UNIVERSITY OF SOUTHAMPTON COMP6208W1

SEMESTER 2 EXAMINATION 2014/2015

ADVANCED MACHINE LEARNING

Duration: 120 mins

Answer THREE *out of* FIVE *questions (each question is worth 20 marks)*

This examination is worth 60%. The coursework was worth 40%.

University approved calculators MAY be used.

A foreign language translation dictionary (paper version) is permitted provided it contains no notes, additions or annotations.

 Number of pages 17

Question 1 Given a set of data $\mathcal{D} = \{(\boldsymbol{x}_k, y_k)|k = 1, 2, \ldots, P\}$, we can perform ridge regression in an extended feature space by minimising the cost function

$$C(\boldsymbol{w}) = \sum_{k=1}^{P} \left(\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}_k) - y_k\right)^2 + \nu\|\boldsymbol{w}\|^2.$$

(a) What is the interpretation of these two terms?

(Students have not seen kernel ridge regression in any detail although the calculations are very close to what they have seen before)

(i) The first term is the squared error between the prediction of the machine and target

(ii) The second term is a weight decay regularisation term

(2 marks)

(b) The minimum cost weight vector will lie in a space spanned by the data-points in the extended feature space so that

$$\boldsymbol{w} = \sum_{\ell=1}^{P} \alpha_\ell \boldsymbol{\phi}(\boldsymbol{x}_\ell).$$

Explain why this is the case?

Any component orthogonal to this subspace does not change the squared error terms (if ϕ is orthogonal to $\phi(x_k)$ for all k, then $\phi^\mathsf{T}\phi(x_k) = 0$). Adding such a term would only increase the length of the weight vector $\|w\|$. Thus, the minimum cost weight vector will have no component outside this subspace.

(3 marks)

(c) Defining the kernel function $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}^\mathsf{T}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{y})$ rewrite the cost function in terms of the kernel and the parameters α_k.

$$C(\boldsymbol{\alpha}) = \sum_{k=1}^{P} \left(\sum_{\ell=1}^{P} \alpha_\ell K(\boldsymbol{x}_\ell, \boldsymbol{x}_k) - y_k\right)^2 + \nu \sum_{k,\ell=1}^{P} \alpha_k\, \alpha_\ell\, K(\boldsymbol{x}_\ell, \boldsymbol{x}_k)$$

(3 marks)

(d) Defining the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{y}$ with components α_k and y_k respectively, and the matrix $\mathbf{K}$ with components $K(\boldsymbol{x}_\ell, \boldsymbol{x}_k)$, rewrite the cost function in matrix form.

$$\begin{aligned} C(\boldsymbol{\alpha}) &= \|\mathbf{K}\,\boldsymbol{\alpha} - \boldsymbol{y}\|^2 + \nu \boldsymbol{\alpha}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\mathsf{T}\mathbf{K}(\mathbf{K} + \nu\mathbf{I})\boldsymbol{\alpha} - 2\,\boldsymbol{\alpha}^\mathsf{T}\mathbf{K}\boldsymbol{y} + \boldsymbol{y}^\mathsf{T}\boldsymbol{y} \end{aligned}$$

(2 marks)

(e) Find the parameters $\boldsymbol{\alpha}^*$ that minimise the cost function.

To compute this we first compute the gradient

$$\boldsymbol{\nabla} C(\boldsymbol{\alpha}) = 2\,\mathbf{K}(\mathbf{K} + \nu\mathbf{I})\boldsymbol{\alpha} - 2\mathbf{K}\boldsymbol{y}$$

Setting this to zero we find

$$\boldsymbol{\alpha} = (\mathbf{K} + \nu\mathbf{I})^{-1}\boldsymbol{y}$$

(5 marks)

(f) Explain why it is important that the kernel function is positive semi-definite.

In this case the kernel function is defined to be $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}^\mathsf{T}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{y})$ where $\phi(\boldsymbol{x})$ is real. This requires the kernel function to be positive semi-definite. If $\phi(\boldsymbol{x})$ was complex then the least-squared would be meaningless.

(2 marks)

(g) Give three conditions that a positive semi-definite kernel must satisfy.

(i) The kernel can always be rewritten as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}^\mathsf{T}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{y}) = \sum_i \phi_i(\boldsymbol{x})\,\phi_i(\boldsymbol{y})$$

for some set of real functions $\phi_i(\boldsymbol{x})$

(ii) The eigenvalues of the kernel function are non-negative

(iii) For any function $f(\boldsymbol{x})$

$$\int f(\boldsymbol{x})\, K(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}$$

It would also be acceptable to say that for any finite set of points the Gram matrix was positive semi-definite, although students have not been told this.

(3 marks)

Question 2

(a) Describe a deep belief network and explained how it is trained.

For full marks I would expect to see the following

(i) Large number of layers

(ii) The layers are pre-trained using unlabelled data

(iii) The pre-training is performed one layer at a time

(iv) In the pre-training the layers are very often trained to be auto-encoders with restricted Boltzmann machine frequently used

(v) After pre-training back-propagation is used on the whole network

(5 marks)

(b) Explain what deep learning attempts to do.

Deep learning attempts to re-represent the input pattern, obtaining a high abstraction of the data in each layer. To achieve this each layer captures common correlations that occur in previous layers.

(3 marks)

(c) What are the main drawbacks of using deep learning?

Deep learning is very slow, requiring sometimes weeks of computation on large problems. In addition it requires a considerable amount of both labelled and unlabelled data.

(2 marks)

(d) Explain how dropout is implemented in deep belief networks.

Dropout is carried out during back-propagation. For each pattern presented to the network a random selection of nodes in the network are switched off. Typically 50% will be switched off except in the input layer where 20% are. Only weights between nodes that are switched on are used and updated. In the final network (after training) all nodes are used, but the weights are decreased according to the probability of the nodes being on.

(5 marks)

(e) Explain what dropout does and why it is used.

Dropout is used to prevent over-fitting, which a multi-layer network is liable to do. Dropout approximates taking the geometric average of a large number of machines (the machines formed by removing half the nodes). By averaging over machines it very significantly reduces the "variance" term in the bias variance-dilemma (this is used in other ensemble methods).

(5 marks)

Question 3

This question will need you to use the KL divergence $KL(p\|q)$ between two distributions p and q:

$$KL(p\|q) = \mathbb{E}_p\left(\log(p/q)\right),$$

where $\mathbb{E}_p f = \sum_x p(x)f(x)$ denotes the expectation of f taken with respect to the distribution p.

(a) For the dataset $\mathcal{X} := \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$, we can compute the *empirical distribution*

$$\tilde{p}(\mathbf{x}) = \frac{1}{N}(\sum_{n=1}^{N} \mathbb{I}[\mathbf{x} = \mathbf{x}^{(n)}]),$$

where

$$\mathbb{I}[x = y] = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases}$$

Find the probability distribution $p(x)$ that minimises the KL divergence $KL(\tilde{p}\|p)$. What is the log likelihood of $p(x)$?

(4 marks)

(b) You are given a data set containing observed values $v^{(n)}$ of variable V. You construct a probability model $p(V = v^{(n)}, H = h^{(n)}|\theta)$ that introduces model hidden variables H that take values $h^{(n)} \in H$ that are associated with observations $v^{(n)}$. To infer the parameters θ you will need to maximise the log likelihood, $\ell(\theta; \mathcal{V})$, of the observed data

$$\sum_{n=1}^{N} \log p(V = v^{(n)}|\theta).$$

Using the KL divergence between a variational distribution $q(h^{(n)}|v^{(n)})$ and the probability model for the hidden variables $p(H = h^{(n)}|V = v^{(n)}, \theta)$, show that the log likelihood $\ell(\theta; \mathcal{V})$ has a lower bound

$$\ell(\theta; \mathcal{V}) \geq \ell_{LB}(\{q\}, \theta) = \sum_{n=1}^{N} S_n + E_n,$$

where

$$S_n = -\sum_h q(h^{(n)}|v^{(n)}) \log \Big(q(h^{(n)}|v^{(n)}) \Big)$$

and E_n is the expected complete (hidden and observed) data log likelihood:

$$E_n = \sum_h q(h^{(n)}|v^{(n)}) \log \Big(p(h^{(n)}, v^{(n)}|\theta) \Big).$$

For what choice of the variational distribution $q(h^{(n)}|v^{(n)})$ is the lower bound reached? *(6 marks)*

(c) A data set $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ is shown in the table below, where the variable $\mathbf{X} = (A, B)$ with A and B binary:

$\mathbf{X}$	A	B
$\mathbf{x}^{(1)}$	0	0
$\mathbf{x}^{(2)}$	1	0
$\mathbf{x}^{(3)}$	0	0
$\mathbf{x}^{(4)}$	1	?
$\mathbf{x}^{(5)}$	?	0
$\mathbf{x}^{(6)}$	1	1

This questions uses the EM algorithm to find the parameters $\theta_{ij} := p(A = i, B = j)$, (for example, $\theta_{00} := p(A = 0, B = 0)$) from $\mathcal{X}$ which contains missing values denoted by '?'.

(i) Write down the conditional probabilities for the missing values $p(A = a|B = b)$ and $p(B = b|A = a)$ for a given (perhaps random) choice of parameters $\theta_{ij}^{(old)}$.

(ii) Write down the complete log likelihood of the data $\mathcal{X}$ using the estimated conditional probabilities for the missing values $q_t(A = a|B = b)$ and $q_t(B = b|A = a)$.

(iii) For what set of parameters is the log likelihood of the hidden and visible data maximised?

(10 marks)

(a) (4 marks) The KL divergence $KL(p\|q)$ is minimised for $p = q$. Hence $p = \tilde{p}$ in this case. The log likelihood of the empirical distribution is thus maximal.

(b) (6 marks) Since the KL divergence

$$\begin{aligned} 0 \leq KL\left(q(h^{(n)}|v^{(n)})\|p(h^{(n)}|v^{(n)},\theta)\right) &= \left\langle \log\left(\frac{q(h^{(n)}|v^{(n)})}{p(h^{(n)}|v^{(n)},\theta)}\right)\right\rangle_{q(h^{(n)}|v^{(n)})} \\ &= \left\langle \log\left(q(h^{(n)}|v^{(n)})\right)\right\rangle_{q(h^{(n)}|v^{(n)})} - \left\langle \log\left(p(h^{(n)}|v^{(n)},\theta)\right)\right\rangle_{q(h^{(n)}|v^{(n)})}, \end{aligned}$$

and

$$\log p(h^{(n)}, v^{(n)}|\theta) = \log p(h^{(n)}|v^{(n)},\theta) + \log p(v^{(n)}|\theta),$$

we obtain

$$0 \leq \left\langle \log\left(q(h^{(n)}|v^{(n)})\right)\right\rangle_{q(h^{(n)}|v^{(n)})} - \left\langle \log\left(p(h^{(n)}, v^{(n)}|\theta)\right)\right\rangle_{q(h^{(n)}|v^{(n)})} + \log p(v^{(n)}|\theta).$$

The result follows. The lower bound is reached when the variational distribution is equal to the posterior distribution over the hidden variables:

$$q(h^{(n)}|v^{(n)}) = p(h^{(n)}|v^{(n)},\theta).$$

(c) (10 marks)

(i) $P(A = a|B = b) = P(A = a, B = b)/\sum_{a=0,1} P(A = a, B = b) = \theta_{ab}/(\theta_{0b} + \theta_{a0})$

(ii) I'll give the general solution here. Let n_{ij} be the number of observations of the pair of values (i, j):

$$\textbf{For } i,j \in \{0,1\},\ n_{ij} = \sum_n \mathbb{I}[(a^{(n)}, b^{(n)}) = (i,j)],$$

and $h_{\circ,i}$ be the number of times the first entry is missing while the second entry is observed and $h_{i,\circ}$ for the other way around:

$$\begin{aligned} h_{\circ,i} &= \textstyle\sum_n \mathbb{I}[(a^{(n)}, b^{(n)}) = (?, i)], \\ h_{i,\circ} &= \textstyle\sum_n \mathbb{I}[(a^{(n)}, b^{(n)}) = (i, ?)]. \end{aligned}$$

• For the solution expected of the students, we will need to substitute specific values for $n_{ij}, h_{i\circ}, h_{\circ j}$ in the expressions below.

n_{00}	n_{01}	n_{10}	n_{11}
2	0	1	1
$h_{\circ 0}$	$h_{\circ 1}$	$h_{0\circ}$	$h_{1\circ}$
1	0	0	1

Using the shorthand notation $(H = h^{(n)}, V = v^{(n)})$ to stand for the presence of both hidden and seen variables, where either or both of A, B could be H or V, we denote by $q_t(H = h|V = v, \theta_t)$, the variational distribution over the hidden variables conditioned on the observed data and the estimate of parameters θ. The expected complete data log

likelihood is expressed as:

$$\langle \log p(h^{(n)}, v^{(n)}|\theta)\rangle_{q_t(h|v,\theta)} = \sum_n \mathbb{I}[(a^{(n)}, b^{(n)}) = (i,j)] \log p(a^{(n)} = i, b^{(n)} = j) \quad A, B \textbf{ observed,}$$
$$+ \sum_{n=1}^{N} \mathbb{I}[(a^{(n)}, b^{(n)}) = (?,j)] \sum_{i=0,1} q_t(a^{(n)} = i|b^{(n)} = j, \theta) \log p(a^{(n)} = i, b^{(n)} = j \quad B \textbf{ observed}, A \textbf{ hidden},$$
$$+ \sum_{n=1}^{N} \mathbb{I}[(a^{(n)}, b^{(n)}) = (i,?)] \sum_{j=0,1} q_t(b^{(n)} = j|a^{(n)} = i, \theta) \log p(a^{(n)} = i, b^{(n)} = j) \quad A \textbf{ observed}, B \textbf{ hidden}.$$

(iii) Collecting terms involving $\log p(a^{(n)} = i, b^{(n)} = j) = \theta_{ij}$, we have

$$\langle \log p(h^{(n)}, v^{(n)}|\theta)\rangle_{q_t(h|v,\theta)} = \log \theta_{ij} \left(n_{ij} + h_{i\circ} q_t(b^{(n)} = j|a^{(n)} = i) + h_{\circ j} q_t(a^{(n)} = i|b^{(n)} = j) \right).$$

M-step: Upon setting the derivative of the expected complete log likelihood with respect to θ_{ij} to zero:

$$\begin{aligned} \theta_{ij}^{new} &\propto \left(n_{ij} + h_{i\circ} q_t(b^{(n)} = j|a^{(n)} = i) + h_{\circ j} q_t(a^{(n)} = i|b^{(n)} = j) \right) \\ &= \frac{1}{N} \left(n_{ij} + h_{i\circ} \frac{\theta_{ij}}{\sum_j \theta_{ij}} + h_{\circ j} \frac{\theta_{ij}}{\sum_i \theta_{ij}} \right). \end{aligned}$$

E-step: These new values of θ are used to update the conditionals $q_{t+1}(h|v, \theta^{new})$.

Question 4

(a) In a graphical model a node is shaded if a possible value taken by the variable it represents is observed. By writing down the joint probabilities of the following graphical models determine when A and B are (conditionally or otherwise) independent.

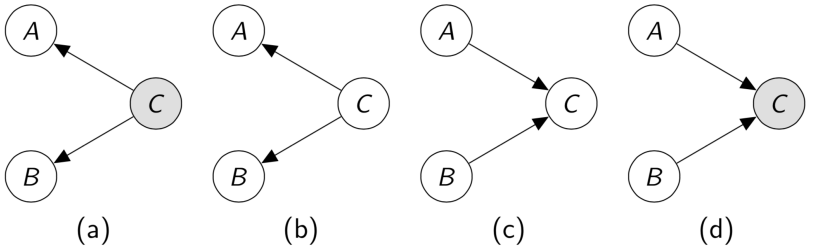

FIGURE 1: A shaded node indicates that a value taken by the corresponding variable is observed.

(4 marks)

(b) A dataset $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$ consists of a 1-dimensional output $y^{(n)}$ for each D-dimensional input data point $\mathbf{x}^{(n)} = (x_1^{(n)}, \ldots, x_D^{(n)})$ for $n = 1, \ldots, N$, and the collections of inputs and outputs are denoted $\mathcal{X}$ and $\mathcal{Y}$ respectively. The data is fit by a linear function $f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^T\mathbf{x} = \beta_0 + \sum_j \beta_j x_j$. The residual errors $\epsilon^{(n)} := (y^{(n)} - f(\mathbf{x}^{(n)}))$ are identically and independently distributed (iid) according to a Gaussian distribution with zero mean and standard deviation σ. Write down the log likelihood $f(\mathcal{Y}|\beta_0, \boldsymbol{\beta}, \mathcal{X})$ of the data in this model.

(3 marks)

(c) If the weights $\boldsymbol{\beta}$ are drawn from a prior distribution,

$$p(\boldsymbol{\beta}) = \prod_{j=1}^{D} p(\beta_j),$$

write down an expression proportional to the posterior distribution

$$p(\boldsymbol{\beta}|\mathcal{X}, \mathcal{Y}, \beta_0)$$

given the data set $\mathcal{D}$.

(3 marks)

(d) Show that by choosing a Gaussian $p(\beta_j|a) \propto \exp(-\beta_j^2/a)$ or a double-exponential (Laplace) prior $p(\beta_j|a) \propto \exp(-|\beta_j|/a)$, the mode of the posterior distribution in the previous part gives the ridge regression or lasso formulation of the parameter estimation task framed as an optimisation problem. Contrast the effects of altering $a > 0$ on the coefficients $\boldsymbol{\beta}$ in ridge regression and lasso.

(5 marks)

(e) In the AdaBoost algorithm, the classification rule $F : X \rightarrow Y$ where $x \in X$ is an input vector and $Y = \{1, -1\}$ a binary output, and $F(x) = \text{sign}(\sum_m c_m f_m(x))$. Prove that the expectation, taken with respect to the conditional distribution $p(Y = y|X = x)$ of the loss function $L(y, f_m(x)) = \exp(-y f_m(x))$:

$$E_{Y|X=x}(e^{-y f_m(x)})$$

is minimised at

$$f_m(x) = \frac{1}{2} \log \left(\frac{p(y = 1|x)}{p(y = -1|x)} \right).$$

(5 marks)

(a) (4 marks)

a $A \perp\!\!\!\perp B|C$

b $A \not\perp\!\!\!\perp B$

c $A \perp\!\!\!\perp B$

d $A \not\perp\!\!\!\perp B|C$

(b) (3 marks) The log likelihood of the model $\ell(\beta_0, \boldsymbol{\beta}; \mathcal{X}, \mathcal{Y})$ with parameters $\beta_0, \boldsymbol{\beta}$ is obtained by adding the contributions of each data point:

$$\begin{aligned} p(y^{(n)}|\beta_0, \boldsymbol{\beta}, \mathbf{x}^{(n)}) &\propto \exp\left(-\frac{1}{2\sigma^2}\left(y^{(n)} - \beta_0 - \sum_j \beta_j x_j^{(n)}\right)^2\right) \\ \Rightarrow \ell(\beta_0, \boldsymbol{\beta}; \mathcal{X}, \mathcal{Y}) = -\sum_n \log p(y^{(n)}|\beta_0, \boldsymbol{\beta}, \mathbf{x}^{(n)}) &= \frac{1}{2\sigma^2}\sum_n (y^{(n)} - \beta_0 - \sum_j \beta_j x_j^{(n)})^2 + \textbf{ const} . \end{aligned}$$

(c) (3 marks)

$$p(\boldsymbol{\beta}|\mathcal{X}, \mathcal{Y}, \beta_0) \propto p(\mathcal{Y}|\mathcal{X}, \beta_0, \boldsymbol{\beta})p(\boldsymbol{\beta})$$

(d) (5 marks) From the previous answer for the posterior distribution over $\boldsymbol{\beta}$,

$$\begin{aligned} -\log p(\boldsymbol{\beta}|\mathcal{X}, \mathcal{Y}, \beta_0) &= \frac{1}{2\sigma^2}\sum_n (y^{(n)} - \beta_0 - \sum_j \beta_j x_j^{(n)})^2 + \frac{1}{a}\sum_j \beta_j^2 \textbf{ Gaussian} \\ &= \frac{1}{2\sigma^2}\sum_n (y^{(n)} - \beta_0 - \sum_j \beta_j x_j^{(n)})^2 + \frac{1}{a}\sum_j |\beta_j| \textbf{ Laplace} \end{aligned}$$

(e) (5 marks) The derivative of the expectation value of the loss function $\mathcal{L}(y, f_m(x)) = \exp(-y f_m(x))$ is:

$$\frac{\partial}{\partial f_m(x)}\mathbb{E}_{Y|x}\exp(-y f_m(x)) = \frac{\partial}{\partial f_m(x)}\left[p(Y=1|x)e^{-f_m(x)} + p(Y=-1|x)e^{f_m(x)}\right].$$

The result follows upon setting the derivative to 0.

Question 5

(a) **Provide** a description of the MapReduce framework. **Include** details of both the *programming model* and underlying *distributed data model*.

(8 marks)

(b) Assume that you have a large number of feature vectors stored on a distributed file system across a cluster of machines. **Describe** how you might apply the MapReduce programming model to distribute the problem of applying *k-means clustering* to the data. Also **describe** any *limitations* of your approach and any possible *workarounds*.

(12 marks)

(a)
Looking for knowledge and coverage of both the programming model and data model. Marks for: Describing the three key phases (map, shuffle, reduce), including details of what the inputs and outputs are; Mentioning that the model makes it easy to run the processing in parallel; Mentioning that in real frameworks, the data being processed is distributed and that the framework aims to push computation to the data.

The MapReduce programming model is founded in ideas from functional programming, although they are used in a much more flexible way in the MapReduce model. The MapReduce model transforms a list of $< key, value >$ pairs into a list of values. The Map function takes in a key-value pair from the data being processed and outputs zero or more key-value pairs of data (which can be of a different type to the input):

$Map(k1, v1) \rightarrow list(k2, v2)$

The 'shuffle' phase collates all the Map function outputs by their key, producing a list of $< key, [values] >$, sorted by key:

$list(k2, v2) \rightarrow list(k2, list(v2))$

The Reduce function is handed each $< key, [values] >$ pair and produces zero or more values as a result:

$Reduce(k2, list(v2)) \rightarrow list(v3)$

The MapReduce programming model is useful because it is easily parallelizable; Provided that each map operation is independent of the others, all maps can be performed in parallel (in practice this is limited by the number of data sources and processing units). Similarly, the reduce phase can be performed in parallel as the data for each key is processed independently. It should be noted that practical MapReduce programs often contain multiple Map and Reduce operations, and sometimes even omit the Reduce operation.

Practical implementations of MapReduce (i.e. Hadoop) work by storing data on a distributed file-system that is spread across a large number of machines. When a MapReduce job is launched, the Map functions are ideally handed key-value pairs from blocks of data that are stored locally to that machine, in order to minimise the overhead from having to copy data around. In essence, the MapReduce framework allows the computation to be pushed to where the data is stored, rather than pulling the data over a network to the processing nodes.

(b)
Looking for a basic workable solution to distributed k-means; full marks for describing limitations and potential workarounds.

Basic solution looks like this:

(a) Generate K initial cluster centroids (e.g. canopy clustering, random vectors, random sampling) and store in a file (on the DFS).

(b) Repeat until convergence (i.e centroids don't change) or until a maximum number of iterations is reached:

- **In the Map:**
  - **(i) Read the cluster centers into memory from the file**
  - **(ii) Input key-value pair from the data being clustered (assume that the feature vector is the value; the key is irrelevant).**
  - **(iii) Compute the distance from the vector to each cluster and find the index of the cluster centre with the smallest distance.**
  - **(iv) Emit the <index, vector> pair**
- **In the Reduce:**
  - **(i) Input the key-list(value) pair from the shuffle (the list(value) is all the vectors belonging to that cluster and the key is the index of the centroid in question)**
  - **(ii) Compute the average vector from list(value)s.**
  - **(iii) Store the new average vector in place of the original vector with the same index in the centroids file.**

The biggest limitations of the above approach are the cost of reading the centroids in each map and the cost of transferring all the vectors to the reducer. In a real implementation (i.e. one based on Hadoop), the first issue can be solved by reading the data once per mapper, and sharing it across multiple map function calls. The second issue could potentially be solved by a Combiner which computes a running average of the updated centroids for each mapper; the reducer would then be only receive the average centroid from each combiner (together with a count of how many vectors were assigned to that centroid), rather than all the vectors, which it could then use to compute the actual updated centroid.
