How to do Experiments

Ruwang Jiao

ruwang.jiao@ecs.vuw.ac.nz

Outline

- Experiment design
- Parameter Settings
- Results and Codes
- Statistical Significance Test

Experiment design

❖ Compared methods

- Classical and representative
- State-of-the-art
    - Published in past three years
    - Published in your target journal or top journals

❖ Dataset

- Training and test sets
    - Which method to create training and test sets
    - Splitting seed
    - Stratified splitting to maintain class-ratio

❖ Parameter settings

Training Set vs Test Set

- **Training set**: to learn/train a model
- **Test set**: to measure the "future" performance of the model

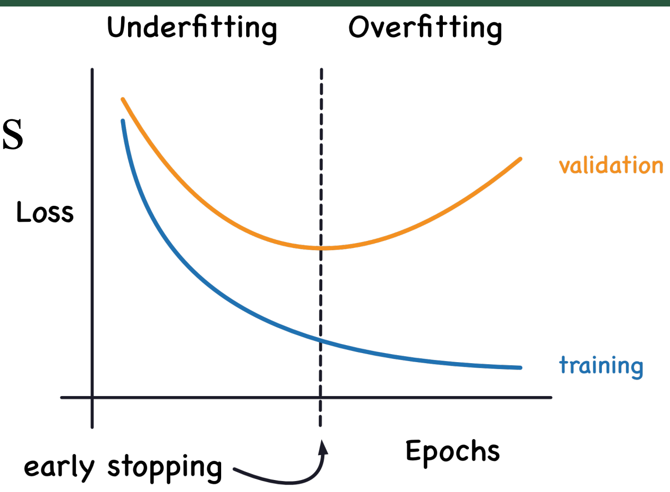
- Training—Test: 50%—50%; 2/3 — 1/3; 70%—30%
- Represent the original data
- Tradeoff: Generalisation vs *overfitting*

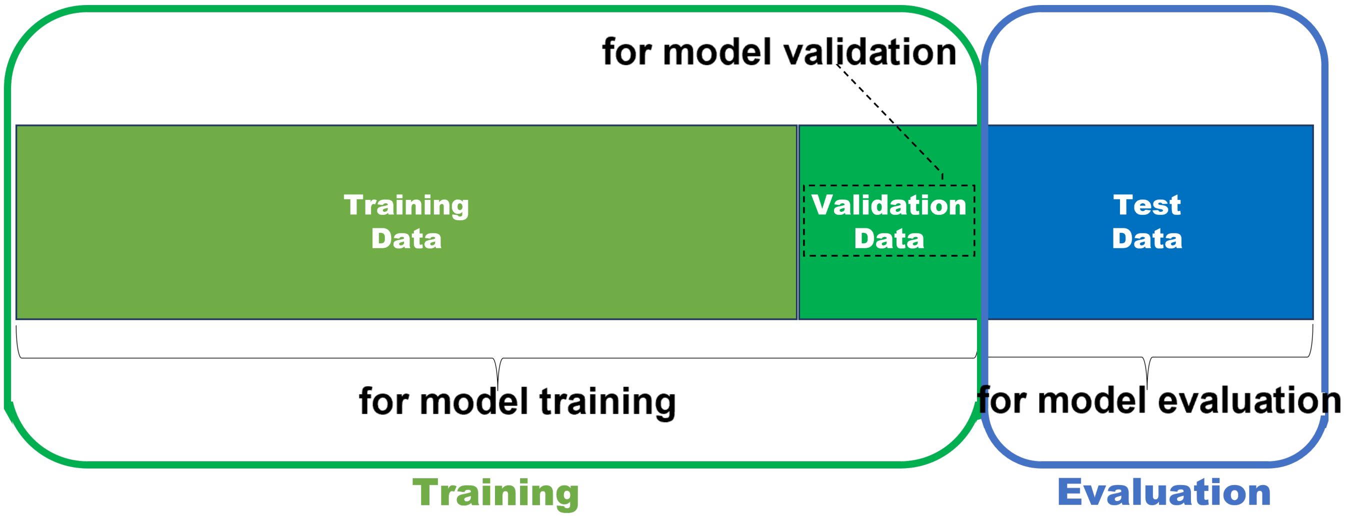
Remember that the test data remains unavailable during the training process.

Validation Set vs Test Set

❖ **Validation set**: monitor the training process

- Hyperparameter tuning
- Monitor overfitting



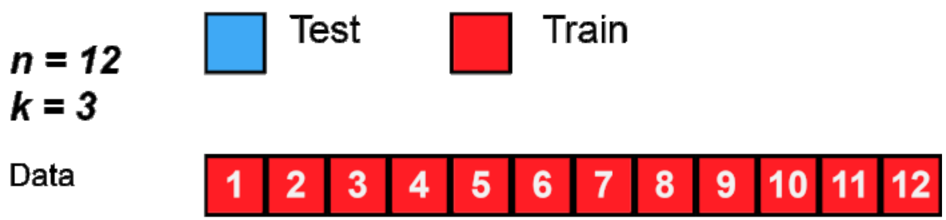


The performance of the model on the validation set cannot be regarded as training performance.

Cross Validation (CV)

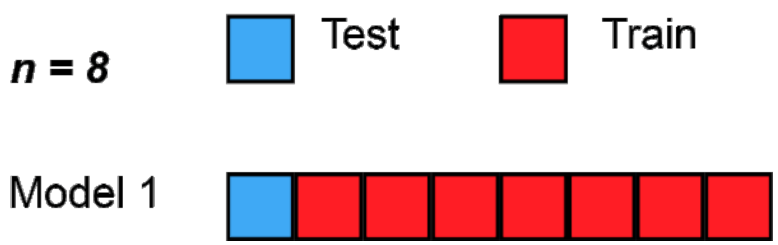
❖ *K*-fold CV

- Split the data to *K* folds with equal size
- Use 1 fold as test subset, and the other *K*-1 folds as training subset
- Repeat *K* times to make sure each fold has a chance to be the test set
- Average the *K* test performances (e.g. error rates)



❖ Leave-one-out CV

- Train learning model *n* times where *n* is the number of training instances
- Each time, only one instance is used as a test set while the rest are training set
- Average the *n* test performances (e.g. error rates)



Parameter Settings

- Please start with commonly used parameter settings from the literature, or settings recommended by good papers.
  - Do not randomly pick up some parameters values unless you have good reasons to use them.
- Figure out what the parameter values exactly mean
- Parameter tuning: one aspect at a time

Performance evaluation

❖ Number of runs:

- at least **30** runs (using 30 different seeds) => WHY?

❖ Performance evaluation:

- Measure: error rate, accuracy, training time, HV and IGD (EMO).

Mean Squared Error (regression)

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

Error Rate (classification)

$$\mathrm{ER} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(y_i \neq \hat{y}_i).$$

❖ Training Error

- The MSE/ER computed from the training data that was used to learn the model.
- We generally don't care too much about training error (it's easy to construct a model with zero training error!)

❖ Testing Error

- The MSE/ER computed from test data that was not used to learn the model.

Random Seeds For EC methods

- DO record your random seeds to make sure you can re-produce the same results later if needed
  - Do NOT use clock time as the random seed

- Use the same random seeds to compare two different versions of the same approach:
  - E.g., two different GP algorithms: GP1 and GP2, run both of them for 50 times. Please make sure you use the same 50 random seeds for GP1 and GP2 to let them have the same starting points for fair comparisons.
  - It is not necessary to use the same random seeds if you compare GP with PSO.

Results

- Please record all useful results
  - E.g., the *gbest* in PSO, the best program from GP, the training, testing performances in each run (you may further check with your supervisors)
  - Computational (training) time of each run: first generation to the last generation — not include test process
- Keep the results of the standard algorithm
- Do keep all the original results from each run.
- Do NOT delete results unless they use too much memory, or they are wrong
- Perform statistical significance tests: T-test; Wilcoxon test; Friedman test

Codes and Programs

- ❖ Backup different versions of your codes using
  - Version control: Gitlab, bitbucket
- ❖ Make clear documentation of your codes
- ❖ Make clear README documentation
- ❖ Organise your files: a single directory for a project, containing all of the data, code, and results for your project
- ❖ Following best programming practices: choose file/method/variable name carefully, avoid hard coding, …

Why Statistical Significance Test

- Suppose we have developed an EC algorithm **A**
- We want to compare with another EC algorithm **B**
- Both algorithms are stochastic
- How can we be sure that **A** is better than **B**?
- Assume we run A and B once, and get the results x and y, respectively.
- If $x < y$ (minimisation), is it because **A is better than B**, or just **randomness**?

Statistical Significance Test (SST)

- **SST** is a way to **evaluate the evidence** the data provides against a null hypothesis- H_0

  H_0 : there is no difference between **A** and **B**

- **SST** helps **quantify** whether a finding is due to chance or some factor

- **SST** study the **probability distribution** of stochastic algorithms' performance metrics

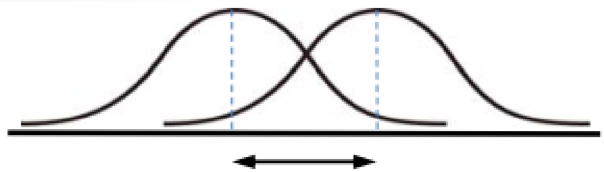
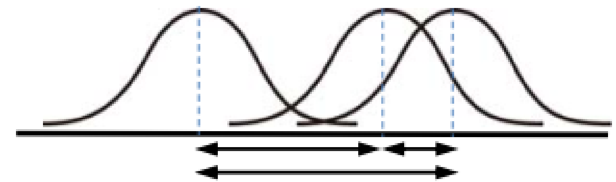
How SST Works

- Calculate a **test statistic** – describes the relationship differs from the null hypothesis H_0 (no difference)

- Calculate a probability value (***p*-value**) – estimates the probability of how likely any observed difference is due to chance
  - A **smaller *p*-value** means stronger evidence

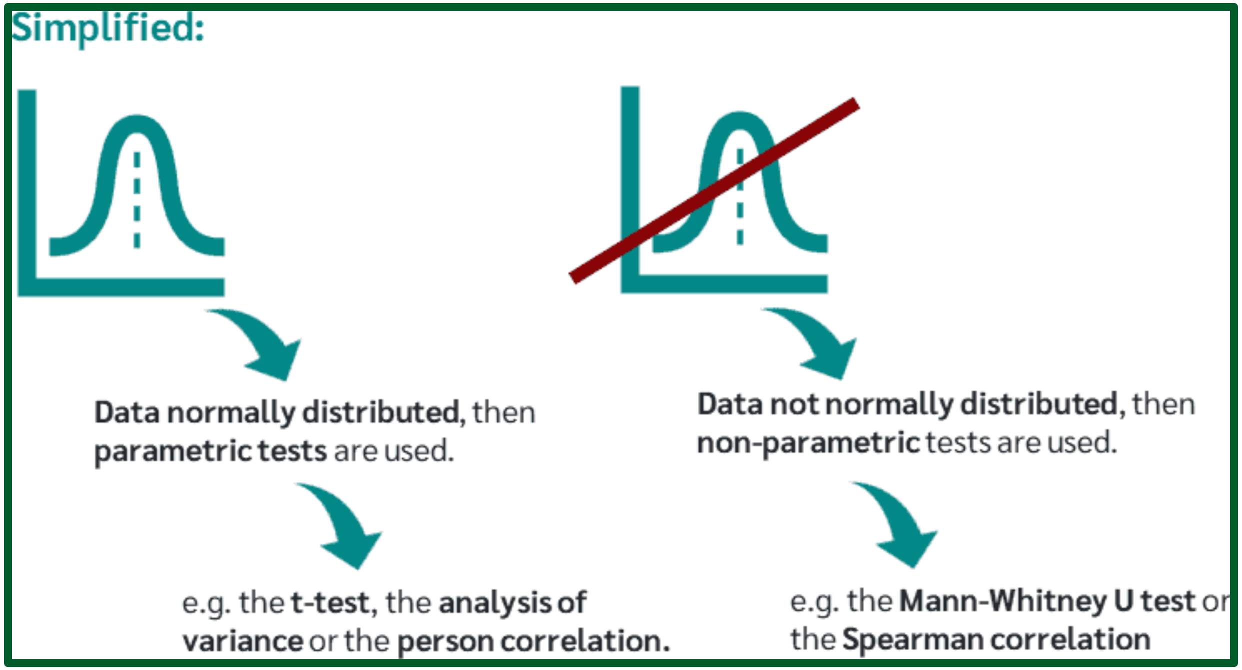
- Significance level $\boldsymbol{\alpha}$: ***p*-value** $\leq \boldsymbol{\alpha}$ means **significant.**
  - Often $\boldsymbol{\alpha = 0.05}$

Types of SST

		2 groups	*n* groups (*n* > 2)	
data distribution				
Parametric Test (normality)	unpaired (independent)	・unpaired *t*-test	ANOVA (Analysis of Variance)	・one-way ANOVA
	paired (related)	・paired *t*-test		・two-way ANOVA
Non-parametric Test (no normality)	unpaired (independent)	・Mann-Whitney *U*-test	one-way data	・Kruskal-Wallis test
	paired (related)	・sign test ・Wilcoxon signed-ranks test	two-way data	・Friedman test

Parametric vs Non-parametric Tests

- **Parametric/Non-parametric**: assume/do not assume the variables follow certain distribution e.g., normal distribution
- Parametric tests have **stricter** requirements.



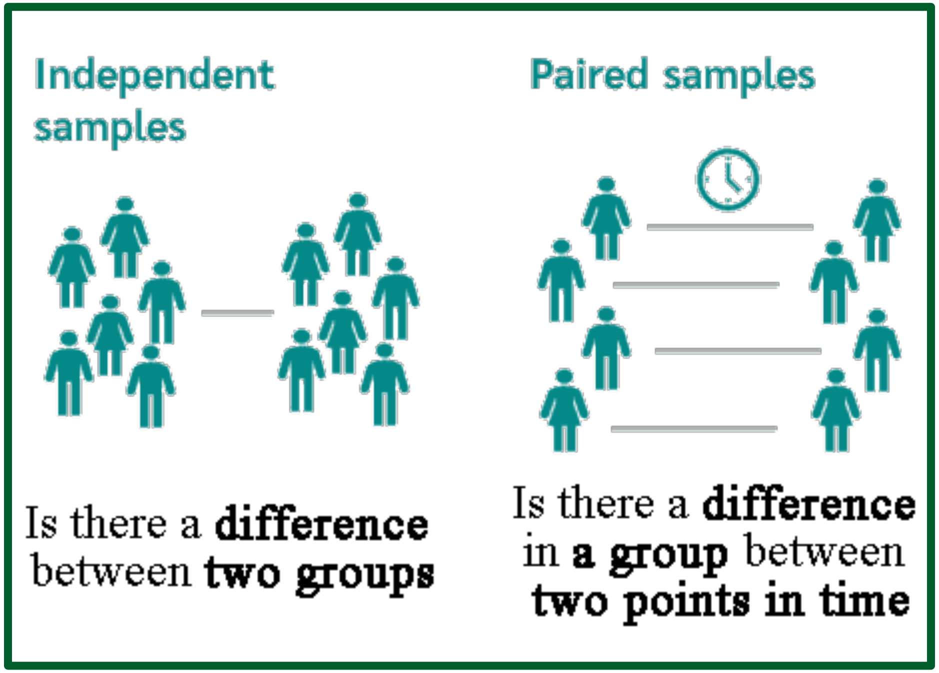
Paired vs Unpaired Tests

- **Paired**: data samples are **dependent** (one subject at 2 different times/scenarios)

  Unpaired: data samples are **independent** (two subjects)

- **Paired** tests can give us **stronger** conclusions

Example: for the compared algorithms, use **the same random seed** to generate the same initial population – use paired test



Any discussions/questions?

ruwang.jiao@ecs.vuw.ac.nz