Ensembles

Jiayu Zhou

[1]Department of Computer Science and Engineering
Michigan State University
East Lansing, MI USA

March 1, 2016

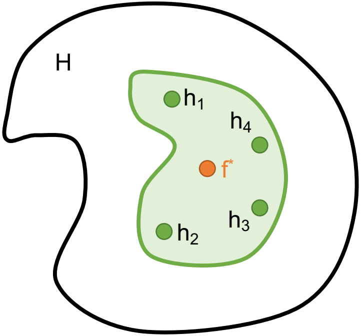
Table of contents

Ensembles

Ensemble of classifiers

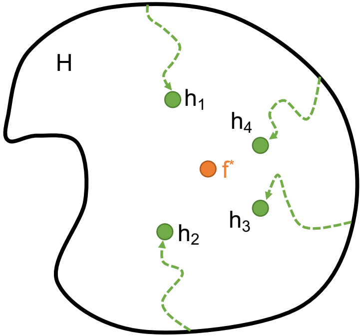
- Ensemble of classifiers
  - Consider a set of classifiers $h_1, h_2, \ldots, h_L$.
  - Construct a classifier by combining their individual decisions.
  - For example by voting their outputs.
- Accuracy
  - The ensemble works if the classifiers have low error rates.
- Diversity
  - No gain if all classifiers make the same mistakes.
  - What if classifiers make different mistakes?

Statistical motivation



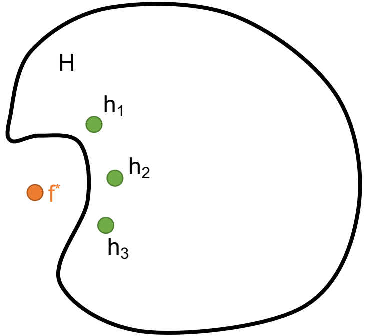
Classifiers may work well on the training set(s)

Computational motivation



Classifier search may reach local optima

Representational motivation



Classifier space may not contain best classifier

Practical Success

- Recommendation system
  - Netflix “movies you may like”.
  - Customers sometimes rate movies they rent.
  - Input: (movie, customer)
  - Output: rating
- Netflix competition
  - $1M for the first team to do 10% better than their system.
  - **Winner**: BellKor team and friends
    - Ensemble of more than 800 rating systems.
  - **Runner-up**: everybody else
    - Ensemble of all the rating systems built by the other teams.

Bayesian ensembles

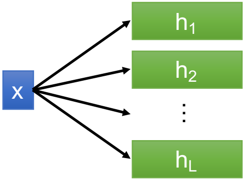
- Let $\mathcal{D}$ represent the training data.
- Enumerating all the classifiers when predicting for a new sample $\mathbf{x}$:

$$
\begin{aligned}
P(y|\mathbf{x}, \mathcal{D}) &= \sum_h P(y, h, |\mathbf{x}, \mathcal{D}) \\
&= \sum_h P(h|\mathbf{x}, \mathcal{D}) P(y|h, \mathbf{x}, \mathcal{D}) \\
&= \sum_h P(h|\mathcal{D}) P(y|\mathbf{x}; h)
\end{aligned}
$$

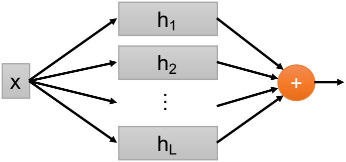
  - $P(h|\mathcal{D})$: how well does h match the training data.
  - $P(y|\mathbf{x}; h)$: what h predicts for pattern $\mathbf{x}$.
- Note that this is a weighted average.

Combining Outputs

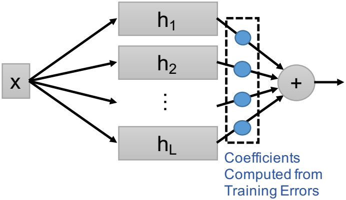
Multiple Existing Classifiers



Simple Averaging

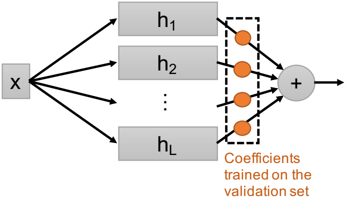


Weighted Averaging *a priori*



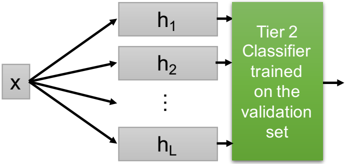
Weights derived from the training errors, e.g. $\exp(-\beta\varepsilon_{\text{Train}}(h_t))$, where $\varepsilon_{\text{Train}}(h_t)$ is the training error. Approximate Bayesian ensemble.

Weighted averaging with trained weights



- Train weights on the validation set.
- Training weights on the training set overfits easily.
- You need another validation set to estimate the performance!

Stacked classifiers (multiple stages)



- Second tier classifier trained on the validation set.
- You need another validation set to estimate the performance!

Constructing Ensembles

Diversification

- Pattern was difficult
  - hopeless
- Overfitting
  - vary the training sets
- Some features were noisy
  - vary the set of input features
- Multi-class decisions were inconsistent
  - vary the class encoding

Manipulating the training examples

- Bootstrap replication simulates training set selection
  - Given a training set of size n, construct a new training set by sampling n examples with replacement.
  - About 30% of the examples are excluded.
- Bagging
  - Create bootstrap replicates of the training set.
  - Build a decision tree for each replicate.
  - Estimate tree performance using out-of-bootstrap data.
  - Average the outputs of all decision trees.
- Boosting aggregates weak classifiers.
- Gradient Boosting (Gradient Descent + Boosting)

Manipulating the features

- Random forests
  - Construct decision trees on bootstrap replicas. Restrict the node decisions to a small subset of features picked randomly for each node (feature bagging).
  - Do not prune the trees.
    Estimate tree performance using out-of-bootstrap data. Average the outputs of all decision trees.
- Multiband speech recognition
  - Filter speech to eliminate a random subset of the frequencies.
  - Train speech recognizer on filtered data.
  - Repeat and combine with a second tier classifier.
  - Resulting recognizer is more robust to noise.

Manipulating the output codes

- Reducing multi-class problems to binary classification
  - We have seen one versus all.
  - We have seen all versus all.
- Error correcting codes for multi-class problems
  - Code the class numbers with an error correcting code.
  - Construct a binary classifier for each bit of the code.
  - Run the error correction algorithm on the binary classifier outputs.

Boosting

Motivation

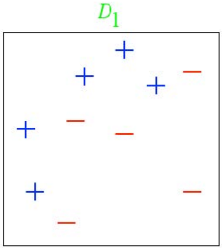
- Easy to come up with rough rules of thumb for classifying data
  - email contains more than 50% capital letters.
  - email contains expression "buy now".
- Each alone isn't great, but better than random.
- Boosting converts rough rules of thumb into an accurate classifier.
  - Adaboost
  - Gradient Boosting

Adaboost

Given examples $(x_1, y_1) \ldots (x_n, y_n)$ with $y_i = \pm 1$.

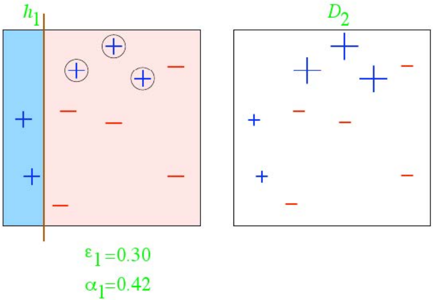
- Let $D_1(i) = 1/n$ for $i = 1 \ldots n$.
- For $t = 1 \ldots T$ do
  - Run weak learner using examples with weights D_t.
  - Get weak (base) classifier h_t
  - Compute error: $\varepsilon_t = \sum_i D_t(i)\mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$
  - Compute magic coefficient $\alpha_t = \frac{1}{2} \log \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$
  - Update weights $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$
- Output the final classifier $f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, and class label is given by $H(\mathbf{x}) = \text{sign}(f_T(\mathbf{x}))$

Toy example



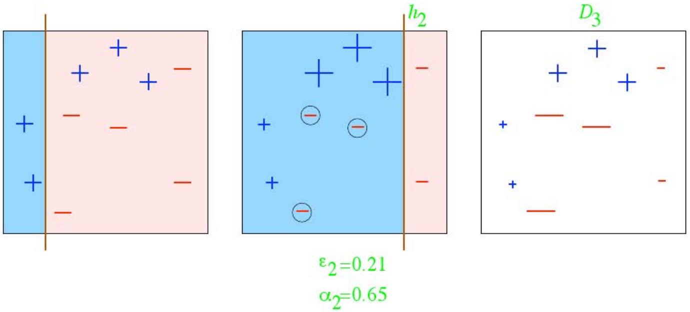
Weak classifiers: vertical or horizontal half-planes.

Adaboost round 1



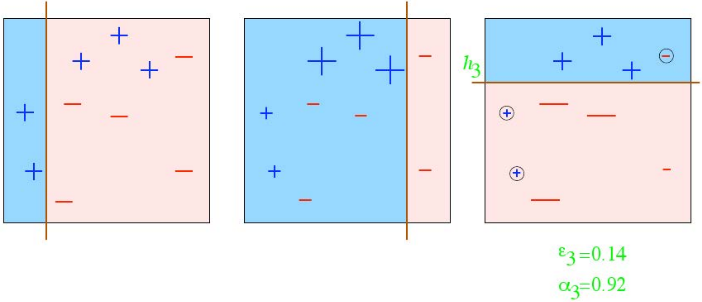
- $\varepsilon_t = \sum_i D_t(i)\mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$: error
- $\alpha_1 = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$: magic coefficient

Adaboost round 2



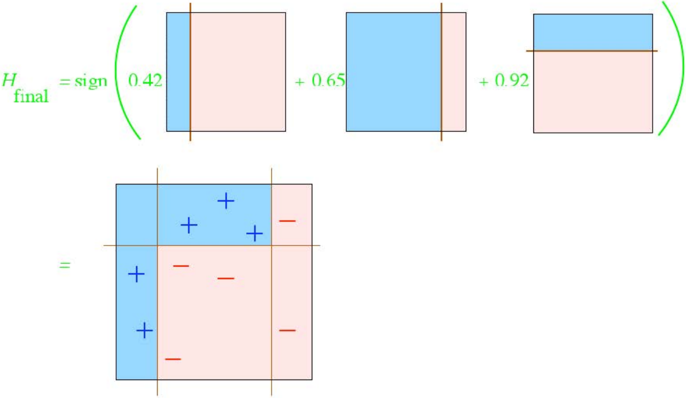
- $\varepsilon_t = \sum_i D_t(i)\mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$: error
- $\alpha_1 = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$: magic coefficient

Adaboost round 3



- $\varepsilon_t = \sum_i D_t(i)\mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$: error
- $\alpha_1 = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$: magic coefficient

Adaboost final classifier



From weak learner to strong classifier (1)

- Preliminary

$$D_{T+1}(i) = D_1(i)\frac{e^{-\alpha_1 y_i h_1(\mathbf{x}_i)}}{Z_1}\cdots\frac{e^{-\alpha_T y_i h_T(\mathbf{x}_i)}}{Z_T} = \frac{1}{n}\frac{e^{-y_i f_T(\mathbf{x}_i)}}{\prod_t Z_t}$$

- Bounding the training error

$$\frac{1}{n}\sum_i \mathbb{I}\{H(\mathbf{x}_i) \neq y_i\} \leq \sum_i e^{-y_i f_T(\mathbf{x}_i)} = \frac{1}{n}\sum_i D_{T+1}(i)\prod_t Z_t = \prod_t Z_t$$

From weak learner to strong classifier (1)

- Preliminary

$$D_{T+1}(i) = D_1(i)\frac{e^{-\alpha_1 y_i h_1(\mathbf{x}_i)}}{Z_1} \cdots \frac{e^{-\alpha_T y_i h_T(\mathbf{x}_i)}}{Z_T} = \frac{1}{n}\frac{e^{-y_i f_T(\mathbf{x}_i)}}{\prod_t Z_t}$$

- Bounding the training error

$$\frac{1}{n}\sum_i \mathbb{I}\{H(\mathbf{x}_i) \neq y_i\} \leq \sum_i e^{-y_i f_T(\mathbf{x}_i)} = \frac{1}{n}\sum_i D_{T+1}(i)\prod_t Z_t = \prod_t Z_t$$

  - $H(\mathbf{x}_i) \neq y_i \Rightarrow y_i f_T(\mathbf{x}_i) \leq 0 \Rightarrow e^{-y_i f_T(\mathbf{x}_i)} \geq 1 = \mathbb{I}\{H(\mathbf{x}_i) \neq y_i\}$

From weak learner to strong classifier (1)

- Preliminary

$$D_{T+1}(i) = D_1(i)\frac{e^{-\alpha_1 y_i h_1(\mathbf{x}_i)}}{Z_1} \dots \frac{e^{-\alpha_T y_i h_T(\mathbf{x}_i)}}{Z_T} = \frac{1}{n}\frac{e^{-y_i f_T(\mathbf{x}_i)}}{\prod_t Z_t}$$

- Bounding the training error

$$\frac{1}{n}\sum_i \mathbb{I}\{H(\mathbf{x}_i) \neq y_i\} \leq \sum_i e^{-y_i f_T(\mathbf{x}_i)} = \frac{1}{n}\sum_i D_{T+1}(i)\prod_t Z_t = \prod_t Z_t$$

  - $H(\mathbf{x}_i) \neq y_i \Rightarrow y_i f_T(\mathbf{x}_i) \leq 0 \Rightarrow e^{-y_i f_T(\mathbf{x}_i)} \geq 1 = \mathbb{I}\{H(\mathbf{x}_i) \neq y_i\}$

- Idea: make Z_t as small as possible.

$$Z_t = \sum_{i=1}^{n} D_t(i) e^{-\alpha_t y_t h_t(\mathbf{x}_i)} = (1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$$

  - Pick α_t to minimize Z_t.

From weak learner to strong classifier (2)

- Pick α_t to minimize Z_t (the magic coefficient)

$$\frac{\partial Z_t}{\partial \alpha_t} = -(1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} = 0 \Rightarrow \alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$$

- Weak learner assumption: $\gamma_t = \frac{1}{2} - \varepsilon_t$ is positive and small.

$$Z_t = (1-\varepsilon)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \sqrt{4\varepsilon_t(1-\varepsilon_t)} = \sqrt{1-4\gamma_t^2} \leq \exp(-2\gamma_t^2)$$

$$\text{Training Error}(f_T) \leq \prod_{t=1}^{T} Z_t \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

- The training error decreases exponentially if $\inf \gamma_t > 0$.

Boosting and exponential loss

- We obtain the bound

$$\text{Training Error}(f_T) \leq \frac{1}{n}\sum_i e^{-y_i f_T(\mathbf{x}_i)} = \frac{1}{n}\sum_i e^{-y_i \sum_t \alpha_t h_t(\mathbf{x}_i)} = \prod_{t=1}^{T} Z_t$$

  - without saying how $\mathcal{D}_t$ relates to h_t
  - without using the value of α_t
- Conclusion
  - Round T chooses the h_T and α_T that maximize the exponential loss reduction from f_{T-1} to f_T.

Summary

aggregation type	**blending**	**learning**
uniform	voting	Bagging
non-uniform	linear	boosting (e.g. Adaboost)
conditional	stacking	Decision Tree