Large-Scale Stochastic EM-algorithm

Polina Kirichenko
Advisor: Dmitry P. Vetrov

Department of Computer Science,
National Research University Higher School of Economics

Abstract

Expectation-Maximization algorithm (EM-algorithm) is a popular method for finding maximum likelihood estimates in probabilistic models. It consists of two alternating steps: in E-step we take expectation of log likelihood and then maximize it in M-step, and at each step the whole dataset is used. For large datasets every step takes considerable time since it is linear in dataset size. The core idea in this work is that, analogously to Stochastic Gradient Descent (SGD), we can sample just one point or a small fixed size mini-batch from the dataset at each iteration and make a gradient step using them at M-step which could make EM-algorithm scalable and lead to faster convergence. This work is devoted to stochastic optimization in Expectation-Maximization algorithm, specifically to predicting parameters of Gaussian mixture models. We compare different stochastic optimization methods for EM and its incremental form.

Index terms — Expectation-Maximization algorithm, Stochastic Optimization, Machine Learning.

Contents

1 Introduction

One of the problems in unsupervised learning is data clustering — dividing points in n-dimensional space into groups so that points within one cluster would be more "similar" than points from different clusters according to a certain metric. In general, it is not known what the number k of different clusters is, and there exist various methods for finding optimal k, but in this work we assume that k is known.

A popular clustering algorithm k-means makes use of Euclidean distance as a metric and, starting from some initial state, iteratively recomputes cluster centers, aiming to minimize the within-cluster sum of squares. It assigns each data point to one and only one of the spherical clusters depending on which of the centers is the nearest. Another approach is to evaluate probabilities that a particular observation belongs to a cluster. Thus distribution-based clustering comprises a mixture of some distributions $\sum_{i=1}^{k} \pi_i p_i(x)$ (where $\sum_{i=1}^{k} \pi_i = 1$, $p_i(x)$ — the probability density function of component i) with parameter θ from which we observe our data, and in this case the goal is to estimate this unknown parameter. The subject of this research is Expectation-Maximization algorithm (further EM algorithm) which is used to find maximum likelihood estimation of parameters for models with latent variables. EM algorithm uses posterior probabilities of latent variables to maximize the expectation of log likelihood and guarantees to increase likelihood at every iteration. However, it converges only to a local optimum. It has broad applicability and is often used for Gaussian mixture models (further GMM). This particular case is covered in this work.

When applying EM algorithm to big data, evaluating all data points at every step may be computationally expensive, and, in fact, there may be no need to do so as information may be excessive. The underlying idea is to take random samples from the dataset at every iteration and make a small improvement of parameters using only this particular example.

Existing alternative methods of dealing with the linear time problem, such as Incremental EM (iEM), covered in [2], are applicable to a narrow class of models such that explicit formulas for sufficient statistics are available (e.g., exponential family). The approach proposed in this work does not suffer from this limitation as we utilize Stochastic Gradient Descent in M-step which can be performed directly for differentiable functions. However, an extra hyperparameter arises, which needs to be chosen and which may affect significantly the algorithm's performance.

We begin from a short theory overview, the description of EM-algorithm and its modifications and finally show results of experiments on synthetic datasets comparing different stochastic optimization approaches.

2 General EM algorithm

Consider a model where $\mathscr{P} = \{P_\theta | \theta \in \Theta\}$ is a family of distributions parametrized by θ. Let $X \sim \mathscr{L}(X) \in \mathscr{P}$ be a vector of observed i.i.d. variables and Z a vector of corresponding latent variables (here assume they are discrete, but the continuous case is analogous).

Our goal is to estimate the unknown parameter, thus we are interested in maximization of likelihood $p(X|\theta)$ with respect to θ. Due to the independence assumption we will work with the logarithm of likelihood as it breaks into the sum of likelihoods of individual data points. Log of marginal probability of X is $\ln p(X|\theta) = \ln \sum_Z p(X, Z|\theta)$ where Z ranges over all possible sequences of Z. Suppose that optimization of $p(X, Z|\theta)$ is straightforward, whereas maximization of incomplete likelihood $p(X|\theta)$ is computationally intractable because of the summation of distributions.

Given both X and Z we could compute the complete likelihood, but as Z is hidden we can only evaluate the expectation of $p(X, Z|\theta)$ instead using posterior probabilities $p(Z|X, \theta)$. It can be proved that if we maximize this complete likelihood, we also maximize the desired incomplete likelihood (see Appendix A). Steps of taking expectation of $p(X, Z|\theta)$ and its maximization are E and M steps of EM algorithm, respectively.

General EM:

1. Make an initial guess for θ.
2. **E-step**: Compute posterior probabilities $p(Z|X, \theta^{old})$ and expectation of log likelihood
$$\mathbb{E}_Z \ln p(X, Z|\theta) = Q(\theta, \theta^{old}) = \sum_Z p(Z|X, \theta^{old}) \ln p(X, Z|\theta).$$
3. **M-step**: $\theta^{new} = \arg\max_\theta Q(\theta, \theta^{old})$
4. Repeat E and M steps until convergence of log likelihood.

3 Gaussian mixture case

The density of the multivariate normal distribution in n-dimensional space is given by the following formula:

$$X \sim \mathcal{N}(\mu, \Sigma) = \left(\frac{1}{\sqrt{2\pi}}\right)^n \frac{1}{\sqrt{\det \Sigma}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

where $\mu = EX$ is a vector of means and $\Sigma = E(x-\mu)(x-\mu)^T$ is a covariance matrix.

A mixture of K Gaussians has a probability density function

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x|\mu_k, \Sigma_k), \qquad \sum_{k=1}^K \pi_k = 1,$$

where π_k, μ_k, Σ_k determine k-th Gaussian component and are its weight, mean and covariance, respectively.

Let X be a sample of size N of independent draws from Gaussian mixture. Then the log likelihood function takes the form

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left(\sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right).$$

In terms of the general EM algorithm discussed in the previous section the latent variable Z is Gaussian component responsible for a particular observation.

The maximum likelihood estimate in this case can be obtained as follows using EM algorithm. Denote by Z_{nk} an indicator that observation n is from k-th component. Then the complete likelihood is

$$\ln p(X, Z|\pi, \mu, \Sigma) = \sum_{n=1}^N \sum_{k=1}^K Z_{nk} \ln (\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k))$$

Its expectation with respect to Z is

$$\mathbb{E}_Z \ln p(X, Z|\pi, \mu, \Sigma) = \sum_{n=1}^N \sum_{k=1}^K \mathbb{E} Z_{nk} (\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k))$$

The expectation of indicator Z_{nk} is exactly the posterior probability that X_n comes from k-th component, which we will further call *responsibility* of k-th Gaussian for observation X_n:

$$\mathbb{E} Z_{nk} = P(Z_{nk} = 1) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_i \pi_i N(x_n|\mu_i, \Sigma_i)} = \gamma_{nk}.$$

$$\mathbb{E}_Z \ln p(X, Z|\pi, \mu, \Sigma) = \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} (\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)) = Q(\pi, \mu, \Sigma)$$

Denote $\sum_n \gamma_{nk}$ by N_k which is the expectation of the number of observations from the k-th Gaussian. Setting the derivatives of $Q(\pi, \mu, \Sigma)$ with respect to μ_k, Σ_k to zero we get

$$\frac{\partial Q}{\partial \mu_k} = \sum_{n=1}^{N} \gamma_{nk} \Sigma_k^{-1} (X_n - \mu_k) = 0 \quad \Rightarrow \quad \mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} X_n$$

$$\frac{\partial Q}{\partial \Sigma_k} = \frac{1}{2} \sum_{n=1}^{N} \gamma_{nk} \left(-\Sigma_k^{-1} + \Sigma_k^{-1} (x - \mu_k)(x - \mu_k) \Sigma_k^{-1} \right) = 0 \quad \Rightarrow \quad \Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (X_n - \mu_k)(X_n - \mu_k)^T$$

For π, we perform constrained maximization of $Q(\pi, \mu, \Sigma) + \lambda(\sum_k \pi_k - 1)$ as it has to be restricted:

$$\frac{1}{\pi_k} \sum_{n=1}^{N} \gamma_{nk} + \lambda = 0, \quad \sum_{k=1}^{K} \pi_k = 1$$

It is clear that $\sum_k \gamma_{nk} = 1$, consequently the first equation yields $\sum_k \sum_n \gamma_{nk} + \lambda \sum_k \pi_k = N + \lambda = 0$ and $\lambda = -N$, hence

$$\pi_k = \frac{\sum_n \gamma_{nk}}{N} = \frac{N_k}{N}$$

We can also obtain the same update formulas by simply taking derivatives of the incomplete log likelihood.

EM for Gaussian mixture:

1. Initialize $\pi_k, \mu_k, \Sigma_k \quad \forall k$.
2. **E-step** (compute posterior responsibilities):

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum\limits_i \pi_i N(x_n | \mu_i, \Sigma_i)}$$

3. **M-step**:

$$N_k = \sum_{n=1}^{N} \gamma_{nk}, \quad \pi_k = \frac{N_k}{N},$$
$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n,$$
$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

4 Incremental form

Various modifications of EM-algorithm have been proposed to avoid looking at the entire samples at each iteration. They involve a partial implementation of E or M step to immediately utilize information gained from a small subset of observations. Incremental EM [2] can take a form of iteratively updating sufficient statistics for parameters using either just one object or a small mini-batch. Specifically, it can be efficiently done for mixture models of exponential family.

Let A be a randomly chosen mini-batch to perform E and M steps. Having to process only this small mini-batch, we first compute posterior probabilities γ_{nk} for all k and all observations x_n from the mini-batch A. Remember that the re-estimation formulas for Gaussian mixture parameters are summing over observations, so we can update just the summands corresponding to data points from A. It will generally still result in an increase in the lower bound $\mathcal{L}(q, \theta)$ of log likelihood at each iteration (and in the GMM case, log likelihood itself increases as well), which is discussed in Appendix A. The old estimates have to be subtracted for performing an update, so we need to save the last values of γ_{nk} for all the sampled observations.

The re-estimation incremental formulas are the following (for a detailed derivation see Appendix B, but the basic idea is to divide the sum in each update formula into two components, one of which ranges over the mini-batch and the other over the rest of the dataset):

$$\pi_k^{new} = \frac{N_k^{new}}{N} \tag{1}$$

$$\mu_k^{new} = \mu_k^{old} + \frac{1}{N_k^{new}} \sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{old}) \tag{2}$$

$$\Sigma_k^{new} = \Sigma_k^{old} + \frac{1}{N_k^{new}} \Big(\sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old}) \big((x_n - \mu_k^{new})(x_n - \mu_k^{new})^T - \Sigma_k^{old} \big) + \\ + N_k^{old} (\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T \Big) \tag{3}$$

However, note that it is highly important that initial values of γ_{nk} are consistent with the initialization of μ and Σ for the following reason. Recall that for classic EM:

$$\mu_k^{new} = \frac{1}{N_k^{new}} \sum_{n=1}^{N} \gamma_{nk}^{new} x_n$$

If one sets mini-batch A to be the whole dataset, the incremental re-estimation is the same as the classic one:

$$\mu_k^{new} = \mu_k^{old} + \frac{1}{N_k^{new}} \sum_{n=1}^{N} (\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{old}) =$$

$$= \mu_k^{old} + \frac{1}{N_k^{new}} \sum_{n=1}^{N} \gamma_{nk}^{new} x_n - \frac{\mu_k^{old}}{N_k^{new}} \sum_{n=1}^{N} \gamma_{nk}^{new} - \frac{1}{N_k^{new}} \sum_{n=1}^{N} \gamma_{nk}^{old} (x_n - \mu_k^{old}) =$$

$$= \mu_k^{old} + \mu_k^{new} - \mu_k^{old} - \frac{1}{N_k^{new}} \left(\sum_{n=1}^{N} \gamma_{nk}^{old} x_n - \sum_{n=1}^{N} \gamma_{nk}^{old} \mu_k^{old} \right)$$

It means that γ_{nk}^{old} must satisfy the equation below:

$$\sum_{n=1}^{N} \gamma_{nk}^{old} x_n = \sum_{n=1}^{N} \gamma_{nk}^{old} \mu_k^{old} = N_k^{old} \mu_k^{old} \quad \text{or} \quad \mu_k^{old} = \frac{1}{N_k^{old}} \sum_{n=1}^{N} \gamma_{nk}^{old} x_n$$

To tackle this, we could perform a full E-step after the parameters initialization, but it would again demand going through the whole dataset. To avoid the full E-step, we initialize μ as k randomly chosen samples, then go through the rest of the dataset treating each sample as a new arriving data point and after the first epoch (one passage through dataset) use incremental updates stated above. This way we get that γ_{ik} for $i = 1 \dots k$ are consistent with starting values for mixture parameters as $\gamma_{ik} = \delta_{ik}$ (Kronecker delta).

Thus we can derive re-estimate formulas for online processing of m-th data point:

$$\sum_k N_k = m$$

$$N_k^{new} = N_k^{old} + \gamma_{mk}$$

$$\pi_k^{new} = \frac{N_k^{new}}{m} \tag{4}$$

$$\mu_k^{new} = \frac{\sum_{i=1}^{m-1} \gamma_{ik} x_i + \gamma_{mk} x_m}{N_k^{new}} = \mu_k^{old} + \frac{\gamma_{mk}^{new}}{N_k^{new}} (x_m - \mu_k^{old}) \tag{5}$$

$$\Sigma_k^{new} = \Sigma_k^{old} + \frac{1}{N_k^{new}} (\gamma_{mk}^{new} \big((x_m - \mu_k^{new})(x_m - \mu_k^{new})^T - \Sigma_k^{old} \big) + N_k^{old} (\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T) \tag{6}$$

Incremental EM

1. Initialize π, μ, Σ setting μ to k randomly chosen points.
2. For the rest of the dataset, successively compute γ_{nk} and update parameters with (4), (5), (6).
3. After one epoch, take a random mini-batch at each iteration and perform E-step computing corresponding γ_{nk} and M-step using (1), (2), (3).
4. Repeat subsequent E and M steps until convergence or for a fixed number of epochs.

5 Stochastic gradient approach

Though the incremental approach described above gives us scalability and faster convergence, it is applicable only in special cases when it is feasible to derive closed-form formulas for M-step re-estimates. Here we propose a more generic method for stochastic modification of the EM-algorithm. First, for an intuitive analogy consider the problem of finding a local extremum which can be solved using gradient descent algorithm.

In gradient descent algorithm our goal is to minimize function $Q(\theta)$ with respect to θ. We iteratively recompute θ making a step along antigradient as follows:

$$\theta^{t+1} = \theta^t - \eta \nabla Q(\theta^t) = \theta^t - \sum_{n=1}^{N} \nabla Q_n(\theta^t)$$

where η is a learning rate (the size of the step we make). Remark that $\nabla Q(\theta^t)$ usually sums over all data points. It again raises the issue of considering whole dataset before each update and motivates stochastic gradient descent. There, we approximate current gradient with a gradient on a single random object:

$$\theta^{t+1} = \theta^t - \eta \nabla Q_n(\theta^t)$$

We can proceed with this idea similarly within the EM-algorithm: with responsibilities γ_{nk} computed for mini-batch A in E-step, we make a step along the approximated gradient (as we maximize log likelihood) for GMM parameters:

$$\frac{\partial \ln p(X|\pi,\mu,\Sigma)}{\partial \mu_k} = \Sigma_k^{-1} \sum_{n \in A} \gamma_{nk}(x_n - \mu_k)$$

$$\frac{\partial \ln p(X|\pi,\mu,\Sigma)}{\partial \Sigma_k} = \sum_{n \in A} \frac{\gamma_{nk}}{2}(-\Sigma_k^{-1} + \Sigma_k^{-1}(x_n - \mu_k)(x_n - \mu_k)\Sigma_k^{-1})$$

$$\frac{\partial \ln p(X|\pi,\mu,\Sigma)}{\partial \pi_k} = \sum_{n \in A} \frac{\gamma_{nk}}{\pi_k}$$

However, note that Σ re-estimation formula contains matrix subtraction and thus we may lose the positive definiteness property (whereas for classic EM it can be easily shown that covariance matrices stay positive definite and symmetric after re-estimation). As well as in classic EM, another problem that may be caused by a covariance matrix is singularity when its determinant is too close to zero (in other words one of the clusters turns out to lie in the subspace of the current space). To overcome both issues we do the following: if a matrix appears to be nonpositive definite, we find its Jordan canonical form (which is diagonal and real-valued for symmetric matrices), then replace nonpositive and very small positive eigenvalues by $+\epsilon$, and finally perform the inverse transformation, so that the matrices are close in the matrix space.

6 Learning rate

For classic EM an important remark is that the algorithm is guaranteed to converge only to a local maximum depending on starting values for parameters. In stochastic gradient modification of EM a particular attention should be paid to a new hyperparameter — learning rate that determines the size of the gradient step. It also affects the speed of convergence as a too small choice for η would delay reaching an extremum, and, on the other hand, too big steps with η can result in skipping an extremum and divergence. It is quite intuitive that the more steps have been already made the smaller η should be as the algorithm is probably approaching the extremum and, therefore, the step size should be reduced in order not to miss it. The simplest choice for η would be $\frac{c}{t}$ or $\frac{c}{\sqrt{t}}$ at iteration t where c is a constant.

There exist several methods where we make use of the information about past gradients and parameter alterations. It is reasonable that if l^2-norm of a gradient is large, we should make a smaller step along it and vice versa. In the next paragraph, we review a few stochastic optimization approaches. Denote g_t as gradient at the t-th iteration.

Momentum It may happen that due to diversity of training examples gradient keeps on fluctuating along some dimensions from iteration to iteration, and this may yield slower convergence. In Momentum method this can be partially mitigated by taking into account previous gradients history as follows:

$$\Delta\theta_{t+1} = \alpha\Delta\theta_t + \beta g_t$$

where α and β are new hyperparameters. This way the progress is sustained for those direction where gradient keeps on pointing and is smoothed for the others.

Adagrad This method proceeds with an idea that the learning rate should be adapted according to gradient's norm and computes parameter change as:

$$\Delta\theta_t = \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2}} g_t$$

As we accumulate gradient norms, the learning rate continues to steadily decay throughout the training which may be unwanted. Moreover, if the large gradient appears at one of the first iterations, it will remain too small all through the training.

Adadelta Adadelta [4] is a more adaptive method for learning rate annealing. Firstly, instead of summing all past gradient norms, the RMS of norms is used, and therefore, a significant progress can be made later in learning as well as at the first iterations. Secondly, the Hessian matrix approximation is made using only first order information. Adadelta also appears to be not so sensitive to hyper parameters in practice comparing to other stochastic optimization methods which is a great benefit as well.

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2$$

$$\Delta\theta_t = \frac{\sqrt{E[\Delta x^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

$$E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x^2$$

However, the above techniques are usually used for non-convex and more complex optimization problems. In our case, the problem is convex at every single M-step, and it changes only slightly between iterations. Thus there may be a justification for using simpler methods like $\frac{c}{t}$ or $\frac{c}{\sqrt{t}}$ for this "almost convex" problem. We shall see that in practice they are comparable to non-convex stochastic optimization methods.

7 Experiments

We evaluate Stochastic Gradient approach testing its performance on synthetic datasets in dimensions 2, 3, 5, 10, 15, and 30 with 2-5 components in the mixture and 5000-100000 number of observations, running each experiment 20-30 times and taking the average for the results stability. It is predictable that the higher the dimension of the space is, the less accurate parameters estimates we obtain.

With learning rate specified as $\frac{c}{t}$ or $\frac{c}{\sqrt{t}}$, one needs first to find a constant c. The first 20-30 iterations were made with a constant learning rate c as otherwise there would be no significant changes in parameters after several iterations. For comparison, here we look at the mean error of predicted means (Euclidean distance from true centers to predicted), as it is a reasonable assumption that from the precise mean estimate follows precise covariance estimate as well.

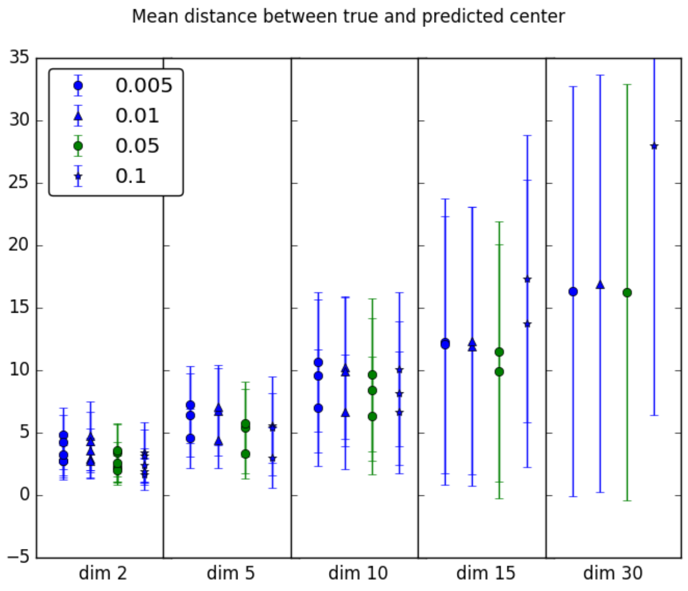

Figure 1: Comparison of stochastic EM algorithm performance with learning $\frac{c}{t}$ with different values for c. 1 epoch training with mini-batch size equal to 1.

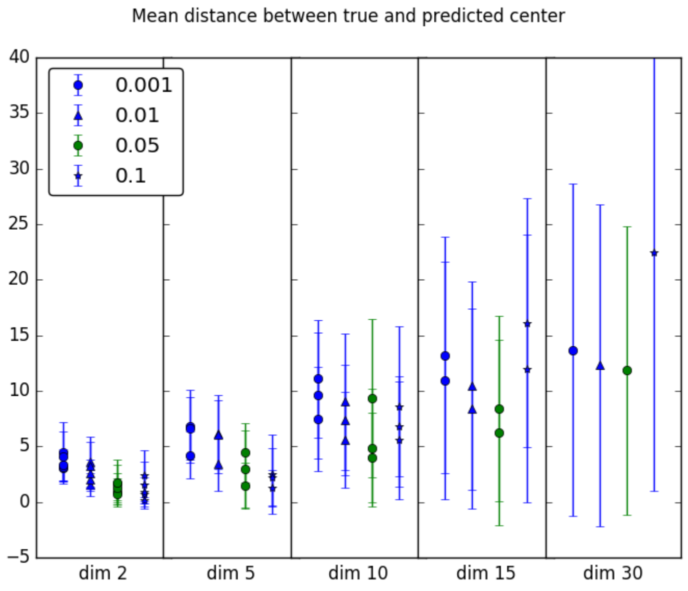

Figure 2: Comparison of stochastic EM algorithm performance with learning $\frac{c}{\sqrt{t}}$ with different values for c. 1 epoch training with mini-batch size equal to 1.

From figures 1 and 2 we can see that that though the prediction quality does not vary dramatically for different constants, the selection of proper constant is still quite subtle. Taking the best performing constants we compare convex optimization to Adadelta method and also to Incremental form of EM.

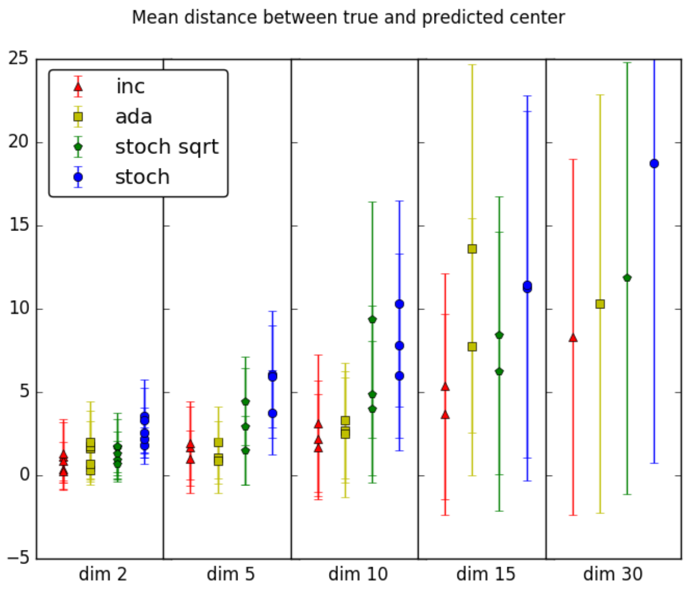

Figure 3: Comparison of Stochastic Gradient EM with learning rates $\frac{c}{t}$ and $\frac{c}{\sqrt{t}}$, Adadelta and incremental EM. 2 epochs training with mini-batch size equal to 1. Constants c are chosen as showing the best average quality in cross-validation, Adadelta hyperparameters are chosen as suggested in the original paper [4].

Analyzing average quality from figure 3 we may conclude that SGD EM with the learning rate $\frac{c}{\sqrt{t}}$ (assuming a proper constant is chosen) performs relatively well comparing to Adedelta. Incremental EM noticeably outperforms stochastic gradient EM but we again stress that it is not applicable to all the family of distributions.

To look more closely on SGD EM and iEM speed of convergence, we compare their mean distances to true mixture centers and mean log likelihood after going through 10, 30, 50 and 100% of sample, also looking at classic EM's behavior after the first two iterations (see Figure 4). It is noteworthy that in many cases after just one epoch both iEM and SGD EM converged to values at least as good as classic EM after two passages through sample. It is notable that quality of SGD EM and iEM are fairly close throughout the learning, although iEM convergence occurs significantly faster.

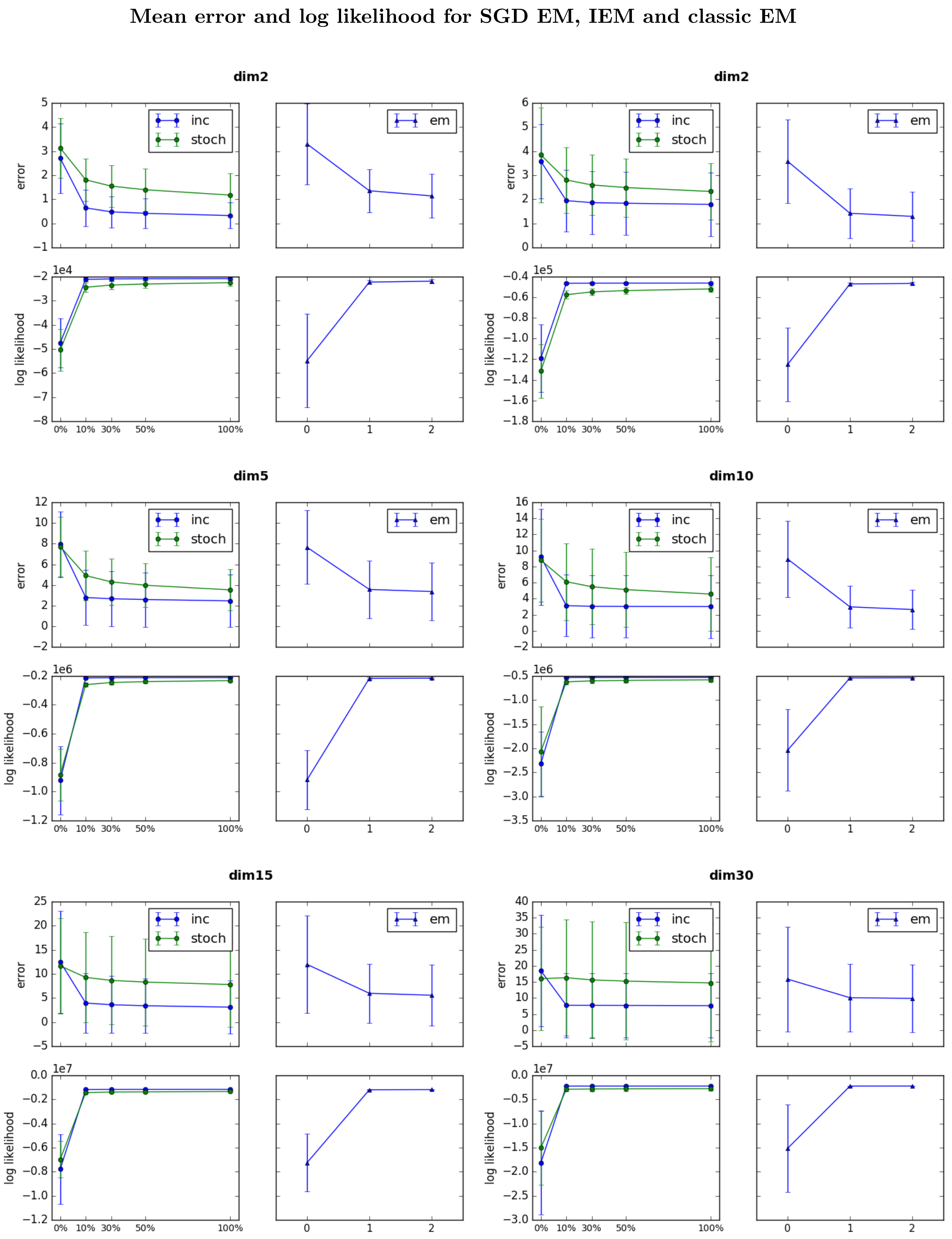

Figure 4: Each set of 4 pictures corresponds to a particular dataset. Picture's left column: mean error and log likelihood after viewing 10, 30, 50 and 100% of the dataset for Stochastic Gradient EM with learning rate $\frac{0.05}{\sqrt{t}}$ and Incremental EM. Picture's right column: mean error and log likelihood after 1 and 2 iterations of classic EM.

Practical issues For numerical stability in high dimensions and reduction in number of operations of SGD EM, we make a Cholesky decomposition for covariance matrices. If Σ is symmetric and positive definite (which is true for a covariance matrix), it can be represented as the following product:

$$\Sigma = LL^T$$

where L is a lower triangular matrix. It is clear that

$$\Sigma^{-1} = (LL^T)^{-1} = (L^T)^{-1}L^{-1}.$$

Let y be a solution to the equation $Ly = x - \mu$ which can be effectively found for triangular L. Then the summand of $\ln \mathcal{N}(x_n|\mu, \Sigma)$ takes form:

$$\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) = (Ly)^T(LL^T)^{-1}(Ly) = y^T L^T (L^T)^{-1}L^{-1}Ly = y^T y$$

The Σ gradient computation takes form:

$$-\Sigma_k^{-1} + \Sigma_k^{-1}(x_n - \mu_k)(x_n - \mu_k)\Sigma_k^{-1} = -(L^T)^{-1}L^{-1} + (L^T)^{-1}L^{-1}(Ly)(Ly)^T(L^T)^{-1}L^{-1} =$$
$$= -(L^T)^{-1}L^{-1} + L^T yy^T L^{-1} = (L^{-1})^T(-I + yy^T)L^{-1}$$

8 Conclusion

In this work we have studied the Expectation-Maximization algorithm and its non-batch modifications. We suggested an idea of using Stochastic Gradient Descent at M-step and compared such algorithm to incremental EM, an existing alternative to batch EM. Our results indicate that one may use the simple $\frac{c}{\sqrt{t}}$ specification for the learning rate on SGD instead of the much more sophisiticated Adadelta method without significant loss in prediction quality. SGD EM shows fast convergence and quality comparable to incremental EM. Among the non-batch EM modifications discussed in this work, incremental EM still showed better efficiency, but here a more generic method was proposed for dealing with the usage of the whole dataset at each step.

Appendices

A Proof of EM

To prove that EM algorithm indeed maximizes incomplete log likelihood we shall introduce a new term.

Definition. Kullback–Leibler divergence (KL-divergence) is a measure of the difference between two probability distributions P and Q, where P is often prior data distribution and Q is estimated approximation of P.

Generally $D_{KL}(P||Q) = E_P\left[\ln \frac{P}{Q}\right]$. When P and Q are discrete $D_{KL}(P||Q) = \sum_i p_i \ln \frac{p_i}{q_i}$, whereas for continuous case $D_{KL}(P||Q) = \int\limits_{\mathbb{R}} p(x) \ln \frac{p(x)}{q(x)} dx$.

Lemma. *For any probability distributions P and Q KL-divergence is non-negative: $D_{KL}(P||Q) \geq 0$ with equality iff $P = Q$ almost everywhere.*

Proof.

$$D_{KL}(P||Q) = E_p\left[\ln \frac{p_i}{q_i}\right] = E_p\left[-\ln \frac{q_i}{p_i}\right]$$

From Jensen's inequality for any convex $f(x)$ it is true that $Ef(x) \geq f(Ex)$. As $-\ln(x)$ is convex

$$E_p\left[-\ln \frac{q_i}{p_i}\right] \geq -\ln\left[E_p \frac{q_i}{p_i}\right] = -\ln\left[\sum_i p_i \frac{q_i}{p_i}\right] = -\ln\left[\sum_i q_i\right] = -\ln 1 = 0$$

Equality in Jensen's inequality takes place iff $f(x)$ is linear, this implies $D_{KL}(P||Q) = 0 \Leftrightarrow P = Q$ a.e. $\square$

Recall incomplete log likelihood $\ln p(X|\theta)$. From $p(X,Z|\theta) = p(Z|X,\theta)p(X|\theta)$ follows $\ln p(X|\theta) = \ln p(X,Z|\theta) - \ln p(Z|X,\theta)$.

$$\ln p(X|\theta) = \sum_Z q(Z) \ln p(X|\theta) = \sum_Z q(Z)\left(\ln p(X,Z|\theta) - \ln p(Z|X,\theta)\right) =$$

$$= \sum_Z q(Z)\left(\ln \frac{p(X,Z|\theta)}{q(Z)} - \ln \frac{p(Z|X,\theta)}{q(Z)}\right) = \sum_Z q(Z) \ln \frac{p(X,Z|\theta)}{q(Z)} - \sum_Z q(Z) \ln \frac{p(Z|X,\theta)}{q(Z)} =$$

$$= \sum_Z q(Z) \ln \frac{p(X,Z|\theta)}{q(Z)} + \sum_Z q(Z) \ln \frac{q(Z)}{p(Z|X,\theta)} = \mathcal{L}(q,\theta) + D_{KL}(q||p)$$

Note that the last summand is KL-divergence between prior distribution $q(Z)$ and posterior distribution $p(Z|X,\theta)$, so it is non-negative and hence $\ln p(X,Z|\theta) \geq \mathcal{L}(q,\theta)$.

1. Make an initial guess for $\theta = \theta^{old}$.

2. In the E-step we keep θ fixed and maximize the lower bound of $p(X|\theta)$ which is $\mathcal{L}(q,\theta) = p(X|\theta) - D_{KL}(q||p)$ with respect to q. As the first summand does not depend on q, the solution lies in setting $D_{KL}(q||p)$ to zero, which implies $q(Z) = p(Z|X,\theta^{old})$ a.e., so $\mathcal{L}(q,\theta) = p(X|\theta)$. Therefore

$$\mathcal{L}(q,\theta) = \sum_Z p(Z|X,\theta^{old}) \ln \frac{p(X,Z|\theta)}{p(Z|X,\theta^{old})} =$$

$$= \sum_Z p(Z|X,\theta^{old}) \ln p(X,Z|\theta) - \sum_Z p(Z|X,\theta^{old}) \ln p(Z|X,\theta^{old}) = Q(\theta,\theta^{old}) + const$$

3. In the M-step we keep $q(Z)$ fixed and find θ^{new} which maximizes $\mathcal{L}(q,\theta)$, in other words we maximize $Q(\theta,\theta^{old})$ with respect to θ. Since $D_{KL}(q||p) > 0$ ($p(Z|X,\theta^{new})$ no longer equals $q(Z)$), $p(X|\theta)$ will increase by more than $\mathcal{L}(q,\theta)$ does.

4. By reaching any local (global) maximum of $\mathcal{L}(q,\theta)$, we also reach local (global) maximum of $p(X|\theta)$.

This concludes the proof.

B Incremental updates

Recall the notations: $\gamma_{nk} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum\limits_i \pi_i \mathcal{N}(x_n|\mu_i,\Sigma_i)}$ – responsibility of k-th Gaussian for observation x_n and $N_k = \sum\limits_n \gamma_{nk}$ – expectation of number of observations from k-th Gaussian.

Let A be a mini-batch which we want to consider. Suppose we have computed γ_{nk} for all $x_n \in A$ and $k = 1 \dots K$. Then the parameters are updated as follows:

$$
\begin{aligned}
N_k^{old} &= \sum_n \gamma_{nk}^{old} \\
N_k^{new} &= \sum_{n \notin A} \gamma_{nk}^{old} + \sum_{n \in A} \gamma_{nk}^{new} \\
N_k^{new} &= N_k^{old} + \sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old}) \\
\mu_k^{old} &= \frac{1}{N_k^{old}} \sum_n \gamma_{nk}^{old} x_n \\
\mu_k^{new} &= \frac{1}{N_k^{new}} \big(\sum_{n \notin A} \gamma_{nk}^{old} x_n + \sum_{n \in A} \gamma_{nk}^{new} x_n \big) = \frac{1}{N_k^{new}} \big(N_k^{old} \mu_k^{old} - \sum_{n \in A} \gamma_{nk}^{old} x_n + \sum_{n \in A} \gamma_{nk}^{new} x_n \big) = \\
&= \frac{1}{N_k^{new}} \Big(\big(N_k^{new} - \sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old}) \big) \mu_k^{old} + \sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old}) x_n \Big) = \\
&= \mu_k^{old} + \frac{1}{N_k^{new}} \sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{old}) \\
\Sigma_k^{old} &= \frac{1}{N_k^{old}} \sum_n \gamma_{nk}^{old} (x_n - \mu_k^{old})(x_n - \mu_k^{old})^T \\
\Sigma_k^{new} &= \frac{1}{N_k^{new}} \big(\sum_{n \in A} \gamma_{nk}^{new} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T + \sum_{n \notin A} \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \big) = \\
&= \frac{1}{N_k^{new}} \big(\sum_{n \in A} \gamma_{nk}^{new} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T + \sum_n \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T - \\
&- \sum_{n \in A} \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \big) = \frac{1}{N_k^{new}} \big(\sum_{n \in A} (\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T + \\
&+ \sum_n \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \big)
\end{aligned}
$$

Let us take a more detailed look on the last summand:

$$
\begin{aligned}
&\sum \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T = \sum \gamma_{nk}^{old} \big((x_n - \mu_k^{old}) - (\mu_k^{new} - \mu_k^{old}) \big) \big((x_n - \mu_k^{old}) - (\mu_k^{new} - \mu_k^{old}) \big)^T = \\
&= \sum \gamma_{nk}^{old} (x_n - \mu_k^{old})(x_n - \mu_k^{old})^T + \sum \gamma_{nk}^{old} (\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T - \\
&\qquad - \sum \gamma_{nk}^{old} (x_n - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T - \sum \gamma_{nk}^{old} (\mu_k^{new} - \mu_k^{old})(x_n - \mu_k^{old})^T
\end{aligned}
$$

The last two summands are equal to zero since $\sum_n \gamma_{nk}^{old} (x_n - \mu_k^{old}) = \sum_n \gamma_{nk}^{old} x_n - \sum_n \gamma_{nk}^{old} \mu_k^{old} = N_k^{old} \mu_k^{old} - N_k^{old} \mu_k^{old} = 0$.

Therefore

$$
\sum \gamma_{nk}^{old} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T = N_k^{old} \Sigma_k^{old} + N_k^{old} (\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T
$$

Finally

$$\Sigma_k^{new} = \frac{1}{N_k^{new}}\big(\sum_{n\in A}(\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T + N_k^{old}\Sigma_k^{old} +$$

$$+ N_k^{old}(\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T\big) = \frac{1}{N_k^{new}}\big(\sum_{n\in A}(\gamma_{nk}^{new} - \gamma_{nk}^{old})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T +$$

$$+ \big(N_k^{new} - \sum_{n\in A}(\gamma_{nk}^{new} - \gamma_{nk}^{old})\big)\Sigma_k^{old} + N_k^{old}(\mu_k^{new} - s\mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T\big) =$$

$$= \Sigma_k^{old} + \frac{1}{N_k^{new}}\big(\sum_{n\in A}(\gamma_{nk}^{new} - \gamma_{nk}^{old})\left((x_n - \mu_k^{new})(x_n - \mu_k^{new})^T - \Sigma_k^{old}\right) + N_k^{old}(\mu_k^{new} - \mu_k^{old})(\mu_k^{new} - \mu_k^{old})^T\big)$$

References

[1] Bishop C. *Pattern Recognition and Machine Learning.* 2006.

[2] Neal R., Hinton G. *A view of the EM algorithm that justifies incremental, sparse, and other variants.* In Learning in Graphical Models, M. Jordan, Ed. Kluwer Academic Publishers, 1998.

[3] Arandjelovic O., Cipolla R., *Incremental learning of temporally-coherent models.* In: British Machine Vision Conference 2005.

[4] Zeiler M. D. *ADADELTA: An adaptive learning rate method.* 2012.