UT Machine Learning: HomeWork1

Mohamad amin Katebsaber

September 27, 2019

1 Class Summary

Machine learning is a field of study in which we design proper instructions (algorithms) for computers, enabling them to learn for themselfs. Generally speaking based on the way computers improve using each algorithm, we can divide them into three types:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

1.1 Supervised Learning

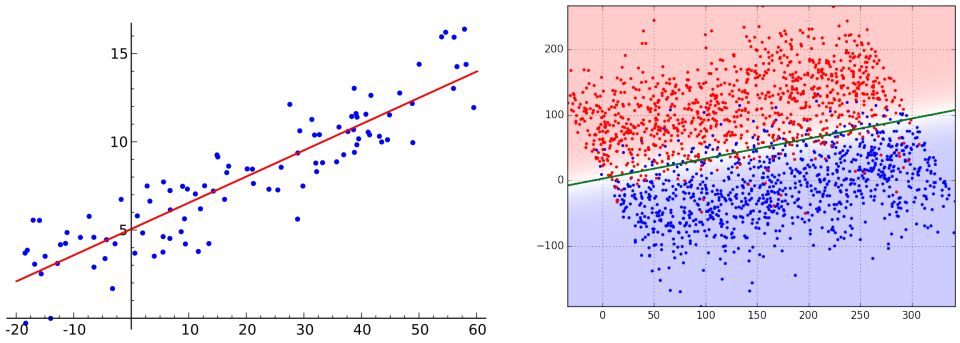
Supervised Learning algorithms can be considered as functions that map *input* space to *output* space. In this manner, *input* space (X) is a set of examples, each of which consisting one or several features that the *output* space (Y) depends upon. Here the objective is to best approximate the mapping function (f) such that new datapoints ($x \in \mathcal{X}$) can be accurately mapped to *output* space Y.Generaly there are two types of Supervised Learning based on the *output* space:

1. Regression: If the *output* space is numerical (continuous). i.e: Prdiction of Housing price in Tehran based on house area and number of rooms.
2. Classification: If the *output* space is categorical (descrete). i.e: Identification of digits (Predicting a label between 0-9) based on pixel intensities.

In Regression setting we aim to predict a real value on the calculated hyperplane but in Classification setting the calculated hyperplane is used as a boundary to devide input space to several zones (Figure **??**).

1.2 Unsupervised Learning

Unsupervised Learning is referred to a class of algorithms capable of learning previously unknows patterns among *inputdata* (there is no defined label for data points). These kind of algorithms are generally used for addressing two types of problem settings:

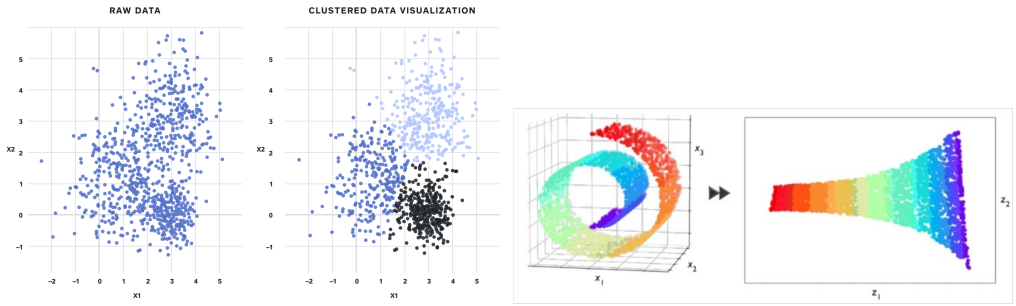


(a) Regression (b) Classification

Figure 1: Regression vs Classification

1. Clustering: In this setting we aim to group data space based upon features that represent the *inputdata*. Clustering is used when the *input* space is discrete (Figure **??**-a).

2. Dimensionality reduction: In Machine Learning problems there are often too many features to be considered for defining the *input* space. In many cases most of these features are by some means correlated and therefore redundant. Dimensionality reduction is the process of reducing the number of features decribing *input* space based on afformentioned correlation, resulting in model's simplicity. This technique is often used when our *input* space is continuous (Figure **??**-b).



(a) Clustering (b) Dimensionality reduction

Figure 2: Dimentionality Reduction

As an example for Uunsupervised Learning we can name Blind Source Separation. Assume that you are in cocktail party where lots of individuals are talking together. In this house there are four microphones recording conversations from different points in the room. It can be concluded that depending on the position of each microphone and its proximity to a conversation, each microphone will record a specific signal. Problem of interest is about separation of signals related to each conversation from other signals (Figure **??**).

The idea behind Blind Source Separation (BSS) can be generalized to a spectrum of problems in Bioinformatics. One of the Bioinformatics problems that can be framed in BSS

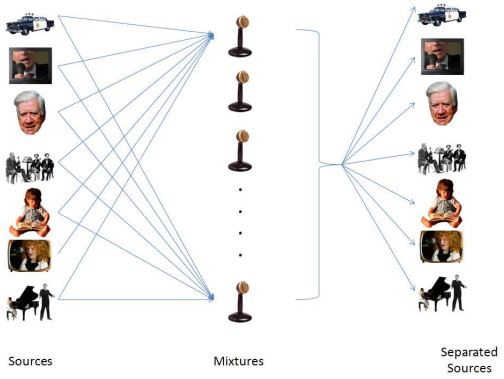
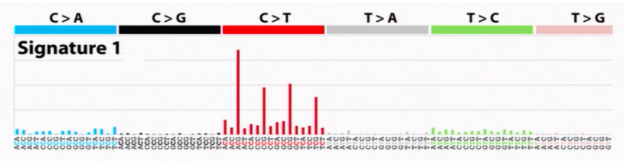


Figure 3: Blind source separation

settings is separation of Cancer Signatures. We know that the root of any cancer is some kind of somatic mutation. There are only six distinct base substitutions (C to A, C to G, C to T, T to A, T to C, T to G) but too many cancer types. Hence it can be concluded that accumulation of these substitutions is the key difference to cause different cancers.

Analysis of the total number of each base substitution type in cancerous genoms reveals that for example in skin cancers, C to T mutations are in majority which makes sense because we know from experiment that UV causes C to T mutations. Considering the six types of base substitutions in the context of immediate previous and next base in the sequence (for example C to T mutation when previous base is A and next base is T) we would have 96 different classes of mutations. Different distributions of mutation classes are considered as cancer signatures (Figure **??**).

(a) Mutation Classes



(b) A cancer signature

Figure 4: From Mutation clases to cancer signatures

Now our challenge is to find the contribution of each cancer signature to a specific cancer type when our *input* data is the accumulated version of base substitutions in a sequence extracted from a cancer cell.

1.3 Reinforcement Learning

The key idea behind Reinforcement Learning (RL) algorithms is the game of "carrot and sticks". In this context we have an agent (here our RL algorithm) taking some actions and getting rewards (and of course punishments!) from the interpreter who observes the game state consequent to taking that action from the agent. Think of a mouse (agent) in a maze (environment) trying his best (taking actions) to avoid traps (major and minor punishments) and getting to cheese (reward). In this case the designer of the game is the interpreter who either automatically or manually punishes or rewards the mouse (Figure **??**).

Figure 5: Reinforcement Learning

After understanding the basics of Machine Learning it is time to roll up our sleeves and deep dive into the math behind the stuff.

1.4 Linear regression

Assume that we have some training data:

$$(x[1], y[1])$$

$$(x[2], y[2])$$

$$(x[3], y[3])$$

$$...$$

$$(x[n], y[n])$$

where n denotes the number of our samples. We want to create a model which passes through all our data points and given a x can output its respective y. In a context where the data is 2-dimensional we can fit a simple line to our sample points:

$$y = \beta_0 + \beta_1 x$$

This model consists of two parameters $\beta_0 and \beta_1$ which their value depend on our data. To find the best parameters, we have to have some notion of accuracy to evaluate our correctness (or in Machine Learning notation Falseness!). To address this problem we can define a function which takes our data and model (parameters) as input and outputs a number indicating how off we are:

$$L(\beta_0, \beta_1, data) = \frac{1}{2} \sum_{i=1}^{n} y[i] - (\beta_0 + \beta_1 x[i])^2$$

Our objective is to find β_0 and β_1 so our Loss function (L) is at its minimum. Mathematically we can frame our objective as:

$$\beta = \beta_0, \beta_1$$

$$\hat{\beta}_0, \hat{\beta}_1 = \underset{\beta}{\operatorname{argmin}}\, L(\beta)$$

In order to get to find best candidate for β, we can use several methods:

1. Gradient descent
2. Analytic method

1.4.1 Gradient descent

Gradient descent is an iterative algorithm for finding best β. We start with an initial guess for β and use the following equations to update our initial guess:

$$\beta_0 \coloneqq \beta_0 - \alpha \frac{\partial L(\beta)}{\partial \beta_0}$$

$$\beta_1 \coloneqq \beta_1 - \alpha \frac{\partial L(\beta)}{\partial \beta_1}$$

In equations above, α is a parameter called "Learning rate" which we will discuss later in this course. For the partial derivatives in our case we have:

$$\frac{\partial L(\beta)}{\partial \beta_0} = \frac{1}{2}\sum_{i=1}^{n}(2)(-1)(y[i] - \beta_0 + \beta_1 x[i])$$

$$\frac{\partial L(\beta)}{\partial \beta_1} = \frac{1}{2}\sum_{i=1}^{n}(2)(-x[i])(y[i] - \beta_0 + \beta_1 x[i])$$

So the complete form of gradient descent algorithm would be:

$$\beta_0 := \beta_0 + \alpha\sum_{i=1}^{n}(y[i] - \beta_0 + \beta_1 x[i])$$

$$\beta_1 := \beta_1 + \alpha\sum_{i=1}^{n}(x[i])(y[i] - \beta_0 + \beta_1 x[i])$$

This flavor of gradient descent is called "Batch". In order to update our parameters in this form, we need to go through all of our data once. This task is computation intensive and as our data grows, It would be almost impossible to compute parameters using this equations. To face this challenge we introduce another form of gradient descent called "Stochastic":

$$\beta_0 := \beta_0 + \alpha(y[i] - \beta_0 + \beta_1 x[i])$$

$$\beta_1 := \beta_1 + \alpha(x[i])(y[i] - \beta_0 + \beta_1 x[i])$$

General notion of Stochastic Gradient Descent (SGD) is to update our parameters for each data point. In fact this alteration adds some noise to the minimization process but on the other hand, makes it possible to actually compute parameters using the formula (Figure **??**).

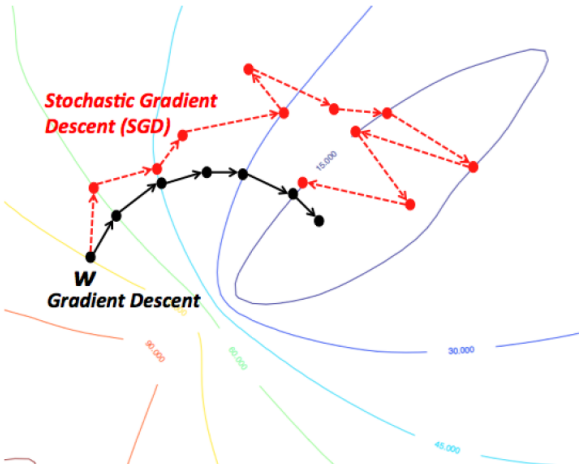


Figure 6: Batch vs Stochastic Gradient Descent