**Course Description**

This course provides a broad introduction to machine learning and statistical pattern recognition. Topics include: supervised learning (generative/discriminative learning, parametric/non-parametric learning, neural networks, support vector machines); unsupervised learning (clustering, dimensionality reduction, kernel methods); learning theory (bias/variance tradeoffs; VC theory; large margins); reinforcement learning and adaptive control. The course will also discuss recent applications of machine learning, such as to robotic control, data mining, autonomous navigation, bioinformatics, speech recognition, and text and web data processing.

**Prerequisites**

Students are expected to have the following background:

* Knowledge of basic computer science principles and skills, at a level sufficient to write a reasonably non-trivial computer program.
* Familiarity with the probability theory. (CS 109 or STATS 116)
* Familiarity with linear algebra (any one of Math 104, Math 113, or CS 205)

All REQUIRED COURSE MATERIALS ON : http://cs229.stanford.edu/materials.html

------------------------------------------------------------------------------------------------------------------------------------------

DAY 1

Hello guys. Roshan here, today I am gonna talk about the ML course provided by Stanford Uni (Andrew NG). So lets do it. Isn’t ML the most interesting topic? ML has large impact in our day to day lives including economy, science, engineering, health etc.

Q. What is Machine Learning?

* A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T , as measured by P, improves with experience E .

Syllabus :

1. **Supervised learning.**

Intro to Machine Learning   
 Supervised learning setup. LMS.  
 Logistic regression. Perceptron. Exponential family.   
  Generative learning algorithms. Gaussian discriminant analysis. Naive Bayes.   
 Support vector machines.   
 Model selection and feature selection.   
 Ensemble methods: Bagging, boosting.   
  Evaluating and debugging learning algorithms.

1. **Learning theory.**

Bias/variance tradeoff. Union and Chernoff/Hoeffding bounds.   
VC dimension. Worst case (online) learning.   
Practical advice on how to use learning algorithms.

1. **Unsupervised learning.**

Clustering. K-means.   
EM. Mixture of Gaussians.   
Factor analysis.   
PCA (Principal components analysis).   
ICA (Independent components analysis).

1. **Reinforcement learning and control.**

MDPs. Bellman equations.   
Value iteration and policy iteration.   
Linear quadratic regulation (LQR). LQG.   
Q-learning. Value function approximation.   
Policy search. Reinforce. POMDPs.

Let’s Start then with some general idea about each of above topics ……………….

* **Supervised Learning**

**Supervised learning** is the [machine learning](https://en.wikipedia.org/wiki/Machine_learning) task of inferring a function from labeled training data.[[1]](https://en.wikipedia.org/wiki/Supervised_learning#cite_note-1) The [training data](https://en.wikipedia.org/wiki/Training_set) consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

**Classes of Supervised Learning Problems:**

* **Regression Problem( Continuous Variables)**

Output can be guessed using two continuous variables range in regression problem. Example would be you selling a house in bay area, but you don’t know the actual selling price. So you can feed your algorithm a set of inputs including the price and specifications of previous houses sold, which acts as generalized formula generator to predict other output.

* **Classification Problem( Discrete Variables)**

Tumor can either be Malignant or Benign (harmless). So such output can be guessed using two Discrete variables 1 or 0. 1 means Malignant here. Such problems come under classification problem. In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [statistics](https://en.wikipedia.org/wiki/Statistics), **classification** is the problem of identifying to which of a set of [categories](https://en.wikipedia.org/wiki/Categorical_data) (sub-populations) a new [observation](https://en.wikipedia.org/wiki/Observation) belongs, on the basis of a [training set](https://en.wikipedia.org/wiki/Training_set) of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into ["spam" or "non-spam"](https://en.wikipedia.org/wiki/Spam_filtering) classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition).

In the terminology of machine learning,[[1]](https://en.wikipedia.org/wiki/Statistical_classification#cite_note-1) classification is considered an instance of [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning), i.e. learning where a training set of correctly identified observations is available.

* **Learning Theory**

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **computational learning theory** (or just **learning theory**) is a subfield of [Artificial Intelligence](https://en.wikipedia.org/wiki/Artificial_Intelligence) devoted to studying the design and analysis of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms

* **Unsupervised Learning**

**Unsupervised learning** is a type of machine **learning** algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common **unsupervised learning** method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.

**Unsupervised machine learning** is the [machine learning](https://en.wikipedia.org/wiki/Machine_learning) task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no objective evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) .  
Unsupervised learning can be implemented in pattern recognition and sound analysis . Example: Andrew Ng’s experiment of (Uno,Dos,Tres…. + One,Two,Three = separate countings) . Also images pixel clustering is an example of UL where cluster of pixels are drawn as per the images provided . Note: That sound analysis part can be done using Matlab .

* **Reinforcement Learning**

Reinforcement learning , unlike one decision taken in supervised learning , here to get a job done, a series of good or bad decisions are made which is then learnt by the machine to maximize the positives thereby minimizing the negatives and getting the result perfectly ! Eg : Training a dog requires both kind of Dog’s decision so that you give rewards and punishments accordingly and dog will learn that also accordingly taking its time which is Reinforcement Learning .

**Wiki says: Reinforcement learning** is an area of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) inspired by [behaviorist psychology](https://en.wikipedia.org/wiki/Behaviorism), concerned with how [software agents](https://en.wikipedia.org/wiki/Software_agent) ought to take [*actions*](https://en.wikipedia.org/wiki/Action_selection) in an *environment* so as to maximize some notion of cumulative *reward*.

DAY 2

Refer : <http://cs229.stanford.edu/notes/cs229-notes1.pdf> for the Notes !

Welcome back. Today we gonna continue to talk about Supervised learning Algorithms. So let’s start.

|  |  |
| --- | --- |
| Living area(Square feet) | Price(1000$) |
| 2104 | 400 |
| 1600 | 330 |
| 2400 | 369 |
| 1416 | 232 |
| …. | … |
| …. | … |

Let’s start by talking about a few examples of supervised learning problems. Suppose we have a dataset giving the living areas and prices of 47 houses from Portland, Oregon:

Given data like this, how can we learn to predict the prices of other houses in Portland, as a function of the size of their living areas?

Let,

X(i) = “input” variables (living area in this example)/input features

y(i) = “output”/target variable that we are trying to predict (price).

A pair ( x(i),y(i) ) is called a training example,

and the dataset that we’ll be using to learn—a list of m training examples {(x (i) , y(i) ); i = 1, . . . , m}—is called a **Training Set** .

Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. For historical reasons, this function h is called a **HYPOTHESIS**.

When the target variable that we’re trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a **classification problem**.

For ease, Let me use Linear representation to the Hypothesis. i.e.

h(x) = Ax+B --------🡪 Linear Equation on x (input)

**Part I Linear Regression**

To make our housing example more interesting, let’s consider a slightly richer dataset in which we also know the number of bedrooms in each house:

|  |  |  |
| --- | --- | --- |
| Living area(Square feet) | #bedrooms | Price(1000$) |
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 2 | 369 |
| 1416 | 1 | 232 |
| …. | … |  |
| …. | … |  |
| …. | … |  |

Here, the x’s are two-dimensional vectors in R^2 . For instance, x1^(i) is the living area of the i-th house in the training set, and x2(i) is its number of bedrooms. (In general, when designing a learning problem, it will be up to you to decide what features to choose, so if you are out in Portland gathering housing data, you might also decide to include other features such as whether each house has a fireplace, the number of bathrooms, and so on. We’ll say more about feature selection later, but for now let’s take the features as given.) To perform supervised learning, we must decide how we’re going to represent functions/hypotheses h in a computer. As an initial choice, let’s say we decide to approximate y as a linear function of x: **hθ(x) = θ0 + θ1.x1 + θ2.x2**

Here, the **θi ’s are the parameters** (also called weights) parameterizing the space of linear functions mapping from X to Y. When there is no risk of confusion, we will drop the θ subscript in hθ(x), and write it more simply as h(x). To simplify our notation, we also introduce the convention of letting **x0 = 1** (this is the intercept term), so that

**hθ(x) = θ0.x0 + θ1.x1 + θ2.x2** AND

**h(x) =**

where on the right-hand side above we are viewing θ and x both as vectors, and here n is the number of input variables (not counting x0).

NOTE :

The thing is , Gradient Descent is a type of algorithm used for Minimizing a function .So we use it to minimize the Cost function . Now if you ask what really is the Cost function : A cost function is something you want to minimize. For example, your cost function might be the sum of squared errors over your training set. Gradient descent is a method for finding the minimum of a function of multiple variables. So you can use gradient descent to minimize your cost function.

Now check this for further understanding:

<https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/>

**COST FUNCTION**

Now, given a training set, how do we pick, or learn, the parameters θ? One reasonable method seems to be to make h(x) close to y, at least for the training examples we have. To formalize this, we will define a function that measures, for each value of the θ’s, how close the h(x (i) )’s are to the corresponding y (i) ’s. We define the cost function:

J(θ) =

If you’ve seen linear regression before, you may recognize this as the familiar least-squares cost function that gives rise to the ordinary least squares regression model. Whether or not you have seen it previously, let’s keep going, and we’ll eventually show this to be a special case of a much broader family of algorithms.