n=1 (0.3, 0.4, -0.5) → 0.25

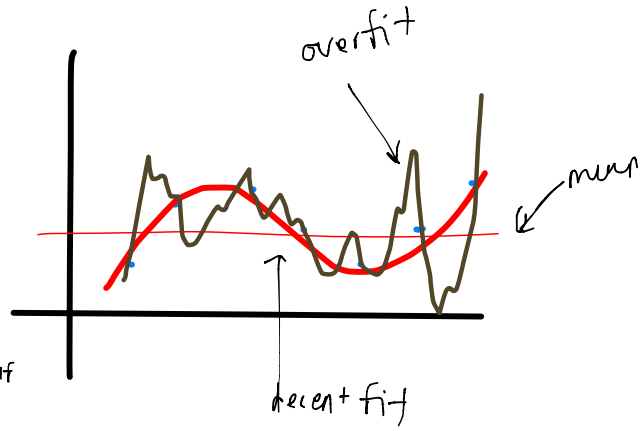
n=2 (0.2, 0.8, 0.6) → 0.32
input output

Only one output → only 1 A node

a second piece of output would require another summation variable (A node)

Slow because # pattern nodes = # inputs

Weights going to pattern units are the training data itself



Pattern-layer nodes/wts process the incoming patterns as:

simply subtract each input pattern/element from the corresponding wt, then take either the squares of these differences or absolute values of the difference across all weights

$$I_j = \sum_{i=1}^{n} |w_{ij} - X_i| \quad \text{or} \quad I_j = \sum_{i=1}^{n} (w_{ij} - X_i)^2$$

"net input"

Overfitting

σ = small, more accurate fit
large, smoothes

The net input is fed into an activation function (exponential: $f(I_j) = \exp\left(\frac{-I_j}{2\sigma^2}\right)$

Summation Units (A and B): see board for how to set these (equal to output wts for A, 1 for B

Both A, B perform a simple dot product between the wt vector and the output signal from the pattern units.

The two dot products go directly to the output layer.

A output

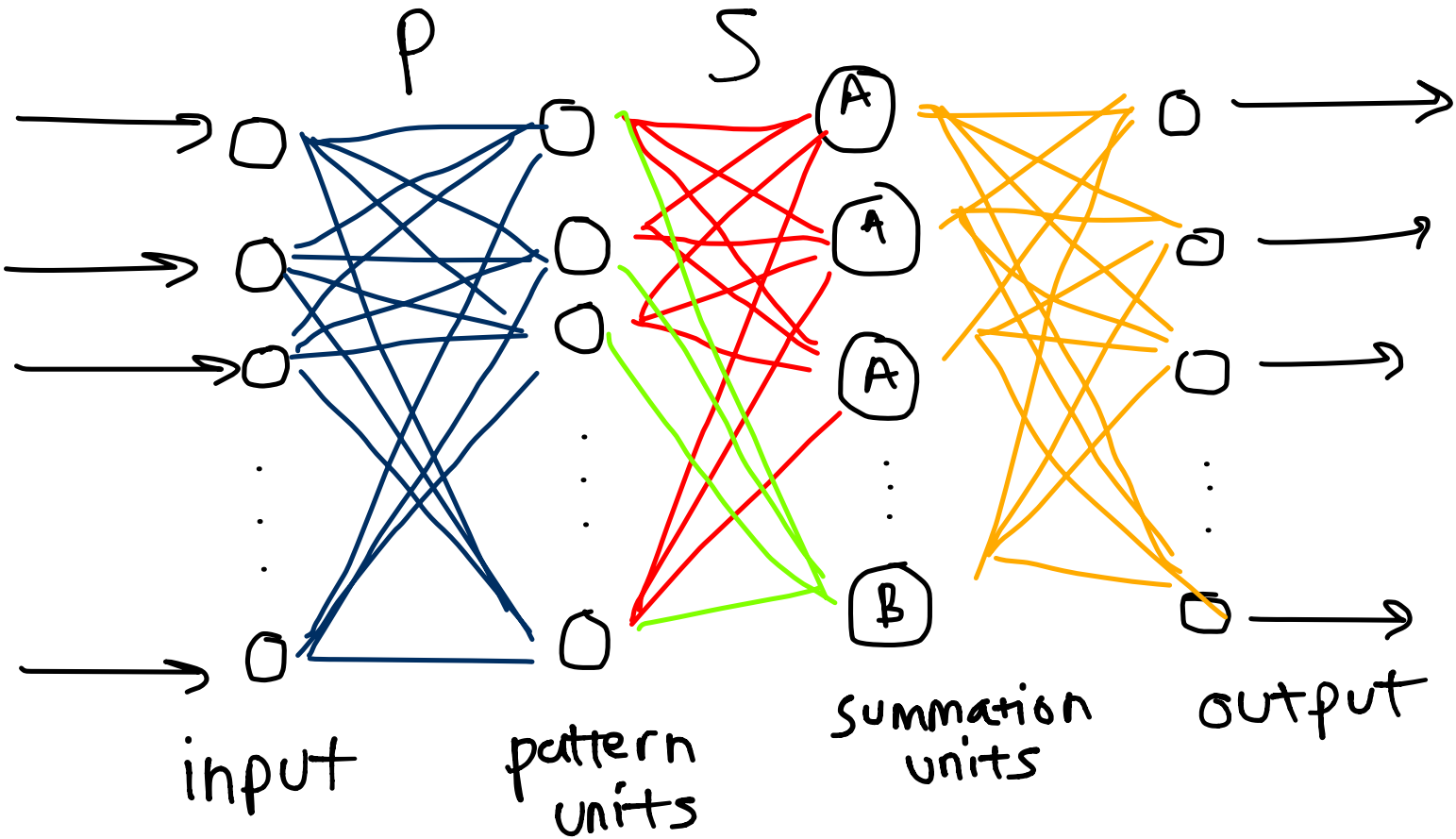
B output

* Read paper for Thursday

When you have multi-dimensional output:

1. The wts of A become corresponding components of the output vector
2. Set the weights prior to the pattern nodes to the centroid of the cluster
3. Also, change how you set the wts on the summation layer nodes. The B-units act as counters. For example, each time a training pattern appears for cluster 3, the wt from the third pattern-layer node to the B-summation node increment by one.

*either sum A for 1 output or add an output node


P
S
A
A
A
B
input
pattern units
summation units
output