Lecture 10: 12 February, 2026

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–April 2025

Bayesian classifiers

- As before
  - Attributes $\{A_1, A_2, \ldots, A_k\}$ and
  - Classes $C = \{c_1, c_2, \ldots c_\ell\}$
- Each class c_i defines a probabilistic model for attributes
  - $Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_i)$
- Given a data item $d = (a_1, a_2, \ldots, a_k)$, identify the best class c for d
- Maximize $Pr(C = c_i \mid A_1 = a_1, \ldots, A_k = a_k)$

Generative models

- To use probabilities, need to describe how data is randomly generated
  - Generative model
- Typically, assume a random instance is created as follows
  - Choose a class c_j with probability $Pr(c_j)$
  - Choose attributes $a_1, \ldots, a_k$ with probability $Pr(a_1, \ldots, a_k \mid c_j)$
- Generative model has associated parameters $\theta = (\theta_1, \ldots, \theta_m)$
  - Each class probability $Pr(c_j)$ is a parameter
  - Each conditional probability $Pr(a_1, \ldots, a_k \mid c_j)$ is a parameter
- We need to estimate these parameters

Bayesian classification

- Maximize $Pr(C = c_i \mid A_1 = a_1, \ldots, A_k = a_k)$
- By Bayes' rule,

$$Pr(C = c_i \mid A_1 = a_1, \ldots, A_k = a_k)$$

$$= \frac{Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_i) \cdot Pr(C = c_i)}{Pr(A_1 = a_1, \ldots, A_k = a_k)}$$

$$= \frac{Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_i) \cdot Pr(C = c_i)}{\sum_{j=1}^{\ell} Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_j) \cdot Pr(C = c_j)}$$

- Denominator is the same for all c_i, so sufficient to maximize

$$Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_i) \cdot Pr(C = c_i)$$

Naïve Bayes classifier

- Strong simplifying assumption: attributes are pairwise independent

$$Pr(A_1 = a_1, \ldots, A_k = a_k \mid C = c_i) = \prod_{j=1}^{k} Pr(A_j = a_j \mid C = c_i)$$

  - $Pr(C = c_i)$ is fraction of training data with class c_i
  - $Pr(A_j = a_j \mid C = c_i)$ is fraction of training data labelled c_i for which $A_j = a_j$

- Final classification is

$$\arg\max_{c_i} \; Pr(C = c_i) \prod_{j=1}^{k} Pr(A_j = a_j \mid C = c_i)$$

Naïve Bayes classifier ...

- Conditional independence is not theoretically justified
- For instance, text classification
  - Items are documents, attributes are words (absent or present)
  - Classes are topics
  - Conditional independence says that a document is a set of words: ignores sequence of words
  - Meaning of words is clearly affected by relative position, ordering
- However, naive Bayes classifiers work well in practice, even for text classification!
  - Many spam filters are built using this model

Zero counts

- Suppose $A = a$ never occurs in the test set with $C = c$

Zero counts

- Suppose $A = a$ never occurs in the test set with $C = c$
- Setting $Pr(A = a \mid C = c) = 0$ wipes out any product $\prod_{i=1}^{k} Pr(A_i = a_i \mid C = c)$ in which this term appears

Zero counts

- Suppose $A = a$ never occurs in the test set with $C = c$
- Setting $Pr(A = a \mid C = c) = 0$ wipes out any product $\prod_{i=1}^{k} Pr(A_i = a_i \mid C = c)$ in which this term appears
- Assume A_i takes m_i values $\{a_{i1}, \ldots, a_{im_i}\}$

Zero counts

- Suppose $A = a$ never occurs in the test set with $C = c$
- Setting $Pr(A = a \mid C = c) = 0$ wipes out any product $\prod_{i=1}^{k} Pr(A_i = a_i \mid C = c)$ in which this term appears
- Assume A_i takes m_i values $\{a_{i1}, \ldots, a_{im_i}\}$
- "Pad" training data with one sample for each value a_j — m_i extra data items

Zero counts

- Suppose $A = a$ never occurs in the test set with $C = c$
- Setting $Pr(A = a \mid C = c) = 0$ wipes out any product $\prod_{i=1}^{k} Pr(A_i = a_i \mid C = c)$ in which this term appears
- Assume A_i takes m_i values $\{a_{i1}, \ldots, a_{im_i}\}$
- "Pad" training data with one sample for each value a_j — m_i extra data items
- Adjust $Pr(A_i = a_i \mid C = c_j)$ to $\dfrac{n_{ij} + 1}{n_j + m_i}$ where
  - n_{ij} is number of samples with $A_i = a_i$, $C = c_j$
  - n_j is number of samples with $C = c_j$

Smoothing

- Laplace's law of succession

$$Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + 1}{n_j + m_i}$$

Smoothing

- Laplace's law of succession

$$Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + 1}{n_j + m_i}$$

- More generally, Lidstone's law of succession, or smoothing

$$Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda m_i}$$

Smoothing

- Laplace's law of succession

$$Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + 1}{n_j + m_i}$$

- More generally, Lidstone's law of succession, or smoothing

$$Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda m_i}$$

- $\lambda = 1$ is Laplace's law of succession

Text classification

- Classify text documents using topics

Text classification

- Classify text documents using topics
- Useful for automatic segregation of newsfeeds, other internet content

Text classification

- Classify text documents using topics
- Useful for automatic segregation of newsfeeds, other internet content
- Training data has a unique topic label per document — e.g., Sports, Politics, Entertainment

Text classification

- Classify text documents using topics
- Useful for automatic segregation of newsfeeds, other internet content
- Training data has a unique topic label per document — e.g., Sports, Politics, Entertainment
- Want to use a naïve Bayes classifier

Text classification

- Classify text documents using topics
- Useful for automatic segregation of newsfeeds, other internet content
- Training data has a unique topic label per document — e.g., Sports, Politics, Entertainment
- Want to use a naïve Bayes classifier
- Need to define a generative model

Text classification

- Classify text documents using topics
- Useful for automatic segregation of newsfeeds, other internet content
- Training data has a unique topic label per document — e.g., Sports, Politics, Entertainment
- Want to use a naïve Bayes classifier
- Need to define a generative model
- How do we represent documents?

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$
- Each topic c has probability $Pr(c)$

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$ with respect to each $c_j \in C$

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$ with respect to each $c_j \in C$
- Generating a random document d
  - Choose a topic c with probability $Pr(c)$
  - For each $w \in V$, toss a coin, include w in d with probability $Pr(w \mid c)$

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$ with respect to each $c_j \in C$
- Generating a random document d
  - Choose a topic c with probability $Pr(c)$
  - For each $w \in V$, toss a coin, include w in d with probability $Pr(w \mid c)$
- $Pr(d \mid c) = \prod_{w_i \in d} Pr(w_i \mid c) \prod_{w_i \notin d} (1 - Pr(w_i \mid c))$

No reference to length of d

Set of words model

- Each document is a set of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Topics come from a set $C = \{c_1, c_2, \ldots, c_k\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$ with respect to each $c_j \in C$
- Generating a random document d
  - Choose a topic c with probability $Pr(c)$
  - For each $w \in V$, toss a coin, include w in d with probability $Pr(w \mid c)$
- $Pr(d \mid c) = \prod_{w_i \in d} Pr(w_i \mid c) \prod_{w_i \notin d} (1 - Pr(w_i \mid c))$
- $Pr(d) = \sum_{c \in C} Pr(d \mid c)$

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C
- $Pr(c_j)$ is fraction of D labelled c_j

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C
- $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ is fraction of documents labelled c_j in which w_i appears

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C
- $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ is fraction of documents labelled c_j in which w_i appears
- Given a new document $d \subseteq V$, we want to compute $\arg\max_c Pr(c \mid d)$

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C
- $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ is fraction of documents labelled c_j in which w_i appears
- Given a new document $d \subseteq V$, we want to compute $\arg\max_c Pr(c \mid d)$
- By Bayes' rule, $Pr(c \mid d) = \dfrac{Pr(d \mid c)Pr(c)}{Pr(d)}$
  - As usual, discard the common denominator and compute $\arg\max_c Pr(d \mid c)Pr(c)$

Naïve Bayes classifier

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each $d_i \subseteq V$ is assigned a unique label from C
- $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ is fraction of documents labelled c_j in which w_i appears
- Given a new document $d \subseteq V$, we want to compute $\arg\max_c Pr(c \mid d)$
- By Bayes' rule, $Pr(c \mid d) = \dfrac{Pr(d \mid c)Pr(c)}{Pr(d)}$
  - As usual, discard the common denominator and compute $\arg\max_c Pr(d \mid c)Pr(c)$
- Recall $Pr(d \mid c) = \prod_{w_i \in d} Pr(w_i \mid c) \prod_{w_i \notin d} (1 - Pr(w_i \mid c))$

Bag of words model

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
  - Count multiplicities of each word

Universe U

Set $S \subseteq U$

$f_S : U \rightarrow \{0, 1\}$

char. fn

$f_S : U \rightarrow \mathbb{N}_0$

Bag of words model

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
  - Count multiplicities of each word
- As before
  - Each topic c has probability $Pr(c)$
  - Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$ with respect to each $c_j \in C$ (but we will estimate these differently)
  - Note that $\sum_{i=1}^{m} Pr(w_i \mid c_j) = 1$
  - Assume document length is independent of the class

Bag of words model

- Generating a random document d
  - Choose a document length ℓ with $Pr(\ell)$ Will disappear
  - Choose a topic c with probability $Pr(c)$
  - Recall $|V| = m$.
    - To generate a single word, throw an m-sided die that displays w with probability $Pr(w \mid c)$
    - Repeat ℓ times

Bag of words model

- Generating a random document d
  - Choose a document length ℓ with $Pr(\ell)$
  - Choose a topic c with probability $Pr(c)$
  - Recall $|V| = m$.
    - To generate a single word, throw an m-sided die that displays w with probability $Pr(w \mid c)$
    - Repeat ℓ times
- Let n_j be the number of occurrences of w_j in d

Bag of words model

- Generating a random document d
  - Choose a document length ℓ with $Pr(\ell)$
  - Choose a topic c with probability $Pr(c)$
  - Recall $|V| = m$.
    - To generate a single word, throw an m-sided die that displays w with probability $Pr(w \mid c)$
    - Repeat ℓ times
- Let n_j be the number of occurrences of w_j in d
- $Pr(d \mid c) = Pr(\ell)\ \ell! \prod_{j=1}^{m} \frac{Pr(w_j \mid c)^{n_j}}{n_j!}$

Parameter estimation

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each d_i is a multiset over V of size ℓ_i

Parameter estimation

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each d_i is a multiset over V of size ℓ_i Ignore
- As before, $Pr(c_j)$ is fraction of D labelled c_j

Parameter estimation

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each d_i is a multiset over V of size ℓ_i
- As before, $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ — fraction of occurrences of w_i over documents $D_j \subseteq D$ labelled c_j
  - n_{id} — occurrences of w_i in d
  - $Pr(w_i \mid c_j) = \dfrac{\sum_{d \in D_j} n_{id}}{\sum_{t=1}^{m} \sum_{d \in D_j} n_{td}}$

Parameter estimation

- Training set $D = \{d_1, d_2, \ldots, d_n\}$
  - Each d_i is a multiset over V of size ℓ_i
- As before, $Pr(c_j)$ is fraction of D labelled c_j
- $Pr(w_i \mid c_j)$ — fraction of occurrences of w_i over documents $D_j \subseteq D$ labelled c_j
  - n_{id} — occurrences of w_i in d
  - $$Pr(w_i \mid c_j) = \frac{\sum_{d \in D_j} n_{id}}{\sum_{t=1}^{m} \sum_{d \in D_j} n_{td}} = \frac{\sum_{d \in D} n_{id}\, Pr(c_j \mid d)}{\sum_{t=1}^{m} \sum_{d \in D} n_{td}\, Pr(c_j \mid d)},$$

fixed

1 only if $d \in D_j$

since $Pr(c_j \mid d) = \begin{cases} 1 & \text{if } d \in D_j, \\ 0 & \text{otherwise} \end{cases}$

Classification

- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$

Classification

- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$
- Want $\arg\max_{c}\ Pr(c \mid d)$

Classification

- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$
- Want $\arg\max_{c}\ Pr(c \mid d)$
- As before, discard the denominator $Pr(d)$

Classification

- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$
- Want $\arg\max_{c}\ Pr(c \mid d)$
- As before, discard the denominator $Pr(d)$
- Recall, $Pr(d \mid c) = Pr(\ell)\ \ell! \prod_{j=1}^{m} \dfrac{Pr(w_j \mid c)^{n_j}}{n_j!}$, where $|d| = \ell$

Classification

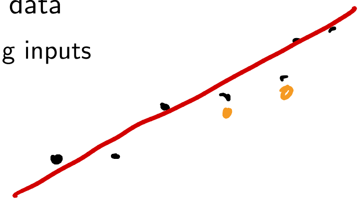
- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$
- Want $\arg\max_c\ Pr(c \mid d)$
- As before, discard the denominator $Pr(d)$
- Recall, $Pr(d \mid c) = Pr(\ell)\ \ell! \prod_{j=1}^{m} \dfrac{Pr(w_j \mid c)^{n_j}}{n_j!}$, where $|d| = \ell$
- Discard $Pr(\ell), \ell!$ since they do not depend on c

Classification

- $Pr(c \mid d) = \dfrac{Pr(d \mid c)\ Pr(c)}{Pr(d)}$
- Want $\arg\max_c\ Pr(c \mid d)$
- As before, discard the denominator $Pr(d)$
- Recall, $Pr(d \mid c) = Pr(\ell)\ \ell! \prod_{j=1}^{m} \dfrac{Pr(w_j \mid c)^{n_j}}{n_j!}$, where $|d| = \ell$
- Discard $Pr(\ell), \ell!$ since they do not depend on c
- Compute $\arg\max_c\ Pr(c) \prod_{j=1}^{m} \dfrac{Pr(w_j \mid c)^{n_j}}{n_j!}$

Limitations of classification models

- Bias : Expressiveness of model limits classification
  - For instance, linear logistic regression
- Variance: Variation in model based on sample of training data
  - Shape of a decision tree varies with distribution of training inputs



Limitations of classification models

- Bias : Expressiveness of model limits classification
  - For instance, linear logistic regression
- Variance: Variation in model based on sample of training data
  - Shape of a decision tree varies with distribution of training inputs

Bias - Variance tradeoff

Models with high variance are expressive but unstable

- In principle, a decision tree can capture an arbitrarily complex classification criterion
- Actual structure of the tree depends on impurity calculation
- Danger of overfitting: model tied too closely to training set

Limitations of classification models

- Bias : Expressiveness of model limits classification
  - For instance, linear logistic regression
- Variance: Variation in model based on sample of training data
  - Shape of a decision tree varies with distribution of training inputs

Models with high variance are expressive but unstable

- In principle, a decision tree can capture an arbitrarily complex classification criterion
- Actual structure of the tree depends on impurity calculation
- Danger of overfitting: model tied too closely to training set
- Is there an alternative to pruning?

Ensemble models

- Sequence of independent training data sets $D_1, D_2, \ldots, D_k$
- Generate models $M_1, M_2, \ldots, M_k$
- Take this ensemble of models and "average" them
  - For regression, take the mean of the predictions
  - For classification, take a vote among the results and choose the most popular one

Ensemble models

- Sequence of independent training data sets $D_1, D_2, \ldots, D_k$
- Generate models $M_1, M_2, \ldots, M_k$
- Take this ensemble of models and "average" them
  - For regression, take the mean of the predictions
  - For classification, take a vote among the results and choose the most popular one
- Challenge: Infeasible to get large number of independent training samples
- Can we build independent models from a single training data set?
  - Strategy to build the model is fixed
  - Same data will produce same model

Bootstrap Aggregating = Bagging

- Training data has N items
  - $TD = \{d_1, d_2, \ldots, d_N\}$
- Pick a random sample with replacement

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

Sample of size K

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

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%

Bootstrap Aggregating = Bagging

- Sample with replacement of size N : bootstrap sample
  - Approx 2/3 of full training data

Bootstrap Aggregating = Bagging

- Sample with replacement of size N : bootstrap sample
  - Approx 2/3 of full training data
- Take k such samples
- Build a model for each sample
  - Models will vary because each uses different training data

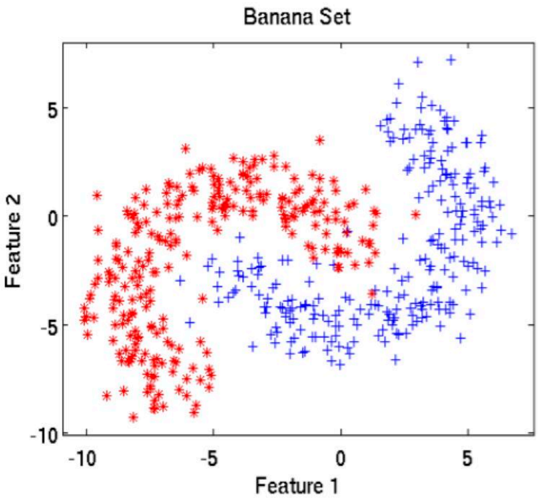
$D_1, \ldots, D_k$

Expect each original item to be in $\approx 2/3$ of D_i's

Bootstrap Aggregating = Bagging

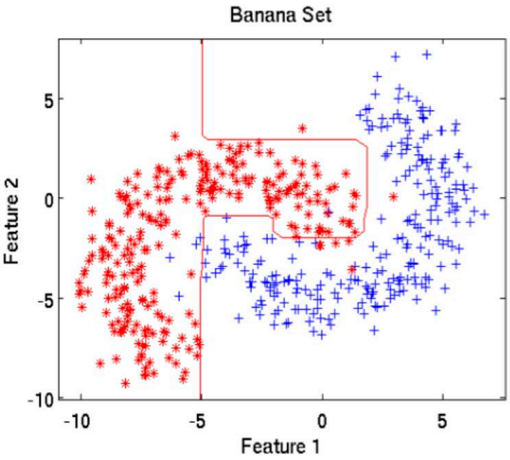
- Sample with replacement of size N : bootstrap sample
  - Approx 2/3 of full training data
- Take k such samples
- Build a model for each sample
  - Models will vary because each uses different training data
- Final classifier: report the majority answer
  - Assumptions: binary classifier, k odd
- Provably reduces variance

Bagging with decision trees



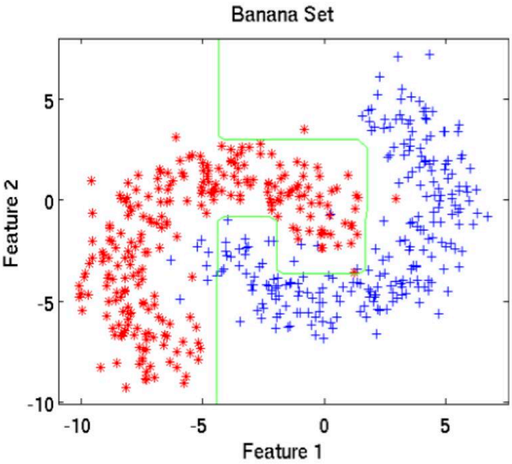
Training data

Bagging with decision trees



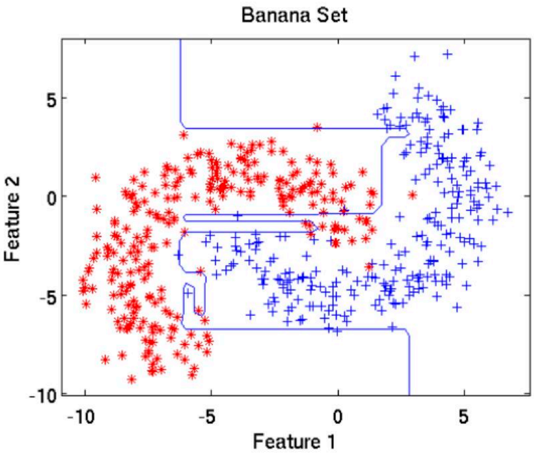
Decision boundary produced by one tree

Bagging with decision trees



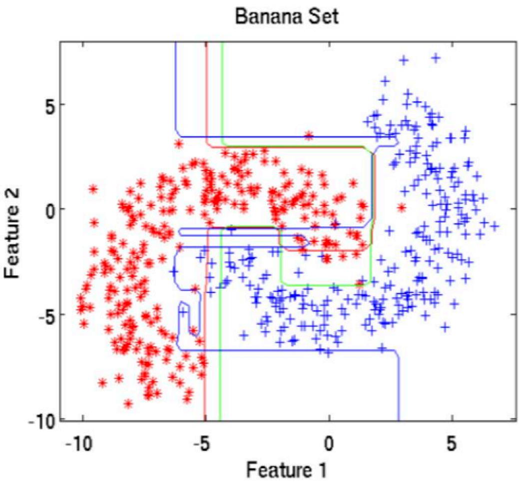
Decision boundary produced by a second tree

Bagging with decision trees



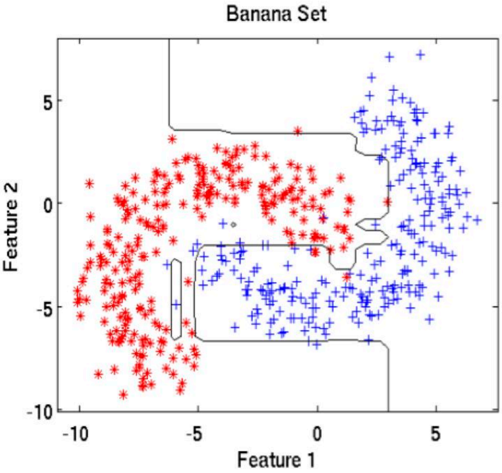
Decision boundary produced by a third tree

Bagging with decision trees



Three trees and final boundary overlaid

Bagging with decision trees



Final result from bagging all trees.

When to use bagging

- Bagging improves performance when there is high variance
  - Independent samples produce sufficiently different models

When to use bagging

- Bagging improves performance when there is high variance
  - Independent samples produce sufficiently different models
- A model with low variance will not show improvement
  - k-nearest neighbour classifier
  - Given an unknown input, find k nearest neighbours and choose majority
  - Across different subsets of training data, variation in k nearest neighbours is relatively small
  - Bootstrap samples will produce similar models

Random Forest

- Applying bagging to decision trees with a further twist

Graph theory

Forest = Collection of trees

Random Forest

- Applying bagging to decision trees with a further twist
- As before, k bootstrap samples $D_1, D_2, \ldots, D_k$

Random Forest

- Applying bagging to decision trees with a further twist
- As before, k bootstrap samples $D_1, D_2, \ldots, D_k$
- For each D_i, build decision tree T_i as follows
  - Each data item has M attributes
  - Normally, choose maximum impurity gain among M attributes, then best among remaining $M-1, \ldots$

Random Forest

- Applying bagging to decision trees with a further twist
- As before, k bootstrap samples $D_1, D_2, \ldots, D_k$
- For each D_i, build decision tree T_i as follows
  - Each data item has M attributes
  - Normally, choose maximum impurity gain among M attributes, then best among remaining $M-1, \ldots$
  - Instead, fix a small limit $m < M$ — say $m = \log_2 M + 1$
  - At each level, choose a random subset of available attributes of size m
  - Evaluate only these m attributes to choose next query

Random Forest

- Applying bagging to decision trees with a further twist
- As before, k bootstrap samples $D_1, D_2, \ldots, D_k$
- For each D_i, build decision tree T_i as follows
  - Each data item has M attributes
  - Normally, choose maximum impurity gain among M attributes, then best among remaining $M - 1, \ldots$
  - Instead, fix a small limit $m < M$ — say $m = \log_2 M + 1$
  - At each level, choose a random subset of available attributes of size m
  - Evaluate only these m attributes to choose next query
  - No pruning — build each tree to the maximum
- Final classifier: vote on the results returned by $T_1, T_2, \ldots, T_k$

Random Forest . . .

- Theoretically, overall error rate depends on two factors
  - Correlation between pairs of trees — higher correlation results in higher overall error rate
  - Strength (accuracy) of each tree — higher strength of individual trees results in lower overall error rate

Random Forest . . .

- Theoretically, overall error rate depends on two factors
  - Correlation between pairs of trees — higher correlation results in higher overall error rate
  - Strength (accuracy) of each tree — higher strength of individual trees results in lower overall error rate
- Reducing m, the number of attributes examined at each level, reduces correlation and strength
  - Both changes influence the error rate in opposite directions

m out of M features

Random Forest . . .

- Theoretically, overall error rate depends on two factors
  - Correlation between pairs of trees — higher correlation results in higher overall error rate
  - Strength (accuracy) of each tree — higher strength of individual trees results in lower overall error rate
- Reducing m, the number of attributes examined at each level, reduces correlation and strength
  - Both changes influence the error rate in opposite directions
- Increasing m increases both correlation and strength

Random Forest . . .

- Theoretically, overall error rate depends on two factors
  - Correlation between pairs of trees — higher correlation results in higher overall error rate
  - Strength (accuracy) of each tree — higher strength of individual trees results in lower overall error rate
- Reducing m, the number of attributes examined at each level, reduces correlation and strength
  - Both changes influence the error rate in opposite directions
- Increasing m increases both correlation and strength
- Search for a value of m that optimizes overall error rate

Out of bag error estimate

- Each bootstrap sample omits about $1/3$ of the data items

Out of bag error estimate

- Each bootstrap sample omits about $1/3$ of the data items
- Hence, each data item is omitted by about $1/3$ of the samples

Out of bag error estimate

- Each bootstrap sample omits about 1/3 of the data items
- Hence, each data item is omitted by about 1/3 of the samples
- If data item d does not appear in bootstrap sample D_i, d is out of bag (oob) for D_i

Out of bag error estimate

- Each bootstrap sample omits about $1/3$ of the data items
- Hence, each data item is omitted by about $1/3$ of the samples
- If data item d does not appear in bootstrap sample D_i, d is out of bag (oob) for D_i
- Oob classification — for each d, vote only among those T_i where d is oob for D_i

Out of bag error estimate

- Each bootstrap sample omits about 1/3 of the data items
- Hence, each data item is omitted by about 1/3 of the samples
- If data item d does not appear in bootstrap sample D_i, d is out of bag (oob) for D_i
- Oob classification — for each d, vote only among those T_i where d is oob for D_i
- Use oob samples to validate the model
  - Estimate generalization error rate of overall model based on error rate of oob classification
  - Do not require a separate test data set

Feature importance

- What is the impurity gain of a feature across trees in ensemble?

Feature importance

- What is the impurity gain of a feature across trees in ensemble?
- Variation due to randomness of samples

Feature importance

- What is the impurity gain of a feature across trees in ensemble?
- Variation due to randomness of samples
- Even greater variation in a random forest

Feature importance

- What is the impurity gain of a feature across trees in ensemble?
- Variation due to randomness of samples
- Even greater variation in a random forest
- Compute weighted average of impurity gain
  - Weight is given by number of training samples at the node

Regression

Discover line $l = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_m x_m$

Higher $\theta_i \Rightarrow$ higher impact of A_i

Requires normalization