Improving Computational Efficiency of Upper Atmospheric Wind Estimations with Gaussian Processes

Oana Mirestean

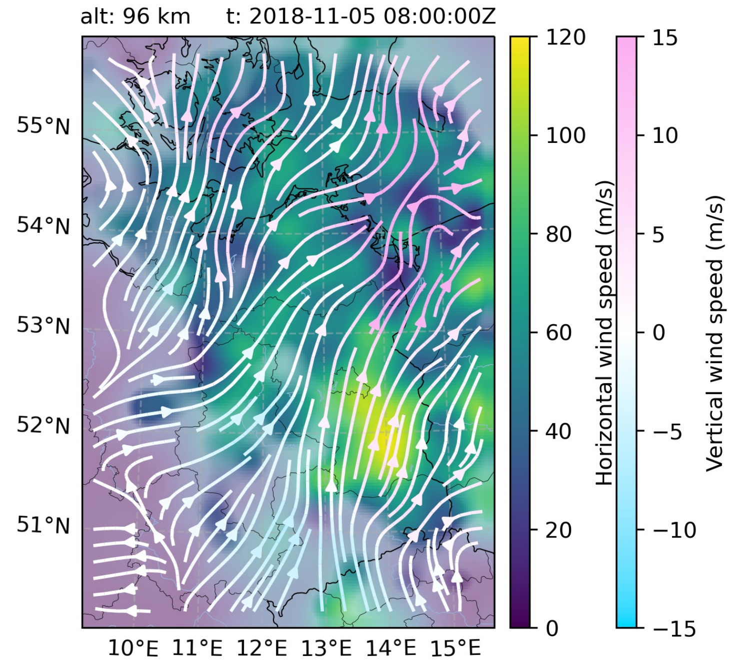
Mentor: Dr. Ryan Volz

MIT HAYSTACK OBSERVATORY

CORNELL UNIVERSITY FOUNDED A.D. 1865

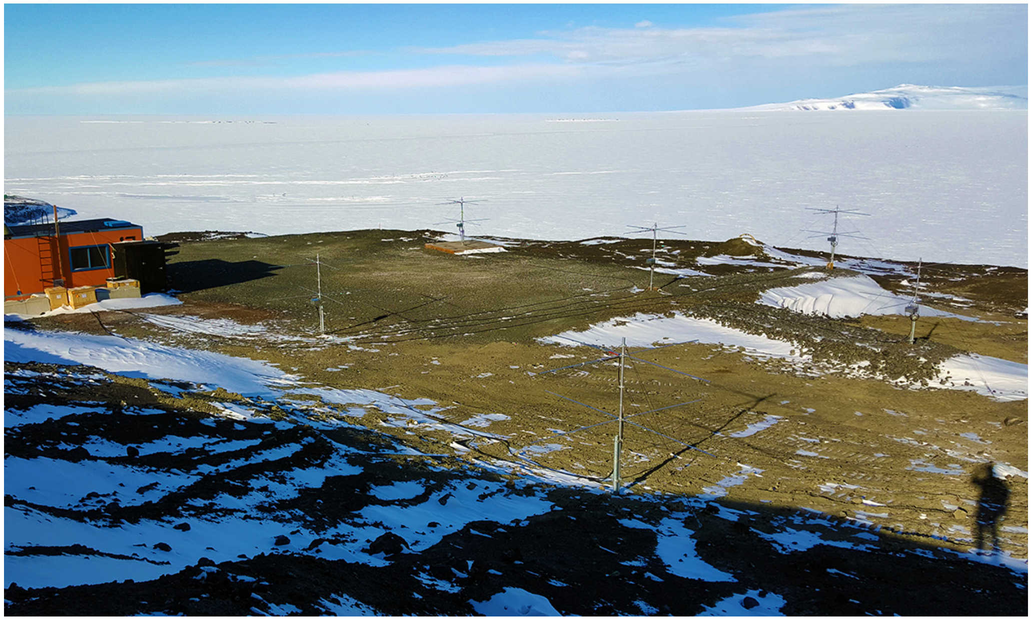
Introduction

- Winds in neutral part of upper atmosphere
  - Mesosphere and Lower Thermosphere (MLT): 60-120 km altitude
- Indirectly measure wind components using meteor radars
  - Measure Doppler shift
- Use Gaussian Processes to estimate wind fields
  - Goal: Improve computational efficiency and estimate different models



MIT
HAYSTACK
OBSERVATORY

Meteor Radars



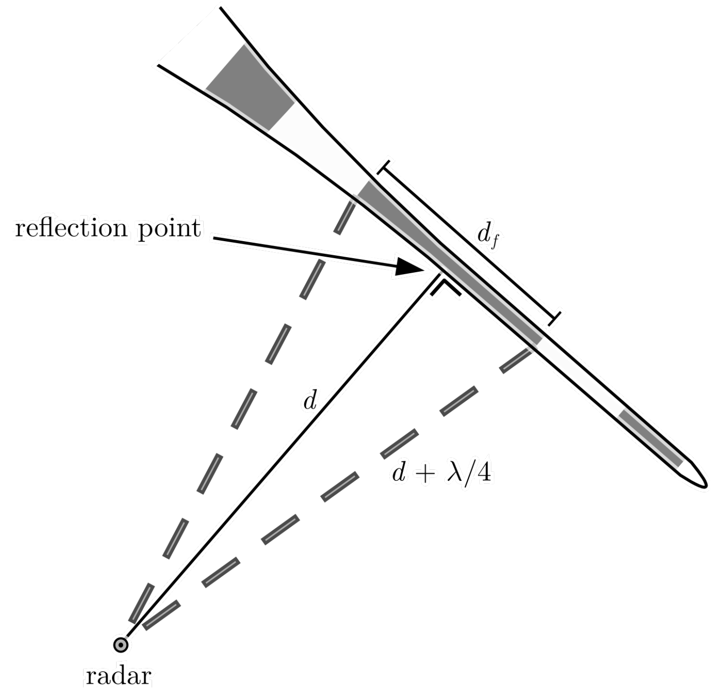
McMurdo Meteor Radar
https://ccar.Colorado.edu/meteors/

- Meteor observations typically occur between 80-100 km altitude
- Types of meteor scatter:
  - Head: plasma ball traveling with the meteoroid
  - Trail: wake of plasma left behind by meteor
- Meteor radars mainly detect trail echoes

MIT
HAYSTACK
OBSERVATORY

Meteor Wind Measurements

- Meteor trail reflects the radar signal like a mirror
  - Detection when Bragg vector is perpendicular to meteor trail
- Bragg Vector $\boldsymbol{k}_B = \boldsymbol{k}_s - \boldsymbol{k}_i$
  - Difference between scattered and incident wave vectors



MIT
HAYSTACK
OBSERVATORY

Wind Field Estimations

- Doppler shift of reflected meteor echo signal comes from the projection of the atmospheric wind vector on the Bragg vector.

Doppler shift f at location (Λ, Φ, z) and time t

measured

measured

$$f(\Lambda, \Phi, z, t) = \frac{1}{2\pi} \begin{bmatrix} k_u & k_v & k_w \end{bmatrix} \begin{bmatrix} u(\Lambda, \Phi, z, t) \\ v(\Lambda, \Phi, z, t) \\ w(\Lambda, \Phi, z, t) \end{bmatrix}$$

Bragg vector components

Wind vector $\boldsymbol{u}$ components

unknown

- Use Gaussian processes to estimate wind vector components.

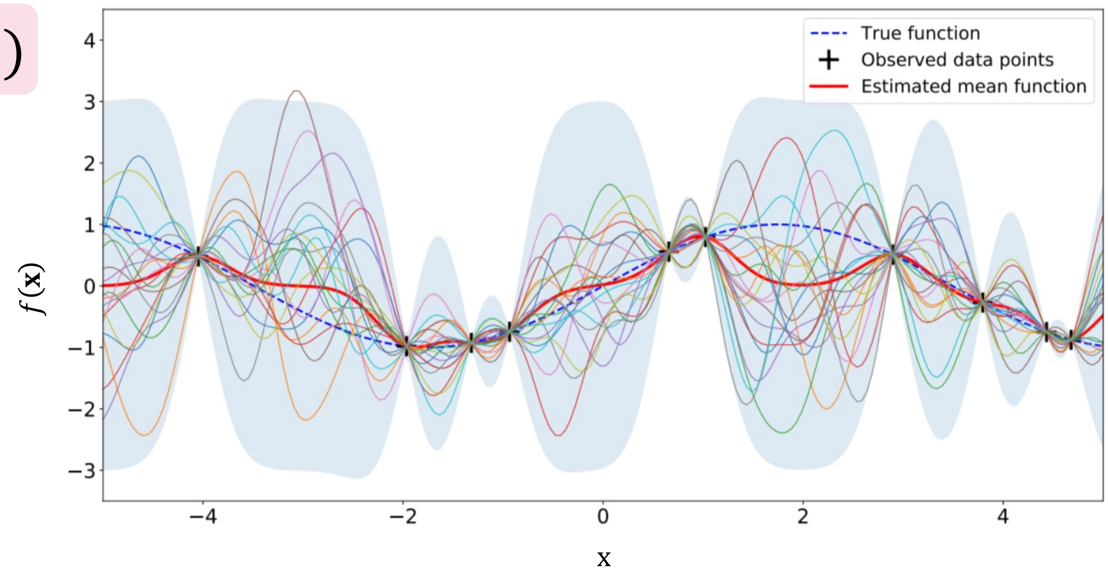
MIT HAYSTACK OBSERVATORY

Gaussian Process Regressions

Gaussian Process Regression (GPR): A statistical inference method for estimating a function $f(\mathbf{x})$.

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')\right)$$

- Fully defined by parameterized mean and covariance functions
- Confidence bound indicates prediction variance.



MIT
HAYSTACK
OBSERVATORY

J. Wang, *An intuitive tutorial to Gaussian processes regression* (2021).

Wind Components as Gaussian Processes

- Model wind components independently as Gaussian processes
- Choose priors for mean and covariance functions

$$u(\mathbf{x}) \sim \mathcal{GP}\big(m_u(\boldsymbol{x}), \kappa_u(\boldsymbol{x}, \boldsymbol{x}')\big)$$
$$v(\mathbf{x}) \sim \mathcal{GP}\big(m_v(\boldsymbol{x}), \kappa_v(\boldsymbol{x}, \boldsymbol{x}')\big)$$
$$w(\mathbf{x}) \sim \mathcal{GP}\big(m_w(\boldsymbol{x}), \kappa_w(\boldsymbol{x}, \boldsymbol{x}')\big)$$

Mean: *cubic spline*

$$m_u(\boldsymbol{x}) = m_u(t, z)$$
$$m_v(\boldsymbol{x}) = m_v(t, z)$$
$$m_w(\boldsymbol{x}) = m_w(t, z)$$

Covariance: *Matérn Kernel*

$$\kappa_u(\boldsymbol{x}, \boldsymbol{x}') = \sigma_u^2\, \kappa_d(\boldsymbol{x}, \boldsymbol{x}')$$
$$\kappa_v(\boldsymbol{x}, \boldsymbol{x}') = \sigma_v^2\, \kappa_d(\boldsymbol{x}, \boldsymbol{x}')$$
$$\kappa_w(\boldsymbol{x}, \boldsymbol{x}') = \sigma_w^2\, \kappa_d(\boldsymbol{x}, \boldsymbol{x}')$$
$$\kappa_d(\boldsymbol{x}, \boldsymbol{x}') = \kappa_{Matérn,\, v={}^5/_2}\big(\boldsymbol{x}, \boldsymbol{x}'; \delta_x, \delta_y, \delta_z, \delta_t\big)$$

- Goal: fit parameters using the GP model
  - Output scale: $\sigma_u^2, \sigma_v^2, \sigma_w^2$
  - Length scale: $\delta_x, \delta_y, \delta_z, \delta_t$

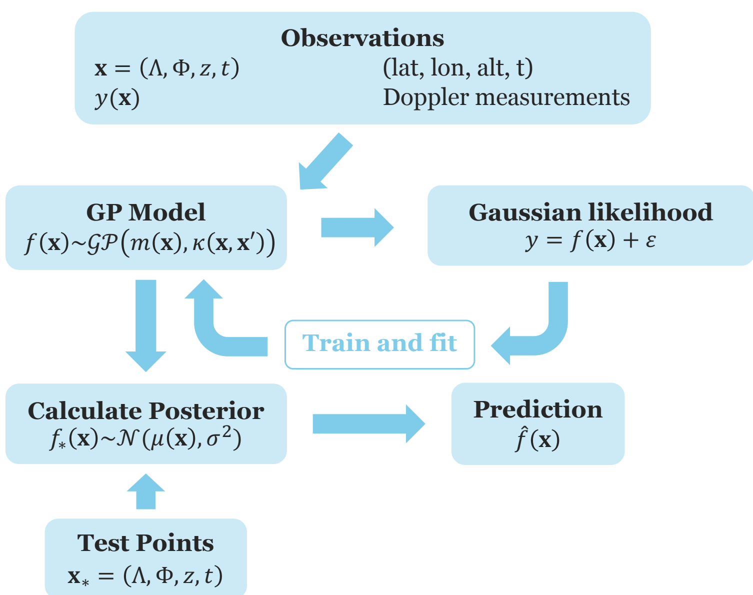
MIT HAYSTACK OBSERVATORY

Implementing the GPR in GPyTorch

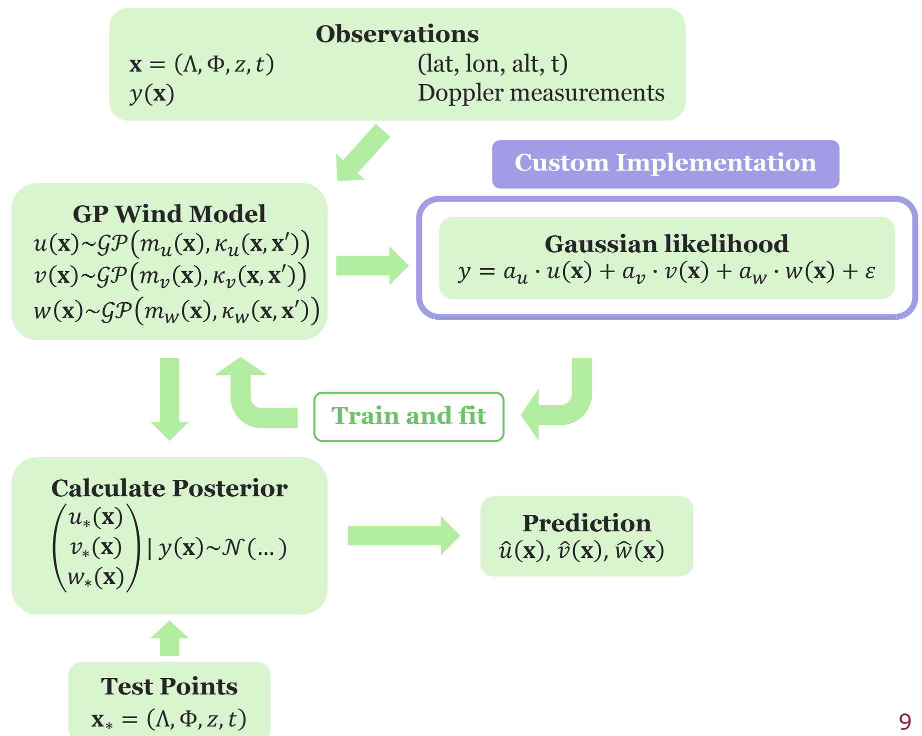
- GPR computational complexity: $\mathcal{O}(n^3)$
  - High computational time
  - Limits amount of that can be data processed
  - Goal: improve the computational efficiency by implementing the GPR using GPyTorch
- GPyTorch: Python library for GPRs based on the machine learning library PyTorch.
  - Provides a straightforward structure for implementing GPRs
    - Easily try different models
  - Has various approximation methods that allow us to scale the problem
  - Built on PyTorch and can incorporate GPU processing

GPyTorch Wind Model

Standard GPyTorch GPR



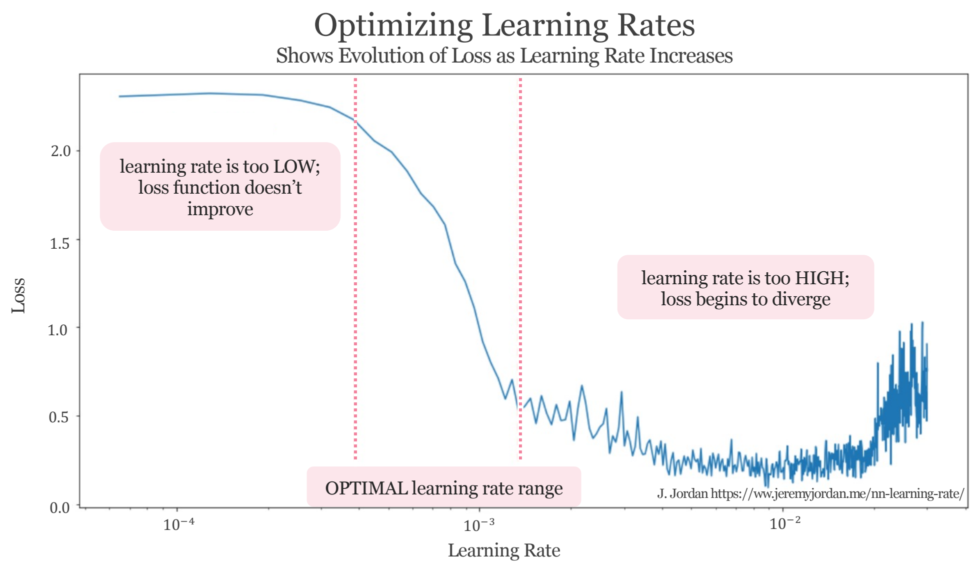
Wind Model GPR



MIT HAYSTACK OBSERVATORY

Optimizing GPyTorch Wind Model

- PyTorch optimizers for minimizing the marginal log likelihood function in fitting
  - Adam
  - Adamax
  - LBFGS
  - Stochastic Gradient Descent (SGD)
  - SLSQP
- Learning rate schedulers
- Don't use approximations to compute log probability of posterior distributions




```
with gpytorch.settings.fast_computations(log_prob = False):
```

MIT HAYSTACK OBSERVATORY

GPyTorch Wind Estimations Using Simulated Data

	True	Jax		GPyTorch														
		SLSQP		Adam			Adamax			LBFGS			SGD			SLSQP		
σ_u^2	900	1307	45.3%	1480	13.2%	64.4%	1155	11.6%	28.4%	1305	0.2%	45.0%	544	58.4%	39.6%	1309	0.1%	45.5%
σ_v^2	900	1486	65.1%	1701	14.5%	89.0%	1291	13.1%	43.5%	1484	0.1%	64.9%	613	58.7%	31.8%	1488	0.1%	65.3%
σ_w^2	90	115	27.8%	136	18.6%	51.5%	113	1.8%	25.5%	115	0.09%	27.9%	29	75.0%	68.0%	115	0.04%	27.7%
δ_x	50e3	60372	20.7%	61564	2.0%	23.1%	58161	3.7%	16.3%	60320	0.09%	20.6%	45449	24.7%	9.1%	60386	0.02%	20.8%
δ_y	50e3	58221	16.4%	59395	2.0%	18.8%	56212	3.4%	12.4%	58197	0.04%	16.4%	44677	23.3%	10.6%	58184	0.06%	16.4%
δ_z	3e3	3407	13.6%	3520	3.3%	17.3%	3288	3.5%	9.6%	3409	0.04%	13.6%	2575	24.4%	14.2%	3410	0.08%	13.7%
δ_t	1800	2060	14.5%	2201	6.8%	22.3%	1978	4.0%	9.9%	2060	0.03%	14.4%	1538	25.4%	14.6%	2062	0.08%	14.6%
Loss		0.547		0.547716			0.547526			0.547028			0.590128			0.547029		

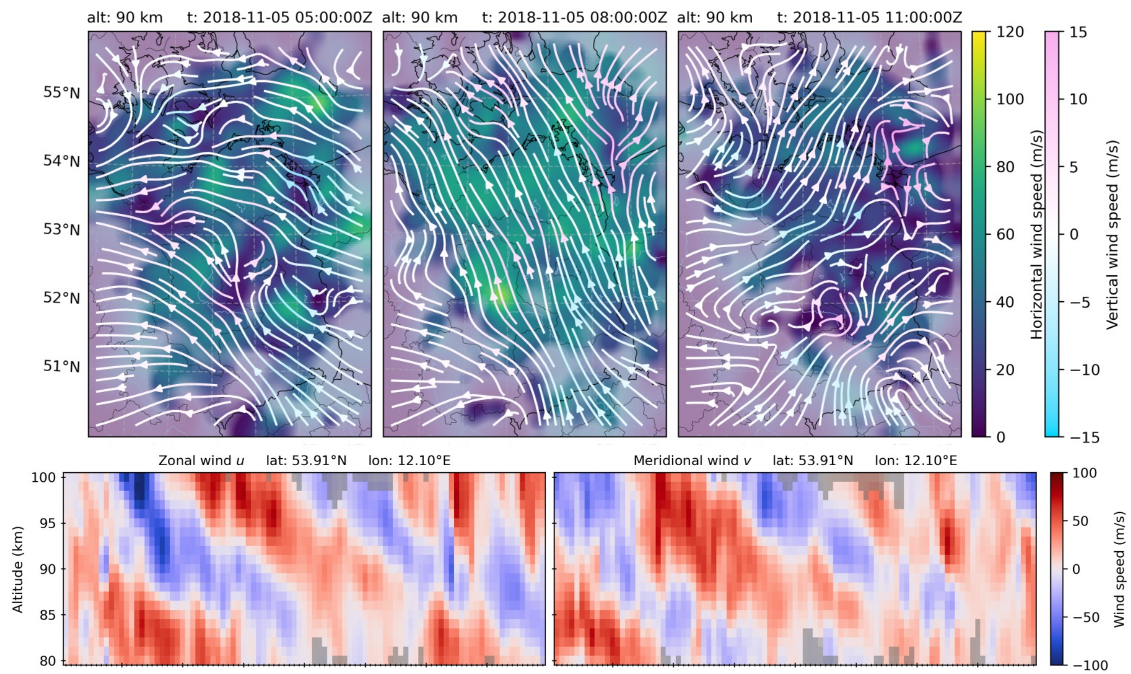
Percent error between GPyTorch (G) parameters and Jax parameters (J): $\frac{|G-J|}{J} \times 100\%$

Percent error between GPyTorch (G) parameters and true parameters (p): $\frac{|G-p|}{p} \times 100\%$

MIT
HAYSTACK
OBSERVATORY

Future Work

- Finish implementing GPyTorch wind field predictions
- Exercising the improved efficiency with larger data sets
- Explore different models with different sets of parameters



MIT HAYSTACK OBSERVATORY