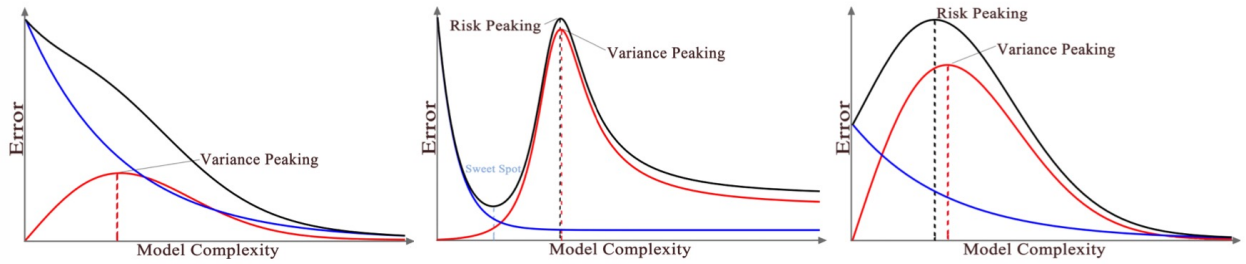
1 Summary of Zitong's paper

1.1 Double descent phenomenon



(a) Case 1 (b) Case 2 (c) Case 3

Figure 1. Typical cases of expected risk curve (in black) in neural networks. Blue: squared bias curve. Red: variance curve.

Recently, for neural networks and other over-parameterized models, it is often observed that larger models generalize better: while the bias is monotonically decreasing as in the classical theory, the variance is unimodal or bell-shaped: it increases then decreases with the width of the network.

The expected risk (test loss) is the sum of bias and variance, monotonic bias and unimodal variance can lead to three characteristic behaviors, depending on the relative size of the bias and variance.

- If the bias completely dominates, we obtain monotonically decreasing risk curve (see Figure 1(a)).
- If the variance dominates, we obtain a bell-shaped risk curve that first increases then de- creases (see Figure 1(c)).
- The most complex behavior is if bias and variance dominate in different regimes, leading to the double-descent risk curve in Figure 1(b).

The most common behavior in their experiments is the first case (monotonically decreasing risk curve) as bias is typically larger than variance. They can observe the double descent risk curve when label noise is added to the training set, and can observe the unimodal risk curve when they use the generalized bias-variance decomposition for cross-entropy loss.

1.2 Estimators for bias and variance

Estimator for variance: Splitting the training dataset into N non-overlapped sets: $\tau = \tau_1 \cup \cdots \cup \tau_N$, and we use the unbiased estimator:

$$\widehat{var}(x, \tau) = \frac{1}{N-1} \sum_{j=1}^{N} \left\| f(x, \tau_j) - \frac{1}{N} \sum_{j=1}^{N} f(x, \tau_j) \right\|_2^2,$$

and obtain bias by subtracting the variance from the risk.

1.3 Other conclusions

- Previous literatures (Mei & Montanari, 2019) suggests that both the risk and the variance achieves a peak at the interpolation threshold ($n = p = \texttt{rff\_dim}$).
- Deeper models decrease bias and increase variance for both in-distribution and out-of-distribution data.
- Conjecture: as the model complexity approaches and then goes beyond the data dimension, it is regularization in model estimation (the ridge penalty) that helps bring down the variance.

2 Our Settings

2.1 Estimators for bias and variance

Since we don't have test sample for the variance importance quantities ψ's, I want to calculate the bias and variance directly: calculate the "true" $\psi_{\texttt{true}}$ using GP, and sample M β's from its posterior distribution, and then calculate the corresponding ψ's with β's. For example, for ψ_1:

$$\text{bias}_1^2 = \left(\frac{1}{M} \sum_{j=1}^{M} \psi_{j1} - \psi_{\texttt{true},1} \right)^2,$$

$$\text{var}_1 = \left(\frac{1}{M} \sum_{j=1}^{M} \psi_{j1}^2 \right) - \left(\frac{1}{M} \sum_{j=1}^{M} \psi_{j1} \right)^2.$$

2.2 Simulation settings

- training sample size $n = 1000, 2000, 4000, 7000, 10000$.
- data dimension $d = 7, 10, 15, 20, 25$.
- noise $\epsilon = 0.01, 0.02, 0.04, 0.07, 0.10$.
- random Fourier features dimension $\texttt{rff\_dim} = \texttt{np.linspace}(10 * d, 2 * n, 10)$.

For computational reasons, we won't consider the case when $\texttt{rff\_dim} > n$, but in order to observe the double descent phenomenon, we can experiment it.

2.3 Partial results and current problems

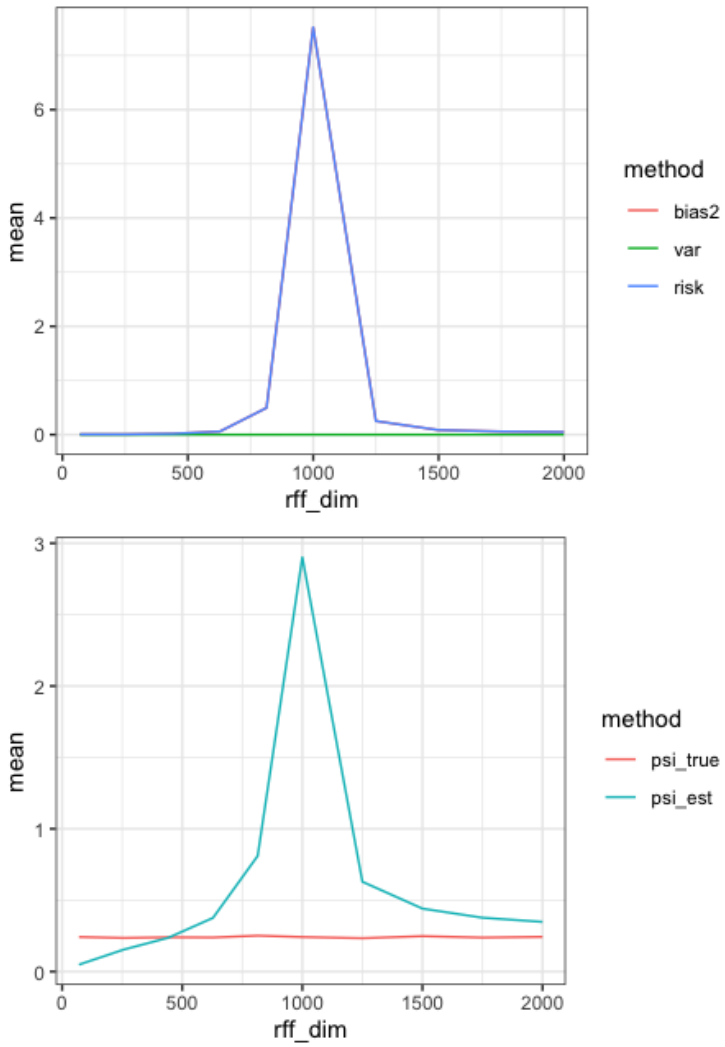


Figure 1: RBF: $d = 7, n = 1000, \epsilon = 0.02$ and repeated for 10 times. Above: average over all data dimension d. Below: for the true non-zero variable x_0.

Questions:

- any problems with the estimators?
- the covariance matrix of β is singular:

```
Traceback (most recent call last):
  File "var_selection.py", line 111, in <module>
    psi_est = m.estimate_psi(Z, psi_true)
  File "/home/wd45/VS_BNN/v02/exp/models.py", line 143, in estimate_psi
    beta_samp = np.random.multivariate_normal(self.beta, self.Sigma_beta, size=n
_samp).T
  File "mtrand.pyx", line 4521, in mtrand.RandomState.multivariate_normal
  File "/home/wd45/deng_env/lib/python3.6/site-packages/scipy/linalg/decomp_svd.
py", line 132, in svd
    raise LinAlgError("SVD did not converge")
numpy.linalg.linalg.LinAlgError: SVD did not converge
```