q Probably wrong in a few places, sorry!

1a)

i.

|  |  |  |
| --- | --- | --- |
|  | 3-NN | 5-NN |
| 1 | + | + |
| 2 | + | + |
| 3 | x | + |
| 4 | x | x |
| 5 | x | x |
| 6 | + | + |

ii. Becomes less sensitive to training data; lower variance; fits a smoother curve

iii. Sum weights in neighborhood per class

X = .7

+ = .3m

Prediction is x

IGfrequency = .1805

1b) h(D) = -3/10\*log(3/10) + -4/10\*log(4/10) + -3/10\*log(3/10) = 1.571

H(D, HF) = -5/50\*log(5/50) + -30/50\*log(30/50) + -15/50\*log(15/50) = 1.2955

H(D, LF) = - 25/50\*log(25/50) + -10/50\*log(10/50)+ -15/50\*log(15/50) = 1.4855

IG = H(D) - (0.5 \* H(D, HF) + 0.5 \* H(D, LF)) = 0.1805

2)

a)

X \* weight = [-0.07]

No bias

Activation = -.06989

\* weight = .30097

Output = 0.30097

b) Assumption: the output (yhat) is NOT normalized

Assume loss function is MSE

DL/dy = 2(y – y) = 2(.3 - 5) = -9.39806

DL/dW = [-.06989 .7]t \* -9.39806 = [.6568 -6.5786]

W = w – LR(dl/dw)wo xi

q

*Dl/da = dl/dy \*dy/da = -9.39806 \* -0.3 = 2.8194*

*DL/dz = dl/da \* g’(z) = 3.7592 \* .9951 = 2.8056*

*DL/dw = [1.4028 1.6834]*

G'(z) = .9951

DL/dX = 2.8194

DL/dZ = dL/dA o g’(z) = dL/dX \* .9951= 2.8056

DL/dW =xt \* dL/dZ = [1.403 1.683]

Updated weights:

w.4 = 3.69

w.3 = -0.63

w.1 = -0.6

w.2 = -1.04

2c) precision = 2/3

* Recall = ½

F1 = 4/7 (.5714)

3a)

|  |  |
| --- | --- |
| index | assignment |
| 1 | 3 |
| 2 | 2 |
| 3 | 2 |
| 4 | 2 |
| 5 | 1 |
| 6 | 3 |
| 7 | 3 |
| 8 | 3 |
| 9 | 1 |
| 10 | 3 |
| 11 | 2 |
| 12 | 3 |
| 13 | 1 |
| 14 | 3 |

Updated clusters

C1 = 2,2

C2 = 8,2

C3 = 6,8

3b) I)

bd = get\_bd(solution)

fitness = get\_fitness(solution)

cluster = find\_cluster(bd)

If cluster.isEmpty():

cluster.add(solution)  
else:

current\_point = cluster.get\_point()

If get\_fitness(current\_point) < fitness:

cluster.remove(current\_point)

cluster.add(solution)

ii)

|  |  |  |
| --- | --- | --- |
| **Cluster** | **Max index** |  |
| c1 | 13 |  |
| c2 | 3 |  |
| c3 | 7 |  |
| c4 | 6 |  |

iii) performance = avg(fitnesses) = 1.05

Diversity = 4

QD = sum(fitnesses) 4.2

c) I) same parent (index 14) as it has the highest fitness function (greater than all children)

ii) Maximising the fitness function (or the negative of a loss function) that we have over a given number of iterations / until a max fitness is reached. Use other operators (selection such as biased roulette wheel, tournament; mutation such as changing random numbers as opposed to adding Gaussian noise; crossover between genotypes at certain gene points).

UNSURE IF THIS IS RIGHT ^