Face verification vs. face recognition

→ Verification 1:1

- Input image, name/ID
- Output whether the input image is that of the claimed person

→ Recognition 1:K

- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or "not recognized")

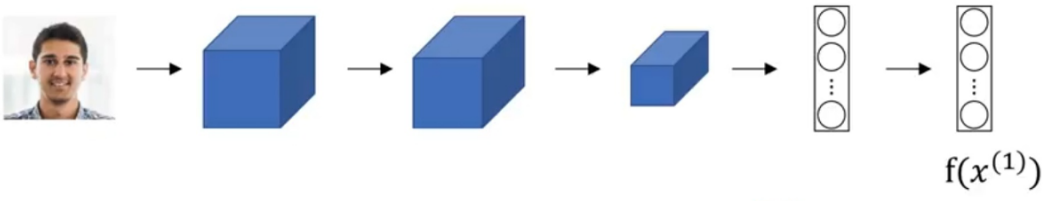
The main challenge of face recognition is one shot learning ⇒ you need to be able to recognize a person, given just one image of that person.
Solution: learning a similarity function, d.

d(img1,img2) = degree of difference between images

If d(img1,img2) $\leq \tau$ "same" } Verification.

$> \tau$ "different"

- Siamese network: running 2 experiments and comparing their results, without softmax.



Parameters of NN define an encoding $f(x^{(i)})$ — 128

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

One way to learn the parameters of the neural network, so that it gives you a good encoding for your pictures of faces, is to define and apply gradient descent on the triplet loss function.

Anchor - positive : small

Anchor - negative : big

$$\|f(\text{Anchor}) - f(\text{Pos})\|^2 - \|f(\text{Anchor}) - f(\text{Neg})\|^2 + \alpha \leq 0$$

α: margin, to avoid trivial outputs = 0.

So, let's say $\alpha = 0'2$, at least the difference between between (a-pos) and (a-neg) must be 0'2.

Triplet loss function: Given 3 images A, P, N:

$$\mathcal{L}(A,P,N) = \max\left(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0\right);$$

So the cost function is:

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)}) \rightarrow \text{minimized with gradient descent.}$$

Note that you need more than 1 image by person.

How to choose A,P,N?

During training, if A,P,N are chosen randomly, $d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

$\|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2$

Choose triplets that're "hard" to train on.

$\mathcal{L}(A,P) + \alpha \leq \mathcal{L}(A,N)$

$\mathcal{L}(A,P) \approx \mathcal{L}(A,N)$

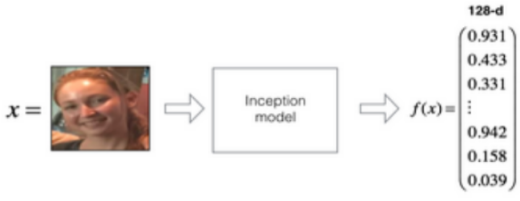
Face Net

Deep Face

3.2 - The Triplet Loss

Important Note: Since you're using a pretrained model, you won't actually need to implement the triplet loss function in this assignment. *However*, the triplet loss is the main ingredient of the face recognition algorithm, and you'll need to know how to use it for training your own FaceNet model, as well as other types of image similarity problems. Therefore, you'll implement it below, for fun and edification. :)

For an image x, its encoding is denoted as $f(x)$, where f is the function computed by the neural network.



Training will use triplets of images (A, P, N):

- A is an "Anchor" image--a picture of a person.
- P is a "Positive" image--a picture of the same person as the Anchor image.
- N is a "Negative" image--a picture of a different person than the Anchor image.

These triplets are picked from the training dataset. $(A^{(i)}, P^{(i)}, N^{(i)})$ is used here to denote the i-th training example.

You'd like to make sure that an image $A^{(i)}$ of an individual is closer to the Positive $P^{(i)}$ than to the Negative image $N^{(i)}$) by at least a margin α:

$$||f\left(A^{(i)}\right) - f\left(P^{(i)}\right)||_2^2 + \alpha < ||f\left(A^{(i)}\right) - f\left(N^{(i)}\right)||_2^2$$

You would thus like to minimize the following "triplet cost":

$$\mathcal{J} = \sum_{i=1}^{m} [\underbrace{|| f(A^{(i)}) - f(P^{(i)}) ||_2^2}_{(1)} - \underbrace{|| f(A^{(i)}) - f(N^{(i)}) ||_2^2}_{(2)} + \alpha]_+ \tag{3}$$

Here, the notation "$[z]_+$" is used to denote $max(z, 0)$.

Notes:

- The term (1) is the squared distance between the anchor "A" and the positive "P" for a given triplet; you want this to be small.
- The term (2) is the squared distance between the anchor "A" and the negative "N" for a given triplet, you want this to be relatively large. It has a minus sign preceding it because minimizing the negative of the term is the same as maximizing that term.
- α is called the margin. It's a hyperparameter that you pick manually. You'll use $\alpha = 0.2$.

Most implementations also rescale the encoding vectors to haven L2 norm equal to one (i.e., $|| f(img) ||_2$=1); you won't have to worry about that in this assignment.