SUMMARY We propose multiscale bagging as a modification of the bagging procedure. In ordinary bagging, the bootstrap resampling is used for generating bootstrap samples. We replace it with the multiscale bootstrap algorithm. In multiscale bagging, the sample size \( m \) of bootstrap samples may be altered from the sample size \( n \) of learning dataset. For assessing the output of a classifier, we compute bootstrap probability of class label; the frequency of observing a specified class label in the outputs of classifiers learned from bootstrap samples. A scaling-law of bootstrap probability with respect to \( \sigma^2 = n/m \) has been developed in connection with the geometrical theory. We consider two different ways for using multiscale bagging of classifiers. The first usage is to construct a confidence set of class labels, instead of a single label. The second usage is to find inputs close to decision boundaries in the context of query by bagging for ordinary bagging procedure is an application of the non-parametric bootstrap of [4] to predictors as described below. Given the training dataset \( \mathcal{D} \), we first generate \( B \) bootstrap samples \( \mathcal{D}^b \), \( b = 1, \ldots, B \), by resampling with replacement. Then, we create \( B \) bootstrap replicates of the classifier \( f(x; \hat{\mathbf{\theta}}(\mathcal{D}^b)) \), \( b = 1, \ldots, B \). For a specified \((x, y)\), the frequency of the outputs \( f(x; \hat{\mathbf{\theta}}(\mathcal{D}^b)) \) being \( y \) is computed as

\[
\alpha(x, y) = \frac{1}{B} \sum_{b=1}^{B} I(f(x; \hat{\mathbf{\theta}}(\mathcal{D}^b)) = y),
\]

where \( I() \) is the indicator function taking 0/1 values, while the dependency of \( \alpha(x, y) \) on \( \mathcal{D} \) through the bootstrap samples is suppressed in this notation. In the bootstrap literature, \( \alpha(x, y) \) is known as the bootstrap probability. This first appeared in [6] in the context of phylogenetic analysis. In this paper, we assume \( B \) is large enough (say \( B = 10000 \)) so that the variance \( \approx \alpha(x, y)(1 - \alpha(x, y))/B \) can be ignored numerically.

Our generalization of bagging is very simple. In ordinary bagging, the sample size of \( \mathcal{D}^* \) was \( n \), the same as that of \( \mathcal{D} \). We intentionally alter it to an arbitrary \( m > 0 \), and define a scale parameter \( \sigma^2 = n/m \). This is the multiscale bootstrap of [9]–[11] applied to bagging, and we shall refer to this procedure as multiscale bagging. Changing \( m \) from \( n \) is not new in the bootstrap literature, and it is found in the subsampling or the \( m \) out of \( n \) bootstrap [2], [8]. An apparent difference is that we use finite values of \( \sigma^2 \), whereas the other methods use \( \sigma^2 \rightarrow \infty \) as \( n \rightarrow \infty \). Ordinary bagging uses \( \sigma^2 = 1 \), but we will change \( \sigma^2 \) to other values. In Sect. 2, we will review a scaling-law of \( \alpha(x, y) \), that tells how it changes along \( \sigma^2 \), developed in [9]–[11] in connection with the geometrical theory of [5].

Typically, ordinary bagging is used for reducing instability of the classifier by averaging the outputs. For this purpose, a stable version of \( f(x; \hat{\mathbf{\theta}}(\mathcal{D})) \) is the classifier defined as arg max\( p \in \{1, \ldots, K\} \) \( \alpha(x; y) \). Although there is nothing wrong with this usage of \( \alpha(x; y) \), we attempt to use the multiscale bagging in two other ways as follows.

The first usage is to construct a confidence set for \( y \) in order to indicate possible values of the true output. For this purpose, \( \alpha(x, y) \) may be interpreted as a confidence level of the classifier. From a frequentist point-of-view, this is a \( p \)-value for statistically testing the null hypothesis stating that the true output for \( x \) is \( y \). As we will see in Sect. 2.3,
α(x, y) of ordinary bagging is biased. We consider an approximately unbiased p-value and apply it to estimate a confidence set of class labels. Computing the bootstrap probabilities α(x, y) for several m > 0 values is important to correct the bias of the ordinary bagging. Detailed arguments are given in Sects. 2.3 and Sect. 3.1.

The second usage is to find an input x which is very close to the decision boundary. In binary classification problems, the query by bagging (QBag) procedure of [1] attempts to find x with α(x, y) = 1/2, hoping that x is close to the decision boundaries. In Sect. 2.4, we will show that ordinary bagging is again biased for this purpose, and we propose an unbiased version of QBag, called query by multiscale bagging (QMBag). Similar to the first usage of multiscale bagging, the approximately unbiased p-value has a significant role to achieve unbiased estimation.

Our contribution here is to introduce the multiscale bootstrap method to bagging, i.e., multiscale bagging. The multiscale bootstrap method is already used frequently for computing approximately unbiased p-values in phylogenetic analysis [9], and doing the same for bagging may not be regarded as really new, yet this paper is the first attempt to active learning. This approach has never be seen in any other applications of the multiscale bootstrap method.

2. Multiscale Bagging in Classification Problems

We review briefly the algorithm and theory of multiscale bootstrap method. The geometrical argument in Sects. 2.2 to 2.4 for connecting x-space and θ-space is newly given in this paper. Also new in this paper is the argument in Sect. 2.4 in which a method of finding points on decision boundary is considered.

2.1 Algorithm of Multiscale Bagging

For the training dataset \(D = \{(x_t, y_t), t = 1, \ldots, n\}\) of sample size n, let \(D_m^n\) be the bootstrap samples \(D_m^n = \{(x_t', y_t'), t = 1, \ldots, m\}\), of size m, and let us define a scale parameter \(σ^2 = \frac{σ^2}{m}\) which may indicate a ratio of the variance of \(\hat{θ}(D_m^n)\) relative to that of \(\hat{θ}(D)\). For a specified m, we generate \(D_m^n, D_m^{n^2}, \ldots, D_m^{n^B}\), and compute the bootstrap probability as

\[
α_{σ^2}(x, y) = \frac{1}{B} \sum_{b=1}^{B} I\left(f(x; \hat{θ}(D_m^{nb})) = y\right).
\]

For \(σ^2 = 1\), \(α_1(x, y)\) reduces to \(α(x, y)\) of Eq. (1).

Let us define a normalized bootstrap probability as

\[
p_{σ^2}(x, y) = \Phi(σ\Phi^{-1}(α_{σ^2}(x; y))).
\]  

where \(Φ^{-1}(\cdot)\) is the inverse function of the cumulative distribution function \(Φ(\cdot)\) of the normal distribution with mean 0 and variance 1. For \(σ^2 = 1\), \(p_1(x, y)\) reduces to \(α(x, y)\) again. In Sect. 2.3, we show that \(p_{σ^2}(x, y)\) with \(σ^2 = -1\) is regarded as an approximately unbiased p-value, and in Sect. 2.4, we attempt to find x with \(p_0(x, y) = 1/2\) in order to obtain points on the decision boundary. We need to estimate \(p_{σ^2}(x, y)\) even on negative \(σ^2\) to obtain approximately unbiased p-value, although one can access \(p_{σ^2}(x, y)\) with only positive \(σ^2\) in practice. Therefore, we need to estimate \(p_{σ^2}(x, y)\) as the function of \(σ^2 = \frac{σ^2}{m}\) in order to extrapolate the normalized bootstrap probability.

The multiscale bagging provides the procedure to estimate the normalized bootstrap probability \(p_{σ^2}(x, y)\). Figure 1 shows the details of the algorithm. This algorithm, particularly Step 4 with several model options, is implemented in the scaleboot package for R available from CRAN.

2.2 Scaling-Law and Geometry of Bootstrap Probability

The decision region of the classifier \(f(x; θ)\) for class label y is defined as \(X(θ, y) = \{x ∈ X | f(x; θ) = y\}\), and the set of \(θ\) satisfying \(x ∈ X(θ, y)\) is defined as

\[
H(x, y) = \{θ ∈ Θ | f(x; θ) = y\}.
\]

These two sets are related to each other by the equivalence

\[
x ∈ X(θ, y) \iff θ ∈ H(x, y),
\]

which connects the geometry in x-space and that in θ-space. Although we are familiar with the geometry of classifiers in terms of \(X(θ, y)\), the geometry of the bootstrap method is best understood by working on \(H(x, y)\).

Let us assume the boundary surface of \(H(x, y)\), denoted as \(∂H(x, y)\), is smooth. Then, the scaling-law of the bootstrap probability is expressed as

\[
σΦ^{-1}(p_{σ^2}(x, y)) = γ_0(x, y) + γ_1(x, y)σ^2 + γ_2(x, y)σ^4 + γ_3(x, y)σ^6 + \cdots,
\]

where \(γ_0(\cdot)\) is a more accurate version of the simple linear model (3). In [11], the negative of (6), that is, \(θ(σ^2) := -σΦ^{-1}(p_{σ^2}(x, y))\), is called normalized bootstrap z-value.
The coefficients $\gamma_i(x,y)$, $i = 0, 1, 2, \ldots$, have geometrical interpretations. $\gamma_0(x,y)$ is the signed distance from $\theta(D)$ to $\partial H(x,y)$, taking the positive/negative sign when $\partial(D)$ is inside/outside of $H(x,y)$. Let $\hat{\theta}$ denote the projection of $\theta(D)$ to $\partial H(x,y)$, that is, the point $\theta \in \partial H(x,y)$ for which $|\theta(D) - \theta|$ is minimized. Using $\hat{\theta}$, the signed distance is written as $\gamma_0(x,y) = \pm |\theta(D) - \hat{\theta}|$. $\gamma_1(x,y)$ is the mean curvature of $\partial H(x,y)$ at $\hat{\theta}$, taking the positive/negative sign when $\partial H(x,y)$ is curved away/toward $H(x,y)$. $\gamma_2(x,y)$ is related to the fourth derivatives of $\partial H(x,y)$, and so on.

We are free to choose any parametrization of $\theta$ for using the multiscale bootstrap algorithm. However, the parametrization of $\theta$ should be chosen carefully for justifying the geometrical interpretations described above. We assume that the multivariate normal model of [5],

$$\theta(D) \sim N_0(\hat{\theta}, I)$$

holds at least approximately. The conditional distribution of the bootstrap replicates given $D$ is approximately

$$\theta(D_m|D) \sim N_0(\hat{\theta}(D), \sigma^2 I).$$

For an arbitrary parametrization of $\theta$, we assume that there exists a transformation of $\theta$ so that the transformed $\hat{\theta}$ satisfies the multivariate normal model. Note again that this assumption is only for the geometrical interpretations, and we do not actually need to know such a transformation when using the multiscale bagging algorithm.

2.3 Computing Confidence Level of Classification

We first recall the properties of $p$-values. For a specified $(x,y)$, the approximately unbiased $p$-value, denoted $p(x,y)$ here, is the probability value indicating how much confidence we have for the null hypothesis stating that the true output is $y$ when the input is $x$. This null hypothesis is expressed as either $x \in X(\hat{\theta}, y)$ or $\hat{\theta} \in H(x,y)$. The value of $p(x,y)$ should become smaller as $x$ moves away from $X(\hat{\theta}, y)$ or as $\hat{\theta}(D)$ moves away from $H(x,y)$. Let $\beta$ denote a level of significance for statistical testing, say $\beta = 0.05$. We reject the null hypothesis if $p(x,y) < \beta$. On the other hand, we may think that $y$ could possibly be the true output if $p(x,y) \geq \beta$.

Let us consider the rejection probability $P(p(x,y) < \beta)$, where the probability is taken with respect to $D$ while $(x,y)$ is fixed. Note that $p(x,y)$ is a random variable, because it depends on the training samples $D$, although this dependency is suppressed in the notation. The approximately unbiased $p$-value should satisfy, by definition,

$$P(p(x,y) < \beta) \approx \beta, \quad \text{for } \beta \in (0, 1), \hat{\theta} \in \partial H(x,y).$$

In other words, $p(x,y)$ is distributed uniformly on $(0, 1)$ when $\hat{\theta}$ is on the boundary of $H(x,y)$ or equivalently when $x$ is on the true decision boundary $\partial X(\hat{\theta}, y)$. The rejection probability increases as $\hat{\theta}$ moves away from $H(x,y)$, or as $x$ moves away from $X(\hat{\theta}, y)$.

We have only explained the properties of $p(x,y)$. It would be nice if we could compute such a $p(x,y)$ from $D$. [11] showed that the approximately unbiased $p$-value is expressed using the geometrical quantities $\gamma_i$’s in (6) as

$$p(x,y) = \Phi(\gamma_0(x,y) - \gamma_1(x,y) + \gamma_2(x,y) - \gamma_3(x,y) + \cdots). \quad (10)$$

An equivalent formula up to the first two terms $\gamma_0(x,y)$ and $\gamma_1(x,y)$ is found also in [5] and [9], [10]. The formula (10) is not sufficient for computing the value of $p(x,y)$, because the values of $\gamma_i$’s are only available through the carefully chosen parametrization of $\theta$. To compute the value of (10) numerically, we use the multiscale bagging algorithm in Fig.1. Comparing (10) with (6), we find that $p(x,y) = p_{\gamma_0}(x,y)$ using $p_{\gamma_0}(x,y)$ of (2).

2.4 Finding Points on Decision Boundaries

The normalized bootstrap probability $p_{\gamma_0}(x,y)$ is available to find the points on decision boundary. For a specified $(x,y)$, we would like to tell if $x \in \partial X(\hat{\theta}, y)$ or how close $x$ is to $\partial X(\hat{\theta}, y)$. This task is often trivial for simple classifiers using specific forms of $f(x; \theta)$. For example, the estimator of conditional probability, $\hat{\rho}(y|x)$, provides the decision boundary between two labels $y$ and $y'$, that is, $\{x \in X | \hat{\rho}(y|x) = \hat{\rho}(y'|x)\}$. Here, however, we discuss methods to find such points using only bootstrap probabilities. That is, our method is applicable to classifiers that do not provide probability estimates of class labels.

Considering that $x \in \partial X(\hat{\theta}, y) \iff \hat{\theta} \in \partial H(x,y)$, the task is equivalent to finding $x$ so that $\hat{\theta}$ is very close to $\partial H(x,y)$. Using the geometrical interpretations of Sect. 2.2 the distance from $\hat{\theta}$ to $\partial H(x,y)$ is expressed as $|\gamma_0(x,y)|$. Therefore, the task is equivalent to finding $x$ with small value of $|\gamma_0(x,y)|$. When $x \in \partial X(\hat{\theta}, y)$, the signed distance becomes $\gamma_0(x,y) = 0$. The issue is then how to compute $\gamma_0(x,y)$ numerically. By recalling the multiscale-Bagging algorithm in Fig.1, we notice that we have already computed $\gamma_0(x,y)$ in the Step 4 of the algorithm, meaning we have already solved the problem.

Look at the scaling-law given in Eq. (6). The right hand side becomes $\gamma_0(x,y)$ by letting $\sigma^2 = 0$, and thus $p_0(x,y) = \Phi(\gamma_0(x,y))$ using $p_{\gamma_0}(x,y)$ of (2). Therefore $p_0(x,y) = 1/2$ if $x \in \partial X(\hat{\theta}, y)$, and $p_0(x,y)$ becomes smaller/larger as $x$ moves away/toward $X(\hat{\theta}, y)$. Hence, the input point $x$ with small $|p_0(x,y) - 1/2|$ for some $y \in \{1, \ldots, K\}$ would be close to the decision boundary on class labels.

3. Applications of Multiscale Bagging

We show various applications of multiscale bagging in classification problems. First, we show a way of estimating the confidence set of class labels. Second application is the active learning [12] in which input points of training samples are chosen appropriately to achieve high prediction accuracy. In both applications, the bias reduction of the bootstrap probability has the significant role as shown below.
3.1 Confidence Set of Class Labels

Given a training dataset $\mathcal{D}$, we estimate the parameter $\hat{\theta}(\mathcal{D})$ and compute the output $f(x; \hat{\theta}(\mathcal{D}))$ for a specified input $x$. We have assumed that the consistency of the estimator so that $\hat{\theta}(\mathcal{D}) \rightarrow \theta$ as $n \rightarrow \infty$. Thus we can expect that the prediction $f(x; \hat{\theta}(\mathcal{D}))$ converges to the true output $f(x; \theta)$ for sufficiently large $n$. We may then wonder how much confidence do we have in the output when $\hat{\theta}$ is used for generating the dataset. It turned out, however, that $\hat{\theta}$ is not extremely large. For a specified $(x, y)$, we interpret $\alpha(x, y)$ as a confidence level for assessing the possibility that $f(x; \theta) = y$. Given a threshold value $\beta = 0.05$, say, we may compute a confidence set of class labels

$$S_\alpha(x) = \{y \in \{1, \ldots, K\} | \alpha(x, y) \geq \beta\},$$

hoping that the true output should be included in $S_\alpha(x)$ at a reasonably high probability.

For illustrating the method, we compute $\alpha(x, y)$ values for the example given in the following: Fig. 2 shows the scatter plot of training samples.

- **Data generation.** $x_i \in X = \mathbb{R}^2$ are generated from a normal mixture of four components. For each label $y = 1, \ldots, 4$, $y_i = y$ and $x_i \sim N_2(\mu_{y}, I)$ with $\mu_1 = (1.5, 0), \mu_2 = (0, 1.5), \mu_3 = (1.5, 0), \mu_4 = (1, 1.5)$.

- **Classifier.** A Bayes classifier of $K = 4$ classes. For each class $i$, $x_i \sim N_2(\mu_i, \Sigma)$ is assumed. The parameter is $\theta = (\mu_1, \mu_2, \mu_3, \mu_4, \Sigma, \pi_1, \pi_2, \pi_3)$, and $\hat{\theta}(\mathcal{D})$ is the maximum likelihood estimator.

- **Multiscale Bootstrap.** $M = 9$ values of the bootstrap sample size are specified as $m_1 = 67, m_2 = 88, m_3 = 91, m_4 = 183, m_5 = 200, m_6 = 263, m_7 = 346, m_8 = 456, m_9 = 600$ so that $\sigma^2$ value ranges from 3 to 1/3. For each $m_i, B = 10000$ bootstrap samples are generated, and the total number of bootstrap samples is $M \times B = 90000$.

A Bayes classifier with linear boundaries is estimated from a training dataset of $n = 200$. The number of classes is $K = 4$. We specify the input $x$ to be $a = (0.48, 0.25)$ as indicated in Fig. 2. The true output is $f(a; \theta) = 2$ with respect to a model used for generating the dataset. It turned out, however, that the estimated classifier predicted wrongly $f(a; \hat{\theta}(\mathcal{D})) = 2$ from this particular training dataset. The bootstrap probabilities are $\alpha(a, 1) = 0.042, \alpha(a, 2) = 0.915, \alpha(a, 3) = 0.003, \alpha(a, 4) = 0.039$ and the confidence set is $S_\alpha(a) = (2)$ for $\beta = 0.05$. Unfortunately, the true output $y = 1$ was not included in $S_\alpha(a)$.

We next redefine the confidence set of class labels using the approximately unbiased $p$-value $p(x, y)$.

$$S_p(x) = \{y \in \{1, \ldots, K\} | p(x, y) \geq \beta\},$$

where $p(x, y)$ is computed numerically as $p_{-1}(x, y)$ by the multiscale bootstrap algorithm. For the example of Fig. 2, the $p$-values are $p(a, 1) = 0.074, p(a, 2) = 0.964, p(a, 3) = 0.009, p(a, 4) = 0.060$ and the confidence set is $S_p(a) = \{1, 2, 4\}$. Fortunately, the true output $y = 1$ was included in $S_p(a)$. Computation of $p(a, 1)$ and $p(a, 2)$ is shown in Fig. 3.

A good point of $S_p(x)$ is that the coverage probability is not less than $1 - \beta$. That is,

$$P\left( f(x; \theta) \in S_p(x) \right) \geq 1 - \beta$$

holds approximately, where $x$ is fixed and the probability is taken with respect to $\mathcal{D}$. Considering the dependency of $p(x, y)$ on $\mathcal{D}$ through the bootstrap samples, $S_p(x)$ is a random variable taking the value in subsets of $\{1, \ldots, K\}$. The unbiasedness shown in Eq. (9) implies that the equality $P\left( f(x; \hat{\theta}) \in S_p(x) \right) = 1 - \beta$, approximately holds, when $x \in \partial(X(\theta, y))$ for some $y \in \{1, \ldots, K\}$. That is, the size of the set $S_p(x)$ is as small as possible under the most ambiguous situation, while satisfying the inequality (13).

On the other hand, the coverage probability of $S_\alpha(x)$ computed based on $\alpha(x, y)$ can be smaller than $1 - \beta$ by making $S_\alpha(x)$ smaller than it should be. As a result, $S_\alpha(x)$ often gives overconfidence to wrong outputs.

3.2 Active Learning

In the pool-based active learning [12], we intend to update
the training dataset $\mathcal{D}$ by choosing a new input from a pool of unlabelled inputs. The class label is unknown until we query it. By choosing informative inputs, we expect to reduce the sample size required for achieving a specified accuracy of the estimated classifier, comparing to the ordinal active learning, we may find points inside of the decision region. Then, the estimated parameters are shown. Each band in the background corresponds to 0.2 difference in the value of $\gamma_{0}(x, 1)$.

Below, we show a simple example to see if the estimated decision boundaries can be identified by the multiscale bagging. The setup is shown below.

- **Data generation.** $x_i \in X = \mathbb{R}^2$ are generated from a normal mixture of three components. For each $y_i \in \{1, 2, 3\}$, $y_i \sim N_2(\mu_i, I)$ with $\mu_1 = (1, 0)$, $\mu_2 = (-1, 0)$, $\mu_3 = (0, -1/\sqrt{3})$. The class probability is $\pi_y = 1/3$ for $y \in \{1, 2, 3\}$. The sample size is $n = 300$.
- **Classifier.** The same Bayesian classifier as that of Sect. 3.1 but $K = 3$.
- **Multiscale Bootstrap.** $M = 9$ values of the bootstrap sample size are specified so that $\sigma_i^2$ value ranges from 3 to 1/3. For each $\sigma_i^2 (i = 1, \ldots, M)$, $B = 10000$ bootstrap samples are generated.

The result is shown in Fig. 5. We computed $p_0(x, 1)$ at $30 \times 40$ grid points of $x$ for $\sigma^2 \in \{1, 0, -1\}$. It is $\alpha(x, 1)$ when $\sigma^2 = 1$, and $p(x, 1)$ when $\sigma^2 = -1$. Then contour lines for $p_0(x, 1) = 1/2$ are drawn. As we expected, the contour line for $p_0(x, 1) = 1/2$ is very close to the estimated decision boundary. The contour line for $\sigma^2 = 1$ moves toward inside of the decision region $X(\hat{\theta}, 1)$, while the contour line for $\sigma^2 = -1$ moves away from the decision region.

The geometry in $x$-space and that in $\theta$-space are closely related by (5). In active learning, we attempt to find a point $x$ which is close to the estimated decision boundary $\partial X(\hat{\theta}, 1)$ by minimizing the distance in $\theta$-space, that is $|\gamma_0(x, 1)|$. This is justified because the equivalence $x \in \partial X(\hat{\theta}, 1) \iff \gamma_0(x, 1) = 0$. Furthermore, the contour lines of $\gamma_0(x, 1)$, drawn for every 0.2 difference, are aligned nearly parallel to $\partial X(\hat{\theta}, 1)$ in Fig. 5. This implies $|\gamma_0(x, 1)|$ can be used as a substitute for a distance to the decision boundary in $x$-space, taking account of the sensitivity of $\theta$ which depends on the direction in $x$-space.

---

**Algorithm:** Query by multiscale bagging (QMBag)

**Input:** Learning algorithm $\mathcal{A}$. Number of iterations $N$. Number of query samples $n'$, labelled training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a pool of unlabelled inputs, denoted as $\mathcal{U} = \{(x_1, \ldots, x_N)\}$.

**Repeat** the following steps $N$ times:

1. Based on the labelled dataset $\mathcal{D}$ and the algorithm $\mathcal{A}$, compute the normalized bootstrap probability $p_2(\mathcal{U})$ for each $\mathcal{U} \in \mathcal{C}$ and each label $y \in \{1, \ldots, K\}$. The multiscale bagging is available to obtain $p_2(\mathcal{U})$.
2. Sort unlabelled data $\mathcal{u} \in \mathcal{U}$ in increasing order of $\text{minimum}_{x \in \mathcal{D}} [p_2(\mathcal{U}) - 1/2]$, and then choose the first $n'$ inputs from $\mathcal{U}$. Then, one has the set of new queries $\mathcal{Q} = \{(x_{n+1}, \ldots, x_{n+n'})\} \subset \mathcal{U}$.
3. Observe the outputs $y$ for $x \in \mathcal{Q}$. We get a new training dataset $\mathcal{L} = \{(x_{n+1}, y_{n+1}), \ldots, (x_{n+n'}, y_{n+n'})\}$.
4. Update $\mathcal{U} := \mathcal{U} \setminus \mathcal{Q}$, $\mathcal{D} := \mathcal{D} \cup \mathcal{L}$.

**Output:** estimated classifier obtained from $\mathcal{D}$.

**Fig. 4** Active learning algorithm with multiscale bagging.
4. Numerical Studies

In this section we show three numerical results. First we confirm the unbiasedness of \( p \)-value estimated by the multiscale bagging. Then, we conduct various numerical simulations about the confidence set of the classification problems and the active learning. Benchmark dataset is used to assess each learning algorithm.

4.1 Unbiasedness of \( p \)-value

We conduct a simulation study for confirming the distributional properties of \( \alpha(x, y) \) and \( p_{\alpha}(x, y) \) with \( \sigma^2 = -1 \). We have generated 1000 training datasets of \( n = 200 \) from the normal mixture model specified in Data generation in Sect. 3.1. The number of bootstrap samples is reduced to \( B = 1000 \). Each histogram in Fig. 6 is plotted from 1000 values of \( \alpha(x, 1) \) and \( p_{\alpha}(x, 1) \), where \( x \) is one of \( b = (0, 0) \), \( c = (0, 1, 0.1) \), and \( d = (0.5, 0.5) \) in Fig. 2. These three points of \( x \) are located on the true decision boundary \( \partial X(\theta, 1) \) as shown in Fig. 2. Therefore the unbiasedness given in Eq. (9) implies that \( p(x, 1) \) for \( x = b, c, d \) should be distributed uniformly on \((0, 1)\). In fact, the histograms of the estimated \( p \)-values, \( p(c, 1) \) and \( p(d, 1) \), are almost flat, meaning they are very close to the uniform distribution on \((0, 1)\).

The histogram of \( p(b, 1) \), however, indicates that \( P(p(b, 1) \leq 0.1) \approx 0.2 \), although it should be 0.1 for the uniform distribution. As a result, the coverage probability \( P(1 \in S_p(b)) \approx 0.8 \) for \( \beta = 0.1 \), while it should be \( \geq 0.9 \).

This discrepancy between the theory and the simulation is attributed to the singularity of the true decision region at \( b \), which leads to a singularity of \( \partial H(b, 1) \). The unbiasedness property is not very accurate when the boundary surface is not smooth, or when the curvature of smooth surface is very large.

Comparing the histograms of \( \alpha(x, 1) \) with those of \( p(x, 1) \), we find that \( \alpha(x, 1) \) is more biased than \( p(x, 1) \). For example, \( P(\alpha(b, 1) \leq 0.1) \approx 0.4 \) and thus \( P(1 \in S_\alpha(b)) \approx 0.6 \) for \( \beta = 0.1 \). The true output is then often not included in the confidence set as we saw in Sect. 3.1. However, \( \alpha(d, 1) \) behaves almost similarly as \( p(d, 1) \). This is because the boundary surface is flat in a reasonably large neighbourhood around \( d \), leading to a flat \( \partial H(d, 1) \). Then, the curvatures \( \gamma_1(x, y), \gamma_2(x, y), \ldots \) in Eq. (6) vanish, and thus Eq. (6) and Eq. (10) lead to \( \alpha(d, 1) \approx p(d, 1) \).

4.2 Numerical Experiments on Confidence Set

In the second numerical experiments, we use some datasets in R library, mlbench and the dataset generated by the R command, mlbench.2dnormals. Each dataset in mlbench is split up into training samples, test samples, and query samples. For each dataset, we repeat numerical experiments 20 times with different random splits of the data in order to evaluate the averaged generalization performance of each learning algorithm. The properties of each datasets are summarized in Table 1, where “dim”, “class”, “#tr.”, “#test.”, and “rep.” denote the input dimension, the number of labels, the size of training set, the size of test set and the number of replication of learning, respectively. As the estimator of the multiclass classification, we use the Fisher linear discriminant method which is implemented in R command, lda.

The confidence set of the class label is estimated based on the bootstrap probability \( \alpha(x, y) \) or the approximately unbiased \( p \)-value \( p_{\alpha}(x, y) \). The former (later) is denoted as QBaQ (QMBag) in Fig. 7 which shows the coverage probability of the confidence set for \( \beta = 0.05 \). For the data set mlbench.2dnormals, the statistical model of 1da is correct, and in this case, QMBag significantly corrects the bias of the coverage probability. When the number of training data is insufficient for the high dimensional data, the QMBag will be unstable. This would be the primal reason that, in the data sets Glass and LetterRecognition, the bias of QBaQ is not improved by QMBag. We see that the \( p \)-value estimated by QMBag still has some bias, even when the bias of the QBaQ is corrected by QMBag. As illustrated in Sect. 4.1, this is probably because the decision boundary is not smooth.

\[ \text{Table 1 Properties of each dataset.} \]

| dataset            | dim | class | #tr.  | #test. | rep. |
|--------------------|-----|-------|-------|--------|------|
| mlbench.2dnormals  | 2   | 15    | 100-500 | 1000   | 20   |
| Vehicle            | 18  | 4     | 200-500 | 300    | 20   |
| Glass              | 9   | 6     | 80-130  | 80     | 20   |
| Satellite          | 36  | 6     | 800-1600| 2000   | 20   |
| Vowel              | 10  | 11    | 300-500 | 400    | 20   |
| LetterRecognition  | 16  | 26    | 800-1600| 2000   | 20   |
and the estimation of the unbiased p-value becomes unstable. We need more accurate and stable method for obtaining the confidence set of the class labels.

4.3 Numerical Experiments on Active Learning

First, we show numerical results for a toy problem. We have generated 1000 datasets to see average performance of active learning methods. Each of the 1000 runs of the simulation is described below.

- **Data generation.** $x_i \in \mathcal{X} = \mathbb{R}^2$ are generated uniformly on $[-10, 10] \times [-10, 10]$. For each $x_i$, $y_i \in \{1, 2, 3\}$ is generated by the conditional probability $P(y = \hat{y}(x)) = \varphi_1(x)/\sum_{j=1}^3 \varphi_j(x)$, where $\varphi_1(x) = 1, \varphi_2(x) = \exp(A_2 \cdot (1, x)), \varphi_3(x) = \exp(A_3 \cdot (1, x))$, where $A_2 = (0, \sqrt{3}/2, 2, 3/2)$ and $A_3 = (0, -\sqrt{3}/2, 2, 3/2)$. We first generate the dataset, $\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, n = 1000$.

- **Classifier.** $f(x; \theta) = \arg \max_{i=1,2,3} P(y = \hat{y}(x))$ using the model of data generation. The parameter is $\theta = (A_2, A_3)$, and $\hat{\theta}(\mathcal{D})$ is the maximum likelihood estimator.

- **Multiscale Bootstrap.** $M = 9$ values of the bootstrap sample size are specified so that $\sigma_i^2$ value ranges from 3 to 1/3. For each $\sigma_i^2$, $B = 1000$ bootstrap resamples are generated.

- **Active Learning.** The initial $\mathcal{D}$ is made by sampling randomly from $\mathcal{T}$, and updating $\mathcal{T} := \mathcal{T} \setminus \mathcal{D}$. The initial sample size is $n = 100$. The initial unlabelled training samples, $\mathcal{U}$, is made from $\mathcal{T}$ by discarding the class labels. We then examine passive learning, QBag, QMBag, and distance method. Here the distance method denotes that the criterion to choose query samples is $\min_{x_i} \|x' - \hat{x}\|$, where the minimum is taken over $x' \in \partial X(\theta, y)$. In QBag, QMBag and distance method, the number of query samples at each iteration is set to $n' = 30$.

Although the distance method directly finds the points on the decision boundary, it demands high computational cost. Indeed, in the distance method, we need to solve the optimization problem $\min_{x_i} \|x' - \hat{x}\|$ subject to $x' \in \partial X(\theta, y)$ which is a complex constraint in general. On the other hand, the QBag and QMBag do not require any complex computation. They only need the bootstrap resampling. Hence, the bagging-based approaches are computationally promising, compared with the distance method.

The numerical result is shown in Fig. 8. Plotted are the mean squared errors of the parameter $A_2$ for the four sampling schemes of active learning. The error reduces as the sample size $n$ increases. Among the four methods, the random sampling performs most poorly. QBag procedure reduces the error much quickly than the random sampling. QMBag procedure performs better than QBag. QMBag is almost equivalent to the distance method, which is shown here as an asymptotically optimal one.

Next, we apply the active learning algorithm to various benchmark dataset shown in Table 1. The Fisher discriminant analysis (1da) is applied as the learning algorithm in the same way as Sect. 4.2. The numbers of training and test samples are not necessarily equal to those in Table 1. For each dataset, we repeat numerical experiments 20 times with different random splits of the data in order to evaluate the averaged generalization performance. Below, passive learning, QBag and QMBag are compared. For QBag and QMBag, the number of bootstrap resampling is set to $B = 1000$. The distance method is excluded, since applying distance method above is not practical for real-world data. In Fig. 9, the test errors to the sample size are presented. In
all dataset but Glass, QBag and QMBag dominate the passive learning. For the parameter estimation, QMBag is superior to QBag in the large sample limit as show in the first numerical studies of this section. The difference of the test error between QBag and QMBag is, however, not significant in these experiments.

5. Discussion

We have proposed multiscale bagging, and discussed its two usages, namely, constructing confidence set of class labels, and finding inputs close to decision boundaries.

[12] have proposed the support vector machine (SVM) with active learning. In their approach, the version space [7], in the parameter space is considered, and the input point is chosen such that the volume of the version space is nearly halved. To avoid the computational difficulty, they proposed the criterion named “simple margin” in which the input point closest to the estimated decision boundary is selected as an alternative. The input point chosen according to the simple margin may lead to a split nearly halving the version space. The version space is, however, well-defined only when the training data is linearly separable and the output label is binary, we need a remedy to apply the SVM active learning to the data which does not meet the condition. The simple margin criterion is close to our approach in active learning. Differently from SVM active learning, the query by (multiscale) bagging does not refer to the version space, and hence our algorithm is directly applicable to non-separable data. In addition query by (multiscale) bagging can deal with multi-category data.

The QBag [1] may have a serious bias for computing p-value in the multi-category classification, when the decision boundary is represented by the piecewise linear function. As shown in Sect. 4.1, the multiscale bagging corrects the bias, and provides more preferable input points to boost the accuracy of the classifier.

This paper is only the first attempt to work on multiscale bagging, and further analysis would be needed in both theory and numerical examples.

References

[1] N. Abe and H. Mamitsuka, “Query learning strategies using boosting and bagging,” International Conference on Machine Learning, pp.1–9, 1998.
[2] P. Bickel, F. Götze, and W. van Zwet, “Resampling fewer than n observations: Gains, losses and remedies for losses,” Statistica Sinica, vol.7, pp.1–31, 1997.
[3] L. Breiman, “Bagging predictors,” Mach. Learn., vol.24, pp.123–140, 1996.
[4] B. Efron, “Bootstrap methods: Another look at the jackknife,” Annals of Statistics, vol.7, pp.1–26, 1979.
[5] B. Efron and R. Tibshirani, “The problem of regions,” Annals of Statistics, vol.26, pp.1687–1718, 1998.
[6] J. Felsenstein, “Confidence limits on phylogenies: An approach using the bootstrap,” Evolution, vol.39, pp.783–791, 1985.
[7] T.M. Mitchell, “Generalization as search,” Artif. Intell., vol.18, no.2, pp.203–226, 1982.
[8] D. Politis and J. Romano, “Large sample confidence regions on subsamples under minimal assumptions,” Annals of Statistics, vol.22, pp.2031–2050, 1994.
[9] H. Shimodaira, “An approximately unbiased test of phylogenetic tree selection,” Systematic Biology, vol.51, pp.492–508, 2002.
[10] H. Shimodaira, “Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling,” Annals of Statistics, vol.32, pp.2616–2641, 2004.
[11] H. Shimodaira, “Testing regions with nonsmooth boundaries via multiscale bootstrap,” J. Statistical Planning and Inference, vol.138, pp.1227–1241, 2008.
[12] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” J. Machine Learning Research, vol.2, pp.45–66, 2001.
Takafumi Kanamori is an Associate Professor at Nagoya University. He received his B.E. and M.E. degrees from University of Tokyo, and Ph.D. degree from the Graduate University for Advanced Studies, Japan, in 1994, 1996, and 1999, respectively. His research interests include statistics, machine learning, and optimization.

Masayoshi Aoki received his B.S. and M.S. from Tokyo Institute of Technology, in 2009 and 2011. His research interests include Statistics and machine learning.

Kouta Mine received his B.S. from Tokyo Institute of Technology in 2010. His research interests include Statistics and machine learning.