Ensemble methods: Gradient boosting

Glenn Bruns
CSUMB

Learning outcomes

After this lecture you should be able to:

1. Describe, in detail, how gradient boosting works
2. Explain how gradient boosting is used for classification and regression
3. Use gradient boosting in Scikit-Learn
4. Describe and use another ensemble method: stacking

The concept of gradient boosting

Like AdaBoost, in gradient boosting the predictors are trained sequentially.

Now, a predictor is fitted to the residual errors of the previous predictor.

The ensemble makes a prediction by successive adjustments.

A regression example; training

First predictor:

```
from sklearn.tree import DecisionTreeRegressor

tree_reg1 = DecisionTreeRegressor(max_depth=2)
tree_reg1.fit(X, y)
```

Second predictor:

```
y2 = y - tree_reg1.predict(X)
tree_reg2 = DecisionTreeRegressor(max_depth=2)
tree_reg2.fit(X, y2)
```

Third predictor:

```
y3 = y2 - tree_reg2.predict(X)
tree_reg3 = DecisionTreeRegressor(max_depth=2)
tree_reg3.fit(X, y3)
```

A regression example; prediction

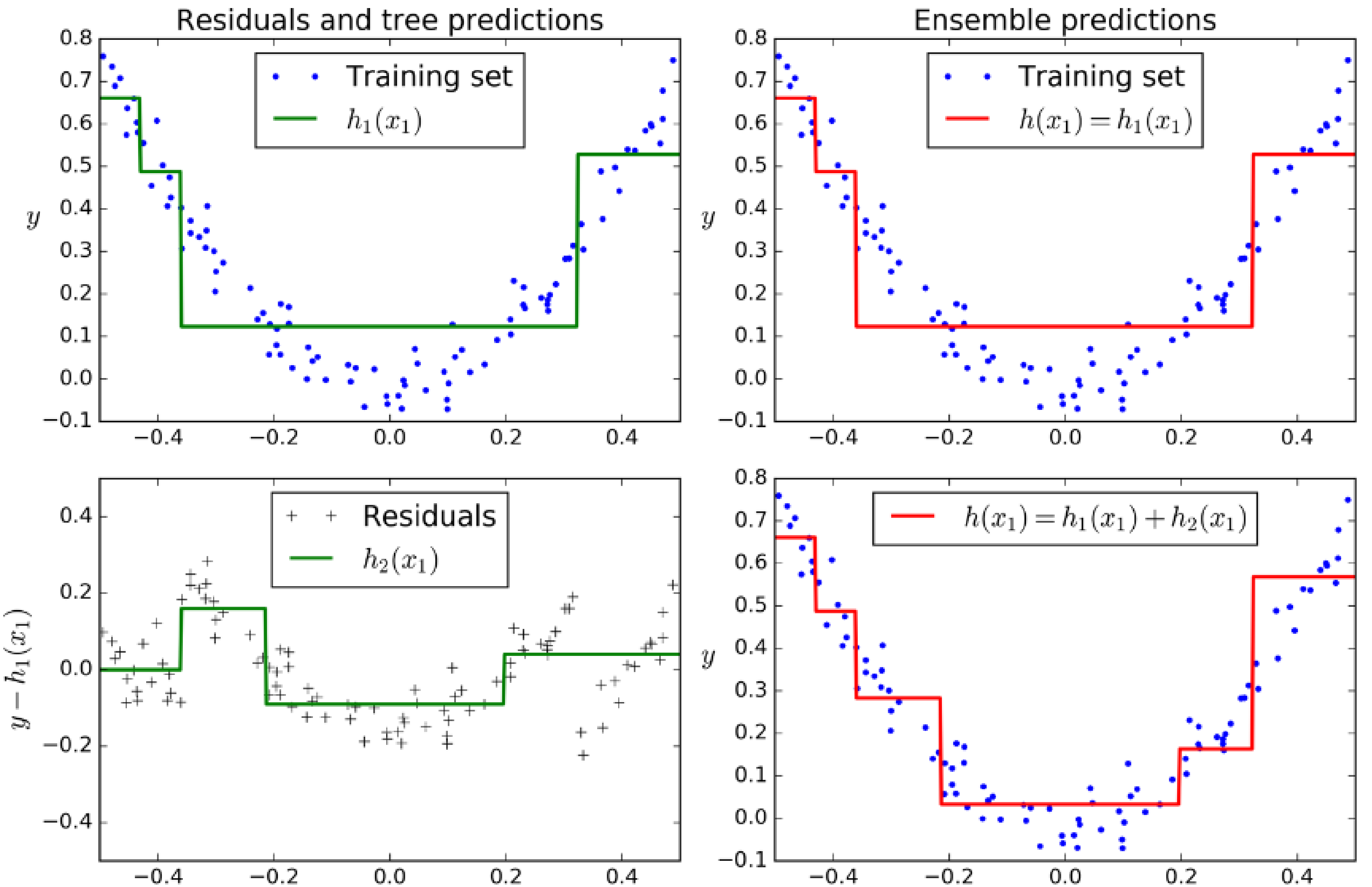
How the ensemble makes a prediction:

```
y_pred = sum(tree.predict(X_new) for tree in
                                     (tree_reg1, tree_reg2, tree_reg3))
```

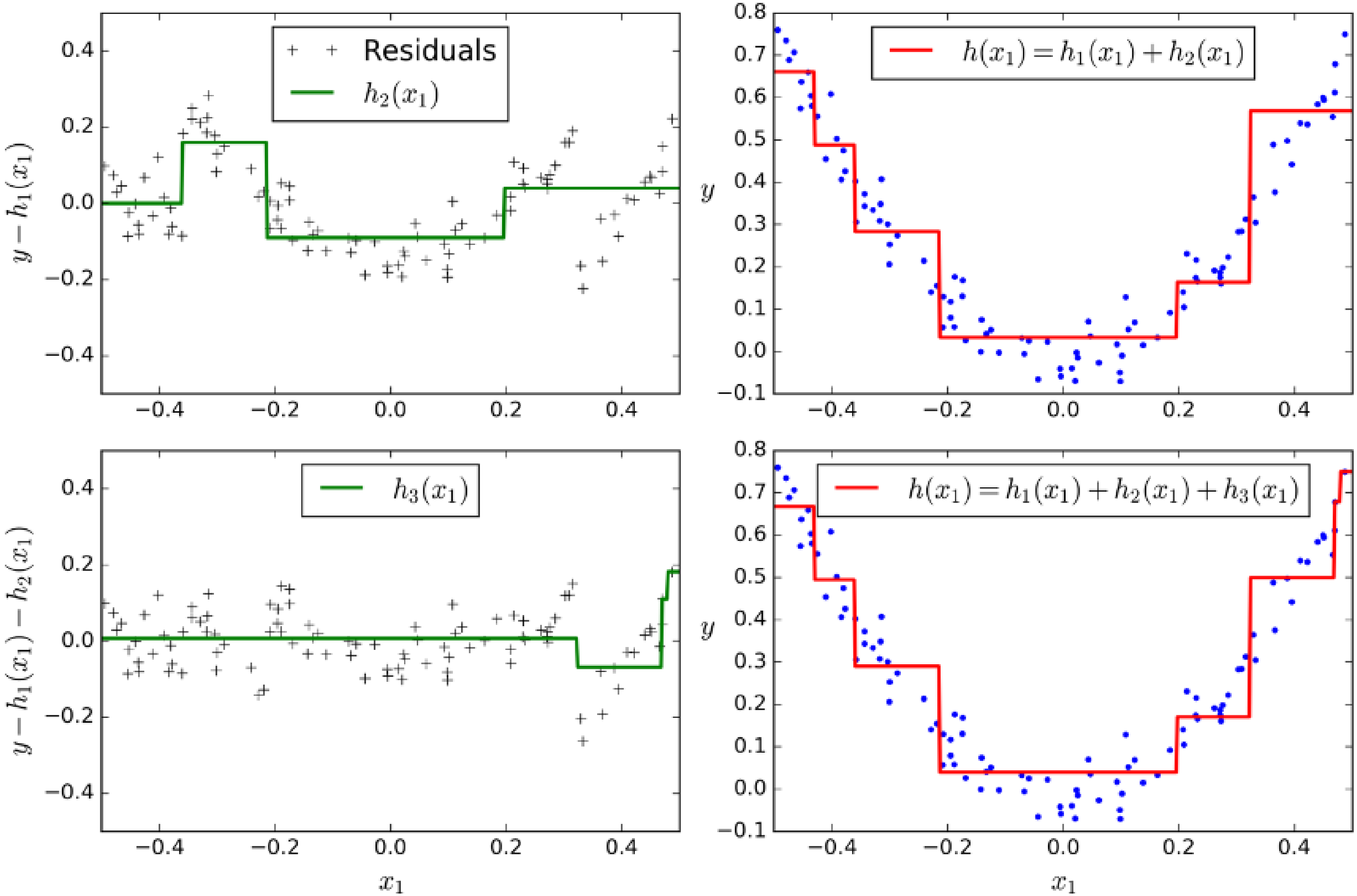
- In AdaBoost, ‘shortcomings’ are identified by high-weight data points.
- In Gradient Boosting, ‘shortcomings’ are identified by gradients.

(“A Gentle Introduction to Gradient Boosting”, Cheng Li, Northeastern University)

Visualization: predictors 1 and 2



Visualization: predictors 2 and 3



Gradient boosting regression in sklearn

```
from sklearn.ensemble import GradientBoostingRegressor

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3,
                                 learning_rate=1.0)
gbrt.fit(X, y)
```

The GradientBoostingRegressor is tree-based, just like the previous example.

Hyperparameters:

- max_depth: concerns the regression tree
- n_estimators: controls the number of trees
- learning_rate: controls the contribution of each tree

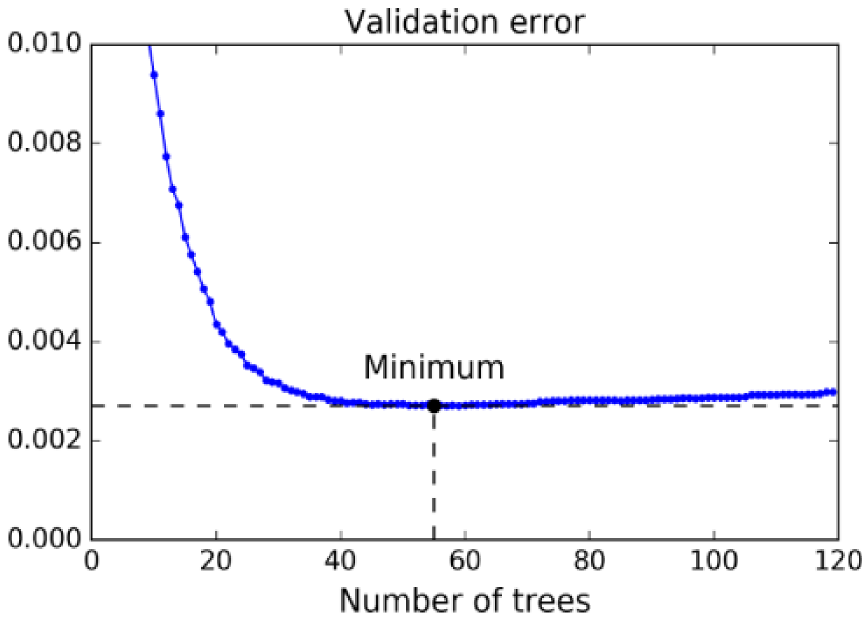
Other tree-related hyperparameters can also be used.

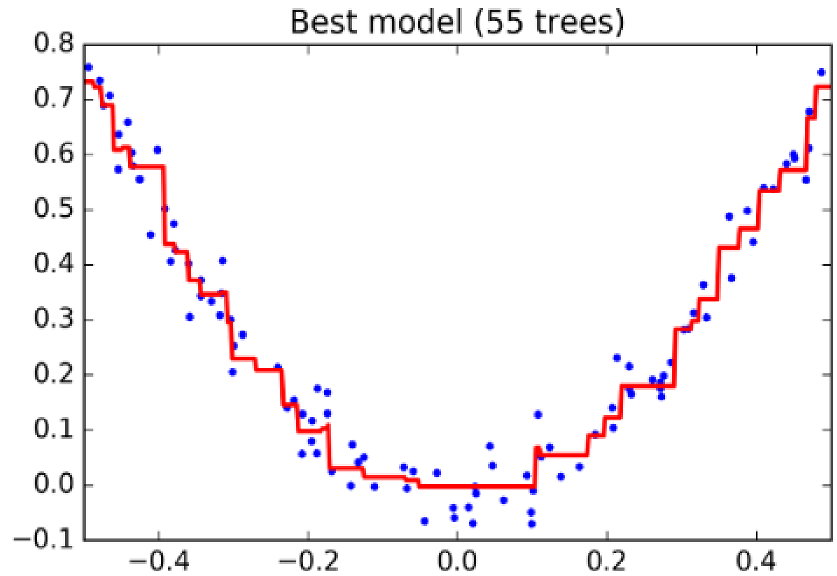
If a low-learning rate is used, more predictors are needed.

Finding the optimal number of trees

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_val, y_train, y_val = train_test_split(X, y)

gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=120)
gbrt.fit(X_train, y_train)

# find the MSE for each successive tree
errors = [mean_squared_error(y_val, y_pred)
          for y_pred in gbrt.staged_predict(X_val)]
bst_n_estimators = np.argmin(errors)

# build a gradient boosting regressor with the best # of trees
gbrt_best = GradientBoostingRegressor(max_depth=2,
                                      n_estimators=bst_n_estimators)
gbrt_best.fit(X_train, y_train)
```

Visualization of optimal # of trees





Question: what variable in the previous slide is being plotted here?

Question: what variable in the previous slide got the value 55?

Classification with gradient boosting

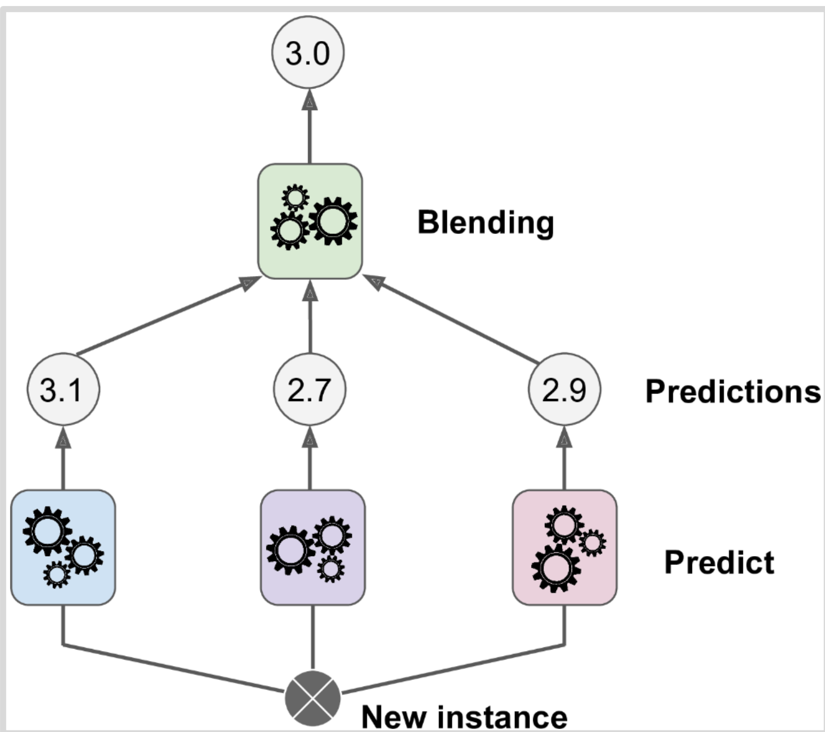
The ‘deviance’ loss function can be used for classification with probabilistic outputs.

The GradientBoostingClassifier is tree-based, like the GradientBoostingRegressor.

Stacking

In a voting classifier the predictor outputs are combined in a "hard wired" manner.

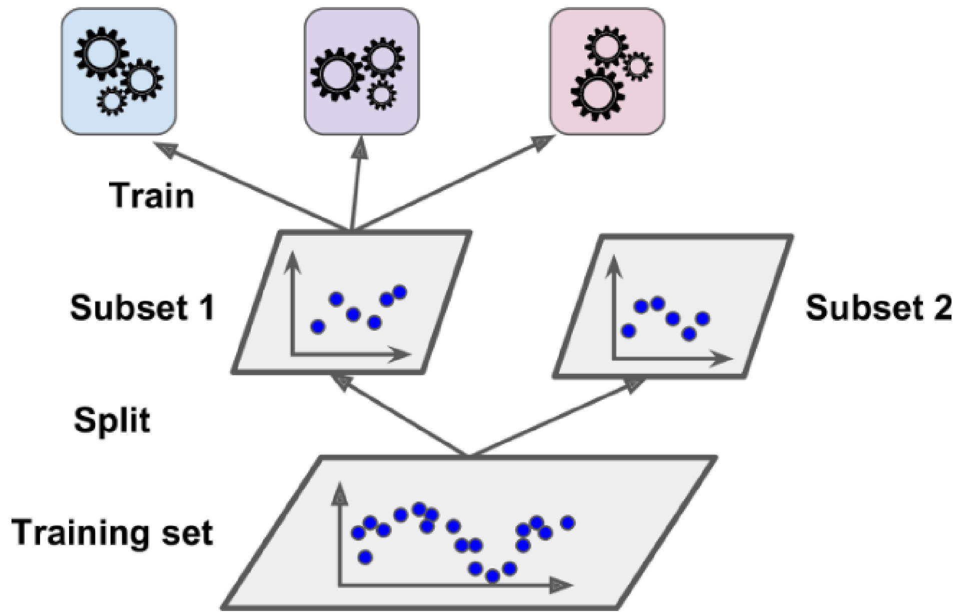
Why not train a model to combine them in an optimal way?



The blender is trained to combine the predictions from the bottom three predictors

Training the blender

1. Split the training set into two parts.
2. Use the first part to train predictors in the first level
3. The first-level predictors then make predictions with the second part.



4. Create a new training set using predictions of previous step, plus target values of the second part (!)
5. Train the blender on this new training set.

Stacking example

x	y
1	3.4
6	8.9
3	5.1
4	5.3
5	6.5
2	3.7

use half of training set to train predictors

trained predictors make predictions from other half

x	y	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$
4	5.3	5.1	5.3	5.0
5	6.5	5.9	6.7	6.2
2	3.7	3.1	4.2	3.9

combine the training data of previous step with the predictions

x_1	x_2	x_3	y
5.1	5.3	5.0	5.3
5.9	6.7	6.2	6.5
3.1	4.2	3.9	3.7

use this data to train blender

Summary

- In gradient boosting, you have a sequence of predictors
- Each predictor is trained on the errors of the previous predictor
- The ensemble prediction is the sum of the individual predictions
- With stacking, you train a blender to combine the results from a diverse set of predictors