Homework #7

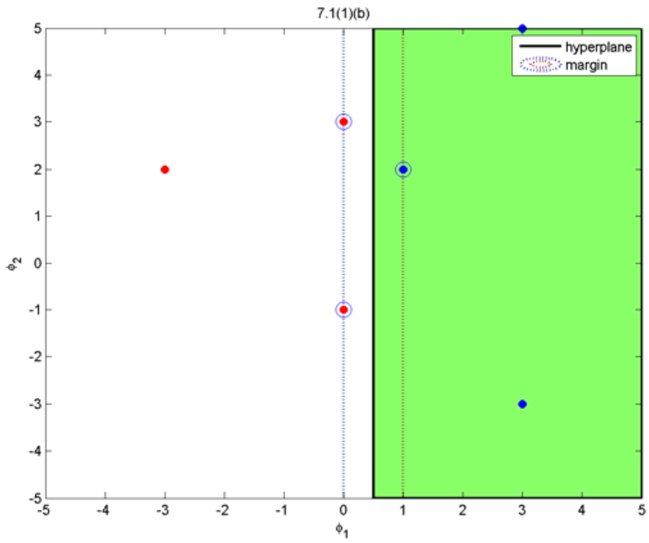
7.1 Transforms: Explicit versus Implicit

(1)

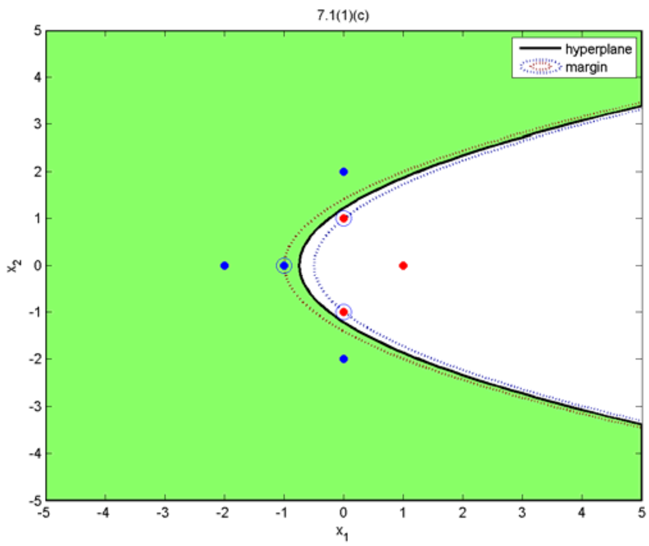
(a) The set Φ would be

$x_1 = (1,0), y = -1$	$x_2 = (0,1), y = -1$	$x_3 = (0,-1), y = -1$	
$x_4 = (-1,0), y = 1$	$x_5 = (0,2), y = 1$	$x_6 = (0,-2), y = 1$	$x_7 = (-2,0), y = 1$

(b) The optimal separating hyperplane in Φ would be $x = 0.5$, show as follow:



(c) The optimal separating hyperplane in χ would be $(x)_2^2 - (x)_1 = 1.5$, show as follow:



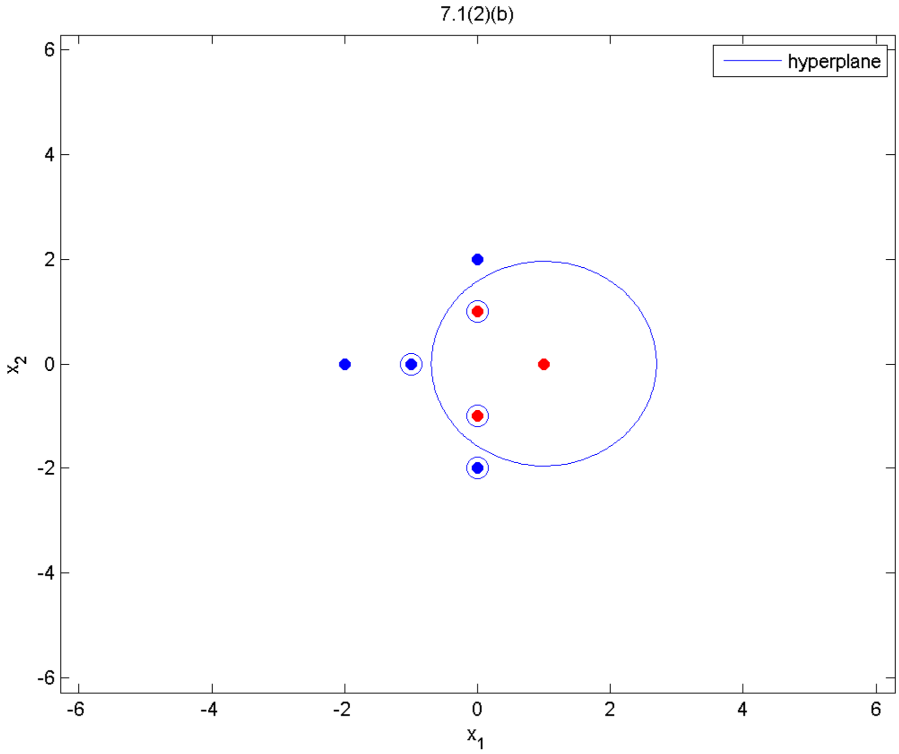
(2)

(a) The optimal α would be

0	0.185	1.222	0.889	0	0.518	0

That mean we could have 4 support vectors in the data.

(b) The equation in χ look like



Where we use ezplot function in MATLAB and get an ugly form of it

$$f(x) = -\frac{668}{360288} * (1 + x2)^2 - \frac{11}{9} * (1 - x2)^2 + \frac{8}{9} * (1 - x1)^2 + \frac{14}{27} * (1 - 2 * x2)^2 - 5/3$$

(3) They aren't the same curve, and they have a little chance to be the same curve. The kernel function $(1 + x \cdot x')^2$ is mapping to coordinate transformation $\phi(x)$ that can represent all of quadratic functions, and the curve we got at problem 7.1(1) is a quadratic function too. Anyway, they are indeed the different function.

7.2 A Leave-One-Out Bound of Support Vector Machine

(1) Consider $E_n(\beta)$ with $\hat{\beta}$

$$\hat{\beta} = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_{N-1}^*)$$

Therefore, the function $\operatorname{argmin}_\alpha E_N(\alpha^*) = \operatorname{argmin}_\beta E_{N-1}(\hat{\beta})$ when $\alpha_N^* = 0$, and we could say that $\hat{\beta}$ satisfies all constraints of (B) because α^* is optimal solution of (A).

(2) Assume that $\operatorname{argmin}_\beta E_{N-1}(\hat{\beta})$ is not the best solution, we got a better solution β'. Then use the β' combine the optimal $\alpha_N^* = 0$, and get the optimal $\min_\alpha E_N(\alpha') \leq \min_\alpha E_N(\alpha^*)$, which is a contradiction. Therefore, we have

$$E_{N-1}(\hat{\beta}) = E_{N-1}(\beta^*)$$

(3) The leave-one-out error of SVM is upper bounded by the percentage of support vectors. This argument implies that if you take off the non-support vector, the training error wouldn't be changed. By the definition of support vector, α_n^* should be zero if the n^{th} point isn't a support vector. So if we take of a non-support vector, $\alpha_n^* = 0$, that imply the inequality

$$y_n(\langle w^*, \phi(x_n)\rangle - \theta^*) \geq 1$$

Which means the n^{th} point is in the right side and behind the max margin line at least 1 unit distance. So the leave-one-out error won't affected by non-support vectors, but only affected by support vector, therefore

$$\nu_c(\mathrm{SVM}, N) \leq \frac{\#SV}{N}$$

7.3 Experiments with Linear Support Vector Machine

(1) The error rates and the equations show as follow:

	$C = 0.01$	$C = 0.1$	$C = 1$	$C = 10$	$C = 100$
$\nu(g_C)$	0.49	0	0	0	0
$\hat{\pi}(g_C)$	0.488	0	0	0	0
θ	-0.8780	-0.0182	0.0114	-0.2394	-0.0218
w_1	0.1635	1.4709	3.6266	7.3035	13.5168
w_2	-0.1548	-1.4175	-3.7287	-7.6591	-13.3826

We can find that when C = 0.01, the max margin line goes very up and loss its function. The result shows that the constraint of ξ is really important to soft margin support vector machine because when C goes smaller, we loss the control under constraint $y_n(\langle\omega, x_n\rangle - \theta) \geq 1 - \xi_n$.

(2) The error rates and the equations show as follow:

	$C = 0.01$	$C = 0.1$	$C = 1$	$C = 10$	$C = 100$
$\nu(g_C)$	0.5	0.11	0.12	0.11	0.11
$\hat{\pi}(g_C)$	0.474	0.146	0.148	0.145	0.146
θ	-0.8339	0.0164	0.2353	0.2482	0.1986
w_1	0.1908	1.3861	2.9767	4.3298	4.4103
w_2	-0.1746	-1.2981	-2.5313	-3.5876	-3.7822

When C = 0.01, the max margin line still not work. However, in the other cases, the soft margin SVM works well.

7.4 Experiments with Nonlinear Support Vector Machine

(1) When kernel is $(1 + \mathrm{x} \cdot \mathrm{x}')^{\mathrm{d}}$. The result show as follow:

(a) Training error $\nu\left(g_{d,C}^{(1)}\right)$

(d, C) combination	$\mathrm{C} = 0.001$	$\mathrm{C} = 1$	$\mathrm{C} = 1000$
$\mathrm{d} = 3$	0.42	0.13	0.08
$\mathrm{d} = 6$	0.22	0.11	0.06
$\mathrm{d} = 9$	0.24	0.08	0.02

(b) Testing error $\hat{\pi}\left(g_{d,C}^{(1)}\right)$

(d, C) combination	$\mathrm{C} = 0.001$	$\mathrm{C} = 1$	$\mathrm{C} = 1000$
$\mathrm{d} = 3$	0.514	0.183	0.128
$\mathrm{d} = 6$	0.220	0.177	0.088
$\mathrm{d} = 9$	0.237	0.107	0.093

(c) Number of support vector $\frac{\#\mathrm{SV}}{\mathrm{N}}$

(d, C) combination	$\mathrm{C} = 0.001$	$\mathrm{C} = 1$	$\mathrm{C} = 1000$
$\mathrm{d} = 3$	0.85	0.48	0.31
$\mathrm{d} = 6$	0.84	0.44	0.27
$\mathrm{d} = 9$	0.68	0.32	0.22

We can see that when C goes up, which mean we have a thinner hyperplane, so our classifier can achieve less error in testing and training. In the other hand, when C drop down, the constraint become useless, it give us lots of support vectors; So we will get a bad classifier, just like we got in 7.3.

About d, the result seems like when we have a big d; thing will go easier. However the best testing error is in $(6{,}1000)$. The trend may over fitting in some sense.

(2) When kernel is $\exp\left(\frac{-(x-x')^2}{2\sigma^2}\right)$. The result show as follow:

(a) Training error $\nu\left(g_{d,C}^{(2)}\right)$

(σ, C) combination	$C = 0.001$	$C = 1$	$C = 1000$
$\sigma = 0.125$	0.42	0.04	0
$\sigma = 0.5$	0.42	0.12	0.04
$\sigma = 2$	0.42	0.13	0.13

(b) Testing error $\hat{\pi}\left(g_{d,C}^{(2)}\right)$

(σ, C) combination	$C = 0.001$	$C = 1$	$C = 1000$
$\sigma = 0.125$	0.514	0.111	0.172
$\sigma = 0.5$	0.514	0.183	0.087
$\sigma = 2$	0.514	0.195	0.177

(c) Number of support vector $\frac{\#SV}{N}$

(σ, C) combination	$C = 0.001$	$C = 1$	$C = 1000$
$\sigma = 0.125$	0.92	0.68	0.45
$\sigma = 0.5$	0.85	0.48	0.24
$\sigma = 2$	0.84	0.74	0.43

Same situation, we almost fail on C = 0.001 in all of cases, and the trend is the same; C goes up, error goes down.

Take a look at σ, it don't like d we look above because we didn't observer the trend like σ goes up and error goes down. We still have the best testing error at $(0.5,1000)$, the same place as 7.4(1) in the table.