Data-driven parameterization for climate modeling (QBO)

Mentor: Prof.Edwin Gerber
Researcher: Zihan Shao

Abstract

Quasi-Biennial Oscillation, QBO in short, is a quasiperiodic oscillation of the equatorial zonal wind between easterlies and westerlies in the tropical stratosphere with a mean period of 28 to 29 months.[1] Atmospheric Gravity Waves, GWs in short, plays an important role in the formation of QBO. This project will be focusing on the data-driven parameterization for the GWs in the QBO climate model, aiming to improve climate model's ability to simulate this vacillation of jets in the tropical stratosphere. In specific, we will focus on the 1-dimensional version of the problem.

In this document, I will basically record everything worth recording of my summer research on Data-Driven parametrization of QBO, under the supervision of Prof. Edwin Gerber. We start with a brief introduction.

Introduction

Atmospheric Gravity Waves(GWs), play an important role in the exchange of momentum between the Earth's surface and the free atmosphere. They slow down the tropospheric jet streams and stratospheric polar night jets, and are a key driver of the QBO. GWs present a challenge to climate prediction because they cannot be properly resolved with available computational power. They must rather be estimated from the larger scale variables that are resolved, or parameterized, as how it is described in the field. However, current physics-based gravity wave parameterizations neglect transient effects and account only for vertical wave propagation.[3] By using data-driven model, parametrizations will be developed to more accurately and efficiently represent GW momentum fluxes in weather and climate models. This, ultimately, will enhance our ability to predict the changes in climate and weather.

In this project, we will try to do the data-driven parametrization for GWs in QBO models. Followed is a mathematical characterization of the problem. In 1-dimensional version of the problem, the unknown zonal wind u, which we expect to oscillates biannually, is governed by the following equation

$$\frac{\partial u}{\partial t} + w\frac{\partial u}{\partial z} - K\frac{\partial^2 u}{\partial z^2} = -S(u, z, A, c)$$

Here u is the unknown zonal wind, t is the time, z is the vertical coordinate, w and K are constant vertical advection and diffusivity, and $S(u, z, A, c)$ is the wave forcing consisting of a sum of monochromatic waves, which are the Gravity Waves we mentioned above[2]. Note that here u is internally generated by the model and A and c are externally forced (that is, supplied as some external condition), which we take to be generated by a random variable. For now, we have a physical parametrization of S. This, together with the PDE above, will generate a series of (S, u, z, A, c), which will be treated as the ground truth.

In that case, we are going to design a Machine Learning model to predict S given (S, u, z, A, c) generated by the physical parametrization. The parameters we might use includes but not restricted to: u(zonal wind), z(vertical coordinates), A(amplitude) and c(phase speed).

Note that the evaluation metric for this problem will be different from the usual Data Science problem. On the one hand, the predicted S should be close to the ground truth. On the other hand, the predicted S, together with the PDE, should generate the correct oscillation. Besides that, the model is expected to have the ability to generalize. Once trained on the current QBO, it should produce the actual QBO under slight perturbation of parameters(which corresponds to climate change). In general, the second criterion is more important, since we want the predicted model to be used in actual climate prediction.

Week 1

Summer research on QBO (at least at the very beginning) can be divided into two parts. On the one hand, we are coming to the conclusion of the linear models. On the other hand, I'm starting to look at nonlinear models, such as Supported vector regression and deep learning models.

Preliminary experiments with SVR didn't even give us a reasonable result. In that case, we decided to look into the mathematical essence of SVR.

Mathematics of SVR

In this section, we will derive Supported Vector Regression in the language of mathematics.

Intuition

Like Supported Vector Machine for classification, the story begins with a hyperplane with its tube. The goal of SVM to separate the data by the hyperplane and make sure most of the data is off the tube and then maximize the diameter of the tube. Instead, the goal of SVR is to make sure most of the data will be lying in the tube and then maximize the tube. The predicted function will then be the hyperplane.

Construction

We construct the model in the following way:

- Hyperplane/predicted function:

  $$\hat{f}(x) = \langle w, x \rangle + b$$

  where $w \in \mathcal{R}^d$, $b \in \mathcal{R}^d$ where d is the number of features. w and b are trainable.

- The vertical distance between boundary, denoted ϵ. That is to say, we do not penalize the data if

  $$|\hat{f}(x_i) - y_i| \leq \epsilon$$

  ϵ is an hyperparameter.

- With this construction, the radius of the tube can be expressed as

  $$\frac{\epsilon}{\|w\|}$$

  Since ϵ is a hyperparameter, it suffices to maximize $\frac{1}{\|w\|}$. In practice, we maximize $\frac{1}{2}\|w\|^2$.

- In real applications, it is not a good idea to make every data point sits in the tube since the model would be very sensitive to outliers. Usually, we denote the upper and lower offset $\xi_i \geq 0$ and $\xi_i^* \geq 0$.

- Penalty term. To penalize the offset, we add a linear penalty term

  $$C \cdot \sum_i (\xi_i + \xi_i^*)$$

  to the objective function.

Optimization

Finally, we convert the problem into a optimization problem.

$$\min \frac{1}{2}\|w\|^2 + C \cdot \sum_i (\xi_i + \xi_i^*)$$

subject to $\forall i$

$$\begin{aligned} y_i - (\langle w, x_i \rangle + b) &\leq \epsilon + \xi_i \\ (\langle w, x_i \rangle + b) - y_i &\leq \epsilon + \xi_i^* \\ \xi_i &\geq 0 \\ \xi_i^* &\geq 0 \end{aligned}$$

This problem can be solved by Langrange Multipliers with K-K-T condition. Its dual problem is

$$\max L := -\frac{1}{2}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle v_i, v_j\rangle - \epsilon\sum_i(\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*)$$

subject to $\forall i$

$$\sum_i(\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C]$$

Supported Vectors

With the above construction, we can derive the expression of w

$$w = \sum_i(\alpha_i - \alpha_i^*)x_i$$

Here we can see, w is in fact a linear combination of x_is. In practice, most of the coefficient will be 0. Those few vectors with nonzero coefficient will then be called supported vectors. In SVR, supported vectors uniquely determine the model.

Kernel

For now, our model can only handle linear relations. But just like what we do in supported vector machine, we map the data into higher dimensional Euclidean space so that the data has a linear relation. That is to say, if we have a map ϕ that endows the data linear relations, we can then build a SVR model on $\{\phi(x_i)\}$.

In practice, it is almost impossible to derive an effective map *phi*. However, we can take advantage of the supported vectors expression. Since in the expression of $\hat{f}$,

$$\hat{f}(x) = \sum_i \alpha_i\langle\phi(x_i), \phi(x)\rangle$$

we can neglect the actual ϕ mapping but focus on the inner product in the high dimensional space. We define the kernel function

$$K(\cdot, \cdot) = \langle\phi(\cdot), \phi(\cdot)\rangle$$

In this way, we can make the ϕ map implicit. Instead, all we need to do is to replace the inner product with kernel K in the optimization problem and the actual expression of w. The most widely used K is the Radial Basis Function, RBF in short.

Power of SVR

SVR has a very strong ability to approximate nonlinearity. We build SVR model on some nonlinear ground truth such as $y = \sin(x)$ or $y = \exp(-\sin x) + \cos x$, and even Trajectory of Brownian motion. All of the models approximated the function well.

Week 2

Considering Supported Vector Regression model's ability to learn non-linearity (shown by its approximation of Brownian Motion trajectory), it has the potential to reach a satisfactory performance, at least on the offline side.

Punchline: Normalization

The most basic SVR model reports terrible results($R^2 < 0$), and the fitted function is a straight line. The problem could find its origin in the objective of SVR. There's no penalty if the data sits in ϵ-tube of the data i.e. $|\hat{f}^{(k)}(u_i) - s_i^{(k)}| < \epsilon$, where ϵ is set as 0.1 by default. However, we should notice that $s \ll \epsilon$, which fails the algorithm, since then a 0-line could easily optimize the objective trivially.

Since setting ϵ too small is harmful(as it would be very close to the single precision), it seems to be necessary to do normalization of the data, especially for s.

Hyperparamters Tuning

Here's the list of hyperparameters:

- K i.e. the kernel used
- ϵ i.e. the radius of the tube
- C i.e. magnitude of penalty
- the proportion of training data

We temporarily fix $K(x, y) = \exp(-\frac{\|x-y\|^2}{\gamma})$ since it is the best built-in kernel($\gamma = 73$ by default). When tuning other hyperparameters, we mainly consider the following aspects.

1. R^2 on testing data

   All hyperparameters can be influential on R^2. ϵ, in some sense, represents the degree of fitting. Before reaching overfitting, smaller ϵ and larger C will give us a better R^2. However, it is harmful to make ϵ too small. It definitely increase the training time as it takes longer to converge, but it is beyond our scope. But it increase the excution time in the mean time, which we'll discuss it later.

Also, it may be anti-intuitive that we don't need, or we should not, pass too many training samples to the model. Despite $(0.8, 0.2)$ as the usual train-test-split, 0.2 or less training data proves to be sufficient and better for this case.

2. Excution time

   We first look into the predicting function of SVR model. Given u, the prediction comes from

$$\hat{f}(u) = \sum_i \alpha_i K(u_i, u) + b$$

   As we can see from the expression, excution time depends on

   - # of supported vectors. Note that in our case(and also in most of cases) K is a nonlinear term, thus it is not possible to merger all $\{u_i\}$s. Therefore, the number of supported vectors(where α_i nonzero) is positively co-related to the excution time.
   - Dimension of u. As inside the kernel function, we need to compute the inner product of $u - u_i$, shorter u will results in shorter runtime.

   It is straightforward to control # of supported vectors by ϵ. Small ϵ will lead to a strict tube resulting in more supported vectors, and vice versa. Usually, we control # of supported vectors between 500-1500.

   As for dimension of u, we can do dimension reduction on u. We begin with the most basic dimension reduction method, Principle Components Analysis, PCA for short.

PCA

We omit the algorithm of PCA here. But PCA is definitely something worth trying as it is a linear transform that has no cost.

Surprisingly, when we insert u into the PCA model, the first two principle components account for 0.97 of the importance. This seems to be a good news, as then the input data can be

$$(\hat{u}_1, \hat{u}_2, s_f, c_w)$$

But the result is not good, but dimension reduction is still possible. We finally keep 16 dimensions of the u.

Problems with SVR

Bad Online Testing Result

For now, the online testing of the SVR model does not reveal the correct SVR.

Slow

SVR, in general, has a really uncontrolable testing time. It takes several minutes to run the PDE solver for 96 years.

One possible solution is to rewrite the kernel function. Now the kernel needs to compute the exponential function, and it is reasonable to use a self-designed kernel only using the first 3 terms of power series of exponential function.

Plan for next week

1. Modify the initial condition of the online testing. Use the last slice of the control run to be the initial condition for online testing.
2. Implement the RBF kernel manually in a (hopefully) more efficient way. Current idea is to only use the first 3 terms of the Taylor series.
3. Keep trying dimension reduction. Try if we can further reduce the dimension with PCA or other more advanced dimension reduction methods.

References

[1] Baldwin, M. P. et al. The quasi-biennial oscillation. Reviews of Geophysics, 39(2), 179–229. https://doi.org/10.1029/1999rg000073

[2] A webpage of 1d QBO probelm. https://datawaveproject.github.io/qbo1d/html/model-description.html

[3] A Data-informed Framework for the Representation of Sub-grid Scale Gravity Waves to Improve Climate Prediction.

[4] Smola, Alex. J et al. A tutorial on support vector regression. Statistics and Computing 14: 199–222, 2004. https://alex.smola.org/papers/2004/SmoSch04.pdf