Deep Learning Loss functions

There are several loss fns defined over time to train DNN's effectively and efficiently. Starting from L_1, L_2 then CE & SMax going towards L SMax, A SMax and AM SMax.

Aditya nigam

Loss fn

L_1 Vs L_2

L_2 Vs Cross entropy

Cross entropy Vs Softmax Loss

Softmax Vs L-Softmax

L-Softmax Vs A-Softmax

A Softmax Vs AM-Softmax.

We are going to cover these transitions

Supervised Learning → Classification, Regression

Features → x
Targets → y
Predictions → $\hat{y}$
Parameters → θ

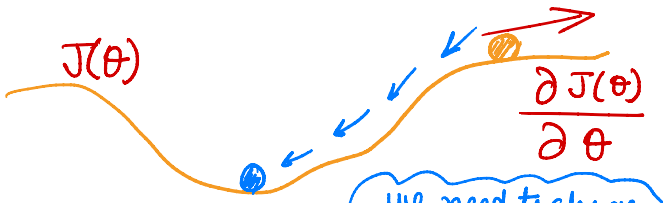
$x \rightarrow f_\theta \rightarrow \hat{y}$

$y = f(x)$ — we are interested to somehow predict using our θ parameters.

Loss fn $\Rightarrow$ Say $J(\theta)$ $\rightsquigarrow$ function of $(y, \hat{y})$

$\therefore$ $\theta \rightarrow$ Randomly initialized

$\hat{\theta} = \arg\min_{\theta} J(\theta)$



we need to choose the direction that will decrease $J(\theta)$

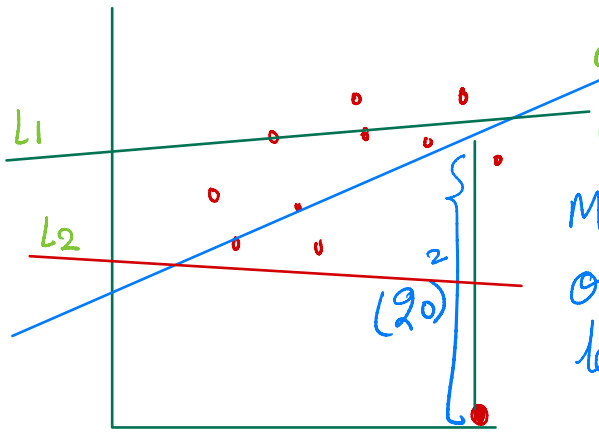
$$\theta^{new} = \theta^{old} - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Gradient descent → Samplewise, full data, Batchwise

SGD

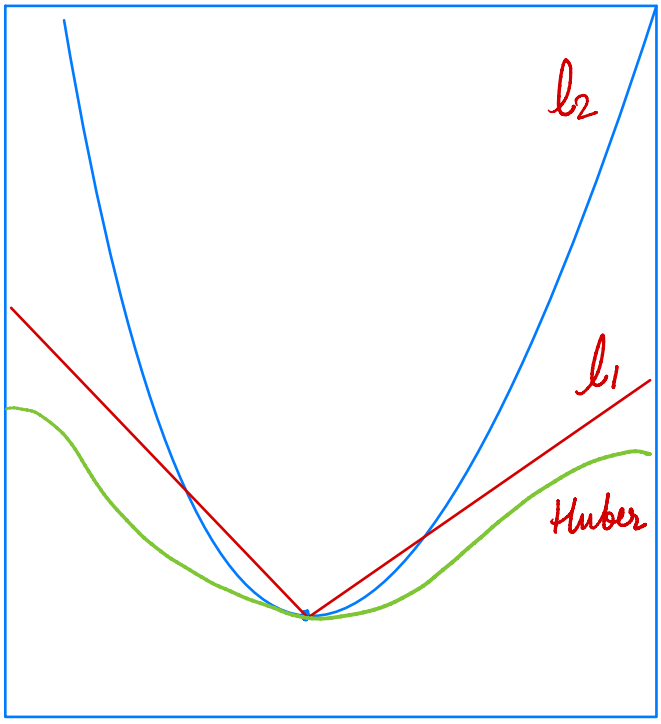
Loss fn or Cost fn choices can be → $L_2 : \frac{1}{2}(y - \hat{y})^2$ (MSE), $L_1 : |y - \hat{y}|$ (MAE)

[Also Called as Manhattan distance.]



(Sensitivity)

MSE heavily penalizes outliers. This causes outliers to manipulates the learning heavily biased towards them



Huber loss takes the best of both

Near to origin $\Rightarrow$ Close to l_2

away from origin $\Rightarrow$ Close to l_1

L_2 loss function Vs Cross Entropy

* Ground truth $\Rightarrow y$ and prediction $\Rightarrow \hat{y}$
* total data points (m) we will Consider $m=1$

$$MSE = \frac{1}{m}\sum (y_i - \hat{y}_i)^2$$

$$\Downarrow$$

$$(y-\hat{y})^2$$

(Regression)

(Naturally Coming for linear regression)

Binary Cross Entropy

$$= \frac{-1}{m}\sum \left[y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i) \right]$$

$$\Downarrow$$

$$\left[y \ln(\hat{y}) + (1-y)\ln(1-\hat{y}) \right]$$

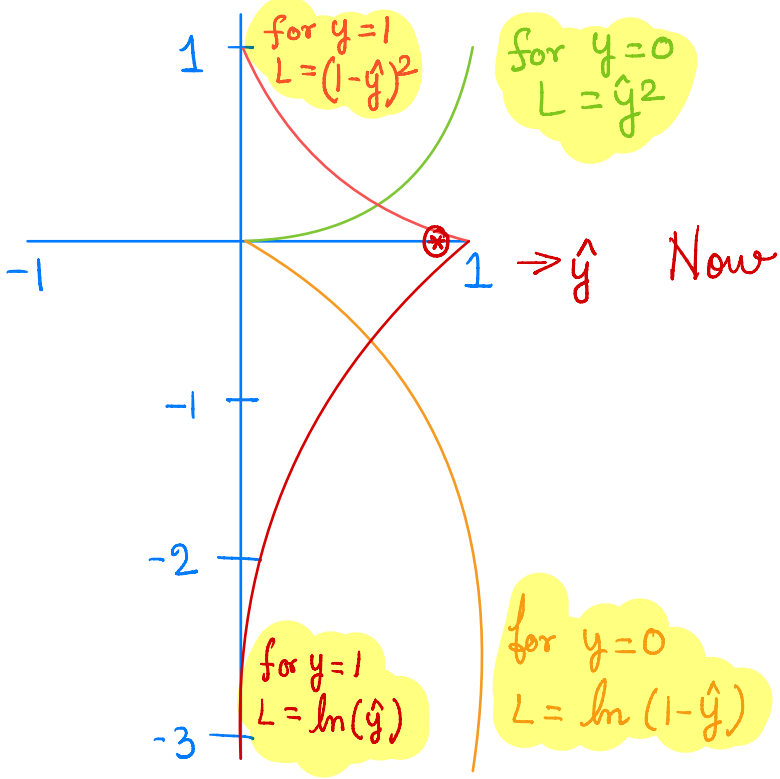
Classification

(-ve sign is omitted as we are only interested in the shape of curve)

(Naturally Coming for logistic regression)

$$\frac{1}{1+e^{-x}}$$

* for classification MSE donot penalizes much the misclassification as Compared to (CE).



$MSE \rightarrow L = (y-\hat{y})^2$

$BCE\ Loss \rightarrow L = y\ln(\hat{y}) + (1-y)\ln(1-\hat{y})$

Now example is that model is predicting wrong

$y = 0$ and $\hat{y} = 0.9$

for MSE → it is predicting a wrong value

for classification → Instead of predicting class "0" it is predicting class "1".

$L_{MSE} \Rightarrow (0.9)^2 = [0.81]$ $\quad L_{CE} = [2.3]$

$\frac{\partial L_{MSE}}{\partial \hat{y}} = 1.8$ $\quad \frac{\partial L_{CE}}{\partial \hat{y}} = 10.00$

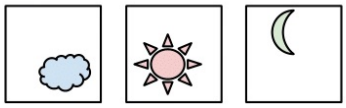
$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w}$$

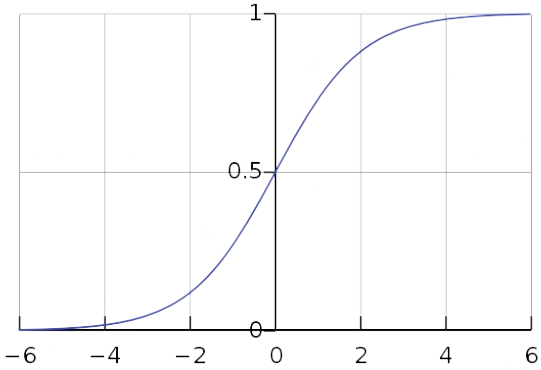
Then which loss function can change/update the weights effectively in the case of mis-classification

Regression : Continous Variable & Classification : Discrete Variable

Small error may lead to small update. Updation scale of both of them is different.

Small variation basically may lead to some other class (we are dealing with classes/categories)

	Multi-Class	Multi-Label
C = 3	Samples	Samples
		
	Labels (t)	Labels (t)
	[0 0 1] [1 0 0] [0 1 0]	[1 0 1] [0 1 0] [1 1 1]



$$f(s_i) = \frac{1}{1 + e^{-s_i}}$$

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

Sigmoid is squashing function [logistic fn]

Softmax fn is not a loss function

It also squashes things b/w [0 & 1], adding upto (1).

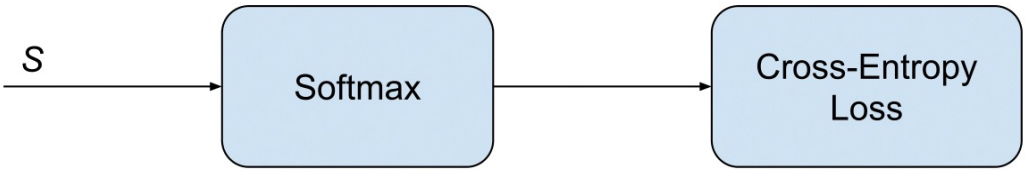
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1)$$

CE loss

Binary Cross Entropy

Activation function (Sigmoid / Softmax) is applied before CE / BCE losses.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Here (s_p) is the (score for the +ve class)

$$\frac{\partial}{\partial s_p}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_p}}{\sum_j^C e^{s_j}} - 1\right)$$

The derivative of CE wrt (s_p) & (s_n)

$$\frac{\partial}{\partial s_n}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_n}}{\sum_j^C e^{s_j}}\right)$$

Focal loss : It is a Cross-Entropy loss that weights the Contribution of each sample to the loss, based on its own classification error.

(*) Sample Correctly classified then its weight is decreased.

(*) Mis-classified $\Rightarrow$ weight is increased.

$$FL = -\sum_{i=1}^{C=2}(1-s_i)^{\gamma} t_i log(s_i)$$

Focusing factor

Ground truth

Score

$(\gamma \geq 0)$ focusing parameter

$(\gamma = 0)$ BCE

$$FL = \begin{cases} -(1-s_1)^{\gamma} log(s_1) & if \quad t_1 = 1 \\ -(1-(1-s_1))^{\gamma} log(1-s_1) & if \quad t_1 = 0 \end{cases}$$

$$FL = (1-s_1)^{\gamma} t_1 \log(s_1) + (1-s_2)^{\gamma} t_2 \log(s_2)$$

$t_1 = 1$

$$(1-s_1)^{\gamma} \log(s_1)$$

$s_2 \Rightarrow (1-s_1)$

$t_1 = 0$

$$(1-s_2)^{\gamma} \log(s_2)$$

$$(1-(1-s_1))^{\gamma} \log(1-s_1)$$

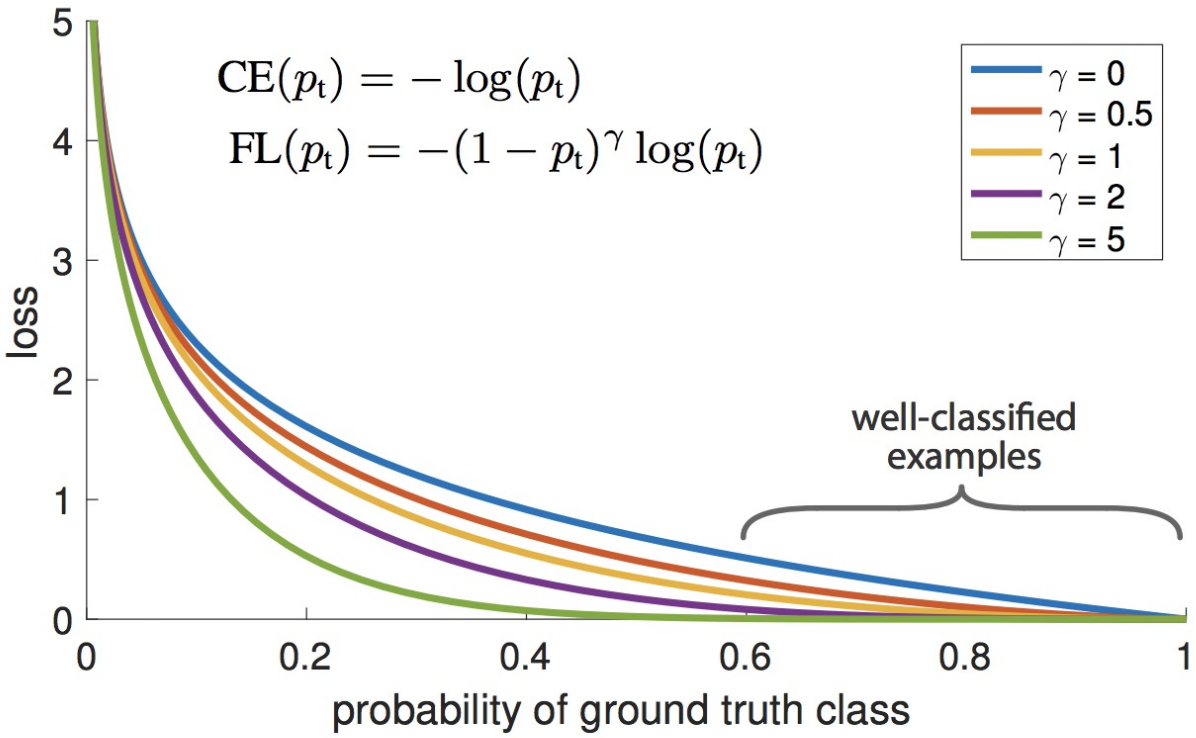
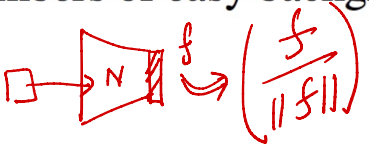
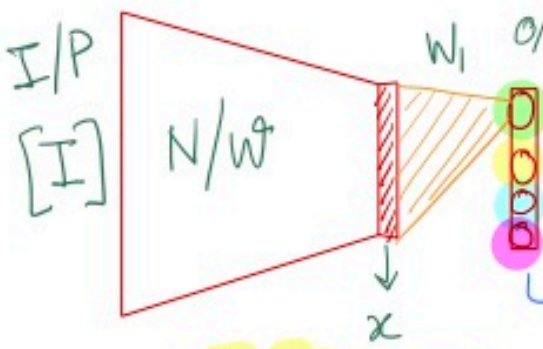


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.



⊛ Just talk about **Triplet loss** and its variants
⇒ Moving towards **metric learning**. [old slides]



Any (W_i) is the weight vector of that $(i)^{th}$ neuron for that $(i)^{th}$ class.

All weights together W $(N \times 1)$ $(4 \times N)$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

y' (Raw value) is the O/P of the N/W we get (4×1) (W/O Softmax)

$$y' = [y_1' \; y_2' \; y_3' \; y_4']$$

$$y_i' = W_i x$$

Apply Softmax

$\hat{y}$ ⇒ after Softmax

Preactivated O/P of our N/W (N-dim) $[N \times 1]$

$x : [x_1 \; x_2 \; \cdots \; x_N] \in \mathbb{R}^N$

y ⇒ 1-Hot encoded ground truth (4×1)

$$\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4]$$

$$\hat{y}_i = \frac{e^{W_i \cdot x}}{\sum_{j=1}^{C} e^{W_i \cdot x}}$$

(i^{th} neuron O/P)

Assuming W_i's also to be 2-D vectors

Considering binary classification

$$\hat{y}_1 = \frac{e^{W_1 x}}{e^{W_1 x} + e^{W_2 x}}, \quad \hat{y}_2 = \frac{e^{W_2 x}}{e^{W_1 x} + e^{W_2 x}}$$

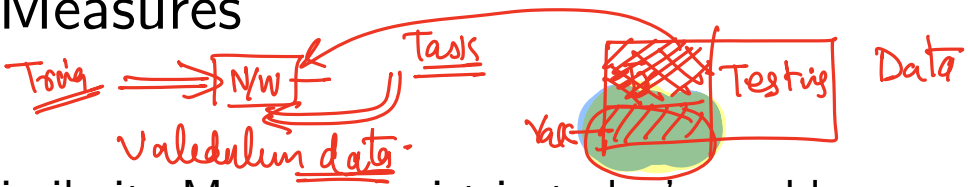
$W_1 \cdot \cos\theta_1 > W_2 \cdot \cos\theta_2$ Class 1 : $\hat{y}_1 > \hat{y}_2$

$W_2 \cdot \cos\theta_2 > W_1 \cdot \cos\theta_1$ Class 2 : $\hat{y}_2 > \hat{y}_1$

Classification boundary

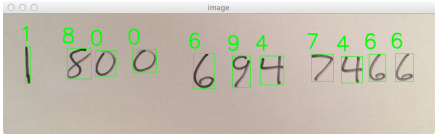
Anglewise (x) need to goto (W_1) but will goto (W_2) !!!

Need for Similarity Measures

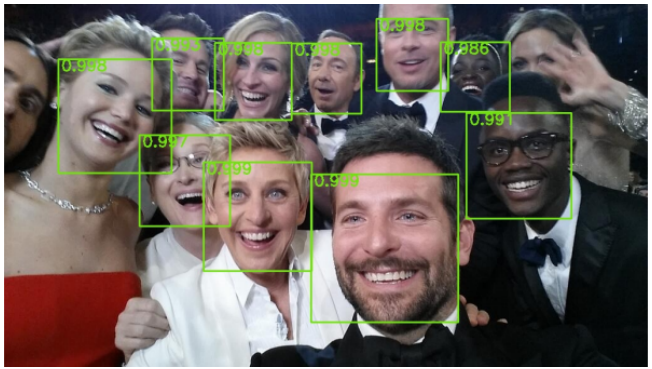


Several applications of Similarity Measures exist in today's world.

- Handwriting recognition.



- Recognition of faces in image.



Notion of a Metric

- A **Metric** is a function that quantifies a "distance" between every pair of elements in a set, thus inducing a measure of similarity.

- A metric $f(x, y)$ must satisfy the following properties for all x, y, z belonging to the set:
  - Non-negativity: $f(x, y)0$
  - Identity of Discernible: $f(x, y) = 0 <=> x = y$
  - Symmetry: $f(x, y) = f(y, x)$
  - Triangle Inequality: $f(x, z) <= f(x, y) + f(y, z)$

Traditional Approaches for Matching

The traditional approach for matching images, relies on the following pipeline:

1. **Extract Features:** For instance, color histograms of the input images.
2. **Learn Similarity:** Use $L_1 - norm$ on the features.

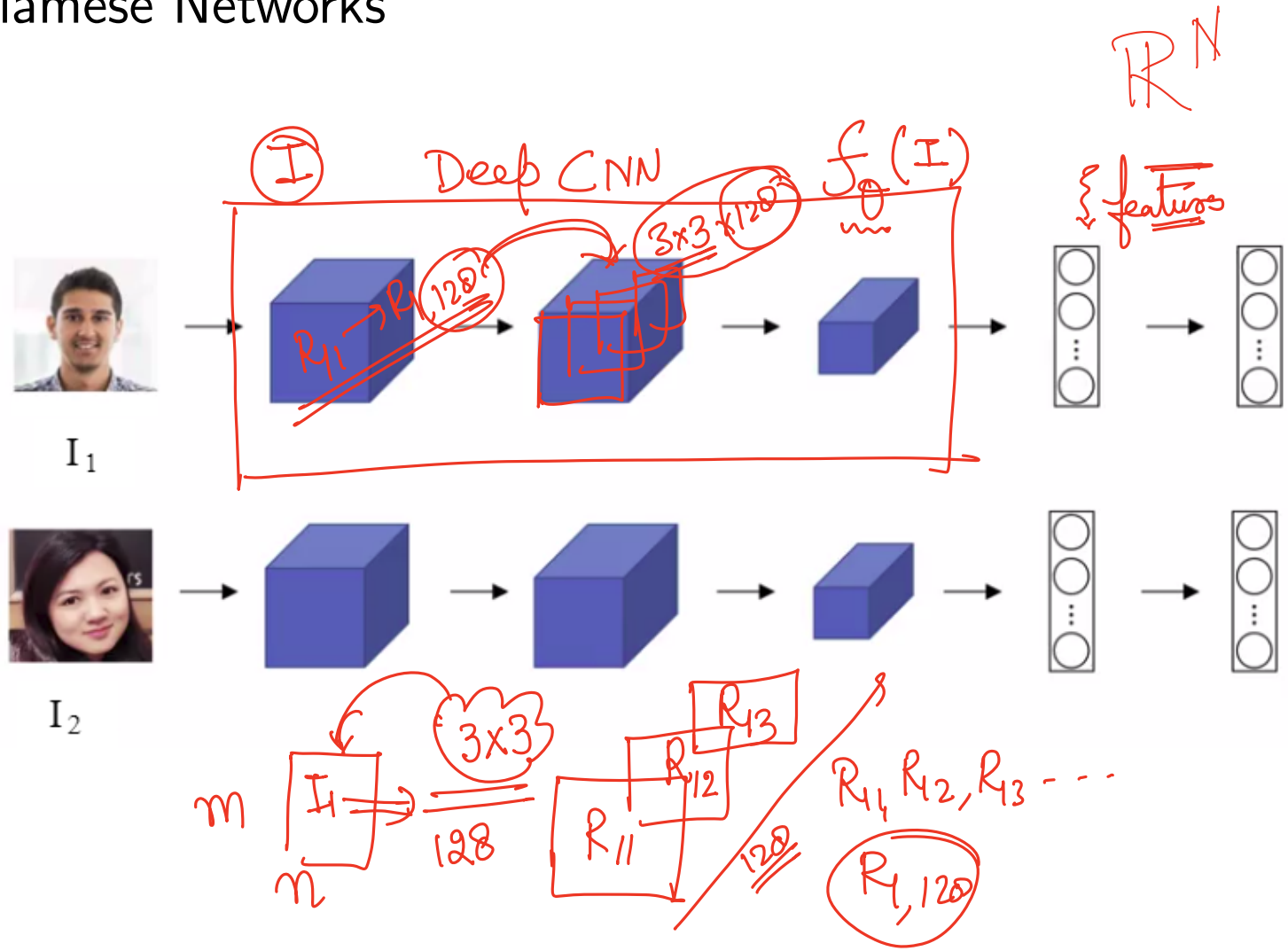
Challenges:
The principal shortcoming of traditional metric learning based methods is that the feature representation of the data and the metric are not **learned jointly**.

Problem Statement

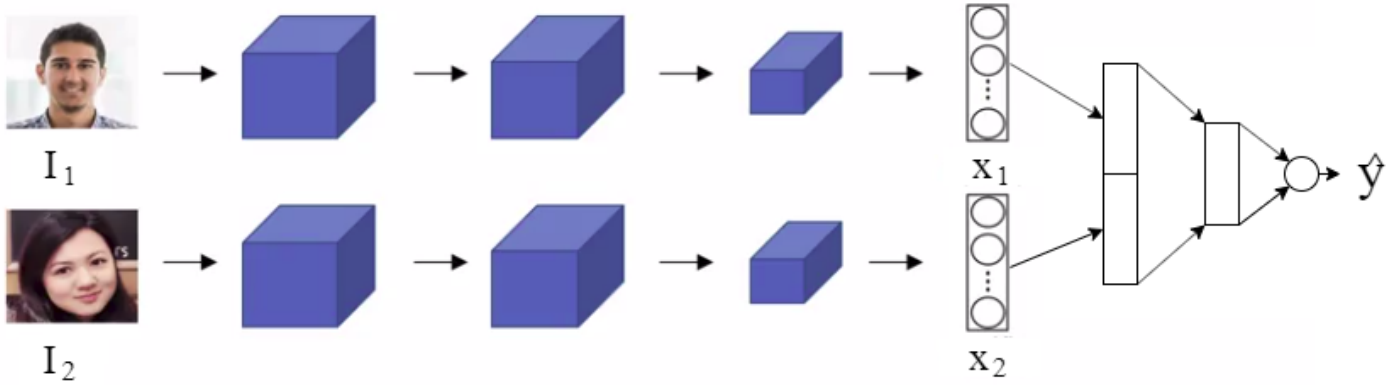
- **Input:** Given a pair of input images, we want to know how "similar" they are to each other.

- **Output:** The output can take a variety of forms:
  - Either a binary label, *i.e.* 0 (same) or 1 (different).
  - A Real number indicating how similar a pair of images are.

Siamese Networks



Siamese Networks as binary classification



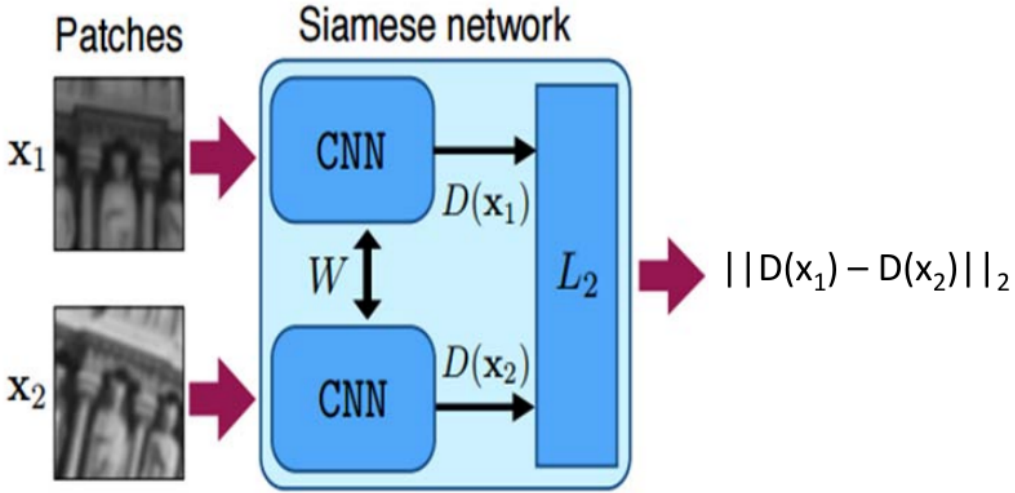
Problem Statement

- **Input:** Given a pair of input images, we want to know how "similar" they are to each other.

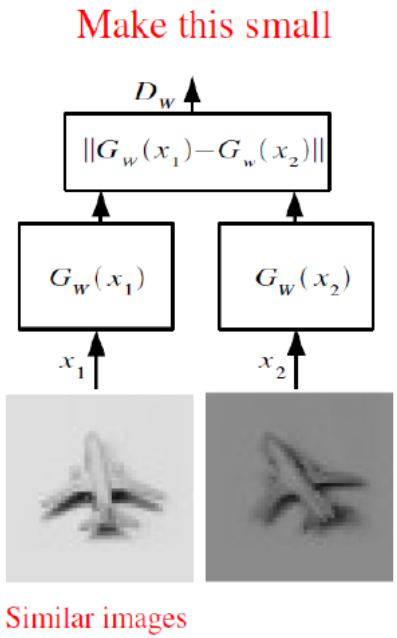
- **Output:** The output can take a variety of forms:
  - Either a binary label, *i.e.* 0 (same) or 1 (different).
  - A Real number indicating how similar a pair of images are.
- $d(img1, img2)$ = Degree of difference between images
  1. $d(img1, img2) < t$ For images belonging to same class
  2. $d(img1, img2) > t$ For images belonging to different class

Standard Siamese CNN

- **Input:** A pair of input images.
- **Output:** A label, 0 for **similar**, 1 **else**.

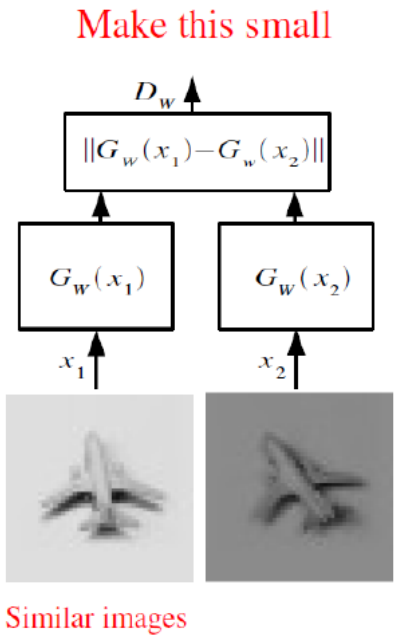


Siamese CNN - Loss Function



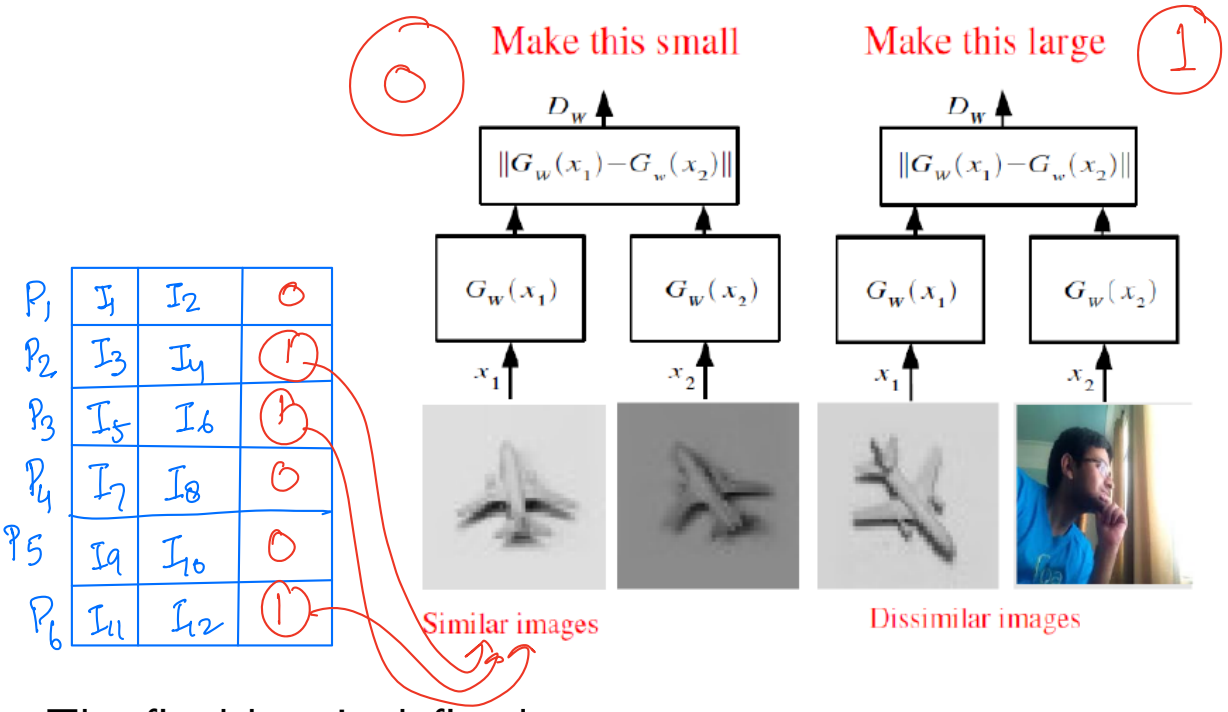
- Is there a problem with this formulation?

Siamese CNN - Loss Function



- Yes.
  - The model could learn to embed every input to a same point i.e. predict a constant as output.
  - In such a case, every pair of input would be categorized as positive pair.

Siamese CNN - Loss Function



The final loss is defined as:

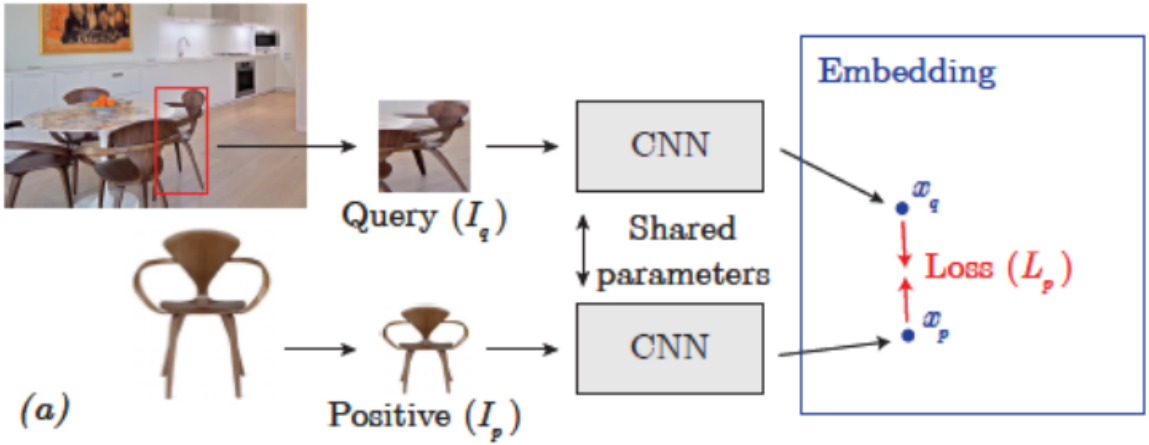
$$L = \Sigma loss\ of\ \ positive\ pairs + \Sigma loss\ of\ \ negative\ pairs$$

Siamese CNN - Loss Function

We can use different loss functions for the two types of inputs pairs.

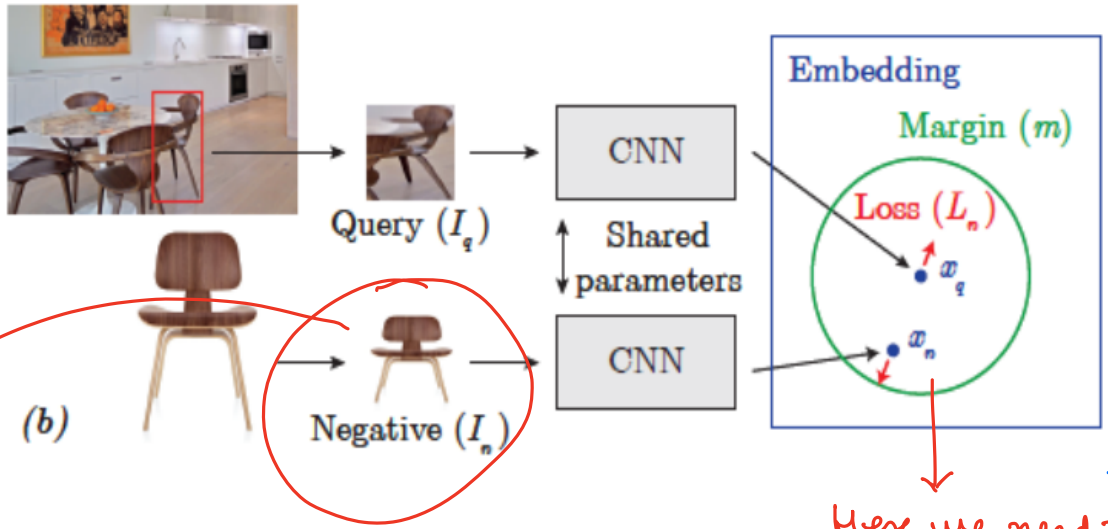
- Typical **positive pair** (x_p, x_q) loss: $L(x_p, x_q) = ||x_p - x_q||^2$

(Euclidean Loss)



Siamese CNN - Loss Function

- Typical **negative pair** (x_n, x_q) loss:
  $L(x_n, x_q) = max(0, m^2 - ||x_n - x_q||^2)$ **(Hinge Loss)**



Other than Chair

Loss is how close x_n & x_q are wrt Margin

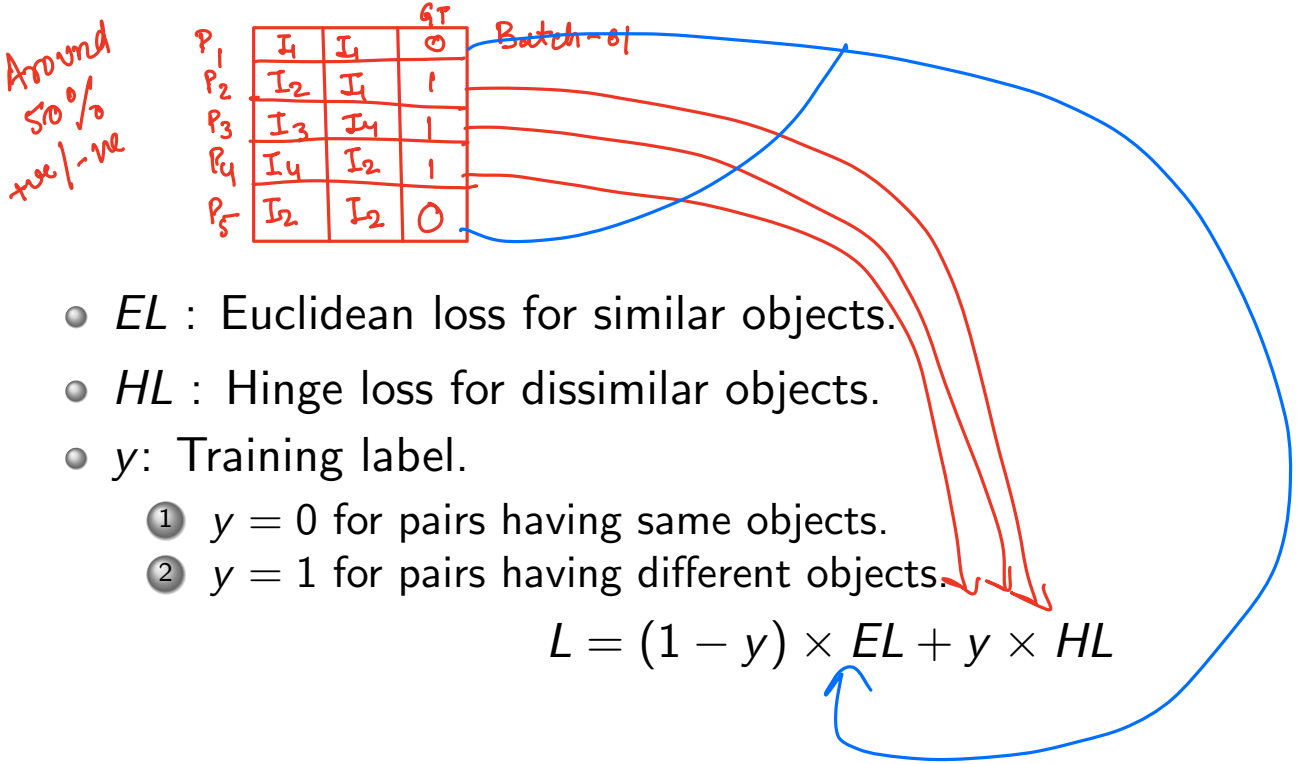
More than margin NO - loss

Less than margin Penalize.

if distance b/w x_n & x_q is more than (m) → No loss.

Here we need to make sure that negative need to be atleast as far as the margin (m).

Siamese CNN - Contrastive Loss Function

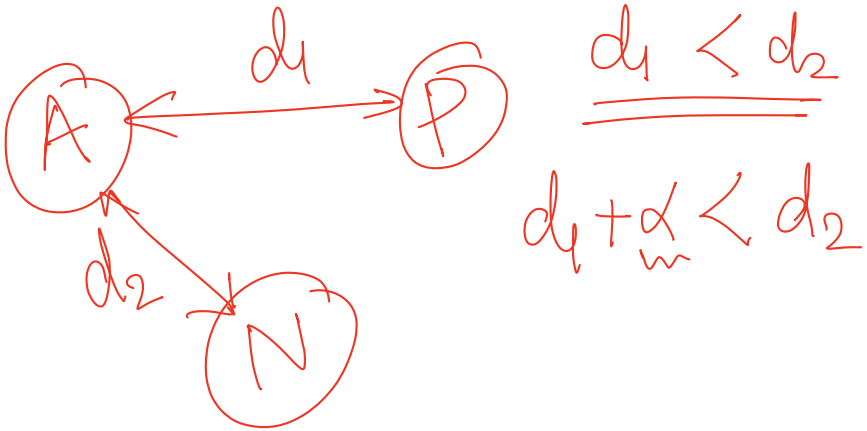


- EL : Euclidean loss for similar objects.
- HL : Hinge loss for dissimilar objects.
- y: Training label.
  1. $y = 0$ for pairs having same objects.
  2. $y = 1$ for pairs having different objects.

$$L = (1 - y) \times EL + y \times HL$$

Triplet Loss Function

- I_a : Anchor Image
- I_p : Positive Image. Belongs to same object as the anchor.
- I_n : Negative Image. Belongs to any other object.
- Let f be the network
- $f(I_a) = x_a$
- $f(I_p) = x_p$
- $f(I_n) = x_n$



Triplet Loss Function

$$D(A, P) = ||x_a - x_p||^2$$

$$D(A, N) = ||x_a - x_n||^2$$

$$D(A, P) \leq D(A, N)??$$

$$D(A, P) + \alpha \leq D(A, N)$$

$$D(A, P) - D(A, N) + \alpha \leq 0$$

Triplet Loss Function

$$L(A, P, N) = D(A, P) - D(A, N) + \alpha$$

$$L(A, P, N) = max(D(A, P) - D(A, N) + \alpha, 0)$$

$$J = \sum_{i=1}^{m} L(A^i, P^i, N^i)$$

Choosing triplets

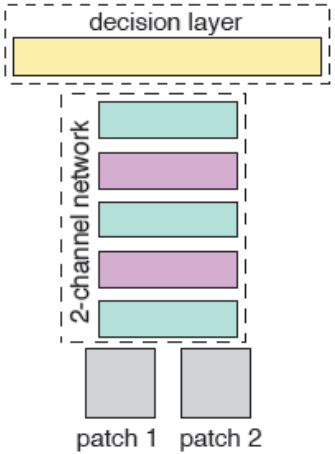
- The number of negative samples are very large as compared to positive samples.
- Hence choosing triplets is an important task.
- One way to choose triplets is for a given anchor, choosing all positive samples and randomly choose negative samples. Leads to easy triplets, causing training loss to be zero while lacking generalizability.
- Hence we use online semi hard negative mining for choosing the triplets.

Online Semi Hard Negative Mining

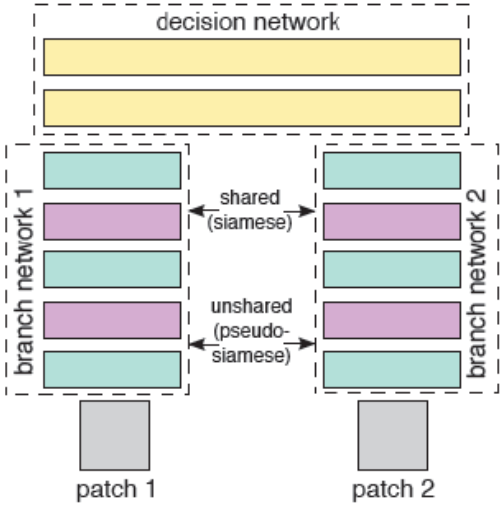
- For a given network, we choose 1000 triplets randomely.
- We pass these 1000 triplets through the network and find $D(A, P)$ and $D(A, N)$ for each triplet.
- We select the triplets which satisfies the following conditions:
  1. $D(A, N) - D(A, P) < \alpha$ These are the triplets which are hard and will produce a loss for the optimization of the network.
  2. $D(A, N) > D(A, P)$ Discards the outliers.
- Out of the 1000 random triplets, we only train the network with those which satisfy above two conditions and then repeat this prosess.

Architecture Varieties

- Design largely governed by what performs well empirically on the task at hand.

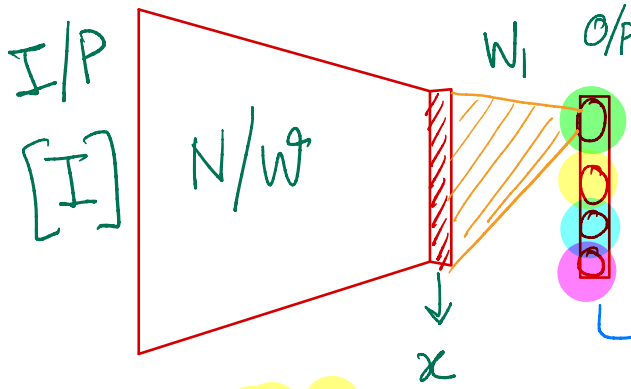


Inputs are merged right at the onset



Inputs are first embedded independently, then merged.

(*) Just talk about **Triplet loss** and its variants
$\Rightarrow$ Moving towards **metric learning**. [old slides]



Any (W_i) is the weight vector of that $(i)^{th}$ neuron for that $(i)^{th}$ class.

All weights together W $(N \times 1)$ $(4 \times N)$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

Preactivated O/P of our N/W (N-dim) $[N \times 1]$
$x : [x_1\ x_2 \cdots x_N] \in \mathbb{R}^N$

y' (Raw value) is the O/P of the N/W we get (4×1) (W/O Softmax)

$\Downarrow$ Apply softmax

$\hat{y} \Rightarrow$ after Softmax

$$y' = [y_1'\ y_2'\ y_3'\ y_4']$$

$$y_i' = W_i x$$

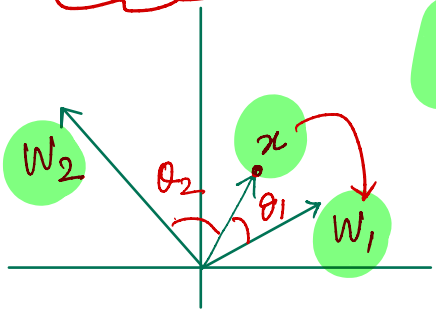
$y \Rightarrow$ 1-Hot encoded ground truth (4×1)

$$\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4]$$

$$\hat{y}_i = \frac{e^{W_i \cdot x}}{\sum_{j=1}^{C} e^{W_i \cdot x}}$$

(i^{th} neuron O/P)

Assuming W_i's also to be 2-D vectors



Considering binary classification

$$\hat{y}_1 = \frac{e^{W_1 x}}{e^{W_1 x} + e^{W_2 x}}, \quad \hat{y}_2 = \frac{e^{W_2 x}}{e^{W_1 x} + e^{W_2 x}}$$

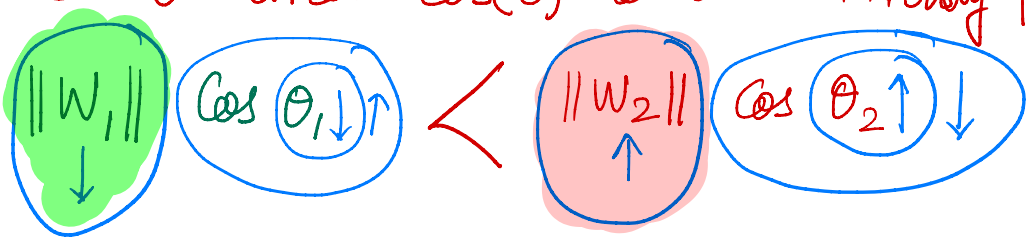
$W_1 \cdot \cos\theta_1 > W_2 \cdot \cos\theta_2$ Class 1 : $\hat{y}_1 > \hat{y}_2$

$W_2 \cdot \cos\theta_2 > W_1 \cdot \cos\theta_1$ Class 2 : $\hat{y}_2 > \hat{y}_1$

Classification boundary

Anglewise (x) need to goto (W_1) but will goto (W_2) !!!

* θ and $\cos(\theta)$ both are inversly proportional

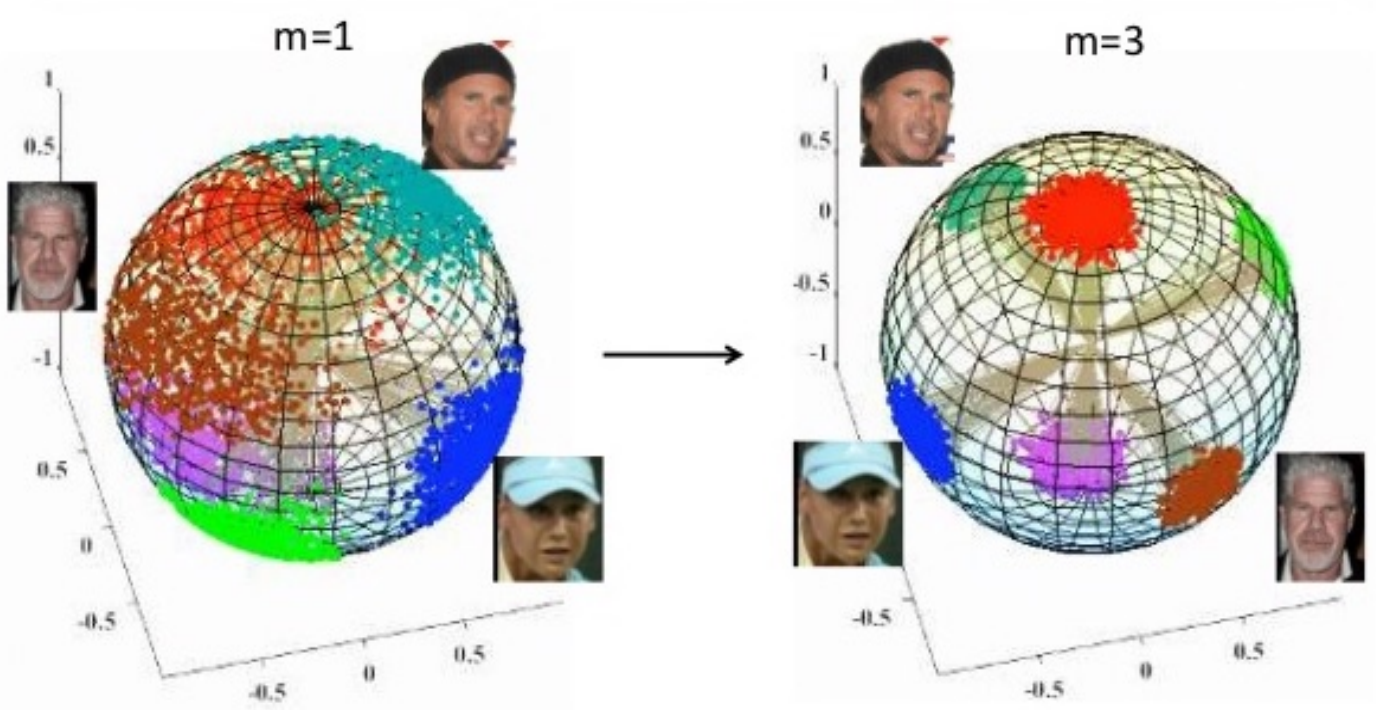


for Softmax loss function the decision boundary depends upon → (a) $\|W_i\|$

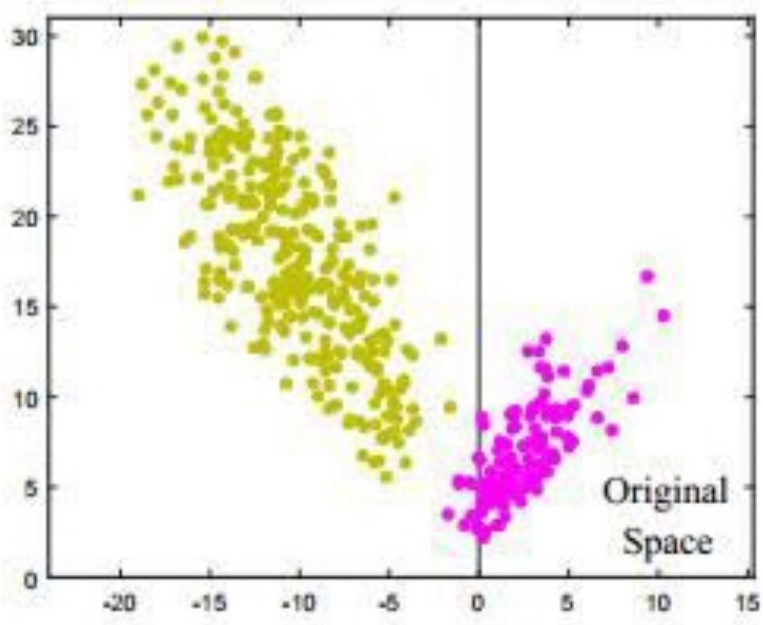
(b) $\cos(\theta_i)$ that is the angle between $[W_i]$ and $[x]$ vector.

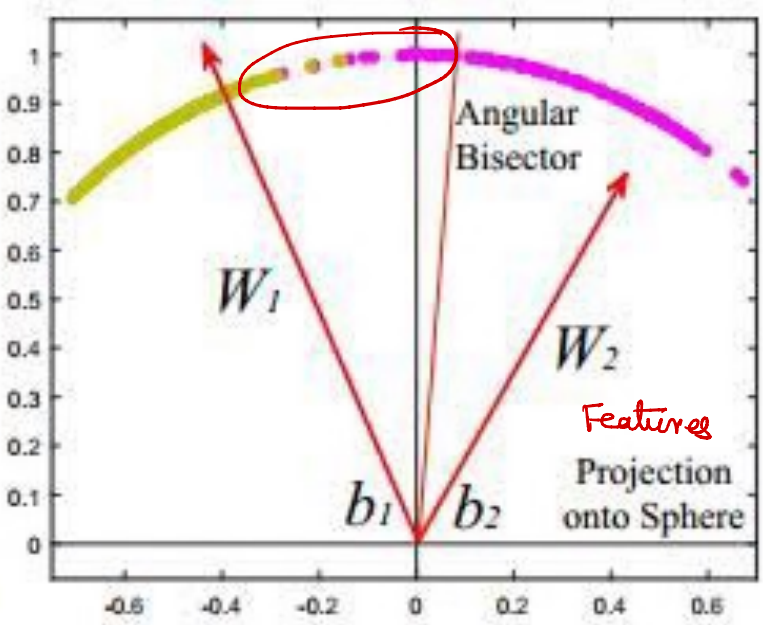
Since $(\cos\theta)$ is in b/w $(-1, 1)$ and weights are unconstraint boundaries are mostly driven by $\|W_i\| \Rightarrow$ (Bahubali)

Sphereface : Deep Hypersphere Embedding for face Recognition.



(*) for classification decision boundary is learned separating the classes. This can be a serious issue when output lie near the decision boundary.





So Normal Softmax will give us infinite amount of space within some angular range (So as one can fit huge data in between)

⇓

Good to capture intra class variation

But

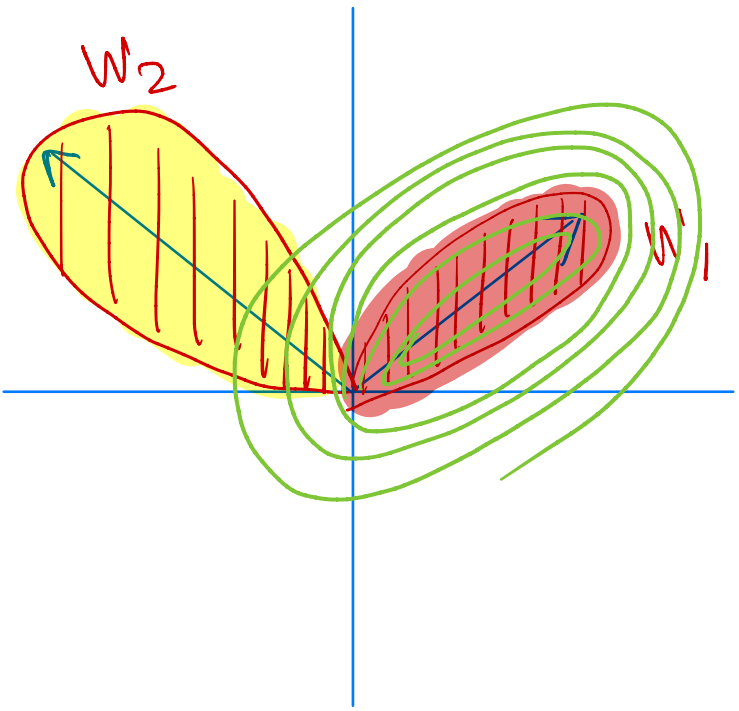
All space got exhausted as features are spread all over

⇐

Besides, Uneven and biased space partitioning

W_i's effect

No class compactness and inefficient feature space utilization



(*) Points "close" to (W_1) will goto class-01.

But how "close"

⇓

Need to decided by $||W_1||$ & $||W_2||$

This is the real nature of Softmax.

Classes are getting (within angular region) W/o any apparent clustering & the N/w is free to project features anywhere within that space.

→ This may not be good for generalization and fails in Open-set scenarios.

Modified Softmax : M-Softmax to L-Softmax

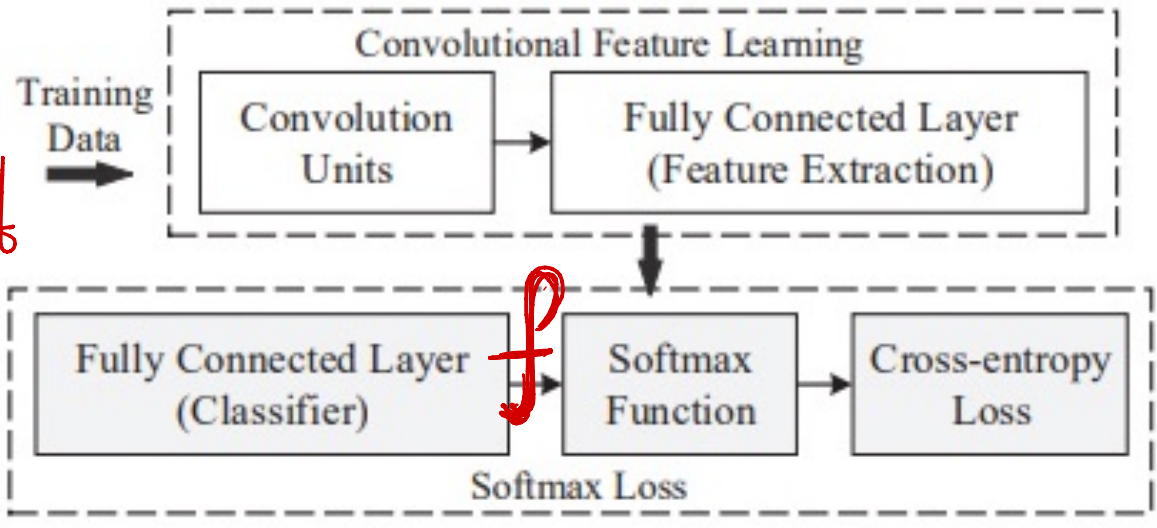
Large Margin Softmax is the first paper that talks about margin similar to Triplet loss.

Helps to achieve

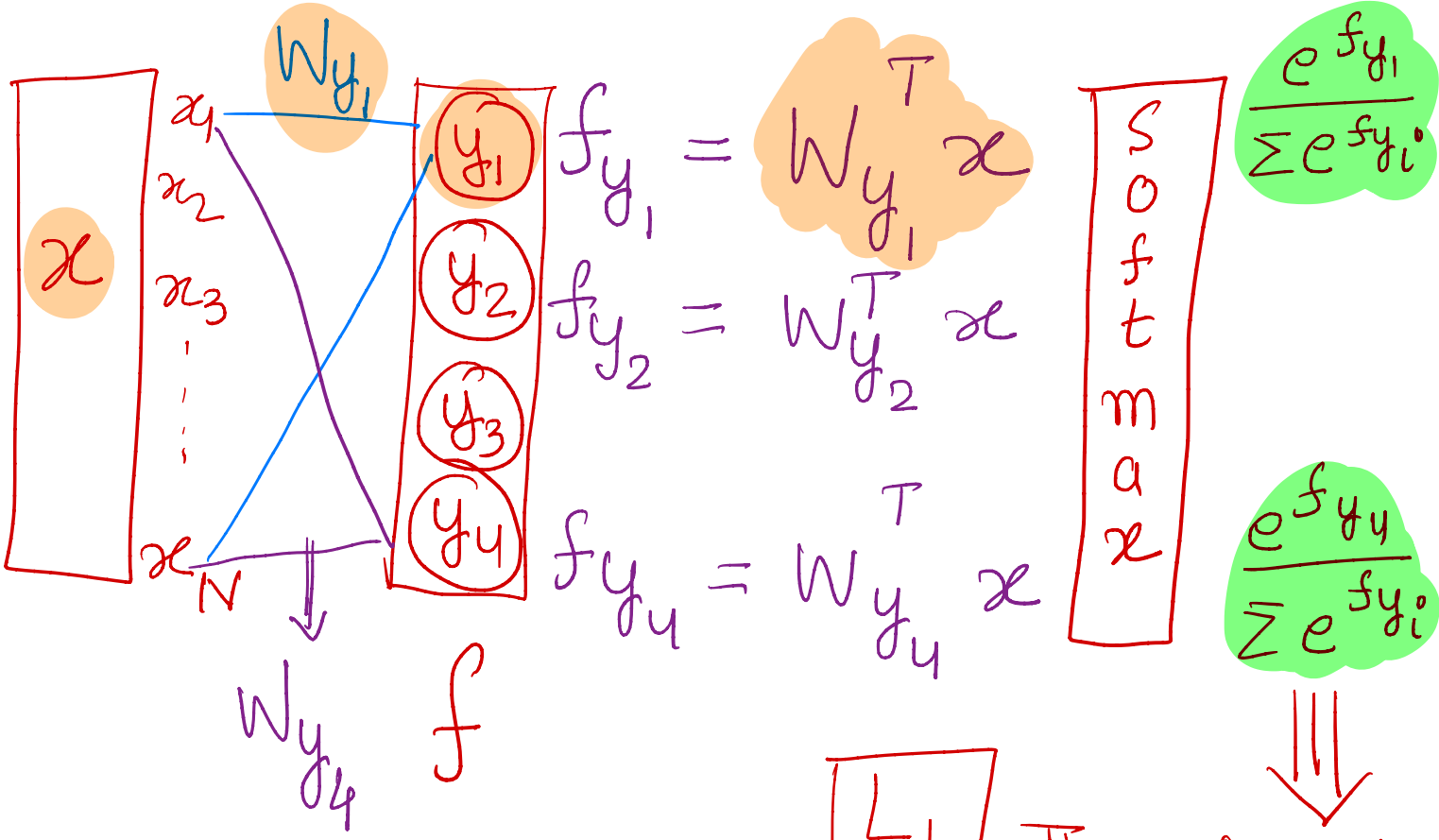
(a) Intra-class Compactness

(b) Inter-class Seperability

Both are essentially required for visual classification or Verification.



$f \Rightarrow$ O/P of Last fully Connected Layer.



$$L = \sum_i -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix}$$

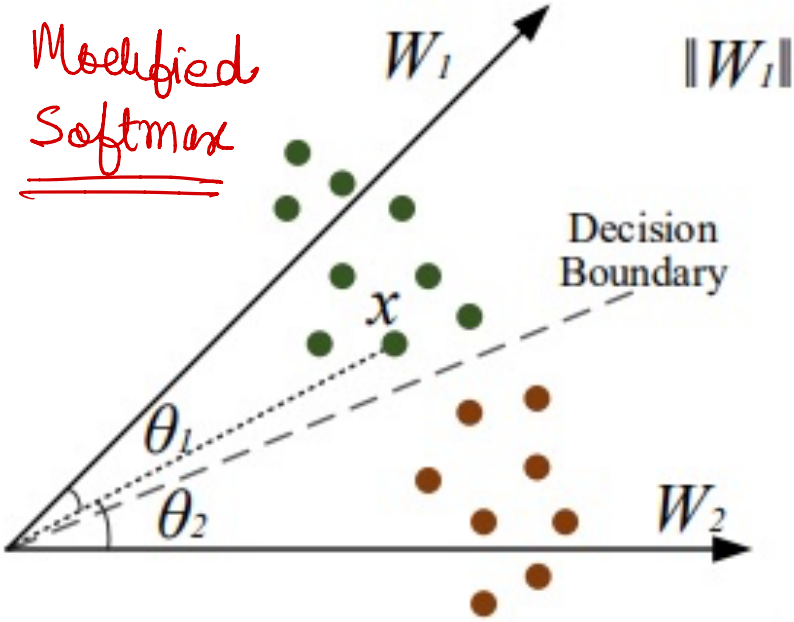
They will goto Cross entropy

$$L = L_1 + L_2 + L_3 + L_4$$

$$L_i = -\log \left(\frac{e^{\|W_{y_i}\| \|x_i\| \cos(\theta_{y_i})}}{\sum_j e^{\|W_j\| \|x_i\| \cos(\theta_j)}} \right)$$

Normal Softmax defined in terms of last layer O/P.

Modified Softmax



Just Considering Binary classification

$(W_1 * x > W_2 * x)$

for classification of (x) to class - 01.

$\|W_1\| \|x\| \cos(\theta_1) >$

$\|W_2\| \|x\| \cos(\theta_2)$

Projecting weights onto unit Sphere $\Rightarrow$ $\frac{W_i}{\|W_i\|}$

Eliminating the effect of $\|W_i\|$.

(M- Softmax)

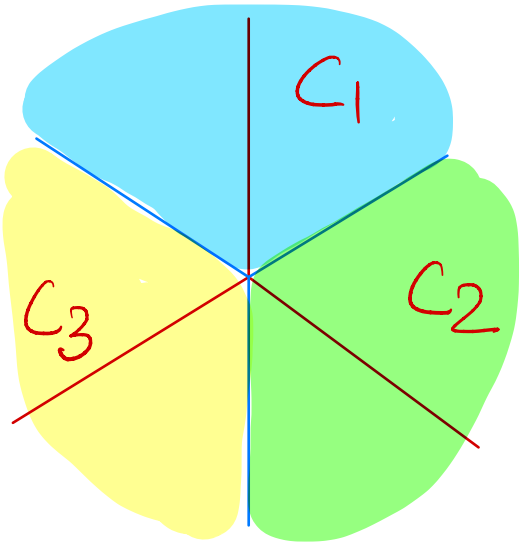
Classification Boundary becomes

$\Rightarrow \|x\| [\cos(\theta_1) - \cos(\theta_2)] > 0$

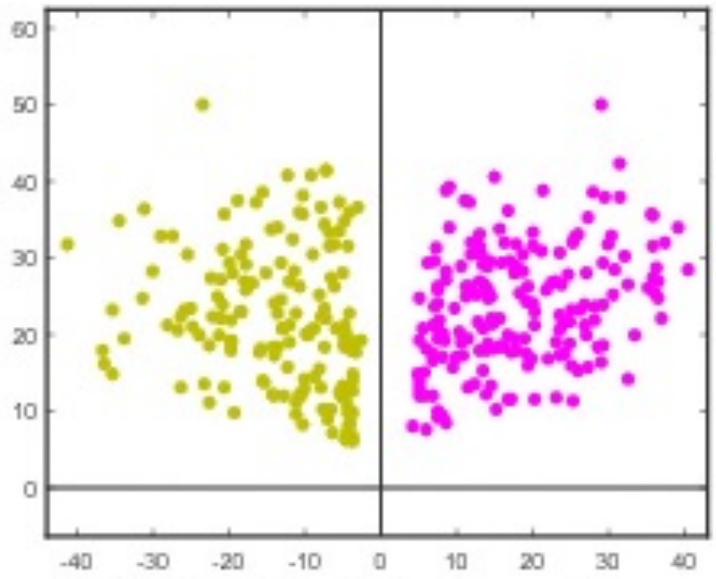
This will lead to angle bisector.

We will have ⓐ Original feature space.

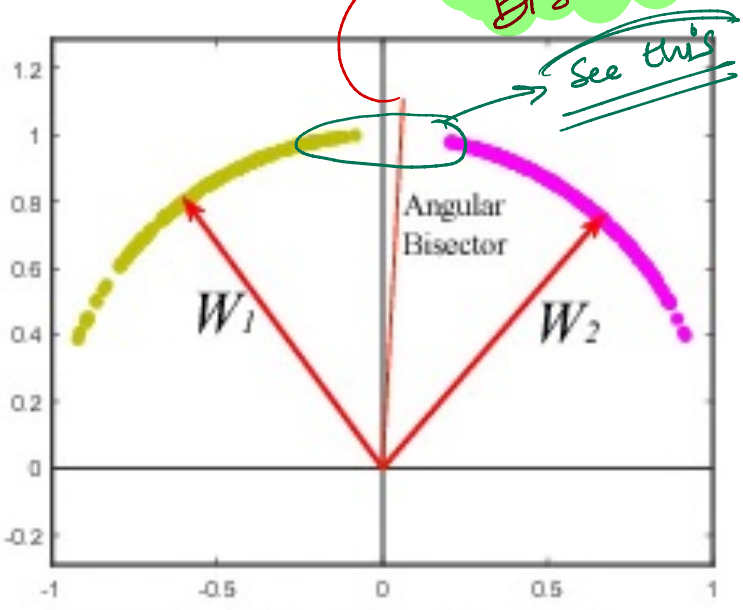
ⓑ Projected feature space.



This boundary becomes Bisector



(c) Modified Softmax Loss



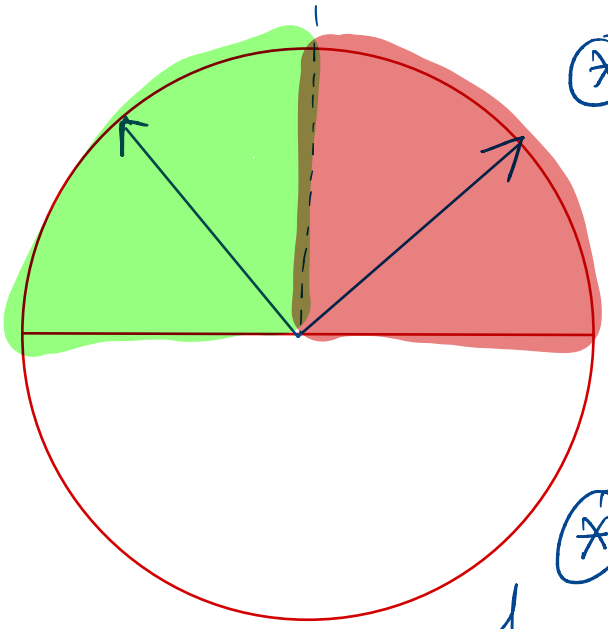
(d) Modified Softmax Loss

2-D (weights), 2D last layer features.

all things are normalized (all on Circle)

⊛ This ensures unbiased space partitioning

⊛ Still not to much Compact but better than basic Softmax → Large margin Softmax



* Points are over unit Circle and angular Boundary Can be properly Seen.

* The angle is ranging from $(0$ to $\pi)$

* $\cos(\theta)$ is monotonically decreasing b/w $(0 - \pi)$

i.e $[\theta \uparrow]$ then $[\cos\theta \downarrow]$

Class belongingness got decreased.

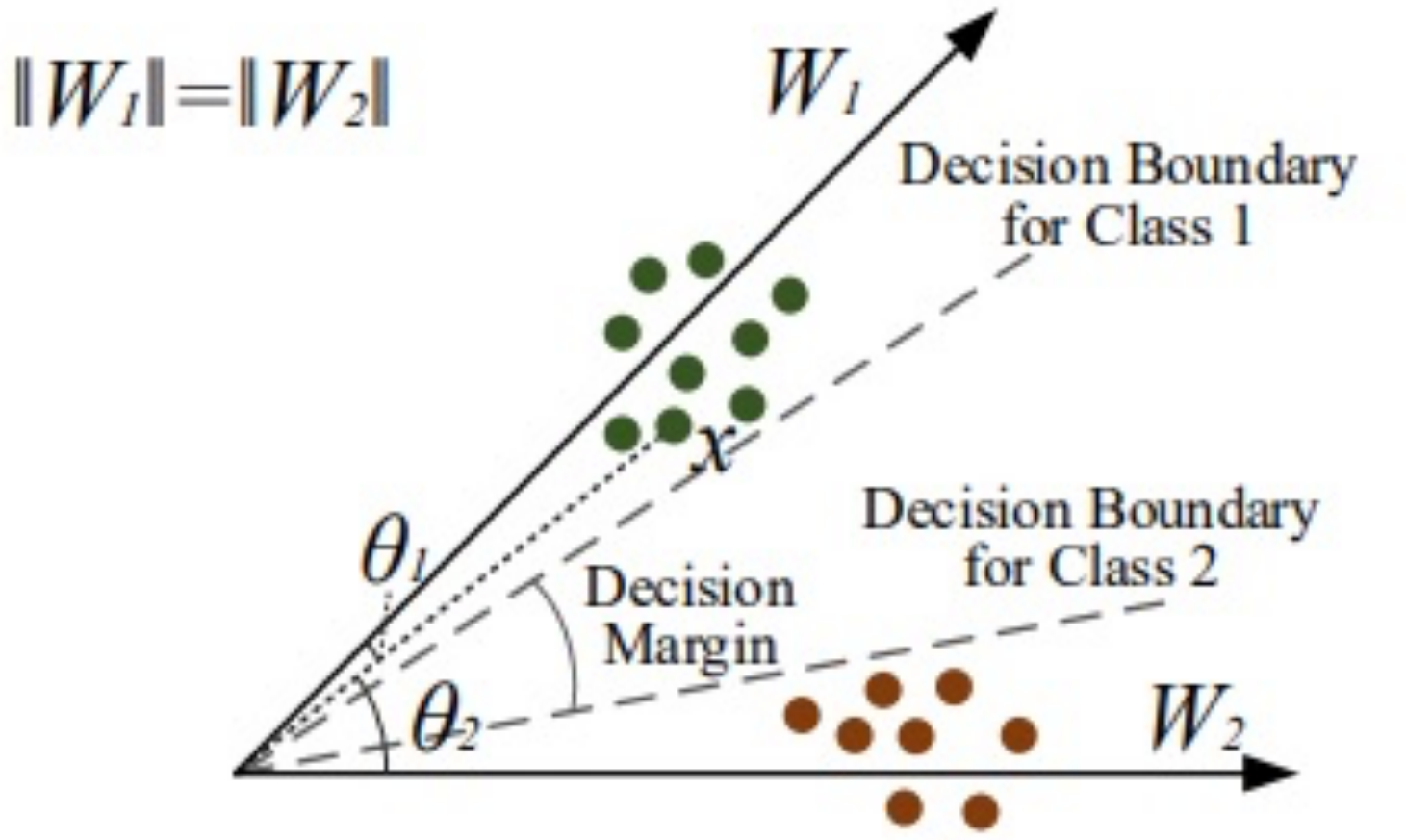
Remember that $\cos(\theta)$ is

(a) monotonically decreasing $(0 \to \pi)$

(b) monotonically increasing $(-\pi \to 0)$

In order to have better class Compactness $\Longrightarrow$ Angular Margin

Angular boundaries to Angular Margins



(*) Classification need to be Stricter with wider decision margin for better Generalization.

L-Softmax (large margin)

$\theta \longrightarrow m\theta$ angle rang $(0 - \pi)$

$(m \Rightarrow$ +ve integer) going to $(0 - \frac{\pi}{m})$

Now the decision boundary for class-01 requires :

$$\|W_1\|\|x\| \cos(m\theta_1) > \|W_2\|\|x\| \cos(\theta_2)$$

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})}}{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\psi(\theta_{y_i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right)$$

$\cos(m(\theta_i))$

$$\psi(\theta) = \begin{cases} \cos(m\theta), & 0 \le \theta \le \frac{\pi}{m} \\ \mathcal{D}(\theta), & \frac{\pi}{m} < \theta \le \pi \end{cases}$$

$m \Rightarrow$ makes the classification margin larger.

$D(\theta)$ is monotonically decreasing fn with $D(\pi/m) = \cos(\pi/m)$

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$

So finally $\psi(\theta)$ is set as defined above where $k \in [0, m-1]$ (k) is an integer.

$k=0$	$k=1$	$k=2$	.....	$k=m-1$
$[0, \pi/m]$	$[\pi/m, 2\pi/m]$	$[2\pi/m, 3\pi/m]$		$[\frac{(m-1)\pi}{m}, \frac{m\pi}{m}]$
$\psi(\theta) =$				
$\cos(m\theta)$	$-\cos(m\theta) - 2$	$\cos(m\theta) - 4$		$(-1)^{m-1}\cos(m\theta) - 2(m-1)$
Range				
$(1, -1)$	$(-1, -3)$	$(-3, -5)$		

Hence $\psi(\theta)$ is a monotonically decreasing continous fuction depending upon $\cos(m\theta)$

Loss Function	Decision Boundary
Softmax Loss	$(\boldsymbol{W}_1-\boldsymbol{W}_2)\boldsymbol{x}+b_1-b_2=0$
Modified Softmax Loss	$\|\boldsymbol{x}\|(\cos\theta_1-\cos\theta_2)=0$
A-Softmax Loss	$\|\boldsymbol{x}\|(\cos m\theta_1-\cos\theta_2)=0$ for class 1 $\|\boldsymbol{x}\|(\cos\theta_1-\cos m\theta_2)=0$ for class 2

Table 1: Comparison of decision boundaries in binary case. Note that, θ_i is the angle between $\boldsymbol{W}_i$ and $\boldsymbol{x}$.

for $m \geqslant 2$, we will ② decision boundaries for each class.

$\cos(m\theta_1) - \cos(\theta_2) \geqslant 0$ (C1)

$\cos(\theta_1) - \cos(m\theta_2) \geqslant 0$ (C2)

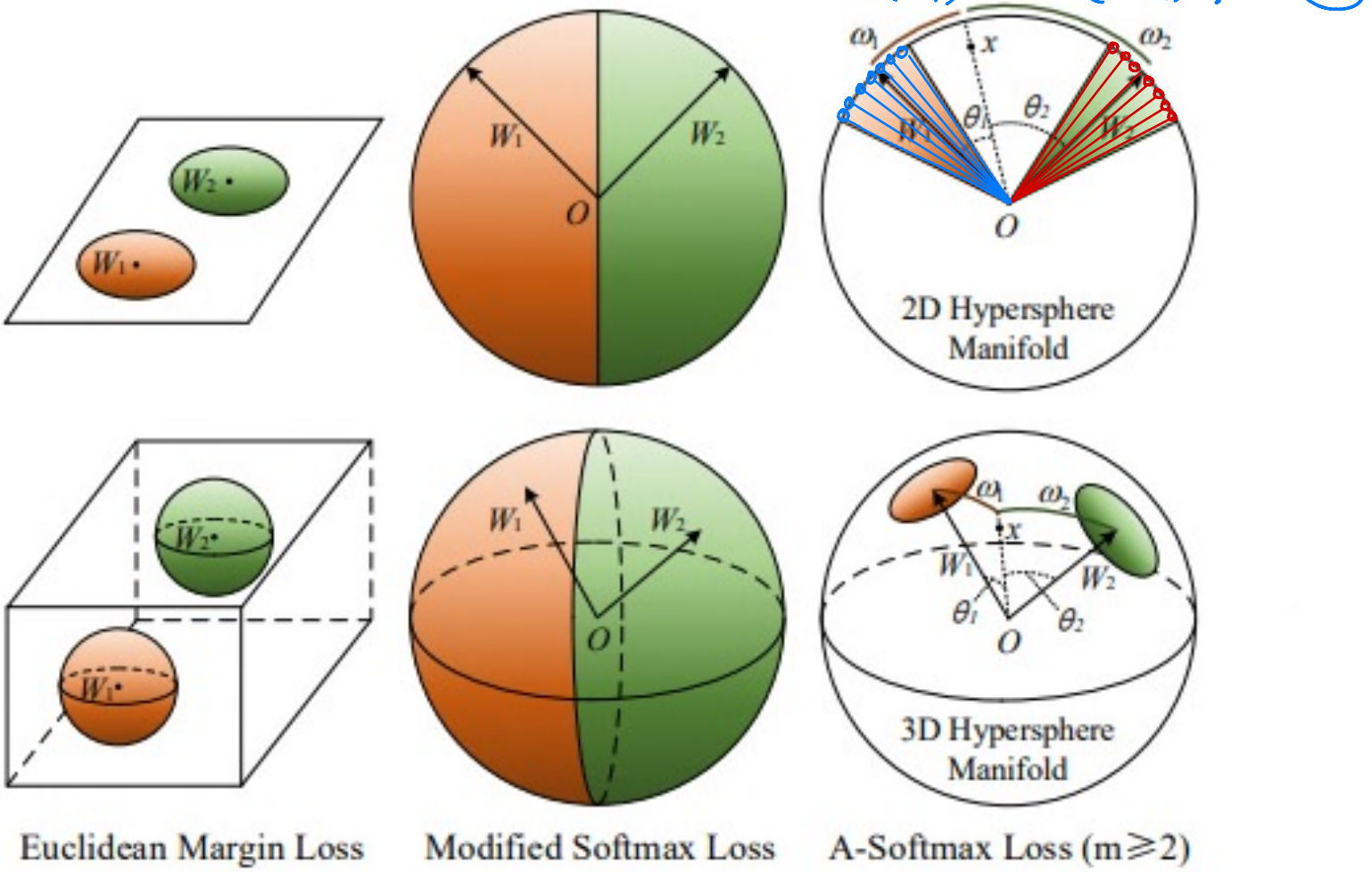


Figure 3: Geometry Interpretation of Euclidean margin loss (e.g. contrastive loss, triplet loss, center loss, etc.), modified softmax loss and A-Softmax loss. The first row is 2D feature constraint, and the second row is 3D feature constraint. The orange region indicates the discriminative constraint for class 1, while the green region is for class 2.

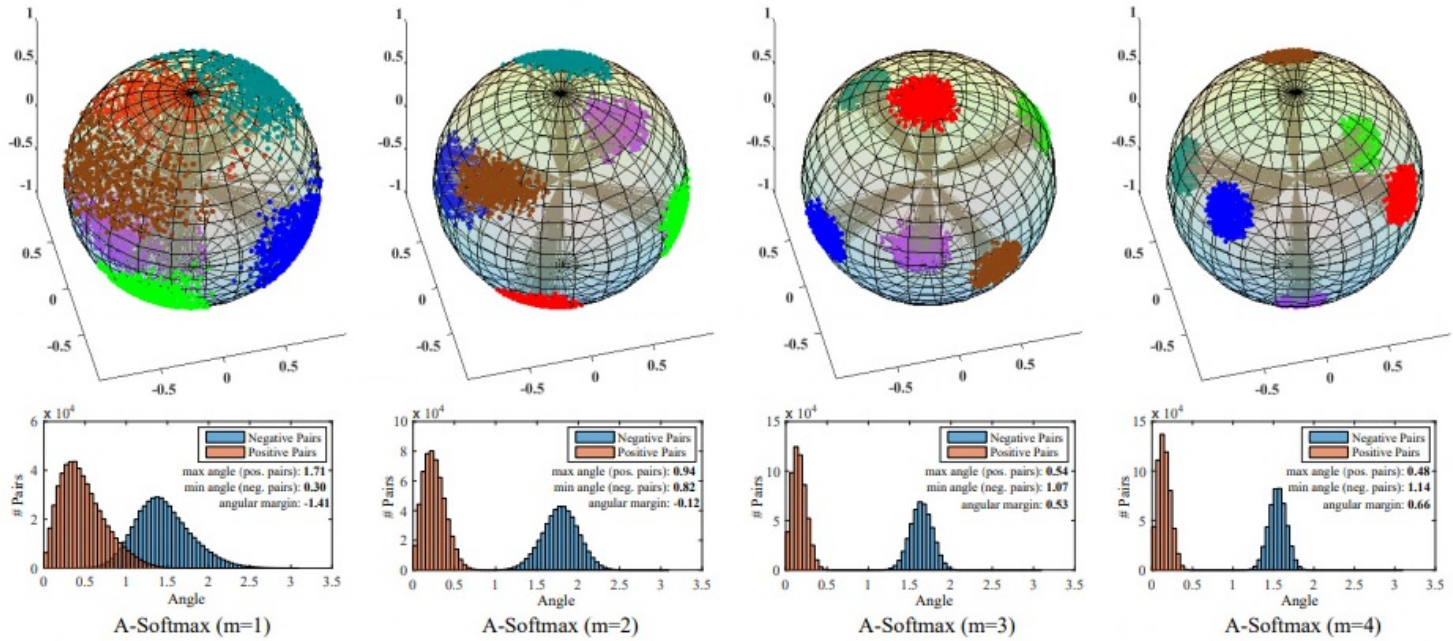


Figure 5: Visualization of features learned with different m. The first row shows the 3D features projected on the unit sphere. The projected points are the intersection points of the feature vectors and the unit sphere. The second row shows the angle distribution of both positive pairs and negative pairs (we choose class 1 and class 2 from the subset to construct positive and negative pairs). Orange area indicates positive pairs while blue indicates negative pairs. All angles are represented in radian. Note that, this visualization experiment uses a 6-class subset of the CASIA-WebFace dataset.

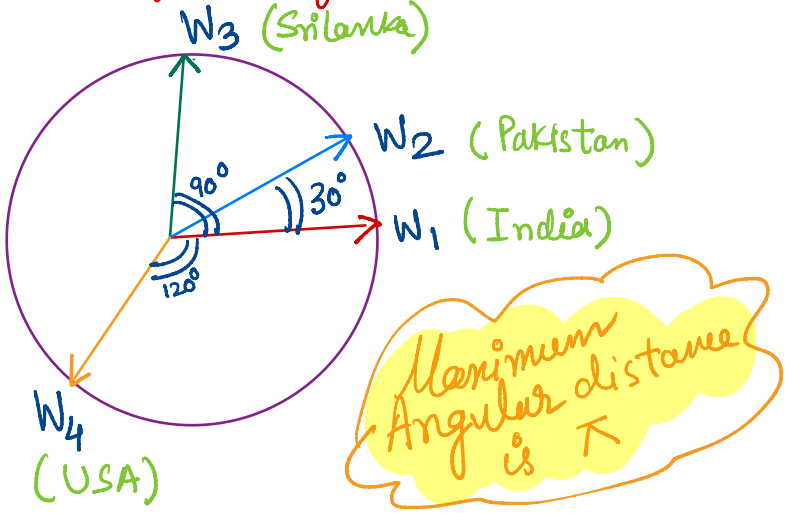
(*) In Conclusion

- Normalize Classification Weights
- Introduce the angular margin Controlled by (m)
- Zero the bias

⟹ Lead to learn discriminative and generalized features for Open Set scenarios

More understanding for A-Softmax.

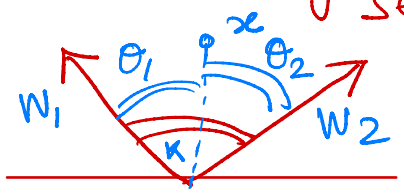
(*) Angle of (θ_1) with the actual ground truth class (say $\vec{W}_1$) it will actually be considered as $(m\theta_1)$ and its class likelihood further got decreased from $\cos(\theta_1) \longrightarrow \cos(m\theta_1)$



In A-Softmax Case for decision boundary : $m\theta_1 = \theta_2$

& (Binary) $\theta_1 + \theta_2 = K$

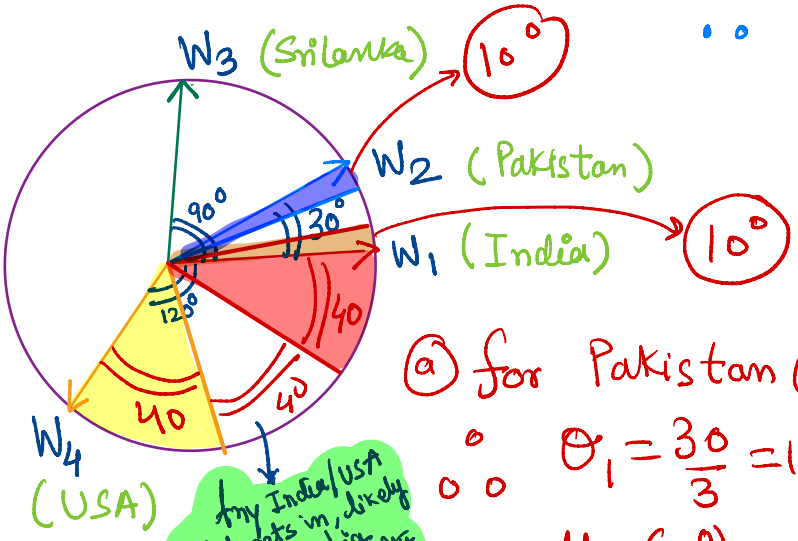
$K \Rightarrow$ Angular class seperation



$\therefore \quad \theta_1 + m\theta_1 = K$

$\Rightarrow \boxed{\theta_1 = K/m+1}$

This means that for $m=2$ $(\theta_1 = K/3)$ [Angular seperation got divided into 3 parts]



(a) for Pakistan $(K=30)$

$\therefore \quad \theta_1 = \frac{30}{3} = 10°$ (for decision boundary)

(b) Remember you are allowed to goto other side (USA), and that too upto 40° as for USA $(K=120)$

$\therefore \theta_1 = \frac{120}{3} = 40°$ (for dB)

$30 \longrightarrow 90 \Rightarrow \cos(90) = 0$

$\hookrightarrow (60) \Rightarrow \cos(60)$

$\rightarrow$ after(10°) you are not allowed to go close to Pakistan.

$\rightarrow$ If some point with GT = India have

$\theta_1 = 11 \Rightarrow (\theta_2 = 19) \Longrightarrow \cos(19)$ ✓

$\Downarrow$ Considered as (22) $\Longrightarrow \cos(22)$

$15 \;\&\; 15 \Longrightarrow \cos(15)$ ✓

$\hookrightarrow (30) \Longrightarrow \cos(30)$

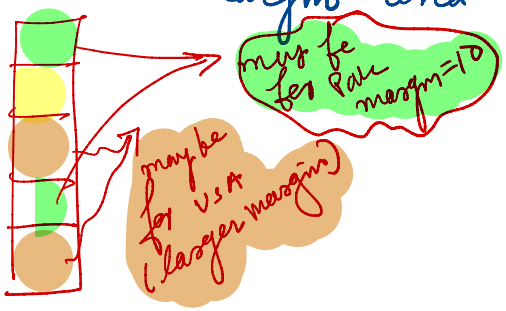
Parameter (m) ensures that margin between classes got maintained.

Problems with A-Softmax (Sphereface loss)

↳ Due to integer value of (m), the curve of target logit for GT class is very steep.
(This will hinder convergence)
(Softmax dominates)

⟹ Different margins for different classes.

So in decision space some inter class features may have larger margin and others will have smaller margin. → Reduces its discrimination



CosFace :

(*) Decision margin is defined in Cosine space unlike Sphereface & Softmax where it is defined in angular (θ)-space.

for margin parameter (m)
assuming angular 1 class margin b/w
(2) classes = (K) then

$$\text{Margin} = \left(\frac{m-1}{m+1}\right)K$$

(*) Large Margin Cosine loss [LMCL]

$$L_{lmc} = \frac{1}{N}\sum_i -\log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s\cos(\theta_{j,i})}}$$

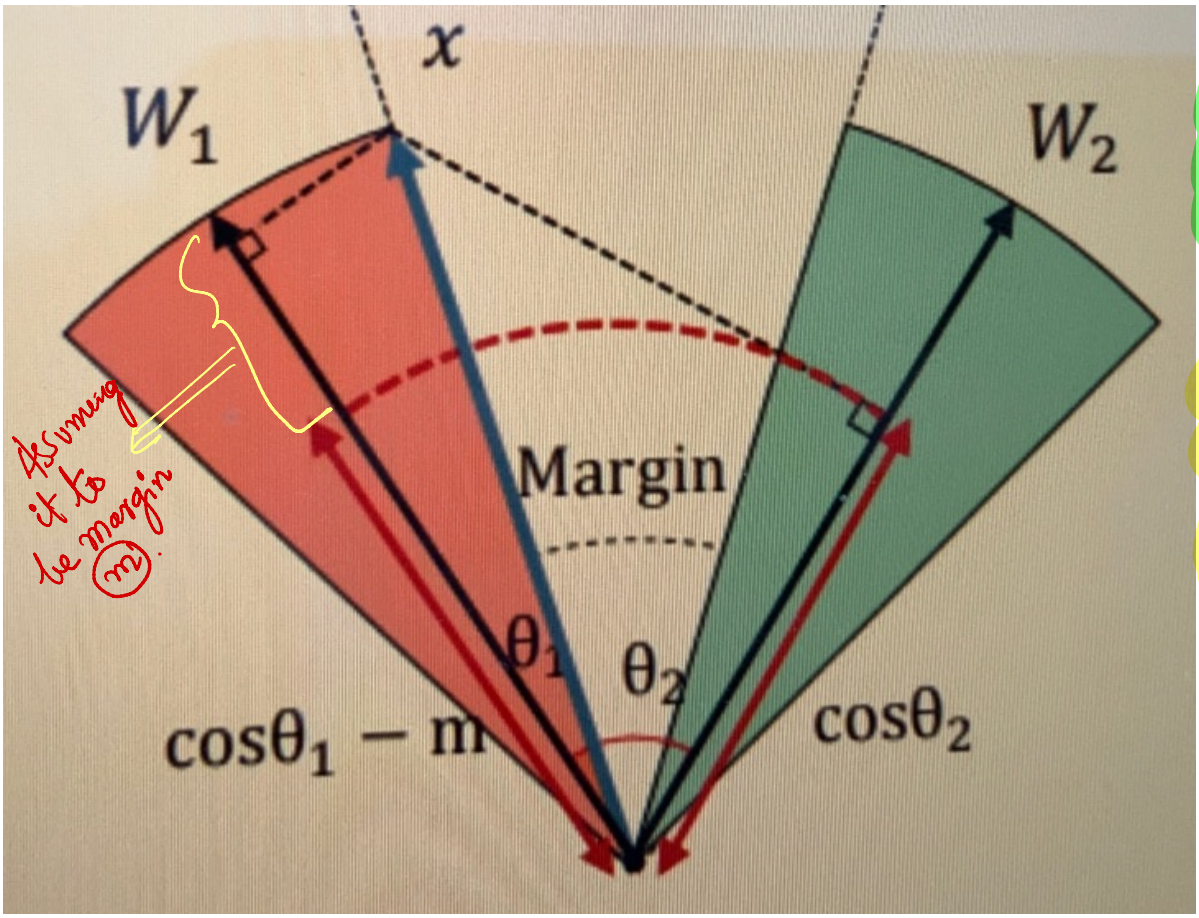
[$m \geq 0$, magnitude of Cosine margin]

$\|x\| = s$

(*) $W = W^*/\|W^*\|$

$x = x^*/\|x^*\|$

$\cos(\theta_{j,i}) = W_j^T x_i$



Assuming it to be margin (m)

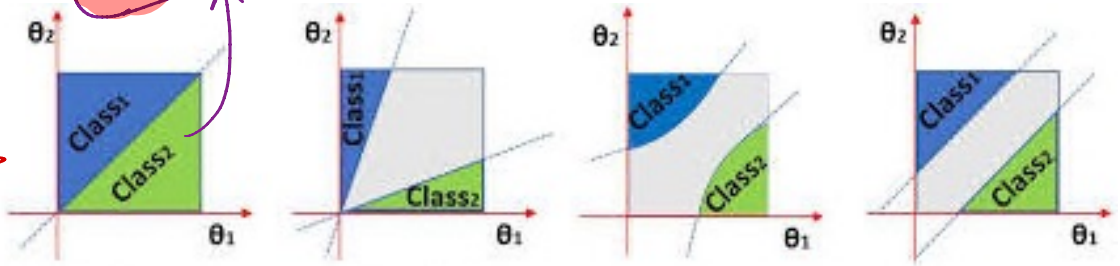
C_1 :
$\cos(\theta_1) \geq \cos(\theta_2) + m$

C_2 :
$\cos(\theta_2) \geq \cos(\theta_1) + m$

$\therefore$ for class 1 : $\cos(\theta_1)$ is maximized ($\theta_1 \downarrow$) while $\cos(\theta_2)$ is minimized ($\theta_2 \uparrow$)

All figures in (θ) space

Why class ② is close to θ_1



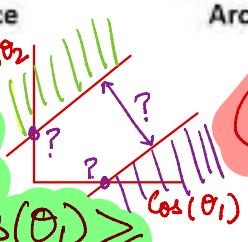
Softmax | SphereFace | CosFace | ArcFace

(a) $\|W_1\|\|x\| \cos\theta_1 \geq \|W_2\|\|x\| \cos\theta_2$

(b) $\cos\theta_1 \geq \cos\theta_2$ Angular bisector

(c) $\cos(m\theta_1) \geq \cos(\theta_2)$

(d) $\cos(\theta_1) \geq \cos(\theta_2) + m$



[It only have non-linear angular margin]

(e) $\cos(\theta_1 + m) \geq \cos(\theta_2)$