Analytics

With spark and mllib

Content pre lunch

1. Intro to Analytics
    a. What is analytics (machine learning)
    b. When can it be useful
    c. Different kinds of machine learning
    d. Different applications of machine learning

2. Some Common algorithms
    a. Linear and Logistic Regression
    b. Artificial Neural Networks
    c. Tree based
    d. KMeans
    e. Collaborative Filtering

3. ML Tools and Spark mllib

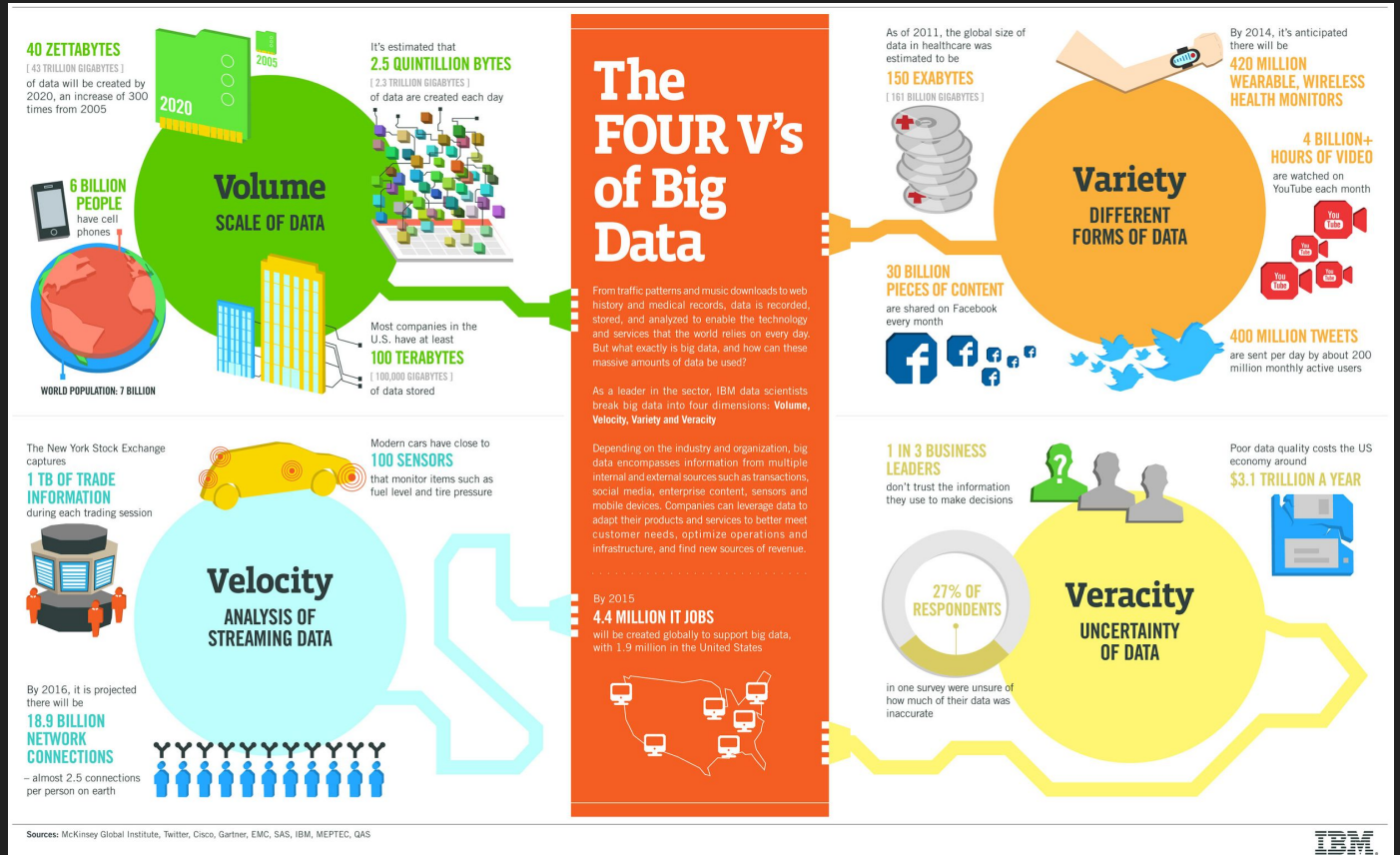
Content post lunch

Individual hands-on exercises

What is ~~analytics~~?

Big Data

What is Big Data?



Big Data

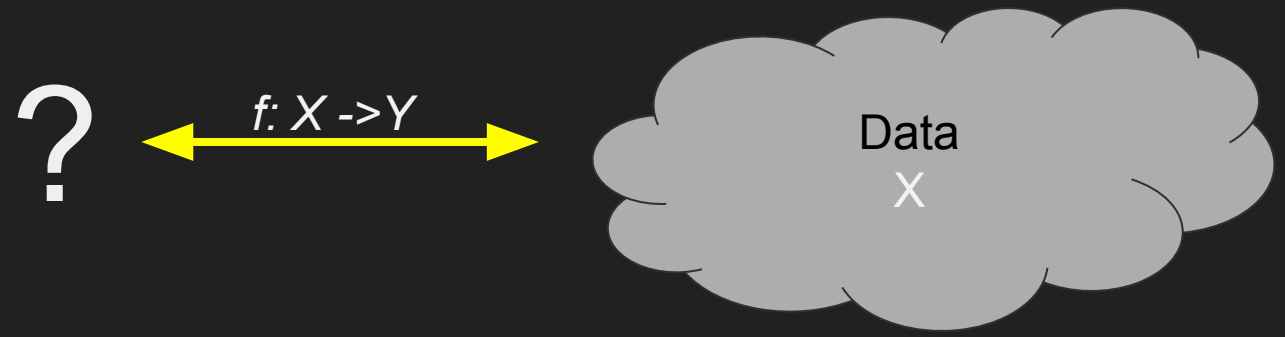
- Data Storage
- Data Processing
- Visualization
- **Analytics**

What is analytics?

aka

- Predictive Analytics (Business)
- **Machine Learning** (CS)
- Statistical Learning (Statistics)
- etc

What is Analytics?

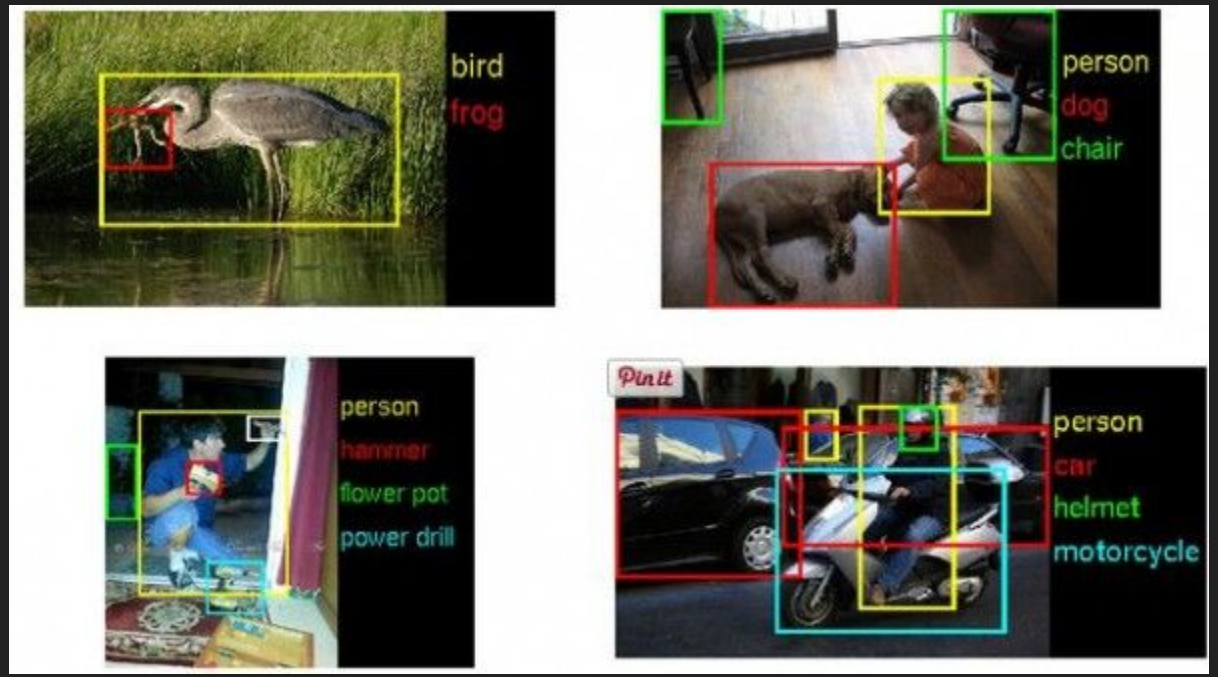


$$p(v) = \frac{4v^2}{\sqrt{\pi}}\left(\frac{m}{2kT}\right)^{3/2} e^{-\frac{mv^2}{2kT}}$$

When can it be useful

Physical model not available or too complex/expensive to construct, e.g. ?=”What’s in the image”

When can it be useful



Different kinds of Machine Learning

Supervised - “training model **with** a teacher”. f(**x**)=**y**. Your data D={(**x**,**y**)}

Unsupervised - “training model **without** a teacher”. f(**x**). Your data D={x}.

Reinforcement Learning - Feedback from actions are used to train the model.

Applications (Supervised)

Regression

- How much (mm) will it rain tomorrow?
- How many t-shirts will we sell next week?
- How much energy will this wind power plant produce tomorrow, given this conditions?

Classification

- Will it rain tomorrow?
- Which size of t-shirts will sell most next week?
- Is the weather conditions good enough for the wind power plant to be on tomorrow?

Application (Unsupervised)

Clustering

- Who likes the same kind of movies (social network analysis)?
- Group homologous sequences into gene families.
- Customer Segmentation.

Anomaly Detection

- Is this bearing faulty based on its vibrations?
- Is this a fraudulent use of this credit card?

Recommender Systems

- This user likes movies x, y and z, which movie is he/she most likely to like besides those?

Application (Reinforcement)

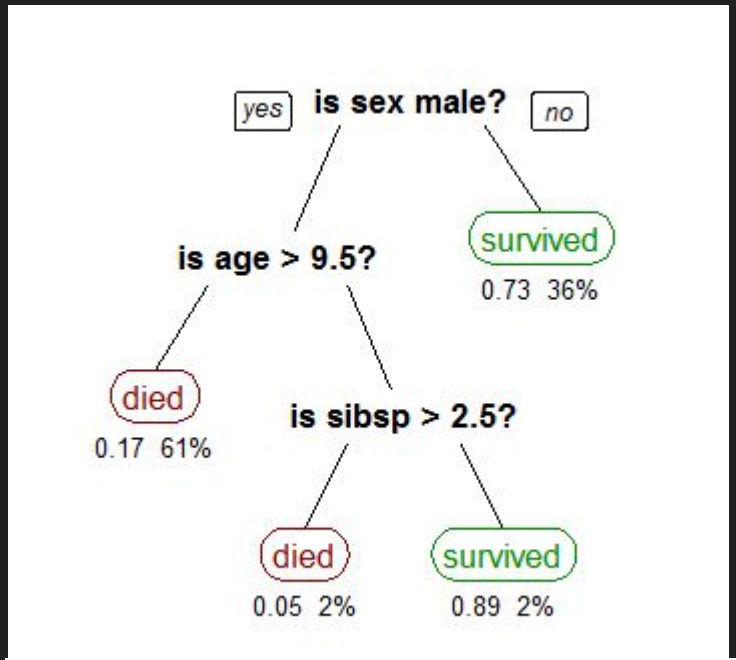
- Computer playing games
- Autonomous driving


ima...
021 3 1

2. Some ML Algorithms

Linear Reg. & Logistic Reg. & Neural Networks

Tree based models (CART)



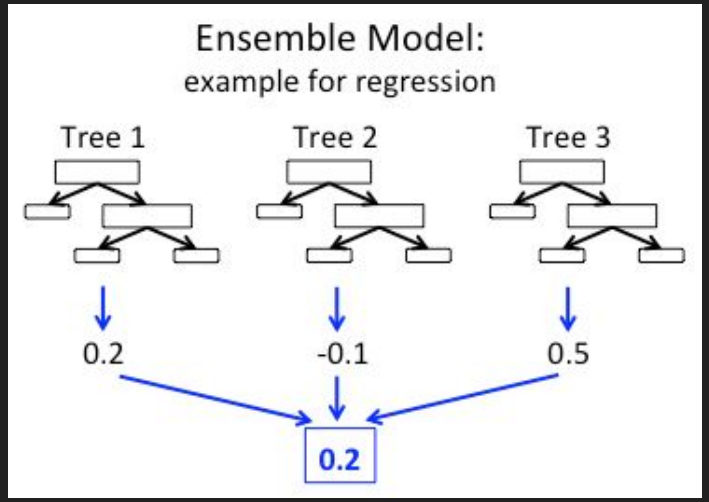
A tree showing survival of passengers on the Titanic

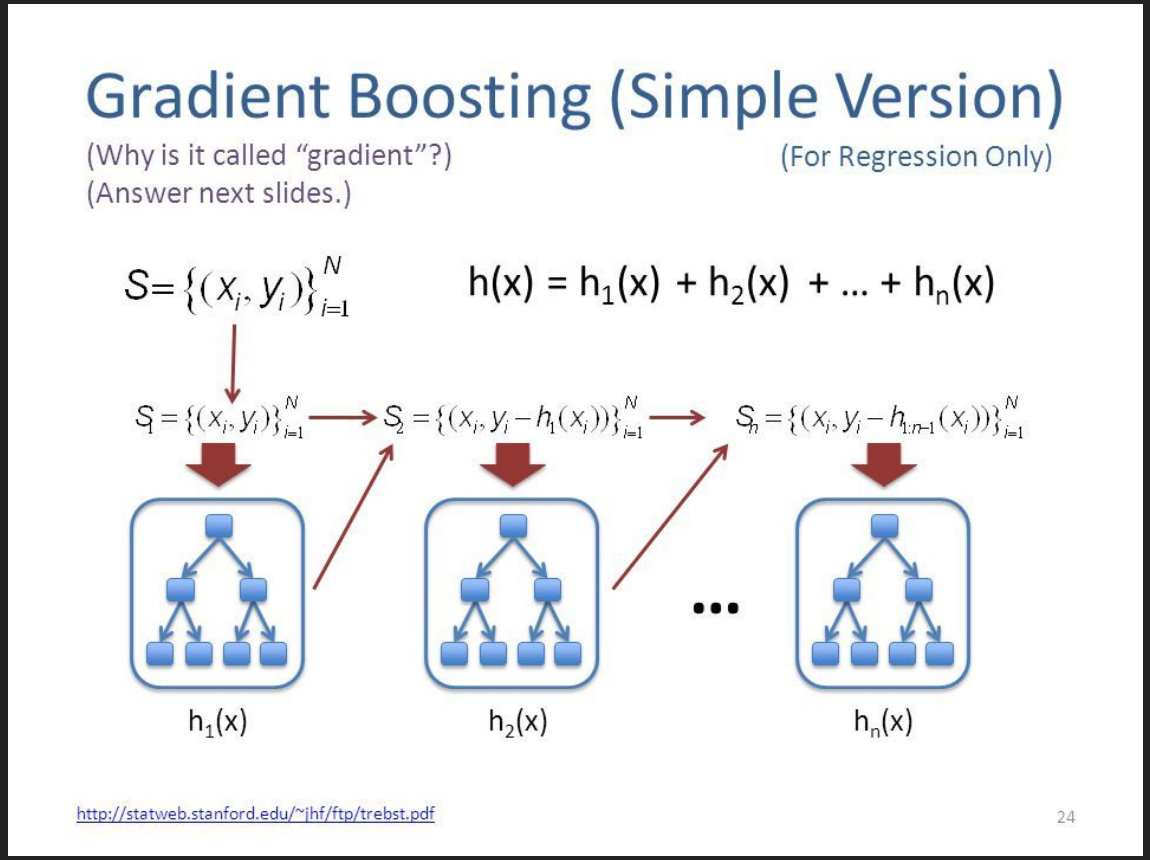
Tricks with trees (Ensemble and Boosting)

Random Forest

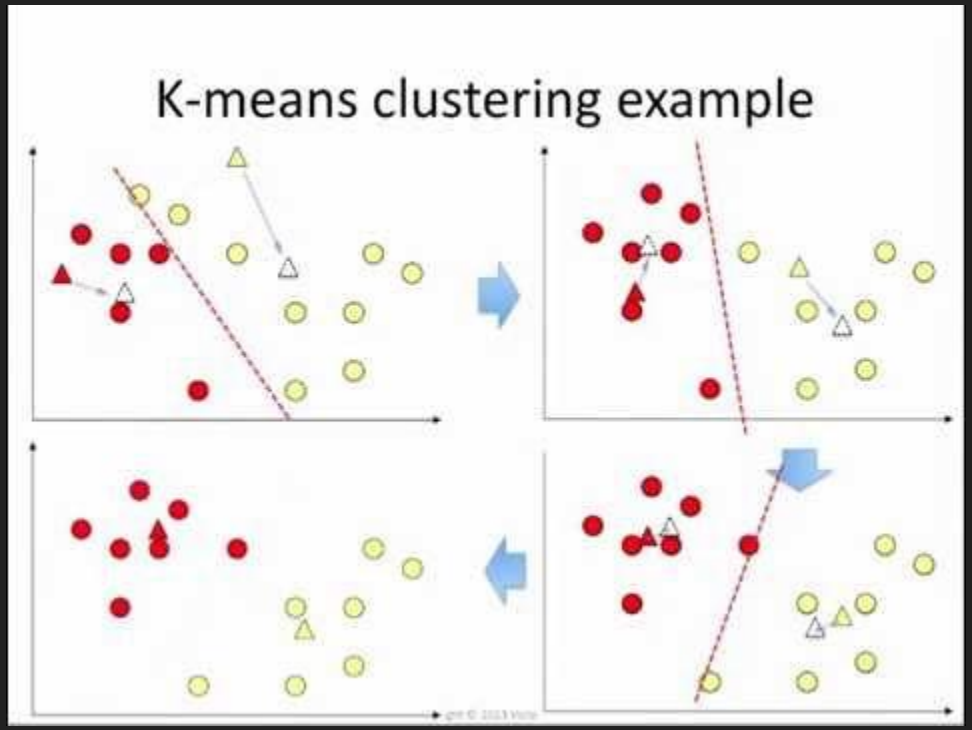
70 % data

70 % variables





KMeans



Collaborative Filtering

Item

mxn

User

User/ Item	A	B	C	D
Ted		4		3
Carol	3		2	
Bob		5		2
Alice			4	

mxf

Ted	1.4	.9
Carol	1.2	1
Bob	1.5	.9
Alice	1.2	.8

User

*

fxn

A	B	C	D
1.4	1.3	.9	1.2
.8	1.1	2	.8

Item

TED A = 1.4 * 1.4 + .8 * .9 = 2.68
B = 1.4 * 1.3 + .9 * 1.1 = 2.81
C = 3.06
D = 2.4

Trained using Alternating Least Squares (ALS):
http://yifanhu.net/PUB/cf.pdf

Part 3 Tools for ML

Some Open Source ML Library Contributors

=> Many free open source ml libraries with high quality
=> No need to write everything from scratch

Status of ML Computation

More data && larger models => better models

Distribute

Accelerators

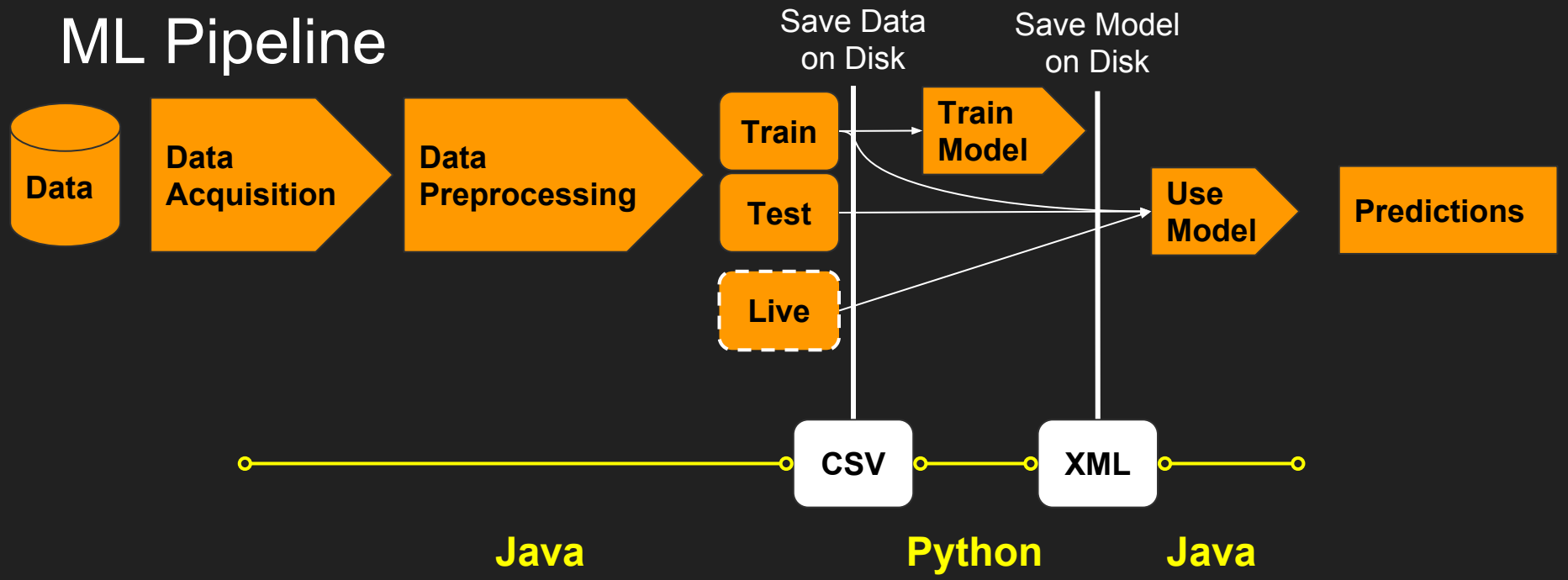
=> Difficult and time consuming to write everything from scratch

What prog. language is most used for data science?

Your primary programming language for Analytics, Data Mining, Data Science tasks: [512 voters]

2015 primary programming language:	
R (and its packages) (263)	51% (of 2015 votes)
Python (including scikit-learn and other libraries) (151)	29%
Other (Java, MATLAB, SAS, Scala, etc) (89)	17%
none (9)	1.8%
2014 primary programming language:	
R (and its packages) (237)	46% (of 2014 votes)
Python (including scikit-learn and other libraries) (117)	23%
Other (Java, MATLAB, SAS, Scala, etc) (118)	23%
none (40)	7.8%

ML Pipeline



mongoDB FOR GIANT IDEAS

mongoDB
java-driver

MyJavaCleaner

scikit learn

spring

“Python is slow”

Python

```
a = 0.1
x = FloatVector(100)
y = FloatVector(100)

x_sum = x.sum()
y.add(a, x)
```

Libraries

C/C++
Fortran

GSL | LAPACK

MPI | BLAS

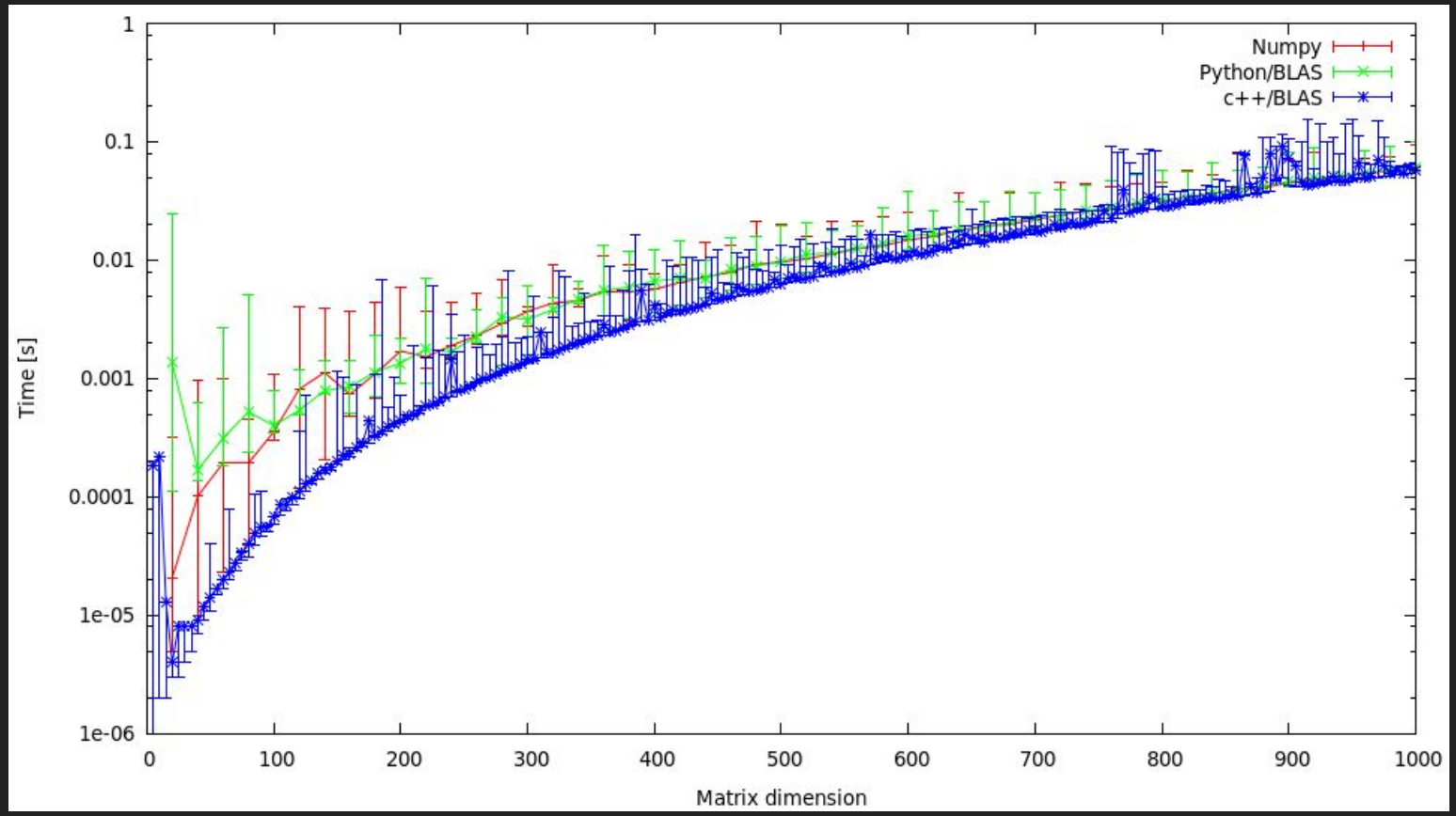
CUDA

cuBLAS | cuDNN

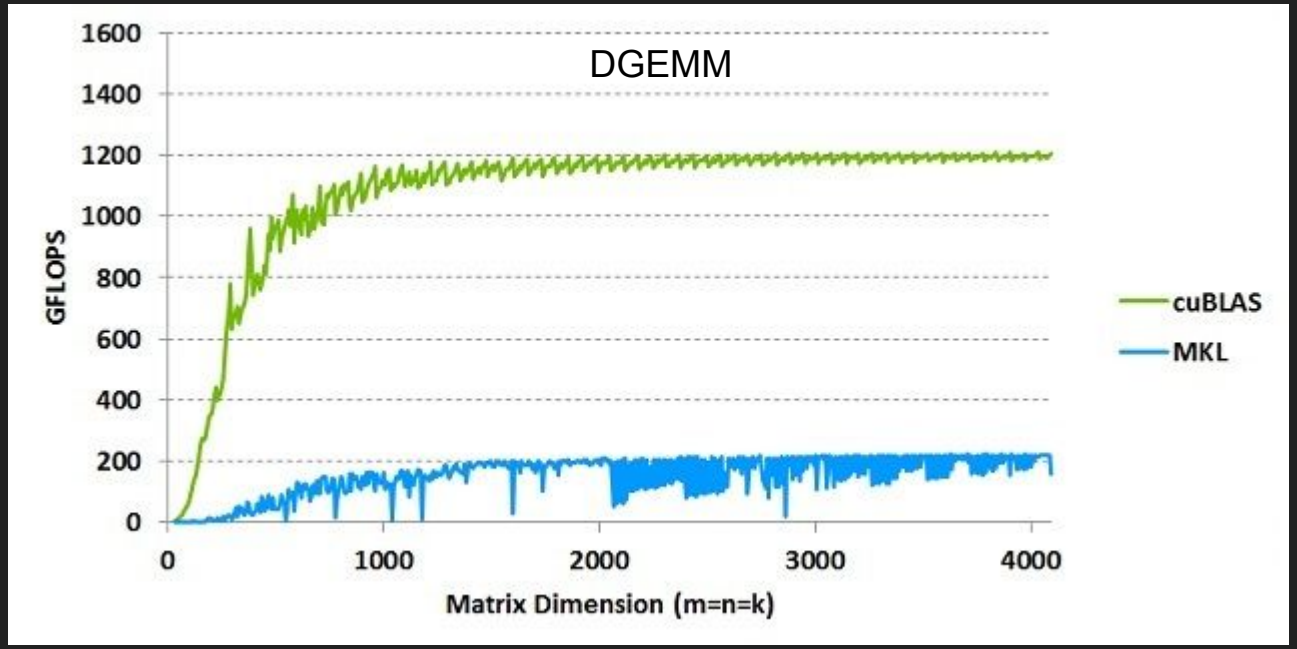
Your Code

```
class FloatVector {
        float* array;
        FloatVector(int size)
                array = new float[size];
        float sum();
                for (...) {}
        void add(float a, Vector& x)
                saxpy(...);
}
```

“Python is slow!”



“Python is slow!”



cuBLAS on K40m, ECC ON, input and output data on device. MKL 11.0.4 on Intel IvyBridge single socket 12 -core E5-2697 v2 @ 2.70GHz

Abstraction Level

Today

Only provide model arguments

Using pre-defined building blocks

Using Matrix and Statistics Libraries

Roll everything yourself

Abstraction Level

How easy it can be with Python and scikit-learn

mm_rain	temp	humidity	month
1.2	17	78	5
3.1	16	84	5
...	...	...	...

```
mm_rain,temp,humidity,month
1.2,17,78,5
3.1,16,84,5
...,...,...,...
```

1. Load csv/json/xml/txt/bin
2. Split into train and test set
3. Create Model
4. Train model
5. Evaluate model
6. Persist model

```
data = np.genfromtxt(path_to_csv,
    delimiter=',', names=True)
X, y = data[:, 1:], data[:, 0]
X_tr, X_te, y_tr, y_te = train_test_split(X, y, 0.2)
model = SVC(C=1.0, kernel='rbf')
model.fit(X_tr, y_tr)
accuracy = model.score(X_te, y_te)
joblib.dump(model, 'filename.pkl')
```

Machine learning in Spark

spark.mllib contains the original API built on top of RDDs.

spark.ml provides higher-level API built on top of DataFrames for constructing ML pipelines.

http://spark.apache.org/docs/latest/mllib-data-types.html

Data types

- Supports local vectors and matrices stored on a single machine, as well as distributed matrices backed by one or more RDDs.
- Numpy arrays and python lists are recognized as local dense vectors and MLlib's SparseVector and SciPy's csc_matrix as local sparse vectors.
- Local Matrix

```
import org.apache.spark.mllib.linalg.{Matrix, Matrices}

// Create a dense matrix ((1.0, 2.0), (3.0, 4.0), (5.0, 6.0))
dm2 = Matrices.dense(3, 2, [1, 2, 3, 4, 5, 6])

// Create a sparse matrix ((9.0, 0.0), (0.0, 8.0), (0.0, 6.0))
sm = Matrices.sparse(3, 2, [0, 1, 3], [0, 2, 1], [9, 6, 8])
```

Data types cont.

Distributed Matrices

- A RowMatrix is a row-oriented distributed matrix without meaningful row indices, backed by an RDD of its rows, where each row is a local vector.
- An IndexedRowMatrix is similar to a RowMatrix but with meaningful row indices. It is backed by an RDD of indexed rows, so that each row is represented by its index and a local vector.
- A CoordinateMatrix is a distributed matrix backed by an RDD of its entries. Each entry is a tuple of (i: Long, j: Long, value: Double), where i is the row index, j is the column index, and value is the entry value.
- A BlockMatrix is a distributed matrix backed by an RDD of MatrixBlocks, where a MatrixBlock is a tuple of ((Int, Int), Matrix), where the (Int, Int) is the index of the block, and Matrix is the sub-matrix at the given index with size rowsPerBlock x colsPerBlock.

LabeledPoint for Supervised Learning

```
from pyspark.mllib.linalg import SparseVector
from pyspark.mllib.regression import LabeledPoint

# Create a labeled point with a positive label and a dense feature vector.
pos = LabeledPoint(1.0, [1.0, 0.0, 3.0])
                   y          x

# Create a labeled point with a negative label and a sparse feature vector.
neg = LabeledPoint(0.0, SparseVector(3, [0, 2], [1.0, 3.0]))
                   y          x
```

Mllib in a slide

- logistic regression and linear support vector machine (SVM)
- classification and regression tree
- random forest and gradient-boosted trees
- recommendation via alternating least squares (ALS)
- clustering via k-means, bisecting k-means, Gaussian mixtures (GMM), and power iteration clustering
- topic modeling via latent Dirichlet allocation (LDA)
- survival analysis via accelerated failure time model
- singular value decomposition (SVD) and QR decomposition
- principal component analysis (PCA)
- linear regression with L_1, L_2, and elastic-net regularization
- isotonic regression
- multinomial/binomial naive Bayes
- frequent itemset mining via FP-growth and association rules
- sequential pattern mining via PrefixSpan
- summary statistics and hypothesis testing
- feature transformations
- model evaluation and hyper-parameter tuning

Exercise 1

::USAGE:: see file download_data.ipnb
from sklearn.datasets import load_iris
data = load_iris()

1. Find a new dataset (regression or classification)

`load_boston` ()	Load and return the boston house-prices dataset (regression).
`load_iris` ()	Load and return the iris dataset (classification).
`load_diabetes` ()	Load and return the diabetes dataset (regression).
`load_digits` ([n_class])	Load and return the digits dataset (classification).
`load_linnerud` ()	Load and return the linnerud dataset (multivariate regression).

Or find dateset from http://mldata.org/

2. Train a gradient boosted trees model with default parameters. Verify result on a testset. Similar to linear and logistic reg.. Guide at http://spark.apache.org/docs/latest/mllib-ensembles.html#Gradient-Boosted-Trees-(GBTS)

3. Try some different combinations of hyper-parameter settings (learningRate and maxDepth) and choose the parameter settings with the best score on a validation set (a held out part of the training set). Verify result on same test set as above.

Exercise 2

1. Train a movie recommender system. Use dataset (ratings.csv) from yesterday and append some ratings of movies you have seen. You can follow this guide: https://databricks-training.s3.amazonaws.com/movie-recommendation-with-mllib.html
2. From the model, extract the user features (model.userFeatures()) and cluster them in 20 clusters. Where did you end up? Can you find something interesting from that cluster (you might need to join with other files, e.g. Movies.csv to get title and genre etc.).

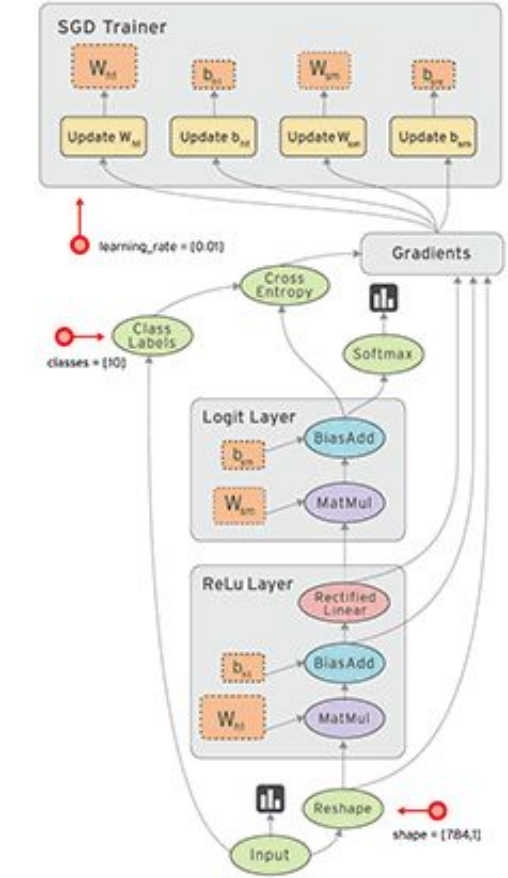
Linear Regression Demo

Homework for Friday

Think about how the tools that we have talked about this week can be used for your research. Prepare 2 min presentation till Friday (informal, no slides).

TensorFlow

- Nodes are operations
- Directed edges show the flow of data



Start with these nodes:

- tf.placeholder(...)
- tf.Variable(...)
- tf.constant(...)

Connect them via operations:

- MatMul
- Add
- Reshape
- Slice
- ...

Define loss function and Optimizer

Execute Graph!