Machine Learning

Harish Tayyar Madabushi
www.harishtayyarmadabushi.com

These notes provide an overview of the topics covered in class and are not meant to replace the lectures. This is intended as supplementary material which will hopefully make it easier for you to revise the content of what was taught.

I will be referring to sections of "Artificial Intelligence: A Modern Approach, Global Edition by Stuart Russell, and Peter Norvig (2016). **Chapter 18, Learning from Examples**" You can access an online version of this through the resource list.

Introduction

Learning algorithms can broadly be classified into three different kinds of algorithms:

1. Unsupervised
2. Supervised
3. Reinforced

Unsupervised algorithms are those that classify input data based on certain inherent properties of that data. An example of such an algorithm is the k-means algorithm. This section of your course does not deal with these algorithms.

Supervised algorithms are those that require *annotated* data to learn the trend or classes of the input data before then being able to make predictions about previously unseen input. In this section of the course we study Linear and non-linear Regression, Logistic Regression and Neural Networks, all of which are supervised learning algorithms (because they need training data).

Reinforced Learning algorithms are required to achieve a specific task in a predefined environment and learn to do so based on "rewards" that inform the algorithm of how well it is doing in terms of getting closer to achieving that task.

Further Reading: Artificial Intelligence: A Modern Approach, Global Edition by Stuart Russell, and Peter Norvig (2016). **[Section 18.1]**

Supervised Learning

The majority of what we will discuss is related to supervised learning. Supervised learning is the process of learning some aspects of data that is annotated. This annotation will inform us as to what the values of the dependent variable "y" are for some (set) of input variables which are independant.

Independant variables take on different values based on the environment. **Dependent variables** take on values that are dependent on other variables, often independent variables.

These dependent variables can take on continuous values, such as in the case of the relation between temperature in degrees fahrenheit given temperature in degrees celsius, or might take on discrete values as in the case of whether or not someone has the flu. The prediction of continuous values is called "**Regression**" and the prediction of which class an element belongs to is called "**Classification**".

The data that is used to train a model is called the **training data**. In the context of supervised learning, training data always has the associated value of the dependent variable, usually referred to as "y".
We use the following notation to represent the training data:

$$(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), (X^{(3)}, y^{(3)}), \ldots (X^{(N)}, y^{(N)})$$

What this means is that for some input $X^{(1)}$, the corresponding output is $y^{(1)}$ and for $X^{(2)}$ it is $y^{(2)}$ and so on for the N training data elements that we have available. Notice how the X in the above is capitalised, whereas the y is not. This is to highlight the fact that "X" can consist of more than one element (Multivariate regression or classification discussed later)

The **test data** is an independent set of data that we test our model on. While it shares the characteristics of the training data, its elements are different from those in the training data.

The elements of Supervised Learning

Regardless of classification or regression, supervised learning models consist of the following elements :

1. Hypothesis Function
2. Cost Function
3. The (partial) differential of the cost function

The **hypothesis function** represents the hypothesis we have about how the input data ("X" or the independent variable) affects the value (or class) of the output data ("y" or the dependent variable). It is therefore an equation that relates these two and, *for regression*, is of the general form:

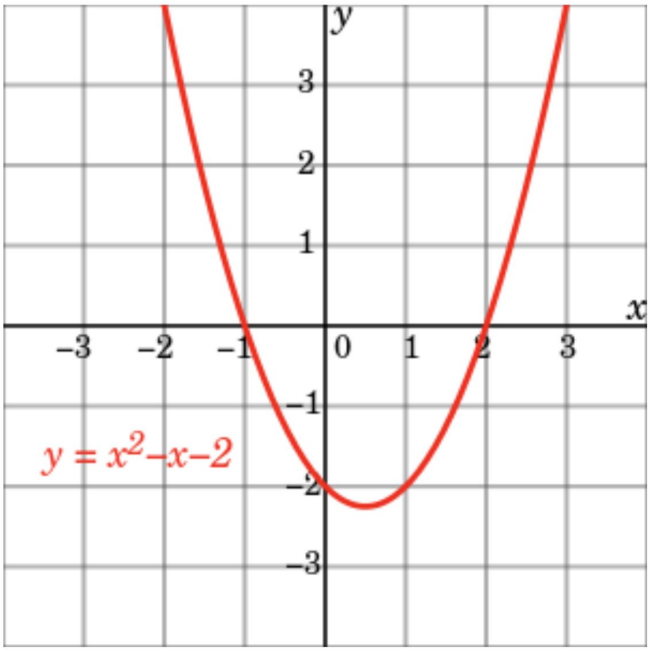
$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_kx_k \qquad \textit{Equation 1}$$

You might see the function $h_w(x)$ written as f(x) or h(x). In the above equation we assume that there are going to be an arbitrary number of terms (k). Remember that this does not necessarily mean that there are k independent variables, it just means that there are k terms, some of which might be polynomial terms of the form x^2 (x squared, not x superscript 2), x_1x_2 and so on (NOTE: These are non-linear terms and this will apply to non-linear regression).

In the above equation, w_0, w_1 … are referred to as **weights** and take on some numeric values. The purpose of all supervised learning algorithms is to find those values of w for which the hypothesis function best estimates the training data. This is also true of classification, although the hypothesis function is of a slightly different form in that case.

We measure how good or bad a hypothesis function is using the **cost function**. When the cost function is at its lowest the cost of the hypothesis function is at its minimum and therefore the hypothesis function is set to "fit" the training data well.

Recall that the differential of a function gives the equation of the slope of that function. When this slope is zero, we can determine the "point of inflection" of the original function. This was discussed in class using the following slide. You might want to listen in on the associated bits of the lecture:



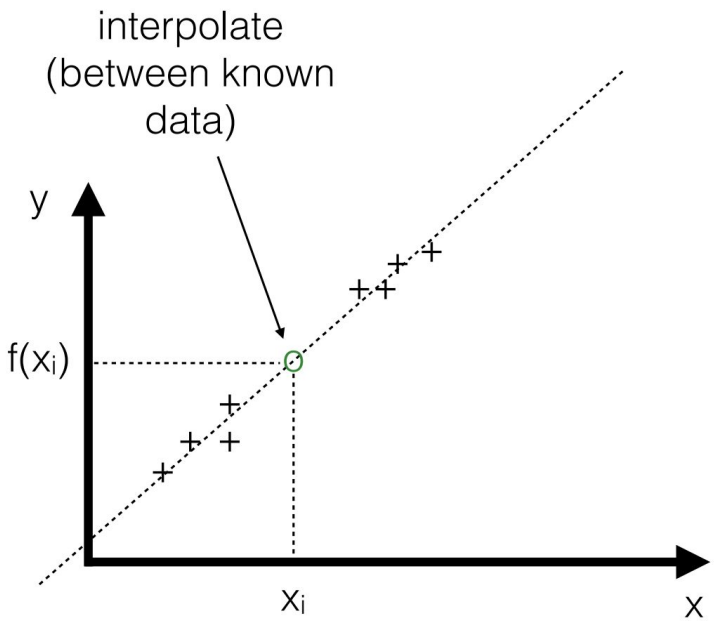
As finding the minimum of the cost function is not always trivial (as in the case of Non-Linear Multivariate Regression/Classification) we make use of an algorithm to do this. This algorithm which utilises the (partial) derivative of the cost function to find those values of w_0, w_1, w_2, … at which the cost is minimum is called **Gradient Descent**.

Further Reading: Artificial Intelligence: A Modern Approach, Global Edition by Stuart Russell, and Peter Norvig (2016). **[Section 18.2]**

Regression

Regression involves the use of training data to create a model so we can establish a trend in the training data which can be used to predict values of "y" given previously unseen values of "X".

An example from the lectures:



IntroA.I. M.Mistry Lect09 9/2/16

Univariate Linear Regression

Let us first consider the case where we have one independent variable "x" related to the dependent variable *linearly*. This problem is called **Univariate Linear Regression.**

Notice how, in this case, what we are trying to fit to the training data is a straight line. The equation of a straight line is y = mx + c, where m is the slope and c the y-intercept. This equation now represents our hypothesis because we know that some straight line "fits" our training data. The aim of Univariate Linear Regression is to find the *specific* line that fits the training data and this requires us to find the values of m and c.

We translate the equation of a straight line into the form of the general hypothesis function given by Equation 1 by replacing c by w_0 and m by w_1.

Hypothesis Function

For **Univariate Linear Regression**, the hypothesis function is of the form:

$$h_w(x) = w_0 + w_1 x \qquad \textit{Equation 2}$$

Cost Function

As discussed in the previous section, we must now establish how "bad" a particular line (defined by specific values of w_0 and w_1) is. This is done using the **Cost Function**.

There are two cost functions associated with Regression. The first is the average of the sum of the absolute value of the distance between the prediction and the observed value of "y" over all training examples. The second is the average of the sum of the squares of the same distance.

Mathematically, this is given by:

$$(1/m) \sum_{i=0}^{m} \text{abs}(y^{(i)} - h_w(x^{(i)})) \qquad \textit{Equation 3}$$

$$(1/m) \sum_{i=0}^{m} (y^{(i)} - h_w(x^{(i)}))^2 \qquad \textit{Equation 4}$$

In the above equations *m* is the number of training examples (the size of the training set) $y^{(i)}$ and $x^{(i)}$ the i^{th} output and input respectively, and h_w the hypothesis function, parametrized by some values of w. Notice that as we change the values of w, in an attempt to find the line that better fits the training data, the corresponding cost will change.

Equation 4 represents the more commonly used Cost and is called the L2-Cost (or L2-Loss for our purposes). While there is a subtle difference between “Loss” and “Cost”, we will use them interchangeably for the purpose of this course. Equation 4 is more common than Equation 3 as it more heavily penalises the hypothesis function for points that are “very far” from it.

Gradient Descent for Univariate Linear Regression

Gradient descent is an algorithm that allows one to progressively update the values of w so the Cost of the hypothesis function, parametrized by w, progressively reduces.

The general form of Gradient Descent is:

```
while not converged :
    for example j in observations/training data:
        update all w using loss on j
            (Simultaneously for all w)
```

What this says about the algorithm is that the values of w are updated based on the loss on each training element, j. This iteration over all training elements is called an **epoch**.

We run through this process multiple times until we reach a state where the algorithm is said to have **converged**. This is the state where the values of w represent a particular parameterization of the hypothesis function at which the Cost is minimal. One simple way to check for convergence is to wait until the cost of Consecutive epochs is very small.

In the case of Univariate Linear Regression, Gradient Descent is as follows:

```
while not converged :
    for example j in observations:
        w1 = w1 + α . ( yj - hw( xj ) ) . xj
        w0 = w0 + α . ( yj - hw( xj ) )
```

Regression with Multiple Variables and Polynomial Terms

There is no reason to limit our regression to one variable or to a linear hypothesis function. You are encouraged to pay specific attention to what these functions look like in the corresponding lectures.

Hypothesis Functions

For **Multivariate Linear Regression**, the hypothesis function is of the form:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 \qquad \textit{Equation 5}$$

For **Univariate Non-Linear Regression**, the hypothesis function is of the form:

$$h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + \ldots \qquad \textit{Equation 6}$$

Notice that the above equation can be quadratic, cubic or contain higher polynomials. **We will talk about how to choose a hypothesis function later in this document.**

Finally, for **Multivariate Non-Linear Regression**, the hypothesis function is of the form:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + \dots$$

Equation 7

Notice that the above equation can similarly contain multiple polynomial terms. Here are some different polynomial terms: x_1^2, x_2^3, x_1x_2

Cost Function

The Cost function for Regression is always the same. We continue to use the L2-Loss function which was given by Equation 4:

$$(1/m) \sum_{i=0}^{m} (y^{(i)} - h_w(x^{(i)}))^2$$

Equation 4

The difference, however, is that the function used to calculate the predicted values (h_w) changes based on the specific hypothesis function used.

Gradient Descent

Gradient Descent is also the same with two modifications: a) Like with the cost function the hypothesis function needs to be the one being optimised and b) multiple w values might need to be updated.

See lecture demonstration relating to this!

Logistic Regression

The most important thing to remember about logistic regression is that it is not regression. **Logistic regression is a form of classification** and uses the word "regression" for historical reasons.

Unlike regression which predicts a trend, logistic regression predicts if a particular sample is part of a class or not. As a consequence of this the output of the hypothesis function of logistic regression produces a number between 0 (not part of the class) and 1 (part of the class). Since the hypothesis function produces continuous values between 0 and 1 (as opposed to discrete values 0 or 1) the output is interpreted as the probability of the sample being part of the class.

Logistic Regression - Hypothesis function

The hypothesis function for Logistic Regression "pushes" the hypothesis function for regression through the sigmoid function (g). This is the only difference in each of the cases discussed for regression.

For **Univariate Linear Logistic Regression**, the hypothesis function is of the form:

$$h_w(x) = g(w_0 + w_1x_1) \qquad \textit{Equation 8}$$

For **Multivariate Linear Logistic Regression**, the hypothesis function is of the form:

$$h_w(x) = g(w_0 + w_1x_1 + w_2x_2) \qquad \textit{Equation 9}$$

For **Univariate Nonlinear Logistic Regression**, the hypothesis function is of the form:

$$h_w(x) = g(w_0 + w_1x + w_2x^2 + w_3x^3 + \dots)$$

Equation 10

Finally, for **Multivariate Non-Linear Logistic Regression**, the hypothesis function is of the form:

$$h_w(x) = g(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + \dots)$$

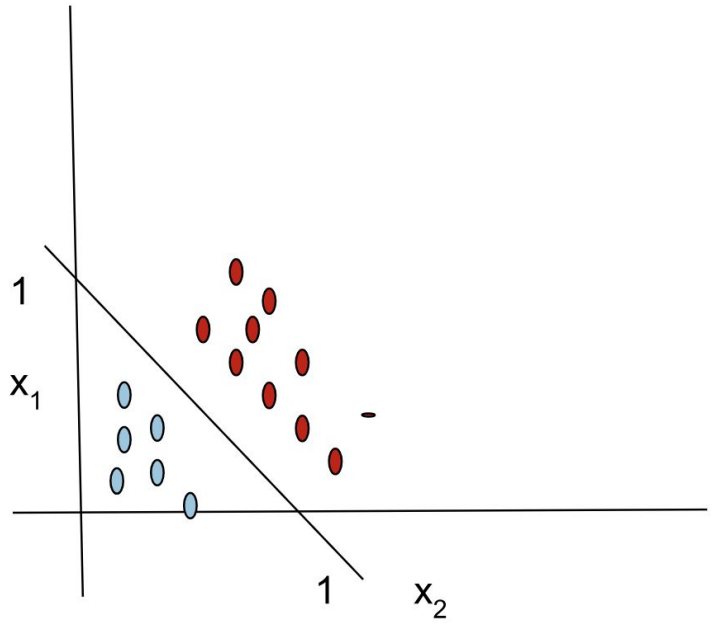
Equation 11

Logistic Regression - Decision Boundary

An important element of logistic regression to be aware of - which was previously not required in regression - is the understanding of how to plot the decision boundary. The decision boundary is an interpretation of the hypothesis function which allows us to visualise which regions of a plot will be assigned to the relevant class and which will not.

Recall that the hypothesis function was previously (in regression) plotted on a chart where *x* was the input and *y* the output. In logistic regression, however, we often have multiple independent variables (x_1 and x_2) and the decision boundary is plotted on the same chart.

The chart below provides a visual representation of samples associated with two independent variables, x_1 and x_2, the class they belong to (red or blue) and the decision boundary.



It is important to understand what the different elements on this chart represent. Notice that both axes (plural of axis) on this chart represent the *input* (of which there are two in this case). The decision boundary in the above chart connects the points (0,1) and (1,0). The points "below" the decision boundary are training examples that do not belong to the class (in blue) and above it, in red, are those that do.

At the decision boundary, the hypothesis function (be definition) evaluates to 0.5 (the probability of being a part of the class or not is exactly half). This implies that at the decision boundary

$$g(w_0 + w_1x_1 + w_2x_2) = 0.5$$

Recall that the sigmoid function outputs 0.5 when the input is 0:

$$\begin{aligned} & 1/(1 + e^{-0}) \\ = \quad & 1/(1 + 1) \\ = \quad & 0.5 \end{aligned}$$

And so, at the decision boundary:

$$g(w_0 + w_1x_1 + w_2x_2) = 0.5 \qquad \text{[As above]}$$
$$\text{Therefore} \quad w_0 + w_1x_1 + w_2x_2 = 0$$

Now going back to the chart with the decision boundary, remember it meets the axes at (0,1) and (1,0).

So when x_1 is 0, for $w_0 + w_1x_1 + w_2x_2$ to be 0, w_0 must be -1

You must work this out so you understand why this is the case. Write it out for both cases ($x_1 = 0$ and $x_2 = 0$) and solve the equations if this is not clear.

Logistic Regression - Cost function

Recall what a cost function is: It is a function that tells us how good or bad our hypothesis function is with respect to a set of training examples. The "closer" the hypothesis function's predictions are to the actual training examples, the better (with the exception of overfitting).

Once we have a cost function, we use it's differential to perform gradient descent. For gradient descent to work the cost function must be a convex function. The cost function we studied for Linear Regression cannot be used in Logistic Regression (which has the sigmoid function) as the resultant combination is not convex.

Convex

Not Convex

Gradient descent tends to get "stuck" at local minima if the cost function is not convex and since L1 and L2 loss are not convex, we need a new cost function.

The cost function that we will make use of for Logistic regression is derived using numerical methods that are beyond the scope of this module. Unlike in linear regression where we justified the specific cost function used, in logistic regression, we will first describe the cost function used and then show, by exploring its properties, that it is a reasonable cost function.

The cost function "J" is given by the equation:

$$J(\ w\ ) = 1/m \sum_{i=1}^{m} \text{Cost}(\ h_w(x^{(i)}),\ y^{(i)})$$

Equation 12

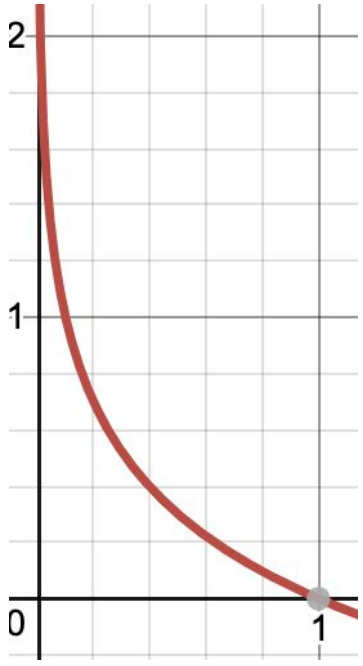
where m is the number of training elements, h_w is the hypothesis function parameterized by (the matrix, i.e. multiple values) w and $y^{(i)}$ and $x^{(i)}$ are the i^{th} output and input example respectively. *Cost* in the the above equation is the cost associated the prediction based on one training input and the corresponding training output and is given by the equation:

$$\text{Cost}(\, h_w(x), y \,) = \begin{cases} -\log(\, h_w(x) \,) & \text{if } y = 1 \\ -\log(\, 1 - h_w(x) \,) & \text{if } y = 0 \end{cases}$$

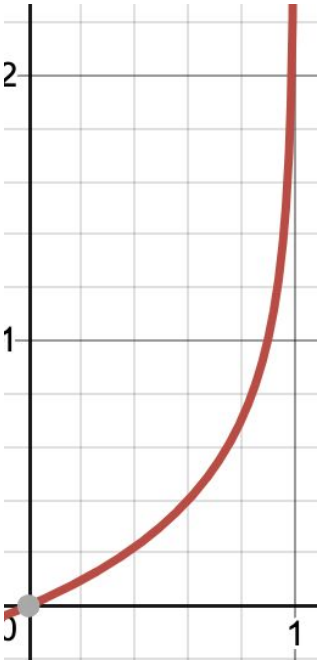
Equation 13

The first thing to note about *Equation* 13 is that it has no superscripts or subscripts for *x* and *y*. This must not confuse you: Notice that this cost is defined on one training example and so we don't need to "index" x and y.

Let us now try to understand what this cost function does and explain why it is a good cost function. Remember that we are not going to explore how we come up with this cost function.



-log(x)



-log(1-x)

Let us first consider the plot of *-log(x)* above. We make use of this cost if $y = 1$. What this cost tells us is that when $y = 1$ and $h_w(x) = 1$, then the cost is 0. However, as our prediction starts to move towards 0 (the opposite of what the true value is), the cost associated with this prediction starts to tend towards infinity. Symmetrically, *-log(1 - x)* will penalise an incorrect prediction exponentially, based on how far from the actual output the prediction is. You

are urged to spend some time thinking about why this cost function is a good cost function.

Now this two part cost function is not easy to differentiate and we need a single equation to represent Equation 13. The following Equation is equivalent to *Equation 13* and you are urged to work this out. The way to do it is to remember that *y* can take on the values of *0 or 1*. So replace the value of y with 0 and 1 in the following equation and see what it simplifies to.

$$J(w) = -1/m \; [\, y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)})(\log(1 - h_w(x^{(i)})) \,]$$

Equation 14

where *J* is the cost function associated with the hypothesis function h_w, parameterized by (the matrix) *w*, *m* is the number of training elements, and $y^{(i)}$ and $x^{(i)}$ are the output and input of the i^{th} training example.

Logistic Regression - Gradient Descent

The steps for gradient descent for logistic regression are the same as that for linear regression except that the hypothesis function used is that of logistic regression. Of course, we might have multiple w values, all of which need to be updated simultaneously:

```
while not converged :
     for example j in observations:
          (A) update each w_i using loss on j

          (A) w_i = w_i + α . ( y^(j) - h_w( x^(j) ) ) . x_i^(j)
                    Simultaneously for all w_i
```

where w_i is the i^{th} parameter of h_w , *alpha* the learning rate, $y^{(j)}$ and $x^{(j)}$ the j^{th} output and input of the j^{th} example respectively, and $x_i^{(j)}$ the i^{th} training input (associated with w_i)

Notes on Notes

Remember that these notes are supplementary material and do not cover overfitting, underfitting, Neural networks, hyperparameter optimisation, and similar.

These notes are better understood with the help of the lecture videos.