Lecture 23: Gradient Boosting, Bagging, Decision Forest CS480/680 Intro to Machine Learning

2023-4-4

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

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**

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

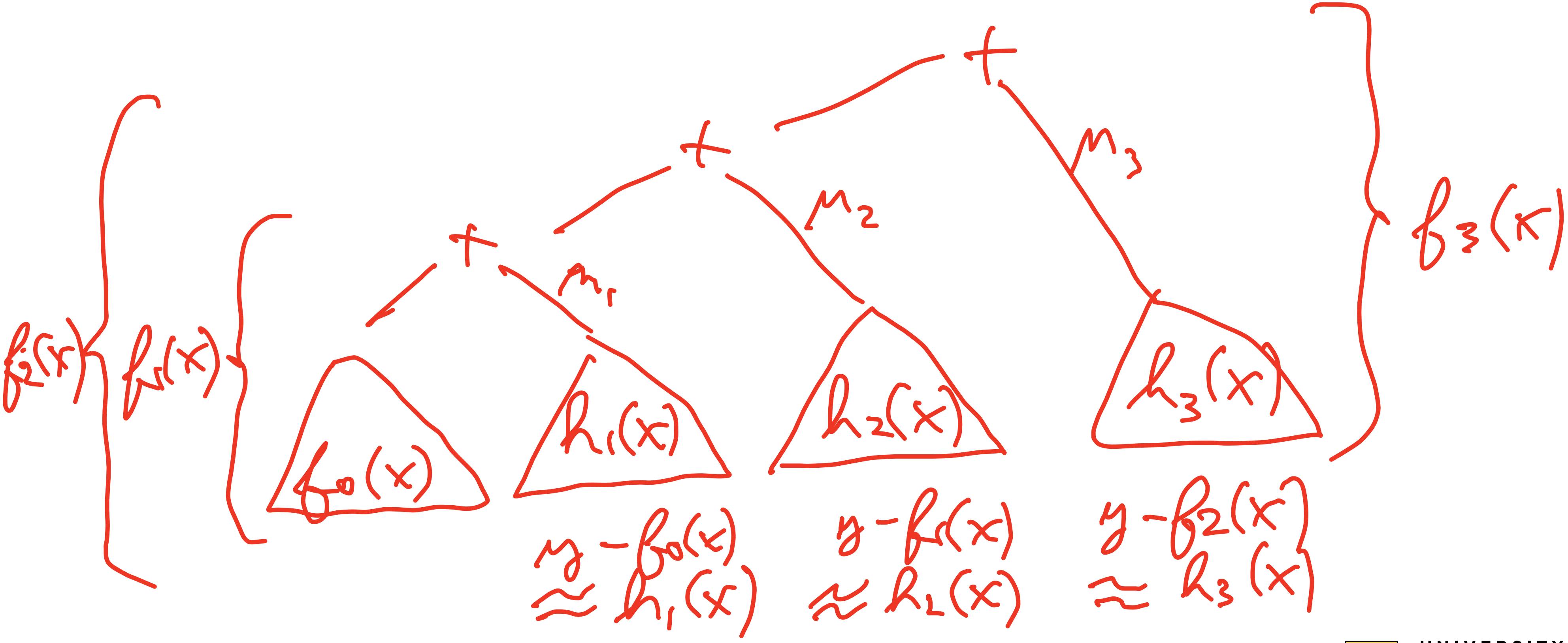
- Negative gradient corresponds to **residual** r

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

- 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)

Illustration



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

Bagging
- majority vote
- Assumptions
  - independent hypotheses
  - hypotheses with similar accuracies

Boosting
- weighted predictions
- Allows:
  - Correlated hypotheses
  - Hypotheses with imbalanced accuracies

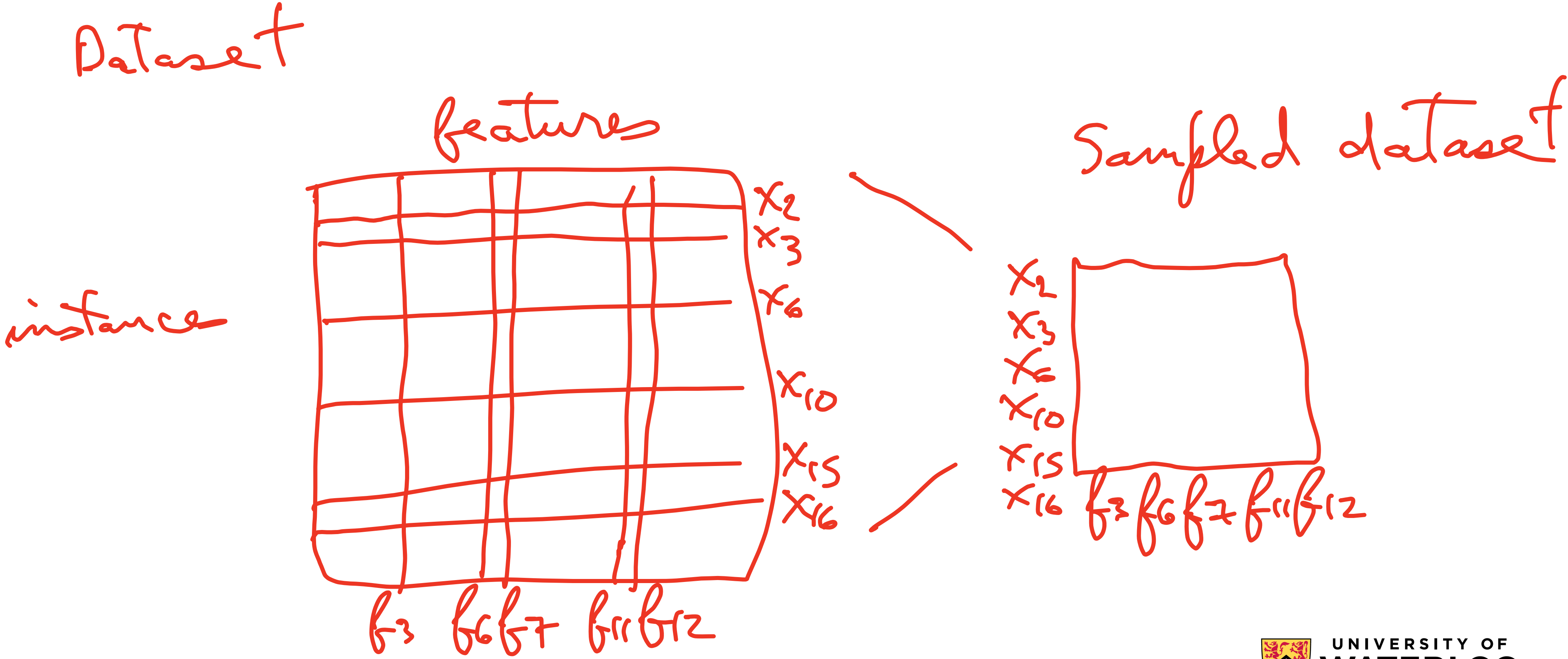
Independent classifiers/predictors

- 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

Illustration of Bootstrap Sampling and Random Projection



UNIVERSITY OF WATERLOO

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

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

Random forest: bag of decision trees

Application: Xbox 360 Kinect

- Microsoft Cambridge
- Body part recognition: supervised learning

Depth camera

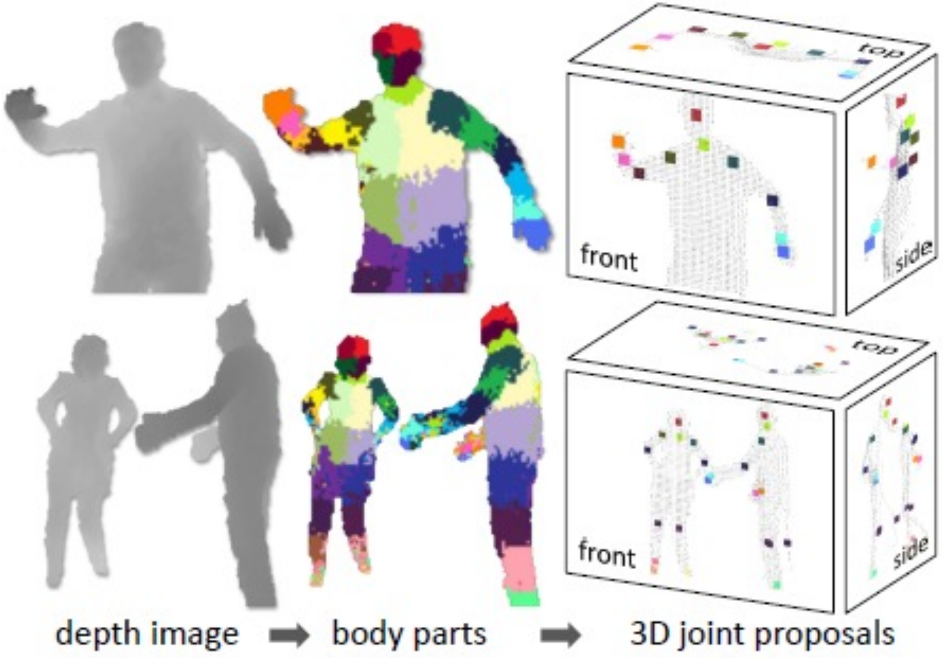
- Kinect

Infrared image

Gray scale depth map

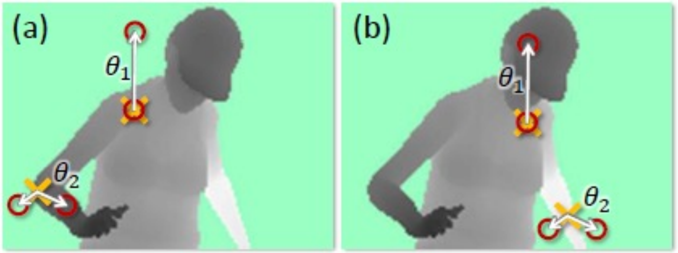
Kinect Body Part Recognition

- Problem: label each pixel with a body part

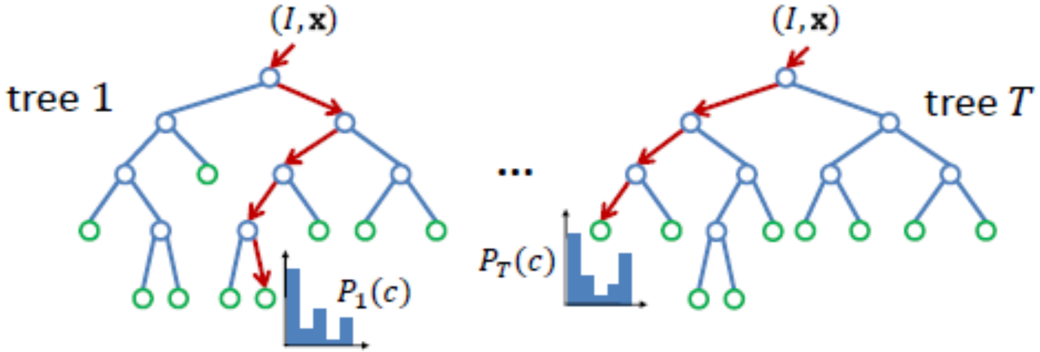


Kinect Body Part Recognition

- 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

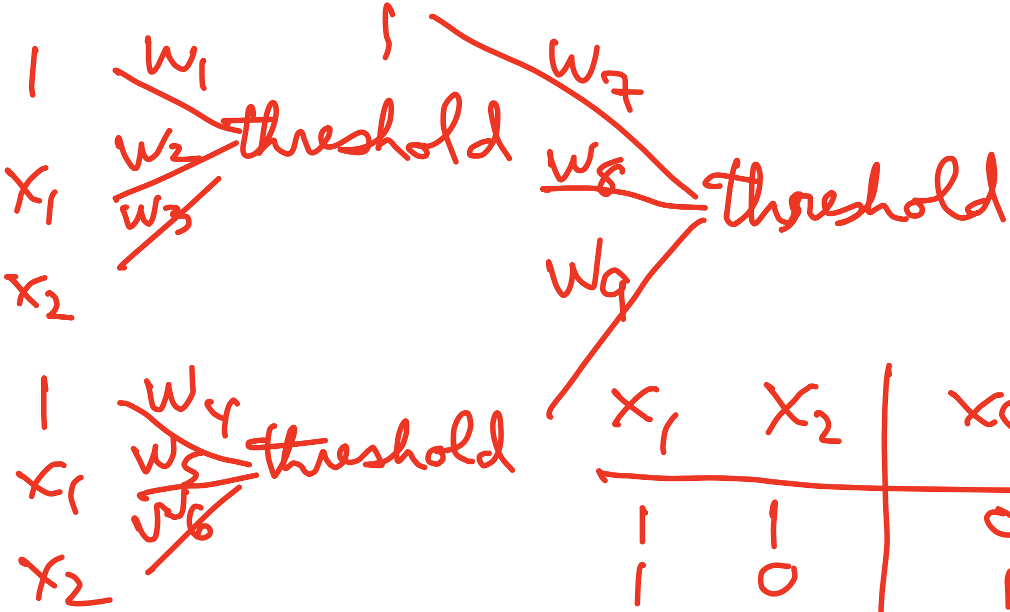
- Idea: Train a different classifier/predictor with a subset of the data on each core

- 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 nets that encode the exclusive-or Boolean function
- Averaging the weights yields new neural net that does not encode exclusive-or



x_1	x_2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

$w_1 =$	-0.5	-0.5	-0.5
$w_2 =$	1	-1	0
$w_3 =$	-1	1	0
$w_4 =$	-0.5	-0.5	-0.5
$w_5 =$	-1	1	0
$w_6 =$	1	-1	0
$w_7 =$	-0.5	-0.5	-0.5
$w_8 =$	1	1	1
$w_9 =$	1	1	1

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

Knowledge Distillation

- Technique to train a **small student network $\tilde{h}$** from a **large teacher network h**.
  - Can be used to compress an ensemble of networks into a single network
- Idea: minimize negative log likelihood of target y and cross entropy between teacher and student:

$$\min_{\tilde{h}} \sum_{(x,y)\in D} \left[-\log p_{\tilde{h}}(y|x) - \lambda \sum_{y'} p_h(y'|x) \log p_{\tilde{h}}(y'|x) \right]$$

Course Perception

- When you have a chance, please fill up the survey and provide feedback about the course (CS480/680) at

https://perceptions.uwaterloo.ca/