Info 411
Machine Learning and Data Mining
Lecture 10: Combining Multiple Learners

Brendon J. Woodford

19-Sep, 2017

UNIVERSITY of OTAGO Te Whare Wānanga o Otāgo NEW ZEALAND
BUSINESS SCHOOL
Department of Information Science

Rationale

- No Free Lunch theorem: There is no algorithm that is always the most accurate
- Generate a group of base-learners which when combined has higher accuracy
- Different learners use different:
  - Algorithms
  - Hyperparameters
  - Representations (Modalities)
  - Training sets
  - Subproblems

Using Different Algorithms

- Combining base learners based on multiple algorithms
  - E.g., Mix parametric methods with non-parametric ones
- Free us from the burden / decision of choosing a "right" one

Using different hyper-parameters

- Initial centres/membership in k-means
- Different k in k-nearest neighbour classifier / predictor
- Different number of hidden units in Multi-layer Perceptrons (MLP)s
- Initial weights in neural models such as Self Organising Maps (SOM)s/MLPs
- Different kernels, c values in Support Vector Machine (SVM)
- ...

Using different 'views'

- Different representations of the same input object
- Different types of sensory data or features (sensor fusion)
- Examples:
  - Multi-modal speech recognition (audio recognition assisted by lip shape analysis)
  - Content-based image retrieval: using features of colour, texture and shape to assess the similarity of images
  - Better to combine classifier decisions rather than concatenating features:
    - Simple concatenation gives higher dimensionality and results in more complex systems that are usually harder to train

Using different training sets

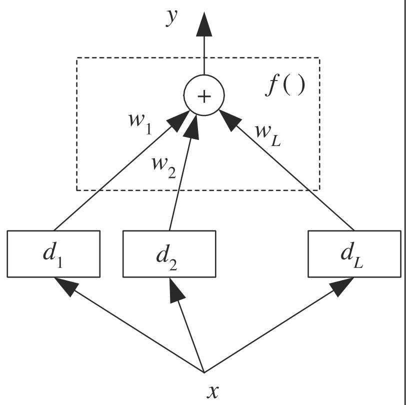
- Let weak base-learners each learn from a different input (sub)space
- Classifiers that are most "sure" will vote with more conviction
- Classifiers will be most "sure" about a particular part of the space On average, do better than a single classifier!

Weak Classifiers: Trade-Off

- Simple (a.k.a. weak) learners are good:
  - E.g., Naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - Low variance, don't usually overfit
- Simple (a.k.a. weak) learners are bad:
  - High bias, can't solve hard learning problems
- Often weak classifiers can be very useful!
  - How to make them work (positively)?

Voting[1]

- Linear combination:
  - $y = \sum_{j=1}^{L} w_j d_{ji}$
  - $w_j \geq 0$ and $\sum_{j=1}^{L} w_j = 1$
- Classification:
  - $y_i = \sum_{j=1}^{L} w_j d_{ji}$
- where:
  - L = number of base learners
  - d_{ji} = prediction of base learner $\mathcal{M}_j$ on input x
  - w_j = weighting of vote by learner d_j
  - $f()$ = function used to combine the outputs of d_j



[1] Sourced and reproduced from [Alpaydin;2010, p. 424].

What difference it makes

- From a Bayesian perspective where $w_j \equiv P(\mathcal{M}_j)$ & $d_{ji} = P(C_i|x, \mathcal{M}_j)$:

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j)P(\mathcal{M}_j)$$

- If d_j are i.i.d:

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L \cdot E[d_j] = E[d_j]$$

$$\mathrm{Var}(y) = \mathrm{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \mathrm{Var}(d_j) = \frac{1}{L}\mathrm{Var}(\mathrm{d_j})$$

Bias does not change & variance decreases by L

- If dependent, variance & error increase with positive correlation,

$$\mathrm{Var}(y) = \frac{1}{L^2}\mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}(d_j) + 2\sum_j \sum_{i<j} \mathrm{Cov}(d_j, d_i)\right]$$

Fixed Combination Rules[2]

Table: Classifier combination rules

Rule	Fusion function $f()$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \mathrm{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

[2] Sourced and adapted from [Alpaydin;2010, p. 425].

Fixed Combination Rules[3] (continued)

Table: Example of combination rules on three learners and three classes.

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.024

[3] Sourced and adapted from [Alpaydin;2010, p. 425].

Bagging

- Use *bootstrapping* to generate L training sets and train one base-learner with each [Brieman;1996]
- Given a training set $\mathcal{X}$ of size N, draw N instances randomly from $\mathcal{X}$ with replacement into $\mathcal{X}_i$.
- Use voting (average or median with regression) in testing
- **Unstable** algorithms profit from bagging ⇒ reduced variance:
  - Decision Trees
  - Multi-Layer Perceptron (MLP)
  - Condensed k-NN

Boosting

- How to:
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers?
- In bagging, the construction of complementary base-learners is left to chance and to the instability of the learning methods
- Boosting ***actively*** generates complementary base-learners by training the next learner on the mistakes of the previous learners.

Boosting – the idea[4]

- Given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – α_t
- AdaBoost by [Fruend and Schapire;1996].

[4] Description of algorithm on next slide sourced and reproduced from [Schapire;2013, p. 38].

The AdaBoost Algorithm

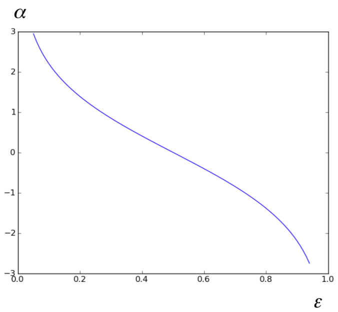
```
Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$;
for $i = 1, \ldots, M$ do
    Initialise: $D_1(i) = 1/m$;
end
for $t = 1, \ldots, T$ do
    Train weak learner using distribution $D_t$;
    Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$ from $\mathcal{H} = \{h(x)\}$;
    Aim: select $h_t$ with low weighted error;
    $\varepsilon_t = \sum_{i=1}^{m} D_t(i) [h_t(x_i) \neq y_i]$;
    Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$;
    for $i = 1, \ldots, m$ do
        Update $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$;
        where $Z_t$ is a normalization factor s.t. $\sum_{i=1}^{m} D_{t+1}(i) = 1$;
    end
end
Output the final hypothesis;
$H(x) = \mathrm{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$;
```

Inside AdaBoost

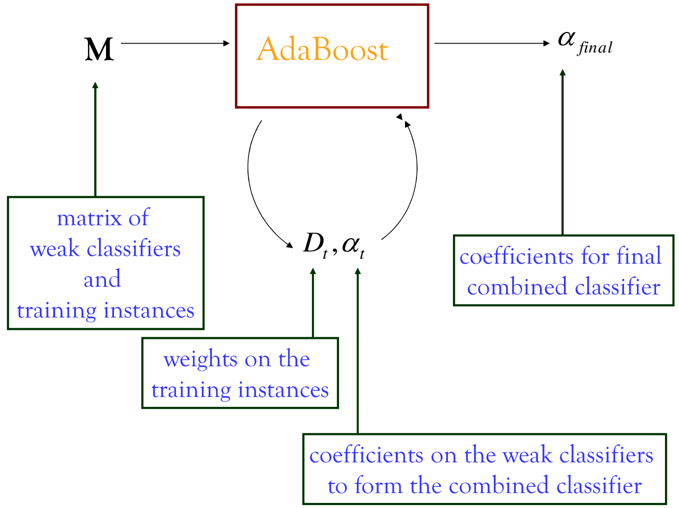
- The distribution D_t is updated with the effect of increasing the weight of examples misclassified by h_t, and decreasing the weight of correctly classified examples.
- Thus, the weight tends to concentrate on "hard" examples.
- The final hypothesis H is a weighted majority vote of the T weak hypotheses where α_t is the weight assigned to h_t.

Inside AdaBoost (continued)

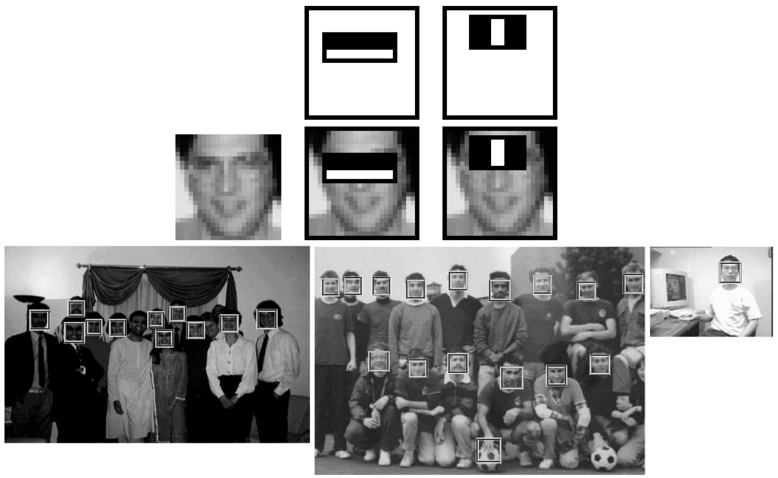
- Requires weak classifiers ($\epsilon < 0.5$)
- Smaller ϵ gives a higher α value
- Accurately classified examples get less weight
- "Hard" examples get more chances in further training



Inside AdaBoost (continued)



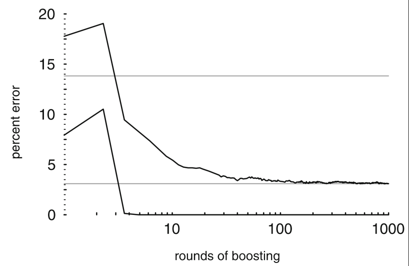
Application: Face Detection[5]



[5] Sourced and reproduced from [Viola and Jones;2001, p. 514 & p. 518].

Performance?[6]

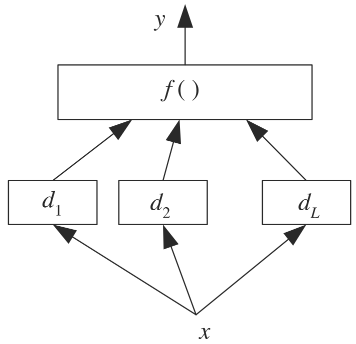
- 👍 Boosting is often rather robust to over-fitting:
  - Testing performance continues to decrease even when training error becomes zero
- 👍 Hundreds of papers published using AdaBoost
- 👎 Does not maximise classification margins [Rudin et al.;2004]



[6] Sourced and reproduced from [Schapire;2013, p. 41].

Stacking[7]

- An extension of voting: combination of d_i can be non-linear
- Combiner $f()$ is another learner [Wolpert;1992]



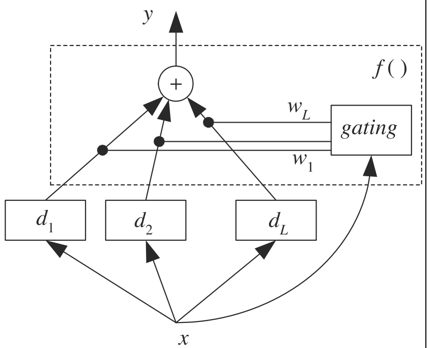
[7] Sourced and reproduced from [Alpaydin;2010, p. 436].

Fine-Tuning an Ensemble

- Given an ensemble of dependent classifiers, do not use it as is, try to get independence
- Subset selection: Forward (growing)/Backward (pruning) approaches to improve accuracy/diversity/independence
- Train metaclassifiers: From the output of correlated classifiers, extract new combinations that are uncorrelated. E.g., with PCA get "eigenlearners"
- Similar to ***feature selection*** vs. ***feature extraction***

Mixture of Experts (MoE)[8]

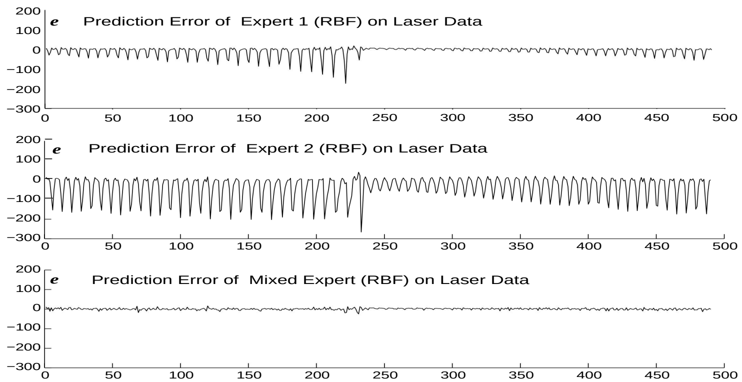
- Voting where weights are input-dependent (gating):
  $$y = \sum_{j=1}^{L} w_j d_j$$
- Experts or gating can be non-linear [Jacobs et al.;1991]



[8] Sourced and reproduced from [Alpaydin;2010, p. 434].

MoE Prediction of Chaotic Time Series[9]

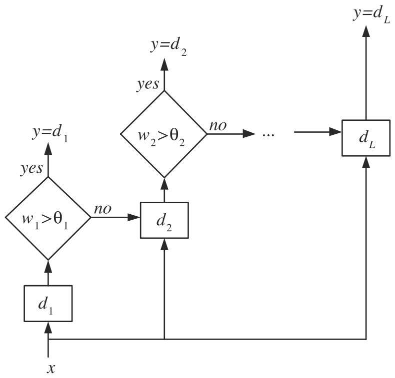
- RBF / MLP experts' prediction combined with an on-line Hidden Markov Model (HMM)
- HMM state-transition modelled by a MLP



[9] Sourced and reproduced from [Wang et al.;2003, p. 13].

Cascading[10]

- Classifiers associated with confidence w_i
- Use d_j only if preceding ones are not confident enough
- Cascade learners: simple ones for majority of data; complex ones for minorities
- Generate "rules"



[10] Sourced and reproduced from [Alpaydin;2010, p. 439].

Random Forests (RF) [Brieman;2001]

- Two layers of randomness introduced to a decision-tree based bagging approach:
  - Bagging: create new training sets by random sampling with replacement; aggregation – parallel combination of learners independently trained on distinct bootstrap samples
  - Final prediction is the mean prediction (regression) or class with maximum votes (classification)
  - Rather than using the full attribute set to determine each split in decision tree, RF selects a random subset of the predictors for each split
- Generalisation error of the forest converges as the number of trees in the forest becomes large

Recap

- [Alpaydin;2010] Chapter 17
- Freund & Schapire, A Short Introduction to Boosting
  - `http://www.yorku.ca/gisweb/eats4400/boost.pdf`[11]
- "No Free Lunch Theorems"
  - `http://www.no-free-lunch.org`[12]
- Random Forests from [Brieman;2001]

[11] Last accessed 15th September, 2017.
[12] Last accessed 15th September, 2017.

References

E. Alpaydin
Introduction to Machine Learning, 2nd Edition
The MIT Press: Cambridge, Massachusetts, 2010

L. Brieman
"Bagging Predictors"
Machine Learning, **24**(2): 123–140, 1996

L. Brieman
"Random Forests"
Machine Learning, **45**(1): 5–32, 2001

Y. Freund & R. E. Schapire
"Experiments with a new boosting algorithm"
International Conference on Machine Learning (ICML), 148–156, 1996

References (continued)

- R. A. Jacobs, M. I. Jordan, S. J. Nowlan, & G .E .Hinton
  "Adaptive Mixtures of Local Experts"
  Neural Computation, **3**: 79–87, 1991
- C. Rudin, I. Daubechies, & R. E. Schapire
  "The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins"
  Journal of Machine Learning Research, **5**: 1557–1595, 2004
- R. E. Schapire
  "Explaining AdaBoost"
  In ***Empirical Inference***, B. Schölkopf et al. (eds.), 37–51
  Springer-Verlag Berlin Heidelberg, 2013

References (continued)

- P. Viola & M. Jones
  "Rapid Object Detection Using a Boosted Cascade of Simple Features"
  IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 511–518, 2001
- X. Wang, P. Whigham, & D. Deng
  "Time-line Hidden Markov Experts and its Application in Time Series Prediction"
  Technical Report 2003/03, University of Otago, New Zealand
- D. H. Wolpert
  "Stacked Generalization"
  Neural Networks, **5**(2): 241–259, 1992