special lecture 03 | numerical methods 1 | spring 2024 | quick overview of perceptrons

perceptrons are the basic building blocks of neural networks.

frank rosenblatt, a psychologist who studied cognitive systems, is he author of "perceptrons". while others earlier than him had ideas about algorithms for self-learning, **he wrote a computer program that learned to classify images**. in 1958. a computer program. it was called mark 1. so he usually gets the prize.

…

forward propagation:

so x\*w is linear, b is an offset and the activation function adds nonlinearity, **another way to associate x to y**.

wrt neurons, resulting output decides if a condition is met. ie, sigma-y is sufficient to, say, buy stock. in fact, it can be viewed as **hyperplane**.

bias (may not have been rosenblatt) and even tho its not included in the first diagram, he probably did factor in the activation function.

for perceptrons use a binary step activation function - ie, its 1 or 0 - but here lets use a sigmoid. 1/(1+e^-z).

…

backward propagation:

thats about **backward error**. ie, how a change in output y relates to expected change in input x. ie, some stuff from the first week of this class. fancy that.

so given data (x,y). approximate y-hat and compare to given y for residual r. what influences the residual? ignore the activation function for now and go for the low hanging fruit: w,b. ie, w,b gets solved for to minimize the residual.

but lets back up for a second. first choose how to define error.

error = f(y-hat,y). a common one is MSE.

SE = (y-hat[i] - y[i])^2 ~> MSE = 1/n \* sum SE = C, the cost function.

thats if you need regression (continuous output ~ weather, stock price); for classifcation problems (its a dog, its a cat, its a radioactive flying squirrel), your loss function might be class entropy.

so then the "learning part". adjusting w,b to minimize the cost function.

repeat: its not always "x" wrt x[new] = f(x[old])

notation for w[i] = [] vs w0[i], b0. and (i+1) values for w,b are obtained by, lets say, gradient descent, thats a popular flavor.

p2 x\*w = y is a linear system of equations [lecture 05, topic 02]

p3 bias [other places but: lecture 01, topic 00|01 fp notation]

p3 backward error [lecture 02, topic 00|02]

p4 gradient descent ~> conjugate gradient is a slightly more complicated version [lecture 06, topic 02|03]]

so the only new news might be the activation function. otherwise, youve seen all these lego pieces.