Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 11

Slides adapted from Chenhao Tan, Jordan Boyd-Graber, Chris Ketelsen

Logistics

- HW1 solutions will be out soon
- HW2 available on Github

Outline

Train-val-test

K-fold cross validation

Precision, recall, F1

Introduction

We've seen several machine learning models now

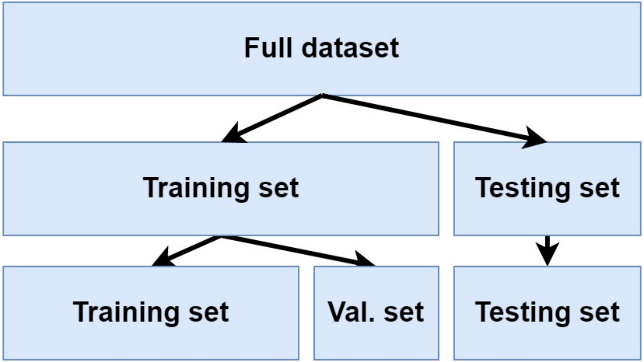
One common phrase so far is "This is another hyperparameter"

- K in K-nearest neighbors
- number of epochs in perceptron

We've talked about the importance of evaluating a learning model on unseen validation data
Next:

- Validation
- Evaluation metrics

Train-val-test



Typical ratio:

- 70%/10%/20%
- 80%/10%/10%

Outline

Train-val-test

K-fold cross validation

Precision, recall, F1

K-fold cross validation

When the number of training instances is small, it seems wasteful to have a separate validation set. What can we do? A. **cross validation**

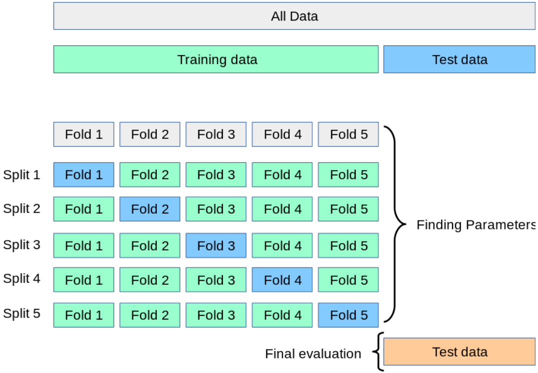


Image from `https://scikit-learn.org/stable/modules/cross_validation.html`

K-fold Cross Validation

Example use:
5-fold CV: Randomly split $N = 25$ examples into five folds $F_i, i = 1, 2, 3, 4, 5$,

train on	test on	error rate
F_1, F_2, F_3, F_4	F_5	1/5
F_1, F_2, F_3, F_5	F_4	0/5
F_1, F_2, F_4, F_5	F_3	0/5
F_1, F_3, F_4, F_5	F_2	2/5
F_2, F_3, F_4, F_5	F_1	0/5

Average error rate: $\frac{1}{5}\sum_{i=1}^{5} \mathrm{Err}_{F_i} = 12\%$

K-fold Cross Validation

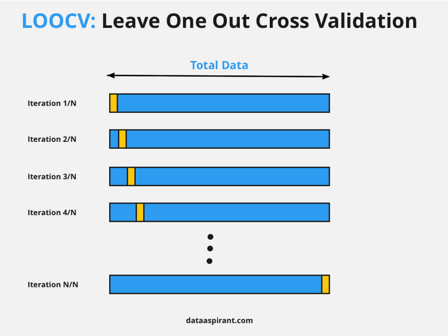
Example use:
5-fold CV: Randomly split $N = 25$ examples into five folds $F_i, i = 1, 2, 3, 4, 5$,

train on	test on	error rate
F_1, F_2, F_3, F_4	F_5	1/5
F_1, F_2, F_3, F_5	F_4	0/5
F_1, F_2, F_4, F_5	F_3	0/5
F_1, F_3, F_4, F_5	F_2	2/5
F_2, F_3, F_4, F_5	F_1	0/5

Average error rate: $\frac{1}{5}\sum_{i=1}^{5} \mathrm{Err}_{F_i} = 12\%$
Repeat this process for different hyperparameters and find the hyperparameter with the lowest error rate.

Leave-one out cross validation (LOOCV)

A special case where $k = N$



LOOCV error rate

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i^{h_{-i}}),$$

where h_{-i} represents the model trained using all the instances other than i.

Image from

https://dataaspirant.com/7-loocv-leave-one-out-cross-validation/

Nested cross validation

Purpose: You want to select best models AND compare the performance estimation of the models

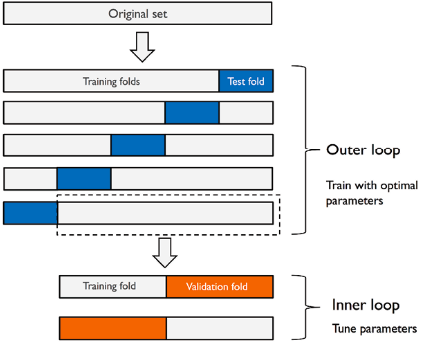


Image from `https://vitalflux.com/python-nested-cross-validation-algorithm-selection/`

K-fold cross validation

K-fold cross validation can be used for

- selecting best models from training data
- nested cross-validation for performance estimation

Outline

Train-val-test

K-fold cross validation

Precision, recall, F1

Accuracy

Thus far, for classification problems we've been primarily concerned with the misclassification error rate and the standard definition of accuracy:

$$\text{error rate} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i \neq y_i)$$

$$\text{accuracy} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i = y_i) = 1 - \text{error rate}$$

And for many classification tasks, this makes perfect sense. In fact, many classification techniques are designed specifically to minimize this error rate.

But the misclassification error rate can be misleading in many scenarios.

Accuracy

Consider the case when your test set is heavily skewed towards a particular class. If 98% of test data is from the negative class, should you feel good about a model with a 98.5% classification accuracy?

Accuracy

Consider the case when your test set is heavily skewed towards a particular class. If 98% of test data is from the negative class, should you feel good about a model with a 98.5% classification accuracy?

What about when there are different consequences for false positives vs. false negatives?

Can you think of specific examples of this case?

Precision

Confusion matrix:

		predicted labels	
		positive (1)	negative (0)
true labels	positive (1)	true positive (TP)	false negative (FN)
	negative (0)	false positive (FP)	true negative (TN)

Precision

Confusion matrix:

$$
\begin{array}{cc|c|c|}
 & & \multicolumn{2}{c}{\text{predicted labels}} \\
 & & 1 & 0 \\
\cline{3-4}
\text{true labels} & 1 & TP & FN \\
\cline{3-4}
 & 0 & FP & TN \\
\cline{3-4}
\end{array}
$$

Precision measures how accurate the predicted positive class are (exactness).

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall

		predicted labels 1	predicted labels 0
true labels	1	TP	FN
	0	FP	TN

Recall measures the fraction of positives that are correctly identified (completeness).

Recall

		predicted labels	
		1	0
true labels	1	TP	FN
	0	FP	TN

Recall measures the fraction of positives that are correctly identified (completeness).

$$\text{recall} = \frac{TP}{TP + FN}$$

F1

F1 score strikes a balance between precision and recall.

$$F1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F1 score of the minority class is usually used when evaluating classifiers on imbalanced datasets.
$F1$ is a special case of $F_\beta = (1 + \beta^2)\frac{\text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}}$.

Example

		predicted labels 1	predicted labels 0
true labels	1	80	20
	0	10	90

- Baseline (majority accuracy)
- Accuracy
- Precision
- Recall
- F1

Example

		predicted labels 1	predicted labels 0
true labels	1	80	20
	0	10	90

- Baseline (majority accuracy): 50%
- Accuracy: 85%
- Precision: 0.889
- Recall: 0.8
- F1: 0.842

Example

		predicted labels 1	predicted labels 0
true labels	1	10	10
	0	20	160

- Baseline (majority accuracy)
- Accuracy
- Precision
- Recall
- F1

Example

		predicted labels	
		1	0
true labels	1	10	10
	0	20	160

- Baseline (majority accuracy): 90%
- Accuracy: 85%
- Precision: 0.333
- Recall: 0.5
- F1: 0.4