SVM 的实现以及核函数探究

522030910152 涂宇清

1. 简介

支持向量机（SVM）是一种常用的分类算法，其基本思想是找到一个最优的超平面，使得数据点到超平面的距离最大化，从而实现对数据的分类。

在本次大作业中，我们将不使用现有的机器学习库函数，从零开始实现 SVM 算法，探究不同核函数、多个核函数组合对 SVM 算法的影响，并在 MNIST 数据集和 CIFAR-10 数据集上进行测试。

2. 算法描述

2.1. 支持向量机

对于两类数据，如果其线性可分，那么存在一个超平面 $w^Tx+b=0$，使得对于任意数据点 x_i，若 $w^Tx_i+b>0$，则 x_i 属于第一类；若 $w^Tx_i+b<0$，则 x_i 属于第二类。而我们需要找到一个最优的超平面，即数据点中到超平面的最小距离最大。这些距离超平面最近的点就是支持向量。

由数据点到超平面的距离公式可知，对于任意数据点 x_i，其到超平面的距离 $d=\frac{1}{||w||}\cdot|w^Tx_i+b|$。若数据点在超平面上方，则 $d>0$；若数据点在超平面下方，则 $d<0$。但是这样在计算时会有一定的困难，因此，我们将在超平面上方数据的标签设为 1，超平面下方数据的标签设为-1，这样我们可以将数据点到超平面的距离公式改写为 $d=y_i(w^Tx_i+b)\cdot\frac{1}{||w||}$。当数据点被正确分类时，$y_i(w^Tx_i+b)\cdot\frac{1}{||w||}>0$，且数据点距离超平面越远，$d$ 越大。

2.2. 优化算法

在 SVM 算法中，我们需要找到一个最优的超平面，使得数据点到超平面的距离最大化。这是一个凸优化问题。

凸优化的目标函数为:

$$\arg\max_{w,b}\left\{\min_{n}(y_i(w^Tx_i+b))\cdot\frac{1}{||w||}\right\}$$

我们可以将目标函数改写为:

$$\begin{aligned}&\arg\max_{w,b}\frac{\hat{\gamma}}{||w||}\\ \text{s.t.}\quad & y_i(w^Tx_i+b)\geq\hat{\gamma},\quad i=1,2,\ldots,n\end{aligned}$$

其中 $\hat{\gamma}$ 是支持向量到超平面的距离。

当我们等比例改变 w 和 b 时，目标函数的值不变，因此我们可以将 w 和 b 同时除以 $\hat{\gamma}$，将目标函数改写为:

$$\begin{aligned}&\arg\max_{w,b}\frac{1}{||w||}\\ \text{s.t.}\quad & y_i(w^Tx_i+b)\geq 1,\quad i=1,2,\ldots,n\end{aligned}$$

同时，我们可以将求最大值问题转化为求最小值问题，将目标函数改写为：

$$\begin{aligned}&\arg\min_{w,b}\frac{1}{2}||w||^2\\ \text{s.t.}\quad & y_i(w^Tx_i+b)\geq 1,\quad i=1,2,\ldots,n\end{aligned}$$

我们可以使用拉格朗日乘子法求解上述问题，得到拉格朗日函数：

$$L(w,b,\alpha)=\frac{1}{2}||w||^2-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)-1)$$

其中 α_i 是拉格朗日乘子，$\alpha_i\geq 0$。

接下来我们通过获取目标函数的对偶问题来求解原问题。其对偶问题为：

$$\begin{aligned}&\arg\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_jy_iy_j\langle x_i^T,x_j\rangle\\ \text{s.t.}\quad & \alpha_i\geq 0,\quad i=1,2,\ldots,n\\ & \sum_{i=1}^{n}\alpha_iy_i=0\end{aligned}$$

而对于这个对偶问题，我们可以使用最小化算法（SMO）[?] 来求解。对于一系列需要优化的变量 α，我们每次选择两个变量 α_i, α_j 进行优化，固定其他变量，使得目标函数最小化。这样我们可以通过迭代来求解最优的 α。这样做可以将原本有 n 个变量的二次优化问题转化为 n 个只有两个变量的问题，从而降低了算法的时间复杂度。

我们先来看看 SMO 算法中的一些关键步骤：

计算误差

对于每个 α_i，我们需要计算其对应的误差 E_i，即

$$E_i = f(x_i) - y_i$$

其中 $f(x_i) = \sum_{j=1}^{n} \alpha_j y_j \langle x_j^T, x_i \rangle + b$。

计算 α 的上下界 H, L

如果 $y_i \neq y_j$，则

$$L = \max\{0, \alpha_j - \alpha_i\}$$
$$H = \min\{C, C + \alpha_j - \alpha_i\}$$

如果 $y_i = y_j$，则

$$L = \max\{0, \alpha_i + \alpha_j - C\}$$
$$H = \min\{C, \alpha_i + \alpha_j\}$$

其中 C 是一个常数，用来控制 α 的取值范围。

计算目标函数的二阶导数 η

$$\eta = 2\langle x_i, x_j \rangle - \langle x_i, x_i \rangle - \langle x_j, x_j \rangle$$

更新 α_j

$$\alpha_j = \alpha_j - \frac{y_j(E_i - E_j)}{\eta}$$

更新 α_i

$$\alpha_i = \alpha_i + y_i y_j (\alpha_j^{old} - \alpha_j)$$

更新 b

$$b_1 = b - E_i - y_i(\alpha_i - \alpha_i^{old})\langle x_i, x_i \rangle - y_j(\alpha_j - \alpha_j^{old})\langle x_i, x_j \rangle$$
$$b_2 = b - E_j - y_i(\alpha_i - \alpha_i^{old})\langle x_i, x_j \rangle - y_j(\alpha_j - \alpha_j^{old})\langle x_j, x_j \rangle$$

$$b = \begin{cases} b_1 & \text{if } 0 < \alpha_i < C, \\ b_2 & \text{if } 0 < \alpha_j < C, \\ \frac{b_1 + b_2}{2} & \text{otherwise.} \end{cases}$$

最终，SMO 算法的伪代码如下：

Algorithm 1 SMO 算法

$\forall i, \alpha_i = 0, b = 0$
$passes = 0$
while do $passes < max\_passes$
    $num\_changed\_alphas = 0$
    for $i = 1 \to n$ **do**
        计算误差 E_i
        if α_i 不满足 KKT 条件 **then**
            随机选择 $\alpha_j \neq \alpha_i$
            计算误差 E_j
            保存旧的 α_i 和 α_j
            计算 α_j 的上下界 H 和 L
            if $L == H$ **then**
                无法优化，继续下一个 α
                continue
            end if
            计算目标函数的二阶导数 η
            if $\eta \geq 0$ **then**
                无法优化，继续下一个 α
                continue
            end if
            更新 α_j
            if α_j 的变化太小，则这次优化无显著提升，放弃优化 **then**
                continue
            end if
            更新 α_i
            更新 b
            $num\_changed\_alphas \mathrel{+}= 1$
        end if
    end for
    if $num\_changed\_alphas == 0$ **then**
        $passes \mathrel{+}= 1$
    else
        $passes = 0$
    end if
end while

2.3. 软间隔

普遍情况下，数据往往含有噪声，这导致数据不是线性可分的。这时我们可以引入软间隔，允许数据点在超平面上方或下方一定的范围内。我们可以引入一个松弛变量 ξ_i，使得约束条件变为：

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

同时，我们需要限制松弛变量 ξ_i 的值，即在目标函数上加上一个惩罚项，使原本的目标函数变为：

$$\arg\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

其中 C 是一个常数，表示对误分类的惩罚程度。

所以，该优化问题就变为：

$$\begin{aligned} &\arg\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, 2, \ldots, n \end{aligned}$$

同样，我们可以求得其对偶问题：

$$\begin{aligned} &\arg\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle x_i^T, x_j\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n \\ & \sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned}$$

可以发现，添加软间隔后，对偶问题的形式不变，依然可以使用 SMO 算法来求解。

2.4. 启发式算法

启发式算法会在两个由 α 组成的集合之间交替：

- 整个训练集中的 α。
- 上次迭代中非零的 α。

通过这种方式，算法尝试先在更有可能找到违反 KKT 条件的支持向量上优化，从而减少计算量，并在必要时才遍历整个数据集来寻找其他可能的优化点。这种方法可以显著提高 SMO 算法的效率。

2.5. 核函数

在实际应用中，数据往往不是线性可分的，这时我们可以使用核函数将数据映射到高维空间，使得数据在高维空间中线性可分。

引入核函数，不仅能够将数据映射到高维空间，使得数据在高维空间中线性可分，还能降低算法的时间复杂度。这是因为核函数可以代替内积运算，在 SMO 算法中需要进行内积运算时，我们可以直接使用核函数来代替，从而降低了计算量。即 $K(x_i, x_j) = \langle\phi(x_i), \phi(x_2)\rangle$。

2.5.1. 常用核函数

线性核函数

$$K(x_i, x_j) = c \cdot x_i^T x_j$$

其中 c 是一个常数，用于控制核函数的权重。

多项式核函数

$$K(x_i, x_j) = c \cdot (x_i^T x_j + \theta)^d$$

其中 c 是一个常数，用于控制核函数的权重，θ 是一个常数，用于控制核函数的偏移，d 是一个常数，用于控制核函数的最高幂次。

高斯核函数

$$K(x_i, x_j) = c \cdot \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

其中 c 是一个常数，用于控制核函数的权重，σ 是一个常数，用于控制核函数的宽度。

Sigmoid 核函数

$$K(x_i, x_j) = c \cdot \tanh(\beta \cdot x_i^T x_j + \theta)$$

其中 c 是一个常数，用于控制核函数的权重，β 是一个常数，用于控制核函数的斜率，θ 是一个常数，用于控制核函数的偏移。

2.5.2. 多核函数组合

在实际应用中，我们可以将多个核函数组合起来，使得数据在更高维度的空间中更好地线性可分。多核函数计算公式如下：

$$K(x_i, x_j) = \sum_{k=1}^{n} w_k K_k(x_i, x_j)$$

3. 实验

3.1. 数据集

3.1.1. MNIST 数据集

MNIST 数据集是一个手写数字数据集，包含 60000 个训练样本和 10000 个测试样本。每个样本是 28×28 的灰度图像，标签为 0-9 的数字。

3.1.2. CIFAR-10 数据集

CIFAR-10 数据集是一个包含 50000 个训练样本和 10000 个测试样本的数据集。每个样本是 32×32 的 RGB 图像的数据集，共分为 10 类。

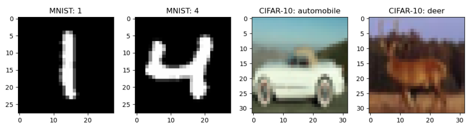

图 1: 数据集预览

在本次实验中，对于 MNIST 数据集和 CIFAR-10 数据集，我们分别取每类训练集前 500 张图片为训练集，每类测试集前 100 张图片为测试集。共计 5000 张训练图片与 1000 张测试图片。

3.1.3. 数据预处理

归一化 在实验中，我们对数据进行了归一化处理，将数据的像素值从 $[0, 255]$ 归一化到 $[0, 1]$。

通过归一化，我们可以使得数据的特征值在同一量级上，从而加快算法的收敛速度并提高算法的准确率。

MNIST 数据集特征提取 对于 MNIST 数据集，我们将 28×28 的图像展开成一个 784 维的向量，作为 SVM 的输入特征。

CIFAR-10 数据集特征提取 对于 CIFAR-10 数据集，我们通过`skimage`库的`hog()`函数提取图像的梯度直方图（HOG）特征，将图像的 32×32 的 RGB 图像转换为 324 维的 HOG 特征向量，作为 SVM 的输入特征。

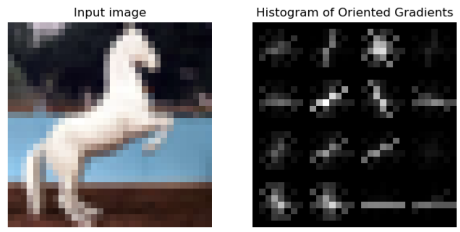

图 2: *HOG* 特征提取

3.1.4. 多分类策略

OVO 策略 OVO（One-Vs-One）策略是一种多分类策略，即将每两类数据进行一次分类，最终将所有分类结果进行投票，得到最终的分类结果。

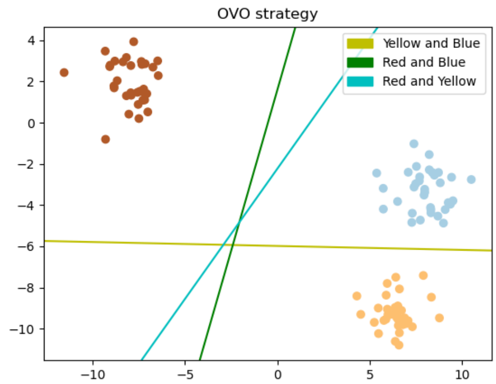

图 3: *OVO* 策略

OVA 策略 OVA（One-Vs-All）策略是一种多分类策略，即将每一类数据与其他所有类数据进行一次分类，最终将所有分类结果进行投票，得到最终的分类结果。

图 4: *OVA 策略*

策略对比 在实验中，我们将 OVO 策略和 OVA 策略进行对比，分析两种策略的优劣。通过实验，我们发现 OVO 策略相较于 OVA 策略，准确率更高且运行时间更短。

在准确率方面，在 OVO 策略中，每次训练 SVM 时，训练数据中两个标签的比例为 1:1，而在 OVA 策略中，训练数据中两个标签的比例为 1:9。这导致在 OVA 策略中出现了类别不平衡，SVM 更倾向于将数据分类为数量较多的类别，从而导致准确率下降。

而在运行时间方面，OVO 策略中，每次训练 SVM 时，训练数据量更小，而 SMO 算法的时间复杂度介于 $O(n^2)$ 和 $O(n^3)$ 之间。OVO 策略将数据集分割后训练，每次训练的数据量更小，因此运行时间更短。

3.2. 实验结果

3.2.1. 单核 SVM

参数消融 在实验中，我们对单核 SVM 算法中的核函数参数进行消融实验，以下是参数消融结果：

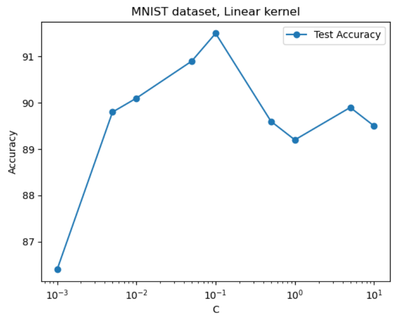

图 5: *MNIST + 线性核*

当 $C = 0.1$ 时，模型在测试集的准确率最高，为 91.1%。

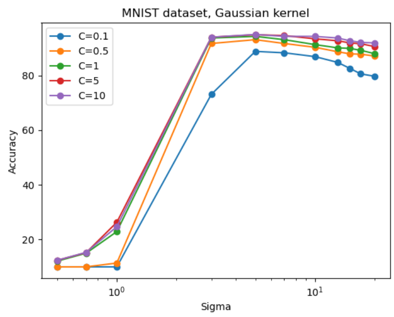

图 6: *MNIST + 高斯核*

当 $C = 10, \sigma = 5$ 时，模型在测试集的准确率最高，为 94.8%。

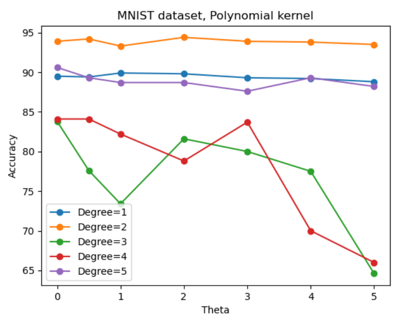

图 7: *MNIST + 多项式核*

当 $d = 2, \theta = 2$ 时，模型在测试集的准确率最高，为 93.7%。

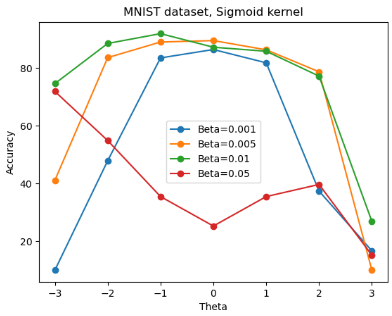

图 8: *MNIST +Sigmoid 核*

当 $\beta = 0.01, \theta = -1$ 时，模型在测试集的准确率最高，为 91.7%。

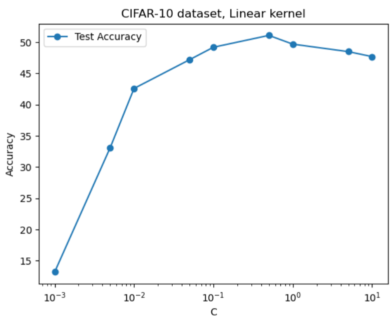

图 9: *CIFAR-10 + 线性核*

当 $C = 0.5$ 时，模型在测试集的准确率最高，为 50.6%。

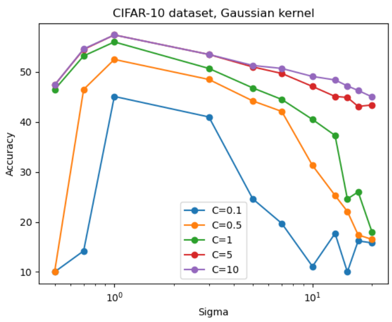

图 10: *CIFAR-10 + 高斯核*

当 $C = 10, \sigma = 1$ 时，模型在测试集的准确率最高，为 57.4%。

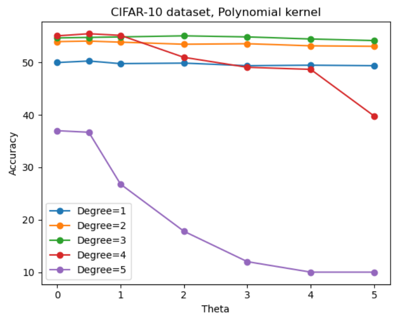

图 11: *CIFAR-10 + 多项式核*

当 $d = 4, \theta = 0.5$ 时，模型在测试集的准确率最高，为 55.4%。

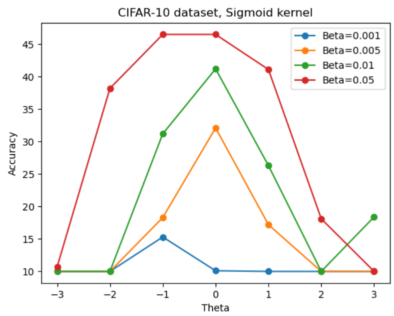

图 12: *CIFAR-10 +Sigmoid 核*

当 $\beta = 0.05, \theta = -1$ 时，模型在测试集的准确率最高，46.9%。

不同核函数在 MNIST 数据集和 CIFAR-10 数据集上的最优参数准确率和运行时间见下表：

表 1: *MNIST 数据集*

核函数	准确率	训练时间	测试时间
线性核	91.1%	16.97(s)	0.70(s)
高斯核	**94.8%**	179.36(s)	160.83(s)
多项式核	93.7%	17.15(s)	0.85(s)
Sigmoid 核	91.7%	16.74(s)	1.12(s)

表 2: *CIFAR-10* 数据集

核函数	准确率	训练时间	测试时间
线性核	50.6%	17.88(s)	0.59(s)
高斯核	**57.4%**	96.61(s)	62.81(s)
多项式核	55.4%	30.55(s)	1.31(s)
Sigmoid 核	46.9%	15.81(s)	0.95(s)

由上述实验结果可知，对于两个数据集，高斯核函数的效果最好，准确率分别为 94.8% 和 57.4%；线性核函数的效率最高，训练时间分别为 16.97s 和 17.88s。

3.2.2. 多核 SVM

参数消融 在实验中，我们采用单核 SVM 中各个核函数的最有参数，对多核 SVM 算法中的核函数权重进行消融实验，以下是参数消融结果：

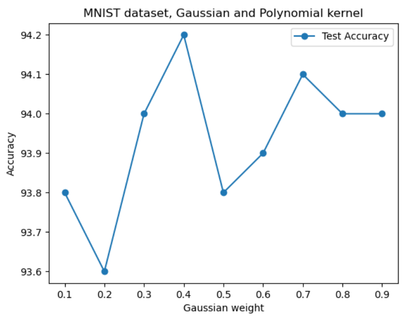

图 13: *MNIST* + 高斯核 + 多项式核

当 $Gweight = 0.4, Pweight = 0.6$ 时，模型在测试集的准确率最高，为 94.2%。

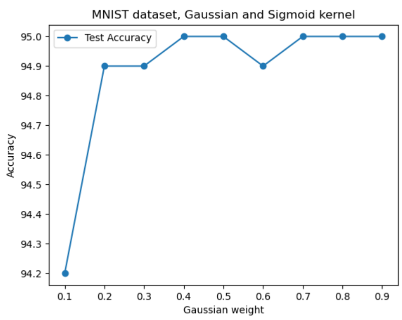

图 14: *MNIST* + 高斯核 + *Sigmoid* 核

当 $Gweight = 0.4/0.5/0.7/0.8/0.9$, $Sweight = 0.6/0.5/0.3/0.2/0.1$ 时，模型在测试集的准确率最高，为 95.0%。

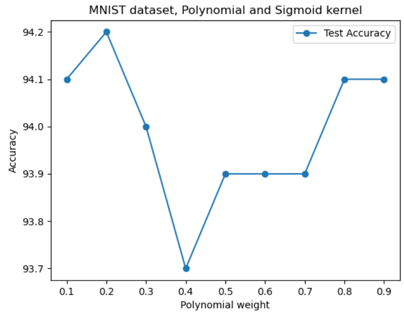

图 15: *MNIST* + 多项式核 + *Sigmoid* 核

当 $Pweight = 0.2, Sweight = 0.8$ 时，模型在测试集的准确率最高，为 94.2%。

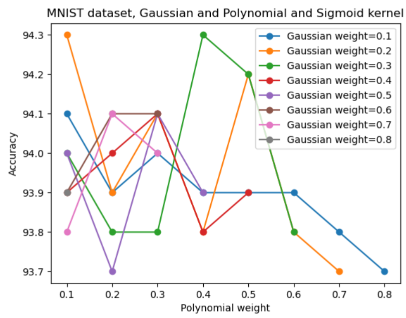

图 16: *MNIST*+ 高斯核 + 多项式核 +*Sigmoid* 核

当 $Gweight = 0.3$, $Pweight = 0.4$, $Sweight = 0.3$ 时，模型在测试集的准确率最高，为 94.3%。

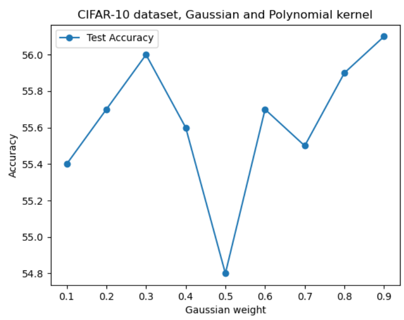

图 17: *CIFAR-10* + 高斯核 + 多项式核

当 $Gweight = 0.9, Pweight = 0.1$ 时，模型在测试集的准确率最高，为 56.1%。

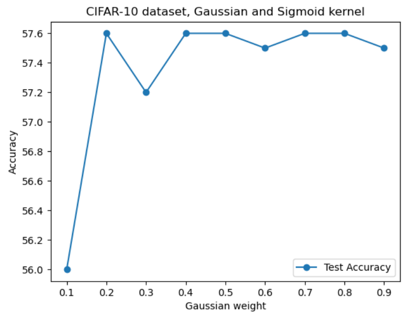

图 18: *CIFAR-10* + 高斯核 + *Sigmoid* 核

当 $Gweight = 0.2/0.4/0.5/0.7/0.8$, $Sweight = 0.8/0.6/0.5/0.3/0.2$ 时，模型在测试集的准确率最高，为 57.6%。

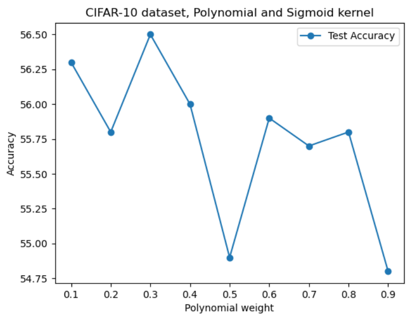

图 19: *CIFAR-10* + 多项式核 + *Sigmoid* 核

当 $Pweight = 0.3, Sweight = 0.7$ 时，模型在测试集的准确率最高，为 56.5%。

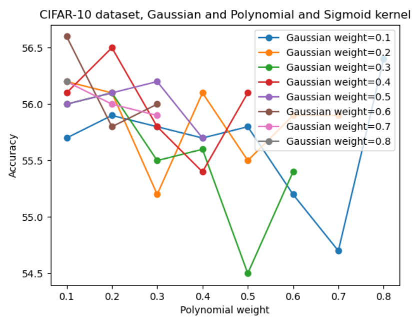

图 20: *CIFAR-10*+ 高斯核 + 多项式核 +*sigmoid* 核

当 $Gweight = 0.6$, $Pweight = 0.1$, $Sweight = 0.3$ 时，模型在测试集的准确率最高，为 56.6%。

由上述实验结果可知，除了在 CIFAR-10 上高斯核与 Sigmoid 核混合之外，当多核 SVM 的核函数为高斯核与其他核函数混合时，效果都没有单核 SVM 的高斯核效果好。而高斯核也是单核 SVM 中效果最好的核函数。由此，我们可以推断，当其他核函数与高斯核混合时，会影响高斯核的效果。

因此，我们尝试将多个参数不同的高斯核混合，核函数参数如下：

表 3: 高斯核混合

C	σ
9	4.5
10	5.0
11	5.5

对上述三个高斯核构成的多核函数进行消融实验，以下是参数消融结果：

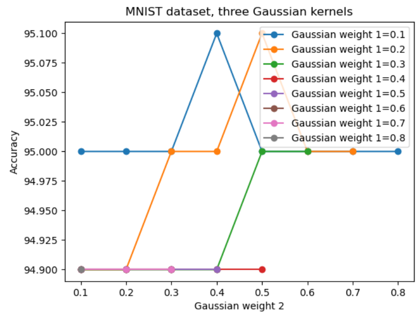

图 21: *MNIST +3* 个高斯核

当 $Gweight1 = 0.1/0.2$, $Gweight2 = 0.4/0.5$, $Gweight3 = 0.5/0.3$ 时，模型在测试集的准确率最高，为 95.1%。

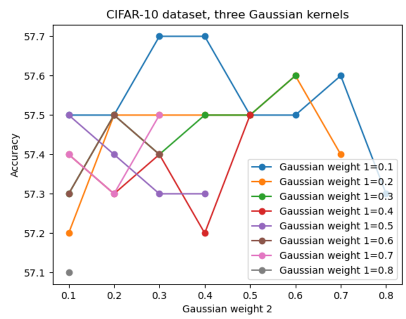

图 22: *CIFAR-10 +3* 个高斯核

当 $Gweight1 = 0.1$, $Gweight2 = 0.3/0.4$,

$Gweight3 = 0.6/0.5$ 时，模型在测试集的准确率最高，为 57.7%。

在 MNIST 数据集上，模型在测试集的准确率为 95.1%，在 CIFAR-10 数据集上，模型在测试集的准确率为 57.7%。均略高于单高斯核 SVM 的效果。

不同多核函数在 MNIST 数据集和 CIFAR-10 数据集上的最优参数准确率和运行时间见下表（G：高斯核、P：多项式核、S：Sigmoid 核）：

表 4: *MNIST* 数据集

核函数	准确率	训练时间	测试时间
G+P	94.2%	186.82(s)	173.95(s)
G+S	95.0%	188.97(s)	171.34(s)
P+S	94.2%	18.43(s)	1.83(s)
G+P+S	94.3%	187.91(s)	172.02(s)
G+G+G	**95.1%**	527.15(s)	518.82(s)

表 5: *CIFAR-10* 数据集

核函数	准确率	训练时间	测试时间
G+P	56.1%	94.42(s)	66.22(s)
G+S	57.6%	95.95(s)	65.83(s)
P+S	56.5%	33.46(s)	2.61(s)
G+P+S	56.6%	96.94(s)	67.30(s)
G+G+G	**57.7%**	236.12(s)	194.98(s)

4. 总结

在本次大作业中，我们实现了 SVM 算法，并在 MNIST 和 CIFAR-10 数据集上进行了实验。

在多分类策略上，我们发现 OVO 策略相较于 OVA 策略，准确率更高且运行时间更短。并合理分析了这种情况出现的原因。

通过参数消融实验，我们分析了 SVM 算法中的核函数参数对模型性能的影响。

我们发现，高斯核函数在两个数据集上的效果最好，准确率分别为 94.8% 和 57.4%；线性核函数的效率最高，训练时间分别为 16.97s 和 17.88s。

在多核 SVM 算法中，我们发现高斯核函数与其他核函数混合时，效果不如单核 SVM 中的高斯核效果好。在尝试将多个参数不同的高斯核混合后，发现效果略高于单高斯核 SVM 的效果。最终，我们在 MNIST 数据集上的准确率达到了 95.2%，在 CIFAR-10 数据集上的准确率达到了 57.7%。

通过这次大作业，我对 SVM 算法有了更深入的了解，对 SVM 算法的核函数、多分类策略等方面有了更深入的认识。同时，我也学会了如何使用 Python 从零实现 SVM 算法，并在实际数据集上进行实验。

5. 参考文献

[1] Stanford University, CS229 Course. *The Simplified SMO Algorithm.* 2009.