Analysis Tools and Libraries for BigData

Lecture 02
Abhijit Bendale
Office Hours

- Terry Boult (Waiting to Confirm)
- Abhijit Bendale (Tue – 2:45 to 4:45 pm). Best if you email me in advance, but I will try to be in my office in this time.
  - abendale@vast.uccs.edu, abhijitbendale@gmail.com
Agenda

- Introduction to various Statistics and Machine Learning libraries
  - Interface Languages: Python, Matlab, R, Java, C++
  - Case Study with Scikits

- Dataset Repositories
  - Amazon Public Datasets
  - Government Data Resources

- Tips for getting started on your Semester Project
Statistics and Machine Learning Tools

- What do you want to accomplish?
  - Basic Statistical Function
  - Advanced statistical and learning algorithm
  - Using plotting tools to understand trends in Data
  - Performance Evaluation Metrics

- Using the right tools for right task with right amount of abstraction
The speed/flexibility tradeoff

- Matlab Code
- Java Code
- Python Code
- Machine code
- Digital Hardware
- Analog Hardware

Flexibility vs. Speed
Theory Vs. Practice

- **Theoretician**: I want a polynomial-time algorithm which is guaranteed to perform arbitrarily well in “all” situations.
  
  - I prove theorems.

- **Practitioner**: I want a real-time algorithm that performs well on my problem.
  
  - I experiment.

- **Approach for BigData**: I want combining algorithms whose performance and speed is guaranteed relative to the performance and speed of their components.
  
  - You want to do both, or at least the latter.
Tools at hand...

scikit-learn
machine learning in Python

For use with Python

lib C++ Library
Machine Learning With C++

GUI based, for use with Java
# Introduction to Scikits-Learn

## scikit-learn

*Machine Learning in Python*

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

---

### Classification

- Identifying to which set of categories a new observation belong to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ...

---

### Regression

- Predicting a continuous value for a new example.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ...

---

### Clustering

- Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ...

---

### Dimensionality reduction

- Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, Isomap, non-negative matrix factorization.

---

### Model selection

- Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross validation, metrics.

---

### Preprocessing

- Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction.
Objectives

- Understanding Features and Feature Extraction
- Knowing basics of Classification / Regression
  - Supervised Classification
  - Unsupervised Classification
  - Understand Difference between linearly separable and non-linearly separable data
What is Machine Learning?

- A sub-field of Artificial Intelligence
- Often called as applied statistics
- Goal is to learn (understand) nature of given set of observation and build a predictive model for new observation
Features and Feature Extraction

Most machine learning algorithms implemented in scikit-learn expect a numpy array as input X. The expected shape of X is \((n\_samples, n\_features)\).

\[
\begin{bmatrix}
10 & 11 & 25 & 128 & 220 & 245 & \ldots \\
\end{bmatrix}_{nrows \times ncols}
\]
Why feature extraction?

- Data often unstructured:
  - Text Documents
  - Sound
  - Climate measurements
  - Images
  - Videos

- Transform into more structured format
Feature Space and Linear Decision Boundary

Class A

Class B

Linear Decision boundary

Dimension 2

Dimension 1
Iris Flower Dataset

Iris Setosa  Iris Versicolor  Iris Virginica

Measurements:
- Sepal length in cm
- Sepal width in cm
- Petal length in cm
- Petal width in cm

Target Classes
- 0 - Iris Setosa
- 1 - Iris Versicolor
- 2 - Iris Virginica
## Iris Flower Dataset

### Iris Setosa

### Iris Versicolor

### Iris Virginica

<table>
<thead>
<tr>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.1</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td><em>I. setosa</em></td>
</tr>
<tr>
<td>7.0</td>
<td>3.2</td>
<td>4.7</td>
<td>1.4</td>
<td><em>I. versicolor</em></td>
</tr>
<tr>
<td>6.4</td>
<td>3.2</td>
<td>4.5</td>
<td>1.5</td>
<td><em>I. versicolor</em></td>
</tr>
<tr>
<td>6.9</td>
<td>3.1</td>
<td>4.9</td>
<td>1.5</td>
<td><em>I. versicolor</em></td>
</tr>
<tr>
<td>6.8</td>
<td>3.0</td>
<td>5.5</td>
<td>2.1</td>
<td><em>I. virginica</em></td>
</tr>
<tr>
<td>5.7</td>
<td>2.5</td>
<td>5.0</td>
<td>2.0</td>
<td><em>I. virginica</em></td>
</tr>
<tr>
<td>5.8</td>
<td>2.8</td>
<td>5.1</td>
<td>2.4</td>
<td><em>I. virginica</em></td>
</tr>
</tbody>
</table>

| 6.1          | 2.6         | 5.6          | 1.4         | ?             |
2D PCA on Iris Dataset

Transforming the data so that it is easy to operate on
Re-using existing code-base

```python
from itertools import cycle
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.decomposition import PCA

def plot_2D(data, target, target_names):
    colors = cycle('rgbcmykw')
    target_ids = range(len(target_names))
    plt.figure()
    for i, c, label in zip(target_ids, colors, target_names):
        plt.plot(data[target == i, 0],
                 data[target == i, 1], 'o',
                 c=c, label=label)
    plt.legend(target_names)

# Load iris data
iris = load_iris()
X, y = iris.data, iris.target

# First figure: PCA
pca = PCA(n_components=2, whiten=True).fit(X)
X_pca = pca.transform(X)
plot_2D(X_pca, iris.target, iris.target_names)
```
Supervised Learning Model

Training Text, Documents, Images, etc.

Feature Vectors

Machine Learning Algorithm

New Text, Document, Image, etc.

Feature Vector

Predictive Model

Expected Label
Supervised Learning Model

Training Text, Documents, Images, etc.

Labels

New Text, Document, Image, etc.

Feature Vectors

Machine Learning Algorithm

clf.fit(X, y)

X = vec.fit_transform(input)

X_new = vec.transform(input)

Predictive Model

y_new = clf.predict(X_new)

Expected Label
Unsupervised Learning Model

No Labels ..!!!
2.6. Hyperparameters, training set, test set and overfitting

The above SVM example displays an example of hyperparameters, which are model parameters set before the training process. For example, when using an RBF model, we choose the kernel coefficient $\gamma$ before fitting the data. We must be able to then evaluate the goodness-of-fit of our model given this choice of hyperparameter.

The most common mistake beginners make when training statistical models is to evaluate the quality of the model on the same data used for fitting the model:

If you do this, you are doing it wrong!
2.6.1. The overfitting issue

Evaluating the quality of the model on the data used to fit the model can lead to overfitting. Consider the following dataset, and three fits to the data (we’ll explore this example in more detail in the next section).

![Graph showing overfitting and underfitting with different degrees of fit](image)

Examples of over-fitting and under-fitting a two-dimensional dataset.

2.6.2. Solutions to overfitting

The solution to this issue is twofold:

1. Split your data into two sets to detect overfitting situations:
   - one for training and model selection: the **training set**
   - one for evaluation: the **test set**

2. Avoid overfitting by using simpler models (e.g. linear classifiers instead of gaussian kernel SVM) or by increasing the regularization parameter of the model if available (see the docstring of the model for details)
2. Machine Learning 101: General Concepts

Objectives

By the end of this section you will

1. Know how to extract features from real-world data in order to perform machine learning tasks.
2. Know the basic categories of supervised learning, including classification and regression problems.
3. Know the basic categories of unsupervised learning, including dimensionality reduction and clustering.
4. Understand the distinction between linearly separable and non-linearly separable data.

In addition, you will know several tools within scikit-learn which can be used to accomplish the above tasks.

In this section we will begin to explore the basic principles of machine learning. Machine Learning is about building programs with tunable parameters (typically an array of floating point values) that are adjusted automatically so as to improve their behavior by adapting to previously seen data.

More details on sklearn website
There is ready made code for you to try out..!
sklearn.metrics: Metrics

See the Model evaluation: quantifying the quality of predictions section and the Pairwise metrics, Affinities and Kernels section of the user guide for further details.

The sklearn.metrics module includes score functions, performance metrics and pairwise metrics and distance computations.

Model Selection Interface

See the The scoring parameter: defining model evaluation rules section of the user guide for further details.

metrics.make_scorer(score_func[, ...]) Make a scorer from a performance metric or loss function.

Classification metrics

See the Classification metrics section of the user guide for further details.

metrics.accuracy_score(y_true, y_pred[, ...]) Accuracy classification score.
metrics.auc(x, y[, reorder]) Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score(y_true, y_score) Compute average precision (AP) from prediction scores
metrics.classification_report(y_true, y_pred) Build a text report showing the main classification metrics
metrics.confusion_matrix(y_true, y_pred[, ...]) Compute confusion matrix to evaluate the accuracy of a classification
metrics.f1_score(y_true, y_pred[, labels, ...]) Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score(y_true, y_pred, beta[, ...]) Compute the F-beta score
metrics.hamming_loss(y_true, y_pred[, classes]) Compute the average Hamming loss.
metrics.hinge_loss(y_true, y_pred_decision[, ...]) Average hinge loss (non-regularized)
C-Support Vector Classification.

The implementations is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each, see the corresponding section in the narrative documentation: Kernel functions.

**Parameters**

- **C**: float, optional (default=1.0)
  
  Penalty parameter C of the error term.

- **kernel**: string, optional (default='rbf')
  
  Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.

- **degree**: int, optional (default=3)
  
  Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

- **gamma**: float, optional (default=0.0)
  
  Kernel coefficient for 'rbf', 'poly' and 'sigm'. If gamma is 0.0 then 1/n_features will be used instead.
Installing scikit-learn

There are different ways to get scikit-learn installed:

- Install the version of scikit-learn provided by your operating system or Python distribution. This is the quickest option for those who have operating systems that distribute scikit-learn.
- Install an official release. This is the best approach for users who want a stable version number and aren’t concerned about running a slightly older version of scikit-learn.
- Install the latest development version. This is best for users who want the latest-and-greatest features and aren’t afraid of running brand-new code.

Note: If you wish to contribute to the project, it’s recommended you install the latest development version.

Installing an official release

Getting the dependencies

Installing from source requires you to have installed Python (>= 2.6), NumPy (>= 1.3), SciPy (>= 0.7), setuptools, Python development headers and a working C++ compiler. Under Debian-based operating systems, which include Ubuntu, you can install all these requirements by issuing:

```
sudo apt-get install build-essential python-dev python-numpy python-setuptools python-scipy libatlas3-base
```

Note: In order to build the documentation and run the example code contains in this documentation you will need matplotlib:

```
sudo apt-get install python-matplotlib
```
Machine Learning in Action

Peter Harrington

April, 2012 | 384 pages
ISBN: 9781617290183

$44.99 pBook + eBook (includes PDF, ePub, and Kindle)

$35.99 eBook Only (includes PDF, ePub, and Kindle)

Browse all our mobile format ebooks.

RESOURCES

Look Inside
- Preface
- About this book
- Table of Contents
- Index
- Errata

Resources
- Author Online
- Machine Learning with Watson: The Greatest Achievement of Artificial Intelligence to Date (PDF)
- Stochastic Gradient Ascent (PDF)
- The Premise of Machine Learning (PDF)

Downloads
- Source code (34.2 MB)
- Sample chapter 1
- Sample chapter 7
Which method shall I use?
Which method should I use?

- **Standard Answer:** Not really that important
- **Cynical Answer:** Whichever one performs the best
- **Less Cynical Answer:** The model that makes the most reasonable assumptions about your problem domain
Things to do:

- Setup your laptop/desktop with tools of choice
  - Install Machine Learning Library
  - Try out demo examples given with that library

- Try out different algorithms
  - E.g. Support Vector Machines
  - Principal Component Analysis
  - Performance Evaluation Metrics (e.g. Area Under the Curve, Average Precision, Average Classification Error etc.)

- Familiarize yourself with reading documentation and understanding parameters
Introduction to Weka
Machine Learning with Weka

- Comprehensive set of tools:
  - Pre-processing and data analysis
  - Learning algorithms (for classification, clustering, etc.)
  - Evaluation metrics

- Three modes of operation:
  - GUI
  - command-line (not discussed today)
  - Java API (not discussed today)
Sample database: the sensus data ("adult")

- **Binary classification:**
  - Task: predict whether a person earns > $50K a year
  - Attributes: age, education level, race, gender, etc.
  - Attribute types: nominal and numeric
  - Training/test instances: 32,000/16,300

- **Original UCI data available at:**
  ftp.ics.uci.edu/pub/machine-learning-databases/adult

- **Data already converted to ARFF:**
  http://www1.cs.columbia.edu/~galley/weka/datasets/
What we will use today in Weka:

I. Pre-process:
   - Load, analyze, and filter data

II. Visualize:
   - Compare pairs of attributes
   - Plot matrices

III. Classify:
   - All algorithms seem in class (Naive Bayes, etc.)
visualize attributes
Linear classifiers

- Prediction is a linear function of the input

- In the case of binary predictions, a linear classifier splits a high-dimensional input space with a hyperplane (i.e., a plane in 3D, or a straight line in 2D).

- Many popular effective classifiers are linear: perceptron, linear SVM, logistic regression (a.k.a. maximum entropy, exponential model).
Comparing classifiers

- Results on “adult” data
  - Majority-class baseline: \(76.51\%\)  
    (always predict \(\leq 50K\))  
    \texttt{weka.classifier.rules.ZeroR}
  - Naive Bayes: \(79.91\%\)  
    \texttt{weka.classifier.bayes.NaiveBayes}
  - Linear classifier: \(78.88\%\)  
    \texttt{weka.classifier.function.Logistic}
  - Decision trees: \(79.97\%\)  
    \texttt{weka.classifier.trees.J48}
Why this difference?

- A linear classifier in a 2D space:
  - it can classify correctly ("shatter") any set of 3 points;
  - not true for 4 points;
  - we say then that 2D-linear classifiers have capacity 3.

- A decision tree in a 2D space:
  - can shatter as many points as leaves in the tree;
  - potentially unbounded capacity! (e.g., if no tree pruning)
Weka Experimenter

- If you need to perform many experiments:
  - Experimenter makes it easy to compare the performance of different learning schemes
  - Results can be written into file or database
  - Evaluation options: cross-validation, learning curve, etc.
  - Can also iterate over different parameter settings
  - Significance-testing built in.
Weka Experiment Environment

Experiment Configuration Mode:
- Simple
- Advanced

Results Destination
- JDBC database
- Filename

Experiment Type
- Cross-validation
- Classification
- Regression

Iteration Control
- Number of repetitions:
- Data sets first
- Algorithms first

Datasets
- Add new...
- Delete selected

Use relative paths

Algorithms
- Add new...
- Delete selected

Notes
10:33:04: Started
13:41:15: Finished
13:41:15: There were 0 errors
10:33:04: Started
13:41:15: Finished
13:41:15: There were 0 errors
Beyond the GUI

- How to reproduce experiments with the command-line/API
  - GUI, API, and command-line all rely on the same set of Java classes
  - Generally easy to determine what classes and parameters were used in the GUI.
  - Tree displays in Weka reflect its Java class hierarchy.

```java
> java -cp ~/galley/weka/weka.jar weka.classifiers.trees.J48 -C 0.25 -M 2 -t <train_arff> -T <test_arff>
```
Matlab and BigData
## BigData with Matlab

<table>
<thead>
<tr>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Easy to use</td>
</tr>
<tr>
<td>- Great for data analysis: Really nice plotting tools</td>
</tr>
<tr>
<td>- Access to wide array of toolboxes</td>
</tr>
<tr>
<td>- Parallel Computing toolbox</td>
</tr>
<tr>
<td>- GPU computing</td>
</tr>
<tr>
<td>- Almost all machine learning algorithms are available</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Costly</td>
</tr>
<tr>
<td>- Parallelizing means need multiple licenses</td>
</tr>
<tr>
<td>- Rarely used as a backend in production</td>
</tr>
<tr>
<td>- Memory heavy</td>
</tr>
<tr>
<td>- Proprietary</td>
</tr>
</tbody>
</table>

01/28/2014
Machine Learning in Matlab

Classification
Build models to classify data into different categories.

Algorithms: support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbor, Naïve Bayes, discriminant analysis, neural networks, and more

» Get started with introductory examples

Applications: credit scoring, tumor detection, image recognition

Regression
Build models to predict continuous data.

Algorithms: linear model, nonlinear model, regularization, stepwise regression, boosted and bagged decision trees, neural networks, and more

» Get started with introductory examples

Applications: electricity load forecasting, algorithmic trading, drug discovery

Clustering
Find natural groupings and patterns in data.

Algorithms: k-means, hierarchical clustering, Gaussian mixture models, hidden Markov models, self-organizing maps, and more

» Get started with introductory examples

Applications: pattern mining, medical imaging, object recognition
Duct-taping external libraries

Bit tedious and time consuming. You need lot of experience doing this to get good performance.
Matlab Tools for BigData

MATLAB

Single Machine
Single Processor

Multithreaded

Single Machine
Use cores / processors efficiently

Parallel Computing

Use all processors on a machines:
e.g. efficiently using 8-core machine

Distributed Computing

Distribute across the cloud:
e.g. using Amazon EC2 cloud

GPU Computing

Harness power of GPUs

01/28/2014
Statistics Toolbox in MATLAB

- Iris Data Analysis using matlab demo
A major design goal of this portion of the library is to provide a highly modular and simple architecture for dealing with kernel algorithms. Towards this end, dlib takes a generic programming approach using C++ templates. In particular, each algorithm is parameterized to allow a user to supply either one of the predefined dlib kernels (e.g., RBF operating on column vectors), or a new user defined kernel. Moreover, the implementations of the algorithms are totally separated from the data on which they operate. This makes the dlib implementation generic enough to operate on any kind of data, be it column
Dlib C++ Library

Machine Learning Guide

Classification
- svm_one_class_trainer with radial_basis_kernel
- svm_c_linear_dcd_trainer (see one_class_classifiers_ex.cpp example program)
- svm_c_linear_trainer with radial_basis_kernel or histogram_intersection_kernel
- svm_c_linear_trainer
- svm_multiclass_linear_trainer

Data Transformations
- linear_manifold_regularizer
- vector_normalizer_froebetric
- discriminant_pca
- sammon_projection
- cca

Regression
- svr_c_linear_trainer
- krr_trainer with radial_basis_kernel or histogram_intersection_kernel
- svr_trainer with radial_basis_kernel or histogram_intersection_kernel

Clustering
- newman_cluster or chinese_whispers
- kmeans or find_clusters_using_kmeans

Structured Prediction
- structural_assignment_trainer
- structural_object_detection_trainer
- structural_svm_problem (Used to build your own structured prediction model)

Markov Random Fields
- structural_graph_labeling_trainer
- structural_sequence_segmentation_trainer
- structural_sequence_labeling_trainer

START

01/28/2014
Recap
Don’t need to know everything..!

- Pick language of your choice
- Use machine learning tools related to that
- Install ML library
- Try out examples given with the library
- That’s it..!
Publicly Available Large Datasets
Public Data Sets on AWS provides a centralized repository of public data sets that can be seamlessly integrated into AWS cloud-based applications. AWS is hosting the public data sets at no charge for the community, and like all AWS services, users pay only for the compute and storage they use for their own applications. Learn more about Public Data Sets on AWS and visit the Public Data Sets forum.

**Featured Public Data Sets**

**Common Crawl Corpus**
A corpus of web crawl data composed of over 5 billion web pages. This data set is freely available on Amazon S3 and is released under the Common Crawl Terms of Use.

**1000 Genomes Project**
The 1000 Genomes Project, initiated in 2008, is an international public-private consortium that aims to build the most detailed map of human genetic variation available.

**Google Books Ngrams**
A data set containing Google Books n-gram corpuses. This data set is freely available on Amazon S3 in a Hadoop friendly file format and is licensed under a Creative Commons Attribution 3.0 Unported License. The original dataset is available from http://books.google.com/ngrams/.
Amazon Public Datasets

- Wide Range of Domains
  - Astronomy, Biology, Climate, Economics, Mathematics, Encyclopedic, Geographic etc

- Wide Range of Sizes
  - 5 GB to 100s of TB

- Wide Range of Variations
  - Raw Data
  - Annotated Data
Example Datasets

Human Microbiome Project

Wikipedia Traffic Statistics

University of Florida Sparse Matrix Collection

1000 Genomes
A Deep Catalog of Human Genetic Variation

OpenStreet Map
Where can I find large datasets open to the public?

Big data of web log files

5+ Comments • Share (136) • Report • Options

Answer Wiki

Here are many of the links mentioned so far:

Cross-disciplinary data repositories, data collections and data search engines:

- http://usgovxml.com
- http://aws.amazon.com/datasets
- http://databib.org
- http://datacite.org
- http://figshare.com
- http://linkeddata.org
- http://reddit.com/r/datasets
- http://quandl.com
- Social Network Analysis Interactive Dataset Library (Social Network Datasets)
- Datasets for Data Mining
- http://enigma.io

Single datasets and data repositories
Jeff Hammerbacher, Curious.
Votes by Gustav Meeuwenflatser, Robert Morton, Rob McQueen, Michael R. Bernstein, and 147 more.

I'll try to restrict my answers to datasets greater than 1 GB in size, and order my answers by the size of the dataset.

More than 1 TB
- The **1000 Genomes** project makes 260 TB of human genome data available [13]
- The **Internet Archive** is making an 80 TB web crawl available for research [17]
- The TREC conference made the **ClueWeb09** [3] dataset available a few years back. You'll have to sign an agreement and pay a nontrivial fee (up to $610) to cover the sneakernet data transfer. The data is about 5 TB compressed.
- **ClueWeb12** [21] is now available, as are the Freebase annotations, **FACC1** [22]
- **CNetS** at Indiana University makes a 2.5 TB click dataset available [19]
- **ICWSM** made a large corpus of blog posts available for their 2011 conference [2]. You'll have to register (an actual form, not an online form), but it's free. It's about 2.1 TB compressed.
- The **Proteome Commons** makes several large datasets available. Their Personal Genome Project [11], is 1.1 TB in size. There are also over 100 GB in size.

More than 1 GB
- The **Reference Energy Disaggregation Data Set** [12] has data on home energy use; it's about 500 GB compressed.
Tips for Getting Started with your semester Project

- Identify Problem Domain
- Identify dataset you want to work with
- Understand Nature of Data
Tips for Getting Started with your semester Project

Data Source

Identify Problem Domain

Understand Nature of Data

Identify dataset you want to work with

Develop Analysis Techniques

Iterate till you get it right

Can start right away

Performance Analysis, Conclusions, Deploy
Tips for Getting Started with your semester Project

Data Source

Understand Nature of Data

Develop Analysis Techniques

Iterate till you get it right

Performance Analysis, Conclusions, Deploy

Get Feedback from instructors **BEFORE** Project Proposal Presentation
Tips for Getting Started with your semester Project

Identify Problem Domain

Identify dataset you want to work with

Understand Nature of Data

Data Source

Develop Analysis Techniques

Iterate till you get it right

Performance Analysis, Conclusions, Deploy

Develop a detailed design strategy, partial implementation. Get Feedback DURING proposal presentation

01/28/2014
Tips for Getting Started with your semester Project

Data Source
Identify dataset you want to work with

Understand Nature of Data

Develop Analysis Techniques
Iterate till you get it right

Performance Analysis, Conclusions, Deploy

Read data, develop parsing scripts. Understand disk i/o issues related to your data when reading large files
EARLY PROTOTYPE
Tips for Getting Started with your semester Project

1. Identify Problem Domain
2. Identify dataset you want to work with
3. Understand Nature of Data
4. Create a small subset of the data. Quickly prototype your idea
   Test on small subset. Iterate...
5. Develop Analysis Techniques
6. Performance Analysis, Conclusions, Deploy
7. Iterate till you get it right
8. Deploy it to scale
Putting it all together

Pipelining

We have seen that some estimators can transform data and that some estimators can predict variables. We can also create combined estimators:

```python
import numpy as np
from sklearn import linear_model, decomposition, datasets
import matplotlib.pyplot as plt

logistic = linear_model.LogisticRegression()

pca = decomposition.PCA()
from sklearn.pipeline import Pipeline
pipe = Pipeline(steps=[('pca', pca), ('logistic', logistic)])

digits = datasets.load_digits()
X_digits = digits.data
y_digits = digits.target

# Plot the PCA spectrum
pca.fit(X_digits)
plt.figure(1, figsize=(4, 3))
plt.clf()
plt.axes([.2, .2, .7, .7])
plt.plot(pca.explained_variance_, linewidth=2)
plt.axis('tight')
plt.xlabel('n_components')
plt.ylabel('explained_variance_')
```

Highly Recommended..!
Resources


CS341
Project in Mining Massive Data Sets
Spring 2012

Project reports

- Resolving Ambiguity in Product/Brand Names by Evie Gillie, Saahil Shenoy, Corey Stein.
- Halloween Costume Predictions using Twitter by Anirudh Venkatesh, Onkar Dalal, Praveen Bommannavar.
- Social Data and College Statistics by Sean Choi, Elena Grewal, Kai Wen.
- Resolving Student Entities in the Facebook Social Graph by Jim Sproch, Jason Jong.
- Wikipedia: Nowhere to grow by Austin Gibbons, David Vetrano, Susan Biancini.
- Predicting User Purpose on BranchOut by Ben Holtz, Ben Lasley, Garrett Schlesinger.
- Adverse Event Profiles for Multi-drug Combinations by Srinivasan Iyer, Kushal Tayal, Siddhi Soman.
- Wikipedia Mathematical Models and Reversion Prediction by Jia Ji, Bing Han, Dingyi Li.
Example Project Report

Social Data and College Statistics

Sean Choi
Stanford University
yo2seol@stanford.edu

Elena Grewal
Stanford University
etgrewal@stanford.edu

Kai Wen
Stanford University
kaiwen@stanford.edu

ABSTRACT
We correlate aspects of Twitter data with college statistics. We find that the amount of “buzz” about a college on twitter predicts the number of applicants to the college, even when controlling for the number of applicants in the previous year. We also explore various methods to classify the sentiment of tweets about a school. We find that the sentiment of insiders at a college predicts the freshman retention rate, but that this result is explained by average SAT score and school size. The sentiment of Twitter messages about a college does not predict the number of applicants, the acceptance rate, or the graduation rate. The paper adds to the growing literature on the predictive power of social data, documenting its strengths and limitations, and applies these techniques to a novel set of outcomes.

We hypothesize that the “buzz” about a school as measured by the number of tweets that mention a school will be a positive predictor of the number of applicants to the school and the acceptance rate of applicants at the school. In addition, we hypothesize that the sentiment of tweets about a school will predict the number of applicants, as well as other outcomes such as the freshman retention rate and the graduation rate. The sentiments of Twitter users who know more about a school and possibly attend the school should be an even better indicator of measures such as the freshman retention rate and the graduation rate. If those people express positive sentiments about a school then we predict
5.3 Regression results

The regression results indicate that the count of the number of mentions of a college name is predictive of the number of applicants in 2011. The coefficient on the count is positive and statistically significant even when controlling for the number of applicants in 2010 and the size of the school and mean SAT score of the school. In contrast, the sentiment of the tweets is not predictive of the number of applicants when the number of applicants in the prior year is included as well as the other variables. The sample size changes because in some schools there were no negative tweets and so the ratio

Add concrete results from your analysis.
Write Report in IEEE Conference Format

Manuscript Templates for Conference Proceedings

Although IEEE does not require a specific format for their conference articles, IEEE eXpress Conference Publishing provides these optional MS Word and LaTeX templates free for use. If you wish, you may link to this Web page in its entirety. However, we do not recommend that you link to individual files because they may be updated or replaced without notice.

Grateful acknowledgment is made to the IEEE Computational Intelligence Society, which provided the current LaTeX template.

Note: Other templates (maintained by trans@ieee.org) that more closely align with the printed Transactions format are available.

Accessing the templates

- Select Save when the File Download window appears. The files cannot open directly from the server.

<table>
<thead>
<tr>
<th>Microsoft Word 2003</th>
<th>LaTeX Archive Contents (PDF, 63 KB)</th>
<th>LaTeX (Bibliography Files)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>US letter (DOC, 58 KB)</td>
<td>Unix (TAR.GZ, 683 KB)</td>
<td>Unix (TAR.GZ, 307 KB)</td>
</tr>
</tbody>
</table>
Deliverables

- During Proposal Presentation
  - 1 Page Report
  - Project Proposal Presentation

- During Final Presentation
  - Detailed Report (IEEE Conference Format)
  - Presentation
  - You might be asked to show demo, data, results, code
KEEP CALM
WE'RE HERE TO HELP
Use emails aggressively

- Terry Boult: tboult@vast.uccs.edu
- Abhijit Bendale abendale@vast.uccs.edu, abhijitbendale@gmail.com

- We know it is a tough course
  - Start early
  - Seek more help earlier in the semester and gain more independence later on
  - There is not much homework: doesn’t mean you should not do anything..
  - Use code given in documentation to develop your understanding

- For each library/tool there are tons of tutorials/videos available online.