基于Spark的机器学习：LDA and CNN

Yuhao Yang 杨玉皓

SSG, Big Data Technology

Agenda

- 主题模型 Topic modeling and LDA
- 在线LDA算法
- 应用和性能调优
- 深度学习
- 卷积神经网络

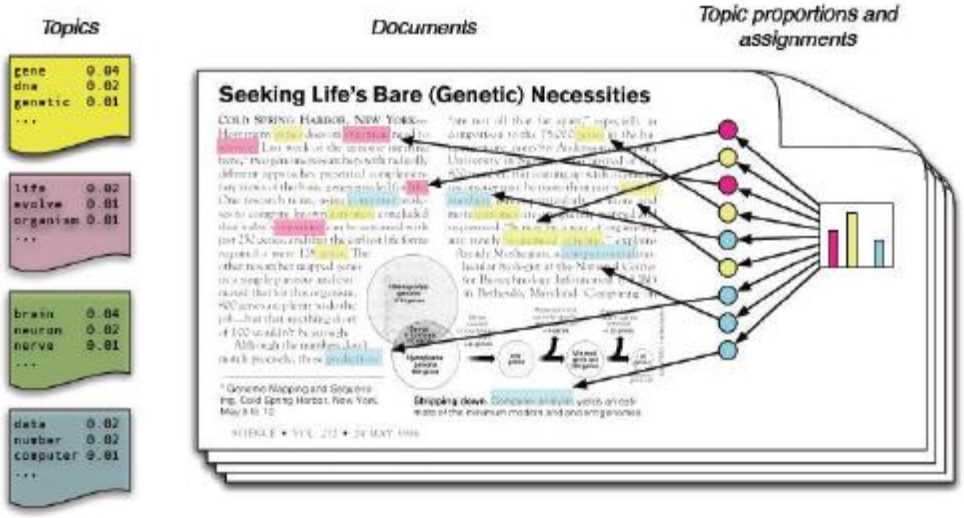
Topic Modeling

一页纸，一本书，一个图书馆，wiki，twitter…

Topic Modeling

- Automatically infers the topics discussed in a collection of documents.
- These topics can be used to summarize and organize documents, or used for featurization and dimensionality reduction
  - What is document X discussing?
  - How similar are documents X and Y?
  - If I am interested in topic Z, which documents should I read first?
- Widely applied
  - Document, image clustering
  - feature deduction
  - Social network, advertising …

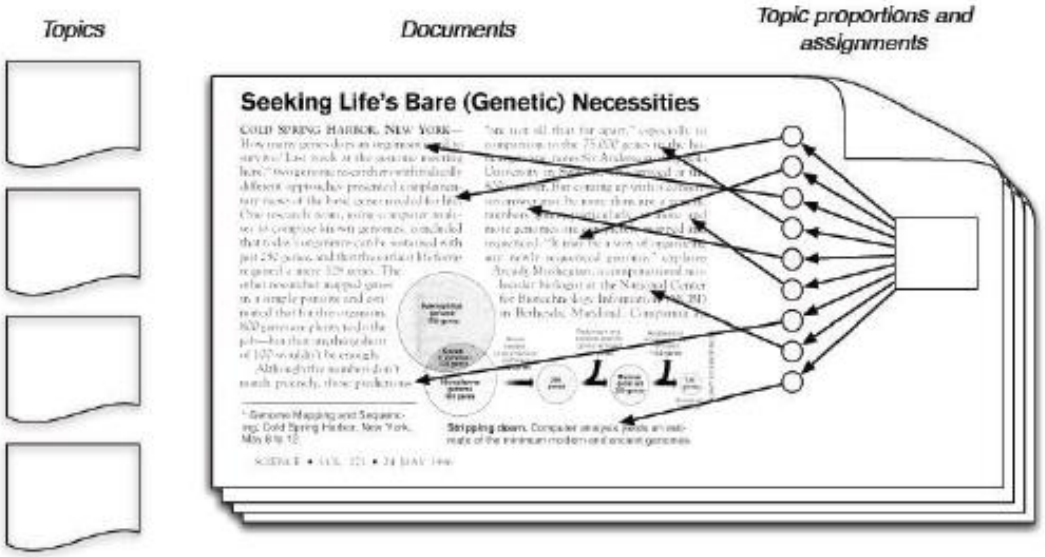
Latent Dirichlet Allocation (LDA)



(from David Blei)

- Each document is a random mixture of corpus-wide topics
- Each word is drawn from one of those topics

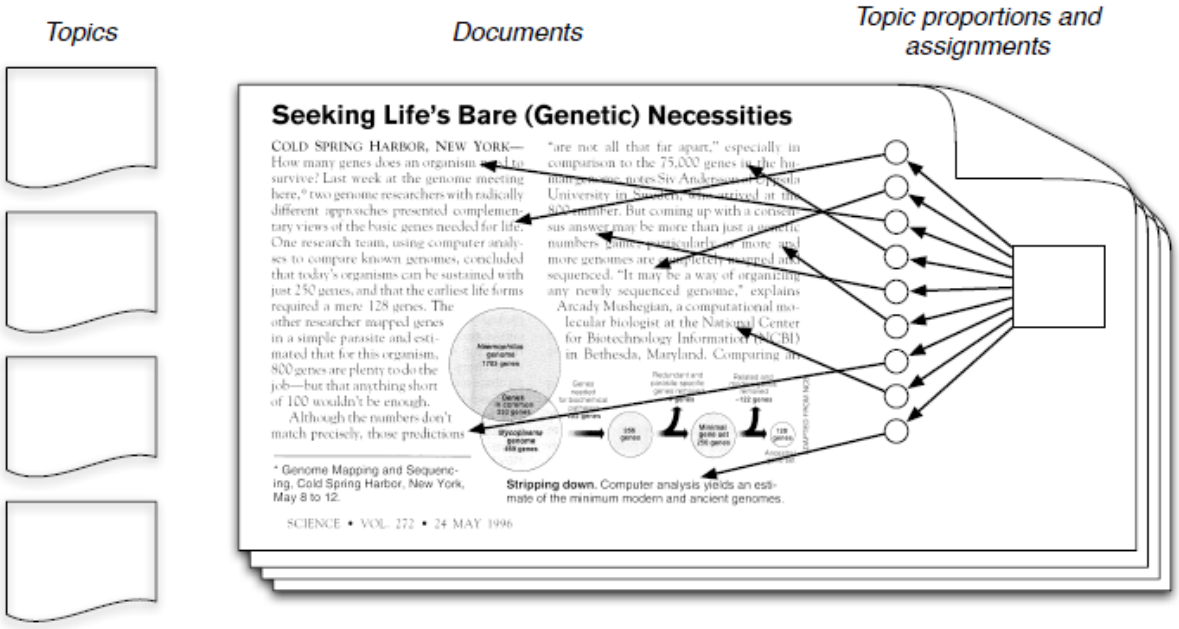
Latent Dirichlet Allocation (LDA)



(from David Blei)

- We only observe the documents
- Our goal is to **infer** the underlying topic structure

Latent Dirichlet Allocation (LDA)



(from David Blei)

- Our goal is to **infer** the hidden variables
- I.e., compute their distribution conditioned on the documents

$$p(\text{topics, proportions, assignments} \mid \text{documents})$$

Intuition of LDA

Corpus

doc1. apple banana

doc2. apple orange

doc3. banana orange

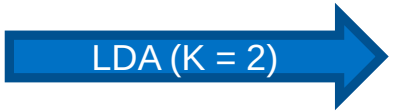
doc4. tiger cat

doc5. tiger dog

doc6. cat dog

doc7. Cat dog apple

Input (what we have)



	Topic 1	Topic2
Apple	33%	0%
Banana	33%	0%
Orange	33%	0%
Tiger	0%	33%
Cat	0%	33%
Dog	0%	33%

	Topic 1	Topic2
doc1	100%	0%
doc2	100%	0%
doc3	100%	0%
doc4	0%	100%
doc5	0%	100%
doc6	0%	100%
doc7	**33%**	**66%**

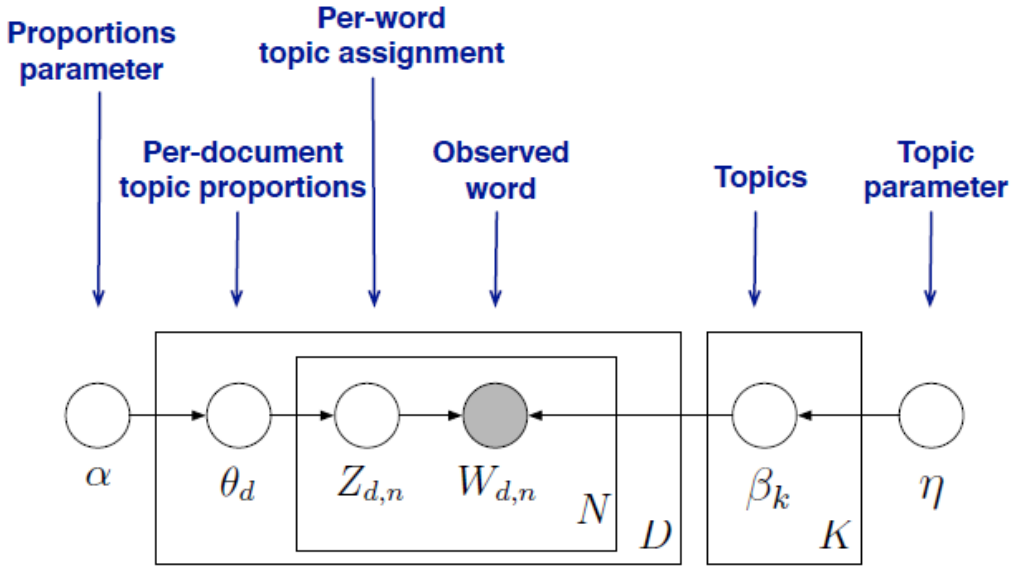
Output (what we want)

Intuition behind LDA

Run LDA on Spark document:

"Spark Core"		"GraphX"		"MLlib"		"SQL"		"Streaming"	
term	**weight**	**term**	**weight**	**term**	**weight**	**term**	**weight**	**term**	**weight**
cluster	0.014	graph	0.029	model	0.023	schema	0.012	kinesis	0.014
mesos	0.013	vertex	0.018	training	0.021	dataframe	0.01	more	0.009
driver	0.008	vertices	0.013	features	0.014	table	0.01	streaming	0.009
executor	0.008	edges	0.011	feature	0.012	hive	0.009	java	0.008
executors	0.008	graphx	0.009	load	0.01	create	0.009	dstream	0.007

Topic modeling with Latent Dirichlet Allocation



Blei, Ng, and Jordan 2003

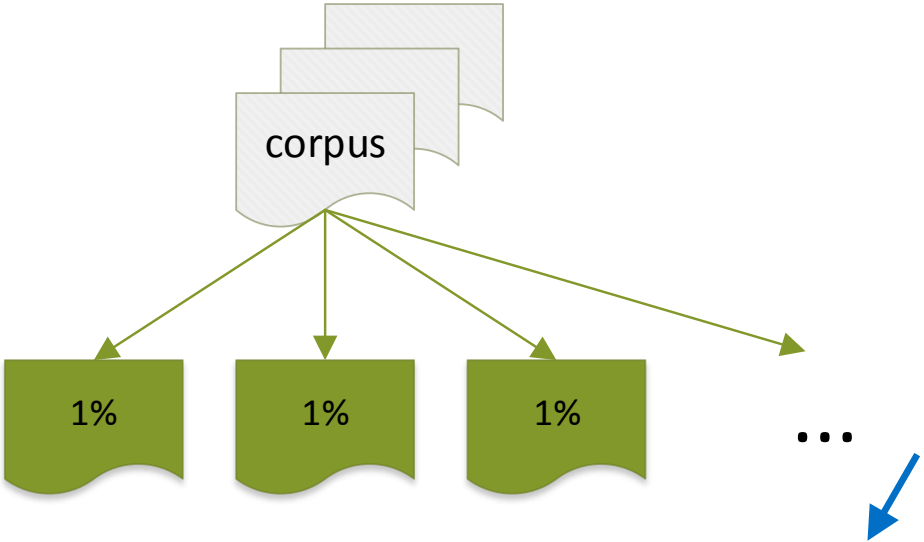
Existing Solution

- Variational EM (Expectation-maximization)
  - Numerical approximation using lower-bounds
  - Results in biased solutions
  - Convergence has numerical guarantees
- Gibss Sampling
  - Stochastic simulation
  - unbiased solutions
  - Stochastic convergence

OOM~

Both needs to load the entire corpus into memory and scan through the corpus in each iteration !!!

Make it online



Can be discarded after processing

Don’t need to load into memory in advance

Online LDA

Algorithm 2 Online variational Bayes for LDA

Define $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$
Initialize $\boldsymbol{\lambda}$ randomly.
for $t = 0$ to ∞ **do**
  E step:
  Initialize $\gamma_{tk} = 1$. (The constant 1 is arbitrary.)
  repeat
    Set $\phi_{twk} \propto \exp\{\mathbb{E}_q[\log \theta_{tk}] + \mathbb{E}_q[\log \beta_{kw}]\}$
    Set $\gamma_{tk} = \alpha + \sum_w \phi_{twk} n_{tw}$
  until $\frac{1}{K} \sum_k |\text{change in} \gamma_{tk}| < 0.00001$
  M step:
  Compute $\tilde{\lambda}_{kw} = \eta + D n_{tw} \phi_{twk}$
  Set $\boldsymbol{\lambda} = (1 - \rho_t)\boldsymbol{\lambda} + \rho_t \tilde{\boldsymbol{\lambda}}$.
end for

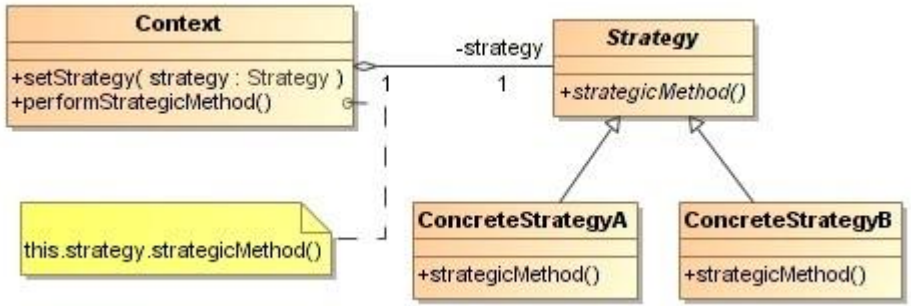
$$\rho_t \triangleq (\tau_0 + t)^{-\kappa}$$

Online LDA

- Split the corpus and processes one split at a time
- Scan the corpus only once
- As many documents as you want
- Fast and Memory-friendly
- Hoffman, Blei and Bach, "Online Learning for Latent Dirichlet Allocation." NIPS, 2010.
- (SPARK-5563)

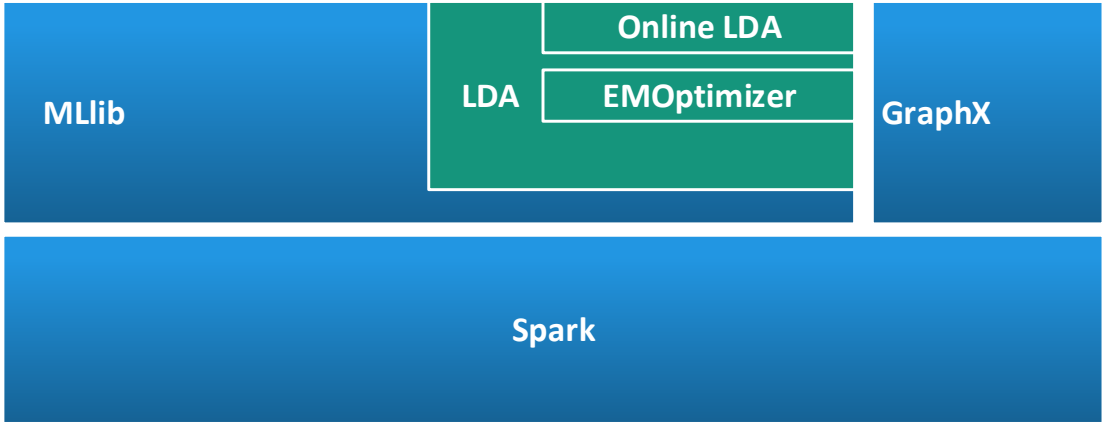
LDA in Spark Mllib 1.4

- Supports different inference algorithms via setOptimizer.
  - EMLDAOptimizer learns clustering using expectation-maximization on the likelihood function and yields comprehensive results
  - OnlineLDAOptimizer uses iterative mini-batch sampling for online variational inference and is generally memory friendly.



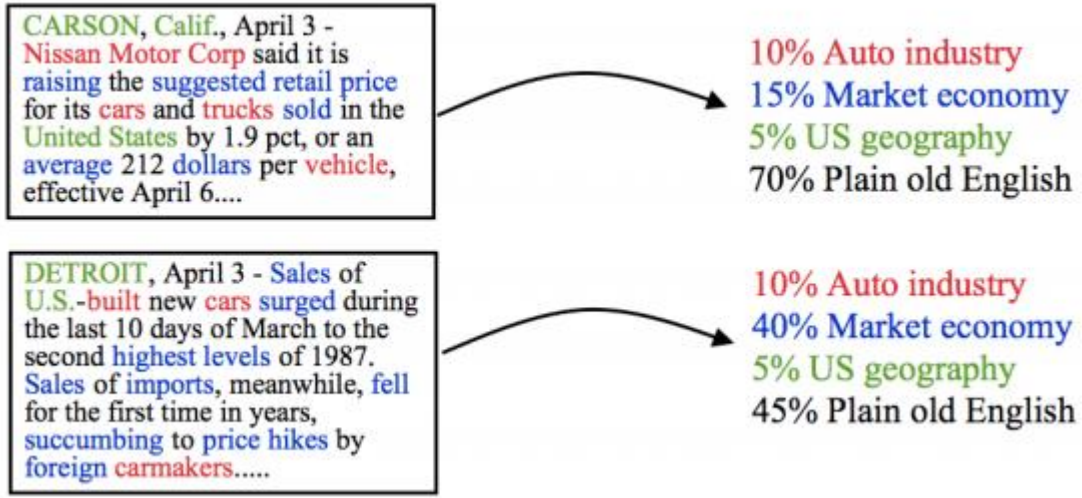
Intel play a key role

- Shape LDA to the optimizer framework (SPARK-7090)
  - Flexible structure that supports multiple algorithms
- Online LDA Optimizer (SPARK-5563)



Online Hierarchical Dirichlet Process

- We model documents as coming from an underlying set of topics.
  - Summarize documents.
  - Document/query comparisons.
  - Do not know the number of topics a priori—use DP mixtures somehow.
  - But: topics have to be shared across documents...



Hierarchical Dirichlet process (HDP)

In statistics and machine learning, the **hierarchical Dirichlet process** (HDP) is a nonparametric Bayesian approach to clustering grouped data.[1][2]

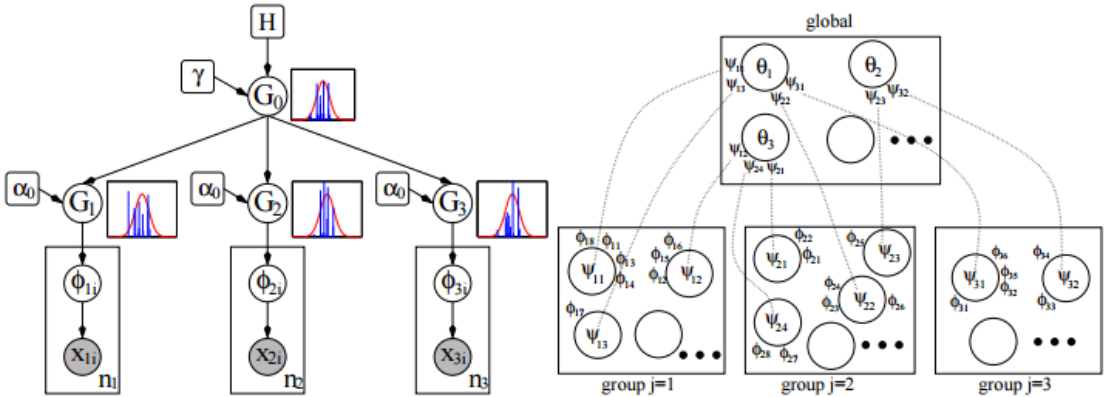
It uses a Dirichlet process for each group of data, with the Dirichlet processes for all groups sharing a base distribution which is itself drawn from a Dirichlet process. This method allows groups to share statistical strength via sharing of clusters across groups.

It was developed by Yee Whye Teh, Michael I. Jordan, Matthew J. Beal and David Blei and published in the *Journal of the American Statistical Association* in 2006.[1]

Hierarchical Dirichlet process (HDP)

The HDP mixture model is a natural nonparametric generalization of Latent Dirichlet allocation, where the number of topics can be unbounded and learnt from data.[1]

Here each group is a document consisting of a bag of words, each cluster is a topic, and each document is a mixture of topics.



Online Hierarchical Dirichlet Process

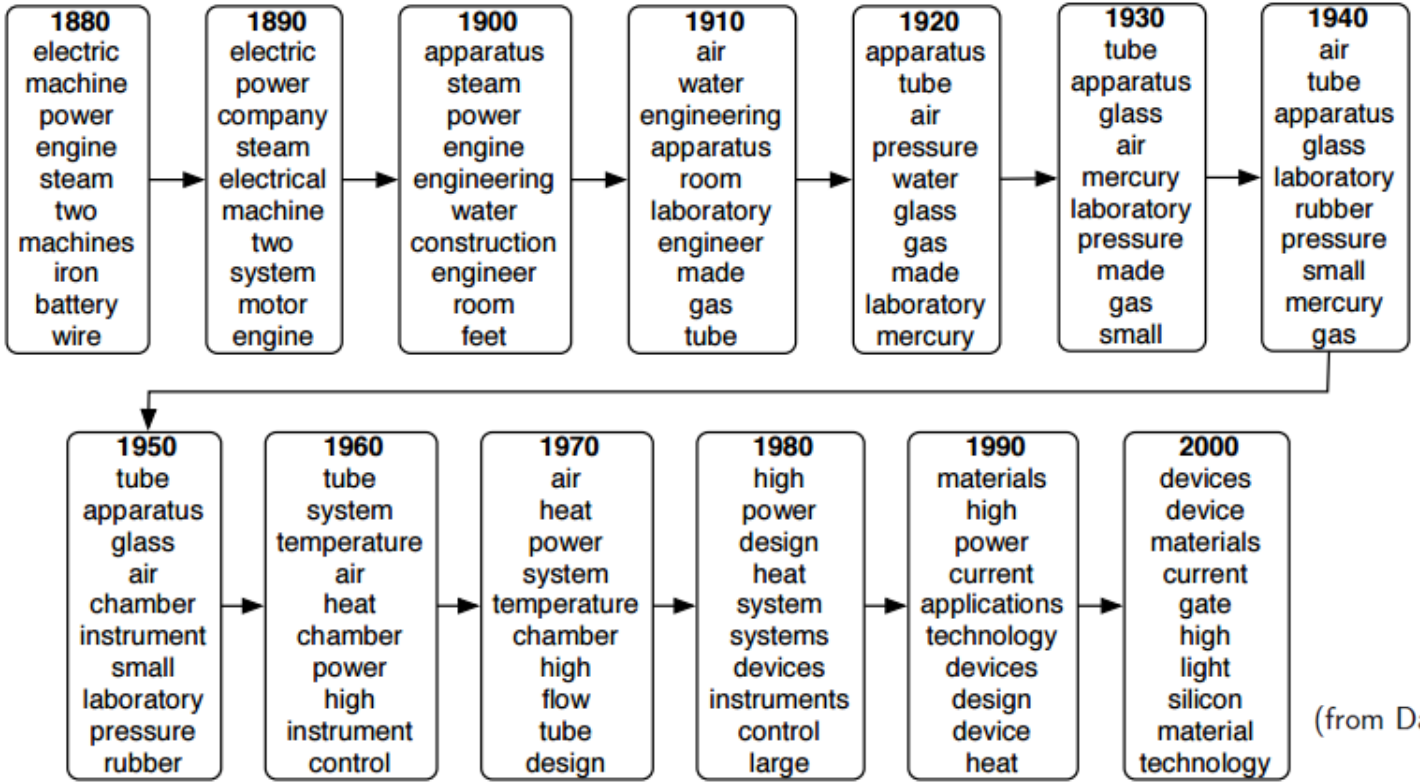
- A hierarchical, nonparametric model for clustering problems involving multiple groups of data.
  - automatically determine the appropriate number of topics
  - Comes with a price
    - Careful tuning of other hyper parameters
    - Extra complexities for running and maintenance.

Typical scenarios for OnlineLDA

- Be able to handle very large dataset
  - All the titles from StackOverflow, 8M short articles, in 15 minutes.
  - Whole English wiki, 5876K documents (avg length ~1000 words/per doc, 30G in total) in 2 hours
  - A 4-node cluster, each with 16 cores and 30G memory. Without native BLAS installed.
- Streaming, real online
  - Suitable for Social analysis

Streaming Application

Top ten for Science



(from David Blei)

Tuning experience

- For Online LDA, how to decide K
  - Experience
  - Perplexity
  - nonparametric Bayesian model

Tuning experience

- Several important parameter for Online LDA
  - S: batchSize for each split
  - κ, τ_0 : decides the weight of each iteration $\rho_t \triangleq (\tau_0 + t)^{-\kappa}$

Best parameter settings for Wikipedia corpus

S	1	4	16	64	256	1024	4096	16384
κ	0.9	0.9	0.8	0.7	0.6	0.5	0.5	0.5
τ_0	1024	1024	1024	1024	1024	1024	64	1
Perplexity	675	640	611	595	588	584	580	584

Text pre-processing

Use transformers from ML.feature

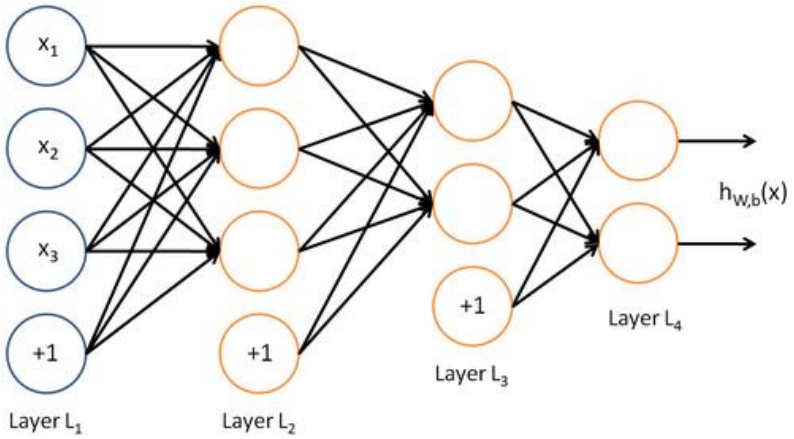
```
val tokenizer = new RegexTokenizer()
  .setInputCol("text")
  .setOutputCol("tokens")
  .setPattern("\\W+")
val cvModel = new CountVectorizerModel(dict)
  .setInputCol(tokenizer.getOutputCol)
  .setOutputCol("vector")
val pipeline = new Pipeline()
  .setStages(Array(tokenizer, cvModel))

val model = pipeline.fit(df)
val result = model.transform(df)
val vecs = result.select("vector").map(r => r.get(0).asInstanceOf[SparseVector])
```

One scan only

- Currently online LDA in Spark uses sampling for corpus splitting, this is to align with the other clustering model.
  - For large corpus, feel free to customize the processing and invoke OnlineLDAOptimizer directly
  - It is also supported to stop/resume the inference, though the interface is not public for now.

Neural Network

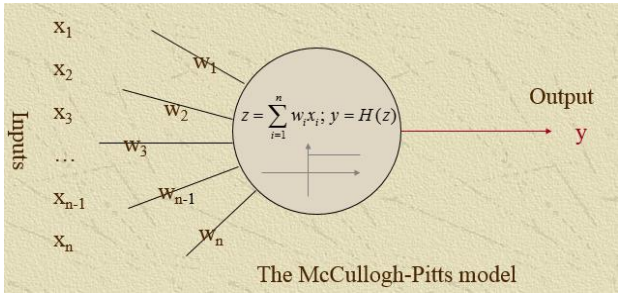
An algorithm born before PC, and inspires the future



Neural Network

Biological inspiration

- React to external
- Learn adaptively
- Simple Units



A most widely used paradigm

- Classification
- Robot/Vision
- Image/audio
- Regression…

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

Deep learning

深度学习是机器学习研究中的一个新的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，例如图像，声音和文本。深度学习是无监督学习的一种。

深度学习的概念由Hinton等人于2006年提出。基于深信度网(DBN)提出非监督贪心逐层训练算法，为解决深层结构相关的优化难题带来希望，随后提出多层自动编码器深层结构。此外Lecun等人提出的卷积神经网络是第一个真正多层结构学习算法，它利用空间相对关系减少参数数目以提高训练性能。

Deep Learning的常用方法

AutoEncoder

Sparse Coding

Restrict Boltzmann Machine (RBM)

Convolutional Neural Network

Neural Network on Spark

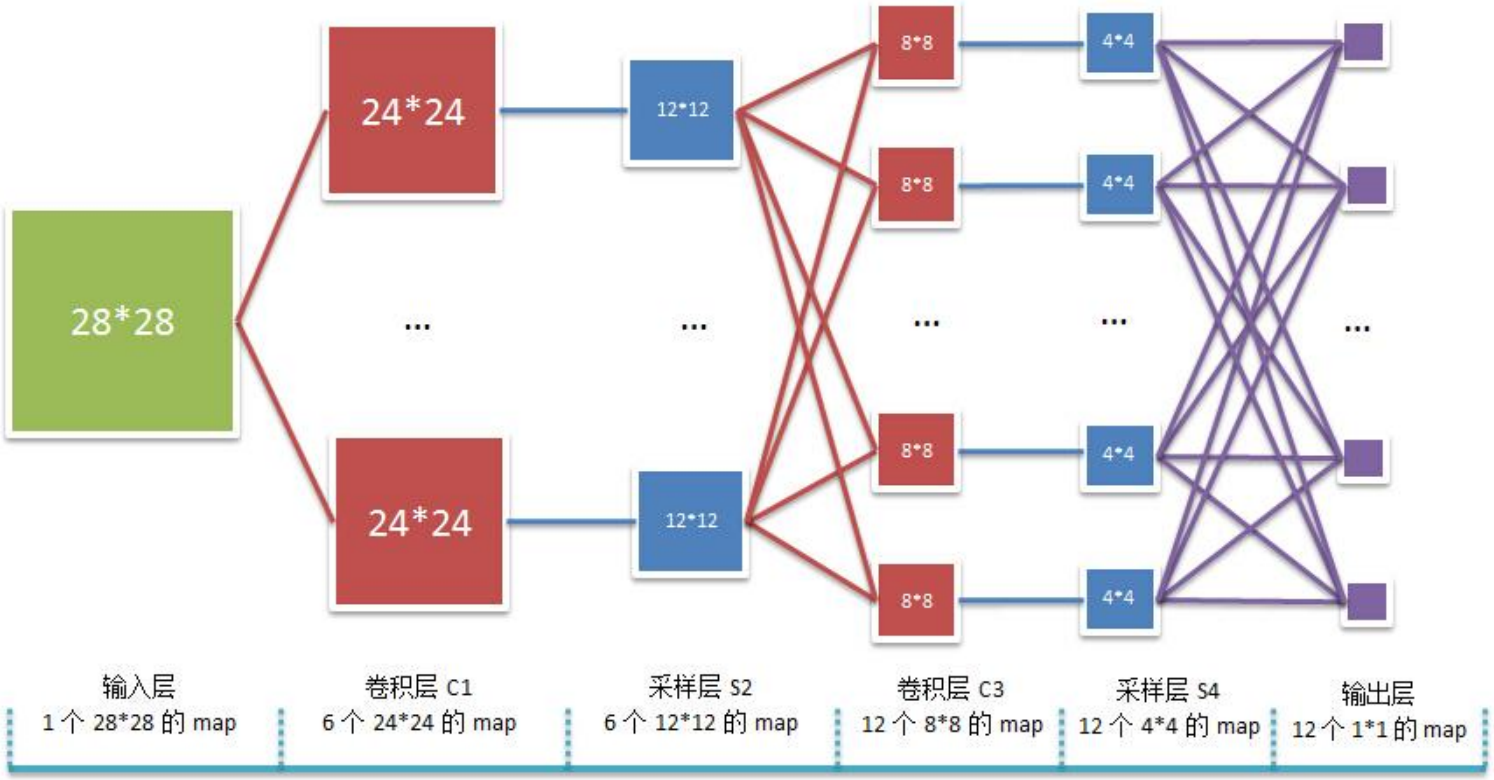
Spark has a very active community competitive!

About ten different versions from different contributor

Review for 1 year…

Finally a mild MLP in 1.5

Convolutional Neural Network



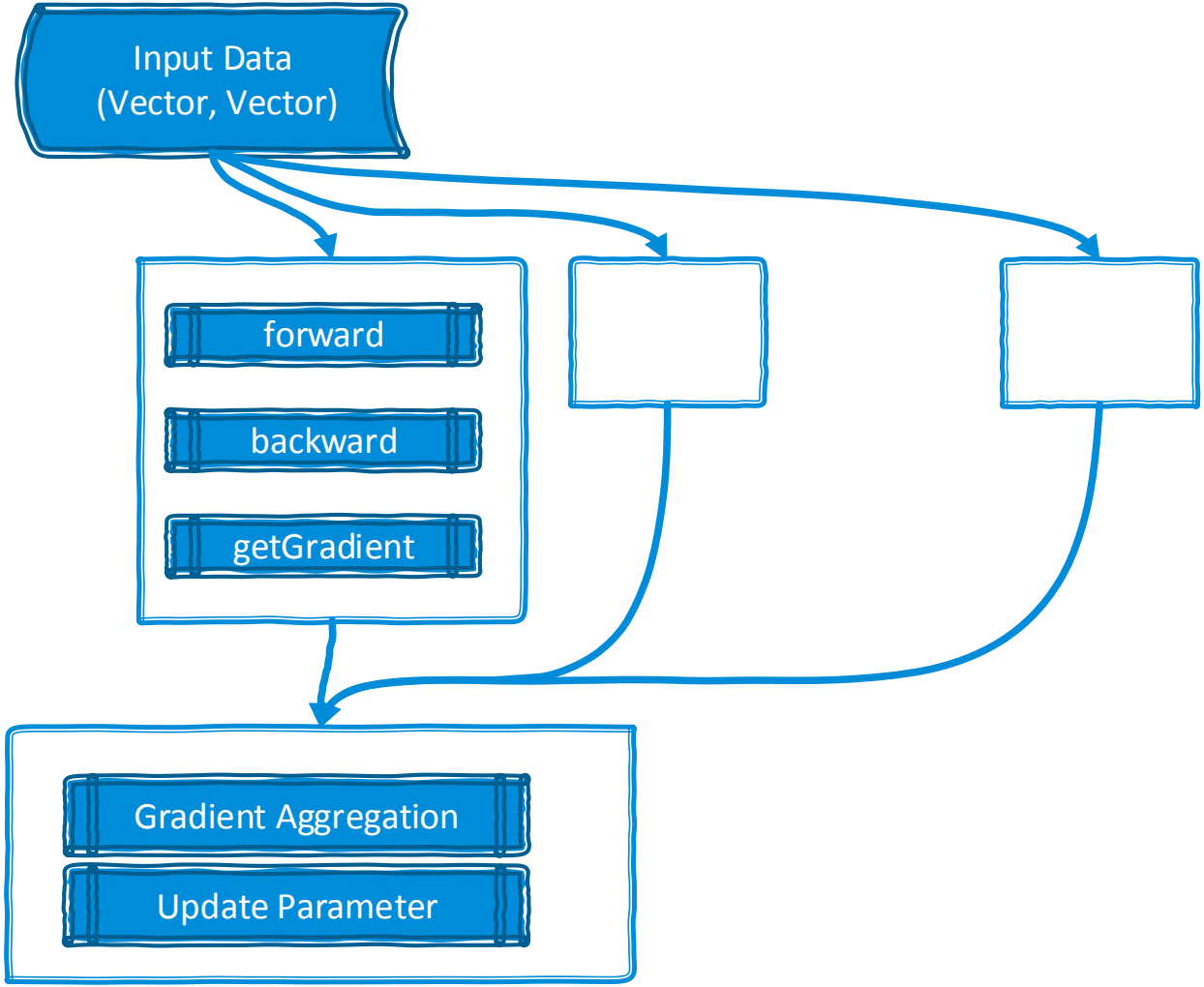
Convolutional Neural Network on Spark

The first CNN implementation on Spark

One implementation, two interfaces.

- torch
- Spark community

Mnist accuracy over 99.7%, matching the best record


Input Data
(Vector, Vector)
forward
backward
getGradient
Gradient Aggregation
Update Parameter

Torch Modules

- Modules are the bricks used to build neural networks. Each are themselves neural networks, but can be combined with other networks using containers to create complex neural networks:
  - Module: abstract class inherited by all modules;
  - Containers: container classes like `Sequential`, `Parallel` and `Concat`;
  - Transfer functions: non-linear functions like `Tanh` and `Sigmoid`;
  - Simple layers: like `Linear`, `Mean`, `Max` and `Reshape`;
  - Table layers: layers for manipulating `table`s like `SplitTable`, `ConcatTable` and `JoinTable`;
  - Convolution layers: `Temporal`, `Spatial` and `Volumetric` convolutions;

Torch Interface, module-based topology

```
val mlp = new Sequential
mlp.add(new ConvLayer(6, new Scale(5, 5)))
mlp.add(new SampLayer(new Scale(2, 2)))
mlp.add(new ConvLayer(12, new Scale(5, 5)))
mlp.add(new SampLayer(new Scale(2, 2)))
mlp.add(new ConvLayer(12, new Scale(4, 4)))

val trainer = new StochasticGradient(mlp, new Criterion)
trainer.train(data)
```

Work closely with community

Interface, scalability improvement for existing solutions

Code at https://github.com/hhbyyh

Trial and suggestion are most welcome

Q & A

Algorithms in MLlib ?

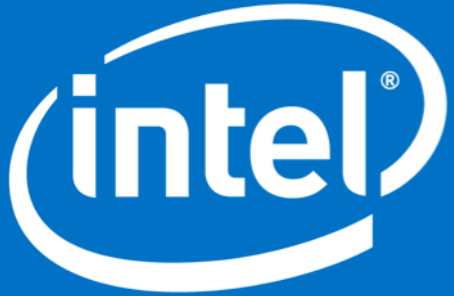
Contributing to Spark ?

New Requirement ?

My data is just that big ?

….

- 主题模型 Topic modeling and LDA
- 在线LDA算法
- 应用和性能调优
- 深度学习
- 卷积神经网络



Yuhao Yang
Github: hhbyyh
yuhao.yang@intel.com

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Forecasts: Any forecasts of requirements for goods and services are provided for discussion purposes only. Intel will have no liability to make any purchase pursuant to forecasts. Any cost or expense you incur to respond to requests for information or in reliance on any forecast will be at your own risk and expense.

Business Forecast: Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.