# COMP 440 Homework 7

Tony Chen(xc12) and Adam Wang(sw33)

November 2016

## 1 Naive Bayes, Perceptrons, Decision Trees, Neural Networks

a

- | Education | $P(Education \mid \leq 50K)$ | $P(Education \mid > 50K)$ |
  |---|---|---|
  | BS | 66.67% | 0% |
  | MS | 0% | 50% |
  | PhD | 33.33% | 50% |

  | Gender | $P(Gender \mid \leq 50K)$ | $P(Gender \mid > 50K)$ |
  |---|---|---|
  | male | 50% | 50% |
  | female | 50% | 50% |

  | Gender | $P(Citizenship \mid \leq 50K)$ | $P(Citizenship \mid > 50K)$ |
  |---|---|---|
  | US | 50% | 75% |
  | nonUS | 50% | 25% |

- | Education | Gender | Citizenship | Income |
  |---|---|---|---|
  | PhD | male | US | $> 50$ |
  | PhD | male | nonUS | $\leq 50$ |
  | MS | female | nonUS | $> 50$ |

b

- $x = (1, I(Education = BS), I(Education = MS), I(Gender = male), I(Citizenship = US))$
  $y = +1$ if $Income \leq 50K$, $y = -1$ if $Income > 50K$
  So $X$, in row-based order, can be represented as following:

  | Observation # | $x_0$ | $x_{BS}$ | $x_{MS}$ | $x_{male}$ | $x_{US}$ |
  |---|---|---|---|---|---|
  | 1 | 1 | 1 | 0 | 1 | 1 |
  | 2 | 1 | 0 | 1 | 1 | 0 |
  | 3 | 1 | 1 | 0 | 0 | 1 |
  | 4 | 1 | 0 | 0 | 1 | 0 |
  | 5 | 1 | 0 | 1 | 0 | 1 |
  | 6 | 1 | 0 | 0 | 0 | 0 |
  | 7 | 1 | 1 | 0 | 1 | 1 |
  | 8 | 1 | 0 | 0 | 1 | 1 |
  | 9 | 1 | 1 | 0 | 0 | 0 |
  | 10 | 1 | 0 | 0 | 0 | 1 |

  And $Y$ is $[+1, -1, +1, +1, -1, +1, +1, -1, +1, -1]$

- Based on the encoding above, we have:

| # of observations presented | $w_0$ | $w_{BS}$ | $w_{MS}$ | $w_{male}$ | $w_{US}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | -1 | 0 | 1 |
| 3 | 0 | 1 | -1 | 0 | 1 |
| 4 | 1 | 1 | -1 | 1 | 1 |
| 5 | 0 | 1 | -2 | 1 | 0 |
| 6 | 1 | 1 | -2 | 1 | 0 |
| 7 | 1 | 1 | -2 | 1 | 0 |
| 8 | 0 | 1 | -2 | 0 | -1 |
| 9 | 0 | 1 | -2 | 0 | -1 |
| 10 | 0 | 1 | -2 | 0 | -1 |

- Yes.

  At the 7th pass, $w$ will converge to $(1, 4, -3, 1, -3)$, which correctly label all observations.

c
- 

$$\begin{aligned} H(I) &= -p(I \leq 50K)log(p(I \leq 50K)) - p(I > 50K)log(p(I > 50K)) \\ &= -0.6 \times log(0.6) - 0.4 \times log(0.4) \\ &= 0.2923 \end{aligned}$$

$$\begin{aligned} H(I|E) &= -\sum_{(i,e)} p(I = i|E = e)p(E = e)log(p(I = i|E = e)) \\ &= -0.4 \times 1 \times 0 - 0.4 \times 0 - 0.2 \times 0 - 0.2 \times 1 \times 0 - 0.4 \times 0.5 \times log(0.5) - 0.4 \times 0.5 \times log(0.5) \\ &= 0.1204 \end{aligned}$$

$$\begin{aligned} H(I|G) &= -\sum_{(i,g)} p(I = i|G = g)p(G = g)log(p(I = i|G = g)) \\ &= -0.5 \times 0.6 \times log(0.6) - 0.5 \times 0.4 \times log(0.4) - 0.5 \times 0.6 \times log(0.6) - 0.5 \times 0.4 \times log(0.4) \\ &= 0.2929 \end{aligned}$$

$$\begin{aligned} H(I|C) &= -\sum_{(i,c)} p(I = i|C = c)p(C = c)log(p(I = i|C = c)) \\ &= -0.6 \times 0.5 \times log(0.5) - 0.6 \times 0.5 \times log(0.5) - 0.4 \times 0.75 \times log(0.75) - 0.4 \times 0.25 \times log(0.25) \\ &= 0.2783 \end{aligned}$$

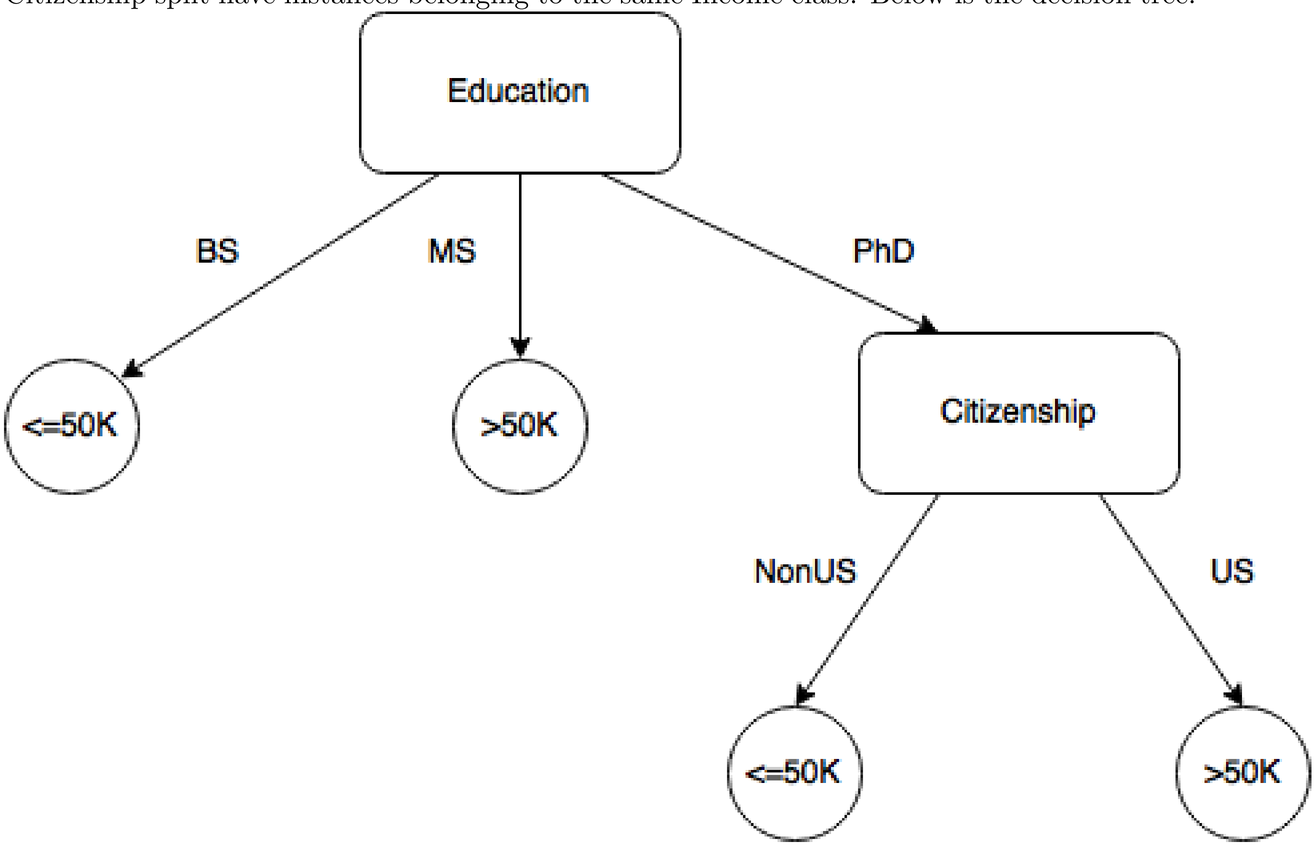
  So the information gain of education is $0.2923 - 0.1204 = 0.1719$, of gender is $0.2923 - 0.2923 = 0$, of citizenship is $0.2923 - 0.2783 = 0.014$. So education should be chosen as the root.

- The BS branch and the MS branch already have all instances belonging to the same Income class. For the PhD branch (all probabilities are those given $Education$ = PhD):

$$\begin{aligned}
H(I|G) &= -\sum_{(i,g)} p(I=i|G=g)p(G=g)log(p(I=i|G=g)) \\
&= -0.5 \times 0.5 \times log(0.5) - 0.5 \times 0.5 \times log(0.5) - 0.5 \times 0.5 \times log(0.5) - 0.5 \times 0.5 \times log(0.5) \\
&= 0.3010
\end{aligned}$$

$$\begin{aligned}
H(I|C) &= -\sum_{(i,c)} p(I=i|C=c)p(C=c)log(p(I=i|C=c)) \\
&= -0.5 \times 1 \times 0 - 0.5 \times 0 - 0.5 \times 1 \times 0 - 0.5 \times 0 \\
&= 0
\end{aligned}$$

So the next level of split on the PhD branch should be on Citizenship. Both branches of that Citizenship split have instances belonging to the same Income class. Below is the decision tree:




- 

| Education | Gender | Citizenship | Income |
|---|---|---|---|
| PhD | male | US | $> 50K$ |
| PhD | male | nonUS | $\leq 50K$ |
| MS | female | nonUS | $> 50K$ |

d
- $x = (I(Education = BS), I(Education = MS), I(Gender = male), I(Citizenship = US))$
  $y = 1$ if $Income \leq 50K$, $y = 0$ if $Income > 50K$
  So $X$, in row-based order, can be represented as following:

| Observation # | $x_{BS}$ | $x_{MS}$ | $x_{male}$ | $x_{US}$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 1 | 1 |
| 8 | 0 | 0 | 1 | 1 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 |

And $Y$ is $[1, 0, 1, 1, 0, 1, 1, 0, 1, 0]$

- ```
  import numpy as np
  from sklearn.neural_network import MLPClassifier
  from sklearn.model_selection import cross_val_score
  from sklearn.model_selection import KFold

  trainX = [[1,0,1,1],[0,1,1,0],[1,0,0,1],[0,0,1,0],[0,1,0,1],
          [0,0,0,0],[1,0,1,1],[0,0,1,1],[1,0,0,0],[0,0,0,1]]
  trainY = [1,0,1,1,0,1,1,0,1,0]
  testX = [[0,0,1,1],[0,0,1,0],[0,1,0,0]]
  trainX = np.asarray(trainX)
  trainY = np.asarray(trainY)
  testX = np.asarray(testX)

  for i in range(2,6):
          kf = KFold(n_splits=5)
          total_error = 0
          for train_index, test_index in kf.split(trainX):
                  clf = MLPClassifier(solver='lbfgs',hidden_layer_sizes=(i,))
                  clf.fit(trainX[train_index],trainY[train_index])
                  total_error += 2 - np.sum(trainY[test_index] ==
                          clf.predict(trainX[test_index]))
          print("With " + str(i) + " hidden units, cross validation error is "
                  + str(total_error / 5.0))
  ```

  The cross-validated training error is shown in the following table:

  | # of neurons | error |
  |---|---|
  | 2 | 0.6 |
  | 3 | 0.4 |
  | 4 | 0.2 |
  | 5 | 0 |
  ```

- The prediction is as following:

| Education | Gender | Citizenship | Income |
|---|---|---|---|
| PhD | male | US | $> 50K$ |
| PhD | male | nonUS | $\leq 50K$ |
| MS | female | nonUS | $> 50K$ |

The prediction is the same with the other methods.

# Text classification

## Spam classification

### Rule-based system

- dev error results of $n$ and $k$ thresholds:

| | $k = 10000$ | $k = 20000$ | $k = 30000$ |
|---|---|---|---|
| $n = 1$ | 0.1059 | 0.1639 | 0.4897 |
| $n = 2$ | 0.1651 | 0.1184 | 0.4779 |
| $n = 3$ | 0.2044 | 0.1065 | 0.4642 |

### Learning to distinguish spam

- Bigram features error table:

| # of examples | train error | dev error |
|---|---|---|
| 500 | 0 | 0.0910 |
| 1000 | 0 | 0.0636 |
| 1500 | 0 | 0.0505 |
| 2000 | 0 | 0.0424 |
| 2500 | 0 | 0.0380 |
| 3000 | 0.0003 | 0.0380 |
| 3500 | 0.0006 | 0.0349 |
| 4000 | 0.0035 | 0.0312 |
| 4500 | 0.0002 | 0.0324 |
| 5000 | 0.0026 | 0.0368 |

Generally, the more examples for training the better accuracy will be achieved on development set. However after the number of training exceeds certain amount (4000), no additional benefit is gained through adding examples.

## Sentiment classification

- The error table:

| | train error | dev error |
|---|---|---|
| unigram | 0.0328 | 0.1685 |
| bigram | 0 | 0.1629 |

- The error table:

| # of iterations | train error | dev error |
|---|---|---|
| 1 | 0.2920 | 0.3652 |
| 2 | 0.4538 | 0.4831 |
| 3 | 0.1423 | 0.2921 |
| 4 | 0.5024 | 0.5112 |
| 5 | 0.0499 | 0.1629 |
| 6 | 0.4793 | 0.5056 |
| 7 | 0.1509 | 0.3708 |
| 8 | 0.0195 | 0.1629 |
| 9 | 0.0292 | 0.1573 |
| 10 | 0.0146 | 0.1461 |
| 11 | 0.1046 | 0.3090 |
| 12 | 0.0085 | 0.1685 |
| 13 | 0.0389 | 0.2528 |
| 14 | 0.0170 | 0.1966 |
| 15 | 0.0085 | 0.1798 |
| 16 | 0.0049 | 0.1798 |
| 17 | 0 | 0.1629 |
| 18 | 0 | 0.1629 |
| 19 | 0 | 0.1629 |
| 20 | 0 | 0.1629 |

The dev set error does not monotonically decrease with iteration number.
This is because during the process of convergence the perceptron temporarily overfitted on (biased by) a subset of the training data that are not representative of the population, resulting in a temporary huge drop in accuracy.

### Document categorization

- The error table:

| | train error | dev error |
|---|---|---|
| unigram | 0.0039 | 0.1196 |
| bigram | 0 | 0.1003 |

## Image classification

## Relating Naive Bayes classifiers and perceptrons

First, for naive Bayes classifier we have $P(y=+1|f) \sim P(y=+1)\prod_i P(f_i|y=+1)$ and $P(y=-1|f) \sim P(y=-1)\prod_i P(f_i|y=-1)$. Since naive Bayes will classify $y$ as $+1$ when $P(y=+1|f) > P(y=-1|f)$ and vice versa, some perceptron will always classify $y$'s same as the Bayes as long as $w^T f = P(y = +1|f) - P(y=-1|f)$.
We can first get the intercept $w_0$ by setting all $f_i = 0$, which means $w^T f = w_0$, so we get:

$$w_0 = P(y=+1)\prod_i P(f_i=0|y=+1) - P(y=-1)\prod_i P(f_i=0|y=-1)$$

Similarly, we can get any weight $w_j$ by setting all $f_i = 0$ except $f_j$, which means $w^T f = w_0 + w_j$, so we get:

$$\begin{aligned}
w_j &= \frac{P(y=+1)P(f_j=1|y=+1)\prod_i P(f_i=0|y=+1)}{P(f_j=0|y=+1)} \\
&\quad - \frac{P(y=-1)P(f_j=1|y=-1)\prod_i P(f_i=0|y=-1)}{P(f_j=0|y=-1)} - w_0 \\
&= \frac{P(y=+1)[P(f_j=1|y=+1) - P(f_j=0|y=+1)]\prod_i P(f_i=0|y=+1)}{P(f_j=0|y=+1)} \\
&\quad - \frac{P(y=-1)[P(f_j=1|y=-1) - P(f_j=0|y=-1)]\prod_i P(f_i=0|y=-1)}{P(f_j=0|y=-1)}
\end{aligned}$$

Since all the probabilities are non-zero, we have proven that such binary naive Bayes can be represented by a perceptron that always produces the same decision.