1 Question 1

1. a) B

2. b)Approximation error is defined as the risk of the best predictive function in our hypothesis space, minus the risk of the bayes predictor: $R(F) - R(f^*)$.

   Since we expand our Hypothesis Space, when we use predicting functions that include $10,000$ words, ($F_{10,000}$) the distance that defines approximation error either decreases or stays the same. Conversely, when using only 100 features (F_{100}) the hypothesis space is smaller, and therefore the approximation risk is either greater or the same than $F_{10,000}$, therefore:

   $$Approximation\ Error(F_{100}) \geq Approximation\ Error(F_{10,000})$$

   The equal case should be considered, take the hypothetical situation where only spam emails are written in Chinese. Your bayes predictor function would operate using chinese characters as features, and therefore changing the amount of english features added wouldnt decrease approximation error.

3. c) Bayes Error stays the same regardless of which amount of features we use. Bayes Error is the expected Risk of the Bayes Predictor, which is a theoretical optimal function/model that would predict whether or not emails were spam. Since we're only tweaking our empirical models within our hypothesis space, this theoretical optimal function/model is not changed.

4. d) Approximation Error would remain unchanged, as it is not affected by the amount of observations we have in our dataset. Estimation Error however, is a random variable defined as $R(\hat{f}) - R(F)$. Since the data we use is sampled iid, we cannot determine if the estimation error would increase, decrease, or stay the same.

5. e) Approximation is not a random variable. This is because once we select our Hypothesis Space, F, the set of all functions we are considering is fixed, and the there is a deterministic relationship between the best function in our hypothesis space and the hypothesis space we have selected.

   Estimation Error, on the other hand, is a random variable. This is because the estimation error represents the loss we incur for not having all possible datapoints.

   We expect Estimation Error to decrease as we collect n datapoints with n approaching infinity, but because we have limited samples, and we do not necessarily know the true underlying distribution of how the data is generated, there is noise inherent to the randomness in our sampling. Therefore, the estimation error we could theoretically calculate on our sample is a random variable, which has variance, and thus fluctuates depending on our sampling. It is a random variable since its value is defined by our RANDOM sampling method.

6. f) In this case i would be a hyper parameter, and we would like to find the best hyper parameter i such that our model performs the best on new data.

   When dealing with hyper parameters, its best practice to use a training set, a validation set, and a test set. We could even add a regularization term, like LASSO regularization, to our model that would help us identify which of the i features help us with our predictive power

if an email is spam or not spam. This would help 0 out features that do not contribute, and show the optimal weights for features that are informative.

We would want to fit our model with i parameters on our training set, and by using regularization, examine which model performs best on our validation set. With the help of our validation set, we would find the model that performed the best, and we would use the i features of the model that performed the best. That is how we would find our i features.

Lastly, to be thorough, we would then test if it generalizes well on unseen data, using our test set, which would give us an approximation of how it would perform on real world data.

2 Question 2

1. a) Update rule for gradient descent:

$$w_t = w_{t-1} - \eta \times \nabla_w f(w_{t-1})$$

2. b) This procedure is not garuanteed to converge. When using gradient descent, it is paramount to choose the right step size. If a step size is too big, say $\eta = 1$, it is likely that the model bounces around the minimum and never converges. (In this case think of gradient descent on $f(x) = x^2$ at point $x = -2$).

   Additionally, if our step size is too large, our gradient descent algorithm might diverge towards infinity, with w bouncing around in ever increasing fashion. If we set our gradient step correctly, we can expect a convergence, but maybe not necessarily to the global minimum, as the w vector could get trapped in a local minimum, and never reach w^*

3. c) Yes we would expect this to converge, as we would find the w that minimizes our loss function, however if we add a constriant that defines w^* as the w that minimizes our loss function within a certain value for w like $||w||_2 \leq 10$ then the w we find might not be w^*.

4. d)SDG update rule with only one point:

$$w_t = w_{t-1} - \eta \times \nabla_w L_i(w)$$

5. e) Full Batch gradient descent is the most computationally expensive, and therefore the slowest, but will tend to converge to a lower local minimum with less noise in each step (much smoother).

   SGD with a single data point is the fastest to compute, though it is the most noisy, with lots of variance in each step and it is not that smooth. It will however converge faster than full Batch gradient descent.

   Mini-Batch methods meet in the middle of SGD and Full Batch, using a fixed n datapoints to calculate the gradient. They have a mix of being slower than SGD, but converging to a lower minimum than SGD (but higher than full batch), while being faster than Full batch but slower than SGD.

   Our tradeoff comes down to speed (time and compute) vs accuracy (convergence to global minimum and to the paramater w that corresponds to lowest loss)

3 Question 3

1. a) Algorithm A would be more selective and end up with less features, algorithm B will end up with more features than A. This is because Algorithm B is using training error to select its features, and when you add features to linear regression your training error can never go down (if it doesn't help explain the variance of the data then its coefficient will be 0). Validation error though forces the model to be picky with its features, as it needs to select the ones only that explain the variance of the data.
2. b) False
3. c) True
4. d) False
5. e) True
6. f) True
7. g) True
8. h) Plot *a* corresponds to LASSO regression, which uses the L1 Norm, while plot *b* corresponds to Ridge regression which uses the L2 norm.

   Lasso regression is effective at 0'ing out features, while the L2 norm used in Ridge tends to decrease features in a proportionate matter but tends not to 0 them out fully.

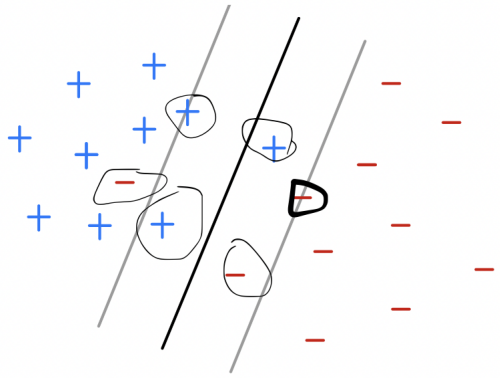
   When dealing with two highly correlated features, RIDGE regression preferring an equal weighting between two correlated features. As the L1 norm looks like a diamond when graphed in 2-d, this form of regularization does a better job at 0ing out correlated features, preferring to take the one with higher magnitude.

4 Question 4

1. a) A (convex)
2. b) B (concave)
3. c) A
4. d)

   i) B

   ii) All of the points between the gray lines or on the gray lines (add screenshot).



5. e) Increasing C leads to:
   i) SMALLER GEOMETRIC MARGIN
   ii) FEWER SUPPORT VECTORS

5 Question 5

1. a) This follows from the definition of w as it is initialized to 0, then given in the update rule: for t iterations we would do the following step whenever $y_i(w^T x_i) \leq 0$: $w = w + x_i y_i$, where $y_i \in [-1, 1]$

   This produces w, which is a linear combination of the x_i data-points in our data matrix, X (Note: Linear Combination means a summation of the x_i terms scaled by their respective α_i).

   $$w = \sum_{i=1}^{n} \alpha_i \times x_i = X^T \alpha$$

   Since w is a linear combination of our data, it is therefore in the span of our data.

2. b) A prediction is calculated using $w^T x$. Therefore we can show:

   $$w^T x = x^T (X^T \alpha) = \sum_{i=1}^{n} \langle x, X_i \rangle \times \alpha_i$$

3. c) The update rule for α_i is as follows:

   $$\alpha_{it+1} = \alpha_{it} + y_i$$

4. d) The faster algorithm would be the SVM as SVM identifies the support vectors necessary to classify and predict observations. The support vectors n are a subset of the total amount of observations m. Perceptorns however need to go through all of the m data points and calculate m^2 dot products, while SVM would only need to compute n^2 dot products.