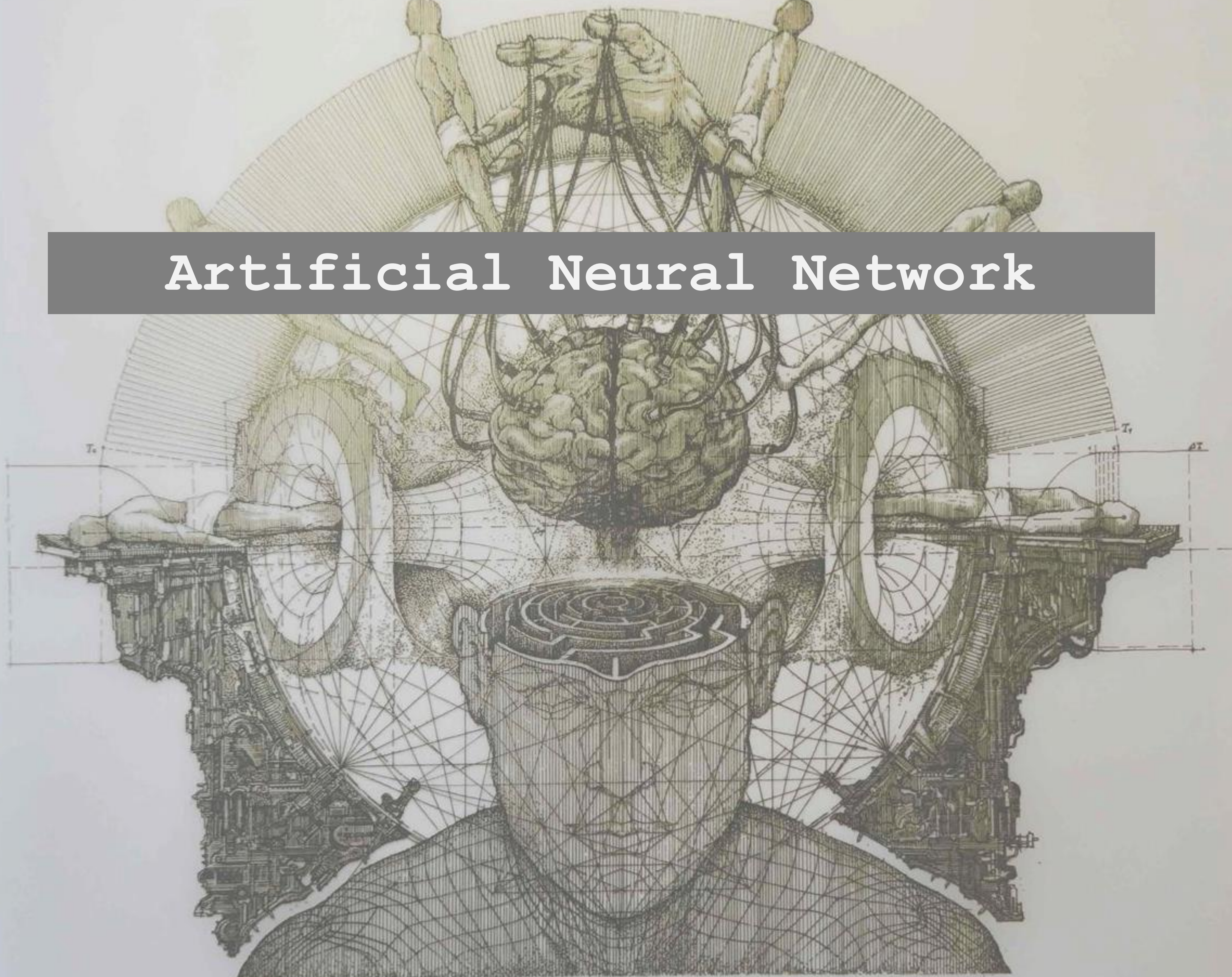
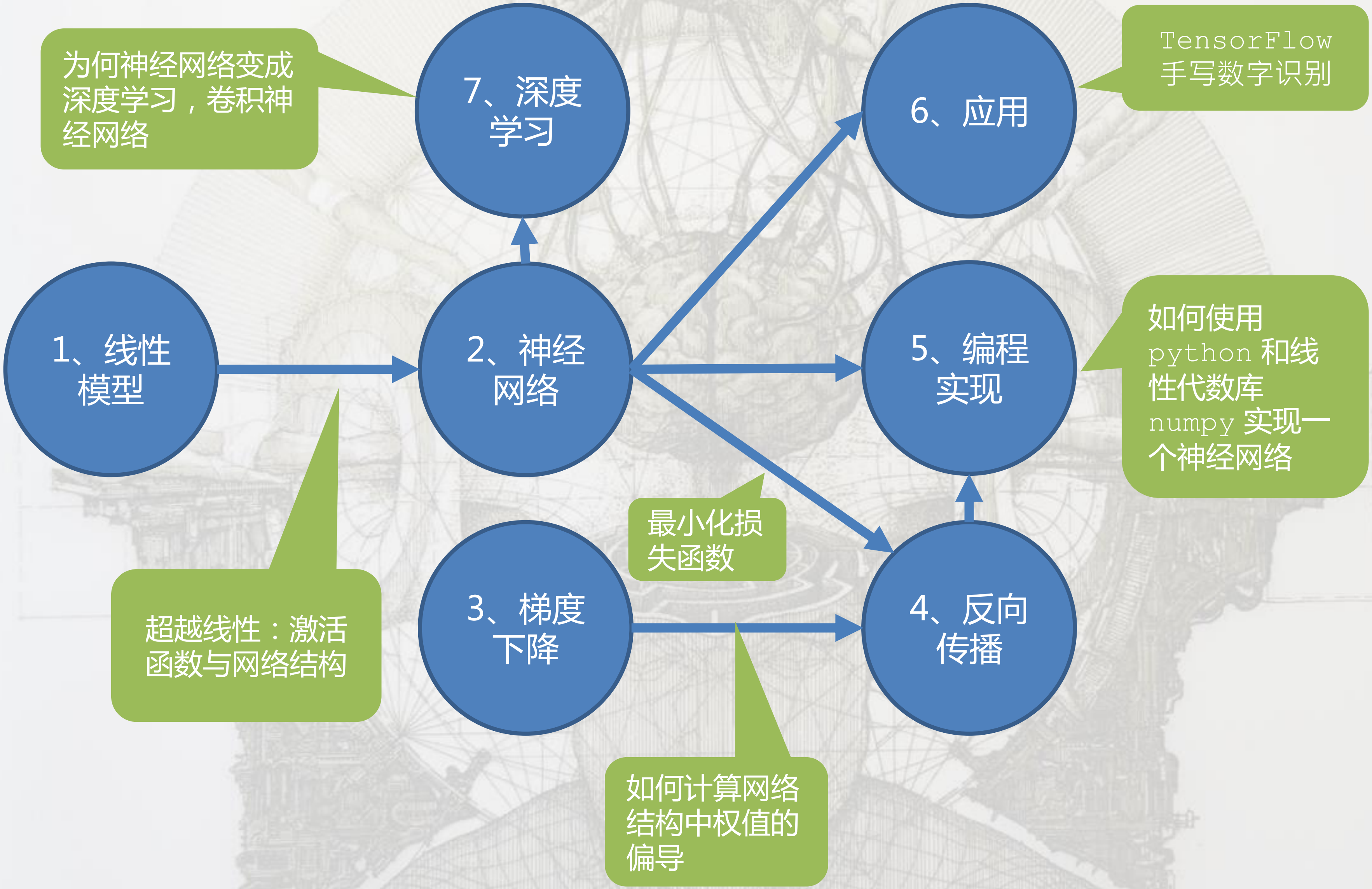
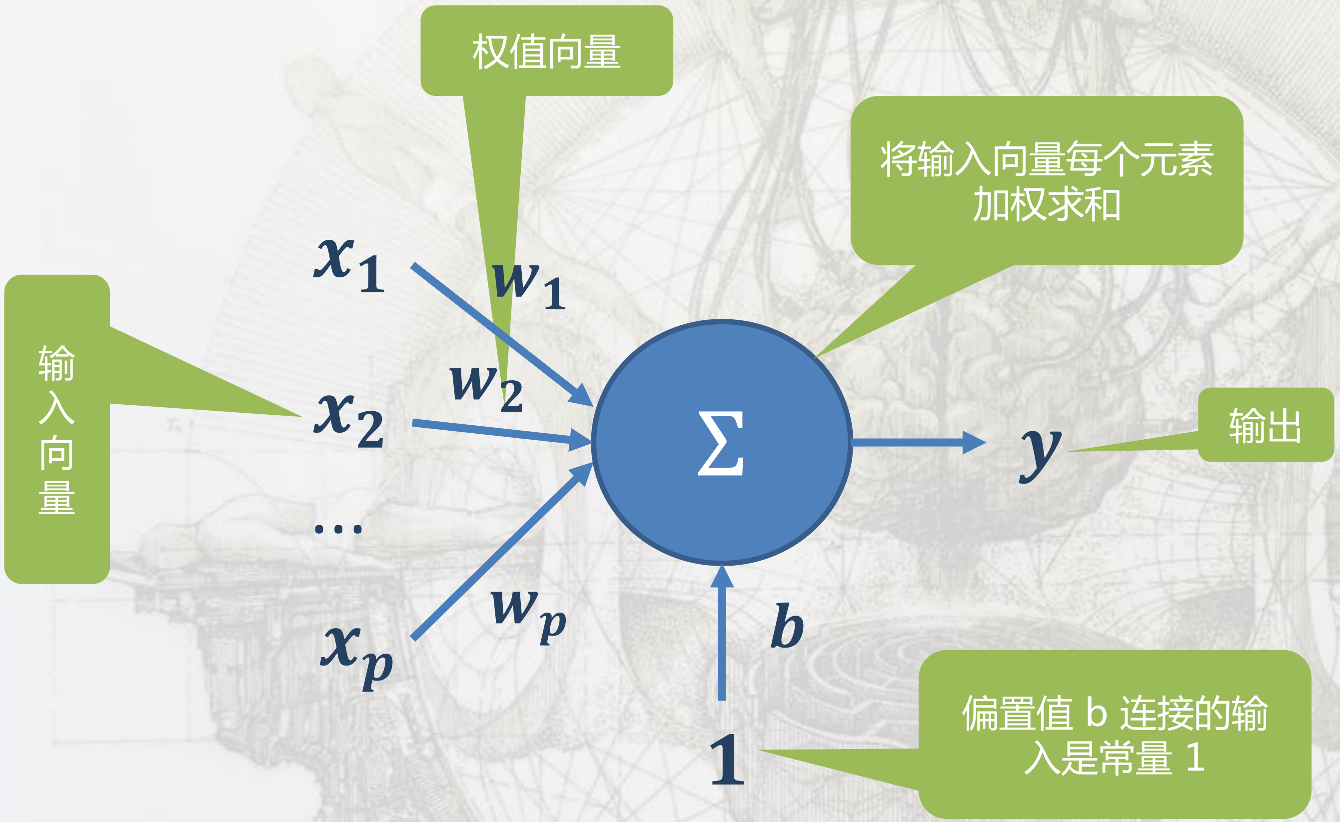
Artificial Neural Network



内容导图


为何神经网络变成深度学习，卷积神经网络
7、深度学习
6、应用
TensorFlow 手写数字识别
1、线性模型
2、神经网络
5、编程实现
如何使用 python 和线性代数库 numpy 实现一个神经网络
最小化损失函数
3、梯度下降
4、反向传播
超越线性：激活函数与网络结构
如何计算网络结构中权值的偏导

1、线性模型



最小二乘线性回归

对于训练集中的样本:

$\{x_i, y_i{}^{true}\}_{i=1\ldots N}$

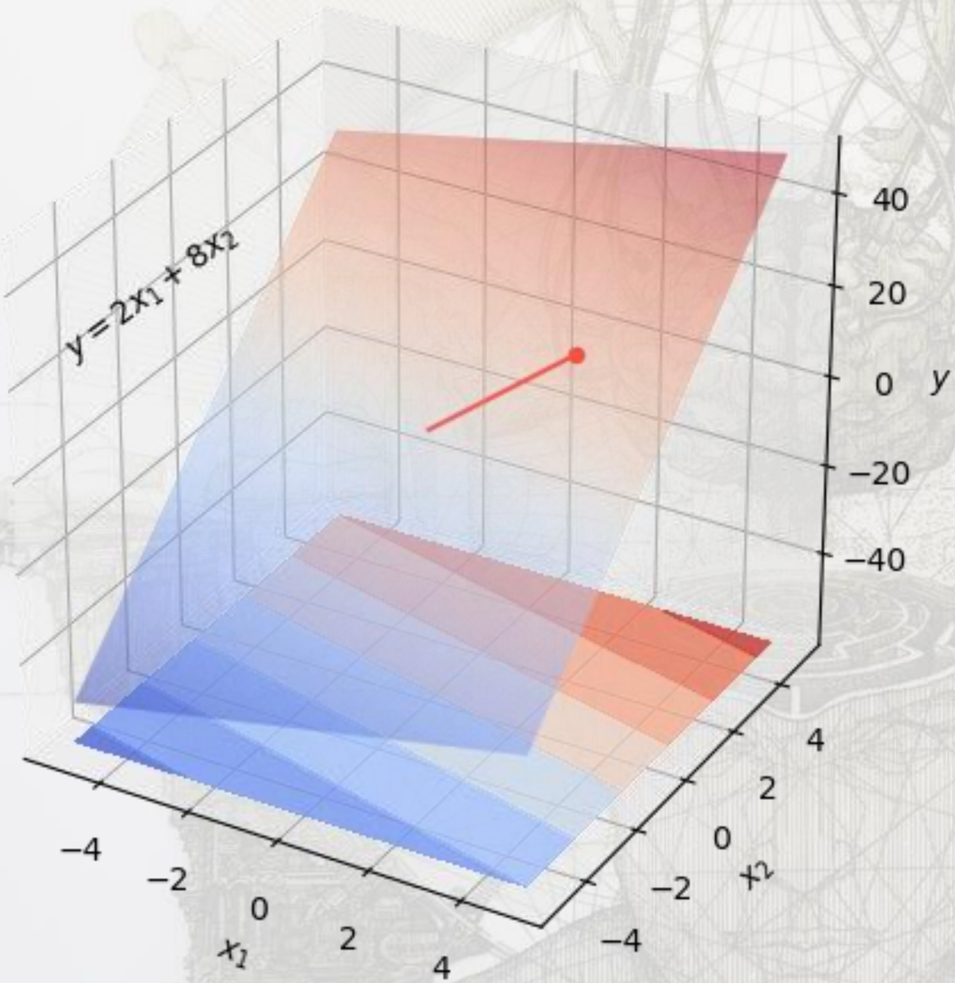
求权值向量 w 和偏置 b ，使损失函数：

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i{}^{true})^2$$

达到最小。

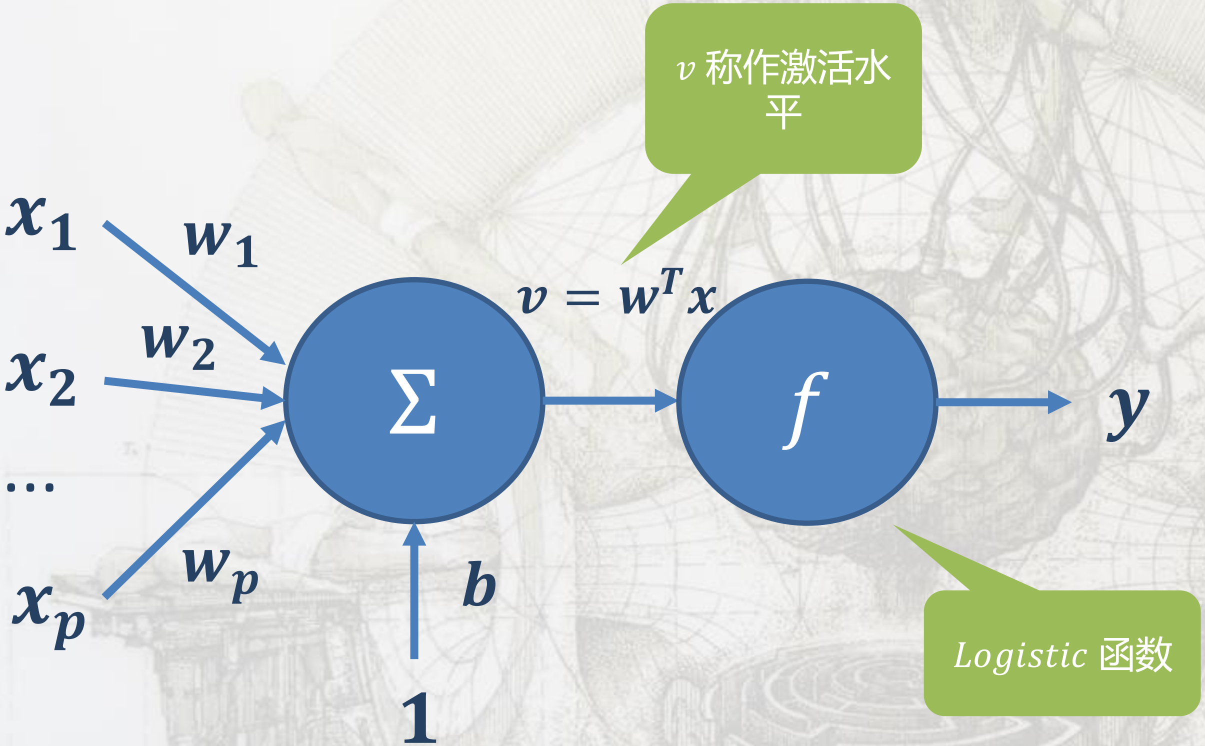
$$\boldsymbol{y} = \boldsymbol{b} + \sum_{\boldsymbol{i}=1}^{\boldsymbol{p}} \boldsymbol{w}_{\boldsymbol{i}}\boldsymbol{x}_{\boldsymbol{i}} = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b} = \|\boldsymbol{w}\| \cdot \|\boldsymbol{x}\| \cdot \cos\boldsymbol{\theta} + \boldsymbol{b}$$

1、线性模型



- 线性模型图形：超空间中一张平面；
- 响应仅在权值向量方向上变化；
- 用阈值来卡响应，得到的分界线是一条直线。

1、线性模型



$$y = Logistic(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

逻辑回归

模型的输出 y 位于区间 (0,1) 。
将其视作二分类问题正类的概率。

对于训练集中的样本:

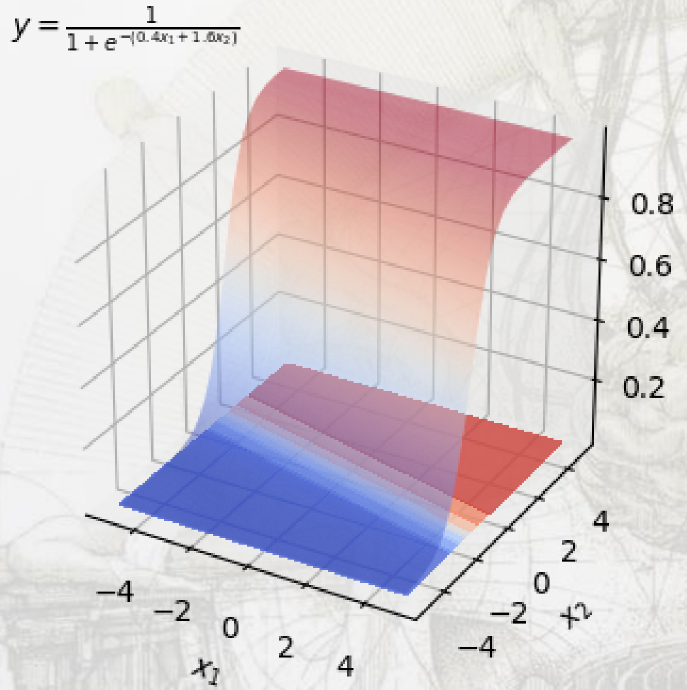
$$\{x_i, y_i^{true}\}_{i=1 \dots N}$$

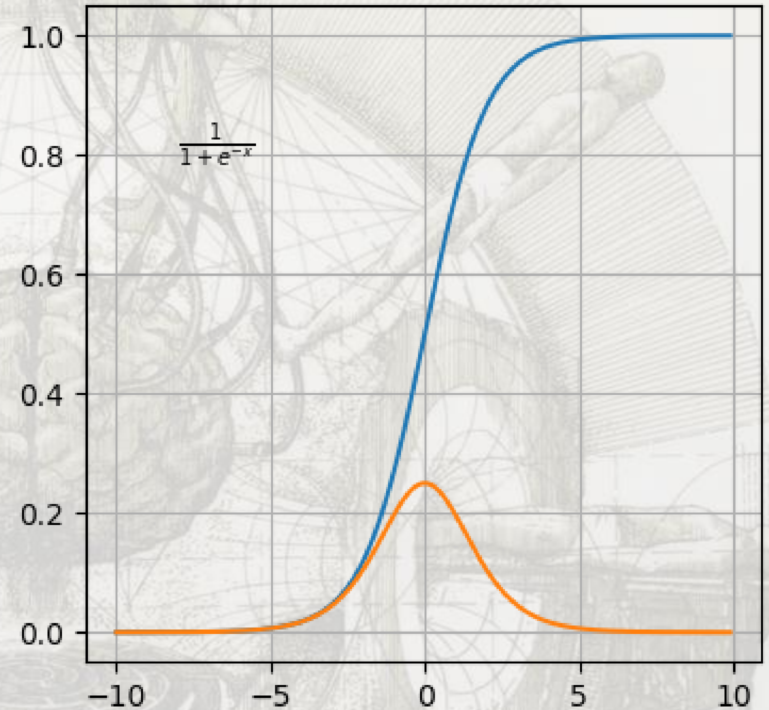
其中 y_i^{true} 取 1 或 0 ，分别标识正类和负类。求权值 w 和偏置 b ，使损失函数：

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i^{true} \, log y_i + (1 - y_i^{true}) log(1 - y_i)$$

（*交叉熵，也是对数似然的相反数*）达到最小。

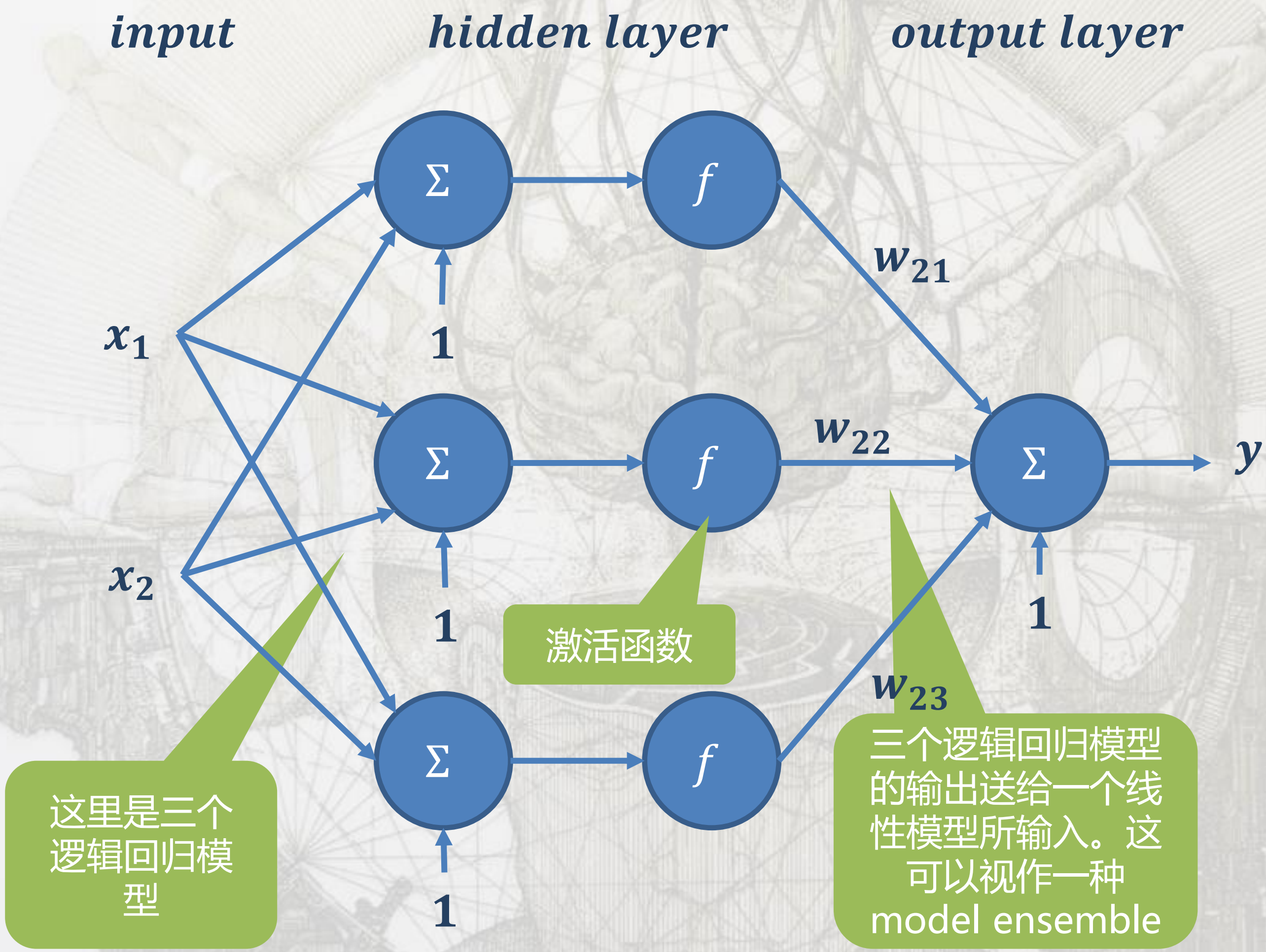
1、线性模型





- 响应值仍然只沿着权值向量 w 的方向变化，但变化呈非线性（*Sigmoid 曲线，右图*）；
- 用阈值来卡响应值，得到的分界线仍然是一条直线；
- *Logistic* 函数（ *Sigmoid 曲线*）在 0 点附近接近线性，在远离 0 点接近常量（*导数近似于 0*）。

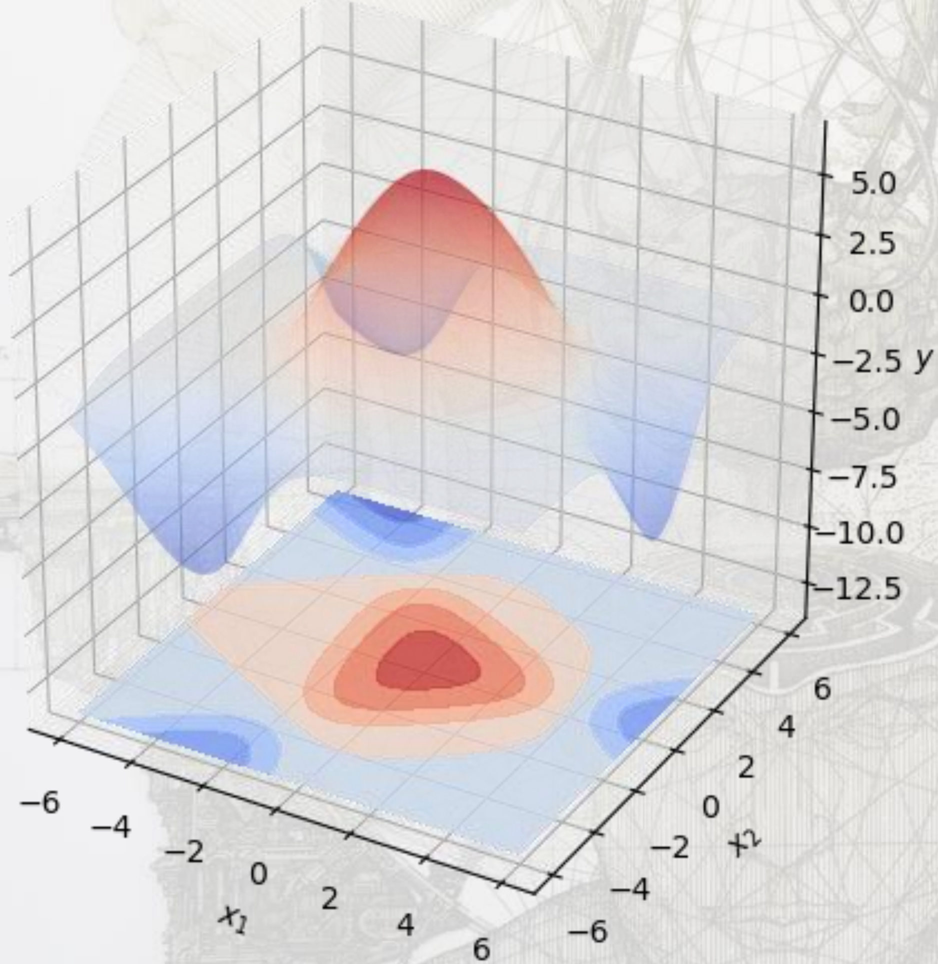
2、神经网络



input
hidden layer
output layer
x_1
x_2
Σ
f
1
w_{21}
w_{22}
w_{23}
y
激活函数
这里是三个逻辑回归模型
三个逻辑回归模型的输出送给一个线性模型所输入。这可以视作一种model ensemble

2、神经网络

$$y = -7.7\frac{1}{1+e^{-(0.98x_1 - 1.3x_2 - 3.5)}} + 7.4\frac{1}{1+e^{-(1.8x_1 + 0.2x_2 + 3.4)}} - 7.6\frac{1}{1+e^{-(0.67x_1 + 1.5x_2 - 3.5)}}$$



- 图形是复杂形状的曲面；
- 用阈值卡响应值，可得到非直线的分界线；
- 如果激活函数是恒等函数，则网络不能引入非线性，因为：

$$y = {w_2}^T(Wx) = (W^T w_2)^T x = w'^T x$$

仍然是一个线性函数。 w_2 是输出层的 3×1 权值向量，W 是隐藏层的 3×2 权值矩阵。

2、神经网络

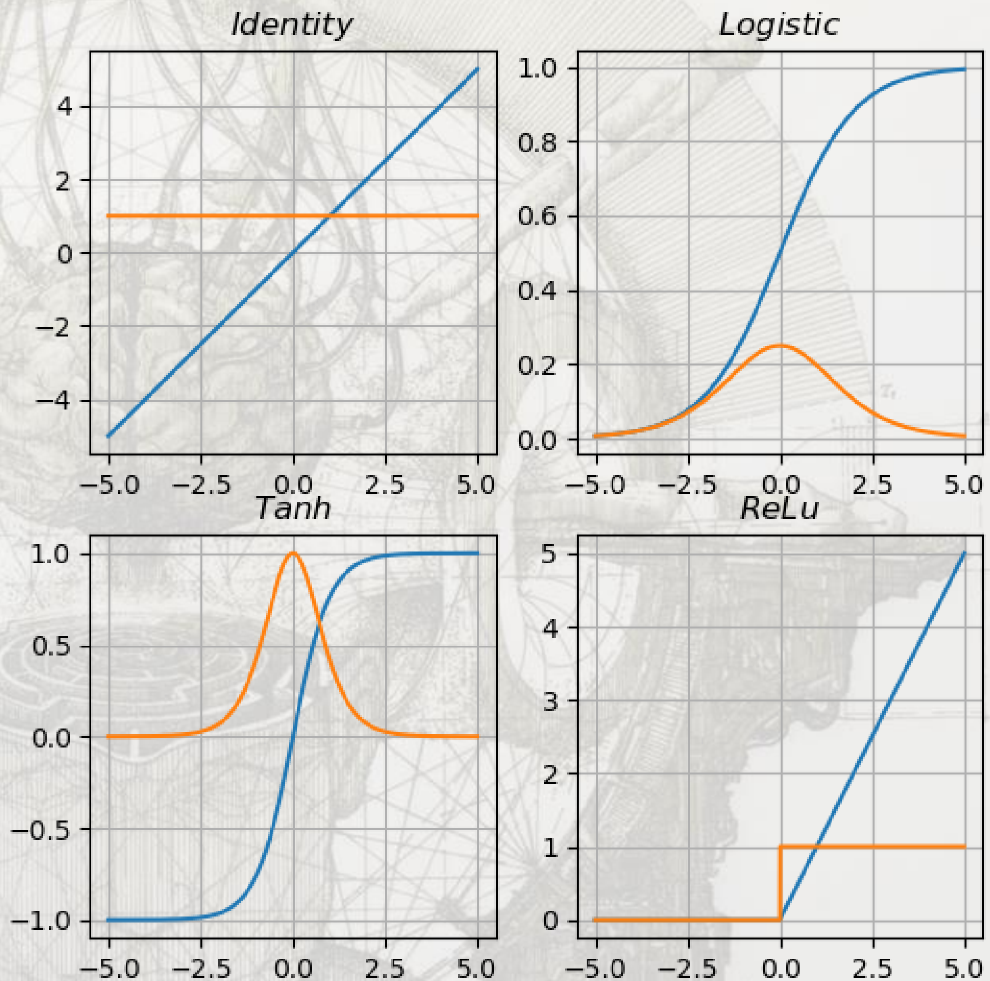
- 神经元：将输入线性组合后施加激活函数。激活函数须可导，例如：

$$Identity\colon\ f(x) = x$$

$$Logistic\colon\ f(x) = \frac{1}{1+e^{-x}}$$

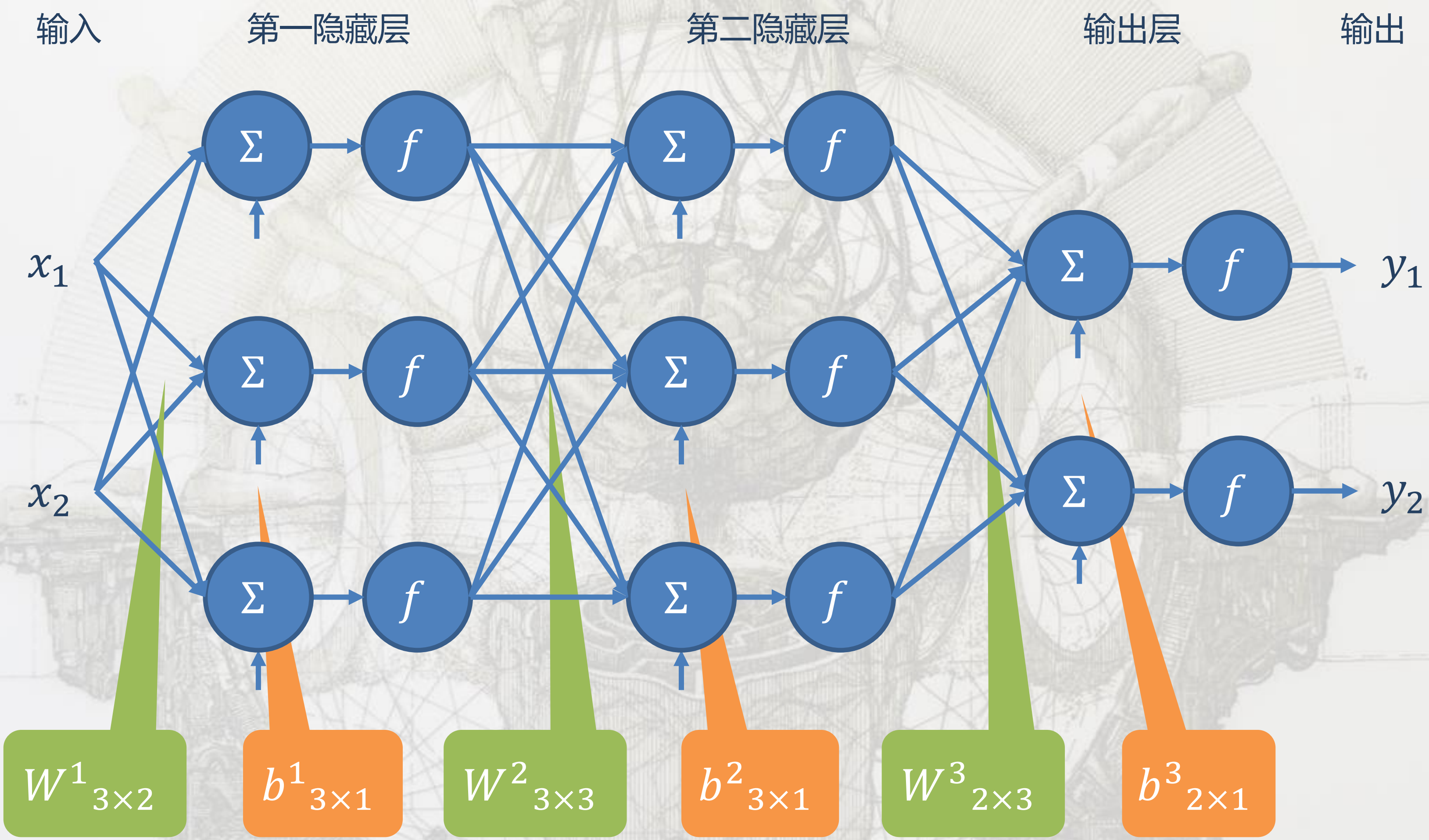
$$Tanh\colon\ f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

$$ReLu\colon\ f(x) = \begin{Bmatrix} x\ x \geq 0 \\ 0\ x < 0 \end{Bmatrix}$$



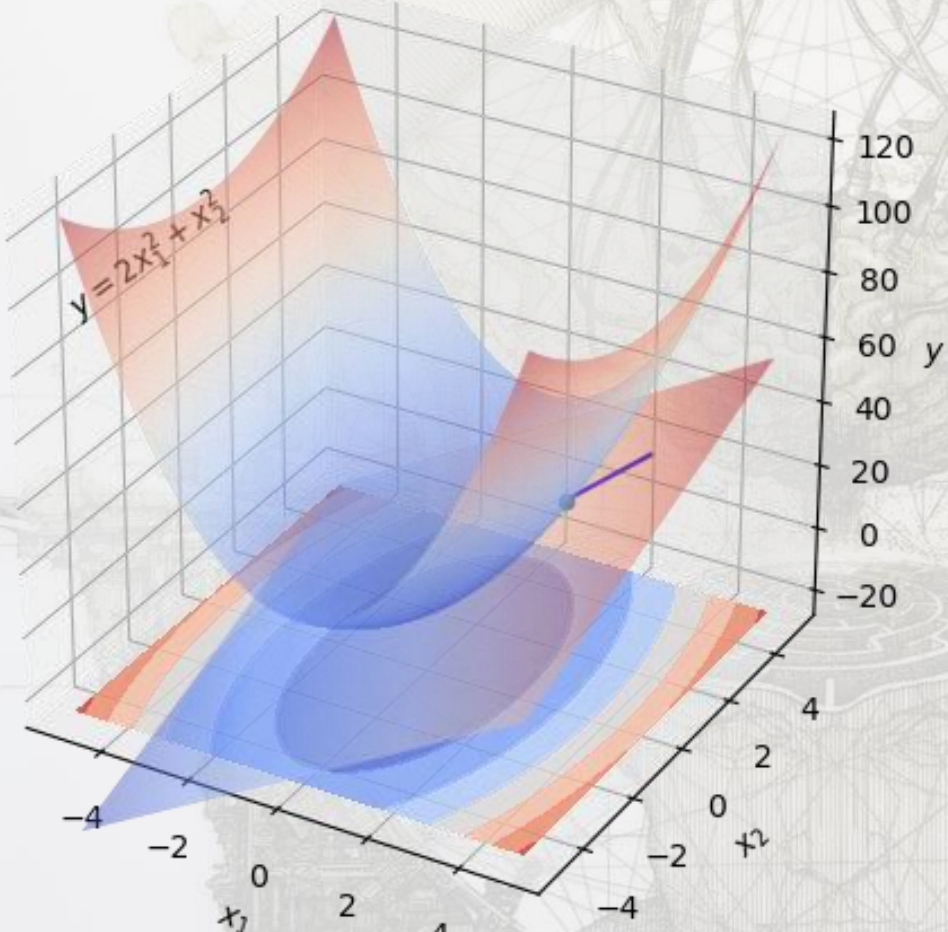
- 若干神经元形成一层，前一层的输出提供给后一层做输入。输入向量作为第一层的输入。除最后一层外都是隐藏层。即为全连接神经网络。本质是一个函数 $f\colon R^{m} \rightarrow R^{n}$

2、神经网络



$$y = f \odot (W^3 f \odot (W^2 f \odot (W^1 x + b^1) + b^2) + b^3)$$

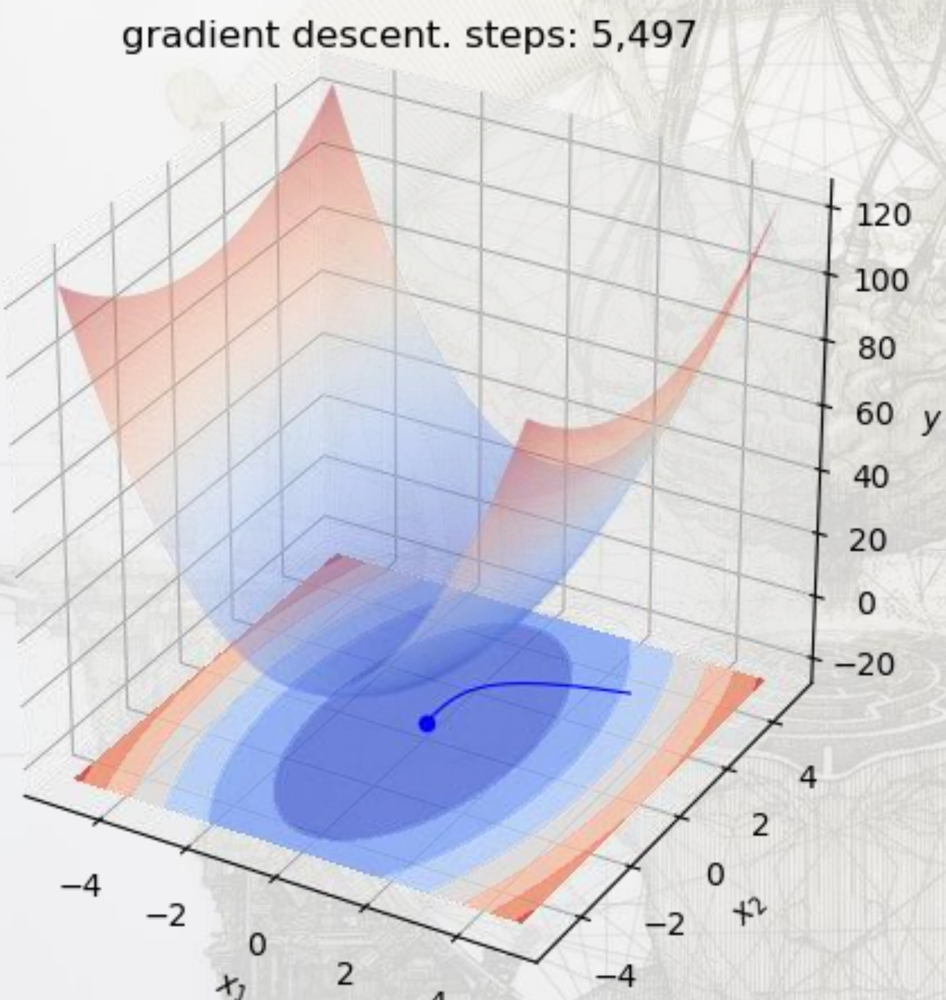
3、梯度下降



- 函数在某个点附近的一阶泰勒展开是一张平面，该平面在该点附近很好地近似了原函数；
- 函数在某个点的梯度是自变量空间中的一个向量，它指向一阶近似平面上升最快的方向，也是原函数上升最快的方向；
- 梯度向量的元素是函数在该点对各个自变量的偏导数：

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_p} \end{bmatrix}$$

3、梯度下降



- 梯度下降算法迭代公式：

$$x^{(s+1)} = x^{(s)} - \eta \nabla f\left(x^{(s)}\right)$$

- η 为称学习率 learning rate；
- 基本梯度下降法的各种变体：
  - ✓ 加入冲量
  - ✓ 学习率衰减
  - ✓ 针对自变量每一个分量取不同的步长
- wiki 介绍常用变体:

  en.wikipedia.org/wiki/Stochastic_gradient_descent

3、梯度下降

- 神经网络的训练：运用梯度下降法寻找损失函数 L 的最小值；
- 损失函数（ y *和* y^{true} *都是 m 向量*）：
  - 回归：$L(y, y^{true})=\|y-y^{true}\|^2$；
  - 多分类：对 m 个输出施加 $Softmax$ 产生 m 个 (0,1) 区间内的值：$\frac{e^{y_1}}{\sum_{j=1}^{m} e^{y_j}}, \frac{e^{y_2}}{\sum_{j=1}^{m} e^{y_j}}, \cdots \frac{e^{y_m}}{\sum_{j=1}^{m} e^{y_j}}$。

$$L(y, y^{true})=-\sum_{i=1}^{m} y_i^{true} log\left(\frac{e^{y_i}}{\sum_{j=1}^{m} e^{y_j}}\right)$$ （*交叉熵、对数似然的相反数*）

- 训练集 $\{x_i, y_i^{true}\}_{i=1 \dots N}$ 包含 N 对输入和输出向量，以网络全体权值和偏置为自变量，运用梯度下降算法求使平均损失函数 $\frac{1}{N}\sum_{i=1}^{N} L(y_i, \ y_i^{true})$ 最小的权值和偏置。
- 损失函数非凸，存在局部极小值点。
- 变体：
  - 随机梯度下降：每次迭代用一个样本的 L 计算梯度并更新；
  - mini batch ：每次迭代用 m<N 个样本计算平均 L 的梯度并更新。

3、梯度下降

- 信息熵（*Entropy*）：

$$H(p)=\mathrm{E}_p\left(log\frac{1}{p}\right)=\sum\nolimits_x plog\frac{1}{p}=-\sum\nolimits_x plog(p)$$

- K-L 散度（*Kullback-Leibler Divergence，相对熵*）：

$$KLD(\mathrm{p}||\mathrm{q})=\mathrm{E}_p\left(log\frac{p}{q}\right)=\sum\nolimits_x plog\frac{p}{q}=\sum\nolimits_x\big(plog(p)-plog(q)\big)=-\sum\nolimits_x plog(q)-H(p)$$

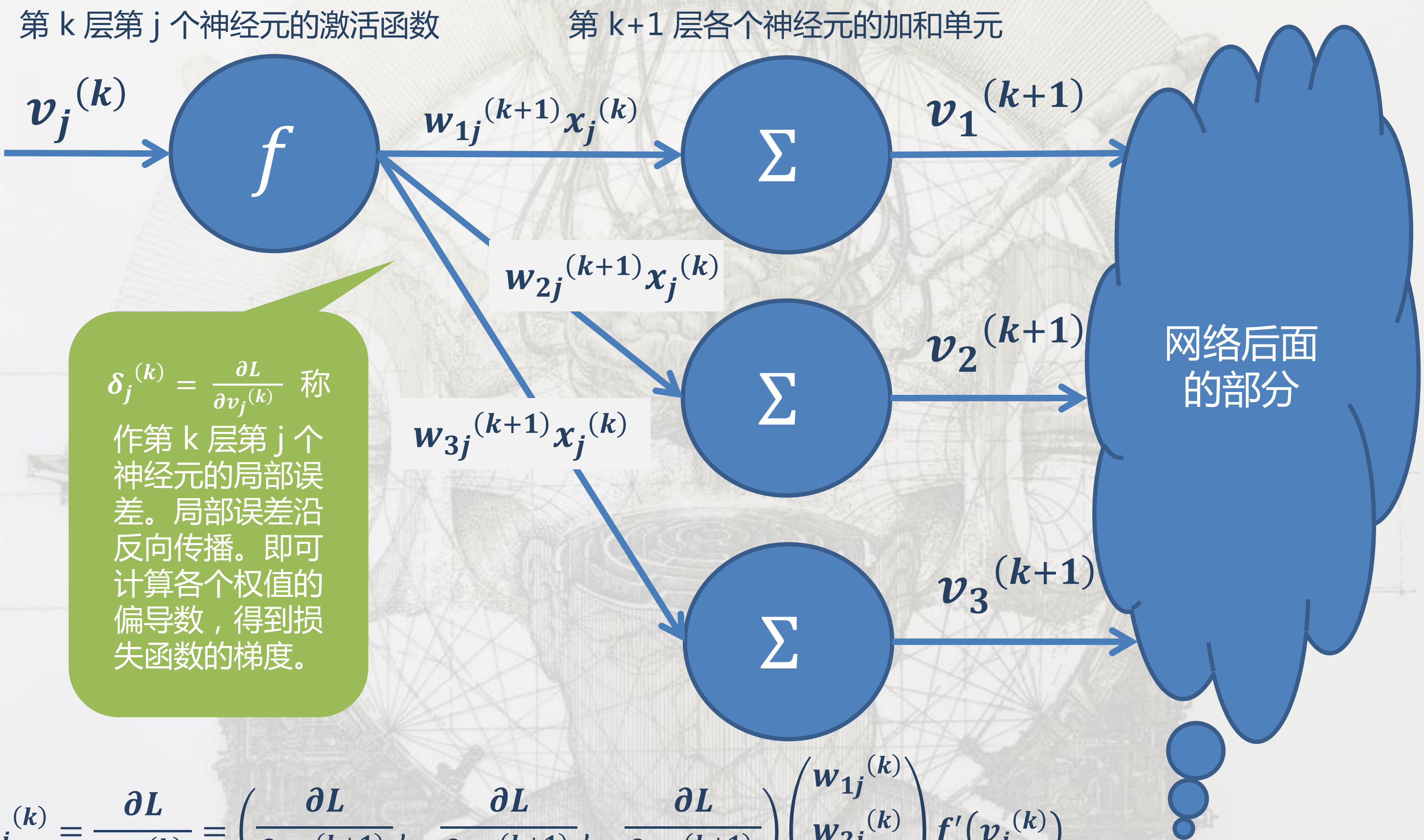
- 交叉熵（*Cross Entropy*）:

$$H(p,q)=\mathrm{E}_p\left(log\frac{1}{q}\right)=\sum\nolimits_x plog\frac{1}{q}=-\sum\nolimits_x plog(q)$$

- 所以分布 p 与 q 的交叉熵等于 p 与 q 的 K-L 散度加上分布 p 的熵：

$$H(p,q)=KLD(p||q)+H(p)$$

4、反向传播



$$\delta_j^{(k)} = \frac{\partial L}{\partial v_j^{(k)}} = \left(\frac{\partial L}{\partial v_1^{(k+1)}}, \quad \frac{\partial L}{\partial v_2^{(k+1)}}, \quad \frac{\partial L}{\partial v_3^{(k+1)}}\right) \begin{pmatrix} w_{1j}^{(k)} \\ w_{2j}^{(k)} \\ w_{3j}^{(k)} \end{pmatrix} f'\left(v_j^{(k)}\right)$$

4、反向传播

反向传播：

输出层：$\Delta^{(K)} = -(y^{true} - y)^T \begin{pmatrix} f'\left({v_1}^{(K)}\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'\left({v_{n_K}}^{(K)}\right) \end{pmatrix}$

隐藏层：$\Delta^{(k)} = \Delta^{(k+1)} W^{(k+1)} \begin{pmatrix} f'\left({v_1}^{(k)}\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f'\left({v_{n_k}}^{(k)}\right) \end{pmatrix}$

权值更新：

权值：$W^{(k)} = W^{(k)} - \eta\left(x^{(k-1)}\Delta^{(k)}\right)^T$

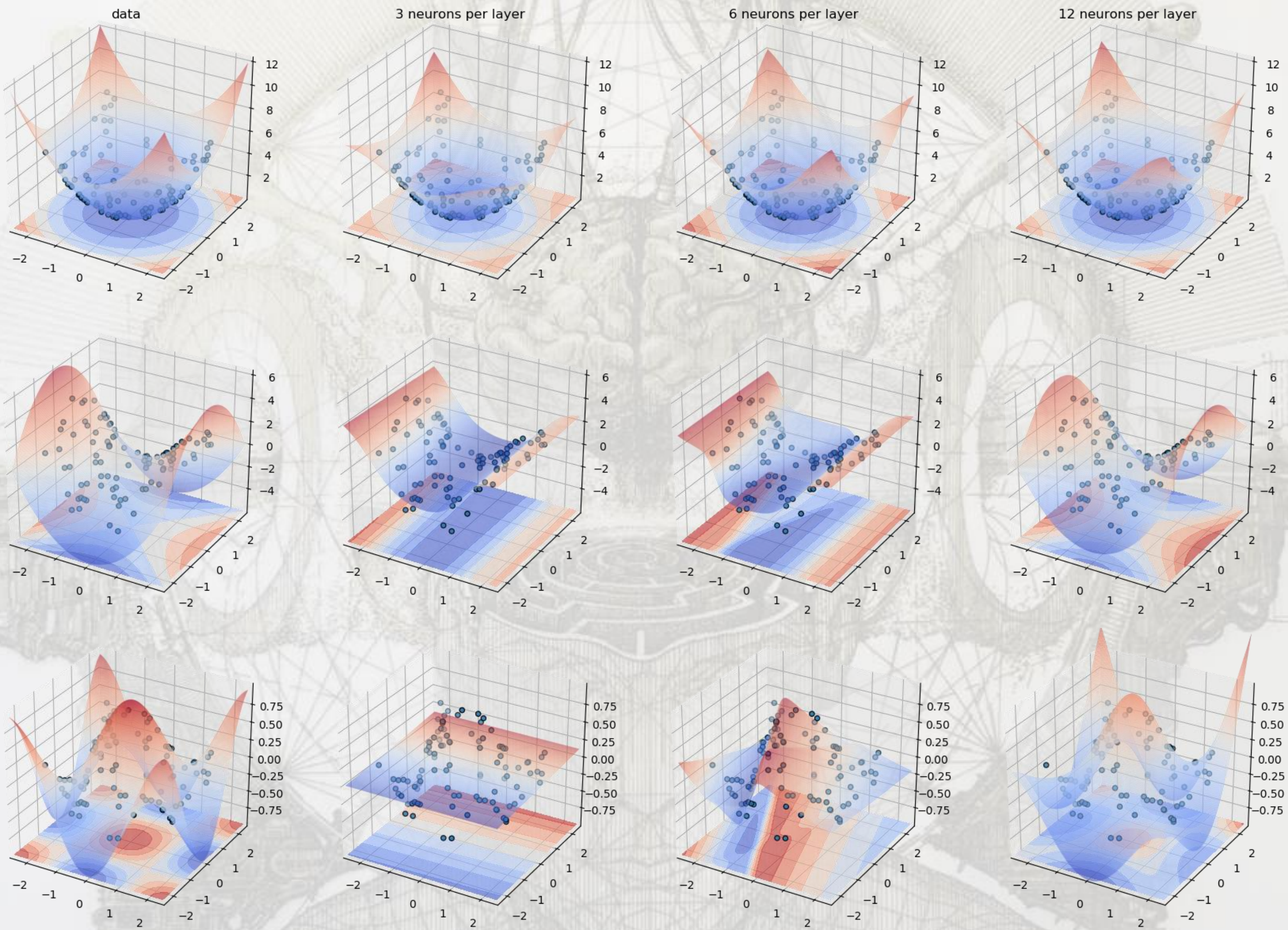
偏置：$b^{(k)} = b^{(k)} - \eta\left(\Delta^{(k)}\right)^T$

其中：

$$\Delta^{(k)} = \left(\frac{\partial L}{\partial {v_1}^{(k)}}, \frac{\partial L}{\partial {v_2}^{(k)}}, \cdots \frac{\partial L}{\partial {v_{n_k}}^{(k)}}\right)$$

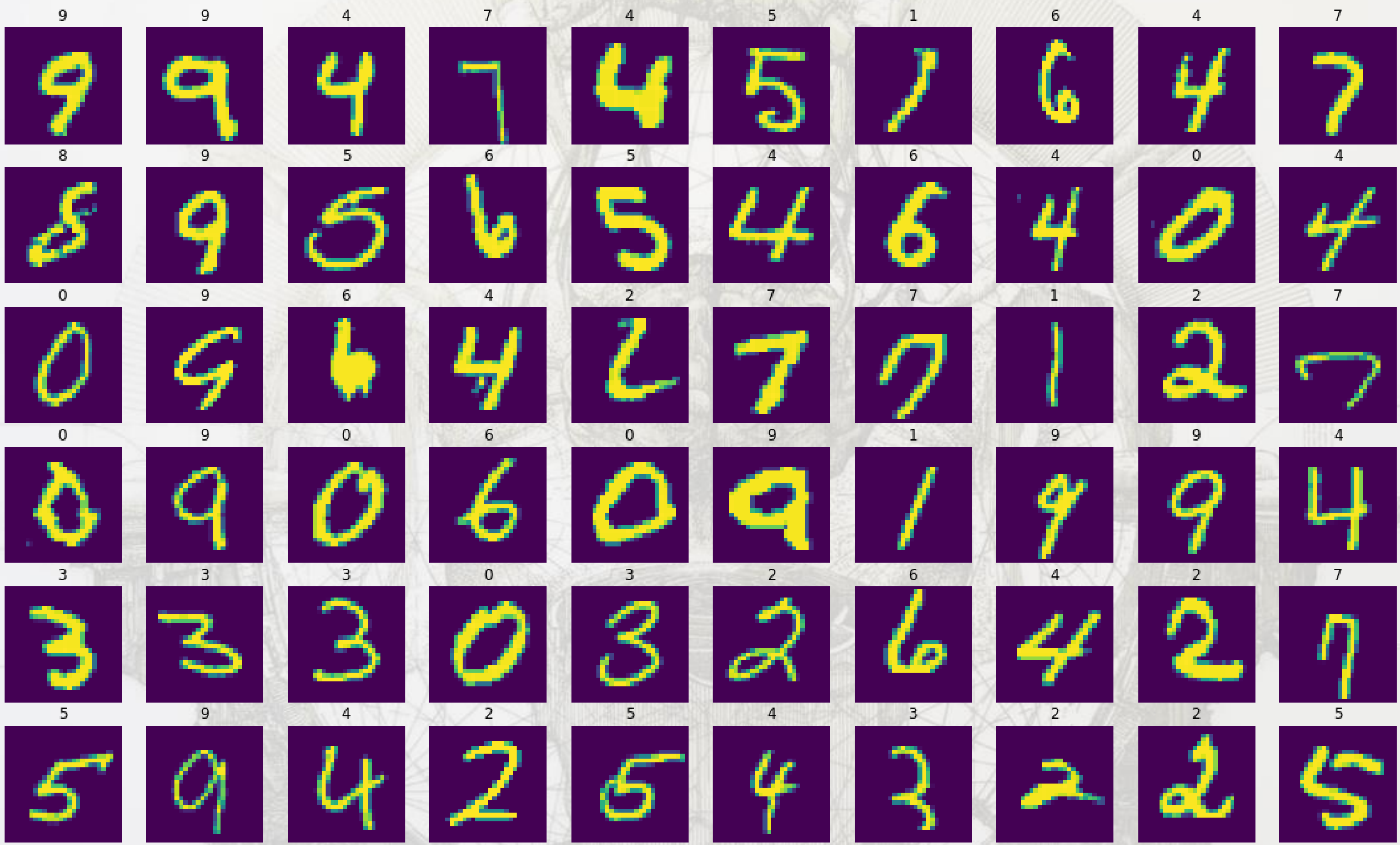
5、Python 实现

单隐藏层神经网络（*隐藏层神经元个数分别为3、6 和 12*）拟合三个函数：



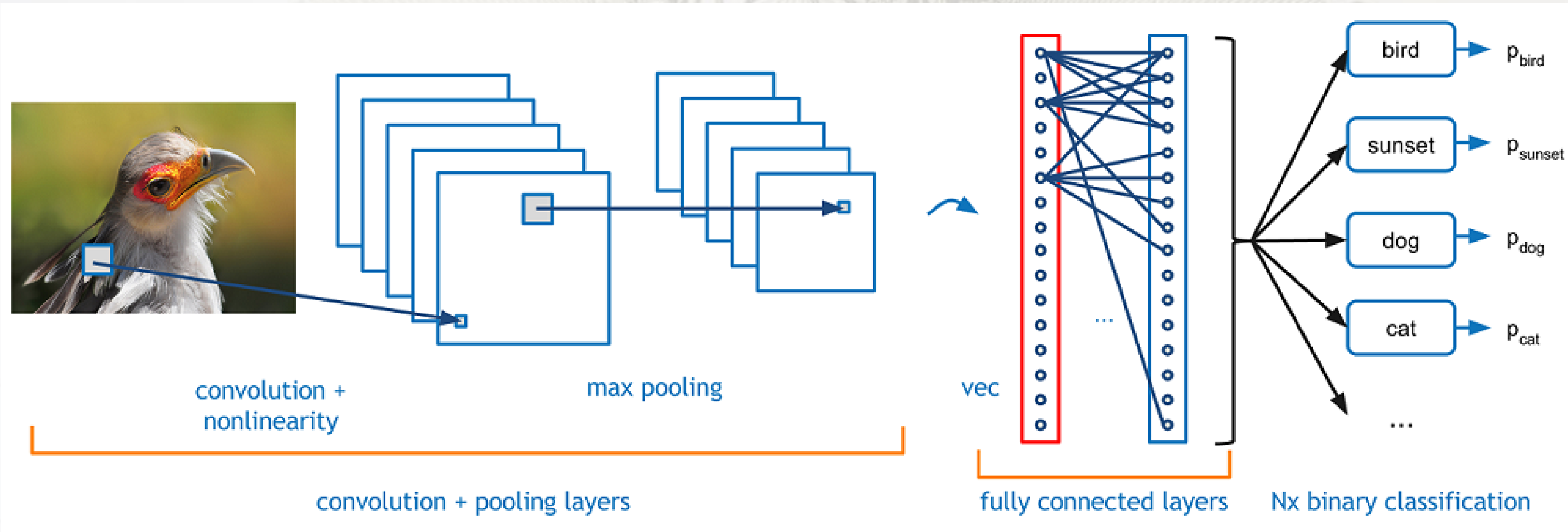
github: github.com/zhangjuefei/mentat/blob/master/mentat/model/dnn.py

6、应用：TensorFlow 手写数字识别



MNIST 手写数字样例（*此处有代码讲解*）

7、深度学习（以卷积神经网络 CNN 为例）



- 卷积层本质是二维离散卷积算子：

  $\mathrm{Conv}(F,G)=\sum_{t=-\infty}^{\infty}F(s,t)\times G(x-s,y-t)\Delta s\Delta t$ Continous: $\mathrm{Conv}(f,g)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(s,t)g(x-s,y-t)dsdt$；

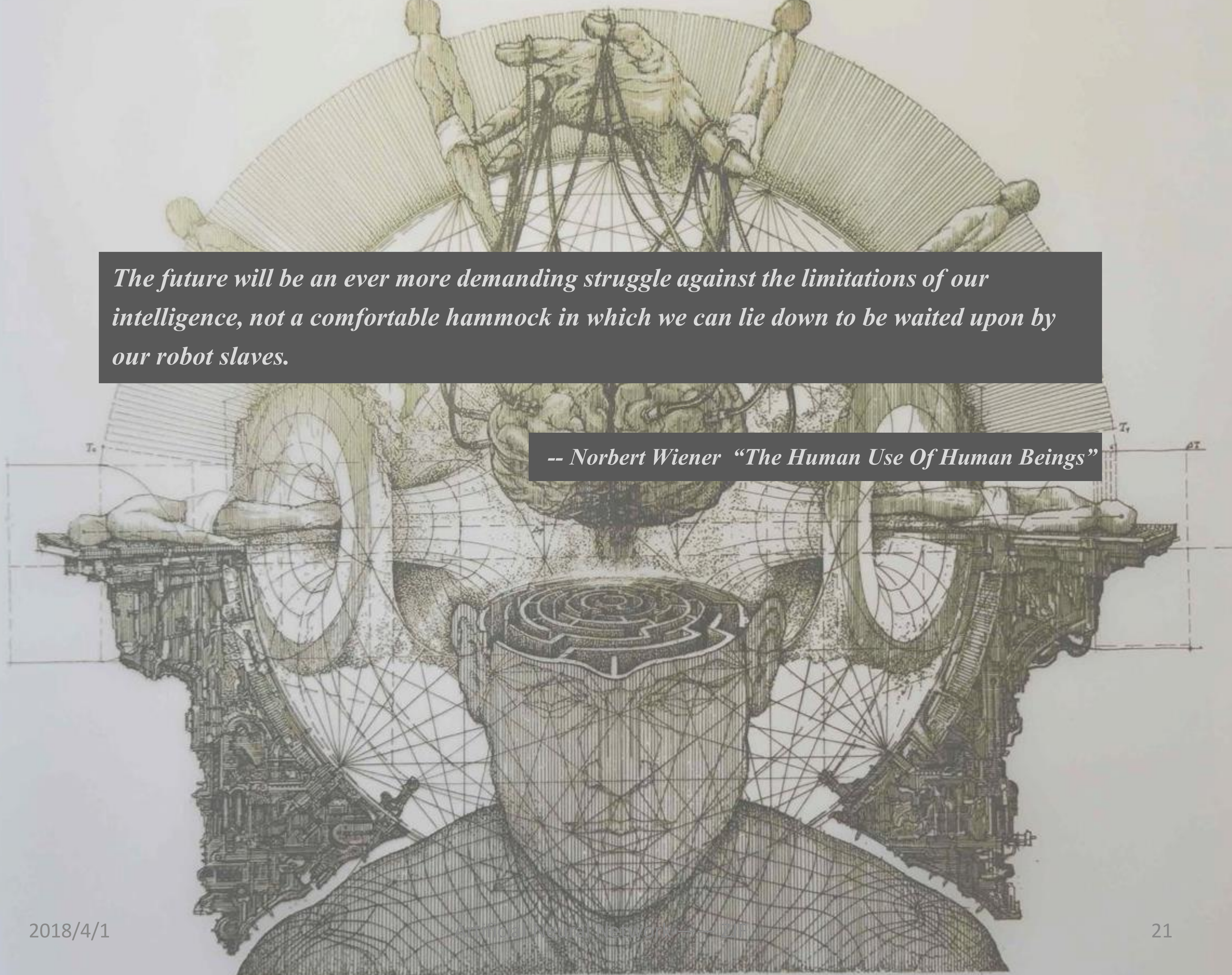
- 若干个卷积层 / 池化层后，将所有神经元展开成一维，后连若干全连接层，最后施加 $Softmax$；

- 相当于依靠训练过程自动发现合适的滤波器（*卷积核*），对原始图像以及后续的每一层的多个 feature map（*可看做是图像 channel*）做滤波，逐层提取特征。

7、深度学习

深度神经网络面临的问题	深度学习的解决办法
权值过多	减少连接数量、权值共享
梯度消失或梯度爆炸	适当的权值初始化、ReLu 激活函数、Batch Normalization
模型自由度高，容易过拟合	ℓ_1、ℓ_2 正则化、early stopping、数据增强、drop out
训练速度慢	梯度下降法的变体：AdaGrad\RMSProp\Adam 等

- ✓ 知乎专栏“计算主义” ： zhuanlan.zhihu.com/pillgrim
- ✓ python 机器学习库 Mentat ： github.com/zhangjuefei/mentat



The future will be an ever more demanding struggle against the limitations of our intelligence, not a comfortable hammock in which we can lie down to be waited upon by our robot slaves.

-- Norbert Wiener "The Human Use Of Human Beings"