Social Data Science: Machine Learning & Econometrics

Exercise class 0

February 10, 2020

About

About me:

- Kristian Urup Olesen Larsen, kuol@econ.ku.dk
- BA in econ; now MA+Ph.D. student at CEBI (*not* SODAS)
- Self taught python, picked up some data science in TSDS.

About you: Say a short sentence about yourself

- Your name and educational background.
- Why you chose this course.
- Your experience with python (and generally data science).

About this course:

- Exercise classes are primarily going to be learning-by-doing.
- We will focus on the intuition behind technical aspects of the curriculum.

Todays quick warmup

I will try to bring a *quick warmup* for every session. These are pure python exercises that I hope will help develop your coding skills

- feel free to skip them if you think they are easy.
- Generally *irrelevant* to this course, but *relevant* as part of a data-scientists basic toolkit.

Q: The factorial is defined as

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1, \quad 0! = 1, \quad n \in \mathbb{Z}^{\geq}. \tag{1}$$

Write a function `factorial(n)` that *recursively* computes $n!$.

Todays quick warmup - solution

```
def factorial(n):
    if n == 0: return 1
    return n*factorial(n-1)
```

Imagine this as evaluating in stages:

```
1: n*factorial(n-1)
2: n*(n-1)*factorial(n-2)
3: n*(n-1)*(n-2)*factorial(n-3)
 ⋮
n: n*(n-1)*(n-2)*...*factorial(0)
```

Note: python is not build for recursion, often recursive solutions perform worse than iterative ones. Also

```
import sys; sys.getrecursionlimit()
```

Last lecture in a nutshell; bias, variance and regularization

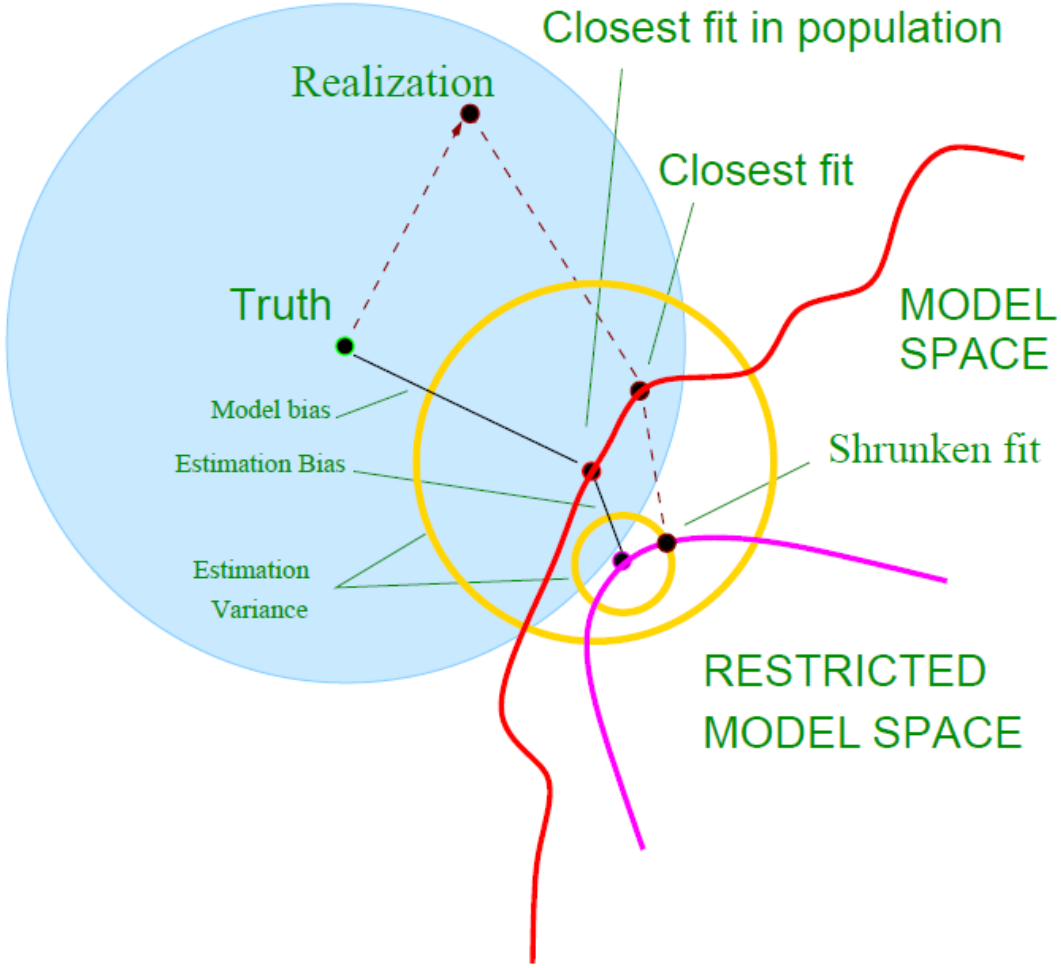

Figure: (Elements of Statistical Learning II, p. 225)

Last lecture in a nutshell; cross validation

Two fundamental problems with building models

1. *Model selection* - need unbiased estimate of generalization error to know how good our model is.
2. *Model assessment* - to report how good your model is; test it on truly independent data.

- Think of CV as an estimator for (1).
- We need one good error-estimate per model (e.g. per hyperparameter value) so we need n CV loops to compare n models.

And still, train data should be completely left alone in order to do (2) after model selection!

CV pitfalls

As a consultant for a worldwide logistics provider your database continuously receive new GPS data for their fleet. You have developed a neural network that needs to be retrained every day. Your coworker has written the following code to do so. What are your thoughts?

```
def update_nnet(new_data, nnet, db = MainDatabase):
    db.add_data(new_data)
    old_weights = nnet.weights
    X_tr, y_tr, X_te, y_te = make_split(db.data)
    w_update = update_weights_CV(nnet, X_tr, y_tr)
    nnet.weights = old_weights + w_update
    return nnet
```

CV pitfalls

As a consultant for a worldwide logistics provider your database continuously receive new GPS data for their fleet. You have developed a neural network that needs to be retrained every day. Your coworker has written the following code to do so. What are your thoughts?

```
def update_nnet(new_data, nnet, db = MainDatabase):
    db.add_data(new_data)
    old_weights = nnet.weights
    X_tr, y_tr, X_te, y_te = make_split(db.data)
    w_update = update_weights_CV(nnet, X_tr, y_tr)
    nnet.weights = old_weights + w_update
    return nnet
```

- Data are reshuffled, but the weights are not reset to random! Over time test points pollute the training data! What was test yesterday becomes train today.
- Correct way: split *new incoming data* in test and train first!