Bayes' Theorem

Bayes' rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- In practice we are most interested in dealing with hypotheses h and data D
  - $h =$ "I have a cold"
  - D="runny nose", "watery eyes", "coughing"
- $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$

Some Terminology

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$: **Prior probability** of h
- $P(D|h)$: **Conditional probability** of D given that h happened
- $P(D)$: Normalizing probability
- $P(h|D)$: is the **posterior probability** of h given D ✓

Naive Bayes Classifier

- Training input: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^n$ independent
- Each input $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)})$
- $x_i, i \in \{1, 2, \ldots n\}$ are the features of the data points x
- Output labels $y^{(1)}, y^{(2)}, \ldots, y^{(m)}, y^{(j)} \in \{0, 1, \ldots, k\}$
- Goal: predict labels of new unseen data x
- $P(y = j|x) =?$ for $j \in \{0, 1, \ldots, k\}$

Naive Bayes Classifier: Recap

$$
\begin{aligned}
\hat{y} &= \arg\max_j P(y = j | x_1, x_2, \ldots, x_n) \\
&= \arg\max_j \frac{P(x_1, x_2, \ldots, x_n | y = j) P(y = j)}{P(x_1, x_2, \ldots, x_n)} \\
&= \arg\max_j P(x_1, x_2, \ldots, x_n | y = j) P(y = j) \\
&= \arg\max_j log P(y = j) + \sum_{i=1}^{n} log P(x_i | y = j)
\end{aligned}
$$

Naive Bayes Algorithm

Naive_bayes_train (examples):

For each target y :

$\hat{P}(y = j) \leftarrow$ estimate $P(y = j)$

For each feature x_i:

$\hat{P}(x_i|y = j) \leftarrow$ estimate $P(x_i|y = j)$

Classify_new_instance (x):

$$\hat{y} = \arg\max_j log\hat{P}(y = j) + \sum_{i=1}^{n} log\hat{P}(x_i|y = j)$$

Implementation

How to represent our Data?

- In Class we saw the Bernoulli model (Bernoulli NB)
- $p(x = 1) = p$ and $p(x = 0) = 1 - p$
- In the homework we will be implementing the Multinomial model (Multinomial NB)
- It models the probability of counts for each side of a k-sided die rolled n times independently
- $P(x_1, x_2, \ldots, x_k) = \frac{n!}{x_1!x_2!\cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$

Spam Filter (Bernoulli Model)

- Need to convert text into something suitable for ML
- Let $|V|$ denote the total number of distinct words in the training set (vocabulary)
- Then represent each review as a $|V|$ dimensional vector
- Where $x_i^{(j)} \in \{0, 1\}$ whether word i appears in review j or not
- Example

$$x^{(j)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{abacus} \\ \text{abandon} \\ \text{abate} \\ \text{abbey} \\ \vdots \\ \text{zymurgy} \end{matrix}$$

- $y^{(j)} = 1$ if review is good and 0 otherwise

Spam Filter (Multinomial Model)

- Similar to Bernoulli only instead of 0 or 1 we use the number of times he word appears in the email
- More information ==> better estimator in practice for text classification
- $x_i^{(j)} \in \{0, 1, 2, \ldots\}$ whether word i appears in review j or not
- Example

$$x^{(j)} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{abacus} \\ \text{abandon} \\ \text{abate} \\ \text{abbey} \\ \vdots \\ \text{zymurgy} \end{matrix}$$

- $y^{(j)} = 1$ if review is good and 0 otherwise

Naive Bayes Classifier

Bernoulli NB Parameters: For each category (e.g. spam or no spam) $|V|$ Bernoulli parameters and 1 more Bernoulli for the prior. From lecture...

$$\hat{\theta}_{j|y=1} = \frac{N_{j,y=1}}{N_{y=1}}, \qquad \hat{\theta}_{j|y=0} = \frac{N_{j,y=0}}{N_{y=0}}, \hat{\theta}_{y=1} = \frac{N_{y=1}}{N_{y=0} + N_{y=1}}$$

where $N_{j,y=1}$ is the number of times feature j appeared in data labeled with 1 and $N_{y=1}$ is the total number of data labeled with 1

Multinomial NB Parameters: For each category (e.g. spam or no spam) $|V|$ parameters (the probabilities of the Multinomial) and 1 more Bernoulli (still) for the prior

$$\hat{\theta}_{j|y=1} = \frac{N_{j,y=1}}{N_{y=1}}, \qquad \hat{\theta}_{j|y=0} = \frac{N_{j,y=0}}{N_{y=0}}, \hat{\theta}_{y=1} = \frac{N_{y=1}}{N_{y=0} + N_{y=1}}$$

where $N_{j,y=1}$ is the number of times feature j appears in data labeled with 1 and $N_{y=1}$ is the total number of appearances of all features in data labeled with 1

Prediction

- We predict given a new e-mail by

$$\hat{y} = argmax_k \{ log\hat{\theta}_{y=k} + \sum_{i=1}^{|V|} log\hat{\theta}_{i|y=k} \}$$

- Laplace smoothing to avoid $\frac{0}{0}$ cases ?

Prediction

- We predict given a new e-mail by

$$\hat{y} = argmax_k \{ log\hat{\theta}_{y=k} + \sum_{i=1}^{|V|} log\hat{\theta}_{i|y=k} \}$$

- Laplace smoothing to avoid $\frac{0}{0}$ cases ?

$$\hat{\theta}_{j|y=1} = \frac{N_{j,y=1} + 1}{N_{y=1} + |V|}, \qquad \hat{\theta}_{j|y=0} = \frac{N_{j,y=0} + 1}{N_{y=0} + |V|},$$

$$\hat{\theta}_{y=1} = \frac{N_{y=1} + 1}{N_{y=0} + N_{y=1} + 2}$$

Multinomial Example

$$X_{y=1} = \begin{bmatrix} 4 & 1 & 1 & 2 & 3 \\ 0 & 5 & 1 & 3 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 & 1 \\ 0 & 4 & 3 & 2 & 2 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} \text{at} \\ \text{break} \\ \text{home} \\ \text{safe} \\ \text{spring} \\ \text{stay} \end{matrix}$$

What is $\hat{\theta}_{\text{safe}|y=1}$ and $\hat{\theta}_{\text{home}|y=1}$?

Multinomial Example

$$X_{y=1} = \begin{bmatrix} 4 & 1 & 1 & 2 & 3 \\ 0 & 5 & 1 & 3 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 & 1 \\ 0 & 4 & 3 & 2 & 2 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} \text{at} \\ \text{break} \\ \text{home} \\ \text{safe} \\ \text{spring} \\ \text{stay} \end{matrix}$$

What is $\hat{\theta}_{\text{safe}|y=1}$ and $\hat{\theta}_{\text{home}|y=1}$?

$$\hat{\theta}_{\text{safe}|y=1} = \frac{7+1}{41+6}$$

$$\hat{\theta}_{\text{home}|y=1} = \frac{1+1}{41+6}$$

Model Evaluation

- Classification accuracy: number of correct predictions divided by total number of predictions
- Confusion matrix

	Predicted: NO	Predicted: YES
Actual: NO	T_1	T_2
Actual: YES	T_3	T_4

- False positive rate $= \frac{T_2}{T_1+T_2}$
- Accuracy $= (T1 + T_4)/\sum_i T_i$

Laplace Smoothing

- Laplace smoothing
- $\hat{\theta}_{j|y=1} = \frac{N_{j,y=1}+a}{N_{y=1}+a|V|}$,
- $a \geq 0$ is now a tuning parameter

Interpolation Smoothing

- In practice Laplace smoothing often performs poorly:
  - when domain of X is large
  - when domain of Y is large
- If we don't know much about (X|Y), we should avoid allowing that probability to have an influence over the posterior
- Linear interpolation biases P(x|y) toward P(x)
  - $LIN_\alpha P(x|y) = \alpha P(x|y) + (1-\alpha)P(x)$
  - When α large, we trust MLE estimates
  - When α small, we trust our priors
- Many varieties of smoothing
  - All try to compensate for a lack of training data
  - Allow parameters to generalize better to new data

Naive Bayes Algorithm

Naive_bayes_train (examples):

For each target y :

$\hat{P}(y = j) \leftarrow$ estimate $P(y = j)$

For each feature x_i:

$\hat{P}(x_i|y = j) \leftarrow$ estimate $P(x_i|y = j)$

Classify_new_instance (x):

$$\hat{y} = \arg\max_j log\hat{P}(y = j) + \sum_{i=1}^{n} log\hat{P}(x_i|y = j)$$