Introduction to machine learning Exercise set 5 report

Elias Annila 014328901

Pen and paper:

Problem 1:

a) There is positive correlation between a and b if $a = cb + d$ where c>0. In other words if b increases, a also increase. In this situation we can write $y = w_0 x_0 + w_1 x_{1...} w_n x_n$. As w_j is positive that means that if x_j increases y also increases, so there is positive correlation.

b) $y = w_0 x_0 + w_1 x_{1...} w_n x_n$ Again for there to be positive correlation between x_j and y it must be that y increases as x_j increases. This does not however mean that $w_j > 0$ as there is also possibility that for example $w_j = 0$ in which case there is no correlation between x_j and y , but if x_k is correlating positively with x_j and y with $w_k > 0$ it can still be that as x_j increases y also increases, but $w_j = 0$.

Problem 2:

a)

Let's suppose that there is a coefficient vector $(w0, w1, w2)$ such that
$XOR(x_{1,} x_2) = sign(w_0 + w_1 x_1 + w_2 x_2)$

As $XOR(x_{1,} x_2) = sign(w_0 + w_1 x_1 + w_2 x_2)$ we can make a following table:

x_1	x_2	$sign(w_0 + w_1 x_1 + w_2 x_2)$
-1	-1	-1
-1	$+1$	$+1$
$+1$	-1	$+1$
$+1$	$+1$	-1

If we insert our x_1 and x_2 values into our XOR functions we get following for equations which must all be true:

$$\begin{matrix} w_0 + w_1(-1) + w_2(-1) < 0 \\ w_0 + w_1(1) + w_2(1) < 0 \\ w_0 + w_1(-1) + w_2(1) > 0 \\ w_0 + w_1(1) + w_2(-1) > 0 \end{matrix} = \begin{matrix} w_0 - w_1 - w_2 < 0 \\ w_0 + w_1 + w_2 < 0 \\ w_0 - w_1 + w_2 > 0 \\ w_0 + w_1 - w_2 > 0 \end{matrix}$$

if we solve w_1 from these we get:

$$\begin{matrix} w_0 - w_2 < w_1 \\ -w_0 - w_2 > w_1 \\ w_0 + w_2 > w_1 \\ -w_0 + w_2 < w_1 \end{matrix}$$

from this it would result that $\begin{matrix} w_0 - w_2 < -w_0 - w_2 \\ w_0 < 0 \end{matrix}$ and $\begin{matrix} -w_0 + w_2 < +w_0 + w_2 \\ w_0 > 0 \end{matrix}$

which is a contradiction.

b)

As above let's write the XOR function as a table:

x_1	x_2	$sign(w_0+w_1x_1+w_2x_2+w_3x_1x_2)$
-1	-1	-1
-1	$+1$	$+1$
$+1$	-1	$+1$
$+1$	$+1$	-1

Again let's insert our x values:
$$\begin{array}{c} w_0+w_1(-1)+w_2(-1)+w_3(-1)(-1)<0 \\ w_0+w_1(1)+w_2(1)+w_3(+1)(+1)<0 \\ w_0+w_1(-1)+w_2(1)+w_3(-1)(+1)>0 \\ w_0+w_1(1)+w_2(-1)+w_3(+1)(-1)>0 \end{array} = \begin{array}{c} w_0-w_1-w_2+w_3<0 \\ w_0+w_1+w_2+w_3<0 \\ w_0-w_1+w_2-w_3>0 \\ w_0+w_1-w_2-w_3>0 \end{array}$$

from this we can try to find a suitable order of w_0, w_1, w_2, w_3

$$\begin{array}{cc} w_0-w_1-w_2+w_3<w_0-w_1+w_2-w_3 & w_0-w_1-w_2+w_3<w_0+w_1-w_2-w_3 \\ -w_2+w_3<w_2-w_3 & -w_1+w_3<+w_1-w_3 \\ w_3<w_2 & w_3<w_1 \\ w_0+w_1+w_2+w_3<w_0-w_1+w_2-w_3 & w_0+w_1+w_2+w_3<w_0+w_1-w_2-w_3 \\ w_1+w_3<-w_1-w_3 & w_2+w_3<-w_2-w_3 \\ 2w_1+2w_3<0 & 2w_2+2w_3<0 \\ w_1<-w_3 & w_2<-w_3 \end{array}$$

from these solutions we can gather following (although we don't know the order of w_1 and w_2):

```
-a                    0                    a
-|---------------|----|-----|-----|-
w3                           w1    w2
```

So let's try to use some weights that follow these rules for our XOR function and see if it behaves correctly (we guess w_0 to be 0):

x_1	x_2	$sign(0+\frac{a}{3}x_1+\frac{a}{3}x_2-a\,x_1x_2)$
-1	-1	$sign(0-\frac{a}{3}-\frac{a}{3}-a)=sign(-\frac{5}{3}a)=-1$
-1	$+1$	$sign(0-\frac{a}{3}+\frac{a}{3}+a)=sign(a)=+1$
$+1$	-1	$sign(0+\frac{a}{3}-\frac{a}{3}+a)=sign(a)=+1$
$+1$	$+1$	$sign(0+\frac{a}{3}+\frac{a}{3}-a)=sign(-\frac{1}{3}a)=-1$

As the resulting XOR values match actual XOR values we can conclude that for example $w=(0,\frac{a}{3},\frac{a}{3},-a)\, when\, a<0$ is appropriate coefficient vector, so we can indeed represent XOR as linear classifier.

Problem 3:

a)

As a XOR b is (a OR b) AND !(a AND b) we want to make z_1 represent OR and z_2 represent !AND so we want to choose u weights so that following table is true.

x_1	x_2	z_1	z_2
+1	+1	$s(u_{10}+u_{11}x_1+u_{12}x_2)=+1$	$s(u_{20}+u_{21}x_1+u_{22}x_2)=-1$
+1	-1	$s(u_{10}+u_{11}x_1+u_{12}x_2)=+1$	$s(u_{20}+u_{21}x_1+u_{22}x_2)=+1$
-1	+1	$s(u_{10}+u_{11}x_1+u_{12}x_2)=+1$	$s(u_{20}+u_{21}x_1+u_{22}x_2)=+1$
-1	-1	$s(u_{10}+u_{11}x_1+u_{12}x_2)=-1$	$s(u_{20}+u_{21}x_1+u_{22}x_2)=+1$

If We choose $\begin{array}{l} u_{11}=1 \\ u_{12}=1 \\ u_{21}=-1 \\ u_{22}=-1 \end{array}$ we get:

x_1	x_2	z_1	z_2
+1	+1	$s(u_{10}+1+1)=s(u_{10}+2)$	$s(u_{20}-1-1)=s(u_{20}-2)$
+1	-1	$s(u_{10}+1-1)=s(u_{10})$	$s(u_{20}-1+1)=s(u_{20})$
-1	+1	$s(u_{10}-1+1)=s(u_{10})$	$s(u_{20}+1-1)=s(u_{20})$
-1	-1	$s(u_{10}-1-1)=s(u_{10}-2)$	$s(u_{20}+1+1)=s(u_{20}+2)$

From above table it is obvious we choose both u_{10} and u_{20} to be 1, as this results in z values on given x values as listed in below table, which corresponds to behavior of OR and !AND:

x_1	x_2	z_1	z_2
+1	+1	+1	-1
+1	-1	+1	+1
-1	+1	+1	+1
-1	-1	-1	+1

As we have now simulated a OR b as well as !(a AND b) we then need to simulate a AND b to combine the two conditions, so we choose v weights so that:

z_1	z_2	y
+1	+1	$s(v_0+v_1z_1+v_2z2)=+1$
+1	-1	$s(v_0+v_1z_1+v_2z2)=-1$
-1	+1	$s(v_0+v_1z_1+v_2z2)=-1$
-1	-1	$s(v_0+v_1z_1+v_2z2)=-1$

Again we choose $\begin{matrix} v_1=1 \\ v_2=1 \end{matrix}$ so we get:

z_1	z_2	y
+1	+1	$s(v_0+1+1)=s(v_0+2)$
+1	-1	$s(v_0+1-1)=s(v_0)$
-1	+1	$s(v_0-1+1)=s(v_0)$
-1	-1	$s(v_0-1-1)=s(v_0-2)$

So we choose $v_0=-1$

So to finish the neural network solves XOR when weights are: $\begin{matrix} v_0=-1 \\ u_{21}=-1 \\ u_{22}=-1 \\ v_1=1 \\ v_2=1 \\ u_{10}=1 \\ u_{21}=1 \\ u_{11}=1 \\ u_{12}=1 \end{matrix}$ so it is proven that the network can compute XOR operation given suitable weights.

b)

We are trying to show that $y=sign(v_0+v_1 z_1+v_2 z_2)$ doesn't give advantage over $y=sign(w_0+w_1 x_1+w_2 x_2)$ in regard to computing XOR function if we disregard sign functions in forming of z values. As $y=sign(w_0+w_1 x_1+w_2 x_2)$ was unable to solve XOR as shown in 2 a), we must show that neither can $y=sign(v_0+v_1 z_1+v_2 z_2)$ if we ignore sign function in forming of z. As our final choice $y=sign(v_0+v_1 z_1+v_2 z_2)$ is the same as in 3 a) where we were able to compute XOR I'll try to show that we can't choose u weights in determination of z values so that z_1 and z_2 would behave as OR and !AND, and thus the whole network can't compute XOR if we ignore the sign function.

Let's consider this table from part a):

x_1	x_2	z_1	z_2
+1	+1	$s(u_{10}+u_{11} x_1+u_{12} x_2)=+1$	$s(u_{20}+u_{21} x_1+u_{22} x_2)=-1$
+1	-1	$s(u_{10}+u_{11} x_1+u_{12} x_2)=+1$	$s(u_{20}+u_{21} x_1+u_{22} x_2)=+1$
-1	+1	$s(u_{10}+u_{11} x_1+u_{12} x_2)=+1$	$s(u_{20}+u_{21} x_1+u_{22} x_2)=+1$
-1	-1	$s(u_{10}+u_{11} x_1+u_{12} x_2)=-1$	$s(u_{20}+u_{21} x_1+u_{22} x_2)=+1$

Now let's remove the sign functions and insert x values:

x_1	x_2	z_1	z_2
+1	+1	$u_{10}+u_{11}+u_{12}=+1$	$u_{20}+u_{21}+u_{22}=-1$
+1	-1	$u_{10}+u_{11}-u_{12}=+1$	$u_{20}+u_{21}-u_{22}=+1$
-1	+1	$u_{10}-u_{11}+u_{12}=+1$	$u_{20}-u_{21}+u_{22}=+1$
-1	-1	$u_{10}-u_{11}-u_{12}=-1$	$u_{20}-u_{21}-u_{22}=+1$

It is clear from the table that we can't choose u_{10} or u_{20} so that we would get the desired OR and !AND behavior. For example $\begin{array}{c} u_{10}+u_{11}+u_{12}=+1 \\ u_{10}=1-(u_{11}+u_{12}) \end{array}$ $\begin{array}{c} u_{10}-u_{11}-u_{12}=-1 \\ 1-u_{11}-u_{12}-u_{11}-u_{12}=-1 \\ -2u_{11}-2u_{12}=0 \\ u_{11}=-u_{12} \end{array}$ as we can see u_{11} and u_{12} should cancel each other, which would mean that in $\begin{array}{c} u_{10}+u_{11}+u_{12}=+1 \\ u_{10}=+1 \end{array}$ and in $\begin{array}{c} u_{10}-u_{11}-u_{12}=-1 \\ u_{10}=-1 \end{array}$ which is a contradiction. Similary for z_2. As we cannot choose u values so that our z values would behave as OR and !AND our final $y=sign(v_0+v_1z_1+v_2z_2)$ function cannot function as XOR.

Programming:

Problem 4

a) Below are plots showing Kth degree polynomials fitted to the 30 data points.

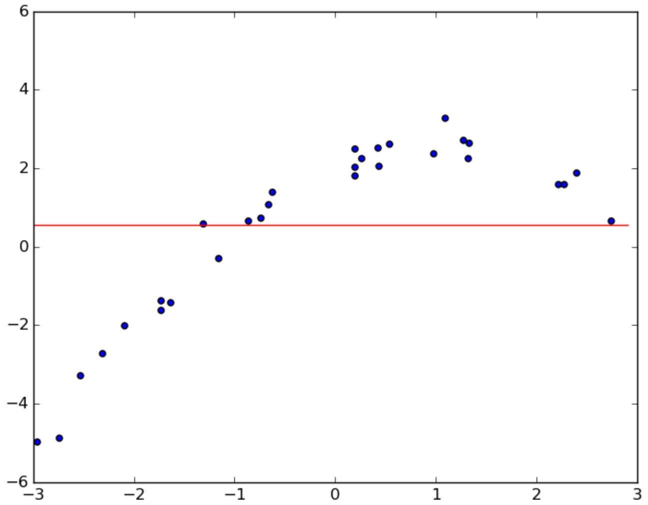

Illustration 2: K=0

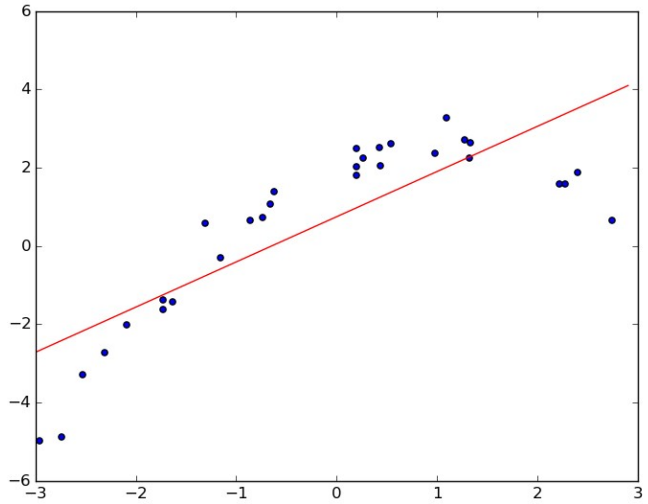

Illustration 1: K=1

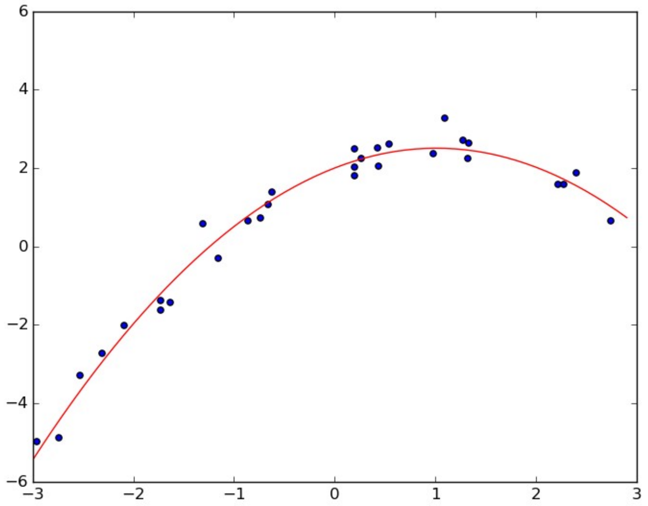

Illustration 4: K=2

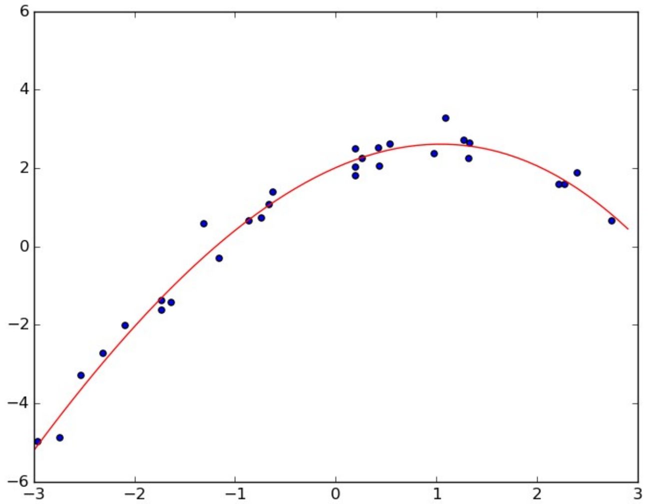

Illustration 3: K=3

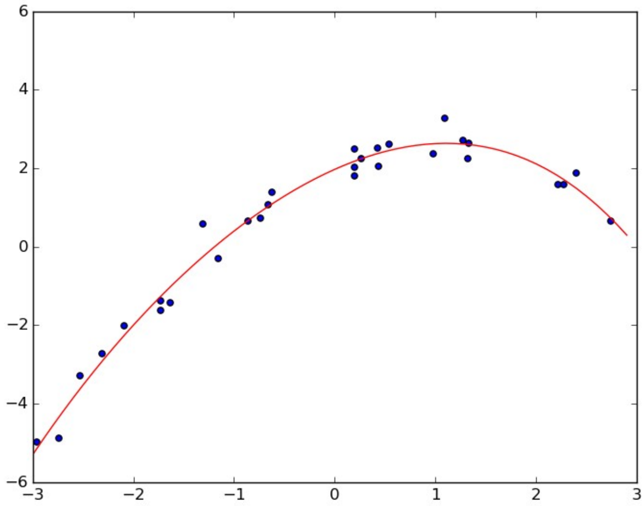

Illustration 5: K=4

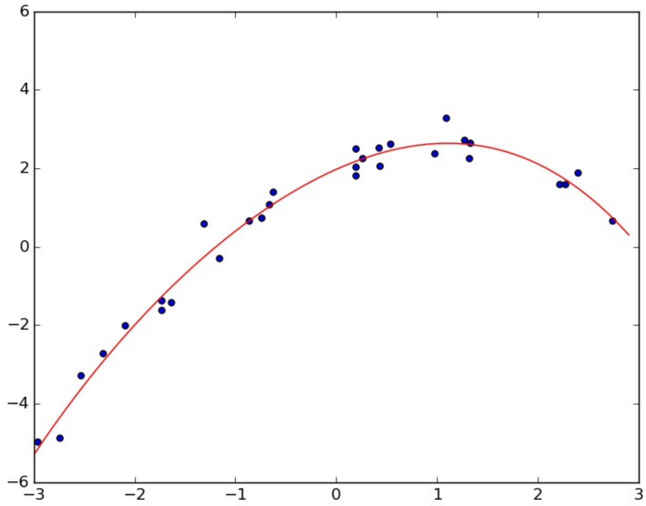

Illustration 6: K=5

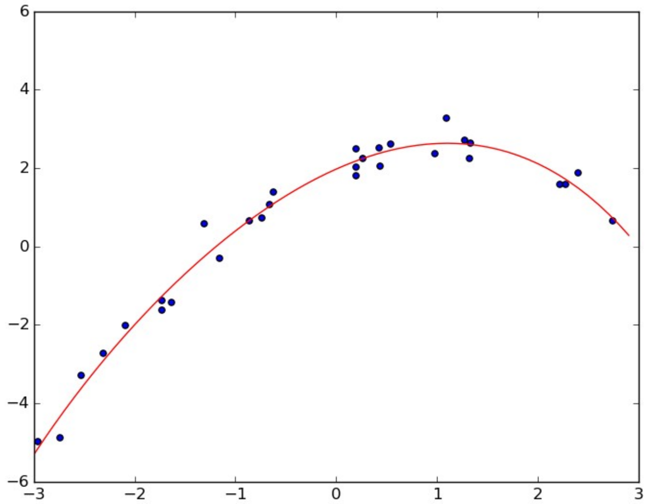

Illustration 8: K=6

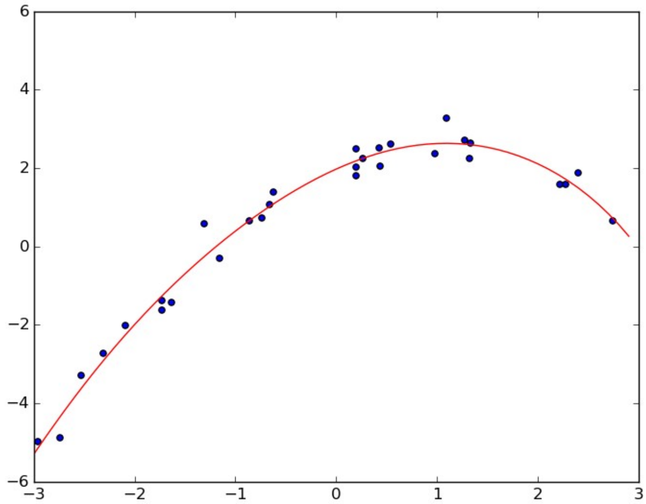

Illustration 7: K=7

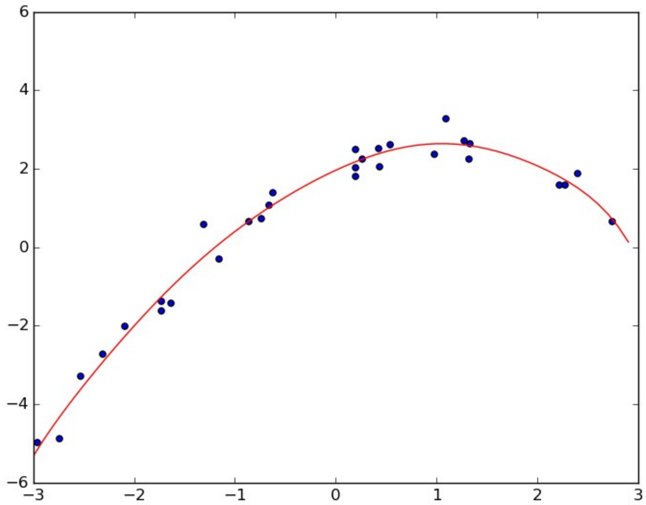

Illustration 9: K=8

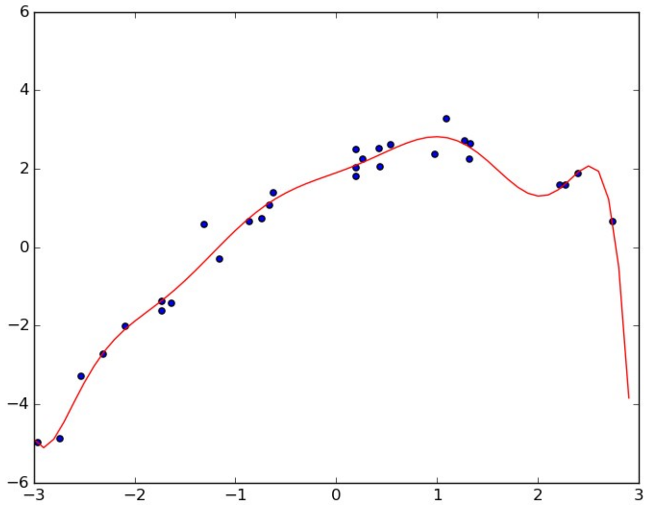

Illustration 10: K=9

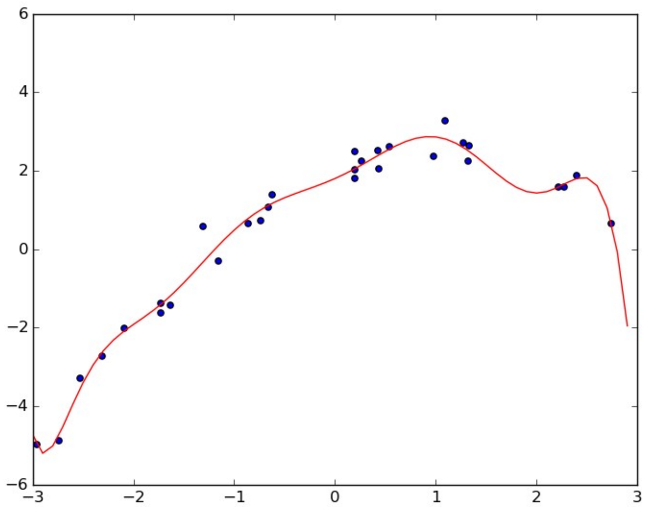

Illustration 11: K=10

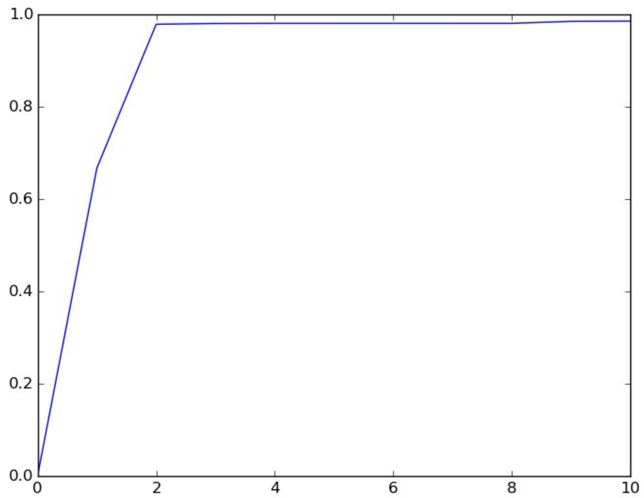

Illustration 12: Coefficient of determination plotted on each fitted polynomial

From the plots 1-11 it is apparent that as K increases the polynomial passes closer to data points. As can bee seen from illustration 12, R^2 increases rapidly on polynomial orders of 0-2 but starting from polynomial K=2 the increase is marginal. This is due to the original function that was used to draw y values being second order polynomial. Further increase of K is overfitting to the noise created to datapoints.

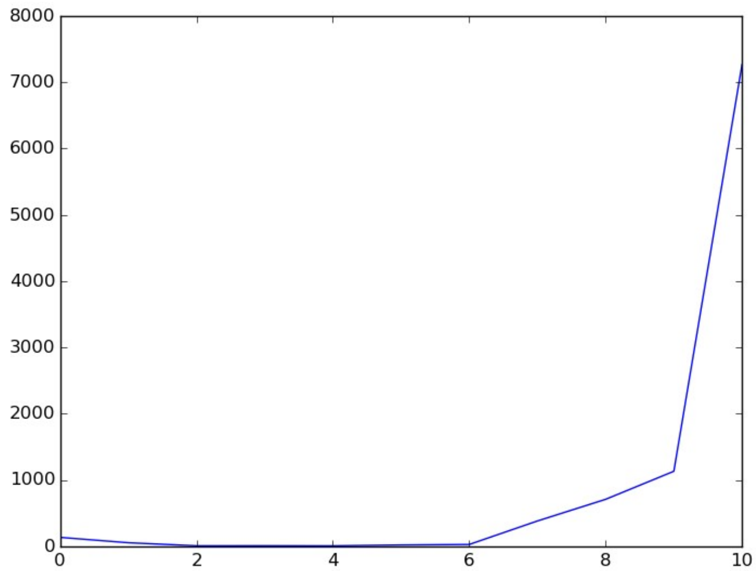

Illustration 13: Sum of square errors for each value of K as result of 10 fold cross validation

As can (maybe) be seen from illustration 13, or the output of the code, the sum of square errors improves until K=2 from where it starts to increase. As K gets larger the performance starts to degrade rapidly, as the high degree polynomial starts to oscillate wildly. The estimated coefficients for second degree polynomial were $-0.482\ x^2 + 1.043\ x + 1.836$ (obtained from the entire data) which are similar to actual training polynomial used: $-0.5x^2 + x + 2$, but as exptected don't quite match.