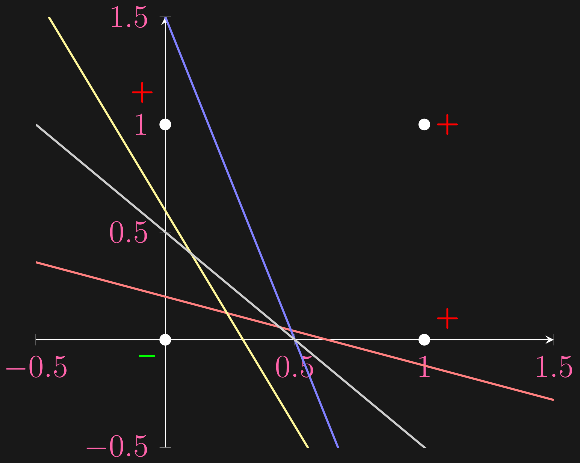
OR Dataset

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



OR Dataset

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



The Early Hype in AI...

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.

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

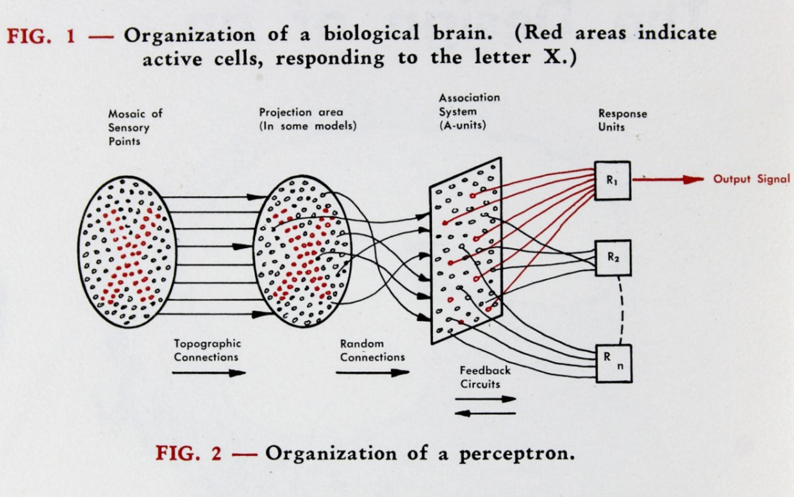
In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

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.

New York Times, 1958

...due to Perceptron



Frank Rosenblatt
(1928 – 1971)

Algorithm 1: Perceptron

Input: Dataset $\mathcal{D} = \wr(\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n\int$, initialization $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, threshold $\delta \geq 0$

Output: approximate solution $\mathbf{w}$ and 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) ≤ δ then
        w ← w + y_{I_t} x_{I_t}                     // update after a "mistake"
        b ← b + y_{I_t}
```

- Typically $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$, $b = 0$
  - $\mathsf{y}\hat{y} > 0$ vs. $\mathsf{y}\hat{y} < 0$ vs. $\mathsf{y}\hat{y} = 0$, where $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Lazy update: "if it ain't broke, don't fix it"

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

Algorithm 2: Perceptron

Input: Dataset $\mathcal{D} = \wr(\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n\int$, initialization $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, threshold $\delta \geq 0$

Output: approximate solution $\mathbf{w}$ and 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) ≤ δ then
        w ← w + y_{I_t} x_{I_t}                      // update after a "mistake"
        b ← b + y_{I_t}
```

- Typically $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$, $b = 0$
  - $\mathsf{y}\hat{y} > 0$ vs. $\mathsf{y}\hat{y} < 0$ vs. $\mathsf{y}\hat{y} = 0$, where $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Lazy update: "if it ain't broke, don't fix it"

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

Algorithm 3: Perceptron

Input: Dataset $\mathcal{D} = \wr (\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n \int$, initialization $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, threshold $\delta \geq 0$

Output: approximate solution $\mathbf{w}$ and 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) ≤ δ then
        w ← w + y_{I_t} x_{I_t}                        // update after a "mistake"
        b ← b + y_{I_t}
```

- Typically $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$, $b = 0$
  - $\mathsf{y}\hat{y} > 0$ vs. $\mathsf{y}\hat{y} < 0$ vs. $\mathsf{y}\hat{y} = 0$, where $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Lazy update: "if it ain't broke, don't fix it"

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

Algorithm 4: Perceptron

Input: Dataset $\mathcal{D} = \lbag(\mathbf{x}_i, \mathsf{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n\rbag$, initialization $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, threshold $\delta \geq 0$

Output: approximate solution $\mathbf{w}$ and 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) ≤ δ then
        w ← w + y_{I_t} x_{I_t}                        // update after a "mistake"
        b ← b + y_{I_t}
```

- Typically $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$, $b = 0$
  - $\mathsf{y}\hat{y} > 0$ vs. $\mathsf{y}\hat{y} < 0$ vs. $\mathsf{y}\hat{y} = 0$, where $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle + b$
- Lazy update: "if it ain't broke, don't fix it"

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

Perceptron as an Optimization 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 optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

$$\begin{aligned} \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_{k+1} \rangle + b_{k+1}] &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k + \mathsf{y}\mathbf{x} \rangle + b_k + \mathsf{y}] \\ &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k] + \|\mathbf{x}\|_2^2 + 1 \end{aligned}$$

Perceptron as an Optimization Problem

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

- Perceptron solves the above optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

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

Perceptron as an Optimization 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 optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

$$\begin{aligned} \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_{k+1} \rangle + b_{k+1}] &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k + \mathsf{y}\mathbf{x} \rangle + b_k + \mathsf{y}] \\ &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k] + \|\mathbf{x}\|_2^2 + 1 \end{aligned}$$

Perceptron as an Optimization Problem

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

- Perceptron solves the above optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

$$\begin{aligned} \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_{k+1} \rangle + b_{k+1}] &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k + \mathsf{y}\mathbf{x} \rangle + b_k + \mathsf{y}] \\ &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k] + \|\mathbf{x}\|_2^2 + 1 \end{aligned}$$

Perceptron as an Optimization Problem

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

- Perceptron solves the above optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

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

Perceptron as an Optimization Problem

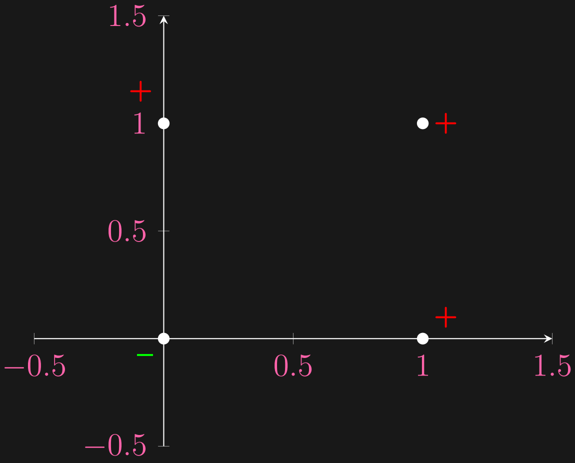
$$\text{find } \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \;\; \text{such that} \;\; \forall i, \;\; \mathsf{y}_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) > 0$$

- Perceptron solves the above optimization problem!
  - it is iterative: going through the data one by one
  - it converges faster if the problem is "easier"
  - it behaves benignly even if no solution exists
- Key insight whenever a mistake happens:

$$\begin{aligned} \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_{k+1} \rangle + b_{k+1}] &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k + \mathsf{y}\mathbf{x} \rangle + b_k + \mathsf{y}] \\ &= \mathsf{y}[\langle \mathbf{x}, \mathbf{w}_k \rangle + b_k] + \|\mathbf{x}\|_2^2 + 1 \end{aligned}$$

Does it work?

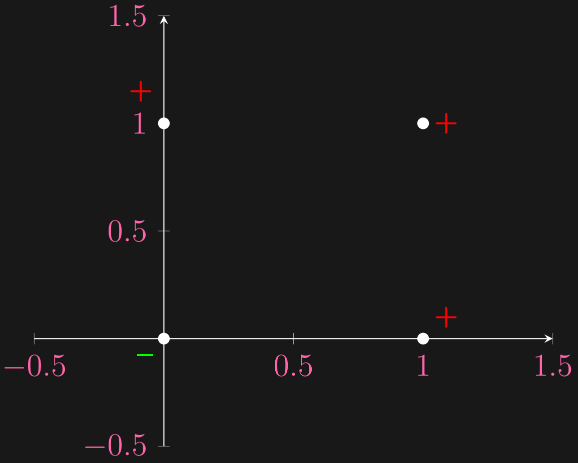
☞ code



$$\hat{y} = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\text{sign}(0)$ is undefined (e.g., always counted as a mistake).

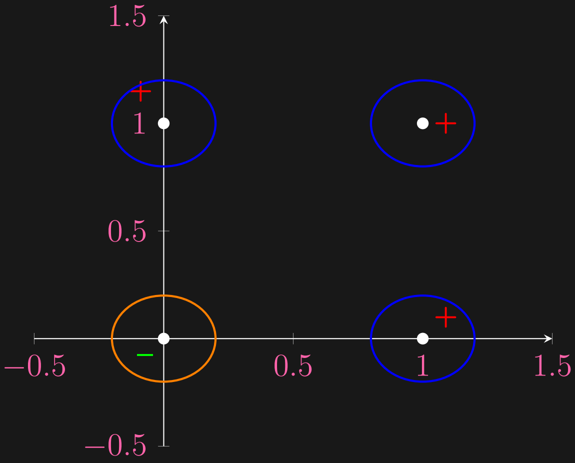
Does it work? ☞ code



$$\mathbf{w} = [0, 0], \ b = 0, \quad \hat{y} = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\text{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work? ☞ code

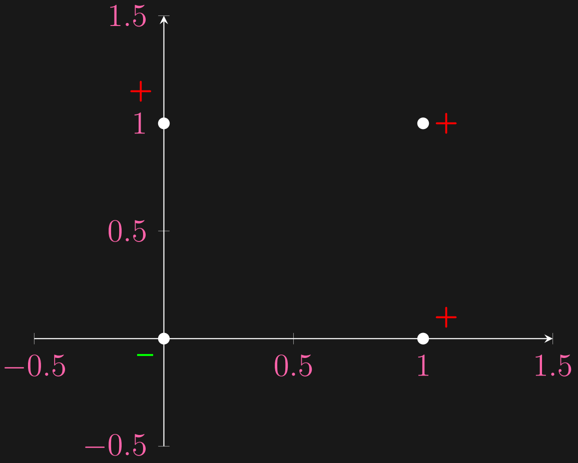


$$\mathbf{w} = [0, 0],\ b = 0,\quad \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

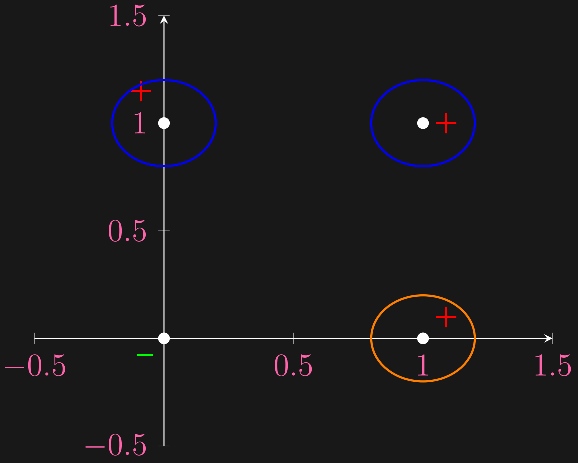
☞ code



$$\mathbf{w} = [0, 0],\ \ b = -1,\ \ \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work? ☞ code

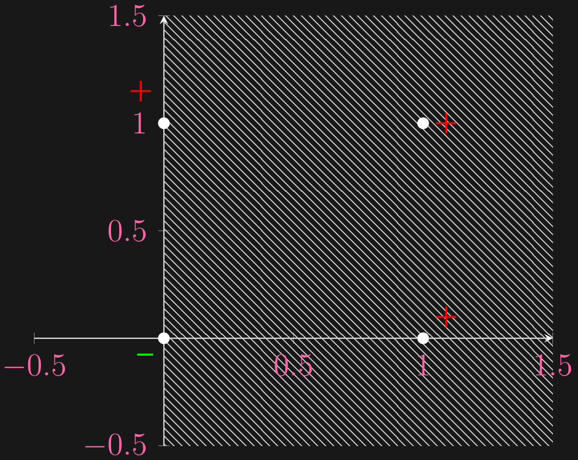


$$\mathbf{w} = [0, 0], \; b = -1, \quad \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code

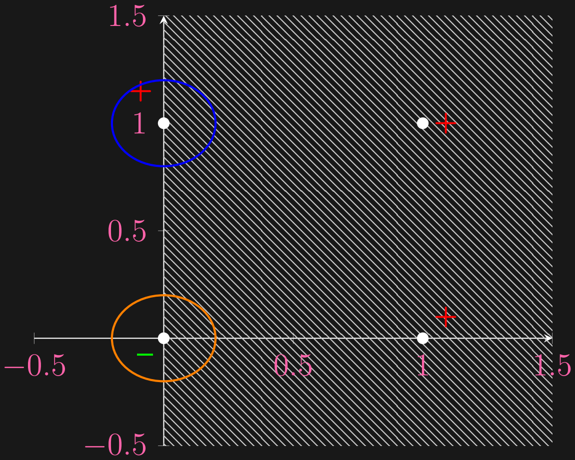


$$\mathbf{w} = [1, 0], \; b = 0, \quad \hat{y} = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\text{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

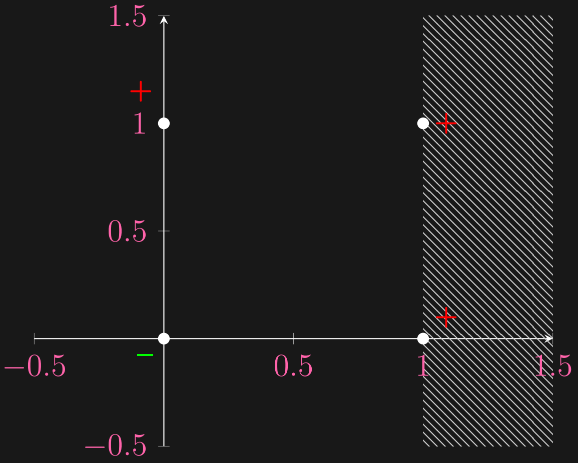
☞ code



$$\mathbf{w} = [1, 0],\ b = 0,\ \ \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

☞ code

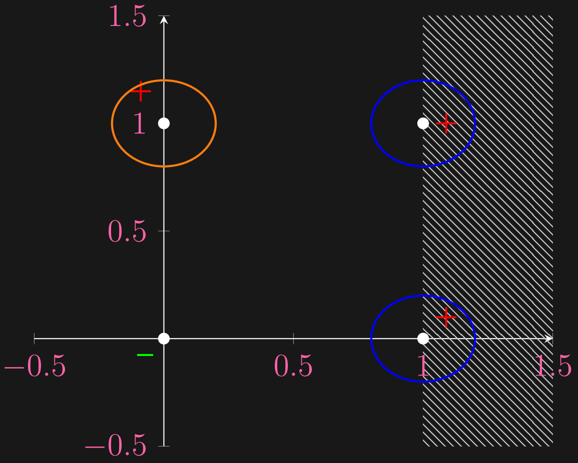


$$\mathbf{w} = [1, 0], \; b = -1, \quad \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code

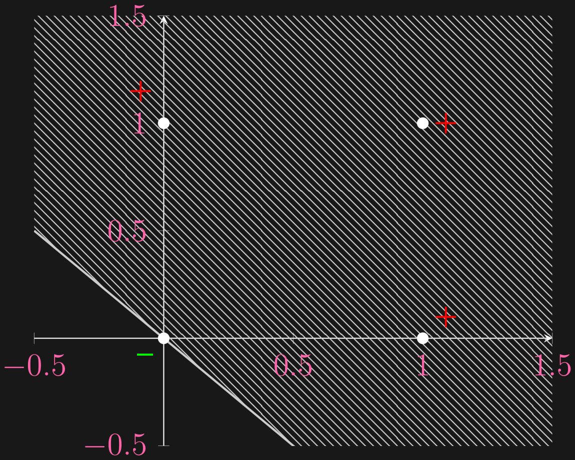


$$\mathbf{w} = [1, 0], \; b = -1, \quad \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code

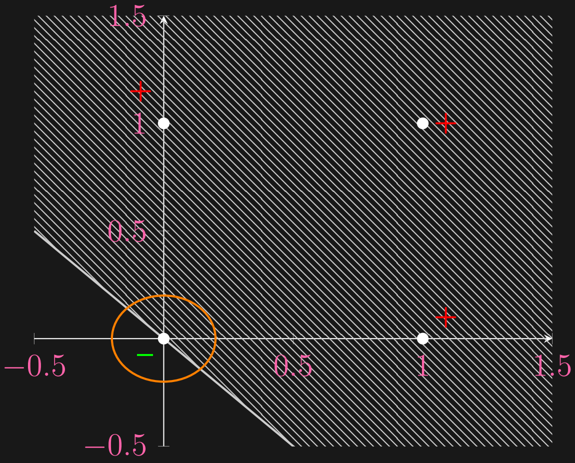


$$\mathbf{w} = [1, 1],\ b = 0,\quad \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code

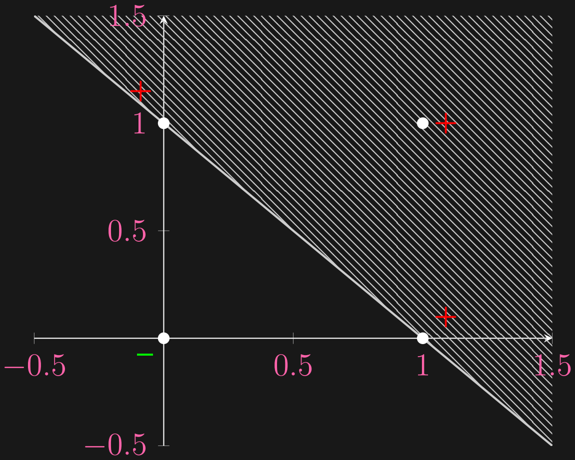


$$\mathbf{w} = [1,1],\; b = 0,\quad \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

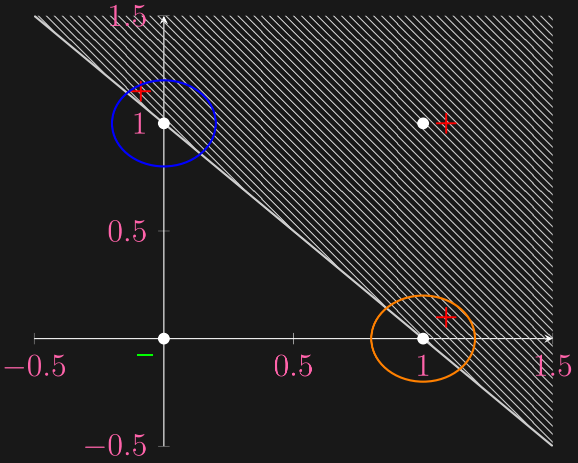
☞ code



$$\mathbf{w} = [1, 1],\ b = -1,\quad \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work? ☞ code

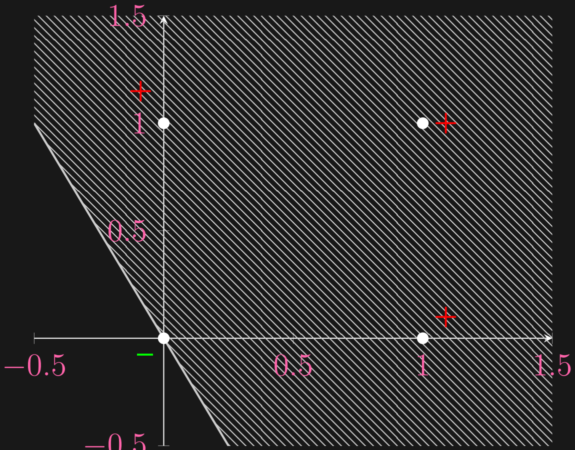


$$\mathbf{w} = [1, 1],\ b = -1,\ \ \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

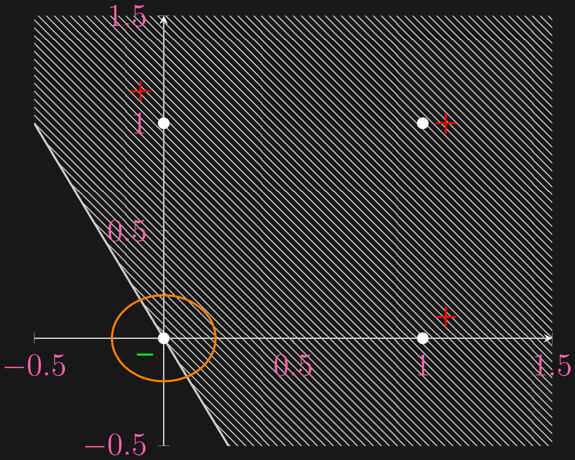
☞ code



$$\mathbf{w} = [2, 1],\ b = 0,\ \ \hat{y} = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\text{sign}(0)$ is undefined (e.g., always counted as a mistake).

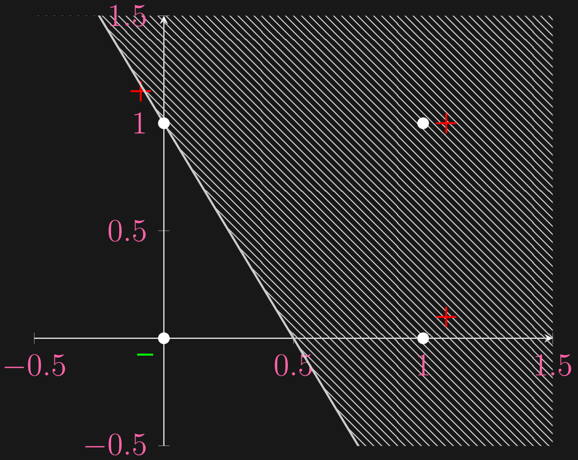
Does it work? ☞ code



$$\mathbf{w} = [2, 1], \; b = 0, \quad \hat{y} = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\text{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work? ☞ code

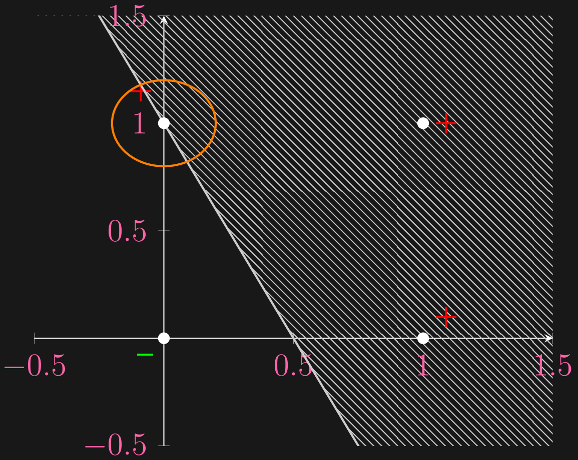


$$\mathbf{w} = [2, 1],\ b = -1,\quad \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

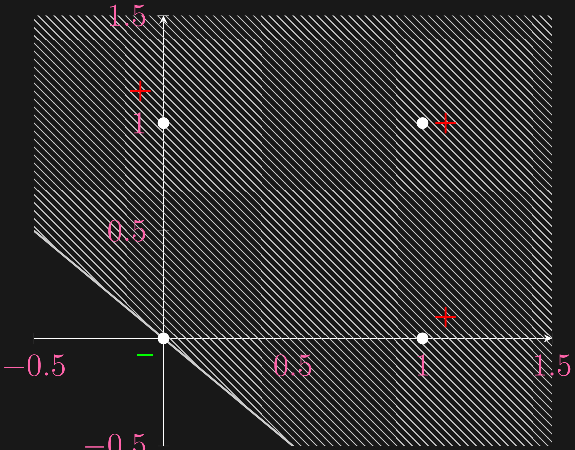
☞ code



$$\mathbf{w} = [2, 1],\; b = -1,\; \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work? ☞ code

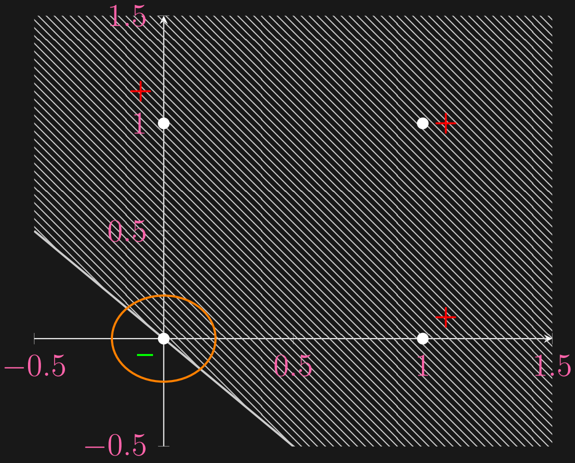


$$\mathbf{w} = [2, 2],\ \ b = 0,\ \ \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code

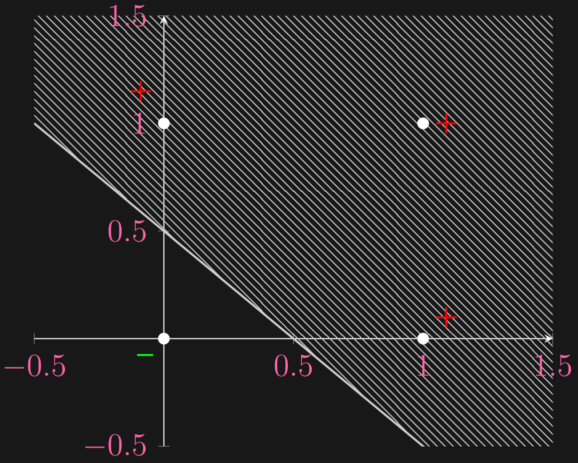


$$\mathbf{w} = [2, 2], \;\; b = 0, \quad \hat{y} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

Does it work?

☞ code



$$\mathbf{w} = [2, 2],\ b = -1,\ \ \hat{\mathrm{y}} = \mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b),$$

where $\mathrm{sign}(0)$ is undefined (e.g., always counted as a mistake).

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_5 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_6 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_5 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_6 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0,0,0,0,0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1,1,0,1,1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1,1,-1,0,1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$ ✗
  - $\mathbf{w}_3 = [1,2,0,0,1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$
  - $\mathbf{w}_4 = [0,2,0,-1,1], b_4 = 0 \implies \hat{\mathsf{y}}_5 = +$
  - $\mathbf{w}_4 = [0,2,0,-1,1], b_4 = 0 \implies \hat{\mathsf{y}}_6 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$ ✗
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}\mathbf{x}, \quad b \leftarrow b + \mathrm{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathrm{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathrm{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathrm{y}}_3 = -$ ✗
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathrm{y}}_4 = +$ ✗
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_5 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_6 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$ ✗
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$ ✗
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_5 = +$
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_6 = -$

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: $\mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}\mathbf{x}, \quad b \leftarrow b + \mathrm{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathrm{y}}_1 = -$ ✗
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathrm{y}}_2 = +$ ✗
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathrm{y}}_3 = -$ ✗
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathrm{y}}_4 = +$ ✗
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_5 = +$ ✅
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_6 = -$

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	+	−	+	−	+	−

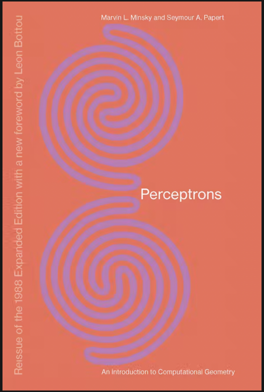
- Recall the update: $\mathbf{w} \leftarrow \mathbf{w} + \mathsf{y}\mathbf{x}, \quad b \leftarrow b + \mathsf{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathsf{y}}_1 = -$ ❌
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathsf{y}}_2 = +$ ❌
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathsf{y}}_3 = -$ ❌
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathsf{y}}_4 = +$ ❌
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_5 = +$ ✅
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathsf{y}}_6 = -$

Spam Filtering Revisited

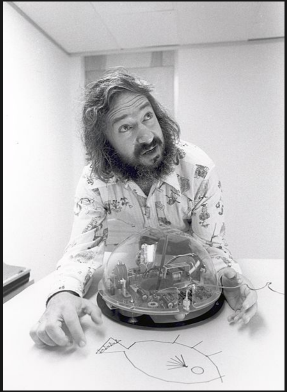
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
and	1	0	0	1	1	1
viagra	1	0	1	0	0	0
the	0	1	1	0	1	1
of	1	1	0	1	0	1
nigeria	1	0	0	0	1	0
y	+	−	+	−	+	−

- Recall the update: $\mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}\mathbf{x}, \quad b \leftarrow b + \mathrm{y}$
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0], \quad b_0 = 0 \implies \hat{\mathrm{y}}_1 = -$ ❌
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1], \quad b_1 = 1 \implies \hat{\mathrm{y}}_2 = +$ ❌
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \hat{\mathrm{y}}_3 = -$ ❌
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \hat{\mathrm{y}}_4 = +$ ❌
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_5 = +$ ✅
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \hat{\mathrm{y}}_6 = -$ ✅

Perceptron and the 1st AI Winter



Marvin Minsky
(1927 – 2016)

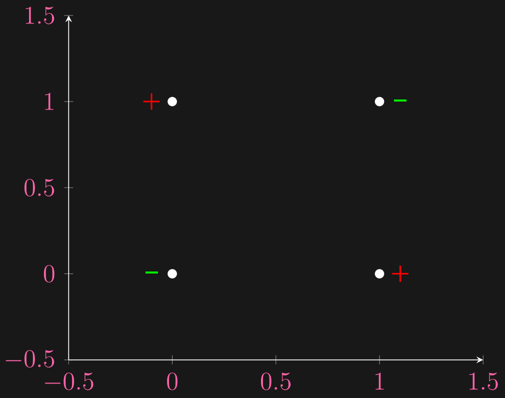


Seymour Papert
(1928 – 2016)

M. L. Minsky and S. A. Papert. "Perceptron". MIT press, 1969.

XOR Dataset

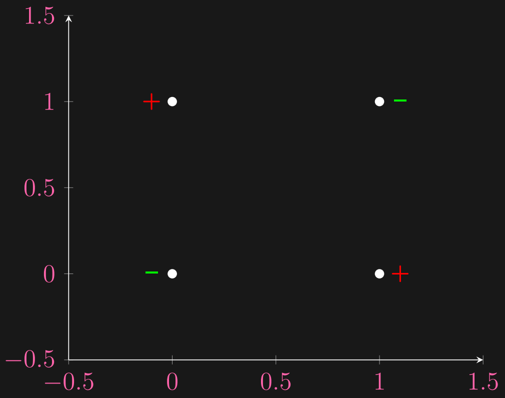
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
$\mathbf{y}$	−	+	+	−



- Prove that no line can separate + from −
- What happens if we run Perceptron regardless?

XOR Dataset

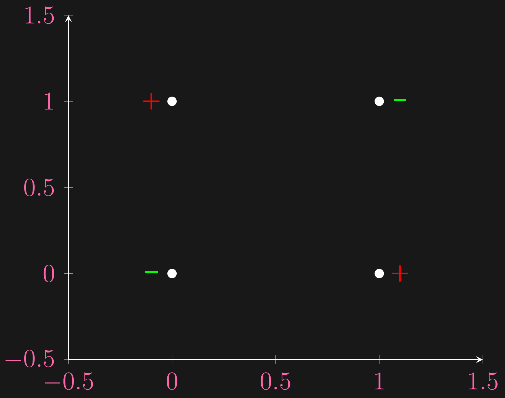
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
$\mathbf{y}$	$-$	$+$	$+$	$-$



- Prove that no line can separate $+$ from $-$
- What happens if we run Perceptron regardless?

XOR Dataset

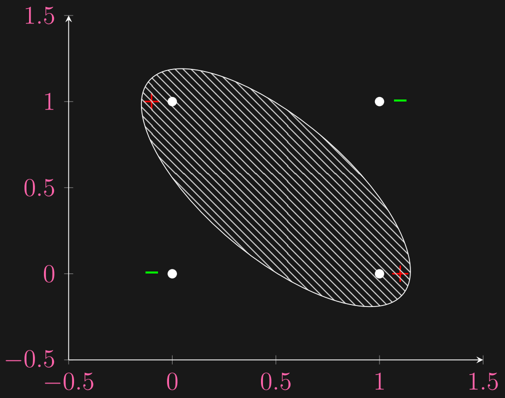
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
	0	1	0	1
	0	0	1	1
$\mathbf{y}$	$-$	$+$	$+$	$-$



- Prove that no line can separate $+$ from $-$
- What happens if we run Perceptron regardless?

XOR Dataset

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



- Prove that no line can separate + from −
- What happens if we run Perceptron regardless?

Notation Simplification

- Padding constant 1 to the (start) end of each $\mathbf{x}$:

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

- Pre-multiply $\mathbf{x}$ with its label y:

$$\mathsf{y}[\langle \mathbf{x}, \mathbf{w} \rangle + b] = \left\langle \underbrace{\mathsf{y}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{a}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}} \right\rangle$$

- The problem "simplifies" to:

$$\boxed{\text{find } \mathbf{w} \in \mathbb{R}^p \text{ such that } \mathbf{A}^\top \mathbf{w} > \mathbf{0}, \text{ where } \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}}$$

Notation Simplification

- Padding constant 1 to the (start) end of each $\mathbf{x}$:

$$\langle \mathbf{x}, \mathbf{w} \rangle + b = \Big\langle \underbrace{\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{x}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}} \Big\rangle$$

- Pre-multiply $\mathbf{x}$ with its label y:

$$\mathsf{y}[\langle \mathbf{x}, \mathbf{w} \rangle + b] = \Big\langle \underbrace{\mathsf{y}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{a}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}} \Big\rangle$$

- The problem "simplifies" to:

$$\boxed{\text{find } \mathbf{w} \in \mathbb{R}^p \quad \text{such that} \quad \mathbf{A}^\top \mathbf{w} > \mathbf{0}, \quad \text{where } \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}}$$

Notation Simplification

- Padding constant 1 to the (start) end of each $\mathbf{x}$:

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

- Pre-multiply $\mathbf{x}$ with its label y:

$$\mathsf{y}[\langle \mathbf{x}, \mathbf{w} \rangle + b] = \left\langle \underbrace{\mathsf{y}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{a}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}} \right\rangle$$

- The problem "simplifies" to:

$$\text{find } \mathbf{w} \in \mathbb{R}^p \text{ such that } \mathbf{A}^\top \mathbf{w} > \mathbf{0}, \text{ where } \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}$$

Notation Simplification

- Padding constant 1 to the (start) end of each $\mathbf{x}$:

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

- Pre-multiply $\mathbf{x}$ with its label y:

$$\mathsf{y}[\langle \mathbf{x}, \mathbf{w} \rangle + b] = \left\langle \underbrace{\mathsf{y}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}}_{\mathbf{a}}, \underbrace{\begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}}_{\mathbf{w}} \right\rangle$$

- The problem "simplifies" to:

$$\boxed{\text{find } \mathbf{w} \in \mathbb{R}^p \text{ such that } \mathbf{A}^\top \mathbf{w} > \mathbf{0}, \text{ where } \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}}$$

Interpreting Perceptron

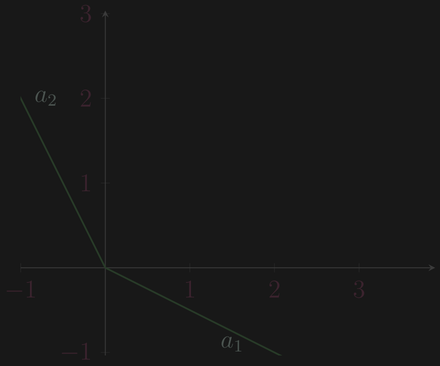
Theorem:

$$\operatorname{int}\operatorname{cone}^* A \neq \emptyset \iff \operatorname{int}\operatorname{cone}^* A \cap \operatorname{cone} A \neq \emptyset.$$

$$\operatorname{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$

$$\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$

$$\operatorname{int}\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$



Interpreting Perceptron

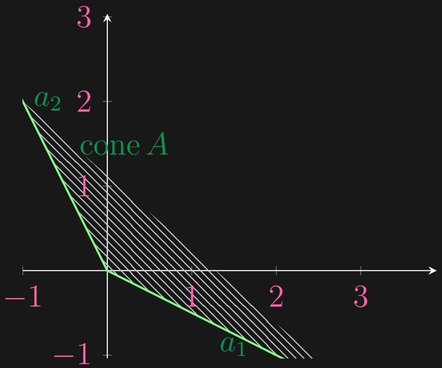
Theorem:

$$\operatorname{int}\operatorname{cone}^* A \neq \emptyset \iff \operatorname{int}\operatorname{cone}^* A \cap \operatorname{cone} A \neq \emptyset.$$

$$\operatorname{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$

$$\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$

$$\operatorname{int}\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$

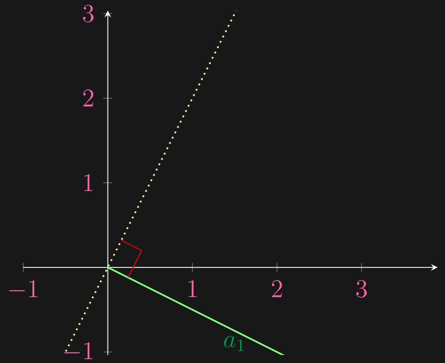


Interpreting Perceptron

Theorem:

$\operatorname{int} \operatorname{cone}^* A \neq \emptyset \iff \operatorname{int} \operatorname{cone}^* A \cap \operatorname{cone} A \neq \emptyset.$

$$\operatorname{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$
$$\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$
$$\operatorname{int} \operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$

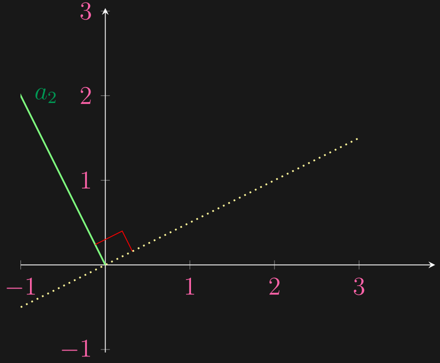


Interpreting Perceptron

Theorem:

$\operatorname{int}\operatorname{cone}^* A \neq \emptyset \iff \operatorname{int}\operatorname{cone}^* A \cap \operatorname{cone} A \neq \emptyset$.

$$\operatorname{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$
$$\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$
$$\operatorname{int}\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$



Interpreting Perceptron

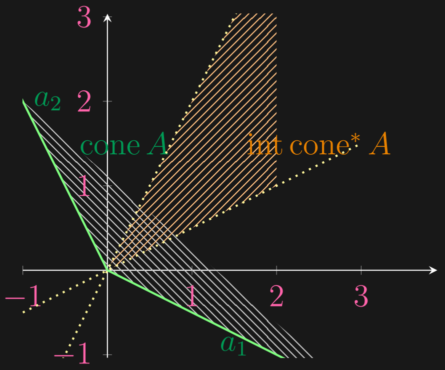
Theorem:

$\text{int cone}^* A \neq \emptyset \iff \text{int cone}^* A \cap \text{cone} A \neq \emptyset.$

$$\text{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$

$$\text{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$

$$\text{int cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$



Interpreting Perceptron

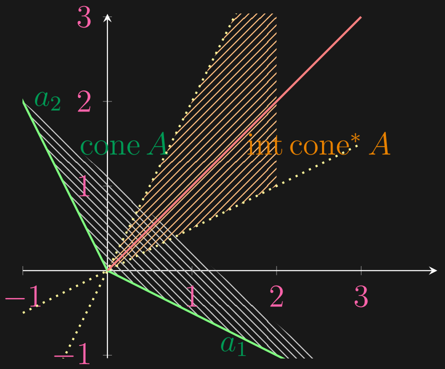
Theorem:

$$\operatorname{int}\operatorname{cone}^* A \neq \emptyset \iff \operatorname{int}\operatorname{cone}^* A \cap \operatorname{cone} A \neq \emptyset.$$

$$\operatorname{cone} A := \{A\boldsymbol{\lambda} : \boldsymbol{\lambda} \geq \mathbf{0}\}$$

$$\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} \geq \mathbf{0}\}$$

$$\operatorname{int}\operatorname{cone}^* A := \{\mathbf{w} : A^\top \mathbf{w} > \mathbf{0}\}$$



Convergence Theorem

Theorem: (Block, 1962; Novikoff, 1962)

Provided that there exists a (strictly) separating hyperplane, the Perceptron iterate converges to some $\mathbf{w}$. If each training data is selected infinitely often, then for all i, $\langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w} \rangle > \delta$.

Corollary:

Let $\delta = 0$ and initial $\mathbf{w} = \mathbf{0}$. Then, Perceptron converges after at most $(R/\gamma)^2$ mistakes, where

$$R := \max_i \|\mathbf{x}_i\|_2, \quad \gamma := \max_{\|\mathbf{w}\|_2 \le 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w} \rangle$$

H. D. Block. "The perceptron: A model for brain functioning". *Reviews of Modern Physics*, vol. 34, no. 1 (1962), pp. 123–135, A. Novikoff. "On Convergence proofs for perceptrons". In: *Symposium on Mathematical Theory of Automata*. 1962, pp. 615–622.

Convergence Theorem

Theorem: (Block, 1962; Novikoff, 1962)

Provided that there exists a (strictly) separating hyperplane, the Perceptron iterate converges to some $\mathbf{w}$. If each training data is selected infinitely often, then for all i, $\langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w} \rangle > \delta$.

Corollary:

Let $\delta = 0$ and initial $\mathbf{w} = \mathbf{0}$. Then, Perceptron converges after at most $(R/\gamma)^2$ mistakes, where

$$R := \max_i \|\mathbf{x}_i\|_2, \quad \gamma := \max_{\|\mathbf{w}\|_2 \leq 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w} \rangle$$

H. D. Block. "The perceptron: A model for brain functioning". *Reviews of Modern Physics*, vol. 34, no. 1 (1962), pp. 123–135, A. Novikoff. "On Convergence proofs for perceptrons". In: *Symposium on Mathematical Theory of Automata*. 1962, pp. 615–622.

The Proof

- By assumption:

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle > 0 \iff \text{for some and hence for all } s > 0$$
$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle \geq s$$

- Update after a mistake:

$$\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k + \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k, \mathbf{w}^\star \rangle + \overbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle}^{\geq s}$$
$$\|\mathbf{w}_{k+1}\|_2 = \|\mathbf{w}_k + \mathsf{y}\mathbf{x}\|_2 = \sqrt{\|\mathbf{w}_k\|_2^2 + \underbrace{\|\mathbf{x}\|_2^2}_{\leq R^2} + 2\underbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}_k \rangle}_{\leq \delta}}$$

- The angle approaches 0 ?

$$\cos \angle(\mathbf{w}_{k+1}, \mathbf{w}^\star) := \frac{\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle}{\|\mathbf{w}_{k+1}\|_2 \cdot \|\mathbf{w}^\star\|_2} = \frac{\Omega(k)}{O(\sqrt{k})} \xrightarrow{?} 1$$

The Proof

- By assumption:

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle > 0 \iff \text{for some and hence for all } s > 0 \quad \exists \mathbf{w}^\star \text{ s.t. } \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle \geq s$$

- Update after a mistake:

$$\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k + y\mathbf{x}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k, \mathbf{w}^\star \rangle + \overbrace{\langle y\mathbf{x}, \mathbf{w}^\star \rangle}^{\geq s}$$

$$\|\mathbf{w}_{k+1}\|_2 = \|\mathbf{w}_k + y\mathbf{x}\|_2 = \sqrt{\|\mathbf{w}_k\|_2^2 + \underbrace{\|\mathbf{x}\|_2^2}_{\leq R^2} + 2\underbrace{\langle y\mathbf{x}, \mathbf{w}_k \rangle}_{\leq \delta}}$$

- The angle approaches 0 ?

$$\cos \angle(\mathbf{w}_{k+1}, \mathbf{w}^\star) := \frac{\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle}{\|\mathbf{w}_{k+1}\|_2 \cdot \|\mathbf{w}^\star\|_2} = \frac{\Omega(k)}{O(\sqrt{k})} \xrightarrow{?} 1$$

The Proof

- By assumption:

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle > 0 \iff \text{for some and hence for all } s > 0$$

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle \geq s$$

- Update after a mistake:

$$\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k + \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k, \mathbf{w}^\star \rangle + \overbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle}^{\geq s}$$

$$\|\mathbf{w}_{k+1}\|_2 = \|\mathbf{w}_k + \mathsf{y}\mathbf{x}\|_2 = \sqrt{\|\mathbf{w}_k\|_2^2 + \underbrace{\|\mathbf{x}\|_2^2}_{\leq R^2} + 2\underbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}_k \rangle}_{\leq \delta}}$$

- The angle approaches 0 ?

$$\cos \angle(\mathbf{w}_{k+1}, \mathbf{w}^\star) := \frac{\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle}{\|\mathbf{w}_{k+1}\|_2 \cdot \|\mathbf{w}^\star\|_2} = \frac{\Omega(k)}{O(\sqrt{k})} \overset{?}{\to} 1$$

The Proof

- By assumption:

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle > 0 \iff \text{for some and hence for all } s > 0$$

$$\exists \mathbf{w}^\star \text{ s.t. } \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle \geq s$$

- Update after a mistake:

$$\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k + \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle = \langle \mathbf{w}_k, \mathbf{w}^\star \rangle + \overbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}^\star \rangle}^{\geq s}$$

$$\|\mathbf{w}_{k+1}\|_2 = \|\mathbf{w}_k + \mathsf{y}\mathbf{x}\|_2 = \sqrt{\|\mathbf{w}_k\|_2^2 + \underbrace{\|\mathbf{x}\|_2^2}_{\leq R^2} + 2 \underbrace{\langle \mathsf{y}\mathbf{x}, \mathbf{w}_k \rangle}_{\leq \delta}}$$

- The angle approaches 0 ?

$$\cos \angle(\mathbf{w}_{k+1}, \mathbf{w}^\star) := \frac{\langle \mathbf{w}_{k+1}, \mathbf{w}^\star \rangle}{\|\mathbf{w}_{k+1}\|_2 \cdot \|\mathbf{w}^\star\|_2} = \frac{\Omega(k)}{O(\sqrt{k})} \stackrel{?}{\to} 1$$

The Margin

$$\sqrt{\|\mathbf{w}_0\|_2^2 + kR^2 + 2k\delta} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$
$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \langle \mathbf{w}_0, \mathbf{w}^\star \rangle + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2\|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

$\leq$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

The Margin

$$\sqrt{\cancel{\|\mathbf{w}_0\|_2^2} + kR^2 + \cancel{2k\delta}} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$
$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \cancel{\langle \mathbf{w}_0, \mathbf{w}^\star \rangle} + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2 \|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

$$\gamma := \max_{\|\mathbf{w}^\star\|_2 = 1} \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle = \max_{\|\mathbf{w}^\star\|_2 \leq 1} \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle$$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

The Margin

$$\sqrt{\cancel{\|\mathbf{w}_0\|_2^2} + kR^2 + \cancel{2k\delta}} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$
$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \cancel{\langle \mathbf{w}_0, \mathbf{w}^\star \rangle} + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2 \|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

$$\gamma := \max_{\|\mathbf{w}^\star\|_2 = 1} \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle = \max_{\|\mathbf{w}^\star\|_2 \leq 1} \min_i \langle y_i \mathbf{x}_i, \mathbf{w}^\star \rangle$$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

The Margin

$$\sqrt{\cancel{\|\mathbf{w}_0\|_2^2} + kR^2 + \cancel{2k\delta}} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$

$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \cancel{\langle \mathbf{w}_0, \mathbf{w}^\star \rangle} + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2 \|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

$$\gamma := \max_{\|\mathbf{w}^\star\|_2 = 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle = \max_{\|\mathbf{w}^\star\|_2 \leq 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle$$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

The Margin

$$\sqrt{\cancel{\|\mathbf{w}_0\|_2^2} + kR^2 + \cancel{2k\delta}} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$
$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \cancel{\langle \mathbf{w}_0, \mathbf{w}^\star \rangle} + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2 \|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

$$\gamma := \max_{\|\mathbf{w}^\star\|_2 = 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle = \max_{\|\mathbf{w}^\star\|_2 \leq 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle$$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

The Margin

$$\sqrt{\cancel{\|\mathbf{w}_0\|_2^2} + kR^2 + \cancel{2k\delta}} \cdot \|\mathbf{w}^\star\|_2 \geq \|\mathbf{w}_k\|_2 \cdot \|\mathbf{w}^\star\|_2$$

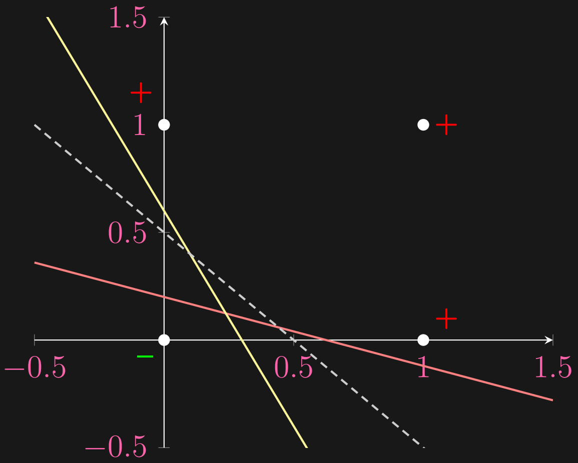
$$\geq \langle \mathbf{w}_k, \mathbf{w}^\star \rangle \geq \cancel{\langle \mathbf{w}_0, \mathbf{w}^\star \rangle} + ks$$

- With $\delta = 0$ and $\mathbf{w}_0 = \mathbf{0}$: the number of mistakes $k \leq \frac{R^2 \|\mathbf{w}^\star\|_2^2}{s^2}$
- What is s and $\mathbf{w}^\star$? Can we choose them to our advantage?

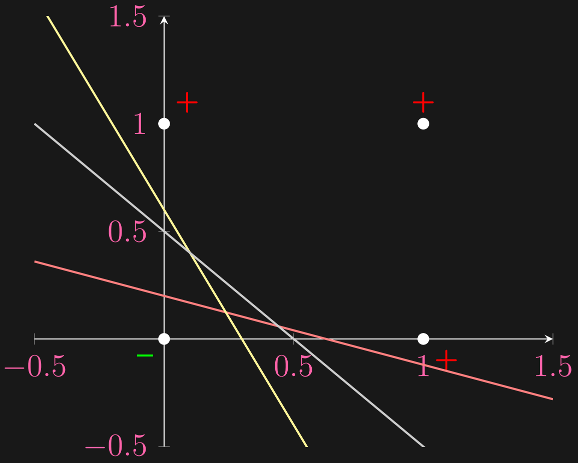
$$\gamma := \max_{\|\mathbf{w}^\star\|_2 = 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle = \max_{\|\mathbf{w}^\star\|_2 \leq 1} \min_i \langle \mathsf{y}_i \mathbf{x}_i, \mathbf{w}^\star \rangle$$

- The larger the margin γ is, the more (linearly) separable the data is, and hence the faster Perceptron converges!

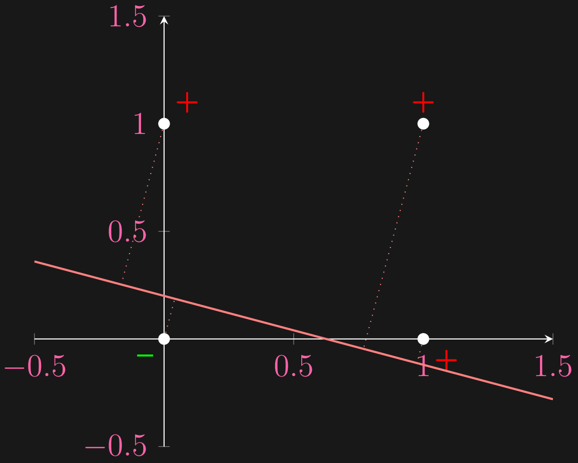
But...Is Perceptron Unique?



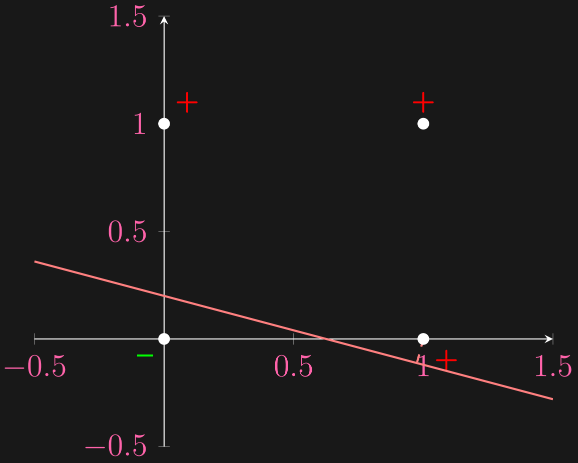
Support Vector Machines: Primal



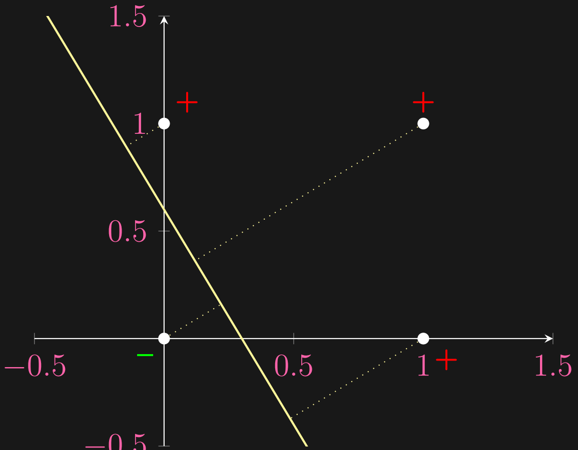
Support Vector Machines: Primal



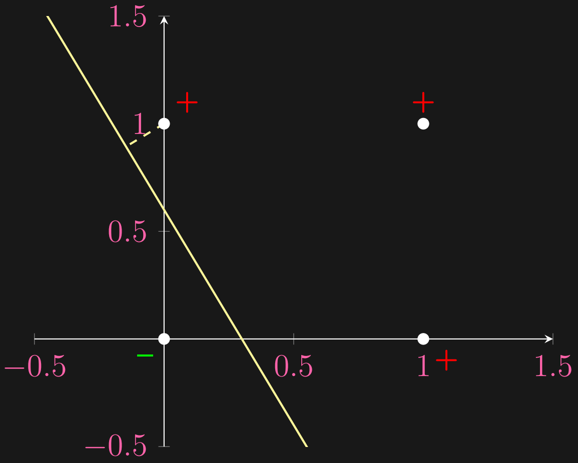
Support Vector Machines: Primal



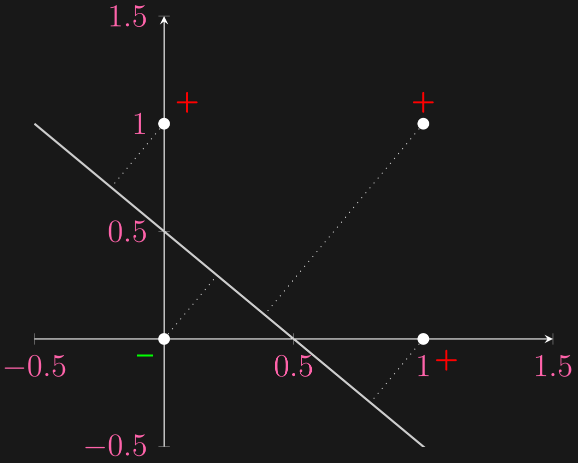
Support Vector Machines: Primal



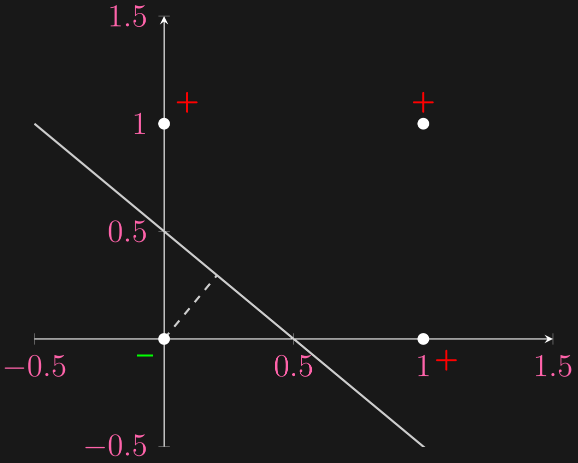
Support Vector Machines: Primal

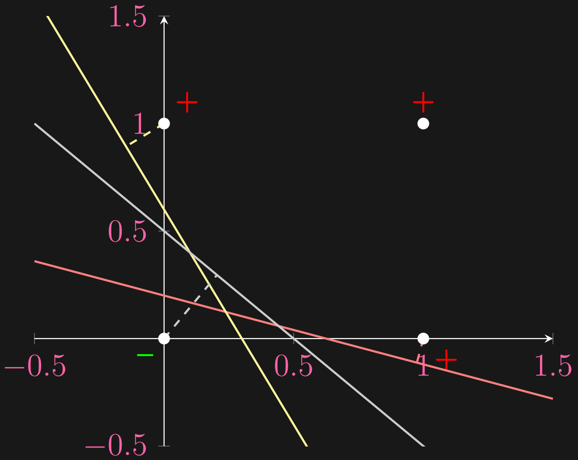


Support Vector Machines: Primal

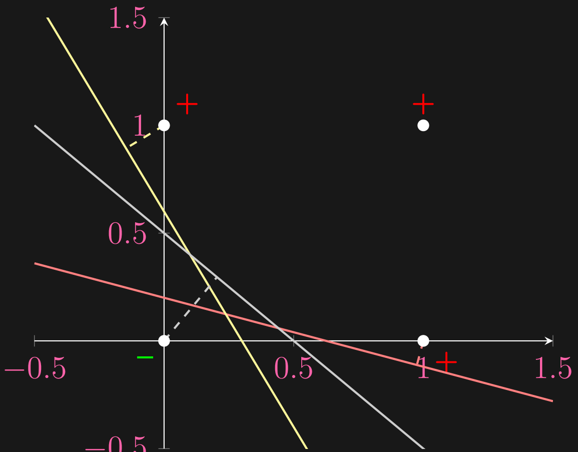


Support Vector Machines: Primal



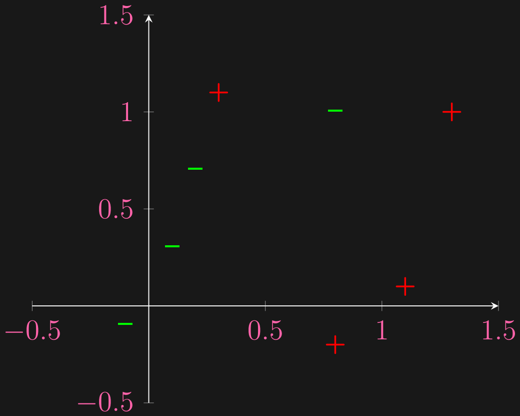

1.5
1
0.5
−0.5
−0.5
0.5
1
1.5
+
+
+
−

Support Vector Machines: Primal



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

Beyond Separability

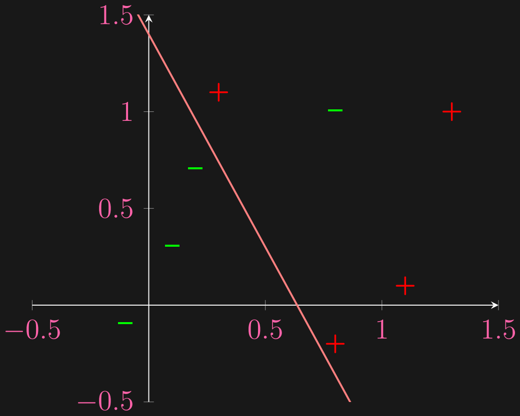


- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

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

- Deeper model through a better feature representation

Beyond Separability

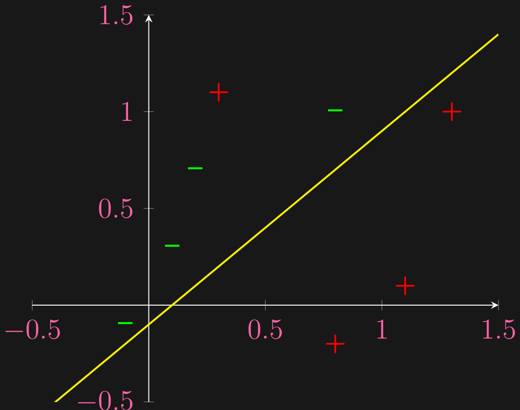


- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

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

- Deeper model through a better feature representation

Beyond Separability

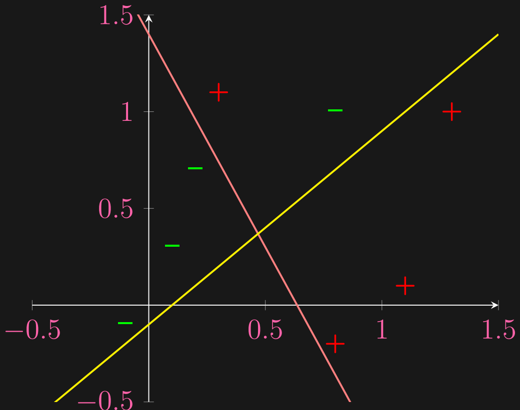


- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

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

- Deeper model through a better feature representation

Beyond Separability

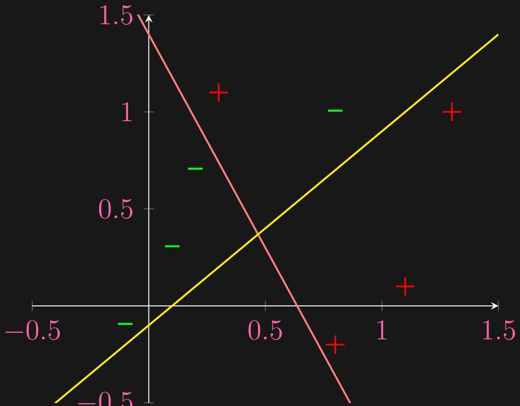


- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

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

- Deeper model through a better feature representation

Beyond Separability

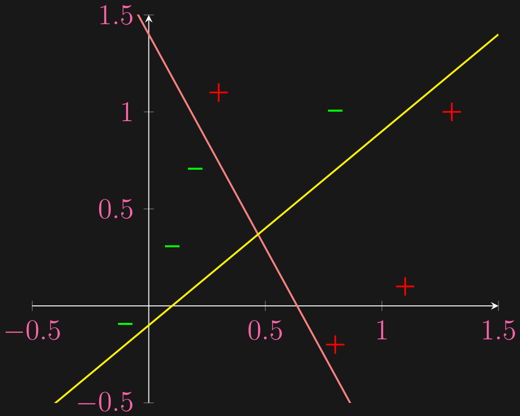


- Soft-margin induced by a reasonable loss ℓ and regularizer reg:

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

- Deeper model through a better feature representation

Beyond Separability



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

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

- Deeper model through a better feature representation

Boundedness Theorem

- Perceptron convergence hinges on the existence of a perfect classifier (i.e., a separating hyperplane)
- What if such an assumption fails? (It will in practice.)

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

The Perceptron iterate $(\mathbf{w}, b)$ is always bounded. In particular, if there is no separating hyperplane, then perceptron cycles.

- "...proof of this theorem is complicated and obscure..." (Minsky and Papert, 1969); see also (Amaldi and Hauser, 2005)

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

Boundedness Theorem

- Perceptron convergence hinges on the existence of a perfect classifier (i.e., a separating hyperplane)
- What if such an assumption fails? (It will in practice.)

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

The Perceptron iterate $(\mathbf{w}, b)$ is always bounded. In particular, if there is no separating hyperplane, then perceptron cycles.

- "...proof of this theorem is complicated and obscure..." (Minsky and Papert, 1969); see also (Amaldi and Hauser, 2005)

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

Boundedness Theorem

- Perceptron convergence hinges on the existence of a perfect classifier (i.e., a separating hyperplane)
- What if such an assumption fails? (It will in practice.)

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

The Perceptron iterate $(\mathbf{w}, b)$ is always bounded. In particular, if there is no separating hyperplane, then perceptron cycles.

- "...proof of this theorem is complicated and obscure..." (Minsky and Papert, 1969); see also (Amaldi and Hauser, 2005)

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

Boundedness Theorem

- Perceptron convergence hinges on the existence of a perfect classifier (i.e., a separating hyperplane)
- What if such an assumption fails? (It will in practice.)

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

The Perceptron iterate $(\mathbf{w}, b)$ is always bounded. In particular, if there is no separating hyperplane, then perceptron cycles.

- "...proof of this theorem is complicated and obscure..." (Minsky and Papert, 1969); see also (Amaldi and Hauser, 2005)

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

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t y_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t y_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathsf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \rightarrow 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathsf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \rightarrow 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathsf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathsf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathbf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

When to Stop Perceptron?

- Online setting: never
- Batch setting
  - maximum number of iterations reached, e.g. iter == maxiter
  - maximum allowed runtime reached
  - training error stops changing
  - validation error stops deceasing
  - weights change falls below tolerance (if using a diminishing step size)

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \mathsf{y}_{I_t} \mathbf{x}_{I_t}, \quad \eta_t \to 0$$

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_{k} \langle \mathbf{x}, \mathbf{w}_{k} \rangle$
- One 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 voting: $\hat{y} := \operatorname{argmax}_{k} \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{\mathsf{y}} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{\mathsf{y}} := \operatorname{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{\mathrm{y}} := \mathrm{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{\mathrm{y}} := \underset{k}{\mathrm{argmax}} \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{\mathrm{y}} := \mathrm{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{\mathrm{y}} := \mathrm{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \mathrm{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \underset{k}{\mathrm{argmax}} \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \underset{k}{\operatorname{argmax}} \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \operatorname*{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \operatorname*{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \operatorname*{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment

Multiclass Perceptron

- 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 highest score: $\hat{y} := \operatorname{argmax}_k \langle \mathbf{x}, \mathbf{w}_k \rangle$
- One 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 voting: $\hat{y} := \operatorname*{argmax}_k \sum_{l \neq k} [\![\langle \mathbf{x}, \mathbf{w}_{k,l} \rangle > 0]\!]$
- Direct extension: assignment