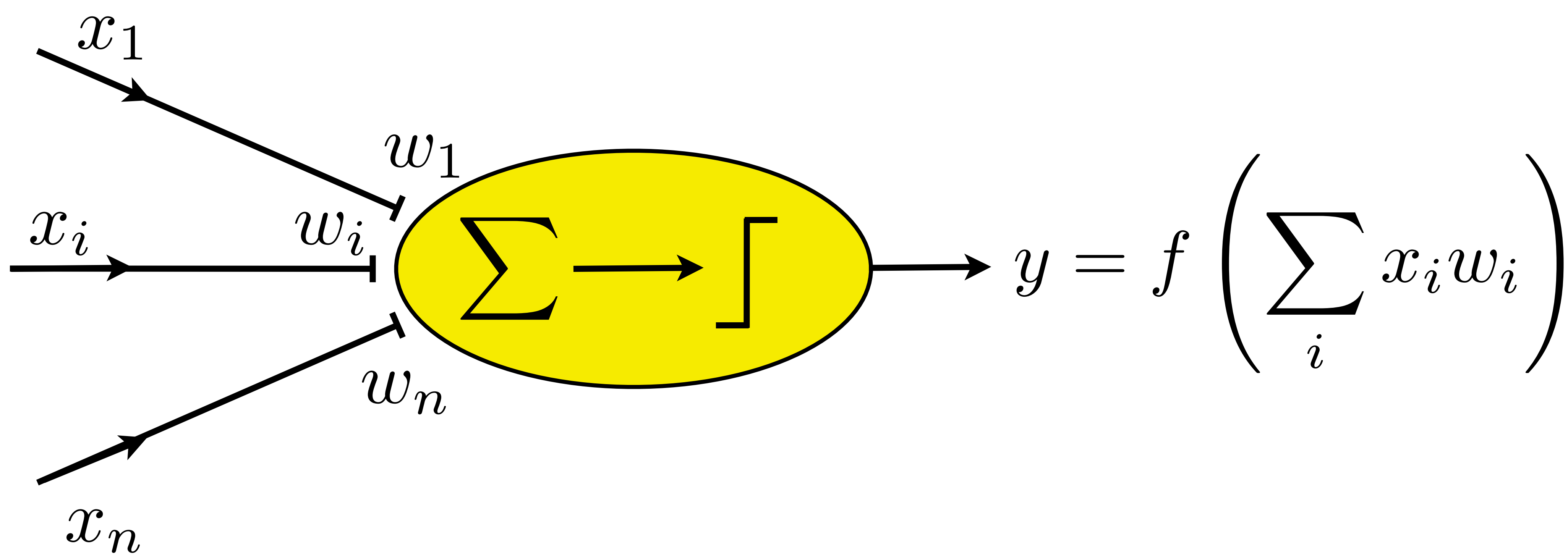
Mathematical simplification

- Neurons are either on or off: represented by binary value.

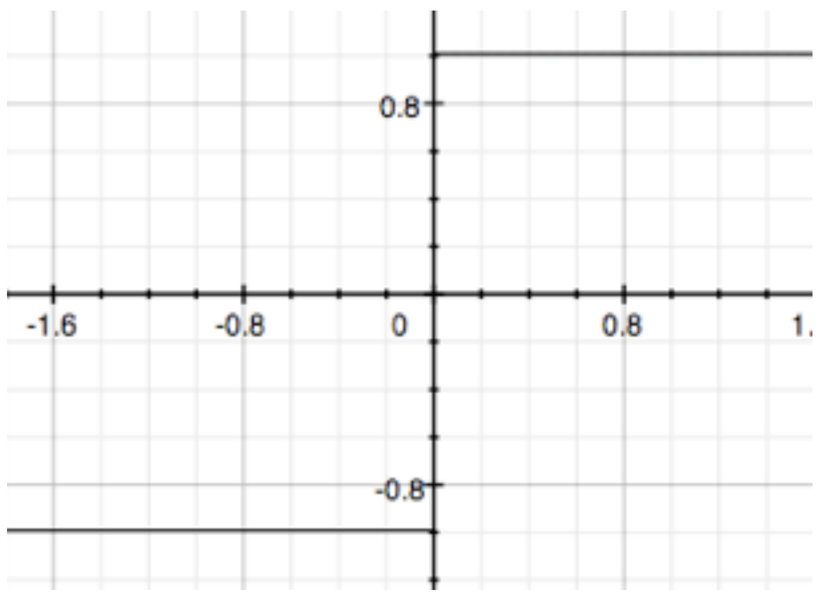


- Inputs form n neurons: x_i
- get multiplied by weights at "synapses": $x_i w_i$
- then added $\sum_i x_i w_i$
- if above threshold, then neuron is "on".

- Remark: add x_0 to the input vector, in order to write the bias as $b = x_0 w_0$ to get more compact form **wx**, rather than **wx** + b

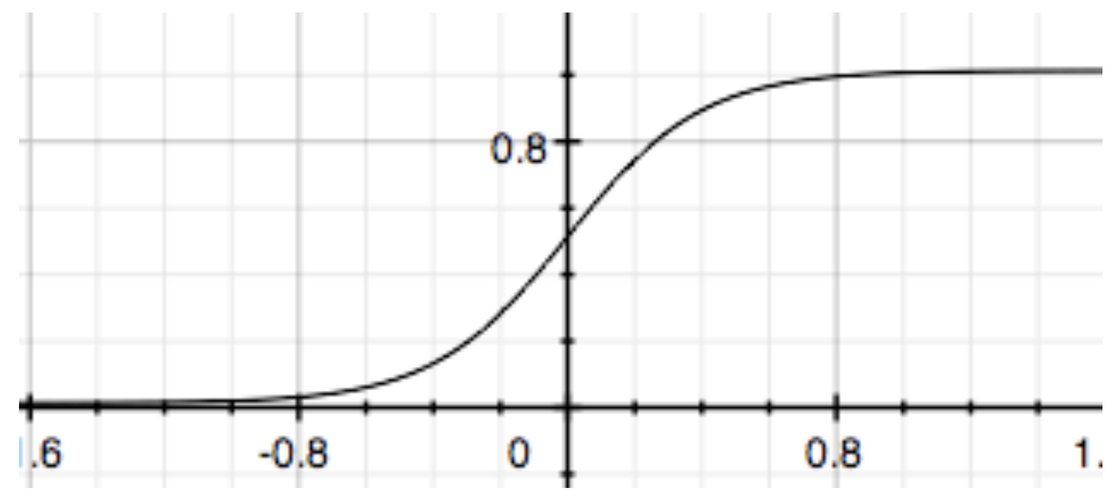
Transfer function

- Step function:



- Used by Pitts & McCulloch (1942), and in Rosenblatt's Perceptron (1957)

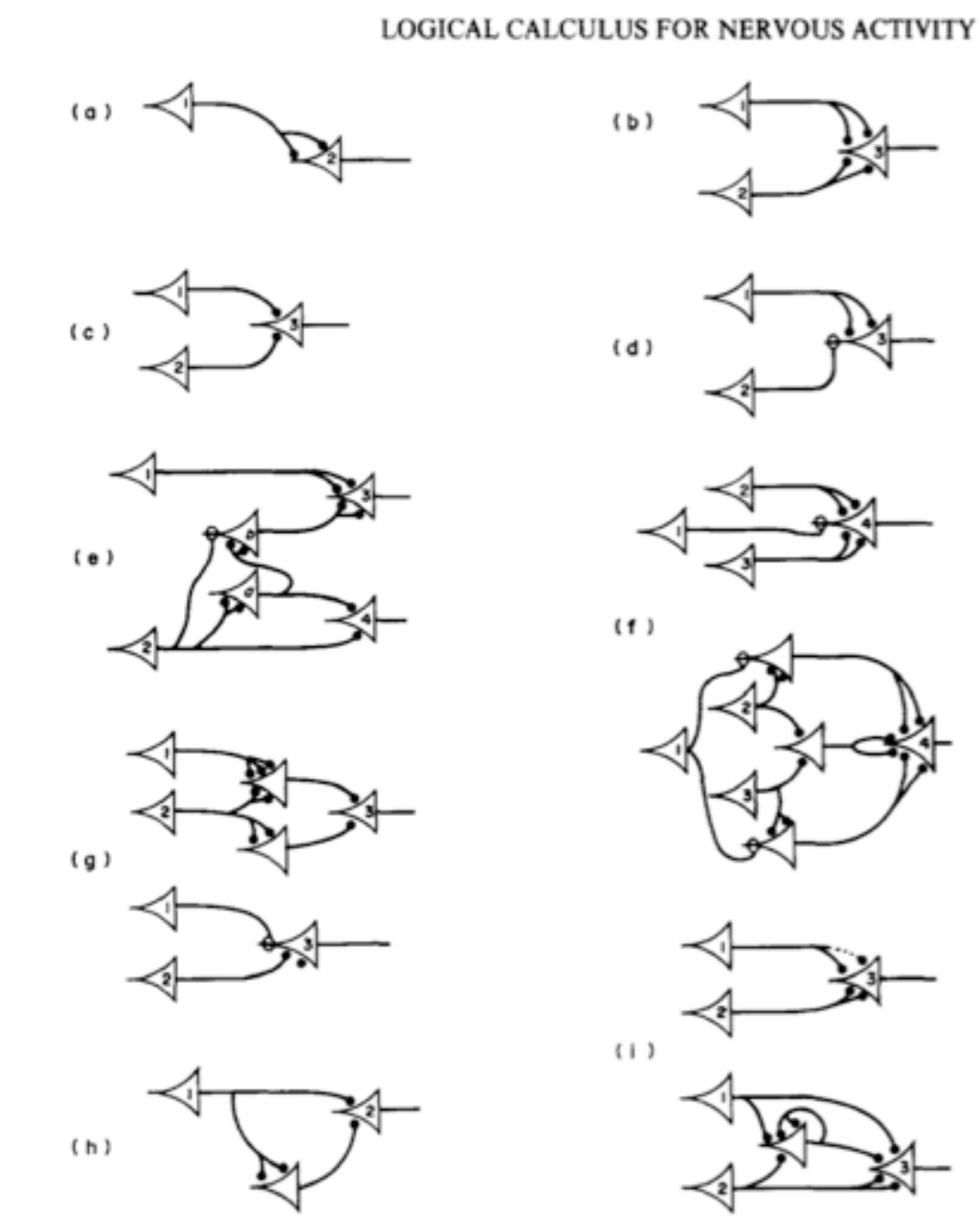
- Sigmoid (used in many "modern" feed forward neural nets):



- Sigmoidal function is differentiable (good for deriving gradient decent learning rule; Backpropagation algorithm, Werbos 1974)

Neural networks (1940s/50s)

- Pitts and McCulloch (1942/3): "Formal" (mathematical) neurons thought of as processing units
- *Networks* can emulate any logical function.
- D. Hebb: Connections between neurons can change. "Learning rule": strengthen proportional to correlation between activity of pre- and post synaptic neuron.



Perceptron (Rosenblatt, 1957)

- Probably the first "learning machine" that was actually built. Artificial neuron that solved a classification task (supervised learning):

- Given: N input vectors **x** together with labels l (these labels are the "teaching" signal).

- Goal: for any given input, the output of the classifier should be the same as the label.

Perceptron learning

- Simplify: assume labels are binary, either -1 or +1.

  ➡ *Binary classification*.

- *adjust weights according to the correctness of the output* until all input data in training set are classified correctly.

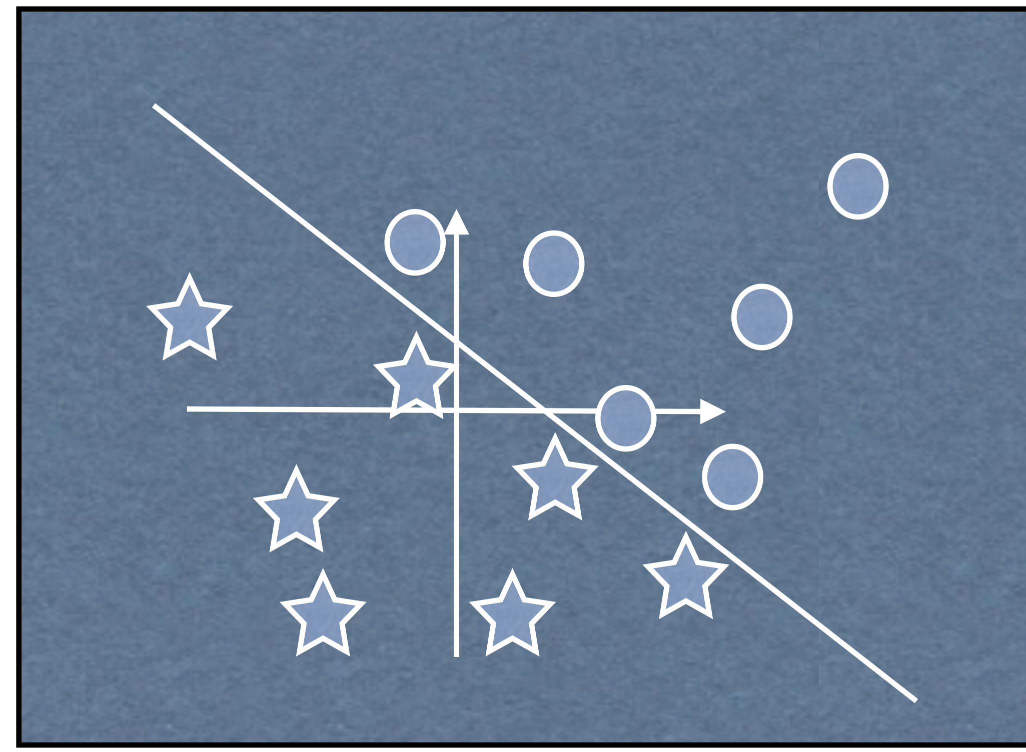
- **error measure**: compare output of the neuron (y) to desired output (= label, l): both are either -1 or 1, so:
  - y*l = 1: correct classification, then: l - y = 0
  - y*l = -1: incorrect classification, then: l - y = 2l

- adjust weight vector **w** *for each misclassified* input **x**, by adding l***x**. This turns **w** towards/away from **x** if l=1/-1

- can do $\boxed{\mathbf{w} \longleftarrow \mathbf{w} + c(l-y)\mathbf{x}}$ for all training examples **x**

- c: step size parameter called “learning rate”.

Representational power



- the decision boundary of the perceptron is a line:

  wx + b

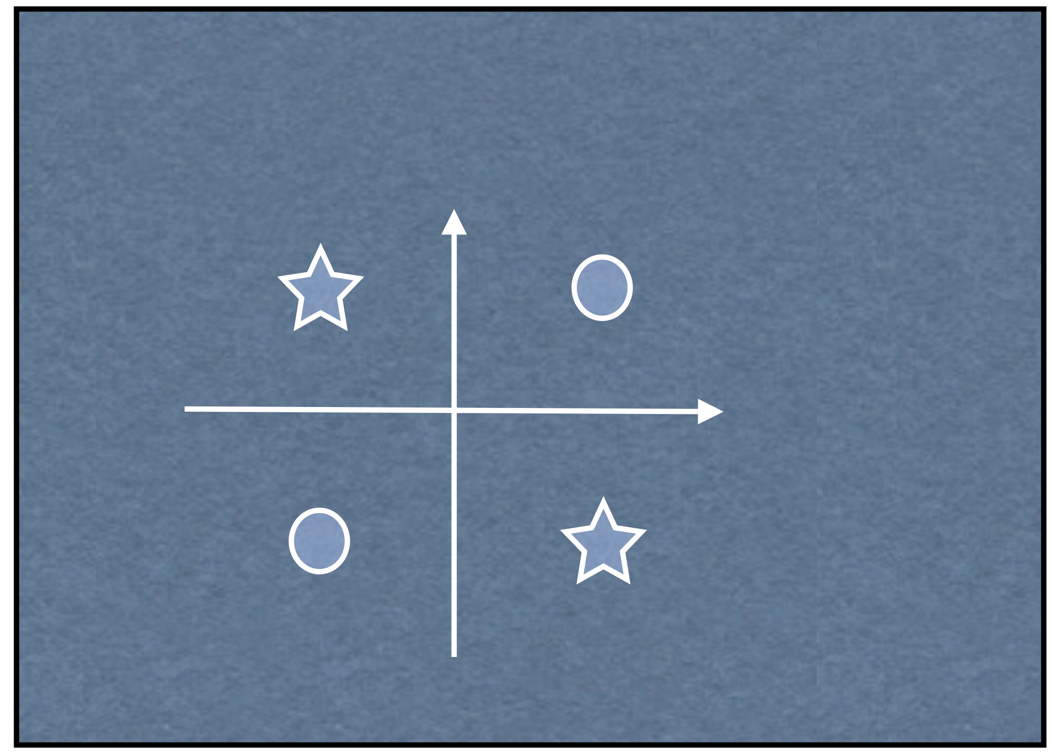
- the perceptron learning algorithm *converges* if input data are **linearly separable.**

- Novikoff (1962): Perceptron Convergence Theorem; first *margin-based error bound* (in a way the "dawn" of statistical learning theory)

XOR problem

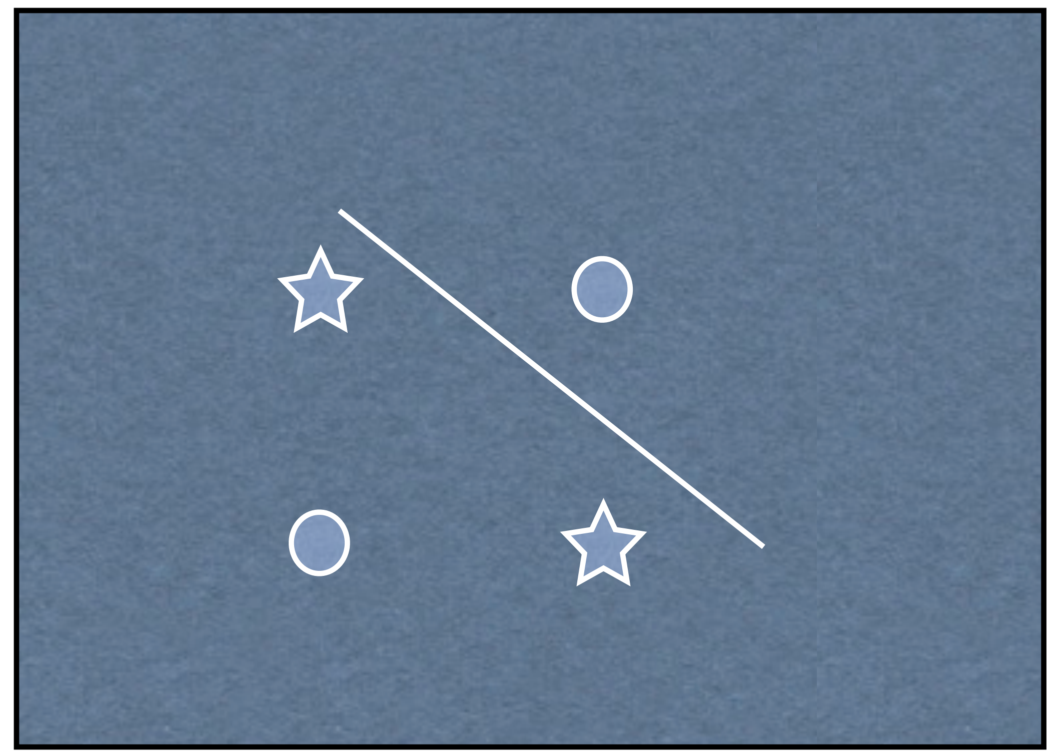
- Perceptron algorithm can not classify input data which can not be separated by a line.

- For example XOR



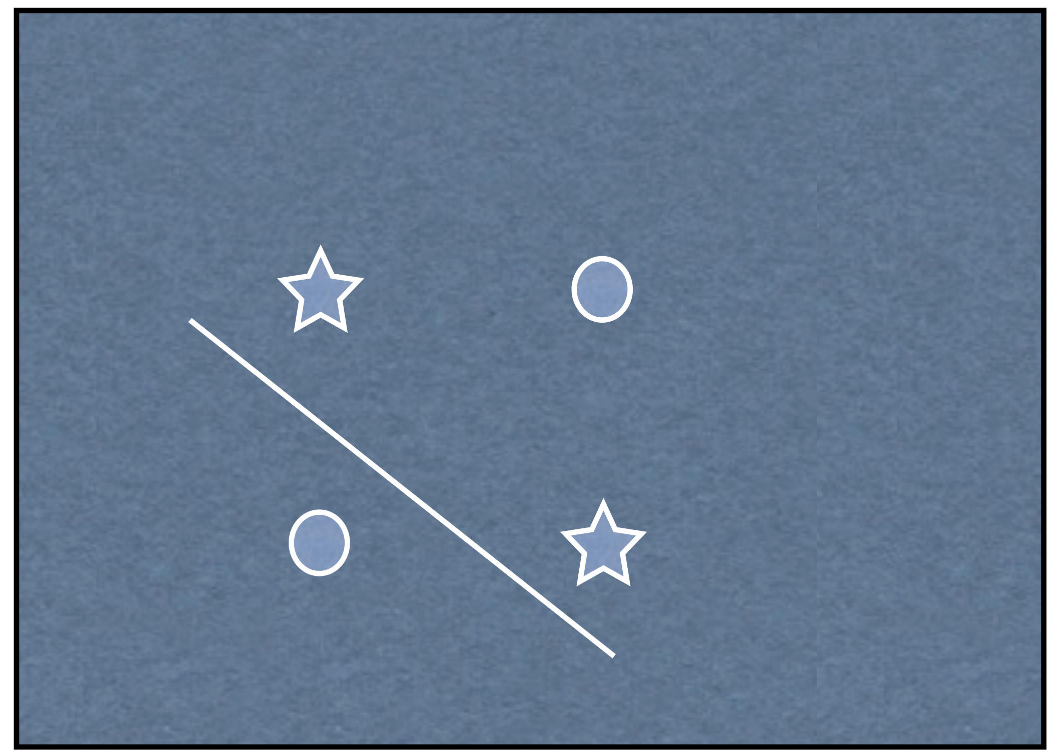
XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



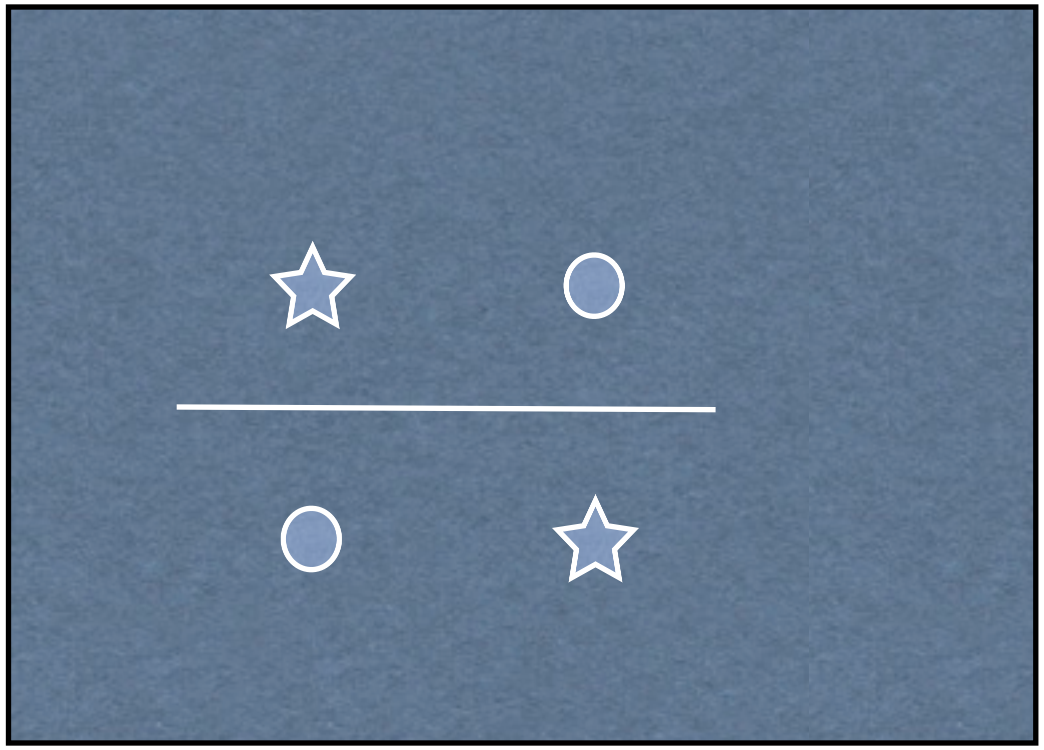
XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



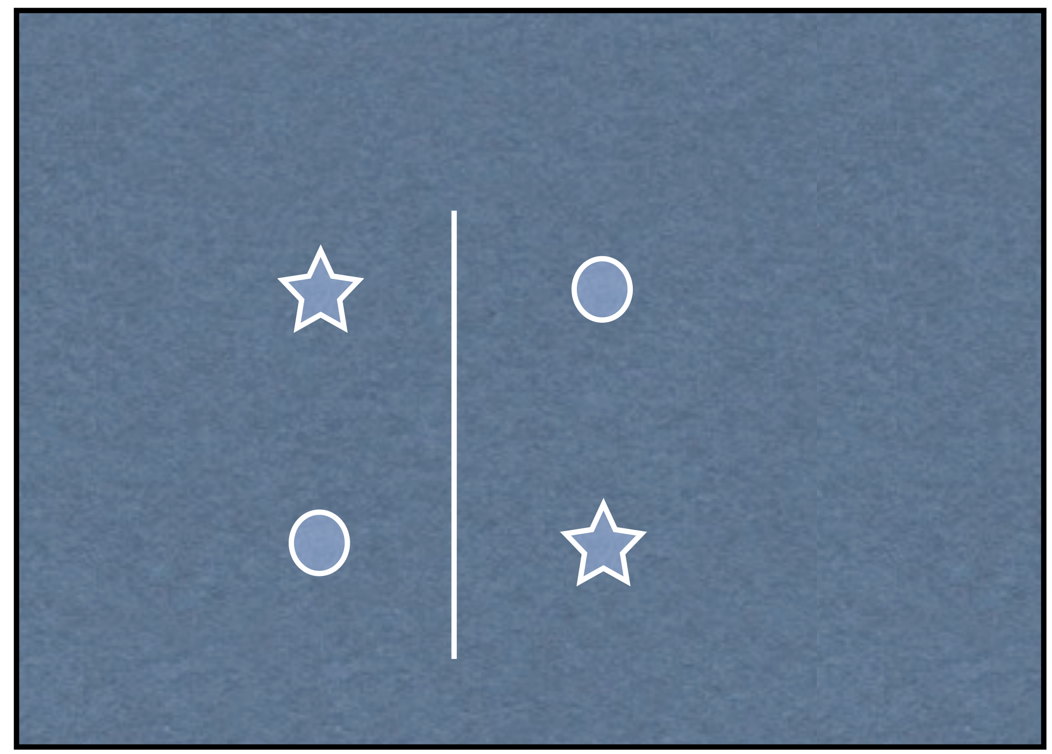
XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



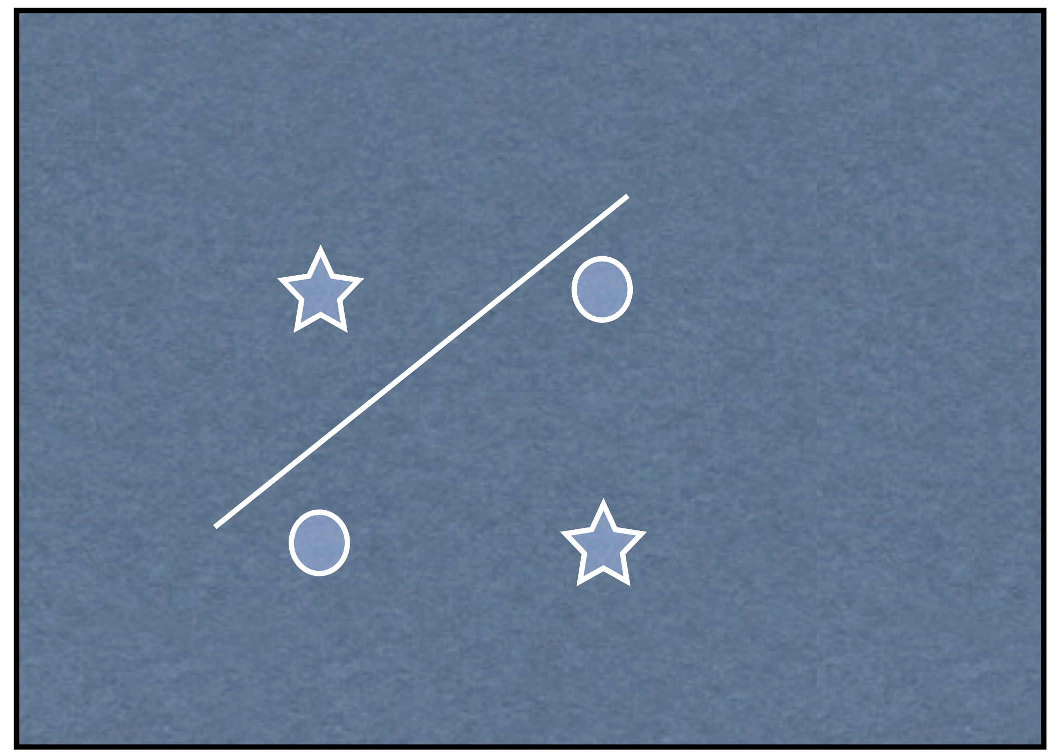
XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



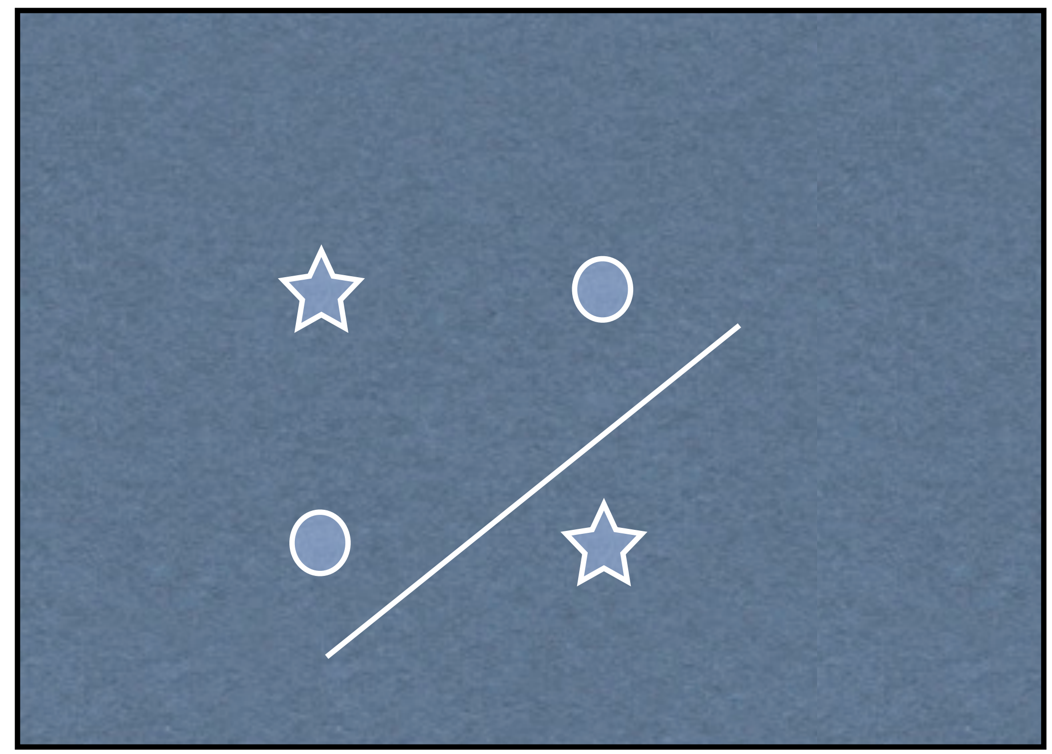
XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



XOR problem

- Perceptron algorithm can not classify input data which can not be separated by a line.
- For example XOR



- A ***single artificial neuron*** can express the boolean functions AND, OR, and NOT, **but not XOR**.

Homework

- Implement perceptron learning algorithm: N Inputs $\mathbf{x}_j$ & labels l_j
    - Initialize weight vector $\mathbf{w}$
    - While there exist misclassified examples:
        - Compute output $y_j = \theta(\mathbf{w}\mathbf{x}_j)$
        - For each example, update the weights: $\mathbf{w} \mathrel{+}= c(l_j - y_j)\mathbf{x}_j$
- Play around with the parameter (learning rate) and the input data, and verify for yourself what the Perceptron can and can not do
    - Make a movie of Perceptron converging, and one of Perceptron failing on the XOR.
- What else do you notice?
    - Is every solution the same? If not, are some "better" than others in some sense?