Data-Driven Gravity Wave Parametrization in QBO-1D

Zihan Shao, under supervision of Prof. Edwin Gerber, thanks Dr. Ofer Shamir and Dave Connelly

Motivation

Need for Gravity Wave Parameterization

- The tropical stratospheric Quasi-Biennial Oscillation is driven largely by parametrized gravity wave breaking.

Data-driven i.e. utilizing Machine Learning methods

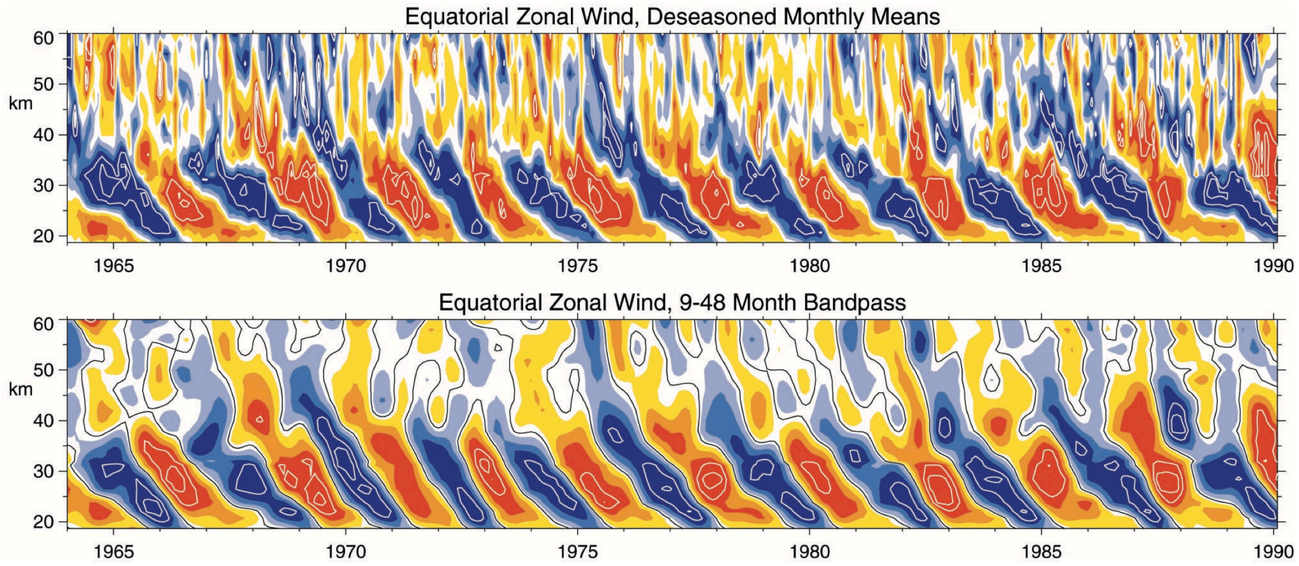
- Currently Gravity Waves are physically parametrized.
- Accuracy.
- Efficiency.

Set-up and Construction

QBO in a nutshell

Quasi-Biennial Oscillation

- Equatorial zonal wind oscillates between easterlies and westerlies in the tropical stratosphere
- with a mean period of 28 to 29 months (quasi-biennial)
- due to gravity waves forcing.
- In particular, our work is largely based on the 1-dim QBO model



QBO-1d Model

The model is a hybrid of the models used by Holton and Linden (1972) and Plumb (1977). The (advection-diffussion) equation reads:

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

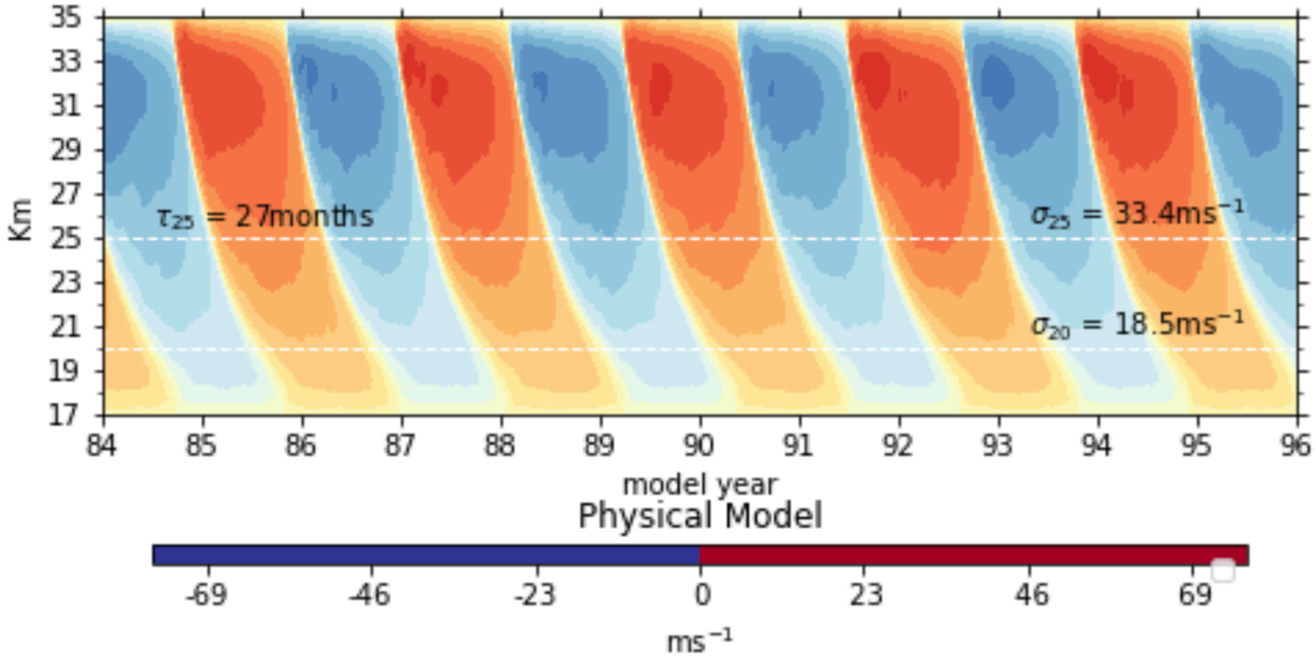
Where:

u: zonal wind

S: gravity wave forcing term

w, κ: advection and difussion constants

z: height(vertical coordinate)



*Cited from Ofer's documentation of qbo1d repository

Physical Parametrization of S with Stochastic forcing

$$S(u,z) = \frac{1}{\rho}\frac{\partial}{\partial z}F(u,z)$$

where the wave flux $F(u,z)$ is parameterized as follows:

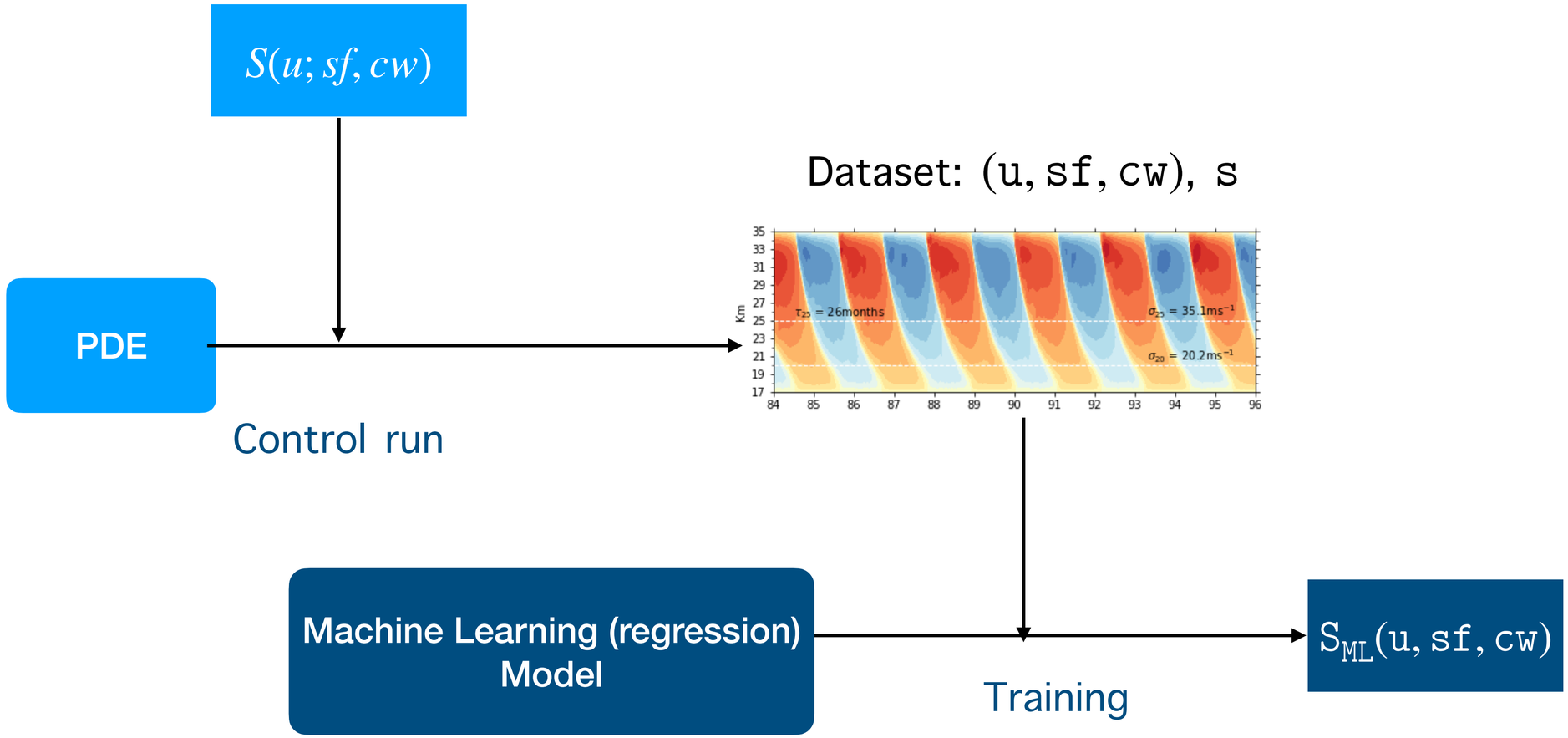
$$F(u,z) = \sum_i A_i \exp\left\{-\int_{z_1}^{z} g_i(u,z')\,dz'\right\}, \quad A(c) = \mathrm{sgn}(c)B_m \exp\left[-\ln 2\left(\frac{c}{c_w}\right)^2\right]$$

Note that, when $z = z_1$, $F(u,z_1) = \sum_i A_i$, denoted F_{S_0} (total source flux).

Stochasticity enters by making F_{S_0}(total source flux) and c_w(spectral width) Random Variables of time. (In practice, these two array are denoted `sf`, `cw`).

*Cited from Ofer's documentation of qbo1d repository

Data-Driven Parametrization

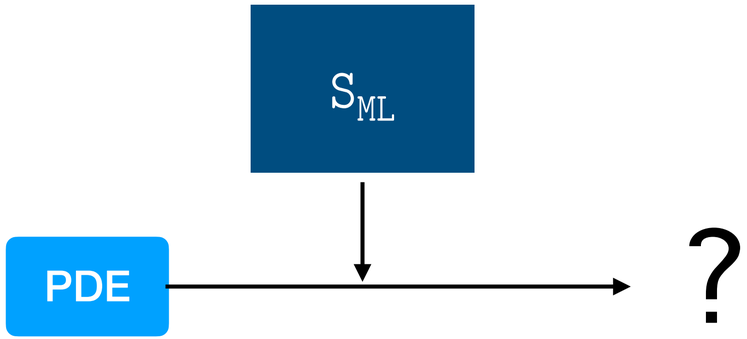


Goals and Evaluation Metric

$$R^2 = 1 - \frac{\sum_i^N (\hat{y}_i - y_i)^2}{\sum_i^N (y_i - \bar{y}_i)^2}$$

Offline and Online

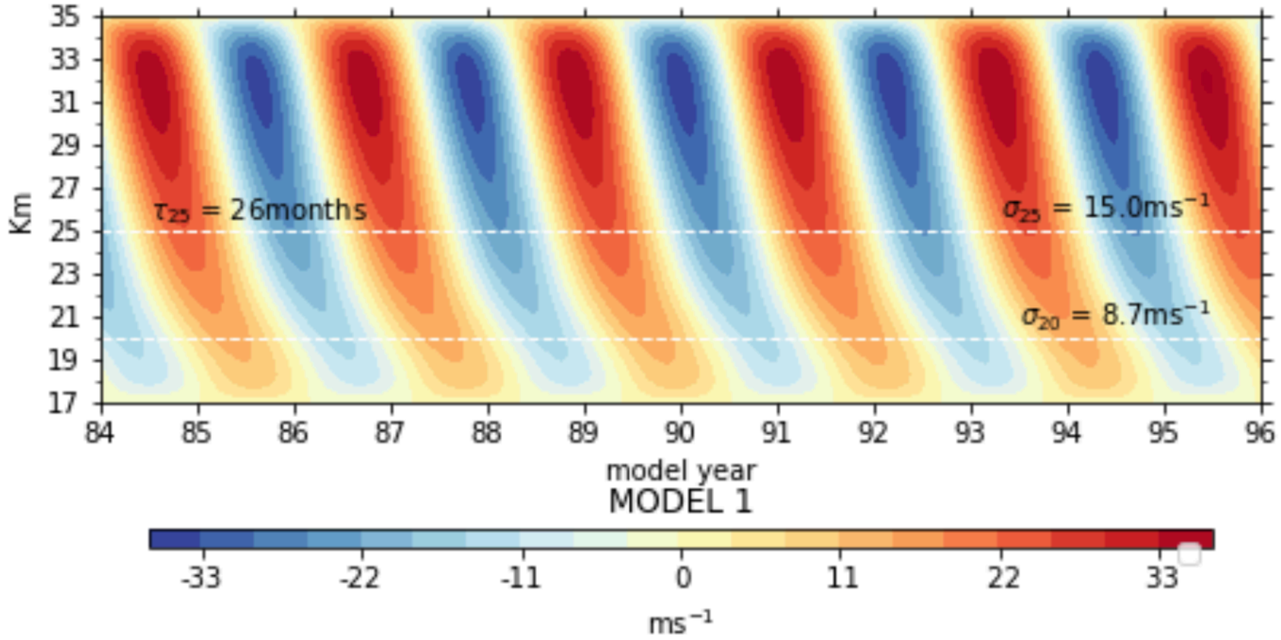
- Offline: Machine Learning model's prediction on test dataset should be satisfactory. For offline performance we mainly use R^2(level-wise mean) / RMSE as the metric.
- Online: Machine Learning model should function similarly as the Physical Model when inserted into the PDE, with better accuracy and efficiency (hopefully).

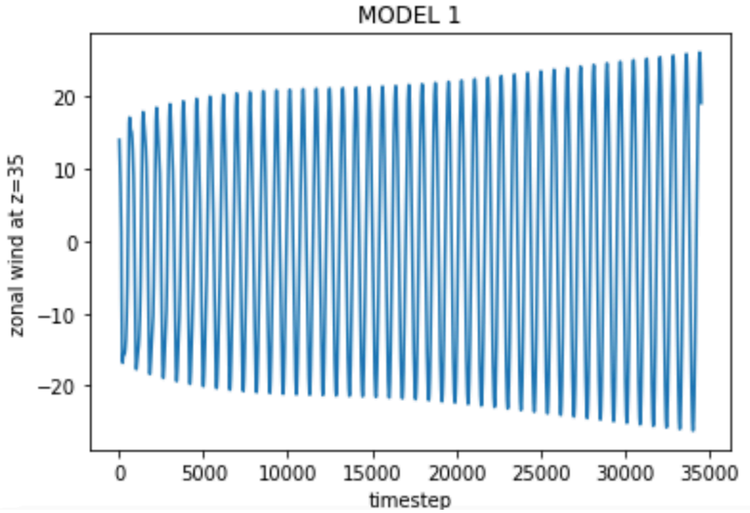


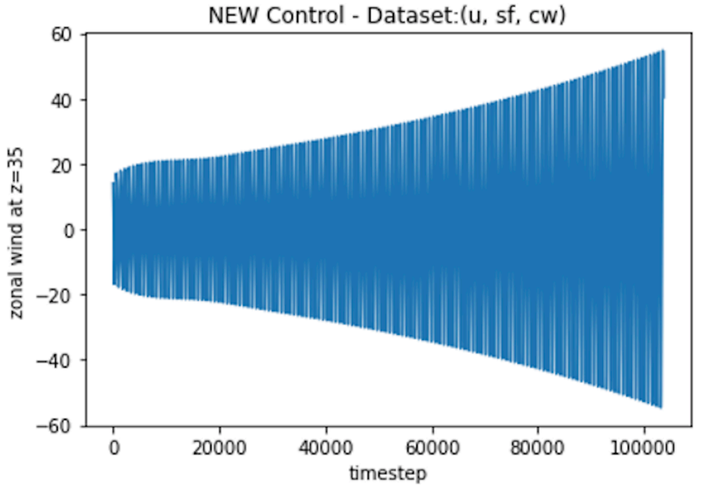
Online Emulation

Example: Linear Models (Baseline)

R^2(Linear Model) $= 0.88$, with 'plausible' Online Emulation performance







Method: Supported Vector Machine

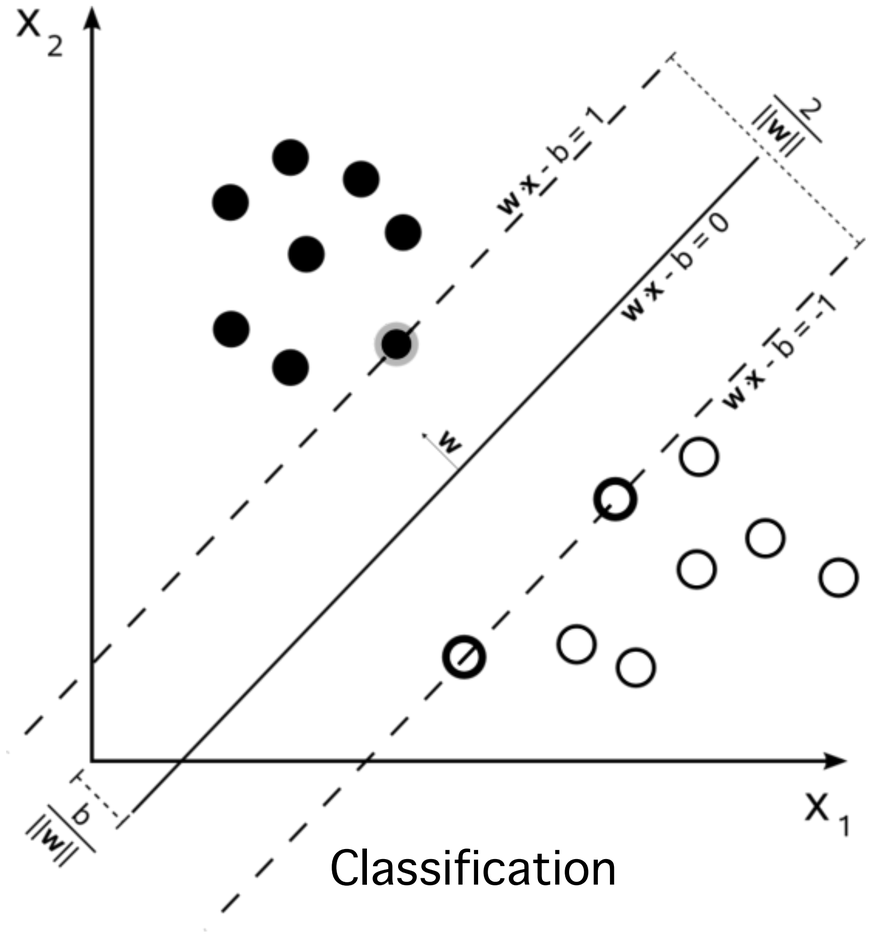
Machine Learning Methods

Essentially a regression problem

- Linear Models (Baseline)
- Regression Trees/Random Forest
- Fully-Connected NN
- CNN/Encoder-Decoder
- Supported/Relevance Vector Machine for Regression

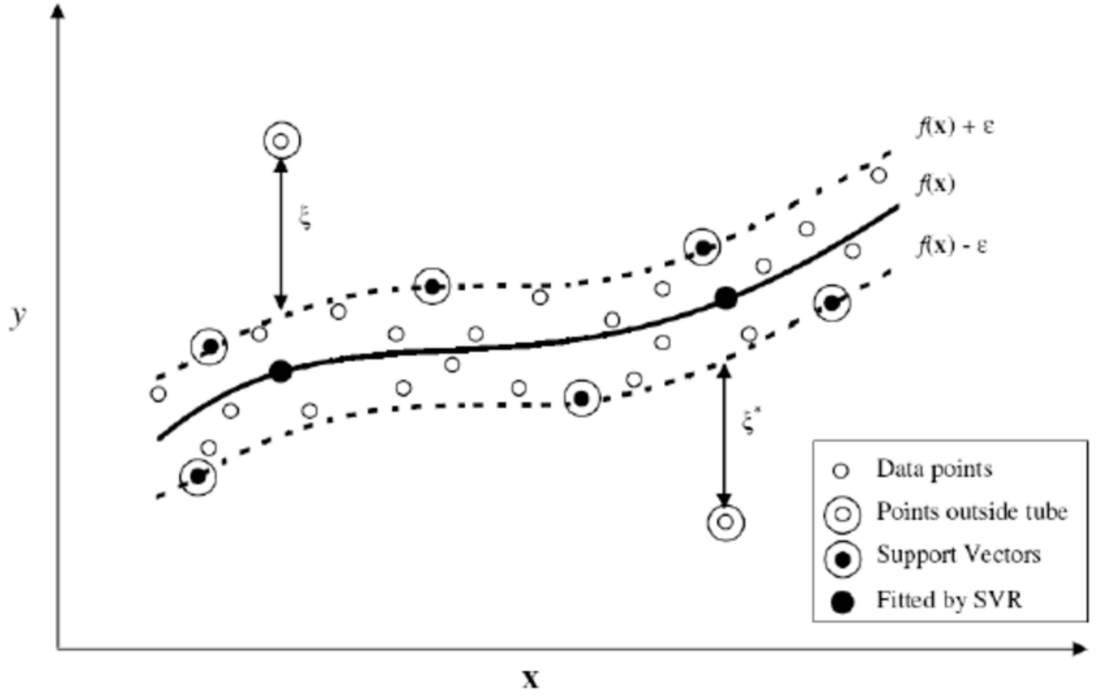
Supported Vector Regression (SVR)

Intuition inherited from classification. Want to find a function such that all targets lies in the ϵ-tube of the function



Classification

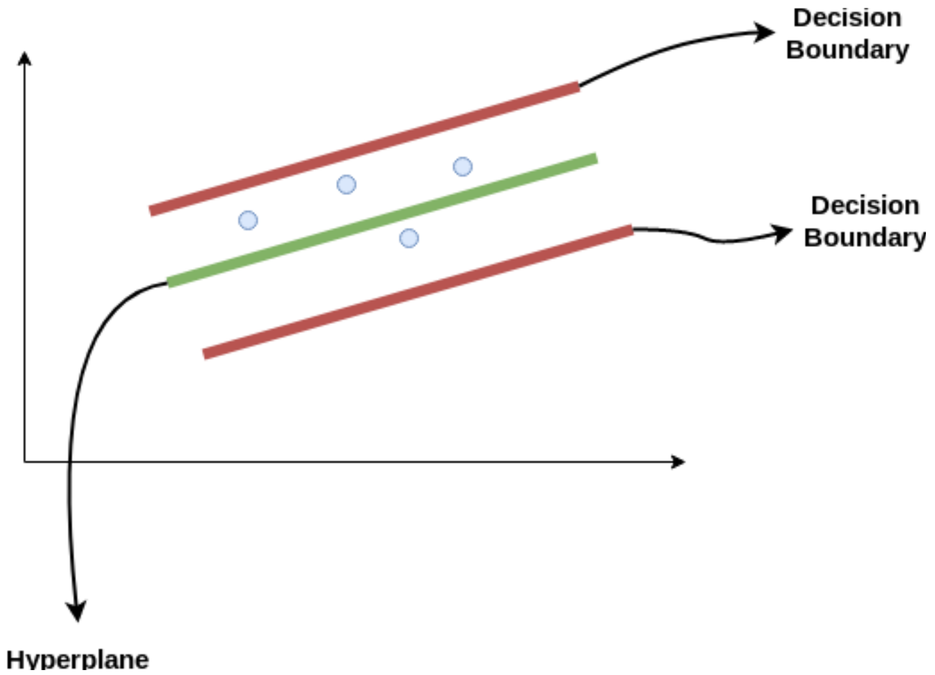
⇒



Regression

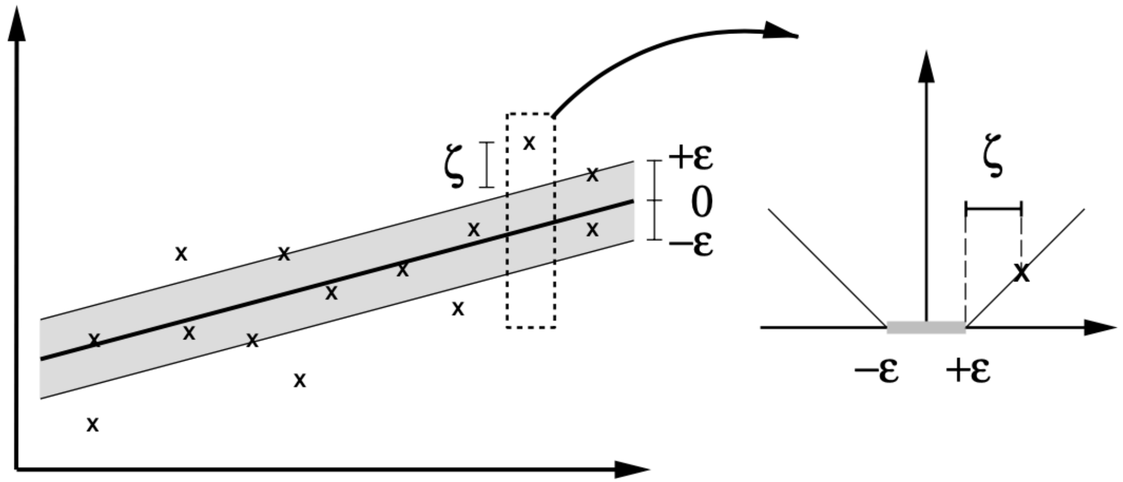
SVR starts with a 1-D linear estimator…

- Dataset $\{x_i, y_i\}_{i=1}^{N}$ ($x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, assume in linear relation)
- A linear estimator: $f(x) = \langle w, x \rangle + b$
- Restrictions: $|f(x_i) - y_i| \leq \epsilon, \forall i$ (ϵ is a hyper-parameter)
- optimize w by maximizing radius of the tube, which is the $\frac{\epsilon}{\|w\|}$ (slope)
- $\min \frac{1}{2}\|w\|^2$ in practice.



Supported Vector regression for linear data

Drawback: Sensitivity to Outliers



$\Rightarrow$

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

Subject to: $\forall i$,

$$y_i - (\langle w, x_i \rangle + b) \leq \epsilon + \xi_i$$
$$(\langle w, x_i \rangle + b) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i \geq 0$$
$$\xi_i^* \geq 0$$

- Remedy: allow the y_i to be ξ_i - away from the tube(boundaries).
- We treat ξ_i as the penalty term in the loss function, with coefficient C (hyper-parameter).

Convex Optimization: Langrange Multipliers

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

Subject to $\forall i$,

$$y_i - (\langle w, x_i \rangle + b) \leq \epsilon + \xi_i$$
$$(\langle w, x_i \rangle + b) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i \geq 0$$
$$\xi_i^* \geq 0$$

With Lagrange Multipliers:

$\{\alpha_i, \alpha_i^*, \eta_i, \eta_i^*\}$

$\Rightarrow$

$$L := \frac{1}{2}\|w\|^2 + C \cdot \sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i - \eta_i^* \xi_i^*)$$
$$- \sum_i \alpha_i (\epsilon + \xi_i - (y_i - \langle w, x_i \rangle + b))$$
$$- \sum_i \alpha_i^* (\epsilon + \xi_i^* - (y_i - \langle w, x_i \rangle + b))$$

Subject to $\forall i$,

$$\alpha_i, \alpha_i^*, \eta_i, \eta_i^* > 0$$

Lagrange Multipliers with K-K-T condition

$$L := \frac{1}{2}\|w\|^2 + C \cdot \sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i - \eta_i^* \xi_i^*)$$
$$- \sum_i \alpha_i (\epsilon + \xi_i - (y_i - \langle w, x_i \rangle + b))$$
$$- \sum_i \alpha_i^* (\epsilon + \xi_i^* - (y_i - \langle w, x_i \rangle + b))$$

$\Rightarrow$

$$\max L(\alpha) := -\frac{1}{2} \sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle$$
$$-\epsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i (\alpha_i - \alpha_i^*)$$

Subject to $\forall i$:

$$\sum_i (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

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

K-K-T Conditions:

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

$$\partial_w L = w - \sum_i (\alpha_i - \alpha_i^*) x_i = 0$$

$$\partial_{\xi_i^{(*)}} = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0$$

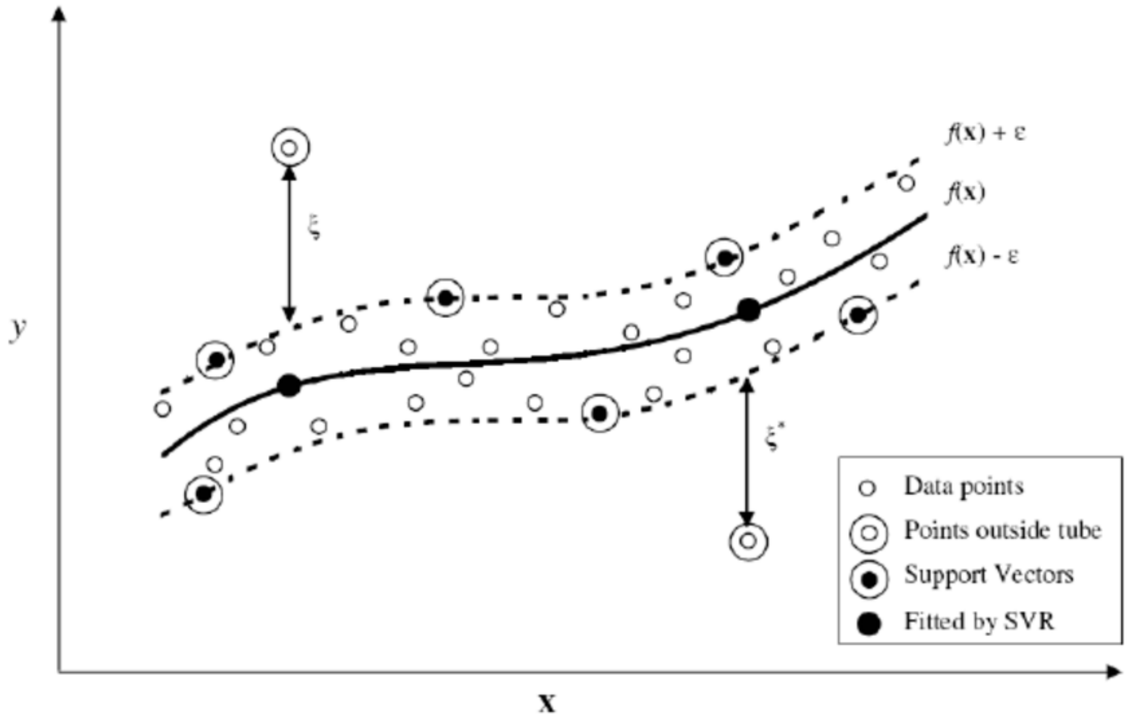
Finally,

$$f(x) = \langle x, w \rangle + b = \sum_i (\alpha_i - \alpha_i^*)\langle x, x_i \rangle + b$$

Supported Vectors

$$f(x) = \sum_i (\alpha_i - \alpha_i^*)\langle x, x_i \rangle + b$$

- After optimization, most of the $(\alpha_i - \alpha_i^*)$ are zero (sparse estimator).
- Vectors with nonzero coefficients are called supported vectors
- The model is uniquely determined by the supported vectors and their coefficients.
- Usually, supported vectors are those near the boundary



Nonlinearity: High-dimensional space

- Main Idea: Convert nonlinear case into linear case by mapping the feature space $\{x_i\} \subset \mathbb{R}^d$ to a higher dimension.

- For example, define feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^3$, with

$$\phi(x = [x^1, x^2]^T) := [(x^1)^2, x^1 x^2, (x^2)^2]^T$$

- Goal: With the help of ϕ, the data are now in linear relation.

- Question: How do we find ϕ?

ShortCut: Kernel Function

- The only way $\{x_i\}$ involves in optimization and the predicting function is via inner products.
- We define the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, with

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

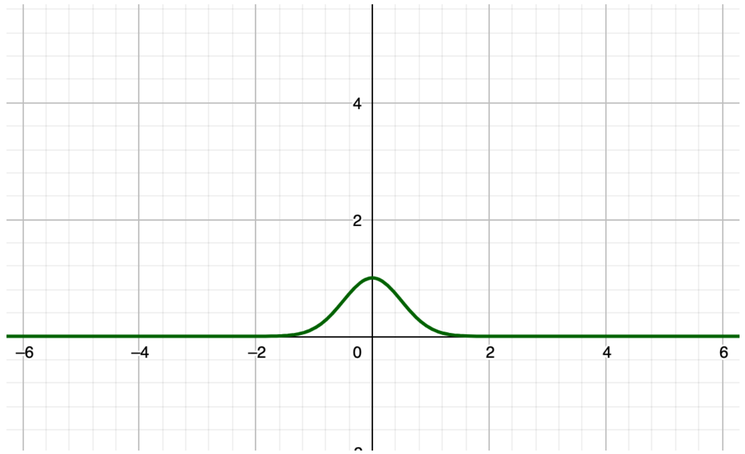
- It suffices use the kernel function to replace all inner products, i.e.

Optimization: $\max L(\alpha) := -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) - \epsilon\sum_i(\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*)$

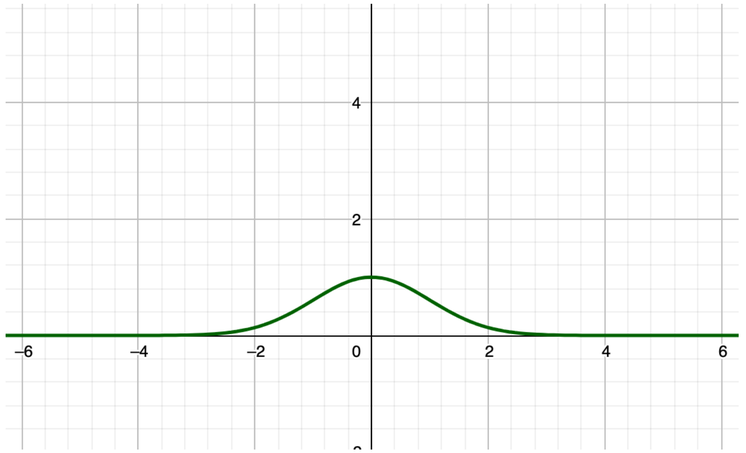
Predicting: $f(x) = \sum_i (\alpha_i - \alpha_i^*)K(x, x_i) + b$

Radial Basis Function kernel (RBF)

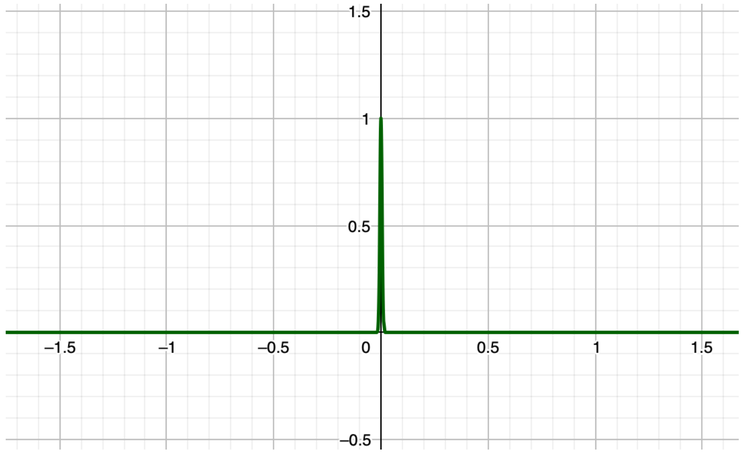
$K(\cdot, \cdot) = \exp\{-\gamma \| \cdot - \cdot \|^2\}$, γ is a hyperparameter



$\gamma = 0.5$



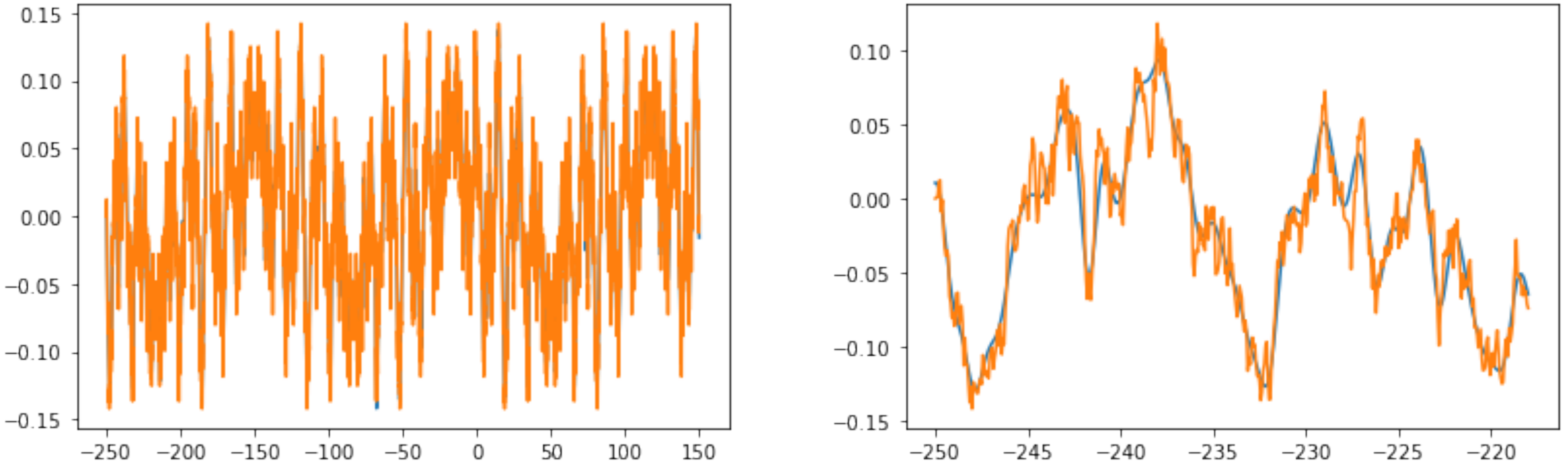
$\gamma = 2.0$



$\gamma \to +\infty$

$$f(x) = \sum_i (\alpha_i - \alpha_i^*) K(x, x_i) + b$$

Power of SVR with RBF kernel



Approximations of trajectory of Brownian motion by SVR with RBF kernel

Experiments with SVR

SVR in QBO-1d

Things we need to know before get started

- Goal: A function $\mathtt{S}_{\mathtt{ML}}(\mathtt{u}, \mathtt{sf}, \mathtt{cw})$.
- Dataset:

  Feature: An augmented u(zonal wind) by stacking $\mathtt{u}$ and $\mathtt{sf}, \mathtt{cw}$, i.e. $\mathtt{U} = [\mathtt{u}^1, \cdots, \mathtt{u}^{73}, \mathtt{sf}, \mathtt{cw}]$

  Target: $\mathtt{s} = [\mathtt{s}^1, \mathtt{s}^2, \ldots . \mathtt{s}^{73}]$ generated in control run (with physical parametrization)
- Multi-output Regression: Since SVR does not support multi output regression natively, we do dimensional-wise independent SVR.

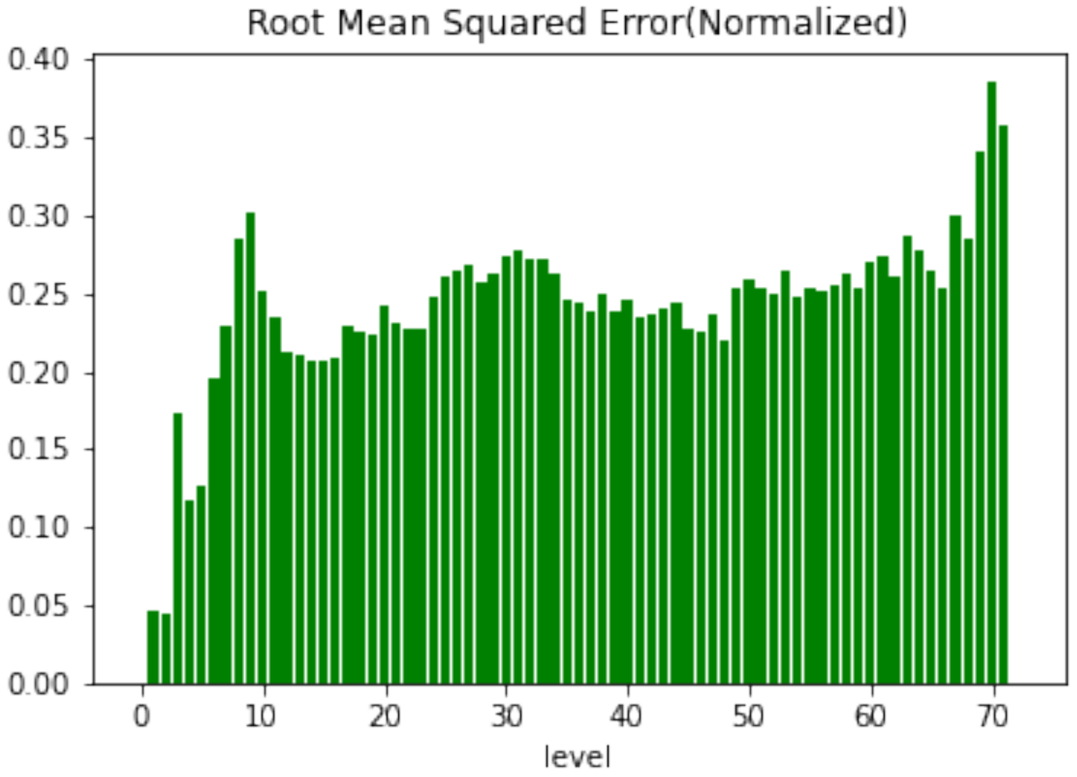
(*)Comment: with this setting, the model ignores the both spatial and temporal correlation.

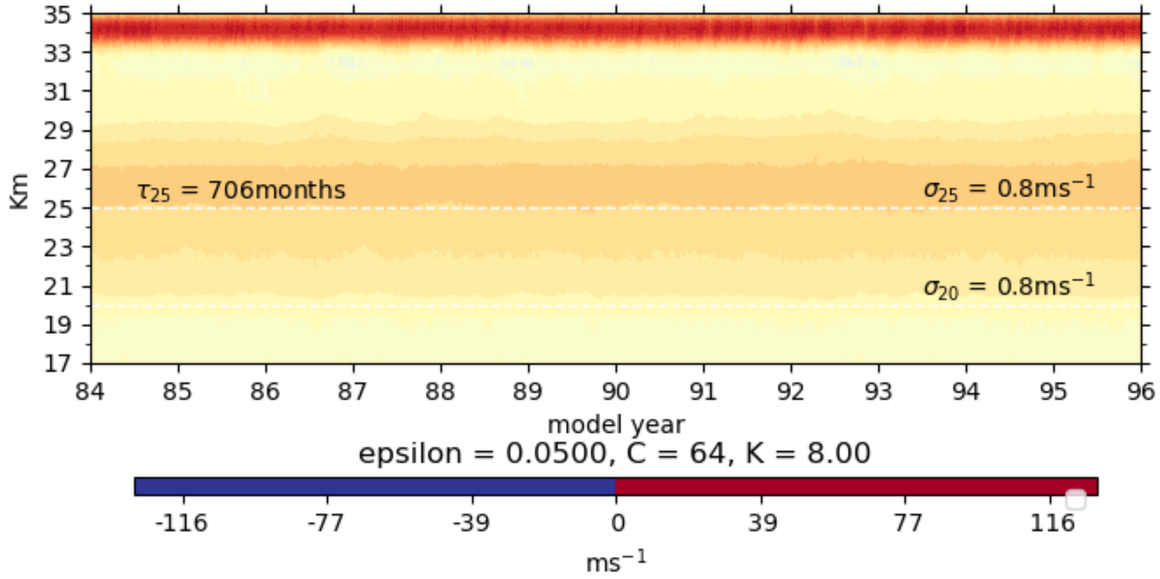
Goals Revisited

- Offline: (at least) Better than the baseline i.e. $R^2(\text{SVR}) \geq R^2(\text{Linear Models})$
- Online:
    1. A periodic oscillation, with reasonable QBO statistics: period(τ_{25}), higher-level amplitude(σ_{25}), lower-level amplitude(σ_{20})
    2. Ability of Generalization
    3. Efficiency.

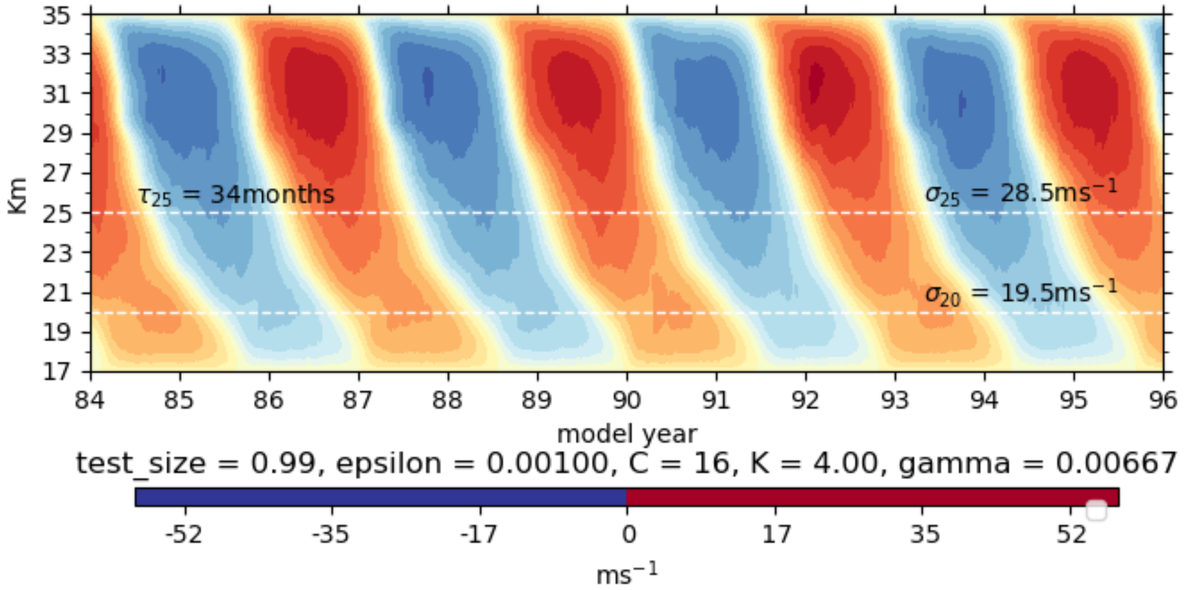
Offline performance of SVR

- Normalization is highly-recommended (even necessary)
- With normalization, the SVR's model can reach average $R^2 = 0.95$ ➡️

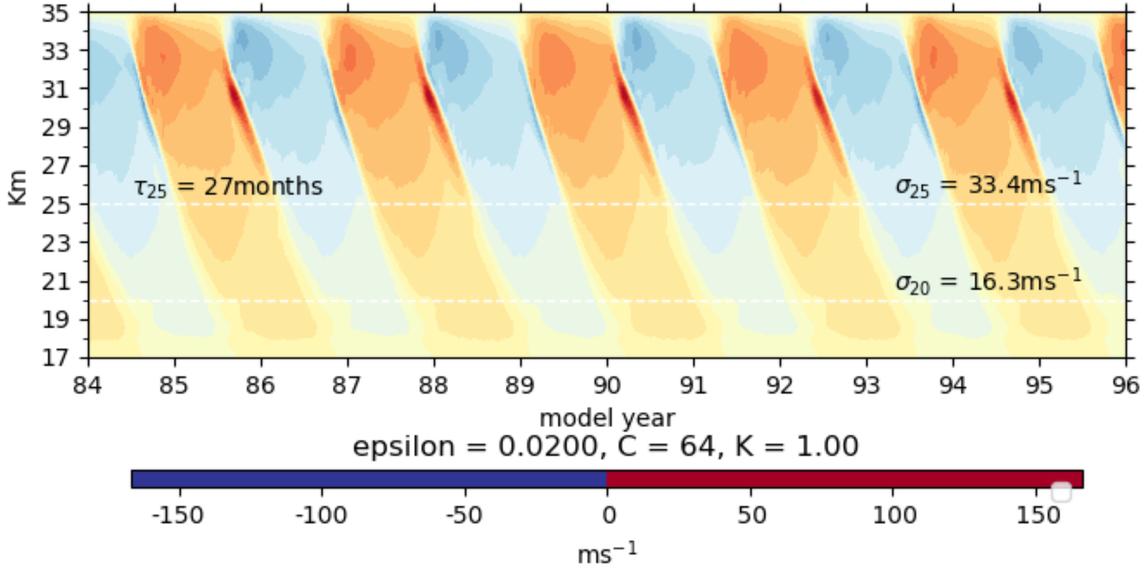




(R^{2} = 0.91) Non-periodic Oscillation (Most of the cases)



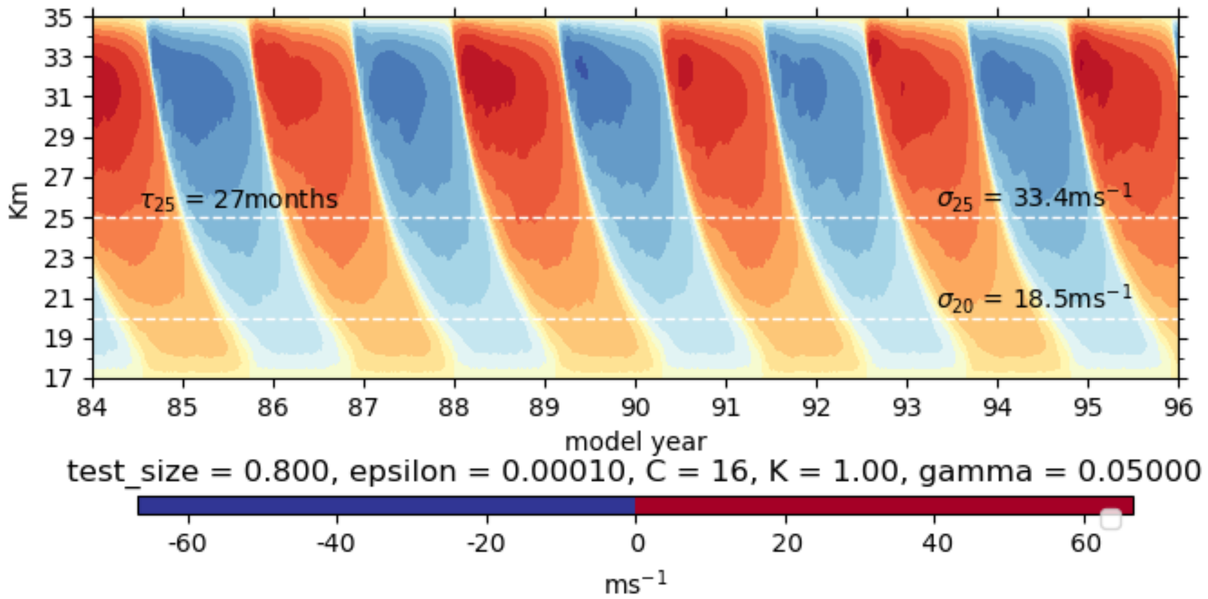
(R^{2} = 0.64) Not accurate emulation

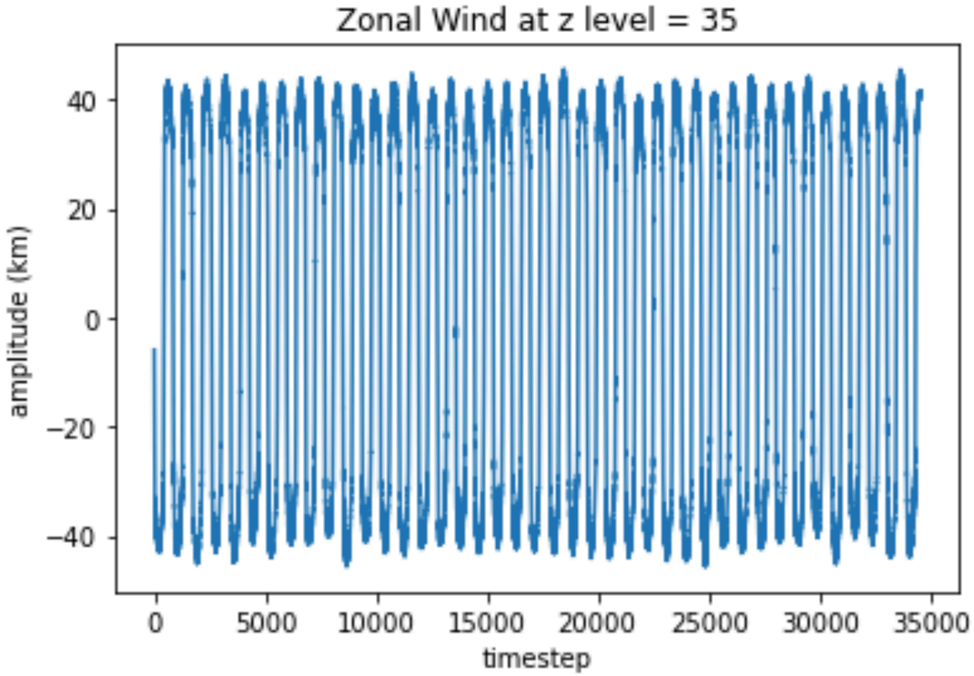


(R^{2} = 0.94) freaky Oscillation

‘Bad’ Online Emulations of some SVR models, despite good offline performance.

Successful Online Emulation





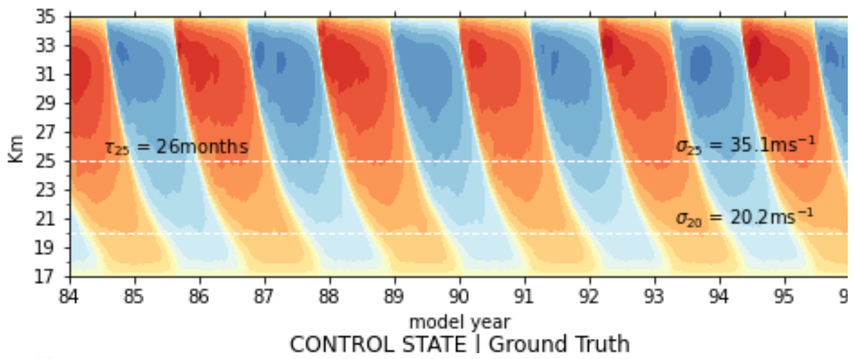
- Of all the hyper-parameters, ϵ and γ are most important in terms of online emulation.
- ⚠️ Relationship with Offline performance.

Generalization

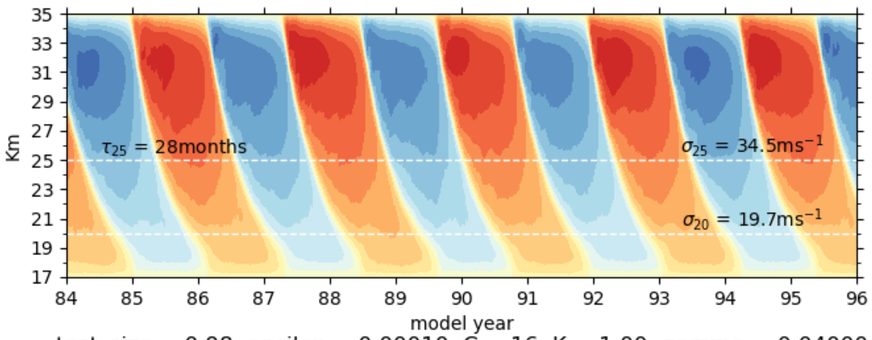
SVR model have some ability of generalization

- The model is expected to adapt to potential climate change
- Mathematically, it means changing the distribution of F_{S_0}(`sf`)and c_w(`cw`)
- Goal: online emulation of SVR model will change in a similar way as the physical model changes.
- SVR model can handle the perturbation cases qualitatively correct.

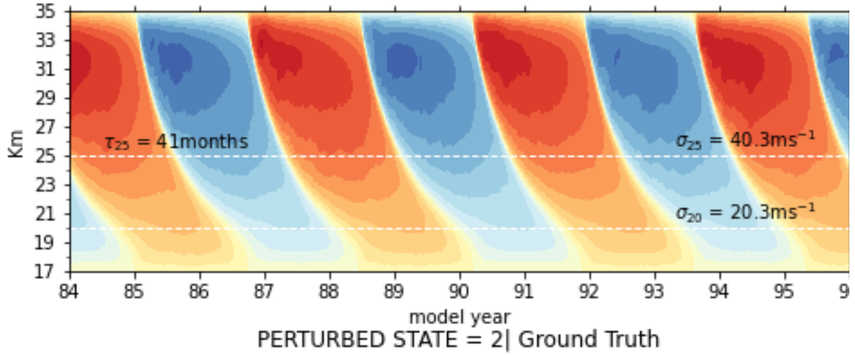
Generalization of SVR Model



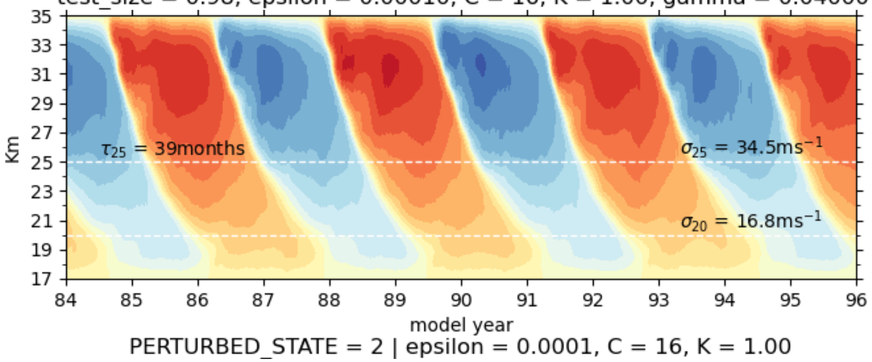
CONTROL STATE | Ground Truth



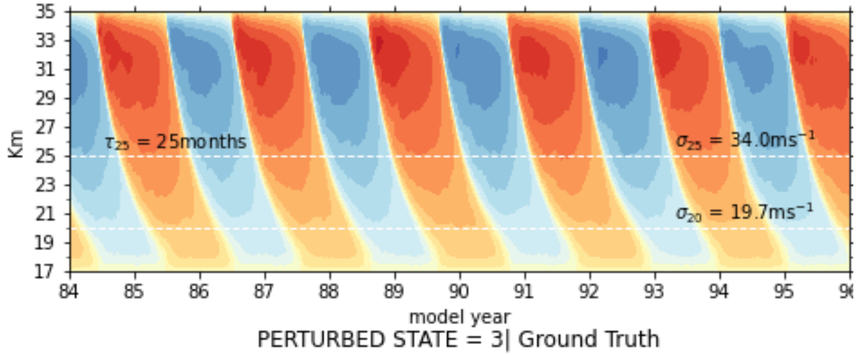
test_size = 0.98, epsilon = 0.00010, C = 16, K = 1.00, gamma = 0.04000



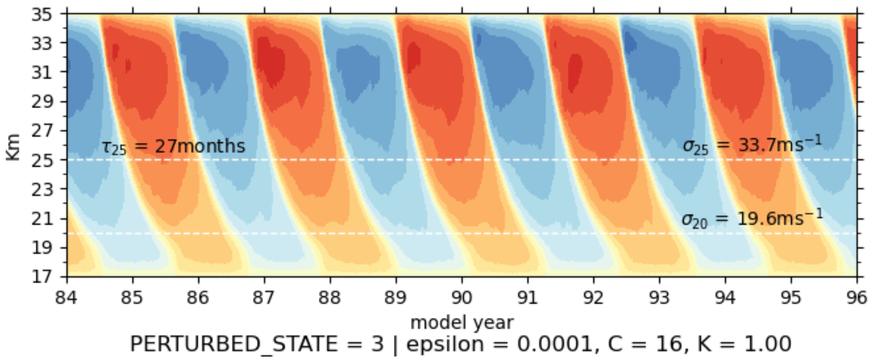
PERTURBED STATE = 2| Ground Truth



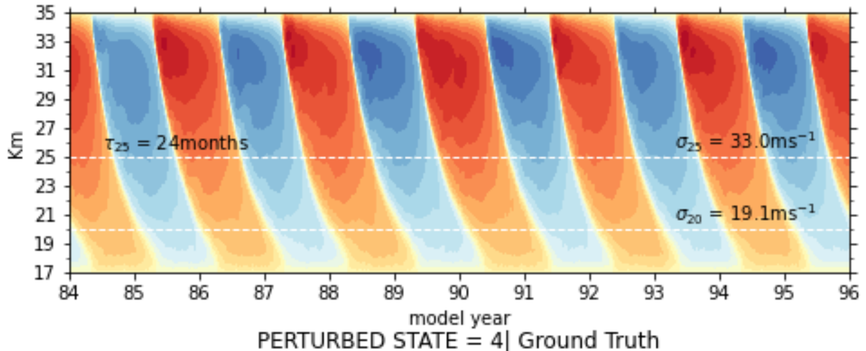
PERTURBED_STATE = 2 | epsilon = 0.0001, C = 16, K = 1.00



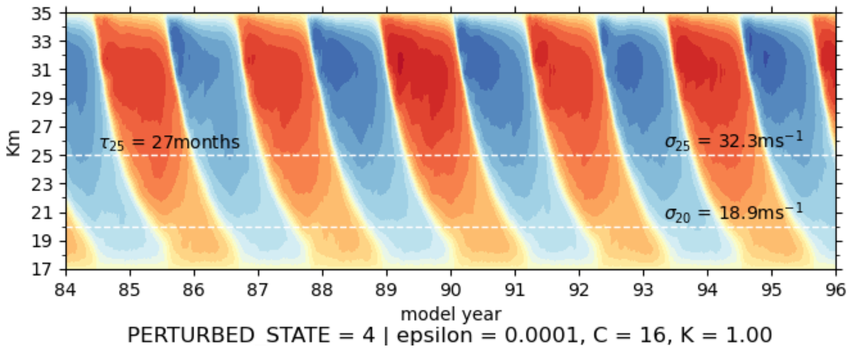
PERTURBED STATE = 3| Ground Truth



PERTURBED_STATE = 3 | epsilon = 0.0001, C = 16, K = 1.00



PERTURBED STATE = 4| Ground Truth



PERTURBED_STATE = 4 | epsilon = 0.0001, C = 16, K = 1.00

⬅️ Ground Truth

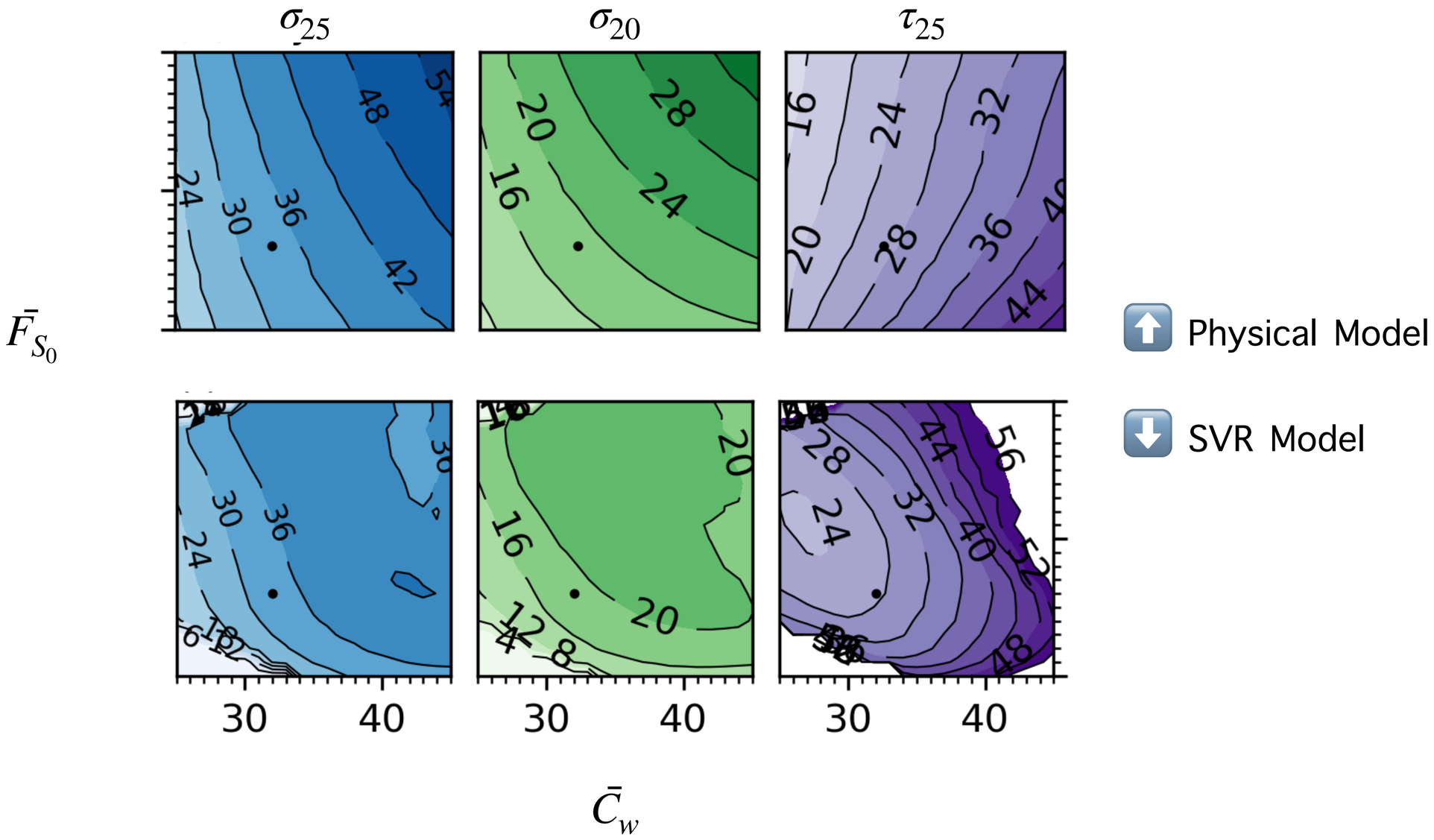
➡️ SVR Model

STATE=2: Biased Mean

STATE=3: Biased Variance

STATE=4: Anti-Correlated

Generalization



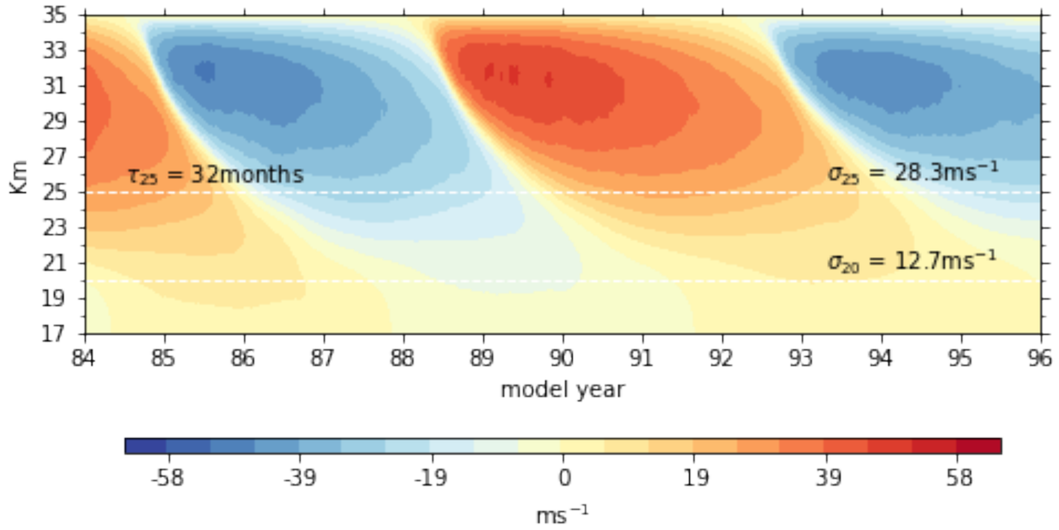
Trials to improve generalization

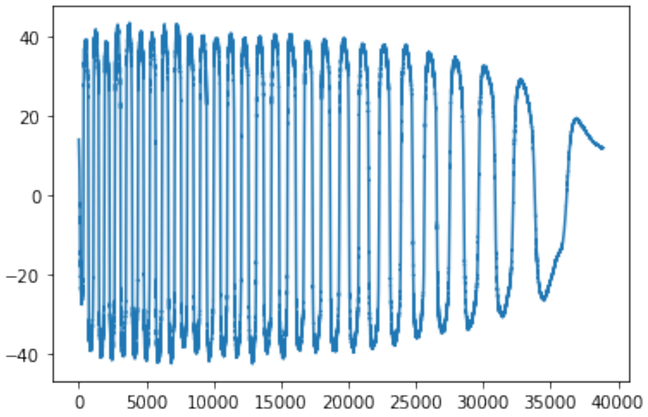
- Intuition: Using a more comprehensive dataset

  Changing the the mean and variance of sf and cw with time.

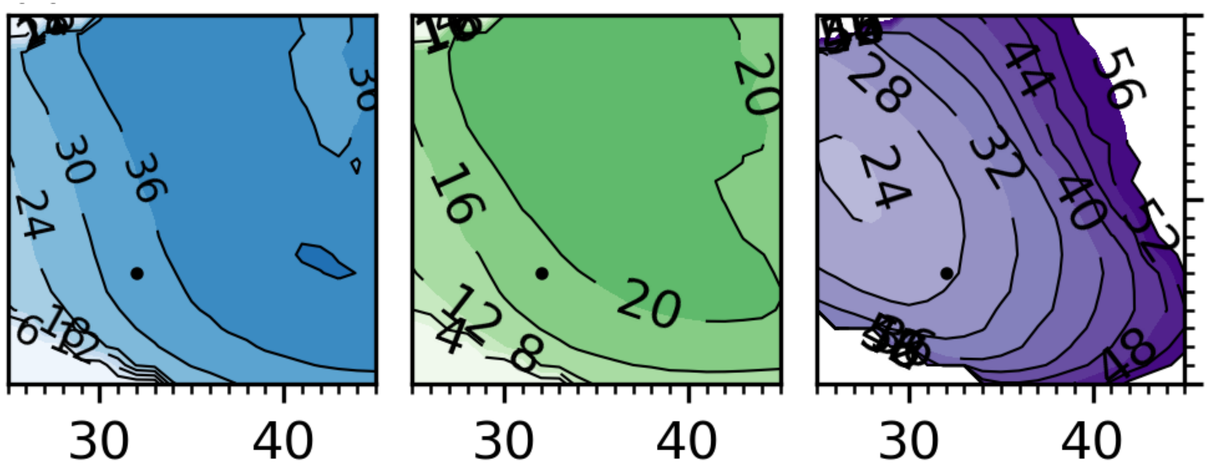
  Let the model see how the physical model adapted to changing distribution of `sf` and `cw`.

  As a compensation, we don't let the model see the oscillation with fixed period.



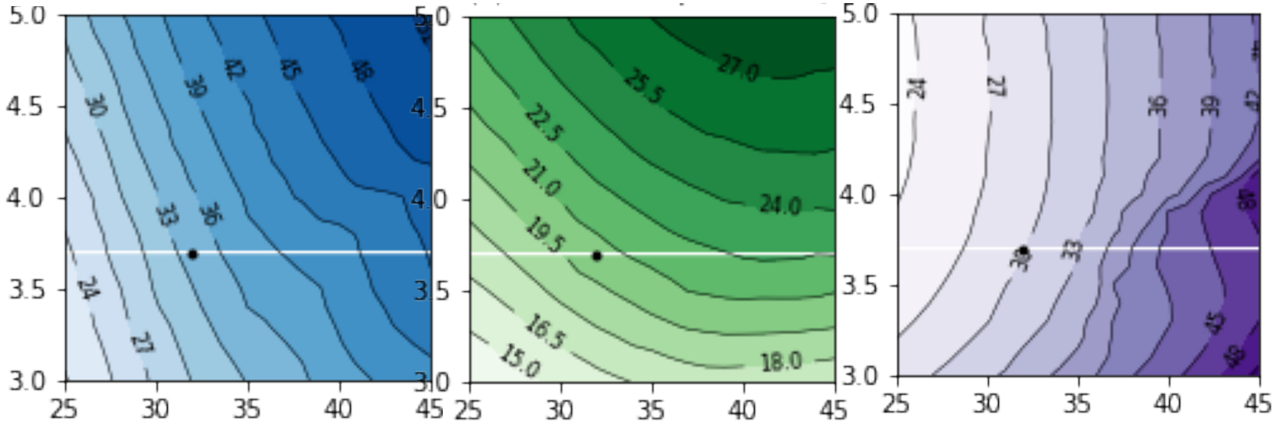


The results are good, but may not be legitimate



⬆️ Regular Data

⬇️ Comprehensive Data



Efficiency

Counter-intuitively, SVR is not good at the efficiency (compared with Neural Nets)

- Efficiency $\Longleftrightarrow$ Execution Time Complexity
- Major complexity: exponential function inside kernel function
- Kernel function is computed for each supported vector(*), i.e. Execution Time $\propto$ #SV
- By (1) $\uparrow \epsilon$ (2) $\downarrow \gamma$ (3) $\downarrow$ #training data , #SV decreases
- ⚠️ #SV $\neq$ #trainable parameters (thus difficult to do the Apple-to-Apple comparison with NN)

(*) Not rigorous, sometimes we need to compute for a non-supported vector, will explain later.

Trainable parameters

$$S(u) = \begin{bmatrix} \alpha_1^1 & \alpha_1^2 & \alpha_1^3 & \alpha_1^4 & \dots & \alpha_1^{n-4} & \alpha_1^{n-3} & \alpha_1^{n-1} & \alpha_1^n \\ \alpha_2^1 & \alpha_2^2 & \alpha_2^3 & \alpha_2^4 & \dots & \alpha_2^{n-4} & \alpha_2^{n-3} & \alpha_2^{n-1} & \alpha_2^n \\ \vdots & & & & \ddots & & & & \vdots \\ \alpha_{74}^1 & \alpha_{74}^2 & \alpha_{74}^3 & \alpha_{75}^4 & \dots & \alpha_{75}^{n-4} & \alpha_{74}^{n-3} & \alpha_{74}^{n-1} & \alpha_{74}^n \\ \alpha_{75}^1 & \alpha_{75}^2 & \alpha_{75}^3 & \alpha_{75}^4 & \dots & \alpha_{75}^{n-4} & \alpha_{75}^{n-3} & \alpha_{75}^{n-1} & \alpha_{75}^n \end{bmatrix} \begin{bmatrix} K(u, u_1) \\ K(u, u_2) \\ \vdots \\ K(u, u_{n-1}) \\ K(u, u_n) \end{bmatrix} + b$$

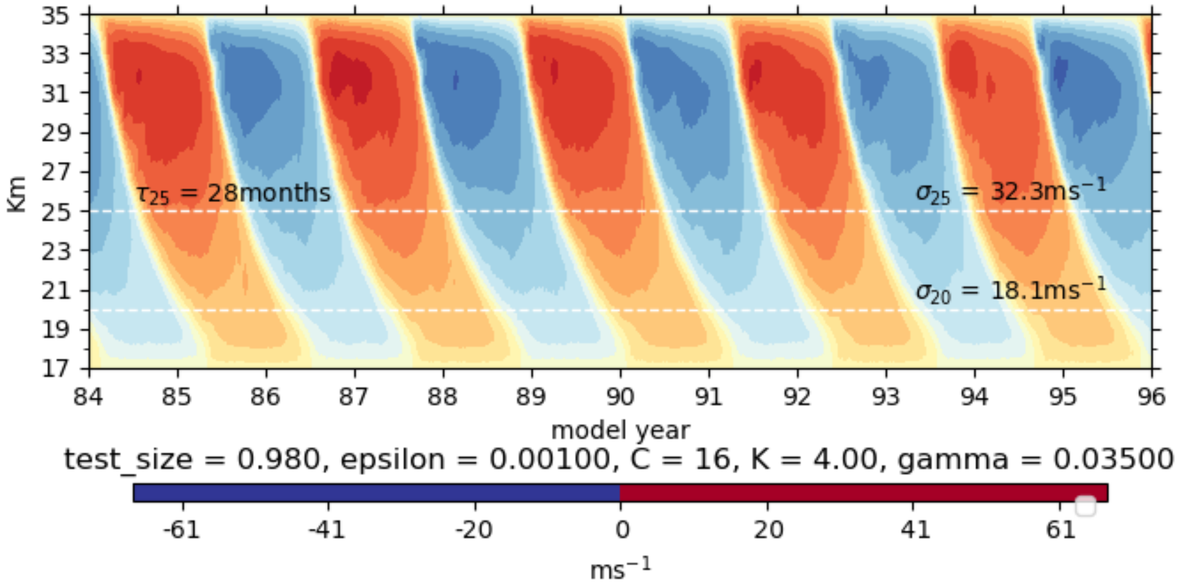
- Set of trainable parameters: $\{\alpha_i^j, b\}$, most of which are zero.
- # trainable parameters is uniquely determined by the size of training data that we put in.

Trade off

$$S(u) = \begin{bmatrix} \alpha_1^1 & \alpha_1^2 & \alpha_1^3 & \alpha_1^4 & \dots & \alpha_1^{n-4} & \alpha_1^{n-3} & \alpha_1^{n-1} & \alpha_1^n \\ \alpha_2^1 & \alpha_2^2 & \alpha_2^3 & \alpha_2^4 & \dots & \alpha_2^{n-4} & \alpha_2^{n-3} & \alpha_2^{n-1} & \alpha_2^n \\ \vdots & & & & \ddots & & & & \vdots \\ \alpha_{74}^1 & \alpha_{74}^2 & \alpha_{74}^3 & \alpha_{75}^4 & \dots & \alpha_{75}^{n-4} & \alpha_{74}^{n-3} & \alpha_{74}^{n-1} & \alpha_{74}^n \\ \alpha_{75}^1 & \alpha_{75}^2 & \alpha_{75}^3 & \alpha_{75}^4 & \dots & \alpha_{75}^{n-4} & \alpha_{75}^{n-3} & \alpha_{75}^{n-1} & \alpha_{75}^n \end{bmatrix} \begin{bmatrix} K(u, u_1) \\ K(u, u_2) \\ \vdots \\ K(u, u_{n-1}) \\ K(u, u_n) \end{bmatrix} + b$$

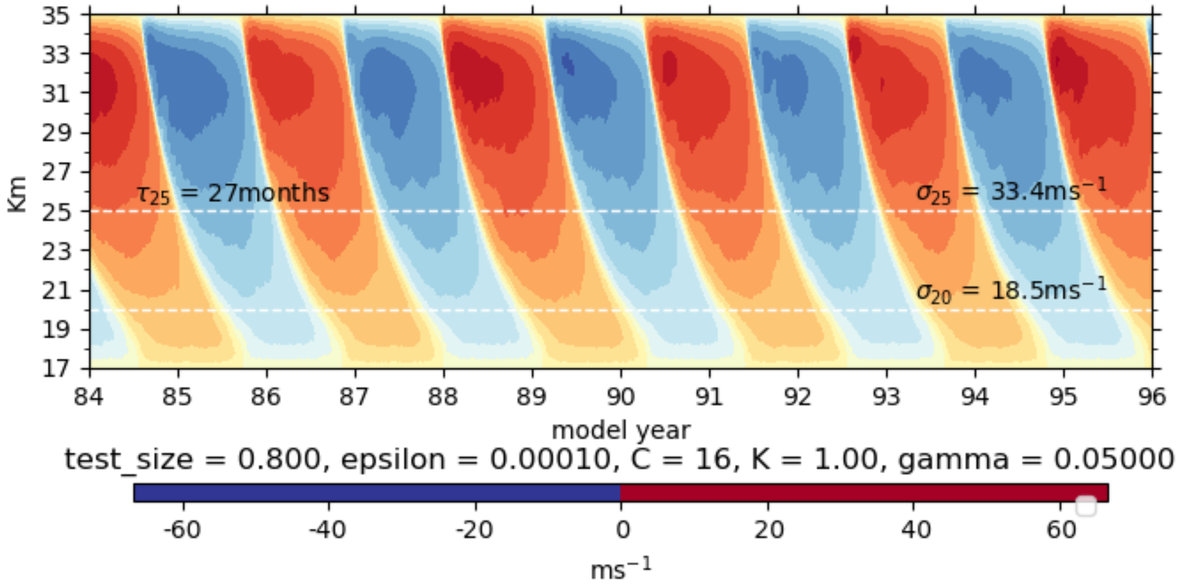
- Case 1: Big but sparse
- Case 2: Small but dense
- Trade Off: Sparsity & #training data feed in

Punchline: ‘Tiny’ training dataset + ‘harsh’ optimization condition



#SV = #training data = 692

trainable parameters = 5×10^4



#SV = #training data = 6900

trainable parameters = 5×10^5

Relevance Vector Regression

An updated version of SVR

- Intuition of RVR: the functional expression of SVR + Baysian Statistcs

  Advantages: Better offline performance with incredibly sparse model (thus being very efficient)

  Disadvantages: Incredibly long training time

- However, despite RVR's efficient and good offline performance ($R^2 = 0.97$), it fails in online emulations.

Implications of RVR's Failure

Key: RVR can be viewed as a special case of SVR

- The failure of RVR refutes the assumption that a model with better offline result has a larger probability to have a better online result.

- The other assumption is: RVR models don't have enough supported/Relevance vectors to do the online emulation.

  However, when increasing the number of relevance vector machine manually (adjusting γ), RVR models' online emulation performances are not improved.

Conclusion

- Supported Vector Regression model can emulate physics-based gravity wave parametrization (in QBO-1d model)
- The relation between offline performance and online performance is unknown.
- Supported Vector Regression model has some ability of generalization.
- A small but dense model outperforms a big but sparse model in efficiency with comparable online emulation results. However, SVR is essentially not a method with efficiency.

References

Appendix

Numerical Scheme

- We generate u by solving the advection-diffusion PDE (by discretization + backward Euler)
- For F_{S_0} and c_w, we use a bivariate log-normal distribution to generate two arrays. For simplicity, denote these 2 arrays sf and cw.
- with 5 parameters (mean & variance for both, and correlation of them).
- Therefore, the source term S is a function of $\{u, sf, cw\}$.
- At each time step, we record the value of the source term s, where `s[i]` $=$ `S(u[i], sf[i], cw[i])`, with time step i.

RBF kernel

- https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf

Mercer's condition

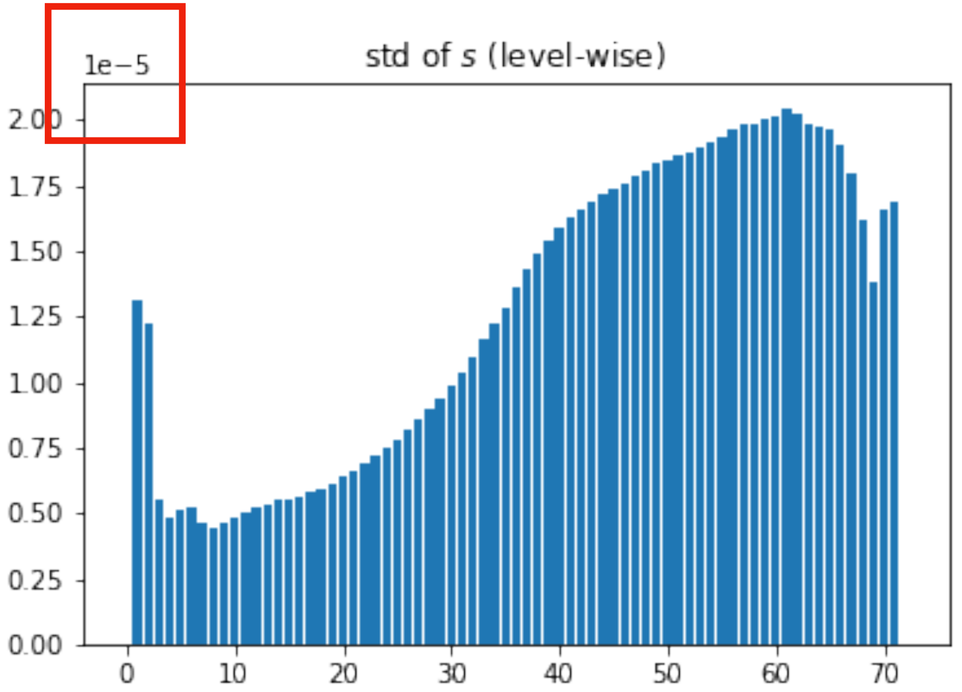
- A kernel is valid if and only if it can be interpreted as the inner product in higher dimensional
- The kernel function represents an inner product in higher-dimensional as long as it satisfies the Mercer's theorem (sufficient condition).
- Theorem(Mercer): Suppose K is a continuous symmetric non-negative definite kernel. Then there is an orthonormal basis $\{e_i\}_i$ of $L^2[a,b]$ consisting of eigenfunctions of T_k such that the corresponding sequence of eigenvalues $\{\lambda_i\}_i$is nonnegative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[a,b]$ and K has the representation

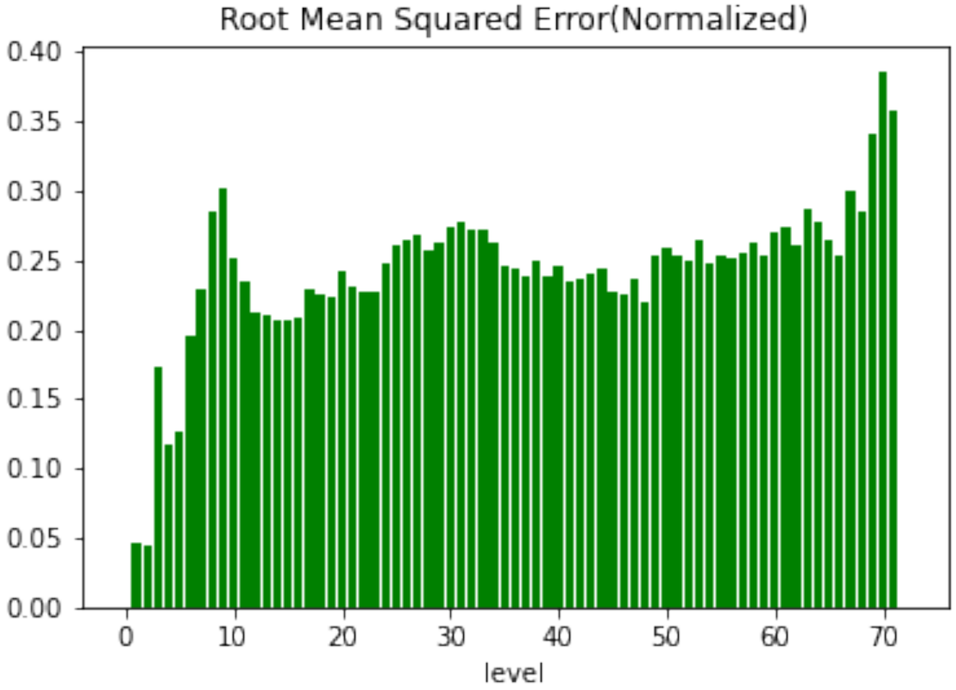
$$K(s,t) = \sum_{j}^{\infty} \lambda_j e_j(s) e_j(t)$$

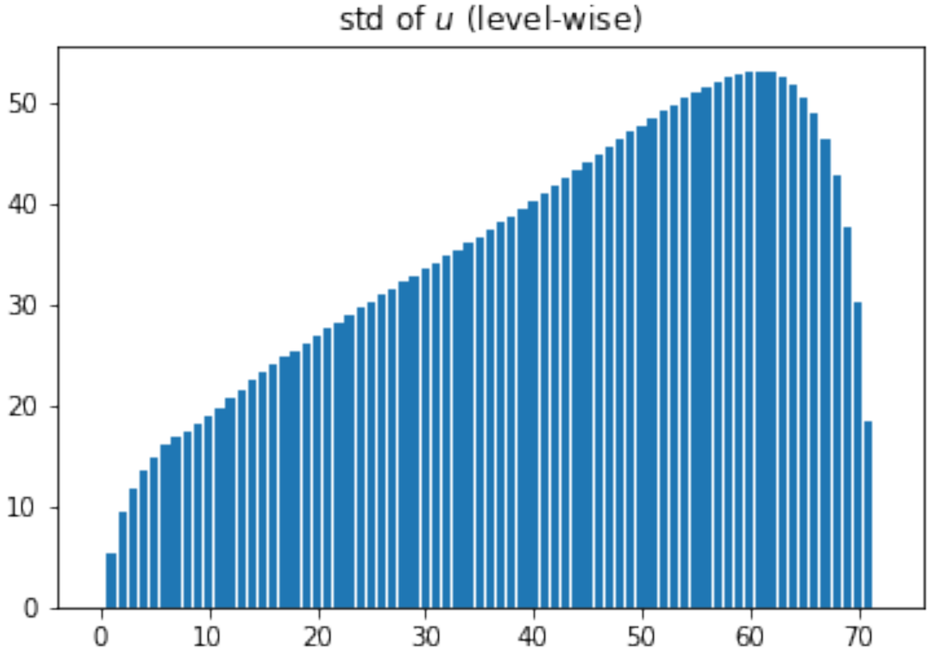
Where the convergence is absolute and uniform

Offline performance of SVR

- Normalization is highly-recommended (even necessary) ➡️
- With normalization, the SVR's model can reach $R^2 = 0.95$ ⬇️







Hyper-parameters

- Some Obvious Hyper-Parameters:

  1. ϵ, in the restriction, $|f(x_i) - y_i| \leq \epsilon, \forall i$

  2. C, the penalty coefficient, $\frac{1}{2}\|w\|^2 + C \cdot \sum_i (\xi_i + \xi_i^*)$

  3. K, in augmented u, $U := \begin{bmatrix} u^1 & u^2 & \cdots & u^{73} & K \cdot sf & K \cdot cw \end{bmatrix}^T$

- Some Less-Obvious Hyper-Parameters:

  4. γ, in kernel function $\exp\{-\gamma\|u - u_i\|^2\}$

  5. The amount of the training data we insert.

- Of all the parameters, ϵ and γ are most important to make the online emulation works.

  ϵ represents how strict the optimization is. Lowering ϵ improves the online performance.

  γ represents how localized the kernel function is. A suitable γ helps with the online performance. In particular, It improve the model when the training dataset is small.

Metric: R^2 and rMSE

- $$R^2 = 1 - \frac{\sum_i^N (\hat{y}_i - y_i)^2}{\sum_i^N (y_i - \bar{y}_i)^2}$$
- $$\text{rMSE} = \sqrt{\frac{\sum_i^N (\hat{y}_i - y_i)^2}{N}}$$

- Some literature point out that R^2 is not a good metric for nonlinear regression. However, in our case, since the data is repeating. We treat the variance of the testing data all the same i.e. the denominator are the same.

- In that case, rMSE is equivalent to R^2

IDEAs for improving generalizations

- IDEA 1: Increase the weight of sf and cw (by modifying K).

  Increasing K in $U := \begin{bmatrix} u^1 & u^2 & \cdots & u^{73} & K \cdot sf & K \cdot cw \end{bmatrix}^T$ does make model more sensitive to the change in sf and cw. But does not contribute significantly to the ability of generalization.

- IDEA 2: Using a more comprehensive dataset

  Changing the the mean and variance of `sf` and `cw` with time.