# Generative Models

Swati Mishra

Applications of Machine Learning (4AL3)

Fall 2024

McMaster University

ENGINEERING

# Generative Models

So far we have modeled are problem as:

$$P(Y = k|X = x)$$

An alternate approach is, we model the distribution of the predictors X separately in each of the response classes and then use Bayes Theorem to flip them around.

$$f_k(X) \equiv P(X|Y = k)$$

$f_k(X)$ denote the density function of X for an observation that comes from the kth class.

# Generative Models

So far we have modeled are problem as:

$$P(Y = k|X = x)$$

An alternate approach is, we model the distribution of the predictors X separately in each of the response classes and then use Bayes Theorem to flip them around.

$$f_k(X) \equiv P(X|Y = k)$$

$f_k(X)$ denote the density function of X for an observation that comes from the kth class.

Bayes theorem states that:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

$\pi_k$ = prior probability that a randomly chosen observation comes from the kth class.

$\pi_k$ and $f_k(x)$ are estimates

# Generative versus Discriminative model

Generative models model the problem

$$P(x, y) = X, Y \rightarrow [0,1]$$

Discriminative models model the problem

$$P(y|x) = X, Y \rightarrow [0,1]$$

Bayes theorem states that:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

# Generative Models

- Most useful when :
  - Differences between classes are too huge to quantify
  - If the distribution of the predictors is approximately normal in each of the classes.
  - Sample size is small.

# Generative Models

- Most useful when :
  - Differences between classes are too huge to quantify
  - If the distribution of the predictors is approximately normal in each of the classes.
  - Sample size is small.

“I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I’ve seen it several times, and I'm always happy to see it again whenever I have a friend who hasn’t seen it yet!”

Example Task: Text Classification

# Converting Text to Vectors

- Techniques used:
  - Bag of Words

| word | frequency |
|---|---|
| It | 6 |
| I | 5 |
| the | 4 |
| satirical | 1 |
| whimsical | 1 |
| would | 1 |
| adventure | 1 |
| and | 3 |

"I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!"

# Converting Text to Vectors

- Techniques used:
  - TF- IDF

| word | position |
|---|---|
| It | 6 |
| I | 1 |
| the | 4 |
| satirical | 9 |
| whimsical | 1 |
| would | 1 |
| adventure | 1 |
| and | 3 |

"I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!"

McMaster University

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $\mathrm{c} \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{\mathrm{c} \in C}{argmax}\ P(c|d)$$

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $\mathrm{c} \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{\mathrm{c} \in C}{argmax}\, P(c|d)$$

- The intuition is to use Bayesian classification to transform into other probabilities.

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

McMaster University

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $\mathrm{c} \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{\mathrm{c} \in C}{argmax}\, P(c|d)$$

- The intuition is to use Bayesian classification to transform into other probabilities.

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

Simplifying:

$$c' = \underset{\mathrm{c} \in C}{argmax}\, P(c|d) \qquad = \underset{\mathrm{c} \in C}{argmax}\, \frac{P(d|c)\, P(c)}{P(d)}$$

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c'$ which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\, P(c|d)$$

- The intuition is to use Bayesian classification to transform into other probabilities.

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

Simplifying:

$$c' = \underset{c \in C}{argmax}\, P(c|d) \quad = \underset{c \in C}{argmax} \frac{P(d|c)\, P(c)}{P(d)} \quad = \underset{c \in C}{argmax}\, P(d|c)P(c)$$

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\, P(c|d)$$

- The intuition is to use Bayesian classification to transform into other probabilities.

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

Simplifying:

$$c' = \underset{c \in C}{argmax}\, P(c|d) \quad = \underset{c \in C}{argmax} \frac{P(d|c)\,P(c)}{P(d)} \quad = \underset{c \in C}{argmax}\, \overbrace{P(d|c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Prior}}$$

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c'$ which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\, P(c|d) = \underset{c \in C}{argmax}\, \underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}$$

$$= \underset{c \in C}{argmax}\, P(f_{1,} f_{2,} f_3 \ldots\ldots f_n|c)P(c)$$

McMaster University

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $\mathrm{c} \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{\mathrm{c} \in C}{argmax}\, P(c|d) \quad = \underset{\mathrm{c} \in C}{argmax}\, \underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}$$

$$= \underset{\mathrm{c} \in C}{argmax}\, P(f_{1,} f_{2,} f_3 \ldots\ldots f_n|c)P(c)$$

1: Bag of words assumption!

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c$' which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\, P(c|d) = \underset{c \in C}{argmax}\, \underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}$$

$$= \underset{c \in C}{argmax}\, P(f_{1,} f_{2,} f_3 \ldots\ldots f_n|c)P(c)$$

2: Conditional independence assumption!

$$P(f_{1,} f_{2,} f_3 \ldots\ldots f_n|c) = \mathrm{P(f_1|c).P(f_2|c).P(f_3|c)\ldots P(fn|c)}$$

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c'$ which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\, P(c|d) \quad = \underset{c \in C}{argmax}\, \underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}$$

$$= \underset{c \in C}{argmax}\, P(f_{1,} f_{2,} f_3 \ldots\ldots f_n|c)P(c)$$

$$C = argmax \prod_{f \in F} P(c)\, P(f|c)$$

1: Bag of words assumption!

2: Conditional independence assumption!

McMaster University

# Naïve Bayes

- It is probabilistic classifier, meaning that for a input document $d$, out of all classes $c \in C$ the classifier returns the class $c'$ which has the maximum posterior probability given $d$.

$$c' = \underset{c \in C}{argmax}\ P(c|d) = \underset{c \in C}{argmax}\ \underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}$$

$$= \underset{c \in C}{argmax}\ P(f_1, f_2, f_3 \ldots\ldots f_n|c)P(c)$$

$$C = argmax \log(P(c)) + \sum_{i \in word\ positions} P(f_i|c)$$

McMaster University

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c))$$

$P(c)$ What percentage of the documents in our training set are in each class c.

$$\frac{\text{Number of d in class c}}{\text{Number of documents (d)}}$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

$P(f_i|c)$ What fraction of times the word $f_i$ appears among all words in all documents of class c.

$$\frac{\text{Count }(f_i\text{ , c})}{\sum_{f \in V} \text{Count }(f\text{, c})}$$

V = vocabulary

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

$P(f_i|c)$ What fraction of times the word $f_i$ appears among all words in all documents of class c.

$$\frac{\text{Count}\ (f_i\ ,\ \text{c})}{\sum_{f \in V} \text{Count}\ (f, \text{c})}$$

V = vocabulary

What could go wrong?

McMaster University

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

$P(f_i|c)$ What fraction of times the word $f_i$ appears among all words in all documents of class c.

$$\frac{\text{Count }(f_i\text{ , c})+1}{\sum_{f \in V} (\text{Count }(f\text{, c})+1)}$$

Add-1 smoothening

Solution: Laplacian smoothening

V = vocabulary

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

$P(f_i|c)$ What fraction of times the word $f_i$ appears among all words in all documents of class c.

$$\frac{\text{Count}\ (f_i\ ,\ \text{c})+1}{\sum_{f \in V} (\text{Count}\ (f, \text{c})) + |V|}$$

V = vocabulary

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

$P(f_i|c)$ What fraction of times the word $f_i$ appears among all words in all documents of class c.

$$\frac{\text{Count}\ (f_i\ ,\ \text{c})+1}{\sum_{f \in V} (\text{Count}\ (f, \text{c})) + |V|}$$

V = vocabulary

McMaster University

# Training Naïve Bayes

**function** TRAIN NAIVE BAYES(D, C) **returns** $V$, $\log P(c)$, $\log P(w|c)$

**for each** class $c \in C$ # Calculate $P(c)$ terms
 $N_{doc}$ = number of documents in D
 $N_c$ = number of documents from D in class c
 $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
 $V \leftarrow$ vocabulary of D
 $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class $c$
 **for each** word $w$ in V # Calculate $P(w|c)$ terms
  $count(w,c) \leftarrow$ # of occurrences of $w$ in $bigdoc[c]$
  $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
**return** *logprior*, *loglikelihood*, $V$

**function** TEST NAIVE BAYES(*testdoc*, *logprior*, *loglikelihood*, C, V) **returns** best $c$

**for each** class $c \in C$
 $sum[c] \leftarrow logprior[c]$
 **for each** position $i$ in *testdoc*
  $word \leftarrow testdoc[i]$
  **if** $word \in V$
   $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
**return** $\text{argmax}_c\ sum[c]$

McMaster University

# Training Naïve Bayes

- Goal is to learn probabilities $C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c))$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$P(-) = ?$

$P(+) = ?$

$$\frac{\text{Number of d in class c}}{\text{Number of documents (d)}}$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$$P(-) = \frac{3}{5}$$

$$P(+) = \frac{2}{5}$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$P(\text{predictable}|\,-)$

$P(\text{predictable}|\,+)$

$$\frac{\text{Count}(f_i, \text{c})}{\sum_{f \in V} \text{Count}(f, \text{c})}$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax\ \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$$P(\text{predictable}|-) = \frac{1+1}{14+20}$$

$$P(\text{predictable}|+) = \frac{0+1}{9+20}$$

$$\frac{\text{Count}(f_i, \text{c})}{\sum_{f \in V} \text{Count}(f, \text{c})}$$

# Training Naïve Bayes

- Goal is to learn probabilities $C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$P(\text{with}|-)$

$P(\text{with}|+)$

$$\frac{\text{Count}(f_i, \text{c})}{\sum_{f \in V} \text{Count}(f, \text{c})}$$

McMaster University

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$P(\text{no}|-)$

$P(\text{no}|+)$

$$\frac{\text{Count}(f_i, c)}{\sum_{f \in V} \text{Count}(f, c)}$$

McMaster University

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

$$P(\text{fun}|-) = \frac{0+1}{14+20}$$

$$P(\text{fun}|+) = \frac{1+1}{9+20}$$

$$\frac{\text{Count}(f_i, c)}{\sum_{f \in V} \text{Count}(f, c)}$$

# Training Naïve Bayes

- Goal is to learn probabilities

$$C = argmax \log(P(c) + \sum_{i \in word\ positions} P(f_i|c)$$

| | Label | documents |
|---|---|---|
| Training | - | just plain boring |
| Training | - | entirely predictable and lacks energy |
| Training | - | no surprises and very few laughs |
| Training | + | very powerful |
| Training | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

P(-)P(S|-)

P(+)P(S|+)

Maximum of the two

McMaster University

# Readings

***Required Readings:***

Introduction to Statistical Learning

- Chapter 10 – Section 10.3 page 406 - 412

***Supplemental Readings (Not required but recommended)*:**

Deep Learning

- Chapter 9 – page 330 – 340

**Reference**: Lecture Material Adopted from Dan Jurafsky and James H. Martin Book on Speech and Language Processing Chapter 4

# Thank You