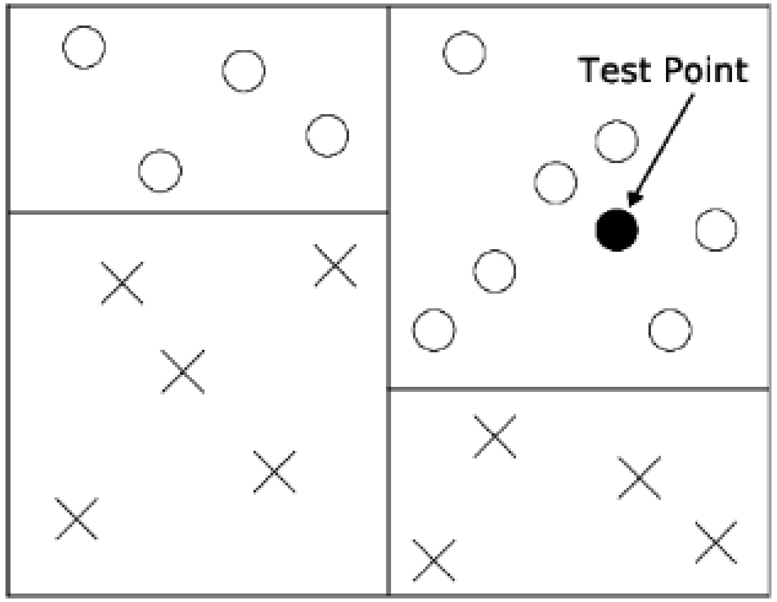
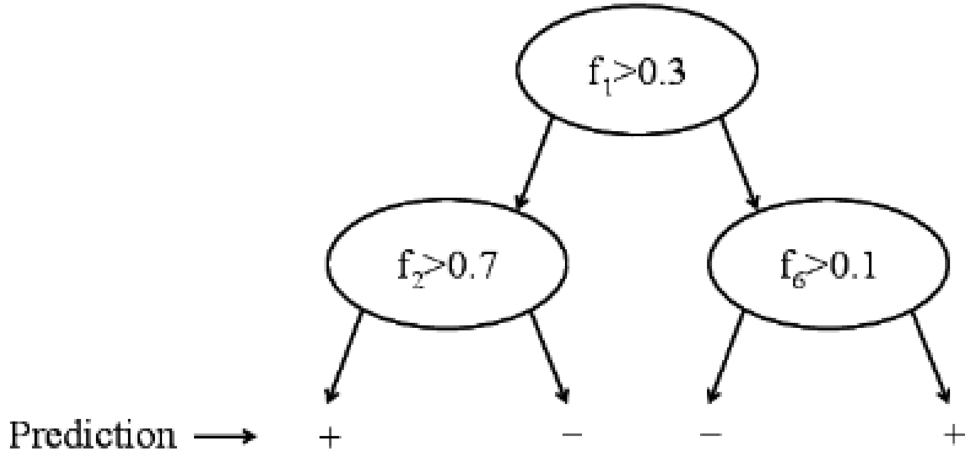
Decision trees

- Our ongoing assumption: **similar inputs have similar outputs**
- The way kNN reformulated this is **close points have similar labels**
- But defining labels in terms of training data points led to problems:
  - Memory consumption (at inference time)
  - Computational cost (at inference time)
  - Makes output dependent of the distance function, which then might trigger its own learning problem.
- NOTE: in this lecture we will assume that we are doing **classification**. Many of the results here can be also adapted to regression.

Decision trees (cont'd)

- A new formulation: **points from the same region of space have similar labels**
- Benefits:
    - We don't need to carry the data set
    - If we split space in an efficient way, the inference will be very fast
    - The various scales of features are captured in the way we split space, we don't need to learn a metric.


Test Point


f1>0.3
f2>0.7
f6>0.1
Prediction
+
−
−
+

Binary decision tree - outline of the training process

- We recursively divide the space into regions until a region has only one label.
- The root node represents the whole dataset
- The set is split roughly in half along one dimension using a threshold t on a single feature f
    - Left subtree $x_f \leq t$
    - Right subtree $x_f > t$
- We stop subdividing once a region has only one label, or if it cannot be subdivided (eg. there are two identical points with different labels)
    - This is a **pure leaf node**

Binary decision tree - inference

- For a datapoint with an unknown label $\boldsymbol{x} = (x_1, \ldots, x_n)$
- Start at the top of the tree
- Follow the branches down using the features
- Return the label associated with the leaf node we reached!

What does a decision tree remind us?

- What does a decision tree reminds us:
  - embedded collections of **if-then-else** statements...
  - in artificial intelligence: an **expert** system
  - **business processes**
  - **laws** and **regulations**
- Decision trees are a good match to the way in which humans **explain** the decisions they make
  - They often show up in **explainable artificial intelligence**
- By the way: it is not quite sure that this is indeed the way humans make decisions. It is likely more like a neural network.

Training a minimum size tree

- The critical aspect of decision tree training is how to pick the feature f and threshold t when we split a branch.
- Objective: find a maximally compact tree, which only has pure leaves
    - This is always possible if there are no two identical training datapoints with different labels
    - But it is an NP-hard problem!
    - Fortunately, it can be approximated well with a greedy algorithm.

Impurity functions

- Eventually, we want pure leaves, so our greedy approach will be to maximally increase purity at every split.
- $\mathcal{D} = \{(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_n, y_n)\}$, where $y_i \in \{1, 2, \ldots, c\}$ where c is the number of classes
- Gini impurity measure
  - $\mathcal{D}_k \in \mathcal{D}$ the subsets of the inputs where the label is k
  - $p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ - the fraction of inputs with label k

$$G(\mathcal{D}) = \sum_{k=1}^{c} p_k(1 - p_k)$$

Gini impurity of a tree

- Let us say that we are at a branch that splits the data into two subsets $\mathcal{D}_L$ and $\mathcal{D}_R$
- The Gini impurity of the tree will be:

$$G^T(\mathcal{D}) = \frac{|\mathcal{D}_L|}{|\mathcal{D}|} G^T(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{|\mathcal{D}|} G^T(\mathcal{D}_R)$$

- NOTE: there are other measures of impurity for classification (for instance, using entropy)

The ID3 algorithm

- "Iterative Dichotomiser 3" invented by Ross Quinlan
- Base cases (no further splits are needed):
  - If all datapoints have the same label y, return the label
  - If all datapoints have the same input $\boldsymbol{x}$, return the mode label (for classification) or median (for regression)
- Splitting:
  - Try all the features and all possible splits (it is a discrete number, it only makes sense to split between values)
  - Pick the split that minimizes impurity
- NOTE: even if a split does not improve the impurity, keep splitting until we reach a base case.

More advanced decision tree algorithms

- There are several more advanced decision tree algorithms that can:
  - handle a mix of discrete and continuous features
  - handle missing values
  - weights / costs on different features
  - perform pruning of the trees
  - identify and remove unhelpful attributes
- Examples C4.5, C5.0, See5, etc.
  - some of these are commercial
- Practical rule: use the most advanced algorithm to which you have access.

CART: classification and regression trees

- Assume continuous labels $y_i \in \mathbb{R}$
- Define the returned label as the average $\overline{y}_{\mathcal{D}}$

$$\overline{y}_{\mathcal{D}} = \frac{\sum_{i=1}^{|\mathcal{D}|} y_i}{|\mathcal{D}|}$$

- Define the impurity as the average squared difference from the label

$$\overline{L}_{\mathcal{D}} = \frac{\sum_{i=1}^{|\mathcal{D}|} (\overline{y}_{\mathcal{D}} - y_i)^2}{|\mathcal{D}|}$$

CART

- Split based on this definition of impurity
- Very cheap! (costs only O(n logn))
- Is it any good as a regressor or classifier?
  - Nope. Especially if they are shallow, they are sometimes referred as **weak learners**: just barely better than random guessing
- But they can become very strong using **ensemble methods** such as
  - **bagging** (Random Forests)
  - **boosting** (Gradient Boosted Trees, Adaboost etc.)