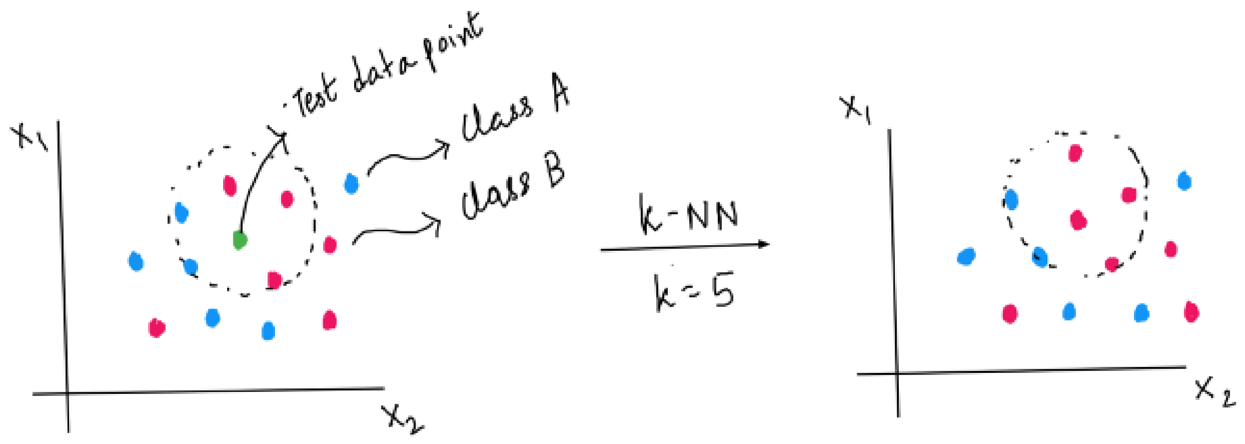
METRIC LEARNING:

What does a basic machine learning algorithm do? — Given the data and the corresponding output labels, the goal is to come up with a set of **rules** or some **complex function** that maps those inputs to the corresponding output labels.

One of the simplest machine learning algorithms where **distance** information is captured is the **KNN** (k- Nearest Neighbours) algorithm, where the idea is to find a neighborhood of **k** nearest data points for a new data point and assign this data point to the class to which the majority of the **k** data points belong.



A simple explanation of k-NN

Similarly, the goal of metric learning is to learn a **similarity function** from data. Metric Learning aims to learn data embeddings/feature vectors in a way that reduces the distance between feature vectors corresponding to faces belonging to the same person and increases the distance between the feature vectors corresponding to different faces.

Euclidean distances are less meaningful in high dimensions. They fail to capture nonlinearity in data because of a number of reasons:

1. They represent an **isotropic** (same in every direction) distance metric.
2. These distances don't capture the class structure.

One can use non-isotropic distances that use properties of the data to capture the structure of class relationships. **Mahalanobis Distance Metric** is one such metric. The distance between two samples on the metric space is given by:

In a 'd' dimensional space; $x, y \in \mathbb{R}^d$

$$D(x,y) = \left((x-y)^T M (x-y)\right)^{1/2}$$

Mahalanobis Distance Metric

Here, M is the inverse of the covariance matrix and acts as a weight term to the square Euclidean distance. The Mahalanobis distance equation represents a dissimilarity measure between two random vectors **x** and **y**, following the same distribution with Covariance Matrix **Σ.**

The Covariance Matrix in 'd' dimensional space:

$$M = \Sigma^{-1} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \vdots & & & \vdots \\ \sigma_{d1} & \cdots & \cdots & \sigma_{dd} \end{bmatrix}^{-1}_{d \times d} ;$$

σ_{ij} is the covariance between variables i, j.

$$\because \sigma_{ij} = \sigma_{ji} \Rightarrow M = \text{Symmetric}$$

M is an inverse of the Covariance Matrix.

If M = **I**(Identity Matrix) we have the special case of Mahalanobis distance — Euclidean Distance, which proves an underlying assumption of the Euclidean Distance that the features are independent of each other.

Matrix M can be decomposed as:

$$M = W^T W$$

$\therefore$ Mahalanobis Distance becomes -

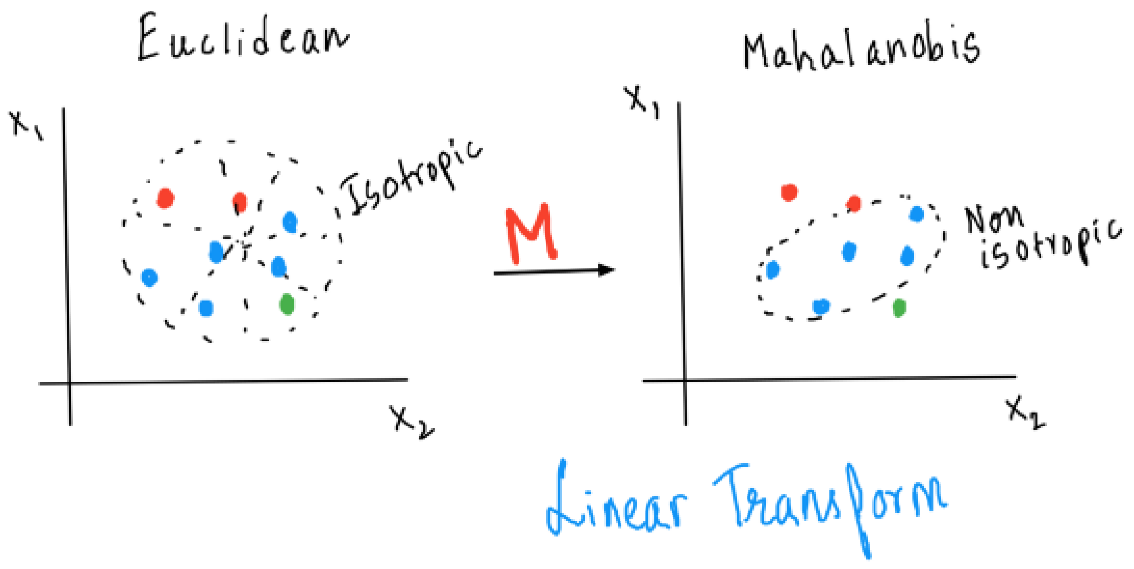
$$D(x,y) = \left((x-y)^T W^T W (x-y)\right)^{\frac{1}{2}}$$

$$\Rightarrow D(x,y) = \left((W(x-y))^T (W(x-y))\right)^{1/2}$$

$$\Rightarrow D(x,y) = \| Wx - Wy \|_2 \sim eq(*)$$

Linear Transformation property of the Mahalanobis distance metric

The equation(*) can be interpreted as a linear projection in euclidean space. So, W has a linear transformation property because of which Euclidean distance in transformed space is equal to the Mahalanobis distance in the original space.



Isotropic Euclidean distance V/S Non-isotropic Mahalanobis distance metric

This distance metric provides a new data representation in the transformed space which is easily able to distinguish the items of different classes. This Linear transformation shows the inter-class separability power this approach posses. So, Metric Learning is basically a goal to learn this transformation matrix **W**.

Metric learning is an approach based directly on a distance metric that aims to establish similarity or dissimilarity between images. **Deep Metric Learning** on the other hand uses Neural Networks to automatically learn discriminative features from the images and then compute the metric.

Why there is a need for deep metric learning?

1. Faces of the same person when presented in different poses, expressions, illuminations, etc. might be able to fool a face verification/face recognition system very efficiently.
2. The task of building an efficient, large scale Face Verification system is radically a task of designing appropriate loss functions that best discriminate the classes under study. In recent years, Metric/Distance learning using Deep learning has been shown to output highly satisfying results for many computer vision tasks such as face recognition, face verification, image classification, Anomaly detection, etc.
3. Metric Learning only has a limited capability to capture non-linearity in the data.

Deep Metric Learning helps capture Non-Linear feature structure by learning a non-linear transformation of the feature space.

DEEP METRIC LEARNING

There are two ways in which we can leverage deep metric learning for the task of face verification and recognition:

1. Designing appropriate loss functions for the problem.

Most widely used loss functions for deep metric learning are the contrastive loss and the triplet loss.

** Contrastive Loss — Siamese Networks:

One of the very fundamental ideas where **explicit metric learning** is performed is the siamese network model. Siamese network is a Symmetric neural network architecture consisting of two identical subnetworks that share the same sets of parameters (hence computing the same function) and learns by calculating a metric between highest level feature encodings of each subnetwork each with a distinct input.

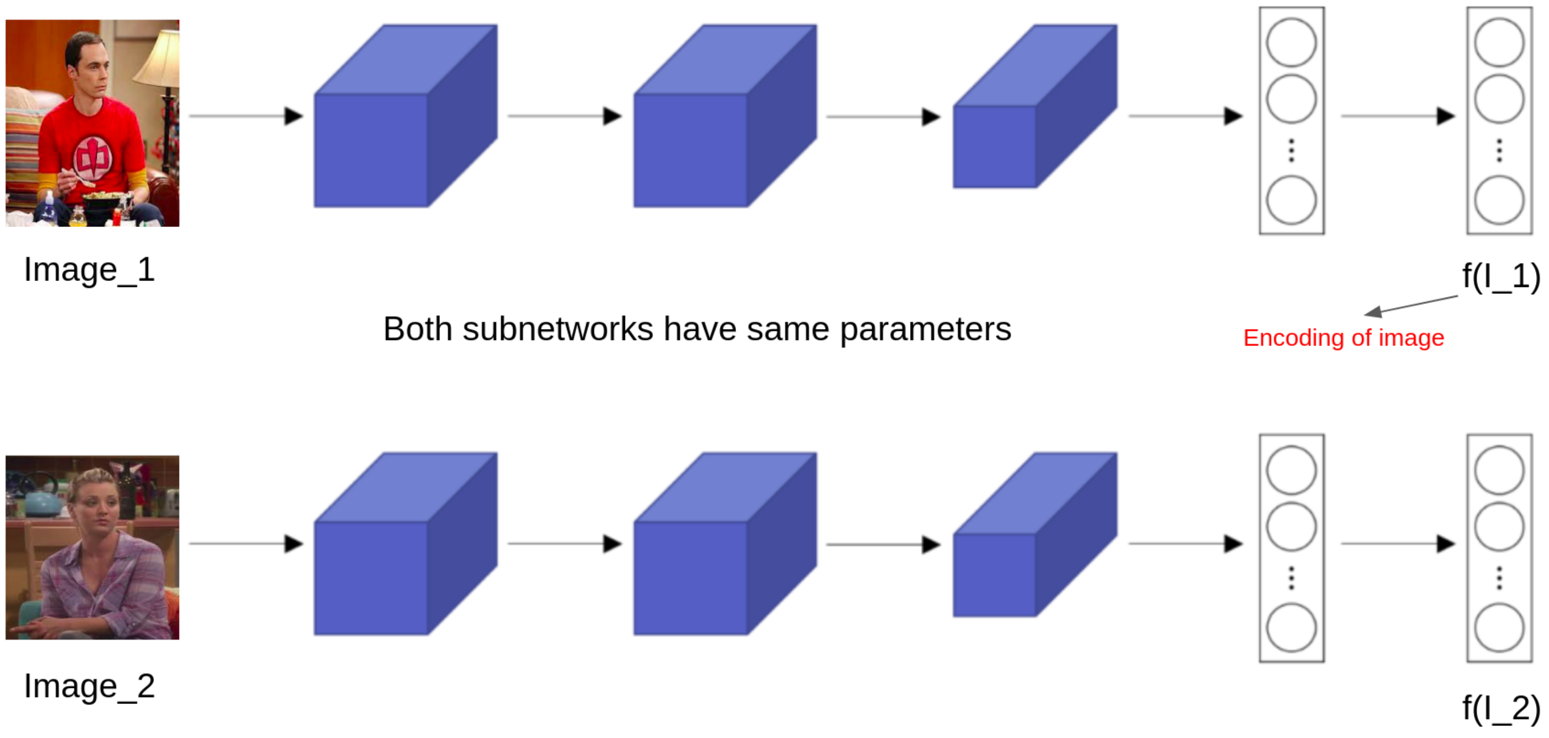


Image by Author: Subnetworks in a siamese net share the same sets of parameters

Mathematically, **d(x, y) = ||f(x)-f(y)||** ^{2} , Here, **“x”** and **“y”** represent the training examples/images. **“d(.)”** represents the metric distance between two images and **“f”** represents the encoding of the input images. The goal is to learn parameters so that:

- If **x** and **y** are the same person — d(x, y) is **small**,
- If **x** and **y** are a different person — d(x, y) is **large**.

Siamese Networks are mainly used in combination with a **Contrastive Loss function aka Pairwise ranking Loss.** The loss function is mainly used to learn embeddings (feature vectors) in a way that the metric distance between two examples from the same class is small and that between different classes is large in a metric space. Now, let’s understand the maths.

The Contrastive loss is given by:

$$L = L(f(x), f(y), Z) = Z\underbrace{\|f(x)-f(y)\|_2}_{\text{distance bw (+) pairs}} + (1-Z)\underbrace{\max(0, m-\|f(x)-f(y)\|_2)}_{\text{distance bw (-) pairs}}$$

Here, $Z \in \{0,1\}$; $Z=0 \Rightarrow x, y$ are positive pairs
$Z=1 \Rightarrow x, y$ are Negative pairs

Contrastive Loss

Here, m>0 is the margin, z is a boolean which is 1 if x, y are positive pairs and 0 otherwise, ||.|| is the L2 norm, other notations are the same as above.

Let's understand this equation. If x and y are actually positive pairs, **z is 1,** So, the second term vanishes (**margin m** has no effect). If the network computes a large distance between x and y, the loss will be high as the first term has a significant contribution to the loss.

Now, if x and y are actually negative pairs, **z is 0.** So, the first term vanishes. If the network computes a small distance (~0) between x and y, we will have, **L=m,** again a significant contribution to the loss.

Role of margin m: It is to be noted that the representations of negative pairs will only contribute to the loss if the estimated distance **||f(x)-f(y)|| < m**. Meaning that it will no longer care how far the negative pairs x and y are once this limit reaches. So, it can focus more on the difficult to embed points.

Note, Siamese Network is mainly used when the number of data points per class is very small — **One-Shot Learning**.