Logistics

• Homework 1 Returned
  • Regrade requests till this Saturday
  • Please be concise and polite

• Homework 3: Due **Mar 5** (Sat)
  • Keep track of your own late-day usages

• Exam 1: **Mar 10 (Thursday)**
  • Topics: LFD Chapters 1 to 5
  • Covid-permitting
    • Timed exam (75 min) during lecture time in the classroom
    • Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
      • No format limitations (it can be typed, written, or a combination)
  • Mar 8 (Tuesday) will be a review lecture
Recap
Decision Tree
### Decision Tree Hypothesis

*Pros*
- Easy to interpret (interpretability is getting attention and is important in some domains)
- Can handle multi-type data (Numerical, categorical, ...)
- Easy to implement (Bunch of if-else rules)

*Cons*
- Generally speaking, bad generalization
- VC dimension is infinity
- High variance (small change of data leads to very different hypothesis)
- Easily overfit

*Why we care?*
- One of the classical model
- Building block for other models (e.g., random forest)

---

Credit Card Approval Example

```
Annual Income

≥ 100k

≥ 20k
< 100k

< 20k

Approve

have debt?

yes

no

Deny

Approve

Deny

Approve
```
Learning Decision Tree from Data

• Given dataset $D$, how to learn a decision tree hypothesis?

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

• Potential approach:
  • Empirical risk minimization
  • Find $g = \arg\min_{h \in H} E_{in}(h)$

• Multiple decision trees with zero $E_{in}$

How to avoid overfitting?
Learning Decision Tree from Data

• Conceptual intuition to deal with overfitting
  • Regularization: Constrain $H$

• Informally,

$$\text{minimize } E_{in}(\overline{w})$$
$$\text{subject to } \text{size(tree)} \leq C$$

• This optimization is generally computationally intractable.
• Most decision tree learning algorithms rely on heuristics to approximate the goal.
Greedy-Based Decision Tree Algorithm

• Greedily, recursively, choose the next feature to split

• DecisionTreeLearn(D): Input a dataset \( D \), output a decision tree hypothesis
  • Create a root node
  • If termination conditions are met
    • return a single node tree with leaf prediction based on \( D \)
  • Else: Greedily find a feature \( A \) to split according to split criteria
    • For each possible value \( v_i \) of \( A \)
      • Let \( D_i \) be the dataset containing data with value \( v_i \) for feature \( A \)
      • Create a subtree DecisionTreeLearn(\( D_i \)) that being the child of root

• Most decision tree learning algorithms follow this template, but with different choices of heuristics
ID3: Using Information Gain as Selection Criteria

• Information gain of choosing feature $A$ to split
  • $Gain(D, A) = H(D) - \sum_i \frac{\lvert D_i \rvert}{\lvert D \rvert} H(D_i)$ [The amount of decrease in entropy]

• ID3: Choose the split that maximize $Gain(D, A)$

DecisionTreeLearn($D$)
Create a root node $r$
If termination conditions are met
  return a single node tree with leaf prediction based on
Else: Greedily find a feature $A$ to split according to split criteria
For each possible value $v_i$ of $A$
  Let $D_i$ be the dataset containing data with value $v_i$ for feature $A$
  Create a subtree DecisionTreeLearn($D_i$) that being the child of root $r$

• ID3 termination conditions
  • If all labels are the same
  • If all features are the same
  • If dataset is empty

• ID3 leaf predictions
  • Most common labels (majority voting)

• ID3 split criteria
  • Information gain

Notations:
$H(D)$: Entropy of $D$
$\lvert D \rvert$ is the number of points in $D$
Illustration of "High Variance" of Decision Trees

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rain</td>
<td>Before</td>
<td>Both</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>Rain</td>
<td>During</td>
<td>Both</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>Rain</td>
<td>During</td>
<td>Both</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>Rain</td>
<td>After</td>
<td>Backpack</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>Rain</td>
<td>After</td>
<td>Backpack</td>
<td>Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>Rain</td>
<td>After</td>
<td>Lunchbox</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>No Rain</td>
<td>Before</td>
<td>Backpack</td>
<td>Tired</td>
<td>Bike</td>
</tr>
<tr>
<td>No Rain</td>
<td>Before</td>
<td>Lunchbox</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>No Rain</td>
<td>Before</td>
<td>Lunchbox</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>No Rain</td>
<td>During</td>
<td>Backpack</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>No Rain</td>
<td>During</td>
<td>Both</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>No Rain</td>
<td>After</td>
<td>Backpack</td>
<td>Not Tired</td>
<td>Bike</td>
</tr>
<tr>
<td>No Rain</td>
<td>After</td>
<td>Backpack</td>
<td>Tired</td>
<td>Bike</td>
</tr>
<tr>
<td>No Rain</td>
<td>After</td>
<td>Both</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
<tr>
<td>No Rain</td>
<td>After</td>
<td>Both</td>
<td>Tired</td>
<td>Drive</td>
</tr>
<tr>
<td>No Rain</td>
<td>After</td>
<td>Lunchbox</td>
<td>Not Tired</td>
<td>Metro</td>
</tr>
</tbody>
</table>
High variance: A small deviation of data would lead to very different learned hypothesis
Decision Tree Hypothesis

Pros
- Easy to interpret (interpretability is getting attention and is important in some domains)
- Can handle multi-type data (Numerical, categorical, ...)
- Easy to implement (Bunch of if-else rules)

Cons
- Generally speaking, bad generalization
- VC dimension is infinity
- High variance (small change of data leads to very different hypothesis)
- Easily overfit

Why we care?
- One of the classical model
- Building block for other models

Credit Card Approval Example

```
<table>
<thead>
<tr>
<th>Annual Income</th>
<th>have debt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 100k</td>
<td>Approve</td>
</tr>
<tr>
<td>≥ 20k</td>
<td>have debt?</td>
</tr>
<tr>
<td>&lt; 20k</td>
<td>Deny</td>
</tr>
<tr>
<td>&lt; 100k</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>Deny</td>
</tr>
<tr>
<td>no</td>
<td>Approve</td>
</tr>
</tbody>
</table>
```

Annual Income ≥ 100k: Approve
Annual Income ≥ 20k: have debt?
Annual Income < 20k: Deny
Annual Income < 100k: no

Have debt?
- yes: Deny
- no: Approve
Today’s Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook. Let me know if you spot errors.
Ensemble Learning
Ensemble Learning

• Assume we are given a set of learned hypothesis
  • $g_1, g_2, \ldots, g_M$

• What can we do?
  • Select the best one: use validation for model selection
  • What if all of them are not good enough?

• Can we aggregate them?
Aggregation

• Given a set of weak learners $g_1, \ldots, g_M$, how to output a stronger learner that performs better?

• Uniform aggregation
  • Regression (average): $\bar{g}(\vec{x}) = \frac{1}{M} \sum_{m=1}^{M} g_m(\vec{x})$
  • Classification (majority vote): $\bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^{M} g_m(\vec{x}) \right)$

• Weighted aggregation
  • Regression (average): $\bar{g}(\vec{x}) = \frac{1}{M} \sum_{m=1}^{M} \alpha_m g_m(\vec{x})$
  • Classification (majority vote): $\bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^{M} \alpha_m g_m(\vec{x}) \right)$

• Stacking (won’t talk about this in this course)
  • Take the prediction of $g_1$ to $g_m$ as input features, train another model on top of that
Is Aggregation a Good Idea?

• Some illustrative examples
Is Aggregation a Good Idea?

- Some illustrative examples
Is Aggregation a Good Idea?

• Maybe
  • If the hypothesis is diverse, and “on average” they seem good
  • (If you take humans as weak learners, this is almost democracy)

• Question:
  • How do we find a set of hypothesis that are diverse and “on average” good
  • How do we aggregate the set of hypothesis

• Ensemble learning
  • Bagging – Random Forest (This lecture)
  • Boosting – AdaBoost (Next lecture)
Diverse Weak Learners

• One common way to construct weak learners is via decision trees

• Fully grown decision trees
  • High variance
  • Low bias

• Decision stump (One-depth decision trees, split on only one attribute)
  • Low variance
  • High bias
Bagging

Bootstrapped Aggregating

(Using randomization to construct diverse weak learners)
Review: Bias-Variance Decomposition

• $f$: sine function, $H: h(x) = ax + b$, $N=2$

• Observations
  • The variance of each learned hypothesis is high
  • The variance of “average” of them ($\bar{g}(\hat{x})$) is lower

• Can we apply similar intuitions?
  • Generate a lot of high-variance but low bias weak learners
  • Aggregate them using uniform aggregation

We only have one dataset in practice!
Bootstrapping

• Intuition:
  • Use the dataset $D$ we have to approximate the data distribution
  • Sample (with replacement) from $D$ to create bootstrapped datasets

• Bootstrapping:
  • Let $D = \{ (\hat{x}_1, y_1), ..., (\hat{x}_N, y_N) \}$ be the dataset we have
  • Repeatedly uniformly sample $N$ points from $D$ with replacement
    • The number of sampled points doesn’t have to be $N$, but it’s a reasonable/common choice.
  • Obtain many bootstrapped datasets
    • $\tilde{D}^{(1)} = \{ (x_1, y_1), (x_1, y_1), (x_4, y_4), ... \}$
    • ...
    • $\tilde{D}^{(M)}$
Bagging - Bootstrapped Aggregating

• Bootstrap $M$ datasets $\{\tilde{D}^{(m)}\}$ and learn a hypothesis from each of them

• Aggregate the learned hypothesis
Bagging - Bootstrapped Aggregating

• Bootstrap $M$ datasets $\{\mathcal{D}^{(m)}\}$ and learn a hypothesis from each of them

$$H \rightarrow \mathcal{D}^{(1)} \rightarrow g_1$$
$$H \rightarrow \mathcal{D}^{(2)} \rightarrow g_2$$
$$\vdots$$
$$H \rightarrow \mathcal{D}^{(M)} \rightarrow g_M$$

• Aggregate the learned hypothesis

$$G(\hat{x}) = \bar{g}(\hat{x}) = \text{sign}\left(\frac{1}{M} \sum_{m=1}^{M} g_m(\hat{x})\right)$$

(assume we are doing classification)
Why/When Bagging Might Be Helpful?

- What we know from statistics
  - Consider $M$ independent random variables $x_1, x_2, \ldots, x_M$, each with variance $\sigma^2$
  - The variance of $\frac{1}{M} \sum_{m=1}^{M} x_m$ is $\frac{\sigma^2}{M}$

- If we have “weak learners” that have high variance but low bias
  - Bagging helps reduce the variance and maintain low bias
  - From bias-variance decomposition, this implies a strong learner
Break and Question

Exercise:
Given a dataset $D$ with $N$ points. Consider we bootstrap a dataset $\tilde{D}^{(m)}$ by sampling $N$ points with replacement from $D$, what’s the probability that a given point $(\tilde{x}_n, \tilde{y}_n)$ is not in $\tilde{D}^{(m)}$?
Out-Of-Bag (OOB) Error
Probability for a Point to be Out of Bag

• Consider we bootstrap a dataset $\tilde{D}^{(m)}$ by sampling $N$ points from $D$, what’s the probability that a given point $(\tilde{x}_n, y_n)$ is not in $\tilde{D}^{(m)}$.

\[
\left(1 - \frac{1}{N}\right)^N = \left(\frac{1}{1+\frac{1}{N-1}}\right)^N \\
\approx \frac{1}{e} \approx 0.36 \text{ when } N \to \infty
\]

When $N$ is large, for each bootstrapped dataset $\tilde{D}^{(m)}$, a significant proportion of points in $D$ is not included.

• A point that is not in $\tilde{D}^{(m)}$ is not involved in training $g_m$
  • Can we utilize it to validate the performance of $g_m$?
  • Yes, but we care about the overall performance, not just the performance of $g_m$...
Out-Of-Bag (OOB) Error

<table>
<thead>
<tr>
<th></th>
<th>$\tilde{D}^{(1)}$</th>
<th>$\tilde{D}^{(2)}$</th>
<th>$\tilde{D}^{(3)}$</th>
<th>$\tilde{D}^{(4)}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{x}_1, y_1$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>$\tilde{x}_2, y_2$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$\tilde{x}_N, y_N$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>...</td>
</tr>
</tbody>
</table>

Whether a point is in a bootstrapped dataset

- Recall that we learn $g_1, \ldots, g_M$ using $\tilde{D}^{(1)}, \ldots, \tilde{D}^{(M)}$

- Which set of hypothesis can $(\tilde{x}_1, y_1)$ be used for validation?
Out-Of-Bag (OOB) Error

\[ G_n^- : \] the aggregation of hypothesis that \( \tilde{x}_n \) is OOB of

- \( G_1^- = \text{aggregate}(g_3, g_4, \ldots) \)
- \( G_2^- = \text{aggregate}(g_2, g_3, g_4, \ldots) \)
- \( G_N^- = \text{aggregate}(g_1, \ldots) \)

• OOB Error

\[ E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{error}(G_n^-(\tilde{x}_n), y_n) \]
Out-Of-Bag (OOB) Error

\[ E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{error}(G_n^{-}(\hat{x}_n), y_n) \]

- Bagging provided an intrinsic mechanism for us to perform validation
  - We don’t need to split the dataset into training and validation

**Practical issues (you might face this in HW4)**
- What if some \( \hat{x}_n \) appears in all bootstrapped datasets?
  - The probability of this happening is small when the number of bags \( M \) is large
- Let \( S \) be the set of points that is out of bag for at least one bootstrapped dataset
  - \( E_{OOB}(G) = \frac{1}{|S|} \sum_{(\hat{x}_n, y_n) \in S} \text{error}(G_n^{-}(\hat{x}_n), y_n) \)
Random Forest
What We Have Learned

Bagging:
A method to generate and aggregate many high-variance weak learners into a stronger one.

Decision tree:
Various nice properties
Bad generalization
- Due to high variance

Random Forest:
1. Construct many random trees
2. Aggregate the random trees
Random Forest

- Construct many random trees
  - Bootstrapping datasets and learn a max-depth tree for each of them
  - Other randomizations (not required in HW4)
    - When choosing split features, choose from a random subset (instead of all features)
    - Randomly project features (similar to non-linear transformation) for each tree

- Aggregate the random trees
  - Classification: Majority vote $\bar{g}(\hat{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^{M} g_m(\hat{x}) \right)$
  - Regression: Average $\bar{g}(\hat{x}) = \frac{1}{M} \sum_{m=1}^{M} g_m(\hat{x})$
Questions?

• Note that in HW4, you will be asked to implement Bagging Decision Tree and calculate the OOB errors.

• Make sure you know the definitions/algorithm well.
Brief Discussion on Feature Importance

• Not all features are equally important
  • Some features could be redundant -- (birth year, age)
  • Some features might be irrelevant -- feature: name, label: prob of heart attack

• How do we know which features are more important?
  • Linear models:
    • The size of the weight is a proxy for feature importance
    • Applying L1 regularization is one way to reduce the number of features
  • Decision tree:
    • The feature closer to the root is probably more important
  • Random forest:
    • Average “information gain” of all trees is a proxy for feature importance

• See LFD e-Chap 9.2 for more discussion on feature selection
Boosting
Ensemble Learning

• Goal: Utilize a set of weak learners to obtain a strong learner.

• Format of ensemble learning
  • Construct many diverse weak learners
  • Aggregate the weak learners

Bagging:
  • Construct diverse weak learners
    • (Simultaneously) bootstrapping datasets
    • Train weak learners on them
  • Aggregate the weak learners
    • Uniform aggregation

Boosting:
  • Construct diverse weak learners
    • Adaptively generating datasets
    • Train weak learners on them
  • Aggregate the weak learners
    • Weighted aggregation
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

      Alice: Apples are circular

      Teacher:
      Circular is a good feature, but using this feature might make some mistakes

      Let me highlight the mistakes.
      • Make correct images smaller
      • Make incorrect images larger

Example created by Hsuan-Tien Lin
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

  Alice: Apples are circular
  Bob: Apples are red
  Teacher: Yes, many apples are red but it could still make mistakes.

Let me highlight the mistakes.

  • Make correct images smaller
  • Make incorrect images larger

Example created by Hsuan-Tien Lin
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

  • Alice: Apples are **circular**
  • Bob: Apples are **red**
  • Charlie: Apples could be **green**

Example created by Hsuan-Tien Lin
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

  • Alice: Apples are circular
  • Bob: Apples are red
  • Charlie: Apples could be green
  • David: Apples have stems at the top
  • Class: Apples are somewhat circular, somewhat red, possibly green, and may have stems at the top

Example created by Hsuan-Tien Lin
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

Alice: Apples are circular
Bob: Apples are red
Charlie: Apples could be green
David: Apples have stems at the top

Class: Apples are somewhat circular, somewhat red, possibly green, and may have stems at the top

Key steps of this process:
• Learn a simple hypothesis for each dataset
• Iteratively update the dataset to focus on what we got wrong (i.e., create diversity)
• Aggregate the learned simple hypothesis
Outline of a Boosting Algorithm

• Initialize $D_1$ (usually the same as the initial dataset $D$)

• For $t = 1$ to $T$
  • Learn $g_t$ from $D_t$
  • Reweight the distribution and obtain $D_{t+1}$ based on $g_t$ and $D_t$

• Output weighted-aggregate($g_1, ..., g_T$)
  • Classification: $G(\vec{x}) = \bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{T} \sum_{t=1}^{M} \alpha_t g_t(\vec{x}) \right)$

Questions
- How to learn $g_t$ from $D_t$
- How to reweight the distribution and obtain $D_{t+1}$
- How to perform weighted aggregation
Discussion on Re-weighted $D_t$ (What does re-weighting mean?)

• Original Dataset $D = \{(\vec{x}_1, y_1), ..., (\vec{x}_N, y_N)\}$

• Notation of $D_t$
  • $D_t(n)$ is the weight/probability of data point $(\vec{x}_n, y_n)$ in $D_t$
  • $\sum_{n=1}^{N} D_t(n) = 1$

• What is $E_{in}(h)$ on $D_t$? (Expressed as $E_{in}^{(D_t)}(h)$)
  • Re-sample dataset (noisier)
    • Re-sample the dataset from $D$ according to distribution $D_t$
    • Calculate $E_{in}$ on the re-sampled dataset as usual

• Calculate weighted error
  • $E_{in}^{(D_t)}(h) = \sum_{n=1}^{N} D_t(n) \text{error}(h(\vec{x}_n), y_n)$

When $D_t(n) = 1/N$. This reduces to standard definition of $E_{in}$. 