Introduction to Data Analysis and Machine Learning in Physics: 4. Decisions Trees

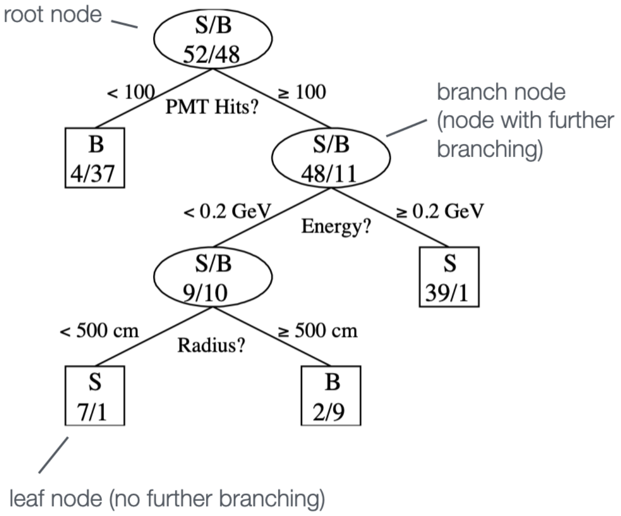
Martino Borsato, Jörg Marks, Klaus Reygers

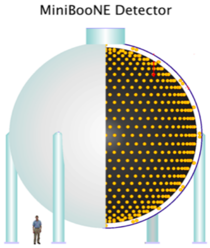
Studierendentage, 6-9 April 2021

Exercises

- Exercise 1: Compare different decision tree classifiers
  - `04_decision_trees_ex_1_compare_tree_classifiers.ipynb`
- Exercise 2: Apply XGBoost classifier to MAGIC data set
  - `04_decision_trees_ex_1_compare_tree_classifiers.ipynb`
- Exercise 3: Feature importance
- Exercise 4: Interpret a classifier with SHAP values

Decision trees

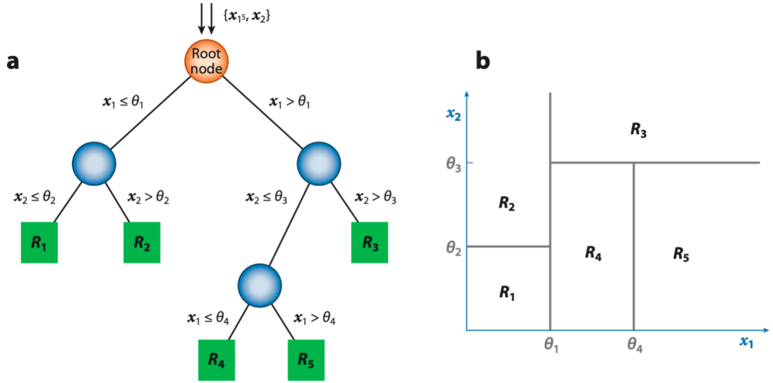




MiniBooNE: 1520 photomultiplier signals, goal: separation of ν_e from ν_μ events

Leaf nodes classify events as either signal or background

Decision trees: Rectangular volumes in feature space



- Easy to interpret and visualize: Space of feature vectors split up into rectangular volumes (attributed to either signal or background)
- How to build a decision tree in an optimal way?

Finding optimal cuts

Separation btw. signal and background is often measured with the Gini index (or Gini impurity):

$$G = p(1-p)$$

Here p is the purity:

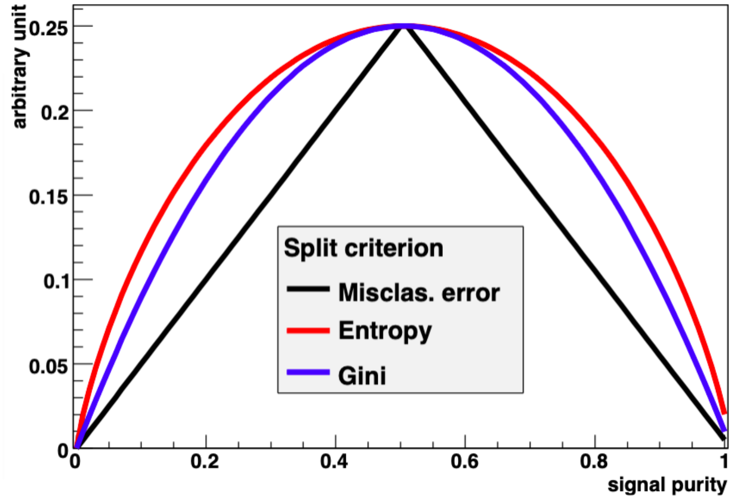
$$p = \frac{\sum_{\text{signal}} w_i}{\sum_{\text{signal}} w_i + \sum_{\text{background}} w_i}, \quad w_i = \text{weight of event } i$$

Usefulness of weights will become apparent soon.

Improvement in signal/background separation after splitting a set A into two sets B and C:

$$\Delta = W_A G_A - W_B G_B - W_C G_C \quad \text{where} \quad W_X = \sum_X w_i$$

Gini impurity and other purity measures



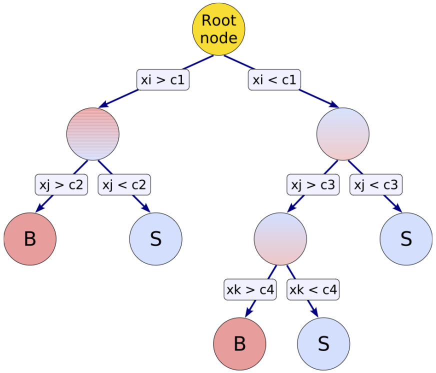
Decision tree pruning

When to stop growing a tree?

- When all nodes are essentially pure?
- Well, that's overfitting!

Pruning

- Cut back fully grown tree to avoid overtraining, i.e., replace nodes and subtrees by leaves



Single decision trees: Pros and cons

Pros:

- Requires little data preparation (unlike neural networks)
- Can use continuous and categorical inputs

Cons:

- Danger of overfitting training data
- Sensitive to fluctuations in the training data
- Hard to find global optimum
- When to stop splitting?

Ensemble methods: Combine weak learners

- Bootstrap Aggregating (Bagging)
  - Sample training data (with replacement) and train a separate model on each of the derived training sets
  - Classify example with majority vote, or compute average output from each tree as model output

$$y(\vec{x}) = \frac{1}{N_{\text{trees}}} \sum_{i=1}^{N_{trees}} y_i(\vec{x})$$

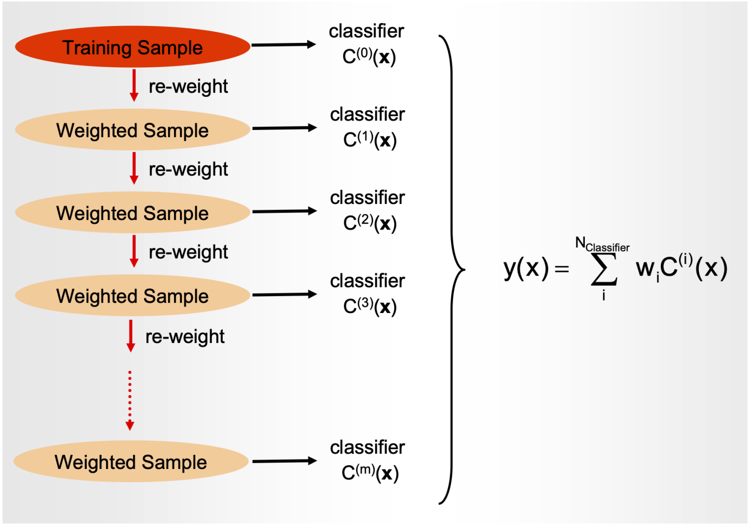
- Boosting
  - Train N models in sequence, giving more weight to examples not correctly classified by previous model
  - Take weighted average to classify examples

$$y(\vec{x}) = \frac{\sum_{i=1}^{N_{\text{trees}}} \alpha_i y_i(\vec{x})}{\sum_{i=1}^{N_{\text{trees}}} \alpha_i}$$

Random forests

- “One of the most widely used and versatile algorithms in data science and machine learning” arXiv:1803.08823v3
- Use bagging to select random example subset
- Train a tree, but only use random subset of features at each split
  - this reduces the correlation between different trees
  - makes the decision more robust to missing data

Boosted decision trees: Idea


Training Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
re-weight
Weighted Sample
classifier $C^{(0)}(\mathbf{x})$
classifier $C^{(1)}(\mathbf{x})$
classifier $C^{(2)}(\mathbf{x})$
classifier $C^{(3)}(\mathbf{x})$
classifier $C^{(m)}(\mathbf{x})$
$y(x) = \sum_{i}^{N_{\text{Classifier}}} w_i C^{(i)}(x)$

AdaBoost (short for Adaptive Boosting)

Initial training sample

$\vec{x}_1, \ldots, \vec{x}_n$:	multivariate event data
$y_1, \ldots, y_n$:	true class labels, $+1$ or -1
$w_1^{(1)}, \ldots, w_n^{(1)}$	event weights

with equal weights normalized as

$$\sum_{i=1}^{n} w_i^{(1)} = 1$$

Train first classifier f_1:

$f_1(\vec{x}_i) > 0$	classify as signal
$f_1(\vec{x}_i) < 0$	classify as background

AdaBoost: Updating events weights

Define training sample $k+1$ from training sample k by updating weights:

$$w_i^{(k+1)} = w_i^{(k)} \frac{e^{-\alpha_k f_k(\vec{x}_i) y_i / 2}}{Z_k}$$

$i =$ event index, $\quad Z_k$: normalization factor so that $\sum_{i=1}^{n} w_i^{(k)} = 1$

Weight is increased if event was misclassified by the previous classifier
$\rightarrow$ "Next classifier should pay more attention to misclassified events"

At each step the classifier f_k minimizes error rate:

$$\varepsilon_k = \sum_{i=1}^{n} w_i^{(k)} I(y_i f_k(\vec{x}_i) \leq 0), \quad I(X) = 1 \text{ if } X \text{ is true, 0 otherwise}$$

AdaBoost: Assigning the classifier score

Assign score to each classifier according to its error rate:

$$\alpha_k = \ln \frac{1 - \varepsilon_k}{\varepsilon_k}$$

Combined classifier (weighted average):

$$f(\vec{x}) = \sum_{k=1}^{K} \alpha_k f_k(\vec{x})$$

Gradient boosting

Basic idea:

- Train a first decision tree
- Then train a second one on the residual errors made by the first tree
- And so on

In slightly more detail:

- Consider labeled training data: $\{\vec{x}_i, y_i\}$
- Model prediction at iteration *m*: $F_m(\vec{x}_i)$
- New model: $F_{m+1}(\vec{x}) = F_m(\vec{x}) + h_m(\vec{x})$
- Find $h_m(\vec{x})$ by fitting it to $\{(\vec{x}_1, y_1 - F_m(\vec{x}_1)),\ (\vec{x}_2, y_2 - F_m(\vec{x}_2)),\ \dots\ (\vec{x}_n, y_n - F_m(\vec{x}_n))\}$

Example 1: Predict critical temperature for superconductivty (Regression with XGBoost) (1)

04_decision_trees_critical_temp_regression.ipynb

Superconductivty data set:
Predict the critical temperature based on 81 material features.
https://archive.ics.uci.edu/ml/datasets/Superconductivty+Data

From the abstract:
We estimate a statistical model to predict the superconducting critical temperature based on the features extracted from the superconductor's chemical formula. The statistical model gives reasonable out-of-sample predictions: $\pm 9.5\,\mathrm{K}$ based on root-mean-squared-error. Features extracted based on thermal conductivity, atomic radius, valence, electron affinity, and atomic mass contribute the most to the model's predictive accuracy.

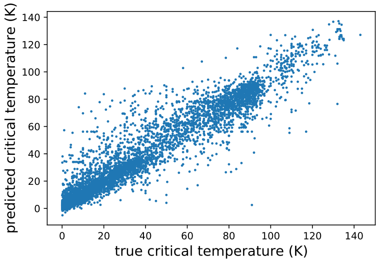
https://doi.org/10.1016/j.commatsci.2018.07.052

Example 1: Predict critical temperature for superconductivty (Regression with XGBoost) (2)

```
import xgboost as xgb

XGBreg = xgb.sklearn.XGBRegressor()

XGBreg.fit(X_train, y_train)

y_pred = XGBreg.predict(X_test)

from sklearn.metrics import mean_squared_error
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"root mean square error {rms:.2f}")
```

This gives:

```
root mean square error 9.68
```



Exercise 1: Compare different decision tree classifiers

04_decision_trees_ex_1_compare_tree_classifiers.ipynb

Compare scikit-learns's `AdaBoostClassifier`, `RandomForestClassifier`, and `GradientBoostingClassifier` by plotting their ROC curves for the heart disease data set.

Is there a classifier that clearly performs best?

Exercise 2: Apply XGBoost classifier to MAGIC data set

04_decision_trees_ex_2_magic_xgboost_and_random_forest.ipynb

```
# train XGBoost boosted decision tree
import xgboost as xgb
XGBclassifier = xgb.sklearn.XGBClassifier(nthread=-1, seed=1, n_estimators=1000)
```

a) Plot predicted probabilities for the test sample for signal and background events (`plt.hist`)

b) Which is the most important feature for discriminating signal and background according to XGBoost? Hint: use plot_impartance from XGBoost (see XGBoost plotting API). Do you get the same answer for all three performance measures provided by XGBoost ("weight", "gain", or "cover")?

c) Visualize one decision tree from the ensemble (let's say tree number 10). For this you need the the graphviz package (`pip3 install graphviz`)

d) Compare the performance of XGBoost with the **random forest classifier** from **scikit learn**. Plot signal and background efficiency for both classifiers in one plot. Which classifier performs better?

Exercise 3: Feature importance

04_decision_trees_ex_3_magic_feature_importance.ipynb

Evaluate the importance of each of the n features in the training of the XGBoost classifier for the MAGIC data set by dropping one of the features. This gives n different classifiers. Compare the performance of these classifiers using the AUC score.

Exercise 4: Interpret a classifier with SHAP values

SHAP (SHapley Additive exPlanations) are a means to explain the output of any machine learning model. Shapeley values are a concept that is used in cooperative game theory. They are named after Lloyd Shapley who won the Nobel Prize in Economics in 2012.

Use the Python library `SHAP` to quantify the feature importance.

a) Study the documentation at `https://shap.readthedocs.io/en/latest/tabular_examples.html`
b) Create a summary plot of the feature importance in the MAGIC data set with `shap.summary_plot` for the XGBoost classifier of exercise 2. What are the three most important features?
c) Do the same for the superconductivity data set? What are the three most important features?