1 Unified Framework

Unified Framework in Artificial Intelligence, in a nutshell, is the general map of how inputs are processed and how outputs are derived from necessary functions.

Basically, training inputs are picked apart into concerned features through **basis functions**, and when graphed in a suitable coordinate system, they are called **coordinate vectors**. Now, we have a collection of those coordinate vectors that we can make use of later on. To use these data, we have to have an input, say, a requirement, to get the data we want. The input requirement then will also be picked apart into necessary coordinate vectors that will be used to compare it with our data set.

But there is a problem! We cannot compare incompatible things together, like comparing a stone and a human, we cannot get a definable similarity and difference. What we need to do is to make all those disparate coordinate vectors comparable. How? By projecting them into a same interface. Let's look at an example.

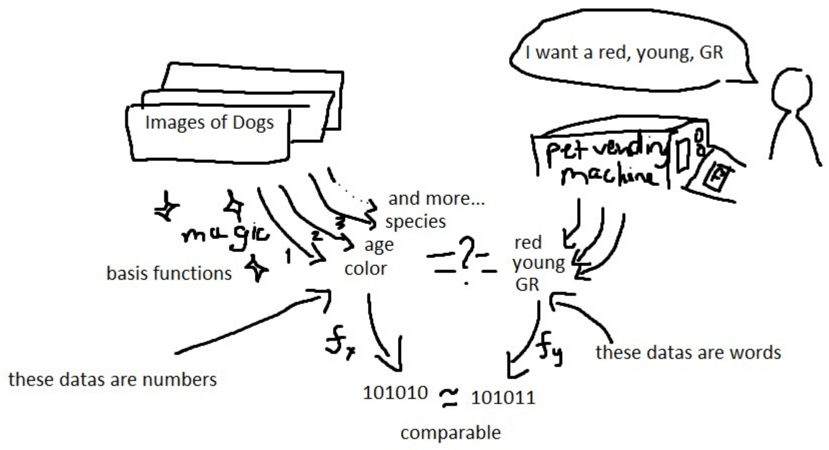


Image 1.1: Unified Framwork in an AI Dog Vending Machine

Example: You are going out to buy a dog, and when driving through a corner, you catch a glimpse of a dog vending machine. "Cool," you thought, "let's give it a try". You walk up to the machine, insert a hundred dollar bill, state your order, and voila, your brand new pet. Driving home with your new best companion, you cannot stop thinking about how that weird machine works.

The answer is simple: The machine has already been trained of thousands and thousands of dog images, and inside its data cache is a huge collection of features, or **coordinate vectors**, that was taken from the myriads of dog images using "magic", or **basis functions**. When you speak your orders, the machine *also* picks apart those words, going through a different "magic", but before actually finding your dog, the machine has to make its data collection and your order **comparable**. Going through yet another "*magic*", the data collection and the order now reside in a same place, same properties. The last thing to do is to find the set of data in the dog collection that has the **minimum** error compared to your order. Magical, is it not?

2 PCA (Principal Component Analysis)

You are confused by my explanation about how the machine picks apart a bunch of picture. Very, in fact. "Why, and how?", enraged you walk to the vending dog machine with a sledgehammer, trying to *actually* picking the machine apart to take its magic. Before you do something radical, let me assure you that you will get nothing but a bunch of chips and boards (you may even kill some puppies in it!), and there is no magic. The *magic* that I have mentioned above, that can analyze important features of the pictures, is a concept called **PCA**, or **Principal Component Analysis**.

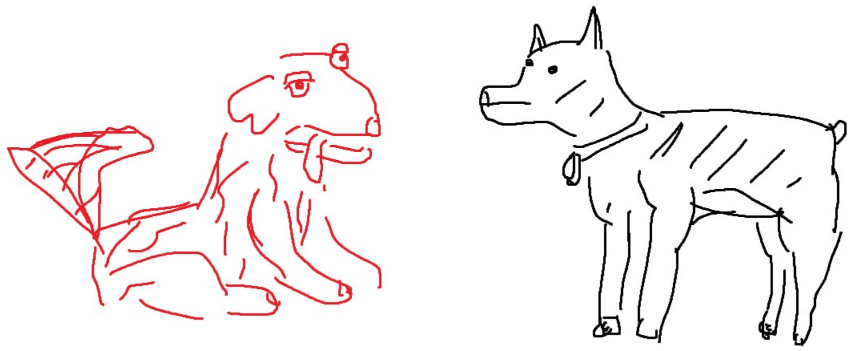
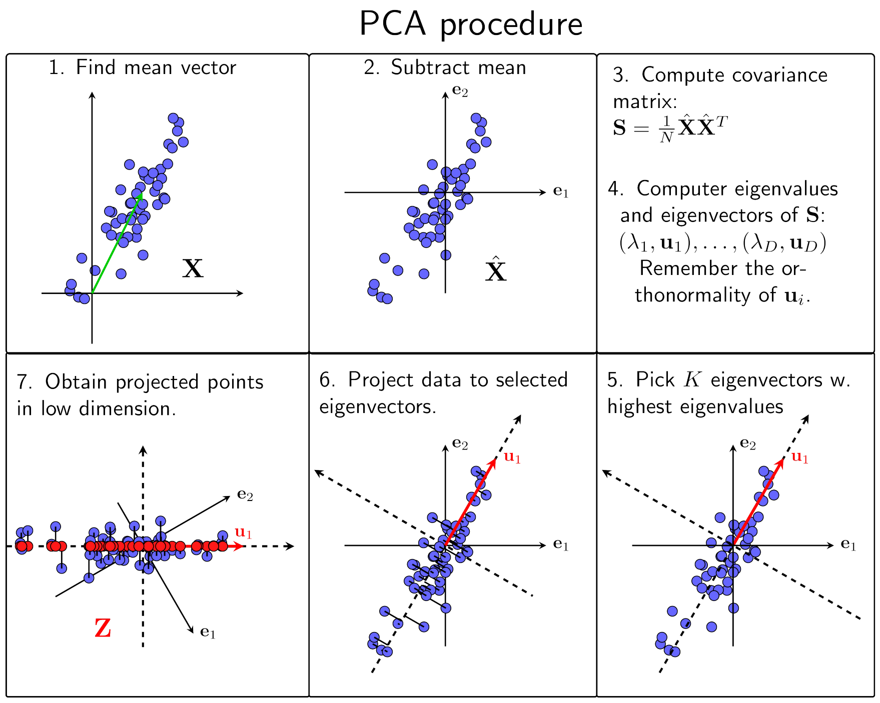


Image 2.1: Picture of your dog and neighbor Ray's German Shepherd

PCA is smart, I tell you. Look at two of these dogs. What are the differences between your dog and Ray's? The color, the breed, the posture of the dogs, it is obvious, isn't it? You readily know which is different, and what would be the features that **most** differs between the two dogs. ***PCA*** *do the exact same thing.* **It mathematically picks out the feature that varies the most among the many objects it observed**. Train the machine to do just that a couple of a hundred more and you get yourself your own nice and well-equipped dog collection.

Now drop that sledgehammer.

Extra information about how PCA works in theory.


PCA procedure
1. Find mean vector
$\mathbf{X}$
2. Subtract mean
$\mathbf{e}_2$
$\mathbf{e}_1$
$\hat{\mathbf{X}}$
3. Compute covariance matrix: $\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$
4. Computer eigenvalues and eigenvectors of $\mathbf{S}$: $(\lambda_1, \mathbf{u}_1), \ldots, (\lambda_D, \mathbf{u}_D)$ Remember the orthonormality of $\mathbf{u}_i$.
7. Obtain projected points in low dimension.
$\mathbf{Z}$
$\mathbf{u}_1$
6. Project data to selected eigenvectors.
5. Pick K eigenvectors w. highest eigenvalues

3 Linear Regression

You seems satisfied that the magic is not magic at all. "All math and stuff, huh?", you murmur,"I am definitely not touching those." Suddenly, you catch the glimpse of my Fx scribble. "Wait what is that?" you ask, assuring yourself that curiosity killed the cat, not you. Well, it sure killed the cat, but it is also killing my time for sure.

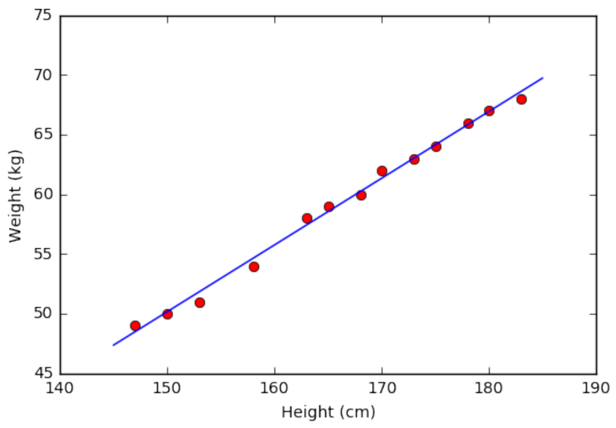


Image 3.1: Graph that represents linear regression

Linear regression is the simplest way to convert your dog collection, or raw and incompatible data generally, to something that would be comparable to your input data.

Let's take two data types from your dog collection - height and weight - and graph it.

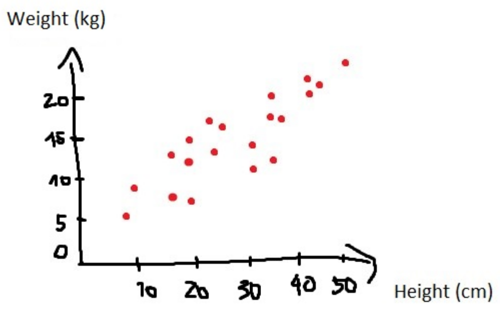


Image 3.2: Graph of dogs' height versus weight, data taken from the Dog Collection (imaginary)

Looking at this graph, as a human, you can obviously see there is a trend in the dog's height versus weight. Using mathematical functions, a machine can "see" this trend through the means of the *line of best fit*, and from there, it can predict height or weight of a dog with a more or less acceptable errors. For example your red Golden Retriever is 40cm in height. According to this machine-generated trend line, your dog should weigh about 20kg. (graph shown in next page)

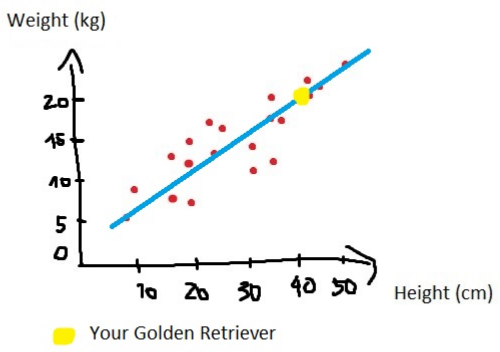


Image 3.3: Your Golden Retriever's data on the graph

Though keep in mind that **Linear Regression** is not the optimal function to make your data comparable to the input. Due to the properties of the linearity, the line of best fit can be terribly wrong if there is an extreme outlier.

4 Logistic Regression

Beside linear regression, there are many more kind of regressions that can be used for many purposes. For example, the above linear regression is used in fitting. But is *all* kind of regressions for fitting. "Interesting" you says to yourself. Well, there sure is.

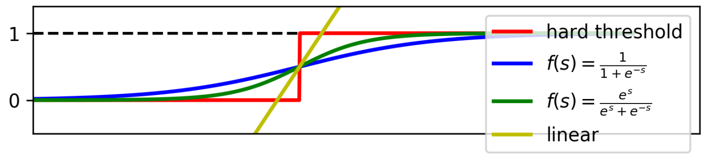


Image 4.1: Graph comparing between different regressive methods

Logistic Regression is just another regression, fundamentally. Though also used in fitting, this kind of regression is used for **classification**, as it uses a sigmoid equation, like this

$$\frac{L}{1+e^{-k(x-x_0)}}$$

or more simply

$$\frac{1}{1+e^{-s}}$$

to have a result which is a number between 0 and 1, that would allow the machine to classify accordingly. But how does it work?

Classification, as it name suggests, is the method of classifying objects into different categories, like your Golden Retriever and your neighbor's German Shepherd is a *dog*, and the Russian Blue I have at home is a *cat*. But how can a machine do it, with no whatsoever the elaborate senses of humans?

Easily and logically, **classification** works on the basis of probability. In order to understand more what in the world I am talking about, let see how **classification** works in the AI unified framework.

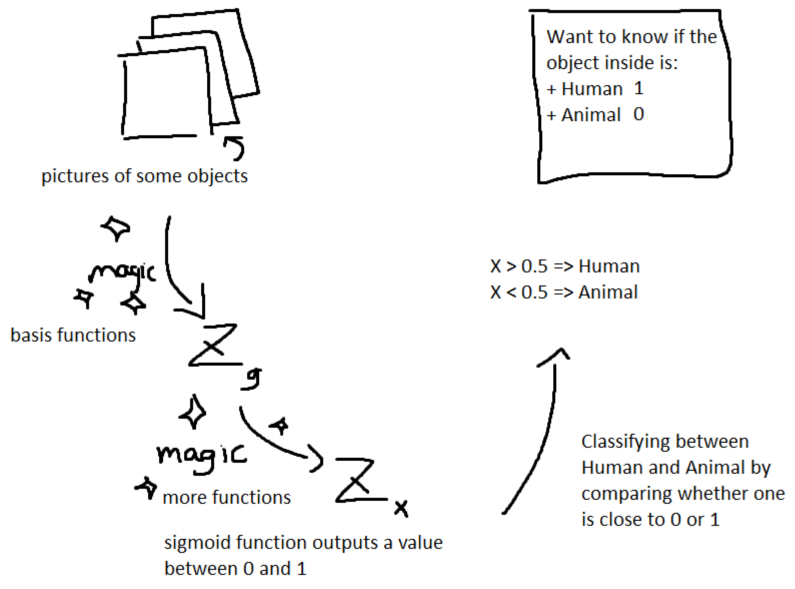

Want to know if the object inside is:
+ Human 1
+ Animal 0
pictures of some objects
magic
basis functions
X > 0.5 => Human
X < 0.5 => Animal
magic
more functions
Classifying between Human and Animal by comparing whether one is close to 0 or 1
sigmoid function outputs a value between 0 and 1

Image 4.2: Diagram of how logistic regression works in AI unified framework

Example: A machine trying to "know" what is in the picture: Human or Animal. After some "magic", we have a set of data, and after going through another transforming function, and through sigmoid function, we have a number between 0 and 1. Using that result and comparing it with 0.5, the machine can classify the object in the picture to be either in the Human class or the Animal Class.

5 Softmax Regression

Softmax Regression (you know it) is another regression, in itself, and is also used in **classification** like Logistic Regression. Softmax Regression, in some ways, is the generalization including Logistic Regression. It uses the Softmax function to transform a vector Zx into a **probability vector** and uses that vector to classify. The following examples explain it.

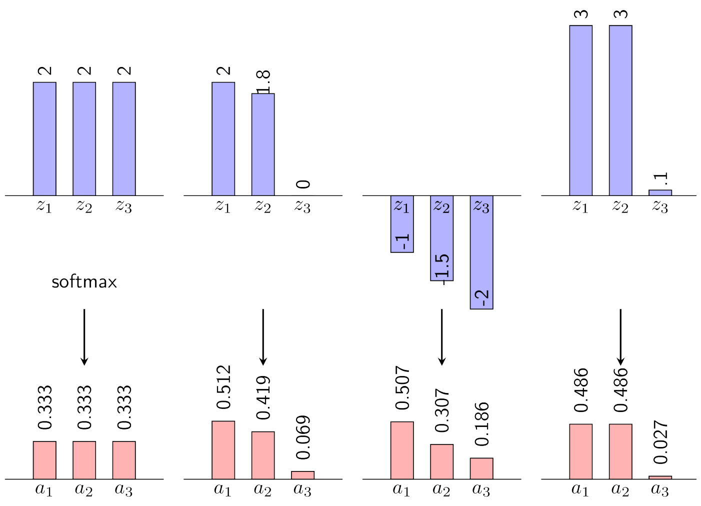


Image 5.1: Examples of how softmax function works

Supposed the Human-Animal classification machine has an extra class, "Non-living things". Doing the same thing like the machine has done with Logistic Regression, but with 3 classes, instead the comparing result would be in a form of probability vectors longer than 2.

For example, Zx = [0.3, 0.1, 0.6]. Comparing it with the one-hot data from the 3 class, the closest number to 1 would be 0.6, which classifies the above Zx to a "Non-living things" Class. The trick here is having to convert the coordinate vectors here into probability vectors in order to compare it with the class, using Softmax function.

6 Multi-layer Perceptron

Softmax Regression is linear classification. That means its data set has to be *linearly separable* (each group has to be linearly separated) Why is that a problem? Take a look at this graph below.

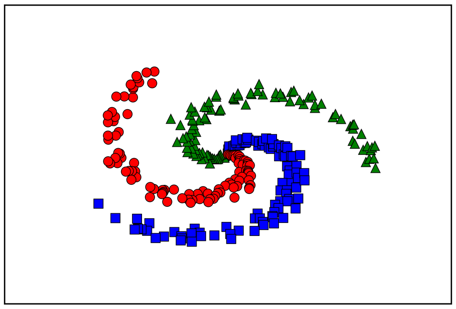


Image 5.1: Graph that shows the problem with linear classfication

You cannot draw a straight line anyhow to properly separate these groups. And in order to draw an acceptable line to separate these, you have to find a way to **non-linearize** the boundary lines. How can you do that? That is where **Multi-layer Perceptron** comes into handy.

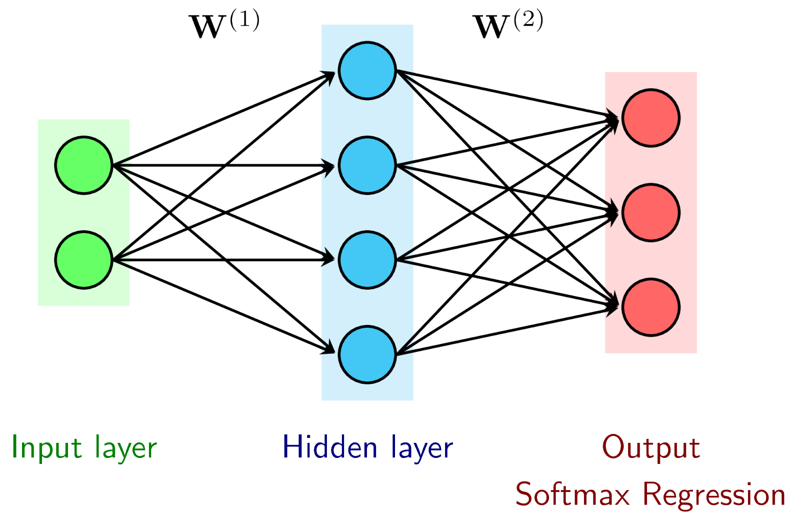


Image 5.2: Diagram explaining Multi-layer Perceptron

Basically, the Zx, after the magic transforming from the Zg, goes through another transformation by multiplying some more W, and each time it changes, it is considered to have gone through a **layer**. The number of layers are decided depending on the context and requirements of the program, but the results has a comparatively greater accuracy.