

# ML with Spark

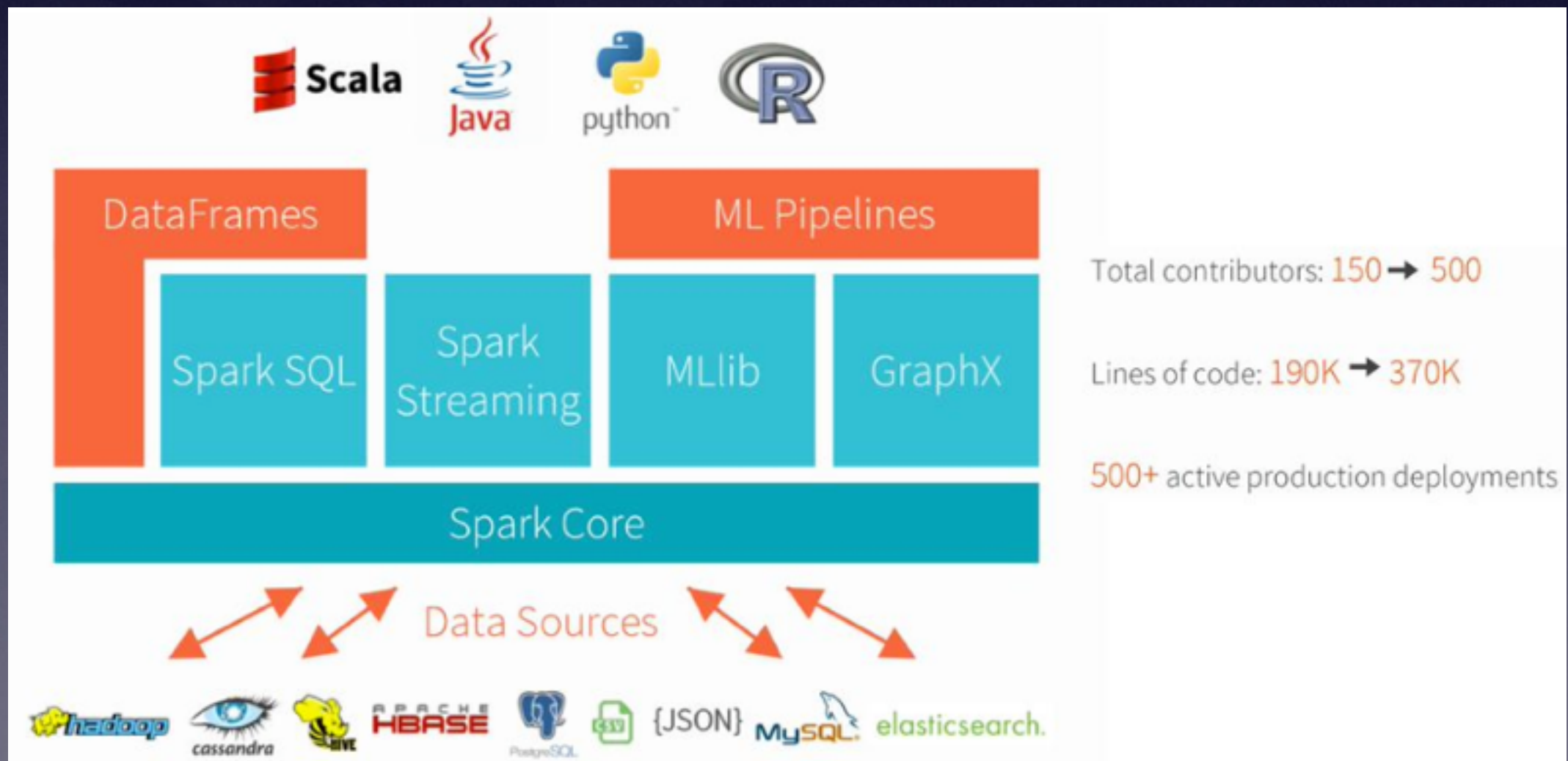
Maochen G.  
[mguan@us.ibm.com](mailto:mguan@us.ibm.com)

2016.05



# Spark overview

- General-purpose cluster computing system
- Written in Scala, also have high-level API's for Java, Python and R(pre-alpha)
- Environment Requirements:
- Java 7+, Python 2.6+, R 3.1+, Scala 2.10



# MLLib in Spark

Maven dependencies

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-mllib_2.11</artifactId>
  <version>1.6.1</version>
</dependency>
```

Two set of libraries:

1. org.apache.spark.ml
2. org.apache.spark.mllib

**Use 2 only if the functionality doesn't exist in 1 !!!**

ml

- Classification
  - Logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - One-vs-Rest classifier (a.k.a. One-vs-All)
- Regression
  - Linear regression
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
- Decision trees
  - Inputs and Outputs
    - Input Columns
    - Output Columns
- Tree Ensembles
  - Random Forests
    - Inputs and Outputs
      - Input Columns
      - Output Columns (Predictions)
  - Gradient-Boosted Trees (GBTs)

Problem Type	Supported Methods
Binary Classification	linear SVMs, logistic regression, decision trees, random forests, gradient-boosted trees, naive Bayes
Multiclass Classification	logistic regression, decision trees, random forests, naive Bayes
Regression	linear least squares, Lasso, ridge regression, decision trees, random forests, gradient-boosted trees, isotonic regression



# MLLib in Spark

Spark Concept Recall:

- SparkConf
- JavaSparkContext

Data Types:

- DataFrame
- JavaRDD<T> - Typical implementation for T is LabeledPoint or Vectors

SQLContext

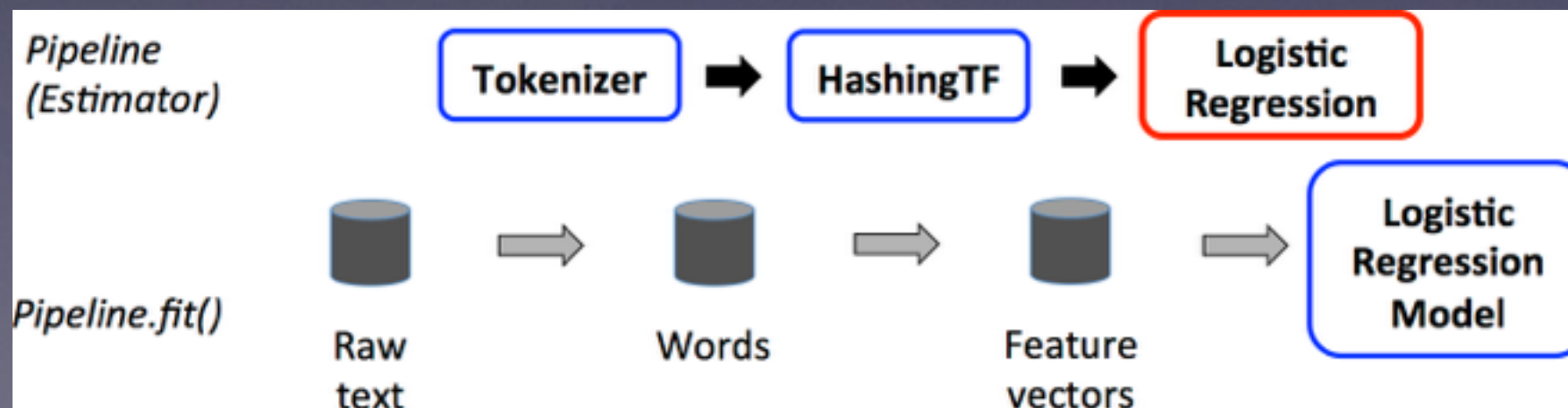
- Create DF.
- Execute SQL-format query over DF.

Pipeline

- Aggregation of stages, stage can be either learning algorithm or any annotat
- Pipeline pipeline = new Pipeline().setStages(new PipelineStage[]{lr});

Estimator

- Can be either a single specific learning alg. or Pipeline



# MLLib in Spark

## Pros:

1. Works well with large dataset.
2. Several good optimization libraries implemented.
3. Implemented full pipelines concept instead of discrete classifiers.
4. Have all training, prediction and cv pipelines.

## Cons:

1. Pure ML library means you need to write adapter for application.
2. Only some of the core ML algorithms implemented currently.
3. Learning Curves for Developer.

## Caveat:

Parallel data processing is ok.

Parallel core training process won't work for iterative ml algorithms

# Problem Definition

Dataset: Student with 2 exam scores ranging from [0 - 100]  
Predict: If grant admission for a potential student.

Dataset Size: 100

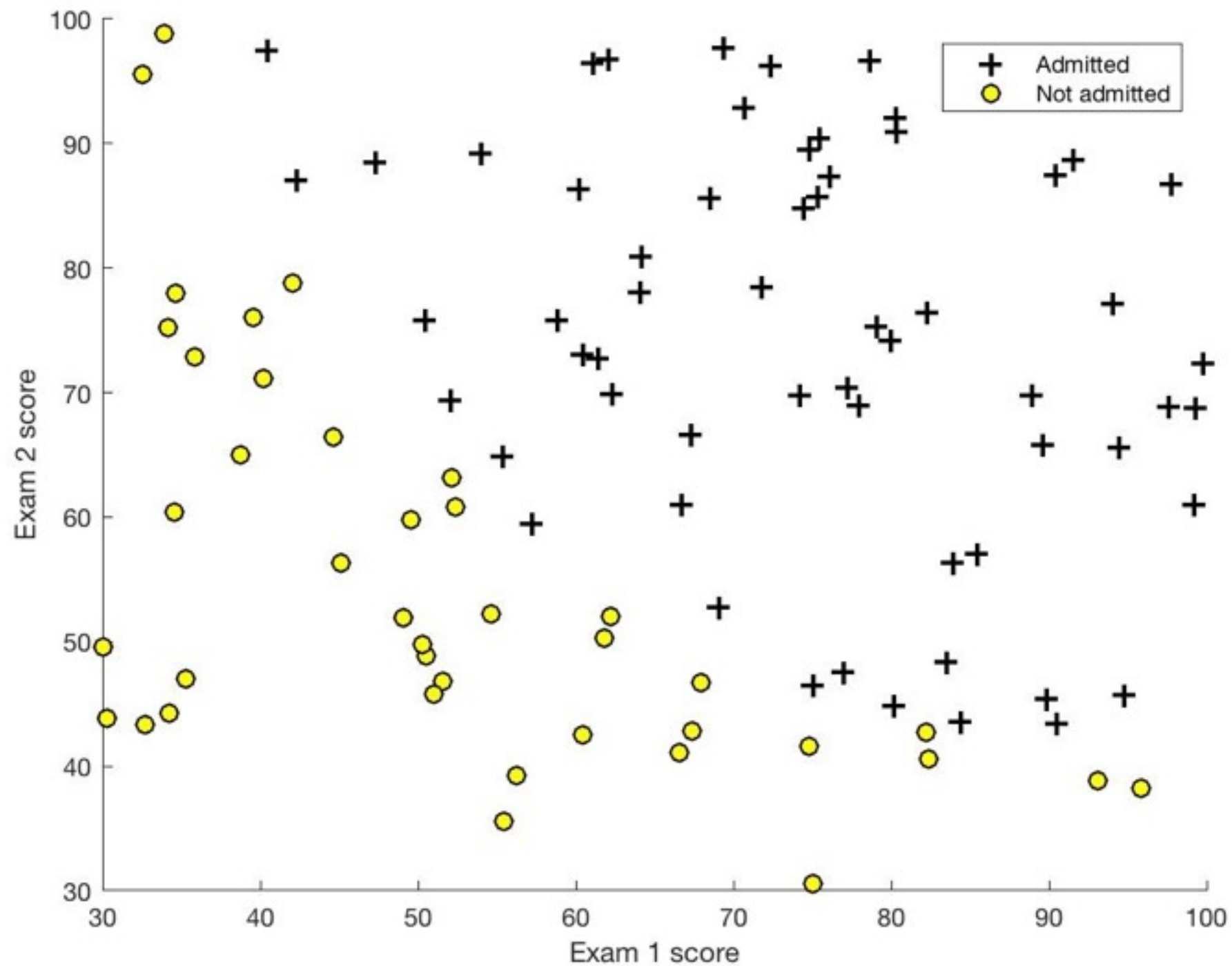
Binary Category: 0 - Decline, 1 - Admit

Examples:

	Exam1,	Exam2,	Label
Student 1:	34.62365962451697,	78.0246928153624,	0
Student 2:	30.28671076822607,	43.89499752400101,	0
Student 3:	35.84740876993872,	72.90219802708364,	0

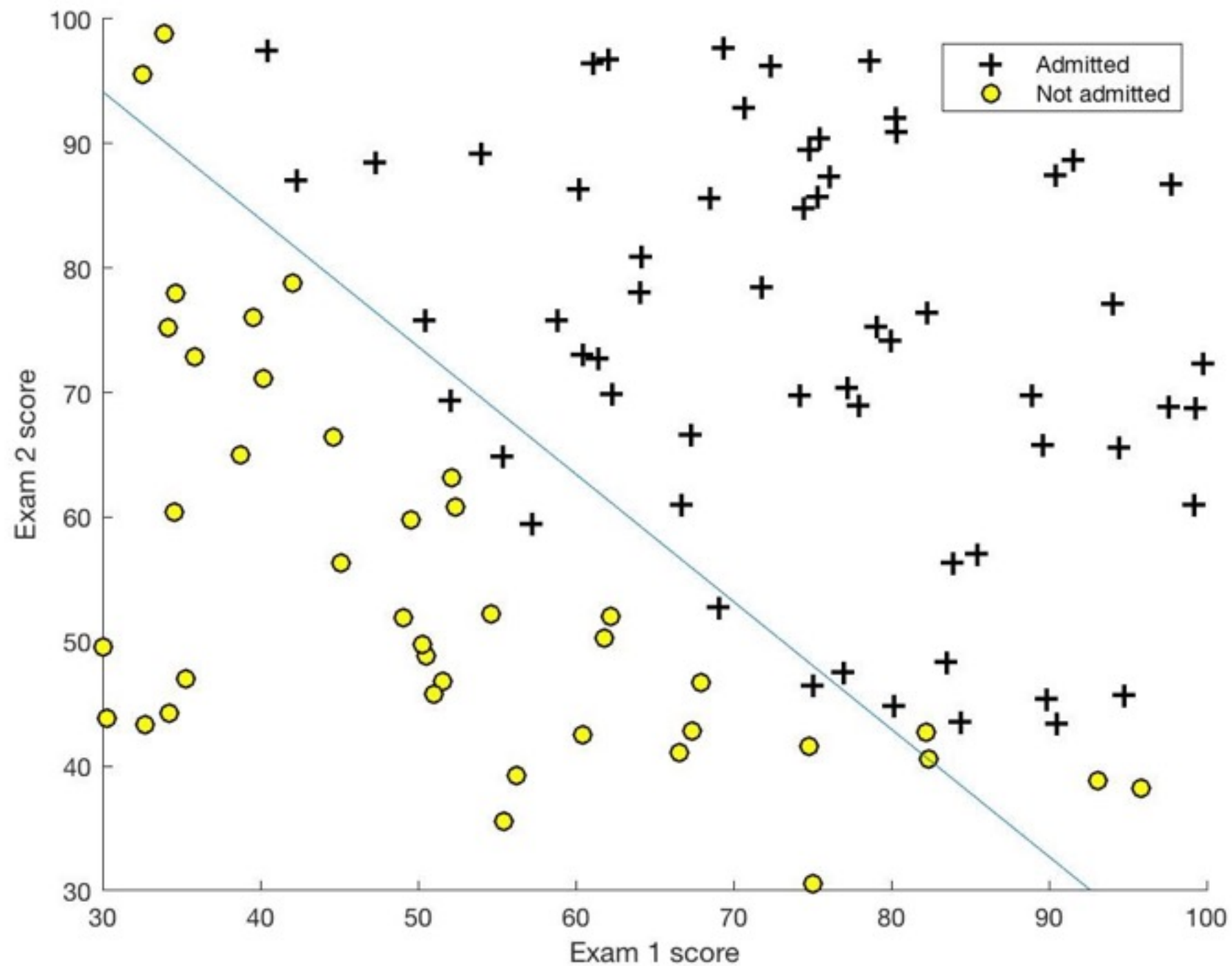
...

# Visualize Dataset





# Visualize Dataset



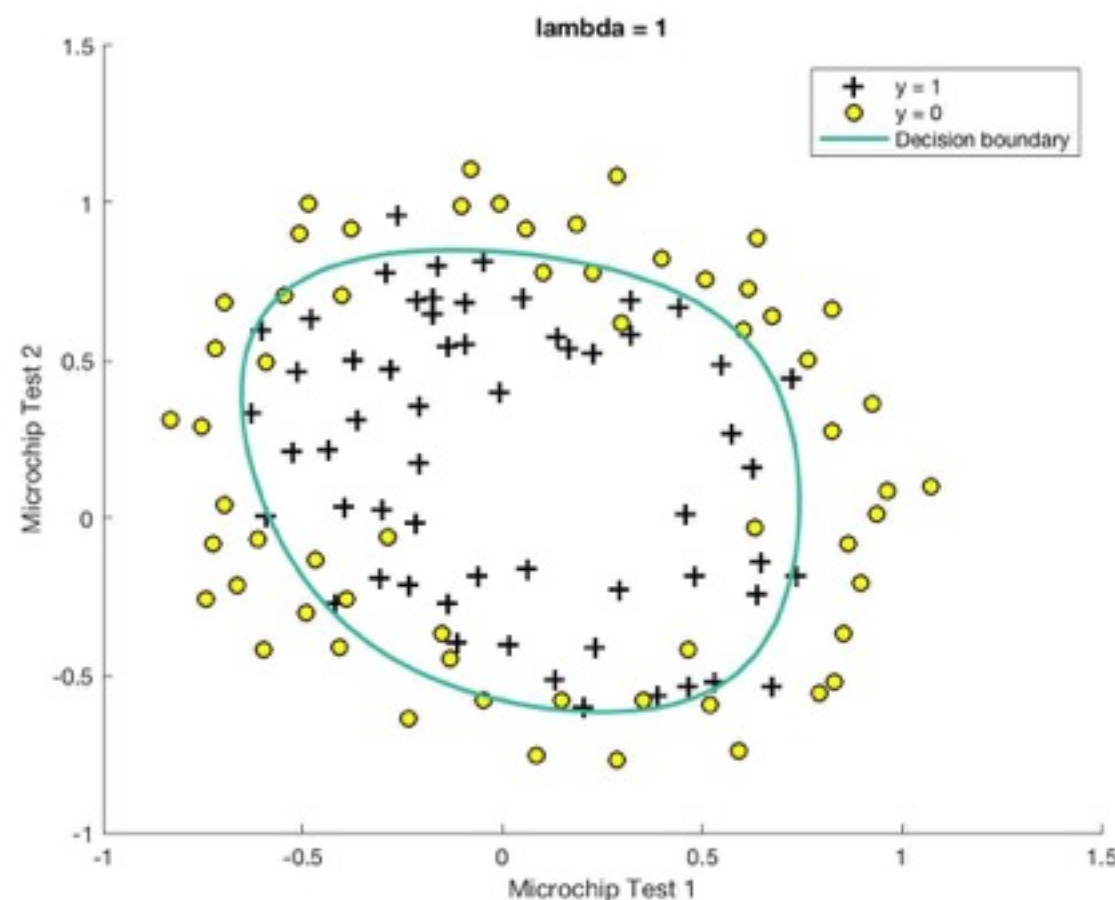
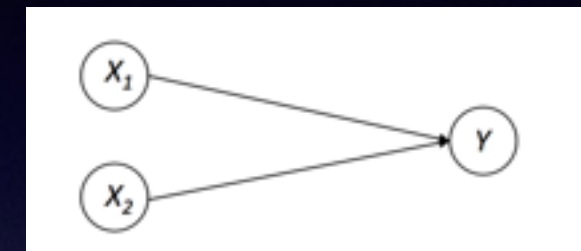


# Logistic Regression (LR)

Log Linear model

This is NOT a regression

- Discriminative model  $P(y|x)$
- LR is a special kind (binomial) of MaxEnt
- Features do not necessary to scale, but encouraged to do so.
- Not necessary for linear boundary.
- The plot uses polynomial feats,  $x^2$ ,  $x_1 \cdot x_2$ , etc.



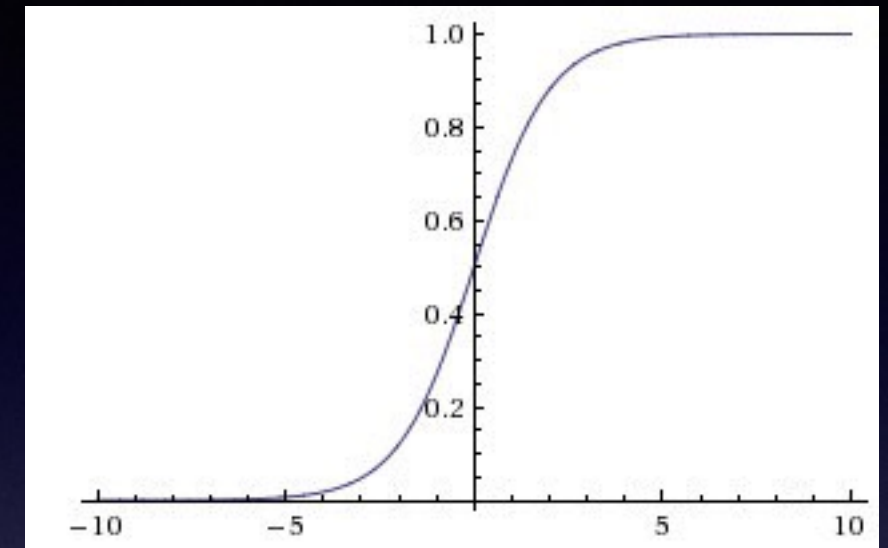
Non-linear decision boundary

# Logistic Regression (LR)

Term Def:

1. Total Training Sample Size:  $m$
2. Total features:  $n$
3.  $i$ -th training sample:  $x_{(i)}$
4.  $j$ -th dimension of  $i$ -th training sample:  $x_{(i)_j}$
5.  $i$ -th training sample label:  $y_{(i)}$
6. Weight vector  $\theta$  ( $n$  dimensional)
7.  $h(\theta)$  is the hypothesis function
8. sigmoid function == logistic function

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}} \quad h(\theta) = \text{sigmoid}\left(\sum_{j=0}^n x_j^{(i)} \cdot \theta_j\right)$$



Sigmoid Function

Training:

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right] \end{aligned}$$

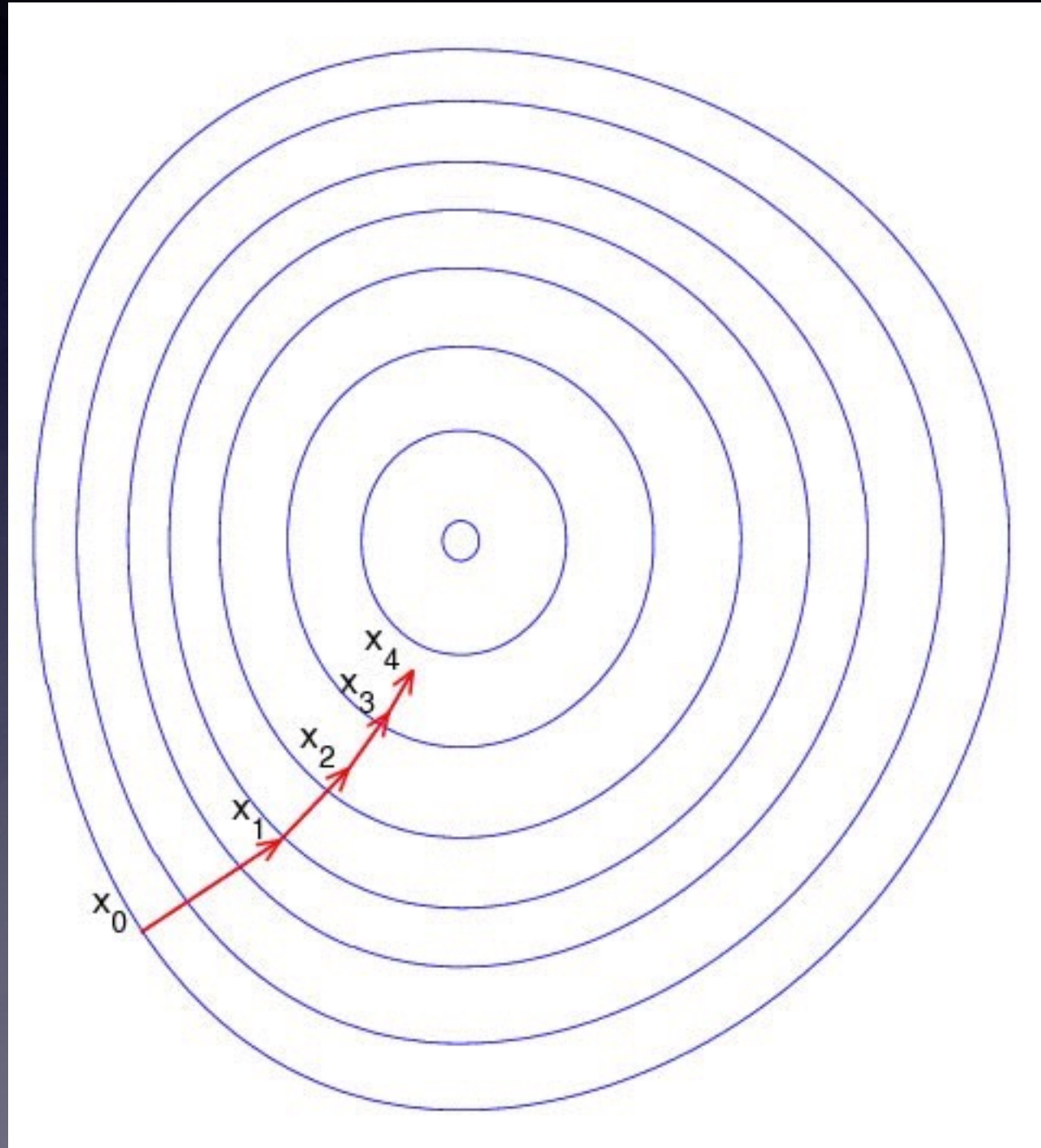
Predict:

$$y = \begin{cases} 0 & h(\theta) < 0.5 \\ 1 & \text{otherwise} \end{cases}$$

Goal: Optimize  $\theta$  vector to minimize  $J(\theta)$

Optimizer: Gradient Descent/SGD/LBFGS

# Gradient Descent





# Coding Time

Weights learned with LR+GD in Matlab by fminunc:

-24.932775, 0.204406, 0.199616 (First is intercept)

Code snippet also available at my github:

<https://github.com/maochen/Playground/blob/master/playground/src/main/java/org/maochen/playground/spark/ClassificationExample.java>

Dataset:

[https://github.com/maochen/Playground/blob/master/playground/src/main/resources/spark/student\\_exam\\_data.txt](https://github.com/maochen/Playground/blob/master/playground/src/main/resources/spark/student_exam_data.txt)