Chapter 14 – Combining Models

T-61.6020 Special Course II: Pattern Recognition and Machine Learning Spring 2007

Jukka Parviainen

Laboratory of Computer and Information Science
TKK

April 30th 2007

Outline

Chapter 14

- shortest chapter in the book
- examples in regression and classification
- Bishop style: exponential error functions introduced with which boosting can be expressed in a flexible way, etc.
- BiShop-Bingo
  - three crosses in row, column or diagonal
  - erase the counters – BS-Bingo starts...

Chapter 14

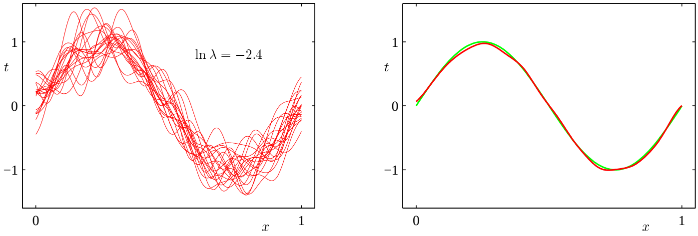
- shortest chapter in the book
- examples in regression and classification
- Bishop style: exponential error functions introduced with which boosting can be expressed in a flexible way, etc.
- BiShop-Bingo
  - three crosses in row, column or diagonal
  - erase the counters – BS-Bingo starts...
  - ...NOW!

Committees

- ensemble of statistical classifiers are more accurate than a single classifier
- *weak learner* or *weak classifier*: slightly better than chance
- final results by voting (classification) or averaging (regression)
- some techniques: bagging, boosting

Bootstrap aggregating – Bagging

- a committee technique based on bootstrapping the data set and model averaging
- bootstrapping: given a data set of size *N*, create *M* datasets of size *N* with replacement
- averaging low-bias models produce accurate predictions – bias-variance decomposotion (Section 3.5)



Bagging in Regression

- example on regression $y(\mathbf{x}) = h(\mathbf{x}) + \epsilon(\mathbf{x})$
- from a single data set $\mathbf{D}$ M bootstrap data sets $\mathbf{D}_m$, and from which regressors $y_m(\mathbf{x})$ with errors $\epsilon_m(\mathbf{x})$
- sum-of-squares error

$$\mathbb{E}_{\mathbf{x}}\{(y_m(\mathbf{x}) - h(\mathbf{x}))^2\} = \mathbb{E}_{\mathbf{x}}\{\epsilon(\mathbf{x})^2\}$$

- average of individual errors

$$E_{\mathrm{AV}} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}\{\epsilon_m(\mathbf{x})^2\}$$

Averaging Gives Better Permormance

- committee prediction is the average of y_m

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

- expected error from the committee

$$E_{\mathrm{COM}} = \mathbb{E}_{\mathbf{x}}\{\big(y_{\mathrm{COM}}(\mathbf{x}) - h(\mathbf{x})\big)^2\} = \mathbb{E}_{\mathbf{x}}\{\big(\frac{1}{M} \sum_{m=1}^{M} \epsilon_m(\mathbf{x})\big)^2\}$$

- under assumptions that errors $\epsilon_m(\mathbf{x})$ zero-mean and uncorrelated we obtain

$$E_{\mathrm{COM}} = \frac{1}{M} E_{\mathrm{AV}}$$

Not As Good As in Theory

- assumptions do not hold generally
- however, it can be proved that $E_{COM} \leq E_{AV}$, e.g.,

$$\mathbb{E}_{\mathbf{x}}\{(h(\mathbf{x})-y_m(\mathbf{x}))^2\} = h(\mathbf{x})^2-2h(\mathbf{x})\mathbb{E}_{\mathbf{x}}\{y_m(\mathbf{x})\}+\mathbb{E}_{\mathbf{x}}\{y_m(\mathbf{x})^2\}$$

- using inequality $\mathbb{E}_{\mathbf{x}}\{X^2\} \geq \mathbb{E}_{\mathbf{x}}\{X\}^2$ and $\mathbb{E}_{\mathbf{x}}\{y_m(\mathbf{x})\} = y_{\mathrm{COM}}(\mathbf{x})$ we get

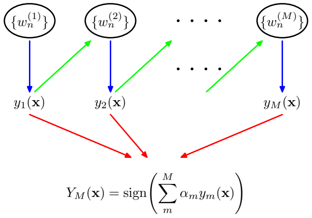
$$(h(\mathbf{x}) - y_{\mathrm{COM}}(\mathbf{x}))^2 \leq \mathbb{E}_{\mathbf{x}}\{(h(\mathbf{x}) - y_m(\mathbf{x}))^2\}$$

Boosting

- training in sequence
- misclassified data point gets more weight in the following classifier
- final prediction given by a weighted majority voting scheme
- example on two-class classification problem with most widely used algorithm AdaBoost

Adaptive Boosting – AdaBoost

- weights w_i for each training sample
- M weak classifiers in sequence
- indicator function $I(y_m(\mathbf{x}_n) \neq t_n)$, which equals 1 if the argument is true, i.e., in case of misclassification
- misclassified data points will have more weight in the following classifier
- weights α_m for each classifier



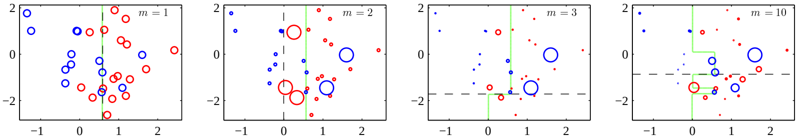
AdaBoost: Algorithm

Algorithm

1. Initialize data weights $w_n^{(1)} = 1/N$
2. For $m = 1, \ldots, M$,
   2.1 Fit a classifier $y_m(x)$ by minimizing
   $J_m = \sum w_n^{(m)} I(y_m(\mathbf{x}_n \neq t_n))$
   2.2 Evaluate quantity ϵ_m (ratio of misclassified)
   $\epsilon_m = \frac{\sum_n w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_n w_n^{(m)}}$
   Evaluate quantity α_m (weight for classfier m)
   $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
   2.3 Update the data weighting coefficients
   $w_n^{(m+1)} = w_n^{(m)} e^{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)}$
3. Make prediction by $Y_M(\mathbf{x}) = \mathrm{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$

AdaBoost: Example

- base learners consist of a threshold on one of the input variables
- misclassified samples by classifier at $m = 1$ get greater weight for $m = 2$
- final classification: $Y_m(\mathbf{x}) = \mathrm{sign}(\sum_m \alpha_m y_m(\mathbf{x}))$



Boosting as Sequential Minimization

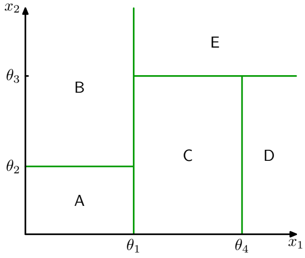
- boosting was originally motivated by statistical learning theory
- here sequential optimization of exponential error function ("in a Bishop Style")
  - error function $E = \sum_{n=1}^{N} e^{-t_n f_m(\mathbf{x}_n)}$
  - combined classifier $f_m(\mathbf{x}) = 0.5 \sum_l \alpha_l y_l(\mathbf{x})$
  - keeping base classifiers $y_1(\mathbf{x}) \dots y_{m-1}(\mathbf{x})$ with corresponding α_l fixed and minimizing only "the last" α_m and $y_m(\mathbf{x})$ leads to the same equations as in AdaBoost

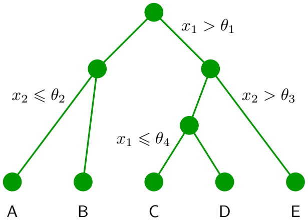
Error Functions for Boosting

- lots of boosting-like algorithms by altering of error function
- exponential error function
  - sequential minimization leads to simple AdaBoost
  - penalizes large negative values of $ty(\mathbf{x})$
- cross-entropy error function for $t \in \{-1, 1\}$: $\log(1 + e^{-yt})$
  - more robust to outliers
  - log likelihoods for any distribution exist
  - multi-class problems possible to solve

Classification and Regression Trees – CART

- input space is splitted into cuboid regions; axis-aligned boundaries
- only one model, e.g., constant, in one region
- human interpretation is easy





CART: Learning from Data

- determine from data
  - structure of a tree
  - input variable for each node
  - threshold values θ_i for a split
  - values of prediction
- combinatorially infeasible $\rightarrow$ greedy algorithm
  - from a single node start growing
  - stopping criterion
  - pruning criterion

Drawbacks of CART

- learning of a tree is sensitive to data
- splits aligned with axes of feature space
- hard splitting: each region of input space belongs to one and only one node
- piecewise-constant predictions of a tree not smooth
- $\rightarrow$ hierarchical mixture of expers

Mixture of Linear Regression Models

- simple probabilistic cases for regression and classification
  - mixtures of linear regression models
  - mixtures of logistic models
- Gaussians with mixing coefficients independent from input variables

$$p(t|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(t|\mathbf{w}_k^T \phi, \beta^{-1})$$

EM for Maximizing Log Likelihood

- log likelihood function given a data set of $\{\phi_n, t_n\}$

$$\log p(\mathbf{t}|\theta) = \sum_{n=1}^{N} \log\Big(\sum_{k=1}^{K} \pi_k \mathcal{N}(t_n|\mathbf{w}_k^T \phi_n, \beta^{-1})\Big)$$

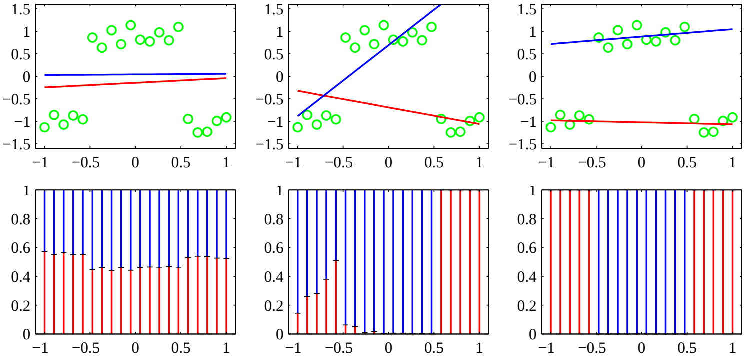
- complete-data log likelihood function with binary latent variables z_{nk}

$$\log p(\mathbf{t}, \mathbf{Z}|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log\big(\pi_k \mathcal{N}(t_n|\mathbf{w}_k^T \phi_n, \beta^{-1})\big)$$

- EM for γ_{nk}, $Q(\theta, \theta^{\text{old}})$, π_k, $\mathbf{w}_k$, and β

Example

- mixture of two linear regressors
- drawback: lot of probability mass with no data
- solution: input dependent mixing coefficients



Mixture of Experts

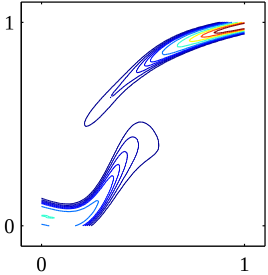
- mixture of linear regression models
  $p(t|\theta) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{t}|\theta)$
- mixture of experts model

$$p(\mathbf{t}|\mathbf{x}, \theta) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p_k(\mathbf{t}|\mathbf{x}, \theta)$$

- mixing coefficients, *gating functions*, as functions of input
- individual component densities, *experts*

Hierarchical Mixture of Experts

- probabilistic version of decision trees
  - each component in the mixture is itself a mixture distribution
  - nodes: probabilistic splits of all input variables
  - leaves: probabilistic models
- mixture density network (Section 5.6)



(c)

Summary

- multiple models to increase capabilities of the regressor or classifier
- basic methods bagging and boosting improve results compared to a single learner
- decision trees are easy to interpret
- probabilistic networks extend models

Course Feedback

`http://www.cs.hut.fi/Opinnot/Palaute/kurssipalaute.html`
→ Kevään 2007 kurssikyselyt
→ T-61.6020