Natural Language Processing

Machine learning for Language Understanding

Tien-Lam Pham

PIAS
VIỆN NGHIÊN CỨU
TIÊN TIẾN PHENIKAA

PHENIKAA
UNIVERSITY

Contents

- Text classification problem
- Machine learning
- Naive Bayes
- Logistic Regression
- Neural network

Examples

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

Examples

	label	message
0	ham	Go until jurong point, crazy.. Available only …
1	ham	Ok lar… Joking wif u oni…
2	spam	Free entry in 2 a wkly comp to win FA Cup fina…
3	ham	U dun say so early hor… U c already then say…
4	ham	Nah I don't think he goes to usf, he lives aro…

Examples

What is the subject of this medical article?

MEDLINE Article

Syntactic frame and verb bias in aphasia: Plausibility judgments of undergoer-subject sentences

?

MeSH Subject Category Hierarchy

Antogonists and Inhibitors

Blood Supply

Chemistry

Drug Therapy

Embryology

Epidemiology

…

Examples

Positive or negative movie review?

\+ *...zany characters and richly applied satire, and some grea plot twists*

_ *It was pathetic. The worst part about it was the boxing scenes...*

\+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*

_ *...awful pizza and ridiculously overpriced...*

Examples

\+ *...zany characters and **richly** applied satire, and some **great** plot twists*

_ *It was **pathetic**. The **worst** part about it was the boxing scenes...*

\+ *...**awesome** caramel sauce and sweet toasty almonds. I **love** this place!*

_ *...**awful** pizza and **ridiculously** overpriced...*

Examples

Movie: is this review positive or negative?

Products: what do people think about the new iPhone?

Public sentiment: how is consumer confidence?

Politics: what do people think about this candidate or issue?

Prediction: predict election outcomes or market trends from sentiment

Text Classification

Input:

- a document *d*
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $c \in C$

Intelligent Machine

Give machine set of rules

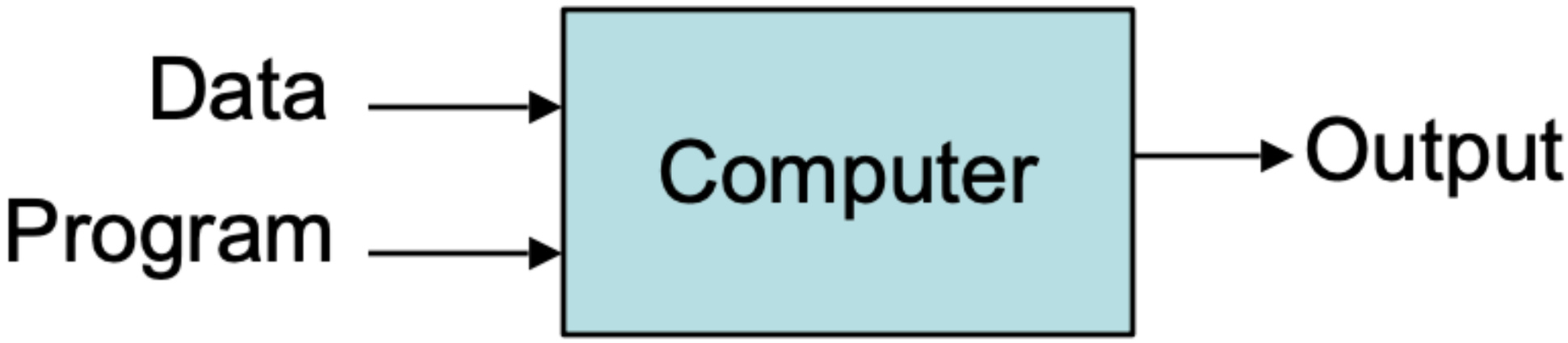
Learning from data
(Machine learning)

INPUT
Information
(X)

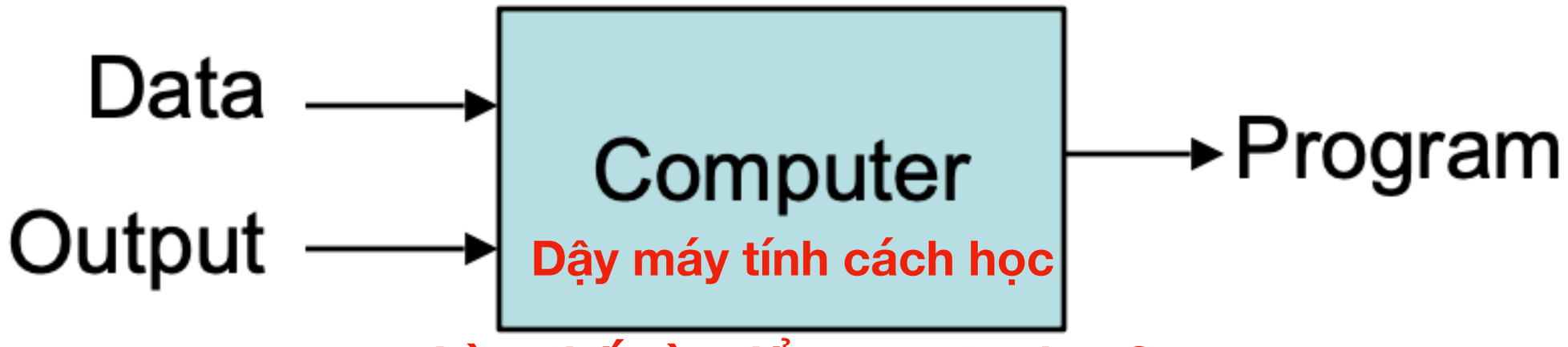
OUTPUT
Prediction (y)
Action (0, 1)

Data processing

Traditional Programming



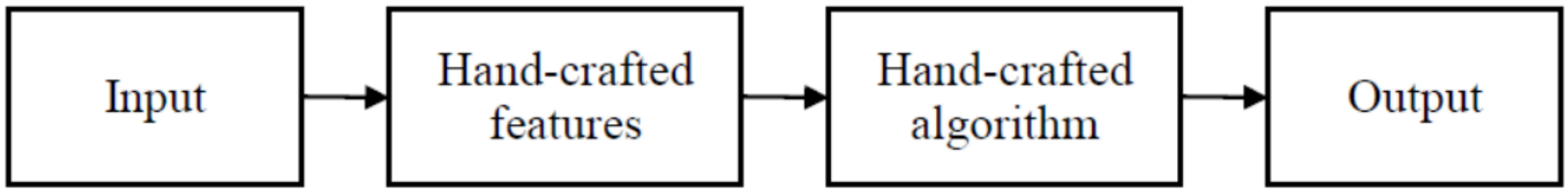
Machine Learning



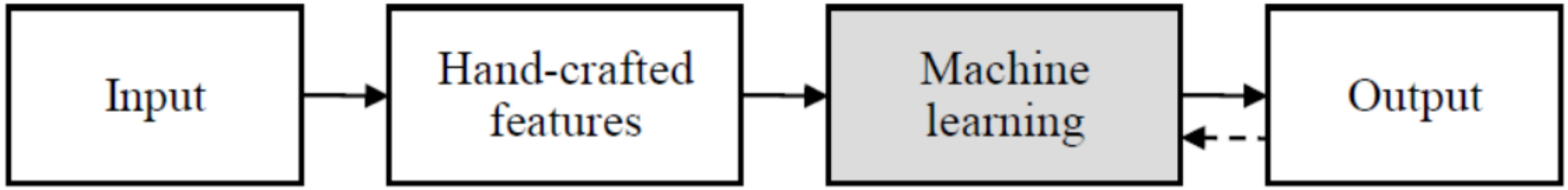
Dậy máy tính cách học

Làm thế nào để computer học?

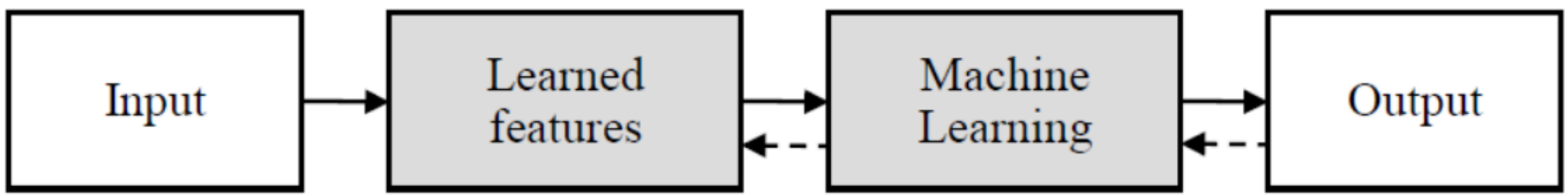
Traditional ML vs DL



(a) Traditional



(b) Classic machine learning pipeline



(c) Deep learning pipeline

Rule-based Classification: Hand-code rules

Rules based on combinations of words or other features

- spam: black-list-address OR (“dollars” AND “you have been selected”)

Accuracy can be high

- If rules carefully refined by expert

But building and maintaining these rules is expensive

ML-based Classification: Learn for Rules

Input:

- a document *d*
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- A training set of *m* hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

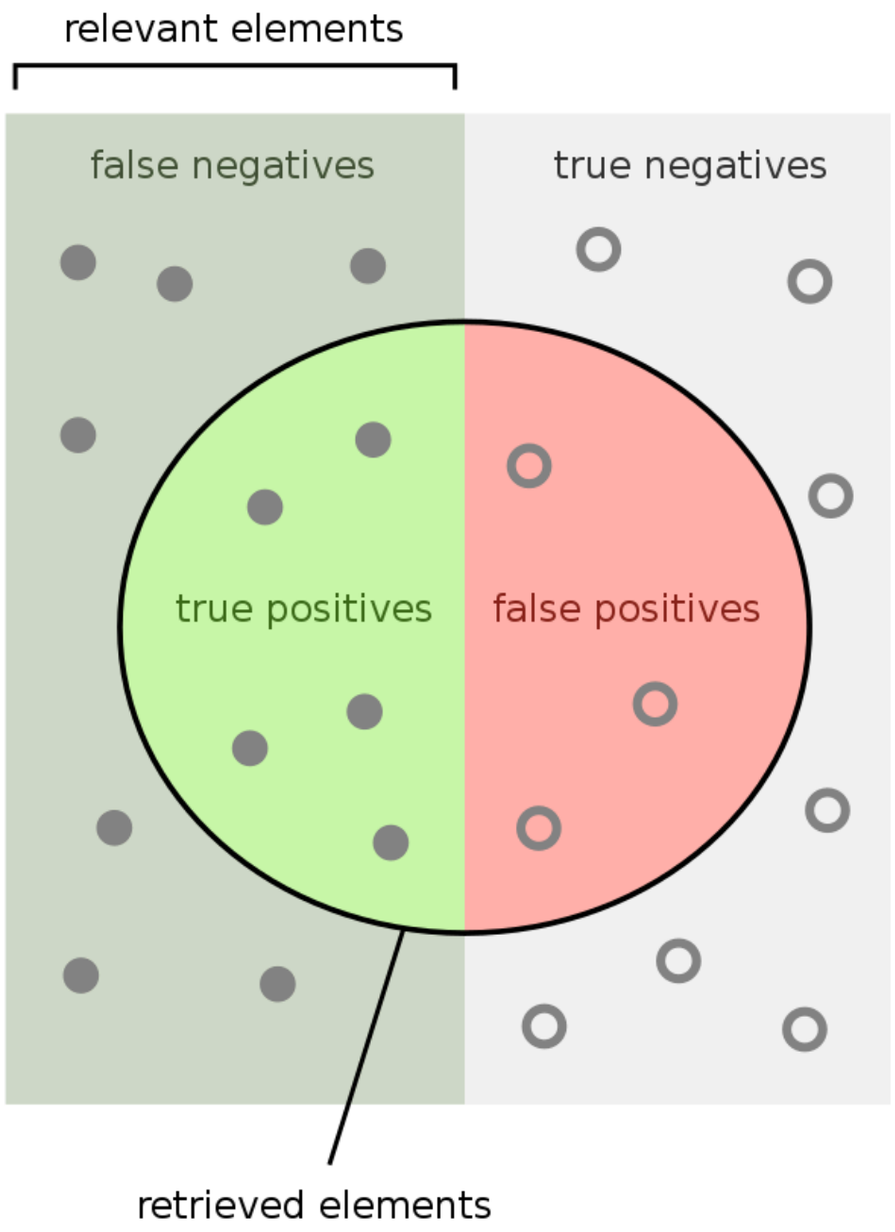
Output:

- a learned classifier $\gamma:d \rightarrow c$

Any kind of classifier

- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors
- …

Classification Evaluation



How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{retrieved elements}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Naive Bayes

Simple ("naive") classification method based on Bayes rule

Relies on very simple representation of document

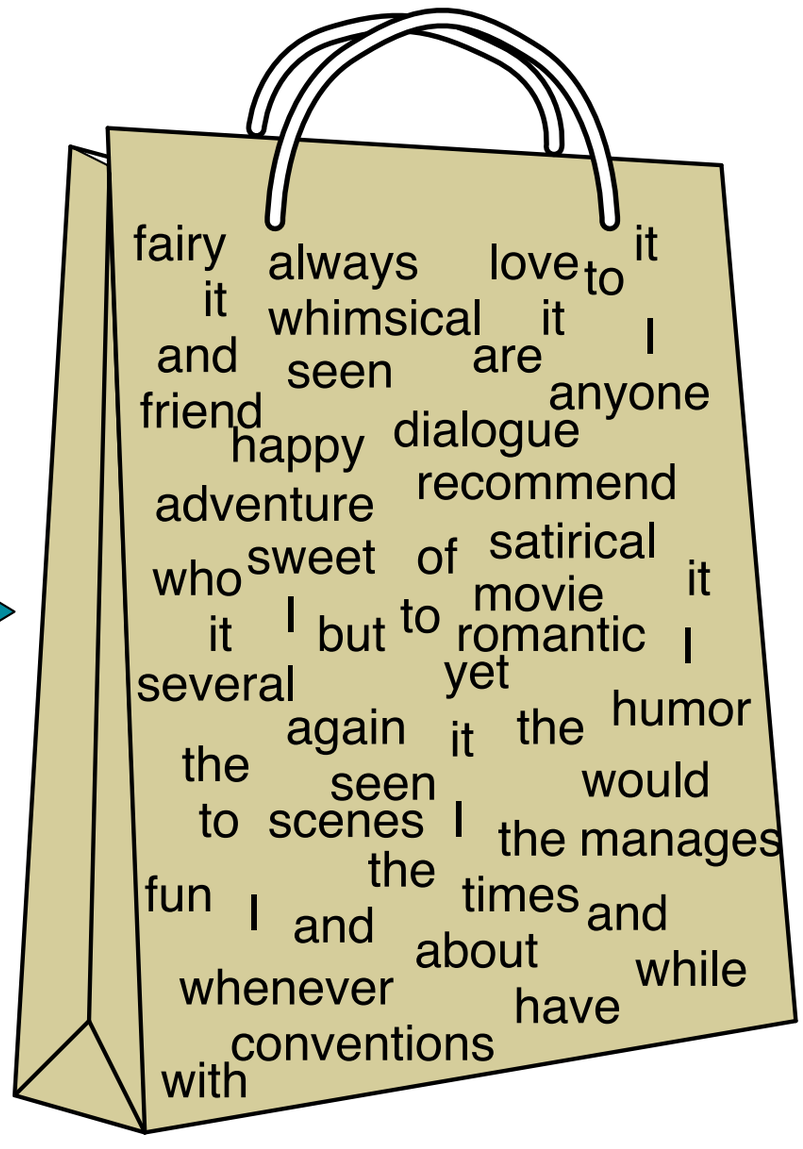
- **Bag of words**

For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Bag of Words

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!



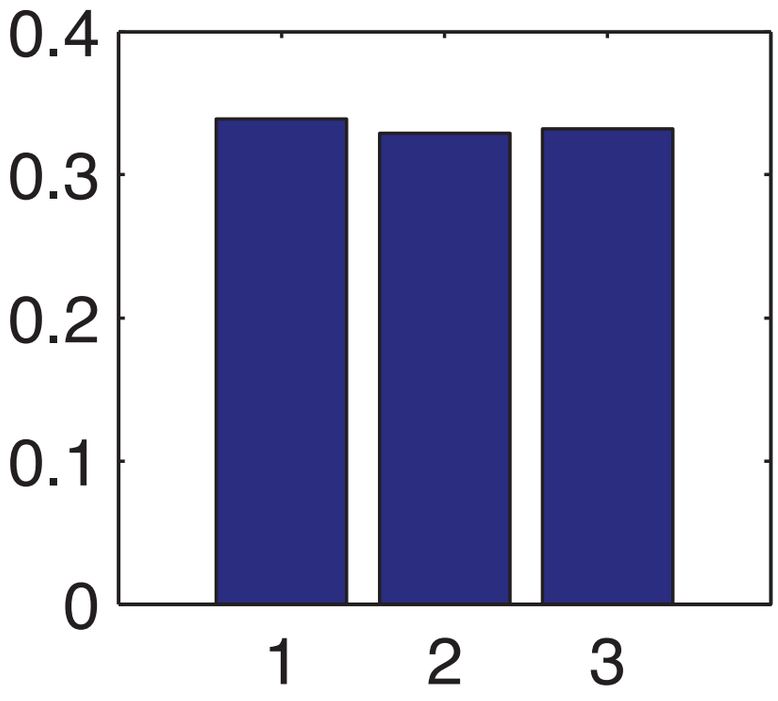
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

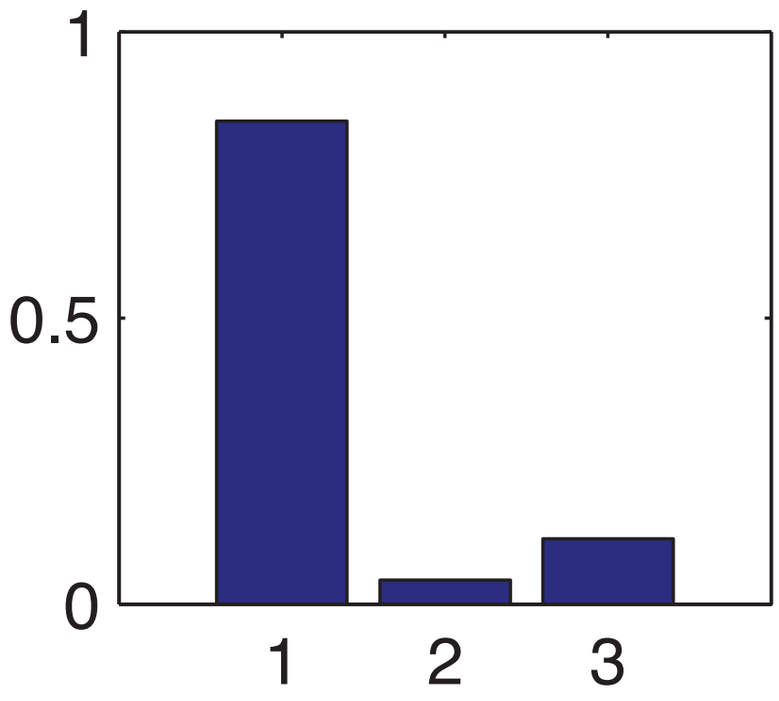
Bayesian Inference

- Classification

$$\hat{y} = arg\max_{c} P(y = c \mid x)$$

$$P(y = c \mid x) = \frac{P(x \mid y = c)P(y = c)}{P(x)}$$





Bayesian Learning

- Prior: probability a hypothesis is True
- Likelihood: probability that a data is observed given a hypothesis
- Posterior: probability a hypothesis is True given a data

$$\hat{h} = arg\max_{h} P(h \mid D)$$

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

Bayesian Learning

- Prior: probability a hypothesis is True
- Likelihood: probability that a data is observed given a hypothesis
- Posterior: probability a hypothesis is True given a data

$$\hat{\theta} = arg\max_{\theta} P(\theta \mid D)$$

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

Document Classification

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Document Classification

"Likelihood" "Prior"

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Document Classification

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

O(|X|^n•|C|) parameters

How often does this class occur?

Could only be estimated if a very, very large number of training examples was available.

We can just count the relative frequencies in a corpus

Naive Bayes Independence Assumption

$$P(x_1, x_2, \ldots, x_n \mid c)$$

Bag of Words assumption: Assume position doesn't matter

Conditional Independence: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

Naive Bayes Independence Assumption

positions ← all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naive Bayes Independence Assumption

There's a problem with this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because $\log(ab) = \log(a) + \log(b)$

We'll sum logs of probabilities instead of multiplying probabilities!

Naive Bayes Independence Assumption

Instead of this: $$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

This: $$c_{\mathrm{NB}} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \mathrm{positions}} \log P(x_i | c_j) \right]$$

Notes:

1) Taking log doesn't change the ranking of classes!
   The class with highest probability also has highest log probability!
2) It's a linear model:
   Just a max of a sum of weights: a **linear** function of the inputs
   So naive bayes is a **linear classifier**

Naive Bayes Learning

- simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

Naive Bayes Learning

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of topic c_j

Create mega-document for topic *j* by concatenating all docs in this topic

- Use frequency of *w* in mega-document

Naive Bayes Learning

What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Naive Bayes Learning

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Naive Bayes Learning

- From training corpus, extract *Vocabulary*

Calculate $P(c_j)$ terms

- For each c_j in C do

$docs_j \leftarrow$ all docs with class $=c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Unknown Words

What about unknown words

- that appear in our test data
- but not in our training data or vocabulary?

We **ignore** them

- Remove them from the test document!
- Pretend they weren't there!
- Don't include any probability for them at all!

Why don't we build an unknown word model?

- It doesn't help: knowing which class has more unknown words is not generally helpful!

Stop Words

Some systems ignore stop words

- **Stop words:** very frequent words like *the* and *a*.
  - Sort the vocabulary by word frequency in training set
  - Call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from both training and test sets
    - As if they were never there!

But removing stop words doesn't usually help

- So in practice most NB algorithms use **all** words and **don't** use stopword lists

Activity

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

Activity

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

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5

P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$