Machine Learning

Lecture 24: Boosting

Hiroshi Shimodaira

11 November 2022

Ver. 1.0

Questions you should be able to answer after this week

- Ensemble learning (committee method)
- Theoretical background of ensemble learning
- Bagging
- Boosting
- AdaBoost - training algorithm, optimisation problem, loss function
- Applications of boosting

Ensemble learning - committee method

- Single models: logistic regression, SVM, decision tree
- Mixture models: GMM
- Multiple models:

  $$f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}) \quad \cdots \text{base models / weak learners}$$

  - Model linear averaging

    $$y = F(\boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} f_i(\boldsymbol{x})$$

    $$y = F(\boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} \alpha_i \, f_i(\boldsymbol{x})$$

  - Training strategies for multiple models
    - Train each model separately
    - Train models one-by-one sequentially

Theoretical background for ensemble learning

A Committee model for a regression task

$$y = F(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\boldsymbol{x})$$

$$f_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \varepsilon_i(\boldsymbol{x})$$

$$E_{\text{avr}} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}\left[\{f_i(\boldsymbol{x}) - f(\boldsymbol{x})\}^2\right] = \frac{1}{M}\sum_{i=1}^{M} \mathbb{E}[\varepsilon_i(\boldsymbol{x})^2]$$

$$E_{\text{comm}} = \mathbb{E}\left[\{F(\boldsymbol{x}) - f(\boldsymbol{x})\}^2\right] = \mathbb{E}\left[\left\{\frac{1}{M}\sum_{i=1}^{M} f_i(\boldsymbol{x}) - f(\boldsymbol{x})\right\}^2\right] = \mathbb{E}\left[\left\{\frac{1}{M}\sum_{i=1}^{M} \varepsilon_i(\boldsymbol{x})\right\}^2\right]$$

(Assuming that the errors have zero mean and are uncorrelated)

$$= \frac{1}{M}\left\{\frac{1}{M}\sum_{i=1}^{M} \mathbb{E}[\varepsilon_i(\boldsymbol{x})^2]\right\} = \frac{1}{M} E_{\text{avr}}$$

Theoretical background for ensemble learning *(cont.)*

A Committee model of majority voting for a classification task

Base models $\{f_i(\boldsymbol{x})\}$ – binary classifiers with classification accuracy p.

$$y_i = \mathbb{1}(f_i(\boldsymbol{x}) > 0), \quad \text{where } y_i \in \{0, 1\}$$

Let S denote the number of votes for class 1,

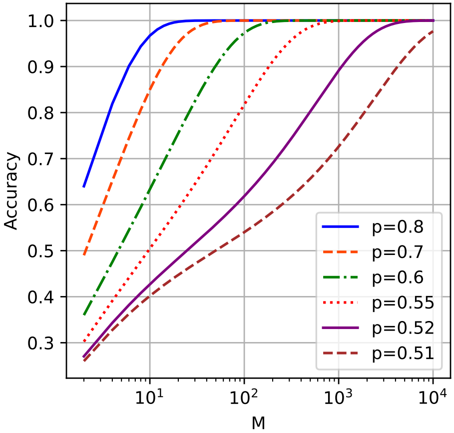
$$S = \sum_{i=1}^{M} y_i$$

Accuracy of the committee model:

$$Pr(S > M/2) = 1 - B(M/2, M, p)$$

where $B(k, n, p)$ is the cdf of the binomial distribution of k with parameters n and p.

Background theories for ensemble learning (*cont.*)



Accuracy of the committee model

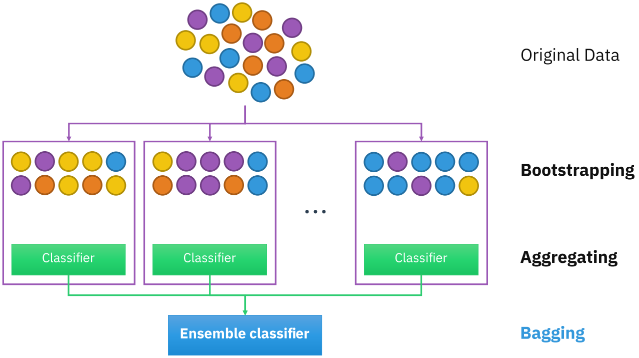
NB: strong assumption - each base model makes independent errors

Brief history of committee methods

1992	Stacking (*Wolpert*)
1994	Bagging (*Brieman*)
1995	AdaBoost (*Freund & Schapire*)
	Random Forests (*Tin Kam Ho*)
1997	Speech recognition with ROVER (*Fiscus*)
2001	Face detection with AdaBoost (*Viola & Jones*)

Bagging – bootstrap aggregating L. Brieman, 1994, 1996

- Train each $f_i(\boldsymbol{x})$ on a training dataset D_i of size n - sampled from the original data set D uniformly and with replacement.
- Employ the same training algorithm over all $\{f_i(\boldsymbol{x})\}$



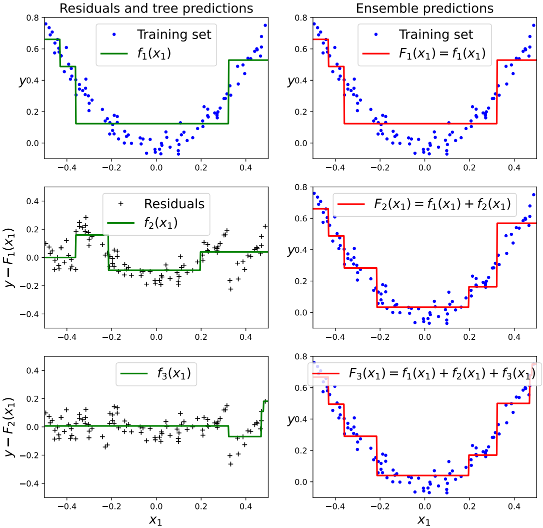
After Bootstrap aggregating (bagging) method of Wikimedia common

Boosting

$$F(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{x};\boldsymbol{\theta}_i)$$

$$\min_{\boldsymbol{\theta},\boldsymbol{\alpha}} L(F) = \min_{\boldsymbol{\theta},\boldsymbol{\alpha}} \sum_{i=1}^{n} \ell(y_i, F(\boldsymbol{x}_i;\boldsymbol{\theta}))$$

- $f_m \in \{-1,+1\}\,,\ m = 1,\ldots,M$
- No closed-form solutions normally
- Fit additive models sequentially



Adapted from `boosted_regr_trees.ipynb`

AdaBoost – adaptive boosting (Freund, Schapire 1995)

Training data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, Classifiers: $\{f_1(\boldsymbol{x}; \boldsymbol{\theta}_1), \ldots, f_M(\boldsymbol{x}; \boldsymbol{\theta}_M)\}$

Step 1 Initialise the weights $\mathbf{w} = (w_1, \ldots, w_n)$, where $w_i = \frac{1}{n}$.

Step 2 For $m = 1$ to M:

(a) Fit a classifier $f_m(\boldsymbol{x})$ to the training data using $\mathbf{w}$.

(b) Compute: $\mathrm{err}_m = \dfrac{\sum_{y_i \neq f_m(\boldsymbol{x}_i)} w_i}{\sum_{i=1}^{n} w_i}$

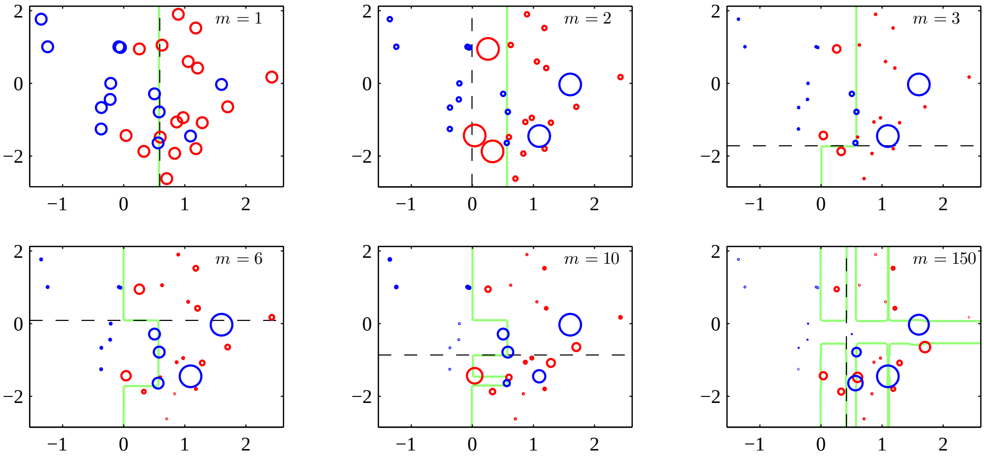
(c) Compute: $\alpha_m = \dfrac{1}{2} \ln\left(\dfrac{1 - \mathrm{err}_m}{\mathrm{err}_m}\right)$ $\cdots$ cf. logit function

(d) Update the weights: $w_i \leftarrow w_i \, e^{\{\alpha_m \, \mathbb{1}(y_i \neq f_m(\boldsymbol{x}_i))\}}$, $i = 1, \ldots, n$

Step 3 Output the final model: $F(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{m=1}^{M} \alpha_m \, f_m(\boldsymbol{x})\right)$

AdaBoost: example

Weak learners: one-level decision trees (i.e. one nodes) – "decision stumps".



From Figure 14.2 of C.Bishop's Pattern Recognition and Machine Learning

Optimisation problem in AdaBoost

Assume we have trained $f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x})$ and obtained $\alpha_1, \ldots, \alpha_{m-1}$.

$$F_{m-1}(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \cdots \alpha_{m-1} f_{m-1}(\boldsymbol{x})$$
$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \alpha_m f_m(\boldsymbol{x})$$

We now want to train $f_m(\boldsymbol{x})$ and estimate α_m with the exponential loss function L_m defined below.

$$\begin{aligned}
L_m &= \sum_{i=1}^{n} e^{-y_i F_m(\boldsymbol{x}_i)} \;=\; \sum_{i=1}^{n} e^{(-y_i F_{m-1}(\boldsymbol{x}_i) - y_i \alpha_m f_m(\boldsymbol{x}))} \quad \cdots \text{ convex function of } \boldsymbol{\alpha} \\
&= \sum_{i=1}^{n} w_{m,i}\, e^{(-y_i \alpha_m f_m(\boldsymbol{x}))}\,, \quad \text{where } w_{m,i} = e^{(-y_i F_{m-1}(\boldsymbol{x}_i))} \\
&= e^{-\alpha_m} \sum_{y_i = f_m(\boldsymbol{x})} w_{m,i} \;+\; e^{\alpha_m} \sum_{y_i \neq f_m(\boldsymbol{x})} w_{m,i} \\
&= \left(e^{\alpha_m} - e^{-\alpha_m}\right) \sum_{y_i \neq f_m(\boldsymbol{x})} w_{m,i} \;+\; e^{-\alpha_m} \sum_{i=1}^{n} w_{m,i} \quad \rightarrow \quad \min_{f_m} \sum_{y_i \neq f_m(\boldsymbol{x})} w_{m,i}
\end{aligned}$$

Optimisation problem in AdaBoost *(cont.)*

We now would like to minimise L_m with respect to α_m

$$\begin{aligned}\frac{\partial L_m}{\partial \alpha_m} &= \frac{\partial}{\partial \alpha_m}\left(e^{-\alpha_m}\sum_{y_i=f_m(\boldsymbol{x})} w_{m,i} + e^{\alpha_m}\sum_{y_i\neq f_m(\boldsymbol{x})} w_{m,i}\right)\\ &= -e^{-\alpha_m}\sum_{y_i=f_m(\boldsymbol{x})} w_{m,i} + e^{\alpha_m}\sum_{y_i\neq f_m(\boldsymbol{x})} w_{m,i} = 0\end{aligned}$$

$$e^{-\alpha_m}\sum_{y_i=f_m(\boldsymbol{x})} w_{m,i} = e^{\alpha_m}\sum_{y_i\neq f_m(\boldsymbol{x})} w_{m,i}$$

Taking logarithm yields:

$$-\alpha_m \ln\left(\sum_{y_i=f_m(\boldsymbol{x})} w_{m,i}\right) = \alpha_m \ln\left(\sum_{y_i\neq f_m(\boldsymbol{x})} w_{m,i}\right)$$

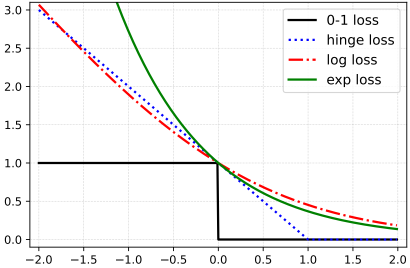
Hence

$$\alpha_m = \frac{1}{2}\ln\left(\frac{\sum_{y_i=F_m(\boldsymbol{x})} w_{m,i}}{\sum_{y_i\neq F_m(\boldsymbol{x})} w_{m,i}}\right) = \frac{1}{2}\ln\left(\frac{\left(\sum_{y_i=F_m(\boldsymbol{x})} w_{m,i}\right)/\sum_{i=1}^n w_{m,i}}{\left(\sum_{y_i\neq F_m(\boldsymbol{x})} w_{m,i}\right)/\sum_{i=1}^n w_{m,i}}\right) = \frac{1}{2}\ln\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right)$$

Exponential loss in AdaBoost

$$\ell(y, F(\boldsymbol{x})) = e^{-yF(\boldsymbol{x})}$$

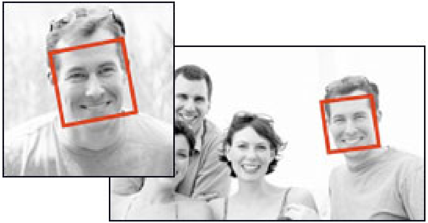
- Differentiable and an approximation of the ideal misclassification error function.
- Its sequential minimisation leads to the simple AdaBoost algorithm.
- It penalises large negative values of $yF(x)$, putting a lot of weight on misclassified samples $\rightarrow$ very sensitive to outliers (mislabelled examples)



Applications of boosting – face detection

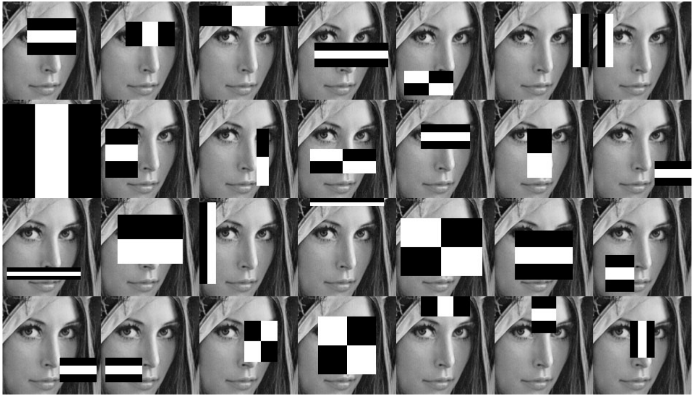
How would you detect a face?

(R. Vaillant, C. Monrocq and Y. LeCun, 1994)



How does album software tag your friends?

Viola–Jones Face detection (2001)



- Face detector consists of linear combination of 'weak' classifiers that utilise five types of primitive features.
- The detector is trained on a training data set of a large number of positive and negative samples.
- Scan the input image with a sub-window (24 x 24 pixels) to detect a face.

Viola & Jones' paper: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

A nice demo: http://vimeo.com/12774628

Other methods and software tools for boosting

- LogitBoost

$$L_m = \sum_{i=1}^{n} \log\left(1 + e^{-y_i F_m(\boldsymbol{x}_i)}\right)$$

- Gradient boosting
- Extreme gradient boosting – dominating approach for small, tabular data sets
  - XGBoost (eXtreme Gradient Boosting) [Tianqi Chen & Carlos Guestrin, 2016] - a software tool