A Quick Introduction to Machine Learning

Paul Rodriguez
(and Mai Nguyen)

SDSC
Overview

- Terminology and Key concepts
- Modeling and Machine Learning
- Main Activities of Modeling
- R and HPC
- Deep Learning and Digit Recognition
Lots of Terms:
Lots of Terms: what are the key ideas?

Lior Rokach
Department of Information Systems Engineering
Ben-Gurion University of the Negev
Lots of Terms: what are the key ideas?

Artificial intelligence
Lots of Terms: what are the key ideas?

Some human-like processing
e.g. face detection, person tracking, object recognition
Lots of Terms: what are the key ideas?
Lots of Terms: what are the key ideas?

gathering data, looking for patterns and associations –

e.g. who buys what products?
Lots of Terms: what are the key ideas?

**BIG DATA:**

e.g. social networks, internet activity, big science, etc…
Lots of Terms: what are the key ideas?

BIG DATA:

e.g. social networks, internet activity, big science, etc…

PREDICTIVE ANALYTICS:

predicting unobserved data,
e.g. recommending movies
(in contrast to classical inferencing about in-sample statistics)
Lots of Terms: what are the key ideas?
Lots of Terms: what are the key ideas?

improving system performance with data

*e.g.* statistical learning,

*e.g.* models with algorithms for fitting parameters
Machine Learning Steps

- Gather & Prepare data
- Explore data – e.g. *know your variables*
- Build Model – e.g. *simple and complex*
- Evaluate

*then perhaps iterate*
Gather and Prepare Data

• Cleaning variables
• Transforming variables
• Organizing variables into data matrix
  (aka ‘data wrangling’ or ‘data munging’) 
• Variable Selection/Dimension Reduction
Explore Data – know your variables

- Summary statistics
- Check missing values
- Visualize:
  - Plot pairwise correlations
  - Plot histograms
Summary Statistics on Variable in R

• Summary statistics on individual variables
  
  \textit{(df} is a data frame \textit{)}

\begin{verbatim}
> mean(df$MinTemp)
[1] 7.265574
>
> var(df$MinTemp)
[1] 36.31026
>
> sd(df$MinTemp)
[1] 6.0258
>
> summary(df$MinTemp)

  Min. 1st Qu.  Median   Mean 3rd Qu.   Max.  
   -5.300  2.850  7.900  7.743  12.800  20.900
\end{verbatim}
Histogram in R

```r
hist(df$Sunshine, col="lightblue", main="Histogram of Daily Sunshine", xlab="Sunshine (Hours)")
```
```
ggpairs(df[c("Sunshine","Pressure9am","Humidity9am","Temp9am")])
```
Machine Learning Models

- Classification
- Regression/Predictive
- Cluster
- Matrix Factorization
- Bayesian (i.e. learning probability distributions)

- Supervised (dependent variable or outcome labels given)
- Unsupervised (no labels)
- Statistical but comes up in HPC settings
A data example: Home Runs and W-L percent

W-L %
(call it ‘y’)

Number of Team HRs 2012 (call it ‘x’)
Recall Linear Regression is Fitting a Line

**the Model:** \( y = f(x, b) = b_0 * 1 + b_1 * x \)
Recall Linear Regression is Fitting a Line – to minimize error

the Model: \( y = f(x, b) = b_0 \times 1 + b_1 \times x \)
Can we just classify winners vs losers based on home runs?

Number of Team HRs 2012 (call it ‘x’)

W-L %
(call it ‘y’)
Can we just classify winners vs losers based on home runs?

W-L %
(call it ‘y’)

Number of Team HRs 2012  (call it ‘x’)

0.35 0.40 0.45 0.50 0.55 0.60

100 120 140 160 180 200 220 240
Can we just classify winners vs losers based on home runs?

W-L %
(call it ‘y’)

Number of Team HRs 2012 (call it ‘x’)
Can we just classify winners vs losers based on home runs?

W-L %
(call it ‘y’)

Number of Team HRs 2012 (call it ‘x’)
Classification uses labelled outcomes

Winners

Losers

Number of Team HRs 2012 (call it ‘x’)
Classification uses labelled outcomes

With one input dimension make a cut off – eg HR>150
A Linear Model for Classification

- 2 classes: +1=Black (WL%\geq .5) -1=Red (WL%<.5)

Q: Classify winning records based on HRs and ERA?

![Graph showing relationship between Number of Team HRs 2012 and ERA]
A Linear Model for Classification

- 2 classes: +1=Black (WL%\geq=.5)   -1=Red (WL%<.5)

Q: Classify winning records based on HRs and ERA?

![Graph showing classification of winning records based on HRs and ERA.]
Model Choices

- **What kinds of functions to use**
  - e.g. Linear vs NonLinear

- **What to Optimize**
  - Minimize Prediction Error
  - Minimize Classification Errors
  - Maximize Probabilities

- **How to Find Parameters**
  - Search space of solutions
  - Constraints and Assumptions
Use logistic function to bound $x$ in $(0,1)$

Linear Reg: $y = f(X^*b)$

- Logistic Reg.
  - Choose class based on $P(y|x)$ and assume independence among $x$
  - Naïve Bayes

- Penalized Reg. (Ridge)
  - Add constraints on $b$ sizes

- Piecewise Linear Reg.
  - Find splits in variables

- Perceptron
  - Minimize class errors

- Classification/Regression Tree
  - Find splits recursively

- Bayes Network
  - Don’t assume indep., get $P(y,x)$
  - Add layers and back-propagate errors

- Neural Network
  - Transform inputs first instead of a ‘hidden’ layer

- Support Vector Machine
  - Take ensemble
  - Random Forest
  - And maximize error margins

---

Model Space Map – in a nutshell
Model Complexity vs Overfitting

more parameters =>
  more complex =>
  more potential to overfit

(so use training and test datasets)
Modeling Recommendations

• Start simple
• Consider trade off as you go more complex
• Find what works in your domain
• Find what works for this model

• R, Python, Matlab: scripting languages with train/predict/test functions
• Weka, KNIME: GUI tools
Pause
Clustering Idea

- Given a set of data can we find a natural grouping?

Essential R commands:

```R
D = rnorm(12, 0, 1) # generate 12 random normal
X1 = matrix(D, 6, 2) # put into 6x2 matrix
X1[,1] = X1[,1] + 4;   # shift center
X1[,2] = X1[,2] + 2;

# repeat for another set of points

# bind data points and plot
plot(rbind(X1, X2),
     xlim = c(-10, 10), ylim = c(-10, 10));
```
Clustering

• **A good grouping implies some structure**
  Interpret and label clusters
  Characterize new points by the closest cluster

• **Kmeans is a standard algorithm**
  Get distances of all points to K cluster centers
  Assign points to closest center
  Recalculate cluster centers

Try different K and select best inter- vs intra-class separations.
Imagine these 2 dimensional input spaces:
Which of these is easy or hard to cluster? (no class labels)
Potential clusters

A

X1

easy

X2

B

easy

X1

X2

C

Hard?

X1

X2

D

X1

X2
Potential clusters

A

X1

easy

X2

B

easy

X1

C

Hard?

X1

X2

D

easy, 4 clusters

X1

X2
Now imaging there are two classes

A

X1

X2

B

X1

X2

C

X1

X2

D

X1

X2
Which are easy or hard to classify?

(i.e., separate red and blue dots with lines)

A

X1

X2

easy separation

B

X1

X2

C

X1

X2

D

X1

X2
Which are easy or hard to classify?
(ie separate red or blue with lines)

A

B

easy separation

C

D
Which are easy or hard to classify? (ie separate red or blue with lines)

A

B

C

D
Which are easy or hard to classify?
(ie separate red or blue with lines)

A

X1

X2

easy separation

B

X1

X2

easy still

C

X1

X2

harder (nonlinear)

D

X1

X2
Which are easy or hard to classify?
(ie separate red or blue with lines)

A

easy separation

B

easy still

C

harder (nonlinear)

D

harder (nonlinear)
Which are easy or hard to classify? (ie separate red or blue with lines)

A  
\[ \begin{align*} 
\text{easy separation} \\
X1 & \quad X2 
\end{align*} \]

B  
\[ \begin{align*} 
\text{easy still} \\
X1 & \quad X2 
\end{align*} \]

C  
In summary: No easy relationship between clusters and classification

D  
\[ \begin{align*} 
\text{harder (nonlinear)} \\
X1 & \quad X2 
\end{align*} \]
Pause
Given a numeric matrix, can we reduce the number of columns?
Given a numeric matrix, can we reduce the number of columns?

- Yes, if features are constant or redundant
Matrix Factorization:

Given a numeric matrix, can we reduce the number of columns?

- Yes, if features are constant or redundant
- Yes, if features only contribute noise (conversely, want features that contribute to variations of the data)
Example: Athletes’ Height by Weight

Find a line that aligns with the data.
Example: Athletes’ Height by Weight

Find a line that aligns with the data.
Find a line that aligns with the data.

the most variance (or spread)
Find a line that aligns with the data.

\[ H = 0.8 * W \]
Find a line that aligns with the data.

\[ H = 0.8 \times W \]

Note that (0,0) and (1,0.8) are points on the line

- OR -

\[
\begin{pmatrix}
1 \\
0.8
\end{pmatrix}
\]

describes a vector
The next direction of most variance.

Weight - Kg (mean centered)

Height - cm (mean centered)

Describes 2nd vector

- AND -

Both vectors are columns in a matrix:

\[
\begin{pmatrix}
1 & -0.8 \\
0.8 & 1
\end{pmatrix}
\]
New axis (AKA features) defined as combinations of old features

Now Rotate Axis
Project all points to first axis. It keeps much of the variance. Possibly reduce dimensions.
• Best Known Factorization Algorithms:
  SVD (singular value decomposition)
  PCA (principle component analysis)

*SVD more generally works on non square matrices*
• pause
1. Get a compute node:

   [Unix]$ :  srun --partition=debug --pty --nodes=1 --ntasks-per-node=24 -t 00:30:00 --wait=0 --export=ALL -A your-account /bin/bash

2. Start R

   [Unix]$ module load R
   [Unix]$ R  (this gets an interactive R session)

quit() to exit R

exit to exit the compute node
Running Jupyter notebook on Comet

1. Login to comet
2. Access compute node: getcpunode
   (or: srun --partition=debug --pty --nodes=1 --ntasks-per-node=24 -t 00:30:00 --wait=0 --export=ALL -A your-account /bin/bash )
3. Start singularity shell
   1. module load singularity
   2. IMAGE=/oasis/scratch/comet/zonca/temp_project/datascience-notebook-e1677043235c_fixjulia_keras_tf.img
   3. singularity exec $IMAGE jupyter notebook --ip=*  
4. on local machine, in browser url edit box, enter the http string shown, but replace localhost with comet-XX-XX.sdsc.edu
5. Open R-introHPC.ipynb or LabMNIST_Final.ipynb
6. After logging out in browser shutdown notebook on Comet with Ctrl-C


![Running Jupyter notebook on Comet](image-url)
From the Jupyter notebook using R

April 4, 2018

1. In this introduction we will explore some useful R functions for data preparation. We will look very quickly at clustering and classification.

   In [1]: #1. First read the data from a CSV file into an R dataframe
   W_df_orig = read.table('weather_orig.csv',
   header=TRUE,
   sep="",
   stringsAsFactors = TRUE)

   dim(W_df_orig)
   1.3662.24

2. Quick way to view some rows:

   In [2]: head(W_df_orig)

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>MinTemp</th>
<th>MaxTemp</th>
<th>Rainfall</th>
<th>Evaporation</th>
<th>Sunshine</th>
<th>WindGustDir</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-11-01</td>
<td>Canberra</td>
<td>8.0</td>
<td>24.3</td>
<td>0.0</td>
<td>3.4</td>
<td>6.3</td>
<td>NW</td>
<td>30</td>
</tr>
<tr>
<td>2007-11-02</td>
<td>Canberra</td>
<td>14.0</td>
<td>26.9</td>
<td>3.6</td>
<td>4.4</td>
<td>9.7</td>
<td>ENE</td>
<td>39</td>
</tr>
<tr>
<td>2007-11-03</td>
<td>Canberra</td>
<td>13.7</td>
<td>23.4</td>
<td>3.6</td>
<td>5.8</td>
<td>3.3</td>
<td>NW</td>
<td>85</td>
</tr>
<tr>
<td>2007-11-04</td>
<td>Canberra</td>
<td>13.3</td>
<td>15.5</td>
<td>39.8</td>
<td>7.2</td>
<td>9.1</td>
<td>NW</td>
<td>54</td>
</tr>
<tr>
<td>2007-11-05</td>
<td>Canberra</td>
<td>7.6</td>
<td>16.1</td>
<td>2.8</td>
<td>5.6</td>
<td>10.6</td>
<td>SSE</td>
<td>50</td>
</tr>
<tr>
<td>2007-11-06</td>
<td>Canberra</td>
<td>6.2</td>
<td>16.9</td>
<td>0.0</td>
<td>5.8</td>
<td>8.2</td>
<td>SE</td>
<td>44</td>
</tr>
</tbody>
</table>

In [3]: tail(W_df)

   Error in tail(W_df): object 'W_df' not found
   Traceback:

   1. tail(W_df)

In [4]: str(W_df)  #Quick view of the basic 'structure' of the data frame
head(W_long)

# optional: write.csv(W_cast, file='Weather_castwide.csv')

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>MinTemp</th>
<th>MaxTemp</th>
<th>Rainfall</th>
<th>Evaporation</th>
<th>Sunshine</th>
<th>WindDir9am</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-11-01</td>
<td>Canberra</td>
<td>8.0</td>
<td>24.3</td>
<td>0.0</td>
<td>3.4</td>
<td>6.3</td>
<td>SW</td>
<td>NW</td>
</tr>
<tr>
<td>2007-11-02</td>
<td>Canberra</td>
<td>14.0</td>
<td>26.9</td>
<td>3.6</td>
<td>4.4</td>
<td>9.7</td>
<td>E</td>
<td>W</td>
</tr>
<tr>
<td>2007-11-03</td>
<td>Canberra</td>
<td>13.7</td>
<td>23.4</td>
<td>3.6</td>
<td>5.8</td>
<td>3.3</td>
<td>N</td>
<td>NNE</td>
</tr>
<tr>
<td>2007-11-04</td>
<td>Canberra</td>
<td>13.3</td>
<td>15.5</td>
<td>39.8</td>
<td>7.2</td>
<td>9.1</td>
<td>WNW</td>
<td>W</td>
</tr>
<tr>
<td>2007-11-05</td>
<td>Canberra</td>
<td>7.6</td>
<td>16.1</td>
<td>2.8</td>
<td>5.6</td>
<td>10.6</td>
<td>SSE</td>
<td>ESE</td>
</tr>
<tr>
<td>2007-11-06</td>
<td>Canberra</td>
<td>6.2</td>
<td>16.9</td>
<td>0.0</td>
<td>5.8</td>
<td>8.2</td>
<td>SE</td>
<td>E</td>
</tr>
</tbody>
</table>

5 Get factors using SVD

In [8]: #1 Get numeric columns only
cols_numeric = sapply(W_df,is.numeric)    # get column classes as a list
W_dfnum = W_df[,which(cols_numeric)]
dim(W_dfnum)

1.328 2.16

In [9]: #2 turn it into a matrix
W_matrix = as.matrix(W_dfnum)

In [10]: #3 mean center data
W_mcntr = scale(W_dfnum,center=TRUE, scale=FALSE)

In [11]: #4 run SVD command
Wsvd = svd(W_mcntr)
str(Wsvd)

List of 3
$ d: num [1:16] 367 307 215 164 109 ... 
$ u: num [1:328, 1:16] -0.03129 -0.01566 0.03569 0.00638 0.00357 ... 
$ v: num [1:16, 1:16] -0.0775 -0.2114 0.0446 -0.0778 -0.128 ...

6 Get some kmean cluster and plot onto first two SVD factors

In [12]: # get Kmeans for 4 clusters, with 10 iterations and 1 starting points
k4 = kmeans(W_mcntr,4,10,1)

# set color scheme
col2use = c('red','blue','black','yellow')

# set cluster assignment in colors
colassignments = col2use[4%*%k4]

W_proj = as.matrix(W_mcntr) %>% Wsvd$v[,1:3]    # project data onto 3 components
plot(W_proj[,1],W_proj[,2],col=colassignments,main='data pts project to 1,2 SVD compone

# to plot center points, first project them into components
c3 = k4%centers%*% Wsvd$v[,1:3]
points(c3[,1],c3[,2],pch=5,cex=2)
data pts project to 1,2 SVD components, colored by kmeans

In [13]: # Y was created above, use it to select 2 colors

# get class assignment in colors
colassignments = col2use[Y]
plot(W_proj[,1],W_proj[,2],col=colassignments,main='data pts project to 1,2 SVD compone
In [16]: #get model predictions (more generally we would use a test set to get prediction accuracy)
   Y_pred=linmodel_result$fitted.values

   #get the indices of predictions NO vs YES
   Y_pred1_indices =which(Y_pred<1.5)
   Y_pred2_indices =which(Y_pred>=1.5)

   #set up No,Yes predictions
   Y_pred_class =matrix(1,length(Y),1)
   Y_pred_class[Y_pred2_indices]=2

   #show a confusion matrix
   table(Y,Y_pred_class)

   Y_pred_class
   Y  1  2
   1 259  9
   2  31 29

In [18]: #Now color the predictions onto the 2 SVD dimensions

   colassignments = col2use[Y_pred_class]

   plot(W_proj[,1],W_proj[,2],col=colassignments,main='data pts project to 1,2 SVD components')
data pts project to 1,2 SVD components, colored by PREDICTED class.
# Now plot the incorrect cases in different colors

Ycol_ind = Y_pred_class
Ycol_ind[Yerr1_ind] = 3  # set the color for errors, where true-value=1
Ycol_ind[Yerr2_ind] = 4  # and true-value = 2

colassignments = col2use[Ycol_ind]

plot(W_proj[,1],W_proj[,2],col=colassignments,main='data pts project to 1,2 SVD components, Correct (R,Bl) and Errors (Y,Bk)')

# Notice some of the incorrect class cases are next to the other class data in these two dimensions, but other error cases
# are next to correctly predicted class cases. It suggests that other dimensions have useful information
data pts project to 1,2 SVD components, Correct (R,B1) and Errors (Y,B1)
• pause
Quick R in HPC
Paul Rodriguez
SDSC
A typical R development workflow

- **R studio**: An Integrated development environment for R on your local machine – good for development

  - **Edit window to Build scripts**
  - **R console**
  - **Menu tab**
  - Environment Information on variables and command history
  - Plots, help docs, package lists
R commands in brief

• A typical R code workflow:

```r
#READ DATA (housing mortage cases)
X = read.csv('hmda_aer.csv', header=T, stringsAsFactors=T)

#SUBSET DATA
indices_2keep = which(X[, 's13'] %in% c(3, 4, 5))
X = X[unique(indices_2keep),]

#CREATE/TRANSFORM VARIABLES
pi_rat = as.numeric(X[, 's46']/100)  #debt2income ratio
race = as.numeric(X[, 's13'] %in% c(3, 4))  #make race values 1-4 into values 0 or 1
deny = as.numeric(X[, 's7']==3)  #make deny values into 0 or 1,
    # 1 only for deny='3'

#RUN MODEL and SHOW RESULTS
lm_result = lm(deny~race+pi_rat)  #lm is 'linearmodel'
summary(lm_result)
```
R strengths for HPC

• Sampling/bootstrap methods
• Data Gathering and Preparation
• Particular Statistical procedures that you won’t find implemented anywhere else, e.g.
  Multiple Imputation methods,
  Instrument Variable (2 stage) Regression
  Matching subjects for pairwise analysis
  MCMC routines
R Scaling In a nutshell

- R takes advantage of math libraries for vector operations
- R packages provide multicore, multimode, or distributed data (SparkR) options
- However, model implementations not necessarily built to use parallel backends
  - Some models more amenable to parallel versions
Solving Linear Systems: Performance with R, 1 compute node

R:
 glm(Y~X,family=gaussian)  # gaussian regression (like lm)
 glm(Y~X,family=binomial)  # logistic regression (Y=0 or 1)

Data Matrix Size (i.e. square, rowsXcol)

GLM: logistic
Gradient descent

Wall Time (secs)
30min

Matrix inverse solutions
GLM: Gaussian
lm()
inverse

QR matrix decomposition solutions
Solve(a,b)
qr()
Machine learning models: Performance on 1 compute node

- GLM: logistic - 20 secs
- Kmeans
- lm()
- SVM classification (support vector machine optimization)
- svd factoring
- Gradient descent
- svd factoring

Data Matrix Size (i.e. square, rowsXcol)

Wall Time (secs)
R multicore

- Run loop iterations on separate cores

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)
allocate workers
```
R multicore

- Run loop iterations on separate cores

```r
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

my_data_frame = .....  

my_results = foreach(i=1:24,.combine=rbind) %dopar%
{   
    your code here

    return( a variable or object )
}
```

allocate workers

%dopar% puts loops across cores, (loops are independent)
%do% runs it serially
R multicore

- Run loop iterations on separate cores

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

my_data_frame = .....  # Allocate workers

my_results = foreach(i=1:24,.combine=rbind) %dopar% {
    ... your code here
    return( a variable or object )
}

%do% runs it serially
%dopar% puts loops across cores, (loops are independent)
returned items ‘combined’ into list, by default
specify to combine results into array with row bind
```
R multicore

• Run loop iterations on separate cores

BEWARE: foreach will copy data it thinks is need to every core

install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

my_data_frame = ..... 

my_results = foreach(i=1:24,.combine=rbind) %dopar%
{ 
    your code here
}

return(a variable or object)

allocate workers
%dopar% puts loops across cores, (loops are independent)
%do% runs it serially

returned items 'combined' into list by default

specify to combine results into array with row bind
**R multinode: parallel backend**

- Run loop iterations on separate nodes

```r
install.packages('doSNOW')
library('doSNOW')
...
cl <- makeCluster( mpi.universe.size()-1, type='MPI' )
clusterExport(cl,c('data'))
registerDoSNOW(cl)

results = foreach(i=1:47,.combine=rbind) %dopar% {
  ... your code here
  return( a variable or object )
}
stopCluster(cl)
```

allocate cluster as parallel backend

%dopar% puts loops across cores and nodes
R multinode: parallel backend

- Run loop iterations on separate nodes

BEWARE: foreach will copy data it thinks is needed to every node – that can take a long time!

```r
install.packages('doSNOW')
library('doSNOW')
...
cl <- makeCluster( mpi.universe.size()-1, type='MPI' )
clusterExport(cl,c('data'))
registerDoSNOW(cl)

results = foreach(i=1:47,.combine=rbind) %dopar% {
  ... your code here

  return( a variable or object )
}
stopCluster(cl)
```

allocate cluster as parallel backend
%dopar% puts loops across cores and nodes
Another option for (embarrassingly)
Parallel R

1. Split up data into N parts
Another option for (embarrassingly) Parallel R

1. Split up data into N parts
2. In slurm batch script:
   `ibrun -np processors My-perl-script`

My-perl-script:
  get cpu-id &
  pass it to R
Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   ibrun -np processors My-perl-script

My-perl-script:
get cpu-id & pass it to R

Init MPI and get MPI rank

No other MPI calls made
Another option for (embarrassingly) Parallel R

1. Split up data into N parts
2. In slurm batch script:
   `ibrun -np processors My-perl-script`

   CPU Core 1
   My-perl-script: get cpu-id & pass it to R

   CPU Core 2
   My-perl-script: get cpu-id & pass it to R

   ...  

   CPU Core N
   My-perl-script: get cpu-id & pass it to R
Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   ibrun -np processors  My-perl-script

   My-perl-script: get cpu-id & pass it to R

   CPU Core 1

   R script: process dataset 1

   CPU Core 2

   R script: process dataset 2

   ... 

   CPU Core N

   My-perl-script: get cpu-id & pass it to R

   R script: process dataset N
Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   `ibrun -np processors My-perl-script`

   - CPU Core 1
     - My-perl-script: get cpu-id & pass it to R
     - R script: process dataset 1

   - CPU Core 2
     - My-perl-script: get cpu-id & pass it to R
     - R script: process dataset 2

   - CPU Core N
     - My-perl-script: get cpu-id & pass it to R

   ... (Repeat for each CPU core)

   Final R script: combine N outputs
Another option for (embarrassingly) Parallel R

1. Split up data into N parts

2. In slurm batch script:
   `ibrun -np processors My-perl-script`

   - CPU Core 1
     - My-perl-script: get cpu-id & pass it to R
     - R script: process dataset 1

   - CPU Core 2
     - My-perl-script: get cpu-id & pass it to R
     - R script: process dataset 2

   ... ...

   - CPU Core N
     - My-perl-script: get cpu-id & pass it to R
     - R script: process dataset N

Final R script: combine N outputs

More programming but more flexible
#!/bin/bash

# -----------------------------
# slurm script for a batch job on comet
# to run a task on individual cores
# -----------------------------
#SBATCH --job-name="packR"
#SBATCH --output="serial-pack.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 1:00:00
#SBATCH -A sds164

bash

#Generate a hostfile from the slurm node list
export SLURM_NODEFILE=`generate_pbs_nodefile`
module load R

#launch 24x2=48 tasks on 48 cores,
# and start this perl script on each task
ibrun --npernode 24 --tpp 1 perl ./bundlerP.pl

#One can also run hybrid:
# launch 1 process per node, with 24 threads, and
# use doParallel
ibrun --npernode 1 --tpp 24 perl ./bundlerP.pl
Get current cpu id and number of processes

execute R and pass the rank id as an argument
Scaling doParallel vs ‘Packing’ R sessions

- Packing *independent* R sessions onto cores is more flexible for:
  - data management
  - large number of separate models
  - large variation in time per model
  - large matrix operations repeated
  - hybrid multimode/multicore scripts

*But requires more programming or preprocessing*
Example: scaling MCMC

Distributed Markov Chain Monte Carlo for Bayesian Hierarchical Models, Frederico Bumbaca, UCIrvine, et al in print

- Probabilities of user web activity interdependent through a hierarchical model
- MCMC search for probabilities made independent through a phased approach.
- Ran on SDSC Comet with ‘serial packing’ parallelization

(Using rhierMnlRwMixturefunction in the R package, bayesm)

<table>
<thead>
<tr>
<th># Individuals</th>
<th>Cores</th>
<th>Individ per Core</th>
<th>Total Minutes (I/O time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 million</td>
<td>1,7282 (max)</td>
<td>~ 58K</td>
<td>206 (38)</td>
</tr>
</tbody>
</table>
Example: scaling MCMC
Localizing social media hot spots (work in progress with UCIrvine)

• Individual spatial mixture models for users’ geocoded social media use

• MCMC search for location probabilities are independent across users, but convergence time varies depending on user variations

• Ran on SDSC Comet with ‘serial packing’ parallelization, with many cores for short runs, then few cores for longer runs

(using Rgeoprofile package with MCMC)

<table>
<thead>
<tr>
<th># Individuals</th>
<th>Cores</th>
<th>Approx Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>~3000</td>
<td>192-288</td>
<td>2-3</td>
</tr>
<tr>
<td>~2000</td>
<td>48-96</td>
<td>4-8</td>
</tr>
<tr>
<td>~100</td>
<td>24</td>
<td>12-24</td>
</tr>
</tbody>
</table>
Example: scaling likelihoods
Social network evolution (work in progress with UTDallas)

- A large model of users’ connections with interdependent variance terms for different actions

- Optimization, with ~70M observations (5-8Gb), takes > 48 hours on 1 compute node.

- R parallel copies too much data across nodes or cores
- R-mpi not flexible enough with nodes and cores

- Ran with ‘serial packing’ parallelization on parts of data across nodes, with R parallel across cores (but not all cores),
  
  (using Optim, doParallel, and send results back to main node through files)

<table>
<thead>
<tr>
<th># Connections</th>
<th>Cores</th>
<th>Approx Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>~70M</td>
<td>288</td>
<td>2-3</td>
</tr>
</tbody>
</table>
Installing your own R Packages

- In R:
  ```r
  install.packages(‘package-name’)  
  ```

  (see [https://cran.r-project.org/](https://cran.r-project.org/) for package lists and reviews)

- on Comet:
  ```r
  install.packages(‘package-name’,  
    repos='http://cran.us.r-project.org',dependencies=TRUE)
  ```

If compiling is required and you get an error, call support
R-studio

- R studio: An Integrated development environment for R on your local machine – good for development

*now available on XSEDE through Jetstream cloud*
Other R packages:

- Rspark - R interface to Spark
- pdbR - MPI-based support for distributed matrix (better for dense matrices vs Spark)
- Rgputools – GPU support
- Ff, bigmemory; Revolution Scale R – map data to files

Quick-R cheat sheets: https://www.statmethods.net/index.html
distribute data across nodes
distribute data across nodes

Hold data **in memory** across nodes
Hold data in memory across nodes

Run code on each part and gather as requested
- Distributed implementations of common ML algorithms and utilities
- APIs for Scala, Java, Python, and R
- Scales well for independent processes
• pause
Quick Deep Learning
Paul Rodriguez SDSC
Deep Learning

• 3 characterizations:

  1. Learning complicated interactions about input

  2. Discovering complex feature transformations

  3. Using neural networks with many layers

My explanation strategy: Start with linear regression and go deep
Recall Linear Regression is Fitting a Line

the Model: \( y = f(x, b) = b_0 \times 1 + b_1 \times x \)
Classification uses labelled outcomes

Winners

Losers

Number of Team HRs 2012 (call it ‘x’)
Can do better: fit a nonlinear function

The Model: \( y = f(x, b) = \frac{1}{1 + \exp[-(b_0 * 1 + b_1 * x)]} \)
Can do better: fit a nonlinear function

The Model: \( y = f(x, b) = 1/(1 + \exp[-(b_0 \times 1 + b_1 \times x)]) \)

Output is ‘squashed’ to (0,1) range -
Can do better: fit a nonlinear function

the Model: \( y = f(x, b) = 1/(1 + \exp[-(b_o * 1 + b_1 * x)]) \)

Output is ‘squashed’ to (0,1) range - And predictions are like probabilities
Logistic Regression as 1 node network

Input $X_1$

Call $b_0$ ‘bias’

Note: call betas weights

$w_1$

logistic function to transform input to output – call it the ‘activation’ function

Output Value = $1/(1 + \exp[-("bias" \times 1 + "weight"1 \times x)])$
Logistic Regression as 1 node network

Call $b_0$ ‘bias’

$w_1$'s output

Input $X_1$

logistic function to transform input to output – call it the ‘activation’ function

Note: other activations are possible,

RELUs (rectified linear units)
Next step: More general networks

Add input variables
More general networks

(assume bias present)

\[ X_1 \quad \quad \quad \quad X_2 \]

Add input variables    Add logistic transformations …
More general networks

(assume bias)

\[ X_1 \rightarrow \circ \rightarrow \circ \rightarrow \text{output value} \]

Add input variables \quad Add logistic transformations \ldots

\textit{Combine transformations!}
More general networks

(assume bias)

\[ X_1 \xrightarrow{\text{hidden layer}} \text{output value} \]

\[ X_2 \xrightarrow{\text{hidden layer}} \text{output value} \]

Add input variables \quad Add logistic transformations ...
Logistic function w/various weights

\[ f(y) = \frac{1}{1 + \exp(- (b + w_1 x))} \]

\( b = 0, w_1 = 1 \)
Logistic function w/various weights

\[ f(x) = \frac{1}{1 + \exp(- (b + w_1 * x))} \]

- \( b = 0, w_1 = 1 \)
- \( b = 0, w_1 = 5 \)
Logistic function w/various weights

For \( y = \frac{1}{1 + \exp(- (b + w_1 * x))} \)

- \( b = 0, w_1 = 1 \)
- \( b = 0, w_1 = 5 \)
- \( b = 2, w_1 = 1 \)
- \( b = 0, w_1 = -1 \)
So combinations are highly flexible and nonlinear

\[ b = 1, w_1 = 1 \]

\[ b = 1, w_1 = -1 \]
So combinations are highly flexible and nonlinear
So combinations are highly flexible and nonlinear

\[ b = 1, w_1 = 1 \]

\[ b = 1, w_1 = -1 \]

\[ X_1 \]

\[ X_2 \]
So combinations are highly flexible and nonlinear

\[ b = 1, w_1 = 1 \quad + \quad b = 1, w_1 = -1 \]

\[ = \]

\[ b = 1, w_1 = 1 \]

\[ b = 1, w_1 = -1 \]
Why stop at 1 hidden layer?

More hidden layers => More varied features and transformation
Layers of function combinations

\[
x = \text{seq}(-5, 5, .1)
\]
\[
y_1 = 1/(1+\exp(10+10x))
\]
\[
y_2 = 1/(1+\exp(-5+(-10)x))
\]
\[
y_3 = 1/(1+\exp(1+1*y_1+1*y_2))
\]
\[
\text{plot}(x, y_3, \text{type}="l")
\]
\[
y_4 = 1/(1+\exp(10+(-10)x))
\]
\[
y_5 = 1/(1+\exp(-5+(10)x))
\]
\[
y_6 = 1/(1+\exp(1-1*y_4-1*y_5))
\]
\[
\text{plot}(x, y_6, \text{type}="l")
\]
\[
y_7 = 1/(1+\exp(1+2*y_3+1*y_6))
\]
\[
\text{plot}(x, y_7, \text{type}="l")
\]
But parameter fitting is harder too

For each data instance:

\[ \text{Error} = \text{Output} - \text{Target} \]
But parameter fitting is harder too

For each data instance:

\[ \text{Error} = \text{Output} - \text{Target} \]

The objective is to minimize \( \text{Error} \) related to output weights (same as for logistic regression)
But parameter fitting is harder too

For each data instance:
\[ \text{Error} = \text{Output} - \text{Target} \]

But, error signals are only known for output layer, what is error for hidden layer?

The objective is to minimize \( \text{Error} \) related to output weights (same as for logistic regression).
For each data instance:

\[
\text{Error} = \text{Output} - \text{Target}
\]

\(X_1\)
\(X_2\)

Hidden Layer → Output Layer

For each data instance: \(\text{Error} = \text{Output} - \text{Target}\)

But, error signals are only known for output layer, what is error for hidden layer?

The objective is to minimize \(\text{Error}\) related to output weights (same as for logistic regression)

Solution: Minimize \(\text{Error}\) related to output weights, that is also related to hidden weights

(Use derivatives to ‘back-propagate’ errors, “stochastic gradient descent”)

But parameter fitting is harder too
Train with Care

More hidden layers => More varied features and transformations

But:
More layers => more parameters
Train with Care

More hidden layers => More varied features and transformations

But:
More layers => more parameters => Smaller error for each especially at lower layers
Train with Care

More hidden layers => More varied features and transformations

But:
More layers => more parameters => Smaller error for each especially at lower layers

Need:
More data and computing power (gpu)
Train with Care

More hidden layers => More varied features and transformations

But:
More layers => more parameters => Smaller error for each especially at lower layers

Need:
More data and computing power (gpu), functions that don’t saturate(RELU)
Feature Transformations, Projections, and Convolutions
A Simple Transformation

START WITH:
3 input variables fully connected (dense) to 1 hidden node

Assume $b_0=0$, assume all $X$ normalized between 0 and 1

Use Linear Unit: Output = $W \times X$
A Simple Transformation

START WITH:
3 input variables fully connected (dense) to 1 hidden node

Assume $b_0=0$, assume all $X$ normalized between 0 and 1

Use Linear Unit: Output = $W \times X$

Take:
Let $[w_1 \ w_2 \ w_3] = [1 \ 0 \ 0]$

What feature transformation $W \times X$ is that?
A Simple Transformation

START WITH:
3 input variables fully connected (dense) to 1 hidden node

Assume $b_0=0$, assume all $X$ normalized between 0 and 1

Use Linear Unit: Output = $W \times X$

Take:
Let $[w_1, w_2, w_3] = [1, 0, 0]$

What feature transformation $W \times X$ is that?

Informally, squash $X_1$ and ignore $X_2, X_3$
A Simple Transformation

3 input variables fully connected (dense) to 3 hidden nodes

For node 1 let \[ w_1 \ w_2 \ w_3 \] = \[ 1 \ 0 \ 0 \]
For node 2 let \[ w_1 \ w_2 \ w_3 \] = \[ 0 \ 1 \ 0 \]
For node 3 let \[ w_1 \ w_2 \ w_3 \] = \[ 0 \ 0 \ 1 \]

*What feature transformation \( W^*X \) are these together?*
A Simple Transformation

3 input variables fully connected (dense) to 3 hidden nodes

For node 1 let $[w_1 \ w_2 \ w_3] = [1 \ 0 \ 0]$

For node 2 let $[w_1 \ w_2 \ w_3] = [0 \ 1 \ 0]$

For node 3 let $[w_1 \ w_2 \ w_3] = [0 \ 0 \ 1]$

*What feature transformation $W^*X$ are these together?*

Informally, squash 3D to another 3D space
A Factor Transformation

3 input variables fully connected (dense) to 3 hidden nodes

For node 1 let $[w_1, w_2, w_3] = [1, 0.8, 0]$

For node 2 let $[w_1, w_2, w_3] = [-0.8, 1, 0]$

For node 3 let $[w_1, w_2, w_3] = [0, 0, 0]$

What feature transformation $W*X$ are these together?
A Factor Transformation

3 input variables fully connected (dense) to 3 hidden nodes

For node 1 let \([w_1 \ w_2 \ w_3] = [1 \ 0.8 \ 0]\)

For node 2 let \([w_1 \ w_2 \ w_3] = [-0.8 \ 1 \ 0]\)

For node 3 let \([w_1 \ w_2 \ w_3] = [0 \ 0 \ 0]\)

What feature transformation \(W^T X\) are these together?

Informally, like projection onto 2 orthogonal dimensions (recall PCA example on Athletes Height and Weight!)
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)

For node 1 let $W = [w_1 w_2 w_3] = [-1 1 -1]$

What values of $X$ have maximize $W^*X$?
(assuming $W$ are just +/- 1)
Many X input, but only 3 connections to each hidden node
*(a local connectivity pattern, aka receptive field)*

For node 1 let $W = [w_1, w_2, w_3] = [-1, 1, -1]$

*What values of $X$ have maximize $W^*X$?*  
*(assuming $x$ are just +/- 1)*

Let $X = W = [-1, 1, -1]$

Informally, node 1 has max activation for a ‘spike’,
e.g. when $X_2$ is positive and $X_1, X_3$ are negative
For node 1 let $W = [w_1 \ w_2 \ w_3 ] = [-1 \ 1 \ -1]$

For node 2, 3, etc… copy $W$ for node 1

What is the hidden layer doing?
For node 1 let $W = [w_1, w_2, w_3] = [-1, 1, -1]$

For node 2, 3, etc… copy $W$ for node 1

**What is the hidden layer doing?**

Informally, looking for a spike at every node.
A Filter

Many X input, but only 3 connections to each hidden node
(a local connectivity pattern, aka receptive field)

For node 1 let $W = [w_1, w_2, w_3] = [-1, 1, -1]$  
For node 2, 3, etc… copy W for node 1

What is the hidden layer doing?  
Informally, looking for a spike at every node.

Copying weights is like sliding W across input.
A Filter

Many X input, but only 3 connections to each hidden node (a local connectivity pattern, aka receptive field)

For node 1 let \( W = [w_1, w_2, w_3] = [-1, 1, -1] \)

For node 2,3, etc… copy \( W \) for node 1

What is the hidden layer doing?

Informally, looking for a spike everywhere.

Copying weights is like sliding \( W \) across input.

Note: This is essentially a convolution operator.
For node 1 let $W = [w_1, w_2, w_3] = [-1, 1, -1]$

For node 2, 3, etc… copy $W$ for node 1

What is the hidden layer doing?

Informally, looking for a spike everywhere.

Copying weights is like sliding $W$ across input.

Note: This is essentially a convolution operator.

Note: if we take max activation across nodes (‘Max Pool’) then it’s like looking for a spike anywhere.
2D Convolution

Now let input be a 2D binary matrix, e.g. a binary image) fully connected to 1 node.

What $W$ matrix would ‘activate’ for a upward-toward-left diagonal line?
2D Convolution

Now let input be a 2D binarized 3x3 matrix fully connected to 1 node

What $W$ matrix would ‘activate’ for a upward-toward-left diagonal line?

How about:

$$W = \begin{bmatrix}
-1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & -1
\end{bmatrix}$$
2D Convolution

For full image, 1 filter is applied to 1 region in 1 color channel at a time, and then slid across regions (or done in parallel with copied weights) and produces 1 new 2D image (hidden) layer.

Convolution Layer parameters:
- filter size depends on input:
  - smaller filters for smaller details
  - 2 layers of 3x3 ~ 1 layer of 5x5
- sliding amount:
  - smaller better but less efficient
- number of filters:
  - depends on task
  - each filter is a new 2D layer

Convolution Network:
- many layers and architecture options
Large Scale Versions

• Large (deep) Convolution Networks are turning out to be feasible with GPUs (some are 100+ layers)
• Need large amounts of data and many heuristics to avoid overfitting and increase efficiency
Large Scale Versions

- Zooming in:

Each convolution layer uses RELU (rectified linear activation units instead of logistic function) and is followed by Max Pooling layer (over 2D regions with sliding)

The thickness is the number of different convolutions, i.e. different transformations, sometimes called “channels”
Large Scale Versions

- Zooming in:

  Last convolution layer is laid out as a vector for input into classification layers. Classification uses dense, i.e. fully connected, hidden layers and output layer.
What Learned Convolutions Look Like

What Learned Convolutions Look Like
Summarizing Deep Layers

• **Hidden layers transform input into new features:**
  • Feature can be highly nonlinear
  • Features as a new space of input data
  • Features as projection onto lower dimensions (compression)
  • Features as filters, which can be used for convolution

• **But also:**
  • Many algorithm parameters
  • Many weight parameters
  • Many options for stacking layers
Feature Coding vs Discovery

- Edge detection with Support Vector Machine
  OR
  Convolution Neural Network?

- With small datasets and obvious features, SVMs can work well

- But building features is hard, and large classification problems can benefit from common features, so CNNs are better to discover features for multiclass outputs
• pause
Tutorial

- MNIST database of handwritten printed digits
- The ‘hello world’ of Conv. Neural Networks
- Use Keras front end (high level neural functions) to Tensorflow engine (neural math operations)
- Works with GPU or CPUs
1. Login to comet

2. Access compute node: 
   `srun --partition=debug --pty --nodes=1 --ntasks-per-node=24 -t 00:30:00 --wait=0 --export=ALL -A your-account /bin/bash`

3. Start singularity shell
   1. module load singularity
   2. `IMAGE=/oasis/scratch/comet/zonca/temp_project/datascience-notebook-e1677043235c_fixjulia_keras_tf.img`
   3. `singularity exec $IMAGE jupyter notebook --ip=*`

4. On local machine, in browser url edit box, enter the http string shown, but replace localhost with comet-XX-XX.sdsc.edu

5. Open R-introHPC.ipynb or LabMNIST_Final.ipynb

6. After logging out in browser shutdown notebook on Comet with Ctrl-C
for i in range(0,3):
    im = Image.fromarray(Y_train[i,:,:])
    im.save("Xtrain_num"+str(i)+"_cat"+str(Y_train[i])+".jpeg")

plt.figure()
plt.imshow(im,'gray')
plt.show()

print('img_load done')
print (time.strftime("%H:%M:%S"))

(5000, 28, 28)
<matplotlib.figure.Figure at 0x2ad45d04f2e8>
3x3 first convolution layer filter and activation
9x9 first convolution layer filter and activation

Check out:  http://scs.ryerson.ca/~aharley/vis/conv/flat.html
Transfer Learning – Feature Extraction

Extract features here and feed to separate classifier

Source: https://www.cs.toronto.edu/~frossard/post/vgg16/
Learning Segmentation (deconvolve)
The Zoo

• Machine learning/convolution network frameworks:

Tensorflow, pyTorch (libraries and API to build graphs of networks and processing)

Keras - higher level CNN library with tensorflow (best for learning)
Caffe – C/C++ library with many pretrained models
Caffe2 – Facebook tookover Caffe, Pytorch (has a good model for people detection)
YOLO/Darknet – A C++ library, with object detection
Matlab – CNN functions, and pretrained networks

• Also many networks pretrained on large or particular object classes are available: AlexNet, VGG19, Googlenet, Detectron
Google tool for objects, faces, text

- Google Vision api – object recognition network
Caffe2, Facebook “Detectron” networks

Object Detection
ie getting a bounding box

Object Segmentation
ie getting a mask

Object Parts
ie getting keypoints
References

- **Book**: https://mitpress.mit.edu/books/deep-learning
- **Documentation**: https://keras.io/
- **Tutorials I used (borrowed)**: