

Shirin Mohebbi

Pattern Recognition

Assignment #3

January 8, 2021

Part A: Bayesian Classification (Quadratic Multiclass Classification)

For this classification we use Bayesian probability:

$$P(y|\mathbf{X}) = \frac{\overbrace{P(\mathbf{X}|y)}^{\text{Likelihood}} \overbrace{P(y)}^{\text{Prior}}}{\underbrace{P(\mathbf{X})}_{\text{normalizing factor}}}$$

$$y^{new} = \operatorname{argmax}_y P(y|\mathbf{X})$$

$p(\mathbf{X})$ is the same for all Y so, it doesn't have any effect.
we should find $p(\mathbf{x}|y)$ and $p(y)$

- Model $P(y)$ and $P(\mathbf{X}|y)$ for each class y :

$$P(y) = \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_c^{1\{y=c\}}$$

$$P(\mathbf{X}|\mathbf{y} = i) = \frac{1}{(\sqrt{2\pi})^n |\Sigma_i|^{\frac{1}{2}}} \exp\left(\frac{-1}{2}(\mathbf{X} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1}(\mathbf{X} - \boldsymbol{\mu}_i)\right)$$

for finding parameter i use below formula:

$$\phi_i^{MLE} = \frac{\sum_{j=1}^m 1\{y^{(j)} = i\}}{m}$$

$$\mu_i^{MLE} = \frac{\sum_{j=1}^m 1\{y^{(j)} = i\} \mathbf{X}^{(j)}}{\sum_{j=1}^m 1\{y^{(j)} = i\}}$$

$$\Sigma^{MLE} = \frac{1}{m} \sum_{j=1}^m (\mathbf{X}^{(j)} - \mu_{y^{(j)}})(\mathbf{X}^{(j)} - \mu_{y^{(j)}})^T$$

so first i generate data with given mean and covariance for each class separately and given each class label, i used 80% of data of each class for train.

```
x0, y0 = np.random.multivariate_normal(mean0, cov0, numData).T
x1, y1 = np.random.multivariate_normal(mean1, cov1, numData).T
x2, y2 = np.random.multivariate_normal(mean2, cov2, numData).T
```

then i find mean, covariance and phi of each class based on above formula.

```
self.meanClass0 = np.array( [(sum(x0[:numTrain])/numTrain),
                             (sum(y0[:numTrain])/numTrain)] )
```

```
self.phiClass0 = 1/3
```

```
self.sigma0 = np.round( (1/self.countData) * (demeanX0.T @ de-
meanX0) , 5)
```

we find probability of x being in each class like this, then we find arg max to predict x label.

```
coefficient0 = 1 / ( ((2 * np.pi) ** (n/2)) *
np.sqrt(abs(np.linalg.det(self.sigma0))) )
```

```
pXY0 = coefficient0 * (np.exp( -0.5 * ( (demean0.T @ sigmaInver-
s0) @ demean0 ) ))
pXY1 = coefficient1 * (np.exp( -0.5 * ( (demean1.T @ sigmaInver-
s1) @ demean1 ) ))
pXY2 = coefficient2 * (np.exp( -0.5 * ( (demean2.T @ sigmaInver-
s2) @ demean2 ) ))
```

```
l0 = pXY0 * self.phiClass0
l1 = pXY1 * self.phiClass1
l2 = pXY2 * self.phiClass2
```

because each we use different sigma for each class, the Decision boundary is not linear.

i obtain boundary from below formula.

$$\log \frac{P(y=i)}{P(y=j)} - \frac{1}{2} \log \frac{|\Sigma_i|}{|\Sigma_j|} - \frac{1}{2} [\mathbf{X}^T (\Sigma_i^{-1} - \Sigma_j^{-1}) \mathbf{X} + \mu_i^T \Sigma_i^{-1} \mu_i - \mu_j^T \Sigma_j^{-1} \mu_j - 2 \mathbf{X}^T (\Sigma_i^{-1} \mu_i - \Sigma_j^{-1} \mu_j)] = 0$$

$$\Rightarrow \mathbf{X}^T a \mathbf{X} + b^T \mathbf{X} + c = 0$$

we should obtain a, b and c from above formula.

implemented like this.

```
c = ( np.log(self.phiClass0/self.phiClass1) )
- ( 0.5 * np.log( np.linalg.det(self.sigma0)/ np.linalg.det(self.sigma1) ) )
+ ( self.meanClass0.T @ (np.linalg.inv(self.sigma0) @ self.meanClass0) )
- ( self.meanClass1.T @ (np.linalg.inv(self.sigma1) @ self.meanClass1) )
b = -2 * ( (np.linalg.inv(self.sigma0) @ self.meanClass0) - (np.linalg.inv(self.sigma1) @ self.meanClass1) )
a = -0.5 * (np.linalg.inv(self.sigma0) - np.linalg.inv(self.sigma1))
```

to plot we should calculate x1 according to x0. in order to do that we should solve above equation.

$$\begin{cases} x = \%r_3 \\ y = -\frac{\sqrt{\%r_3^2 (k^2 + 2fk - 4dg + f^2) + \%r_3 (2bk - 4gh + 2bf) - 4gj + b^2} + \%r_3 (k + f) + b}{2g} \end{cases}$$

no we find relation between x0 and x1. we can plot boundaries.

result:

dataSet1 results

accuracy train: 77.5 %

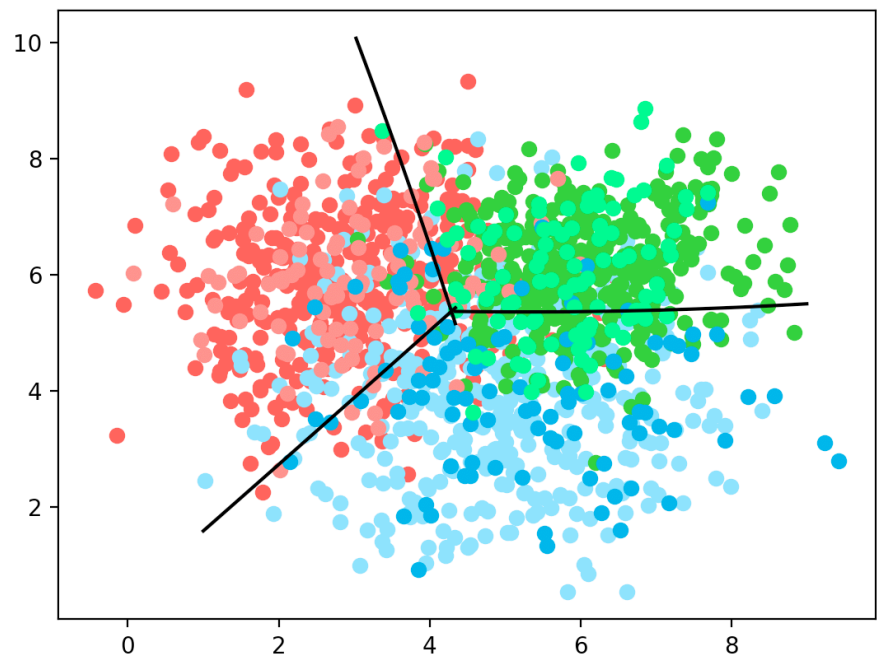
accuracy test: 77.0 %

dataSet2 results

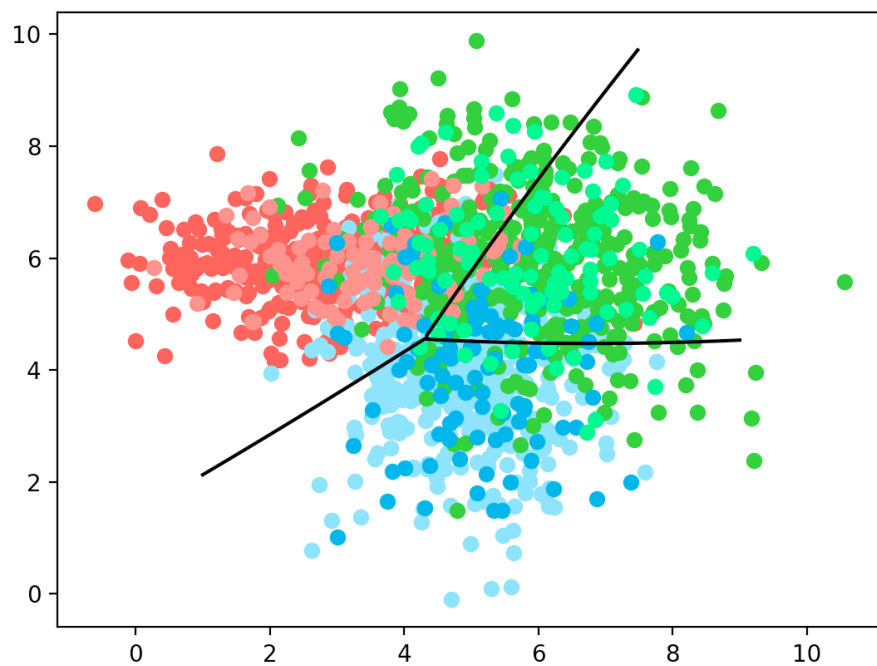
accuracy train: 79.75 %

accuracy test: 79.33 %

dataset 1 plot



dataset 2 plot



Part B: Naïve Bayes Classification

for applying naive bayes, we should first create bag of words. first i eliminate non word character from these sentences. each word is a feature. because datasets are big and have a lots of words, i eliminate stop words. then i only consider frequent word for better performance. after creating the bag of words.

then we use bayes theorem and maximum likelihood to predict probability of each sentence considered positive or negative.

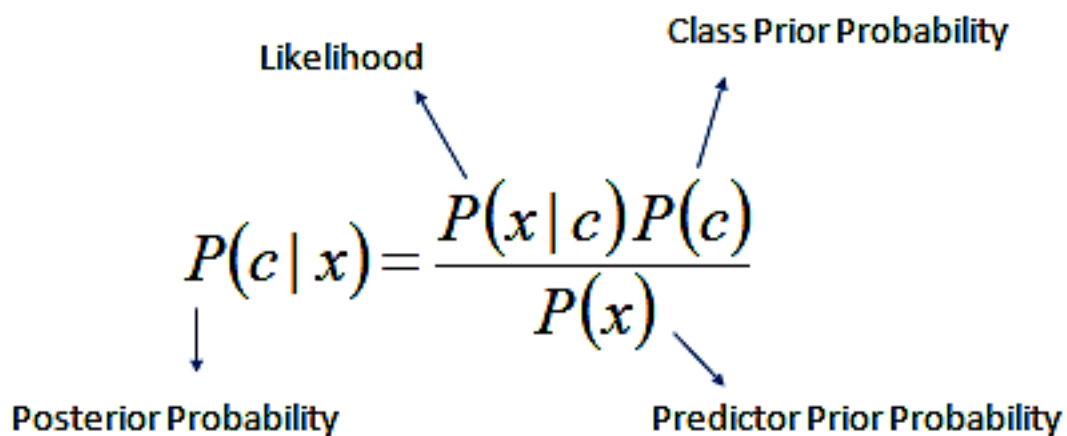
$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$


Diagram illustrating the components of the Naive Bayes formula:

- $P(c | x)$ is labeled as **Posterior Probability**.
- $P(x | c)$ is labeled as **Likelihood**.
- $P(c)$ is labeled as **Class Prior Probability**.
- $P(x)$ is labeled as **Predictor Prior Probability**.

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

we calculate $\log(P)$ instead of P because it doesn't affect $\arg \max$. and to avoid **problem of multiply many small probabilities**. when we log all multiplication converts to summation. we have one problem when we use \log . $\log(0)$ is undefined. to avoid this i use **Laplace Smoothing**.

$(\text{countX} + 1) / (\text{countY} + 2)$ instead of $(\text{countX}) / (\text{countY})$ to **avoid zero value**.

the base **assumption** of Naïve Bayes classifier is that **samples** assumed to be **independent**. so we calculate $P(X|c) = p(X_1|c) * p(X_2|c) * \dots * p(X_n|c)$. without this assumption, we couldn't calculate likelihood this easily.

results:

```
imdb
accuracyTrain 94.25
accuracyTest 82.0
yelp
accuracyTrain 89.625
accuracyTest 82.0
amazon
accuracyTrain 93.25
accuracyTest 79.5
```