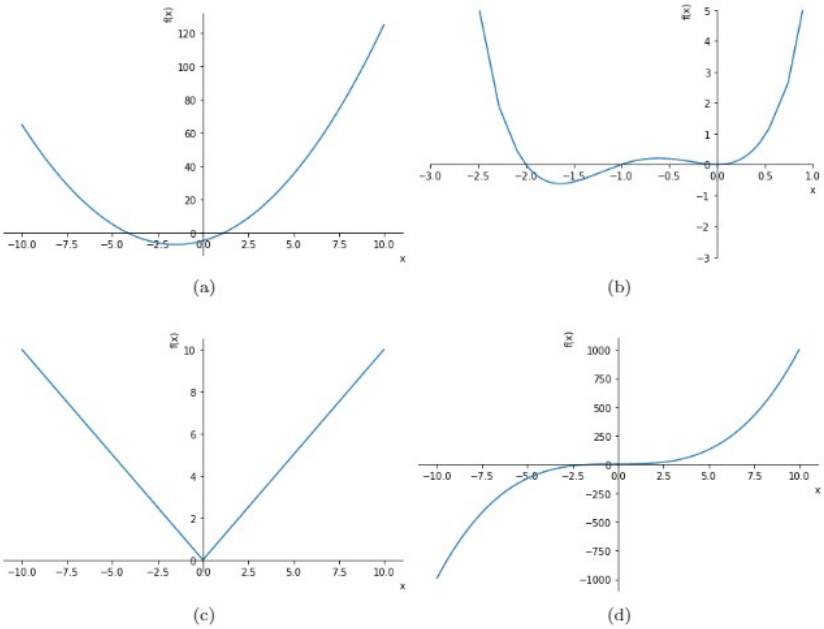
AI기술 자연어 처리 전문가 양성 과정 1기
Machine Learning Quiz

1. 다음 명제에 대해, True / False 를 판단하시오.
   (a) Linear regression 문제에서 input 변수가 많다면 linear regression으로 해결할 수 없다.False
   (b) Logistic regression의 output은 모든 실수 범위이다. False
   (c) Training data 에서의 accuracy가 가장 높은 model 이 가장 좋은 model이다. False
   (d) K fold cross validation은 총 K번 평가를 진행해야 한다. True
   (e) Principle component anlaysis 는 축소 이전과 축소 이후 거리를 최대한 보존하는 방식의 dimensionality reduction 방법이다. False
   (f) K-means clustering 알고리즘은 항상 optimal 하게 작동한다. False
   (g) x_i, y_i가 각각 데이터이고, $w_1, w_2, ...w_k$가 각각 linear layer일 때 $y_i = w_k(w_{k-1}(...w_2(w_1(x_i))..)$ 모델은 non-linear 한 관계도 표현할 수 있다. False

2. 다음 중 linear equation 이 아닌 것을 모두 고르시오.
   (a) $3x + 6 = y$
   (b) $\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = c$
   (c) $a\sqrt{x} + b = y$
   (d) $ax_1 + bx_2 + cx_3 + dx_4 = e$
   (e) $ax^2 + bx^{10} + cx^4 = d$

   c, e

3. 다음 중 gradient descent 로 항상 최솟값에 도달할 수 있는 그래프를 모두 고르시오.



   a

4. Linear regression과 Logistic regression에서의 cost function을 적으시오. (하나만 적으시면 됩니다.)
Linear regression : MSE loss $(y_{\text{true}} - \hat{y})^2$, Logistic regression : - $[y_{\text{true}} \log(\hat{y}) + (1 - y_{\text{true}}) \log(1 - \hat{y})]$

5. Logistic function 을 미분한 결과를 logistic function으로 나타내시오.

$$\text{logistic function} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial \sigma(z)}{\partial z} = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \times \frac{e^{-z}}{1 + e^{-z}} = \sigma(z) \times \frac{e^{-z}}{1 + e^{-z}}$$

$$\frac{e^{-z}}{1 + e^{-z}} = \frac{1 + e^{-z} - 1}{1 + e^{-z}} = 1 - \frac{1}{1 + e^{-z}} = 1 - \sigma(z)$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

6. Overfitting 과 underfitting 을 bias, variance 관점에서 서술하고, 각각을 방지하기 위한 방법을 서술하시오.
overfitting은 high variance, low bias이고, Underfitting는 high bias, low variance이다.
overfitting을 막는 방법은 regulariation term 을 추가해서 우리의 weight 값이 너무 커지는 것을 방지해줍니다. underfitting을 막는 방법은 모델이 너무 간단하기 때문에 일어나는 일이므로 모델의 차수를 올려주거나 layer를 더 추가해줍니다.

7. Dataset을 train, validation, test로 나누는 이유에 대해서 설명하시오.
우리는 모델을 학습할 때 단순히 주어진 dataset에서만 잘 작동하는 모델을 만드는 것이 목적이 아닙니다. 주어진 dataset에서 잘 작동하고, 그리고 아직 보지 못한 dataset에서도 잘 작동하는 모델을 만드는 것이 목적입니다. 그래서 우리는 dataset을 train, validation, test으로 나누게 됩니다. 학습을 할 때 train으로 학습을 하고 학습을 할 때 못본 validation set으로 얼마나 우리가 못 본 데이터 셋에서 잘 작동하는지 확인을 하면서 parameter 조정을 합니다. Validation도 parameter 서칭 등 학습에 관여를 하였기 때문에 그것으로 최종 성능을 측정하는 것은 정말 보지 못한 데이터셋으로 성능을 측정하였다고 할 수 없습니다. 그래서 test set으로 정말 마지막 우리 모델의 성능을 측정합니다.

8. Supervised learning 과 unsupervised learning 의 차이에 대해서 서술하시오.
Ground truth 인 label 이 존재해서 학습할 때 사용할 수 있다면 supervised learning, label이 없어서 input data만을 가지고 학습을 해야 한다면 unsupervised learning 이다.

9. K-means clustering에서 k 값을 찾는 방법에 대해서 서술하시오. 여러개의 K 값을 가지고 실험을 모두 진행한 뒤, 우리가 정한 cost function의 값의 변화를 관찰한다. k 값이 커짐에 따라서 cost function 의 값이 감소하는데, 급격한 감소가 없어지고 감소의 폭이 작아지기 시작하는 부분을 보통 K값으로 선택하고 이를 elbow method 라 한다.

10. Neural Network 에서 non-linear function 이 필요한 이유에 대해서 설명하시오. 단순히 linear layer 를 여러개 쌓는다면 그것은 결국 matrix multiplication을 통해서 하나의 linear layer가 되기 때문에 더 복잡한 함수를 모델링 해주기 위해서는 linear layer 의 사이사이에 non-linear function을 추가해 주어야 한다.

11. 다음 4가지 단어에 대해 설명하고, accuracy, precision, recall을 각각 그 값으로 나타내시오.

    (a) True Positive (TP) : 결함이 있다고 예측한 것들 중 실제로 결함이 있는 것
    (b) False Positive (FP) :결함이 있다고 예측한 것들 중 실제로 결함이 없는 것
    (c) True Negative (TN) :결함이 없다고 예측한 것들 중 실제로 결함이 없는 것
    (d) False Negative (FN) :결함이 없다고 예측한 것들 중 실제로 결함이 있는 것

$$\text{accuracy} = \frac{\text{정확하게 예측한 개수}}{\text{예측한 전체 개수}} \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FN} + N_{FP} + N_{TN}}$$

$$\text{precision} = \frac{\text{결함이 있다고 정확하게 예측한 개수}}{\text{결함이 있다고 예측한 총 개수}} = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

$$\text{recall} = \frac{\text{결함이 있다고 정확하게 예측한 개수}}{\text{실제로 결함이 있는 개수}} = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

12. 다음 코드를 보고 빈칸을 채우시오.

```
def logistic(x):
    """Logistic/sigmoid function.

    Arguments
    ---------
    x : numpy.ndarray
        The input to the logistic function.

    Returns
    -------
    numpy.ndarray
        The output.

    Notes
    -----
    The function does not restrict the shape of the input array. The output
    has the same shape as the input.
    """
    out = 1. / (1. + numpy.exp(-x))

    return out
```

```
def logistic_model(x, params):
    """A logistic regression model.

    A a logistic regression is y = sigmoid(x * w + b), where the operator *
    denotes a mat-vec multiplication.

    Arguments
    ---------
    x : numpy.ndarray
        The input of the model. The shape should be (n_images, n_total_pixels).
    params : a tuple/list of two elemets
        The first element is a 1D array with shape (n_total_pixels). The
        second element is a scalar (the intercept)

    Returns
    -------
    probabilities : numpy.ndarray
        The output is a 1D array with length n_samples.
    """

    out = logistic(numpy.dot(x, params[0]) + params[1])

    return out
```

```
def model_loss(x, true_labels, params, _lambda=1.0):
    """Calculate the predictions and the loss w.r.t. the true values.

    Arguments
    ---------
    x : numpy.ndarray
        The input of the model. The shape should be (n_images, n_total_pixels).
    true_labels : numpy.ndarray
        The true labels of the input images. Should be 1D and have length of
        n_images.
    params : a tuple/list of two elements
        The first element is a 1D array with shape (n_total_pixels). The
        second elenment is a scalar.
    _lambda : float
```

```
        The weight of the regularization term. Default: 1.0

    Returns
    -------
    loss : a scalar
        The summed loss.
    """
    pred = logistic_model(x, params)

    loss = - (
        numpy.dot(true_labels, numpy.log(pred+1e-15)) +
        numpy.dot(1.-true_labels, numpy.log(1.-pred+1e-15))
    ) + _lambda * numpy.sum(params[0]**2)

    return loss
```

```
def MLP_model(x, params):
    """ A MLP model.

    A MLP is y = sigmoid(max((x * w1 + b1), 0) *w2 +b2), where the operator *
    denotes a mat-vec multiplication.
    Arguments
    ---------
    x : numpy.ndarray
        The input of the model. The shape should be (n_images, n_total_pixels).
    params : a tuple/list of four elemets
        The first element is a 1D array with shape (n_total_pixels). The
        second element is a scalar (the intercept)

    Returns
    -------
    probabilities : numpy.ndarray
        The output is a 1D array with length n_samples.
    """
    x = numpy.dot(x, params[0]) + params[1]
    x = numpy.maximum(x, 0)

    return logistic(numpy.dot(x, params[2]) + params[3])
```