

On Regression

after all, what is regression

Regression: some definition/intuition/explanation/examples...

a statistical method used to model the relationship between
one (or more) independent variables & one (or more) dependent variable

能为一个或多个自变量与因变量之间关系建模的一类方法
(输入、输出之间的关系)

Regression finds "best fit" line or equation to describe how the
dependent variable changes as independent variables change

...

but does the word "regression" have anything to do with the tasks mentioned above?

NOTHING!

"回归"之所以叫"回归"是因为历史原因。英文"regression"本意是"倒退、回到原来的状态"。这个名称源于1880s英国统计学家高尔顿(Francis Galton)的发现：他研究父母与子女身高关系时发现，极端高高的父母，其子女身高会“回归”到人群平均水平附近。高尔顿称此现象为"regression toward mediocrity"（向平庸回归）。高尔顿用"regression"命名时，关注的是数据向均值回归的现象，后来发现这种统计关系可以用来做预测，但“回归”这个名称已经固定下来了……

takeaway: regression (回归) 关心的是 input (输入) 和 output (输出) 之间的映射

Linear Regression 线性回归

1. Representation (模型定义..)

$$(\text{Training}) \quad \text{Dataset} \quad \vec{X} = \begin{bmatrix} x_1^{(1)} & \dots & x_i^{(1)} & \dots & x_d^{(1)} \\ \vdots & & \vdots & & \vdots \\ x_1^{(n)} & \dots & x_i^{(n)} & \dots & x_d^{(n)} \end{bmatrix}_{n \times d} \quad i\text{-th feature} \quad j\text{-th sample} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Assumptions: 1. \vec{x}, y 之间的关系是线性的 (除去观测值的噪声)

2. 观测值噪声比较正常，如服从正态分布

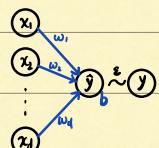
even if we believe that the best model for predicting y using \vec{x} is linear, we would not expect to find a real-world dataset of n examples where $y^{(i)}$ exactly equals $w^T \vec{x}^{(i)} + b$, $i=1, \dots, n$. thus, we will incorporate a noise term ε to account for errors

for a single sample point $\vec{x} = [x_1, \dots, x_d]^T$, the linear model:

$$\hat{y} = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b = [w_1, \dots, w_d] \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + b = \vec{w}^T \cdot \vec{x} + b$$

for X :

$$\vec{y} = \vec{X} \vec{w} + b \quad (\text{||}_\text{row} + b, \text{broadcasting})$$



* include (Gaussian) noise term: $y = \hat{y} + \varepsilon = \vec{w}^T \vec{x} + b + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2)$

★ now, our goal is searching for the best parameters \vec{w} and b

2. Learning

remember one thing "regression" cares is fitness (模型对数据的拟合程度)

Loss functions measures/quantify the distance between real/ground truth and predicted/model output values.

for regression problem, the most common loss function is mean squared error (MSE). given by:

$$L(\vec{w}, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (\hat{y}^{(i)} - y^{(i)})^2$$

对训练数据平均。MSE

$\Rightarrow \vec{w}^*, b^* = \arg \min_{\vec{w}, b} L(\vec{w}, b)$ *在正态噪声假设下，最优参数 \vec{w}^*, b^* 是极大似然估计 (MLE)

至此，模型/参数的学习转化为了优化问题，Learning as optimization

* Closed-form/Analytical Solution: $\vec{w}^* = (X^T X)^{-1} X^T y$ *线性回归模型可直接得到解析解，但并不具有一般意义

一般地，采用基于梯度更新的迭代优化算法解决这类优化问题。*事实上，线性回归解析解涉及多次矩阵乘法和求逆运算，在时间复杂度上要差于迭代优化算法。

[Gradient Descent (GD) Stochastic Gradient Descent (SGD) & Mini-batch SGD]

(这里不再重复算法内容)。

* Gradient of L_{MSE} w.r.t. \vec{w} & b : $L_{MSE} = \frac{1}{2n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2n} \sum_{i=1}^n (\frac{1}{2} \sum_{j=1}^d w_j x_j^{(i)} + b - y^{(i)})^2$

$$\frac{\partial L_{MSE}}{\partial w_j} = \frac{1}{n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)} \Rightarrow \nabla_{\vec{w}} L_{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)}) \vec{x}^{(i)}$$

$$\cdot \frac{\partial L_{MSE}}{\partial b} = \frac{1}{n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})$$

{“逻辑回归”和“逻辑”有关系吗？Logistic, logic 都翻译成了“逻辑”…}

Logistic Regression 逻辑回归

{“逻辑回归”在“回归”什么？
什么是“逻辑(Logits)”？} 在回归 logit (对数几率, log-odds), $\log(\frac{p}{1-p}) \in \mathbb{R}$, $p \in (0,1)$
发生

逻辑回归是分类还是回归？用概率作中转站——回归(预测)概率。
再做基于概率的分类(决策)
.....

1. Representation

i-th feature

(Training) Dataset $X = \begin{bmatrix} x_1^{(1)} & \dots & x_i^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & \dots & x_i^{(2)} & \dots & x_d^{(2)} \\ \vdots & & \vdots & & \vdots \\ x_1^{(n)} & \dots & x_i^{(n)} & \dots & x_d^{(n)} \end{bmatrix}_{n \times d}$ j-th sample $\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$, $y^{(i)} \in \{0,1\}$. 两个类别标签。

Goal: 为了实现“分类”，我们需要在线性回归的基础上做一些改进：

1. 分类中类别变量属于离散空间 (e.g. {正面、反面}, {晴天、阴天}, {清华、北大、人大附} ...)

而线性回归模型的输出/预测/响应变量属于连续空间 (具体而言 \mathbb{R}^d)

这里离散 \leftrightarrow 连续的 gap 是本质的

2. 为了处理这一 gap, 可借助概率 (probability), 将线性回归模型的输出概率化处理, 即如果能让

模型输出一个概率值, 那么我们就可以实现基于概率的分类了! (这样做利用了概率的本质特性)

3. 什么是“合法”的概率，概率分布？

对于离散概率分布“取值非负”，“求和等于1”，
“对于每一个不可能发生的事件，概率均有定义”，就可以是一个概率分布

e.g. 1. $P\{X=0\} = 0.7, P\{X=1\} = 0.3 \rightarrow x$

2. $P\{X=0\} = \alpha_0, P\{X=1\} = \alpha_1, \dots, P\{X=k\} = \alpha_k, \text{ where } \sum_{i=0}^k \alpha_i = 1; \alpha_i \geq 0, i=0, \dots, k-1 \rightarrow$

不难发现 1. 实际上是 2. 的特例，前者为 Bernoulli 分布，后者为 Categorical 分布。

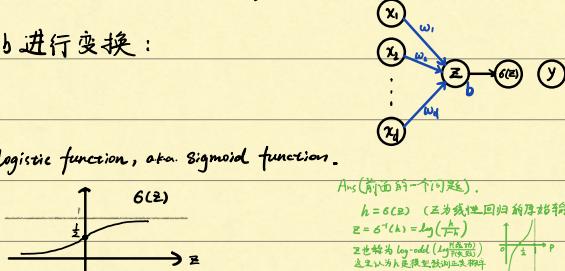
4. 为了让模型输出是概率值，即要满足 非负性和归一化 条件，

我们要对线性回归模型的输出 $\bar{z} = \bar{w}^\top \bar{x} + b$ 进行变换：

$$\sigma(\bar{z}) = \sigma(\bar{w}^\top \bar{x} + b),$$

其中 $\sigma(z) = \frac{1}{1+e^{-z}} \in (0, 1), z \in \mathbb{R}$, 称为 Logistic function, aka. Sigmoid function.

记 $h(\bar{x}; \bar{w}, b) = \sigma(\bar{z})$ 为改进后模型的输出
模型输入 表示前向参数



Volla! With a simple transformation, we can "make" the model output a probability

这里有一些有趣的讨论：上面这种“直接套一个 $\sigma(\cdot)$ ”的操作，似乎很不讲道理——但是又似乎没错（符合概率定义，本质上是一些不同视角）产生这种错觉原因并不在于 $h(\bar{x})$ 是不是概率（它是！）而在于它是谁的概率（2）以及它为什么是一的概率（2）

(1): 它是模型基于输入得到的样本正类的概率。 (e.g. 基于病人特征(输入)，感冒的概率)
 (或反面，取决于定义)

(2): 这个问题关心的是“我们认为它是样本正类的概率，和它“真的是”样本正类的概率这两件事之间的 gap。
 也即“我们认为模型能判断病人是否感冒”和“模型真的能判断病人是否感冒”之间的 gap。
 到这儿，相信我们能意识到，这个问题的答案自然在模型训练/学习之中！

甚至由此可直接引出我们对模型学习目标(损失函数)的构建：

A (概率的角度) 我们希望能缩小模型输出的概率
 (基于病人特征(输入)，得到是感冒的概率)
 与真实概率 ($P\{\text{病人感冒} = 1\}$) 之间的 gap！

B (更直观) 我们希望能缩小模型的判断
 (基于病人特征(输入)，得到是感冒的概率，进而 (e.g.) 是否是感冒)
 与真实情况 (病人是否感冒) 之间的 gap！

如果我们能通过 A/B, 训练模型，那么我们就能减小 gap！

(前面所有 gap, 本质上是一样的)

虽然 B 更加直观 (直接与分类任务相关)，但“通过概率是否 $> 0.5 / P(X=1) \Rightarrow P(X=0) = 1 - P(X=1)$ 得到正类/负类”
 $\Leftrightarrow P(X=1) \neq 0.5$

这一操作是将连续的概率取值 离散化了 (与标签/类别对应)
 而“离散”这件事儿显然不利于后续基于梯度/微分/梯度的训练/学习优化。

这里又可以看出，概率取值的连续性是有利的！

于是从 A 出发，我们引入交叉熵损失 (CE, Cross Entropy)

直观上 交叉熵损失刻画了两个概率分布间的距离。 (A)

(optional) Def: 考虑两个概率分布 p, q , 交叉熵 $H(p, q)$ 定义为

$$H(p, q) = -E_p[\log q] = H(p) + D_{KL}(p||q)$$

H 表示信息熵, $H(p) = E_p[\log p]$; $E[\cdot]$ 表示关于 p 的数学期望; $D_{KL}(\cdot||\cdot)$ 表示 K-L 散度，一种不完备的距离

在离散情况下: $H(p, q) = -\sum_x p(x) \log q(x)$.

交叉熵损失 (二分类; logistic regression) :

Recap & notation: 首先为了优化两个概率分布间的距离, 我们要先给出两个分布的形式: (这里记号有点复杂)

$P\{Y=k \vec{x}, y\} = y \cdot I_{k=1} + (1-y) \cdot I_{k=0}, k \in \{0, 1\}$ (单点/退化分布)	$Q\{Y=k \vec{x}\} = h(\vec{x})^k (1-h(\vec{x}))^{1-k}, k \in \{0, 1\}$ (两点/伯努利)	$H(P, Q) = H(P) + D_{KL}(P Q)$ (因为真实分布 P 的熵是确定的) 最好的情况下, $D_{KL}(P Q) = 0$, 也即 $P = Q$, $H(P, Q) = H(P)$; 即预测分布 = 真实分布
--	--	---

for one sample point (\vec{x}, y) :

$$\begin{aligned}
 L_{CE} &= H(P, Q) = - \sum_{k \in \{0, 1\}} P\{Y=k\} \log [Q\{Y=k\}] \\
 &= - \sum_{k \in \{0, 1\}} P\{Y=k\} \log [h(\vec{x})^k (1-h(\vec{x}))^{1-k}] \\
 &= - P\{Y=1\} \log (h(\vec{x})) - P\{Y=0\} \log (1-h(\vec{x})) \\
 &= -y \log (h(\vec{x})) - (1-y) \log (1-h(\vec{x})) \\
 &= \begin{cases} -\log (1-h(\vec{x})) & \text{if } y=0 \\ -\log (h(\vec{x})) & \text{if } y=1 \end{cases}
 \end{aligned}$$

for a batch of samples $\vec{x}^{(1)}, \dots, \vec{x}^{(m)}$:

$$\begin{aligned}
 L_{CE} &= \frac{1}{m} \sum_{i=1}^m H(P^{(i)}, Q^{(i)}) \\
 &= \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log (h(\vec{x}^{(i)})) - (1-y^{(i)}) \log (1-h(\vec{x}^{(i)}))]
 \end{aligned}$$

* Gradient of L_{CE} w.r.t. \vec{w} & b

$$\frac{\partial L_{CE}}{\partial w_j} = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log (h(\vec{x}^{(i)})) - (1-y^{(i)}) \log (1-h(\vec{x}^{(i)}))]$$

$$= \frac{1}{m} \sum_{i=1}^m (h(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\Rightarrow \nabla_{\vec{w}} L_{CE} = \frac{1}{m} \sum_{i=1}^m (h(\vec{x}^{(i)}) - y^{(i)}) \vec{x}^{(i)}$$

$$\frac{\partial L_{CE}}{\partial b} = \frac{1}{m} \sum_{i=1}^m (h(\vec{x}^{(i)}) - y^{(i)})$$

同样可使用 Mini-batch SGD 优化。(略)

Softmax Regression (Softmax 回归)

Softmax Regression 是 Logistic Regression 的直接推广，任务从二分类推广至多分类

1. Representation

$$\text{(Training) Dataset } X = \begin{bmatrix} x_1^{(1)} & \cdots & x_i^{(1)} & \cdots & x_d^{(1)} \\ \vdots & & \vdots & & \vdots \\ x_1^{(j)} & \cdots & x_i^{(j)} & \cdots & x_d^{(j)} \\ \vdots & & \vdots & & \vdots \\ x_1^{(n)} & \cdots & x_i^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}_{n \times d} \quad \text{j-th sample} \quad \vec{y} = \begin{bmatrix} \vec{y}^{(1)} \\ \vdots \\ \vec{y}^{(j)} \\ \vdots \\ \vec{y}^{(n)} \end{bmatrix}, \quad y^{(j)} = (0, \dots, 1, \dots, 0), \text{ one-hot encoding.}$$

$\dim y^{(j)} = C$ (C类)
 $y_k^{(j)} = 1$ 表示是第 k 类

类似地，Goal：为了实现“多分类”，我们需要在逻辑回归的基础上做一些改进：

从逻辑回归的目标/损失出发，idea仍是优化两个概率分布间的距离，

其中真实分布是由 $y^{(i)}$ (one-hot 向量)诱导出的一个单点/退化分布,与二分类中类似

$$P\{Y=k\} = y_k^{(j)}$$

这就自然要求我们的模型也能输出一个向量(维度为K), 每个分量代表第*j*美的概率.

(当然这就是一个 Categorical 分布：

$$Q\{Y=k\} = \hat{P}_k$$

k	o	\dots	i	\dots	C
$\hat{Y}_k^{(j)}$	o	\dots	l	\dots	o
$P\{Y=l\}$	o	\dots	l	\dots	o
$Q\{Y=k\}$	\hat{P}_i	\dots	\hat{P}_i	\dots	\hat{P}_o

1. 所以 backbone 模型不能再是单输出的线性回归模型,

输出应是向量

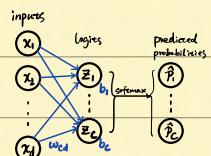
(for one sample point:)

$$\hat{y} = w_1x_1 + w_2x_2 + \dots + w_dx_d + b = [w_1, \dots, w_d] \begin{bmatrix} x_1 \\ | \\ | \\ x_d \end{bmatrix} + b = \vec{w}^T \cdot \vec{x} + b$$

$$\widehat{\vec{y}} = \left(\sum_{i=1}^d w_{1i} x_i + b_1, \dots, \sum_{i=1}^d w_{ci} x_i + b_c \right)^T = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1d} \\ w_{21} & w_{22} & \dots & w_{2d} \\ \vdots & & & \\ w_{c1} & w_{c2} & \dots & w_{cd} \end{bmatrix}_{c \times d} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}_{d \times 1} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix}_{c \times 1} = W \vec{x} + \vec{b}$$

(for a batch of samples)

$$Y = Wx + b$$



2. 与 Logistic Regression 中的动机完全一样, 为了让输出“是”概率(与前一致的讨论✓),

我们需要对输出 $\vec{z} = W\vec{x} + \vec{b}$ 做变换 ($R^c \rightarrow [0, 1]^c$)

Softmax Function (完全是 Logistic function 的推广)

现在我们可以完整得到预测分布 Q : $Q\{Y=k\} = \text{softmax}(Z_k)$

3. 最后，关于损失函数：

事实上，损失函数无需改变，我们的目标始终是优化两个概率分布间的距离 \Leftrightarrow 优化 $H(P, Q)$

只需将预测概率代入：

for one sample point (\vec{x}, \vec{y}) :

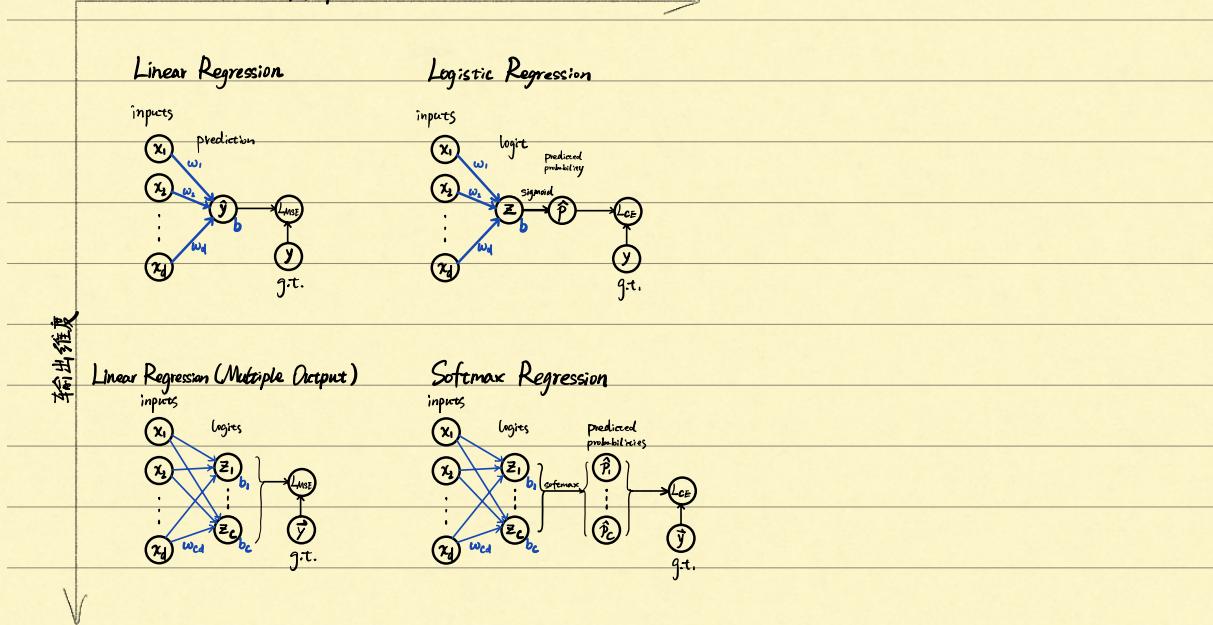
$$L_{CE} = H(P, Q) = - \sum_{k=1}^C P(Y=k) \log [Q(Y=k)] = - \sum_{k=1}^C y_k \log \left(\frac{e^{z_k}}{\sum_{j=1}^C e^{z_j}} \right)$$

for a batch of samples $\vec{x}^{(1)}, \dots, \vec{x}^{(m)}$:

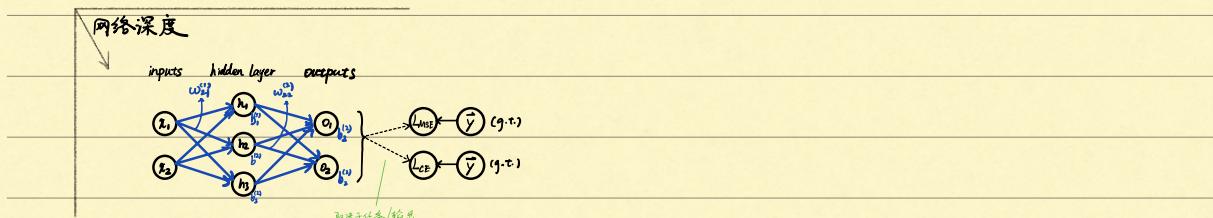
$$\begin{aligned} L_{CE} &= \frac{1}{m} \sum_{i=1}^m H(P^{(i)}, Q^{(i)}) = \frac{1}{m} \sum_{i=1}^m \left[- \sum_{k=1}^C P^{(i)}(Y=k) \log [Q^{(i)}(Y=k)] \right] \\ &= \frac{1}{m} \sum_{i=1}^m \left[- \sum_{k=1}^C y_k^{(i)} \log [Q^{(i)}(Y=k)] \right] \end{aligned}$$

A review and beyond.....

任务 / 输出



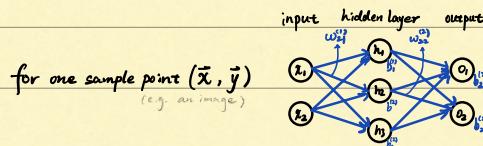
如果从“推广”/“拓展”的角度来看，上述四种模型还有一个“维度”没有拓展——网络层数



从这个角度：Linear Regression \Leftrightarrow MLP (退化) with MSE Loss

Softmax Regression \Leftrightarrow MLP (退化) with CE Loss

On Multi-Layer Perceptron (MLP)



\rightarrow

$$h_1 = g_1(\omega_{11}^{(1)}x_1 + \omega_{12}^{(1)}x_2 + b_1^{(1)})$$

$$h_2 = g_1(\omega_{21}^{(1)}x_1 + \omega_{22}^{(1)}x_2 + b_2^{(1)}) \quad \text{or} \quad \vec{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = g_1 \left(\begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} \\ \omega_{31}^{(1)} & \omega_{32}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix} \right) = g_1(W^{(1)}\vec{x} + \vec{b}^{(1)})$$

$$h_3 = g_1(\omega_{31}^{(1)}x_1 + \omega_{32}^{(1)}x_2 + b_3^{(1)})$$

element-wise

Similarly:

$$o_1 = g_2(h_1 + \omega_{11}^{(2)}h_2 + \omega_{12}^{(2)}h_3 + b_1^{(2)})$$

$$o_2 = g_2(h_2 + \omega_{21}^{(2)}h_1 + \omega_{22}^{(2)}h_3 + b_2^{(2)}) \quad \text{or} \quad \vec{o} = \begin{bmatrix} o_1 \\ o_2 \end{bmatrix} = g_2 \left(\begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} & \omega_{13}^{(2)} \\ \omega_{21}^{(2)} & \omega_{22}^{(2)} & \omega_{23}^{(2)} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} \right) = g_2(W^{(2)}\vec{h} + \vec{b}^{(2)})$$

* together: $\vec{o} = g_2 \left(W^{(2)} \cdot g_1(W^{(1)}\vec{x} + \vec{b}^{(1)}) + \vec{b}^{(2)} \right)$

* we can easily extend \vec{x} (one sample point) to X (a (mini) batch of samples e.g. m images)

$$\vec{o} = g_2 \left(W^{(2)} \cdot g_1(W^{(1)}X + \vec{b}^{(1)}) + \vec{b}^{(2)} \right), \text{ where } X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ x_d^1 & x_d^2 & \dots & x_d^m \end{bmatrix}, \vec{o} = \begin{bmatrix} o_1^1 & o_1^2 & \dots & o_1^m \\ o_2^1 & o_2^2 & \dots & o_2^m \\ \vdots & \vdots & \ddots & \vdots \\ o_d^1 & o_d^2 & \dots & o_d^m \end{bmatrix} = [\vec{o}^1 \ \vec{o}^2 \ \dots \ \vec{o}^m]$$

this matrix-base format is great for code implementation ✓ (mini batch SGD)
outputs = model(inputs)

→ with O , we can compute loss L e.g. 1: MSE loss: (in this case, $Y = [\vec{y}^1, \dots, \vec{y}^m]$, \vec{y}^i has same dim with \vec{o}^i)

$$\text{loss} = \text{criterion}(outputs, y) \quad L_{\text{MSE}}(O) = \frac{1}{m} \sum_{k=1}^m \left[\frac{1}{d} \sum_{j=1}^d (O_j^k - y_j^k)^2 \right] = \frac{1}{m} \sum_{k=1}^m \frac{1}{d} \| \vec{o}^k - \vec{y}^k \|_2^2$$

MSE loss for one sample point
MSE loss average over a batch of samples

e.g. 2: CE loss: (in this case, $Y = [\vec{y}^1, \dots, \vec{y}^m]$, \vec{y}^i is a one-hot vector represents class

$$L_{\text{CE}}(O) = \frac{1}{m} \sum_{k=1}^m \left[\frac{1}{c} \sum_{j=1}^c y_j^k \log(O_j^k) \right]$$

optimizer.zero_grad()

loss.backward()

→ use backpropagation, we can obtain gradient of loss w.r.t. each parameter e.g. $\frac{\partial L}{\partial w_i} \cdot \frac{\partial L}{\partial b_j}$

optimizer.step()

→ then we can update parameters use gradient descent

Remark: "total number of parameters" (模型参数量) is simply the total number of parameters in the model (trainable)

in above case (a FNN/MLP): it's just number of weights (w_i) and biases (b_j) e.g. $12 + 5 = 17 \checkmark$
 $6+6$ $3+2$