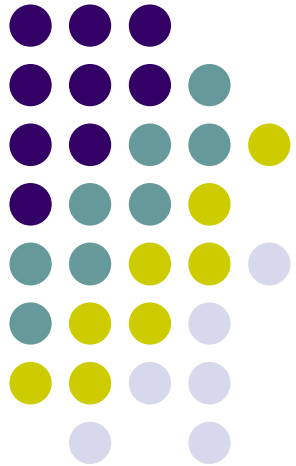
Big Data Analytics

Lecture 10 (part 1)

Supervised classification methods with human tagging:

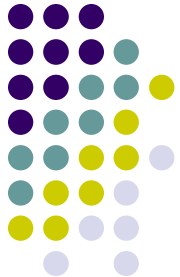
Proportional Algorithms



References

- ✓ Ceron, Andrea, Curini, Luigi and Stefano M. Iacus Justin, and Stewart, Brandon M. (2016). iSA: a fast, scalable and accurate algorithm for sentiment analysis of social media content, *Information Sciences*, 367–368 (1), 105–124

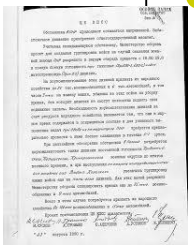
Measuring proportions

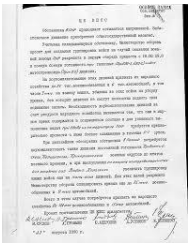


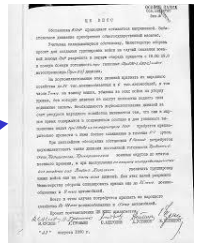
For many social science applications, only the proportion of documents in a category is needed, not the categories of each individual document

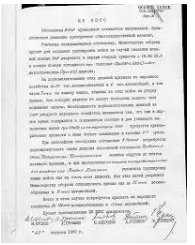
That is…

From here: individual classification



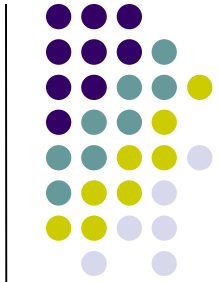






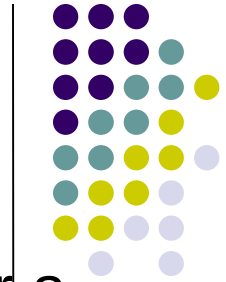
To here: proportional classification

Measuring proportions



To understand how this approach actually works, we have to introduce a change in the DfM of a corpus as we discussed up to now

Measuring proportions



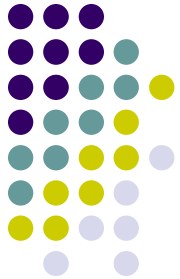
Now we include in the DfM an **indicator** (0/1) of whether a word occurred in a document, rather than the **counts** of the words (one-hot-encoding, remember?)

Post	Cat	Word: nuclear	Word: fear	Word: radiation	Word: pollution	Word: waste	Word: economic
post#1	like	1	0	0	0	0	1

Using this representation, let's define a "word profile" $p(W_1)$ for document 1 as the distribution of 0/1 within our DfM (1,0,0,0,0,1 in our example)

p(**W**) is therefore the proportion of documents in the corpus observed with each pattern of word profiles

Measuring proportions



The data-generating process for the documents can then be written as:

$$p(\boldsymbol{W}) = p(\boldsymbol{W}|\boldsymbol{C}) * p(\boldsymbol{C}),$$

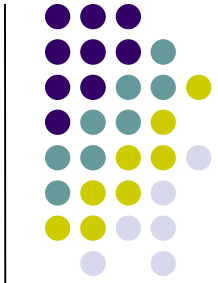
where:

$p(\boldsymbol{W})$ is the proportion of documents in the corpus observed with a given pattern of word profiles

$p(\boldsymbol{W}|\boldsymbol{C})$ is the proportion of documents in the corpus observed with a given pattern of word profiles **conditional** on categories

$p(\boldsymbol{C})$ is the proportion of documents in each class in the corpus - **the quantity of our interest**

Measuring proportions



$$p(\boldsymbol{W}) = p(\boldsymbol{W}|\boldsymbol{C}) * p(\boldsymbol{C})$$

$p(\boldsymbol{W})$ is the distribution of the features in the whole set (train + test). We have an **accurate estimation** here!

And what about $p(\boldsymbol{W}|\boldsymbol{C})$? It requires labeled documents - which are unavailable for the test set!

But if we assume that the conditional distributions **are identical in the training and in the test set**, then we can estimate $p(\boldsymbol{W}|\boldsymbol{C})$ directly from the training-set

We have therefore also here an **accurate estimation** (as long as the coders did a good job!)

Estimating $p(\boldsymbol{C})$ is now therefore easy by solving the equation via standard regression algebra!

Measuring proportions

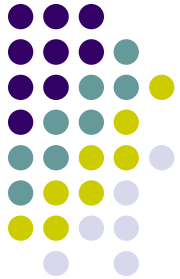
$$p(\boldsymbol{W}) = p(\boldsymbol{W}|\boldsymbol{C}) * p(\boldsymbol{C})$$

If we think of $p(\boldsymbol{C})$ as the unknown "regression coefficients" (the $\boldsymbol{\beta}$), $p(\boldsymbol{W}|\boldsymbol{C})$ as the "explanatory variables" matrix *X*, and $p(\boldsymbol{W})$ as the "dependent variable" *Y*, then this equation becomes the usual: $\mathrm{Y} = X\beta$ (with no error term)

From here, we can move to estimate $p(\boldsymbol{C})$ (via standard constrained least squares to ensure that elements of $p(\boldsymbol{C})$ are each in [0,1] and collectively sum to 1):

$$p(\boldsymbol{C}) = p(\boldsymbol{W}) * p(\boldsymbol{W}|\boldsymbol{C})^{-1}$$

Measuring proportions



In other words: instead of modeling the relation between features (i.e., words) and classes for **each single training document**, this approach uses a regression model that associates feature distribution ($p(\boldsymbol{W})$) with class distributions $p(\boldsymbol{W}|\boldsymbol{C})$ in the **entire training collection**

A key point is that this calculation does not require classifying individual documents into categories and then aggregating; it estimates the **aggregate proportions** $p(\boldsymbol{C})$ for target collections of unlabeled documents **directly**!

Measuring proportions

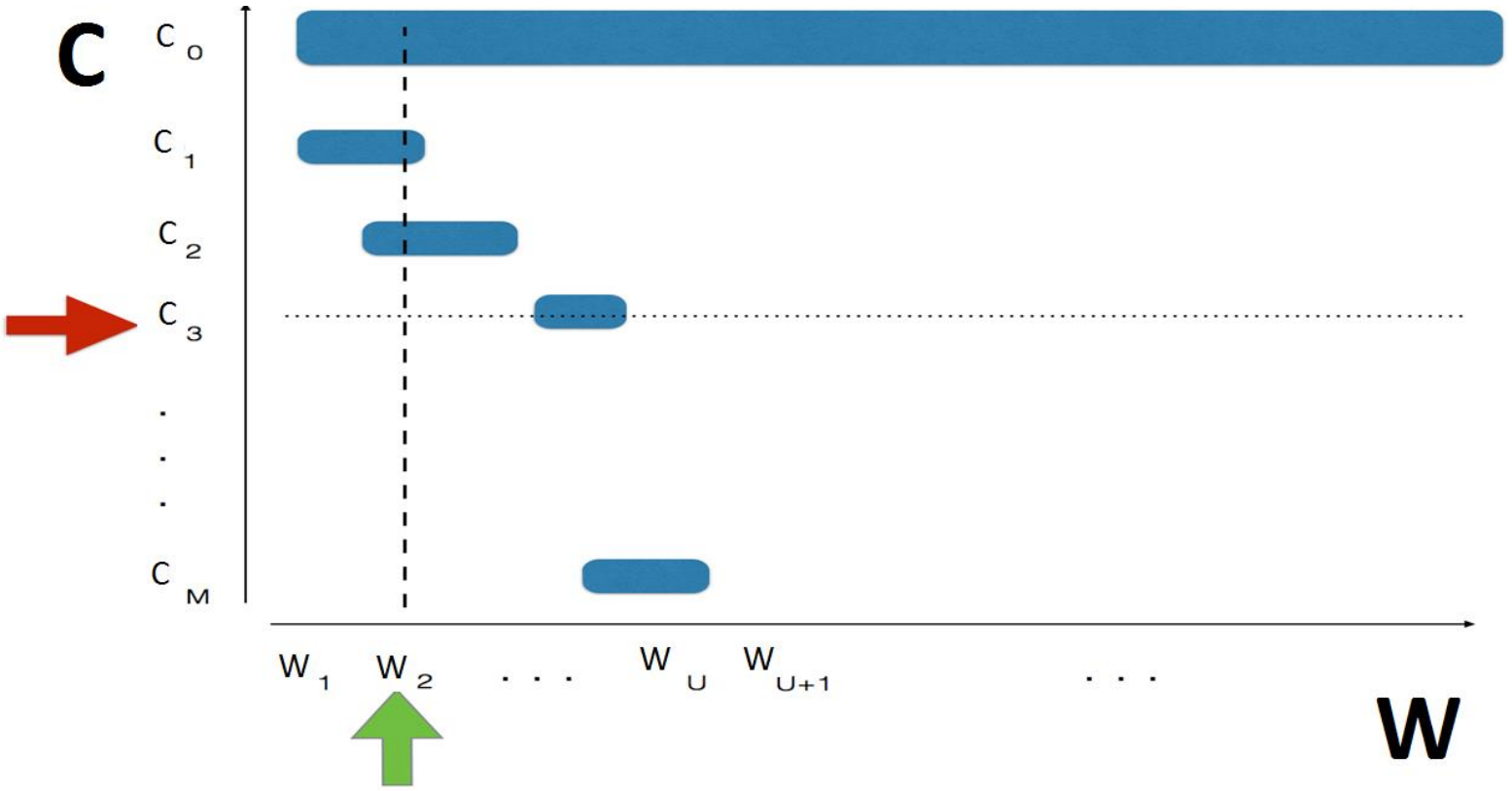
Focusing on $p(\boldsymbol{W}|\boldsymbol{C})$ rather than $p(\boldsymbol{C}|\boldsymbol{W})$ as done in the machine learning approach (remember!), has two main advantages

Theoretically: $p(\boldsymbol{W}|\boldsymbol{C})$ means: «given a post that is associated to a given content, which are the sequence of stems effectively employed to express that specific content»?

This makes a lot of sense: you do not start writing and only **afterwards** discover your sentiment toward for example a party. You start with a view, with a “category” in your mind (good, bad, support or not), and then set it out in words

Measuring proportions

Empirically: the existence of a category C_k extremely frequent in a training-set can negatively affect $p(\boldsymbol{C}|\boldsymbol{W})$, i.e., ML approaches, but not $p(\boldsymbol{W}|\boldsymbol{C})$ - fewer problems with very imbalanced training-set!

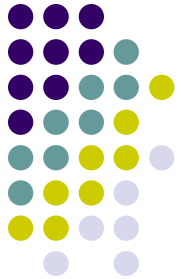


Measuring proportions

Moreover…choosing a classifier by maximizing the percent correctly classified at the individual level (as it happens with the ML algorithms discussed up to now) can sometimes drastically increase the **bias of aggregate quantities**

It is easier to look at the shape of the haystack rather than trying to find a needle in it!

Measuring proportions



Summing up: shifting focus to **estimating proportions**, can lead therefore to substantial improvements in accuracy (if you are mainly interested in such statistics!)

For example, if you are interested in the average support towards ISIS in 2015 on Twitter, you are interested in the proportion of tweets showing a positive attitude towards ISIS on a daily basis, not in classifying each single tweet as positive/negative

You want therefore to minimize the error on the former value (the %), not on the individual classification

Measuring proportions

No statistical property must be satisfied by the training set for this approach to work properly: the training set is **not a representative sample** of the distribution of opinions in the population of texts to be analyzed!

However, the language used in the training-set to express some given concept is **assumed** to be the same as in the whole population of posts, i.e. social media users use the same language

✓ Is it a **reasonable assumption**?

Measuring proportions

After all, in the **Oxford Dictionary** (English) you have **650k terms**

In reality, for any given topic, in the everyday language there is a tendency to use at the maximum between **200 and 500 *stems***

This is what **makes possible** the statistical analysis

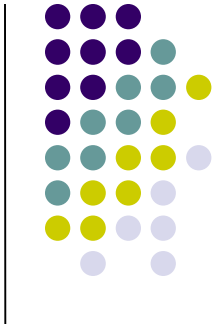
Validation when measuring proportions

How to run a cross-validation given that you do not make any individual classification in this case?

Well, you can still run a **cross-validation** procedure on your training-set (but ONLY at the aggregate level)!

For example, what you can do is estimating the **MAE** (mean average error) across categories – in this case, contrary to what discussed in the Lab class 7 (do you remember?) this is the most appropriate way for doing it!

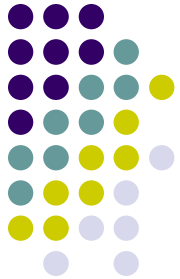
Validation when measuring proportions: estimating MAE



		Actual class		
		Cat	**Dog**	**Rabbit**
Predicted class	**Cat**	**5**	2	0
	Dog	3	**3**	2
	Rabbit	0	1	**11**

A *True # classes*	*B* *% True*	*C* *Predicted # classes*	*D* *% Predicted*	*Asbolute difference B-D*	
8	29.6%	7	25.9%	3.7%	
6	22.2%	8	29.6%	7.4%	
13	48.1%	12	44.4%	3.7%	
				4.9%	***MAE***

Measuring proportions

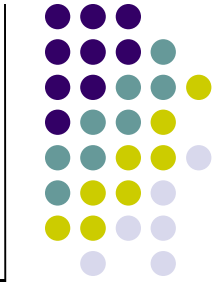


There are two main algorithms available for this type of proportional analysis:

ReadMe (Hopkins and King 2010) and **iSA** (different implementations of the same idea explained above)

We will see in the lab class iSA. If you are interested about ReadMe, just drop me an email

Measuring proportions



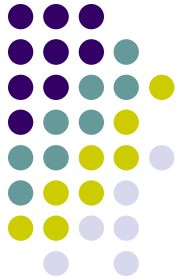
iSA: collapses the vector of stems into one-dimensional entity

More in details:

- ✓ Each vector of stems, e.g. *sj* = (0, 1, 1, 0, . . . , 0, 1) is transformed into a string-sequence that we denote by *Cj* = "0110 · · · 01"; this is the first level of dimensionality reduction of the problem: from a *N* x *K* matrix to a one-dimensional vector *N* x *1*
- ✓ This sequence of 0's and 1's can be further translated into hexadecimal notation such that the sequence '11110010' is recorded as *λ* = 'F2' or '11100101101' as *λ* = 'F2D', and so forth. So each text is represented by a label *λ* of shorter length

Implications: fast, memory saving (dimension reduction), reduced variability of the estimates, stable and scalable

R pakcages to install



install.packages("tm", repos='http://cran.us.r-project.org')

devtools::install_github("blogsvoices/iSAX")

If you have problems in installing iSAX…

For Windows ppl out there: install Java 64bit (if you have a 64bit machine!)

For Mac ppl out there: apparently isax loads Java developer kit version, instead of normal java. So install two javas in your Mac: normal java, and java developer kit