AI, Machine Learning, and NLP Services at CHPC

Sean Igo
Center for High Performance Computing
Sean.Igo@utah.edu
February 10, 2015
Purpose of Presentation

• Brief overview of CHPC environment / access
• Overview of NLP / ML / AI
• Presentation of available NLP/ML/etc software
  – Concentrating on command line utilities on batch-scheduled systems.
    • Interactive software is available
Overview of CHPC Resources

• CHPC's mission: **Support Research!**
  – Faculty, staff, or thesis/dissertation
  – Can also be class research

• CHPC's main focus is High Performance Computing:
  – Massively parallel-processing
  – Remote access to a single high-powered computer
  – Useful for large amounts of data
  – Or if you need an application you don't have
Overview of CHPC Resources

• Other services available:
  – VM Farm: web application hosting, grid, etc.
  – Database consultation and hosting
  – Windows/Macs hosted in datacenter, remote login
  – Limited coding/scripting consultation

• My area of expertise is the HPC (cluster) side
  – Suitable for many researchers' needs
  – Ordinary software as well as supercomputing
Overview of CHPC Resources

• Protected environment
  – For identifiable, IRB-governed PHI data
  – Like general CHPC environment, but smaller
    • Different home directory
    • Same applications available (linux)
  – Cluster, VM farm, dedicated machines, etc.
  – Higher security: HIPAA compliant, soon other standards
    • Requires VPN connection from off campus
    • Physically secure backups
VMs / Dedicated Machines

• Protected VM farm
  – VMs only for use by PI-approved projects
    • Web-based applications using PHI
    • Remote Desktop access

• Dedicated machines
  – Usually purchased by PI from grant funds
    • Can run any OS
    • Maintained and backed up by CHPC staff
Cluster Environment

Recommended approach:

- Edit/compile on interactive
- Create script for batch
- Submit script to scheduler
- Script does this:
  - Copy data to scratch
  - Process data
  - Write output to scratch
  - Copy output to account
  - Clean up scratch
Getting Started at CHPC

• NOTE: CHPC website is currently under construction

• Links in this presentation may change
Getting Started at CHPC

• Account application
  – online: https://www.chpc.utah.edu/apps/profile/account_request.php
  – You need to be doing research for a PI with a CHPC account
    • If your PI isn't signed up, let me know
  – If the online application is unsuitable, paper one is available
    • www.chpc.utah.edu/docs/forms/application.html

• Protected environment access
  – Application not yet automated
  – Mention in application notes that you need HIPAA/Protected
  – Access based on IRB number
Getting Started at CHPC

• Getting started guide
  – www.chpc.utah.edu/docs/manuals/getting_started

• Problem reporting system
  – Easiest: email to issues@chpc.utah.edu
  – OR go to website http://jira.chpc.utah.edu
  – OR find me
Security Policies (1)

• No clear text passwords - use ssh and scp
• Do not share your account under any circumstances
• Don’t leave your terminal unattended while logged into your account
• Do not introduce classified or sensitive work onto CHPC systems
  – Except in the protected environment
• Use a good password and protect it – see gate.acs.utah.edu for tips on good passwords
Security Policies (2)

• Do not try to break passwords, tamper with files, look into anyone else’s directory, etc. – your privileges do not extend beyond your own directory
• Do not distribute or copy privileged data or software
• Report suspicions to CHPC (security@chpc.utah.edu)
• Protected environment data subject to additional policies
General Information

- Available on CHPC web pages
  - https://www.chpc.utah.edu/
- Can get to software info from https://wiki.chpc.utah.edu
  - Again, new site phasing in at the moment
- Available for most packages
  - Can get dated; refer to software homepages
- Has information on licensing restrictions, example batch scripts, where to get more information on a specific package
- Also has useful information on running of jobs
Artificial Intelligence

- The attempt to make computers behave “intelligently”
  - Whatever that means
- A very broad field; practitioners have different aims
  - “problems for which there is no algorithm”
  - modeling of human intelligence (CogSci)
- CHPC use to date has been application-oriented
- **What would you like to hear about?**
  - these slides are mostly general overview
Artificial Intelligence

• I will concentrate on simple theory
• Many different subtypes
  – State-space
  – Rule based
  – Logic
  – Classification
  – Clustering
  – Sequencing
  – ...
• Currently mostly means machine learning
Artificial Intelligence

- GENERALLY:
- Many approaches exist to solving problems
- Best way to proceed:
  - Become familiar with different AI techniques
    - (don't need to be expert, just get a feel)
  - Decide which is the best fit for your problem (may be > 1)
  - Develop a representation accordingly
  - Apply the technique
- This is true of both “classical” AI and ML
Artificial Intelligence

• Often, people will choose the algorithm and try to force their application to fit it.

• Don't do that!

• AI is faddish:
  – 1980s – neural nets
  – 1990s – Genetic Algorithms
  – 2000s – Support Vector machines
  – 2010s – (not sure yet.)

• All of these have their uses but none is magic
Artificial Intelligence

• The central challenge in AI:

  REPRESENTATION!

• How do you describe your problem to be “computable?”
  – able to be operated upon by your chosen algorithm / software?
• I'll demonstrate with Machine Learning (ML)
Quick ML overview

• Common ML task: Supervised Classification
  – Gather a large number of training examples
  – Have human experts classify them
  – Represent as feature vectors
  – ML algorithm trains a model
  – Classify novel instances with the model

• Example: Classifying days wrt whether to play tennis
Machine Learning

- Using data to “train” models by example
- Operates on “instances”: things of interest
  - Patients, customers, sentences, ...
  - You want to know something about the instances, e.g:
    - Patient: likely to be diabetic?
    - Customers: interested in buying Product X?
    - Sentences: relevant to drug interactions?
- Models learned from example instances used to make predictions about new instances
Machine Learning

• Supervised
  – Initial set of example instances given with correct answers
  – Assigned by human experts
  – “gold standard”

• Unsupervised / Semi-supervised
  – No or little human expertise needed
  – Tends to be less reliable and need more data
Feature Engineering

• This is the ML version of representation
• Decide which attributes describe your instances
  – Animals: color, hair length, is_carnivorous,...
  – Patients: age, sex, weight, ...
  – Depends on what you want to find out
• Most of the effort in many ML projects
• A quick example helps to explain...
Quick ML overview

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
</tbody>
</table>

Classic toy ML problem – given the weather, should I play tennis?

(Decision support!)
Quick ML overview

@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
rainy,mild,normal,FALSE,yes
sunny,mild,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes
rainy,mild,high,TRUE,no

Convert the examples to file format for the ML software.

Here, Weka ARFF format
Quick ML overview

ML software trains a model

Here, J48 decision tree

J48 pruned tree

outlook = sunny
  |  humidity = high: no (3.0)
  |  humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
  |  windy = TRUE: no (2.0)
  |  windy = FALSE: yes (3.0)
Decision Tree Classifier

- The type just shown
  - Finds most important feature for discriminating class
  - Creates a branching based on that
  - Does this recursively, creating a “tree”
  - Can also be viewed as a set of rules
- Advantage is that tree / rules human readable
- May not be the best for a given application
Other Classifiers: SVM

• Support Vector Machine
  – treats instances as points on a graph
  – each feature represents a dimension (may be many)
  – finds boundaries separating the classes
    • to the best of its ability

• Default is “linear kernel”
  – Boundary is a line
  – really, multidimensional plane in # features...
Other Classifiers: SVM

• linear kernel, 2 features, 2 classes (red/grn)
Other Classifiers: SVM

- polynomial kernel order 3, 2 features, 2 classes
Other Classifiers: Naive Bayes

• Uses Bayes' Rule
  – Find probability that an instance belongs to class n
  – Given its combination of features
  – Simplification: Assumes all features are independent
  – Often that's close enough
  – Depends on feature engineering
Classifiers: For best results

- Keep number of classes small
  - Ideally, 2
  - For multi-class can use multiple 2-class classifiers
    - A or not-A, B or not-B, etc.
    - Combine predictions through various means
  - Some algorithms do well with multiclass (e.g. neural nets)

- Best if # of instances of each class ~equal
  - Often not true
  - Depends on feature engineering
Evaluating Classifiers

• Supervised
  – With respect to class \( n \)
  – Recall: \( \frac{\text{# correct predictions}}{\text{# of class } n \text{ in gold standard}} \)
  – Precision: \( \frac{\text{# correct predictions}}{\text{# of total predictions made}} \)
  – F-measure: harmonic mean of Recall and Precision
  – Recall aka Sensitivity; Precision = Positive Predictive Value
  – Other metrics exist; R/P/F is common and informative

• Unsupervised
  – Need a different metric... Possibly clustering?
Clustering

• Similar to classification
• Typically unsupervised
  – Can use a gold standard to evaluate cluster purity
• Gather instances into “clusters” by similarity
  – trick is to define similarity
• Different methods:
  – Fixed # clusters
  – Flexible #, e.g. Go until all instances clustered
  – Clusters may overlap or not
Clustering

- Here, fixed 4 clusters (4 classes) – can evaluate w/gold std.
Association Mining

• Looks for patterns of association
• Typically unsupervised
• Find instances that often occur together
  – e.g. in department store data:
    – People who buy baby formula often buy diapers
    – Examples: shopping websites' “you might like”
  – Feedback e.g. customer can click “not interested”
Sequence Finding

• Given n samples in order, predict next
• OR given a sequence, find likelihood it conforms to a model
• Techniques:
  – Conditional Random Fields (CRF)
  – Hidden Markov Models (HMM)
  – ...
So now what?

• There are many more AI techniques
• These have been some of the common ones
  – and the ones I know best
• Moving now to discussing how to use AI / ML
Feature Extraction

• Converting input data to ML software format
• Capturing / synthesizing the features you need
• Input data can be
  – files – text, xml, EHR, ...
  – instrument signals – ECG, Audio...
  – multiple sources
• Output format depends on ML software
  – Common ones: CSV, Weka ARFF
Feature Extraction

• Can be an unexpectedly large software development effort and/or deployment!
  – Be sure to allow time for it in development

• Input data may be difficult to interpret
  – May contain garbage e.g. Email headers
  – A common problem in NLP...
  – May be in undocumented format

• Input and output data may be large
  – Feature extraction takes longer than expected
Typical ML workflow: development

- Decide on representation (feature engineering)
- Gather data / create gold standard
  - Can be time-consuming and expensive!
  - Not as bad for unsupervised methods
- Extract features to make training set / test set
  - Again, different for unsupervised tasks?
- Train model on training set
- Evaluate model against test set
  - Or other metrics e.g. For clustering
Typical ML workflow: deployment

• Export trained model
• Create production software incorporating:
  – Feature extraction methods
  – Prediction method based on model
• For each instance software encounters:
  – Extract features
  – Get prediction from model
  – Continue processing
ML Software at CHPC

• Weka
  – Multi-algorithm ML package; java
• SVMLight, SVMLin, SVMLib
• Orange
  – Python extension for ML
• Others: MegaM, VW
NLP Overview

• Application of many AI techniques
• Get computers to understand human language
  – current CHPC software limited to text, not speech
  – also limited to understanding, not generation
• Started in the 1950s
• Turned out to be much harder than it seemed
• Now NLP consists mainly of distinct tasks
  – Some of the most common supported at CHPC
Typical Initial NLP Pipeline

Input Text

The man's dog was brown.

Tokenizer

Tok. Stream

The, man, 's, dog, was, brown, .

POS Tagger

Tok / Tag Str

The_DT, man_NN, 's_POS, dog_NN, was_VBD, brown_JJ, .

NLP Software at CHPC

- Stanford Tools:
  - Core, Tagger, NER, Parser
- Berkeley Parser
- Pipeline development: GATE, UIMA
- Sundance (U of U Information extraction)
- Metamap (Biomedical NLP)
- NLTK (Python)
MetaMap – Biomedical NLP

- `/uufs/chpc.utah.edu/sys/pkg/metamap/std`
  - Current version metamap10
  - Installed and has java api

- I have done parallelized runs with this

- Same as the online MetaMap utility

- To use, you must sign UMLS agreement
MetaMap – Biomedical NLP

Processing 00000000.tx.1: common cold

Phrase: "common cold"
Meta Candidates (6):

1000 C0009443: Common Cold [Disease or Syndrome]
   Cold
861 C0009264: Cold (Cold Temperature) [Natural Phenomenon or Process]
861 C0205214: Common [Functional Concept, Quantitative Concept]
861 C0234192: Cold (Cold Sensation) [Physiologic Function]
827 C1949981: Colds [Pharmacologic Substance]

Meta Mapping (1000):
1000 C0009443: Common Cold [Disease or Syndrome]

• Can e.g. extract CUI from output
• Output can be XML (structured but VERY verbose)
UIMA and Eclipse

- UIMA is NLP pipeline framework
- Eclipse: IDE w/ UIMA integration
  - Installed on CHPC app tree but deprecated
  - Recommend installing your own instance
  - Unless you can't (e.g. need to open HIPAA data)
- Eclipse requires X-Windows server to be running
  - and ssh -Y to forward X calls from CHPC
Other Software at CHPC

• R
  – Not (?) parallelized versions; if you need it or newer versions let me know

• Matlab + many toolboxes / DCS
  – single node 8-core shared-memory parallel
  – 64 proc. distributed memory (embarrassingly parallel only)

• Python
  – 2.x numpy, scipy, matplotlib; 3.x?

• SAS (subject to licensing)
Finally...

- Let us know if you need some other package
  - Factors: cost, hardware/OS requirements, licensing
- Report problems by emailing issues@chpc.utah.edu
- Any questions – contact me!
  - Sean.Igo@utah.edu
  - CHPC Cubicle: Ask at desk in 405 INSCC (president's circle)
  - BMI: 421 Wakara, #225 (near Dr. Chapman's office)
Questions?
Parallelizing Your Task

• Most packages I've shown are single-processor!
  – Dedicated parallel software would be nice...
  – Can write your own, not always practical
• Different ways to run existing sw on parallel clusters
• Many tasks are “embarrassingly parallel”
  – i.e., capable of being decomposed into sub-problems that do not depend on one another
  – e.g., parsing several text files; can just send \( \frac{1}{n} \) of the files to parsers running on \( n \) processors
Parallelizing Your Task

• Embarrassingly Parallel is nothing to be ashamed of!
• However, dividing tasks naively may not give each processor the same amount of work
• That is, 1/n of the documents/sentences/etc. may not be 1/n of the work
• Initial tests with MetaMap show that dividing over documents is a terrible way to use it
Parallelizing Your Task

- Slightly better: “deal” tasks to processors as they become available
  - with fine granularity should be very efficient
- Other things to consider:
  - Per-invocation overhead
  - Integrating results
- I'm creating scripts to wrap / distribute software calls (command lines)
  - Talk to me if you need something like this
Default Login Scripts

• CHPC maintains default login scripts that set up environment, now updated to work in protected environment
  – www.chpc.utah.edu/docs/manuals/getting_started/code/chpc.tcshrc
  – www.chpc.utah.edu/docs/manuals/getting_started/code/chpc.bashrc

• Copy to your home directory as .tcshrc or .bashrc
  – This is being done on new accounts

• Can comment out setups for packages not used

• Currently NLP/ML packages not included

• Can customize by creating .aliases file that is sourced at end of the CHPC script
Location of Software

- Currently we place most installations at:
  - /uufs/chpc.utah.edu/sys/pkg
    - Accessible on clusters and some desktops
    - all NLP / ML software is here
- Protected environment also uses the chpc.utah.edu tree
- That is for linux only; Windows / Mac interactive nodes have software installed natively