Understanding Loss Functions: Driving Model Success

Presented By: Massa Baali

Carnegie Mellon University

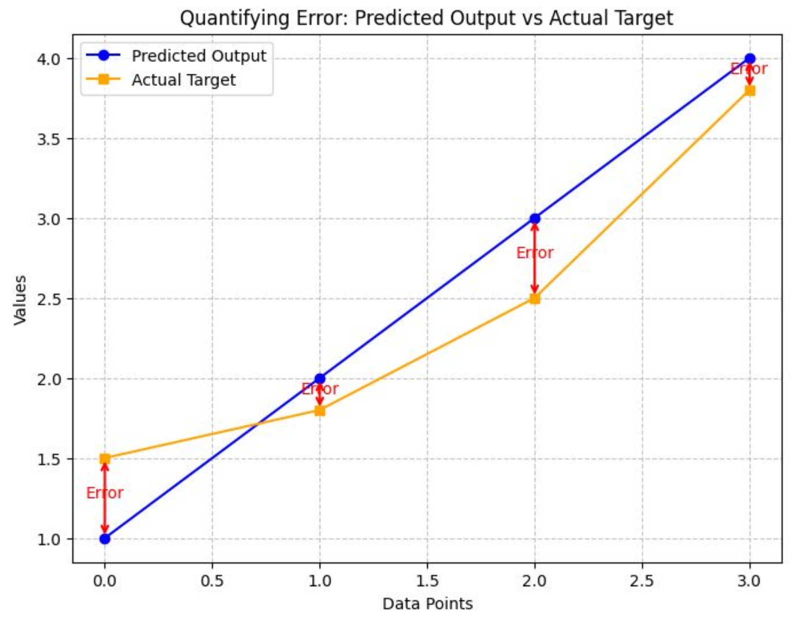
Outline

Introduction

- Think of training a model like driving a car toward a destination But how does the car know where to go? How does it correct its path when it veers off?
- The loss function tells the model how far it is from the truth. It quantifies the “wrongness” of a prediction, then helps the mode steer back on course.
- Without a loss function, the model is like a car without a steering wheel. “ U have no idea how to reach the destination”.

Definition

- A **loss** is a metric that measures the performance of a model.
- **How?**
- It quantifies the error between the predicted output of a model and the actual target.



Carnegie Mellon University

Importance of Loss

How does it improve the model?

- It tells us how much the model’s predictions differ from the actual answers and helps **improve** the model during training by updating the model's weights.

We need Loss to:

- Evaluate the model's performance.
- Guide the learning process to improve predictions

Type of Losses

1. Basic Losses

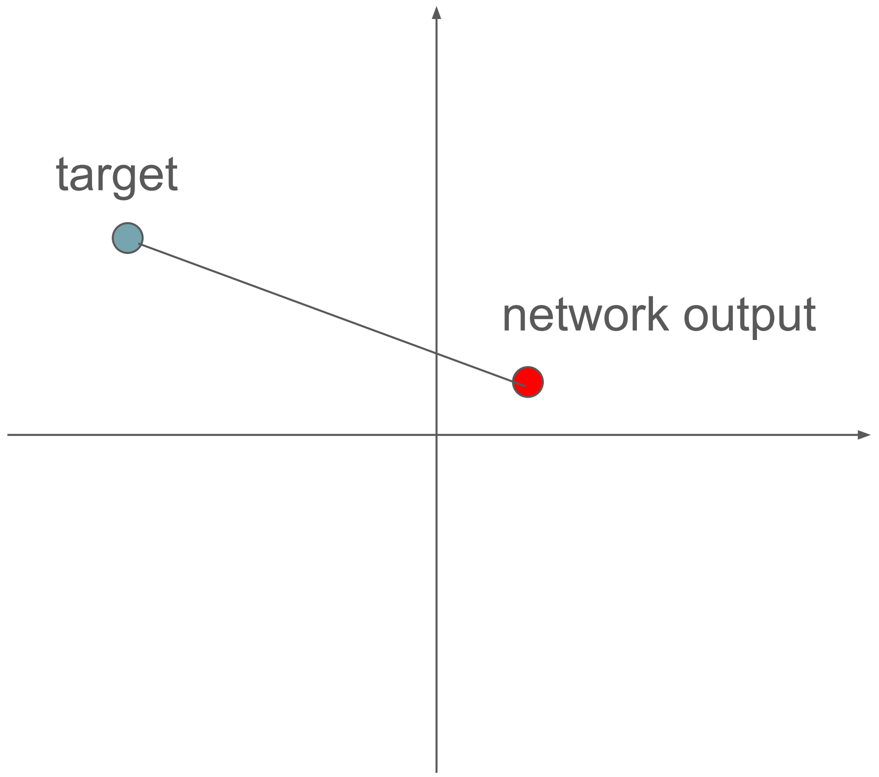
- Definition: Widely used in foundational deep learning tasks.
- Use Cases:
  - Regression Tasks:
    - Mean Squared Error (MSE)
  - Classification Tasks:
    - Cross-Entropy Loss (CE)
    - Binary Cross-Entropy (BCE)

2. Zero-Shot Losses

- Definition: Designed for tasks where the model must generalize to unseen classes.
- Use Cases:
  - Applications: Face recognition, speaker verification, and image retrieval.
    - Angular SoftMax
    - Additive Margin SoftMax
    - Triplet Loss

Distance-based losses for regression

- In regression problems we want our network's prediction to be as close to the target value as possible
  - We want to *minimize the distance between the prediction and target*
- A standard measure for the distance between two points is the squared Euclidean distance between the two
  - The L2 divergence
- The average Euclidean distance between the points in your batch of training data is the "mean-squared error"



Mean Squared Error

Predicting House Prices

- A real estate model predicts the price of houses based on features like size, location, and number of rooms.

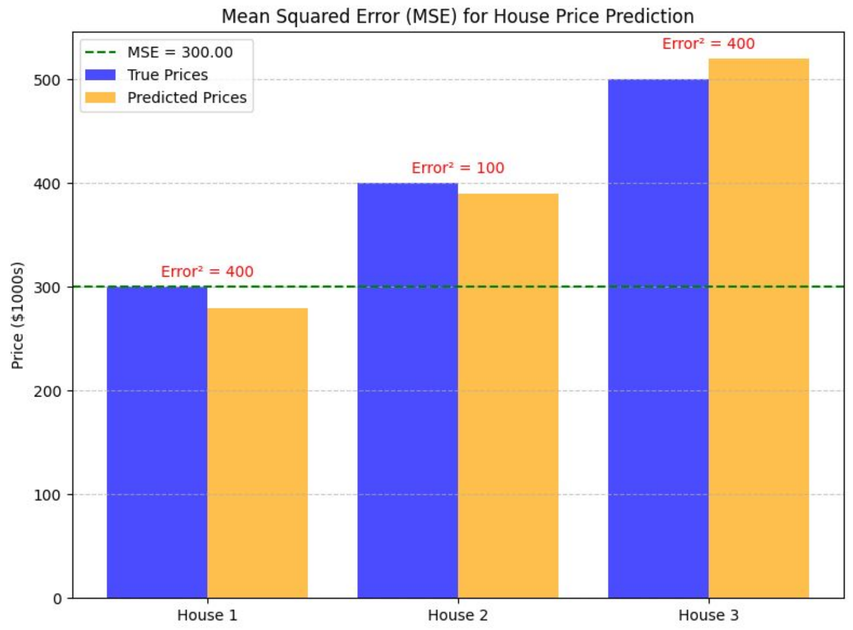
Objective: minimize MSE during training

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Number of data points

Predicted value

Groundtruth



Carnegie Mellon University

Classification losses

- In **classification** problems, the network must output the probability for the class of the training data
  - Network output: $P(y_i)$ for each of the classes i
  - Typically computed using a ***softmax* function** (more on it shortly)
- Ideal output:
  - $P(y_i)$ must be high (ideally 1) for the target class
  - $P(y_i)$ must be low for the remaining classes
- This is captured by the cross-entropy loss
  - -\sum_i ytarget_i log (yhat_i)
  - ytarget_i is either 1 (for target class) or 0 (for non-target classes)
  - So the formula actually gives us loss = -log(yhat_{target_class})
  - Note that this 0 when the network outputs a perfect 1.0 for the target class, and greater than 0 otherwise

Cross Entropy

Classifying Images

- A model predicts the category of images. (Cat. Dog, Bird)

Objective: Minimize the loss, ensuring predictions are closer to the actual values.

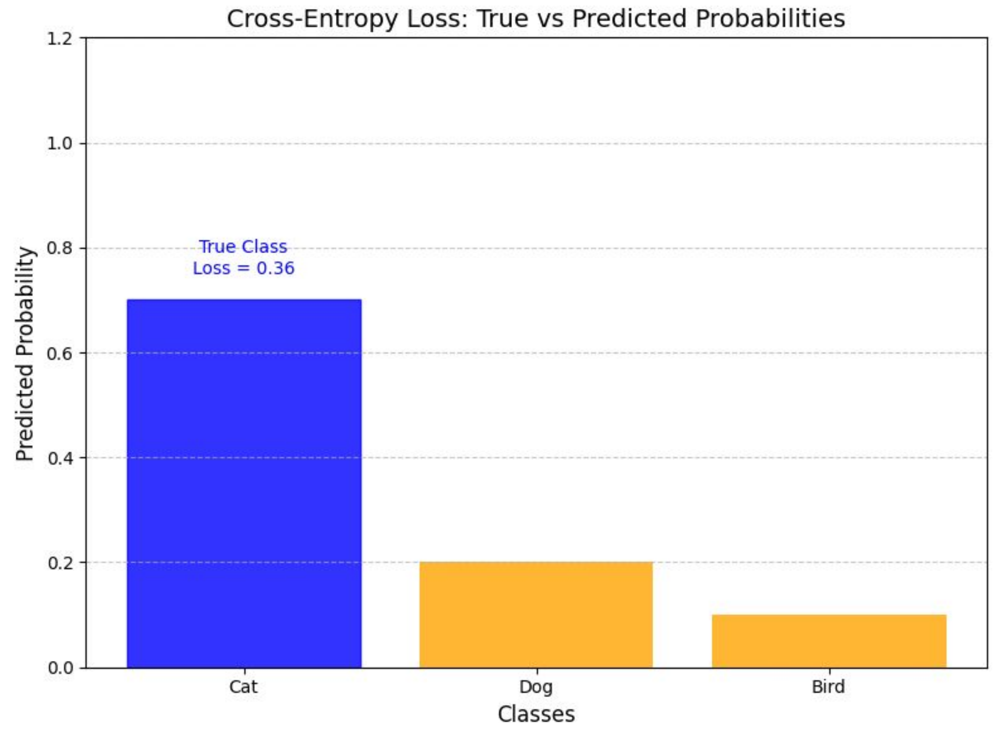
Number of classes

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

Number of data points

Target probability for class c

Predicted probability for class c



Binary Cross Entropy

- In some problems the network only identifies a single class.
  - E.g. “Is this the picture of a leaf infected by fungus”
- In this case the network only outputs P(y) for the target class
  - Must be 1 for positive training instances (e.g.   a picture of a real leaf with fungus)
  - Must be 0 for negative training instances (e.g. a picture of a healthy leaf)
- Here, although there is only one output, there are, *implicitly,* two outputs:
  - Probability of class P(ypos) = P(y),   and probability of *not the* class, P(yneg) = 1-P(y)
- The *binary cross entropy* is just the standard cross entropy for single-class classifiers, where the CE is computed over both outputs P(ypos) and P(yneg)

Binary Cross Entropy

Spam Email Detection

- A model predicts whether an email is spam or not spam.

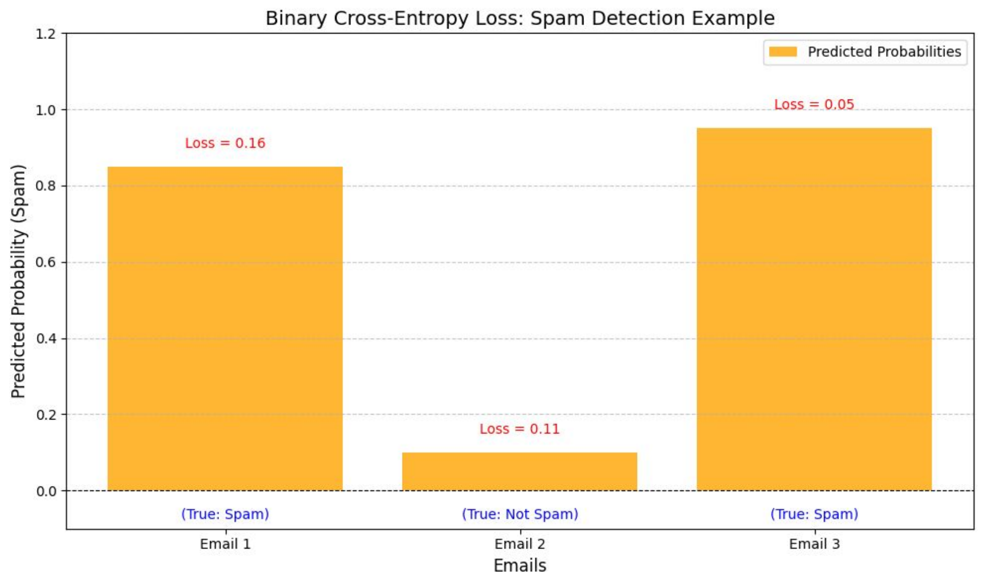
Objective: Minimize the loss, encouraging the model to output probabilities close to the true label.

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Number of data points

Target probability

Predicted probability



Carnegie Mellon University

Zero-shot loss

- In the classification losses seen so far we have tried to train a network to predict the specific classes in the training data
  - The objective is to maximize classification accuracy of the specific classes seen

- In other “zero-shot” problems, our intention is not to teach the network about specific classes, but rather to derive representations where instances of a class **cluster together**, and far away from instances of other classes
  - The specific classes we provide during training are now just examples of classes
  - The network must learn to learn the generic concept of clustering instances from a class together from these example classes

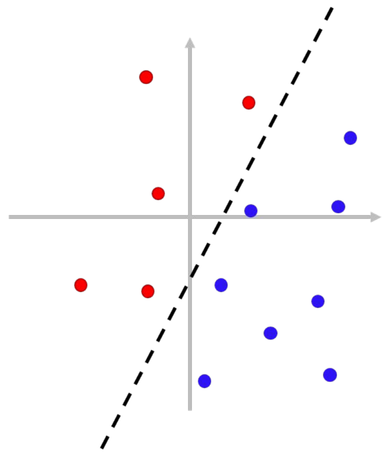
- Zero-shot losses promote such learning

Carnegie Mellon University

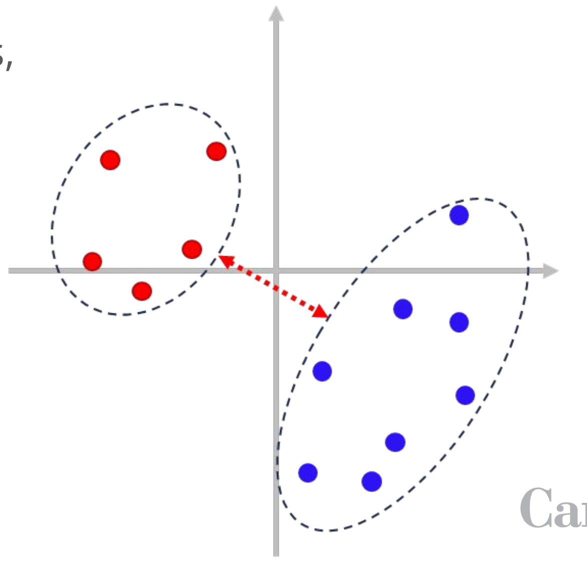
Zero-shot loss vs. conventional classification loss

- In conventional classification problems, the network merely learns to model the classification boundary between classes in the feature space
    - As long as the classes are perfectly separated, it is happy

- In zero-shot problems we try to minimize the distance between data points of the same class, while maximizing the distance between data points of different classes



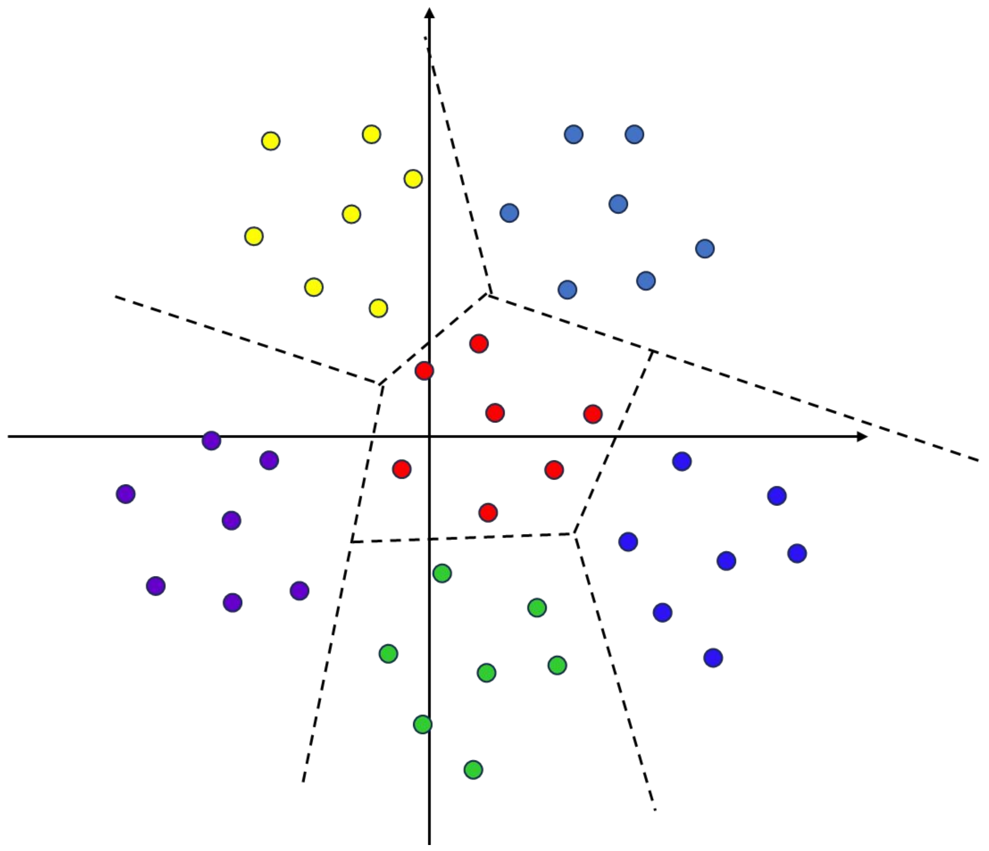
Conventional classification models learn the boundary between classes



Zero-shot training maximizes separation between classes

Cross-Entropy loss

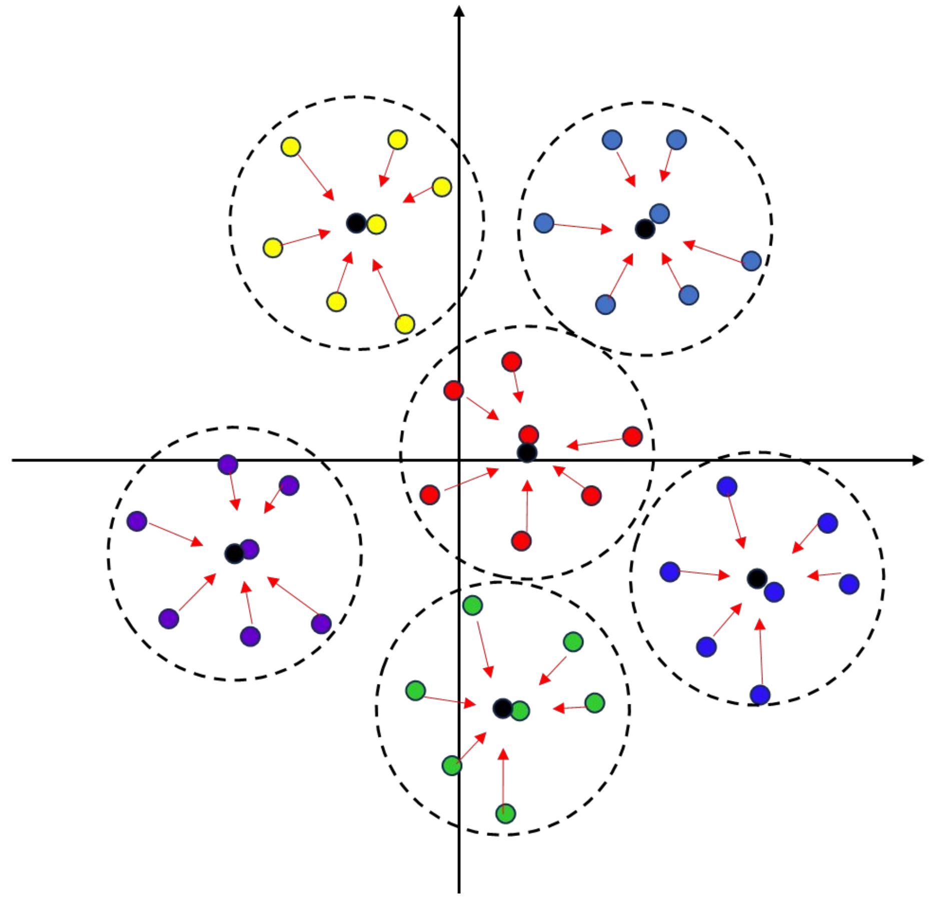
- The cross-entropy loss is also actually a zero-shot loss
- The network learns to extract features that are separable from each other by linear boundaries
- In doing this, it learns to generate features where features from the same class cluster together
  - But only provided there are a **sufficiently large** number of "example" classes
- We have already seen the CE loss



Carnegie Mellon University

Centerloss

- In Centerloss we explicitly try to “shrink” the classes by moving the feature vector for each data instance towards the center of its class.
- The loss itself is the total squared distance of all data points from the center of their respective classes.
  a. The center is also **learned**
- This loss is used along with the CE loss



Carnegie Mellon University

Centerloss

Face Recognition Systems

- Center Loss is used in conjunction with other losses (e.g., CE) to enhance face recognition systems.
- A model learns embeddings for face images, ensuring that all embeddings belonging to the same person are grouped closer to their "center" in the embedding space.

Objective: Minimize the distance between the feature vectors of samples from the same class and their class center.

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

Feature vector for ith sample

Center for class y_i

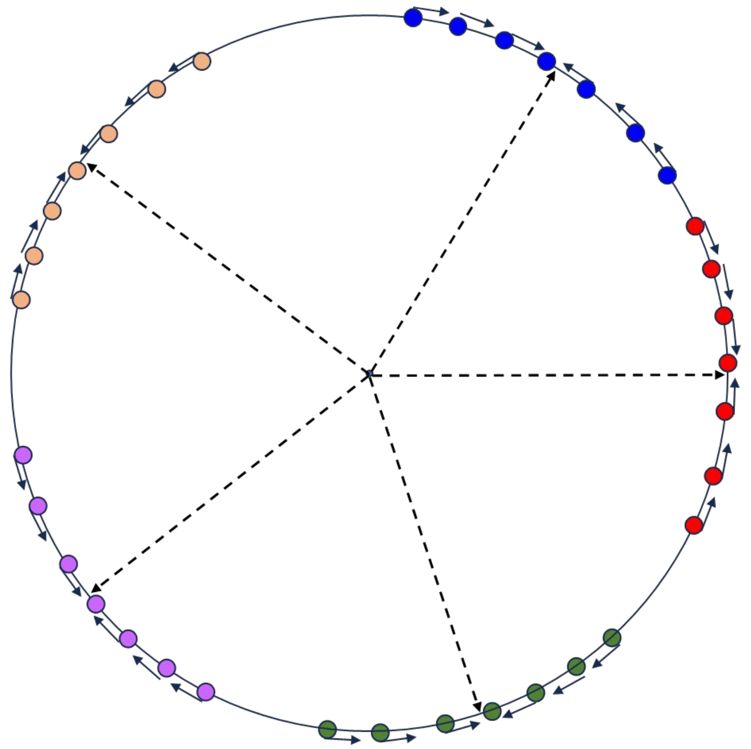
Total number of samples in the batch

Carnegie Mellon University

Angular Losses

- Angular losses distribute the feature vectors on the surface of a sphere by normalizing their length

- Subsequently they work by just minimizing the ***angle*** between each data point and the centre for its class
  - This also allows us to introduce a "margin" – an additional penalty for being away from the center



Carnegie Mellon University

Angular SoftMax

Face Recognition Systems

- Think of a face recognition system that checks people's identities by comparing their faces to photos in a database.
- The system needs to make sure that face representations (embeddings) for different people are very different, while face representations for the same person are very similar.

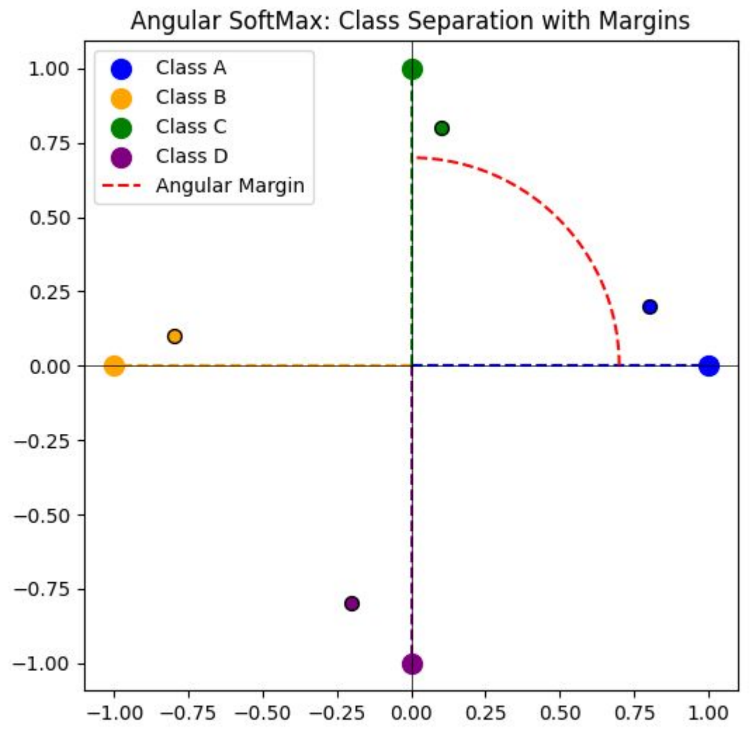
Objective: Maximize the cosine similarity for the true class while enforcing a **margin** between classes in the **angular space**.

$$L = -\log \frac{e^{s\cdot(\cos(m\theta_y))}}{e^{s\cdot(\cos(m\theta_y))} + \sum_{j\neq y} e^{s\cdot\cos(\theta_j)}}$$

Scaling factor to stabilize gradients

Angle between the embedding and the correct class center

Cosine similarity with an angular margin m

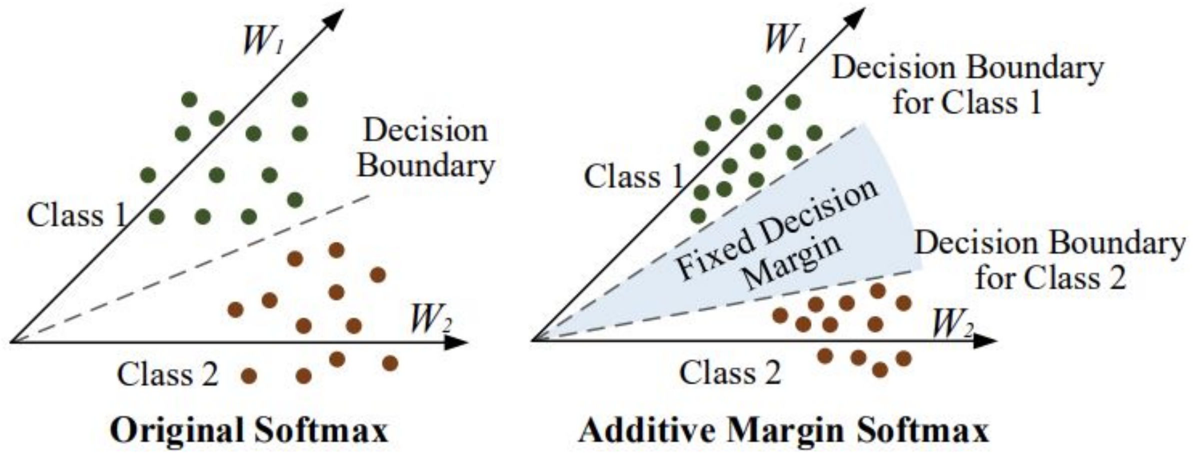


Additive Angular Margin SoftMax

- It helps make classes more distinct in a classification problem. It does this by adding an angular margin between classes in the decision boundary.
- It introduces an angular margin to make it harder for the model to classify samples.
- By adding the margin, the decision boundary becomes stricter:
    a. **Same-class** samples are pulled closer together.
    b. **Different-class** samples are pushed farther apart.
- This improves the model's ability to separate classes, especially in challenging tasks.

The angular softmax, as described by Liu et al. (2017a), can only impose an **unfixed angular margin**, whereas the additive margin softmax incorporates a **fixed hard angular margin**.

Wang, F., Cheng, J., Liu, W., & Liu, H. (2018). Additive margin softmax for face verification. *IEEE Signal Processing Letters*, *25*(7), 926-930.



Additive Margin SoftMax

Speaker Verification

- A system needs to determine if two speakers belong to the same person.
- The embeddings of different speakers should be well-separated, while embeddings of the same speaker should remain compact.

Objective: Maximize the cosine similarity for the true class while enforcing an **additive margin** to separate classes in the **embedding** space.

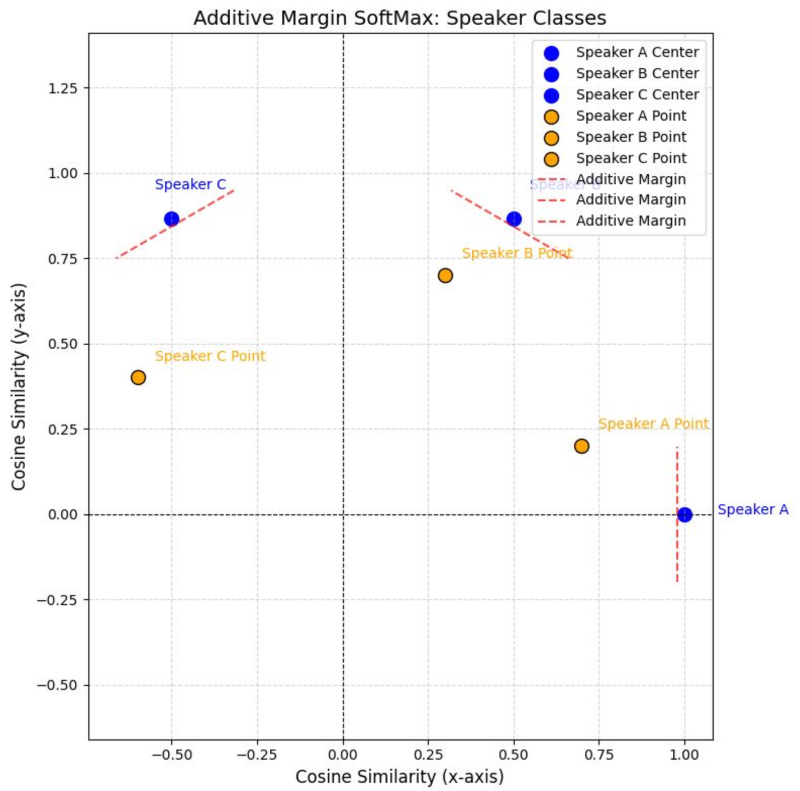
$$L = -\log \frac{e^{s\cdot(\cos(\theta_y)-m)}}{\sum_{j=1}^{C} e^{s\cdot\cos(\theta_j)}}$$

Additive margin

Cosine similarity for the true class

Scaling factor to stabilize gradients

Number of classes

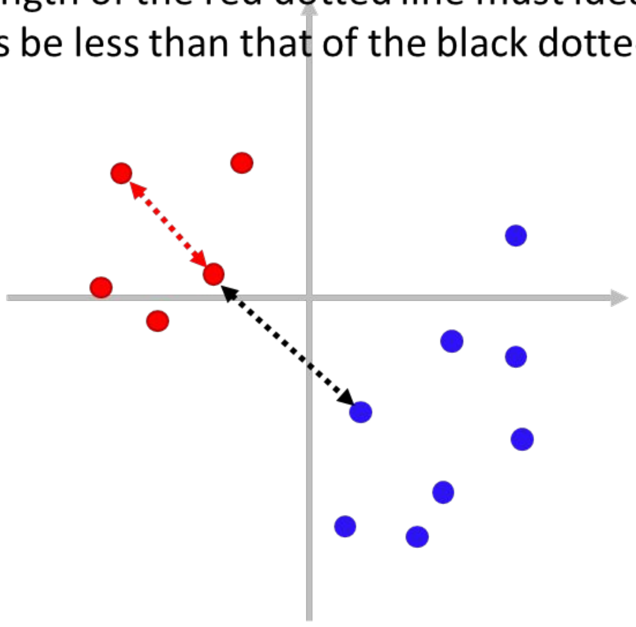


Contrastive Losses

- The losses we have seen so far have all focused on the relationship of data instances to their classes
  - As a result, the loss for each data instance can be individually calculated

- *Contrastive losses* are computed by comparing data points to *each other*
- They are built on the principle that a data point **must always be closer** to another data point from *its* ***own class*** than it is to a data point from any other class

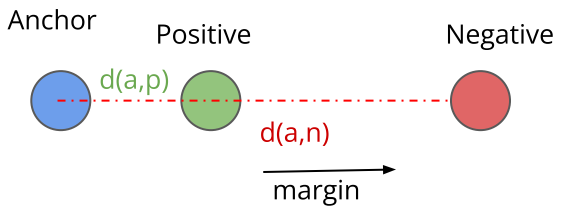
The red dashed arrow shows the distance between two points of the same class. The black arrow shows the distance between two points from different classes.
The length of the red dotted line must ideally always be less than that of the black dotted line



Triplet (the most common contrastive loss)

Face Recognition Systems

- A system needs to determine if two face images belong to the same person (e.g., unlocking a phone with your face).
- Example:
  a. Anchor: Your stored face image.
  b. Positive: A current image of your face.
  c. Negative: A face image of someone else.



Objective: Minimize the distance between anchor and positive embeddings while ensuring they are farther from the negative embedding by a margin α

$$L = \max(0, \|f(x_a) - f(x_p)\|^2 - \|f(x_a) - f(x_n)\|^2 + \alpha)$$

Anchor

Positive (same class as anchor)

Negative (different class from anchor)

Margin that ensures the negative is sufficiently farther than the positive

Carnegie Mellon University

Thank you :-)