CS 581 – Advanced Artificial Intelligence

Topic: Classification

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

Classification

- AI → ML → Supervised Learning → Classification
- Given a dataset $D = \{\langle X_i, y_i \rangle\}$ where
  - X_i is the input
  - y_i is the discrete-valued output
- Learn a function $f\left(X_j\right) \rightarrow y_j$
- We would like f to **generalize** to **unseen** data
  - As opposed to memorizing the given data

Classification Examples

- Email classification
- Medical diagnosis
- Face recognition
- Optical character/digit recognition
- Sentiment classification
- …

ALGORITHMS

- Decision trees
- Nearest neighbor classification
- Naïve Bayes
- Logistic regression
- Support vector machines
- Neural networks
- …

Generalization

- The purpose of *f*
  - Is not to memorize the "seen" data
  - Is to generalize to "unseen" data
- We need a performance metric
- We need to test *f*'s performance on a dataset that it has not seen

Types of Errors – Classification

- Pick which cases should be called "positive"
  - Spam, HasHeartDisease, etc.
- *False positive*
  - Falsely classifying an object as positive
    - E.g., classifying a legitimate email as spam, diagnosing a healthy patient as having heart disease, etc.
  - Also called *Type I* error
- *False negative*
  - Falsely classifying an object as negative
    - E.g., classifying a spam email as not-spam, claiming that a heart-disease patient is healthy, etc.
  - Also called *Type II* error

A Few Performance Measures

- 0/1 loss; error or accuracy
- Precision
- Recall
- F1
- Log-loss
- Fairness
- Domain-specific performance measures
- …

Confusion Matrix

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Accuracy

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$Accuracy = \frac{Num\ Correct}{Data\ Size} = \frac{TP + TN}{TP + TN + FP + FN}$$

PRECISION

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$Precision = \frac{True\ Positive}{Predicted\ Positive} = \frac{TP}{TP + FP}$$

True Positive Rate – Recall – Sensitivity

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$TPR = Recall = \frac{True\ Positive}{Actual\ Positive} = \frac{TP}{TP + FN}$$

F1

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$\frac{\frac{1}{P} + \frac{1}{R}}{2}$$

True Negative Rate – Specificity

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$TNR = Specificity = \frac{True\ Negative}{Actual\ Negative} = \frac{TN}{TN + FP}$$

False Positive Rate – Fall-out

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$FPR = FallOut = \frac{False\ Positive}{Actual\ Negative} = \frac{FP}{TN + FP}$$

False Negative Rate – Miss Rate

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$FNR = Miss\ Rate = \frac{False\ Negative}{Actual\ Positive} = \frac{FN}{TP + FN}$$

Splitting the dataset

1. Train-test splits
2. Train-validation-test splits
3. Cross-validation

TRAIN-TEST SPLIT

- Randomly split the data into two disjoint sets
- A typical approach: 2/3 for train and 1/3 for test
- Train your model on training data and evaluate it on the test data
  - Use your favorite performance metric
- Report your performance as the expected performance on unseen data
- Caveats
  - You need a large dataset for this to work
  - You cannot tune your parameters on the test data

Train-Validation-Test Split

- Split your data into three disjoint sets
  - Train, validation, test
- Train your model(s) on the training data
- Evaluate your model(s) on the validation data
- Pick the model that performs best on the validation data
- Test the model on the test data, and report its performance
- Caveat
  - You need a really big dataset for this to work

Cross-Validation

- Split your data into k disjoint sets
- Each time, one set is the test set and the rest is the training set
- See OneNote for more detailed explanation and illustration

Real Life Measures

- Not as clean as the ones we discussed
- Consider self-driving cars, medical diagnosis, crime prediction, fraud detection, and so on
- Often, there is not a single performance measure
- Performance is handled on a case-by-case basis; not on an aggregate level