



# Machine learning application



## Universal approximation theorem



### 什么是万能近似定理？

万能近似定理的核心思想是：只要有足够多的神经元和适当的激活函数，一个具有单隐藏层的前馈神经网络就可以以任意精度逼近任何连续函数。

换句话说：神经网络的结构非常强大，可以用来模拟从输入到输出的各种复杂关系。

---



### 公式

#### 1. 输入和输出空间：

○ 输入空间  $\mathcal{X} \subseteq \mathbb{R}^M$ : 比如一个学生的学习成绩包含  $M$  门课程的分数。

○ 输出空间  $\mathcal{Y} \subseteq \mathbb{R}$ : 比如预测这个学生未来考试的总分。

#### 2. 神经网络的假设空间：

$$\mathcal{M}_\varphi = \left\{ h : h(x) = \langle w^{(2)}, \varphi(W^{(1)}x + b^{(1)}) \rangle \mid p \in \mathbb{N}, W^{(1)} \in \mathbb{R}^{p \times M}, b^{(1)} \in \mathbb{R}^p, w^{(2)} \in \mathbb{R}^p \right\}$$

○ 这表示神经网络通过一系列权重和偏置映射输入到输出。

---



### 神经网络的步骤

#### 1. 线性变换：

$$W^{(1)} \cdot x + b^{(1)}$$

○ 这是输入数据的线性组合。比如， $x$  是学生的各科成绩， $W^{(1)}$  是每科成绩的重要性权重， $b^{(1)}$  是偏移量。

#### 2. 非线性激活：

$$\varphi(W^{(1)}x + b^{(1)})$$

○ 激活函数  $\varphi$  用于引入非线性，使神经网络可以拟合复杂关系。常见的激活函数有 ReLU 和 Sigmoid。

○ 实例： $\text{ReLU}(z) = \max(0, z)$ ，可以理解为负分数清零，只有正分数起作用。

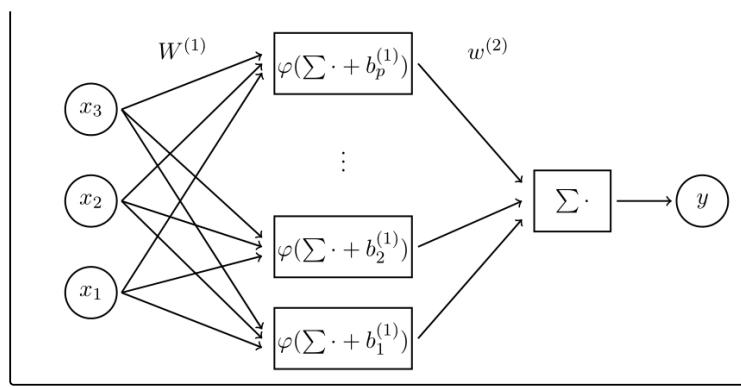
#### 3. 线性组合输出：

$$\langle w^{(2)}, \cdot \rangle$$

○ 输出层会将隐藏层的结果做线性组合，再得出最终结果（比如预测分数）。

---

下图展示了一个典型的 多层感知机 (MLP)，通过以下几步：



1. 输入层：输入变量  $x_1, x_2, x_3$  代表学生的三门成绩。

2. 隐藏层：

- 权重矩阵  $W^{(1)}$  和偏置  $b^{(1)}$  对输入进行线性变换。
- 激活函数  $\varphi$  对每个节点的输出进行非线性处理。

3. 输出层：

- 权重向量  $w^{(2)}$  对隐藏层的输出加权求和，得出预测结果  $y$ 。

### 💡 实际例子

用学生成绩预测总成绩

假设我们想根据三门课程的成绩（输入  $x_1, x_2, x_3$ ）预测学生的总成绩（输出  $y$ ）。神经网络的过程可以这样理解：

1. 输入层：输入  $x_1 = 80, x_2 = 75, x_3 = 85$ 。

2. 隐藏层计算：

- 首先计算线性变换  $W^{(1)} \cdot x + b^{(1)}$ 。
- 假设  $W^{(1)} = [0.5 \ 0.2 \ 0.3 \ 0.1 \ 0.7 \ 0.2]$  和  $b^{(1)} = [0.1 \ 0.2]$ 。
- 激活函数 ReLU 将负值清零。

3. 输出层计算：

- 结果经过  $w^{(2)}$  的线性组合得到预测的  $y$ 。

**重要性**：如果没有激活函数（即  $\varphi$  是恒等函数），神经网络只能表示简单的线性模型。加入激活函数后，网络能捕捉非线性关系，比如学生成绩中某些科目有更复杂的加权规则。

### 💡 Theorem 9.1：万能近似定理

1. 假设输入空间  $\mathcal{X} \subseteq \mathbb{R}^M$  是compact（紧致的）的（即，输入空间是有界且闭合的）。

2. 激活函数  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  是连续的。

那么，神经网络的假设空间  $\mathcal{M}_\varphi$  在连续函数空间  $C(\mathcal{X})$  中是dense（稠密），当且仅当 激活函数  $\varphi$  不是一个多项式。

### 💡 紧致集 ( Compact Sets )

紧致集  $\mathcal{X} \subseteq \mathbb{R}^M$  是一个同时满足以下条件的集合：

1. 有界 (Bounded)：存在一个常数  $c > 0$ ，使得集合中的所有点  $x$  都满足  $|x| < c$ 。

2. 闭 (Closed)：集合包含它的边界点。例如，如果一个点列  $b_n$  的极限是  $b_\infty$ ，那么  $b_\infty$  必须也在集合中。

○ 有界：集合是有限范围的，不能无限扩展。

○ 闭：集合包括所有的边界点，没有遗漏的“空洞”。

### 1. 例子：

○ 闭区间  $[a, b]$  是紧致的，例如  $[1, 3]$ 。

○ 半径  $c$  的闭球  $x : |x| \leq c$  是紧致的。

○ 有限点集合  $x_1, x_2, \dots, x_N$  也是紧致的。

### 2. 反例：

○ 开区间  $(a, b)$  不是紧致的，因为它不包含边界点  $a$  和  $b$ 。

○ 无界区间  $[a, \infty)$  也不是紧致的，因为它没有边界。

💡 在神经网络训练中，我们通常假设输入数据  $\mathcal{X}$  是紧致的，例如：

○ 图片像素值在  $[0, 255]$  范围内。

○ 学生考试分数在  $[0, 100]$  范围内。

紧致性保证了输入空间是有限的

---

## ⚠ 多项式激活函数的限制

### 💡 为什么激活函数不能是多项式？

假设激活函数  $\varphi(x)$  是一个  $n$  次多项式：

$$\varphi(x) = \sum_{j=0}^n \alpha_j x^j$$

对于任意输入  $x$ ，我们神经网络的输出可以表示为：

$$h(x) = \sum_{i=1}^p w_i^{(2)} \varphi \left( \sum_{m=1}^M W_{i,m}^{(1)} x_m + b_i^{(1)} \right)$$

展开后，最终结果仍然是一个  $n$  次多项式的组合，这意味着神经网络本质上仍然是一个多项式。

#### 💡 问题的根源

多项式的阶数是固定的（最大为  $n$ ），即使神经网络增加隐藏层宽度（增加  $p$ ），它的本质仍然是一个有限阶多项式。这会导致以下问题：

1. 无法逼近非多项式函数：例如，正弦函数  $\sin(x)$  或阶跃函数都无法通过有限阶多项式很好地拟合。

2. 过于局限：多项式对函数的拟合能力有限，不能表现出神经网络真正的灵活性。

## 💡 激活函数的选择

○ **ReLU**（线性整流单元）： $\varphi(x) = \max(0, x)$ 。

○ **Sigmoid**： $\varphi(x) = \frac{1}{1+e^{-x}}$ 。

---

## 万能近似定理的推广

### 不连续函数的逼近

神经网络不仅可以逼近连续函数 ( $C(\mathcal{X})$ )，还可以逼近某些不连续函数，比如定义在  $L^2(\mathcal{X}, \mu_{\mathcal{X}}, \mathcal{Y})$  空间中的函数。

○  **$L^2$  函数空间：**包含了平方可积的函数，这些函数可能在某些点有跳跃或不连续，但整体上可以通过积分定义均方误差。

○ **实际意义：**比如阶跃函数（如 Heaviside 函数）在图像分割或分类中可能很重要，尽管它是不连续的。

### 适用于非紧致空间的推广

当输入空间  $\mathcal{X}$  不紧致（比如  $\mathcal{X} = \mathbb{R}^M$ ）时，万能近似定理依然可以适用。这种情况下，我们需要改变对稠密性的度量方式，放宽对“全局一致性”的要求，改用局部测度（比如积分范数）。

○ **实际意义：**神经网络常用来处理具有无限范围的输入（如时间序列中的时间步、自然数序列等），这些数据空间是非紧致的。

## 示例：Example 1

### 关于激活函数的讨论

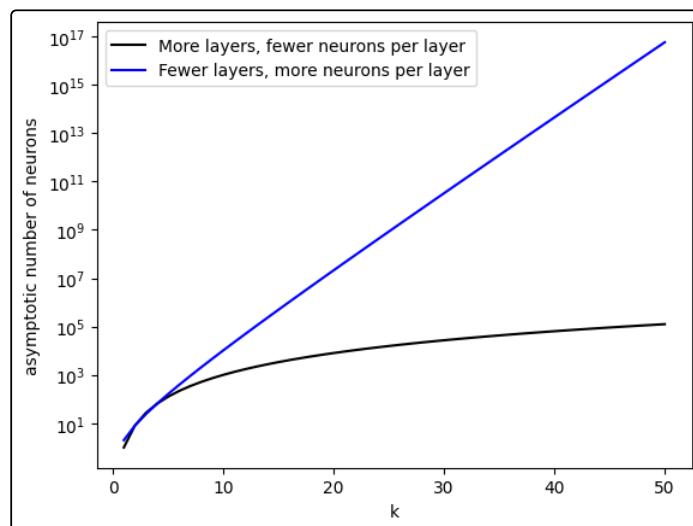
例子中指出，大多数激活函数（如 ReLU、Sigmoid）满足 [定理 9.1](#) 的假设，然而 Heaviside 函数是一个特殊的例子：

○ **Heaviside 函数：** $H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$  它是不连续的（在  $x = 0$  点有跳跃）。

尽管 Heaviside 函数不满足连续性假设，它依然可以在  $L^2$  空间中逼近一些目标函数，表明神经网络对离散或分类问题的潜力。

### 网络深度与神经元数量的比较

下图展示了浅层神经网络和深层神经网络在不同深度下的渐近神经元数量关系：



1. 黑线（更多层，较少神经元）：随着层数增加，单层的神经元数量增长较慢。

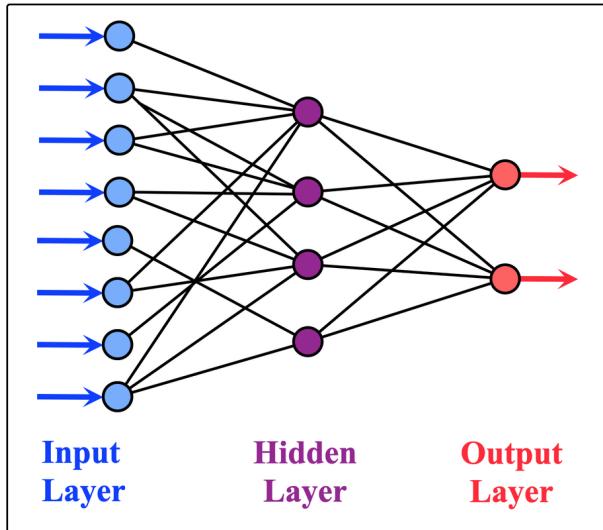
2. 蓝线（较少层，更多神经元）：如果减少层数，则每层需要更大量的神经元才能达到相同的逼近效果。

根据上图的结论我们得出：相比于增加单层神经元的数量，通过增加层数来达到逼近效果通常更高效，比如说在图像分类中，深层网络可以通过逐步提取低级到高级的特征，更易于拟合复杂模式。并且随着层数 ( $k$ ) 增加，深层神经网络所需的总神经元数量增长较慢。

虽然万能近似定理说明了只有两层的神经网络（单隐藏层网络）可以逼近任何连续函数，但实际应用中，深度神经网络（更深的层数）更加高效且实用。

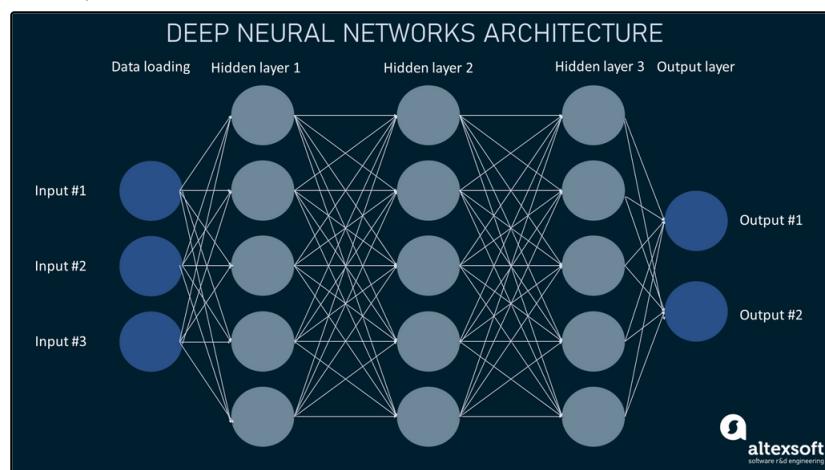
### 💡 深度与宽度的关系

- 浅层神经网络 (shallow neural network) :



浅层神经网络在输入层和输出层之间只有一个（或几个）隐藏层。输入层接收数据，隐藏层处理数据，最后一层产生输出。与深度神经网络相比，浅层神经网络更简单、更容易训练，计算效率更高，深度神经网络可能有数十层中的数千个隐藏单元。浅层网络通常用于较简单的任务，例如线性回归、二元分类或低维特征提取。

- 需要非常大的神经元数量（单层内的宽度）才能逼近复杂的函数。
- 对于某些函数，浅层网络的神经元数量可能以指数级 ( $\Omega(2^k)$ ) 增长。
- 深层神经网络 (deep neural network) :



- 增加网络的深度（层数）后，只需要较少的神经元（宽度）即可实现相同的逼近效果。
- 比如，对于深度为  $\Theta(k^3)$  的网络，单层只需要  $\Theta(1)$  的神经元就可以实现相同功能。

- $O(\cdot)$ : 表示“至多是这个数量级”，即上界。
- $\Omega(\cdot)$ : 表示“至少是这个数量级”，即下界。
- $\Theta(\cdot)$ : 表示“正好是这个数量级”。

Telgarsky (2016) 证明了：

- 深度网络更擅长表达复杂函数，浅层网络无法有效地表达这些函数。

- 浅层网络在实践中效率低下，因为它需要更多的总神经元，导致内存和训练成本过高。
- 

## 复杂函数的表示

深度网络的多层结构可以逐步提取特征，比如：

### 1. 浅层网络：

- 试图用一层直接表达复杂的函数关系，容易导致模型过宽（需要的神经元数量过多）。

### 2. 深层网络：

- 第一层提取低级特征（如边缘或局部信息）。
- 中间层组合低级特征形成更复杂的模式（如形状、轮廓）。
- 最后一层将模式映射到输出（如分类结果）。

## 内存和计算效率

浅层网络虽然在理论上可以逼近任意函数，但它的参数数量（即神经元数量）会随着问题复杂度爆炸性增长。这会导致：

### 1. 内存需求高：存储这些参数需要巨大的计算资源。

### 2. 训练成本高：训练这些参数需要大量数据和时间。

相比之下，深层网络通过引入更多的层次，可以以更少的参数实现相同或更优的逼近效果。



## Convex optimisation

---

优化的核心是通过调整模型参数  $w$  来最小化经验误差，从而提高模型的性能。

## 目标函数与优化问题的定义

---

优化问题通常定义为：

$$\min_{w \in \mathcal{W}} \frac{1}{N} \sum_{n=1}^N L(H(x_n, w), y_n)$$

- $x_n$ : 输入数据点。
- $y_n$ : 数据对应的标签。
- $H(x_n, w)$ : 模型预测结果。
- $L(H(x_n, w), y_n)$ : 损失函数，衡量预测值和真实值之间的差距。
- $\mathcal{W}$ : 参数空间，表示  $w$  可以取的所有可能值。

### ④ 目标

通过选择适当的参数  $w$  来最小化目标函数，即损失的平均值。

---

## 基本概念

### 目标函数 ( Objective Function )

定义为  $\Phi(w)$ , 描述了优化问题的目标。我们希望找到一个  $w^*$ , 使得:

$$\Phi(w^*) \leq \Phi(w), \quad \forall w \in \mathcal{W}$$

其中,  $w^*$  被称为目标函数的全局最优解 (global minimiser)。

### 全局最优解与局部最优解

#### ○ 全局最优解 (Global Minimiser) :

○ 全局最优解是在整个参数空间  $\mathcal{W}$  内达到的最低点。

○ 满足  $\Phi(w^*) \leq \Phi(w)$ , 对所有  $w \in \mathcal{W}$  都成立。

#### ○ 局部最优解 (Local Minimiser) :

○ 局部最优解仅在一个局部区域 (即  $w$  的某个邻域内) 是最低点。

○ 存在一个  $\epsilon > 0$ , 使得  $\Phi(w') \leq \Phi(w)$ , 对所有  $|w - w'| < \epsilon$  都成立。

### 局部与全局的区别

○ 局部最优解不一定是全局最优解。例如, 目标函数有多个“山谷”, 每个山谷的最低点是局部最优解, 但只有最深的山谷是全局最优解。

### 开集与开放球

在定义局部最优解时引入了“开集”和“开放球”的概念:

#### 1. 开集 (Open Set) :

○ 一个集合  $A \subseteq \mathbb{R}^K$  是开集, 如果对任意  $x \in A$ , 存在一个  $\epsilon > 0$ , 使得以  $x$  为中心, 半径为  $\epsilon$  的开放球  $B^\circ(x, \epsilon)$  完全包含在  $A$  内。

#### 2. 开放球 (Open Ball) :

○ 以  $x$  为中心,  $\epsilon$  为半径的开放球定义为:  $B^\circ(x, \epsilon) = \{x' \in \mathbb{R}^K : \|x - x'\| < \epsilon\}$

局部最优解在数学上可以被理解为: 在某个开集中是全局最优解。

## Theorem 9.2: 局部最优解的梯度条件

以上定理给出了局部最优解的一条必要条件:

如果  $w^* \in W$  是  $\Phi$  的局部最优解, 那么  $\nabla \Phi(w^*) = 0$

○  $\Phi$ : 目标函数。

○  $\nabla \Phi(w^*)$ : 目标函数  $\Phi$  在  $w^*$  处的梯度 (即偏导数向量)。

○ 这个条件的意义是: 在局部最优解点, 目标函数的梯度必须为零。

## 💡 梯度为零的意义

### ○ 梯度为零的直观解释：

梯度  $\nabla \Phi(w)$  指向目标函数增长最快的方向。在局部最优点，函数的变化率为零，因此梯度必须为零。

### ○ 例子：

- 在一维情况下，局部最优点（极值点）是函数曲线的切线斜率为零的地方。例如， $f(x) = x^2$  在  $x = 0$  处达到最小值，其一阶导数  $f'(x) = 2x$  在  $x = 0$  处为零。
- 

### 梯度为零是必要但不充分条件

#### ○ 梯度 $\nabla \Phi(w) = 0$ 表示 $w$ 是一个候选点，可能是：

1. 局部最小值：函数在该点附近是最低的。
2. 局部最大值：函数在该点附近是最高的。
3. 鞍点：函数在某些方向上增加，在另一些方向上减少。

## 💡 例子

假设目标函数  $\Phi(w) = w^3$ ：

- 一阶导数  $\Phi'(w) = 3w^2$ 。
  - 在  $w = 0$  时，梯度为零，但  $w = 0$  是一个 turning point，因为：
    - 当  $w > 0$  时， $\Phi(w)$  增加。
    - 当  $w < 0$  时， $\Phi(w)$  减少。
- 

## 💡 局部 vs. 全局最优

- 梯度是一个局部性质：它只反映了函数在某点附近的变化，而无法判断该点是否是全局最优点。
- 解决方法：如果目标函数  $\Phi$  是凸函数，那么局部最优解必定是全局最优解。

## ⚠ Theorem 9.3: 凸优化中的局部最优与全局最优

### (i) 如果目标函数是凸函数，局部最优解也是全局最优解

- 假设  $\mathcal{W} \subseteq \mathbb{R}^K$  是一个凸集合， $\Phi : \mathcal{W} \rightarrow \mathbb{R}$  是一个凸函数。
- 那么：
  - 如果  $w' \in \mathcal{W}$  是  $\Phi$  的局部最优解（即  $w'$  在局部区域内达到最小值），那么  $w'$  也是  $\Phi$  的全局最优解。

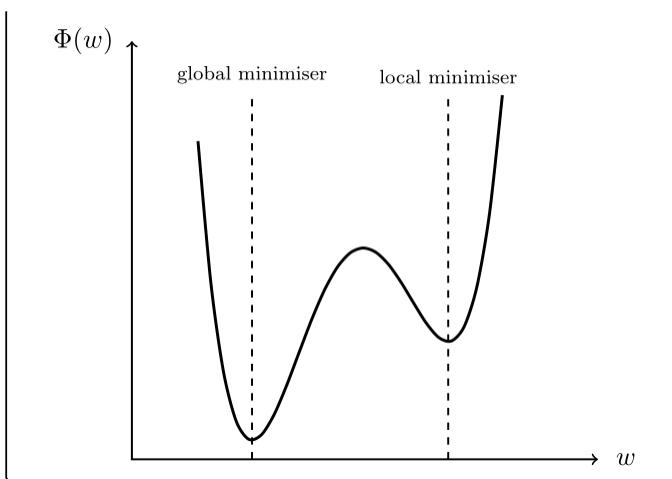
### (ii) 梯度条件与全局最优

- 如果  $\mathcal{W}$  是开集，并且目标函数  $\Phi$  是连续可微的（continuously differentiable），
- 那么，如果  $w'' \in \mathcal{W}$  满足  $\nabla \Phi(w'') = 0$ ，则  $w''$  是  $\Phi$  的全局最优解。

### 💡 定理 (ii) 梯度条件的意义

- 如果目标函数  $\Phi$  是凸函数，且在  $w''$  处的梯度为零 ( $\nabla \Phi(w'') = 0$ )，那么该点  $w''$  就是全局最优解。
  - 原因是：对于凸函数，梯度为零的点（即函数的平坦点）一定是最低点。
- 

下图展示了一个典型函数  $\Phi(w)$  的局部最优解和全局最优解：



[We sketch a function  \$\Phi\$  and illustrate its local and global minimisers.](#)

- 全局最优解 (global minimiser) : 在整个定义域  $\mathcal{W}$  内, 目标函数值最低的点。
- 局部最优解 (local minimiser) : 在某个局部区域内, 目标函数值最低的点, 但可能不是全局最优。

### 💡 凸函数的特殊性质

- 凸函数的定义是: 任意两点  $w_1, w_2 \in \mathcal{W}$  和  $t \in [0, 1]$ , 都满足:  $\Phi(tw_1 + (1 - t)w_2) \leq t\Phi(w_1) + (1 - t)\Phi(w_2)$  直观来说, 凸函数的图像总是“向上弯曲”的, 不会有多个局部最优解。
- 因此, 凸函数的局部最优解必然是全局最优解。

Proof:

- 假设  $w'$  是局部最优解, 但  $w'$  不是全局最优解。
- 由于  $w'$  不是全局最优解, 存在一个点  $w^* \in \mathcal{W}$ , 使得  $\Phi(w^*) < \Phi(w')$ 。
- 利用凸性定义和构造中间点的方式, 证明  $w'$  不满足局部最优的条件, 从而得到矛盾。

#### (i) Proof:

1. 根据凸性的定义, 对于任意  $\lambda \in [0, 1]$ , 有:

$$\Phi(\lambda w' + (1 - \lambda)w^*) \leq \lambda\Phi(w') + (1 - \lambda)\Phi(w^*).$$

2. 因为假设  $\Phi(w^*) < \Phi(w')$ , 所以:

$$\Phi(\lambda w' + (1 - \lambda)w^*) < \Phi(w')$$

3. 这说明: 中间点  $\lambda w' + (1 - \lambda)w^*$  的目标函数值小于  $w'$  的值。

4. 接下来, 考虑  $w'$  的局部最优性:

- 对于局部最优解, 应该在某个邻域内的所有点  $w$ , 满足  $\Phi(w') \leq \Phi(w)$ 。
- 但通过上述构造的  $\lambda w' + (1 - \lambda)w^*$  属于  $w'$  的邻域 (利用  $\epsilon$  的开放球定义), 并且  $\Phi(\lambda w' + (1 - \lambda)w^*) < \Phi(w')$ , 矛盾。

5. 因此,  $w'$  必须是全局最优解。

#### (ii) Proof:

🔑 证明思路

- 利用凸性和目标函数的可微性。
  - 梯度为零的点满足一阶最优条件, 结合凸性的特性, 证明该点是全局最优解。
1. 凸性的条件: 对于可微的凸函数  $\Phi$ , 任意  $w, \hat{w} \in \mathcal{W}$ , 有:

$$\Phi(w) \geq \Phi(\hat{w}) + \langle \nabla \Phi(\hat{w}), w - \hat{w} \rangle.$$

2. 令  $\hat{w} = w''$  且  $\nabla \Phi(w'') = 0$ , 代入上式:

$$\Phi(w) \geq \Phi(w'') + \langle 0, w - w'' \rangle = \Phi(w'').$$

3. 由此可知:

$$\Phi(w) \geq \Phi(w''), \quad \forall w \in \mathcal{W}.$$

这正是  $w''$  是全局最优解的定义。

### 💡 Remark: 凸优化在损失函数和模型中的应用

#### 📁 凸优化与损失函数

许多损失函数和模型的组合会导致经验误差 (empirical error) 是关于参数  $w$  的凸函数, 这使得可以使用凸优化方法来解决问题。

○ 例子:

○ 线性回归:

- 损失函数: 均方误差 (Mean Squared Error, MSE)。
- 函数形式是关于参数  $w$  的二次函数, 是凸的。

○ 逻辑回归:

- 损失函数: 交叉熵 (Cross Entropy)。
- 函数形式是关于  $w$  的凸函数。

○ 为什么重要?

- 对于这些凸问题, 局部最优解就是全局最优解, 因此可以通过梯度下降等方法高效找到最优解。

### 💡 非凸优化问题

○ 在深度神经网络的上下文中, 经验误差通常是非凸的。

○ 损失函数的复杂性和神经网络的多层结构会导致多个局部最优点和鞍点。

○ 非凸问题的挑战:

- 很难保证找到的是全局最优解。
- 需要使用诸如随机梯度下降 (SGD) 等启发式方法找到足够好的解。

○ 例子:

- 神经网络的交叉熵损失结合复杂的多层参数模型, 通常会引入非凸性。
- 深度学习中的优化方法需要处理高维空间中的非凸问题。