CS-E4820 Machine Learning: Advanced Probabilistic Methods (spring 2021)

Pekka Marttinen, Santosh Hiremath, Tianyu Cui, Yogesh Kumar, Zheyang Shen, Alexander Aushev, Khaoula El Mekkaoui, Shaoxiong Ji, Alexander Nikitin, Sebastiaan De Peuter, Joakim Järvinen.

Exercise 5, due on Tuesday February 23 at 23:00.

Problem 1: EM for missing observations

Suppose random variables X_i follow a bivariate normal distribution $X_i \sim \mathcal{N}_2(0, \Sigma)$,
where $\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$.

Suppose further that we have observations on $X_1 = (X_{11}, X_{12})^T$, $X_2 = (X_{21}, X_{22})^T$ and $X_3 = (X_{31}, X_{32})^T$, such that X_1 and X_3 are fully observed, and from X_2 we have observed only the second coordinate. Thus, our data matrix can be written as

$$\begin{bmatrix} x_{11} & x_{12} \\ ? & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

where the rows correspond to the transposed observations $\mathbf{x}_1^T, \mathbf{x}_2^T, \mathbf{x}_3^T$. Suppose we want to learn the unknown parameter ρ using the EM-algorithm. Denote the missing observation by Z and derive the E-step of the algorithm, i.e., **(a)** write the complete data log-likelihood $\ell(\rho)$, **(b)** compute the posterior distribution of the missing observation, given the observed variables and current estimate for ρ, and **(c)** evaluate the expectation of $\ell(\rho)$ with respect to the posterior distribution of the missing observations.

Hints:

1. In general, for $X \sim \mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $X = (X_1, X_2)^T$, $\boldsymbol{\mu} = (\mu_1, \mu_2)^T$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$, we have
$$X_1 \mid X_2 = x_2 \sim \mathcal{N}\left(\mu_1 + \frac{\sigma_1}{\sigma_2}\rho(x_2 - \mu_2), (1-\rho^2)\sigma_1^2\right),$$
with ρ being the correlation coefficient.
2. For evaluating the expectation of $\ell(\rho)$, you can make use of the following two rules:
   - $\mathbf{x_2}^T\boldsymbol{\Sigma}^{-1}\mathbf{x_2} = trace(\boldsymbol{\Sigma}^{-1}\mathbf{x_2}\mathbf{x_2^T})$.
   - if $X \sim \mathcal{N}(\mu, \sigma^2)$ then $\langle X^2 \rangle = \mu^2 + \sigma^2$.

Solution

(a) The complete data log likelihood is given by

$$\begin{aligned}\ell(\rho) &= \sum_{i=1}^{3} \log p(\mathbf{x}_i|\rho) \\ &= \sum_{i=1}^{3} -\frac{1}{2}\log\det(2\pi\boldsymbol{\Sigma}(\rho)) - \frac{1}{2}\mathbf{x}_i^T\boldsymbol{\Sigma}(\rho)^{-1}\mathbf{x}_i \\ &= -\frac{3}{2}\log\det(2\pi\boldsymbol{\Sigma}(\rho)) - \frac{1}{2}\sum_{i=1}^{3}\mathbf{x}_i^T\boldsymbol{\Sigma}(\rho)^{-1}\mathbf{x}_i\end{aligned}$$

where $\mathbf{x}_2 = (Z, x_{22})^T$ includes the missing observation Z (which is a random variable).

(b) The posterior distribution for the missing variable Z, given the current estimate ρ_0 of ρ, follows from a standard result for conditional distributions of bivariate Gaussians:

$$Z|X_{22} = x_{22} \sim \mathcal{N}(\rho_0 x_{22}, 1-\rho_0^2).$$

(c) For the expectation of $\ell(\rho)$, we note that only the term $-\frac{1}{2}\mathbf{x}_2^T\boldsymbol{\Sigma}(\rho)^{-1}\mathbf{x}_2$ involves Z. For the moment disregarding the leading coefficient, we compute the expectation of $\mathbf{x}_2^T\boldsymbol{\Sigma}(\rho)^{-1}\mathbf{x}_2$:

$$\begin{aligned}\left\langle \mathbf{x}_2^T\boldsymbol{\Sigma}^{-1}\mathbf{x}_2\right\rangle_{p(Z|x_{22},\rho_0)} &= \left\langle \operatorname{tr}(\mathbf{x}_2^T\boldsymbol{\Sigma}^{-1}\mathbf{x}_2)\right\rangle \\ &= \left\langle \operatorname{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{x}_2\mathbf{x}_2^T)\right\rangle \\ &= \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\left\langle \mathbf{x}_2\mathbf{x}_2^T\right\rangle\right) \\ &= \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\left\langle\begin{pmatrix} Z^2 & Zx_{22} \\ Zx_{22} & x_{22}^2\end{pmatrix}\right\rangle\right) \\ &= \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\begin{pmatrix}\rho_0^2x_{22}^2 + (1-\rho_0^2) & \rho_0x_{22}^2 \\ \rho_0x_{22}^2 & x_{22}^2\end{pmatrix}\right) \\ &= \operatorname{tr}\left(\frac{1}{1-\rho^2}\begin{pmatrix}1 & -\rho \\ -\rho & 1\end{pmatrix}\begin{pmatrix}\rho_0^2x_{22}^2 + (1-\rho_0^2) & \rho_0x_{22}^2 \\ \rho_0x_{22}^2 & x_{22}^2\end{pmatrix}\right) \\ &= \frac{1}{1-\rho^2}\left(\rho_0^2x_{22}^2 + (1-\rho_0^2) - 2\rho\rho_0x_{22}^2 + x_{22}^2\right).\end{aligned}$$

On the first line, we have used the (obvious) fact that the trace of a scalar is the scalar itself. On the second line, we have used the fact that the terms in a matrix product can be changed in cyclic order inside the trace. On the fifth line, we have used the fact that if $Z|X_{22} = x_{22} \sim \mathcal{N}(\rho_0x_{22}, 1-\rho_0^2)$ then $E\left[Z^2\right] = \rho_0^2x_{22}^2 + (1-\rho_0^2)$.

Other terms in the complete data log-likelihood $\ell(\rho)$ do not include Z, i.e. taking the expectation does nothing to them. The function $Q(\rho, \rho_0)$ now follows by replacing the second term in the sum of $\ell(\rho)$ derived in **(a)** with the expectation computed here (including the leading coefficient).

Problem 2: Extension of 'simple example' from the lecture

Suppose that we have N independent observations $x = (x_1, \dots, x_N)$ from a two-component mixture of univariate Gaussian distributions with unknown mixing co-efficients and unknown mean of the second component:

$$p(x_n \mid \theta, \tau) = (1-\tau)\mathcal{N}(x_n|0,1) + \tau\mathcal{N}(x_n \mid \theta, 1).$$

(a) Write down the complete data log-likelihood and derive the EM-algorithm for learning the maximum likelihood estimates for θ and τ.

(b) Simulate some data from the model ($N = 100$ samples) with the true values of parameters $\theta = 3$ and $\tau = 0.5$. Run your EM algorithm to see whether the learned parameters converge close to the true values (by e.g. just listing the estimates from a few iterations or plotting them). Use the code template below (after the answer cell) as a starting point.

HINT: The E and M steps for simple example.pdf from the lecture material looks as follows

```
# E-step: compute the responsibilities r2 for component 2
    r1_unnorm = scipy.stats.norm.pdf(x, 0, 1)
    r2_unnorm = scipy.stats.norm.pdf(x, theta_0, 1)
    r2 = r2_unnorm / (r1_unnorm + r2_unnorm)

# M-step: compute the parameter value that maximizes
# the expectation of the complete-data log-likelihood.
    theta[it] = sum(r2 * x) / sum(r2)
```

Solution

The complete data log-likelihood is given by

$$\begin{aligned}
\log p(x, z|\theta, \tau) &= \sum_n \log p(x_n, z_n|\theta, \tau) \\
&= \sum_n \log p(x_n|z_n, \theta) + \log p(z_n|\tau) \\
&= \sum_n z_{n1} \log \mathcal{N}(x_n|0,1) + z_{n1}\log(1-\tau) + z_{n2}\log\mathcal{N}(x_n|\theta,1) + z_{n2}\log\tau.
\end{aligned}$$

E-step

We compute the posterior distribution of the latent variables, given the current estimate (θ_0, τ_0) of (θ, τ):

$$\begin{aligned}
p(z_{n1} = 1|x_n, \theta_0, \tau_0) &\propto (1-\tau_0)\mathcal{N}(x_n|0,1) \\
p(z_{n2} = 1|x_n, \theta_0, \tau_0) &\propto \tau_0\mathcal{N}(x_n|\theta_0,1) \\
p(z_{n2} = 1|x_n, \theta_0, \tau_0) &= \frac{\tau_0\mathcal{N}(x_n|\theta_0,1)}{\tau_0\mathcal{N}(x_n|\theta_0,1) + (1-\tau_0)\mathcal{N}(x_n|0,1)} = \gamma(z_{n2}).
\end{aligned}$$

The expected complete-data log-likelihood over the posterior distribution of the latent variables is now

$$\begin{aligned} Q(\theta,\tau|\theta_0,\tau_0) &= E_{z|x,\theta_0,\tau_0}[\log(p(x,z|\theta,\tau)] \\ &= \sum_n \gamma(z_{n1}) \log \mathcal{N}(x_n|0,1) + \gamma(z_{n1}) \log(1-\tau) + \\ &\quad \gamma(z_{n2}) \log \mathcal{N}(x_n|\theta,1) + \gamma(z_{n2}) \log \tau. \end{aligned}$$

M-step

Maximizing for θ

$$\frac{d}{d\theta}Q = 0 \quad \Rightarrow \quad \theta = \frac{1}{N_2}\sum_n \gamma(z_{n2})x_n$$

is the same as in the case when $\tau = 0.5$, which was done in the lecture notes.

Maximizing for τ:

$$\begin{aligned} \frac{d}{d\tau}Q &= \sum_n \left[\frac{\gamma(z_{n2})}{\tau} - \frac{\gamma(z_{n1})}{1-\tau}\right] \\ &= \frac{N_2}{\tau} - \frac{N_1}{1-\tau} = 0, \end{aligned}$$

where we defined $N_k = \sum_n \gamma(z_{nk})$. The solution is then

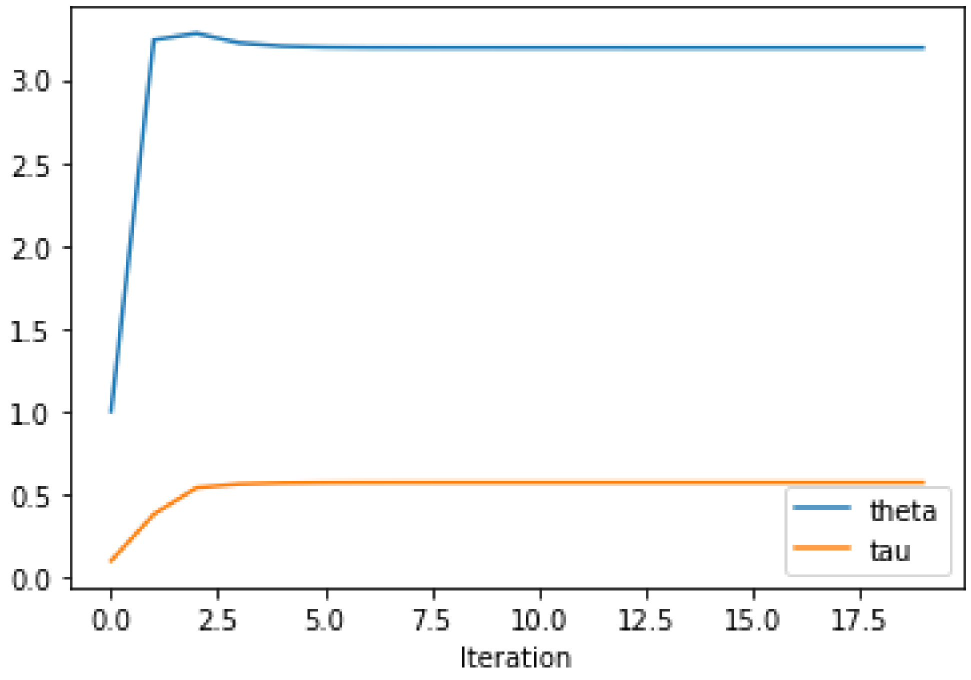
$$\tau = \frac{N_2}{N_1+N_2}$$

.

[1]:
```
# template for Problem 2(b)
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt


### Simulate data:

np.random.seed(0)

theta_true = 3
tau_true = 0.5
n_samples = 100

x = np.zeros(n_samples)
for i in range(n_samples):
# Sample from N(0,1) or N(theta_true,1)
if np.random.rand() < 1 - tau_true:
x[i] = np.random.normal(0, 1)
else:
x[i] = np.random.normal(theta_true, 1)
```

```
### The EM algorithm:

n_iter = 20
theta = np.zeros(n_iter)
tau = np.zeros(n_iter)

# Initial guesses for theta and tau
theta[0] = 1
tau[0] = 0.1

for it in range(1, n_iter):
# The current estimates for theta and tau,
# computed in the previous iteration
theta_0 = theta[it-1]
tau_0 = tau[it-1]

# E-step: compute the responsibilities r1 and r2
# r1 = ?
# r2 = ?

### BEGIN SOLUTION
r1_unnorm = (1 - tau_0) * scipy.stats.norm.pdf(x, 0, 1)
r2_unnorm = tau_0 * scipy.stats.norm.pdf(x, theta_0, 1)
r1 = r1_unnorm / (r1_unnorm + r2_unnorm)
r2 = 1 - r1
### END SOLUTION

# M-step: compute the parameter values that maximize
# the expectation of the complete-data log-likelihood.
# theta[it] = ?
# tau[it] = ?

### BEGIN SOLUTION
N_1 = sum(r1)
N_2 = sum(r2)
theta[it] = sum(r2 * x) / N_2
tau[it] = N_2 / (N_1 + N_2)
### END SOLUTION


# Print and plot the values of theta and tau in each iteration
print("theta       tau")
for theta_i, tau_i in zip(theta, tau):
print("{0:.7f}  {1:.7f}".format(theta_i, tau_i))
```

```
        plt.plot(range(n_iter), theta, label = 'theta')
        plt.plot(range(n_iter), tau, label = 'tau')
        plt.xlabel('Iteration')
        plt.legend()
        plt.show()
```



Problem 3: PyTorch

Go through the PyTorch tutorials in the three links and answer the questions given below

1) What is PyTorch: https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#sphx-glr-beginner-blitz-tensor-tutorial-py
2) Autograd: https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html#sphx-glr-beginner-blitz-autograd-tutorial-py
3) Linear regression with PyTorch: https://github.com/yunjey/pytorch-tutorial/blob/master/tutorials/01-basics/linear_regression/main.py

(a) What are PyTorch Tensors and how do you run a CPU tensor on GPU?

(b) What is Automatic differentiation and autograd?

(c) PyTorch constructs the computation graph dynamically as the operations are defined. In the 'linear regression with PyTorch' tutorial which line numbers indicates the completion of the computation graph, computation of the gradients and update of the weights, respectively?

Solution

(a) Tensors are similar to NumPy's ndarrays but can run on GPUs to accelerate computing. GPU utilization is via the `torch.cuda` package which is used to set up and run CUDA operations.

A tensor can be run on a GPU by allocating it on a GPU by setting the 'device' parameter declaration:

```
device = torch.device('cuda')
x = torch.tensor([1., 2.]), device=device)
```

or also by sending a (CPU) Tensor to to GPU device using the `.to()` or `.cuda()` methods:

```
x = torch.tensor([1., 2.]).cuda()

y = torch.tensor([3., 4.]).to(device=device)
```

(b) Automatic differentiation is a process of creating an algorithm to compute the derivatives of a function. This involves creating a computation graph of all the operations involved in evaluating a function so that the gradient of the function can be computed efficiently. Autograd is the PyTorch package that implements Automatic Differentiation. Note that, Automacic differentiation is different from Symbolic or numeric differentiation.

(c) Computation graph construction is complete at line 37. Line 41 computes the gradients and line 42 updates the weights.