MAX PLANCK INSTITUTE FOR INTELLIGENT SYSTEMS

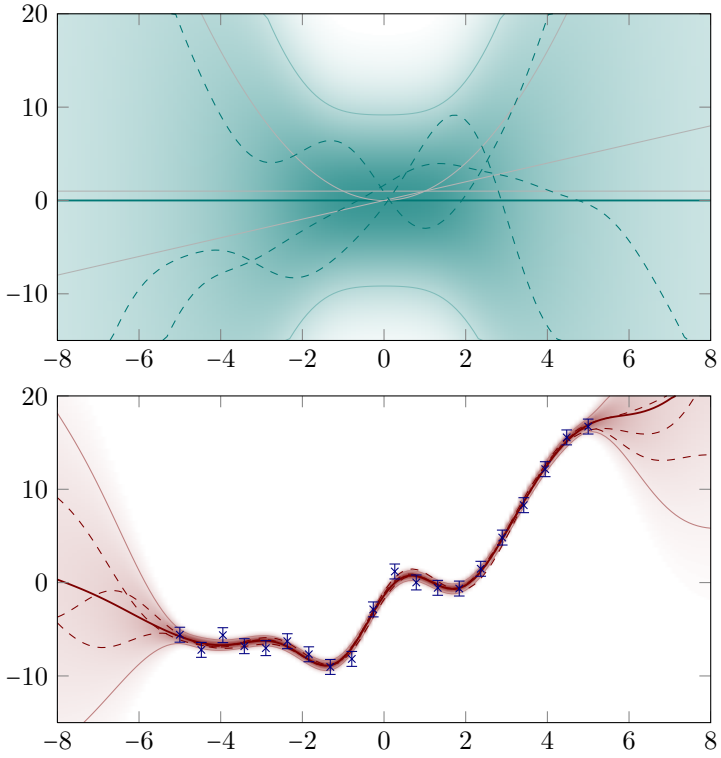
Technical Report No. 8 12 September 2013

ANIMATING SAMPLES FROM GAUSSIAN DISTRIBUTIONS — TECHNICAL NOTE —

Philipp Hennig
Max Planck Institute for Intelligent Systems
Department of Empirical Inference
Spemannstraße 38, 72070 Tübingen, Germany
phennig@tue.mpg.de

Abstract

The animations displayed below are animated samples from correlated Gaussian beliefs, following closed trajectories along equipotential lines of the probability distribution. They offer a more expressive view of the structure of samples from Gaussian processes than static samples. This document explains how to generate them, using Matlab, tikz, and LaTeX, in that order. If you do not see two animations with wobbly lines on the first proper page of this document, try opening it with Adobe Reader.



1 Introduction

There are two main reasons to use animations in slide presentations. The first one is that the human mind likes seeing animations in presentations. They attract our attention. The second one is that time offers a third dimension that can help display information beyond the two spatial dimensions of the projection screen. This document explains how to make smooth, looping animations of Gaussian distributions. The procedure is mathematically well founded, and can be applied to every set of jointly Gaussian variables, including function values drawn from Gaussian process priors, matrix or vector-valued operators drawn from multi-output Gaussian processes, and even structured outputs (though I have not yet tried this).
This text is a short technical note on how to generate these kind of animations. Their main value is educational. This technical document is intended to inform the community about relevant implementation details for this relatively simple idea, and to provide a citeable document.

1.1 Prior work

While preparing this document, it came to my attention that a 1991 paper by John Skilling[1] mentions a related idea, which gives a continuously changing animated plot that is perhaps better suited for the analysis of Gaussian models on a computer display (as opposed to the presentation of a model on a projected slide).

1.2 How to cite this document

If you are using the code in this document to construct animated slides, I would be grateful for a citation of

- Philipp Hennig
  Animating Samples from Gaussian Distributions.
  Technical Report No. 8 of the Max Planck Institute for Intelligent Systems.
  September 2013

BibTex:

```
@TechReport{hennig-tr-is-8,
  author      = {Philipp Hennig},
  title       = {Animating Samples from Gaussian Distributions},
  institution = {Max Planck Institute for Intelligent Systems},
  year        = 2013,
  type        = {Technical Report},
  number      = 8,
  address     = {Spemannstra{\ss}e, 72076 T{\"u}bingen, Germany},
  month       = {September}
}
```

[1] J. Skilling. *Bayesian solution of ordinary differential equations.* in C.R. Smith et al (eds.), Maximum Entropy and Bayesian Methods, Seattle, 1991.

1.2.1 License

The code quoted in this document is provided by the author "as is" without any expressed or implied warranties, and without any copyright, as far as the code written by the author is concerned (each of the used packages has their own copyright, see the corresponding webpages of the individual projects for more information). So everyone is free to change and adapt the code as they see fit. I would be grateful if you could mention the name of the original author in changed code, but there is no legal requirement to do so. If you are using the ideas described herein to generate animations in your presentation, I would be grateful if you could cite this document in the way mentioned above.

2 Mathematical Background

Assume we want to display samples from a joint Gaussian distribution $\mathcal{N}(x;\mu,\Sigma)$ over an N-dimensional vector $x \in \mathbb{R}^N$ (this includes matrices of size $M \times T$, stacked into a vector of length $N = MT$, tensors of higher order, etc). The widely known numerically stable way of drawing such samples $s \sim \mathcal{N}(x;\mu,\Sigma)$ is

1. construct the Cholesky decomposition R of Σ (that is, an upper triangular matrix such that $R^\top R = \Sigma$)
2. draw a vector $u \in \mathbb{R}^N$ from the standard Gaussian distribution

$$p(u) = \prod_i^N \mathcal{N}(u_i;0,1)$$

3. set $s = \mu + R^\top u$

The rest of this document focusses on step 2. Steps 1 and 3 can be used to map the results shown here to any jointly Gaussian distribution.
Notice that $\mathcal{N}(u;0,\boldsymbol{I})$ is a spherically symmetric distribution. So another way to draw u would be to draw a radius r from a Gaussian, then choose u uniformly at random on the surface of the $(N-1)$-sphere. This also means that, if we plot u in a graph, then there is this entire $(N-1)$-dimensional manifold of entirely equivalent points that we do not plot. It would be great if we could somehow show this degeneracy in a plot, but it is hard to plot an $N-1$ dimensional space. Instead, we here use the one dimension afforded by time, to make an animation looping through a 1-dimensional sub-space of that manifold, which at least gives a vague feeling for the degeneracy. How should we choose the sub-space within the manifold? The manifold is a Lie group. We will choose a 1-dimensional orbit within that group – a great circle on the $(N-1)$ sphere – drawn uniformly at random. Here is one out of several possible ways to do this:

- draw a u uniformly at random as above
- draw a second $v \sim \mathcal{N}(0;\boldsymbol{I}_N)$. Orthogonalize v and u by Gram-Schmidt projection:

$$t = v - (v^\top u)u \tag{1}$$

this gives a *tangent* t to the $(N-1)$-sphere at u. That tangent is an element of the Lie algebra, a generator of an orthodrome (a great circle, an orbit) on the $(N-1)$-dimensional sphere.[2]

- For each point $\ell_i, i = 1, \ldots, F+1$ on a grid from 0 to 2π, (F is the total number of frames on the resulting animation), map $x_i = \mathrm{Exp}(u, \ell_i t)$, where Exp is the exponential map (not the exponential function) from the tangent bundle at x to the tangent orbit at $t_i = \ell_i t$. The matlab code at the end of this document provides a function that does all the above.

3 Making Ti*k*Z plots

The following programming pipeline turns the resulting data into tikz-plots that can be used in the LaTeX beamer/pgf environment (using Adobe Reader)

1. loop from 1 to F, plotting $x_i = \mu + R^{\mathsf{T}}\mathbb{E}(u, \ell_i t)$
2. for each plot, construct TikZ output, using the `matlab2tikz` package[3]
3. compile each TikZ output into a pdf file, to save compile time (otherwise your TeX source compilation will become very slow). For this, I use a trick described in[4], but apparently the newest version of TikZ can do this out of the box (try searching for "tikz externalize")
4. use the animate package[5] to construct animated loops. Your LaTeX source will contain lines like these here:

```
\begin{animateinline}[autoplay,loop]{10}
  \inputTikZ{./figs/nameofyourfigure_frame_1}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_2}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_3}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_4}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_5}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_6}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_7}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_8}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_9}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_10}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_11}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_12}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_13}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_14}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_15}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_16}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_17}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_18}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_19}\newframe
  \inputTikZ{./figs/nameofyourfigure_frame_20}
\end{animateinline}
```

[2] Neil Lawrence has pointed out to me that the orthogonalization is not strictly required when drawing from a Gaussian process prior, because the two draws are orthogonal to each other with probability one when assuming that they are only a finite-dimensional restriction of a truly nonparametric (infinite-dimensional) draw from a white-noise process. I am keeping the orthogonalization in the formulation above to ensure it is also applicable to finite-rank Gaussian distributions.

[3] `https://github.com/nschloe/matlab2tikz`

[4] `http://kogs-www.informatik.uni-hamburg.de/~meine/tikz/process/`

[5] `http://www.ctan.org/tex-archive/macros/latex/contrib/animate/`

(the number 10 in curly brackets denotes the frame-rate. I have found that 10 looks relatively smooth). If you get out-of-LaTeX-memory errors, try reducing the dimensionality M of your Gaussian (see code below).

A Matlab code for generation of the raw samples

```
function X = GPanimation(d,n)
% returns a matrix X of size [d,n], representing a grand circle on the
% unit d-sphere in n steps, starting at a random location. Given a kernel
% matrix K, this can be turned into a tour through the sample space, simply by
% calling chol(K)' * X;
%
% Philipp Hennig, September 2012

x = randn(d,1);                             % starting sample
r = sqrt(sum(x.^2));
x = x ./ r;                                 % project onto sphere
t = randn(d,1);                             % sample tangent direction
t = t - (t'*x) * x;                         % orthogonalise by Gram-Schmidt.
t = t ./ sqrt(sum(t.^2));                   % standardise
s = linspace(0,2*pi,n+1); s = s(1:end-1);  % space to span
t = bsxfun(@times,s,t);                     % span linspace in direction of t
X = r.* exp_map(x,t);                       % project onto sphere, re-scale
end

function M = exp_map (mu, E)
% Computes exponential map on a sphere
%
% many thanks to Soren Hauberg!
D = size(E,1);
theta = sqrt(sum((E.^2)));
M = mu * cos(theta) + E .* repmat(sin(theta)./theta, D, 1);
if (any (abs (theta) <= 1e-7))
    for a = find (abs (theta) <= 1e-7)
        M (:, a) = mu;
    end % for
end % if
% M (:,abs (theta) <= 1e-7) = mu;
end % function
```

B Matlab code for turning raw samples into animation

The code I use to generate the animations looks like this. Your actual implementation will have to vary in some obvious places to accommodate your local setup.

```
function MakeGaussPlot(k,filename,phi)
% the standard gauss plot, using the nonlinear dataset
% Philipp Hennig, 11 Dec 2012

dgr = [0,0.4717,0.4604]; % color [0,125,122]
dre = [0.4906,0,0]; % color [130,0,0]

lightdgr = [1,1,1] - 0.5 * ([1,1,1] - dgr);
lightdre = [1,1,1] - 0.5 * ([1,1,1] - dre);

dgr2white = bsxfun(@minus,[1,1,1],bsxfun(@times,(linspace(0,0.6,2024)').^0.5,[1,1,1]-dgr));
dre2white = bsxfun(@minus,[1,1,1],bsxfun(@times,(linspace(0,0.6,2024)').^0.5,[1,1,1]-dre));

addpath ~/Documents/MATLAB/matlab2tikz-0.1.2/

M  = 150; % # dimensions
```

```
F  = 20;  % # frames
x  = linspace(-8,8,M)';
s1 = GPanimation(M,F);
s2 = GPanimation(M,F);
s3 = GPanimation(M,F);

dummytext = @(filename)(['\documentclass[10pt,usepdftitle=false]{beamer}',...
    '\pgfrealjobname{dummy} \input{preamble.tex}\begin{document} \begin{frame}',...
    '\inputTikZ{',filename,'} \end{frame}\end{document}']);

GaussDensity = @(y,m,v)(bsxfun(@rdivide,exp(-0.5*...
    bsxfun(@rdivide,bsxfun(@minus,y,m').^2,v'))./sqrt(2*pi),sqrt(v')));

%% prior
kxx = k(x,x); % kernel function (enter your favorite here)

m   = zeros(M,1);
V   = kxx;
L   = chol(V + 1.0e-8 * eye(M)); % jitter for numerical stability

y = linspace(-15,20,250)';
P = GaussDensity(y,m,diag(V+eps)); colormap(dgr2white);

for f = 1:F
    clf; hold on


    imagesc(x,y,P);
    plot(x,max(min(m,20),-15),'-','Color',dgr,'LineWidth',0.7);
    plot(x,max(min(m + 2 * sqrt(diag(V)),20),-15),'-','Color',lightdgr,'LineWidth',.5);
    plot(x,max(min(m - 2 * sqrt(diag(V)),20),-15),'-','Color',lightdgr,'LineWidth',.5);

    if nargin > 2
        plot(x,phi(x),'-','Color',0.7*ones(3,1));
    end

    plot(x,m + L' * s1(:,f),'--','Color',dre);
    plot(x,m + L' * s2(:,f),'--','Color',dre);
    plot(x,m + L' * s3(:,f),'--','Color',dre);
    xlim([-8,8]);
    ylim([-15,20]);

    drawnow; pause(0.1)
    if nargin > 1 && ~isempty(filename)
        fname = [filename, 'priorfuncs',num2str(f)];
        matlab2tikz([fname,'.tikz'],'width','0.8\textwidth','height','0.4\textwidth')
        file = fopen('dummy.tex','w');
        fwrite(file,dummytext(fname));
        fclose(file);

        command = ['/usr/local/texlive/2011/bin/x86_64-darwin/pdflatex -jobname="',...
            fname,'-external" ~/Documents/dummy.tex'];
        system(command);
    end
end
```