MLPR Tutorial[1] Sheet 2

Reminders: Attempt the tutorial questions before your tutorial. Many of the best-performing students discuss the class and tutorial work with their peers during the week. You can seek clarifications and hints on the class forum. Full answers will be released after the tutorial.

This tutorial is largely just maths. While I ask for a small numerical experiment in the middle, there's no real data, and no machine learning. However, throughout the course we will derive models and algorithms that use multivariate Gaussian distributions. And other machine learning methods share some of the same maths. I've put this material on the tutorial, because it's useful stuff, and you need to work through it at your own pace. You'll get an assignment soon that involves some data!

---

1. **Warm-up exercise:**

   If $\mathbf{a}$ and $\mathbf{b}$ are $D\times 1$ column vectors and M is a $D\times D$ symmetric matrix, show that

   $$\mathbf{a}^\top M\mathbf{b} = \mathbf{b}^\top M\mathbf{a}.$$

   You wouldn't need to show this result in an exam unless you were explicitly asked to. In some working (like for the next question) you could just state that it's true for symmetric M.

2. **Identifying a Gaussian:**

   As part of a derivation, we may need to identify the probability density function of a vector up to a constant. For example:

   $$p(\mathbf{x}) \propto \exp\left(-\mathbf{x}^\top A\mathbf{x} - \mathbf{x}^\top\mathbf{c}\right),$$

   where A is a symmetric invertible matrix. As this distribution is proportional to the exponential of a quadratic in $\mathbf{x}$, it is a Gaussian: $p(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma)$.

   Identify which Gaussian $\mathbf{x}$ comes from by identifying the mean $\boldsymbol{\mu}$ and covariance Σ in terms of A and $\mathbf{c}$. The easiest method is to compare $p(\mathbf{x})$ to the standard form for the multivariate Gaussian PDF (given in class).

   The answer you should be able to show is:

   $$\Sigma = \frac{1}{2}A^{-1}, \quad \boldsymbol{\mu} = -\frac{1}{2}A^{-1}\mathbf{c}.$$

3. **Creating a 2D multivariate Gaussian, and a simple experiment:**

   The first element of a vector has $p(x_1) = \mathcal{N}(x_1; m, \sigma^2)$.

   A second element is generated according to the following process:

   $$x_2 = \alpha x_1 + \nu, \quad \nu \sim \mathcal{N}(0, n^2).$$

   Here x_2 depends on x_1, but the noise term ν is independent of x_1.

   Recall that a linear combination of Gaussian values is Gaussian distributed.

   a) The joint distribution of the vector $\mathbf{x} = [x_1\ x_2]^\top$ is Gaussian, and so takes the form $p(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma)$. Identify $\boldsymbol{\mu}$ and Σ.

   b) Turning to a computer: pick a setting for each of the parameters m, σ^2, α, and n, and simulate samples from the above process. Estimate the mean and covariance

1. Parts of this tutorial sheet are based on previous versions by Amos Storkey, Charles Sutton, and Chris Williams

from the samples. Do your estimates of the mean and covariance agree with their theoretical values?

Putting a standard error on your estimates of the means should be straightforward. You may have to use some creativity to put error bars on your estimates of the covariances.

4. **Sampling Gaussians, and matrix decompositions:**

[This question has quite a lot of detail on computation and linear algebra, which you can skim over on first reading. You don't actually need to understand the decompositions in detail, or know how to call them in Matlab or Python, to be able to answer some of the questions.]

In lectures we saw that we can sample from a multivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ by drawing a vector of standard normals, $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$, and setting $\mathbf{x} = A\boldsymbol{\nu}$, for a matrix A where $AA^\top = \Sigma$.

The lower-triangular Cholesky decomposition will decompose a symmetric positive-definite covariance into $\Sigma = LL^\top$.
Matlab/Octave: `L = chol(Sigma, 'lower');`
Numpy: `L = np.linalg.cholesky(Sigma)`
This decomposition can be used to draw samples, with $A = L$ above.

A triangular decomposition makes computing most things we might want to know about a covariance quick and easy. Cholesky decompositions are widely used. We can quickly find the determinant: $|L| = \prod_d L_{dd}$ where $|\Sigma| = |L|^2$, or more frequently $\log|L| = \sum_d \log L_{dd}$. We can also solve linear systems[2]: $L^{-1}\mathbf{b}$ takes similar time to a matrix-vector multiply $L\mathbf{b}$. In Matlab/Octave replace `inv(L)*b` with `L\b` and `inv(Sigma)*b` with `L'\(L\b)`. In Python use `scipy.linalg.solve_triangular`, and `scipy.linalg.cho_solve`.

a) Sometimes instead of decomposing the covariance matrix, we have the Cholesky decomposition of the precision matrix, $\Sigma^{-1} = CC^\top$, where C is lower-triangular. How would we use C to sample from $\mathcal{N}(\mathbf{0}, \Sigma)$?

b) Real symmetric matrices, like covariance matrices, also have a decomposition of the following form[3]:

$$\Sigma = Q\Lambda Q^\top,$$

where Λ is a diagonal matrix of eigenvalues, and the columns of Q are the eigenvectors of Σ.

i) Describe how to sample from $\mathcal{N}(\mathbf{0}, \Sigma)$ using this decomposition.

ii) Q is an orthogonal matrix, corresponding to a rigid rotation (and possibly a reflection). Describe geometrically (perhaps in 2D) how your sampling process transforms a cloud of points drawn from a standard normal.

c) *Yet another* possible decomposition is the principal square root[4]: $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$, where $\Sigma^{1/2}$ is symmetric. None of the decompositions discussed so far are the same. In this part we'll try to understand how they're related.

---

2. We do not usually evaluate an expression $A^{-1}\mathbf{c}$ by inverting A and then multiplying $\mathbf{c}$ by A^{-1}. There are faster and more numerically stable way to solve $A^{-1}\mathbf{c}$. The method you should use depends on the properties of A. In common situations, Matlab's `A\c` does something sensible and should be preferred to `inv(A)*c`. But if you've cached a decomposition of `A`, you should probably make use of it.
3. https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix#Real_symmetric_matrices
4. Non-examinable: $\Sigma^{1/2} = Q\Lambda^{1/2}Q^\top$, using Q and Λ from the eigendecomposition, and $\Lambda^{1/2}$ simply replaces each eigenvalue on the diagonal with its square root. However, it would be better to compute it with `sqrtm`, and you are unlikely to use it at all. I have only once found it useful.

i) Consider two different decompositions $\Sigma = AA^\top = BB^\top$. We'll assume the matrices are full rank so that we can write $B = AU$. Show that $UU^\top = \mathbb{I}$, the identity matrix, which means that U is an orthogonal matrix.

ii) Explain geometrically why if computing $A\boldsymbol{\nu}$ from $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ is a way to sample from $\mathcal{N}(\mathbf{0}, \Sigma)$, computing $B\boldsymbol{\nu} = AU\boldsymbol{\nu}$ will be as well.