Machine learning for hackers and why it matters for Free Software

Pablo Ariel Duboue

Les Laboratoires Foulab
Montreal, Quebec

Observe, Hack, Make 2013

Outline

Three Potential Attendees

- Activists (lighting talk)
- **Hackers** (makers, developers, tinkerers)
- Machine learning practitioners

Dream Outcomes

- Creation of a SourceForge-like site but for FLOSS data
    - See http://commoncrawl.org (100Tb of Web data!)
- Creation of a cross over of TikiWiki with Wikipedia but for trained "programs"
- A hacker approaches ML as a practitioner rather than from theoretical perspective
- An activist helps create a GPL/CC crossover license that protects community driven data efforts

ML & FLOSS

- Issues
  - Are ML models (discussed next) the preferred form for modification?
  - Practical aspects are several order of magnitude more complicated
- Threats:
  - Obsolescence (source code is less valuable)
  - Yet-another-GPL-circumvention trick
- Opportunities
  - Contributing code is difficult, contributing and curating data is easier
  - Turning users into contributors

What is Machine Learning

- Statistical modelling with focus on predictive applications.
- Common case (“supervised learning”):
  - Training/estimation/”compilation?”
    - input: vectors of features, including target feature (**data**)
    - output: trained **model**
  - Execution/prediction/”interpretation?”
    - input: vector of features (w/o target feature) plus trained model
    - output: predicted target feature

Example.

- Stanford Syntatic Parser.
  - Java, GPL licensed
  - Mature code, surprisingly well-written
- Probabilistic Context Free Grammar (2Mb trained model)
  - Source: Penn Treebank 640Mb (compressed)

-
  ```
  (S (NP (DT An) (VBG operating) (NN system))
    (VP (VBZ is)
      (NP
        (NP (DT the) (NN set))
        (PP (IN of)
          (NP (JJ basic) (NNS programs)
          (CC and)
          (NNS utilities)))
       (SBAR (WHNP (WDT that)) (S (VP (VBP make) (NP (PRP$ your) (NN
  ```

What is a Model?

- Depends on the machine learning methodology employed.
  - Some models **are** easy to understand and modify by hand.
  - Example from a biology text classification system (to be discussed later)
    - if the word DATA is NOT present *before* the term
      and the word DEVELOPMENT is present *after* the term$\Longrightarrow$ class gene [91.7%]
    - if the word FRAGMENT is NOT present *before* the term
      and the word ALLELE is present *before* the term
      and the word THAT is NOT present *after* the term$\Longrightarrow$ class gene [93.9%]
    - if the word ENCODES is NOT present *before* the term
      and the word ENCODES is present *after* the term$\Longrightarrow$ class gene [96.5%]

Incomprehensible Models

- Most models being used nowadays are not intendend to be understood as such nor modified by hand
    - Neural networks
    - Support Vector Machines
    - Markov Models
    - Conditional Random Fields

Threats to Freedom

- The main threat is **obsolescence**
  - What are we going to do if type of applications users grow to expect and enjoy in privative platforms rely on large train sets?
  - Not unlike the threat posed by cloud services being addressed by the FreedomBox foundation.
- Applications such as
  - OCR (book scanning)
  - Speech Recognition (dictation)
  - Computer Vision (automatically tag your friends on photos)
  - Question Answering (Siri / Watson)

Diminishing Value Behind Source Code

- Value on the data
  - Facebook
  - LinkedIn
  - Google+
  - Flickr
- Data vendors
  - http://www.infochimps.com/marketplace (general data, including Twitter data)
  - http://www.ldc.upenn.edu (linguistic data)

Yet-another-clever-GPL-circumvention trick?

- Vendor releases the source code but keeps the data behind the trained model closed.
- Not unlike firmware binary blobs?
  - To me, the firmware binary blobs are a much better analogy to machine learning models than video game assets.

Threats to Practicality

- Training machine learning models takes a whole different type of build-machine
  - 64Gb of RAM for 3 days, sure!
  - Why? Oh my, why?
- Distributing training data involves order of magnitude more space and bandwidth
  - Comparable to wikimedia mirroring (or more)

Opportunities for Specific Projects

- Main challenge for Free Software IMO is to change users into contributors
- Contributors volunteering new training data can follow the success case of Translators
- Data contributors can
  - Annotate more data to fix a bug ("data patches")
  - Curate existing data (think Wikipedia)

Opportunities for Multi-project Collaboration

- Inter-project collaboration opportunities.
  - Sharing data is easier than sharing code as its format seldom changes.
    - Think object-orientation.
    - All syntactic parsers in the last 15 years of work in the field have used the same Penn Treebank data set.
  - Sharing annotation work is easier than sharing data patches.
    - Think work on i10n

Sky is the limit

- Machine learning can enable the creation of computer programs without programming
- "Teaching" the computer what to do
- While we are still far from that, current $f(x_1, \ldots, x_n) \rightarrow y$ type of functions are possible

About the Speaker

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

- Columbia University
  - Natural Language Generation
  - Thesis: "Indirect Supervised Learning of Strategic Generation Logic", defended Jan. 2005.
- IBM Research Watson
  - Question Answering
  - Deep QA - Watson system (Jeopardy)
- Independent Researcher, living in Montreal
  - Collaboration with Universite de Montreal
  - Free Software projects and consulting for startups

Outline

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
  - They considered bugs in the code same are the same as model issues
  - My experience has been quite the opposite, the code you write implementing 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)
- ...
- ...

Outline

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

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

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

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

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

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

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

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

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

Decision Trees

- Find the partition of the data with higher information gain

  Value of a piece of gossip

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

- 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];
}
```

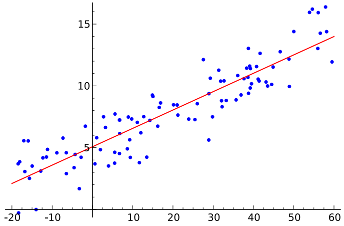
Biology: Results

- 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



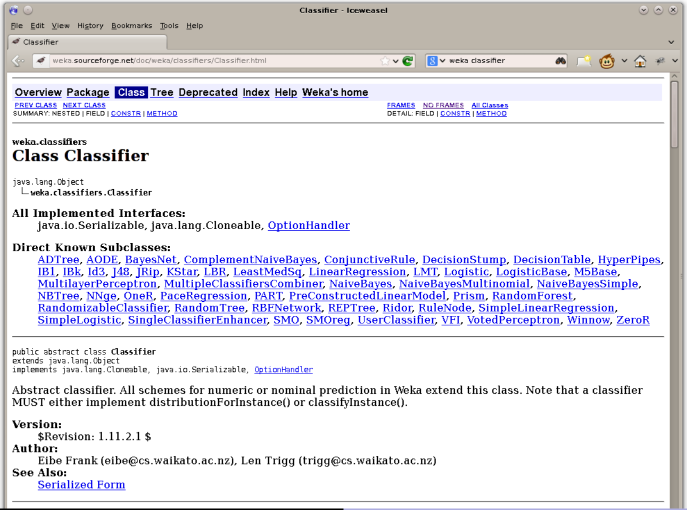
(Wikipedia)

- 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
FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD
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
```

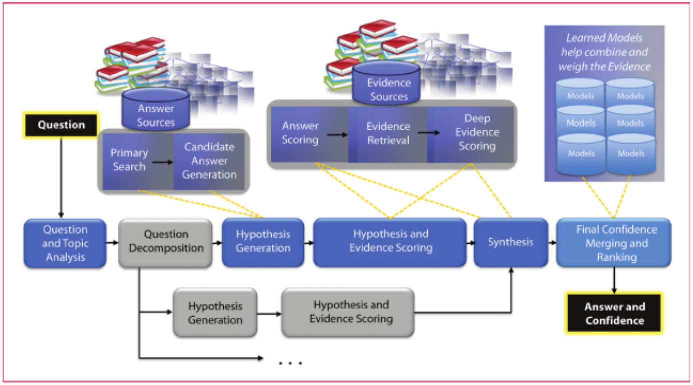
Abstract classifier. All schemes for numeric or nominal prediction in Weka extend this class. Note that a classifier MUST either implement distributionForInstance() or classifyInstance().

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

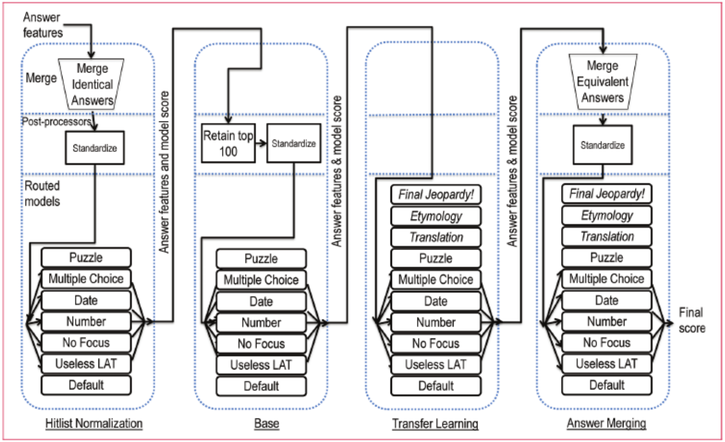
- Learning to rank
  - Rather than predicting a class, choose the best one among many instances
  - In the Jeopardy!™ case, the instances were candidate answers
- Features related to each particular answer candidate
  - “evidence”
- As logistic regression produces a goodness of fit, it can be used for 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

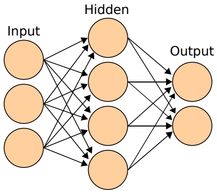
- An existing part-of-speech tagger for the French language was a mixture of Python and Perl
  - Instead of re-engineering it, ran it on a many French docs
  - Trained a new MaxEnt model on it
- Took less than 2 days of work and produced a Java POS tagger at about 5% the same performance as the original
- More complicated than WSD, as it involves more classes and every word has to be tagged
  - with MaxEnt everything thinkable can be used as a feature
- Features include
  - The word itself, previous words, with their identified tags
  - Suffixes and preffixes, up to the 1st & last 4 chars of the word
  - 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

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`
    - `26 new org.apache.tools.ant.BuildException`

Theano

- My current efforts on K4B are centered around Theano, a Deep Learning Python library written at Universite de Montreal
  - http://deeplearning.net/software/theano/
- Deep Learning focuses on
  - using multi-layer neural networks with many layers (deep networks)
  - some layers are trained from the inputs directly
  - they synthesize complex features without access to the output classes
- Key for hackers, Theano creates de neural networks as symbolic structures than then can compile to be run on GPUs

How to Come Up with Features

1. Throw everything (and the kitchen sink) at it
2. Stop and think
    1. What information would **you** use 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 a 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

The Bad News

- Difficult to maintain
    - 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 has began 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

Outline

debian-legal circa 2009.

- Original message: http://lists.debian.org/debian-legal/2009/05/msg00028.html
- Mathieu Blondel asked two questions:
    - Can Debian ship models in main without distributing the original data?
        - Yes, because the model is considered the preferred form for modification.
        - The reasoning followed a pre-existing decision from 2D rendered images for games (rendered from an underlining 3D model).
    - Can violations of data licensing be detected?
        - Artificially introduced errors for fingerprinting

Some Quotes.

- "Free data is important for the very same reason that free programs are!"
  - Mark Weyer (Wed, 27 May 2009 11:36:55 +0200) <20090527093654.GF24759@athen.informatik.hu-berlin.de>
- "[then do not ship] pictures that are initially photographs of an object (the preferred form of modification is the original object; if you want to see it at another angle, you need to take another photograph)"
  - Josselin Mouette (Wed, 27 May 2009 10:33:52 +0200) <1243413232.14420.49.camel@shizuru>

Training Data vs. Features

- Feature vectors are not unlike generated YACC (or Bison) C files.
- Examples
  - Speech
    - Training data: transcribed speech
    - Feature data: wave segments with associated transcription
  - Spelling correction
    - Training data: Wikipedia history
    - Feature data: edits that modify a word with less than 3 characters total edit
  - Syntactic Parsing
    - Training data: newspaper articles bracketed and annotated with syntactic categories
    - Feature data: trees of height one, with the most important word of it (“lexical head”)

Outline

Some Crazy Ideas

- Ever heard of Tiki Wiki? http://tiki.org
  - A project run like a wiki
  - Everybody is granted commit access and the code is very accessible (PHP)
- Imagine a ML equivalent
  - Users can edit the data and download newly trained models right away
  - The models get combined into an end-to-end software system that solves a given issue
- Otherwise acquiring data for general use
  - Maybe build a Free Software-volunteer driven Mechanical Turk-like tool?

Thoughtland

- My current project, 100% Free Software
- Visualizing n-dimensional error surfaces
  - Input: training data + machine learning algorithm
  - Output: a paragraph of text describing how the error surface "looks like" in n-dimensions
- 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 into your existing/new projects
  - But don't forget about testing
  - And keeping track of the input data
  - And don't train at the users' 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 Pablo

- Visit the Foulab Village + unofficial Canadian Consulate!
- Email: pablo.duboue@gmail.com
- Website: http://duboue.net
- Twitter: @pabloduboue
- IRC: DrDub (FreeNode, ##foulab)
- GitHub: https://github.com/DrDub
- LinkedIn: http://linkedin.com/in/pabloduboue