中国科学技术大学
University of Science and Technology of China

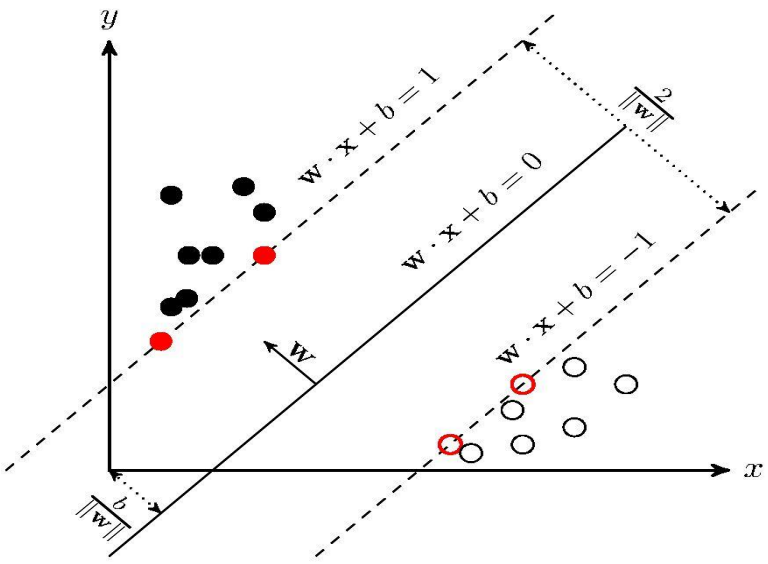
SVM实验

时间：11/22/2020

基本原理：

基本原理：支持向量机（support vector machines，SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM还包括核技巧，这使它成为实质上的非线性分类器。SVM的的学习算法就是求解凸二次规划的最优化算法。

算法原理：SVM学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，$w \cdot x + b = 0$即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。



SVM中W·X+b=0作为分离超平面，分割两类数据集合

基本原理：

线性支持向量机：求解线性支持向量机的过程是凸二次规划问题，所谓凸二次规划问题，就是目标函数是凸的二次可微函数。

近似线性支持向量机：当数据集并不是严格线性可分时，即满足部分样本点是线性可分，存在极少部分异常点；这里也就是说存在部分样本不能满足约束条件，此时我们可以引入松弛因子，这样这些样本点到超平面的函数距离加上松弛因子，就能保证被超平面分隔开来。

非线性支持向量机：显然，当数据集不是线性可分的，即我们不能通过前面的线性模型来对数据集进行分类。此时，我们必须想办法将这些样本特征符合线性模型，才能通过线性模型对这些样本进行分类。这就要用到核函数，核函数的功能就是将低维的特征空间映射到高维的特征空间，而在高维的特征空间中，这些样本进过转化后，变成了线性可分的情况。

基本原理：

几何间隔：对于给定的数据集T，分割超平面$w \cdot x + b = 0$，定义超平面关于样本(x_i, y_i)的几何间隔为

$$\gamma_i = y_i \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \cdot \boldsymbol{x_i} + \frac{b}{\|\boldsymbol{w}\|} \right)$$

函数间隔转化成几何间隔

超平面关于所有样本点的几何间隔的最小值为:

$$\gamma = \min_{i=1,2...,N} \gamma_i$$

实际上这个距离就是我们所谓的支持向量到超平面的距离，转化为以下约束最优化问题:

$$\max_{\boldsymbol{w},b} \ \gamma$$

$$s.t. \quad y_i \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \cdot \boldsymbol{x_i} + \frac{b}{\|\boldsymbol{w}\|} \right) \geq \gamma \, , i = 1, 2, \ldots, N$$

基本原理：

每个支持向量到分类超平面的距离：

$$d = \frac{|w^T x + b|}{||w||}$$

$$\max 2 * \frac{y(w^T x + b)}{||w||} \xrightarrow{\text{转化}} \min \frac{1}{2}||w||^2$$

求解目标函数：

$$\min \frac{1}{2}||w||^2 \ s.t. \quad y_i \left(w^T x_i + b \right) \geq 1 \qquad i=1,2,3\cdots m$$

优化方法：

梯度下降法：软间隔SVM的优化中使用hinge loss

输入：$D = \{(x_1, y_1), \cdots\cdots (x_i, y_i)\}$，超参数迭代次数M，惩罚因子C，学习率$\eta$

1.优化过程：

1) 初始化参数 $W = \{0,0 \cdots\cdots 0\}^{\mathrm{T}}$ ， $b = 0$

2) 遍历数据集 D:

1.计算误差向量 $e = \{C_1, \cdots, C_i\}^{\mathrm{T}}$

$$e = 1 - y_i(w \cdot x_i + b)$$

2.取出误差最大的一项：

$$i = arg \min_i e_i$$

3.若$e_i \leq 0$则直接退出循环体，否则对应的样本进行随机梯度下降

$$w \leftarrow (1-\eta)w + \eta C x_i y_i$$

$$b \leftarrow b + \eta C y_i$$

2.输出：hinge loss优化的SVM模型 $g(x) = sign\big(f(x)\big) = sign(w \cdot x + b)$

实验部分：

SMO算法：SMO是一种用于训练SVM的强大算法，它将大的优化问题分解为多个小的优化问题来进行求解。而这些小优化问题往往很容易求解，并且对它们进行顺序求解和对整体求解结果是一致的。在结果一致的情况下，显然SMO算法的求解时间要短很多，这样当数据集容量很大时，SMO就是一种十分高效的算法。

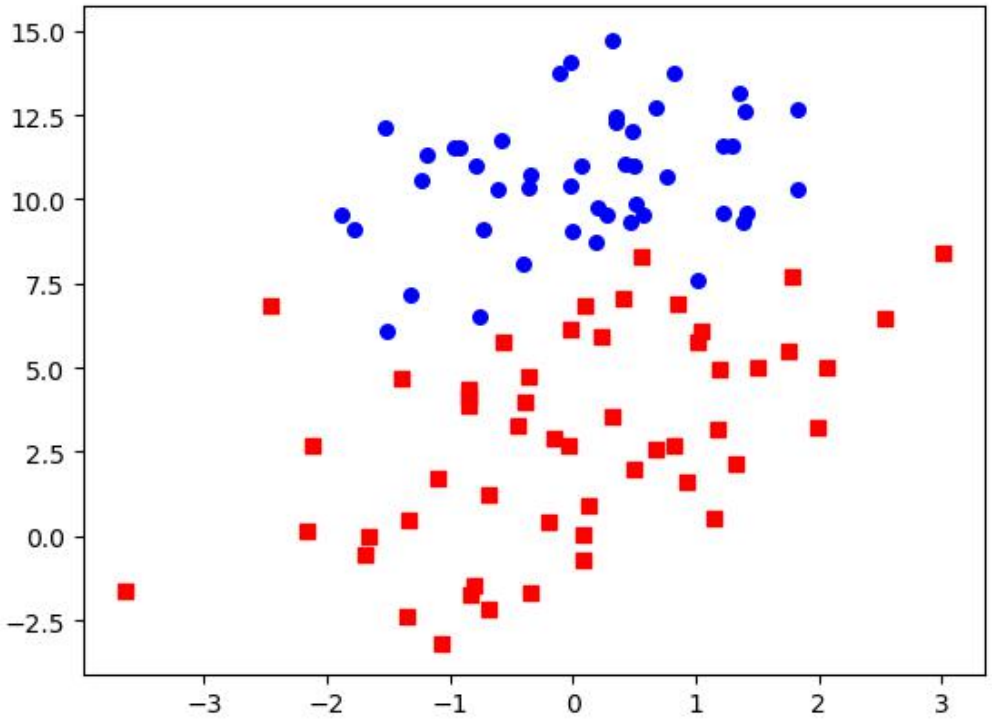
SVM的对偶问题：

$$arg\max_{\alpha}\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_iy_j\alpha_i\alpha_j\langle x_i,x_j\rangle$$

我们需要对上式中，对所有的($\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_n$)进行优化。SMO算法将原始问题转化成求N对参数的二次规划问题。我们在每一进行参数优化时，首先固定一对参数（α_i,α_j），逐步对模型进行优化。

数据集：

数据集1： Horse-colic数据集，由于SVM主要用于二分类问题。数据集进行了一定的调整，修改后的数据集大小为100，标签值为1，-1。



Horse-colic数据集，数据集来自2010年1月11日UCI机器学习数据库。如左图所示，我们只选择其中100个样本，每个样本中包含两个特征。

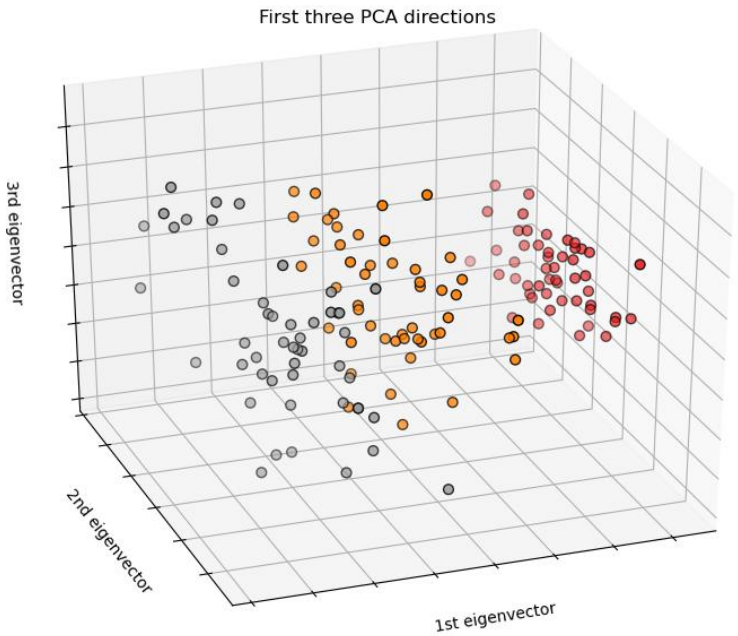
数据集：

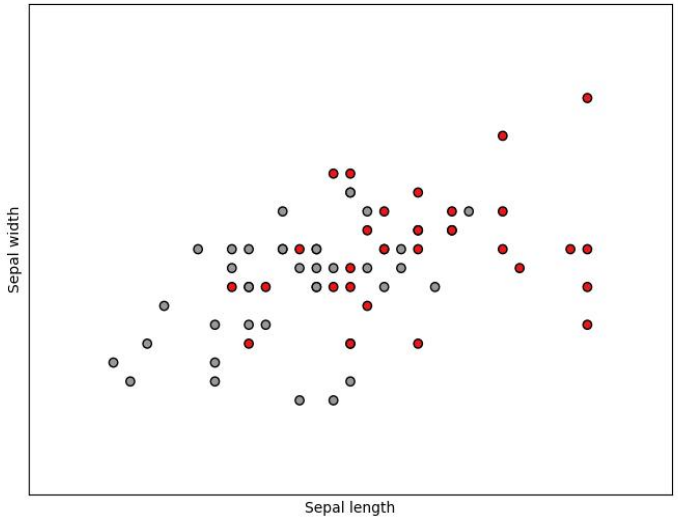
数据集2：采用sklearn中的iris数据集，由于SVM主要用于二分类问题。数据集进行了一定的调整，修改后的数据集大小为100，标签值为1，-1。

Train_data：数据集大小为70

Test_data：数据集大小为30

抽取前两个特征，数据转化成二维数据特征，便于可视化。





实验要求：

选做

1. 使用数据集1中数据，采用梯度下降法优化SVM模型.

2. 使用数据集2中数据，采用SMO算法优化SVM模型.

实验要求：

1. 训练SVM模型，数据存放在data文件夹中。使用梯度上升算法/SMO算法优化SVM，获得在训练集和测试集中的精度结果。

test_data.npy	2020/9/17 18:47	NPY 文件	2
test_target.npy	2020/9/17 18:47	NPY 文件	1
train_data.npy	2020/9/17 18:47	NPY 文件	2
train_target.npy	2020/9/17 18:47	NPY 文件	1

2. 数据load方法

```
39  x_train = np.load("./Data/svm/train_data.npy")
40  y_train = np.load("./Data/svm/train_target.npy")
41  x_test = np.load("./Data/svm/test_data.npy")
42  y_test = np.load("./Data/svm/test_target.npy")
```

3. Baseline

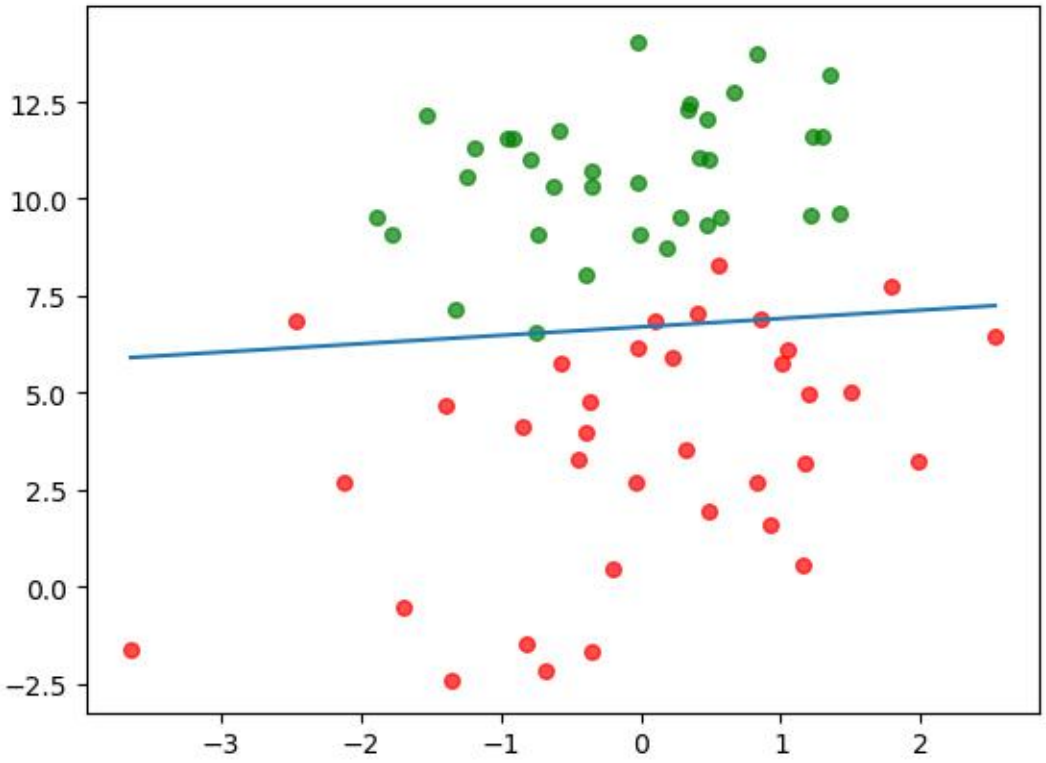
测评指标：精度值，正确预测占整体的比例

数据集1：
训练集精度：0.9
测试集精度：0.85

数据集2：
训练集精度：0.75
测试集精度：0.7

实验要求：

SVM模型分类示意图(数据集1)：



实验要求：

SVM模型分类示意图(数据集2)：