CS480/680
Lecture 24: July 29, 2019

Gradient Boosting, Bagging, Decision Forest

[RN] Sec. 18.10, [M] Sec. 16.2.5, 16.4.5
[B] Chap. 14, [HTF] Chap 10, 15-16,
[D] Chap. 13

Gradient Boosting

- AdaBoost designed for classification
- How can we use boosting for regression?

- Answer: **Gradient Boosting**

Gradient Boosting

Idea:

- Predictor f_k at stage k incurs loss $L(f_k(\boldsymbol{x}), y)$
- Train h_{k+1} to approximate negative gradient:

$$h_{k+1}(\boldsymbol{x}) \approx -\frac{\partial L(f_k(\boldsymbol{x}), y)}{\partial f_k(\boldsymbol{x})}$$

- Update predictor by adding a multiple η_{k+1} of h_{k+1}:

$$f_{k+1}(\boldsymbol{x}) \leftarrow f_k(\boldsymbol{x}) + \eta_{k+1}\, h_{k+1}(\boldsymbol{x})$$

Squared Loss

- Consider **squared loss**

$$L(f_k(\boldsymbol{x}_n), y_n) = \frac{1}{2}(f_k(\boldsymbol{x}_n) - y_n)^2$$

- Negative gradient corresponds to **residual** r

$$-\frac{\partial L(f_k(\boldsymbol{x}_n), y_n)}{\partial f_k(\boldsymbol{x}_n)} = y_n - f_k(\boldsymbol{x}_n) = r_n$$

- Train **base learner** h_{k+1} with **residual dataset** $\{(\boldsymbol{x}_n, r_n)_{\forall n}\}$
- Base learner h_{k+1} can be any **non-linear predictor** (often a small decision tree)

Gradient Boosting Algorithm

- Initialize predictor with a constant c:

$$f_0(\boldsymbol{x}_n) = argmin_c \sum_n L(c, y_n)$$

- For $k = 1$ to K do
  - Compute pseudo residuals: $r_n = -\frac{\partial L(f_{k-1}(\boldsymbol{x}_n), y_n)}{\partial f_{k-1}(\boldsymbol{x}_n)}$
  - Train a base learner h_k with residual dataset $\{(\boldsymbol{x}_n, r_n)_{\forall n}\}$
  - Optimize step length:

$$\eta_k = argmin_\eta \sum_n L(f_{k-1}(\boldsymbol{x}_n) + \eta h_k(\boldsymbol{x}_n), y_n)$$

  - Update predictor: $f_k(\boldsymbol{x}) \leftarrow f_{k-1}(\boldsymbol{x}) + \eta_k h_k(\boldsymbol{x})$

XGBoost

- eXtreme Gradient Boosting
  - Package optimized for speed and accuracy
  - XGBoost used in >12 winning entries for various challenges https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

Boosting vs Bagging

- Review

Independent classifiers/predictors

- How can we obtain independent classifiers/predictors for bagging?

- Bootstrap sampling
  - Sample (without replacement) subset of data
- Random projection
  - Sample (without replacement) subset of features

- Learn different classifiers/predictors based on each data subset and feature subset

Bagging

For k = 1 to K

$\boldsymbol{D}_k \leftarrow$ sample data subset

$\boldsymbol{F}_k \leftarrow$ sample feature subset

$h_k \leftarrow$ train classifier/predictor based on $\boldsymbol{D}_k$ and $\boldsymbol{F}_k$

Classification: $majority(h_1(\boldsymbol{x}), \ldots, h_K(\boldsymbol{x}))$

Regression: $average(h_1(\boldsymbol{x}), \ldots, h_K(\boldsymbol{x}))$

Random forest: bag of decision trees

Application: Xbox 360 Kinect

- Microsoft Cambridge
- Body part recognition: supervised learning

Depth camera

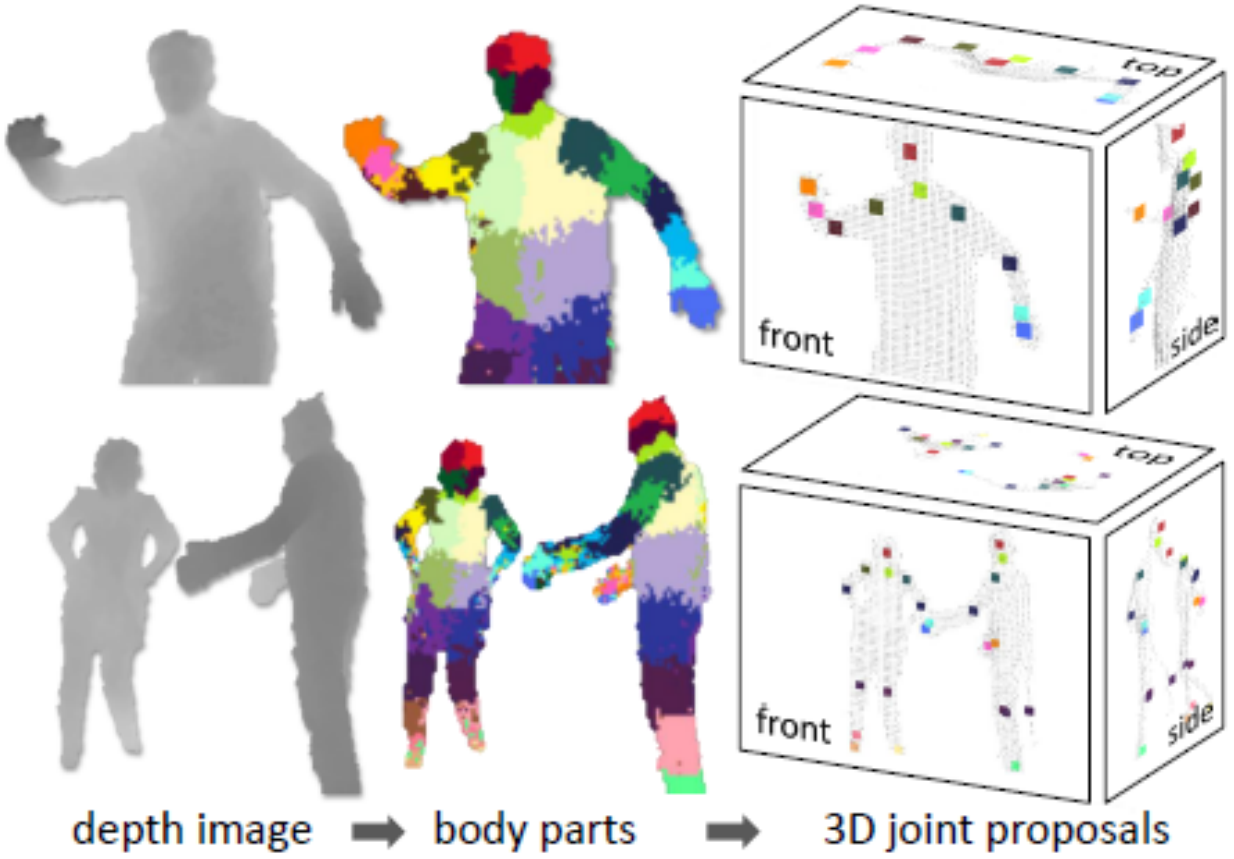
- Kinect

Infrared image

Gray scale depth map

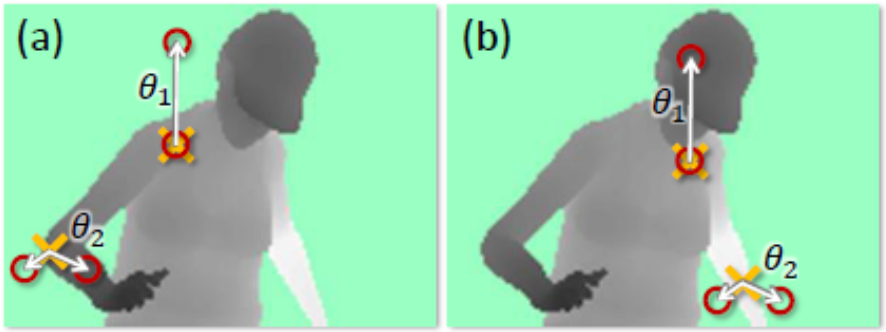
Kinect Body Part Recognition

- Problem: label each pixel with a body part

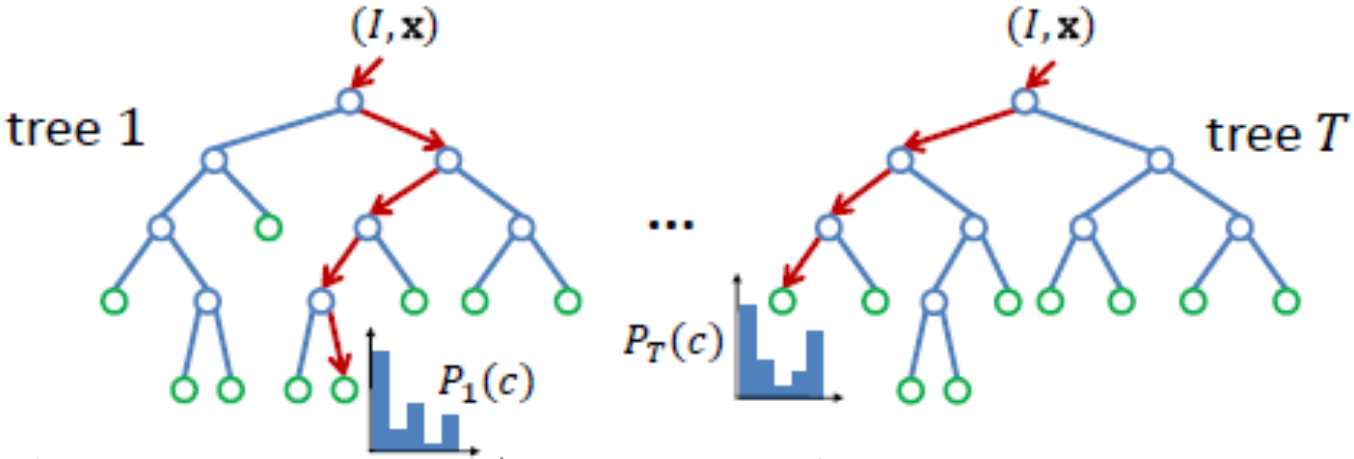


Kinect Body Part Recognition

- Features: depth differences between pairs of pixels



- Classification: forest of decision trees



Large Scale Machine Learning

- Big data
  - Large number of data instances
  - Large number of features

- Solution: distribute computation (parallel computation)
  - GPU (Graphics Processing Unit)
  - Many cores

GPU computation

- Many Machine Learning algorithms consist of vector, matrix and tensor operations
  - A tensor is a multidimensional array
- GPU (Graphics Processing Units) can perform arithmetic operations on all elements of a tensor in parallel
- Packages that facilitate ML programming on GPUs: Keras, PyTorch, TensorFlow, MXNet, Theano, Caffe, DL4J

Multicore Computation

- Idea: Train a different classifier/predictor with a subset of the data on each core

- How can we combine the classifiers/predictors?
- Should we take the average of the parameters of the classifiers/predictors?

No, this might lead to a worse classifier/predictor. This is especially problematic for models with hidden variables/units such as neural networks and hidden Markov models

Bad case of parameter averaging

- Consider two threshold neural networks that encode the exclusive-or Boolean function

- Averaging the weights yields a new neural network that does not encode exclusive-or

Safely Combining Predictions

- A safe approach to ensemble learning is to **combine the predictions** (not the parameters)

- **Classification:** majority vote of the classes predicted by the classifiers

- **Regression:** average of the predictions computed by the regressors

Other UW Courses Related to ML

- CS486/686: Artificial Intelligence
- CS475/675: Computational Linear Algebra
- CS485/685: Theoretical Foundations of ML (Shai Ben-David)
- CS794 Optimization for Data Science
- CS795 Fundamentals of Optimization
- CS870: Biologically Plausible Neural Networks (Jeff Orchard)
- CS898: Deep Learning and its Applications (Ming Li)
- CS885: Reinforcement Learning (Pascal Poupart)
- STAT440/840: Computational Inference
- STAT441/841: Statistical Learning – Classification
- STAT442/890: Data visualization
- STAT444/844: Statistical Learning – Regression
- STAT450/850: Estimation and hypothesis testing

Data Science at UW

- https://uwaterloo.ca/data-science/
- Intersection of AI, Machine Learning, Data Systems, Statistics and Optimization
- Bachelor in Data Science
- Master in Data Science (and Artificial Intelligence)
  - Course-based option
  - Thesis-based option