s

\1.

a ) dLoss/dW = XT\*dLoss/dZ

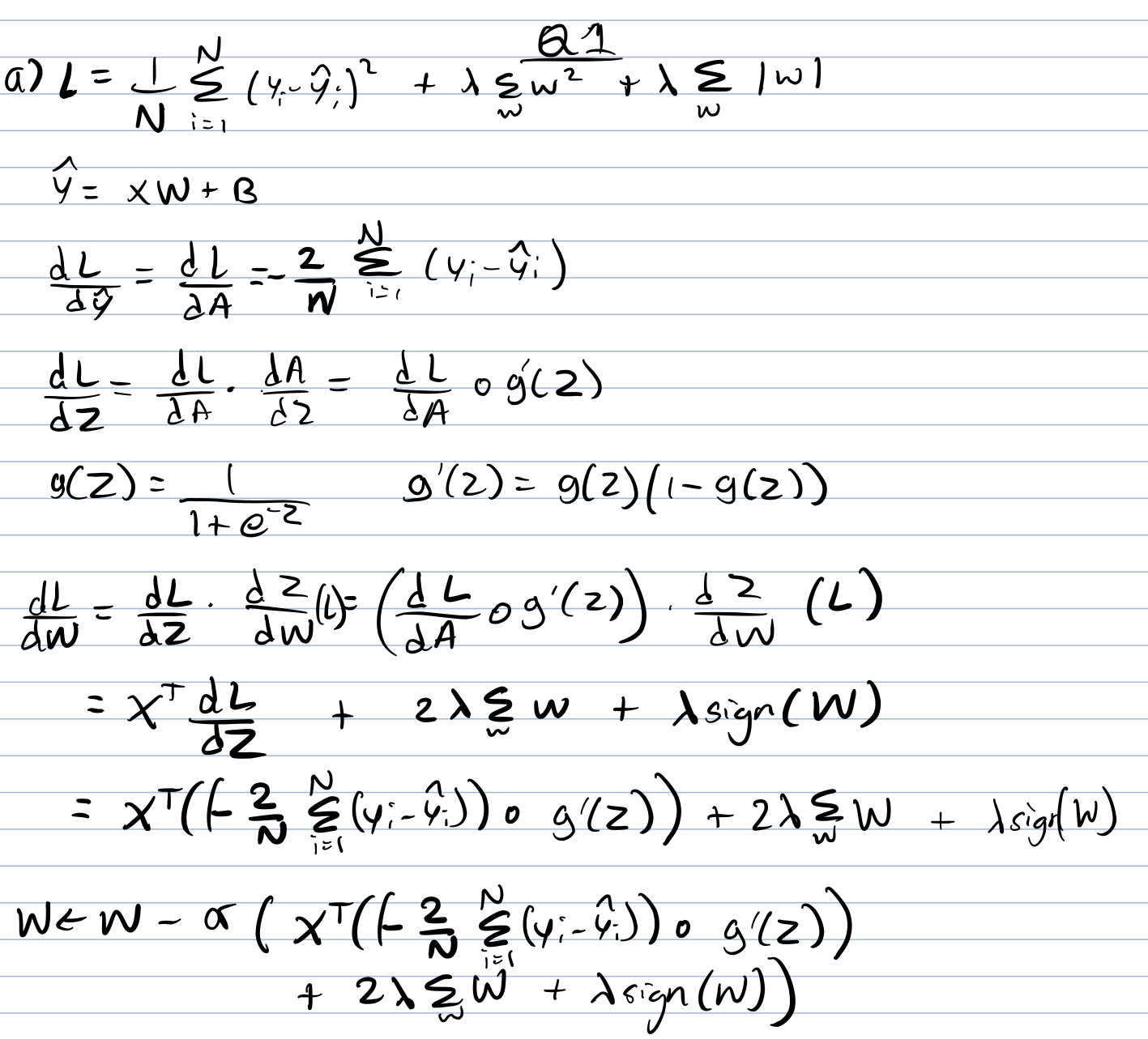
= XT\*dLoss/dpredicted\*dpredicted/dz

= XT\*(-2/N\*(predicted-labels)\*dpredicted/dz

= XT\*(-2/N\*(predicted-labels)sigmoid(z\_out)(1-sigmoid(z\_out))

W <- W - (alpha)\*(dLoss/dW + 2\*lambda2w + lambda1\*sign(w))

N.B. to anyone confused about the element-wise product below: The element-wise product of a scalar and a vector works because the vector is in fact a single value since it must be the case that the output layer only has 1 neuron (we’re using MSE loss => we’re solving a regression problem => there’s only 1 output neuron).



b) Learning rate is the step size we take when updating a parameters value to optimise our model (by measure of MSE or cross entropy). When too low, learning is too slow and may take long before converging to high accuracy. When too high, it may bounce-off and never even converge. In practice, adaptive learning rates are used - ones that, as convergence starts, the learning rate progressively decreases (ie, see ADAM).

c) Early stopping is a means for preventing overfitting. It consists of evaluating our model’s hyperparameters on the validation set. If the accuracy hasn’t increased in a consecutive number of epochs, we halt the process there (missing a bit of the answer and copying the overfitting diagram).

d)

Z = 2.58

Classifier A: error rate=12% -> [0.00, 0.24] (0.12 +- z\*sqrt((0.12\*0.88)/50))

Classifier B: error rate = 16% -> [0.14, 0.17] (0.16 +- z\*sqrt((0.16\*0.84)/5000))

B better than A

[because the difference in error at the 99% level for B is significantly smaller than that of A?]

[Also A has a maximum error rat e of 24% while B has a maximum error rate of 17%]

2.

a)

-We shuffle the data

-We perform k-fold validation on, say, 5 folds.

-We iteratively leave one fold out for testing and we perform training+validation on the remaining k-1 folds - on these it is we do hyperparameter tuning.2

-We repeat this for all permutations.

-We aggregate and average out our accuracies.

1. -Thereon, when deploying, we train our model on the full dataset.

b)

Acc = 83.72%

Avg recall = 0.64

F1: 1->0.33 2->0.933 3->0 Choose F1 score since it is imbalanced and one of the classes is completely misclassified.

c) not covered

d)

First split:

Total entropy: - 3/5 log(3/5) - 2/5 log(2/5) = 0.9710

Weighted entropy

Sky = air = wind : 0.649 <= choose any one of those as 1st split point

Humid = forecast : 0.95097

Water: 0.8

Second split:

Total: 0.81

We can see that we would have entropy 0 using wind as split point so this is the optimal split point

Final tree

Total:

* Sunny sky:
  + Strong wind +
  + Weak wind -
* Rainy sky -

-----------------------------------------------------------------------------------------------------------------

Disagree with the above answer, here’s what I got

E = enjoy sport, s= sky, a = air, h = humid ...

H(E) = - ⅗ log(3/5) - ⅖ log(2/5) = 0.97

IG(E, s) = IG(E, a) = IG(E, w) = 0.97 - (⅕ \* 0 + ⅘ \* (-¾ log¾ -¼ log¼)) = 0.322

Using a similar method:

IG(E, wa) = 0.17095

IG(E, h) = IG(E, f) =0.019

Thus we split on sky (or air or wind) as it has the highest info gain. Then I did the rest by inspection and stated that the IG values were higher than the others.

**Sky/Air/Wind** -- rainy→ -

-- sunny → **Wind** -- strong → +

-- weak → -

e) not covered