Lecture 6: Feature learning, Logistic Regression

Jan/26

Lecturer: *Anshumali Shrivastava* Scribe By: *Ye Cao, Sharath Giri, Anitesh Reddy*

Overview:

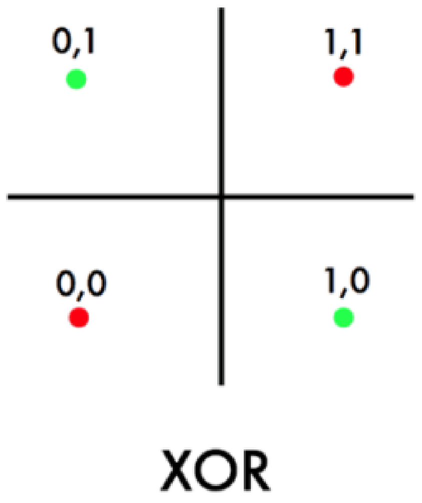
- Linear binary classification
  - Linear separability
  - XOR problem
- Example of linear inseparability
  - Feature transformation
  - This is why feature engineering is needed
- What is feature Engineering
  - Example of how feature engineering can solve the problem
  - Can we always find the fee for all problems? Yes if d > number of data points --- using feature engineering
  - Now how do we find the fee?
  - Theory of kernel transformation
  - Kernel matrix
  - Kernels & spaces are used on SV, instead of normal SVM, kernel SVMs are used – this way we don't need to know the fee , especially if it is very large
- Logistic regression
- Example
  - How to classify into more than one class – one vs all
    - Give example of One vs All
- How do we choose between W1, W2, W3 in linear binary classifiers – Loss is calculated independently and then choose the max distance
  - This way Loss functions are not efficient – this is were logistic equations come in
  - We use softmax as outer layer of logistic equation
- Cross entropy loss function for softmax – Probabilities are either 1 or 0
  - But probabilities need not be 1 or 0 – this is cross entropy – used in deep learning
- End of the day – finding fee for SVMs still remains a question
- Non-linear function combined with linear function
  - Deep Learning

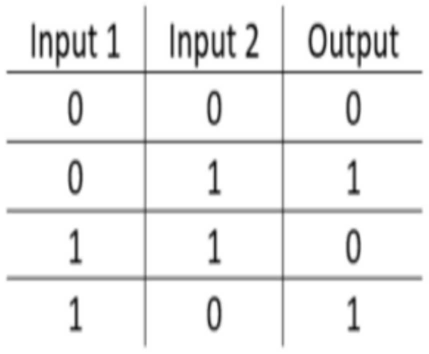
1. Classification problem and Linear separator

1.1) Linear Binary Classification: Linear Binary Classification is a machine learning method that finds a straight line boundary to divide data points into two distinct categories.

1.2) Linear separability: It refers to a linear boundary's ability to correctly separate two categories. If this is possible, simple classification models such as linear regression or support vector machines can be used.

1.3) XOR Problem: The XOR problem, on the other hand, arises when a linear boundary is unable to accurately separate the categories. Linear classifiers can be used only for linear separable use cases and XOR is one of the logical operations which are not linearly separable as the data points will overlap the data points of the linear line or different classes occur on a single side of the linear line.





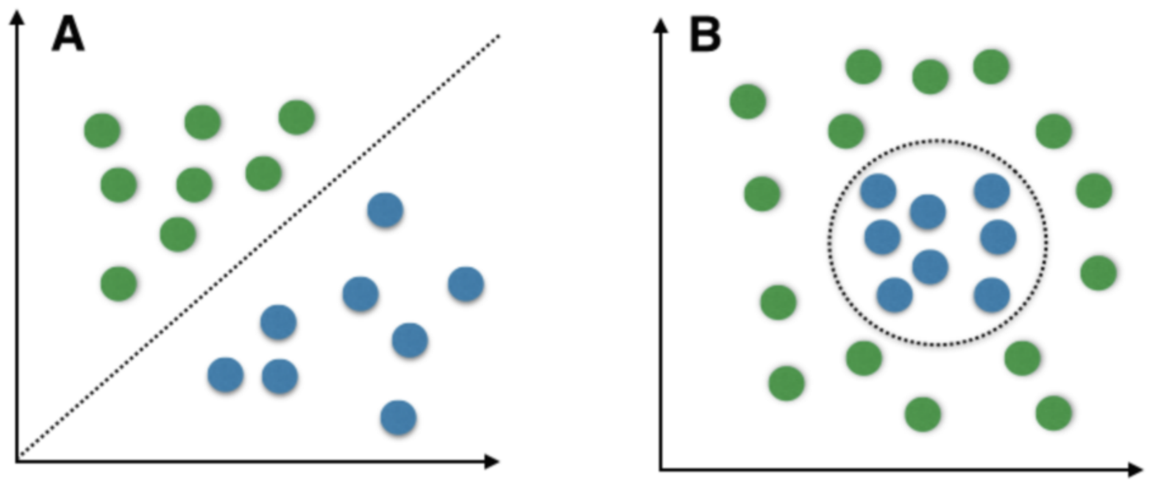
Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

(Above fig depicting the XOR mapped on the (x, y) co-ordinate plot)

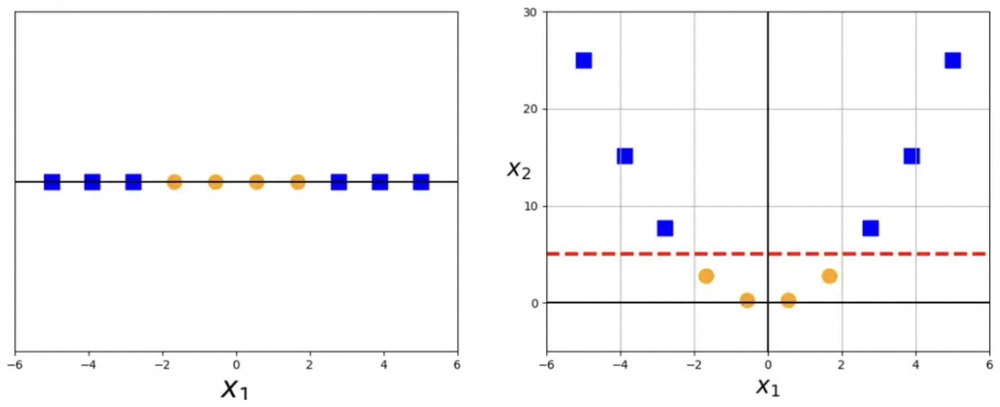
(Above fig depicts the inputs and outputs of XOR logical operation)

2. Feature transformation

2.1) Feature Engineering: In the real world, in general we come across situations which require models to predict subjects into more than two classes. Possible solution is to use a linear separator. In the following picture, we see that the first image adopts a linear separator to separate the feature vectors. However, the second picture contains feature vectors that cannot be separated using a linear separator. This leads us to adopt something like feature transformation. Feature learning is automated by the model through projection that amplifies certain features and diminishes others.



2.2) Feature transformation: A simple example of feature transformation is to apply a transformation function to existing feature vectors. For example, given a set of feature vectors {(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), …} and their class label {+,+,+, -,-,-, +,+,+}. There does not exist a good linear separator. However, if we apply transformation such that for each feature vector (x_1, x_2), $x_2 = x_1^2$. Then we have a set of transformed feature vectors {(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), …}. The transformed feature vectors turn out to have a good linear separator.



This data becomes linearly separable after a quadratic transformation to 2-dimensions.

2.3) Theory of kernel transformation: The example above describes an example that uses the function of feature transformation. The question is if there is some universal feature transformation function that could make the feature vectors in any problem linear separable. This is quite hard to find because specific problems are very different from each other and potentially require different feature transformations. However, it turns out that there is a theory of kernel transformation that states that the data set X is represented by an n x n kernel matrix of pairwise similarity comparisons where the entries (i, j) are defined by the kernel function: k(xi, xj). This kernel function has a special mathematical property. The kernel function acts as a modified dot product. Our kernel function accepts inputs in the original lower dimensional space and returns the dot product of the transformed vectors in the higher dimensional space. There are also theorems which guarantee the existence of such kernel functions under certain conditions.

Because, our data is only linearly separable as the vectors $\phi(x)$ in the higher dimensional space, and we are finding the optimal separating hyperplane in this higher dimensional space without having to calculate or in reality even know anything about $\phi(x)$.

2. Logistic Regression

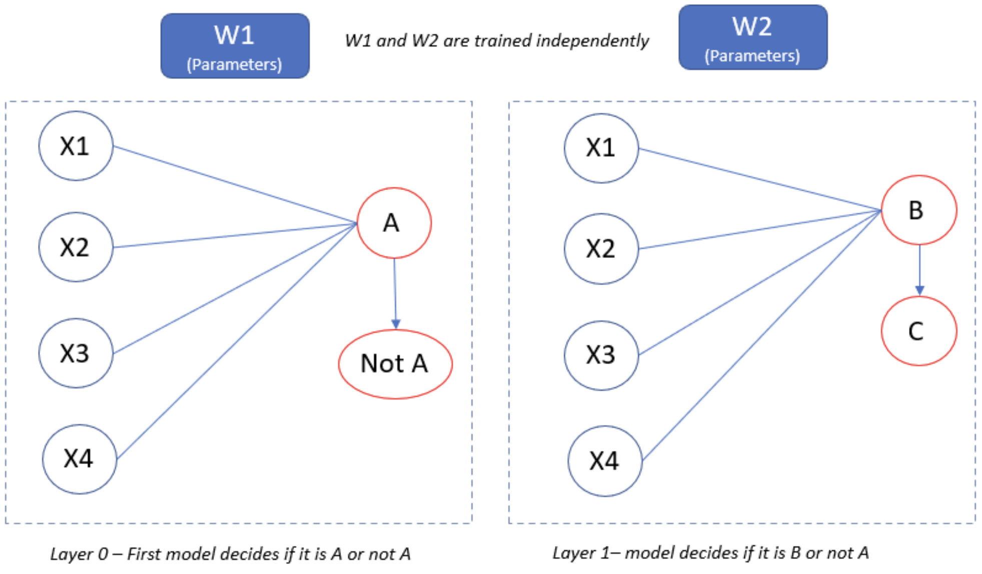
3) Multiple Category Classification: In the real world, in general, we come across situations which require models to predict subjects into more than two classes. How do we go about building such a model? Can we build on Binary classification models? Is there a better approach?

3.1) Usage of Logistic Binary Classification to predict multiple classes: Even though Binary classification models bifurcate input subject, we can still tweak it to work for Multiple class predictions:

"**One Vs All**" strategy: Building multiple layers of Binary models allow us to achieve it.

For example: If we have to classify into three classes {A, B, C}. First build a model which predicts either A or (Not A) and add another layer of binary model which predicts either B or C.

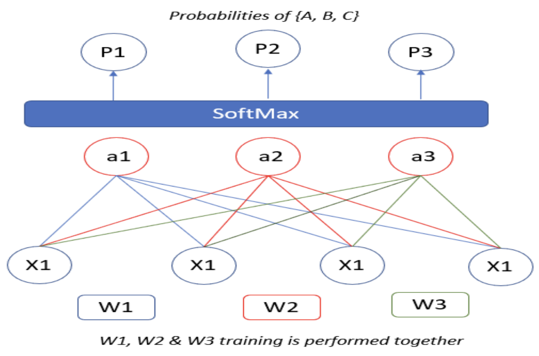
Illustration:


W1
(Parameters)
W1 and W2 are trained independently
W2
(Parameters)
X1
X2
X3
X4
A
Not A
X1
X2
X3
X4
B
C
Layer 0 – First model decides if it is A or not A
Layer 1– model decides if it is B or not A

Though we can build a "One vs All" model, working with Loss functions becomes very inefficient. Loss functions for every layer are worked independently while working on Maximal Margin Separator. This pain point is solved by the Multinomial Logistic equation with a Softmax outer layer.

3.2) Multinomial Logistic equation with Cross Entropy: This model enables prediction for more than two classes while it trains all the parameters together, unlike "One vs All " model. Generally, a Softmax function is applied as the outermost layer to ensure all the predicted probabilities are between [0,1] and summation of all the probabilities equates to 1.

Illustration:


Probabilities of {A, B, C}
P1
P2
P3
SoftMax
a1
a2
a3
X1
X1
X1
X1
W1
W2
W3
W1, W2 & W3 training is performed together

3.3) Softmax function:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$\vec{Z}$ - is the input vector to the softmax function, made up of (z0, ... zK)
Z_i - are the elements of the input vector to the softmax function
K - The number of classes to be classified

Once all the probabilities are evaluated by the Softmax function. The class with the highest probability is the prediction done by the model. Categorical Cross-entropy is commonly used to derive a Loss Function for a Multinomial Logistic Equation. In this Loss Function, the category labels are one-hot encoded. Meaning, for the predicted vector comprises only one and zeroes (One is assigned to the predicted class and zeroes to rest of them)

There are other Loss Functions that can be used, such as:

- MSE (Mean Square Error)
- Hinge Loss
- Kullback-Leibler (KL)
- General Cross-Entropy

In General Cross – Entropy, the labels are not one-hot encoded which are used in deep learning.

(Φ) for Support Vector Machines: The mapping function (Φ) transforms the input data into a higher-dimensional space where the classes can be differentiated by a linear boundary.
The value of (Φ) can have a significant impact on the performance of the SVM classifier. Selecting an appropriate (Φ) that captures the underlying structure of the information is essential for good SVM results. End of the day, finding (Φ) for SVMs still remains a challenge.

After 2010, when there was a data boom, it has become infeasible to find the (Φ) value manually and that gave a prominent edge to deep learning.

3.4) Why Deep Learning: Deep learning models, such as neural networks, are extremely adaptable and can learn complex non-linear relationships between inputs and outputs. As a result, unlike SVMs, they do not require an explicit mapping function. The XOR problem with neural networks can be solved by using Multi-Layer Perceptrons or a neural network architecture with an input layer, hidden layer, and output layer. So during the forward propagation through the neural networks, the weights get updated to the corresponding layers and the XOR logic gets executed. The Neural network architecture to solve the XOR problem will be as shown below.