1 Kernel Methods

Answer the following questions regarding Kernel functions:
(a) Suppose you have a constant Kernel function, $k(x_i, x_j) = c$ for all x_i, x_j. Is k a symmetric positive semi-definite kernel?
$k(x_i, x_j) = c \quad \forall x_i, x_j$.
$k(x_i, x_j) = c = k(x_j, x_i)$ thus k is symmetric.
$v^T K v = c(\sum_i v_i, \sum_i v_i, \ldots, \sum_i v_i)^T v = c(v_1 \sum_i v_i, v_2 \sum_i v_i, \ldots, v_n c \sum_i v_i) = c(\sum_i v_i)^2$ which is ≥ 0 if and only if $c \geq 0$.
Thus if $c \geq 0$, then k is a symmetric semi positive definite.

(b) Assume that k_1 and k_2 are two symmetric, positive semi-definite kernels defined on the same space, let k be their minimum, $k(u, v) = min k_1(u, v), k_2(u, v)$. Is k also a symmetric positive semi-definite kernel?
$k(u, v) = min[k_1(u, v), k_2(u, v)] = min[k_1(v, u), k_2(v, u)] = k(v, u)$, then k is symmetric.
But K might not be positive semi definite, and below is a counter example:
Let

$$K_1 = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 5 & 1 \\ 0.5 & 1 & 1 \end{bmatrix}$$

which have eigen values $(0.3802, 1.3806, 5.2392)$ that are greater than 0. Thus K_1 is a symmetric positive semidefinite matrix.
Let

$$K_2 = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 1 \\ 0.5 & 1 & 5 \end{bmatrix}$$

which have eigen values $(0.7087, 1, 5.2913)$ that are greater than 0. Thus K_2 is a symmetric positive semidefinite matrix.
Then

$$K = min(K_1, K_2) = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 1 \\ 0.5 & 1 & 1 \end{bmatrix}$$

which have eigen values $(-0.118, 1, 2.118)$. Since one of the eigen values is negative, then K is not a positive semidefinite matrix.

(c) A symmetric, positive semi-definite kernel, k is defined on X. Let K denote the kernel

matrix corresponding to the set $x_1, \dots, x_n \in X$(that is, $K_{ij} = k(x_i, x_j)$).For $u \in X$, define $k_x(u) = (k(u, x1), \dots, k(u, xn))$, and for $u, v \in X$, define $\tilde{k}(u, v) = k(u, v) - k_x(u)^T K^{-1} k_x(v)$ assuming K has full rank. Is $\tilde{k}$ also a symmetric, positive semi-definite kernel?
Since $k(x_i, x_j)$ is a kernel function, then $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.

Thus $k_x(u)^T K^{-1} k_x(v) = \sum_i^n \sum_j^n k(u, x_i)^T k(v, x_j) K_{ij}^{-1} = \sum_i^n \sum_j^n \phi(u)\phi(x_i)^T \phi(x_j)\phi(v) K_{ij}^{-1} = \sum_i^n \sum_j^n \phi(u) K_{ij} K_{ij}^{-1} \phi(v)$
$\sum_i^n \sum_j^n \phi(u)\phi(v) = n^2 k(u, v)$.

Thus $\tilde{k}(u, v) = k(u, v) - n^2 k(u, v) = (1 - n^2) k(u, v)$.
So $\tilde{K} = (1 - n^2)K \leftrightarrow a^T \tilde{K} a = (1 - n^2) a^T K a \leq 0$ if $n > 1$. Thus $\tilde{k}$ is not a positive semi definite function. (Not a kernel function).

2 Support Vector Machines

Assume we have a dataset of $D = x_1, x_2, \dots, x_N$, each sample being d-dimensional raw feature vector.
Normal data "look like" $\leftrightarrow ||\phi(x_j) - \boldsymbol{c}||^2 \leq R^2 \quad \forall j$
Non-Normal data "look different" $\leftrightarrow ||\phi(x_j) - \boldsymbol{c}||^2 \geq R^2 \quad \forall j$

If the circle is small, then many data points become abnormal, so we need a small circle but not too small.
Primal:

$$\begin{aligned} min_{R, \boldsymbol{c}, \{\eta\}_n} \quad & 0.5R^2 + c \sum_n \xi_n \\ subject \quad to \quad ||\phi(x_n) - \boldsymbol{c}||_2^2 \quad & \leq \quad R^2 + \xi_n \quad \forall \quad n \\ \xi_n \quad & \geq \quad 0 \end{aligned}$$

Lagrangian Function:

$$L(R, \boldsymbol{c}, \{\xi\}_n, \{\alpha\}_n, \{\lambda\}_n) = 0.5R^2 + c \sum_n \xi_n - \sum_n \lambda_n \xi_n + \sum_n \alpha_n [||\phi(x_n) - \boldsymbol{c}||_2^2 - R^2 - \xi_n] \quad (2.1)$$

Derivation:

$$\frac{\partial L}{\partial R} = R - 2R\sum_n \alpha_n = 0 \leftrightarrow 1 - 2\sum_n \alpha_n = 0 \leftrightarrow \sum_n \alpha_n = 0.5 \tag{2.2}$$

$\frac{\partial L}{\partial \boldsymbol{c}} = \sum_n \alpha_n[-2\phi(x_n) + 2\boldsymbol{c}] = 0 \leftrightarrow \sum_n \alpha_n \phi(x_n) = \boldsymbol{c}\sum_n \alpha_n$

Using 2.2,

$$\boldsymbol{c} = [\sum_n \alpha_n \phi(x_n)]/\sum_n \alpha_n \leftrightarrow \boldsymbol{c} = 2\sum_n \alpha_n \phi(x_n) \tag{2.3}$$

$$\frac{\partial L}{\partial \xi n} = c - \lambda_n - \alpha_n = 0 \quad \forall n \tag{2.4}$$

$$\begin{aligned} L(R, \boldsymbol{c}, \{\xi\}_n, \{\alpha\}_n, \{\lambda\}_n) &= R^2(0.5 - \sum_n \alpha_n) + \sum_n \xi_n(c - \alpha_n - \alpha_n) + \sum_n \alpha_n ||\phi(x_n) - \boldsymbol{c}||_2^2 \\ &= \sum_n \alpha_n k(x_n, x_n) - 4\sum_n\sum_m \alpha_n\alpha_m \phi(x_n)^T\phi(x_m) + 4\sum_n\sum_m \alpha_n\alpha_m\phi(x_n)^T\phi(x_m) \\ &= \sum_n \alpha_n k(x_n, x_n) - 2\sum_n\sum_m \alpha_n\alpha_m k(x_n, x_m) \end{aligned}$$

Dual:

$$\begin{aligned} max_{\{\alpha\}_n} \quad \sum_n \alpha_n k(x_m, x_n) - 2\sum_m\sum_n \alpha_n\alpha_m k(x_m, x_n) & \\ subject \quad to \quad \sum_n \alpha_n &= 0.5 \\ c - \alpha_n - \lambda_n &= 0 \quad \forall n \\ \alpha_n &\geq 0 \quad \forall n \\ \alpha_n &\geq 0 \quad \forall n \end{aligned}$$

As the objective function does not depend an α_n, thus we can convert the equality constraint involving λ_n with an inequality constraint $\alpha_n \leq c$

$$\begin{aligned} max_{\{\alpha\}_n} \quad & \sum_n \alpha_n k(x_m, x_n) - 2\sum_m \sum_n \alpha_n \alpha_m k(x_m, x_n) \\ & subject \quad to \quad \sum_n \alpha_n = 0.5 \\ & 0 \le \alpha_n \le c \quad \forall n \end{aligned}$$

Optimal Solutions:
$\boldsymbol{c} = 2\sum_n \alpha_n \phi(x_n)$
Points on the boundary, $\xi_n = 0$, support vectors $\alpha_n > 0$, by complementary slackness (KKT conditions):
$\alpha_n[||\phi(x_n) - \boldsymbol{c}||_2^2 - R^2 - \xi_n] = 0$. Thus for a point on the boundary $x_l : ||\phi(x_l) - \boldsymbol{c}||_2^2 = R^2 \leftrightarrow R^2 = k(x_l, x_l) - 4\sum_n \alpha_n k(x_l, x_n) + 4\sum_n \sum_m \alpha_n \alpha_m k(x_m, x_n)$
Decision Given x_t and we need to check if it is normal, check $||\phi(x_n) - \boldsymbol{c}||_2^2 < R^2$ then it is normal.
check $||\phi(x_n) - \boldsymbol{c}||_2^2 > R^2$ then it is not normal.
$k(x_t, x_t) - 4\sum_n \alpha_n k(x_t, x_n) + 4\sum_n \sum_m \alpha_n \alpha_m k(x_m, x_n) - (k(x_l, x_l) - 4\sum_n \alpha_n k(x_l, x_n) + 4\sum_n \sum_m \alpha_n \alpha_m k(x_m, x_n)) = k(x_t, x_t) - k(x_l, x_l) + 4\sum_n \alpha_n[k(x_l, x_n) - k(x_t, x_n)]$
If this is greater than 0 then x_t is not normal, otherwise it is normal.

3 Boosting

a) (Exponential Loss and Logistic Loss: Consider the following binary classification problem with exponential loss and log loss:

$$L_e(f) = E_{x,y}[e^{-yf(x)}] = \frac{1}{N}\sum_{n=1}^{N} e^{-y_i f(x_i)} \tag{3.5}$$

$$L_\sigma(f) = E_{x,y}\log[1 + e^{-2yf(x)}] = \frac{1}{N}\sum_{n=1}^{N} \log[1 + e^{-2y_i f(x_i)})] \tag{3.6}$$

where $y \in \{-1, +1\}$ is a binary class label.

Show that for both loss functions, the conditional distribution of y is:
$p(y|x) = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}}$

Exponential Loss: $E_{x,y}[e^{-yf(x)}|x] = p(y=1|x)e^{-f(x)} + p(y=-1|x)e^{f(x)}$

$\frac{\partial E(e^{-yf(x)}|x)}{\partial f(x)} = -p(y=1|x)e^{-f(x)} + p(y=-1|x)e^{f(x)} = 0 \leftrightarrow e^{2f(x)} = \frac{p(y=1|x)}{p(y=-1|x)}$

$\leftrightarrow f^*(x) = 0.5\log\frac{p(y=1|x)}{p(y=-1|x)}$

$e^{2f^*(x)} = \frac{p(y=1|x)}{p(y=-1|x)} \leftrightarrow e^{2f^*(x)} = \frac{p(y=1|x)}{1-p(y=1|x)}$

$\leftrightarrow e^{2f^*(x)} - e^{2f^*(x)}p(y|x) - p(y|x) = 0$

$\leftrightarrow p(y=1|x) = p(y|x) = \frac{e^{2f^*(x)}}{1+e^{2f^*(x)}} = \frac{e^{f^*(x)}}{e^{f^*(x)}+e^{-f^*(x)}}$

Log loss: $E(\log[1+e^{-2yf(x)}]|x) = p(y=1|x)\log[1+e^{-2f(x)}] + p(y=-1|x)\log[1+e^{2f(x)}]$

$\frac{\partial E(\log[1+e^{-2yf(x)}]|x)}{\partial f(x)} = \frac{-2p(y=1|x)e^{-2f(x)}}{1+e^{-2f(x)}} + \frac{2p(y=-1|x)e^{2f(x)}}{1+e^{2f(x)}} = 0$

$\leftrightarrow \frac{-2p(y=1|x)}{1+e^{2f(x)}} + \frac{2p(y=-1|x)e^{2f(x)}}{1+e^{2f(x)}} = 0$

$\frac{-2p(y=1|x)+2(1-p(y=1|x))e^{2f(x)}}{1+e^{2f(x)}} = 0$
$\leftrightarrow 2e^{2f(x)} - p(y=1|x)[2+2e^{2f(x)}] = 0$

$p(y=1|x) = p(y|x) = \frac{e^{2f^*(x)}}{1+e^{2f^*(x)}} = \frac{e^{f^*(x)}}{e^{f^*(x)}+e^{-f^*(x)}}$

Show that the log loss can be considered as negative log-likelihood, while the exponential loss cannot.

Let $y^* = (y+1)/2$, then $y^* = 1$ if $y = 1$ and $y^* = 0$ if $y = -1$

$$\begin{aligned}
l(y^*, p) &= y^* \log(p) + (1-y^*) log(1-p) \\
&= \log p^{y^*} + \log \frac{1-p}{(1-p)^{y^*}} \\
&= \log (\frac{p}{1-p})^{y^*} (1-p) \\
&= \log \frac{\exp^{2f(x)y^*} \exp^{-(f(x)}}{\exp^{-f(x)} + \exp^{f(x)}} \\
&= \log \frac{\exp^{f(x)y}}{\exp^{f(x)y} + \exp^{-f(x)y}} \\
&= \log \frac{1}{1+\exp^{-2f(x)y}} \\
&= -\log(1+\exp^{-2f(x)y})
\end{aligned}$$

Thus maximizing $l(y^*, p)$ is equivalent to minimizing $\log(1+\exp^{-2f(x)y})$ which is the log loss function.
Thus starting the conditional probability we can get the log loss function and not the exponential loss.

4 Bias-Variance Trade-off

a. Find the closed form solution for $\hat{\beta}_\lambda$ and its distribution.
$\frac{\partial}{\partial\beta}[\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda||\beta||^2] = \frac{\partial}{\partial\beta}[\frac{1}{n}(\boldsymbol{y} - X\beta)^T(\boldsymbol{y} - X\beta) + \lambda||\beta||^2] = -2X^T(\boldsymbol{y} - X\beta) + 2\lambda\beta = 0$
$\hat{\beta}_\lambda = (X^TX + \lambda T)^{-1}X^T\boldsymbol{y}$

Let $\hat{\beta}_\lambda \to \mathcal{N}(0, 1/2(\lambda)$, then joint likelihood of both training data and parameter:
$\log p(D,\beta) = \sum_{i=1}^{n} \log p(y_i|x_i,\beta) + \log p(\beta) = -\sum_n \frac{(y_i - x_i^T\beta)^2}{2\sigma^2} - \sum_d \frac{2\lambda}{2}\beta_d^2 + constant$
Using MAP estimate:
$\beta^{MAP} = argmax \log p(\beta, D)$
$\varepsilon(\beta) = \sum_n (y_i - x_i^T\beta)^2 + \lambda||\beta||^2$
Thus distribution of $\hat{\beta}_\lambda$ is $\mathcal{N}(0, 1/2(\lambda)$

b. Calculate the bias term $E[\boldsymbol{x}^T\hat{\beta}_\lambda] - \boldsymbol{x}^T\beta^*$ as a function of λ and some feature vector x
Using the model $y = X\beta$, and using the optimal $\hat{\beta}_\lambda$, we get:
$\hat{\beta}_\lambda\boldsymbol{y} = (X^TX + \lambda I)^{-1}X^TX\beta = (X^TX+\lambda I)^{-1}(X^TX + \lambda I - \lambda I)\beta = [I - \lambda(X^TX + \lambda I)^{-1}]\beta.$
Thus the bias term $= -\lambda W\beta$, where $W = (X^TX + \lambda I)^{-1}$

c. Calculate the variance term $E[(\boldsymbol{x}^T\hat{\beta}_\lambda - E[\boldsymbol{x}^T\hat{\beta}_\lambda])^2]$ as a function of λ and some feature vector x
$Var = \sigma^2(X^TX+\lambda I)^{-1}(X^TX)(X^TX+\lambda I)^{-1} = \sigma^2 WX^TXW$

d. Use the results from parts (b) and (c) and the bias–variance theorem to analyze the impact of λ in the squared error. (i.e. which term dominates when λ is small or large?)
$tr(Var(\beta^*)) = \sigma^2\sum_{j=1}^{p}\frac{1}{d_j^2}$, while $tr(Var(\hat{\beta}_\lambda)) = \sigma^2\sum_{j=1}^{p}\frac{d_j^2}{(d_j+\lambda)^2}$
Thus $tr(Var(\beta^*)) \geq tr(Var(\hat{\beta}_\lambda))$, thus we have a reduction in the variance as lambda increases.
$E(\hat{\beta}_\lambda|x) = |\beta - \lambda W\beta|$ which increases as λ increases.
So for large λ, the variance is small, but we have a high bias, and when λ is small, variance is high and we have a low bias.

5 Support Vector Machines

5.1 Data preprocessing

We started by processing the given data to scale them into range $[0, 1]$ for every feature. We also modified provided label vector, by changing the zeros to -1, since SVM requires having a two classes $y \in \{-1, 1\}$.

5.2 Implement linear SVM

We implemented the two functions trainsvm.m and testsvm.m.

5.3 Cross validation for linear SVM

a. Report the 3-fold cross-valiation accuracy (averaged accuracy over each validation set) and average training time (averaged over each training subset) on different C taken from $4^{-6}, 4^{-5}, \ldots, 4, 4^2$. How does the value of C affect the cross validation accuracy and average training time? Explain your observation.

By increasing the variable C, we are increasing the penalty given to wrong classifications; thus reducing the number of midclassified data samples, i.e. increasing the accuracy of training data.

Table 1: *3-fold cross-validation Average Accuracy and Average Time for different C values (linear SVM model)*

C	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}
Avg. Accuracy	0.531	0.531	0.531	0.531	0.533	0.576	0.608	0.652	0.688
Avg. Time	0.0491	0.0628	0.0730	0.0766	0.0690	0.0592	0.0587	0.0659	0.0923

We also notice from the table that by increasing C, the average time tends to increase.

b. The C chosen according to the resulting data is $C = 4^2 = 16$.

c. We developed the script *ownlinear.m* that uses *trainsvm.m* and *testsvm.m* to train the linear SVM with the selected C and compute the corresponding testing accuracy. We got the following: Testing Accuracy = 0.7311

5.4 Use linear SVM in LIBSVM

a. Is the cross validation accuracy the same as that in 5.3? Note that LIBSVM solves linear SVM in dual form while your implementation does it in primal form.

Table 2: *3-fold cross-validation Average Accuracy and Average Time for different C values (linear LIBSVM SVM model)*

C	4^{-6}	4^{-5}	4^{-4}	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}
Avg. Accuracy	0.531	0.531	0.531	0.531	0.534	0.590	0.607	0.644	0.675
Avg. Time	0.0193	0.0154	0.0148	0.0145	0.0153	0.0148	0.0147	0.0143	0.0154

Note that the average accuracy values are very similar to the values we got from solving the primal problem. This is expected as optimality requires similar dual and primal objective values.

b. How faster is LIBSVM compared to your implementation, in terms of training?
LIBSM is much faster than the function we wrote. It is sometimes 2 times faster and at some other C values we can see it 7 times faster. This is because solving the dual is much faster than solving the primal (need to only check for supporting vectors.

5.5 Use kernel SVM in LIBSVM

Polynomial kernel

Table 3: *3-fold cross-validation Average Accuracy (polynomial LIBSVM SVM model)*

Degree-C	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
1	0.531	0.531	0.531	0.551	0.586	0.608	0.650	0.671	0.694	0.717	0.720
2	0.531	0.531	0.531	0.531	0.567	0.585	0.609	0.651	0.706	0.728	0.744
3	0.531	0.531	0.531	0.531	0.531	0.568	0.592	0.607	0.651	0.711	0.723

Table 4: *3-fold cross-validation Average Time (polynomial LIBSVM SVM model)*

Degree-C	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
1	0.0159	0.0153	0.0149	0.0150	0.0148	0.0147	0.0145	0.0159	0.0220	0.0364	0.0989
2	0.0175	0.0164	0.0157	0.0153	0.0158	0.0154	0.0144	0.0145	0.0157	0.0175	0.0371
3	0.0162	0.0186	0.0152	0.0152	0.0159	0.0156	0.0151	0.0168	0.0150	0.0147	0.0194

RBF kernel

Table 5: *3-fold cross-validation Average Accuracy (RBF LIBSVM SVM model)*

Gamma-C	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
4^{-7}	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5729	0.5977	0.6276
4^{-6}	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5729	0.5977	0.6263	0.6589
4^{-5}	0.5313	0.5313	0.5313	0.5313	0.5313	0.5313	0.5729	0.5977	0.6250	0.6589	0.6888
4^{-4}	0.5313	0.5313	0.5313	0.5313	0.5313	0.5729	0.6016	0.6237	0.6589	0.6888	0.7174
4^{-3}	0.5313	0.5313	0.5313	0.5313	0.5742	0.5990	0.6237	0.6628	0.6979	0.7266	0.7305
4^{-2}	0.5313	0.5313	0.5313	0.5781	0.5859	0.6198	0.6732	0.7031	0.7409	0.7422	0.7383
4^{-1}	0.5313	0.5313	0.5729	0.5833	0.6250	0.6680	0.7188	0.7383	0.7383	0.7292	0.7109
4^{-0}	0.5313	0.5456	0.5742	0.6120	0.6706	0.7031	0.7044	0.7057	0.7096	0.7057	0.6888
4^{1}	0.5313	0.5599	0.5977	0.6432	0.6654	0.6914	0.6706	0.6523	0.6445	0.6380	0.6576
4^{2}	0.5313	0.5313	0.6237	0.6354	0.6589	0.6393	0.6315	0.6250	0.6263	0.6263	0.6263

Based on the cross validation results of Polynomial and RBF kernel, which kernel type and kernel parameters will you choose? You need to write a script libsvm.m using the best kernel and the parameters selected from cross-validation. Your script should train a SVM using libsvm with selected kernel and parameters and output the accuracy on the test set.

Table 6: *3-fold cross-validation Average Time(RBF LIBSVM SVM model)*

Gamma-C	4^{-3}	4^{-2}	4^{-1}	4^{0}	4^{1}	4^{2}	4^{3}	4^{4}	4^{5}	4^{6}	4^{7}
4^{-7}	0.0225	0.0214	0.0211	0.0218	0.0214	0.0219	0.0214	0.0211	0.0210	0.0204	0.0204
4^{-6}	0.0218	0.0220	0.0214	0.0212	0.0214	0.0217	0.0215	0.0210	0.0204	0.0206	0.0204
4^{-5}	0.0212	0.0214	0.0223	0.0212	0.0219	0.0214	0.0214	0.0204	0.0200	0.0201	0.0226
4^{-4}	0.0212	0.0212	0.0218	0.0211	0.0217	0.0218	0.0208	0.0198	0.0195	0.0207	0.0282
4^{-3}	0.0211	0.0222	0.0219	0.0210	0.0217	0.0208	0.0207	0.0192	0.0201	0.0257	0.0511
4^{-2}	0.0215	0.0215	0.0246	0.0212	0.0207	0.0201	0.0191	0.0193	0.0238	0.0396	0.0944
4^{-1}	0.0222	0.0214	0.0231	0.0207	0.0204	0.0195	0.0187	0.0222	0.0403	0.1022	0.2547
4^{-0}	0.0219	0.0219	0.0210	0.0204	0.0201	0.0191	0.0209	0.0319	0.0611	0.1935	0.6162
4^{1}	0.0219	0.0222	0.0207	0.0195	0.0197	0.0209	0.0272	0.0489	0.0835	0.1370	0.1930
4^{2}	0.0218	0.0218	0.0212	0.0210	0.0220	0.0257	0.0317	0.0342	0.0350	0.0358	0.0356

From the tables we can see that the highest accuracy occurs for polynomial type kernel, degree 2, $C = 4^7$, for these parameters we wrote a script that trains the svm model and computes the testing accuracy. We got the following result:
Testing Accuracy= 0.7833.

6 Bias/Variance Trade-off

a. We generated 100 data samples each having 10 data points, then for $g1$ to $g6$ we computed the bias squared and variance, shown in the table below:

Table 7: *$Bias^2$ and Variance for different g functions, sample size 10)*

function	$Bias^2$	$Variance$
g_1	0.4650	0
g_2	0.3502	0.0431
g_3	0.3536	0.1243
g_4	0.0007	0.0295
g_5	0.0030	0.0399
g_6	0.0582	0.0991

The 6 histogram plots are shown in Figure 1:

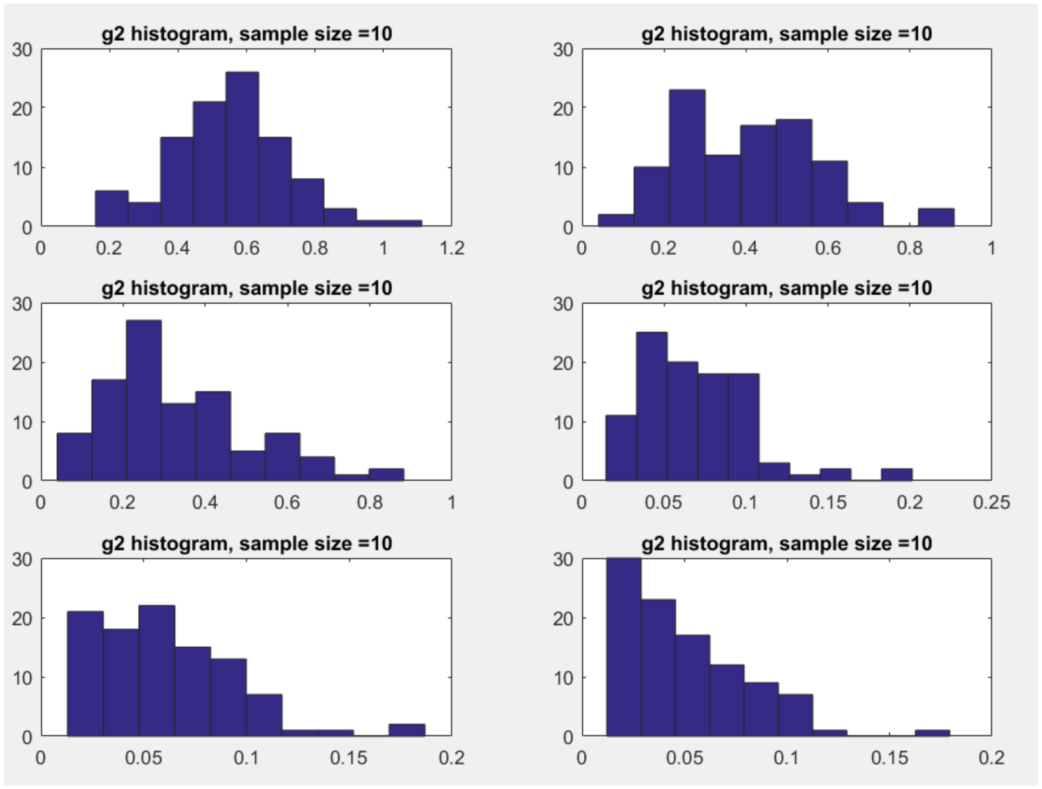

Figure 1: *Histogram plots for the 6 functions, sample size=10*

b. We generated 100 data samples each having 100 data points, then for $g1$ to $g6$ we computed the bias squared and variance, shown in Table 7

The 6 histogram plots are shown in the Figure 2

c. Given your results in (a) and (b), discuss how the model complexity and sample size affect the $bias^2$ and variance.

From Bias-Variance trade-off, we know that as the model complexity increases, the $Bias^2$ decrease and *variance* increase. From Table 1 we can see the $bias^2$ tend to decrease between g_1 and g_4. It reaches a very small value with $g4$ which is a second order polynomial. This is expected since $f(x)$ is also a second order polynomial. After g_4, the bias increases a bit but is still small. Note that we are over-fitting in these cases.
When increasing the number of data points in a data set, we are accounting for over-fitting (less

Table 8: *$Bias^2$ and Variance for different g functions, sample size 100)*

function	$Bias^2$	$Variance$
g_1	0.4671	0
g_2	0.3613	0.0047
g_3	0.3613	0.0111
g_4	0.0000	0.0026
g_5	0.0000	0.0036
g_6	0.0000	0.0045

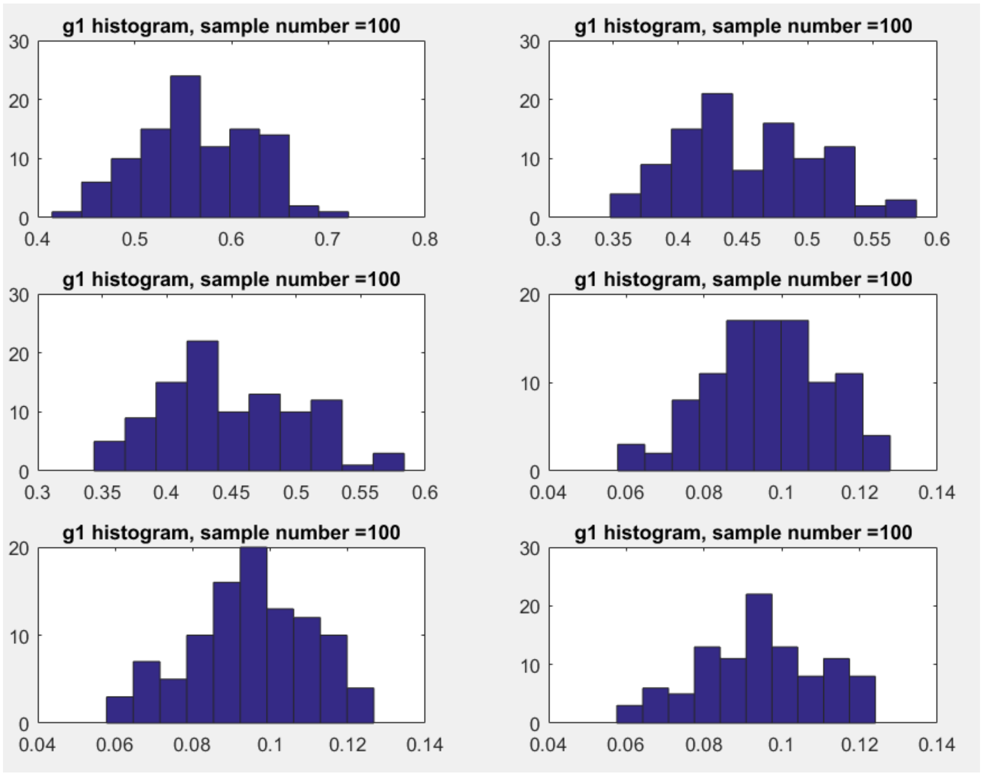

Figure 2: *Histogram plots for the 6 functions, sample size=100*

randomness). Thus, as seen in 8 we get 0 $Bias^2$ for g_4 to g_6 which represent polynomial of degrees $2, 3$ and 4. This is expected since the polynomial we used to generate the data has degree 2. Thus $g1$ to $g3$ result in high $bias^2$ and reduce to 0 when the degree is 2 or more.
In both cases, the variance increase with the increase of complexity between $g1$ and $g3$. Then at $g4$ the variance is minimal which corresponds to the polynomial of degree 2. Then the variance continue to increase after that between $g4$ and $g6$.

d.

Table 9: *$Bias^2$ and Variance for different g functions, sample size 100)*

Lambda	$Bias^2$	$Variance$
0.01	0.0000	0.0027
0.1	0.0001	0.0026
1	0.0033	0.0025
10	0.0919	0.0033
100	0.3588	0.0017

From Table 9, we can see that when lambda increases, $Bias^2$ increase and variance decrease. Note that when $lambda = 10$ the variance increased, so we computed the variance when $lambda = 100$ to make sure the variance will reduce further. The small increase might be due to randomness.