Boosting
with an emphasis on
Adaptive Boosting:
Theory & Intuition

Ensemble Learning

semble models are *composite* models (aka meta-models) that combine lividual models with flaws in a group to create a strong(er) final model.

semble learning/Ensemble Methods/Ensemble Models

Ensemble Learning

semble models are *composite* models that **combine** individual models with
ws in a group to create a strong(er) final model.

ere are different approaches to combining individual models ➢ different
vors of ensembling.

sed on the characteristics of the individual models and the desired
aracteristics of the meta-model.

Ensemble Learning

gging: Individual models have low bias, but high variance (ie, overfitted). W
ate an *equal* voting group by bootstrapping our dataset. Eg Random Fores

osting: Individual models have high bias, but low variance (ie, underfitted)
ate a sequence of *weighed* individual models.

cking: Individual models are well fitted, and may be of different types. We
other model to determine how to determine the contribution of each mod

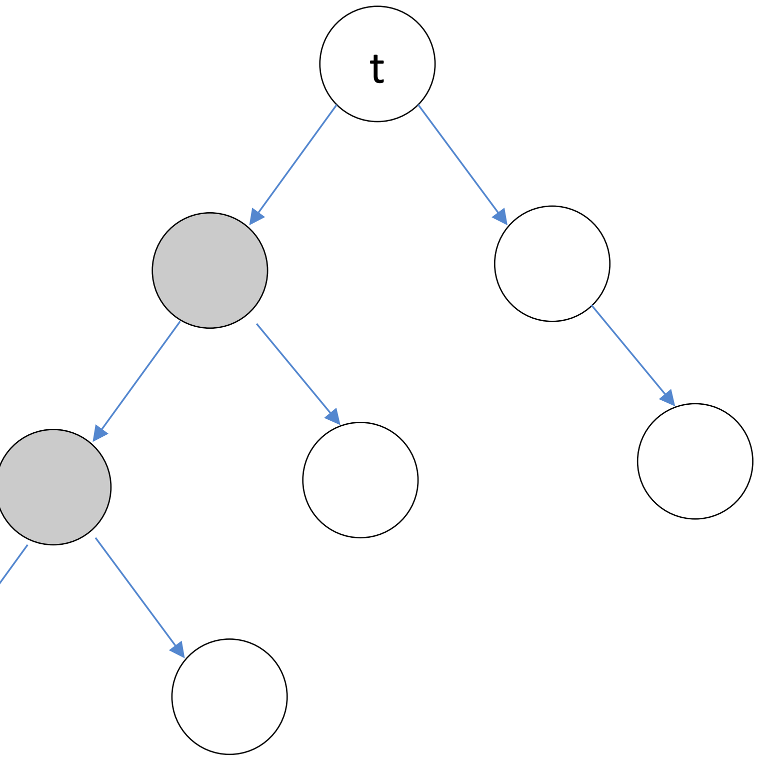
Bagging Versus Boosting

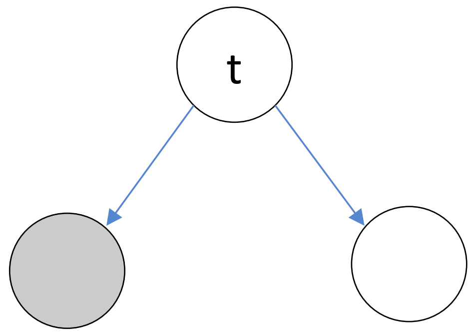
agine that you are participating in the annual Nathan’s Famous Hot Dog Ea
ntest. Your team has to eat 30 hot dogs in 5 minutes. How do you go abou
tting a team together?

Bagging Versus Boosting: Differences

gging (think random forests) relies on individual models having low bias an
h variance. So, the decision trees are (usually) as deep as need be.

osting relies on individual models having high bias and low variance. So, th
cision trees are just *stumps*.

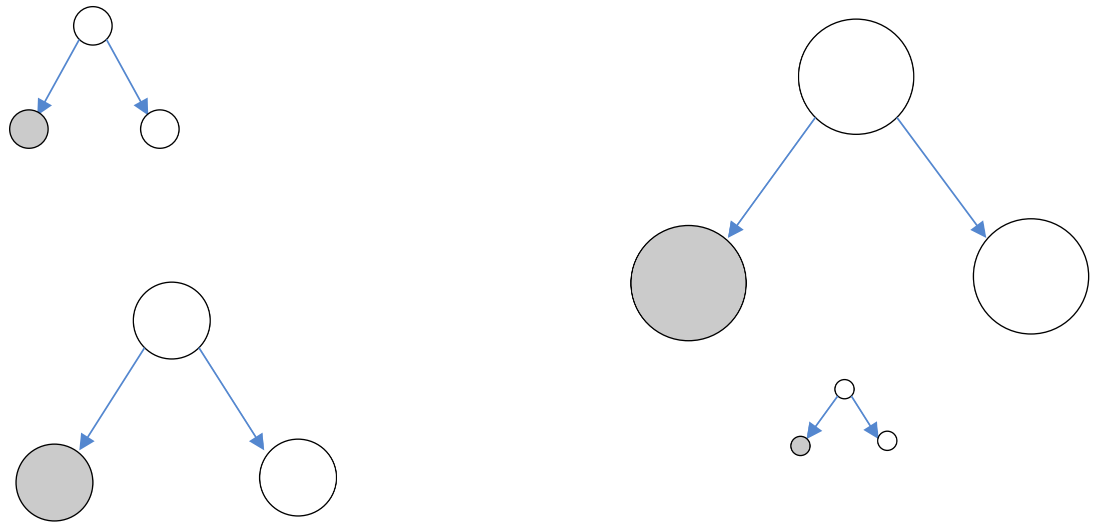




Bagging Versus Boosting: Differences

Bagging approaches, all individual models have an equal vote in the final
cision.

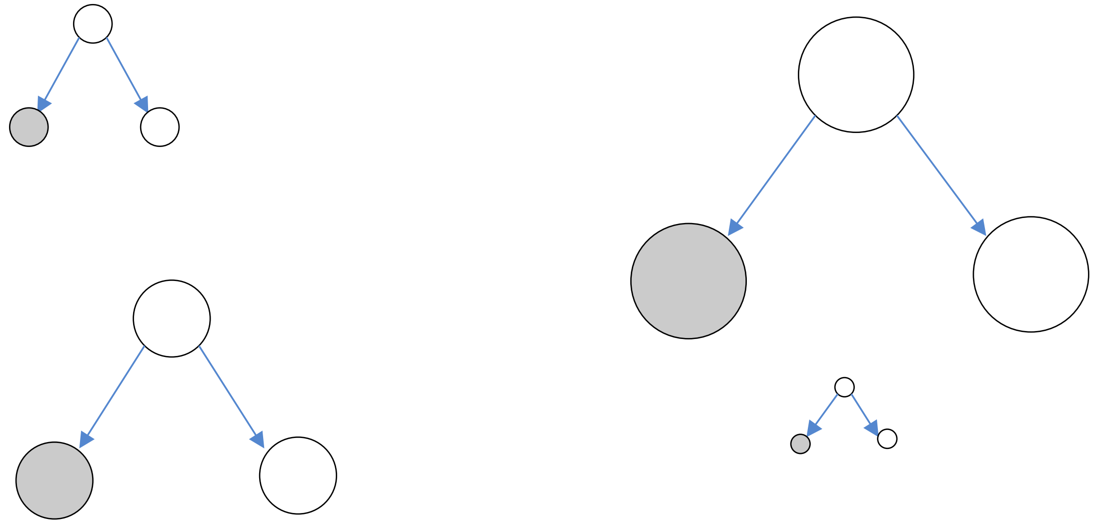
Boosting approaches, different trees have different weights associated wit
eir contributions…based on their accuracy.



Bagging Versus Boosting: Differences

Bagging approaches, all individual models are trained independently.

Boosting approaches, the models are trained in sequence…one after the ot
e errors of prior models affect the subsequent models’ training.



Adaptive Boosting: AdaBoost

Adaptive Boosting: AdaBoost

X1	X2	Y
1	1	c1
0	1	c1
1	0	c1
1	1	c1
0	1	c2
0	1	c2
1	0	c2
1	1	c2

Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Adaptive Boosting: AdaBoost

X1	X2	Y	w
1	1	c1	0.125
0	1	c1	0.125
1	0	c1	0.125
1	1	c1	0.125
0	1	c2	0.125
0	1	c2	0.125
1	0	c2	0.125
1	1	c2	0.125

Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Step 0: Assign weights, w, to the data.

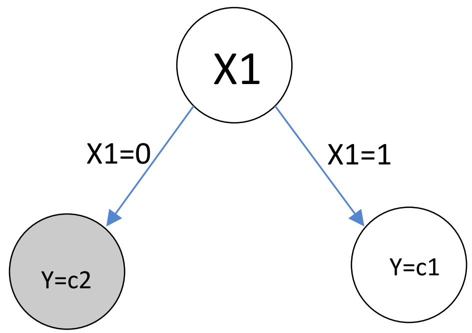
Adaptive Boosting: AdaBoost

X1	X2	Y	w
1	1	c1	0.125
1	1	c1	0.125
1	0	c1	0.125
1	1	c1	0.125
0	1	c2	0.125
0	1	c2	0.125
1	0	c2	0.125
0	1	c2	0.125

Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Step 0: Assign weights, w, to the data.

Step 1: Train "next" model (stump) on the data.



Adaptive Boosting: AdaBoost

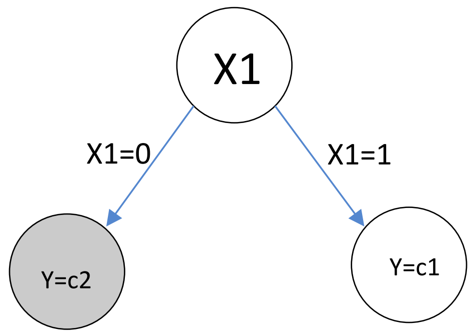
X1	X2	Y	w
1	1	c1	0.125
1	1	c1	0.125
1	0	c1	0.125
1	1	c1	0.125
0	1	c2	0.125
0	1	c2	0.125
1	0	c2	0.125
0	1	c2	0.125

Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Step 0: Assign weights, w, to the data.

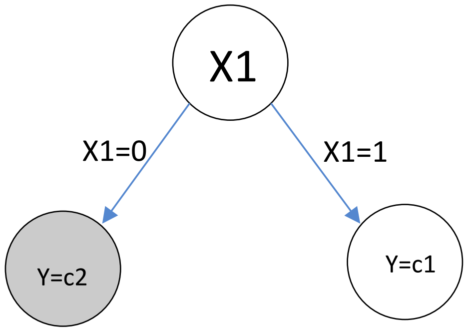
Step 1: Train “next” model (stump) on the data.

Step 2: Total error for model i= Sum of weights of incorrect samples. ei=0.125



Adaptive Boosting: AdaBoost

X1	X2	Y	w	w_new
1	1	c1	0.125	0.05
1	1	c1	0.125	0.05
1	0	c1	0.125	0.05
1	1	c1	0.125	0.05
0	1	c2	0.125	0.05
0	1	c2	0.125	0.05
1	0	c2	0.125	0.33
0	1	c2	0.125	0.05



Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Step 0: Assign weights, w, to the data.

Step 1: Train "next" model (stump) on the data.

Step 2: Total error for model i= Sum of weights of incorrect samples. ei=0.125

Step 3: Calculate weight associated with model i:

$$\alpha_i = 1/2 \log(1 - e_i / e_i)$$

$$\alpha_0 = 0.97$$

Step 4: Adjust weights for the samples, based on the errors made by model i.

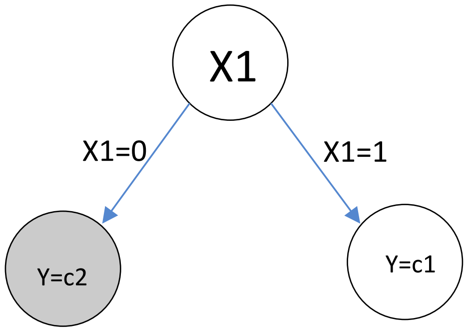
Increase weights of *incorrectly* classified samples, *decrease* weights of *correctly* classified samples.

$$w_k^{incorrect} = w_k \, .\exp(\alpha_i)$$

$$w_k^{correct} = w_k \, .\exp(-\alpha_i)$$

Adaptive Boosting: AdaBoost

X1	X2	Y	w	w_new
1	1	c1	0.125	0.07
1	1	c1	0.125	0.07
1	0	c1	0.125	0.07
1	1	c1	0.125	0.07
0	1	c2	0.125	0.07
0	1	c2	0.125	0.07
1	0	c2	0.125	0.51
0	1	c2	0.125	0.07



Create a meta-model using Adaboost in conjunction with decision tree stumps to fit this classification problem.

Step 0: Assign equal weights, w, to the data.

Step 1: Train "next" model (stump) on the data.

Step 2: Total error for model i= Sum of weights of incorrect samples. ei=0.125

Step 3: Calculate weight associated with model i:

$$\alpha_i = 1/2 \log(1-e_i / e_i)$$

$\alpha_0 = 0.97$

Step 4: Adjust weights for the samples, based on the errors made by model i.

Increase weights of *incorrectly* classified samples, *decrease* weights of *correctly* classified samples.

$$w_k^{incorrect} = w_k . \exp(\alpha_i)$$

$$w_k^{correct} = w_k . \exp(-\alpha_i)$$

Step 5: Normalize weights and goto step 1.