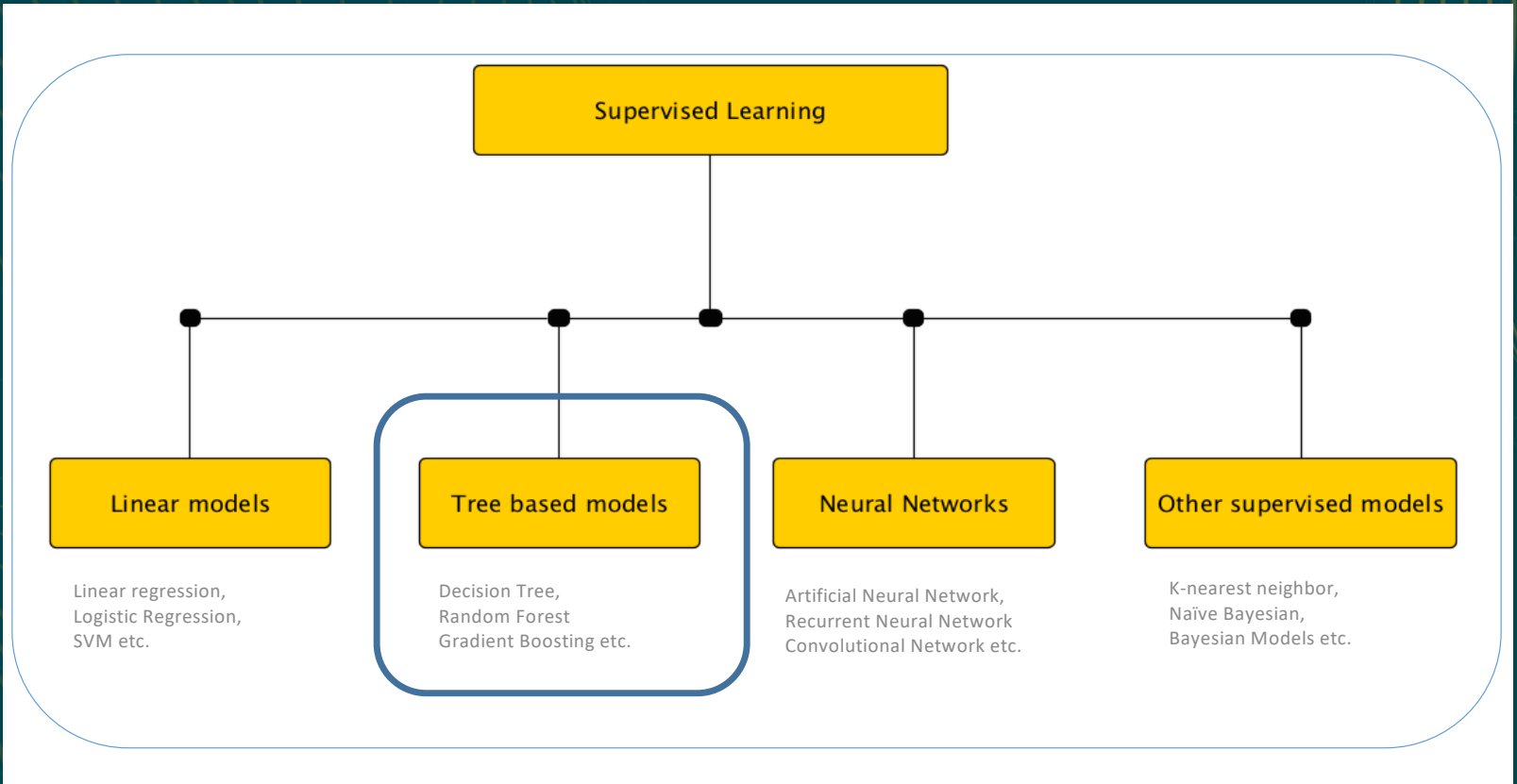
360° view.

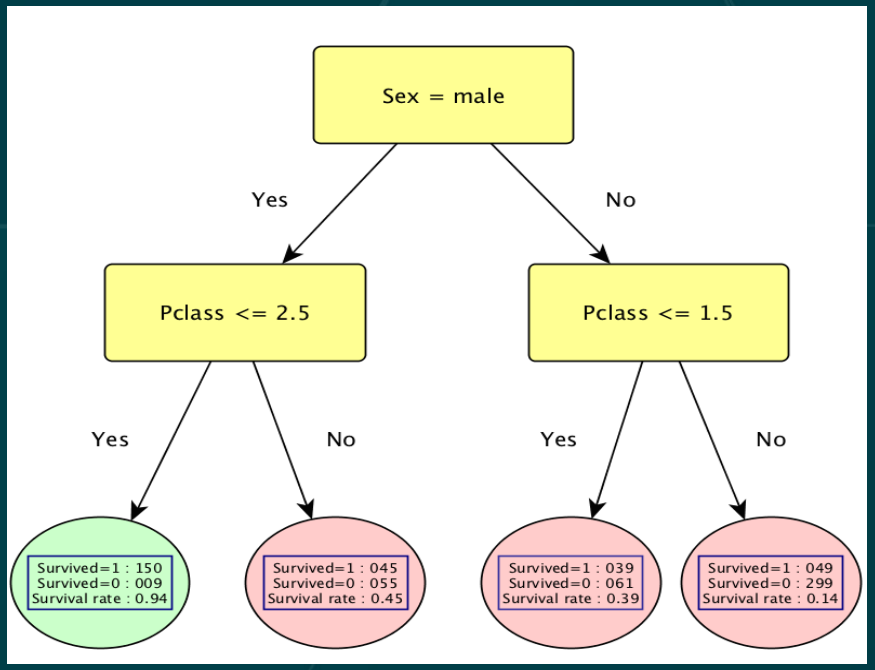

Supervised Learning
Linear models
Linear regression,
Logistic Regression,
SVM etc.
Tree based models
Decision Tree,
Random Forest
Gradient Boosting etc.
Neural Networks
Artificial Neural Network,
Recurrent Neural Network
Convolutional Network etc.
Other supervised models
K-nearest neighbor,
Naïve Bayesian,
Bayesian Models etc.

Decision Trees

Decision Trees are a non-parametric supervised method that predicts the target variable by learning simple decision rules inferred from the training data.

ID	Pclass	Sex	Age	Fare	Survived
0	3	male	22	7.3	0
1	1	female	38	71.3	1
2	3	female	26	7.9	1
3	1	female	35	53.1	1
4	3	male	35	8.1	0
5	1	male	54	51.9	0
6	3	male	2	21.1	0
7	3	female	27	11.1	1
8	2	female	14	30.1	1
9	3	female	4	16.7	1

Training Data



Model : Decision Tree

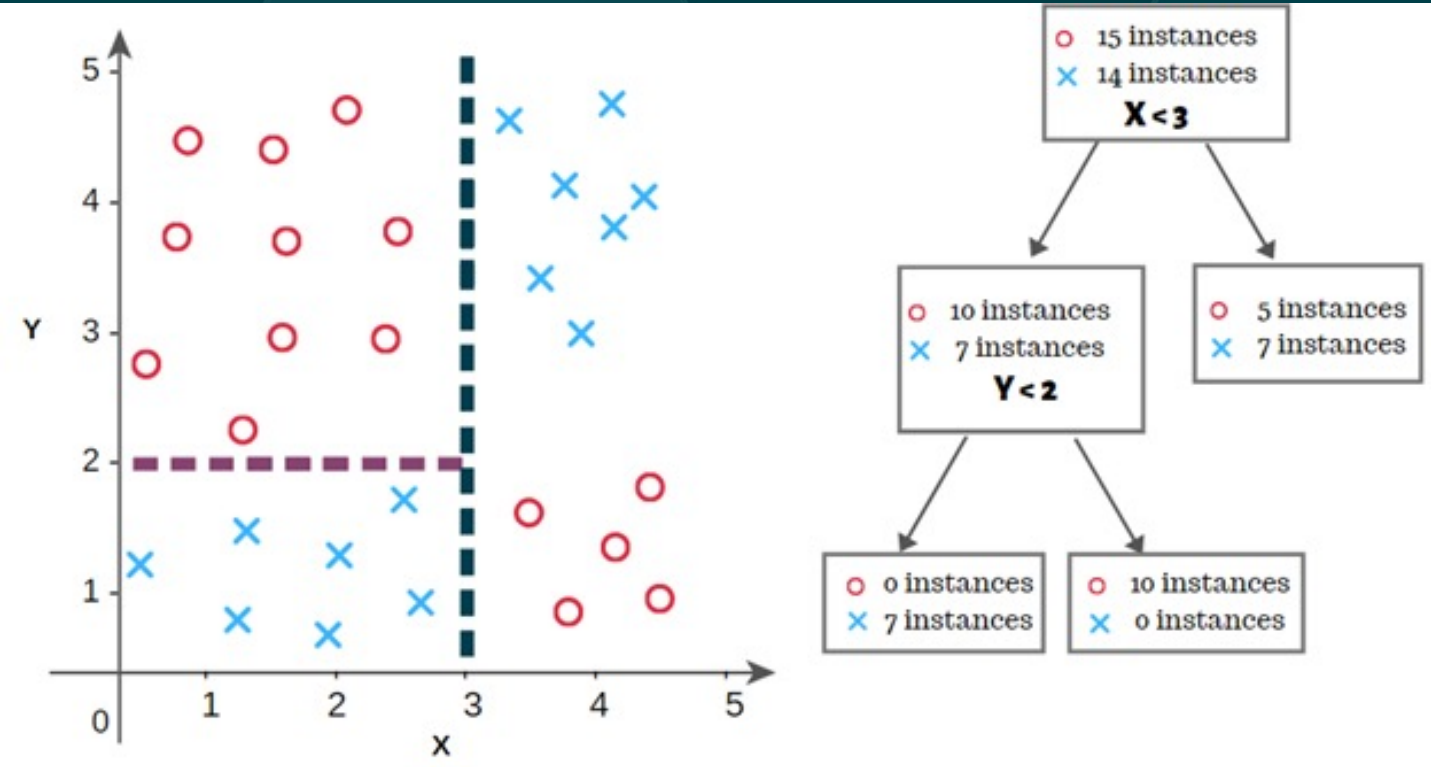
Classification Trees: Leaf nodes are associated with class label or class probability

Regression Trees: Leaf nodes are associated with numeric values

Data source : https://www.kaggle.com/c/titanic/data

Decision Trees

Decision Trees break the input space into Regions (Leaves) where all the points in one region make up the predicted value for that region (in regression trees, we average the values in a region, in classification trees, we take the majority vote).



Data source : https://www.kaggle.com/c/titanic/data

Decision Trees Algorithm

Recursive partitioning algorithm

- Greedily select best split variable (lets say x_i), and best split criterion (lets say s_i)
- Push training examples into left and right nodes ($x_i < s_i$ → left and $x_i > s_i$ → right)
- Further divide each of these two nodes (left and right) in similar manner.

Selecting Split variable and split criterion

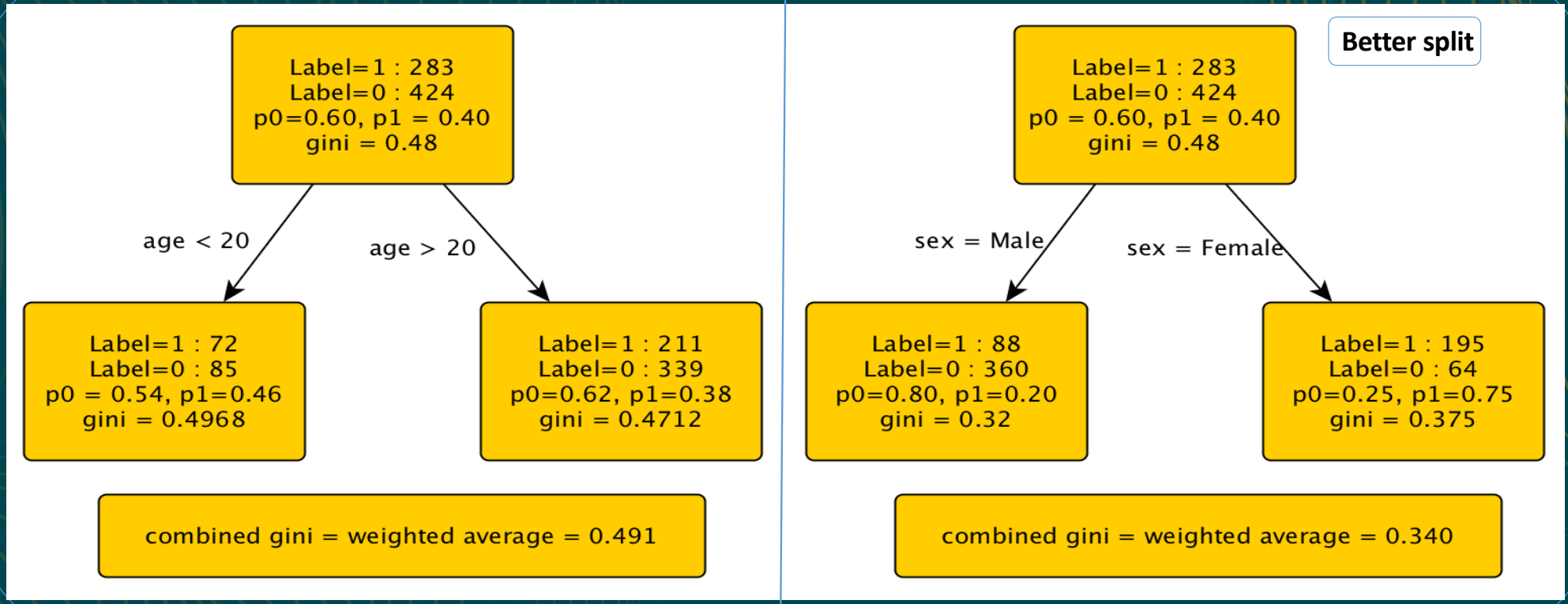
Consider all possible splits and chose the best split (variable and criterion).
The **best split** is the one which leads to the **greatest decrease in node impurity**.

E.g. Possible splits at step 1:

- **Gender:**
  - {male} and {female}
- **Pclass (Ticket class)**:
  - {1} and {2,3}
  - {2} and {1,3}
  - {3} and {1,2}
- **Age**:
  - {age<=1} and {age>1}
  - {age<=2} and {age>2}
  - {age<=3} and {age>3}
  - …

ID	Pclass	Sex	Age	Fare	Survived
0	3	male	22	7.3	0
1	1	female	38	71.3	1
2	3	female	26	7.9	1
3	1	female	35	53.1	1
4	3	male	35	8.1	0
5	1	male	54	51.9	0
6	3	male	2	21.1	0
7	3	female	27	11.1	1
8	2	female	14	30.1	1
9	3	female	4	16.7	1

Splitting example



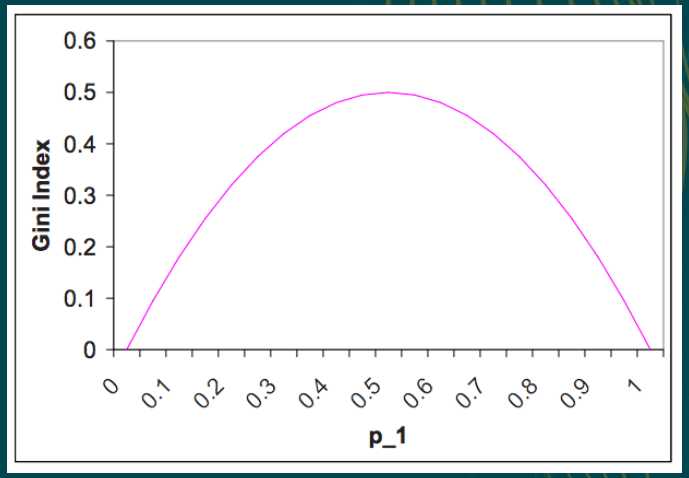
Node Impurity

Classification Trees

- **Gini Impurity Index** $Gini\ (P) = 1 - \sum_{k=1}^{m} p_k^2$

  where p_k is the proportion of observations that belong to class k in population P

  - Gini index takes value between 0 and (m-1)/m
  - For binary class, it is maximized at p_k=0.5
  - Gini is minimum for a heterogenous node (all samples for same class)



- **Entropy measure** $Entropy(P) = -\sum_{k=1}^{m} p_k * log_2(p_k)$

  Like gini index, entropy is also maximized at p_k=0.5 for binary class.

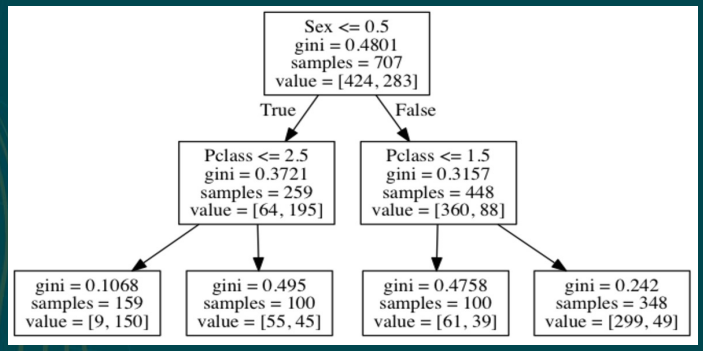
Regression Trees

- **Square Deviation** $Square\ Deviation(P) = \sum_{k=1}^{m} (X_k - \overline{X})^2$ where $\overline{X}$ is mean for the population P

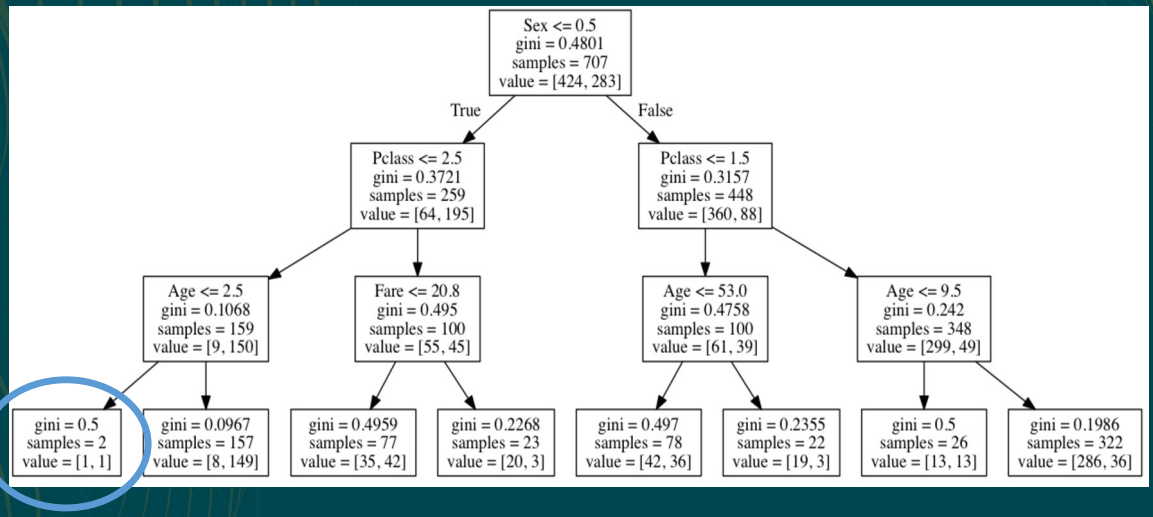
Decision Tree is trying to split the data into heterogenous nodes (or nodes with minimum impurity).

Decision Trees example

Max depth = 2



Max depth = 3

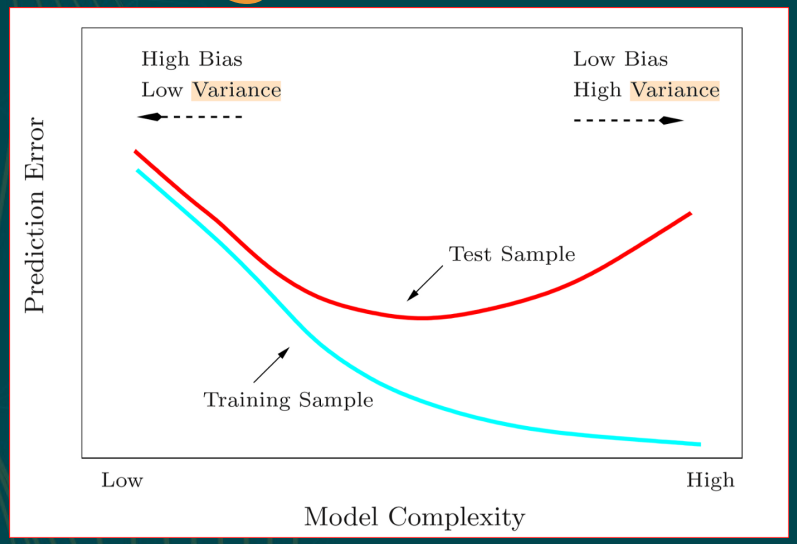


Only two samples in leaf node. Overfitting the training data

Decision Tree Pruning can be used to remove subtree that are non-critical (e.g. subtree that have the least information gain), thus reducing overfitting.

Amazon ML Summer School

Ensemble Learning



- **Ensemble techniques:**
  - **Bagging:**
    - Ensemble technique where bootstrap samples (random samples with replacement) of training data are created.
    - Multiple models (having low bias and high variance) are trained on each bootstrap sample and then the prediction from these models can be combined (e.g by voting or averaging).
    - The final model has low variance due to aggregation of multiple model predictions.
  - **Boosting:**
    - An ensemble technique where multiple weak learners are trained in iterative fashion.
    - Each learner tries to reduce error of previous models.

Bagging (Bootstrap-aggregation)

Bootstrap: a general statistical tool, which can be used to quantify the uncertainty of a learning algorithm.

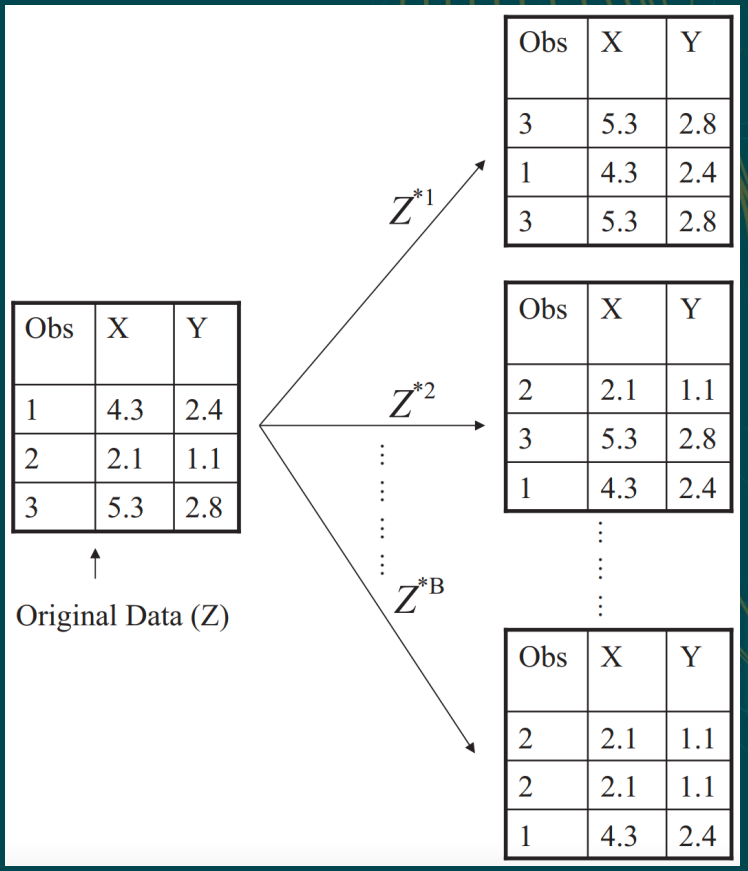
- In the Bootstrap approach, we sample (with replacement) *m* datapoints from our original dataset Z (which also has *m* datapoints). This gives us a new dataset Z*.
- We repeat this *B* times, giving us Z^{*1}, Z^{*2}, … , Z^{*B}

Bagging idea: we use our *B* bootstrapped samples to train *B* models, f^{*1}, … , f^{*B} . Our final "bagged" model predicts the average of these:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

(for classification, we use the majority vote instead)

The final bagged model has reduced variance, which makes bagging a good fit for decision trees (low bias, high variance).



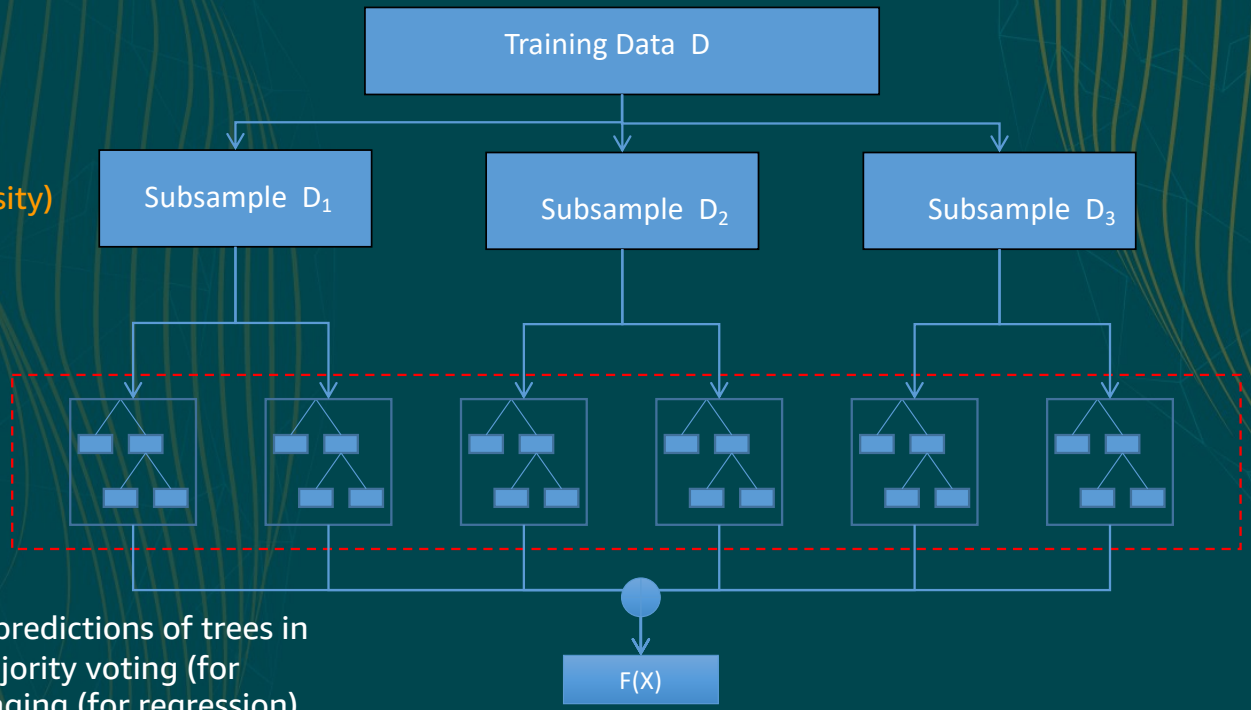
Random Forest

Bagging of *Random* Decision Trees

Bootstrap samples (induces diversity)

Construct *Random* decision trees considering a random subset of attributes for each node split (reduces correlation)

Prediction: Combine predictions of trees in ensemble through majority voting (for classification) or averaging (for regression) (reduces variance)



Boosting & AdaBoost

Boosting idea: train *n* "weak learners" sequentially, combine to reduce **model bias**.

AdaBoost

input:

training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$

weak learner WL

number of rounds T

initialize $\mathbf{D}^{(1)} = (\frac{1}{m}, \ldots, \frac{1}{m})$.

for $t = 1, \ldots, T$:

invoke weak learner $h_t = \mathrm{WL}(\mathbf{D}^{(t)}, S)$

compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$

let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$

update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))}$ for all $i = 1, \ldots, m$

output the hypothesis $h_s(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\right)$.

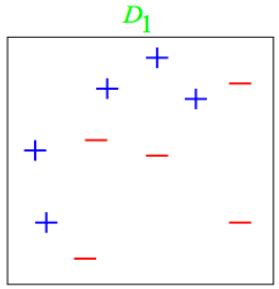
AdaBoost properties:

- In each round, we a new version of training set S, called S', is created by resampling the data points according to the probability vector $D^{(t)}$. We then train a weak learner h_t on S'.
- ε_t is the sum of weights for the examples where h_t made mistakes. We use ε_t to calculate w_t, the final weight for the weak learner h_t.
- At the end, we get a combined model h_s which predicts the average of weak learners h_1, ... h_T, weighted by w_1, ... w_T
- The training error of the combined model decreases **exponentially** with the number of rounds *T*.

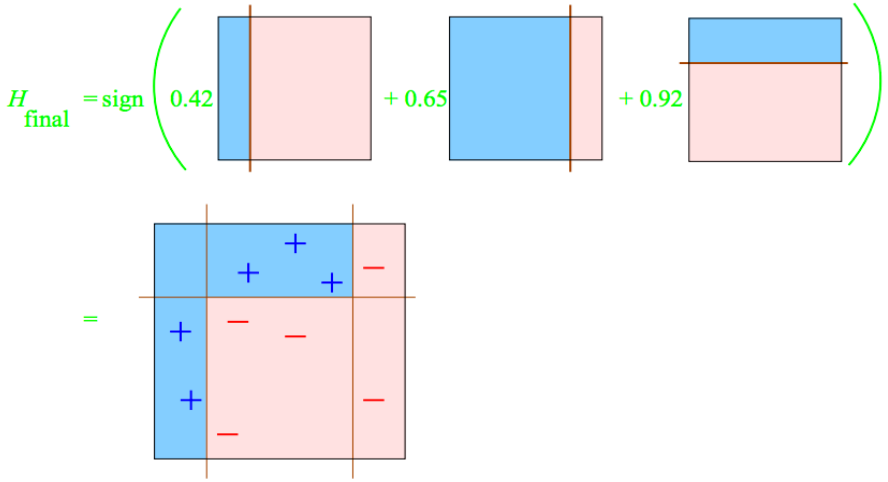
Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

AdaBoost

Toy Example



Final Classifier



weak classifiers = vertical or horizontal half-planes

Slide taken from http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Gradient Boosting

- **Lets play a game ...**
- You are given dataset S = { (x_1,y_1), (x_2,y_2), ... ,(x_m, y_m) } and the task is to fit a model f(x) to minimize **square loss**.
- Suppose your friend wants to help you and gives you a model **f**.
- You check his model and find that model is good but not perfect. How can you improve his model?
- Rules of the game:
  - You are not allowed to change anything in f.
  - You *can* add an additional model "g" to f, so the new prediction will be f(**x**) + g(**x**).

Li, Cheng. "A Gentle Introduction to Gradient Boosting." URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf (2014).

Gradient Boosting

We want g(x) such that :

$$g(x_1) + f(x_1) = y_1$$
$$g(x_2) + f(X_2) = y_2$$
$$\dots$$
$$g(x_n) + f(x_n) = y_n$$

Or Equivalently

$$g(x_1) = y_1 - f(x_1)$$
$$g(x_2) = y_2 - f(x_2)$$
$$\dots$$
$$g(x_n) = y_n - f(x_n)$$

Simple solution: Just train a regression tree on the dataset

$$S' = \{ (x_1, y_1 - f(x_1)), \quad (x_2, y_2 - f(x_2)), \quad \dots, \quad (x_n, y_n - f(x_n)) \}$$

We have got a better model and, we can keep doing it!

Li, Cheng. "A Gentle Introduction to Gradient Boosting." URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf (2014).

Gradient Boosting (GBDT or GBM)

How is this related to gradient descent?

Loss function $L(y, F(x)) = (y - F(x))^2/2$
We want to minimize $J = \sum_i L(y_i, F(x_i))$ by adjusting $F(x_1), F(x_2), ..., F(x_n)$.

Notice that $F(x_1), F(x_2), ..., F(x_n)$ are just some numbers. We can treat $F(x_i)$ as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

Li, Cheng. "A Gentle Introduction to Gradient Boosting." URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf (2014).

Gradient Boosting (GBDT or GBM)

How is this related to gradient descent?

$$F(x_i) := F(x_i) + h(x_i)$$
$$F(x_i) := F(x_i) + y_i - F(x_i)$$
$$F(x_i) := F(x_i) - 1\frac{\partial J}{\partial F(x_i)}$$
$$\theta_i := \theta_i - \rho\frac{\partial J}{\partial \theta_i}$$

- **So, we are actually updating our model using gradient descent !**
- It turns out that concept of gradient is more general. Can be applied to any differentiable loss function.

Li, Cheng. "A Gentle Introduction to Gradient Boosting." URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf (2014).

Gradient Boosting (GBDT or GBM)

Algorithm *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

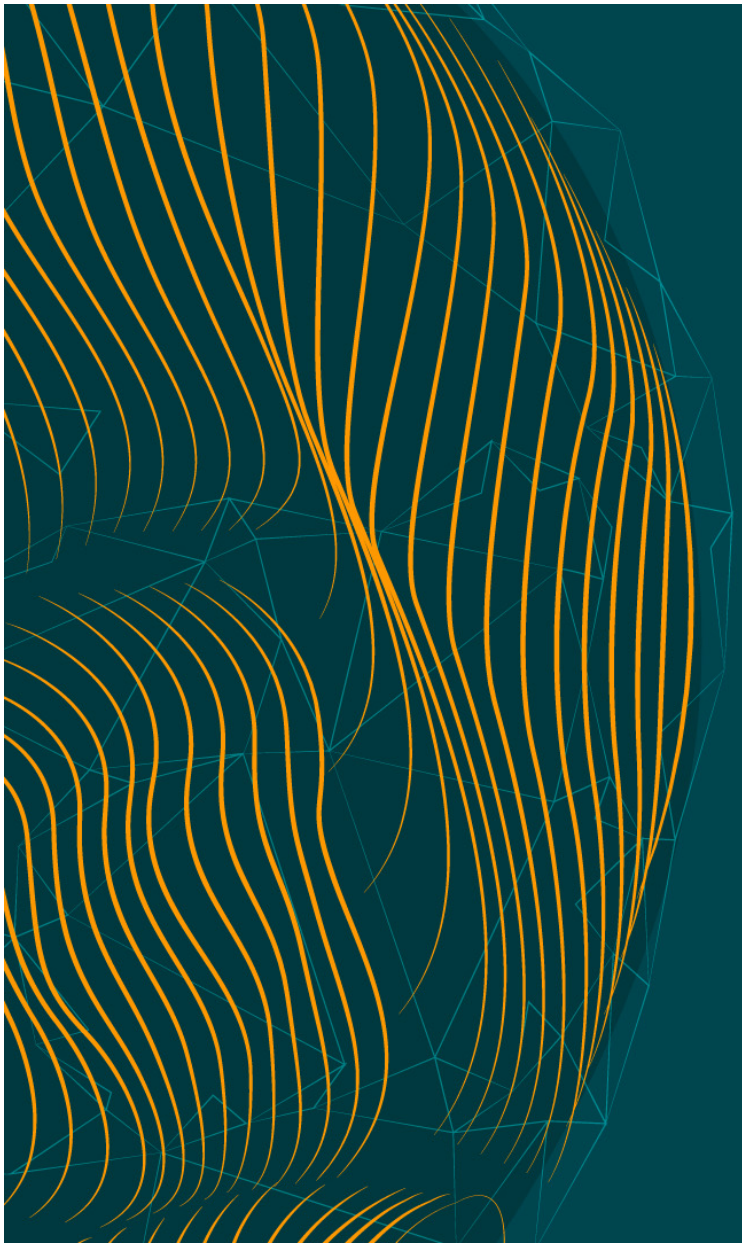
Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

Gradient Boosting (GBDT or GBM)

Gradient Boosting = Boosting + Gradient Descent

- Fit an additive model (ensemble) in a number of rounds.
- In each round, introduce a weak learner to compensate the shortcomings of existing weak learners.
- **In Adaboost, shortcomings are identified by high-weight data points.**
- **In Gradient Boosting, shortcomings are identified by gradients.**
- Both high-weight data points and gradients tell us how to improve our model.

Li, Cheng. "A Gentle Introduction to Gradient Boosting." URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf (2014).



Thank You.