CS6140: Machine Learning

Homework Assignment # 2

Assigned: 02/03/2019 *Due: 02/17/2019, 11:59pm, through Blackboard*

Five problems, 130 points in total. Good luck!
Prof. Predrag Radivojac, Northeastern University

Problem 1. (20 points) Suppose that the number of accidents occurring daily in a certain plant has a Poisson distribution with an unknown mean λ. Based on previous experience in similar industrial plants, suppose that our initial feelings about the possible value of λ can be expressed by an exponential distribution with parameter $\theta = \frac{1}{2}$ is, the prior density is

$$p(\lambda) = \theta e^{-\theta\lambda}$$

where $\lambda \in (0, \infty)$. If there are 79 accidents over the next 9 days, determine

a) (5 points) the maximum likelihood estimate of λ

b) (5 points) the maximum a posteriori estimate of λ

c) (10 points) the Bayes estimate of λ.

Solution:

1. Maximum likelihood estimate

We need to estimate parameter λ(accidents per day) for dataset D with total of 79 accidents in 9 days. The maximum likelihood function $p(D|\lambda)$ can be written as follows:

$$l = p(D|\lambda) = \prod \frac{\lambda^{x_i}.e^{-\lambda}}{x_i!} = \frac{\lambda^{\sum x_i}.e^{-n\lambda}}{\prod x_i!}$$

where, $x_i$is a data point in set D.

$\lambda_{MLE} = argmax(p(D|\lambda))$

Taking log of likelihood function we get the following,

$ll = \sum x_i.(log\lambda) - n.\lambda - log(\prod x_i!) = \sum x_i.(log\lambda) - n.\lambda - \sum log(x_i!)$ ($\forall$i$\in${1,2,3,..,n})

$\frac{\partial}{\partial\lambda}(ll) = \frac{\sum x_i}{\lambda} - n$

At the maximum of likelihood function the partial derivative will be 0.

$\Rightarrow \frac{\sum x_i}{\lambda} - n = 0$

$\Rightarrow \lambda = \frac{\sum x_i}{n}$ ($\forall$i$\in${1,2,3,..,n})

We know that total of D is 79($\sum x_i$) and number of points is 9(n).

$\Rightarrow\lambda = \frac{79}{9} = 8.7778$

2. Maximum a posteriori estimate

$\lambda_{MAP} = argmax(p(D|\lambda).p(\lambda))$

Substituting $p(\lambda) = \theta.e^{-\theta\lambda}$,

$p(D|\lambda).p(\lambda) = \prod(\frac{\lambda^{x_i}.e^{-\lambda}}{x_i!}.\theta.e^{-\theta\lambda}) = (\frac{\lambda^{\sum x_i}}{\prod x_i}.\theta.e^{-\lambda(n+\theta)})$

Taking log of this function,

$log(p(D|\lambda).p(\lambda)) = log(\frac{\lambda^{\sum x_i}e^{-n\lambda}}{\prod x_i}.\theta.e^{-\lambda(n+\theta)})$

$= log\lambda.\sum x_i + log\theta - \lambda(n+\theta) - \sum log(x_i!)$

To get maximum point of above MAP function we differentiate w.r.t λ

$\frac{\partial log(p(D|\lambda).p(\lambda))}{\partial\lambda} = \frac{\sum x_i}{\lambda} - (n+\theta) = 0$ ($\forall$i$\in${1,2,3,..,n})

$\Rightarrow\lambda = \frac{\sum x_i}{n+\theta}$

$\Rightarrow\lambda = \frac{79}{9+0.5} = 8.3157$

3. Bayes' estimate

Bayes Estimate is expectation of paraemeter under the conditions of given data set, mathematically expressed as follows,

$E[\lambda|D] = \int \lambda.p(\lambda|D).d\lambda = \int \lambda.\frac{p(D|\lambda).p(\lambda)}{p(D)}.d\lambda$

We know $p(\lambda)$, and need to calculate $p(D|\lambda)$ and $p(D)$,

$p(D|\lambda) = \prod \frac{\lambda^{x_i}.e^{-\lambda}}{x_i!} = \frac{\lambda^{\sum x_i}.e^{-n\lambda}}{\prod x_i!}$

To find p(D) we can marginalize $p(D,\lambda)$ over λ,

$p(D) = \int p(D|\lambda).p(\lambda).d\lambda$

$=\int \frac{\lambda^{\sum x_i}.\theta.e^{-\lambda(n+\theta)}}{\prod x_i!}.d\lambda$

Refering to the lecture notes, we know integration of the form $\int x^{\alpha-1}.e^{-\beta x}.dx = \frac{\Gamma(\alpha)}{\beta^\alpha}$

To solve above integral form we assume, $\alpha = \sum x_i + 1$ and $\beta = (n+\theta)$

On solving we get,

$p(D) = \frac{\theta}{\prod x_i!}(\frac{\Gamma(\sum x_i+1)}{(n+\theta)^{\sum x_i+1}})$

Now we can write the equation for expectation as follows:

$E[\lambda|D] = \int \frac{\frac{\lambda.\lambda^{\sum x_i}.\theta.e^{-\lambda(n+\theta)}}{\prod x_i!}}{\frac{\theta.\Gamma(\sum x_i+1}{\prod x_i!(n+\theta)^{\sum x_i+1}}}.d\lambda = \frac{(n+\theta)^{\sum x_i+1}}{\Gamma(\sum x_i+1)}\int \lambda^{\sum x_i+1}.e^{-\lambda(n+\theta)}.d\lambda$

Here we can assume, $\alpha = (\sum x_i + 1)$+1 and $\beta = (n+\theta)$

$E[\lambda|D] = \frac{(n+\theta)^{\sum x_i+1}}{\Gamma(\sum x_i+1)}.\frac{\Gamma(\sum x_i+2)}{(n+\theta)^{\sum x_i+1}} = \frac{(\sum x_i+1)}{n+\theta}$

Substituting the known values of $\sum x_i = 79, n = 9, \theta = 0.5$, we get

$E[\lambda|D] = \frac{79+1}{9+0.5} = 8.4210$

Problem 2. (15 points) Let X_1, X_2, ..., X_n be i.i.d. Gaussian random variables, each having an unknown mean θ and known variance σ_0^2. If θ is itself selected from a normal population having a known mean μ and a known variance σ^2, determine

a) (5 points) the maximum a posteriori estimate of θ

b) (10 points) the Bayes estimate of θ.

Solution:

1. Maximum a posteriori estimate

We know that $\theta_{MAP} = argmax\{p(D|\theta).p(\theta)\}$

$p(D|\theta) = \prod p(x_i|\theta) = \prod \frac{1}{\sqrt{2\pi\sigma_o^2}}.e^{-\frac{(x_i-\theta)^2}{2\sigma_o}} = \frac{1}{\sqrt{2\pi\sigma_o^2}}.e^{-\frac{\sum(x_i-\theta)^2}{2\sigma_o}}$

Taking log of $p(D|\theta)$,

$log(p(D|\theta)) = n.log(\frac{1}{\sqrt{2\pi\sigma_o^2}}) - \frac{\sum(x_i-\theta)^2}{2\sigma_o}$

Also, $p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}}.e^{-\frac{(\theta-\mu)^2}{2\sigma}}$

Taking log of $p(\theta) = log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(\theta-\mu)^2}{2\sigma^2}$

Therefore, $log(p(D|\theta).p(\theta)) = n.log(\frac{1}{\sqrt{2\pi\sigma_o^2}}) - \frac{\sum(x_i-\theta)^2}{2\sigma_o} + log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{(\theta-\mu)^2}{2\sigma^2}$

Now to maximize the Posterior we will differentiate the above function w.r.t θ and equate it to 0,

$\frac{\partial log(p(D|\theta).p(\theta))}{\partial\theta} = \frac{\sum x_i-\theta}{\sigma_o^2} - \frac{\theta-\mu}{\sigma^2} = 0$

$\Rightarrow \frac{\sum x_i-\theta}{\sigma_o^2} = \frac{\theta-\mu}{\sigma^2}$

$\Rightarrow \frac{\sigma^2}{\sigma_o^2}\sum x_i - \frac{\sigma^2}{\sigma_o^2}n\theta = \theta - \mu$

$\Rightarrow \frac{\sigma^2}{\sigma_o^2}\sum x_i + \mu = \theta + \frac{\sigma^2}{\sigma_o^2}n\theta$

Solving above equation for θ,we get following MAP solution:

$$\theta = \frac{\frac{\sigma^2}{\sigma_o^2}\sum x_i + \mu}{1 + n\frac{\sigma^2}{\sigma_o^2}}$$

2. Bayes' Estimate

Bayes estimate is expectation of posterior distribution,

$E[\theta|D] = \int p(\theta|D).\theta.d\theta$

The integral of this function is very hard to solve, so we can use a better way of estimating using proportionality by dropping all terms that do not depend on θ.

$p(\theta|D).p(\theta)$ is proportional to $e^{\frac{\sum x_i - \theta}{\sigma_o^2} + \frac{(\theta-\mu)^2}{\sigma^2}}$

Let us solve the component in the exponent above,

$\frac{\sum x_i - \theta}{\sigma_o^2} + \frac{(\theta-\mu)^2}{\sigma^2} = \frac{\sum(x_i - 2x_i) + n\theta^2}{\sigma_o^2} + \frac{\theta^2 - 2\theta\mu + \mu^2}{\sigma^2} = \frac{1}{\sigma_1^2}(\theta - \mu_1)^2 + C$

where, $\mu_1 = \frac{\mu}{\sigma^2} + \frac{\sum x_i}{\sigma_o^2}$, $\frac{1}{\sigma_1^2} = \frac{n}{\sigma_o^2} + \frac{1}{\sigma^2}$ and C is a constant.

We showed that $p(\theta|D).p(\theta)$ is proportional to $e^{\frac{1}{2\sigma_1^2}(\theta-\mu_1)^2}$

This suggests that posterior function is a normal distribution, which will have a mean of μ_1

Therefore, $E[\theta|D] = \mu_1 = \frac{\mu}{\sigma^2} + \frac{\sum x_i}{\sigma_o^2}$

Problem 3. (25 points) Let $\mathcal{D} = \{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}$, be a data set drawn independently from a Gumbel distribution

$$p(x) = \frac{1}{\beta} e^{-\frac{x-\alpha}{\beta}} e^{-e^{-\frac{x-\alpha}{\beta}}},$$

where $\alpha \in \mathbb{R}$ is the location parameter and $\beta > 0$ is the scale parameter.

a) (10 points) Derive an algorithm for estimating α and β.

b) (10 points) Implement the algorithm derived above and evaluate it on data sets of different sizes. First, find or develop a random number generator that creates a data set with $n \in \{100, 1000, 10000\}$ values using some fixed α and β. Then make at least 30 data sets for each n and estimate the parameters. For each n, report the mean and standard deviation on the estimated α and β. If $n = 10000$ is too large for your computing resources, skip it.

c) (5 points) The problem above will require you to implement an iterative estimation procedure. You will need to decide on how to initialize the parameters, how to terminate the estimation process and what the maximum number of iterations should be. Usually, some experimentation will be necessary *before* you run the experiments in part (b) above. Summarize what you did in a short paragraph, no more than two paragraphs.

NB: There are several versions and naming conventions for the Gumbel distribution in the literature.

Solution:

1. Gradient Descent Algorithm for estimating the parameters of Gumbel Distribution:

We will follow the Maximum Likelihood approach (since the prior is unknown) and later use the gradient descent algorithm to minimize the negative log-likelihood function in order to achieve the obective of maximizing the positive of Log-Likelihood function. First, lets calculate the log-likelihood for both the parameters of gumbel distribution.

We know that, $\theta_{MLE} = argmax(p(D|\theta))$

$p(D|\theta) = \prod p(x_i|\theta) = \prod p(x_i|\alpha, \beta)$

$= \frac{1}{\beta^n} e^{\prod - \frac{x_i - \alpha}{\beta}} e^{\prod - e^{-\frac{x_i - \alpha}{\beta}}}$

Taking Log of above equation, we get:

$ll = log(p(D|\theta)) = -n.log\beta - \frac{\sum x_i - \alpha}{\beta} - \sum e^{-\frac{x_i - \alpha}{\beta}}$

$$-ll = n.log\beta + \frac{\sum(x_i-\alpha)}{\beta} + \sum e^{-\frac{x_i-\alpha}{\beta}} = n.log\beta + \frac{\sum x_i}{\beta} - \frac{n\alpha}{\beta} + \sum e^{-\frac{x_i-\alpha}{\beta}}$$

Let us take the negative of this function here itself so that further calculations are done to minimize the negative ll function(-ll)

We need 2 gradients for our algorithm, one for each parameter. so we differentiate -ll w.r.t each parameter.

$$\frac{\partial}{\partial\alpha}(-ll) = -\frac{n}{\beta} + \frac{\sum e^{-\frac{x_i-\alpha}{\beta}}}{\beta}$$

and, $\frac{\partial}{\partial\beta}(-ll) = \frac{n}{\beta} - \frac{\sum x_i - n\alpha}{\beta^2} + \frac{\sum(x_i-\alpha).e^{-\frac{(x_i-\alpha)}{\beta}}}{\beta^2}$

We have 2 gradients for 2 parameters of -ll which we need to minimize, and now we can write our gradient descent algorithm(we will use classic gradient descent without adaptive step size):

Algorithm:

Notations- g: Gradients of 2 parameters (∇c(w)), obj: objective function, w: iterative set of parameters, c: definition of objective function

```
1. Initialize $\eta$(learning rate), w (which is a pair of $\alpha^o$, $\beta^o$), tolerance
2. w ← w$_t$, obj ← c(w)
3. while t is within max iteration limit:
    • temp ←w$_t$-$\eta$g
    • if c(w$_{t+1}$) <absolute(obj – tolerance))
        – w$_{t+1}$ = temp
    • else: break
4. increment t = t+1 until convergence
5. report final w
```

2. Implementation of above derived algorithm:

On performing 30 test cases of above algorithm on data size of 100, 1000 and 10000, following results were observed:

Data Size(100):

VALUES	**alpha**	**beta**	**time(mm:ss.ms)**
TRUE	4	5	-
MEAN	3.962968016	4.877730184	00:00.3
STD DEV	0.36633006	0.438813145	00:00.2

Data Size(1000):

VALUES	**alpha**	**beta**	**time(mm:ss.ms)**
TRUE	4	5	-
MEAN	4.007331661	4.969143788	00:03.6
STD DEV	0.117044495	0.093727892	00:01.6

Data Size(10,000):

VALUES	alpha	beta	time(mm:ss.ms)
TRUE	4	5	-
MEAN	4.004925497	4.974377999	00:02.7
STD DEV	0.053107706	0.039068816	00:00.3

Here we observe that as data size increases the gradient reaches very close to minimum point, making our estimated parameters very close to true paramters.

Since standard deviation is very low in results with data size of 10000, we can say that gradient descent with high data points always reaches the minima. Also the time taken does not increases by a high factor as intuitively we can say that with high data points the gradient descent converges better and quickly.

More detailed results with estimations for each test case are present in the spreadsheet attached with tab name as "Problem3".

3. Summary of experimentation for setting up the gradient descent:

Initial few trials with gradient descent did not show any good results, as estimated values were either too high or too low when selecting learning rate between 0 and 1. This gave an opportunity to look deeper into the functioning of the algorithm and testing the values of w and g after each iteration. Resultingly, I realized that gradient (g) values are too high and according to that my learning rate should be very low. High gradient were shooting the w values over minima and solutions would never converge. I decided to set a learning rate too low, as low as $10^{-6} to 10^{-7}$.Setting the rate low and iterations high, will definitely increase w slowly over the objective function and would result in the better convergence. This experiment and intuitive realization led me to get converging solutions for the gradient descent algorithm.

Problem 4. (30 points) Let $\mathcal{D} = \{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}$, be a data set drawn independently from the mixture of two distributions

$$p(x) = w_1 p_1(x) + w_2 p_2(x),$$

where

$$p_1(x) = \frac{1}{\beta} e^{-\frac{x-\alpha}{\beta}} e^{-e^{-\frac{x-\alpha}{\beta}}},$$

is a Gumbel distribution with parameters α and β,

$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

is a Gaussian distribution with parameters μ and σ, and (w_1, w_2) are positive constants such that $w_1 + w_2 = 1$.

a) (15 points) Derive an EM algorithm for estimating w_1, w_2, α, β, μ and σ.

b) (15 points) Implement the algorithm derived above and evaluate it on data sets of different sizes. Use similar experimentation as in Problem 3.

You can recycle derivations and code from Problem 3. You can also use any result and derivation from the lecture notes posted during the week of January 14 on the class web site.

Solution:

1. EM algorithm for mixture of a gumbel and a gaussian distribution:

Expectation of log-likelihood E, can be expressed as $E[ll|D, \theta^{(t)}] = \sum_i \sum_j log(w_j.p(x_i|\theta_j)).p_{Y_i}(j|x_i, \theta^{(t)})$

$\Rightarrow E = \sum_i [(log w_1.p_{Y_i}(1|x_i, \theta^{(t)})) + (log(p(x_i|\theta_1)).p_{Y_i}(1|x_i, \theta^{(t)})) + (log w_2.p_{Y_i}(2|x_i, \theta^{(t)})) + (log(p(x_i|\theta_2)).p_{Y_i}(2|x_i, \theta^{(t)}))] + \alpha(w_1 - w_2) + 1 = 0$

We have already derived the derivatives of log likelihood for gumbel distribution in problem 3. Solving for the parameters we will get the following results:

$\alpha = -\beta log[\frac{1}{n}\sum e^{-(\frac{x_i}{\beta})}]$ and $\beta = \frac{\sum x_i}{n} - \frac{\sum x_i e^{-(\frac{x_i}{\beta})}}{\sum e^{-(\frac{x_i}{\beta})}}$

Refering to problem 5 here, we have derived the derivative of E w.r.t μ and σ. Solving for these parameters we get,

$\mu = \frac{\sum p_{Y_i}(2|x_i,\theta^{(t)}).x_i}{\sum p_{Y_i}(2|x_i,\theta^{(t)})}$and $\sigma^2 = \frac{\sum p_{Y_i}(2|x_i,\theta^{(t)}).(x_i-\mu)^2}{\sum p_{Y_i}(2|x_i,\theta^{(t)})}$

also, $w_1 = \sum p_{Y_i}$

Given above parameter updates, we can write the EM algorithm as follows:

1. Initialize,$w_1, \alpha, \beta, \mu, \sigma$
2. Set t= 0
3. Repeat until convergence.
   - Calculate the probabilties of components at each x$_i$
   - Calculate the parameters according to above mentioned update formulas for ()
   - t=t+1
4. Report the resulting parameters

Problem 5. (40 points) Let $\mathcal{D} = \{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}$, be a data set drawn independently from the mixture of two distributions

$$p(x) = w_1 p_1(x) + w_2 p_2(x),$$

where

$$p_j(x) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}, \quad j = 1, 2,$$

are Gaussian distributions with parameters μ_j and σ_j, and (w_1, w_2) are positive constants that sum to one.

a) (10 points) Derive a gradient descent algorithm to estimate w_1, w_2, μ_1, σ_1, μ_2 and σ_2.

b) (15 points) Implement the algorithm derived above and evaluate it on data sets of different sizes and with potentially different parameters. Use similar experimentation as in Problems 3 and 4, but note that different accuracy of estimated parameters can be obtained for different values of all six parameters.

c) (15 points) Implement the EM algorithm for estimating the parameters of a mixture of two univariate Gaussians and compare the accuracy of estimation between gradient descent and the EM algorithm. You can derive the update rules for the EM algorithm yourself or find them on the internet.

You can reuse derivations and code from other Problems in this assignment. You can also use any result and derivation from the lecture notes posted during the week of January 14 on the class web site. Similarly, you can use the class code without restrictions whenever it contributes to solving the problem you have at hand.

Solution:

1. Gradient descent algorithm for mixture of 2 gaussian distributions:

We have 6 parameters, w_1, w_2, μ_1, σ_1, μ_2 and σ_2 to optimize, out of which $w_2$can be set as redundant parameter, because of the constraint w_1+$w_2 = 1$

Therefore we need to estimate 5 parameters, for which we will need 5 gradients to maximize the Expectation of log-likelihood function (similar to objective of EM algorithm).

Expectation of log-likelihood E, can be expressed as $E[ll|D, \theta^{(t)}] = \sum_i \sum_j log(w_j.p(x_i|\theta_j)).p_{Y_i}(j|x_i, \theta^{(t)})$

$\Rightarrow E = \sum_i [(log w_1.p_{Y_i}(1|x_i, \theta^{(t)}))+(log(p(x_i|\theta_1)).p_{Y_i}(1|x_i, \theta^{(t)}))+(log w_2.p_{Y_i}(2|x_i, \theta^{(t)}))+(log(p(x_i|\theta_2)).p_{Y_i}(2|x_i, \theta^{(t)}))]+ \alpha(w_1 - w_2) + 1 = 0$

where α is the lagrangian multiplier for contrained optimization of $w_1$and w_2.

We have a mixture of 2 gaussians, so let us solve for parameters of j=1 and we can re-use the results for j=2,

Taking partial derivative w.r.t $w_1$and w_2 we get,

$\frac{\partial E}{\partial w_1} = \sum_i \frac{p_{Y_i}(1|x_i,\theta^{(t)})}{w_1} + \alpha = 0$ and $\frac{\partial E}{\partial w_2} = \sum_i \frac{p_{Y_i}(2|x_i,\theta^{(t)})}{w_2} + \alpha = 0$

Solving for α using above two equations we will get α=-n

Therefore,

$\frac{\partial E}{\partial w_1} = \sum_i \frac{p_{Y_i}(1|x_i,\theta^{(t)})}{w_1} - n$ and $\frac{\partial E}{\partial w_2} = \sum_i \frac{p_{Y_i}(2|x_i,\theta^{(t)})}{w_2} - n$ (although, we will be using w_1+$w_2 = 1$)

Taking partial derivative w.r.t $\mu_1$and μ_2 we get,

$\frac{\partial E}{\partial \mu_1} = \sum_i p_{Y_i}(1|x_i, \theta^{(t)}) \dfrac{x_i - \mu_1}{\sigma_1^2}$ and $\frac{\partial E}{\partial \mu_1} = \sum_i p_{Y_i}(2|x_i, \theta^{(t)}) \dfrac{x_i - \mu_2}{\sigma_2^2}$

Taking partial derivative w.r.t $\sigma_1$and σ_2 we get,

$\frac{\partial E}{\partial \sigma_1} = \sum_i p_{Y_i}(1|x_i, \theta^{(t)})[\dfrac{-1}{\sigma_1} + \dfrac{(x_i - \mu_1)^2}{\sigma_1^3}]$ and $\frac{\partial E}{\partial \sigma_1} = \displaystyle\sum_i p_{Y_i}(1|x_i, \theta^{(t)})[\dfrac{-1}{\sigma_2} + \dfrac{(x_i - \mu_2)^2}{\sigma_2^3}]$

This time we took partial derivatives w.r.t E although we need to minimize -E in algorithm to maximize E,

We write the following algorithm to achieve the same.

Algorithm:

w: a combination of all 6 parameters, g is gradient or partial derivative of objective function w.r.t each parameter.

```
1. Initialize $\eta$(learning rate), w, tolerance
2. w ← w$_t$, obj ← c(w)
3. while t is within max iteration limit:
    • temp ←w$_t$-$\eta$g
    • if c(w$_{t+1}$) <absolute(obj – tolerance))
        – w$_{t+1}$ = temp
    • else: break
```

4. increment t = t+1 until convergence
5. report final w

2. Implementation of Gradient Descent algorithm on mixture of 2 gaussians:

After initial struggle of setting correct learning rate for every parameter (similar approach as in problem 3), we proceed with comparing 2 different data sizes.

We will not consider data size of 10000, due to poor computational resources on this system.

Starting with data size of 100, first set the parameters as follows (with 30 test cases),

VALUES	**w1**	**w2**	**mu1**	**mu2**	**sigma1**	**sigma2**
TRUE	0.4	0.6	30	6	1	1.5
MEAN	0.399697217	0.600302783	29.32584141	6.087003033	1.09189187	1.820815372
STD DEV	0.045463232	0.045463232	0.039307267	0.164624315	0.081123609	0.048115918

And proceeded with data size of 1000, results as follows (30 test cases):

VALUES	**w1**	**w2**	**mu1**	**mu2**	**sigma1**	**sigma2**
TRUE	0.4	0.6	30	6	1	1.5
MEAN	0.403033272	0.596966728	29.82502286	6.031645442	1.055659191	1.567626023
STD DEV	0.017050552	0.017050552	0.046200094	0.058791139	0.043338934	0.013086139

Looking at above results, we can clearly see that standard deviation has decreased signficantly after increasing the data size.

One more thing to observe is convergence of mu1 and sigma2 improving with higher data size although this could also be due to poor learning rate with data size 100.

Detailed test results are included in the spreadsheet attached in the tab named “Problem5GD”.

3. Implementation of Expectation Maximization algorithm on mixture of 2 gaussians:

Good thing about EM algorithm is we need not worry about learning rate, as data size itself takes care of that.

Implementing the algorithm on data size of 100 we get following result:

VALUES	**w1**	**w2**	**mu1**	**mu2**	**sigma1**	**sigma2**
TRUE	0.4	0.6	30	6	1	1.5
MEAN	0.406333333	0.593666667	29.96238998	6.11529746	0.970221413	2.371688138
STD DEV	0.038994547	0.038994547	0.176337877	0.200850044	0.191420879	0.456428999

Next we run the same algorithm 30 times for data size of 1000, and get following results:

VALUES	**w1**	**w2**	**mu1**	**mu2**	**sigma1**	**sigma2**
TRUE	0.4	0.6	30	6	1	1.5
MEAN	0.401466667	0.598533333	29.99909633	5.996576047	1.002495544	2.212757453
STD DEV	0.012802837	0.012802837	0.046583966	0.058615921	0.085331424	0.114404102

As we already expected the standard deviation decreased as data size increased.

Comparison between Gradient Descent and Expectation-Maximization:

Both these algorithms give good estimate for data size of 1000 and are very close to true parameters. Interestingly, EM gives a much more closer result than the GD. Only one parameter not converging in EM is sigma2, which is most likely due to minor error in formula used in the code. EM algorithm seems to have much lower standard deviation for each parameter compared to GD. This can be a result of having readymade probabilities at each point in EM as opposed to trial and error convergence condition in case of GD. Most likely a GD with adaptive learning rate could perform better than our regular GD, although according to what we observe here, EM performs really well. Afterall, probabilities of each point is a guided way of convergence which is a better way to converge parameters to true values. One better thing about GD could be that, it will converge much faster than EM as break loop condition might reach quickly depending on initial parameter selection. GD will not be calculating probabilities at each point so clearly, complexity order is also simpler than EM. Also, we have seen in problem 3 that STD DEV for higher data sizes in GD is also low. So keeping in mind the resources , we may choose GD over EM. It is justice in saying that EM is accurate but GD is fast.