**Plan**

1. General idea of the topic
2. Multivariate Gaussian distribution
   1. Gaussian distribution
   2. Multivariate Gaussian distribution
3. Gaussian processes
   1. Matlab example
   2. Example with conditions
4. Understand the code
   1. List of tools used in the code
   2. Learn Python
   3. Try to rewrite the code in Python

**General idea**

Input data: video data (high-dimensional time series)

Output data: generated video data

**Steps:**

1. **Dimensionality reduction**

Dimensionality: N x M x F, where

N x M – resolution of a video, F – number of frames

For instance: 300x200 with 250 frames (25 frames per second)

***Apply reduction algorithm, choose top 20 features***

As an algorithm it can be linear (like PCA) or nonlinear

Linear algorithms cannot capture complex dynamic textures

Some nonlinear algorithms produce irreversible mapping

Some nonlinear algorithms produce different coordinate systems

**Find an algorithm which is free of these weak points**

* Infer reduction function using Gaussian Process

1. **Dynamical texture modeling**

Learn dynamic texture

It cannot be linear, most of dynamic textures are not linear

It can be switching or piecewise linear, but not for all dynamic textures

**Find a more flexible model**

* Dynamic texture can be modeled using first-order Markov model based on Gaussian Process

1. **Dynamical texture synthesis**

Generate new video data using learned dynamic texture

Estimate necessary parameters (latent variable vector, observed dynamic texture vector, kernel matrix mapping hyperparameter, weights for kernel functions and different kernel parameters), then predict new sequence of dynamic textures

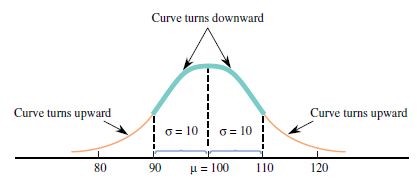
It is necessary to have a good performance at this step

**Adopt mean-prediction method**

* Based on first-order Markov model using Gaussian prediction

**Gaussian distribution**

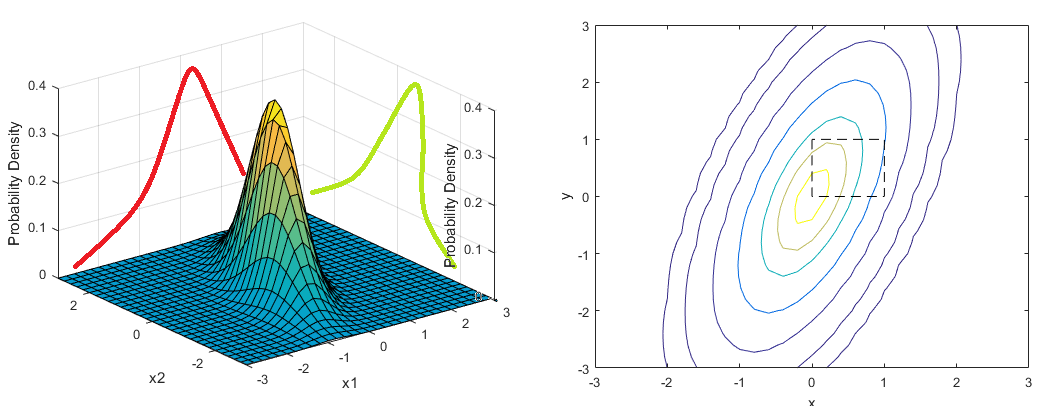
Gaussian (normal) distribution – continuous probability distribution, bell shaped and symmetric. Characterized by mean and standard deviation. Total area under the distribution curve equals to 1.



Mean – describes where corresponding curve is centered.

Standard deviation – describes how much the curve spreads out around center.

Joint distribution of two dependent/independent events

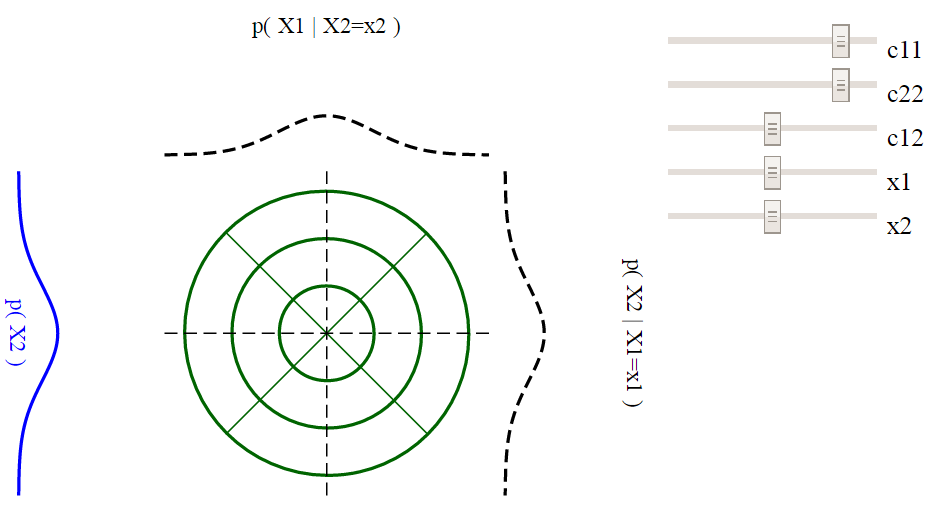


**Multivariate Gaussian distribution**

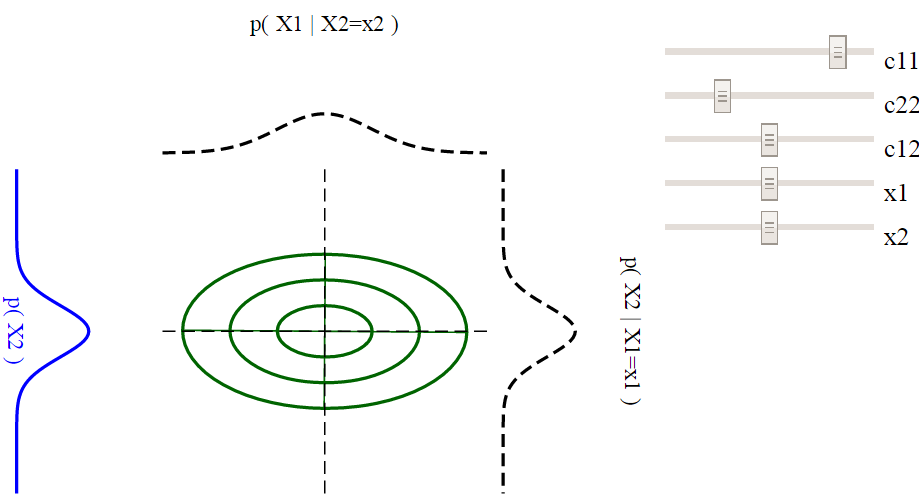
Bivariate case – for any fixed X1 value the distribution of associated X2 values is normal and for any fixed X2 value the distribution of X1 value is normal.

Multivariate case (2 or more dimensions) – characterized by:

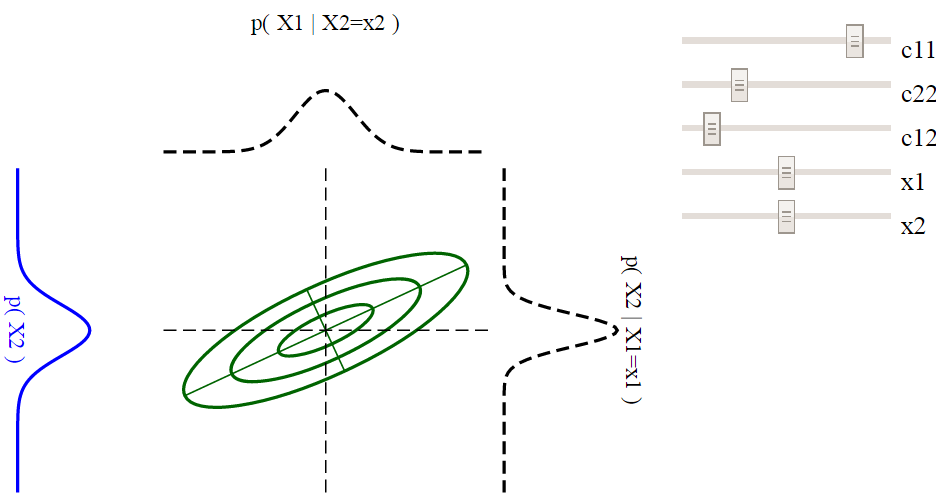
1. Mean vector – the same size as data
2. Covariance matrix – squared matrix DxD, where D – dimensionality (in bivariate case 2x2)
3. Shape of a cut (2D projection) – diagonal elements of covariance matrix are equal, sizes of directions are the same – contour is spherical



otherwise contour is shaped



also, it can be rotated as well

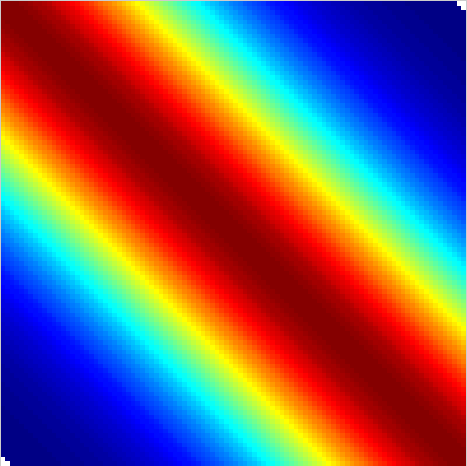


size and orientation of ellipse can be understood by looking at eigenvalues and eigenvectors of covariance matrix.

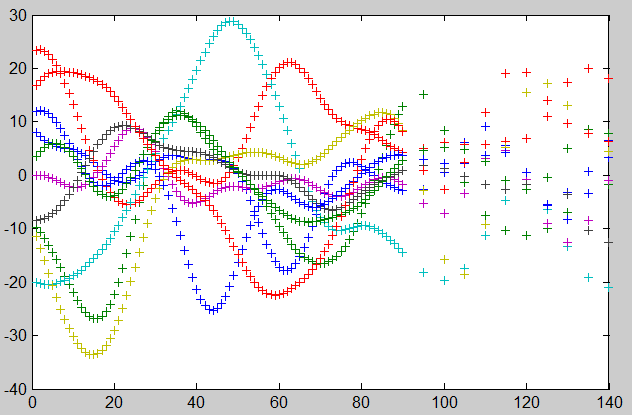
Eigenvectors shows directions, eigenvalues – scale.

**Matlab example**

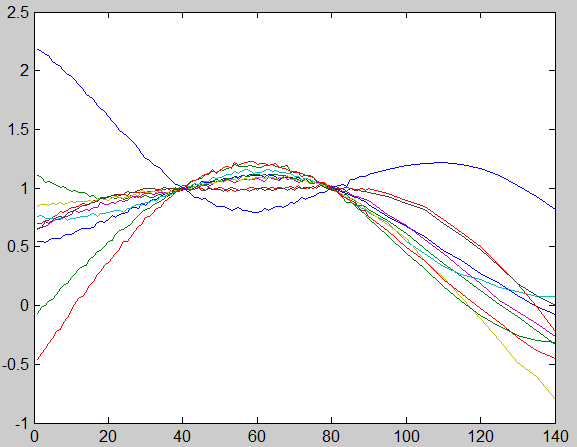
Kernel matrix



10 possible realisations with respect to this kernel matrix



**Example of conditional distribution (with known parameters)**



**List of tools used in the code**

*Preprocessing:*

1. Open a video file

2. Extract frames (images)

3. Resize frames (images)

*Processing:*

1. Convert image to grayscale

2. PCA

3. Matrix multiplacation

4. Kernels: linear, RBF (Radial Basis Function), Polynomial, RATQUAD, MLP (Multilayer Perceptron), Matern32

5. Creation of a kernel

6. Function initKernWeight for combined kernels defenition

7. Function weightsConstrain for hyperparameters initialization

8. Function kernExpandParam for combined kernel structure definition

9. Function updateKernWeight

10. Hyperparameters optimization function gpdm

11. Prediction and sample reconstruction

12. floor, ceil

13. reshape an array into the matrix

14. Save data back to video file

**Second meeting:** What have done in last 2 weeks?

1. Git
2. Course on Software carpentry
3. Jupyter Notebook
4. Reimplementation in Python

* Univariate Gaussian and joint distribution
* Multivariate Gaussian (library function, own function, conditional case, noisy kernel)

1. Model review in MD and Latex
2. OpenCV and preprocessing stage

**Questions:**

1. Can we use different kernel functions in conditional case?

In our setting – no.

1. How to rewrite in matrix form?

Just do it. Python uses broadcasting. However, be careful.

1. What is the best way to use matrices in Python?

Arrays. However, there is another class called *matrix* in numpy, sometimes it is more convenient to use arrays. I should be careful with using different classes at the same time.

1. Why do they use PCA?

They use it for initialization. Read the Algorithm in the paper.

**Following plan**

How to estimate missing frame?

* How to represent missing frame in original set Y? - go inside PCA and think (defend something).
* Theoretically, we'll find x related to missing y from PCA, but it is not a main point. We need to know Y for optimization and for prediction as well => understand mk\_gpdm, mk\_prediction - more important than PCA.
* I need to find SCG optimization function if I still want to reimplement the code in Python. Update the list of functions I need for Python as well.

**Next meeting:** What have done in last 2 weeks?

1. General learning algorithm understanding
2. Exact functions and gradients to be optimized and dimensionality analysis
3. Ideas of how to represent missing frames in original dataset Y
4. Little help about functions used in original code
5. Pre-processing function to make a video from rendered set of frames
6. Starting of code reimplementation using Python

Questions:

1. How to estimate one missing frame?

* Use N frames in X while use N-1 only for Y

1. How to compute Ky in matrix form?

* Reshape from N x Q to N x 1 x Q and to sum over last axis

1. Why I did't receive money?

* Check again and contact Marianne

**Next meeting:** What have done in last 3 weeks?

1. General learning algorithm understanding
2. Exact functions and gradients to be optimized and dimensionality analysis
3. How to represent and estimate missing frames
4. Starting of code reimplementation using Python
5. Gradient test module

Questions:

1. Gradient of Kronecker delta (in Ky and Kx)?
2. Gradient of Kx wrt lambda?
3. Gradient of Ky wrt theta (=Ky with theta=[1,1,1]?)
4. Problems with Poly kernel
5. Gradient of function 2 wrt W - what is the dimensionality?
6. Difference between simple GPLVM and our model?
7. Video generator module

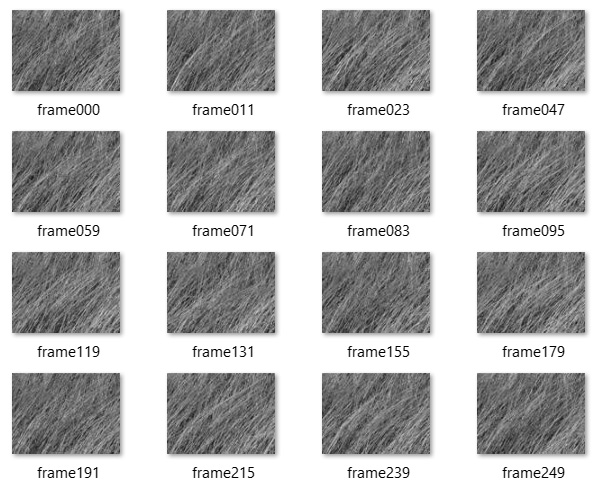
**Last meeting:**

1. Check is there initialization of a random generator in the code

* Was not at all. Now it is.

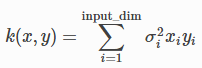
2. Try to use all the kernels separately and compare weights and latent variables

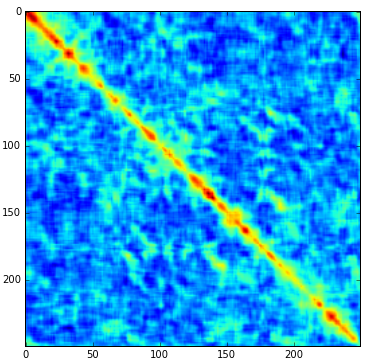
Original sample video **straw.avi** from Dyntex database preprocessed to grayscale 120x90 pixels with 10 seconds length (250 frames):



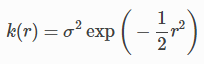
Kernels used:

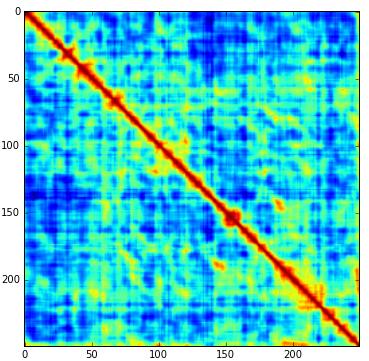
1. **Linear**





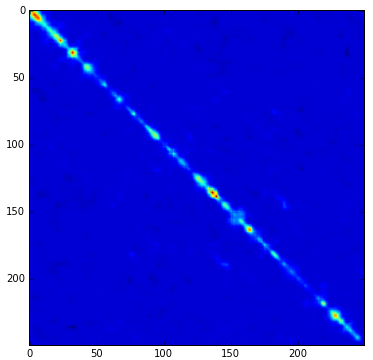
1. **RBF -** Radial Basis Function kernel



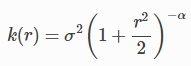


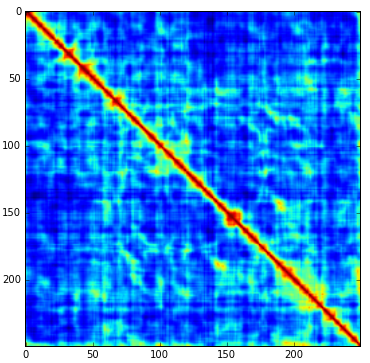
1. **Poly -** Polynomial kernel



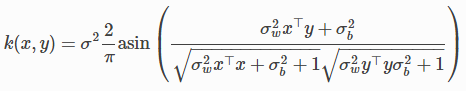


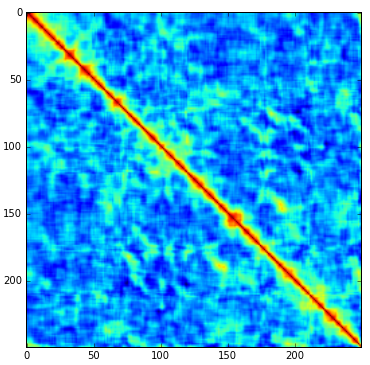
1. **RatQuad – Rational Quadratic kernel**

****

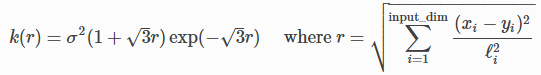


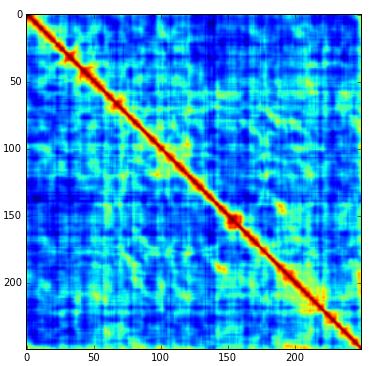
1. **MLP -** Multi layer perceptron kernel





1. **Matern32** - Matern 3/2 kernel:





Weights used in a given video:

Kernels used separately:

Weights used in different videos:

3. Plot X wrt to time and try to see some dependancies (like loop etc)

4. Missing frames estimation trow conditions (if it is not possible why?)