

Robot Learning Course

Metric Learning

▼ TABLE OF CONTENTS

- 1 [Metric Learning](#)
 - a [Introduction](#)
 - b [Embedding Space](#)
 - c [Distance Functions](#)
 - a [Euclidean Distance](#)
 - b [Cosine Distance](#)
 - d [Triplet Loss](#)
 - a [Triplet Mining](#)
 - e [Multi-task Learning with Metric Learning](#)
 - a [Combined Loss Function](#)
 - b [Model Architecture](#)
 - f [Evaluation Metrics](#)
 - a [TPR and FPR](#)
 - b [ROC Curve and AUC](#)
 - c [TPR@FPR](#)
 - g [Training Considerations](#)
 - a [Normalization](#)
 - b [Margin Selection](#)
 - c [Batch Composition](#)
 - h [Expected Knowledge](#)
 - a [1. Distance Functions and Embedding Space](#)
 - b [2. Triplet Loss Understanding](#)
 - c [3. Multi-task Learning](#)
 - d [4. Evaluation Metrics](#)

Introduction

Metric learning is a machine learning approach that learns an embedding space where items naturally form clusters: similar items end up grouped close together, while dissimilar items are

pushed farther apart.

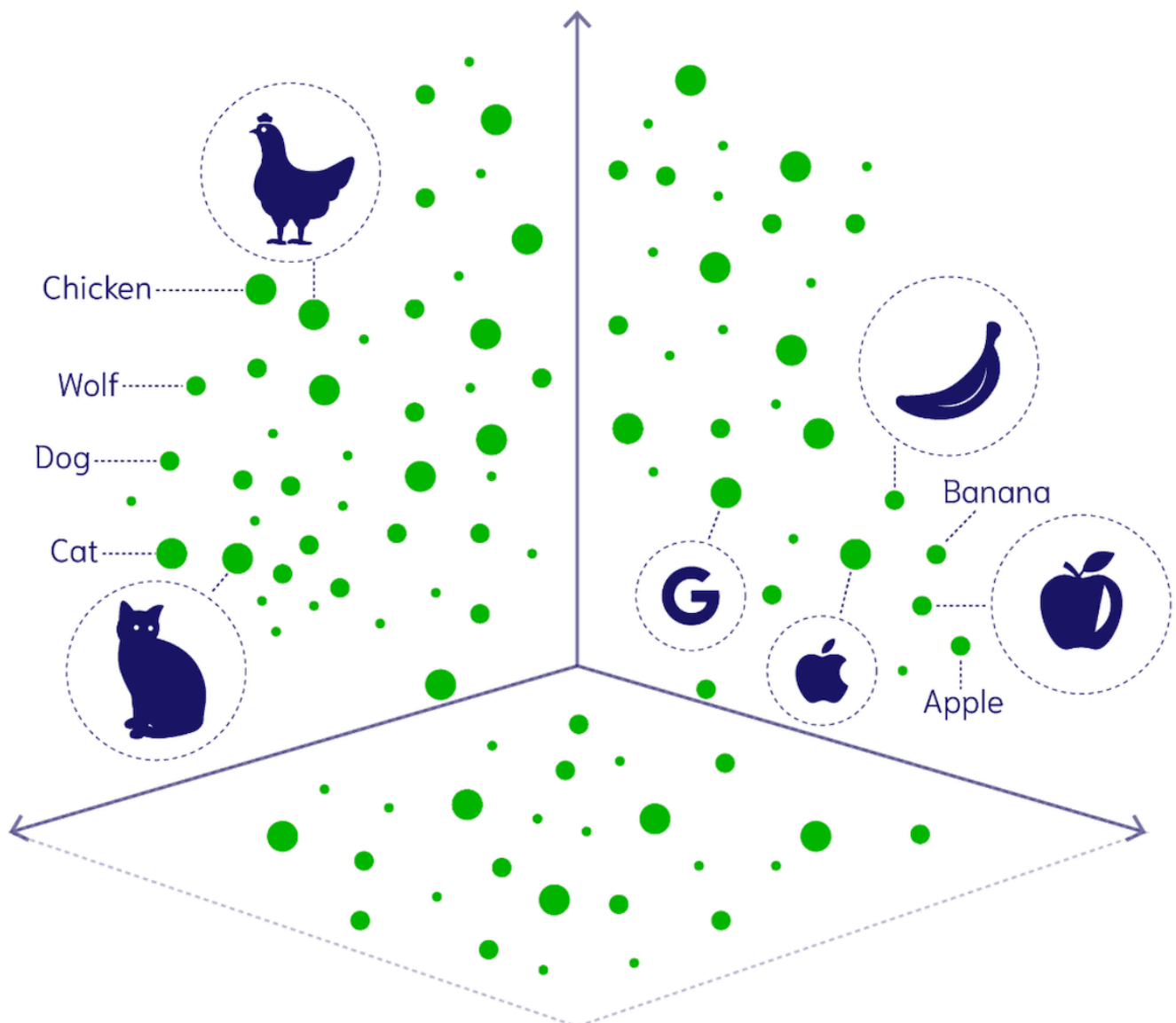
Metric learning is particularly useful for tasks such as face verification, image retrieval, recommendation systems, and few-shot learning where the goal is to understand relationships between data points rather than just categorizing them.

Embedding Space

DEFINITION

An **embedding space** is a learned vector representation where each data point is mapped to a high-dimensional vector. The key property is that the distance or similarity between vectors should reflect the semantic similarity between the original data points.

The embedding is typically extracted from a neural network's intermediate layers, often from the backbone feature extractor before the final classification layers. These embeddings capture the essential features of the input data in a compact vector form.



Distance Functions

Metric learning relies on distance functions to measure similarity between embeddings. The most commonly used distance functions are:

Euclidean Distance

DEFINITION

Euclidean distance measures the straight-line distance between two points in the embedding space:

$$d_2(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^n (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

Cosine Distance

DEFINITION

Cosine distance measures the angle between two vectors, making it invariant to the magnitude of the embeddings:

$$d_c(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

When embeddings are normalized (unit vectors), cosine similarity becomes equivalent to the dot product, and cosine distance can be computed as:

$$d_c(\mathbf{x}, \mathbf{y}) = 1 - \mathbf{x}^T \mathbf{y}$$

Triplet Loss

DEFINITION

Triplet loss is a contrastive loss function that learns embeddings by comparing triplets of examples: an anchor, a positive (same class), and a negative (different class).

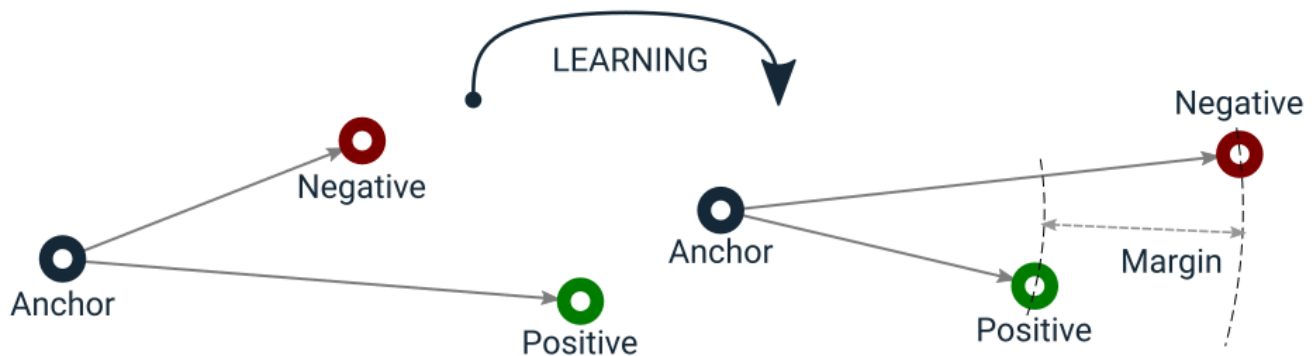
The triplet loss function encourages the distance between anchor and positive to be smaller than the distance between anchor and negative by at least a margin α :

$$\ell_{\text{triplet}} = \max(0, d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + \alpha)$$

Where:

- \mathbf{a} = anchor embedding

- p = positive embedding (same class as anchor)
- n = negative embedding (different class from anchor)
- $d(x, y)$ = distance function (Euclidean or cosine)
- α = margin parameter



Triplet Mining

IMPORTANT

Effective triplet loss training requires careful **triplet mining** - the selection of informative triplets that contribute to learning. Random triplet selection often leads to many trivial triplets that don't improve the model.

Common triplet mining strategies include:

- **Hard negative mining:** Select the closest negative examples
- **Semi-hard negative mining:** Select negatives that are farther than positive but within the margin
- **Random sampling:** Randomly select triplets

Multi-task Learning with Metric Learning

In practical applications, metric learning is often combined with other tasks to create more robust models. A common approach involves:

Combined Loss Function

In our task we combine triplet loss with classification and segmentation losses to train a multi-task model. The total loss combines multiple objectives:

$$\ell_{\text{total}} = \alpha \cdot \ell_{\text{cls}} + \beta \cdot \ell_{\text{seg}} + \gamma \cdot \ell_{\text{triplet}}$$

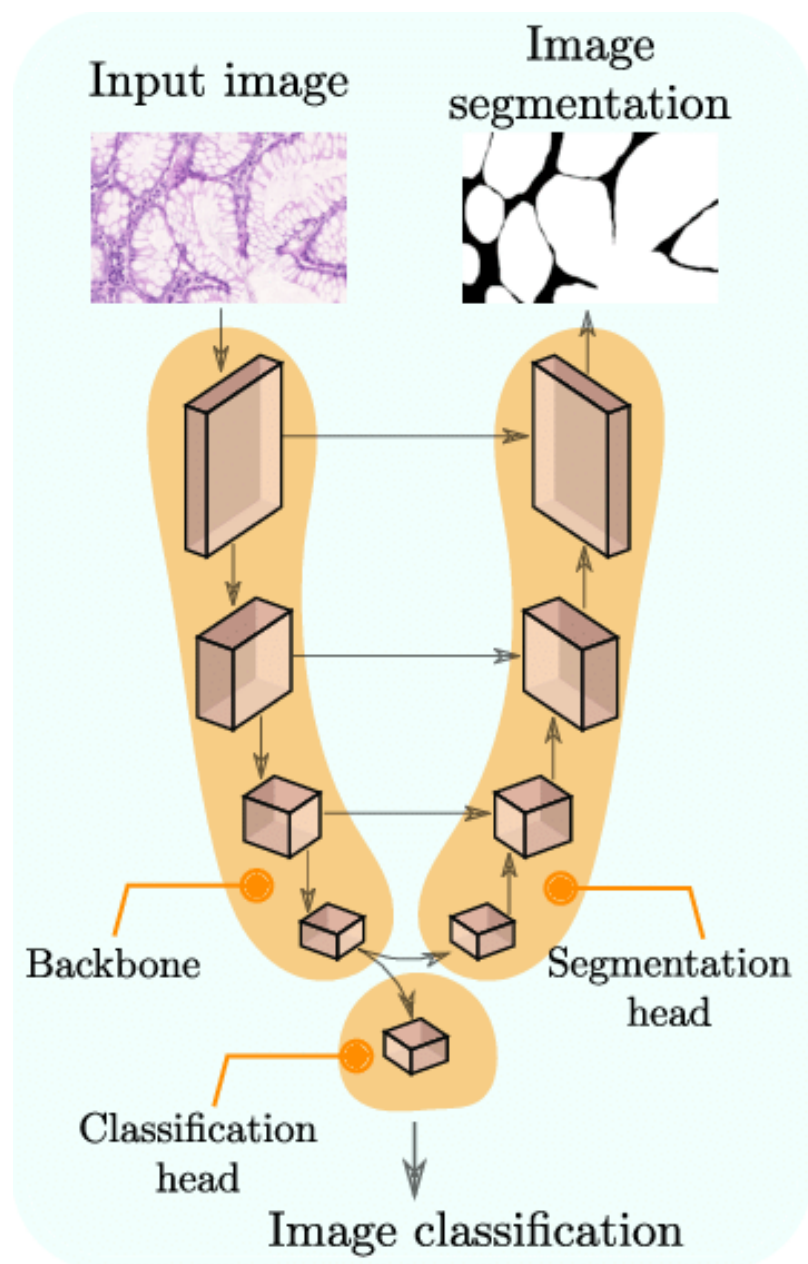
Where:

- ℓ_{cls} = Classification loss (e.g., Cross Entropy)
- ℓ_{seg} = Segmentation loss (e.g., Binary Cross Entropy)
- ℓ_{triplet} = Triplet loss for embedding learning
- α, β, γ = Weight coefficients for balancing different losses

Model Architecture

A typical multi-task model with metric learning consists of:

- 1 **Backbone:** Feature extractor (usually CNN) that processes input data
- 2 **Classification Head:** Predicts discrete class labels
- 3 **Segmentation Head:** Performs pixel-wise predictions
- 4 **Embedding Extraction:** Uses backbone features for similarity learning



Evaluation Metrics

TPR and FPR

DEFINITION

True Positive Rate (TPR), also known as sensitivity or recall, measures the proportion of actual positives correctly identified: $TPR = \frac{TP}{TP+FN}$ Where:

- TP = True Positives
- FN = False Negatives

DEFINITION

False Positive Rate (FPR) measures the proportion of actual negatives incorrectly identified as positives: $FPR = \frac{FP}{FP+TN}$ Where:

- FP = False Positives
- TN = True Negatives

ROC Curve and AUC

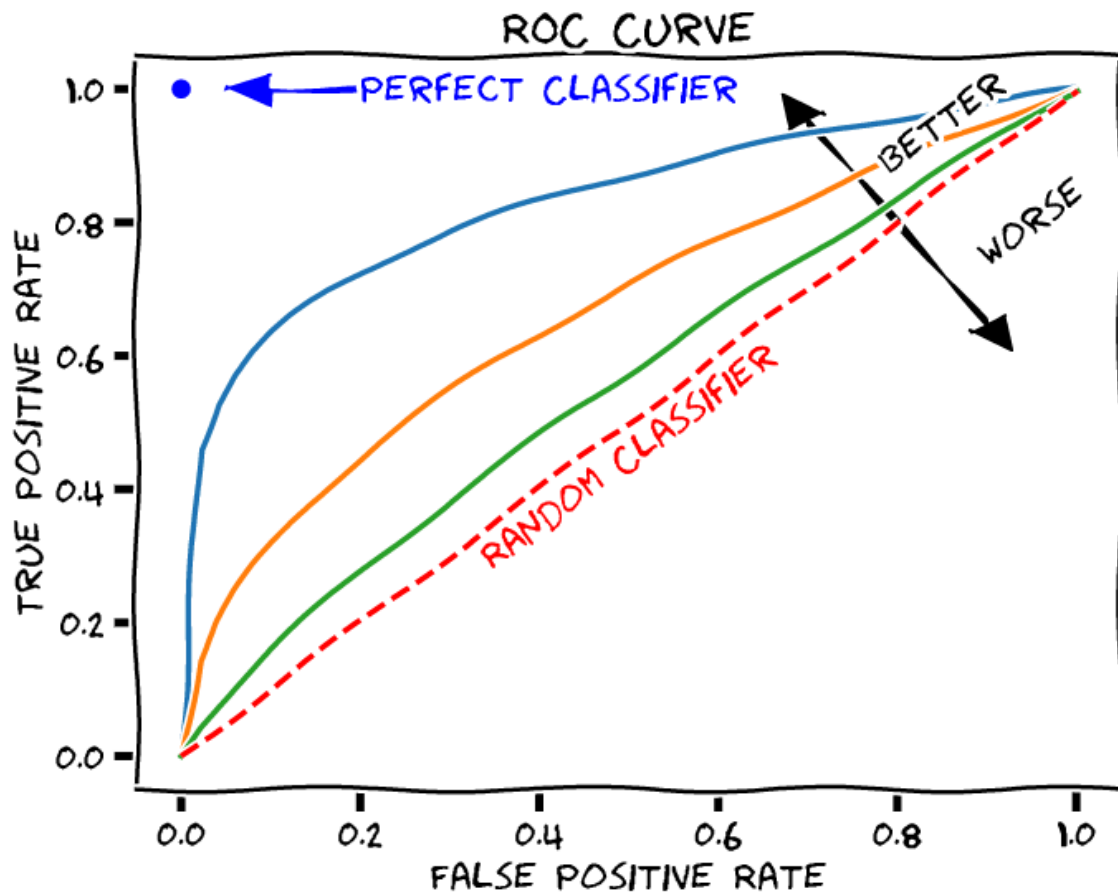
DEFINITION

The **ROC (Receiver Operating Characteristic) curve** plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings for binary classification.

DEFINITION

The **Area Under the Curve (AUC)** provides a single scalar value summarizing model performance across all thresholds. Higher AUC indicates better discriminative ability.

$$AUC = \int_0^1 TPR(FPR) dFPR = \int_0^1 TPR(FPR^{-1}(u)) du$$



TPR@FPR

DEFINITION

TPR@FPR=5% represents the True Positive Rate when the False Positive Rate is constrained to 5%. This metric is particularly useful when the cost of false positives is high.

Training Considerations

Normalization

IMPORTANT

Embedding normalization is crucial for effective metric learning. Normalizing embeddings to unit vectors ensures that cosine similarity equals dot product and prevents magnitude differences from dominating the distance computation.

Margin Selection

The margin parameter α in triplet loss controls the separation between positive and negative pairs. A well-chosen margin helps the model learn meaningful embeddings without collapsing them.

Batch Composition

Effective triplet loss training requires batches with sufficient diversity. Each batch should contain multiple examples from different classes to enable meaningful triplet formation.

Expected Knowledge

Answer the following questions to test your understanding of metric learning concepts.

1. Distance Functions and Embedding Space

Scenario: You have two normalized embeddings (unit vectors):

- Embedding A: [0.6, 0.8, 0.0]
- Embedding B: [0.8, 0.6, 0.0]

Calculate:

- The Euclidean distance between A and B
- The cosine distance between A and B
- Explain why the cosine distance equals $1 - \text{dot_product}$ for normalized embeddings
- Which distance metric is more appropriate for normalized embeddings? Why?

2. Triplet Loss Understanding

Given a triplet:

- Anchor: face image of Person X
- Positive: another face image of Person X
- Negative: face image of Person Y
- Margin $\alpha = 0.2$

After computing embeddings:

- $\text{Distance}(\text{anchor}, \text{positive}) = 0.5$
- $\text{Distance}(\text{anchor}, \text{negative}) = 0.6$

Questions:

- Calculate the triplet loss for this example
- Do you think that this is a "hard" or "easy" triplet?
- What would happen if $\text{distance}(\text{anchor}, \text{negative}) = 1.0$ instead?
- Why do we need the margin parameter α ?

3. Multi-task Learning

Architecture: You're training a model with three objectives:

- Classification loss: 2.5
- Segmentation loss: 0.8
- Triplet loss: 0.3

With loss weights: $\alpha = 1.0$, $\beta = 2.0$, $\gamma = 5.0$

Calculate:

- The total combined loss
- Which task contributes most to the gradient updates?
- If triplet loss becomes 0.0 (all triplets satisfied), what happens to embedding learning?
- Explain why we might want different weights for different losses

4. Evaluation Metrics

Verification task results at threshold=0.5:

- True Positives (TP): 85 (correctly identified as same person)
- False Positives (FP): 10 (incorrectly identified as same person)
- True Negatives (TN): 180 (correctly identified as different people)
- False Negatives (FN): 25 (incorrectly identified as different people)

Calculate:

- True Positive Rate (TPR)
- False Positive Rate (FPR)
- If we require $\text{TPR@FPR}=5\%$, what does this constraint mean in practical terms?
- Why might we prefer $\text{TPR@FPR}=5\%$ over overall accuracy for face verification?