Machine Learning homework 6

Bodong Zhang u0949206

1 Warming up: Probabilities

1 $P(A_1, A_2)=P(A_2)*P(A_1|A_2)= P(A_2)*P(A_1)$,

$P(A_2|A_1)=\frac{P(A_1,A_2)}{P(A_1)}=\frac{P(A_1)P(A_2)}{P(A_1)}=P(A_2)$ So the value of $A_1(A_2)$ would not influence the probability of the other one, so A_1 and A_2 are independent.

2 A_1, A_2, A_3 are mutually exclusive, also $P(A_1)+P(A_2)+P(A_3)=1$, so they are also collectively exhaustive. So $P(A_4)=P(A_1,A_4)+P(A_2,A_4)+P(A_3,A_4)=P(A_1)P(A_4|A_1)+ P(A_2)P(A_4|A_2)+ P(A_3)P(A_4|A_3)=\frac{1}{3}\frac{1}{6}+\frac{1}{3}\frac{2}{6}+\frac{1}{3}\frac{3}{6}=\frac{1}{3}$.

3 The possibility equals the number of choices the two of coin tosses are heads times the possibility of each event. So $P(\text{two heads})=\binom{n}{2}(\frac{1}{2})^2(\frac{1}{2})^{n-2}$.

4 $P(A_1|A_2)=\frac{P(A_1A_2)}{P(A_2)}=\frac{P(A_1)+P(A_2)-P(A_1\cup A_2)}{P(A_2)}=\frac{a_1+a_2-P(A_1\cup A_2)}{a_2}\geq\frac{a_1+a_2-1}{a_2}$.

5 (a) A1 and A2 are independent events, so $E[A1+A2]=\sum_{A_1}\sum_{A_2}(A_1+A_2)P(A_1A_2)=\sum_{A_1}\sum_{A_2}(A_1+A_2)P(A_1)P(A_2)=\sum_{A_1}P(A_1)\sum_{A_2}P(A_2)(A_1+A_2)=\sum_{A_1}P(A_1)(A_1+\sum_{A_2}P(A_2)A_2)=\sum_{A_1}P(A_1)(A_1+E(A_2))=E(A_2)+\sum_{A_1}P(A_1)(A_1)=E(A_1)+E(A_2)$

5(b) First to simplify proof, we assume E(A1)=E(A2)=0 since it does not affect variance if we move the mean, then $var(A1+A2)=\sum_{A_1}\sum_{A_2}(A_1^2+A_2^2)P(A_1A_2)=\sum_{A_1}\sum_{A_2}(A_1^2+A_2^2)P(A_1)P(A_2)=\sum_{A_1}P(A_1)\sum_{A_2}P(A_2)(A_1^2+A_2^2)=\sum_{A_1}P(A_1)(A_1^2+\sum_{A_2}P(A_2)A_2^2)=\sum_{A_1}P(A_1)(A_1^2+var(A_2))=var(A_2)+\sum_{A_1}P(A_1)(A_1^2)= var(A_1)+ var(A_2)$.

2 Naive Bayes

1(a) Since we have infinite data, so the estimated possibility would be the true possibility. $\hat{P}(x_1=-1|y=-1)$=0.8, $\hat{P}(x_1=1|y=-1)$=0.2, $\hat{P}(x_1=-1|y=1)$=0.1, $\hat{P}(x_1=1|y=1)$=0.9, $\hat{P}(y=-1)$=0.1, $\hat{P}(y=1)$=0.9

1(b)

$\hat{P}(x_1=-1,y=-1)=\hat{P}(y=-1)\,\hat{P}(x_1=-1|y=-1)$=0.1*0.8=0.08

$\hat{P}(x_1=1,y=-1)=\hat{P}(y=-1)\,\hat{P}(x_1=1|y=-1)$=0.1*0.2=0.02

$\hat{P}(x_1=-1,y=1)=\hat{P}(y=1)\,\hat{P}(x_1=-1|y=1)$=0.9*0.1=0.09

$\hat{P}(x_1=1,y=1)=\hat{P}(y=1)\,\hat{P}(x_1=1|y=1)$=0.9*0.9=0.81

Input x_1	$\hat{P}(x_1,y=-1)$	$\hat{P}(x_1,y=1)$	Prediction: y'=arg $\max_y \hat{P}(x_1,y)$
-1	0.08	0.09	y'=1
1	0.02	0.81	y'=1

1(c) P(y'≠y)=P(y'≠y,x_1=-1)+P(y'≠y,x_1=1)= P(y=-1,x_1=-1)+P(y=-1,x_1=1)=0.08+0.02=0.1

2(a) x_2 is actually identical to the first feature x_1, so x1=x2

So P(x_1=-1, x_2=1|y=1)=0 because $x_1 \neq x_2$, but P(x_1=-1|y=1)P(x_2=1|y=1)=0.1*0.9=0.09, so P(x_1=-1, x_2=1|y=1) ≠ P(x_1=-1|y=1)P(x_2=1|y=1). So they are not conditionally independent given y.

2(b)

$\hat{P}(x_1 = 1, x_2 = -1, y)$= $\hat{P}(x_1 = -1, x_2 = 1, y)$=0

$\hat{P}(x_1 = -1, x_2 = -1, y = -1)$= $\hat{P}(x_1 = -1, y = -1)$- $\hat{P}(x_1 = -1, x_2 = 1, y = -1)$= $\hat{P}(x_1 = -1, y = -1)$=0.08

$\hat{P}(x_1 = -1, x_2 = -1, y = 1)$= $\hat{P}(x_1 = -1, y = 1)$=0.09

x1	x2	$\hat{P}(x_1, x_2, y = -1)$	$\hat{P}(x_1, x_2, y = 1)$	Prediction: y'=arg max$_y$ $\hat{P}(x_1, x_2, y)$
-1	-1	0.08	0.09	y'=1
-1	1	0	0	Make no sense, y'=1 or -1
1	-1	0	0	Make no sense, y'=1 or -1
1	1	0.02	0.81	y'=1

2(c) P(y'≠y)=P(y'≠y,x_1=x_2=-1)+P(y'≠y,x_1=x_2=1)=0.08+0.02=0.1

2(d) Yes, we can use linear function to represent the prediction of y. In this case, y'=1.

3 Naive Bayes and Linear Classifiers

$\frac{\Pr(\boldsymbol{x}|y=1)\Pr(y=1)}{\Pr(\boldsymbol{x}|y=0)\Pr(y=0)} = \frac{\Pr(y=1)\prod_{j=0}^{d}\Pr(x_j|y=1)}{\Pr(y=0)\prod_{j=0}^{d}\Pr(x_j|y=0)} \geq 1$

Take log and we get logPr(y=1)-logPr(y=0)+$\sum_{j=0}^{d}\log\Pr(x_j|y=1) - \sum_{j=0}^{d}\log\Pr(x_j|y=0) \geq 0$.

$\Pr(x_j|y)$ is defined using a Gaussian/Normal probability density function, so

logPr(x_j|y)=log($\frac{1}{\sqrt{2\sigma^2\mu_{j,y}}} e^{-\frac{(x_j-\mu_{j,y})^2}{2\sigma^2}}$)=-$\frac{1}{2}$log($2\sigma^2$) -$\frac{1}{2}$log($\mu_{j,y}$)-$\frac{(x_j-\mu_{j,y})^2}{2\sigma^2}$=-$\frac{1}{2}$log($2\sigma^2$) -$\frac{1}{2}$log($\mu_{j,y}$)-$\frac{x_j^2-2x_j\mu_{j,y}+\mu_{j,y}^2}{2\sigma^2}$

So we get logPr(y=1)-logPr(y=0)+$\sum_{j=0}^{d}(-\frac{1}{2}log\mu_{j,y=1} - \frac{-2x_j\mu_{j,y=1}+\mu_{j,y=1}^2}{2\sigma^2})$-$\sum_{j=0}^{d}(-\frac{1}{2}log\mu_{j,y=0} - \frac{-2x_j\mu_{j,y=0}+\mu_{j,y=0}^2}{2\sigma^2}) \geq 0$

logPr(y=1)-logPr(y=0)+$\frac{d}{2}log\mu_{j,y=0} - \frac{d}{2}log\mu_{j,y=1}$+$\sum_{j=0}^{d}(-\frac{-2x_j\mu_{j,y=1}+\mu_{j,y=1}^2}{2\sigma^2})$-$\sum_{j=0}^{d}(-\frac{-2x_j\mu_{j,y=0}+\mu_{j,y=0}^2}{2\sigma^2}) \geq 0$

logPr(y=1)-logPr(y=0)+$\frac{d}{2}log\mu_{j,y=0} - \frac{d}{2}log\mu_{j,y=1}$+$\sum_{j=0}^{d}(-\frac{\mu_{j,y=1}^2}{2\sigma^2})$-$\sum_{j=0}^{d}(-\frac{\mu_{j,y=0}^2}{2\sigma^2})$+$\sum_{j=0}^{d}(\frac{\mu_{j,y=1}-\mu_{j,y=0}}{\sigma^2}x_j) \geq 0$

So logPr(y=1)-logPr(y=0)+ $\frac{d}{2}log\mu_{j,y=0} - \frac{d}{2}log\mu_{j,y=1}$+$\sum_{j=0}^{d}(-\frac{\mu_{j,y=1}^2}{2\sigma^2})$-$\sum_{j=0}^{d}(-\frac{\mu_{j,y=0}^2}{2\sigma^2})$ is constant,

$\sum_{j=0}^{d}(\frac{\mu_{j,y=1}-\mu_{j,y=0}}{\sigma^2}x_j)$ is linear related to **x**, so this naive Bayes classifier has a linear decision boundary.

4 Experiment

1 derivative of g(f(t))=g'(f(t))f'(t), so g'(w)=$\frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(-y_i \boldsymbol{x_i})$

2 The objective would be $\min_w\{\log(1+\exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)) + \frac{1}{\sigma^2}\boldsymbol{w}^T\boldsymbol{w}\}$,

The gradient with respect to weight vector is $\frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(-y_i \boldsymbol{x_i})+\frac{2}{\sigma^2}\boldsymbol{w}$.

3

a initialize **w**=0∈ R^n

b For epoch =1,2,…, T

a shuffle the training set

b For each training example $(\boldsymbol{x}_i, y_i)$∈ S:

if $y_i \boldsymbol{x_i}$<0

Update w ← w-$\gamma_t \nabla J^t$, that is, w=w-$\gamma_t(\frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(-y_i \boldsymbol{x_i})+\frac{2}{\sigma^2}\boldsymbol{w})$=(1-$\gamma_t \frac{2}{\sigma^2}$) $\boldsymbol{w}$+$\gamma_t \frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(y_i \boldsymbol{x_i})$

c return **w**

4

First choose different σ and use cross validation with fold 5 to test accuracy. Then choose sigma that has best accuracy to train on the whole training set. For each iteration, we use the weight we learned to get accuracy on test set and also get the value of formula: $\log(1+\exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)) + \frac{1}{\sigma^2}\boldsymbol{w}^T\boldsymbol{w}$

Here are the results generated by code.

(Please be aware that each time the results may not be the same because of random shuffle)

```
sigma= 0.1    Accuracy in cross validation is 0.6529466791393825

sigma= 0.2    Accuracy in cross validation is 0.48066729030246336

sigma= 0.5    Accuracy in cross validation is 0.6987839101964453

sigma= 0.7    Accuracy in cross validation is 0.6741502962270034

sigma= 1.0    Accuracy in cross validation is 0.7441534144059869
```

```
sigma= 2.0    Accuracy in cross validation is 0.8024633613969442

sigma= 5.0    Accuracy in cross validation is 0.8062051761771125

sigma= 10.0    Accuracy in cross validation is 0.8160274399750546

sigma= 50.0    Accuracy in cross validation is 0.7737761147489866

sigma= 100.0    Accuracy in cross validation is 0.8085438104147178
The best choice of sigma is sigma= 10.0 with accuracy= 0.8160274399750546
Now use the whole data to train with different epoch and test accuracy...
After 1 th iteration, the accuracy is 0.6649328794890428   The object value is 17711.57443026635
After 2 th iteration, the accuracy is 0.804145791104142   The object value is 17430.263532344157
After 3 th iteration, the accuracy is 0.6623704440279956   The object value is 17748.57093962526
After 4 th iteration, the accuracy is 0.8164990247447126   The object value is 17460.839818307
After 5 th iteration, the accuracy is 0.8140130798944429   The object value is 17146.918656958824
After 6 th iteration, the accuracy is 0.8207060083374765   The object value is 17370.82207343707
After 11 th iteration, the accuracy is 0.8225417829961372   The object value is 17411.761750768386
After 16 th iteration, the accuracy is 0.8174934026848204   The object value is 17352.163029208314
After 21 th iteration, the accuracy is 0.7447125865300035   The object value is 17593.81798965109
After 26 th iteration, the accuracy is 0.6661949745668719   The object value is 17754.985202173684
After 31 th iteration, the accuracy is 0.7513290243622596   The object value is 17444.34410023909
After 36 th iteration, the accuracy is 0.8283168241098405   The object value is 17168.304715058293
After 41 th iteration, the accuracy is 0.7577542356675718   The object value is 17585.293465607643
After 46 th iteration, the accuracy is 0.8268252571996787   The object value is 17363.06752422531
So to get best accuracy, epoch= 36  accuracy= 0.8283168241098405
```

The relationship between epoch and object formula: $\log(1+\exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)) + \frac{1}{\sigma^2}\boldsymbol{w}^T\boldsymbol{w}$ is below.

First the java program output txt file that saves information, then use matlab to plot it.