GRADIENT DESCENT

* Sample, Batch, Epoch

→ A sample is a single row of data.

→ The batch size defines the number of samples to work through, before updating the internal model parameters.

- Depending on the batch size different learning algos are defined.

→ The number of epochs is a hyperparameter that defines the number of times the learning algo will work through the entire training dataset.

- One epoch

* An epoch

One epoch:

1. Randomly divide training set into $m = N/K$ batches.
2. Use a batch of training samples to compute $J(\omega)$.
3. Update ω as: $\omega^{t+1} = \omega^{t} - \eta \frac{\partial J}{\partial \omega}$
4. Repeat 2 & 3 using different subsets all samples are used once.

Q. What is the size of the batch?

$1 \dots K$

$K+1 \dots 2K$

$2K+1$

- $1 \dots \dots K \dots \dots N$
- May depend on hardware.
- We repeat epochs until convergence.

* ## Batch Gradient Descent

→ Training set: Ex: ImageNet has 14 M images

→ Approach:

- Compute the loss J(w) on the entire training, update the parameters w.
- At the next epoch, shuffle the training data, and repeat ~~a~~bove process.

→ Typical batch size: Size of the training set.

* ## Mini-batch Gradient Descent

→ It is wasteful to compute the loss over the entire set to perform a single parameter update for large datasets.

- Ex: ImageNet has 14 M images
- GD is replaced with mini-batch GD

→ Mini-batch ~~eg~~ GD

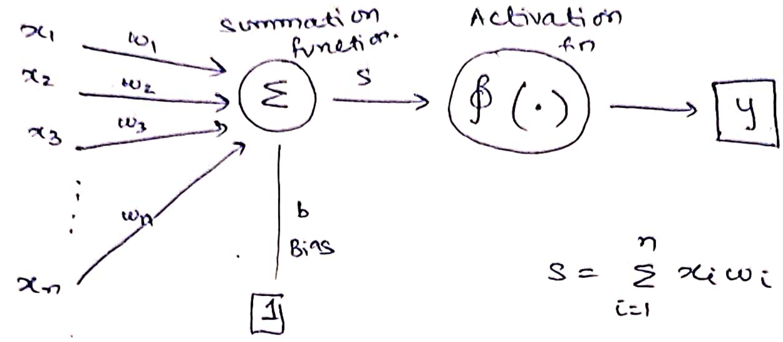
- Approach:
  - Compute the loss L(w) on a batch of images, update the parameters w

How to learn w & b? How will our model learn?

learn loss fn

gradient descent

How many neurons in one layer.



$$s = \sum_{i=1}^{n} x_i w_i + b \qquad y = \phi(s)$$

error / loss / cost / objective function

→ supervised learning

→ The loss function provides the cost of being wrong, by measuring the quality of a particular set of parameters based on how well the output of the network agrees with the ground ~~labe~~ truth labels in the training data.

$$L(\theta) = \text{distance}\,((f_\theta(x), y))$$

Input, features

label, ground truth

error

label (true)

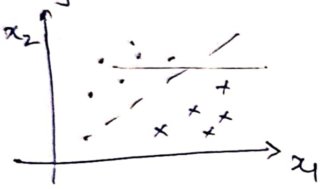
$y_{pred} = f_\theta(x)$

learned by the model

difference b/w actual & predicted

θ = parameters (weights, biases)

learning process



$y = w_0 + w_1 x_1 + w_2 x_2$

depending upon feedbock.

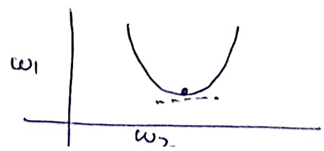
start random value (for slope).

1. Start with random values of w_i (training data)
2. Evaluate the goodness of the line, determined with a loss function, $J(w)$.
3. The weights w_i are changed acc moving the line to a better position.
   → Note: $J(w)$ should be min when the training samples are correctly classified.
4. Repeat 2 & 3 until $J(w) < Y$

Gradient Descent → linear regr^n / NN / CNN / RN

learns parameters.

low loss function: Gradiont Descent in Action.



w_1
w_2

Force you to go to dir^n in which max descent is less.