Machine Learning Big Data using Map Reduce

By

Michael Bowles, PhD

Where Does Big Data Come From?

-Web data (web logs, click histories)

-e-commerce applications (purchase histories)

-Retail purchase histories (Walmart)

-Bank and credit card transactions

What is Data Mining?

-What page will visitor next visit? Given:

- Visitor's browsing history
- Visitor's demographics

-Should card company approve transaction that's waiting? Given:

- User's usage history
- Item being purchased
- Location of merchant.

-What isn't data mining

- What pages did visitors view most often?
- What products are most popular?

Data mining tells us something that isn't in the data or isn't a simple summary.

Approaches for Data Mining Large Data

-Data Mine a Sample

Take a manageable subset (fits in memory, runs in reasonable time)
Develop models

-Limitations of this method?

Generally, more data supports finer grained models
e.g. making specific purchase recommendations
"customers who bought …. " requires much more data than "top ten most popular are …"

Side Note on Large Data:

-Rhine's paradox –

ESP experiment in 50's

1 in 1000 can correctly identify color (red or blue) of 10 cards they can't see

Do they have ESP?

-Bonferroni's principle – Given enough data any combination of outcomes can be found

Is this a reason to avoid large data sets?

No

It's a reason to not draw conclusions that the data don't support (and this is true no matter how large the data set)

Use Multiple Processor Cores

-What if we want to use the full data set? How can we devote more computational power to our job?

-There are always performance limits with a single processor core => Use multiple cores simulataneously.

-Traditional approach – add structure to programming language (C++, Java)

-Issues with this approach

High communication costs (if data must be distributed over a network)

Difficult to deal with CPU failures at this level – (Processor failures are inevitable as scale increases)

-To deal with these issues Google developed Map-Reduce Paradigm

What is Map-Reduce?

-Arrangement of compute tasks enabling relatively easy scaling.

-Includes:

Hardware arrangement – racks of CPU's with direct access to local disk, networked to one another

File System – Distributed storage across multiple disks, redundancy

-Software processes running on various CPU in the assembly

Controller – manages mapper and reducer tasks, fault detection and recovery

Mapper – Identical tasks assigned to multiple CPU's for each to run over its local data.

Reducer – Aggregates output from several mappers to form end product

-Programmer only needs to author mapper and reducer. The rest of the structure is provided.

Dean, Jeff and Ghemawat, Sanjay. **MapReduce: Simplified Data Processing on Large Clusters** http://labs.google.com/papers/mapreduce-osdi04.pdf

Mike Bowles – Machine Learning on Big Data using Map Reduce Winter, 2012

Simple Code Example (using mrJob) – (this is running code)

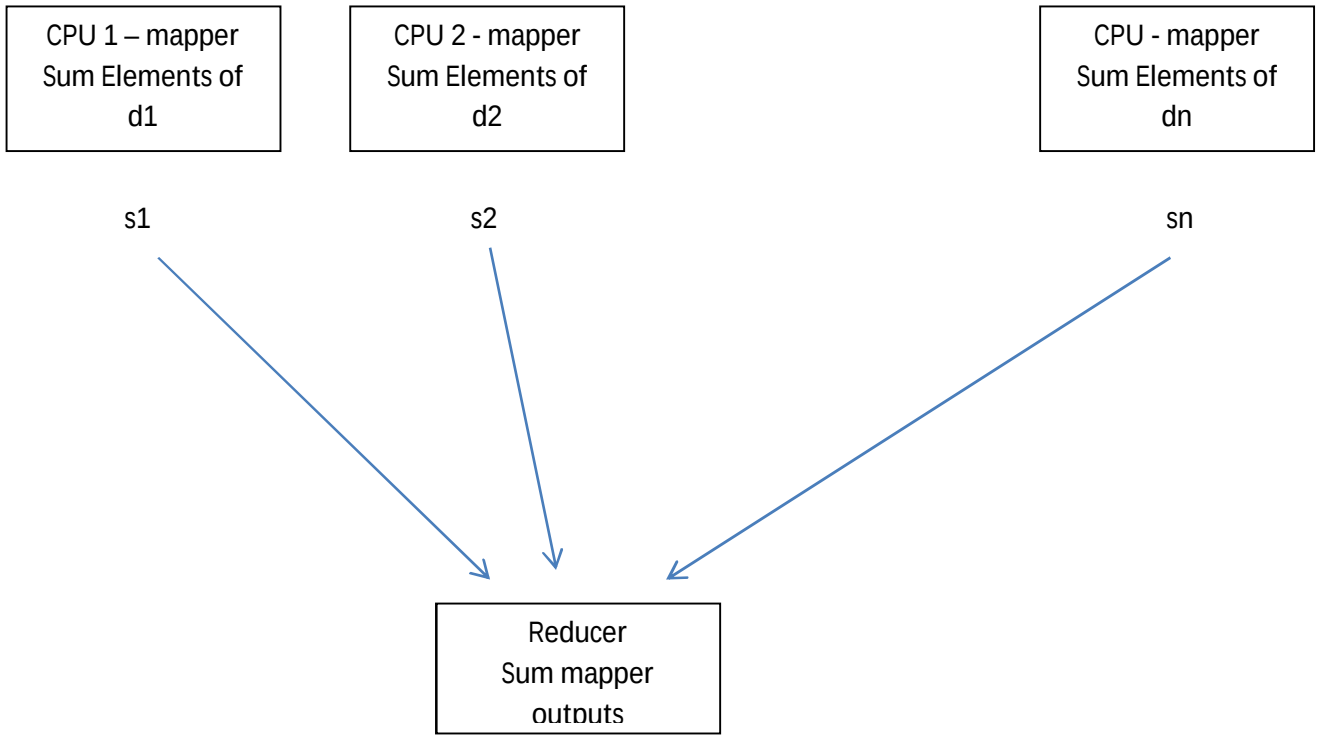
```
from mrjob.job import MRJob
from math import sqrt
import json
class mrMeanVar(MRJob):
   DEFAULT_PROTOCOL = 'json'

   def mapper(self, key, line):
      num = json.loads(line)
      var = [num,num*num]
      yield 1,var

 def reducer(self, n, vars):
      N = 0.0
      sum = 0.0
      sumsq = 0.0
      for x in vars:
         N += 1
         sum += x[0]
         sumsq += x[1]
      mean = sum/N
      sd = sqrt(sumsq/N - mean*mean)
      results = [mean,sd]
      yield 1,results

if __name__ == '__main__':
   mrMeanVar.run()
```

Example – Sum Values from a Large Data Set

-Data set D divided into d1, d2, … dn (each a list of things that can be summed – say real numbers or vectors)

-Mappers running on CPU 1, CPU 2, … CPU n

-Each mapper forms a sum over its piece of the data and emits the sum s1, s2, … sn.



Machine Learning w. Map Reduce

-Statistical Query Model (see ref below) – A two-step process

1. Compute sufficient statistics by summing some functions over the data
2. Perform calculation on sums to yield data mining model
3. Conceptually similar to the "sum" example above.

-Consider ordinary least squares regression

Given m outputs y_i (also called labels, observations, etc.)

And m corresponding attribute vectors x_i (also called regressors, predictors, etc.)

Fit a model of the form $y = \theta^T x$ by solving $\theta^* = \min_\theta \sum_{i=1}^{m} (\theta^T x_i - y_i)^2$

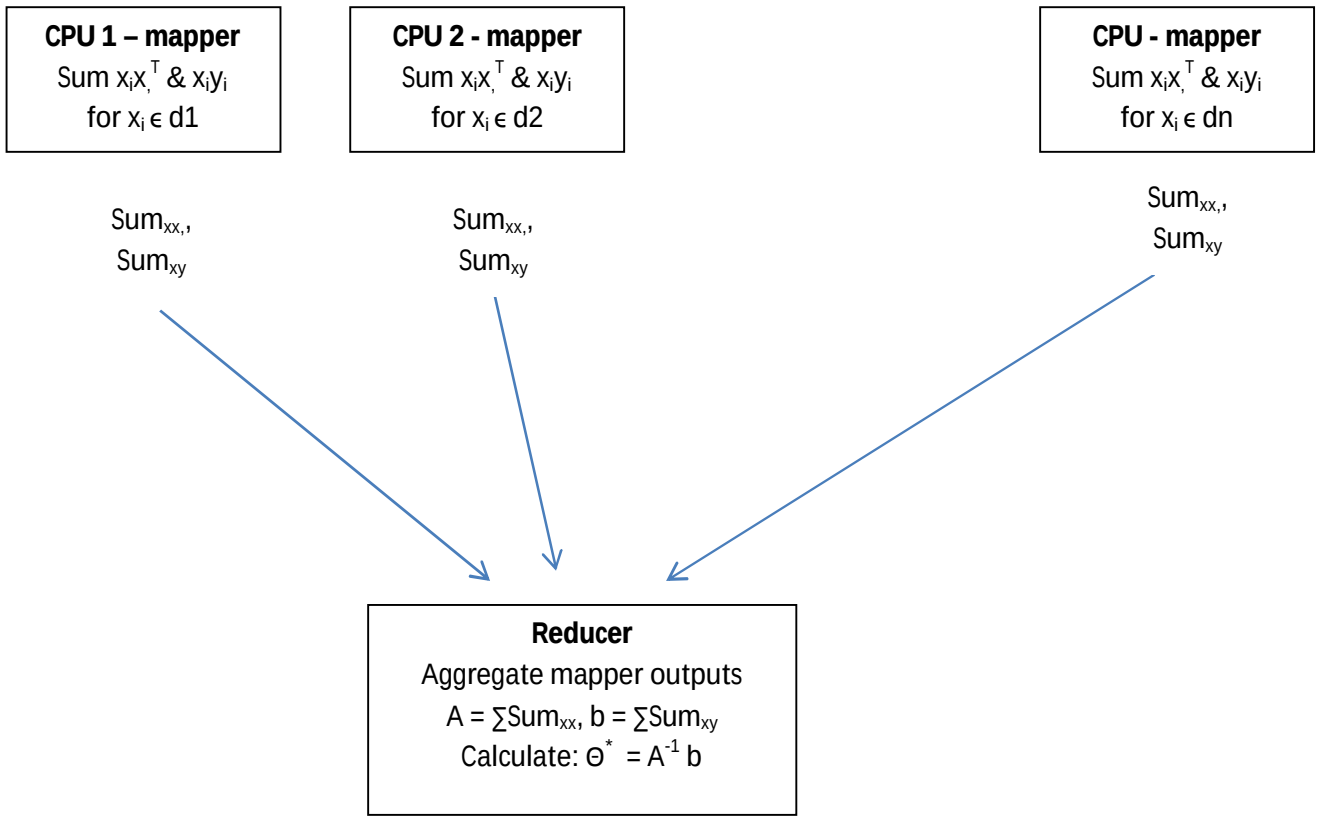
$\Theta^* = A^{-1} b$ where $A = \sum_{i=1}^{m} (x_i x_i^T)$ and $b = \sum_{i=1}^{m} x_i y_i$

-See the natural division into mapper and reducer? (Hint: Look for a ∑)

Map-Reduce for Machine Learning on Multicore http://www.cs.stanford.edu/people/ang/papers/nips06-mapreducemulticore.pdf

Machine Learning with Map Reduce

With OLS the mappers compute partial sums of the form $\sum(x_i x_i^T)$ and $\sum x_i y_i$
The reducer aggregates the partial sums into totals and completes the calculation $\Theta^* = A^{-1} b$



Mike Bowles – Machine Learning on Big Data using Map Reduce Winter, 2012

Machine Learning w. Map Reduce

-Referenced paper demonstrates that the following algorithms can all be arranged in this Statistical Query Model form:

Locally Weighted Linear Regression,
Naïve Bayes,
Gaussian Discriminative Analysis,
k-Means,
Neural Networks,
Principal Component Analysis,
Independent Component Analysis,
Expectation Maximization,
Support Vector Machines

-In some cases, iteration is required. Each iterative step involves a map-reduce sequence

-Other machine learning algorithms can be arranged for map reduce – but not in Statistical Query Model form (e.g. canopy clustering or binary decision trees)

More Map Reduce Detail

-Mappers emit a key-value pair <key, value>

controller sorts key-value pairs <key, value> by key

Reducer gets pairs grouped by key

-Mapper can be a two-step process

-With OLS, for example, we might have had the mapper emit each $x_i x_i^T$ and $x_i y_i$, instead of emitting sums of these quantities

Post processing the mapper output (e.g. forming $\sum$) is a mapper function called "combiner"

Reduces network traffic

Some Algorithms

-Canopy Cluster -

-K-means

-EM algo for Gaussian Mixture Model

-Glmnet

-SVM

Canopy Clustering

-Usually used as rough clustering to reduce computation (e.g. search zip code for the closest pizza versus search the world)

-Find well distributed initial conditions for other clustering also (e.g. kmeans)

-Algorithm:

Given set of points P and distance measure d(,)

Pick two distance thresholds $T1 > T2 > 0$

Step 1: Find cluster centers

1. Initialize set of centers C = null
2. Iterate over points $p_i \in P$

   If there isn't $c \in C$ s.t. $d(c,p_i) < T2$

   Add p_i to C

   get next p_i

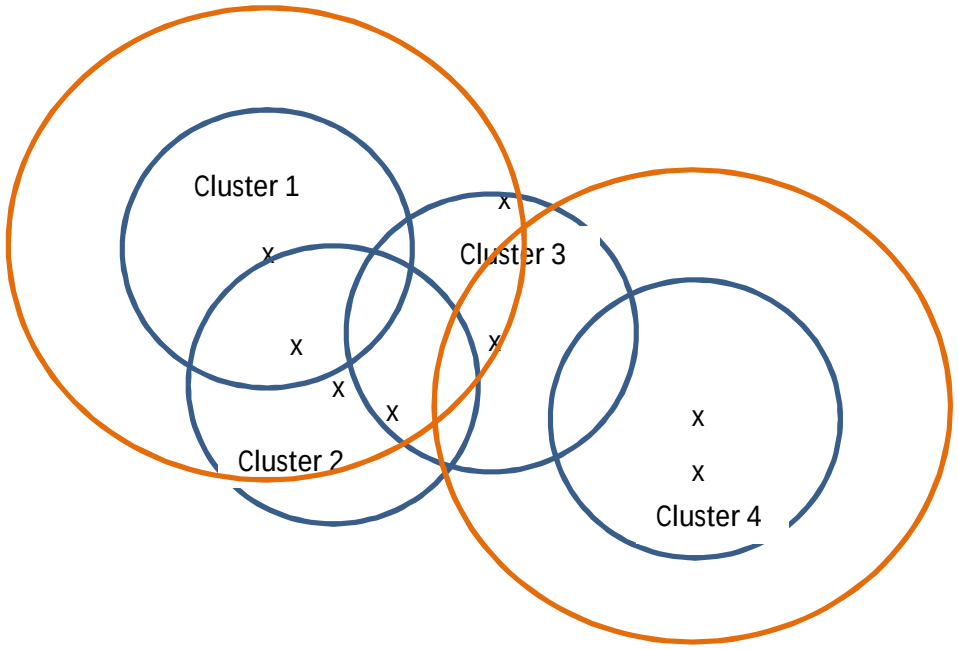
Step 2: Assign point to clusters

For $p_i \in P$ assign p_i to $\{c \in C : d(p_i , c) < T1\}$

-Notice that points generally get assigned to more than one cluster.

McCallum, Nigam, and Ungar "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching", http://www.kamalnigam.com/papers/canopy-kdd00.pdf

Canopy Clustering Picture



Canopy Clustering w Map Reduce

1st Pass – find centers

Mappers – Run canopy clustering on subset, pass centers to reducer

Reducer – Run canopy clustering on centers from mappers.

2nd Pass – Make cluster assignments (if necessary)

Mappers – compare points p_i to centers to form set $c_i = \{c \in C \mid d(p_i , c) < T1\}$

Emit <key, value> = $<c , p_i>$ for each $c \in c_i$

Reducer – Since the reducer input is sorted on key value (here, that's cluster center), the reducer input will be a list of all the points assigned to a given center.

-One small problem is that the centers picked by the reducer may not cover all the points of the combined original set. Pick T1 > 2*T2 or use larger T2 in reducer in order to insure that all points are covered.

The Apache Mahout project has a lot of great algorithms and documentation. https://cwiki.apache.org/MAHOUT/canopy-clustering.html

Mike Bowles – Machine Learning on Big Data using Map Reduce Winter, 2012

K-Means Clustering

-K-means algorithm seeks to partition a data set into K disjoint sets, such that the sum of the within set variances is minimized.

-Using Euclidean distance, within set variance is the sum of squared distances from the set's centroid

-Lloyd's algorithm for K-means goes as follows:

Initialize:

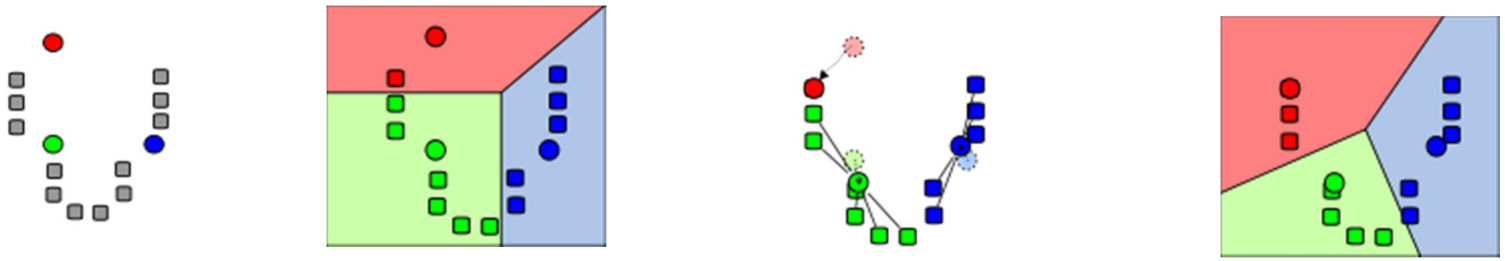
Pick K starting guesses for centroids (at random)

Iterate:

Assign points to cluster whose centroid is closest

Calculate cluster centroids

-Here's a sequence from Wikipedia page on k-means clustering http://en.wikipedia.org/wiki/K-means_clustering



K-means in Map Reduce

Mapper – given K initial guesses, run through local data and for each point, determine which centroid is closest, accumulate vector sum of points closest to each centroid, combiner emits <$centroid_i$, (sum_i , n_i)> for each of the i centroids.

Reducer – for each old $centroid_i$, aggregate sum and n from all mappers and calculate new centroid.

This map-reduce pair completes an iteration of Lloyd's algorithm.

Support Vector Machine - SVM

-For classification, SVM finds separating hyperplane

-H_3 does not separate classes. H_1 separates, but not with max margin. H_2 separates with max margin

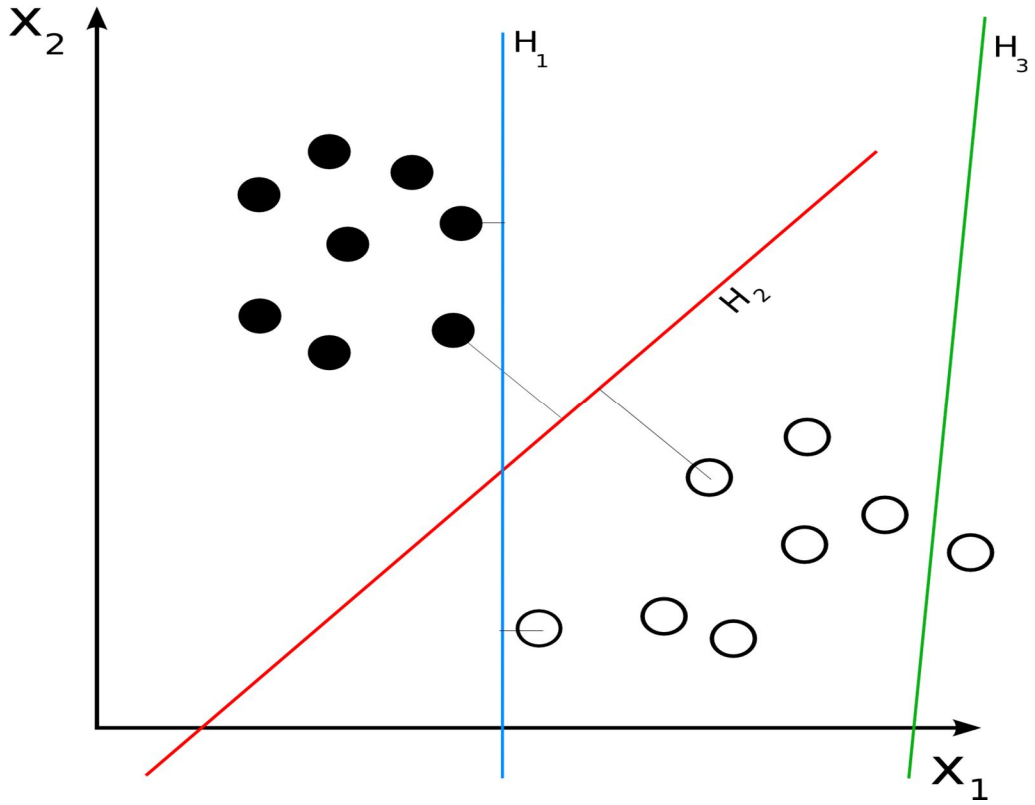


Figure from http://en.wikipedia.org/wiki/Support_vector_machine

Mike Bowles – Machine Learning on Big Data using Map Reduce Winter, 2012

SVM as Mathematical Optimization Problem

-Given a training set S = $\{(x_i, y_i)\}_{i=1}^{m}$

-Where $x_i \in R^n$ and $y_i \in \{+1, -1\}$

-We can write any linear functional of x as $w^T x + b$, where $w \in R^n$ and b is scalar
Find weight vector w and constant b to minimize

$$min_{w,b} \frac{1}{2} ||w||^2 + \frac{C}{m} \sum_{(x,y)\in S} l(w, b; (x, y))$$

Where

$l(w,b;(x,y)) = \max\{0, 1 - y(w^T x + b)\}$

Solution by Batch Gradient Descent

-We can solve this by using batch gradient descent. Take a derivative wrt w & b.

Notice that if the 1 - y ($w^T x + b$) < 0, then l() = 0 and $\nabla_{w,b}\, l(\quad) = 0$.

Denote by $S^+ = \{(x,y) \in S \mid 1 - y\,(w^T x + b) > 0\}$. Then the gradient wrt w is

$$w + \frac{C}{m} \sum_{(x,y) \in S^+} -yx$$

and wrt b is:

$$\frac{C}{m} \sum_{(x,y) \in S^+} -y$$

For reference see "Map Reduce for Machine Learning on Multicore" mentioned earlier. Also see Shalev-Shwartz, "Pegasos: Primal Estimated sub-Gradient Solver for SVM"

Calculating a Gradient Step

-What's important about what we just developed? Several things

1. In the equation we wound up with terms like $(w + \frac{C}{m}\sum_{(x,y)\in S^+} -yx)$, only the term inside the sum is data dependent.
2. The data dependent terms sum(-yx) is summed over the points in the input data where the constraints are active.

-Let's summarize by drawing up the mapper and reducer functions.

Initialize w and b, C and m (# of instances).

Iterate

Mapper – Each mapper has access to a subset S_m of S. For points $(x,y) \in S_m$ check $1 - y(w^T x + b) > 0$. If yes, accumulate $-y$ and $-yx$. Emit <,(∑-y, ∑ -yx)>. We'll put in a dummy key value, but all the output gets summarized in a single (vector) quantity to be processed by a single reducer

Reducer – Accumulate ∑-y and ∑ -yx and update estimate of w and b using

$w_{new} = w_{old} - \epsilon (w_{old} + \frac{C}{m} \sum\text{-}yx)$

and

$b_{new} = b_{old} - \epsilon (\frac{C}{m} \sum\text{-}y)$

GLMNet Algorithm

-Regularized Regression

-To avoid over-fitting have to exert control over degrees of freedom in regression

-cut back on attributes – subset selection

-penalize regression coefficients – coefficient shrinkage, ridge regression, lasso regression

-With Coefficient Shrinkage, different penalties give solutions with different properties

"Regularization Paths for Generalized Linear Models, via Coordinate Descent" Friedman, Hastie and Tibshirani
http://www.stanford.edu/~hastie/Papers/glmnet.pdf

Regularizing Regression with Coefficient Shrinkage

-Start with OLS problem formulation and add a penalty to OLS error

-Suppose

$y \in R$ and vector of predictors $x \in R^n$

-As with OLS, seek a fit for y of the form

$y \approx \beta_0 + x^T\beta$

-Assume that x_i have been standardized to mean zero and unit variance

-Find

$$min_{(\beta_0,\beta)} \left[\frac{1}{2N} \sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \pi P_\alpha(\beta)\right]$$

-The part in the sum is the ordinary least squares penalty. The bit at the end ($\pi P_\alpha(\beta)$) is new.

-Notice that the minimizing set of β's is a function of the parameter π. If π = 0 we get OLS coefficients.

Penalty Term

-The coefficient penalty term is given by

$$P_\alpha(\beta) = \sum \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\right]$$

-This is called "elasticnet" penalty (see ref below).

- $\alpha = 1$ gives l_1 penalty (sum of absolute values)
- $\alpha = 0$ gives l_2 – squared penalty (sum of squares)

-Why is this important? The choice of penalty influences that nature of the solutions. l_1 gives sparse solutions. It ignores attributes. l_2 tends to average correlated attributes.

-Consider the coefficient paths as functions of the parameter π.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. J. Royal. Stat. Soc. B., 67(2):301{320, 2005.

Coefficient Trajectories

-Here's Figure 1 from Friedman et. al. paper

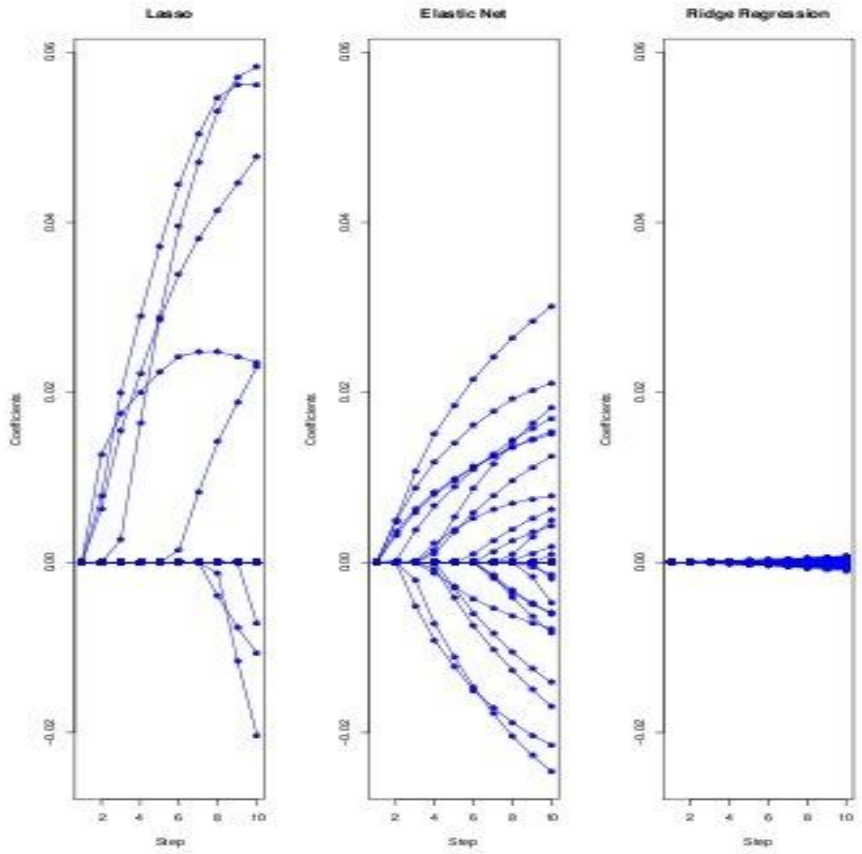


Figure 1: *Leukemia data: profiles of estimated coefficients for three methods, showing only first 10 steps (values for λ) in each case. For the elastic net, $\alpha = 0.2$.*

Pathwise Solution

-This algorithm works as follows:

Initialize with value of π large enough that all β's are 0.

Decrease π slightly and update β's by taking a gradient step from the old β based on new value of π.

Small changes in π means that the update converges quickly.

In many cases faster to generate entire coefficient trajectory with this algo, than to generate point solution

-Friedman et. al. show that the elements of the coefficient vector β_j satisfies

$$\beta_j = \frac{S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - y_i^{(j)}\right),\ \pi\alpha\right)}{1 + \pi(1-\alpha)}$$

where the function S() is given by

S(z,γ) = z – γ if z > 0 and γ < |z|

= z + γ if z < 0 and γ < |z|

= 0 if γ >= |z|

-The point is:

This algorithm fits the Statistical Query Model.

The sum inside the function S() can be spread over any number of mappers

This algorithm handles elasticnet regularized regressions, logistic regression and multiclass logistic regression

Summary

-Here's what we covered

1. Where do big-data machine learning problems arise?
   -e-commerce, retail, narrow targeting, bank and credit cards

2. What ways are there to deal with big-data machine learning problems?
   -sample, language level parallelization, map-reduce

3. What is map-reduce?
   -programming formalism that isolates programming task to mapper and reducer functions

3. Application of map-reduce to some familiar machine learning algorithms

Hope you learned something from the class. To get into more detail, come to the big-data class.