Classification for High Dimensional Problems Using Bayesian Neural Networks and Dirichlet Diffusion Trees

Rafdord M. Neal and Jianguo Zhang

Presented by Jiwen Li

Feb 2, 2006

Outline

- Bayesian view of feature selection
- The approach used in the paper
- Univariate tests and PCA
- Bayesian neural networks
- Dirichlet diffusion trees
- NIPS 2003 Experiment Result
- Conclusion

Feature selection, why?

- **Improve learning accuracy.**

  Too many features cause overfitting for maximum likelihood approaches, but may not for Bayesian methods.
- **Reduce the computational complexity.**

  It is especially a problem for Bayesian methods. But Dimensionality can be reduced by other means such as PCA.
- **Reduce the cost of measuring features in the future.**

  To make an optimal tradeoff by balancing the costs of feature measurements and the prediction errors.

The Bayesian Approaches

- **Fit the Bayesian model, include your beliefs into the prior by**

$$P(\theta \mid X_{train}, Y_{train}) = \frac{P(\theta)P(Y_{train} \mid X_{train}, \theta)}{\int P(\theta)P(Y_{train} \mid X_{train}, \theta)d\theta}$$

  Note: the model could be complex, and uses all the feature.

- **Make predictions using the Bayesian model on the test cases by**

$$P(Y_{new} \mid X_{new}, X_{train}, Y_{train}) = \int P(Y_{new} \mid X_{new}, \theta)P(\theta \mid X_{train}, Y_{train})d\theta$$

  Predictions are found by integrating over the parameter space of the model.

- **Find the best subset of features based on the posterior distribution of the mode parameters, and the cost of features used.**

  Note: Knowing the cost of measuring the feature is essential for making the right trade-off.

Using feature construction instead of feature selection (1/3)

- Use a learning method that is invariant to rotations in the input space and which ignores inputs that are always zero.
- Rotate the training cases so that only n inputs are non-zero for the training cases, then drop all but one of the zero inputs.
- Rotate test cases accordingly, setting one input to the distance from the space of training cases.

Using feature construction instead of feature selection (2/3)

Use a learning method that is invariant to rotations in the input space and which ignores inputs that are always zero.

Example: The Bayesian logistic regression model with spherically symmetric prior.

$$P(Y_i = 1 \mid X_i = x_i) = \left[1 + \exp(-(\alpha + \sum_{j=1}^{n} \beta_j x_{ij})) \right]^{-1} \qquad (1)$$

while β has a multivariate Gaussian prior with zero mean and diagonal covariance.

Given any orthogonal matrix R, doing linear transform

$X'_i = R X_i$, $\beta' = R^{-1} \beta$

since $\beta'^T X'_i = \beta^T X_i$, $R^{-1}\beta$ has no effect on probabilities in (1).

Using feature construction instead of feature selection (3/3)

Rotate the training cases so that only n inputs are non-zero for the training cases, then drop all but one of the zero inputs.

Example: The Bayesian logistic regression model with spherically symmetric prior.

There always exist an orthogonal transformation R, for which all but m of the components of the transformed features RX_i are zero in all m training cases.

PCA is an approximate approaches to doing this transformation. It projects X_i onto the m principal components found from the training cases, and projects the portion of X_i normal to the space of these principle components onto some set of (n – m) additional orthogonal directions. For the training cases, the projections in these (n - m) other directions will all approximately be zero, so that X'_{ij} will be approximately zero for j > m. Clearly, one then need only compute the first m terms of the sum in (1)

The approach used in the paper

1. Reduce the number of features used for classification to no more than a few hundred, either by selecting a subset of features using simple univariate significance tests, or by performing a global dimensionality reduction using PCA on all training, validation and test set.

2. Apply a neural network based on Bayesian learning as a classification method, using an ARD prior that allows the model to determine which of these features are more relevant.

3. If a smaller number of features is desired, use the relevance hyperparameters from the Bayesian neural network to pick a smaller subset.

4. Apply Dirichlet diffusion trees (an Bayesian hierarchical clustering method) as classification methods, using an ARD prior that allows the model to determine which of these features are most relevant.

Feature selection using univariate tests

An initial feature subset was found by simple univariate significance tests.

Assumption: Relevant variables will be at least somewhat relevant to the target on their own.

Three significance tests were used:

- Pearson Correlation
- Spearman Correlation
- A runs test

A p-value is calculated by permutation test.

Spearman correlation

- Definition: A linear correlation to cases where X and Y are measured on a merely ordinal scale.
- Formulary:

$$r_s = 1 - \frac{6\sum D^2}{m(m^2 - 1)}$$

where m is the number of data, and $D = x_i - y_i$

- Advantage for feature selection: Invariant to any monotonic transformation of the original features, and hence can detect any monotonic relationship with the class.
- Preprocessing: Transform the feature value to rank.

Runs test

- Purpose: The runs test is used to decide if a data set is from a random process.
- Definition: Run R is length of the number of increasing, or decreasing, values.
- Step.
  1. computer the mean of the sample.
  2. Going through the sample sequence, replace any observation with +, or – depending on whether it is above or below the mean.
  3. Computer R.
  4. Construct hypothesis test on R.
- Advantage: could be used to detect non-monotonic relation.
- Detail see: Gibbon, J. D. Nonparametric Statistical Inference, Chapter 3.

Permutation Test

- Purpose: calculate the p-value for each feature.
- Principle: If there is no real association, the class labels might as well be matched up with the features in a completely different way.
- Formulary:

$$p = 2\min\left(\frac{1}{m!}\sum_{\pi} I(r_{xy_\pi} \geq r_{xy}), \frac{1}{m!}\sum_{\pi} I(r_{xy_\pi} \leq r_{xy})\right)$$

Where the sums are over all m! possible permutations of $y_1, y_2, \ldots y_m$, with y_π denoting the class labels as permuted by π.

Dimensionality reduction with PCA

Benefits:

1. PCA is unsupervised approach, which could use all training, validation and test example.
2. Bayesian model with the spherically symmetric prior is invariable to PCA
3. PCA is feasible even when n is huge, if m is not too large – time required is of order $\min(mn^2, nm^2)$.

Practice:

Power transformation is chosen for each feature to increase correlation with the class. Whether to use these transformations, and the other choices, were made manually based on validation set results.

Some issues proposed in the paper:

1. Should features be transformed before PCA? Eg, take square roots.
2. Should features be centered? Zero may be informative.
3. Should features be scaled to have the same variance? Original scale may carry information about relevance.
4. Should principle components be standardized before use? May be not.

Tow Layers Neural Networks (1/2)

- Multilayer perceptron networks, with two hidden layers with tanh activation function.

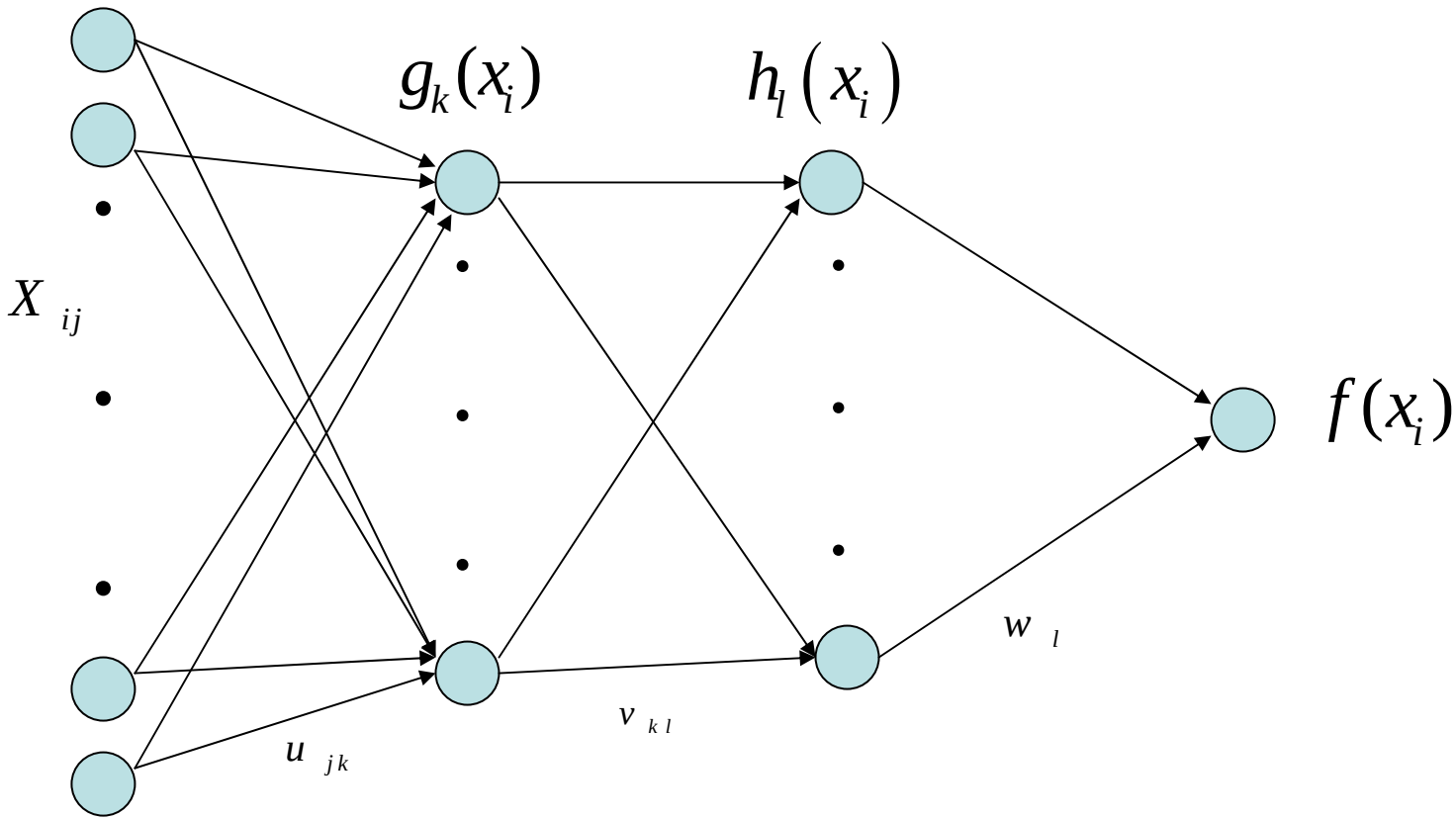
$$P(Y_i = 1 \mid X_i = x_i) = \left[1 + \exp(-f(x_i))\right]^{-1}$$

$$f(x_i) = c + \sum_{l=1}^{H} w_l h_l(x_i)$$

$$h_l(x_i) = \tanh(b_l + \sum_{k=1}^{G} v_{kl} g_k(x_i))$$

$$g_k(x_i) = \tanh(a_k + \sum_{j=1}^{n} u_{jk} x_{ij})$$

Two Layer Neural Networks (2/2)



N input features

G hidden units

H hidden units

Conventional neural network learning

- Learning can be viewed as maximum likelihood estimation for the network parameters.
- Value of weights ϖ is computed by maximizing

$$\prod_{i=1}^{m} P\left(y_i \mid x_i, \omega\right)$$

Where (x_i, y_i) are training case i.

- Make predictions for a test case x using the conditional distribution

$$P\left(y \mid x, \varpi\right)$$

- Overfitting happen when the number of network parameters is large than the number of training cases.

Bayesian Neural Network Learning

- Bayesian predictions are found by integration rather than maximization. For a test case x, y is predicted using

$$P(y \mid x, (x_1, y_1), \ldots (x_m, y_m))$$
$$= \int (d\omega) P(y \mid x, \omega) \times P(\omega \mid (x_1, y_1), \ldots, (x_m, y_m))$$

- The posterior distribution used above is

$$P(\omega \mid (x_1, y_1), \ldots, (x_m, y_m)) \propto P(\omega) \prod_{i=1}^{m} P(y_i \mid x_i, \omega)$$

- We need to define a prior distribution .

ARD Prior

- What for?

  By using a hierarchical prior, we can automatically determine how relevant each input is to predicting the class.

- How?

  Each feature is associated with a hyper-parameter that expresses how relevant that feature is. Conditional on these hyper-parameters, the input weight have a multivariate Gaussian distribution with zero mean and diagonal covariance matrix, with the variance as hyper-parameter, which is itself given a higher-level prior.

- Result.

  If an input feature x is irrelevant, its relevance hyper-parameter β will tend to be small, forcing the relevant weight from that input to be near zero.

Dirichlet Diffusion Trees Procedure(1/2)

A Dirichlet diffusion tree model defines a procedure for generating data sets one point at a time in an exchangeable way.

Procedure

- The first point is generated by a simple Gaussian diffusion process from time 0 to 1
- The second point follows the path of the first one initially.
- The second point diverges from the path at a random time t.
- After the divergence, the second point follows a Gaussian diffusion process independent of the first one.

Dirichlet Diffusion Trees Procedure(2/2)

Procedure (continue)

- The nth point follows the path of those before it initially.
- The nth point diverges at a random time t
- At a branch, the nth point selects an old path with probability proportional to the numbers of points that went each way previously
- After the divergence, it follows a Gaussian diffusion process independently.

Divergence Function

- If a point following a path of n points did not diverge at time t, the probability of divergence during *dt* is *a(t)dt/n*.

- With more points passing the way, the probability of divergence for a new point is smaller.

Selection of divergence function

- Select *a(t)* such that

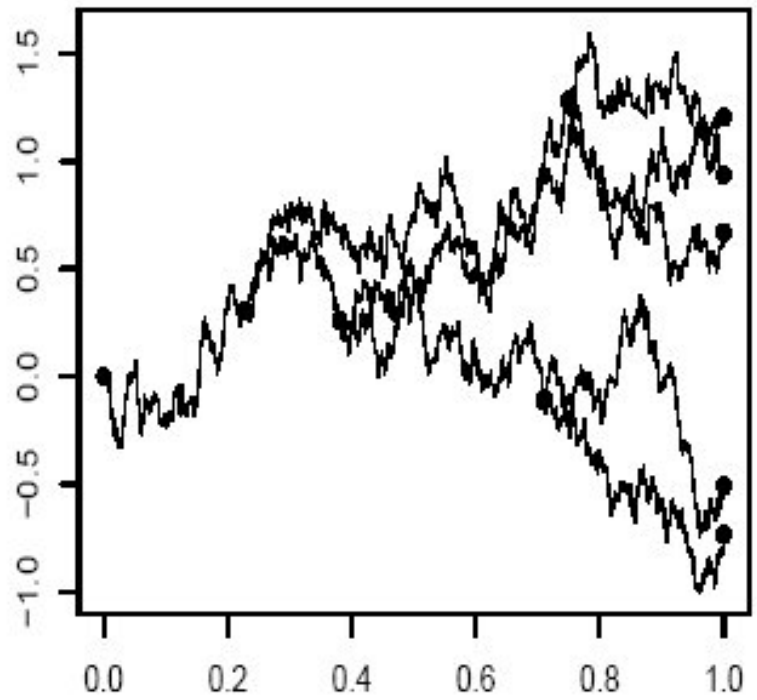
$$\int_0^1 a(t)\,dt = \infty$$

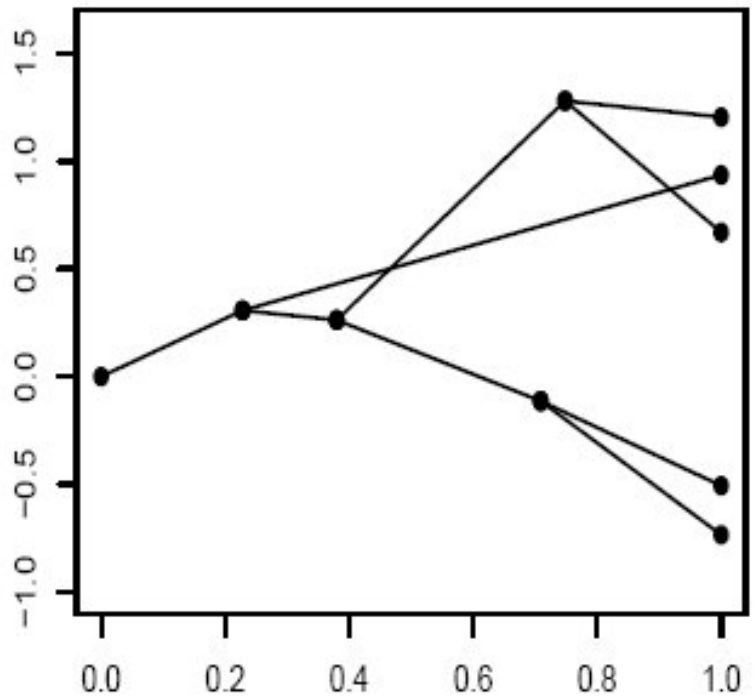
- To keep the distribution continuous.
- Two possibilities

$$a(t) = \frac{c}{1-t}$$

- Or

$$a(t) = b + \frac{c}{(1-t)^2}$$


1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

Dirichlet Diffusion Trees Model

- Structure

  The structure of the tree

  The divergence times for non-terminal nodes

- Hyper-parameter

  Parameters of the divergence function – for example, *c* in *a(t) = c/(1-t).*

  Diffusion standard deviations for each variable.

  Noise standard deviations for each variable.

Bayesian learning of Dirichlet diffusion tree

- Likelihood

  The probability of obtaining a given tree and data set can be written as a product of two factor.

  -The tree factor is the probability of obtaining the given tree structure and divergence times.

  -The data factor is the probability of obtaining the given locations for divergence points and final data points, given the tree structure and divergence times.

- Prior

  Using ARD again.

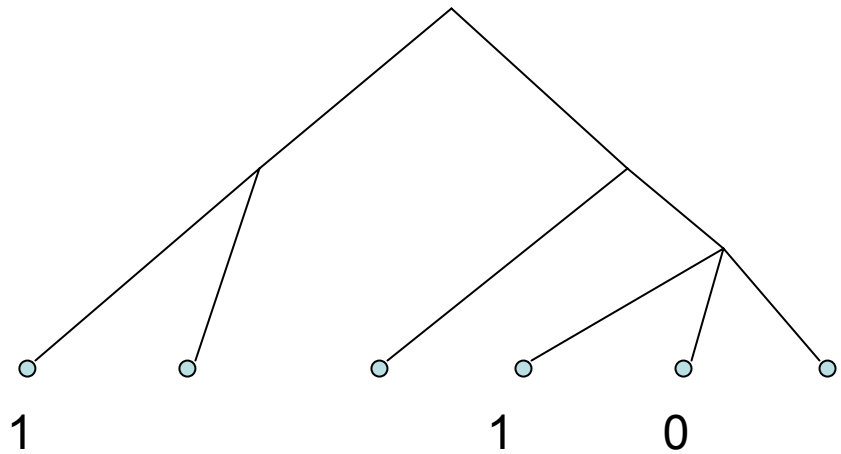
  -By using a hierarchical prior, we can automatically determine how relevant each input is to predicting the class.

Classification from tree (1/2)

- Classify using the structure of the tree

  -Use the nearest neighbor

  -Use the nearest neighbor with branch weights



- Take the average over trees from the posterior

Classification from tree (2/2)

- Classify by using divergence time as a measure of distance

  -Take the average of the divergence times

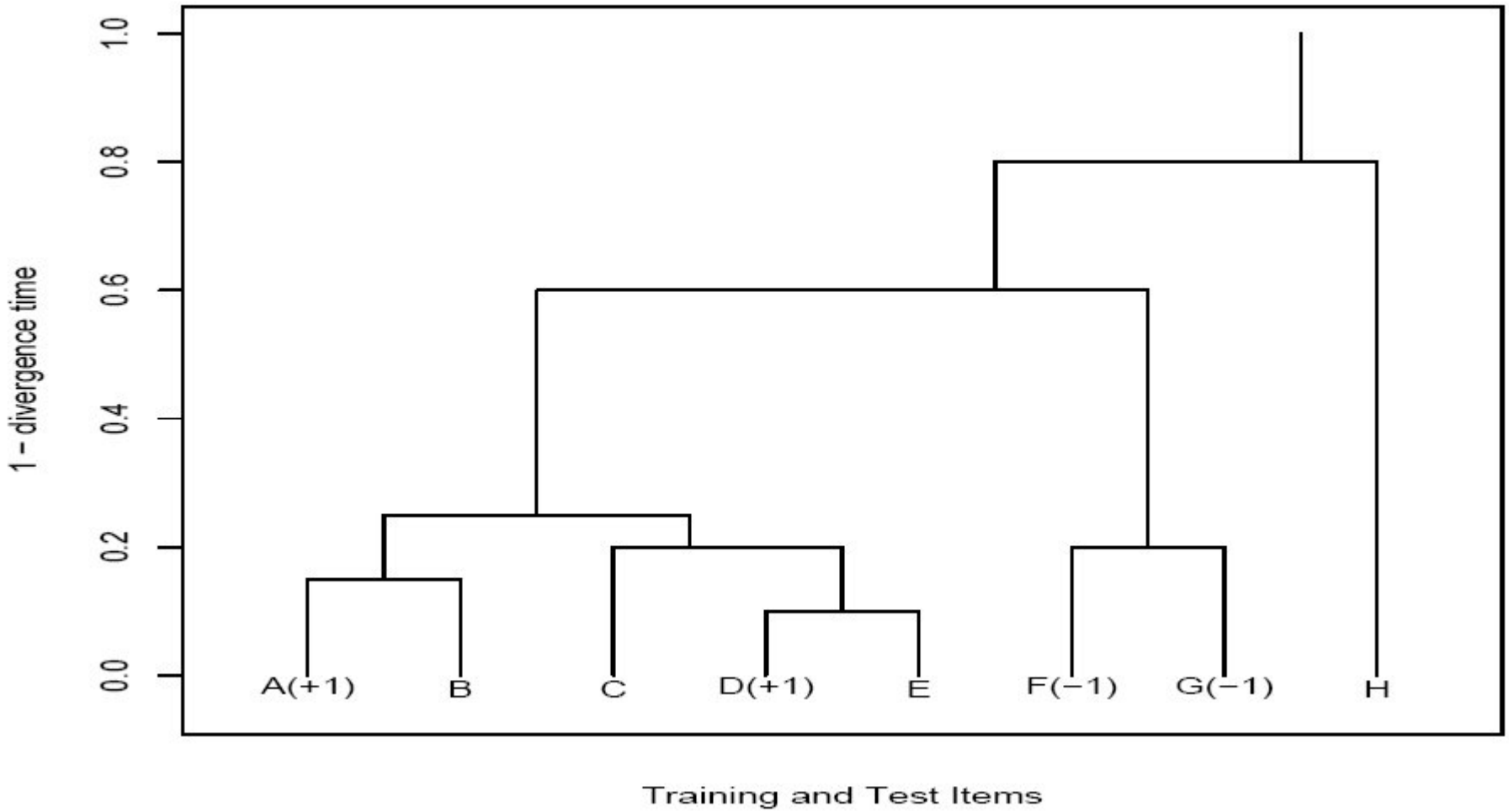
  -Use nearest neighbor method

  -Use (1- divergence time) as a measure of dissimilarity

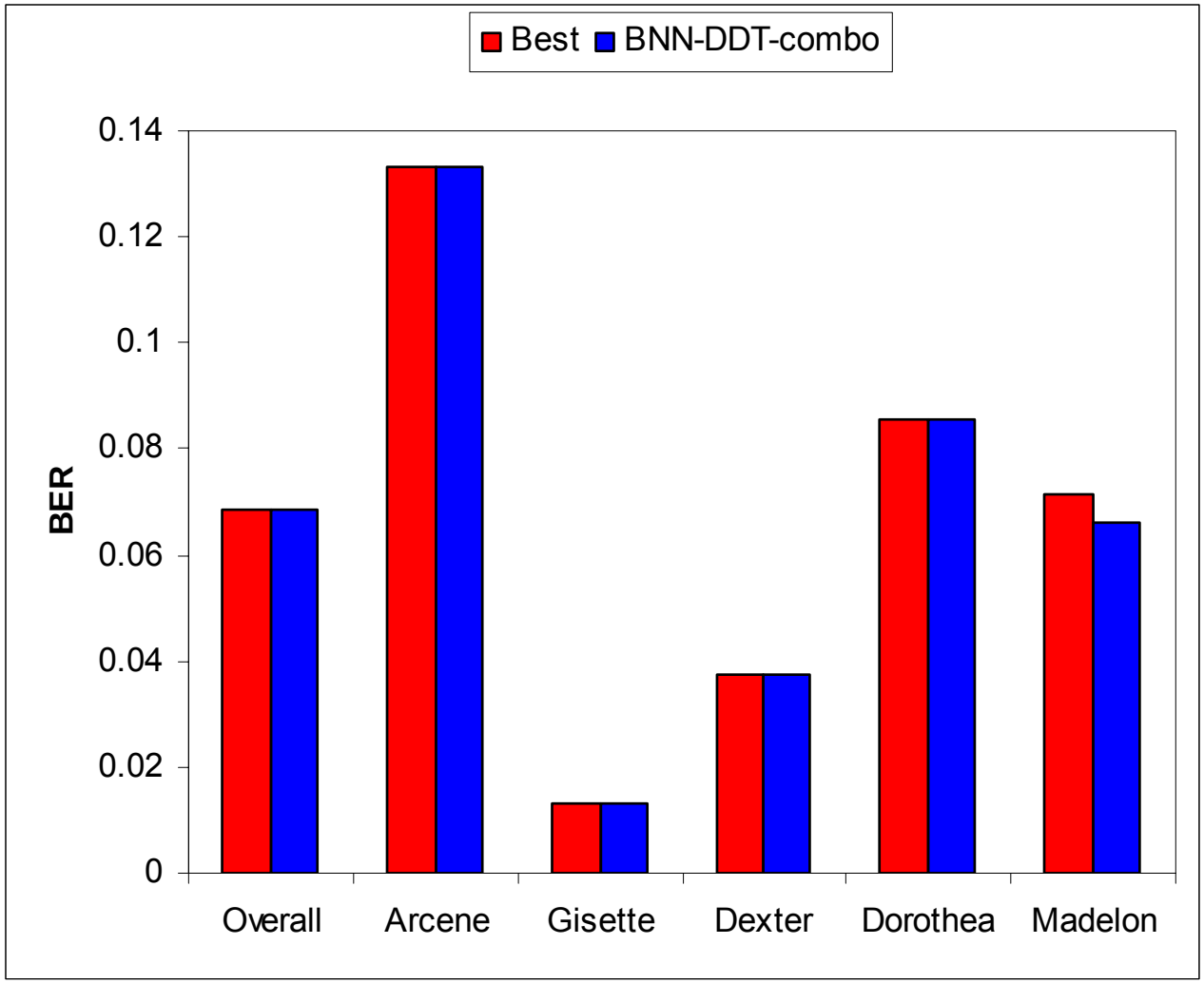
  -Use a weighted average based on divergence times.

$$P(y_i = 1) = \frac{\sum_{y_j = 1,\ j \neq i} (e^{r d_{ij}} - 1)}{\sum_{j \neq i} (e^{r d_{ij}} - 1)}$$

  -Note: r can be selected by cross validation


1 – divergence time
0.0
0.2
0.4
0.6
0.8
1.0
A(+1)
B
C
D(+1)
E
F(−1)
G(−1)
H
Training and Test Items

NIPS2003 Result



Conclusion

- It is unclear whether it is the Bayesian model that contribute most on learning performance.
- It is normally untrue to have a spherically symmetric prior for most real application, I doubt if the Bayesian model is really invariable to PCA.
- I doubt whether we could use the Dirichet diffusion tree model without knowing that if there is hidden tree structure behind the data?