Effectiveness of Social Media Ads.

RELEVANCE VECTOR MACHINE - Classification | Assignment - 3 | Pattern Recognition

RAMIT NANDI || MD2211

2023-10-02

Contents

INTRODUCTION

Relevance Vector Machine - is a kernel-trick machine learning algorithm , based on Bayesian Inference. It can deal with both 'Regression' & 'Classification' setup by specifying suitable Likelihood function. Here we will explain it for Classification case only.

Basic WorkFlow

Suppose we have n i.i.d. observations $x_{1.}, x_{2.}, ..., x_{n.}$ of some p covariates ; $t_1, t_2, ..., t_n$ corresponding observed responses ($1-of-k$ encoding for a categorical response with k responses) , $\phi : \mathbb{R}^p \to \mathbb{R}^m$ any feature map.

$$\mathbb{X}^{n\times p} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}, \boldsymbol{t}^{n\times k} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix}, \Phi^{n\times m} = \begin{bmatrix} \phi^T(x_1) \\ \phi^T(x_2) \\ \vdots \\ \phi^T(x_n) \end{bmatrix}$$

- *Choose Likelihood:* Let $\boldsymbol{W}^{m\times k} = [w_1, w_2, ..., w_k]$ be our parameter of interest, such that $p_{ij} = \frac{exp(w_j^T \phi(x_i))}{\sum_{r=1}^k exp(w_r^T \phi(x_i))}$
  Assume $t_i|x_i, \boldsymbol{W} \sim Multinomial(1; [p_{i1}, p_{i2}, ..., p_{ik}]) \ \ \forall i = 1, 2..., n$, then we have joint likelihood

$$f(\boldsymbol{t}|\mathbb{X}, \boldsymbol{W}) = \prod_{i=1}^n f(t_i|x_i, \boldsymbol{W}) \propto \prod_{i=1}^n \prod_{j=1}^k p_{ij}$$

- *Choose Prior:* We choose ARD prior for $\boldsymbol{W} = ((w_{ij}))$ i.e.

$$w_{ij} \overset{independent}{\sim} \mathcal{N}(0, \alpha_{ij}^-1) \ \forall i = 1, 2, ..., n; j = 1, 2, ..., k$$

- *Compute Posterior:* For bayesian inference, we need

$$f(\boldsymbol{W}|\mathbb{X}, \boldsymbol{t}) = \frac{f(\boldsymbol{t}|\mathbb{X}, \boldsymbol{W}) \times f(\boldsymbol{W})}{\int f(\boldsymbol{t}|\mathbb{X}, \boldsymbol{W}) f(\boldsymbol{W}) d\boldsymbol{W}}$$

  But the exact calculation is very difficult, instead Laplace Approximation is used

- *Hyperparameter Tuning:* In practice, α_{ij}s are not known to us, we need to tune them to the value that maximizes $f(\boldsymbol{t}|\mathbb{X}) = \int f(\boldsymbol{t}|\mathbb{X}, \boldsymbol{W}) f(\boldsymbol{W}) d\boldsymbol{W}$

- *Inference & Prediction:* Let based on optimal hyperparameters , our posterior density is $f_{optimal}(\boldsymbol{W}|\mathbb{X},\boldsymbol{t})$ Then we can have the point estimate

  $$\hat{\boldsymbol{W}}_{MAP} = argmax_{\boldsymbol{W}}[f_{optimal}(\boldsymbol{W}|\mathbb{X},\boldsymbol{t})]$$

  For a new observation x , we can compute posterior predictive distribution

  $$f(t_x|x,\mathbb{X},\boldsymbol{t},\boldsymbol{W}) = \int f(t_x|x,\boldsymbol{W})f_{optimal}(\boldsymbol{W}|\mathbb{X},\boldsymbol{t})d\boldsymbol{W}$$

  or, can predict the class as $argmax_j[exp(\hat{w}_j^T\phi(x))]$

- *Kernel Trick:* Take $w_j^{m\times 1} = \sum_{i=1}^{n}\lambda_{ji}\phi(x_i) = \Phi^T\boldsymbol{\lambda_j}\ \forall j = 1,2,...,k$ Then we have $p_{ij} = \frac{exp(\boldsymbol{\lambda_j}^T\Phi\phi(x_i))}{\sum_{r=1}^{k}exp(\boldsymbol{\lambda_r}^T\Phi\phi(x_i))}$, and we can reparametrize the entire model in term of $[\boldsymbol{\lambda_1},\boldsymbol{\lambda_2},...,\boldsymbol{\lambda_k}]^{n\times k}$. Now,

  $$\Phi^{n\times m}[\phi(x_i)]^{m\times 1} = \begin{bmatrix}\phi^T(x_1)\\ \phi^T(x_2)\\ \vdots\\ \phi^T(x_n)\end{bmatrix}\phi(x_i) = \begin{bmatrix}k(x_1,x_i)\\ k(x_2,x_i)\\ \vdots\\ k(x_n,x_i)\end{bmatrix}$$

  Notice that , this model access $\mathbb{X}$ only in terms of kernel $k(x_i,x_j) = \phi^T(x_i)\phi(x_j)$, explicit calculation of feature maps is not needed.

NOTE: Theoretically RVM can handle any number of classes , but computation is so much involved that till date both in **R** & **python** , direct implementation is available for Binary Classification only. Multi-class problems are solved like ‘one-vs-one’ or ‘one-vs-rest’ combination of binary classifications.

DATA

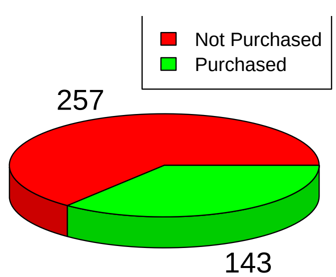
The dataset contains details of the purchase of a product based on social network advertisements. The data has 400 observations, looks as follows . . .

User.ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0

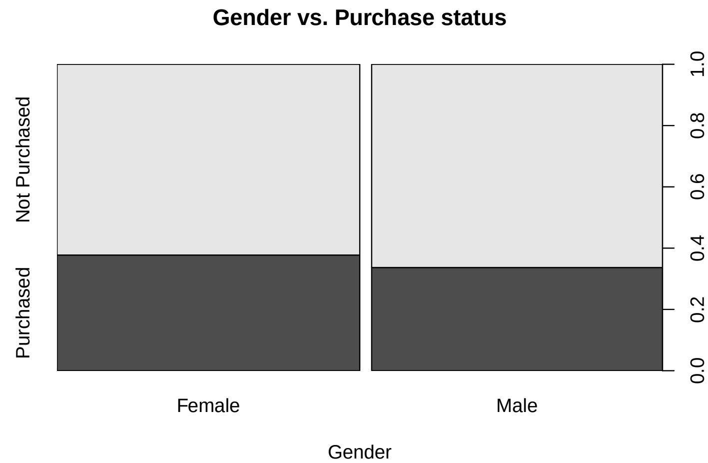
GOAL: Predicting whether a person will buy a product displayed on a social network advertisement based on his/her gender , age and approximate Salary.

EDA

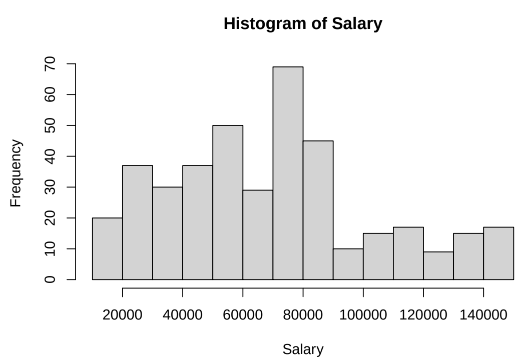
- Target class belongs to two discrete categories of purchased and not purchased. [Throughout the report , Red colour will denote 'Not Purchased' , Green colour will denote 'Purcased']

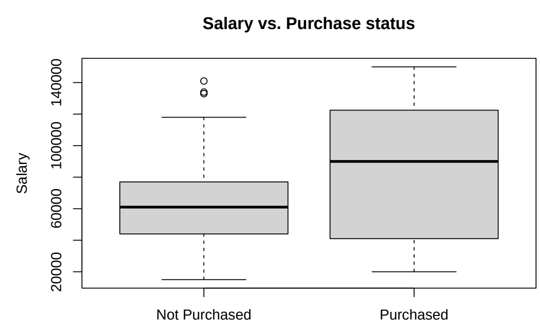


- *Gender:* Gender does not affect the purchase status much here. Given a person is male, the chance that he will buy the product is very similar to the chance of purchase ,given the person is female .

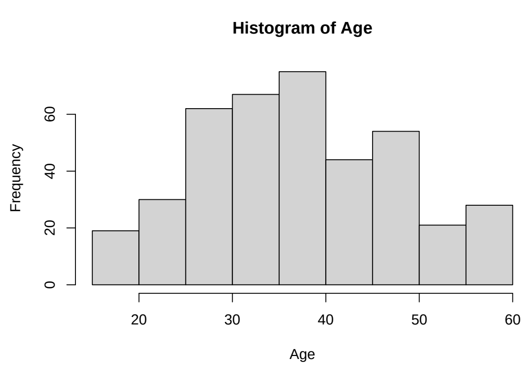


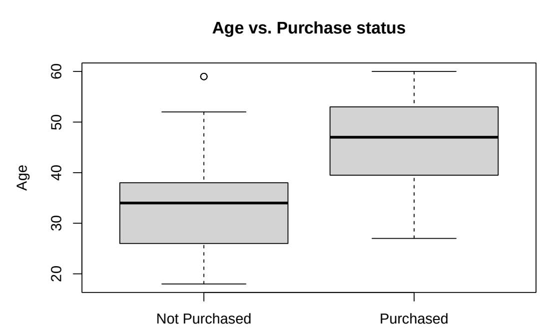
- *Salary:* Those who purchase the product , have on average higher salary than those who do not.





- *Age:* Those who purchase the product , are on average older than those who do not.





Applying RVM

We will use 80% data for training and 20% leftout data as Test set.

fitting

Lets, train a RVM

```
import numpy as np
import pandas as pd
from sklearn_rvm import EMRVC
from sklearn.model_selection import  train_test_split
from sklearn.preprocessing import StandardScaler

Data = r.Data.iloc[:,1:-1]
Data = pd.get_dummies(Data,drop_first=1)
TrainX,TestX,TrainY,TestY = train_test_split(Data,r.Y,train_size=0.8,random_state=101)
Scale = StandardScaler()
TrainX.iloc[:,:-1] = Scale.fit_transform(TrainX.iloc[:,:-1])
TestX.iloc[:,:-1] = Scale.transform(TestX.iloc[:,:-1])


Model_lin = EMRVC(kernel='linear',bias_used=True)
Model_lin.fit(TrainX,TrainY)
```

```
## EMRVC(init_alpha=9.70487475859124e-06, kernel='linear')
```

We can see , the 'number of relevance vectors' is small compared to total number of observations, as RVM finds out sparse model representation based on a few observations only. Those observations , which have non-null importance , are called 'relevant vectors' (and hence the name RVM)

```
## Number of relevance vectors: 6
```

```
## Dimension of training data:  320 3
```

Also, Let us train the same model with RBF kernel

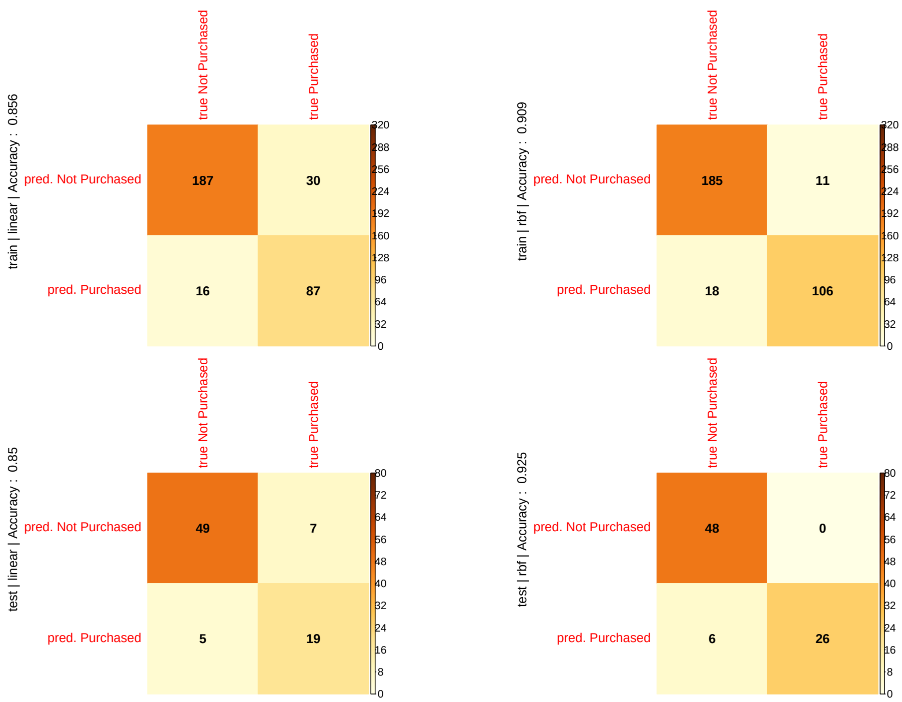
```
Model_rbf = EMRVC(bias_used=True,gamma='scale')
Model_rbf.fit(TrainX,TrainY)
```

```
## EMRVC(gamma='scale', init_alpha=9.70487475859124e-06)
```

```
## Number of relevance vectors: 15
```

performance

Compare the performance on train set & test set, using confusion matrix.



Since RBF kernel works with a infinite-dimensional feature-space implicitly, it is giving better fit & prediction than linear kernel.

	Linear.Kernel	RBF.Kernel
train_set_accuracy	0.856	0.909
test_set_accuracy	0.850	0.925

Comparison with SVM

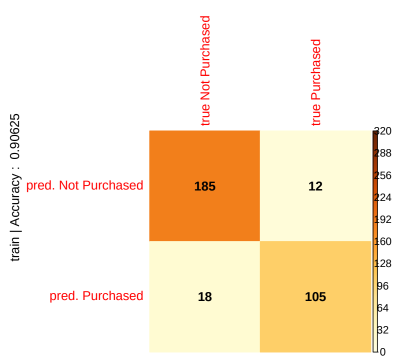
We can do the same classification using SVM also.

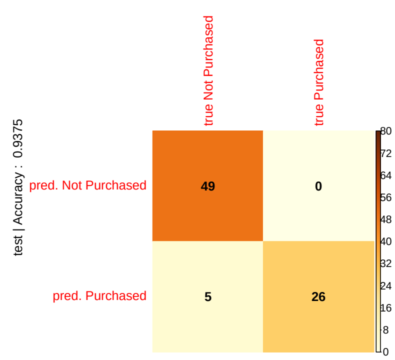
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV,RepeatedStratifiedKFold

Model_svm = SVC(kernel='rbf')
param = {'C':[0.01,0.05,0.1,0.4,0.8,1,2,5]}
CV = RepeatedStratifiedKFold(n_repeats=5,n_splits=3,random_state=101)
opt_hyp = (GridSearchCV(Model_svm,param_grid=param,
cv=CV,
refit=False)).fit(TrainX,TrainY)

Model_svm = SVC(C=opt_hyp.best_params_['C'],kernel='rbf').fit(TrainX,TrainY)
```

```
## SVC(C=1)
```

```
## Number of support vectors: 46 49
```

	accuracy
train_set	0.90625
test_set	0.93750





RVM over SVM

Advantages

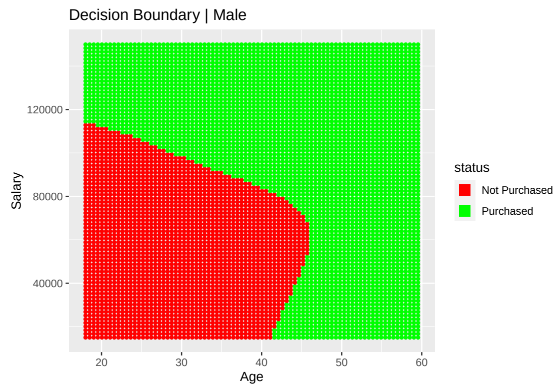
- RVM yields sparser model, less number of observations having non-null importance, so performs faster prediction
- It gives ‘class posterior probabilities’ , while SVM is non probabilistic
- No need for cross-validation of penalty cost hyperparameter

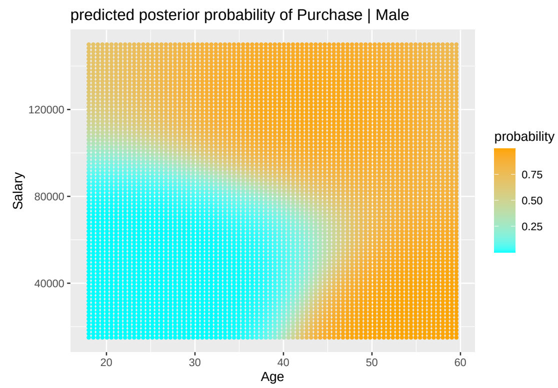
Disadvantage

- More computation intensive , so RVM needs more training time

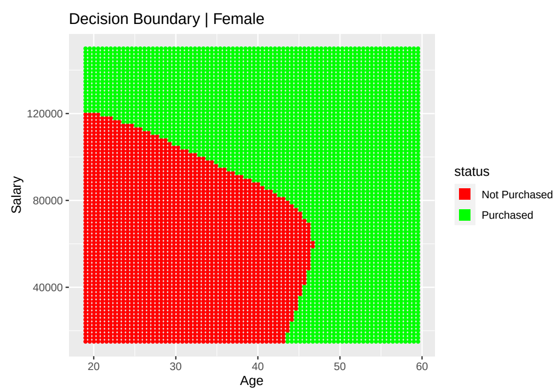
Plotting RVM Decision Boundaries

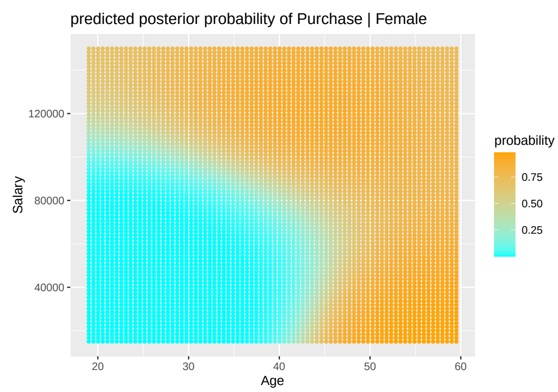
for Males





for Females





CONCLUSION

Looking at the nature of decision boundary we have, we can conclude
Those with high Salary are likely to purchase a product , irrespective of their Age. But younger persons are unlikely to purchase a product if the Salary is not enough.