Comp 6321 - Machine Learning - Assignment 4

Federico O'Reilly Regueiro

December 4^{th}, 2016

Question 1: VC dimensions

1.a $[a, \infty)$

We can shatter a single point p_1, $p_1 \in \mathbb{R}$.:

point	label	h
p_1	$\oplus$	$[a, \infty), a < p_1$
p_1	$\ominus$	$[a, \infty), a > p_1$

But if we have two points, $p_1, p_2 \mid p_1 < p_2, p_1 \in \oplus, p_2 \in \ominus$, then $[a, \infty)$ cannot shatter them. Therefore, for this class of hypothesis: $VC_{dim} = 1$

1.b $(-\infty, a]$ or $[a, \infty)$

Similarly to the previous question, we can shatter one point. Additionally, we can shatter two points, $p_0, p_1 \mid p_0 < p_1, p_0$:

point	label	h
p_1	$\ominus$	$(-\infty, a], a < p_1$
p_2	$\ominus$	
p_1	$\ominus$	$[a, \infty), p_1 < a < p_2$
p_2	$\oplus$	
p_1	$\oplus$	$(-\infty, a], p_1 < a < p_2$
p_2	$\ominus$	
p_1	$\oplus$	$[a, \infty), a < p_1$
p_2	$\oplus$	

However, three points $p_1, p_2, p_3, \mid p_1 < p_2 < p_3, p_1 \in \ominus, p_2 \in \oplus, p_3 \in \ominus$ cannot be shattered. Therefore, for this class of hypothesis: $VC_{dim} = 2$

1.c Finite unions of one-sided intervals

The union of more than one left-side interval $(-\infty, a] \cup (-\infty, b] \ldots \cup (-\infty, n]$ is equivalent to a single left-side interval $(-\infty, max(a, b, \ldots n)]$. The same applies for one or more right-side intervals being equivalent to $[min(a, b, \ldots n), \infty)$. Therefore, this hypothesis class is of the form $(-\infty, a] \cup [b, \infty)$.

Since $\{(-\infty, a]$ or $[b, \infty)\} \subset \{(-\infty, a] \cup [b, \infty)\}$, we know this class of hypothesis to be capable of shattering 2 points. But once again, three points $p_1, p_2, p_3, \mid p_1 < p_2 < p_3, p_1 \in \ominus, p_2 \in \oplus, p_3 \in \ominus$ cannot be shattered with this class of hypothesis. Therefore, for this class: $VC_{dim} = 2$

1.d $[a, b] \cup [c, d]$

This class of hypothesis can shatter four points due to the following:

a Any four positives can be correctly classified by a single interval as can any labeling with a single positive.

b Any two positives and two negatives can be classified with two intervals, given that a single interval is assigned to each positive.

c Labeling three positives and one negative will always yield at most two groups of contiguous positive labels, each of which can be contained in one of the two intervals.

However, if we have five points $p_1, p_2, p_3, p_4, p_5, | \; p_1 < p_2 < p_3 < p_4 < p_5, p_1 \in \oplus, p_2 \in \ominus, p_3 \in \oplus, p_4 \in \ominus, p_5 \in \oplus$ cannot be shattered with this class of hypothesis. Therefore, for this class: $VC_{dim} = 4$

1.e Unions of k intervals

We prove that it is $2k$ by induction:

Base step: One interval, $k = 1, h = [a, b]$, and two points, $p_1, p_2 \mid p_1 < p_2, p_1$:

point	label	h
p_1 p_2	$\ominus$ $\ominus$	$[a, b], b < p_1$
p_1 p_2	$\ominus$ $\oplus$	$[a, b], p_1 < a < p_2 < b$
p_1 p_2	$\oplus$ $\ominus$	$[a, b], a < p_1 < b < p_2$
p_1 p_2	$\oplus$ $\oplus$	$[a, b], a < p_1, p_2 < b$

We increase the set to three points with the following labels $p_1, p_2, p_3, | \; p_1 < p_2 < p_3, p_1 \in \oplus, p_2 \in \ominus, p_3 \in \oplus$, it cannot be shattered Therefore, for the base step $VC_{dim} = 2 = 2k$.

Now suppose that for the union of k intervals,VC dimension is $2k$[1], then we need to prove that with $k + 1$ intervals we are able to shatter $2(k + 1)$.

Firstly we note that the most *difficult* configuration to classify would be an alternation of $\oplus$ and $\ominus$ points, since it would require using each one of the k intervals to classify a single point each; any other configuration would require less than k intervals and we would have some *leftover* intervals to be consumed in classifying newly inserted points.

Inductive step: We add points p_{2k+1}, p_{2k+2}, with no inequality constraints, to the $2k$ points shattered with k intervals. Without loss of generality, we suppose the previous points to be in an alternating configuration of labels as we mentioned above. We can contemplate three possible scenarios for the added points:

i $p_{2k+1}, p_{2k+2} \in \ominus$

ii $p_{2k+1} \in \oplus, p_{2k+2} \in \ominus$, note[2]

iii $p_{2k+1}, p_{2k+2} \in \oplus$

[1] ie, we can shatter $2k$ points but not $(2k) + 1$ points.

[2] Equivalent to $p_{2k+2} \in \oplus, p_{2k+1} \in \ominus$

case i

Since the previous $2k$ points could be shattered and there are no two contiguous $\oplus$ labels in the previous set of $2k$ points, introducing two $\ominus$ labels anywhere will not disrupt prior labeling if the intervals capturing the adjacent $\oplus$ points are adjusted accordingly.

case ii

As above, the $\ominus$ point will not disrupt prior labeling. The $\oplus$ point will either fall beside another $\oplus$ point where it can be included in the interval[3] capturing the adjacent $\oplus$, or at either end of the set, besides an $\ominus$ point, in which case the $k+1^{th}$ interval will correctly classify it.

case iii

If the previous $2k$ points are labeled with alternating $\ominus$ and $\oplus$, then one end of the set will have $\ominus$ and the other $\oplus$. Thus on inserting points p_{2k+1} and p_{2k+2} one of them will necessarily fall beside another $\oplus$ and, in the worst case, the other point could be placed at the end of the interval on the end with the $\ominus$, in which case the $k+1^{th}$ interval would correctly classify it.

Thus $k+1$ intervals shatter $2(k+1)$ points. Conversely, with $2(k+1)+1$ points and the following configuration $\oplus, \ominus, \ldots \oplus$ we would not be able to shatter the set of points with $k+1$ intervals.

Thus the inductive step holds.

Then, for this class with k intervals, $VC_{dim} = 2k$.

[3] Once the bounds of said interval have been adjusted

Question 2: KL Divergence

2.a $KL(P||Q) \geq 0, \forall P, Q$

Since $log(x)$ is a concave function, in order to use Jensen's inequality as stated for convex functions, we make the expression convex by proving $-KL(P||Q) \leq 0, \forall P, Q$. We use $P(x_i) > 0, \forall P(x_i)$ for convenience.

$$
\begin{aligned}
-KL(P||Q) &= -\sum_{i=1}^{m} P(x_i) log\left(\frac{P(x_i)}{Q(x_i)}\right) \\
&= \sum_{i=1}^{m} P(x_i) log\left(\frac{Q(x_i)}{P(x_i)}\right)
\end{aligned}
$$

And by Jensen's inequality, taking $\frac{P(x_i)}{Q(x_i)}$ to be a random variable uniformly distributed over i, we can then write:

$$
\begin{aligned}
-KL(P||Q) &\leq log\left(\sum_{i=1}^{m} P(x_i)\frac{Q(x_i)}{P(x_i)}\right) \\
&\leq log\left(\sum_{i=1}^{m} Q(x_i)\right) \\
&\leq log\left(1\right) \\
&\leq 0
\end{aligned}
$$

2.b $KL(P||Q) = 0$?

When both distributions are equal, i.e. $P(x_i) = Q(x_i), \forall i$, $KL(P||Q)$ becomes:

$$
\begin{aligned}
KL(P||Q) &= -\sum_{i} P(x_i) log\left(\frac{Q(x_i)}{P(x_i)}\right) \\
&= -\sum_{i} P(x_i) log\left(1\right) \\
&= 0
\end{aligned}
$$

Which makes sense since the divergence of two equal distributions should be zero.

2.c Max $KL(P||Q)$?

$$
\begin{aligned}
KL(P||Q) &= \sum_{i} P(x_i) log\left(\frac{P(x_i)}{Q(x_i)}\right) \\
\lim_{Q(x_i)\to 0}(KL(P||Q)) &= \infty, \text{for some } i, |\ P(x_i) \not\to 0
\end{aligned}
$$

Which can be interpreted as the divergence between the true distribution $P(x)$ and the modeling distribution $Q(x)$ approaching infinite if $Q(x)$ cannot represent an event x_i with a non-zero probability in $P(x)$.

2.d $KL(P||Q) = KL(Q||P)$? Justify

No, suppose $x_i = \{0,1\}$ with $P(0) = \frac{1}{3}, P(1) = \frac{2}{3}$ modelled by $Q(0) = \frac{1}{2}, Q(1) = \frac{1}{2}$. Then:

$$\begin{aligned}
KL(P||Q) =& \frac{1}{3} log\left(\frac{\frac{1}{3}}{\frac{1}{2}}\right) + \frac{2}{3} log\left(\frac{\frac{2}{3}}{\frac{1}{2}}\right) \\
=& \frac{1}{3} log\left(\frac{2}{3}\right) + \frac{2}{3} log\left(\frac{4}{3}\right) \\
=& 0.056633 \\
KL(Q||P) =& \frac{1}{2} log\left(\frac{\frac{1}{2}}{\frac{1}{3}}\right) + \frac{1}{2} log\left(\frac{\frac{1}{2}}{\frac{2}{3}}\right) \\
=& \frac{1}{2} log\left(\frac{3}{2}\right) + \frac{1}{2} log\left(\frac{3}{4}\right) \\
=& 0.058891 \\
KL(P||Q) \neq& KL(Q||P)
\end{aligned}$$

2.e Prove $KL(P(Y,X)||Q(Y,X)) = KL(P(X)||Q(X)) + KL(P(Y|X)||Q(Y|X))$

$$\begin{aligned}
KL(P(Y,X)||Q(Y,X)) =& \sum_x \sum_y P(x,y) log\left(\frac{P(x,y)}{Q(x,y)}\right) \\
=& \sum_x \sum_y P(x,y) log\left(\frac{P(y|x)P(x)}{Q(y|x)Q(x)}\right) \\
=& \sum_x \sum_y P(x,y) \left(log\left(\frac{P(x)}{Q(x)}\right) + log\left(\frac{P(y|x)}{Q(y|x)}\right)\right) \\
=& \sum_x \sum_y P(x,y) log\left(\frac{P(x)}{Q(x)}\right) + P(x,y) log\left(\frac{P(y|x)}{Q(y|x)}\right) \\
=& \sum_x \sum_y P(x|y)P(y) log\left(\frac{P(x)}{Q(x)}\right) + P(y|x)P(x) log\left(\frac{P(y|x)}{Q(y|x)}\right) \\
=& \sum_x \sum_y \left[P(x|y)P(y) log\left(\frac{P(x)}{Q(x)}\right)\right] + \sum_y \left[P(y|x)P(x) log\left(\frac{P(y|x)}{Q(y|x)}\right)\right] \\
=& \sum_x log\left(\frac{P(x)}{Q(x)}\right) \sum_y [P(x|y)P(y)] + P(x) \sum_y \left[P(y|x) log\left(\frac{P(y|x)}{Q(y|x)}\right)\right] \\
=& \sum_x P(x) log\left(\frac{P(x)}{Q(x)}\right) + P(x) \sum_y P(y|x) log\left(\frac{P(y|x)}{Q(y|x)}\right) \\
=& KL(P(X)||Q(X)) + KL(P(Y|X)||Q(Y|X))
\end{aligned}$$

2.f Prove $\arg\min_\theta KL(\hat{P}||P) = \arg\max_\theta \sum_{i=1}^m logP_\theta(x_i)$

First, we develop the LHS and we note that:

$$\begin{aligned}
\arg\min_\theta KL(\hat{P}||P_\theta) =& \arg\min_\theta \sum_x \hat{P}(x) log\left(\frac{\hat{P}(x)}{P_\theta(x)}\right) \\
=& \arg\min_\theta \sum_x \hat{P}(x) log(\hat{P}(x)) - \sum_x \hat{P}(x) log(P_\theta(x))
\end{aligned}$$

And since $\sum_x \hat{P}(x)log(\hat{P}(x))$ depends solely on the observations and is fixed w.r.t. θ, we can then equivalently write:

$$\begin{aligned}
\arg\min_\theta KL(\hat{P}||P_\theta) &= \arg\min_\theta -\sum_x \hat{P}(x)log(P_\theta(x)) \\
&= \arg\max_\theta \sum_x \hat{P}(x)log(P_\theta(x))
\end{aligned} \tag{1}$$

Then, for a given set of m observations $x_i, x \in X, i \in \{1, 2, \ldots m\}$ we can have $n \leq m$ unique values, which we index as $x_j, x \in X, j \in \{1, 2, \ldots n\}$ to avoid confusion with observation indexing $i \in \{1, 2, \ldots m\}$. Thus we can substitute $\hat{P}(x_j) = \frac{|x_j|}{m}$ into equation 1:

$$\arg\min_\theta KL(\hat{P}||P_\theta) = \arg\max_\theta \sum_{j=1}^{n} \frac{|x_j|}{m} log(P_\theta(x_j)) \tag{2}$$

Now we develop the RHS:

$$\begin{aligned}
M.L.E(x_i, \theta) &= \arg\max_\theta \sum_{i=1}^{m} log(P_\theta(x_i)) \\
&= \arg\max_\theta \sum_{i=1}^{m} \frac{1}{m} log(P_\theta(x_i)) \\
&= \arg\max_\theta \sum_{j=1}^{n} \frac{|x_j|}{n} log(P_\theta(x_j))
\end{aligned}$$

Which is precisely equal to equation 2.

Question 3: Implementation: K-means

K-means clustering has been implemented in the `A4_Q3_driver.m` matlab script. The file is included as part of this submission.

We reproduce, at the end of the answer, the main par of code for the reader's convenience, omitting non-essential plotting and file manipulation. The reprint includes some useful comments regarding the procedure.

The script outputs some useful information both as simple text and in the form of plots. The text output is printed here and the plots can be seen in figures 1, 2 and 3.

The original and resulting images are also reprinted in figures 4a and 4b

```
There are 6 total clusters with pixels in them

Final cluster membership count is respectively:
    4930    15190    52535        0    22075        0    40365    74917

The final centroids are:
   241.22961   238.62515   233.86288
   194.41159   136.33311    90.94365
   136.26556    61.08973    10.10385
     0.00000   255.00000     0.00000
   157.29173    97.59398    51.43330
     0.00000     0.00000   255.00000
    78.92744    37.10829    13.07070
    25.97800    23.23575    23.60599
```

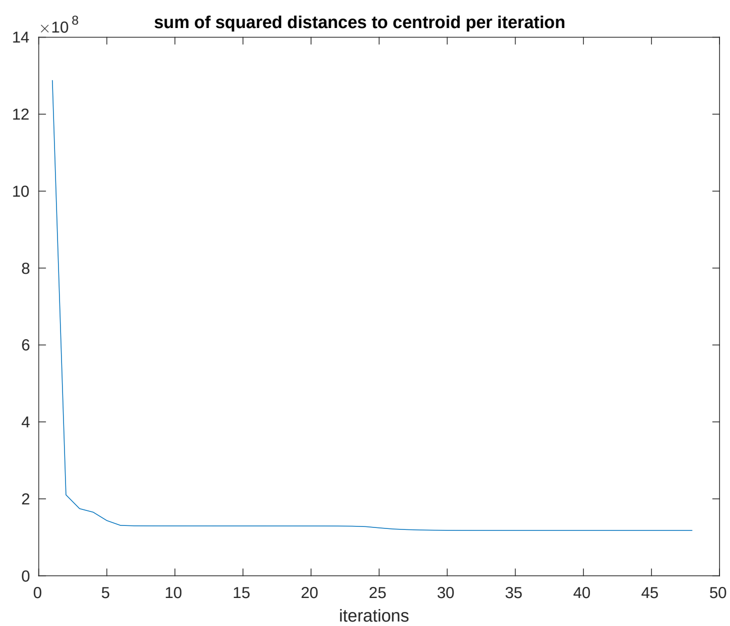

Figure 1: The sum of all squared distances from all pixels towards their respective centroids over iterations.

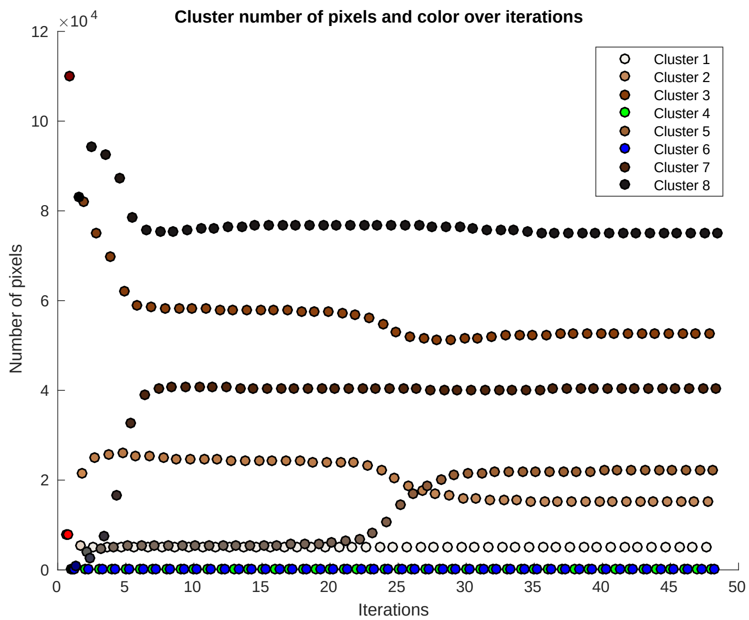

Figure 2: The number of pixels pertaining to each cluster per iteration. The marker colors represent the resulting color of centroids per iteration.

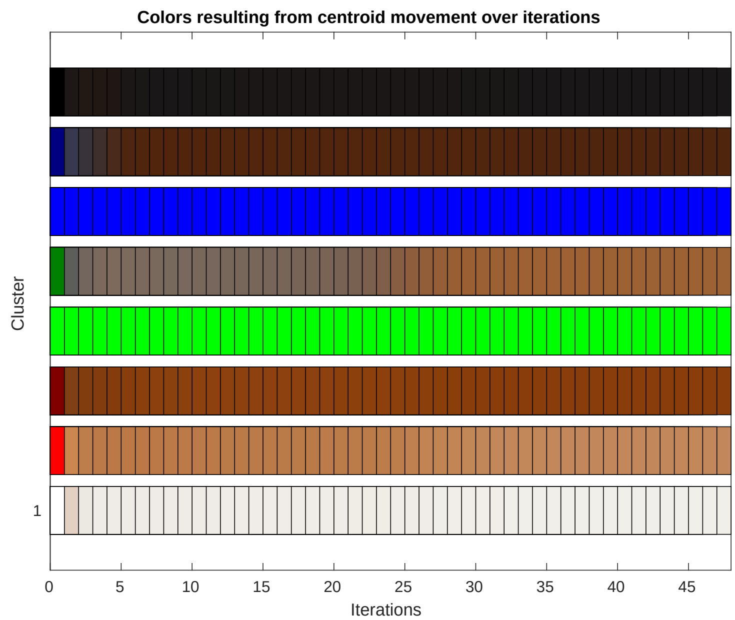

Figure 3: Centroid's resulting color per iteration. Most centroids shift rapidly during the first few iterations, only cluster 5 seems to evolve more slowly.

(a) original image - 256 colors

(b) image with 6 actual clusters for pixel color-value

Figure 4: The original image, left, and the clustered image to the right.

The k-means algorithm part of `A4_Q3_driver.m`:

```
init_matx = ...
[255, 255, 255; ...
255, 0, 0; ...
128, 0, 0; ...
0, 255, 0; ...
0, 128, 0; ...
0, 0, 255; ...
0, 0, 128; ...
0, 0, 0];

data = load('hw4-image.txt');

m = length(data(:,1));
d = length(data(1,:));
k = length(init_matx(:,1));

% guesstimated number of iterations for preallocation,
% This is mostly for efficiency as the loop will stop before if converged
ITERS = 50;

% Initial centroids or cluster means
k_means = repmat(reshape(init_matx', 1, d, k), m, 1, 1);
new_means = init_matx;
k_labels = 1:m;
k_settled = false;
% how many pixels per cluster per iteration
k_membership_counts = zeros(ITERS, k);
```

```
% keep a trace of the centroids per iteration
means_trajectory = zeros(k,d,ITERS);
sum_squared_dist = zeros(1, ITERS);
iteration_count = 1;
% keep k-copies of the data to quickly get the distance to centroids
unfolded_data = repmat(data, 1, 1, k);
% get the L2 norm of the row-slice of the difference between 8 pixel copies and centroids
slice_sq_norm = @(tensor)reshape(sum((tensor.^2),2), size(tensor, 1), size(tensor, 3));

% Repeat the process until done...
while (~k_settled)
  loop = tic();
  disp(sprintf('Clustering all pixels, iteration %d', iteration_count));
  flush();
  k_means = repmat(reshape(new_means', 1, d, k), m, 1, 1);
  k_means_flat = reshape(k_means(1,:,:), d, k)';
  [min_l2_sq, idxs] = min(slice_sq_norm(unfolded_data - k_means), [], 2);
  sum_squared_dist(iteration_count) = sum(min_l2_sq);
  k_labels = idxs;
  disp(sprintf('Reassigning means'));
  flush();
  for kth = 1:k
      temp = zeros(size(data));
      kth_pixels_idx = (k_labels == kth);
      temp(kth_pixels_idx, :) = data(kth_pixels_idx, :);
      temp(~kth_pixels_idx, :) = [];
      k_membership_counts(iteration_count, kth) = size(temp, 1);
      if (size(temp, 1) >= 1)
        new_means(kth, :) = mean(temp);
      end
  end
  means_trajectory(:,:,iteration_count) = k_means_flat;
  disp(sprintf(...
    'Norm of the difference between this iteration and last''s means: %d',...
     norm(new_means-k_means_flat)))
  flush();
  disp(sprintf('iteration %d took %d seconds.', iteration_count, toc(loop)));
  flush();

  % See if we're done
  if (sum(sum(new_means ~= k_means_flat)) == 0)
    k_settled = true;
  else
    iteration_count = iteration_count + 1;
  end

end

% Now convert pixels to their respective centroid
clustered = data;

for kth = 1:k
      clustered(k_labels == kth, :) = k_means(k_labels == kth, :, kth);
end
```

Question 4: K-medoids - advantages and disadvantages vs K-means

K-medoids has two main advantages: it can use any measure of similarity between points, which means it is more flexible than K-means which uses euclidean distance between vectors. The use of medoids as opposed to means also makes the partitioning more robust towards outliers.

Conversely, a disadvantage of K-medoids is that its runtime cost is $O(n^2)$ whereas K-means' is $O(n)$.