딥러닝 기본기 다지기

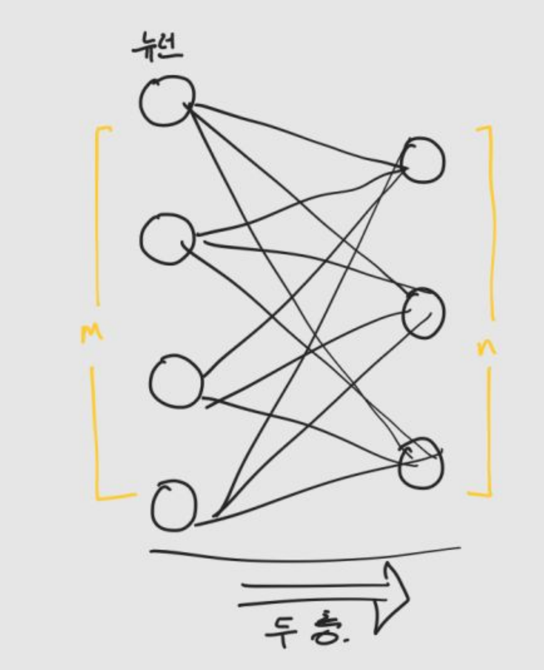
SMARCLE 2022 AI STUDY 1팀

21 노지민
19 오승현
20 김지은
21 김민솔

Table of Contents

01
모델 설계하기

순전파



$$W = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ W_{m1} & W_{m2} & \cdots & W_{mn} \end{bmatrix}$$

$$\vec{x} = (x_1, x_2, \cdots x_m) \quad \vec{b} = (b_1, b_2, \cdots b_n)$$

$$\vec{y} = (y_1, y_2, \cdots y_n)$$

$$\vec{x}W = (x_1, x_2, \cdots x_m) \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix}$$

$$= \left(\sum_{k=1}^{m} x_k w_{k1}, \sum_{k=1}^{m} x_k w_{k2}, \cdots \sum_{k=1}^{m} x_k w_{kn} \right)$$

$$\vec{y} = \vec{x}W + \vec{b}$$

$$= (x_1, x_2, \cdots x_m) \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix} + (b_1, b_2, \cdots b_n)$$

$$= \left(\sum_{k=1}^{m} x_k w_{k1} + b_1, \sum_{k=1}^{m} x_k w_{k2} + b_2, \cdots \sum_{k=1}^{m} x_k w_{kn} + b_n \right)$$

$$\vec{y} = (y_1, y_2 \cdots y_n)$$

$$= f(\vec{u})$$

$$= f(\vec{x}w + b)$$

$$= f\left(\sum_{k=1}^{m} x_k w_{k1} + b_1\right), f\left(\sum_{k=1}^{m} x_k w_{k2} + b_2\right), \cdots, f\left(\sum_{k=1}^{m} x_k w_{kn} + b_n\right)$$

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ \vdots & \ddots & & \\ x_{h1} & \cdots & & x_{hm} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1m} \\ \vdots & \ddots & & \vdots \\ y_{h1} & \cdots & & y_{hn} \end{pmatrix}$$

$$B = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ b_1 & b_2 & \cdots & b_n \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \cdots & b_n \end{pmatrix}$$

$$U = XW + B$$

$$Y = f(U)$$

$$\Rightarrow \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ \vdots & \ddots & & \\ x_{h1} & \cdots & & x_{hm} \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ \vdots & \ddots & & \vdots \\ w_{m1} & \cdots & & w_{mn} \end{pmatrix} + \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ b_1 & b_2 & \cdots & b_n \\ \vdots & \vdots & \ddots & \vdots \\ b_1 & b_2 & \cdots & b_n \end{pmatrix}$$

$$Y = f(U)$$

$$= \begin{pmatrix} f(u_{11}) & f(u_{12}) & \cdots & f(u_{1n}) \\ \vdots & \ddots & & \vdots \\ f(u_{h1}) & f(u_{h2}) & \cdots & f(u_{hn}) \end{pmatrix}$$

역전파

$$U = XW + B$$

$$Y = f(U)$$

$$u = \sum_{k=1}^{m} x_k w_k + b, \quad y = f(u)$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial w_i}$$

$$\delta = \frac{\partial E}{\partial u} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial u}$$

$$\frac{\partial u}{\partial w_i} = \frac{\partial\left(\sum_{k=1}^{m} x_k w_k + b\right)}{\partial w_i}$$

$$= \frac{\partial}{\partial w_i}(x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b)$$

$$= x_i$$

$$\delta = \frac{\partial E}{\partial u} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial u}$$

$$\frac{\partial E}{\partial w_i} = x_i \delta \qquad \frac{\partial E}{\partial b} = \delta$$

$$\frac{\partial E}{\partial x_i} = \sum_{k=1}^{n} \frac{\partial E}{\partial u_k}\frac{\partial u_k}{\partial x_i}$$

$$= \sum_{k=1}^{n} w_{ik}\delta_k$$

$$\frac{\partial E}{\partial w_i} = x_i\delta, \quad \frac{\partial E}{\partial b} = \delta, \quad \frac{\partial E}{\partial x_i} = \sum_{k=1}^{n} w_{ik}\delta_k$$

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{h} \overset{=1}{\boxed{\frac{\partial E}{\partial E_k}}} \frac{\partial E_k}{\partial w_i}$$

$$= \sum_{k=1}^{h} \frac{\partial E_k}{\partial w_i}$$

$$\frac{\partial E}{\partial W} = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{h1} \\ \vdots & \ddots & & \vdots \\ x_{1m} & & \cdots & x_{hm} \end{pmatrix} \begin{pmatrix} \delta_{11} & \cdots & \delta_{1n} \\ \vdots & \ddots & \vdots \\ \delta_{h1} & \cdots & \delta_{hn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{k=1}^{h} x_{k1}\delta_{k1} & \cdots & \sum_{k=1}^{h} x_{k1}\delta_{kn} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{h} x_{km}\delta_{k1} & \cdots & \sum_{k=1}^{h} x_{km}\delta_{kn} \end{pmatrix}$$

$$\frac{\partial E}{\partial B} = \begin{pmatrix} \sum_{k=1}^{h} \delta_{k1} & \cdots & \sum_{k=1}^{h} \delta_{kn} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{h} \delta_{k1} & \cdots & \sum_{k=1}^{h} \delta_{kn} \end{pmatrix}$$

$$\frac{\partial E}{\partial X} = \begin{pmatrix} \delta_{11} & \cdots & \delta_{1n} \\ \vdots & \ddots & \vdots \\ \delta_{h1} & \cdots & \delta_{hn} \end{pmatrix} \begin{pmatrix} w_{11} & \cdots & w_{m1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{mn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{k=1}^{n} w_{1k}\delta_{1k} & \cdots & \sum_{k=1}^{n} w_{mk}\delta_{1k} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} w_{1k}\delta_{hk} & \cdots & \sum_{k=1}^{n} w_{mk}\delta_{hk} \end{pmatrix}$$

2. 오차 함수

- `KLD(...)`: Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.
- `MAE(...)`: Computes the mean absolute error between labels and predictions.
- `MAPE(...)`: Computes the mean absolute percentage error between `y_true` and `y_pred`.
- `MSE(...)`: Computes the mean squared error between labels and predictions.
- `MSLE(...)`: Computes the mean squared logarithmic error between `y_true` and `y_pred`.
- `binary_crossentropy(...)`: Computes the binary crossentropy loss.
- `categorical_crossentropy(...)`: Computes the categorical crossentropy loss.
- `categorical_hinge(...)`: Computes the categorical hinge loss between `y_true` and `y_pred`.
- `cosine_similarity(...)`: Computes the cosine similarity between labels and predictions.
- `deserialize(...)`: Deserializes a serialized loss class/function instance.
- `get(...)`: Retrieves a Keras loss as a `function` / `Loss` class instance.
- `hinge(...)`: Computes the hinge loss between `y_true` and `y_pred`.
- `huber(...)`: Computes Huber loss value.
- `kl_divergence(...)`: Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.
- `kld(...)`: Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.
- `kullback_leibler_divergence(...)`: Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.
- `log_cosh(...)`: Logarithm of the hyperbolic cosine of the prediction error.
- `logcosh(...)`: Logarithm of the hyperbolic cosine of the prediction error.
- `mae(...)`: Computes the mean absolute error between labels and predictions.
- `mape(...)`: Computes the mean absolute percentage error between `y_true` and `y_pred`.
- `mean_absolute_error(...)`: Computes the mean absolute error between labels and predictions.
- `mean_absolute_percentage_error(...)`: Computes the mean absolute percentage error between `y_true` and `y_pred`.
- `mean_squared_error(...)`: Computes the mean squared error between labels and predictions.
- `mean_squared_logarithmic_error(...)`: Computes the mean squared logarithmic error between `y_true` and `y_pred`.

Classes

- `class BinaryCrossentropy`: Computes the cross-entropy loss between true labels and predicted labels.
- `class CategoricalCrossentropy`: Computes the crossentropy loss between the labels and predictions.
- `class CategoricalHinge`: Computes the categorical hinge loss between `y_true` and `y_pred`.
- `class CosineSimilarity`: Computes the cosine similarity between labels and predictions.
- `class Hinge`: Computes the hinge loss between `y_true` and `y_pred`.
- `class Huber`: Computes the Huber loss between `y_true` and `y_pred`.
- `class KLDivergence`: Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.
- `class LogCosh`: Computes the logarithm of the hyperbolic cosine of the prediction error.
- `class Loss`: Loss base class.
- `class MeanAbsoluteError`: Computes the mean of absolute difference between labels and predictions.
- `class MeanAbsolutePercentageError`: Computes the mean absolute percentage error between `y_true` and `y_pred`.
- `class MeanSquaredError`: Computes the mean of squares of errors between labels and predictions.
- `class MeanSquaredLogarithmicError`: Computes the mean squared logarithmic error between `y_true` and `y_pred`.
- `class Poisson`: Computes the Poisson loss between `y_true` and `y_pred`.
- `class Reduction`: Types of loss reduction.
- `class SparseCategoricalCrossentropy`: Computes the crossentropy loss between the labels and predictions.
- `class SquaredHinge`: Computes the squared hinge loss between `y_true` and `y_pred`.

이미지 출처 :https://www.tensorflow.org/api_docs/python/tf/keras/losses

02
10장 실습

03

11장 : 데이터 다루기

피마 인디언의 당뇨병 예측하기

1. 딥러닝과 데이터

성공적인 딥러닝 프로젝트를 위해서는

"데이터가 얼마나 효율적으로 사용되게끔 가공됐는지"

2. 피마 인디언 데이터 분석하기

- 샘플 수: 768
- 속성: 8
  - 정보 1 (pregnant): 과거 임신 횟수
  - 정보 2 (plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
  - 정보 3 (pressure): 확장기 혈압(mm Hg)
  - 정보 4 (thickness): 삼두근 피부 주름 두께(mm)
  - 정보 5 (insulin): 혈청 인슐린(2-hour, mu U/ml)
  - 정보 6 (BMI): 체질량 지수(BMI, weight in kg/(height in m)^2)
  - 정보 7 (pedigree): 당뇨병 가족력
  - 정보 8 (age): 나이
- 클래스: 당뇨(1), 당뇨 아님(0)

딥러닝을 구동하려면 속성과 클래스를 반드시 먼저 구분!

2. 피마 인디언 데이터 분석하기

데이터 조사 → 데이터 가공 → 시각화

2. 피마 인디언 데이터 분석하기

https://colab.research.google.com/drive/1WuxeFjYkwtBz1CgsgYTaykJYBqiL8qI-?usp=sharing

Thanks

1팀
21 노지민
19 오승현
20 김지은
21 김민솔