Introduction to Machine Learning
Using Python

Vikram Kamath
Contents:

1. Introduction/Definition
2. Where and Why ML is used
3. Types of Learning
4. Supervised Learning – Linear Regression & Gradient Descent
5. Code Example
7. Code Example
8. Neural Networks
9. Code Example
10. Introduction to Scikit-Learn
**Definition:**

A computer program is said to 'learn' from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$. 
Example:
Classification: Digit Recognition

Input ($X_i$): Image Features

Output ($Y$): Class Labels \{ $y^0$, $y^1$, $y^9$ \}

Features($X_i$):
Proportion of pixels in Each of the 12 cells $X_i$ where $i=1,2,\ldots,12$

\[
x_i^0 = 0-10\%
\]
\[
x_i^1 = 10-20\%
\]

\[
Val(X_i) = 10
\]

No of parameters $= 10^{12} - 1$
Handcrafted Rules will result in a large number of rules and exceptions.

- We need ML in cases where we cannot directly write a program to handle every case.

So it's better to have a machine that learns from a large training set.

So, according to the definition earlier:

Task (T): recognizing and classifying handwritten words within images.

Performance measure (P): percent of words correctly classified.

Training experience (E): a database of handwritten words with given classifications.
Speech Recognition
When humans are unable to explain their expertise
Where ML is used...
Major Classes of Learning Algorithms:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
Supervised Learning:

- The set of data (training data) consists of a set of input data and correct responses corresponding to every piece of data.

- Based on this training data, the algorithm has to generalize such that it is able to correctly (or with a low margin of error) respond to all possible inputs.

- **In essence:** The algorithm should produce sensible outputs for inputs that weren't encountered during training.

- Also called learning from exemplars
Supervised Learning

\{ 
\text{Regression Problems} \\
\text{Classification Problems} 
\}
**Supervised Learning: Classification Problems**

“Consists of taking input vectors and deciding which of the N classes they belong to, based on training from exemplars of each class.”

- Is discrete *(most of the time)*. i.e. an example belongs to precisely one class, and the set of classes covers the whole possible output space.

**How it's done:** Find *decision boundaries* that can be used to separate out the different classes.

Given the features that are used as inputs to the classifier, we need to identify some values of those features that will enable us to decide which class the current input belongs to.
**Supervised Learning**: Regression Problems

Given some data, you assume that those values come from some sort of function and try to find out what the function is.

In essence: You try to fit a mathematical function that describes a curve, such that the curve passes as close as possible to all the data points.

So, regression is essentially a problem of function approximation or interpolation.
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5236</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5708</td>
<td>3.0</td>
</tr>
<tr>
<td>2.0944</td>
<td>-2.5981</td>
</tr>
<tr>
<td>2.6180</td>
<td>1.5</td>
</tr>
<tr>
<td>3.1416</td>
<td>0</td>
</tr>
</tbody>
</table>

**To Find:** $y$ at $x = 0.44$
Unsupervised Learning:

- Conceptually Different Problem.

- No information about correct outputs are available.

- No Regression No guesses about the function can be made

-Classification?

No information about the correct classes. But if we design our algorithm so that it exploits similarities between inputs so as to cluster inputs that are similar together, this might perform classification automatically

In essence: The aim of unsupervised learning is to find clusters of similar inputs in the data without being explicitly told that some datapoints belong to one class and the other in other classes. The algorithm has to discover this similarity by itself
Training Text Documents, Images, Sounds...

features vectors

Machine Learning Algorithm

New Text Document, Image, Sound...

features vector

Model

Likelihood or Cluster Id or Better representation
Reinforcement Learning:
Stands in the middle ground between supervised and unsupervised learning.

The algorithm is provided information about whether or not the answer is correct but not how to improve it.

The reinforcement learner has to try out different strategies and see which works best.

In essence: The algorithm searches over the state space of possible inputs and outputs in order to maximize a reward.
Good Robot Bad Robot
Supervised Learning: Linear Regression & Gradient Descent
**Notation:**

- $m$: Number of training examples
- $x$: Input variables (Features)
- $y$: Output variables (Targets)
- $(x, y)$: Training Example (Represents 1 row on the table)
- $(x^{(i)}, y^{(i)})$: $ith$ training example (Represents $ith$ row on the table)
- $n$: Number of features (Dimensionality of the input)
Representation of the Hypothesis (Function):

- In this case, we represent 't' as a linear combination of the inputs (x)

- Which leads to:

\[ h(\Theta) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 \]

where (\Theta_i 's) are the parameters (also called weights).

For convenience and ease of representation: \( x_0 = 1 \)

So that, the above equation becomes:

\[ h(x) = \sum_{i=0}^{n} (\Theta^T x) \]

The objective now, is to 'learn' the parameters \( \Theta \)

So that \( h(x) \) becomes as close to 'y' at least for the training set.
Define a function that measures for each value of the theta's, how close the \( h(x^{(i)}) \)'s are to the corresponding \( y^{(i)} \)'s.

We define the 'cost function' as:

\[
J(\Theta) = \frac{1}{2} \sum_{i=1}^{m} ( h(\Theta)(x^{(i)}) - y^{(i)} )^2
\]

We want to choose the parameters so as to minimize \( J(\Theta) \).

The LMS (Least Mean Squares) algorithm begins with some initial value of \( \Theta \) and repeatedly changes \( \Theta \) so as to make \( J(\Theta) \) smaller.
We now come to the **Gradient Descent** algorithm:

Gradient Descent starts off with some initial $\Theta$, and continually performs the following update:

$$
\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\Theta)
$$

(This update is simultaneously performed for all values of $j=0,1,...,n$)

$\alpha$ is called the **learning rate**

This, in effect assumes the following form:

$$
\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \theta_j} \left( \frac{1}{2} \sum_{i=1}^{m} \left( h_{\Theta}(x^{(i)}) - y^{(i)} \right)^2 \right)
$$

I'll leave this to you :)
A little mathematical 'hacking' of the above equation yields: (for a single training example)

\[
\Theta_j := \Theta_j + \alpha (y^{(i)} - h_\Theta(x^{(i)})) x_j^{(i)}
\]

There are 2 ways of generalizing the above equation for more than one training example:

The first one is:

Repeat until convergence

\{
\Theta_j := \Theta_j - \alpha \sum_{i=1}^{m} (y^{(i)} - h_\Theta(x^{(i)})) x_j^{(i)} \quad \text{For every } j
\}

This above method is called batch gradient descent
The second one is:

Loop
{
    for i=1 to m
    {
        \[ \Theta_j := \Theta_j + \alpha(y^{(i)} - h_\Theta(x^{(i)}))x_j^{(i)} \] (for every j)
    }
}

This is called stochastic gradient descent or incremental gradient descent.
Code Time
In this example, we generate random points and try use Stochastic Gradient Descent to fit a straight line.
Unsupervised Learning:
Clustering & K-Means
Clustering

Clustering is considered to be the most important unsupervised learning problem.

Deals with finding structure in unlabeled data

i.e. unlike supervised learning, target data isn't provided

**In essence:**
Clustering is “the process of organizing objects into groups whose members are similar in some way”.

A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.
The Goals of Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data.

But how does one decide what constitutes a good clustering? It can be shown that there is no absolute “best” criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.
Most common applications:

**Marketing:** finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records

**Biology:** classification of plants and animals given their features

**Insurance:** Fraud Detection

**City-planning:** identifying groups of houses according to their house type, value and geographical location;

**Earthquake studies:** clustering observed earthquake epicenters to identify dangerous zones;

**WWW:** document classification; clustering clickstream data to discover groups of similar access patterns. Creating recommender systems
The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.

2. Assign each object to the group that has the closest centroid.

3. When all objects have been assigned, recalculate the positions of the K centroids.

4. Repeat steps 2 and 3 until the centroids no longer move.
Code Time
Meet the data: The Iris Dataset

No. of training examples (instances): 150

Number of features (x’s) : 4

Number of classes : 3

Attribute Information:
Features:
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Classes:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica
Neural Networks
Motivation:

Animals learn and learning occurs within the brain.

If we can understand how the brain works then there are probably things that we can copy and use for our machine learning system.

The brain is massively complex and impressively powerful, but the basic atomic building blocks are simple and easy to understand.

The brain does exactly what we want it to. It deals with noisy and inconsistent data, and produces answers that are usually correct from very high dimensional data (like images) very quickly.
Basic Processing unit of the brain are **neurons**

Each neuron can be thought of as a processor. Each performing a very simple computation: deciding whether to fire or not.

The brain is hence a massively parallel computer made up of billions of 'processors'

**How does learning occur in the brain?**

**Plasticity:** modifying the strength of connections between neurons and creating new connections
Hebbs Rule:

“Changes in the strength of interneuron (synaptic) connections are proportional to the correlation in the firing of the two connecting neurons.”

Basically: “Neurons that fire together, wire together”
### Before conditioning

<table>
<thead>
<tr>
<th>FOOD</th>
<th>SALIVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Bone" /></td>
<td><img src="image2.png" alt="Dog" /></td>
</tr>
</tbody>
</table>

### During conditioning

<table>
<thead>
<tr>
<th>BELL + FOOD</th>
<th>SALIVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Bell" /></td>
<td><img src="image1.png" alt="Bone" /></td>
</tr>
</tbody>
</table>

### After conditioning

<table>
<thead>
<tr>
<th>BELL</th>
<th>SALIVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Bell" /></td>
<td><img src="image2.png" alt="Dog" /></td>
</tr>
</tbody>
</table>
WATCH WHAT I CAN MAKE PAVLOV DO. AS SOON AS I DROOL, HE’LL SMILE AND WRITE IN HIS LITTLE BOOK.
The Perceptron:
Code Time
Resources:

Datasets:

UCI Repo: http://archive.ics.uci.edu/ml/

Discussions

Machine Learning subReddit: http://www.reddit.com/r/MachineLearning/

Books:
Pattern Classification (Duda, Hart)
Machine Learning (Tom Mitchell)
Introduction to Machine Learning (Ethem Alpaydin)
Neural Networks (Simon Haykin)
Machine Learning: an algorithmic approach (Marsland)
Thank You

Kosshans?