深度学习技术与应用（3）

Deep Learning: Techniques and Applications (3)

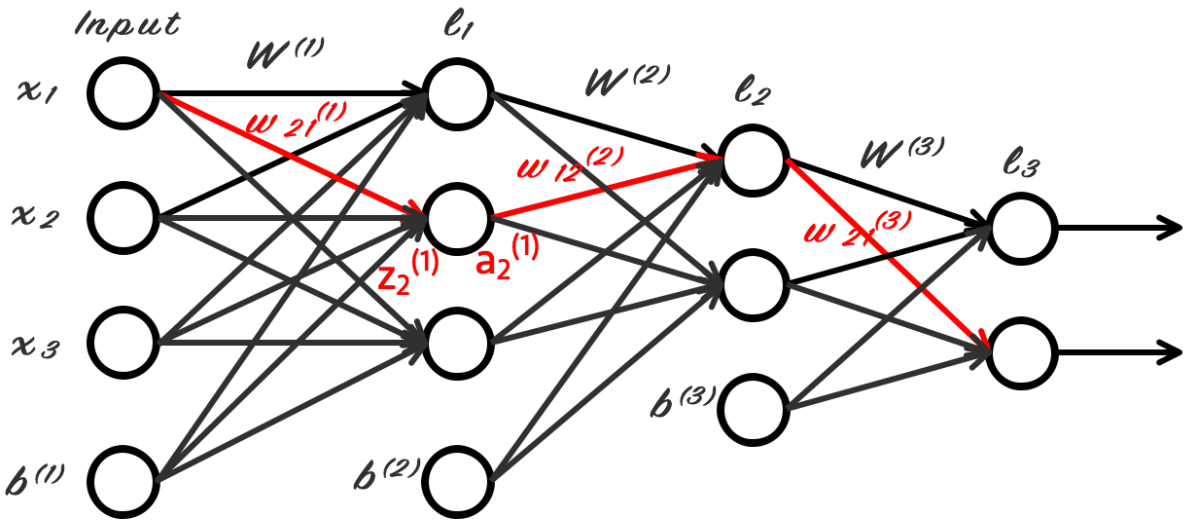
Ge Li

Peking University

Table of contents

关于训练方法

前向传播计算



$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{21}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{3} w_{1i}^{(1)} x_i + b_1^{(1)} \\ \sum_{i=1}^{3} w_{2i}^{(1)} x_i + b_2^{(1)} \\ \sum_{i=1}^{3} w_{3i}^{(1)} x_i + b_3^{(1)} \end{bmatrix}$$

批量前向计算

$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{21}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix} + \begin{bmatrix} b_1^{(1)} & b_1^{(1)} \\ b_2^{(1)} & b_2^{(1)} \\ b_3^{(1)} & b_3^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{3} w_{1i}^{(1)} x_{1i} + b_1^{(1)} & \sum_{i=1}^{3} w_{1i}^{(1)} x_{2i} + b_1^{(1)} \\ \sum_{i=1}^{3} w_{2i}^{(1)} x_{1i} + b_2^{(1)} & \sum_{i=1}^{3} w_{2i}^{(1)} x_{2i} + b_2^{(1)} \\ \sum_{i=1}^{3} w_{3i}^{(1)} x_{1i} + b_3^{(1)} & \sum_{i=1}^{3} w_{3i}^{(1)} x_{2i} + b_3^{(1)} \end{bmatrix}$$

用 A^{l+1} 表示与一个 Batch 对应的 $l+1$ 层所有神经元的输出值（其他变量含义相应可知），则：

$$Z^l = W^{(l)} A^{(l-1)} + B^{(l)} \qquad A^{(l)} = f(Z^{(l)})$$

批量权重更新

设一个 Batch 所对应的输入为：$X =$

$$\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{m1} \\ x_{12} & x_{22} & \cdots & x_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1n} & x_{2n} & \cdots & x_{mn} \end{bmatrix}$$

1. 设 $\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$ 分别为“当输入第 r 列数据时，第 l 层上的权重和偏置所对应的更新量”；则：
2. $\sum_{r=1}^{m} \Delta W_r^{(l)}$ 和 $\sum_{r=1}^{m} \Delta b_r^{(l)}$ 分别为“当输入完一个 Batch 的数据（X）时，第 l 层上的权重和偏置所对应的更新量的和”；
3. 这时，我们取上述更新量和的平均 $\frac{1}{m}\sum_{r=1}^{m} \Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m} \Delta W_r^{(l)}$，分别作为对第 l 层上的权重和偏置所应进行的更新。

批量梯度下降训练算法

设 X 为 M 列输入向量（一个 Batch），设 $\Delta W^{(l)}$ 和 $\Delta b^{(l)} = 0$ 分别为输入一个 Batch 后，第 l 层进行调整的权重和偏置的更新量；

1. 初始化：$\Delta W^{(l)} = 0$，$\Delta b^{(l)} = 0$
2. 计算权重和偏置的更新量矩阵（共 M 列），其中第 r 列分别为：$\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$；
3. 对上述两个矩阵，分别计算：$\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$
4. 利用上述结果进行权重更新：

$$W^{(l)} = W^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)})$$
$$b^{(l)} = b^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta b_r^{(l)})$$

循环执行上述过程，直到 Loss Function 输出值达到要求。

批量梯度下降的训练流程

设 X 为 M 列输入向量（一个 Batch），设 $\Delta W^{(l)}$ 和 $\Delta b^{(l)} = 0$ 分别为输入一个 Batch 后，第 l 层进行调整的权重和偏置的更新量；

1. 初始化：$\Delta W^{(l)} = 0$，$\Delta b^{(l)} = 0$
2. 计算权重和偏置的更新量矩阵（共 M 列），其中第 r 列分别为：$\Delta W_r^{(l)}$ 和 $\Delta b_r^{(l)}$；
3. 对上述两个矩阵，分别计算：$\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$ 和 $\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)}$
4. 利用上述结果进行权重更新：

$$W^{(l)} = W^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta W_r^{(l)})$$
$$b^{(l)} = b^{(l)} - \alpha(\frac{1}{m}\sum_{r=1}^{m}\Delta b_r^{(l)})$$

循环执行上述过程，直到达到收敛条件。

各种训练方法

1. 批量梯度下降（Batch GD）
   1. 每轮权重更新所有样本都参与训练；
   2. 迭代多轮，直到达到收敛条件；
2. 随机梯度下降（SGD）
   1. 每轮权重更新只随机选取一个样本参与训练；
   2. 迭代多轮，达到收敛条件便可终止；
3. 小批量梯度下降（Mini-Batch SGD）
   1. 每轮权重更新（随机）取一部分样本参与训练；
   2. 迭代多轮，直到满足收敛条件；

关于 *Loss Function*

代价函数设计的基本方法

“Rather than guessing that some function might make a good estimator and then analyzing its bias and variance, we would like to have some principle from which we can derive specific functions that are good estimators for different models.”
“The most common such principle is the maximum likelihood principle.”

最大似然估计

- Consider: a set of m examples $\mathbb{X} = x^{(1)}, ..., x^{(m)}$, drawn independently from the true but unknown data generating distribution $p_{data}(x)$.
- Let $p_{model}(x, \theta)$ be a parametric family of probability distributions over the same space indexed by θ.
- Then, the maximum likelihood estimator for θ is defined as:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, p_{model}(\mathbb{X}; \theta) \tag{1}$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{m} p_{model}(x^{(i)}; \theta) \tag{2}$$

最大似然估计

To obtain a more convenient but equivalent optimization problem, we transform a product into a sum:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} \log p_{model}(x^{(i)}; \theta) \tag{3}$$

Because the arg max does not change when we rescale the cost function, we can divide by m to obtain a version of the criterion that is expressed as an expectation with respect to the empirical distribution $p_{\hat{data}}$ defined by the training data:

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \mathbb{E}_{x \sim p_{\hat{data}}} \log p_{model}(x; \boldsymbol{\theta}) \tag{4}$$

最大似然估计的合理性

- One way to interpret maximum likelihood estimation is to view it as minimizing the ***dissimilarity*** between the empirical distribution $p_{\hat{data}}$ defined by the training set and the model distribution.
- The degree of dissimilarity between the two measured by the KL divergence:

$$D_{ML}(p_{\hat{data}}||p_{model}) = \mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{\hat{data}}(x) - \log p_{model}(x)\right] \quad (5)$$

- The term on the left is a function only of the training data, not the model.(not include θ) This means when we try to minimize the KL divergence, we need only minimize:

$$-\mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{model}(x)\right] \quad (6)$$

This is the same as the maximization in equation (4).

信息熵、交叉熵

- 信息熵，表示一个信源发出的信号的不确定程度。在信源发出的信号中，某信号出现的概率越大，熵越小；反之越大。
- 因此，信息熵的估算需要满足两个条件：
  1. 单调递减性，信息熵的值是信号 i 出现概率 p_i 的单调递减函数
  2. 可加性，两个独立符号所对应的不确定程度应等于各自不确定程度之和
- Shannon 用 log 函数定义信号 i（样本 i）的信息熵：

$$f(p(i)) = \log \frac{1}{p(i)} = -\log p(i)$$

- 则，包含 n 个样本的样本集合的信息熵定义为：

$$E(P) = \sum_{i}^{n} p(i) \log \frac{1}{p(i)}$$

注意：这里的 $p(i)$ 表示样本的真实分布；

信息熵、交叉熵

- 然而，在机器学习中，我们通常使用模型分布 $q(i)$ 来逼近真实分布，这时，样本集合的信息熵为：

$$E(P,Q) = \sum_{i}^{n} p(i) \log \frac{1}{q(i)} = -\sum_{i}^{n} p(i) \log q(i)$$

- 根据 Gibbs 不等式，有：$E(P,Q) \geq E(P)$
- 我们将 $E(P,Q)$ 与 $E(P)$ 的差，定义为“使用模型分布 Q 来逼近真实分布 P 时的 相对熵”（又称 KL 散度）：

$$D(P \parallel Q) = E(P,Q) - E(P) = \sum P(i) \log \frac{P(i)}{Q(i)}$$

- KL 散度，表示 2 个概率分布的差异程度；

最大似然估计的合理性

- One way to interpret maximum likelihood estimation is to view it as minimizing the ***dissimilarity*** between the empirical distribution $p_{\hat{data}}$ defined by the training set and the model distribution.
- The degree of dissimilarity between the two measured by the KL divergence:

$$D_{ML}(p_{\hat{data}}||p_{model}) = \mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{\hat{data}}(x) - \log p_{model}(x)\right] \quad (7)$$

- The term on the left is a function only of the training data, not the model.(not include θ) This means when we try to minimize the KL divergence, we need only minimize:

$$-\mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{model}(x)\right] \quad (8)$$

This is the same as the maximization in equation (4).

关于交叉熵与 KL 散度

Minimizing this KL divergence corresponds exactly to minimizing the crossentropy between the distributions.

- Many authors use the term "cross-entropy" to identify specifically the negative log-likelihood of a Bernoulli or softmax distribution, but that is a misnomer.

Any loss consisting of a negative log-likelihood is a cross entropy between the empirical distribution defined by the training set and the probability distribution defined by model.

- For example, mean squared error is the cross-entropy between the empirical distribution and a Gaussian model.

关于高斯分布

- Many distributions we wish to model are truly close to being normal distributions.
- Out of all possible probability distributions with the same variance, the normal distribution encodes the maximum amount of uncertainty over the real numbers.
- The central limit theorem shows that the sum of many independent random variables is approximately normally distributed.
- This means that in practice, many complicated systems can be modeled successfully as normally distributed noise, even if the system can be decomposed into parts with more structured behavior.
- We can think of the normal distribution as being the one that inserts the least amount of prior knowledge into a model.

关于高斯分布

- 若随机变量 X 服从一个位置参数为 μ、尺度参数为 σ 的概率分布，且其概率密度函数为:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- 其中 μ 为随机变量 X 的数学期望，决定了正态分布的位置，σ^2 为随机变量 X 的方差，σ 为随机变量 X 的标准差，决定了正态分布的幅度。且正态分布函数在 $\mu \pm \sigma$ 处，有拐点。
- 记为：$N \sim (\mu, \sigma^2)$

Multivariate Gaussian Distribution

若 p 维随机向量 $X = (X_1, X_2, ...X_p)'$ 的概率密度函数为：

$$f(x_1, x_2, ...x_p) = \frac{1}{(\sqrt{2\pi})^p |\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(X-\mu)'\Sigma^{-1}(X-\mu)\} \quad (9)$$

其中：μ 为 p 维向量，是 X 的均值向量，Σ 是 $p \times p$ 维协方差矩阵，即 p 阶正定矩阵，Σ^{-1} 是 Σ 的逆矩阵，$|\Sigma|$ 是 Σ 的行列式；

则称：$X = (X_1, X_2, ...X_p)'$ 服从 p 维正态分布，记为 $X \sim N_p(\mu, \Sigma)$

特别的：当 $p = 1$ 时，Σ 成为一个 1×1 的矩阵，$|\Sigma|^{\frac{1}{2}}$ 也就是标准差 σ，Σ^{-1} 也就是 σ^{-2}，而 $(X-\mu)'(X-\mu)$ 也变成 $(X-\mu)^2$，因此，多元正态分布就变成了一元正态分布。

同分布中心极限定理

设 $X_1, X_2, ..., X_n, ...$ 是独立同分布的随机变量序列，
$EX_i = \mu, DX_i = \sigma^2 \neq 0 (i = 1, 2, ...)$，则对于任意的实数 x，有：

$$\lim_{n\to\infty} P\left\{\frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma} < x\right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t \tag{10}$$

定理说明：

- 对独立同分布的随机变量序列 $X_1, X_2, ..., X_n, ...$，当 n 无限增大时，$Y_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$ 服从标准正态分布 $N(0, 1)$，而 $\sum_{i=1}^{n} X_i = \sqrt{n}\sigma Y_n + n\mu$ 则服从正态分布 $N(n\mu, n\sigma^2)$.
- 可见：当 n 足够大时，n 个独立同分布的随机变量之和，服从正态分布。

非同分布的李雅普诺夫定理

设 $X_1, X_2, ..., X_n, ...$ 是互相独立的随机变量，且 $EX_i = \mu, DX_i = \sigma^2 \neq 0 (i = 1, 2, ...)$，若存在 $\delta > 0$，使得：

$$\lim_{n\to\infty} \frac{1}{B_n^{2+\delta}} \sum_{i=1}^{n} E\left|X_i - \mu_i\right|^{2+\delta} = 0 \text{ 其中：} B_n^2 = \sum_{i=1}^{n} \sigma_i^2 \tag{11}$$

则对于任意实数 x，有：

$$\lim_{n\to\infty} P\left\{\frac{1}{B_n}\sum_{i=1}^{n}(X_i - \mu_i) < x\right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t \tag{12}$$

非同分布的李雅普诺夫定理

李雅普诺夫定理表明：当 n 很大时：

$$Z_n = \frac{1}{B_n}\sum_{i=1}^{n}(X_i - \mu_i) = \frac{1}{B_n}(\sum_{i=1}^{n} X_i - \sum_{i=1}^{n}\mu_i) \tag{13}$$

近似服从正态分布 $N(0,1)$，即 $\sum_{i=1}^{n} X_i = B_n Z_n + \sum_{i=1}^{n}\mu_i$ 近似服从正态分布 $N(\sum_{i=1}^{n}\mu_i, B_n^2)$

- 不管随机变量 $X_1, X_2, ..., X_n, ...$ 各自具有怎样的分布，当 n 很大时，它们的和近似服从正态分布。
- 从理论上再次肯定了：大量随机因素叠加的结果，近似服从正态分布。

最大似然估计的合理性

- One way to interpret maximum likelihood estimation is to view it as minimizing the ***dissimilarity*** between the empirical distribution $p_{\hat{data}}$ defined by the training set and the model distribution.
- The degree of dissimilarity between the two measured by the KL divergence:

$$D_{ML}(p_{\hat{data}}||p_{model}) = \mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{\hat{data}}(x) - \log p_{model}(x)\right] \quad (14)$$

- The term on the left is a function only of the training data, not the model.(not include θ) This means when we try to minimize the KL divergence, we need only minimize:

$$-\mathbb{E}_{x\sim p_{\hat{data}}}\left[\log p_{model}(x)\right] \quad (15)$$

This is the same as the maximization in equation (4).

条件概率的最大似然估计

The maximum likelihood estimator can readily be generalized to the case where our goal is to estimate a conditional probability $P(y|x;\theta)$.
If X represents all our inputs and Y all our observed targets, then the conditional maximum likelihood estimator is:

$$\theta_{ML} = \operatorname*{argmax}_{\theta} P(Y|X;\boldsymbol{\theta}). \tag{16}$$

If the examples are assumed to be i.i.d. (independent identically distributed), then this can be decomposed into:

$$\theta_{ML} = \operatorname*{argmax}_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\boldsymbol{\theta}). \tag{17}$$

基于上述原则推导 MSE 的合理性

- 最小均方误差函数 (Mean Squared Error, MSE)：对于一组有 m 个样本的训练集，代价函数 MSE 定义如下:

$$J(\theta) = \frac{1}{2}\Sigma_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

- 由前文推导，可合理假设 $p(y|x) \sim \mathcal{N}(y; \hat{y}(x; w), \sigma^2)$. 其中，函数 $\hat{y}(x; w)$ 表示对高斯平均值的预测；

基于上述原则推导 MSE 的合理性

此时，条件概率的最大似然估计为:

$$\sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\boldsymbol{\theta}) \tag{18}$$

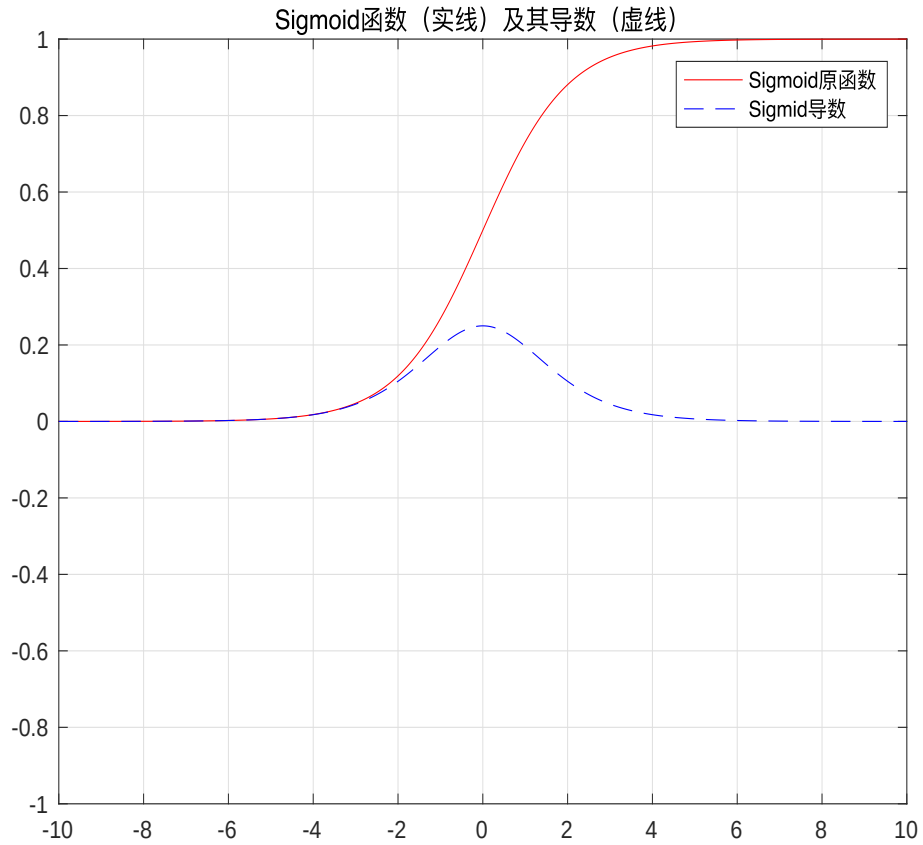
$$= -m\log\sigma - \frac{m}{2}\log(2\pi) - \sum_{i=1}^{m}\frac{\|\hat{y}^{(i)} - y^{(i)}\|^2}{2\sigma^2} \tag{19}$$

$\hat{y^{(i)}}$ 为输入 $x^{(i)}$ 时模型的输出，m 是训练数据集的样本数.
可知，对上式的最大化等同于对如下等式的最小化:

$$MSE_{train} = \frac{1}{m}\sum_{i=1}^{m}\|\hat{y}^{(i)} - y^{(i)}\|^2 \tag{20}$$

关于激活函数

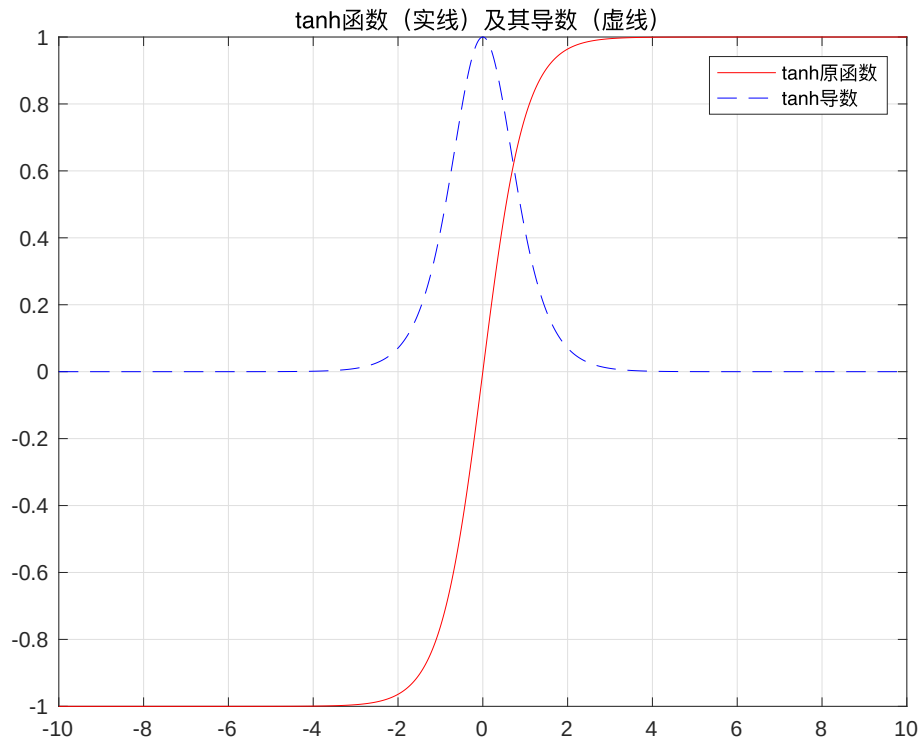
Sigmoid



$$\delta(z) = \frac{1}{1 + exp(-z)} \in (0, 1)$$

$$\begin{aligned}\delta'(z) &= \frac{-exp(-z)}{(1 + exp(-z))^2} \\ &= \delta(z)(1 - \delta(z))\end{aligned}$$

Tanh

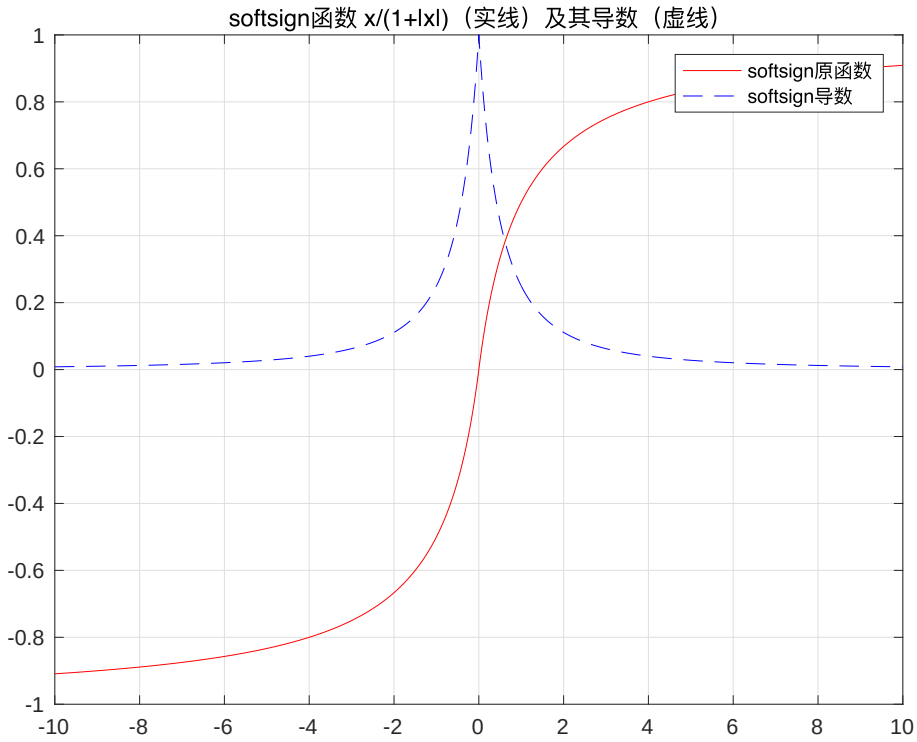


$$tanh(z) = \frac{exp(z) - exp(-z)}{exp(z) + exp(-x)}$$
$$= 2\delta(2z) - 1$$

$$tanh(z) \in (-1, 1)$$

$$tanh'(z) = 1 - (\frac{exp(z) - exp(-z)}{exp(z) + exp(-z)})^2$$
$$= 1 - tanh^2(z)$$

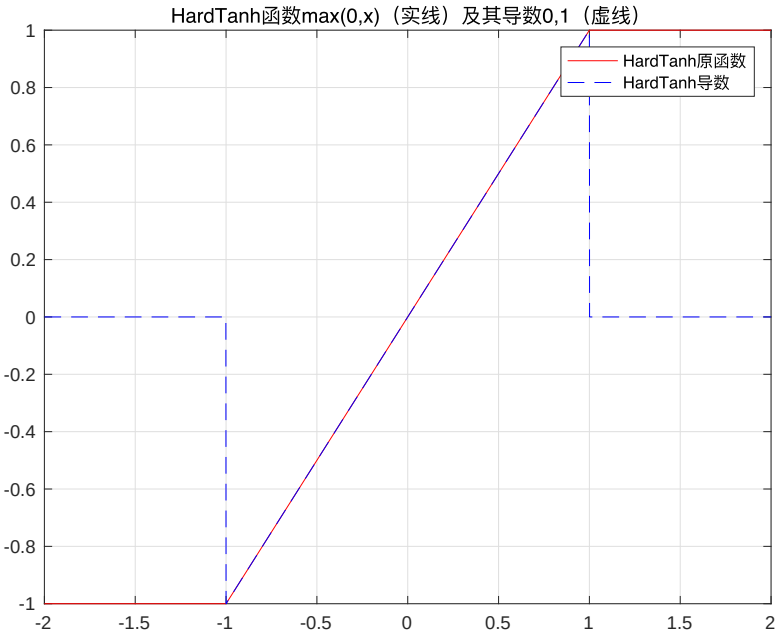
Softsign



$$softsign(z) = \frac{z}{1+|z|}$$

$$softsign'(z) = \frac{sgn(z)}{(1+z)^2}$$

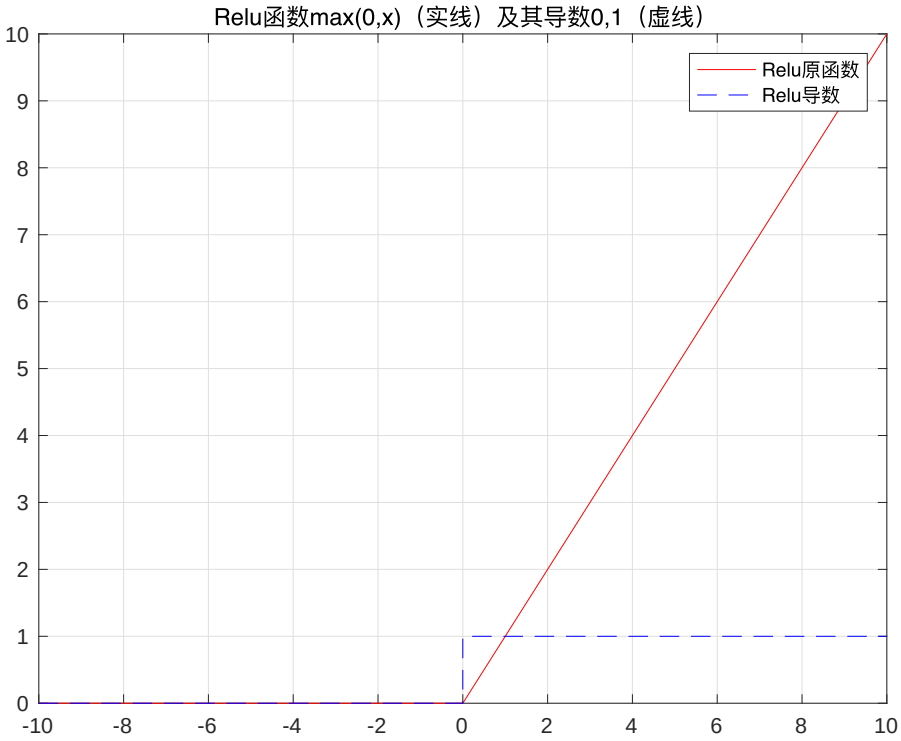
Hard Tanh



$$hardtanh(z) = \begin{cases} -1 & : z < -1 \\ z & : otherwise \\ 1 & : z > 1 \end{cases}$$

$$hardtanh'(z) = \begin{cases} 1 & : -1 \geq z \leq 1 \\ 0 & : otherwise \end{cases}$$

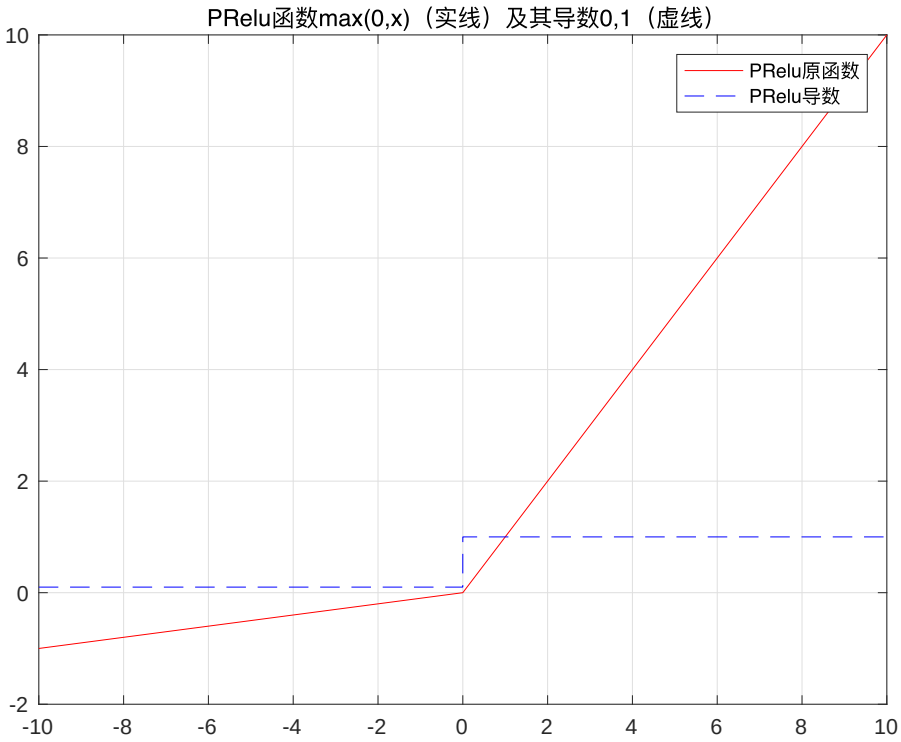
ReLU - Rectified Linear Unit



$$relu(z) = \begin{cases} 0 & : z < 0 \\ z & : z \geq 0 \end{cases}$$

$$relu'(z) = \begin{cases} 1 & : z > 0 \\ 0 & : otherwise \end{cases}$$

PReLU - Parametric ReLU

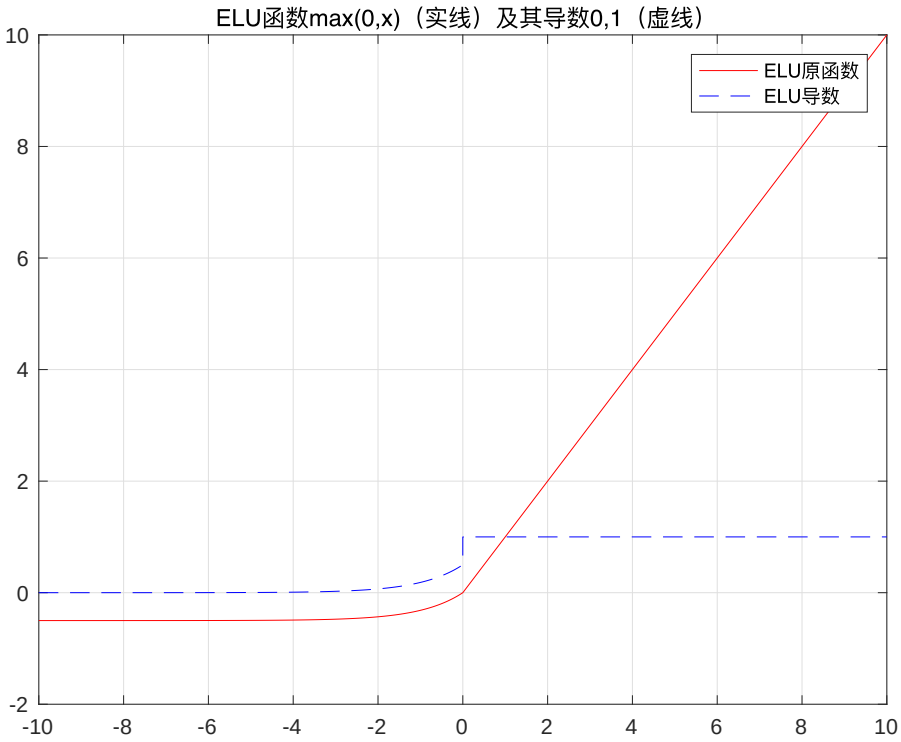


$$prelu(z) = \begin{cases} \alpha z & : z < 0 \\ z & : z \geq 0 \end{cases}$$

$$where\ 0 < \alpha < 1$$

$$prelu'(z) = \begin{cases} 1 & : z > 0 \\ \alpha & : otherwise \end{cases}$$

ELU - Exponential Linear Unit

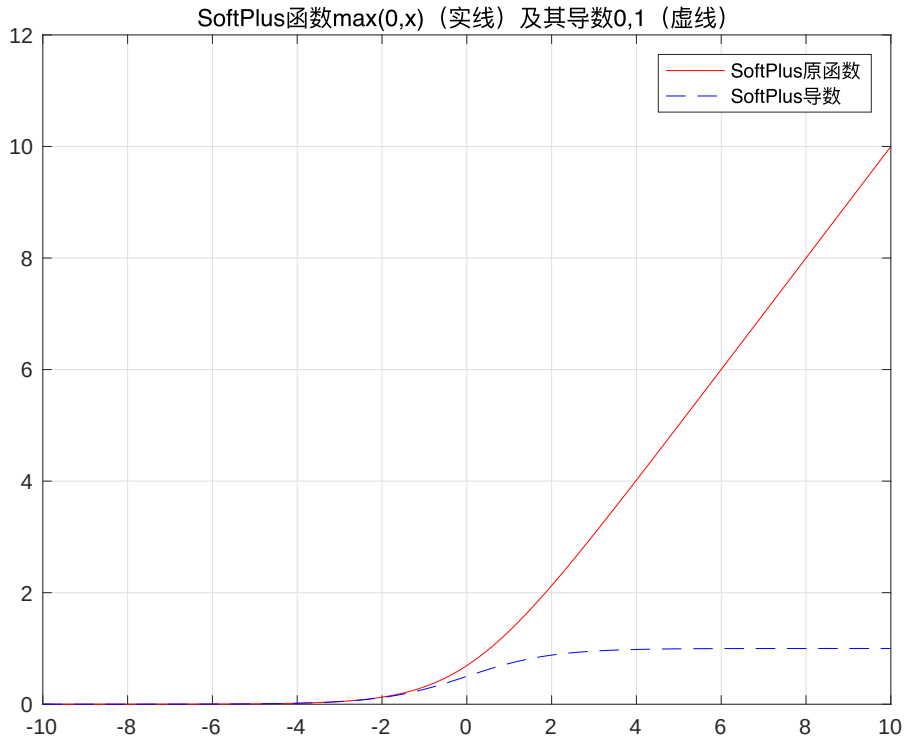


$$elu(z) = \begin{cases} \alpha(e^z - 1) & : z < 0 \\ z & : z \geq 0 \end{cases}$$

$$where\ 0 < \alpha < 1$$

$$elu'(z) = \begin{cases} elu(z) + \alpha & : z < 0 \\ 1 & : z >= 0 \end{cases}$$

SoftPlus

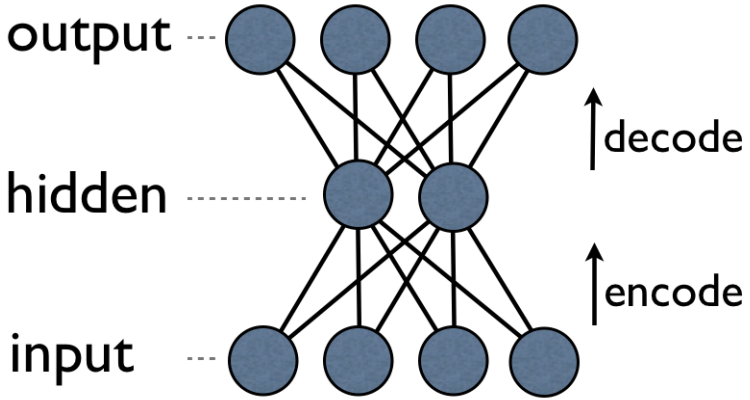


$$softplus(z) = \log_e(1 + e^z))$$

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

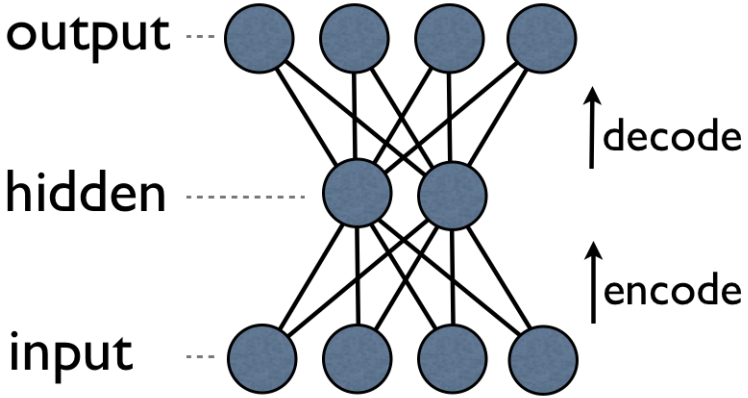
AutoEncoder

- Auto-Encoder（AE）
  - 自编码器，80年代晚期出现
  - 主要用于降维，后用于主成分分析



n 输入输出层神经元个数
m 隐藏层神经元个数
x, y, h 各层神经元上的向量
p，q 各层神经元的偏置
w 输入层与隐藏层之间的权值
$\tilde{w}$ 隐藏层与输出层之间的权值

Basic AutoEncoder

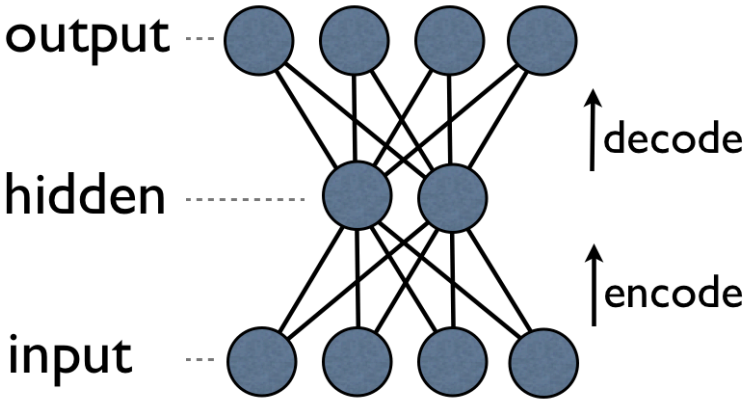


n 输入输出层神经元个数
m 隐藏层神经元个数
x, y, h 各层神经元上的向量
p，q 各层神经元的偏置
w 输入层与隐藏层之间的权值
$\tilde{w}$ 隐藏层与输出层之间的权值

$$\mathbf{h} = f(\mathbf{x}) := s_f(W\mathbf{x} + \mathbf{p});$$
$$\mathbf{y} = g(\mathbf{h}) := s_g(\widetilde{W}\mathbf{h} + \mathbf{q}),$$

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

$$s_g(z) = \frac{1}{1+e^{-z}} \text{ 或 } s_g(z) = z.$$



Basic AutoEncoder

$$\mathbf{h} = f(\mathbf{x}) := s_f(W\mathbf{x} + \mathbf{p});$$
$$\mathbf{y} = g(\mathbf{h}) := s_g(\widetilde{W}\mathbf{h} + \mathbf{q}),$$

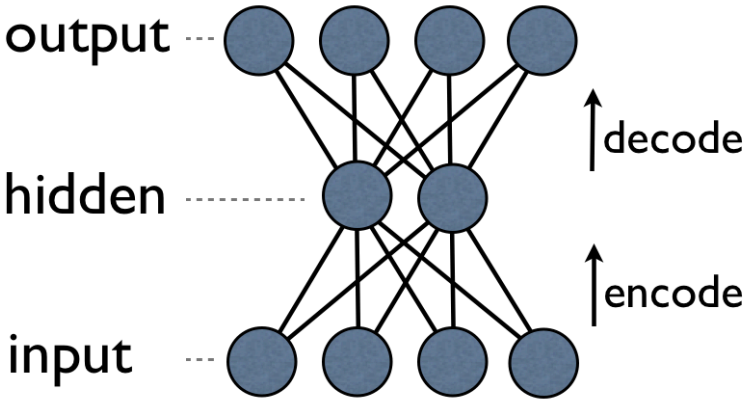
- 若 $s_g(z) = z$. 通常取 重构误差 函数为 平方误差：

$$L(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2.$$

- 若 $s_g(z) = \frac{1}{1+e^{-z}}$ 通常取 重构误差 函数为 交叉熵：

$$L(\mathbf{x}, \mathbf{y}) = -\sum_{i=1}^{n} \left[x_i \log(y_i) + (1 - x_i) \log(1 - y_i) \right].$$

Basic AutoEncoder



$$L(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2.$$

$$L(\mathbf{x}, \mathbf{y}) = -\sum_{i=1}^{n} \left[x_i \log(y_i) + (1 - x_i) \log(1 - y_i) \right].$$

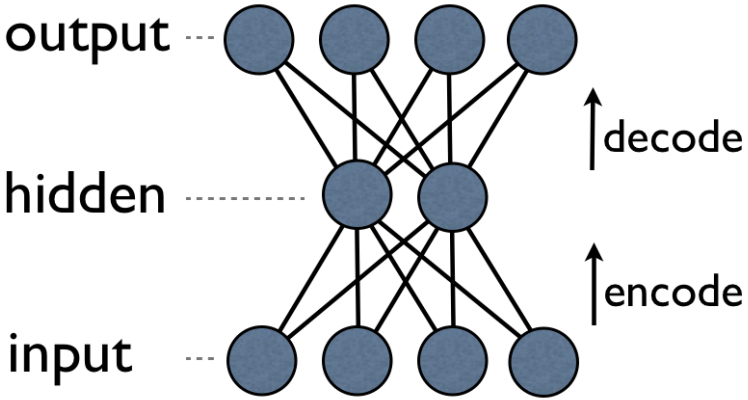
- 整体 损失函数 为：

$$\mathcal{J}_{AE}(\theta) = \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

- 或：

$$\mathcal{J}_{AE}(\theta) = \frac{1}{N} \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

Basic AutoEncoder

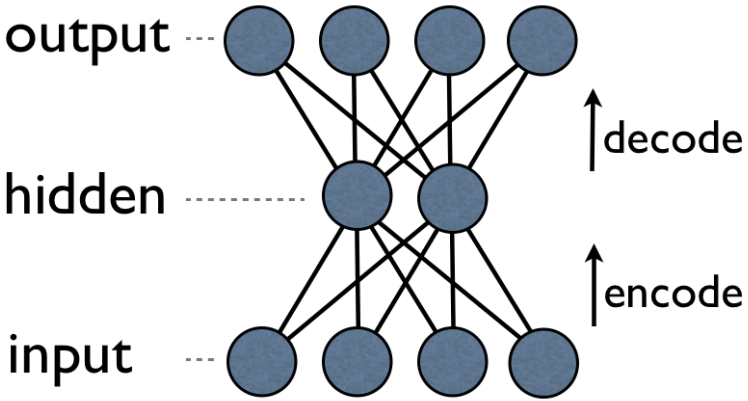


$$\mathcal{J}_{AE}(\theta) = \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

$$\mathcal{J}_{AE}(\theta) = \frac{1}{N} \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

- 问题：
  - 训练过程中，如果不对 重构函数 进行 限制，而直接进行最小化求解
  - 则，训练结果 极有可能 成为一个恒等函数
  - 因此，必须对 损失函数 进行正则化处理，以限制其表现

Regularized AutoEncoder



$$\mathcal{J}_{AE}(\theta) = \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

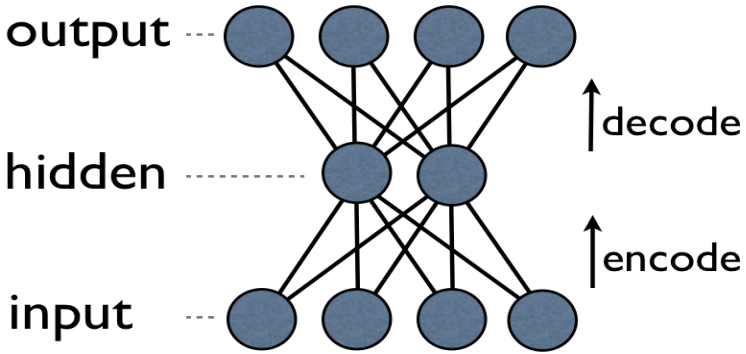
$$\mathcal{J}_{AE}(\theta) = \frac{1}{N} \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))),$$

- 增加权重衰减项的 损失函数：

$$\mathcal{J}_{AE+wd}(\theta) = \sum_{\mathbf{x} \in S} L(\mathbf{x}, g(f(\mathbf{x}))) + \lambda \sum_{i,j} W_{i,j}^2,$$

权重衰减项

Sparse AutoEncoder



n 输入输出层神经元个数
m 隐藏层神经元个数
x, y, h 各层神经元上的向量
p，q 各层神经元的偏置
w 输入层与隐藏层之间的权值
$\tilde{w}$ 隐藏层与输出层之间的权值

- 隐藏层上第 j 号神经元在训练集 $S = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ 上的平均激活度.

$$\widehat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} h_j(\mathbf{x}^{(i)}), \quad \text{令} \quad \widehat{\rho}_j = \rho,\ j = 1, 2, \cdots, m,$$

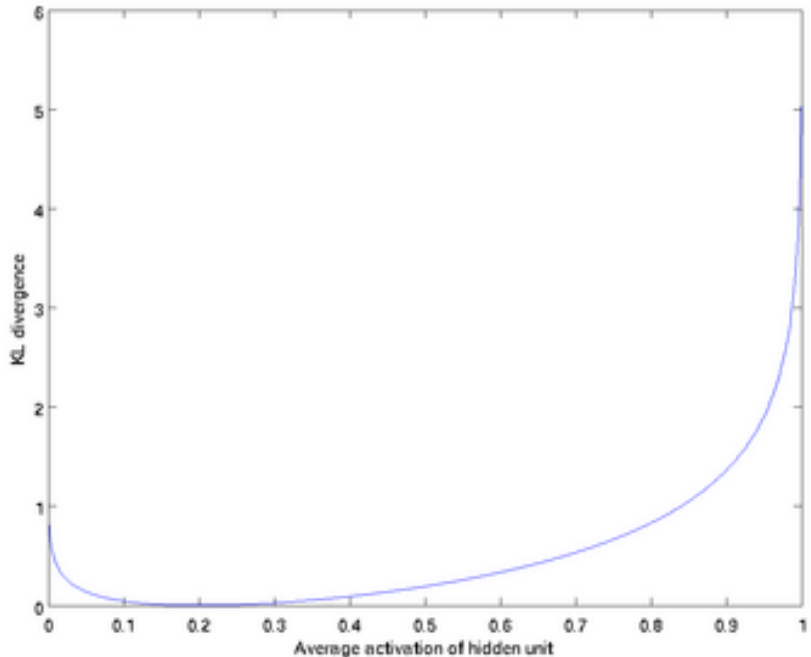
其中ρ为一个很小的数，例如$\rho < 0.05$

目的：当某隐藏层神经元的平均激活度超过ρ时，对其进行惩罚处理

如何进行惩罚处理？

- 引入 相对熵值 函数

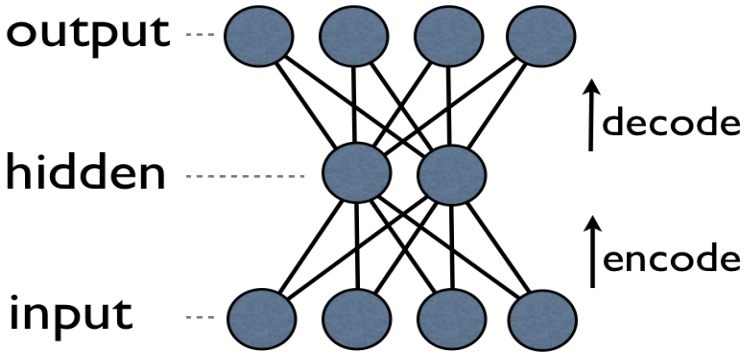
$$KL(\rho||\hat{\rho}_j) = \rho * \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) * \log \frac{1-\rho}{1-\hat{\rho}_j}.$$



特点：

- 在 $\hat{\rho}_j = \rho$ 时达到最小值0
- 当 $\hat{\rho}_j$ 靠近0或者1的时候，相对熵则变得非常大；

Sparse AutoEncoder



n 输入输出层神经元个数
m 隐藏层神经元个数
x, y, h 各层神经元上的向量
p，q 各层神经元的偏置
w 输入层与隐藏层之间的权值
$\tilde{w}$ 隐藏层与输出层之间的权值

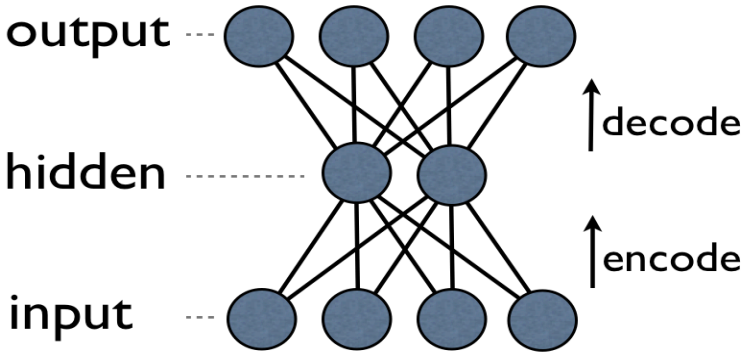
- 整体 损失函数 为：

$$\mathcal{J}_{AE+sp}(\theta) = \sum_{x \in S} L(x, g(f(x))) + \beta \sum_{j=1}^{m} KL(\rho || \hat{\rho}_j),$$

- 结合正则化的 损失函数 为：

$$\mathcal{J}_{AE+wd+sp}(\theta) = \sum_{x \in S} L(x, g(f(x))) + \lambda \sum_{i,j} W_{i,j}^2 + \beta \sum_{j=1}^{m} KL(\rho || \hat{\rho}_j).$$

Sparse AutoEncoder



$$\mathcal{J}_{AE+sp}(\theta) = \sum_{\mathrm{x}\in S} L(\mathrm{x}, g(f(\mathrm{x}))) + \beta \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j),$$

- 反向传播的 残差 计算：

$$\delta_i^{(2)} = \left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} \right) f'(z_i^{(2)}),$$

现在我们将其换成

$$\delta_i^{(2)} = \left(\left(\sum_{j=1}^{s_2} W_{ji}^{(2)} \delta_j^{(3)} \right) + \beta \left(-\frac{\rho}{\hat{\rho}_i} + \frac{1-\rho}{1-\hat{\rho}_i} \right) \right) f'(z_i^{(2)}).$$

Thanks.