

Model combination

- The winner-takes-all approach is intuitively the approach which should work the best.
- However, recent results in machine learning show that the performance of the final model can be improved not by choosing the model structure which is expected to predict the best but by creating a model whose output is the combination of the output of models having different structures.
- The reason is that in reality any chosen hypothesis $h(\cdot, \alpha_N)$ is only an estimate of the real target and, like any estimate, is affected by a bias and a variance term.
- See the theoretical results on the combination of estimators.

Combination of two estimators

Consider two unbiased estimators $\hat{\theta}_1$ and $\hat{\theta}_2$ of the same parameter θ

$$E[\hat{\theta}_1] = \theta \quad E[\hat{\theta}_2] = \theta$$

having the same variance

$$\text{Var}[\hat{\theta}_1] = \text{Var}[\hat{\theta}_2] = v$$

and being uncorrelated, i.e. $\text{Cov}[\hat{\theta}_1, \hat{\theta}_2] = 0$.

Combination of two estimators(II)

Let $\hat{\theta}_{\text{cm}}$ be the combined estimator

$$\hat{\theta}_{\text{cm}} = \frac{\hat{\theta}_1 + \hat{\theta}_2}{2}$$

This estimator has the nice properties of being unbiased

$$E[\hat{\theta}_{\text{cm}}] = \frac{E[\hat{\theta}_1] + E[\hat{\theta}_2]}{2} = \theta$$

and with a reduced variance

$$\text{Var}[\hat{\theta}_{\text{cm}}] = \frac{\text{Var}[\hat{\theta}_1] + \text{Var}[\hat{\theta}_2]}{4} = \frac{v}{2}$$

This trivial statistic computation shows that the simple average of two unbiased estimators with a non zero variance returns a combined estimator with reduced variance.

Combination of m estimators

Here, we report the general formula of the linear combination of a number m of estimators. Assume we want to estimate the unknown parameter θ by combining of a set of m estimators $\{\hat{\theta}_j\}$, $j = 1, \dots, m$.

Let

$$E[\hat{\theta}_j] = \mu_j \quad \text{Var}[\hat{\theta}_j] = v_j \quad \text{Bias}[\hat{\theta}_j] = b_j$$

be the expected values, the variances and the bias of the m estimators, respectively.

We are interested in estimating θ by forming a linear combination

$$\hat{\theta}_{\text{cm}} = \sum_{j=1}^m w_j \hat{\theta}_j = w^T \hat{\theta} \quad (1)$$

where $\hat{\theta} = [\hat{\theta}_1, \dots, \hat{\theta}_m]^T$ is the vector of estimators and $w = [w_1, \dots, w_m]^T$ is the weighting vector.

The mean-squared error of the combined system is

$$\begin{aligned}\text{MSE} &= E[(\hat{\boldsymbol{\theta}}_{\text{cm}} - \theta)^2] = E[(w^T \hat{\boldsymbol{\theta}} - E[w^T \hat{\boldsymbol{\theta}}])^2] + (E[w^T \hat{\boldsymbol{\theta}}] - \theta)^2 \\ &= E[(w^T (\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}]))^2] + (w^T \boldsymbol{\mu} - \theta)^2 = \\ &= w^T \boldsymbol{\Omega} w + (w^T \boldsymbol{\mu} - \theta)^2\end{aligned}$$

where $\boldsymbol{\Omega}$ is a $[m \times m]$ covariance matrix whose ij^{th} term is

$$\boldsymbol{\Omega}_{ij} = E[(\hat{\boldsymbol{\theta}}_i - \mu_i)(\hat{\boldsymbol{\theta}}_j - \mu_j)]$$

and $\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)^T$ is the vector of expected values.

The MSE of the combined estimator is minimized for

$$w^* = (\boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{\Omega})^{-1} \theta \boldsymbol{\mu}$$

Linear constrained combination

A commonly used constraint is

$$\sum_{j=1}^m w_j = 1, \quad w_j \geq 0, \quad j = 1, \dots, m$$

Let us write w as

$$w = (u^T g)^{-1} g$$

where $u = (1, \dots, 1)^T$ is an m -dimensional vector of ones, $g = (g_1, \dots, g_m)^T$ and $g_j > 0, \forall j = 1, \dots, m$.

The constraint can be enforced in minimizing the MSE by using the Lagrangian function

$$L = w^T \Omega w + (w^T \mu - \theta)^2 + \lambda(w^T u - 1)$$

with λ Lagrange multiplier.

The optimum is achieved if we set

$$g^* = [\Omega + (\mu - \theta u)(m - \theta u)^T]^{-1}u$$

For these optimal weights

$$\min_w E[(\hat{\theta}_{\text{cm}} - \theta)^2] = \frac{1}{u^T (\Omega + (\mu - \theta u)(\mu - \theta u)^T)^{-1}u}$$

Note that the combined estimator is unbiased if the individual estimators are unbiased, which is the main reason for employing the constraint $\sum_{j=1}^m w_j = 1$. With unbiased estimators we obtain

$$g^* = \Omega^{-1}u$$

and with uncorrelated estimators

$$g_j^* = \frac{1}{V_j} \quad j = 1, \dots, m$$

Combining models

- The previous approach requires the knowledge of the correlation among the different estimators. Sometimes this quantity is not available. Alternative methods have been proposed.
- Suppose we have m distinct predictors $h_j(\cdot, \alpha_N)$, $j = 1, \dots, m$ obtained from the initial training set D_N . For example one of this predictor could be a linear model fit on some subset of the variables or a neural network or a regression tree.
- We want to find how to best linearly combine these estimates in order to have an estimator that is better than any of the individual ones

$$\sum_{j=1}^m w_j h_j(\cdot, \alpha_N)$$

Combining models (II)

- One way to obtain estimates of w_j is to perform a least-squares regression of the output y on the m inputs $h_j(\cdot, \alpha_N)$.
- The training set for this regression is then made by $D_N = \{h_i, y_i\}$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} = \begin{bmatrix} h_1(x_1, \alpha_N) & h_2(x_1, \alpha_N) & \dots & h_m(x_1, \alpha_N) \\ h_1(x_2, \alpha_N) & h_2(x_2, \alpha_N) & \dots & h_m(x_2, \alpha_N) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N) & h_2(x_N, \alpha_N) & \dots & h_m(x_N, \alpha_N) \end{bmatrix}$$

where $h_i, i = 1, \dots, N$ is a vector of m terms.

- Once computed the least-squares solution $\hat{\beta}$ the combined estimator is

$$h_{\text{cm}}(x) = \sum_{j=1}^m \hat{\beta}_j h_j(x, \alpha_N)$$

Combining models (III)

- The simple least-squares approach might produce poor results because it does not take into account the correlation of the h_j induced by the fact that all of them are estimated on the same training set D_N .
- Wolpert (1992) presented an interesting idea, called stacked generalization for combining estimators without suffering of the correlation problem.
- This proposal was translated in statistical language by Breiman, in 1993.
- It was the stacked regression

Stacked regression

- We want to estimate the function $f(\cdot)$ by combining a collection of m different models $\{h_j(\cdot)\}$, $j = 1, \dots, m$.
- The m models could have been obtained by training on different subsets of data or by using different techniques.
- **Stacked regression** combines linearly the m models

$$\sum_{j=1}^m \hat{\beta}_j h_j(x)$$

- How to compute the set of parameters $\hat{\beta}_j$, $j = 1, \dots, m$?

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^N \left(y_i - \sum_{j=1}^m \beta_j h_j^{(-i)}(x_i) \right)^2$$

where $h_j^{(-i)}(x_i)$ is the leave-one-out estimate of the j th model.

- In other terms the parameters are obtained by performing a least-squares regression of the output y on the m inputs $h_j(\cdot, \alpha_N^{(-i)})$.
- The training set for this regression is then made by $D_N = \{h_i^-, y_i\}, i = 1, \dots, N$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1^- \\ h_2^- \\ \vdots \\ h_N^- \end{bmatrix} = \begin{bmatrix} h_1(x_1, \alpha_N^{(-1)}) & h_2(x_1, \alpha_N^{(-1)}) & \dots & h_m(x_1, \alpha_N^{(-1)}) \\ h_1(x_2, \alpha_N^{(-2)}) & h_2(x_2, \alpha_N^{(-2)}) & \dots & h_m(x_2, \alpha_N^{(-2)}) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N^{(-N)}) & h_2(x_N, \alpha_N^{(-N)}) & \dots & h_m(x_N, \alpha_N^{(-N)}) \end{bmatrix}$$

where $h_j(x_i, \alpha_N^{(-i)})$ is the predicted outcome in x_i of the j th model trained on D_N with the i th sample (x_i, y_i) set aside.

Considerations on stacked regression

- By using the cross-validated predictions $h_j(x_i, \alpha_N^{(-i)})$ stacked regression avoids giving unfairly high weight to models with higher complexity.
- It was shown by Breiman, that the performance of the stacked regressor improves when the coefficients $\hat{\beta}$ are constrained to be non negative.
- There is a close connection between stacking and winner-takes-all model selection. If we restrict the minimization to weight vectors w that have one unit weight and the rest zero, this leads to the model choice returned by the winner-takes-all based on the leave-one-out.
- Rather than choose a single model, stacking combines them with estimated optimal weights. This will often lead to better prediction, but less interpretability than the choice of only one of the m models.

Unstable learners

- A learning algorithm is informally called *unstable* if *small* changes in the training data lead to significantly different models and relatively *large* changes in accuracy.
- Unstable learners can have low bias but have typically high variance.
- Unstable methods can have their accuracy improved by *perturbing* (i.e. generating multiple versions of the predictor by perturbing the training set or learning method) and *combining*. Breiman call these techniques P&C methods.
- Combining multiple estimator is a variance reduction technique.
- The *bagging* technique is a P&C technique which aims to improve accuracy for unstable learners by averaging over such discontinuities.
- The philosophy of bagging is to improve the accuracy by reducing the variance.

Bagging

- Consider a dataset D_N and a learning procedure to build an hypothesis α_N from D_N .
- The idea of **bagging** or **bootstrap aggregating** is to imitate the stochastic process underlying the realization of D_N .
- A set of B repeated bootstrap samples $D_N^{(b)}$, $b = 1, \dots, B$ are taken from D_N .
- A model $\alpha_N^{(b)}$ is built for each $D_N^{(b)}$.
- A final predictor is built by aggregating the B models $\alpha_N^{(b)}$.
- In the regression case the bagging predictor is

$$h_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B h(x, \alpha_N^{(b)})$$

- In the classification case a majority vote is used.

R TP: Bagging against overfitting

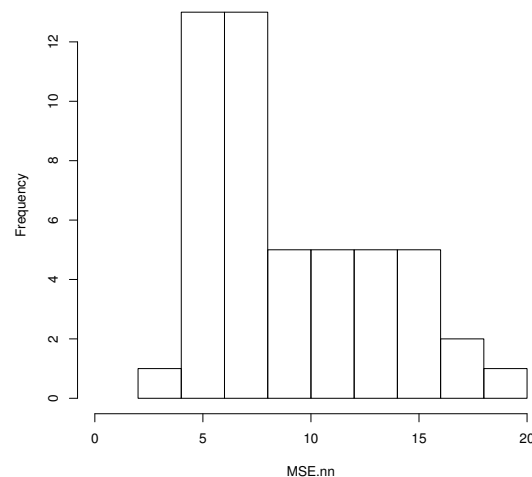
- Consider a dataset $D_N = \{x_i, y_i\}$, $i = 1, \dots, N$ of $N = 100$ i.i.d. normally distributed inputs $\mathbf{x} \sim \mathcal{N}([0, 0, 0], I)$.
- Suppose that y is linked to \mathbf{x} by the input/output relation

$$y = x_1^2 + 4 \log(|x_2|) + 5x_3 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 0.25)$ represents the noise.

- Let's train a single-hidden-layer neural network with $s = 15$ hidden neurons on the training set.
- The prediction accuracy on the test set ($N_{ts} = 100$) is $\widehat{\text{MSE}}_{ts} = 9.95$.

- Let us apply a bagging combination with $B = 50$ (R-file `bagging.r`).
- The prediction accuracy on the test set of the bagging predictor is $\widehat{\text{MSE}}_{\text{ts}} = 2.26$. This shows that the bagging combination reduces the overfitting of the single neural network.
- Below there is the histogram of the $\widehat{\text{MSE}}_{\text{ts}}$ accuracy of each bootstrap repetition. We can see that the performance of the bagging predictor is much better than the average performance.



Considerations

- Tests on real and simulated datasets showed that bagging can give a substantial gain in accuracy.
- The vital element is the instability of the prediction method.
- If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.
- On the other hand it can slightly degrade the performance of stable procedures.
- There is a cross-over point between instability and stability at which bagging stops improving.

Considerations (II)

- How many bootstrap replicates? In his experiments Breiman suggests that $B \approx 50$ is a reasonable figure.
- Bagging demands the repetition of B estimations of $h(\cdot, \alpha_N^{(b)})$ but avoids the use of expensive validation techniques (e.g. cross-validation).
- Bagging is an ideal procedure for parallel computing. Each estimation of $h(\cdot, \alpha_N^{(b)})$, $b = 1, \dots, B$ can proceed independently of the others.
- Bagging is a relatively easy way to improve a existing method. It simply needs adding
 1. a loop that selects the bootstrap sample and sends it to the learning machine and
 2. a back end that does the agregation.
- Note however that if the original learning machine has an interpretable structure (e.g. classification tree) this is lost for the

Boosting methods

- Boosting is one of the most powerful learning ideas introduced in the last ten years.
- Boosting is a general method which attempts to *boost* the accuracy of any given learning algorithm.
- It was originally designed for classification problems but it can profitably be extended to regression as well.
- Boosting [Freund Schapire1996,Schapire1990] encompasses a family of methods.
- The focus of boosting methods is to produce a series of “weak” learners in order to produce a powerful combination.
- A *weak learner* is a learner that has accuracy only slightly better than chance.

Boosting methods (II)

- The training set used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series.
- Examples that are incorrectly predicted by previous classifiers in the series are chosen more often than examples that were correctly predicted.
- Thus Boosting attempts to produce new classifiers that are better able to predict examples for which the current ensemble's performance is poor.
- Unlike Bagging, the resampling of the training set is **dependent** on the performance of the earlier classifiers.
- The two most important types of boosting algorithms are the Ada Boost (Adaptive Boosting) algorithm (Freund, Schapire, 1997) and the Arcing algorithm (Breiman, 1996).

The Ada Boost algorithm

- Consider a classification problem where the output can take as value one of the two classes $\{-1, 1\}$. Let D_N be the training set.
- A classifier is a predictor $h(\cdot)$ which given an input x , produces a prediction taking one of the values $\{-1, 1\}$.
- A *weak classifier* is one whose misclassification error rate is only slightly better than random guessing.
- The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of classifiers $h_j(\cdot)$, $j = 1, \dots, m$.
- The predictions of the m *weak* classifiers are then combined through a **weighted majority vote** to produce the final prediction

$$h_{\text{boo}} = \text{sign} \left(\sum_{j=1}^m \alpha_j h_j(x, \alpha_N) \right)$$

The Ada Boost algorithm (II)

- The weights α_j of the different classifiers are computed by the algorithm. The idea is to give higher influence to the more accurate classifiers in the sequence.
- At each step, the boosting algorithm samples N times from a distribution w on the training set which put a weight w_i on each sample (x_i, y_i) , $i = 1, \dots, N$ of D_N
- Initially the weights are all set to $w_i = 1/N$ so that the first step simply trains the classifier in the standard manner.
- For each successive iteration $j = 1, \dots, m$ the probability weights are individually modified and the classification algorithm is re-applied to the resampled training set.

The Ada Boost algorithm (III)

- At a generic step j these observations that were misclassified by the classifier $h_{j-1}(\cdot)$ trained at the previous step, have their weights w_i increased, whereas the weights are decreased for those that were classified correctly.
- The rationale is that, as iterations proceed, observations that are difficult to correctly classify receive ever-increasing influence and the classifier is forced to concentrate on them.
- Note the presence in the algorithm of **two types of weights**: the weights $\alpha_j, j = 1, \dots, m$ that measure the importance of the classifiers and the weights $w_i, i = 1, \dots, N$ of the samples.
- Weak learners are added until some desired low training error has been achieved.

Ada Boost in detail

1. Initialize the observation weights $w_i = 1/N, i = 1, \dots, N$.
2. For $j = 1$ to m :
 - (a) Fit a classifier $h_j(\cdot)$ to the training data obtained by resampling D_N using weights w_i .
 - (b) Compute the misclassification error on the training set

$$\widehat{\text{MME}}_{\text{emp}}^{(j)} = \frac{\sum_{i=1}^N w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^N w_i}$$

- (c) Compute

$$\alpha_j = \log\left(\frac{1 - \widehat{\text{MME}}_{\text{emp}}^{(j)}}{\widehat{\text{MME}}_{\text{emp}}^{(j)}}\right)$$

Note that $\alpha_j > 0$ if $\widehat{\text{MME}}_{\text{emp}}^{(j)} \leq 1/2$ (otherwise we stop or we restart) and that α_j gets larger as $\widehat{\text{MME}}_{\text{emp}}^{(j)}$ gets smaller.

Ada Boost in detail (II)

2. (d) For $i = 1, \dots, N$ set

$$w_i \leftarrow w_i \begin{cases} \exp[-\alpha_j] & \text{if correctly classified} \\ \exp[\alpha_j] & \text{if incorrectly classified} \end{cases}$$

(e) The weights are normalized to ensure that w_i represents a true distribution.

3. Output of the weighted majority vote

$$h_{\text{boo}} = \text{sign} \left(\sum_{j=1}^m \alpha_j h_j(x, \alpha_N) \right)$$

R TP: Ada Boost for classification

- Consider the medical dataset *Pima* obtained by a statistical survey on women of Pima Indian heritage. This dataset reports the presence of diabetes in Pima indian women together with other clinical measures (blood pressure, insulin, age,...).
- The classification task is to predict the presence of diabetes as a function of the clinical measures.
- We consider a training set of $N = 40$ and a test set of 160 samples.
- The classifier is a simple classification tree which returns a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.36$.
- We use a boosting procedure with $m = 15$ to improve the performance of the weak classifier. The misclassification rate of the boosted classifier is $\widehat{\text{MME}}_{\text{ts}} = 0.24$.
- R file `boosting.r`.

Theoretical foundation of boosting

- Boosting has its roots in a theoretical framework for studying machine learning called the PAC learning model.
- Freund and Scapire proved that the empirical error of the final hypothesis h_{boo} is at most

$$\prod_{j=1}^m \left[2\sqrt{\widehat{\text{MME}}_{\text{emp}}^{(j)} * (1 - \widehat{\text{MME}}_{\text{emp}}^{(j)})} \right]$$

- They showed also how to bound the generalization error.

The arcing algorithm

- This was proposed as a modification of the original Ada Boost algorithms by Breiman.
- It is based on the idea that the success of boosting is related to the adaptive resampling property where increasing weight is placed on those samples more frequently misclassified.
- ARCing stands for *Adaptive Resampling and Combining*.
- The complex updating equations of Ada Boost are replaced by much simpler formulations.
- The final classifier is obtained by unweighted voting

ARcing in detail

1. Initialize the observation weights $w_i = 1/N, i = 1, \dots, N$.
2. For $j = 1$ to m :
 - (a) Fit a classifier h_j to the training data obtained by resampling D_N using weights w_i .
 - (b) Let e_i the number of misclassifications of the i th sample by the j classifiers h_1, \dots, h_j .
 - (c) The updated weights are defined by

$$w_i = \frac{1 + e_i^4}{\sum_{i=1}^N (1 + e_i^4)}$$

3. The output is obtained by unweighted voting of the m classifiers h_j .

R example: ARcing for classification

- Consider the medical dataset *Breast Cancer* obtained by Dr. William H. Wolberg (physician) at the University of Wisconsin Hospital in USA. This dataset reports the class of cancer (malignant and benign) together with other properties (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion,...)
- The classification task is to predict the class of the breast cancer as a function of the clinical measures.
- We consider a training set of $N = 400$ and a test set of 299 samples.
- The classifier is a simple classification tree which returns a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.063$.
- We use an arcing procedure with $m = 15$. It gives a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.016$.
- R file `arcining.r`

Some considerations

- Boosting is a very recent technique. According to most researchers, it is one of the most promising techniques. However, it is probably too early to have a thorough assessment of the technique.
- Boosting techniques are very simple and easy to program. Moreover, they have few parameters (e.g. max number of classifiers) to tune.
- They advocate a shift in the attitude of the learning-system designer: instead of trying to design a learning algorithm which should be accurate over the entire space, he can instead focus on finding weak algorithms that only need to be better than random.
- A nice property of Ada Boost is its ability to identify outliers (hard samples).

Bagging and boosting

- Like bagging, boosting avoid the cost of heavy validation procedures.
- Like bagging, boosting trades accuracy for interpretability.
- As for bagging, the main effect of boosting is to reduce variance. It works effectively for high variance classifiers.
- Unlike bagging, boosting cannot be implemented in parallel, since it is based on a sequential procedure.
- Boosting seems to do better than bagging (see Breiman's work)

Some caveats

- The actual performance of boosting on a particular problem is clearly dependent on the data and the weak learner.
- Boosting can fail to perform well given
 - insufficient data,
 - overly complex weak hypothesis
 - weak hypothesis that are too weak.