Text classification III

CE-324: Modern Information Retrieval

Sharif University of Technology

M. Soleymani

Spring 2020

Some slides have been adapted from: Profs. Manning, Nayak & Raghavan (CS-276, Stanford)

Classification Methods

- Naive Bayes (simple, common)
- k-Nearest Neighbors (simple, powerful)
- Support-vector machines (newer, generally more powerful)
- Decision trees → random forests → gradient-boosted decision trees (e.g., xgboost)
- Neural networks
- … plus many other methods
- No free lunch: need hand-classified training data
- But data can be built up by amateurs
- Many commercial systems use a mix of methods

Linear classifiers for doc classification

- We typically encounter high-dimensional spaces in text applications.
- With increased dimensionality, the likelihood of linear separability increases rapidly
- Many of the best-known text classification algorithms are linear.
  - More powerful nonlinear learning methods are more sensitive to noise in the training data.
- Nonlinear learning methods sometimes perform better if the training set is large, but by no means in all cases.

Evaluation: Classic Reuters-21578 Data Set

- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters Text Categorization data set (**Reuters-21578)** document

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>

Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data
  - Easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set)

- Validation (or developmental) set is used for parameter tuning.

Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000

- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Evaluating classification

- Final evaluation must be done on test data that are independent of the training data
  - training and test sets are disjoint.

- Measures: Precision, recall, F1, accuracy
  - F1 allows us to trade off precision against recall (harmonic mean of P and R).

Precision P and recall R

	actually in the class	actually in the class
predicted to be in the class	tp	fp
Predicted not to be in the class	fn	tn

$$\text{Precision } P = tp/(tp + fp)$$

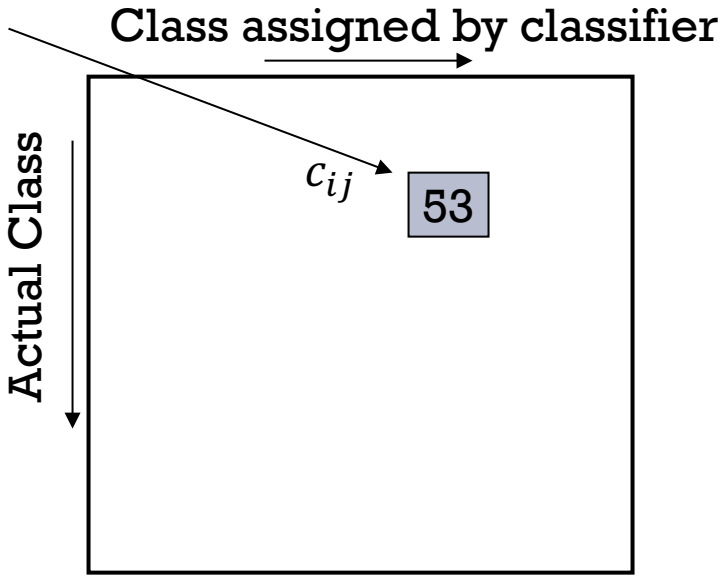
$$\text{Recall } \quad R = tp/(tp + fn)$$

$$F1 = 2PR/(P+R)$$

$$\text{Accuracy } Acc = (tp+tn)/(tp+tn+fp+fn)$$

Good practice department: Make a confusion matrix

- This (*i*, *j*) entry means 53 of the docs actually in class *i* were put in class *j* by the classifier.



- In a perfect classification, only the diagonal has non-zero entries
- Look at common confusions and how they might be addressed

Per class evaluation measures

- Recall: Fraction of docs in class *i* classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

- Precision: Fraction of docs assigned class *i* that are actually about class *i*:

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

- Accuracy: (1 - error rate) Fraction of docs classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Averaging: macro vs. micro

- We now have an evaluation measure (F1) for one class.
- But we also want a single number that shows aggregate performance over all classes

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
  - Compute F1 for each of the C classes
  - Average these C numbers
- Microaveraging: Collect decisions for all classes, aggregate them and then compute measure.
  - Compute TP, FP, FN for each of the C classes
  - Sum these C numbers (e.g., all TP to get aggregate TP)
  - Compute F1 for aggregate TP, FP, FN

Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Microaveraged score is dominated by score on common classes

Imbalanced classification

- Accuracy is not a proper criteria
- Micro-F1 for multi-class classification is equal to Accuracy
- Macro-F1 is more suitable for this purpose

(a)

	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

Evaluation measure: F1

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Hand-written rules solution
  - If (wheat or grain) and not (whole or bread) then Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- Explore methods like semi-supervised training:
  - Pretraining, transfer learning, semi-supervised learning, …
- Get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- Perfect!
- We can use all our clever classifiers
- Roll out the SVM!

- You should probably be prepared with the "hybrid" solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

A huge amount of data?

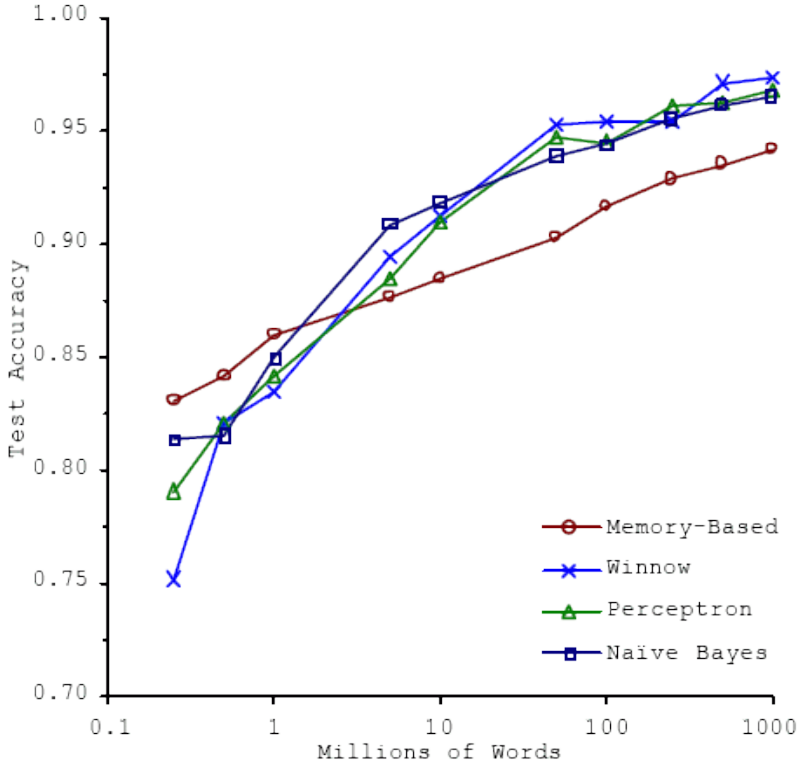
- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) and kNN (test time) are quite impractical

Amount of data?

- Little amount of data
  - stick to less powerful classifiers
- Reasonable amount of data
  - We can use all our clever classifiers
- Huge amount of data
  - Expensive methods like SVMs (train time) or kNN (test time) are quite impractical
  - With enough data the choice of classifier may not matter much, and the best choice may be unclear

Accuracy as a function of data size

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Data: Brill and Banko on context-sensitive spelling correction
- But the fact that you have to keep doubling your data to improve performance is a little unpleasant



Improving classifier performance

- Features
  - Feature engineering, feature selection, feature weighting, …
- Large and difficult category taxonomies
  - Hierarchical classification

Features: How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn't be.
  - E.g., part numbers, chemical formulas
- Sub-words and multi-words
- Does putting in "hacks" help?
  - You bet!
    - Feature design and non-linear weighting is *very* important in the performance of real-world systems

Does stemming/lowercasing/… help?

- As always, it's hard to tell, and empirical evaluation is normally the gold standard

- But note that the role of tools like stemming is rather different for TextCat vs. IR:
  - For IR, we collapse *oxygenate* and *oxygenation*, since all of those documents will be relevant to a query for *oxygenation*
  - For TextCat, with sufficient training data, stemming *does no good*.
    - It only helps in compensating for data sparseness (which can be severe in TextCat applications).
    - *Overly aggressive stemming can easily degrade performance.*

Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- Selection may make a particular classifier feasible
  - Some classifiers can't deal with 1,000,000 features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Makes runtime models smaller and faster
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting

Feature Selection: Frequency

- The simplest feature selection method:
  - Just use the commonest terms
  - No particular foundation
  - But it make sense why this works
    - They' re the words that can be well-estimated and are most often available as evidence
  - In practice, this is often 90% as good as better methods

- Smarter feature selection:
  - Mutual Infromation, chi-squared, etc.

Mutual Infromation

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t) P(C = e_c)},$$

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}$$

- N_{10} shows the number of documents that contain t but are not in c

Upweighting

- You can get a lot of value by differentially weighting contributions from different document zones

- That is, you count as two instances of a word when you see it in, say, the abstract
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al*, 2002)

Two techniques for zones

1. Have a completely separate set of features/parameters for different zones like the title
2. Use the same features (pooling/tying their parameters) across zones, but upweight the contribution of different zones

- Commonly the second method is more successful: it costs you nothing in terms of sparsifying the data, but can give a very useful performance boost
  - Which is best is a contingent fact about the data

Text Summarization as feature tweeking

- Text Summarization: Process of extracting key pieces from text, normally by features on sentences reflecting position and content

- Much of this work can be used to suggest weightings for terms in text categorization
  - See: Kolcz, Prabakarmurthi, and Kalita, CIKM 2001: Summarization as feature selection for text categorization
  - title
  - first paragraph only
  - first and last paragraphs, etc
  - paragraph with most keywords

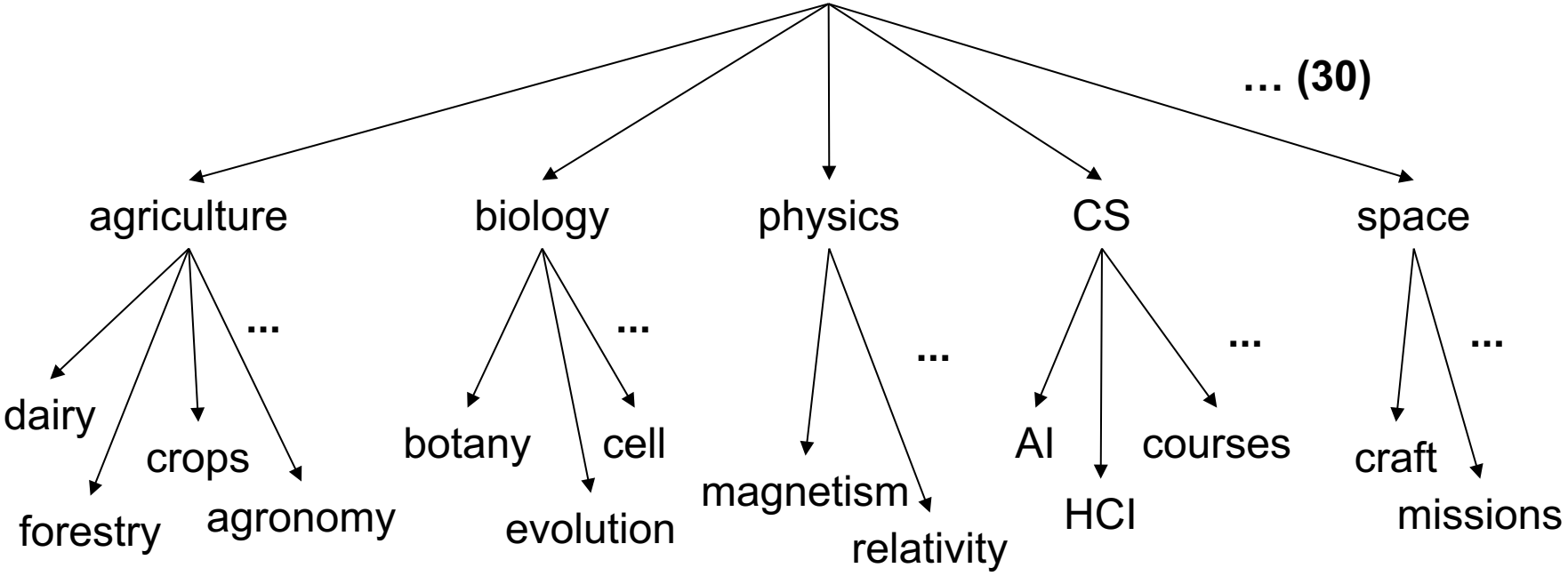
A common problem: Concept Drift

- Categories change over time
- Example: "president of the united states"
  - 1999: clinton is great feature
  - 2010: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Favors simpler models like Naïve Bayes
- Feature selection: can be bad in protecting against concept drift

How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Much literature on **hierarchical classification**
      - Mileage fairly unclear, but helps a bit (Tie-Yan Liu et al. 2005)
      - Definitely helps for scalability, even if not in accuracy
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - May need a hybrid automatic/manual solution

Yahoo! Hierarchy



www.yahoo.com/Science

Resources

- *IIR, Chapter* 13.5-13.6 and 15.3.