Algorithms in MapReduce

Analytics Algorithms

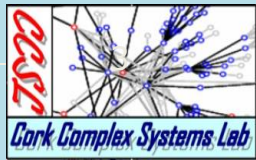
Plan for today

- Analytics algorithms
  - k-means clustering
  - Classification with Naïve Bayes

Analytics Toolbox

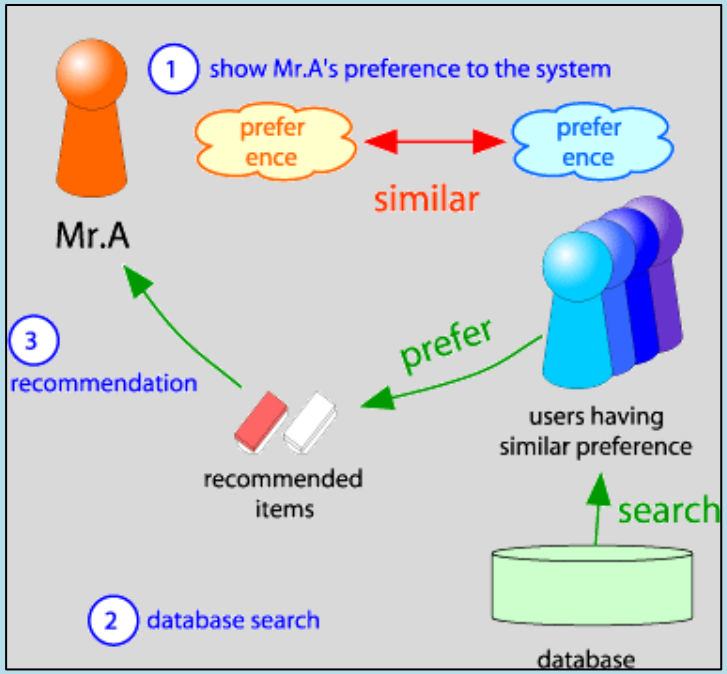
- We need a toolbox of algorithms useful for analyzing data that has both relationships and properties

- Learning algorithms are critical



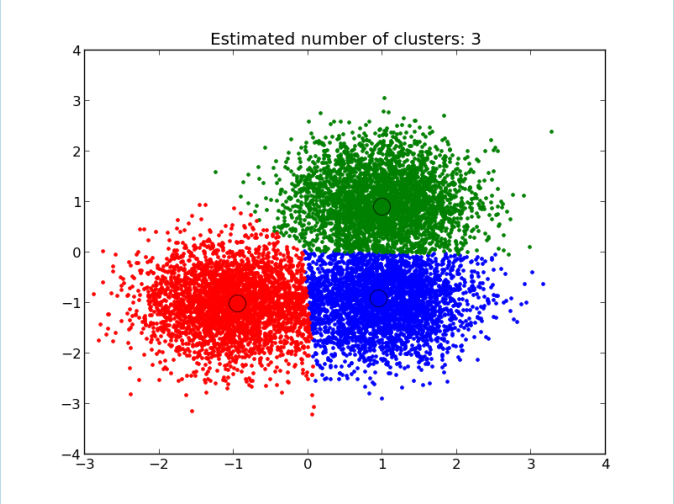
C1: Collaborative Filtering

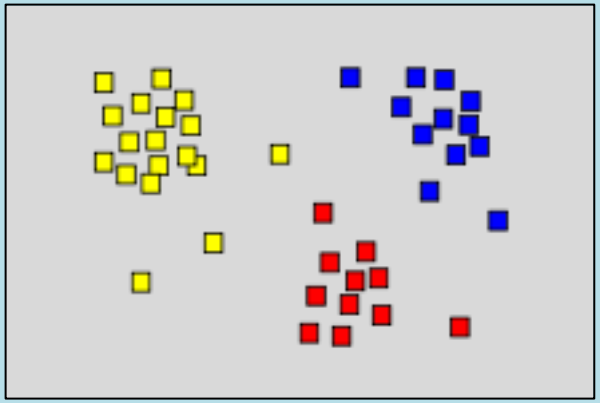
- Extensive framework for collaborative filtering (recommenders)
- Recommenders
  - User based
  - Item based
- Online and Offline support
  - Offline can utilize Hadoop
- Many different Similarity measures
  - Cosine, LLR, Tanimoto, Pearson, others



C2: Clustering

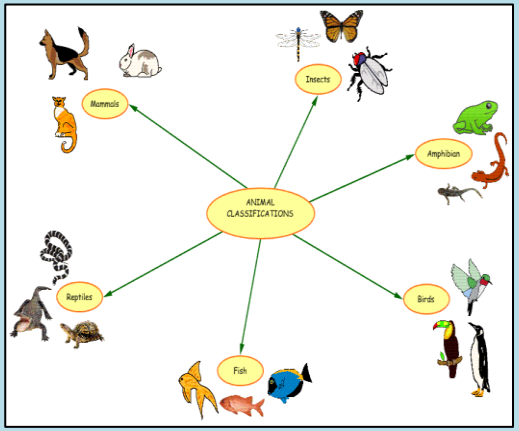
- Group similar objects together
- K-Means, Fuzzy K-Means, Density-Based,…
- Different distance measures
  - Manhattan, Euclidean, …

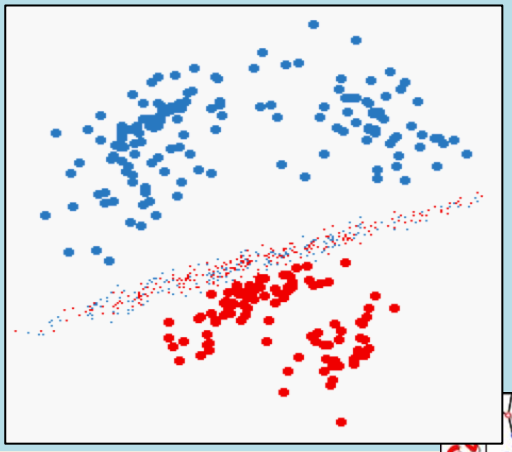




C3: Classification

- Place new items into predefined categories:
  - Sports, politics, entertainment
  - Recommenders
- Implementations
  - Naïve Bayes (M/R)
  - Compl. Naïve Bayes (M/R)
  - Decision Forests (M/R)
  - Linear Regression (Seq. but Fast!)





Cork Complex Systems Lab

FPM: Frequent Pattern Mining

- **Find the frequent itemsets**
  - <milk, bread, cheese> are sold frequently together

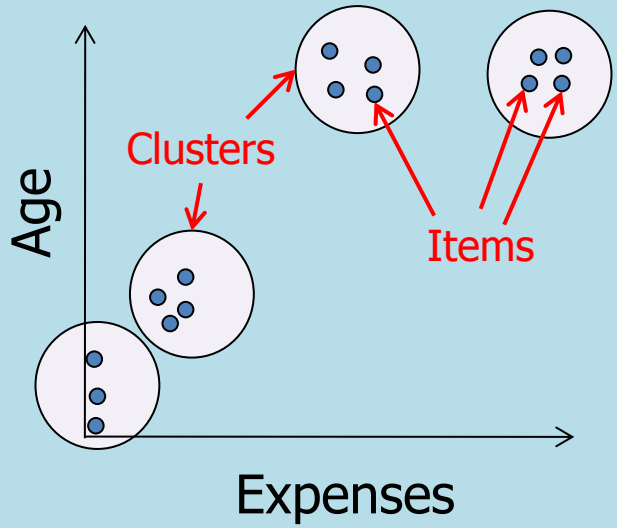
- Very common in market analysis, access pattern analysis, etc…

Learning (clustering / classification)

- Sometimes our goal is to take a set of entities, possibly related, and group them
  - If the groups are based on similarity, we call this clustering
  - If the groups are based on putting them into a semantically meaningful class, we call this classification

- Both are instances of machine learning

The k-clustering Problem



- Given: A set of items in a n-dimensional feature space
  - Example: data points from survey, people in a social network
- Goal: Group the items into k “clusters”
  - What would be a 'good' set of clusters?

K-Means Algorithm

- **Step 1:** Select K points at random (Centers)
- **Step 2**: For each data point, assign it to the closest center
  - Now we formed K clusters
- **Step 3:** For each cluster, re-compute the centers
  - E.g., in the case of 2D points ➔
    - X: average over all x-axis points in the cluster
    - Y: average over all y-axis points in the cluster
- **Step 4:** If the new centers are different from the old centers (previous iteration) ➔ Go to Step 2

Approach: k-Means

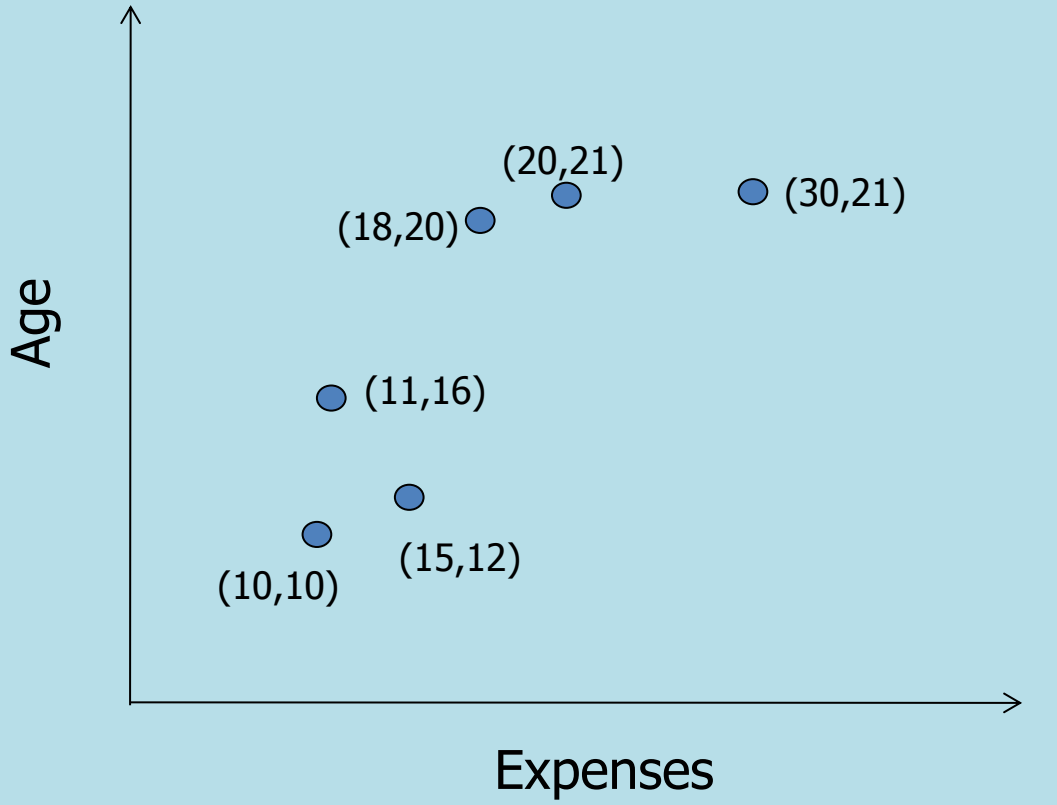
- Let m_1, m_2, …, m_k be representative points for each of our k clusters
  - Specifically: the centroid of the cluster
- Initialize m_1, m_2, …, m_k to random values in the data
- For t = 1, 2, …:
  - Map each observation to the closest mean

$$S_i^{(t)} = \left\{ x_j : \left\| x_j - m_i^{(t)} \right\| \leq \left\| x_j - m_{i^*}^{(t)} \right\|, i^* = 1, \ldots, k \right\}$$

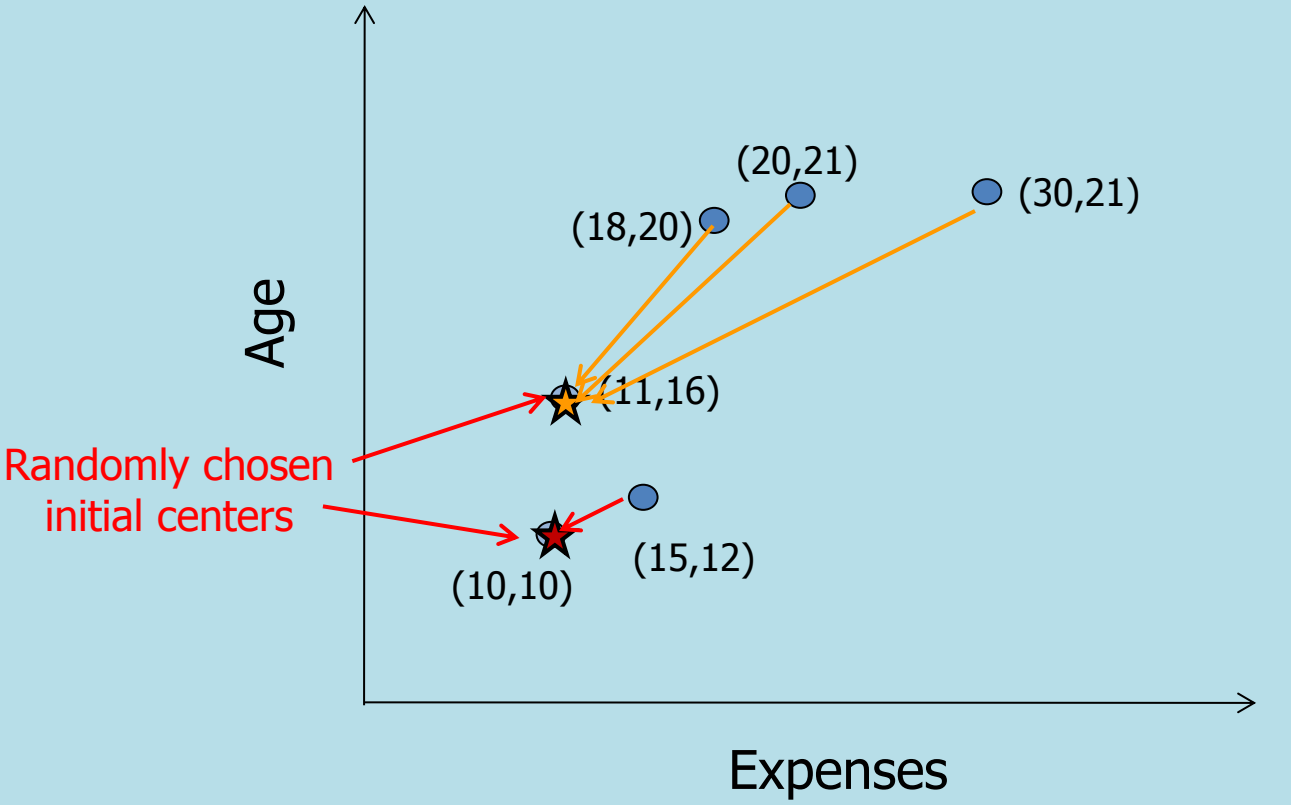
  - Assign the m_i to be a new centroid for each set

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_j \in S_i^{(t)}} x_j$$

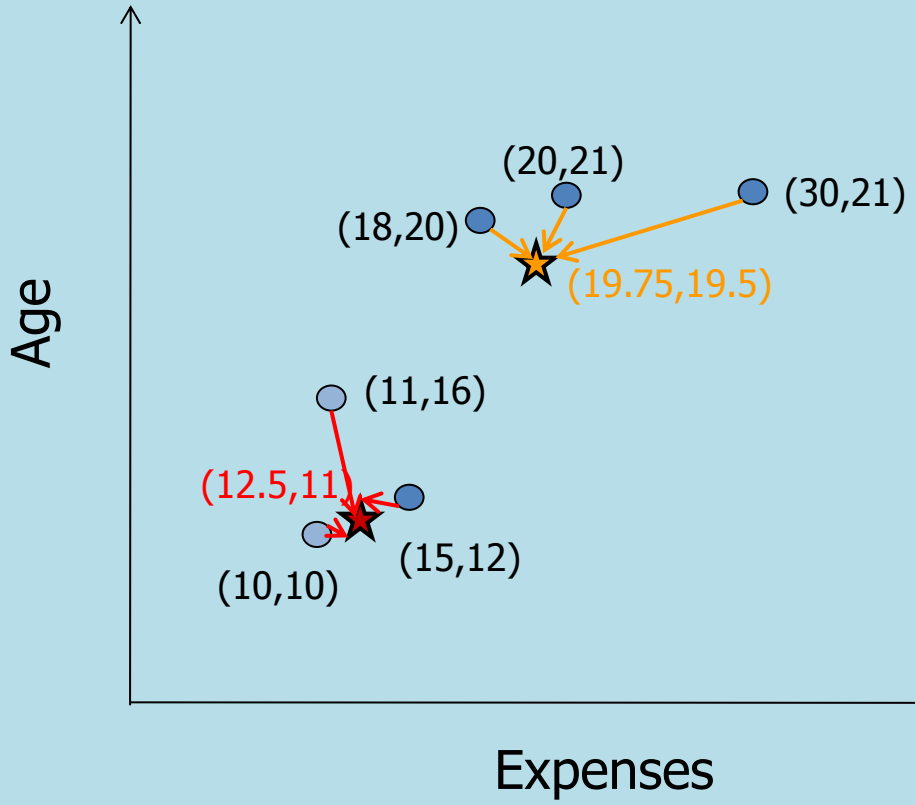
A simple example (1/4)



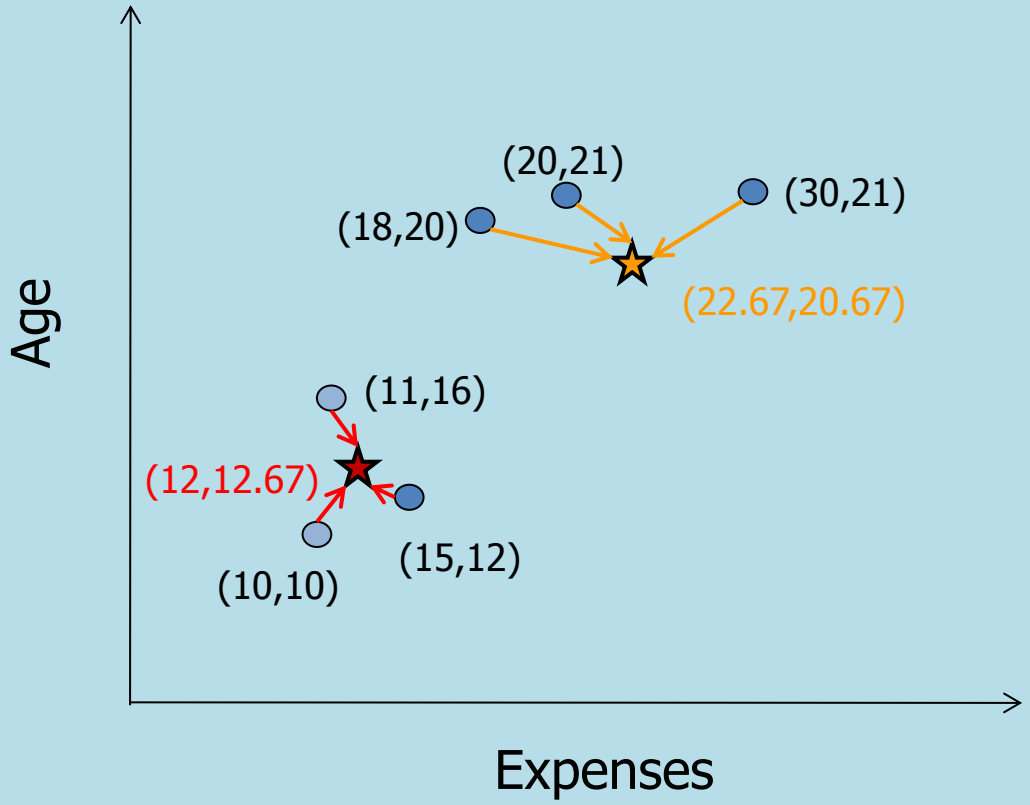
A simple example (2/4)



A simple example (3/4)



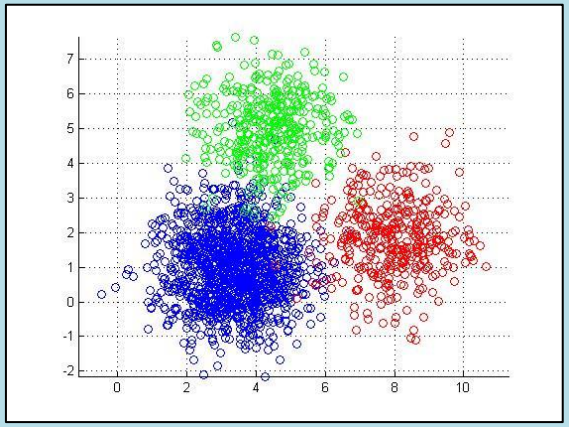
A simple example (4/4)



Stable!

K-Means in MapReduce

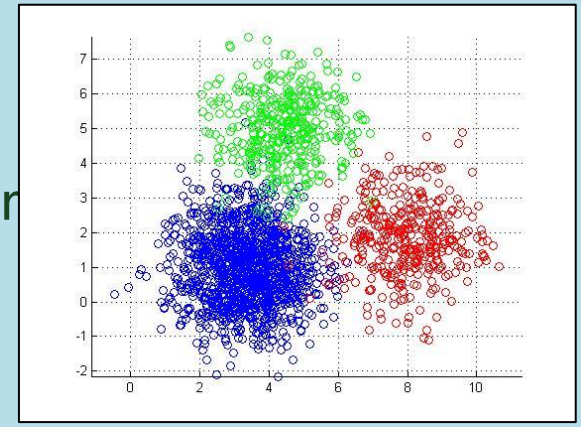
- **Input**
  - Dataset (set of points in 2D) --Large
  - Initial centroids (K points) --Small



- **Map Algorithm**
  - Each map reads the K-centroids + one block from dataset
  - Assign each point to the closest centroid
  - Output <centroid, point>

K-Means in MapReduce (Cont'd)

- **Reduce Algorithm**
  - Gets all points for a given centroid
  - Re-compute a new centroid for this cluster
  - Output: <new centroid>



- **Iteration Control**
  - Compare the old and new set of K-centroids
    - If similar ➔ Stop
    - Else
      - If max iterations has reached ➔ Stop
      - Else ➔ Start another Map-Reduce Iteration

k-Means in MapReduce

- Map:#1:
  - Input: node ID → <position, centroid ID, [centroid IDs and positions]>
  - Compute nearest centroid; emit centroid ID → <node ID, position>
- Reduce:#1:
  - Recompute centroid position from positions of nodes in it
  - ~~Emit node ID → <position, centroid ID, [centroid IDs and positions]>~~ and for all other centroid IDs, emit otherCentroidID → centroid(centroidID,X,Y)
    - Each centroid will need to know where all the other centroids are
- Map #2:
  - Pass through values to Reducer #2
- Reduce #2:
  - For each node in the current centroid, emit
    node ID → <position, centroid ID, [centroid IDs and positions]>
    - Input for the next map iteration
  - Also, emit <X, <centroid ID, position>>
    - This will be the 'result' (remember that we wanted the centroids!)
- Repeat until no change

Plan for today

- A toolbox of algorithms
  - k-means clustering 
  - Classification with Naïve Bayes 

Classification

- Suppose we want to learn what is spam (or interesting, or …)
  - Predefine a set of classes with semantic meanin
  - Train an algorithm to look at data and assign a c
    - Based on giving it some examples of data in each class
    - … and the sets of features they have

- Many probabilistic techniques exist
  - Each class has probabilistic relationships with others
    - e.g., p (spam | isSentLocally), p (isSentLocally | fromBob), …
    - Typically represented as a graph(ical model)!
  - But we'll focus on a simple model: Naïve Bayes

A simple example

- Suppose we just look at the keywords in the email's title:

  Message(1, “Won contract”)
  Message(2, “Won award”)
  Message(3, "Won the lottery")
  Message(4, “Unsubscribe”)
  Message(5, "Millions of customers")
  Message(6, "Millions of dollars")

- What is **probability** message "Won Millions" is

$$p(spam|containsWon,containsMillions)$$

$$= \frac{p(spam)\, p(containsWon,containsMillions|spam)}{p(containsWon,containsMillions)}$$

Bayes’ Theorem

Classification using Naïve Bayes

- Basic assumption: Probabilities of events are independent
  - This is why it is called 'naïve'

- Under this assumption,

$$\frac{p(spam)\ p(containsWon, containsMillions \mid spam)}{p(containsWon, containsMillions)}$$

$$= \frac{p(spam)\ p(containsWon \mid spam)\ p(containsMillions \mid spam)}{p(containsWon)\ p(containsMillions)}$$

$$= 0.5 * 0.67 * 0.33 \ / \ (0.5 * 0.33) = 0.67$$

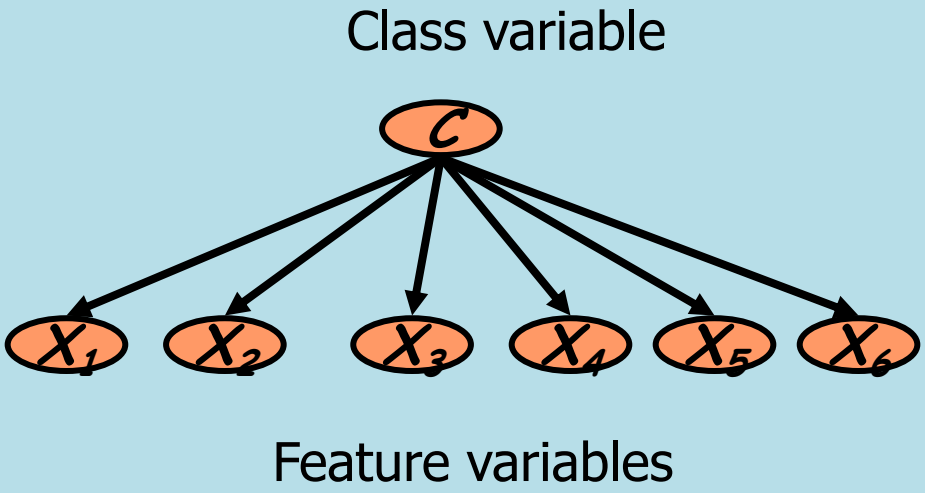
- So how do we “train” a learner (compute the above probabilities) using MapReduce?

What do we need to train the learner?

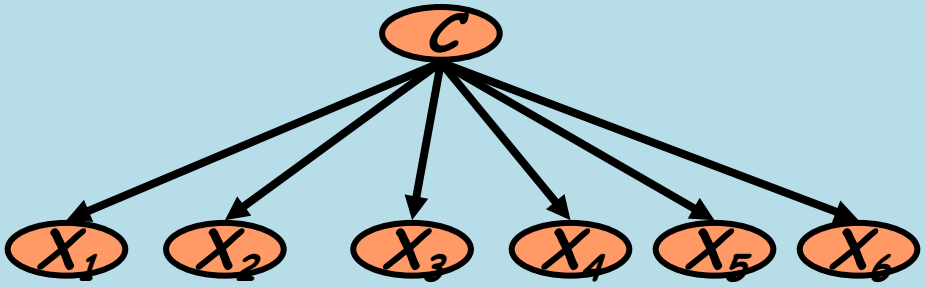
- p(spam)
  - Count how many spam emails there are — Easy
  - Count total number of emails — Easy
- p(containsXYZ | spam)
  - Count how many spam emails contain XYZ
  - Count how many emails are spam overall — 1 — Easy
- p(containsXYZ)
  - Count how many emails contain XYZ overall
  - Count total number of emails — 2 — Easy

Probabilistic relevance feedback

- User has told us some relevant and some irrelevant documents
- Build a probabilistic classifier: Naive Bayes model
- Classifies relevant/irrelevant based on features



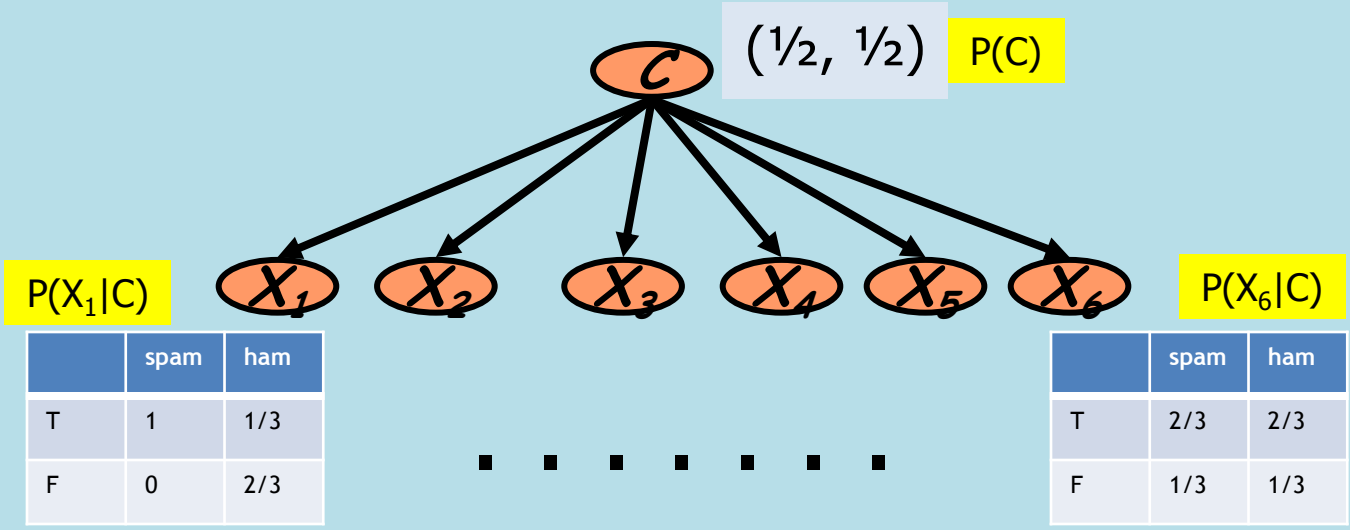
1. Learning the Model



- *Input: Classified data*
  - simply compute the frequencies in the data

Class	X1	X2	X3	X4	X5	X6
spam	T	F	F	T	F	F
Ham	F	T	T	F	F	T
spam	T	F	T	T	F	F
spam	T	T	F	T	T	T
Ham	F	T	T	F	T	T
Ham	T	F	T	T	t	F

1. Learning the Model



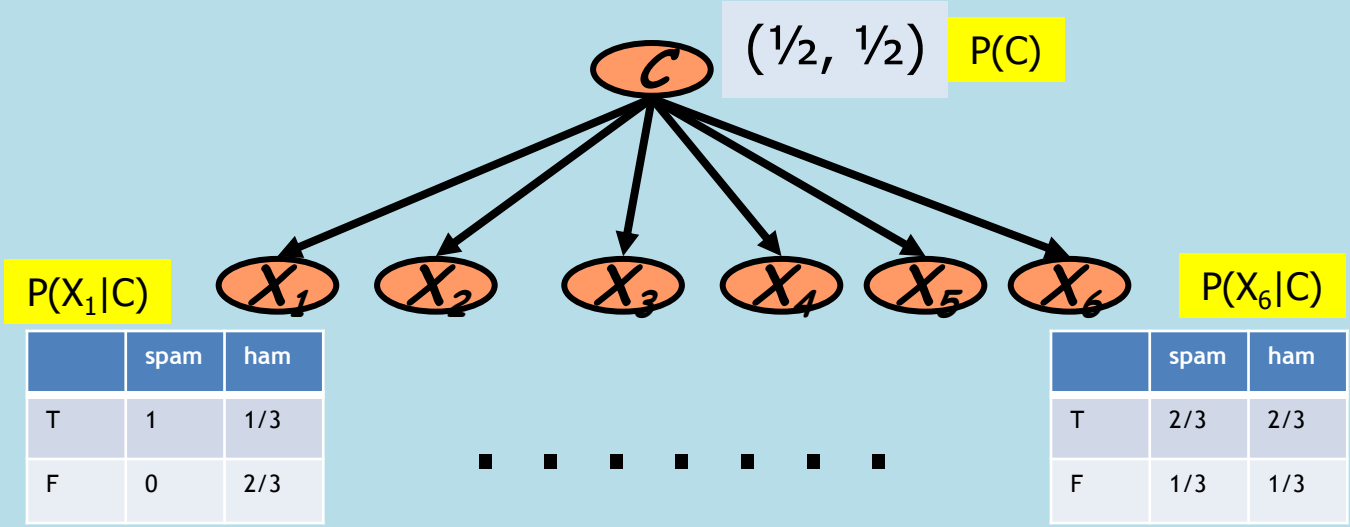
	spam	ham
T	1	1/3
F	0	2/3

	spam	ham
T	2/3	2/3
F	1/3	1/3

- Compute the frequencies in the data

Class	X1	X2	X3	X4	X5	X6
spam	T	F	F	T	F	F
Ham	F	T	T	F	F	T
spam	T	F	T	T	F	F
spam	T	T	F	T	T	T
Ham	F	T	T	F	T	T
Ham	T	F	T	T	T	F

Learning the Model



	spam	ham
T	1	1/3
F	0	2/3

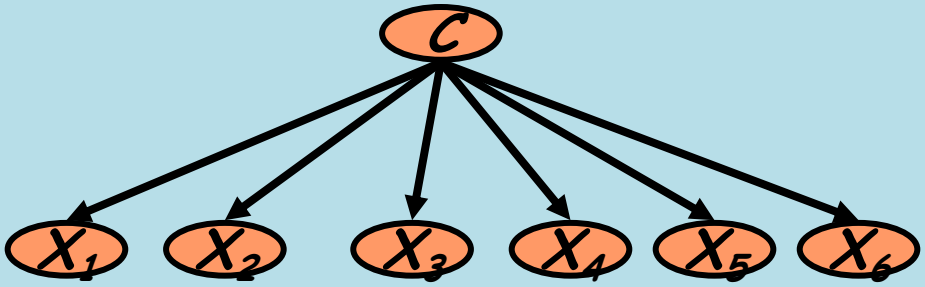
	spam	ham
T	2/3	2/3
F	1/3	1/3

- Computing the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

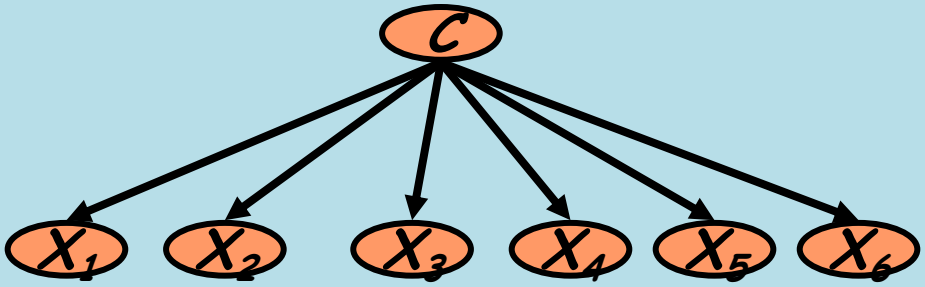
$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

2: Classifying New Instances



- *Input: features (X_1,…,X_6)*
- *Must compute*
  - *P(spam | X_1,…,X_6)*
  - *P(ham | X_1,…,X_6)*
- *Class assigned has higher probability*

2: Classifying New Instances



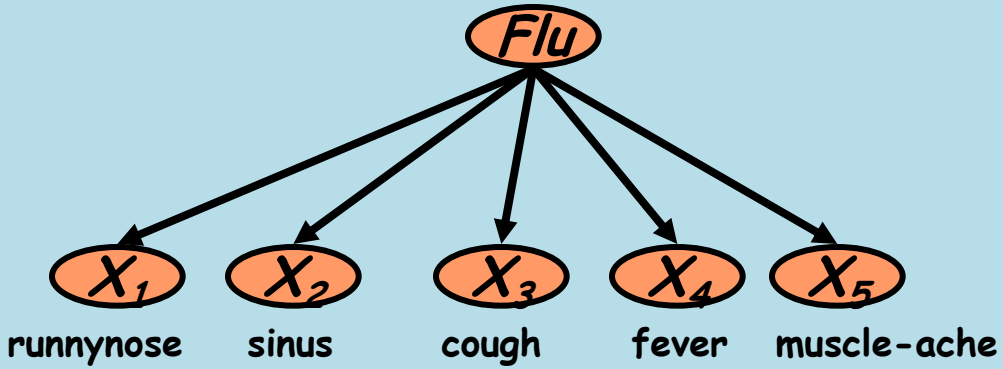
- *Input: features (X_1,…,X_6)*
- *We know:*
  - *$P(X_1|spam)$…,$P(X_6|spam)$, $P(spam)$*
  - *$P(X_1|ham)$…,$P(X_6|ham)$, $P(ham)$*
- *Must compute*
  - *$P(spam \mid X_1,…,X_6)$*
  - *$P(ham \mid X_1,…,X_6)$*
- *How to do this?*

Bayes' Rule

$$P(C, X) = P(C \mid X)P(X) = P(X \mid C)P(C)$$

$$\boxed{P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}}$$

The Naïve Bayes Classifier

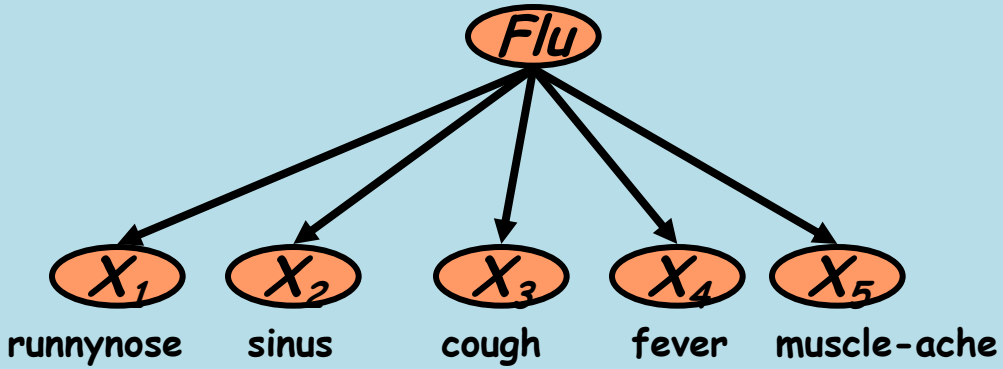


- **Conditional Independence Assumption:** Features (term presence) are *independent* of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables
  - Multivariate Bernoulli model

The Naïve Bayes Classifier



$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- Use Bayes' Rule to "invert" the model

$$P(C \mid X) \propto P(C)\prod_{i=1}^{n} P(X_i \mid C)$$

$$P(\neg C \mid X) \propto P(\neg C)\prod_{i=1}^{n} P(X_i \mid \neg C)$$

Summary of model and parameters

- Naïve Bayes model:

$$P(spam \mid message) \propto P(spam) \prod_{i=1}^{n} P(w_i \mid spam)$$

$$P(\neg spam \mid message) \propto P(\neg spam) \prod_{i=1}^{n} P(w_i \mid \neg spam)$$

- Model parameters:

prior

$P(spam)$

$P(\neg spam)$

Likelihood of spam

$P(w_1 \mid spam)$

$P(w_2 \mid spam)$

$\ldots$

$P(w_n \mid spam)$

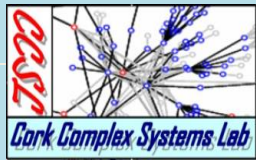
Likelihood of ¬spam

$P(w_1 \mid \neg spam)$

$P(w_2 \mid \neg spam)$

$\ldots$

$P(w_n \mid \neg spam)$



Naive Bayes Classifiers

Task: Classify a new instance *D* based on a tuple of attribute values $D = \langle x_1, x_2, \ldots, x_n \rangle$ into one of the classes $c_j \in C$

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

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

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

MAP = Maximum Aposteriori Probability

Naïve Bayes Classifier: Naïve Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

Example

- Test Phase
  - Given a new instance, predict its label

    $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables achieved in the learning phrase

P(Outlook=*Sunny*|Play=*Yes*) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=S*unny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

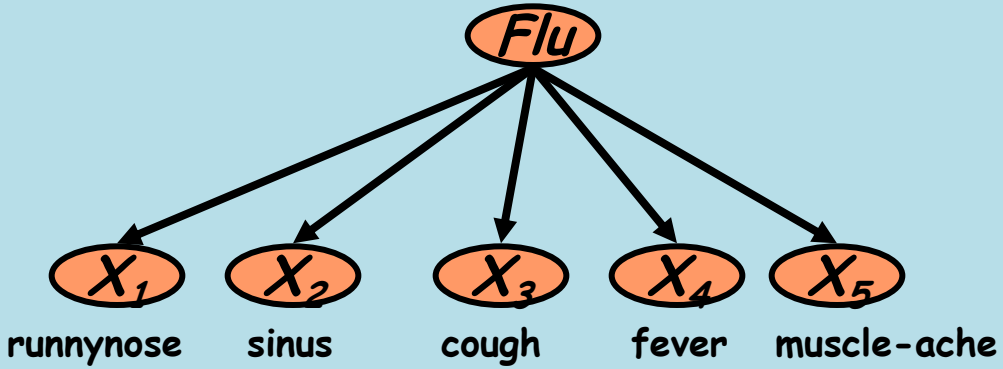
  - Decision making with the MAP rule

P(*Yes*|$\mathbf{x}'$): [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|$\mathbf{x}'$): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

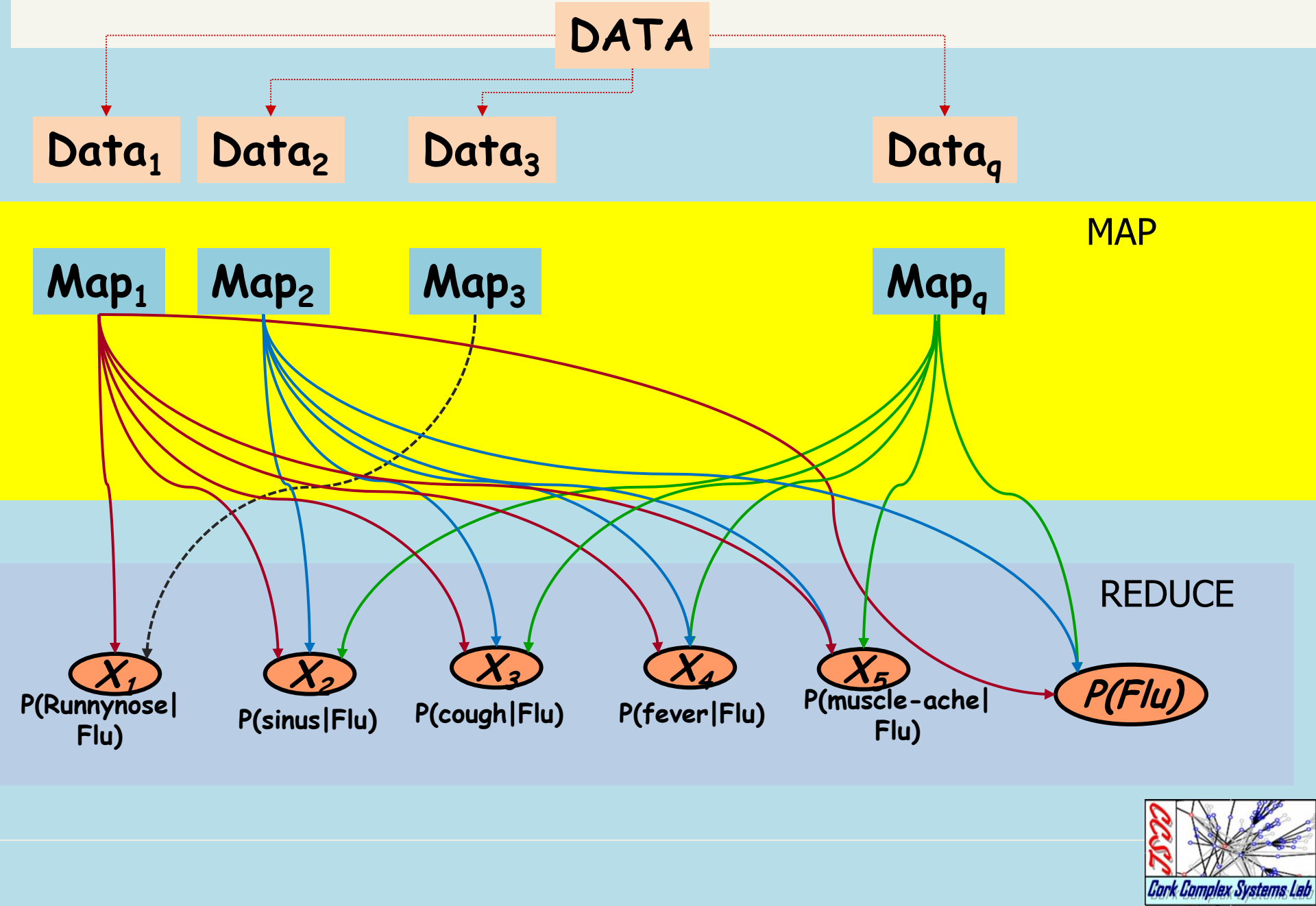
Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

MapReduce for Naïve Bayes



- Example: Medical classification
- Map
  - Learn the probability tables
- Reduce
  - Estimate class posteriors given attribute vector

MapReduce Approach: Training



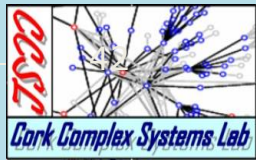
Training Naïve Bayes #1

- map 1:
  - takes messageId → <class, {words}>
  - emits <word, class> → 1
- reduce 1:
  - emits <word, class> → <count>

Count how many emails in the class contain the word (modified WordCount)

- map 2:
  - takes messageId -> <class, {words}>
  - emits word → 1
- reduce 2:
  - emits word → <totalCount>

Count how many emails contain the word overall (WordCount)



Summary: Learning and MapReduce

- Clustering algorithms typically have multiple aggregation stages or iterations
  - k-means clustering repeatedly computes centroids, maps items to them
  - Fixpoint computation

- Classification algorithms can be quite complex
  - In general: need to capture conditional probabilities
  - Naïve Bayes assumes everything is independent
  - Training is a matter of computing probability distribution
    - Can be accomplished using two Map/Reduce passes