E6895 Advanced Big Data Analytics Lecture 3:

Big Data Foundations (II)

Ching-Yung Lin, Ph.D.
Adjunct Professor, Dept. of Electrical Engineering and Computer Science

February 8th, 2019
Key Components of Spark MLlib

MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics
Correlation: Calculating the correlation between two series of data is a common operation in Statistics

- Pearson’s Correlation
- Spearman’s Correlation
MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

- K-means
  - Input Columns
  - Output Columns
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)
  - Input Columns
  - Output Columns
Example: clustering

Feature Space
Clustering a collection involves three things:

- *An algorithm*—This is the method used to group the books together.
- *A notion of both similarity and dissimilarity*—In the previous discussion, we relied on your assessment of which books belonged in an existing stack and which should start a new one.
- *A stopping condition*—In the library example, this might be the point beyond which books can’t be stacked anymore, or when the stacks are already quite dissimilar.
Clustering — on feature plane
Clustering example

(1, 1)  
(2, 1)  
(1, 2)  
(2, 2)  
(3, 3)  
(8, 8)  
(8, 9)  
(9, 8)  
(9, 9)  

(0, 0)  

y-axis  

x-axis  

(8, 9) (9, 9)  
(8, 8) (9, 8)  
(1, 2) (2, 2)  
(1, 1) (2, 1)  
(3, 3)
Steps on clustering

- Generate Vectors from input data
- Write Vectors to input directory
- Write initial cluster centers
- Run clustering job
- Read clusters from output directory
Making initial cluster centers
K-mean clustering in action. Starting with three random points as centroids (top left), the map stage (top right) assigns each point to the cluster nearest to it. In the reduce stage (bottom left), the associated points are averaged out to produce the new location of the centroid, leaving you with the final configuration (bottom right). After each iteration, the final configuration is fed back into the same loop until the centroids come to rest at their final positions.
HelloWorld clustering scenario result

1.0: [1.000, 1.000] belongs to cluster 0
1.0: [2.000, 1.000] belongs to cluster 0
1.0: [1.000, 2.000] belongs to cluster 0
1.0: [2.000, 2.000] belongs to cluster 0
1.0: [3.000, 3.000] belongs to cluster 0
1.0: [8.000, 8.000] belongs to cluster 1
1.0: [9.000, 8.000] belongs to cluster 1
1.0: [8.000, 9.000] belongs to cluster 1
1.0: [9.000, 9.000] belongs to cluster 1
Other clustering algorithms

Hierarchical clustering
Different clustering approaches

 FIXED NUMBER OF CENTERS

 BOTTOM-UP APPROACH: FROM POINTS TO CLUSTERS VIA GROUPING

 TOP-DOWN APPROACH: SPLITTING THE GIANT CLUSTER
Spark ML Classification and Regression

MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

- Classification
  - Logistic regression
    - Binomial logistic regression
    - Multinomial logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - Linear Support Vector Machine
  - One-vs-Rest classifier (a.k.a. One-vs-All)
  - Naïve Bayes

- Regression
  - Linear regression
  - Generalized linear regression
    - Available families
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression
Spark ML Classification and Regression

MLlib: Main Guide

- Basic statistics
- Pipelines
- Extracting, transforming and selecting features
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

- Linear methods
- 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)
    - Inputs and Outputs
      - Input Columns
      - Output Columns (Predictions)
**DEFINITION**  Computer classification systems are a form of machine learning that use learning algorithms to provide a way for computers to make decisions based on experience and, in the process, emulate certain forms of human decision making.
Classification example: using SVM to recognize a Toyota Camry

Non-ML

Rule 1. Symbol has something like bull’s head
Rule 2. Big black portion in front of car.
Rule 3. .....????
Classification example: using SVM to recognize a Toyota Camry

ML — Support Vector Machine

\[ P_{\text{Camry}} > 0.95 \]

Feature Space

Positive SVs

Negative SVs
### When to use Big Data System for classification?

<table>
<thead>
<tr>
<th>System size in number of examples</th>
<th>Choice of classification approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 100,000</td>
<td>Traditional, non-Mahout approaches should work very well. Mahout may even be slower for training.</td>
</tr>
<tr>
<td>100,000 to 1 million</td>
<td>Mahout begins to be a good choice. The flexible API may make Mahout a preferred choice, even though there is no performance advantage.</td>
</tr>
<tr>
<td>1 million to 10 million</td>
<td>Mahout is an excellent choice in this range.</td>
</tr>
<tr>
<td>&gt; 10 million</td>
<td>Mahout excels where others fail.</td>
</tr>
</tbody>
</table>
How does a classification system work?
Supervised vs. Unsupervised Learning

Classification algorithms are related to, but still quite different from, clustering algorithms such as the k-means algorithm described in previous chapters. Classification algorithms are a form of supervised learning, as opposed to unsupervised learning, which happens with clustering algorithms. A supervised learning algorithm is one that’s given examples that contain the desired value of a target variable. Unsupervised algorithms aren’t given the desired answer, but instead must find something plausible on their own.

Supervised and unsupervised learning algorithms can often be usefully combined. A clustering algorithm can be used to create features that can then be used by a learning algorithm, or the output of several classifiers can be used as features by a clustering algorithm. Moreover, clustering systems often build a model that can be used to categorize new data. This clustering system model works much like the model produced by a classification system. The difference lies in what data was used to produce the model. For classification, the training data includes the target variables; for clustering, the training data doesn’t include target variables.
### Work flow in a typical classification project

<table>
<thead>
<tr>
<th>Stage</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Training the model</td>
<td>Define target variable. Collect historical data. Define predictor variables. Select a learning algorithm. Use the learning algorithm to train the model.</td>
</tr>
<tr>
<td>2. Evaluating the model</td>
<td>Run test data. Adjust the input (use different predictor variables, different algorithms, or both).</td>
</tr>
<tr>
<td>3. Using the model in production</td>
<td>Input new examples to estimate unknown target values. Retrain the model as needed.</td>
</tr>
</tbody>
</table>
Fundamental classification algorithms

Example of fundamental classification algorithms:

- Naive Bayesian
- Complementary Naive Bayesian
- Stochastic Gradient Descent (SDG)
- Random Forest
- Support Vector Machines
Ardi Machine Learning Components

Classification
Support Vector Machine
XGBoost
LightGBM
Random Forest
Decision Tree

Regression
Ordinary Linear Regression
Ridge Regression
Logistic Regression

• Clustering
  • K-means
  • Birch

• Deep Learning
  • Recurrent Neural Network (RNN)
  • Deep neural network (DNN)
  • Convolutional neural network (CNN)
Example of Multi-Modality Multi-Layer Understanding

- Structure Learning
- Evolutionary Modeling & Prediction

Sensor Layer

Feature Layer

Concept Layer

Semantics Layer

Cognition Layer

HR records, Travel records, Badge/Location records, Phone records, Mobile records

Transmitted images, speech content, video content

● observations  ○ hidden states
Example of Advanced Functions

**Linked Big Data**
Most entities are connected. We need a novel AI infrastructure to handle billions connections like our brains.

**Advanced Visualization**
Visualization is the key to understand analysis -- need to combine data mining, big data analytics, design, rendering, and interaction.

**Autonomous Learning**
Continuous Imperfect Learning makes machines to self-learn the environments and keep improving itself, without explicit teachers.

**Machine Reasoning**
Human brains differ from animals mainly in the way we can figure out the hidden causes, intentions and motivations.

**Strategic Thinking**
Intelligence is demonstrated on whether we can possibly anticipate other people's behavior and try to find a way a win-win situation.

**Human Understanding**
Understanding people behavior, personality, needs, and values, is key to better serve customers and advise and recommend what to do.

**Cognitive Feeling**
Machines may evolve into humanlike capabilities such as feeling images, reading body language, and capturing the nuance in speech, etc.

**Prediction**
We are providing various tools for a better prediction on multi-modality time series that might be affected from various factors.
Spark GraphX

- The Property Graph
  - Example Property Graph
- Graph Operators
  - Summary List of Operators
  - Property Operators
  - Structural Operators
  - Join Operators
  - Neighborhood Aggregation
    - Aggregate Messages (aggregateMessages)
    - Map Reduce Triplets Transition Guide (Legacy)
    - Computing Degree Information
    - Collecting Neighbors
  - Caching and Uncaching
- Pregel API
- Graph Builders
- Vertex and Edge RDDs
  - VertexRDDs
  - EdgeRDDs
- Optimized Representation
- Graph Algorithms
  - PageRank
  - Connected Components
  - Triangle Counting
Graph Analytics
Graph Definitions and Concepts

- A graph:
  \[ G = (V, E) \]

- \( V \) = Vertices or Nodes
- \( E \) = Edges or Links

- The number of vertices: “Order”
  \[ N_v = |V| \]

  \[ N_e = |E| \]
Property Graph

Vertex Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Property (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(rxin, student)</td>
</tr>
<tr>
<td>7</td>
<td>(jgonzal, postdoc)</td>
</tr>
<tr>
<td>5</td>
<td>(franklin, professor)</td>
</tr>
<tr>
<td>2</td>
<td>(istoica, professor)</td>
</tr>
</tbody>
</table>

Edge Table

<table>
<thead>
<tr>
<th>SrcId</th>
<th>DstId</th>
<th>Property (E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
<td>Collaborator</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Advisor</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Colleague</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>PI</td>
</tr>
</tbody>
</table>
GraphX Graph Operations

// Information about the Graph ======
val numEdges: Long
val numVertices: Long
val inDegrees: VertexRDD[Int]
val outDegrees: VertexRDD[Int]
val degrees: VertexRDD[Int]

In-degree = 8

Out-degree = 8

// Views of the graph as collections ===
val vertices: VertexRDD[VD]
val edges: EdgeRDD[ED]
val triplets: RDD[EdgeTriplet[VD, ED]]
Reference
A usual example

Figure 2-1. Semantic relationships are hidden in a relational database
Query Example – 1

Figure 2-2. Modeling friends and friends-of-friends in a relational database

Asking “who are Bob’s friends?” is easy, as shown in Example 2-1.

Example 2-1. Bob’s friends

```
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
    ON PersonFriend.FriendID = p1.ID
JOIN Person p2
    ON PersonFriend.PersonID = p2.ID
WHERE p2.Person = 'Bob'
```
Query Examples – II & III

Example 2-2. Who is friends with Bob?

```sql
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
    ON PersonFriend.PersonID = p1.ID
JOIN Person p2
    ON PersonFriend.FriendID = p2.ID
WHERE p2.Person = 'Bob'
```

Example 2-3. Alice’s friends-of-friends

```sql
SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
    ON pf1.PersonID = p1.ID
JOIN PersonFriend pf2
    ON pf2.PersonID = pf1.FriendID
JOIN Person p2
    ON pf2.FriendID = p2.ID
WHERE p1.Person = 'Alice' AND pf2.FriendID <> p1.ID
```

Computational intensive
Graph Database Example

Figure 2-5. Easily modeling friends, colleagues, workers, and (unrequited) lovers in a graph
Execution Time in the example of finding extended friends (by Neo4j)

Partner and Vukotic's experiment seeks to find friends-of-friends in a social network, to a maximum depth of five. Given any two persons chosen at random, is there a path that connects them that is at most five relationships long? For a social network containing 1,000,000 people, each with approximately 50 friends, the results strongly suggest that graph databases are the best choice for connected data, as we see in Table 2-1.

**Table 2-1. Finding extended friends in a relational database versus efficient finding in Neo4j**

<table>
<thead>
<tr>
<th>Depth</th>
<th>RDBMS execution time (s)</th>
<th>Neo4j execution time (s)</th>
<th>Records returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.016</td>
<td>0.01</td>
<td>~2500</td>
</tr>
<tr>
<td>3</td>
<td>30.267</td>
<td>0.168</td>
<td>~110,000</td>
</tr>
<tr>
<td>4</td>
<td>1543.505</td>
<td>1.359</td>
<td>~600,000</td>
</tr>
<tr>
<td>5</td>
<td>Unfinished</td>
<td>2.132</td>
<td>~800,000</td>
</tr>
</tbody>
</table>
Modeling Order History as a Graph

Figure 2-6. Modeling a user’s order history in a graph
A query language on Property Graph – Cypher

Figure 3-1. A simple graph pattern, expressed using a diagram

This pattern describes three mutual friends. Here’s the equivalent ASCII art representation in Cypher:

(a)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)
Building Application Example – Collaborative Filtering

Figure 4-1. Data model for the book reviews user story

Because this data model directly encodes the question presented by the user story, it lends itself to being queried in a way that similarly reflects the structure of the question we want to ask of the data:

```
START reader=node:users(name={readerName})
MATCH reader-[LIKE]->book<-[LIKE]-other_readers-[LIKE]->books
RETURN books.title
```
What's this query for?

START bob=node:user(username='Bob')
MATCH (bob)-[:SENT]->(email)-[:CC]->(alias),
    (alias)-[:ALIAS_OF]->(bob)
RETURN email

Figure 3-10. A graph of email interactions
Data Visualization
D3 Website

http://d3js.org

Data-Driven Documents
D3 Gallery

- **Box Plots**
- **Bubble Chart**
- **Bullet Charts**
- **Calendar View**

- **Non-contiguous Cartogram**
- **Chord Diagram**
- **Dendrogram**
- **Force-Directed Graph**

- **Circle Packing**
- **Population Pyramid 2000**
- **Stacked Bars**
- **Streamgraph**
### D3 Gallery

<table>
<thead>
<tr>
<th>Sunburst</th>
<th>Node-Link Tree</th>
<th>Treemap</th>
<th>Voronoi Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Sunburst" /></td>
<td><img src="image" alt="Node-Link Tree" /></td>
<td><img src="image" alt="Treemap" /></td>
<td><img src="image" alt="Voronoi Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hierarchical Edge Bundling</th>
<th>Voronoi Diagram</th>
<th>Symbol Map</th>
<th>Parallel Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Hierarchical Edge Bundling" /></td>
<td><img src="image" alt="Voronoi Diagram" /></td>
<td><img src="image" alt="Symbol Map" /></td>
<td><img src="image" alt="Parallel Coordinates" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scatterplot Matrix</th>
<th>Zoomable Pack Layout</th>
<th>Hierarchical Bars</th>
<th>Epicyclical Gears</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Scatterplot Matrix" /></td>
<td><img src="image" alt="Zoomable Pack Layout" /></td>
<td><img src="image" alt="Hierarchical Bars" /></td>
<td><img src="image" alt="Epicyclical Gears" /></td>
</tr>
<tr>
<td>D3 Gallery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collision Detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collapsible Force Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force-Directed States</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azimuthal Projections</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Choropleth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collapsible Tree Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoomable Treemap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoomable Partition Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zoomable Area Chart</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drag and Drop Collapsible Tree Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radial Cluster Layout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sankey Diagram</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fisheye Distortion</td>
<td>Hive Plot</td>
<td>Co-occurrence Matrix</td>
<td>Motion Chart</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><img src="image1" alt="Fisheye Distortion" /></td>
<td><img src="image2" alt="Hive Plot" /></td>
<td><img src="image3" alt="Co-occurrence Matrix" /></td>
<td><img src="image4" alt="Motion Chart" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chord Diagram</th>
<th>Animated Béziers</th>
<th>Zoomable Sunburst</th>
<th>Collatz Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Chord Diagram" /></td>
<td><img src="image6" alt="Animated Béziers" /></td>
<td><img src="image7" alt="Zoomable Sunburst" /></td>
<td><img src="image8" alt="Collatz Graph" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parallel Sets</th>
<th>Word Cloud</th>
<th>Obama's Budget Proposal</th>
<th>Facebook IPO</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9" alt="Parallel Sets" /></td>
<td><img src="image10" alt="Word Cloud" /></td>
<td><img src="image11" alt="Obama's Budget Proposal" /></td>
<td><img src="image12" alt="Facebook IPO" /></td>
</tr>
</tbody>
</table>
D3 Gallery

US Trade Deficit

Sequences
sunburst

NFL Predictions

Koalas to the Max

Sankey Creator

Convert any page into bubbles

D3 Builder
D3 Installation

Installing

For NPM, `npm install d3`. For Yarn, `yarn add d3`. Otherwise, download the latest release. The released bundle supports AMD, CommonJS, and vanilla environments. Create a custom bundle using Rollup or your preferred bundler. You can also load directly from d3js.org:

```html
<script src="https://d3js.org/d3.v5.js"></script>
```

For the minified version:

```html
<script src="https://d3js.org/d3.v5.min.js"></script>
```

You can also use the standalone D3 microlibraries. For example, `d3-selection`:

```html
<script src="https://d3js.org/d3-selection.v1.min.js"></script>
```
D3 Selections

In Javascript:

```javascript
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "white", null);
}
```

D3 employs a declarative approach, operating on arbitrary sets of nodes called *selections*. For example, you can rewrite the above loop as:

```javascript
d3.selectAll("p").style("color", "white");
```
D3 Dynamic Properties

Randomly color paragraphs:

```javascript
D3.selectAll("p").style("color", function() {
    return "hsl(" + Math.random() * 360 + ",100%,50%";
});
```

Alternate shades of gray for even and odd nodes:

```javascript
D3.selectAll("p").style("color", function(d, i) {
    return i % 2 ? "#fff" : "#eee";
});
```
Randomly color paragraphs:

d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .style("font-size", function(d) { return d + "px"; });

Computed properties often refer to bound data. Data is specified as an array of values, and each value is passed as the first argument (d) to selection functions. With the default join-by-index, the first element in the data array is passed to the first node in the selection, the second element to the second node, and so on. For example, if you bind an array of numbers to paragraph elements, you can use these numbers to compute dynamic font sizes:
Using D3’s `enter` and `exit` selections, you can create new nodes for incoming data and remove outgoing nodes that are no longer needed.

When data is bound to a selection, each element in the data array is paired with the corresponding node in the selection. If there are fewer nodes than data, the extra data elements form the `enter` selection, which you can instantiate by appending to the `enter` selection. For example:

```javascript
let data = [4, 8, 15, 16, 23, 42];

d3.select("body")
  .selectAll("p")
  .data(data)
  .enter().append("p")
  .text(function(d) { return "I’m number " + d + "!"; });
```
// Update...
var p = d3.select("body")
  .selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .text(function(d) { return d; });

// Enter...
p.enter().append("p")
  .text(function(d) { return d; });

// Exit...
p.exit().remove();
D3 Transformation

For example, to fade the background of the page to black:

```javascript
d3.select("body").transition()
 .style("background-color", "black");
```

Or, to resize circles in a symbol map with a staggered delay:

```javascript
d3.selectAll("circle").transition()
 .duration(750)
 .delay(function(d, i) { return i * 10; })
 .attr("r", function(d) { return Math.sqrt(d * scale); });
```
D3 Bar Chart Tutorial

https://bost.ocks.org/mike/bar/

```javascript
var data = [4, 8, 15, 16, 23, 42];
```

Selecting an Element

Javascript:

```javascript
var div = document.createElement("div");
div.innerHTML = "Hello, world!";
document.body.appendChild(div);
```

D3:

```javascript
var body = d3.select("body");
var div = body.append("div");
div.html("Hello, world!");
```
Chaining Methods

```javascript
var body = d3.select("body");
body.style("color", "black");
body.style("background-color", "white");

d3.select("body")
  .style("color", "black")
  .style("background-color", "white");

var section = d3.selectAll("section");
section.append("div")
  .html("First!");
section.append("div")
  .html("Second.");
```
Coding a Chart, Manually

```html
<!DOCTYPE html>
<style>
.chart div {
  font: 10px sans-serif;
  background-color: steelblue;
  text-align: right;
  padding: 3px;
  margin: 1px;
  color: white;
}
</style>
<div class="chart">
  <div style="width: 40px;">4</div>
  <div style="width: 80px;">8</div>
  <div style="width: 150px;">15</div>
  <div style="width: 160px;">16</div>
  <div style="width: 230px;">23</div>
  <div style="width: 420px;">42</div>
</div>
```
D3 Bar Chart Tutorial

Coding a Chart, Automatically

d3.select(".chart")
  .selectAll("div")
    .data(data)
    .enter().append("div")
    .style("width", function(d) { return d * 10 + "px"; })
    .text(function(d) { return d; });

First, we select the chart container using a class selector.

var chart = d3.select(".chart");

Next we initiate the data join by defining the selection to which we will join data.

var bar = chart.selectAll("div");
Coding a Chart, Automatically

```javascript
var barUpdate = bar.data(data);

var barEnter = barUpdate.enter().append("div");

barEnter.style("width", function(d) { return d * 10 + "px"; });

barEnter.text(function(d) { return d; });
```
Scaling to Fit

```javascript
var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, 420]);
```
D3 Bar Chart Tutorial

```html
<!DOCTYPE html>
<style>
.charts rect {
  fill: steelblue;
}
.charts text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
<svg class="chart" width="420" height="120">
  <g transform="translate(0,0)">
    <rect width="40" height="19"></rect>
    <text x="37" y="9.5" dy=".35em">4</text>
  </g>
  <g transform="translate(0,20)">
    <rect width="80" height="19"></rect>
    <text x="77" y="9.5" dy=".35em">8</text>
  </g>
  <g transform="translate(0,40)">
    <rect width="150" height="19"></rect>
    <text x="147" y="9.5" dy=".35em">15</text>
  </g>
  <g transform="translate(0,60)">
    <rect width="160" height="19"></rect>
    <text x="157" y="9.5" dy=".35em">16</text>
  </g>
  <g transform="translate(0,80)">
    <rect width="230" height="19"></rect>
    <text x="227" y="9.5" dy=".35em">23</text>
  </g>
  <g transform="translate(0,100)">
    <rect width="420" height="19"></rect>
    <text x="417" y="9.5" dy=".35em">42</text>
  </g>
</svg>

Full code to do it manually

---

© 2019 CY Lin, Columbia University
D3 Bar Chart Tutorial

Full code to do it automatically

```html
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.chart rect {
    fill: steelblue;
}
.chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
}
</style>
<svg class="chart"></svg>
<script src="//d3js.org/d3.v3.min.js" charset="utf-8"></script>
```
D3 Bar Chart Tutorial

Full code to do it automatically

```html
<script>
var data = [4, 8, 15, 16, 23, 42];

var width = 420,
    barHeight = 20;

var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, width]);

var chart = d3.select(".chart")
    .attr("width", width)
    .attr("height", barHeight * data.length);

var bar = chart.selectAll("g")
    .data(data)
    .enter().append("g")
    .attr("transform", function(d, i) { return "translate(0," + i * barHeight + ")"; });

bar.append("rect")
    .attr("width", x)
    .attr("height", barHeight - 1);

bar.append("text")
    .attr("x", function(d) { return x(d) - 3; })
    .attr("y", barHeight / 2)
    .attr("dy", "0.35em")
    .text(function(d) { return d; });
</script>
```
Load data

```javascript
// 1. Code here runs first, before the download starts.
d3.tsv("data.tsv", function(error, data) {
  // 3. Code here runs last, after the download finishes.
});

// 2. Code here runs second, while the file is downloading.
```

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locke</td>
<td>4</td>
</tr>
<tr>
<td>Reyes</td>
<td>8</td>
</tr>
<tr>
<td>Ford</td>
<td>15</td>
</tr>
<tr>
<td>Jarrah</td>
<td>16</td>
</tr>
<tr>
<td>Shephard</td>
<td>23</td>
</tr>
<tr>
<td>Kwon</td>
<td>42</td>
</tr>
</tbody>
</table>

The equivalent of Javascript code:

```javascript
var data = [
  {name: "Locke", value: 4},
  {name: "Reyes", value: 8},
  {name: "Ford", value: 15},
  {name: "Jarrah", value: 16},
  {name: "Shephard", value: 23},
  {name: "Kwon", value: 42}
];
```
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.chart rect {
    fill: steelblue;
}
.chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
}
</style>
<svg class="chart"></svg>
<script src="/d3js.org/d3.v3.min.js" charset="utf-8"></script>
<script>
var width = 420,
    barHeight = 20;

var x = d3.scale.linear()
    .range([0, width]);

var chart = d3.select(".chart")
    .attr("width", width);

d3.tsv("data.tsv", type, function(error, data) {
    x.domain([0, d3.max(data, function(d) { return d.value; })])
    chart.attr("height", barHeight * data.length);

    var bar = chart.selectAll("g")
        .data(data)
        .enter().append("g")
        .attr("transform", function(d, i) { return "translate(0," + i * barHeight + ")"; })

    bar.append("rect")
        .attr("width", function(d) { return x(d.value); })
        .attr("height", barHeight - 1);

    bar.append("text")
        .attr("x", function(d) { return x(d.value) - 3; })
        .attr("y", barHeight / 2)
        .attr("dy", ".35em")
        .text(function(d) { return d.value; });
});

function type(d) {
    d.value = +d.value; // coerce to number
    return d;
}
</script>