Overfitting

- Model is too specific
  - Tailored to fit anomalies in training data
  - Performs suboptimally on general data
- Prune the tree
  - Top-down: stop expanding tree if information gain drops below a threshold
  - Bottom-up:
    - Use statistical estimate of error
    - Remove children of a node if estimated error across children is more than for original

Overfitting . . .

Party affiliation of US legislators based on voting pattern, after pruning

```
physician fee freeze = n: democrat (168/2.6)
physician fee freeze = y: republican (123/13.9)
physician fee freeze = u:
|   mx missile = n: democrat (3/1.1)
|   mx missile = y: democrat (4/2.2)
|   mx missile = u: republican (2/1)
```

Overfitting . . .

Party affiliation of US legislators based on voting pattern

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151)
|   adoption of the budget resolution = u: democrat (1)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6)
|   |   education spending = y: democrat (9)
|   |   education spending = u: republican (1)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97/3)
|   synfuels corporation cutback = u: republican (4)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2)
|   |   duty free exports = u: republican (1)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5/2)
|   |   |   education spending = y: republican (13/2)
|   |   |   education spending = u: democrat (1)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0)
|   water project cost sharing = y: democrat (4)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0)
|   |   mx missile = y: democrat (3/1)
|   |   mx missile = u: republican (2)
```

Bottlenecks in building a classifier

- Noise : Uncertainty in classification function
- Bias : Systematic inability to predict a particular value
- Variance: Variation in model based on sample of training data

Models with high variance are unstable

- Decision trees: choice of attributes influenced by entropy of training data
- Overfitting: model is tied too closely to training set
- Is there an alternative to pruning?

Multiple models

- Build many models (ensemble) and “average” them
- How do we build different models from the same data?
  - Strategy to build the model is fixed
  - Same data will produce same model
- Choose different samples of training data

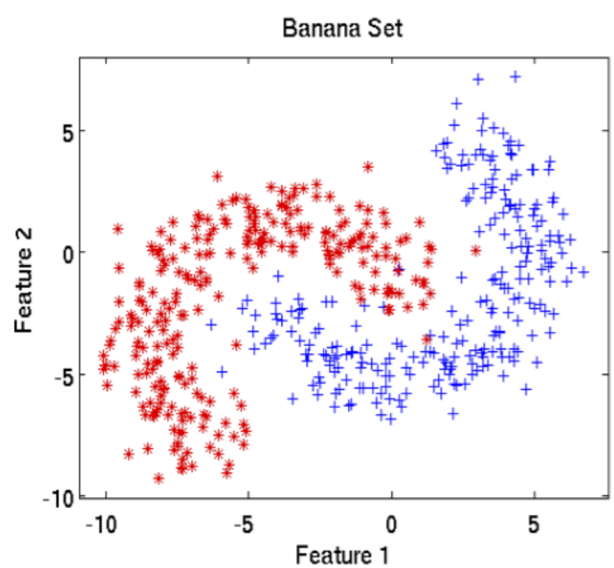
Bootstrap Aggregating = Bagging

- Sample with replacement of size N : bootstrap sample
  - Approx 60% of full training data
- Take K such samples
- Build a model for each sample
  - Models will vary because each uses different training data
- Final classifier: report the majority answer
  - Assumptions: binary classifier, K odd
- Provably reduces variance

Bootstrap Aggregating = Bagging

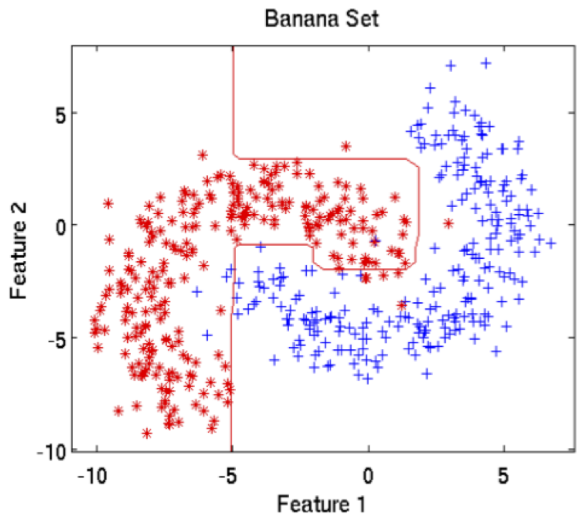
- Training data has N items
  - $TD = \{d_1, d_2, \ldots, d_N\}$
- Pick a random sample with replacement
  - Pick an item at random (probability $\frac{1}{N}$)
  - Put it back into the set
  - Repeat K times
- Some items in the sample will be repeated
- If sample size is same as data size ($K = N$), expected number of distinct items is $(1 - \frac{1}{e}) \cdot N$
  - Approx 63.2%

Bagging with decision trees



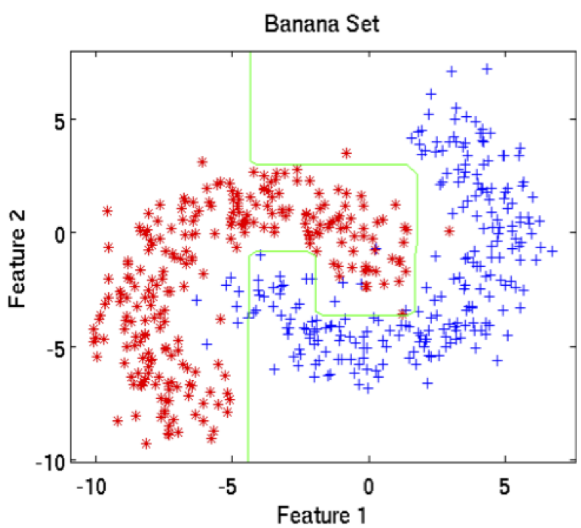
Training data

Bagging with decision trees



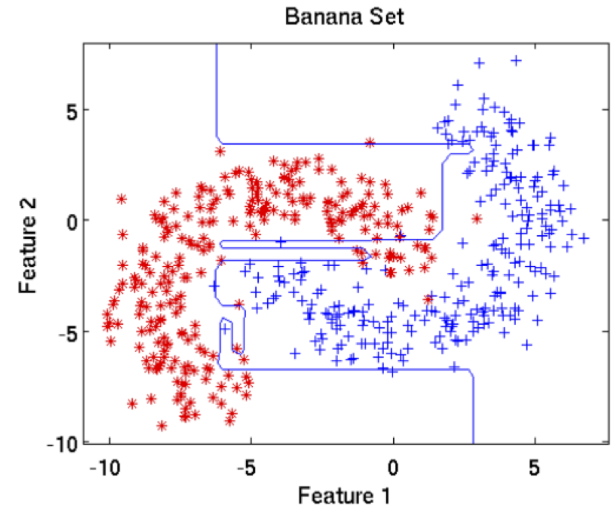
Decision boundary produced by one tree

Bagging with decision trees



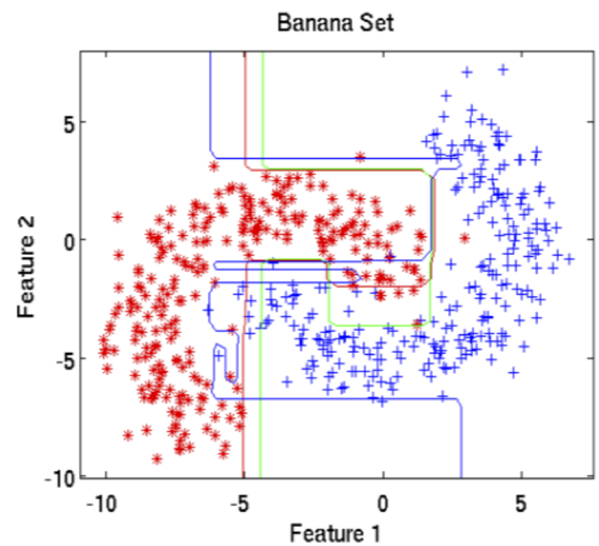
Decision boundary produced by a second tree

Bagging with decision trees



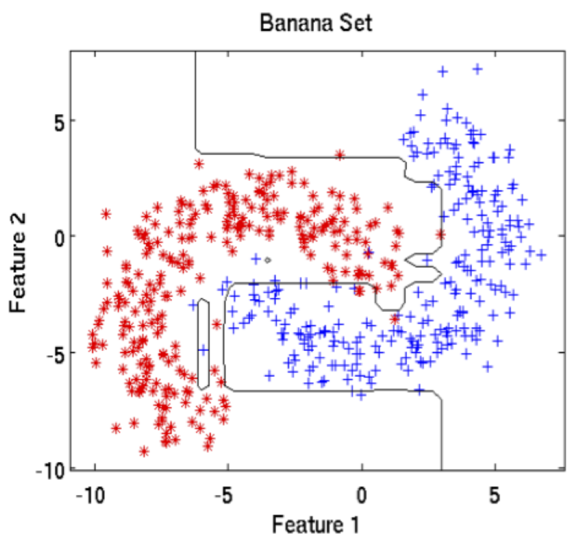
Decision boundary produced by a third tree

Bagging with decision trees



Three trees and final boundary overlaid

Bagging with decision trees



Final result from bagging all trees.

Boosting

- Looking at a few attributes gives "rule of thumb" heuristic
  - If Amla does well, South Africa usually wins
  - If opening bowlers take at least 2 wickets within 5 overs, India usually wins
  - ...
- Each heuristic is a weak classifier
- Can we combine such weak classifiers to boost performance and build a strong classifier?

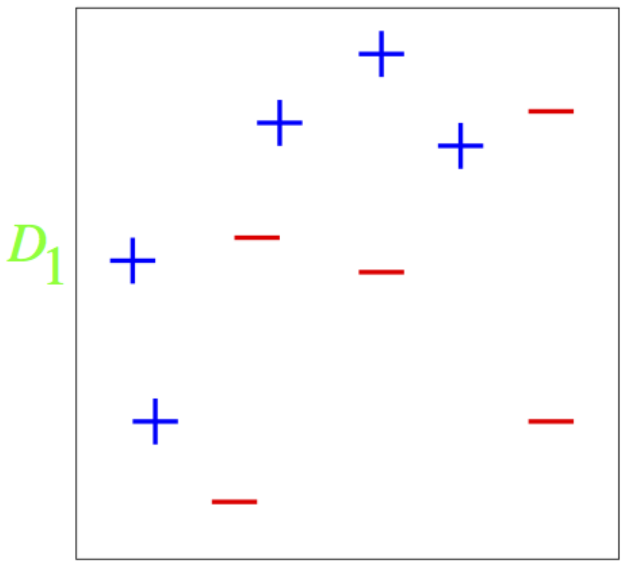
Random Forest

- Applying bagging to decision trees with a further twist
- Each data item has M attributes
- Normally, decision tree building chooses one among M attributes, then one among remaining $M-1$, ...
- Instead, fix a small limit $m < M$
- At each level, choose m of the available attributes at random, and only examine these for next split
- No pruning
- Seems to improve on bagging in practice

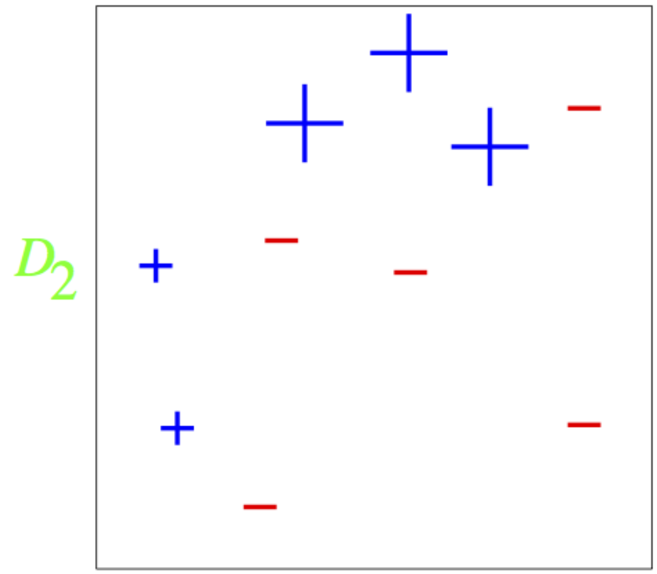
Adaptively boosting a weak classifier (AdaBoost)

- Weak binary classifier: output is $\{-1, +1\}$
- Initially, all training inputs have equal weight, D_1
- Build a weak classifier C_1 for D_1
  - Compute its error rate, e_1 (Details suppressed)
  - Increase weightage to all incorrectly classified inputs, D_2
- Build a weak classifier C_2 for D_2
  - Compute its error rate, e_2
  - Increase weightage to all incorrectly classified inputs, D_3
- ...
- Combine the outputs $o_1, o_2, \ldots, o_k$ of $C_1, C_2, \ldots, C_k$ as $w_1 o_1 + w_2 o_2 + \cdots + w_k o_k$
  - Each weigth w_j depends on error rate e_j
- Report the sign (negative $\mapsto -1$, positive $\mapsto +1$)

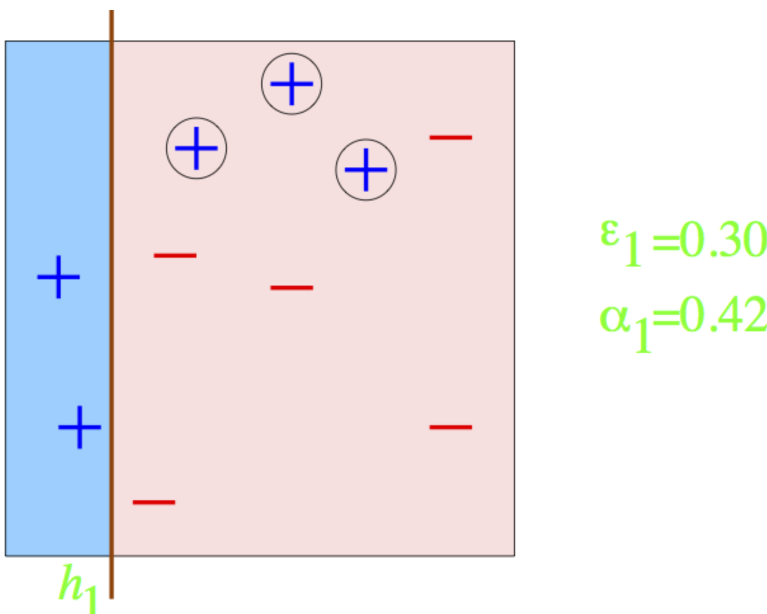
Boosting



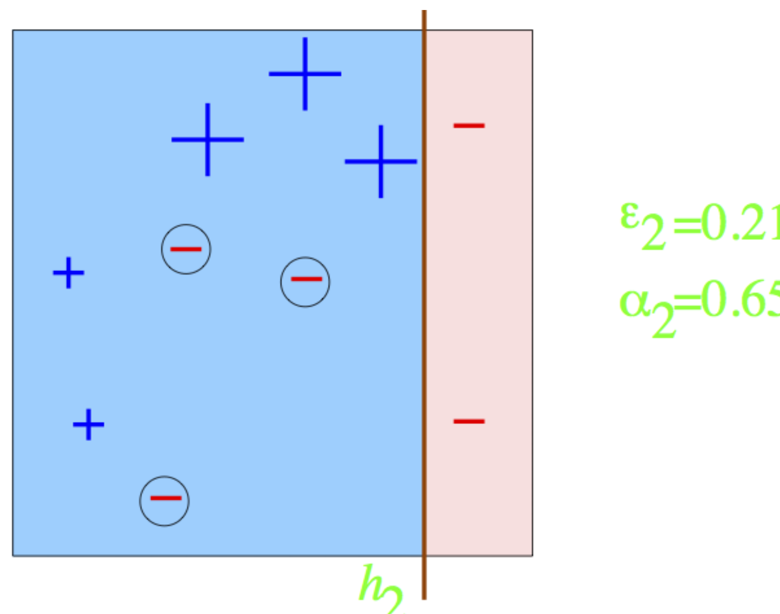
Boosting



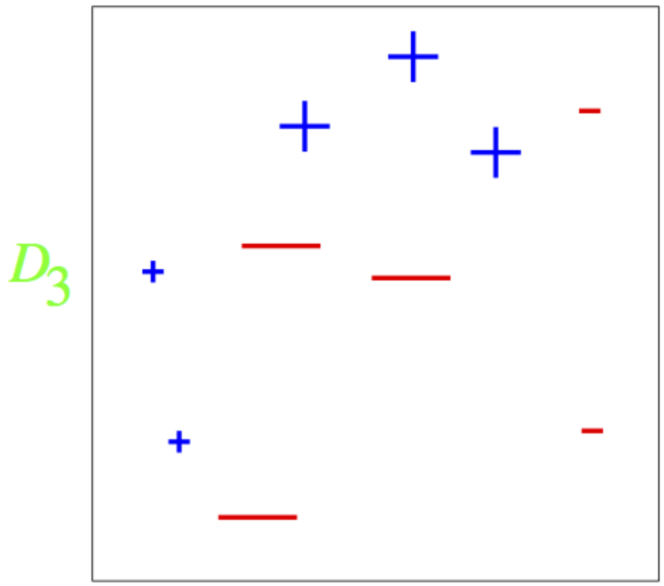
Boosting



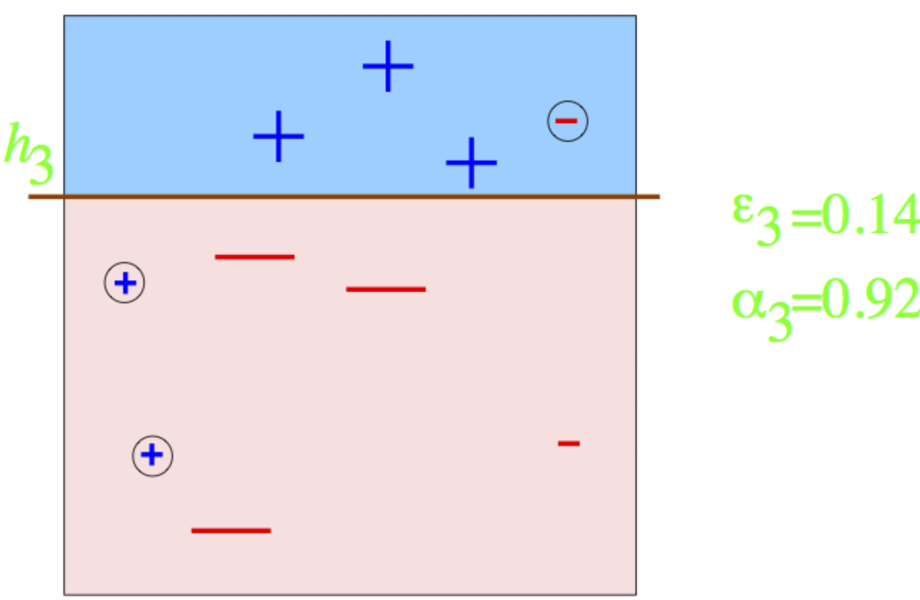
Boosting



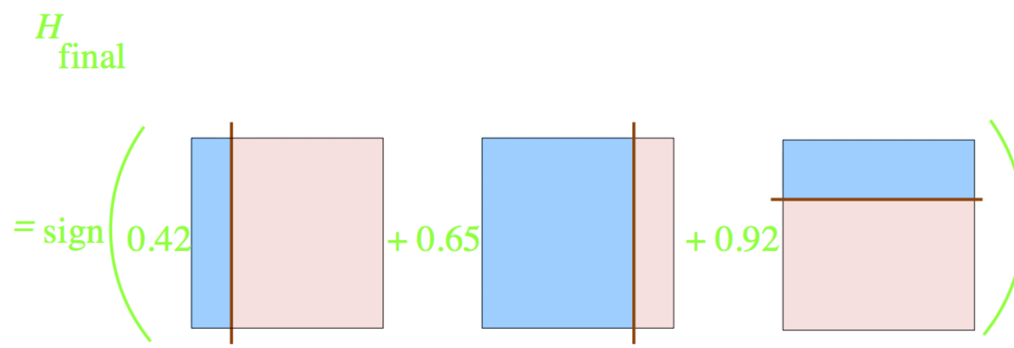
Boosting



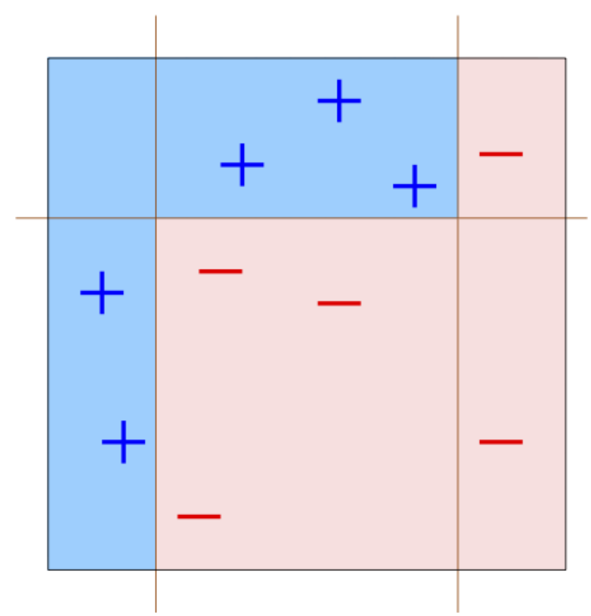
Boosting



Boosting



Boosting



Summary

- Variance in unstable models (e.g., decision trees) can be reduced using an ensemble — bagging
- Further refinement for decision tree bagging
  - Choose a random small subset of attributes to explore at each level
  - Random Forest
- Combining weak classifiers ("rules of thumb") — boosting

Market Basket Analysis

References

- *Bagging Predictors*, Leo Breiman, http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
- *Random Forests*, Leo Breiman and Adele Cutler, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- *A Short Introduction to Boosting*, Yoav Fruend and Robert E. Schapire, http://www.site.uottawa.ca/~stan/csi5387/boost-tut-ppr.pdf
- *AdaBoost and the Super Bowl of Classifiers A Tutorial Introduction to Adaptive Boosting*, Raúl Rojas, http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

Market Basket Analysis

- A shopping basket contains a set of items
- Analyze the content of a large number of shopping baskets
- Find associations—co-occurrence relationships
  - Customers who buy breakfast cereal often buy packed juice
- Express this as a rule

$$\text{Cereal} \longrightarrow \text{Juice}$$

- When is an association worth recording?
  - Need a minimum threshold of baskets containing cereal and juice — support
  - Of the baskets containing cereal, a reasonable fraction should contain juice — confidence

Market Basket Analysis

More formally ...

- $I = \{i_1, i_2, \ldots, i_m\}$ is a set of items
- $T = \{t_1, t_2, \ldots, t_n\}$ is a set of transactions
  - Each transaction t_i is a subset of I—an itemset
  - For an itemset X, $X.count$ is number of transactions in T containing X.
- An association rule is of the form $X \to Y$, where X and Y are itemsets

Support of a rule $X \to Y$

$$\frac{(X \cup Y).count}{n}$$

Confidence of a rule $X \to Y$

$$\frac{(X \cup Y).count}{X.count}$$

Example

Let T be as follows, with $\sigma = 0.3$, $\kappa = 0.7$

- Noodles, Biscuits, Milk
- Noodles, Cheese
- Cheese, Boots
- Noodles, Biscuits, Cheese
- Noodles, Biscuits, Detergent, Cheese, Milk
- Biscuits, Detergent, Milk
- Biscuits, Milk, Detergent

Some valid association rules

- Biscuits, Detergent $\to$ Milk [support 3/7, confidence 3/3]
- Noodles $\to$ Cheese [support 3/7, confidence 3/4].

Mining association rules

Given

- Items I
- Transactions T
- Minimum support threshold σ
- Minimum confidence threshold κ

Objective

Find all association rules with support at least σ and confidence at least κ

- Fixing σ, κ uniquely fixes the set of valid rules
- Association rule mining is complete and exact

Computing association rules

Basic strategy

- Generate all frequent itemsets (support above σ)
- Among these, identify valid rules (confidence above κ)

Brute force is infeasible, even if we restrict to items appearing in T

- ℓ items $\to 2^\ell$ candidate itemsets

How many itemsets can be frequent?

- Suppose 10^6 items, 10^8 transactions with 10 items each, $\sigma = 0.01$
- At most 1000 frequently appearing items!
  - A frequent item must appear in $10^6 = 0.01 \times 10^8$ baskets
  - Number of distinct items bounded by $10^9 = 10 \times 10^8$

A priori algorithm

Key insight

If an itemset X is frequent, so is every subset Y of X

If Y is not frequent and $Y \subset X$, X cannot be frequent

A priori algorithm

- Compute frequent itemsets level by level
- Scan T to identify F_1, frequent itemsets of size 1
- Candidate itemsets of size 2, $C_2 = F_1 \times F_1$
- Scan T to identify $F_2 \subseteq C_2$
- Compute C_3 such that all 2-subsets are in F_2
- Scan T to identify $F_3 \subseteq C_3$
- ...

A priori algorithm ...

Generating candidate set C_{k+1} from F_k

- Assume I is ordered as $i_1 < i_2 < \cdots$
- Sort each $X \in F_k$ according to this ordering
- Include $Y = \{i_1, i_2, \ldots, i_{k-1}, i_k, i_{k+1}\}$ in C_{k+1} if
  - $Y_1 = \{i_1, i_2, \ldots, i_{k-1}, i_k\}$
  - $Y_2 = \{i_1, i_2, \ldots, i_{k-1}, i_{k+1}\}$

  both belong to F_k
- Compute in single scan of F_k, using sliding window
- Conservative approximation to exact C_{k+1}

A priori algorithm ...

- Computing F_k from C_k involves one scan of T
  - Maintain an incremental count for each $X \in C_k$
- Bottleneck is computing C_{k+1} from F_k
- Naive strategy
  - Enumerate all $k{+}1$-subsets of I and
  - check which ones have all k-subsets in F_k
- Infeasible, both in terms of time and space

A priori algorithm ...

When do we stop?

- Transaction size is an upper bound on size of frequent itemset
- Before this bound, stop if $F_k = \emptyset$ for some k
- In practice, may only want small itemsets, so impose a bound

From frequent itemsets to rules

Let F be the set of frequent itemsets

Naive strategy

- For each X in F
  - Split X in all possible ways as $X_\ell \uplus X_r$
  - Check confidence of rule $X_\ell \rightarrow X_r$

$$\frac{X.count}{X_\ell.count} \geq \kappa$$

Can we be more efficient?

From frequent itemsets to rules ...

Consider candidate rules for $X \in F$

- $(X \setminus \{x\}) \rightarrow \{x\}$
- $(X \setminus \{x, y\}) \rightarrow \{x, y\}$

Clearly

- $(X \setminus \{x\}).count \leq (X \setminus \{x, y\}).count$

Hence

- $\dfrac{X.count}{(X \setminus \{x\}).count} \geq \dfrac{X.count}{(X \setminus \{x, y\}).count}$

Use a-priori again!

- For $(X \setminus \{x, y\}) \rightarrow \{x, y\}$ to be a valid rule, both $(X \setminus \{x\}) \rightarrow \{x\}$ and $(X \setminus \{y\}) \rightarrow \{y\}$ must be valid