Face recognition:

face verification(1:1) VS face recognition(1:n)

one-shot learning problem: learn from one example to recognize the person again.

we can learn a similarity function:

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

Siamese network

x1----> f(x1): the encoding generated from network

x2----> f(x2): the encoding generated from the same network (same parameters)

d(x1, x2)=||f(x1)-f(x2)||^2

what you want to do is to learn parameters so that if any pair (xi, xj) are the same person, then d(xi, xj) is small. And what you can do is to use backpropagation to vary these parameters in order to make sure these conditions are satisfied.

Triplet loss

A: anchor, P: positive, N: negative

||f(A)-f(P)||^2<=||f(A)-f(N)||^2

but if both are zeros or both are equal, then the network cannot be trained

so we add the hyperparameter alpha as margin, pushing two sides further away from each other, like SVM

||f(A)-f(P)||^2-||f(A)-f(N)||^2+alpha<=0

So the final loss function can be written as:

L(A, P, N)=max(||f(A)-f(P)||^2-||f(A)-f(N)||^2+alpha, 0)

Minimize the loss function to update parameters so that the distance of two encodings of two images is small if two images are similar.

when you train the network, you need the training set like 10K pictures of a 1K person, during training, if A, P, N are chosen randomly, d(A, P)+alpha <=d(A, N) is easily satisfied. So you should choose triplets that are hard to train on, that is d(A, P)~d(A,N)

After train the network, you can use it in a one-shot learning problem.

Face verification and binary classification

x1--->f(x1)

---->logisitic regression classification(binary classification, alternative to triplet loss)

x2--->f(x2)

rather than just take in the two encodings, you can compute the difference of two encodings like the following: