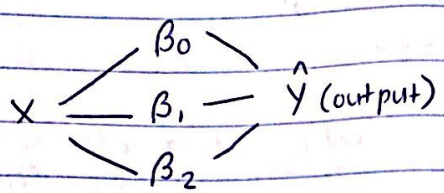
*unsupervized learning

Machine Learning 9/18 Artificial Neural Networks

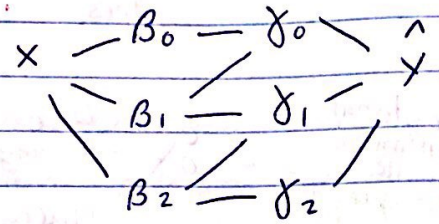
$\sum_i^N (f_i^{true} - \hat{f}(x_i))^2$ = residual sum squared RSS

task: minimize this quantity

Bayesian classification
Linear classification (regression) - not enough flexibility
Nearest neighbor classification - too much flexibility (not consistently reproducible for other unknown datasets)



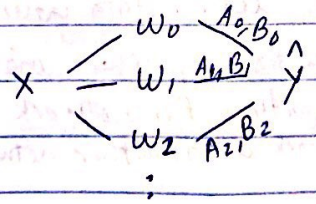
making model more effective by adding more parameters to get output



more complicated → but, still linear, just redefining parameters but not making it more expressive/flexible

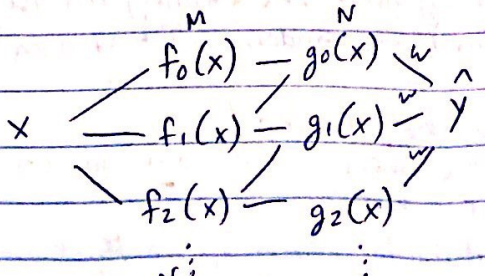
How to make more expressive?

$\hat{y}(x) = \sum_i (A_i \cos(w_i x) + B_i \sin(w_i x))$ Fonier (sp?) function



each line is linear weight applied to item before it

(would need an infinite number of these) to describe function → genesis of idea of Artificial Neural Networks



↓ replace w with functions $f(x)$ and also include multiple layers of connections between nodes

$\hat{y} = \sum_i^N w_i g_i(*)$

$* = \sum_j^M k_j f_j(\text{inputs})$

→

☆ Chapter 11 in Hastie et al
☆ Bishop, Pattern Recognition

Sigmoid function $G(x) = \frac{1}{1+e^{-x}}$ → no longer used for intermediate layers b/c vanishing gradient

only free parameters are weights (connections between nodes)

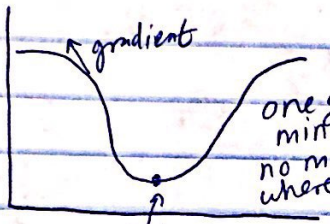
Universal approximation theorem
→ system on prev. page capable of modeling ANY function

Artificial Neural Networks = stacking nonlinear functions on top of one another
inside → "hidden layer"

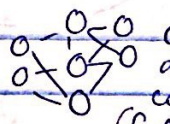
minimize $RSS(w_i)$ ← weights that we don't know

gradient → $\frac{\partial RSS}{\partial w_i}$ fit to data

feed forward → model has inputs
send back → use labeled data, send backward to see how well model does



can also be local minima, saddle points, complications w/ finding global min



can choose any # nodes, connections, layers
(fully connected model = all links)
lots of nodes & connections, lots of degrees of freedom

lg models → high variance

Tradeoff
Capacity → amt your model can do
generability → ability to do well on data your model has never seen
↓ ☆ need to stop the training before model becomes overfit
sm models (can train one portion of a network at a time, can train a network with another network, lots of possibilities)

Deep Learning → lots of layers w/ lots of nodes but then don't let it overtrain by limit the training dataset (stochastic inputs, take just one random batch of data at a time)

Even if you don't find global min w/ these complex networks, a local min can still work as model fit

Inputs can be anything → where to start?
don't want to start at place where it will converge on local minimum, not global min
- can scale your input data to clean up so that everything on the same level of magnitude (but need to keep consistent when you enter test data later after training data)

Just because a min exists doesn't mean you will find it (∞ or numbers very close to zero complicate)

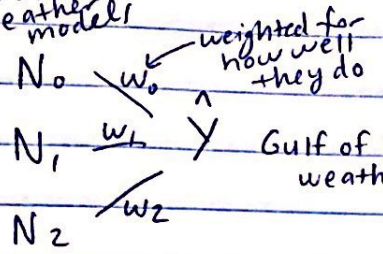
weight decay → if you have same network but only change this lambda term, get different results for fit (no weight decay term, overfit in example)

$RSS(w_i) + \lambda \sum_i w_i^2$

λ → penalty term, hyperparameter

→ can help w/ generalizability and make better model for future datasets,
keeps your weights from getting too large
BUT
means that you are now changing the minimum you are looking for

3 different weather models



weighted for how well they do

Gulf of Mexico weather, simple example

* Lots of different options & dials to turn but not any correct way to change them

Training error vs test error

↓ learns training set well

↓ better at fitting new dataset

What parameters do we get to choose? = hyperparameters → can be learned from data, but hard to do

- weights are determined by model

$\hat{y}$ → choose what you are trying to predict to choose which function

to predict probability → softmax = $f_i(\vec{x}) = \frac{e^{x_i}}{\sum_j e^{y_j}}$ (always less than 1, adds to 1)