Bagging (bootstrap aggregating)

TAs : Louis: Series 8, 9, 10
Claude: " 1, 5, 6, 7
Peter: 2, 3, 4

- Data: $(X_1, Y_1), \dots, (X_n, Y_n) \sim P$ $\quad X_i \in \mathbb{R}^p, \; Y_i \in \mathbb{R}$
- Base procedure: $\hat{g} : \mathbb{R}^p \to \mathbb{R}$ eg. regression tree
  - Take B bootstrap samples $(X_1^{*b}, Y_1^{*b}), \dots, (X_n^{*b}, Y_n^{*b}), \; b = 1, \dots, B$ — b^* bootstrap samples
  - Then, $\hat{g}^{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{g}^{*b}(x)$, where $\hat{g}^*(\cdot)$ is the estimate based on the b^{th} bootstrap sample.
  - Note that $\hat{g}^{bag}(\cdot) \to E^*[g^*(\cdot)]$ as $B \to \infty \approx E_P[g^*(\cdot)]$

((Ref Slides Fig 8.11. (bagging)) If $\exists$ learners (even mediocre ones) then agg. performance improves substantially)

- Idea: Mimic sampling from the distribution P, and then aggregate estimates
  - ! * Bagging does not do anything for linear predictors; linear predictors stay the same under bagging.
- Simplest example: $\hat{g}(x) = \bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$ for X
  - Then $E^*(g^*(x)) = E^*[\bar{Y}^*] = E\left[\frac{1}{n} \sum_{i=1}^{n} Y_i^*\right] = \frac{1}{n} \sum_{i=1}^{n} E^*[Y_i^*] \overset{iid}{=}$

$$\overset{iid}{=} \frac{1}{n} \sum_{i=1}^{n} E^*[Y_1^*] = E^*[Y_1^*] \quad \text{and given } Y_1^* = \begin{cases} Y_1 & \text{w.p. } 1/n \\ \vdots & \vdots \\ Y_n & \vdots \end{cases} \quad \text{SO}$$

$$= \sum_{i=1}^{n} Y_i \cdot \frac{1}{n} = \bar{Y}$$

- Bagging should be used for non-linear estimators! ex. trees.

(Key point: Bagging - decrease Variance without increasing bias by aggregating the estimators, predicted outcome probabilities, ...

- Bagging for Classification: Given data $(X_1, Y_1), \dots, (X_n, Y_n)$, $X_i \in \mathbb{R}^p$, $Y_i \in \{1, \dots, K\}$ categories
  - Base procedure: $\hat{g}(\cdot) : \mathbb{R}^p \to \{1, \dots, K\}$ (doesn't make sense to average the category #s)

Methods. (1) Majority Vote: $\hat{g}^{bag}(x) = \underset{k=1,\dots,K}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbb{1}\{\hat{g}^*(x) = k\}$ ← i.e. # of times you predicted class k

- ex. a thought experiment: ~~Given input X and we know~~ classes $\{1, 2\}$
  - For some given input X we have B independent classifiers $\hat{f}^b(x)$, $b = 1, \dots, B$ and each classifier has a misclassification rate of 0.4.
    - Note that this is a thought experiment; the classifiers are not independent for bagging because we take bootstrap samples from the same training data.
  - Assume wlog that the true class is 1. This implies that $P(\hat{g}^b(x) = 1) = 0.6$, $P(\hat{g}^b(x) = 2) = 0.4$
  - Now we form a bagged classifier $\hat{g}^{bag}(x) = \underset{k=1,2}{\operatorname{argmax}} \sum_{b=1}^{B} \mathbb{1}\{\hat{g}^b(x) = k\}$

↳ what is the misclassification rate of $\hat{g}^{bag}(x)$? ie. What is $P(\hat{g}^{bag}(x) = 2)$?

Computational Statistics
Lecture, cont'd

Fri 07 June 18

Bagging for Classification, cont'd.

(ex. continued) (What is the misclassification rate of $\hat{g}^{bag}(x)$? ie.:

- $P(\hat{g}^{bag}(x) = 2) = P\left(\underbrace{\sum_{b=1}^{B} \mathbb{1}\{\hat{g}^b(x) = 2\}}_{W} > B/2\right)$

Let $W = \sum_{b=1}^{B} \mathbb{1}\{\hat{g}^b(x) = 2\}$ $= P(W > B/2)$

$W \sim \text{Binom}(B, 0.6)$
$E[W] = 0.6B$
$\text{Var}[W] = B \cdot 0.6 \cdot 0.4 = 0.24B$
so then $W \approx \mathcal{N}(0.6B, 0.24B)$
$\approx$ (Gaussian approximation)

$\rightarrow = P\left(\frac{W - 0.6B}{\sqrt{0.24B}} > \frac{B/2 - 0.6B}{\sqrt{0.24B}}\right)$

$\approx P\left(Z > \underbrace{-\frac{0.1B}{\sqrt{0.24B}}}_{\to -\infty}\right) \to 1$ as $B \to \infty$

- So, we get a perfectly bad classifier.

- Bagging a good classifier can improve performance, but bagging a bad classifier can degrade performance

(Suppose we want to know the class probabilities)

↳ Voting probabilities $\frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\{\hat{g}^b(x) = k\}$, $k = 1, \ldots, K$ are not good estimates of the estimates of the class probabilities $P(Y = k | X = x)$

- ex. Suppose that $P(Y = 1 | X = x) = 0.75$ and that all $\hat{g}^{*b}(x) = 1 \quad \forall b = 1, \ldots, B$
  Then voting probability $\frac{1}{B}\sum \mathbb{1}\{g^{*b}(x) = 1\} = 1$

- Each tree gives predicted class probabilities $\hat{p}_k^{*b}(x)$, where

  $\hat{p}_k^{*b}(x)$:= proportion of observations with class k among all observations in the leaf node of tree g^{*b} that contains x, $\forall$ classes $(k = 1, \ldots, K)$

- We can bag the predicted class probabilities $\hat{p}_k^{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{p}_k^{*b}(x)$

- The final bagged classifier chooses the class with the highest $\hat{p}_k^{bag}(x)$
  ie. $\hat{g}^{bag}(x) = \underset{k=1,\ldots,K}{\text{argmax}}\ \hat{p}_k^{bag}(x)$

- Advantages of bagging: We get estimated class probabilities, sometimes better prediction performance

  ↳ (Slides: Figure 8.10) - consensus majority vote vs. bagged (from this figure) ↳ get an idea about the uncertainties

- (Ref Rcode 13)
  - line 130: can use randomForest for regular bagging by setting mtry (# of variables to consider for each split)
  - line 142: default uses 1/3 of variables @ each split

Random Forests

- Actual random forests: trees will be more independent (than from bagging) so you get better performance
- Pros / Cons of random forests: better classification and regression performance, but harder to interpret the model.
- "Out of bag" (OOB) (~~from~~ samples not bagged in training) - provides error estimates so we don't need to do X Valid'n. train ~~#~~OOB values are comparable to test OOB values.
  - ~~OOB~~ OOB error estimates (MSE) ≢ training MSE !
- Comparing Bagged trees and random forests:

	Trees	Bagged trees	Random Forests
Interpretation	+	−	− (we however can see/get variable importance.
Computation	+	−	~
Performance	−	+	++
OOB error estimates	(need XV)	+	+
Overfitting	Yes (need XV)	OK	OK

→ Variable Importance: Metric for # times used to split, ~~[illegible]~~ a variable was used in distinguishing data //

Lecture

Fri. 01 June 2018

(Ref Rcode14): 3 training sets sampled from the data (~ line 20)

MSE	train1	train2	train3
Bagging OOB	13.7	15.7	11.8
Bagging testset	15.8	11.0	15.4
Random forest OOB	12.4	14.7	12.8
Random forest test set	16.8	12.5	14.1

- Note that training MSE ≠ OOB MSE, and that OOB is comparable to test MSE.
- Also note that variability ~~is a~~ MSE (as a random var?)
  - ie. OOB is neither optimistic nor pessimistic (generally).
  - Train and Test MSEs are in similar ranges

(line 80) - randomForest(·): maxnotes = # arg ≡ max size (~~# of datatab~~ depth of the tree. (ie (# of data in each node))

(line 88) df: eg. 0.044··· | 0.955··· is proportion predicted class 1 | class 2

min nodesize (# at least, in each node)

- Deep trees: bias is less of an issue; the
  - larger the node (⇝ shallower tree) the more problematic
  - deep tree results: more scatter/variability but less bias. (mean closer to true value (2/3))

(line 106) ranger (library). Using probability method.

└ Shallow trees with prob. method (preserves) low bias while also lowering variability (lower var than in ~~high~~ deep trees (with prob. method)

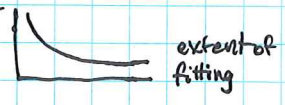
★ Random Forests (with Probability Method) on Exam!

⇨

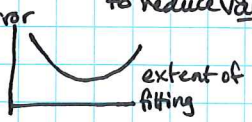
Computational Statistics
Lecture, cont'd

Fri 01 June 18

- **Bagging** – Start with a deep tree; flexible estimate
  ↪ low bias, high variance
  – bootstrap aggregating reduces variance
  – as $B \to \infty$, we don't have to worry about overfitting (since we are combining things to reduce variance)



- **Boosting** – Start with a simple estimator; ("tree stump") with 1 split
  ↪ high bias, low variance
  – combine estimates to reduce bias
  – as $B \to \infty$, overfitting tends to occur.



(Ref Algo 8.2 from ISLR - in slides)

- line (2b): λ is usually a small value that is fixed (usually)
  (This is a "slow learner"
- choose B by XValid'n ↪ from big B you can do XValid'n from small B subsets.
- reweight (increase weight of) misclassified outcomes in XValid'n ~~for~~ so training improves performance more, ie "pay more attention" to them.

(Ref Rcode14) gbm library for boosting (line 123)

- gbm(.) arg n.tree ≡ B
  interaction.depth ≡ # of variables from splits that are allowed to interact
- line 130: partial dependence plot
- Adjust params (λ, # of trees, ...) to improve mse. Can use XValid'n to choose both λ and # of trees, but usually λ is fixed and XValid'n is done for B.

//

Section

Fri 01 June 18

(Series question is from ISLR). - see online section notes.

- check quality of model by XValid'n.
- (Assume that we can randomly shuffle the data
- "oof" := out-of-fold : vector of whole data set and we fit on all data \ {current fold} → vector of prediction
  (gefa)
- What is the probability threshold of decision? Use not 0.5 (fixed value) but the mean of oof.

- **Boosting** : fit tree based on ~~previous~~ the residues of previous trees

- Importance (plot): of variables – in splits, how many times was a variable used to split, distinguish the data?

- **Bagging vs. RF**.
  - First question: using trees, "How ~~do~~ we deal with overfitting?"
    → combine / aggregate trees, ie bagging
  - Can use randomForests library (with bagging as a special case)
    (so you don't have to use a specific bagging library.
    (Bagging – instead of resampling features @ each step, for take all the features (for each step).

//

(Last class!)