Strata+ Hadoop WORLD

PRESENTED BY

O'REILLY cloudera

strataconf.com

#StrataHadoop

Building Machine Learning Apps with Spark

Vartika / Jayant

Download & Install

- http://spark.apache.org/downloads.html

- spark-shell --driver-memory 2G --executor-memory 2G --executor-cores 2

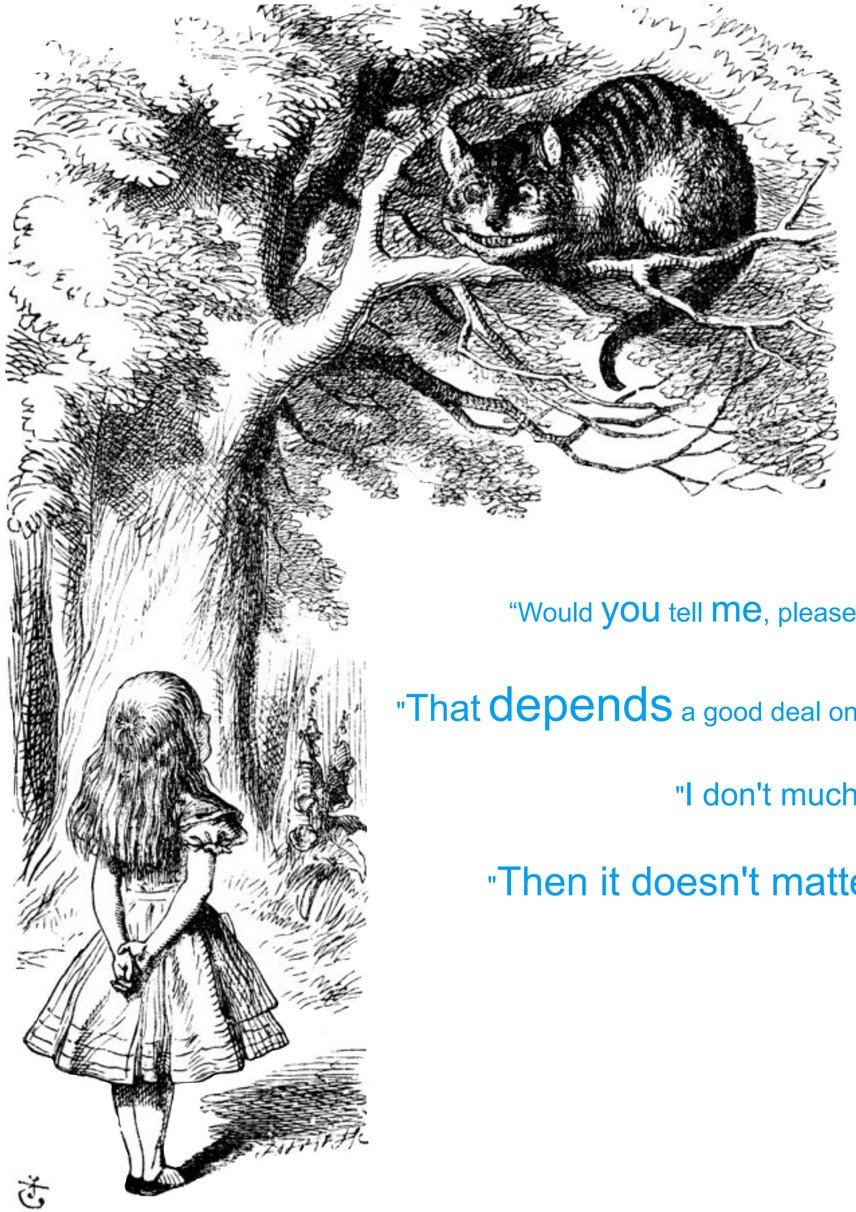
https://github.com/WhiteFangBuck/strata-2016-ny

About Us

Vartika Singh is a Solutions Architect at Cloudera with over 10 years of experience in applying machine learning techniques to big data problems.

Jayant is the CEO of Sparkflows.io with over 10 years of experience building big data products and applying machine learning.

Prev : Cloudera / Yahoo / eBay



"Would you tell me, please, which road do I take?"

"That depends a good deal on where you want to get to."

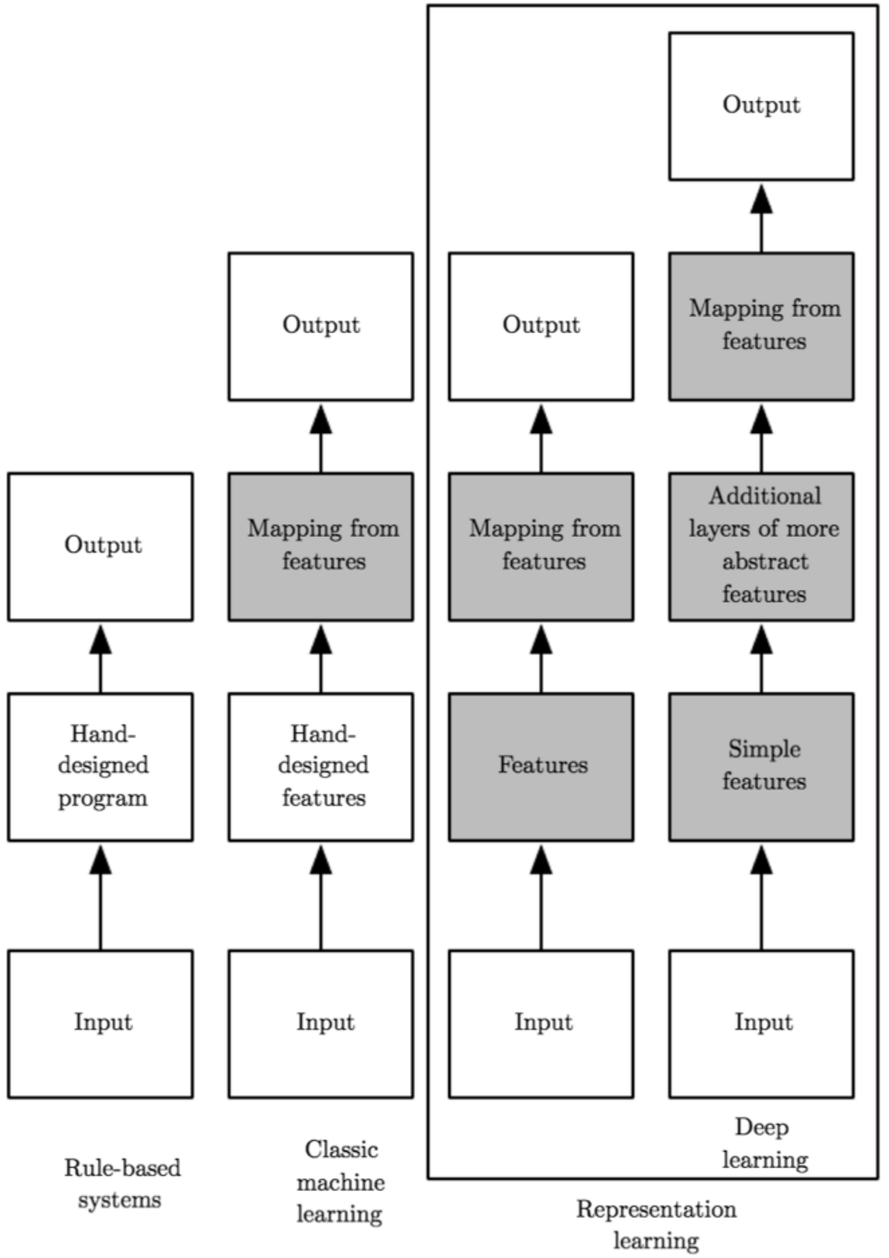
"I don't much care where –"

"Then it doesn't matter which way you go."

A Data Scientists Nightmare

Which Algorithm???!!!!

- Class of algorithms
  - Supervised
  - Unsupervised
- Examples
  - Supervised: Classification, Regression
  - Unsupervised: Clustering



CDH Stack - Typical Machine Learning Workflow

- Data Ingest Load
- Feature Extraction
- Training/Tuning the model
- Prediction/Inference

- Split each document's text into words.
- Convert each document's words into a numerical feature vector.
- Learn a prediction model using the feature vectors and labels.

ML Pipelines and Workflow

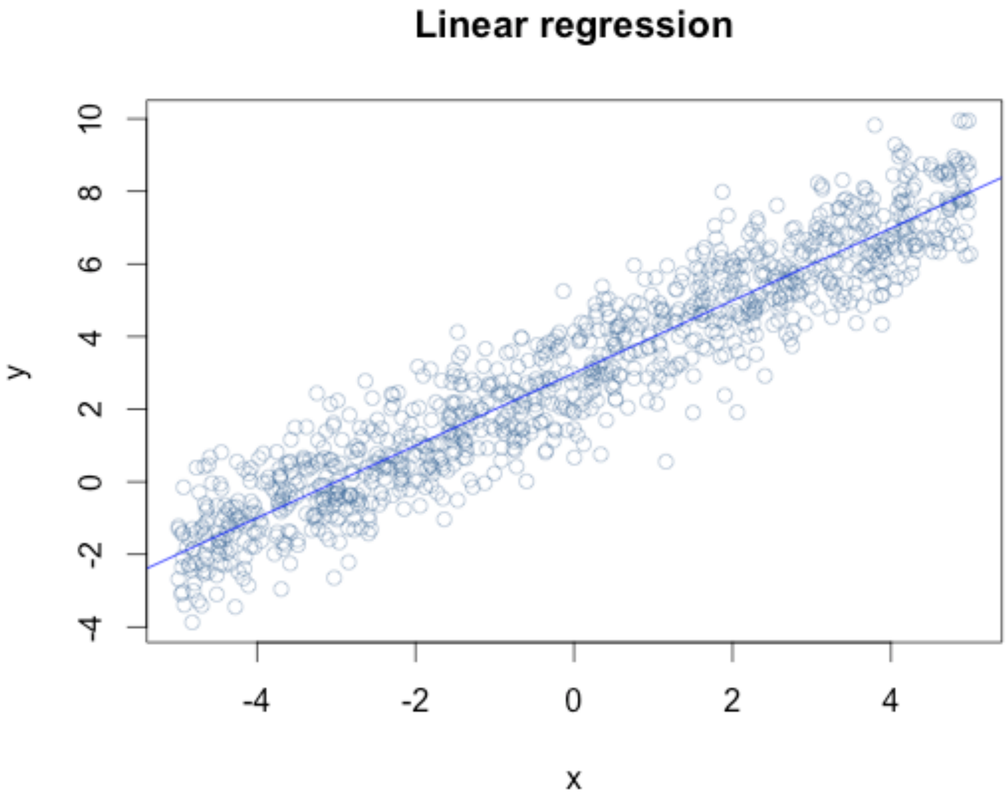
- Facilitates a quick and easy assembly and configuration of practical machine learning pipelines.
- Are like DAG of nodes – sequence of stages – Estimators and Transformers.
- Can be saved and loaded when needed.
- Hyperparameter Tuning
- Flexible coding and Easy debugging – Use DataFrames

Regression - Linear Methods

- Modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. => $y\hat{} = w` x$, where $w \in R^{(n)}$, is a vector of parameters.

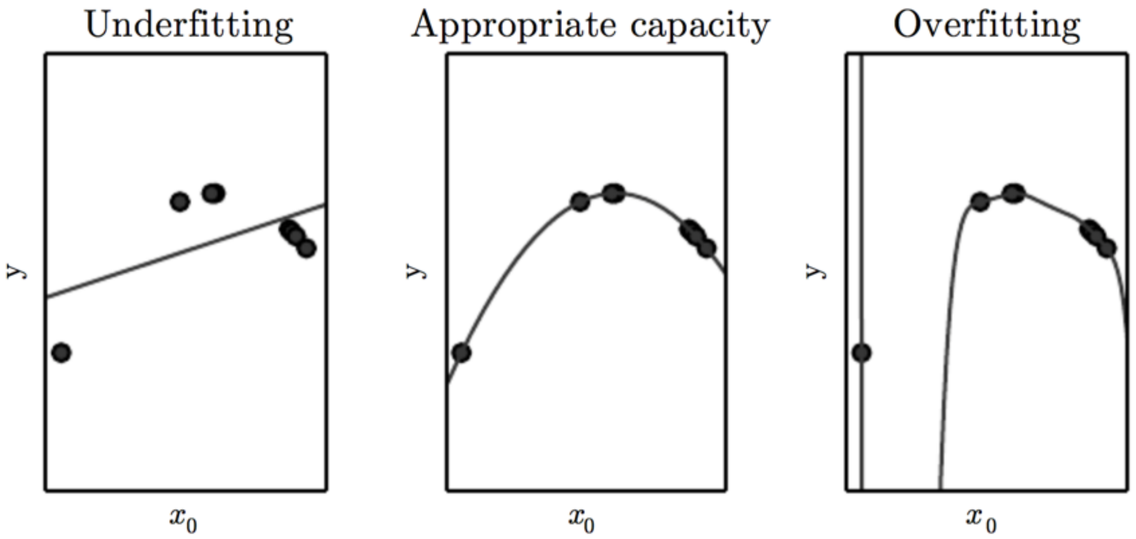
- The relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.

- Parameters are values that control the behavior of the system.



Capacity and Hypothesis Space

- Models with insufficient capacity are unable to solve complex tasks. Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task they may overfit.
  - Linear - Underfitting
  - Degree-9 - Overfitting
  - Quadratic - Appropriate
- Representational Capacity
- Effective Capacity
- Occam's razor (c. 1287-1347). This principle states that among competing hypotheses that explain known observations equally well, one should choose the "simplest" one.



Working with the data

- Categorical Variables

id	price	lot size	bedrooms	bathrooms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea
1	42000	5850	3	1	2	yes	no	yes	no	no	1	no
2	38500	4000	2	1	1	yes	no	no	no	no	0	no
..	..	..	..	..	..	..	..	..	..	..	..	..

```
val categoricalVariables = Array(
        "driveway","recroom", "fullbase", "gashw", "airco", "prefarea"
)
```

drivewayIndex	recroomIndex	fullbaseIndex	gashwIndex	aircoIndex	prefareaIndex
0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
..	..	..	..	..	..

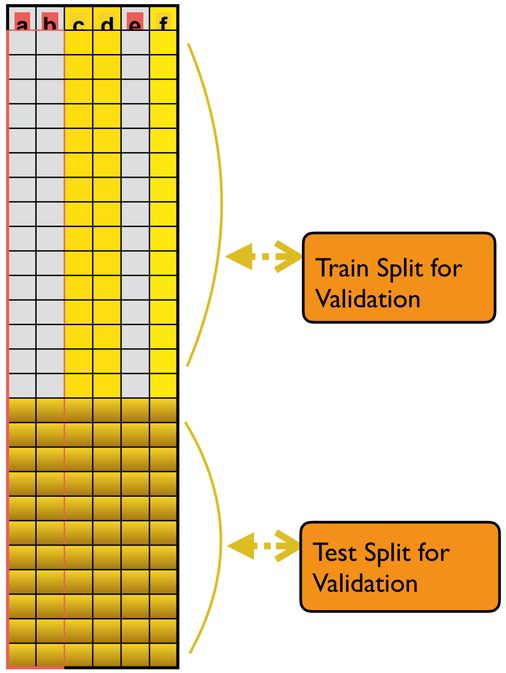
drivewayVec	recroomVec	fullbaseVec	gashwVec	aircoVec	prefareaVec
(1,[0],[1.0])	(1,[0],[1.0])	(1,[],[])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])
(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])
..	..	..	..	..	..

org.apache.spark.ml.feature.StringIndexer

org.apache.spark.ml.feature.OneHotEncoder

org.apache.spark.ml.feature.VectorAssembler

Model Selection and Tuning



- Use data to find the best model or parameters for a given task.
- Can be done for single estimator or an entire pipeline.
- Require
  - Estimater -> algorithm to tune
  - ParamMap -> set of parameters to tune over
  - Evaluator -> metric to measure how well a fitted Model does on *holdout* data
- Can be done via CrossValidation or TrainValidation Split
  - CrossValidation
    - Goes over multiple folds of data.
    - Slower but more reliable
  - TrainValidation split
    - Evaluate combination of parameters only once.
    - Faster but less reliable

CrossValidation Split
- Split Data into folds
- Fits the estimator on all the folds
- After identifying the best parammap, it re-fits the estimator using the best param map

Linear Regression - Train and Predict

```
case class X(
        id: String ,price: Double, lotsize: Double,
        bedrooms: Double, bathrms: Double,stories: Double,
        driveway: String,recroom: String,fullbase: String,
        gashw: String, airco: String, garagepl: Double, prefarea: String)
```

The case class to map the data in to

```
val paramGrid = new ParamGridBuilder()
        .addGrid(lr.regParam, Array(0.1, 0.01, 0.001))
        .addGrid(lr.fitIntercept)
        .addGrid(lr.elasticNetParam, Array(0.0, 1.0))
```

Construct the Parameter Grid

```
val tvs = new TrainValidationSplit()
        .setEstimator( pipeline )
        .setEvaluator( new RegressionEvaluator()
                .setLabelCol("price") )
        .setEstimatorParamMaps(paramGrid)
        .setTrainRatio(0.75)
```

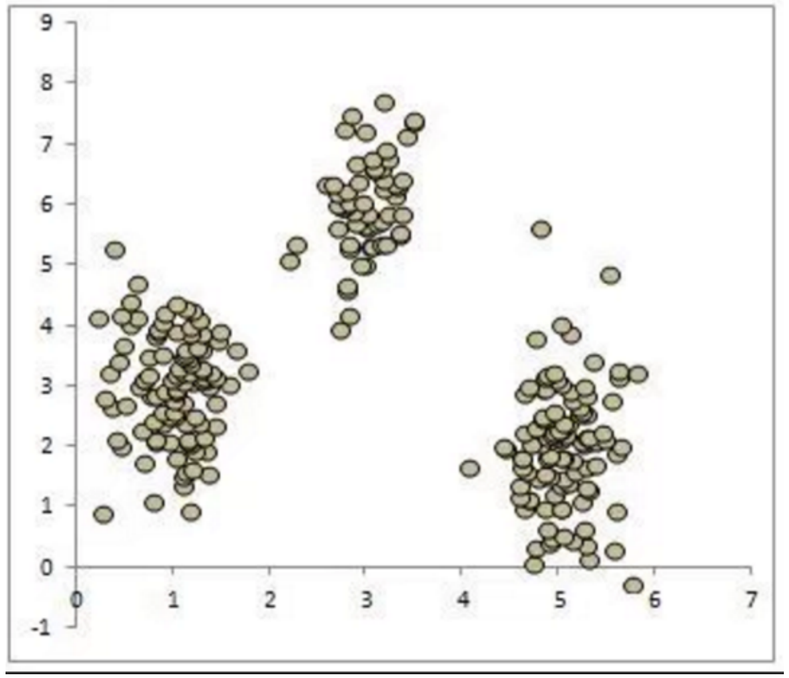
Form the Validator

```
val Array(training, test) = data.randomSplit(Array(0.75, 0.25), seed = 12345)

val model = tvs.fit(training)
```

Split and train the data

Clustering



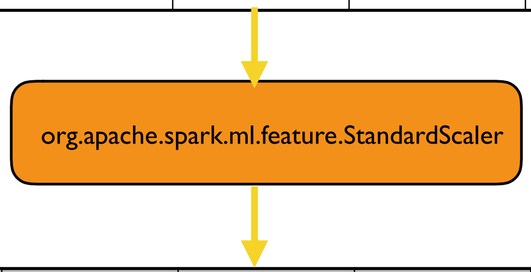
- A good clustering has predictive power.
- Predictions while uncertain, are useful, because we believe that the underlying cluster labels are meaningful and can help us take meaningful actions.
- Failures of the cluster model may highlight interesting objects that deserve special attention, a.k.a outliers.
- Dimensionality reduction.
- Compression

KMeans - Reading the data and scaling

```
val ds = spark.read.option("inferSchema", "true").option("header", "true").option("nullValue", "?").csv("data/mtcars.csv")
```

Infer the schema from the header

_c0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.9	2.62	16.46	0	1	4	4
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.25	17.98	0	0	3	4
..	..	..	..	..	..	..	..	..	..	..	..



_c0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	0.150884824647656 15	..	-0.570619818667 9038	..	..	..	..	..	..	..	-0.610399567481 5357
Cadillac Fleetwood	-1.607882615918 843	..	.9467538146582	..	..	..	..	..	..	…	2.077504764835 6484
..	..	..	..	..	..	..	..	..	..	..	..

Looking at the results

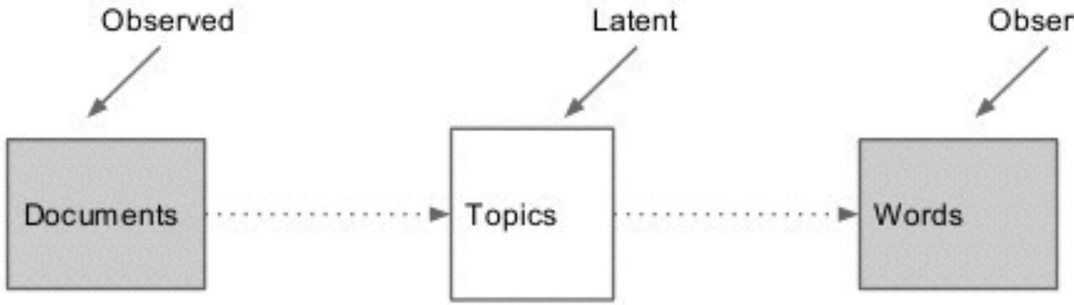
Cluster 1	
Duster 360,[14.3,8.0,360.0,245.0,3.21,3.57],9]	
[Merc 450SE,[16.4,8.0,275.8,180.0,3.07,4.07],9]	
[Merc 450SL,[17.3,8.0,275.8,180.0,3.07,3.73],9]	
[Merc 450SLC,[15.2,8.0,275.8,180.0,3.07,3.78],9]	
[Dodge Challenger,[15.5,8.0,318.0,150.0,2.76,3.52],9]	
[AMC Javelin,[15.2,8.0,304.0,150.0,3.15,3.435],9]	

Cluster 2	
[Datsun 710,[22.8,4.0,108.0,93.0,3.85,2.32],3]	
[Merc 240D,[24.4,4.0,146.7,62.0,3.69,3.19],3]	
[Merc 230,[22.8,4.0,140.8,95.0,3.92,3.15],3]	
[Toyota Corona,[21.5,4.0,120.1,97.0,3.7,2.465],3]	
[Volvo 142E,[21.4,4.0,121.0,109.0,4.11,2.78],3]	

Cluster 3	
[Cadillac Fleetwood,[10.4,8.0,472.0,205.0,2.93,5.25],5]	
[Lincoln Continental,[10.4,8.0,460.0,215.0,3.0,5.424],5]	
[Chrysler Imperial,[14.7,8.0,440.0,230.0,3.23,5.345],5]	

Clustering - LDA

Goal of Topic Modeling



Documents are about several topics at the same time.
Topics are associated with different words.

Topics in the documents are expressed through the words that are used.

Get to know the data - sample email

Re: Short-fuse NSC Washfax

BW60M;10/R,PART
B5,B6 qqey<

I approve sendaresponse! OfLM!,, Cheryl D <CD9.OXCc: Sullivan, Jacob J <il67>~ 15:56:09~FWv~8FYI - See below!jDKoh, Harold Hongju5:F"071:50I#%rBTMacmanus, Joseph E; F4; Syed, Zia S;>T; SPberg!4mes B; Amory, Elizabeth R; Ashraf, MadeehaMDonoghueanTownley,Whphen G; Johnson, Clifton M;R%@pector, Phillip M! ~ Since ourUc(is due bac"thi@by Tues, IL sto clear)4brief proposedTb%xhe

q@4received earlix<his weekend fromZ$Ops ctr. aMax2xVcc
MKoh<he Legal Adviser^e ite 64202201 C St. NW-xingtA[DC 20520-,@02 647 9598 offic!mobil 7096 H
V@Macris, Gregory Pai)V;v5u
VF-; iMxES-O_SWO-Only; SES_DutyDeputies!aent: Sat $5 17:01:41nColleageL

Aua#$attached sq5()yWh!Ho<Situation Room a C time agoa>(Per instruc)sIExecSec S staff,e{isQwarŏaa copy
e#
is;JbyaTassist in
any way necryQ(I8)f(ior Watch OEEk&kZ]
  2

*2*4*04:0 Nf
 Y Blackb^

Some house-keeping

- Get the word distribution
- Clean up the data
  - DataFrame => UDF => Word Distribution again to get a new stop word list

```
val cleanData = udf ({ text: String => text.replaceAll("[^A-Za-z ]", " ")})
```

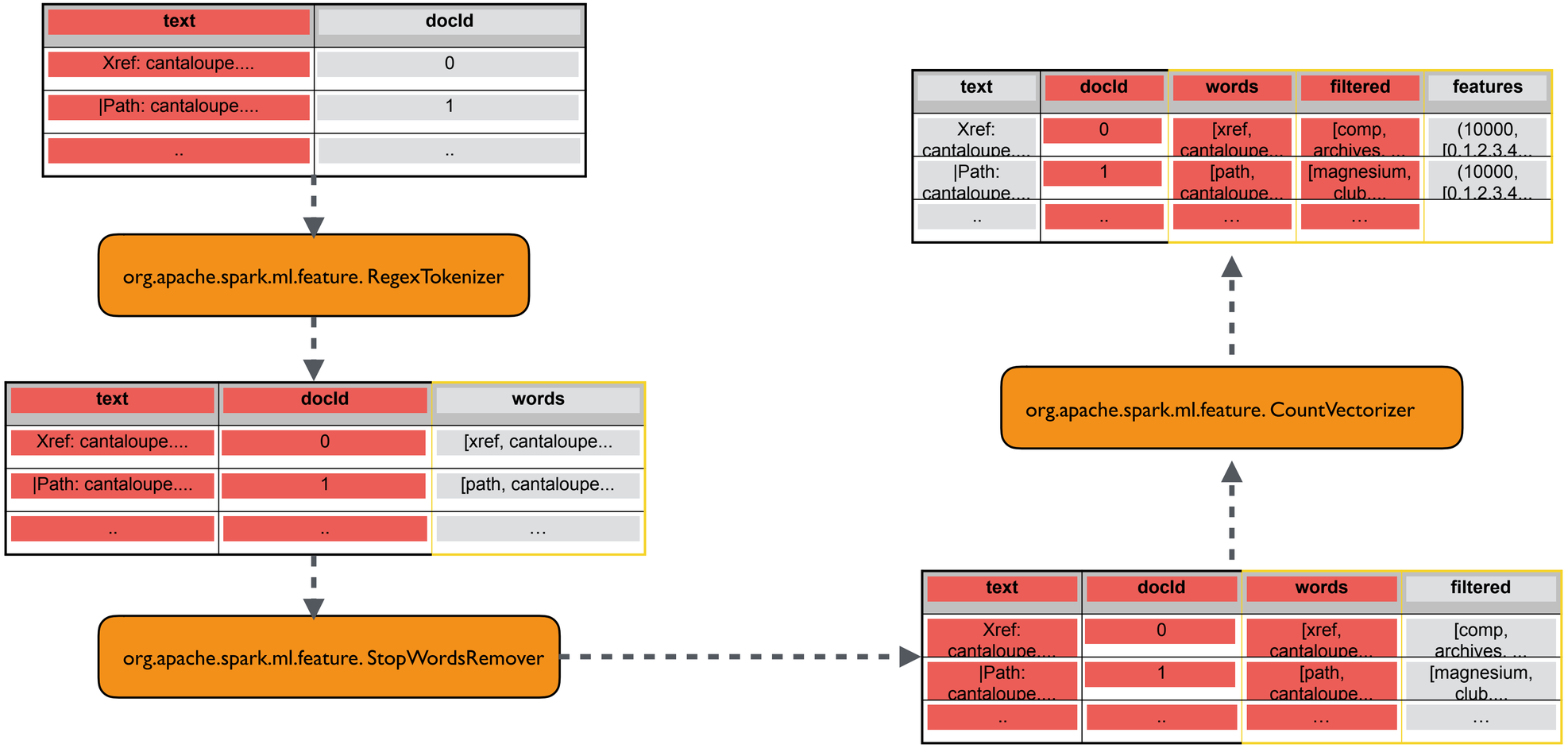
- Splitting
- Stemming

TOPIC 4	
saudi king rest slipped	0.18592574
reuters saudi king rest	0.18592574
king rest slipped disc	0.18592574
mcdonough denis mcdonough denis	0.15367810
denis mcdonough denis mcdonough	0.11988294
crowley philip reines philippe	0.03178703
reines philippe crowley philip	0.00381955
hanley monica hanleymr jake	0.00111982
decided iftar tonite unga	0.00109838

TOPIC 9	
benghazi agreement benghazi agreement	0.45488224
agreement benghazi agreement benghazi	0.41133721
hanley monica hanleymr jake	0.00112136
decided iftar tonite unga	0.00109987
hanley monica hanleymr left	0.00106407
samuelson heather toiv nora	0.00104709
pascual carlos mexico city	0.00102610
jeffrey feltman feltman jeffrey	0.00100359
laszczych joanne toiv nora	0.00099235
lona valmoro direct valmoro	0.00098503

Middle
East
Dominates

Preparing the data



Looking at the results

Topic 1	
space	0.012660059181
nasa	0.005330283688
program	0.005239824606
available	0.005144048115
system	0.004966346014
data	0.004792352055

Topic 2	
colorado	0.02660943268
guns	0.01395920562
ucsu	0.01308461546
udel	0.01298423116
firearms	0.01224593594
intercon	0.01082974947

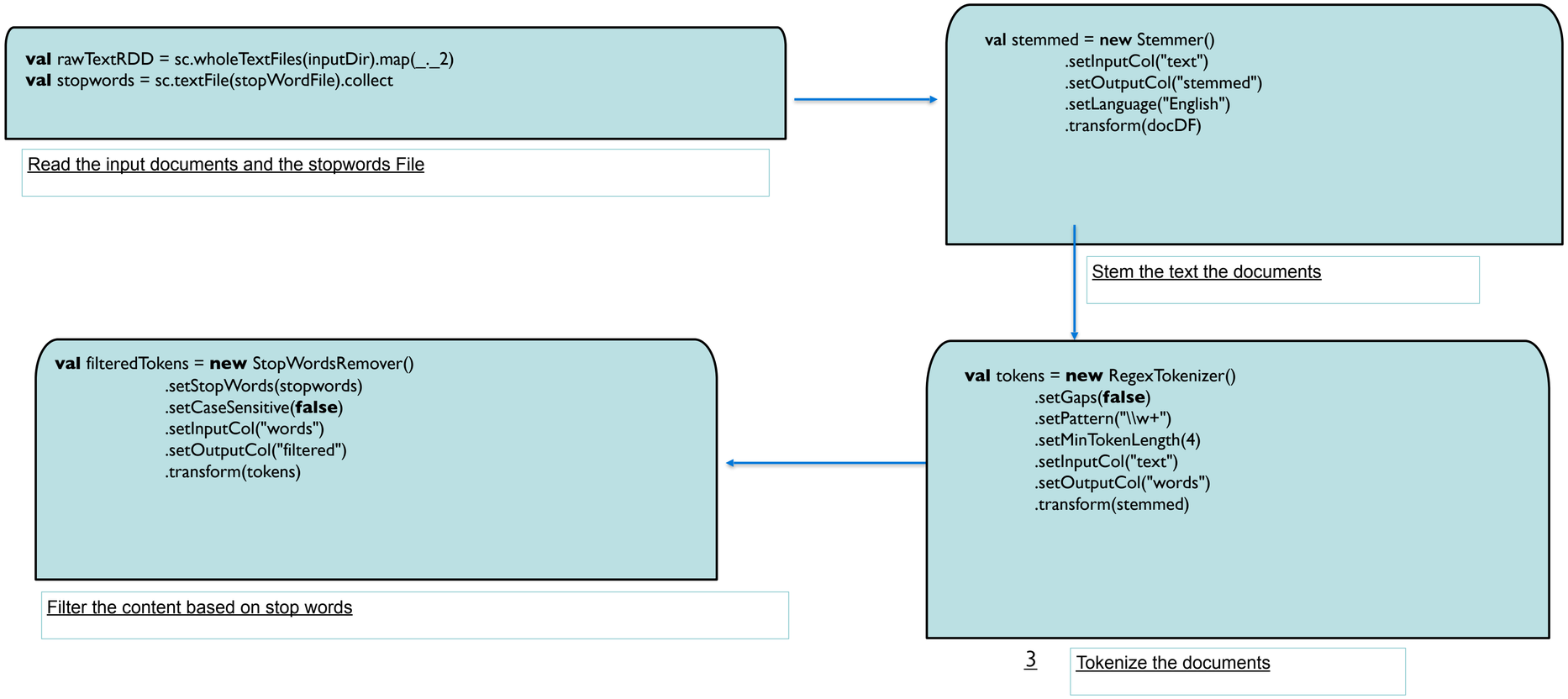
Topic 3	
people	0.00774559688
turkish	0.00763194932
israel	0.00673282049
mideast	0.00664783679
jewish	0.00590779415
jews	0.00563967327

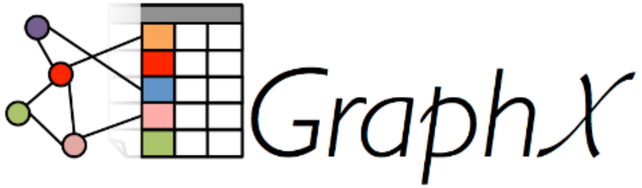
Topic 4	
news	0.020640614429
baseball	0.019538283096
sport	0.012547926648
game	0.010113930332
subject	0.009844881506
organization	0.009577969868

Topic 5	
duke	0.013943643237
team	0.009899371553
hockey	0.008075055082
sport	0.008021120554
news	0.007865084086
ulowell	0.007315961791
league	0.007098242372
baseball	0.006199597687

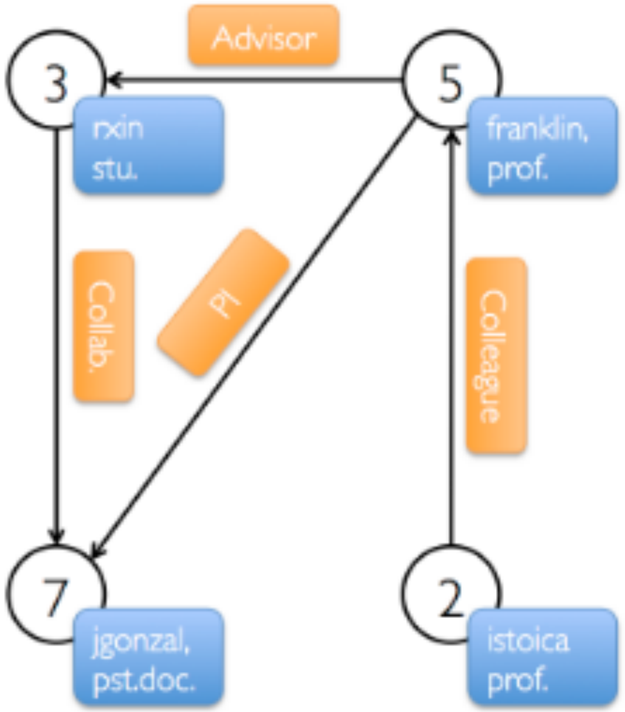
Topic 6	
rutgers	0.025988077707
christian	0.017065131308
religion	0.013231107212
writes	0.009754402317
talk	0.009659720709
lines	0.009513981142

Topic Modeling - Code Walk-through





Property Graph



Vertex Table

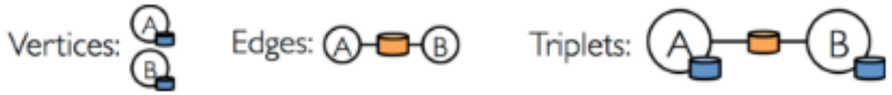
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

EdgeTriplet

The EdgeTriplet class extends the Edge class by adding the srcAttr and dstAttr members which contain the source and destination properties respectively. We can use the triplet view of a graph to render a collection of strings describing relationships between users.



List of Graph Operators

```scala
/** Summary of the functionality in the property graph */
class Graph[VD, ED] {
  // Information about the Graph ===================================================================
  val numEdges: Long
  val numVertices: Long
  val inDegrees: VertexRDD[Int]
  val outDegrees: VertexRDD[Int]
  val degrees: VertexRDD[Int]
  // Views of the graph as collections =============================================================
  val vertices: VertexRDD[VD]
  val edges: EdgeRDD[ED]
  val triplets: RDD[EdgeTriplet[VD, ED]]
  // Functions for caching graphs ==================================================================
  def persist(newLevel: StorageLevel = StorageLevel.MEMORY_ONLY): Graph[VD, ED]
  def cache(): Graph[VD, ED]
  def unpersistVertices(blocking: Boolean = true): Graph[VD, ED]
  // Change the partitioning heuristic  ============================================================
  def partitionBy(partitionStrategy: PartitionStrategy): Graph[VD, ED]
  // Transform vertex and edge attributes ==========================================================
  def mapVertices[VD2](map: (VertexID, VD) => VD2): Graph[VD2, ED]
  def mapEdges[ED2](map: Edge[ED] => ED2): Graph[VD, ED2]
  def mapEdges[ED2](map: (PartitionID, Iterator[Edge[ED]]) => Iterator[ED2]): Graph[VD, ED2]
  def mapTriplets[ED2](map: EdgeTriplet[VD, ED] => ED2): Graph[VD, ED2]
  def mapTriplets[ED2](map: (PartitionID, Iterator[EdgeTriplet[VD, ED]]) => Iterator[ED2])
    : Graph[VD, ED2]
  // Modify the graph structure ====================================================================
  def reverse: Graph[VD, ED]
  def subgraph(
      epred: EdgeTriplet[VD,ED] => Boolean = (x => true),
      vpred: (VertexID, VD) => Boolean = ((v, d) => true))
    : Graph[VD, ED]
  def mask[VD2, ED2](other: Graph[VD2, ED2]): Graph[VD, ED]
  def groupEdges(merge: (ED, ED) => ED): Graph[VD, ED]
  // Join RDDs with the graph ======================================================================
  def joinVertices[U](table: RDD[(VertexID, U)])(mapFunc: (VertexID, VD, U) => VD): Graph[VD, ED]
  def outerJoinVertices[U, VD2](other: RDD[(VertexID, U)])
      (mapFunc: (VertexID, VD, Option[U]) => VD2)
    : Graph[VD2, ED]
  // Aggregate information about adjacent triplets =================================================
  def collectNeighborIds(edgeDirection: EdgeDirection): VertexRDD[Array[VertexID]]
  def collectNeighbors(edgeDirection: EdgeDirection): VertexRDD[Array[(VertexID, VD)]]
  def aggregateMessages[Msg: ClassTag](
      sendMsg: EdgeContext[VD, ED, Msg] => Unit,
      mergeMsg: (Msg, Msg) => Msg,
      tripletFields: TripletFields = TripletFields.All)
    : VertexRDD[A]
  // Iterative graph-parallel computation ==========================================================
  def pregel[A](initialMsg: A, maxIterations: Int, activeDirection: EdgeDirection)(
      vprog: (VertexID, VD, A) => VD,
      sendMsg: EdgeTriplet[VD, ED] => Iterator[(VertexID,A)],
      mergeMsg: (A, A) => A)
    : Graph[VD, ED]
  // Basic graph algorithms ========================================================================
  def pageRank(tol: Double, resetProb: Double = 0.15): Graph[Double, Double]
  def connectedComponents(): Graph[VertexID, ED]
  def triangleCount(): Graph[Int, ED]
  def stronglyConnectedComponents(numIter: Int): Graph[VertexID, ED]
}
```

Pregel takes two argument lists (i.e., graph.pregel(list1)(list2))

- The first argument list contains configuration parameters including
  - **the initial message**
  - **the maximum number of iterations**
  - **the edge direction in which to send messages (by default along out edges).**
- The second argument list contains the **user defined functions for**
  - **receiving messages (the vertex program vprog)**
  - **computing messages (sendMsg)**
  - **and combining messages mergeMsg.**

Pregel - Code Walk-through

```
val graph: Graph[Long, Double] =
  GraphGenerators.logNormalGraph(sc, numVertices = 100).mapEdges(e =>
e.attr.toDouble)
```

Create Graph

```
val sourceId: VertexId = 42 // The ultimate source
// Initialize the graph such that all vertices except the root
have distance infinity.
val initialGraph = graph.mapVertices((id, _) =>
    if (id == sourceId) 0.0 else Double.PositiveInfinity)
```

Initialize the graph

```
val sssp = initialGraph.pregel(Double.PositiveInfinity)(
  (id, dist, newDist) => math.min(dist, newDist), //
Vertex Program
  triplet => {  // Send Message
    if (triplet.srcAttr + triplet.attr <
triplet.dstAttr) {
      Iterator((triplet.dstId, triplet.srcAttr +
triplet.attr))
    } else {
      Iterator.empty
    }
  },
  (a, b) => math.min(a, b) // Merge Message
)
```

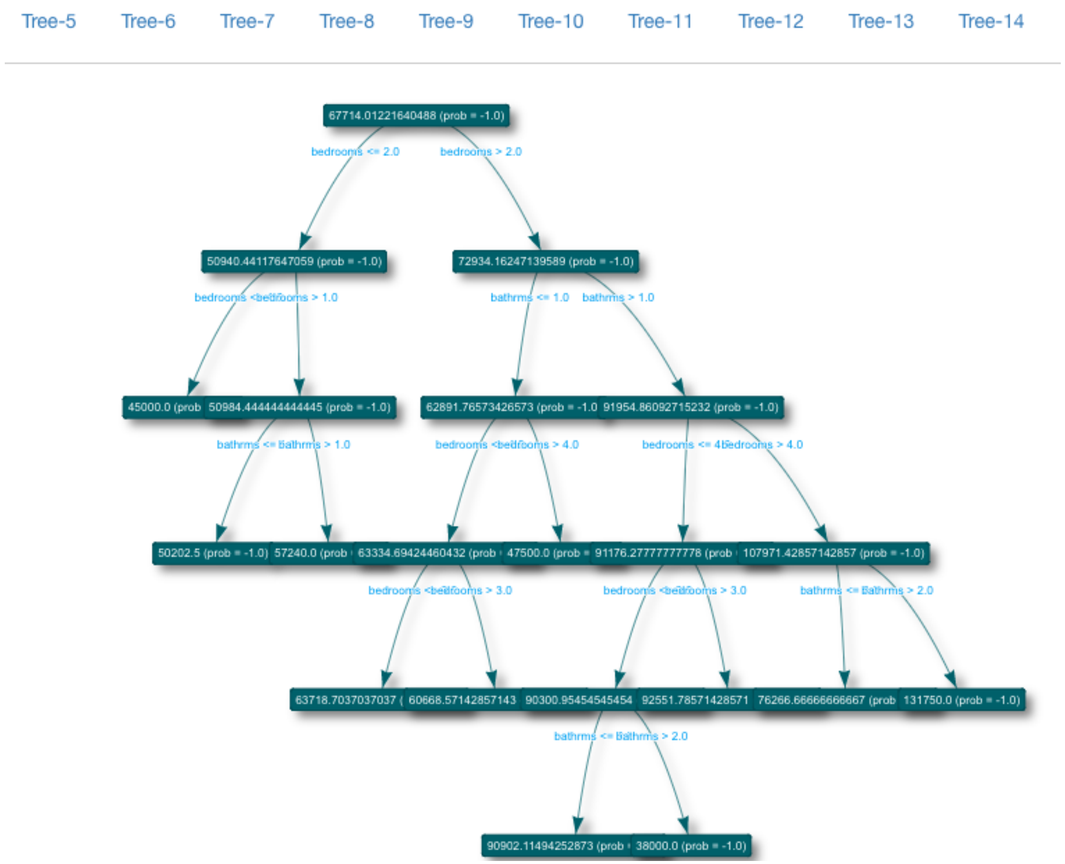
Pregel

```
println(sssp.vertices.collect.mkString("\n"))
```

Print Vertices and min distance

Classification - Random Forest

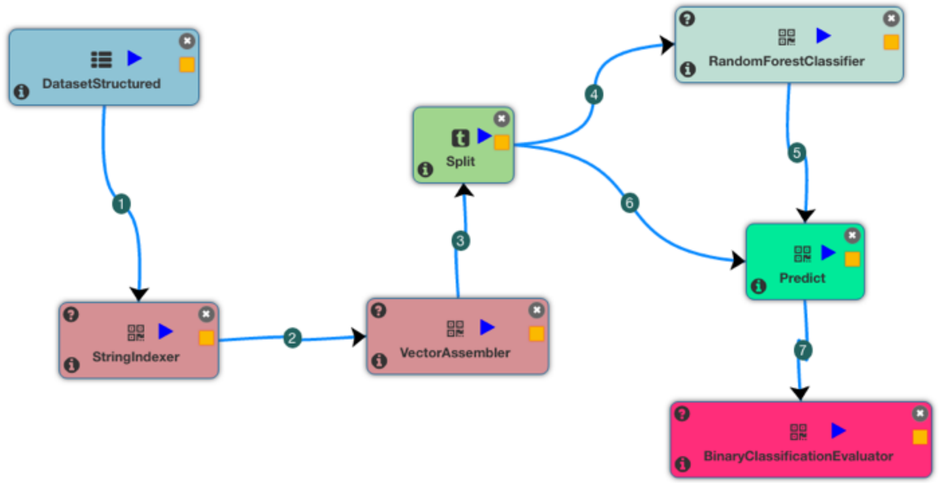
- Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

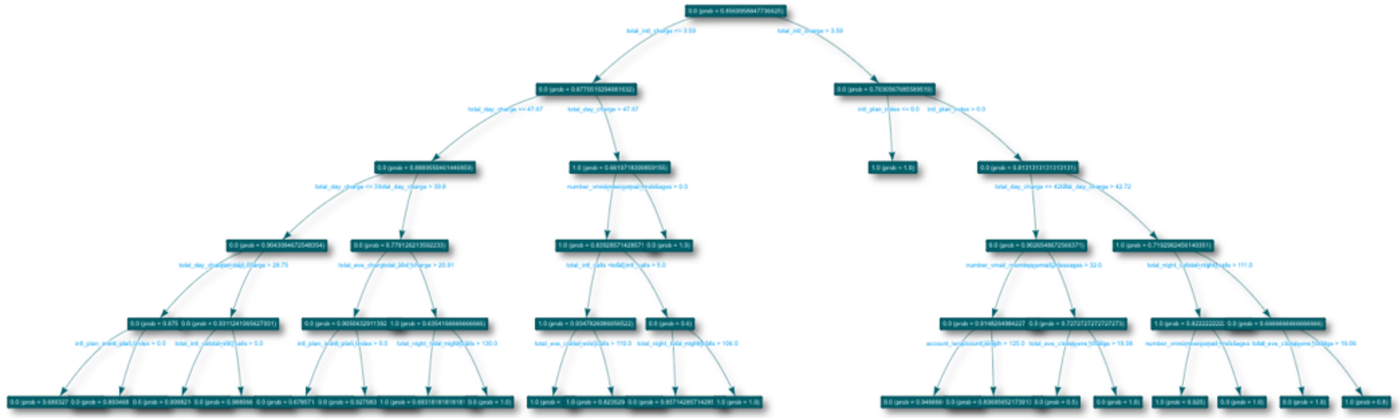


Classification - Churn Prediction for Telco

state	account_length	area_code	phone_number	intl_plan	voice_mail_plan	number_vmail_messages
StringType	DoubleType	DoubleType	StringType	StringType	StringType	DoubleType
OH	107.0	415.0	371-7191	no	yes	26.0
NJ	137.0	415.0	358-1921	no	no	0.0
OH	84.0	408.0	375-9999	yes	no	0.0
OK	75.0	415.0	330-6626	yes	no	0.0
AL	118.0	510.0	391-8027	yes	no	0.0
MA	121.0	510.0	355-9993	no	yes	24.0

total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_eve_charge
DoubleType	DoubleType	DoubleType	DoubleType	DoubleType	DoubleType
161.6	123.0	27.47	195.5	103.0	16.62
243.4	114.0	41.38	121.2	110.0	10.3
299.4	71.0	50.9	61.9	88.0	5.26
166.7	113.0	28.34	148.3	122.0	12.61
223.4	98.0	37.98	220.6	101.0	18.75
218.2	88.0	37.09	348.5	108.0	29.62





Random Forest - Code Walk-through

```
val ds = spark.read.option("inferSchema",
"true").schema(customSchema).csv("data/churn.all")
```

Read the input documents

```
val indexer = new StringIndexer()
      .setInputCol("intl_plan")
      .setOutputCol("intl_plan_idx")
val indexed = indexer.fit(ds).transform(ds)
```

Index the intl_plan column

```
// vector assembler
val assembler = new VectorAssembler()
  .setInputCols(Array("account_length",
"intl_plan_idx", "number_vmail_messages",
"total_day_minutes", "total_day_calls"))
  .setOutputCol("features")

val assemdata = assembler.transform(churned)
```

Assemble the features

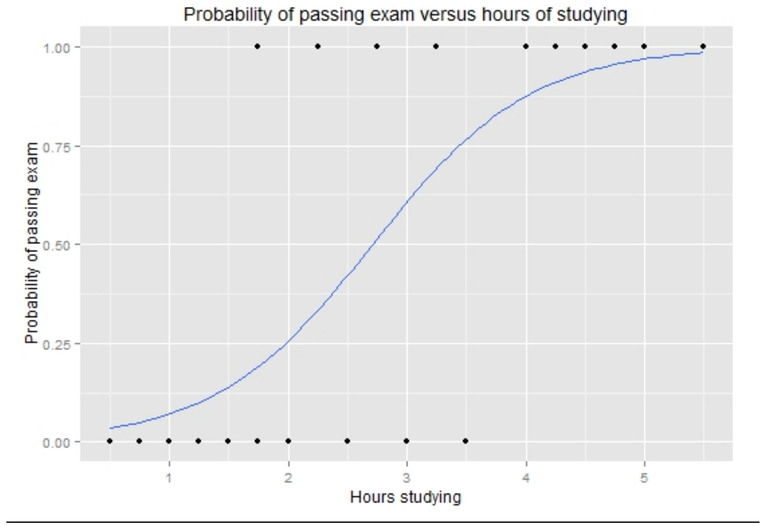
```
// Train a RandomForest model.
val rf = new RandomForestClassifier()
  .setLabelCol("churned_idx")
  .setFeaturesCol("features")
  .setNumTrees(10)

// Fit the model
val rfModel = rf.fit(trainingData)
```

Train the Model

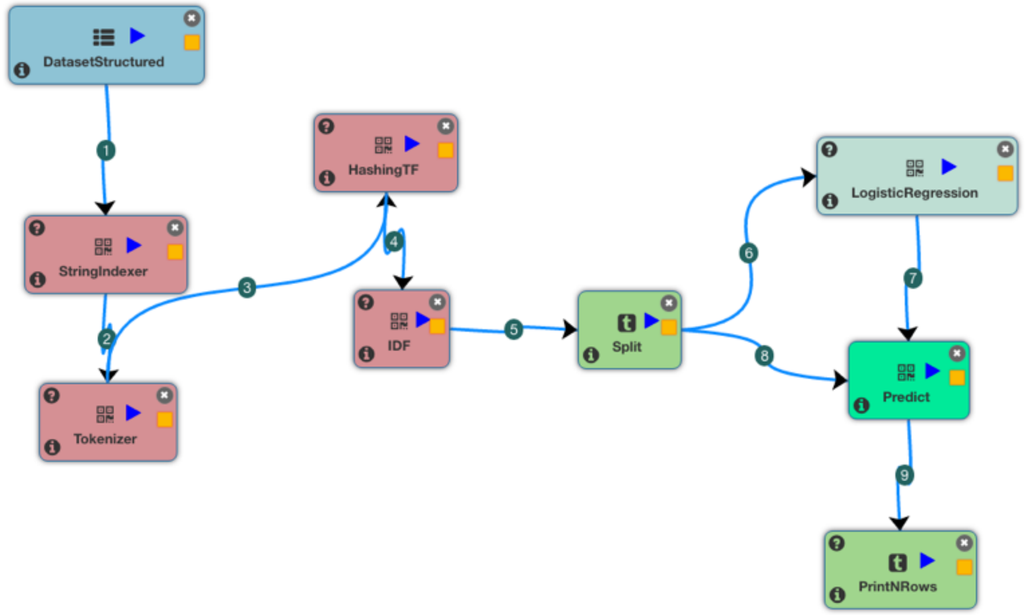
Classification - Logistic Regression

- Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.



Spam Detection

spam	message
StringType	StringType
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam	WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.



LogisticRegression

```
Label Column : spam_idx Intercept : -13.549728579531946 Coefficients : [-1.5268677627599565,-0.2372794241775128,-1.1072674016618658,-0.718543651988402,-1.2060471817854415,-0.5121405670718475,
```

Logistic Regression - Code Walk-through

```
val customSchema = StructType(Array(
      StructField("spam", StringType, true),
      StructField("message", StringType, true)
   ))
val ds = spark.read.option("inferSchema", "true").option("delimiter",
"\t").schema(customSchema).csv("data/SMSSpamCollection.tsv")
```

Read the input documents

```
// tokenize
val tokenizer = new Tokenizer().setInputCol("message").setOutputCol("tokens")
val tokdata = tokenizer.transform(indexed)

// tf
val hashingTF = new HashingTF()
  .setInputCol("tokens").setOutputCol("tf")//.setNumFeatures(20)
val tfdata = hashingTF.transform(tokdata)

// idf
val idf = new IDF().setInputCol("tf").setOutputCol("idf")
val idfModel = idf.fit(tfdata)
val idfdata = idfModel.transform(tfdata)
```

Tokenize/TF/IDF the message column

```
val assembler = new VectorAssembler()
  .setInputCols(Array("idf"))
  .setOutputCol("features")

val assemdata = assembler.transform(idfdata)

// split
val Array(trainingData, testData) =
assemdata.randomSplit(Array(0.7, 0.3), 1000)
```

VectorAssembler & Split

```
val lr = new LogisticRegression()
  .setLabelCol("label")
  .setFeaturesCol("features")


// Fit the model
val lrModel = lr.fit(trainingData)

// predict
val predict = lrModel.transform(testData)

predict.show(100)

val evaluator = new BinaryClassificationEvaluator()
  //.setLabelCol("indexedLabel")
  .setRawPredictionCol("prediction")
  .setMetricName("precision")

val accuracy = evaluator.evaluate(predict)
```

Train, Predict, Evaluate the model

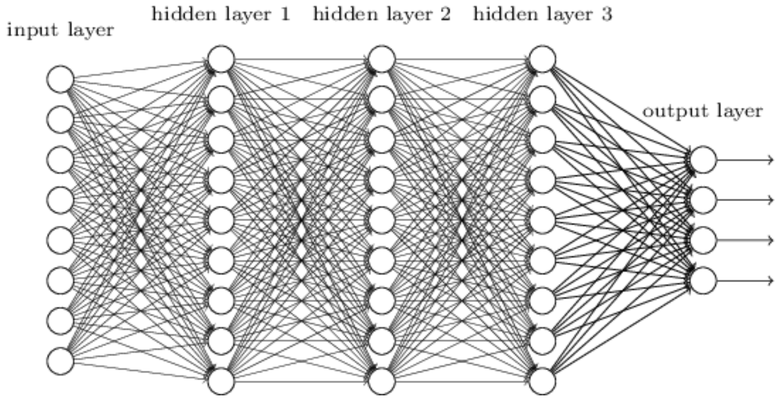
Spam Detection

LogisticRegression

```
Label Column : spam_idx Intercept : -13.549728579531946 Coefficients : [-1.5268677627599565,-0.2372794241775128,-1.1072674016618658,-0.718543651988402,-1.2060471817854415,-0.5121405670718475,
```

spam	message	spam_idx	ner	tf	idf	rawPrediction	probability	prediction
StringType	StringType	DoubleType	ArrayType(StringType,true)	org.apache.spark.mllib.linalg.VectorUDT@f71b0bce	org.apache.spark.mllib.linalg.VectorUDT@f71b0	org.apache.spark.mllib.linalg.VectorUDT@f71b0bce	org.apache.spark.mllib.linalg.VectorUDT@f71b0bce	DoubleType
ham	Ok lar... Joking wif u oni...	0.0	WrappedArray(ok, lar..., joking, wif, u, oni...)	(1000,[117,401,508,548,596,716],[1.0,1.0,1.0,1.0,1.0,1.0])	(1000,[117,401,508,548,596,716],[1.880013366	[28.976524354438865,-28.976524354438865]	[0.9999999999997395,2.6040862723062254E-13]	0.0
ham	U dun say so early hor... U c already then say...	0.0	WrappedArray(u, dun, say, so, early, hor..., u, c, already, then, say...)	(1000,[91,99,117,371,643,676,837,872,941,995],[1.0,1.0,2.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])	(1000,[91,99,117,371,643,676,837,872,941,995]	[20.939194751756823,-20.939194751756823]	[0.9999999991942075,8.057925791066503E-10]	0.0
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune	0.0	WrappedArray(as, per, your, request, 'melle, melle, (oru, minnaminunginte, nurungu, vettam)', has, been, set, as, your, callertune, for, all, callers., press, *9, to, copy, your, friends, callertune)	(1000,[63,66,122,123,267,329,359,434,573,577,673,685,707,762,798,806,820,877,917,943,955,962],[1.0,1.0,2.0,3.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,2.0,1.0])	(1000,[63,66,122,123,267,329,359,434,573,577,[3.371799124996704,3.8037909874382967,5.6	[18.539452471207944,-18.539452471207944]	[0.9999999911198952,8.880104783324248E-9]	0.0
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030	1.0	WrappedArray(had, your, mobile, 11, months, or, more?, u, r, entitled, to, update, to, the, latest, colour, mobiles, with, camera, for, free!, call, the, mobile, update, co, free, on, 08002986030)	(1000,[47,51,101,114,117,123,180,324,329,338,468,496,501,541,551,555,568,577,707,734,737,801,819,866,982],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,2.0,1.0,2.0,2.0,1.0,2.0,1.0])	(1000,[47,51,101,114,117,123,180,324,329,338,[4.5297279908212325,3.4592865791198193,3.:	[-33.55460470089267,33.55460470089267]	[2.6755947551760234E-15,0.9999999999999973]	1.0

Deep Learning



Deep Learning - Exploring Trends

- Thus far finding the algorithmic basis for process of creating visual experiences through complex interplay between the content and style of an image, as done by humans, is unknown and there exists no artificial system with similar capabilities.
- However, in other key areas of visual perception such as object and face recognition near-human performance by DNN.
- Path forward to an algorithmic understanding of how humans create and perceive artistic imagery.

Reference: http://arxiv.org/pdf/1508.06576v1.pdf

Finer Points

- A major source of difficulty in many real-world AI applications is that many of the factors of variation influence every single piece of data we are able to observe.
  - Example of car: Color in night, viewing angle

- It can be very difficult to extract such high-level, abstract features from raw data.

- When is it nearly as difficult to obtain a representation as to solve the original problem, representation learning does not, at first glance seem to help us.

- Deep Learning, solves this central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations .

- Deep learning allows the computer to build complex concepts out of simpler concepts.

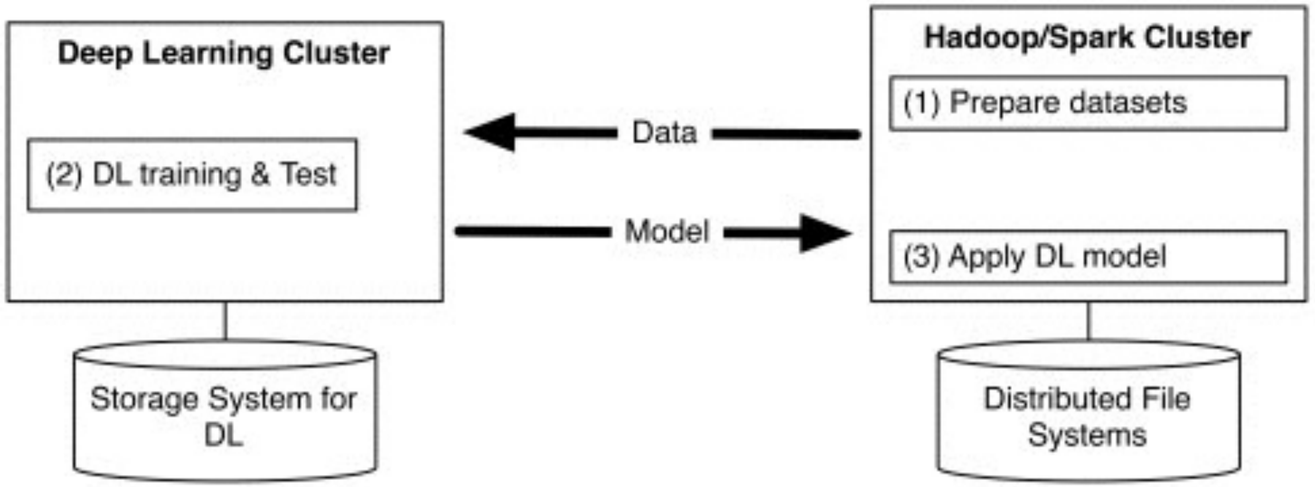
CPU and GPU - Best of both worlds!

- CaffeOnSpark
- DeepLearning4j

- GPU is Optimized for taking huge batches of data and performing the same operation over and over
- GPUs are special purpose and can compute vector maths, matrix maths, pixel transforms and rendering jobs about 10-100x faster than the equivalent CPU

- Architecturally, the CPU is composed of just few cores with lots of cache memory that can handle a few software threads at a time.
- They excel in serial tasks, branching operations and file operations.

Typical Scenario



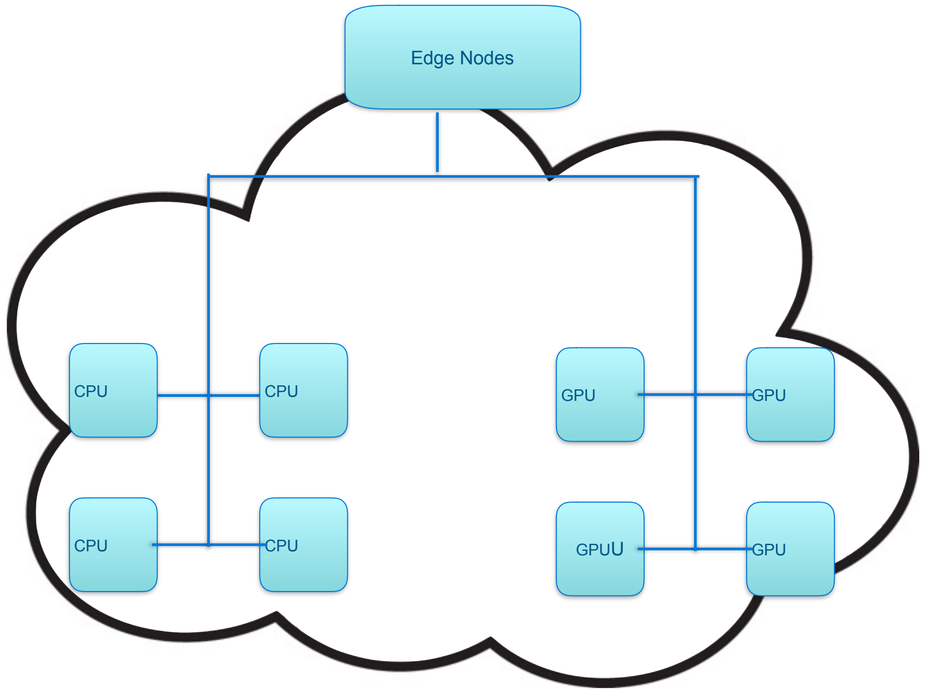
- Large data transfers
- Complicated pipelines

Intro to Caffe

- Developed by Berkley Vision and Learning Center
- Caffe defines a net layer-by-layer in its own model schema.
- As data and derivatives flow through the network in the forward and backward passes Caffe stores, communicates, and manipulates the information as **blobs**:
  - the blob is the standard array and unified memory interface for the framework.
- The layer comes next as the foundation of both model and computation.
  - The net follows as the collection and connection of layers.
  - The details of blob describe how information is stored and communicated in and across layers and nets.

- We will go over the details of these components in more detail.

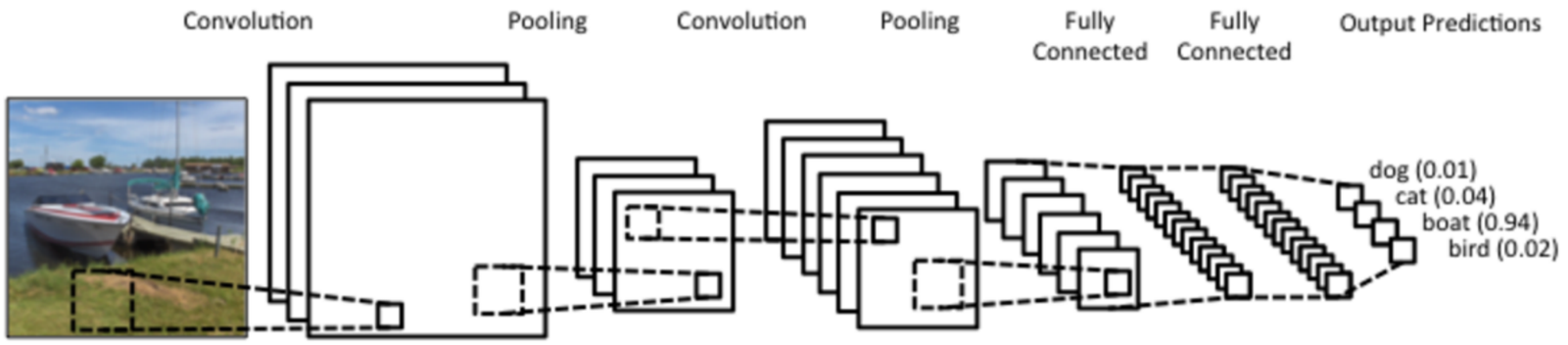
CaffeOnSpark



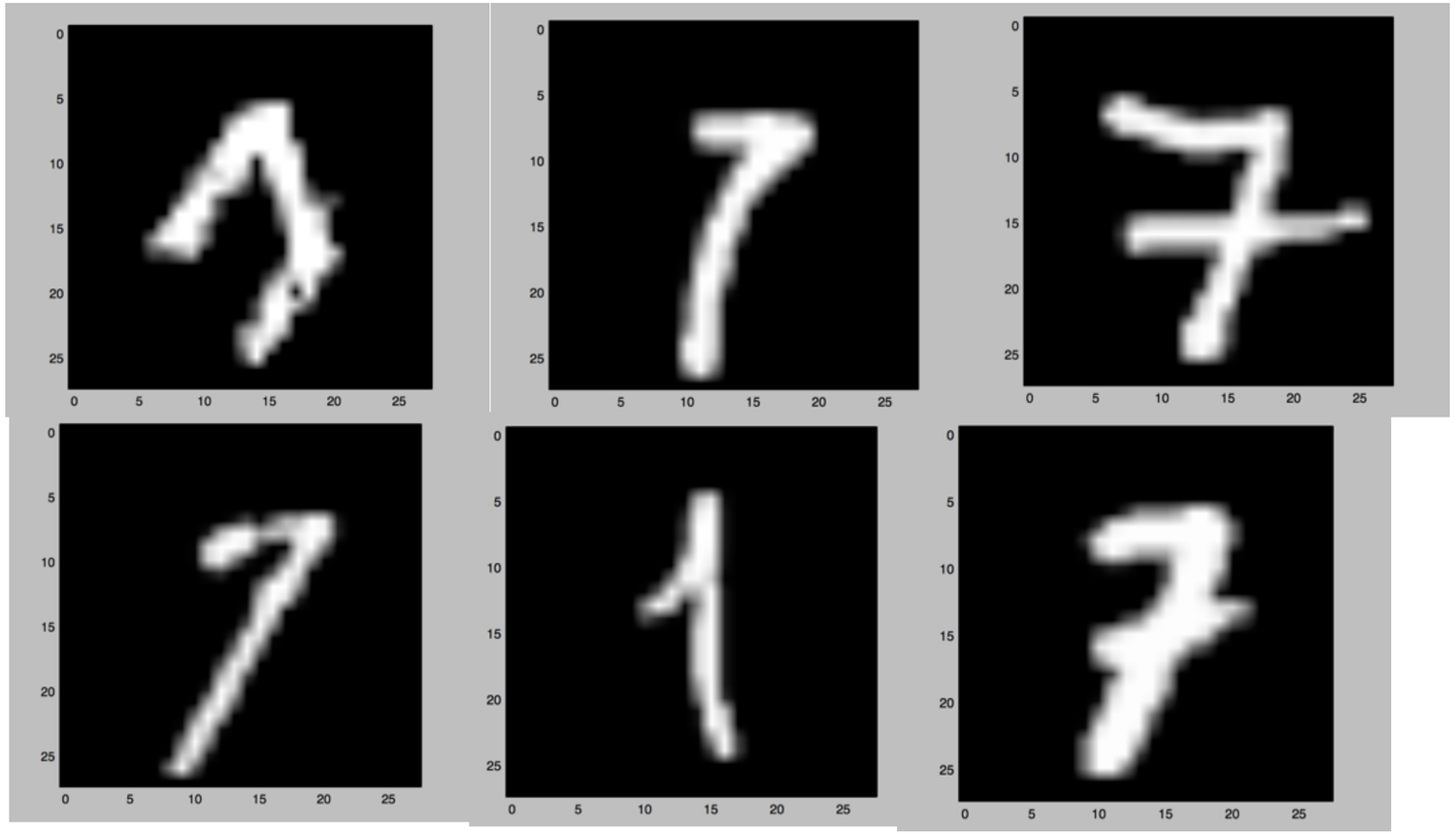
- Launch Caffe engines on GPU devices or CPU devices within the Spark executor.
- Invokes a JNI layer with fine-grain memory management.
- Unlike traditional Spark applications, CaffeOnSpark executors communicate to each other via MPI allreduce style interface via TCP/Ethernet or RDMA/ Infiniband.
- This Spark+MPI architecture enables CaffeOnSpark to achieve similar performance as dedicated deep learning clusters.
- Checkpointing: CaffeOnSpark enables training state being snapshotted periodically, thus we could resume from previous state after a failure of a CaffeOnSpark job.

Basis of it all - Conv Neural Nets

- Convolution can be thought of as a sliding window function applied to a matrix, also known as kernel/filter/feature detector.
- CNNs are basically just several layers of convolutions with nonlinear activation functions like ReLU or tanh applied to the results.
- Convolutions over the input layer is used to compute the output, which results in local connections, where each region of the input is connected to a neuron in the output.
- During the training phase, a CNN automatically learns the values of its filters based on the task you want to perform.
- Example: In Image Classification a CNN may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features.
- Locality Invariance and Composition



Getting evaluations from checkpoint data



Take aways

•Deep learning allows the computer to build complex concepts out of simpler concepts.

- Choices in CDH stack
  - SparkML
  - MLLib

- Other new exciting technologies
  - Tensor flow
  - Caffe

- Use the algorithms to unlock the value in data as oppose to lock yourself with a specific technology

- For Examples of Technology Application:
  - http://github.mtv.cloudera.com/DataScience/nlp

Thank you!!!!! :)