1) 

a) The number of degrees of freedom is six and that is the VC dimension. You can use function counting lemma to get growth number. $\Pi(5) = 32$, $\Pi(6) = 64$, $\Pi(7) = 126$, and $\Pi(8) = 240$.

b) Adaboost is an implementation of the boosting algorithm done by resampling the data. This is an ensemble learning method where each learner is presented the same set of training examples, however the distribution where training examples are chosen differs. The sampling method is chosen so that training examples that are difficult to classify are chosen with higher probability. Each learner or agent satisfies a weak learning condition with the ensemble aggregating decision satisfying strong learning conditions.

The generalization error improves even after the training error goes to 0 as we can view boosting as transforming inputs to a higher dimensional feature space. Like kernel methods boosting also attempts to maximize margins in feature space. The methodologies are different as kernel methods increase margins in higher dimensional feature spaces where as boosting uses base learning algorithm which explore the high dimensional feature space one dimension at a time.

c) Note that for any $x$, $x^T K x \geq 0$. If we set $x = e_i$ which is the all zero vector except a one at the $i$th component we have that $K(i,i) \geq 0$.

d) For Metropolis Hastings (MH) transitions from $x$ to $y$ are described by the probability transition $q(x,y)$ which are accepted with probability $\alpha(x,y) = \min(1, p(y)q(y,x)/(p(x)q(x,y)))$ with $p(x)$ being the probability distribution that we are attempting to simulate. The probability of going from $x$ to $y$ is then given by $q(x,y)\alpha(x,y)$.

The detailed balance equations are satisfied if

$$p(x)q(x,y)\alpha(x,y) = p(y)q(y,x)\alpha(y,x)$$

If $p(y)q(y,x) = p(x)q(x,y)$ then $\alpha(x,y) = \alpha(y,x) = 1$ and detailed balance equations are satisfied.

If $p(y)q(y,x) > p(x)q(x,y)$ then $\alpha(x,y) = 1$ and $\alpha(y,x) = p(x)q(x,y)/(p(y)q(y,x))$. By substitution detailed balance equations are satisfied.

If $p(x)q(x,y) > p(y)q(y,x)$ then $\alpha(y,x) = 1$ and $\alpha(x,y) = p(y)q(y,x)/(p(x)q(x,y))$. By substitution detailed balance equations are again satisfied.
2)

a) If we train the PLA sequentially starting with an initial weight vector of 0, then we end up going through 19 training examples before the algorithm terminates. The last update of weights is at time 14 with \( w(14)^T = [0, -3, 7] \) and \( b(14) = -2 \). The updates occur at times 1, 2, 4, 5, 6, 8, 9, 13, 14.

b) The optimum margin weight classifier is \( w^T = [-.4706, -.1176, 1] \) and \( b = 1.2941 \). The margin is \( 1/||w|| = .8997 \) and the optimum value of the objective function is .6176. The support vectors are \( x(2), x(3), \) and \( x(4) \).

c) Use the LS SVM algorithm with positive points labeled 2.5 and negative points labeled -1.667. The problem can either be solved in primal or dual space. This gives \( w^T = [-.6042, -.6667, 1.2917] \) and \( b = -.8542 \). This results in a MSE of 1.3176.

3) Matlab code is given below. Found good value of \( C = 2 \) and \( \sigma = .77 \) resulting in MSE of about .01.

```matlab
iter = 100;
n=10;
msetab= zeros(10,10);
tic
for kk=1:n;
    gamma = exp(-.7+kk*.2);
    for j=1:n;
        sigma = exp(-.6+j*.05);

        mse = 0;
        for i=1:iter;

            train = 10*rand(100,1)-5;
            dtrain=sinc(train) +.3*randn(100,1);
            dis=train*ones(1,100)-ones(100,1)*train';
            k=exp(-dis.*dis/(2*sigma^2));
            A=[0 ones(1,100);ones(100,1) k+eye(100)/gamma];
            alpha= A\[0;dtrain];
            test=10*rand(200,1)-5;
            distest=test*ones(1,100)-ones(200,1)*train';
            ktest=exp(-distest.*distest/(2*sigma^2));
            error =sinc(test)-ktest*alpha(2:101)-alpha(1);
```
4) The posterior density is Gaussian. To find conditional pdf work with product of likelihood pdf and prior pdf. Note that the exponent is a quadratic function of \(d\). By completing square we can compute conditional mean and conditional variance. The estimate is given by the conditional mean and the mean squared error is the conditional variance.

\[
E(D|X(1), \ldots, X(m)) = \frac{\sigma_D^2}{\sigma_D^2 + \sigma_X^2/m}(1/m) \sum_{i=1}^{m} X(i) + \frac{\sigma_X^2/m}{\sigma_D^2 + \sigma_X^2/m} \mu
\]

and

\[
\text{VAR}(D|X(1), \ldots, X(m)) = \frac{\sigma_D^2 \sigma_X^2/m}{\sigma_D^2 + \sigma_X^2/m}
\]

5) Note that \(\frac{dy(s)}{ds} = \beta(1 - y(s))(1 + y(s))\).

a) MSE cost function is \(J_{LMS}(w) = 0.5 \sum_{k=1}^{m} (e(k))^2\) where \(e(k) = d(k) - y(k)\). LMS is an online approximate gradient algorithm using most recent example to estimate gradient and results in

\[
w(k+1) = w(k) + \mu \beta e(k) (1 - y(k))(1 + y(k)) x(k).
\]

b ) Take derivative of cost function and using most recent example to approximate gradient and this results in

\[
w(k+1) = w(k) + \mu \beta e(k) x(k).
\]

When output node is saturated, magnitude of synaptic strength, \(s\) is large, then derivative of activation is close to zero and there is relatively little changes in weight updates. The entropy based learning rule learns better in these flat saturated regions as updates do not depend on \((1 - y(k))(1 + y(k))\).