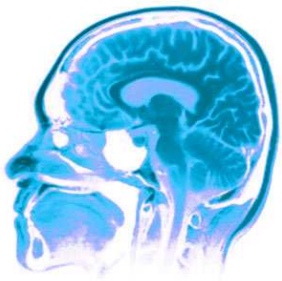
CPSC540



Boosting and Random Forests

Nando de Freitas
November, 2011
University of British Columbia

Ensemble methods

Ensemble methods (boosting, random forests) are powerful algorithm design paradigms. They

- enable us to combine many weak learners to produce a strong learner.
- can be applied to nonlinear regression, density estimation, classification and are brilliant tools for feature selection.
- are easy understand, implement and optimize.

The boosting slides used in this lecture use material from *Peter Bühlmann and Bin Yu (2009). Boosting. Wiley Interdisciplinary Reviews: Computational Statistics.* I strongly recommend the book of Friedman, Hastie and Tibshirani for this section of the course.

Problem setup

Suppose that we observe

$$(X_1, Y_1), \ldots, (X_n, Y_n),$$

where $X_i \in \mathbb{R}^p$ denotes a p-dimensional predictor variable and Y_i a univariate response, for example taking values in $\mathbb{R}$ as in regression or in $\{-1, +1\}$ as in binary classification. In the sequel, we denote by $X^{(j)}$ the jth component of a vector $X \in \mathbb{R}^p$. We usually assume that the pairs (X_i, Y_i) are i.i.d. or from a stationary process. The goal is to estimate the regression function $F(x) = \mathbb{E}[Y|X = x]$ or to find a classifier $\text{sign}(F(x))$ where $F(x) : \mathbb{R}^p \to \mathbb{R}$.

Expected and Empirical Loss

The estimation performance is evaluated via a real-valued loss function in the sense that we want to minimize the expected loss or risk:

$$\mathbb{E}L(Y, F(X)),$$

based on data $(X_i, Y_i)(i = 1, ..., n)$. The loss function L is assumed to be smooth and convex in the second argument so that the gradient method can be applied. Boosting algorithms are obtainable by minimizing the empirical loss function

$$n^{-1} \sum_{i=1}^{n} L(Y_i, F(X_i)),$$

From weak learners to strong ones: functional gradient descent

The boosting methodology in general builds on a user-determined base procedure or weak learner and uses it repeatedly on modified data which are typically outputs from the previous iterations. The final boosted procedure takes the form of linear combinations of the base procedures. Precisely, given a base learner $h(x, \theta)$, boosting is derivable as functional gradient descent on the loss function L.

Boosting (gradient descent view)

1. Start with $F_0(x) = 0$.
2. Given $F_{m-1}(x)$, let

$$(\beta_m, h(x, \hat{\theta}_m))) = argmin_{\beta \in \mathbb{R}, \theta} \sum_{i=1}^{n} L(Y_i, F_{m-1}(X_i) + \beta h(x, \theta)).$$

3. Set

$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m).$$

4. Stop when $m = M$.

The AdaBoost classifier is $\text{sign}(F_M(x))$.

L2Boosting: coordinatewise descent for linear models

1. Start with $F_0 = 0$.
2. Given $F_{m-1}(x)$, Compute residuals $U_i = Y_i - F_{m-1}(X_i)$ $(i = 1, \ldots, n)$. Let $X_i^{(j)}$ be the jth component of $X_i \in \mathbb{R}^p$,

$$\hat{j}_m = argmin_{j=1,\ldots,p} \sum_{i=1}^{n} (U_i - \hat{\beta}_m X_i^{(j)}))^2,$$

$$\hat{\beta}_m = argmin_{\beta} \sum_{i=1}^{n} (U_i - \beta X_i^{(j_m)})^2,$$

3.

$$F_m(x) = F_{m-1}(x) + \hat{\beta}_m X^{(\hat{j}_m)},$$

4. Stop when $m = M$ and $F_M(x)$ is the final estimator of the linear regression function.

L2 boosting

```
# Assume the input data is X and Y
NX, NA = X.shape # NA is the number of features and NX the number of data cases.
F = zeros(Y.shape, dtype=float32)
feats = []
betas = []
for _ in xrange(100):
    U = Y-F # compute residuals
    minerr = 0
    minind = -1
    for a in xrange(NA):
        for b in arange(-1, 1, .2):
            err = sum((U-b*X[:,a])**2)
            if err < minerr or minind < 0:  # print '\t', err
                minerr = err
                minind = a
                minbeta = b

    feats.append(minind)
    betas.append(minbeta)

    F = F + minbeta * X[:,minind]
```

Adaboost

In binary classification, $y \in \{-1, +1\}$ and the most commonly used loss is the 0-1 loss. That is, for a classifier $\text{sign}(F(x)) \in \{-1, +1\}$ if the label of x is $y \in \{-1, +1\}$, the 0-1 loss can be written as a function of the margin $yF(x)$:

$$L_{01}(y, F(x)) = I\{yF(x) < 0\}.$$

It is easy to see that the exponential loss function

$$L_{exp}(y, F(x)) = \exp(-yF(x))$$

is an upper bound on L_{01} and its population minimizer is half of the log odds ratio

$$F(x) = \frac{1}{2} \log \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = -1|X = x)}. \tag{1}$$

AdaBoost

1. Start with $F_0(x) = 0$;
2. Given $F_{m-1}(x)$, let

$$w_i^{(m)} = \exp(-Y_i F_{m-1}(X_i)),$$

$$h(x, \hat{\theta}_m)) = argmin_\theta \sum_{i=1}^{n} w_i^{(m)} I(Y_i \neq h(X_i, \theta)),$$

and denote $h(\cdot, \hat{\theta}_m)$'s associated error by

$$err_m = \frac{\sum_{i=1}^{n} w_i^{(m)} I(Y_i \neq h(X_i, \hat{\theta}_m))}{\sum_{i=1}^{n} w_i^{(m)}}.$$

Furthermore let

$$\beta_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}.$$

3. Set

$$F_m(x) = F_{m-1}(x) + \beta_m h(x, \hat{\theta}_m).$$

4. Stop when $m = M$.
5. The AdaBoost classifier is $y = \text{sign}(F_M(x))$.

BoosTexter for text classification

In news categorization, a possible term is *Bill Clinton*. A corresponding weak learner is: If the term *Bill Clinton* appears in the document predict that the document belongs to News with high confidence.

Formally, denote a possible term by w, and let us define (abusively) $w \in x$ to mean that w occurs in document x. Based on the term, we will be interested in weak hypotheses h which make predictions of the form:

$$h(x, \ell) = \begin{cases} c_{0\ell} & \text{if } w \notin x \\ c_{1\ell} & \text{if } w \in x \end{cases}$$

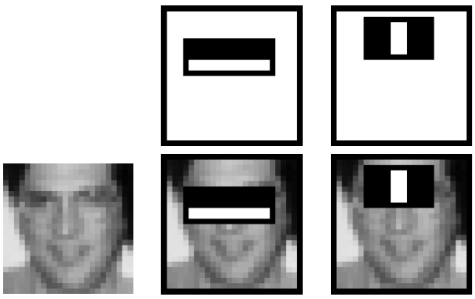
[Schapire and Singer, 2000]

Boosting for object detection

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

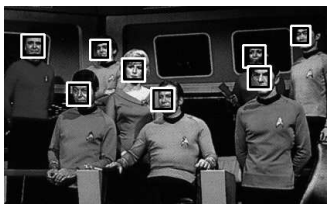
where $\alpha_t = \log \frac{1}{\beta_t}$



x is a 24x24 pixel sub-window of an image

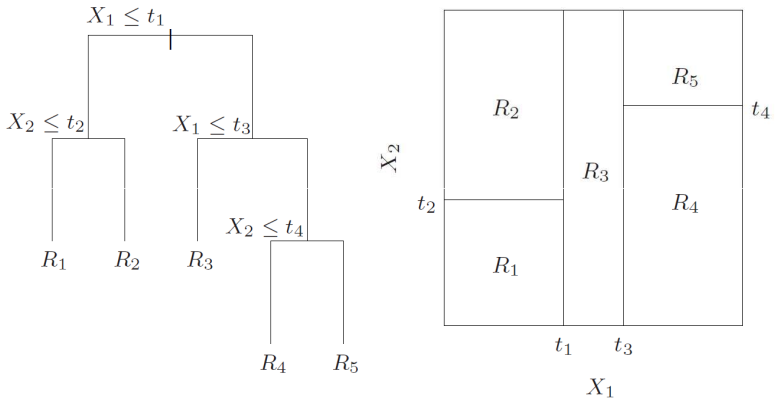
$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

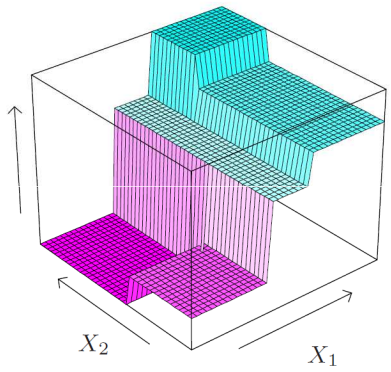
feature f_j parity p_j threshold θ_j



[Viola and Jones, 2001]

Trees for classification and regression





$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}$$

[From the book of Hastie, Friedman and Tibshirani]

Regression Trees

$$R_1(j,s) = \{X | X_j \le s\} \quad \text{and} \quad R_2(j,s) = \{X | X_j > s\}.$$

Then we seek the splitting variable j and split point s that solve

$$\min_{j,\ s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right].$$

For any choice j and s, the inner minimization is solved by

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j,s)) \quad \text{and} \quad \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j,s)).$$

[From the book of Hastie, Friedman and Tibshirani]

Classification Trees

In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),$$

the proportion of class k observations in node m.

Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$.

Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

[From the book of Hastie, Friedman and Tibshirani]

Random Forests

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

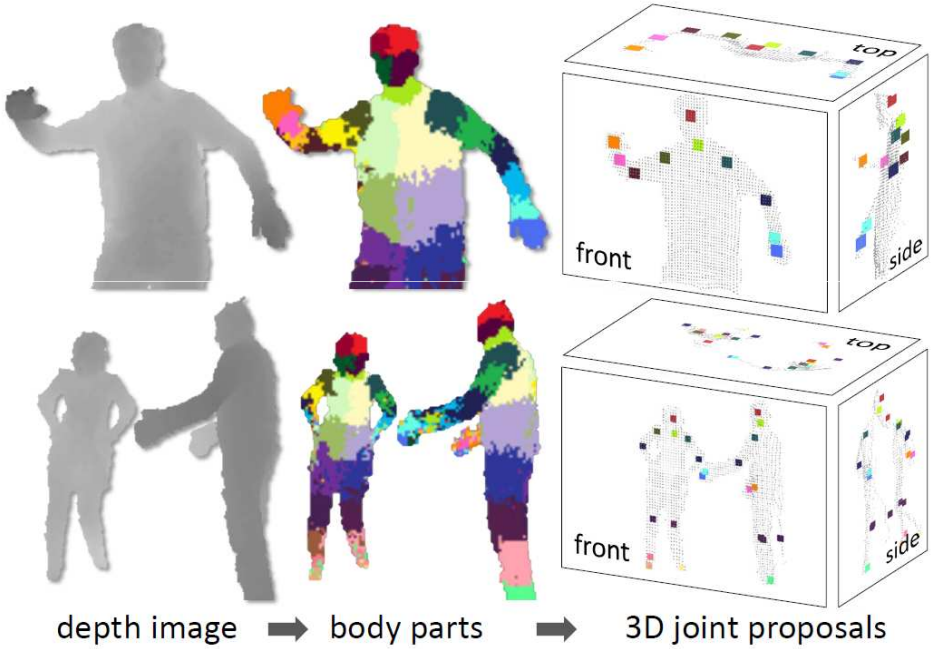
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

[From the book of Hastie, Friedman and Tibshirani]

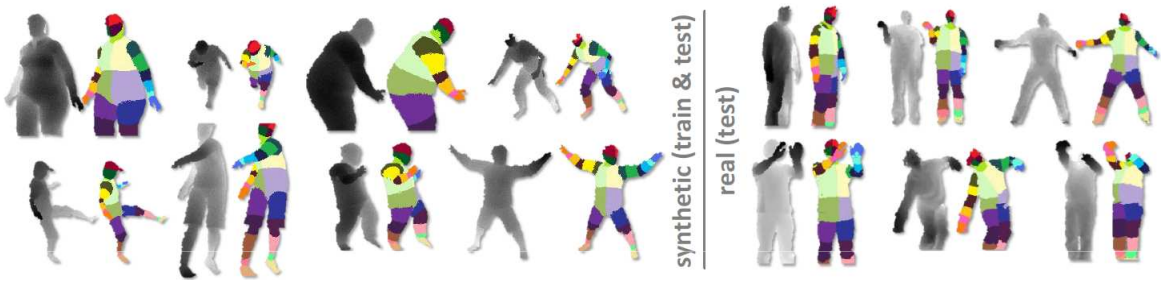
Random Forests and the Kinect



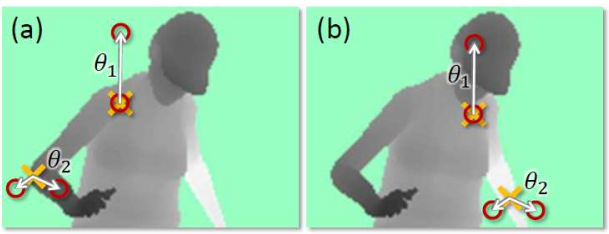
[Jamie Shotton et al 2011]

Random Forests and the Kinect

Lesson 1: Use computer graphics to generate plenty of data.

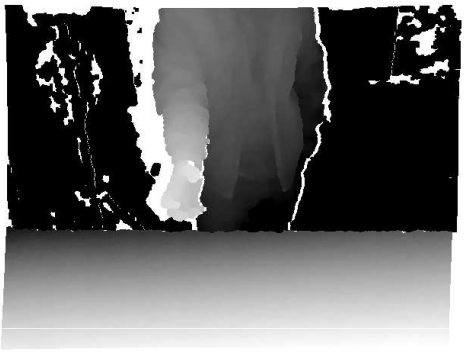


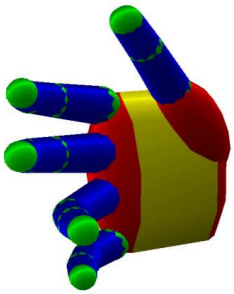
Lesson 2: Use simple depth features within random forests algorithm.



[Jamie Shotton et al 2011]

Random Forests and the Kinect





[Iason Oikonomidis et al 2011]