Lecture 10: Random Forest

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science & Information Engineering
National Taiwan University (國立台灣大學資訊工程系)
Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 9: Decision Tree
- recursive branching (purification) for conditional aggregation of constant hypotheses

Lecture 10: Random Forest
- Random Forest Algorithm
- Out-Of-Bag Estimate
- Feature Selection
- Random Forest in Action

3. Distilling Implicit Features: Extraction Models
Random Forest

Recall: Bagging and Decision Tree

Bagging

function Bag(\mathcal{D}, \mathcal{A})
For t = 1, 2, \ldots, T

1. request size-\(N'\) data \(\tilde{\mathcal{D}}_t\) by bootstrapping with \(\mathcal{D}\)
2. obtain base \(g_t\) by \(\mathcal{A}(\tilde{\mathcal{D}}_t)\)

return \(G = \text{Uniform}(\{g_t\})\)
Bagging

function \text{Bag}(\mathcal{D}, A) 

For \( t = 1, 2, \ldots, T \)

1. request size-\( N' \) data \( \tilde{\mathcal{D}}_t \) by bootstrapping with \( \mathcal{D} \)
2. obtain base \( g_t \) by \( A(\tilde{\mathcal{D}}_t) \)

return \( G = \text{Uniform}(\{g_t\}) \)

Decision Tree

function \text{DTree}(\mathcal{D})

if termination return base \( g_t \)
else

1. learn \( b(x) \) and split \( \mathcal{D} \) to \( \mathcal{D}_c \) by \( b(x) \)
2. build \( G_c \leftarrow \text{DTree}(\mathcal{D}_c) \)
3. return \( G(x) = \sum_{c=1}^{C} \left[ b(x) = c \right] G_c(x) \)
### Bagging

**function** `Bag(D, A)`

1. For $t = 1, 2, \ldots, T$
   - request size-$N'$ data $\tilde{D}_t$ by bootstrapping with $D$
   - obtain base $g_t$ by $A(\tilde{D}_t)$
2. return $G = \text{Uniform}\{g_t\}$

—reduces variance by voting/averaging

### Decision Tree

**function** `DTree(D)`

1. if termination return base $g_t$
2. else
   1. learn $b(\mathbf{x})$ and split $D$ to $D_c$ by $b(\mathbf{x})$
   2. build $G_c \leftarrow \text{DTree}(D_c)$
   3. return $G(\mathbf{x}) = \sum_{c=1}^{C} \left[ b(\mathbf{x}) = c \right] G_c(\mathbf{x})$

—large variance especially if fully-grown
Recall: Bagging and Decision Tree

**Bagging**

function $\text{Bag}(\mathcal{D}, \mathcal{A})$

For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$

2. obtain base $g_t$ by $\mathcal{A}(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

—reduces variance by voting/averaging

**Decision Tree**

function $\text{DTree}(\mathcal{D})$

if termination return base $g_t$

else

1. learn $b(x)$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(x)$

2. build $G_c \leftarrow \text{DTree}(\mathcal{D}_c)$

3. return $G(x) = \sum_{c=1}^{C} [b(x) = c] G_c(x)$

—large variance especially if fully-grown
Random Forest

Random Forest Algorithm

Recall: Bagging and Decision Tree

Bagging

function $\text{Bag}(\mathcal{D}, \mathcal{A})$

For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain base $g_t$ by $\mathcal{A}(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

—reduces variance by voting/averaging

Decision Tree

function $\text{DTree}(\mathcal{D})$

if termination

return base $g_t$

else

1. learn $b(x)$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(x)$
2. build $G_c \leftarrow \text{DTree}(\mathcal{D}_c)$
3. return $G(x) = \sum_{c=1}^{C} \left[ b(x) = c \right] G_c(x)$

—large variance especially if fully-grown

Putting them together?
Random Forest

Recall: Bagging and Decision Tree

Bagging

function Bag(D, A)
For t = 1, 2, ..., T
    1. request size-$N'$ data $\tilde{D}_t$ by bootstrapping with $D$
    2. obtain base $g_t$ by $A(\tilde{D}_t)$
return $G = \text{Uniform}(\{g_t\})$

—reduces variance by voting/averaging

Decision Tree

function DTree(D)
if termination return base $g_t$
else
    1. learn $b(x)$ and split $D$ to $D_c$ by $b(x)$
    2. build $G_c \leftarrow \text{DTree}(D_c)$
    3. return $G(x) = \frac{1}{c} \sum_{c=1}^{C} [b(x) = c] G_c(x)$

—large variance especially if fully-grown

putting them together? (i.e. aggregation of aggregation :-) )

Hsuan-Tien Lin (NTU CSIE)
Random Forest (RF)

random forest (RF) = bagging + fully-grown C&RT decision tree
Random Forest (RF)

random forest (RF) = bagging + fully-grown C&RT decision tree

function RandomForest(D)
For t = 1, 2, ..., T
Random Forest (RF)

random forest (RF) = bagging + fully-grown C&RT decision tree

function RandomForest(D)
For t = 1, 2, . . . , T
   1. request size-N’ data ˜D_t by bootstrapping with D

- highly parallel/efficient
- inherit pros of C&RT
- eliminate cons of fully-grown tree
Random Forest (RF)

random forest (RF) = bagging + fully-grown C&RT decision tree

**function RandomForest**($\mathcal{D}$)

For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by DTree($\tilde{\mathcal{D}}_t$)

**function DTree**($\mathcal{D}$)

if termination return base $g_t$
else

1. learn $b(x)$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(x)$
2. build $G_c \leftarrow$ DTree($\mathcal{D}_c$)
3. return $G(x) = \sum_{c=1}^{C} [b(x) = c] G_c(x)$
Random Forest (RF)

**random forest (RF) = bagging + fully-grown C&RT decision tree**

**function RandomForest(D)**
For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{D}_t$ by bootstrapping with $D$
2. obtain tree $g_t$ by DTree($\tilde{D}_t$)

return $G = \text{Uniform}(\{g_t\})$

**function DTree(D)**
if termination return base $g_t$
else

1. learn $b(x)$ and split $D$ to $D_c$ by $b(x)$
2. build $G_c \leftarrow \text{DTree}(D_c)$
3. return $G(x) =$

$$\sum_{c=1}^{C} [b(x) = c] G_c(x)$$
Random Forest (RF)

**random forest (RF) = bagging + fully-grown C&RT decision tree**

function RandomForest($\mathcal{D}$)
For $t = 1, 2, \ldots, T$
  1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
  2. obtain tree $g_t$ by DTree($\tilde{\mathcal{D}}_t$)
return $G = \text{Uniform}(\{g_t\})$

function DTree($\mathcal{D}$)
if termination return base $g_t$
else
  1. learn $b(x)$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(x)$
  2. build $G_c \leftarrow \text{DTree}(\mathcal{D}_c)$
  3. return $G(x) =$
     $$\sum_{c=1}^{C} \mathbb{I}[b(x) = c] G_c(x)$$

• highly **parallel/efficient** to learn
Random Forest (RF) = bagging + fully-grown C&RT decision tree

function RandomForest($D$)
For $t = 1, 2, \ldots, T$
① request size-$N'$ data $\tilde{D}_t$ by bootstrapping with $D$
② obtain tree $g_t$ by DTree($\tilde{D}_t$)
return $G = \text{Uniform}(\{g_t\})$

function DTree($D$)
if termination return base $g_t$
else
① learn $b(x)$ and split $D$ to $D_c$ by $b(x)$
② build $G_c \leftarrow \text{DTree}(D_c)$
③ return $G(x) = \sum_{c=1}^{C} \left[ b(x) = c \right] G_c(x)$

- highly parallel/efficient to learn
- inherit pros of C&RT
Random Forest (RF)

random forest (RF) = bagging + fully-grown C&RT decision tree

**function RandomForest**($\mathcal{D}$)

For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\mathcal{D}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by DTree($\mathcal{D}_t$)

return $G = \text{Uniform}(\{g_t\})$

**function DTree**($\mathcal{D}$)

if termination

return base $g_t$

else

1. learn $b(x)$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(x)$
2. build $G_c \leftarrow \text{DTree}(\mathcal{D}_c)$
3. return $G(x) = \sum_{c=1}^{C} \left[ b(x) = c \right] G_c(x)$

- highly *parallel*/efficient to learn
- inherit pros of C&RT
- eliminate cons of fully-grown tree
Diversifying by Feature Projection

recall: **data randomness** for **diversity** in **bagging**

randomly **sample** $N'$ **examples** from $\mathcal{D}$
Diversifying by Feature Projection

recall: **data randomness** for **diversity** in **bagging**

randomly **sample** $N'$ **examples** from $\mathcal{D}$

another possibility for **diversity**:  

randomly **sample** $d'$ **features** from $\mathbf{x}$
Diversifying by Feature Projection

recall: data randomness for diversity in bagging

randomly sample $N'$ examples from $\mathcal{D}$

another possibility for diversity:

randomly sample $d'$ features from $\mathbf{x}$

- when sampling index $i_1, i_2, \ldots, i_{d'}$: $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}})$
Diversifying by Feature Projection

recall: **data randomness** for **diversity** in **bagging**

randomly **sample** \( N' \) **examples** from \( D \)

another possibility for **diversity**:

randomly **sample** \( d' \) **features** from \( x \)

- when sampling index \( i_1, i_2, \ldots, i_{d'} \): \( \Phi(x) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}}) \)
- \( Z \in \mathbb{R}^{d'} \): a **random subspace** of \( X \in \mathbb{R}^d \)
Diversifying by Feature Projection

recall: **data randomness** for **diversity** in bagging

randomly **sample** $N'$ **examples** from $D$

another possibility for **diversity**:

randomly **sample** $d'$ **features** from $x$

- when sampling index $i_1, i_2, \ldots, i_{d'}$: $\Phi(x) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}})$
- $\mathcal{Z} \in \mathbb{R}^{d'}$: a **random subspace** of $\mathcal{X} \in \mathbb{R}^d$
- often $d' \ll d$, efficient for large $d$
Diversifying by Feature Projection

recall: data randomness for diversity in bagging

randomly sample \( N' \) examples from \( D \)

another possibility for diversity:

randomly sample \( d' \) features from \( x \)

- when sampling index \( i_1, i_2, \ldots, i_{d'} \): \( \Phi(x) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}}) \)
- \( Z \in \mathbb{R}^{d'} \): a random subspace of \( X \in \mathbb{R}^d \)
- often \( d' \ll d \), efficient for large \( d \)
  —can be generally applied on other models
Diversifying by Feature Projection

Recall: **data randomness** for **diversity** in bagging

Randomly **sample** $N'$ examples from $\mathcal{D}$

Another possibility for **diversity**:

Randomly **sample** $d'$ features from $\mathbf{x}$

- When sampling index $i_1, i_2, \ldots, i_{d'}$: $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}})$
- $\mathcal{Z} \in \mathbb{R}^{d'}$: a **random subspace** of $\mathcal{X} \in \mathbb{R}^{d}$
- Often $d' \ll d$, efficient for large $d$ — can be generally applied on other models
- Original RF **re-sample new subspace for each** $b(\mathbf{x})$ in C&RT
Diversifying by Feature Projection

Recall: **data randomness** for **diversity** in **bagging**

Randomly sample \( N' \) examples from \( D \)

Another possibility for **diversity**:

Randomly sample \( d' \) features from \( x \)

- When sampling index \( i_1, i_2, \ldots, i_{d'}: \Phi(x) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}}) \)
- \( Z \in \mathbb{R}^{d'} \): a random subspace of \( X \in \mathbb{R}^d \)
- Often \( d' \ll d \), efficient for large \( d \)
  —can be generally applied on other models
- Original RF re-sample new subspace for each \( b(x) \) in C&RT

\[ RF = \text{bagging} + \text{random-subspace C&RT} \]
Diversifying by Feature Expansion

randomly sample \( d' \) features from \( \mathbf{x} \): \( \Phi(\mathbf{x}) = \mathbf{P} \cdot \mathbf{x} \)
with row \( i \) of \( \mathbf{P} \) sampled randomly \( \in \) natural basis
Diversifying by Feature Expansion

randomly sample $d'$ features from $x$: $\Phi(x) = P \cdot x$
with row $i$ of $P$ sampled randomly $\in$ natural basis

more powerful features for diversity: row $i$ other than natural basis
Diversifying by Feature Expansion

randomly **sample \( d' \) features** from \( \mathbf{x} \): \( \Phi(\mathbf{x}) = \mathbf{P} \cdot \mathbf{x} \)

with **row \( i \) of \( \mathbf{P} \) sampled randomly** \( \in \) **natural basis**

more **powerful** features for **diversity**: row \( i \) other than natural basis

- **projection** (combination) with random row \( \mathbf{p}_i \) of \( \mathbf{P} \): \( \phi_i(\mathbf{x}) = \mathbf{p}_i^T \mathbf{x} \)
Diversifying by Feature Expansion

randomly **sample** $d'$ **features** from $\mathbf{x}$: $\Phi(\mathbf{x}) = \mathbf{P} \cdot \mathbf{x}$
with *row* $i$ of $\mathbf{P}$ sampled randomly $\in$ **natural basis**

more **powerful** features for **diversity**: row $i$ other than natural basis

- **projection** (combination) with random row $\mathbf{p}_i$ of $\mathbf{P}$: $\phi_i(\mathbf{x}) = \mathbf{p}_i^T \mathbf{x}$
- often consider **low-dimensional** projection:
  only $d''$ **non-zero** components in $\mathbf{p}_i$
Diversifying by Feature Expansion

randomly \textbf{sample} \(d'\) features from \(x: \Phi(x) = P \cdot x\)
with \textit{row} \(i\) of \(P\) sampled randomly \(\in\) natural basis

more \textbf{powerful} features for \textit{diversity}: \textit{row} \(i\) other than natural basis

- \textbf{projection} (combination) with random row \(p_i\) of \(P: \phi_i(x) = p_i^T x\)
- often consider \textbf{low-dimensional} projection: only \(d''\) \textbf{non-zero} components in \(p_i\)
- includes \textbf{random subspace} as \textbf{special case}: \(d'' = 1\) and \(p_i \in\) natural basis
Diversifying by Feature Expansion

randomly **sample** $d'$ **features** from $\mathbf{x}$: $\Phi(\mathbf{x}) = \mathbf{P} \cdot \mathbf{x}$

with *row* $i$ of $\mathbf{P}$ sampled randomly $\in$ natural basis

more **powerful** features for **diversity**: row $i$ other than natural basis

- **projection** (combination) with random row $\mathbf{p}_i$ of $\mathbf{P}$: $\phi_i(\mathbf{x}) = \mathbf{p}_i^T \mathbf{x}$
- often consider **low-dimensional** projection: only $d''$ non-zero components in $\mathbf{p}_i$
- includes **random subspace** as special case: $d'' = 1$ and $\mathbf{p}_i \in$ natural basis
- original RF consider $d'$ random **low-dimensional** projections for each $b(\mathbf{x})$ in C&RT
Diversifying by Feature Expansion

Randomly sample \(d'\) features from \(x\): \(\Phi(x) = P \cdot x\) with \(\text{row } i \text{ of } P \text{ sampled randomly} \in \text{natural basis}\).

More powerful features for diversity: row \(i\) other than natural basis

- **projection** (combination) with random row \(p_i\) of \(P\): \(\phi_i(x) = p_i^T x\)
- Often consider **low-dimensional** projection: only \(d''\) non-zero components in \(p_i\)
- Includes **random subspace** as special case: \(d'' = 1\) and \(p_i \in \text{natural basis}\)
- Original RF consider \(d'\) random low-dimensional projections for each \(b(x)\) in C&RT

\[RF = \text{bagging} + \text{random-combination C&RT}– \text{randomness everywhere!}\]
Within RF that contains random-combination C&RT trees, which of the following hypothesis is equivalent to each branching function $b(x)$ within the tree?

1. a constant
2. a decision stump
3. a perceptron
4. none of the other choices
Within RF that contains random-combination C&RT trees, which of the following hypothesis is equivalent to each branching function $b(x)$ within the tree?

1. a constant
2. a decision stump
3. a perceptron
4. none of the other choices

Reference Answer: 3

In each $b(x)$, the input vector $x$ is first projected by a random vector $v$ and then thresholded to make a binary decision, which is exactly what a perceptron does.
Bagging Revisited

**Bagging**

function $\text{Bag}(\mathcal{D}, \mathcal{A})$

For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$
   by bootstrapping with $\mathcal{D}$

2. obtain base $g_t$ by $\mathcal{A}(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$
Bagging Revisited

**Bagging**

```markdown
function Bag(D, A)
For t = 1, 2, ..., T
  1. request size-N' data \( \tilde{D}_t \) by bootstrapping with \( D \)
  2. obtain base \( g_t \) by \( A(\tilde{D}_t) \)
return \( G = \text{Uniform}([g_t]) \)
```

<table>
<thead>
<tr>
<th></th>
<th>( g_1 )</th>
<th>( g_2 )</th>
<th>( g_3 )</th>
<th>( \cdots )</th>
<th>( g_T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x_1, y_1) )</td>
<td>( \tilde{D}_1 )</td>
<td>_*_\</td>
<td>( \tilde{D}_3 )</td>
<td></td>
<td>( \tilde{D}_T )</td>
</tr>
<tr>
<td>( (x_2, y_2) )</td>
<td>_*_\</td>
<td>_*_\</td>
<td>( \tilde{D}_3 )</td>
<td></td>
<td>( \tilde{D}_T )</td>
</tr>
<tr>
<td>( (x_3, y_3) )</td>
<td>_*_\</td>
<td>( \tilde{D}_2 )</td>
<td>_*_\</td>
<td></td>
<td>( \tilde{D}_T )</td>
</tr>
<tr>
<td>( \cdots )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (x_N, y_N) )</td>
<td>( \tilde{D}_1 )</td>
<td>( \tilde{D}_2 )</td>
<td>_*_\</td>
<td></td>
<td>_*_\</td>
</tr>
</tbody>
</table>
```
Bagging Revisited

**Bagging**

function \( \text{Bag}(D, A) \)

For \( t = 1, 2, \ldots, T \)

1. request size-\( N' \) data \( \tilde{D}_t \) by bootstrapping with \( D \)
2. obtain base \( g_t \) by \( A(\tilde{D}_t) \)

return \( G = \text{Uniform}(\{g_t\}) \)

| \((x_1, y_1)\) | \(\tilde{D}_1\) | \(\star\) | \(\tilde{D}_3\) | \(\tilde{D}_T\) |
| \((x_2, y_2)\) | \(\star\) | \(\star\) | \(\tilde{D}_3\) | \(\tilde{D}_T\) |
| \((x_3, y_3)\) | \(\star\) | \(\tilde{D}_2\) | \(\star\) | \(\tilde{D}_T\) |
| \cdots | \(\star\) | \(\tilde{D}_2\) | \(\cdots\) | \(\cdots\) |
| \((x_N, y_N)\) | \(\tilde{D}_1\) | \(\tilde{D}_2\) | \(\star\) | \(\star\) |

\(\star\) in \( t \)-th column: not used for obtaining \( g_t \)
—called **out-of-bag (OOB) examples** of \( g_t \)
Number of OOB Examples

**OOB (in ★) ⇔ not sampled after \( N' \) drawings**

\[
\text{OOB size per } g \approx \frac{1}{e}
\]
Number of OOB Examples

OOB (in ⋄) \iff \text{not sampled after } N' \text{ drawings}

if \( N' = N \)
Number of OOB Examples

OOB (in ★) $\iff$ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: 

\[
\left(1 - \frac{1}{N}\right) \frac{N}{N} \approx \frac{1}{e}
\]

\[
\frac{N}{N-1} \approx 1 + \frac{1}{N}
\]

\[
\approx \frac{1}{e}
\]
Number of OOB Examples

OOB (in ⋄) ⇐⇒ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: $\left(1 - \frac{1}{N}\right)^N$
Number of OOB Examples

OOB (in ⋄) ⇐⇒ not sampled after \( N' \) drawings

\[
\text{if } N' = N
\]

- probability for \((x_n, y_n)\) to be OOB for \(g_t\): \((1 - \frac{1}{N})^N\)
- if \(N\) large:

\[
\left(1 - \frac{1}{N}\right)^N = e^{-1/N}
\]
Number of OOB Examples

OOB (in ⋄) ⇐⇒ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: $(1 - \frac{1}{N})^N$
- if $N$ large:

$$\left(1 - \frac{1}{N}\right)^N = \left(\frac{1}{N}\right)^N = \approx e^{-\frac{1}{N}}$$
Number of OOB Examples

OOB (in ⋄) $\iff$ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: $\left(1 - \frac{1}{N}\right)^N$
- if $N$ large:

$$\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(\frac{N}{N-1}\right)^N} = \left(\frac{N}{N-1}\right)^N$$
Number of OOB Examples

OOB (in ⋄) ⇐⇒ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: $\left(1 - \frac{1}{N}\right)^N$
- if $N$ large:

$$\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(\frac{N}{N-1}\right)^N} = \frac{1}{\left(1 + \right)^N}$$
### Number of OOB Examples

**OOB (in ⭐) ↔ not sampled after \( N' \) drawings**

<table>
<thead>
<tr>
<th>if ( N' = N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>• probability for ((x_n, y_n)) to be OOB for ( g_t ): ( \left(1 - \frac{1}{N}\right)^N )</td>
</tr>
<tr>
<td>• if ( N ) large:</td>
</tr>
</tbody>
</table>
| \[
\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(\frac{N}{N-1}\right)^N} = \frac{1}{\left(1 + \frac{1}{N-1}\right)^N}
\]

Number of OOB Examples

OOB (in ★) ⇐⇒ not sampled after $N'$ drawings

if $N' = N$

- probability for $(x_n, y_n)$ to be OOB for $g_t$: $(1 - \frac{1}{N})^N$
- if $N$ large:

\[
\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(1 + \frac{1}{N-1}\right)^N} \approx \frac{1}{e}
\]
Random Forest

Out-Of-Bag Estimate

Number of OOB Examples

OOB (in *) ⇐⇒ not sampled after \( N' \) drawings

if \( N' = N \)

- probability for \((x_n, y_n)\) to be OOB for \( g_t \): \( \left(1 - \frac{1}{N}\right)^N \)
- if \( N \) large:

\[
\left(1 - \frac{1}{N}\right)^N = \frac{1}{\left(\frac{N}{N-1}\right)^N} = \frac{1}{\left(1 + \frac{1}{N-1}\right)^N} \approx \frac{1}{e}
\]

OOB size per \( g_t \) \( \approx \frac{1}{e} N \)
**Random Forest**

**Out-Of-Bag Estimate**

### OOB versus Validation

#### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>...</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$\star$</td>
<td>$\tilde{D}_2$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
<td></td>
</tr>
</tbody>
</table>
### OOB versus Validation

#### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$\cdots$</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>$,*$</td>
<td>$\tilde{D}_3$</td>
<td>$\cdots$</td>
<td>$\tilde{D}_T$</td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>$,*$</td>
<td>$,*$</td>
<td>$\tilde{D}_3$</td>
<td>$\cdots$</td>
<td>$\tilde{D}_T$</td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$,*$</td>
<td>$\tilde{D}_2$</td>
<td>$,*$</td>
<td>$\cdots$</td>
<td>$\tilde{D}_T$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$,*$</td>
<td>$,*$</td>
<td>$,*$</td>
<td>$,*$</td>
</tr>
</tbody>
</table>

#### Validation

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$\cdots$</th>
<th>$g_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Random Forest**

**Out-Of-Bag Estimate**

$\text{OOB versus Validation}$

- **OOB**
  - Random examples not used during training
  - "Enough" examples used

- **Validation**
  - $\pi$ used to validate $g_t$?
  - Example: $\text{OOB}(g) = \frac{1}{N} \sum_{n=1}^{N} \text{err}(y_n, g_n(x_n))$
  - where $g_n$ contains only trees that $x_n$ is OOB of, such as $g_N(x) = \text{average}(g_2, g_3, g_T)$
### OOB versus Validation

#### OOB

<table>
<thead>
<tr>
<th>((x_1, y_1))</th>
<th>(g_1)</th>
<th>(g_2)</th>
<th>(g_3)</th>
<th>(\ldots)</th>
<th>(g_T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tilde{D}_1)</td>
<td>*</td>
<td>(\tilde{D}_3)</td>
<td>(\tilde{D}_T)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((x_2, y_2))</td>
<td>*</td>
<td>*</td>
<td>(\tilde{D}_3)</td>
<td>(\tilde{D}_T)</td>
<td></td>
</tr>
<tr>
<td>((x_3, y_3))</td>
<td>*</td>
<td>(\tilde{D}_2)</td>
<td>*</td>
<td>(\tilde{D}_T)</td>
<td></td>
</tr>
<tr>
<td>(\ldots)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>((x_N, y_N))</td>
<td>(\tilde{D}_1)</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

#### Validation

<table>
<thead>
<tr>
<th>((x_1, y_1))</th>
<th>(g_1^{-})</th>
<th>(g_2^{-})</th>
<th>(\ldots)</th>
<th>(g_M^{-})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_{train})</td>
<td>(D_{train})</td>
<td>(D_{train})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D_{val})</td>
<td>(D_{val})</td>
<td>(D_{val})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D_{val})</td>
<td>(D_{val})</td>
<td>(D_{val})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D_{train})</td>
<td>(D_{train})</td>
<td>(D_{train})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- * like \(D_{val}\): ‘enough’ random examples unused during training
### OOB versus Validation

#### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$\cdots$</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$\star$</td>
<td>$\tilde{D}_2$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
</tr>
</tbody>
</table>

#### Validation

<table>
<thead>
<tr>
<th></th>
<th>$g_1$-</th>
<th>$g_2$-</th>
<th>$\cdots$</th>
<th>$g_M$-</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $\star$ like $D_{\text{val}}$: ‘enough’ random examples unused during training
- use $\star$ to validate $g_t$?
### OOB versus Validation

#### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$\cdots$</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$\star$</td>
<td>$\tilde{D}_2$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
<td></td>
</tr>
</tbody>
</table>

#### Validation

<table>
<thead>
<tr>
<th></th>
<th>$g_{\tilde{1}}$</th>
<th>$g_{\tilde{2}}$</th>
<th>$\cdots$</th>
<th>$g_{\tilde{M}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td>$\mathcal{D}_{\text{val}}$</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td>$\mathcal{D}_{\text{train}}$</td>
<td></td>
</tr>
</tbody>
</table>

- $\star$ like $\mathcal{D}_{\text{val}}$: ‘enough’ random examples unused during training
- use $\star$ to validate $g_t$? easy, but **rarely needed**
Random Forest
Out-Of-Bag Estimate

OOB versus Validation

---

**OOB**

<table>
<thead>
<tr>
<th>$(x_1, y_1)$</th>
<th>$\tilde{D}_1$</th>
<th>$\star$</th>
<th>$\tilde{D}_3$</th>
<th>$\tilde{D}_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_2, y_2)$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$\star$</td>
<td>$\tilde{D}_2$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
</tr>
</tbody>
</table>

**Validation**

<table>
<thead>
<tr>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$\ldots$</th>
<th>$g_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
</tr>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
</tr>
</tbody>
</table>

- $\star$ like $D_{\text{val}}$: ‘enough’ random examples unused during training
- use $\star$ to validate $g_t$? easy, but rarely needed
- use $\star$ to validate $G$?
Random Forest

Out-Of-Bag Estimate

**OOB versus Validation**

### OOB

<table>
<thead>
<tr>
<th>(x₁, y₁)</th>
<th>g₁</th>
<th>⃗D₁</th>
<th>⃗D₃</th>
<th>⃗Dₜ</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x₂, y₂)</td>
<td>⋆</td>
<td>⋆</td>
<td>⃗D₃</td>
<td>⃗Dₜ</td>
</tr>
<tr>
<td>(x₃, y₃)</td>
<td>⋆</td>
<td>⃗D₂</td>
<td>⋆</td>
<td>⃗Dₜ</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(xₙ, yₙ)</td>
<td>⃗D₁</td>
<td>⋆</td>
<td>⋆</td>
<td>⋆</td>
</tr>
</tbody>
</table>

### Validation

<table>
<thead>
<tr>
<th>g₁</th>
<th>g₂</th>
<th>⋮</th>
<th>gₘ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dₚrain</td>
<td>Dₚrain</td>
<td></td>
<td>Dₚrain</td>
</tr>
<tr>
<td>Dₚval</td>
<td>Dₚval</td>
<td></td>
<td>Dₚval</td>
</tr>
<tr>
<td>Dₚval</td>
<td>Dₚval</td>
<td></td>
<td>Dₚval</td>
</tr>
</tbody>
</table>

• ⋆ like Dₚval: ‘enough’ random examples unused during training
• use ⋆ to validate gₜ? easy, but rarely needed
• use ⋆ to validate G?
  with G⁻ₙ contains only trees that xₙ is OOB of,
  such as G⁻ₙ(x) = average(g₂, g₃, gₜ)
**OOB versus Validation**

### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$\cdots$</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>$\star$</td>
<td>$\tilde{D}_2$</td>
<td>$\star$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>$\star$</td>
<td>$\star$</td>
<td>$\star$</td>
<td></td>
</tr>
</tbody>
</table>

### Validation

<table>
<thead>
<tr>
<th></th>
<th>$g^-_1$</th>
<th>$g^-_2$</th>
<th>$\cdots$</th>
<th>$g^-_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
</tr>
</tbody>
</table>

- $\star$ like $D_{\text{val}}$: ‘enough’ random examples unused during training
- use $\star$ to validate $g_t$? easy, but **rarely needed**
- use $\star$ to validate $G$?

\[
\text{err}(y_n, G^-_n(x_n)),
\]

with $G^-_n$ contains only trees that $x_n$ is OOB of,

\[
\text{such as } G^-_N(x) = \text{average}(g_2, g_3, g_T)
\]
### OOB versus Validation

<table>
<thead>
<tr>
<th>OOB</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x_1, y_1) )</td>
<td>( \tilde{D}_1 )</td>
</tr>
<tr>
<td>( (x_2, y_2) )</td>
<td>( \tilde{D}_3 )</td>
</tr>
<tr>
<td>( (x_3, y_3) )</td>
<td>( \tilde{D}_2 )</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>( (x_N, y_N) )</td>
<td>( \tilde{D}_1 )</td>
</tr>
</tbody>
</table>

- \( \star \) like \( D_{val} \): ‘enough’ random examples unused during training
- use \( \star \) to validate \( g_t \)? easy, but rarely needed
- use \( \star \) to validate \( G \)?
  
  \[
  E_{oob}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{err}(y_n, G^-_n(x_n)),
  \]

  with \( G^-_n \) contains only trees that \( x_n \) is OOB of,

  such as \( G^-_N(x) = \text{average}(g_2, g_3, g_T) \)
## OOB versus Validation

### OOB

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>⋯</th>
<th>$g_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1, y_1)$</td>
<td>$\tilde{D}_1$</td>
<td>*</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_2, y_2)$</td>
<td>*</td>
<td>*</td>
<td>$\tilde{D}_3$</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>$(x_3, y_3)$</td>
<td>*</td>
<td>$\tilde{D}_2$</td>
<td>*</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
<tr>
<td>⋮</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(x_N, y_N)$</td>
<td>$\tilde{D}_1$</td>
<td>*</td>
<td>*</td>
<td>$\tilde{D}_T$</td>
<td></td>
</tr>
</tbody>
</table>

### Validation

<table>
<thead>
<tr>
<th></th>
<th>$g_1^-$</th>
<th>$g_2^-$</th>
<th>⋯</th>
<th>$g_M^-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td>$D_{\text{val}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td>$D_{\text{train}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ⋆ like $D_{\text{val}}$: ‘enough’ random examples unused during training
- use ⋆ to validate $g_t$? easy, but rarely needed
- use ⋆ to validate $G$? \( E_{\text{oob}}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{err}(y_n, G_n^-(x_n)) \), with $G_n^-$ contains only trees that $x_n$ is OOB of, such as $G_N^-(x) = \text{average}(g_2, g_3, g_T)$

\( E_{\text{oob}} \): self-validation of bagging/RF
Random Forest
Out-Of-Bag Estimate

Model Selection by OOB Error

Previously: by Best $E_{val}$

\[
g_{m^*} = A_{m^*}(\mathcal{D})
\]
\[
m^* = \arg\min_{1 \leq m \leq M} E_m
\]
\[
E_m = E_{val}(A_m(\mathcal{D}_{train}))
\]
Model Selection by OOB Error

Previously: by Best $E_{\text{val}}$

\[
\begin{align*}
g_{m^*} &= A_{m^*}(D) \\
m^* &= \text{argmin}_{1 \leq m \leq M} E_m \\
E_m &= E_{\text{val}}(A_m(D_{\text{train}}))
\end{align*}
\]

RF: by Best $E_{\text{oob}}$

\[
\begin{align*}
G_{m^*} &= \text{RF}_{m^*}(D) \\
m^* &= \text{argmin}_{1 \leq m \leq M} E_m \\
E_m &= E_{\text{oob}}(\text{RF}_m(D))
\end{align*}
\]

- use $E_{\text{oob}}$ for self-validation
Random Forest  
Out-Of-Bag Estimate

Model Selection by OOB Error

Previously: by Best $E_{\text{val}}$

$$g_{m^*} = \mathcal{A}_{m^*}(\mathcal{D})$$
$$m^* = \arg\min_{1 \leq m \leq M} E_m$$
$$E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}}))$$

RF: by Best $E_{\text{oob}}$

$$G_{m^*} = \text{RF}_{m^*}(\mathcal{D})$$
$$m^* = \arg\min_{1 \leq m \leq M} E_m$$
$$E_m = E_{\text{oob}}(\text{RF}_m(\mathcal{D}))$$

• use $E_{\text{oob}}$ for self-validation —of RF parameters such as $d''$
Model Selection by OOB Error

Previously: by Best $E_{\text{val}}$

\[
g_m^* = \mathcal{A}_m^*(\mathcal{D})
\]
\[
m^* = \arg\min_{1 \leq m \leq M} E_m
\]
\[
E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}}))
\]

RF: by Best $E_{\text{OOB}}$

\[
G_m^* = \text{RF}_m^*(\mathcal{D})
\]
\[
m^* = \arg\min_{1 \leq m \leq M} E_m
\]
\[
E_m = E_{\text{OOB}}(\text{RF}_m(\mathcal{D}))
\]

- use $E_{\text{OOB}}$ for self-validation —of RF parameters such as $d''$
- no re-training needed
**Model Selection by OOB Error**

### Previously: by Best $E_{\text{val}}$

<table>
<thead>
<tr>
<th>$g_{m^*}$</th>
<th>$= \mathcal{A}_{m^*}(\mathcal{D})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m^*$</td>
<td>$= \arg\min_{1 \leq m \leq M} E_m$</td>
</tr>
<tr>
<td>$E_m$</td>
<td>$= E_{\text{val}}(\mathcal{A}<em>m(\mathcal{D}</em>{\text{train}}))$</td>
</tr>
</tbody>
</table>

### RF: by Best $E_{\text{oob}}$

<table>
<thead>
<tr>
<th>$G_{m^*}$</th>
<th>$= \text{RF}_{m^*}(\mathcal{D})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m^*$</td>
<td>$= \arg\min_{1 \leq m \leq M} E_m$</td>
</tr>
<tr>
<td>$E_m$</td>
<td>$= E_{\text{oob}}(\text{RF}_m(\mathcal{D}))$</td>
</tr>
</tbody>
</table>

- use $E_{\text{oob}}$ for self-validation —of RF parameters such as $d''$
- no re-training needed

$E_{\text{oob}}$ often **accurate** in practice
For a data set with $N = 1126$, what is the probability that $(x_{1126}, y_{1126})$ is not sampled after bootstrapping $N' = N$ samples from the data set?

1. 0.113
2. 0.368
3. 0.632
4. 0.887
For a data set with $N = 1126$, what is the probability that $\mathbf{(x_{1126}, y_{1126})}$ is not sampled after bootstrapping $N' = N$ samples from the data set?

1. 0.113
2. 0.368
3. 0.632
4. 0.887

Reference Answer: 2

The value of $(1 - \frac{1}{N})^N$ with $N = 1126$ is about 0.367716, which is close to $\frac{1}{e} = 0.367879$. 
Feature Selection

for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- redundant features: like keeping one of 'age' and 'full birthday'
- irrelevant features: like insurance type for cancer prediction

$\Phi(\mathbf{x}) = (x_{i1}, x_{i2}, \ldots, x_{id}')$ with $d' < d$ for $g(\Phi(\mathbf{x}))$

Advantages:
- efficiency: simpler hypothesis and shorter prediction time
- generalization: 'feature noise' removed

Disadvantages:
- computation: 'combinatorial' optimization in training
- overfit: 'combinatorial' selection
- mis-interpretability

Decision tree: a rare model with built-in feature selection
for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction
for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ subset-transform \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}}) \)

with \( d' < d \) for \( g(\Phi(\mathbf{x})) \)
Feature Selection

for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ **subset-transform** \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}}) \)

with \( d' < d \) for \( g(\Phi(\mathbf{x})) \)

advantages:

• **efficiency**: simpler hypothesis and shorter prediction time
• **generalization**: ‘feature noise’ removed
• **interpretability**

disadvantages:

• **computation**: ‘combinatorial’ optimization in training
• **overfit**: ‘combinatorial’ selection
• mis-interpretability decision tree: a rare model

Hsuan-Tien Lin (NTU CSIE)
Random Forest

Feature Selection

for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ **subset-transform** \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}}) \)

\[ \text{with } d' < d \text{ for } g(\Phi(\mathbf{x})) \]

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ subset-transform $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}})$

with $d' < d$ for $g(\Phi(\mathbf{x}))$

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ **subset-transform** \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}}) \)

with \( d' < d \) for \( g(\Phi(\mathbf{x})) \)

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**
Random Forest

Feature Selection

for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ **subset-transform** $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_d'})$

with $d' < d$ for $g(\Phi(\mathbf{x}))$

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**

disadvantages:
for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ subset-transform \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_d'}) \)

- with \( d' < d \) for \( g(\Phi(\mathbf{x})) \)

**Advantages:**
- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**

**Disadvantages:**
- **computation**: ‘combinatorial’ optimization in training

Hsuan-Tien Lin (NTU CSIE)
Feature Selection

for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ subset-transform $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}})$ with $d' < d$ for $g(\Phi(\mathbf{x}))$

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**

disadvantages:

- **computation**: ‘combinatorial’ optimization in training
- **overfit**: ‘combinatorial’ selection
for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ subset-transform \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_d'}) \)

with \( d' < d \) for \( g(\Phi(\mathbf{x})) \)

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**

disadvantages:

- **computation**: ‘combinatorial’ optimization in training
- **overfit**: ‘combinatorial’ selection
- **mis-interpretability**
Feature Selection

for \( \mathbf{x} = (x_1, x_2, \ldots, x_d) \), want to remove

- **redundant** features: like keeping one of ‘age’ and ‘full birthday’
- **irrelevant** features: like insurance type for cancer prediction

and only ‘learn’ **subset-transform** \( \Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}}) \)

\[ \text{with } d' < d \text{ for } g(\Phi(\mathbf{x})) \]

**advantages:**

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: ‘feature noise’ removed
- **interpretability**

**disadvantages:**

- **computation**: ‘combinatorial’ optimization in training
- **overfit**: ‘combinatorial’ selection
- **mis-interpretability**

decision tree: a rare model with **built-in feature selection**
Feature Selection by Importance

idea: if possible to calculate

$$\text{importance}(i) \text{ for } i = 1, 2, \ldots, d$$

then can select $$i_1, i_2, \ldots, i_{d'}$$ of top-$$d'$$ importance
Feature Selection by Importance

idea: if possible to calculate

\[ \text{importance}(i) \text{ for } i = 1, 2, \ldots, d \]

then can select \( i_1, i_2, \ldots, i_{d'} \) of top-\( d' \) importance

importance by linear model

\[
\text{score} = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{d} w_i x_i
\]
Feature Selection by Importance

idea: if possible to calculate

\[
\text{importance}(i) \quad \text{for} \quad i = 1, 2, \ldots, d
\]

then can select \(i_1, i_2, \ldots, i_{d'}\) of top-\(d'\) importance

importance by linear model

\[
\text{score} = w^T x = \sum_{i=1}^{d} w_i x_i
\]

- intuitive estimate: \(\text{importance}(i) = |w_i|\) with some ‘good’ \(w\)
Feature Selection by Importance

idea: if possible to calculate

$$\text{importance}(i) \text{ for } i = 1, 2, \ldots, d$$

then can select \(i_1, i_2, \ldots, i_{d'}\) of top-\(d'\) importance

importance by linear model

$$\text{score} = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{d} w_i x_i$$

- intuitive estimate: \(\text{importance}(i) = |w_i|\) with some ‘good’ \(\mathbf{w}\)
- getting ‘good’ \(\mathbf{w}\): learned from data
Feature Selection by Importance

idea: if possible to calculate

\[ \text{importance}(i) \text{ for } i = 1, 2, \ldots, d \]

then can select \( i_1, i_2, \ldots, i_{d'} \) of top-\( d' \) importance

importance by linear model

\[ \text{score} = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{d} w_i x_i \]

- intuitive estimate: \( \text{importance}(i) = |w_i| \) with some 'good' \( \mathbf{w} \)
- getting 'good' \( \mathbf{w} \): learned from data
- non-linear models? often much harder
Feature Selection by Importance

idea: if possible to calculate

\[ \text{importance}(i) \text{ for } i = 1, 2, \ldots, d \]

then can select \( i_1, i_2, \ldots, i_{d'} \) of top-\( d' \) importance

importance by linear model

\[
\text{score} = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{d} w_i x_i
\]

- intuitive estimate: \( \text{importance}(i) = |w_i| \) with some ‘good’ \( \mathbf{w} \)
- getting ‘good’ \( \mathbf{w} \): learned from data
- non-linear models? often much harder

next: ‘easy’ feature selection in RF
Feature Importance by Permutation Test

idea: random test

- if feature $i$ needed, 'random' values of $x_n$, $i$ degrades performance

- which random values?
  - uniform, Gaussian, ...
  - bootstrap, permutation ($\{x_n, i\}_n=1$):
    $$P(x_i) \approx \text{remained}$$

- permutation test: a general statistical tool for arbitrary non-linear models like RF
Feature Importance by Permutation Test

idea: random test
—if feature $i$ needed, 'random' values of $x_n,i$ degrades performance
Feature Importance by Permutation Test

**idea: random test**
—if feature $i$ needed, ‘random’ values of $x_{n,i}$ degrades performance

- which *random values*?
Feature Importance by Permutation Test

idea: random test
—if feature $i$ needed, ‘random’ values of $x_{n,i}$ degrades performance

• which random values?
  • uniform, Gaussian, . . .:
Feature Importance by Permutation Test

idea: random test
— if feature $i$ needed, ‘random’ values of $x_{n,i}$ degrades performance

- which random values?
  - uniform, Gaussian, ...: $P(x_i)$ changed
Random Forest

Feature Selection

Feature Importance by Permutation Test

idea: random test
— if feature $i$ needed, ‘random’ values of $x_{n,i}$ degrades performance

- which random values?
  - uniform, Gaussian, ...: $P(x_i)$ changed
  - bootstrap, permutation (of $\{x_{n,i}\}_{n=1}^N$): $P(x_i)$ approximately remained
Feature Importance by Permutation Test

idea: random test
—if feature \( i \) needed, ‘random’ values of \( x_{n,i} \) degrades performance

- which random values?
  - uniform, Gaussian, \ldots: \( P(x_i) \) changed
  - bootstrap, **permutation** (of \( \{x_{n,i}\}_{n=1}^{N} \)): \( P(x_i) \) approximately remained

- **permutation** test:

\[
\text{importance}(i) = \text{performance}(D) - \text{performance}(D^{(p)})
\]

with \( D^{(p)} \) is \( D \) with \( \{x_{n,i}\} \) replaced by permuted \( \{x_{n,i}\}_{n=1}^{N} \)
Feature Importance by Permutation Test

idea: random test
—if feature $i$ needed, ‘random’ values of $x_{n,i}$ degrades performance

- which random values?
  - uniform, Gaussian, . . .: $P(x_i)$ changed
  - bootstrap, permutation (of $\{x_{n,i}\}_{n=1}^N$): $P(x_i)$ approximately remained

- permutation test:

  $$\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}^{(p)})$$

  with $\mathcal{D}^{(p)}$ is $\mathcal{D}$ with $\{x_{n,i}\}$ replaced by permuted $\{x_{n,i}\}_{n=1}^N$

**permutation** test: a general statistical tool for arbitrary non-linear models like RF
Feature Importance in Original Random Forest

**permutation** test:

\[
\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}^{(p)})
\]

with \(\mathcal{D}^{(p)}\) is \(\mathcal{D}\) with \(\{x_{n,i}\}\) replaced by permuted \(\{x_{n,i}\}_{n=1}^{N}\)
Feature Importance in Original Random Forest

**permutation** test:

\[
\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}^{(p)})
\]

with \( \mathcal{D}^{(p)} \) is \( \mathcal{D} \) with \( \{x_{n,i}\} \) replaced by permuted \( \{x_{n,i}\}_{n=1}^{N} \)

- \( \text{performance}(\mathcal{D}^{(p)}) \): needs re-training and validation in general
Feature Importance in Original Random Forest

**permutation** test:

$$\text{importance}(i) = \text{performance}(D) - \text{performance}(D^{(p)})$$

with $D^{(p)}$ is $D$ with $\{x_{n,i}\}$ replaced by permuted $\{x_{n,i}\}_{n=1}^N$

- **performance($D^{(p)}$)**: needs re-training and validation in general
- ‘escaping’ validation?
Feature Importance in Original Random Forest

**permutation** test:

\[
\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}(p))
\]

with \( \mathcal{D}(p) \) is \( \mathcal{D} \) with \( \{x_{n,i}\} \) replaced by permuted \( \{x_{n,i}\}_{n=1}^{N} \)

- \( \text{performance}(\mathcal{D}(p)) \): needs re-training and validation in general
- ‘escaping’ validation? OOB in RF
Feature Importance in Original Random Forest

**permutation** test:

\[
\text{importance}(i) = \text{performance}(D) - \text{performance}(D^{(p)})
\]

with \(D^{(p)}\) is \(D\) with \(\{x_{n,i}\}\) replaced by permuted \(\{x_{n,i}\}_{n=1}^{N}\)

- **performance**\((D^{(p)})\): needs re-training and validation in general
- **‘escaping’ validation? OOB** in RF
- original RF solution: \(\text{importance}(i) = E_{\text{ooob}}(G) - E_{\text{ooob}}^{(p)}(G)\), where \(E_{\text{ooob}}^{(p)}\) comes from replacing each request of \(x_{n,i}\) by a permuted OOB value
**Feature Importance in Original Random Forest**

**permutation** test:

\[
\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}^{(p)})
\]

with \(\mathcal{D}^{(p)}\) is \(\mathcal{D}\) with \(\{x_{n,i}\}\) replaced by permuted \(\{x_{n,i}\}^{N}_{n=1}\)

- \(\text{performance}(\mathcal{D}^{(p)})\): needs re-training and validation in general
- ‘escaping’ validation? OOB in RF
- original RF solution: \(\text{importance}(i) = E_{oob}(G) - E_{oob}^{(p)}(G)\), where \(E_{oob}^{(p)}\) comes from replacing each request of \(x_{n,i}\) by a permuted OOB value

**RF feature selection** via **permutation** + OOB: often efficient and promising in practice
For RF, if the 1126-th feature within the data set is a constant 5566, what would importance(i) be?

1. 0
2. 1
3. 1126
4. 5566
For RF, if the 1126-th feature within the data set is a constant 5566, what would \text{importance}(i) be?

\begin{itemize}
  \item 0
  \item 1
  \item 1126
  \item 5566
\end{itemize}

\textbf{Reference Answer: 1}

When a feature is a constant, permutation does not change its value. Then, \( E_{oob}(G) \) and \( E_{oob}^{(p)}(G) \) are the same, and thus \( \text{importance}(i) = 0 \).
A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees
A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees
A Simple Data Set

$g_{\text{C\&RT}}$ with random combination

$g_t \ (N' = N/2)$

$G$ with first $t$ trees

Hsuan-Tien Lin (NTU CSIE)
A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees

$G$ with first $t$ trees
A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees

$t = 300$
A Simple Data Set

\( g_{C&RT} \) with random combination

\( g_t (N' = N/2) \)

\( G \) with first \( t \) trees

\( t = 400 \)
A Simple Data Set

$g_{C&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees
Random Forest

Random Forest in Action

A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees

Hsuan-Tien Lin (NTU CSIE)
A Simple Data Set

\( g_{C\&RT} \) with random combination

\( g_t (N' = N/2) \)

\( G \) with first \( t \) trees
A Simple Data Set

\( g_{C\&RT} \)  
with random combination

\( g_t \ (N' = N/2) \)

\( G \) with first \( t \) trees

Hsuan-Tien Lin (NTU CSIE)
A Simple Data Set

\[ g_{C\&RT} \]
with random combination

\[ g_t \ (N' = N/2) \]

\[ G \text{ with first } t \text{ trees} \]
A Simple Data Set

$g_{C\&RT}$ with random combination

$g_t (N' = N/2)$

$G$ with first $t$ trees

‘smooth’ and large-margin-like boundary with many trees
A Complicated Data Set

\[ g_t (N' = N/2) \]

\[ G \text{ with first } t \text{ trees} \]
Random Forest

Random Forest in Action

A Complicated Data Set

\( g_t \ (N' = N/2) \)

\( G \) with first \( t \) trees

Hsuan-Tien Lin (NTU CSIE)
Random Forest

Random Forest in Action

A Complicated Data Set

\( g_t \ (N' = N/2) \)  

\( G \) with first \( t \) trees

Hsuan-Tien Lin (NTU CSIE)  

Machine Learning Techniques  

18/22
Random Forest

Random Forest in Action

A Complicated Data Set

g_t (N' = N/2)

G with first \( t \) trees

Hsuan-Tien Lin (NTU CSIE)
A Complicated Data Set

\[ g_t \left( N' = \frac{N}{2} \right) \]

\[ G \text{ with first } t \text{ trees} \]

‘easy yet robust’ nonlinear model
A Complicated and Noisy Data Set

\[ g_t (N' = N/2) \]

\[ G \text{ with first } t \text{ trees} \]
A Complicated and Noisy Data Set

\[ g_t (N' = N/2) \]

\[ G \text{ with first } t \text{ trees} \]
A Complicated and Noisy Data Set

$g_t \ (N' = N/2)$

$G$ with first $t$ trees
A Complicated and Noisy Data Set

\[ g_t \ (N' = N/2) \]

\[ G \] with first \( t \) trees
A Complicated and Noisy Data Set

\( g_t (N' = N/2) \)

\( G \) with first \( t \) trees

noise corrected by voting
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\bar{g} = \lim_{T \to \infty} G$
How Many Trees Needed?

- almost every theory: the more, **the ‘better’**
  - assuming good \( \bar{g} = \lim_{T \to \infty} G \)

Our NTU Experience

- KDDCup 2013 Track 1
  - predicting author-paper relation
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\bar{g} = \lim_{T \to \infty} G$

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)):
  predicting author-paper relation
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\bar{g} = \lim_{T \to \infty} G$

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)):
predicting author-paper relation
- $E_{val}$ of thousands of trees: [0.015, 0.019] depending on seed;
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\bar{g} = \lim_{T \to \infty} G$

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)):
predicting author-paper relation
- $E_{\text{val}}$ of thousands of trees: $[0.015, 0.019]$ depending on seed;
  $E_{\text{out}}$ of top 20 teams: $[0.014, 0.019]$
How Many Trees Needed?

almost every theory: the more, **the ‘better’**
assuming good $\bar{g} = \lim_{T \to \infty} G$

Our NTU Experience

- **KDDCup 2013 Track 1** *(yes, NTU is world champion again! :-)*)
  predicting author-paper relation
- $E_{\text{val}}$ of **thousands** of trees: [0.015, 0.019] depending on seed;
  $E_{\text{out}}$ of top 20 teams: [0.014, 0.019]
- decision: take **12000 trees**
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good \( \bar{g} = \lim_{T \to \infty} G \)

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)]: predicting author-paper relation
- \( E_{val} \) of thousands of trees: [0.015, 0.019] depending on seed;
  \( E_{out} \) of top 20 teams: [0.014, 0.019]
- decision: take 12000 trees with seed 1
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\tilde{g} = \lim_{T \to \infty} G$

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)): predicting author-paper relation
- $E_{\text{val}}$ of thousands of trees: $[0.015, 0.019]$ depending on seed; $E_{\text{out}}$ of top 20 teams: $[0.014, 0.019]$
- decision: take 12000 trees with seed 1

cons of RF: may need lots of trees if the whole random process too unstable
How Many Trees Needed?

almost every theory: the more, the ‘better’
assuming good $\bar{g} = \lim_{T \to \infty} G$

Our NTU Experience

- KDDCup 2013 Track 1 (yes, NTU is world champion again! :-)):
  predicting author-paper relation
- $E_{\text{val}}$ of thousands of trees: [0.015, 0.019] depending on seed;
  $E_{\text{out}}$ of top 20 teams: [0.014, 0.019]
- decision: take 12000 trees with seed 1

cons of RF: may need lots of trees if the whole random process too unstable
—should double-check stability of $G$
  to ensure enough trees
Which of the following is **not** the best use of Random Forest?

1. train each tree with bootstrapped data
2. use $E_{oob}$ to validate the performance
3. conduct feature selection with permutation test
4. fix the number of trees, $T$, to the lucky number 1126
Which of the following is **not** the best use of Random Forest?

1. train each tree with bootstrapped data
2. use $E_{oob}$ to validate the performance
3. conduct feature selection with permutation test
4. fix the number of trees, $T$, to the lucky number 1126

Reference Answer: 4

A good value of $T$ can depend on the nature of the data and the stability of the whole random process.
Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

**Lecture 10: Random Forest**
- Random Forest Algorithm
  - bag of trees on randomly projected subspaces
- Out-Of-Bag Estimate
  - self-validation with OOB examples
- Feature Selection
  - permutation test for feature importance
- Random Forest in Action
  - ‘smooth’ boundary with many trees

- next: boosted decision trees beyond classification
3. Distilling Implicit Features: Extraction Models