

# Quantization of Gaussian processes

GP lunch seminar 111225

Marnix Van Soom [VUB]

# Speech inversion [1]

```
graph-easy --from=dot --as_boxart << 'EOF'  
digraph {  
    rankdir = LR;  
    many -> one [label="to"];  
}  
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

```
x = load("bush.wav")
```

```
play(x)
```

```
play(x**2)
```

```
play(exp(1 + 5*x - x**2))
```

————— [finished with error] —————

```
Traceback (most recent call last):  
  File "/tmp/.presenterm92g7V8/snippet.py", line 1, in  
<module>  
    from play import *  
ModuleNotFoundError: No module named 'play'
```

*I know the human being and fish can coexist peacefully!*

-- George W. Bush

## Speech inversion [2]

Problem:

```
graph-easy --from=dot
--as_boxart << 'EOF'
digraph {
    "u * h = d"
}
EOF
```

snippet +exec\_replace is disabled,  
run with -X to enable

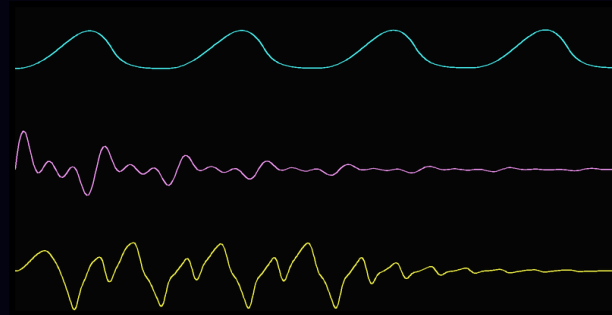
Given d, find (u, h).

This a blind identification problem:

- Ill-posed
- Needs regularization

```
graph-easy --from=dot --as_boxart << 'EOF'
digraph {
    rankdir = LR;
    "∞ many" -> one [label="to"];
}
EOF
```

snippet +exec\_replace is disabled, run with -X to enable



# Ill-posed problems

## ■ Ill-posed inverse problems are everywhere

- Many-to-one mappings
- Repeated measurements
- Any interpolation problem!
- Any extrapolation problem!

The data admit many solutions.

=> Which solution(s) do we want?

## ■ Simplest ill-posed problem

$$y = a_1 + a_2$$

Suppose we observe:  $y = 1$ .

=> What is the solution space?

```
gnuplot <<'GP'  
set term pngcairo size 560,560 background rgb "black"  
enhanced  
  
set size square  
unset key  
  
set border lc rgb "#4c566a"  
set tics textcolor rgb "#4c566a"  
set xlabel "a_1" textcolor rgb "#d8dee9"  
set ylabel "a_2" textcolor rgb "#d8dee9"  
set zeroaxis lc rgb "#434c5e"  
  
set label "a_1 + a_2 = 1" at -1.3,1.2 tc rgb "#88c0d0"  
# set label "ridge cost" at 0.7,0.6 tc rgb "#5e81ac"  
  
set xrange [-1.5:1.5]  
set yrange [-1.5:1.5]  
  
set parametric  
set trange [-2*pi:2*pi]  
  
# data constraint: a1 + a2 = 1  
x_line(t) = t  
y_line(t) = 1 - t  
  
# ridge circle radii  
r1 = 0.2  
r2 = 0.5  
r3 = 1.0  
  
# ridge solution (closest point to origin)  
a_star = 0.5  
  
plot \
```

# Ridge regression

## Linear regression

Given  $m$  basis functions  $\Phi = [\phi_1 \mid \phi_2 \mid \dots \mid \phi_m]$ , model a function  $f(x)$ :

$$f(x) = a_1\phi_1(x) + a_2\phi_2(x) + \dots + a_m\phi_m(x)$$

and fit the amplitudes  $a$  to data  $y$  by least squares:

$$\hat{a} = \operatorname{argmin}_a \|\Phi a - y\|^2 \\ = (\Phi^T \Phi)^{-1} \Phi^T y$$

=> This crashes numerically for ill-posed problems

## Ridge regression

Introduce a minimal engineering fix:

$$\hat{a} = \operatorname{argmin}_a \|\Phi a - y\|^2 + \lambda \|a\|^2 \\ = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$$

This stabilizes inversion by making large amplitudes costly. Eigenvalues shifted by  $+\lambda$ .

## Anisotropic ridge regression

Assign different costs to different amplitudes:

$$\hat{a} = \operatorname{argmin}_a \|\Phi a - y\|^2 + \lambda a^T \Sigma^{-1} a \\ = (\Phi^T \Phi + \lambda \Sigma^{-1})^{-1} \Phi^T y$$

```
python live/ridge.py
```

[finished]

# Gaussian processes

```
python live/gaussian.py
```

[finished]

## ■ Good

Gaussian processes are naturally born ill-posed problem killers because hyperparameter optimization regularizes automatically. No need to choose  $\Sigma$ !

Instead, costs are specified at a high functional level, not at the amplitude level, and determine model properties such as:

- lengthscale (smoothness)
- differentiability
- periodicity
- relevance of input dimensions

They compose in a kernel algebra:  $k_1 * k_2 + k_3$ , etc.

Gaussian process	ridge regression
condition on data	move along the nullspace
change hyperparameters	change costs $\Sigma$

## ■ Bad

- $O(N^3)$  inference cost
- Inference quality depends on kernel...
- ... but this can feel like black-box magic at times

# Ridge regression $\Leftrightarrow$ Gaussian processes

```
graph-easy --from=dot --as_boxart << 'EOF'  
digraph {  
    rankdir = LR;  
    "Gaussian process" -> "ridge" [label="quantize!"];  
}  
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

## ■ Good

- $O(N)$  inference, not  $O(N^3)$
- Works really well in practice
- Almost all kernels used in daily life are easily quantifiable
- Plugs into VI
- Turns it into a white-box
  - Level 1 and level 2 learning

## ■ Bad

- Still an approximation, which might or might not be appropriate
- Nonstationary kernels are harder to quantize
- Might require many basisfunctions  $m$  in higher dims
- Kernel algebra rapidly increases  $m$

# Quantization [1]

## Hilbert-GP

A stationary kernel is translation invariant, so it admits a spectral representation:

$$k(r) = \int S(\omega) e^{i\omega r} d\omega$$

Hilbert-GP approximates this integral by regular quadrature. This results in the following ridge regression model:

$$f(x) = a_1\phi_1(x) + a_2\phi_2(x) + \dots + a_m\phi_m(x)$$

$$\phi_k(x) = \sqrt{2/L} \cdot \sin(\omega_k x), \quad \omega_k = k\pi / L$$
$$\sum_k S(\omega_k)$$

So higher frequencies have higher cost => determines lengthscale and smoothness.

```
graph-easy --from=dot --as_boxart << 'EOF'
digraph {
    rankdir = LR;
    "Stationary GP" -> "ridge" [label="Hilbert!"];
}
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

```
gnuplot <<'GP'
set term pngcairo size 800,600 background rgb "black"
enhanced

set border lc rgb "#4c566a"
set tics textcolor rgb "#4c566a"

set xlabel "frequency  $\omega$ " tc rgb "#d8dee9"
set ylabel "S( $\omega$ )" tc rgb "#d8dee9"

set xrange [0:10]
set yrange [1e-5:2]
set logscale y

# Legend
set key top right
set key tc rgb "#d8dee9"
set key spacing 1.2
set key box lc rgb "#4c566a"

set title "Kernel magnitude spectra ( $\ell = 1$ )" tc rgb
"#d8dee9"

# lengthscale
ell = 1.0
```

## Quantization [2]

### ■ Spherical harmonics

```
convert -density 300 assets/embedding.png -background none  
-channel RGB -negate png:-
```

snippet +image is disabled, run with -X to enable

[Dutordoir+ 2020]

```
graph-easy --from=dot --as_boxart << 'EOF'  
digraph {  
    rankdir = LR;  
    "Stationary/NN GP" -> "ridge" [label="Spherical!"];  
}  
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

```
convert -density 300 assets/spherical_harmonics.png  
-background none -channel RGB -negate png:-
```

snippet +image is disabled, run with -X to enable

## Quantization [3]

### ■ Spherical harmonics

```
convert -density 300 assets/uvx_learning_curve.png  
-background none -channel RGB -negate png:-
```

snippet +image is disabled, run with -X to enable

```
graph-easy --from=dot --as_boxart << 'EOF'  
digraph {  
    rankdir = LR;  
    "Stationary/NN GP" -> "ridge" [label="Spherical!"];  
}  
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

```
convert -density 300 assets/uvx.png -background none  
-channel RGB -negate png:-
```

snippet +image is disabled, run with -X to enable

# Learning on levels 1 & 2

Quantization creates a more white-box model, which allows for a bonus:

more effective *learning from examples*

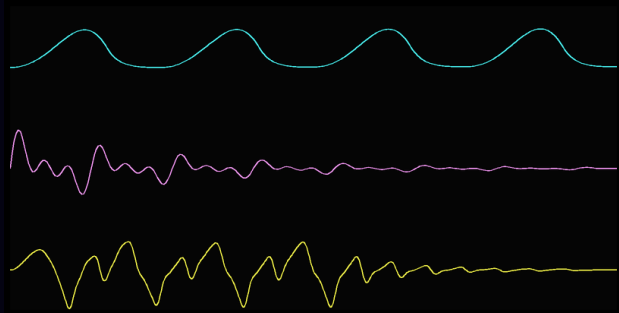
level	Gaussian process	ridge regression
0	prior	prior
1	hyperparam learning	learning the basis $\Phi$ and costs $\Sigma$
2		learning the amplitudes $a$

```
python live/levels.py
```

[finished]

# Results

On the OPENGL0T-I benchmark of [Alku+ 2019] for testing speech inversion algorithms.



```
convert -density 300 assets/score.png -background none -channel RGB  
-negate png:-
```

snippet +image is disabled, run with -X to enable

# Summary

## Takeaway:

- Many problems are ill-posed and need regularization, or "cost control"
- Use GPs to specify high-level information about costs
- Quantize them to quick ridge regression models which thusly acquire sophistication
- Set hyperparameters, or fit to examples
  - Level 1
  - Level 1 & 2
- Enjoy better inference when lucky

## Advantages:

- $O(N)$  inference, not  $O(N^3)$
- Level 1 & 2 learning
- Principled cost control
- Any likelihood

```
graph-easy --from=dot --as_boxart << 'EOF'
digraph {
  "Gaussian process" -> "Ridge regression"
  [label="quantize!"];
  "Ridge regression" -> "Gaussian process"
  [label="special\ncase of"];
}
EOF
```

snippet +exec\_replace is disabled, run with -X to enable

## Disadvantages:

- Still an approximation
- Requires lower input dimensions:  $O(10)$ 
  - Though variations exist that go up to  $O(200)$

# References

## ■ References [1]

Hilbert GP:

1. Solin, A. & Särkkä, S. **Hilbert space methods for reduced-rank Gaussian process regression**. Stat Comput 30, 419–446 (2020)
2. Riutort-Mayol, G., Bürkner, P.-C., Andersen, M. R., Solin, A. & Vehtari, A. **Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming**. arXiv:2004.11408

Spherical harmonics:

1. Dutordoir, V., Durrande, N. & Hensman, J. **Sparse Gaussian Processes with Spherical Harmonic Features**. in Proceedings of the 37th International Conference on Machine Learning 2793–2802 (PMLR, 2020).

## ■ References [2]

Ways to higher input dimensions:

1. Rahimi, A. & Recht, B. **Random Features for Large-Scale Kernel Machines**. in Advances in Neural Information Processing Systems (2007)
2. Eleftheriadis, S., Richards, D. & Hensman, J. **Sparse Gaussian Processes with Spherical Harmonic Features Revisited**. arXiv:2303.15948
3. Mutny, M. & Krause, A. **Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features**. in Advances in Neural Information Processing Systems (2018)

■ Any questions or contact for further discussing or colab:

marnix@ai.vub.ac.be