Introduction to Machine Learning

Lecture 13 Classification - Information Theory, Decision Tree, and Practical Challenges

Hongteng Xu

中国人民大學 RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院 Gaoling School of Artificial Intelligence

Outline

Review

- Support-vector machine (SVM)
- Kernelized SVM

Outline

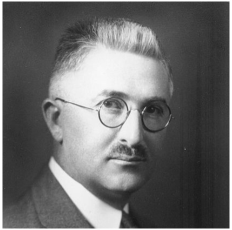
Review

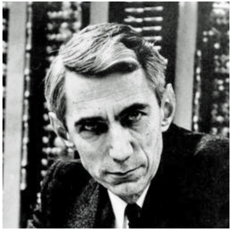
- Support-vector machine (SVM)
- Kernelized SVM

Today

- Information theory in Statistical ML
- Decision tree model and its learning

Information Theory: Origin





Ralph Hartley and Harry Nyquist 1920's   Claude Shannon 1940's

Claude E. Shannon, "A Mathematical Theory of Communication" Bell System Technical Journal in July and October 1948.

The Most Important Concept in Information Theory: Entropy

Motivation:

- We need a measurement for the uncertainty of information

The Most Important Concept in Information Theory: Entropy

Motivation:

- We need a measurement for the uncertainty of information

Principle:

- If X is a symbol (in Information theory, a r.v. in Statistics), and $\mathcal{X} = \{x_1, ..., x_N\}$ contains all possible value it could be.

The Most Important Concept in Information Theory: Entropy

Motivation:

- We need a measurement for the uncertainty of information

Principle:

- If X is a symbol (in Information theory, a r.v. in Statistics), and $\mathcal{X} = \{x_1, ..., x_N\}$ contains all possible value it could be.
- Based on the probability (density) function of X, in units of bits (per symbol), we define

$$H(X) = -\sum_{x \in \mathcal{X}} p(X = x) \log p(X = x)$$

The Most Important Concept in Information Theory: Entropy

Motivation:

- We need a measurement for the uncertainty of information

Principle:

- If X is a symbol (in Information theory, a r.v. in Statistics), and $\mathcal{X} = \{x_1, ..., x_N\}$ contains all possible value it could be.
- Based on the probability (density) function of X, in units of bits (per symbol), we define

$$H(X) = -\sum\nolimits_{x\in\mathcal{X}} p(X=x)\log p(X=x) = \mathbb{E}_{x\sim p_X}[-\log p(x)]. \tag{1}$$

The Most Important Concept in Information Theory: Entropy

Entropy provides a good measure for the uncertainty of symbol.

The Most Important Concept in Information Theory: Entropy

Entropy provides a good measure for the uncertainty of symbol.

- Suppose that $X \sim \text{Bernoulli}(p)$

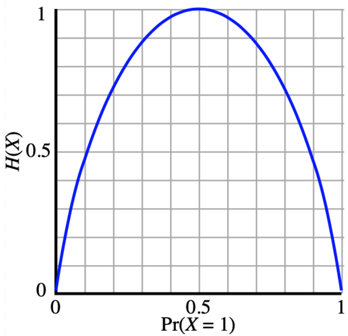
$$H(X) = -p \log p - (1-p)\log(1-p)$$

The Most Important Concept in Information Theory: Entropy

Entropy provides a good measure for the uncertainty of symbol.

- Suppose that $X \sim \text{Bernoulli}(p)$

$$H(X) = -p \log p - (1-p) \log(1-p) \tag{2}$$



Extension: Joint Entropy

- The entropy of paired symbols (random variables):

$$H(X,Y) = \mathbb{E}_{(x,y)\ p_{X,Y}}[-\log p(x,y)] = -\sum\nolimits_{x,y \in \mathcal{X} \times \mathcal{Y}} p(x,y) \log p(x,y)$$

Extension: Joint Entropy

- The entropy of paired symbols (random variables):

$$H(X,Y) = \mathbb{E}_{(x,y)\ p_{X,Y}}[-\log p(x,y)] = -\sum_{x,y\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log p(x,y) \tag{3}$$

- What if X and Y are independent?

Extension: Joint Entropy

- The entropy of paired symbols (random variables):

$$H(X,Y) = \mathbb{E}_{(x,y)\ p_{X,Y}}[-\log p(x,y)] = -\sum\nolimits_{x,y \in \mathcal{X} \times \mathcal{Y}} p(x,y) \log p(x,y) \tag{3}$$

- What if X and Y are independent?

$$p(X,Y) = p(X)p(Y)$$

Extension: Joint Entropy

- The entropy of paired symbols (random variables):

$$H(X, Y) = \mathbb{E}_{(x,y)\ p_{X,Y}}[-\log p(x, y)] = -\sum\nolimits_{x,y \in \mathcal{X} \times \mathcal{Y}} p(x, y) \log p(x, y) \tag{3}$$

- What if X and Y are independent?

$$p(X, Y) = p(X)p(Y) \quad \Leftrightarrow \quad H(X, Y) = H(X) + H(Y) \tag{4}$$

(Derive it)

Extension: Joint Entropy

- The entropy of paired symbols (random variables):

$$H(X,Y) = \mathbb{E}_{(x,y)\ p_{X,Y}}[-\log p(x,y)] = -\sum\nolimits_{x,y \in \mathcal{X}\times\mathcal{Y}} p(x,y)\log p(x,y) \tag{3}$$

- What if X and Y are independent?

$$p(X,Y) = p(X)p(Y) \quad \Leftrightarrow \quad H(X,Y) = H(X) + H(Y) \tag{4}$$

  (Derive it)

- We more care about the case that X and Y are dependent, and how much can we know X given Y?

Extension: Conditional Entropy

- Conditional entropy measures the uncertainty of X given Y:

$$H(X|Y) = \mathbb{E}_{y \sim p_Y}[H(X|\boldsymbol{y})]$$

Extension: Conditional Entropy

- Conditional entropy measures the uncertainty of X given Y:

$$H(X|Y) = \mathbb{E}_{y \sim p_Y}[H(X|y)] = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y)$$

Extension: Conditional Entropy

- Conditional entropy measures the uncertainty of X given Y:

$$
\begin{aligned}
H(X|Y) = \mathbb{E}_{y \sim p_Y}[H(X|y)] &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) \\
&= -\sum_{x,y} p(x,y) \log p(x|y)
\end{aligned}
$$

Extension: Conditional Entropy

- Conditional entropy measures the uncertainty of X given Y:

$$\begin{aligned} H(X|Y) = \mathbb{E}_{y\sim p_Y}[H(X|y)] &= -\sum\nolimits_{y\in\mathcal{Y}} p(y) \sum\nolimits_{x\in\mathcal{X}} p(x|y)\log p(x|y) \\ &= -\sum\nolimits_{x,y} p(x,y)\log p(x|y) \end{aligned} \tag{5}$$

- Therefore, in general we have

$$H(X|Y) = H(X,Y) - H(Y) \tag{6}$$

(Derive it)

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.
- The mutual information of X relative to Y:

$$I(X;Y) = \mathbb{E}_{x,y \sim P_{X,Y}}\left[\log \frac{p(x,y)}{p(x)p(y)}\right]$$

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.
- The mutual information of X relative to Y:

$$I(X;Y) = \mathbb{E}_{x,y\sim P_{X,Y}}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \tag{7}$$

- Recall the NMI we learned in Lecture 8.

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.
- The mutual information of X relative to Y:

$$I(X;Y) = \mathbb{E}_{x,y\sim P_{X,Y}}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \tag{7}$$

- Recall the NMI we learned in Lecture 8.
- Maximizing the amount of information shared between X and Y = Maximizing their mutual information.

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.
- The mutual information of X relative to Y:

$$I(X;Y) = \mathbb{E}_{x,y \sim P_{X,Y}} \left[\log \frac{p(x,y)}{p(x)p(y)} \right] \tag{7}$$

- Recall the NMI we learned in Lecture 8.
- Maximizing the amount of information shared between X and Y = Maximizing their mutual information.
- Obviously, we have

$$I(X;Y) = H(X) - H(X|Y)$$

Extension: Mutual Information

- Mutual Information measures the amount of information that can be obtained about one r.v. by observing another.
- The mutual information of X relative to Y:

$$I(X;Y) = \mathbb{E}_{x,y \sim P_{X,Y}}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] \tag{7}$$

- Recall the NMI we learned in Lecture 8.
- Maximizing the amount of information shared between X and Y = Maximizing their mutual information.
- Obviously, we have

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ I(X;Y) &= I(Y;X) = H(X) + H(Y) - H(X,Y) \end{aligned} \tag{8}$$

Extension: Information Gain / Relative Entropy / KL-Divergence

- A way to compare two r.v.'s distributions
- Given a true distribution $p(X)$ and its estimation $q(X)$:

$$\mathrm{KL}(p_X \| q_X) = \mathbb{E}_{x \sim p_X}[\log \frac{p(x)}{q(x)}] = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

Extension: Information Gain / Relative Entropy / KL-Divergence

- A way to compare two r.v.'s distributions
- Given a true distribution $p(X)$ and its estimation $q(X)$:

$$\mathrm{KL}(p_X \| q_X) = \mathbb{E}_{x \sim p_X}[\log \frac{p(x)}{q(x)}] = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{9}$$

- Minimizing KL-divergence works for fitting model

Extension: Information Gain / Relative Entropy / KL-Divergence

- A way to compare two r.v.'s distributions
- Given a true distribution $p(X)$ and its estimation $q(X)$:

$$\mathrm{KL}(p_X \| q_X) = \mathbb{E}_{x \sim p_X}[\log \frac{p(x)}{q(x)}] = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{9}$$

- Minimizing KL-divergence works for fitting model
- Revisit mutual information:

$$I(X;Y) = \mathrm{KL}(p(X,Y) \| p(X)p(Y))$$

Extension: Information Gain / Relative Entropy / KL-Divergence

- A way to compare two r.v.'s distributions
- Given a true distribution $p(X)$ and its estimation $q(X)$:

$$\mathrm{KL}(p_X \| q_X) = \mathbb{E}_{x \sim p_X}[\log \frac{p(x)}{q(x)}] = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{9}$$

- Minimizing KL-divergence works for fitting model
- Revisit mutual information:

$$I(X;Y) = \mathrm{KL}(p(X,Y) \| p(X)p(Y)) \tag{10}$$

- In other words, maximize mutual information = maximize the KL-divergence between $p(X,Y)$ and $p(X)p(Y)$ = ensure that X and Y are heavily dependent.

Is KL-Divergence Always Nonnegative?

- For convex function ϕ, we have **Jensen's inequality**:

$$\mathbb{E}_Y[\phi(Y)] \geq \phi(\mathbb{E}[Y]) \tag{11}$$

(Actually, just the basic property of convex function)

Is KL-Divergence Always Nonnegative?

- For convex function ϕ, we have **Jensen's inequality**:

$$\mathbb{E}_Y[\phi(Y)] \geq \phi(\mathbb{E}[Y]) \tag{11}$$

(Actually, just the basic property of convex function)

- Therefore

$$-\mathrm{KL}(p \| q) = \int_{x \in \mathcal{X}} p(x) \log \frac{q(x)}{p(x)} \mathrm{d}x$$

Is KL-Divergence Always Nonnegative?

- For convex function ϕ, we have **Jensen's inequality**:

$$\mathbb{E}_Y[\phi(Y)] \geq \phi(\mathbb{E}[Y]) \tag{11}$$

  (Actually, just the basic property of convex function)

- Therefore

$$-\mathrm{KL}(p\|q) = \int_{x\in\mathcal{X}} p(x) \log \frac{q(x)}{p(x)} \mathrm{d}x \leq \log \left(\int_{x\in\mathcal{X}} p(x) \frac{q(x)}{p(x)} \mathrm{d}x \right)$$

Is KL-Divergence Always Nonnegative?

- For convex function ϕ, we have **Jensen's inequality**:

$$\mathbb{E}_Y[\phi(Y)] \geq \phi(\mathbb{E}[Y]) \tag{11}$$

  (Actually, just the basic property of convex function)

- Therefore

$$\begin{aligned}
-\mathrm{KL}(p\|q) = \int_{x\in\mathcal{X}} p(x) \log \frac{q(x)}{p(x)} \mathrm{d}x &\leq \log\left(\int_{x\in\mathcal{X}} p(x) \frac{q(x)}{p(x)} \mathrm{d}x\right) \\
&= \log\left(\int_{x\in\mathcal{X}} q(x) \mathrm{d}x\right) = 0
\end{aligned}$$

Is KL-Divergence Always Nonnegative?

- For convex function ϕ, we have **Jensen's inequality**:

$$\mathbb{E}_Y[\phi(Y)] \geq \phi(\mathbb{E}[Y]) \tag{11}$$

(Actually, just the basic property of convex function)

- Therefore

$$\begin{aligned} -\mathrm{KL}(p\|q) = \int_{x\in\mathcal{X}} p(x)\log\frac{q(x)}{p(x)}\mathrm{d}x &\leq \log\left(\int_{x\in\mathcal{X}} p(x)\frac{q(x)}{p(x)}\mathrm{d}x\right) \\ &= \log\left(\int_{x\in\mathcal{X}} q(x)\mathrm{d}x\right) = 0 \end{aligned} \tag{12}$$

Revisit Some Methods/Models through Information Theory

- Recall the cross entropy loss used in classification:

$$H(p,q) = -\mathbb{E}_{x\sim p}[\log q(x)] = H(p) + \mathrm{KL}(p\|q)$$

Revisit Some Methods/Models through Information Theory

- Recall the cross entropy loss used in classification:

$$H(p, q) = -\mathbb{E}_{x \sim p}[\log q(x)] = H(p) + \mathrm{KL}(p \| q) \tag{13}$$

- Recall the objective function of t-SNE.

Revisit Some Methods/Models through Information Theory

- Recall the cross entropy loss used in classification:

$$H(p, q) = -\mathbb{E}_{x \sim p}[\log q(x)] = H(p) + \mathrm{KL}(p \| q) \tag{13}$$

- Recall the objective function of t-SNE.
- Recall the EM algorithm of GMM in Lecture 9. Can you rewrite the surrogate objective function $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ based on KL-divergence?

Revisit Some Methods/Models through Information Theory

- Recall the cross entropy loss used in classification:

$$H(p, q) = -\mathbb{E}_{x\sim p}[\log q(x)] = H(p) + \mathrm{KL}(p\|q) \tag{13}$$

- Recall the objective function of t-SNE.
- Recall the EM algorithm of GMM in Lecture 9. Can you rewrite the surrogate objective function $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ based on KL-divergence?
- The decision criterion used in Decision Tree Models

Decision Tree Model: An Example

Motivation:

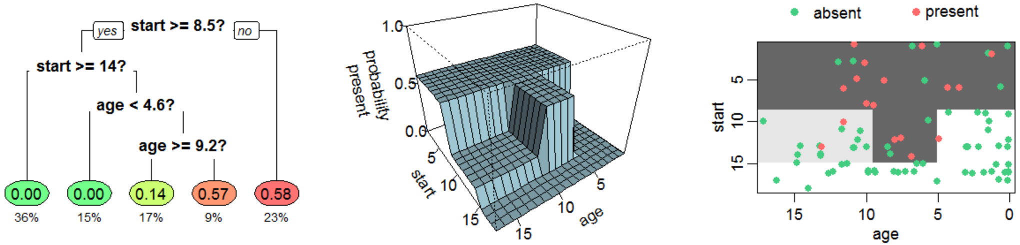
- Design/Learn simple and interpretable rules for different features may be sufficient to achieve complicated nonlinear classification.

Decision Tree Model: An Example

Motivation:

- Design/Learn simple and interpretable rules for different features may be sufficient to achieve complicated nonlinear classification.

Example: The probability of kyphosis after spinal surgery, given the age of the patient and the vertebra at which surgery was started.



Decision Tree Learning

Many learning algorithms are proposed

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- ...

Decision Tree Learning

Many learning algorithms are proposed

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- ...

Principle:

- Tree-generation algorithms: Generate a tree by the splitting driven by information gain maximization

Decision Tree Learning

- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \ldots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.

Decision Tree Learning

- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \ldots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.
- In the beginning, all the data points are in the root of the tree.

Decision Tree Learning

- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \ldots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.
- In the beginning, all the data points are in the root of the tree.
- The number of features, i.e., D, indicates how many trees we need to compare in each split.

Decision Tree Learning

- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \dots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.
- In the beginning, all the data points are in the root of the tree.
- The number of features, i.e., D, indicates how many trees we need to compare in each split.
- To construct a decision tree on this data, we need to compare the information gain of D trees, each split on one of the D features.

Decision Tree Learning

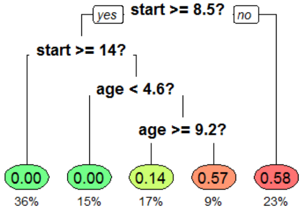
- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \ldots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.
- In the beginning, all the data points are in the root of the tree.
- The number of features, i.e., D, indicates how many trees we need to compare in each split.
- To construct a decision tree on this data, we need to compare the information gain of D trees, each split on one of the D features.
- The split with the highest information gain will be taken as the first split

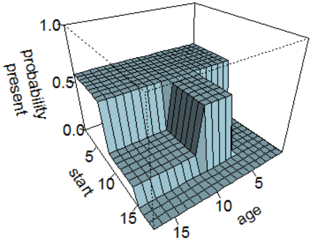
Decision Tree Learning

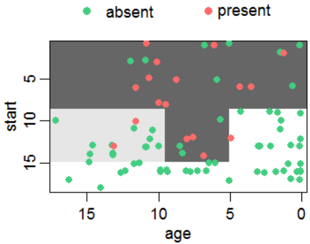
- Given classified training data, $\mathcal{S} = \cup_{i=1}^{C} \mathcal{S}_i$ and $\boldsymbol{x}_n = [x_{1,n}, \ldots, x_{D,n}] \in \mathcal{S}_i$ is the n-th D-dimensional sample of the i-th class.
- In the beginning, all the data points are in the root of the tree.
- The number of features, i.e., D, indicates how many trees we need to compare in each split.
- To construct a decision tree on this data, we need to compare the information gain of D trees, each split on one of the D features.
- The split with the highest information gain will be taken as the first split
- The above process will continue until all children nodes have consistent data or until the information gain is 0.

Recall The Example

Example: The probability of kyphosis after spinal surgery, given the age of the patient and the vertebra at which surgery was started.







Decision Tree Learning: Determine The Criteria for Splitting

- Given current tree $T = \{p_1, ..., p_J\}$, where $p_1, ..., p_J$ represent the percentage of each class present in the child node that results from a split in the tree.

Decision Tree Learning: Determine The Criteria for Splitting

- Given current tree $T = \{p_1, ..., p_J\}$, where $p_1, ..., p_J$ represent the percentage of each class present in the child node that results from a split in the tree.
- To update the tree, we need to split its node to maximize the information gain

$$\underbrace{I(T;A)}_{\text{Mutual Information}} = \underbrace{H(T)}_{\text{entropy of parent}} - \underbrace{H(T|A)}_{\text{weighted sum of children's entropies}}$$

Decision Tree Learning: Determine The Criteria for Splitting

- Given current tree $T = \{p_1, \ldots, p_J\}$, where $p_1, \ldots, p_J$ represent the percentage of each class present in the child node that results from a split in the tree.
- To update the tree, we need to split its node to maximize the information gain

$$
\begin{aligned}
\underbrace{I(T;A)}_{\text{Mutual Information}} &= \underbrace{H(T)}_{\text{entropy of parent}} - \underbrace{H(T|A)}_{\text{weighted sum of children's entropies}} \\
&= -\sum_{i=1}^{J} p_i \log p_i - \sum_{a \in A} p(a) \underbrace{\sum_{i=1}^{J} -p(i|a) \log p(i|a)}_{H(T|a)} .
\end{aligned}
$$

Decision Tree Learning: Determine The Criteria for Splitting

- Given current tree $T = \{p_1, \dots, p_J\}$, where $p_1, \dots, p_J$ represent the percentage of each class present in the child node that results from a split in the tree.
- To update the tree, we need to split its node to maximize the information gain

$$\begin{aligned}\underbrace{I(T;A)}_{\text{Mutual Information}} &= \underbrace{H(T)}_{\text{entropy of parent}} - \underbrace{H(T|A)}_{\text{weighted sum of children's entropies}} \\ &= -\sum_{i=1}^{J} p_i \log p_i - \sum_{a \in A} p(a) \underbrace{\sum_{i=1}^{J} -p(i|a) \log p(i|a)}_{H(T|a)} .\end{aligned} \tag{14}$$

- $\max I(T;A)$ helps us to find the best feature used to split current tree.
- $\max H(T|a)$ helps us to find the best splitting criterion.

Decision Tree Learning: Determine The Criteria for Splitting

- Given current tree $T = \{p_1, ..., p_J\}$, where $p_1, ..., p_J$ represent the percentage of each class present in the child node that results from a split in the tree.
- To update the tree, we need to split its node to maximize the information gain

$$\begin{aligned} \underbrace{I(T;A)}_{\text{Mutual Information}} &= \underbrace{H(T)}_{\text{entropy of parent}} - \underbrace{H(T|A)}_{\text{weighted sum of children's entropies}} \\ &= -\sum_{i=1}^{J} p_i \log p_i - \sum_{a \in A} p(a) \underbrace{\sum_{i=1}^{J} -p(i|a) \log p(i|a)}_{H(T|a)} . \end{aligned} \tag{14}$$

- $\max I(T;A)$ helps us to find the best feature used to split current tree.
- $\max H(T|a)$ helps us to find the best splitting criterion.

Refer to https://en.wikipedia.org/wiki/Decision_tree_learning

Pros and Cons of Decision Tree

Pros:

- Simple and interpretable

Cons:

- Non-robust to noise and randomness

Pros and Cons of Decision Tree

Pros:

- Simple and interpretable

Cons:

- Non-robust to noise and randomness
- But can be suppressed by ensemble modeling (Next lecture)

Challenges of Trustworthy ML

Non-i.i.d. Observations

- The samples and their labels are streaming, which dependent on historical data.

Challenges of Trustworthy ML

Non-i.i.d. Observations

- The samples and their labels are streaming, which dependent on historical data.

Unbalanced Observations

- Majority v.s. Minority

Challenges of Trustworthy ML

Non-i.i.d. Observations

- The samples and their labels are streaming, which dependent on historical data.

Unbalanced Observations

- Majority v.s. Minority

Uncertain Outputs

- Make mistakes, and confidently :(

Challenges of Trustworthy ML

Non-i.i.d. Observations

- The samples and their labels are streaming, which dependent on historical data.

Unbalanced Observations

- Majority v.s. Minority

Uncertain Outputs

- Make mistakes, and confidently :(

Fairness, Interpretability, Privacy, ...

Data Augmentation for Suppressing Data Unbalance

Reweighting

- Assign more weights/significance on the samples of minority classes.

Data Augmentation for Suppressing Data Unbalance

Reweighting

- Assign more weights/significance on the samples of minority classes.
- Equivalently, when applying SGD, we can sample minority classes with higher probability.

Data Augmentation for Suppressing Data Unbalance

Reweighting

- Assign more weights/significance on the samples of minority classes.
- Equivalently, when applying SGD, we can sample minority classes with higher probability.

Simulation/Sampling

- When some scalable augmentation mechanisms are applicable, we can simulate new samples easily (e.g., image/text augmentation)

Data Augmentation for Suppressing Data Unbalance

Reweighting

- Assign more weights/significance on the samples of minority classes.
- Equivalently, when applying SGD, we can sample minority classes with higher probability.

Simulation/Sampling

- When some scalable augmentation mechanisms are applicable, we can simulate new samples easily (e.g., image/text augmentation)
- If a generative model is applicable, it may work better.

Ensemble Model: Bagging (or Called Bootstrap Aggregating)

Motivation:

- Ensemble all possible models (hypotheses) in the model (hypothesis) space.

Ensemble Model: Bagging (or Called Bootstrap Aggregating)

Motivation:

- Ensemble all possible models (hypotheses) in the model (hypothesis) space.
- Inspired by voting — even if each model is weak, they can make strongly reliable decision.

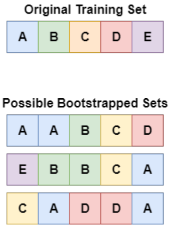
Ensemble Model: Bagging (or Called Bootstrap Aggregating)

Motivation:

- Ensemble all possible models (hypotheses) in the model (hypothesis) space.
- Inspired by voting — even if each model is weak, they can make strongly reliable decision.

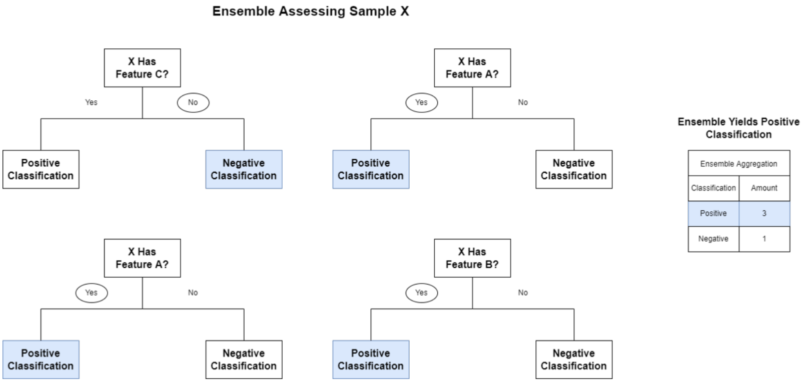
Principle:

- Applying bootstrap to generate bootstrapped datasets.



Ensemble Model: Bagging (or Called Bootstrap Aggregating)

- Training several models based on the bootstrapped datasets.
- In the testing phase, output the voting results of the models.



Ensemble Model: Boosting (e.g., AdaBoost)

Motivation:

- Can we incrementally build an ensemble model?

Ensemble Model: Boosting (e.g., AdaBoost)

Motivation:

- Can we incrementally build an ensemble model?
- Can we make each model as weak as possible (e.g., slightly better than random guessing), such that the training will be cheap?

Ensemble Model: Boosting (e.g., AdaBoost)

Motivation:

- Can we incrementally build an ensemble model?
- Can we make each model as weak as possible (e.g., slightly better than random guessing), such that the training will be cheap?
- More suitable for streaming data (consider the few-shot learning achieved by meta-learning)

Ensemble Model: Boosting (e.g., AdaBoost)

Motivation:

- Can we incrementally build an ensemble model?
- Can we make each model as weak as possible (e.g., slightly better than random guessing), such that the training will be cheap?
- More suitable for streaming data (consider the few-shot learning achieved by meta-learning)

Principle:

- Training each new model to emphasize the samples that previous models mis-classified.

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

Ensemble Model: Boosting (e.g., AdaBoost)

Motivation:

- Can we incrementally build an ensemble model?
- Can we make each model as weak as possible (e.g., slightly better than random guessing), such that the training will be cheap?
- More suitable for streaming data (consider the few-shot learning achieved by meta-learning)

Principle:

- Training each new model to emphasize the samples that previous models mis-classified.

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{15}$$

- f_t is a weak learner/classifier created at step t.
- F_T is the ensemble model obtained after T steps.

Ensemble Model: Boosting (e.g., AdaBoost)

At each step t,

- **Training:** A model (hypothesis h) is selected and assigned a coefficient α_t, and we learn it via minimizing the total training error

$$h^* = \arg\min_h \sum_{(y,x)\in\mathcal{D}} \mathrm{loss}(y, F_{t-1}(x) + \alpha_t h(x))$$

$$f_t(x) = \alpha_t h(x).$$

Ensemble Model: Boosting (e.g., AdaBoost)

At each step t,

- **Training:** A model (hypothesis h) is selected and assigned a coefficient α_t, and we learn it via minimizing the total training error

$$\begin{aligned} h^* &= \arg\min_h \sum\nolimits_{(y,x)\in\mathcal{D}} \mathrm{loss}(y, F_{t-1}(x) + \alpha_t h(x)) \\ f_t(x) &= \alpha_t h(x). \end{aligned} \tag{16}$$

- **Weighting:** A weight $w_{n,t}$ is assigned to the sample (x_n, y_n) at step t.

Ensemble Model: Boosting (e.g., AdaBoost)

At each step t,

- **Training:** A model (hypothesis h) is selected and assigned a coefficient α_t, and we learn it via minimizing the total training error

$$\begin{aligned} h^* &= \arg\min_h \sum\nolimits_{(y,x)\in\mathcal{D}} \text{loss}(y, F_{t-1}(x) + \alpha_t h(x)) \\ f_t(x) &= \alpha_t h(x). \end{aligned} \tag{16}$$

- **Weighting:** A weight $w_{n,t}$ is assigned to the sample (x_n, y_n) at step t.
- The weights are initialized uniformly.

Ensemble Model: Boosting (e.g., AdaBoost)

At each step t,

- **Training:** A model (hypothesis h) is selected and assigned a coefficient α_t, and we learn it via minimizing the total training error

$$
\begin{aligned}
h^* &= \arg\min_h \sum\nolimits_{(y,x)\in\mathcal{D}} \operatorname{loss}(y, F_{t-1}(x) + \alpha_t h(x)) \\
f_t(x) &= \alpha_t h(x).
\end{aligned} \tag{16}
$$

- **Weighting:** A weight $w_{n,t}$ is assigned to the sample (x_n, y_n) at step t.
- The weights are initialized uniformly.
- When using exponential loss and denoting the current error rate of t-th model is ϵ_t, we have

$$
\epsilon_t = \frac{\sum_{y_n \neq f_t(x_n)} w_{n,t}}{\sum_{n=1}^N w_{n,t}}, \quad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)
$$

Ensemble Model: Boosting (e.g., AdaBoost)

At each step t,

- **Training:** A model (hypothesis h) is selected and assigned a coefficient α_t, and we learn it via minimizing the total training error

$$\begin{aligned} h^* &= \arg\min_h \sum\nolimits_{(y,x)\in\mathcal{D}} \text{loss}(y, F_{t-1}(x) + \alpha_t h(x)) \\ f_t(x) &= \alpha_t h(x). \end{aligned} \tag{16}$$

- **Weighting:** A weight $w_{n,t}$ is assigned to the sample (x_n, y_n) at step t.
- The weights are initialized uniformly.
- When using exponential loss and denoting the current error rate of t-th model is ϵ_t, we have

$$\epsilon_t = \frac{\sum_{y_n \neq f_t(x_n)} w_{n,t}}{\sum_{n=1}^N w_{n,t}}, \quad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{17}$$

Note: we need to ensure each $\epsilon_t < 0.5$ (weak, but cannot weaker anymore.)

Bagging v.s. Boosting

- Boosting often shows better performance on bagging.
- Boosting tends to be more likely to over-fit the training data.

Bagging v.s. Boosting

- Boosting often shows better performance on bagging.
- Boosting tends to be more likely to over-fit the training data.
- If each model is a decision tree:
  - Bagging: Random forest
  - Boosting: Adaptive boosting (AdaBoost)

More Challenges:(

More Challenges:(

but relax$\sim$ just some stories:)

More Challenges:(

but relax$\sim$ just some stories:)

- **Privacy:** One story about healthcare

More Challenges:(

but relax$\sim$ just some stories:)

- **Privacy:** One story about healthcare
- **Fairness:** Two stories about gender fairness

More Challenges:(

but relax$\sim$ just some stories:)

- **Privacy:** One story about healthcare
- **Fairness:** Two stories about gender fairness

What we missed in this course (but you will learn it in the future)

More Challenges:(

but relax$\sim$ just some stories:)

- **Privacy:** One story about healthcare
- **Fairness:** Two stories about gender fairness

What we missed in this course (but you will learn it in the future)

- Neural networks, deep generative modeling, semi-supervised learning, contrastive learning, meta-learning, few/zero-shot learning, self-supervised learning, transfer learning, domain adaptation, Bayesian inference, Bayesian optimization, probabilistic graphical model, causal inference, metric learning, large-scale pretraining, distributed learning, federated learning, lifelong learning, reinforcement learning, time series, PAC learning theory, stochastic control theory, ...

In Summary

- Basic concepts of information theory: entropy, mutual information, etc.
- Revisit some loss functions and machine learning models from the viewpoint of information theory
- Decision tree model and its learning
- More practical problems and potential solutions.

In Summary

- Basic concepts of information theory: entropy, mutual information, etc.
- Revisit some loss functions and machine learning models from the viewpoint of information theory
- Decision tree model and its learning
- More practical problems and potential solutions.

Next...

- Review and happy ending of this course:)
- Review all homework