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 14 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.

©2014 Cloudera, Inc. All rights reserved.

Agenda

- Environment Setup
- Spark Apps 101
- Spark ML Extractors
- Spark ML Transformers
- Hyperparameter Tuning
- Spark ML Algorithms
- Deep Learning
- Q&A

Set up the notebook

- Point your browser to
  - http://cdsw.strata-workshop-cloudera.com/
  - We will walk you through the rest

©2014 Cloudera, Inc. All rights reserved.

Download & Install - if you like the shell

CDSW	Spark Shell
http://cdsw.strata-workshop-cloudera.com/	▪ http://spark.apache.org/downloads.html ▪ spark-shell --driver-memory 2G --executor-memory 2G --num-executors 2 --executor-cores 2

Optional : https://github.com/WhiteFangBuck/strata-singapore-2017

©2014 Cloudera, Inc. All rights reserved.

Get the code - once you are logged in

- Get the code:

After you create the project:

cp -r /usr/local/strata .

©2014 Cloudera, Inc. All rights reserved.

“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.”

©2014 Cloudera, Inc. All rights reserved.

cloudera
Ask Bigger Questions

Spark, Spark ML

- There are two Spark Machine Learning libraries: ML (DataFrame based) and MLLib(RDD based - 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 the RDD-based API will be deprecated.
- The RDD-based API is expected to be removed in Spark 3.0.

©2014 Cloudera, Inc. All rights reserved.

A overview of ML pipeline

cloudera

© Cloudera, Inc. All rights reserved.

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. 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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

Determining Memory Usage

- Use Cache
- SizeEstimator.estimate

©2014 Cloudera, Inc. All rights reserved.

Tuning Data Structures

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

©2014 Cloudera, Inc. All rights reserved.

Other

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

©2014 Cloudera, Inc. All rights reserved.

cloudera
Ask Bigger Questions

Invoking third party libraries

Apache Spark

- The Spark storage abstraction called Resilient Distributed Datasets (RDDs) is essentially a distributed fault-tolerant vector on which programmers can perform a subset of operations expected from a regular local vector.
- RDDs permit user-defined data partitioning

©2014 Cloudera, Inc. All rights reserved.

©2014 Cloudera, Inc. All rights reserved.

```
def convertFunc (
    file: (String:
            org.apache.spark.input.PortableDataStream) )
     : Unit = {
       ..
    ImageIO.write(
    x.asInstanceOf[RenderedImage], "png", imageByteStream )
    ..
    val api: TessBaseAPI = new TessBaseAPI( )
    ..
}
```


©2014 Cloudera, Inc. All rights reserved.

ML Pipelines

©2014 Cloudera, Inc. All rights reserved.

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 storing text, feature vectors, true labels, and predictions.

©2014 Cloudera, Inc. All rights reserved.

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
- Immutable

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

Storage and Retrieval

- Often times it is worth it to save a model or a pipeline to disk for later use.
- In Spark 1.6, a model import/export functionality was added to the Pipeline API.
- Most basic transformers are supported as well as some of the more basic ML models.
- Please refer to the algorithm's API documentation to see if saving and loading is supported.

©2014 Cloudera, Inc. All rights reserved.

Transformers

StringIndexer, IndexToString, OneHotEncoder, VectorIndexer, Tokenizer, StopWordsRemover

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.

©2014 Cloudera, Inc. All rights reserved.

StringIndexer

```
id | category
----|----------
 0  | a
 1  | b
 2  | c
 3  | a
 4  | a
 5  | c
```

```
id | category | categoryIndex
----|----------|---------------
 0  | a        | 0.0
 1  | b        | 2.0
 2  | c        | 1.0
 3  | a        | 0.0
 4  | a        | 0.0
 5  | c        | 1.0
```


©2014 Cloudera, Inc. All rights reserved.

StringIndexer - Unseen Labels

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

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


©2014 Cloudera, Inc. All rights reserved.

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

```
id | categoryIndex
----|---------------
 0  | 0.0
 1  | 2.0
 2  | 1.0
 3  | 0.0
 4  | 0.0
 5  | 1.0
```

→

```
.setInputCol("categoryIndex")

.setOutputCol("originalCat")
```

→

```
id | categoryIndex | originalCat
----|---------------|------------
 0  | 0.0           | a
 1  | 2.0           | b
 2  | 1.0           | c
 3  | 0.0           | a
 4  | 0.0           | a
 5  | 1.0           | c
```


©2014 Cloudera, Inc. All rights reserved.

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.

id	category
0	a
1	b
2	c
3	a
4	a
5	c

```
new OneHotEncoder()
  .setInputCol("categoryIndex")
  .setOutputCol("categoryVec")
  .transform(indexed)
```

id	category	categoryVec
0	a	(2,[0],[1.0])
1	b	(2,[],[])
2	c	(2,[1],[1.0])
3	a	(2,[0],[1.0])
4	a	(2,[0],[1.0])
5	c	(2,[1],[1.0])

©2014 Cloudera, Inc. All rights reserved.

VectorIndexer

- `VectorIndexer` helps index categorical features in datasets of `Vector`s.
- 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)
```


©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

StopWordsRemover

```
new StopWordsRemover()
  .setInputCol(“raw”).stopWords(<Array>)
  .setOutputCol("filtered")
  .caseSenstive(false)
  .transform(sentenceDataFrame)
```

- StopWords are the words which should be excluded from the input,
- Typically because the words appear frequently and don’t carry as much meaning.

©2014 Cloudera, Inc. All rights reserved.

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.
- n

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

A few others

- `Binarizer`
- `PCA`
- `Polynomial Expansion`
- `Discrete Cosine Transform`
- Bucketizer
- QuantileDiscretizer
- StandardScaler, MinMaxScaler, MaxAbsScaler

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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 document

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

Code walk through - Q&A

Code walk through - Bonus Exercise

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

©2014 Cloudera, Inc. All rights reserved.

Matrix Operations - Algorithms

Vectors and Matrices

- Spectral programs: Singular Value Decomposition (SVD) and PCA
- Convex programs: Gradient Descent, LBFGS, Accelerate Gradient, and other unconstrained optimization methods

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

Model Selection and Tuning

Model Selection

Using data to find the best model or parameters for a given task.

©2014 Cloudera, Inc. All rights reserved.

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
  - Estimator -> algorithm/pipeline 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
  - TrainValidation split
    - Evaluate combination of parameters only once.
    - Faster but less reliable
  - CrossValidation
    - Goes over multiple folds of data.
    - Slower but more reliable

©2014 Cloudera, Inc. All rights reserved.

TrainValidationSplit

label	features
-9.49	(10 [0 1 2 3 4 5 6 7 8 9] [0.45 0.36 -0.38 -0.44 0.33 ….]
0.25	(10 [0 1 2 3 4 5 6 7 8 9] [0.83 -0.12 0.50 -0.22 -0.64 …]
-4.43	(10 [0 1 2 3 4 5 6 7 8 9] [0.50 0.14 0.16 0.50 -0.93 …]
-19.78	(10 [0 1 2 3 4 5 6 7 8 9] [-0.03 -0.41 0.90 0.64 0.27 …]
-7.96	(10 [0 1 2 3 4 5 6 7 8 9] [-0.06 0.65 -0.69 0.66 -0.07 …]
-7.89	(10 [0 1 2 3 4 5 6 7 8 9] [-0.15 0.26 0.40 -0.36 0.14 …]
-8.46	(10 [0 1 2 3 4 5 6 7 8 9] [0.39 0.81 -0.60 0.61 0.25 …]
2.12	(10 [0 1 2 3 4 5 6 7 8 9] [-0.00 -0.94 -0.92 -0.03 0.31 …
1.07	(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	(1,0,false)	(0.1,0,false)	(0.01,0,false)	FALSE
0.25	(1,0.25,false)	(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,false)	(0.1,0.75,false)	(0.01,0.75,false)	

↓

org.apache.spark.ml.regression.LinearRegression

↓

org.apache.spark.ml.evaluation.RegressionEvaluator

↓

©2014 Cloudera, Inc. All rights reserved.

cloudera
Ask Bigger Questions

Parameters: Model tuning

```
new TrainValidationSplit()
 .setEstimator(new LinearRegression)
 .setEvaluator(new RegressionEvaluator)
 .setEstimatorParamMaps(paramGrid)
 .setTrainRatio(0.8)
```

```
new CrossValidator()
   .setEstimator(new LogisticRegression)
   .setEvaluator(new BinaryClassificationEvaluator)
   .setEstimatorParamMaps(paramGrid)
   .setNumFolds(2) // Use 3+ in practice
```


©2014 Cloudera, Inc. All rights reserved.

Code walk through

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

©2014 Cloudera, Inc. All rights reserved.

Clustering - Kmeans

1. Choose K number of Centroids
2. Assign each object to the group that has the closest centroid.
3. Recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move.

©2014 Cloudera, Inc. All rights reserved.

cloudera
Ask Bigger Questions

Tuning parameters: org.apache.spark.ml.clustering.KMeans

```
new KMeans()
  .setK(20)
  .setFeaturesCol(“my_features")
  .setPredictionCol(“clusterId”)
  .setTol(0.002)
  .setMaxIter(10)
  .setSeed(1000)
```

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

©2014 Cloudera, Inc. All rights reserved.

Clustering - Bisecting Kmeans

1. Start the whole dataset as a single cluster

2. Splits each cluster into 2 sub clusters on each iteration - bisecting step
   - Using classical kmeans with K = 2

©2014 Cloudera, Inc. All rights reserved.

Tuning parameters: org.apache.spark.ml.clustering.BisectingKMeans

```
new BisectingKMeans()
  .setK(10)
  .setFeaturesCol("my_features")
  .setPredictionCol("clusterId")
  .setMaxIter(10)
  .setSeed(1000)
  .setMinDivisibleClusterSize(10)
```

featuresCol - dataframe column that specifies features
k - the desired number of leaf clusters. Default 4
maxIter - maximum number of iterations
predictionCol - dataframe column that will have cluster ID prediction
minDivisibleClusterSize - minimum number of points of a divisible cluster

©2014 Cloudera, Inc. All rights reserved.

Clustering - Gaussian Mixture Model

1. K-means does hard assignments to clusters

2. GaussianMixture class performs **expectation maximization** for multivariate Gaussian Mixture Models

3. GMM gives us the ability to do probabilistic assignments

4. GMM uses weights, means and variances to determine the clusters

©2014 Cloudera, Inc. All rights reserved.

Tuning parameters: org.apache.spark.ml.clustering.GaussianMixture

```
val gem = new GaussianMixture()
  .setK(20)
  .setFeaturesCol("my_features")
  .setPredictionCol("clusterId")
  .setProbabilityCol("my_probs")
  .setTol(0.002)
  .setMaxIter(10)
  .setSeed(1000)
```

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
probabilityCol - Column name for predicted class conditional probabilities

©2014 Cloudera, Inc. All rights reserved.

Why LDA?

- What is document discussing?
- How similar are documents?
- If I am interested in a topic, which documents should I read first

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

How LDA Works

Assumption

Algorithm

$$P(Z|W,D) = \frac{\#\ of\ word\ W\ in\ topic\ Z + \beta_w}{total\ tokens\ in\ Z + \beta} * (\#\ words\ in\ D\ that\ belong\ to\ Z + \alpha)$$

https://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/

©2014 Cloudera, Inc. All rights reserved.

Tuning parameters: org.apache.spark.ml.clustering.LDA

```
val lda = new LDA()
  .setOptimizer("online")
  .setK(10)
  .setMaxIter(15)
  .setCheckpointInterval(-1)
  .setDocConcentration(Array(0.2,0.3))
  .setFeaturesCol("my_features")
  .setSeed(3000)
  .setSubsamplingRate(0.5)
  .setTopicConcentration(1.5)
  .setTopicDistributionCol("result")
```

featuresCol - dataframe column that specifies features
k - number of clusters to create. Default: 2
maxIter - maximum number of iterations
tol - convergence tolerance
checkpointInterval - param for set checkpoint interval
docConcentration - concentration parameter for the prior placed on documents' distributions over topics
optimizer - optimizer to estimate the LDA model (EM or online)
subsamplingRate - fraction of the corpus to be sampled and used in each iteration of mini-batch
topicConcentration - concentration parameter for the prior placed on topics' distributions over terms
topicDistributionCol - output column with estimates of the topic mixture distribution for each document

©2014 Cloudera, Inc. All rights reserved.

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

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

©2014 Cloudera, Inc. All rights reserved.

LDA Code WalkThrough

Data Source: NewsGroup

LDA

- experiment with cluster numbers and vocabulary size
- Read files into data frame
- RegTokenizer
- StopWordsRemover
- NGram
- CountVectorizer
- LDA
- Print Clusters

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

Tuning parameters: org.apache.spark.ml.clustering.LinearRegression

```
val gem = new LinearRegression()
  .setElasticNetParam(1)
  .setFeaturesCol("my_features")
  .setFitIntercept(true)
  .setLabelCol("price")
  .setMaxIter(100)
  .setPredictionCol("predicted")
  .setRegParam(0.1)
  .setSolver("l-bfgs")
  .setStandardization(true)
  .setTol(0.001)
  .setWeightCol("weights")
```

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

©2014 Cloudera, Inc. All rights reserved.

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	prefareaI
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])

©2014 Cloudera, Inc. All rights reserved.
cloudera

Linear Regression - Feature Extraction

id	price	lotsize	bedrooms	athrm	stories	drivewayr	recroom	ullbase	gashw	airco	garagepl	prefarea	drivewayIdx	recroomIdx	fullbaseIdx	gashwIdx	aircoIdx	efareaId	drivewayV	recroomV	fullbaseV	gashwV	airco	prefareaV
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	0.0	(1,[0],[1.0])	(1,[0],[1.0])	(1,[],[])	(1,[0],[1.0])	[0],[1.0	[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	0.0	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	[0],[1.0	[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	0.0	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	(1,[0],[1.0])	[0],[1.0	[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...

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

Vector Assembler assembles:

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

©2014 Cloudera, Inc. All rights reserved.

Classification

Logistic Regression, Random Tree Classifier, Naive 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

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

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 (confidence)
threshold - threshold in binary classification prediction
thresholds - thresholds in multi-class classification to adjust the probability of predicting each class

©2014 Cloudera, Inc. All rights reserved.

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

©2014 Cloudera, Inc. All rights reserved.

Logistic Regression

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

©2014 Cloudera, Inc. All rights reserved.

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.

©2014 Cloudera, Inc. All rights reserved.

LogisticRegression - Feature Extraction

spam	message
ham	<DECIMAL>...
ham	gonnaletmekno...
ham	whatnumberdou...

org.apache.spark.ml.feature.Tokenizer

spam	message	label	tokens
ham	<DECIMAL>...	0.0	[,<decimal>...
ham	gonnaletmekno...	0.0	[,gonna,let,me...
ham	whatnumberdou...	0.0	[,what,number,...

org.apache.spark.ml.feature.IDF

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...

©2014 Cloudera, Inc. All rights reserved.
cloudera
Ask Bigger Questions

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	rawPrediction	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

©2014 Cloudera, Inc. All rights reserved.
cloudera

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

©2014 Cloudera, Inc. All rights reserved.

Logistic Regression

Code Walk through

Big Data and DNN

Environment set up - Start a Python session

- cp -r /usr/local/strata .
- pip install keras

©2014 Cloudera, Inc. All rights reserved.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

©2014 Cloudera, Inc. All rights reserved.

A overview of ML pipeline

cloudera

© Cloudera, Inc. All rights reserved.

Deep Learning in Big Data

- 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.

©2017 Cloudera, Inc. All rights reserved.

DL at Scale

Scaling Out and Up

- Using multiple machines in a large cluster
- Leveraging graphics processing units (GPUs).

©2017 Cloudera, Inc. All rights reserved.

Scaling-Out

- Frameworks have evolved that enable model parallelism
  - Within a machine (via multithreading)
  - Across machines (via message passing).

Details of parallelism, synchronization and communication managed by the framework.

©2017 Cloudera, Inc. All rights reserved.

Data Parallelism

Same task operates on different data sets across multiple nodes.

©2017 Cloudera, Inc. All rights reserved.

Model Parallelism

When the model is partitioned across several machines, so that responsibility for the computation for different nodes is assigned to different machines.

©2017 Cloudera, Inc. All rights reserved.

JOBS I'VE BEEN FIRED FROM

FEDEX DRIVER
CRANE OPERATOR
SURGEON
AIR TRAFFIC CONTROLLER
PHARMACIST
MUSEUM CURATOR
WAITER
DOG WALKER
OIL TANKER CAPTAIN
VIOLINIST
MARS ROVER DRIVER
MASSAGE THERAPIST

©2017 Cloudera, Inc. All rights reserved.
cloudera
Ask Bigger Questions

GPUs

- The use of GPUs is a significant advance in recent years that makes the training of modestly sized deep networks practical.
- A known limitation of the GPU approach is that the training speed-up is small when the model does not fit in GPU memory (typically less than few gigabytes).
- To use a GPU effectively, practitioners often reduce the size of the data or parameters so that CPU-to-GPU transfers are not a significant bottleneck.

©2017 Cloudera, Inc. All rights reserved.

Scaling-out, GPUs and Scaling-up

The data, workloads or a combination of both, are partitioned into worker machines, which access the globally shared model as they simultaneously perform local computations to refine the model.

©2017 Cloudera, Inc. All rights reserved.

Distributed computation in neural networks Model Parallelism

- In model parallelism, different machines in the distributed system are responsible for the computations in different parts of a single network - for example, each layer in the neural network may be assigned to a different machine.

- Parallelizing the gradient computations with “model parallelism”, where each GPU is responsible for only a piece of the whole neural network, reduces band- width requirements considerably but also requires frequent synchronization

©2017 Cloudera, Inc. All rights reserved.

Model Parallelism

- This approach works well for GPUs in a single server (which share a high- speed bus) (Krizhevsky, 2010; Krizhevsky et al., 2012) but is still too inefficient to be used with Ethernet networks.
- Use of high- end networking infrastructure to remove the communications bottleneck between servers and enable us to exploit both fast GPU computation and to “scale out” to many servers.

©2017 Cloudera, Inc. All rights reserved.

©2017 Cloudera, Inc. All rights reserved.

Data parallelism

- Employ the GPUs in a “data parallel” mode
- Each GPU keeps a complete copy of the neural network parameters
- But computes a gradient using a different subset of the training data

©2017 Cloudera, Inc. All rights reserved.

Data parallelism

- Data parallel training approaches all require some method of combining results and synchronizing the model parameters between each worker.
  - Parameter averaging vs gradient based approaches
  - Sync vs Asynch
  - Centralized vs Distributed

©2017 Cloudera, Inc. All rights reserved.

Data parallelism

- With parameter averaging, training proceeds as follows:
  - Initialize the network parameters randomly based on the model configuration
  - Distribute a copy of the current parameters to each worker
  - Train each worker on a subset of the data
  - Set the global parameters to the average the

©2017 Cloudera, Inc. All rights reserved.

Data parallel - issues

- Each GPU keeps a complete copy of the neural network parameters

- GPU code is capable of computing a gradient for these parameters in just milliseconds per training image, yet copying parameters or gradients to other machines will take at least 8 seconds over commodity Ethernet—several orders of magnitude slower.

©2017 Cloudera, Inc. All rights reserved.

Frameworks

Caffe2

- A veteran
- Leader in Computer Vision
- Obscure documentation
- Synchronous SGC and Data parallel

©2017 Cloudera, Inc. All rights reserved.

Caffe2 - Synch SGD

- Data parallel
- Using 8 GPUS to run a batch of 32 each is equivalent to one GPU running a mini-batch of 256.

©2017 Cloudera, Inc. All rights reserved.

Tensorflow - Synch and Asynch SGD

- Data parallel
  - Synchronous SGD
  - Asynchronous SGD
- Model parallel
- Concurrent Steps for Model computation in a pipeline

©2017 Cloudera, Inc. All rights reserved.

CaffeOnSpark

- 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.

© Cloudera, Inc. All rights reserved.

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.

© Cloudera, Inc. All rights reserved.

DL4J

- Support Apache Spark (1 and 2) for distributed training on a cluster.
- Supports data parallel synchronous parameter averaging
- Recently added support for asynchronous gradient descent.
- Use Aeron for message passing

cloudera

© Cloudera, Inc. All rights reserved.

WHAT IS BIGDL ?

- **Open Source Deep Learning framework** for **Apache Spark***
- **High Performance & Efficient Scale out** leveraging Spark architecture
- **Feature Parity** with Caffe, Torch etc.
- Efficient implementations of synchronous stochastic gradient descent (SGD) and all-reduce communications in Spark.

Github: github.com/intel-analytics/BigDL

http://software.intel.com/ai

© Cloudera, Inc. All rights reserved.

Tensorflow/Caffe on Spark

- 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.

©2017 Cloudera, Inc. All rights reserved.

DL at Scale

Transfer Learning

- *Transfer learning* involves two interrelated learning problems with the goal of using knowledge about one set of tasks to improve performance on a related task.

©2017 Cloudera, Inc. All rights reserved.

Assumption

Many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most statistical models need to be rebuilt from scratch using newly collected training data.

©2017 Cloudera, Inc. All rights reserved.

"A survey of transfer learning | SpringerLink." https://link.springer.com/article/10.1186/s40537-016-0043-6.

©2017 Cloudera, Inc. All rights reserved.

Three Points

(1)What to transfer

(2) How to transfer

(3)When to transfer.

©2017 Cloudera, Inc. All rights reserved.

Transfer Learning Approaches	Brief Description	Size of Data
Instance-Transfer	To re-weight some labeled data in the source domain for use in the target domain	
Feature-representation transfer	Find a “good” feature representation that reduces difference between the source and the target domains and the error of classification and regression models.	
Parameter transfer	Discover shared parameters or priors between the source domain and target domain models, which can benefit for transfer learning.	
Relational knowledge transfer	Build mapping of relational knowledge between the source domain and the target domains. Both domains are relational domains and i.i.d assumption is relaxed in each domain.	

©2017 Cloudera, Inc. All rights reserved.

Deciding Factors

- New dataset is small and similar to original dataset
- New dataset is large and similar to the original dataset
- New dataset is small but very different from the original dataset
- New dataset is large and very different from the original dataset.

©2017 Cloudera, Inc. All rights reserved.

Validation Loss vs Accuracy

```
…..
Epoch 37/50 - 177s - loss: 4.4437 - acc: 0.7243 - val_loss: 8.0550 - val_acc: 0.5002
Epoch 38/50 - 177s - loss: 4.3835 - acc: 0.7280 - val_loss: 7.9720 - val_acc: 0.5053
Epoch 39/50 - 177s - loss: 4.2265 - acc: 0.7376 - val_loss: 8.0542 - val_acc: 0.4984
Epoch 40/50 - 179s - loss: 2.5197 - acc: 0.7727 - val_loss: 1.0689 - val_acc: 0.8584
Epoch 41/50 - 179s - loss: 0.2653 - acc: 0.9512 - val_loss: 0.2383 - val_acc: 0.9529
Epoch 42/50 - 179s - loss: 0.1081 - acc: 0.9650 - val_loss: 0.1315 - val_acc: 0.9671
Epoch 43/50 - 179s - loss: 0.0837 - acc: 0.9693 - val_loss: 0.1154 - val_acc: 0.9744
…..
Epoch 48/50 - 179s - loss: 0.0654 - acc: 0.9746 - val_loss: 0.0897 - val_acc: 0.9758
Epoch 49/50 - 179s - loss: 0.0581 - acc: 0.9766 - val_loss: 0.0641 - val_acc: 0.9804
Epoch 50/50 - 179s - loss: 0.0470 - acc: 0.9824 - val_loss: 0.0714 - val_acc: 0.9749
```


©2017 Cloudera, Inc. All rights reserved.

©2017 Cloudera, Inc. All rights reserved.

Data Science Success Kit

Fast, easy, and secure self-service data science for the enterprise.

Machine learning is all about the data, but it's often out of reach for analytics teams working at scale. The Data Science Success Kit is the simplest way to kick-start analytics for your business. Whether it's tuning an existing analytics process, converting a Python, SAS or R model, or prototyping a new analytic result, the Data Science Success Kit is the fastest, most reliable way to get started.

The Data Science Success Kit includes:

- Cloudera Data Science Workbench (twenty seats)
- Cloudera Data Engineering (one node)
- Cloudera Data Science Professional Services Pilot (3 weeks)

©2014 Cloudera, Inc. All rights reserved.

Take aways

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

- Choices in CDH stack
  - SparkML
  - MLLib
  - Many other libraries

- Other new exciting technologies
  - Tensor flow
  - Caffe2

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

©2014 Cloudera, Inc. All rights reserved.

Thank you!!!!! :)

Vartika Singh (Cloudera), Jeffrey Shmain (Cloudera)

©2014 Cloudera, Inc. All rights reserved.

cloudera
Ask Bigger Questions

CaffeOnSpark - Demo

- 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.

©2014 Cloudera, Inc. All rights reserved.

CaffeOnSpark - Demo

- Talk about Implementation details

©2014 Cloudera, Inc. All rights reserved.

Thank you

©2014 Cloudera, Inc. All rights reserved.