Weka and Machine Learning

Abram Hindle

2019-02-12 Fri

Outline

Weka

Machine Learning Classifiers

Learners

Evaluation/Validation

Let's Experiment!

Weka Stuff

Licensing

Intro

- AI that learns from data
  - Learn what spam looks likes to filter it out
- Classify data into types
  - Learning spam
- Cluster data by similarity
  - Finding messages that are similar to spam
- Find important and distinct properties of the data.
  - Viagra is a spam keyword!

Kinds of ML

- Supervised
  - we give it classified examples and hope it can classify more
- Unsupervised
  - labels unknown, let the algorithm find them
- Semi Supervised learning
  - labelled and unlablled.
- Representation Learning
  - How to represent data
- Reinforcement Learning
  - policies to reward the learner

WEKA

- ML Toolkit
- Good for initial exploration if something will work
- Can use it as a library later
    - Java
- Can use it from the commandline
- Nice way to debug classic ML learners
- Major Weakness: Tuning

Some Kinds of Learners for Classification

- Tree Based
  - C4.5 (J48)
  - Random Forest
  - Decision Tree
- Rule Learners
  - Ripper (jRip)
- Support Vector Machines
  - SVM/LibSVM
- Bayesian Nets

Weka Makes some disinction

- Bayes
  - probabilities based on priors
- Functions
  - learn functions from data
- Lazy
  - when evaluating just go over the data again
- Meta
  - Combine classifiers together
- Misc
  - Whatever
- Rules
  - Learn boolean rules
- Trees
  - learn decision trees

Learners operate on different classes and values

- Some learners are boolean (True/False or 0/1)
- Some learners are nominal (class) (A/B/C/..)
- Some learners learn counts (1,2,3,..)
- Some learners learn real functions ($Y = b + ax$)

ZeroR Learner

- The smartest monkey
- Always chooses the class with the largest number of entities
- Good as a base line.
- You have to beat ZeroR.

C4.5/J48

- Produces a decision tree
- The model is code and interpretable
- Sometimes trees are too big.
- each branch is a conditional
- each leaf is a class

JRip/Ripper

- learns and prunes a small set of rules
- copy & paste into code

Naive Bayes

- Asks the question what is the probability of this value belonging to this class?
- multiplies all of these probabilities together

Logistic Regression

- We've already discussed this
- Regression used for true false

K-NN

- nearest neighbor
- use euclidean distance to find the

SVM

- support vector machine
- increase the dimensionality of your data to find ways to segment it in higher dimensional space
- tunable. Works well.
-

Matrix of classification

- True Positives (TP) - An action or label is properly applied
  - A classifier for buggy code says buggy code is buggy
- True negative (TN) - An action or label is properly not applied
  - A classifier for buggy code says NOT buggy code is NOT buggy
- False positive (FP) - An action or label is improperly applied
  - A classifier for buggy code says NOT buggy code IS buggy
- False negative (FN) - An action or label is improperly NOT applied
  - A classifier for buggy code says buggy code IS NOT buggy

Accuracy

- Given X things how often is out automated tool right?
- E.g. given 100 samples of not working source code
  - how good is our tool at fixing the source code?
- Answer: correct / total
- TP / (TP+TN+FN+FP)
- TP / Everything
- Bad in situations where 90% of the dataset is positive
  - you just guess positive and you get 90%!
- If 90% of your data is 1 class you want better than 90% accuracy
- How many classifications were correct?
- Bad for class imbalance

Kappa

- Cohen's Kappa
- like correlation
- agreement between classifier and actual data
- Very good for class imbalance
- Check it out on Wikipedia `https://en.wikipedia.org/wiki/Cohen's_kappa`

Precision

- How many of your classifications are right
- Of what was evaluated or returned what are relevant?
  - e.g. of the buggy code snippets returned how many are actually buggy?
- When I give you a positive, how right am I?
- TP / (TP + FP)
- Ignores the fact that I missed lots of buggy code.

Recall

- How much of the class did you find
- Might depend on the class
- You can have high precision for a class and have low recall
- Of what was evaluated or returned did I at least return most of what was relevant?
  - e.g. of the buggy code snippets returned did I return MOST of them
- Can only use when you know the population size
- When I return results do I return most of relevant results?
- TP / (TP + FN)

F-1 Measure

- Combination of Precision and Recall
- Geometric mean
- Can tune to one or the other
- Can I take precision and recall and balance them?
- F1 = 2 * Precision * Recall / (Precision + Recall)
    - geometric mean of precision and recall

TP/FP Rate

- True Positives
- True Negatives
- Actual accuracy for all classes

ROC Area

- Area under the Receiver Operating Characteristic Curve
- We plot True Positive versus True Negative
- sensitivity (TPR) versus specificity (TNR)
- AUC ROC 0.5 - garbage
- AUC ROC 0.7 - good

More resources

- The wikipedia page is actually great
    - https://en.wikipedia.org/wiki/Precision_and_recall
    - https://en.wikipedia.org/wiki/Cohen’s_kappa

Coffee Stain

- Find the coffee stain!
- 2 classes
- load ./data/coffee-ring/coffee-ring.arff
- Try ZeroR
- Try 1BK
- Try SVM

Captcha

- Multiple classes (26 characters!)
- load ./data/captcha/char3.arff
- Try ZeroR
- Try NaiveBayes
- Try SVM

Is a document reliability relevant?

- load ./data/general/pgsqla$_{\text{reliabilitysmallerdataset.arff}}$
- Word based
- Can you predict if it is reliability related?
- too many features!

Dupe Bugs in Open Office

- load ./data/dupe-bugs/off$_{\text{swe}\_\_}$-$_{\text{-.arff}}$
- Comparisons, can you tell which comparison will be a dupe bug or not?

How to handle text :((1/2)

- load data/triage/data/angular.js/largewo.arff
- None of this is useful!
- delete id
- Filter the owner to nominal
  - click owner
  - Weka -> Filters -> Unsupervised -> Attribute -> StringToNomial
    - click the arguments and change the index to 1
    - click apply
- . . .

How to handle text :((2/2)

- Filter content to words!
  - click content
  - click filter
    - Weka -> Filters -> Unsupervised -> Attribute -> StringToWordVector
    - click arguments
    - click attributeindices
    - change to 2 or last
    - click OK
  - click apply
- Go classify and try naivebayes

ARFF Files

- Class should be the last element of the data
- Like CSV but with a type header
- String, Bool, Char, Class, Int, Float types
  - note different types for different types of jobs