Notes on Gaussian Processes

Compiled by D.Gueorguiev, 12/7/2024

# Introductory Notes

## Gaussian Models

The PDF of an MVN in dimensions is defined by the following:

(1)

where is the mean vector , and is the covariance matrix.

is known as the *precision matrix* or *concentration matrix*. The normalization constant ensures that the pdf integrates to 1.

A *full* *covariance matrix* has parameters since is symmetric. A *diagonal* *covariance* *matrix* has parameters, and has 0s in the off-diagonal terms. A spherical (or isotropic) covariance, , has one free parameter.

The expression inside the exponent is the Mahalanobis distance between a data vector and the mean vector .

Let us look more closely into the Mahalanobis distance by performing *eigendecomposition* of **.**

So we can write , where is an orthonormal matrix of eigenvectors satisfying , and is a diagonal matrix of eigenvalues. Thus we can write:

where is the th column of , containing the th eigenvector. Hence, we can rewrite the Mahalanobis distance as follows:

where .

Thus the contours of equal probability density of a Gaussian lie along ellipses. The eigenvectors determine the orientation of the ellipse, and the eigenvalues determine how elongated it is. In general the Mahalanobis distance corresponds to Euclidean distance in a transformed coordinate system , shifted by and rotated by .

**Theorem**: (MLE for a Gaussian)

If we have N iid samples , then the MLE for the parameters is given by

(1)

(2)

Proof:

We use the following notation and identities

– trace of a matrix

(3)

Here . We use the notation .

(4)

In order to prove (4) we use the following tensor notation:

(superscript corresponds to row vector, subscript - to column vector)

, (superscript-subscript combination on the same index implies summation)

( subscripts or superscripts on the same index do not imply summation along that index )

Thus we write

## Gaussian Processes (GP)

A Gaussian Process is a Gaussian distribution over functions:

## Noiseless Gaussian Process Regression

We observe a training set where

Given the test set of size , we want to predict the function outputs .

where is matrix, is matrix, and is

Here

Computing predictions is just a question of deriving the conditional distribution from the joint distribution.

# Appendix

## Covariance and Correlation

Covariance between two r.v.’s and measures the degree to which X and Y are related linearly. Covariance is defined as:

A group of symbols with numbers

Description automatically generated with medium confidence

Figure: several sets of points, with the correlation coefficient of and for each set. Note that the correlation reflects the noisiness and the direction of a linear relationship (top row) but not the slope of the relationship (middle), nor the shape indicating non-linear relationship (bottom). Source: <http://en.wikipedia.org/wiki/File:Correlation_examples.png>

If x is a d-dimensional vector, its covariance matrix is defined as the following symmetric positive definite matrix

The Pearson correlation coefficient between and is defined as:

The correlation matrix has the form

**Statement**:

Proof: We have to prove that (A.1)

Cauchy-Schwarz in : (A.2)

Clearly for discrete r.v.’s and (A.1) follows from (A.2).

For continuous r.v.’s and we use the Cauchy-Schwarz in

(A.3) where the integration is performed over some domain of .

A substitution of the distribution function of with and the distribution function of with in (A.3) yields (A1) for the case of continuous r.v.’s.

## Gaussian Process Regression Code

Nando de Freitas’s code for GP regression. It assumes a zero mean GP prior.

import numpy as np

import matplotlib.pyplot as pl

# This is the true unknown function we are trying to approximate

f = lambda x: np.sin(0.9\*x).flatten()

# Define the kernel

def kernel(a, b):

“”” GP squared exponential kernel “””

kernel\_parameter = 0.1

sqdist = np.sum(a\*\*2,1).reshape(-1,1) + np.sum(b\*\*2,1) – 2\*np.dot(a, b.T)

return np.exp(-.5 \* (1/kernel\_parameter) \* sqdist)

N = 10 # number of training points

n = 50 # number of test points

s = 0.00005 # noise variance

# Sample some input points and noisy versions of the function evaluated at these points

X = np.random.uniform(-5, 5, size=(N,1))

y = f(X) + s\*np.random.randn(N)

K = kernel(X, X)

L = np.linalg.cholesky(K + x\*np.eye(N))

# points we are going to make predictions at

Xtest = np.linspace(-5, 5, n).reshape(-1,1)

# compute the mean at our test points

Lk = np.linalg.solve(L, kernel(X, Xtest))

mu = np.dot(Lk.T, np.linalg.solve(L, y))

# compute the variance at our test points

K\_ = kernel(Xtest, Xtest)

s2 = np.diag(K\_) – np.sum(Lk\*\*2, axis=0)

s = np.sqrt(s2)

# plotting the results

pl.figure(1)

pl.clf()

pl.plot(X, y, ‘r+’, ms=20)

pl.plot(Xtest, f(Xtest), ‘b-‘)

pl.gca().fill\_between(Xtest.flat, mu-3\*s, mu+3\*s, color=”#dddddd”)

pl.plot(Xtest, mu, ‘r—-‘, lw=2)

pl.title(‘Mean predictions plus 3 standard deviations’)

pl.axis([-5, 5, -3, 3])

# draw samples from the prior at our test points

L = np.linalg.cholesky(K\_ + 1e-6\*np.eye(n))

f\_prior = np.dot(L, np.random.normal(size=(n,10)))

pl.figure(2)

pl.clf()

pl.plot(Xtest, f\_prior)

pl.title(‘Ten samples from the GP prior’)

pl.axis([-5, 5, -3, 3])

# draw samples from the posterior at our test points

L = np.linalg.cholesky(K\_ + 1e-6\*np.eye(n) – np.dot(Lk.T, Lk))

f\_post = mu.reshape(-1,1) + np.dot(L, np.random.normal(size=(n,10)))

pl.figure(3)

pl.clf()

pl.plot(Xtest, f\_post)

pl.title(‘Ten samples from the GP posterior’)

pl.axis([-5, 5, -3, 3])

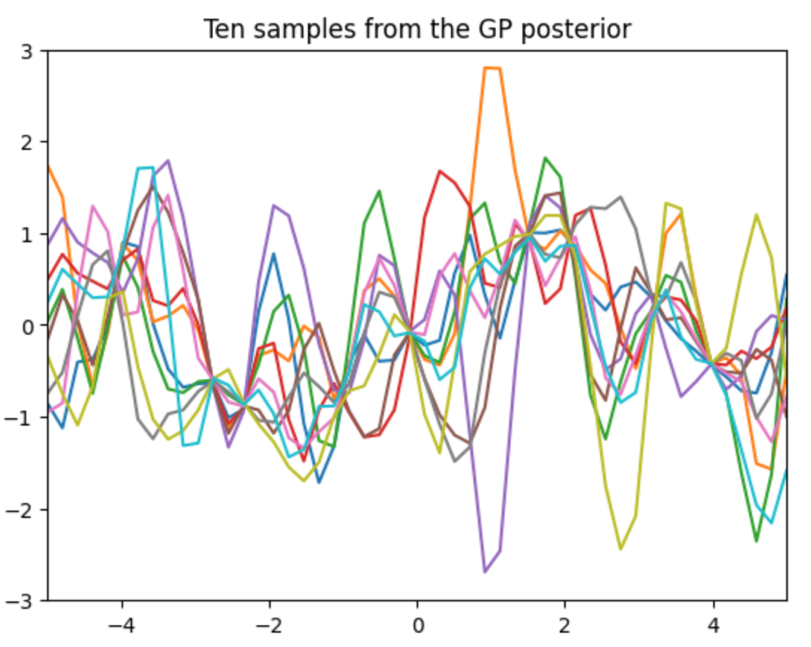
pl.show()

A graph with red lines and blue lines

Description automatically generated

A graph of colorful lines

Description automatically generated



# References

[1] [Gaussian Processes, Lecture by Nando Freitas, Feb 5th, 2013, UBC course CPSC 540-2013](https://youtu.be/MfHKW5z-OOA?si=QgoG1JPk40GsiXEI)

[2] [Introduction to Gaussian Processes, Lecture by Nando Freitas, Feb 4th, 2013, UBC course CPSC 540-213](https://youtu.be/4vGiHC35j9s?si=PZM9-E5xedToeWlA)

[3] [Introduction to Gaussian Processes part 1: Bayesian Linear Regression, Lecture by Stefan Harmeling, TU Dortmund, Jan 9th, 2023](https://youtu.be/148EUutsU8Q?si=Quh1V_pPAAJzvfMw)

[4] [Introduction to Gaussian Processes part 2: Implementation, Lecture by Stefan Harmeling, TU Dortmund, Jan 11th, 2023](https://youtu.be/wyCj9y1dFFY?si=j1q_oz13OlctTX4K)

[5] [Introduction to Gaussian Processes part 3: Kernel Design, Model Selection, GP Classification, Laplace Approximation, Lecture by Stefan Harmeling, TU Dortmund, Jan 16th, 2023](https://youtu.be/LFu7DAJcGKI?si=1hxx01IlX4ssJNBg)

[6] [Machine Learning Tutorial at Imperial College London: Gaussian Processes, Richard Turner, U of Cambridge, Nov 23th, 2016](https://youtu.be/92-98SYOdlY?si=K6cMS8998JFbpMi9)

[7] [Machine Learning: Probabilistic Perspective, Kevin P. Murphy, 2012](https://github.com/dimitarpg13/probabilistic_machine_learning/blob/main/books/MachineLearningProbabilisticPerspective.pdf)