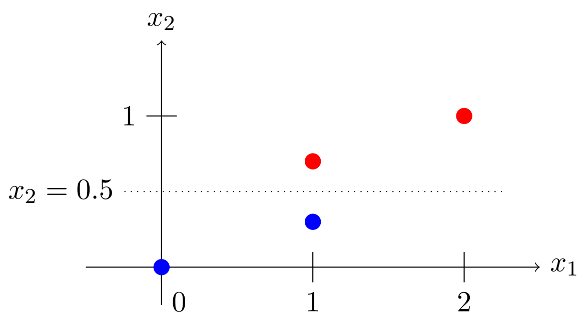
Introduction to Machine Learning
Problem Solutions: Support Vector Machines

Prof. Sundeep Rangan

1. (a) The best way to do this is first draw a scatter plot of the four points as shown in the figure below where the red circles represent $y = 1$ and the blue circles are for the class $y = -1$.



We see that we can separate the points with the classifier,

$$\hat{y} = \begin{cases} 1 & \text{if } x_2 > 0.5 \\ -1 & \text{if } x_2 < 0.5. \end{cases}$$

We then rewrite the classifier as,

$$\hat{y} = \begin{cases} 1 & \text{if } z > 0, \\ -1 & \text{if } z < 0, \end{cases} \qquad z = b + w_1 x_1 + w_2 x_2 < 0,$$

where $b = -0.5$, $w_1 = 0$ and $w_2 = 1$.

(b) Let

$$z_i = b + w_1 x_{i1} + w_2 x_{i2} = -0.5 + x_{i2}.$$

We evaluate z_i and $y_i z_i$ for each sample:

x_{i1}	0	1	1	2
x_{i2}	0	0.3	0.7	1
y_i	-1	-1	1	1
z_i	-0.5	-0.2	0.2	0.5
$y_i z_i$	0.5	0.2	0.2	0.5

Since we require $z_i y_i \geq \gamma$ for all i, the largest value of γ we can take is $\gamma = 0.2$.

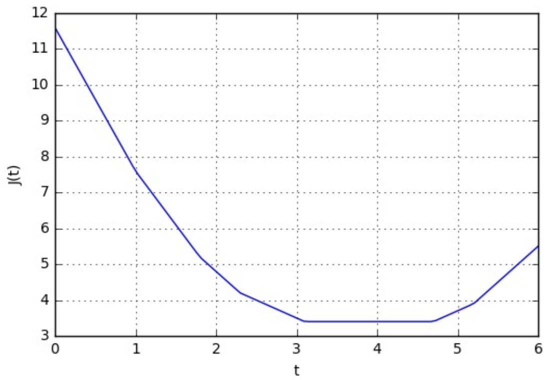


Figure 1: The objective function $J(t)$

(c) The margin is

$$m = \frac{\gamma}{\|\mathbf{w}\|} = \frac{0.2}{\sqrt{0+1}} = 0.2.$$

(d) The points on the margin are the ones where $z_i y_i = \gamma$. These are $(1, 0.3)$ and $(1, 0.7)$.

2. (a) You can compute and plot $J(t)$ with the following code. The figure is shown in Fig. 1.

```
x = np.array([0,1.3,2.1,2.8,4.2,5.7])
y = np.array([-1,-1,-1,1,-1,1])
tvals = np.linspace(0,6,100)
J = []
for t in tvals:
    z = x-t
    Jt = np.sum(np.maximum(0,1-y*z))
    J.append(Jt)
J = np.array(J)

plt.plot(tvals, J)
plt.xlabel('t')
plt.ylabel('J(t)')
plt.grid()
plt.savefig('Jt.jpg')
```

(b) From Fig. 1, we see that $t = 4$ is a minimizer. In fact, $J(t)$ is flat around the minima, so other values of t close to $t = 4$ would also minimize the function.

(c) We can compute the slack variables by the python code:

```
t = 4
z = x-t
eps = np.maximum(0, 1-y*z)
eps
```

```
>> array([ 0. ,  0. ,  0. ,  2.2,  1.2,  0. ])
```

(d) We see that $\epsilon_i > 1$ for samples $i = 3, 4$ so both of these samples will be misclassified (and violate the margin).

3. Consider an image recognition problem, where an image $\mathbf{X}$ and filter $\mathbf{W}$ are 4×4 matrices:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

(a) Going down the columns of $\mathbf{X}$ and $\mathbf{W}$, the vectors are:

$$\begin{aligned} \mathbf{x} &= [0,0,0,0,\ 0,0,0,0,\ 0,1,1,1,\ 0,0,0,0] \\ \mathbf{w} &= [0,0,0,0,\ 0,1,1,0,\ 0,1,1,0,\ 0,0,0,0] \end{aligned}$$

(b) The inner product will be

$$z = \mathbf{w}^\mathsf{T}\mathbf{x} = \sum_{i=1}^{16} x_i w_i.$$

Since x_i and $w_i = 0$ or 1, $x_i w_i = 1$ only on the pixels where $x_i = w_i = 1$. Hence $z = \mathbf{w}^\mathsf{T}\mathbf{x}$ is the number of pixels where two images overlap. Thus, we have $z = 2$.

(c) If $\mathbf{X}$ is shifted to the right we have

$$\mathbf{X}_{\text{right}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This has no overlap with $\mathbf{W}$, so $z = \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{right}} = 0$.

(d) If $\mathbf{X}$ is shifted to the left we have

$$\mathbf{X}_{\text{left}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

This overlaps with two pixels in $\mathbf{W}$. Hence, $z = \mathbf{w}^\mathsf{T}\mathbf{x}_{\text{left}} = 2$.

(e) The python commands are `x = Xmat.ravel()` and `Xmat = x.reshape([4,4])`.

4. (a) You can use the python code below. Note the use of `meshgrid` command. The matrix `xpmat` has rows equal to `xp[j]` and `xmat` has columns equal to `x[i]`, Therefore, the matrix `dist` has elements `dist[i,j] = (x[i]-xp[j])**2`, which are the squared distances need to compute z.

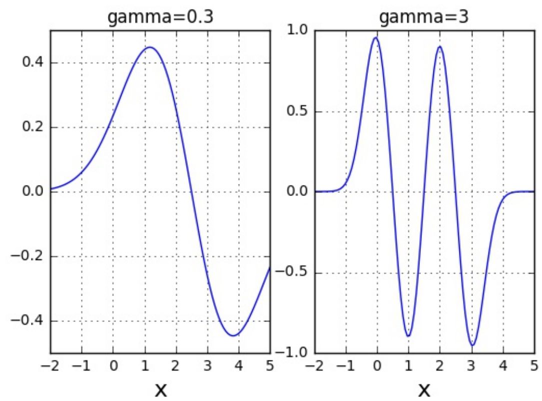


Figure 2: The function z in Problems 4(a) and (b)

```
x = np.array([0,1,2,3])
y = np.array([1,-1,1,-1])

def plot_zrbf(x,y,a,gam):
    xp = np.linspace(-2,5,100)
    xpmat,xmat = np.meshgrid(xp,x)
    dist = (xpmat-xmat)**2
    z = (y*a).dot(np.exp(-gam*dist))
    plt.plot(xp,z)
    plt.grid()
    plt.xlabel('x',fontsize=16)

plt.subplot(1,2,1)
a = np.array([0,0,1,1])
plot_zrbf(x,y,a,gam=0.3)
plt.title('gamma=0.3')

plt.subplot(1,2,2)
a = np.array([1,1,1,1])
plot_zrbf(x,y,a,gam=3)
plt.title('gamma=3')

plt.savefig('rbf.png')
```

The result function z is plotted in the left of Fig. 2.

(b) The function z for $\gamma = 3$ is plotted in the right of Fig. 2.

(c) The classifier takes $\hat{y}_i = \text{sign}(z_i)$. The resulting values are shown in the table below. We see that the classifier in part (b) makes no errors. Since it uses a higher γ it is able to fit the data better.

x_i	0	1	2	3
y_i	1	-1	1	-1
$\hat{y}_i = \text{sign}(z_i)$ for (a)	1	1	1	-1
$\hat{y}_i = \text{sign}(z_i)$ for (b)	1	-1	1	-1