Classification (continued from last week)

Naive Bayes classifier

- Assume we have p input features $X_1, \dots, X_p$ where the possible values for X_j are $\{ 1, \dots, q_j \}$ for some (small) number q_j of distinct values
- There are $|\mathcal{X}| = \prod_{j=1}^{p} q_j$ possible inputs we may need to classify
- To determine an arbitrary distribution over $\mathcal{X}$, or an arbitrary conditional distribution $P(Y \mid X)$, we would need $|\mathcal{X}| - 1$ parameters (since probabilities sum to one but can otherwise be chosen freely to each $x \in \mathcal{X}$)
- In many realistic scenarios, $|\mathcal{X}|$ is much more than the sample size, so learning such a distribution is out of the question — we need to make some simplifying assumptions

Naive Bayes classifier (2)

- **Naive Bayes** assumption is that input features are conditionally independent given class:

$$P(X_1, \dots, X_p \mid Y) = P(X_1 \mid Y) \dots P(X_p \mid Y)$$

- Each $P(X_i \mid Y)$ is determined by $q_i - 1$ (free) parameters

- For k classes, the number of parameters is $k \sum_{j=1}^{p} (q_i - 1) \ll k(\prod_{j=1}^{p} q_i - 1)$

Conditional independence

- Classical example used to illustrate conditional independence (and also difference between correlation and causation) is correlation between ice cream sales and drowning deaths
- During sunny and warm weather people tend to both eat ice cream and go boating, swimming etc. which increases chances of drowning
- Hence, there is positive correlation between ice cream sales and number of drownings on a given day
- However, if we already know what the weather actually was, then knowing how much ice cream was sold does not help us predict drowning
- Hence, ice cream sales and drownings are *conditionally* independent given weather

Learning a naive Bayes model

- Assume there are k classes $1, \ldots, k$ and p input features where for $j = 1, \ldots, p$ feature X_j has range $\{ 1, \ldots, q_j \}$
- We model $P(X \mid Y = c)$ separately for each class c and feature $X \in \{ X_1, \ldots, X_p \}$:
  - For each $c \leq k$, $j \leq d$, and $x \leq q_j$, let $n_{c,j,x}$ be the number of examples in the training data in class c with feature value $X_j = x$, and $n_c = \sum_{x=1}^{q_j} n_{c,j,x}$
  - We estimate

$$P(X_j = x \mid Y = c) = \frac{n_{c,j,x} + m_{c,j,x}}{n_c + m_{c,j}}$$

    where $m_{c,j,x}$ is a prior pseudocount and $m_{c,j} = \sum_{x=1}^{q_j} m_{c,j,x}$
  - Usual choices for pseudocounts are $m_{c,j,x} = 0$ (maximum likelihood), $m_{c,j,x} = 1$ (Laplace smoothing), and $m_{c,j,x} = 1/2$ (Krichevsky-Trofimov; my favourite!)

Predicting with naive Bayes

- Given an instance $\mathbf{x} = (x_1, \ldots, x_p)$, we use the estimates from previous slide to write

  $$P(X = \mathbf{x} \mid Y = c) = P(X_1 = x_1 \mid Y = c) \ldots P(X_p = x_p \mid Y = c)$$

  for all $c \in \{1, \ldots, k\}$, which gives us, via Bayes theorem

  $$P(Y = c \mid X = \mathbf{x}) = \frac{P(X = \mathbf{x} \mid Y = c) P(Y = c)}{\sum_{c'=1}^{k} P(X = \mathbf{x} \mid Y = c') P(Y = c')}$$

- The basic version of naive Bayes then predicts class c with maximum posterior probability (MAP):

  $$\hat{c}(\mathbf{x}) = \arg\max_{c} P(Y = c \mid X = \mathbf{x})$$

- Probabilistic predictions are obtained directly from $P(Y = c \mid X = \mathbf{x})$

Predicting with naive Bayes (2)

- Since the denominator $\sum_{c'=1}^{k} P(X = \mathbf{x} \mid Y = c')P(Y = c')$ does not depend on c, the MAP classification is the same as

$$\hat{c}(\mathbf{x}) = \arg\max_{c} P(X = \mathbf{x} \mid Y = c)P(Y = c)$$

- If the class prior $P(Y)$ is uniform, this simplifies to maximum likelihood (ML) prediction

$$\hat{c}(\mathbf{x}) = \arg\max_{c} P(X = \mathbf{x} \mid Y = c)$$

Predicting with naive Bayes (3)

- Substituting the product formula for $P(X = \mathbf{x} \mid Y = c)$ in the previous yields

$$\hat{c}(\mathbf{x}) = \arg\max_c \left[P(Y = c) \cdot \prod_{j=1}^{p} P(X_j = x_j \mid Y = c) \right]$$

- Taking log doesn't change the maximum, so

$$\hat{c}(\mathbf{x}) = \arg\max_c \left[\log P(Y = c) + \sum_{j=1}^{p} \log P(X_j = x_j \mid Y = c) \right]$$

- In other words, the score of each class is a sum composed of one term per feature: a class c for which $X_j = x_j$ is more likely gains more "points"

- Recall that the probabilities $P(X_j = x_j \mid Y = c)$ are estimated from training data (usually smoothed empirical frequencies)

About naive Bayes assumption

- The assumption that features are independent conditioned on class is
  - very strong
  - often quite untrue
- Therefore in particular the probabilities produced by a naive Bayes model should not be trusted too much
- However the classification performance (zero–one loss) of naive Bayes is often quite hard to beat in practice
- An informal justification for using naive Bayes is that often the data are collected in a way that aims to ensure (approximate) conditional independence
  - for example, in medical diagnosis, obtaining each feature requires that we carry out a test: it makes no sense to measure temperature from both armpits, or other redundant variables that we know to be strongly dependent (given the class)

NB for real-valued features

- When features x_i are real-valued, we can use the naive Bayes assumption as before, but now we deal with densities instead of discrete probabilities:

$$\begin{aligned} p(\mathbf{x} \mid Y=+1) &= p(X_1 = x_1 \mid Y=+1) \dots p(X_p = x_p \mid Y=+1) \\ p(\mathbf{x} \mid Y=-1) &= p(X_1 = x_1 \mid Y=-1) \dots p(X_p = x_p \mid Y=-1) \end{aligned}$$

- We model $p(X_j = x \mid Y = c)$ as a Gaussian separately for each feature X_j and class c

- As before, we (usually) choose the maximum a posteriori prediction

$$\hat{c}(\mathbf{x}) = \arg\max_c p(X = \mathbf{x} \mid Y = c) P(Y = c)$$

where $P(Y)$ is again the class prior

Gaussians for naive Bayes

- Consider binary classification with positive examples Tr^+ and negative examples Tr^- in training set
- Using maximum likelihood estimates for parameters, we get $p(X_j = x \mid Y = +1) = \mathcal{N}(x \mid \hat{\mu}_{+,j}, \hat{\sigma}^2_{+,j})$ where

$$\begin{aligned}\hat{\mu}_{+,j} &= \frac{1}{Tr^+} \sum_{x \in Tr^+} x_j \\ \hat{\sigma}^2_{+,j} &= \frac{1}{Tr^+} \sum_{x \in Tr^+} (x_j - \hat{\mu}_{+,j})^2\end{aligned}$$

  and similarly for negative examples

- The Gaussian assumption on $p(X_j = x_j \mid Y = c)$ leads to (a special case of) LDA/QDA! (Exercise)

Probabilistic models: summary

- Generative probabilistic models involve modelling both $P(X \mid Y = c)$ and $P(Y = c)$ for different classes c
- Important tools for this include
  - multivariate Gaussians (LDA, QDA): very important overall in statistics and machine learning, important to be familiar with them
  - Naive Bayes: especially discrete NB commonly used in practice, important to understand its uses and limitations
- Discriminative probabilistic learning aims directly at $P(Y = c \mid X)$.
  - Logistic regression is a good example

Probabilistic models in the textbook

- We have more or less covered Sec. 4 (“Classification”) except pages 145–149 (class-specific accuracy, ROC curves), including **logistic regression, LDA,** and **QDA**
- In addition, we discussed **Naive Bayes** which is required for this course (and the exam) but is not covered in the book at all
- Next up: **k-NN**, **decision trees**, and **SVM**

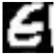
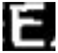
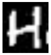
Classification: Discriminative methods

Similarity and dissimilarity

- Notions of similarity and dissimilarity between objects are important in machine learning
  - clustering tries to group similar objects together
  - many classification algorithms are based on the idea that similar objects tend to belong to same class
  - etc.

Similarity and dissimilarity (2)

- Examples: think about suitable similarity measures for
  - handwritten letters
  - segments of DNA
  - text documents

TCGATTGC ATCCTGTG ACCTGTCG

"Parliament overwhelmingly approved amendments to the Firearms Act on Wednesday. The new law requires physicians to inform authorities of individuals they consider unfit to own guns. It also increases the age for handgun ownership from 18 to 20."

"Parliament's Committee for Constitutional Law says that persons applying for handgun licences should not be required to join a gun club in the future. Government is proposing that handgun users be part of a gun association for at least two years."

"The cabinet on Wednesday will be looking at a controversial package of proposed changes in the curriculum in the nation's comprehensive schools. The most divisive issue is expected to be the question of expanded language teaching."

Similarity and dissimilarity (3)

- Similarity: s
  - Numerical measure of the degree to which two objects are *alike*
  - Higher for objects that are alike
  - Typically between 0 (no similarity) and 1 (completely similar)
- Dissimilarity: d
  - Numerical measure of the degree to which two objects are *different*
  - Higher for objects that are different
  - Typically between 0 (no difference) and ∞ (completely different)

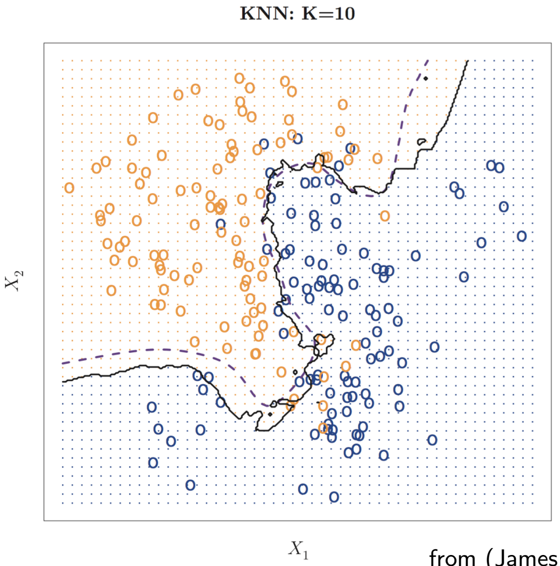
Nearest neighbour classifier

- Remember Problem 2.3?
- Nearest-neighbour classifier is a simple geometric model based on distances:
  - store all the training data
  - to classify a new data point, find the closest one in the training set and use its class
- More generally, *k*-nearest-neighbour classifier finds *k* nearest points in the training set and uses the majority class (ties are broken arbitrarily)
- Different notions of distance can be used, but Euclidean is the most obvious

Nearest neighbour classifier (2)

- Despite its utter simplicity, the k-NN classifier has remarkably good properties: both in theory and practice
- If $k = 1$, the error rate of 1-NN approaches $2\mathcal{E}_{\mathrm{Bayes}}$, where $\mathcal{E}_{\mathrm{Bayes}}$ is the **Bayes error**, i.e., the minimum achievable error by *any* classifier
- If $k \to \infty$ and $k/n \to 0$ as $n \to \infty$, then the error rate of k-NN approaches $\mathcal{E}_{\mathrm{Bayes}}$
- These properties hold under very mild assumptions on the data source $P(X, Y)$
- However, can be *veeeery* slow — as you may have noticed!
    - (But a nice application for *approximate* nearest neighbor search — point this out to Ville to make him happy!)

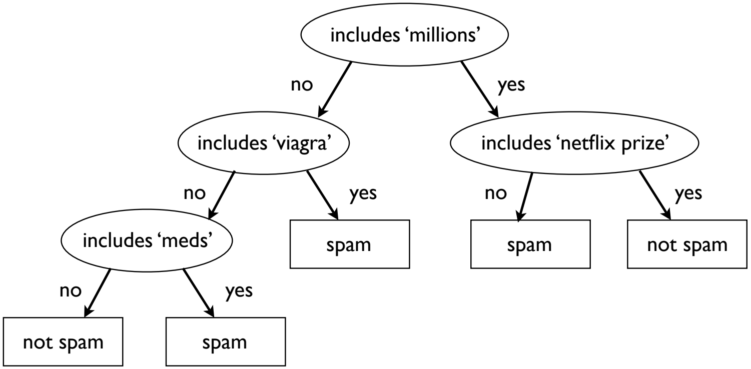
Nearest neighbour classifier (3)



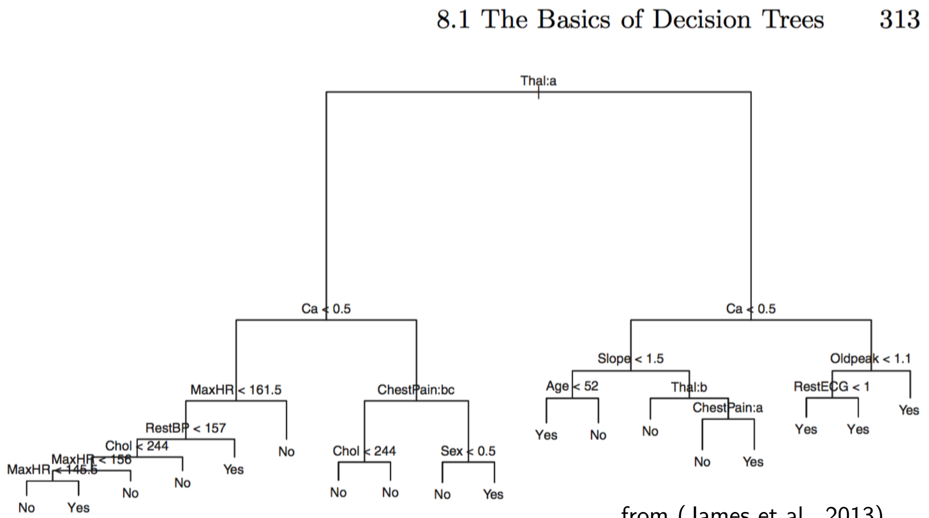
from (James et al., 2013)

Decision trees: An example

- Idea: Ask a sequence of questions to infer the class



Decision trees: A second example



from (James et al., 2013)

- In R: `library(tree)`

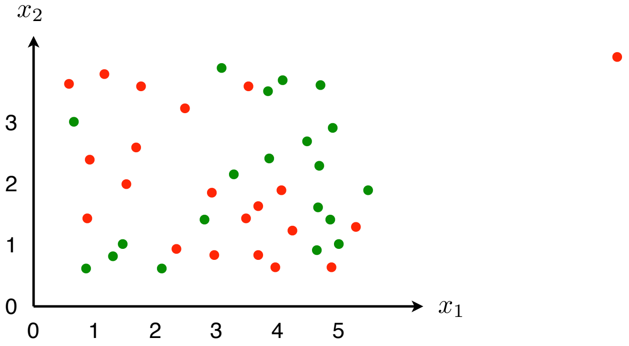
Decision trees: Structure

- Structure of the tree:
  - A single *root node* with no incoming edges, and zero or more outgoing edges (where edges go downwards)
  - *Internal nodes*, each of which has exactly one incoming edge and two or more outgoing edges
  - *Leaf* or *terminal* nodes, each of which has exactly one incoming edge and no outgoing edges

- Node contents:
  - Each terminal node is assigned a prediction (here, for simplicity: a definite class label).
  - Each non-terminal node defines a test, with the outgoing edges representing the various possible results of the test (here, for simplicity: a test only involves a single feature)

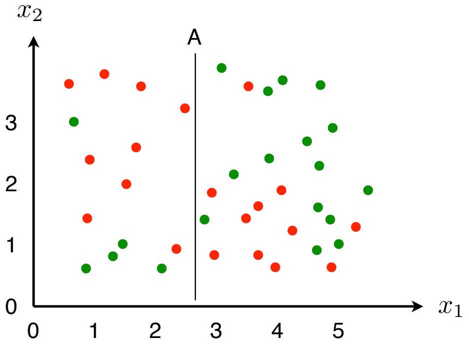
Learning a decision tree from data: General idea

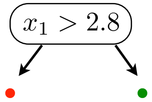
- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible



Learning a decision tree from data: General idea

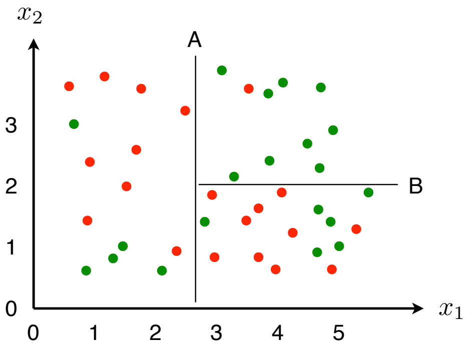
- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible

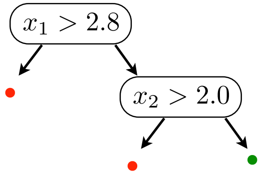




Learning a decision tree from data: General idea

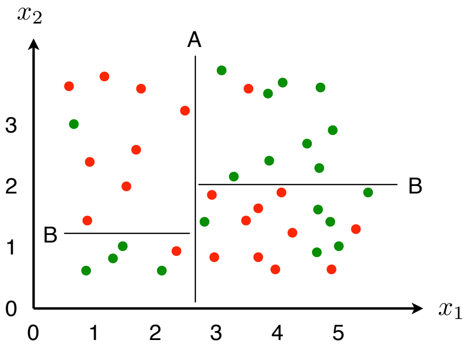
- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible

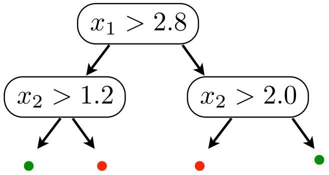




Learning a decision tree from data: General idea

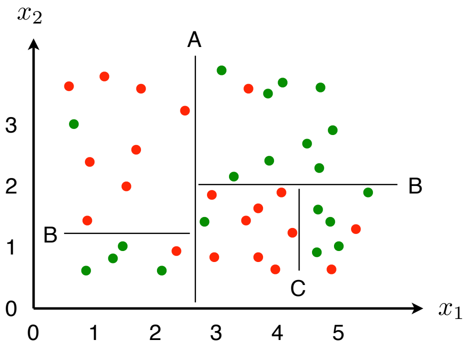
- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible

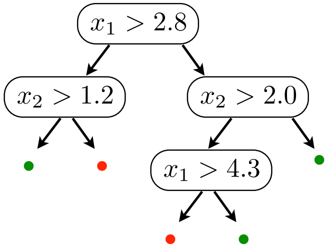




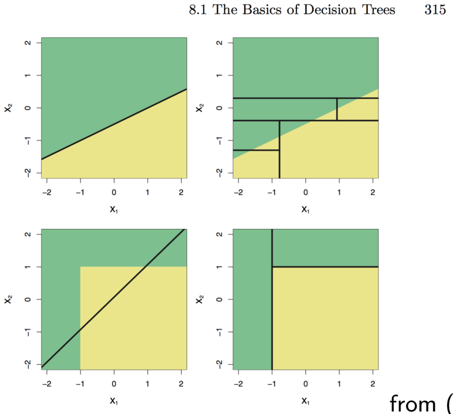
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible





Decision tree vs linear classifier (e.g., LDA)



from (James et al., 2013)

- Two continuous features X_1, X_2; Top: true model linear; Bottom: true model “rectangular”
- Left: linear classifier; Right: (small) decision tree

Learning a decision tree from data: basic algorithm

- Notation: Let D denote the set of examples corresponding to a node t. For the root node, D is the set of all training data.
- Basic algorithm:
  1. `if` there are less than $n_{\max}$ examples in D, `then` t is a leaf node
  2. `else` *Select a feature test* that partitions D into two subsets. Create a child node for each subset.
  3. Repeat recursively to each child node.

Regression trees

- The textbook (Sec. 8) first presents *regression trees* and then *classification trees*
- The idea in regression trees is to predict a continuous outcome Y by a representative (usually the average) outcome $\bar{y}$ within each leaf
- Otherwise, the idea is the same
- You should (have already) read the textbook to learn more
- We cover classification trees a some more detail than the book: e.g., splitting criteria and pruning are only superficially defined in the book

Feature test conditions

- Binary features: yes / no (left / right)
- Nominal (categorical) features with L values:
  - Pick one value (left) vs other (right) — can also do arbitrary subsets
- Ordinal features with L states:
  - Split at a given value: $X_i \leq T$ (left) vs $X_i > T$ (right)
- Continuous features:
  - Same as ordinal
- NB: We could also use multiway (not only binary) splits

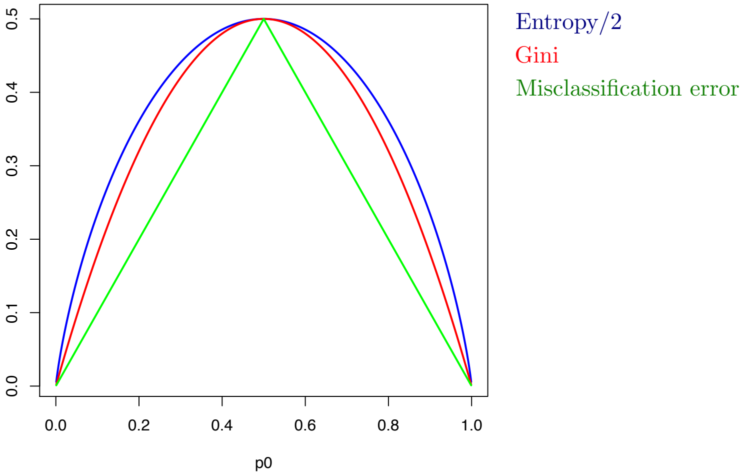
Impurity measures

Key part in choosing test for a node is *purity* of a subset D of training data

- Suppose there are K classes. Let $\hat{p}_{mc}$ be the fraction of examples of class c among all examples in node m:
- Impurity measures:

$$\begin{aligned}
\text{Classification error } E &= 1 - \max_c \hat{p}_{mc} \\
\text{(Cross) entropy } D &= -\sum_{c=1}^{K} \hat{p}_{mc} \log \hat{p}_{mc} \\
\text{Gini index } G &= \sum_{c=1}^{K} \hat{p}_{mc}(1 - \hat{p}_{mc})
\end{aligned}$$

Impurity measures: Binary classification



- The three measures are somewhat similar, but the textbook recommends either Gini or the cross entropy

Selecting the best split

- Let $Q(D)$ be impurity of data set D
- Assume a given feature test splits D into subsets D_1 and D_2
- We define the impurity of this split as

$$Q(\{ D_1, D_2 \}) = \sum_{i=1}^{2} \frac{|D_i|}{|D|} Q(D_i)$$

  and the related *gain* as

$$Q(D) - Q(\{ D_1, D_2 \})$$

- Choose the split with the highest gain

Avoiding over-fitting

- The tree building process that splits until nodes have size $n_{\max}$ (e.g., 5), will overfit
- It would be possible to stop recursion once the subset D is "pure enough". This is known as *pre-pruning* but generally not recommended
- In contrast, *post-pruning* (called simply pruning in the textbook) is an additional step to simplify the tree *after* it has first been fully grown until $|D| \leq n_{\max}$
- Reduced error pruning is a commonly used post-pruning method
  - usually increases understandability to humans
  - typically increases generalisation performance

Reduced-error pruning

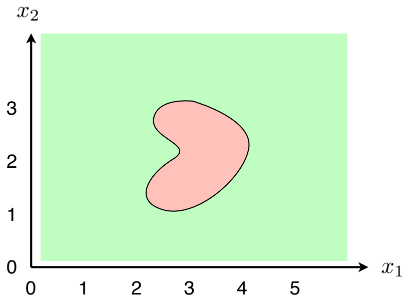
- Can be done by either withholding a part of the data as a test set or by cross-validation (CV)
- Supposing we choose to use a hold-out test set:

```
1. use the training set to build the full decision tree
2. for each tree size $N_m$ (# leaf nodes) in $1, \ldots, n/n_{\max}$:
3.     while # nodes is more than $N_m$:
4.         remove the bottom-level split that has the smallest gain
5.     evaluate the test set error of each pruned tree
6. choose the # nodes $N_m$ that minimizes the test error
```

- With CV, the same is repeated several times with different train–test splits and the test set error is averaged
- The tree building stage can be done using any of the impurity measures (misclass. rate, Gini, entropy), but in the pruning stage, Steps 4–5 almost invariably use misclassification rate

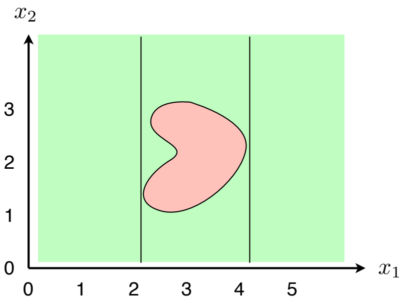
Properties of decision trees

- *Nonparametric* approach:
    - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
    - So with infinite data could in principle always learn the optimal classifier
    - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



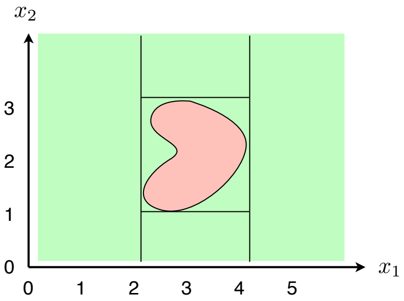
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



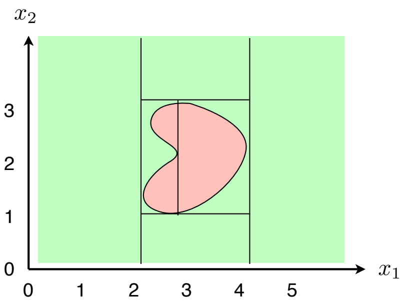
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



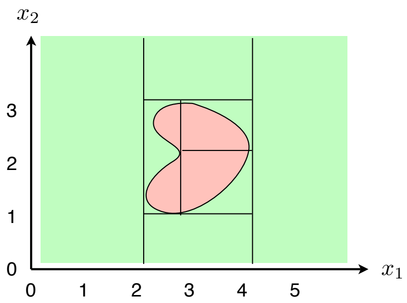
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



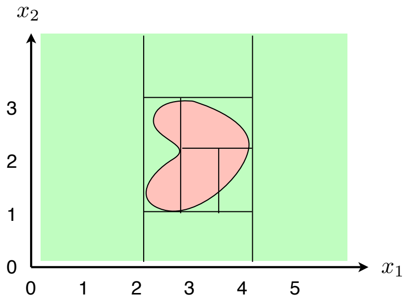
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like k-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



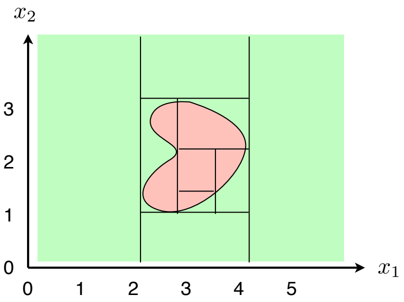
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



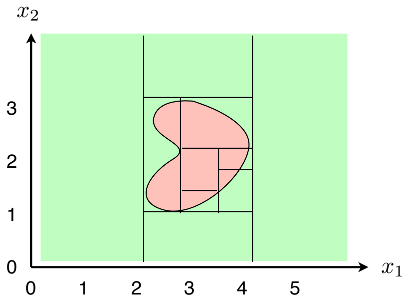
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



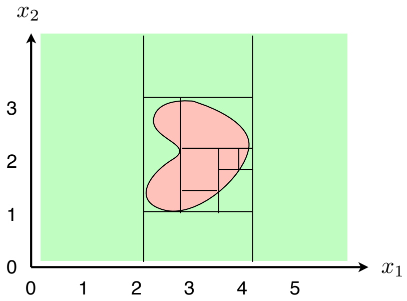
Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



Properties of decision trees

- *Nonparametric* approach:
  - If the tree size is unlimited, can in approximate *any* decision boundary to arbitrary precision (like *k*-NN)
  - So with infinite data could in principle always learn the optimal classifier
  - But with finite training data needs some way of avoiding overfitting (pruning) — bias–variance tradeoff!



Properties of decision trees (contd.)

- *Local, greedy learning* to find a reasonable solution in reasonable time (as opposed to finding a globally optimal solution)
- Relatively *easy to interpret* (by experts or regular users of the system)
- Classification generally very fast (linear in depth of the tree)
- Usually competitive only when combined with ensemble methods: bagging, random forests, boosting
- The textbook gives the impression that ensemble methods and decision trees belong together but we will consider ensemble methods more generally (in a week or two)