Digital Surveillance Systems and Application

FACE RECOGNITION

Jison G.S. Hsu
Artificial Vision Laboratory
Taiwan Tech

Outline

- What is Face Recognition ?
  - Work Flowchart
  - How does it work?
  - Popular Face Recognition Model- VGGFace
- Face Verification
- Face Identification
- Loss Functions for Face Recognition
- Angular Softmax Loss

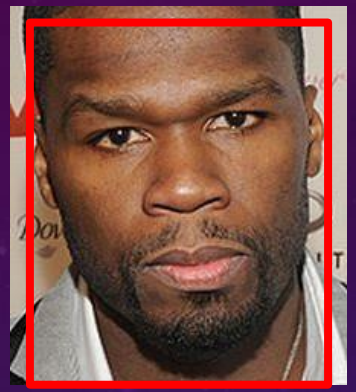
Face Recognition Demo (SenseFace Demo)

https://www.youtube.com/watch?v=k3T2WbRkgvg [8:17]

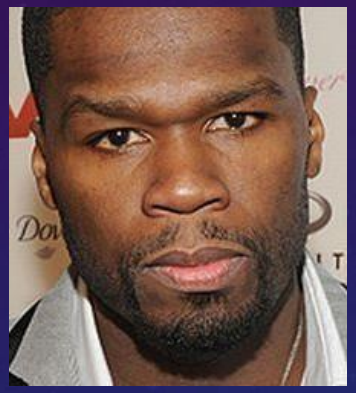
Face Recognition Demo (AVLab)

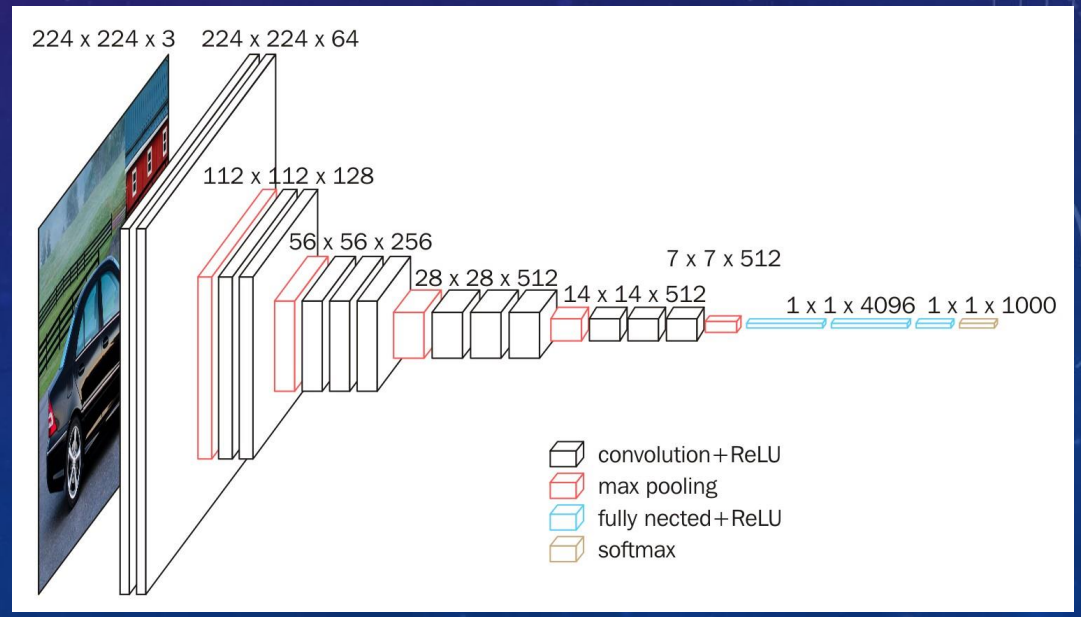
https://www.youtube.com/watch?v=ShrNQpuOVws [2:45]

Work Flowchart



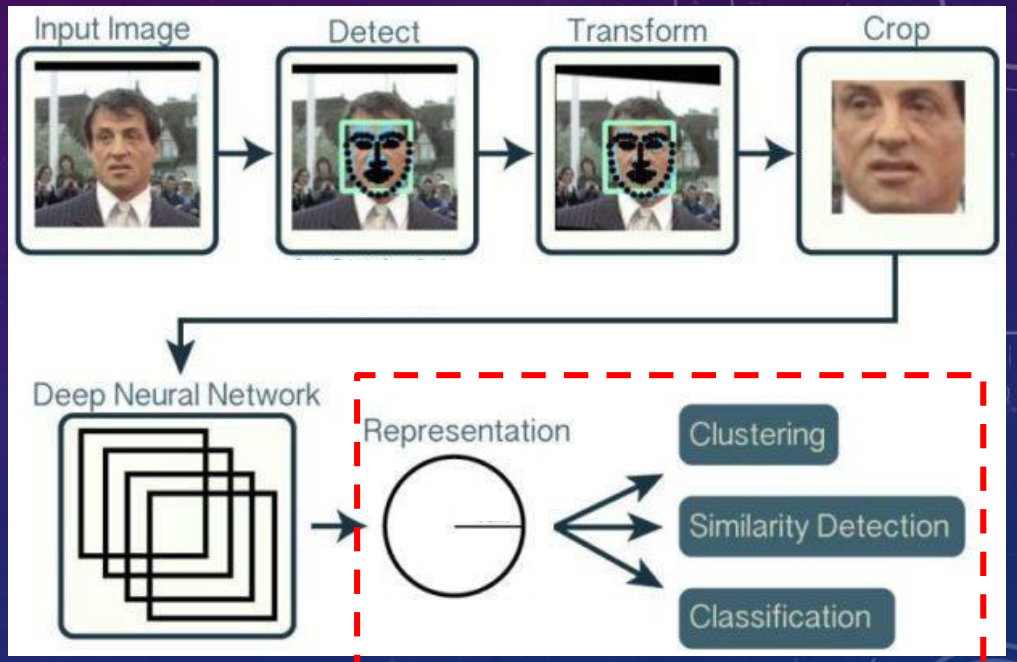
Collect data → Face detector → Facial Landmark → Crop face and resize → Train

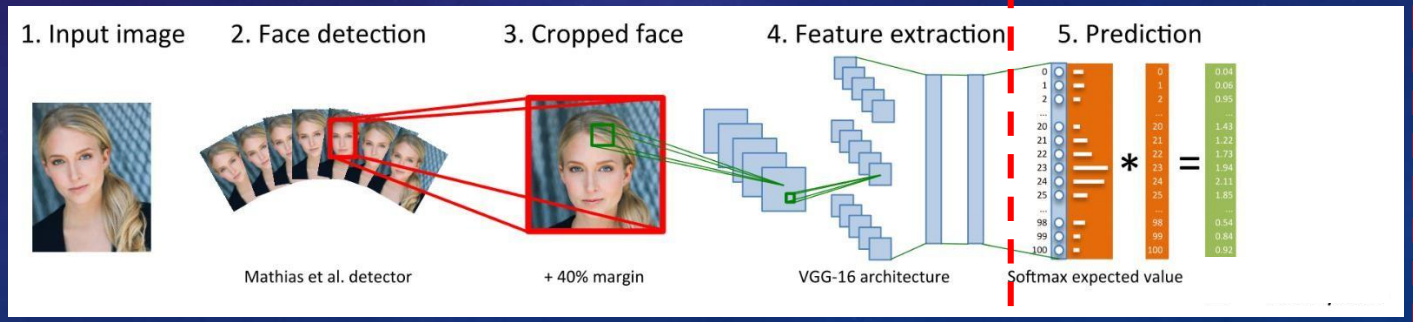




How does it work?

- A model learned about facial attribute (like identity or age)
- The model will extract the image features to predict who you are.





Dlib-models

- Dlib_face_recognition_resnet_model_v1
  - This model is a ResNet network with 29 conv layers. It's essentially a version of the ResNet-34 network from the paper “Deep Residual Learning for Image Recognition by He, Zhang, Ren, and Sun, CVPR 2016” with a few layers removed and the number of filters per layer reduced by half.
  - The network was trained from scratch on a dataset of about 3 million faces.
- Shape_predictor_68_face_landmarks
  - This is trained on the ibug 300-W dataset (https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/)
- Shape_predictor_5_face_landmarks
  - This is a 5 point landmarking model which identifies the corners of the eyes and bottom of the nose. It is trained on the “dlib 5-point face landmark dataset”, which consists of 7198 faces.

Please see the Page 2 in the coding manual.

FAR V.S FRR

- False acceptance rate (FAR): The measure of the likelihood that the biometric security system will incorrectly accept an access attempt by an unauthorized user.
- False rejection rate (FRR) : The measure of the likelihood that the biometric security system will incorrectly reject an access attempt by an authorized user.

Threshold:0.7, FAR:1, FRR:0

FAR V.S FRR

- False acceptance rate (FAR): The measure of the likelihood that the biometric security system will incorrectly accept an access attempt by an unauthorized user.
- False rejection rate (FRR) : The measure of the likelihood that the biometric security system will incorrectly reject an access attempt by an authorized user.

Threshold:0.5, FAR:0, FRR:0.33

Popular Face Recognition Model - VGGFace

- The VGGFace refers to a series of models developed for face recognition and demonstrated on benchmark computer vision datasets by members of the Visual Geometry Group (VGG) at the University of Oxford.
- There are two main VGG models for face recognition at the time of writing; they are VGGFace and VGGFace2.

VGG-Face Model

- The VGGFace model, named later, was described by Omkar Parkhi in the 2015 paper titled “Deep Face Recognition.”
- A contribution of the paper was a description of how to develop a very large training dataset, required to train modern-convolutional-neural-network-based face recognition systems, to compete with the large datasets used to train models at Facebook and Google.

VGG-Face Model

- Even though research paper is named Deep Face, researchers give VGG-Face name to the model. This might be because Facebook researchers also called their face recognition system Deep Face – without blank. VGG-Face is deeper than Facebook's Deep Face, it has 22 layers and 37 deep units.
- The structure of the VGG-Face model is demonstrated below. Only output layer is different than the ImageNet version.

VGG-Face Model

Train a mapping function W' from 4096 to 1024 dimensions by minimizing the triple loss function, and obtain a 1024-dimensional face embedding.
The Triplet loss function is as follows:

- a≥0 is a boundary interval, and T is a set of training triples, which are formed online at the beginning of each iteration.
- Fine-tuning the target data set, only learning the mapping layer.

$$E(W') = \sum_{(a,p,n)\in T} \max\{0, \alpha - \|\mathbf{x}_a - \mathbf{x}_n\|_2^2 + \|\mathbf{x}_a - \mathbf{x}_p\|_2^2\}, \quad \mathbf{x}_i = W' \frac{\phi(\ell_i)}{\|\phi(\ell_i)\|_2}.$$

- Verify identity by comparing whether the Euclidean distance of two 1024-dimensional facial features is less than the threshold.

$$\|W'\phi(\ell_1) - W'\phi(\ell_2)\|_2$$

VGG-Face2 Model

- Qiong Cao, et al. from the VGG describe a follow-up work in their 2017 paper titled "VGGFace2: A dataset for recognizing faces across pose and age."
- They describe VGGFace2 as a much larger dataset that they have collected for the intent of training and evaluating more effective face recognition models.

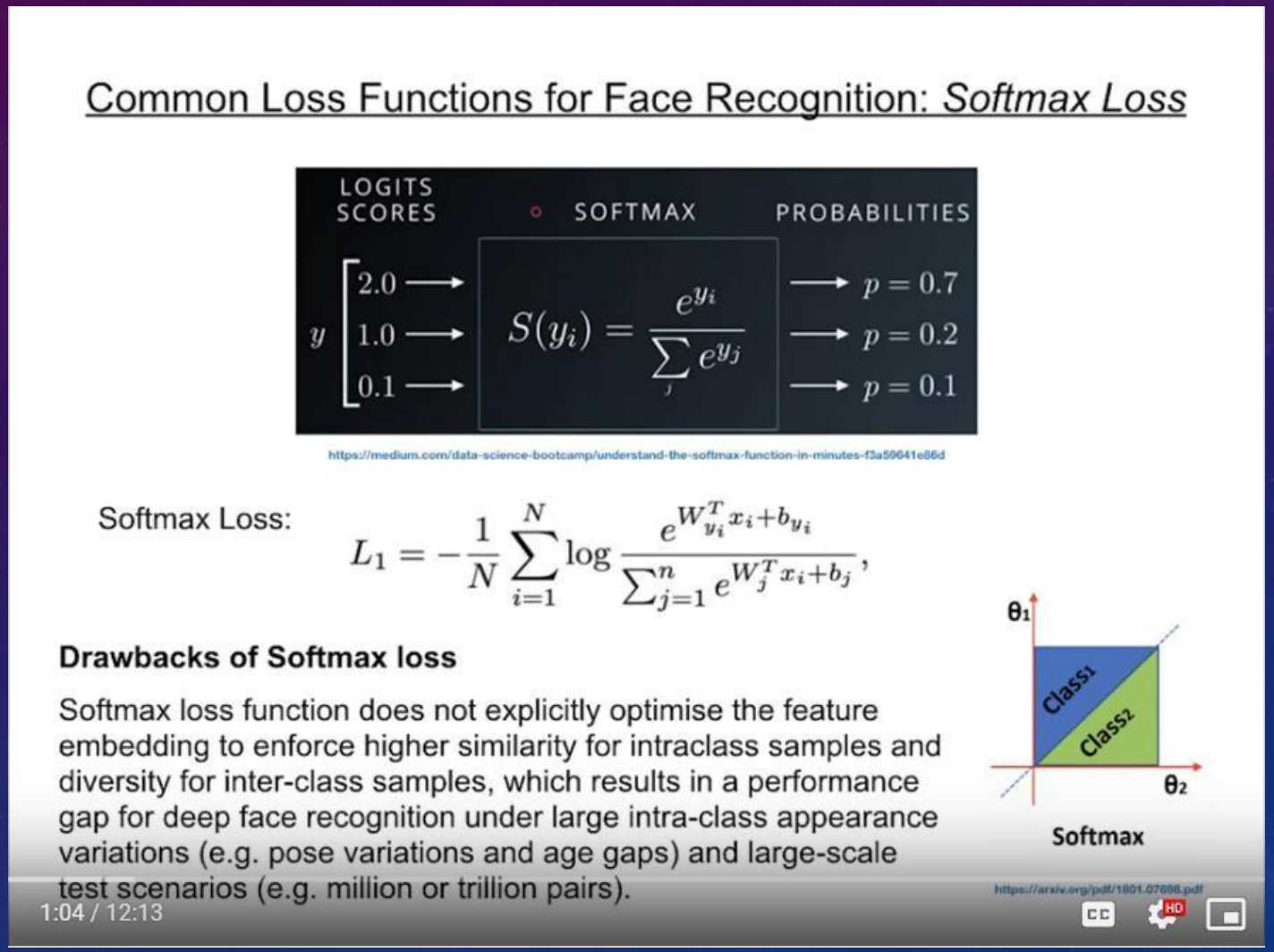
VGG-Face2 Model

- The paper focuses on how this dataset was collected, curated, and how images were prepared prior to modeling.
- VGGFace2 has become the name to refer to the pre-trained models that have provided for face recognition, trained on this dataset.
- Models are trained on the dataset, specifically a ResNet-50 and a SqueezeNet-ResNet-50 model (called SE-ResNet-50 or SENet).
- The models are evaluated on standard face recognition datasets, demonstrating then state-of-the-art performance.

VGG-Face2 Model

- VGGFace2 dataset contains 3.31 million images of 9131 subjects, with an aberage of 362.6 images for each subject.
- Images are downloaded from Google Image Search and have large variations in pose, age, illumination, ethnicity and profession (e.g. actors, athletes, politicians).
- The whole dataset is split to a training set (including 8631 identities) and a test set (including 500 identities).

Loss Functions for Face Recognition



https://www.youtube.com/watch?v=H1qEp_czI1I

Softmax Prediction Boundary

$$P_1 = \frac{\exp(W_1^T x_1 + b_1)}{\exp(W_1^T x_1 + b_1) + \exp(W^T x_2 + b_2)}$$

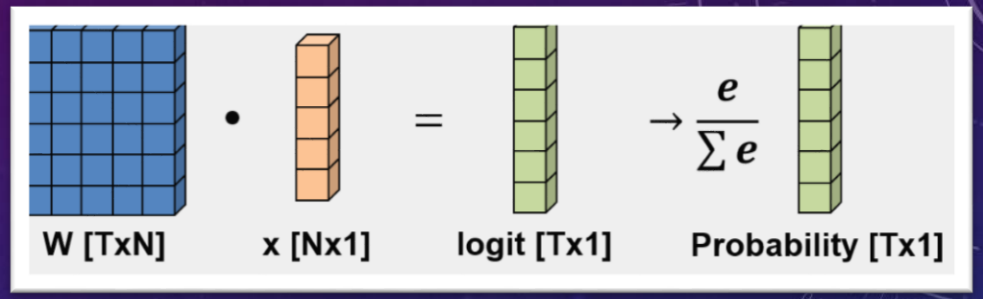
$$P_2 = \frac{\exp(W_2^T x_2 + b_2)}{\exp(W_1^T x_1 + b_1) + \exp(W^T x_2 + b_2)}$$

$Softmax\ Prediction\ Boundary$: $(W_1 - W_2)\mathrm{x} + (b_1 - b_2) = 0$

Which means that $P_1 = P_2$ ， $W^T x_1 + b_1 = W^T x_2 + b_2$

Modified Softmax Loss

$$Wx + b = \|W\|\|x\| + Cos\theta + b$$



Constraint
$\|W_1\| = \|W_2\| = 1$
$b_1 = b_2 = 0$

$$L_i = -\log\left(\frac{e^{a}}{\sum_{k=1}^{T} e^{a_k}}\right) = -\log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j} e^{W_{y_j}^T x_j + b_{y_j}}}\right)$$

$$L_i = -\log\left(\frac{e^{\|x_i\|\cos(\theta_{y_i},i)}}{\sum_{j} e^{\|x_j\|\cos(\theta_j,i)}}\right)$$

$$New\ Decision\ Boundary:\ \|x\|(\cos\theta_1 - \cos\theta_2)$$

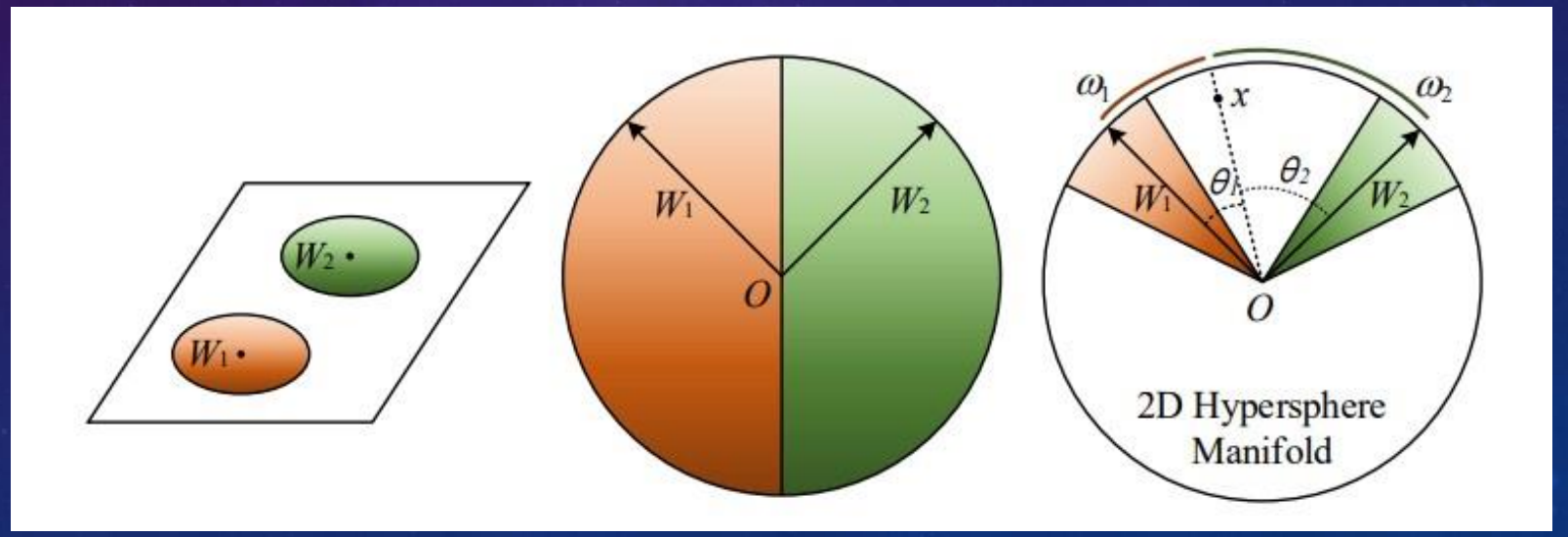
Angular Softmax Loss

$Decision\ Boundary$: $\|x\|(\cos m\theta_1 - \cos\theta_2)$ for Class 1

$\|x\|(\cos\theta_1 - \cos m\theta_2)$ for Class 2

$Produce\ Angular\ Margin = \frac{m-1}{m+1}\theta_{12}$, θ_{12} $is\ angle\ between\ W_1, W_2$, $m \geq 2$

$$L_{ang} = \frac{1}{N}\sum_j -\log\left(\frac{e^{\|x_i\|\cos(m\theta_{y_i},i)}}{e^{\|x\|\cos(m\theta_y,i)} + \sigma_{j\neq y}\, e^{\|x_i\|\cos(m\theta_y,i)}}\right),\ \theta_{y_i} \in [0, \frac{\pi}{m}]$$



Angular Softmax Loss

$$L_{ang} = \frac{1}{N}\sum_{j} -\log\left(\frac{e^{\|x_i\|\cos(m\theta_{y_i,i})}}{e^{\|x\ \|\cos(m\theta_{y}\ ,i)} + \sum_{j\neq y} e^{\|x_i\|\cos(m\theta_{y}\ ,i)}}\right),\ \theta_{y_i} \in [0, \frac{\pi}{m}]$$

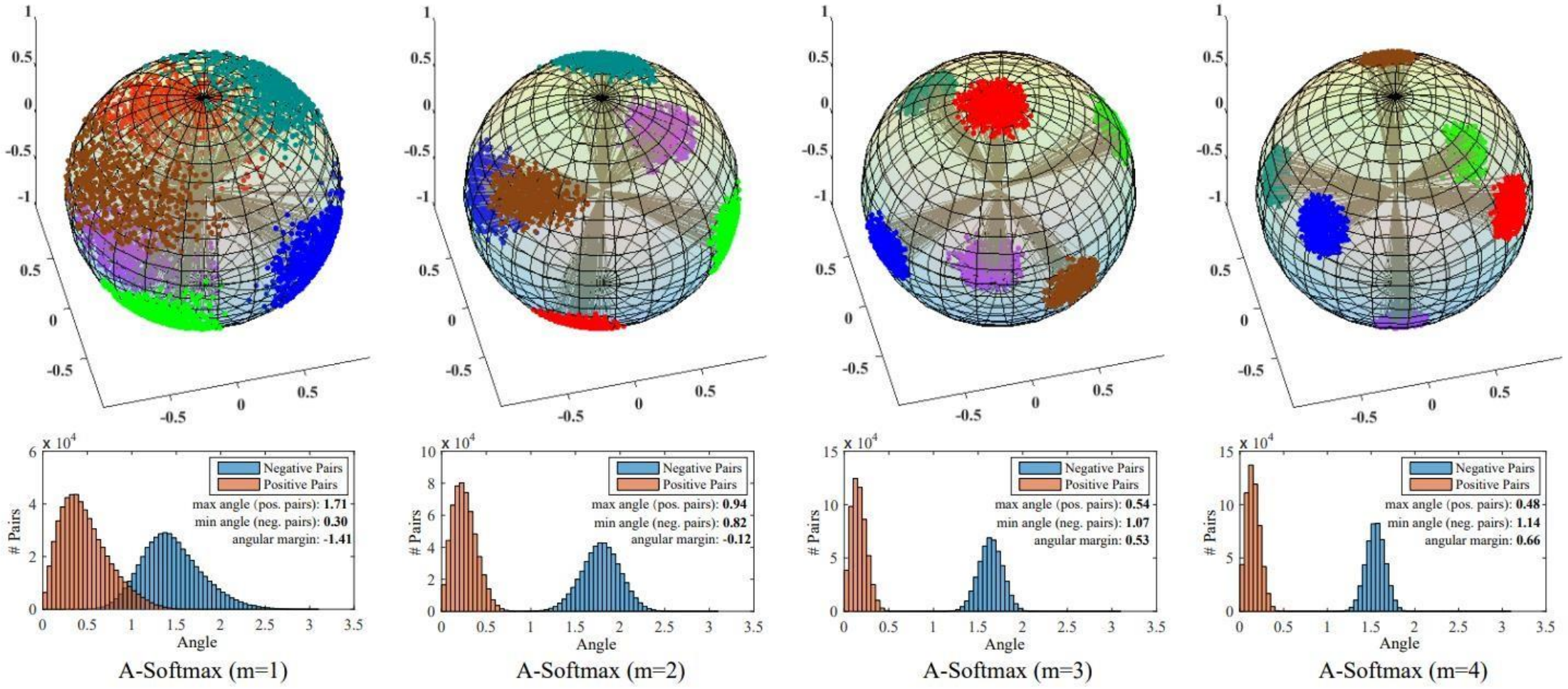
To optimizable in CNN, generalizing it to a monotonically decreasing angle function:

$\varphi(\theta_{y_i}, i) = \cos(\theta_{y_i}, i)$, while $\theta_{y_i} \in \left[0, \frac{\pi}{m}\right]$, m ≥ 2

$$\varphi(\theta_{y_i}, i) = (-1)^k Cos(m\theta_{y_i}, i) - 2k, \theta_{y_i,i} \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$$

$$L_{ang} = \frac{1}{N}\sum_{j} -\log\left(\frac{e^{\varphi(\theta_{y_i},i)}}{e^{\varphi(\theta_{y}\ ,i)} + \sum_{j\neq y} e^{\|x_i\|\cos(m\theta_{y}\ ,i)}}\right),\ \theta_{y_i} \in [0, \frac{\pi}{m}]$$

Angular Softmax



https://www.youtube.com/watch?v=P6jEzzwoYWs

Why do we need regularization?

- An overfitted model performs well on training data but fails to generalize.
- Goal of our machine learning algorithm is to learn the data patterns and ignore the noise in the data set.

How do we solve the problem of overfitting?

- We can solve the problem of overfitting using
  - Regularization
  - Cross Validation
  - Drop out

Reduction of Model Complexity

- Regularization is a technique to discourage the complexity of the model. It does this by penalizing the loss function. This helps to solve the overfitting problem.

$$L(x, y) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

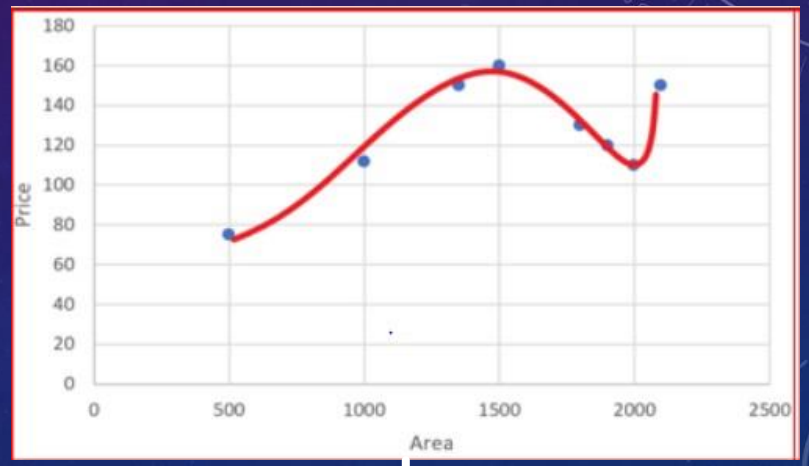
$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x^2_2 + \theta_3 x^3_3 + \theta_4 x^4_4$$

From High-Order to Low-Order Models

- Loss function is the sum of squared difference between the actual value and the predicted value.
- As the degree of the input features increases, the model becomes complex and tries to fit all the data points.
- When we penalize the weights θ_3 and θ_4 and make them too small, very close to zero. It makes those terms negligible and helps simplify the model.

$$L(x, y) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 {x_2}^2 + \theta_3 {x_3}^3 + \theta_4 {x_4}^4$$



$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 {x_2}^2 + \theta_3 {x_3}^3 + \theta_4 {x_4}^4$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 {x_2}^2$$

Weights Kept Small

- To ensure we take into account the input variables, we penalize all the weights by making them small. This also makes the model simpler and less prone to overfitting.
- We have added the regularization term to the sum of squared differences between the actual value and predicted value. Regularization term keeps the weights small making the model simpler and avoiding overfitting.
- λ is the penalty term or regularization parameter which determines how much to penalizes the weights.

$$L(x, y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$$

where $h_\theta x_i = \theta_0 + \theta_1 x_1 + \theta_2 {x_2}^2 + \theta_3 {x_3}^3 + \theta_4 {x_4}^4$

$$L(x, y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

Regularization Parameter

- When λ is zero then the regularization term becomes zero. We are back to the original Loss function.
- When λ is large, we penalizes the weights and they become close to zero. This results is a very simple model having a high bias or is underfitting.

$$L(x, y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + 0$$

where λ is zero

$$h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

where λ is very large

L_1 Regularization (L_1 norm)

- In L_1 norm we shrink the parameters to zero. When input features have weights closer to zero that leads to sparseL_1norm. In Sparse solution majority of the input features have zero weights and very few features have non zero weights.
- L_1 regularization does feature selection. It does this by assigning insignificant input features with zero weight and useful features with a non zero weight

$$L(x, y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$

$L1$ regularization

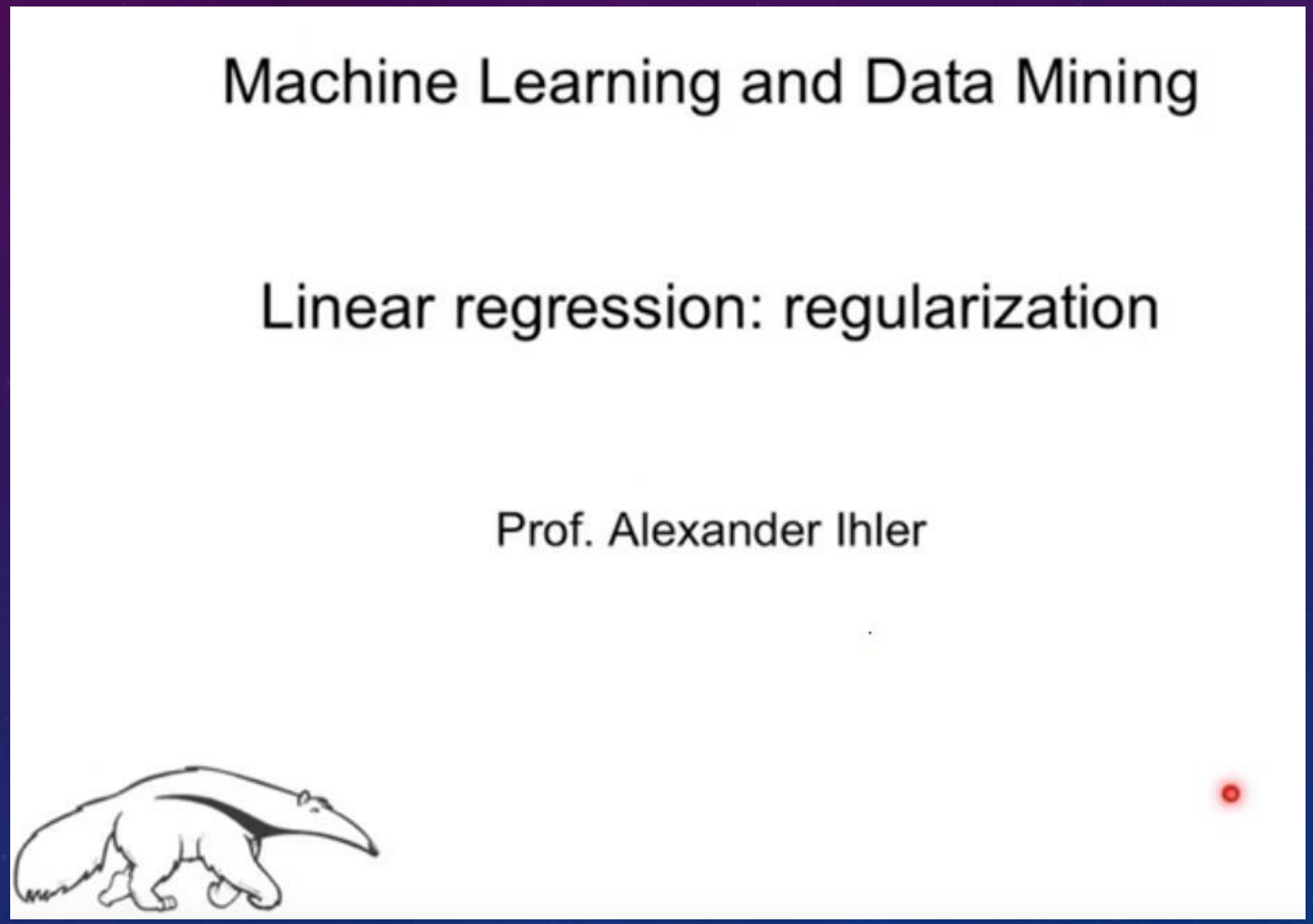
L_2 Regularization

- In L_2 regularization, regularization term is the sum of square of all feature weights as shown in the equation.
- L_2 regularization forces the weights to be small but does not make them zero and does non sparse solution.
- L_2 is not robust to outliers as square terms blows up the error differences of the outliers and the regularization term tries to fix it by penalizing the weights

$$L(x, y) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

L_2 regularization

Linear Regression : Regularization



https://www.youtube.com/watch?v=sO4ZirJh9ds [8:29]