lec13

机器学习中深度学习的基本概念，特别是训练神经网络，包括感知器的工作原理及训练算法、误差反向传播算法、梯度下降法及其在神经网络权重调整中的应用

人工神经网络的评估

基本思想：定义误差函数并测量未训练数据（测试集）的误差

- 通常形式：

$$E = \sqrt{\sum_i (d_i - o_i)^2}$$

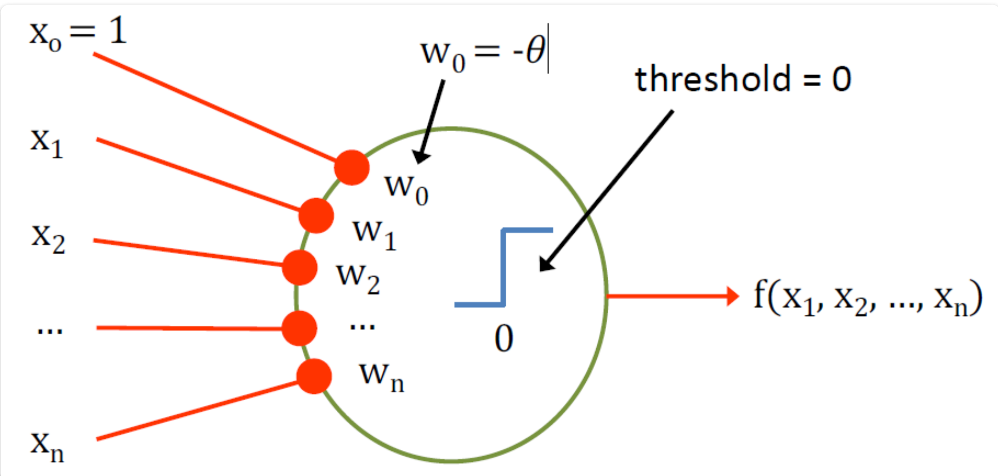
  其中 d 是期望输出，o 是实际输出
- 对于分类任务：

$$E = \frac{\text{正确分类样本数}}{\text{样本总数}}$$

感知器

大火应该很熟悉了

感知器（Perceptron）- 偏置项（Bias）



- 输入信号：$x_0 = 1, x_1, x_2, \dots, x_n$
- 权重：$w_0 = -\theta, w_1, w_2, \dots, w_n$
- 阈值：0
- 净输入信号：

$$\text{net} = \sum_{i=0}^{n} w_i x_i$$

- 输出信号：

$$f(\text{net}) = \begin{cases} +1 & \text{if net} > 0 \\ -1 & \text{otherwise} \end{cases}$$

- 注意：只有权重向量是可调整的，阈值不可调整

感知器计算（Perceptron Computation）

- 类似于阈值逻辑单元（TLU），感知器通过一个 $n-1$ 维超平面将其 n 维输入空间划分开：

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0$$

  - 当 $w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0$ 时，输出为 1
  - 当 $w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n \le 0$ 时，输出为 -1
- 通过正确的权重向量 $(w_0, \ldots, w_n)^\top$，单个感知器可以计算任何线性可分的函数

感知器训练（Perceptron Training）

- 假设输入 x 的类别为 $\text{class}(x) = -1$：
  - 如果 $w \cdot x > 0$，则发生误分类
  - 此时需要将权重向量修改为 $w + \Delta w$，使得 $(w + \Delta w) \cdot x < w \cdot x$ 以可能改进分类
- 我们可以选择 $\Delta w = -\eta x$，因为：

  $(w + \Delta w) \cdot x = (w - \eta x) \cdot x = w \cdot x - \eta x \cdot x$$并且$x \cdot x$是向量x长度的平方，因此是正值

- 对于 $\text{class}(x) = 1$ 的情况类似，但符号相反：
  - 我们引入 $\tilde{x}$ 以统一这两种情况
- 假设误分类的输入 x：
  - 令 $\tilde{x} = \text{class}(x) \cdot x$，则 $w^{k-1} \cdot \tilde{x} < 0$。
  - 此时总能选择 $\Delta w = \eta \tilde{x}$ 使得：

$$w^k = w^{k-1} + \eta \tilde{x}$$

    - 直觉：
      - 如果 $\text{class}(x) = 1$，则 $w^{k-1} \cdot \tilde{x} < 0$ 且 $\Delta w \cdot \tilde{x} > 0$
      - 如果 $\text{class}(x) = -1$，则 $w^{k-1} \cdot \tilde{x} < 0$ 且 $\Delta w \cdot \tilde{x} > 0$
      - 因此，我们总是向正确的方向移动 w

感知器训练算法（Perceptron Training Algorithm）

ALGORITHM Perceptron Training;
Start with a randomly chosen weight vector w^0;
Let $k = 1$;
WHILE there exist input vectors that are misclassified by w^{k-1}, **DO**
  Let x be a misclassified input vector;
  Let $\tilde{x} = \text{class}(x) \cdot x$, implying that $w^{k-1} \cdot \tilde{x} < 0$;
  Update the weight vector to $w^k = w^{k-1} + \eta \tilde{x}$;
  Increment k;
END-WHILE

学习率和终止条件（Learning Rate and Termination）

- 理想情况下：当所有样本都被正确分类时终止

- 如果在大量步骤中误分类的样本数量没有变化，问题可能出在学习率 η 的选择上：
  - 如果 η 太大，分类可能会来回波动，需很长时间才能稳定下来
  - 如果 η 太小，分类的变化会非常缓慢
- 如果调整 η 没有帮助，则样本可能不是线性可分的，应终止训练

在二维空间中可视化感知器（Visualizing Perceptrons in 2D）

- 问题：我们如何可视化由方程 $w_0 + w_1x_1 + w_2x_2 = 0$ 定义的直线？
- 一种可能性：确定直线与坐标轴的交点：
  - $x_1 = 0$

$$w_0 + w_2x_2 = 0 \Rightarrow x_2 = -\frac{w_0}{w_2}$$

  - $x_2 = 0$

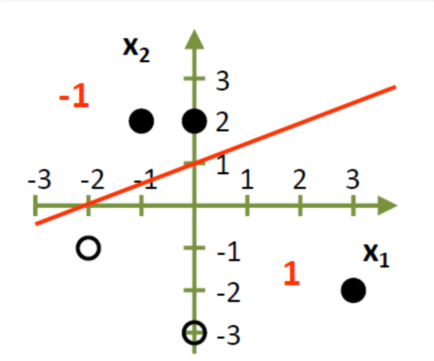
$$w_0 + w_1x_1 = 0 \Rightarrow x_1 = -\frac{w_0}{w_1}$$

- 因此，直线交于 $(0, -\frac{w_0}{w_2})$ 和 $(-\frac{w_0}{w_1}, 0)$
  - 如果 w_1 或 w_2 为 0，则直线分别为水平线或垂直线
  - 如果 w_0 为 0，直线穿过原点，其斜率为 $-\frac{w_1}{w_2}$

示例

2. **初始条件**：

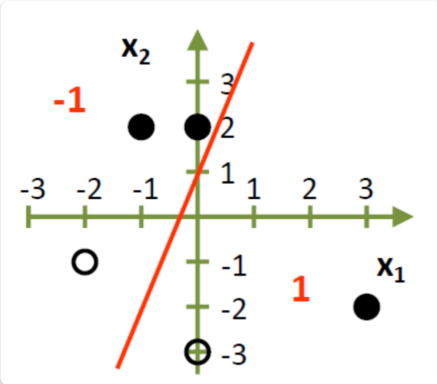
- 随机初始权重向量 $w^0 = (2, 1, -2)^T$。
- 分割线穿过点 $(0, 1)^T$ 和 $(-2, 0)^T$。



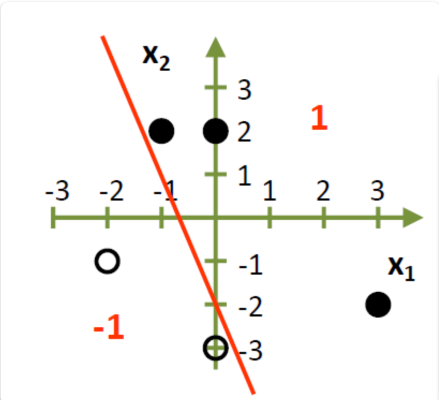
3. **步骤 1**：

- 选择误分类点 $(-2, -1)^T$ 进行学习：
  - $x = (1, -2, -1)^T$（包括偏置 $x_0 = 1$）
  - $\tilde{x} = (-1) \cdot (1, -2, -1)^T$（$x$ 属于类 -1）
  - $\tilde{x} = (-1, 2, 1)^T$

4. **步骤 2**:



- 更新权重：$w^1 = w^0 + \tilde{x}$（假设学习率 $\eta = 1$）：
  - $w^1 = (2,1,-2)^T + (-1,2,1)^T = (1,3,-1)^T$
  - 分割线穿过点 $(0,1)^T$ 和 $(-1/3,0)^T$。
- 选择下一个误分类点 $(0,2)^T$ 进行学习：
  - $x = (1,0,2)^T$（包括偏置 $x_0 = 1$）
  - $\tilde{x} = (1) \cdot (1,0,2)^T$（$x$ 属于类 1）
  - $\tilde{x} = (1,0,2)^T$



5. **步骤 3**:

- 更新权重：$w^2 = w^1 + \tilde{x}$
  - $w^2 = (1,3,-1)^T + (1,0,2)^T = (2,3,1)^T$
  - 现在分割线穿过点 $(0,-2)^T$ 和 $(-2/3,0)^T$。
- 所有点现在都已正确分类，学习过程终止。

其中 $x_0 = 1$ 是一种常见的设置手法，当然可以设置其他值

反向传播

- 该算法解决了“信任分配”问题，即跨层级给特定输出分配或归因于单个神经元
- 输出层的误差向后传播到较低层级的单元，这样所有神经元的权重都可以适当地调整

多类分类（Multiclass Discrimination）

- 每个感知器学习识别一个特定的类别：
  - 即，如果输入属于该类别，则输出 1，否则输出 0
  - 这些单元可以独立和并行训练
- 在生产模式下：
  - 当且仅当 $o_k = 1$ 且对于所有 $j \neq k$，$o_j = 0$ 时，网络判定其当前输入属于第 k 类，否则就是误分类
- 对于具有实值输出的单元：
  - 可以选择输出最大值的神经元来指示输入的类别
  - 这个最大值应明显大于所有其他输出，否则输入就是误分类

多层网络（Multilayer Networks）

- 单层感知器网络虽然可以区分任意数量的类别，但仍然需要输入的线性可分性（linear separability）。它不能解决 XOR 问题
- 为克服这一严重限制，我们可以使用多层神经元网络（multiple layers of neurons）
- 然而，它们的非可微输出函数导致了一种低效且薄弱的学习算法
- 最终导致突破性进展的想法是使用连续输出函数和梯度下降（gradient descent）

反向传播学习

目标:

- 反向传播算法的目标是修改网络的权重，使其输出向量 $o_p = (o_{p,1}, o_{p,2}, \ldots, o_{p,K})$ 尽可能接近期望输出向量 $d_p = (d_{p,1}, d_{p,2}, \ldots, d_{p,K})$，对于 K 个输出神经元和输入模式 $p = 1, \ldots, P$
- 输入输出对（示例）集合 $\{(x_p, d_p) : p = 1, \ldots, P\}$ 构成训练集

过程:

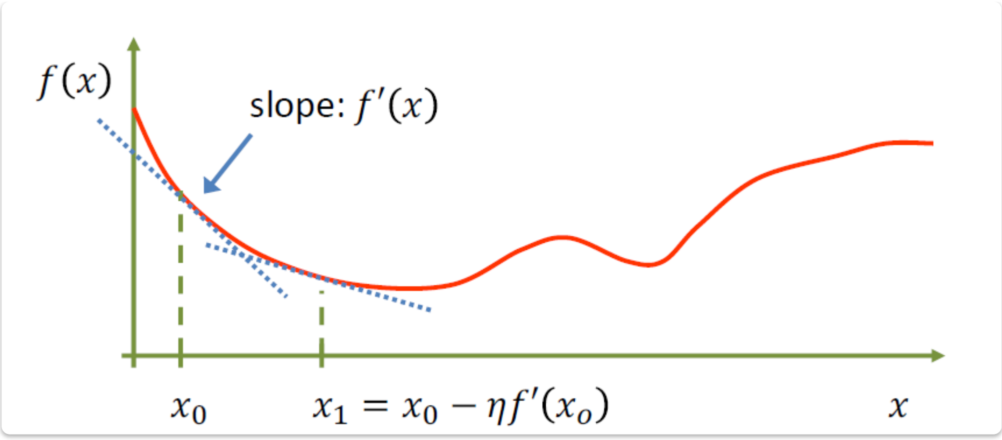
- 首先，我们需要一个累积误差函数来最小化:

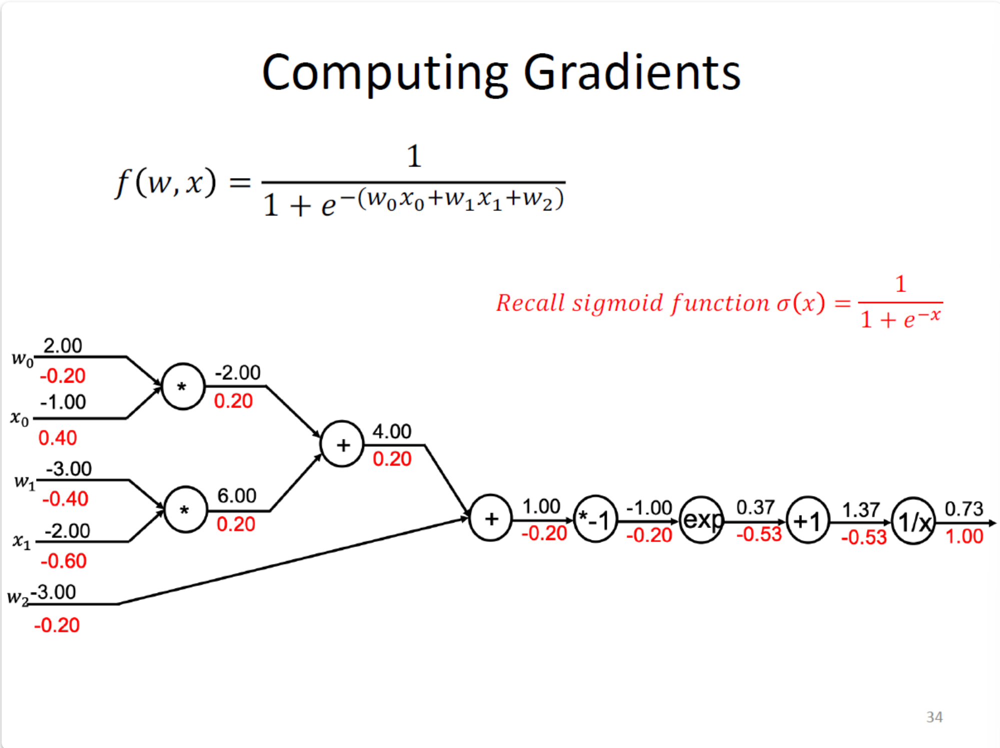
$$\text{Error} = \sum_{p=1}^{P} \text{Err}(o_p, d_p)$$

- 典型选择：均方误差（Mean Square Error，MSE）：

$$\text{MSE} = \frac{1}{P} \sqrt{\sum_{p=1}^{P} \sum_{j=1}^{K} (o_{p,j} - d_{p,j})^2}$$

- 如何最小化MSE：梯度下降（Gradient Descent）
  - 梯度下降是一种非常常用的技术，用于找到函数的绝对最小值
  - 它对于高维函数尤其有用
  - 它通过找到权重空间中误差表面的梯度，并在相反方向调整权重，迭代地最小化网络（或神经元）的误差





一个神经元的反向传播

$$f'^{(\text{net})} = f(\text{net})\big(1 - f(\text{net})\big)$$

$$f_i\big(\text{net}_i(t)\big) = \frac{1}{1 + e^{-(\text{net}_i(t)-\theta)/\tau}}$$