NAG DMC

ユーザガイド

1 はじめに

本資料は NAG DMC の持つ機能を紹介するものであり、そのセクションは共通的なタスクを遂行する上で必要となる機能に基づき構成されています（セクション 2 参照)。ここでは本資料の内容について簡単にまとめておきます。

セクション 3.1 はデータ代入の手法について、またセクション 3.2 は連続値データの場合の外れ値検出手法について記述しています。セクション 4 はデータ値の変換に関する手法について、セクション 5 はクラスタ分析に関する手法について記述しています。分類と回帰モデルについてはそれぞれセクション 6、セクション 7 に記されています。セクション 8 では相関ルールに関する手法について、そして最後にセクション 9 では NAG DMC で提供されているユーティリティ関数について説明します。

2 タスクガイド

本セクションでは共通的な分析タスクを解決する上で NAG DMC のどの関数を選択したら良いかに関する一般的なガイドを記しておきます。特定のタスクに関しいくつかの関数の選択肢が残る場合には、さらなるガイダンスがテーブル形式で示されます。そうでない場合には関連する手法と関数について記述のある本資料中の該当セクションへ読者を誘導します。

共通的な分析タスクは次のように分類されます。

(a) 欠損値を置き換える。セクション 3.1 参照。
(b) 情報をできるだけ保持しつつ変数の数を削減する。セクション 4.2 参照。
(c) 情報をできるだけ保持しつつデータレコード数を削減する。セクション 5 参照。
(d) データの中から変則的なデータレコードを検出する。セクション 3.2 参照。
(e) データの中からグループをさがす。セクション 5 参照。
(f) データレコードを一つ、または複数のグループにアサインする。セクション 6 参照。
(g) データ中における連続的な値を予測する。セクション 7 参照。
(h) データ値間の相関ルールを計算する。セクション 8 参照。

3 データクリーニング

3.1 データ代入

ランダムに値の欠けたデータを扱う一つのやり方は個々の欠損値を適切な値で置き換えてやる方法です。この操作はデータ代入 (data imputation) として知られるもので、それには 3 種類の基本的なアプローチがあります。

第 1 のアプローチは 3 つのうちで最も簡単なもので、連続変数に対する欠損値の場合にはその平均値で、カテゴリ変数に対する欠損値の場合にはその最頻値 (mode) で置換えを行います。

第 2 のアプローチは値の欠けたものに最も類似のデータレコード (donor) を探す方法です。ドナーの候補として複数がある場合には、いくつかの方法（候補中からランダムに選択、または先頭のものを選択）によって一つが選択されます。値を受ける側とドナーとが最良のマッチとなるレコードを探し出すテクニックがポイントとなります。ドナーを選ぶのに有効な方法は変数値に対して距離の基準を設けることです。連続変数の場合にはユークリッド (Euclidean) 距離、マンハッタン (Manhattan) 距離、回帰 (regression) 距離、閾値 (threshold) 距離等の基準があります。カテゴリ変数の場合には単純なブーリアンマッチング (Boolean matching) とスケーリングされたランク差 (scaled rank differences) を用いるのが一般的です。カテゴリ変数の場合には距離を定義したテーブルをユーザが用意する方法もあります。

第 3 のアプローチは欠損値を予測するために数学的モデルを用いる方法です。一般的には反復的な手法により欠損値を含むデータに対しモデルのフィットが行われます。具体的には欠損値に対する推定値からスタートし、データに対しモデルをフィットさせ、それを用いてより良い推定値を導きます。このプロセスを欠損値に対する推定値が収束するまで繰り返します。フィッティングが最尤度に基づき、予測が期待値を用いる場合には、モデルは EM アルゴリズムとなります。

3.1.1 テーブル

アルゴリズム	計算資源への負荷		カテゴリ変数？
	メモリ	CPU	
Simple			Yes
Distance	★	★	Yes
EM			No

表 1　データ代入に関する関数。対象データにカテゴリ変数が含まれる場合には EM 法は使用できません。これらのアルゴリズムに関する詳細についてはセクション 3.1.2 に書かれている関数仕様書を参照ください。

3.1.2 関数

関数 **nagdmc_impute_simp** は欠損値を変数の平均値（連続変数の場合）、または最頻値（カテゴリ変数の場合）で置き換えます。

関数 **nagdmc_impute_dist** は変数値に関する距離基準を用いて欠損値の置換えを行います。

関数 **nagdmc_impute_em** は EM アルゴリズムを用いて欠損値の置換えを行います。

データ代入関数によってアロケートされたメモリは **nagdmc_free_impute** を用いることによって解放することができます。

3.2 外れ値検出

外れ値 (outlier) 検出というのは一群のデータの中から疑わしいデータレコードを見つけ出す操作のことを言います。データレコードが推定される分布に従っていないように見える場合、それらは“疑わしい”(suspect) ものとして特定されます。NAG DMC においては多変数の正規分布が仮定され、データ値は連続な変数値を取るものとします。

最初のステップでは“典型的”とみなせるデータレコードの部分集合を抽出します。この操作は平均値からのマハラノビス距離 (Mahalanobis distance)、または p 変数の中央値 (median) からの距離に基づいて行われます。

次にこの“典型的”な部分集合に対し、マハラノビス距離が与えられた棄却レベル α に対する補正 χ^2 分布（自由度 p）の値よりも小さなデータレコードをすべて追加します。

“典型的”な部分集合のサイズが変化しなくなった段階でそれに含まれないデータレコードが外れ値とみなされます。

マハラノビス距離が最初の“典型的”な部分集合を選択するよう指定された場合には、EM データ代入法（セクション 3.1 参照）を用いて欠損値に対する代入操作が行われます。

3.2.1 関数

関数 **nagdmc_bacon** は“良い”データレコード数と χ^2 分布の棄却レベルを初期値として与えられたときに、前向きの外れ値検出を実行します。BACON という名は Blocked Adaptive Computationally-efficient Outlier Nominator の頭文字を取ったものですが、1620 年に

“*Whoever knows the ways of Nature will more easily notice her deviations; and, on the other hand, whoever knows her deviations will more accurately describe her ways.*”

と書いた Sir Francis Bacon に敬意を表してのものでもあります。

4 データ変換

4.1 データのスケーリング

連続変数のデータ値が距離の計算にどれだけ寄与するかは変数値の範囲に依存します。従ってスケーリング機能を持たない関数の場合には、連続変数のすべてのデータ値が同一のスケールに乗るよう変換する操作が必要になります。連続変数をスケーリングする通常の方法はデータ値から平均値を引き、標準偏差で割る方法です。結果として平均値が 0、分散が 1 のデータが生成されます。

4.2 主成分分析

主成分分析 (principal component analysis, PCA) は分析に必要な変数の数を減らすためのツールです。PCA はオリジナルデータ中において（分散、または 2 乗和で計測される）最も多くの情報を含む直交（すなわち無

相関の）変数を選び出します。主成分と呼ばれる新たな変数はオリジナルデータの線形変換として計算されます。

主成分の数は、データ中のある与えられたパーセンテージの情報を説明する上で必要となる主成分の数、あるいは統計検定の実行に基づいて選択できます。またどのタイプの規格化 (standardisation) を用いるかという問題もあります。すべての変数が類似の大きさと分散を持っている場合には規格化は必要ありません。変数値が大きさの面で異なるときには規格化が適用されるべきです。標準偏差が用いられる場合には相関行列上で PCA を実行することと等価になります。

4.3 関数

4.3.1 スケーリング

関数 **nagdmc_scale** は連続変数のデータ値を平均値が 0、分散が 1 となるようスケーリングします。

4.3.2 主成分分析

関数 **nagdmc_pca** は主成分分析の計算を実行します。規格化が必要な場合の選択肢としては、標準偏差によるものとユーザ提供のスケーリングによる規格化が選択できます。そうでない場合には 2 乗和と分散の選択が可能です。この関数は新たな変数を計算する際に使用される係数と、個々の成分に含まれる変動のパーセンテージに関する情報を返します。ユーザが平均値と（必要な場合には）標準偏差を入力することができるので、それらの再計算をスキップすることも可能です。

関数 **nagdmc_pca_score** は使用する成分の数と nagdmc_pca から返される情報に基づき、データレコードの変換後の値を計算する機能を提供します。

5 クラスタ分析

クラスタ分析とは類似のデータレコードからなるグループを見出すということを狙った統計手法のことを言います。例えば数多くの製品ラインを購入した顧客に関するデータがあった場合、小売業者は類似の購買パターンを持った顧客をグルーピングしたいと望むでしょう。これらのグルーピングができたら、さらなる分析（例えば該当グループを定義付けるキーとなる特徴は他にも存在しないか、等）も行えます。

クラスタ分析は 2 人の個体がどれだけ似ているか、またはどれだけ違っているかを見極めることから始まります。データをコード化し、適当にスケーリングした後に、似通ったデータをグルーピングする操作が行われます。主なアプローチとしては k-means クラスタリングと階層的クラスタリングの 2 種類があります。

5.1 k-means クラスタリング

k-means クラスタリングにおいては、データ中にいくつのグループ（クラスタ）があるかをユーザが決めます。そのグループ数に関し事前にわかっている場合を除けば、いくつかの k の値に対し実験を行い、最適な値を見出すというのが賢明なやり方でしょう。次にこの手法はそれぞれのデータを k 個のグループのいずれかに割り振ります。最初の割振りが終わったら次にグループ間でデータレコードを移動させ、グループ内距離の合計が最小になるまでそれを続けます。

初期の割振りには様々なやり方があります。一つの方法は k 個の個体をランダムに選択し、それらをグループの中心として使用する方法です。次に残りを最も近い中心を持ったグループに割り振って行く方法です。初期のクラスタを見出す他のアプローチとしては、グループ中心を与える事前の情報を用いる方法やキーとなる個体をグループ中心として使用する方法があります。

5.2 階層型クラスタリング

階層型クラスタリング (hierarchical clustering) は通常データレコードの集合からスタートし、それらを 1 ステップずつグルーピング化して行き、最後に 1 グループに到達するものです。このアプローチの利点はグループが結合される様子を見ることによって、データ中にいくつグループが存在し、それらがどの程度明確に規定されたものかに関する情報が得られる点にあります。また開始点を指定する必要もありません。階層型クラスタリングには良く使われる 6 種類の手法があります。単連結法 (single link)、完全連結法 (complete link)、群平均法 (group average)、重心法 (centroid)、メディアン法 (median)、最小分散法 (minimum variance) の 6 つです。違いは個体間の距離からグループ間の距離をどう算出するかという点にあります。この選択はクラスタリング手法の性格を左右することになります。

階層型クラスタリングを遂行する上での標準的なアプローチは最初にすべての個体に対し距離行列 (distance matrix) を計算し、その後この行列を更新して行く方法です。これは個体数が高々数 1000 までのケースに適しています。しかし階層型クラスタリングの 6 つの手法のうち 2 つ、すなわち群平均法と最小分散法については異なるアプローチが存在します。これらの 2 つの手法は単連結法に比べより丸められた (rounded) クラスタを構成します。これら 2 手法の双方とも、グループ間の距離を計算する際、グループメンバをグループの重心 (centroid, mean) で置き換えられるという特質を持っています。従ってある一時点においてはグループ平均のみを記憶しておけば良いことになります。なるべく多くの併合操作を平行して進めなければならないことを考えると、効率の面で優位な特質と言えます。

5.3 テーブル

アルゴリズム	計算資源への負荷		グループ数既知？
	メモリ	CPU	
k-means クラスタリング			Yes
階層型クラスタリング	★	★	No

表 2　クラスタ分析用関数。k-means クラスタリングの場合、データセット中のグループ数について値を指定する必要があります。これらのアルゴリズムに関する詳細についてはセクション 5.4 に書かれている関数仕様書を参照ください。

5.4 関数

5.4.1 k-means クラスタリング

ユーティリティ関数 **nagdmc_rints** はクラスタの中心を構成するために使用されるデータをランダムに選択する機能を提供します。一方、**nagdmc_nrgp** はデータレコードを最も近傍のクラスタに割り振る機能を提供します。

関数 **nagdmc_kmeans** はユークリッド距離に対する k-means クラスタ分析を実行します。

クラスタリングが得られた場合、関数 **nagdmc_wcss** はクラスタ内の 2 乗和を計算する機能を提供します。

5.4.2 階層型クラスタリング

関数 **nagdmc_hclust** は群平均法、または最小分散法に基づく階層型クラスタ分析機能を提供し、グルーピングがどのように行われたかを示す履歴情報を返します。

関数 **nagdmc_cind** は階層型クラスタ分析の結果に基づきデータレコードを指定された数のグループに割り振ります。

6 分類

6.1 決定木

決定木 (decision trees) は再帰的な区分化手法を用いてデータレコードをリーフノード (leaf nodes) に対応付けます。図 1 は簡単なバイナリ決定木の例を示したものです。この例の場合、ルートノード A でのテストの結果、データは 2 つの子ノード、すなわち中間ノード (internal node)B とリーフノード C に区分されます。ノード B のデータはさらにリーフノードの D と E に区分されます。それぞれの親ノードには 2 つの子ノードが付帯することから、このような決定木は二分木 (binary tree) と呼ばれます。一般に親ノードが 1 より大きな任意の数の子ノードを持ち得る場合の決定木は n-分木 (n-ary decision trees) として知られています。

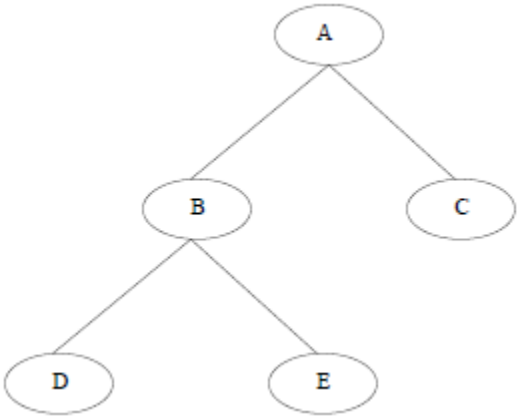

図 1　2 分木の例。楕円はノードを表し、親ノードは線分によって子ノードにリンクされます。

データの区分はノード上でデータのテストを行い、その結果を評価することによって行われますが、その場合、ある基準に基づき最良のテストが保持されます。NAG DMC 中における決定木分類関数の場合、この基準としては不純度に関する Gini インデックス、もしくは最小エントロピーが用いられます。個々のリーフノードを従属変数値に対してラベリングするためには平均値や最頻値といった基本統計量が用いられます。

NAG DMC には分類用の機能として二分木と n-分木を計算する関数が用意されています。

6.2 一般化線形モデル

一般化線形モデル (generalised linear models) は広範なモデルのフィットを可能にします。2 値データに対してはロジスティックとプロビット (probit) 回帰モデルが、分割表 (contingency tables) に対しては対数線形 (log-linear) モデルが用意されています。一般化線形モデルは次の 3 つの要素から構成されます。

(a) 従属変数 Y の分布
(b) 独立変数の線形結合によって定義されるモデル（線形予測子 (linear predictor)）
(c) Y の期待値と線形予測子を結び付ける連結関数 (link function)

NAG DMC においては次の統計分布が利用できます。

(a) 主としてバイナリの分類型タスクに使用される 2 項分布。使用できる連結関数には次のものがあります。
  (i) ロジスティックリンク
  (ii) プロビットリンク
  (iii) complementary log-log
(b) 主として係数値に使用されるポアッソン分布。使用できる連結関数には次のものがあります。
  (i) 指数リンク
  (ii) 恒等 (identity) リンク
  (iii) 対数リンク
  (iv) 平方根リンク
  (v) 逆数リンク

いずれの場合もモデル中のフリーパラメータに対する最尤推定値は反復的重み付き最小二乗法 (iterative weighted least-squares procedure) を用いて求められます。

6.3 多層パーセプトロンニューラルネットワーク

多層パーセプトロン (multi-layer perceptrons (MLP)) モデルの概要についてはセクション 7.3 を参照ください。

分類型タスクが 2 つのクラスを持つ場合には、最適化の性格上、一つのクラスを 0、他方を 1 とコード化してください。クラスが $c(>2)$ クラスある場合には、c 個の 2 値従属変数（クラスメンバシップを表す 1 とそれ以外の 0 から構成されるもの）を設定してください。

6.4 最近傍モデル

データレコード x が与えられ、その従属変数に対応した値が未知の場合、k-最近傍モデル (k-nearest neighbour model) はトレーニングデータ中から x に類似の k 個のデータレコードを抽出します。x とトレーニングレコード間の類似性は独立変数上で計算される距離関数によって測られます。その場合、従属変数の値は k 個の最近傍データの中で最も高い確率を持ったクラス値に設定されます。個々のクラス値に対し事前確率を設定することもできます。

データレコードを比較する際使用される類似性に関する尺度としては 2 種類の距離関数が選択できます。第 1 はユークリッド距離の 2 乗で、データ値間の差の 2 乗和を評価することによって距離を計測します。第 2 は l_1 ノルム、すなわちマンハッタン距離で、データ値間の差の絶対値を合計したものを評価し距離の尺度とします。

6.5 テーブル

アルゴリズム	計算資源への負荷		クラス数 > 2?	不均等のクラス比率?
	メモリ	CPU		
決定木			Yes	Yes
一般化線形モデル			No	Yes
多層パーセプトロン		★	Yes	No
最近傍モデル	★		Yes	Yes

表 3　データ分類用関数。MLP モデルの場合、個々のカテゴリに区分されるデータレコードの数は概ね等しいものである必要があります。これらのアルゴリズムに関する詳細についてはセクション 6.6 に書かれている関数仕様書を参照ください。

6.6 関数

6.6.1 決定木

関数 **nagdmc_gini_tree** は不純度に関する Gini インデックスに基づく基準により二分木を作成します。計算された二分木は関数 **nagdmc_save_gini_tree** によってセーブでき、逆に **nagdmc_load_gini_tree** によってバイナリファイルからロードできます。Gini インデックスの決定木を含むメモリは関数 **nagdmc_free_gini_tree** によって解放できます。

関数 **nagdmc_predict_gini_tree** は nagdmc_gini_tree によって作成された決定木が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

関数 **nagdmc_entropy_tree** は最小エントロピー基準に基づき n-分木を作成します。新しいデータに対するエントロピー決定木の精度は、フィットされたモデルに対する悲観的な枝狩り (pessimistic error pruning) によって改善できます。そのためには関数 **nagdmc_prune_entropy_tree** を使用します。

フィットされたエントロピー決定木は関数 **nagdmc_save_entropy_tree** によってセーブでき、逆に **nagdmc_load_entropy_tree** によってバイナリファイルからロードできます。

エントロピー決定木を含むメモリは関数 **nagdmc_free_entropy_tree** によって解放できます。

関数 **nagdmc_predict_entropy_tree** は nagdmc_entropy_tree によって作成された決定木が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

6.6.2 一般化線形モデル

関数 **nagdmc_binomial_reg** は 2 項分布に従う誤差を持った一般化線形モデルを計算します。

関数 **nagdmc_logit_reg** はロジスティック回帰モデル（すなわち 2 項分布に従う誤差とロジスティック連結関数による一般化線形モデル）の計算に際し、より簡便なインタフェースを提供します。

関数 **nagdmc_probit_reg** は 2 項分布に従う誤差とプロビット連結関数による一般化線形モデルの計算に際し、より簡便なインタフェースを提供します。

関数 **nagdmc_poisson_reg** はポアッソン分布に従う誤差を持った一般化線形モデルを計算します。

関数 **nagdmc_loglinear_reg** はポアッソン分布に従う誤差と対数-線形連結関数による一般化線形モデルの計算に際し、より簡便なインタフェースを提供します。

フィットされた回帰モデルに関する情報は関数 **nagdmc_extr_reg** を用いることによって得ることができます。また **nagdmc_predict_reg** は独立変数上の新たなデータに対する予測値を計算します。

フィットされた一般化線形モデルは関数 **nagdmc_save_reg** によってディスク上にセーブでき、逆に **nagdmc_load_reg** によってメモリ上にロードできます。一般化線形モデルにアロケートされたメモリは関数 **nagdmc_free_reg** によって解放できます。

6.6.3 多層パーセプトロンニューラルネットワーク

関数 **nagdmc_mlp** は MLP モデルの計算に使用できます。フリーパラメータに対し乱数値を用いることによって数多くの初期条件を試すようこの関数に指示することができます。この場合、これら初期の最適化に関する最良値はメインの最適化プロセスに対する初期値として使用されます。ユーザは初期の最適化の数と最適化プロセスにおける反復回数とを制御できます。

関数 **nagdmc_predict_mlp** は **nagdmc_mlp** から返される MLP モデル情報が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

6.6.4 最近傍モデル

NAG DMC の場合、最近傍の近似は 2 段階のステージを経て求められます。まず最初に **nagdmc_kdtree** を使って k-d tree が計算されます。この tree は関数 **nagdmc_save_kdtree** によってバイナリファイルにセーブでき、逆に **nagdmc_load_kdtree** によってファイルからロードできます。k-d tree を含むメモリは関数 **nagdmc_free_kdtree** によって解放できます。

次に **nagdmc_knnc** によるクラスメンバシップに関する事前確率を用い、最近傍の近似計算が行われます。

7 回帰

7.1 決定木

回帰のタスクを支援するため NAG DMC が用意している 2 種類の決定木は共に二分木 (binary trees) です (セクション 6.1 参照)。ノードに位置するデータの平均値に関し共に最小 2 乗和を判定基準として使用します。しかし一方はノードにおける従属変数の平均値の推定に際しロバスト推定 (robust estimate) を行いますが、他方は単純平均を使用します。

7.2 線形回帰

線形回帰モデルは多くの独立変数から y の帰結を予測するのに使用されます。その際用いられる予測モデルは独立変数と定数項を線形に結合したものです。モデル中の係数（重み）は y の値がわかっているトレーニングデータを用いて推定され、最小 2 乗近似によりフィットされます。係数の符号と大きさはフィットされたモデルの解釈に使用することができます。係数値とその標準誤差との比はその統計的な有意性に関する尺度となります。ある変数に対するこの比の値が 2 より大きなときに、一般的にその変数はモデル中に保持しておいて意味があると考えられます。

独立変数がカテゴリデータの場合にはデータ値の前処理が必要となることがあります。カテゴリ変数が 2 つより多くのレベルを持つ場合（例えば赤、青、緑の 3 種類の値を取る場合）にはダミー変数の導入が必要になります。2 値のカテゴリ変数の場合には 0 と 1 を用いてコード化されます。

7.3 多層パーセプトロンニューラルネットワーク

多層パーセプトロン (multi-layer perceptrons (MLPs)) は図 2 に示されるような有向グラフ (directed graph) として表現される柔軟性の高い非線形モデルです。

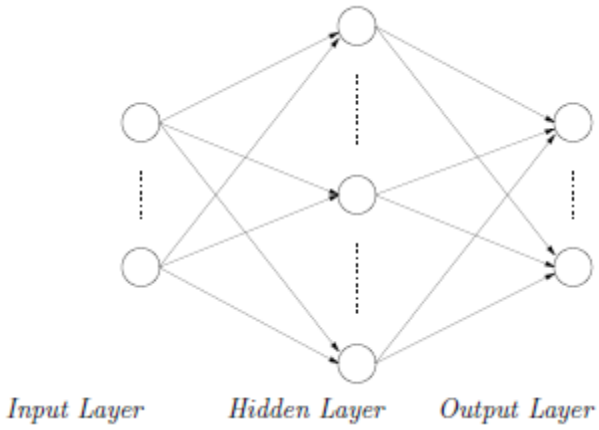

図 2　一つの隠れ層を持つフィードフォワード MLP。MLP では計算ノードは円で表されます。円同士をつなぐ線分は重みを表します。入力層は独立変数の数分のノードから構成されます。一方、出力層には従属変数の数だけのノードがあります。隠れ層中のノード数はユーザが指定しますが、これによって MLP モデル中のフリーパラメータ数が規定されます。

入力層のノードは独立変数の値を取り、これに重みをかけます。隠れ層や出力層のノードは 1 つ前の階層内のノードで接続されているものからの寄与を合計し、それに対し伝達関数 (transfer function) を適用します。伝達関数としては S 字状 (sigmoidal (logistic)) 関数、双曲正接 (hyperbolic tangent) 関数、1 次関数といったものが用いられます。

MLP におけるフリーパラメータを最適化するプロセスはトレーニングと呼ばれます。トレーニングにおいては MLP の予測値とトレーニングデータの値との差（誤差）の 2 乗和を最小化する処理が行われます。通常誤差関数の曲面は非常に複雑なものとなります。従って MLP のトレーニングには多くの計算時間が必要となるため、一般的に言うと MLP は大きなデータセットには不向きです。

連続変数のデータ値は事前にスケーリングしておくことを推奨します。これによって MLP のトレーニングが容易になり、より精度の高い結果が得られるようになります。

多層パーセプトロンにおいてパラメータ値を最適化する際、グローバルな最小値がたとえ見つかったにせよ、over-fitting の問題を伴うために望ましい結果が得られる確率は余り高くありません。この over-fitting を回避するために、通常最適化の処理は検証データの使用によって途中で打ち切られます。このためフリーパラメータに対する初期値の設定が近似解の精度を左右する重要な要素となります。いくつか異なる初期値を用いて最適化を何回か行ってみるのが良いでしょう。

7.4 最近傍モデル

データレコード x が与えられ、その従属変数に対応した値が未知の場合、k-最近傍モデル (k-nearest neighbour model) はトレーニングデータ中から x に類似の k 個のデータレコードを抽出します。x とトレーニングレコード間の類似性は独立変数上で計算される距離関数によって測られます。その場合、従属変数の値は k 個の最近傍データの従属変数の値を平均した値に設定されます。

データレコードを比較する際使用される類似性に関する尺度としては 2 種類の距離関数が選択できます。第 1 はユークリッド距離の 2 乗で、データ値間の差の 2 乗和を評価することによって距離を計測します。第 2 は l_1 ノルム、すなわちマンハッタン距離で、データ値間の差の絶対値を合計したものを評価し距離の尺度とします。

7.5 放射基底関数モデル

放射基底関数 (radial basis function (RBF)) は、その中心点から独立変数の値までの距離（ユークリッドノルムを用いて計測）に基づくあるスカラー関数を計算します。RBF として良く使用される関数には 1 次関数、3 次関数、thin plate spline 関数、ガウス関数、multiquadric 関数、逆 multiquadric 関数、コーシー関数等があります。RBF の中心点はドメインに関する知識に基づき、あるいはクラスタ分析（セクション 5 参照）によって選択されます。

RBF モデルは RBF 出力を線形に結合したものです。その場合の重みはトレーニングデータレコードに対しモデルを最小 2 乗フィットさせることで与えられます。

RBF モデルの場合、距離の計算を伴うため、2 レベルよりも多いカテゴリ変数はダミー変数によって置き換える必要があります。2 レベルのカテゴリ変数の場合には 0 と 1 でラベリングしてください。

7.6 テーブル

アルゴリズム	計算資源への負荷		非線形タスク？
	メモリ	CPU	
回帰決定木			Yes
線形回帰			No
多層パーセプトロン		★	Yes
最近傍モデル	★		Yes
放射基底関数モデル			Yes

表 4　回帰分析用関数。線形回帰モデルの場合、非線形のタスク（直線によって近似できないタスク）に対し正確でない予測を行う場合があります。これらのアルゴリズムに関する詳細についてはセクション 7.7 に書かれている関数仕様書を参照ください。

7.7 関数

7.7.1 決定木

関数 **nagdmc_reg_tree** は二分木を作成しますが、その際、それぞれのノードにおいて平均値に関する 2 乗和が最小となるようにデータを区分します。計算された決定木は関数 **nagdmc_save_reg_tree** によってバイナリファイルにセーブでき、その後 **nagdmc_load_reg_tree** によってメモリにロードできます。nagdmc_reg_tree によって作成された決定木を含むメモリは関数 **nagdmc_free_reg_tree** によって解放できます。

関数 **nagdmc_predict_reg_tree** は nagdmc_reg_tree によって作成された決定木が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

関数 **nagdmc_waid** は二分木を作成しますが、その際、それぞれのノードにおいて平均値のロバスト推定値に関する 2 乗和が最小となるようにデータを区分します。計算された決定木は関数 **nagdmc_save_waid** によってバイナリファイルにセーブでき、その後 **nagdmc_load_waid** によってメモリにロードできます。nagdmc_waid によって作成された決定木を含むメモリは関数 **nagdmc_free_waid** によって解放できます。

関数 **nagdmc_predict_waid** は nagdmc_waid によって作成された決定木が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

7.7.2 線形回帰

関数 **nagdmc_linear_reg** は係数を算出し、モデルの適合度に関する情報を返します。例えば R^2 という値はフィットされたモデルによって説明される従属変数の変動の量を表します。

関数 **nagdmc_stepwise_reg** はモデル選択のための自動化ツールとして使用できます。それは線形回帰モデルに含める候補として与えられた集合の中から独立変数の組合せを選び出します。

関数 **nagdmc_extr_reg** を使うとフィットされた線形回帰モデルから結果を抽出することができます。また **nagdmc_predict_reg** は独立変数上の新たなデータに対する予測値を計算します。

フィットされた回帰モデルは関数 **nagdmc_save_reg** によってディスクにセーブでき、逆に **nagdmc_load_reg** によってメモリにロードできます。回帰モデルにアロケートされたメモリは関数 **nagdmc_free_reg** によって解放できます。

7.7.3 多層パーセプトロンニューラルネットワーク

関数 **nagdmc_mlp** は MLP モデルの計算に使用されます。フリーパラメータに対し乱数値を用いることによって数多くの初期条件を試すようこの関数に指示することができます。この場合、これら初期の最適化に関する最良値はメインの最適化プロセスに対する初期値として使用されます。ユーザは初期の最適化の数と最適化プロセスにおける反復回数とを制御できます。

関数 **nagdmc_predict_mlp** は **nagdmc_mlp** から返される MLP モデル情報が与えられたとき、新たなデータレコードに対する予測を計算する機能を提供します。

7.7.4 最近傍モデル

NAG DMC の場合、最近傍の近似は 2 段階のステージを経て求められます。まず最初に **nagdmc_kdtree** を使って k-d tree が計算されます。この tree は関数 **nagdmc_save_kdtree** によってバイナリファイルにセーブでき、逆に **nagdmc_load_kdtree** によってファイルからロードできます。k-d tree を含むメモリは関数 **nagdmc_free_kdtree** によって解放できます。

次に **nagdmc_knnp** によって最近傍の近似計算が行われます。

7.7.5 放射基底関数モデル

関数 **nagdmc_rbf** は選択されたタイプの RBF をトレーニングデータレコードに適用し最小 2 乗フィットを計算します。

関数 **nagdmc_predict_rbf** はフィットモデルが与えられたとき、新たなデータレコードに対する RBF モデルに基づく予測値を計算します。

8 相関ルール

相関分析 (association analysis) の狙いは名義データ (nominal data) の値の間の関係を決定することです。この関係はルールの形式を取り、次のように表現されます。

条件部 (Antecedent) $\longrightarrow$ 帰結部 (Consequent)

例えば次のようなルールがあったとします。

$\{1, 41, 6\} \longrightarrow 19$

これはマーケットバスケット分析の場合であれば次のように解釈されます。

“アイテム $1, 41, 6$ を購入した顧客はアイテム 19 を買う可能性が高い。”

一般に大きなデータセットの場合にはルールの数も非常に大きなものとなります。例えばスーパマーケットの品目が数千に及ぶ場合、ルールの数は億のオーダとなることもあります。しかしルールの探索範囲を確率の高いものに絞ることによって、生成されるルールの数をおさえることができます。

NAG DMC の相関ルールジェネレータに対するデータは個々のトランザクションごとに上昇順にソートされていなくてはなりません。

8.1 関数

トランザクションデータをソートするためにはユーティリティ関数が利用できます（セクション 9 参照）。

ソート後のトランザクションデータは **nagdmc_assoc_data** を使用して配列中に読み込むことができます。関数 **nagdmc_assoc** はソート済みのデータを用いて相関ルールを作成します。作成された相関ルールは **nagdmc_assoc_print** を用いてプリントできます。

9 ユーティリティ関数

ユーティリティ関数は上記の主要な関数をサポートするものとして用意されています。またプロトタイピングを支援する機能もあります。

(a) 実数や整数の乱数を発生させる機能（置換えの有無は選択可）。これらの関数はデータレコードのランダムなサブセットを計算するときなどに使用されます。
(b) 実数、整数からなる配列をランク付けしソートする機能。
(c) データレコードから平均値や分散等の基本統計量を計算する機能。
(d) 分類に続いて結果をクロス集計 (cross-tabulate) する機能。

9.1 関数

nagdmc_rng は基本的な乱数ジェネレータで、**nagdmc_srs** によって設定された初期値、またはシステムクロックを使って、0 から 1 までの範囲の乱数を発生させ、それを返します。

関数 **nagdmc_rints** は基本的な乱数ジェネレータを使って指定された範囲内の整数のサンプルをランダムに発生させます。これはデータセット中からデータレコードをランダムに抽出する用途で使用できます。サンプルには同じ数の繰返しを含めないという設定も行えます。

nagdmc_rank_real は実数データを、**nagdmc_rank_long** は整数データをランク付けします。関数 **nagdmc_index** はランクの順序をインデックスの順序に変換します。

ランク順、またはインデックス順が与えられたとき、**nagdmc_order_real** は実数値のデータを、**nagdmc_order_long** は整数値のデータを並べ替えます。

nagdmc_dsu はデータの平均値、及びその 2 乗和を更新します。これはデータを一塊ごとに処理してサンプルの平均値と分散を計算する際に使用されます。

関数 **nagdmc_tab2** は単純な二元 (two-way) の集計機能を提供するもので、分類結果を比較する際に使用されます。