Data Mining & Machine Learning

CS37300
Purdue University

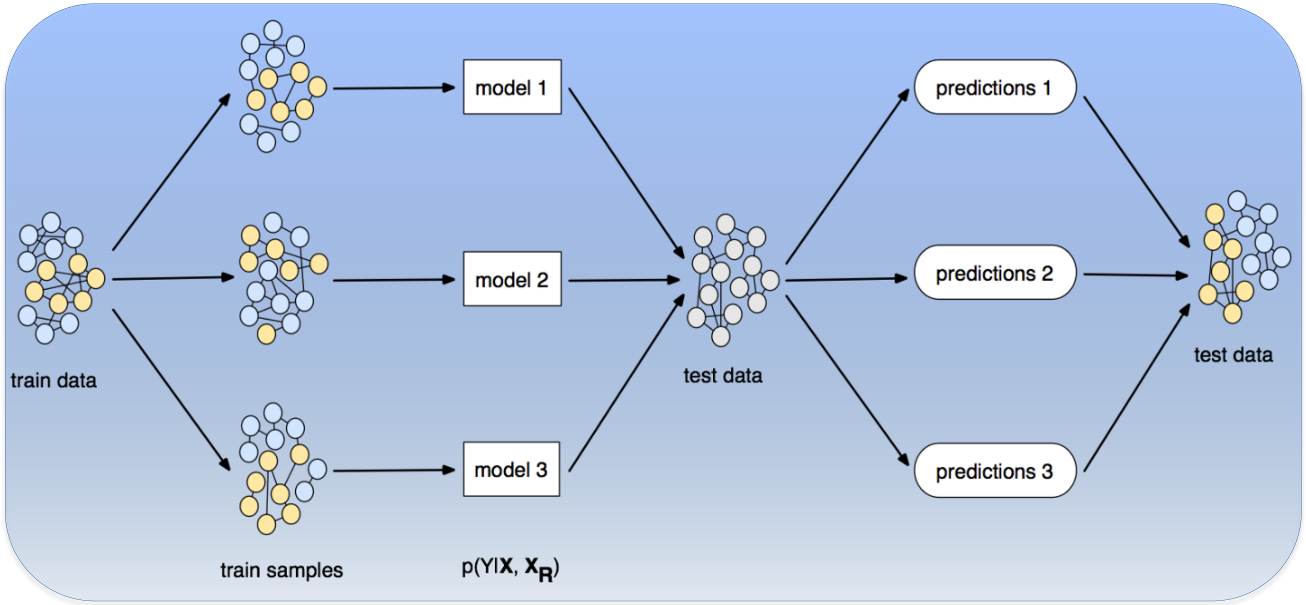
Oct 4, 2023

Ensemble methods

Boosting

- Main assumption
  - Combining many *weak* (but stable) predictors in an ensemble produces a *strong* predictor (i.e., reduces bias)
  - Weak predictor: only weakly predicts correct class of instances (e.g., tree stumps)
- Model space: non-parametric, can model any function if an appropriate base model is used

Boosting



TREATMENT OF INPUT DATA

- *reweight examples*

CHOICE OF BASE CLASSIFIER

- *weak predictor e.g., decision stump*

PREDICTION AGGREGATION

- *weighted vote*

Adaboost Algorithm

- Takes training data (x_i, y_i) (y -1 or 1), weights w_i
  - Initialize weights w_i to 1/n
- For $m=1..M$
  - Learn classifier f_m
    - $Error_m = \sum_{i=1}^{n} w_i^m \mathbf{I}\{f_{m(x_i)} \neq y_i\}$
  - Computer classifier coefficient $\alpha_m = \frac{1}{2}\log\frac{1-Error_m}{Error_m}$
  - Update weights $w_i^{m+1} = \frac{w_i^m \exp(-\alpha_m y_i f_m(x_i))}{\sum_{j=1}^{n} w_j^m \exp(-\alpha_m y_i f_m(x_i))}$
- Final classifier $f^*(x) = \operatorname{sign}(\sum_{m=1}^{M} \alpha_m f_m(x))$

Boosting Caveats

- While theoretically sound, Adaboost not that robust to noisy labels
    - Weights of mislabeled data grow until classifier fits the noise
- **Must** use *weak* classifiers
    - Otherwise easily overfits training data

Random Forests

- Problem: Decision Trees prone to overfitting
- Solution: Decision tree on fewer features
- Ensemble idea
  - Randomly select subsets of features
  - Choose best candidate split from just within subset
- Algorithm the same as standard decision tree, except instead of applying information gain / gini index / …, first randomly select subset, then apply
  - All features (except the one use) passed to the next level

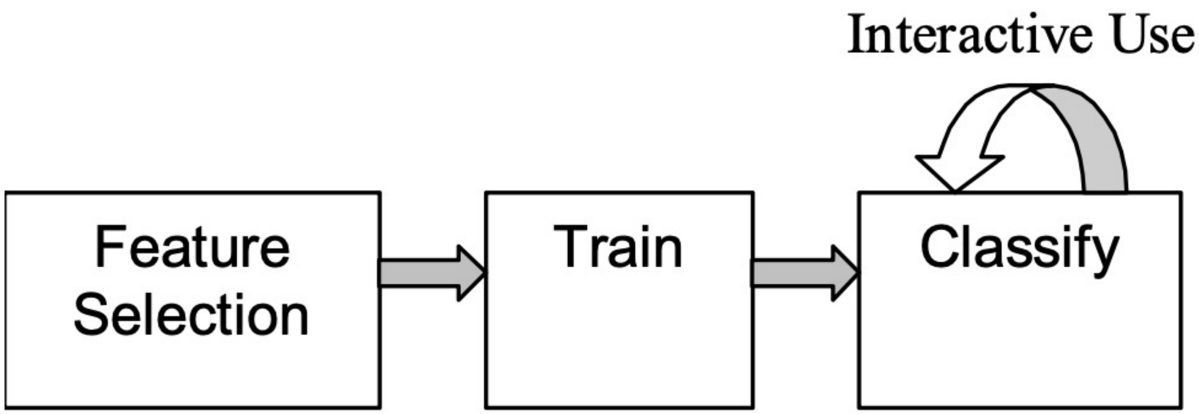
Ensemble summary

- Two approaches for Ensemble learning:
  - Boosting – reduce bias
  - Bagging – reduce variance
- Applicable in different situations

Interactive Machine Learning

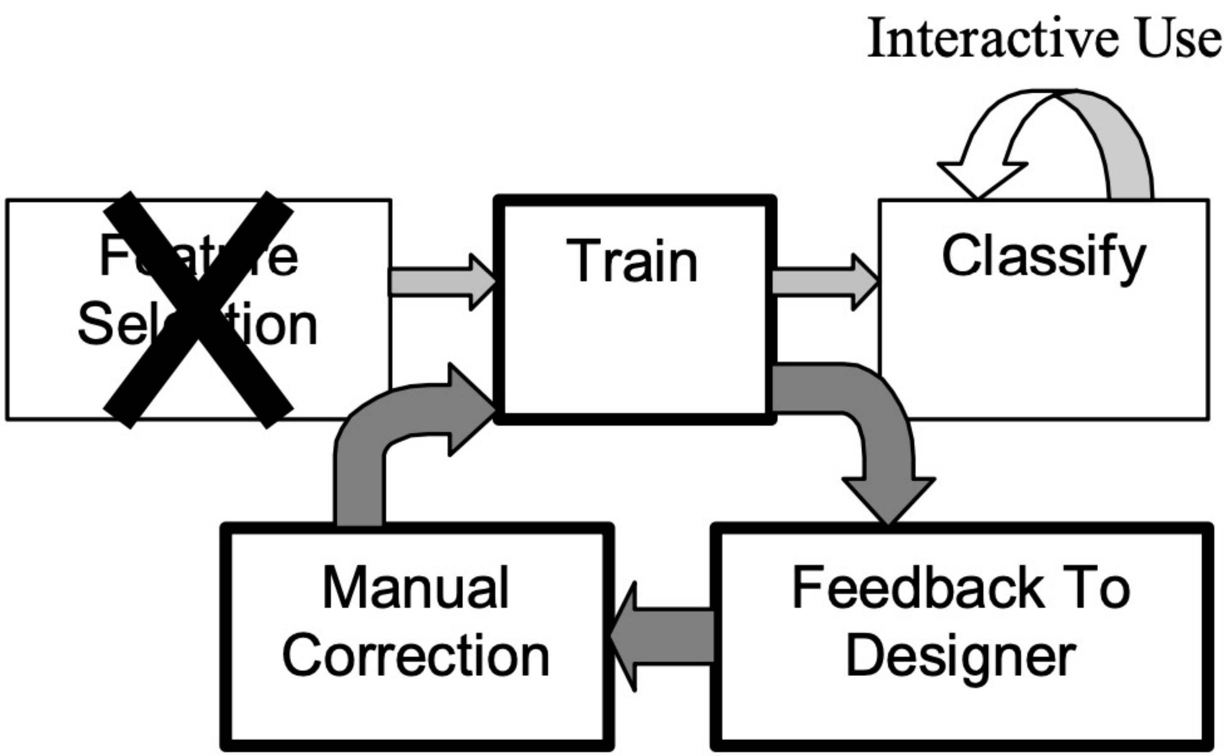
Classical Machine Learning

- Source: Interactive Machine Learning. *Fails* and *Olsen*.



Interactive Machine Learning

- Source: Interactive Machine Learning. *Fails* and *Olsen*.
- Goal: Create systems that allows quick feedback and quick training



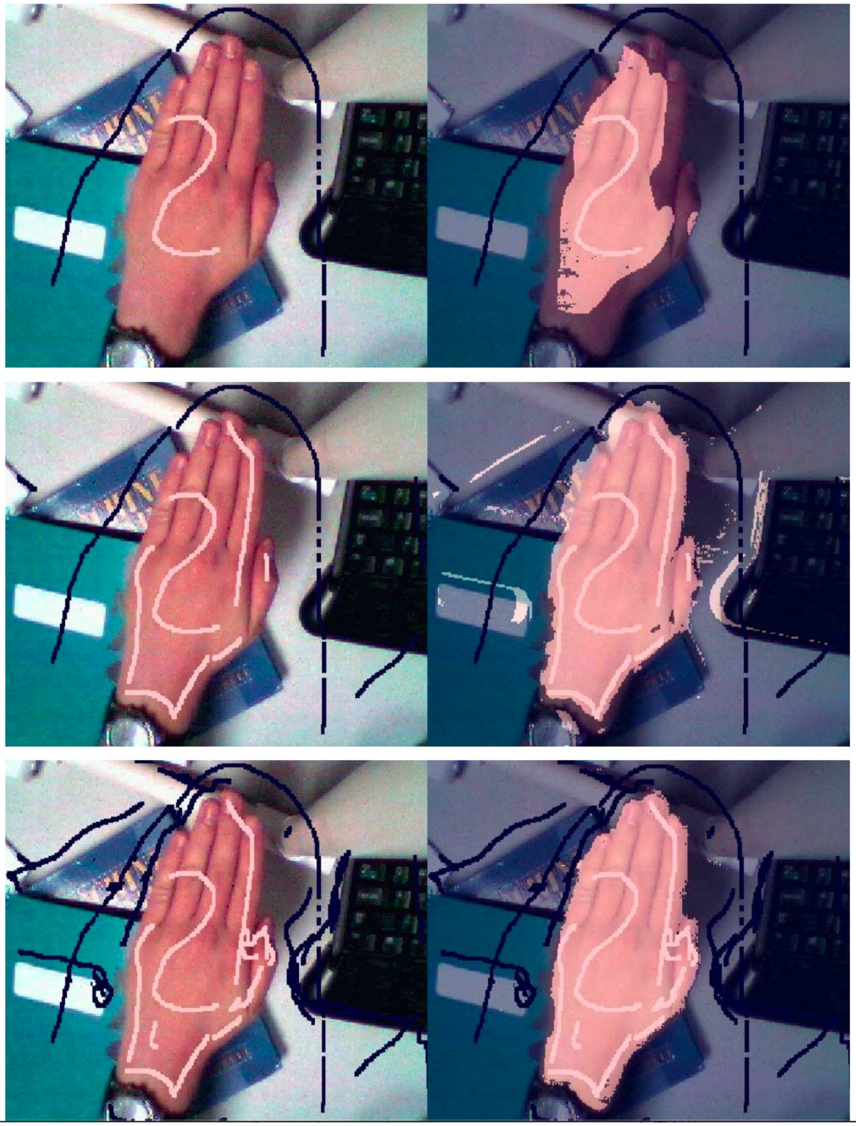
Side by Side comparison

- Classical setting
  - “Offline” training, time taken not very important
  - Limited training data
  - Feature selection and engineering
  - Systems components: Build ML pipeline, no User Interface (UI)
  - Explicit corrections for Bias/Variance
- Interactive setting
  - “online” training, needs to be done within seconds
  - Can have unlimited training loops
  - Limited focus on feature engineering
  - UI as important as machine learning model itself
  - Bias/Variance to be handled by the designer

Model choices for IML

- Neural Networks (too slow)
- K-NN (curse of dimensionality, especially for high-dimensional image datasets; too slow in classification)
- Boosted stumps (too slow)
- Decision trees: Fast re-training and classification, flexible modifications that help with speed of training and classification

Example



Active Learning

- Limited number of labels, several un-labelled examples available
- “Interaction” with a human involves the model *selectively* asking for additional labels
- Transductive Settings