CS6510: Applied Machine Learning

Assignment 2

Saksham Mittal

12.03.2019

CS16BTECH11032

Assignment - 2 (Applied Machine Learning)

CS6510

Theory questions:

(1.) K_1 and K_2 are valid kernels.
So they can be decomposed.

$$K_1(x,y) = \phi_1(x)^T \phi_1(y)$$
$$K_2(x,y) = \phi_2(x)^T \phi_2(y)$$

(a) $K_1(x,z) + K_2(x,z)$

$$= \phi_1(x)^T\phi_1(z) + \phi_2(x)^T\phi_2(z)$$
$$= \underbrace{[\phi_1(x), \phi_2(x)]}_{\phi_3(x)}{}^T \underbrace{[\phi_1(z), \phi_2(z)]}_{\phi_3(z)}$$
$$= \phi_3(x)^T \phi_3(z)$$

$\therefore$ $K(x,z)$ can be decomposed to $\phi_3(x)^T\phi_3(z)$

$\therefore$ $K(x,z)$ is a valid kernel function.

(b) Let ϕ_1 be a M dimension vector.

$\Rightarrow$ $\phi_1(x) = [\phi_{11}(x), \phi_{12}(x), \phi_{13}(x), \dots \phi_{1M}(x)]$

similarly,

$\phi_2(x) = [\phi_{21}(x), \phi_{22}(x), \dots \phi_{2N}(x)]$

$$K(x,z) = K_1(x,z) \cdot K_2(x,z)$$
$$= \left(\sum_{i=1}^{M} \phi_{1i}(x)\phi_{1i}(z)\right)\left(\sum_{j=1}^{N} \phi_{2j}(x)\phi_{2j}(z)\right)$$
$$= \phi_{11}(x)\phi_{11}(z)\left[\sum_{j=1}^{N} \phi_{2j}(x) \Sigma \phi_{2j}(z)\right]$$
$$+$$
$$\vdots$$
$$+ \phi_{1M}(x)\phi_{1M}(z)\left[\sum_{j=1}^{N} \phi_{2j}(x)\phi_{2j}(z)\right]$$
$$= \sum_{i=1}^{M}\sum_{j=1}^{N} [\phi_{1i}(x)\phi_{1i}(z)][\phi_{2j}(x)\phi_{2j}(z)]$$

$\therefore \quad K(x,z) = \Sigma\Sigma\, \phi_3^T(x)\, \phi_3(z)$

From part (a) we know that summation produces valid kernel.

$\therefore$ $K(x,z)$ is a valid kernel.

(c) $K(x,z) = h(K_1(x,z))$

h is a polynomial function with positive coefficients

i.e. $h(x) = a_0 + a_1 x + a_2 x^2 + \ldots,$

where $a_0, a_1, a_2 \ldots \geq 0$

$\therefore$ h can be seen as a composition of summation of powers of kernel K_1.

Since powers of K_1 are just ~~pro~~ repeated product of K_1 to itself.

$\therefore$ $h(K_1(x,z))$ is a valid kernel using results from (a) & (b) part.

(d) $K(x,z) = \exp(K_1(x,z))$

$\exp(x)$ can be expanded using Taylor's series

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots\right)$$

$\therefore$ $\exp(x)$ is just a polynomial function with positive coefficients. (which is proved to be a valid kernel in part (c)).

$\therefore$ $K(x,z)$ is a valid kernel.

(e) $K(x,z) = \exp\left[-\frac{\|x-z\|^2}{\sigma^2}\right]$

we can write $-\frac{\|x-z\|^2}{\sigma^2}$ as:

$$-\frac{1}{\sigma^2}\left(\|x\|^2 + \|z\|^2 - 2x^Tz\right) = -\frac{x^Tx}{\sigma^2} - \frac{z^Tz}{\sigma^2} + \frac{2x^Tz}{\sigma^2}$$

$$\therefore \exp\left(\frac{-\|x-z\|^2}{\sigma^2}\right) = \exp\left(-\frac{x^T x}{\sigma^2}\right)\exp\left(-\frac{z^T z}{\sigma^2}\right)\exp\left(\frac{-2x^T z}{\sigma^2}\right)$$

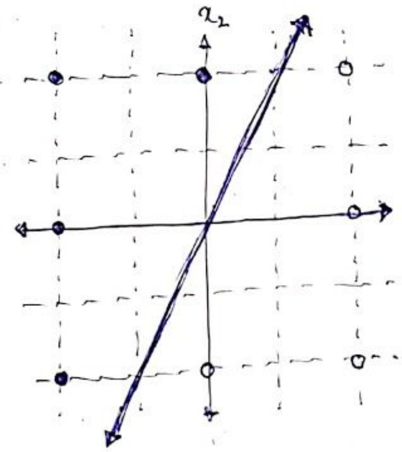
since $x^T z$, $x^T x$, & $z^T z$ is an inner product, & from part (d), we get

$\exp\left(-\frac{x^T x}{\sigma^2}\right)$, $\exp\left(-\frac{z^T z}{\sigma^2}\right)$, $\exp\left(-\frac{2x^T z}{\sigma^2}\right)$ are valid kernels.

So, their product will also be a valid kernel from part (b).

$\Rightarrow K(x,z)$ is a valid kernel.

(2.) (a)



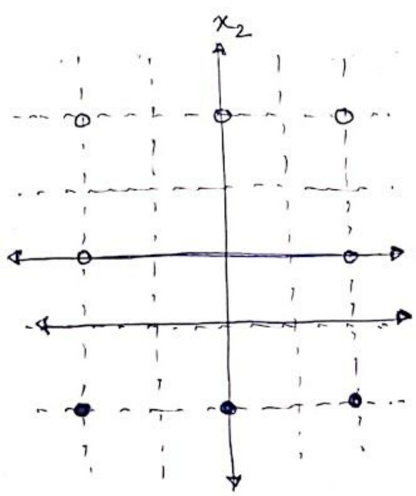
$w^T x + b = 0$

equ^n →

$2x_1 - x_2 + 0 = 0$

$b = 0$
$w^T = [2, -1]$

(b)



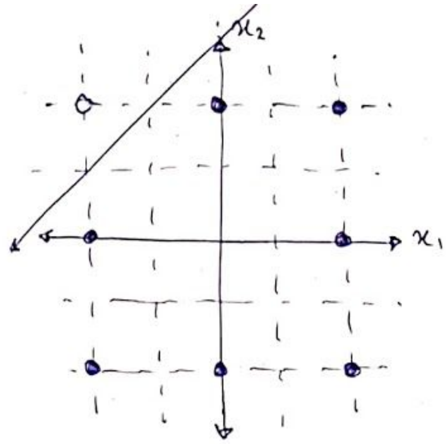
$w^T x + b = 0$

equ^n →

$0 \cdot x_1 + x_2 + 1 = 0$

$b = 1$
$w^T = [0, 1]$

(c)



$\omega^T x + b = 0$

equ^n →

$x_1 - x_2 + 3 = 0$

$b = 0$

$\omega^T = [1, -1]$

(3.) $x_1 = [1, 1]$ $\quad$ $x_2 = [1, -1]$

$y_1 = 1$ $\quad$ $y_2 = -1$.

$$\text{Error} = \frac{1}{n} \Sigma (\omega^T x - y)^2$$

$$= \frac{1}{2} (\omega_1 x_{11} + \omega_2 x_{12} - y_1)^2 + \frac{1}{2}(\omega_1 x_{21} + \omega_2 x_{22} - y_2)^2$$

$$E = \frac{1}{2}(\omega_1 + \omega_2 - 1)^2 + \frac{1}{2}(\omega_1 - \omega_2 + 1)^2$$

(a) $\frac{\partial E}{\partial \omega_1} = 0$ & $\frac{\partial E}{\partial \omega_2} = 0$

$\Rightarrow (\omega_1 + \omega_2 - 1) + (\omega_1 - \omega_2 + 1) = 0 \Rightarrow \boxed{\omega_1 = 0}$

$\frac{\partial E}{\partial \omega_2} = 0 \Rightarrow (\omega_1 + \omega_2 - 1) + (\omega_1 - \omega_2 + 1)(-1) = 0$

$\Rightarrow \boxed{\omega_2 = 1}$

checking $\frac{\partial^2 E}{\partial \omega_1^2}$ & $\frac{\partial^2 E}{\partial \omega_2^2}$

Also, $\frac{\partial E}{\partial \omega_1 \omega_2} = 0$

$\frac{\partial E}{\partial \omega_2 \omega_1} = 0$

$\frac{\partial^2 E}{\partial \omega_1^2} = 2 > 0$ $\quad$ $\frac{\partial^2 E}{\partial \omega_2^2} = 2 > 0$

∴ $\omega_1 = 0, \omega_2 = 1$ is the point of minima (i.e. the ~~p~~ surface is centered on $(0,1)$) and the curvature of surface along $\omega_1 = (2\omega_1)$

& " " " $\omega_2 = (2\omega_2 - 2)$

∴ Curvature is upward.

(b) $\frac{\partial^2 E}{\partial w_1^2} = 2$, $\frac{\partial^2 E}{\partial w_2^2} = 2$, $\frac{\partial^2 E}{\partial w_1 w_2} = 0$, $\frac{\partial^2 E}{\partial w_2 w_1} = 0$

$\therefore$ Hessian matrix $= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

For finding eigen values,

$$\det\left(\begin{bmatrix} 2-\lambda & 0 \\ 0 & 2-\lambda \end{bmatrix}\right) = 0$$

$$\Rightarrow (\lambda - 2)^2 = 0$$

$\therefore$ The hessian matrix has a repeated eigen value $= 2$.

Programming Questions

[Python version used in assignment = 2.7.15]

4. SVMs:

[NOTE: For this problem, I have cleaned the data set(training and test data) by removing the extra columns(with spaces). I have included the data set in the submission.]

a) **To run the code: python SVM-4.py**

   Kernel used: linear

   Accuracy over test set = **0.9788**

   Number of support vectors = **28**

b) **To run the code: python SVM-4.py**

   Number of points used = **50**

   Accuracy over test set = **0.9811**

   Number of support vectors = **2**

   Number of points used = **100**

   Accuracy over test set = **0.9811**

   Number of support vectors = **4**

   Number of points used = **200**

   Accuracy over test set = **0.9811**

   Number of support vectors = **8**

   Number of points used = **800**

   Accuracy over test set = **0.9811**

   Number of support vectors = **14**

c) **To run the code: python SVM-4-c.py <Q> <C>**

i)

When C = 0.0001,

Training error(Q = 2) = 0.0089

Training error(Q = 5) = 0.0045

So, **False**.

ii)

When C = 0.001,

Support vectors(Q = 2) = 76

Support vectors(Q = 5) = 25

So, **True**.

iii)

When C = 0.01,

Training error(Q = 2) = 0.0045

Training error(Q = 5) = 0.0038

So, **False**.

iv)

When C = 1,

Test error(Q = 2) = 0.0189

Test error(Q = 5) = 0.0212

So, **False**.

d) **To run the code: python SVM-4-d.py**

Test error for C(0.01) = 0.0236

Train error for C(0.01) = 0.0038

Test error for C(1) = 0.0212

Train error for C(1) = 0.0045

Test error for C(100): 0.0189

Train error for C(100) = 0.0032

Test error for C(10000) = 0.0236

Train error for C(10000) = 0.0026

Test error for C(1000000) = 0.0236

Train error for C(1000000) = 0.0006

So, lowest Training error = 0.0006 for C = 10^6, and lowest Test error = 0.0189 for C = 100

5. SVMs (contd):

To run the code: python2 SVM-5.py

a) Number of support vectors used in linear kernel: **1084**

Train error using linear kernel: **0.0000**

Test error using linear kernel: **0.0240**

b) Number of support vectors used in RBF kernel: **6000**

Train error using RBF kernel: **0.0000**

Test error using RBF kernel: **0.5000**

Number of support vectors used in polynomial kernel: **1755**

Train error using polynomial kernel: **0.0000**

Test error using polynomial kernel: **0.0210**

6. Random Forests:

a) To run the random forest classifier: **python2 decision-tree-forest.py**

We consider m attributes for splitting, selected randomly from all columns of the data. The value m can be varied as 1, sqrt(columns), sqrt(columns) / 2, 2 * sqrt(columns), etc.

The data is split in 70-30 as training and test data. The code uses the code for decision tree from Assgn-1, and creates 100 trees(forest), and makes predictions for the test data. For a given test instance, the predicted value is the majority of the value predicted, and accuracy is computed.

Accuracy: **95.08**

To run the sklearn random forest classifier: **python2 sklearn-forest.py**

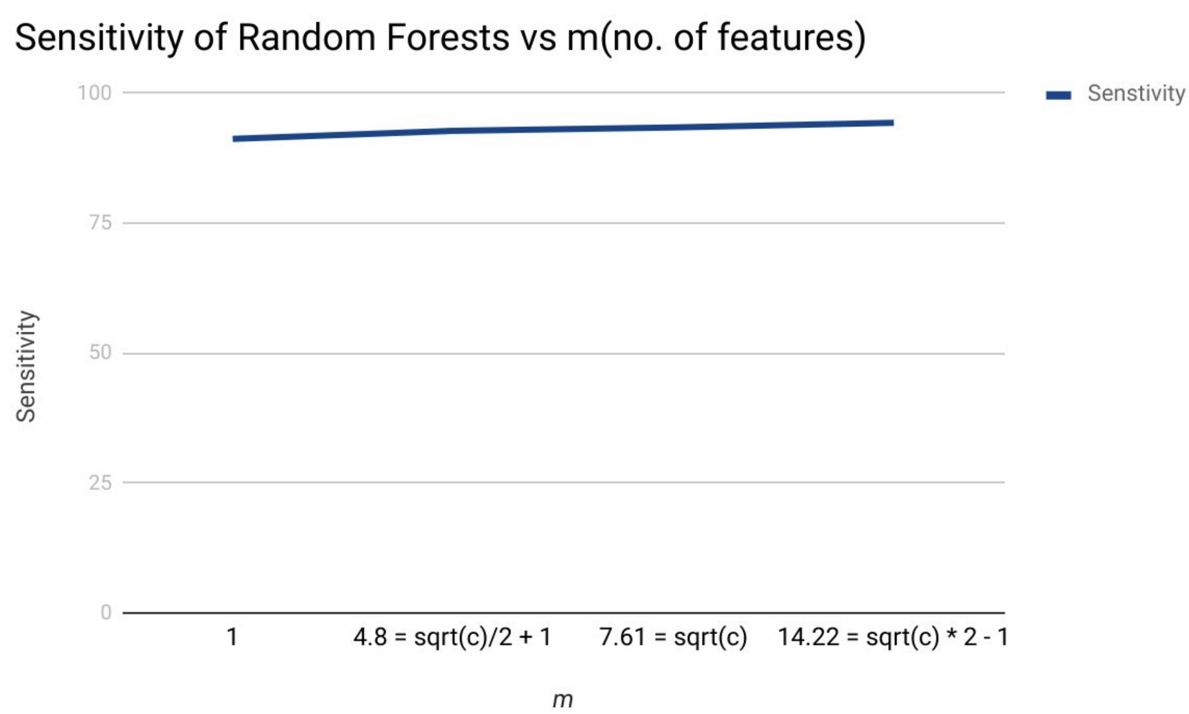
On using sklearn built-in random forest classifier,

Accuracy: **95.58**

[NOTE: The accuracy values may vary a little on every code execution]

b) For sensitivity, the formula used is:

Number of True positives/(Number of True positives + Number of false negatives)



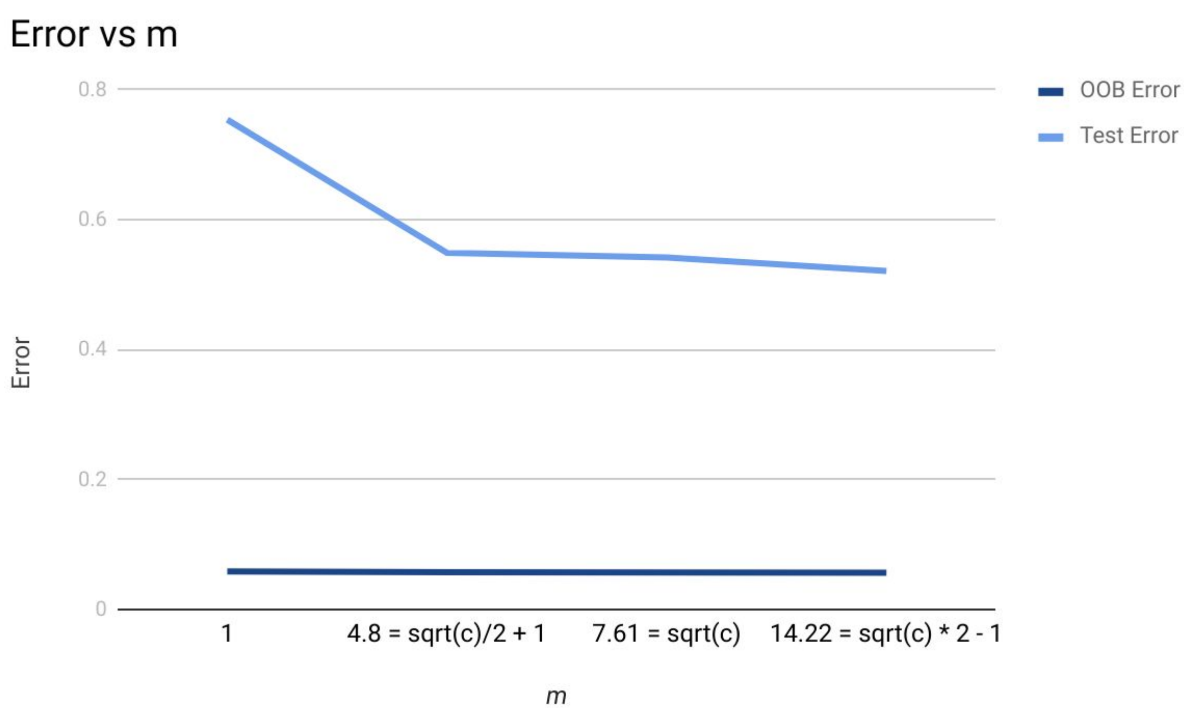
For exact values, see the following values:

m	Sensitivity
1	91.22
4.8 = sqrt(c)/2 + 1	92.77
7.61 = sqrt(c)	93.41
14.22 = sqrt(c) * 2 - 1	94.35

c) The code for OOB Error is written in the decision-tree-forest-oob.py file.

To run the code: **python decision-tree-forest-oob.py**

The plot of OOB Error and Test error on varying m is:



For Exact values, see the following:

m	OOB Error	Test Error
1	0.0584	0.754
4.8 = sqrt(c)/2 + 1	0.0575	0.549
7.61 = sqrt(c)	0.0568	0.542
14.22 = sqrt(c) * 2 - 1	0.0565	0.521