1

<https://cs230.stanford.edu/files/cs230exam_win21_soln.pdf>

a

1. 1? - Batch Normalization computes the mean and standard deviation over a batch of training examples, which cannot be done for a single test example. Therefore, the Batch Norm layers are skipped during test time, and the learned parameters from the training phase are used to normalize the test data

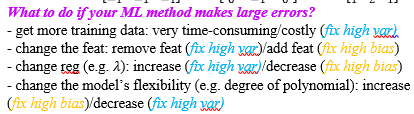
3, 4?

1. 4 [(k\*k\*c\_in + 1) \* c\_out = (1\*1\*16 + 1) \* 1 = 17]
2. 1, 2
3. 2, 4

Isn’t 3 just another way of saying y = x, I don’t think it would be a valid act fn (assuming we want non linearity), because max(x, 0.1x) for x > 0 is x. Min(x, 0.1x) for negative x is x (which is more negative than 0.1x), so you’re effectively just getting a y = x curve. Can someone attest?

1. 1 – is this the answer because for all the other ones we compute a division so we get floating points operations?
2. 2, 3, (is 4 not also valid somewhat? We could be getting high training set error as we have not trained on enough data to converge to minimum with, idk).

Does high training set error mean high variance? If so, we can also increase lambda in regularisation (answer 1) and I think 2 is wrong



1. 3, 4 ?

b)

i) These parameters allow the network to learn to adjust the mean and variance of the activations to better fit the desired output distribution

ii) not enough variety in the training data: fix by getting more data

overfitting: fix by removing features, increasing the lambda in the regularisation factor

iii) use multiple smaller conv instead of bigger ones to increase speed (less operations)

1x1 conv are used for pooling on the channel dimension

1x1 conv acts like dense layers

iv) derivative is very small when the input is very large or very small -> slow convergence or even complete stagnation -> vanishing gradient

c) input: 64x64x8

Conv: in = 8, out = 32, k = 3, s = 1, p = 1: (64 – 3 + 2 \* 1) / 1 + 1 = 64

output is 64x64x32

(3 \* 3 \* 8 + 1) \* 32 = 2336 parameters

Max pool: k = s = 2: (64 – 2 + 2 \* 1) / 2 + 1 = 33

outputx32

0 parameters

Batch norm

output is same as input: 33x33x32

2 (gamma, beta~~, mean, var?~~) \* 32 parameters

d) i) b1 has the dim of W1 \* xi: (D(alpha1) x Dx) \* (Dx x 1) = D(alpha1) x 1

Alpha1 has the same dim as b1 (from alhpa1 = Relu(b1)), so D(alpha1) x 1

B2 has the dim of W2 \* alpha1, (1 x D(alpha1)) \* (D(alpha1) x 1) = 1 (scalar)k

ii) to fix inbalance (if we have more data of one class than the other one)?

iii) alpha = 1/11, beta= 10/11

iv) help anyone? y^ is (y^(1), y^(2), …, y^(N)), how do we differentiate wrt y^ if we have y^(i) in the formula?

J is a scalar value so it’s essentially a derivative of a scalar w.r.t a vector

So you would have a vector of \partial J/ \partial \hat y^{(I)}.

v) f(x) + x?

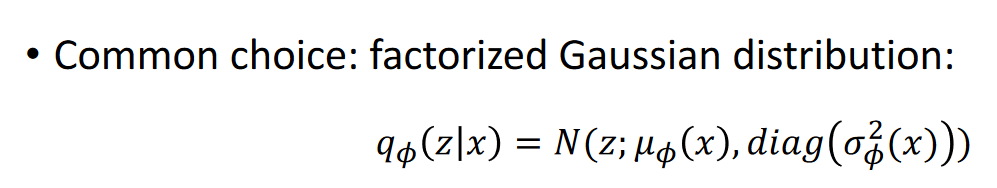
Alternative : I would say f(x)-x, if h is your hypothesis and f the actual function to be approximated.

2a

i) P(x1, x2, x3, z1, z2, z3) = p(x1|z1) p(x2|z2) p(x3|z3) p(z1) p(z2|x1,z1) p(z3|x2,z2)

Q(z1,z2,z3|x1,x2,x3) = q(z1|x1) q(z2|x2,z1) q(z3|x3,z2)

iii) 1 ()why?



2b) i) use L1 to compute the loss between x and F(G(x)), and also the loss between y and G(F(y))?

ii) adversarial training uses (x, label\_x) here x is the input img and and we are also given the label

We will have the inputs (x, real), (y, real), (F(G(x)), fake) and (G(F(y)), fake)

Compute the loss using binary cross entropy

2c)

ii) The derivative used in BPTT depends on the largest singular value in the weight matrix. If you solve for eigenvalues in this matrix the standard way you’ll find that the largest singular value is > 1, which means that there will be gradient explosion during BPTT.

Iii) 3 and 4 (see lecture notes)

2d)2

I) QK^T, then apply 1 hot on it (set col with largest value to 1, rest in the row to 0 for all rows), multiply the resulting matrix with the V one.

II) time complexity:

CNN: O(HWhwc) - kernel goes around the image (HW) and compute for one filter (hw) in channels(c)

Attention: Let a = H/h, b = W/w. Number of queries is ab. So Q’s dimension is (ab, hw), since self attention, K’s dimension is (ab, hw). Time complexity of QK^T is O(abh^2w^2) = O(HWhw). Since V’s dimension is (hw, v), hmm dimension doesn’t match? Shouldn't it be (ab, v)?

Q: (H/h \* W/w, hw)

K: (H/h \* W/w, hw)

V: (H/h \* W/w, v)