Lecture 19: Ensemble methods: Bagging CS 3780/5780, Sp25

Tushaar Gangavarapu (TG352@cornell.edu)

So far, all of our learning algorithms have been focused on *one* method to solve a specific problem. Now, we ask a simple question, "can the *crowd* be smarter than the participants in the crowd?" Simply put, instead of running a single model, we aggregate the outputs from several trained models. On the face of it, this seems like a reasonable thing to do; we will attempt to make this more concrete through our understanding of the bias-variance tradeoff.

1 Ensemble methods

To understand why (and when) we might benefit from ensembling, let us consider a toy example. Say we have m IID random variables, $X^{(j)}$s, each with variance $\text{Var}[X^{(j)}] = \sigma^2$, we realize that the variance of the mean of $X^{(i)}$s is

$$\text{Var}\left[\frac{1}{m}\sum X^{(j)}\right] = \frac{\sigma^2}{m}.$$

The key takeaway here is that the more *independent* $X^{(j)}$s we factor in (i.e., m increases), the more the variance goes down. While this gives us a crude idea of what we expect to achieve—factor in more models would decrease the variance—it is quite optimistic in its assumption of independence. If instead, we only assume $X^{(j)}$s to be identically distributed, i.e., $X^{(j)}$s are pairwise correlated by a factor of ρ, then,

$$\begin{aligned}
\text{Var}\left[\frac{1}{m}\sum X^{(j)}\right] &= \frac{1}{m^2}\sum_k\sum_j \text{Cov}(X^{(j)}, X^{(k)}) \\
&= \frac{1}{m^2}\sum_k\sum_j \sigma^2\text{Corr}(X^{(j)}, X^{(k)}) \\
&= \frac{\sigma^2}{m^2}(m + \underbrace{(m^2 - m)}_{j\neq k}\rho) \\
&= \frac{\sigma^2}{m} + \sigma^2\rho - \frac{\sigma^2}{m}\rho) \\
&= \sigma^2\rho + (1-\rho)\frac{\sigma^2}{m}.
\end{aligned}$$

Observe that when $X^{(j)}$s are fully correlated, i.e., $\rho = 1$, we have the variance of the mean to be σ^2—averaging m completely correlated variables results in the same variance as $X^{(j)}$s. On the other hand, if $X^{(j)}$s are completely decorrelated, i.e., $\rho = 0$, we have the variance of the mean to be σ^2/m, which is same as when we assumed independence of random variables.

Now, if we consider each random variable to be the error of a given model, then increasing the number of models (increase in m) and decrease in the correlation between the models (decrease in ρ) drives the variance in the error of the ensemble to go down. Some ways of achieving this are:

(a) Train different algorithms—say, naive Bayes, logistic regression, SVM—on the training data; the downside being you would have to end up spending computational resources to train each individual model.

(b) Use different training datasets to train multiple models, one per dataset. The downside

of this being collecting additional datasets can be impractical, if not infeasible, in certain situations.

In this lecture, we will focus on *bagging*, which is akin to collecting multiple datasets, without actually requiring additional data collection. A popular example of bagging, which we will explore later, is the random forests classifier. While we motivated ensembling from the perspective of variance reduction, there is a complementary approach that focuses on bias reduction: boosting—of which AdaBoost and XGBoost are popular examples—which we will discuss in the next lecture.

2 Bagging: Bootstrap aggregation

Bootstrapping. Say, we have a true distribution $\mathcal{P}$, from which our training dataset $\mathcal{D}$ was sampled, i.e., $(x^{(j)}, y^{(j)}) \in \mathcal{D} \sim \mathcal{P}$. Ideally, for our "use different training sets" approach, we would have to draw $\mathcal{D}^{(1)} \sim \mathcal{P}, \mathcal{D}^{(2)} \sim \mathcal{P}, \ldots$ and then train $h^{(1)}, h^{(2)}, \ldots$ on each of these datasets. Since this is impractical, what "bootstrapping" offers is: assume $\mathcal{P} = \mathcal{D}$ and now, *under this assumption*, drawing samples from $\mathcal{P}$ is the same as drawing samples from $\mathcal{D}$.

More precisely, we generate a bootstrap set Z, by drawing samples with replacement from $\mathcal{D}$; $Z \sim \mathcal{D}$ and $|Z| = |\mathcal{D}|$ $(= n)$. It is important to stress here that we sample *with replacement*, which allows our assumption of treating the empirical distribution $\mathcal{D}$ as the true distribution $\mathcal{P}$ to hold. Now, we can generate many such bootstrap samples (say m), $Z^{(1)}, Z^{(2)}, \ldots, Z^{(m)}$.

Aggregation. Now, we can train a machine learning model on $Z^{(1)}, Z^{(2)}, \ldots, Z^{(m)}$ to obtain $h^{(1)}, h^{(2)}, \ldots, h^{(m)}$. Following this, we can define our aggregate predictor as:

$$h(x) = \frac{1}{m} \sum_{j=1}^{m} h^{(j)}(x).$$

For classification, this can be majority voting, instead of an average.

2.1 Bias-variance analysis

Recall from our earlier discussion that the variance of the average error of m correlated models is,

$$\text{Var}\left[\frac{1}{m} \sum_{j=1}^{m} X^{(j)}\right] = \sigma^2 \rho + (1 - \rho)\frac{\sigma^2}{m}.$$

Observe that bagging creates less correlated models that what would've been, had we trained them repeatedly on $\mathcal{D}$, thereby decreasing ρ, which in turn decreases the "$\sigma^2\rho$" term. Additionally, since we can draw as many bootstrap sets as needed, we can increase m to drive down the second, "$\cdots/m$" term.[1] Finally, we also note that the variance is upper-bounded by σ^2 (when all models are fully correlated), i.e., ensembling cannot lead to additional overfitting than the individual models.

This might seem magical at first: we're getting variance reduction for free by simply sampling (with replacement) from $\mathcal{D}$ and aggregating different models trained on these bootstrap sets. Well, as they say, there's no such thing as a free lunch; since each model is trained on a subset of the entire data, we are increasing the bias of the individual models. However, in practice, it

[1] This in no way is a silver bullet; it stands to reason that there is some lower bound on the number of bootstrap sets we can draw, before we observe no gains.

so happens that the decrease in variance often outweighs the increase in bias. If it helps, you can think of subjecting low-bias-high-variance models to bagging-based ensembling.

2.2 Out-of-bag error

Starting with a dataset of n instances, if we are sampling with replacement, we have with probability $1 - 1/n$ of not selecting a specific instance. This follows that, with probability $(1 - 1/n)^n$, we don't select a specific instance in a bootstrap set. As it turns out, we have

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.37.$$

So, in every bootstrap set, about 37% of the original training dataset remains unused, which can be used to estimate the error of the underlying model, often called the out-of-bag error. As the number of bootstrap sets, m, increases, each instance would get evaluated multiple times as "held-out," very similar to leave-one-out cross-validation. In the limit, $m \to \infty$, out-of-bag error converges to that of leave-one-out, and is (much) cheaper to compute than leave-one-out.

3 (Almost) random forest

From our discussion on decision trees, we note that decision trees are low bias and high variance models, which makes them a perfect fit for bagging. Bagging with decision trees is "almost" random forest, but before we get to random forests, let us outline some properties of this "almost" random forest algorithm, a.k.a., bagged trees.

It is fairly obvious to realize that bagged trees allow for handling missing features. If a specific feature is missing for a test point, we can exclude those trees in the ensemble that employ splitting over the missing feature. Even so, there might be cases where most, if not all, trees include splitting over a specific, powerful, attribute. Nonetheless, at least in some cases, we are able to handle missing features.

One major upside to using decision trees was the high degree of interpretability they offered, and with bagged trees, we lose that. One way of understanding which features are important is to look at the average information gain offered by a specific feature across the trees in the ensemble. Note: Since decision trees are built greedily, this is not the same as asking how much the performance would degrade if we excluded the feature.

3.1 Random forests

To motivate the failure of bagged trees, consider a feature that is a strong predictor (say, latitude in our ski example); it is highly likely that most trees would involve splitting on this feature, which in turn drives up the correlation ρ between the trees.

Random forests add additional randomness over bagged trees by only allowing a subset of k randomly-chosen features (often $k \overset{\text{set}}{=} \sqrt{d}$ or $\log_2 d$) to be used at each split. This further de-correlates the individual predictors, which further reduces ρ, in turn reducing the overall variance. Since we are restricting the feature space, there is an increase in bias (more than that for decision trees), but again, the decrease in variance often outweighs the increase in bias.

4 Conclusion

In this lecture, we saw bagging as a way of reducing variance, where we aggregated predictions from several models trained on bootstrapped sets of the original training data. While we noted

several benefits of bagging: variance reduction, (possibly) better performance, free validation set, and ease of handling missing features (specifically for decision trees), there are some downsides to bagging, including, increase in bias, being not as easily interpretable as decision trees, and more expensive. Finally, it is also easy to realize that even with bagging, we are unable to model additive structures.

In the next lecture, we will see a different ensembling approach: *boosting*, which focuses on bias reduction and allows us to model additive structures.

A Notation

m	The number of classifiers being ensembled (or, the number of random variables under consideration)
$X^{(j)}$	The j-th random variable (or, the error of the j-th predictor)
$\mathrm{Var}[X]$ or σ_X^2	The variance of a random variable X
$\mathrm{Cov}(X, Y)$	The covariance of two random variables X and Y
$\mathrm{Corr}(X, Y)$ or ρ_{XY}	The correlation between two random variables X and Y; computed as $\mathrm{Cov}(X, Y)/\sigma_X\sigma_Y$
$\mathcal{D}$	The training dataset of n samples
n	The number of training samples in the dataset $\mathcal{D}$
$\mathcal{P}$	The true distribution from which the data samples are drawn; we write $(x^{(j)}, y^{(j)}) \in \mathcal{D} \sim \mathcal{P}$ to indicate that all instances of $\mathcal{D}$ are sampled from $\mathcal{P}$
$Z^{(j)}$	The j-th bootstrapped set generated by sampling n times from $\mathcal{D}$, with replacement
$h^{(j)}$	The predictor resultant from training a model on the j-th bootstrap set
h	The aggregate hypothesis obtained from averaging or majority voting of individual bootstrap predictors
k	The number of features sampled at each split, in random forests; often set to $\sqrt{d}$ or $\log_2 d$

References

S. George. CS229 Lecture notes: Decision Trees. *CS229 Lecture notes*, 1(1):6–7, 2023. URL https://cs229.stanford.edu/cs229-notes-decision_trees.pdf#page=10.70.

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL http://probml.github.io/book1. See Chapter 18, §18.2–§18.4; pp. 608–611.

R. J. L. Townshend. CS229 Lecture notes: Decision Trees. *CS229 Lecture notes*, 1(1): 1–4, 2018. URL https://cs229.stanford.edu/notes_archive/cs229-notes-ensemble.pdf.

(Last compiled: 4/10/2025, 4.09pm ET.)