一、基本概念

- 感知器是**二分类**的**线性分类**模型，主要构造一个超平面将两种类别的数据分隔开来

> **定义 2.1（感知机）**假设输入空间（特征空间）是$X \subseteq R^n$，输出空间是$Y = \{+1, -1\}$，输入$x \in X$表示实例的特征向量，对应于输入空间（特征空间）的点；输出$y \in Y$表示实例的类别，由输入空间到输出空间的函数如下：
>
> $$y = sign(w \cdot x + b)$$
>
> 称为感知机（Perceptron），其中$sign(\cdot)$是符号函数，即：
>
> $$sign(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

- 线性可分数据集（Linearly separable dataset）：如果存在某个超平面能够将数据集的正实例点和负实例点**完全正确地划分到超平面的两侧**，则称数据集T为线性可分数据集
- Perceptron学习策略：极小化Loss Function

  $$\min_{w,b} Loss(w, b) = -\sum_{x_i \in M} y_i (w \cdot x_i + b)$$

  Loss Function「对应」**误分类点到分离超平面的总距离**，这里使用对应代表着Loss Function和总距离之间差一个比例项，但因为比例项并不影响Perceptron的优化，也就在Loss Function中忽略。

- 多解性：当训练数据集线性可分时，Perceptron学习算法存在**无穷多个解**，其解由于不同的参数初值/不同的迭代顺序而有所不同
  - 如果对多解性进行约束：则可能只有唯一的超平面满足要求，采用此类思想的算法有**线性支持向**

量机

- Gram矩阵：一组向量$\{x_1, x_2, \cdots, x_n\}$的所有可能的inner product，存储在矩阵中
- 两种Perceptron的学习策略
  1. 原始形式
     - 输入：训练数据集$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$，其中$x_i \in X = R^n$，$y_i \in Y = \{+1, -1\}, i = 1, 2, \cdots, N$，学习率$\eta(0 < \eta \leq 1)$
     - 输出：参数w, b
     - 算法过程：
       1. 选取初值w_0, b_0[1]
       2. 在训练集中随机选取单个数据(x_i, y_i)[2]
       3. 如果$y_i(w \cdot x_i + b) \leq 0$，代表选取到误分类点，更新参数[3]。

$$\begin{aligned} w &\leftarrow w + \eta y_i x_i \\ b &\leftarrow b + \eta y_i \end{aligned}$$

          否则，代表选取到正确分类点，不更新参数
       4. 转至（2.），直到训练集中没有误分类点
     - 算法的直观解释（超平面移动）：当一个实例点被误分类，即位于超平面的错误一侧时，调整参数w, b，让超平面向该误分类点的一侧移动，以减少该误分类点与超平面间的距离，直到超平面越过该误分类点使其被正确分类。
  2. 对偶形式
     - 输入：训练数据集$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$，其中$x_i \in X = R^n$，$y_i \in Y = \{+1, -1\}, i = 1, 2, \cdots, N$，学习率$\eta(0 < \eta \leq 1)$
     - 输出：参数$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_N)^T, b$，其中$\alpha_i = n_i \eta$，$n_i$代表点$(x_i, y_i)$被误分类的次数
       - 对偶形式的基本思路：将参数w, b转化为以x_i, y_i线性表达的形式，从而将求解w, b的问题转化为求解x_i, y_i系数的问题

$$\begin{aligned} w &\leftarrow w + \eta y_i x_i \\ b &\leftarrow b + \eta y_i \end{aligned}$$

这里假设$w_0 = 0, b_0 = 0$

在原始形式中我们每次更新参数都是选取一个误分类点进行更新，假设这个点是(x_i, y_i)，其被选取到（也就是被误分类的次数）为n_i次，则它的两个参数变化如下：（为了表示方便，这里暂时用N代替n_i，但仍表示(x_i, y_i)）被误分类的次数

$$
\begin{gathered}
w_{(x_i,y_i)} = w_N = w_{N-1} + \eta y_i x_i = w_{N-2} + 2\eta y_i x_i = \ldots = w_0 + N\eta y_i x_i = 0 + \alpha_i y_i x_i \\
b_{(x_i,y_i)} = b_N = b_{N-1} + \eta y_i = b_{N-2} + 2\eta y_i = \ldots = b_0 + N\eta y_i = 0 + \alpha_i y_i
\end{gathered}
$$

综上，对于w, b，我们可以替换为

$$
\begin{gathered}
w = \sum_{i=1}^{N} \alpha_i y_i x_i \\
b = \sum_{y=1}^{N} \alpha_i y_i
\end{gathered}
$$

这里的α_i的含义见注释[4]

- 算法过程：
  1. $\alpha \leftarrow 0, b \leftarrow 0$
  2. 在训练集中随选取单个数据(x_i, y_i)
  3. 如果$y_i(\sum_{j=1}^{N} \alpha_j y_j x_j + b) \leq 0$，代表选取到误分类点，更新参数

$$
\begin{gathered}
\alpha_i \leftarrow \alpha_i + \eta \\
b \leftarrow b + \eta y_j
\end{gathered}
$$

  4. 转至（2.），直到训练集中没有误分类点

- Gram Matrix - 对偶形式中，(x_i, y_i)仅以inner product的形式出现，为了方便，可以预先将所有向量之间的内积计算出来并存储在Matrix中

- Perceptron**原始形式**学习的收敛性：误分类的次数k是有上界的

> **定理2.1（Novikoff）** 设训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$是线性可分的，$y_i\in Y=\{+1,-1\}$，则
>
> 1. 存在满足条件$||w_{opt}||=1$的超平面$w_{opt}x+b_{opt}=0$将训练集完全正确分开，且存在$\gamma>0$ [5]
>
> $$y_i(w_{opt}\cdot x_i+b_{opt})\geq\gamma$$
>
> 2. 令$R=\max_{1\leq x\leq N}||\hat{x_i}||$，则感知机算法的原始形式在训练数据集上的误分类次数k满足不等式
>
> $$k\leq(\frac{R}{\gamma})^2$$

二、疑难解释

1. 为什么Perceptron会有两种形式的学习策略？对偶形式的优越性在哪里？
   - 对偶形式的目的是降低每次迭代的运算量，但是并非在任何情况下都能降低运算量，只有在特征空间的维度远大于数据集大小时才起作用。
   - 对偶形式的感知机，将原来的问题从频繁计算$w\cdot x$上，转换为频繁计算$x_i\cdot y_i$。前者在高维度的情况下计算较慢，后者我们可以使用Gram Matrix进行优化
   - 也就是说，对偶形式的Perceptron，把每轮迭代的时间复杂度的数据规模从**特征空间维度**n转移到了训练集数据大小N上，但是增加了预先计算Gram矩阵的时间，所以对于维度高，数量少的训练数据，可以提高每次迭代的性能。[6]

---

1. 不同的初值可能导致最后的超平面不同 ↩

2. 这里的随机梯度下降不是一次使所有误分类点的梯度下降，而是一次随机选择一个误分类点使其梯度下降 ↩

3. 这里的参数梯度更新策略由梯度计算得到 $\nabla_w Loss(w,b) = -\sum y_i x_i, \nabla_b Loss(w,b) = -\sum y_i$ ↩

4. $\alpha_i = n_i \eta$，即被误分类的次数和学习率的乘积。因为学习率是固定的，所以如果α_i越大，代表它被误分类的次数越多，也就是越难被正确分类。这样的数据点对学习结果影响也是最大的 ↩

5. 因为线性可分，则所有点都有$y_i(w_{opt} \cdot x_i + b_{opt}) > 0$，显然有一个最小值$\gamma = \min\{y_i(w_{opt} \cdot x_i + b_{opt})\}$ ↩

6. 如何理解感知机学习算法的对偶形式？ - 张伯翰的回答 - 知乎 https://www.zhihu.com/question/26526858/answer/253579695 ↩