Bagging and Random Forests
We briefly review a technique called bootstrap on which bagging and random forests are based.

**Bootstrap**

*Bootstrap* (or *resampling*) is a technique for improving the quality of estimators.

Resampling = sampling from the empirical distribution

**Application to ensemble methods**

- We will use resampling to generate weak learners for classification.
- We discuss two classifiers which use resampling: Bagging and random forests.
- Before we do so, we consider the traditional application of Bootstrap, namely improving estimators.
Given

- A sample \( \tilde{x}_1, \ldots, \tilde{x}_n \).
- An estimator \( \hat{S} \) for a statistic \( S \).

Bootstrap algorithm

1. Generate \( B \) bootstrap samples \( B_1, \ldots, B_B \). Each bootstrap sample is obtained by sampling \( n \) times with replacement from the sample data. (Note: Data points can appear multiple times in any \( B_b \).)
2. Evaluate the estimator on each bootstrap sample:
   \[
   \hat{S}_b := \hat{S}(B_b)
   \]
   (That is: We estimate \( S \) pretending that \( B_b \) is the data.)
3. Compute the bootstrap estimate of \( S \) by averaging over all bootstrap samples:
   \[
   \hat{S}_{BS} := \frac{1}{B} \sum_{b=1}^{B} \hat{S}_b
   \]
Recall: Plug-in estimators for mean and variance

\[ \hat{\mu} := \frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i \]
\[ \hat{\sigma}^2 := \frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i - \hat{\mu})^2 \]

Bootstrap Variance Estimate

1. For \( b = 1, \ldots, B \), generate a bootstrap sample \( B_b \). In detail:
   For \( i = 1, \ldots, n \):
   - Sample an index \( j \in \{1, \ldots, n\} \).
   - Set \( \tilde{x}_i^{(b)} := \tilde{x}_j \) and add it to \( B_b \).
2. For each \( b \), compute mean and variance estimates:
   \[ \hat{\mu}_b := \frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i^{(b)} \]
   \[ \hat{\sigma}_b^2 := \frac{1}{n} \sum_{i=1}^{n} (\tilde{x}_i^{(b)} - \hat{\mu}_b)^2 \]
3. Compute the bootstrap estimate:
   \[ \hat{\sigma}_{BS}^2 := \frac{1}{B} \sum_{b=1}^{B} \hat{\sigma}_b^2 \]
Sample \{\tilde{x}_1, ..., \tilde{x}_n\}, bootstrap resamples \mathcal{B}_1, ..., \mathcal{B}_B.

In how many sets does a given \(x_i\) occur?
Probability for \(x_i\) not to occur in \(n\) draws:

\[ \Pr\{\tilde{x}_i \notin \mathcal{B}_b\} = (1 - \frac{1}{n})^n \]

For large \(n\):

\[ \lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679 \]

- Asymptotically, any \(\tilde{x}_i\) will appear in \(~63\%\) of the bootstrap resamples.
- Multiple occurrences possible.

How often is \(\tilde{x}_i\) expected to occur?
The \textit{expected} number of occurrences of each \(\tilde{x}_i\) is \(B\).

Bootstrap estimate averages over reshuffled samples.
Estimate variance of estimators

- Since estimator \( \hat{S} \) depends on (random) data, it is a random variable.
- The more this variable scatters, the less we can trust our estimate.
- If scatter is high, we can expect the values \( \hat{S}_b \) to scatter as well.
- In previous example, this means: Estimating the variance of the variance estimator.

Variance reduction

- Averaging over the individual bootstrap samples can reduce the variance in \( \hat{S} \).
- In other words: \( \hat{S}_{BS} \) typically has lower variance than \( \hat{S} \).
- This is the property we will use for classification in the following.

As alternative to cross validation
To estimate prediction error of classifier:

- For each \( b \), train on \( B_b \), estimate risk on points not in \( B_b \).
- Average risk estimates over bootstrap samples.
BAGGING

Idea

• Recall Boosting: Weak learners are deterministic, but selected to exhibit high variance.
• Strategy now: Randomly distort data set by resampling.
• Train weak learners on resampled training sets.
• Resulting algorithm: Bagging (= Bootstrap aggregation)
For Bagging with $K$ classes, we represent class labels as vectors:

$$x_i \text{ in class } k \quad \text{as} \quad y_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow k\text{th entry}$$

This way, we can average together multiple class labels:

$$\frac{1}{n} (y_1 + \ldots + y_n) = \begin{pmatrix} p_1 \\ \vdots \\ p_k \\ \vdots \\ p_K \end{pmatrix}$$

We can interpret $p_k$ as the probability that one of the $n$ points is in class $k$. 
Bagging: Algorithm

Training
For $b = 1, \ldots, B$:
1. Draw a bootstrap sample $B_b$ of size $n$ from training data.
2. Train a classifier $f_b$ on $B_b$.

Classification

• Compute

$$f_{\text{avg}}(x) := \frac{1}{B} \sum_{b=1}^{B} f_b(x)$$

This is a vector of the form $f_{\text{avg}}(x) = (p_1(x), \ldots, p_k(x))$.

• The Bagging classifier is given by

$$f_{\text{Bagging}}(x) := \arg \max_k \{p_1(x), \ldots, p_k(x)\},$$

i.e. we predict the class label which most weak learners have voted for.
**Example: Bagging Trees**

- Two classes, each with Gaussian distribution in $\mathbb{R}^5$.
- Note the variance between bootstrapped trees.
Bagging vs. Boosting

• Bagging works particularly well for trees, since trees have high variance.
• Boosting typically outperforms bagging with trees.
• The main culprit is usually dependence: Boosting is better at reducing correlation between the trees than bagging is.

Random Forests
Modification of bagging with trees designed to further reduce correlation.

• Tree training optimizes each split over all dimensions.
• Random forests choose a different subset of dimensions at each split.
• Optimal split is chosen within the subset.
• The subset is chosen at random out of all dimensions \( \{1, \ldots, d\} \).
Training

Input parameter: $m$ (positive integer with $m < d$)

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $B_b$ of size $n$ from training data.
2. Train a tree classifier $f_b$ on $B_b$, where each split is computed as follows:
   - Select $m$ axes in $\mathbb{R}_d$ at random.
   - Find the best split $(j^*, t^*)$ on this subset of dimensions.
   - Split current node along axis $j^*$ at $t^*$.

Classification

Exactly as for bagging: Classify by majority vote among the $B$ trees. More precisely:

- Compute $f_{\text{avg}}(x) := (p_1(x), \ldots, p_k(x)) := \frac{1}{B} \sum_{b=1}^{B} f_b(x)$
- The Random Forest classification rule is
  $$f_{\text{Bagging}}(x) := \arg \max_k \{ p_1(x), \ldots, p_k(x) \}$$
Remarks

- Recommended value for $m$ is $m = \lfloor \sqrt{d} \rfloor$ or smaller.
- RF typically achieve similar results as boosting. Implemented in most packages, often as standard classifier.

Example: Synthetic Data

- This is the RF classification boundary on the synthetic data we have already seen a few times.
- Note the bias towards axis-parallel alignment.
APPLICATION: CANCER DIAGNOSIS
Kidney cancer diagnosis: Clinical procedure

- Take tissue sample from patient’s kidney
- Preprocess sample and photograph under microscope
- A pathologist looks at the image and diagnosis patient on scale from healthy to advanced stage cancer

Task

- Empirically, the results vary significantly between pathologists
- The objective is to build a classifier that produces a diagnosis using the same scale as the pathologist, hopefully with more stable results.
DATA ANALYSIS PIPELINE

Experiment

Preprocessing

Raw data (measurements)

Feature extraction

Working data

Mark patterns

Split

Training data (patterns marked)

Training (calibration)

Trained model

Test data (patterns marked)

Apply on test data

Error estimate

Peter Orbanz · Applied Data Mining
Preprocessing

1. Tissue sample is “stained” with marking fluid
2. Thin slice is cut and placed on microscope slide
3. Sample is photographed under microscope
4. Tumor cells absorb more marker fluid and tend to be darker
A pathologist uses a software with a graphical user interface to (1) mark the locations of nuclei and (2) label nuclei as healthy/cancerous.
Five experts label the same set of nuclei (180 in total)
For each data point (nucleus), count the number of votes (0, . . . , 5) in favor of “tumor”
The diagram above is a histogram of the vote counts for the 180 data points
All five experts agree if the count is 0 (all say healthy) or 5 (all say tumor)
(The small red/green bars are the vote proportions, so they encode the same information as the numbers at the bottom.)
Comparison Between Experts (2)

Results for 14 pathologists labeling four patients.

- Each box in the boxplot represents one patient.
- Box plot: The line in the middle of each box is the median. The upper and lower box boundary are the third and first quartile, respectively. The horizontal bars at either end of the dashed vertical line represent one standard deviation around the mean.
- In three of the four cases, disagreement between experts is substantial.
- Plot on the right: The standard deviation increases linearly with the overall number of stained nuclei. (Roughly, the more cancer cells there are, the more volatile the diagnosis becomes.)
Fig. 3. Tissue microarray (TMA): primary tissue samples are taken from a cancerous kidney (a). Then tissue cylinders of a 0.6 mm diameter are extracted from the primary tumor material of different patients and arrayed in a recipient paraffin block (b). Slices of 0.6 μm are cut off the paraffin block and are immunohistochemically stained (c). These slices are scanned as whole slide images and tiled into single images representing different patients. Image (d) depicts a TMA spot of clear cell renal cell carcinoma stained with MIB-1 (Ki-67) antigen. (e) shows details of the same spot containing stained and non-stained nuclei of normal as well as atypical cells.

Fig. 4. (a) A quarter of an RCC TMA spot used for the nuclei detection experiment. (b) Annotations of one expert, indicating atypical nuclei in black and normal ones in red. (c) Overlay of detected nuclei from expert one (blue circles) and expert two (red crosses). (d) Disagreement between the two domain experts regarding the detection task. Nuclei which were labeled only by pathologist one are shown in blue and the nuclei found only by expert two are depicted in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Fig. 5. (a) Inter-pathologist classification variability based on 180 nuclei labeled by five domain experts. The experts agree on 105 out of 180 nuclei (blue bars: 24 normal, 81 atypical). (b–d) Confusion matrices including reader confidence for intra-observer variability in nuclei classification: (b) The combined result of all five experts yields an intra-pathologist classification error of 21.2%. (c) Example of an extremely self-confident pathologist with 30% error. (d) A very cautious pathologist with a misclassification error of 18%.
Data
“Relative images” of individual cell nuclei, i.e. the difference between a nucleus image and a background patch from the same tissue sample image. That makes results less sensitive to variations between tissue samples.

Training
1. Nuclei are hand-labeled by pathologists.
2. A random forest classifier is trained on the relative images (as training data points) and the labels (as training labels).

Diagnosis of a new tissue sample image
1. Input: Entire tissue sample image.
2. Find nuclei using an image segmentation algorithm.
3. Extract subimages of these nuclei and apply random forest classifier.
4. Diagnose according to ratio of healthy to cancerous cells.
DATA ANALYSIS PIPELINE

Experiment

Preprocessing

Raw data (measurements)

Feature extraction

Working data

Mark patterns

Split

Training data (patterns marked)

Training (calibration)

Trained model

Test data (patterns marked)

Apply on test data

Error estimate

extract tissue samples

mark and photograph

images

find and extract cell nuclei

cell nuclei images

annotate as healthy/cancerous

random forest classifier
classifies cells as healthy/cancerous
RESULTS

Fig. 2. Precision/Recall plot of cross validation results on the renal clear cell cancer (RCC) dataset. For Relational Detection Forests (RDF) curves for the nine single patients and their average (bold) are depicted. RDF with the proposed feature base outperforms previous approaches based on SVM clustering [14], mathematical morphology and combined methods [15]. The interpathologist performance is depicted in the top right corner.

3.2 Clear Cell Renal Cell Carcinoma (ccRCC) Detection Accuracy:

Three fold cross validation was employed to analyze the detection accuracy of RDFs. The nine completely labeled patients were randomly split up into three sets. For each fold the ensemble classifier was learned on one patient and tested on the other three. During tree induction, at each split 500 features were sampled from the feature generator. Trees were learned to a maximum depth of 10 and the minimum leave size was set to 1. Therefore, trees converge after 150 to an out of bag (OOB) error of approximately 2%. Finally, on the test image, each pixel was classified and mean shift was run on a grid with \( \delta = 5 \).

Figure 2 shows precision/recall plot for single patients and the average result for the RDF object detector. The algorithm is compared to point estimates of several state of the art methods: SVM clustering was successfully employed to detect nuclei in H&E stained images of brain tissue by [14]. SVMmorph [15] is an unsupervised morphological [16] approach for detection combined with an supervised support vector machine for filtering. The marker for the pathologist shows the detection accuracy if alternately one expert is used as gold standard. On average the pathologists disagree on 15% of the nuclei.

Although only grayscale features were used for RDF it outperforms all previous approaches which also utilize texture and color. This observation can be a cue for further research.

Precision/Recall plot for the random forest method ("RDF") compared to other classifiers. The "true label" for each data point is a randomly selected pathologist. The performance of pathologists (red dot) is the average of the aggregate result for all remaining pathologists.
This application illustrates a number of challenges encountered in applications:

- Generating label information is work-intensive.
- The comparison experiments show that the training/test labels themselves have limited reliability.
- These methods are now several years old. Neural networks developed in the last few years might be able to improve the feature extraction step. (More on neural networks and feature extraction later.)