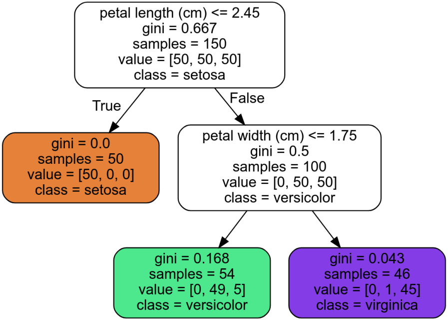
Tree-Based Learning Methods

Alessandro Leite

November 9th, 2019

Outline

Introduction



- **Decision tree** comprises a learning method for approximating discrete-valued target functions

[1] J. R. Quinlan. “Induction of Decision Trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is the decision tree learning?

- **Decision tree** comprises a learning method for approximating discrete-valued target functions
- It is widely used for **inductive inference**

[1] J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is the decision tree learning?

- **Decision tree** comprises a learning method for approximating discrete-valued target functions
- It is widely used for **inductive inference**
- Learned trees can be represented as a set of *if-then* rules

[1] J. R. Quinlan. “Induction of Decision Trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is the decision tree learning?

- **Decision tree** comprises a learning method for approximating discrete-valued target functions
- It is widely used for **inductive inference**
- Learned trees can be represented as a set of *if-then* rules
- Every finite algorithmic decision process can be modeled as a tree

[1] J. R. Quinlan. “Induction of Decision Trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is the decision tree learning?

- **Decision tree** comprises a learning method for approximating discrete-valued target functions
- It is widely used for **inductive inference**
- Learned trees can be represented as a set of *if-then* rules
- Every finite algorithmic decision process can be modeled as a tree
- Typical decision tree learning algorithms includes ID3[1] and C4.5[2]

[1] J. R. Quinlan. “Induction of Decision Trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is the decision tree learning?

- **Decision tree** comprises a learning method for approximating discrete-valued target functions
- It is widely used for **inductive inference**
- Learned trees can be represented as a set of *if-then* rules
- Every finite algorithmic decision process can be modeled as a tree
- Typical decision tree learning algorithms includes ID3[1] and C4.5[2]
- Each method searches a completely expressive hypothesis space

[1] J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**
- Examples

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**
- Examples
  - Medical or equipment diagnosis

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**
- Examples
  - Medical or equipment diagnosis
  - Credit risk analysis

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**
- Examples
  - Medical or equipment diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences

When to consider decision tree learning methods?

- **Instances are represented by attribute-value pairs** (e.g., hot, mild, cold)
- **Target function has discrete output values**
  - a decision tree in a cancer diagnosis assigns a boolean classification (e.g., malign, benign) to each example
- **Disjunctive hypothesis may be required**
- **Possibly noisy training data**
  - decision trees are robust to errors in classifications of the training examples and errors in the feature values that describe the examples
- **The training data may contain missing values**
- Examples
  - Medical or equipment diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences
  - Pattern recognition

Decision tree representation

- Decision trees **classify instances** by sorting them top-down

Decision tree representation

- Decision trees **classify instances** by sorting them top-down
- A **leaf** provides the classification of the instance

Decision tree representation

- Decision trees **classify instances** by sorting them top-down
- A **leaf** provides the classification of the instance
- A **node** specifies a test of some feature of the instance

Decision tree representation

- Decision trees **classify instances** by sorting them top-down
- A **leaf** provides the classification of the instance
- A **node** specifies a test of some feature of the instance
- A **branch** corresponds to a possible values a feature

Decision tree representation

- Decision trees **classify instances** by sorting them top-down
- A **leaf** provides the classification of the instance
- A **node** specifies a test of some feature of the instance
- A **branch** corresponds to a possible values a feature
- An **instance** is classified by starting at the **root node** of the tree, testing the feature specified by the node, then moving down the tree branch corresponding to the value of the feature in the given example

Decision tree representation

- Decision trees **classify instances** by sorting them top-down
- A **leaf** provides the classification of the instance
- A **node** specifies a test of some feature of the instance
- A **branch** corresponds to a possible values a feature
- An **instance** is classified by starting at the **root node** of the tree, testing the feature specified by the node, then moving down the tree branch corresponding to the value of the feature in the given example
- This **process** is then repeated for the subtree rooted at the new node

Decision tree representation (classification)

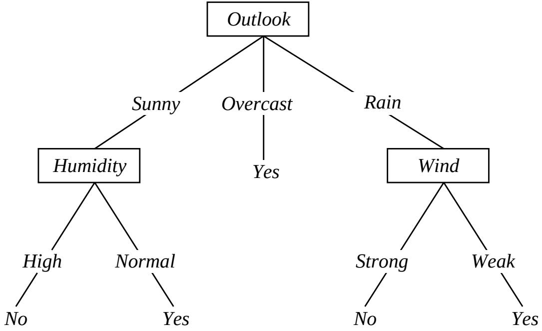


Figure 1: Classifying Saturday mornings according to whether they are suitable to play tennis or not

- We can represent this decision tree through the following logical expression

$$\begin{array}{cc} & (Outlook = Sunny \wedge Humidity = Normal) \\ \wedge & (Outlook = Overcast) \\ \wedge & (Outlook = Rain \wedge Wind = Weak) \end{array}$$

- Most of decision tree algorithms employ a **top-down, greedy search** through the space of possible **decision trees**[3] and its successor C4.5[4]
- ID3, learns decision trees by constructing them top-down, beginning with the question *"which feature should be tested at the root of the tree?"*

[3] J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (1986), pp. 81–106.

[4] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

What is a greedy search?

- At each step, make decision which makes greatest improvement in whatever you are trying to optimize

What is a greedy search?

- At each step, make decision which makes greatest improvement in whatever you are trying to optimize
- Does not backtrack, unless you hit a dead end

What is a greedy search?

- At each step, make decision which makes greatest improvement in whatever you are trying to optimize
- Does not backtrack, unless you hit a dead end
- This type of search is likely to not be a globally optimum solution, but it generally works well

What is a greedy search?

- At each step, make decision which makes greatest improvement in whatever you are trying to optimize
- Does not backtrack, unless you hit a dead end
- This type of search is likely to not be a globally optimum solution, but it generally works well
- At each node of the tree, make decision on which feature best classifies the training data at that point

What is a greedy search?

- At each step, make decision which makes greatest improvement in whatever you are trying to optimize
- Does not backtrack, unless you hit a dead end
- This type of search is likely to not be a globally optimum solution, but it generally works well
- At each node of the tree, make decision on which feature best classifies the training data at that point
- The end tree structure will represent a hypothesis, which works best for the training data

ID3 algorithm

- **Main loop**:
  1. $F \longleftarrow$ the "best" decision feature for next $node$
  2. Assign F as decision feature for $node$
  3. For each value of F, create new descendant of $node$
  4. Sort the training examples to leaf nodes
  5. If training examples perfectly classified, Then stop. Otherwise, iterate over new leaf nodes

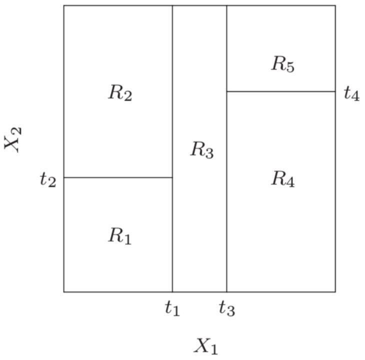
Choosing the feature for the root node

- The goal is to have the resulting decision tree as **small as possible** (Occam's Razor)
- The main decision in the algorithm is the selection of the next attribute to condition on (start from the root node)
- We want features that split the examples to sets that are relatively **pure** in one label; this way we are closer to a leaf node
- A node is **pure** if **all samples** at that node have the **same class label**
- The most popular heuristics is based on **information gain**, originated with the ID3 algorithm

Partition of of the future space of decision tree

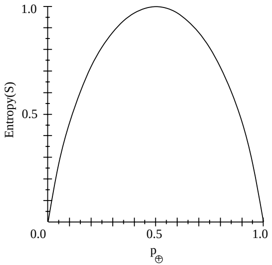
$$f(x) = \sum_{m=1}^{M} c_m I\left(x \in S\right)$$

- **regression**: c_m = average value in the region
- **classification**: c_m = majority vote in region



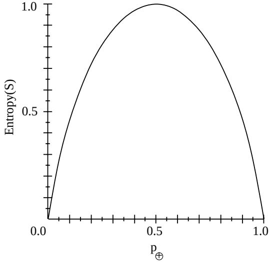
Entropy measures the homogeneity of the examples

- Given a sample S containing positive (+) and negative (-) examples of a target feature, and p_+ and p_- be the proportion of positive and negative examples in S



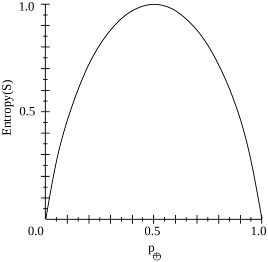
Entropy measures the homogeneity of the examples

- Given a sample S containing positive (+) and negative (-) examples of a target feature, and p_+ and p_- be the proportion of positive and negative examples in S
- Entropy of S is the expected number of bits needed to encode (+) or (-) classes of randomly drawn member of S



Entropy measures the homogeneity of the examples

- Given a sample S containing positive (+) and negative (-) examples of a target feature, and p_+ and p_- be the proportion of positive and negative examples in S
- Entropy of S is the expected number of bits needed to encode (+) or (-) classes of randomly drawn member of S
- The **entropy** measures the impurity of S



$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

- How to compute the entropy of a multi-class classification?

$$Entropy(S) = \sum_{i=1}^{|c|} -p_i \log_2 p_i$$

- where:
  - p_i is the proportion of S belong to class i
  - c is the number of different values that has class i

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature
- Formally, the information gain $Gain(S, F$ of feature F is:

$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{S} Entropy(S_v)$$

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature
- Formally, the information gain $Gain(S, F$ of feature F is:

$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{S} Entropy(S_v)$$

- where

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature
- Formally, the information gain $Gain(S, F$ of feature F is:

$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{S} Entropy(S_v)$$

- where
  - *Values(F)* is the set of all possible values for feature F

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature
- Formally, the information gain $Gain(S, F$ of feature F is:

$$Gain(S, F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{S} Entropy(S_v)$$

- where
  - *Values(F)* is the set of all possible values for feature F
  - S_v is the subset of S for which F has value v (i.e., $S_v = (\{s \in S | F(s) = v\}))$

Information gain measures the expected reduction in entropy

- Information gain gives the expected reduction in entropy caused by partitioning the examples according to a given feature
- Formally, the information gain $Gain(S, F$ of feature F is:

$$Gain(S, F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{S} Entropy(S_v)$$

- where
  - *Values(F)* is the set of all possible values for feature F
  - S_v is the subset of S for which F has value v (i.e., $S_v = (\{s \in S | F(s) = v\}))$
- $Gain(S, F)$ represents the expected reduction in entropy caused by knowing the value of feature F

Example

Day	Outlook	Temperature	Humidity	Wind	**Play Tennis**
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

- Gain(S,Outlook) = 0.246
- Gain(S,Humidity) = 0.151
- Gain(S,Wind) = 0.048
- Gain(S,Outlook) = 0.029

$$\begin{aligned} Entropy(S) &= Entropy([9+, 5-]) \\ &= \tfrac{-9}{14}\log_2 \tfrac{9}{14} - \tfrac{5}{14}\log_2 \tfrac{5}{14} \\ &= 0.94 \end{aligned}$$

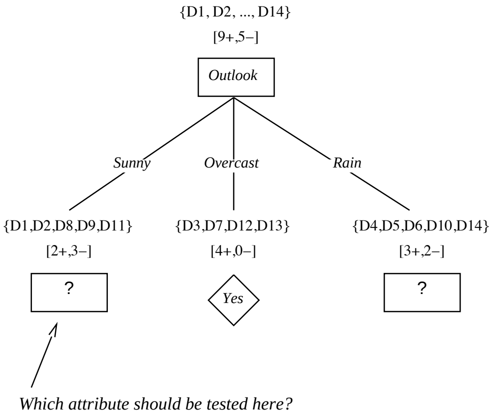
Gini impurity

- **Gini** is another metric to measure the impurity of a node

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

Hypothesis space search in decision tree



$S_{sunny} = \{D1, D2, D8, D9, D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Overfitting

- Consider the errors of a hypothesis h over:

Overfitting

- Consider the errors of a hypothesis h over:
  - training data: $error_{train}(h)$

Overfitting

- Consider the errors of a hypothesis h over:
  - training data: $error_{train}(h)$
  - entire data distribution $\mathcal{D}$: $error_{\mathcal{D}}(h)$

Overfitting

- Consider the errors of a hypothesis h over:
  - training data: $error_{train}(h)$
  - entire data distribution $\mathcal{D}$: $error_{\mathcal{D}}(h)$
- Hypothesis $h \in \mathcal{H}$ **overfits** training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that

  and

Overfitting

- Consider the errors of a hypothesis h over:
  - training data: $error_{train}(h)$
  - entire data distribution $\mathcal{D}$: $error_{\mathcal{D}}(h)$
- Hypothesis $h \in \mathcal{H}$ **overfits** training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that

  $$error_{train}(h) < error_{train}(h')$$

  and

Overfitting

- Consider the errors of a hypothesis h over:
  - training data: $error_{train}(h)$
  - entire data distribution $\mathcal{D}$: $error_{\mathcal{D}}(h)$
- Hypothesis $h \in \mathcal{H}$ **overfits** training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that

  $$error_{train}(h) < error_{train}(h')$$

  and

  $$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

How can we avoid overfitting?

- Stop growing when data split is not statistically significant

- Stop growing when data split is not statistically significant
- Grow the full tree, then **post-prune** it

How can we avoid overfitting?

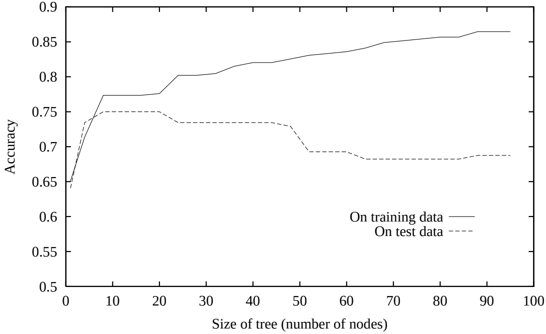
- Stop growing when data split is not statistically significant
- Grow the full tree, then **post-prune** it
- How do we select "best" tree?

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Grow the full tree, then **post-prune** it
- How do we select "best" tree?
  - Measure performance over training data

- Stop growing when data split is not statistically significant
- Grow the full tree, then **post-prune** it
- How do we select “best” tree?
  - Measure performance over training data
  - Measure performance over a separate validation data set

When to stop growing a tree?



- Strategies:
  - grow the tree until a minimum training points in the region is reached
  - prune the tree when the **cost-complexity** increases

When to stop growing a tree

- Cost-complexity pruning:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

- where:
  - T is the pruned tree
  - $|T|$ is the number of classes in T
  - N_m is the number of training samples in S
  - Q_m represents the error on S
  - α represents the trade-off between the model complexity and goodness fit

Advantages & issues of tree learning methods

- Decision trees are easy to explain

Advantages & issues of tree learning methods

- Decision trees are easy to explain
- Decision trees seem to mimic human-decision making process

Advantages & issues of tree learning methods

- Decision trees are easy to explain
- Decision trees seem to mimic human-decision making process
- Decision trees cab be displayed graphically and they can be easily interpreted

Advantages & issues of tree learning methods

- Decision trees are easy to explain
- Decision trees seem to mimic human-decision making process
- Decision trees cab be displayed graphically and they can be easily interpreted
- Decision trees can handle quantitative variables

- Decision trees are easy to explain
- Decision trees seem to mimic human-decision making process
- Decision trees cab be displayed graphically and they can be easily interpreted
- Decision trees can handle quantitative variables
- Decision trees handle multi-class problems naturally

Advantages & issues of tree learning methods

- Decision trees are easy to explain
- Decision trees seem to mimic human-decision making process
- Decision trees cab be displayed graphically and they can be easily interpreted
- Decision trees can handle quantitative variables
- Decision trees handle multi-class problems naturally
- Decision trees do not have very good predictive accuracy

Model evaluation metrics

- **Precision**, **recall**, and **F1-score** are performance metrics that can be used to measure a model's relevance

- **Precision**, **recall**, and **F1-score** are performance metrics that can be used to measure a model's relevance
- The performance of a model can be summarized by means of a **confusion matrix**

		Predicted class	
		+	−
Actual class	+	True Positives (TP)	False Negatives (FN)
	−	False Positives (FP)	True Negatives (TN)

- **Precision**, **recall**, and **F1-score** are performance metrics that can be used to measure a model's relevance
- The performance of a model can be summarized by means of a **confusion matrix**

		Predicted class	
		+	−
Actual class	+	True Positives (TP)	False Negatives (FN)
	−	False Positives (FP)	True Negatives (TN)

- Each row refers to actual classes recorded in the test set, and each column to classes as predicted by the predictor

Model evaluation metrics

- **Precision**, **recall**, and **F1-score** are performance metrics that can be used to measure a model's relevance
- The performance of a model can be summarized by means of a **confusion matrix**

		Predicted class	
		+	−
Actual class	+	True Positives (TP)	False Negatives (FN)
	−	False Positives (FP)	True Negatives (TN)

- Each row refers to actual classes recorded in the test set, and each column to classes as predicted by the predictor
- **False positives** represent **false alarms**, which are also known as **type I errors**

Model evaluation metrics

- **Precision**, **recall**, and **F1-score** are performance metrics that can be used to measure a model's relevance
- The performance of a model can be summarized by means of a **confusion matrix**

		Predicted class	
		+	–
Actual class	+	True Positives (TP)	False Negatives (FN)
	–	False Positives (FP)	True Negatives (TN)

- Each row refers to actual classes recorded in the test set, and each column to classes as predicted by the predictor
- **False positives** represent **false alarms**, which are also known as **type I errors**
- **False negatives** represent **misses classifications**, which are called **type II errors**

Computing precision, recall, and F1-score

- Prediction **error** (**ERR**) and **accuracy (ACC)** provide general information about how many samples are misclassified

Computing precision, recall, and F1-score

- **Specificity = True Negative Rate (TNR)**

$$TNR = \frac{TN}{FP + TN}$$

- **Precision = Positive Predictive Value (PPV)**

$$Precision = \frac{TP}{TP + FP}$$

- **F1-score** represents the harmonic mean of precision and sensitivity

$$F1 = \frac{2TP}{2TP + FP + FN}$$

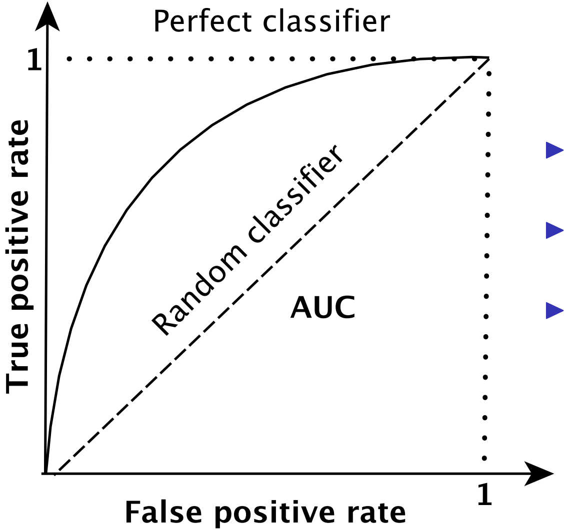
ROC curves

- **Receiver operator characteristic (ROC)** is a tool for selecting models for classification based on their performance with respect to the **false positive** and **true positive** rates

ROC curves

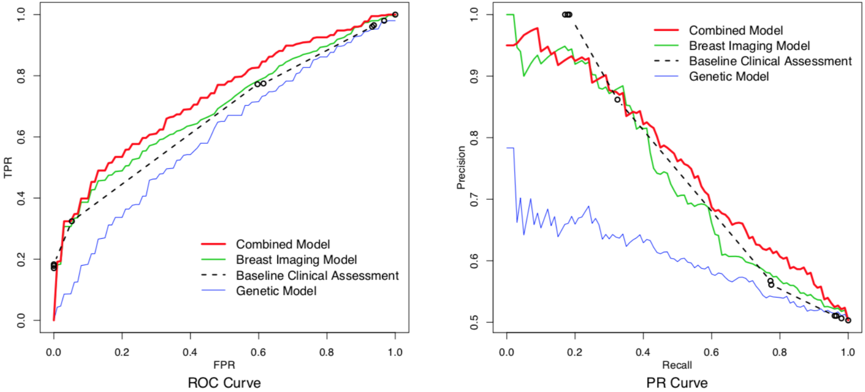
- **Receiver operator characteristic (ROC)** is a tool for selecting models for classification based on their performance with respect to the **false positive** and **true positive** rates
- The diagonal of an ROC plot can be interpreted as a random guessing

- **Receiver operator characteristic (ROC)** is a tool for selecting models for classification based on their performance with respect to the **false positive** and **true positive** rates
- The diagonal of an ROC plot can be interpreted as a random guessing
- It is summarized by the **area under the curve (AUC)**, which characterize the performance of a classification model



- **Perfect classifier**: AUC = 1.0
- **Random classifier**: AUC = 0.5
- **Our classifier**: $0.5 < AUC < 1.0$

Example: breast cancer risk prediction on mammograms

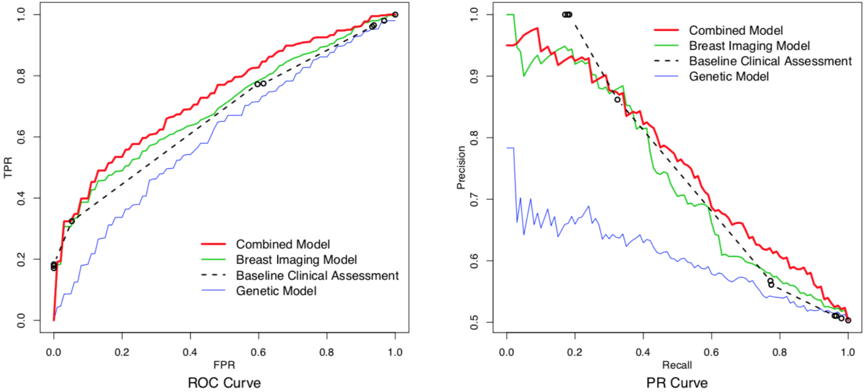


Predicting breast cancer risk based on mammography images. **Source**: Liu et al. (2013)[5]

- **High recall** means less chances to miss a case

[5] Jie Liu et al. "Genetic variants improve breast cancer risk prediction on mammograms". In: *Annual Symposium Proceedings*. Vol. 2013. 2013, p. 876.

Example: breast cancer risk prediction on mammograms

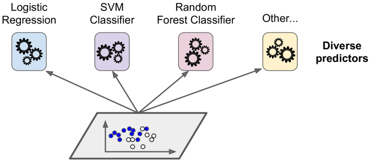


Predicting breast cancer risk based on mammography images. **Source**: Liu et al. (2013)[5]

- **High recall** means less chances to miss a case
- **High precision** means substantially more true diagnoses than false alarms

[5] Jie Liu et al. "Genetic variants improve breast cancer risk prediction on mammograms". In: *Annual Symposium Proceedings*. Vol. 2013. 2013, p. 876.

Building forests through ensemble learning approach

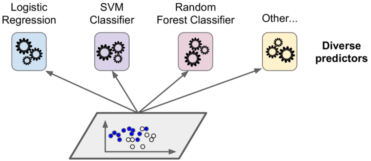


- **Main idea**: aggregating many weak learners can substantially increase their performance

[6] Leo Breiman. “Bagging Predictors”. In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. “The Strength of Weak Learnability”. In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

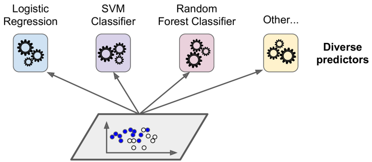


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

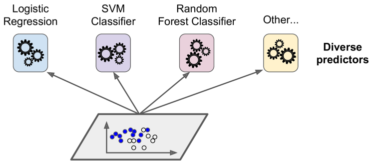


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:

[6] Leo Breiman. “Bagging Predictors”. In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. “The Strength of Weak Learnability”. In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

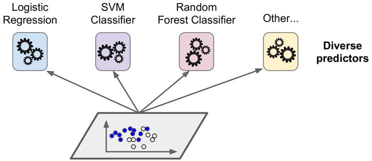


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

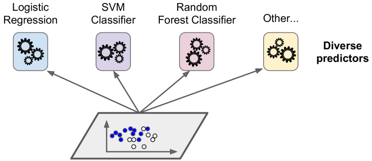


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data
    - Bagging[6]: bootstrap re-sampling

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

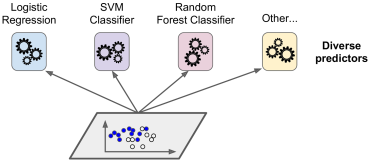


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data
    - Bagging[6]: bootstrap re-sampling
    - Boosting[7]: re-sample based on performance

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

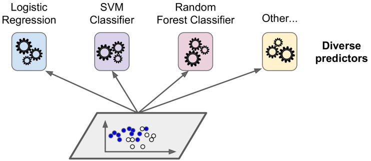


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data
    - Bagging[6]: bootstrap re-sampling
    - Boosting[7]: re-sample based on performance
  - using different features

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

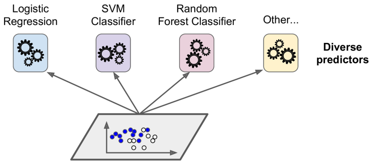


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data
    - Bagging[6]: bootstrap re-sampling
    - Boosting[7]: re-sample based on performance
  - using different features
    - future selection

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

Building forests through ensemble learning approach

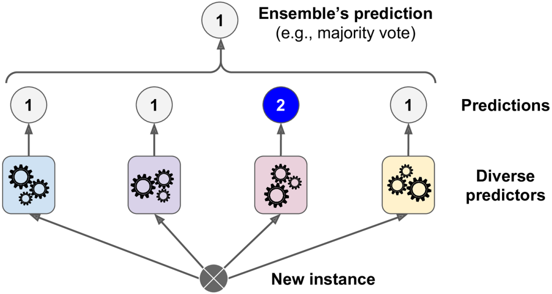


- **Main idea**: aggregating many weak learners can substantially increase their performance
- **Wisdom of crowds**: average the uncorrelated errors of individual classifiers
- Ensembles can be built by:
  - **sub-sampling** the training data
    - Bagging[6]: bootstrap re-sampling
    - Boosting[7]: re-sample based on performance
  - using different features
    - future selection
  - using different **parameters** of the **learning algorithm**

[6] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[7] Robert E. Schapire. "The Strength of Weak Learnability". In: *Machine Learning* 5.2 (1990), pp. 197–227.

How to combine multiple learners?



- **Non-trainable combination**
  - Voting (classification)
  - Averaging (regression)
- **Trainable combination**
  - Weighted averaging: based on the performance on a validation set
  - Meta-leaner: the outputs of individuals learners are features of another learning algorithm

Bagging Tree

- Take repeated samples from the training data (i.e., bootstrap)
- Build one predictor from each of these samples
- Compute the final prediction
- Bagging regression

$\textsc{Bagging}(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

$\quad$ **for** $t \leftarrow 1$ **to** T **do**

$\qquad S_t \leftarrow \textsc{Bootstrap}(S) \triangleright$ i.i.d. sampling with replacement from S.

$\qquad h_t \leftarrow \textsc{TrainRegressionAlgorithm}(S_t)$

$\quad$ **return** $h_S = x \mapsto \textsc{Mean}((h_1(x), \ldots, h_T(x)))$

- Bagging classification

$\textsc{Bagging}(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

$\quad$ **for** $t \leftarrow 1$ **to** T **do**

$\qquad S_t \leftarrow \textsc{Bootstrap}(S) \triangleright$ i.i.d. sampling with replacement from S.

$\qquad h_t \leftarrow \textsc{TrainClassifier}(S_t)$

$\quad$ **return** $h_S = x \mapsto \textsc{MajorityVote}((h_1(x), \ldots, h_T(x)))$

Random Forests

- Similar to bagging trees[8]
- Therefore, before splitting, first randomly sample q of p variables among the one over which to split must be chosen
- This trick help on decorrelating the trees
- q is usually $\sqrt{p}$
- Random forests presents a very good predictive power

[8] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.
[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data
- Decision trees can handle categorical and numerical variables

[9] Leo Breiman. “Bagging Predictors”. In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data
- Decision trees can handle categorical and numerical variables
- Decision trees results are easy to analyze and understand

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data
- Decision trees can handle categorical and numerical variables
- Decision trees results are easy to analyze and understand
- The limited predictive power of decision tree methods can be handled by ensemble methods such as

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

Summary

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data
- Decision trees can handle categorical and numerical variables
- Decision trees results are easy to analyze and understand
- The limited predictive power of decision tree methods can be handled by ensemble methods such as
  - Bagging[9]

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

- Decision trees are robust learning methods: they can handle noisy and missing data
- Decision trees can easily adapt to new data
- Decision trees can handle categorical and numerical variables
- Decision trees results are easy to analyze and understand
- The limited predictive power of decision tree methods can be handled by ensemble methods such as
  - Bagging[9]
  - Random Forests[10]

[9] Leo Breiman. "Bagging Predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

References

- J. R. Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (1986), pp. 81–106
- Tom M. Michell. *Machine Learning*. McGraw-Hill Education, 1997
- Hal Daume III. *A Course in Machine Learning*. 2nd. Self-published, 2017. URL: http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf
  1. **Decision trees**: chapter 1
  2. **Random forests**: session 13.3
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd. Springer, 2016. URL: https://web.stanford.edu/~hastie/Papers/ESLII.pdf
  1. **Decision trees**: session 9.2
  2. **Random forests**: sessions 15.1 and 15.2