Information Retrieval

Expanding classic IR

- Image data
- Classic OCR and IR
- Classic Computer Vision and IR

Development:
Moshe Friedman

Credits:
Yoav Goldberg, Ido Dagan, Reut Tsarfaty , Moshe Koppel, Wei Song, David Bamman, Ed Grefenstette, Chris Manning, Tsvi Kuflik, Hinrich Schütze, Christina Lioma and more

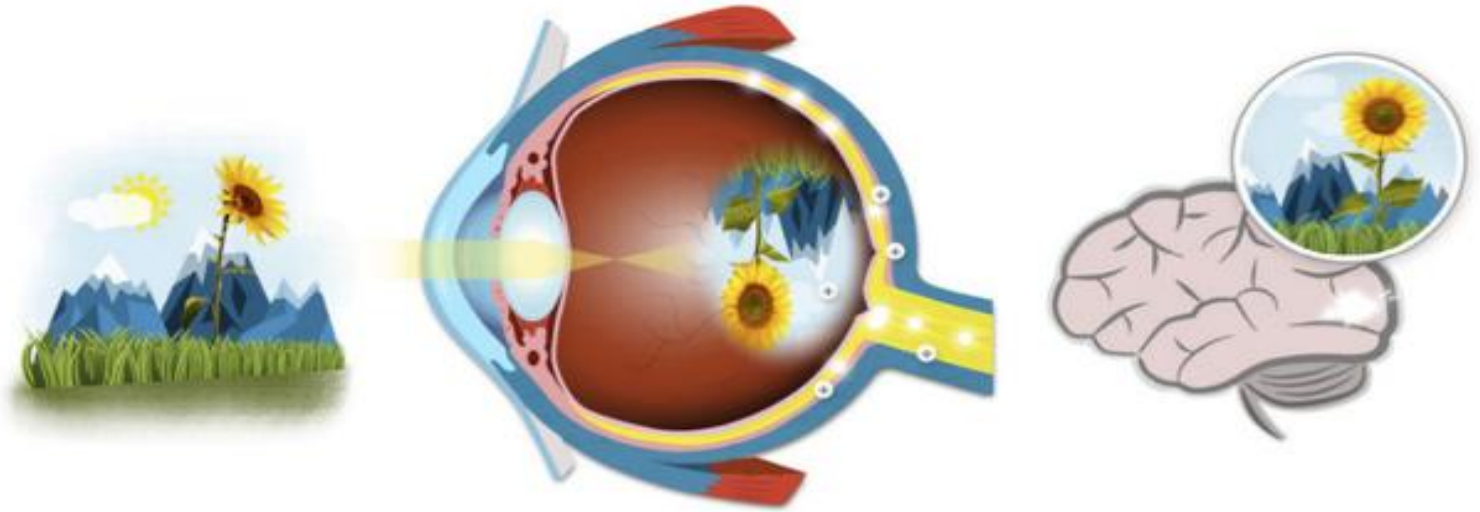
Information Retrieval - administration

Moshe Friedman

Email: moshefr.teach@gmail.com

Reception time: before/after lesson/zoom with coordination

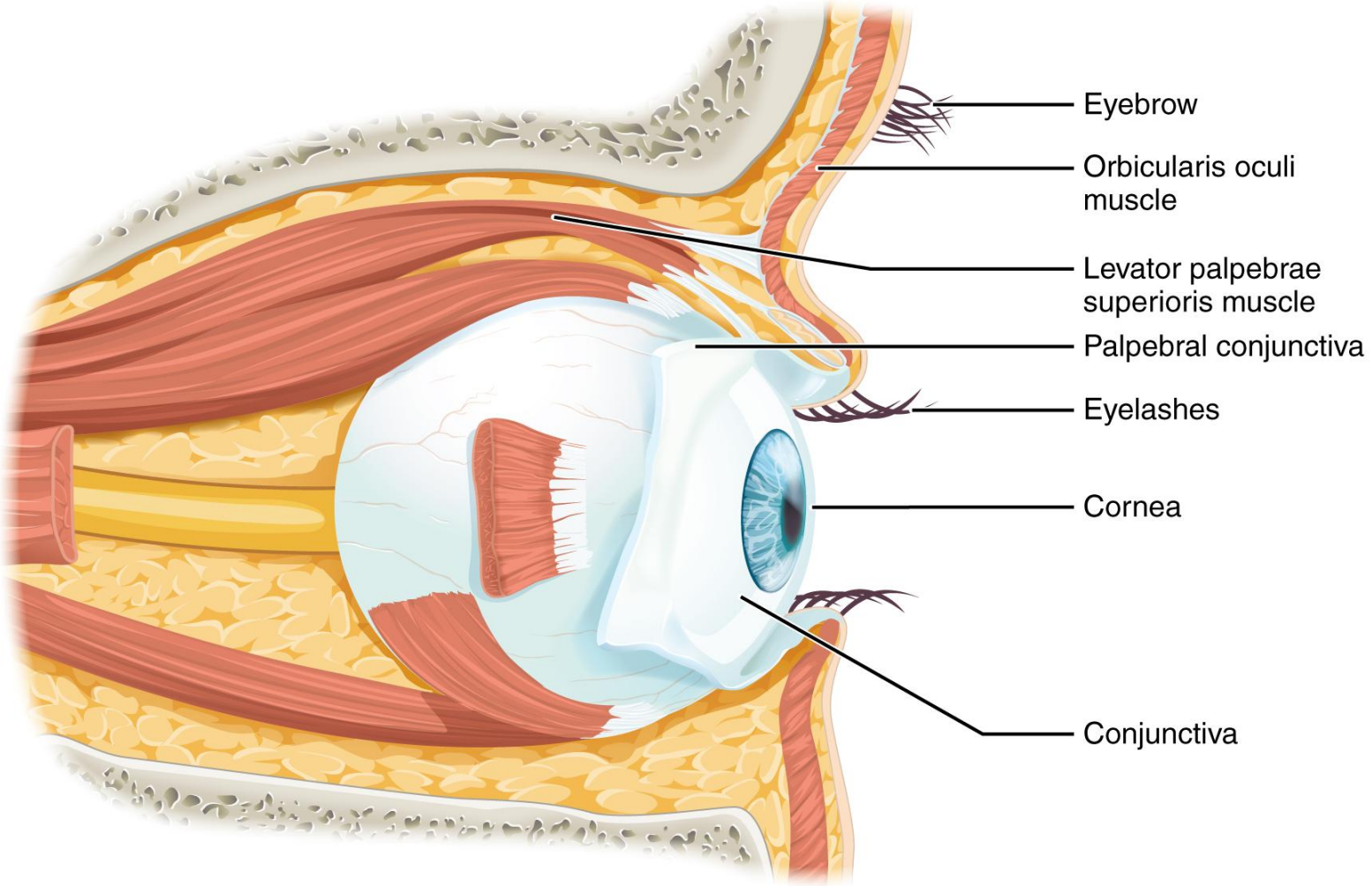
Human Vision - Recap



The eye Translates visual information into electric information

The brain's visual cortex interprets the electrical form of the image

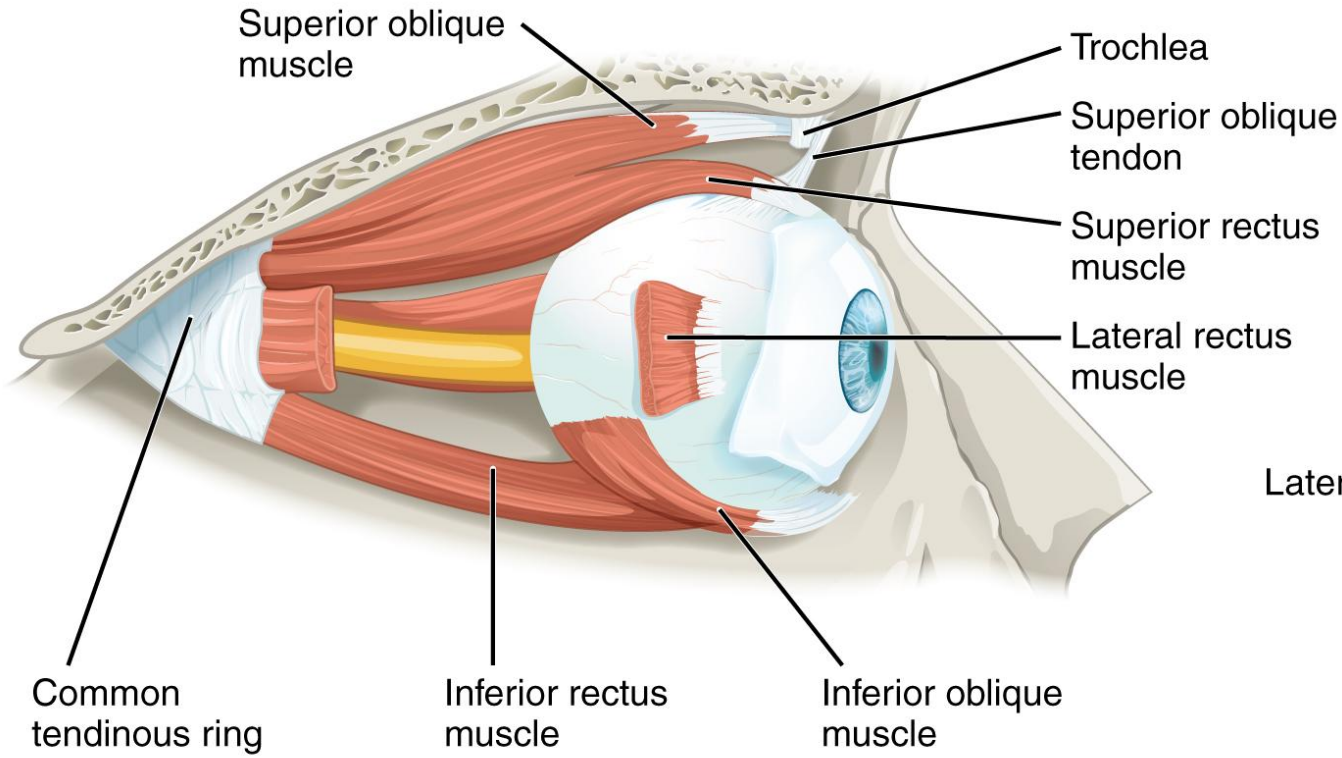
The Eye in the Orbit - Recap



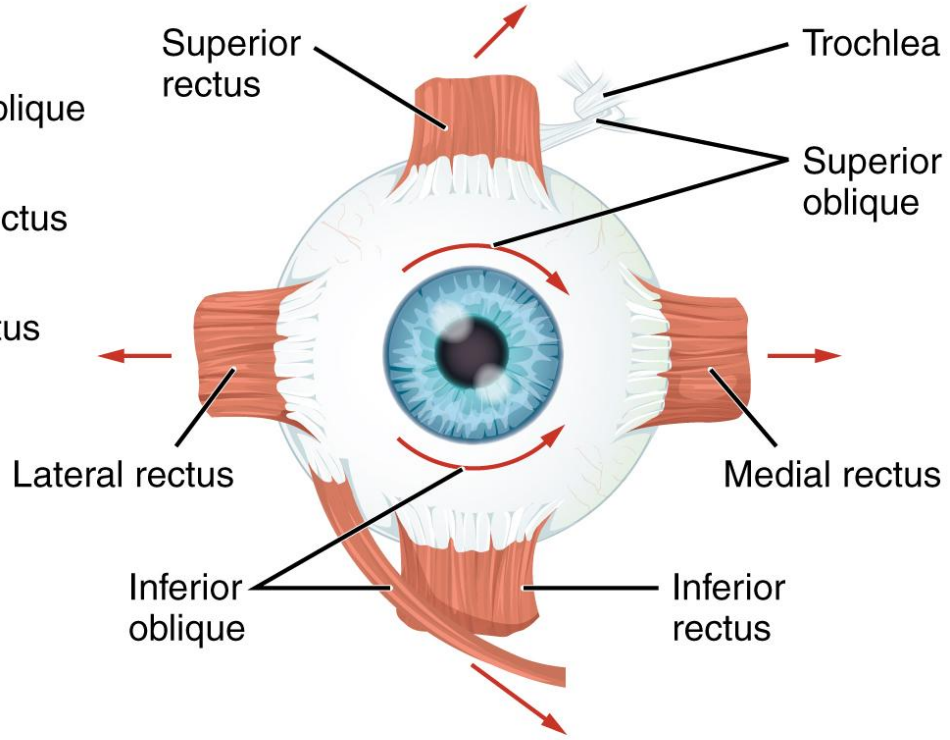
The eye is located within the orbit and surrounded by soft tissues that protect and support its function.

The orbit is surrounded by cranial bones of the skull.

Extraocular Muscles - Recap



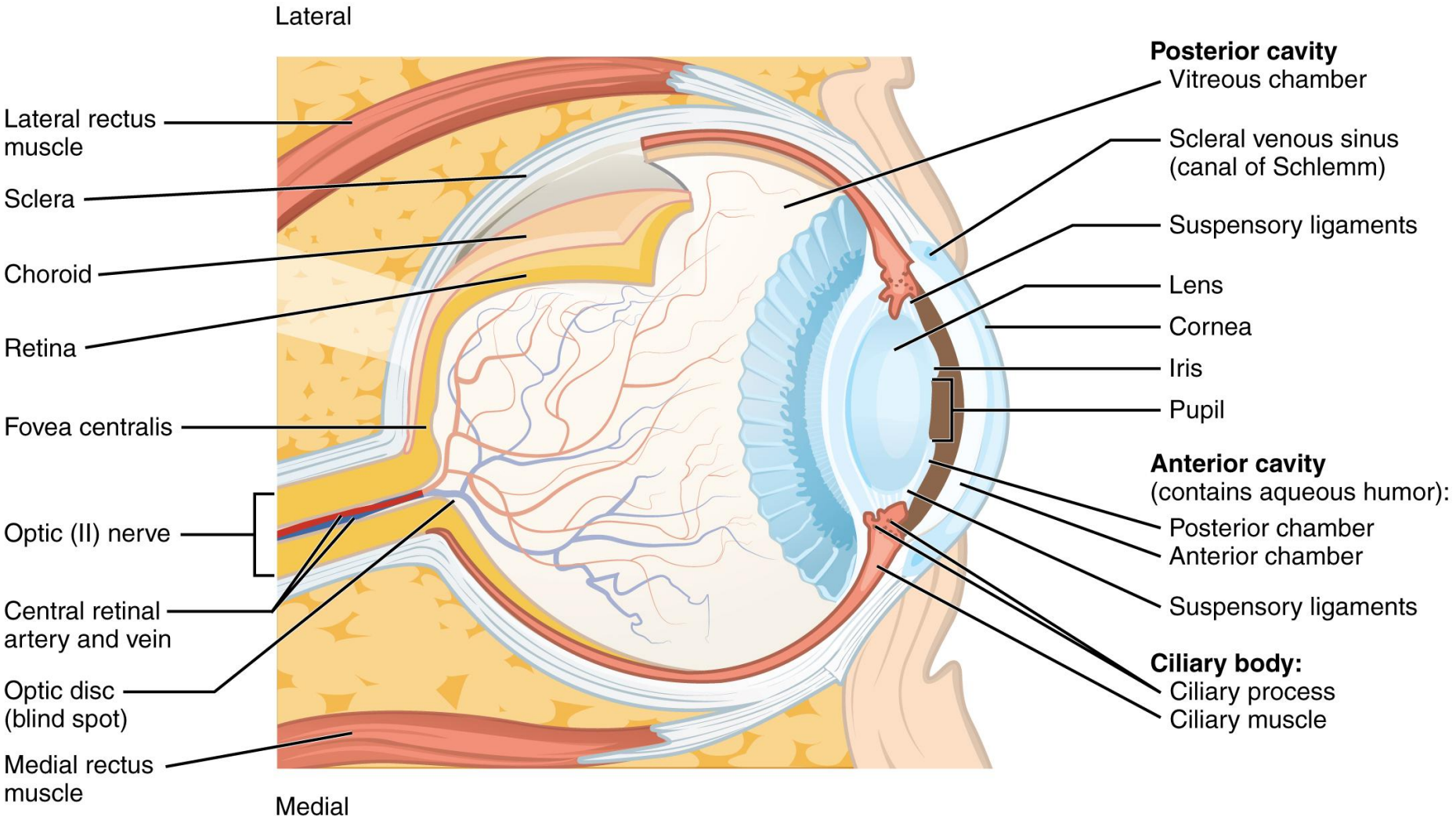
Lateral view of the right eye



Anterior view of the right eye

The extraocular muscles move the eye within the orbit.

Structure of the Eye - Recap



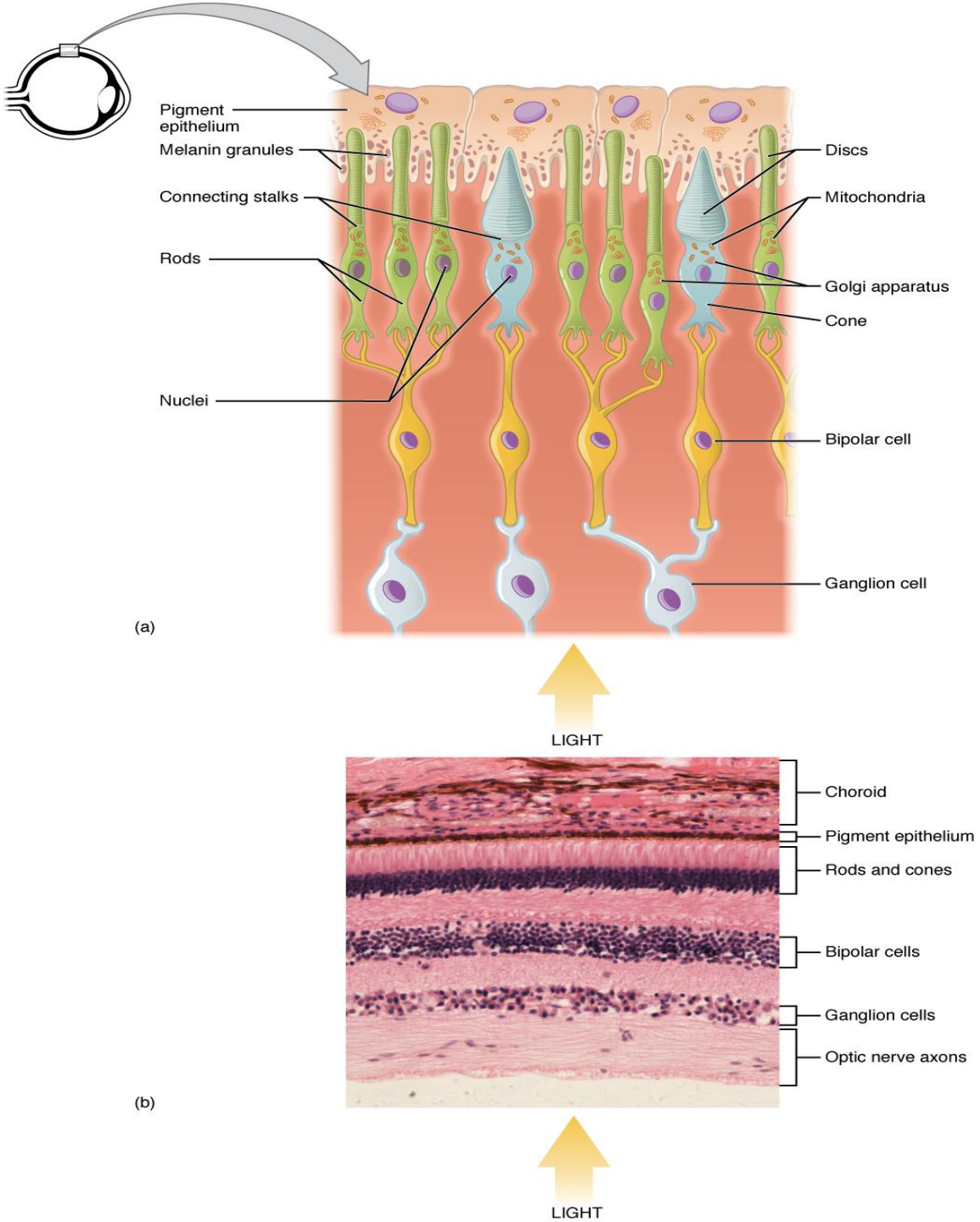
The sphere of the eye can be divided into anterior and posterior chambers.

The wall of the eye is composed of three layers: the fibrous tunic, vascular tunic, and neural tunic.

Within the neural tunic is the retina, with three layers of cells and two synaptic layers in between.

The center of the retina has a small indentation known as the fovea.

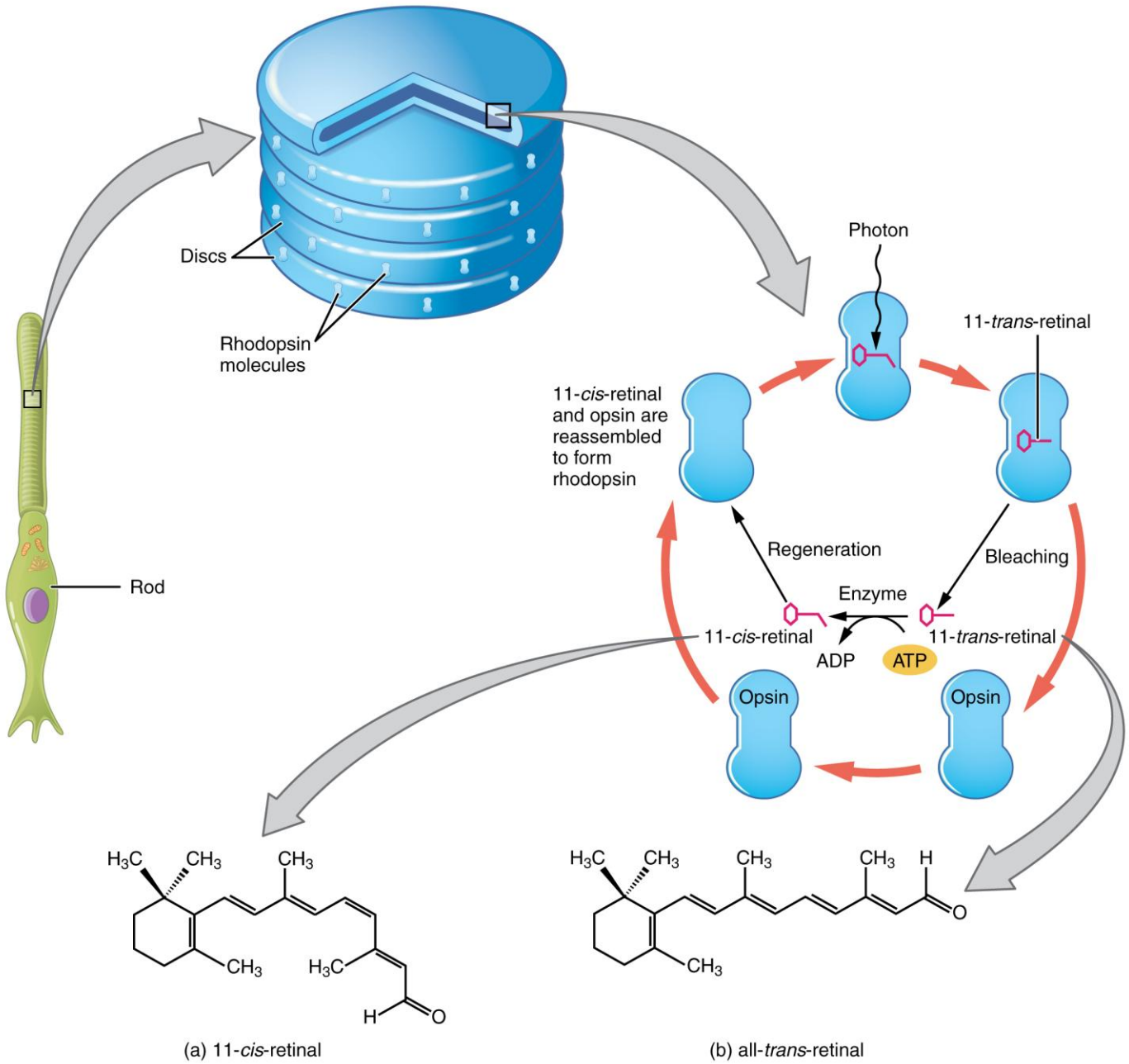
Photoreceptor - Recap



(a) All photoreceptors have inner segments containing the nucleus and other important organelles and outer segments with membrane arrays containing the photosensitive opsin molecules. Rod outer segments are long columnar shapes with stacks of membrane-bound discs that contain the rhodopsin pigment. Cone outer segments are short, tapered shapes with folds of membrane in place of the discs in the rods.

(b) Tissue of the retina shows a dense layer of nuclei of the rods and cones. LM × 800

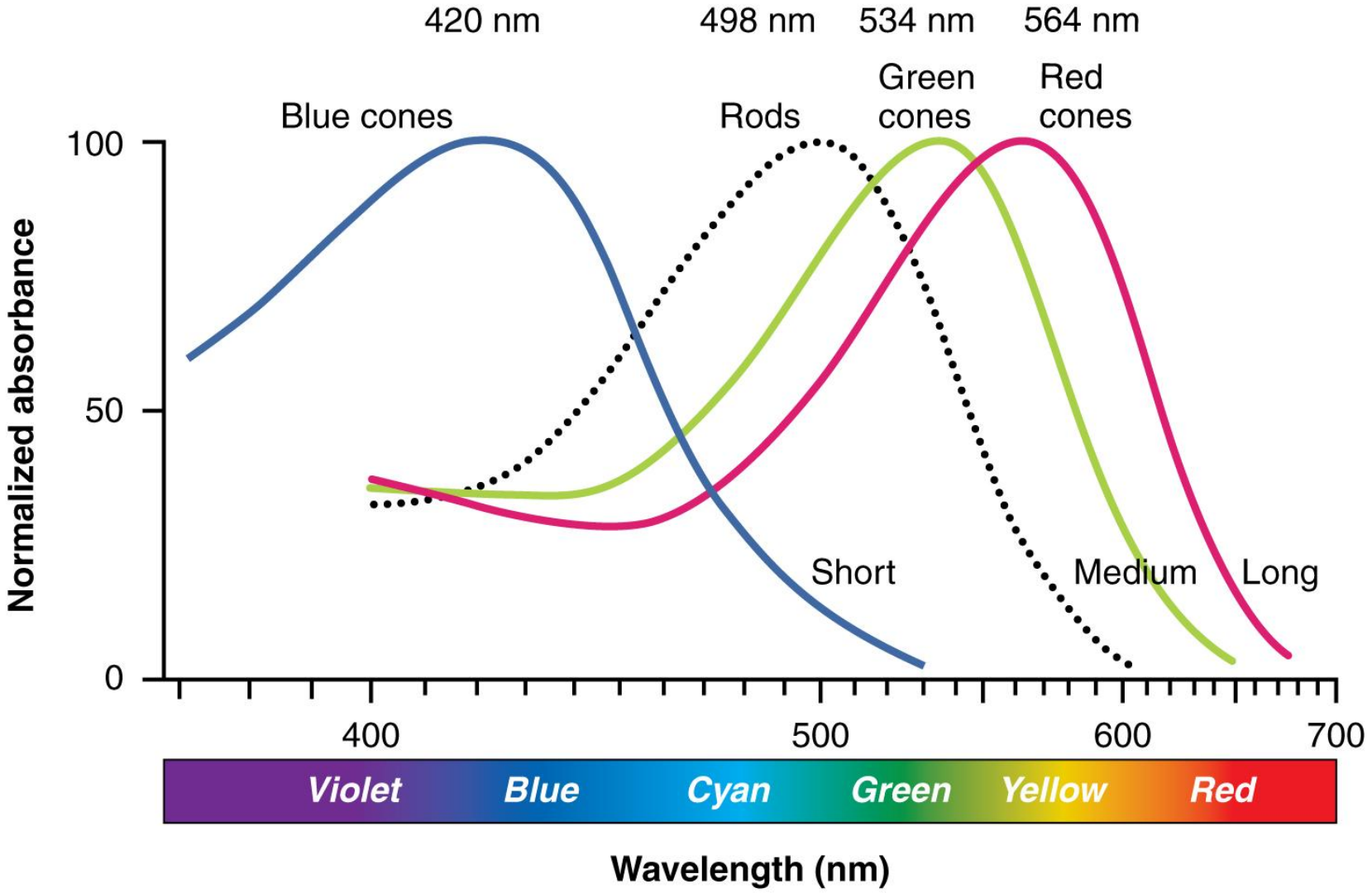
Retinal Isomers - Recap



The retinal molecule has two isomers:

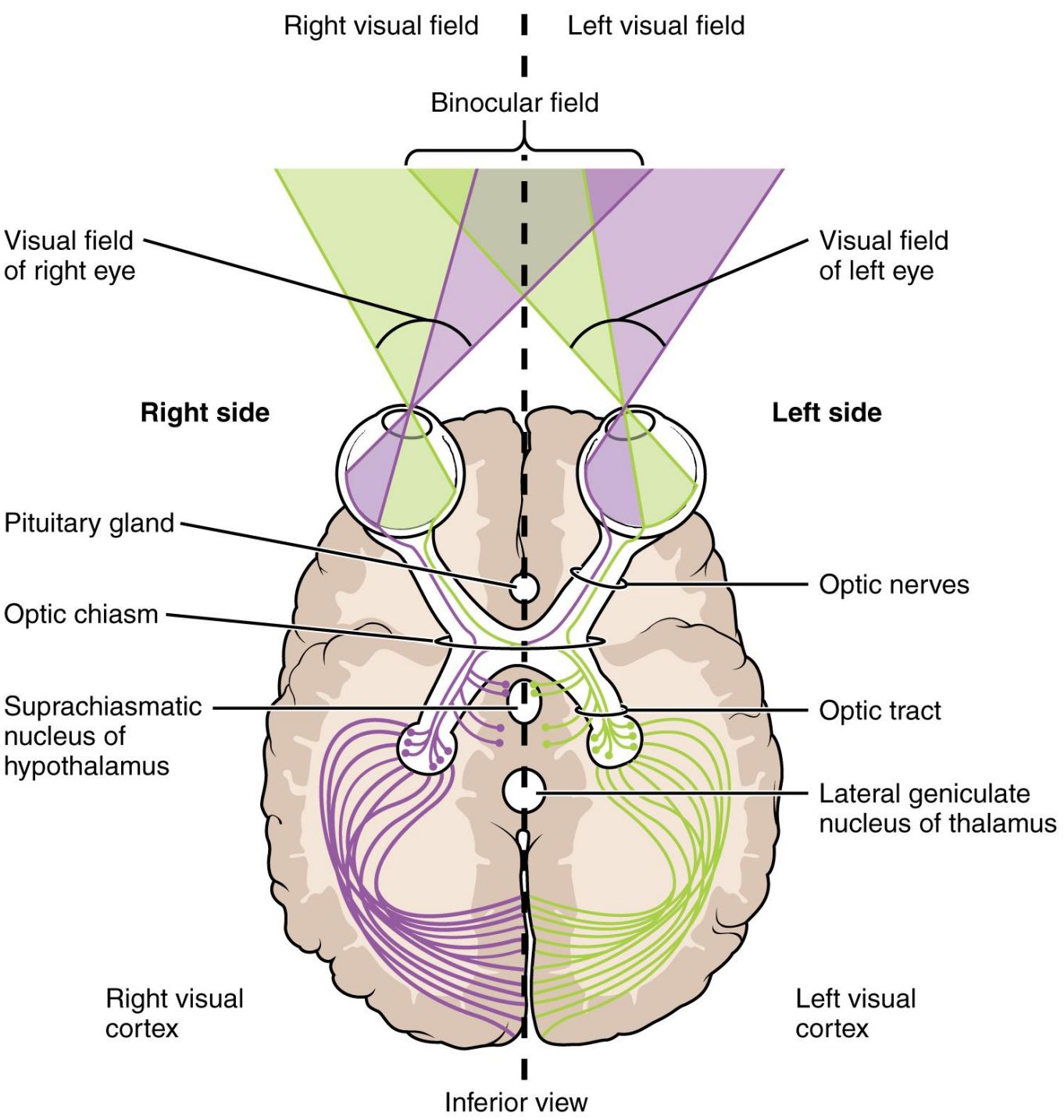
(a) one before a photon interacts with it, and

(b) one that is altered through photoisomerization.



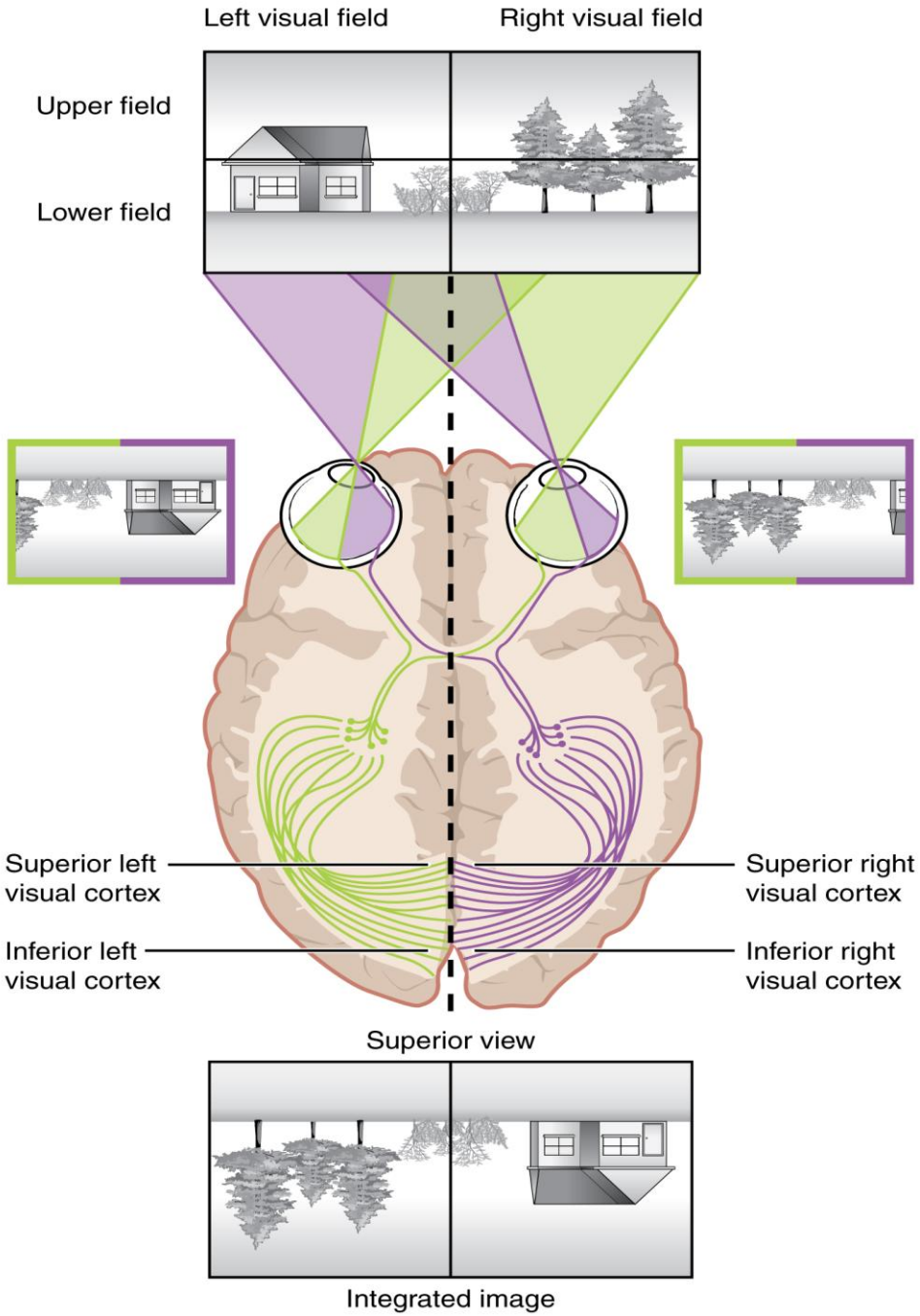
Comparison of Color Sensitivity of Photopigments - Recap

Comparing the peak sensitivity and absorbance spectra of the four photopigments suggests that they are most sensitive to particular wavelengths



Segregation of Visual Field Information at the Optic Chiasm - Recap

Contralateral visual field information from the lateral retina projects to the ipsilateral brain, whereas ipsilateral visual field information has to decussate at the optic chiasm to reach the opposite side of the brain.



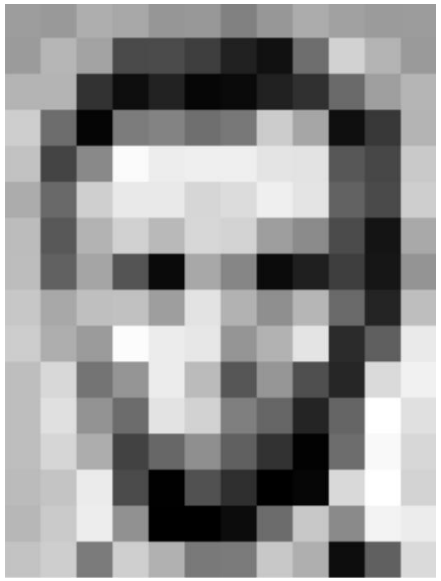
Topographic Mapping of the Retina onto the Visual Cortex - Recap

The visual field projects onto the retina through the lenses and falls on the retinae as an inverted, reversed image.

The topography of this image is maintained as the visual information travels through the visual pathway to the cortex.

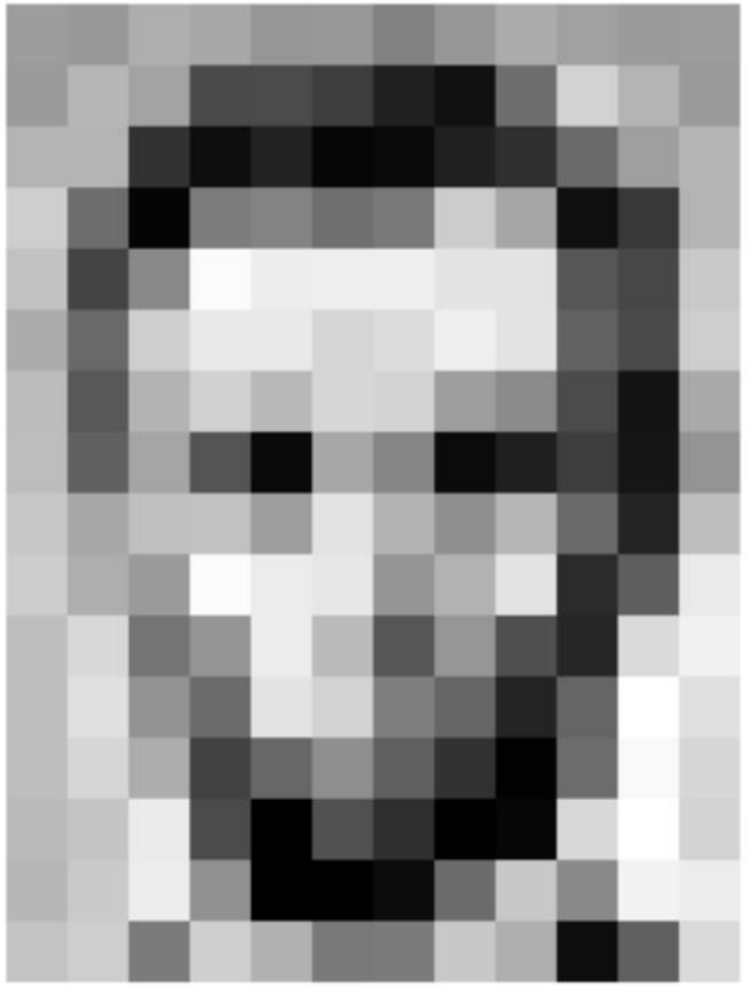
Computer Vision - Recap

What computers “see”?



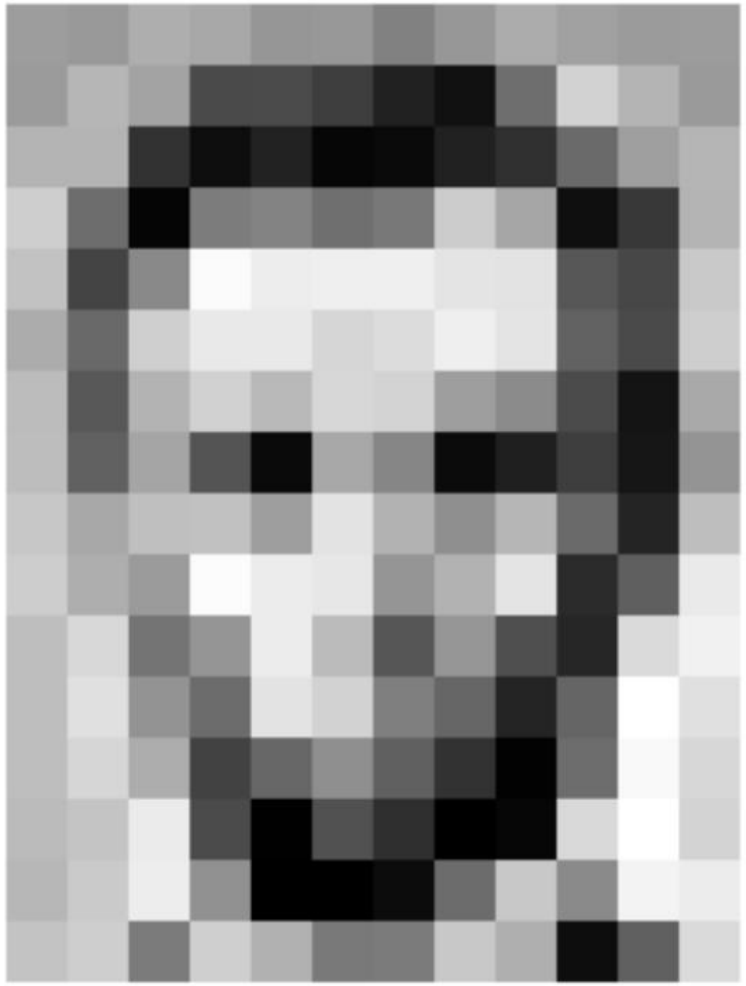
Images are numbers

Images are numbers - Recap



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

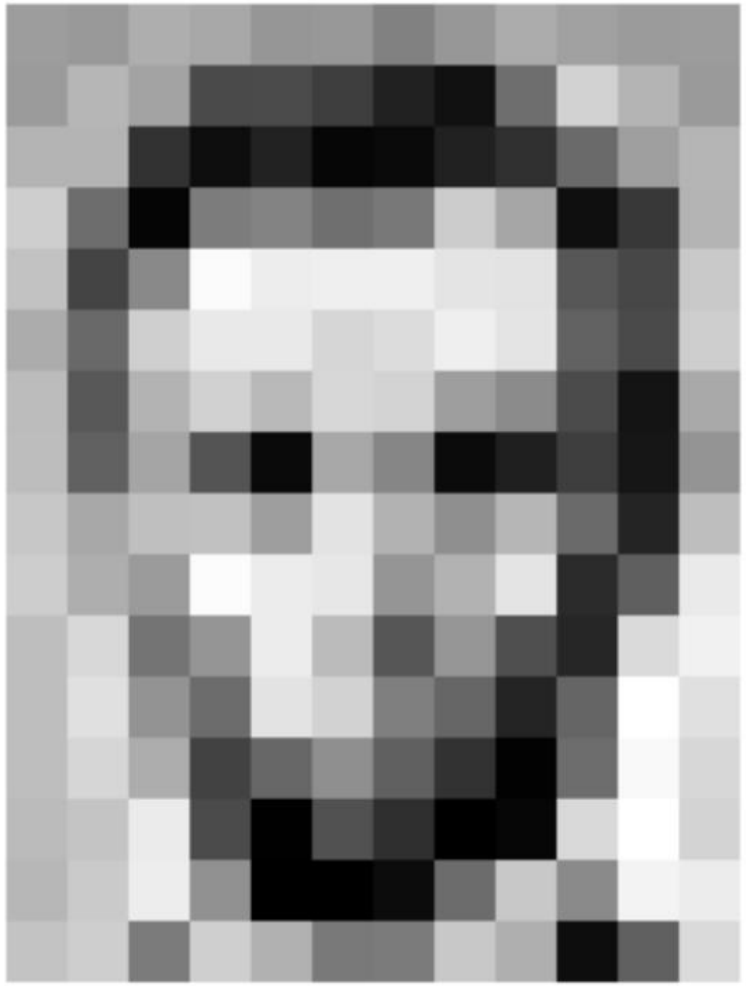
Images are numbers - Recap



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
										22	148
										36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Low numbers are darker

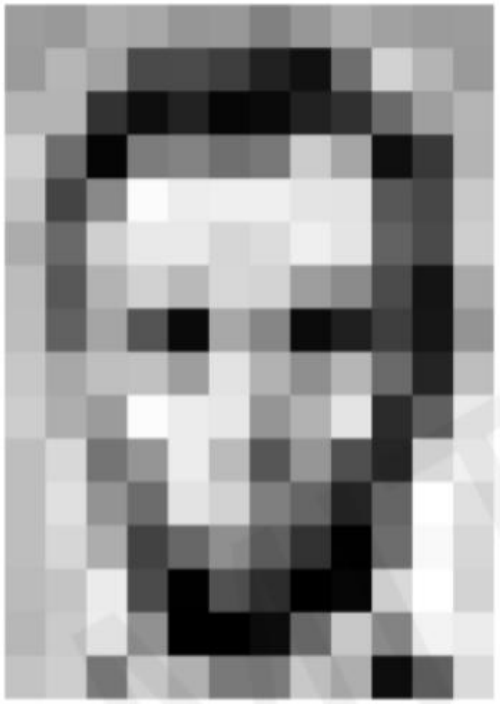
Images are numbers - Recap



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	[illegible]			111	120	204	166	15	56	180
194	68	[illegible]	251	[illegible]	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	[illegible]	74	206
[illegible]										20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

high numbers are brighter

Images are numbers - Recap



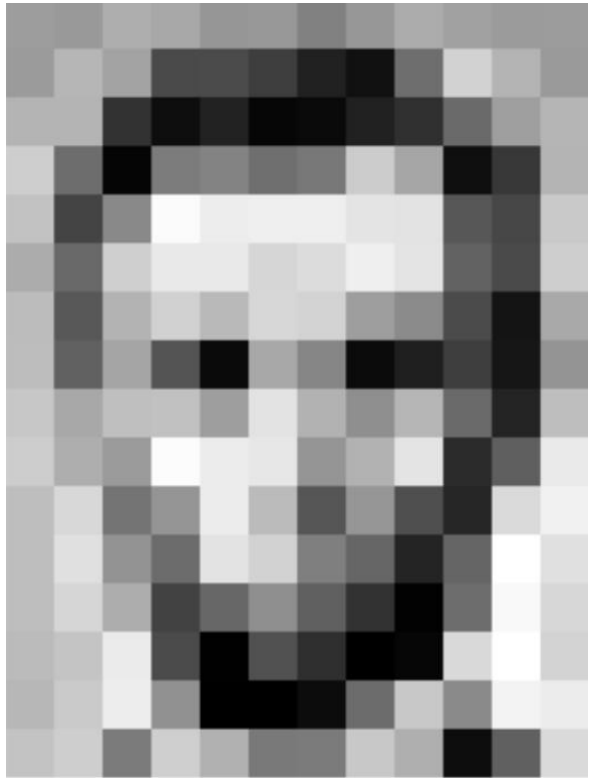
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

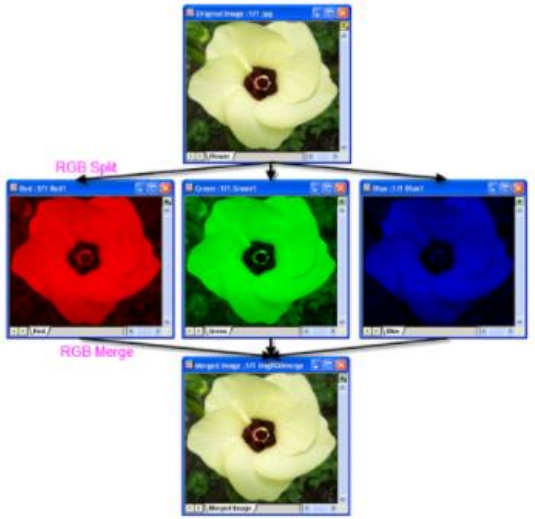
Grey level image - Recap



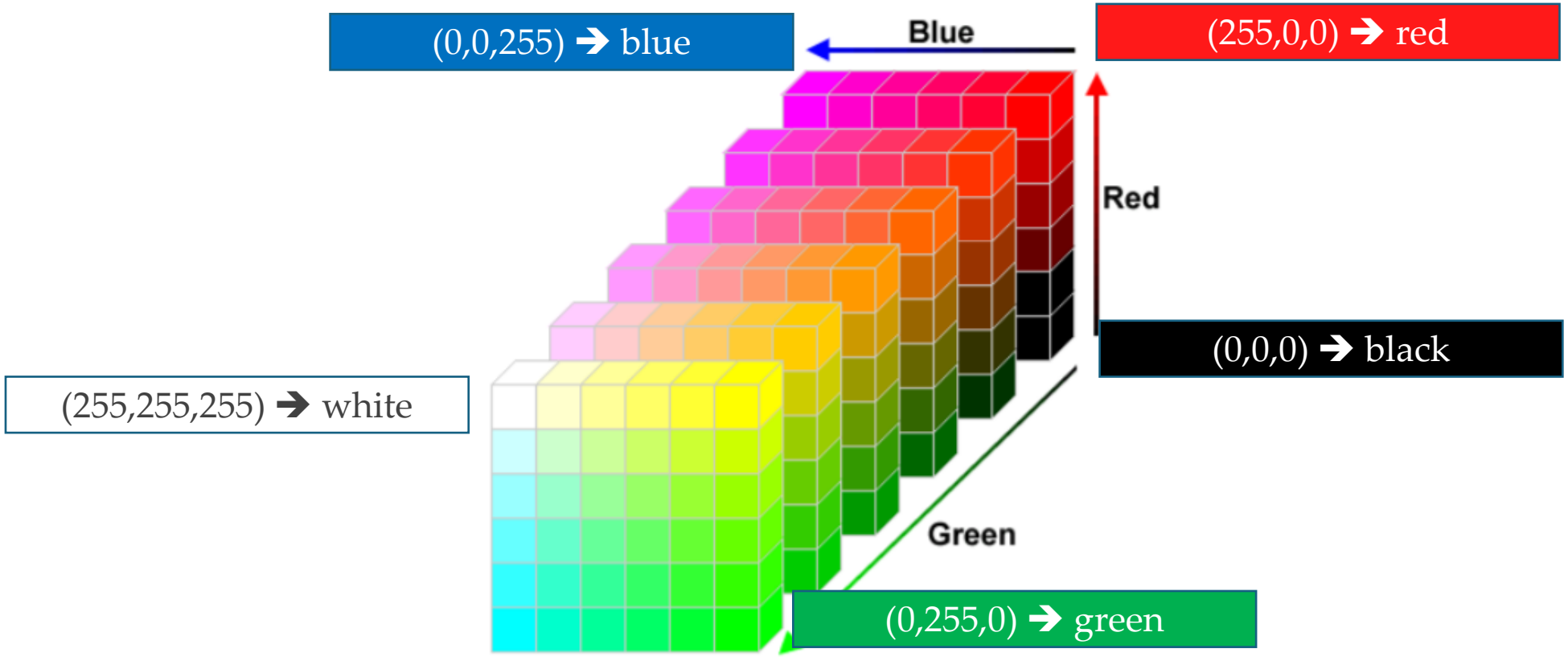
1 b/w band
0-255
0 = black
255 = white

Color image - Recap

תמונה צבעונית היא הרכבה של 3 תמונות



RGB Pallet - Recap



OCR

OCR

Classic OCR (before deep learning) is an Image Processing problem, not a Learning problem.

- It relies on geometry, projection profiles, and template matching.
- The IR systems then had to handle the inevitable typos
  ("teh" instead of "the", "1" instead of "l") using N-Gram (Fuzzy) Indexing.

The Architecture

- **Image Pre-processing**: Binarization & Deskewing.
- **Segmentation**: Using Projection Profiles (histograms of pixel density) to cut out lines and characters.
- **Recognition**: Template Matching (Correlation) or Feature Engineering (HOG/Zoning).
- **Indexing**: N-Gram decomposition to make the search engine resilient to OCR errors.

Step 1: The 'Scanned' Noisy Document

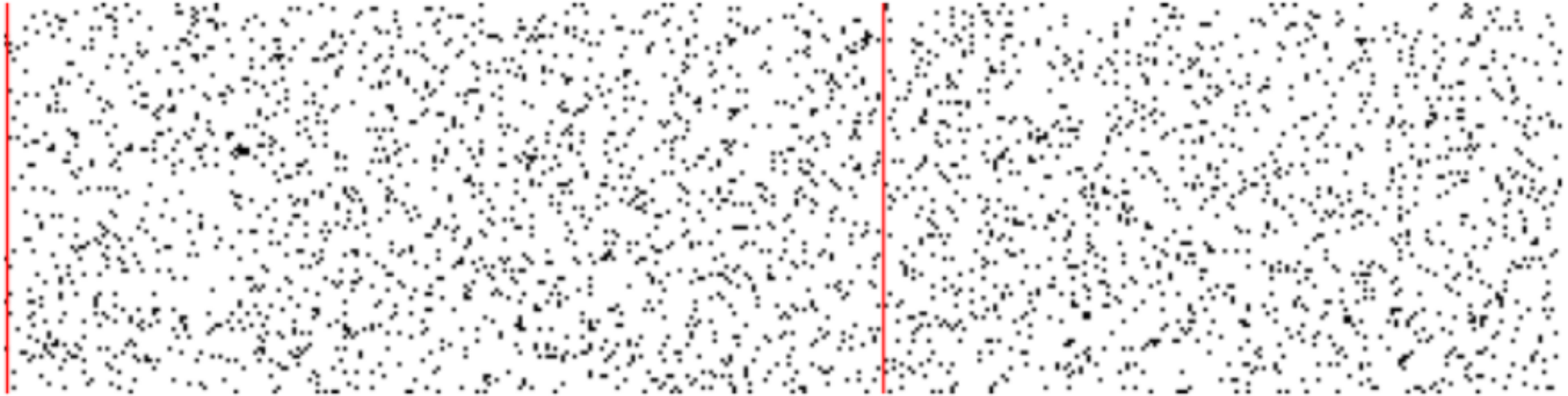
Step 2: Classic Segmentation (Projection Profiles)

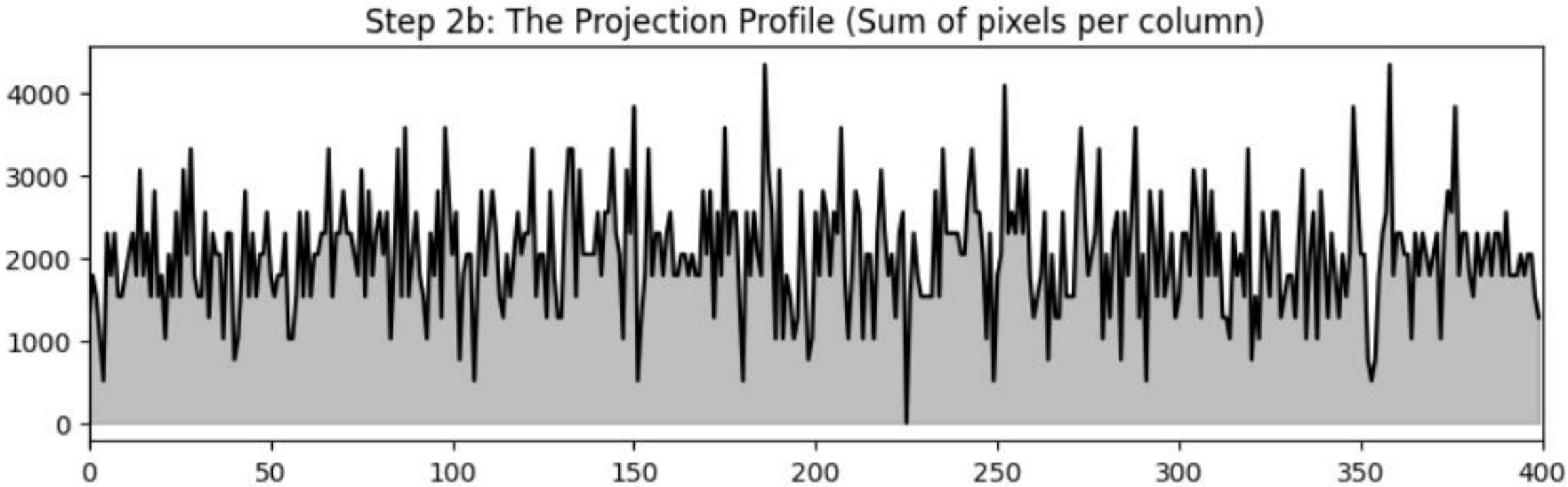
Before Neural Networks looked at the whole image, classic OCR systems sliced the image.

They used Projection Profiles:

- **Horizontal Projection**: Sum pixels in every row.
    - Valleys = whitespace between lines.
- **Vertical Projection**: Sum pixels in every column.
    - Valleys = whitespace between characters.

Step 2a: Character Segmentation via Vertical Projection




Step 2b: The Projection Profile (Sum of pixels per column)
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
400

Step 3: Recognition (Template Matching)

Classic OCR compared the segmented pixel blob against a database of known fonts ("Templates").
It calculated the Correlation (overlap).
The template with the highest overlap score "won".

Note: In this simulation, we will show how this method fails on noisy characters, producing the "Dirty Text" problem.

- Original Text: REFERENCE OCR
- Output : REFERENQE <-- Note the error!

Step 4: The IR Solution (N-Gram Indexing)

This is the critical IR component.

If the user searches for "Reference", but the database contains "Referenqe", a standard Exact Match fails.

Classic systems used Character N-Grams (q-grams).

Step 4: The IR Solution (N-Gram Indexing)

This is the critical IR component.

If the user searches for "Reference", but the database contains "Referenqe", a standard Exact Match fails.

Classic systems used Character N-Grams (q-grams).

Concept:

- **Word**: REFERENCE → `REF`, `EFE`, `FER`, `ERE`, `REN`, `ENC`, `NCE`
- **OCR**: REFERENQE → `REF`, `EFE`, `FER`, `ERE`, `REN`, `ENQ`, `NQE`
- **Overlap**: 5 out of 7 n-grams match. High similarity!

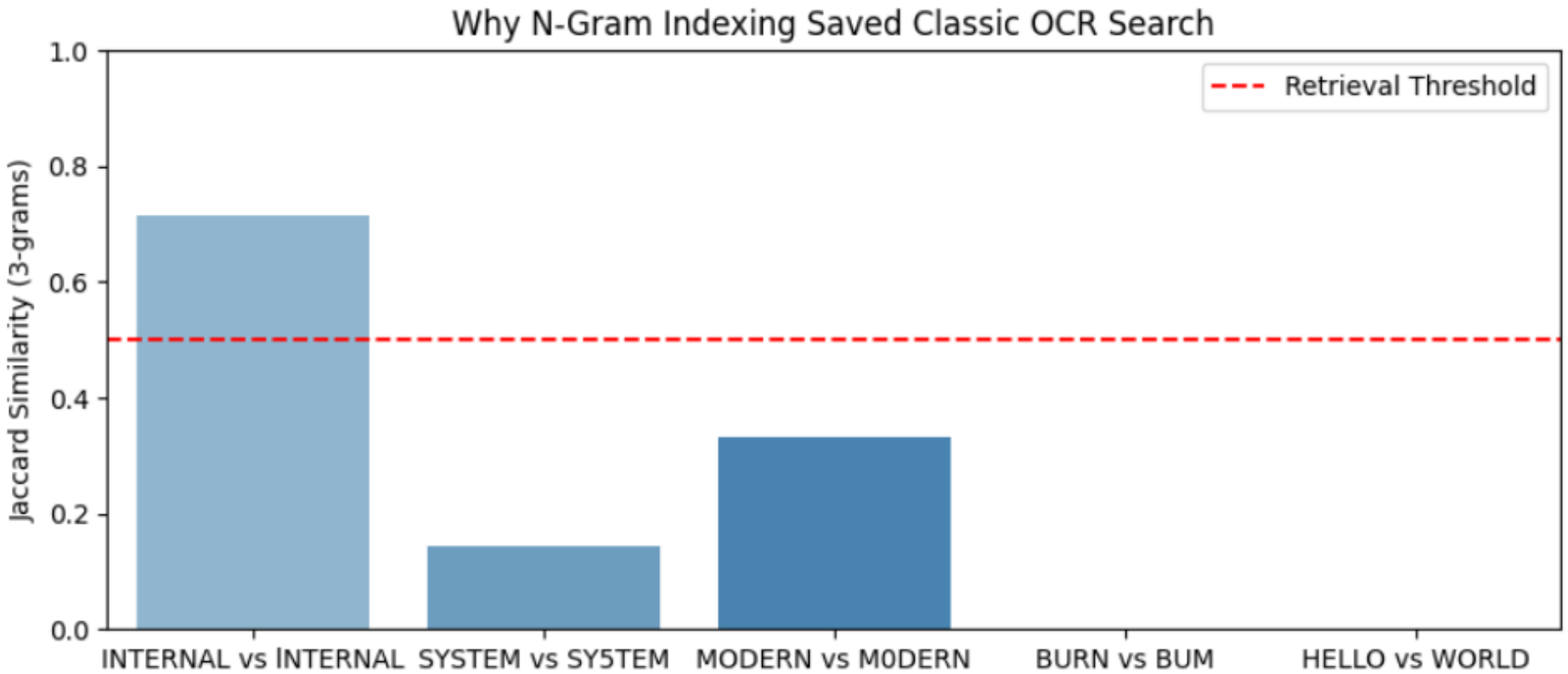
- Query n-grams: ['efe', 'enc', 'ere', 'fer', 'nce', 'ref', 'ren']
- Doc n-grams: ['efe', 'enq', 'ere', 'fer', 'nqe', 'ref', 'ren']
- Matching n-grams: ['efe', 'ere', 'fer', 'ref', 'ren']
- Jaccard Similarity Score: 0.56

→RESULT: Document Missed.

Step 5: Visualizing the "Robustness" Matrix

To verify this approach works for various **classic OCR errors**,
we can visualize the similarity matrix between correct words and common OCR misinterpretations.

- S ↔ 5
- l ↔ 1
- rn ↔ m



Summary of the Flow

- **Input**: Noisy image.
- **Segmentation**: Used Horizontal/Vertical Projections to chop the image into bits.
- **Recognition**: Used Template Matching (pixel overlap).
  This was brittle and produced "Dirty Text" (e.g., "SY5TEM").
- **IR Adaptation**: The search engine didn't index whole words.
  It indexed rolling 3-character windows (trigrams).
- **Result**: Even if rn became m, the other trigrams (bur, urn) would overlap enough to retrieve the document "Burn".

The Classic Computer Vision Flow (before CNNs)

Classic Computer vision is about "*Hand-Crafted Features*."

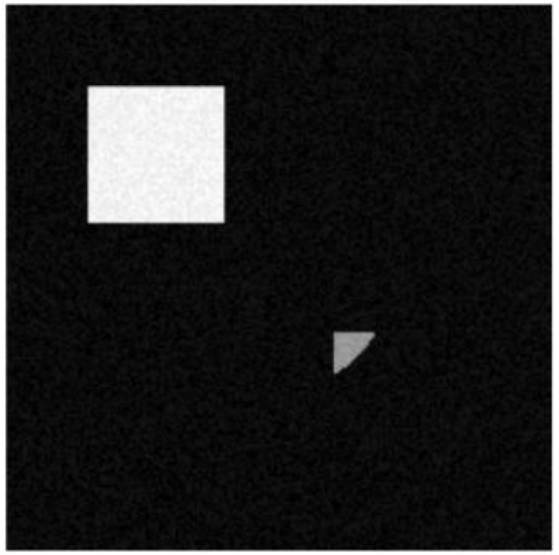
We didn't learn features; we mathematically defined them.

The Pipeline:

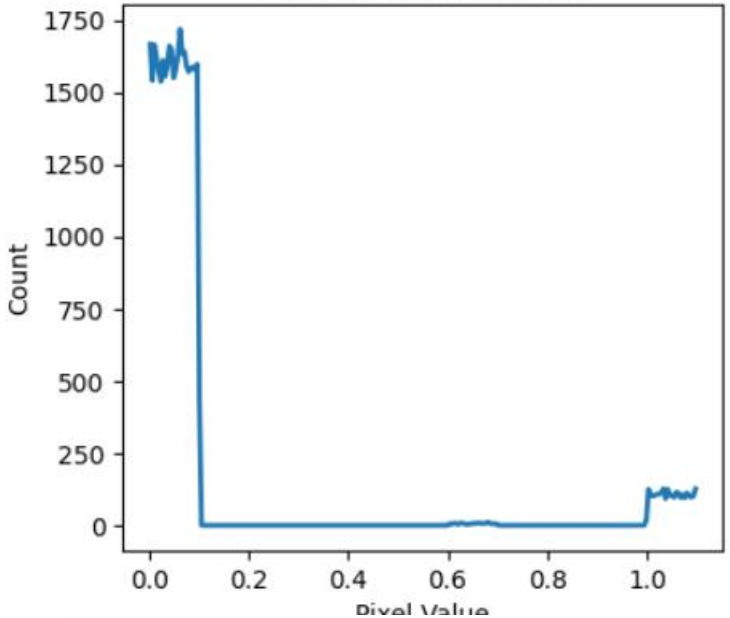
- **Preprocessing**: Grayscale/Normalization.
- **Global Analysis**: Pixel Intensity Histograms.
- **Edge Detection**: Finding structural boundaries (Canny/Sobel).

Feature Extraction: HOG (Histogram of Oriented Gradients).

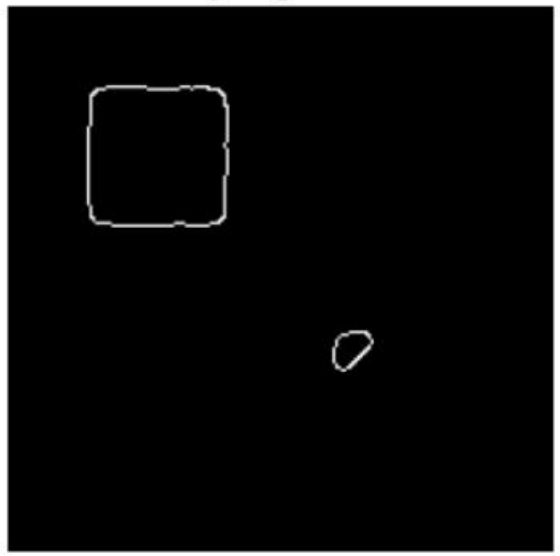
1. Raw Input (Noisy)



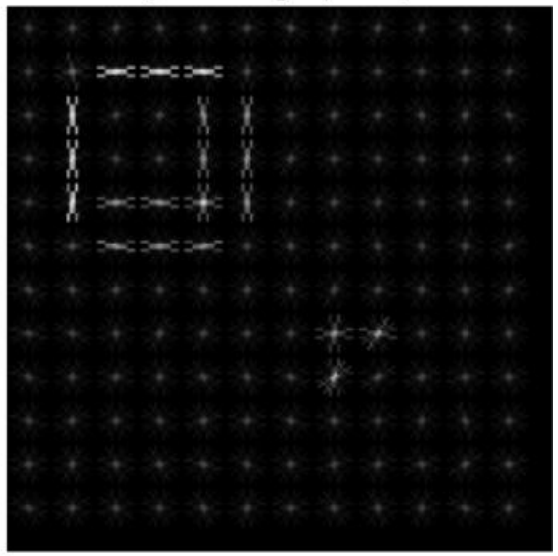
2. Intensity Histogram



3. Canny Edge Detection



4. HOG Features
(The 'Fingerprint')

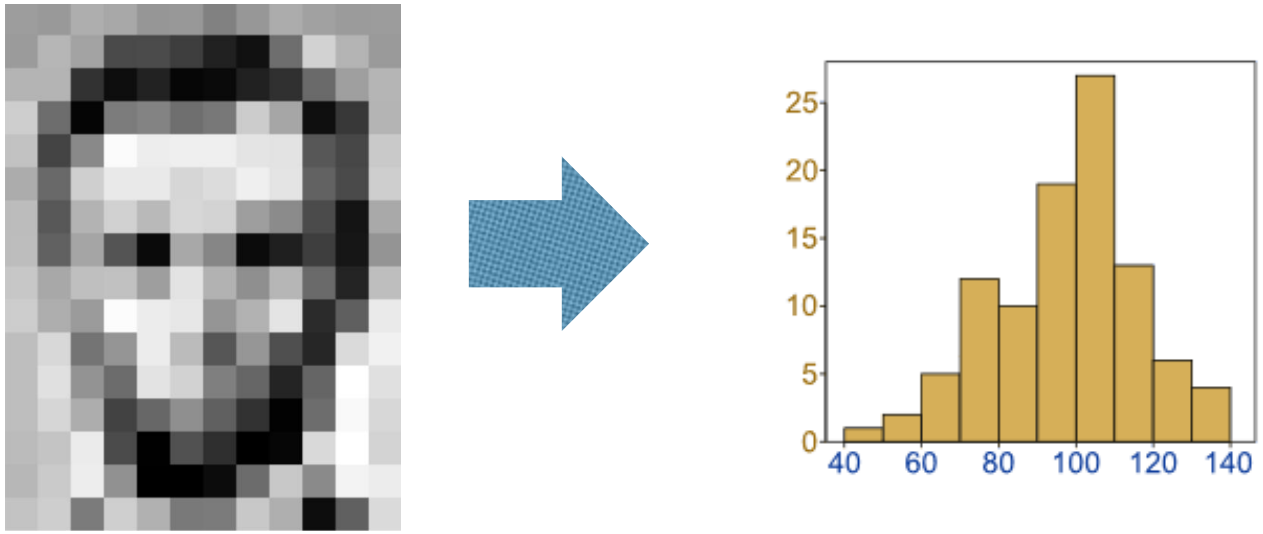


Explanation of Steps:

- **Histogram**: You see three peaks:
  The black background (near 0), the gray circle (near 0.6), and the white square (near 1.0).
  This is simple, but ignores location.
- **Edges**: The Canny filter calculates gradients.
  Note how it successfully ignores the random noise but highlights the borders of the shapes.
- **HOG**: Look closely at the last plot. You see "star-like" lines.
  These represent the dominant direction of the edges in that square.
  This grid of directions is the feature vector used to train classifiers (like SVMs) to recognize objects.

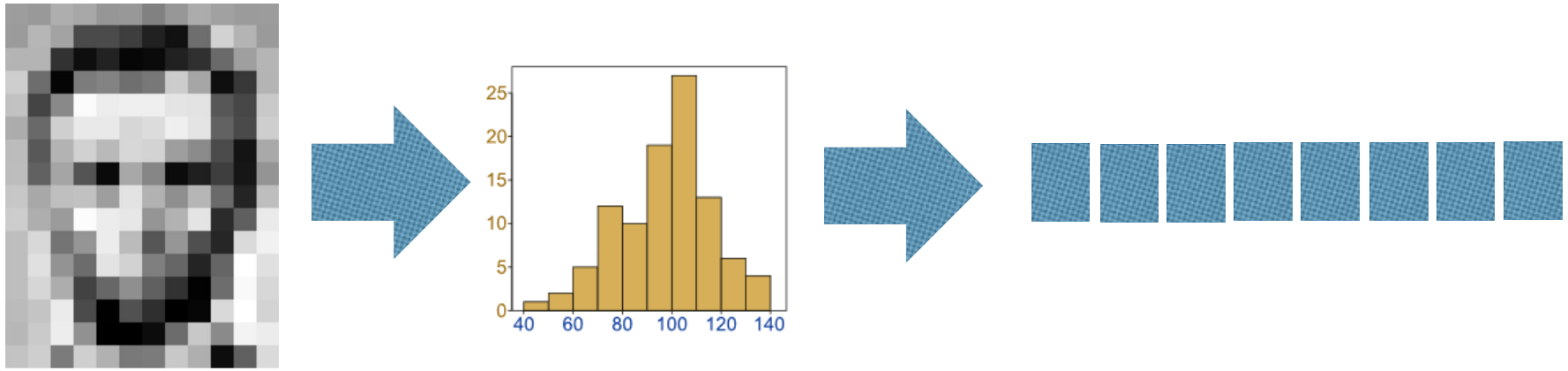
From image to features

Image Descriptors, e.g., Histograms



From image to features

Image Descriptors, e.g., Histograms



Traditional approach

Manually Identify key features in each category.

Nose
Eyes
Mouth

Wheels
License Plate
Headlights

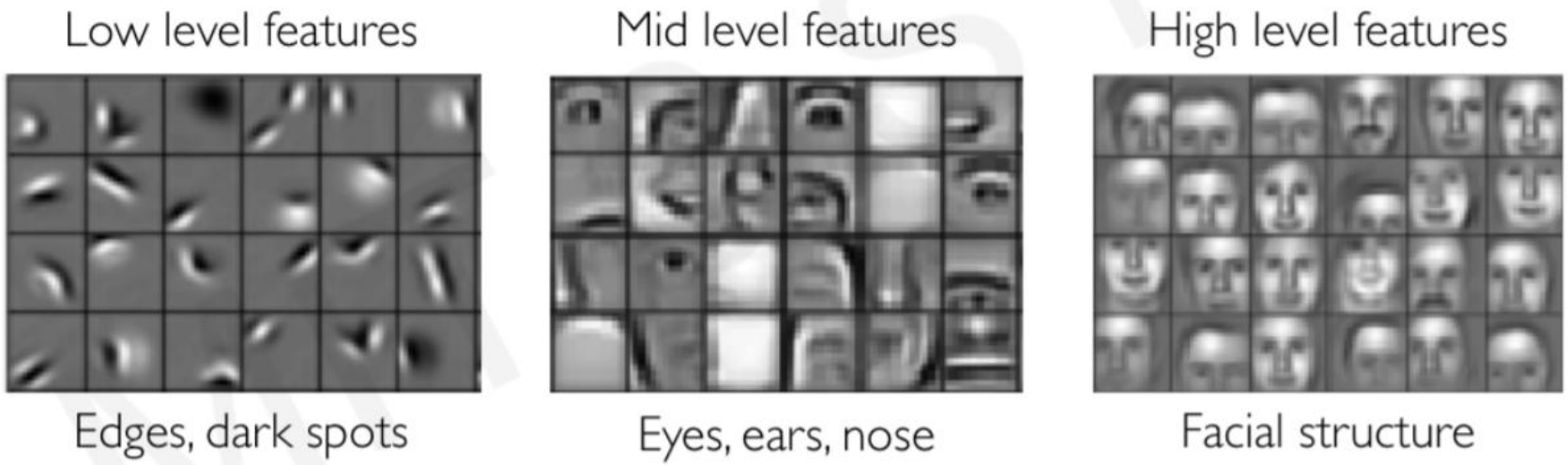
Train ML classifier on these features.

Traditional approach

- This was the traditional computer vision approach
- Engineers handcrafted new features & handed them as vectors to ML algorithms
- Better features / combinations of features improved performance

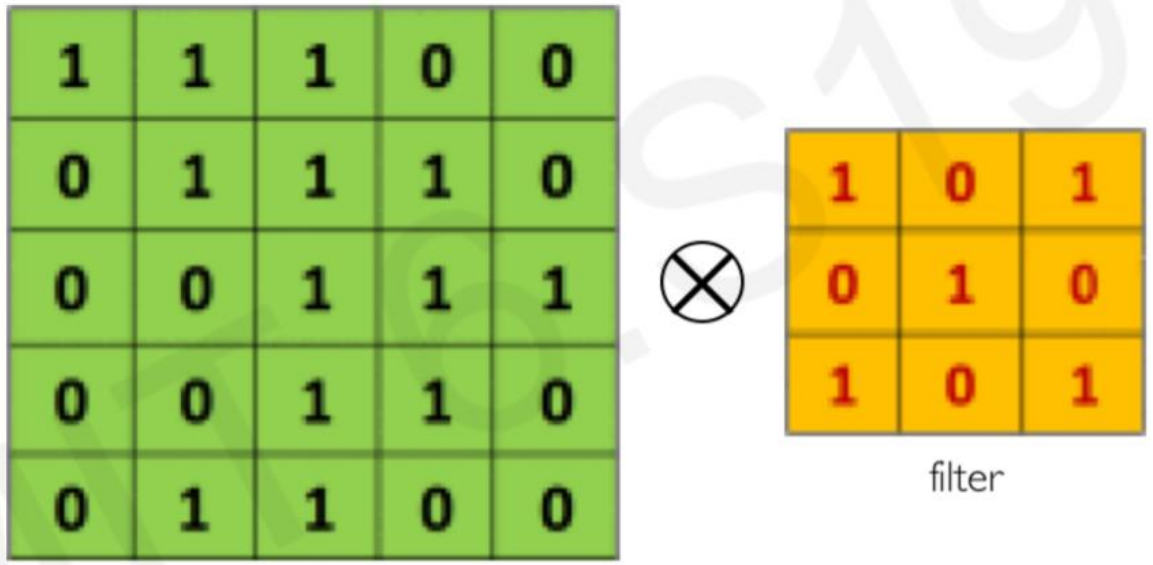
Automatic feature extraction

Can we learn hierarchy of features directly from the data?



convolutions

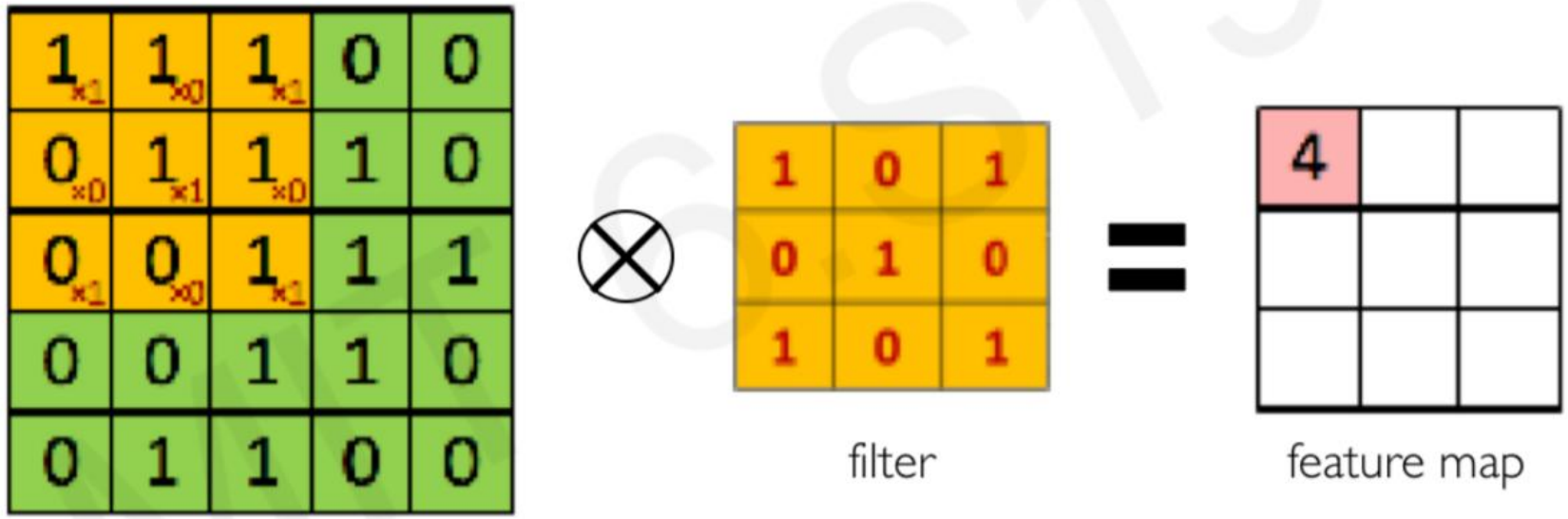
Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:



We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs…

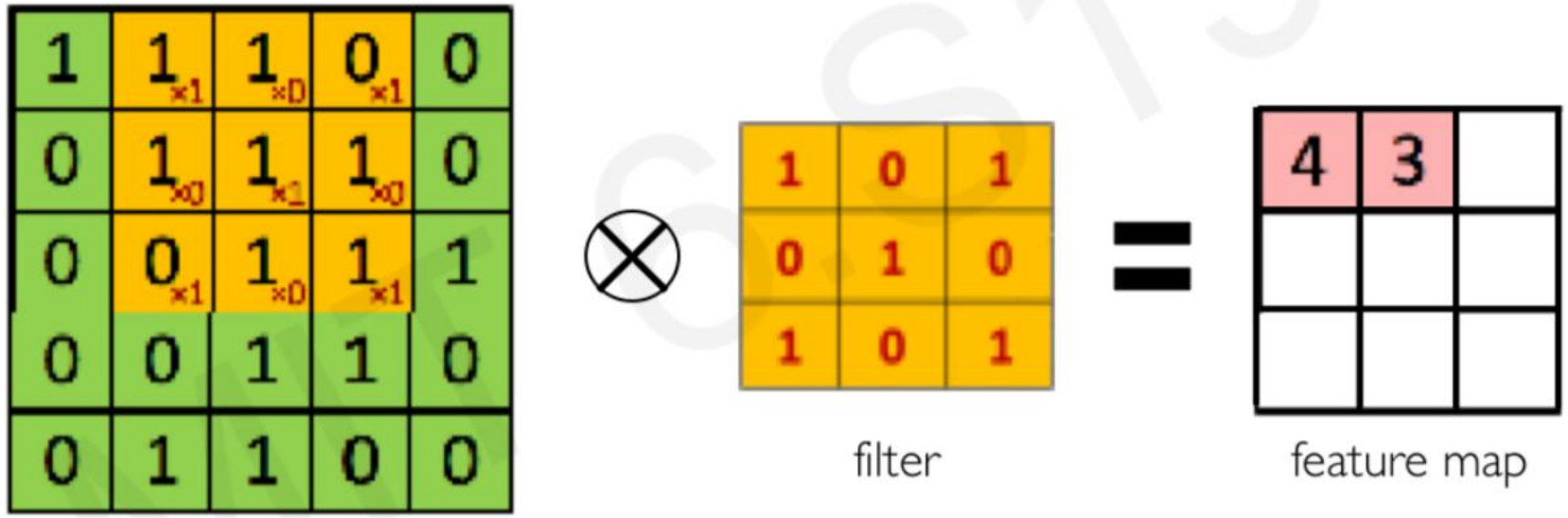
convolutions

We slide the 3x3 filter over the input image, element-wise multiply, and add the output



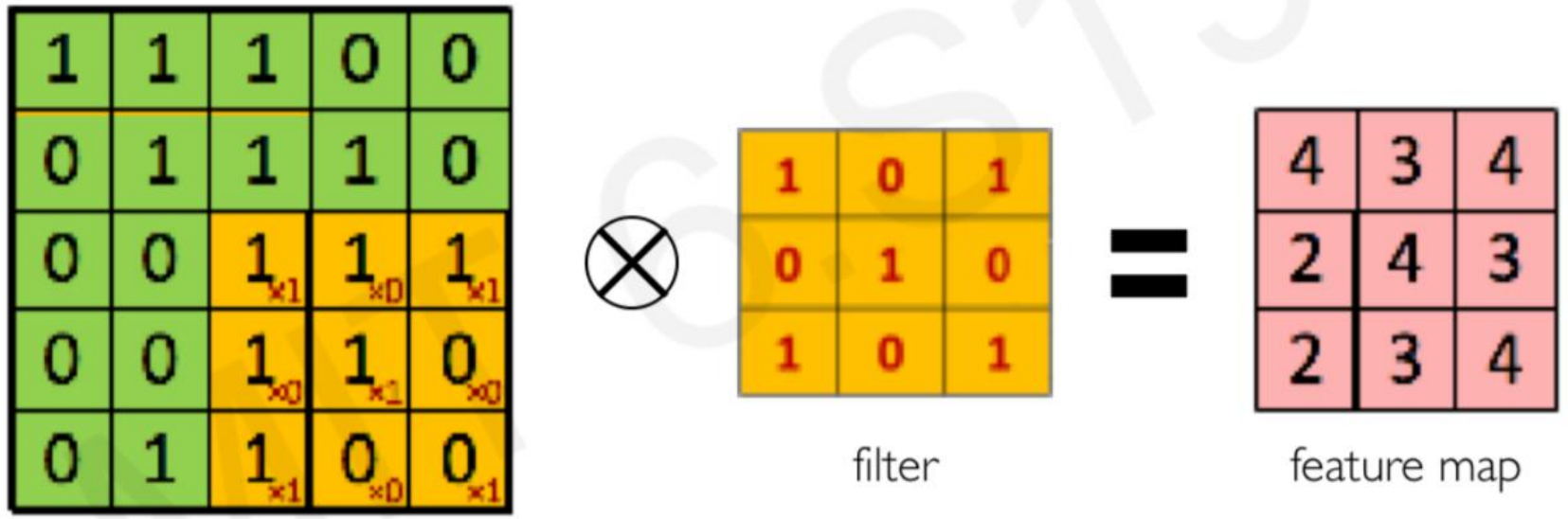
convolutions

We slide the 3x3 filter over the input image, element-wise multiply, and add the outpu



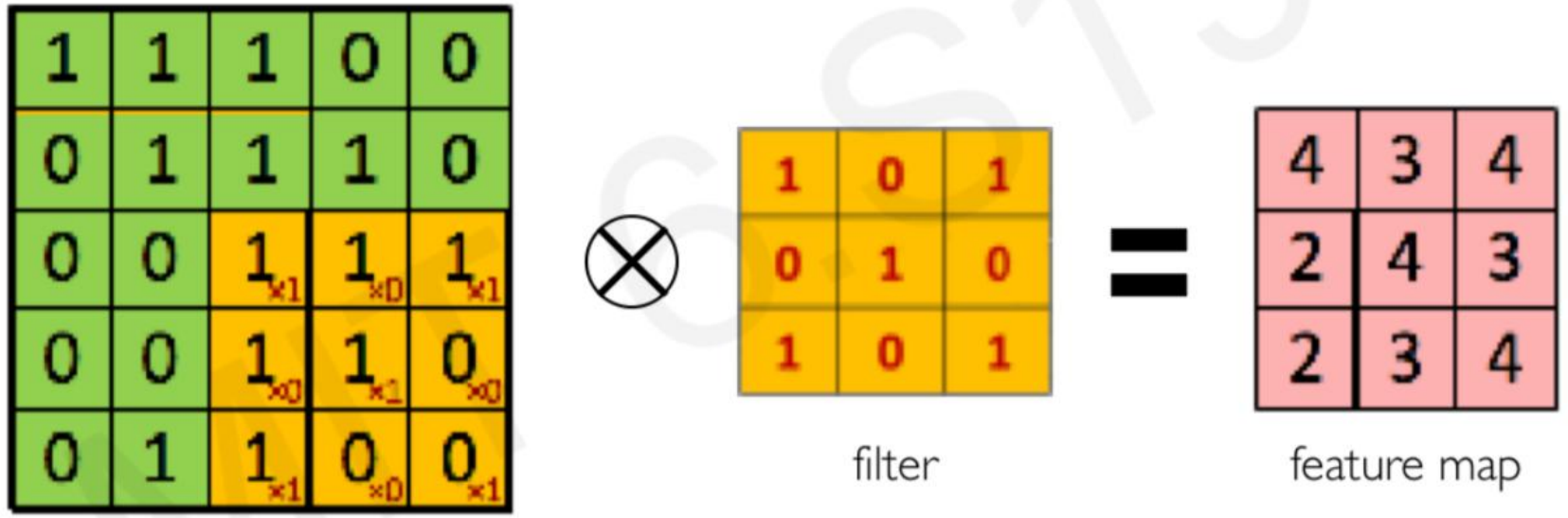
convolutions

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



convolutions

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



Convolutions produce feature maps

Original | Sharpen | Edge Detect | "Strong" Edge Detect

Convolutions large or small?

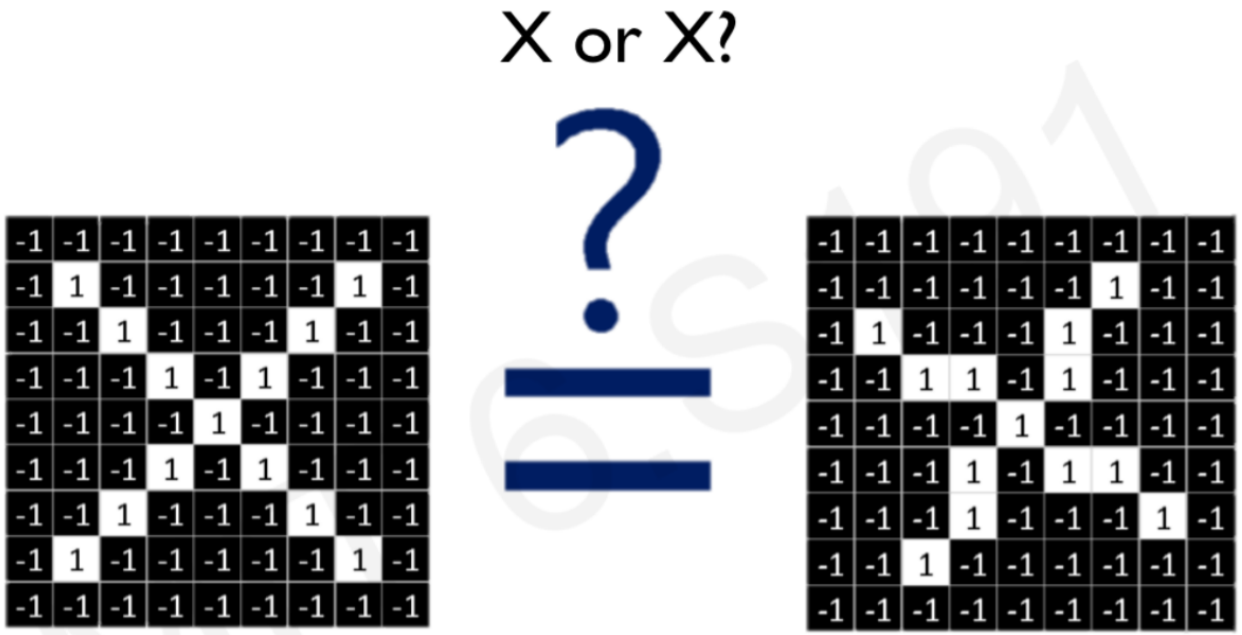
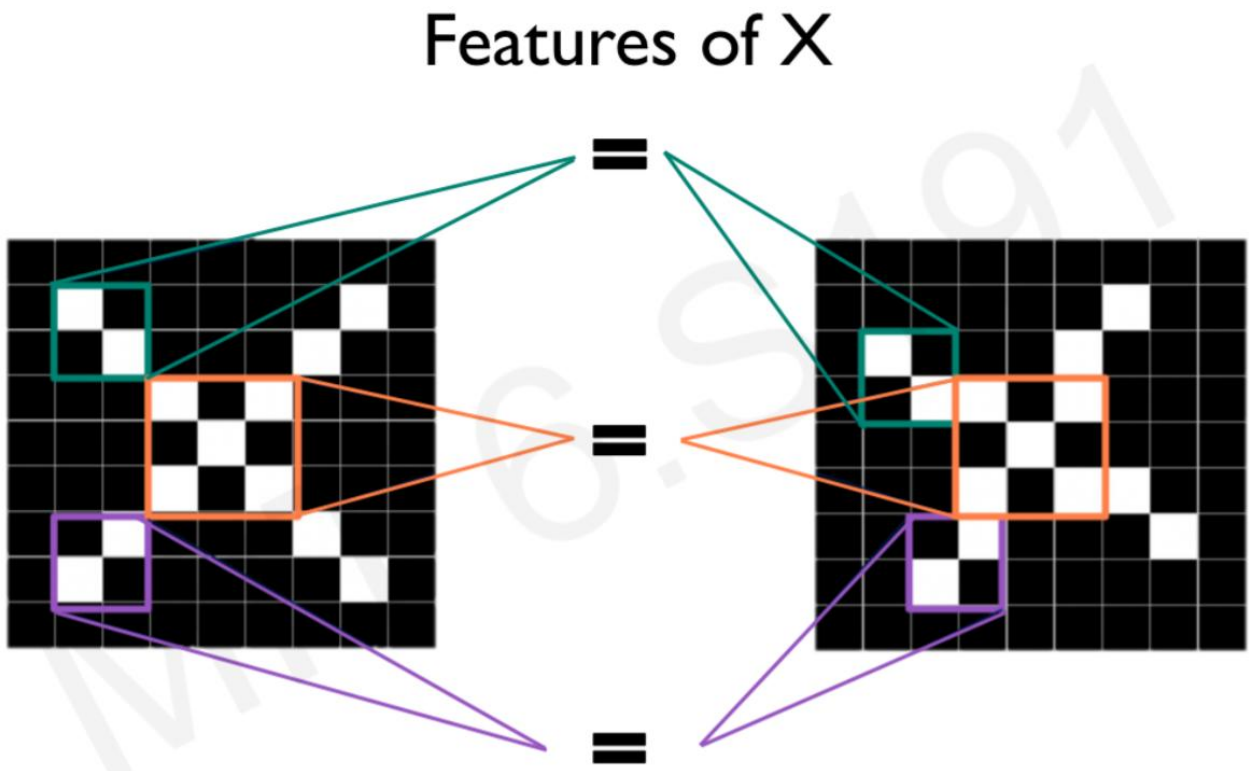


Image is represented as matrix of pixel values… and computers are literal!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

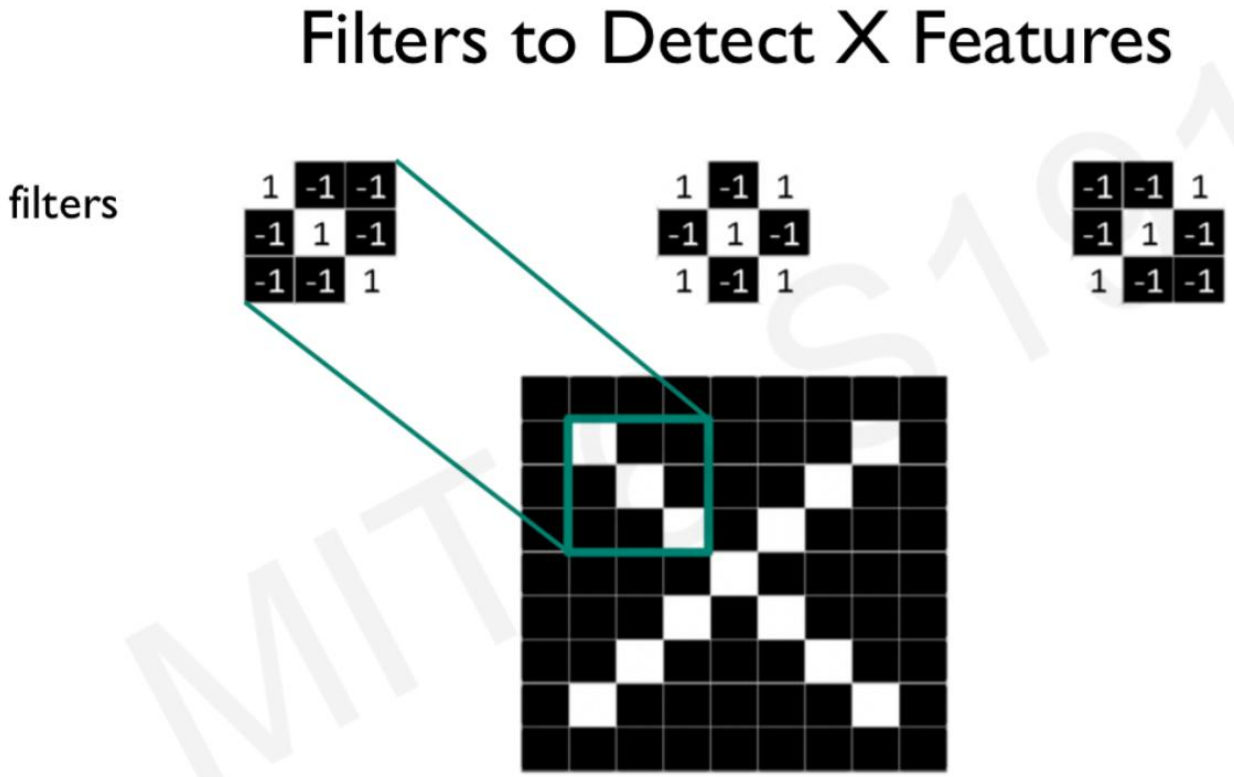
Large convolutions will often fail to capture these variations

Convolutions large or small?



Small convolutions can identify low level features

Convolutions large or small?



Convolutions on the low-level feature map produce mid-level features and so on

מקור השראה – המוח האנושי

יחידת עיבוד אלמנטרית: נוירון

100,000,000,000 נוירונים

כל נוירון מחובר לאלפי נוירונים אחרים

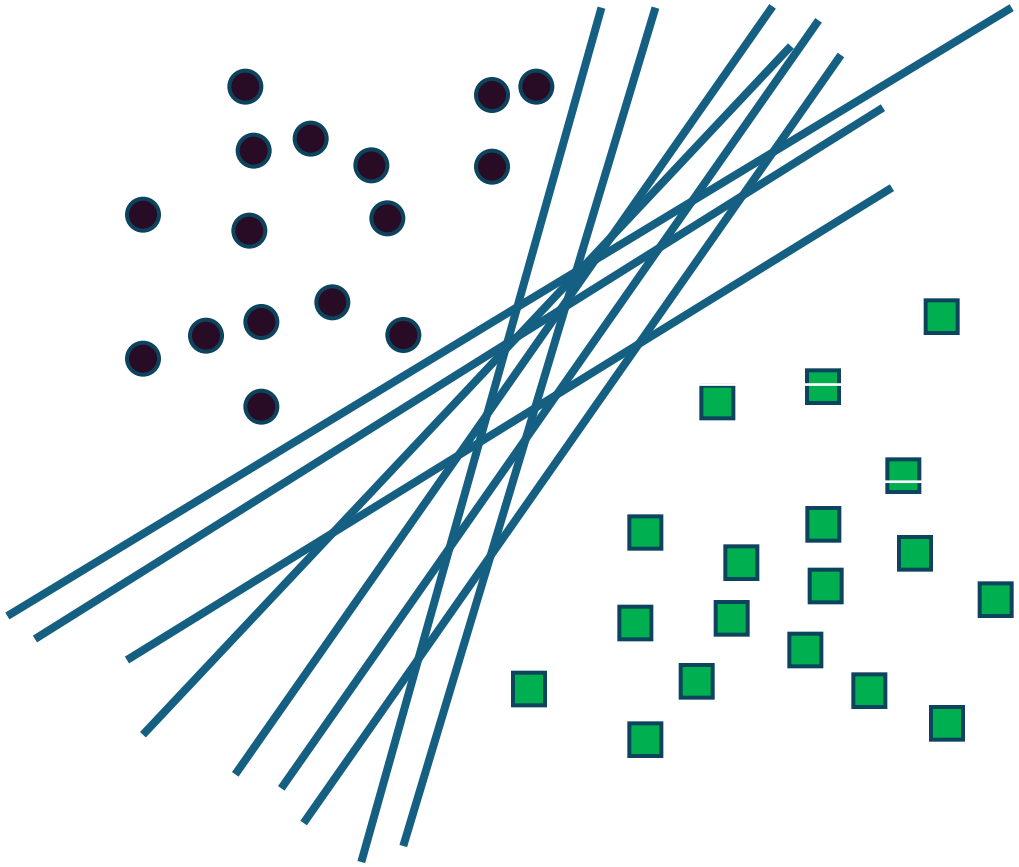
10,000 קישורים בממוצע מכל נוירון

הפלט של נוירון נקבע לפי הקלט של הנוירונים שמחוברים אליו

מדוע לחקות פעולת המח

- המטרה המרכזית של המחקר בבינה מלאכותית היא לבנות מכונה ששקולה ביכולותיה למוח האנושי (או עולה עליו)
- אולי כדאי לחקות את דרך הפעולה (הפיסית/כימית) של המוח
- המוח הוא מכונת חשיבה יעילה במיוחד שמגיעה להחלטות מורכבות תוך זמן קצר ע"י פעולה במקביל של מספר עצום של יחידות עיבוד פרימיטיביות (יחסית)

רקע: מודל דיסקרמינטיבי – מפריד לינארי

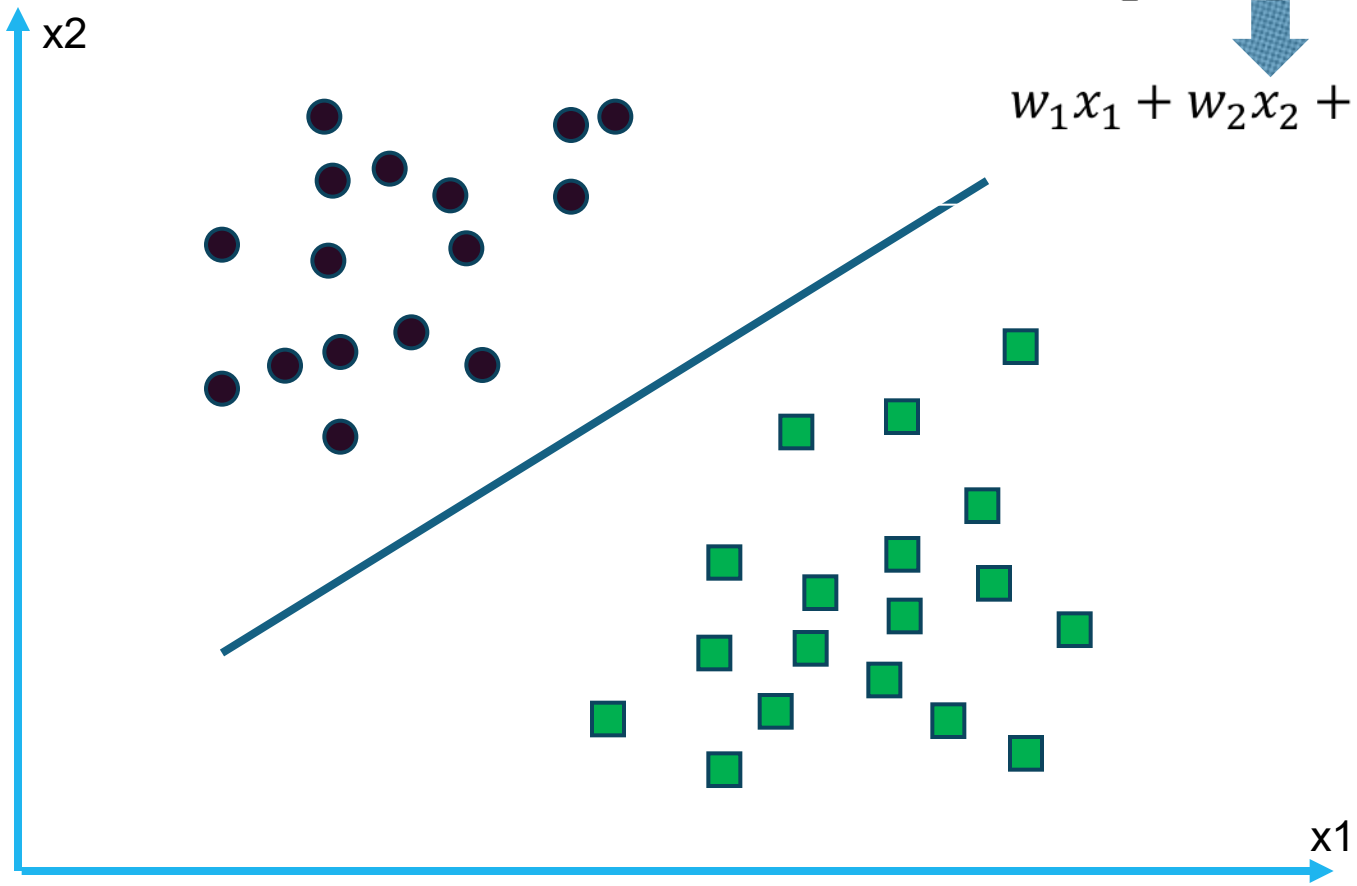


איך היינו מסווגים את
הנתונים הללו

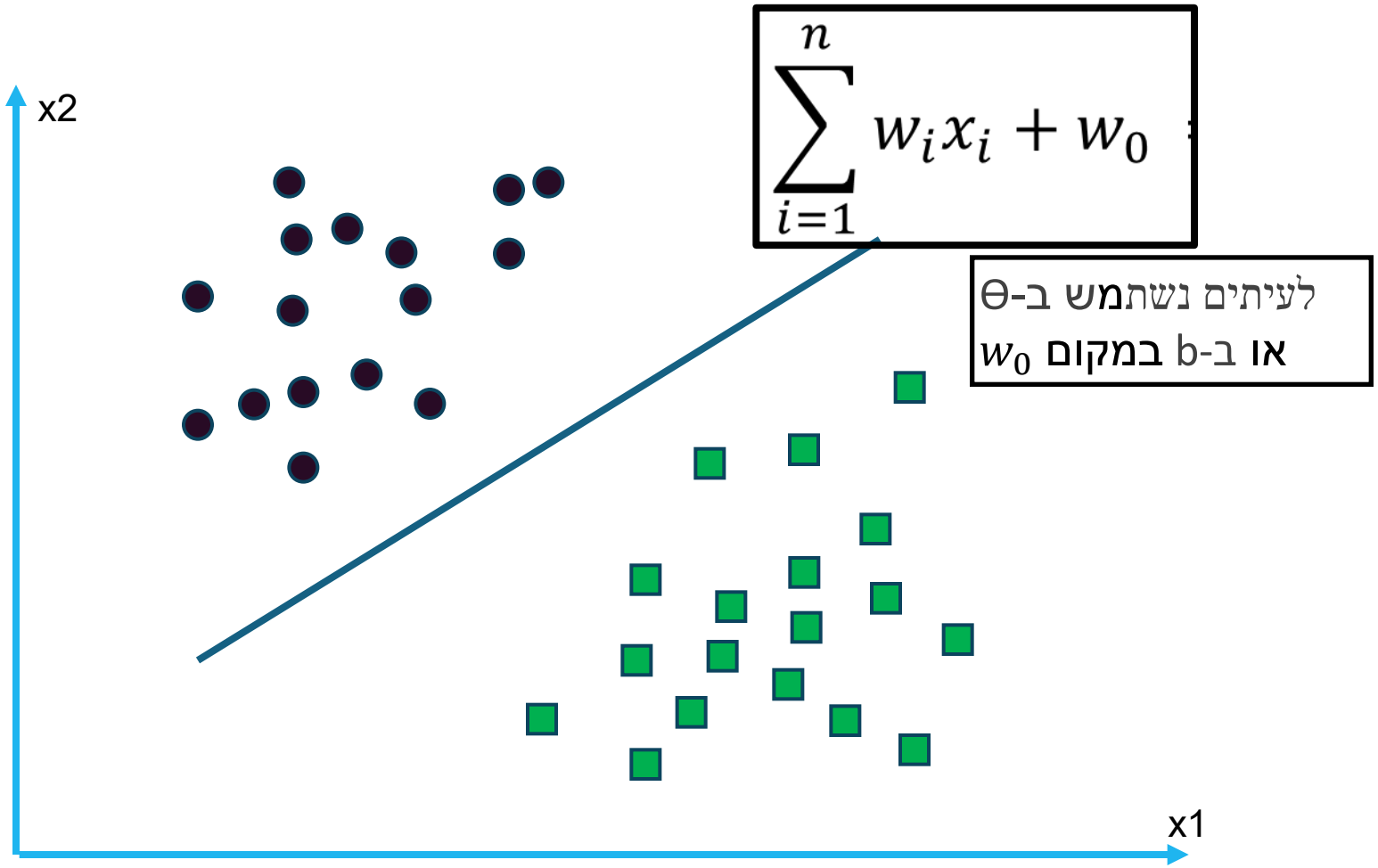
מסווג לינארי – ממשואת קו ישר לקומבינציה לינארית של ערכים ומשקולות

$$ax_1 + bx_2 + c = 0$$

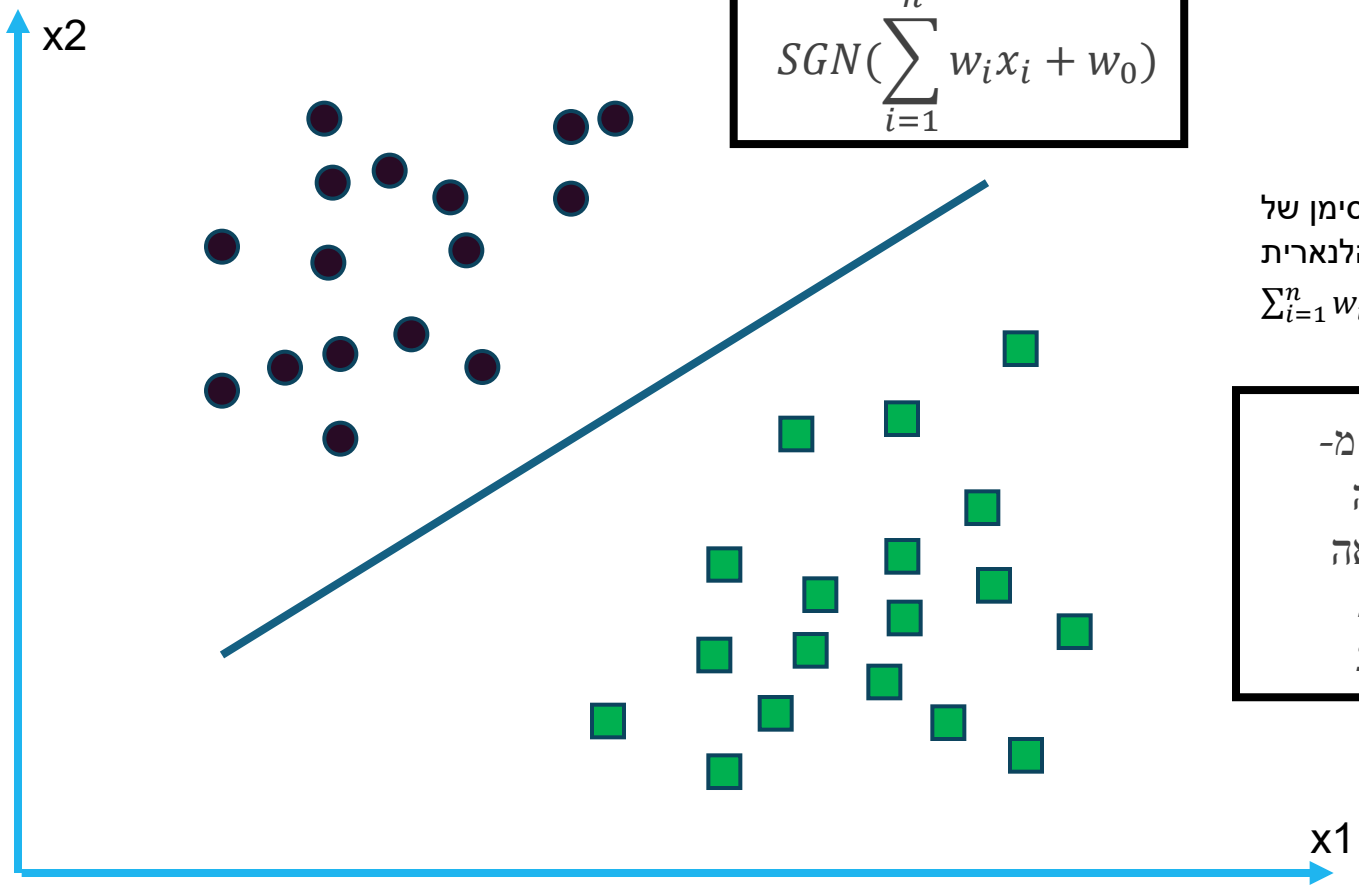
$$w_1x_1 + w_2x_2 + w_0 = 0$$



מסווג לינארי – משוואה רב מימדית



מסווג לינארי – משוואת הסיווג



$$SGN(\sum_{i=1}^{n} w_i x_i + w_0)$$

מסווגים את x, לפי הסימן של הקומבינציה הלנארית

$\sum_{i=1}^{n} w_i x_i + w_0$

אם התוצאה גדולה מ-0, נסווג קטגוריה חיובית, אם התוצאה קטנה מ-0, נסווג קטגוריה שלילית

מסווג לינארי – שאלות ביניים

<u>שאלות</u>:

2. עבור משוואה $3x_1+x_2+5=0$, מה יהיה הסיווג של זוגות הערכים הבאים (האפשרויות: חיובית/ שלילית)?

א. $x_1=-2, x_2=0$ ב. $x_1=1, x_2=-2$

<u>תשובות אפשריות</u>:

1. א – חיובי ב – חיובי
2. א – חיובי ב – שלילי
3. א – שלילי ב – חיובי
4. א – שלילי ב - שלילי

$$SGN(\sum_{i=1}^{n} w_i x_i + w_0)$$

מסווג לינארי – שאלות ביניים

<u>שאלות</u>:

2. עבור משוואה $3x_1+x_2+5=0$, מה יהיה הסיווג של זוגות הערכים הבאים (האפשרויות: חיובית/ שלילית)?

א. $x_1=-2,x_2=0$ ב. $x_1=1,x_2=-2$

<u>תשובות אפשריות</u>:

1. א – חיובי ב – חיובי
2. א – חיובי ב – שלילי
3. א – שלילי ב – חיובי
4. א – שלילי ב - שלילי

<u>תשובה נכונה</u>:

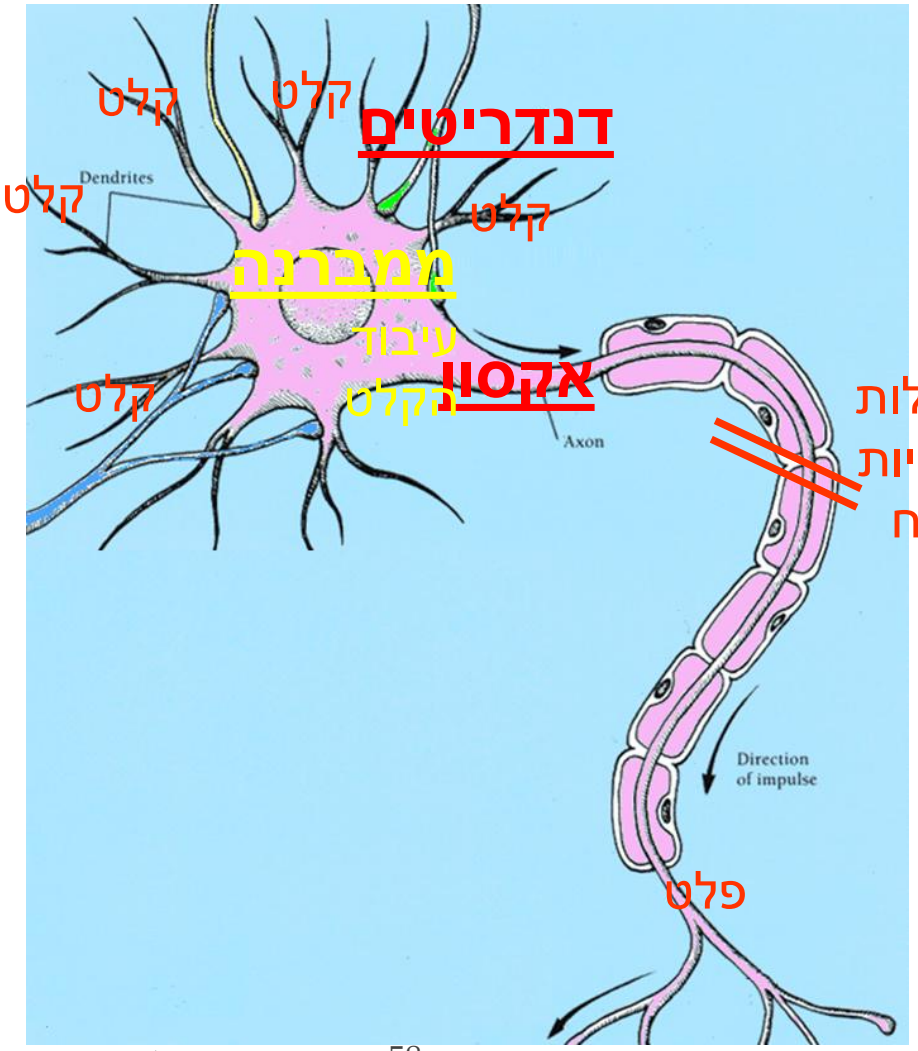
3. א. שלילי ב. חיובי

מנוירונים אנושיים לנוירונים מלאכותיים

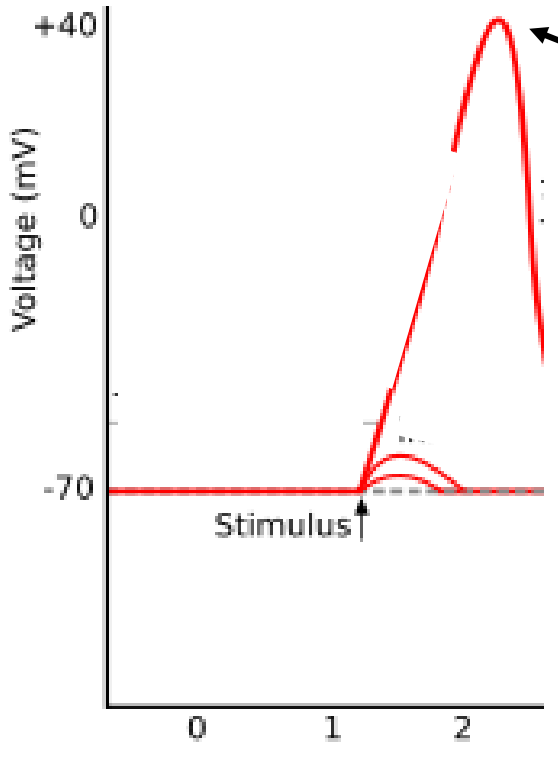
נוירון אנושי – מודל פשטני
- פוטנציאל פעולה

הנוירון המלאכותי:
- perceptron
- פונקציות הפעלה

נוירון אנושי – מודל פשטני



נוירון אנושי – מודל פשטני – פוטנציאל פעולה



פוטנציאל פעולה (אקטיבציה)

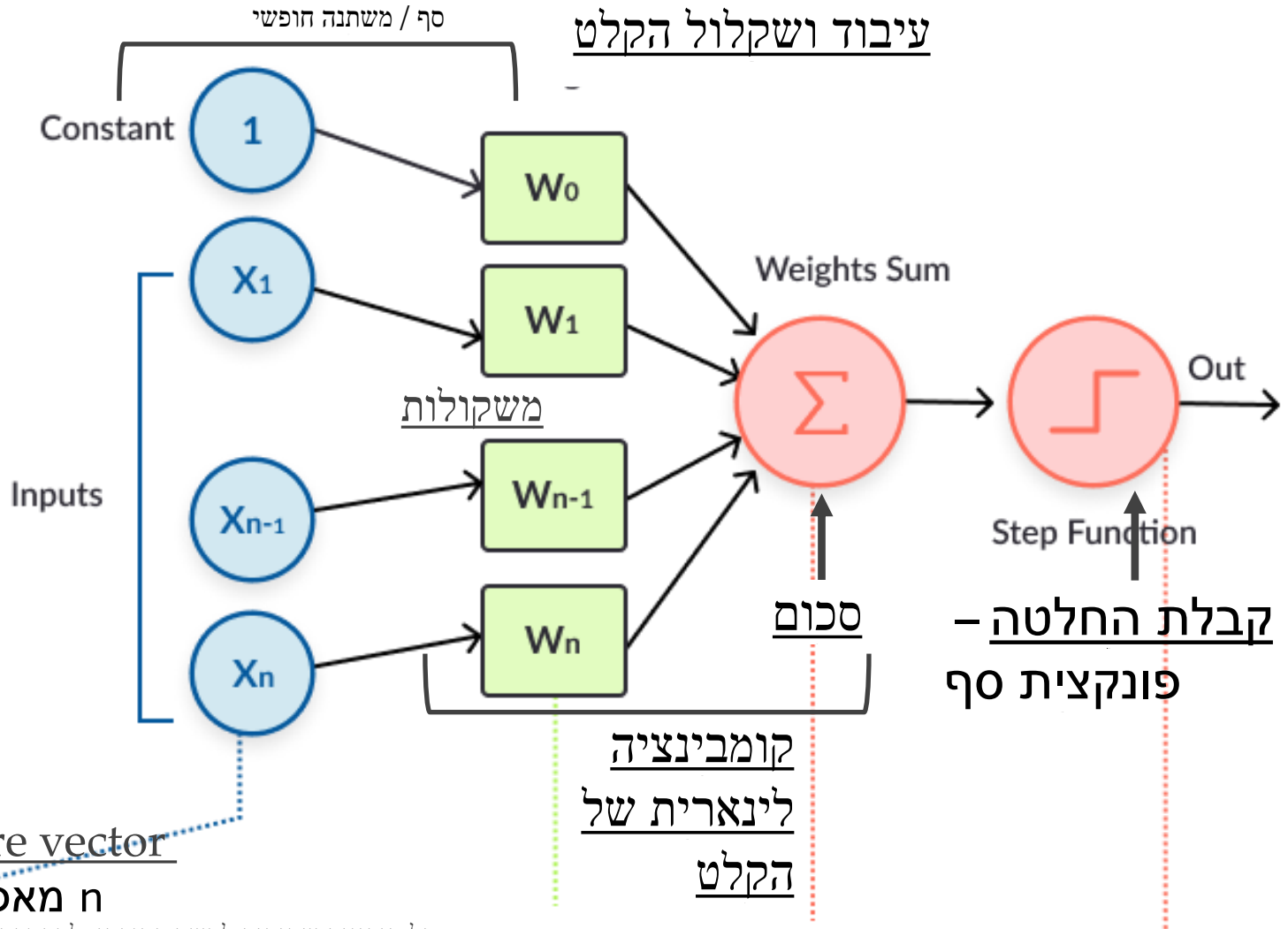
- במידה ונוצר מספיק מתח מהקלט, נוצר פוטנציאל פעולה.
- הפלט הוא יחיד (אקסון אחד)
- הפלט מהווה קלט לנוירונים אחרים

*Perceptron – הנוירון המלאכותי – מודל א'

- היחידה הבסיסית ברשת נוירונים
- יכולה לשמש כמערכת לומדת גם באופן עצמאי
- לפרספטרון יש n קלטים ופלט אחד
- הפרספטרון מחשב קומבינציה לינארית של הקלטים, ופולט 1 אם היא גדולה מסף נתון, 0 (או לעיתים 1-) אחרת

הערה:
* בעברית נקרא ל-perceptron פשוט פרספטרון או פרצפטרון או נוירון מלאכותי

Perceptron – הנוירון המלאכותי – מודל א'



Perceptron - הסבר

נתונים n קלטים: $x_1, \ldots, x_n$

(בבעית סיווג זהו ה-feature vector שלנו ואלו ערכי המאפיינים)

$$o(\langle x_1, \ldots, x_n \rangle) = \begin{cases} 1 & if\ w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n > 0 \\ -1 & otherwise \end{cases}$$

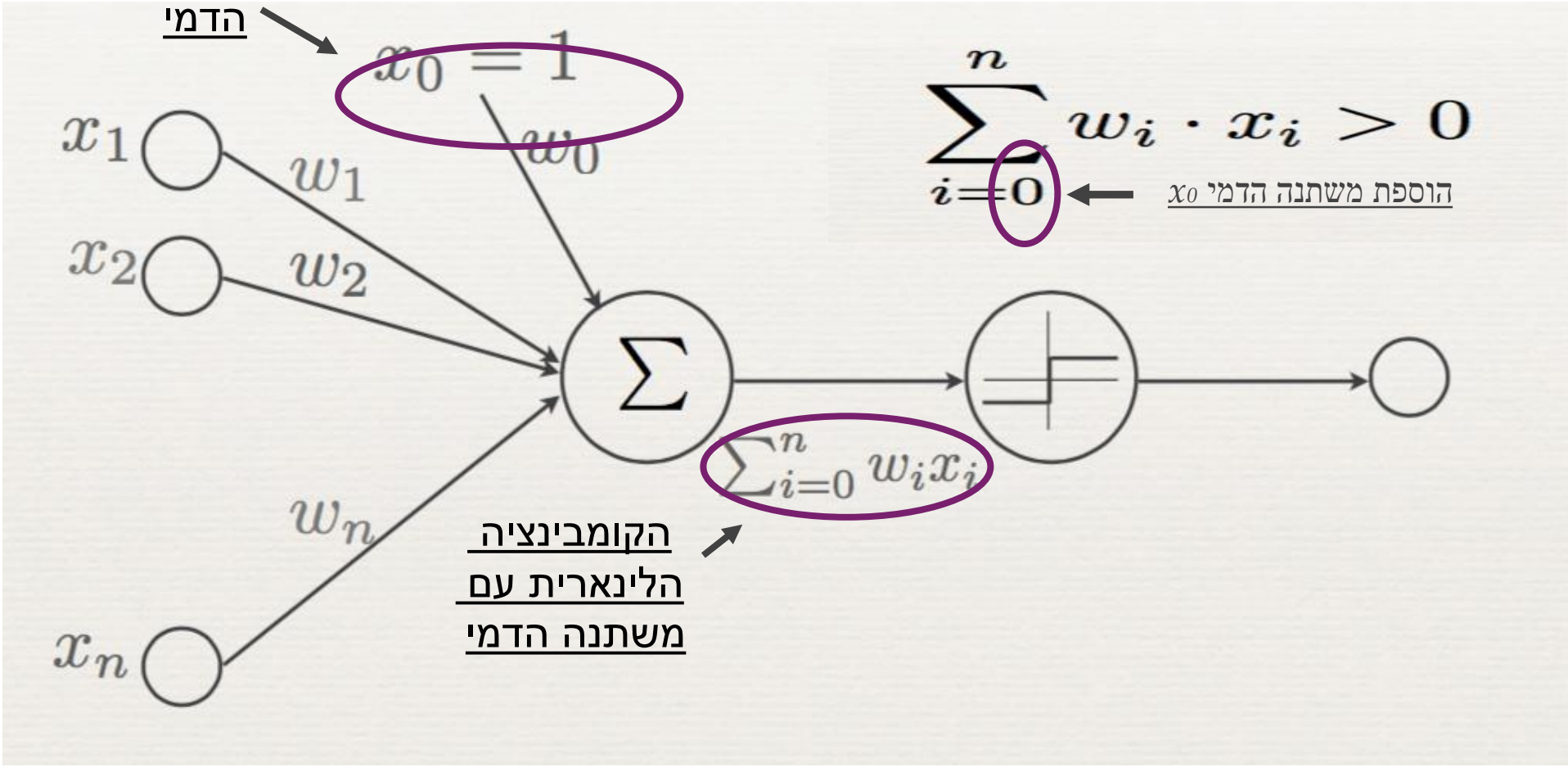
מכיוון שניתן לכתוב את התנאי כך:

$$w_1 \cdot x_1 + \cdots + w_n \cdot x_n > -w_0$$

$-w_0$ מהווה בעצם סף שקובע את תוצאת הסיווג

Perceptron – הסבר – הוספת משתנה דמי

כדי לפשט את הביטוי, מוסיפים משתנה דמי ומקבלים: $x_0 = 1$



Perceptron – הסבר – הוספת משתנה דמי – צורה מקובלת לכתיב מתמטי

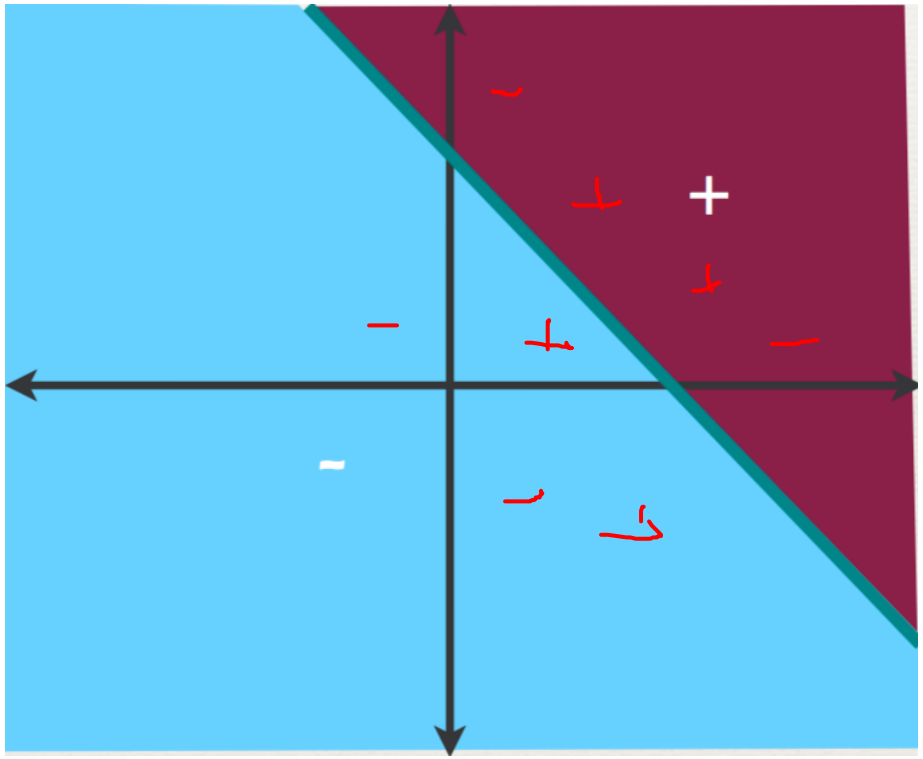
- בהינתן ווקטור לסיווג (x) ובהינתן סט משקולות (w)

$$(w_0 \quad w_1 \quad w_2 \quad \dots \quad \dots \quad w_d)\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ \\ x_d \end{pmatrix} = \sum_{i=0}^{d} w_i x_i = W^T x$$

- נסווג את הווקטור עפ"י תוצאת המכפלה הפנימית בין X ו-W והשוואה לסף.

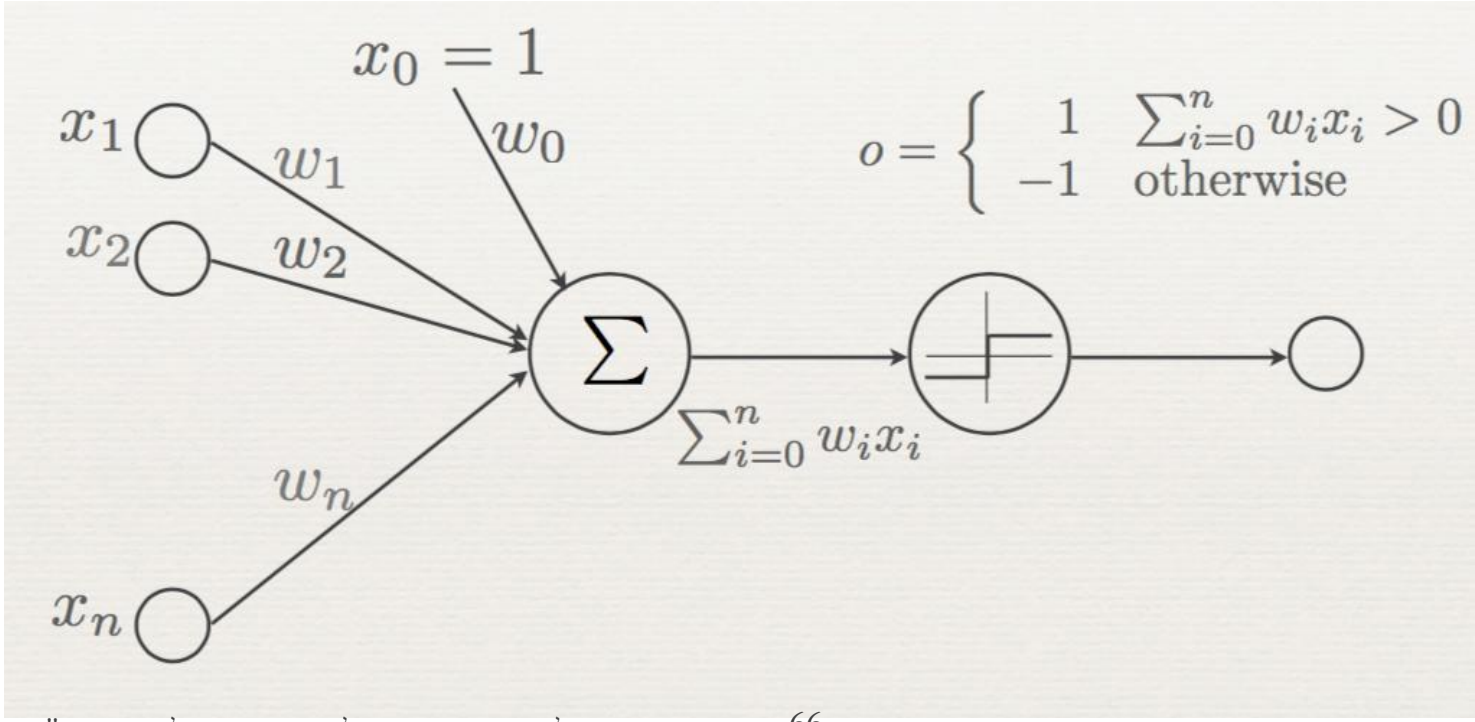
Perceptron – הסבר – פלט

- הפרצפטרון הוא למעשה מפריד ליניארי – מישור שמפריד את המרחב לשני חלקים – חלק עבור כל סיווג



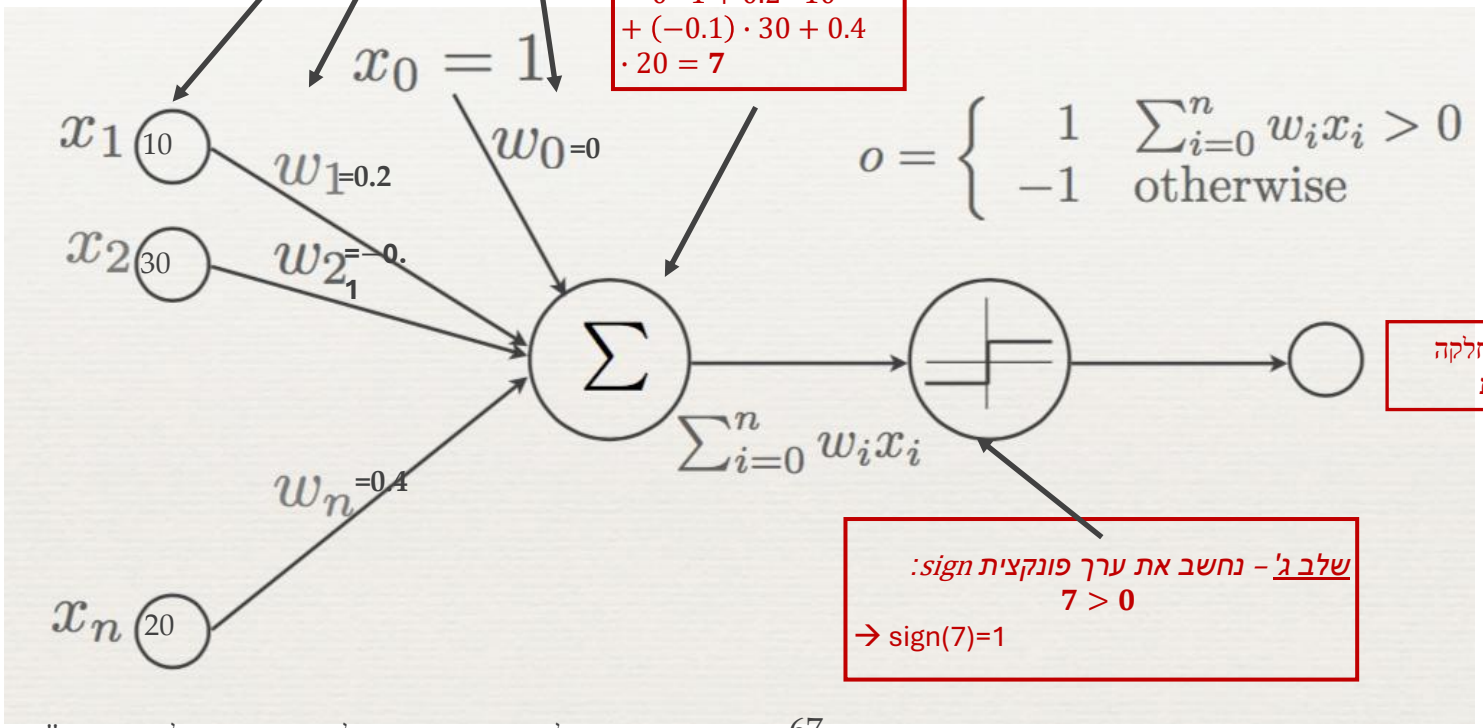
דוגמה לקריאת נוירון בודד עם פונקצית sign

נתונה יחידת הנוירון הבאה, כאשר מישור ההפרדה מיוצג ע"י:
$0.2 \cdot x_1 - 0.1 \cdot x_2 + 0.4 \cdot x_3 = 0$
בצעו סימולציה לקבלת הפלט של הנוירון, תוך שימוש בגרף של נוירון ה-sign, עבור הוקטור: (10,30,20)



דוגמה לקריאת נוירון בודד עם פונקצית sign

נתונה יחידת הנוירון הבאה, כאשר מישור ההפרדה מיוצג ע"י:
$0.2 \cdot x_1 - 0.1 \cdot x_2 + 0.4 \cdot x_3 = 0$
בצעו סימולציה לקבלת הפלט של הנוירון, תוך שימוש בגרף של נוירון ה-sign, עבור הוקטור: (10,30,20)



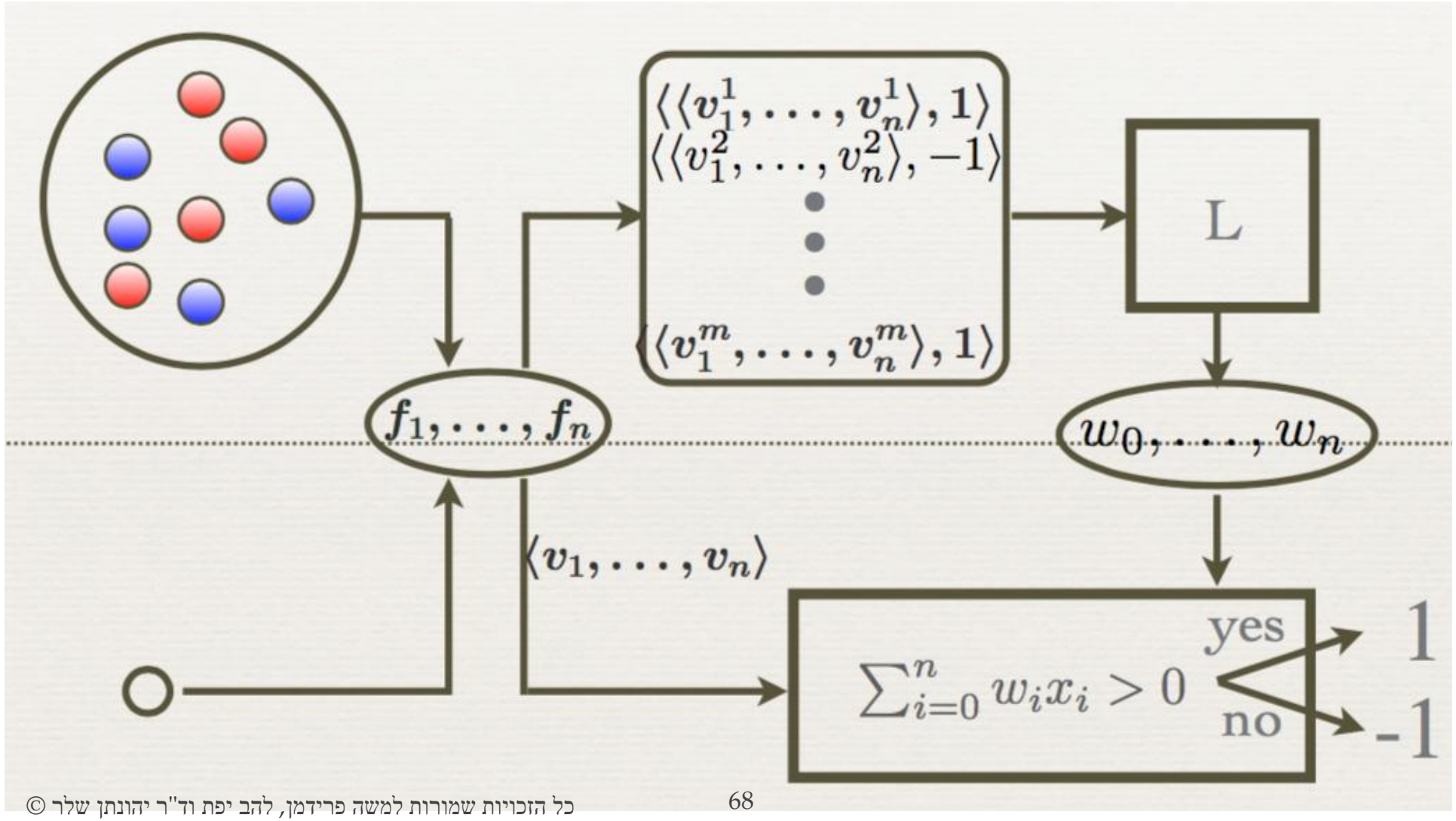
שלב א' – נציב את ערכי המשקולות וערכי ה-feature vector

שלב ב' - נכפיל ונסכום:
$net = \sum_{i=0}^{3} w_i x_i$
$= 0 \cdot 1 + 0.2 \cdot 10 + (-0.1) \cdot 30 + 0.4 \cdot 20 = 7$

שלב ג' – נחשב את ערך פונקצית sign:
$7 > 0$
→ sign(7)=1

הסיווג= המחלקה החיובית

אימון פרצפטרון - תהליך הלמידה



אימון פרצפטרון –אלגוריתם הלמידה

- מעבד דוגמא אחרי דוגמא
- דוגמא שסימונה שונה מזה של ההיפותזה הנוכחית, גורמת לעדכון המקדמים

אם הסכום קטן מידי –
מגדילים את המקדמים של
התכונות עם ערך **חיובי**,
מקטינים כאלה עם ערך **שלילי**

אם הסכום גדול מידי –
מגדילים את המקדמים של
התכונות עם ערך **שלילי**,
מקטינים כאלה עם ערך **חיובי**

אימון פרצפטרון – עדכון המקדמים

- כלל העדכון של המשקולות:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Δw_i - עדכון ל-w_i (הוספה או הורדה ביחס לאיטרציה הקודמת)

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה perceptron עבור הדוגמא הנוכחית

אימון פרצפטרון – תוצאות אפשריות

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

- Correct Output *(t=o)*
  - Weights are unchanged
- Incorrect Output *(t≠o)*
  - Change weights !
- False Negative *(t=1 and o=-1)*
  - Add to w
- False Positive *(t=-1 and o=1)*
  - Subtract from w

אימון פרצפטרון – עדכון המקדמים - הסבר

$$\Delta w_i = \eta(t-o)x_i$$

η קבוע הלמידה

t ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה perceptron

$$t = o \Rightarrow \Delta w_i = 0$$

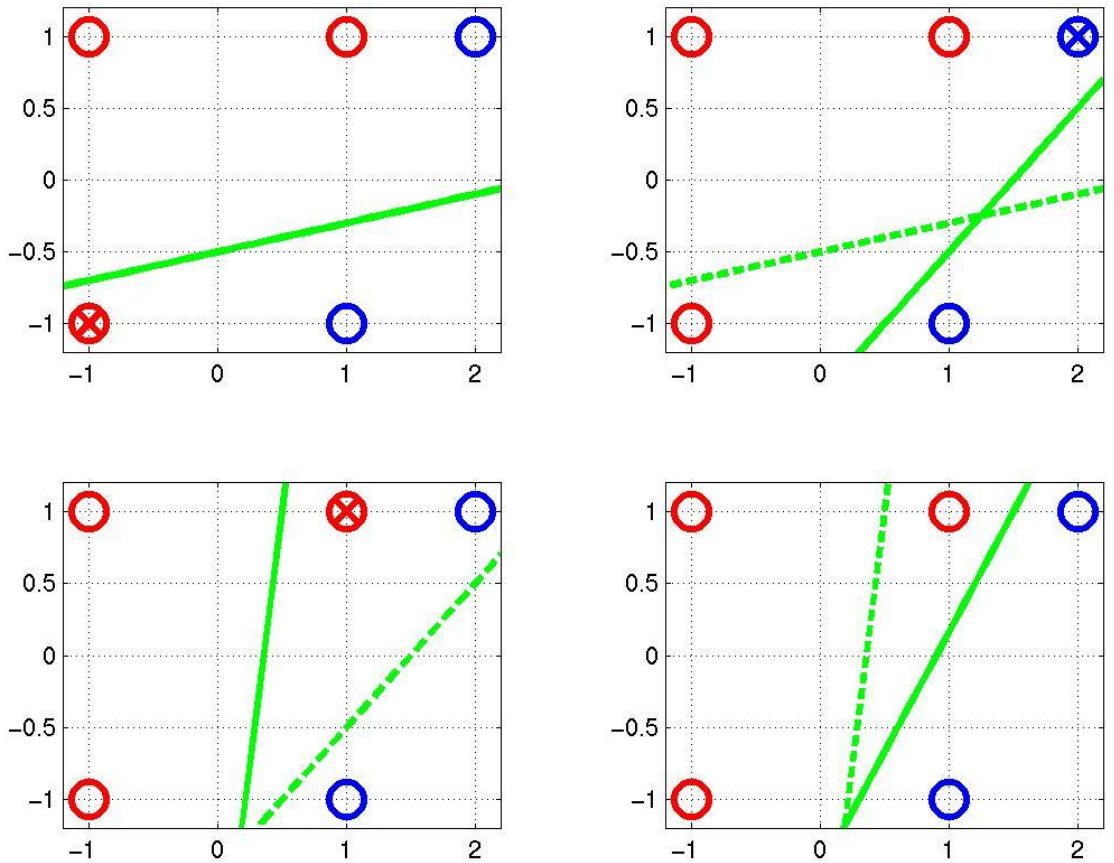
$$t = 1 \wedge o = -1 \wedge x_i > 0 \Rightarrow \Delta w_i > 0$$

$$t = 1 \wedge o = -1 \wedge x_i < 0 \Rightarrow \Delta w_i < 0$$

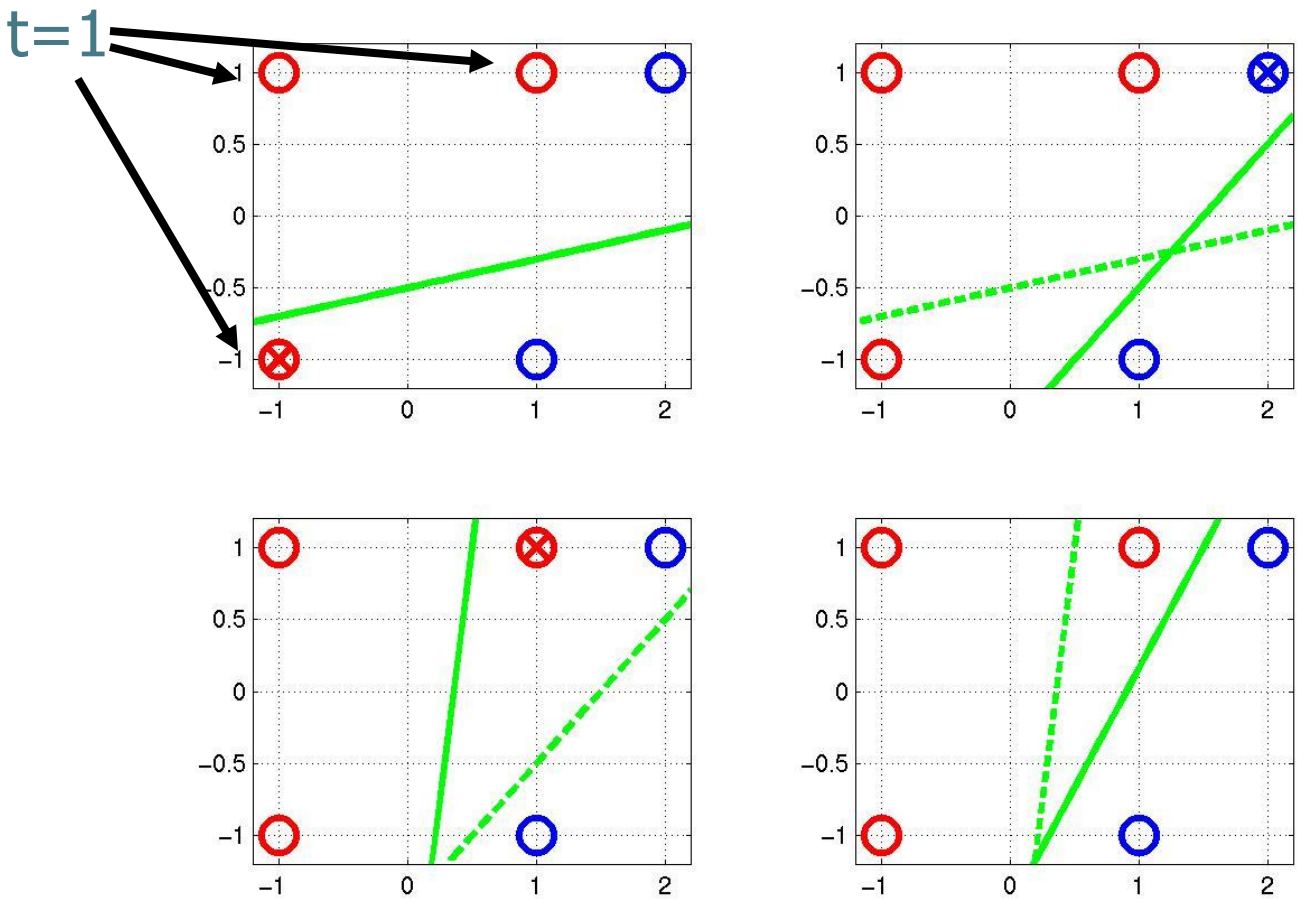
$$t = -1 \wedge o = 1 \wedge x_i < 0 \Rightarrow \Delta w_i > 0$$

$$t = -1 \wedge o = 1 \wedge x_i > 0 \Rightarrow \Delta w_i < 0$$

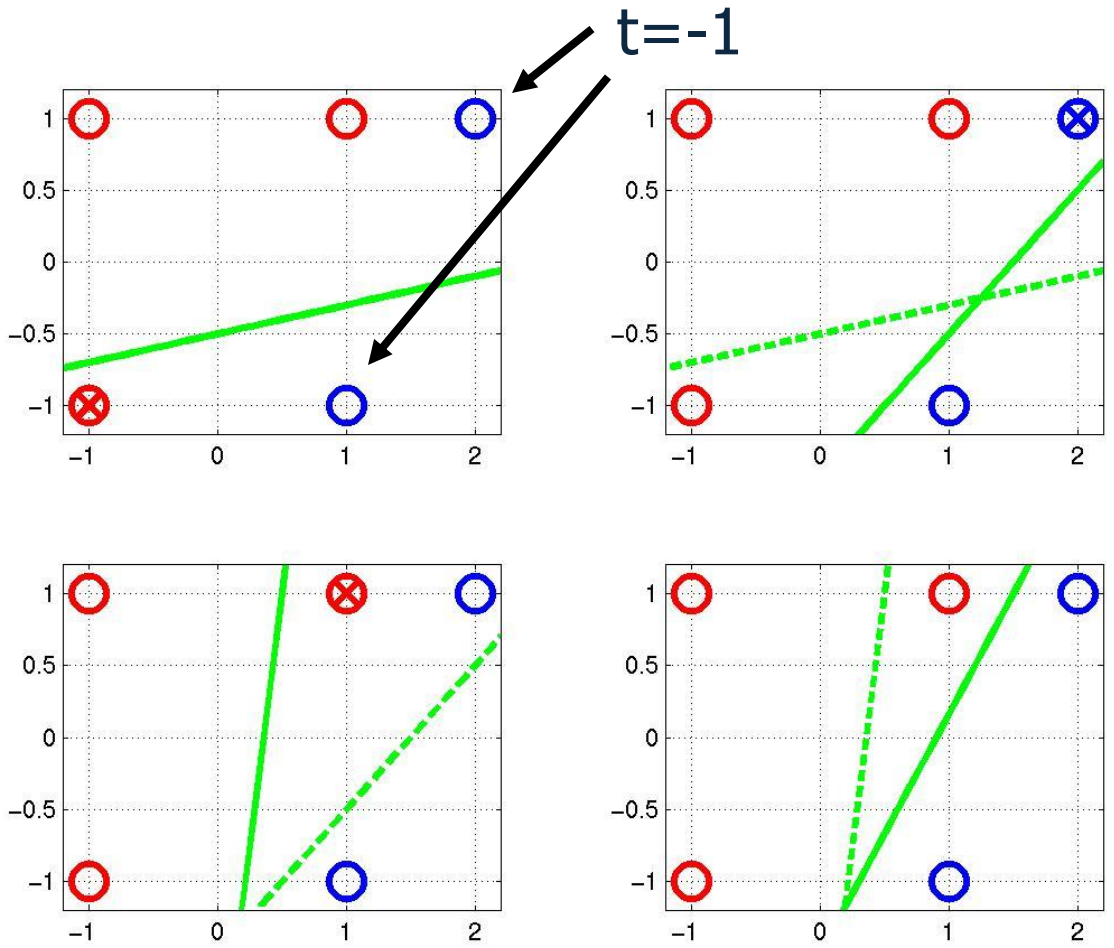
Perceptron Learning Rule (SGD)



Perceptron Learning Rule (SGD)

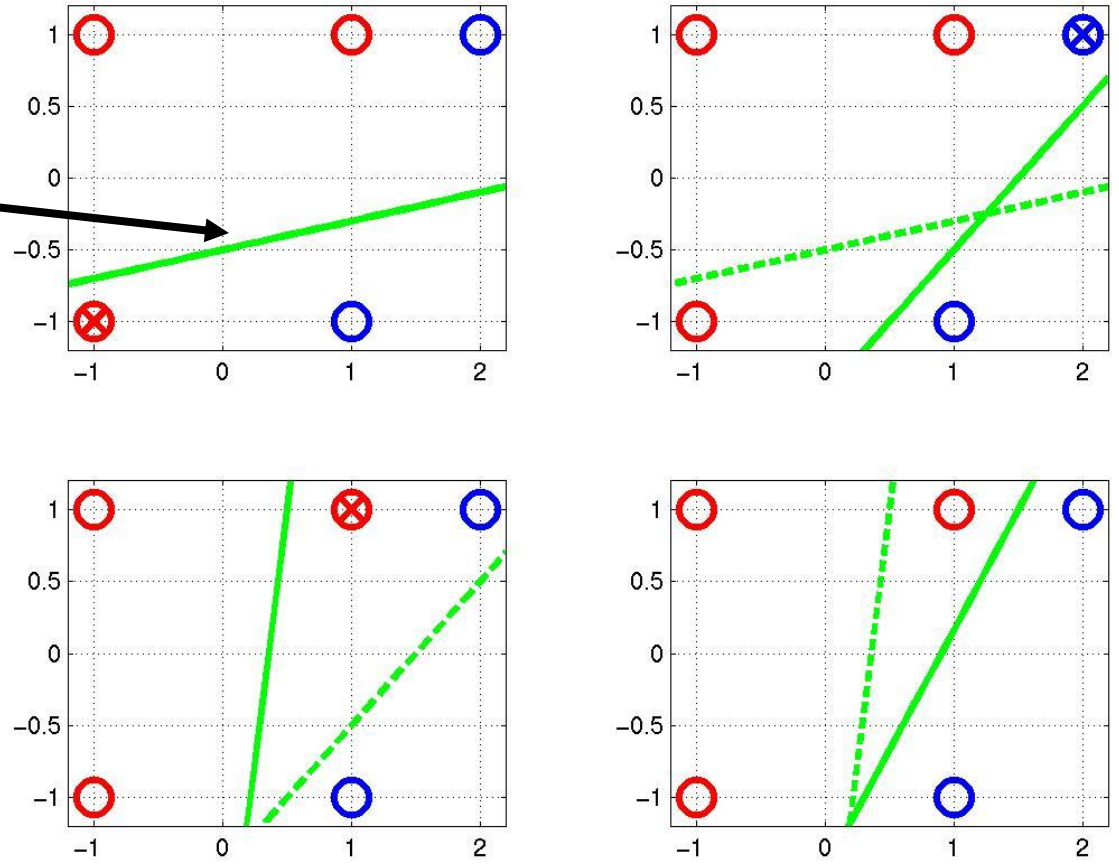


Perceptron Learning Rule (SGD)



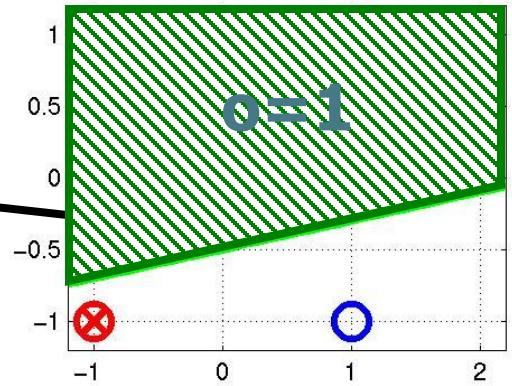
Perceptron Learning Rule (SGD)

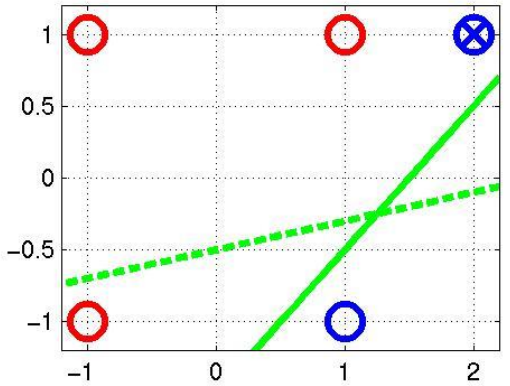
w=[0.25 –0.1 0.5]

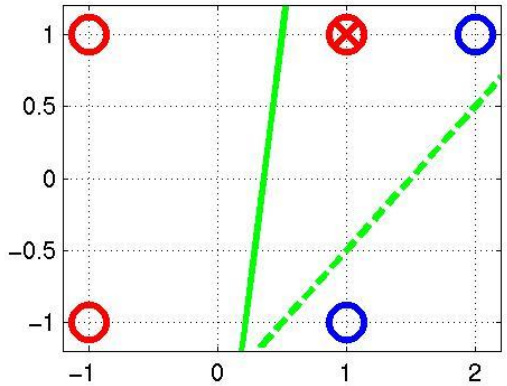


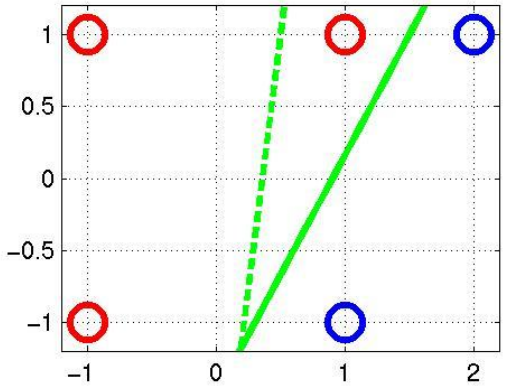
Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]



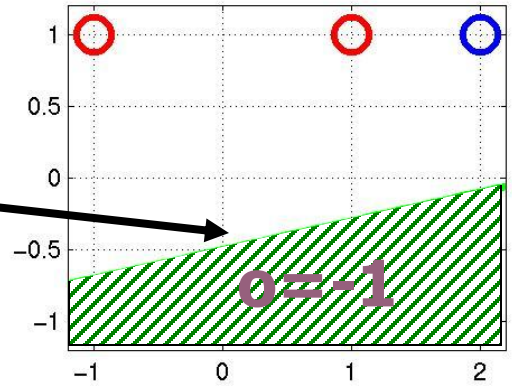


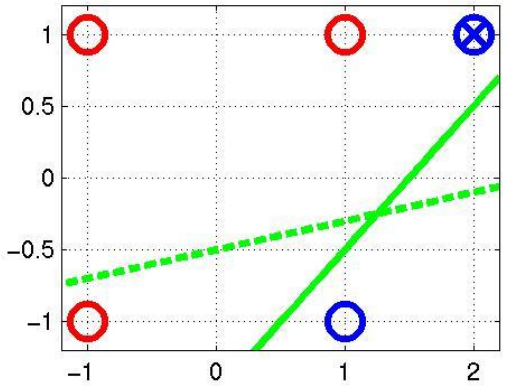


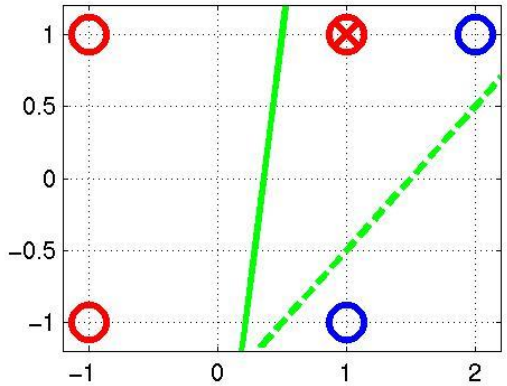


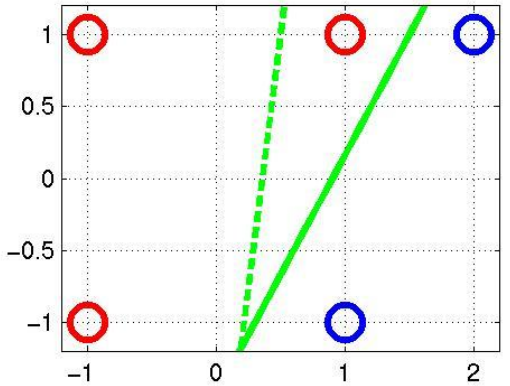
Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]









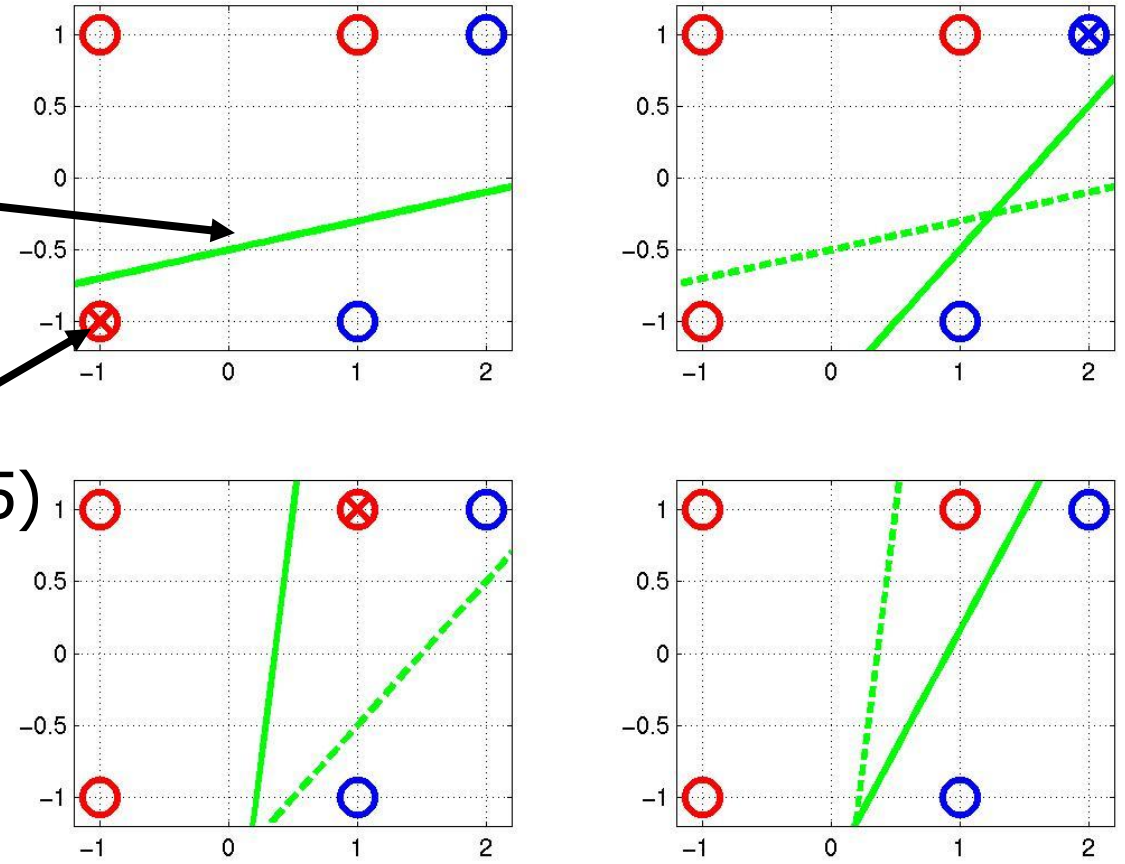
Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]

(x,t)=([-1,-1],1)

o=sgn(0.25+0.1-0.5)

=-1



Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]

t=1, o=-1

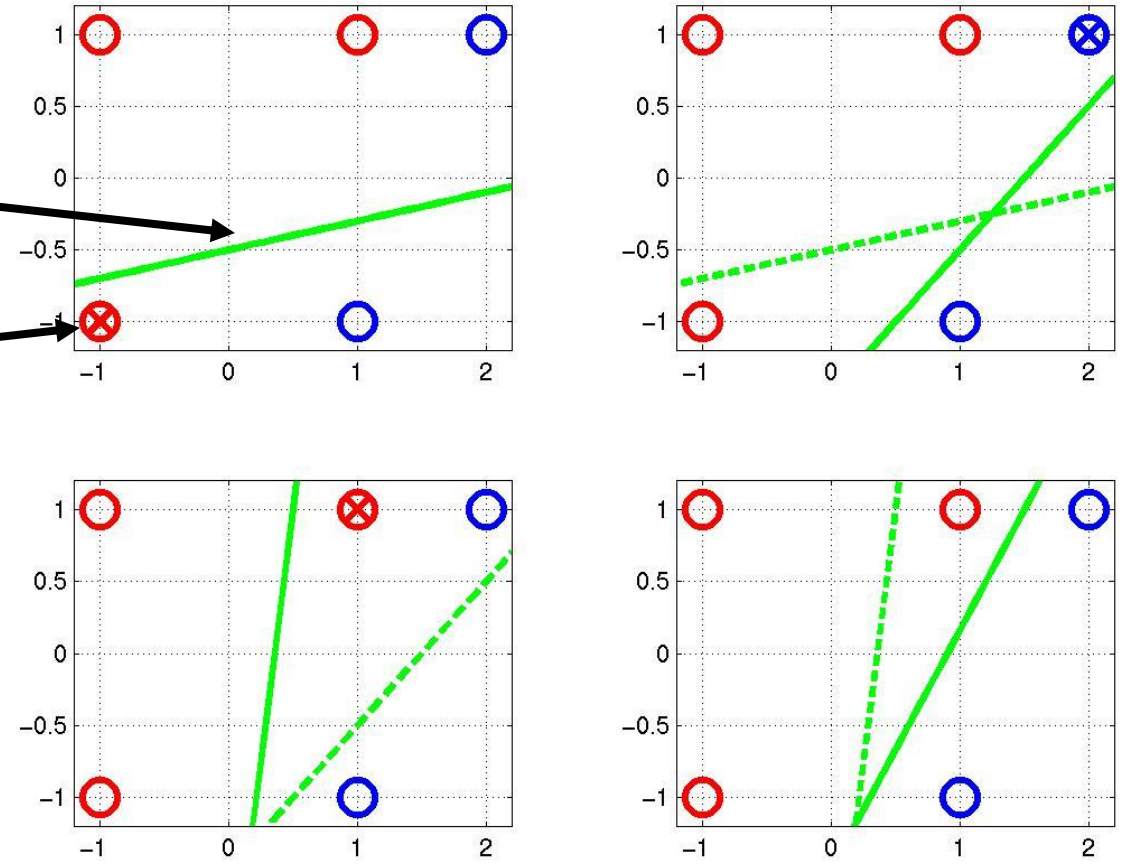
→ t-o=2

η=0.1 →

$\Delta w = \eta(t-o)\, x_i$ →

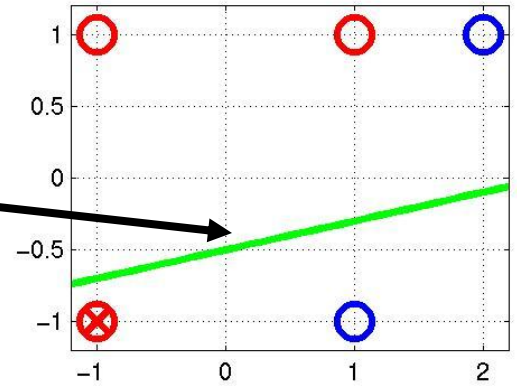
$\Delta w = 0.1*2*[1\ -1\ -1] =$

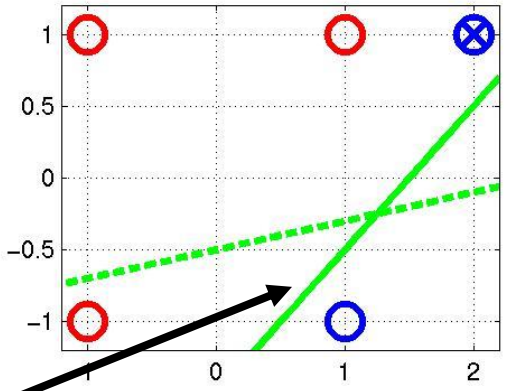
$[0.2\ -0.2\ -0.2]$



Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]



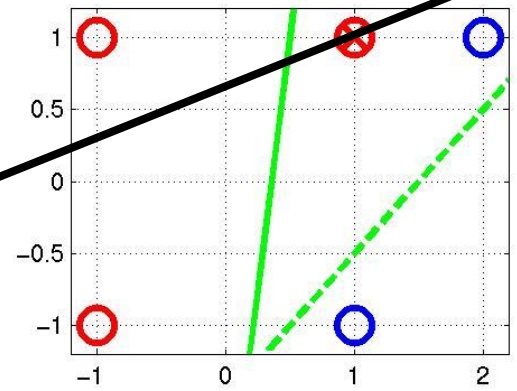


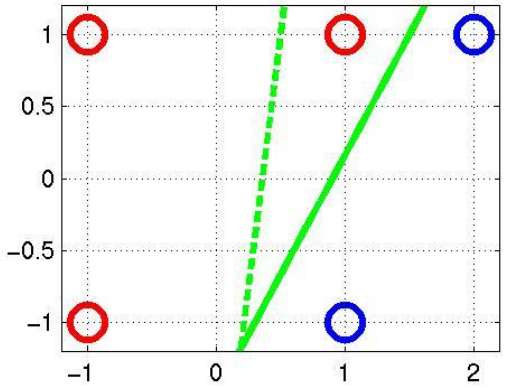
Δw=[0.2 -0.2 -0.2]

w=w+ Δw

→

w=[0.45 –0.3 0.3]



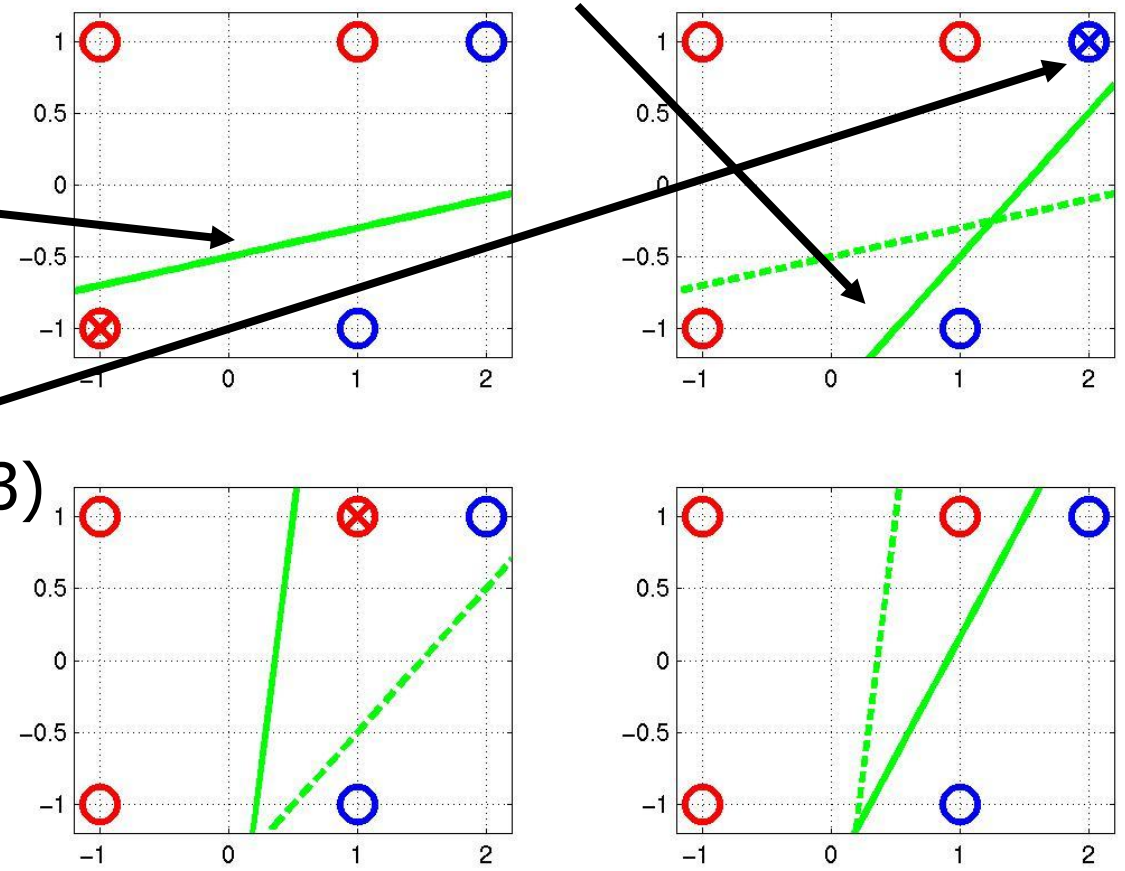


Perceptron Learning Rule (SGD)

w=[0.45 –0.3 0.3]

w=[0.25 –0.1 0.5]

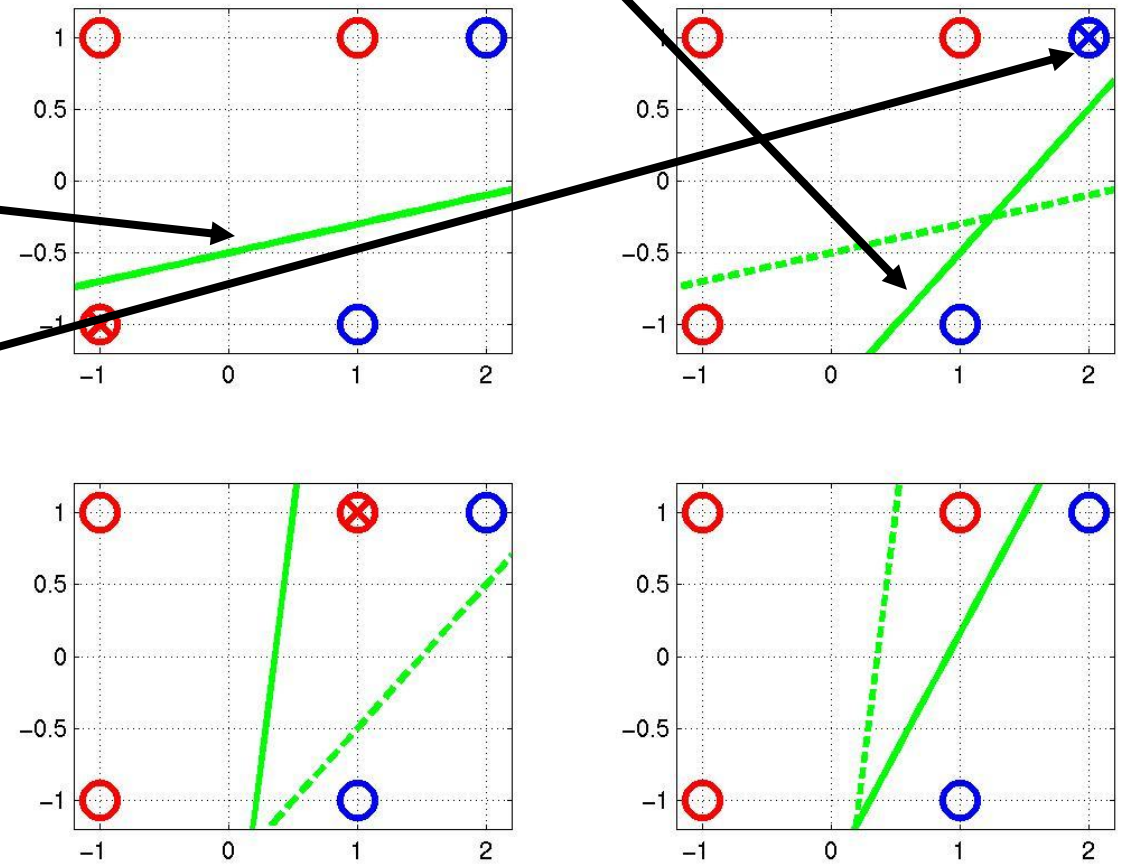
(x,t)=([2,1],-1)
o=sgn(0.45-0.6+0.3)
=1



Perceptron Learning Rule (SGD)

w=[0.45 –0.3 0.3]

w=[0.25 –0.1 0.5]

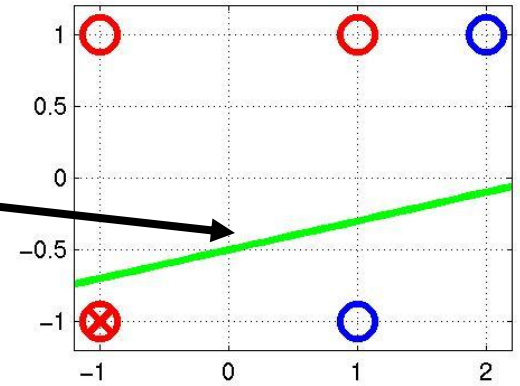


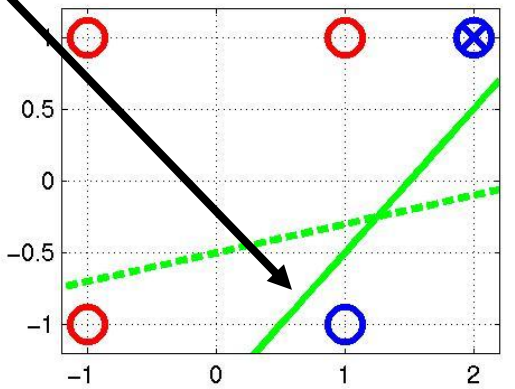
t=-1, o=1
→ t-o=-2
η=0.1 →
$\Delta w = \eta(t-o)\, x_i$ →
Δw=0.1*-2*[1 2 1]=
[-0.2 -0.4 -0.2]

Perceptron Learning Rule (SGD)

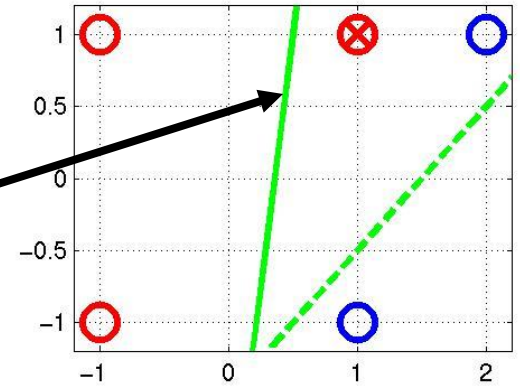
w=[0.45 –0.3 0.3]

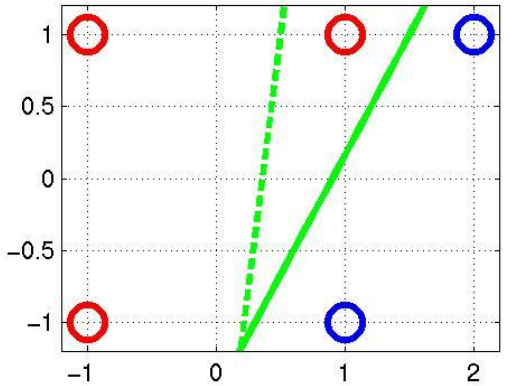
w=[0.25 –0.1 0.5]





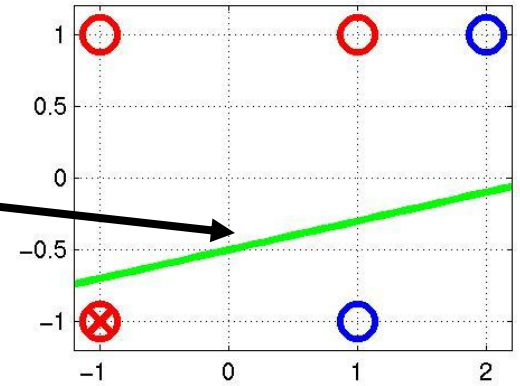
Δw=[-0.2 –0.4 –0.2]
w=w+ Δw
→
w=[0.25 -0.7 0.1]



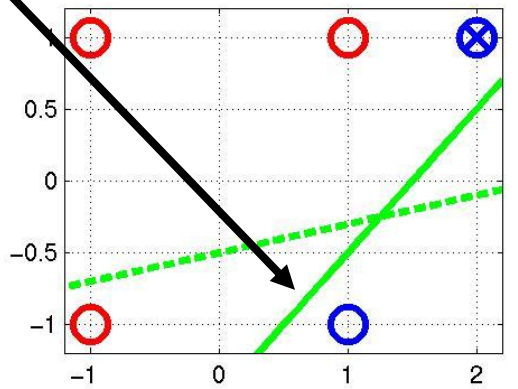


Perceptron Learning Rule (SGD)

w=[0.25 –0.1 0.5]

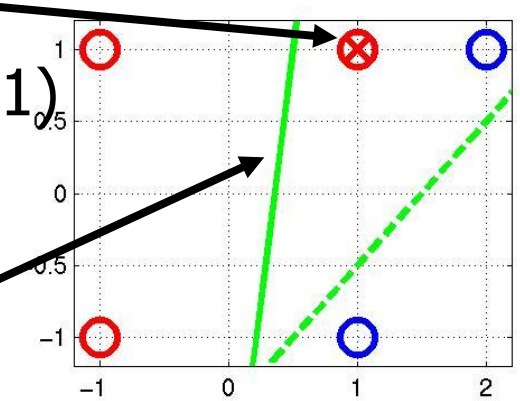


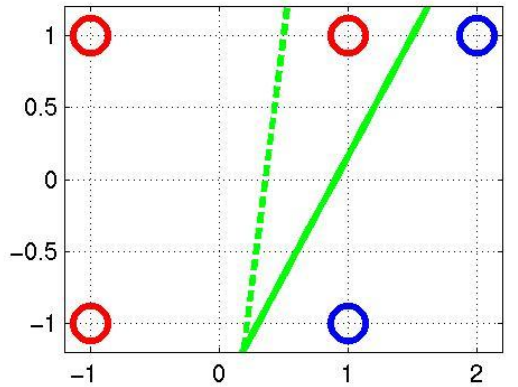
w=[0.45 –0.3 0.3]



(x,t)=([1,1],1)
o=sgn(0.25-0.7+0.1)
=-1

w=[0.25 -0.7 0.1]

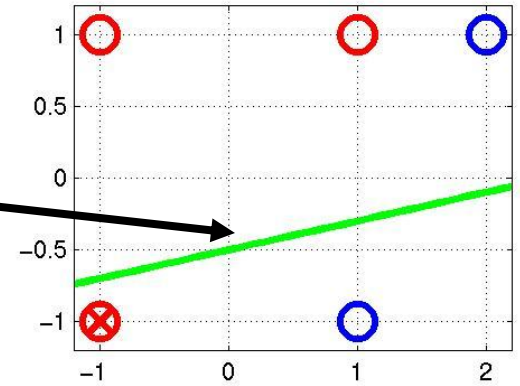


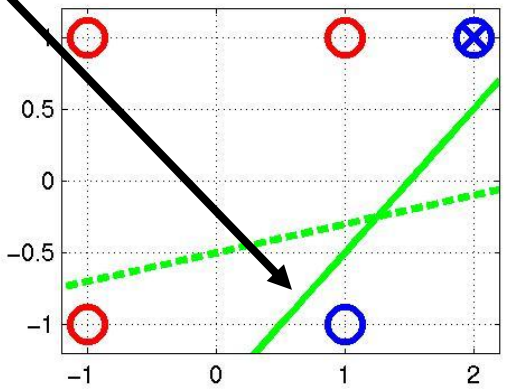


Perceptron Learning Rule (SGD)

w=[0.45 –0.3 0.3]

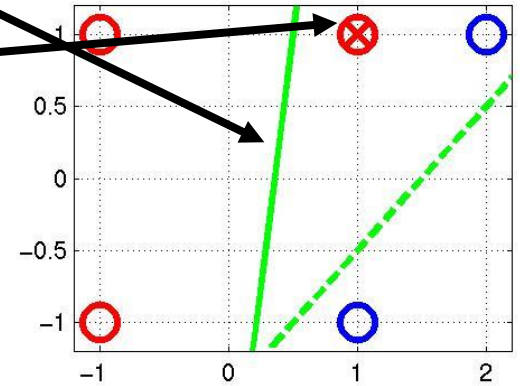
w=[0.25 –0.1 0.5]

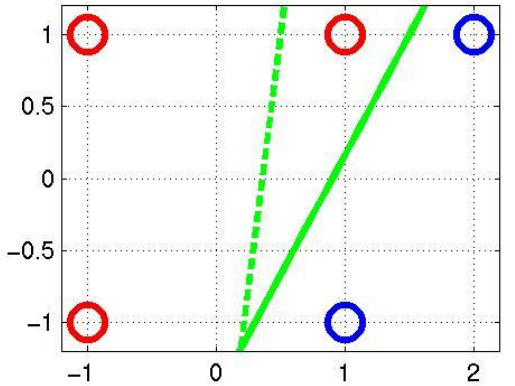




w=[0.25 -0.7 0.1]

t=1, o=-1
→ t-o=2
η=0.1 →
$\Delta w = \eta(t-o)\, x_i$ →
Δw=0.1*2*[1 1 1]=
[0.2 0.2 0.2]





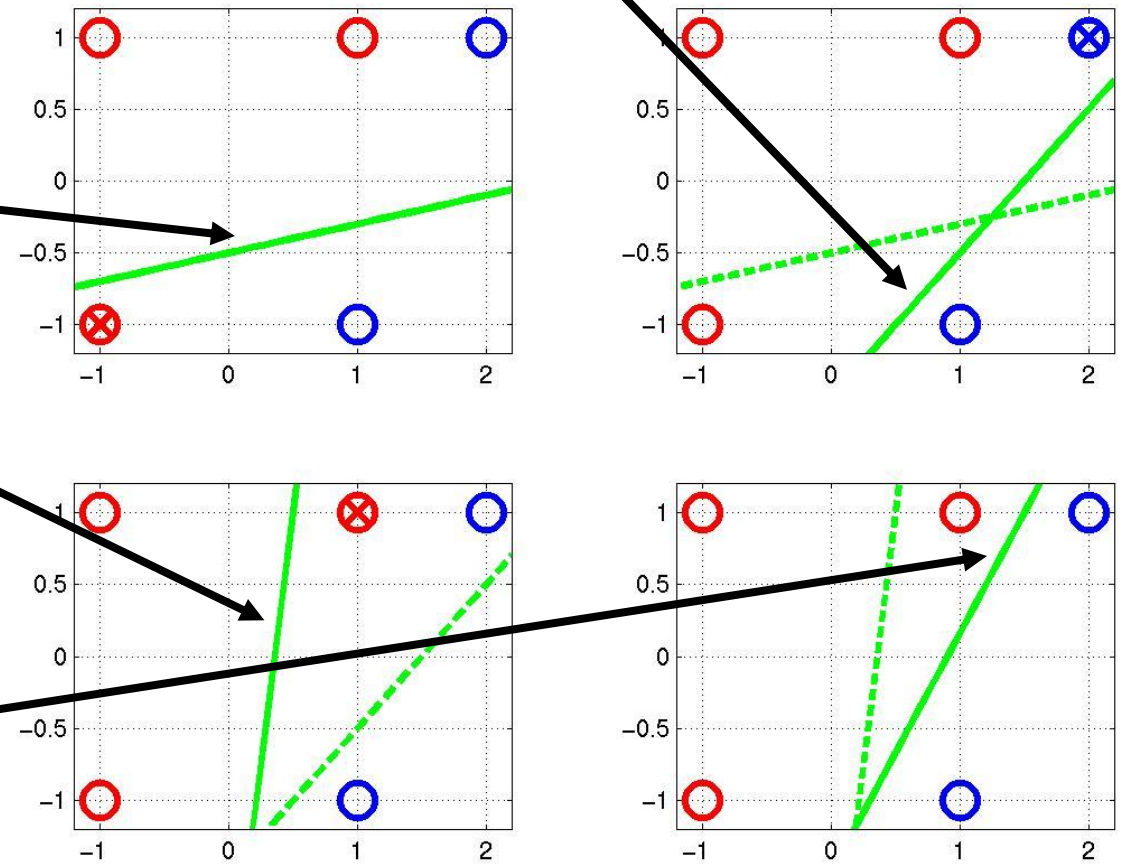
Perceptron Learning Rule (SGD)

w=[0.45 –0.3 0.3]

w=[0.25 –0.1 0.5]

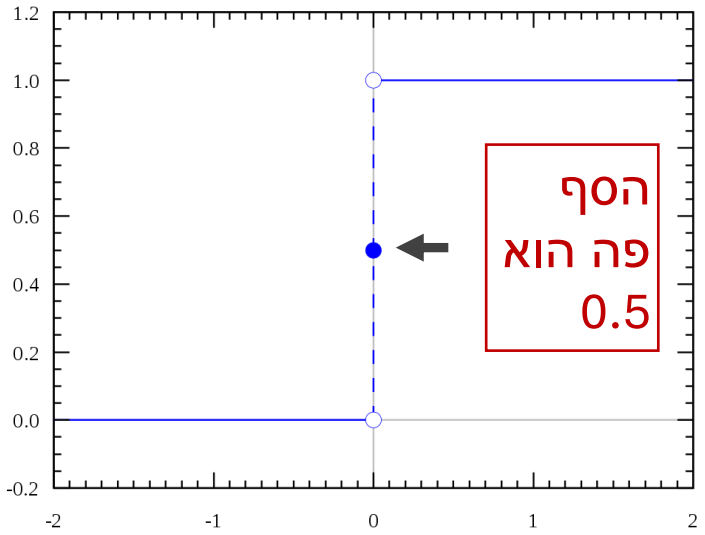
w=[0.25 -0.7 0.1]

Δw=[0.2 0.2 0.2]
w=w+ Δw
→
w=[0.45 -0.5 0.3]

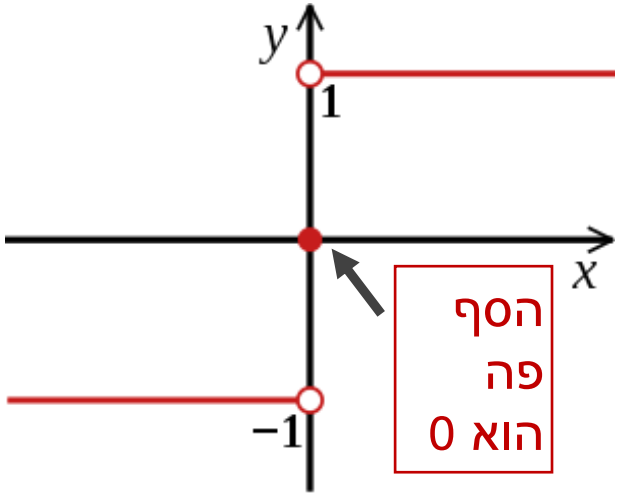


Perceptron – מודל א' – פונקצית ההפעלה

<u>פונקצית הסף</u> – פונקצית ההפעלה היא פונקצית מדרגה ובעצם ראינו 2 כאלה:



Heaviside step function
(פונקצית מדרגה)

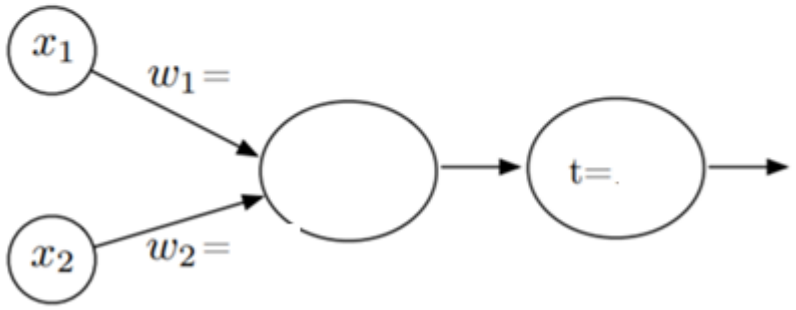


Signum function
(פונקצית הסימן – sign)

פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

Logical AND Gate

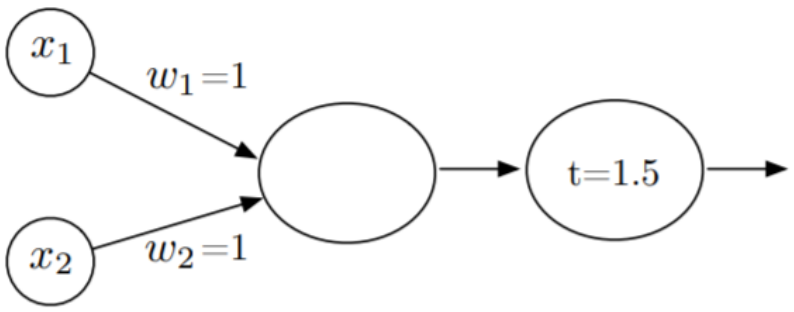
x_1	x_2	Out
0	0	0
0	1	0
1	0	0
1	1	1



פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

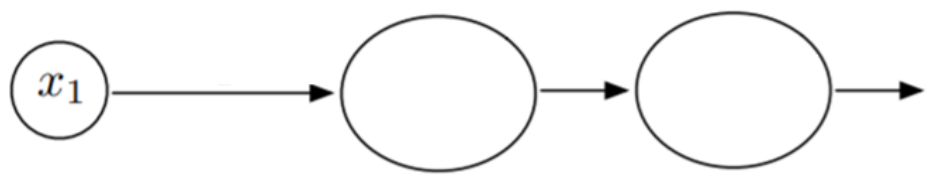
Logical AND Gate

x_1	x_2	Out
0	0	0
0	1	0
1	0	0
1	1	1



x_1	Out
0	1
1	0

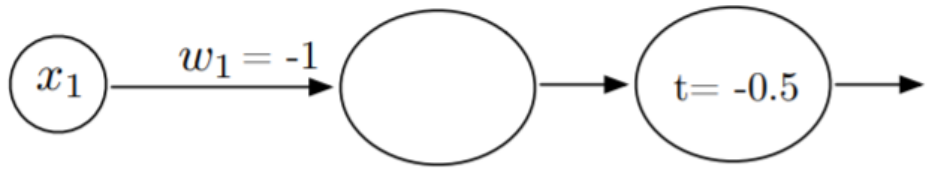
Logical NOT Gate



פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

Logical NOT Gate

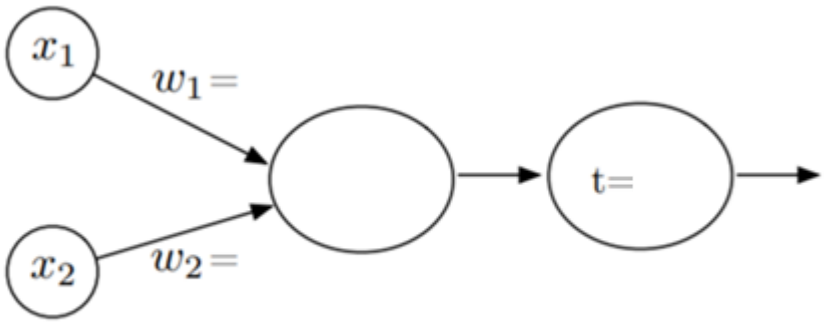
x_1	Out
0	1
1	0



פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

Logical OR Gate

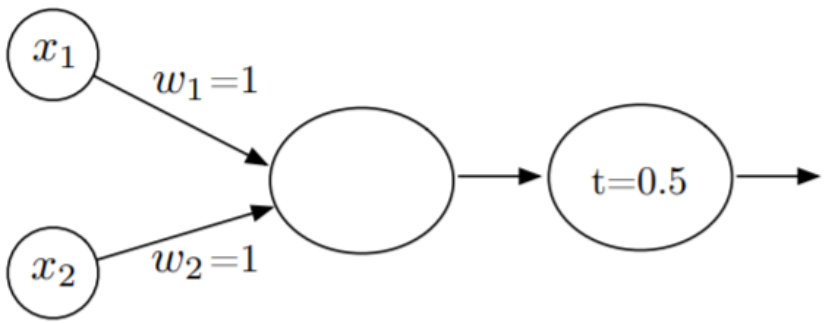
x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	1



פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

Logical OR Gate

x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	1

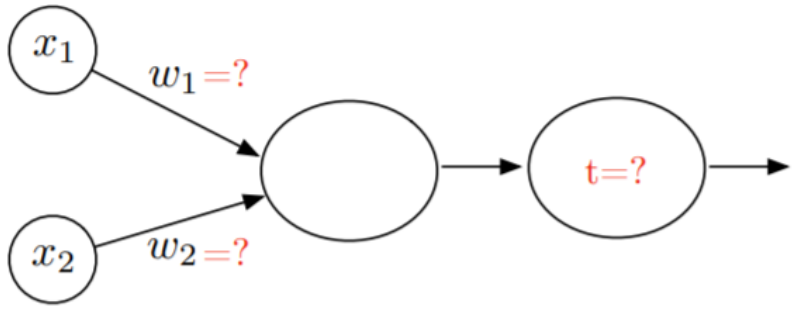


פרצפטרון (עם פו' Heaviside) ושערים לוגיים - דוגמאות

Logical XOR Gate

(Take-home exercise)

x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	0



פונקצית הפסד (Loss function) או
פונקצית עלות (Cost function) - תזכורת

<u>פונקצית הפסד או פונקצית מחיר</u> - פונקציה הממפה מאורע או ערכים של משתנה אחד או יותר למספר ממשי המייצג "עלות" של מאורע.

- בלמידה נסתכל למשל עלות של טעות בסיווג למשל

<u>נסמן</u>: J או loss – כפונקצית ההפסד

J(misclassification of i) – מחיר ההפסד של סיווג לא נכון

- סך ההפסד בלמידה, משתמש בשיטות כפי שראינו (ונראה עוד) בשערוך המודל.

<u>פונקצית מטרה (objective function)</u> –

בהקשר שלנו - פונקצית המטרה תהווה, בדרך כלל, פונקצית ההפסד או פונקצית הטעות, אותה נגדיר.

בעיית סיווג - תזכורת

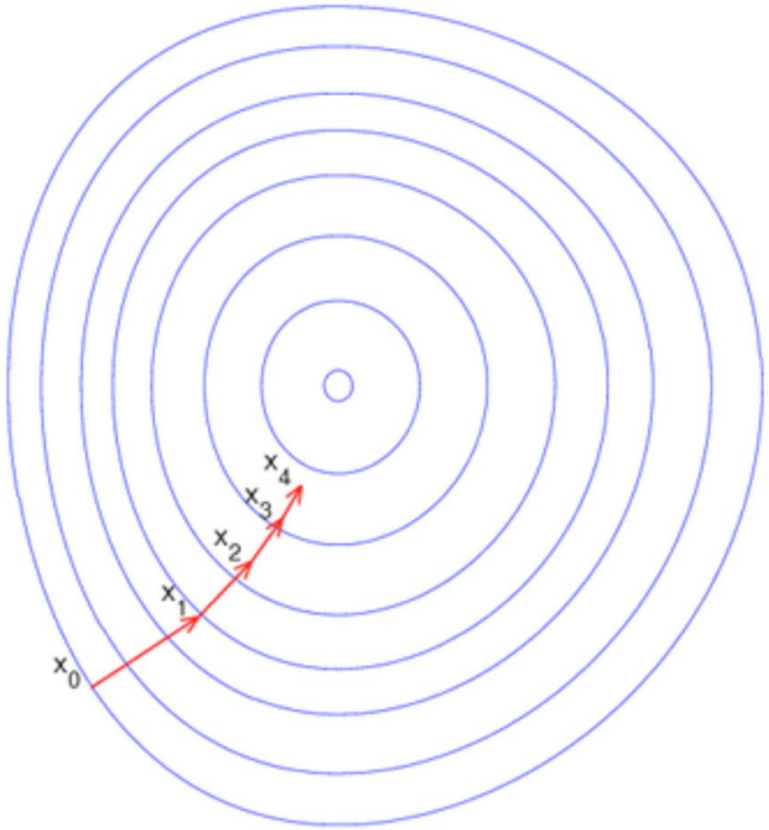
כלומר, אנו רוצים למצוא היפותזה במרחב ההיפותזות H, כך ש –

$$H \in \Re^D \longrightarrow \{-1, 1\}$$

כך שבהינתן "מחירון ענישה" לטעות $J\left(y,H(x)\right)= loss\left(y,H(x)\right)$

- אנו נמצא את h שממזערת לנו את הטעות על ה-training set

תזכורת- Gradient Descent



- שיטה כללית לאופטימיזציה של פונקציה רבת משתנים
- השיטה למעשה מבצעת חיפוש hill climbing כשהצעד נקלח בכיוון הנגזרת (או השלילה שלה במקרה חיפוש מינימום)

תזכורת – Gradient Descent

<u>המטרה</u>:

- ניתן לראות תהליך למידה כתהליך חיפוש במרחב היפותזות (במקרה שלנו כל מודלי הסיווג האפשריים לבעיה שלנו)
- אנחנו מחפשים היפותזה שתתאים בצורה הטובה ביותר לקבוצת דוגמאות האימון

<u>הרעיון</u>:

- התאמה של המודל (ההיפותזה) <u>בדרך כלל ממזערת שגיאה אמפירית</u>
- <u>loss function</u> - לפונקציה שאותה רוצים למזער קוראים loss function

<u>השיטה</u>:

- אנחנו בעצם מבצעים חיפוש הפוך של המשקולות (המקדמים).
- במקום למצוא את המקדמים ישירות, מחפשים מקדמים בעלי השגיאה הקטנה ביותר על דוגמאות האימון (מודל שיסווג נכון ככל הניתן את דוגמאות האימון)
  - מזעור בד"כ באמצעות בשיטה הדומה לנגזרת (נסביר על כך עוד בשיעורים הבאים).
- תנאי סיום – ראו בסוף המצגת

Gradient Descent Learning Rule

- Consider linear unit without threshold and continuous output o (not just –1,1)
  - $o=w_0 + w_1 x_1 + \ldots + w_n x_n$
- Train the w_i's such that they minimize the squared error
  - $E[w_1,\ldots,w_n] = \frac{1}{2} \sum_{d \in S} (t_d - o_d)^2$

  where S is the set of training examples

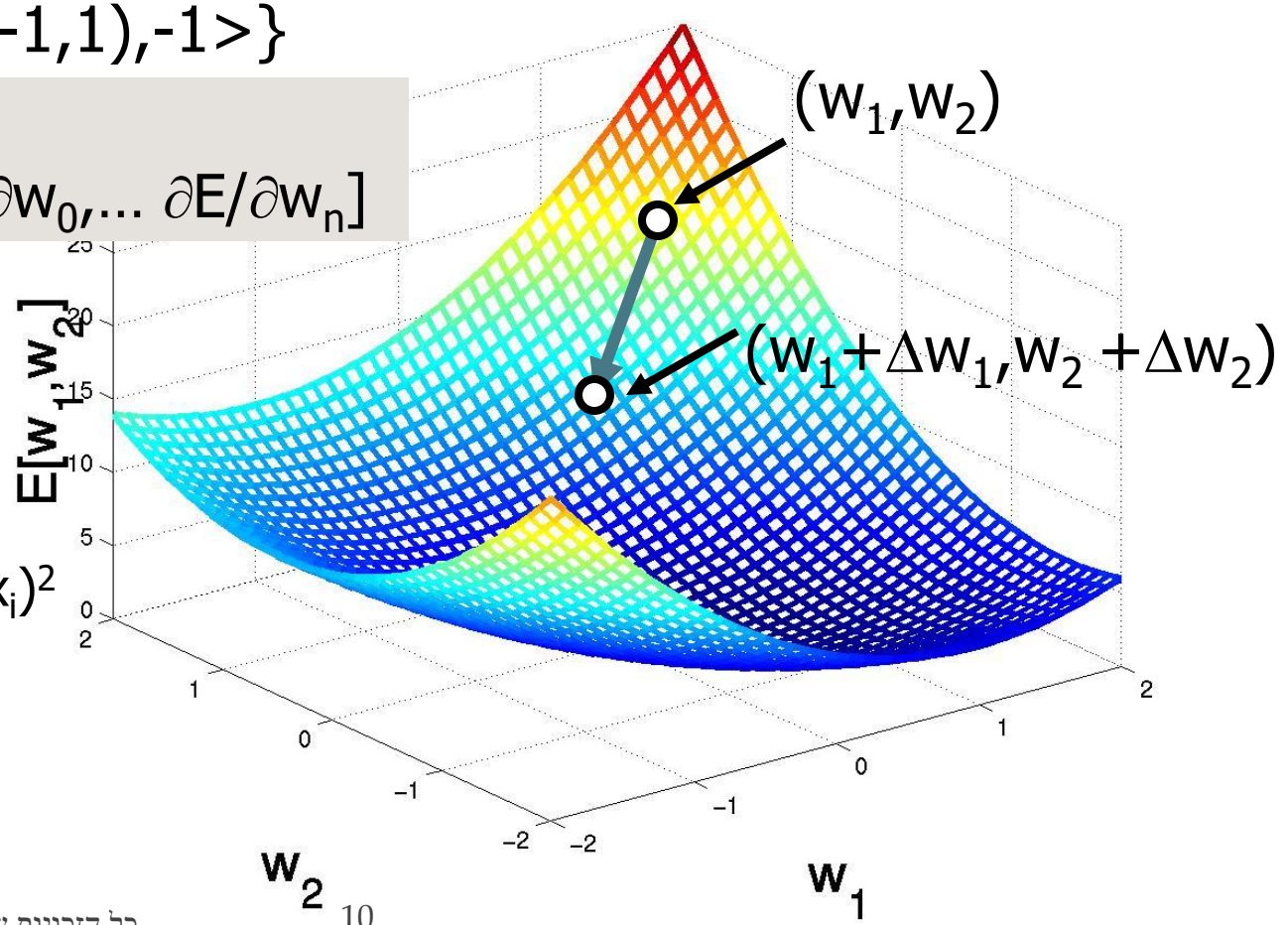
Gradient Descent

$S=\{<(1,1),1>,<(-1,-1),1>,$
$<(1,-1),-1>,<(-1,1),-1>\}$

Gradient:
$\nabla J[w] = \nabla E[w] = [\partial E/\partial w_0, \ldots \partial E/\partial w_n]$

$\Delta w = -\eta \nabla E[w]$

$\Delta w_i = -\eta \partial E/\partial w_i$
$= \partial/\partial w_i \ 1/2 \sum_d (t_d - o_d)^2$
$= \partial/\partial w_i \ 1/2 \sum_d (t_d - \sum_i w_i x_i)^2$
$= \sum_d (t_d - o_d)(-x_i)$



Perceptron with Gradient Descent

Loss Function

We can say the perceptron optimizes a loss function analogous to the squared error in least-squares regression, except that we have targets and outputs:

$$\mathcal{L}(\mathbf{w}, b) = \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2$$

where $\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} + b \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} + b > 0 \end{cases}$

Perceptron with Gradient Descent

$$\mathcal{L}(\mathbf{w}, b) = \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \qquad \text{where} \quad \hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} + b \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} + b > 0 \end{cases}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \\
&= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]})^2 \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \frac{\partial}{\partial w_j}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) \frac{\partial}{\partial w_j} \mathbf{w}^T\mathbf{x}^{[i]} \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) x_j^{[i]}
\end{aligned}$$

Perceptron with Gradient Descent

$$\mathcal{L}(\mathbf{w}, b) = \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \qquad \text{where} \quad \hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} + b \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} + b > 0 \end{cases}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \\
&= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]})^2 \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \frac{\partial}{\partial w_j}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) \frac{\partial}{\partial w_j} \mathbf{w}^T\mathbf{x}^{[i]} \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) x_j^{[i]}
\end{aligned}$$

not differentiable!

Perceptron with Gradient Descent

$$\mathcal{L}(\mathbf{w}, b) = \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \qquad \text{where} \quad \hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} + b \leq 0 \\ 1, & \mathbf{w}^T\mathbf{x} + b > 0 \end{cases}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \\
&= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]})^2 \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \frac{\partial}{\partial w_j}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) \frac{\partial}{\partial w_j} \mathbf{w}^T\mathbf{x}^{[i]} \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) x_j^{[i]}
\end{aligned}$$

not differentiable!

However, perceptron does something very similar to stochastic gradient descent:

(not a real derivative)

$$\frac{\partial \mathcal{L}}{\partial w_j} = (\hat{y}^{[i]} - y^{[i]}) x_j$$

$$w_j := w_j - \frac{\partial \mathcal{L}}{\partial w_j}$$

$$w_j := w_j + (y^{[i]} - \hat{y}^{[i]}) x_j$$

אימון פרצפטרון (with sign function) – פסאודו קוד

סימונים:

- η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)
- t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה
- o הערך שנותן ה נוירון עבור הדוגמא הנוכחית
- $\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה
- Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

perceptron_train(training examples, η):

- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta(t-o)x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

גרסה זו קיימת, למרות שפונקציית ה-sign, אינה גזירה ואינה דפרנציבלית

Comparison Perceptron and Gradient Descent Rule

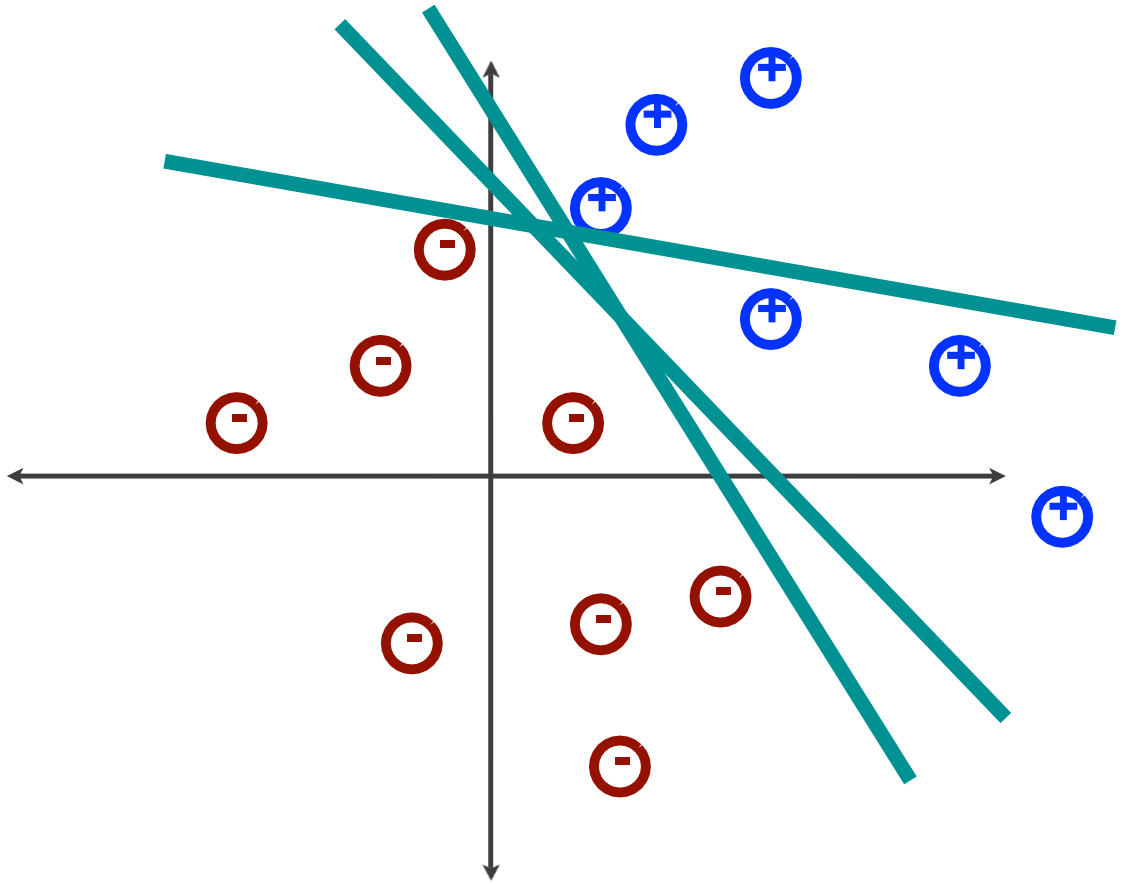
Perceptron learning rule guaranteed to succeed if

- Training examples are linearly separable
- No guarantee otherwise

Linear unit using Gradient Descent

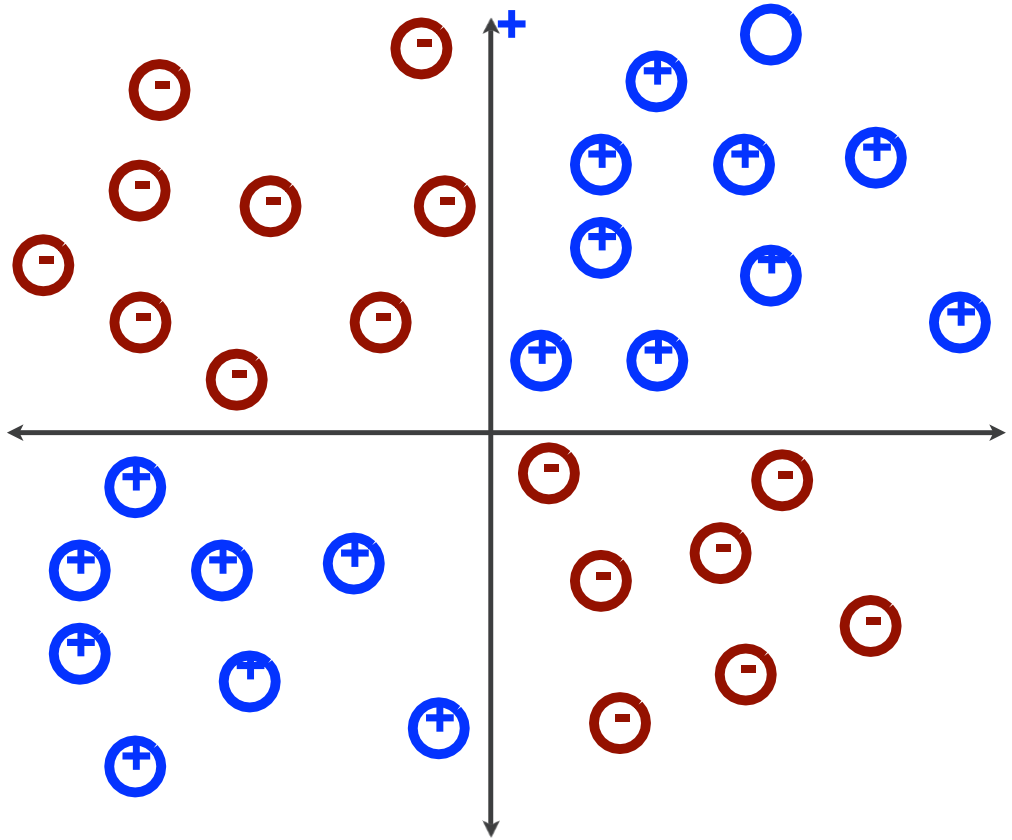
- Converges to hypothesis with minimum squared error.
- Given sufficiently small learning rate η
- Even when training data contains noise
- Even when training data not linearly separable

האם כל בעית למידה ניתנת למידול באמצעות מפריד לינארי?



קבוצת דוגמאות שלא ניתנת להפרדה לינארית

שאלה: האם ישנו מודל דיסקרמינטיבי (מפריד), שיכול לטפל במקרים ללא הפרדה לינארית?

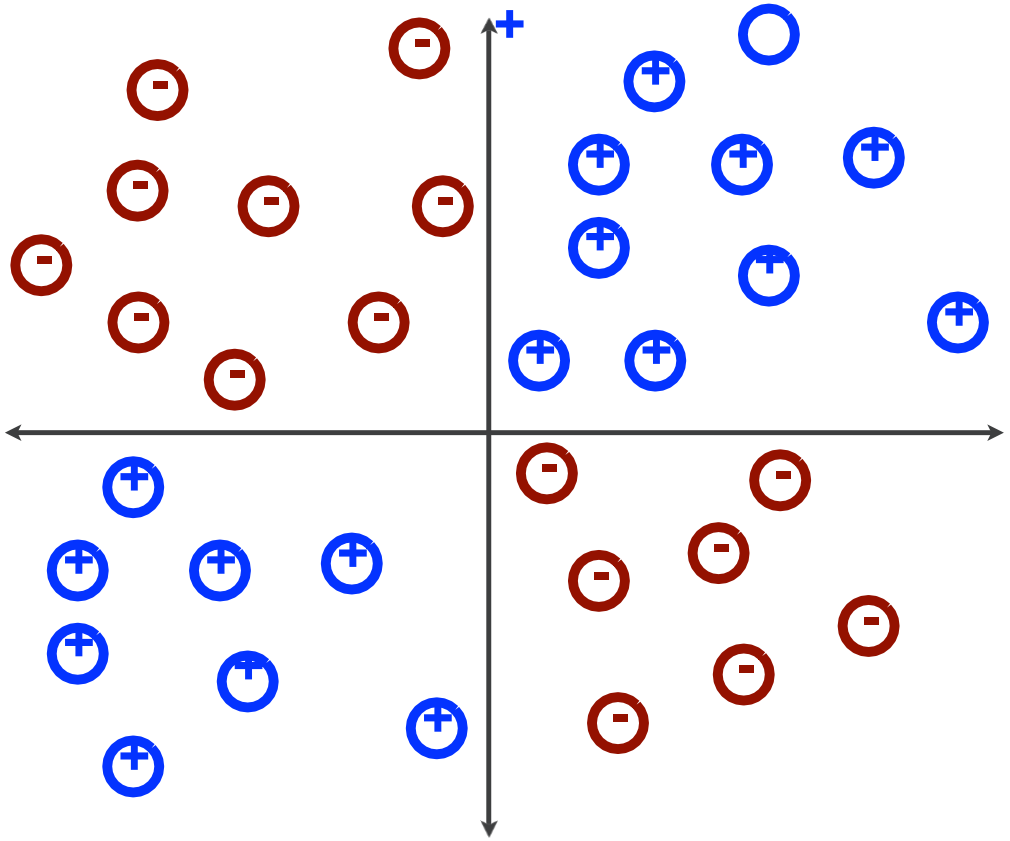


קבוצת דוגמאות שלא ניתנת להפרדה לינארית

שאלה: האם ישנו מודל דיסקרמינטיבי (מפריד), שיכול לטפל במקרים ללא הפרדה לינארית?

תשובה: ישנם 2 אפשרויות:

1. משוואה לא לינארית.
2. תרכובת של מפרידים לינאריים (נראה אפשרות זו בהמשך)

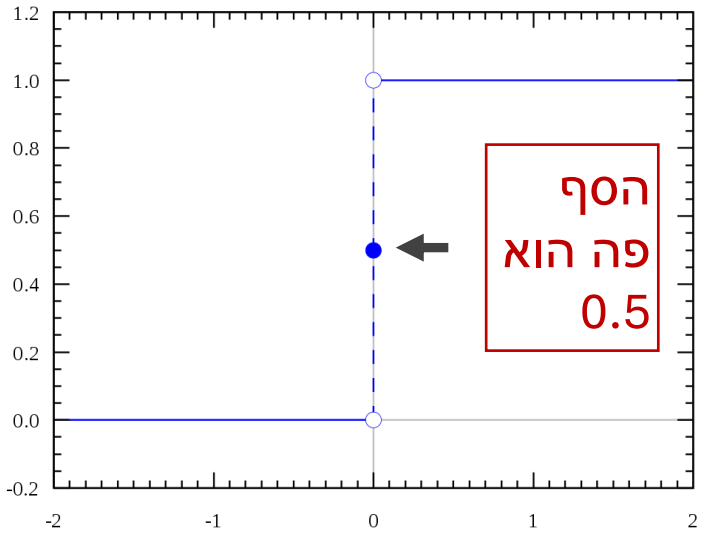


מודל ב' – Logistic Regression - החלפת היחידה הבסיסית - sigmoid

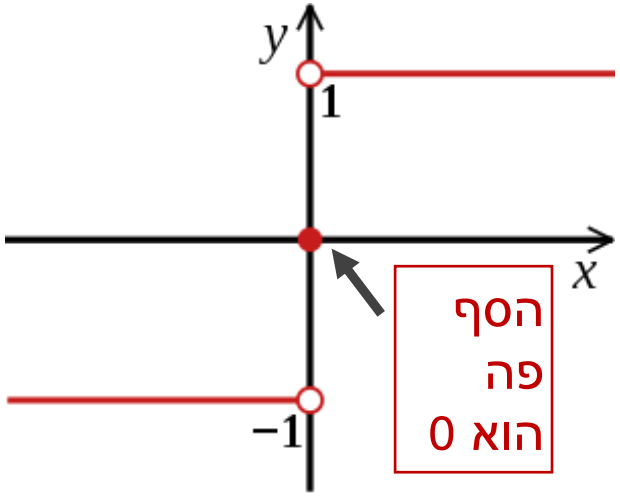
- החלפת פונקצית sign
- פונקצית הפסד ופונקצית מטרה
- טענת התכנסות
- Gradient Decent

Perceptron – מודל א' – פונקצית ההפעלה

<u>פונקצית הסף</u> – פונקצית ההפעלה היא פונקצית מדרגה ובעצם ראינו 2 כאלה:

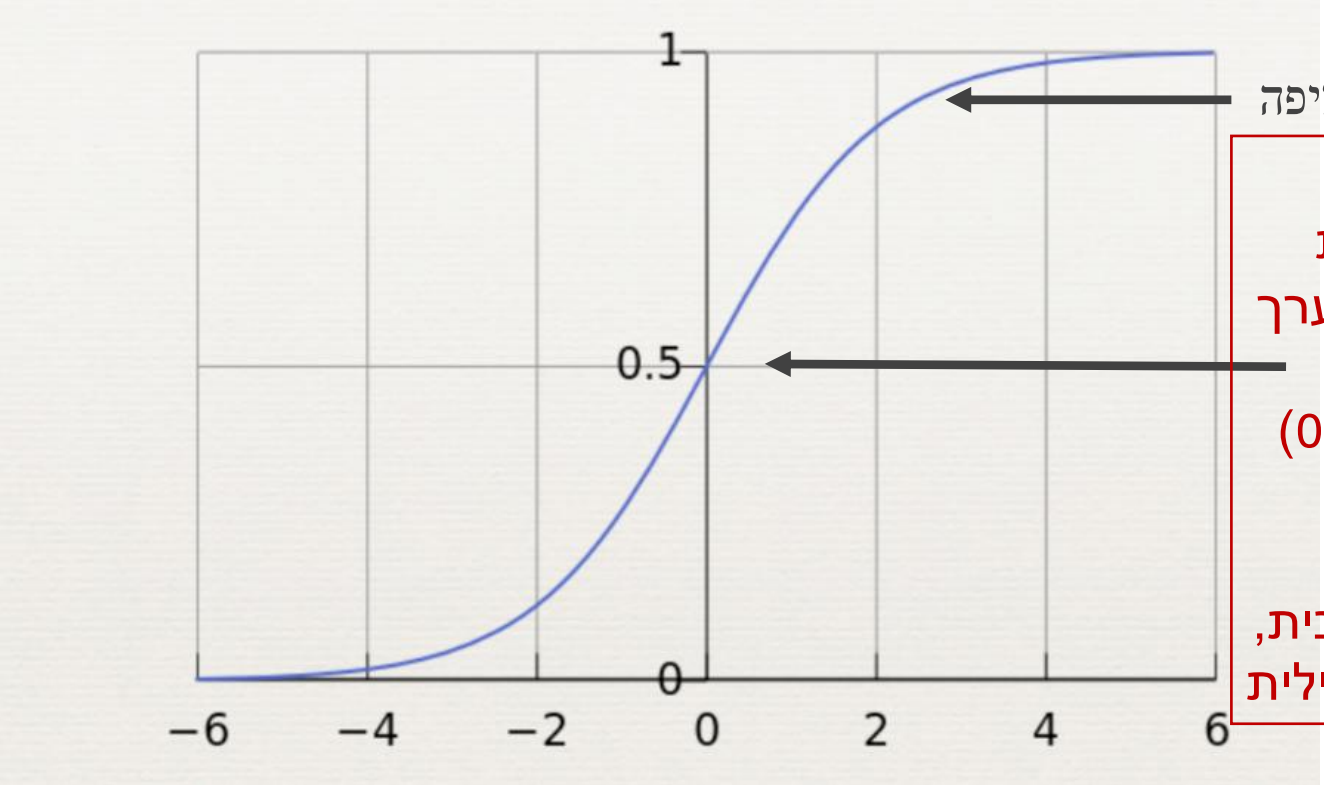


Heaviside step function
(פונקצית מדרגה)



Signum function
(פונקצית הסימן – sign)

מודל ב' – Logistic Regression - החלפת "פונקציית ההפעלה" - sigmoid *

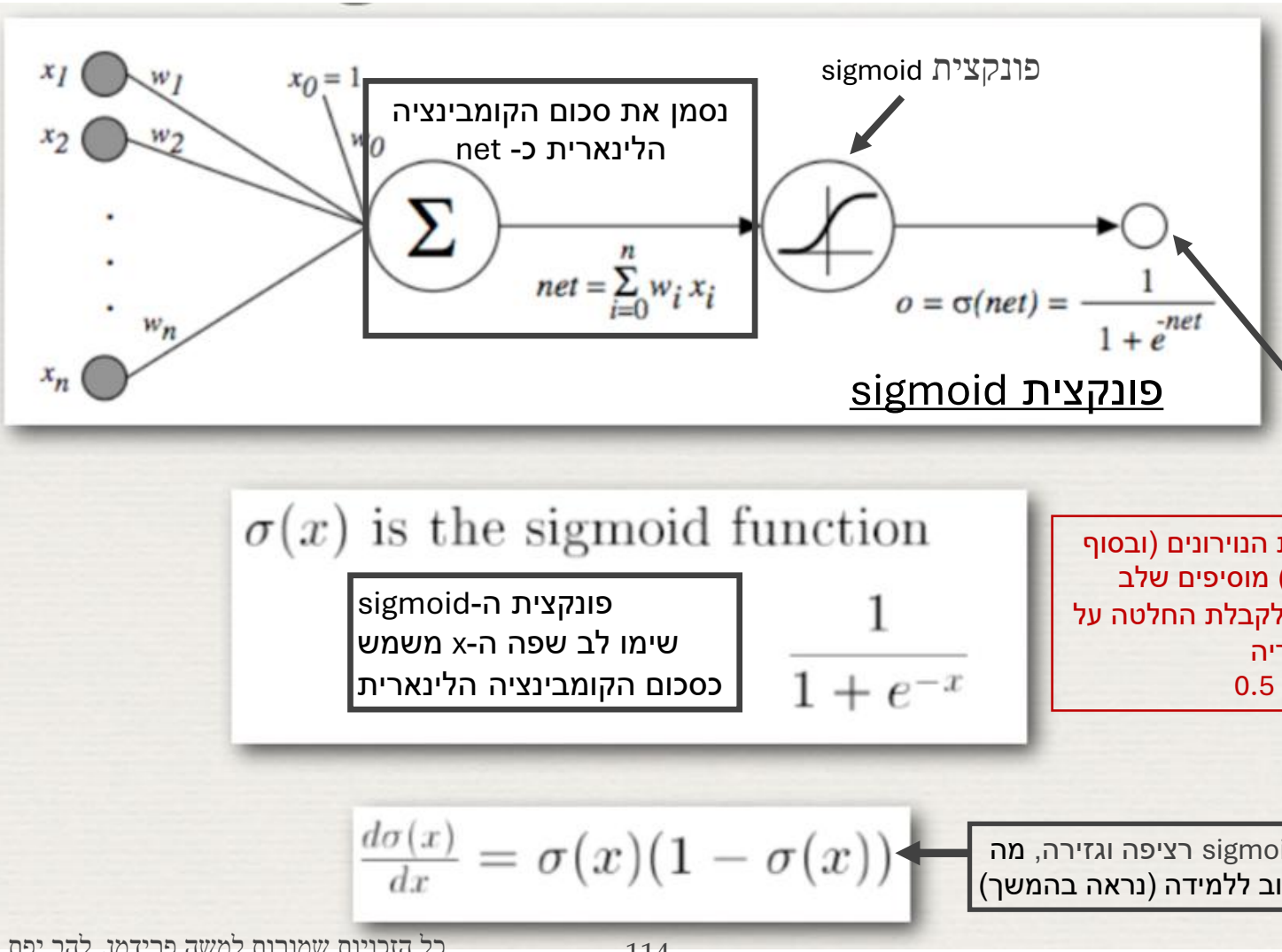


פונקציית ה-sigmoid רציפה

* מוסיפים כ- post processing שלב בדיקת סף, לקבלת החלטה על ערך הקטגוריה
* הסף פה הוא 0.5 (ולא 0)
- אם תוצאת הפונקציה גדולה מ-0.5,
- נסווג כקטגוריה החיובית, אחרת, כקטגוריה שלילית

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$

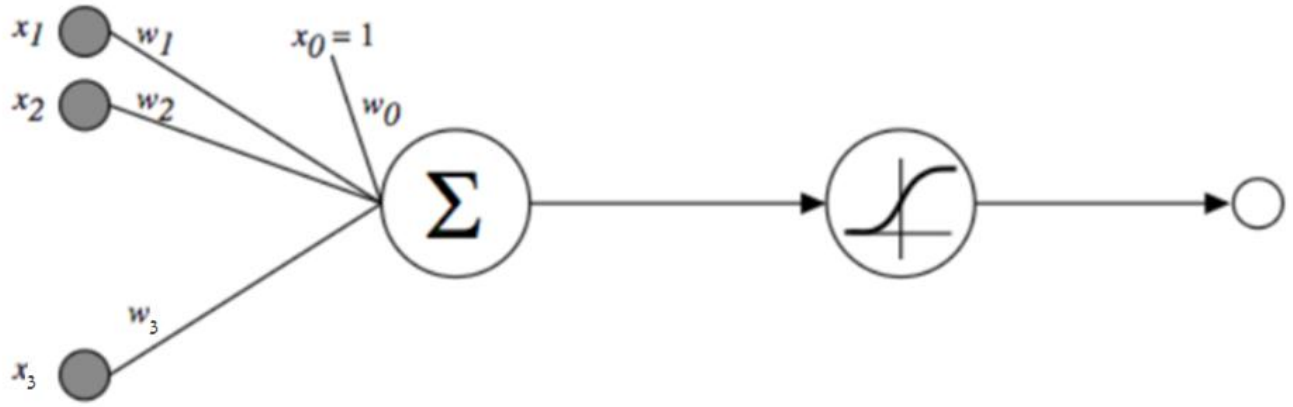
- Logistic Regression – מודל ב'
יחידת sigmoid



דוגמה לקריאת Logistic Regression

$$S(x) = \frac{1}{1 + e^{-x}}$$

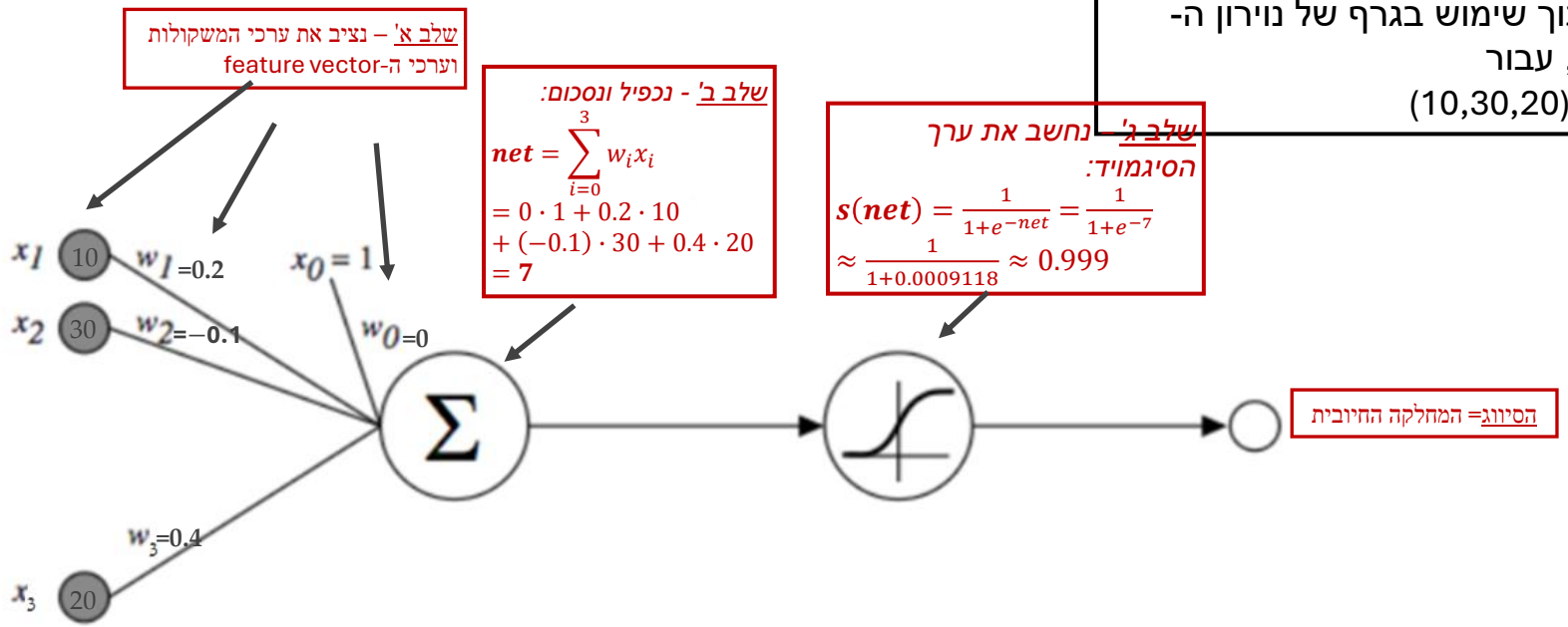
נתונה יחידת הנוירון הבאה, כאשר מישור ההפרדה מיוצג ע"י:
$0.2 \cdot x_1 - 0.1 \cdot x_2 + 0.4 \cdot x_3 = 0$
בצעו סימולציה לקבלת הפלט של הנוירון, תוך שימוש בגרף של נוירון ה-sigmoid, עבור הוקטור: (10,30,20)



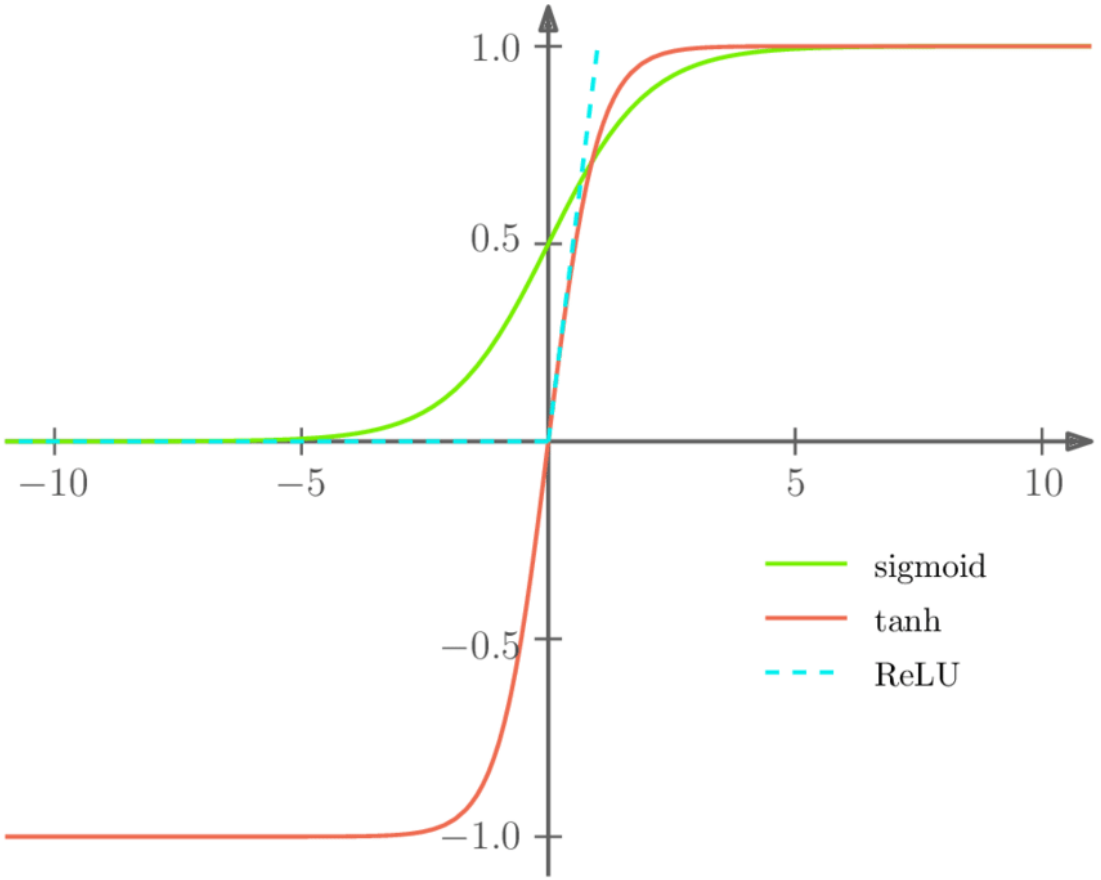
דוגמה לקריאת Logistic Regression

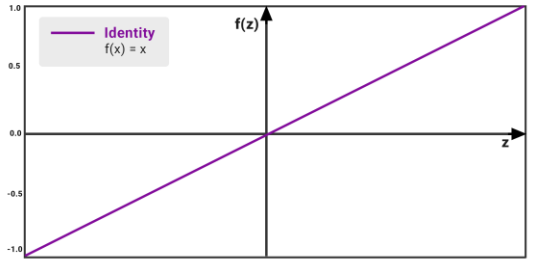
$$S(x) = \frac{1}{1+e^{-x}}$$

נתונה יחידת הנוירון הבאה, כאשר מישור ההפרדה מיוצג ע"י:
$0.2 \cdot x_1 - 0.1 \cdot x_2 + 0.4 \cdot x_3 = 0$
בצעו סימולציה לקבלת הפלט של הנוירון, תוך שימוש בגרף של נוירון ה-sigmoid, עבור הוקטור: (10,30,20)



פונקציות אקטיבציה נוספות





identity
ident(z) = z

sigmoid
$s(z) = \frac{1}{1+e^{-z}}$

tanh
$\tanh(z) = \frac{e^{2z}-1}{e^{2z}+1}$

ReLU (Rectified Linear Unit)
R(z) = max(0,z)

Gradient Descent – תכונות - תזכורת

<u>פונקצית הפסד קמורה</u>

במקרה של perceptron (עם sigmoid) פונקציית ההפסד היא פונקציה קמורה

בגלל שפונקציית ההפסד קמורה, מובטח לנו שהאלגוריתם מתכנס להיפותזה בעלת שגיאה מינימלית, עבור קבוע למידה קטן מספיק

- הדבר נכון גם כאשר יש רעש בדוגמאות, וגם עבור קבוצת דוגמאות שאינה ניתנת להפרדה ליניארית

with sigmoid) Logistic Regression ב Gradient Descent – (function

$$\mathcal{L}(\mathbf{w}, b) = \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2$$

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2 \\
&= \frac{\partial}{\partial w_j} \sum_i \frac{1}{2}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]})^2 \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \frac{\partial}{\partial w_j}(\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) \frac{\partial}{\partial w_j} \mathbf{w}^T\mathbf{x}^{[i]} \\
&= \sum_i (\sigma(\mathbf{w}^T\mathbf{x}^{[i]}) - y^{[i]}) \sigma'(\mathbf{w}^T\mathbf{x}^{[i]}) x_j^{[i]}
\end{aligned}$$

לכן, הנגזרת החלקית של w_j:

$$\frac{\partial J}{\partial w_j} = (o - t) \cdot o(1 - o) \cdot x_j$$

$$= -(t - o) \cdot o(1 - o) \cdot x_j$$

Sebastian Raschka http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/

Logistic Regression ב Gradient Descent – (with sigmoid function)

סימונים:

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה נוירון עבור הדוגמא הנוכחית

$\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה

Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

Gradient Descent (training examples, η):

- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta \cdot 2(t-o) \cdot o \cdot (1-o) \cdot x_i$
    - $w_i^{t+1} := w_i^t + \Delta w_i$

Gradient Descent ב Logistic Regression (with sigmoid – (function

סימונים:

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה נוירון עבור הדוגמא הנוכחית

$\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה

Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

Gradient Descent (training examples, η):

- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta \cdot 2(t-o) \cdot o \cdot (1-o) \cdot x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

לעיתים קרובות מתעלמים מה-2 ע"י הכפלת פונקצית הטעות ב ½

J= $\sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2$

ב Gradient Descent
Logistic Regression
with sigmoid)
– (function

```
E=1/2 sum((t[i]-o[i])^2
dJ/dw[j]=(t-o)*-1*(o*(1-o))*x[j]
=-(t-o)*(o*(1-o))*x[j]
delta(w[j])=delta(w[j])-eta*dJ/dw[j]
= delta(w[j])=delta(w[j])-[-(t-o)*(o*(1-o))*x[j]]
= delta(w[j])=delta(w[j])+t-o)*(o*(1-o))*x[j]
```

סימונים:

- η קבוע (קטן מ-1) הקוב
- t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה
- o הערך שנותן ה נוירון עבור הדוגמא הנוכחית
- $\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה
- Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

אלגוריתם:

Gradient Descent (training examples, η):

- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x}, t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta \cdot (t-o) \cdot o \cdot (1-o) \cdot x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

לעיתים קרובות מתעלמים מה-2 ע"י הכפלת פונקצית הטעות ב ½

J= $\sum_i \frac{1}{2}(\hat{y}^{[i]} - y^{[i]})^2$

Full batch Gradient Descent

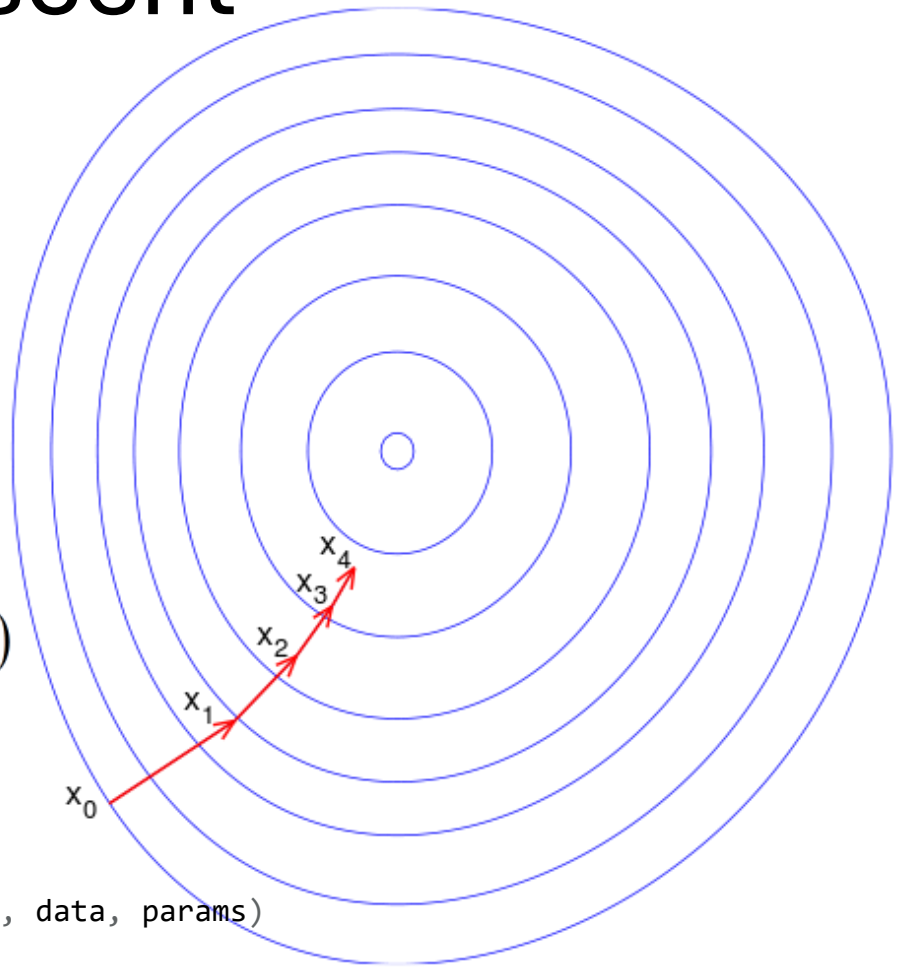
Repeat until convergence:

$$\theta_{t+1} = \theta_t - \alpha \nabla L$$

$$\nabla L \triangleq \frac{\partial L}{\partial \vec{\theta}}$$

where:

α : *Learning rate* $(sometimes\ \eta)$




```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

Gradient descent types

•Batch gradient descent

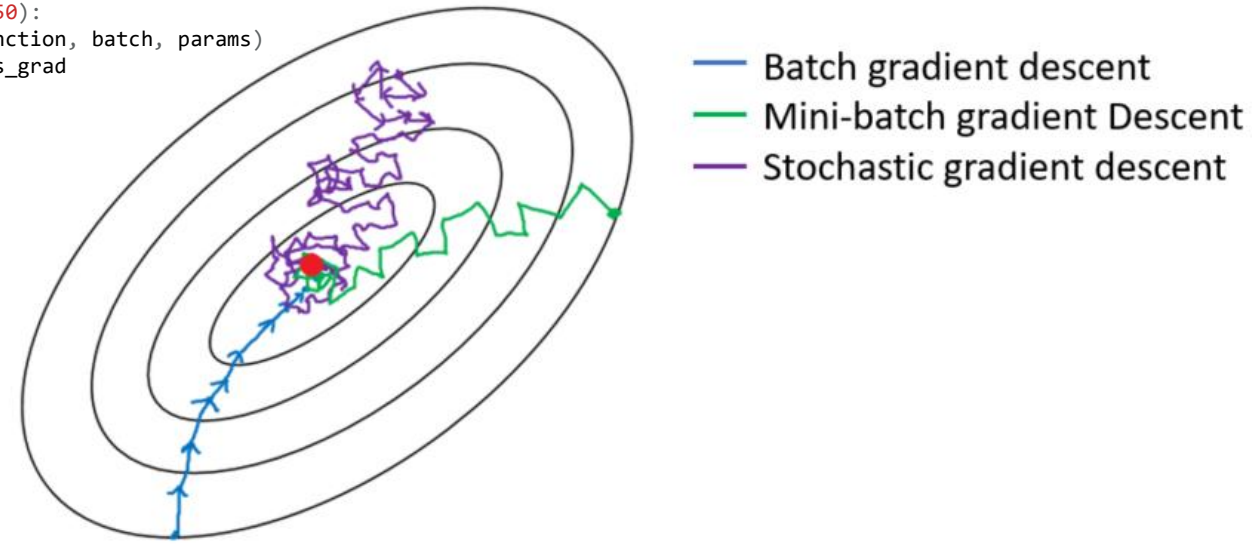
```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

•Stochastic gradient descent

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

•Mini-batch gradient descent

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
     params_grad = evaluate_gradient(loss_function, batch, params)
     params = params - learning_rate * params_grad
```



Momentum

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta).$$

$$\theta = \theta - v_t.$$

Image 2: SGD without momentum

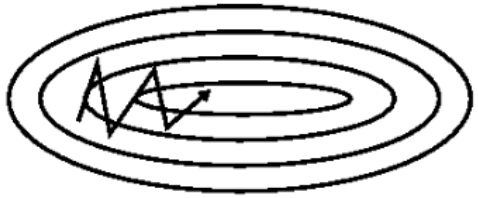
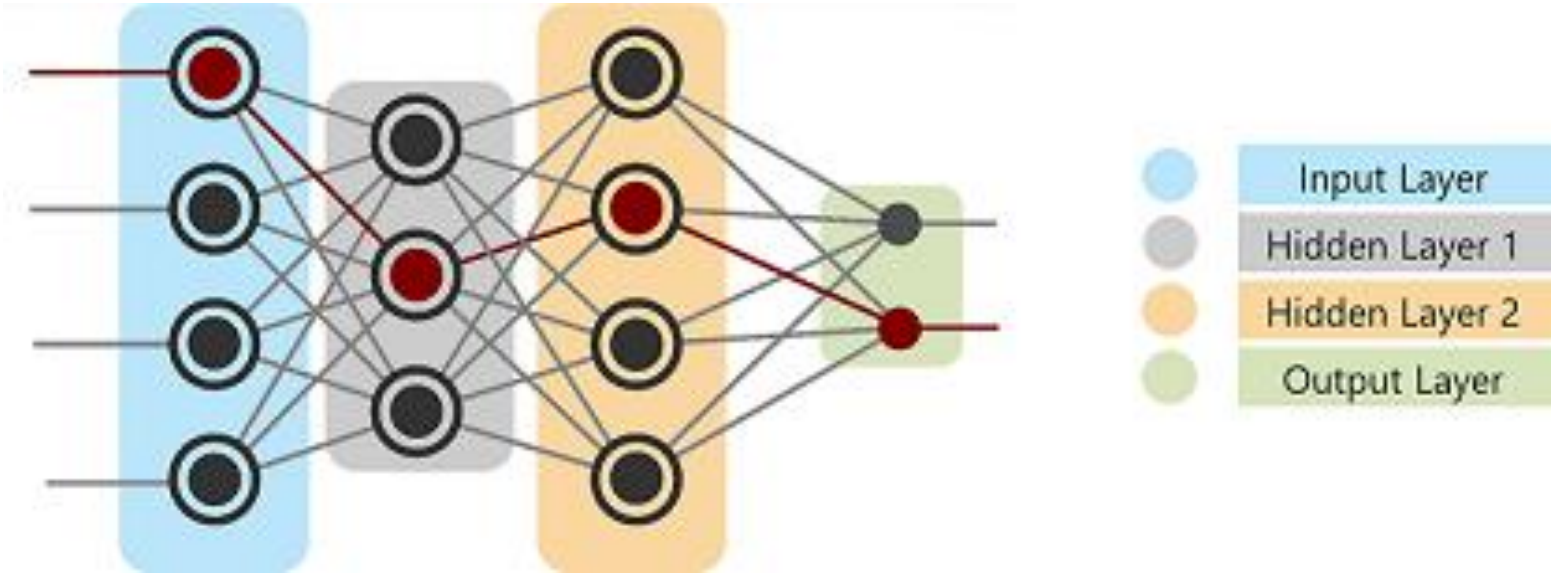


Image 3: SGD with momentum

```
grads=calc_grad(weights,data,labels)
momentum = gamma * momentum + lr * grads
weights = weights - momentum
```

What’s next

Artificial Neural networks – part 2:
Multi-layer perceptrons / neurons



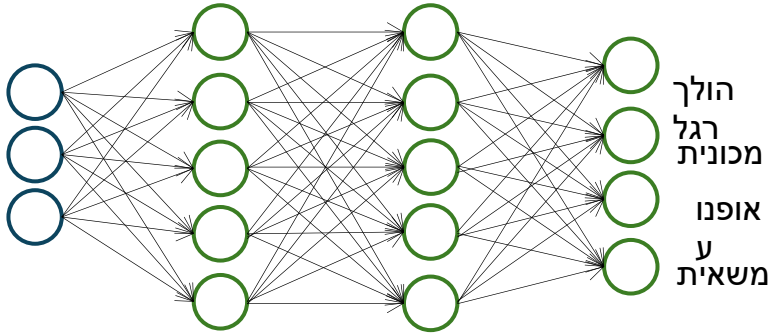
רשת רב שכבתית – מוטיבציה

Pedestrian

Car

Motorcycle

Truck



$$h_w(x) \in \mathbb{R}^4$$

$$h_w(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

הולך רגל

$$h_w(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

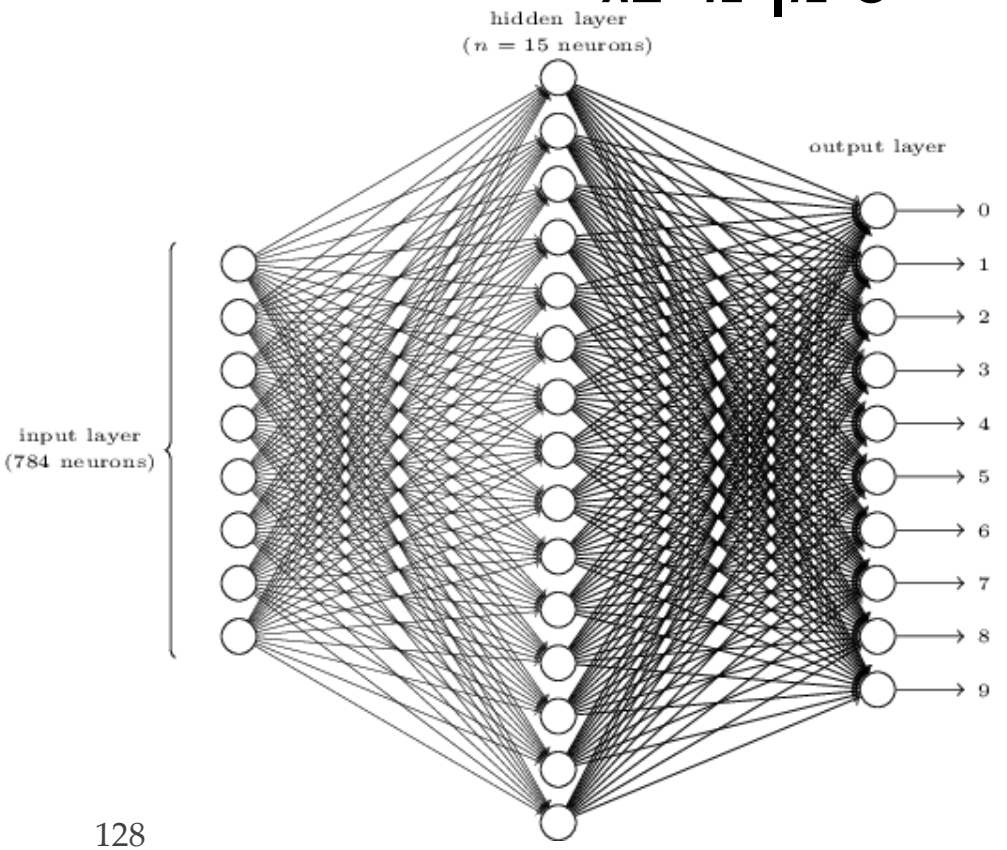
מכונית

$$h_w(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

אופנוע

רשתות רב שכבתיות - דוגמא

- אבחנה בין 10 סימנים שונים כדי להבין את הספרה שכל סימן מייצג



רשת רב שכבתית – מציאת מודל הסיווג - השיטה

אתחול פרמטרים אקראי

ננסה לשנות את הפרמטרים - במטרה לשפר את האיכות של המודל

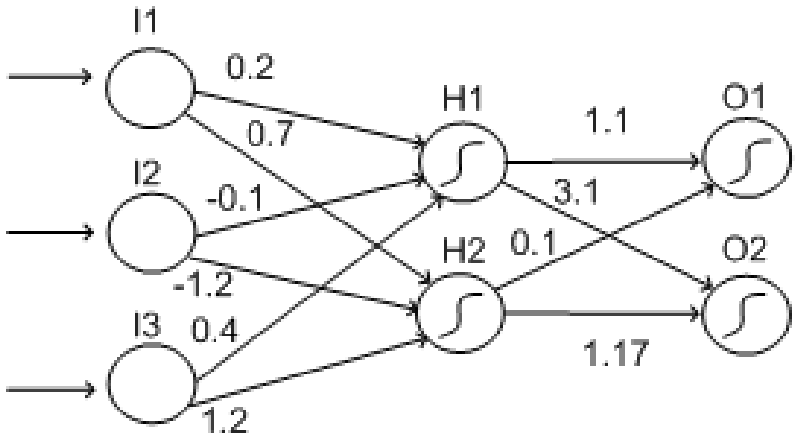
- השיטה: חיפוש ערכים טובים יותר של הפרמטרים, כדי שתוצאות המודל ישתפרו

איכות המודל = loss function =J= טעויות סיווג של דוגמאות האימון.
שיפור המודל = מזעור פונקציות הloss=מזעור פונקציית הטעות E

חיפוש ערכים טובים יותר = gradient descent = חיפוש חמדני על מגוון הערכים של הפרמטרים (מרחב ההיפותזה) – כדי לקבל טעות (E) [או הפסד (loss), כלומר מחיר (J)] נמוך יותר

דוגמה לקריאת רשת רב שכבתית

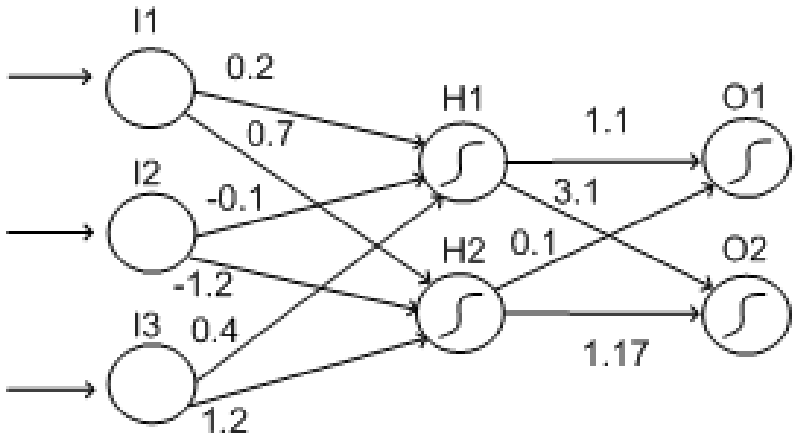
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



בצעו סימולציה לקריאת הרשת הבאה,
הדוגמה אותה נרצה לסווג: (10,30,20)

דוגמה לקריאת רשת רב שכבתית

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



בצעו סימולציה לקריאת הרשת הבאה, הדוגמה אותה נרצה לסווג: (10,30,20)

אם היה מדובר בדוגמה חדשה:
פלט: O2 > O1 --> נסווג את הדוגמה כ-O2

שלב קריאת הרשת וחלחול הערך משכבת הקלט עד לשכבת הפלט נקרא Feed Forward

Input units		Hidden units			Output units		
Unit	Output	Unit	Weighted Sum Input	Output	Unit	Weighted Sum Input	Output
I1	10	H1	7	0.999	O1	1.0996	0.750
I2	30	H2	-5	0.0067	O2	3.1047	0.957
I3	20						

אם היה מדובר בדוגמה חדשה

רשתות רב שכבתיות – אלגוריתם backpropagation

backpropagation(train_set, η, topology):

Initialization: Initialize all weights $(w_{i,j})$ to small random numbers.

Outer loop: for all train_set Until satisfied, Do

- For each training example $\langle \vec{x}, t_k \rangle$, Do

Feed forward:

1. Input the training example to the network and compute the network outputs (o_k)

Back propogation:

2. For each output unit k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

נתון כקלט לאלגוריתם:

Train set – כל זוגות האימון $\langle \vec{x}, t_k \rangle$.

$\vec{x}$ - מייצג את ה-feature vectors של האימון.

x_i - ערך מאפיין i (x_iים מהווים את שכבת הקלט)

t_k - יכול להיות כמה ערכים (אחד עבור כל יחידה חיצונית).

η – קבוע הלמידה (מספר הקטן מ-1, גדול מ-0)

topology – המבנה של רשת הנוירונים

יחידה חיצונית:

o_k - פלט עבור יחידה חיצונית k

t_k - פלט מצופה עבור יחידה חיצונית k

δ_k - הטעות (בעזרת גרדיאנט) בשכבה חיצונית k

יחידה נסתרת:

o_h - פלט עבור יחידה נסתרת h

הערה: אין t_h - פלט מצופה עבור יחידה נסתרת h

δ_h - הטעות עבור יחידה נסתרת h (לפי התרומה היחסית ליחדות הפלט החיצוניות)

$w_{h,k}$ - המשקולת בין יחידה נסתרת h ליחידה חיצונית k

כללי (i - קלט / יחידה נסתרת, j – יחידה נסתרת / חיצונית)

$w_{i,j}$ - משקולת הקשת בין i ל-j,

$\Delta w_{i,j}$ - התיקון למשקולת הקשת בין i ל-j;

$x_{i,j}$ - הקלט המחובר לקשת $w_{i,j}$

δ_j - הטעות בשכבה j

Classic IR + Classic CV (Content-Based Image Retrieval)

In the classic approach, searching for images

("Find me images that look like this sunset")

is treated as an Information Retrieval problem.

Bag of Visual Words

The Concept: **"Bag of Visual Words" (BoVW)** or Global Color Histograms.

Just as Classic IR counts word frequencies (TF-IDF),

Classic CV counts Color Frequencies.

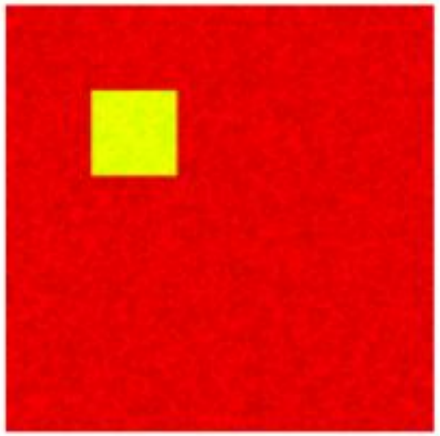
- **Word Dictionary** ≈ Color Palette (bins)
- **Document Vector** ≈ Color Histogram This technique, popularized by Swain & Ballard (1991), allows for incredibly fast image indexing.

  Demonstration: The Color Histogram Search Engine

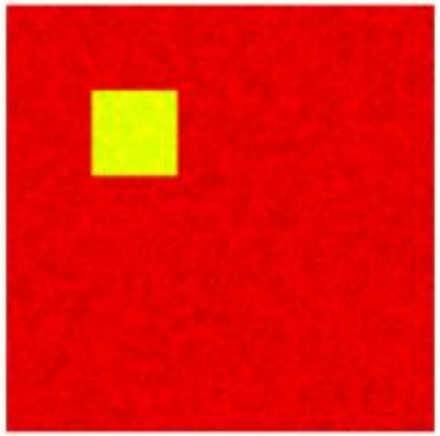
  We will create a mini-database of images,

  them by their color distribution, and run a "**Visual Query**."

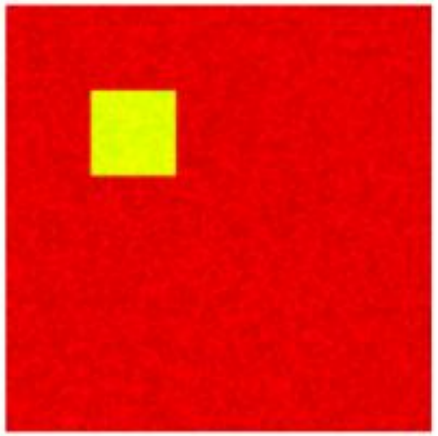
Query Image
(Red Dominant)



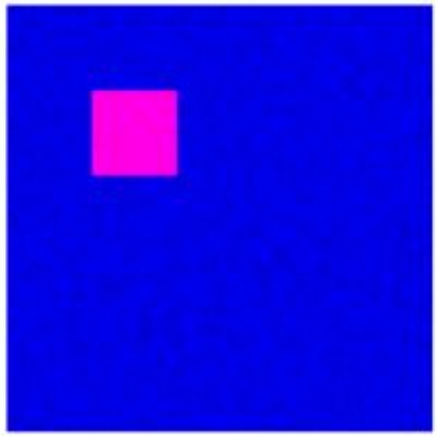
Rank 1: doc_2_apple
Sim: 0.998

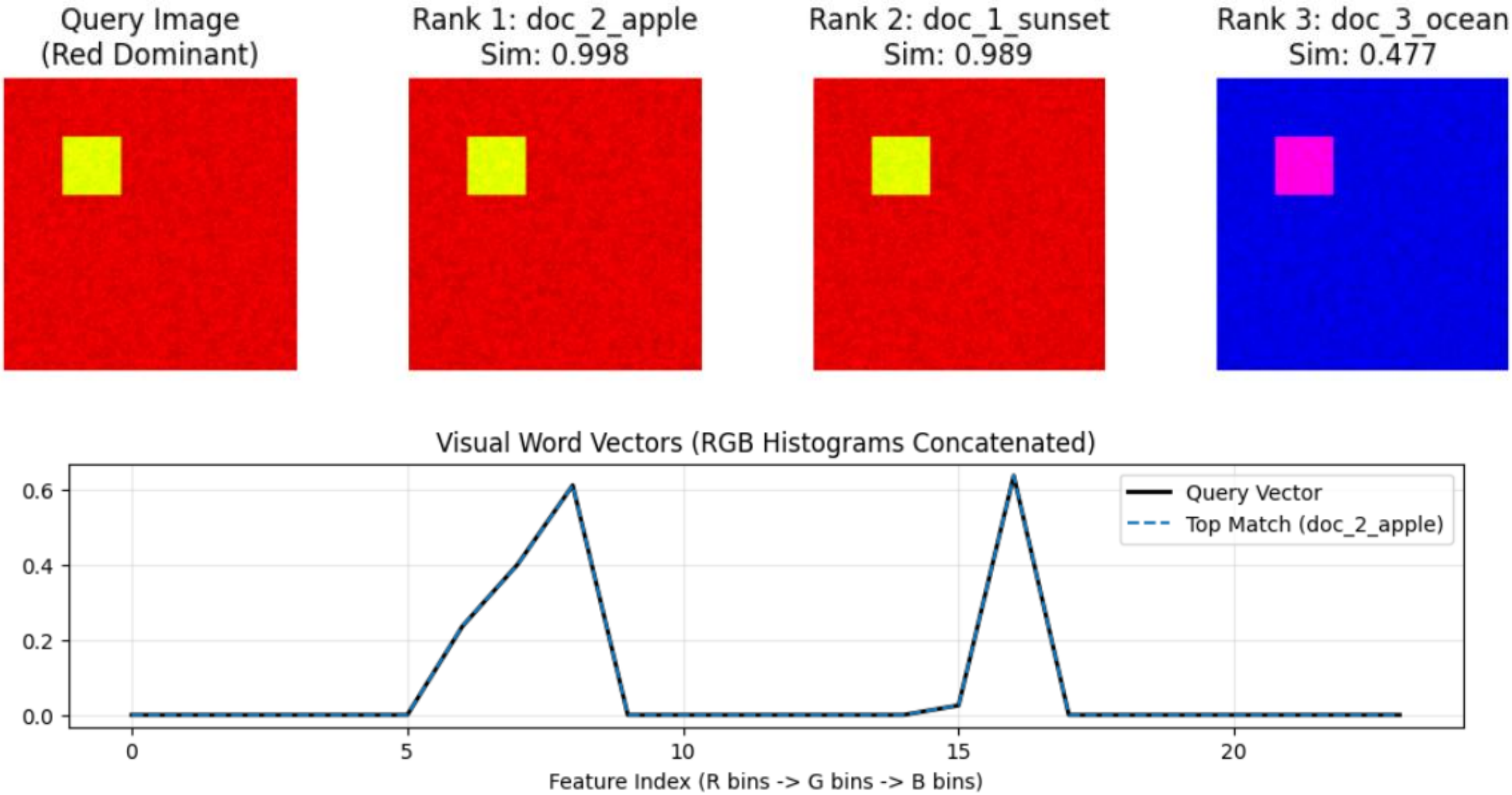


Rank 2: doc_1_sunset
Sim: 0.989



Rank 3: doc_3_ocean
Sim: 0.477




Query Image
(Red Dominant)
Rank 1: doc_2_apple
Sim: 0.998
Rank 2: doc_1_sunset
Sim: 0.989
Rank 3: doc_3_ocean
Sim: 0.477
Visual Word Vectors (RGB Histograms Concatenated)
Query Vector
Top Match (doc_2_apple)
0.6
0.4
0.2
0.0
0
5
10
15
20
Feature Index (R bins -> G bins -> B bins)

Understanding the IR+CV Flow:

1. **Feature Vector**: Look at the bottom chart. The "Vector" is just a wavy line representing how much Red, Green, and Blue is in the image.
    - **Red Section** (Left): High peaks (because the image is red).
    - **Blue/Green Sections** (Right): Low/Flat (mostly background noise).
2. **Indexing**: The database stores these vectors, not the pixels.
3. **Retrieval**: The system compares the Query Vector to the Database Vectors.
    - **Result**: The "Ocean" image (Blue) has a completely different vector shape, so it gets a low score. The "Sunset" and "Apple" images have similar vectors to the query, so they are ranked high.

This is the grandfather of modern Reverse Image Search. While modern systems use semantic vectors (Deep Learning), the logic of Vector Space Retrieval remains the same.

Classic Multimedia Information Retrieval - MMIR (Before Deep Learning)

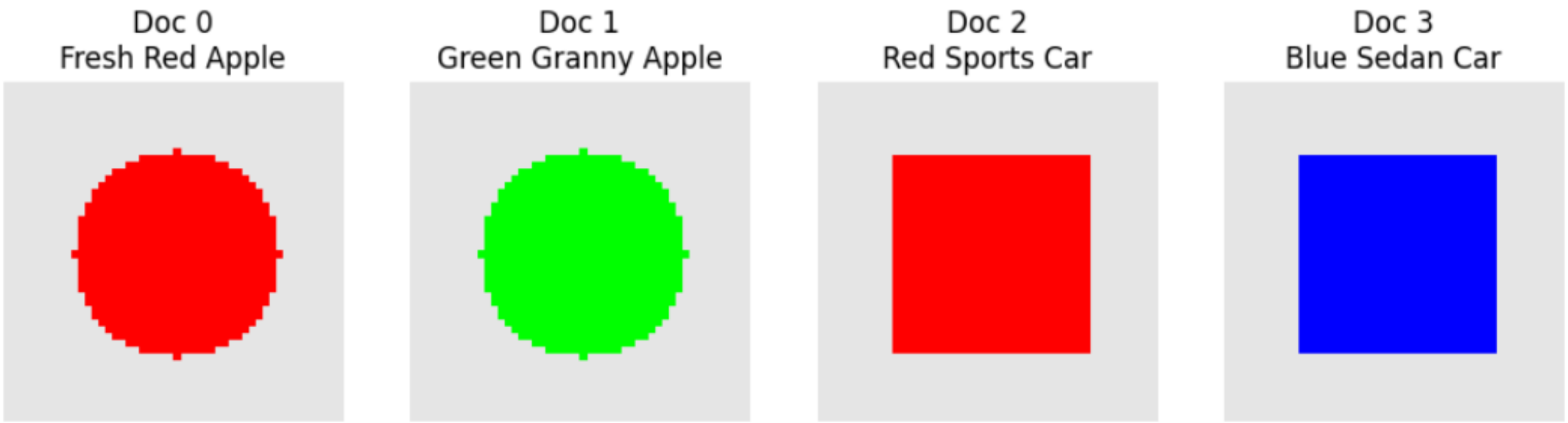
MMIR systems use Late Fusion.

1. **Text Engine**: Solved the semantic matching (e.g., "Find me a car").
2. **Vision Engine**: Solved the stylistic matching (e.g., "Find me something red").
3. **Fusion**: A weighted mathematical combination of the two scores.

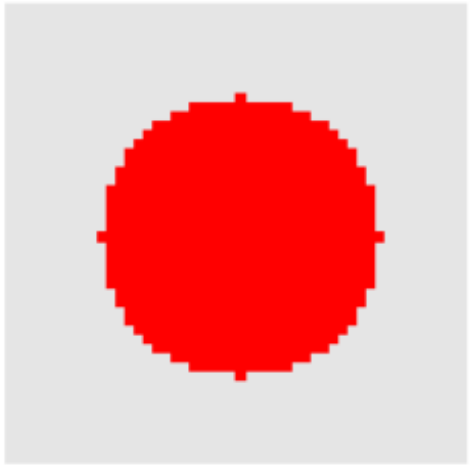
For example:

We build a "**Visual Reranking**" engine.

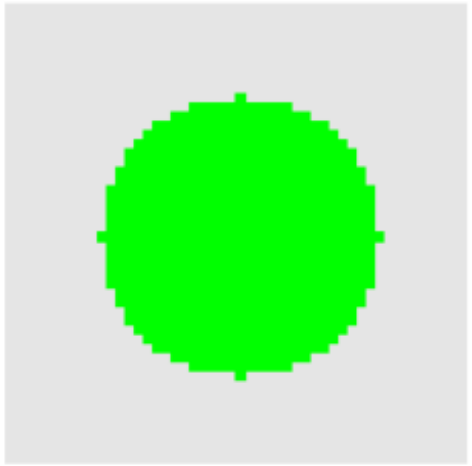
The user searches for "Apple", but provides a visual hint (a red color swatch) to imply they want Red Apples, not Green ones.


Doc 0
Fresh Red Apple
Doc 1
Green Granny Apple
Doc 2
Red Sports Car
Doc 3
Blue Sedan Car

Doc 0
Fresh Red Apple

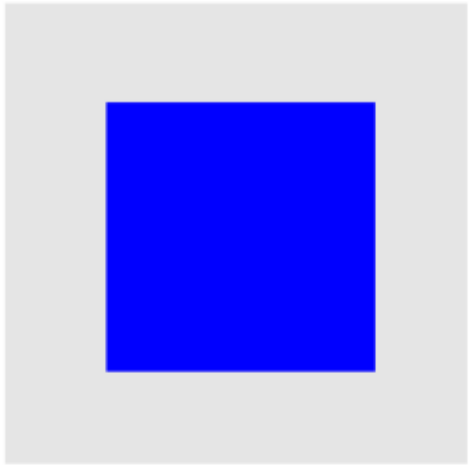


Doc 1
Green Granny Apple

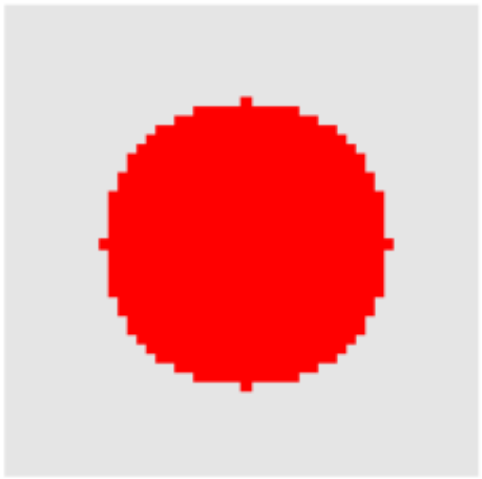


Doc 2
Red Sports Car

Doc 3
Blue Sedan Car



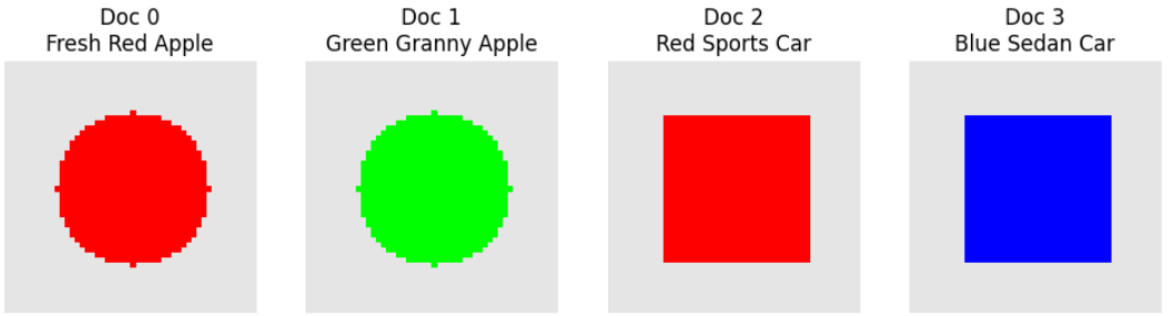
Query:
'Apple' + [Image]

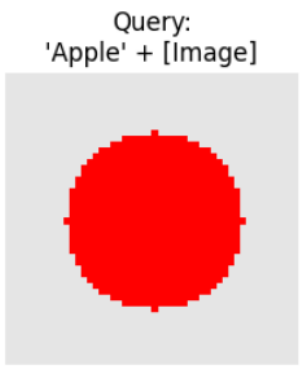


Step 2: The Text Pipeline (TF-IDF)

This handles the semantics.

It ensures that if the user asks for "Apple", we don't return a "Car" just because it's red.

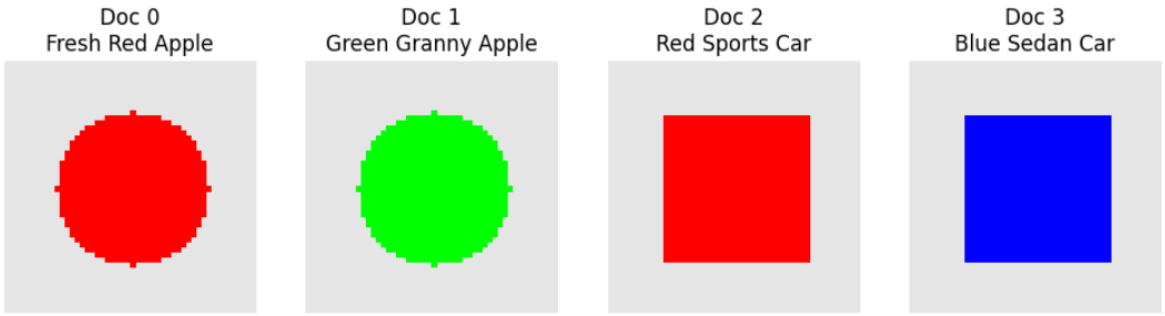




Step 2: The Text Pipeline (TF-IDF)

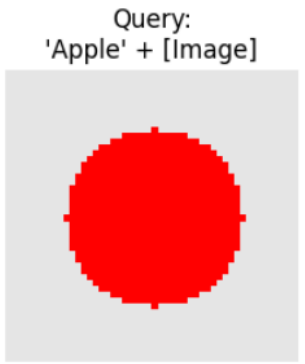
This handles the semantics.

It ensures that if the user asks for "Apple", we don't return a "Car" just because it's red.



--- Text Only Scores (TF-IDF) ---

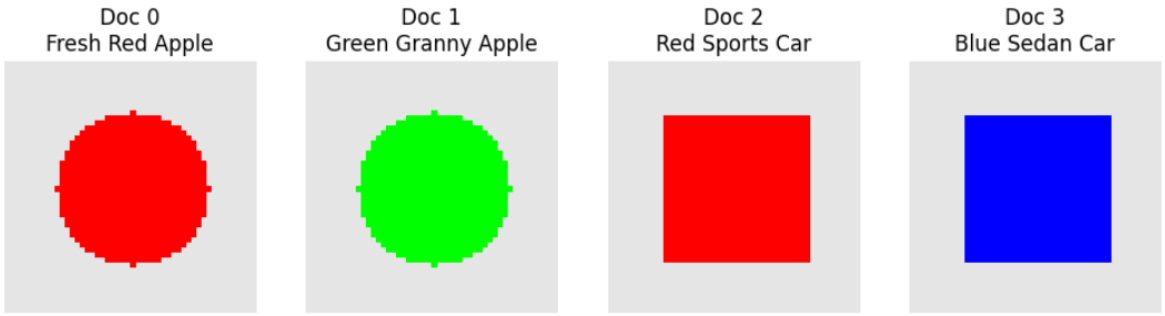
- Doc 0 (Fresh Red Apple): 0.5264
- Doc 1 (Green Granny Apple): 0.4869
- Doc 2 (Red Sports Car): 0.0000
- Doc 3 (Blue Sedan Car): 0.0000



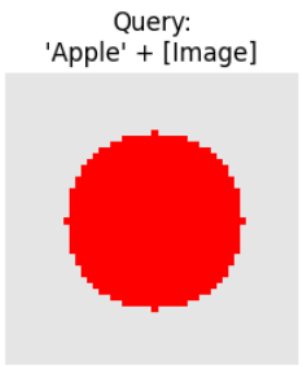
Step 2: The Text Pipeline (TF-IDF)

This handles the semantics.

It ensures that if the user asks for "Apple", we don't return a "Car" just because it's red.



Result so far:
The "Red Apple" and "Green Apple" likely have identical or very similar scores,
because they both contain the word "Apple".

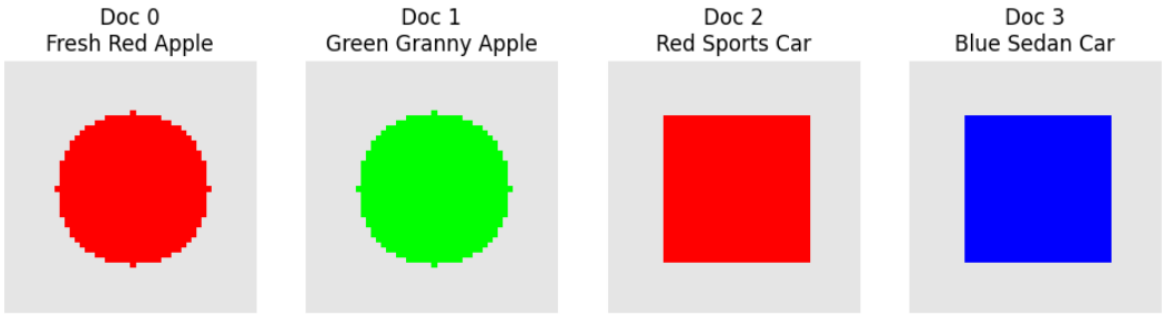


The system cannot distinguish the user's visual preference yet.

Step 3: The Vision Pipeline (Color Histograms)

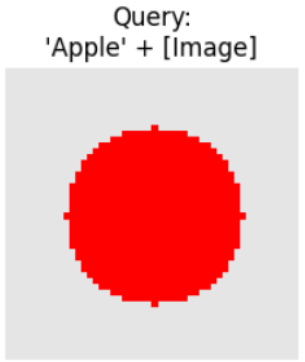
This handles the style/appearance.

We extract color features to fi



--- Visual Only Scores (Histogram) ---

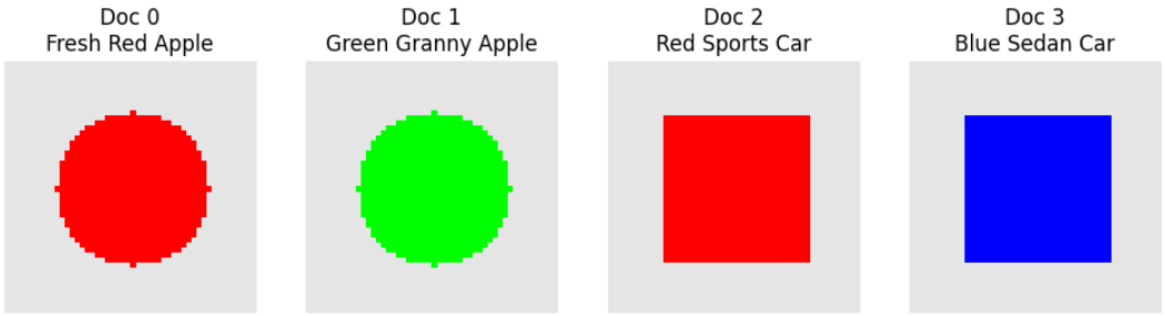
- Doc 0 (Fresh Red Apple): 1.0000
- Doc 1 (Green Granny Apple): 0.9265
- Doc 2 (Red Sports Car): 0.9976
- Doc 3 (Blue Sedan Car): 0.9087



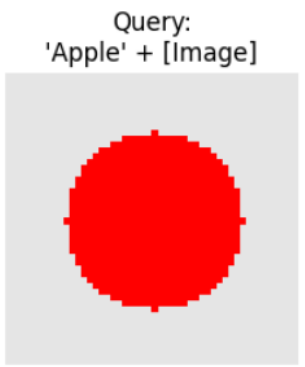
Step 3: The Vision Pipeline (Color Histograms)

This handles the style/appearance.

We extract color features to fi



Result so far:
The "Red Apple" and "Red Car" have high scores because they are red.



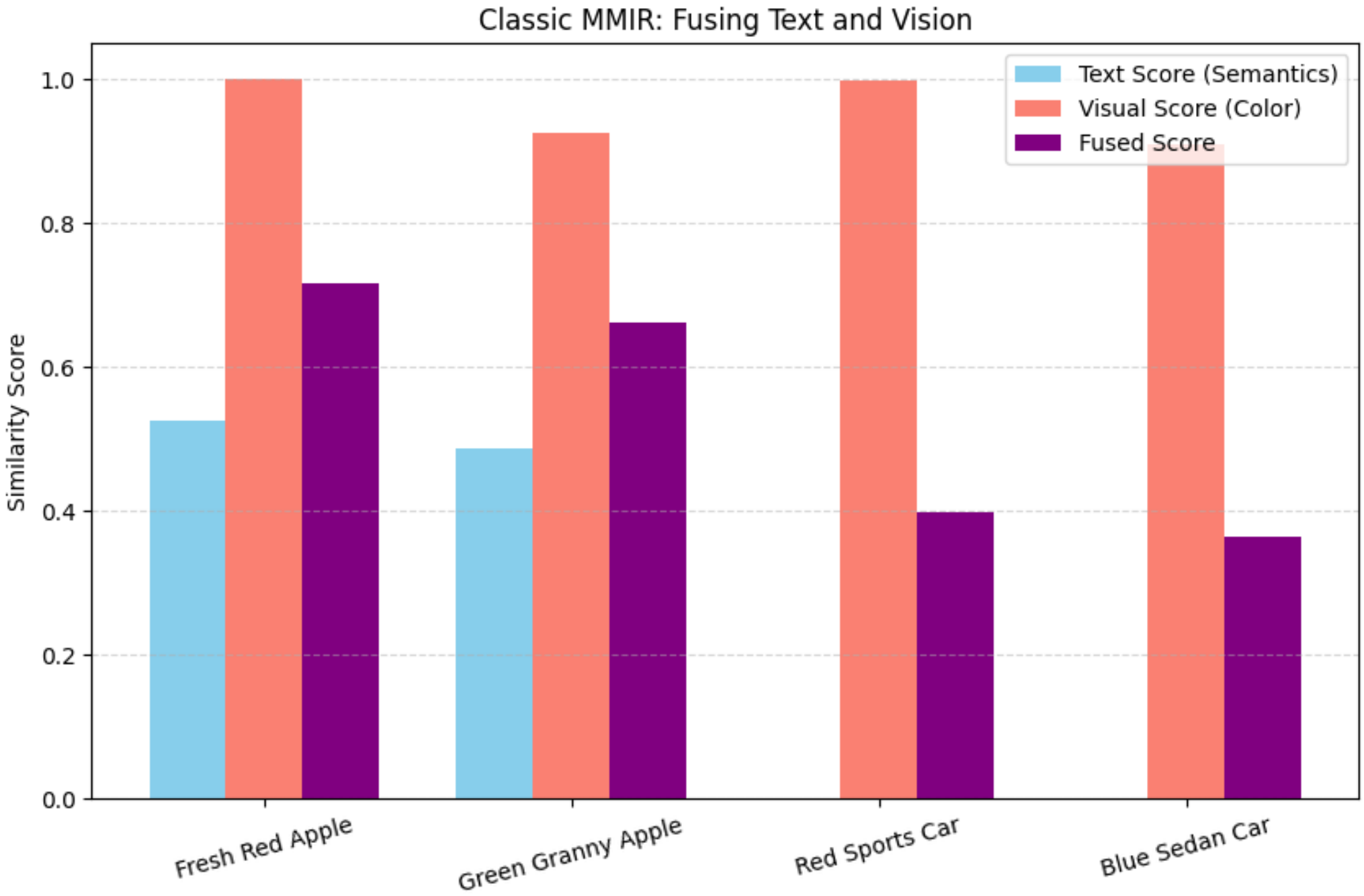
The "Green Apple" has a low score.
The vision system ignores the fact that a car isn't a fruit

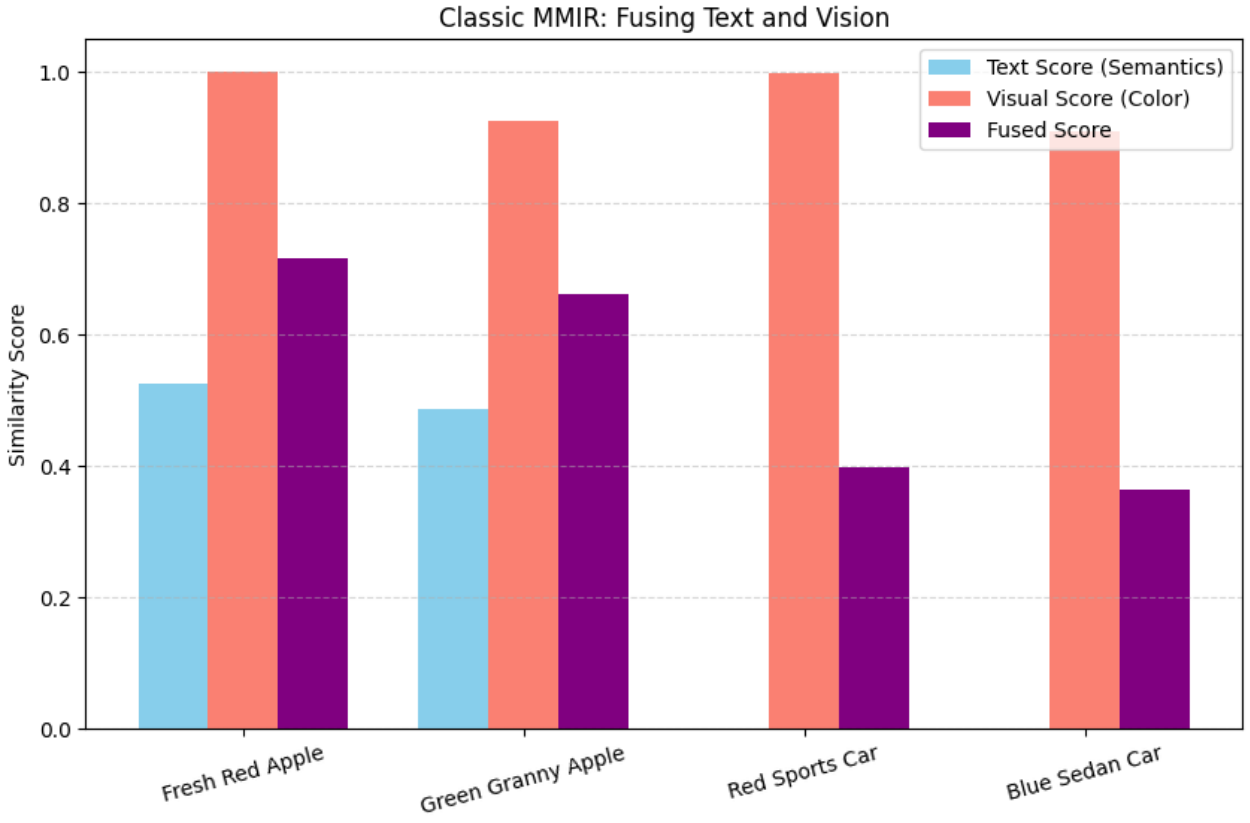
Step 4: Late Fusion (The "Mix")

We combine the scores using a weighting parameter (α).

$$\text{FinalScore} = \alpha \cdot \text{TextScore} + (1 - \alpha) \cdot \text{VisualScore}$$

In this scenario, we usually weight Text higher (to get the right object)
and use Vision to rerank (to get the right variant).


Classic MMIR: Fusing Text and Vision
Text Score (Semantics)
Visual Score (Color)
Fused Score
Similarity Score
1.0
0.8
0.6
0.4
0.2
0.0
Fresh Red Apple
Green Granny Apple
Red Sports Car
Blue Sedan Car



- --- Final Ranked Results ---
- Rank 1: Fresh Red Apple (Score: 0.7158)
- Rank 2: Green Granny Apple (Score: 0.6627)
- Rank 3: Red Sports Car (Score: 0.3990)
- Rank 4: Blue Sedan Car (Score: 0.3635

Analysis of the Plot

1. Red Apple (The Winner):
    - Text (Blue Bar): High. Matches "Apple".
    - Visual (Red Bar): High. Matches Red color.
    - Result: Highest Purple bar.
2. Green Granny Apple:
    - Text (Blue Bar): High. Matches "Apple".
    - Visual (Red Bar): Low. It's green!
    - Result: Pushed down to 2nd place. This is the "Visual Reranking" working.
3. Red Sports Car:
    - Text (Blue Bar): Zero. "Car" != "Apple".
    - Visual (Red Bar): High. It is red.
    - Result: Even though it looks right, the text filter kills it.

      This proves the system respects semantics.

Summary of the "Classic Mixed Flow"

- **Independence**: The text engine and the computer vision engine ran completely separately.
  They didn't "know" about each other.
- **Normalization**: A major challenge in classic systems was scaling.
  If Text scores ranged 0-100 and Visual scores ranged 0-1, the fusion would fail.
  We used Cosine Similarity here (always 0-1) to solve this.
- **The "Semantic Gap"**: Notice the system didn't know the Red Car was a "vehicle".
  It just knew it was "Red".
  Classic CV described signals (color, texture), not concepts.

Why study perception?

- **Perception is messy: Can we avoid it?**

  No!

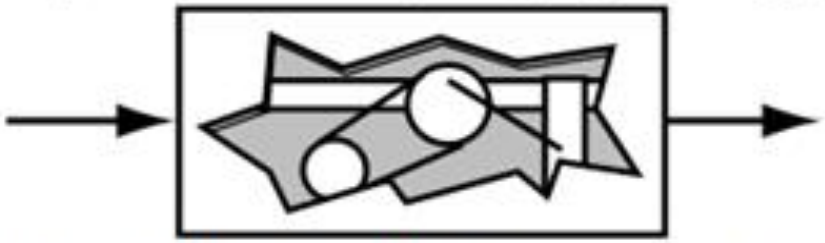
- **Audition provides the 'ground truth' in audio**
  - what is relevant and irrelevant
  - subjective importance of distortion

- **Some sounds are 'designed' for audition**
  - co-evolution of speech and hearing
- **The auditory system is very successful**
  - we would do extremely well to duplicate it

How to study perception?

Three different approaches:

- **Analyze the example: physiology**



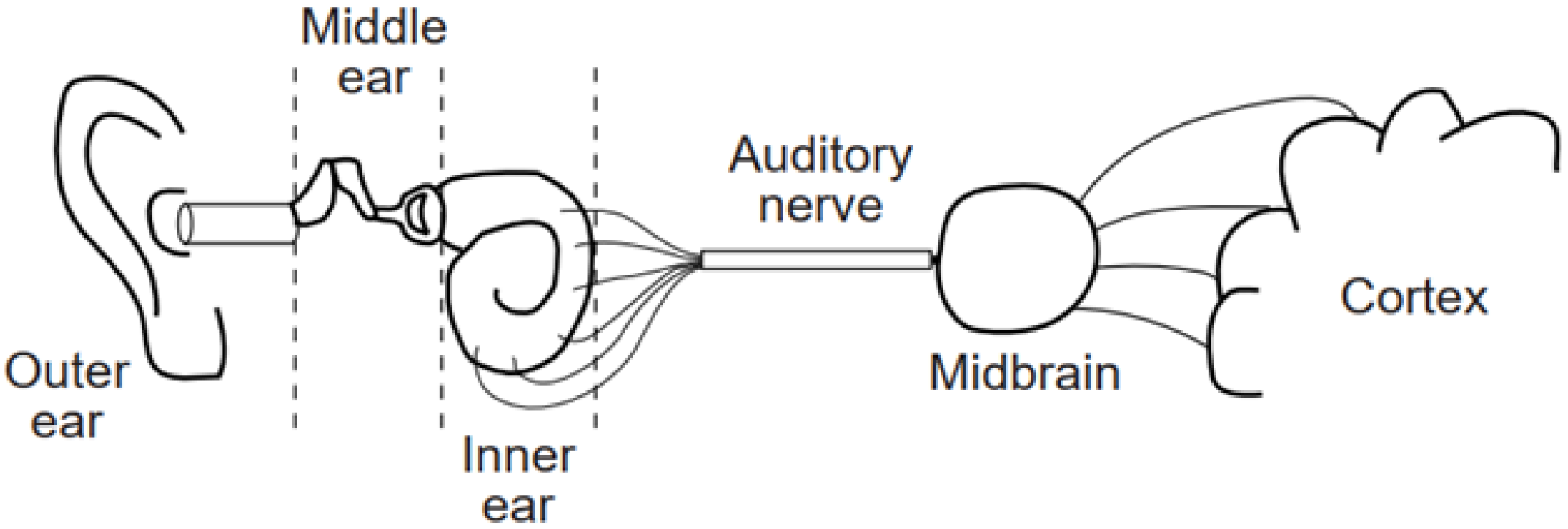
  - dissection & nerve recordings

- **Black box input/output: psychophysics**

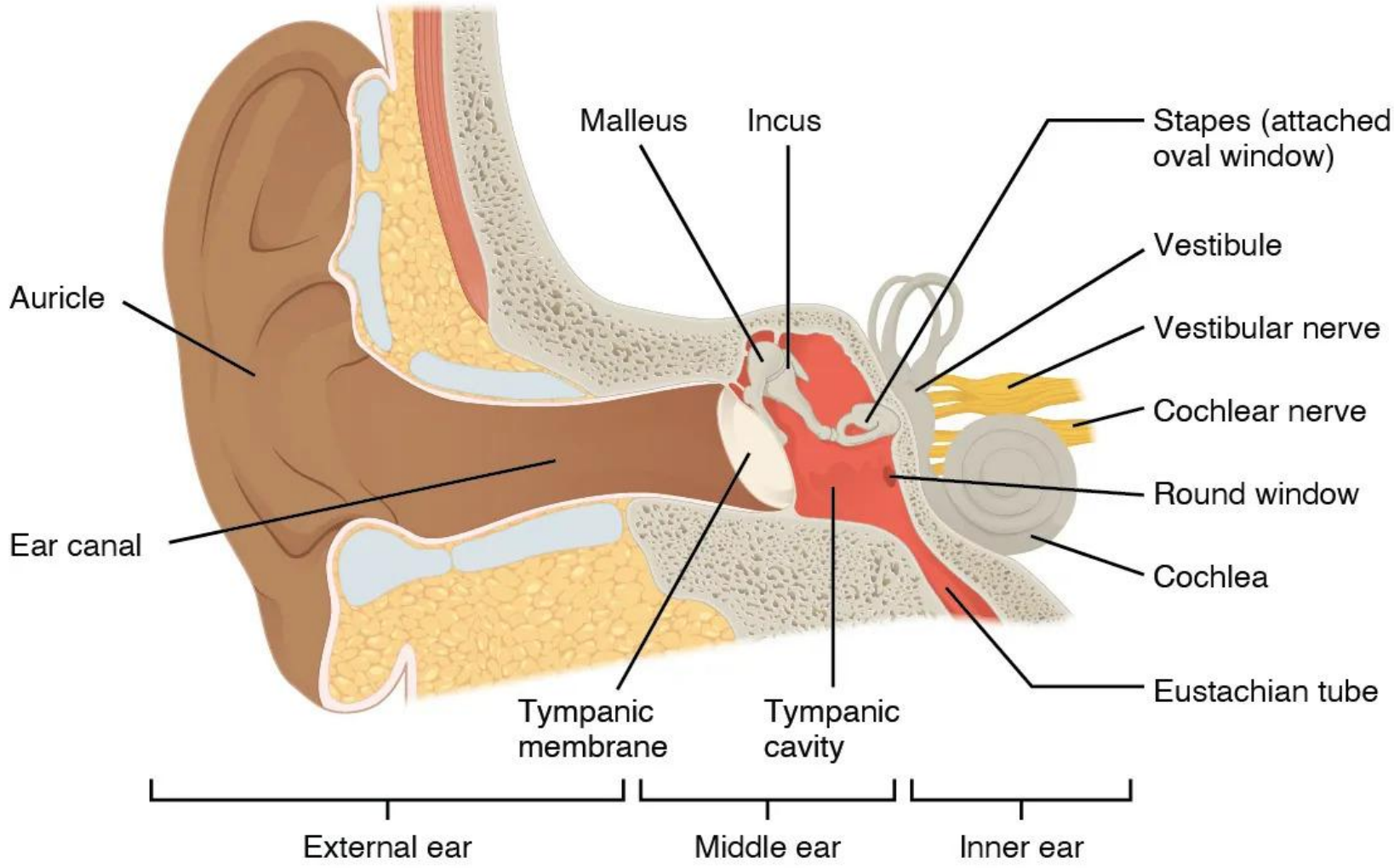
  - fit simple models of simple functions

- **Information processing models**
  - investigate and model complex functions
  - e.g. scene analysis, speech perception

- **Processing chain from air to brain:**



Structures of the Ear

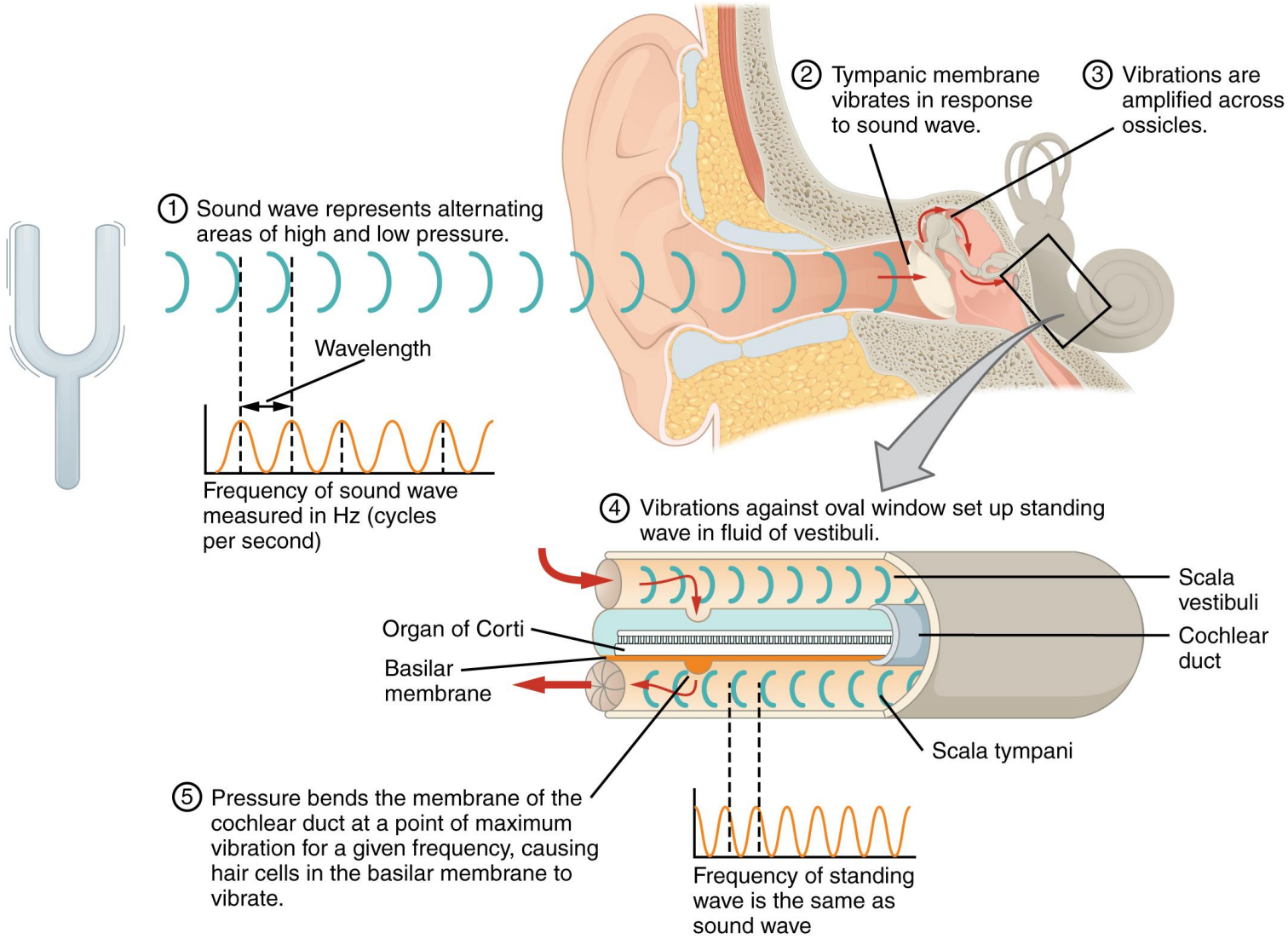


<u>The external ear</u> contains the auricle, ear canal, and tympanic membrane.

<u>The middle ear</u> contains the ossicles and is connected to the pharynx by the Eustachian tube.

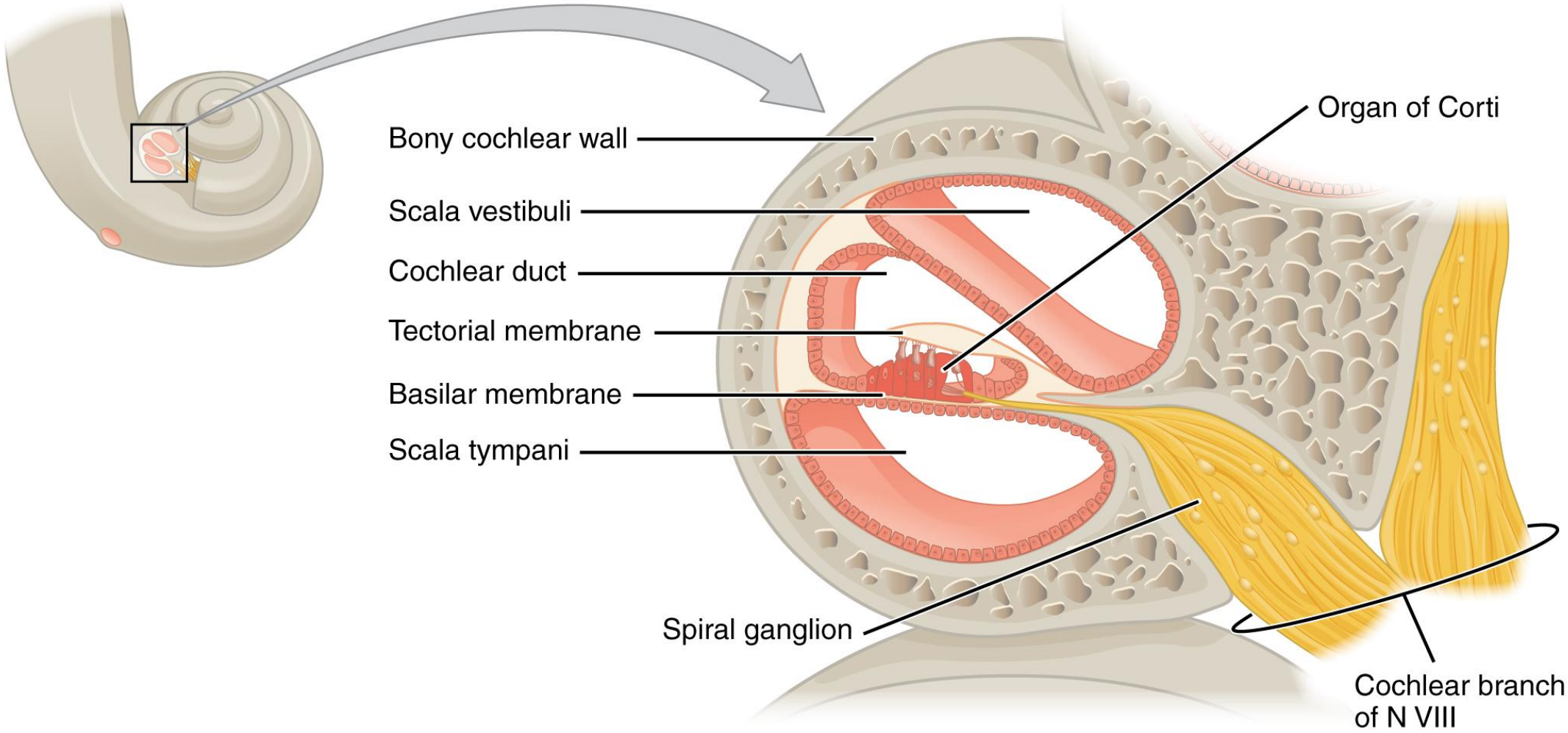
<u>The inner ear</u> contains the cochlea and vestibule, which are responsible for audition and equilibrium, respectively

Transmission of Sound Waves to Cochlea

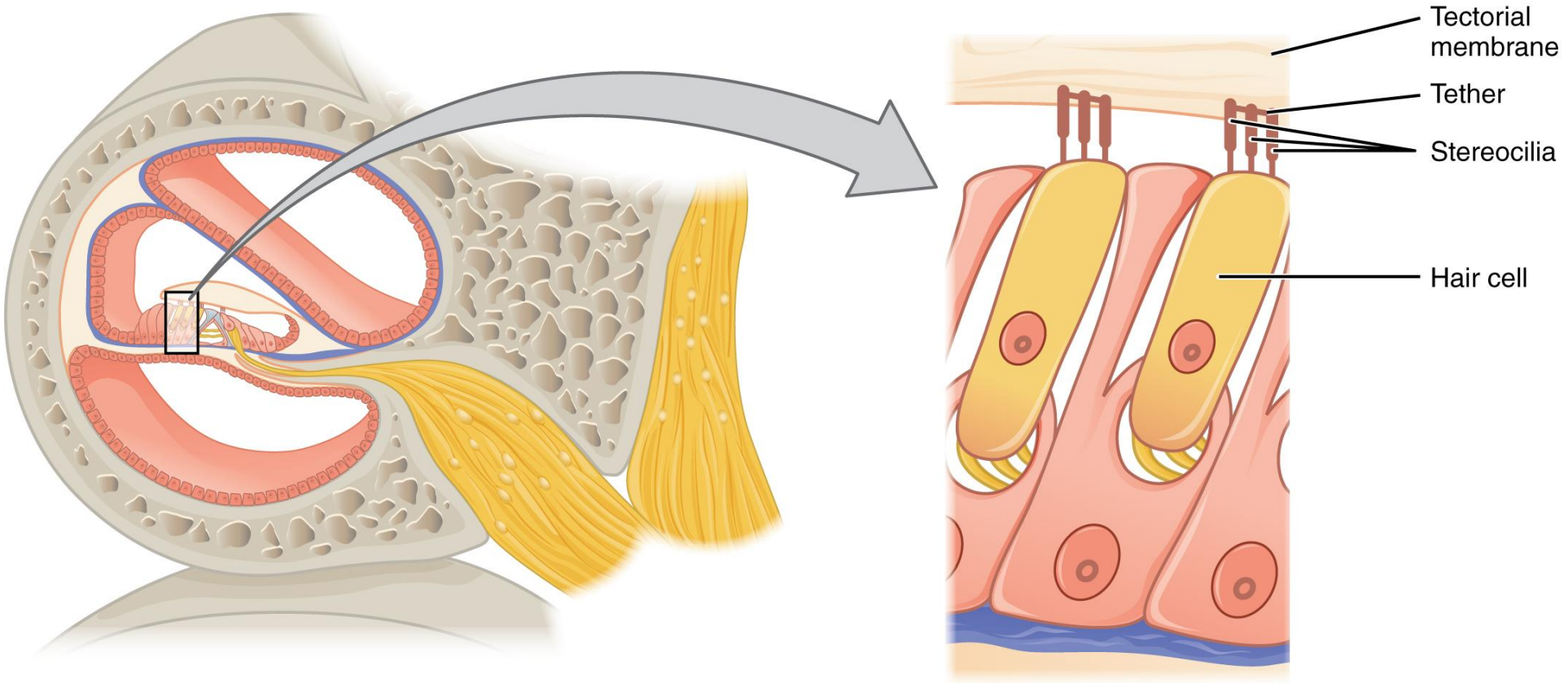


- *A sound wave causes the tympanic membrane to vibrate.*
- *This vibration is amplified as it moves across the malleus, incus, and stapes.*
- *The amplified vibration is picked up by the oval window causing pressure waves in the fluid of the scala vestibuli and scala tympani.*
- *The complexity of the pressure waves is determined by the changes in amplitude and frequency of the sound waves entering the ear.*

Cross Section of the Cochlea



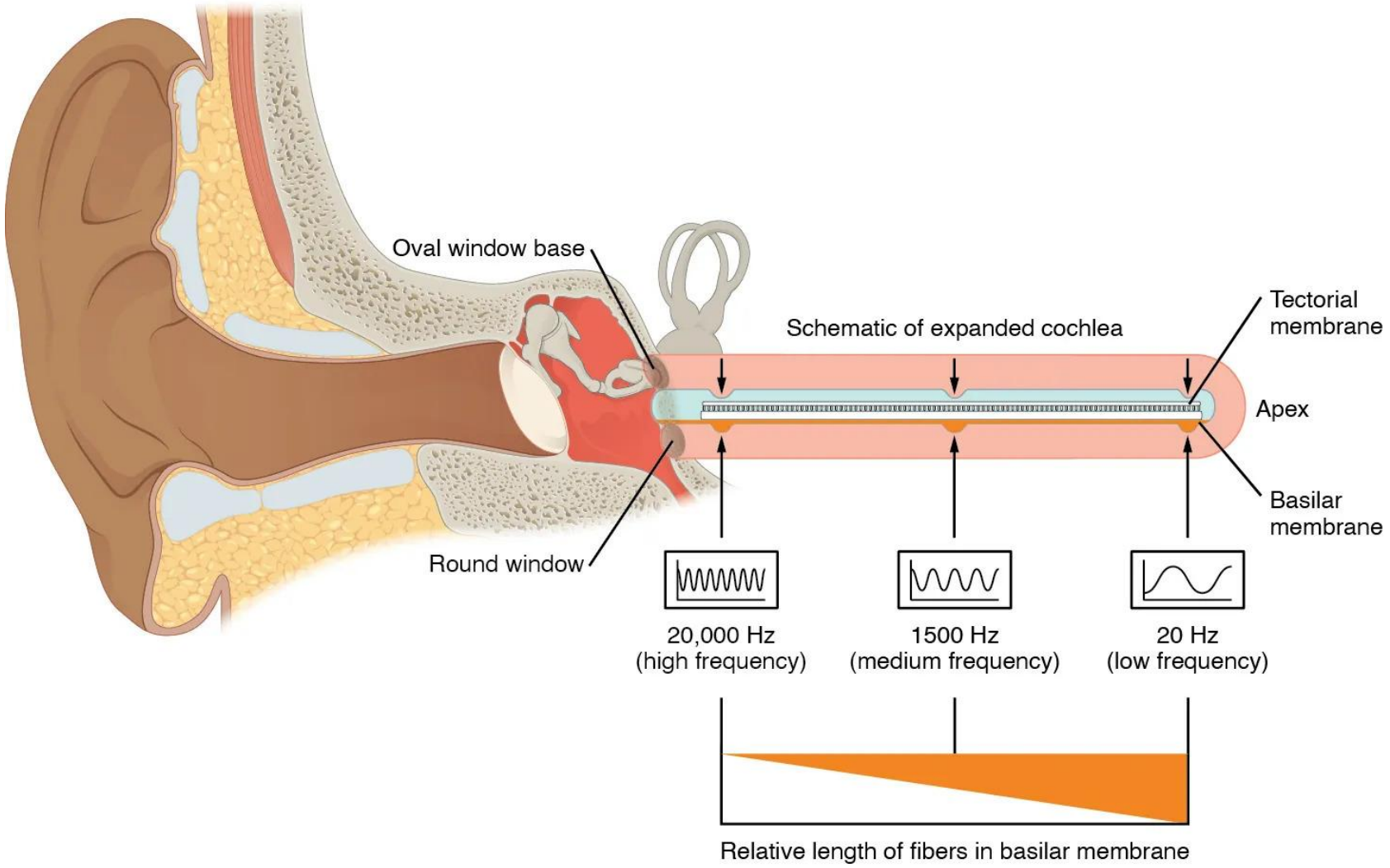
Hair Cell



The hair cell is a mechanoreceptor with an array of stereocilia emerging from its apical surface.

The stereocilia are tethered together by proteins that open ion channels when the array is bent toward the tallest member of their array, and closed when the array is bent toward the shortest member of their array.

Frequency Coding in the Cochlea



- *The standing sound wave generated in the cochlea by the movement of the oval window deflects the basilar membrane on the basis of the frequency of sound.*
- *Therefore, hair cells at the base of the cochlea are activated only by high frequencies, whereas those at the apex of the cochlea are activated only by low frequencies.*

Until the next time ☺