BUSINESS INTELLIGENCE LABORATORY

MICROSOFT AZURE ML

Anna Monreale
Computer Science Department, University of Pisa
Deploying a ML model

- A model can be **deployed** so that it be accessed by applications
- Models often need to be recreated and redeployed frequently
- Azure ML provides fast, frequent deployments **through an Azure ML API service**
- The model exposes its **functionality** through the **Azure ML API’s RESTful interface**
  - applications access this model and use the results it provides
- The API exposes both a synchronous service for requesting single predictions and an asynchronous service for sending in bulk requests.
Using a model
Web Service

- Describe a service that a client (a computer) can call remotely over the internet, via web protocols like HTTP
- A Web service is an application that:
  - Runs on a Web server
  - Exposes Web methods to interested callers
  - Listens for HTTP requests representing commands to invoke Web methods
  - Executes Web methods and returns the results
Web Service API

- Software developers can use web service by API
- **Message format**
  - Some web services use SOAP, a particular XML format, others use generic XML markup and still others use JSON.
- **Syntax of the service request**
RESTFUL WEB SERVICES

Anna Monreale
Computer Science Department, University of Pisa
Hypertext Transfer Protocol (HTTP)

- A communications protocol
- Allows retrieving inter-linked text documents (hypertext)

- HTTP Verbs
  - HEAD
  - GET
  - POST
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT

```plaintext
GET /index.html HTTP/1.1
Host: www.pitt.edu

HTTP/1.1 200 OK
Content-Type: text/html

<html><head>…
```
REST and HTTP

- The motivation for REST was to capture the characteristics of the Web which made the Web successful.
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response

- Exploits the use of the HTTP protocol
  - HTTP POST, HTTP GET, HTTP PUT, HTTP DELETE
Main Concepts

Nouns (Resources)
unconstrained
i.e., http://example.com/employees/12345

Verbs
constrained
i.e., GET

REST

Representations
constrained
i.e., XML
The key abstraction of information in REST is a resource.

A resource is a conceptual mapping to a set of entities

- Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on

Represented with a global identifier (URI in HTTP)

- http://www.boeing.com/aircraft/747
Naming Resources

- REST uses URI to identify resources

  - http://localhost/books/
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students

- As you traverse the path from more generic to more specific, you are navigating the data
Verbs

- Represent the actions to be performed on resources
  - HTTP GET
  - HTTP POST
  - HTTP PUT
  - HTTP DELETE
HTTP GET

- How clients ask for the information they seek.

- Issuing a GET request transfers the data from the server to the client in some representation

  - GET http://localhost/books
    - Retrieve all books

    - Retrieve book identified with ISBN-0011021

  - GET http://localhost/books/ISBN-0011021/authors
    - Retrieve authors for book identified with ISBN-0011021
HTTP PUT, HTTP POST

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST [http://localhost/books/](http://localhost/books/)
  - Content: {title, authors[], ...}
  - Creates a new book with given properties

- PUT [http://localhost/books/isbn-111](http://localhost/books/isbn-111)
  - Content: {isbn, title, authors[], ...}
  - Updates book identified by isbn-111 with submitted properties
HTTP DELETE

- Removes the resource identified by the URI

  - Delete book identified by ISBN-0011
Representations

- How data is represented or returned to the client for presentation.

- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML

- It is common to have multiple representations of the same data
Representations

- **XML**
  - `<COURSE>`
    - `<ID>CS2650</ID>`
    - `<NAME>Distributed Multimedia Software</NAME>`
  - `</COURSE>`

- **JSON**
  - `{course
    - {id: CS2650}
    - {name: Distributed Multimedia Software}
  }`
Architecture Style

Web/Proxy Server

HTTP GET
Request (XML doc)

URL 1
HTTP Response

Response (XML doc)

HTTP POST
Request (XML doc)

URL 1
HTTP Response

Response (JSON doc)

HTTP DELETE
PO (XML doc)

URL 1
HTTP Response

Response (TEXT doc)

REST Engine (locate resource and generate response)

doGet()
doPost(id)
doDelete()
DEPLOYMENT OF A MODEL

Anna Monreale
Computer Science Department, University of Pisa
**Deployment by Web Interface**

- **RETRAINING WEB Services**
  - This saves the model as a Trained Model and adds web service Input and Output modules
- **Deploy Web Service [Classic]**
Deployment from Python Notebooks

- The **azureml python client library** makes it possible to deploy machine learning web services on Azure based on models developed in Python
- You need to install from PyPI:
  - `pip install azureml`
- You need to access your workspace
  - The code snippets include: **your workspace id, authorization token**, and other necessary identifiers to get to your datasets
Accessing your workspace

```python
from azureml import Workspace

ws = Workspace(
    workspace_id='4c29e1adeba2e5a7cbeb0e4f4adfb4df',
    authorization_token='f4f3ade2c6aefdb1afbb043cd8bcf3daf',
)
```

If you're using AzureML in a region other than South Central US you'll also need to specify the endpoint:

```python
from azureml import Workspace

ws = Workspace(
    workspace_id='4c29e1adeba2e5a7cbeb0e4f4adfb4df',
    authorization_token='f4f3ade2c6aefdb1afbb043cd8bcf3daf',
    endpoint='https://europeweststudio.azureml.net/
)
```
Service Publishing

```python
@services.publish(workspace, workspace_token)
@services.types(a = float, b = float)
@services.returns(float)
def func(a, b):
    return a / b
```

After the function is published there will be a "service" property on the function. This object has several properties of interest:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>this is the end point for executing the function</td>
</tr>
<tr>
<td>api_key</td>
<td>this is the API key which is required to invoke the function</td>
</tr>
<tr>
<td>help_url</td>
<td>this is a human readable page which describes the parameters and results of the function. It also includes sample code for executing it from various languages.</td>
</tr>
<tr>
<td>service_id</td>
<td>this is a unique GUID identifying the service in your workspace. You can re-use this ID to update the service once it's published</td>
</tr>
</tbody>
</table>