CSE 546 Machine Learning, Autumn 2013
Homework 4

Shrainik Jain, 1323338

1 Learning Theory [30 points]

1. For
$$h : \{0,1\}^d \rightarrow \{0,1\}$$
the feature space is of dimension d. Since the features are binary, they can take 2 values, i.e. 0 or 1. This means that there are 2^d feature vectors in feature space. For each of these feature vector the output can take the value 0 or 1, i.e. 2 values possible per feature vector. Hence, total number of functions which map feature vector space to output space
$$|H| = 2^{2^d}$$

2. Chernoff bound to estimate error of union over a total of $|H|$ hypotheses is:
$$P(error_{true}(h) - error_{train}(h) > \epsilon) \leq |H| \exp^{-2N\epsilon^2} \tag{1}$$
for this problem, we want this to be bounded by δ:
$\Rightarrow |H| \exp^{-2N\epsilon^2} \leq \delta$
This gives,
$$N \geq \frac{ln(\frac{|H|}{\delta})}{2\epsilon^2} \Rightarrow N \geq \frac{ln(\frac{2^{2^d}}{\delta})}{2\epsilon^2} \tag{2}$$

3. From part 2, we see that the bound on N is $O(ln(2^{2^d})) = O(2^d)$ which means that the number of points needed to be sure of having generalization error less than ϵ with a high probability $1 - \delta$ increases exponentially with the incerease in the number of features. In practice, exponential number of data points are not available, hence this bound is not very useful.

4. From the lecture slides on learning theory, we know that the upper bound on number of decision trees of depth k ($|H|$) is $2^{(2^k-1)(1+log_2 d)+1}$. (Slide 34, learning theory.)
Which becomes $2^{2^k} d^{2^k-1}$ and $k = 2 \Rightarrow |H| = 16d^3$

   Plugging in $|H|$ to equation 1 above, we get,
$$N \geq \frac{ln(\frac{16d^3}{\delta})}{2\epsilon^2} \tag{3}$$

5. The bound we got earlier (in part 2) was $O(2^d)$, where as the newer bound is $O(ln(16d^3)) = O(ln(d))$. This means that by restricting the structure of the problem, i.e. by restricting the hypothesis space to have only decision trees of depth 2, we require O(ln(d)) number of data points to get the similar PAC bound (instead of exponential number of points as in part 2 where there was no restriction on hypothesis space). Also, for a fixed depth, the number of data points required for the same PAC bound, are no longer exponential in dimension d. Which is a big big advantage.

2 PCA via Successive Deflation [35 points]

1. We have:

   $\tilde{\mathbf{C}} = \frac{1}{n}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T = \frac{1}{n}((\mathbf{I} - v_1 v_1^T)\mathbf{X}((\mathbf{I} - v_1 v_1^T)\mathbf{X})^T)$

   (using the fact that $(AB)^T = B^T A^T$ and $(\mathbf{I} - v_1 v_1^T)$ is symmetric)

   $$\tilde{\mathbf{C}} = \frac{1}{n}(\mathbf{I} - v_1 v_1^T)XX^T(\mathbf{I} - v_1 v_1^T) \tag{4}$$

   $$\tilde{\mathbf{C}} = \frac{1}{n}(XX^T - v_1 v_1^T XX^T - XX^T v_1 v_1^T + v_1 v_1^T XX^T v_1 v_1^T) \tag{5}$$

   We know that $XX^T v_1 = n\lambda_1 v_1 \Rightarrow (XX^T v_1)^T = (n\lambda_1 v_1)^T \Rightarrow v_1^T XX^T = n\lambda_1 v_1^T$
   Plugging into equation 5 we get,

   $$\tilde{\mathbf{C}} = \frac{1}{n}(XX^T - v_1 n\lambda_1 v_1^T - n\lambda_1 v_1 v_1^T + v_1 n\lambda_1 v_1^T v_1 v_1^T) \tag{6}$$

   Finally, since $v_1^T v_1 = 1$

   $$\tilde{\mathbf{C}} = \frac{1}{n}XX^T - \lambda_1 v_1 v_1^T \ \ ...Q.E.D. \tag{7}$$

2. We have: $\tilde{\mathbf{C}}v_j = (\frac{1}{n}XX^T - \lambda_1 v_1 v_1^T)v_j$

   $= \frac{1}{n}(XX^T v_j) - \lambda_1 v_1 v_1^T v_j$

   $= \lambda_j v_j - \lambda_1 v_1 v_1^T v_j$ (since $XX^T v_j = n\lambda_j v_j$)

   $\Rightarrow \tilde{\mathbf{C}}v_j = \lambda_j v_j$ (since $v_1^T v_j = 0$ for $j \neq 1$)
   and
   $\tilde{\mathbf{C}}v_1 = \lambda_1 v_1 - \lambda_1 v_1 v_1^T v_1 = (\lambda_1 - \lambda_1)v_1 = 0v_1 = 0$ for $j = 1$

   hence, for $j \neq 1$, v_j is also a principle eigenvector of $\tilde{\mathbf{C}}$ with same eigenvalue λ_j. **Also, v_1 is an eigenvector of $\tilde{\mathbf{C}}$ with eigenvalue 0.**

3. Since $v_1, v_2, \ldots v_k$ are the first k eigenvectors with largest eigenvalues of $\mathbf{C}$, i.e., the principal basis vectors, therefore

   $$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \tag{8}$$

   From part 2, we know that for $\tilde{\mathbf{C}}$, v_j are the principle eigenvectors with eigenvalues $(0, \lambda_2, \lambda_3, \ldots, \lambda_k)$. Therefore from equation 8 above, λ_2 is the largest eigenvalue of $\tilde{\mathbf{C}}$ (since λ_1 is not an eigenvalue of $\tilde{\mathbf{C}}$), hence v_2 is the first principle eigenvector.

4. Pseudocode for finding the first K principal eigenvectors of $\mathbf{C}$:

```
def findEigenVectors(C,K,f):
        list_Lambda = []
        list_v = []
        for i in range(K):
                lambda, v = f(C)
                C = C - lambda * v * v.Transpose
                list_Lambda.append(lambda)
                list_v.append(v)
        return list_v, list_Lambda
```

3 Programming Question (clustering with K-means) [35 points]

3.1 The Data

No Questions in this part.

3.2 The algorithm

No Questions in this part.

3.3 Within group sum of squares

No Questions in this part.

3.4 Mistake Rate

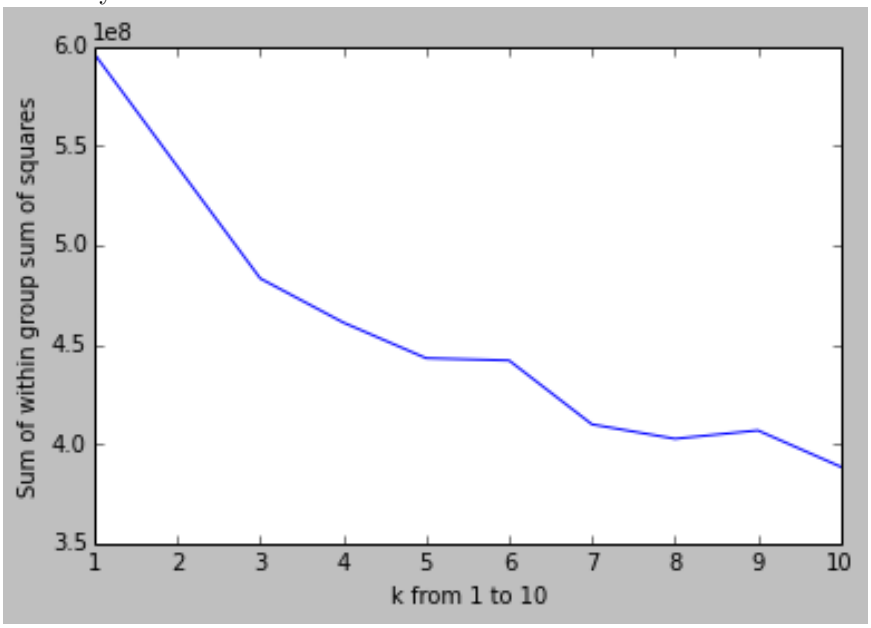
No Questions in this part.

3.5 Questions

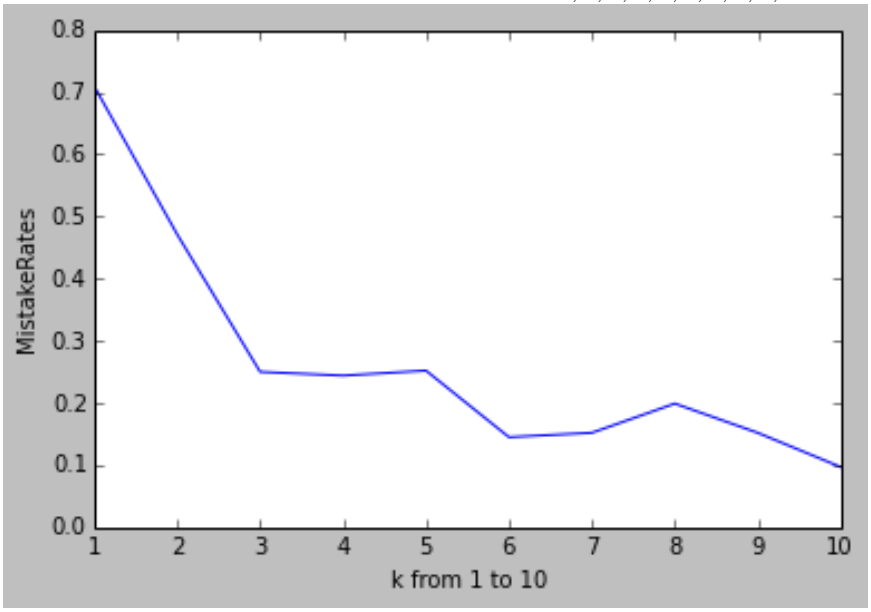
1. For $k = 2$
   - Sum of within group sum of squares = 536477102.543
   - Mistake Rate = 0.52

   For $k = 4$
   - Sum of within group sum of squares = 461110943.962
   - Mistake Rate = 0.243

   For $k = 6$
   - Sum of within group sum of squares = 431349182.916
   - Mistake Rate = 0.18
2. The number of iterations that k-means ran for $k = 6$: **8 iterations.**
3. Plot for the sum of within group sum of squares versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. Centers chosen randomly:



4. Plot for the total mistake rate versus k for $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$. Centers chosen randomly:



3.6 Some useful functions

No questions in this part.