PERIODIC VARIABLES

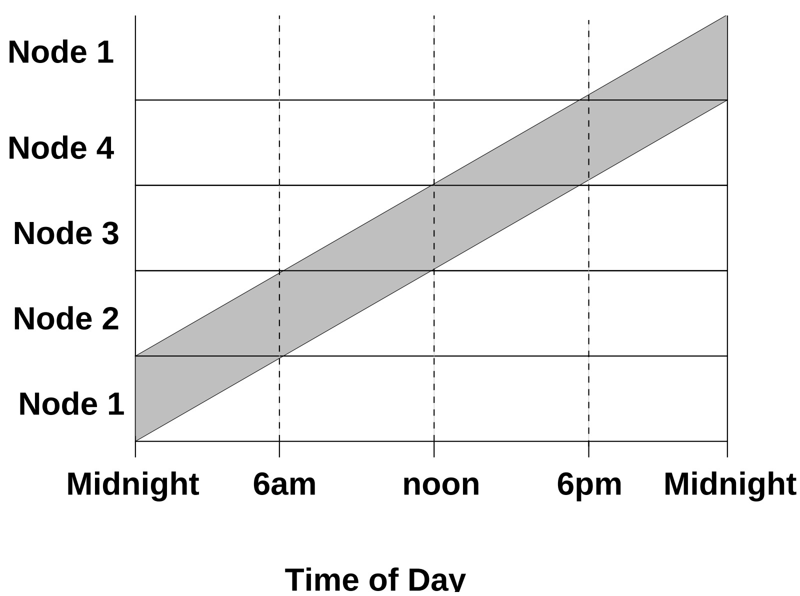
- The wind blowing from a direction of 1 degree is very close to a wind from 359 degrees

- Dec 31 (day 365) of one year should have a representation close to Jan 1 (day 1) of the next year.

- Avoid ‘‘Representational Cliffs’’

1. Use a periodic function like $sine$.

2. Use an interpolation representation.

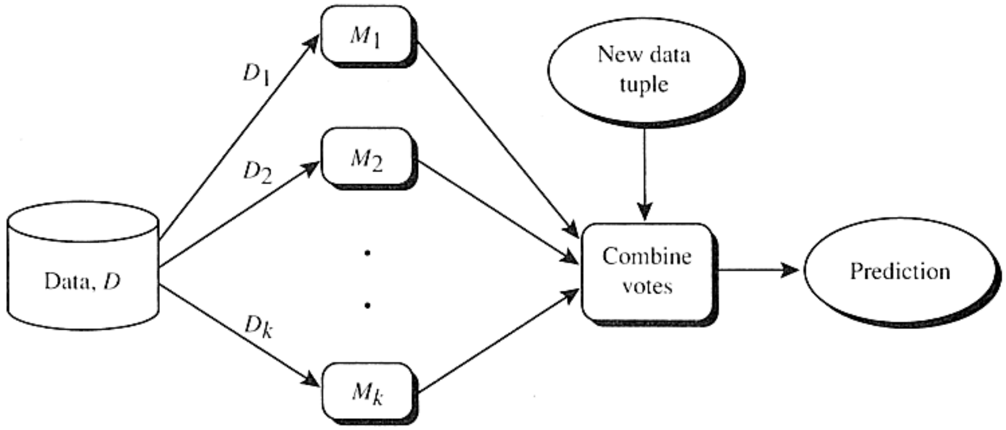


DEEP NETWORKS DEEP LEARNING

- 10-20 Layers, hundreds of nodes in a layer
- Convolutional Networks for image classification and object detection (CNN)
- Long Short Term Memory (LSTM)
- Generative Adversarial Networks (GAN)
- Recursive Neural Networks
- Images, voice, music
- Unsupervised feature leaning followed by classification
- Software: Tensorflow (Google), Caffe, Theano, Torch, Keras, Neuro Studio
- GPU Processors
- The basic procedures for deep learning are the same as for JavaNNS

COMBINING MULTIPLE MODELS

- Ensembles
- The opinions of several experts will be better than a single one
- In data mining we can combine the output of several classifiers
- Combination Schemes
  - Bagging
  - Boosting
  - Stacking
- Advantage: Improves predictive performance
- Disadvantage: Hard to analyze decision making
- Can be used for
  - Classification
  - Numeric Prediction
- WEKA Meta Algorithms



- A series of k learned models (Base classifiers) $M_1, M_2, ...M_k$

- learned from a series of datasets $D_1, D_2, ...D_k$

- give a combined model $M*$ which classifies a new instance by a voting procedure

Bagging

- Bagging = Bootstrap Aggregating
- Combine predictions by voting/averaging
  - Simplest way
  - Each model receives equal weight
- Method
  - Create k training sets of size d by sampling with replacement
  - Build a classifier for each training set
  - Combine k predictions by majority voting
- If learning scheme is unstable bagging almost always improves accuracy
- Unstable means
  - Small changes in training data can make a big difference in the model
  - Eg Decision trees
  - Curiously, might want to force insability

BOOTSTRAPPING

- Bootstrapping means proceeding without external input. (Baron Munchhausen pulled himself out of a swamp by his bootstraps)

- A method for estimating generalization error, similar to cross-validation

- Usually used for a small number of examples

- The 0.632 bootstrap. For a data set of size n

  - Randomly pick n examples with replacement for new training set
  - Some examples will be repeated
  - Some examples will not be used
  - Put unused examples in test set

- Probability of not being picked for training is
  $(1 - \frac{1}{n})^n \sim e^{-1} = 0.368 \qquad (e = 2.7183)$

- Probability of being in training set is 0.632

- $error = 0.632 \times error_{test} + 0.368 \times error_{train}$

BAGGING ALGORITHM

Input:

D a set of d training examples

k number of models

a classification scheme (eg. Decision tree)

Output:

The ensemble $M*$

Model generation

Build k models:

Create bootstrap sample D_i by sampling d instances with replacement from training data

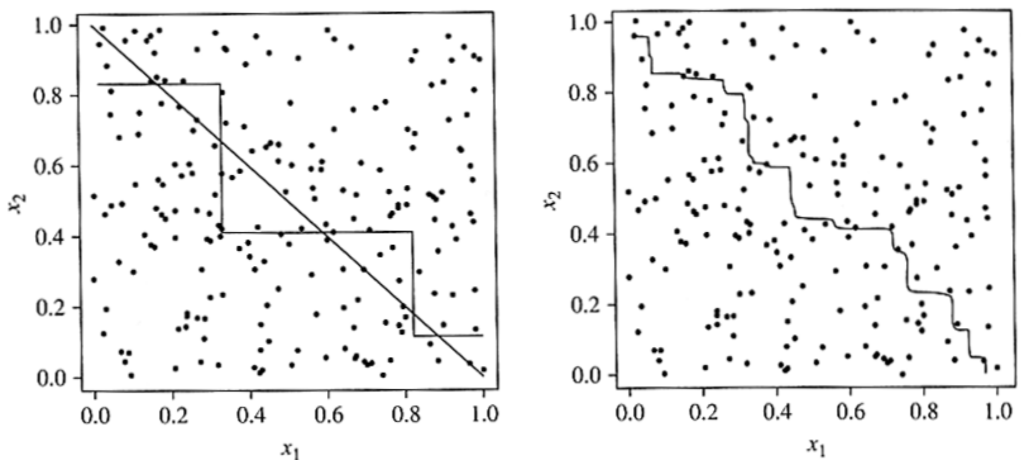
Use D_i with the learning scheme to create M_i

Classification of new instance

For each of the M_k models:

Predict class of instance using model.

Return class that has been predicated most often.



(a) (b)

- Synthetic data
- Diagonal line in (a) is true boundary
- Step in (a) is boundary of one decision tree
  Why is it a step?
- Boundary from meta classifier in (b) is closer to true boundary

BIAS-VARIANCE DECOMPOSITION

- Analyse how much any specific training set affects performance
- Assume infinitely many classifiers built from different training sets of size d
- For any learning scheme
  - Bias = Expected error of the due to choosing a bad classifier
  - Variance = Expected error due to the training set used
  - Target shooting analogy
- Total expected error: bias + variance
- Note, when selecting a learning method, low variance might come with high bias or vice versa

BAGGING BENEFITS/DRAWBACKS

- Bagging reduces variance by voting/averaging the error
- Addresses instability problem, reduces over-fitting
- Improved accuracy is not guaranteed. Sometimes combined decisions are worse
- Good for noisy data
- Usually, the more classifiers the better
- Use for numeric prediction by averaging predicitions

BOOSTING

- Also uses voting/averaging
- Weights models according to performance
- Weights examples by difficulty
- Iterative, new models are influenced by performa of previously built ones
  - Encourage a new model to become and ‘expert’ for instances misclassified by earlier models
  - Intuitively models should be experts that complement each other
- Kearns question, 1988 [Learning Theory]: Can a set of weak learners create a single strong learner?
- Weak learners, or simple learners, are models with weak correlation with the true classes, eg OneR.
- Several Variants, popular is Adaboost (Adaptive Boost); LogitBoost is more sophisticated

ADABOOST ALGORITHM

```
Input:
    D a set of d training examples
    k number of models
    a classification scheme (eg.  Decision tree)
Output:
   The ensemble M*
```

Model generation

Assign weight of $1/d$ to each training instance

For each of k iterations:

  Sample D with weights and replacement, get D_i

  Use training set D_i to get model M_i

  Compute $error(M_i)$ of model on weighted dataset (later)

      If $error(M_i) >= 0.5$

         Go back to sample.

     For each instance in dataset:

         If instance in D_i classified correctly by model

             Multiply weight by $error(M_i)/(1-error(M_i))$

     Normalize weight of all instances (later).

Classification

Assign weight $w_i = 0$ to all classes.

For each of the k (or less) models:

  $w_i = log(\frac{1-error(M_i)}{error(M_i)})$

  c = class prediction for example from M_i

  Add w_i to weight for c

Return class with highest weight

BOOSTING - LEARNING

- Weighted error:
  $error(M_i) = \sum_{j=1}^{d} w_j \times err(Instance_j)$

  $err(Instance_j) = 1$ if misclassified, else 0
- Normalizing weights:

  $weight \leftarrow weight \times \frac{\sum Weight(old)}{\sum Weight(new)}$
- Equal weights to start
- Weight of correctly classified instances decreases
- Weight of misclassified instances increases

  $weight \leftarrow weight \times (\frac{error(M_i)}{1-error(M_i)})$

- Weight update does not apply to correctly classified instances
- Normalization does and decreases the weight
- If error is zero weight of all instances will be zero
- Needs a learning method that can deal with weighted instances, eg C4.5/J48

BOOSTING - CLASSIFICATION

- [Note there are weights for instances and weights for classifiers]

- There will k or fewer models (if early terminati

- The influence of a model on a new instance is

  $weight \leftarrow= -log(\frac{error(M_i)}{1-error(M_i)})$

- A classifier that performs well ($error(M_i)$ close to 0) receives a high weight

- A classifier that performs poorly ($error(M_i)$ close to 0.5) receive a low weight

- A weight is a positive number between 0 and infinity.

- The weights of all classifiers that vote for a particular class are summed.

- The class with the highest sum is assigned.

BOOSTING - UNWEIGHTED DATA

- What if the learning algorithm can not deal with weighted instances?

- Use the same techniques that bagging uses, resampling the unweighted dataset to form a weighted dataset.

- The instances with high weight have more duplicates.

- The instances with low weight may not be selected.

- The new dataset should have the same size as the original dataset.

- The other processes remain unchanged.

- However one can continue the iterations even when the error exceeds 0.5, simply by re-generat a new dataset using a different random seed.

BAGGING VS BOOSTING

- Both use voting for classification or averaging for numeric predication to combine multiple models.
- Both combine models of the same type, e.g. decision trees.
- Boosting is iterative. Bagging is not.
- In bagging individual models are built separately. In boosting a new model is influenced by the previous models.
- Boosting encourages new models to become experts for instances handled incorrectly previously.
- Boosting weights the contribution of a model by its performance. Bagging gives equal weight to all models.

STACKING

- To combine predictions of base learners, don't vote, use *Meta learner*
  - Base learners: *level-0 models*
  - Meta Learner: *level-1 models*
  - Predictions of base learners are input to meta learner
- Base learners are usually different schemes
- Can't use predictions on training data to generate data for level-1 model
  - Use a cross-validation scheme
- Hard to analyze theoretically, 'black magic'

RANDOM FOREST

- A variation of bagging
- Build the base classifier by C4.5/J48 algorithm BUT choose split attribute at random
- Choose class of new instance by majority voting
- Usually an accurate classifier