ITCS 5356: Introduction to Machine Learning

Naïve Bayes

Razvan C. Bunescu

Department of Computer Science @ CCI

razvan.bunescu@charlotte.edu

Three Parametric Approaches to Classification

1) Discriminant Functions: construct $h : X \rightarrow T$ that directly assigns a vector $\mathbf{x}$ to a specific class C_k.
   - Inference and decision combined into a single learning problem.
   - *Linear Discriminant*: the decision surface is a hyperplane in X:
     - Fisher 's Linear Discriminant
     - **Perceptron**
     - Support Vector Machines

Three Parametric Approaches to Classification

2) Probabilistic Discriminative Models: directly model the posterior class probabilities $p(C_k \mid \mathbf{x})$.

   - Inference and decision are separate.
   - Less data needed to estimate $p(C_k \mid \mathbf{x})$ than $p(\mathbf{x} \mid C_k)$.
   - Can accommodate many overlapping features.
     - **Logistic Regression**
     - Conditional Random Fields

Three Parametric Approaches to Classification

3) Probabilistic Generative Models:

- Model class-conditional $p(\mathbf{x} \mid C_k)$ as well as the priors $p(C_k)$, then use Bayes's theorem to find $p(C_k \mid \mathbf{x})$.
  - or model $p(\mathbf{x}, C_k)$ directly, then marginalize to obtain the posterior probabilities $p(C_k \mid \mathbf{x})$.
- Inference and decision are separate.
- Can use $p(\mathbf{x})$ for *outlier* or *novelty detection*.
- Need to model dependencies between features.
  - **Naïve Bayes**.
  - Hidden Markov Models.

Text Classification

- **Sentiment analysis**.
- **Spam detection**.
- Authorship identification.
- Language identification.
- Assigning subject categories, topics, or genres.

Text classification: Spam detection

From: Tammy Jordan
jordant@oak.cats.ohiou.edu
Subject: Spring 2015 Course

CS690: Machine Learning

Instructor: Razvan Bunescu
Email: bunescu@ohio.edu
Time and Location: Tue, Thu 9:00 AM , ARC 101
Website: http://ace.cs.ohio.edu/~razvan/courses/ml6830

Course description:
Machine Learning is concerned with the design and analysis of algorithms that enable computers to automatically find patterns in the data. This introductory course will give an overview …

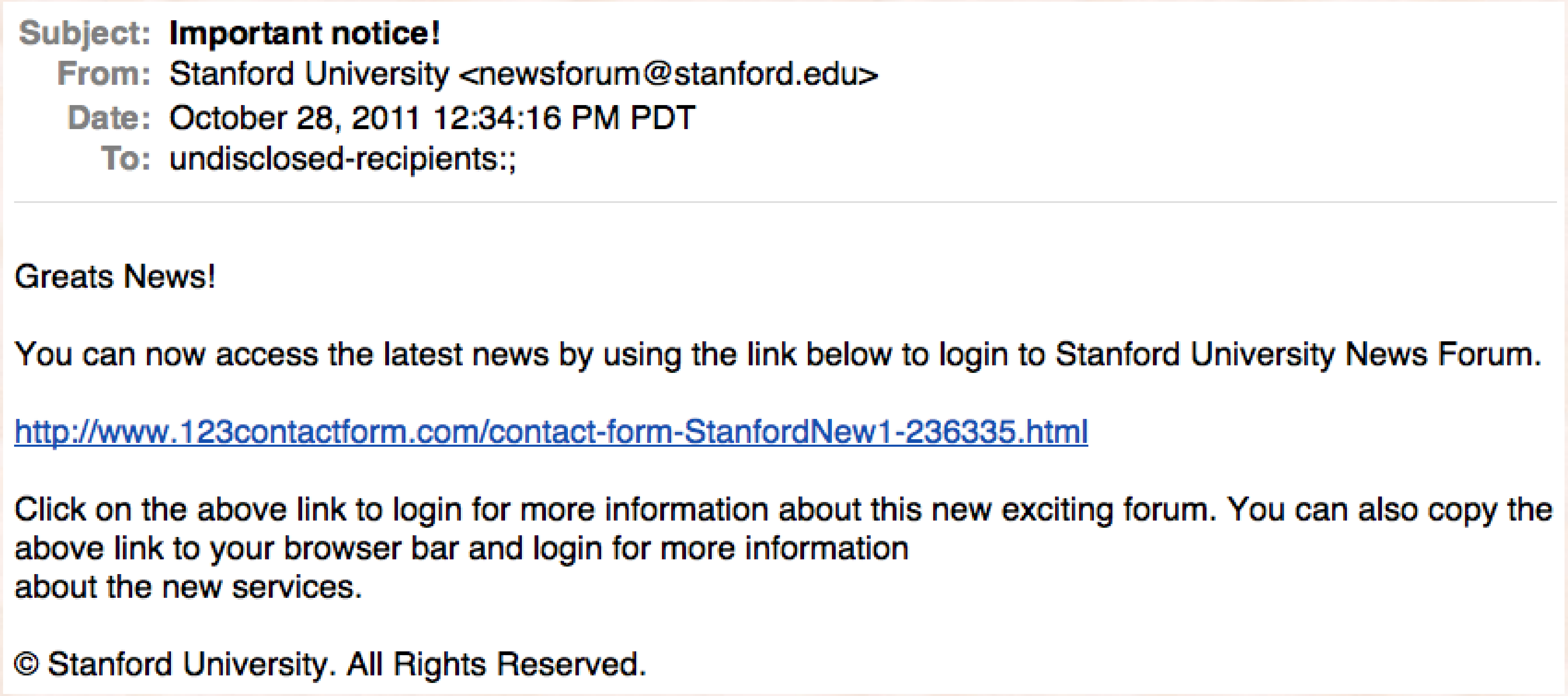
From: UK National Lottery
edreyes@uknational.co.uk
Subject: Award Winning Notice

UK NATIONAL LOTTERY. GOVERNMENT ACCREDITED LICENSED LOTTERY. REGISTERED UNDER THE UNITED KINGDOM DATA PROTECTION ACT;

We happily announce to you the draws of (UK NATIONAL LOTTERY PROMOTION) International programs held in London , England Your email address attached to ticket number :3456 with serial number :7576/06 drew the lucky number 4-2-274, which subsequently won you the lottery in the first category …

- Email filtering:
  - Provide emails labeled as *{Spam, Ham}*.
  - Train *Naïve Bayes* model to discriminate between the two.
    - [Sahami, Dumais & Heckerman, AAAI'98]

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.

© Stanford University. All Rights Reserved.

- **Adversarial setting**: text encapsulated in images, misspelled words, …

Text Classification: Rule-based vs. Machine learning

- *Input*: document $d \in \mathrm{D}$
- *Output:* a predicted class $C_k \in \{C_1, C_2, \ldots, C_K\}$

- **Hand-coded rules** based on combinations of words or other features:
  - **Spam filtering**: 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.

- **Supervised learning**:
  - **Input**:
    - A fixed set of classes $\mathrm{C} = \{C_1, C_2, \ldots, C_K\}$
    - A training set of N hand-labeled documents $(d_1, t_1), (d_2, t_2), \ldots, (d_N, t_N)$, where $t_n \in \mathrm{C}$
  - **Output**:
    - A learned classifier h: $\mathrm{D} \rightarrow \mathrm{C}$

Classification Algorithms

- Train a classification algorithm on the labeled feature vectors, i.e. training examples.
  - Use trained model to determine the class of new (unseen) documents.

- Machine learning models:
  - **Naïve Bayes**
  - **Logistic Regression**
  - **Perceptron**
  - Support Vector Machines
  - **Neural networks**
  - **k-Nearest Neighbors**
  - …

Bayes' Rule Applied to Documents and Classes

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

Likelihood

Posterior

Prior

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

$$P(d) = P(d, c_1) + P(d, c_2)$$

$$P(d) = P(d|c_1)P(c_1) + P(d|c_2)P(c_2)$$

$$P(d) = \sum_{c \in C} P(d|c)P(c)$$

- Inference ≡ find c_{MAP} to minimize misclassification rate:

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

Maximum a Posteriori

What if we want to compute this too?

Naive Bayes Classifier

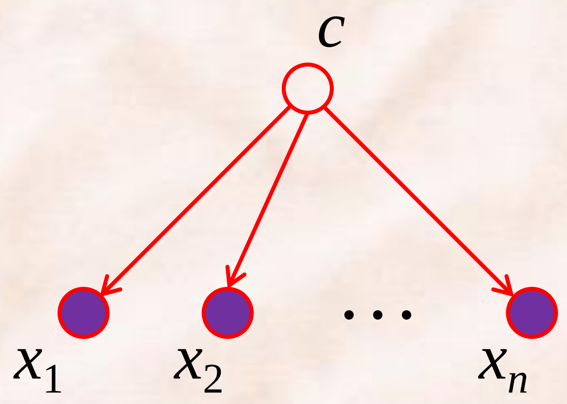
- Inference (at test time): find *maximum a posteriori* (MAP) class:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d \mid c)}^{\textit{Likelihood}} \overbrace{P(c)}^{\textit{Prior}} = \underset{c \in C}{\operatorname{argmax}} P(\underbrace{x_1, x_2, \ldots, x_n}_{\text{document } d \text{ represented as features } x_1,\, x_2,\, \ldots,\, x_n} \mid c) P(c)$$

- If each feature $x_j \in$ X and class c $\in$ C, then $|X|^n \times |C|$ params $P(x_1, x_2, \ldots, x_n \mid c)$:
  - x_j could be word at position *j* in document *d*, X could be entire vocabulary.
    - Number of params is **polynomial** in the size of vocabulary!
      - Could only be estimated if a very, very large number of training examples was available.
        - » Unfeasible in practice.

The Naïve Bayes Model

- **NB assumption**: features are conditionally independent given the target class.



$$P(d|c) = P(x_1,\ldots,x_n|c) = P(x_1|c)\,P(x_2|c)\ldots P(x_n|c)$$

$$\cancel{P(x_1,\ldots,x_n) = P(x_1)\,P(x_2)\ldots P(x_n)}$$

- **BoW assumption**: assume position doesn't matter.

$$P(d|c) = P(x_1|c)\,P(x_2|c)\ldots P(x_n|c) = \prod_{x\in d} P(x|c)$$

- Assuming binary features, i.e. word *w* appears (or not) at position *j*:

  ⇒ need to estimate only $|X|\times|C|$ parameters, a lot less than $|X|^n\times|C|$

Multinomial Naive Bayes Classifier

- MAP inference at test time, using Naïve Bayes model:

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

$$c_{MAP} = c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{x \in d} P(x|c)$$

- Use probabilities over all word *positions* in the document *d*:

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

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

Naïve Bayes: Use log-space to avoid underflow

- Instead of this:

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

$$\log(ab) = \log(a) + \log(b)$$

- Work in log-space:

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

- This is ok since *log* doesn't change the ranking of the classes:
  - class with highest prob still has highest log prob.

- Model is now just max of sum of weights:
  - A **linear** function of the parameters. So Naïve Bayes is a **linear classifier**.

Learning the Multinomial Naive Bayes Model

- **Maximum Likelihood** estimates:
  - Use the frequencies of features in the data.

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}} \qquad \hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

- Create mega-document for topic *j* by concatenating all docs in this topic:
  - Use frequency of *w* in mega-document.

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

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

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **spam**?

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

- 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 Smoothing for Naïve Bayes

- **Laplace (add-1) smoothing**:
  - |V| "hallucinated" examples spread evenly over all |V| values of w_i, for each class c.

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

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

Unknown words and Stop words

- **Unknown words** appear in the test data, but not in our training data or vocab.
  - Ignore unknown words.
    - Remove them from the test document, i.e. 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 useful to know.

- **Stop words:** very frequent words like *the* and *a*:
  - Sort the vocabulary by frequency in the training, call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from the training and test sets, as if they were never there.
    - But in most text classification applications, removing stop words don't help, so it's more common to **not** use stopword lists and use all the words in naive Bayes.

Text Categorization with Naïve Bayes

- Generative model of documents:
  1) Generate document category by sampling from $p(c_j)$.
  2) Generate a document as a bag of words by repeatedly sampling with replacement from a vocabulary V = $\{w_1, w_2, \ldots, w_{|V|}\}$ based on $p(w_i \mid c_j)$.

When do we stop generating words? Provide two solutions …

- Inference with Naïve Bayes:
  - Input :
    - Document d with n words $x_1, x_2, \ldots x_n$.
  - Output:
    - Category $c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$

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

Text Categorization with Naïve Bayes

- Training with Naïve Bayes:
  - Input:
    - Dataset of training documents D with vocabulary V.
  - Output:
    - Parameters $p(C_k)$ and $p(w_i \mid C_k)$.

```
1. for each category $C_k$:
2.     let $D_k$ be the subset of documents in category $C_k$
3.     set $p(C_k) = |D_k| / |D|$
4.     let $n_k$ be the total number of words in $D_k$
5.     for each word $w_i \in V$:
6.         let $n_{ki}$ be the number of occurrences of $w_i$ in $D_k$
7.         set $p(w_i \mid C_k) = (n_{ki}+1) / (n_k + |V|)$
```

A worked sentiment analysis example

	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

Prior from training:

P(−) = 3/5
P(+) = 2/5

Drop **unknown words**, i.e. "with”.

Likelihoods from training:

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

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

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

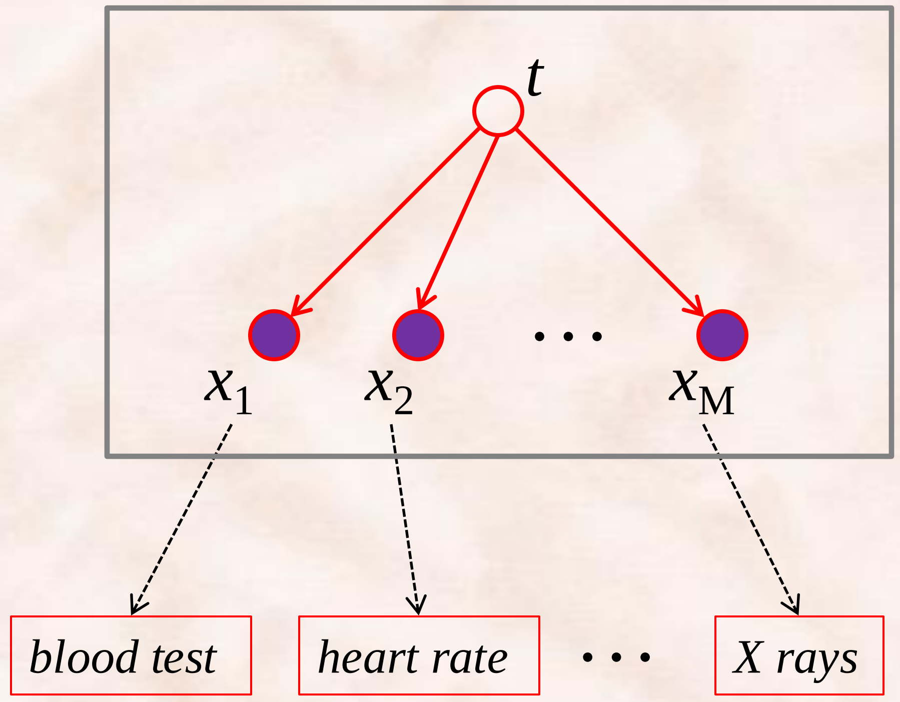
Scoring the test document:

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

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

Medical Diagnosis with Naïve Bayes

- Diagnose a disease T={*Yes*, *No*}, using information from various medical tests.



$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{M} p(x_i \mid C_k)$$

Medical tests may result in continuous values $\Rightarrow$ need Naïve Bayes to work with *continuous features*.

Naïve Bayes with Continuous Features

- Assume $p(x_i \mid C_k)$ are Gaussian distributions $N(\mu_{ik},\sigma_{ik})$.
- Training: use ML or MAP criteria to estimate μ_{ik},σ_{ik}:

$$\hat{\mu}_{ik} = \frac{\sum_{(\mathbf{x},t)\in D} x_i \delta_{C_k}(t)}{\sum_{(\mathbf{x},t)\in D} \delta_{C_k}(t)} \qquad \hat{\sigma}_{ik}^2 = \frac{\sum_{(\mathbf{x},t)\in D} (x_i - \hat{\mu}_{ik})^2 \delta_{C_k}(t)}{\sum_{(\mathbf{x},t)\in D} \delta_{C_k}(t)}$$

- Inference:

$$C_* = \arg\max_{C_k} p(C_k \mid \mathbf{x}) = \arg\max_{C_k} p(C_k) \prod_i p(x_i \mid C_k)$$

Naïve Bayes

- Often has good performance, despite strong independence assumptions:
  - Quite competitive with other classification methods on UCI datasets.
- It does not produce accurate probability estimates when independence assumptions are violated:
  - The estimates are still useful for finding max-probability class.
- Does not focus on completely fitting the data $\Rightarrow$ resilient to noise.
- NB model is sum of weights = a **linear** function of the inputs.

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

Required Reading

- Chapter 4 on Naïve Bayes and Text Classification (also in Canvas):
  – Speech and Language Processing, by Daniel Juraksfy and James E. Martin. 2024.