机器学习算法

一．PCA 降维
二．SVM 支持向量机
三．PCA+支持向量机的应用

一．PCA 降维

1.1 算法原理：

通过一定的数学运算，把数据的高维特征转换到低维特征，去掉冗余特征

1.2 算法步骤：（以 2 维举例）

❖ 对数据进行中心化处理，有 m 个样本，每个样本有 n 个特征，对每一个特征，通过这 m 个样本求其均值，得到 $\overline{X}$

$$\overline{X}=\sum_{i=1}^{m}x_i/m,\overline{X}\in R^{1\times n}$$

❖ 找一个方向，让这些样本在这个方向上分得最开，相当于对每一个样本，有 n 个特征，分别让这些特征在这个方向做投影，得到一个一维的特征（因为这里只找了一个方向），对 m 个样本，求新的特征的方差，使得方差最大，便得到了这个方向；

设 $\begin{matrix} j\in(1,n) \\ i\in(1,m) \end{matrix}$ $\quad X\_new_i^j=X_i^j-\overline{X}^j$

找这个方向的一个单位向量 $a\in n*1$

对于每个样本所对应的 $X\_new^j\in 1*n$

则得到的新的特征为每一个样本对应的特征在这个方向上的投影，单位向量做点积就是投影，故得到新的特征为 $X\_new_j\bullet a$

在求方差时，原本的方差定义为 $\sum_{i=1}^{m}(X_i^j\bullet a-\overline{X}^j\bullet a)^2/m$

转换便可得到方差为 $s^2=\sum_{i=1}^{m}(X\_new_i^j\bullet a)^2/m$，约束条件为 $\|a\|^2=1$

❖ 使用拉格朗日乘数法求极值

$$L=\sum_{i=1}^{m}(X\_new_i^j\bullet a)^T(X\_new_i^j\bullet a)/m+\lambda(1-a^T\bullet a)$$

$$=a^T(\sum_{i=1}^{m}X\_new_i^{jT}\bullet X\_new_i^j)/ma+\lambda(1-a^T\bullet a)$$

$$X\_new_i^{jT}=\begin{pmatrix} x^1 \\ \dots \\ x^n \end{pmatrix} \quad X\_new_i^j=\begin{pmatrix} x^1 & \dots & x^n \end{pmatrix}$$

$$conv(x,x)=\frac{\sum_{i=1}^{m} x_i^{\ 2}}{m}$$（因为数据中心化了）

$$\sum=(\sum_{i=1}^{m} X\_new_i^{j^T} \bullet X\_new_i^j)/m=\begin{pmatrix} \sum_{i=1}^{m} x^1x^1/m & \sum_{i=1}^{m} x^1x^2/m & \dots & \sum_{i=1}^{m} x^1x^n/m \\ \sum_{i=1}^{m} x^2x^1/m & \sum_{i=1}^{m} x^2x^2/m & \dots & \sum_{i=1}^{m} x^2x^n/m \\ \dots & \dots & \dots & \dots \\ \sum_{i=1}^{m} x^nx^1/m & \sum_{i=1}^{m} x^nx^2/m & \dots & \sum_{i=1}^{m} x^nx^n/m \end{pmatrix}$$

$\sum$ 称为协方差矩阵

$L=a^T\sum a+\lambda(1-a^T\bullet a)$

对矩阵求偏导可得

$\frac{\partial L}{\partial a}=2\sum a-2\lambda a=0$

故$\sum a=\lambda a$

由矩阵特征值和特征向量的定义可知：

λ为协方差矩阵的特征值

a为特征之对应的特征向量，由于协方差矩阵为实对称阵，且为单位列向量，所以a可以作为 n 维空间的一个基。

通过求解λ和a，得到的便是数据投影后方差最大的方向，数据也分得最开，也就是找到了这些数据的主成分，也叫主特征；

进一步推广可得，选取前 k 个最大的特征值，得到 k 个对应的特征向量，作为 n 维空间里的一组单位正交基，每一个样本的 n 维特征经过投影后，变成 k 维，即(m*n)*(n*k)=m*k 所以实现了降维操作，去除了冗余特征，但是通过这种方法得到的特征不再具有原来的物理意义。

1.3 奇异值分解

对于非实对称矩阵$A\in R^{m\times n}$，存在矩阵$U\in R^{m\times m}$、$V\in R^{n\times n}$，使得

$A=U\sum V^T$

U方阵称为左奇异矩阵，为单位正交矩阵

$\sum$ 是对角矩阵，对角线上的元素称为奇异值，奇异值的个数为 m、n 的最小值

V方阵称为右奇异矩阵，也为单位正交矩阵

对于 PCA 算法而言，由于协方差矩阵为实对称阵，故奇异值就是特征值，左奇异和右

奇异矩阵都是 n 维方阵，若选取了 k 个特征值，则 $U \in R^{n\times k}$，新的样本矩阵为

$$X\_new^{m\times k} = X^{m\times n} \bullet U^{n\times k}\text{，数据重建则是 } X^{m\times n} = X\_new^{m\times k} \bullet U^{T(k\times n)}$$

对于 SVD 算法而言，与 PCA 算法最大的不同就是不需要求协方差矩阵，而是直接对样本矩阵进行奇异值分解，通过计算发现 SVD 与 PCA 算法没有明显的差异

计算过程如下：

```
import numpy as np
a = np.array([[1,5,7],[2,3,6]])
C = np.dot(a.T,a)
print(C)
u,s,v = np.linalg.svd(C,full_matrices=False,compute_uv=True)
u1,s1,v1 = np.linalg.svd(a,full_matrices=False,compute_uv = True)
print(s)
print(u)
print(v.T)
print(s1)
print(u)
print(v.T)
```

结果为：

```
C=[[ 5 11 19]
  [11 34 53]
  [19 53 85]]
s = [1.22415230e+02 1.58477013e+00 3.87057470e-15]
u = [[-0.1836866    0.74076546 -0.64616234]
    [-0.52210838 -0.63048071 -0.57436653]
    [-0.83286378  0.23186334  0.50257071]]
v.T = [[-0.1836866    0.74076546 -0.64616234]
      [-0.52210838 -0.63048071 -0.57436653]
      [-0.83286378  0.23186334  0.50257071]]
s1 = [11.06414162  1.25887654]
u1 = [[-0.7794798  -0.62642736]
     [-0.62642736  0.7794798 ]]
v1.T = [[-0.1836866    0.74076546]
       [-0.52210838 -0.63048071]
       [-0.83286378  0.23186334]]
```

可以发现奇异值的数目取决于 m,n 中的最小值，对应基底的在 v1.T 中，它的个数与奇异值的个数相等。

1.4 PCA 算法与 LDA 算法的区别：

LDA 算法是一种线性判别分析算法，是有监督的降维算法，它也是在寻找一个投影矩阵，使其投影之后的数据样本同类的接近而不同类的远离；

二．SVM 支持向量机

2.1 相关概念

二分类模型
基本模型是定义在特征空间上的最大间隔线性分类器
学习策略是间隔最大化
最终转化为一个求解凸二次规划的问题
学习算法是求解凸二次规划的最优化算法

以二维平面为例，SVM 算法就是要在两类不同的数据之间找到一条直线，把他们分开，若在高维空间中，就是找到一个超平面把两类数据分开，类比直线，定义超平面为

$$W^T X + b = 0$$

对于每一条能够分开两类样本的直线或者超平面，平行地将其向正负样本点移动，直到与样本点相交为止，得到的两条直线的距离为 d,为了保证其唯一性，让最优的直线取在 d 的中间，优化的目的就是为了寻找到最大的间隔 d,找到其对应的参数 W 和 b；
而支持向量就是指与移动的平行线相交的向量点；必然有正样本的支持向量也有负样本的支持向量；
可以看出，最优化直线（超平面）的过程只与支持向量有关，不需要其他向量，所以处理的数据比较少，适合于小样本的训练。

2.2 数学模型

定义超平面的模型

$$W^T X + b = 0$$

对于已知的样本集
遵循以下两个公式：

$$\begin{cases} W^T X_i + b < 0, Y_i = -1 \\ W^T X_i + b > 0, Y_i = +1 \end{cases}$$

因此用一种更简单的形式表示为：

$$Y_i(W^T X_i + b) \geq 0$$

从这种表示形式中可以发现：

$\left| W^T X_i + b \right|$ 能够表示 X_i 距离超平面的远近

而 $W^T X_i + b$ 的符号与其标记 Y_i 的符号是否一致又能够表示分类是否正确，对于样本集我们在找直线（超平面）的时候，一定是分类正确的，所以 $Y_i(W^T X_i + b)$ 越大，分类的确信度越大，但是我们不需要对每一个样本点都求其到超平面的距离，只需要使支持向量到超平面的距离 d/2 最大即可；
对此给出函数间隔的定义：

定义超平面 $W^T X + b = 0$ 关于样本点 (X_i, Y_i) 的函数间隔 $\hat{r}_i$ 为

$\hat{r}_i = Y_i(W^T X_i + b)$
$\hat{r} = \min \hat{r}_i (i = 1,.....m)$

但是如果用函数距离来表示 d,则存在问题：只要成比例地改变W 和b ，超平面并没有改变，但是函数距离却成倍数变换，所以是不准确的。
所以根据点到平面的距离公式：

设平面的方程为：$W_1 X + W_2 Y + b = 0$

则点(X_0, Y_0)到平面的距离为$d = \dfrac{|W_1 X_0 + W_2 Y_0 + b|}{\sqrt{W_1^2 + W_2^2}}$

超平面：$W^T X + b = 0$

定义几何间隔为$d_i = \dfrac{(W^T X_i + b) Y_i}{\|W\|} = \dfrac{\hat{r}_i}{\|W\|}$

$$d = \min d^i (i = 1,...m) = \frac{\hat{r}}{\|W\|}$$

因为$(W^T X_i + b) Y_i = |W^T X_i + b|$

所以几何间隔的定义没有稀奇之处，无非就换了一种形式；

但是如果直接求此时 d 的最大值，显然不太方便，并且我们知道$W^T X + b = 0$与$aW^T X + ab = 0$是一个超平面，故可以通过调整 a 的值使得支持向量到超平面的函数距离$\hat{r}$正好等于 1，因为最短的函数距离为 1，其他样本点的函数距离均会大于 1；取 1 只是为了问题更加简单，当然也可以取其他常数，最终求解出来的参数与原来只是相差了一个比例系数，而不会影响超平面的选取。
此时便得到支持向量机要求解的原始问题：

$$\begin{cases} \min \|W\| \\ s.t. Y_i(W^T X_i + b) \geq 1 \end{cases}$$

为了后续求解的方便，做一次变形，得到最终的优化问题：

$$\begin{cases} \min \frac{1}{2} \|W\|^2 \\ s.t. Y_i(W^T X_i + b) \geq 1 \end{cases}$$

2.3 求解上述问题：
引入拉格朗日乘子法，把约束优化问题变成无约束优化问题

$$L(W, b, \alpha) = \frac{1}{2}\|W\|^2 + \sum_{i=1}^{m} \alpha_i (1 - Y_i(W^T X_i + b)), \quad \alpha_i \geq 0$$

当$(1 - Y_i(W^T X_i + b)) = 0$时

参数的最小值在$\frac{1}{2}\|W\|^2$与$\sum_{i=1}^{m}\alpha_i(1-Y_i(W^TX_i+b))$相切的地方取到，相当于等式约束。

当$(1-Y_i(W^TX_i+b))\prec 0$时，通过使α_i等于 0，保证$L(W,b,a)$的最小值就在$\frac{1}{2}\|W\|^2$处取到，不需要再相切；

所以要满足的条件是$\alpha_i(1-Y_i(W^TX_i+b))=0$，这个条件又称为互补松弛条件

为了保证在$L(W,b,a)$处取得的最小值就是在$\frac{1}{2}\|W\|$处的最小值，所以要求

$\sum_{i=1}^{n}\alpha_i(1-Y_i(W^TX_i+b))=0$，也刚好满足了互补松弛条件。

进一步的可知由于$\alpha_i\geq 0$且$(1-Y_i(W^TX_i+b))\leq 0$，所以

$$[\max\sum_{i=1}^{m}\alpha_i(1-Y_i(W^TX_i+b))]=0$$

因此得到一个新的优化问题：

$$\underset{w,b}{Min}\,L(W,b,\alpha)=\min_{w,b}\frac{1}{2}\|W\|^2+\max_{\alpha}\sum_{i=1}^{n}\alpha_i(1-Y_i(W^TX_i+b))=\min_{w,b}\max_{\alpha}(\frac{1}{2}\|W\|^2+\sum_{i=1}^{n}\alpha_i(1-Y_i(W^TX_i+b)))$$

$$\alpha_i\geq 0$$

根据原问题与对偶问题的关系，得到其对偶问题为

$$\max_{\alpha}\min_{w,b}L(W,b,\alpha)$$

假设原问题的最优值为p^*，对偶问题的最优值为d^*，原问题与对偶问题满足弱对偶条件（$p^*\geq d^*$），对偶间隙为（p^*-d^*）

若满足：

- 原问题是凸问题（目标函数是凸函数，约束构成凸集）
- 约束为线性约束（$1-Y_i(W^TX_i+b)\leq 0$）
- 满足 KKT 条件：a. $\frac{\partial L}{\partial W^*}=0,\frac{\partial L}{\partial b^*}=0(W^*,b^*$为最优解）

  b. 满足互补松弛条件（$\alpha_i(1-Y_i(W^TX_i+b))=0$）

  c. 原问题的可行性（$f_i(W,b)\leq 0,h_i(W,b)=0$）

  d. 对偶问题的可行性（$\alpha_i\geq 0$）

则对偶间隙等于 0，原问题的最优值等于对偶问题的最优值，即强对偶的充要条件；

由上述强对偶的充要条件可知，支持向量机算法里的原问题与对偶问题满足强对偶条件
在 KKT 条件下解对偶问题

$$L(w,b,\alpha)=\frac{1}{2}\|W\|^2+\sum_{i=1}^{m}\alpha_i(1-Y_i(W^TX_i+b)),\alpha_i\geq 0$$

由 KKT 的第一个条件得（固定 α 求解 W和b 的最小值，然后代入求解对偶问题的最大值）：

$$\frac{\partial L}{\partial W}=W-\sum_{i=1}^{m}\alpha_iY_iX_i=0$$

$$\frac{\partial L}{\partial b}=-\sum_{i=1}^{m}\alpha_iY_i=0$$

代入拉格朗日函数：

$$L(W,b,\alpha)=\frac{1}{2}\|W\|^2+\sum_{i=1}^{m}\alpha_i(1-Y_i(W^TX_i+b))$$
$$=\frac{1}{2}W^TW+\sum_{i=1}^{m}\alpha_i-\sum_{i=1}^{m}\alpha_iY_iW^TX_i$$
$$=\frac{1}{2}W^T\sum_{i=1}^{m}\alpha_iY_iX_i-\sum_{i=1}^{m}\alpha_iY_iW^TX_i+\sum_{i=1}^{m}\alpha_i$$
$$=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\alpha_iY_iW^TX_i$$
$$=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jY_iY_jX_i^TX_j$$

消去了 W 和 b，只剩下 α_i，原问题求解最小值变成求解对偶问题的最大值

$$\max[\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_jY_iY_jX_i^TX_j]，\sum_{i=1}^{m}\alpha_iY_i=0$$

求解 α，采用 SMO 算法，每次留出两个参数，固定其他参数，循环求最使得函数值最大的 α 的值。

2.4 非线性 SVM 和核函数
问题：数据在低维不可线性分割
解决方法：数据从低维向高维做映射，在高维空间里线性可分（用异或问题举例，在二维平面不可线性分割，但在三维空间里便可以线性分割了）
理论上已经证明，把一个数据映射到无限维，它一定是线性可分的。
非线性 SVM 的核心是把低维数据映射到高维数据，找到一个超平面，变成线性分类问题
数学描述：

把 X_i 变成它的高维映射 $\phi(X_i)$

此时求解的问题变成了

$$\max[\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_iY_i\alpha_jY_j\phi(X_i)^T\phi(X_j)]$$

如果直接计算 $\phi(X_i)$，然后再计算 $\phi(X_i)^T\phi(X_j)$，计算量会非常大，所以 SVM 在解决非线性问题的时候，引入核函数 $K(X_i,X_j)=\phi(X_i)^T\phi(X_j)$，把高维的数据运算又降到了低维；

非线性 SVM 最终的求解问题变为：

$$\max[\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_iY_i\alpha_jY_jK(X_i,X_j)]$$

$$\sum_{i=1}^{m}\alpha_iY_i=0$$

下面举一个简单的例子来说明

设 $d_i=\begin{pmatrix}x_1^i\\x_2^i\end{pmatrix}\xrightarrow{\phi(x)}\begin{pmatrix}(x_1^i)^2\\\sqrt{2}x_1^ix_2^i\\(x_2^i)^2\end{pmatrix}$， $\phi(d_1)=\begin{pmatrix}(x_1^1)^2\\\sqrt{2}x_1^1x_2^1\\(x_2^1)^2\end{pmatrix}$， $\phi(d_2)=\begin{pmatrix}(x_1^2)^2\\\sqrt{2}x_1^2x_2^2\\(x_2^2)^2\end{pmatrix}$， $d_1=\begin{pmatrix}x_1^1\\x_2^1\end{pmatrix}$，

$d_2=\begin{pmatrix}x_1^2\\x_2^2\end{pmatrix}$

$$\phi(d_1)^T\bullet\phi(d_2)=(x_1^1)^2(x_1^2)^2+2x_1^1x_1^2x_2^1x_2^2+(x_2^1)^2(x_2^2)^2$$

$$K(d_1,d_2)=(d_1^Td_2)^2=(x_1^1x_1^2+x_2^1x_2^2)^2$$

经过展开，可得 $K(d_1,d_2)=\phi(d_1)^T\bullet\phi(d_2)$

这便是通过一个核函数将高维运算又变成了低维运算

所以，总结非线性 SVM 算法的精髓：

1. 将低维线性不可分的数据进行高维特征映射，在高维特征下，即可寻找到一个线性可分的超平面，转换为线性分类。

2. 由于高维数据计算复杂，所以并没有去求特征映射，而是通过求核函数又把高维运算变到低维运算了。

核函数介绍

❖ 多项式核函数：$(X_i^TX_j)^d(d\geq1)$，d=1 为线性分类器，它的维度是 d,属于有限维。

❖ 高斯核函数：$e^{-\frac{\|X_i-X_j\|^2}{2\sigma^2}}$，它的维度是无穷维

证明（利用泰勒公式）

$$e^{-\frac{\|X_i-X_j\|^2}{2\sigma^2}}=e^{-\frac{(X_i{}^2-2X_iX_j+X_j^2)}{2\sigma^2}}=e^{-\frac{X_i^2}{2\sigma^2}}\bullet e^{-\frac{X_j^2}{2\sigma^2}}\bullet e^{-\frac{X_iX_j}{\sigma^2}}$$

$$=K(1+\frac{X_i}{\sigma}\bullet\frac{X_j}{\sigma}+\frac{(\frac{X_i}{\sigma})^2}{\sqrt{2!}}\bullet\frac{(\frac{X_j}{\sigma})^2}{\sqrt{2!}}+...+\frac{(\frac{X_i}{\sigma})^n}{\sqrt{n!}}\bullet\frac{(\frac{X_j}{\sigma})^n}{\sqrt{n!}})$$

行向量为：

$$(1,\frac{X_i}{\sigma},\frac{(\frac{X_j}{\sigma})^2}{\sqrt{2!}},.....\frac{(\frac{X_j}{\sigma})^n}{\sqrt{n!}})$$

列向量是其对应的转置，**所以高斯核函数将特征映射到无穷维。**

σ 称为高斯核函数的带宽，影响分类的效果。

证明：假设两个样本的映射分别为$\phi(X_1)$、$\phi(X_2)$

在高维空间里，两个样本点之间距离的平方为：

$$d_{1-2}{}^2=\|\phi(X_1)-\phi(X_2)\|^2=\phi(X_1)^2-2\phi(X_1)\phi(X_2)+\phi(X_2)^2$$
$$=\phi(X_1)\phi(X_1)+\phi(X_2)\phi(X_2)-2\phi(X_1)\phi(X_2)$$
$$=2-2e^{-\frac{\|X_1-X_2\|^2}{2\sigma^2}}$$

当$\sigma\to\infty$时，$d_{1-2}=0$，区分度下降，样本分不开，模型容易欠拟合

当$\sigma\to 0$时，$d_{1-2}=2$，区分度上升，分得太开，模型容易过拟合

Sklearn 中的 gamma 参数是方差σ 的倒数；

2.5 支持向量软间隔

硬间隔下的支持向量机算法最原始的问题是：

$$\begin{cases}\min\frac{1}{2}\|W\|^2\\ s.t.Y_i(W^TX_i+b)\ge 1\end{cases}$$

在支持向量处，约束条件取到 1，对于其他样本点，约束条件大于 1。而在实际问题中，由于存在噪声样本点，它到由支持向量决定的超平面的距离小于 1，此时如果我们选择这些样本点作为支持向量，则得到的间隔会变小，模型的泛化能力大大减弱；所以选择引入松弛变量，容忍一些样本点到超平面的距离比支持向量到超平面的距离还小；

引入**松弛变量**ξ_i，$\xi_i\ge 0$，

约束条件变为: $Y_i(W^T X_i + b) \ge 1 - \xi_i$

当样本点为支持向量的时候，$\xi_i = 0$，当样本点为噪声数据时，$\xi_i \succ 0$；

进一步得到软间隔 SVM 要求解的问题：

$$\begin{cases} \min\left(\frac{1}{2}\|W\|^2 + C\sum_{i=1}^{m}\xi_i\right)(C > 0, \xi_i > 0) \\ s.t. Y_i(W^T X_i + b) \ge 1 - \xi_i \end{cases}$$

C 称为惩罚系数，$C\sum_{i=1}^{m}\xi_i$ **称为损失项；**在损失为定值时，若 C 越大，则松弛变量的值越小，

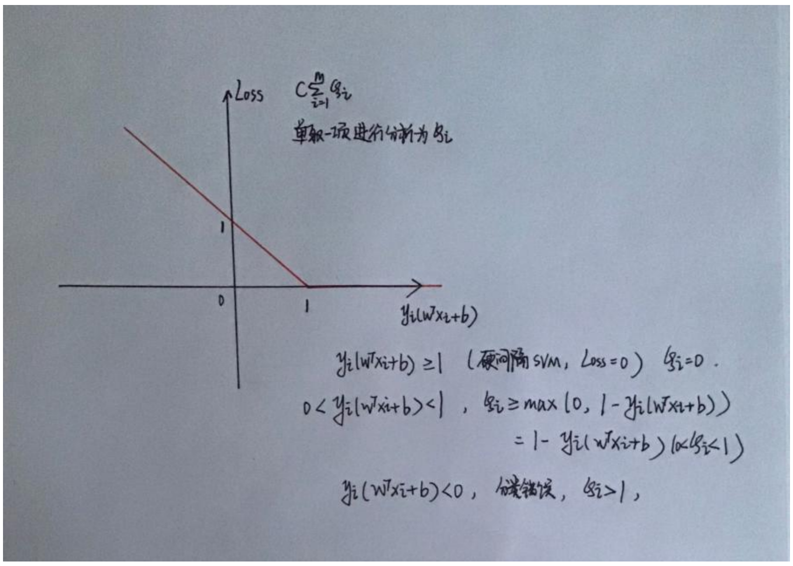
容忍度越小，模型越容易过拟合，当松弛变量的值为 0 时，变成了硬间隔 SVM；

在 KKT 条件中加入 $\frac{\partial L}{\partial \xi_i} = 0$ 这一项，继续用用对偶问题和 SMO 算法求解即可。

下面推导以下折页损失函数：

$$\begin{matrix} \xi_i \ge 1 - Y_i(W^T X_i + b) \\ \xi_i \ge 0 \end{matrix} \rightarrow \xi_i \ge \max(0, 1 - Y_i(W^T X_i + b))$$

绘制损失函数的图像：



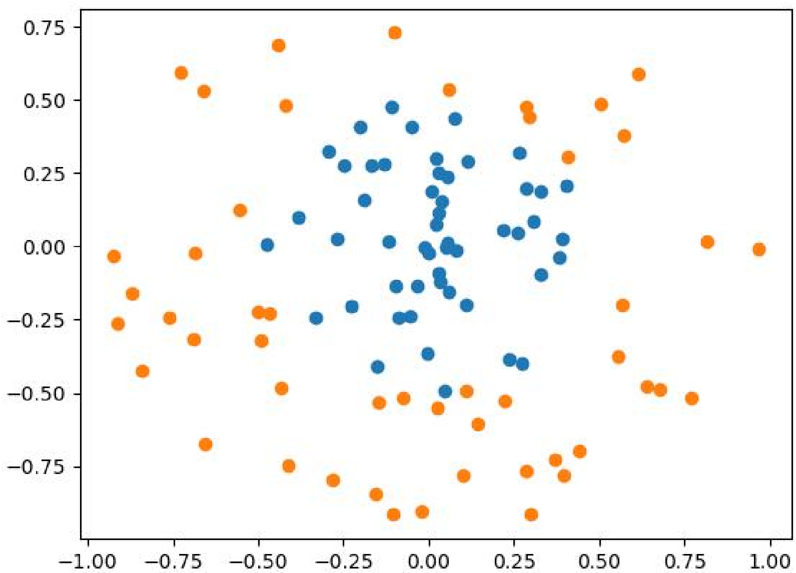
这部分知识与深度学习中的 SVM 损失函数有密切关系，但在此报告中不做研究。

2.6 支持向量机算法的代码实现

为了实现简单，以二维平面上的数据为例，数据格式如下：

```
testSetRBF2.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
0.676771  -0.486687 -1.000000
0.008473  0.186070  1.000000
-0.727789 0.594062  -1.000000
0.112367  0.287852  1.000000
0.383633  -0.038068 1.000000
-0.927138 -0.032633 -1.000000
-0.842803 -0.423115 -1.000000
-0.003677 -0.367338 1.000000
0.443211  -0.698469 -1.000000
-0.473835 0.005233  1.000000
0.616741  0.590841  -1.000000
0.557463  -0.373461 -1.000000
-0.498535 -0.223231 -1.000000
-0.246744 0.276413  1.000000
-0.761980 -0.244188 -1.000000
0.641594  -0.479861 -1.000000
-0.659140 0.529830  -1.000000
-0.054873 -0.238900 1.000000
-0.089644 -0.244683 1.000000
```

数据分布情况大致如下图所示：



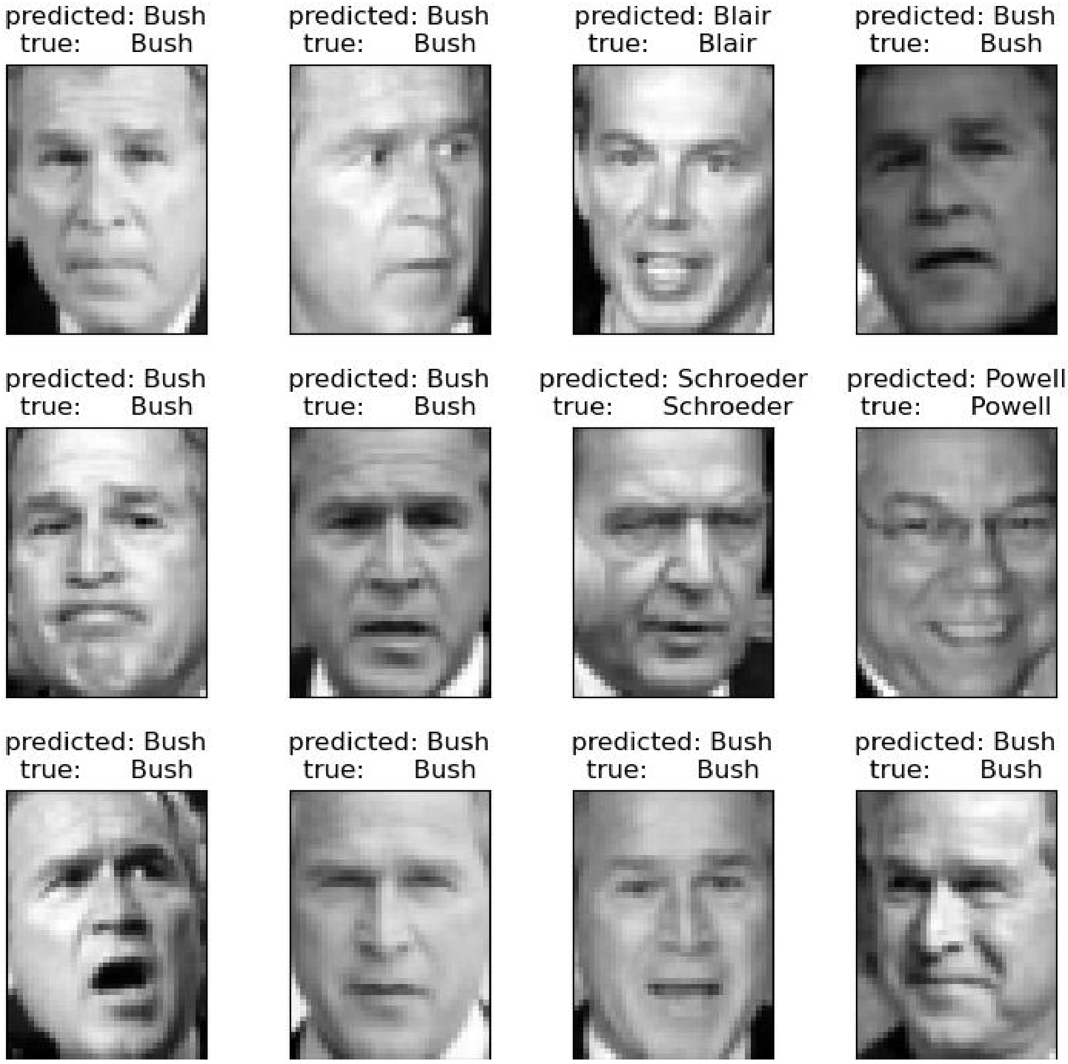
用测试集进行测试后，分类的准确率在 83.3%，效果还不错；

实验测试代码在附件 1 中

三．利用 PCA 降维和支持向量机算法实现人脸的分类任务

选用的数据集是 lfw-funneled 数据集，采集了 7 个知名人物的面部图片，共有 1288 张照片，每个图片的特征为 1850，然后利用 PCA 降维使特征减小到 150 个，利用 SVM 进行分析,最后的准确率也不错。

最后的输出结果为：



代码见附件 2