ECS 607/766 Data Mining
Missing Data, Outliers & Ensembles

Dr. Ioannis Patras
EECS, Queen Mary University of London

Slide thanks: Tim Hospedales

Overview

- Recap
  - Anomalies
  - Association Rules
- Dealing with missing data
  - General options
  - Model-specific options
- Dealing with outliers
  - Filtering
  - Built-in
- Improving Performance with Ensembles
  - Bagging & Randomization
    - Decision Forests
  - Stacking
  - Boosting

+ Recap: Anomalies I

Anomaly Detection Algorithm

Train

- Input: Training data, {x}
- Compute the model (e.g., Gaussian u,S) that best explains/predicts the data {x}

Test

- Input: New example x and u,S
- Compute p(x;u,S)
- If p(x)<T, output: Anomaly
- Else, output: Ok

Supervised Learning Algorithm

Train

- Input: Training data, {x,y}
- Compute the model (e.g., MaxEnt, w) that predicts y given x.

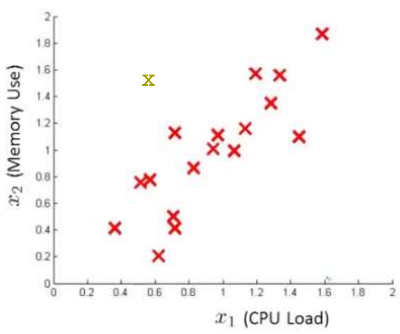
Test

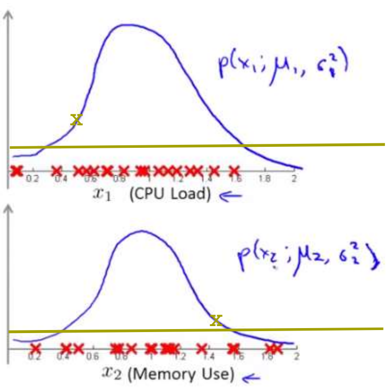
- Input: New example x
- Compute f(x)=$w^T x$
- If $w^T x>0$ output y=1
- Else output y=0.

+ Recap: Anomalies II

Algorithm:

Anomaly if p(x)<T

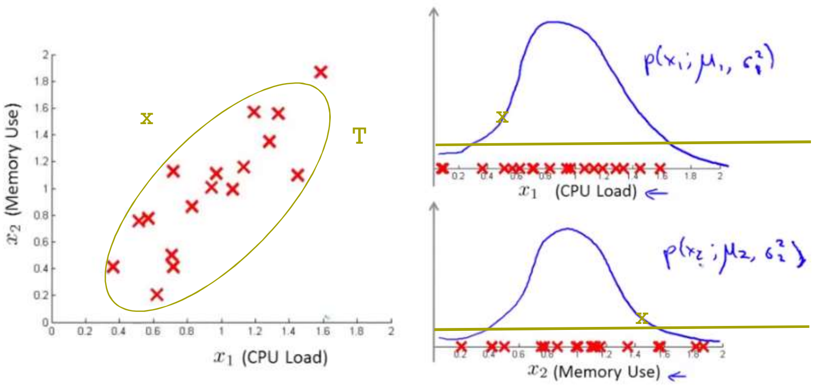




Recap: Anomalies II

Algorithm:

Anomaly if p(x)<T



Recap: Anomalies III

- Algorithm:
  - Anomaly if p(x) < T

$$p(\mathbf{x}) \propto \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u})^T S^{-1}(\mathbf{x}-\mathbf{u})\right) \Rightarrow p(\mathbf{x}) \propto \sum_k \pi_k \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u}_k)^T S_k^{-1}(\mathbf{x}-\mathbf{u}_k)\right)$$

Recap: Association Rules I

- What:

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples
{bread} ⇒ {milk}
{soda} ⇒ {chips}
{bread} ⇒ {jam}

- Why?
  - Marketing, Store Layout, Catalog Design

Recap: Association Rules II

- Challenge
  - Find all rules X=>Y where:
    - S(X=>Y) > MinSup
    - C(X=>Y) > MinConf
  - (frequent and confident)
- Support of Itemset s(X)
  - % transactions with itemset: N(X)/N
- Support of Rule S(X=>Y)
  - % transactions with X+Y: N(X+Y)/N
- Confidence of Rule: C(X=>Y)
  - Given you saw X, how confident about Y?
  - N (X+Y)/N (X)

- Apriori Algorithm
  - Recursively construct K+1 sized frequent itemsets from K sized frequent itemsets
  - (Non-frequent K-itemsets never need to be included)
  - (Most of the exponentially many itemsets are never considered)
  - Check for confidence

+ # Overview

- Dealing with missing data
  - General options
  - Model-specific options
- Dealing with outliers
  - Filtering
  - Built-in
- Ensembles
  - Bagging & Randomization
    - Decision Forests
  - Stacking
  - Boosting

+ # Missing Data

- Typically indicated by an out of range value
  - E.g., -1 for positive numbers like ‘weight’, NaN for real numbers, or ???
- Important to know why they may be missing
  - E.g., Equipment failure during measurement (random, systematic?)
  - E.g., Survey respondent declined to answer a question
  - E.g., In product purchased column: Did not buy versus, don’t know.
- Systematic missing may convey information and may be useful to record a categorical state for it.
  - E.g., decline to answer is information in itself!
- Significance depends on if missing at train or test time

Missing Data: Ideas?

- Recall: Most algorithms need fixed sized input with correspondence
  - Missing data is a problem.
  - Both at training and at testing
  - Any ideas for how to address this?

Car ID	Type	Make	Doors	Price
1	SUV	BMW	?	10,000
2	Coupe	Audi	?	20,000
3	Saloon	?	4	15,000
4	Estate	Ford	4	5,000

Missing Data: General options

Missing data strategies:

- 1. "Denial": Drop rows with missing values
  - +: Simple, fast. O(1).
  - -: If missing values are correlated, could loose key information
  - -: Less data (=> More overfitting, less accuracy)
  - -: Train time only
- In this example
  - Only one row left!
- Variant: If missing only in one column,
  - Drop column.

Car ID	Type	Make	Doors	Price
1	SUV	BMW	?	10,000
2	Coupe	Audi	?	20,000
3	Saloon	?	4	15,000
4	Estate	Ford	4	5,000

Missing Data: General options

Missing data strategies:

- 2. Treat missing data as special category
  - +: Simple, fast. O(1).
  - -: More useful if missing is significant.
    - 'Did not buy' as opposed to 'don't know'
  - -: Only for categorical attributes

Missing Data: General options

Missing data strategies:

- 3. "Average": Replace with mode (discrete) or mean (continuous)
  - +: Simple, fast. O(1)
  - -: If missing values are correlated, could loose key information
    - E.g., all Aston Martins are missing => Replace with mean price?
  - -: Oversimplistic, introducing new noise source
- In this example.
  - Make => Ford
    - (but what about price?)
  - Doors => 3
    - Does it make sense for SUV?

Car ID	Type	Make	Doors	Price
1	SUV	Ford	?	5,000
2	Coupe	Audi	2	35,000
3	Saloon	?	4	35,000
4	Estate	Ford	4	5,000

Missing Data: General options

Missing data strategies:

- 4. Conditional Average: Replace with mode (discrete) or mean (continuous) of same target value
  - +: Simple
  - -: Slow. O(N)
  - -: Only for classification
  - +: Likely better than naïve mean/mode replacement
  - -: Doesn't exploit inter-attribute correlation
- In this example.
  - Make => Audi
    - …Ok
  - Doors => 4
    - Does it make sense for Convertible?

Car ID	Type	Make	Doors	On Sale?
1	Convertible	Ford	?	Y
2	Coupe	Audi	2	N
3	Saloon	?	4	N
4	Estate	Ford	4	Y

Missing Data: General options

- 5. KNN & Impute
  - Find K nearest neighbors using available attributes
  - Use the mode/mean of those KNNs to fill in missing data.
  - +: Probably better than #4: Exploits inter attribute correlation
  - - : Slow/Non-scalable: O(N) <u>for each missing element</u>!
  - - : Depends on inter-attribute correlation
  - -: NN matching could be noisy.

+ Missing Data: General options

- 6. Cluster & Impute
  - Cluster the data using Kmeans
  - Associate to cluster
  - Use the mode/mean of those clusters to fill in missing data
  - Speed:
    - -: Slow. Clustering takes O(KN)
    - +: Scalable. Subsequent assignment takes O(K) per missing data
  - - : Depends on inter-attribute correlation
  - +: Probably more accurate

  - NOTE: Both steps of k-means need to be slightly altered.

+ Missing Data: General options

- 7. Setup a predictive model
  - Typically one predictive model f(IndependentVariables) -> Dependent
  - But you can setup a predictive model for any. E.g., classifier/regressor for:
    - f(Type, Doors,Price)->Make, f(Make,Price,Doors)-> Type, etc.
  - +: Probably most accurate
  - -: Very expensive. Need O(D) to O(D*2^D) classifiers
  - -:The predictive model may suffer also from missing values

Car ID	Type	Make	Doors	Price
1	Convertible	Ford	?	5,000
2	Coupe	Audi	2	35,000
3	Saloon	?	4	35,000
4	Estate	Ford	4	5,000

Missing Data: General options

- 7. Setup a predictive model
  - Typically one predictive model f(IndependentVariables) -> Dependent
  - But you can setup a predictive model for any. E.g., classifier/regressor for:
    - f(Type, Doors,Price)->Make, f(Make,Price,Doors)-> Type, etc.
  - +: Probably most accurate
  - -: Very expensive. Need O(D) to O(D*2^D) classifiers
  - -:The predictive model may suffer also from missing values
  - -: Depends on inter-attribute correlation.

Car ID	Type	Make	Doors	Price
1	Convertible	Ford	?	5,000
2	Coupe	Audi	2	35,000
3	Saloon	?	4	35,000
4	Estate	Ford	4	5,000

Missing Data: Model Specific

- Naïve Bayes
  - Train time missing:
    - Recall that train time procedure fits the Gaussian p(Feature|Category)
    - Simply fit this Gaussian using available data
  - Test time missing:
    - Recall that test time procedure is: $p(H \mid D_{1..N}) \propto \prod_i p(D_i \mid H)p(H)$
    - Simply take product only over visible feats
  - (This is not a hack!)
    - See www.youtube.com/watch?v=EqjyLfpv5oA
- DTs also have model specific solution

House ID	Rooms	Sq M	Built	Price
1	2	1000	1981	Low
2	4	?	2000	Med
3	?	5000	1700	High
4	3	3000	?	Med

Missing Data: Model Specific

- Naïve Bayes
  - Train time missing:
    - Recall that train time procedure fits the Gaussian p(Feature|Category)
    - Simply fit this Gaussian using available data
  - Test time missing:
    - Recall that test time procedure is: $p(H \mid D_{1..N}) \propto \prod_i p(D_i \mid H) p(H)$
    - Simply take product only over visible feats
  - (This is not a hack!)
    - See www.youtube.com/watch?v=EqjyLfpv5oA
- DTs also have model specific solution

House ID	Rooms	Sq M	Built	Price
1	2	1000	1981	Low
2	4	?	2000	Med
3	?	5000	1700	High
4	3	3000	?	Med

Missing Data: Model Specific

- Independent models/classifier/regressor for subsets of data dimensions (create an ensemble)
  - Train time missing:
    - Simply fit using available data
  - Test time missing:
    - Combine/fuse the decisions taken from all different classifiers/regressors

$$f(x) = \frac{1}{d} \sum_i f_i(x)$$

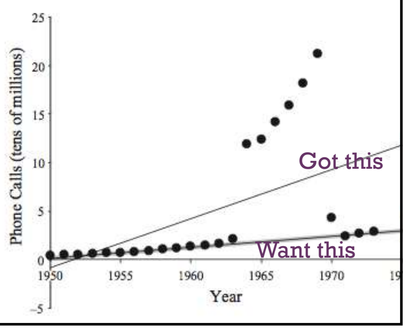
House ID	Rooms	Sq M	Built	Price
1	2	1000	1981	Low
2	4	?	2000	Med
3	?	5000	1700	High
4	3	3000	?	Med

Overview

- Dealing with missing data
  - General options
  - Model-specific options
- Dealing with outliers
  - Filtering
  - Built-in
- Ensembles
  - Bagging & Randomization
    - Decision Forests
  - Stacking
  - Boosting

Outliers: Problem

- What's the problem?
  - A single/few unusual example(s) dramatically distort the outcome
- E.g., Famous "Belgian phonecall" data
  - What happened?
  - # minutes rather than # calls recorded
- Conventional regression:
  - Minimise square deviation
  - => Vulnerable to outliers
- Ideas about how to deal with it?



Outliers: Preprocessing Methods

Options for filtering outliers: Anomaly Detection

- Use any anomaly detection method:
- Workflow:
  - Fit Gaussian or GMM.
  - Check likelihood under learned distribution
  - Exclude anomalous (highly unlikely) data
  - Then train supervised learner as usual on the remaining data

Outliers: Preprocessing Methods

Options for filtering outliers: Anomaly Detection

- Use any anomaly detection method
  - Exclude highly unlikely data
- How to know what counts as <u>too unlikely</u>?
  - 1. If Gaussian, know how many unlikely points to expect. Where there are unusual number, prune them.
    - -: Only correct if data is truly Gaussian
  - 2. If non-Gaussian, take top K or top K% most unlikely.
    - -: Maybe outlier, or maybe rare but important
    - => Could loose key data
  - 3. Solution?
    - Crossvalidate (can be expensive)

+

Outliers: Built-in methods: Robust Regression

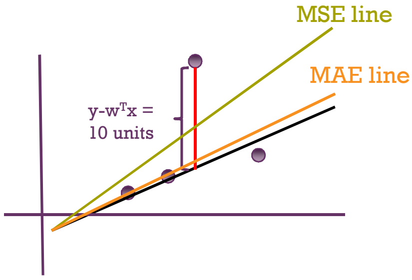
Robust Regression

$$E_{MSE}(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- Recall, regular regression minimises:
  - Instead of minimising the square deviation, minimise the absolute value deviation.

$$E_{MSE}(\mathbf{w}) = \sum_i \left| y_i - \mathbf{w}^T \mathbf{x}_i \right|$$

- MSE:
  - 10 unit deviation = 100 penalty
- MAE:
  - 10 unit deviation = 10 penalty
- MAE:
  - No closed form solution.
  - …have to use gradient



+

Outliers: Case Study: EECS Research ☺

- May have come across flickr "interestingness"
- Goal is to learn content-based: video interestingness, image interestingness, age regression, etc.
  - Mechanical Turk Crowd sourced data => label noise.
- Key mechanism is MAE robust regression.
- [ECCV, 2014, IEEE PAMI 2015]

$$E_{MSE}(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

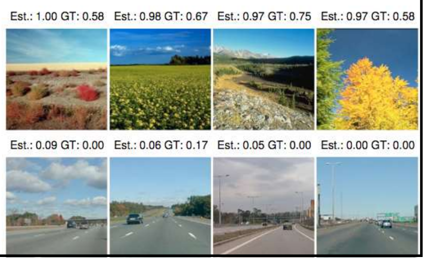
$$E_{MSE}(\mathbf{w}) = \sum_i \left| y_i - \mathbf{w}^T \mathbf{x}_i \right|$$

Outliers: Case Study: EECS Research ☺

- May have come across flickr "interestingness"
- Goal is to learn content-based: video interestingness, image interestingness, age regression, etc.
  - Mechanical Turk Crowd sourced data => label noise.
    - Q: Contrast outlying features?
- Key mechanism is MAE robust regression.
- [ECCV, 2014, IEEE PAMI 2015]

$$E_{MSE}(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

$$E_{MSE}(\mathbf{w}) = \sum_i |y_i - \mathbf{w}^T \mathbf{x}_i|$$



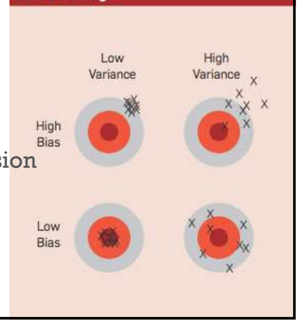
Overview

- Dealing with missing data
  - General options
  - Model-specific options
- Dealing with outliers
  - Filtering
  - Built-in
- Ensembles
  - Bagging & Randomization
    - Decision Forests
  - Stacking
  - Boosting

+ Ensembles: Big Picture

- What is the source of error in supervised learning?
  - We have seen under/over fitting.
  - Another way to understand this: "Bias-Variance Tradeoff".
- Bias: Tendency of a model to learn the same wrong thing.
  - E.g., if trying to fit a linear model to a curve.
- Variance: Tendency of a model to learn random fluctuations independent of the underlying signal.
  - E.g., Every time you fit a complicated model like decision tree, you can get quite a different tree.
- Ideal: Zero bias + zero variance

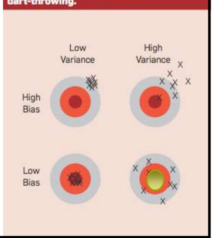
Figure 1. Bias and variance in dart-throwing.



+ Ensembles: Big Picture

- We saw complexity control by cross-validation tries to find a good trade-off between under & over-fitting (bias & variance).
- Ensembles provide a way to further reduce variance.
- Key idea:
  - Committee of experts collective opinion more reliable than individual expert.
  - => Collection of models' opinion more reliable than individual model.
- Intuition:
  - If each expert's errors are uncorrelated.
  - Then their collective vote averages out their errors.
- Individual expert/model is high variance.
  - Committee/ensemble is lower variance

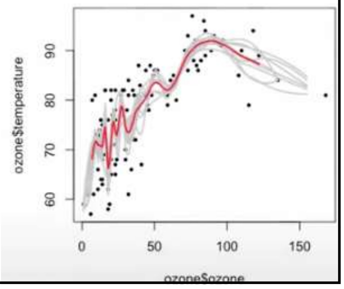
Figure 1. Bias and variance in dart-throwing.



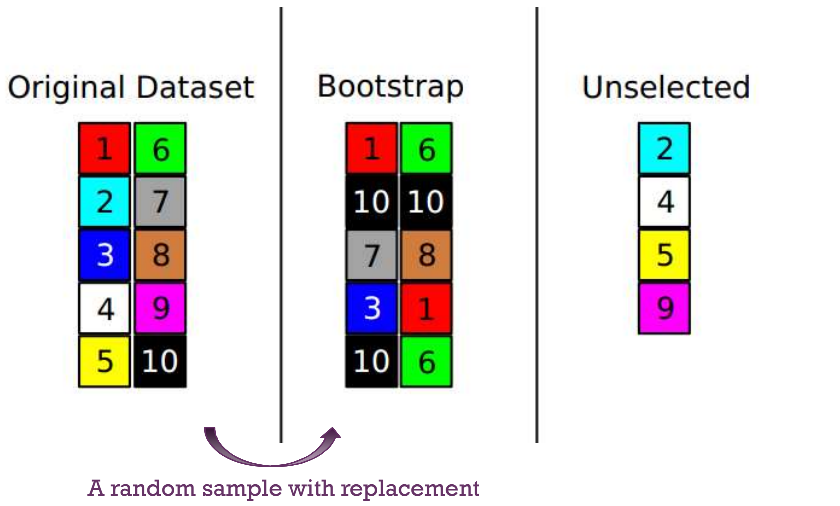
Ensembles: General Methods: Bagging

- Ensembles: useful if diverse & uncorrelated predictions
  - Q: Ideas about how to generate models with diverse predictions?
- Bootstrap Aggregation ("Bagging")
- Method:
  - For N data, take K random samples size N with replacement.
  - Train K models, one for each subsample
  - Average or majority vote the predictions
- E.g., given models $f_1(x)$, $f_2(x)$, .. Etc

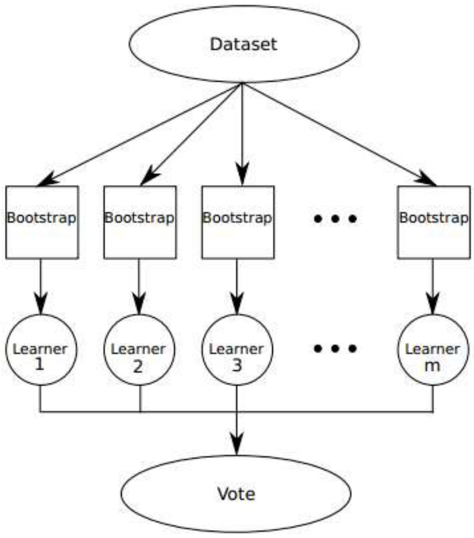
$$f_{bag}(x) = \frac{1}{E}\sum_{e} f_e(x)$$



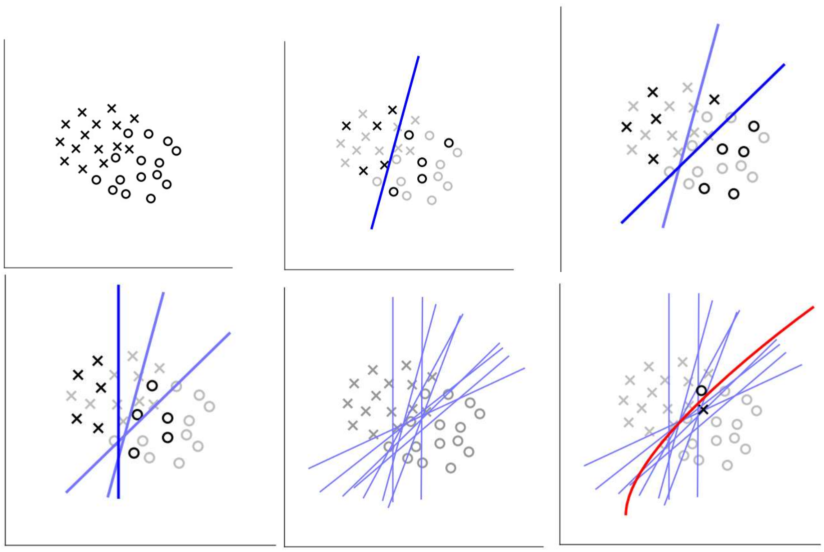
Ensembles: General Methods: Bagging: Bootstrapping



Ensembles: General Methods: Bagging

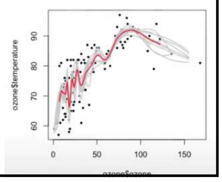


Bagging Illustration



Ensembles: General Methods: Bagging

- Bootstrap Aggregation ("Bagging")
  - Train K models on K subsamples, and average.
- Notes:
  - Requires "unstable" (non-linear) base models: where each can be very different: E.g., decision trees (=> expert diversity)
  - Each ensemble member likely worse than a full model
    - But collectively better!
  - Ensemble will have similar bias
    - … but reduced variance
- Tradeoff:
  - Small bags, worse models, more diversity
  - Big bags, better models, less diversity



Ensembles: General Methods: Random Subspaces and Model Combination

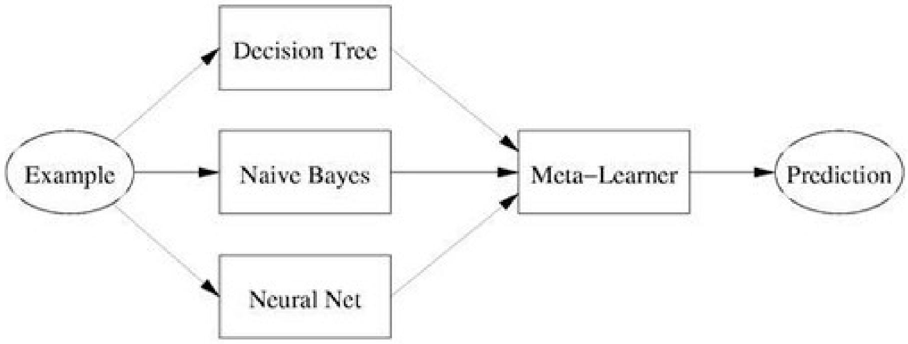
- So far we introduced expert diversity by bagging.
- Bagging randomizes instances:
  - Each expert trained on a random subset of instances
- Q: What else can we randomize?
- "Random Subspaces" randomizes attributes:
  - Each expert trained on a random subset of attributes/dimensions
- Can also do so by any other kind of diversity.
  - E.g., For methods that converge to a local minima only, start in different initial conditions.
- "Model Combination": combine many models: decision tree, logistic regression, naïve bayes, KNN, etc.

Ensembles: General Methods: From Bagging to Stacking

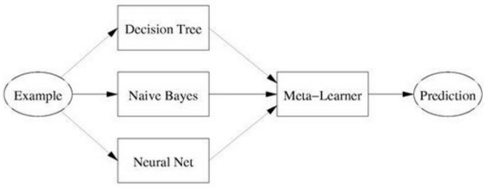
- So far we said average/vote each model/expert $f_{ensemble}(x) = \frac{1}{E}\sum_e f_e(x)$
- Q: What could we do better?
- What if some experts are better than others?
  - Could we do better with a weighted vote? $f_{ensemble}(x) = \frac{1}{E}\sum_e w_e f_e(x)$
- How to determine the per-expert weight?
  - Run another "meta" learner!

Ensembles: General Methods: Stacking

- Example:



Ensembles: General Methods: Stacking



- Base Learners
  - Input: Raw data
  - Output: Class
- Meta Learner
  - Input: Vector of estimated classes from a bank of base learners
  - Output: Class
- Overall, making a complex super-model
  - Very complex model => easy to overfit
  - So keep meta-learner simple: Use linear model
  - Train meta-learner using cross-validation to reduce overfitting
    - Important because individual classifiers may be overfit/unrealistically confident

Case Study: Netflix

- Netflix Prize:
  - Famous Movie recommendation challenge
  - First team to reach a specified test accuracy wins $1M.
- Winner used >80 top ranked models with stacking.
- Netflix didn't use the winning algorithm: too expensive!

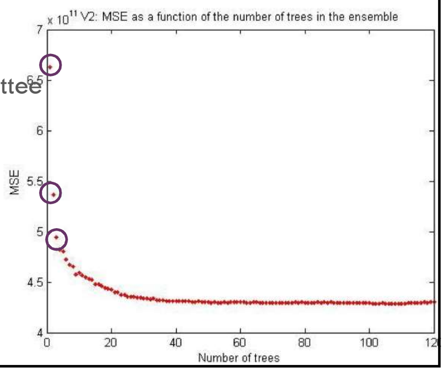
Ensembles: Model Specific: Random Forests

Random Forest

- Grow a whole array of decision trees
  - Average their prediction at test time
- Diversity: Each decision tree randomized with both instances (bagging) and dimensions (random subspaces)
- Pros:
  - Generally great accuracy. State of the art for many problems.
- Cons:
  - Decision trees usually interpretable, forests not.
  - Can be expensive to train many trees (but very parallelizable)

Ensembles: What sized committee to use?

- Tradeoff:
  - Larger committee => better aggregate decision
  - Larger committee => slower to train & test
    - May have real-time test constraints
- More diverse and uncorrelated
  - Good performance with small committee



+ Case Study: Kinect

- Uses Decision Forest ensemble.
- Training the forest
  - 1 million images
  - Depth 20 trees, 2000 random features per tree, 300k images per tree
  - Use distributed implementation: 1 day on 1000 core cluster.

- Shotton et al, IEEE CVPR 2011

+ Ensembles: Boosting: Motivation

- Ensemble methods take a (weighted) sum of model predictions

$$f_{ensemble}(x) = \frac{1}{E}\sum_{e} w_e f_e(x)$$

- Each model $f_e(x)$ is trained independently.
- Rely on Instance/Feature or other randomization to generate diversity among the experts
- Boosting: Seek to explicitly learn complementary models

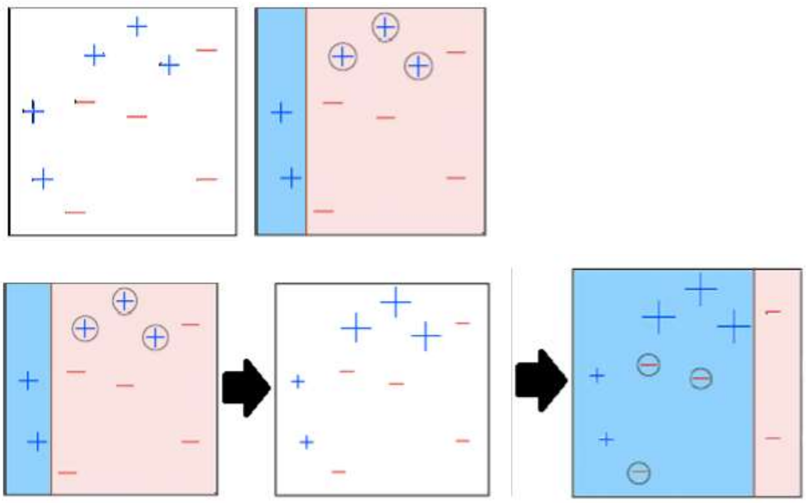
Ensembles: Boosting: Mechanism

Boosting: Algorithm Sketch

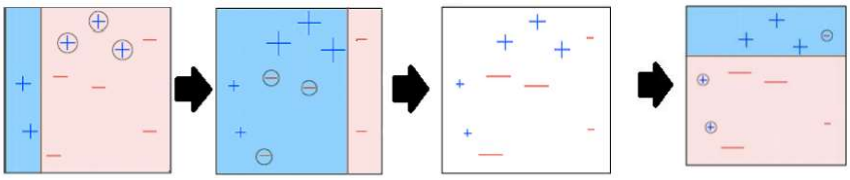
$$f_{ensemble}(x) = \frac{1}{E}\sum_{e} w_e f_e(x)$$

- Train the first model $f_{i=1}(x)$
- Repeat i=2…T
  - Note which data instances the ensemble so far $f_1(x)\ldots f_{i-1}(x)$ gets wrong
  - Train next model $f_i(x)$, but focus on training examples currently predicted wrongly
- Basic models:
  - Now forced to be different, unlike bagging
  - Commonly they are decision stumps
  - Needs to be able to accept instance weights
- Typically boosting builds complex model out of many simple models, instead of out of many complex models (bagging)

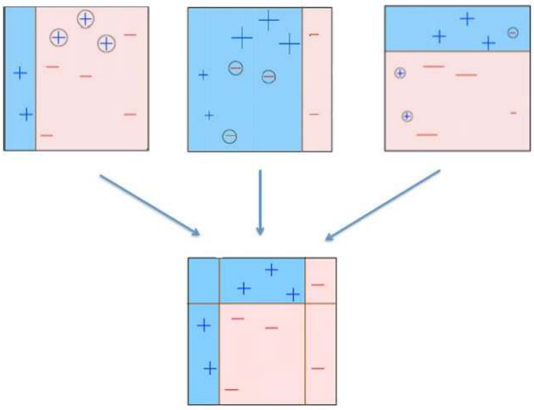
Boosting: Example



Boosting: Example



Boosting: Example



Ensembles: Boosting: Properties

- Ensembles: Weighted sum of model predictions

$$f_{ensemble}(x) = \frac{1}{E}\sum_e w_e f_e(x)$$

- Contrast: Bagging & Randomisation
  - Trained independently with randomisation for diversity
- Contrast: Boosting
  - Trained sequentially to explicitly seek complementarity.
- Boosting Properties
  - Often best performing ensemble type, but can sometimes overfit, since tuning **w** (whole ensemble makes a complex classifier)
  - Can't parallelize the whole thing like independently trained ensembles
- Neat (surprising?) fact: Even if each base learner only 51% accurate, the entire ensemble can be arbitrarily (100%) accurate

Aside: Boosted Cascades + Case Study: Face Detection

- Almost all embedded face detectors use boosting. Why?
  - Recall: Boosting builds an ensemble in order: first models will be most useful, later models will "fine-tune".
- Detection Issue: "Sliding window", means very many classifications needed at test time. 1000x1000 pixel image => 1M-1B classifications.
- Variant: Boosted cascade.
  - Builds an ensemble that also prefers to put cheaper classifiers first.
  - Test time: Evaluate them in order: 1…. E
  - If you are confident that its not a face
    - Then terminate early
  - Most squares are non-face and << full cost

$$f_{ensemble}(x) = \frac{1}{E}\sum_e w_e f_e(x)$$

Case Study: Face Detection

- Almost all embedded face detectors use boosting
  - Recall: Boosting builds an ensemble in order.
- Variant: Boosted cascade.
  - Builds an ensemble that also prefers to put cheaper classifiers first.
  - Test time: Evaluate them in order: 1…. E
  - If you are confident that its not a face
    - Terminate early
  - => Most squares don't evaluate whole ensemble
  - => Most squares use << full cost

$$f_{ensemble}(x) = \frac{1}{E}\sum_e w_e f_e(x)$$

Bagging versus Boosting

Bagging

- Built independently / parallel
- Resample data
- Reduces variance
  - Typically works with complex models
- Parameters:
  - How many members
  - How to introduce diversity?

$$f^{bag}_{ensemble}(x) = \frac{1}{E}\sum_e f_e(x)$$

Boosting

- Built sequentially
- Reweight data
- Reduces variance
  - Typically also used to reduce bias by combining simple models
- Parameters:
  - Termination condition

$$f^{boost}_{ensemble}(x) = \frac{1}{E}\sum_e w_e f_e(x)$$

+ Ensembles: Summary

- Pros
  - Improves accuracy, often a lot!
  - Bagging & Subspace randomisation can make overfitting in the individual models less of an issue: (The overfits "average out")
- Cons
  - More models to train => More expensive
    - (Maybe parallelizable, except boosting)
  - Loss of interpretability
  - Test time is more expensive (have to evaluate many models)
    - Except boosted cascades
  - Boosting is weak to label noise & outliers