CSCI4622 Machine Learning 2020 Spring

Ensemble method

Geena Kim

*Slide contents adopted from ISLR material

Ensemble method- review

Problem: Trees are weak learner and trees overfit

Idea 1: Let's average them (Ensemble)
Idea 2: Let's make decorrelated trees (samples, features)

Random Forest

Random Forest

Bagging : random sampling of data

+

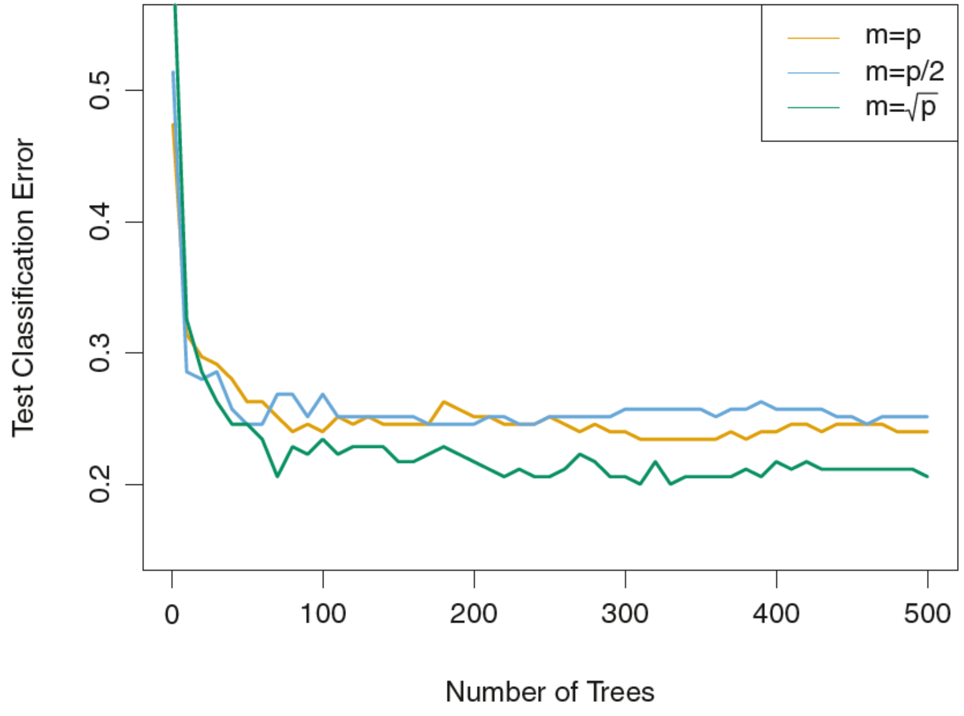
Decorrelation : random sampling of features

=

Random Forest

How do we sample features?

-> Rule of thumb : $\sqrt{n}$



Ensemble method- review

Problem: Trees are weak learner and trees overfit

Idea 3: Let's make the trees a strong learner
How: Grow a small tree (stump) to fit residual

Boosting

Boosting

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

ISLR. *p.323*

Popular Boosted Tree Methods

- AdaBoost (Adaptive Boosting)
- GBM (Gradient Boosting Machine)
- XGBoost (Extreme Gradient Boosting)

Popular Boosted Tree Methods

- AdaBoost (Adaptive Boosting)
- GBM (Gradient Boosting Machine)
- XGBoost (Extreme Gradient Boosting)

AdaBoost

Idea: Focus on the misclassified samples

Initialize data weights to $w_i = \frac{1}{m}, i = 1, \ldots, m$
For $k = 1$ to K:

Fit estimator $f_k(\mathbf{x})$ to training data with weights w_i
Compute weighted error $\epsilon_k = \frac{\sum_{i=1}^{m} w_i \mathrm{I}(y_i \neq f_k(\mathbf{x}_i))}{\sum_{i=1}^{m} w_i}$
Compute estimator weight $\lambda_k = \frac{1}{2} \log((1 - \epsilon_k)/\epsilon_k)$
Update sample weight $w_i \leftarrow \frac{w_i}{Z_k} \cdot \exp[-\lambda_k y_i f_k(\mathbf{x}_i)]$

Final model $F(\mathbf{x}) = \mathrm{sign}\left[\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x})\right]$

AdaBoost

Initialize data weights to $w_i = \frac{1}{m}, i = 1, \ldots, m$

	Age	Sex	ChestPain	Chol	AHD
0	63	1	typical	233	No
1	67	1	asymptomatic	286	Yes
2	67	1	asymptomatic	229	Yes
3	37	1	nonanginal	250	No
4	41	0	nontypical	204	No
5	56	1	nontypical	236	No
6	62	0	asymptomatic	268	Yes
7	57	0	asymptomatic	354	No
8	63	1	asymptomatic	254	Yes
9	53	1	asymptomatic	203	Yes

AdaBoost

Fit estimator $f_k(\mathbf{x})$ to training data with weights w_i

	Age	Sex	ChestPain	Chol	AHD	weight	Yp
0	63	1	typical	233	No	0.1	Yes
1	67	1	asymptomatic	286	Yes	0.1	Yes
2	67	1	asymptomatic	229	Yes	0.1	Yes
3	37	1	nonanginal	250	No	0.1	No
4	41	0	nontypical	204	No	0.1	No
5	56	1	nontypical	236	No	0.1	No
6	62	0	asymptomatic	268	Yes	0.1	Yes
7	57	0	asymptomatic	354	No	0.1	No
8	63	1	asymptomatic	254	Yes	0.1	Yes
9	53	1	asymptomatic	203	Yes	0.1	No
10	57	1	asymptomatic	192	No	0.1	No

AdaBoost

Compute weighted error $\epsilon_k = \frac{\sum_{i=1}^{m} w_i \mathrm{I}(y_i \neq f_k(\mathbf{x}_i))}{\sum_{i=1}^{m} w_i}$

	Age	Sex	ChestPain	Chol	AHD	weight	Yp
0	63	1	typical	233	No	0.1	Yes
1	67	1	asymptomatic	286	Yes	0.1	Yes
2	67	1	asymptomatic	229	Yes	0.1	Yes
3	37	1	nonanginal	250	No	0.1	No
4	41	0	nontypical	204	No	0.1	No
5	56	1	nontypical	236	No	0.1	No
6	62	0	asymptomatic	268	Yes	0.1	Yes
7	57	0	asymptomatic	354	No	0.1	No
8	63	1	asymptomatic	254	Yes	0.1	Yes
9	53	1	asymptomatic	203	Yes	0.1	No
10	57	1	asymptomatic	192	No	0.1	No

$\epsilon_k = 0.2$

AdaBoost

Compute estimator weight $\lambda_k = \frac{1}{2}\log((1-\epsilon_k)/\epsilon_k)$

	Age	Sex	ChestPain	Chol	AHD	weight	Yp
0	63	1	typical	233	No	0.1	Yes
1	67	1	asymptomatic	286	Yes	0.1	Yes
2	67	1	asymptomatic	229	Yes	0.1	Yes
3	37	1	nonanginal	250	No	0.1	No
4	41	0	nontypical	204	No	0.1	No
5	56	1	nontypical	236	No	0.1	No
6	62	0	asymptomatic	268	Yes	0.1	Yes
7	57	0	asymptomatic	354	No	0.1	No
8	63	1	asymptomatic	254	Yes	0.1	Yes
9	53	1	asymptomatic	203	Yes	0.1	No
10	57	1	asymptomatic	192	No	0.1	No

$\lambda_k = 0.69$

AdaBoost

Update sample weight $w_i \leftarrow \frac{w_i}{Z_k} \cdot \exp[-\lambda_k y_i f_k(\mathbf{x}_i)]$

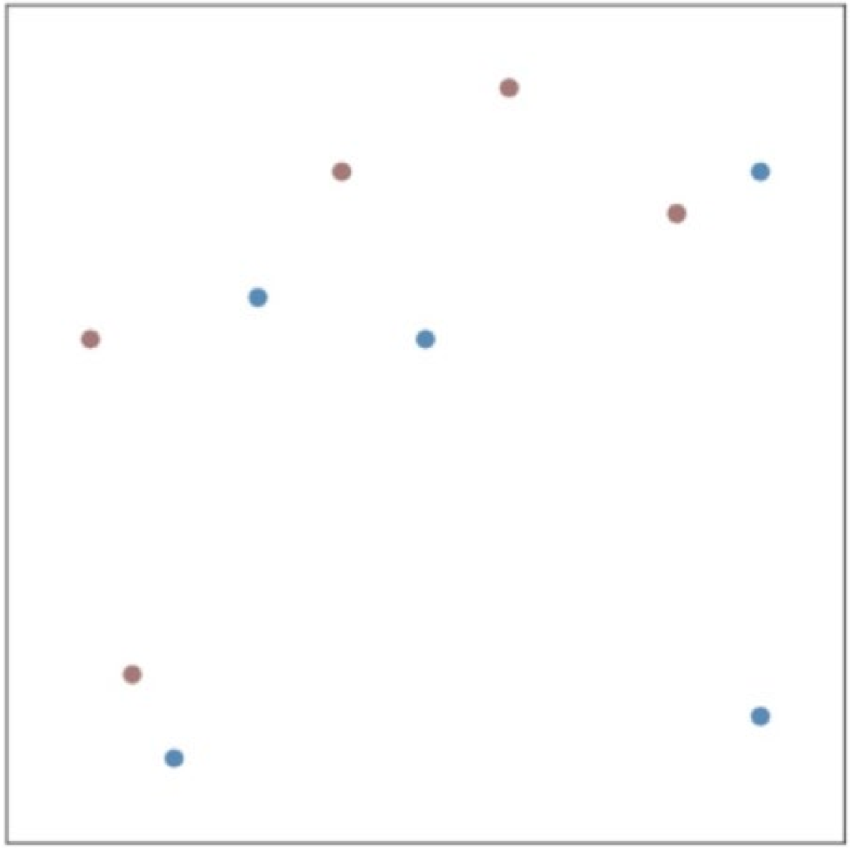
	Age	Sex	ChestPain	Chol	AHD	weight
0	63	1	typical	233	No	0.2500
1	67	1	asymptomatic	286	Yes	0.0625
2	67	1	asymptomatic	229	Yes	0.0625
3	37	1	nonanginal	250	No	0.0625
4	41	0	nontypical	204	No	0.0625
5	56	1	nontypical	236	No	0.0625
6	62	0	asymptomatic	268	Yes	0.0625
7	57	0	asymptomatic	354	No	0.0625
8	63	1	asymptomatic	254	Yes	0.0625
9	53	1	asymptomatic	203	Yes	0.2500
10	57	1	asymptomatic	192	No	0.1000

Repeat until K or the error =0

Final model $F(\mathbf{x}) = \text{sign}\left[\sum_{k=1}^{K} \lambda_k f_k(\mathbf{x})\right]$

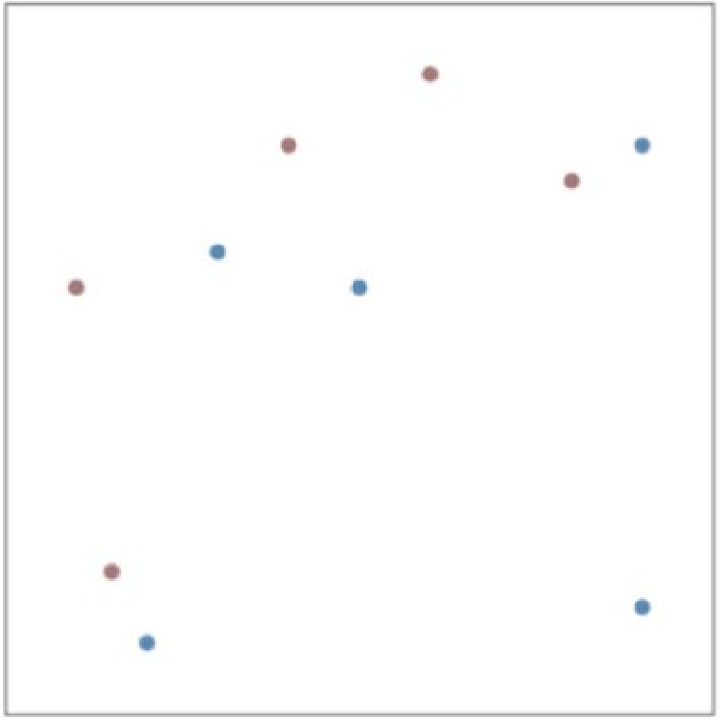
AdaBoost

For a train data

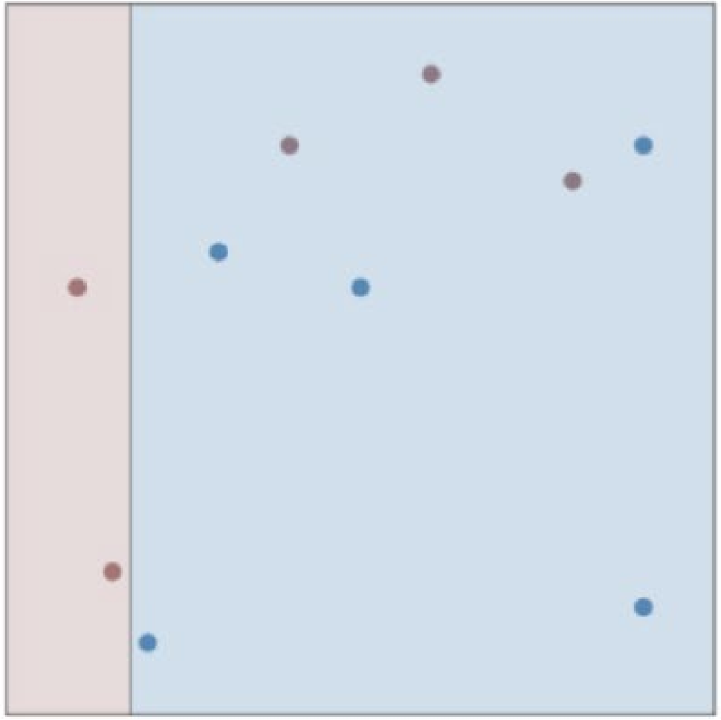


AdaBoost

Fit first stump

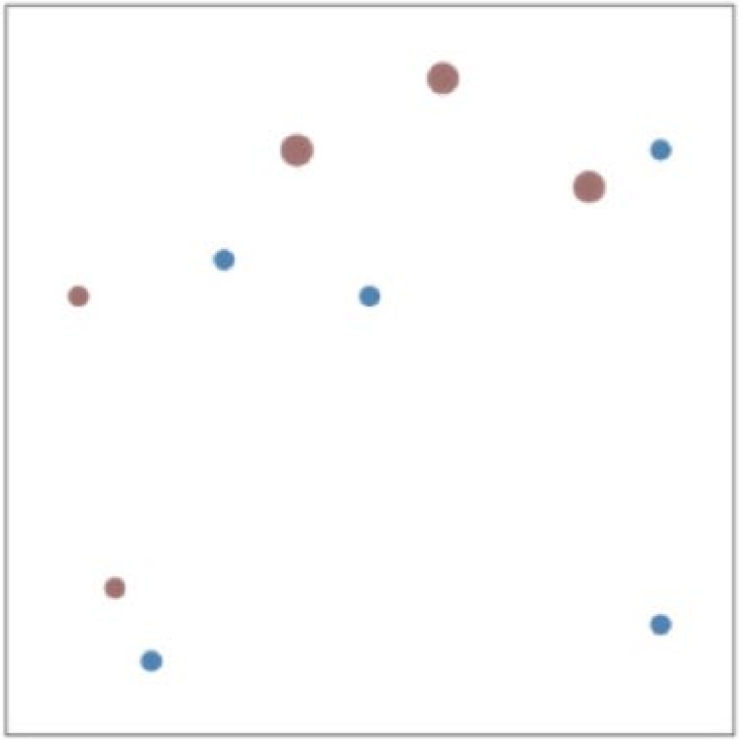


k = 1, ε = 0.3, λ = 0.42



AdaBoost

Fit second stump

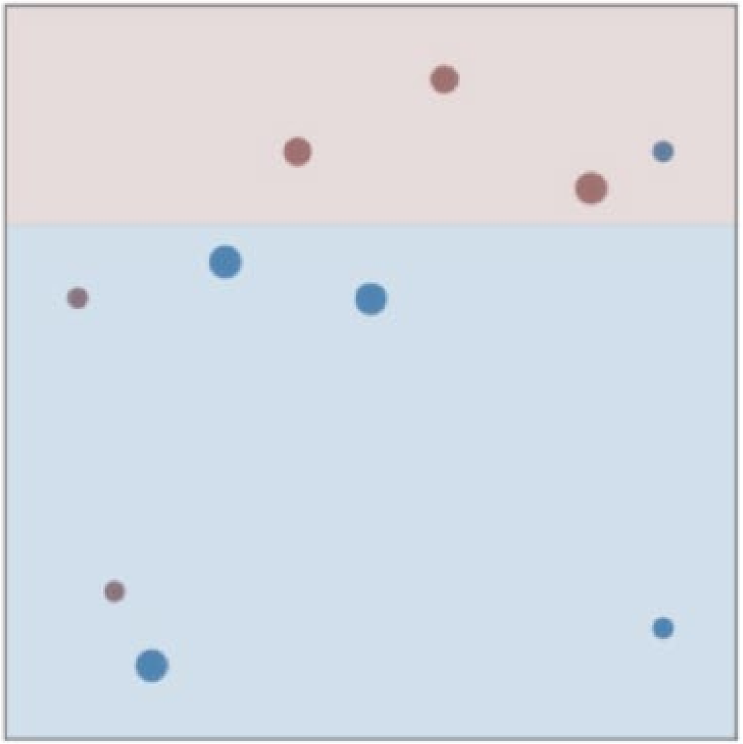


k = 2, ε= 0.21, λ = 0.65

AdaBoost

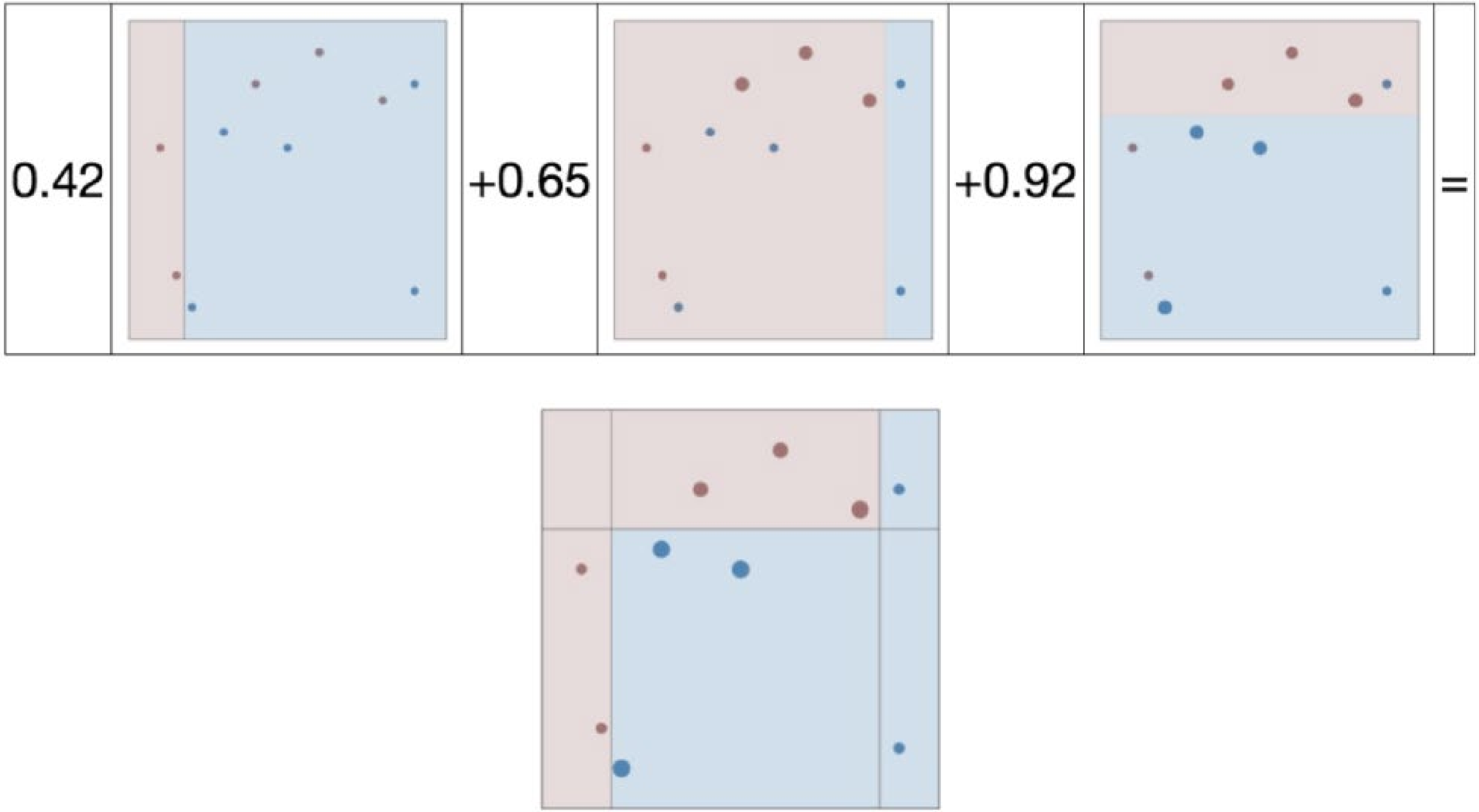
Fit third stump

$k = 2$, $\varepsilon = 0.14$, $\lambda = 0.92$



AdaBoost

The result:



Popular Boosted Tree Methods

- AdaBoost (Adaptive Boosting)
- GBM (Gradient Boosting Machine)
- XGBoost (Extreme Gradient Boosting)

Gradient Boosting

Initialize $F_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$

For k=1 to K:

For i=1,2,...,N compute

$$r_{ik} = -\left[\frac{\partial L(y_i, F_{k-1}(\mathbf{x}_i))}{\partial F_{k-1}(\mathbf{x}_i)}\right]$$

Fit a regression tree to the targets r_{ik}

For the terminals j=1,...,m compute

$$\gamma_{jk} = \arg\min_\gamma \sum_{x_i \in R_{jk}} L(y_i, F_{k-1}(\mathbf{x}_i) + \gamma)$$

Update $F_k(\mathbf{x}) = F_{k-1}(\mathbf{x}) + \sum_{j=1}^{m} \gamma_{jk} I(\mathbf{x} \in R_{jk})$

Output $\hat{f}(\mathbf{x}) = F_K(\mathbf{x})$

T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. *p.361*

Gradient Boosting

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. *p.360*

Popular Boosted Tree Methods

- AdaBoost (Adaptive Boosting)
- GBM (Gradient Boosting Machine)
- XGBoost (Extreme Gradient Boosting)

XGBoost

- Uses Newton tree boosting method
- Implements regularization helping reduce overfit (GB does not have)
- Implements parallel processing being much faster than GB

Another Variant: LightGB (very fast)

Python packages

class `sklearn.ensemble.`**`AdaBoostClassifier`**(*base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

class `sklearn.ensemble.`**`GradientBoostingClassifier`**(*loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, presort='deprecated', validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

https://xgboost.readthedocs.io/en/latest/python/python_intro.html