Boosting and Stacking

Legal Notices and Disclaimers

Learning Objectives

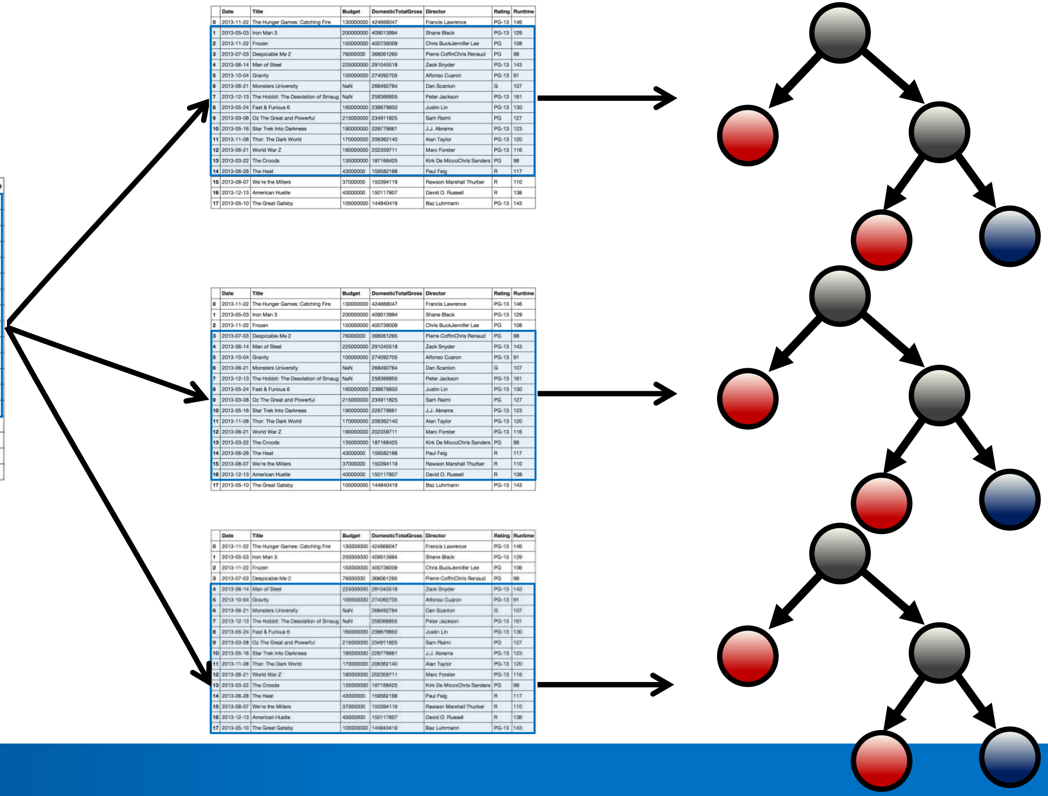
- Explain how the boosting algorithm helps reduce variance and bias.

- Apply Intel® Extension for Scikit-learn* to leverage underlying compute capabilities of hardware

Review of Bagging

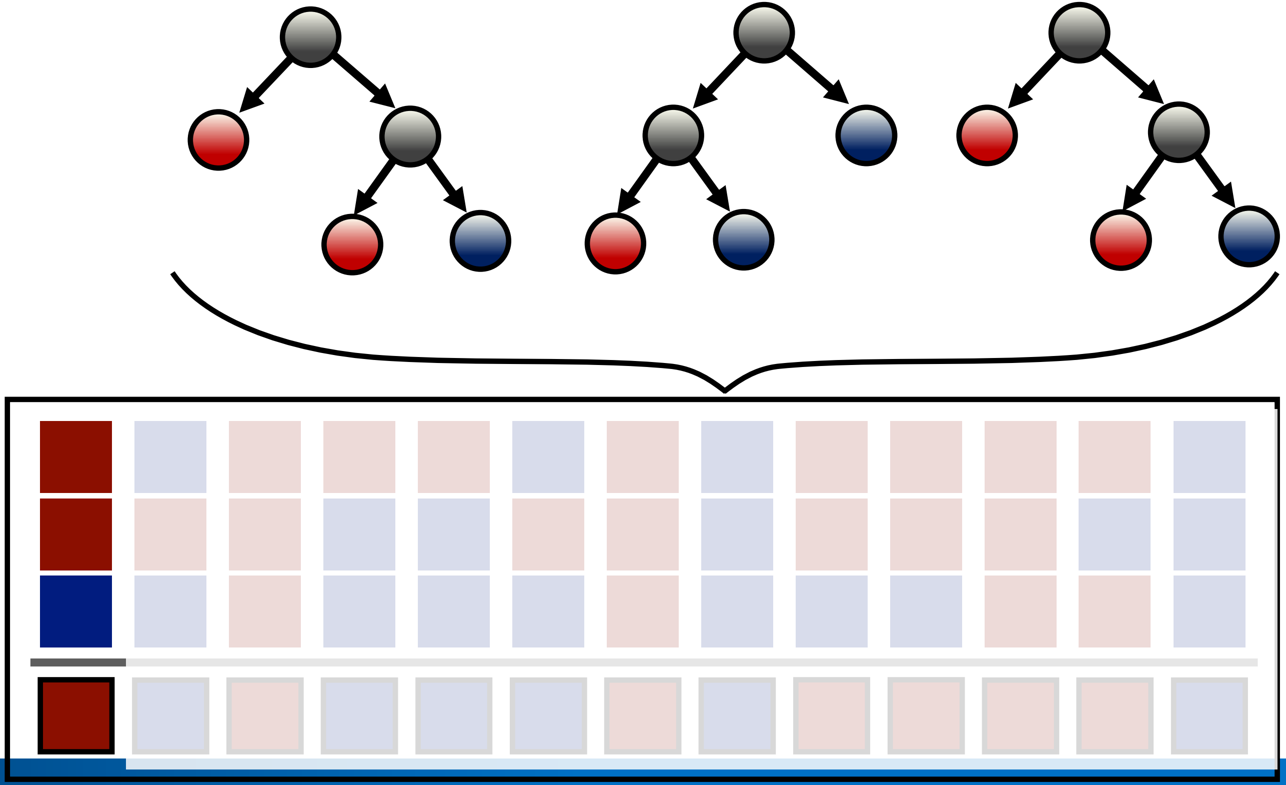
Grow decision tree from multiple bootstrapped samples

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



Review of Bagging

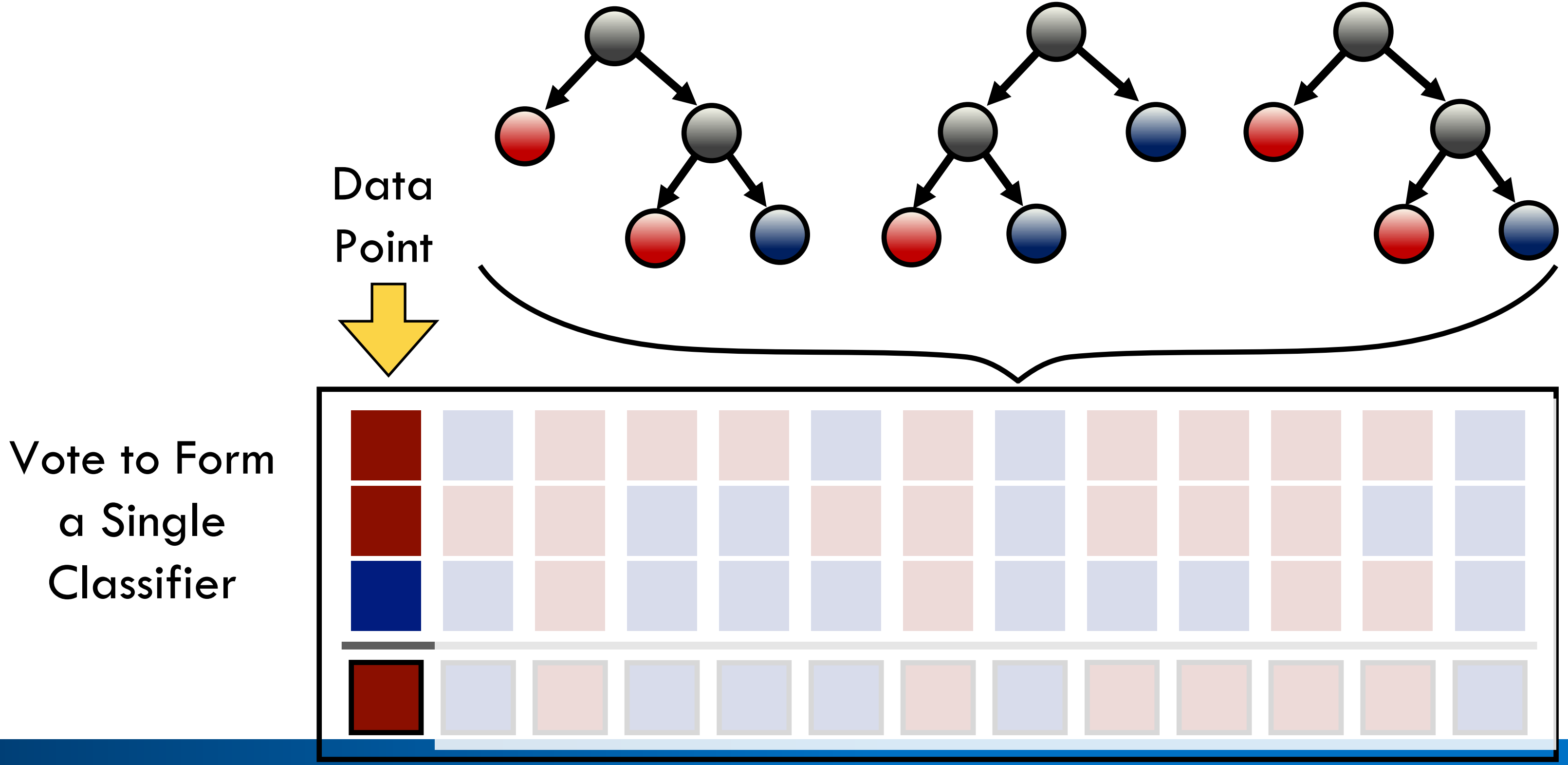
Vote on or average result from each tree for each data point



Vote to Form a Single Classifier

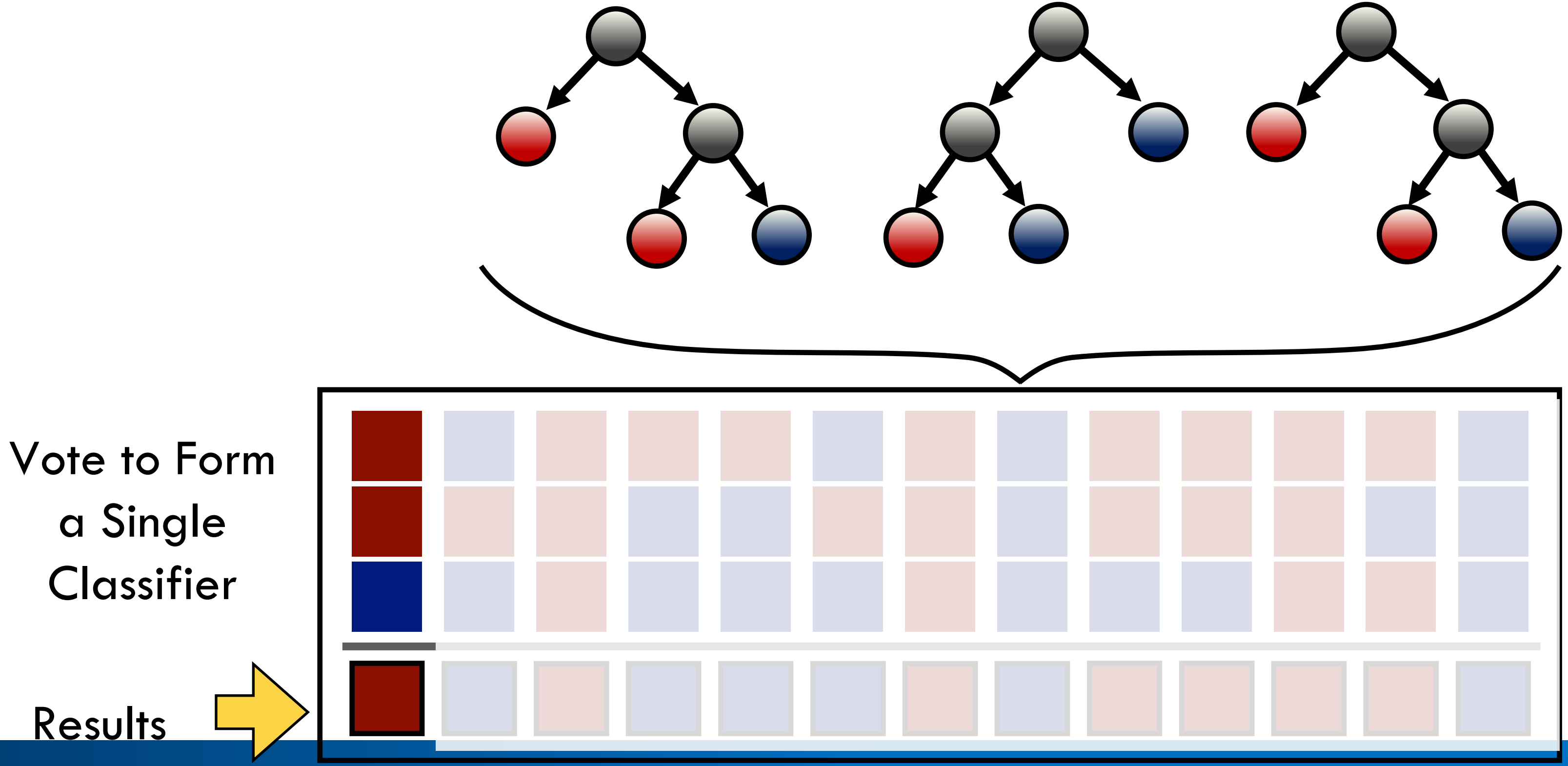
Review of Bagging

Vote on or average result from each tree for each data point



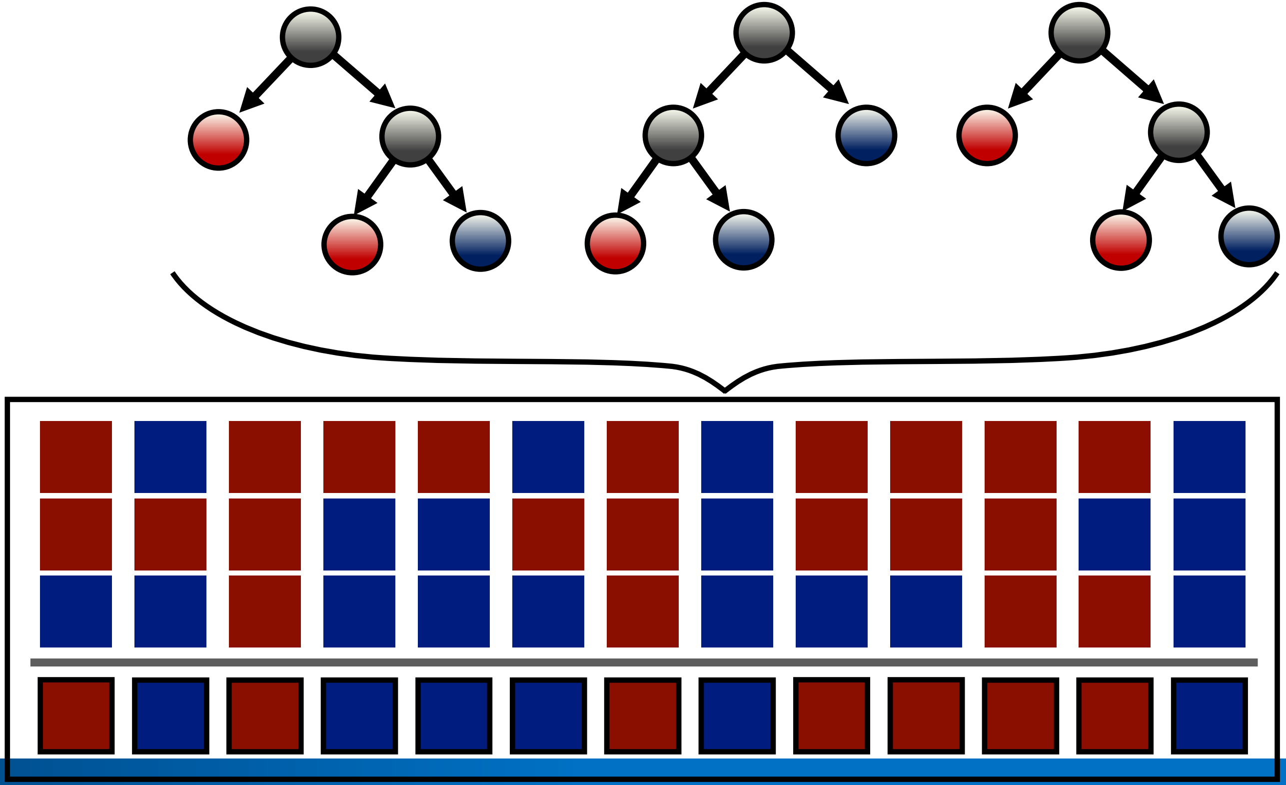
Review of Bagging

Vote on or average result from each tree for each data point



Review of Bagging

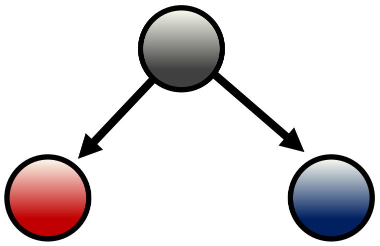
Vote on or average result from each tree for each data point



Vote to Form a Single Classifier

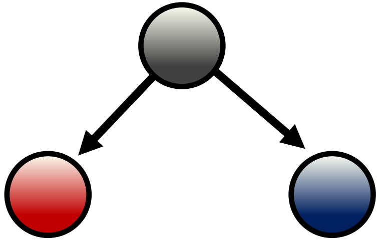
Decision Stump: the Boosting Base Learner

Temperature >50°F

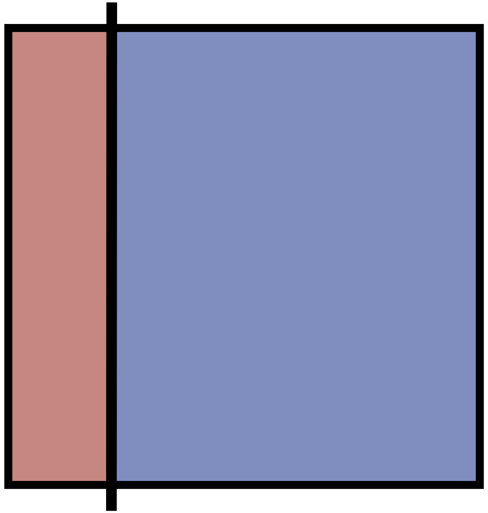


Decision Stump: the Boosting Base Learner

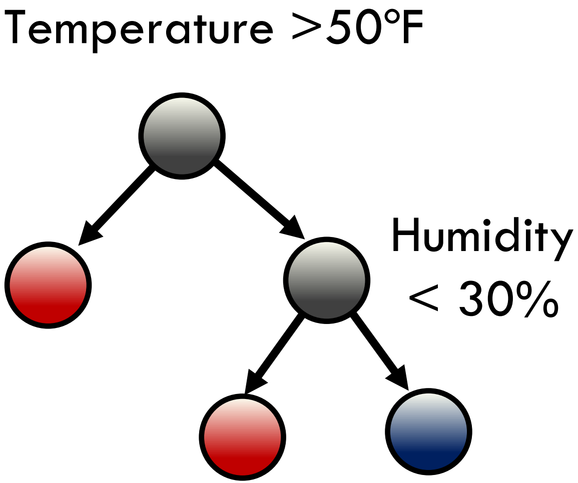
Temperature >50°F

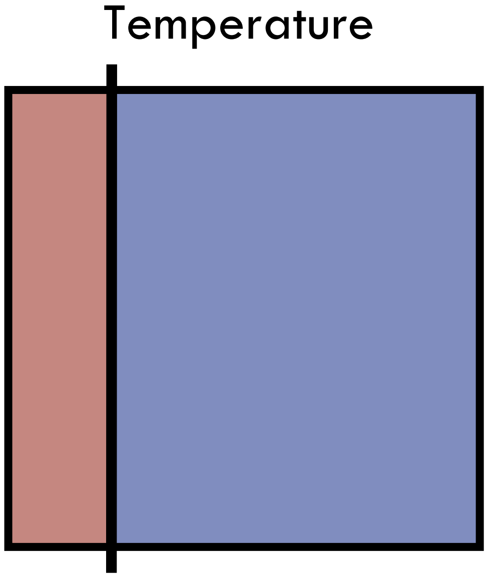


Temperature

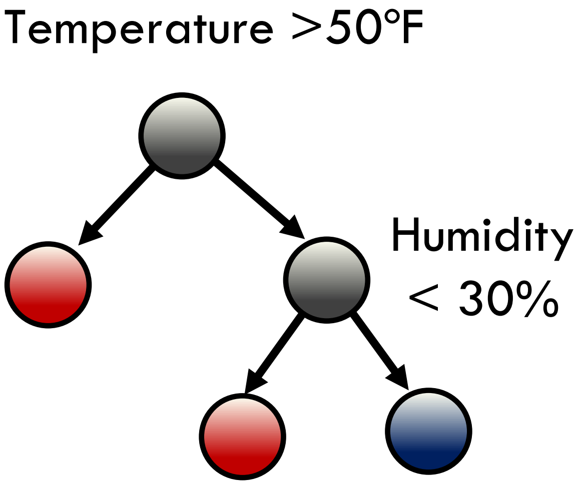


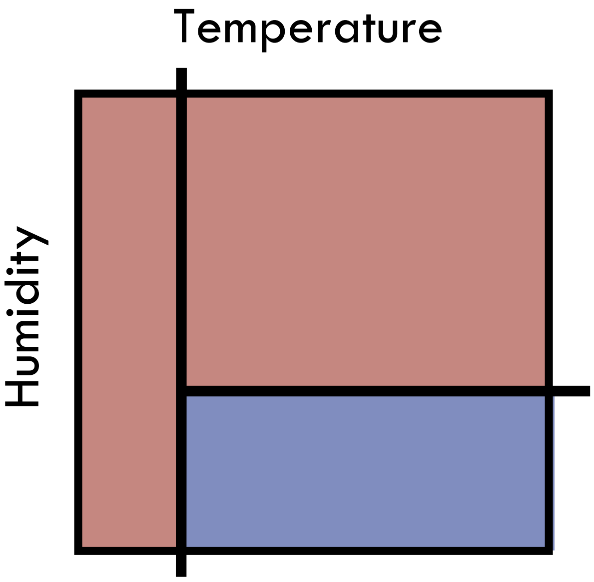
Decision Stump: the Boosting Base Learner





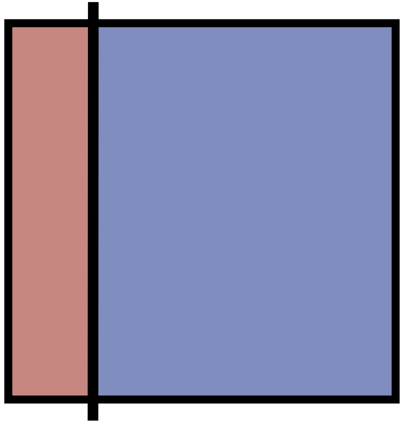
Decision Stump: the Boosting Base Learner





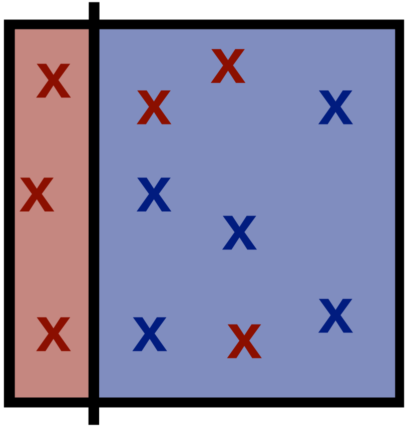
Overview of Boosting

Create initial decision stump



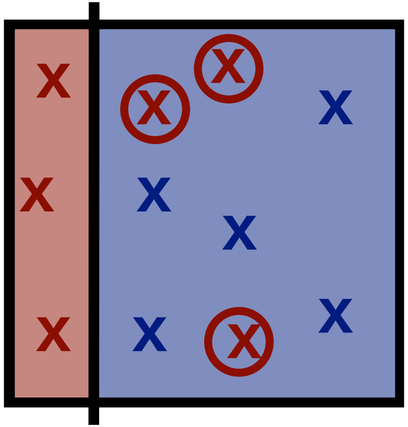
Overview of Boosting

Fit to data and calculate residuals



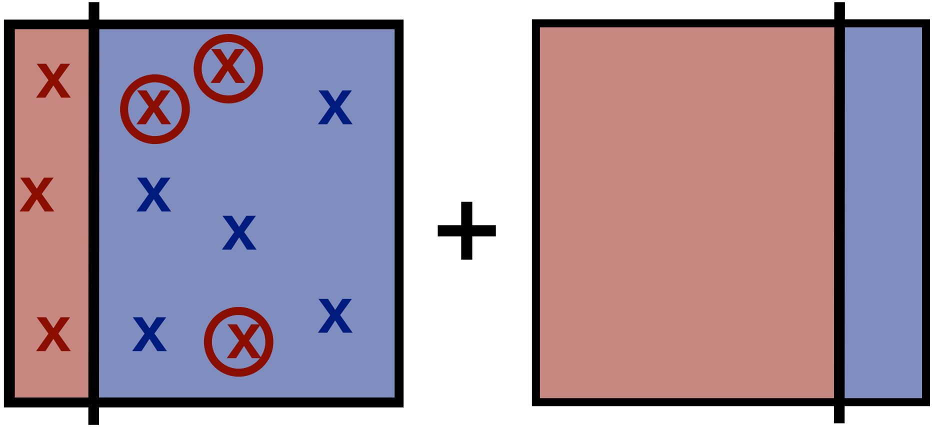
Overview of Boosting

Adjust weight
of points



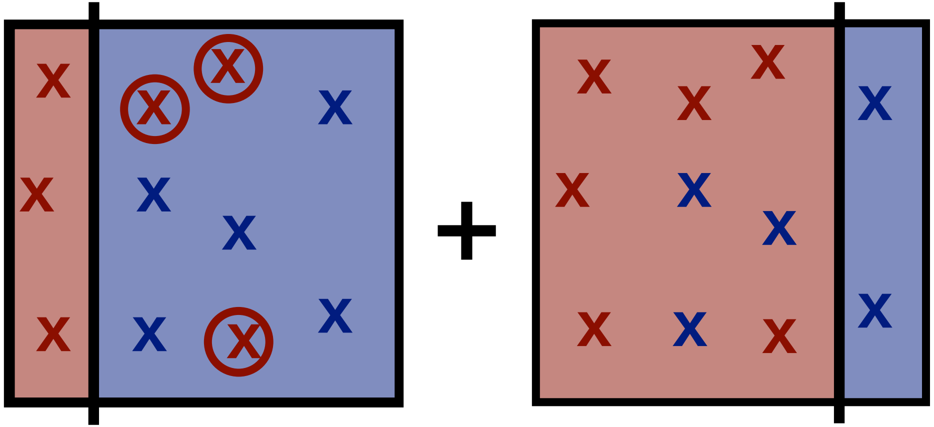
Overview of Boosting

Find new decision stump to fit weighted residuals



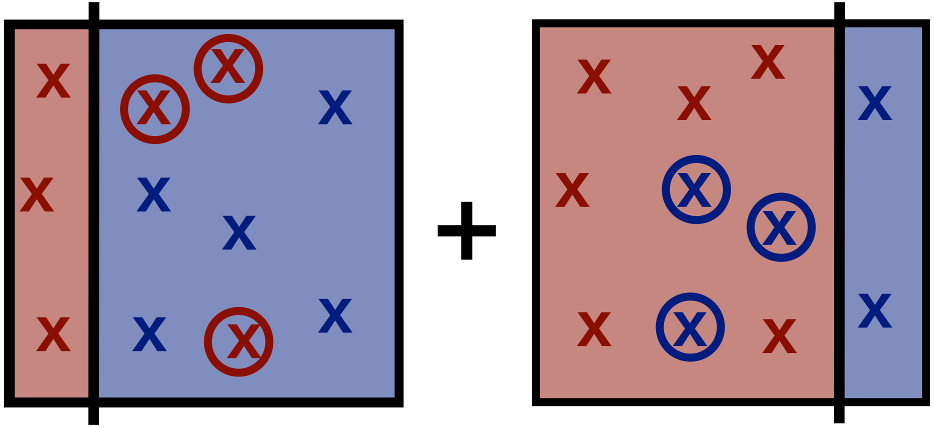
Overview of Boosting

Fit new decision stump to current residuals



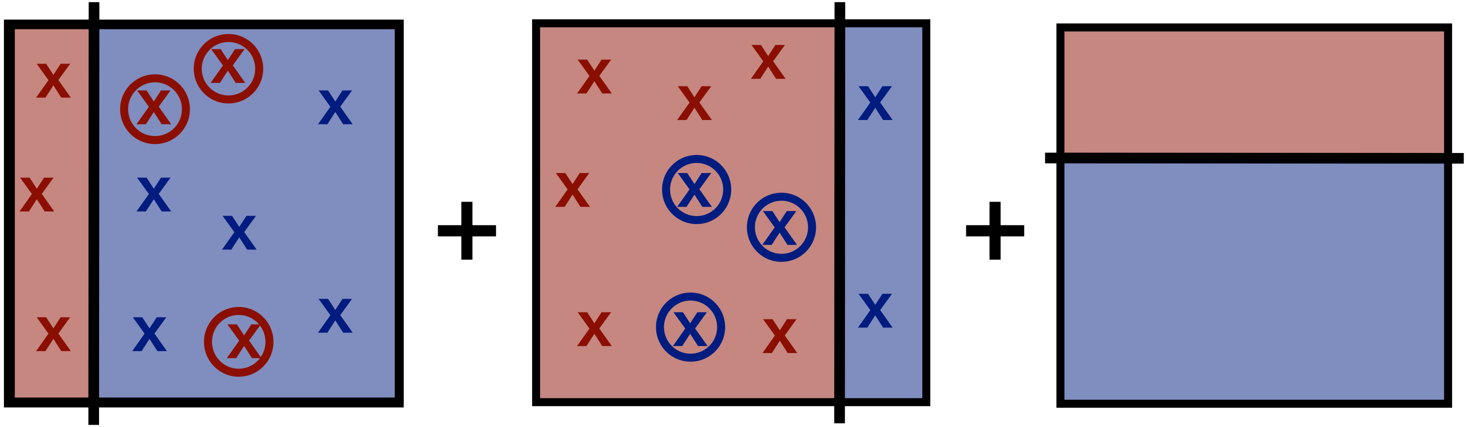
Overview of Boosting

Calculate errors
and weight
data points



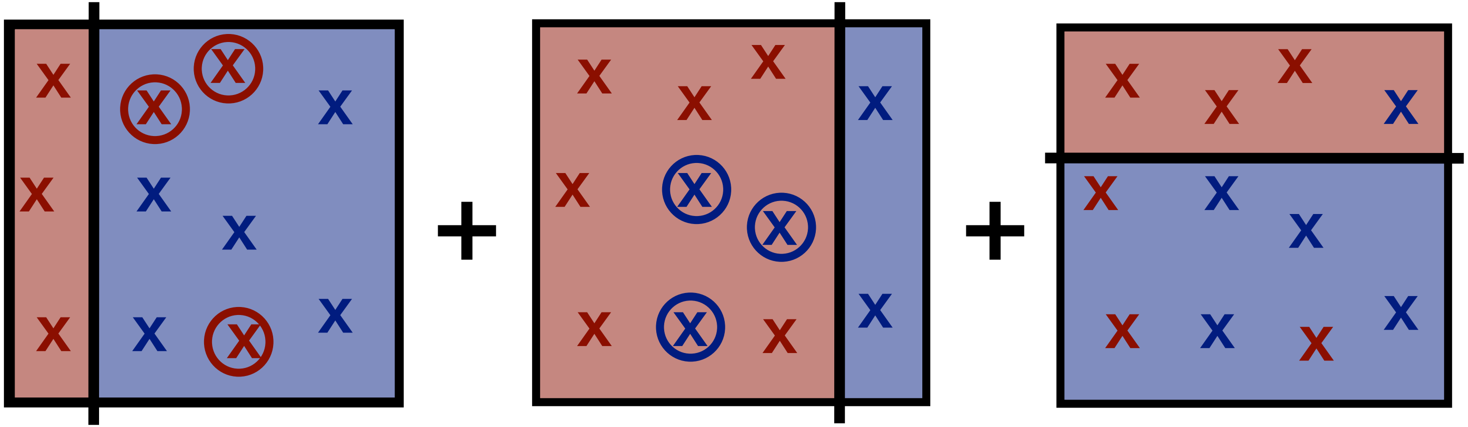
Overview of Boosting

Find new
decision stump
to fit weighted
residuals

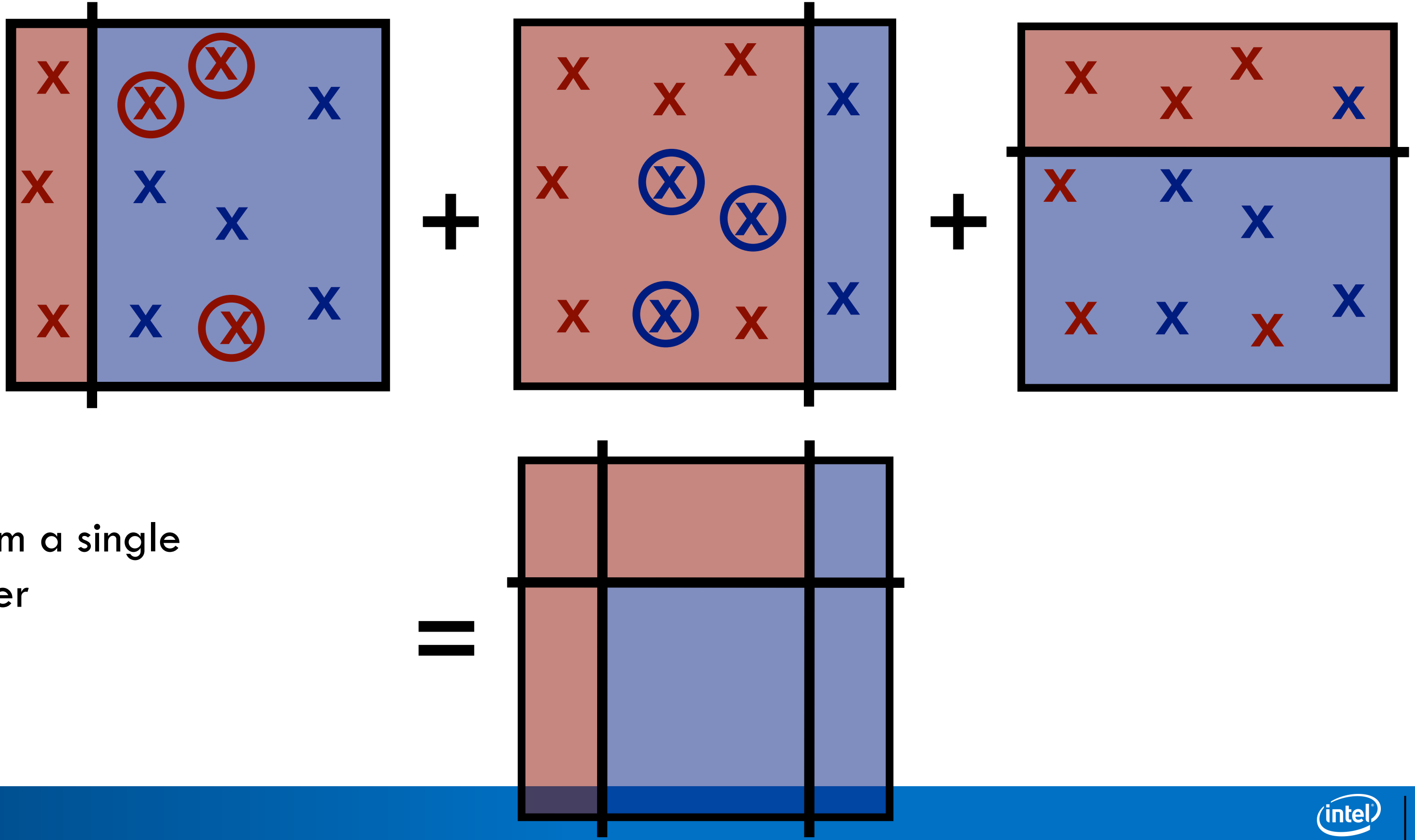


Overview of Boosting

Fit new decision stump to current residuals

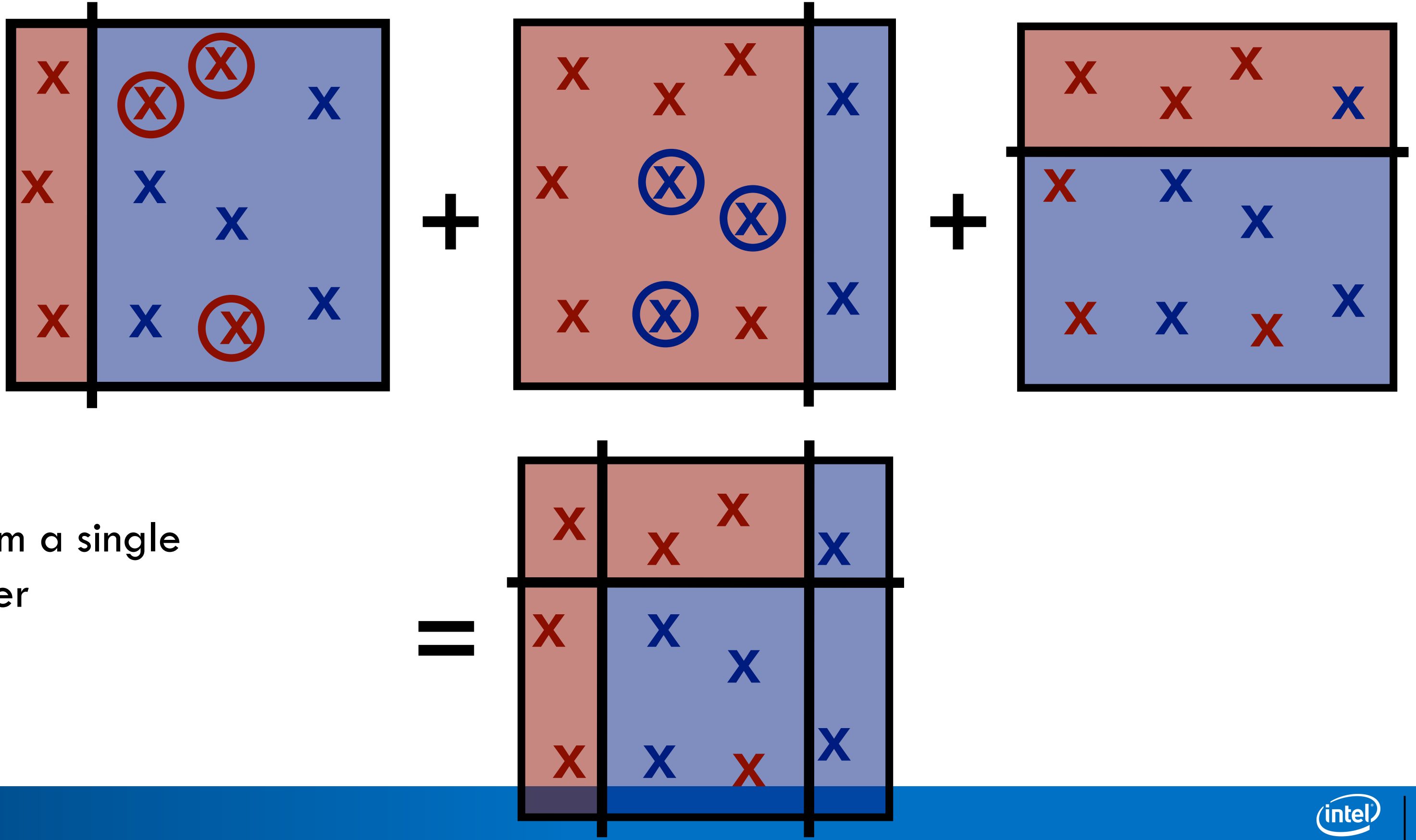


Overview of Boosting



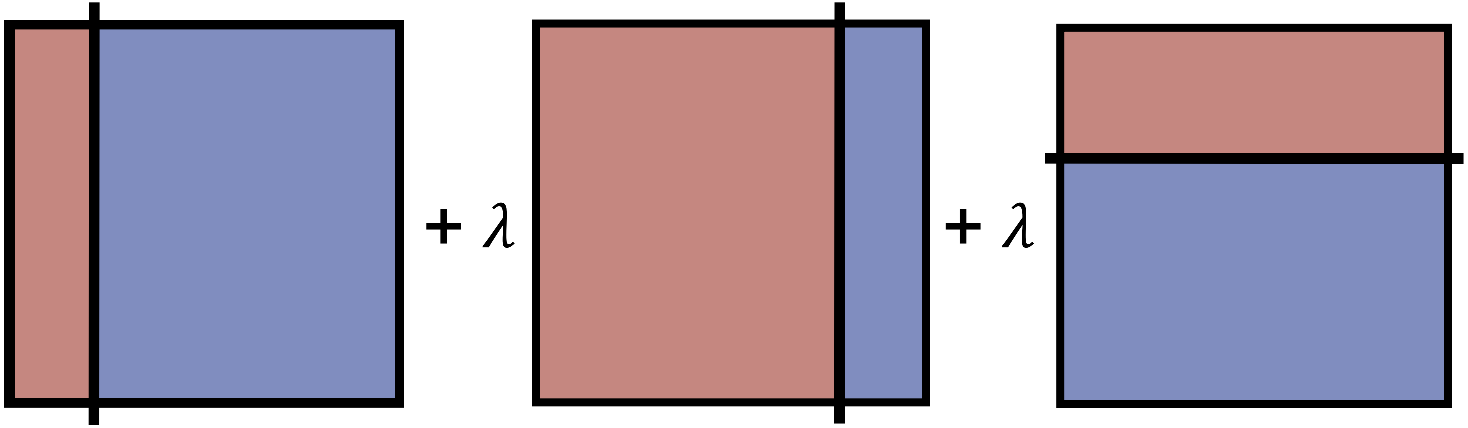
Combine to form a single classifier

Overview of Boosting

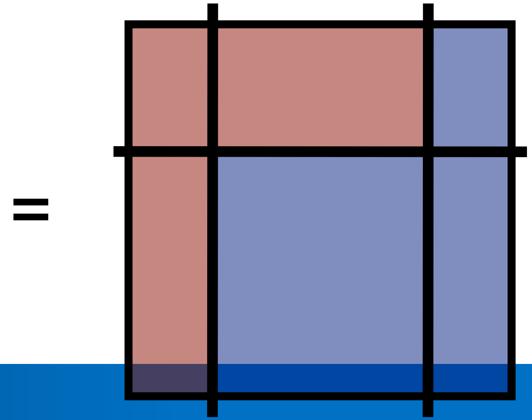


Combine to form a single classifier

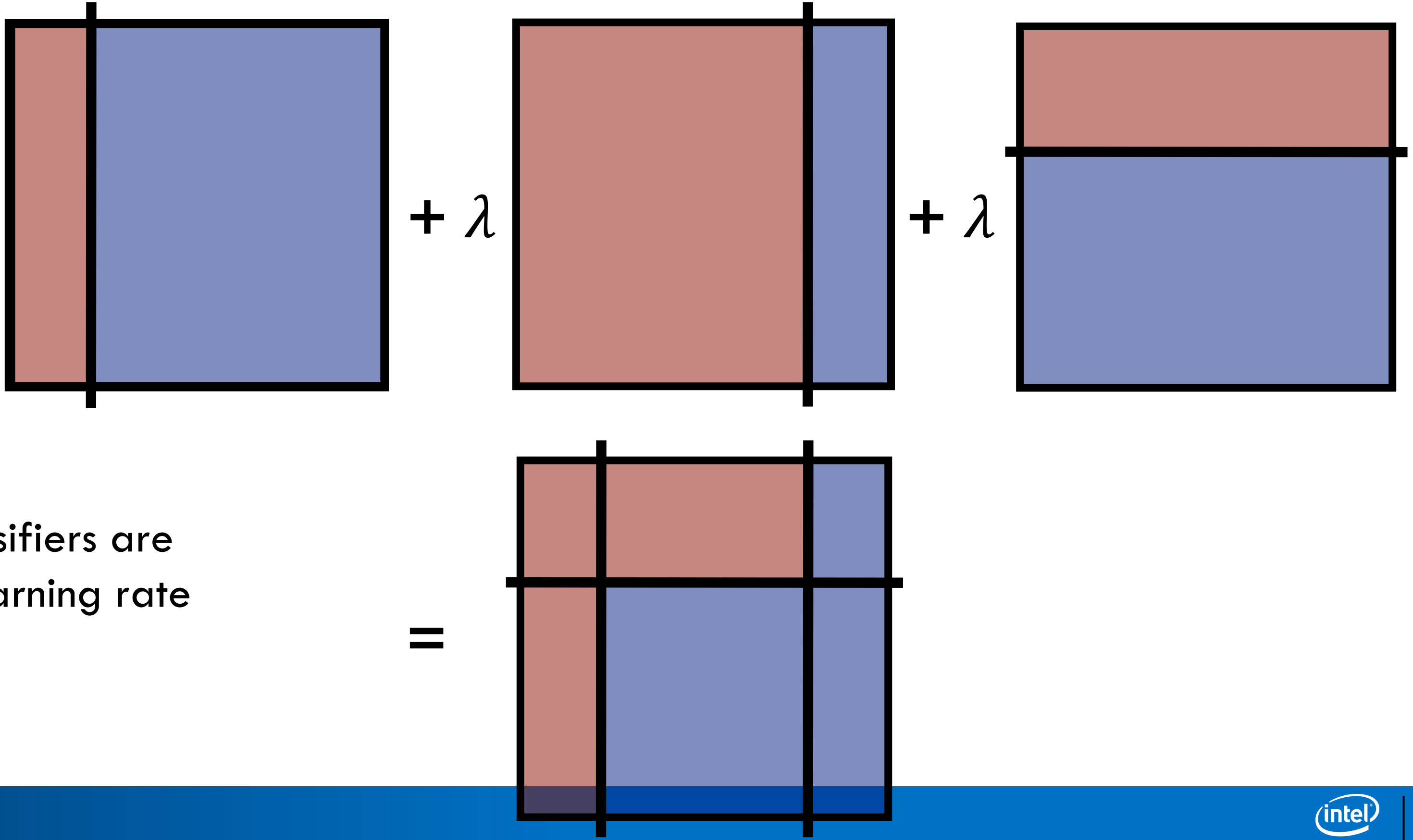
Overview of Boosting



Result is weighted sum of all classifiers

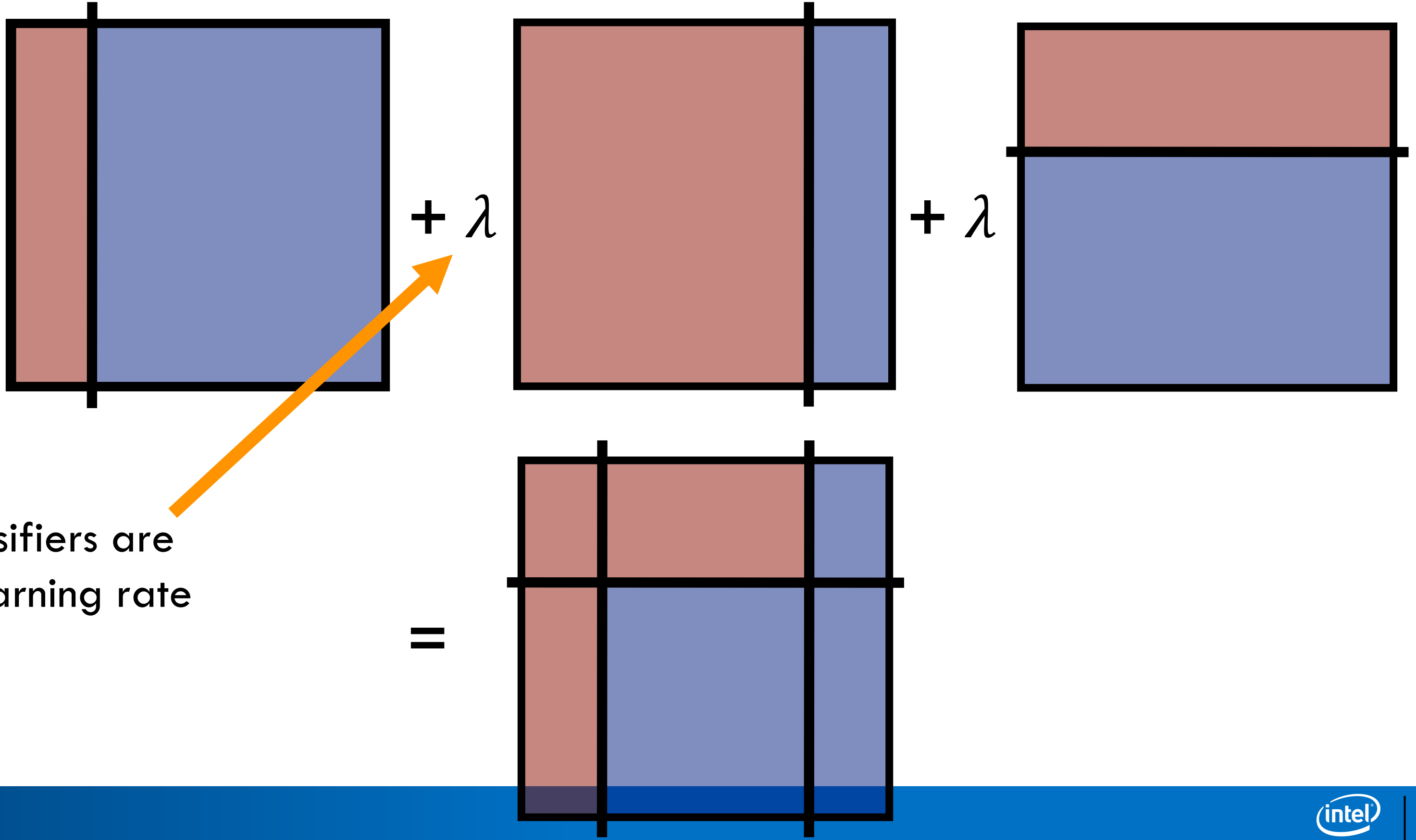


Overview of Boosting



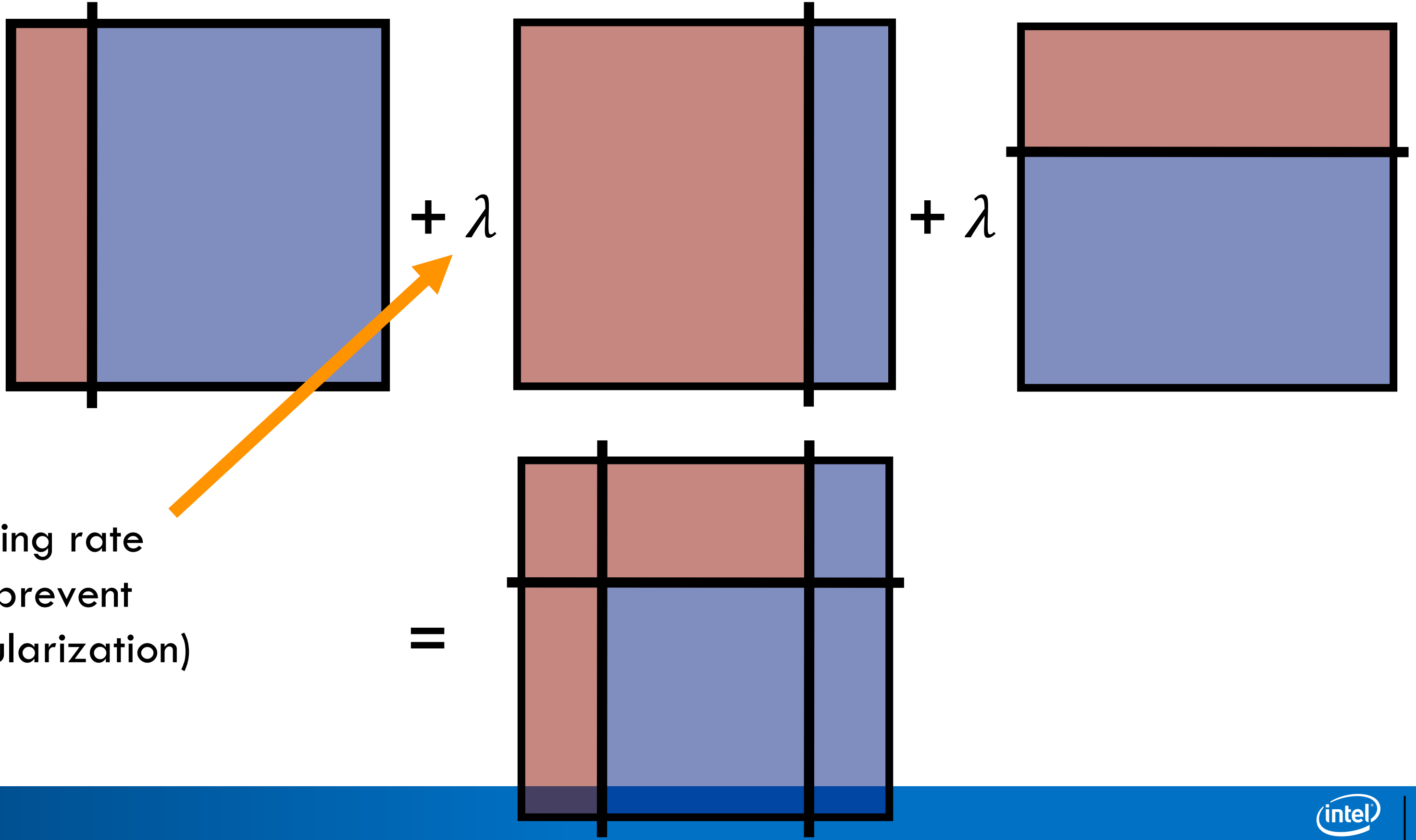
Successive classifiers are weighted by learning rate (λ)

Overview of Boosting



Successive classifiers are weighted by learning rate (λ)

Overview of Boosting



Using a learning rate < 1.0 helps prevent overfitting (regularization)

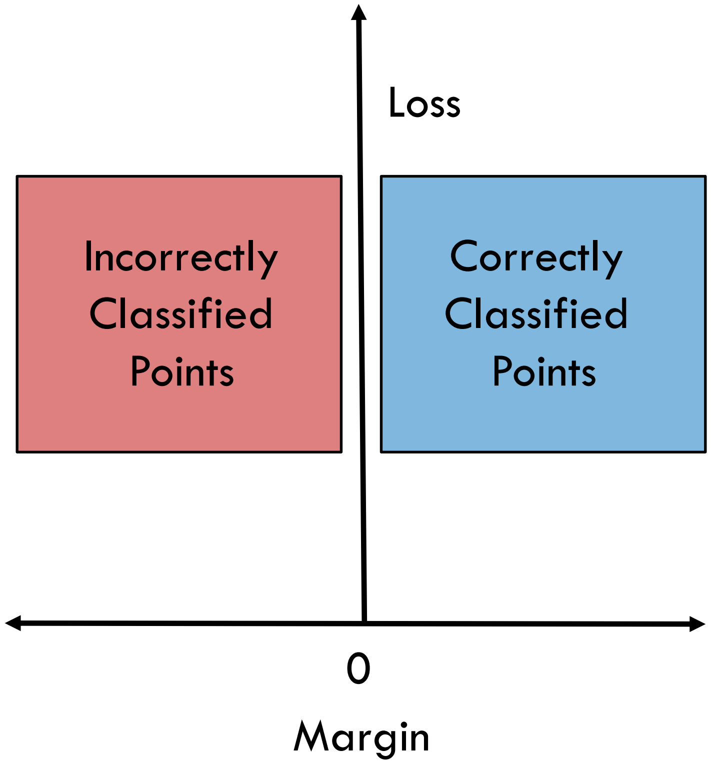
Boosting Specifics

- Boosting utilizes different loss functions

- At each stage, the margin is determined for each point

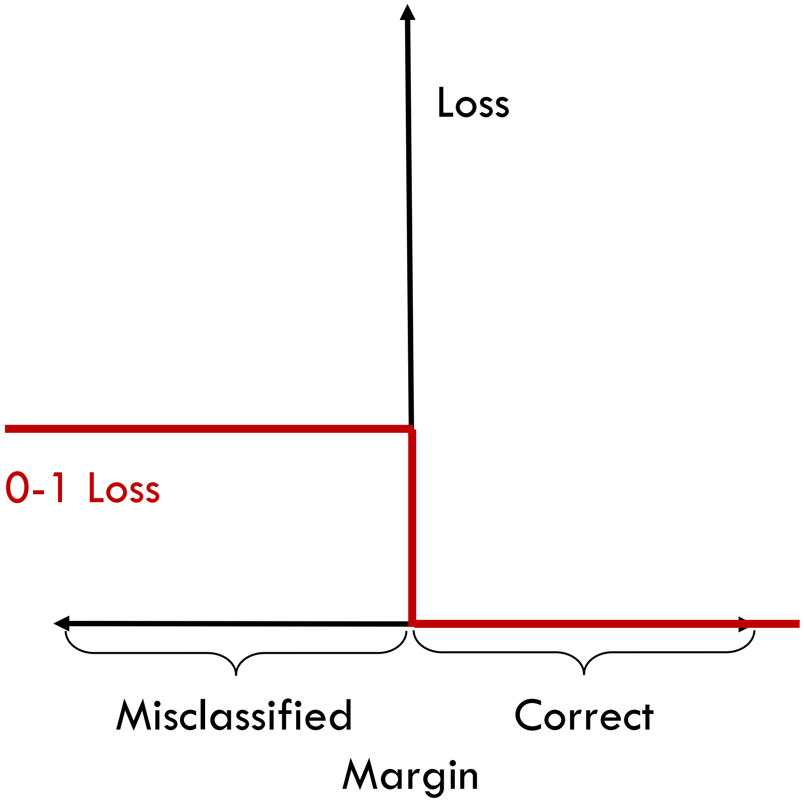
Boosting Specifics

- Boosting utilizes different loss functions
- At each stage, the margin is determined for each point
- Margin is positive for correctly classified points and negative for misclassifications
- Value of loss function is calculated from margin



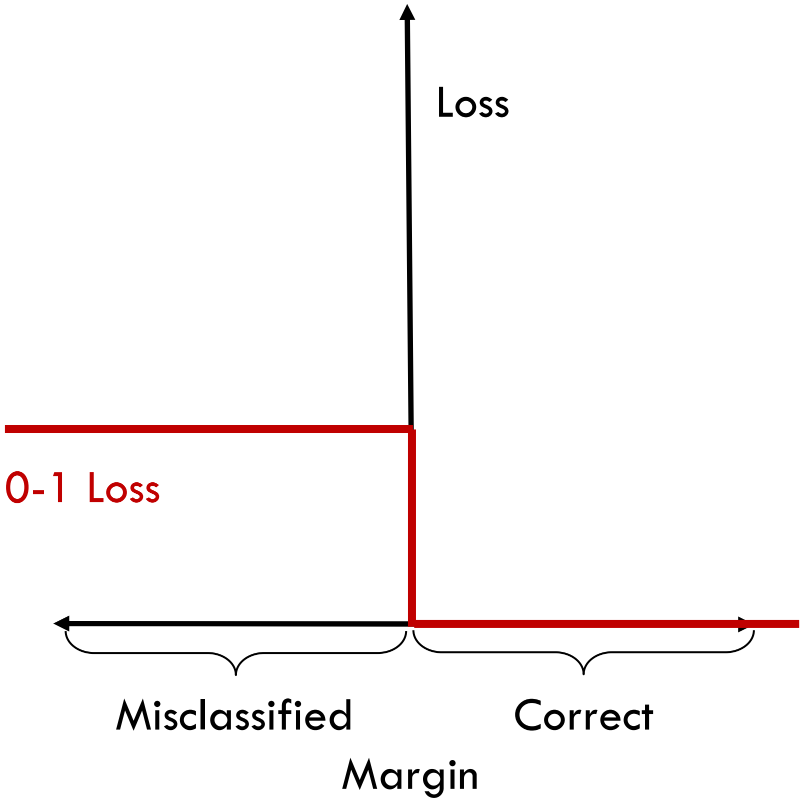
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1



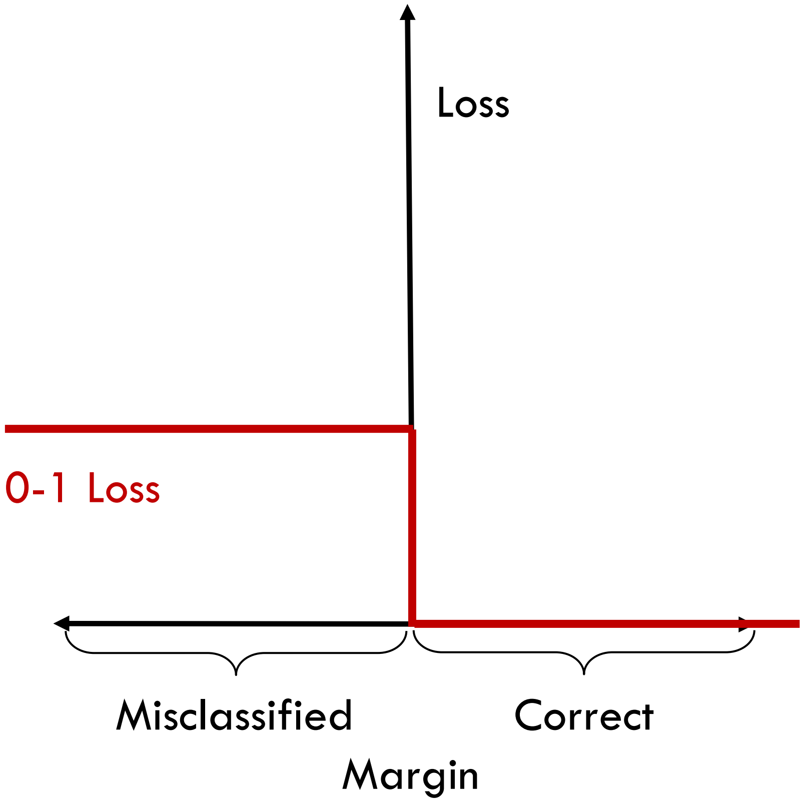
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored



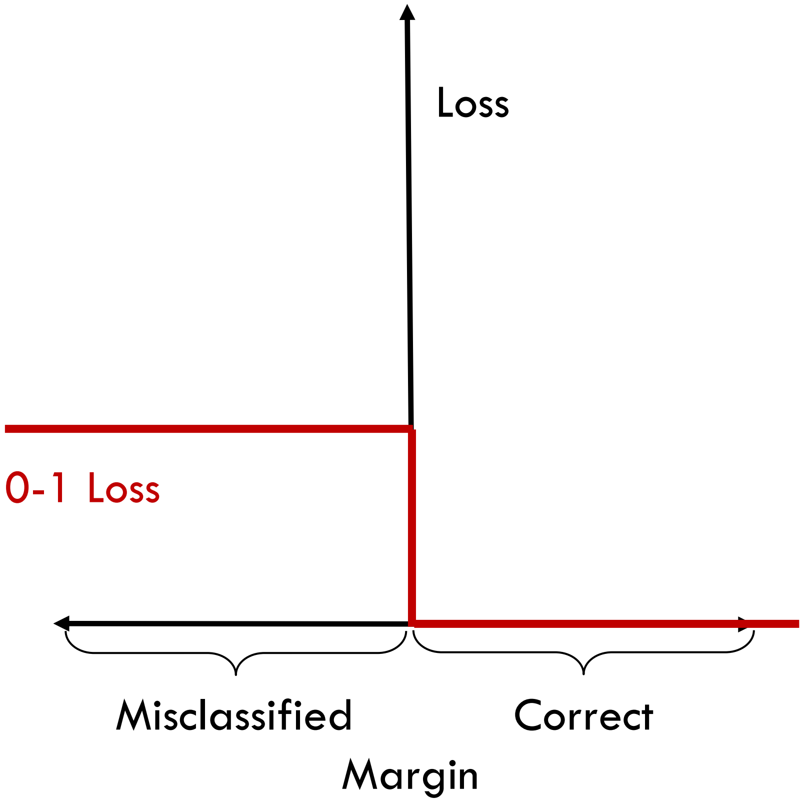
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored
- Theoretical "ideal" loss function
- Difficult to optimize—non-smooth



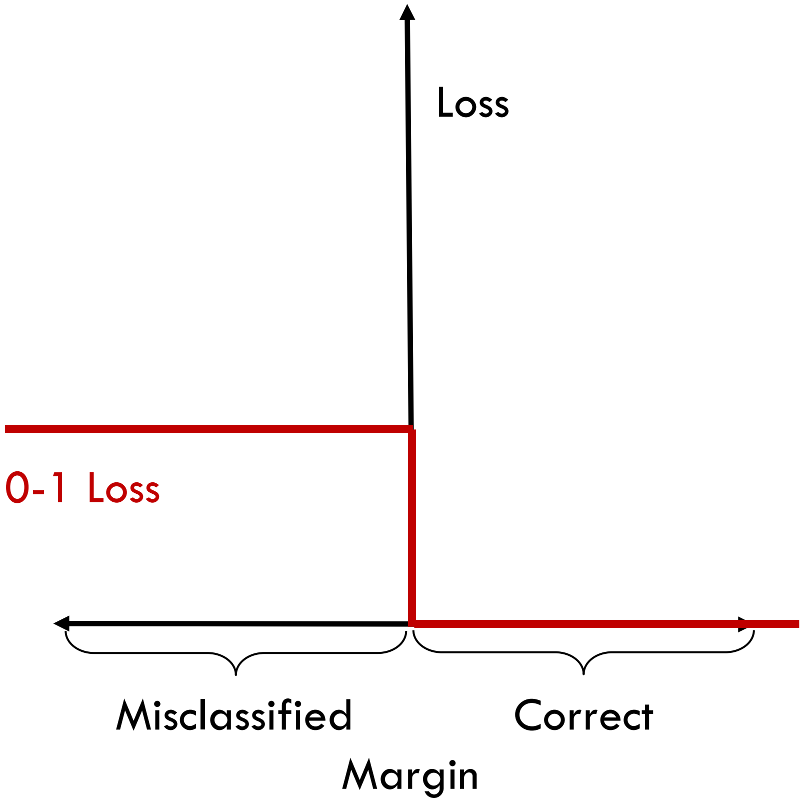
0 – 1 Loss Function

- The 0 – 1 Loss multiplies misclassified points by 1
- Correctly classified points are ignored
- Theoretical "ideal" loss function
- Difficult to optimize—non-smooth and non-convex



AdaBoost Loss Function

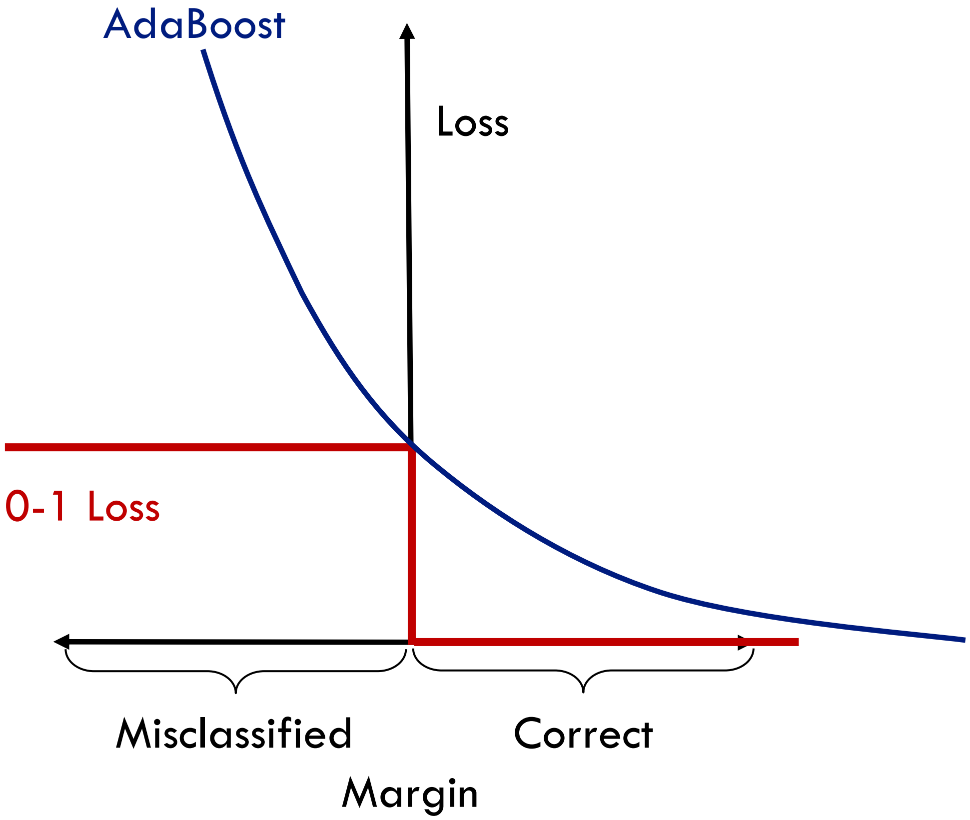
- AdaBoost = Adaptive Boosting



AdaBoost Loss Function

- AdaBoost = Adaptive Boosting
- Loss function is exponential:

$$e^{(-margin)}$$

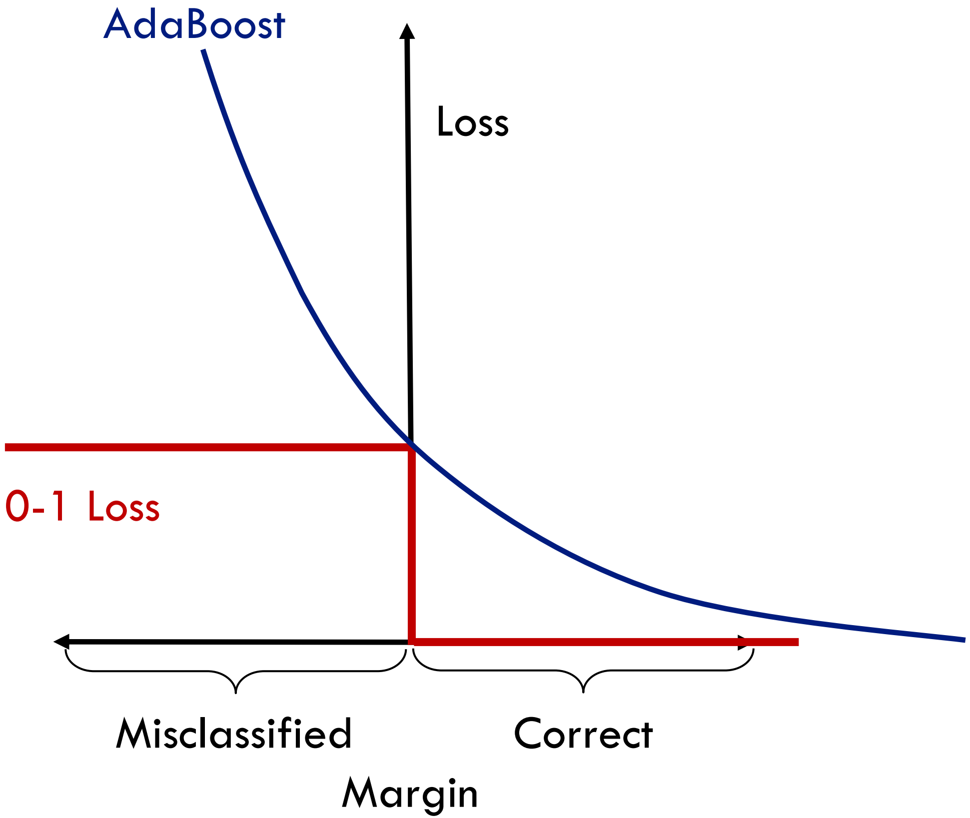


AdaBoost Loss Function

- AdaBoost = Adaptive Boosting
- Loss function is exponential:

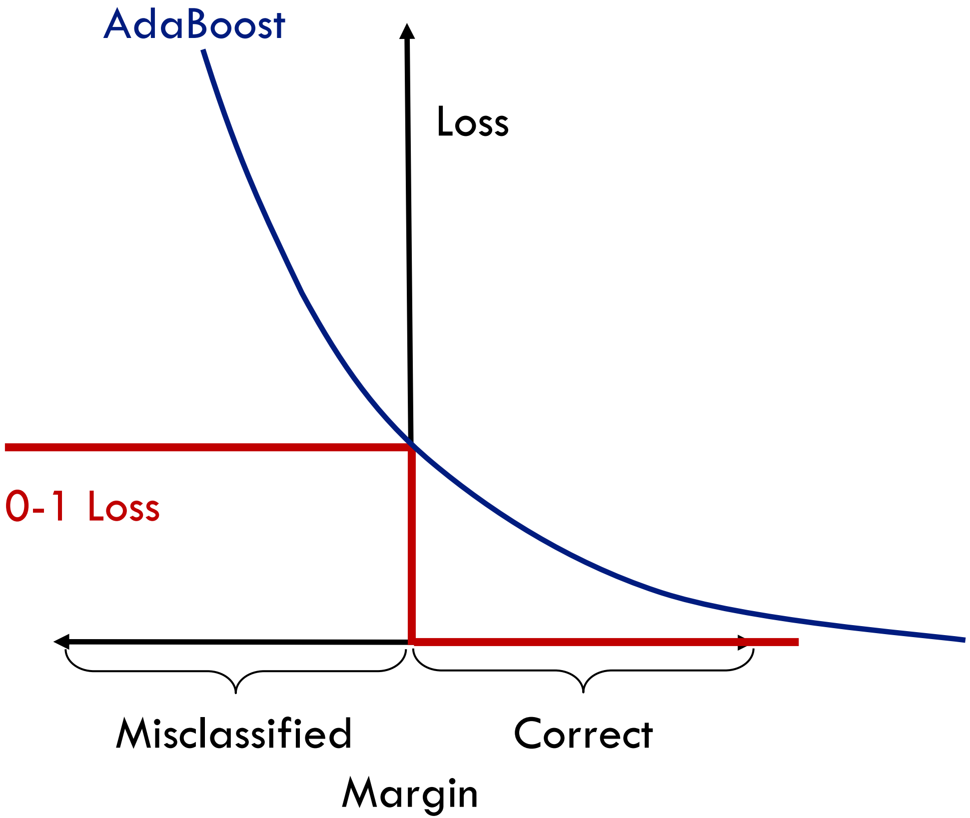
$$e^{(-margin)}$$

- Makes AdaBoost more sensitive to outliers than other types of boosting



Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions
- Common implementation uses

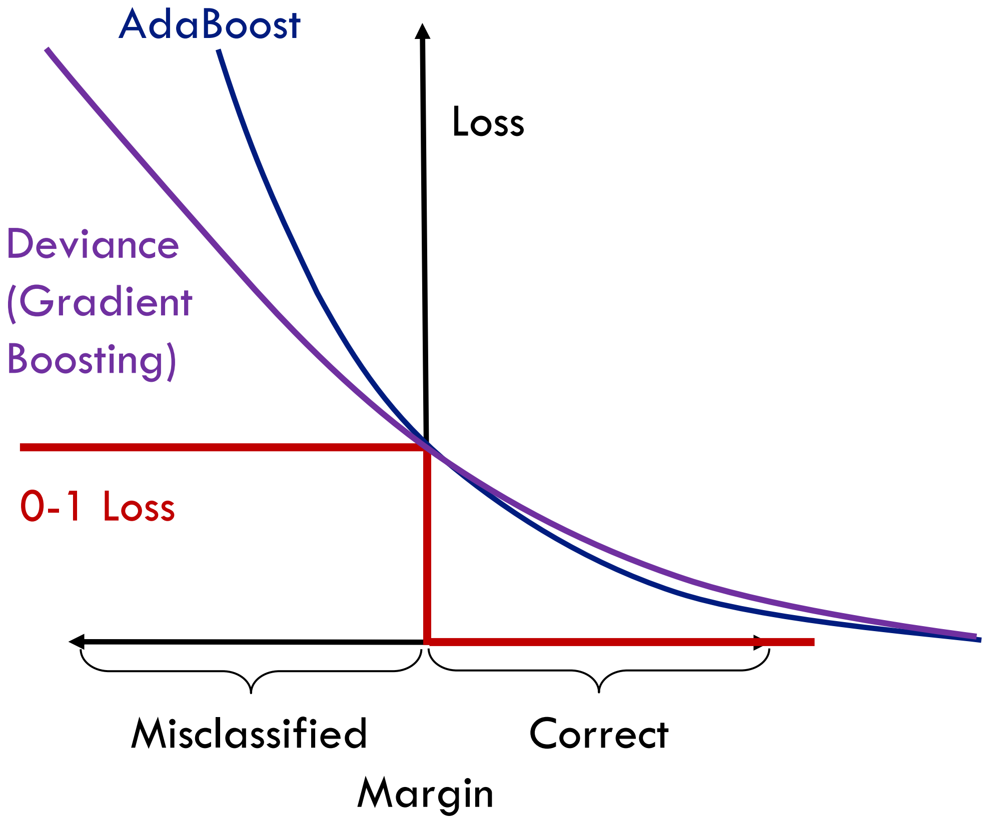


Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions

- Common implementation uses binomial log likelihood loss function (deviance):

$$\log(1 + e^{(-margin)})$$

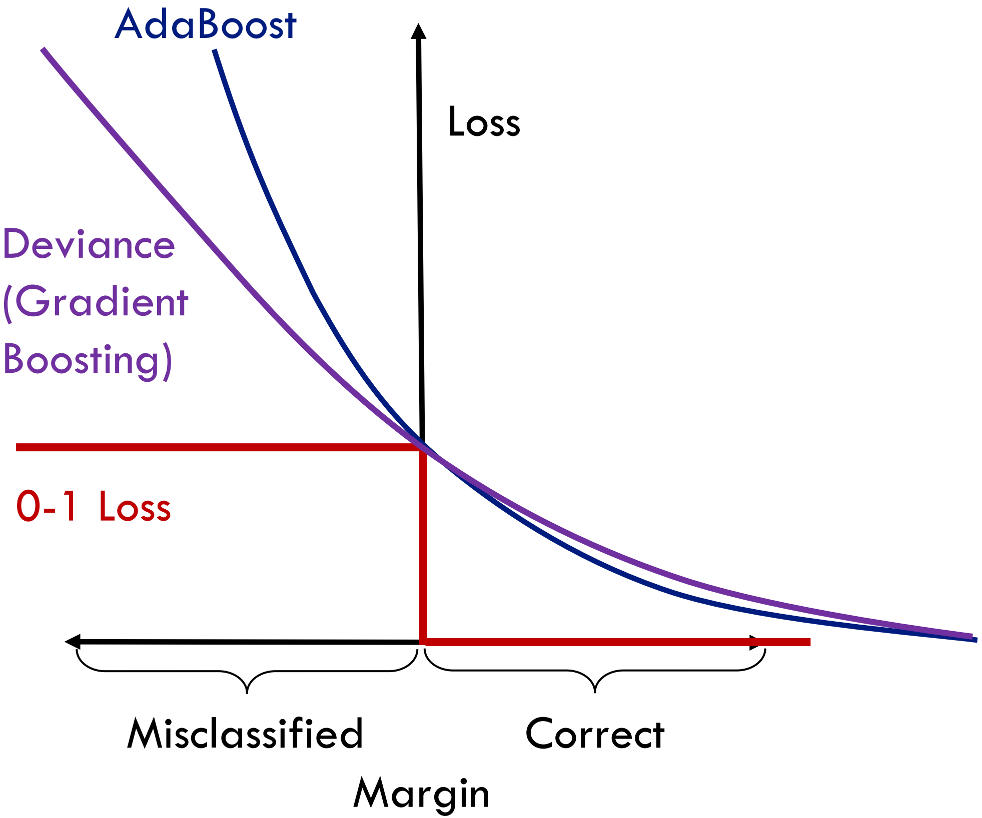


Gradient Boosting Loss Function

- Generalized boosting method that can use different loss functions
- Common implementation uses binomial log likelihood loss function (deviance):

$$\log(1 + e^{(-margin)})$$

- More robust to outliers than AdaBoost



Bagging vs Boosting

Bagging	Boosting
• Bootstrapped samples	• Fit entire data set

Bagging vs Boosting

Bagging

- Bootstrapped samples
- Base trees created independently

Boosting

- Fit entire data set
- Base trees created successively

Bagging vs Boosting

Bagging	Boosting
• Bootstrapped samples	• Fit entire data set
• Base trees created independently	• Base trees created successively
• Only data points considered	• Use residuals from previous models

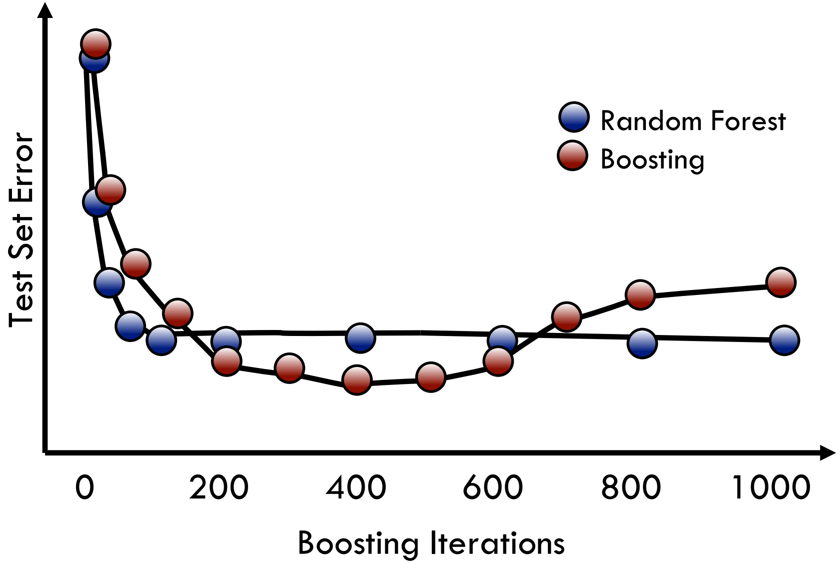
Bagging vs Boosting

Bagging	Boosting
• Bootstrapped samples	• Fit entire data set
• Base trees created independently	• Base trees created successively
• Only data points considered	• Use residuals from previous models
• No weighting used	• Up-weight misclassified points
• Excess trees will not overfit	• Beware of overfitting

Bagging vs Boosting

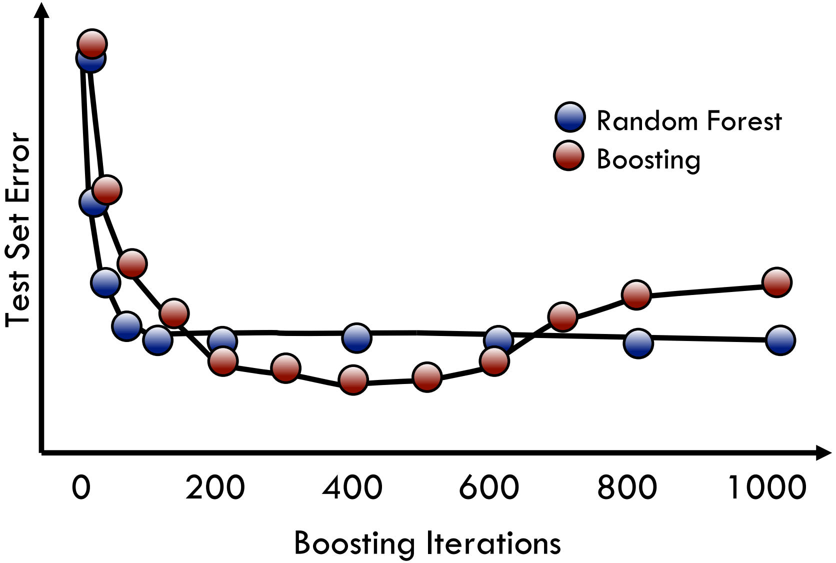
Bagging	Boosting
• Bootstrapped samples	• Fit entire data set
• Base trees created independently	• Base trees created successively
• Only data points considered	• Use residuals from previous models
• No weighting used	• Up-weight misclassified points
• Excess trees will not overfit	• Beware of overfitting

Tuning a Gradient Boosted Model



- Boosting is additive, so possible to overfit

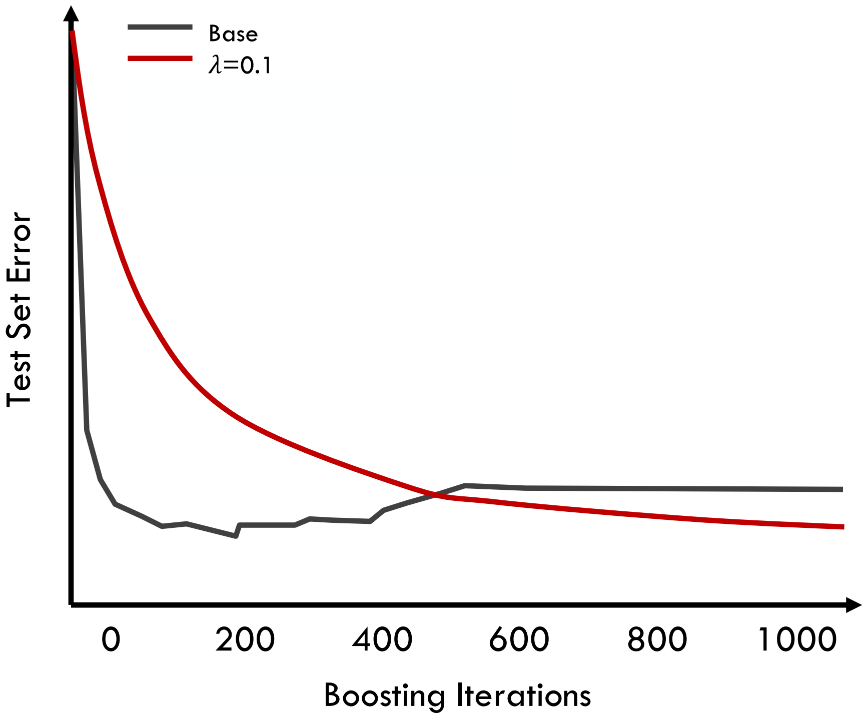
Tuning a Gradient Boosted Model



- Boosting is additive, so possible to overfit

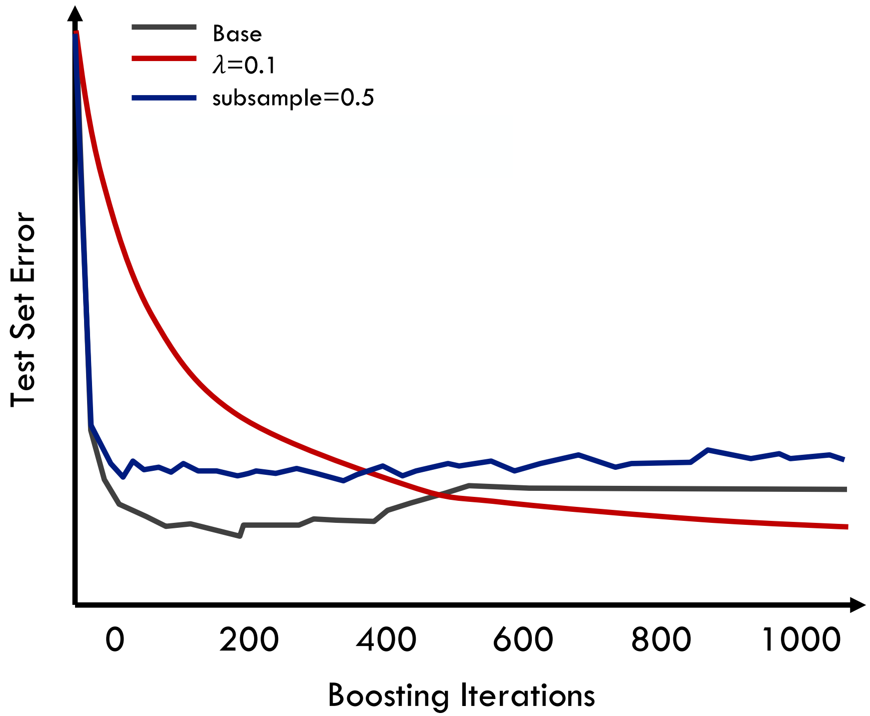
- Use cross validation to set number of trees

Tuning a Gradient Boosted Model



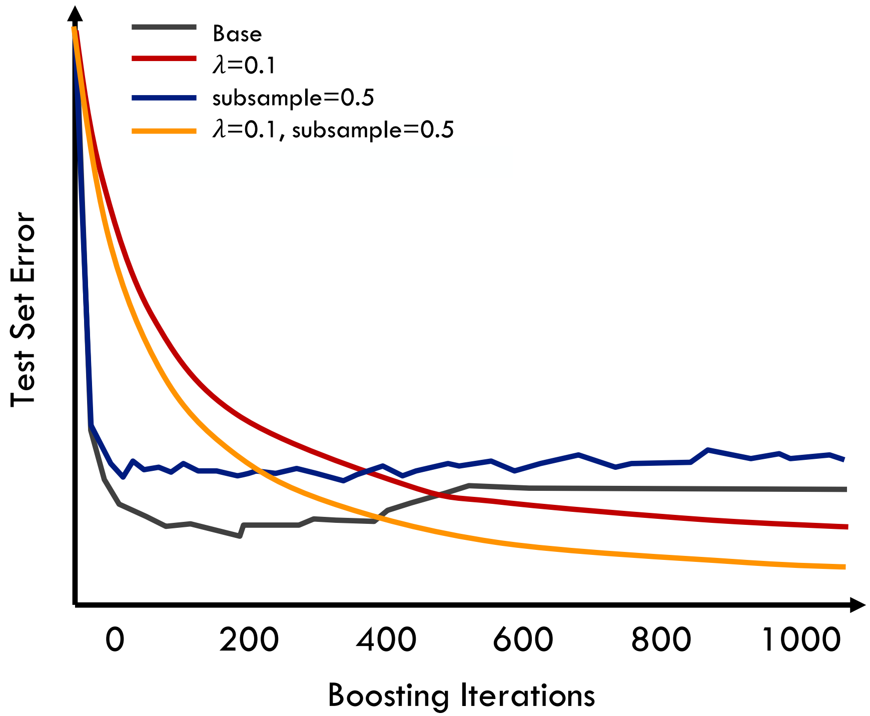
- **Learning rate (λ)**: set to <1.0 for regularization. That's also called "shrinkage"

Tuning a Gradient Boosted Model



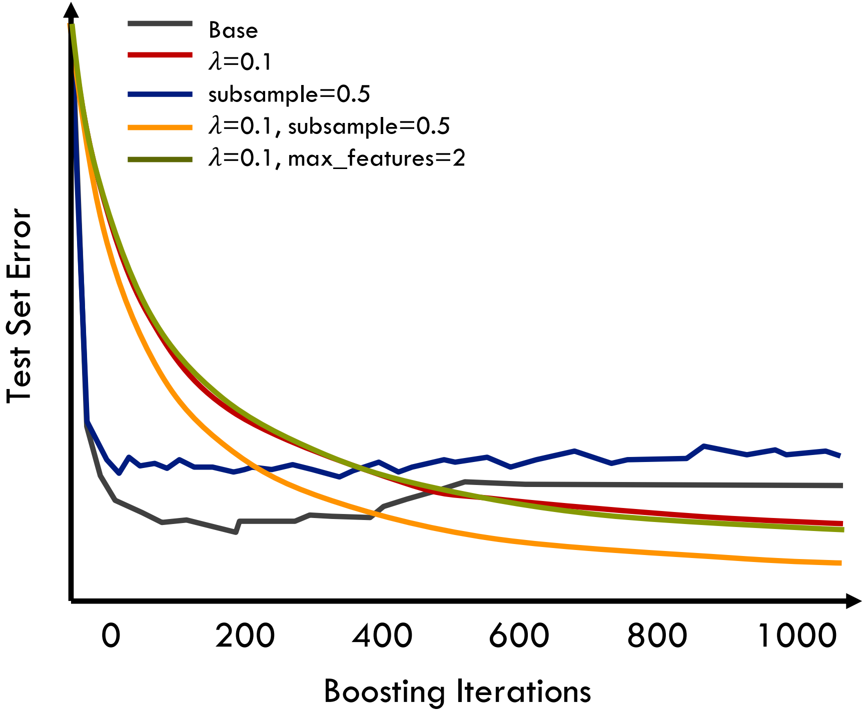
- **Learning rate (λ)**: set to <1.0 for regularization. That's also called "shrinkage"
- **Subsample:** set to <1.0 to use fraction of data for base learners (stochastic gradient boosting)

Tuning a Gradient Boosted Model



- **Learning rate (λ):** set to <1.0 for regularization. That's also called "shrinkage"
- **Subsample:** set to <1.0 to use fraction of data for base learners (stochastic gradient boosting)

Tuning a Gradient Boosted Model



- **Learning rate (λ):** set to <1.0 for regularization. That's also called "shrinkage"
- **Subsample:** set to <1.0 to use fraction of data for base learners (stochastic gradient boosting)
- **Max_features:** number of features to consider in base learners when splitting.

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,
                max_features=1, subsample=0.5,
 n_estimators=200)
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,
                max_features=1, subsample=0.5,
 n_estimators=200)
```

Fit the instance on the data and then predict the expected value

```
GBC = GBC.fit (X_train, y_train)
y_predict = GBC.predict(X_test)
```

GradientBoostingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import GradientBoostingClassifier
```

Create an instance of the class

```
GBC = GradientBoostingClassifier(learning_rate=0.1,
                max_features=1, subsample=0.5,
 n_estimators=200)
```

Fit the instance on the data and then predict the expected value

```
GBC = GBC.fit (X_train, y_train)
y_predict = GBC.predict(X_test)
```

Tune with cross-validation. Use GradientBoostingRegressor for regression.

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

To use the Intel® Extension for Scikit-learn* variant of this algorithm:

- **Install Intel® oneAPI AI Analytics Toolkit (AI Kit)**
- **Add the following two lines of code after the above code:**

```
import patch_sklearn
patch_sklearn()
```

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
                          learning_rate=0.1, n_estimators=200)
```

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
                         learning_rate=0.1, n_estimators=200)
```

base learner can be set manually

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
                         learning_rate=0.1, n_estimators=200)
```

can also set max depth here

AdaBoostClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

Create an instance of the class

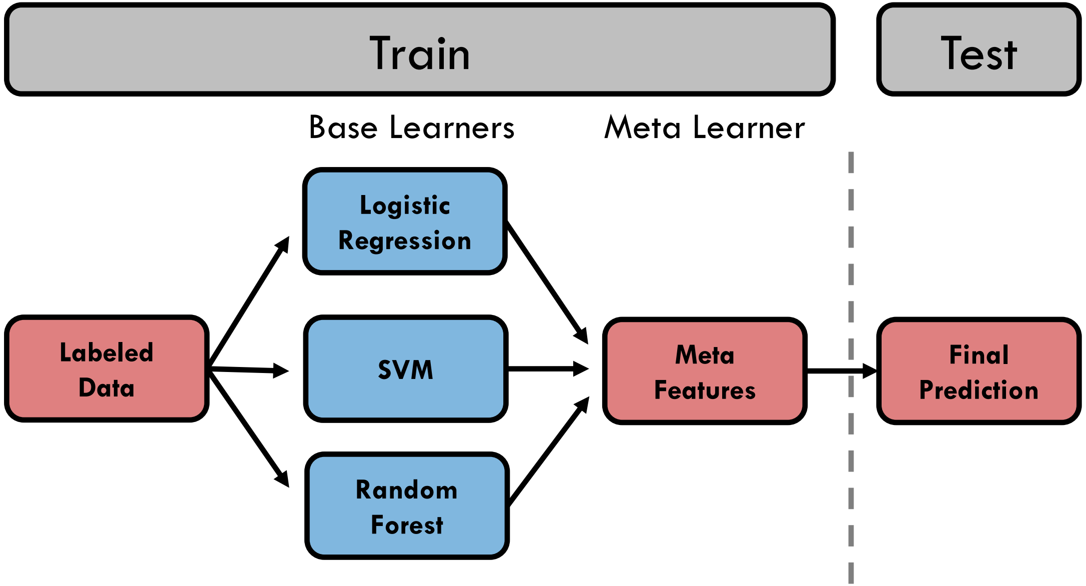
```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
                         learning_rate=0.1, n_estimators=200)
```

Fit the instance on the data and then predict the expected value

```
ABC = ABC.fit(X_train, y_train)
y_predict = ABC.predict(X_test)
```

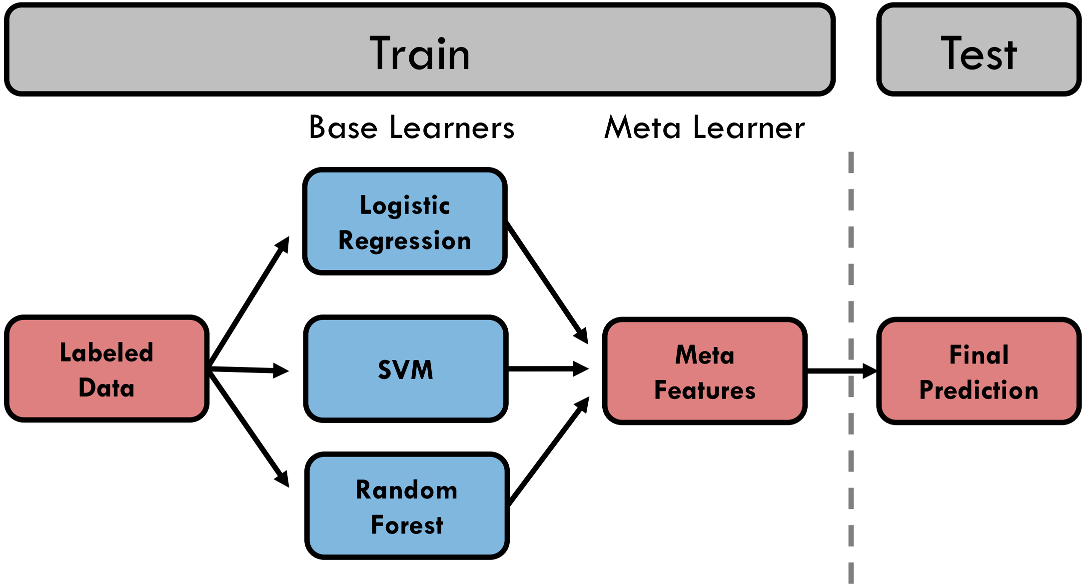
Tune parameters with cross-validation. Use AdaBoostRegressor for regression.

Stacking: Combining Heterogeneous Classifiers



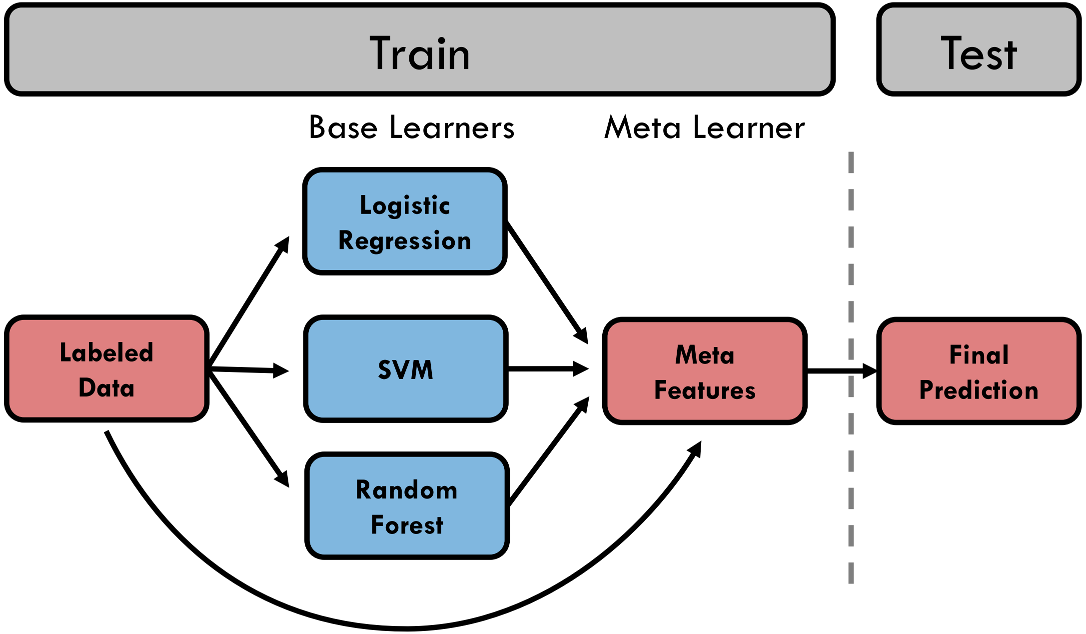
- Models of any kind combined to create stacked model

Stacking: Combining Heterogeneous Classifiers



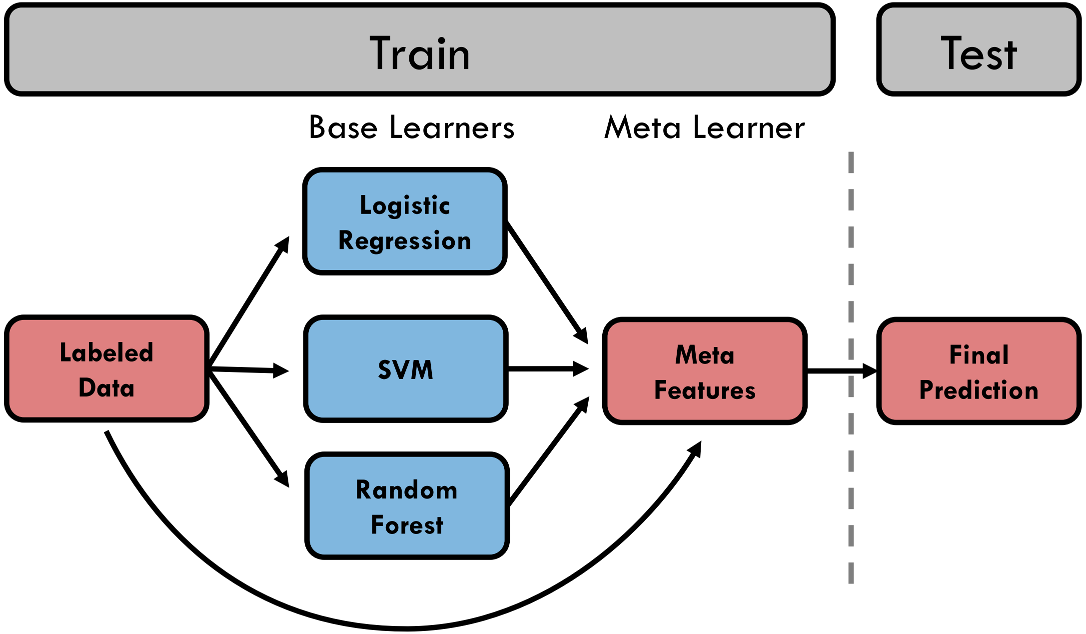
- Models of any kind combined to create stacked model
- Like bagging but not limited to decision trees

Stacking: Combining Heterogeneous Classifiers



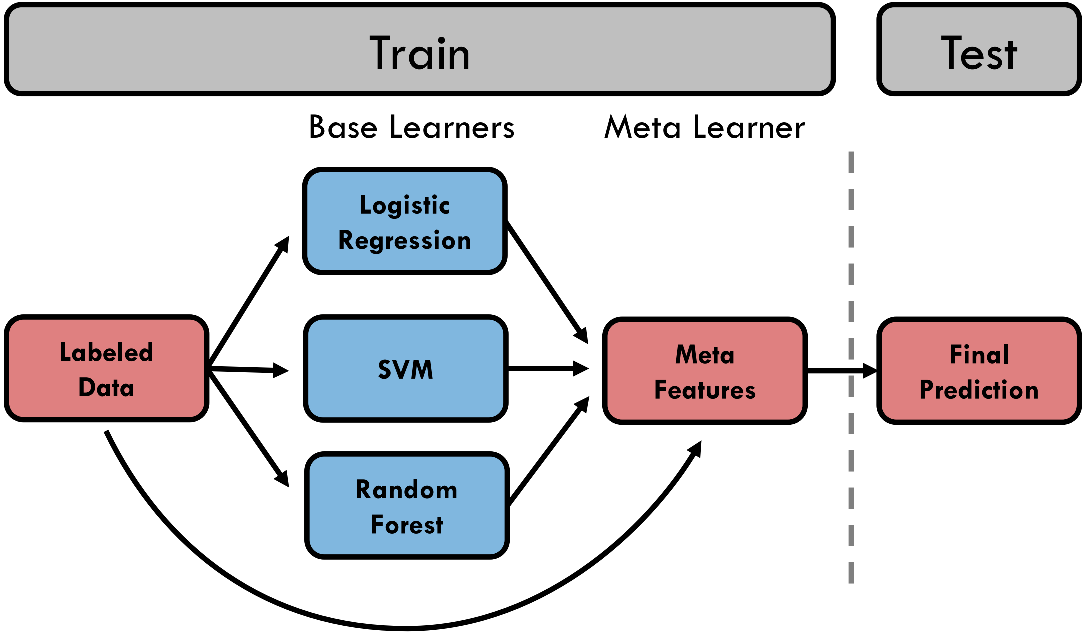
- Models of any kind combined to create stacked model
- Like bagging but not limited to decision trees
- Output of base learners creates features, can recombine with data

Stacking: Combining Heterogeneous Classifiers



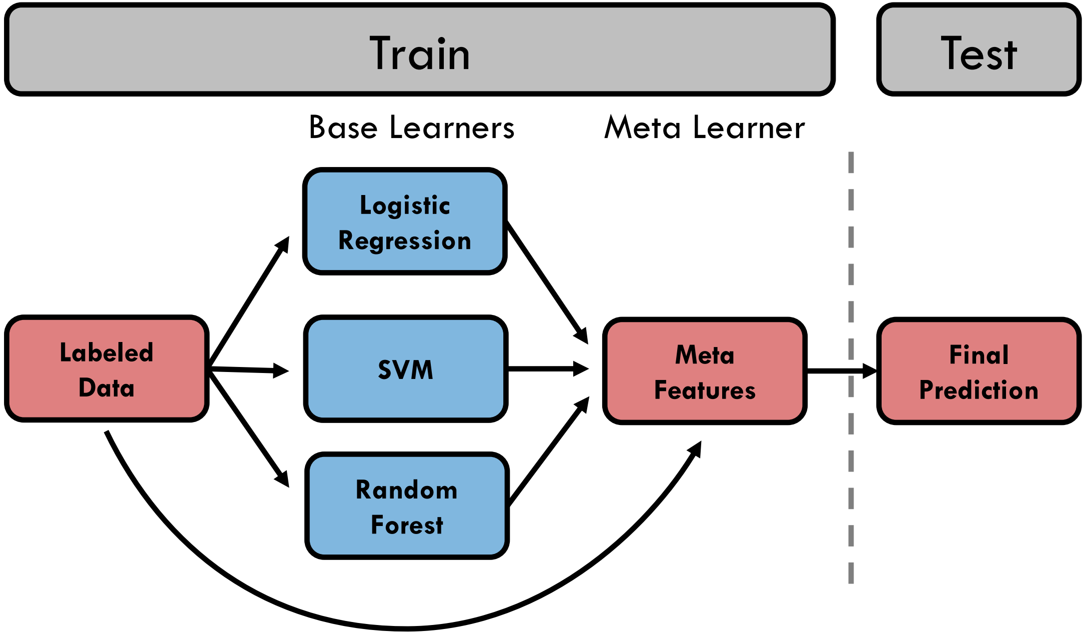
- Output of base learners can be combined via majority vote or weighted

Stacking: Combining Heterogeneous Classifiers



- Output of base learners can be combined via majority vote or weighted
- Additional hold-out data needed if meta learner parameters are used

Stacking: Combining Heterogeneous Classifiers



- Output of base learners can be combined via majority vote or weighted
- Additional hold-out data needed if meta learner parameters are used
- Be aware of increasing model complexity

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',
                      weights=estimator_weight_list)
```

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',
                      weights=estimator_weight_list)
```

list of fitted models and how to combine

VotingClassifier: The Syntax

Import the class containing the classification method

```
from sklearn.ensemble import VotingClassifier
```

Create an instance of the class

```
VC = VotingClassifier(estimator_list, voting='hard',
                      weights=estimator_weight_list)
```

Fit the instance on the data and then predict the expected value

```
VC = VC.fit(X_train, y_train)
y_predict = VC.predict(X_test)
```

Tune with an ADDITIONAL LEVEL of cross-validation or hold-out set.

Software