Methods and metrics: Data organization

Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding

Train/Dev/Test

- Common in large publicly available datasets.
- Presupposes a fairly large dataset.
- We're all on the honor system to do test-set runs only when development is complete.
- The test part ensures consistent evaluations, but encourages hill climbing.

No fixed splits

- Small public datasets might not have predefined splits.
- A challenge for assessment: for robust comparisons, you really have to run all models using your assessment regime on your splits.
- For large datasets, you can impose splits and use them for the entire project:
  - Simplifies your experimental set-up.
  - Reduces hyperparameter optimization.
- For small datasets, imposing a split might leave too little data, leading to highly variable performance.

Cross-validation

In cross-validation, we take a set of examples and partition them into two or more train/test splits, and then we average over the results in some way.

Random splits

Method

For k times:

1. Shuffle.
2. Split: t percent train, usually $1 - t$ test.
3. Conduct an evaluation.

In general (but not always), we want these splits to be *stratified* in the sense that the train and test splits have approximately the same distribution over the classes.

Trade-offs

- **Good**: you can create as many as you want without having this impact the ratio of training to testing examples.
- **Bad**: no guarantee that every example will be used the same number of times for training and testing.

```
from sklearn.model_selection import ShuffleSplit,
StratifiedShuffleSplit, train_test_split
```

K-folds

K-folds

Method

Splits
fold 1
fold 2
fold 3

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

Trade-offs

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

Trade-offs

- **Good**: every example appears in a train set exactly $k-1$ times and in a test set exactly once.

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

Trade-offs

- **Good**: every example appears in a train set exactly $k-1$ times and in a test set exactly once.
- **Bad**: the size of k determines the size train/test:

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

Trade-offs

- **Good**: every example appears in a train set exactly $k-1$ times and in a test set exactly once.
- **Bad**: the size of k determines the size train/test:
  - 3-fold: train 67%, test 33%.
  - 10-fold: train 90%, test 10%.

K-folds

Method

Splits
fold 1 fold 2 fold 3

Experiment 1	
Test	fold 1
Train	fold 2 fold 3

Experiment 2	
Test	fold 2
Train	fold 1 fold 3

Experiment 3	
Test	fold 3
Train	fold 1 fold 2

Trade-offs

- **Good**: every example appears in a train set exactly $k-1$ times and in a test set exactly once.
- **Bad**: the size of k determines the size train/test:
  - 3-fold: train 67%, test 33%.
  - 10-fold: train 90%, test 10%.

```
from sklearn.model_selection import KFold,
StratifiedKFold, LeaveOneOut, cross_val_score
```