한국소음진동공학회 AI혁신위원회 주관
2022년 인공지능 강습회

AI 기계이상진단을 위한 인공지능 학습 기법

제 4강 분류 태스크를 이용한 이상진단

한국과학기술원 전기및전자공학부

최정우
jwoo@kaist.ac.kr

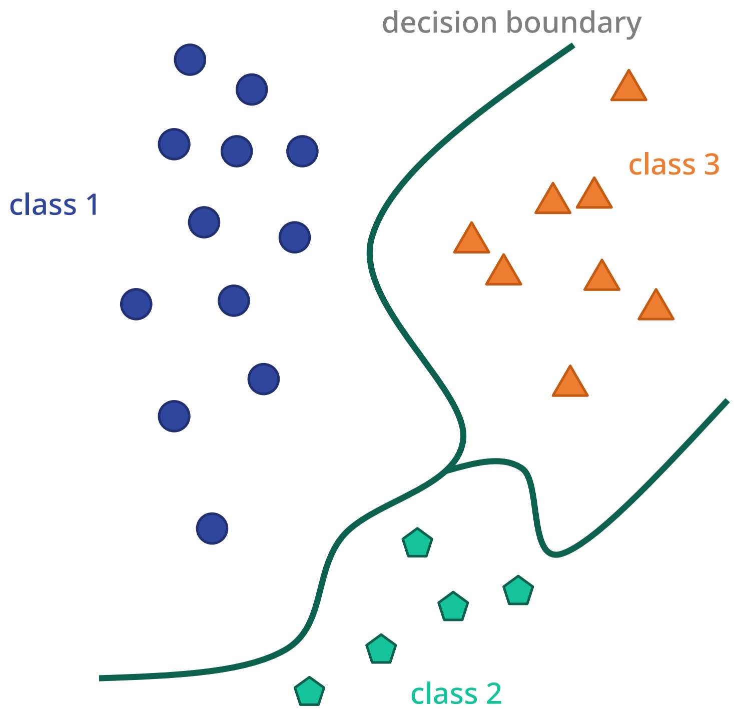
KAIST EE

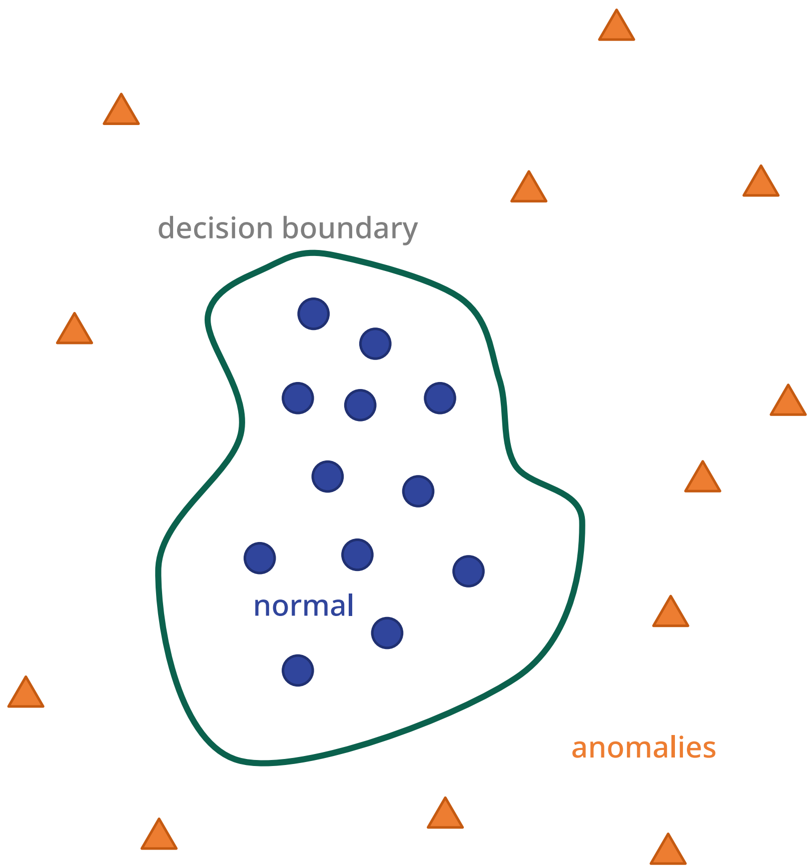
목차

- 단일 클래스 분류 문제
- 심층 신경망을 사용한 분류 기법
- 심층 신경망과 분류 기법을 사용한 이상진단기 설계
- 심층 신경망 분류기 조정 기법
  - Label smoothing
  - Temperature scaling

단일 클래스 분류

- **이상진단과의 관계**

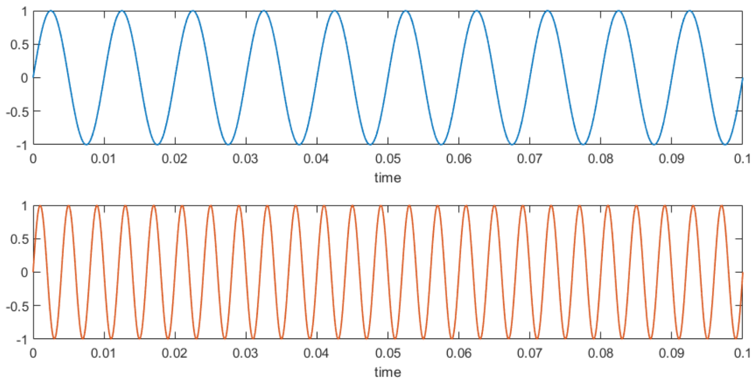




예제: 푸리에 변환

- Distinguishing two sinusoidal signals

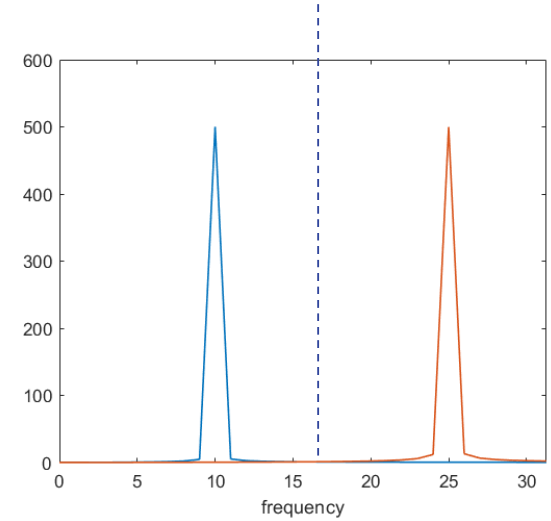
$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$



Input data

transform

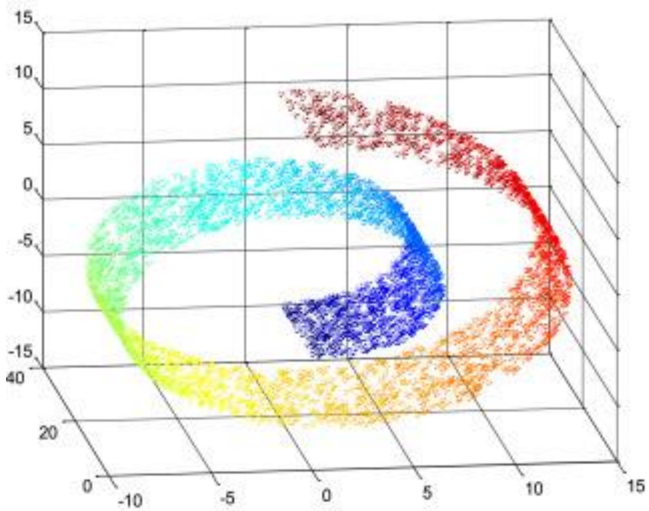
decision boundary



feature space

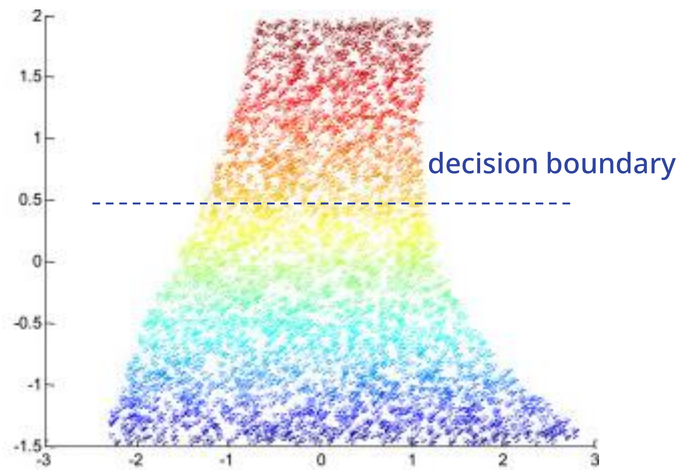
복잡한 차원 변환

- **Representation** learning



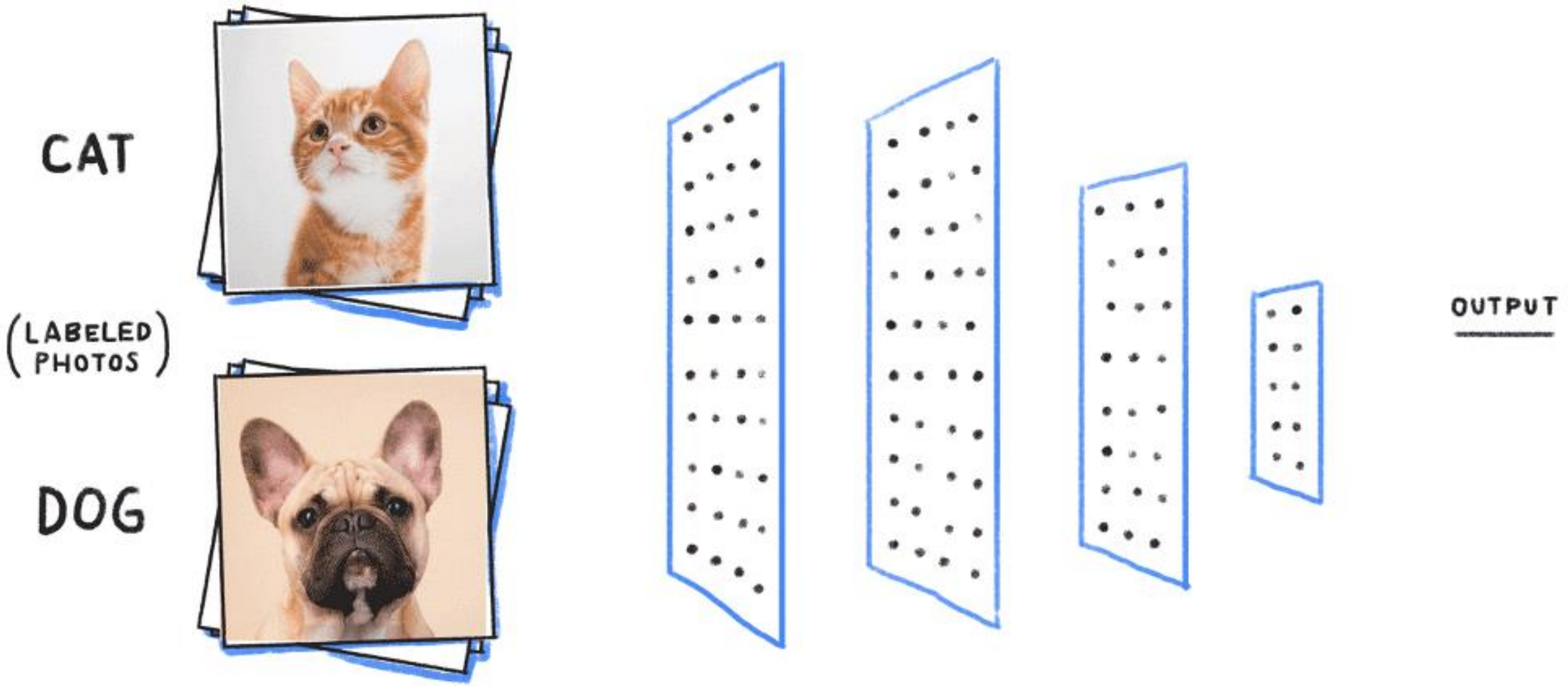
Input data

transform



feature space

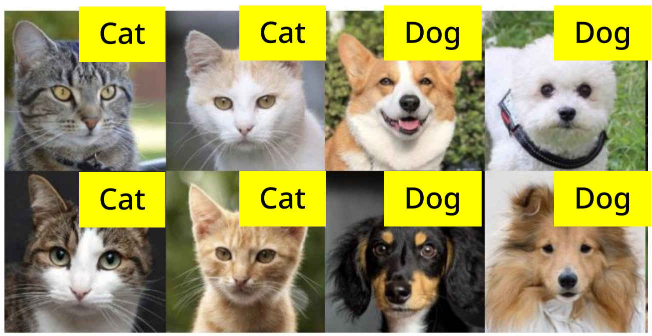
  - Good representation can be more important than finding a good decision boundary


CAT
(LABELED PHOTOS)
DOG
OUTPUT

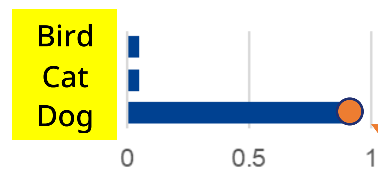
이상진단을 위한 분류 태스크 활용

- 확신(confidence) 기반 강아지와 고양이 데이터

Training dataset



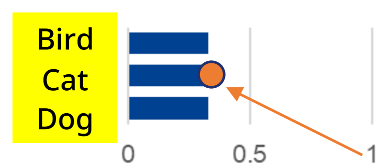
Softmax probability



Maximum Softmax probability
(~1 if model is confident)

Test dataset

Softmax probability



Maximum Softmax probability
(~1/C if the model is less confident)

$$\text{Anomaly Score} = \text{1- Maximum Softmax probability}$$

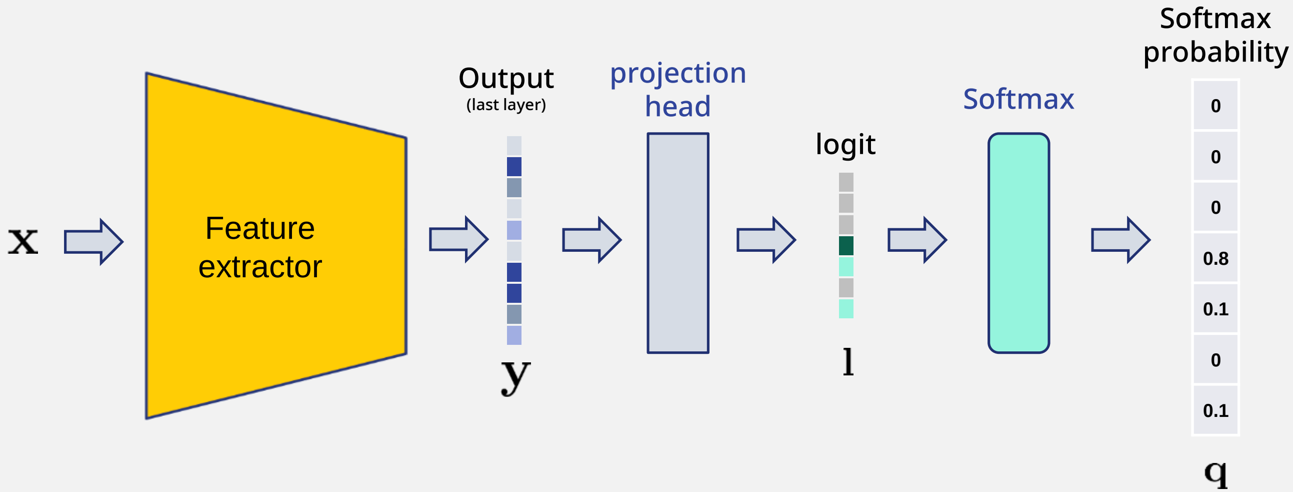
확신 기반 이상진단

- Model’s confidence = Anomaly detection score
  - Confidence should be similar to the model’s actual performance
  - Calibration of confidence is required

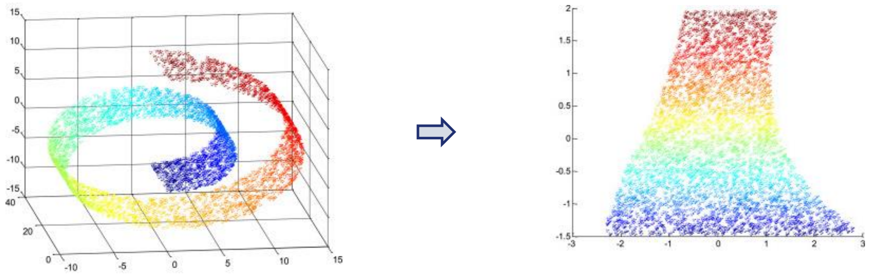
- In-depth study of model’s confidence
  - Logits
  - Softmax
  - Cross-entropy and KL divergence

- Tips & Tricks
  - Temperature scaling
  - Label smoothing

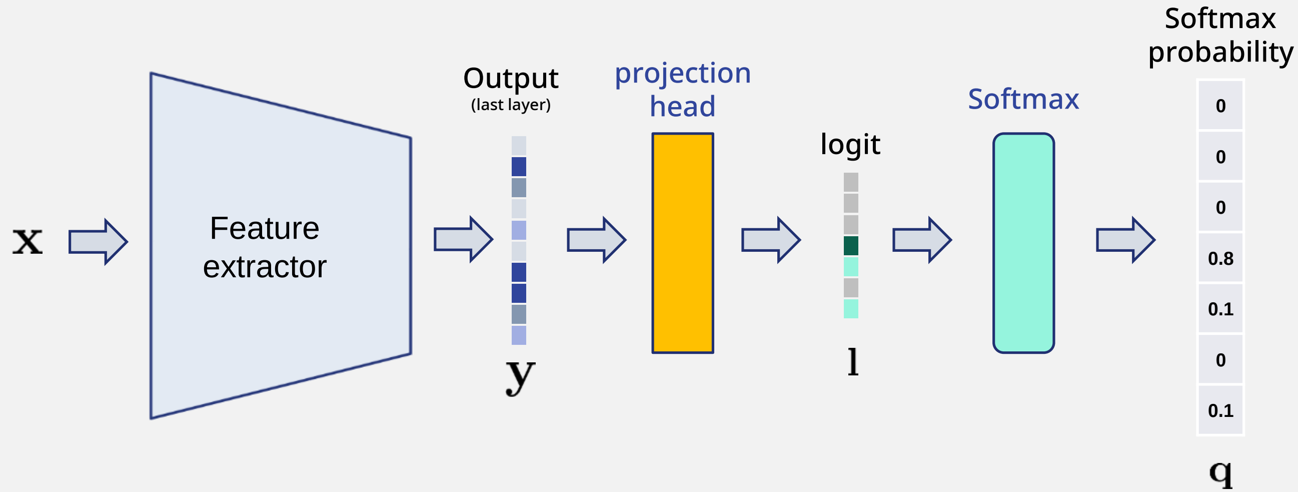
DNN 분류기의 구조



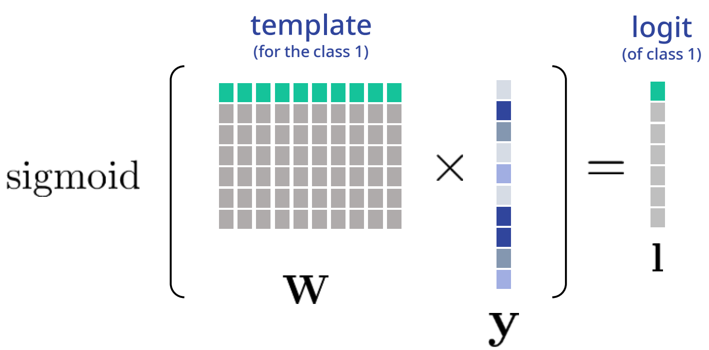
- **특징 추출기 (Feature extractor)**
  - Transform input data **x** to feature space data **y**
  - Trained to find a good mapping for classification



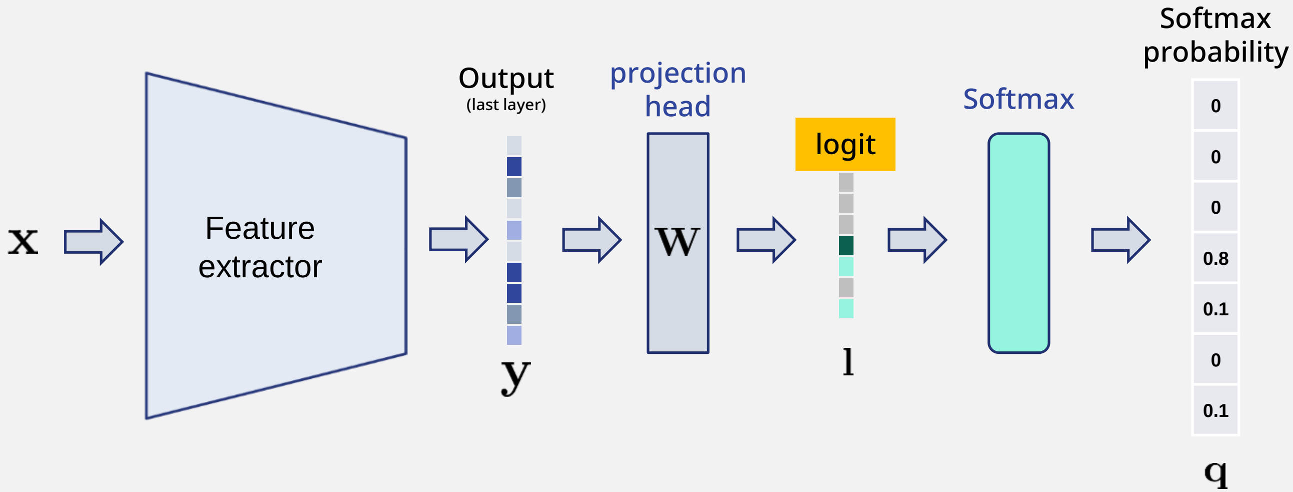
DNN 분류기의 구조



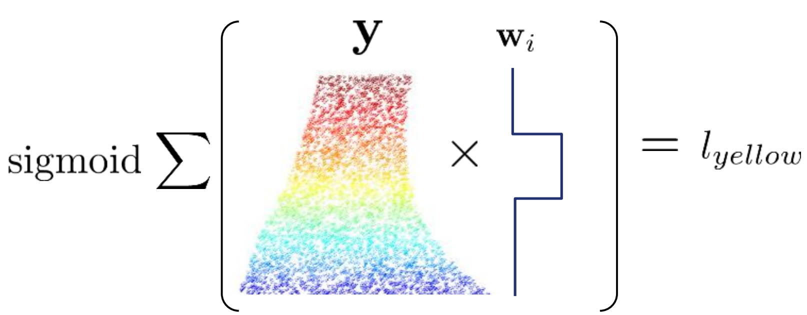
- **투영기 (Projection head)**
  - Compare output from the last layer with a template trained for each class
  - Usually fully connected network
  - Module for finding a decision boundary

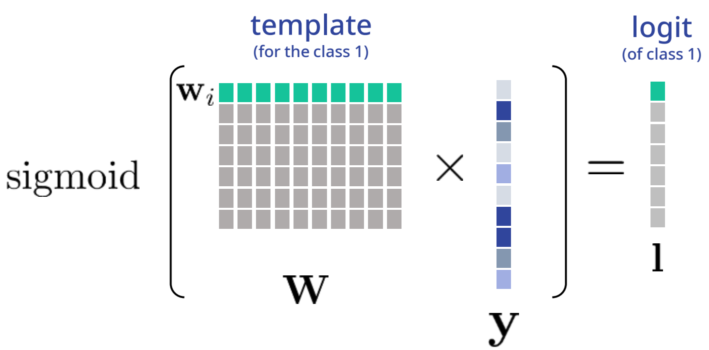


DNN 분류기의 구조

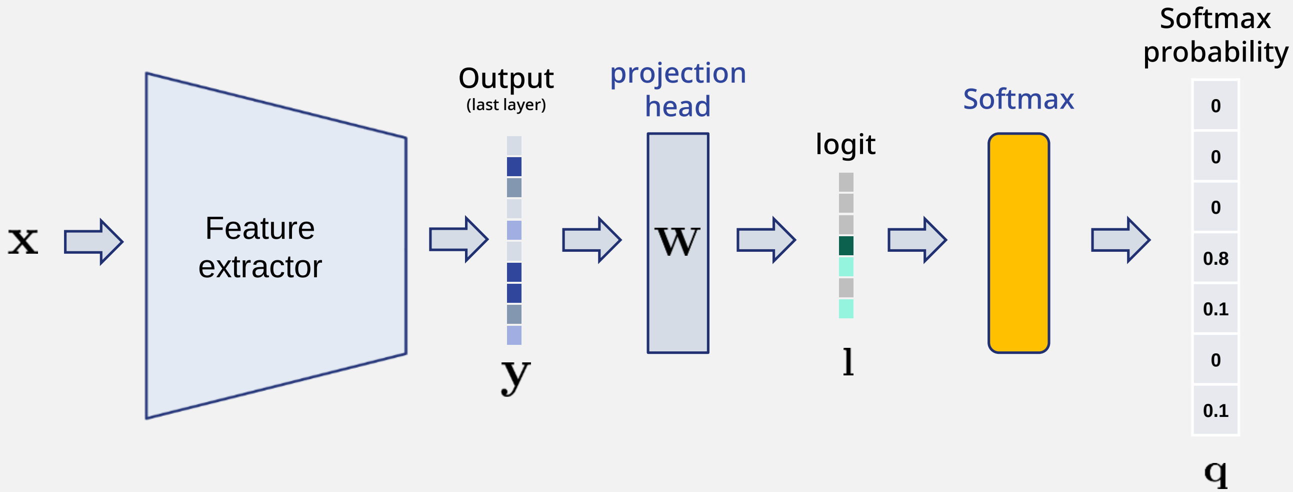


- logit (logistic probit)
  - describes how it is likely to belong to each class
  - not a probability yet



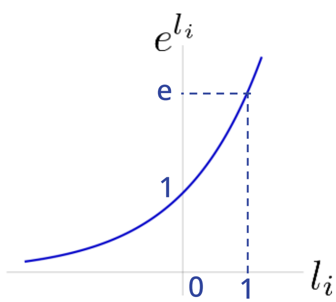


DNN 분류기의 구조



- **Softmax function**
  - Changes logit into a probability
  - Sum of all elements = 1

$$\int q(x)dx = 1$$



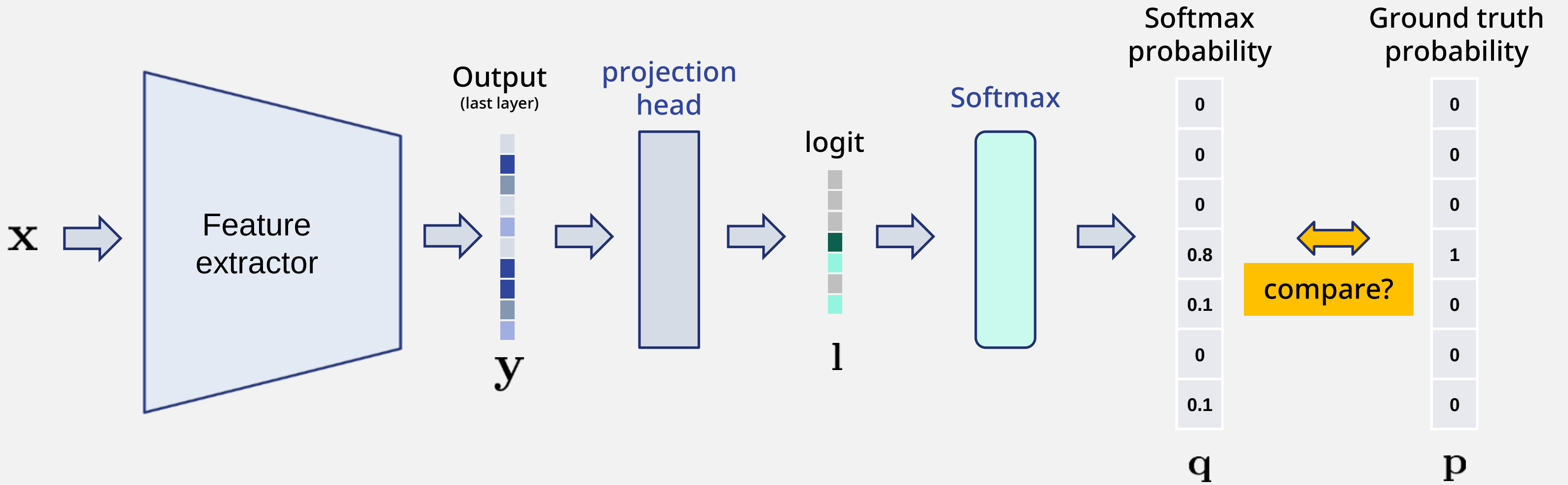
$$\mathbf{q} = \mathrm{softmax}(\mathbf{l})$$

$$q_i = \frac{e^{l_i}}{\sum_{k=1}^{C} e^{l_k}}$$

for two classes

$$= \frac{e^{l_1}}{e^{l_1} + e^{l_2}} = \mathrm{sigmoid}$$

DNN 훈련을 위한 손실 함수



- **분포간의 유사성 척도**
  - Kullback-Leibler (KL) divergence

$$KL(\mathbf{p}||\mathbf{q}) = \sum_{i=1}^{C} p_i \log \frac{p_i}{q_i}$$

  - KLD >= 0
  - asymmetric: $KL(\mathbf{p}||\mathbf{q}) \neq KL(\mathbf{q}||\mathbf{p})$

Kullback-Leibler 발산

- Derivative

$$\mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

one-hot vector

$$KL(\mathbf{p}||\mathbf{q}) = \sum_{i=1}^{C} p_i \log \frac{p_i}{q_i}$$
$$= \underbrace{\sum_{i=1}^{C} p_i \log p_i}_{=0} - \underbrace{\sum_{i=1}^{C} p_i \log q_i}_{\text{cross-entropy}}$$

- For one-hot target labels, KLD = cross-entropy
- Even if **p** is not a one-hot vector, **p** is a static variable. So, derivative of KLD = derivative of cross-entropy
- Cross-entropy loss is used instead of KLD

심층 신경망 조정하기

Tuning of DNN Classifiers

Cross-entropy loss 의 단점

- Backprop을 위한 미분시 문제점 (for one−hot target labels)

$$CE(\mathbf{q}) = -\sum_{i=1}^{C} p_i \log q_i$$

$\mathbf{p}$	$\mathbf{q}$	$\mathbf{q}'$
0	0	0
0	0	0
0	0	0
1	0.8	0.8
0	0.1	0.2
0	0	0
0	0.1	0

$$\frac{\partial CE(\mathbf{q})}{\partial q_j} = -\frac{\partial \left(p_j \log q_j + \sum_{i \neq j} p_i \log q_i \right)}{\partial q_j}$$

$$\text{if } p_j = 1, \; \frac{\partial CE(\mathbf{q})}{\partial q_j} = -\frac{1}{q_j} \qquad \text{if } p_j = 0, \; \frac{\partial CE(\mathbf{q})}{\partial q_j} = 0$$

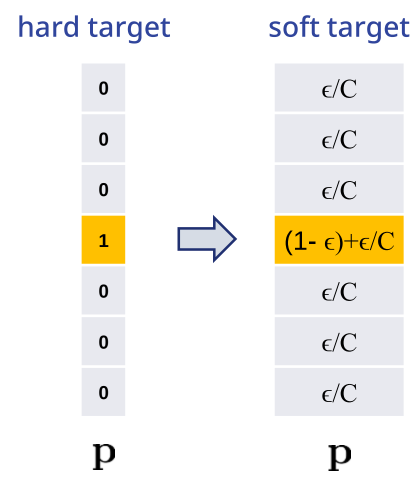
- Ground truth label이 1인 class에 대해서만 미분치가 존재
- 다른 레이블 들에 대해서는 gradient 발생하지 않음 (no backpropagation)

- CE loss only tries to maximize Softmax probability of the **correct label**

Label smoothing 기법

- 미분 (for one−hot target labels)

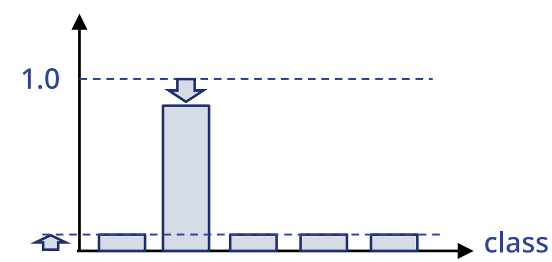
$$\frac{\partial CE(\mathbf{q})}{\partial q_j} = -\frac{\partial \left(p_j \log q_j + \sum_{i \neq j} p_i \log q_i \right)}{\partial q_j}$$



- Label smoothing
  - Adds a small noise to target probability
  - generates gradients for other labels

$$\text{if } p_j = \epsilon, \ \frac{\partial CE(\mathbf{q})}{\partial q_j} = -\epsilon \frac{1}{q_j}$$

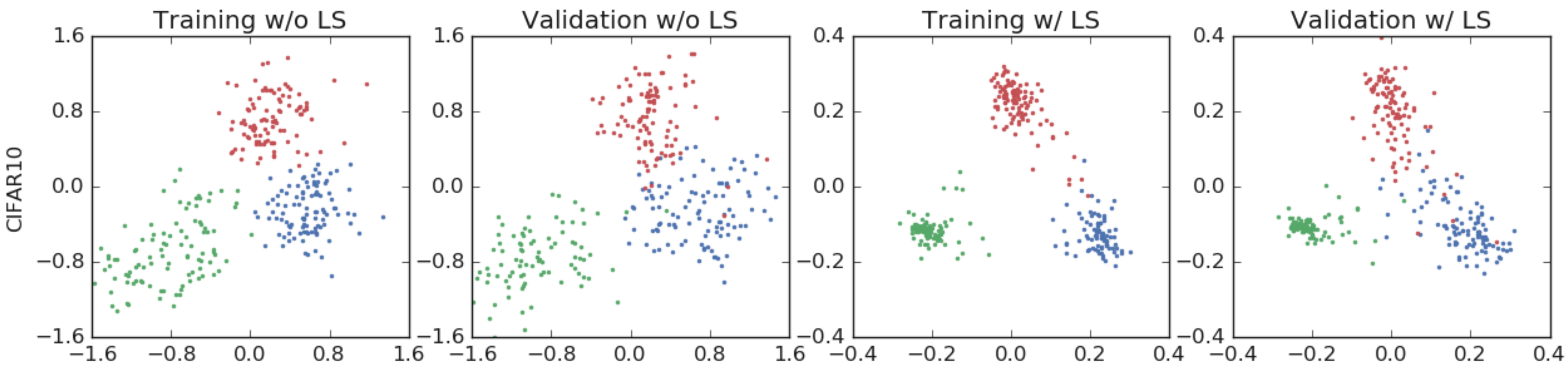
  - Label smoothing can increase the distance to other classes



https://arxiv.org/pdf/1906.02629.pdf

Label smoothing 기법

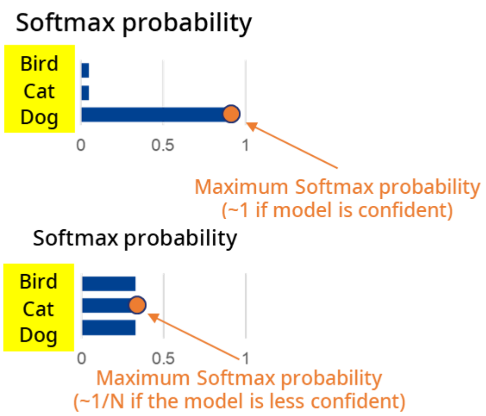
- **예제**



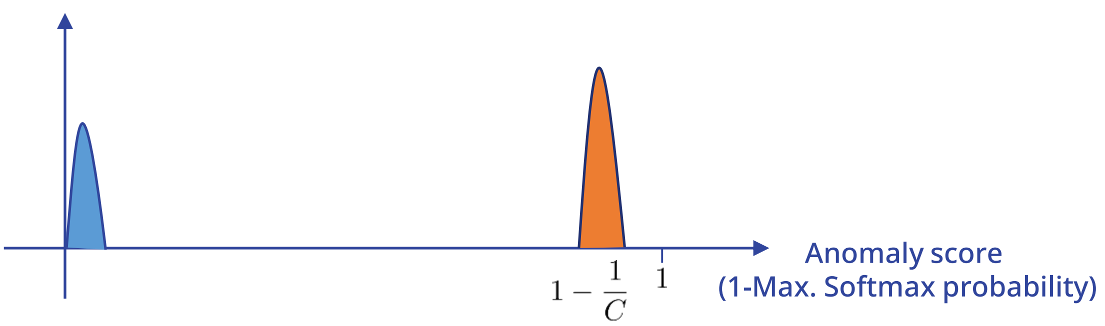
- **Label smoothing 효과**
  - 동일한 클래스의 데이터를 보다 좁은 영역으로 바인딩
  - 다른 클래스 데이터끼리 특징 차원에서의 거리를 더 분리

Temperature scaling

- **확신 (confidence) 기반 이상진단의 문제**
  - Maximum Softmax probability = Confidence of model



- **모델의 과신 (overconfidence) 문제**
  - The confidence of a model is often overrated.
  - Calibration of Softmax probability according to true correctness likelihood

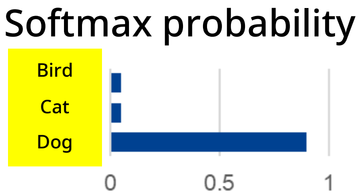


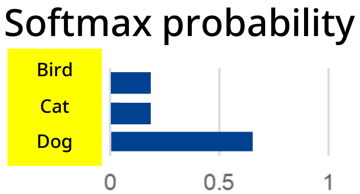
On Calibration of Modern Neural Networks, https://arxiv.org/pdf/1706.04599.pdf

Temperature scaling

- 아이디어
  - Rescale the logit scores before applying Softmax
  - Calibration without affecting the classification result over normal data

$$q_i = \frac{e^{(l_i/T)}}{\sum_{k=1}^{C} e^{(l_k/T)}}$$



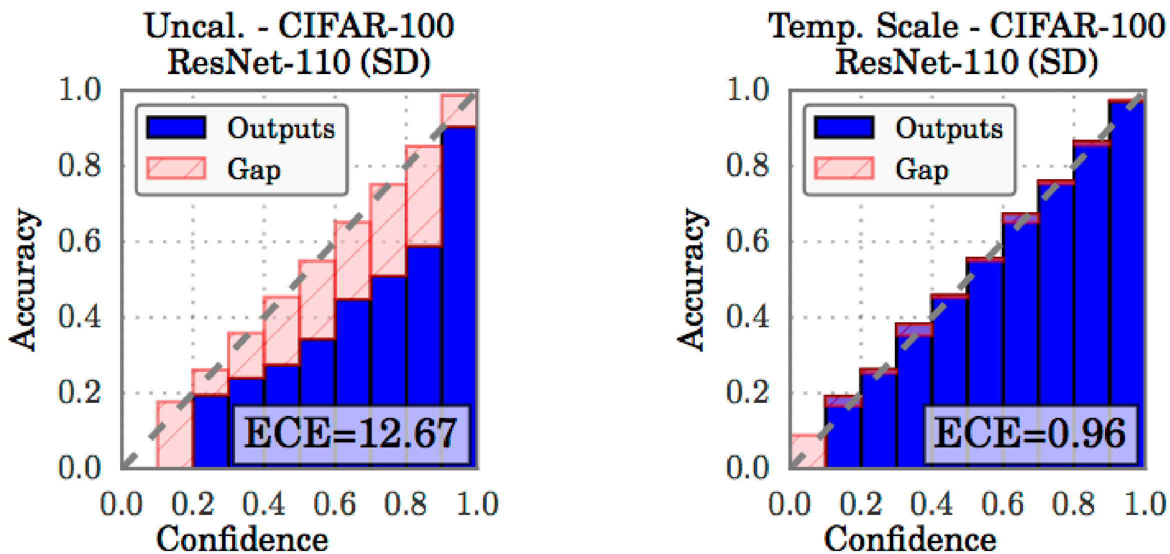


- Temperature *T*
  - Softens the Softmax (increases output entropy) with T>1
  - As T→∞, q_i approaches to 1/K

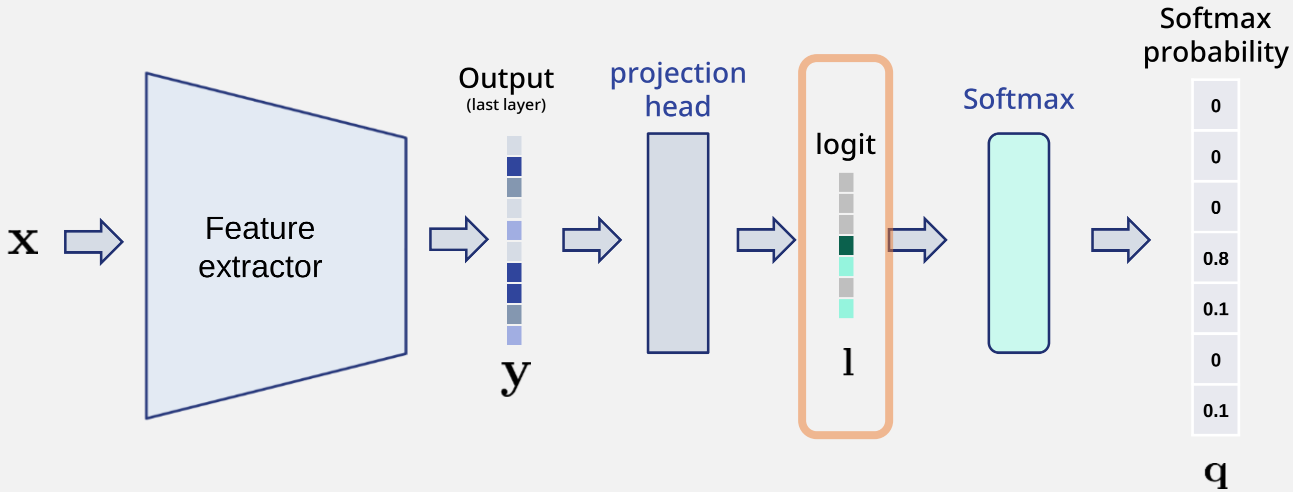
On Calibration of Modern Neural Networks, https://arxiv.org/pdf/1706.04599.pdf

Temperature Scaling을 사용한 학습된 모델 보정

- Using the validation set
  - Accuracy vs. Confidence

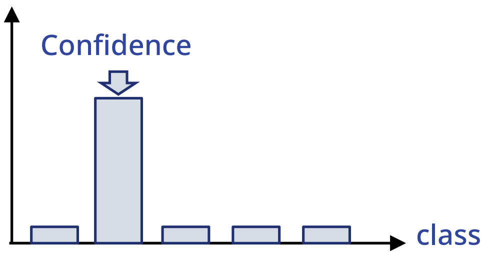


Free-energy 기반 이상진단



- Softmax function & Confidence

$$q_i = \frac{e^{l_i}}{\sum_{k=1}^{C} e^{l_k}}$$



- Free energy

$$E_F = -\log \sum_{k=1}^{C} e^{l_k}$$

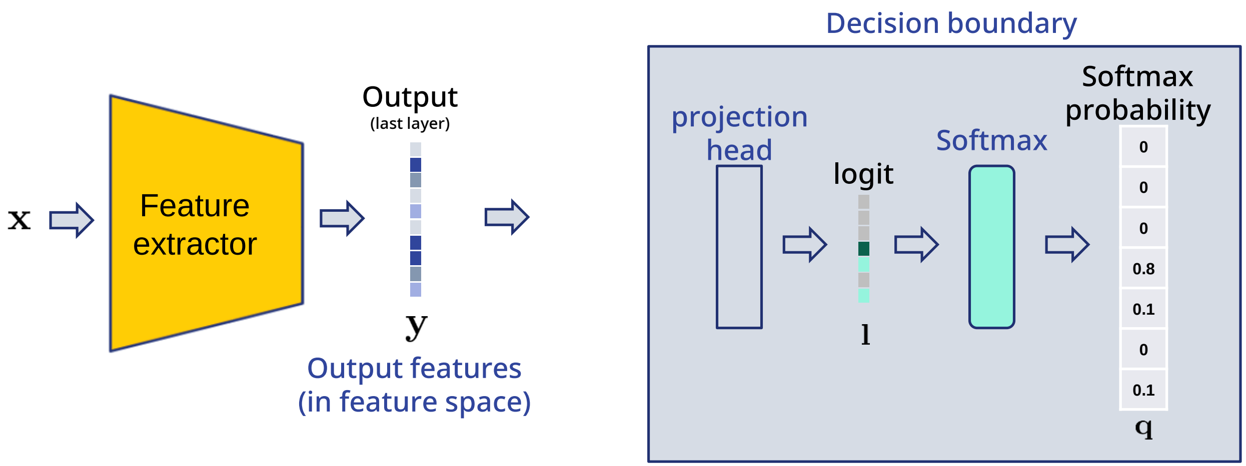
- Overall activation of logits
- Using $-E_F$ for detecting outliers
  → Low overall activation

특징 추출

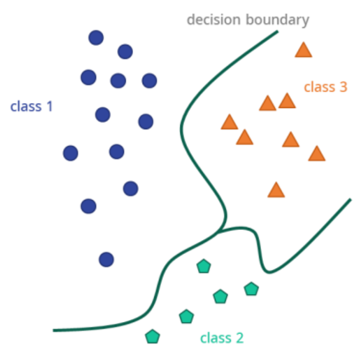
Good transformation?

좋은 분류기란?

- 좋은 representation이 좋은 decision boundary보다 중요
  - 어떻게 feature space 상에서 잘 분리되어 있도록 할까?

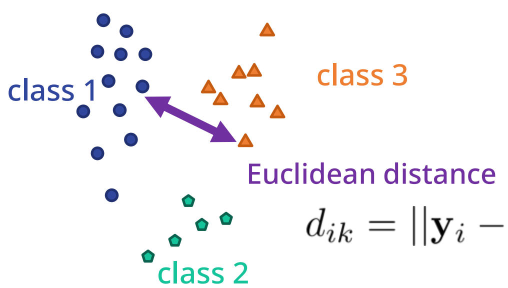


- 좋은 representation?
  - 유사한 레이블들은 가깝게
  - 다른 레이블들은 멀게

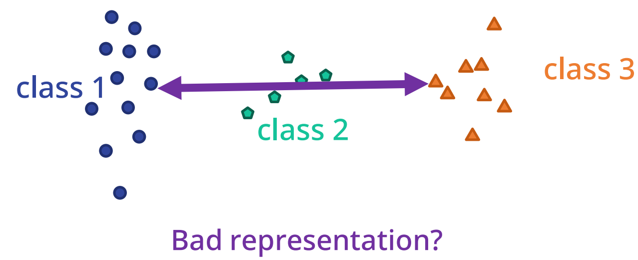


다른 손실 함수(loss fn.)을 사용한 훈련법

- **거리 기반 손실함수**
  - 반드시 먼 거리가 좋은 분류를 뜻하지는 않음



$$d_{ik} = ||\mathbf{y}_i - \mathbf{y}_k||$$

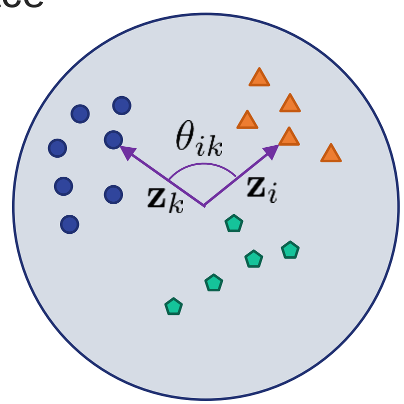


- **각도 기반 손실함수**
  - Normalize: map the output feature onto the spherical surface

$$\mathbf{z}_i = \frac{\mathbf{y}_i}{||\mathbf{y}_i||} \quad \rightarrow ||\mathbf{z}_i|| = 1$$

  - Measure the cosine angle between normalized features

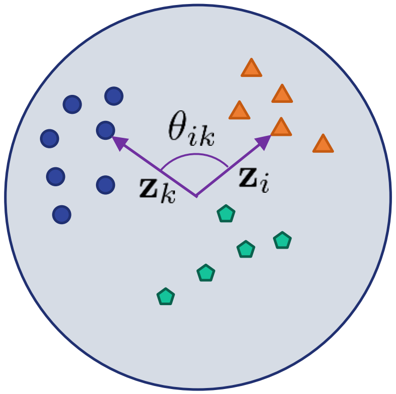
$$\cos\theta_{ik} = \mathbf{z}_i^T \mathbf{z}_k$$

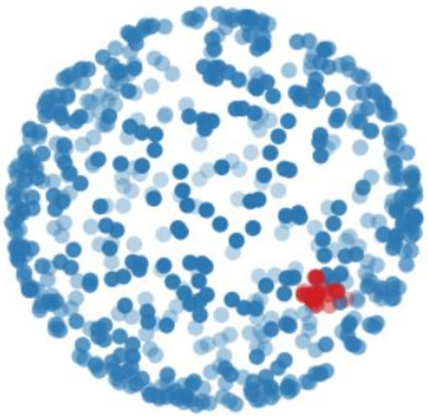


각도 기반 손실 함수 (Angular loss)

- **Cosine similarity loss**
  - Cosine: 1 for similar data, -1 for dissimilar data
  - For gradient descent minimization, we use the following loss function

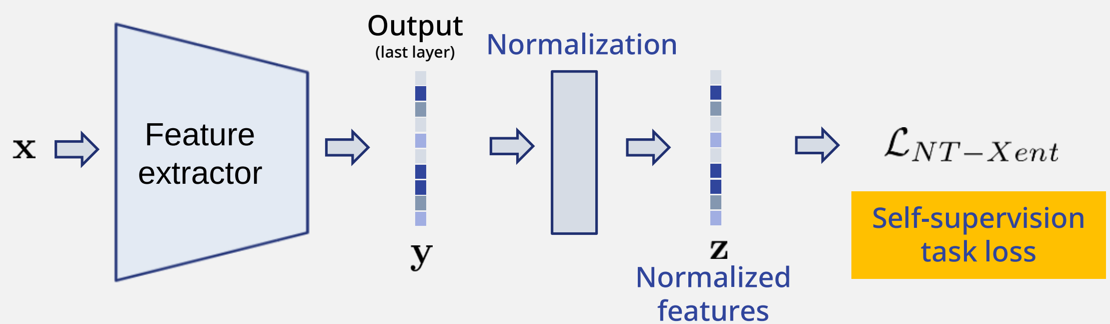
$$\mathcal{L}_{ang} = 1 - \cos\theta_{ik} \in [0, 2]$$

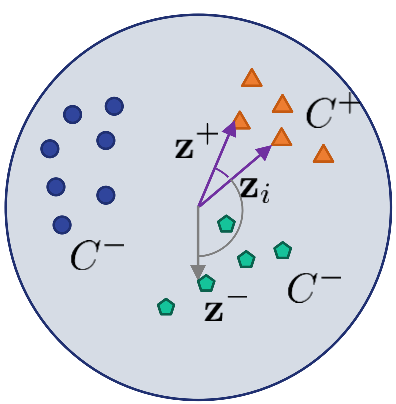




NT-Xent loss

- Contrastive 훈련을 위한 동종 및 이종 레이블 손실 함수
  - 같은 레이블은 가깝게 (minimize angles)
  - 다른 레이블은 멀게 (maximize angles)



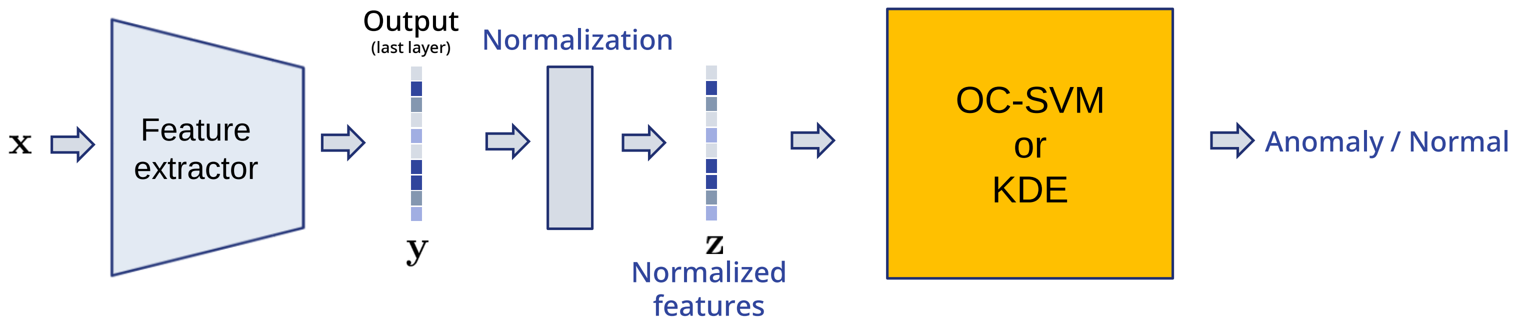


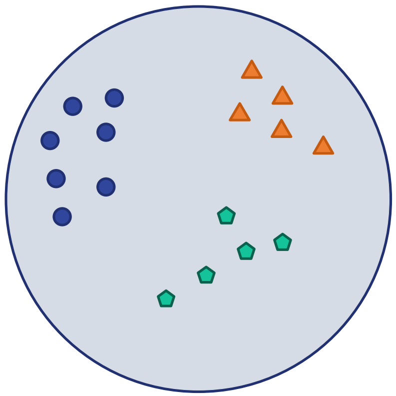
$$\mathcal{L}_{NT-Xent} = -\sum_{\mathbf{z}^+ \in C^+} \log \frac{e^{(\mathbf{z}_i^T \mathbf{z}^+ / T)}}{\sum_{k=1}^{C} e^{(\mathbf{z}_i^T \mathbf{z}^+ / T)} + \sum_{\mathbf{z}^- \in C^-} e^{(\mathbf{z}_i^T \mathbf{z}^-)/T}}$$

c.f.) $$q_i = \frac{e^{(l_i/T)}}{\sum_{k=1}^{C} e^{(l_k/T)}}$$

Downstream Task

- **자기지도학습 후 downstream task를 통한 이상치 검출**
  - 이상진단점수 (Anomaly Score) 설정
  - 기존의 Kernel density estimation 이나 One-class Support Vector Machine (SVM) 사용



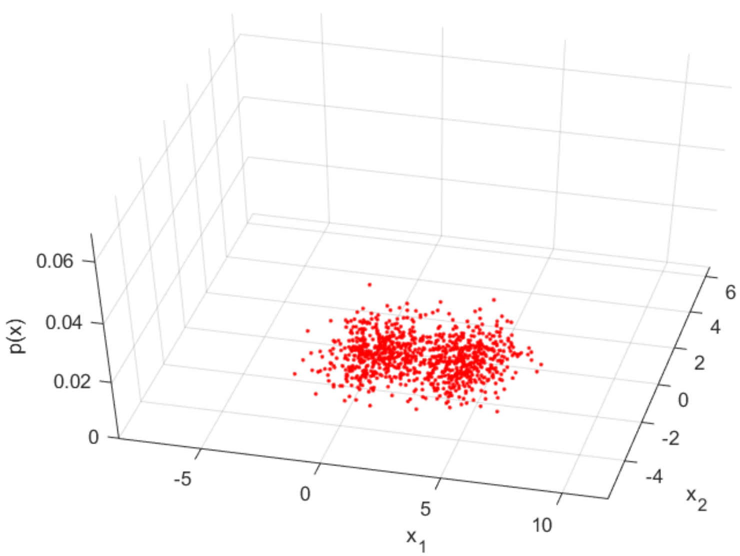


Kernel density estimation (KDE)

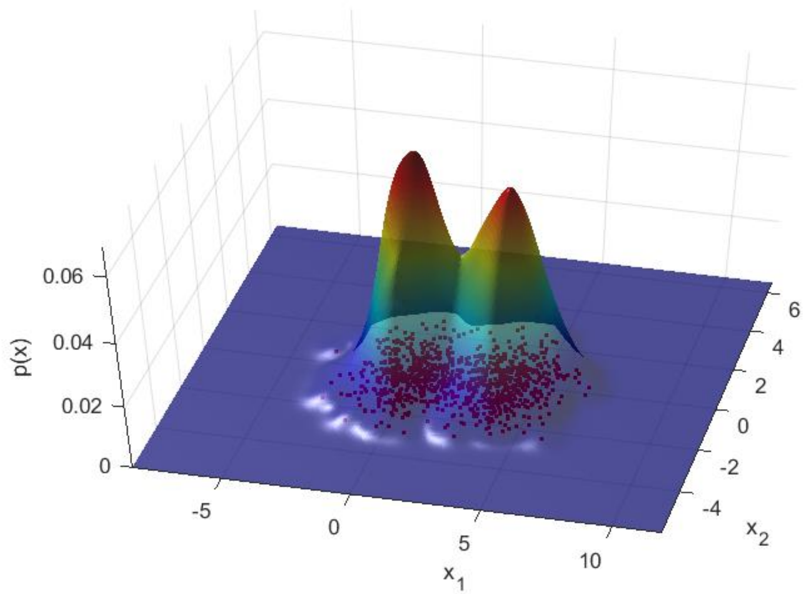
Classical classification techniques

Feature distribution → Density function

- 정상 데이터 분포 (Distribution of normal feature data)

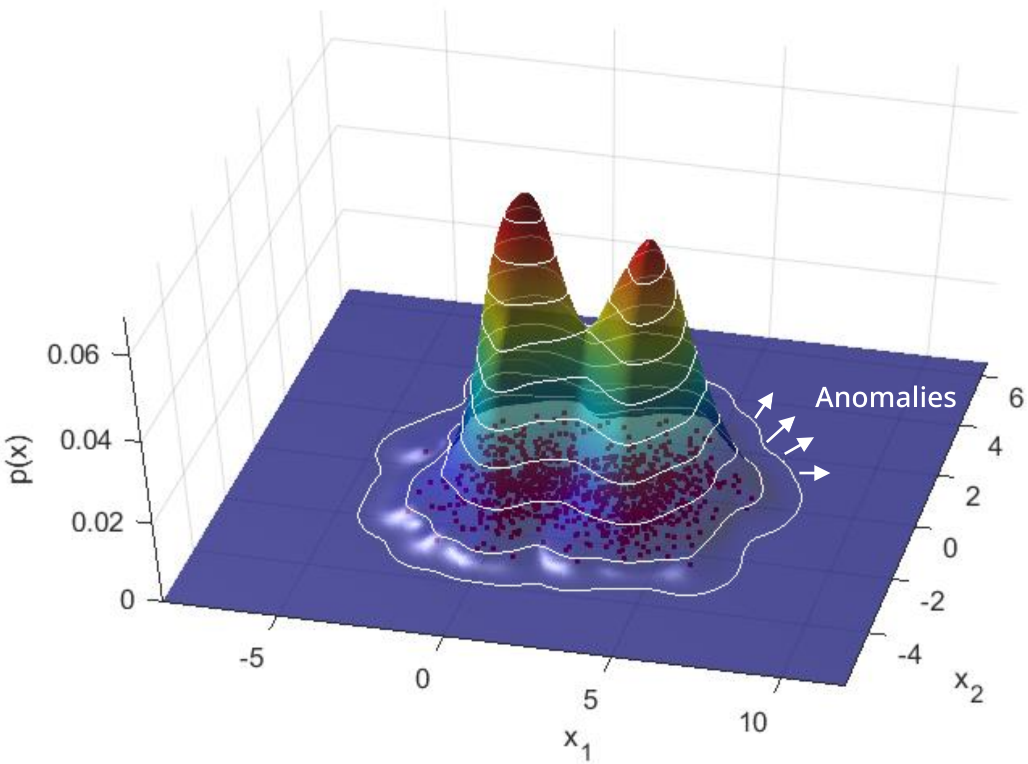


- 밀도 함수 (Density function)



이상 진단

- **밀도 = 이상 점수**
  - Data outside of a certain density level becomes anomalies
  - Level of density function = decision boundary

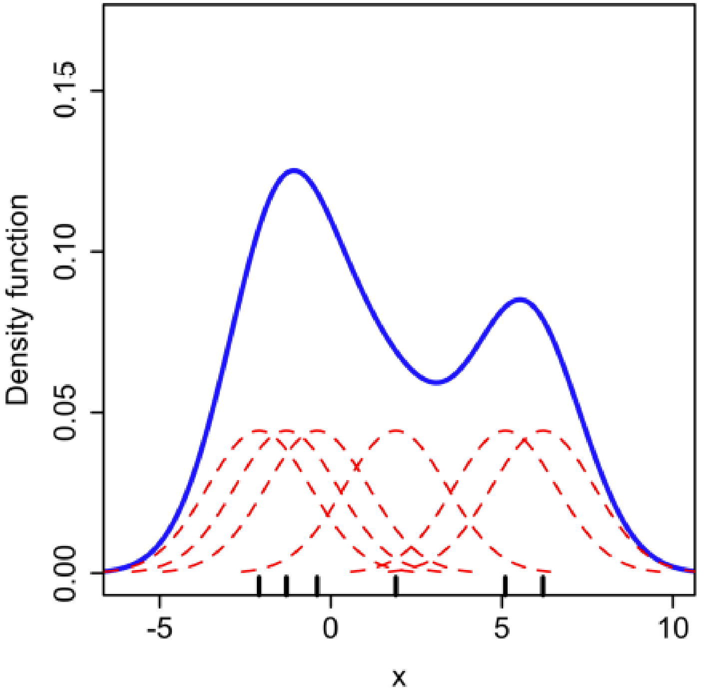


Kernel density estimation (KDE)

- Smoothing discrete observation using a kernel function K

$$f\left(x\right)=\frac{1}{nh}\sum_{i}K\left(\frac{x-x_i}{h}\right)$$

  - n: number of data
  - h: bandwidth (hyperparameter)
  - x_i: position of i^{th} feature data

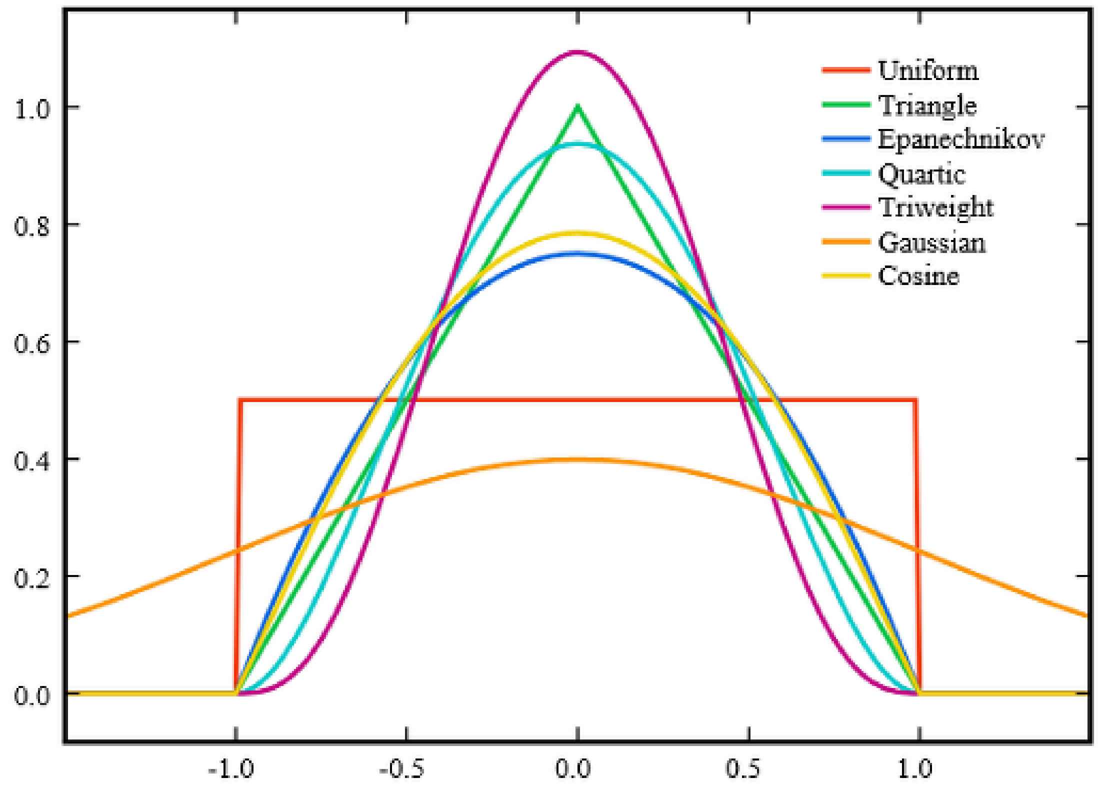


- For multivariate feature

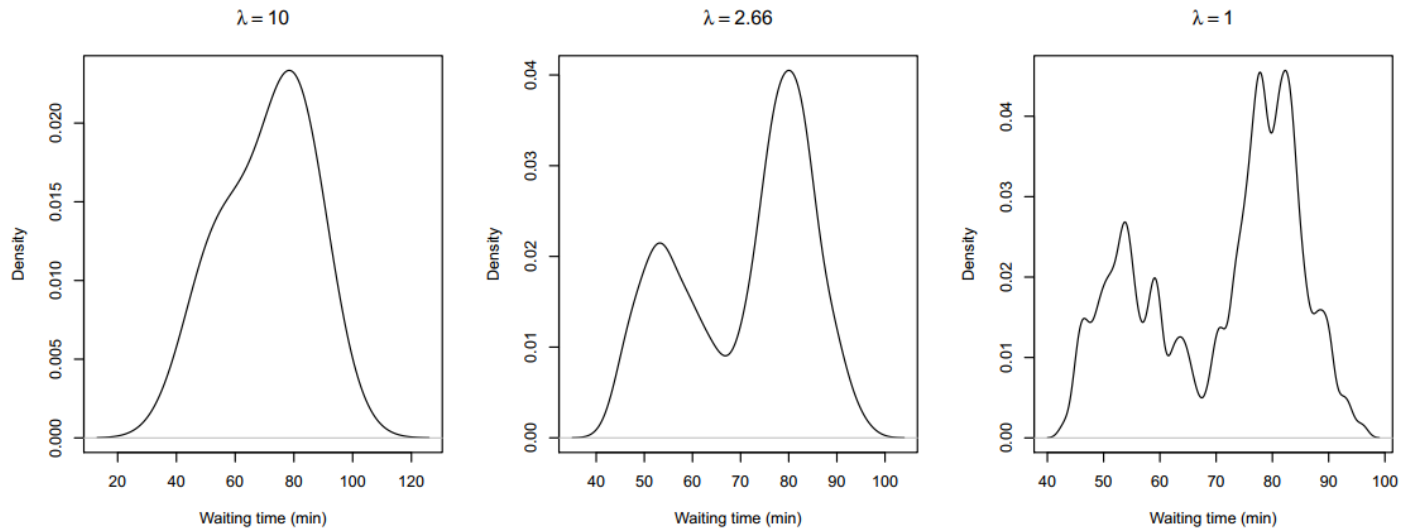
$$f\left(\mathbf{x}\right)=\frac{1}{n}|\mathbf{H}|^{-1/2}\sum_{i}K\left(|\mathbf{H}|^{-1/2}(\mathbf{x}-\mathbf{x}_i)\right)$$

Choice of kernel function

- Various kernel functions



Choice of bandwidth



- Higher bandwidth → reduces variance, increases bias
- Rule of thumb for Gaussian kernel

$$h = \left(\frac{4\sigma^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\sigma n^{-1/5} \qquad \mathbf{H} = n^{-1/(d+4)}\boldsymbol{\Sigma}^{1/2}$$

KDE in Python

- scikit-learn library

class sklearn.neighbors.KernelDensity(***, *bandwidth=1.0*, *algorithm='auto'*, *kernel='gaussian'*, *metric='euclidean'*, *atol=0*, *rtol=0*, *breadth_first=True*, *leaf_size=40*, *metric_params=None*) [source]

```
>>> from sklearn.neighbors import KernelDensity
>>> import numpy as np
>>> rng = np.random.RandomState(42)
>>> X = rng.random_sample((100, 3))
>>> kde = KernelDensity(kernel='gaussian', bandwidth=0.5).fit(X)
>>> log_density = kde.score_samples(X[:3])
>>> log_density
array([-1.52955942, -1.51462041, -1.60244657])
```

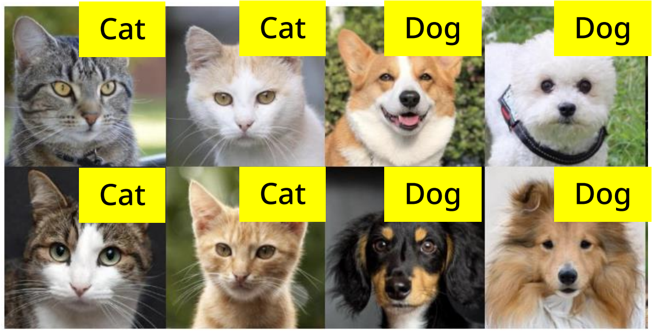
분류 태스크를 활용한 다양한 자기지도 학습기법

Transform-based techniques

클래스 레이블이 없는 경우의 자기지도 분류학습

- For classification task, internal labels are required

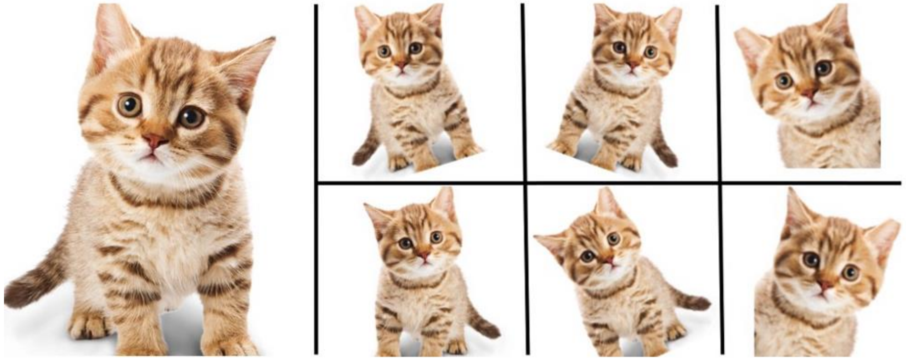
Training dataset



- What if we don't have such labels?
  - Prior information from humans can be utilized
  - 경험에 기반한 상식 → DNN model에 주입

어떤 상식을 활용할 수 있을까?

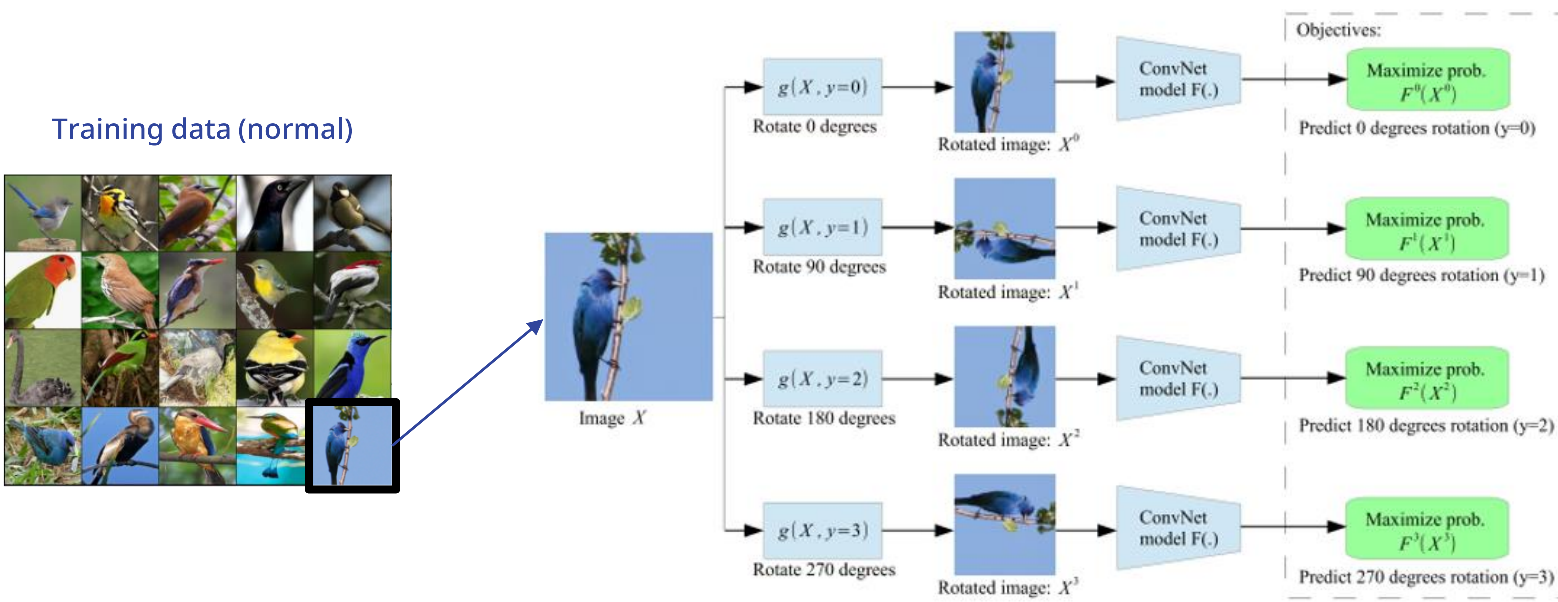
- Common senses
  - Some transforms do not change the identity of data
  - e.g., a cat image is still a cat image after the rotation, stretching, translation, zoom, cutout, brightness/ contrast/ color change …



- How to utilize these common senses in DNN training?

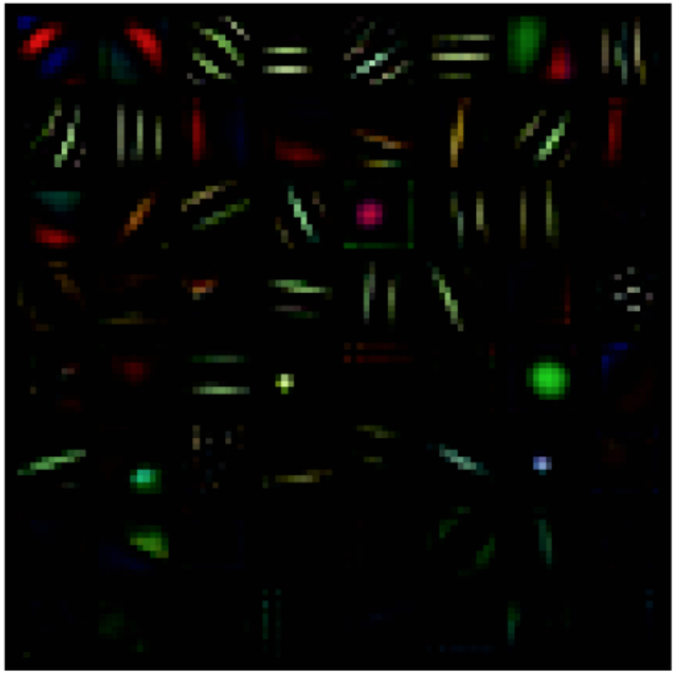
Let the model learn common senses

- **Using transform as classification labels**
- **Example: RotNet (ICLR 2018)**
  - Find the label of rotation angle from a given image

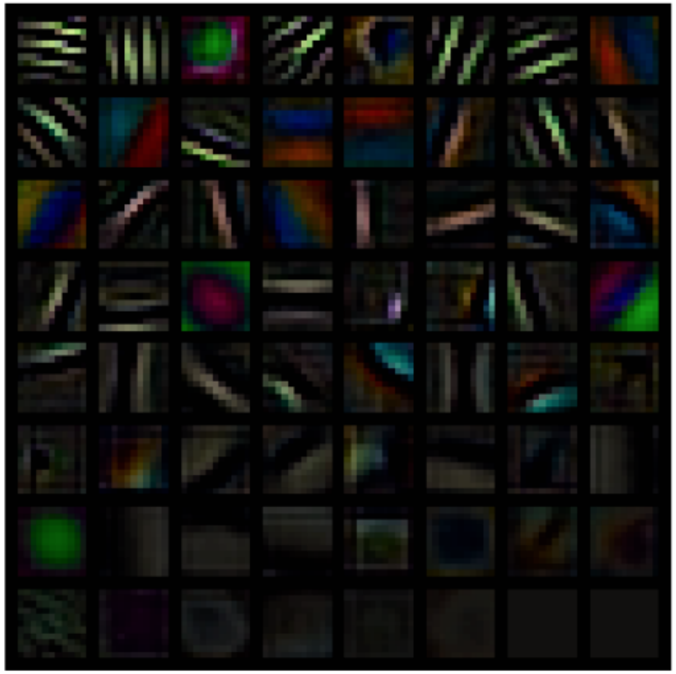


이미지 회전을 통한 자기지도 학습

- To find a right angle, the DNN model tries to learn various features
  - While the scale or resizing transform does not provide useful features for image classification

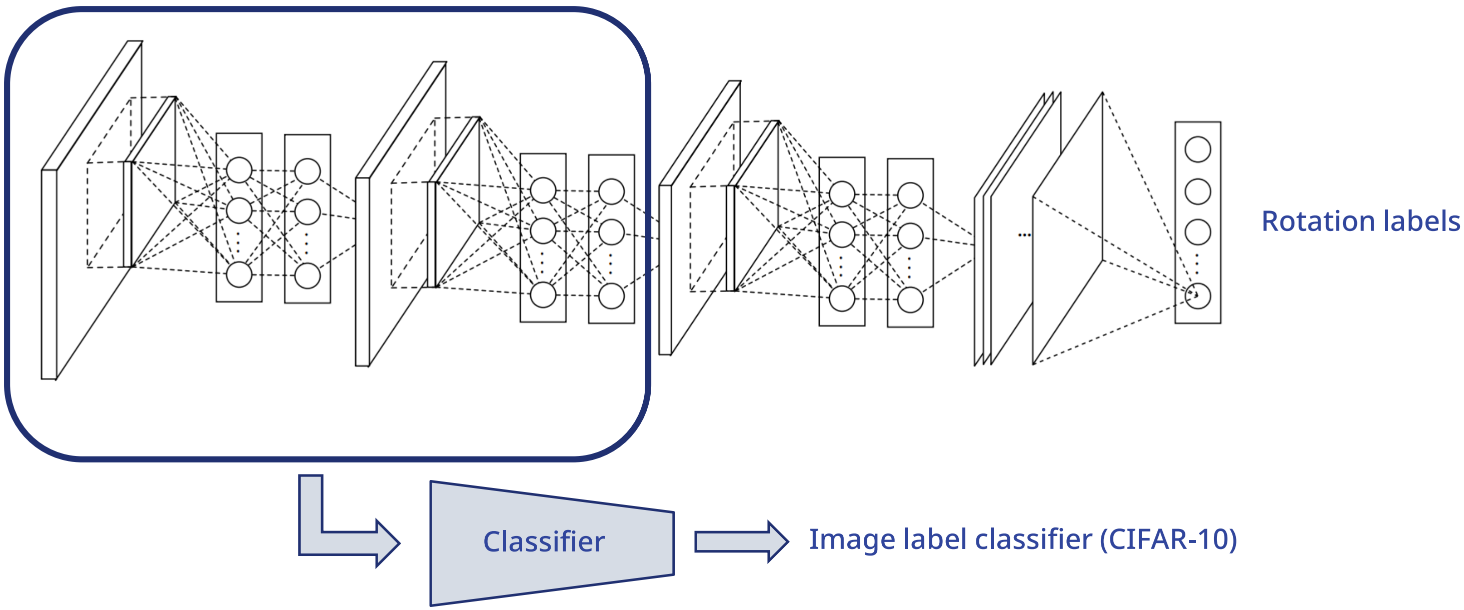


(a) **Supervised**



(b) **Self-supervised to recognize rotations**

Transfer learning with RotNet



Model	ConvB1	ConvB2	ConvB3	ConvB4	ConvB5
RotNet with 3 conv. blocks	85.45	88.26	62.09	-	-
RotNet with 4 conv. blocks	85.07	89.06	86.21	61.73	-
RotNet with 5 conv. blocks	85.04	**89.76**	86.82	74.50	50.37

분류 기반 자기지도 학습의 단점

- Class labels include only small information
  - Cat, dog, tiger … : high level context is missing
  - e.g.) cat eating a Churu (normal)
    cat eating dog food (abnormal)

Normal

Abnormal

  - 간단한 레이블로는 고차원의 학습 정보를 얻기 어려움

Contrastive Learning

- Classification

Cat

Dog

Tiger

- Contrastive learning
  - 남들과 비교를 통해 더 많은 정보 습득

Negative sample

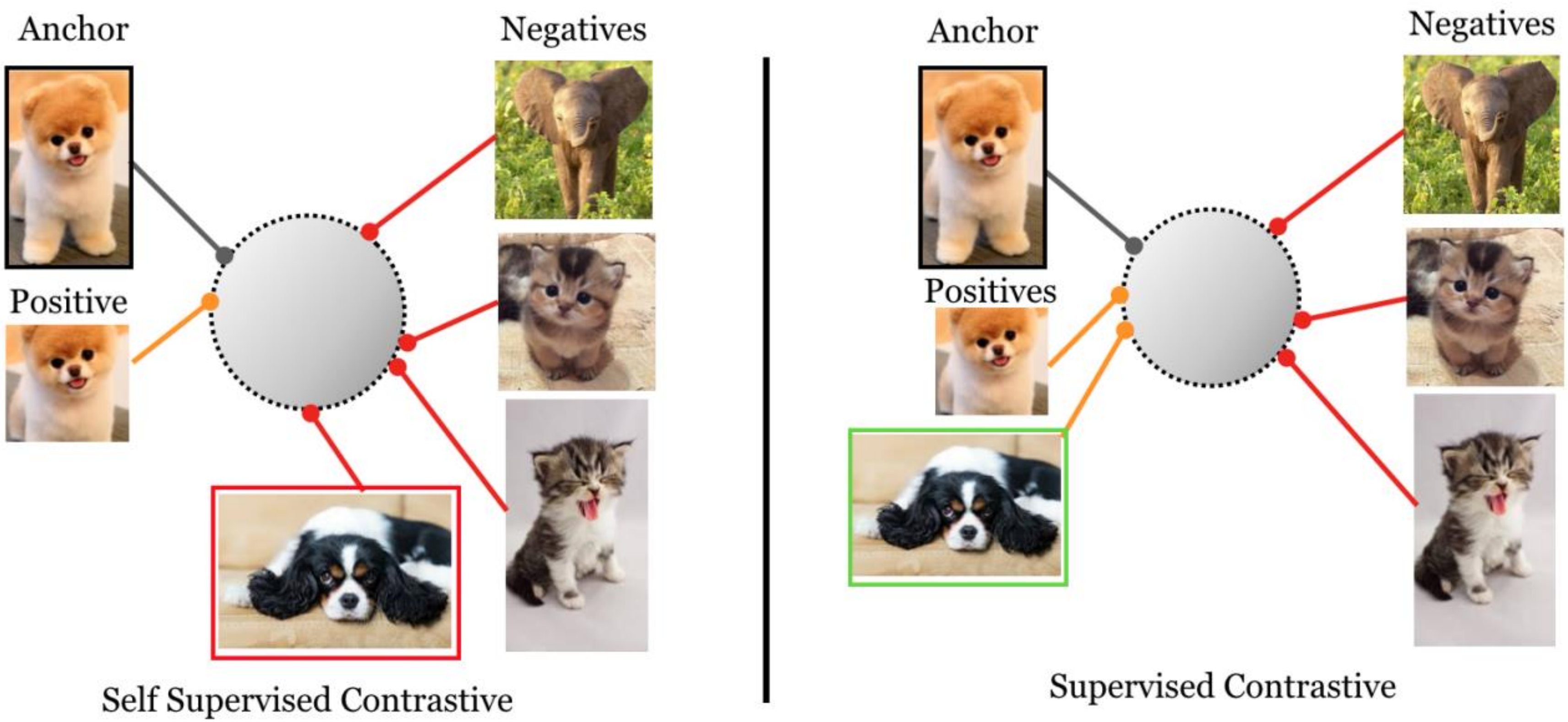
Different

Similar

Positive sample

Negative sample

Contrastive Learning



Supervised Contrastive Learning, NeurIPS, 2020, https://arxiv.org/pdf/2004.11362.pdf

요약

- **분류 태스크를 통한 이상 진단**
  - Using maximum Softmax probability as anomaly score
  - Maximum Softmax probability = Model's confidence
  - Confidence needs calibration: temperature scaling
- **좋은 분류기?**
  - Good representation + Good decision boundary
  - Feature extractor = representation learning
  - Projection head + Softmax = decision boundary
- **더 나은 특징 획득을 위한 기법들**
  - Label smoothing: considering non-target labels
  - NT-Xent loss: push/pull angular distances between inter- and intra-class data
- **레이블이 없는 경우의 분류 학습**
  - Using known transforms to generate labels

Appendix
개발 단계에서 특징 분포 확인

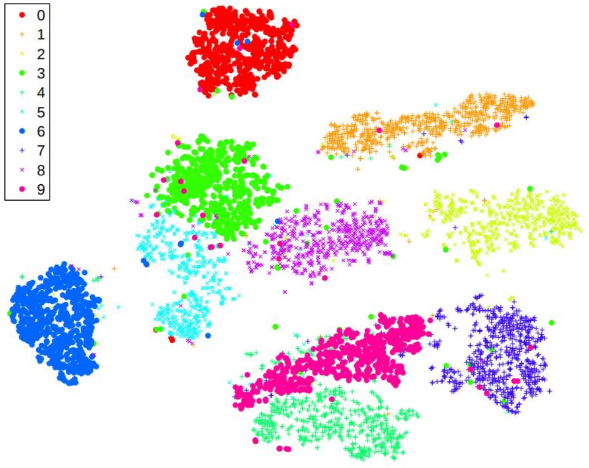
Feature Visualization

t-Stochastic Neighbor Embedding

- **Vector Visualization**
  - High dimensional data → 2D vector mapping

- **Objective**
  - Retaining both the local and the global structure of the data in a single map

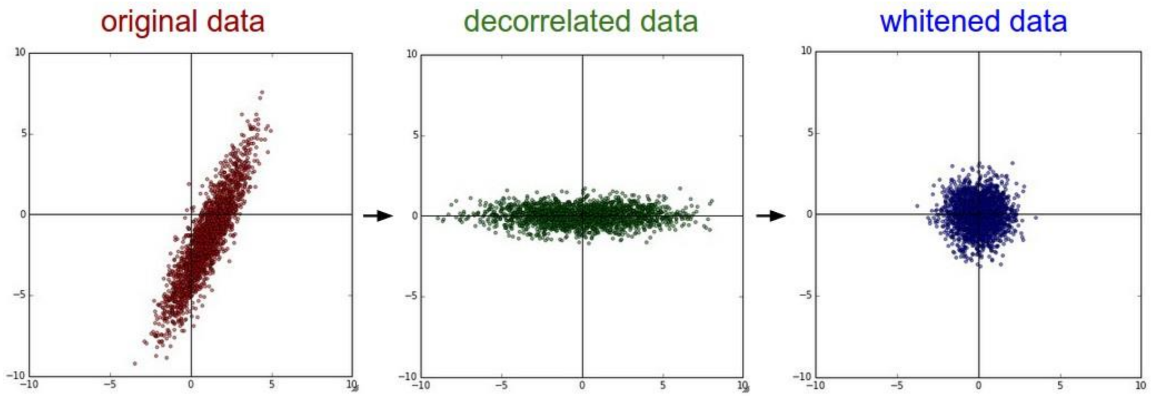
- **Example**
  - 2D visualization of MNIST dataset
  - Original data: 28x28 = 784 dim. → 2 dim.
  - 10 numerical digits (classes)



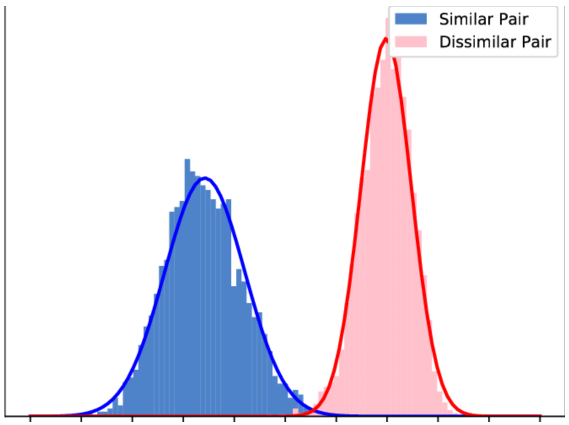
(a) Visualization by t-SNE.

Whitening for measuring distances

- Whitening



- Normalized distances

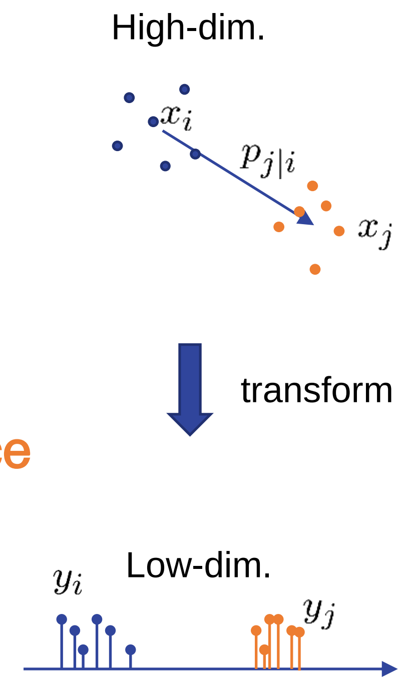


Similarity between vectors

- Similarity in (input) high dim. space
  - Softmax probability of Mahalanobis distance
  - Gaussian assumption

$$p_{j|i} = \frac{\exp\left(-|x_i - x_j|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-|x_i - x_k|^2 / 2\sigma_i^2\right)}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$



- Similarity in (feature embedding) low dim. space

t-distribution

$$q_{ij} = \frac{\boxed{\left(1 + |y_i - y_j|^2\right)^{-1}}}{\sum_{k \neq l} \left(1 + |y_k - y_l|^2\right)^{-1}}$$

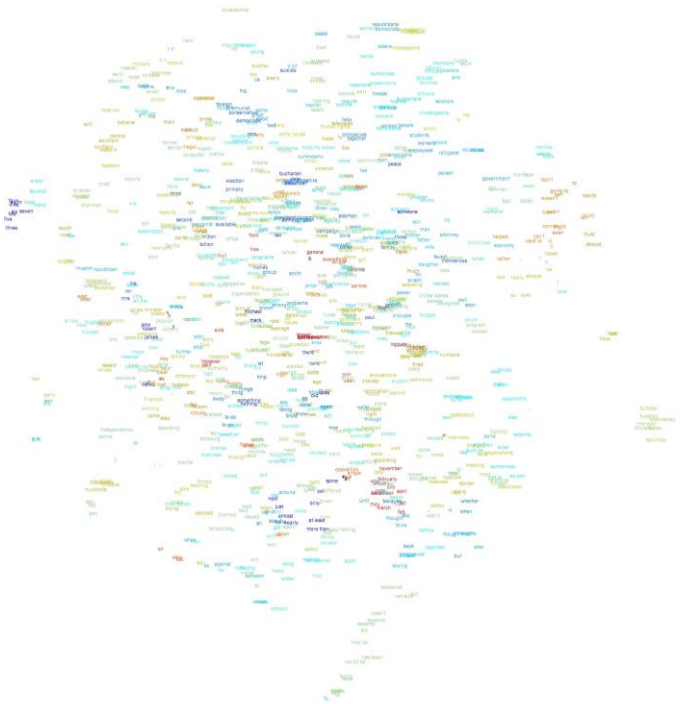
- Making two distributions similar

Optimization

- KL divergence between *p* and *q*

$$Cost = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE $$\frac{\delta C}{\delta y_i} = \sum_j (p_{ij} - q_{ij})(y_i - y_j) \frac{1}{1 + |y_i - y_j|^2}$$



  - Update y_i to minimize the KL divergence
  - Obtain 2D embedding y_i

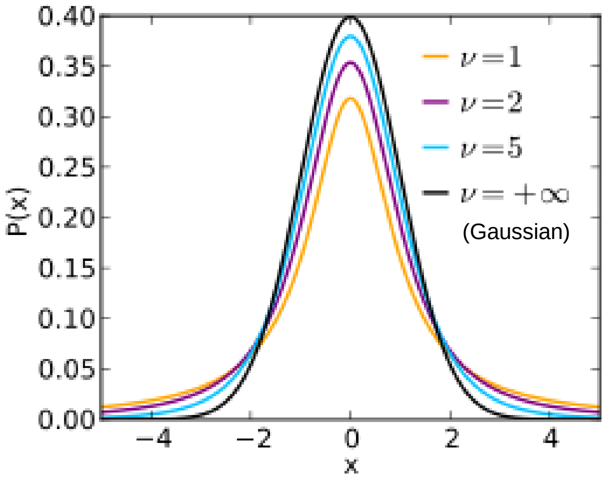
Student t-distribution

- Definition

$$p(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

- Special case (nDOF: $\nu = 1$)

$$p(t) = \frac{1}{\pi\left(1+x^2\right)}$$



Why t-distribution?

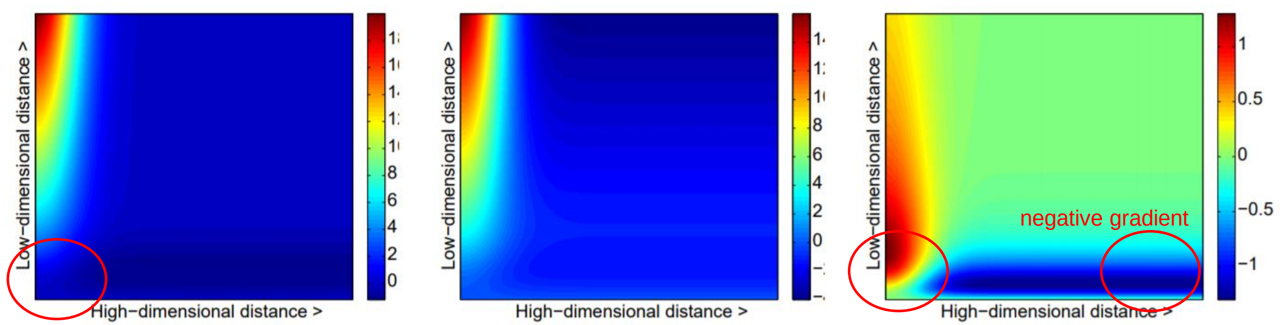
- The Crowding problem

$$p_{j|i} = \text{softmax}[\exp\left(-\frac{|x_i - x_j|^2}{\boxed{2\sigma_i^2}}\right)]$$

Related to perplexity

$$q_{ij} = \frac{\boxed{\left(1 + |y_i - y_j|^2\right)^{-1}}}{\sum_{k \neq l}\left(1 + |y_k - y_l|^2\right)^{-1}}$$

t-distribution



Almost zero gradient
(a) Gradient of SNE.

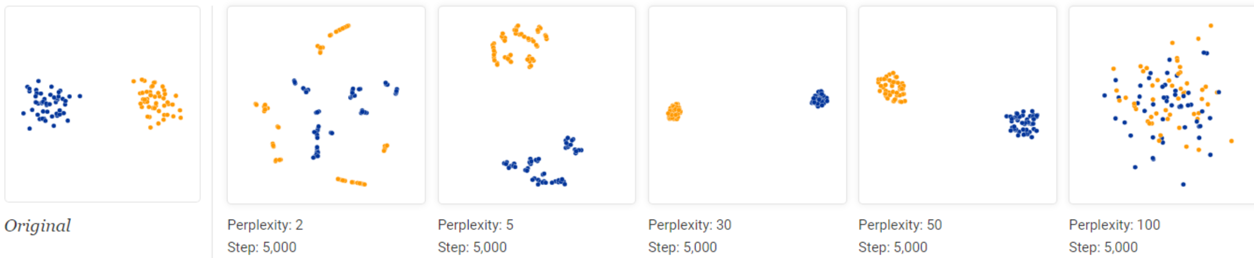
(b) Gradient of UNI-SNE.

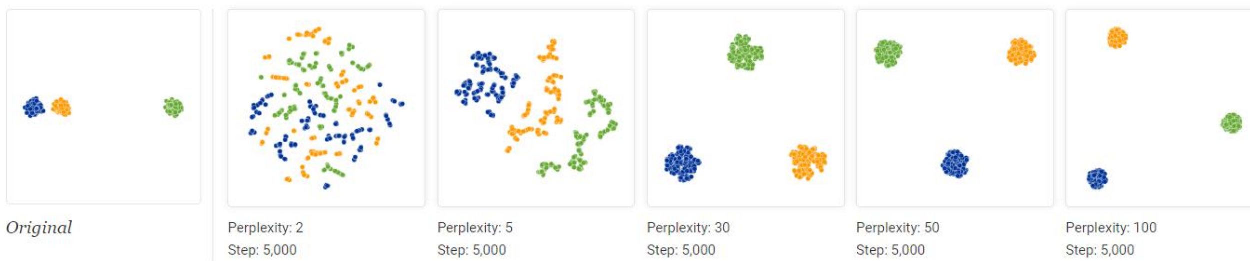
High gradient for small distances
(c) Gradient of t-SNE.

Figure 1: Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation.

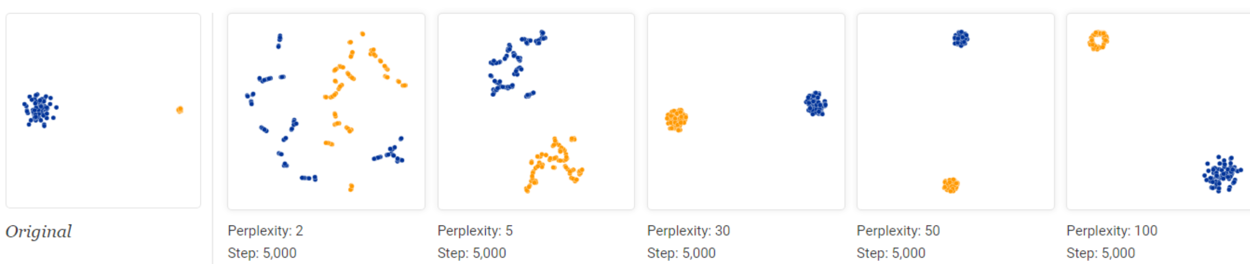
Myth of t-SNE

- Effect of hyperparameters https://distill.pub/2016/misread-tsne/





distances between well-separated clusters in a t-SNE plot may mean nothing



One cannot see relative sizes of clusters in a t-SNE plot

In Python

```
from sklearn.manifold import TSNE


TSNE(n_components=2, perplexity=30.0, early_exaggeration=12.0,
learning_rate=200.0, n_iter=1000, n_iter_without_progress=300,
min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0,
random_state=None, method='barnes_hut', angle=0.5)

tsne = TSNE(n_components=2)
y = tsne.fit_transform(x)
```