Practical Introduction to Machine Learning

Pablo Ariel Duboue, PhD

Les Laboratoires Foulab
Montreal, Quebec

RadialPoint Community Tech Talks

Outline

Outline

Before Montreal

- Columbia University
  - WSD in biology texts (GENIES)
  - Natural Language Generation in medical and intelligence domains (MAGIC, AQUAINT)
  - Thesis: "Indirect Supervised Learning of Strategic Generation Logic", defended Jan. 2005.
    - Advisor: Kathy McKeown
    - Committee: Hirschberg/Jurafsky/Rambow/Jebara
- IBM Research Watson
  - AQUAINT: Question Answering (PIQuAnT)
  - Enterprise Search - Expert Search (TREC)
  - Connections between events (GALE)
  - Deep QA - Watson

In Montreal

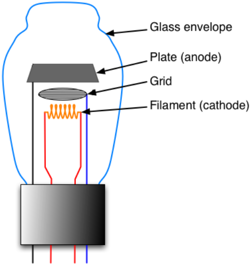
I am passionate about improving society through language technology and split my time between teaching, doing research and contributing to free software projects

- Collaboration with Prof. Nie at GRIUM
  - Hunter Gatherer project (Montreal Python next Monday)
- Taught a graduate class in NLG in Argentina
- Contributed to a number of Free Software projects
- Doing some consulting focusing on startups and small businesses
  - MatchFWD, UrbanOrca, KeaText

Outline

Why This Talk

- Levels of abstraction.
  - Vacuum tubes



- Machine learning from practitioners for practitioners
- https://github.com/DrDub/Thoughtland

What is Machine Learning?

- A new way of programming
- Magic!
- Leaving part of the behavior of your program to be specified by calculating unknown numbers from "data"
  - Two phases of execution: “training” and “application”

The ultimate TDD

- If you're using a library, you almost do no coding, just test!
- But every time you test, your data becomes more and more obsolete
  - No peeking!
- Have met people who didn't have any tests and considered
  - Bugs in the code same are the same as model issues
  - My experience has been quite the opposite, the code you write on top of machine learning algorithms has to be double and triple checked

Taxonomy of Machine Learning Approaches

- **Supervised learning**

  Monkey see, monkey do

  - Classification
- **Unsupervised learning**

  Do I look fat?

  - Clustering
- Others
  - Reinforcement learning: learning from past successes and mistakes (good for game AIs and politicians)
  - Active learning: asking what you don't know (needs less data)
  - Semi-supervised: annotated + raw data

Major Libraries

- Scikit-learn (Python)
- R packages (R)
- Weka (Java)
- Mallet (CRF, Java)
- OpenNLP MaxEnt (Java)
- Apache Mahout (Java)
- ...
- ...

Concepts

- Trying to learn a function $f(x_1, \ldots, x_n) \rightarrow y$
    - x_i are the **input** features.
    - y is the **target** class.
- The key here is *extrapolation*, that is, we want our learned function to **generalize** to unseen inputs.
    - Linear interpolation is on itself a type of supervised learning.

Data

- Collecting the data
    - Data collection hooks
    - Annotating data
        - Annotation guidelines
        - Cross and self agreement
- Representing the data (as **features**, more on this later)
- Understanding how well the system operates over the data
    - Testing on **unseen** data
- A DB is a rather poor ML algorithm
    - Make sure your system is not just memorizing the data
    - "Freedom" of the model

Evaluating

- Held out data
  - Make sure the held out is representative of the problem and the overall population of instances you want to apply the classifier
- Repeated experiments
  - Every time you run something on eval data, it changes you!
- Cross-validation
  - Training and testing on the same data but not quite
  - data = {A,B,C}
    - train in A,B, test in C
    - train in A,C, test in B
    - train in B,C, test in A

Metrics

- Measuring how many times a classifier outputs the right answer ("accuracy") is not enough
  - Many interesting problems are very biased towards a background class
  - If 95% of the time something doesn't happen, saying it'll never happen (not a very useful classifier!) will make you only 5% wrong
- Metrics:

$$precision = \frac{|correctly\ tagged|}{|tagged|} = \frac{tp}{tp + fp}$$

$$recall = \frac{|correctly\ tagged|}{|should\ be\ tagged|} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R}$$

Outline

Naive Bayes

- Count and multiply
- How spam filters work
- Very easy to implement
- Works relatively well but it can seldom solve the problem completely
  - If you add the target class as a feature, it will still has a high error rate
  - It never “trusts” anything too much

Why Naive?

- Bayes Rule

$$p(C \mid F_1, \ldots, F_n) = \frac{p(C)\, p(F_1, \ldots, F_n \mid C)}{p(F_1, \ldots, F_n)}$$

$$posterior = \frac{prior \times likelihood}{evidence}$$

- Naive Part
  - Independence assumption of the F_x, that is $p(F_i \mid C, F_j) = p(F_i \mid C)$

$$p(C|F_1, \ldots, F_n) \propto p(C)\, p(F_1 \mid C) \ldots p(F_n \mid C)$$

Decision Trees

- Find the partition of the data with higher information gain

  Value of a piece of gossip

$$IG\left(splitting\ S\ at\ A\ into\ T\right) = H\left(S\right) - \sum_{t \in T} p\left(t\right) H\left(t\right)$$

- Easy to understand
  - Both algorithm and trained models
- Can overfit badly
  - Underperforming
- Coming back with random forests

Biology: Problem

- "Disambiguating proteins, genes, and RNA in text: a machine learning approach," Hatzivassiloglou, Duboue, Rzhetsky (2001)
- The same term refers to genes, proteins and mRNA:
  - "By UV cross-linking and immunoprecipitation, we show that **SBP2** specifically *binds* selenoprotein *mRNAs* both in vitro and in vivo."
  - "The **SBP2** *clone* used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3' UTR truncated at the polyadenylation site)."
- This ambiguity is so pervasive that in many cases the author of the text inserts the word "gene", "protein" or "mRNA" to disambiguate it itself
  - That happens in only 2.65% of the cases though

Biology: Features

- Take a context around the term, use the occurrence of words before or after the term as features.
- Keep a tally of the number of times each word has appear with which target class:

term	gene	protein	mRNA
PRIORS	0.44	0.42	0.14
D-PHE-PRO-VAL-ORN-LEU		1.0	
NOVAGEN	0.46	0.46	0.08
GLCNAC-MAN	1.0		
REV-RESPONSIVE	0.5	0.5	
EPICENTRE		1.0	
GENEROUSLY	0.33	0.67	

Biology: Methods

- Instead of multiplying, operate on logs

```
float[] predict = (float[]) priors.clone();
// ... for each word in context ...
if (wordfreqs.containsKey(word)) {
  float[] logfreqs = wordfreqs.get(word);
  for (int i = 0; i < predict.length; i++)
    predict[i] += logfreqs[i];
}
```

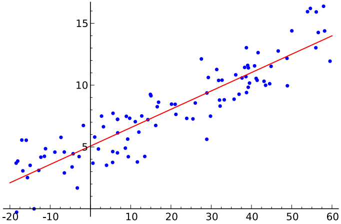
Biology: Results

- Used a number of variations on the features
  - Removed capitalization, stemming, filtered part-of-speech, added positional information
  - Changed the problem from three-way to two-way classification
- Results of Tree-learning and Naive Bayes were comparable (76% two-way and 67% three-way).
- Distilled some interesting rules from the decision trees:
  - after ENCODES is present
    before ENCODES is NOT present
    $\Longrightarrow$class gene [96.5%]

Outline

Logistic Regression

- Won't explain in detail
- It is similar to linear regression but in log space



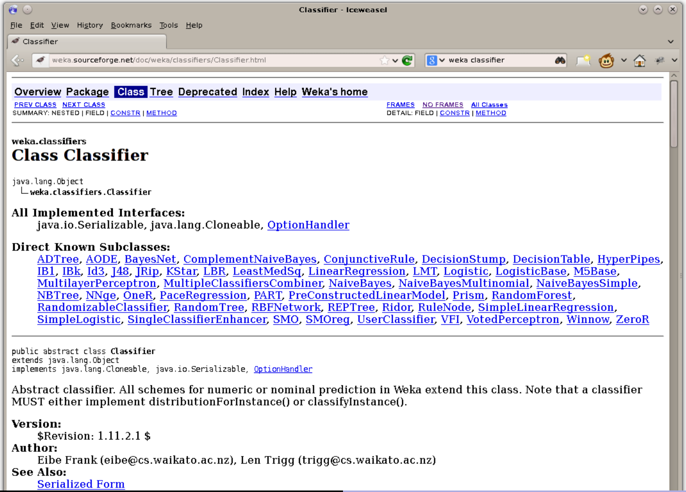
(Wikipedia)

- Can take lots of features and lots of data
- High performance
- Output is a goodness of fit

Weka

- ARFF format
  - Text file, with two sections
    @relation training_name
    @attribute attribute_name numeric *x number of features*
    @data
    7.0,1.1,... . *x number of training instances*
- Training classifiers
  `java -jar weka.jar weka.classifiers.functions.LogisticRegression -t train.arff`
  - Or programmatically:
    - Create an `Instances` class with certain attributes and create objects of type `Instance` to add to it
    - Create an empty classifier and train it on the Instances
- Using the trained classifiers
  `classifyInstance(Instance)` or `distributionForInstance(Instance)`

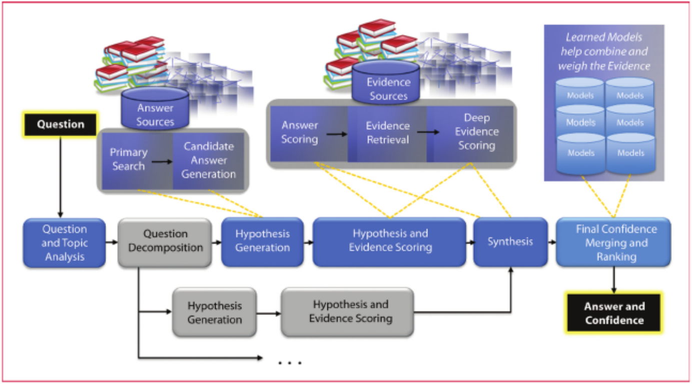
Why Weka?


Classifier - Iceweasel
File Edit View History Bookmarks Tools Help
Classifier
weka.sourceforge.net/doc/weka/classifiers/Classifier.html
weka classifier
Overview Package Class Tree Deprecated Index Help Weka's home
PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONSTR | METHOD
FRAMES NO FRAMES All Classes
DETAIL: FIELD | CONSTR | METHOD
weka.classifiers
Class Classifier
java.lang.Object
└weka.classifiers.Classifier
All Implemented Interfaces:
java.io.Serializable, java.lang.Cloneable, OptionHandler
Direct Known Subclasses:
ADTree, AODE, BayesNet, ComplementNaiveBayes, ConjunctiveRule, DecisionStump, DecisionTable, HyperPipes, IB1, IBk, Id3, J48, JRip, KStar, LBR, LeastMedSq, LinearRegression, LMT, Logistic, LogisticBase, M5Base, MultilayerPerceptron, MultipleClassifiersCombiner, NaiveBayes, NaiveBayesMultinomial, NaiveBayesSimple, NBTree, NNge, OneR, PaceRegression, PART, PreConstructedLinearModel, Prism, RandomForest, RandomizableClassifier, RandomTree, RBFNetwork, REPTree, Ridor, RuleNode, SimpleLinearRegression, SimpleLogistic, SingleClassifierEnhancer, SMO, SMOreg, UserClassifier, VFI, VotedPerceptron, Winnow, ZeroR
public abstract class Classifier
extends java.lang.Object
implements java.lang.Cloneable, java.io.Serializable, OptionHandler
Abstract classifier. All schemes for numeric or nominal prediction in Weka extend this class. Note that a classifier MUST either implement distributionForInstance() or classifyInstance().
Version:
$Revision: 1.11.2.1 $
Author:
Eibe Frank (eibe@cs.waikato.ac.nz), Len Trigg (trigg@cs.waikato.ac.nz)
See Also:
Serialized Form

Jeopardy!™: Problem

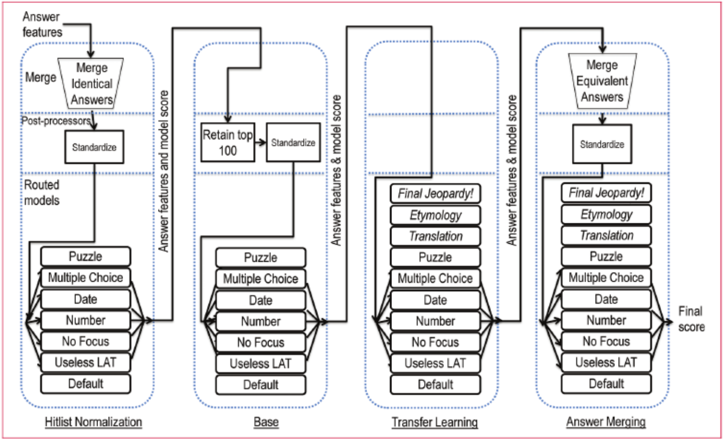
- Learning to rank
  - Rather than predicting a class, choose the best one among many instances
  - In the Jeopardy!™ case, the instances were candidate answers
- Features related to each particular answer candidate
  - "evidence"
- As logistic regression produces a goodness of fit, it can be used for ranking
  - Other classifiers might just give you 0 or 1 independent of relative goodness

Jeopardy!™: Deployment



DeepQA Architecture, from Ferrucci (2012)

Jeopardy!™: Feature Engineering



First four phases of merging and ranking, from Gondek, Lally, Kalyanpur, Murdock, Duboue, Zhang, Pan, Qiu, Welty (2012)

Outline

Maximum Entropy

- Tons and tons of (binary) features
- Very popular at beginning of 2000's
  - CRF has taken some of its glamour
  - Mature code
- OpenNLP MaxEnt uses strings to represent its input data

  previous=succeeds current=Terrence next=D. currentWordIsCapitalized

- Training with `trainModel(dataIndexer, iterations)` and using it with `double[] eval(String[] context)`

KeaText: French POS Tagger

- Work done recently at KeaText, a local company specialized on bilingual information extraction
  - Contracts, legal judgements, etc. extract key information items (who, when, where)
  - Highly specialized staff

KeaText
Transformez vos documents en outils de décision

- An existing part-of-speech tagger for the French language was a mixture of Python and Perl
  - Instead of re-engineering it, we ran it on a large corpus of French documents
  - Trained a new MaxEnt model on it
- Took less than 2 days of work and produced a Java POS tagger at about 5% the same performance as the original

KeaText: Approach

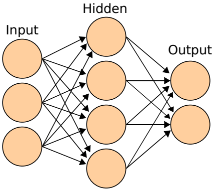
- Part-of-speech tagging is not unlike word sense disambiguation described at the beginning of the talk
- The problem is more complicated, though, as it involves more classes and every word has to be tagged
  - MaxEnt lends itself well to an approach where everything that can be tought of it is considered as a feature
- Features include
  - The word itself
  - Suffixes, up to the last four characters of the word
  - Preffixes, up to the last four characters of the word
  - Previous words, with their identified tags
  - Whether the word has special characters or if it is all numbers or uppercase

Outline

Neural Networks

- The “original” ML
- Second to best algorithm
- Slow
- Most people are familiar with it
- AI winter
- Making a come back with Deep Learning

How to Train ANNs



(Wikipedia)

- Execution: Feed-forward

$$y_q = K\left(\sum_i x_i * w_{iq}\right)$$

- Training: Backpropagation of errors
- Problem: overfitting, use a separate set as the termination criteria
-

K4B: Problem

- Given the bytecodes of a java method, come up with some terms to describe it
- Use all the Java code in the Debian archive as training data
  - Pairs bytecodes / javadoc
- Applications in Reverse Engineering
  - Java malware
- More information:
  - Training data: http://keywords4bytecodes.org
  - Source code: https://github.com/DrDub/keywords4bytecodes
- MEKA: Multi-label Extensions to Weka: http://meka.sourceforge.net/

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
```


```
.e()
.e()
()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Reverse Engineering Example

```
private final int c(int) {
    0 aload_0
```

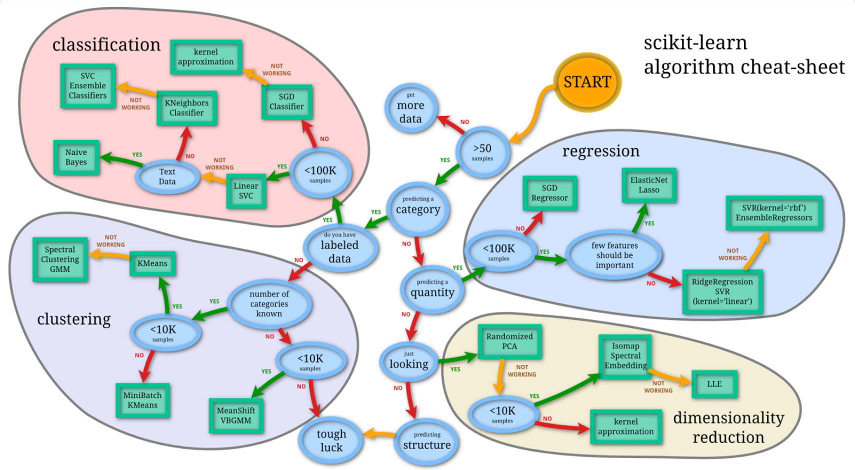
K4B: Data

- Final corpus:
  - 1M methods
  - 35M words
  - 24M JVM instructions
- Example training instance:
  - Class: `net.sf.antcontrib.property.Variable`
  - Method: `public void execute() throws org.apache.tools.ant.BuildException`
  - JavaDoc: *Execute this task.*
  - Bytecodes: (126 in total)
    - `0 aload_0`
    - `1 getfield net.sf.antcontrib.property.Variable.remove`
    - `4 ifeq 45`
    - `7 aload_0`
    - `8 getfield net.sf.antcontrib.property.Variable.name`
    - `11 ifnull 26`
    - `14 aload_0`
    - `15 getfield net.sf.antcontrib.property.Variable.name`
    - `18 ldc ""`
    - `20 invokevirtual java.lang.String.equals(java.lang.Object)`
    - `23 ifeq 36`

K4B: Results

Term	P	R	**F**
@ generated	0.76	0.80	**0.783**
replaced	0.93	0.60	**0.734**
@ param	0.64	0.74	**0.690**
icu	0.75	0.49	**0.600**
o the	0.47	0.75	**0.582**
@ stable	0.72	0.45	**0.561**
@ inheritdoc	0.42	0.60	**0.495**
@ return the	0.41	0.52	**0.463**
receiver	0.72	0.31	**0.440**

How to Choose



by Andreas Mueller
http://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html

How to Come Up with Features

1. Throw everything (and the kitchen sink) at it
2. Stop and think
    1. What information would **you** us to solve that problem?
    2. Look for published work
        - Papers: http://aclweb.org/anthology-new/
        - Blog postings
        - Open source projects
3. Add computable features
    - Learning to sum takes an incredible amount of training!

Improving the Classifier

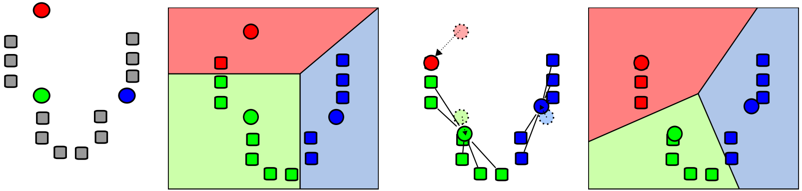
- More data
- Better features
- Solve a different problem
- Shop around for a different classifier / parametrization
  - Procedural overfitting
- Add unlabelled data
- Drop ML and program it by hand

Outline

Clustering

- A little more magical
- Having a model, fitting parameters
- Having parameters, building a model
- Knowing the answer, looking for the question
- Concept of distance between instances

k-means Clustering



(Wikipedia)

Outline

Apache Mahout

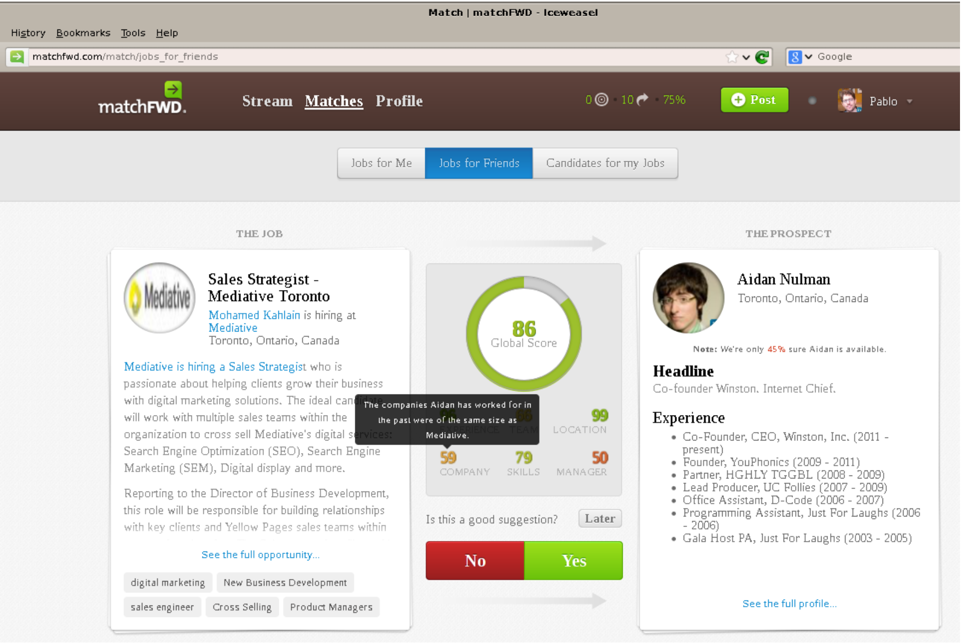
- Recommendation
- Clustering
- Classification
- Hadoop
- Input is in Hadoop sequence file format:

```
SequenceFile.Writer writer = new SequenceFile.Writer(fs, conf, seqFile,
Text.class, VectorWritable.class);
// populate vectorWritable with a boolean vector, one entry per person
writer.append(new Text(companyName), vectorWritable);
```

- Execution is done by calling a “driver” method:

```
KMeansDriver.run(conf, seqFile, clusters, clusteringOutputPath, measure,
convergence_threshold, maxIter, produceClusterOutput, removeOutliers,
userHadoop);
```

MatchFWD: Problem



MatchFWD: Details

- Distance: we want to tell how similar are two companies based on the people who worked for both

$$distance\left(company_1, company_2\right) = \frac{|people\ worked\ for\ both|}{|people\ worked\ in\ either|}$$

- The actual distance incorporates an extra item for company size
- We re-cluster when clusters are too big producing a type of hierarchical clustering
- We then computed relevant statistics about the chances of overlap due to randomness for inter-cluster distances

MatchFWD: Results

- 17k companies into 3k clusters.
- Still not enough data.
  - Only 10% of the matches can use the clusters
- For example, here are some companies “similar” to RadialPoint (a 173 entries cluster):

INM The world's largest independent mobile advertising network.

CNW Group Connecting organizations to relevant news audiences through integrated, intelligent communications and disclosure services.

The Createch Group A leading Canadian consulting company specialized in the integration of ERP and CRM solutions.

Manwin An industry-leading IT firm specialising in entertainment media, web marketing and development.

QNX Software Systems Global leader in realtime operating systems.

Apache Mahout: Some Problems

- The authors have written a book "Mahout in Action"
  - Book is actually very good
- But code has evolved from book
  - To use you'll need to look at the code continuously
- Silly bugs
  - Driver script classpath won't work
- Silly omissions
  - Can't cluster instances with labels
- Hadoop support is on and off

Outline

The Bad News

- Difficult to mantain
  - Link between data and trained model is easy to get lost
  - You'll be dealing with errors (defects) and very few ways to solve them
  - Adding more data, if it helps, will produce lots of regressions (asymptotic behavior)
  - Not all errors are the same, but they look like that in the reported metrics
- Your compile time just begun to be measured in hours (or days)
  - Time to upgrade... your cluster.
- Be prepared to stare into the void every time you are asked about odd system behavior

Thoughtland

- Visualizing n-dimensional error surfaces
- Machine Learning with Weka (cross-validated error cloud)
- Clustering with Apache Mahout (using model based clustering)
- Text Generation (using OpenSchema and SimpleNLG)
- http://thoughtland.duboue.net
  - Scala
  - Open source: https://github.com/DrDub/Thoughtland

Summary

- Don't be afraid of getting your hands dirty
- Try to incorporate some trained models in your existing work
  - But don't forget about testing
  - And keeping track of the input data
  - And don't train at the customer's computer
- Pick a library, any library, and give it a try with existing data sets:
  - UCI Machine Learning Repository: http://www.ics.uci.edu/~mlearn/
  - TunedIT: http://tunedit.org/

Contacting the Speaker

- Email: pablo.duboue@gmail.com
- Website: http://duboue.net
- Twitter: @pabloduboue
- LinkedIn: http://linkedin.com/in/pabloduboue
- IRC: DrDub
- GitHub: https://github.com/DrDub
- Always looking for new collaboration opportunities
  - Very interested in teaching a class either in Montreal or on-line