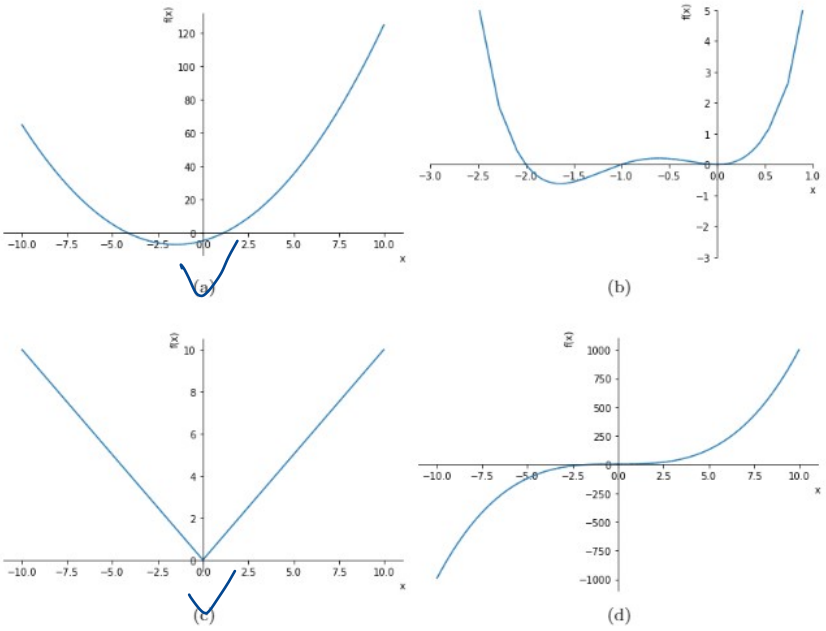
AI기술 자연어 처리 전문가 양성 과정 1기
Machine Learning Quiz

1. 다음 명제에 대해, True / False 를 판단하시오.
   (a) Linear regression 문제에서 input 변수가 많다면 linear regression으로 해결할 수 없다. (T)
   (b) Logistic regression의 output은 모든 실수 범위이다. (T)
   (c) Training data 에서의 accuracy가 가장 높은 model 이 가장 좋은 model이다. (F)
   (d) K fold cross validation은 총 K번 평가를 진행해야 한다. (T)
   (e) Principle component anlaysis 는 축소 이전과 축소 이후 거리를 최대한 보존하는 방식의 dimensionality reduction 방법이다. (T)
   (f) K-means clustering 알고리즘은 항상 optimal 하게 작동한다. (F)
   (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. (T)

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) $ax^2 + bx^{10} + cx^4 = d$

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



4. Linear regression과 Logistic regression에서의 cost function을 적으시오. (하나만 적으시면 됩니다.)

$$\frac{1}{1+e^{-z}}$$

5. Logistic function 을 미분한 결과를 logistic function으로 나타내시오.

6. Overfitting 과 underfitting 을 bias, variance 관점에서 서술하고, 각각을 방지하기 위한 방법을 서술하시오.

7. Dataset을 train, validation, test로 나누는 이유에 대해서 설명하시오.

8. Supervised learning 과 unsupervised learning 의 차이에 대해서 서술하시오.

9. K-means clustering에서 k 값을 찾는 방법에 대해서 서술하시오.

10. Neural Network 에서 non-linear function 이 필요한 이유에 대해서 설명하시오.

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

    (a) True Positive (TP) :

    (b) False Positive (FP) :

    (c) True Negative (TN) :

    (d) False Negative (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 = _____

    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)))

    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])
```