cloudera
Ask Bigger Questions

Unraveling Data with Spark using machine learning

Vartika /Jeff

About Us

Vartika Singh is a Solutions Architect at Cloudera with over 12 years of experience in applying machine learning techniques to big data problems.

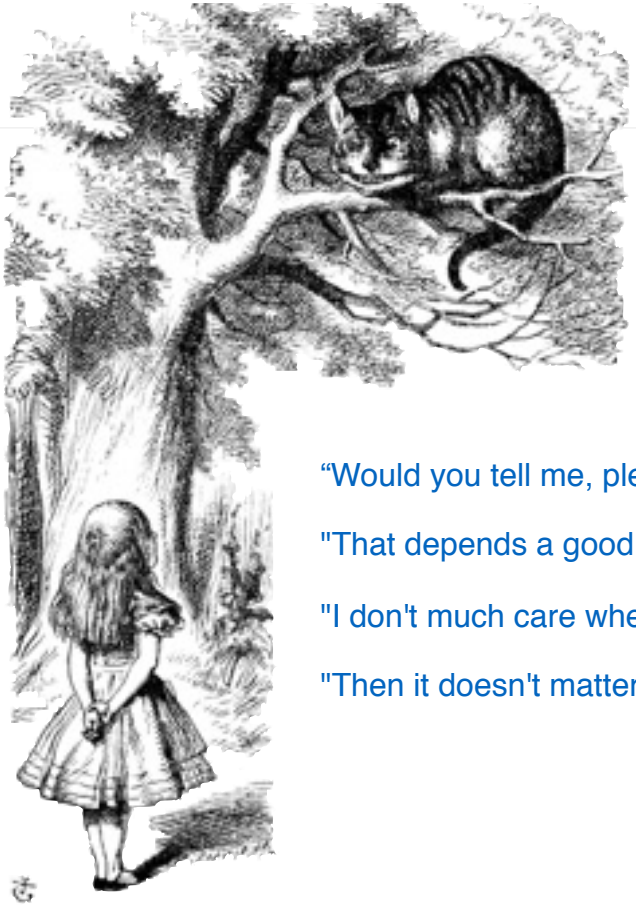
Jeff Shmain is a solution architect at Cloudera. He has 16+ years of financial industry experience with a strong understanding of security trading, risk, and regulations. Over the last few years, Jeff has been helping various clients implement spark applications.

Download & Install

Spark Shell

- http://spark.apache.org/downloads.html
- spark-shell --driver-memory 2G --executor-memory 2G --num-executors 2 --executor-cores 2

https://github.com/WhiteFangBuck/strata-london-2017



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

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

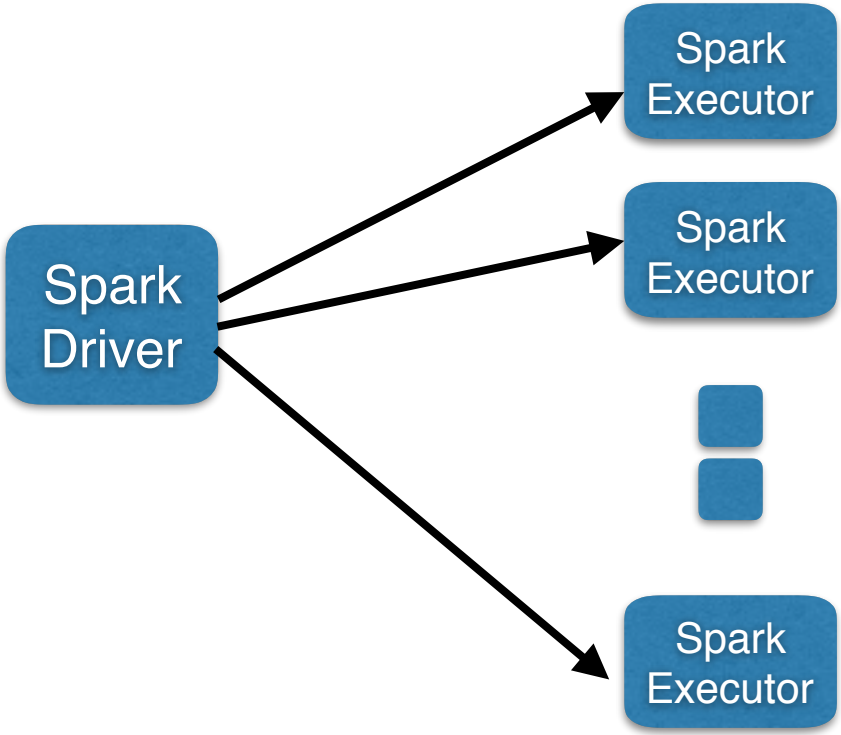
"I don't much care where –"

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

Spark and Spark ML Libraries

- There are two Spark Machine Learning libraries: ML (DataFrame based) and MLLib(RDD bases - which is in maintenance mode now)
- org.apache.spark.ml is the Scala package name
- MLlib will not add new features to the RDD-based API.
- In the Spark 2.x releases, MLlib will add features to the DataFrames-based API to reach feature parity with the RDD-based API.
- After reaching feature parity (roughly estimated for Spark 2.2), the RDD-based API will be deprecated.
- The RDD-based API is expected to be removed in Spark 3.0.

Driver and Executor



- A *program* that runs the user's main function and executes various *parallel operations* on a cluster.
- To execute jobs, Spark breaks up the processing of RDD operations into tasks, each of which is executed by an executor. Prior to execution, Spark computes the task's **closure**. The closure is those variables and methods which must be visible for the executor to perform its computations on the RDD. This closure is serialized and sent to each executor.

Spark config properties

- Spark properties control most application settings and are configured separately for each application.
- These properties can be set directly on a `SparkConf` passed to your `SparkContext`.
- You can supply the configurations doing runtime
- Common properties:
  - spark.master
  - spark.executor.memory
  - spark.executor.instances
- Properties set directly on the SparkConf take highest precedence, then flags passed to `spark-submit` or `spark-shell`, then options in the `spark-defaults.conf` file.

Spark Memory

- Execution
  - Execution memory refers to that used for computation in shuffles, joins, sorts and aggregations
- Storage
  - That used for caching and propagating internal data across the cluster
- Execution and storage share a unified region (M). When no execution memory is used, storage can acquire all the available memory and vice versa.
  - Execution may evict storage if necessary
  - Storage may not evict execution

Determining Memory Usage

- Use Cache
- SizeEstimator.estimate

Tuning Data Structure

- Prefer arrays of Objects and primitive types
- Avoid nested structures
- Numeric IDs or Enumerations

Other

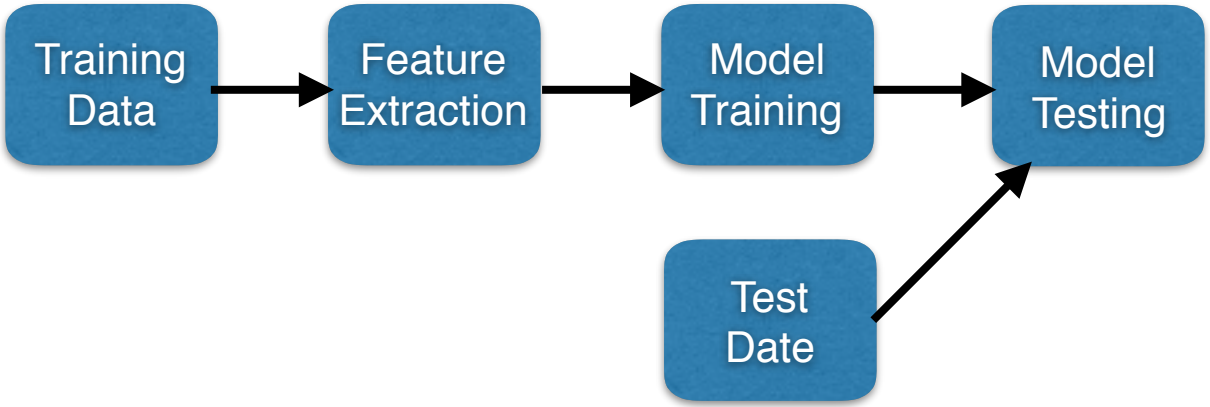
- Level of parallelism
- Memory usage of Reduce tasks
- Broadcasting large variables

cloudera
Ask Bigger Questions

ML Pipelines

ML Pipelines

- Make it easier to combine multiple algorithms into a single pipeline, or workflow.
- This ML API uses DataFrame from Spark SQL as an ML dataset, which can hold a variety of data types. E.g., a DataFrame could have different columns



Transformer

- A Transformer is an abstraction that includes feature transformers and learned models.
- Technically, a Transformer implements a method transform(), which converts one DataFrame into another, generally by appending one or more columns

Estimator

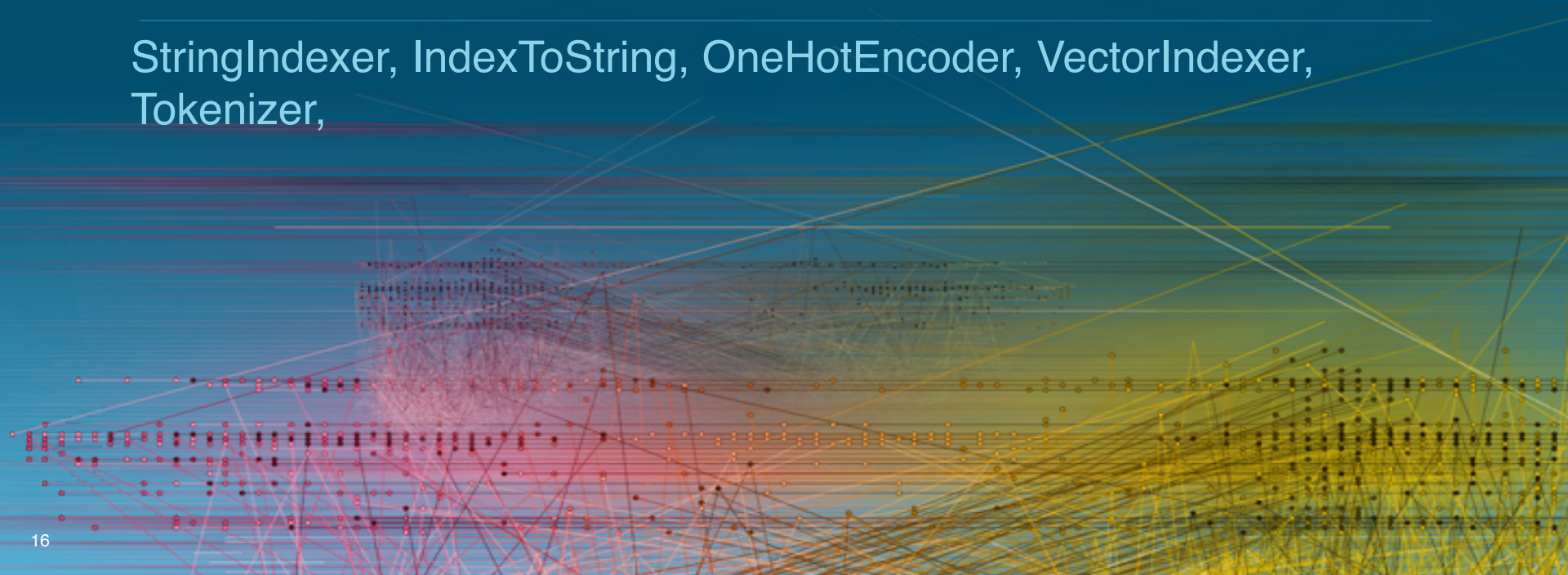
- An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data.
- Technically, an Estimator implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer.

For example, a learning algorithm such as LogisticRegression is an Estimator, and calling fit() trains a LogisticRegressionModel, which is a Model and hence a Transformer.

cloudera
Ask Bigger Questions

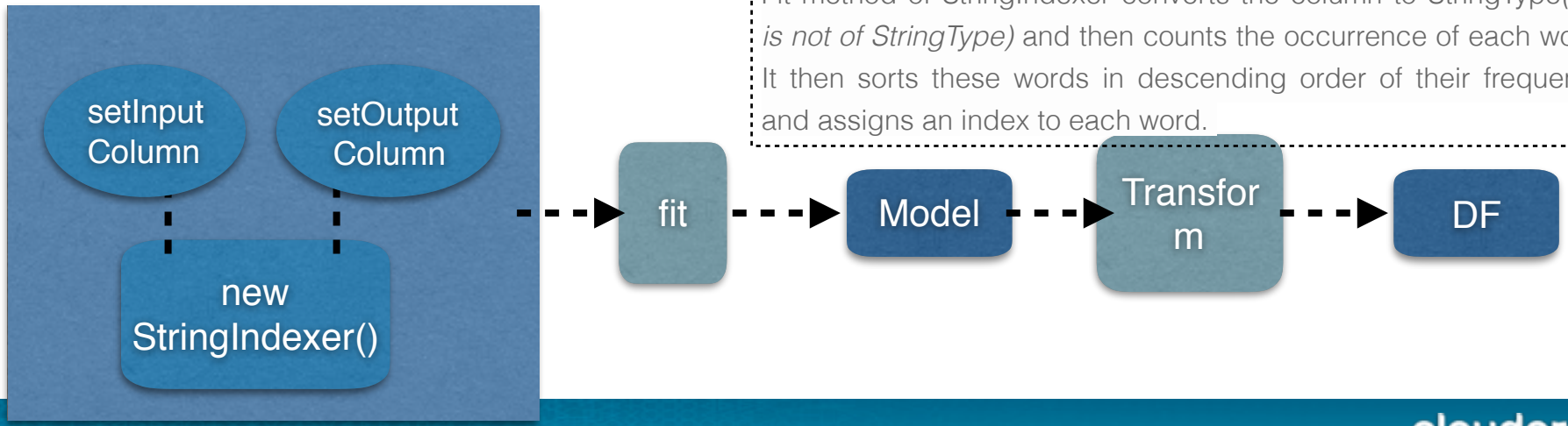
Transformers

StringIndexer, IndexToString, OneHotEncoder, VectorIndexer, Tokenizer,



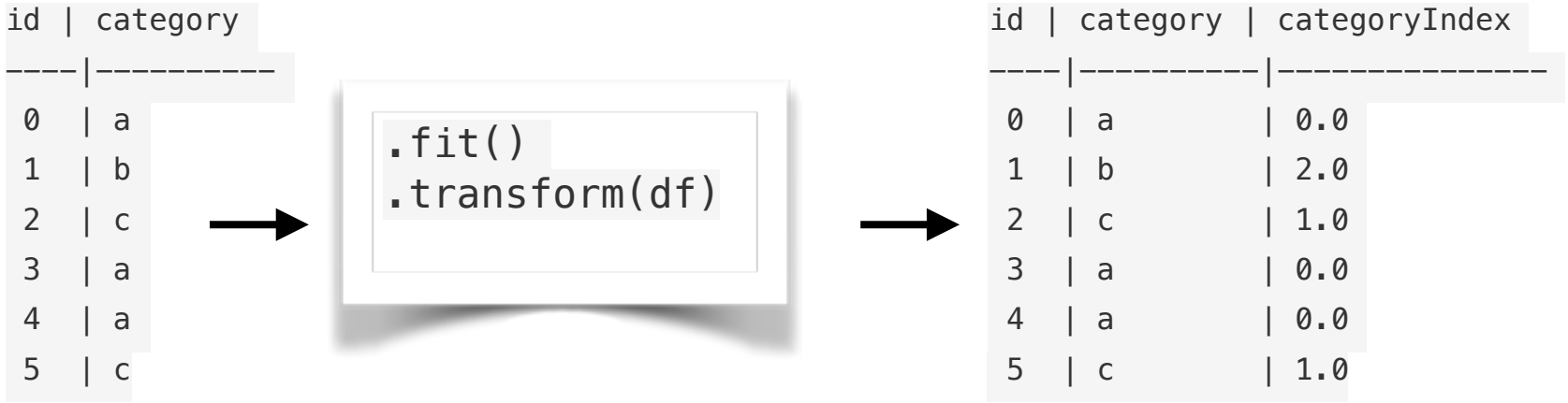
StringIndexer

- To build a model in Spark, the features must be of type Double but we have a few features which are of the type String.
- StringIndexer is a transformer, which encodes a string column of labels to a column of label indices.
- Indices are in (0, numLabels)
- Ordered by label frequencies, so the most frequent label gets index 0.



Fit method of StringIndexer converts the column to StringType*(if it is not of StringType)* and then counts the occurrence of each word. It then sorts these words in descending order of their frequency and assigns an index to each word.

StringIndexer



StringIndexer - Unseen Labels

- Throw an exception -> This is default
- Skip the row containing the unseen label entirely.

```
.handleInvalid(skip/error)
```

IndexToString

- Maps a column of label indices back to a column containing the original labels as Strings.
- A common use case is to produce indices from labels with StringIndexer, train a model with those indices and retrieve the original labels from the column of predicted indices with IndexToString.
- labels: Optional -> param for array of labels specifying index-string mapping

OneHotEncoder

- Maps a column of label indices back to a column of binary vectors, with at most a single one-value.
- Allow algorithms to use categorical features.
- Used to distinguish each word in a vocabulary from every other word in the vocabulary.

VectorIndexer

- `VectorIndexer` helps index categorical features in datasets of `Vectors`.
- It can both automatically decide which features are categorical and convert original values to category indices. Used to distinguish each word in a vocabulary from every other word in the vocabulary.
- maxCategories

```
new VectorIndexer()
  .setInputCol("features")
  .setOutputCol("indexed")
  .setMaxCategories(10)
  .fit(data)
  .transform(data)
```

Tokenizer

```
new RegexTokenizer()
  .setInputCol("sentence")
  .setOutputCol("words")
  .setPattern("\\W")
  .transform(sentenceDataFrame)
```

- Tokenization is the process of taking text (such as a sentence) and breaking it into individual terms (usually words).
- RegexTokenizer allows more advanced tokenization based on regular expression (regex) matching.
- gaps
- minTokenLength
- pattern
- toLowerCase

StopWordsRemover

```
new RegexTokenizer()
  .setInputCol("sentence")
  .setOutputCol("words")
  .setPattern("\\W")
  .transform(sentenceDataFrame)
```

- Tokenization is the process of taking text (such as a sentence) and breaking it into individual terms (usually words).
- RegexTokenizer allows more advanced tokenization based on regular expression (regex) matching.
- gaps
- minTokenLength
- pattern
- toLowerCase

n-gram

```
new NGram()
.setN(2)
.setInputCol(“words")
.setOutputCol("ngrams")
```

- An n-gram is a sequence of n tokens (typically words) for some integer n.
- The parameter n is used to determine the number of terms in each n-gram.gap.

Normalizer

```
new Normalizer()
  .setInputCol("features")
  .setOutputCol("normFeatures")
  .setP(1.0)
```

- `Normalizer` is a `Transformer` which transforms a dataset of `Vector` rows, normalizing each `Vector` to have unit norm.
- It takes parameter `p`, which specifies the p-norm used for normalization.
- This normalization can help standardize your input data and improve the behavior of learning algorithms.

VectorAssembler

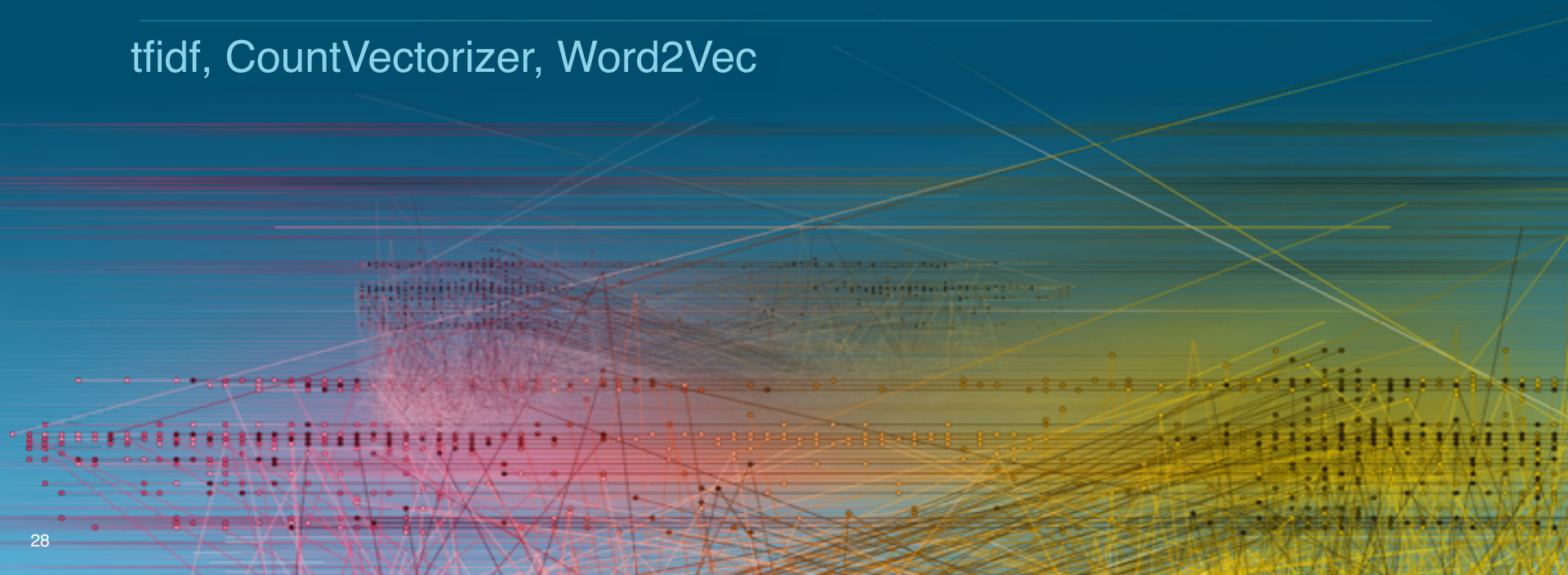
```
val dataset = spark.createDataFrame(
  Seq((0, 18, 1.0, Vectors.dense(0.0, 10.0,
0.5), 1.0))
).toDF("id", "hour", "mobile",
"userFeatures", "clicked")

val assembler = new VectorAssembler()
  .setInputCols( Array
     ("hour", "mobile", “userFeatures”)
     )
  .setOutputCol("features")
```

- `VectorAssembler` is a transformer that combines a given list of columns into a single vector column.
- It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
- `VectorAssembler` accepts the following input column types: all numeric types, boolean type, and vector type.
- In each row, the values of the input columns will be concatenated into a vector in the specified order.

cloudera
Ask Bigger Questions

Extractors

tfidf, CountVectorizer, Word2Vec



TFIDF

- A feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus.gaps
- Term Frequency - it is very easy to over-emphasize terms that appear very often but carry little information about the document.
- Inverse Document Frequency - a numerical measure of how much information a term provides
- In MLLib, TF and IDF are separated.

TF

- HashingTF and CountVectorizer
- Takes sets of terms and converts those sets into fixed-length feature vectors
- Utilizes Hashing trick - a raw feature is mapped into an index (term) by applying a hash function.
- Avoids the need to compute a global term-to-index map, which can be expensive.
- Suffers from potential hash collisions.
- To reduce the chance of collision, we can increase the target feature dimension, i.e. the number of buckets of the hash table.
- An optional binary toggle parameter controls term frequency counts.

IDF

- IDF is an Estimator which is fit on a dataset and produces an IDFModel.
- The IDFModel takes feature vectors (generally created from HashingTF or CountVectorizer) and scales each column
- Intuitively, it down-weights columns which appear frequently in a corpus.

CountVectorizer

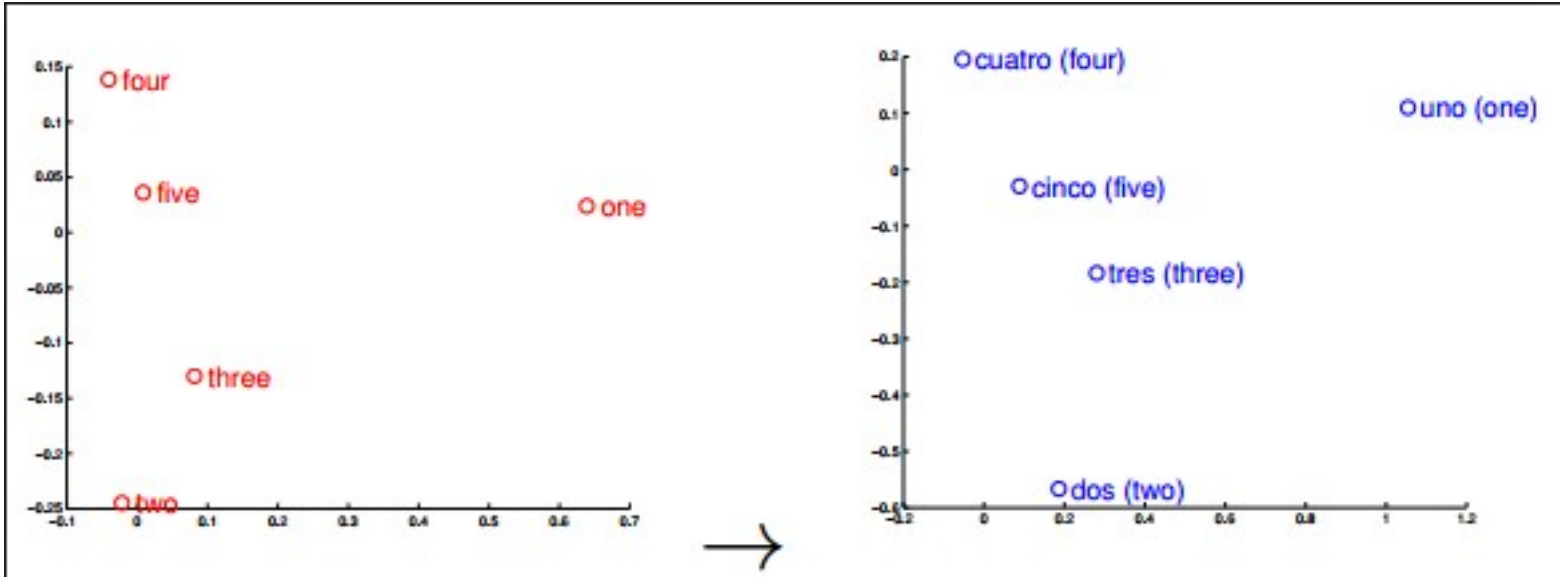
- Convert a collection of text documents to vectors of token counts.
- When an a-priori dictionary is not available
- The model produces sparse representations for the documents over the vocabulary, which can then be passed to other algorithms like LDA.

CountVectorizer - Tuning Parameters

- vocabSize : Top number of words ordered by term frequency across the corpus.
- minDF : Affects the fitting process by specifying the minimum number (or fraction if < 1.0) of documents a term must appear in to be included in the vocabulary.
- minTF: Filter to ignore rare words in a document. For each document, terms with frequency/count less than the given threshold are ignored.
- binary: If True, all nonzero counts (after minTF filter applied) are set to 1.

Word2Vec

- The model maps each word to a unique fixed-size vector.
- The Word2VecModel transforms each document into a vector using the average of all words in the



Word2Vec - Tuning Parameters

- maxIter : Maximum number of Iterations (>0)
- maxSentenceLength : Sets the maximum length (in words) of each sentence in the input data. Any sentence longer than this threshold will be divided into chunks of up to maxSentenceLength size. Default: 1000
- minCount : The minimum number of times a token must appear to be included in the word2vec model's vocabulary. Default: 5
- stepSize : Param for Step size to be used for each iteration of optimization (> 0).
- vectorSize : The dimension of the code that you want to transform from words. Default: 100

cloudera
Ask Bigger Questions

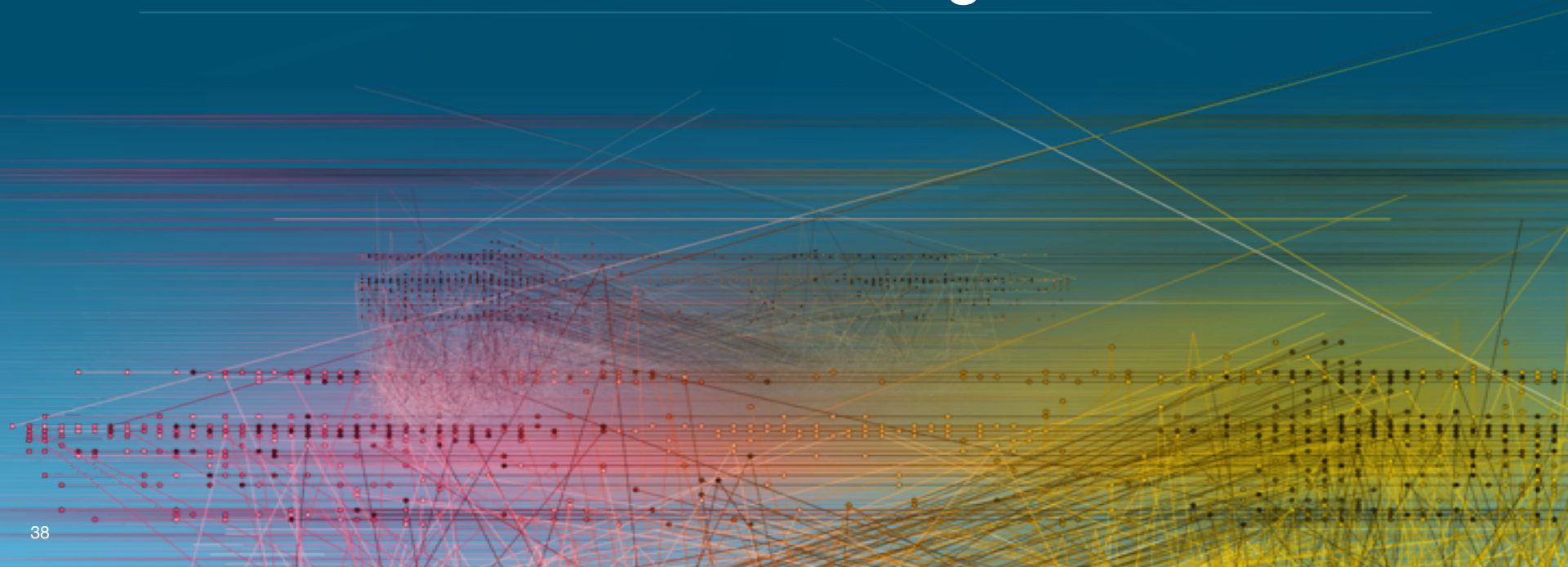
Exercise - Q&A

Exercise

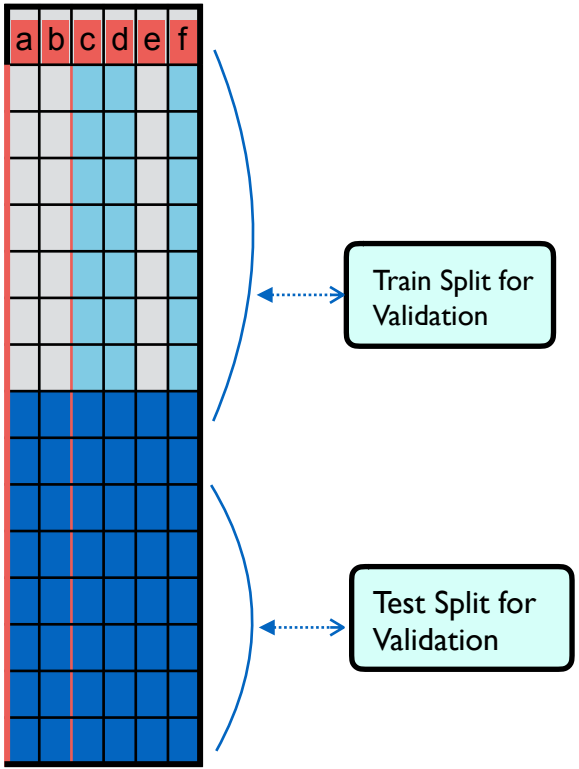
- DataSet = data/books/all
  - Read in all the data
  - Tokenize
  - Apply stop words if needed.
  - Apply HashingTF and scale with IDF.

cloudera
Ask Bigger Questions

Model Selection and Tuning



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

Parameters: Model tuning

Common
estimator - estimator to be evaluated. Examples: LinearRegression, Pipeline, DecisionTreeRegressor
estimatorParamMaps - a grid of parameters to evaluate the model
evaluator - used to select hyper-parameters that maximize the validated metric. Examples: BinaryClassificationEvaluator, RegressionEvaluator, MulticlassClassificationEvaluator

CrossValidator	TrainValidationSplit
numFolds - split input data into this many parts. Default: 3	**trainRatio** - split input data set into training and validation data with this ratio. Default: 0.75

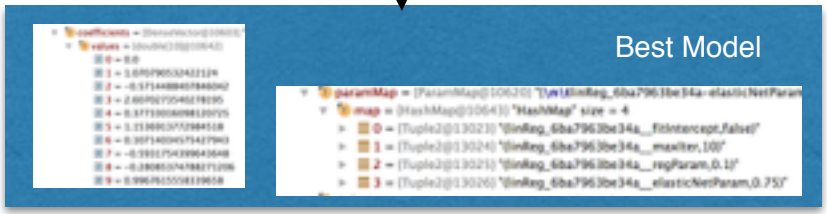
TrainValidationSplit

	features
	(10 [0 1 2 3 4 5 6 7 8 9] [0.45 0.36 -0.38 -0.44 0.33 ….]
	(10 [0 1 2 3 4 5 6 7 8 9] [0.83 -0.12 0.50 -0.22 -0.64 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [0.50 0.14 0.16 0.50 -0.93 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [-0.03 -0.41 0.90 0.64 0.27 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [-0.06 0.65 -0.69 0.66 -0.07 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [-0.15 0.26 0.40 -0.36 0.14 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [0.39 0.81 -0.60 0.61 0.25 …]
	(10 [0 1 2 3 4 5 6 7 8 9] [-0.00 -0.94 -0.92 -0.03 0.31 …
	(10 [0 1 2 3 4 5 6 7 8 9] [0.78 0.19 0.95 -0.84 0.55 …]
	…

Param Grid				
	1	0.1	0.01	
0	(1,0,true)	(0.1,0,true)	(0.01,0,true)	TRUE
0.25	(1,0.25,true	(0.1,0.25,true)	(0.01,0.25,true)	
0.5	(1,0.5,true)	(0.1,0.5,true)	(0.01,0.5,true)	
0.75	(1,0.75,true	(0.1,0.75,true)	(0.01,0.75,true)	
0	(1,0,false)	(0.1,0,false)	(0.01,0,false)	FALSE
0.25	(1,0.25,fals	(0.1,0.25,false	(0.01,0.25,false)	
0.5	(1,0.5,false)	(0.1,0.5,false)	(0.01,0.5,false)	
0.75	(1,0.75,fals	(0.1,0.75,false	(0.01,0.75,false)	

org.apache.spark.ml.regression.LinearRegression

org.apache.spark.ml.evaluation.RegressionEvaluator



TrainValidationSplit - Code Walk Through

org.apache.spark.ml.regression.LinearRegression

```
val lr = new LinearRegression()
        .setMaxIter(10)
```

org.apache.spark.ml.tuning.ParamGridBuilder

```
val paramGrid = new ParamGridBuilder()
    .addGrid(lr.regParam, Array(0.1, 0.01, 1))
    .addGrid(lr.fitIntercept, Array(true, false))
    .addGrid(lr.elasticNetParam, Array(0.0, 0.25, 0.5, 0.75))
    .build()
```

org.apache.spark.ml.tuning. TrainValidationSplit

```
val trainValidationSplit = new TrainValidationSplit()
      .setEstimator(lr)
      .setEvaluator(new RegressionEvaluator)
      .setEstimatorParamMaps(paramGrid)
      .setTrainRatio(0.8)
val model = trainValidationSplit.fit(training)
```

CrossValidator

Param Grid			
	10	100	1000
0.01	(0.01,10)	(0.01,100)	(0.01,1000
0.1	(0.1,10)	(0.1,100)	(0.1,1000)

label	text
1	ad-jerry ad-bruckheimer ad-chase ad-premier
0	ad-rheumatoid ad-arthritis ad-expert ad-tip
0	ad-rheumatologist ad-anju ad-varghese ad-yonker
0	ad-siemen ad-water ad-remediation ad-water
0	ad-symptom ad-muscle ad-weakness ad-genetic
1	ad-animal ad-animal ad-wild ad-sa ad-official
0	ad-dr ad-enrico ad-fazzini ad-parkinson
0	ad-ulcerative ad-colitis ad-uc ad-quiz ad-ulcerative
0	ad-wellcentive ad-registry ad-web ad-base ad-

label	text
1	ad-free ad-raw ad-food ad-video ad-feature ad-mike
0	ad-north ad-shore ad-lij ad-cancer ad-center
1	ad-world ad-finest ad-alpaca ad-breeder
1	ad-vet ad-online ad-veterinarian ad-online
0	ad-gum ad-disease ad-treatment ad-dentist ad-near
1	ad-rabbit ad-guinea ad-pig ad-hutch ad-deal ad-save
0	ad-colitis ad-symptom ad-ulcerative ad-colitis ad-
0	ad-disease ad-sign ad-symptom ad-cause ad-lung ad-
1	ad-pygmy ad-goat ad-pygmy ad-goat title-pygmy title-
1	ad-feed ad-supplement ad-buy ad-nutritional ad-

tokenizer, hashingTF, idf, lr

BinaryClassificaitonEvaluator → (0.01,1000)

tokenizer, hashingTF, idf,

BinaryClassificaitonEvaluator → (0.1,1000)

Best Param
(0.01,1000)
(0.1,1000)

label	text
1	ad-jerry ad-bruckheimer ad-chase ad-premier
0	ad-rheumatoid ad-arthritis ad-expert ad-tip
0	ad-rheumatologist ad-anju ad-varghese ad-
0	ad-siemen ad-water ad-remediation ad-water
0	ad-symptom ad-muscle ad-weakness ad-
1	ad-animal ad-animal ad-wild ad-sa ad-official
0	ad-dr ad-enrico ad-fazzini ad-parkinson
0	ad-ulcerative ad-colitis ad-uc ad-quiz ad-
0	ad-wellcentive ad-registry ad-web ad-base ad-
1	ad-free ad-raw ad-food ad-video ad-feature ad-
0	ad-north ad-shore ad-lij ad-cancer ad-center
1	ad-world ad-finest ad-alpaca ad-breeder
1	ad-vet ad-online ad-veterinarian ad-online
0	ad-gum ad-disease ad-treatment ad-dentist ad-
1	ad-rabbit ad-guinea ad-pig ad-hutch ad-deal
0	ad-colitis ad-symptom ad-ulcerative ad-colitis
0	ad-disease ad-sign ad-symptom ad-cause ad-
1	ad-pygmy ad-goat ad-pygmy ad-goat title-
1	ad-feed ad-supplement ad-buy ad-nutritional

tokenizer
hashingTF
idf
lr

Binary Classification Evaluator

Best Model

CrossValidation - Code Walk Through

org.apache.spark.ml.Pipeline

```
val tokenizer = new Tokenizer()
      .setInputCol("text")
      .setOutputCol("words")

val hashingTF = new HashingTF()
      .setInputCol(tokenizer.getOutputCol)
      .setOutputCol("rawFeatures")

val idf = new IDF()
       .setInputCol("rawFeatures")
       .setOutputCol("features")

val lr = new LogisticRegression()
      .setMaxIter(10)

val pipeline = new Pipeline()
   .setStages(Array(tokenizer, hashingTF, idf, lr))
```

org.apache.spark.ml.tuning.ParamGridBuilder

```
val paramGrid = new ParamGridBuilder()
  .addGrid(hashingTF.numFeatures, Array(10, 100, 1000))
   .addGrid(lr.regParam, Array(0.1, 0.01))
   .build()
```

org.apache.spark.ml.tuning.CrossValidator

```
val cv = new CrossValidator()
   .setEstimator(pipeline)
   .setEvaluator(new BinaryClassificationEvaluator)
   .setEstimatorParamMaps(paramGrid)
  .setNumFolds(2)

val cvModel = cv.fit(trainingDF)
```

Model Tuning

Code Walk Through

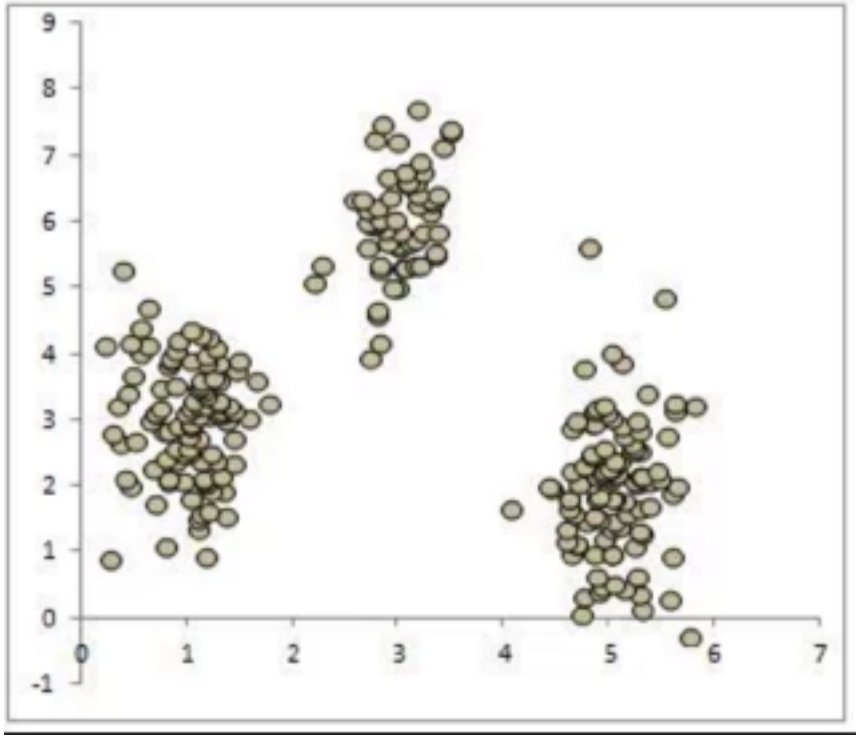
cloudera
Ask Bigger Questions

Clustering

KMeans, LDA, GMM

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



Tuning parameters: KMeans

featuresCol - dataframe column that specifies features
k - number of clusters to create. Default: 2
maxIter - maximum number of iterations
predictionCol - dataframe column that will have cluster ID prediction
tol - convergence tolerance
initMode (Advanced) - Initialization algorithm. **random** to chose random points for centroids. **k-means||** - (default) parallel variant of k-means++
initSteps (Advanced) - Number of steps for initialization mode. Default: 2

KMeans - Reading the Extracting Features

```
val inputData = session.read
                .option(“header","true")
                .option("inferSchema","true").csv(dataset)
```

Infer the schema from the header

Name	Day	Fligh	Arrival	PayGrade
Kevin	Saturda	VX	2:30 PM	1
Darran	Saturda	UA12	6:00 PM	3
..	..	..	..	..

Generate Features

Name	Flig	Flight	Arr	Pa	Saturday	Sunday	Monday	dateFr	Grad
Kevin Doe	Satu	VX 0906	2:30	1	1	0	0	0.6728	1
Darran Doe	Satu	UA1248	6:00	3	1	0	0	0.7935	3
..	..	..	..	..	..	..	..	..	..

org.apache.spark.ml.feature.VectorAssembler

Name	Day	Fligh	Arrival	PayGr	Saturd	Sund	Mond	date	Grad	features
Kevin	Satur	VX	2:30 PM	1	1	0	0	0.6728	1	[1.0,0.0,0.0,0.6728,1.0]
Darran	Satur	UA12	6:00 PM	3	1	0	0	0.7935	3	[1.0,0.0,0.0,0.7935,3.0]
..	..	..	..	..	..	..	..	..	..	..

KMeans - Scaling and Running Model

Nam	Day	Flig	Arrival	Pa	Satu	Su	Mo	dateFr	Gr	features
Kevin	Satu	VX	2:30 PM	1	1	0	0	0.6728	1	[1.0,0.0,0.0,0.67280
Darra	Satu	UA1	6:00 PM	3	1	0	0	0.7935	3	[1.0,0.0,0.0,0.79355
..	..	..	..	..	..	..	..	..	..	..

org.apache.spark.ml.feature.MinMaxScaler

Na	D	Fli	Arr	Pa	Sa	S	M	da	G	features	scaled_features
Ke	Sa	V	2:30	1	1	0	0	0.6	1	[1.0,0.0,0.0,0.0	[1.0,0.0,0.0,0.6935388263
Dar	Sa	U	6:00	3	1	0	0	0.7	3	[1.0,0.0,0.0,0.0	[1.0,0.0,0.0,0.8180201541
..	..	..	..	..	..	..	..	..	..	..	..

org.apache.spark.ml.clustering.KMeans

Nam	Day	Flig	Arriva	Pay	Sat	Su	Mo	date	Gr	features	scaled_features	clusterId
Kevin	Sat	VX	2:30 PM	1	1	0	0	0.672	1	[1.0,0.0,0.0,0.672800	[1.0,0.0,0.0,0.69353	10
Darra	Sat	UA1	6:00 PM	3	1	0	0	0.793	3	[1.0,0.0,0.0,0.793559	[1.0,0.0,0.0,0.81802	13
..	..	..	..	..	..	..	..	..	..	..	..	..

KMeans - Code Walk Through

```
val dataset = "data/kmeans/flightinfo/flights_nofeatures.csv"
```

```
val isSat = udf {(x:String) => if (x.toLowerCase.equals("saturday")) 1 else 0}
val isSun = udf {(x: String) => if (x.toLowerCase.equals("sunday")) 1 else 0}
val isMon = udf {(x: String) => if (x.toLowerCase.equals("monday")) 1 else 0}

val transformedDay = inputData.withColumn("Saturday", isSat(inputData("Day")))
                              .withColumn("Sunday", isSun(inputData("Day")))
                              .withColumn("Monday", isMon(inputData("Day")))
val dayFract = udf {(x:String) =>
                      if (x == null)
                        0
                      else
                      {
                        val formatter = new java.text.SimpleDateFormat("h:m a")
                        val curr = formatter.parse(x).getTime.toDouble
                        val full = formatter.parse("11:59 PM").getTime.toDouble
                        curr/full
                      }
                    }
```

KMeans - Code Walk Through (Cont'd)

```
val assembler = new VectorAssembler()
       .setInputCols(Array("Saturday","Sunday","Monday","dateFract","Grade"))
       .setOutputCol("features")

val featurizedData = assembler.transform(transformedTime)
```



```
val scaler = new MinMaxScaler()
            .setInputCol("features")
            .setOutputCol("scaled_features")
val scalerModel = scaler.fit(featurizedData)
val scaledData = scalerModel.transform(featurizedData)
```

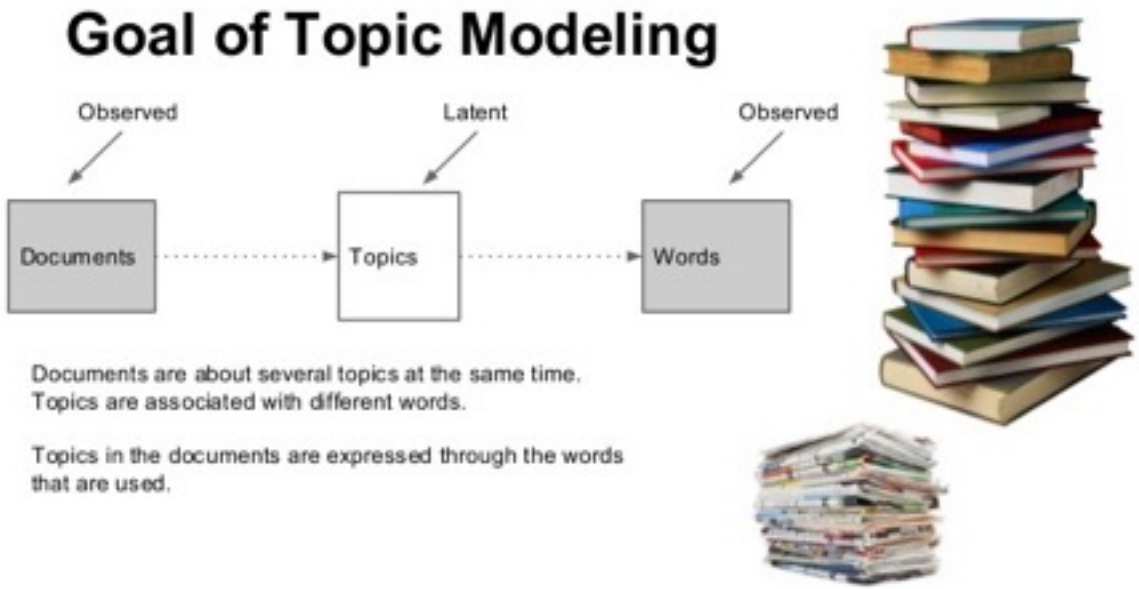


```
val kmeans = new KMeans()
    .setK(20)
    .setFeaturesCol("scaled_features")
    .setPredictionCol("clusterId")
val model = kmeans.fit(scaledData)
```

KMeans

Code Walk Through

Clustering - LDA



Tuning Parameters - LDA

- k : number of topics to be inferred
- maxIter: >0
- optimizer: online/em
- seed
- subSamplingRate: (for online only): should be adjusted in synch with LDA.maxIter so the entire corpus is used. Specifically, set both so that maxIterations * miniBatchFraction greater than or equal to 1.
- topicConcentration
- topicDistributionCol

Looking at the results

Topic 1	
space	0.0127
nasa	0.0053
program	0.0052
available	0.0051
system	0.0050
data	0.0048

Topic 2	
colorado	0.0266
guns	0.0140
ucsu	0.0131
udel	0.0130
firearms	0.0122
intercon	0.0108

Topic 3	
people	0.0077
turkish	0.0076
israel	0.0067
mideast	0.0066
jewish	0.0059
jews	0.0056

Topic 4	
news	0.0206
baseball	0.0195
sport	0.0125
game	0.0101
subject	0.0098
organization	0.0096

Topic 5	
duke	0.0139
team	0.0099
hockey	0.0081
sport	0.0080
news	0.0079
ulowell	0.0073
league	0.0071

Topic 6	
rutgers	0.0260
christian	0.0171
religion	0.0132
writes	0.0098
talk	0.0097
lines	0.0095

Exercise and Q&A

NewsGroup

KMeans

LDA

- experiment with cluster numbers and vocabulary size
- read all the files
- RegTokenizer
- StopWordsRemover
- NGram
- CountVectorizer
- LDA
- Print Clusters

cloudera
Ask Bigger Questions

Regression

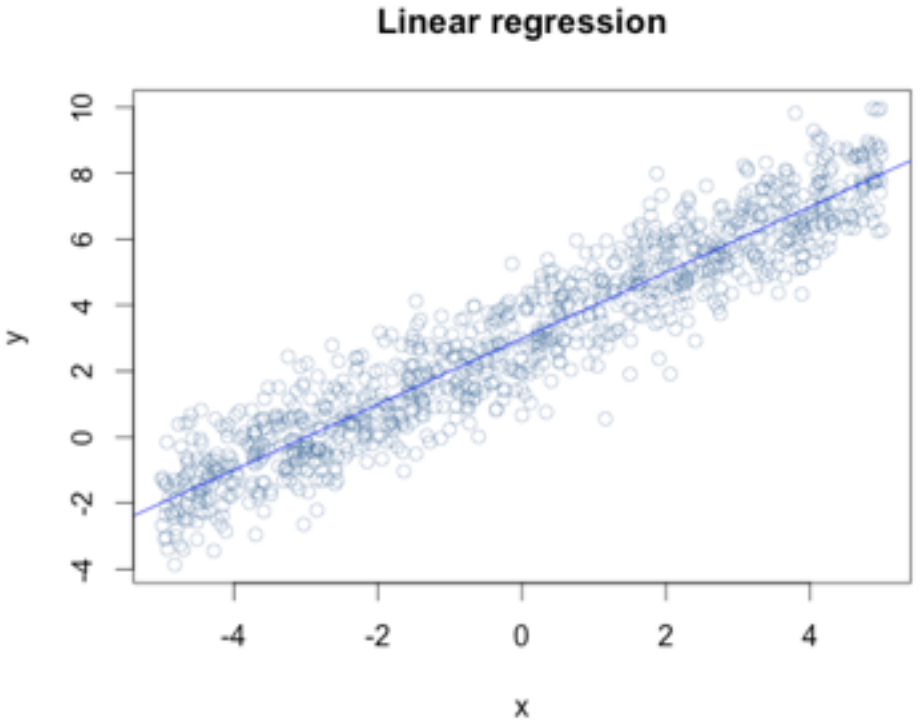
Linear Regression, Random Tree Regression

Regression

- Modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. => $\hat{y} = w` x$, 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.

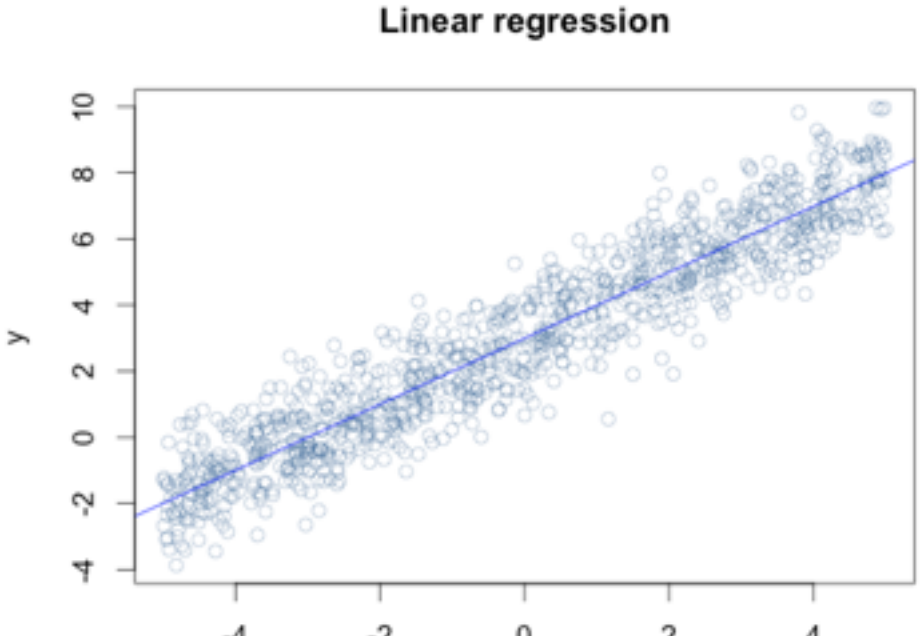
- Parameters are values that control the behavior of the system.



Generalized Linear Regression

- Linear: A constant change in a predictor leads to a constant change in the response variable

- Generalized linear models cover all situations by allowing for response variables that have arbitrary distributions (rather than simply normal distributions), and for an arbitrary function of the response variable (the *link function*) to vary linearly with the predicted values



Tuning parameters: LinearRegression

featuresCol - dataframe column that specifies features
fitIntercept - parameter to instruct to fit intercept
maxIter - maximum number of iterations
predictionCol - dataframe column that will have predicted outcome
labelCol - label column name
elasticNetParam - elasticNet mixing parameter [0 - L2 penalty, 1 - L1 penalty]
regParam - regularization parameter
solver - solver algorithm for optimization (auto, normal, l-bfgs)
standardization - instruct whether to standardize the training features before fitting the model
weightCol - weight column name
tol - convergence tolerance

Linear Regression - Feature Extraction

id	price	lotsize	bedrooms	bathrms	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea
1	79,000	3300	3.0	3.0	2.0	yes	no	yes	no	no	0.0	no
2	58,000	4992	3.0	2.0	2.0	yes	no	no	no	no	2.0	no
3	43,000	4600	2.0	1.0	1.0	yes	no	no	no	no	0.0	no

org.apache.spark.ml.feature.StringIndexer

id	drivewayIdx	recroomIdx	fullbaseIdx	gashwIdx	aircoIdx	prefareaIdx
1	0.0	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0

org.apache.spark.ml.feature.OneHotEncoder

id	drivewayV	recroomV	fullbaseV	gashwV	aircoV	prefareaV
1	(1,[0],[1.0])	(1,[0],[1.0])	(1,[],[])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])
2	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])
3	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])

Linear Regression - Feature Extraction

id	price	lotsize	bedrooms	athrm	stories	driveway	recroom	fullbase	gashw	airco	garagepl	prefarea	drivewayIdx	recroomIdx	fullbaseIdx	gashwIdx	aircoIdx	efareaId	drivewayV	recroomV	fullbaseV	gashwV	airc	prefarea
1	79,000	3300	3.0	3.0	2.0	yes	no	yes	no	no	0.0	no	0.0	0.0	1.0	0.0	0.0	0.(	(1,[0],[1.0])	(1,[0],[1.0])	(1,[],[])	(1,[0],[1.0])	[0],[1.	[0],[1.0
2	58,000	4992	3.0	2.0	2.0	yes	no	no	no	no	2.0	no	0.0	0.0	0.0	0.0	0.0	0.(	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	[0],[1.	[0],[1.0
3	43,000	4600	2.0	1.0	1.0	yes	no	no	no	no	0.0	no	0.0	0.0	0.0	0.0	0.0	0.(	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	[0],[1.	[0],[1.0

org.apache.spark.ml.feature.VectorAssembler

id	features
1	[3300.0,3.0,3.0,2...
2	[4992.0,3.0,2.0,2...
3	[4600.0,2.0,1.0,1...

Vector Assembler assembles:

- lotsize
- bedrooms
- bathrms
- stories
- garagepl
- drivewayVec
- recroomVec
- fullbaseVec
- gashwVec
- aircoVec
- prefareaVec

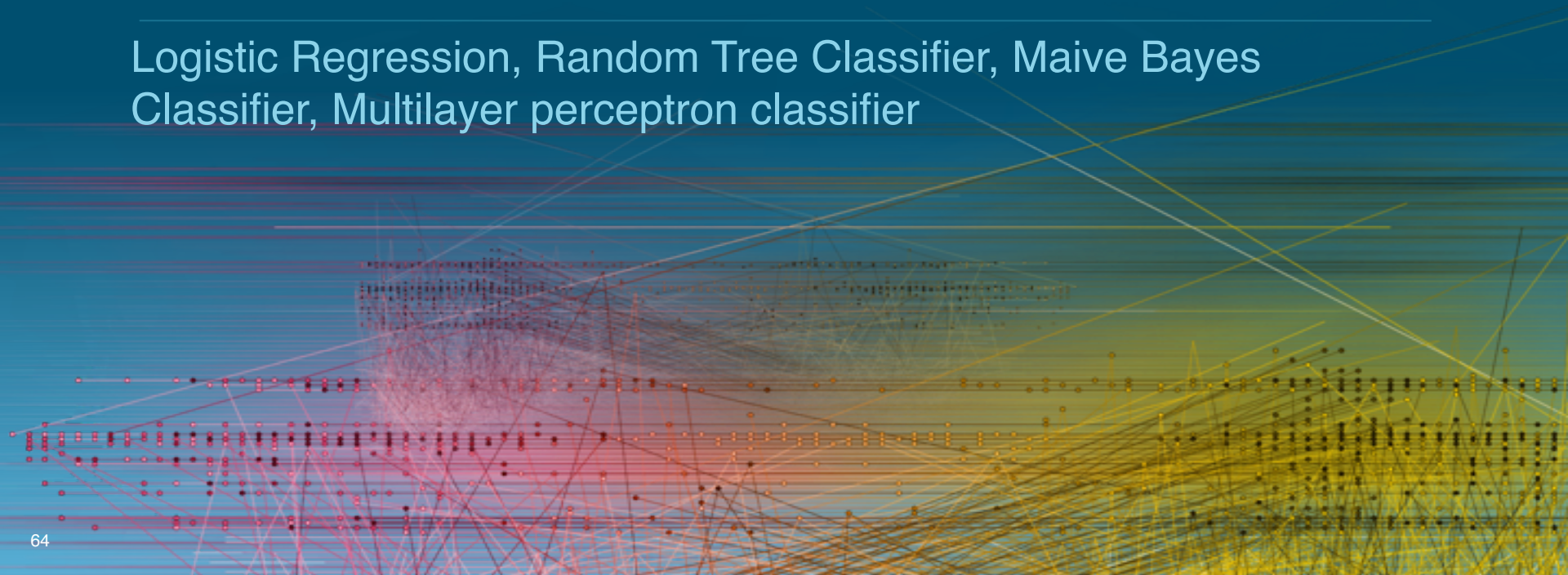
org.apache.spark.ml.regression.LinearRegression

id	features	prediction
1	[3300.0,3.0,3.0,2...	81,214.32
2	[4992.0,3.0,2.0,2...	76,821.57
3	[4600.0,2.0,1.0,1...	44,127.73

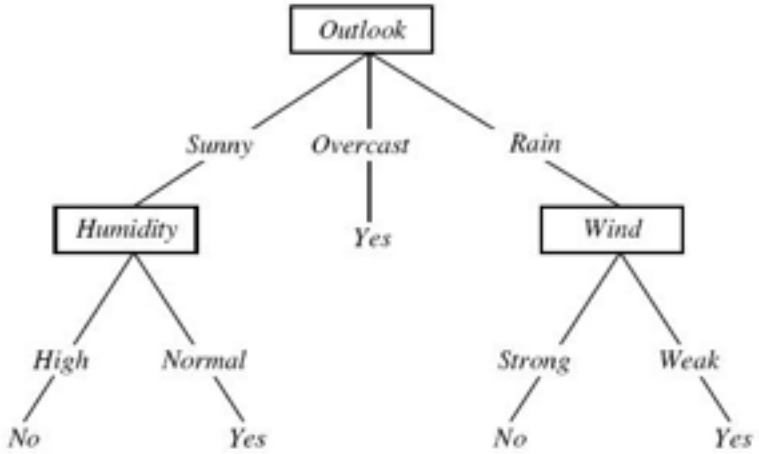
cloudera
Ask Bigger Questions

Classification

Logistic Regression, Random Tree Classifier, Maive Bayes Classifier, Multilayer perceptron classifier



Decision Tree



No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Random Forest Classifier

- Support both binary and multi class classification and regression
- Use both Categorical and Continuous features
- Random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.
- Implements random forests using the existing decision tree implementation.

Random Forest Classifier

Training

- Since training of decisions trees is done separately, the training is done in parallel.
- Randomness injected into the training process includes:
  - Subsampling the original dataset on each iteration to get a different training set.
  - Considering different random subsets of features to split at each tree node

Prediction

- Aggregates predictions from its set of decision trees.
- Classification: Majority vote
- Regression: Averaging

Random Forest Classifier - Tuning Parameters

- numTrees: Number of trees in the Forest => accuracy vs speed
- maxDepth: Expressive vs overfitting. More suitable for random forests than a single decision tree.
- subsamplingRate
- featuredSubsetStrategy – auto, all, onethird, sort, log2, n
- impurity: entropy, gini
- minInfoGain
- minInstancePerNode

Random Forest Classifier - IP/OP Parameters

Input Column	Types	Default	Description
labelCol	Double	“label”	Label to Predict
featuresCol	Vector	“features”	Feature Vector

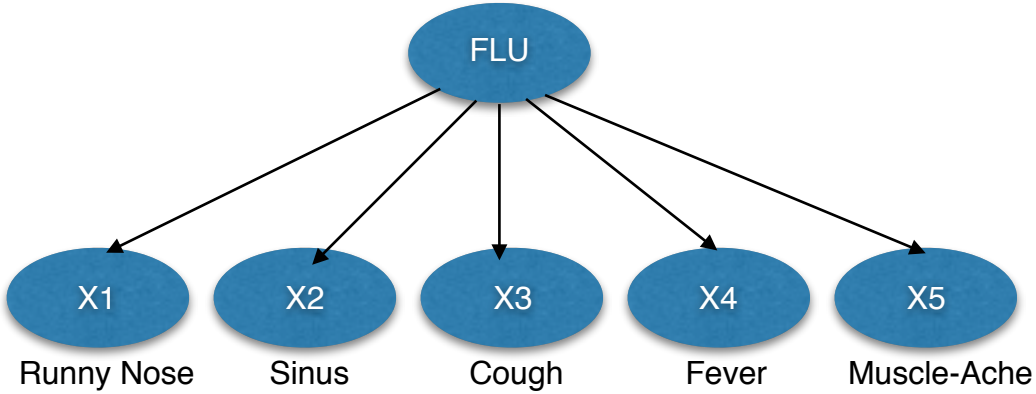
Output Column	Types	Default	Description
predictionCol	Double	"prediction"	Predicted label
rawPredictionCol	Vector	"rawPrediction"	Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction
probabilityCol	Vector	"probability"	Vector of length # classes equal to rawPrediction normalized to a multinomial distribution

Random Forest Classifier - Exercise and Q&A

- Read the data set : data/moviereviews.tsv, data/stopwords.txt
- Would you want to clean the data?
- Tokenize
- Do you want to get rid of noisy words.
- Problems with string labels
- Evaluate on the held out test data.
- Train the classifier
- Get the labels back
- Predict and evaluate

Naive Bayes Classifier

- Conditional Independence Assumption: Features are independent given the class.
- The spark.ml implementation currently supports both multinomial naive Bayes and Bernoulli naive Bayes.
- Similar parameters as to Random Forest Classifier.

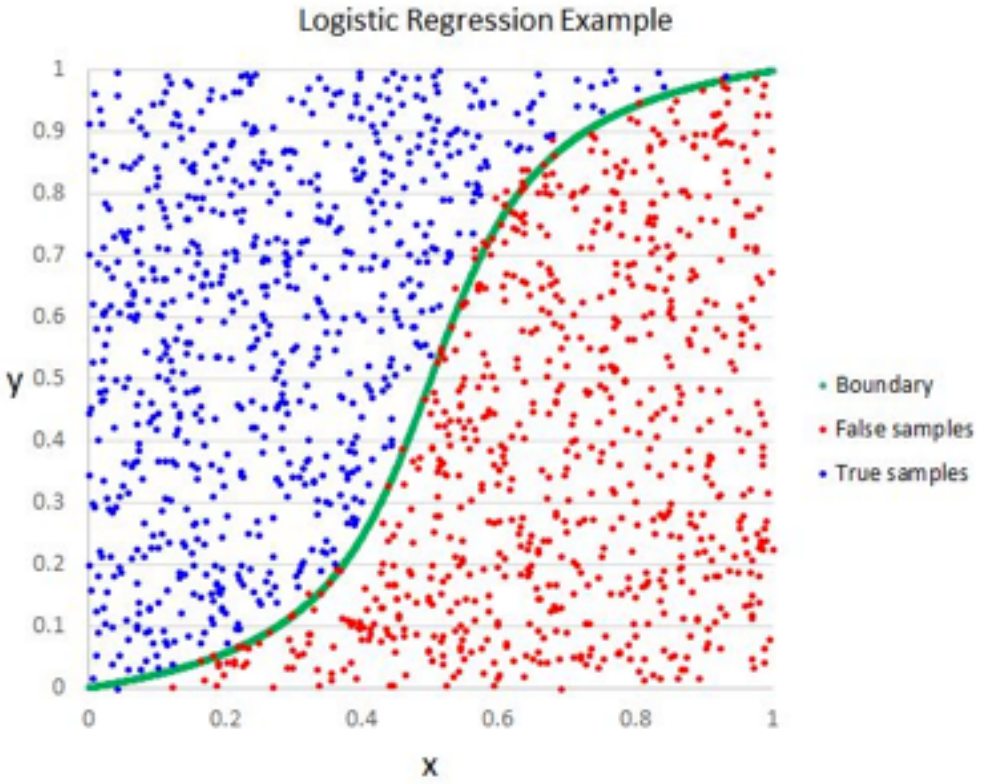


Naive Bayes - Exercise and Q&A

- Read the data set : data/SMSSpamCollection.txt
- Create TF
- Create IDF
- Training and Testing split
- Train the NaiveBayes Model
- Evaluate on the held out test data.

Logistic Regression

- A special case of Generalized Linear models that predicts the probability of the outcomes
- In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression.



Tuning parameters: LogisticRegression

featuresCol - dataframe column that specifies features
fitIntercept - parameter to instruct to fit intercept
maxIter - maximum number of iterations
predictionCol - dataframe column that will have predicted outcome
labelCol - label column name
elasticNetParam - elasticNet mixing parameter [0,1]
regParam - regularization parameter
solver - solver algorithm for optimization
standardization - instruct whether to standardize the training features before fitting the model
weightCol - weight column name
tol - convergence tolerance
family - name of family which is a description of the label distribution
probabilityCol - column name for predicted class conditional probabilities
rawPredictionCol - raw prediction column name
threshold - threshold in binary classification prediction
thresholds - thresholds in multi-class classification to adjust the probability of predicting each class

Input & Output Columns

Input Column	Types	Default	Description
labelCol	Double	“label”	Label to Predict
featuresCol	Vector	“features”	Feature Vector

Output Column	Types	Default	Description
predictionCol	Double	"prediction"	Predicted label
rawPredictionCol	Vector	"rawPrediction"	Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction
probabilityCol	Vector	"probability"	Vector of length # classes equal to rawPrediction normalized to a multinomial distribution

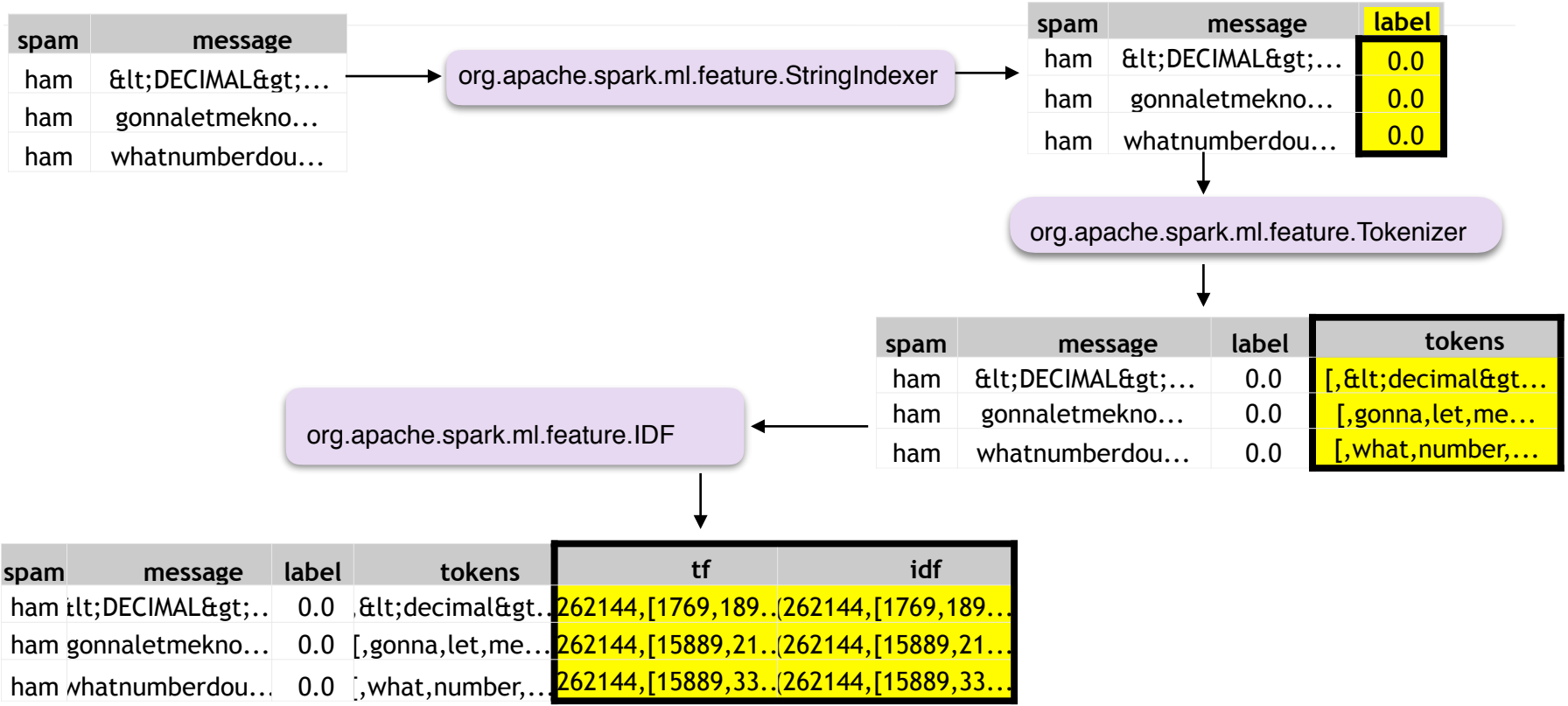
Logistic Regression

- SMS Spam Collection Dataset
- https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection#

Look into the Data...

spam	message
StringType	StringType
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
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.

LogisticRegression - Feature Extraction



LogisticRegression - Feature Extraction (cont'd)

spam	message	label	tokens	tf	idf
ham	<DECIMAL>...	0.0	,<decimal&gt..	(262144,[1769,189...	(262144,[1769,189...
ham	gonnaletmekno...	0.0	[,gonna,let,me...	(262144,[15889,21...	(262144,[15889,21...
ham	whatnumberdou...	0.0	[,what,number,...	(262144,[15889,33...	(262144,[15889,33...

→ org.apache.spark.ml.feature.VectorAssembler

spam	message	label	tokens	tf	idf	features
ham	<DECIMAL>...	0.0	[,<decimal&gt..	262144,[1769,189..	262144,[1769,189..	(262144,[1769,189...
ham	gonnaletmekno...	0.0	[,gonna,let,me...	262144,[15889,21..	262144,[15889,21..	(262144,[15889,21...
ham	whatnumberdou...	0.0	[,what,number,...	262144,[15889,33..	262144,[15889,33..	(262144,[15889,33...

org.apache.spark.ml.classification.LogisticRegression

spam	message	label	tokens	tf	idf	features	rawPredictior	probability	prediction
ham	<DECIMAL>...	0.0	[,<decimal&gt...	(262144,[1769,189...	(262144,[1769,189...	62144,[1769,189	[30.92...	[0.99...	0.0
ham	gonnaletmekno...	0.0	[,gonna,let,me...	(262144,[15889,21...	(262144,[15889,21...	62144,[15889,21	[26.54...	[0.99...	0.0
ham	whatnumberdou...	0.0	[,what,number,...	(262144,[15889,33...	(262144,[15889,33...	62144,[15889,33	[23.98...	[0.99...	0.0

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

```
val tokenizer = new Tokenizer()
                   .setInputCol(“message")
                   .setOutputCol("tokens")
val tokdata = tokenizer.transform(indexed)
val hashingTF = new HashingTF()
                   .setInputCol(“tokens")
                   .setOutputCol("tf")
val tfdata = hashingTF.transform(tokdata)
val idf = new IDF()
                .setInputCol(“tf")
                .setOutputCol("idf")
val idfModel = idf.fit(tfdata)
val idfdata = idfModel.transform(tfdata)
```

Tokenize/TF/IDF the message

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

val assemdata = assembler.transform(idfdata)

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

VectorAssembler & Split

```
val lr = new LogisticRegression()
  .setLabelCol("label")
  .setFeaturesCol("features")
val lrModel = lr.fit(trainingData)
val predict = lrModel.transform(testData)
```

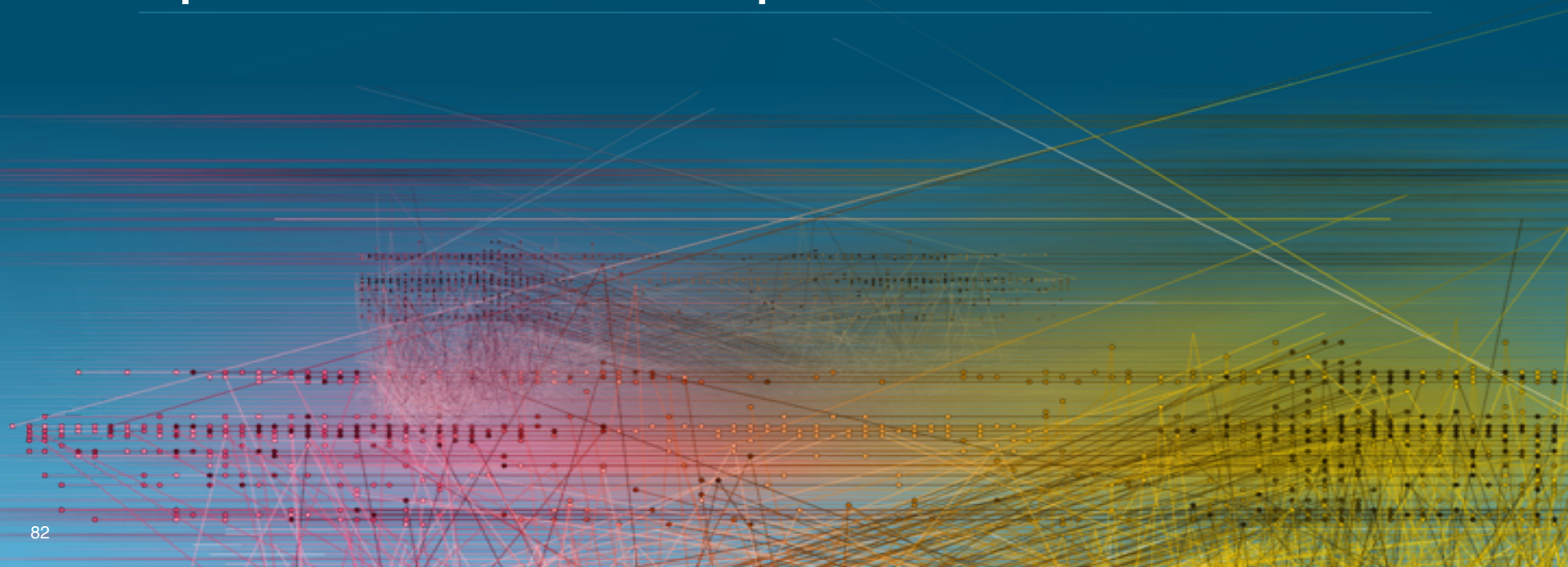
Train, Predict

Logistic Regression

Code Walk Through

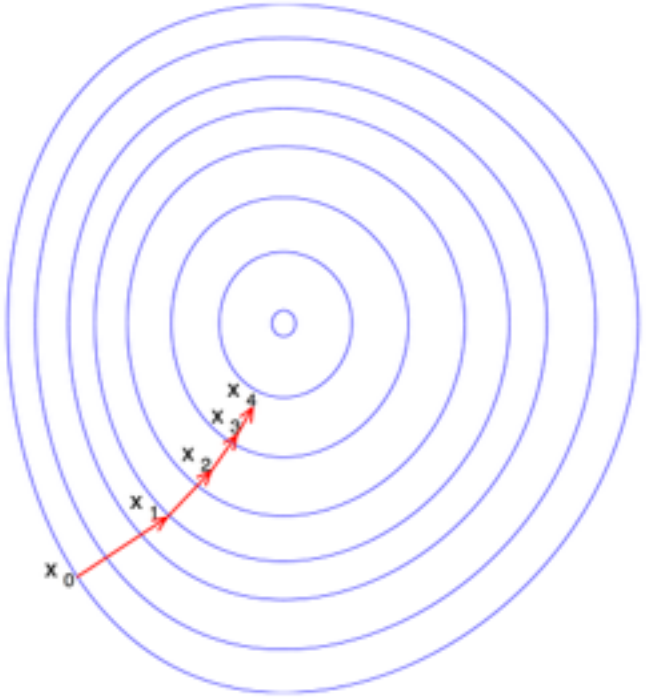
cloudera
Ask Bigger Questions

Optimization Techniques



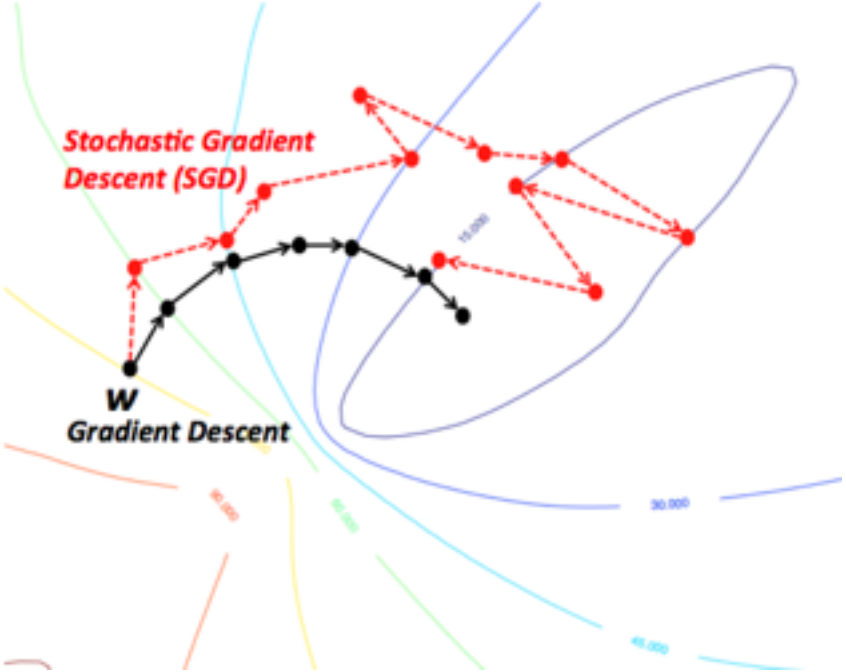
Gradient Descent

- It is a first order iterative optimization algorithm.
- Takes steps proportional to the *negative* of the gradient of the function at the current point.
- The gradient descent is relatively slow close to the minimum.
- For poorly conditioned convex problems, gradient descent increasingly 'zigzags' as the gradients point nearly orthogonally to the shortest direction to a minimum point.



Stochastic Gradient Descent

- Also known as **incremental** gradient descent, is a stochastic approximation of the gradient descent optimizationmethod for minimizing an objective function that is written as a sum of differentiable functions.
- The implementation can make use of vectorization libraries rather than computing each step separately



Limited Memory BFGS

- It is a variation of the BFGS method.
- Only uses a few vectors to represent the approximation of Hessian matrix implicitly.
- Requires Less memory.
- Well suited for optimization problems with a less number of variables.
- The Hessian matrix is approximated by previous gradient evaluations, so there is no vertical scalability issue (the number of training features) unlike computing the Hessian matrix explicitly in Newton's method.
- Converges faster

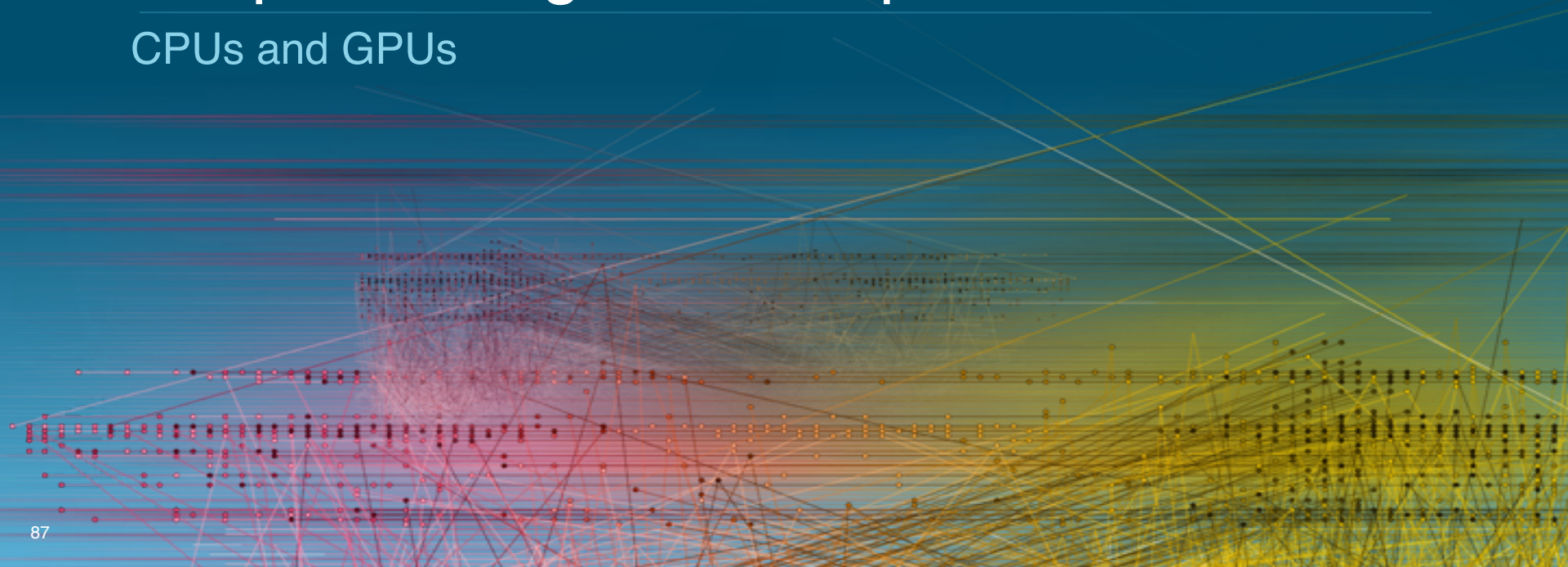
What to use

- In general, when L-BFGS is available, it is recommended using it instead of SGD since L-BFGS tends to converge faster (in fewer iterations).
- L-BFGS is currently only a low-level optimization primitive in MLlib. If you want to use L-BFGS in various ML algorithms such as Linear Regression, and Logistic Regression, you have to pass the gradient of objective function, and updater into optimizer yourself instead of using the training APIs like LogisticRegressionWithSGD.

cloudera
Ask Bigger Questions

Deep Learning in Hadoop

CPUs and GPUs



Multilayer Perceptron Classifier

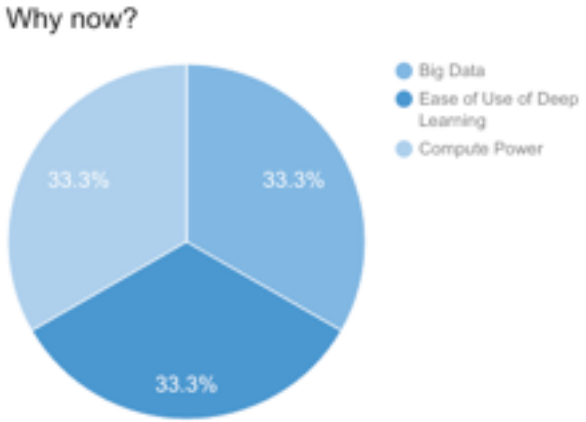
- Multilayer perceptron classifier (MLPC) is a classifier based on the feedforward artificial neural network.
- MLPC consists of multiple layers of nodes.
- Each layer is fully connected to the next layer in the network.
- Nodes in the input layer represent the input data.
- All other nodes map inputs to outputs by a linear combination of the inputs with the node's weights w and bias b and applying an activation function.
- MLPC employs backpropagation for learning the model.

Deep Learning in Hadoop

- However, a major source of difficulty in many real-world artificial intelligence applications is that many of the factors of variation influence every single piece of data we can observe.]
- Deep learning solves this central problem via representation learning by introducing representations that are expressed in terms of other, simpler representations.

Deep Learning in Hadoop

- However, traditional machine learning and feature engineering algorithms are not efficient enough to extract complex and nonlinear patterns, which are hallmarks of the big data.
- Deep Learning, on the other hand, solves this central problem via representation learning by introducing representations that are expressed in terms of other, simpler representations.
- Many approachable and easy to use Deep Learning Frameworks available.
- Compute power



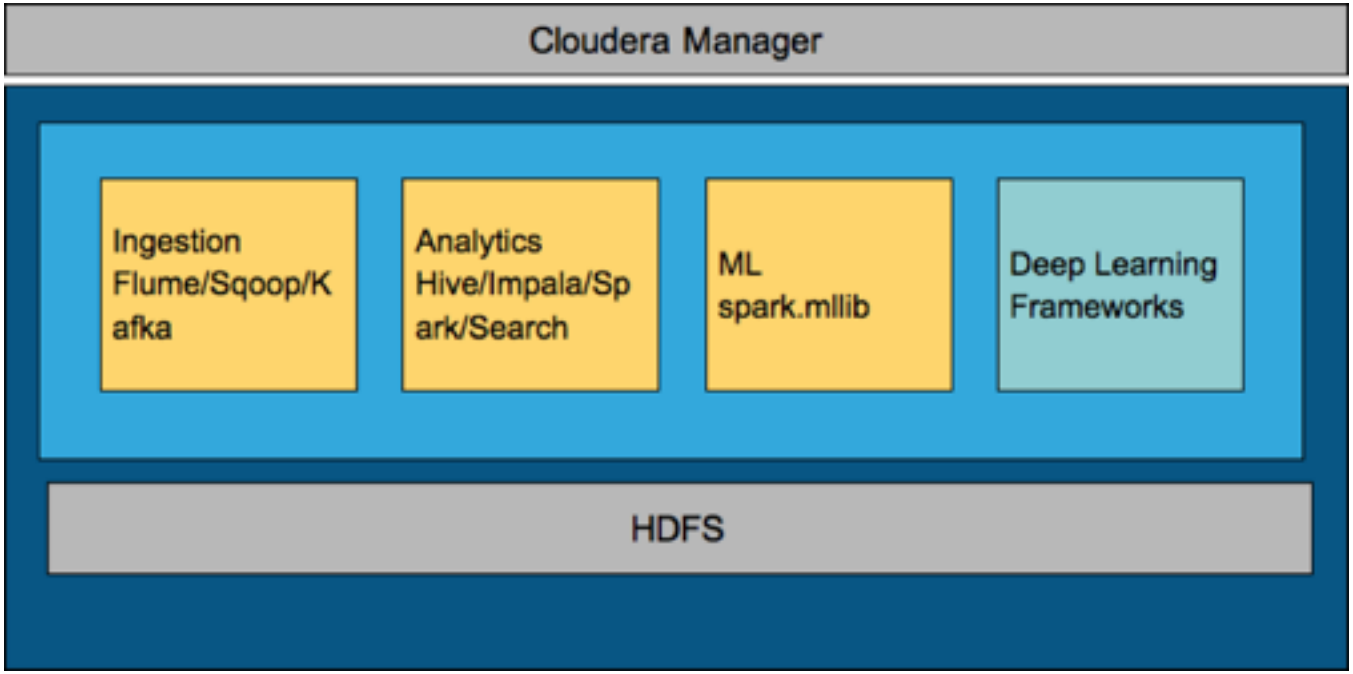
Deep Learning in Hadoop

- Deep Neural Networks (DNNs) require large amounts of computation
- Inherently parallel in nature
- Naturally translates to efficient computations to GPUs
- Accelerated primitive libraries further boost performance. (cuDNNs and MKL)

GPUs and Shared Libraries

- We give steps to leverage cuDNN libraries if you plan to run your DNN workloads on GPU enabled CDH cluster. In brief, you would need to download your respective NVIDIA driver, download and install the CUDA toolkit and then install cuDNN libraries. The frameworks that we discuss below can take advantage of both CPU and or GPU devices.
- Be mindful of the cuDNN versions between Tensorflow and Caffe.
- https://github.com/WhiteFangBuck/CDSW-DL/tree/master/scripts

Deep Learning in Hadoop



- http://blog.cloudera.com/blog/2017/04/deep-learning-frameworks-on-cdh-and-cloudera-data-science-workbench/
- http://blog.cloudera.com/blog/2017/04/bigdl-on-cdh-and-cloudera-data-science-workbench/

TensorFlowOnSpark

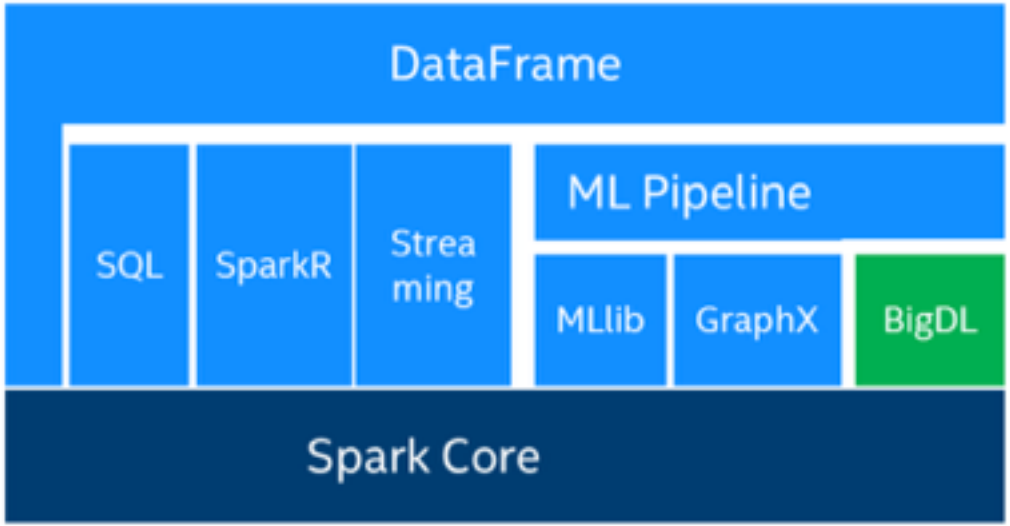
- In sequence, Google releases Tensorflow, enhanced distributed deep learning capabilities in Tensorflow, and then support for HDFS Support
- Supports direct Tensor communication between processes.
- Scales easily by adding more machines
- Tensorflow ingests data using QueueRunners or feed_dict. Does not leverage Spark for data ingestion.

CaffeFlowOnSpark

- Caffe is a Deep Learning Framework from Berkley Vision Lab implemented in C++ where models and optimizations are defined as plaintext schemas instead of code. It has a command line as well as a Python interface and has been widely adopted especially for vision related tasks.
- Yahoo released a Spark interface for Caffe which gives you the ability to run the DNN model within the same cluster where your ingested data and other analytical frameworks reside, conforming to the company wide security and governance policies.

BigDL

- Open Source Deep Learning framework for Apache Spark*
- **Easy** Customer and Developer Experience
- High Performance & Efficient Scale out leveraging Spark architecture
- **Feature Parity** with Caffe, Torch etc.



Github: github.com/intel-analytics/BigDL

http://software.intel.com/ai

Python/PySpark

- When you want a specific version of python or related libraries, it is a very good idea to use an environment maintained by Conda. System environments and utilities are very much tied to the default python which ships with the OS and things may break and stop functioning if tinkered with.
- In our test cluster, the default CentOS 7.2 image shipped with Python 2.5. We instead installed Anaconda
- Use Spark to distribute Python
- HighUse Spark Environment variables

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!!!!! :)

16:35pm–17:15pm Thursday, May 25, 2017

Ask me anything: Unraveling data with Spark using machine learning

Ask Me Anything

Location: Capital Suite 7

Vartika Singh (Cloudera), Jeffrey Shmain (Cloudera)

Thank you