ELEG 6913: Machine Learning for Big Data

Fall 2016

Lecture 9: Text Classification

Dr. Xishuang Dong

Outline

- **Text Classification**
- **Logistic Regression**
- **Naive Bayes Classifier**
- **Evaluation Metrics**
- **Summary**

(Acknowledgment: some parts of the slides are from Paul Bennett, Dan Jurafsky, and various other sources. The copyright of those parts belongs to their original owners.)

Outline

- **Text Classification**
- **Logistic Regression**
- **Naive Bayes Classifier**
- **Evaluation Metrics**
- **Summary**

Text Classification

- **Task of assigning predefined categories to free-text documents.**

Input:

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

Output: a predicted class $c \in C$

Cost of Manual Text Categorization

- **Yahoo!**
  - ✓ **200 (?) people for manual labeling of Web pages**
  - ✓ **using a hierarchy of 500,000 categories**
- **MEDLINE (National Library of Medicine)**
  - ✓ **$2 million/year for manual indexing of journal articles**
  - ✓ **using MEdical Subject Headings (18,000 categories)**

Cost of Manual Text Categorization

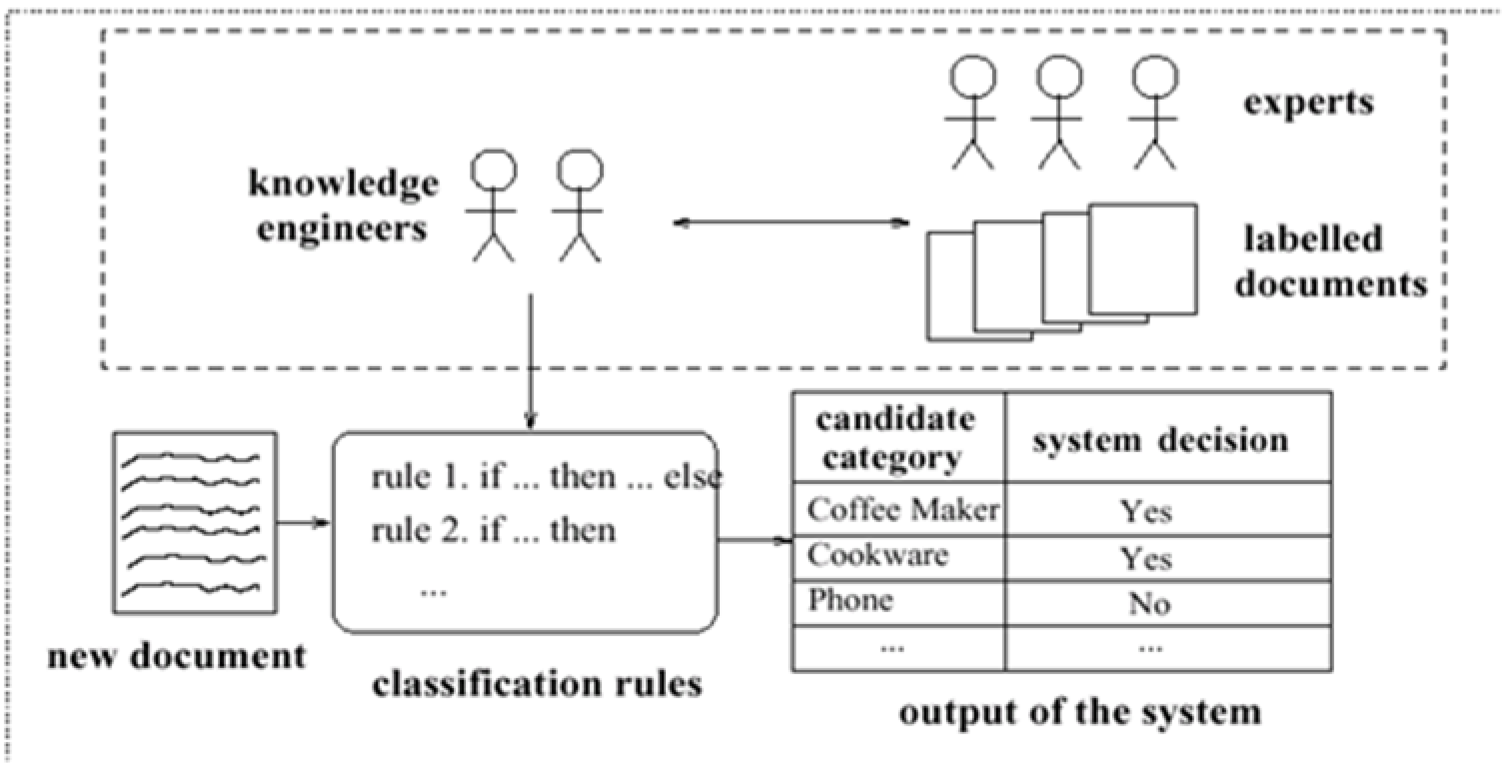
- Mayo Clinic
  - ✓ $1.4 million annually for coding patient-record events
  - ✓ using the International Classification of Diseases (ICD) for billing insurance companies
- US Census Bureau decennial census (1990: 22 million responses)
  - ✓ 232 industry categories and 504 occupation categories
  - ✓ $15 million if fully done by hand

Rule-based Approach to TC

- **Text in a Web Page**
  "Saeco revolutionized *espresso* brewing a decade ago by introducing Saeco SuperAutomatic *machines*, which go from bean to *coffee* at the touch of a button. The all-new Saeco Vienna Super-Automatic home coffee and *cappucino machine* combines top quality with low price!"
- **Rules**
  - ✓ **Rule 1.**
    (*espresso* or *coffee* or *cappucino*) and *machine ⟹ Coffee Maker**
  - ✓ **Rule 2.**
    ***automat** and *answering* and *machine** ⟹ Phone**
  - ✓ **Rule ...**

Expert System for TC (late 1980s)

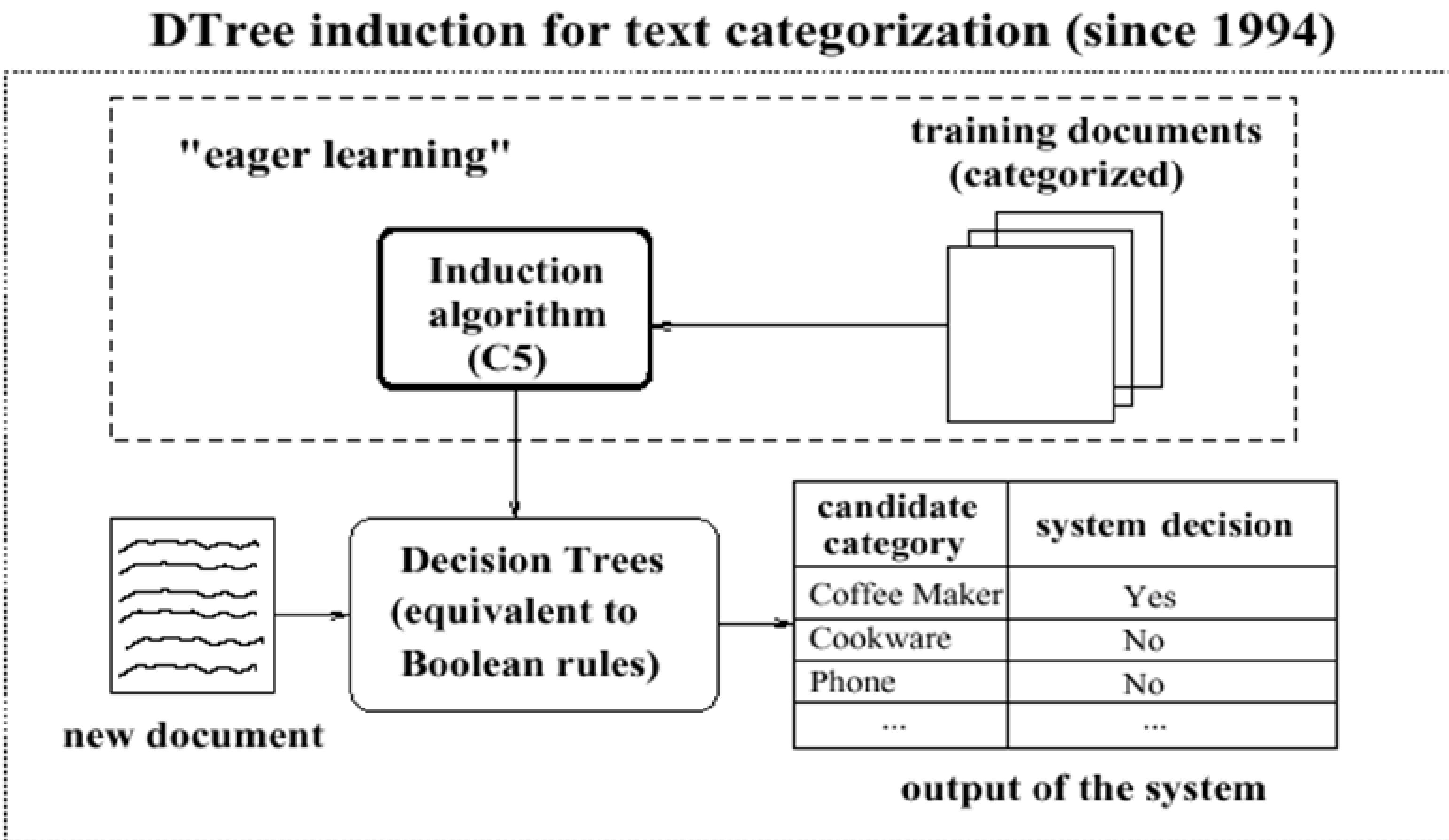
Expert system for text categorization (late 1980s)



Defining Rules By Hand

- **Experience has shown**
  - ✓ **too time consuming**
  - ✓ **too difficult**
  - ✓ **inconsistency issues (as the rule set gets large)**

Replace Knowledge Engineering with a Statistical Learner



Knowledge Engineering VS. Machine Learning

- **For US Census Bureau Decennial Census 1990**
  - ✓ **232 industry categories and 504 occupation categories**
  - ✓ **$15 million if fully done by hand**
- **Define classification rules manually:**
  - ✓ **Expert System AIOCS**
  - ✓ **Development time: 192 person-months (2 people, 8 years)**
  - ✓ **Accuracy = 47%**
- **Learn classification function**
  - ✓ **Nearest Neighbor classification (Creecy '92: 1-NN)**
  - ✓ **Development time: 4 person-months (Thinking Machine)**
  - ✓ **Accuracy = 60%**

Text Classification Based on Machine Learning

- **Supervised Machine Learning**
  - ✓ **Neural Network**
  - ✓ **SVM**
  - ✓ **…**
- **Labels: Text Category**

Text Classification Based on Machine Learning

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 $(d_1,c_1),\ldots.,(d_m,c_m)$

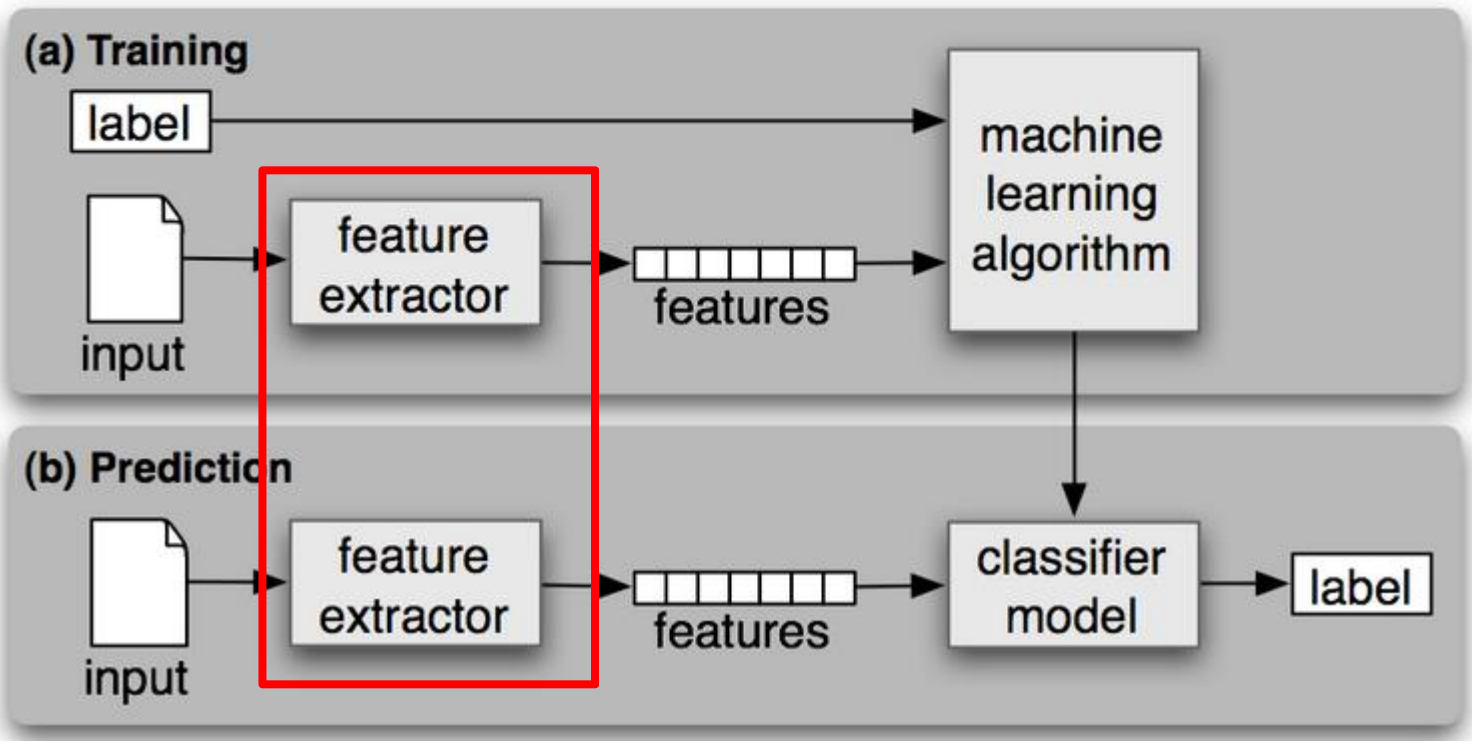
Output:

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

Applications of Text Classification

- **Information Retrieval**
- **Question Answer (Q & A)**
- **Recommendation System**
- **Stock Prediction**
- **Suicide Forecasting**
- **… …**

Supervised Machine Learning

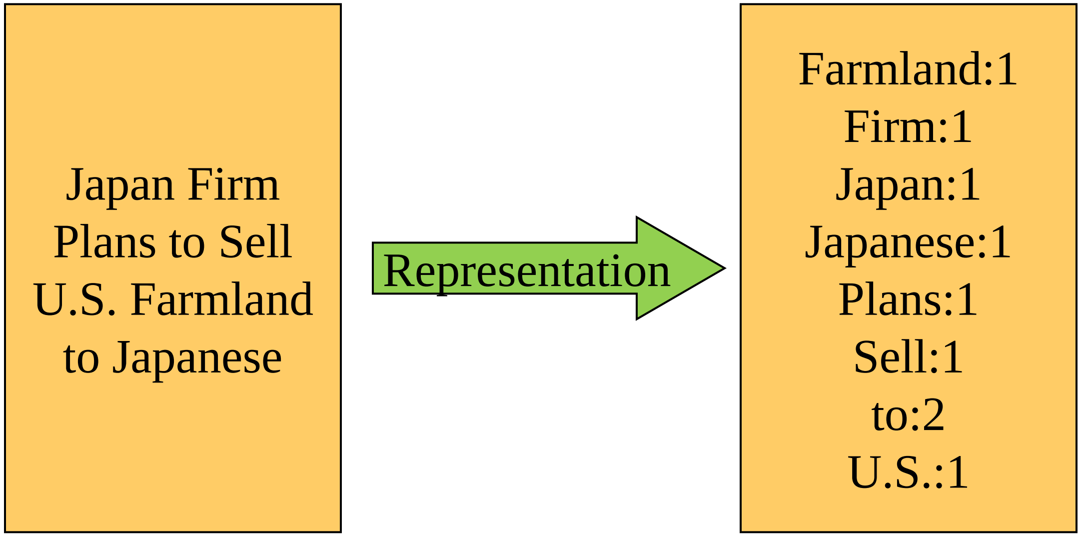


Representing Texts

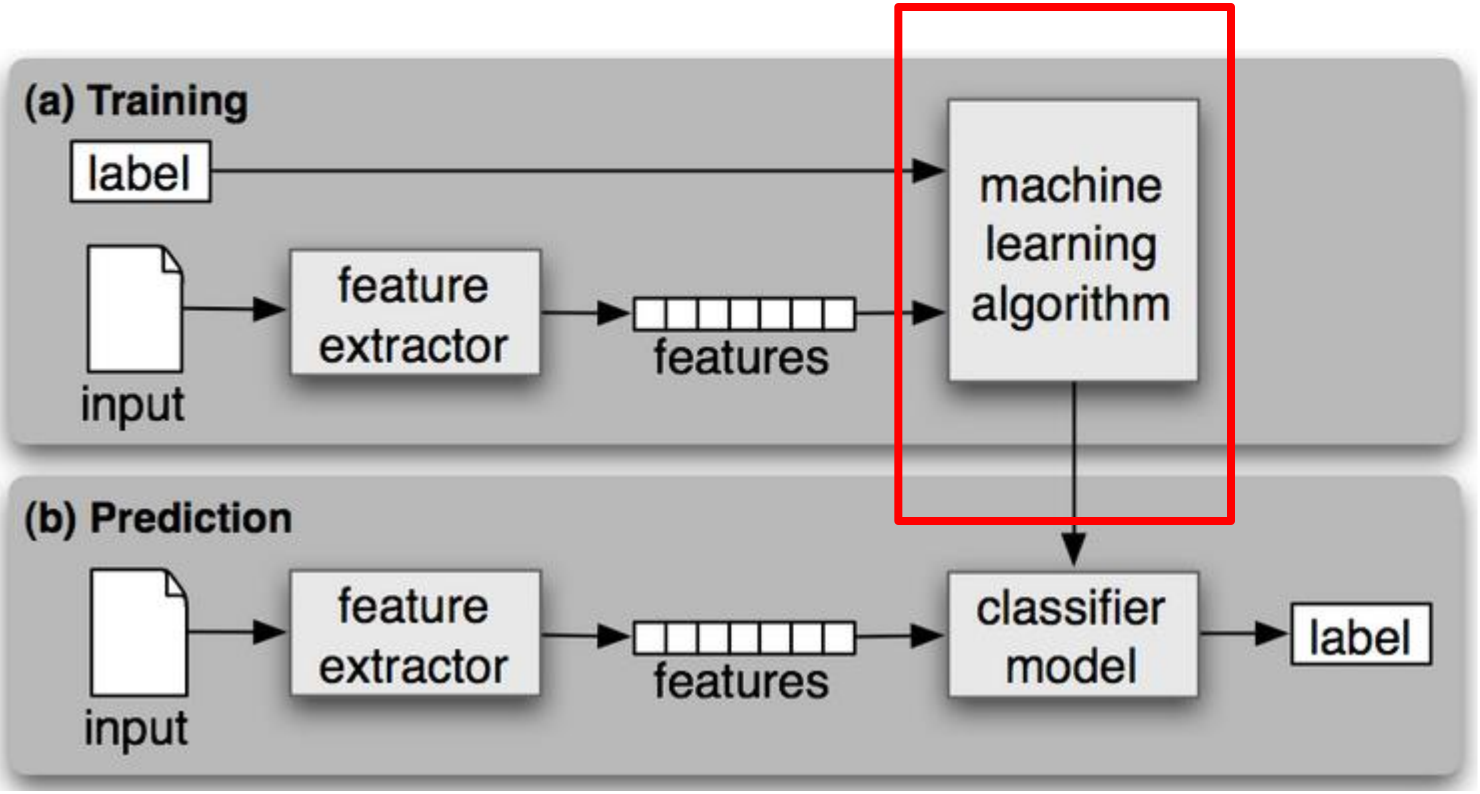
- Usually, an example is represented as a series of feature-value pairs.
- The features can be arbitrarily abstract (as long as they are easily computable) or very simple.

Representing Texts

- **For example, the features could be the set of all words and the values, their number of occurrences in a particular document.**



Supervised Machine Learning



Three Main Approaches

- Learn a classifier: a function f, $\hat{y} = f(\mathbf{x})$
- Learn a probabilistic ***discriminative*** model, i.e., the conditional distribution $P(y \mid \mathbf{x})$
- Learn a probabilistic ***generative*** model, i.e., the joint probability distribution: $P(\mathbf{x}, y)$
- Examples:
  - Learn a classifier: Perceptron, LDA (projection with threshold view)
  - Learn a conditional distribution: ***Logistic regression***
  - Learn the joint distribution: a probabilistic view of Linear Discriminant Analysis (LDA)

Outline

- Text Classification
- **Logistic Regression**
- Naive Bayes Classifier
- Evaluation Metrics
- Summary

Notation Shift

- $S=\{(\mathbf{x}^i, y^i): i=1,\ldots, N\}$ --- superscript for example index. N is the total number of examples
- Subscript for element index within a vector, i.e., x^i_j represents the jth element of the ith training example
- Class labels are 0 and 1 (not +1 and -1)

Logistic Regression

- Given training set D, logistic regression learns the conditional distribution P(*y* | **x**)
- We will assume only two classes *y* = 0 and *y* = 1 and a parametric form for P(y = 1 | x, w) , **and** w is the parameter vector

$$p(y=1 \mid \mathbf{x};\mathbf{w}) = p_1(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$$

$$p(y=0 \mid \mathbf{x};\mathbf{w}) = 1 - p_1(\mathbf{x})$$

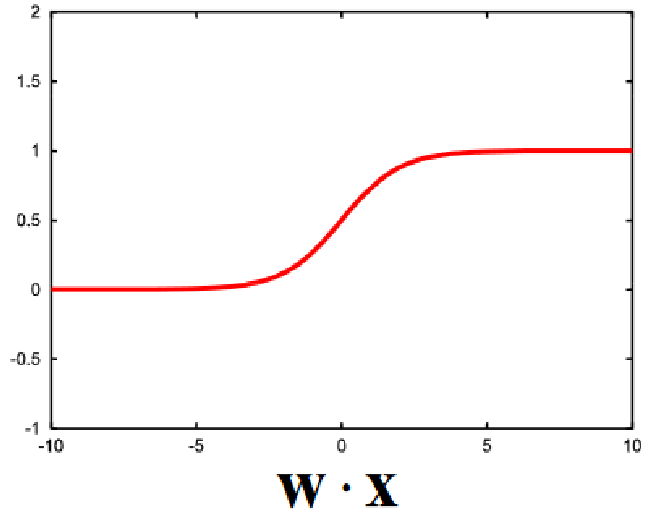
- It is easy to show that this is equivalent to

$$\log\frac{p(y=1 \mid \mathbf{x};\mathbf{w})}{p(y=0 \mid \mathbf{x};\mathbf{w})} = \mathbf{w}\cdot\mathbf{x}$$

- i.e. the ***log odds*** of class 1 is a linear function of **x**.

Why the Logistic (Sigmoid) Function

$$g(\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x})}$$



A linear function has a range from $[-\infty,\infty]$, the logistic function transforms the range to [0,1] to be a probability.

Logistic Regression Yields Linear Classifier

- Recall that given P(y | x) we predict ŷ = 1 if the expected loss of predicting 0 is greater than predicting 1 (for now assume L(0,1) = L(1,0))

$$E_{y|x}[L(0,y)] > E_{y|x}[L(1,y)] \Leftrightarrow$$

$$\sum_y P(y \mid \mathbf{x}) L(0,y) > \sum_y P(y \mid \mathbf{x}) L(1,y) \Leftrightarrow$$

$$P(y=0 \mid \mathbf{x}) L_{00} + P(y=1 \mid \mathbf{x}) L_{01} > P(y=0 \mid \mathbf{x}) L_{10} + P(y=1 \mid \mathbf{x}) L_{11} \Leftrightarrow$$

$$P(y=1 \mid \mathbf{x}) > P(y=0 \mid \mathbf{x}) \Leftrightarrow$$

$$\frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} > 1 \Leftrightarrow \log \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} > 0 \Leftrightarrow$$

$$\mathbf{w} \cdot \mathbf{x} > 0$$

- This assumed L(0,1)=L(1,0)
- A similar derivation can be done for arbitrary L(0,1) and L(1,0).

Maximum Likelihood Learning

- Recall that the likelihood function is the probability of the data **D** given the parameters – p(**D**|**w**)
- It is a function of the parameters
- Maximum likelihood learning finds the parameters that maximize this likelihood function
- A common trick is to work with log-likelihood, i.e., take the logarithm of the likelihood function – log p(**D**|**w**)

Computing the Likelihood

- In our framework, we assume each training example ($\mathbf{x}^i$, y^i) is drawn independently from the same (but unknown) distribution $P(\mathbf{x}, y)$ (the famous **i.i.d** assumption), hence we can write

$$\log P(D \mid \mathbf{w}) = \log \prod_i P(\mathbf{x}^i, y^i \mid \mathbf{w}) = \sum_i \log P(\mathbf{x}^i, y^i \mid \mathbf{w})$$

- Joint distribution P(a,b) can be factored as P(a | b)P(b)

$$\arg\max_{w} \log P(D \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{x}^i, y^i \mid \mathbf{w})$$
$$= \arg\max_{\mathbf{w}} \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w}) P(\mathbf{x}^i \mid \mathbf{w})$$

- Further, P(**x** | **w**) = P(**x**) because it does not depend on w, so:

$$\arg\max_{\mathbf{w}} \log P(D \mid \mathbf{w}) = \arg\max_{\mathbf{w}} \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w})$$

Fitting Logistic Regression by Gradient Ascent

$$L(\mathbf{w}) = \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w}) = \sum_i [y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)]$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \frac{\partial \log P(y^i \mid \mathbf{x}^i, \mathbf{w})}{\partial w_j} = \frac{\partial}{\partial w_j}[y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)]$$

$$= \frac{y^i}{\hat{y}^i}\left(\frac{\partial \hat{y}^i}{\partial w_j}\right) + \frac{1-y^i}{1-\hat{y}^i}\left(-\frac{\partial \hat{y}^i}{\partial w_j}\right) = \left[\frac{y^i}{\hat{y}^i} - \frac{1-y^i}{1-\hat{y}^i}\right]\frac{\partial \hat{y}^i}{\partial w_j}$$

$$= \left[\frac{y^i - y^i\hat{y}^i - \hat{y}^i + y^i\hat{y}^i}{\hat{y}^i(1-\hat{y}^i)}\right]\frac{\partial \hat{y}^i}{\partial w_j} = \left[\frac{y^i - \hat{y}^i}{\hat{y}^i(1-\hat{y}^i)}\right]\frac{\partial \hat{y}^i}{\partial w_j}$$

Recall that $\hat{y}^i = \hat{y}(\mathbf{x}^i, \mathbf{w}) = \dfrac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x}^i)}$

for $g(t) = \dfrac{1}{1+\exp(-t)}$ we have
$g'(t) = \dfrac{\exp(-t)}{(1+\exp(-t))^2} = g(t)(1-g(t))$

So $\dfrac{\partial \hat{y}^i}{\partial w_j} = \hat{y}^i(1-\hat{y}^i)\dfrac{\partial(\mathbf{w}\cdot\mathbf{x}^i)}{\partial w_j} = \hat{y}^i(1-\hat{y}^i)x_j^i$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \sum_{i=1}^{N}(y^i - \hat{y}^i)x_j^i \qquad \nabla L(\mathbf{w}) = \sum_{i=1}^{N}(y^i - \hat{y}^i)\mathbf{x}^i$$

Batch Gradient Ascent for LR

Given : training examples $(\mathbf{x}^i \ , y^i), \ i = 1,\ldots, N$

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat until convergence

$\quad \mathbf{d} \leftarrow (0,0,0,\ldots,0)$

$\quad$ For $i \ = \ 1$ to N do

$$\widehat{y}^i \leftarrow \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}^i}}$$

$$error = y^i - \widehat{y}^i$$

$$\mathbf{d} = \mathbf{d} + error \cdot \mathbf{x}^i$$

$\quad \mathbf{w} \leftarrow \mathbf{w} + \eta \, \mathbf{d}$

- Online gradient *ascent* algorithm can be easily constructed

Summary of Logistic Regression

- Learns conditional probability distribution $P(y \mid \mathbf{x})$
- Local Search
  - begins with initial weight vector. Modifies it iteratively to maximize the log likelihood of the data
- Online or Batch
  - both online and batch variants of the algorithm exist

Three Main Approaches

- Learn a classifier: a function f, $\hat{y} = f(\mathbf{x})$
- Learn a probabilistic ***discriminative*** model, i.e., the conditional distribution $P(y \mid \mathbf{x})$
- Learn a probabilistic ***generative*** model, i.e., the joint probability distribution: $P(\mathbf{x},y)$
- Examples:
  - Learn a classifier: Perceptron, LDA (projection with threshold view)
  - Learn a conditional distribution: ***Logistic regression***
  - Learn the joint distribution: a probabilistic view of Linear Discriminant Analysis (LDA)

Three Main Approaches

- Learn a classifier: a function f, $\hat{y} = f(\mathbf{x})$
- Learn a probabilistic ***discriminative*** model, i.e., the conditional distribution $P(y \mid \mathbf{x})$
- Learn a probabilistic ***generative*** model, i.e., the joint probability distribution: $P(\mathbf{x}, y)$
- Examples:
  - Learn a classifier: Perceptron, LDA (projection with threshold view)
  - Learn a conditional distribution: ***Logistic regression***
  - Learn the joint distribution: a probabilistic view of Linear Discriminant Analysis (LDA)

Outline

- Text Classification
- Logistic Regression
- **Naive Bayes Classifier**
- Evaluation Metrics
- Summary

Bayes' Rule Applied to Documents and Classes

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

Thomas Bayes
1702 - 1761

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

Naive Bayes Classifier (I)

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

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

Naive Bayes Classifier (II)

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

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

Document d represented as features x1..xn

Naive Bayes Classifier (IV)

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

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

Multinomial Naïve Bayes Independence Assumptions

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

Bag of Words assumption: Assume position doesn't matter

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

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

Multinomial Naïve Bayes Classifier

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

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

Applying Multinomial Naïve Bayes Classifiers to Text Classification

positions ← all word positions in test document

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

Learning the Multinomial Naïve Bayes Model

First attempt: maximum likelihood estimates

- simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

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

Parameter estimation

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

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

Problem with Maximum Likelihood

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!

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

Laplace (add-1) smoothing for Naïve Bayes

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

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

Multinomial Naïve 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 $Text_j$

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

Summary: Naive Bayes is Not So Naive

Very Fast, low storage requirements

Robust to Irrelevant Features

Irrelevant Features cancel each other without affecting results

Very good in domains with many equally important features

Decision Trees suffer from *fragmentation* in such cases – especially if little data

Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem

A good dependable baseline for text classification

- **But we will see other classifiers that give better accuracy**

Outline

- Text Classification
- Logistic Regression
- Naive Bayes Classifier
- **Evaluation Metrics**
- Summary

Evaluation Metrics

- **Precision**
- **Recall**
- **F-scores**

Evaluation Metrics

- **true positive (TP), false positive (FP), true negative (TN) and false negative (FN), respectively.**
- **Obviously, N = TP + FP + TN + FN.**

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Evaluation Metrics

A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

The harmonic mean is a very conservative average;

People usually use balanced F1 measure

- i.e., with $\beta = 1$ (that is, $\alpha = ½$): $F = 2PR/(P+R)$

Outline

- Text Classification
- Logistic Regression
- Naive Bayes Classifier
- Evaluation Metrics
- **Summary**

Summary

- **Many available machine learning algorithms for Text Classification**
- **Algorithms have some limitations like some ideal assumptions**
  - ✓ **i.i.d**
  - ✓ **Feature Independent**
  - ✓ **……**
- **Build your algorithms for the projects**

Text Classification

Thank you!

Q&A