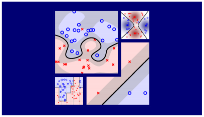
Machine Learning Techniques
(機器學習技法)



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
- Theory versus Practice

3. Distilling Implicit Features: Extraction Models

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

—reduces variance by voting/averaging

Decision Tree

function DTree($\mathcal{D}$)
if termination return base g_t
else

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

—large variance especially if fully-grown

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

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

- 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 $\mathcal{D}$

other possibility for diversity:

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

- sampled index $i_1, i_2, \ldots, i_{d'}$: $\mathbf{\Phi}(\mathbf{x}) = (x_{i_1}, x_{i_2}, \ldots, x_{i_{d'}})$
- $\mathcal{Z} \in \mathbb{R}^{d'}$: a **random subspace** of $\mathcal{X} \in \mathbb{R}^{d}$
- often $d' \ll d$, efficient when d large
  —can be generally used for other learning models
- original RF **re-sample** new subspace **for each** $b(\mathbf{x})$ in C&RT

RF = bagging + random-subspace C&RT

Diversifying by Feature Expansion

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

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

- low-dimensional random projection (combination) with $\mathbf{v}$:

$$\phi_i(\mathbf{x}) = \sum_{j=1}^{d''} v_j x_j$$

- includes **random subspace** as **special case**: $d'' = 1$ and $v_1 = 1$
- original RF consider d' random **combinations for each** $b(\mathbf{x})$ in C&RT

RF = bagging + random-**combination** C&RT
—randomness everywhere!

Fun Time

Within RF that contains random-combination C&RT trees, which of the following hypothesis is equivalent to each branching function $b(\mathbf{x})$ within the tree?

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

Fun Time

Within RF that contains random-combination C&RT trees, which of the following hypothesis is equivalent to each branching function $b(\mathbf{x})$ within the tree?

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

Reference Answer: 3

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

Bagging Revisited

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

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_1$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	$\star$		$\star$

$\star$: not used for obtaining g_t
—called **out-of-bag (OOB) examples**

Number of OOB Examples

OOB (in $\star$) $\Longleftrightarrow$ not sampled after N' drawings

if $N' = N$

- probability for $(\mathbf{x}_n, y_n)$ 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 $g_t \approx \frac{1}{e} N$

OOB versus Validation

OOB

	g_1	g_2	g_3	$\cdots$	g_T
$(\mathbf{x}_1, y_1)$	$\tilde{\mathcal{D}}_1$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_2, y_2)$	$\star$	$\star$	$\tilde{\mathcal{D}}_3$		$\tilde{\mathcal{D}}_T$
$(\mathbf{x}_3, y_3)$	$\star$	$\tilde{\mathcal{D}}_1$	$\star$		$\tilde{\mathcal{D}}_T$
$\cdots$					
$(\mathbf{x}_N, y_N)$	$\tilde{\mathcal{D}}_1$	$\tilde{\mathcal{D}}_2$	$\star$		$\star$

Validation

g_1^-	g_2^-	$\cdots$	g_M^-
$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{train}}$		$\mathcal{D}_{\text{train}}$
$\mathcal{D}_{\text{val}}$	$\mathcal{D}_{\text{val}}$		$\mathcal{D}_{\text{val}}$
$\mathcal{D}_{\text{val}}$	$\mathcal{D}_{\text{val}}$		$\mathcal{D}_{\text{val}}$
$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{train}}$		$\mathcal{D}_{\text{train}}$

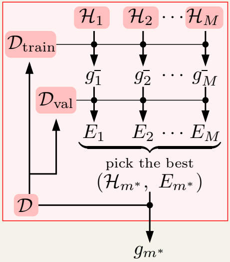
- $\star$ like $\mathcal{D}_{\text{val}}$: 'enough' random examples unused during training
- use $\star$ to validate g_t? easy, but **rarely needed**
- use $\star$ to validate G? $E_{\text{oob}}(G) = \frac{1}{N}\sum_{n=1}^{N} \text{err}(y_n, G_n^-(\mathbf{x}_n))$, with G_n^- contains only trees that $\mathbf{x}_n$ is OOB of

E_{oob}: self-validation of bagging/RF

Model Selection by OOB Error

Previously: by Best E_{val}

$$\begin{aligned} g_{m^*} &= \mathcal{A}_{m^*}(\mathcal{D}) \\ m^* &= \operatorname*{argmin}_{1 \le m \le M} E_m \\ E_m &= E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}})) \end{aligned}$$



RF: by Best E_{oob}

$$\begin{aligned} G_{m^*} &= \text{RF}_{m^*}(\mathcal{D}) \\ m^* &= \operatorname*{argmin}_{1 \le m \le M} E_m \\ E_m &= E_{\text{oob}}(\text{RF}_m(\mathcal{D})) \end{aligned}$$

- use E_{oob} for self-validation
- no re-training needed

E_{oob} often **accurate** in practice

Fun Time

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

Fun Time

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

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

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

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, ...: $P(x_i)$ 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 $\mathcal{D}_p$ containing 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 $\mathcal{D}_p$ containing permuted $\{x_{n,i}\}_{n=1}^{N}$

- calculating performance needs re-training and validating on each $\mathcal{D}_p$ in general
- how to 'escape' validation? OOB in RF
- original RF solution:

$$\text{importance}(i) = E_{\text{oob}}(G, \mathcal{D}) - E_{\text{oob}}(G, \mathcal{D}_p)$$

with $\mathcal{D}_p$ 'dynamically' containing permuted $\{x_{n,i} : n \text{ OOB}\}$ for g_t

original RF solution often efficient and promising in practice

Fun Time

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

Fun Time

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

Reference Answer: 1

When a feature is a constant, permutation does not change its value. Then, performance($G, \mathcal{D}$) and performance($G, \mathcal{D}_p$) are the same, and thus importance(i) $= 0$.

Theory: Does Diversity Help?

strength-correlation decomposition (classification):

$$\lim_{T\to\infty} E_{\text{out}}(G) \leq \rho \cdot \left(\frac{1-s^2}{s^2}\right)$$

- strength: average voting margin within G
- correlation: similarity between g_t
- similar for regression (bias-variance decomposition)

RF good if diverse and strong

Practice: How Many Trees Needed?

theory: the more, the 'better'

- NTU KDDCup 2013 Track 1: predicting author-paper relation
- $1 - E_{\text{val}}$ of thousands of trees: $[0.981, 0.985]$ depending on seed;
  $1 - E_{\text{out}}$ of top 20 teams: $[0.98130, 0.98554]$
- decision: take 12000 trees with seed 1

cons of RF: may need lots of trees **if random process too unstable**

Fun Time

The strength s is a value between $[0, 1]$. For a fixed ρ, which value of s results in the minimum upper bound for the limiting $E_{\text{out}}(G)$?

1. 1.0
2. 0.5
3. 0.0
4. none of the other choices

Fun Time

The strength s is a value between $[0, 1]$. For a fixed ρ, which value of s results in the minimum upper bound for the limiting $E_{\text{out}}(G)$?

1. 1.0
2. 0.5
3. 0.0
4. none of the other choices

Reference Answer: 1

Too simple, huh? :-)

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
- Theory versus Practice
  more or not? that's the question!

- **next: boosted decision trees beyond classification**

3. Distilling Implicit Features: Extraction Models