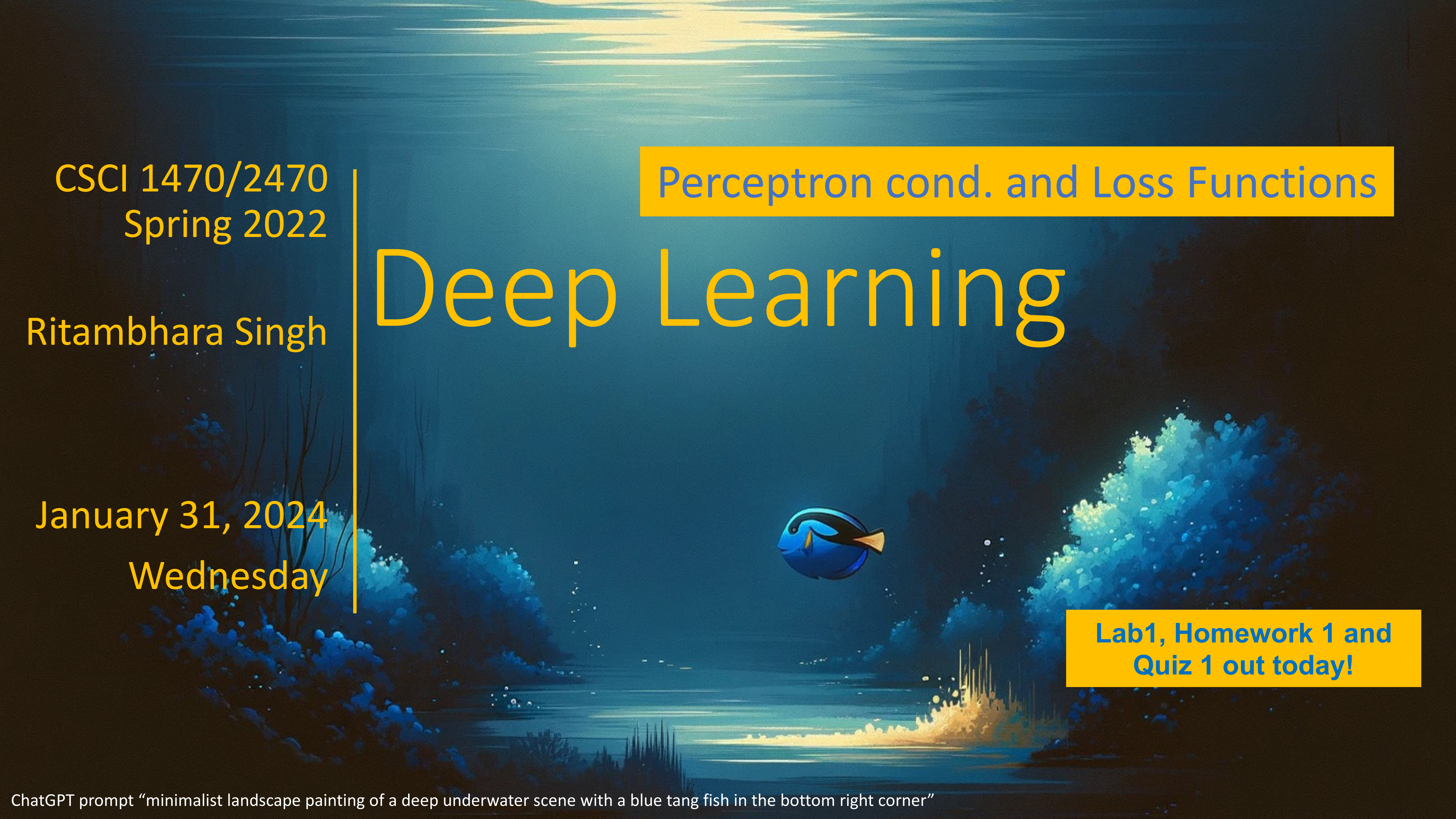


Perceptron cond. and Loss Functions

Deep Learning

CSCI 1470/2470
Spring 2022

Ritambhara Singh

January 31, 2024
Wednesday

Lab1, Homework 1 and Quiz 1 out today!

ChatGPT prompt “minimalist landscape painting of a deep underwater scene with a blue tang fish in the bottom right corner”

Today’s goal – Continue discussion on perceptron and learn about the loss functions

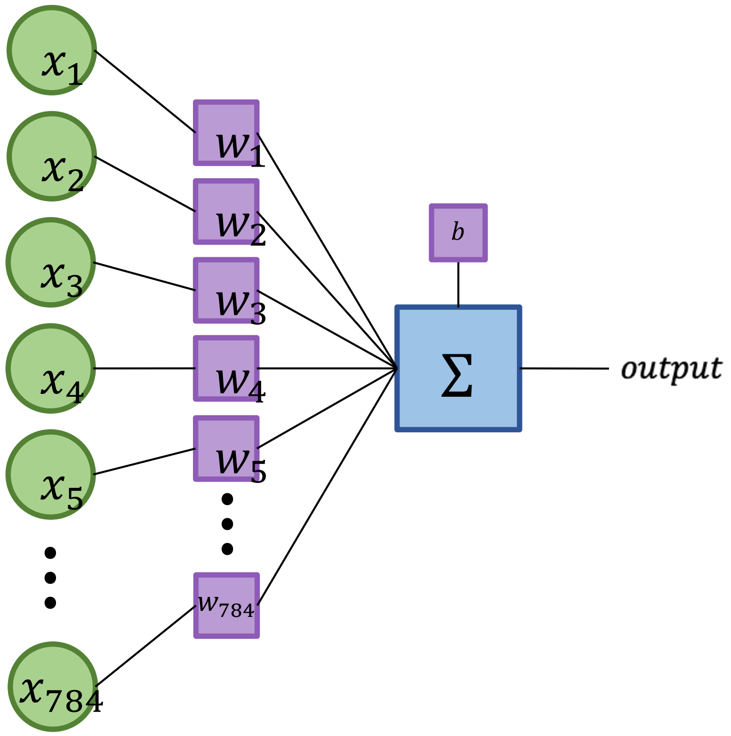
(1) Perceptron learning algorithm

(2) Extending perceptron for Multi-class classification

(3) Loss functions for
- Regression
- Classification

Recap: A Binary Perceptron for MNIST

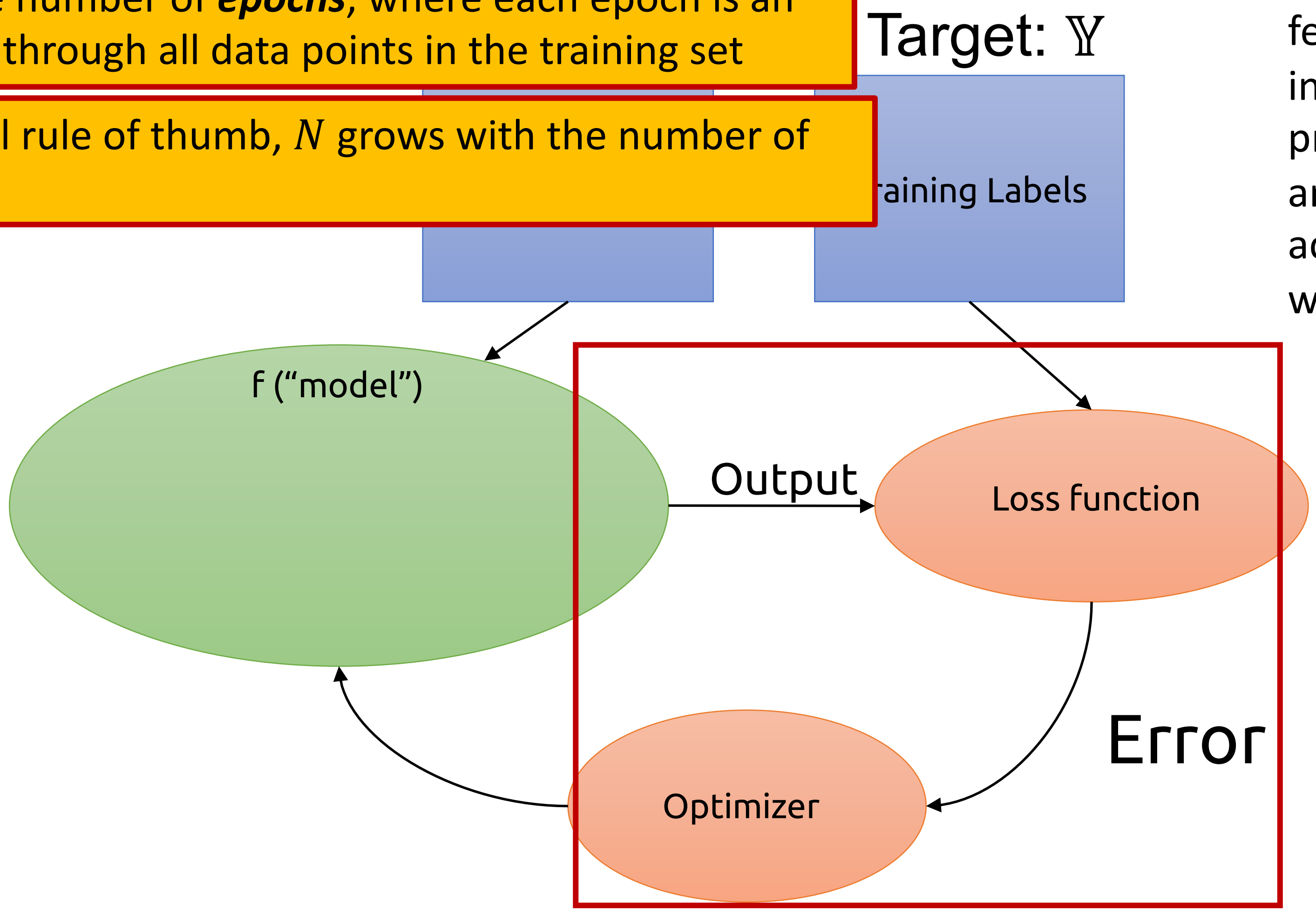
- *Inputs* $[x_1, x_2, \ldots x_n]$ are all positive
  - $n = 784$ (28×28 pixel values)
- *output* is either 0 or 1
  - 0 → input is not the digit type we're looking for
  - 1 → input ***is*** the digit type we're looking for



Training a perceptron

N is known as the number of ***epochs***, where each epoch is an iteration of going through all data points in the training set

As a general rule of thumb, N grows with the number of parameters



0. set the parameters $\Phi = \{w \cup b\}$ to 0

1. Iterate over training set several times,
   feeding in each training example into the model,
   producing an output,
   and adjusting the parameters according to whether that output was right or wrong

2. Stop once we either
   (a) get every training example right or
   (b) after N iterations, a number set by the programmer.

The Perceptron Learning Algorithm

1. set w's to 0.
2. for N iterations, or until the weights do not change:
   a) for each training example $\mathbf{x}^k$ with label y^k
      i. if $y^k - f\left(\mathbf{x}^k\right) = 0$ continue
      ii. else for all weights w_i, $\Delta w_i = \left(y^k - f\left(\mathbf{x}^k\right)\right) x_i^k$

- $b = \text{bias}$
- $w = \text{weights}$
- $N \ = \text{maximum number of training iterations}$
- $\mathbf{x}^k = \mathrm{k}^{\mathrm{th}}$ training example
- y^k = label for the k^{th}example
- w_i = weight for the i^{th} input where $i \leq n$
- n = number of pixels per image
- x_i^k = i^{th} input of the example where $i \leq n$

The Perceptron Learning Algorithm

1. set w's to 0.
2. for N iterations, or until the weights do not change:
    a) for each training example $\mathbf{x}^k$ with label y^k
        i. **if $\boldsymbol{y^k - f(\mathbf{x}^k) = 0}$ continue**
        ii. else for all weights w_i, $\Delta w_i = \left(y^k - f\left(\mathbf{x}^k\right)\right) x_i^k$

- If the output of our model matches the label, we continue
- If the correct label is 1, and our output is 1, $1 - 1 = 0$
- If the correct label is 0, and our output is 0, $0 - 0 = 0$

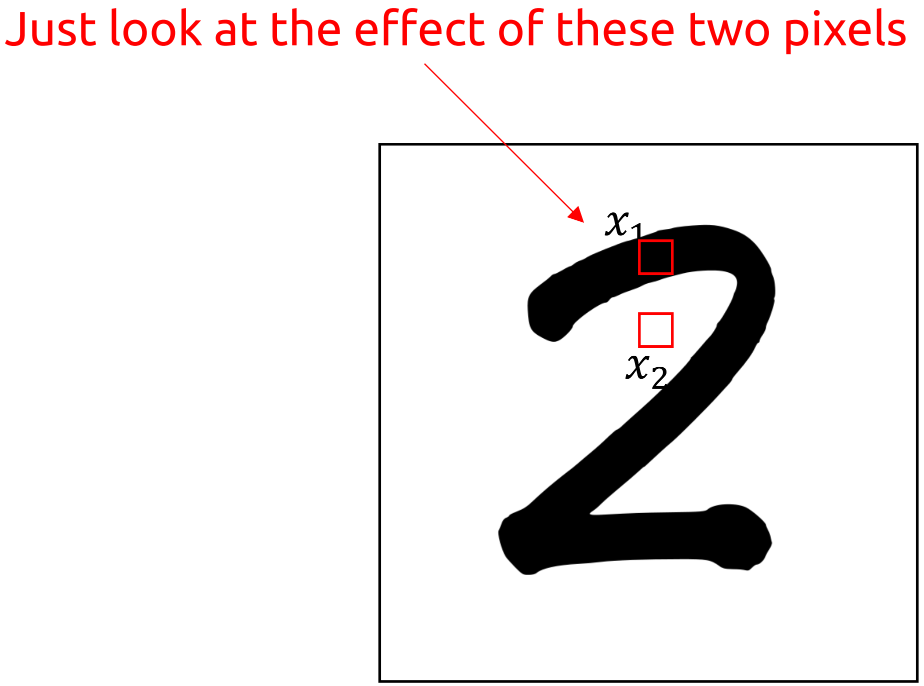
The Perceptron Learning Algorithm

1. set w's to 0.
2. for N iterations, or until the weights do not change:
   a) for each training example $\mathbf{x}^k$ with label y^k
      i. if $y^k - f(\mathbf{x}^k) = 0$ continue
      ii. **else for all weights w_i, $\Delta w_i = \left(y^k - f(\mathbf{x}^k)\right) x_i^k$**

- If our label y^k is a 1, and our model's output is a 0, we update the i^{th} weight by:
  - $(1 - 0) \cdot x_i^k = x_i^k$
  - Output was 0 and should have been 1, so make the output more positive
- If our label y^k is a 0, and our model's output is a 1, we update the i^{th} weight by:
  - $(0 - 1) \cdot x_i^k = -x_i^k$
  - Output was 1 and should have been 0, so make the output more negative

Example: Predict whether a digit is a “2”

Predict whether a digit is a "2"

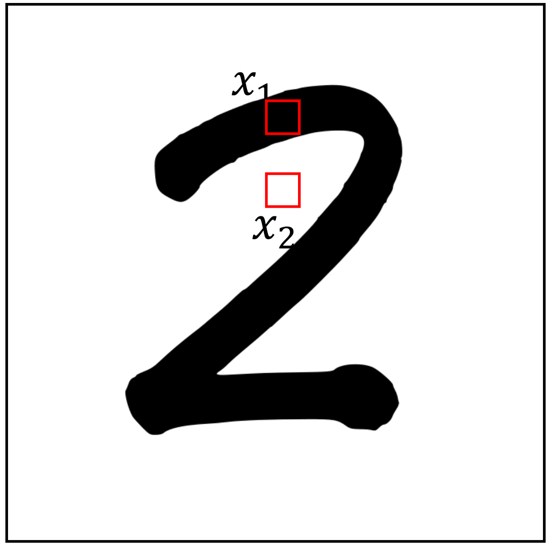


$$x_1 = 0.8$$
$$x_2 = 0$$

Predict whether a digit is a "2"

- Start off training with all parameters as 0, so $w_1 = 0$, $w_2 = 0$, and $b = 0$
- $f(x) = (w_1 \cdot x_1 + w_2 \cdot x_2 + b.1)$
- $f(x) = (0 \cdot 0.8 + 0 \cdot 0 + 0 \cdot 1) = 0$
    - Return 0 because value is not greater than 0
- Predict that it is not a 2!
- Correct answer: it is a 2...
- Parameter update:
    - $\Delta w_1 = (1 - 0) \cdot 0.8 = 0.8$
    - $\Delta w_2 = (1 - 0) \cdot 0 = 0$
    - $\Delta b = (1 - 0) \cdot 1 = 1$
- Now
    - $w_1 = 0.8$
    - $w_2 = 0$
    - $b = 1$

True label = 1



$x_1 = 0.8$

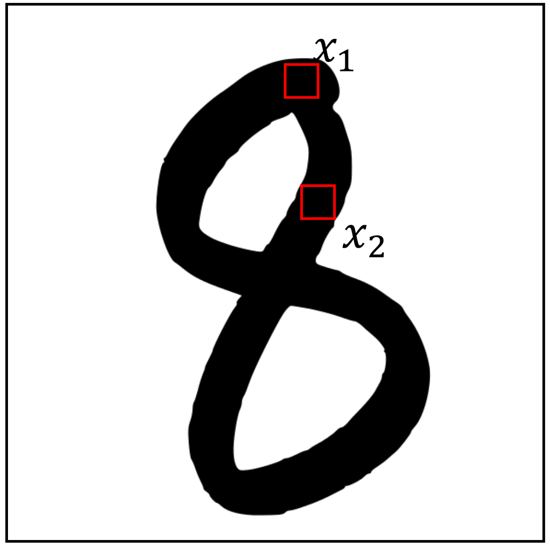
$x_2 = 0$

Predict whether a digit is a "2"

- Next example:

Remember the starting weights are now:

$$w_1 = 0.8$$
$$w_2 = 0$$
$$b = 1$$

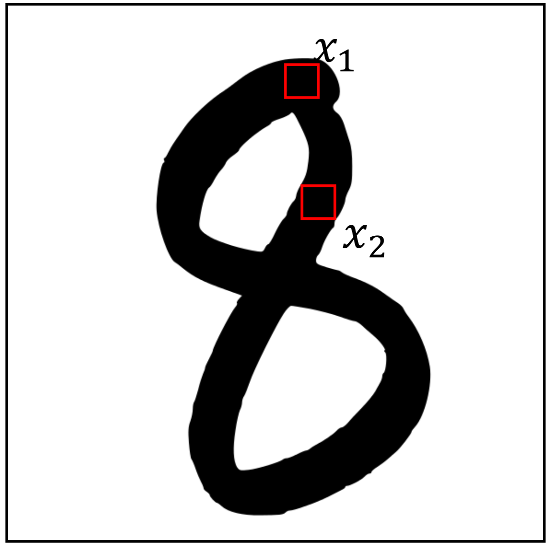


$$x_1 = 0.9$$
$$x_2 = 0.9$$

Predict whether a digit is a "2"

- At end of last iteration:
  - $w_1 = 0.8, w_2 = 0$, and $b = 1$
- $f(x) = (w_1 \cdot x_1 + w_2 \cdot x_2 + b.1)$
- $f(x) = (0.8 \cdot 0.9 + 0 \cdot 0.9 + 1 \cdot 1) > 0$
  - Return 1 because value is greater than 0
- Predict that it is a 2!
- Correct answer: it is not a 2...
- Parameter update:
  - $\Delta w_1 = (0 - 1) \cdot 0.9 = -0.9$
  - $\Delta w_2 = (0 - 1) \cdot 0.9 = -0.9$
  - $\Delta b = (0 - 1) \cdot 1 = -1$
- Now
  - $w_1 = 0.8 - 0.9 = -0.1$
  - $w_2 = 0 - 0.9 = -0.9$
  - $b = 1 - 1 = 0$

True label = 0



$x_1 = 0.9$
$x_2 = 0.9$

Any questions?

Multi-class problem

Bringing back the complexity

Input: $\mathbb{X}$

Classifying MNIST digits requires predicting 1 of 10 possible values

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

$x^{(1)} =$ 

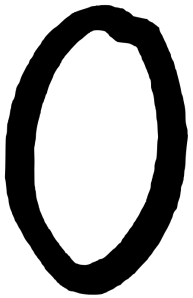
Function: f

$y^{(1)} =$ “2”

28x28 pixels

$\boxed{f(\mathbb{X}) \rightarrow \mathbb{Y}}$

How do we do that?

$x^{(2)} =$ 

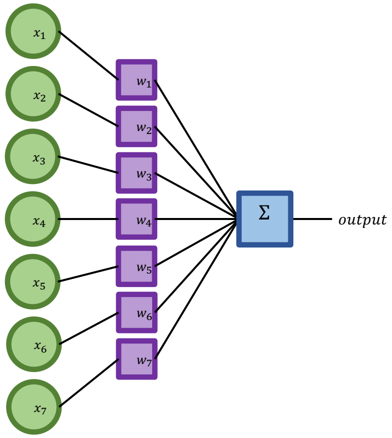
Rather than predicting whether a handwritten digit is of a certain class, we predict the class it is most likely in

$y^{(2)} =$ “0”

Using multiple perceptrons

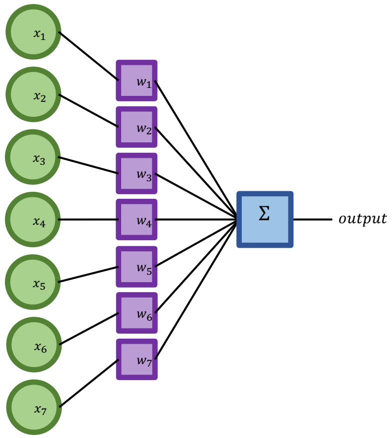
- We can extend perceptrons to multi-class problems by creating m perceptrons, where m= the number of classes
- For MNIST, we would have 10 perceptrons
- Each individual perceptron returns a value, so our model will return the class whose perceptron value is the highest.
  - Here, "perceptron value" refers to the value of the weighted sum before being thresholded.

Using multiple perceptrons



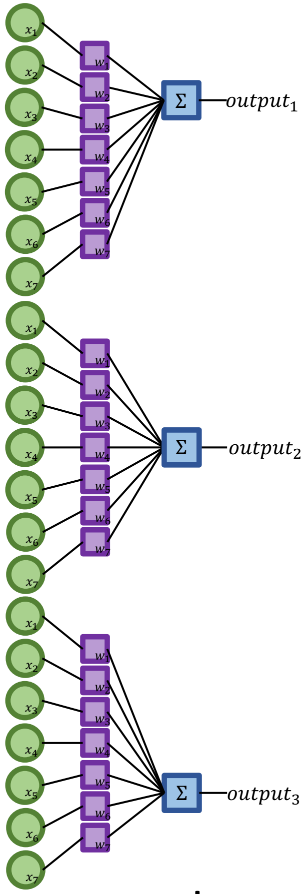
Perceptron for predicting whether handwritten digit is a 0

⋮



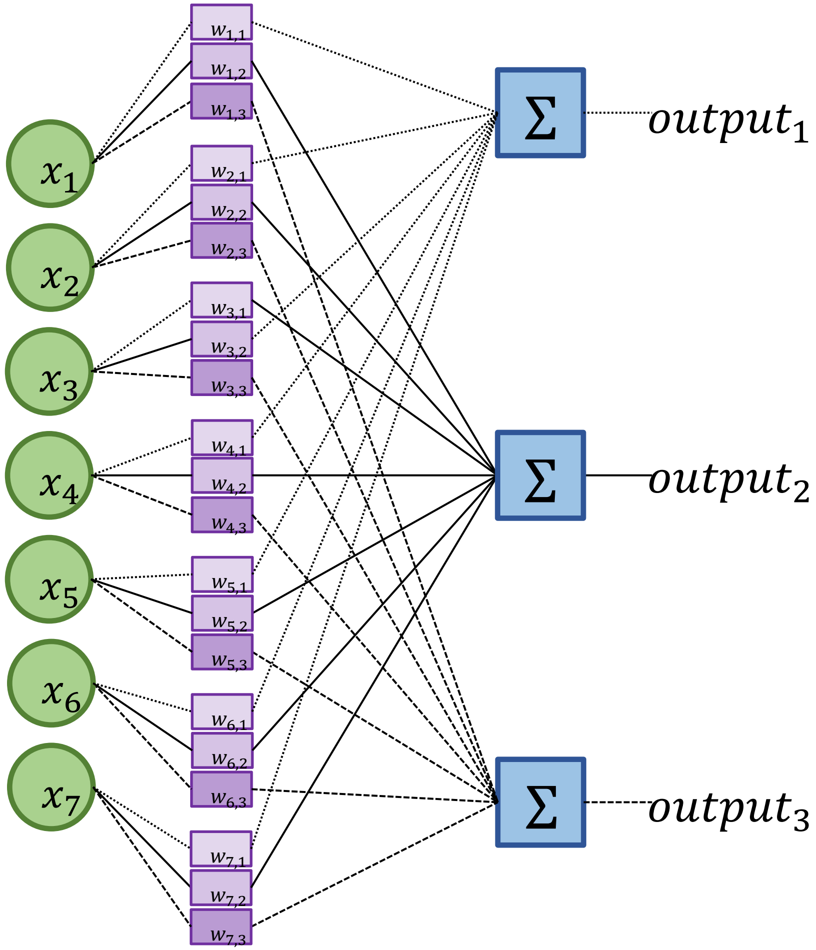
Perceptron for predicting whether handwritten digit is a 9

Multi-class Perceptron



Three separate perceptrons

=

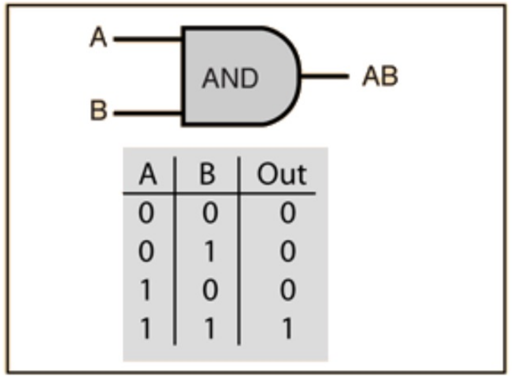


Three perceptrons sharing inputs

Is there anything the Perceptron can't learn?

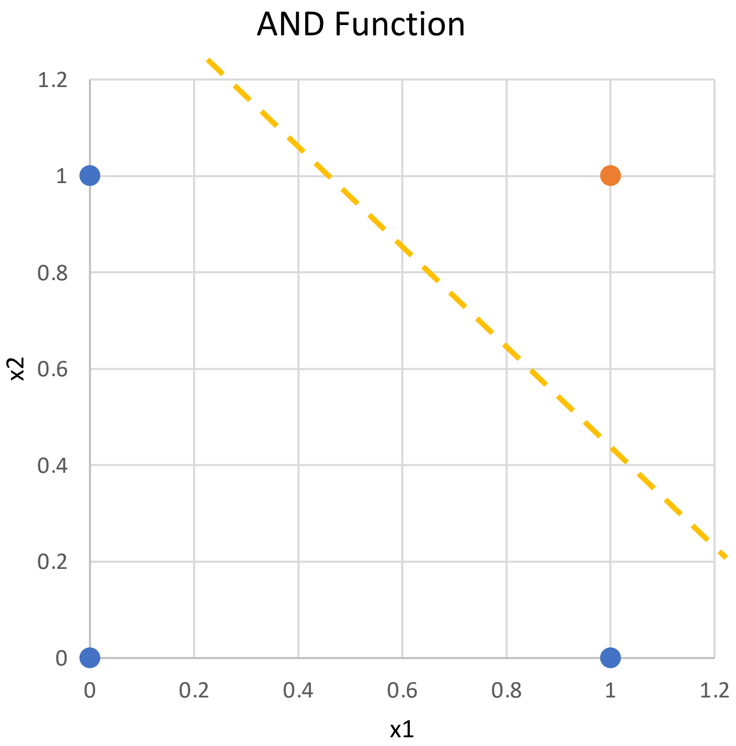
AND Function

Perceptrons work well with linearly separable data



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate



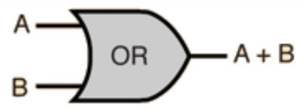
Output = 0

Output = 1

$w_1 \cdot x_1 + w_2 \cdot x_2 + b > 0\,?$

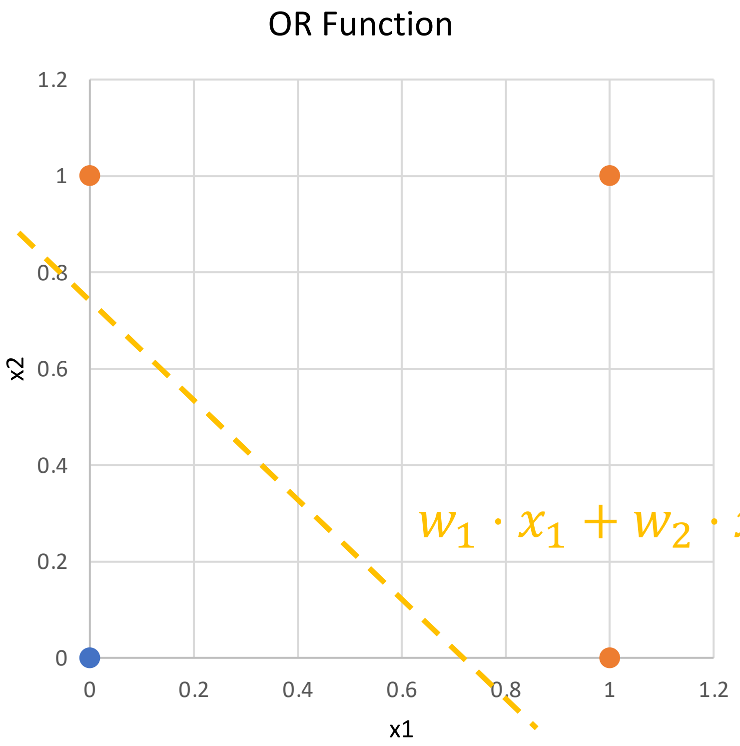
Linear decision boundary

OR Function



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

OR Gate



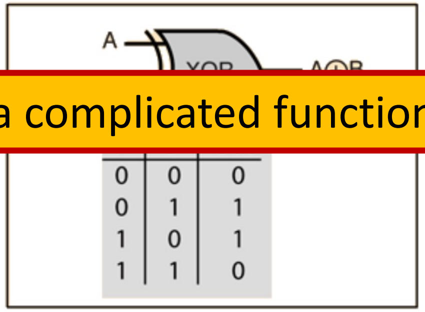
Output = 0

Output = 1

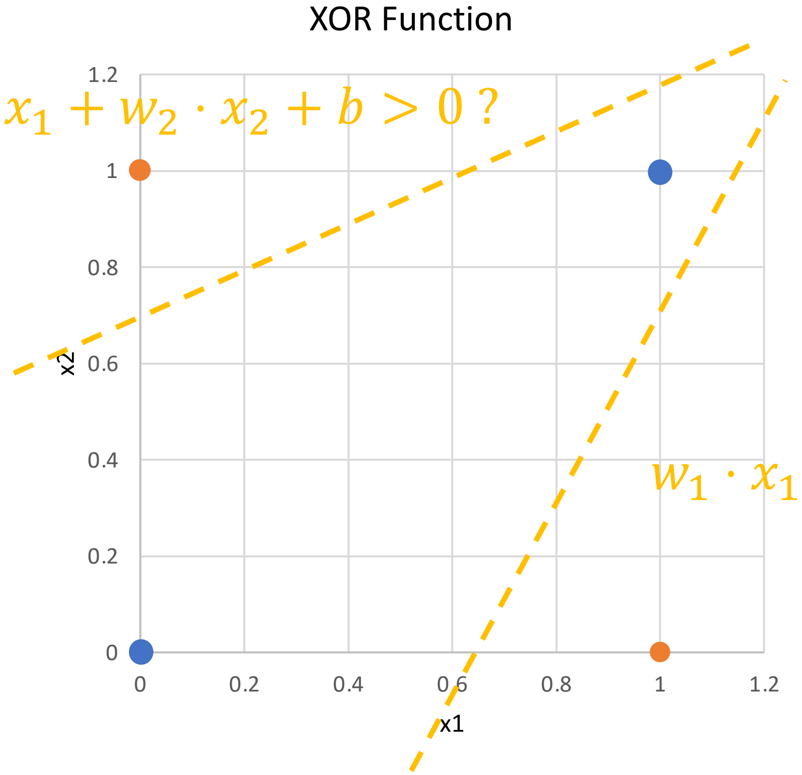
$$w_1 \cdot x_1 + w_2 \cdot x_2 + b > 0\ ?$$

XOR Function

Complicated data would need a complicated function!



XOR Gate



$w_1 \cdot x_1 + w_2 \cdot x_2 + b > 0\ ?$

$w_1 \cdot x_1 + w_2 \cdot x_2 + b > 0\ ?$

Output = 0

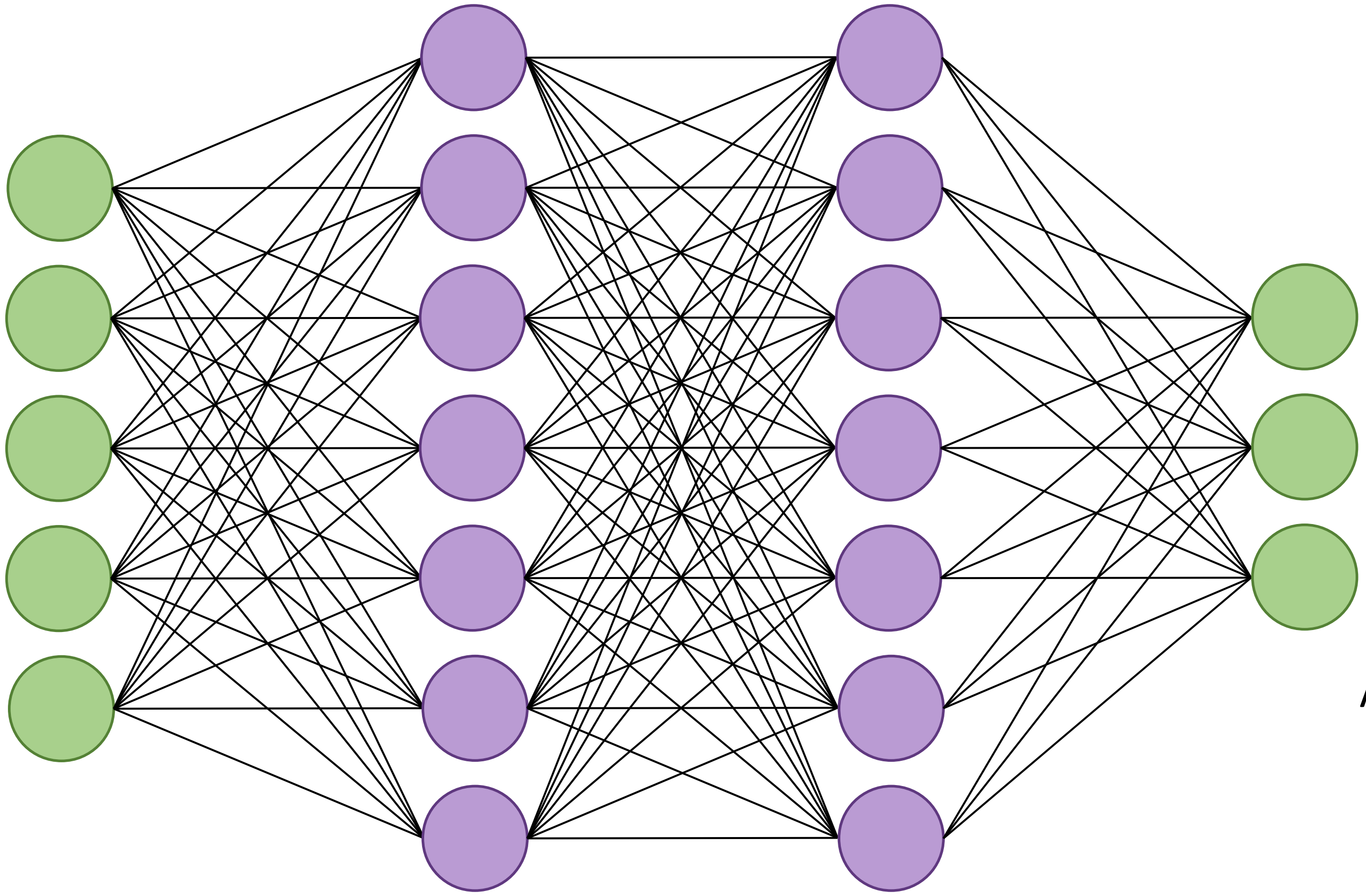
Output = 1

Need ***two*** linear decision boundaries to represent this function…

Multi-Layered Neural Net

May see the term Multi-Layer Perceptron (MLP), HOWEVER "perceptrons" are not perceptrons in the strictest possible sense

We really don’t use the threshold function of a perceptron but still use the linear function

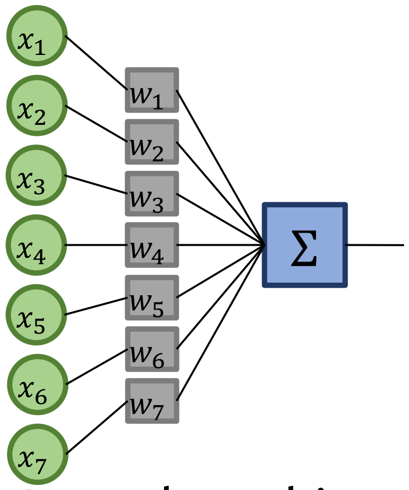


A Multi-Layered Neural Net

Any questions?

How do we train multi-layer networks?

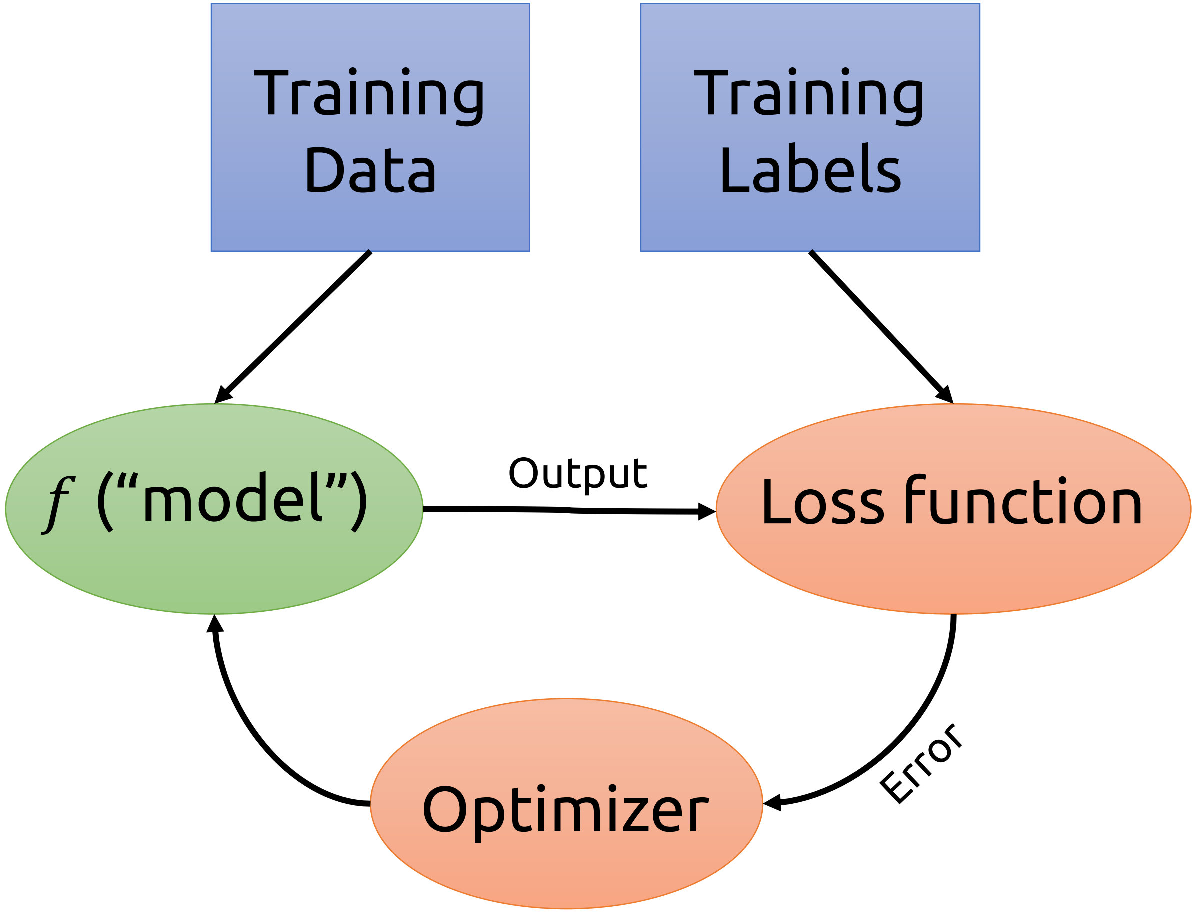
- Unfortunately, the perceptron algorithm doesn't generalize beyond the one-layer case... ☹
- We need a new algorithm...



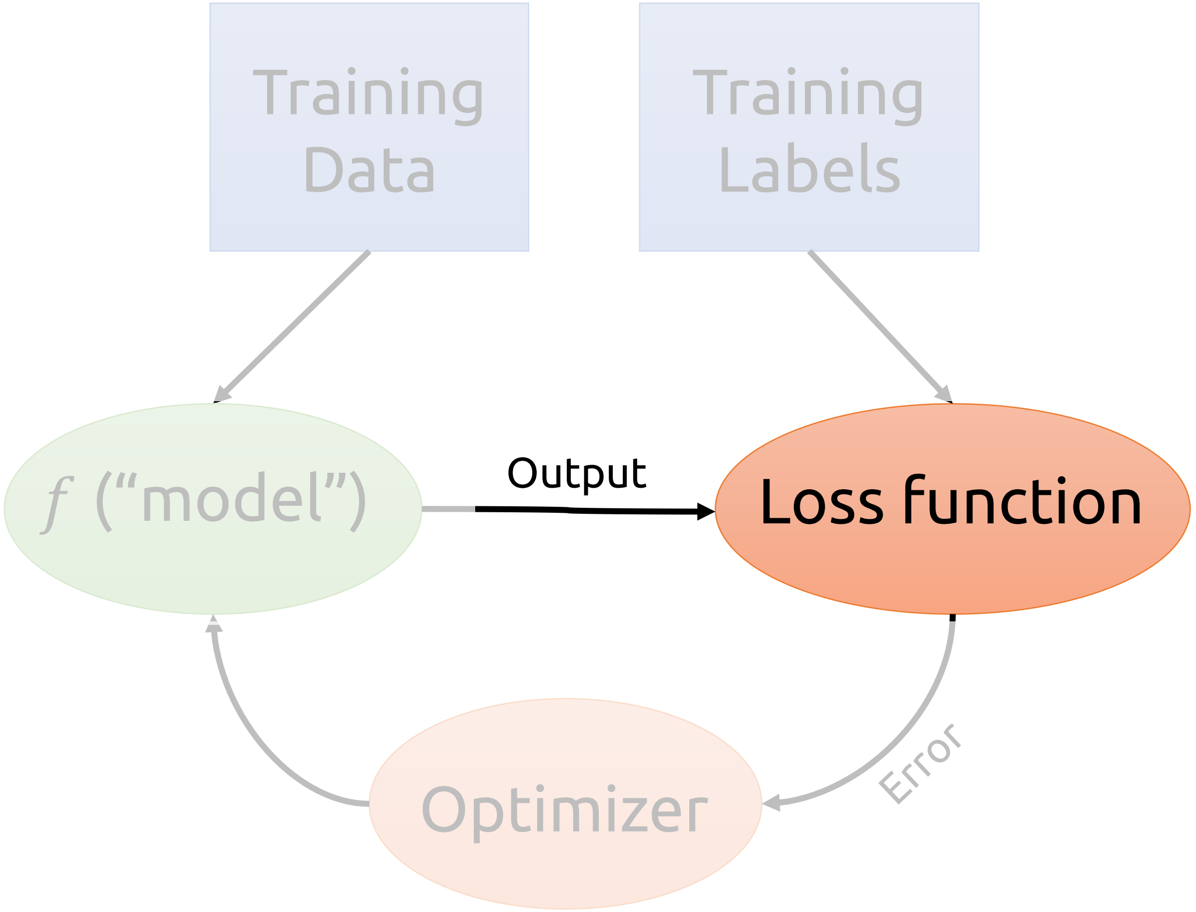
A one-layer binary perceptron

1. set w's to 0.
2. for N iterations, or until the weights do not change:
   a) for each training example $\mathbf{x}^k$ with label a^k
      i. if $a^k - f(\mathbf{x}^k) = 0$ continue
      ii. else for all weights w_i, $\Delta w_i = \left(a^k - f(\mathbf{x}^k)\right) x_i$

A critical ingredient for our new approach: Loss functions



A critical ingredient for our new approach: Loss functions



What is a Loss Function?

- A function L which measures how "wrong" a network is
- L is computed by comparing two values (predicted and true) that shows which is better
- Evaluate L and update parameters to decrease L, making the network "less wrong"

Recap – regression task

Input: $\mathbb{X}$

"Temperature"

Regression

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$x^{(1)} = 100.1$

$x^{(2)} = 60.0$

$x^{(3)} = 30.3$

$\mathbb{X} \in \mathbb{R}$

Function: f

f(X)→Y

$y^{(1)} = 200.0$

$y^{(2)} = 160.5$

$y^{(3)} = 145.1$

$\mathbb{Y} \in \mathbb{R}$

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Recap: Learning function f

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 60.0$

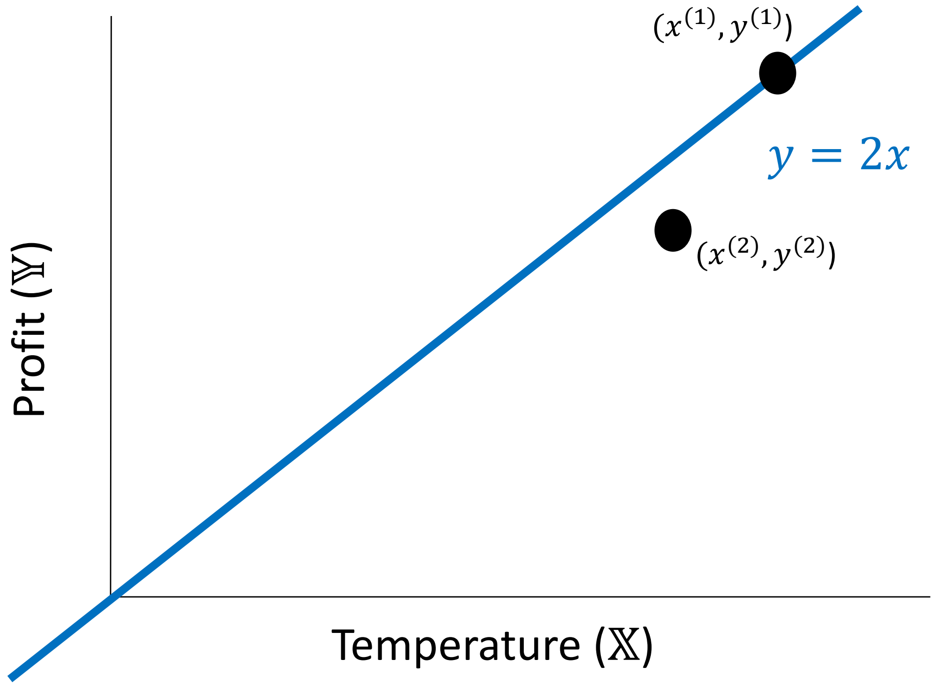
$x^{(3)} = 30.3$

$\mathbb{X} \in \mathbb{R}$

What could be our loss function?

Linear function

$$y = wx + b$$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 160.5$

$y^{(3)} = 145.1$

$\mathbb{Y} \in \mathbb{R}$

(Numerical output)

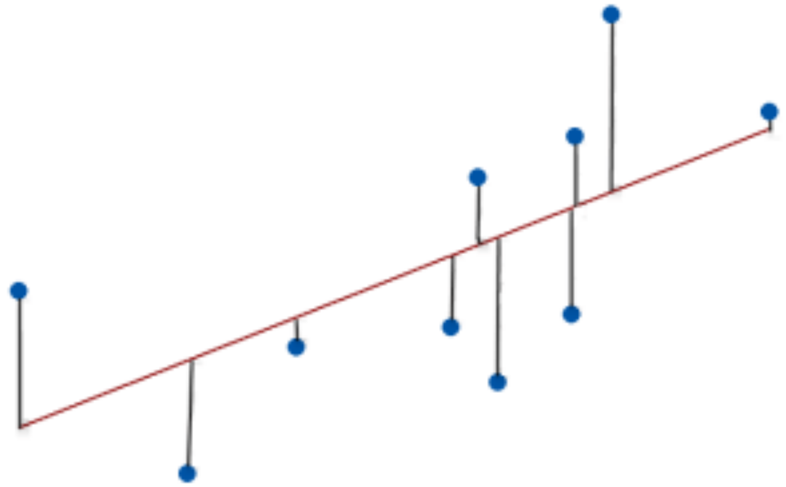
(Image only for explaining concept, not drawn accurately)

Mean Squared Error (MSE)

Average squared residual (residual: difference between predicted and true value)

Decreasing the MSE = the model has less error = data points fall closer to the regression line

$$MSE = \frac{\sum_{k=1}^{n}(y^k - \hat{y}^k)^2}{n}$$



MSE is the average squared distance between the observed and predicted values

y^k : $true\ output\ value$

$\hat{y}^k$: $predicted\ output\ value$

n: $number\ of\ samples$

What could be the purpose of squaring the distance?

Courtesy: https://statisticsbyjim.com/regression/mean-squared-error-mse/

Mean Squared Error (MSE)

Average squared residual (residual: difference between predicted and true value)

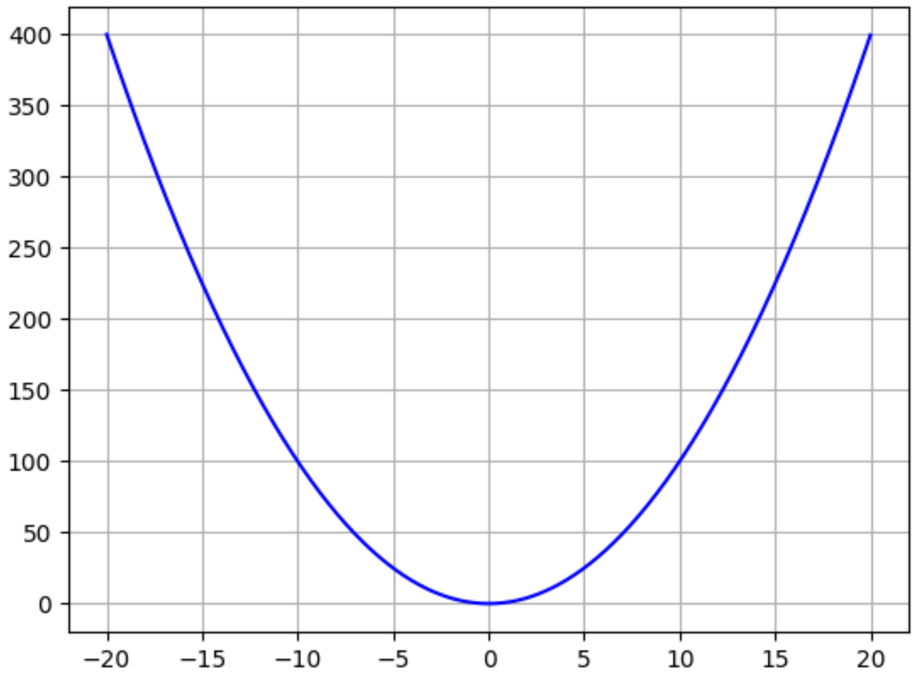
Decreasing the MSE = the model has less error = data points fall closer to the regression line

$$MSE = \frac{\sum_{k=1}^{n} (y^k - \hat{y}^k)^2}{n}$$

$y^k : \textit{true output value}$

$\hat{y}^k : \textit{predicted output value}$

$n : \textit{number of samples}$



What could be the purpose of squaring the distance?

Courtesy: https://statisticsbyjim.com/regression/mean-squared-error-mse/

Recap: Binary classification

Input: $\mathbb{X}$

What is a good loss for our binary classification?

Target: $\mathbb{Y}$

Pixel Grid

We want our network to produce the right answer with high probability

Is it digit 2?

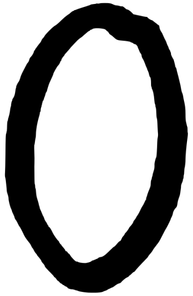
$x^{(1)} =$ 

Function: f

$y^{(1)} = \text{“1”}$

28x28 pixels

1. Make the network output a probability for class 1 (a value between 0 and 1)

$x^{(2)} =$ 

2. Use this probability to compute a loss

$y^{(2)} = \text{“0”}$

Cross Entropy Loss (for Binary classification)

$y = true\ label\ of\ class\ (0\ or\ 1)$

$p = predicted\ probability\ of\ class\ 1$

When the true label is 1 we want higher predicted probability for a digit to be 2

When the true label is 0 we want lower predicted probability for a digit to be 2

$\log(p)$

Some examples:

$\log(0.9) = -0.04$

$\log(0.5) = -0.3$

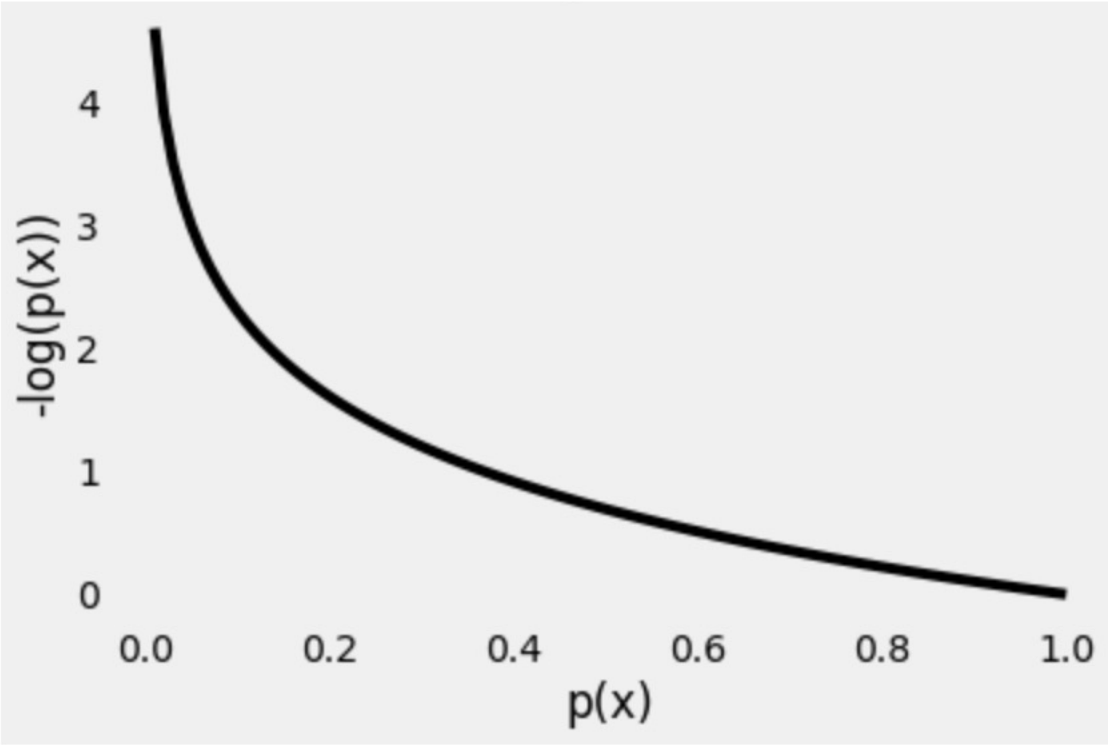
$\log(0.001) = -3$

Cross Entropy Loss (for Binary classification)

$y = true\ label\ of\ class\ (0\ or\ 1)$

$p = predicted\ probability\ of\ class\ 1$

$-\log(p)$



Some examples:

$\log(0.9) = -0.04$

$\log(0.5) = -0.3$

$\log(0.001) = -3$

Cross Entropy Loss (for Binary classification)

$y = true\ label\ of\ class\ (0\ or\ 1)$

$p = predicted\ probability\ of\ class\ 1$

$$-(\mathrm{y} \log(p))$$

Some examples:

$\log(0.9) = -0.04$

$\log(0.5) = -0.3$

$\log(0.001) = -3$

Cross Entropy Loss (for Binary classification)

$y = true\ label\ of\ class\ (0\ or\ 1)$

$p = predicted\ probability\ of\ class\ 1$

y = 1, p = 0.9

$$-(\mathrm{y}\log(p) + (1-y)\log(1-p))$$

y = 0, p = 0.9

y = 1, p = 0.001

Some examples:

$\log(0.9) = -0.04$

$\log(0.5) = -0.3$

$\log(0.001) = -3$

Any questions?

We get this probability by using a Sigmoid function

y = 0, p = 0.001

Recap: Multi-class classification

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

We want our network to produce the right answer with high probability

Which digit is it?

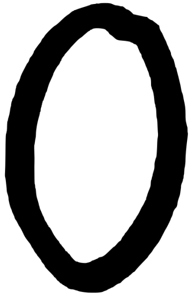
$x^{(1)} =$ 

Function: f

$y^{(1)} =$ “2”

28x28 pixels

1. Make the network output probabilities for all classes (values between 0 and 1)

$x^{(2)} =$ 

2. Use these probabilities to compute a loss

$y^{(2)} =$ “0”

Cross Entropy Loss (for Multi-class classification)

$$-\sum_{j=1}^{m} y_j \log(p_j)$$

p	Classes (m)	**y**
0.3	"0"	0
0.2	"1"	0
0.5	"2"	1

Some examples:

$\log(0.9) = -0.04$

$\log(0.5) = -0.3$

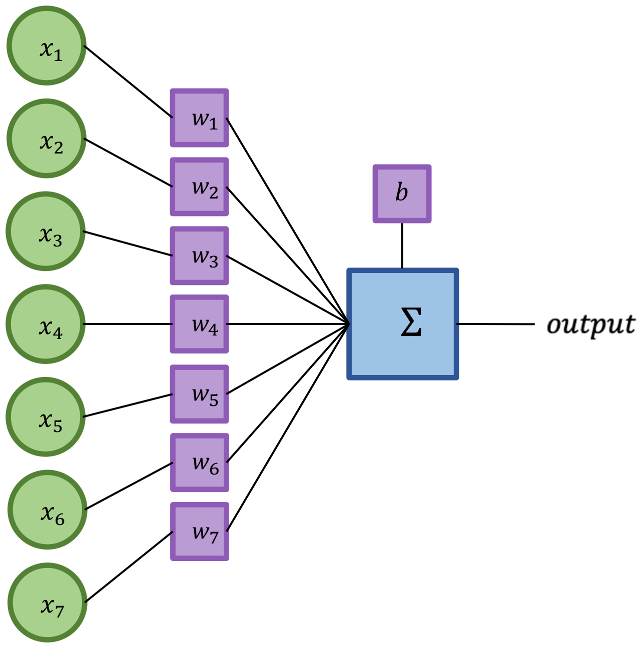
$\log(0.001) = -3$

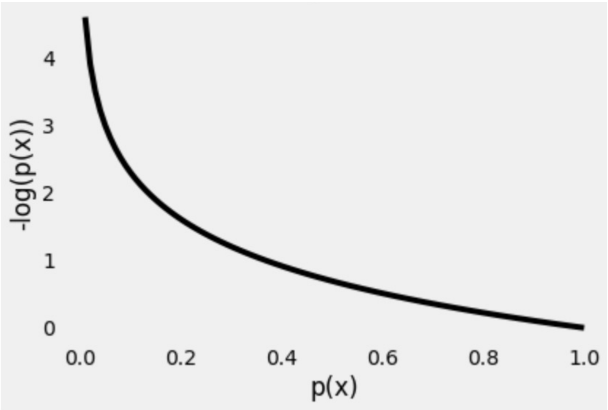
We want model to assign high probability to the true class and low to others

Recap

Perceptron

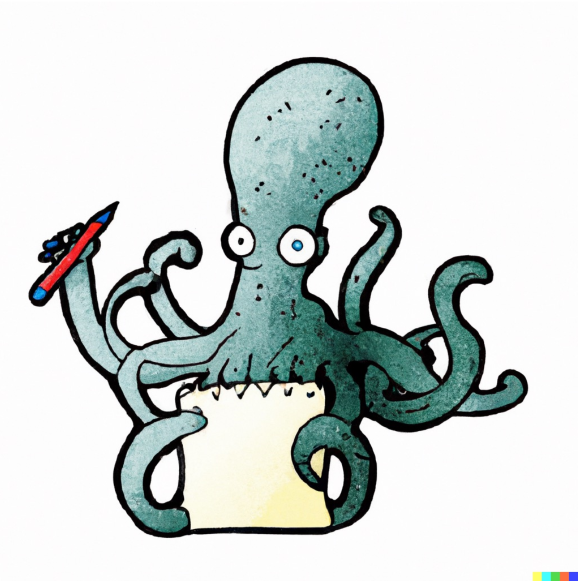
- Perceptron training w/ working example
- Multi-class classification
- When perceptron fails





Loss functions

- MSE loss for regression
- Cross entropy loss for binary classification
- Cross entropy loss for multi-class classification



Some Trivia: The Fall of Perceptrons

- In 1969, Marvin Minsky and Seymour Papert released a book, *Perceptrons*, demonstrating that perceptrons are not able to learn the XOR function
- Many earlier researchers heavily focused on logical reasoning, a feature of high-level human cognition, so a machine's ability for logical reasoning was thought to indicate "artificial intelligence"
- Part of a funding battle: Minksy and Papert wanted federal AI funding to go to their kind of 'symbolic' AI, not the early neural net folks...