Gradient Boosted Regression Trees
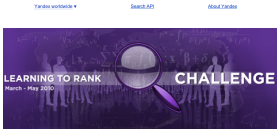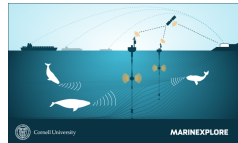
Peter Prettenhofer (@pprett)
DataRobot

Gilles Louppe (@glouppe)
Université de Liège, Belgium

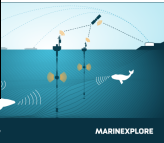# Motivation

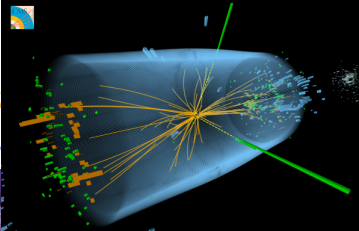# Motivation

Outline

About us

Peter

- @pprett
- Python & ML $\sim$ 6 years
- sklearn dev since 2010

Gilles

- @glouppe
- PhD student (Liège, Belgium)
- sklearn dev since 2011
  Chief tree hugger
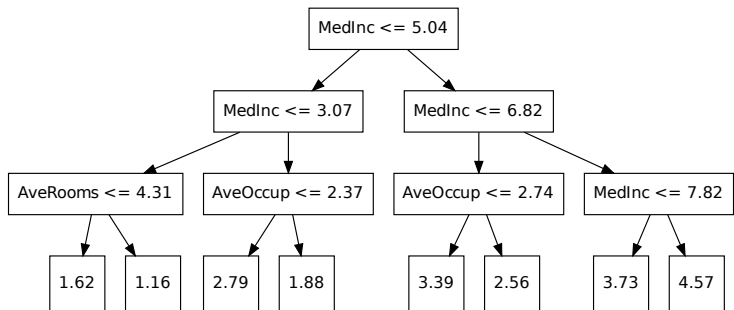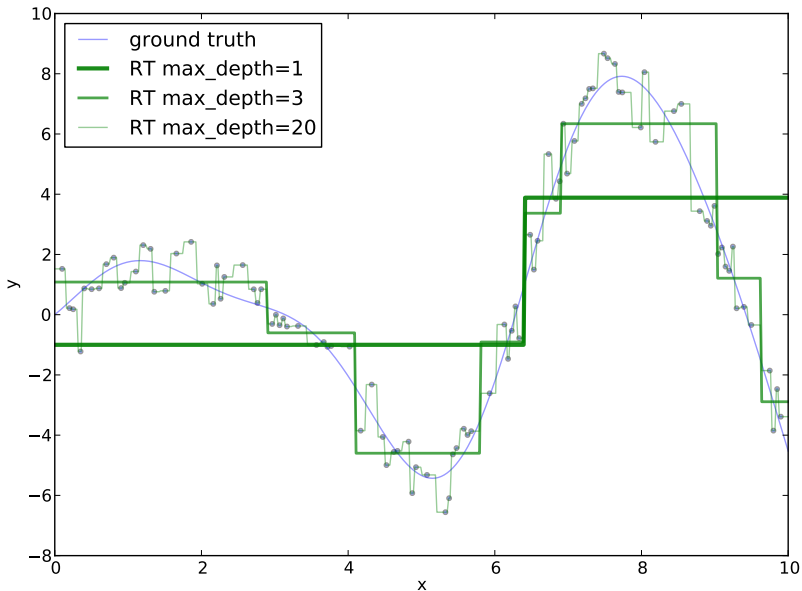
Outline

Machine Learning 101

- Data comes as...
  - A set of examples $\{(\mathbf{x}_i, y_i) | 0 \leq i < \text{n\_samples}\}$, with
  - Feature vector $\mathbf{x} \in \mathbb{R}^{\text{n\_features}}$, and
  - Response $y \in \mathbb{R}$ (regression) or $y \in \{-1, 1\}$ (classification)

- Goal is to...
  - Find a function $\hat{y} = f(\mathbf{x})$
  - Such that error $L(y, \hat{y})$ on new (unseen) $\mathbf{x}$ is minimal
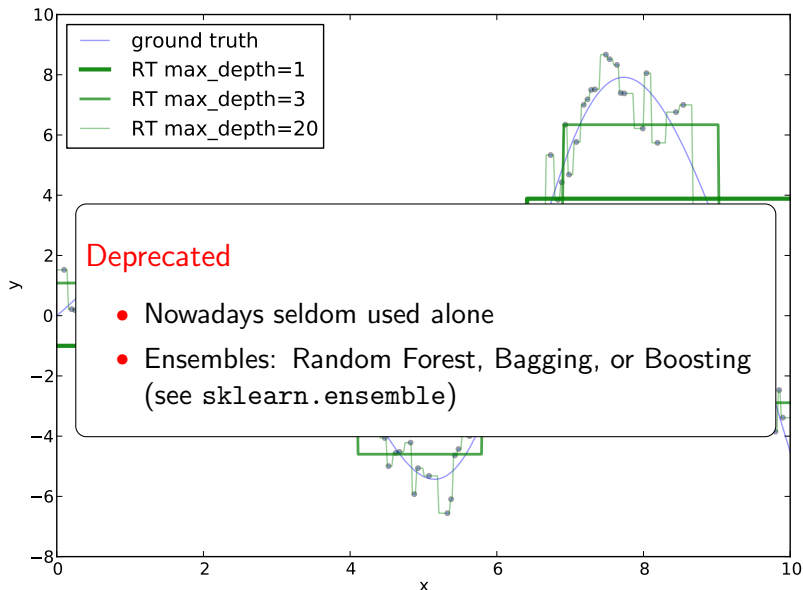
Classification and Regression Trees [Breiman et al, 1984]

learn `sklearn.tree.DecisionTreeClassifier|Regressor`

Function approximation with Regression Trees

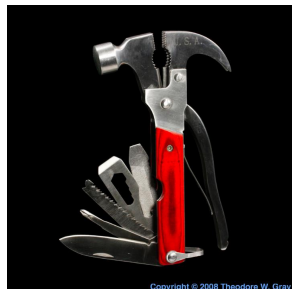# Function approximation with Regression Trees

Outline

Gradient Boosted Regression Trees

Advantages

- Heterogeneous data (features measured on different scale)
- Supports different loss functions (e.g. huber)
- Automatically detects (non-linear) feature interactions
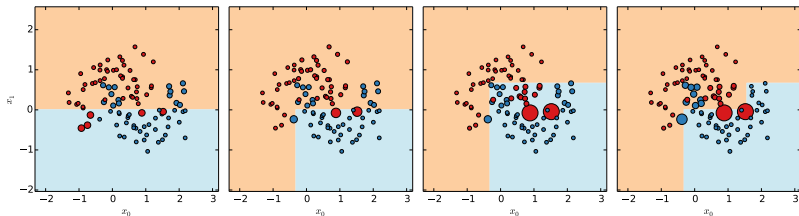
Disadvantages

- Requires careful tuning
- Slow to train (but fast to predict)
- Cannot extrapolate
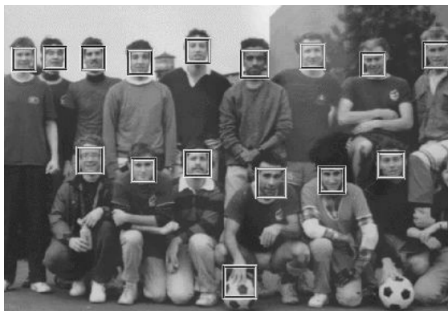
Boosting

AdaBoost [Y. Freund & R. Schapire, 1995]

- Ensemble: each member is an expert on the errors of its predecessor
- Iteratively re-weights training examples based on errors

Boosting

AdaBoost

Huge success

- Viola-Jones Face Detector (2001)

- Freund & Schapire won the Gödel prize 2003

`sklearn.ensemble.AdaBoostClassifier|Regressor`
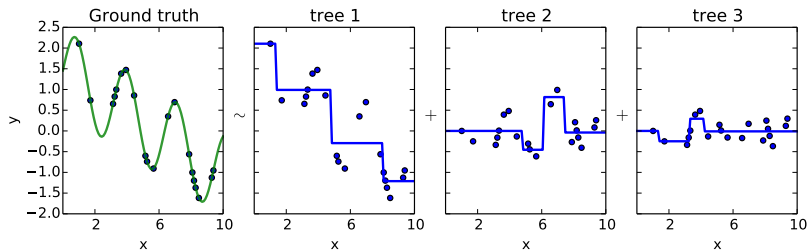
Gradient Boosting [J. Friedman, 1999]

Statistical view on boosting

- ⇒ Generalization of boosting to arbitrary loss functions

Gradient Boosting [J. Friedman, 1999]

Statistical view on boosting

- ⇒ Generalization of boosting to arbitrary loss functions

Residual fitting

Functional Gradient Descent

Least Squares Regression

- Squared loss: $L(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$
- The residual $\sim$ the (negative) gradient $\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}$

Functional Gradient Descent

Least Squares Regression

- Squared loss: $L(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$
- The residual $\sim$ the (negative) gradient $\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}$
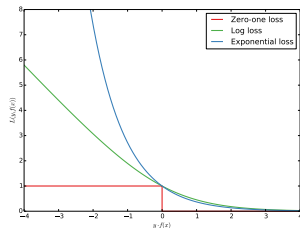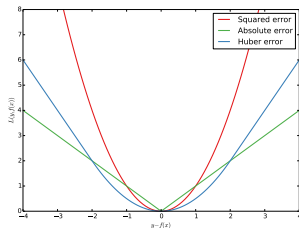
Steepest Descent

- Regression trees approximate the (negative) gradient
- Each tree is a successive gradient descent step

Outline

Notebook
http://github.com/pprett/pydata-gbm-tutorial

Tipps & Tricks 1

Input layout

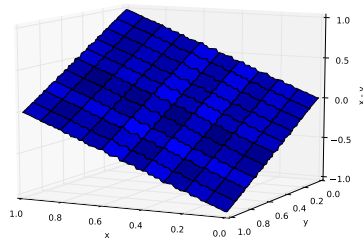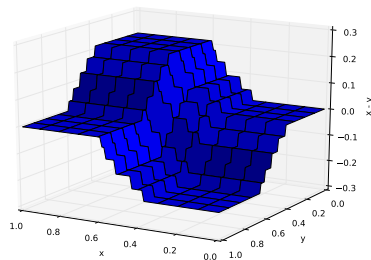Use dtype=np.float32 to avoid memory copies and fortan layout for slight runtime benefit.

```
X = np.asfortranarray(X, dtype=np.float32)
```

Tipps & Tricks 2

Feature interactions

GBRT automatically detects feature interactions but often explicit interactions help.

Trees required to approximate $X_1 - X_2$: 10 (left), 1000 (right).

Tipps & Tricks 3

Categorical variables

Sklearn requires that categorical variables are encoded as numerics. Tree-based methods work well with ordinal encoding:

```python
df = pd.DataFrame(data={'icao': ['CRJ2', 'A380', 'B737', 'B737']})
# ordinal encoding
df_enc = pd.DataFrame(data={'icao': np.unique(df.icao,
                return_inverse=True)[1]})
X = np.asfortranarray(df_enc.values, dtype=np.float32)
```