Naive Bayes Classification

Professor Ameet Talwalkar
Registering for Course

- No more issues – there is now space and you can register online
Homework 2

HW1

- Available online since last Friday
- Due next Wednesday (2/1)
Outline

1 Administration

2 Review of last lecture

3 Naive Bayes
Many decisions are tree structures

**Medical treatment**

- **Fever**
  - $T > 100$
  - $T < 100$
  - **Treatment #1**
  - **Muscle Pain**
    - High
    - Low
    - **Treatment #2**
    - **Treatment #3**

**Salary in a company**

- **Degree**
  - High School
  - College
  - Graduate
  - **Work Experience**
    - < 5yr
    - > 5yr
    - **Salary**
      - $X_1$
      - $X_2$
      - $X_3$
      - $X_4$
      - $X_5$
      - $X_6$
A tree partitions the feature space

\[ x_1 > \theta_1 \quad x_2 \leq \theta_2 \quad x_1 \leq \theta_4 \quad x_2 \leq \theta_2 \quad x_2 > \theta_3 \]

A (A) B (B) C (C) D (D) E (E)
Learning a tree model

Three things to learn:

1. The structure of the tree.
2. The threshold values ($\theta_i$).
3. The values for the leafs (A, B, ...).
Learning a tree model

Three things to learn:

1. The structure of the tree.
2. The threshold values \((\theta_i)\).
3. The values for the leaves \((A, B, \ldots)\).
First decision: at the root of the tree

**Which attribute to split?**

*Patrons?* is a better choice—gives information about the classification

Idea: use information gain to choose which attribute to split
How to measure information gain?

Idea:

Gaining information reduces uncertainty

Use to entropy to measure uncertainty

If a random variable $X$ has $K$ different values, $a_1, a_2, \ldots, a_K$, its entropy is given by

$$H[X] = - \sum_{k=1}^{K} P(X = a_k) \log P(X = a_k)$$

the base can be 2, though it is not essential (if the base is 2, the unit of the entropy is called “bit”)
Examples of computing entropy

**Entropy**

\[ H(X) = 0.8360 \]

\[ H(X) = 1.3863 \]

\[ H(X) = 0 \]
Do we split on “Non” or “Some”? 

No, we do not 

The decision is deterministic, as seen from the training data
What is the optimal Tree Depth?

We need to be careful to pick an appropriate tree depth. If the tree is too deep, we can overfit. If the tree is too shallow, we underfit. Max depth is a hyperparameter that should be tuned by the data. Alternative strategy is to create a very deep tree, and then to prune it (see Section 9.2.2 in ESL for details). If leaves aren’t completely pure, we predict using majority vote.
What is the optimal Tree Depth?

- We need to be careful to pick an appropriate tree depth
  - If the tree is too deep, we can overfit
  - If the tree is too shallow, we underfit
- Max depth is a hyperparameter that should be tuned by the data
- Alternative strategy is to create a very deep tree, and then to prune it (see Section 9.2.2 in ESL for details)
- If leaves aren’t completely pure, we predict using majority vote
We stop after the root (first node)
Computational Considerations

Numerical Features

We could split on any feature, with any threshold. However, for a given feature, the only split points we need to consider are the \( n \) values in the training data for this feature. If we sort each feature by these \( n \) values, we can quickly compute our impurity metric of interest (cross entropy or others).

\[
\text{This takes } O(dn \log n) \text{ time.}
\]

Categorical Features

Assuming \( q \) distinct categories, there are \( 2^q - 1 \) possible partitions we can consider. However, things simplify in the case of binary classification (or regression), and we can find the optimal split (for cross entropy and Gini) by only considering \( q - 1 \) possible splits (see Section 9.2.4 in ESL for details).
Computational Considerations

**Numerical Features**

- We could split on any feature, with any threshold
- However, for a given feature, the only split points we need to consider are the the $n$ values in the training data for this feature.
- If we sort each feature by these $n$ values, we can quickly compute our impurity metric of interest (cross entropy or others)
  - This takes $O(dn \log n)$ time

**Categorical Features**

Assuming $q$ distinct categories, there are $2^q - 1$ possible partitions we can consider. However, things simplify in the case of binary classification (or regression), and we can find the optimal split (for cross entropy and Gini) by only considering $q - 1$ possible splits (see Section 9.2.4 in ESL for details).
Computational Considerations

Numerical Features
- We could split on any feature, with any threshold
- However, for a given feature, the only split points we need to consider are the the $n$ values in the training data for this feature.
- If we sort each feature by these $n$ values, we can quickly compute our impurity metric of interest (cross entropy or others)
  - This takes $O(dn \log n)$ time

Categorical Features
- Assuming $q$ distinct categories, there are $2^{q-1} - 1$ possible partitions we can consider.
- However, things simplify in the case of binary classification (or regression), and we can find the optimal split (for cross entropy and Gini) by only considering $q - 1$ possible splits (see Section 9.2.4 in ESL for details).
Summary of learning trees

**Advantages of using trees**

- Can be interpreted by humans
- Computationally efficient
- Handles both numerical and categorical data
- It is parametric thus compact: unlike NNC, we do not have to carry our training instances around
- Building block for various ensemble methods (more on this later)

**Disadvantages**

- Heuristic training techniques
  - Finding partition of space that minimizes empirical error is NP-hard
  - We resort to greedy approaches with limited theoretical underpinnings
Outline

1. Administration

2. Review of last lecture

3. Naive Bayes
   - Motivating Example
   - Naive Bayes Model
   - Parameter Estimation
I'm going to be rich!!

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floor
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT **US$10 MILLION**

It is my modest obligation to write you this letter in regards to the authorization of your owed payment through our most respected financial institution (AFRI BANK PLC). I am Mr. Aminu Saleh, The Director, Foreign Operations Department, AFRI Bank Plc, NIGERIA. The British Government, in conjunction with the US GOVERNMENT, WORLD BANK, UNITED NATIONS ORGANIZATION on foreign payment matters, has empowered my bank after much consultation and consideration, to handle all foreign payments and release them to their appropriate beneficiaries with the help of a representative from Federal Reserve Bank.

To facilitate the process of this transaction, please kindly re-confirm the following information below:

1) Your full Name and Address:
2) Phones, Fax and Mobile No.:
3) Profession, Age and Marital Status:
4) Copy of any valid form of your Identification:

A daily battle
How to tell spam from ham?

FROM THE DESK OF MR. AMINU SALEH
DIRECTOR, FOREIGN OPERATIONS DEPARTMENT
AFRI BANK PLC
Afribank Plaza,
14th Floor
51/55 Broad Street,
P.M.B 12021 Lagos-Nigeria

Attention: Honorable Beneficiary,

IMMEDIATE PAYMENT NOTIFICATION VALUED AT US$10 MILLION

Dear Ameet,

Do you have 10 minutes to get on a videocall before 2pm?

Thanks,

Stefano
How might we create features?
Intuition

Q: How might a human solve this problem?

A: Simple strategy would be to look for keywords that we often associate with spam

Spam emails

we expect to see words like “money”, “free”, “bank account”, “viagra”

Ham emails

word usage is more spread out with few ‘spammy’ words
Simple strategy: count the words

Bag-of-word representation of documents (and textual data)

\[
\begin{pmatrix}
\text{free} & 100 \\
\text{money} & 2 \\
\vdots & \vdots \\
\text{account} & 2 \\
\vdots & \vdots
\end{pmatrix}
\]

\[
\begin{pmatrix}
\text{free} & 1 \\
\text{money} & 1 \\
\vdots & \vdots \\
\text{account} & 2 \\
\vdots & \vdots
\end{pmatrix}
\]
Weighted sum of those telltale words

different weights for spam and ham: representing how compatible the word pattern is to each category

\[
\begin{pmatrix}
\text{free} & 100 \\
\text{money} & 2 \\
\vdots & \vdots \\
\text{account} & 2 \\
\vdots & \vdots \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
100 \times 0.2 \\
2 \times 0.3 \\
\vdots \\
2 \times 0.3 \\
\vdots \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
100 \times 0.01 \\
2 \times 0.02 \\
\vdots \\
2 \times 0.01 \\
\vdots \\
\end{pmatrix}
\]
Weighted sum of those telltale words

different weights for spam and ham: representing how compatible the word pattern is to each category

\[
\begin{pmatrix}
100 \times 0.2 \\
2 \times 0.3 \\
\vdots \\
2 \times 0.3 \\
\end{pmatrix}
\]

\[= 3.2\]

\[
\begin{pmatrix}
100 \times 0.01 \\
2 \times 0.02 \\
\vdots \\
2 \times 0.01 \\
\end{pmatrix}
\]

\[= 1.03\]
Our intuitive model of classification

Assign weight to each word

Compute compatibility score to “spam”

\[
\text{# of “free” } \times a_{\text{free}} + \text{# of “account” } \times a_{\text{account}} + \text{# of “money” } \times a_{\text{money}}
\]

Compute compatibility score to “ham”:

\[
\text{# of “free” } \times b_{\text{free}} + \text{# of “account” } \times b_{\text{account}} + \text{# of “money” } \times b_{\text{money}}
\]

Make a decision:

if spam score > ham score then spam
else ham
How do we get the weights?
How do we get the weights?

Learn from experience
get a lot of spams
get a lot of hams

But what to optimize?
Naive Bayes model for identifying spam

Class label: binary

\[ y = \{\text{spam, ham}\} \]

Features: word counts in the document (Bag-of-word)

Ex: \[ x = \{('free', 100), ('lottery', 10), ('money', 10), , ('identification', 1)\} \]

Each pair is in the format of \((w_i, \#w_i)\), namely, a unique word in the dictionary, and the number of times it shows up
Naive Bayes Model (Intuitively)

Features: word counts in the document

Ex: \( x = \{('free', 100), ('identification', 2), ('lottery', 10), ('money', 10), \ldots \} \)

Model: Naive Bayes (NB)

\[
p(x | \text{spam}) = p('free'|\text{spam})^{100} p('identification'|\text{spam})^2 p('lottery'|\text{spam})^{10} p('money'|\text{spam})^{10} \ldots
\]
Naive Bayes Model (Intuitively)

Features: word counts in the document

Ex: \( x = \{('free', 100), ('identification', 2), ('lottery', 10), ('money', 10), \ldots \} \)

Model: Naive Bayes (NB)

\[
p(x|\text{spam}) = p('free'|\text{spam})^{100} p('identification'|\text{spam})^2 \]
\[
p('lottery'|\text{spam})^{10} p('money'|\text{spam})^{10} \ldots
\]

Parameters to be estimated are conditional probabilities: \( p('free'|\text{spam}), p('free'|\text{ham}), \text{etc} \)
Naive Bayes Model

- Intuitively this makes some sense (even if it seems simple)
- We’ll now discuss the following:
  - Formal modeling assumptions for NB, and why it’s ‘naive’
  - NB classification rule converges to Bayes Optimal under these assumptions
  - How to estimate model parameters
Naive Bayes Model

\[ p(x|y) = p(w_1|y)^{\#w_1} p(w_2|y)^{\#w_2} \cdots p(w_m|y)^{\#w_m} = \prod_{i} p(w_i|y)^{\#w_i} \]

These conditional probabilities are model parameters.

Recall that each data point is a tuple \((w_i, \#w_i)\), namely, a unique dictionary word and the \# of times it shows up.
What is naive about this?
Strong assumption of conditional independence:

\[ p(w_i, w_j | y) = p(w_i | y)p(w_j | y) \]

Previous example: \( p(x | \text{spam}) = p(’free’ | \text{spam})^{100}p(’identification’ | \text{spam})^{2} \)
\[ p(’lottery’ | \text{spam})^{10}p(’money’ | \text{spam})^{10} \ldots \]

This assumption makes estimation much easier (as we’ll see)
Naive Bayes classification rule

For any document $x$, we want to compare

$$p(\text{spam}|x) \text{ and } p(\text{ham}|x)$$
Naive Bayes classification rule

For any document $x$, we want to compare

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Recall that Bayes Optimal classifier uses the posterior probability
Naive Bayes classification rule

For any document $x$, we want to compare

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Recall that Bayes Optimal classifier uses the posterior probability

$$f^*(x) = \begin{cases} 
  1 & \text{if } p(y = 1|x) \geq p(y = 0|x) \\
  0 & \text{if } p(y = 1|x) < p(y = 0|x) 
\end{cases}$$
Naive Bayes classification rule

For any document $x$, we want to compare

$$p(\text{spam}|x) \quad \text{and} \quad p(\text{ham}|x)$$

Recall that Bayes Optimal classifier uses the posterior probability

$$f^*(x) = \begin{cases} 
1 & \text{if } p(y = 1|x) \geq p(y = 0|x) \\
0 & \text{if } p(y = 1|x) < p(y = 0|x)
\end{cases}$$

NB classification rule looks like the Bayes Optimal classifier under the assumption of conditional independence we just described
Naive Bayes classification rule

For any document $x$, we want to compare $p(\text{spam}|x)$ and $p(\text{ham}|x)$

Axiom of Probability: $p(\text{spam}, x) = p(\text{spam}|x)p(x) = p(x|\text{spam})p(\text{spam})$
Naive Bayes classification rule

For any document $x$, we want to compare $p(\text{spam}|x)$ and $p(\text{ham}|x)$

Axiom of Probability: $p(\text{spam}, x) = p(\text{spam}|x)p(x) = p(x|\text{spam})p(\text{spam})$

This gives us (via bayes rule):

$$p(\text{spam}|x) = \frac{p(x|\text{spam})p(\text{spam})}{p(x)}$$
Naive Bayes classification rule

For any document $x$, we want to compare $p(\text{spam}|x)$ and $p(\text{ham}|x)$

Axiom of Probability: $p(\text{spam}, x) = p(\text{spam}|x)p(x) = p(x|\text{spam})p(\text{spam})$

This gives us (via bayes rule):

$$p(\text{spam}|x) = \frac{p(x|\text{spam})p(\text{spam})}{p(x)}$$

$$p(\text{ham}|x) = \frac{p(x|\text{ham})p(\text{ham})}{p(x)}$$
Naive Bayes classification rule

For any document $x$, we want to compare $p(\text{spam} | x)$ and $p(\text{ham} | x)$

Axiom of Probability: $p(\text{spam}, x) = p(\text{spam} | x)p(x) = p(x | \text{spam})p(\text{spam})$

This gives us (via bayes rule):

$$p(\text{spam} | x) = \frac{p(x | \text{spam})p(\text{spam})}{p(x)}$$

$$p(\text{ham} | x) = \frac{p(x | \text{ham})p(\text{ham})}{p(x)}$$

Denominators are same, and easier to compute logarithms, so we compare:

$$\log[p(x | \text{spam})p(\text{spam})] \quad \text{versus} \quad \log[p(x | \text{ham})p(\text{ham})]$$
Classifier in linear form

\[
\log[p(x|\text{spam})p(\text{spam})] = \log \left[ \prod_i p(w_i|\text{spam})^{\#w_i} p(\text{spam}) \right] = \sum_i \#w_i \log p(w_i|\text{spam}) + \log p(\text{spam})
\]
Classifier in linear form

\[
\log[p(x|\text{spam})p(\text{spam})] = \log \left[ \prod_i p(w_i|\text{spam})^{\#w_i} p(\text{spam}) \right] \\
= \sum_i \#w_i \log p(w_i|\text{spam}) + \log p(\text{spam})
\] (1)

Similarly, we have

\[
\log[p(x|\text{ham})p(\text{ham})] = \sum_i \#w_i \log p(w_i|\text{ham}) + \log p(\text{ham})
\]
Classifier in linear form

\[
\log[p(x|\text{spam})p(\text{spam})] = \log \left[ \prod_{i} p(w_i|\text{spam})^{\#w_i}p(\text{spam}) \right] \\
= \sum_{i} \#w_i \log p(w_i|\text{spam}) + \log p(\text{spam})
\]  

(1)

Similarly, we have

\[
\log[p(x|\text{ham})p(\text{ham})] = \sum_{i} \#w_i \log p(w_i|\text{ham}) + \log p(\text{ham})
\]

We’re back to the idea of comparing weighted sums of word occurrences!

\[\log p(\text{spam}) \text{ and } \log p(\text{ham}) \text{ are called “priors” (in our initial example we did not include them but they are important!)}\]
Mini-summary

**What we have shown**
By assuming a probabilistic model (i.e., Naive Bayes), we are able to derive a decision rule that is consistent with our intuition

**Our next step is to learn the parameters from data**
What are the parameters to learn?
Formal definition of Naive Bayes

**General case**

Given a random variable $X \in \mathbb{R}^D$ and a dependent variable $Y \in [C]$, the Naive Bayes model defines the joint distribution

\[
P(X = x, Y = c) = P(Y = c)P(X = x | Y = c)
\]

\[= P(Y = c) \prod_{d=1}^{D} P(X_d = x_d | Y = c)
\]

(3) and (4)
Special case (i.e., our model of spam emails)

Assumptions

- All $X_d$ are categorical variables from the same domain — $x_d \in [K]$, for example, the index to the unique words in a dictionary.

- $P(X_d = x_d | Y = c)$ depends only on the value of $x_d$, not $d$ itself, namely, order is not important (we only care about counts).
Special case (i.e., our model of spam emails)

Assumptions

- All $X_d$ are categorical variables from the same domain — $x_d \in [K]$, for example, the index to the unique words in a dictionary.
- $P(X_d = x_d | Y = c)$ depends only on the value of $x_d$, not $d$ itself, namely, order is not important (we only care about counts).

Simplified definition

$$P(X = x, Y = c) = P(Y = c) \prod_k P(k | Y = c)^{z_k} = \pi_c \prod_k \theta_{ck}^{z_k}$$

where $z_k$ is the number of times the $k$th word appears in $x$. 
Special case (i.e., our model of spam emails)

Assumptions
- All $X_d$ are categorical variables from the same domain — $x_d \in [K]$, for example, the index to the unique words in a dictionary.
- $P(X_d = x_d|Y = c)$ depends only on the value of $x_d$, not $d$ itself, namely, order is not important (we only care about counts).

Simplified definition

$$P(X = x, Y = c) = P(Y = c) \prod_k P(k|Y = c)^{z_k} = \pi_c \prod_k \theta_c^{z_k}$$

where $z_k$ is the number of times the $k$th word appears in $x$.

*Since order doesn’t matter, we enumerate over the cardinality of the domain, rather than the over the number of features $D$ (as in last slide).*
Learning problem

Training data

\[ \mathcal{D} = \{(x_n, y_n)\}_{n=1}^N \rightarrow \mathcal{D} = \{(\{z_{nk}\}_{k=1}^K, y_n)\}_{n=1}^N \]

Goal
Learning problem

Training data

\[ D = \{(x_n, y_n)\}_{n=1}^{N} \rightarrow D = \{\{z_{nk}\}_{k=1}^{K}, y_n\}_{n=1}^{N} \]

Goal

Learn \( \pi_c, c = 1, 2, \cdots, C \), and \( \theta_{ck}, \forall c \in [C], k \in [K] \) under the constraints
Learning problem

Training data

\[ \mathcal{D} = \left\{ (x_n, y_n) \right\}_{n=1}^{N} \rightarrow \mathcal{D} = \left\{ \left( \left\{ z_{nk} \right\}_{k=1}^{K}, y_n \right) \right\}_{n=1}^{N} \]

Goal

Learn \( \pi_c, c = 1, 2, \ldots, C, \) and \( \theta_{ck}, \forall c \in [C], k \in [K] \) under the constraints

\[ \sum_{c} \pi_{c} = 1 \]

and

\[ \sum_{k} \theta_{ck} = \sum_{k} P(k|Y = c) = 1 \]

as well as those quantities should be nonnegative.
Our hammer: maximum likelihood estimation

Recall our joint probability

\[ P(X = x, Y = c) = \pi_c \prod_k \theta_{ck}^{z_k} \]

where \( z_k \) is the number of times the \( k \)th word appears in \( x \).
Our hammer: maximum likelihood estimation

Recall our joint probability

\[
P(X = x, Y = c) = \pi_c \prod_k \theta_{ck}^{z_k}
\]

where \( z_k \) is the number of times the \( k \)th word appears in \( x \).

Likelihood of the training data

\[
\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N \rightarrow \mathcal{D} = \{(\{z_{nk}\}_{k=1}^K, y_n)\}_{n=1}^N
\]

\[
L = P(\mathcal{D}) = \prod_{n=1}^N \pi_{y_n} P(x_n | y_n)
\]
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[ \mathcal{L} = \log P(D) = \log \prod_{n=1}^{N} \pi_{y_n} P(x_n | y_n) \]
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[ \mathcal{L} = \log P(D) = \log \prod_{n=1}^{N} \pi_{yn} P(x_n|y_n) \]

\[ = \log \prod_{n=1}^{N} \left( \pi_{yn} \prod_{k} \theta_{ynk}^{z_{nk}} \right) \]
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[
\mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{y_n} P(x_n|y_n)
\]

\[
= \log \prod_{n=1}^{N} \left( \pi_{y_n} \prod_{k} \theta_{y_n k}^{z_{nk}} \right)
\]

\[
= \sum_{n} \left( \log \pi_{y_n} + \sum_{k} z_{nk} \log \theta_{y_n k} \right)
\]

Optimize it!

\[(\pi^*, \theta^*) = \arg \max \sum_{n} \left( \log \pi_{y_n} + \sum_{k} z_{nk} \log \theta_{y_n k} \right)\]
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[ \mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{y_n} P(x_n|y_n) \]

\[ = \log \prod_{n=1}^{N} \left( \pi_{y_n} \prod_{k} \theta_{y_n k}^{z_{nk}} \right) \]

\[ = \sum_{n} \left( \log \pi_{y_n} + \sum_{k} z_{nk} \log \theta_{y_n k} \right) \]

\[ = \sum_{n} \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k} \]
Our hammer: maximum likelihood estimation

Log-Likelihood of the training data

\[
\mathcal{L} = \log P(D) = \log \prod_{n=1}^{N} \pi_{yn} P(x_n | y_n)
\]

\[
= \log \prod_{n=1}^{N} \left( \pi_{yn} \prod_{k} \theta_{ynk}^{z_{nk}} \right)
\]

\[
= \sum_{n} \left( \log \pi_{yn} + \sum_{k} z_{nk} \log \theta_{ynk} \right)
\]

Optimize it!

\[
(\pi^*_c, \theta^*_ck) = \arg \max \sum_{n} \log \pi_{yn} + \sum_{n,k} z_{nk} \log \theta_{ynk}
\]
Details

Note the separation of parameters in the likelihood

\[ \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k} \]
Details

Note the separation of parameters in the likelihood

\[ \sum_{n} \log \pi_{y_{n}} + \sum_{n,k} z_{nk} \log \theta_{y_{n,k}} \]

this implies that \( \{\pi_{c}\} \) and \( \{\theta_{ck}\} \) can be estimated separately
Note the separation of parameters in the likelihood

\[
\sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_{nk}}
\]

this implies that \(\{\pi_c\}\) and \(\{\theta_{ck}\}\) can be estimated separately

Reorganize terms

\[
\sum_n \log \pi_{y_n} = \sum_c \log \pi_c \times (#\text{of data points labeled as } c)
\]
Details

Note the separation of parameters in the likelihood

\[ \sum_{n} \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k} \]

this implies that \( \{\pi_c\} \) and \( \{\theta_{ck}\} \) can be estimated separately

Reorganize terms

\[ \sum_{n} \log \pi_{y_n} = \sum_{c} \log \pi_{c} \times (\text{# of data points labeled as } c) \]

and

\[ \sum_{n,k} z_{nk} \log \theta_{y_n k} = \sum_{c} \sum_{n: y_n = c} \sum_{k} z_{nk} \log \theta_{ck} \]
Details

Note the separation of parameters in the likelihood

\[
\sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k}
\]

this implies that \( \{\pi_c\} \) and \( \{\theta_{ck}\} \) can be estimated separately

Reorganize terms

\[
\sum_n \log \pi_{y_n} = \sum_c \log \pi_c \times (\text{# of data points labeled as } c)
\]

and

\[
\sum_{n,k} z_{nk} \log \theta_{y_n k} = \sum_c \sum_{n:y_n=c} \sum_k z_{nk} \log \theta_{ck} = \sum_c \sum_{n:y_n=c,k} z_{nk} \log \theta_{ck}
\]
Details

Note the separation of parameters in the likelihood

$$\sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k}$$

this implies that $\{\pi_c\}$ and $\{\theta_{ck}\}$ can be estimated separately

Reorganize terms

$$\sum_n \log \pi_{y_n} = \sum_c \log \pi_c \times (#\text{of data points labeled as } c)$$

and

$$\sum_{n,k} z_{nk} \log \theta_{y_n k} = \sum_c \sum_{n:y_n=c} \sum_k z_{nk} \log \theta_{ck} = \sum_c \sum_{n:y_n=c,k} z_{nk} \log \theta_{ck}$$

The latter implies $\{\theta_{ck}\}$ and $\{\theta_{c'k}\}$ for $c \neq c'$ can be estimated independently!
Estimating \( \{ \pi_c \} \)

We want to maximize

\[
\sum_c \log \pi_c \times (\text{# of data points labeled as } c)
\]
Estimating $\{\pi_c\}$

We want to maximize

$$\sum_c \log \pi_c \times (\text{# of data points labeled as } c)$$

**Intuition**

- Similar to roll a dice (or flip a coin): each side of the dice shows up with a probability of $\pi_c$ (total $C$ sides)
- And we have total $N$ trials of rolling this dice
Estimating $\{\pi_c\}$

We want to maximize

$$
\sum_c \log \pi_c \times (\# \text{of data points labeled as } c)
$$

Intuition
- Similar to roll a dice (or flip a coin): each side of the dice shows up with a probability of $\pi_c$ (total $C$ sides)
- And we have total $N$ trials of rolling this dice

Solution

$$
\pi^*_c = \frac{\# \text{of data points labeled as } c}{N}
$$
Estimating $\{\theta_{ck}, k = 1, 2, \cdots, K\}$

We want to maximize

$$\sum_{n: y_n = c, k} z_{nk} \log \theta_{ck}$$
Estimating \( \{\theta_{ck}, k = 1, 2, \cdots, K\} \)

We want to maximize

\[
\sum_{n: y_n = c, k} z_{nk} \log \theta_{ck}
\]

Intuition

- Again similar to roll a dice: each side of the dice shows up with a probability of \( \theta_{ck} \) (total K sides)
- And we have total \( \sum_{n: y_n = c, k} z_{nk} \) trials.
Estimating \( \{\theta_{ck}, k = 1, 2, \cdots, K\} \)

We want to maximize

\[
\sum_{n : y_n = c, k} z_{nk} \log \theta_{ck}
\]

Intuition

- Again similar to roll a dice: each side of the dice shows up with a probability of \( \theta_{ck} \) (total K sides)
- And we have total \( \sum_{n : y_n = c, k} z_{nk} \) trials.

Solution

\[
\theta^*_{ck} = \frac{\# \text{of times side } k \text{ shows up in data points labeled as } c}{\# \text{total trials for data points labeled as } c}
\]
Translating back to our problem of detecting spam emails

- Collect a lot of ham and spam emails as training examples
- Estimate the “prior”

\[ p(\text{ham}) = \]

\[ p(\text{spam}) = \]
Translating back to our problem of detecting spam emails

- Collect a lot of ham and spam emails as training examples
- Estimate the “prior”
  \[
  p(\text{ham}) = \frac{\text{# of ham emails}}{\text{# of emails}}, \quad p(\text{spam}) = \frac{\text{# of spam emails}}{\text{# of emails}}
  \]
- Estimate the weights (i.e., \( p(\text{dollar}|\text{ham}) \) etc)
Translating back to our problem of detecting spam emails

- Collect a lot of ham and spam emails as training examples
- Estimate the “prior”

\[
p(\text{ham}) = \frac{\# \text{ of ham emails}}{\# \text{ of emails}}, \quad p(\text{spam}) = \frac{\# \text{ of spam emails}}{\# \text{ of emails}}
\]

- Estimate the weights (i.e., \( p(\text{dollar}|\text{ham}) \) etc)

\[
p(\text{funny\_word}|\text{ham}) = \frac{\# \text{ of funny\_word in ham emails}}{\# \text{ of words in ham emails}} \quad (5)
\]

\[
p(\text{funny\_word}|\text{spam}) = \frac{\# \text{ of funny\_word in spam emails}}{\# \text{ of words in spam emails}} \quad (6)
\]
Classification rule

**Given an unlabeled point** \( x = \{ z_k, k = 1, 2, \ldots, K \} \), how to label it?
Classification rule

Given an unlabeled point \( x = \{z_k, k = 1, 2, \cdots, K\} \), how to label it?

\[
y^* = \arg \max_{c \in [C]} P(y = c | x) \\
= \arg \max_{c \in [C]} P(y = c) P(x | y = c) \\
= \arg \max_c [\log \pi_c + \sum_k z_k \log \theta_{ck}] 
\]  

(7)  

(8)  

(9)
Constrained optimization

Equality Constraints

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad g(x) = 0
\end{align*}
\]

Method of Lagrange multipliers

Construct the following function (Lagrangian)

\[
L(x, \lambda) = f(x) + \lambda g(x)
\]
A short derivation of the maximum likelihood estimation

To maximize

$$\sum_{n: y_n = c, k} z_{nk} \log \theta_{ck}$$

We can use the method of Lagrangian multipliers!
A short derivation of the maximum likelihood estimation

To maximize

\[ \sum_{n: y_n = c, k} z_{nk} \log \theta_{ck} \]

We can use the method of Lagrangian multipliers!

\[ f(x) = - \sum_{n: y_n = c, k} z_{nk} \log \theta_{ck} \]

\[ g(x) = \sum_k \theta_{ck} - 1 = 0 \]

*Chris will review in section on Friday*
You should know or be able to

- What naive Bayes model is
  - write down the joint distribution
  - explain the conditional independence assumption implied by the model
  - explain how this model can be used to classify spam vs ham emails

- Be able to go through the short derivation for parameter estimation
  - The model illustrated here is called discrete/multinomial Naive Bayes
  - HW2 asks you to apply the same principle to Gaussian naive Bayes
  - The derivation is very similar – except there you need to estimate Gaussian continuous random variables (instead of estimating discrete random variables like rolling a dice)
Moving forward

Examine the classification rule for naive Bayes

\[
y^* = \arg \max_c \log \pi_c + \sum_k z_k \log \theta_{ck}
\]

For binary classification, we thus determine the label based on the sign of

\[
\log \pi_1 + \sum_k z_k \log \theta_{1k} - \left( \log \pi_2 + \sum_k z_k \log \theta_{2k} \right)
\]

This is just a linear function of the features \( \{z_k\} \)

\[
w_0 + \sum_k z_k w_k
\]

where we “absorb” \( w_0 = \log \pi_1 - \log \pi_2 \) and \( w_k = \log \theta_{1k} - \log \theta_{2k} \).
Naive Bayes is a linear classifier

Fundamentally, what really matters in deciding decision boundary is

\[ w_0 + \sum_k z_k w_k \]

This motivates many new methods. One of them is logistic regression, to be discussed in next lecture.