1.

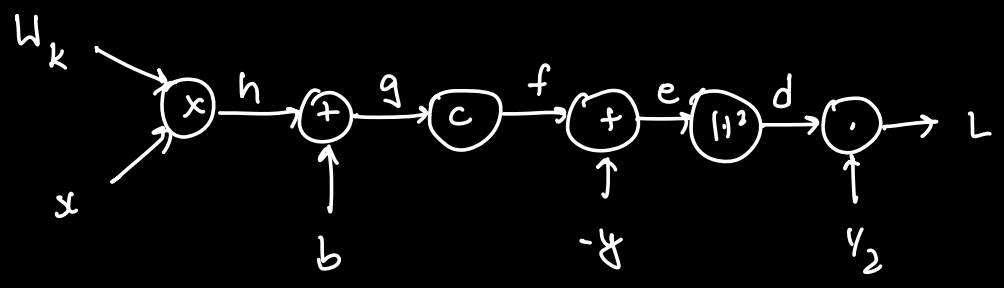
Classification as a task has a discrete output space. So we are trying to predict one of n classes. Therefore, we are either correct or wrong. In case of regression, the goal is to find a value in a more continuous space and hence even the most accurate model may be off by a few decimals.

2. SGD basically takes a subset of the training data to train the model in each iteration. In case of GD, each value of the error is calculated using the entire dataset. GD provides the more accurate estimate of the gradient. We can assume the gradient to be contributed from each training data and hence more the data, more accurate the estimate of the gradient. But this could lead to overfitting and hence noise in GD is helpful.

3. It is difficult to train neural networks with low initializations. This is because the activation values drop quickly to 0 when going into deeper layers. Therefore, during gradient descent, no learning happens as gradient is also 0.

4. This is a bad idea. As all values are equal, all the neurons in the hidden layer observe the same input and hence the same gradient descent and in the end learn the same features.

2)



$$c = \max(\alpha g, g)$$

$$d = |e|^2 \qquad \frac{\partial L}{\partial d} = 1/2$$

$$\frac{\partial d}{\partial e} = 2e$$

$$f = \alpha g \quad , g < 0$$

$$\frac{\partial L}{\partial e} = e \qquad = g \quad , g \geq 0$$

$$\frac{\partial L}{\partial f} = e$$

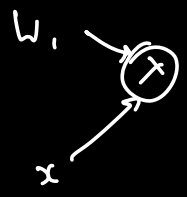
$$\frac{\partial f}{\partial g} = \begin{cases} \alpha & , g < 0 \\ 1 & , g \geq 0 \end{cases}$$

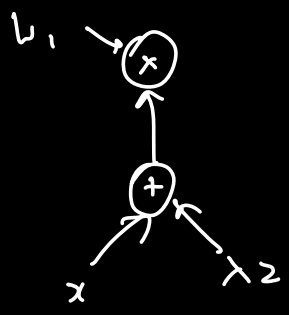
$$\left(I(g<0)\cdot\alpha + I(g\geq 0)\right)$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial h} = \frac{\partial L}{\partial g} = \left(I(g<0)\cdot\alpha + I(g\geq 0)\right)(f-y)$$

$$\frac{\partial L}{\partial W_k} = \left(I(g<0)\cdot\alpha + I(g\geq 0)\right)(f-y)\,x^T$$

c) w_1, x





w_1 update change

x doesn't

$(x + \lambda y)$ instead of

(x)

$a = w_1 (x + \lambda y)$

$\frac{\partial L}{\partial a} \lambda y$ if 0 then same

3) a) hyperparameter → Validation data.

Not too large or small.

b) Not always. False

Very high

c) Dropout acts as ensembling. The model trains only on few inputs. So doesn't overfit

4) a) $-\log(0.3)$

b) $-\log(0.4)$

c) Correct has more error.

→ We only consider the value of correct.

Intention maximize $y^{(i)}$ for $x^{(i)}$.

Not max amongst other classes.

d) How do you train?

Gradients?

c) A → Batch

B → Minibatch

f) change in learning rate

g) → ∞

→ 0

3) a) Label

b) We can use CNN with transfer learning

if not YES!