Smart Applications for ever-smarter Phones

Presenter: Simone Scardapane
What is “smart”? 
Your phone is “smart”... But just HOW smart?

- Performing a lot of functions is not enough
- Collecting data is not enough
- Being user-friendly is not enough
- Taking active decisions is smart
Machine Learning
Machine learning algorithms allow you to “learn” from previous data and adapt to the user experience.
Sample Training Data

<table>
<thead>
<tr>
<th>Example Value</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.71, &quot;M&quot;, 1.74, 1.62, &quot;France&quot;</td>
<td></td>
</tr>
<tr>
<td>1.51, &quot;F&quot;, 1.61, 1.50, &quot;India&quot;</td>
<td></td>
</tr>
<tr>
<td>1.81, &quot;M&quot;, 1.82, 1.66, &quot;Norway&quot;</td>
<td></td>
</tr>
<tr>
<td>1.62, &quot;F&quot;, 1.70, 1.60, &quot;Mali&quot;</td>
<td></td>
</tr>
<tr>
<td>1.69, &quot;M&quot;, 1.74, 1.60, &quot;France&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Examples

Numeric Features

Text Features
What is “data”? What do we “learn”?

- Song + user preferences → song suggestions
- Images + tag of objects → object recognition
- Tweet + emotional content → tweet mood recognition
Native machine learning service for user adaptation on a mobile platform
US 8429103 B1

ESTRATTO

Disclosed are apparatus and methods for providing machine-learning services. A machine-learning service executing on a mobile platform can receive data related to a plurality of features. In some cases, the received data can include data related to features received from an application and data related to features received from the mobile platform. The machine-learning service can determine at least one feature based on the received data. The machine-learning service can generate an output by performing a machine-learning operation on the at least one feature. The machine-learning operation can be selected from among an operation of ranking the at least one feature, an operation of classifying the at least one feature, an operation of predicting the at least one feature, and an operation of clustering the at least one feature. The machine-learning service can send the output.

IMMAGINI (30)
“Smart” applications
Source: http://rottmann.net/2012/12/enable-google-now-google-apps-for-business/
The view was awesome.

Then view was awesome.

The view was awesome.

The View Was Awesome.

The view was awesome.
http://www.telegraph.co.uk/technology/mobile-app-reviews/10413900/Shoto-photo-app-launches-on-Android-and-iOS.html
Training is *computationally intensive* (but may be not needed)...

Solution 1: lightweight training on client (difficult!)

Solution 2: training and classification made on server

Solution 3: interact with a cloud-based service
How to do it 1:
Machine Learning Backend
The simplest solution is to put the machine learning logic on the back-end. Consider the popular WEKA software:

1. Ampio numero di algoritmi disponibili
2. Portabile
3. Sviluppo tramite interfaccia grafica
Because code is like rhyme.
Loading a dataset:

```java
Datasource source = new Datasource("/some/where/data.arff");
Instances data = source.getDataSet();
```

You can also retrieve it from a database with a JDBC driver.
Let us build a classifier:

```java
J48 tree = new J48();
tree.buildClassifier(dataset);
```

Now you have a machine learning model!
Attribute a1 = new Attribute("first_name", 1);
FastVector attrs = new FastVector();
attrs.addElement(a1);

Instance i1 = new Instance(2);
i1.setValue(a1, "Simone");
We can predict a new value easily:

```java
Instances dataset = new Instances("people", attrs, 1);
dataset.add(i1);

tree.classifyInstance(dataset.instance(i));
```

Or you can “update” some models incrementally.
How to do it 2:
Google Prediction
Lato utente

Google Cloud Storage

Copia Dati

Google Prediction Engine

Modello

Dati (file .csv)

Allenamento

Richiesta

Predizione

Utente
String modelID = "languageidentifier";
List<Object> params = new ArrayList<Object>();

// Initialize HTTP Transport
// Enter elements in List

Prediction prediction = new Prediction(httpTransport, requestInitializer, jsonFactory);
Input input = new Input();
InputInput inputInput = new InputInput();
inputInput.setCsvInstance(params);
input.setInput(inputInput);
Output output = prediction.trainedmodels().predict(modelId, input).execute();
200 OK

- Show headers -

- {  
  "kind": "prediction#training",
  "id": "fuelconsumption",
  "selfLink": "https://www.googleapis.com/prediction/v1.6/projects/61234350:
  "created": "2013-07-09T10:48:19.580Z",
  "trainingComplete": "2013-07-09T10:51:31.233Z",
  "modelInfo": {  
    "numberInstances": "398",
    "modelType": "classification",
    "numberLabels": "3",
    "classificationAccuracy": "0.79"
  },
  "trainingStatus": "DONE"
}
