Automatic Differentiation Variational Inference (ADVI) for clustering taxi trajectories in Porto

Gabriel Ben Zenou
gabriel.benzenou@student-cs.fr
Quentin Mace
quentin.mace@student-cs.fr
Selvestrel Alexandre
alexandre.selvestrel@student-cs.fr

April 6, 2024

Introduction

All the code used for this project is available at this github page.

The article we're studying: Kucukelbir et al. [2016] presents ADVI (Automatic Differentiation Variational Inference). This is an algorithm capable of automating Bayesian inference in parametric statistical models. The approach used is variational. This means that the algorithm approximates the posterior probability of the statistical model parameters by a density that is easier to study (whose modes, moments, etc. are known). More precisely, the algorithm aims to minimize the Kullbach-Leibler divergence of the approximation with the real distribution. This minimization is achieved by gradient descent, where the gradients are calculated by automatic differencing (hence the name of the algorithm).

The advantage of this algorithm is that it enables Bayesian inference to be performed in a fairly general framework, without the need to rewrite dozens of lines of code each time the statistical model used is changed. What's more, the use of automatic differencing eliminates the need for tedious manual derivative calculations for gradient descent. This allows the statistician to focus more on his models, relieving him of some of the computational aspects of his work. For example, the moments and extrema of the posterior distribution can be accessed more quickly, thanks to the approximation obtained by ADVI.

1 ADVI Framework

1.1 Theoretical Framework

The aim of ADVI is to find a distribution $q(\theta)$ which approximates $p(\theta|x)$, so that $\mathrm{KL}(q(\theta)||p(\theta|x))$ is minimal. More precisely, we choose a parametric family of distributions $(q(\theta;\phi))$ and look for:

$$\arg\min_{\phi}(\mathrm{KL}(q(\theta;\phi)||p(\theta|x))) \tag{1}$$

To achieve this, ADVI maximizes the evidence lower bound (ELBO):

$$\arg\max_{\phi}(\mathbb{E}_{q(\theta,\phi)}[\log(p(x,\theta))] - \mathbb{E}_{q(\theta,\phi)}[\log(q(\theta;\phi))]) \tag{2}$$

This process takes place in three stages:

1.1.1 First step: Change of variables

First, we need to find a C^1 diffeomorphism $T : \mathrm{supp}(p(\theta|x)) \longrightarrow \mathbb{R}^K$. We choose T such that the new variable $\zeta = T(\theta)$ has $\mathbb{R}^K$ for support. Thanks to this transformation, with ADVI, we can work on $p(\zeta, x)$, which we can approximate by a Gaussian $q(\zeta;\phi)$. The switch to the variable $\zeta \in \mathbb{R}^K$ is important because, in the event of there being a non-zero measurement part of $\mathrm{supp}(q(\theta,\phi))$ not included in $\mathrm{supp}(p(\theta|x))$, we would systematically have $\mathrm{KL}(q(\theta;\theta)||p(\theta|x)) = \infty$: impossible to minimize.
We can link $p(\theta|x)$ and $p(\zeta|x)$ using the change-of-variables formula:

$$p(x,\zeta) = p(x, T^{-1}(\zeta))|Jac_{T^{-1}}(\zeta)| \tag{3}$$

(we call $Jac(f)$ the Jacobian of f, which is the determinant of the jacobian matrix)

1.1.2 Second step: Compute the evidence lower bound (ELBO)

Now, we need to define more precisely the nature of the family $q(\zeta;\phi)$ that we will use: a multivariate gaussian on $\mathbb{R}^K$ or K independant gaussians. The second case is called "mean field approximation". In that situation, the covariance matrix is diagonal, which is faster to compute but misses the correlations between the variables. However, because the mean field approximation is a particular case of the multivariate gaussian, in the following, we call $\Sigma = LL^T$ (with L lower-triangular: Cholesky factorization) the covariance matrix of the gaussian and μ its expectation. But we remember that in the mean field case, we only search μ and the diagonal coefficients of Σ, while in the multivariate case, we search μ and the best matrix L.

Now, the formula of the ELBO is:

$$\begin{aligned}\mathrm{ELBO} =& \mathbb{E}_{\zeta\sim q(\zeta;\phi)}[\log(p(x,\zeta)) + \log(|Jac_{T^{-1}}(\zeta)|)] \\ &+ \mathbb{H}[q(\zeta;\phi)] \quad \text{(where } \mathbb{H} \text{ means "entropy".)}\end{aligned} \tag{4}$$

However, we can't directly calculate the gradient with respect to ϕ of this expression by differentiating inside the expectations, as the expectations are taken over $q(\zeta,\phi)$, which itself depends on ϕ. To avoid this problem, we use the reparametrization trick.

It means that we set $\eta \sim \mathcal{N}(0, Id)$ and
$\zeta = S_\phi(\eta) = \mu + L\eta \sim \mathcal{N}(\mu,\Sigma)$, so the new expres-

sion for the ELBO is

$$\begin{aligned} \text{ELBO} = &\mathbb{E}_{\eta\sim\mathcal{N}(0,Id)}[\log(p(x, S_\phi^{-1}(\eta)) \\ &+ \log(|Jac_{T^{-1}}(S_\phi^{-1}(\eta)|)] + \mathbb{H}[q(\zeta;\phi)] \end{aligned} \tag{5}$$

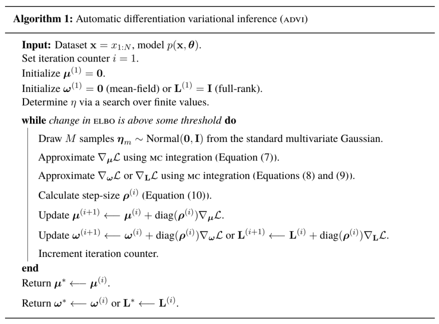
Now, in the first term of this sum, we can take the gradient inside the expectation (because the law of η is independent of ϕ). And the second term can be computed analytically because it is the entropy of a gaussian.

1.1.3 Third step:Stochastic gradient ascent of the ELBO

To perform the gradient ascent of the ELBO, we approximate the expectation term with Monte Carlo sampling. We then take the gradient over ϕ of the whole expression by using auomatic differenciation. This method to compute the gradients works because the law of η is independent of ϕ .

1.1.4 Code

To summarize, we give the pseudo code of the algorithm:



```
Algorithm 1: Automatic differentiation variational inference (ADVI)

Input: Dataset $\mathbf{x} = x_{1:N}$, model $p(\mathbf{x}, \boldsymbol{\theta})$.
Set iteration counter $i = 1$.
Initialize $\boldsymbol{\mu}^{(1)} = \mathbf{0}$.
Initialize $\boldsymbol{\omega}^{(1)} = \mathbf{0}$ (mean-field) or $\mathbf{L}^{(1)} = \mathbf{I}$ (full-rank).
Determine $\eta$ via a search over finite values.
while change in ELBO is above some threshold do
    Draw $M$ samples $\boldsymbol{\eta}_m \sim \text{Normal}(\mathbf{0}, \mathbf{I})$ from the standard multivariate Gaussian.
    Approximate $\nabla_{\boldsymbol{\mu}}\mathcal{L}$ using MC integration (Equation (7)).
    Approximate $\nabla_{\boldsymbol{\omega}}\mathcal{L}$ or $\nabla_{\mathbf{L}}\mathcal{L}$ using MC integration (Equations (8) and (9)).
    Calculate step-size $\boldsymbol{\rho}^{(i)}$ (Equation (10)).
    Update $\boldsymbol{\mu}^{(i+1)} \longleftarrow \boldsymbol{\mu}^{(i)} + \text{diag}(\boldsymbol{\rho}^{(i)})\nabla_{\boldsymbol{\mu}}\mathcal{L}$.
    Update $\boldsymbol{\omega}^{(i+1)} \longleftarrow \boldsymbol{\omega}^{(i)} + \text{diag}(\boldsymbol{\rho}^{(i)})\nabla_{\boldsymbol{\omega}}\mathcal{L}$ or $\mathbf{L}^{(i+1)} \longleftarrow \mathbf{L}^{(i)} + \text{diag}(\boldsymbol{\rho}^{(i)})\nabla_{\mathbf{L}}\mathcal{L}$.
    Increment iteration counter.
end
Return $\boldsymbol{\mu}^* \longleftarrow \boldsymbol{\mu}^{(i)}$.
Return $\boldsymbol{\omega}^* \longleftarrow \boldsymbol{\omega}^{(i)}$ or $\mathbf{L}^* \longleftarrow \mathbf{L}^{(i)}$.
```

Figure 1: ADVI algorithm

1.2 Implementation of ADVI

From Algorithm1, we implemented an ADVI function which can be used within two simple lines of code if the model class is well defined (see the following parts on PPCA with ARD and GMM):

```
advi =ADVI2(model, 1, batch_size=10,
               lr=0.1, mode='meanfield',
               num_epochs=50)
advi.fit(x_train, method='Adam',
          plotting=True)
```

2 Application to taxi trajectories in Porto

As in paper Kucukelbir et al. [2016], we apply ADVI to a specific case: clustering taxi routes in Porto. We collect the data from this dataset.

Two main processes are needed to process the data: a Probabilistic Principal Components Analysis (PPCA) with Automatic Relevance Determination (ARD allows the amount of latent dimensions to be unknown) and a Gaussian Mixture Model (GMM). After presenting how we implemented both of these tools with ADVI, we will detail the whole procedure for clustering the taxi trajectories.

2.1 PPCA with ARD

In order to cluster our taxi trajectories, we first need to reduce the dimensionality of the problem. For that we implement a probabilistic PCA model with Automatic Relevance Determination (ARD). The algorithm gives a probabilistic framework to implement PCA. Its parameters are as follow :

- $z \in \mathbb{R}^{N,M}$ the projected points in the latent dimension, with a $\mathcal{N}(0,1)$ prior (N the number of datapoints, M the reduced dimensionality.
- $\sigma \in \mathbb{R}_+$, with a $lognormal(0,1)$ prior
- $\alpha \in \mathbb{R}_+^M$, with an $InvGamma(1,1)$ prior (for ARD)
- $W \in \mathbb{R}^{D,M}$, with a $\mathcal{N}(0, \sigma diag(\alpha))$ prior on each row, D being the original dimension of the data

We then have $P(x_i|\theta) = \mathcal{N}(Wz_i, \sigma)$. We can optimize this using ADVI and that is what we did.

The transformations used to jump from the application space (where parameters live in the spaces defined above) to the latent space where parameters live in $\mathbf{R}$ are the following:

$$\begin{aligned} T_\alpha : \mathbf{R}_+^{\mathbf{M}} &\to \mathbf{R}^{\mathbf{M}} \\ \sigma &\mapsto log(\alpha) \end{aligned}$$

$$\begin{aligned} T_\sigma : \mathbf{R}_+^{\mathbf{K}} &\to \mathbf{R}^{\mathbf{K}} \\ \sigma &\mapsto log(\sigma) \end{aligned}$$

We tried PPCA on a toy dataset using the following command, and we show below the results :

```
model = PPCA_with_ARD_model(D, M)
```

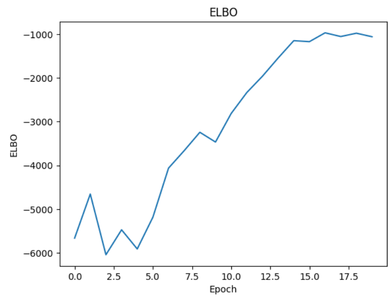


Figure 2: PPCA with ARD ELBO

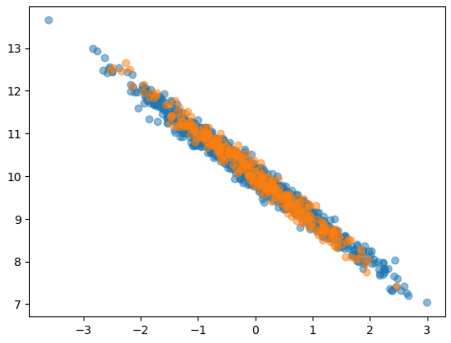


Figure 3: PPCA with ARD prediction

2.2 Gaussian Mixture Model

The second model to implement is a GMM. As what was done before, in order to run it with ADVI we needed to code a python class for this model with methods to compute the log probability of a point under a Gaussian mixture distribution, the log determinant of the jacobian inverse transformation and the inverse transformation from latent space to application space, as well as to sample points from a Gaussian mixture distribution.

The GMM is characterized by three sets of parameters: (for K being a hyperparameter of the model which defines the maximum amount of Gaussians to cluster by) $\pi \in [0,1]^K$ with $\sum_{k\in K} \pi_k = 1$, the weights of each Gaussian, $\mu \in \mathbf{R^K}$, the means of each Gaussian, and $\sigma \in \mathbf{R^K_+}$, the standards variation of each Gaussian.

We consider the following priors for these parameters: a lognormal prior for σ, a Gaussian prior for μ and a Dirichlet prior for π described by a hyperparameter α_0 (a larger value of α_0 encourages the model to use all K components).

The transformations used to jump from the application space (where parameters live in the spaces defined above) to the latent space where parameters live in $\mathbf{R}$ are the following:

$$\begin{aligned} T_\pi \colon \mathbf{P_K} &\to \mathbf{R^K} \\ \pi &\mapsto softmax(\pi) \end{aligned}$$

$$\begin{aligned} T_\sigma \colon \mathbf{R^K_+} &\to \mathbf{R^K} \\ \sigma &\mapsto log(\sigma) \end{aligned}$$

$$\begin{aligned} T_\mu \colon \mathbf{R^K} &\to \mathbf{R^K} \\ \mu &\mapsto \mu \end{aligned}$$

where $\mathbf{P_K} = \{x \in \mathbf{R^K} : \sum_{k\in K} x_k = 1\}$.

With these transformations and priors, all above mentioned computations are possible and implemented in python. The model can now be called with one line of code:

```
model = GMM(D, gmm_K, alpha_0=1000)
```

We tried ADVI with GMM on a toy dataset and got the results shown in Fig 4 and Fig 5.

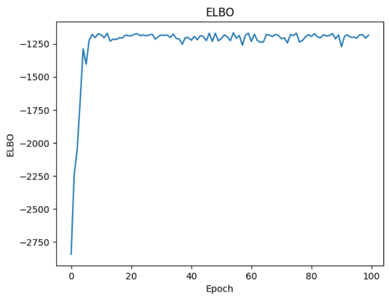


Figure 4: GMM with ADVI ELBO on toy data

2.3 Taxi Dataset

The dataset contains 1.7 million samples, which each sample being the list of 2D coordinates of one taxi ride. In order to work with fixed dimension data, we interpolate each sample into a 50 coordinates list. We then apply PPCA with ARD to the data in order to identify main components. The found subspace is of dimension 11. The whole data is projected onto this subspace. Eventually, the GMM is applied to identify 30 main components from the projected data. The results the paper managed to get are represented in Fig 6.

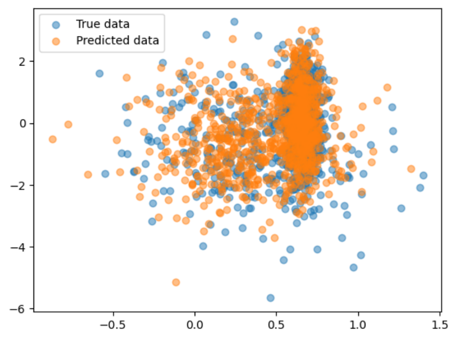


Figure 5: GMM with ADVI prediction on toy data

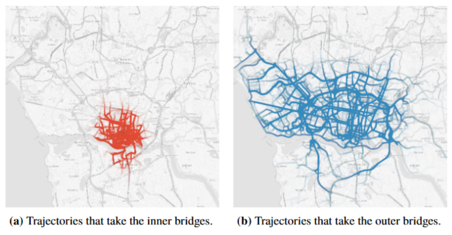

(a) Trajectories that take the inner bridges.
(b) Trajectories that take the outer bridges.

Figure 6: GMM with ADVI prediction from the original paper

Due to very high numerical unstabilty of our PPCA algorithm on the taxi dataset, we decided to apply a standard PCA on the dataset to reduce the dimensionality to 11 and then apply our gaussian mixture model. Also, for computational reasons, we only applied our algorithms to 10000 trajectories from the dataset. We computed the gaussian mixture model for 10 clusters, here is the cluster that is the most interpretable, the gaussian corresponding to the taxis that travel diagonaly from upper-left to bottom-right:

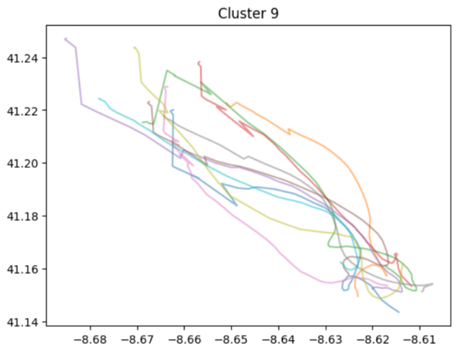


Figure 7: A cluster generated from a gaussian of our GMM

3 Conclusion

In this project, we implemented a working general framework for ADVI, compatible with any probabilistic model. We applied it to the Porto dataset, that was used in Kucukelbir et al. [2016]. For that pur-

pose we implemented probabilistic PCA and Gaussian mixture models.
However there is still room for improvement as we had troubles sclaing our algorithms to "real" data. Our probabilistic PCA is highly unstable for a reason that we couldn't find. We nevertheless manage to get reasonable results on the taxi dataset.

References

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference, 2016.