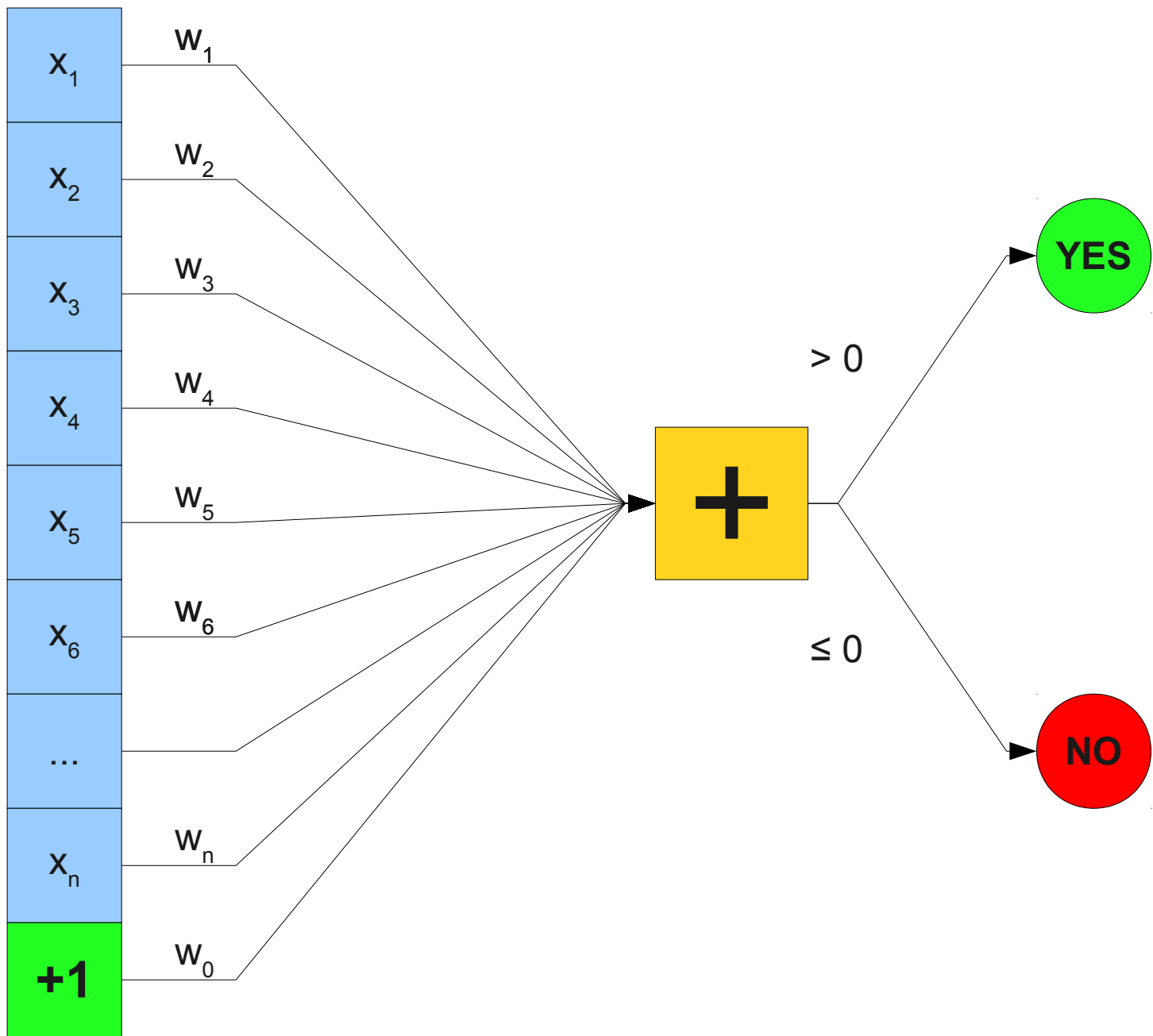


Introduction to Perceptrons



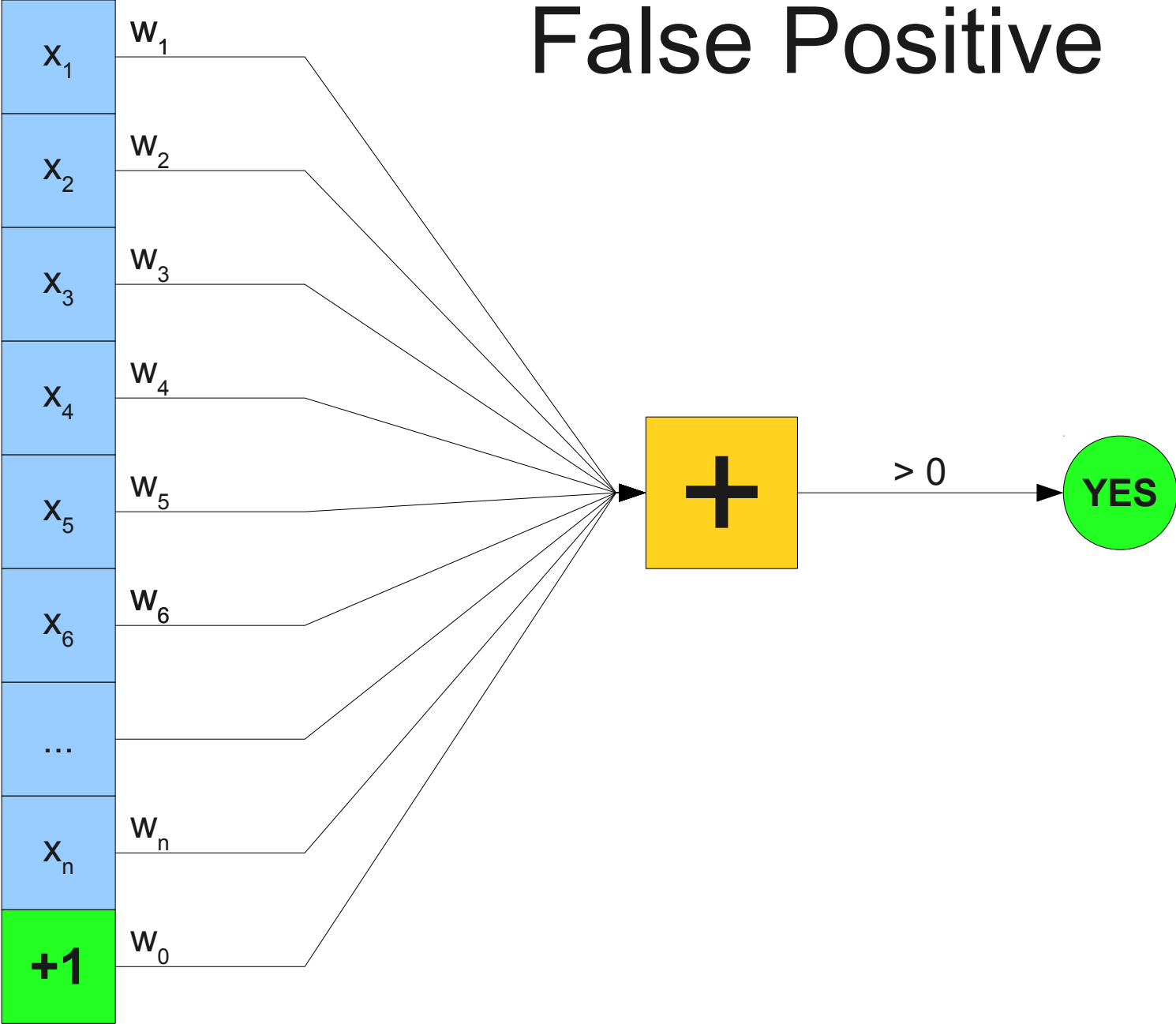
How do we choose good values for $w_0 \dots w_n$?

One Approach

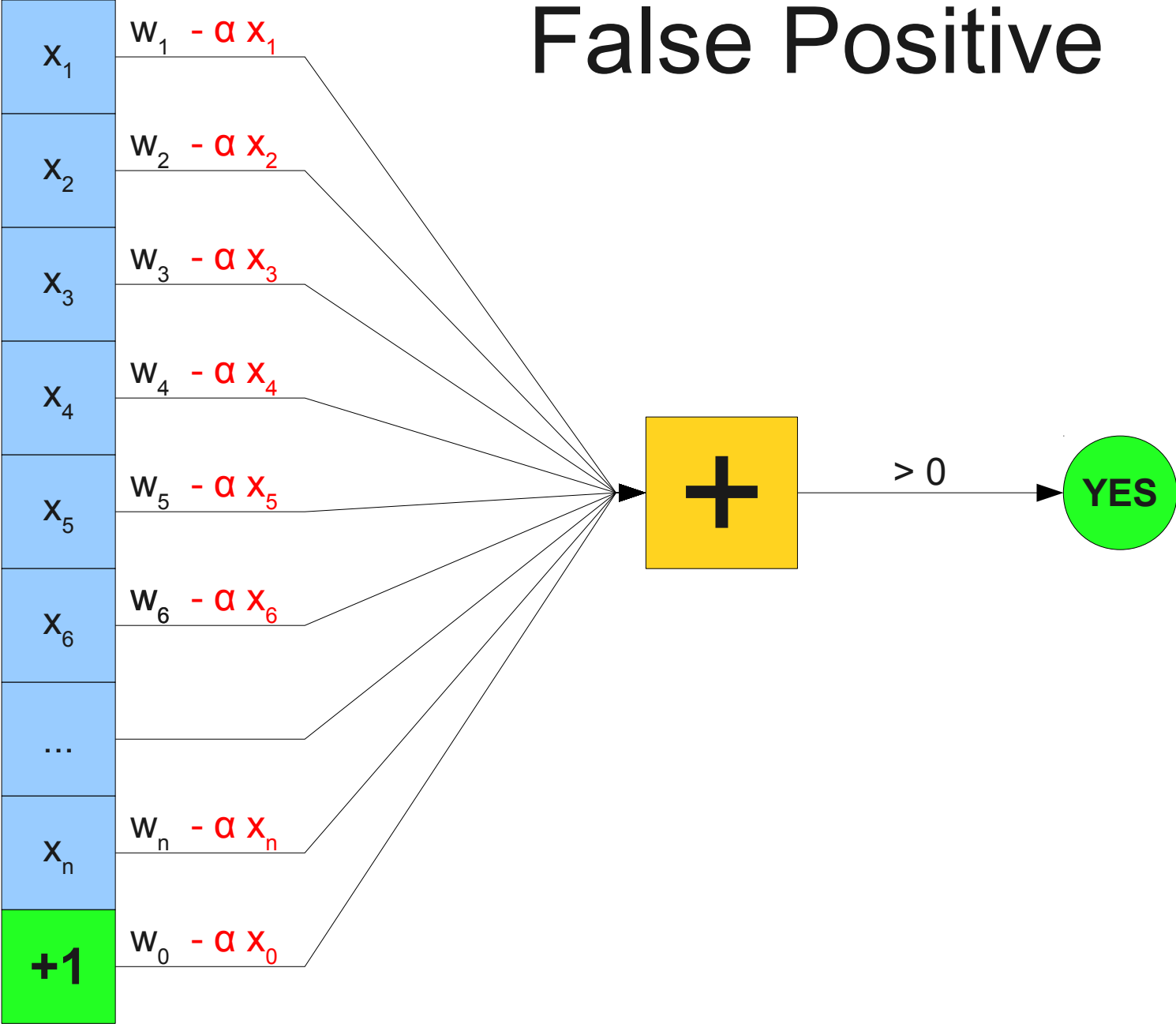
- **Train** the perceptron on valid data.
- For each data point:
 - Ask the perceptron what it thinks.
 - If correct, do nothing.
 - Otherwise, nudge $w_0 \dots w_n$ in the right direction.
- Repeat until number of errors is “small enough.”
- Question: What kind of mistakes can we make?

False Positive

False Positive

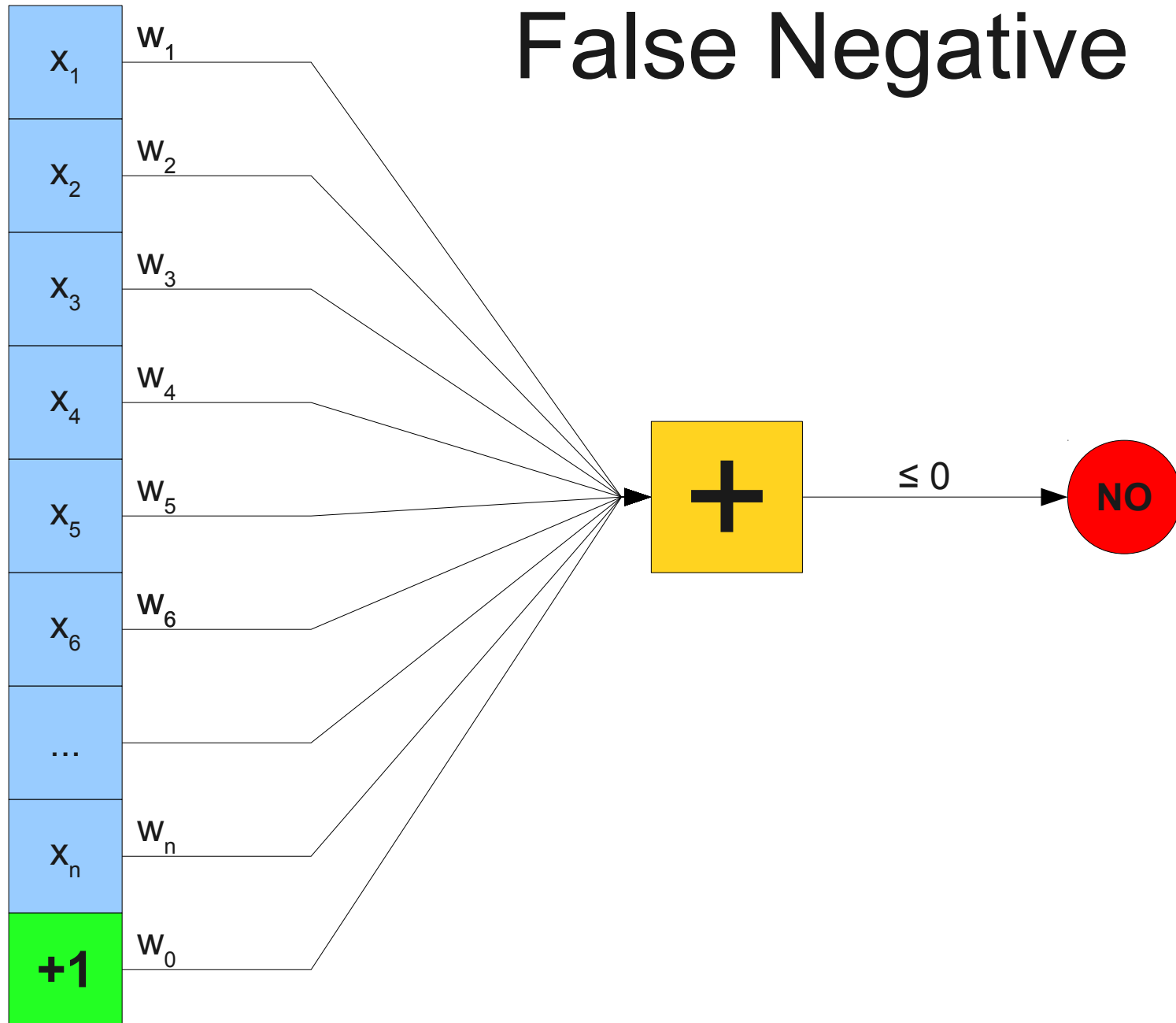


False Positive

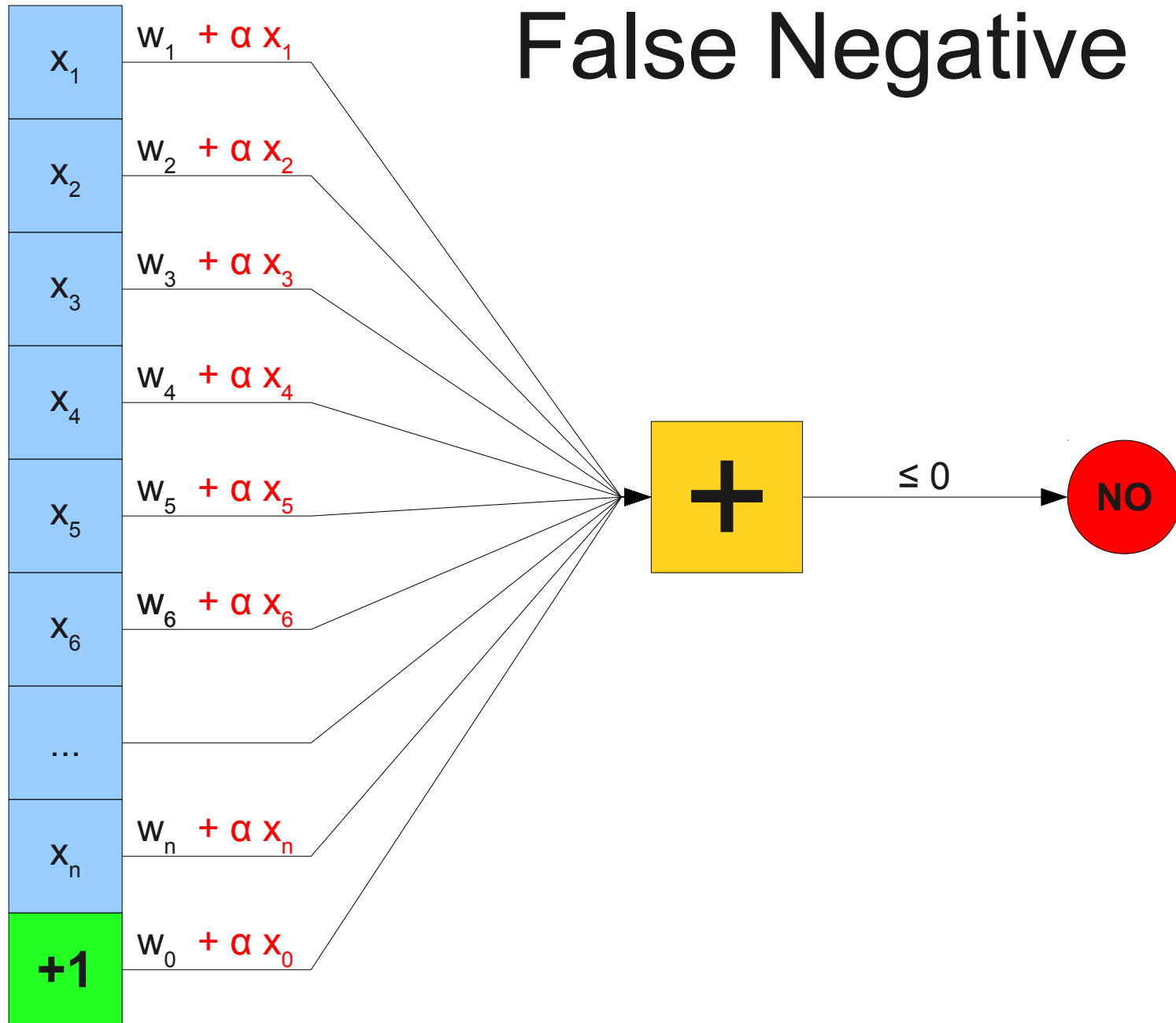


False Negative

False Negative



False Negative



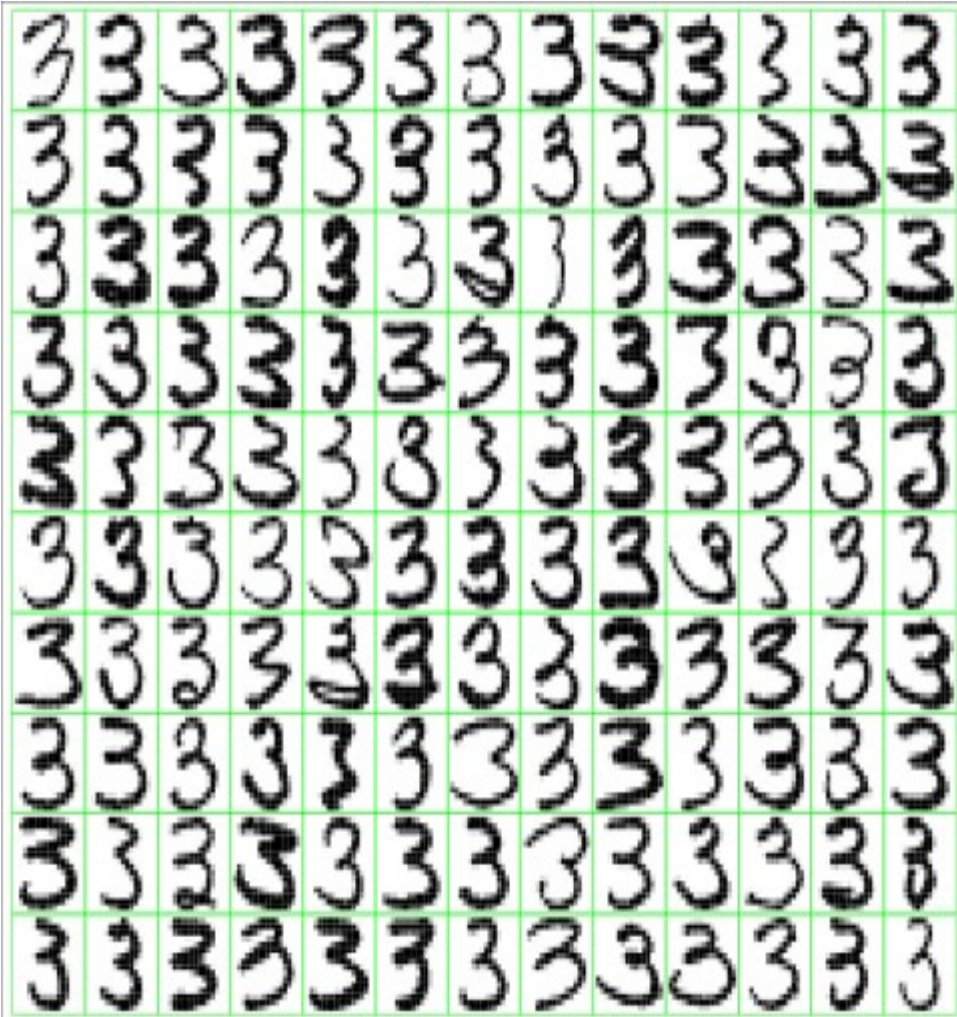
Scary Mathematical Hackery

- For false positives, $w_i := w_i - \alpha x_i$
- For false negatives, $w_i := w_i + \alpha x_i$
- Let “YES” be 1 and “NO” be 0.
- Consider the difference between **actual answer** and **perceptron guess**:
 - False positive: Actually NO, we say YES. Difference is -1.
 - False negative: Actually YES, we say NO. Difference is +1.
- **General update rule: $w_i := w_i + \alpha (\text{real} - \text{guess}) x_i$**

Perceptron Learning Algorithm

- Start with a random guess of each w_i .
- Repeat until perceptron is sufficiently accurate:
 - Choose a training example (x_0, x_1, \dots, x_n) .
 - Let **real** be the real answer, **guess** be the perceptron's guess.
 - For each i , set $w_i := w_i + \alpha (\text{real} - \text{guess}) x_i$
- Note: Use **batching** in practice.

Application: Handwriting Analysis



- Train a computer to recognize handwritten numbers 0 – 9.
- Large training and test set available (MNIST Handwritten Digit Database)

