Lecture 8-1 Perceptron

Machine Learning

Spring Semester ‘2022

2^{nd} Half Course Introduction

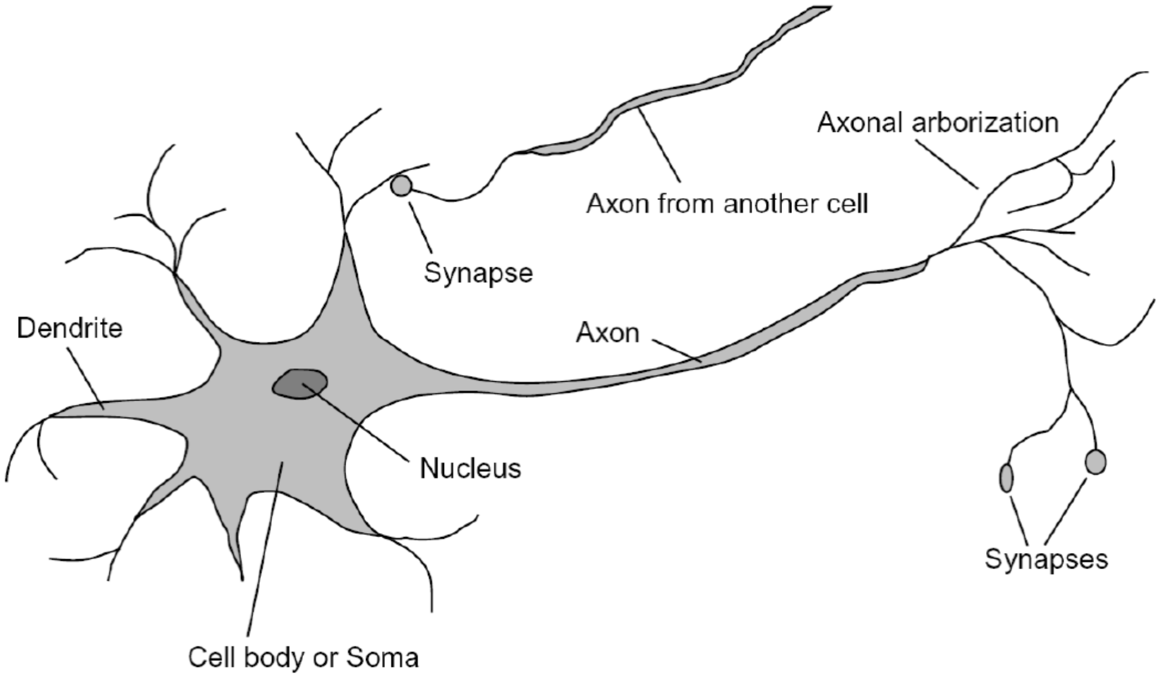
- Instructor: Hyukjoon Lee
  hlee@kw.ac.kr
  새빛관 707
  Tel: 940-5127
  Office Hours: T,W 2-3 pm
- Class Hour: T 4:30–5:45 pm (recorded video), Th 3:00–4:15 pm
- Class Room: 새빛관 202
- Course Homepage: mclab.kw.ac.kr
- Required Text:
  - Lecture slides
- Recommended References:
  - Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, Springer, 2016

Machine learning overview

x	$f(x)$	y
Hello, Do you want free printr cartriges? Why pay more when you can get them ABSOLUTELY FREE! Just	# free : 2 YOUR_NAME : 0 MISSPELLED : 2 FROM_FRIEND : 0 ...	SPAM or +
	PIXEL-7,12 : 1 PIXEL-7,13 : 0 ... NUM_LOOPS : 1 ...	“2”

Biological Model

- Human neurons

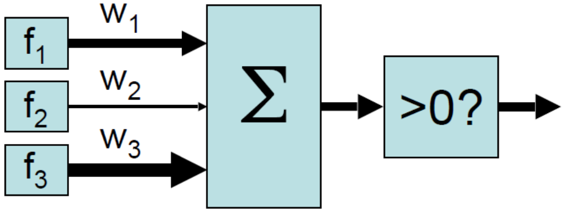


Linear classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation

$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

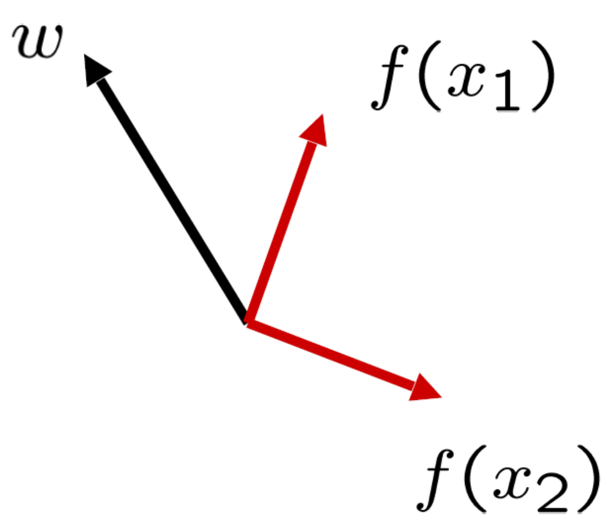
- If the activation is:
  - Positive, output +1
  - Negative, output -1



Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

```
# free      :  4
YOUR_NAME   : -1
MISSPELLED  :  1
FROM_FRIEND : -3
...
```



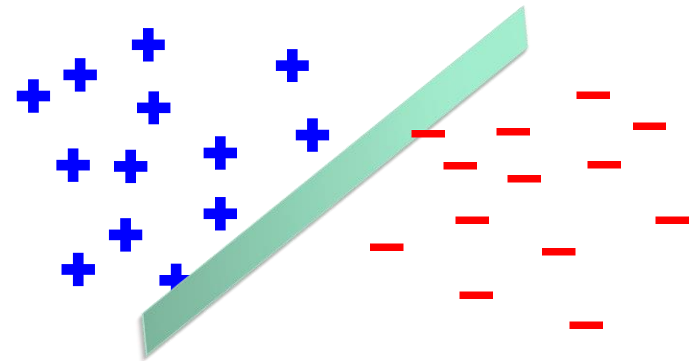

```
# free      : 2
YOUR_NAME   : 0
MISSPELLED  : 2
FROM_FRIEND : 0
...
```

```
# free      : 0
YOUR_NAME   : 1
MISSPELLED  : 1
FROM_FRIEND : 1
...
```

Dot product $w \cdot f$ *positive means the positive class*

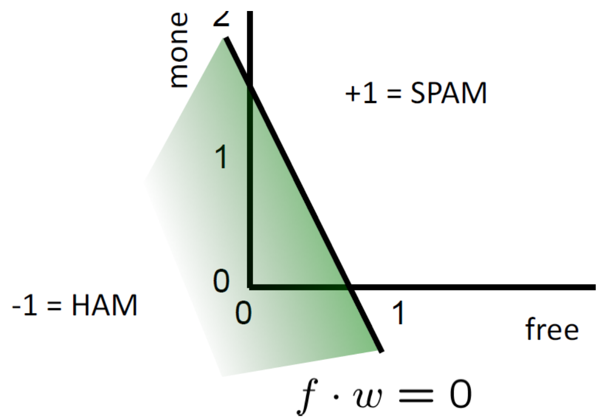
Binary decision rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



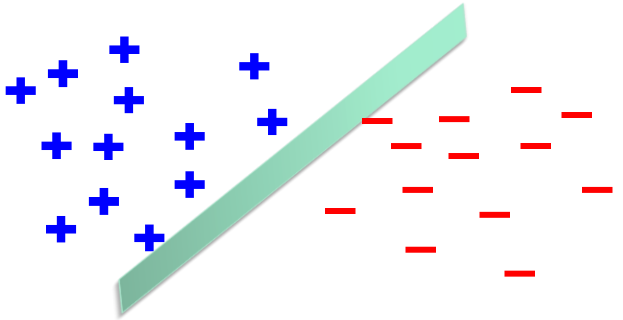
w

```
BIAS  : -3
free  :  4
money :  2
...
```

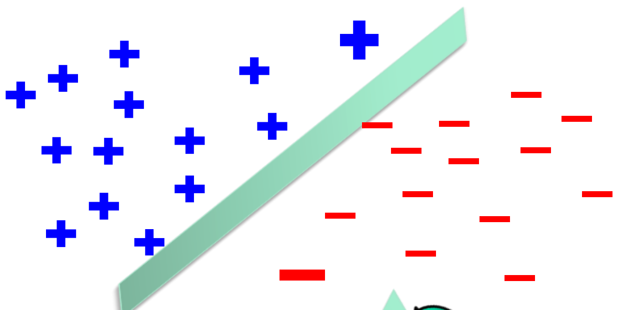


Learning: binary perceptron

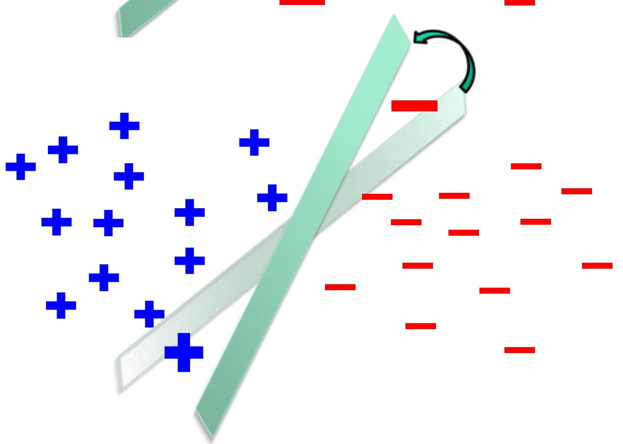
- Start with weights = 0
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



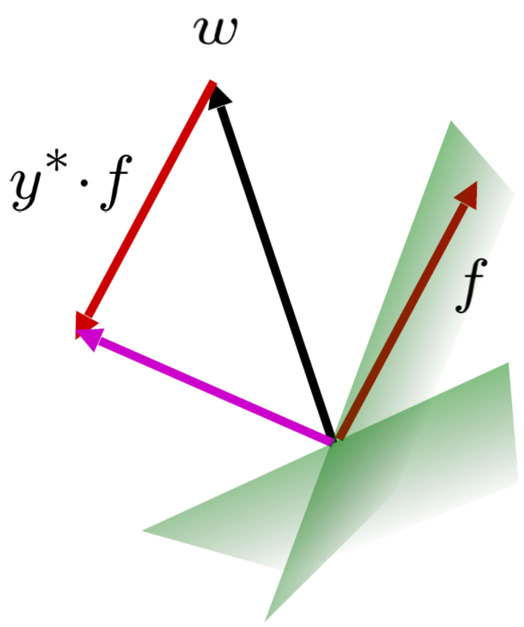
Learning: binary perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } \; w \cdot f(x) \geq 0 \\ -1 & \text{if } \; w \cdot f(x) < 0 \end{cases}$$

  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

$$w = w + y^* \cdot f$$



Multiclass decision rule

- If we have multiple classes:
  - A weight vector for each class:

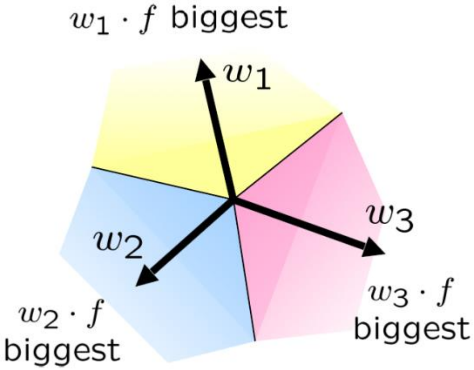
$$w_y$$

  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - Prediction highest score wins

$$y = \underset{y}{\arg\max}\ w_y \cdot f(x)$$



Binary = multiclass where the negative class has weight zero

Learning: multiclass perceptron

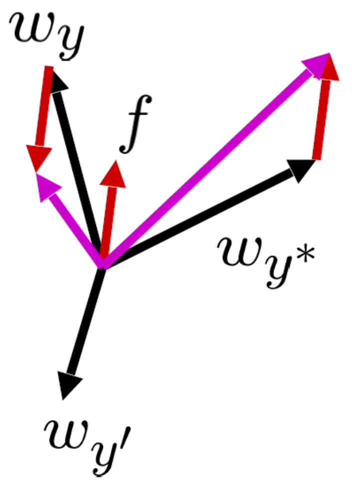
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$



Example: multiclass perceptron (1)

"win the vote"

"win the election"

"win the game"

w_{SPORTS}		$w_{POLITICS}$		w_{TECH}	
BIAS	1	BIAS	0	BIAS	0
win	0	win	0	win	0
game	0	game	0	game	0
vote	0	vote	0	vote	0
the	0	the	0	the	0
...		...		...	

Example: multiclass perceptron (2)

w_s: $[1\ 0\ 0\ 0\ 0]^T$ w_p: $[0\ 0\ 0\ 0\ 0]^T$ w_t: $[0\ 0\ 0\ 0\ 0]^T$

x_1: "win the vote" → $[1\ 1\ 0\ 1\ 1]^T$ y='politics' (p)

x_2: "win the election" → $[1\ 1\ 0\ 0\ 1]^T$ y='politics' (p)

x_3: "win the game → $[1\ 1\ 1\ 0\ 1]^T$ y='sports' (s)

$w_s \cdot x_1 = 1$, $w_p \cdot x_1 = 0$, $w_t \cdot x_1 = 0$, y=s, y*=p ➔ incorrect!

➔ $w_p = [0\ 0\ 0\ 0\ 0]^T + [1\ 1\ 0\ 1\ 1]^T = [1\ 1\ 0\ 1\ 1]^T$

➔ w_s: $[1\ 0\ 0\ 0\ 0]^T - [1\ 1\ 0\ 1\ 1]^T = [0\ {-1}\ 0\ {-1}\ {-1}]^T$

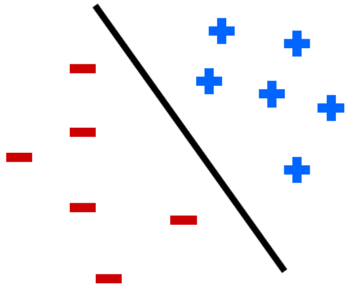
$w_s \cdot x_2 = 1$

Properties of perceptrons

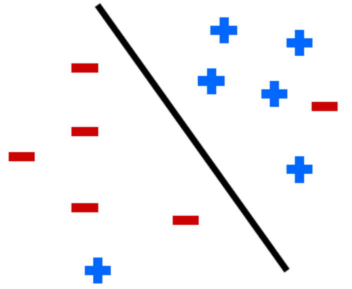
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

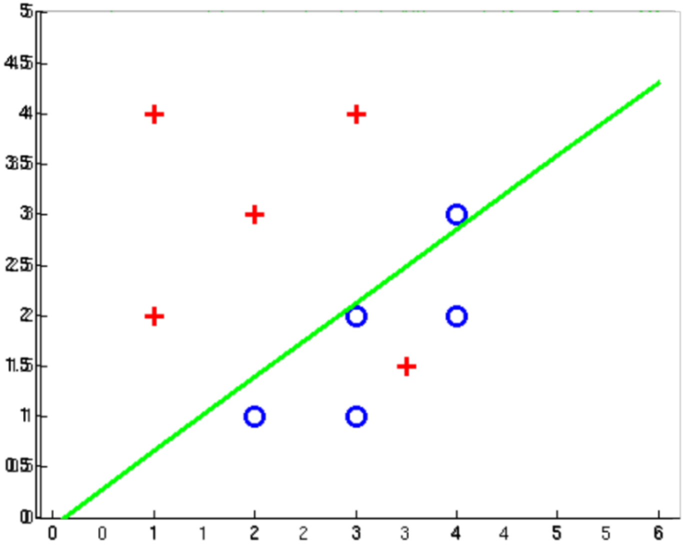


Non-Separable



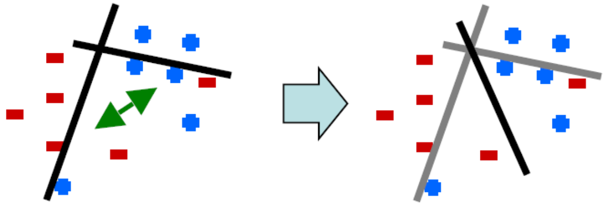
Examples: perceptron

- Non-Separable Case

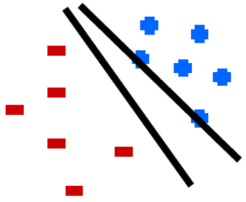


Problems with the perceptron

- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



- Mediocre generalization: finds a "barely" separating solution



- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting