Weekly Report 7

Utkarsh Patel
18EC30048

Topics Covered

1 Risks, Losses and Gains

Risk of generating a true negative result and a false positive result may not be same, as it is assumed until now. For this, every action α_i which classifies an instance $\boldsymbol{x}$ to class C_i, we define the risk as

$$R(\alpha_i|\boldsymbol{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k|\boldsymbol{x})$$

Here, we assumed we have K classes, λ_{ik} being the loss incurred while classifying $\boldsymbol{x}$ to class C_i, while it actually belonged to class C_k, and $P(C_k|\boldsymbol{x})$ represents the likelihood that $\boldsymbol{x}$ belongs to class C_k. Now, the learner should perform the action of which this defined risk is minimum. Hence optimal action is given as α_p, where $p = \min_k R(\alpha_k|\boldsymbol{x})$.

If we define $0/1$ loss, the above expression reduces to

$$R(\alpha_i|\boldsymbol{x}) = 1 - P(C_i|\boldsymbol{x})$$

Sometimes, the posterior probabilities for the classes are not very high, hence there remain a doubt for misclassification. In such a scenario, we define additional action α_{K+1} which represents not classifying instance $\boldsymbol{x}$ into any of the classes, in addition to actions α_i for classifying it to class C_i. We define loss for not classifying any instance as $\lambda_{(K+1)k} = \lambda$. Risk will be $R(\alpha_{K+1}|\boldsymbol{x}) = \lambda$.

Hence the algorithm for classifying an instance $\boldsymbol{x}$ using above framework is as follows

CLASSIFY(x)

```
1. Compute risks for each action $\alpha_i$, $i = 1, 2, \dots, K + 1$
2. Classifying as $C_i$ if $R(\alpha_i|\boldsymbol{x}) \leq R(\alpha_k|\boldsymbol{x})$, $k \neq i$, and $R(\alpha_i|\boldsymbol{x}) \leq R(\alpha_{K+1}|\boldsymbol{x})$
3. Otherwise deny any classification
```

Sometimes, instead of loss, we consider gain defined as U_{ik} as the gain observed for classifying an instance belonging to class C_k as class C_i and expected utility is defined for an action as

$$EU(\alpha_i|\boldsymbol{x}) = \sum_{k=1}^{K} U_{ik} P(C_k|\boldsymbol{x})$$

The action that has maximum expected utility is undertaken.

2 Association Rules

Let X and Y be two items available on a online shopping platform. The task is finding association between the two items, i.e., given some number of customers who bought item X, what fraction of them also bought item Y. This task is of supreme importance in the digital world today. For this, we define three parameters:

- *Support* as probability of customers who bought both the items X, Y out of all the customers of the store.
- *Confidence* as probability of customers who bought item Y given that they already bought item X.
- *Lift* as probability of customers who bought item Y given that they already bought item X out of customers who bought the item Y.

Therefore, we have for association $A \equiv X \rightarrow Y$,

$$S(A) = P(X,Y); \;\; C(A) = P(Y|X); \;\; L(A) = \frac{P(Y|X)}{P(Y)}$$

Now, the strength of this association is good if $S(A)$ is quite high, $C(A)$ is close to 1, and $L(A)$ should be greater than 1. There is an algorithm called Apriori algorithm to get association rules of high support and confidence from a database.

APRORI-ALGORITHM

1. Find frequent item sets (thus having a high support). No exhaustive search is performed. Instead, searches item set in order of increasing cardinality and removes superset whose member is not in already discovered item sets.
2. From the collected item sets, make association rules. Initially for an $k-$ item set, all but 1 are in antecedent, and 1 item is in consequent. Remove those having low confidence and with each pass increase elements in consequent and decrease element in antecedents.

3 Concept of Dimensionality Reduction

Sometime in a machine learning tasks, the dimension d of instance space X is very high, making the algorithms very slow. In such a scenario, it becomes important to reduce the dimension of the problem. There are two approaches to do so:

- *Feature selection* Finding k of the d dimensions which give us the most information, the other $d - k$ dimensions are discarded.
- *Feature extraction* Finding new set of k dimensions which are combination of original d dimensions. *LDA* and *PCA* are most widely used algorithms in this category.
- Add x' to X'.

4 Subset Selection

In this algorithm, best subset containing least number of dimensions which are contributing maximum to accuracy is computed. Not all possible subsets are exhaustively searched, instead a heuristic based approach is employed. There are two ways to determine this reasonable (may not be optimal) subset:

- *Forward Selection*

FORWARD-SELECTION(X)

```
1. Let $F$ represent best subset of attribute, $X_F$ represents the instance space $X$ reduced to vectors of dimensions contained in $F$ only. Let $C$ represents candidate dimensions.
2. Initially $F = \emptyset, C = \{1, 2, 3, \dots, d\}$
3. Define error $E = \infty$
4. Loop forever
  4.1 Define $c = \emptyset$
  4.2 For each candidate $i$ in C do
    4.2.1 Let $F' = F \cup \{i\}$
    4.2.2 Compute $E' = error(X_{F'})$
    4.2.3 if $E > E'$ do
      4.2.3.1 $c = i$
      4.2.3.2 $E = E'$
  4.3 if $c == \emptyset$ then break
  4.4 $F = F \cup \{c\}$
  4.5 $C = C - \{c\}$
```

- *Backward Selection* The algorithm is same as forward selection, but we start with $F = \{1, 2, \dots, d\}$ and C is not required and we delete features from F in each pass till error reduces.

It may be noted that in both forward and backward selection, the testing is done a validation set.

5 Principal Component Analysis

A mapping is defined which maps each instance $x \in X$ from a $d-$ dimensional space to a $k-$ dimensional space ($k < d$). For the mapping, a set W of k vectors in $d-$ dimensional space is defined.

$$W = \{w_1, w_2, \dots, w_k\}, w_i \in \mathbb{R}^d$$

PRINCIPAL-COMPONENT-ANALYSIS(X, W)

```
1. Define $X' = \emptyset$, where $X' \subset \mathbb{R}^k$
2. For each $x \in X$, we define $x' \in \mathbb{R}^k$, such that $x'_i = w_i^T x$, $i = 1, 2, \dots, k$. Add $x'$ to $X'$.
3. Return $X'$
```

Defining the set W requires doing statistical analysis. w_1 is called the first principal component. It must be such that $Var(z_1)$ is maximum possible, where $z_1 = w_1^T x, x \in X$. This problem can be solved by using Lagrange equations. Let Σ denotes the covariance matrix of instances x. Then, $Var(z_1) = w_1^T \Sigma w_1$. Also, the quantity $\|w_1\|$ must be equal to 1 for getting unique solution. We construct $\mathcal{L}(w_1) = \min_{w_1}(w_1^T \Sigma w_1 - \alpha(w_1^T w_1 - 1))$ to get this first principal component. Same algorithm is repeated for finding other principal components.

Novel Ideas

In my project involving fMRI scans, I used PCA to find strongly connected brain modules. This simplifies the analysis by reducing the brain ROIs to just a few 100 modules.

Doubt Clearing Class

The discussion was quite engaging, especially the one regarding the worst-case tree depth in ID3 algorithm.