Career Recommendation For LinkedIn Users From Semantic Set Of Skills

Harsh Patel (1401019) – School Of Engineering And Applied Science

***Abstract*—Until recently career selection has been a tricky, tedious and time consuming process, because people looking for a career had to collect information from many different sources. Career recommendation systems have been proposed in order to automate and simplify this task, also increasing its effectiveness. This paper aims on finding out relationships between career and people skills. A SVM algorithm is implemented to obtain a hierarchical clustering of career recommendation system where a user can enter a career goal and based on this career goal the platform suggest a career path. Also, reading users' profile the system suggests a career path and skills to be acquired.**

***Index Terms*—Support Vector Machine, Multiclass (one versus all) SVM, Gaussian Kernel**

I. Introduction

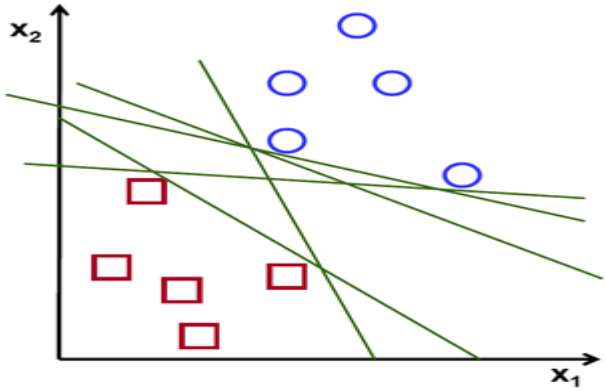
Problem of suggesting careers to users on the basis skill set or recommending certain skills to the user on the basis of his/her career goal, is discussed in this paper. [1] has already suggested many approaches like DNN, Fused Laso, Collaborative filtering for solving this approach. In this paper I try to solve the problem using Support Vector Machine (SVM). As suggested by [2] SVM is one of the best approach when it comes to classification. Here we try to classify users on the basis of the skill sets and a specific career path provided by the users. Thus in any one particular cluster we are looking at the users having a common skill set. This novel approach , makes it then easy to suggest different skills to the user or suggest career path to the users. Different Kernels and hypothesis used in the paper for achieving this goal are discussed in the latter part.

.

II. Support Vector Machine (SVM)

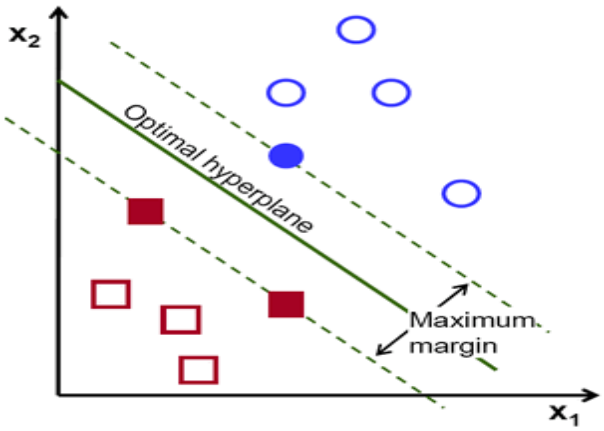
Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line.



In the above picture we can see that there exists multiple lines that offer a solution to the problem. We can intuitively define a criterion to estimate the worth of the lines. A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.



III. MULTICLASS SVM

Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements. The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems.

Common methods for such reduction include:

Building binary classifiers which distinguish between one of the labels and the rest (one versus all) or between every pair of classes (one versus one). Classification of new instances for the one versus all case is done by a winner takes all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.

IV. DATA PREPROCESSING

The data is available in JSON format, which we need to convert into the format which is easy to manage. Several anomalies could occur when it comes to actual processing on the data into our model i.e. unicode symbols, duplicate entries in other job interests, extra white spaces, no. of years in every skill etc. So many tasks were ran as a form of regex queries. Thus, cleaning and preprocessing on the data was done as a necessity in order to fit the model described in this paper. Before applying any algorithm we need to apply text semantic to our data. Because there are different number of designation, skills, company name and method of writing them, which has same meaning. With the help of Latent Semantic Indexing we can put the words having same meaning together and make our feature dimension manageable. Latent Semantic Analysis is useful in this problem because it helps us to search by meaning rather than the content and it is also useful in natural language processing.

V. ALGORITHM

The nonlinear regression problem is solved by using the SVM to establish a classification of the items considered and then to perform the regression based on the obtained classification results. With the proposed Gaussian Kernel Function it is possible to optimize the SVM parameters. The detailed steps of SVM classification based regression, that is personalized recommendation for regression method based on SVM classification optimized by Gaussian Kernel Function, can be presented as follows:

Step 1. Divide the sample data set S into N_q(q = 1,2,..s) classes based on the skill set of career goals.

Step 2. Use a training sample data set of *S* to generate a SVM classifier.

Step 2.1. Normalize the sample data.

Step 2.2. Select a kernel function and make use of Gaussian Kernel to optimize the parameters.

Step 2.3. Train the normalized sample data and then obtain the SVM classification model.

Step 3. Adopt this classifier to forecast class labels of the testing data. Classify the testing data and get the class label *j* of each sample (x_p,y_p), where, *p* is the number of the sample.

Step 4. For $N_q \in type\ j \wedge (x_p,y_p) \in type\ j$, M_q is a training data set, utilize SVM regression algorithm to predict y_p value of each testing samples.

Step 4.1. Normalize N_q and (x_p,y_p), which belong to the same class *j*.

Step 4.2. Select a kernel function to optimize the parameters.

Step 4.3. Train the normalized training data set and establish the SVM regression model.

Step 4.4. Adopt the established SVM regression model to forecast y_p value of each testing samples.

SVM hypothesis:

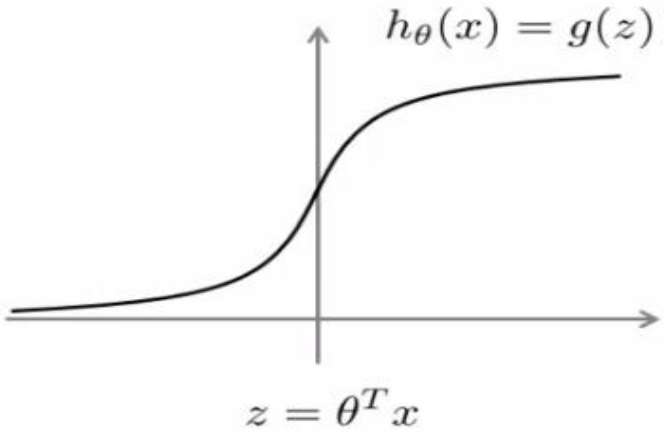
$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

Where, $cost_1(\theta^T x^{(i)}) = (-\log h_\theta(x^{(i)})$

And $cost_0(\theta^T x^{(i)}) = (-\log(1 - h_\theta(x^{(i)}))$

And $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ (Sigmoid Function)

θ = parameter, y = training pair output, x(i) = input skill vector, h(x) = test pair ouput



If y = 1, we want $\theta^T x \geq 1$

If y = 0, we want $\theta^T x \leq -1$

Gaussian Kernel :

k = similarity(x,$l^{(1)}$) = $e^{-(\left\|x - l^{(1)}\right\|^2/2\sigma^2)}$

if x ≈ $l^{(1)}$: k = 1

and if x is far from $l^{(1)}$: k = 0

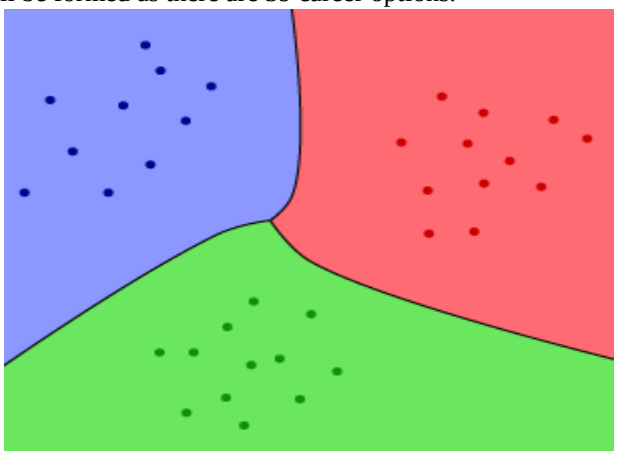
If the users' set of skills are similar to any one of the class (data point is nearer to the training point of that class) then kernel will predict k = 1and match the user to a set of class and will predict k = 0 for the other set of class that are far away from the data point (i.e. the skill set)

VI. Approach For Designing Modules

STEP 1. All the skills present in the data from every career is extracted and a vector is formed. A column vector of dimension (n x 1) is built where n is the number of skills. i.e.If there are total 200 skills in the skill set than a vector of (200 x 1) is built.

STEP 2. The skills required for a particular career are assigned bit-1 to the skill vector and bit-0 for other skills. i.e.If Software Engineer requires first 20 set of skills from the skill vector the first 20 rows are assigned bit-1 and rest of the skills bit-0.

STEP 3. Kernel classifys the vector points in such a way such that LMH(Large Margin Hyperplane) is satisfied and creates boundaries to distinguish careers taking skill vectors into consideration. i.e. each distinguished career will contain skill vector acquired in step 2 with bit-1 assigned to skill requirements of that particular career. In our case 39 classes will be formed as there are 39 career options.



Classes are distinguished in manner as shown in the figure. Here there are only 3 classes shown but in our case 39 classes will be formed for 39 career options and the skill vectors for every candidate will lie in that particular class . Let us understand our approach for designing modules taking this into consideration.

Top left class(Blue) = 1st class
Top right class(Red) = 2nd class
Bottom class(Green) = 3rd class
E.g 1st class requires skill s,w,t
2nd class requires skill a,b,m
3rd class requires skill p,q,r

MODULE I

STEP 4. A user inputs his set of skills and that skills are converted to a skill vector assigning bit-1 to the skills acquired for that user till now. Suppose this user has achieved skills ‘c, s, r’. A vector will be formed where skills c, s and r are assigned bit-1 and the rest is bit-0. Now, here the one vs all algorithm will identify one class where any skill from the users’ skill vector collides with the skill vector of any class. Than the loop goes on for (n-1) times for each class and the class where any skill collides will be suggested as a career option with additional set of skill requirement.

Here the output for the user will be:

Skill w,t required for class 1
Skill p,q required for class 3
2nd class will not be assigned as no skills are acquired till now.

Space Complexity = mn (where, m = no. of classes, n = no. of samples)

Time Complexity = cmn (where c = time complexity for one one sample)

MODULE II

STEP 5. Here the user inputs his career goal along with skill set acquired. Approaching the class of that career goal the skill sets are compared and the 0 bits of the users’ skill set colliding with the 1’s of the class’ skill set and these requirements are suggested for that career goal.

Space Complexity = mn (where, m = no. of classes, n = no. of samples)

Time Complexity = cn (where c = time complexity for one one sample)

VII. Results

Module I

```
For
Software Developer
    'J2EE'

    'Assurance'

    'EducationalMarketing'

For
Automation Engineer
    'ECS'

    'QualityEngineeringAutomation'

    'WebDevelopment'

    'J2EE'

    'BusinessAnalysis'

    'Assurance'

    'EducationalMarketing'

    'SupplyChainManagement'

    'SSIS'
```

Module II

```
Suggested skills for user
    'Manual'

    'Java'
```