손실함수(Loss function)

Open DMQA Seminar

안시후

2021.7.23

발표자 소개

- 안시후 (Sihu Ahn)
  - 고려대학교 산업경영공학과
  - Data Mining & Quality Analytics Lab (지도 교수 : 김성범)
  - 석박사통합과정 (2021.3 ~)
- 관심 연구 분야
  - Data Science
- E-mail : sihuahn@korea.ac.kr

목차

1. 손실함수의 개념

2. 데이터 종류에 따른 손실함수
   - ❖ 데이터 종류
   - ❖ 회귀 모델에서의 손실함수
   - ❖ 분류 모델에서의 손실함수
   - ❖ 다양한 손실함수

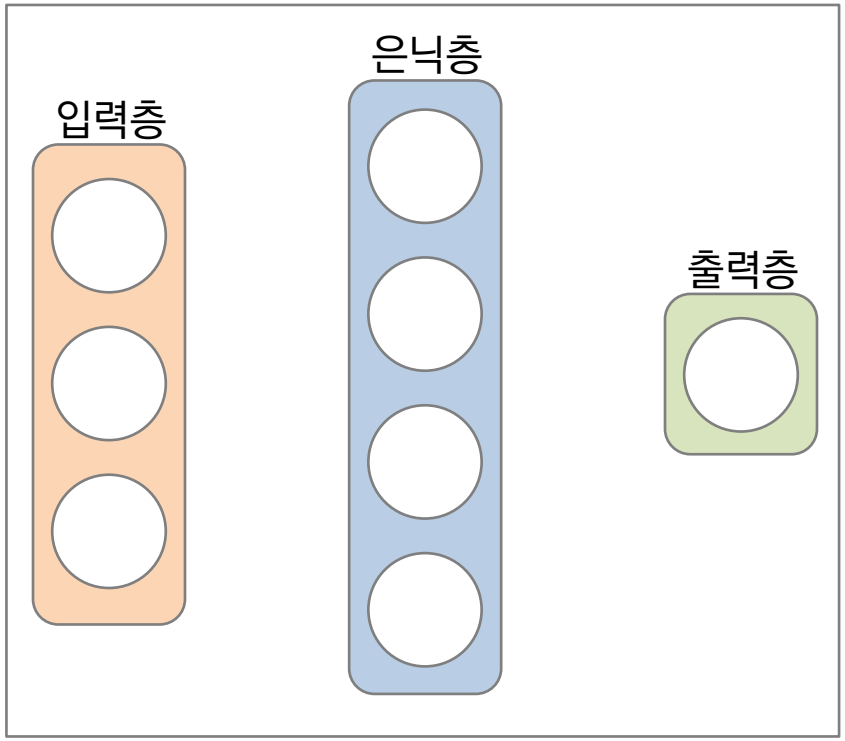
3. 최고의 손실함수 탐색

1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

- 신경망 구조의 딥러닝 모델을 정의

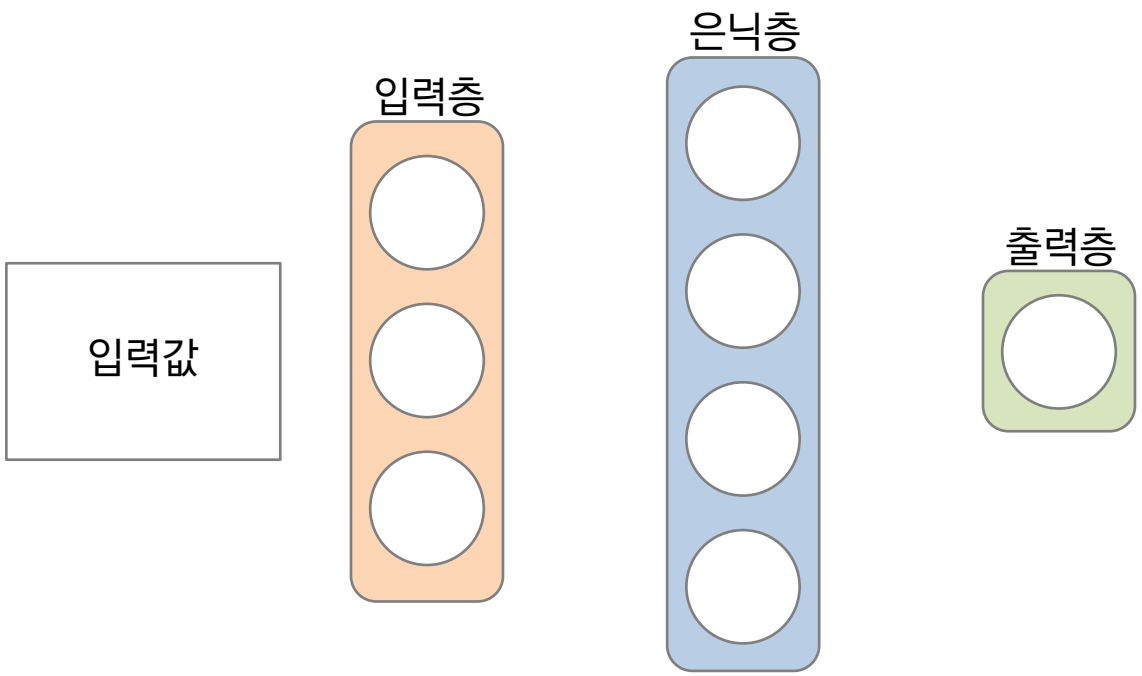
딥러닝 모델



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

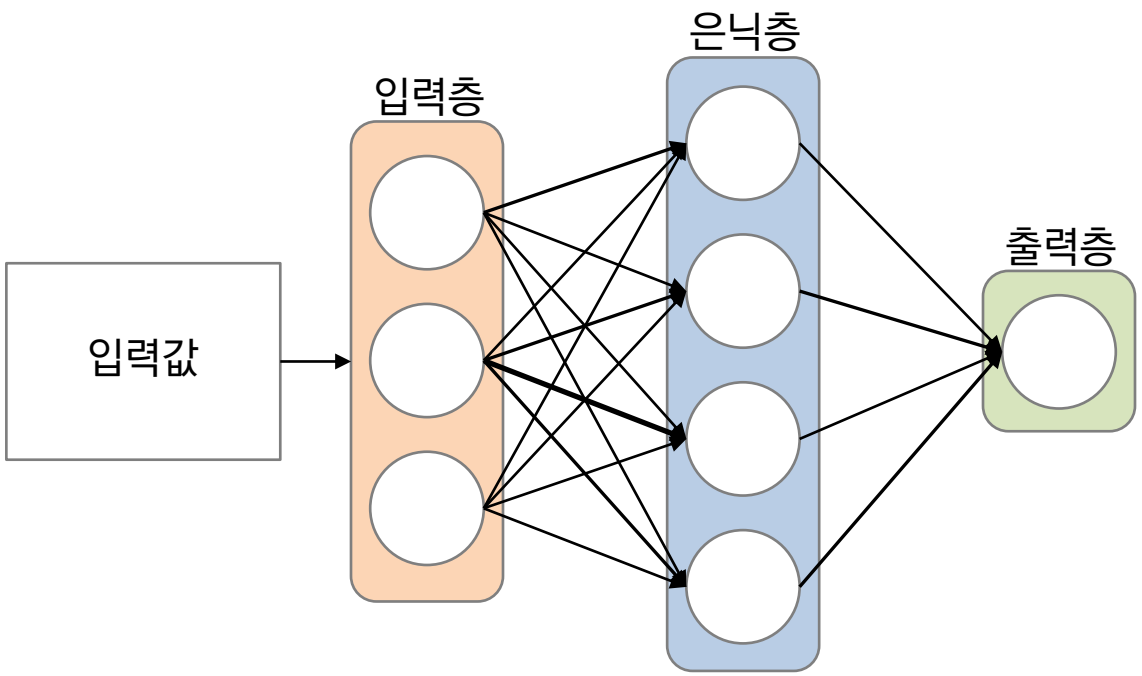
- 임의의 데이터에서 전처리를 수행한 입력값 정의



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

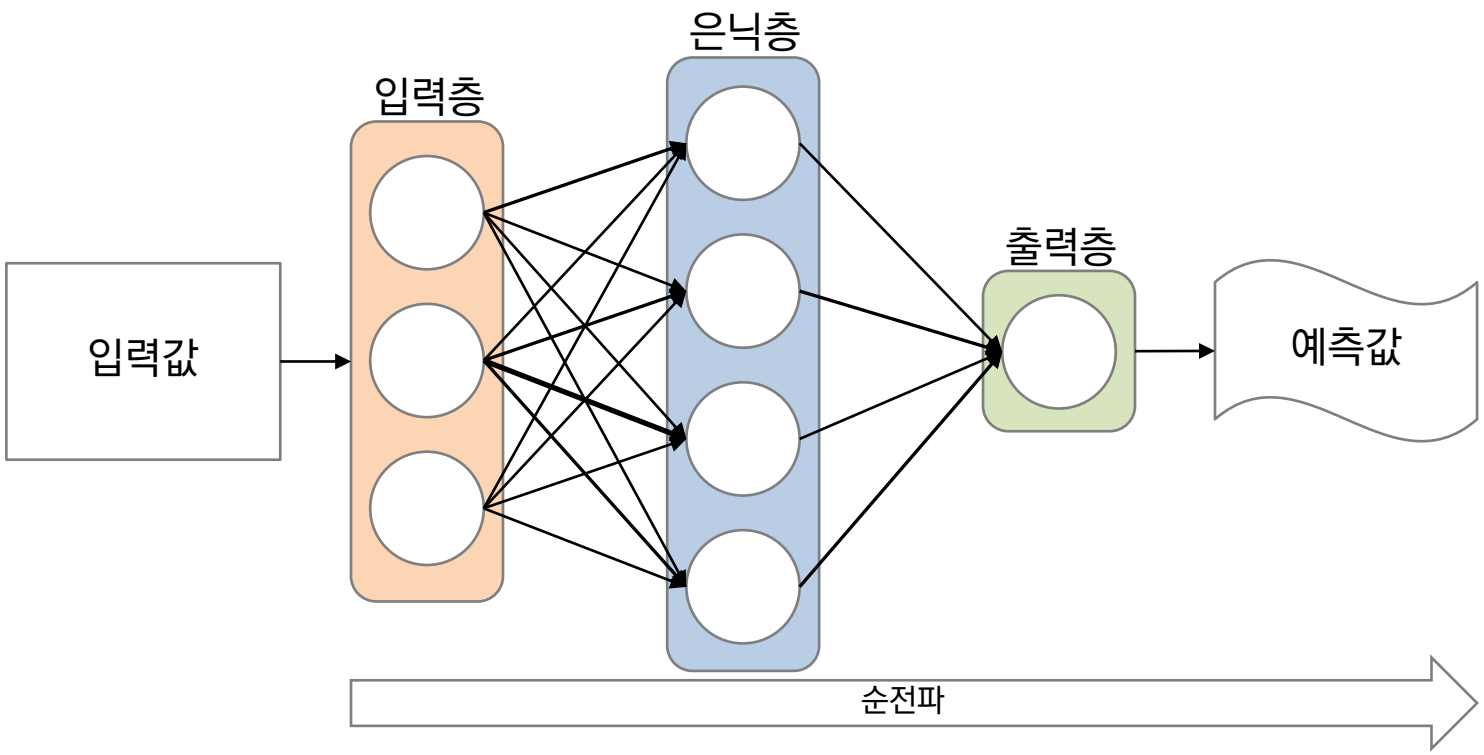
- 입력값을 딥러닝 모델에 넣고 임의의 매개변수를 지정하여 각 층 별로 연결



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

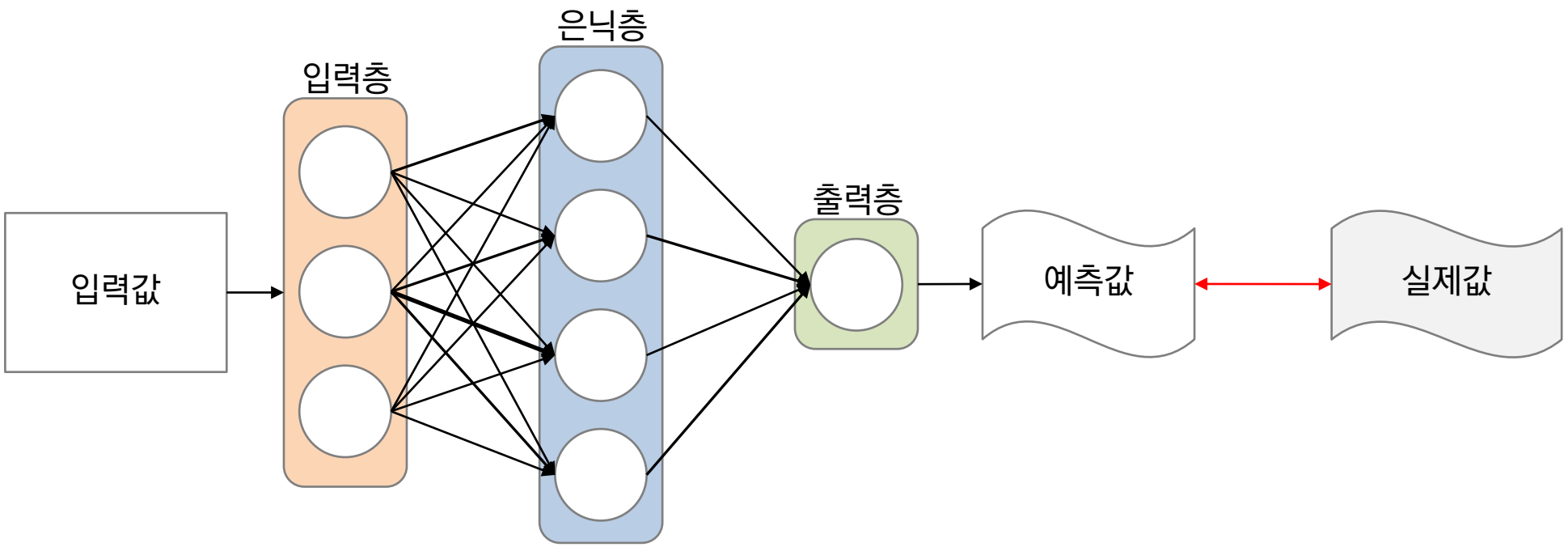
- 딥러닝 모델에서 임의로 지정된 매개변수 값에 따라 순전파를 진행하여 예측값 생성



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

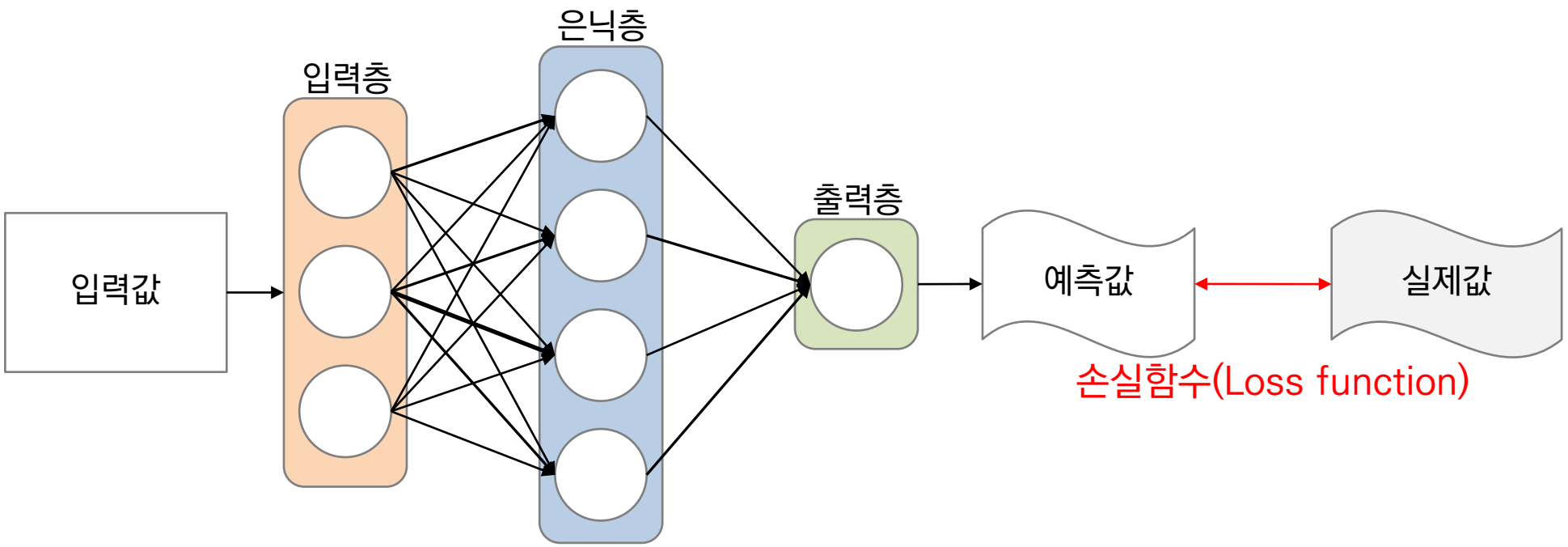
- 예측값을 실제값과 비교



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

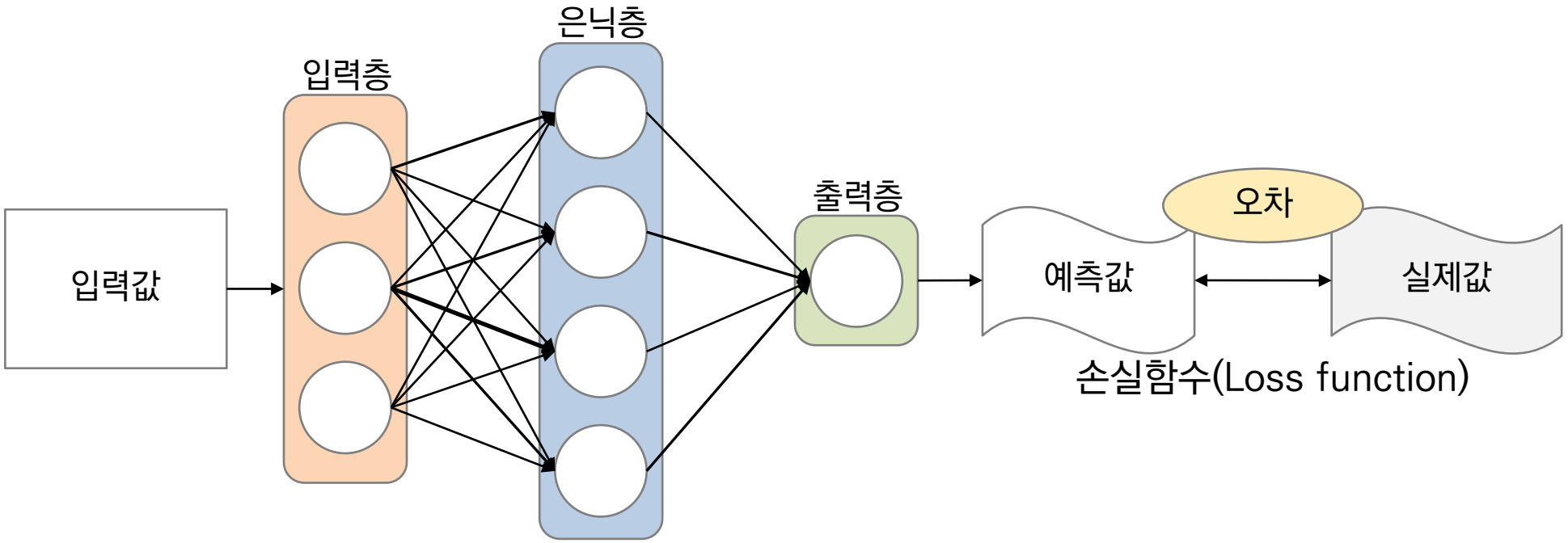
- 예측값을 실제값과 비교



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

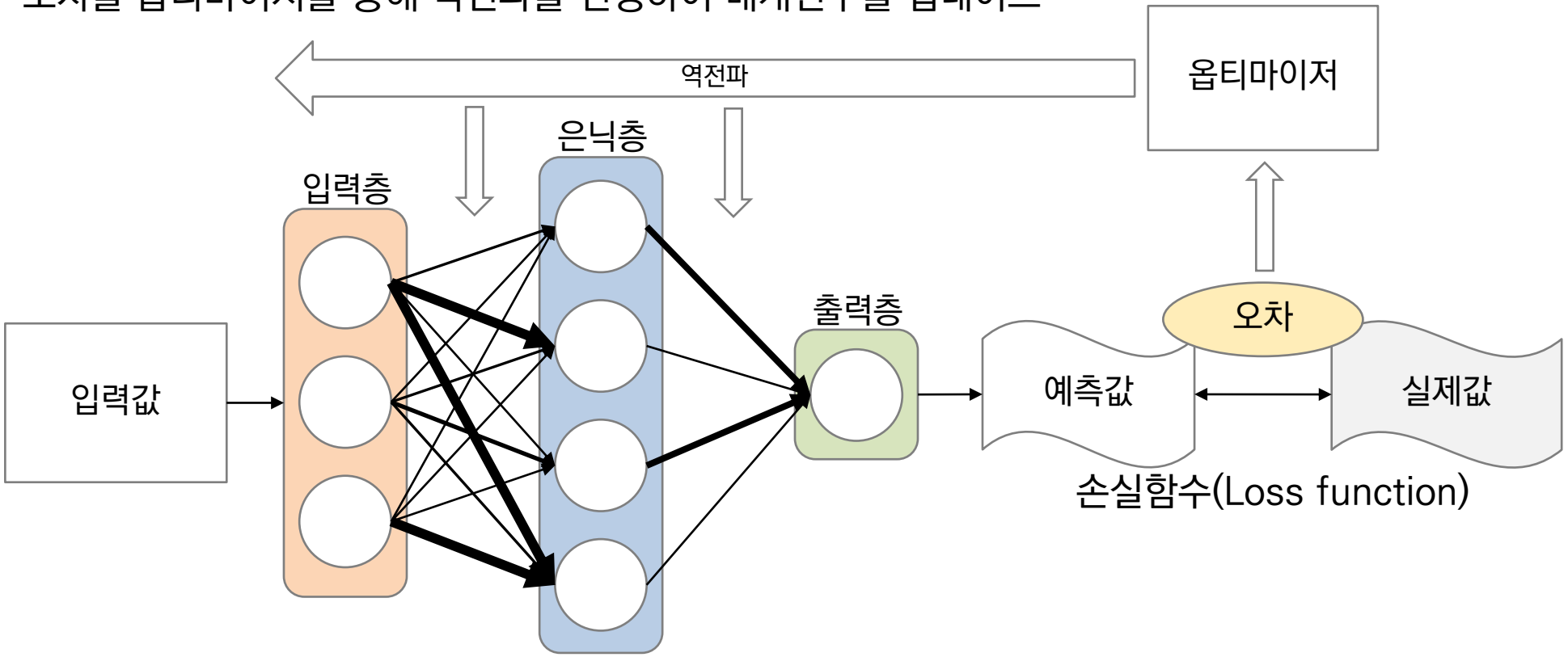
- 예측값을 실제값과 손실함수를 통해 오차 생성



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

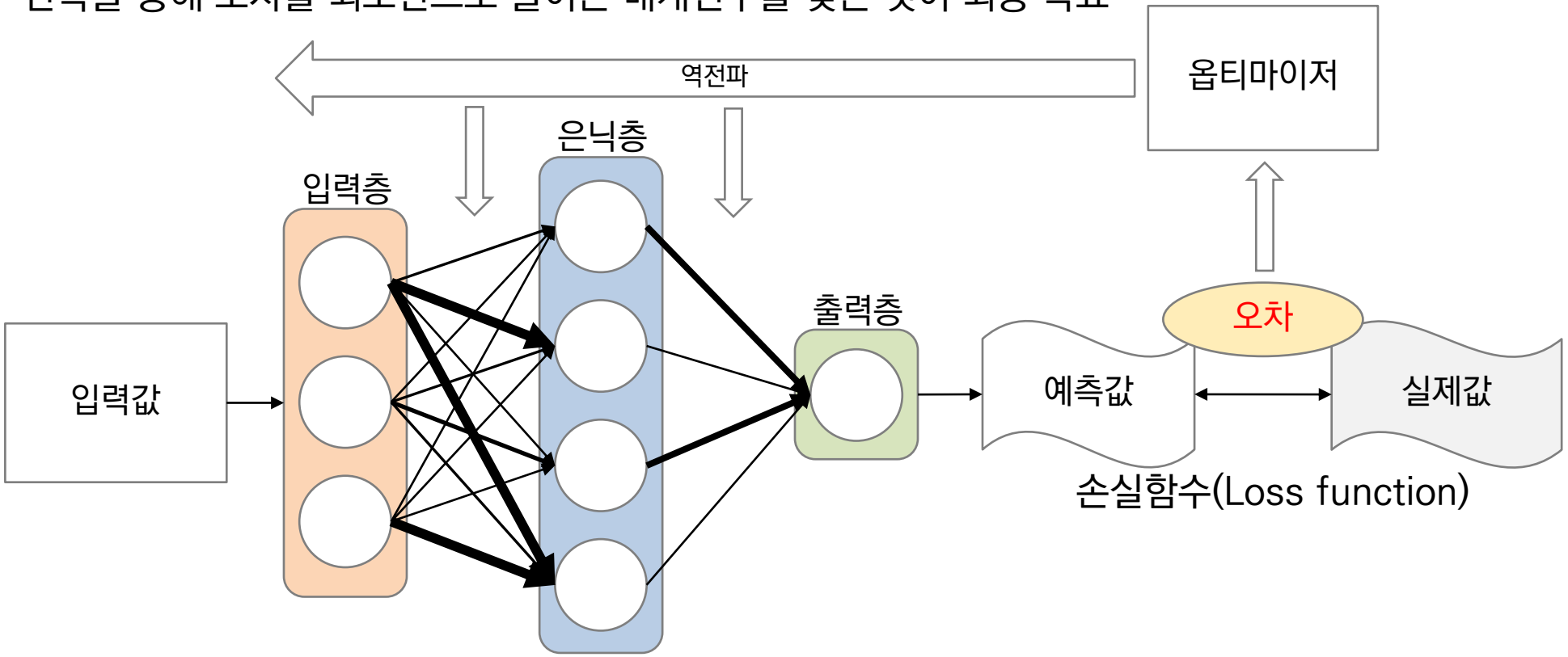
- 오차를 옵티마이저를 통해 역전파를 진행하여 매개변수를 업데이트



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

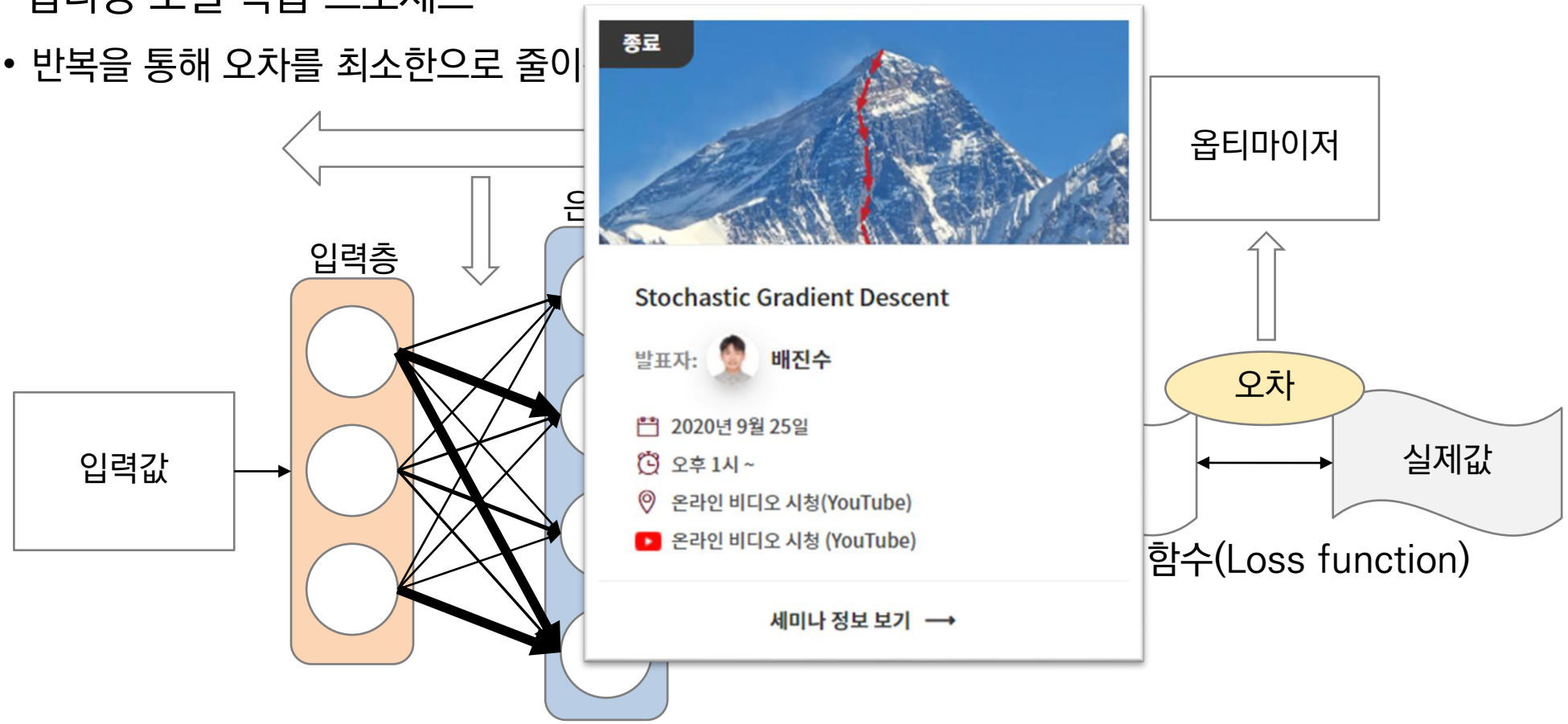
- 반복을 통해 오차를 최소한으로 줄이는 매개변수를 찾는 것이 최종 목표



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

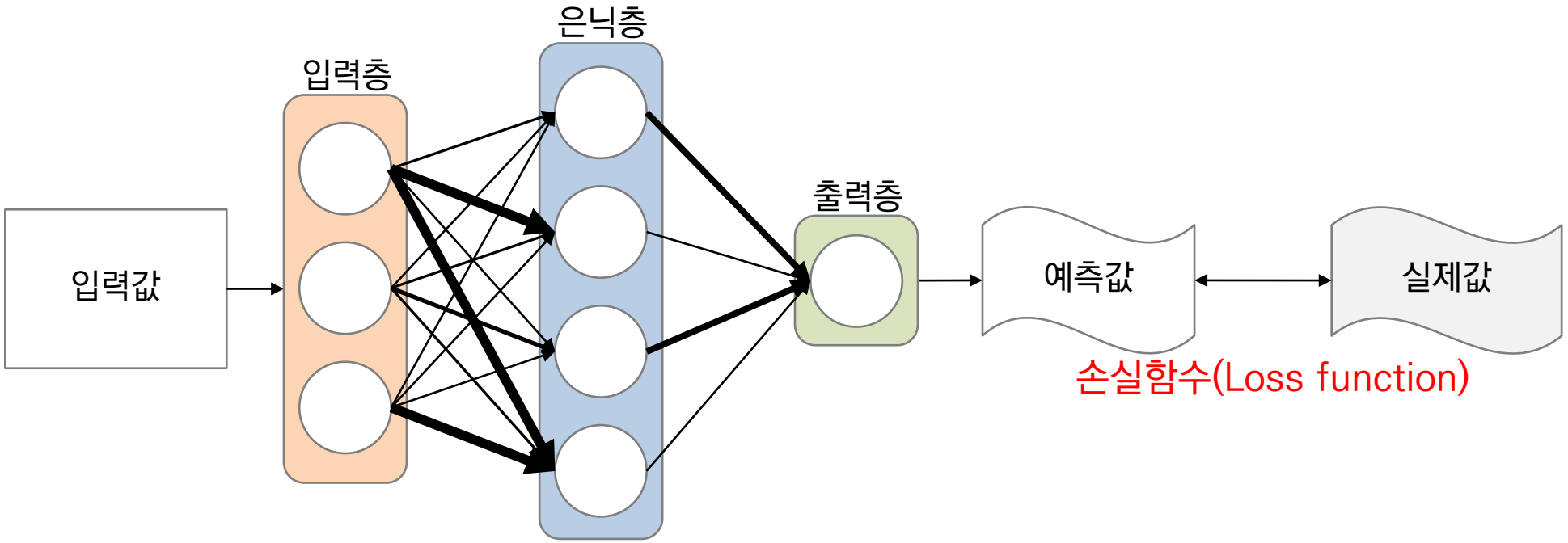
- 반복을 통해 오차를 최소한으로 줄이



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

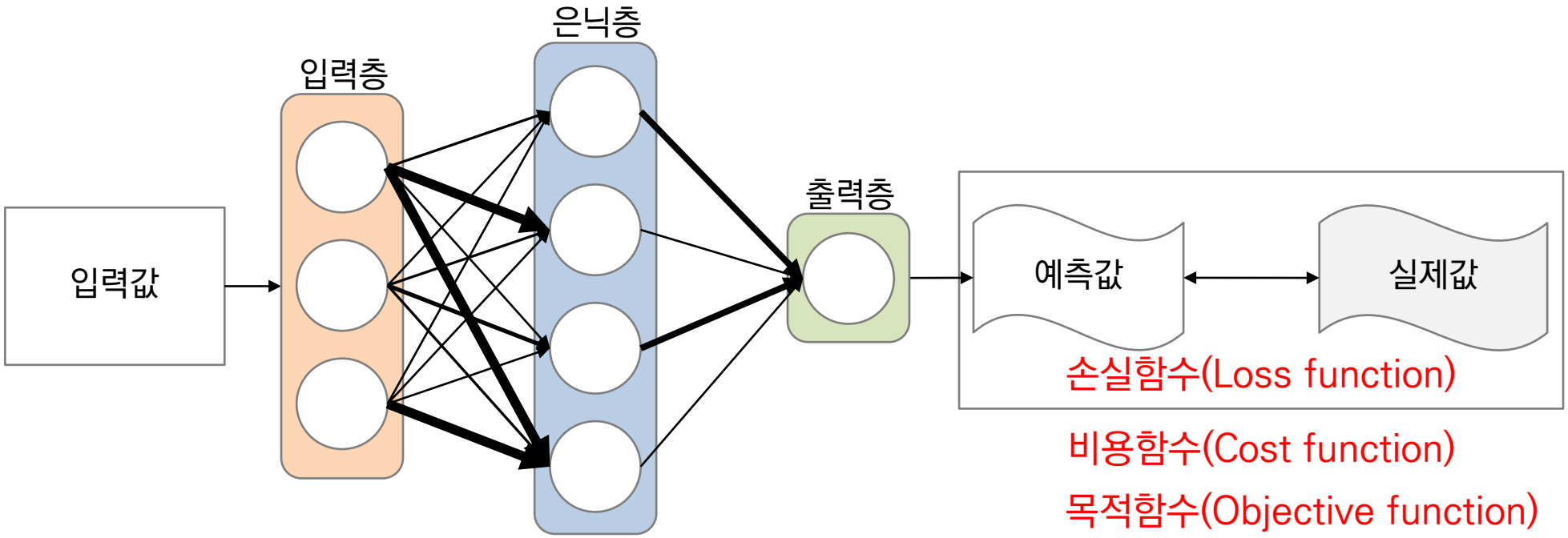
- 딥러닝은 오차를 최소화하는 방향으로 모델 학습을 진행
- 오차를 구하기 위한 손실함수는 필수적



1. 손실함수의 개념

❖ 딥러닝 모델 학습 프로세스

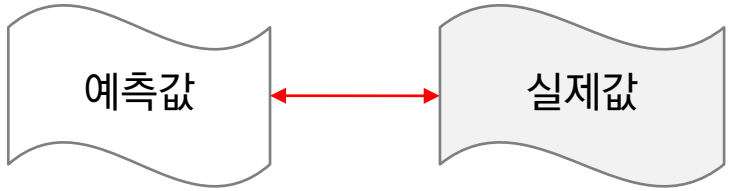
- 딥러닝은 오차를 최소화하는 방향으로 모델 학습을 진행
- 오차를 구하기 위한 손실함수는 필수적



1. 손실함수의 개념

❖ 손실함수(Loss function), 비용함수(Cost function), 목적함수(Objective function)

- 손실함수 : 한 개의 데이터 포인트에서 나온 오차를 최소화하기 위해 정의되는 함수

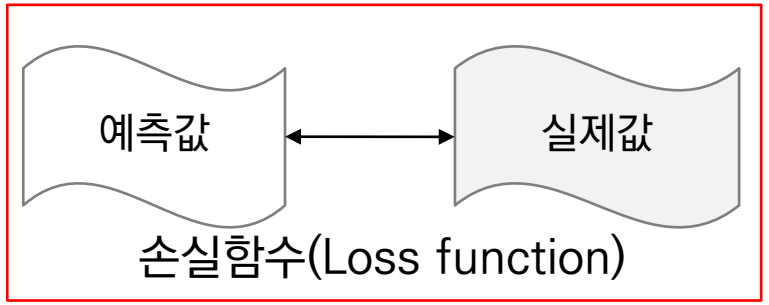


손실함수(Loss function)

1. 손실함수의 개념

❖ 손실함수(Loss function), 비용함수(Cost function), 목적함수(Objective function)

- 손실함수 : 한 개의 데이터 포인트에서 나온 오차를 최소화하기 위해 정의되는 함수
- 비용함수 : 모든 오차를 일반적으로 최소화하기 위해 정의되는 함수

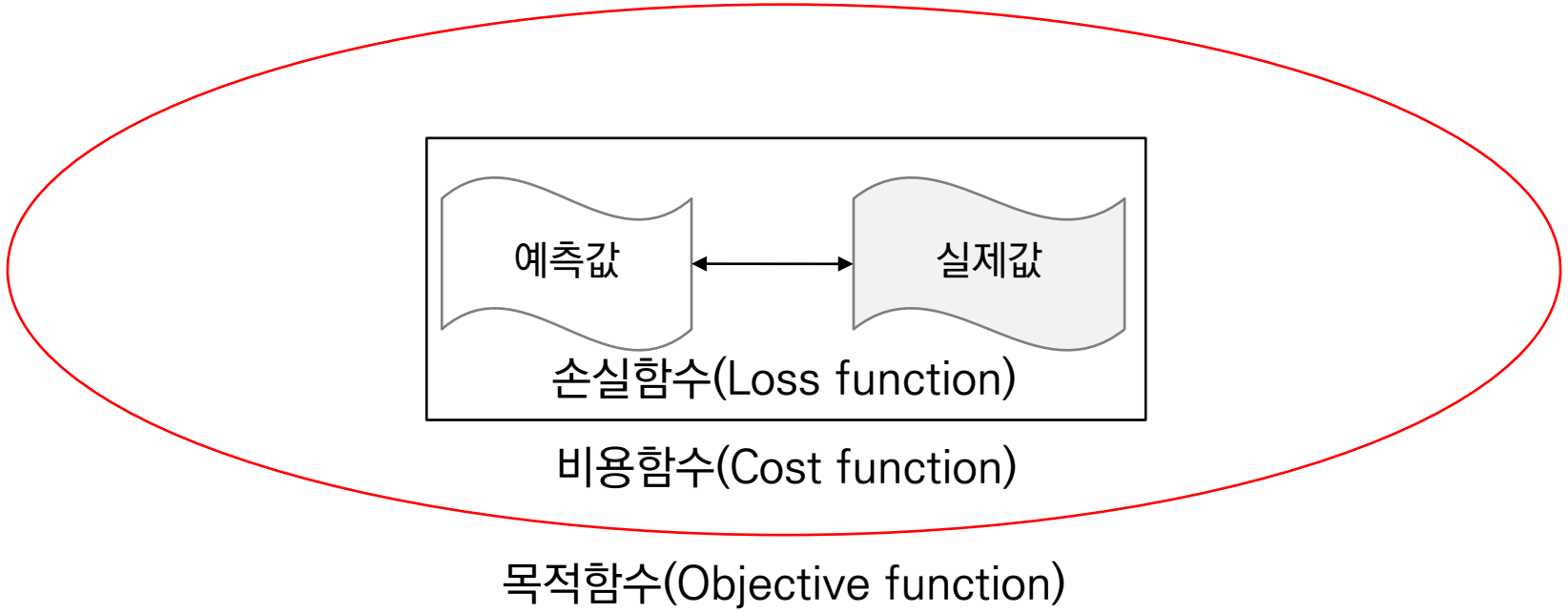


비용함수(Cost function)

1. 손실함수의 개념

❖ 손실함수(Loss function), 비용함수(Cost function), 목적함수(Objective function)

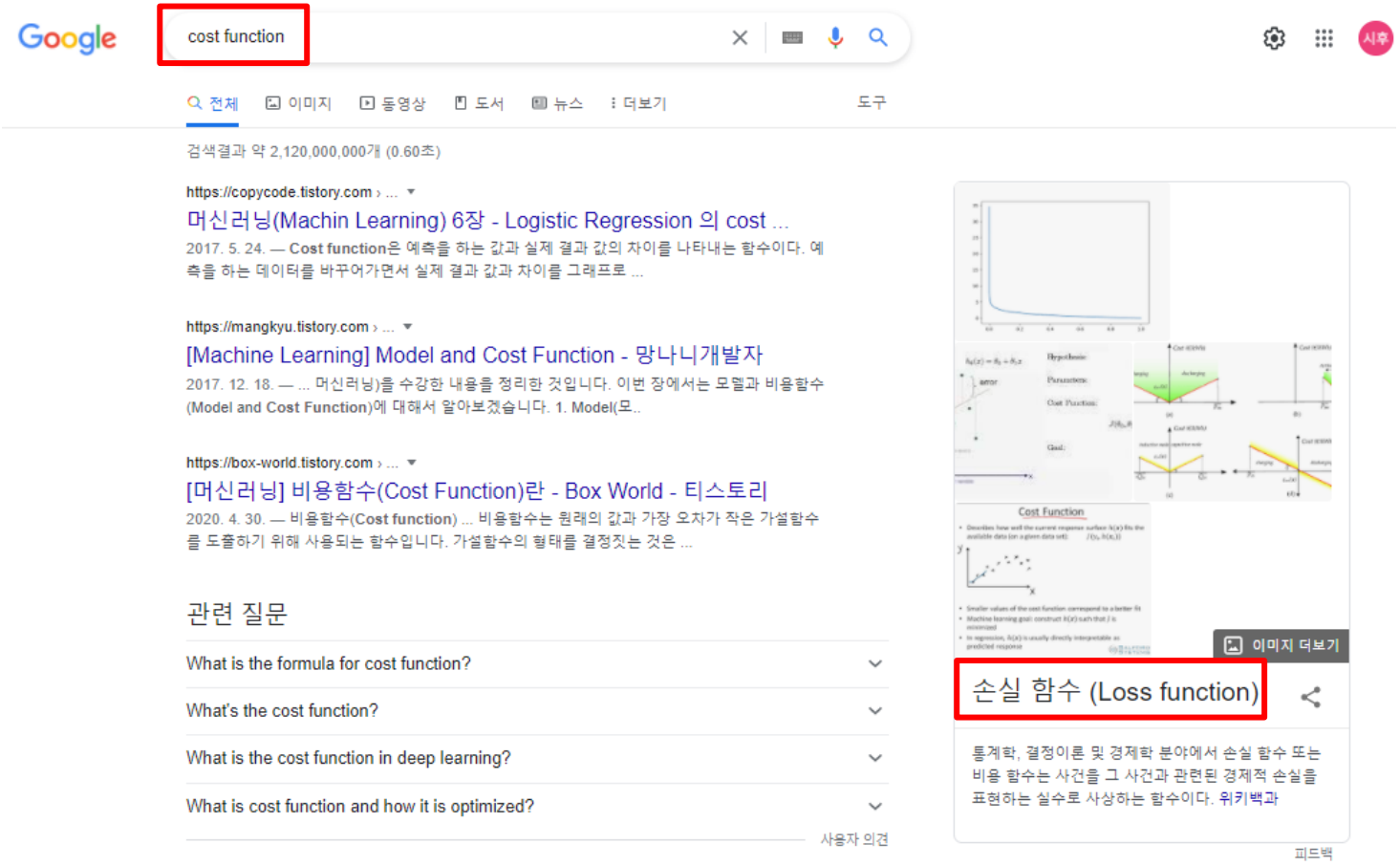
- 손실함수 : 한 개의 데이터 포인트에서 나온 오차를 최소화하기 위해 정의되는 함수
- 비용함수 : 모든 오차를 일반적으로 최소화하기 위해 정의되는 함수
- 목적함수 : 어떠한 값을 최대화 혹은 최소화 시키기 위해 정의되는 함수



1. 손실함수의 개념

❖ 손실함수(Loss function), 비용함수(Cost function), 목적함수(Objective function)

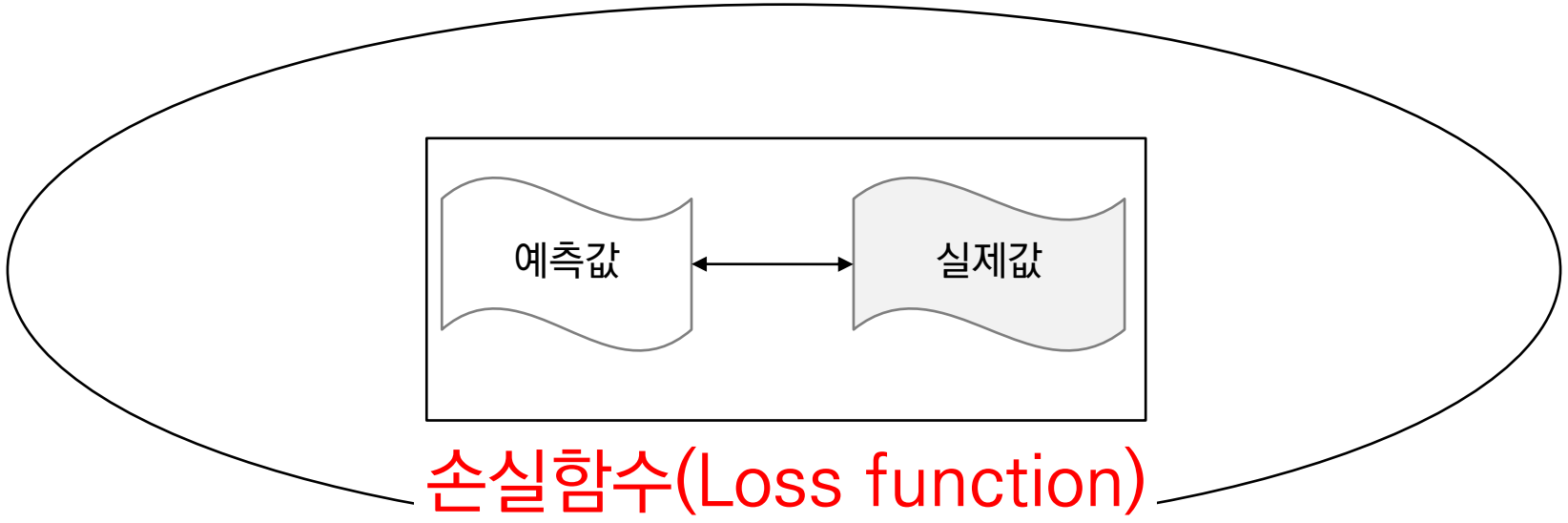
- 손실함수 =< 비용함수 =< 목적함수



1. 손실함수의 개념

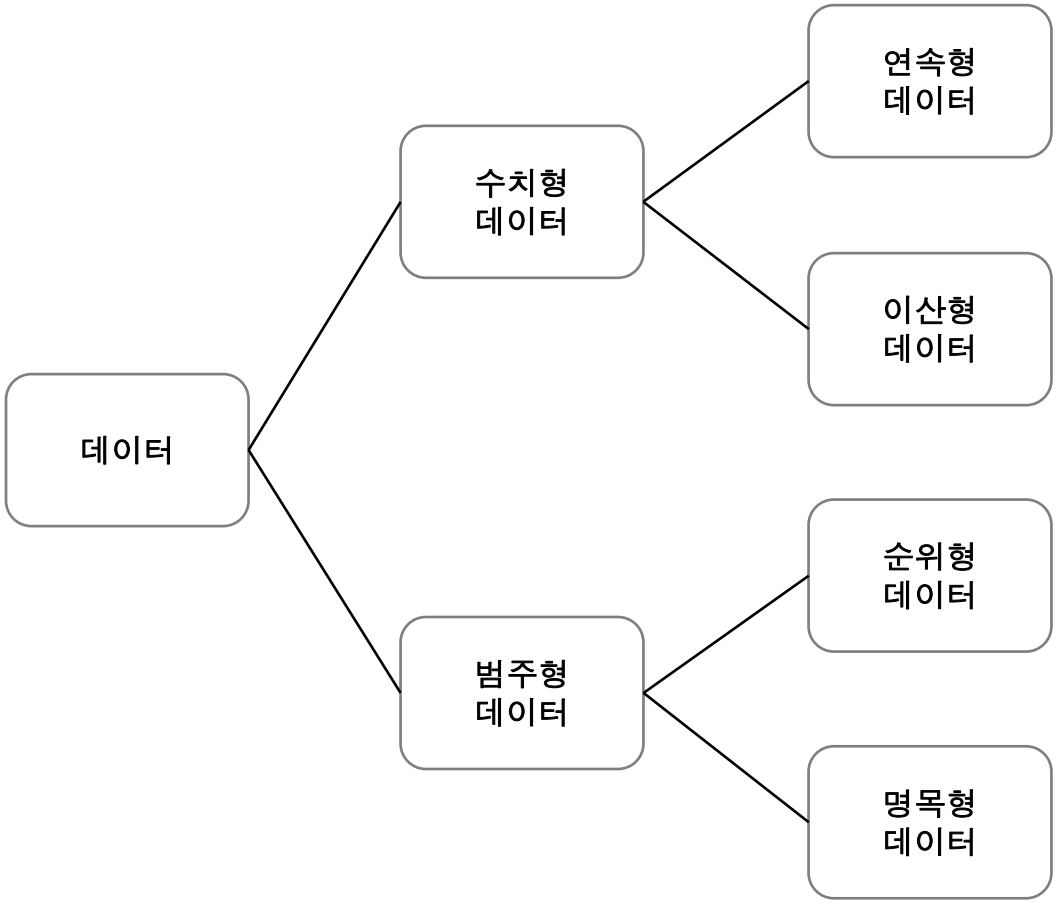
❖ 손실함수(Loss function), 비용함수(Cost function), 목적함수(Objective function)

- 손실함수 =< 비용함수 =< 목적함수



2. 데이터 형식에 따른 손실함수

❖ 데이터 형식



수치가 연속적인 데이터(키, 몸무게 등)

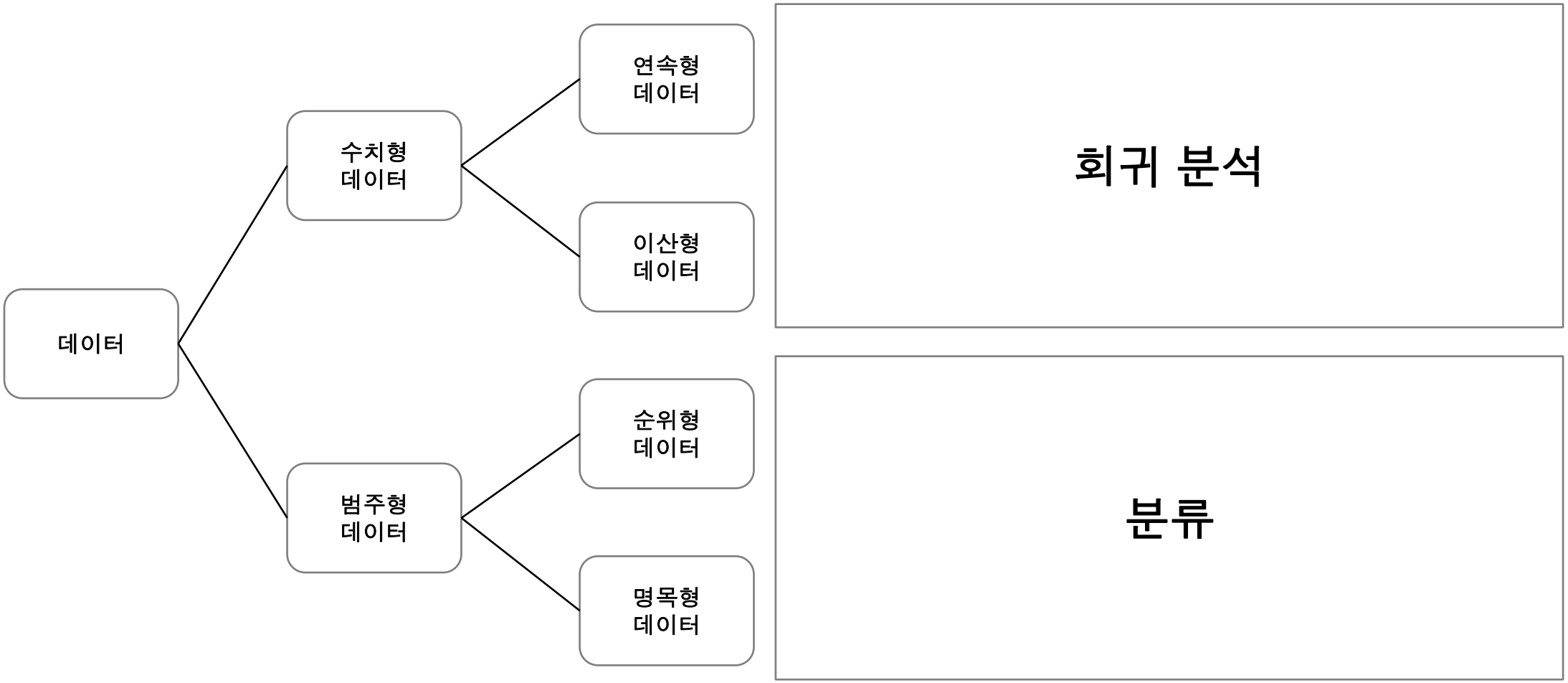
셀 수 있는 데이터(자동차 사고 건수 등)

범주간에 순서의 의미가 있는 데이터(평점, 선호도 등)

범주간에 순서의 의미가 없는 데이터(성별, 혈액형 등)

2. 데이터 형식에 따른 손실함수

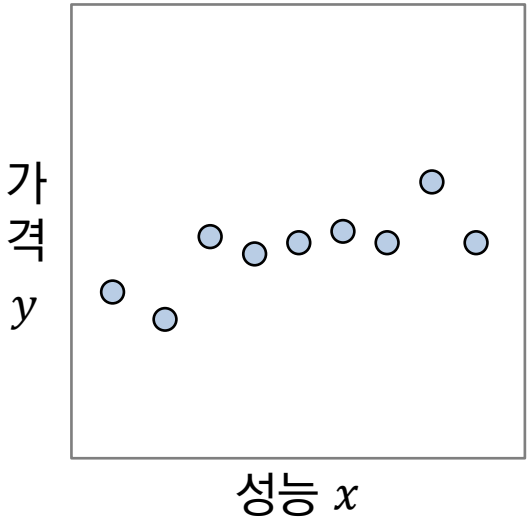
❖ 데이터 형식

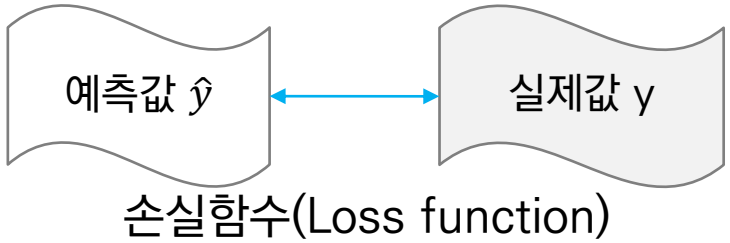


2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 핸드폰 성능 x에 따른 가격 y 정의

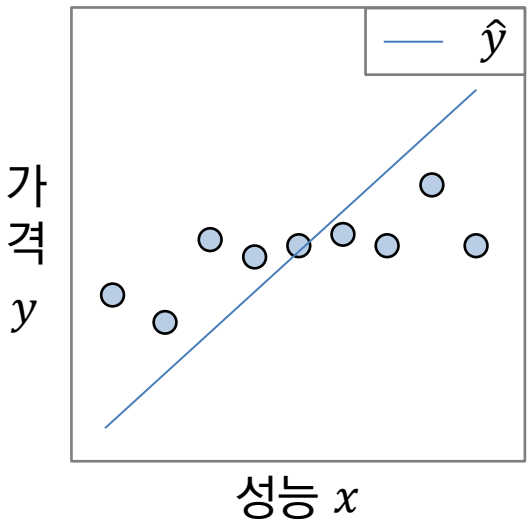


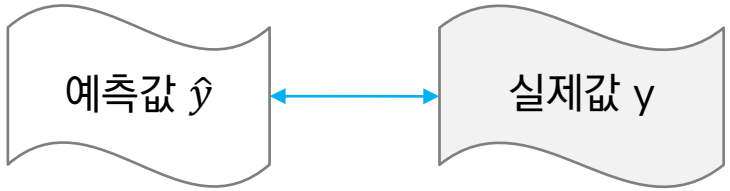


2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 회귀는 성능 x에 따른 가격 y를 예측하여 선 $\hat{y}$을 구하는 문제



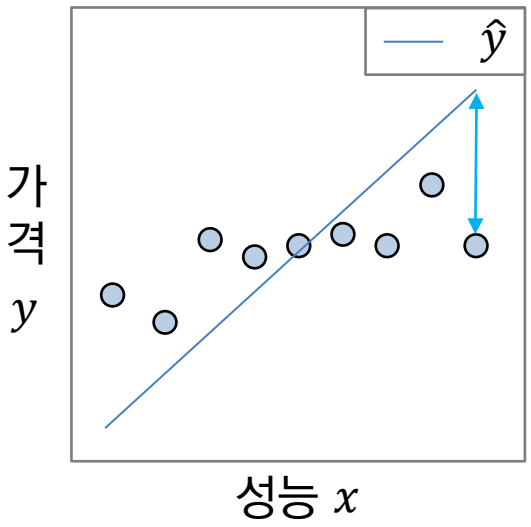


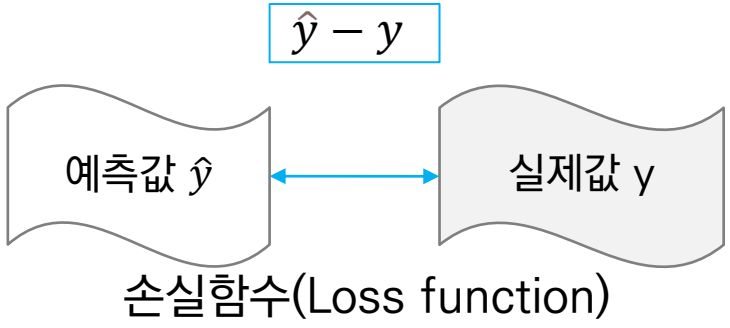
손실함수(Loss function)

2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 손실함수를 이용하여 예측값 $\hat{y}$와 실제값 y의 오차 $\hat{y} - y$를 도출

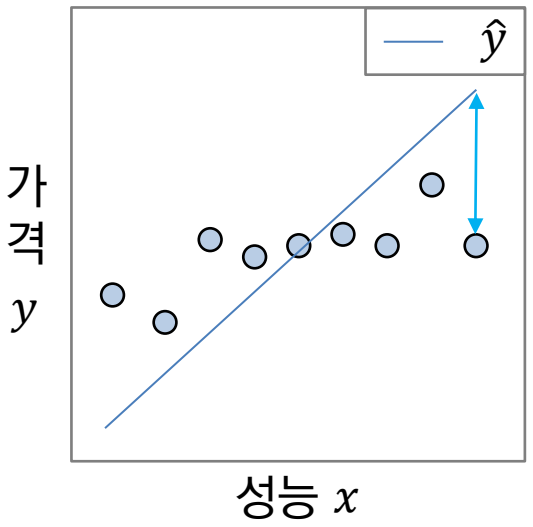


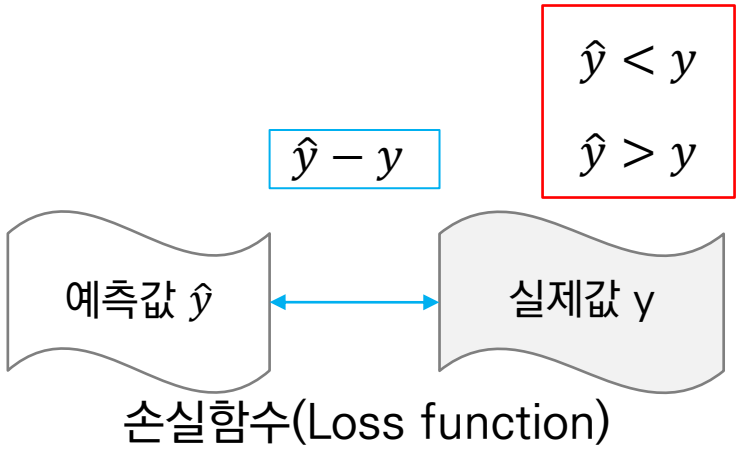


2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 오차 $\hat{y} - y$가 음수 또는 양수로 나올 수 있음

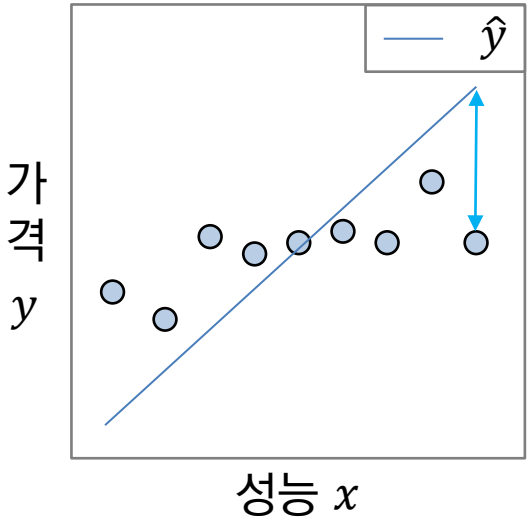


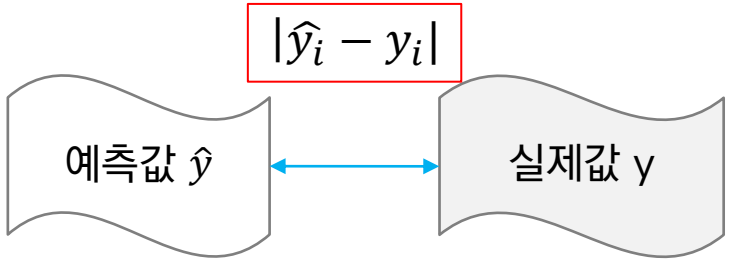


2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 절대값을 씌워 순수한 오차 값을 도출



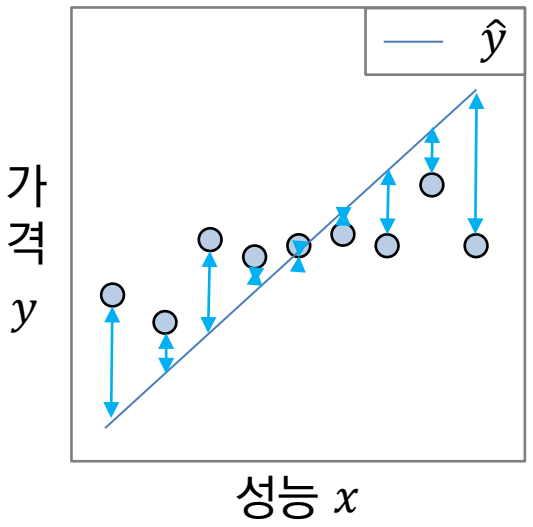


손실함수(Loss function)

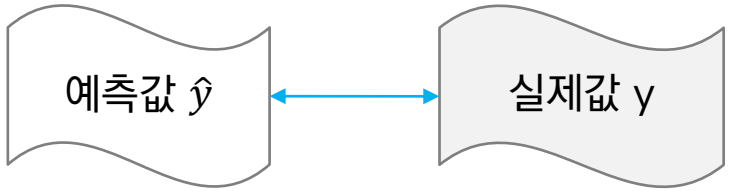
2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 전체 데이터에 대한 오차를 더하고 평균으로 나눈 평균절대오차(MAE) 도출



$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

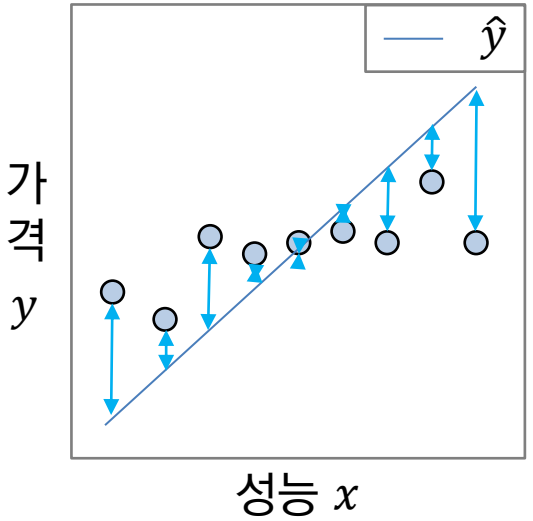


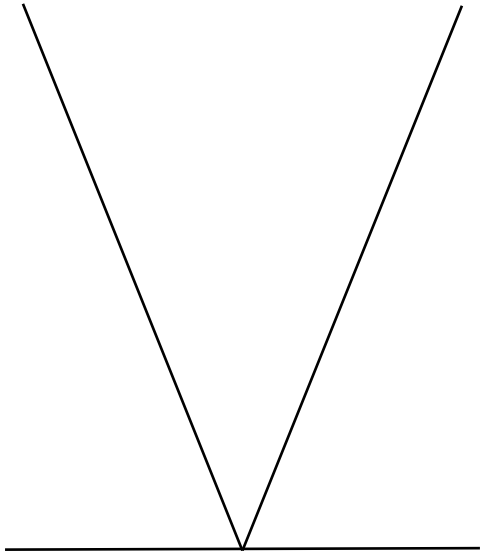
손실함수(Loss function)

2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 평균절대오차(MAE)의 그래프는 미분이 불가능



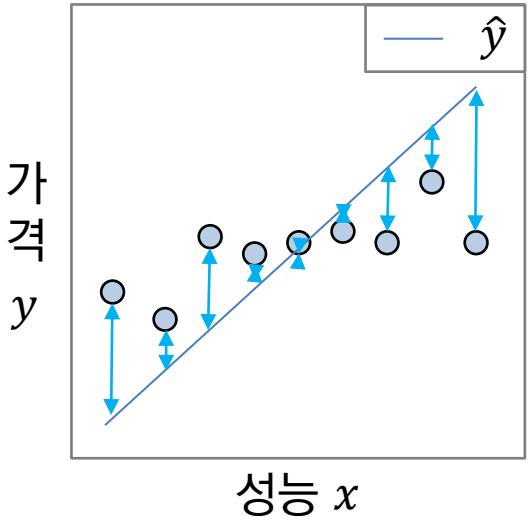


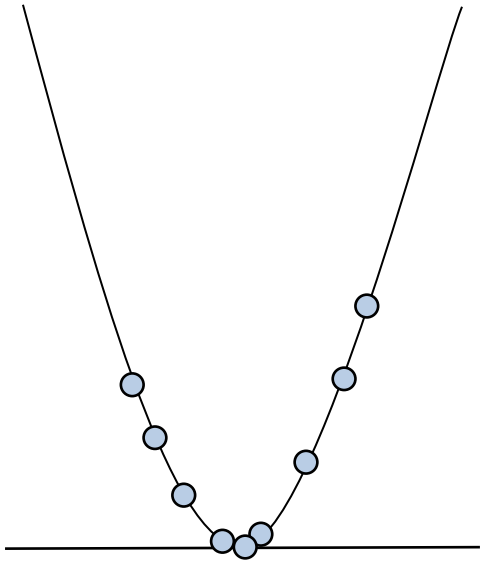
$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 평균제곱오차(MSE)



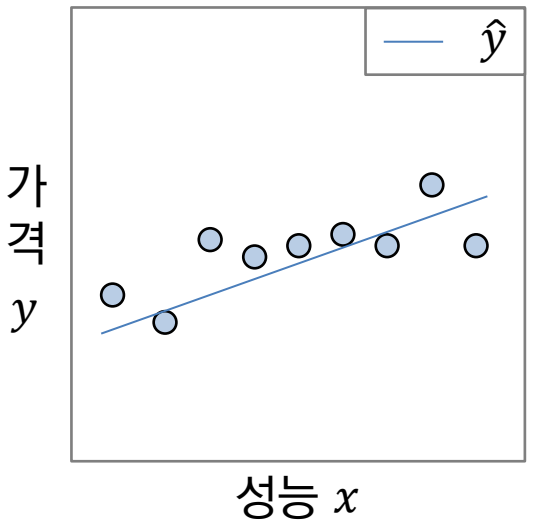


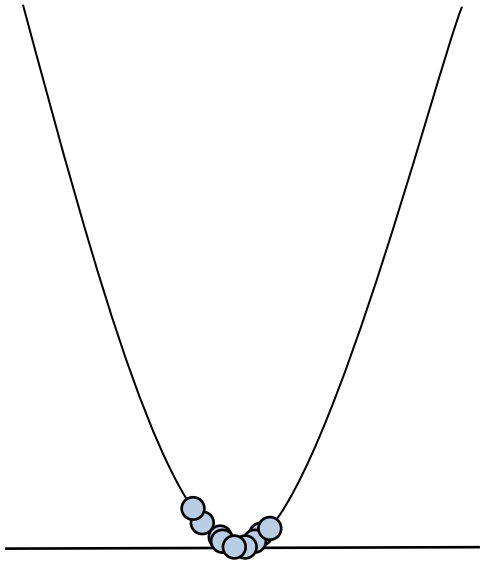
$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

2. 데이터 형식에 따른 손실함수

❖ 회귀 모델에서의 손실함수

- 평균제곱오차(MSE)를 통해 학습을 진행한 모습



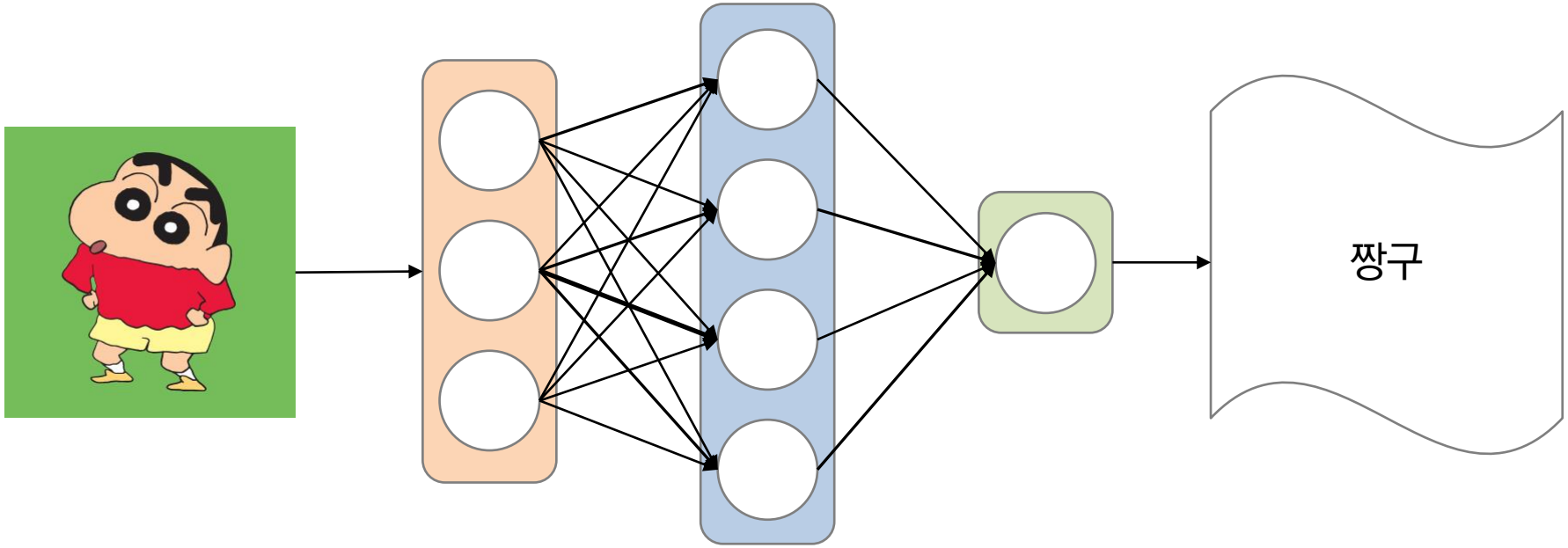


$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

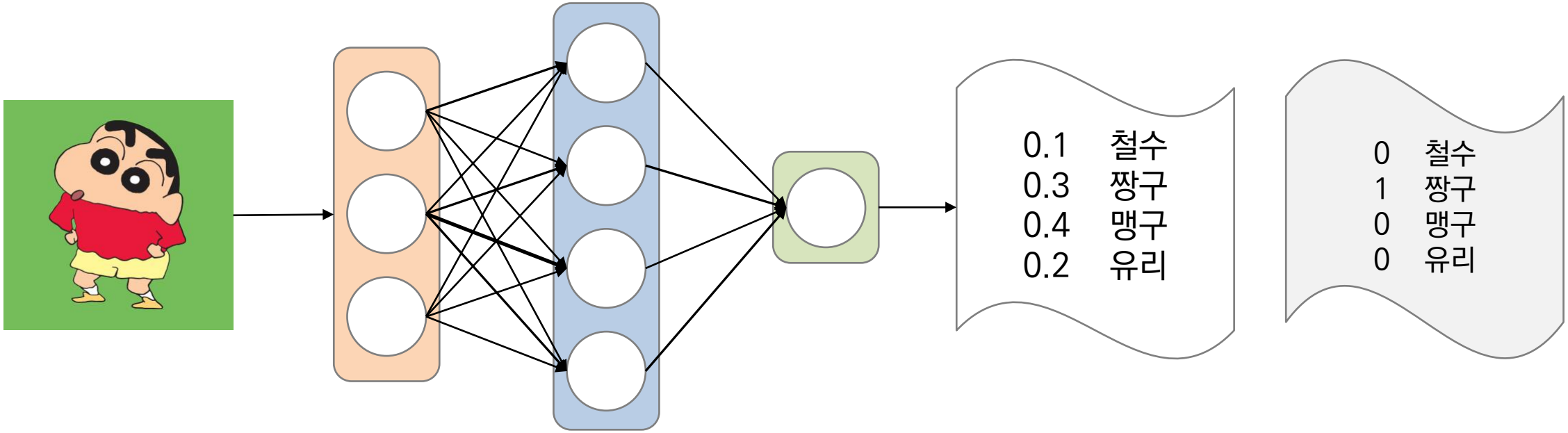
- 사진에 있는 캐릭터 이름을 분류하는 모델 정의



2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

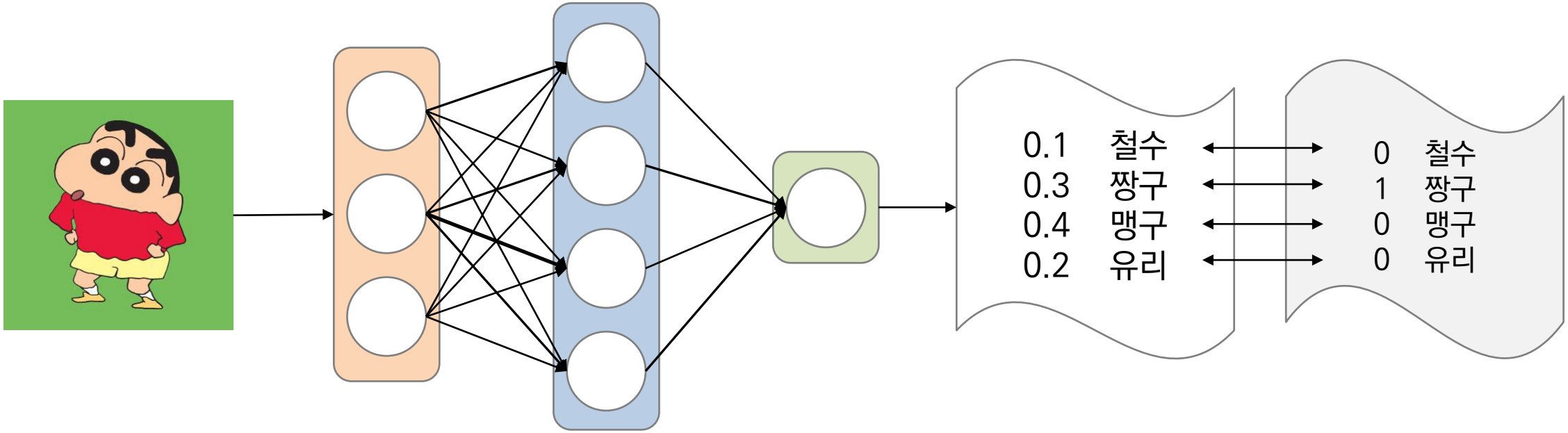
- 순전파를 진행한 모델에서 나온 예측값



2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

- 예측값과 원핫벡터로 이루어진 실제값과 차이를 교차엔트로피 손실함수를 통해 오차 도출

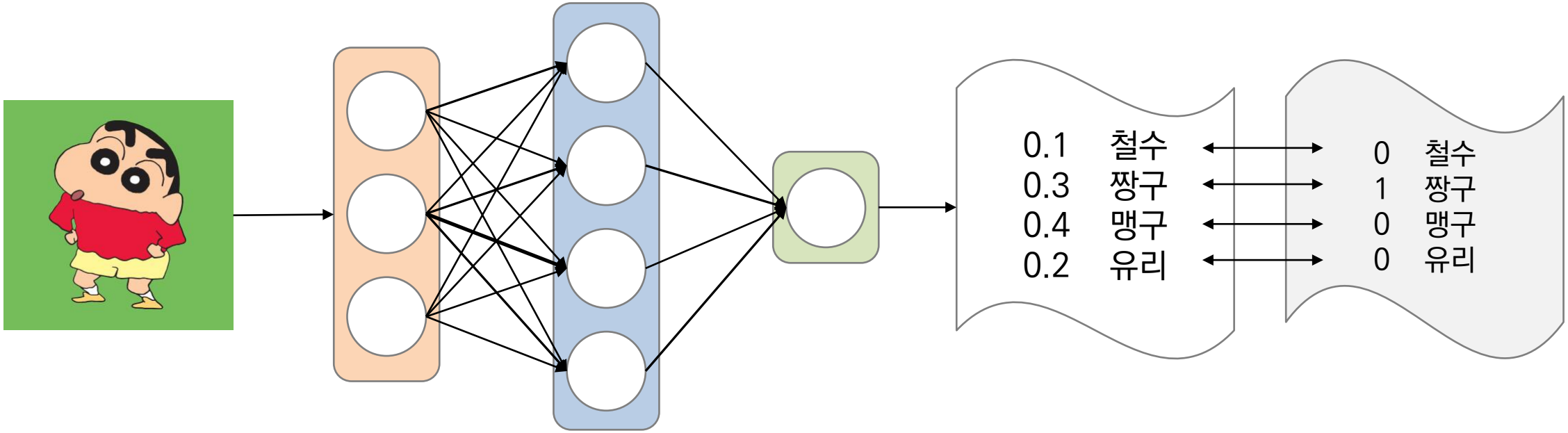


$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} L_{ic}\log(P_{ic})$$

2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

- 예측값과 원핫벡터로 이루어진 실제값과 차이를 교차엔트로피 손실함수를 통해 오차 도출



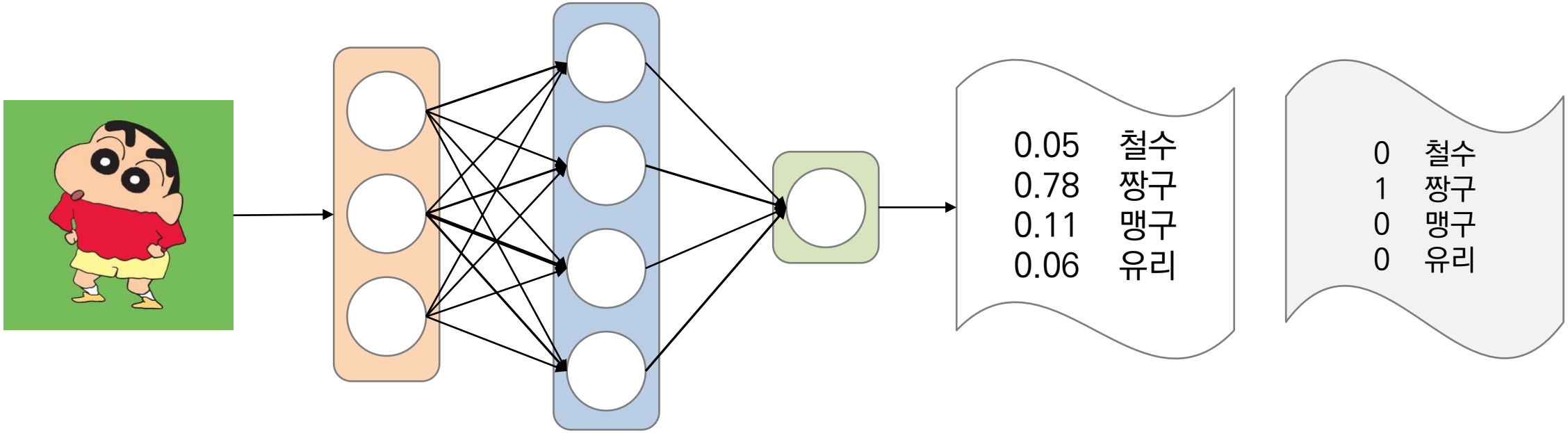
$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} L_{ic}\log(P_{ic})$$

$$-[0\log(0.1) + 1\log(0.3) + 0\log(0.4) + 0\log(0.2)] = 0.5229$$

2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

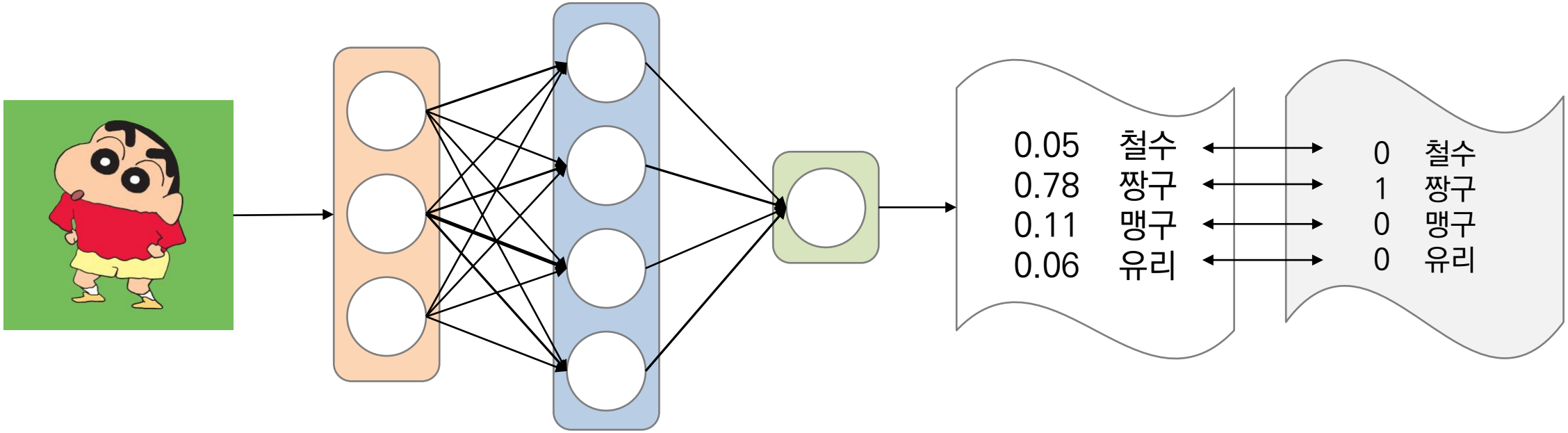
- 역전파를 여러 번 진행한 모델에서 나온 예측값



2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

- 예측값과 원핫벡터로 이루어진 실제값과 차이를 교차엔트로피 손실함수를 통해 오차 도출

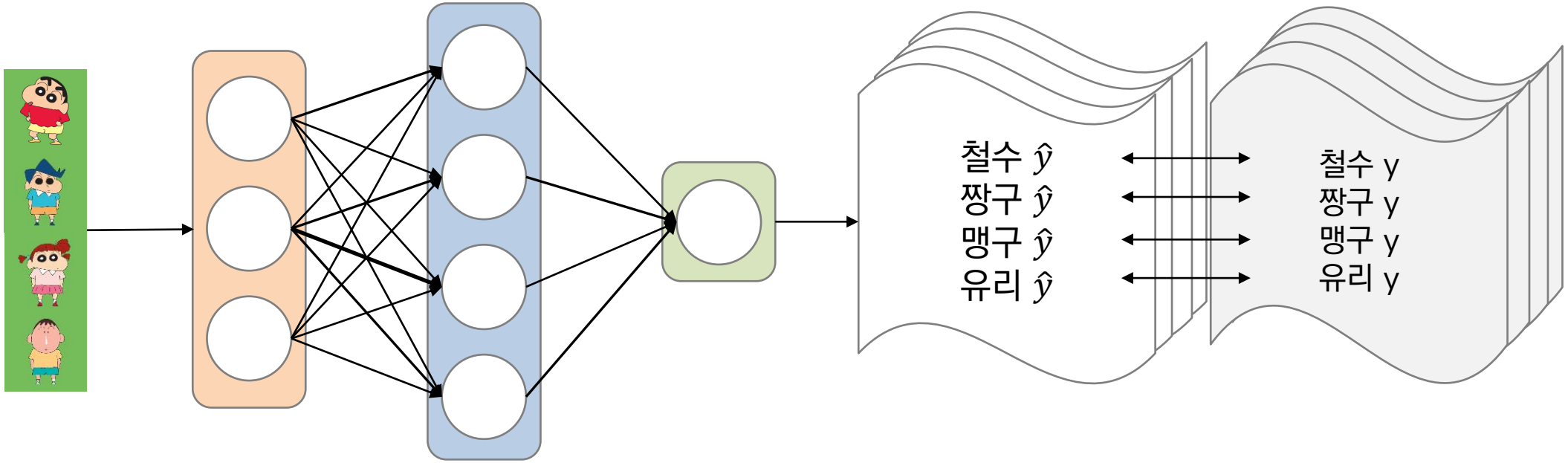


$$-[0\log(0.05) + 1\log(0.78) + 0\log(0.11) + 0\log(0.06)] = 0.1079$$

2. 데이터 형식에 따른 손실함수

❖ 분류 모델에서의 손실함수

- 교차 엔트로피 $-\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} L_{ic}\log(P_{ic})$



2. 데이터 형식에 따른 손실함수

❖ 다양한 손실함수 https://pytorch.org/docs/stable/nn#loss-functions

- Pytorch에서 제공하는 평균절대오차, 평균제곱오차, 교차엔트로피 손실함수

MAE

L1LOSS

CLASS `torch.nn.L1Loss(size_average=None, reduce=None, reduction='mean')` [SOURCE]

Creates a criterion that measures the mean absolute error (MAE) between each element in the input x and target y.

MSE

MSELOSS

CLASS `torch.nn.MSELoss(size_average=None, reduce=None, reduction='mean')` [SOURCE]

Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y.

Cross entropy

BCELOSS

CLASS `torch.nn.BCELoss(weight=None, size_average=None, reduce=None, reduction='mean')` [SOURCE]

Creates a criterion that measures the Binary Cross Entropy between the target and the output:

BCEWITHLOGITSLOSS

CLASS `torch.nn.BCEWithLogitsLoss(weight=None, size_average=None, reduce=None, reduction='mean', pos_weight=None)` [SOURCE]

This loss combines a *Sigmoid* layer and the *BCELoss* in one single class. This version is more numerically stable than using a plain *Sigmoid* followed by a *BCELoss* as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability.

NLLLOSS

CLASS `torch.nn.NLLLoss(weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean')` [SOURCE]

The negative log likelihood loss. It is useful to train a classification problem with C classes.

CROSSENTROPYLOSS

CLASS `torch.nn.CrossEntropyLoss(weight=None, size_average=None, ignore_index=-100, reduce=None, reduction='mean')` [SOURCE]

This criterion combines `LogSoftmax` and `NLLLoss` in one single class.

2. 데이터 형식에 따른 손실함수

❖ 다양한 손실함수 https://pytorch.org/docs/stable/nn#loss-functions

- Pytorch에서 제공하는 손실함수 21종(2021/7/23 기준)

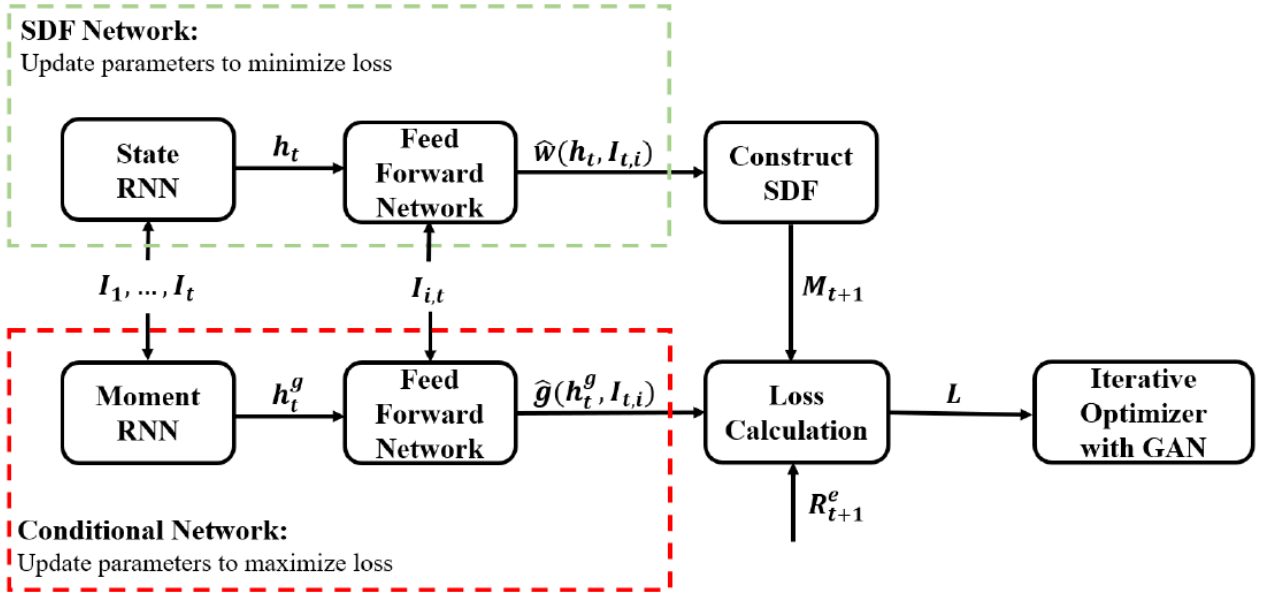
Loss Functions

`nn.L1Loss`	Creates a criterion that measures the mean absolute error (MAE) between each element in the input x and target y.
`nn.MSELoss`	Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y.
`nn.CrossEntropyLoss`	This criterion combines `LogSoftmax` and `NLLLoss` in one single class.
`nn.CTCLoss`	The Connectionist Temporal Classification loss.
`nn.NLLLoss`	The negative log likelihood loss.
`nn.PoissonNLLLoss`	Negative log likelihood loss with Poisson distribution of target.
`nn.GaussianNLLLoss`	Gaussian negative log likelihood loss.
`nn.KLDivLoss`	The Kullback-Leibler divergence loss measure
`nn.BCELoss`	Creates a criterion that measures the Binary Cross Entropy between the target and the output:
`nn.BCEWithLogitsLoss`	This loss combines a *Sigmoid* layer and the *BCELoss* in one single class.
`nn.MarginRankingLoss`	Creates a criterion that measures the loss given inputs $x1$, $x2$, two 1D mini-batch *Tensors*, and a label 1D mini-batch tensor y (containing 1 or -1).
`nn.HingeEmbeddingLoss`	Measures the loss given an input tensor x and a labels tensor y (containing 1 or -1).
`nn.MultiLabelMarginLoss`	Creates a criterion that optimizes a multi-class multi-classification hinge loss (margin-based loss) between input x (a 2D mini-batch *Tensor*) and output y (which is a 2D *Tensor* of target class indices).
`nn.HuberLoss`	Creates a criterion that uses a squared term if the absolute element-wise error falls below delta and a delta-scaled L1 term otherwise.
`nn.SmoothL1Loss`	Creates a criterion that uses a squared term if the absolute element-wise error falls below beta and an L1 term otherwise.
`nn.SoftMarginLoss`	Creates a criterion that optimizes a two-class classification logistic loss between input tensor x and target tensor y (containing 1 or -1).
`nn.MultiLabelSoftMarginLoss`	eates a criterion that optimizes a multi-label one-versus-all is based on max-entropy, between input x and target y of $*(N, C)$.
`nn.CosineEmbeddingLoss`	Creates a criterion that measures the loss given input tensors x_1, x_2 and a *Tensor* label y with values 1 or -1.
`nn.MultiMarginLoss`	Creates a criterion that optimizes a multi-class classification hinge loss (margin-based loss) between input x (a 2D mini-batch *Tensor*) and output y (which is a 1D tensor of target class indices, $0 \leq y \leq \mathrm{x.size}(1) - 1$):
`nn.TripletMarginLoss`	Creates a criterion that measures the triplet loss given an input tensors $x1$, $x2$, $x3$ and a margin with a value greater than 0.
`nn.TripletMarginWithDistanceLoss`	Creates a criterion that measures the triplet loss given input tensors a, p, and n (representing anchor, positive, and negative examples, respectively), and a nonnegative, real-valued function ("distance function") used to compute the relationship between the anchor and positive example ("positive distance") and the anchor and negative example ("negative distance").

2. 데이터 형식에 따른 손실함수

❖ 다양한 손실함수

- 직접 수정하거나 제작하여 사용



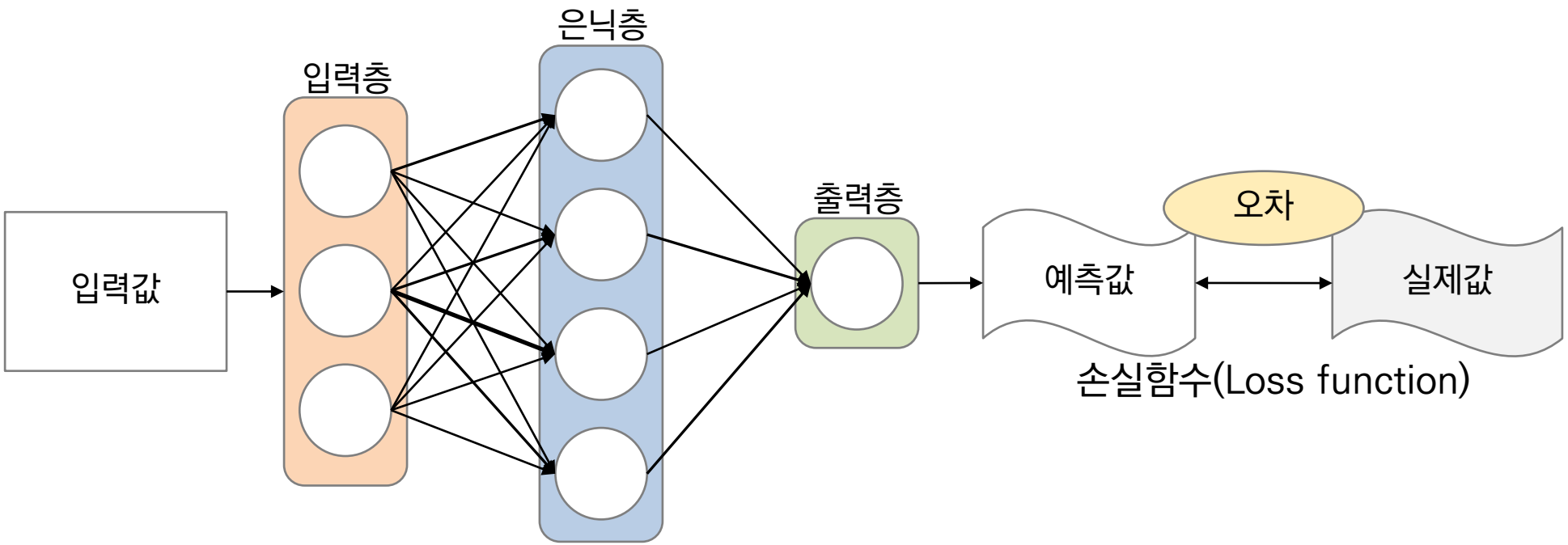
$$L(\omega|\hat{g}, I_t, I_{t,i}) = \frac{1}{N}\sum_{i=1}^{N}\frac{T_i}{T}\left\|\frac{1}{T_i}\sum_{t\in T_i} M_{t+1}R^e_{t+1,i}\hat{g}(\overset{h^g}{\cancel{I_t}}, I_{t,i})\right\|^2.$$

https://discuss.pytorch.org/t/multiple-nn-with-one-custom-loss-function/86862

3. 최고의 손실함수 탐색

❖ 손실함수

- 오차를 최소한으로 줄이는 매개변수를 찾는 것이 학습의 목표
- 오차를 도출하기 위한 식: 손실함수(Loss Function)
- 대표적 손실함수 분류: 교차 엔트로피(Cross Entropy), 회귀 분석: 평균제곱오차(MSE)



3. 최고의 손실함수 탐색

❖ 손실함수의 중요성

- 저해상도 이미지를 고해상도 이미지로 변환하는 Super Resolution 분야
- 평균제곱오차(MSE)를 주로 이용하다가 한계에 부딪혀 다양한 손실함수를 이용하게 됨

Super Resolution 예시
https://hoya012.github.io/blog/SIngle-Image-Super-Resolution-Overview/

3. 최고의 손실함수 탐색

❖ 경험적 탐색

데이터, 분야에 따라 자주 쓰이는 Loss function을 사용

3. 최고의 손실함수 탐색

❖ 경험적 탐색

데이터, 분야에 따라 자주 쓰이는 Loss function을 사용

제공되는 Loss function을 사용

3. 최고의 손실함수 탐색

❖ 경험적 탐색

데이터, 분야에 따라 자주 쓰이는 Loss function을 사용

제공되는 Loss function을 사용

제공되는 Loss function을 수정하여 사용

3. 최고의 손실함수 탐색

❖ 경험적 탐색

데이터, 분야에 따라 자주 쓰이는 Loss function을 사용

제공되는 Loss function을 사용

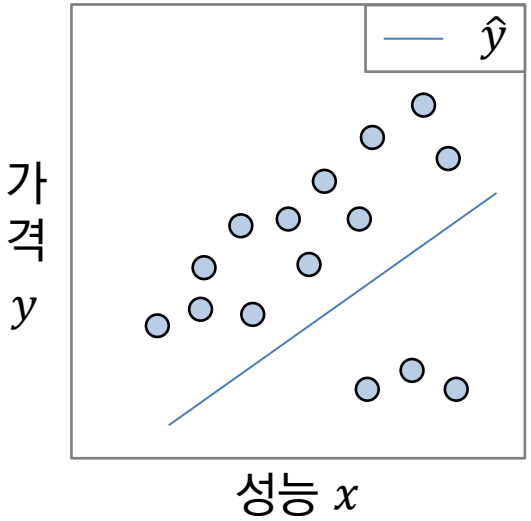
제공되는 Loss function을 수정하여 사용

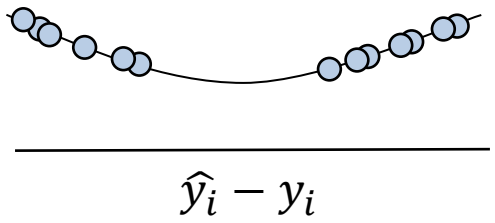
직접 Loss function을 제작하여 사용

3. 최고의 손실함수 탐색

❖ A General and Adaptive Robust Loss Function

- 데이터 분포에 따라 적응하여 손실함수가 변화하는 방식

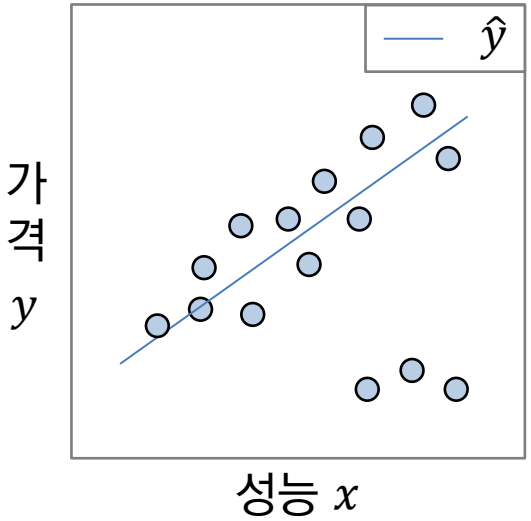


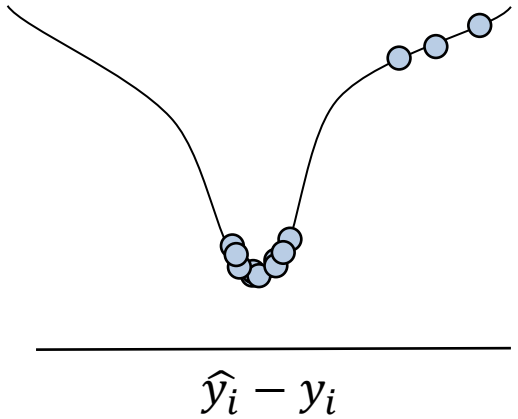


3. 최고의 손실함수 탐색

❖ A General and Adaptive Robust Loss Function

- 이상치를 거르고 중요한 데이터에 맞추어 학습할 수 있도록 함





3. 최고의 손실함수 탐색

❖ A General and Adaptive Robust Loss Function

- 영상에서 깊이를 추정하는 모델의 성능을 높인 예시

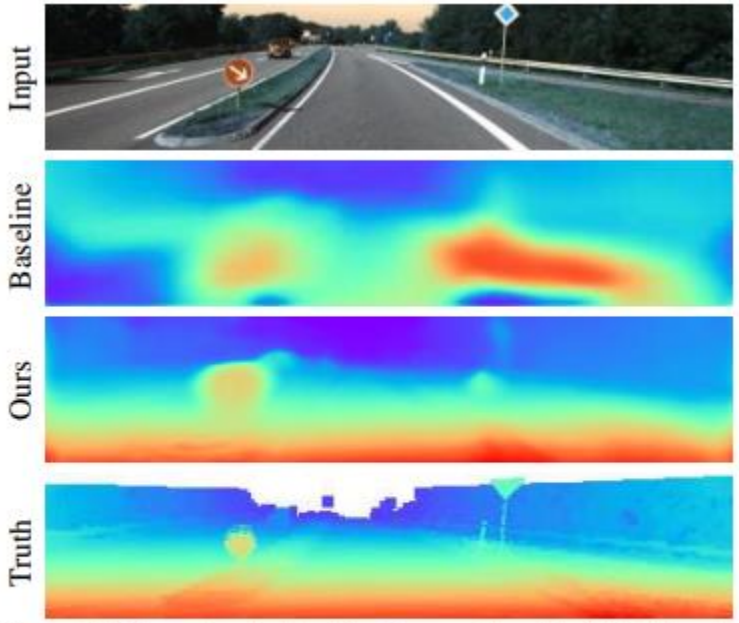
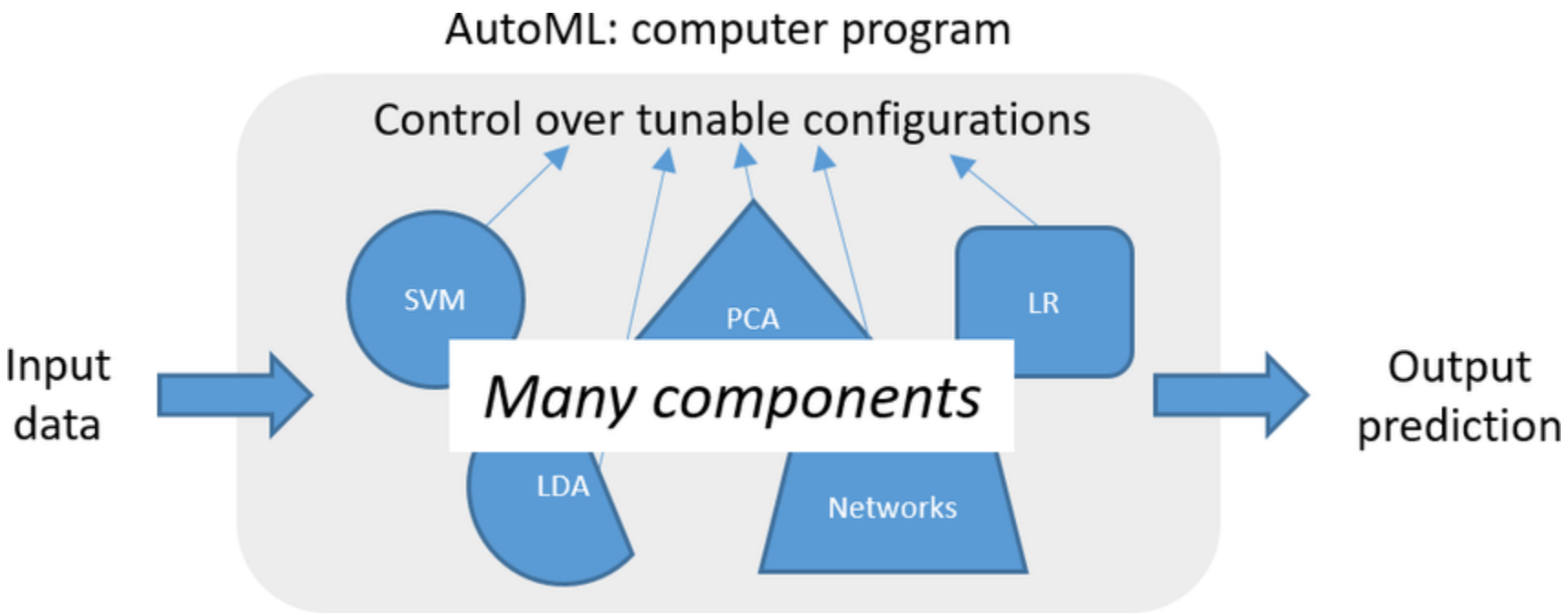


Figure 4. Monocular depth estimation results on the KITTI benchmark using the "Baseline" network of [41]. Replacing only the network's loss function with our "adaptive" loss over wavelet coefficients results in significantly improved depth estimates.

3. 최고의 손실함수 탐색

❖ AM-LFS: AutoML for Loss Function Search

- 강화학습을 이용하여 학습과정에서 최적의 손실함수를 찾는 방식



3. 최고의 손실함수 탐색

❖ AM-LFS: AutoML for Loss Function Search

- 강화학습을 이용하여 학습과정에서 최적의 손실함수를 찾는 방식
- 옵티마이저는 Adam으로 고정 후 초기화 된 모델과 분포를 통해 최적의 모델을 도출

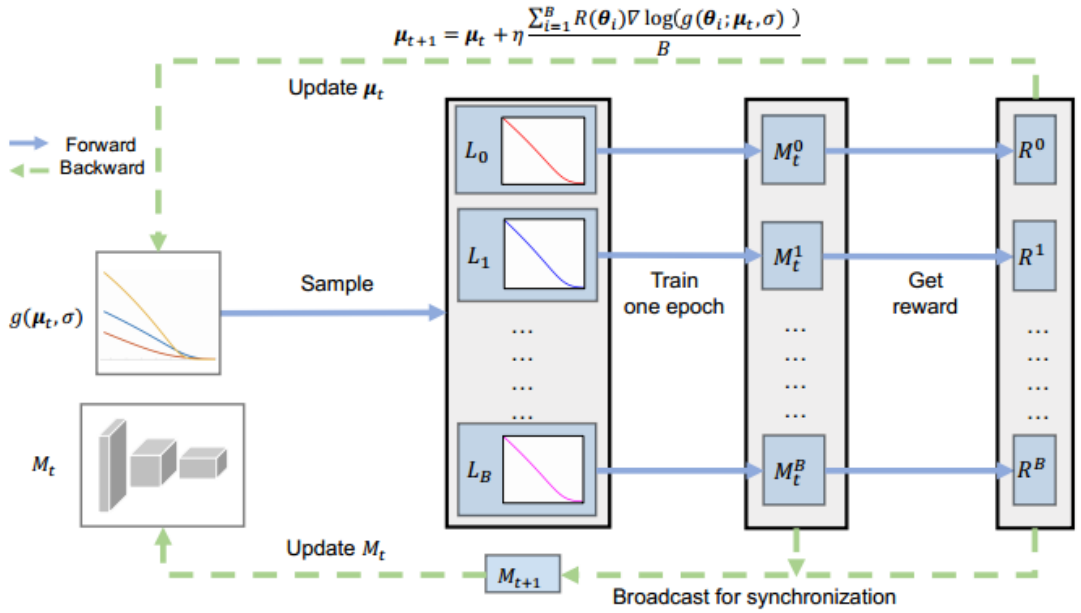


Figure 2. The bilevel optimization framework of our proposed AM-LFS approach. In this bilevel setting, the inner objective is the minimization of sampled loss w.r.t network parameters, while the outer objective is the maximization of rewards (e.g. accuracy or mAP) w.r.t loss function distribution. After each train epoch, we broadcast the model parameters with the highest reward to each sample for synchronization.

Algorithm 1 :AM-LFS

Input: Initialized model M_{ω_0}, initialized distribution μ_0, total training epochs T, distribution learning rate η

Output: Final model M_{ω_T}
for $t = 1$ **to** T **do**
  Sample B hyper-parameters $\theta_1, ...\theta_B$ via distribution $\mathcal{N}(\mu_t, \sigma I)$;
  Train the model M_{ω_t} for one epoch separately with the sampled hyper-parameters and get $M_{\omega_t^1}, ...M_{\omega_t^B}$;
  Calculate the score $R(\theta_1), ...R(\theta_B)$
  Decide the index of model with highest score $i = \arg\max_j R(\theta_j)$;
  Update μ_{t+1} using Eq. (20)
  Update $M_{\omega_{t+1}} = M_{\omega_t^i}$
end for
return M_{ω_T}

3. 최고의 손실함수 탐색

❖ AM-LFS: AutoML for Loss Function Search

- 강화학습을 이용하여 학습과정에서 최적의 손실함수를 찾는 방식
- 옵티마이저는 Adam으로 고정 후 초기화 된 모델과 분포를 통해 최적의 모델을 도출

Table 2. Results on the dataset CIFAR-10 using ResNet-20, showing the ratio of noise label, the top-1 test error rate (%) with standard cross entropy, L2T-DLF and AM-LFS.

Noise ratio(%)	CE [4]	L2T-DLF [41]	AM-LFS
0	8.75	7.63	**6.92**
10	12.05	—	**10.05**
20	15.05	—	**12.73**

Table 3. Comparison with state-of-the-art loss functions on the MegaFace dataset using ResNet-20. For SphereFace and CosineFace, we directly reported the results from the original paper. For ArcFace, we report their results by running the source codes provided by their respective authors to train the models by ourselves following the same setting with CosineFace.

Method	MegaFace rank1@1e6
SphereFace [22]	67.4
CosineFace [37]	72.5
ArcFace [5]	72.4
AM-LFS	**73.5**

Table 4. Comparison with state-of-the-art methods on the Market-1501 dataset using ResNet 50 showing mAP, rank 1 and rank 5. RK refers to implementing re-ranking operation.

Methods	mAP	rank1	rank5
MLFN [2]	74.3	90.0	—
HA-CNN [18]	75.7	91.2	—
DuATM [32]	76.6	91.4	97.1
Part-aligned [34]	79.6	91.7	96.9
PCB [35]	77.4	92.3	97.2
SphereReID [6]	83.6	94.4	98.0
SFT [24]	82.7	93.4	—
MGN [38]	86.9	95.7	—
MGN(RK) [38]	94.2	**96.6**	—
SphereReID+ours	84.4	95.0	98.1
SFT+ours	83.2	93.6	97.9
MGN+ours	88.1	95.8	**98.4**
MGN(RK)+ours	**94.6**	96.1	**98.4**

Table 5. Comparison with state-of-the-art methods on the DukeMTMC-ReID dataset using ResNet 50 showing mAP, rank 1 and rank 5. RK refers to implementing re-ranking operation.

Methods	mAP	rank1	rank5
PSE [30]	62.0	79.8	89.7
MLFN [2]	62.8	81.0	—
HA-CNN [18]	63.8	80.5	—
DuATM [32]	64.6	81.8	90.2
Part-aligned [34]	69.3	84.4	92.2
PCB+RPP [35]	69.2	83.3	—
ShpereReID [6]	68.5	83.9	90.6
SFT [24]	73.2	86.9	93.9
MGN [38]	78.4	88.7	—
MGN(RK) [38]	88.6	90.9	—
ShpereReID+ours	69.8	84.3	92.0
SFT+ours	73.8	87.0	95.1
MGN+ours	80.0	89.9	95.2
MGN(RK)+ours	**90.1**	**92.4**	**95.7**

감사합니다.