Solutions to exercises: week 4

Exercise 4.1
(a) The classification boundary is the perpendicular bisector of the line segment between (0,1) and (2,0). There are two support vectors.
(b) The classification boundary now becomes the vertical line through (1,0). All three points become support vectors.

Exercise 4.2
(a) One possibility is to consider (0,0) for one class, (1,1) and (2,0) for the other class, and (1,0) as a test point and see how the classification of the last point changes with different scalings of the first feature.

Exercise 4.3
(a) The two solutions will be the same when the number of support vectors is three in the 2D case. LDA will always have three "support vectors" in this 2D setting.

Exercise 4.5
(a) Using the Taylor expansion, one gets $\exp(-(x-\chi)^2) = \exp(-x^2)\exp(-\chi^2)\exp(2x\chi) = \exp(-x^2)\exp(-\chi^2)\sum \frac{(2x\chi)^n}{n!}$. Rewriting this to $\exp(-x^2)\exp(-\chi^2)\sum \frac{\sqrt{2^n}x^n}{\sqrt{n!}}\frac{\sqrt{2^n}\chi^n}{\sqrt{n!}}$ (or something similar), one sees that the feature mapping should be $\phi(x) := \exp(-x^2)(1, \sqrt{2}x, \frac{2}{2!}x^2, \frac{2^{3/2}}{3!}x^3, \ldots)^T$. (If you do not see this, simply calculate the inner product $\phi(x)^T\phi(\chi)$.
(b) ∞.

Exercise 4.6
(a) $\|x - \frac{1}{N_C}\sum x_i^C\|^2 = \langle x - \frac{1}{N_C}\sum x_i^C, x - \frac{1}{N_C}\sum x_i^C\rangle$. Expanding and some minor rewriting gives $\langle x,x\rangle - \frac{2}{N_C}\sum\langle x, x_i^C\rangle + \frac{1}{N_C^2}\sum_i\sum_j\langle x_i^C, x_j^C\rangle$.
(b) Realize that for the Gaussian kernel, we have $K(x,\chi) = 1$ if $x = \chi$. Comparing samples from class C and class K gives: $k(x,x) - \frac{2}{N_C}\sum k(x,x_i^C) + \frac{1}{N_C^2}\sum_i\sum_j k(x_i^C, x_j^C) = k(x,x) - \frac{2}{N_K}\sum k(x,x_i^K) + \frac{1}{N_K^2}\sum_i\sum_j k(x_i^K, x_j^K)$. This can be simplified to the following equation for the decision boundary: $\frac{1}{N_C}\sum k(x,x_i^C) - \frac{1}{N_K}\sum k(x,x_i^K) + c = 0$, where c is a constant that merely depends on the training data. $\frac{1}{N_C}\sum k(x,x_i^C)$ basically provides a Parzen estimate of class C.

Exercise 4.7
(a) Considering `tst*svc(a,('rbf',.10**i,1))*testc` for `i=-3:3`, typically gives me an optimum at `i=0`.
(b) Compare the formulas for the Parzen densities on pages 51 and 52 and, for instance, formula (4.72) for the SVM. The latter is basically going to be a weighted version of the Parzen density estimator.
(c) One might not have to calculate the distance to all points in the training set, as for the Parzen classifier, but one only needs the distance to the support vectors, which may considerably fewer than the number of samples in the training set.

Exercise 4.8
(a) My code looks like:

```
import numpy as np
import prtools as pr
import matplotlib.pyplot as plt

a = pr.gendatb([200,200])
s = [0.2,0.5,1.0,2.0,5.0,7.0,10.0,25.];
C = 1.
n = len(s)
k = 10
e_cv = np.zeros((n,1))
```

```
for i in range(n):
    u = pr.svc([],('rbf',s[i],C))
    e = pr.prcrossval(a,u,k)
    e_cv[i] = np.mean(e)

print(s)
print(e_cv)
```
Depending on the exact dataset, often saround 2 is optimal.

Exercise 4.9
(a) Fill in $A = I_d$ and $B = X^T$ and work out.
(b) For x an unobserved vector and X the training data, we have $x^T w = x^T(X^T X + \lambda I)^{-1} X^T Y = x^T X^T (XX^T + \lambda I)^{-1} Y$. (Take note of the confusion between row vectors, in X, and the column vector x!) From the last expression, we see that $x^T X^T$ is a vector with inner products between the test vector and all training vectors. Similarly, XX^T is an $N \times N$-matrix containing all inner products between all training feature vectors. Blimey, we have kernelized ridge regression!
(c) $(x^T z + c)^2 = (x^2 z^2 + 2cxz + c^2) = (x^2, \sqrt{2}x, c)(z^2, \sqrt{2}z, c)^T$.
(g) A change in c only changes the scaling of the different feature and we saw already that nonregularized least squares regression is invariant to such scalings (and for very small λ we are basically performing nonregularized regression).

Exercise 4.10
(b) Of course you expected this behavior! When going up in degree, the variance typically increases, while the bias decreases. When moving to more and more data, the bias remains the same (more or less?), while the variance gets lower and lower.

Exercise 4.11
(a) One should get the impression that regularization can have quite a beneficial effect.
(b) It may be possible to easily do this in a nifty way using `prcrossval`, but I didn't try. I just wrote some loops that make sure that I run over enough random training sets. For all training set sizes, I found a values around 3 to be fairly optimal.
(c) I did not do the experiment :) but you should find a lower bias and a higher variance for the unregularized solutions.

Exercise 4.12
(a) Bias.
(b) variance