root

Go Up

Table of Contents

LinearRegression

Go Up

Name	LinearRegression
Version	3.0.0
Description	Linear Regression Algorithm Bundle
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core, PBblas
Platform	6.2.0

Table of Contents

LinearRegression/

OLS

Go Up

IMPORTS

ML_Core | ML_Core.Types | PBblas | PBblas.Types | PBblas.Converted | PBblas.MatUtils | ML_Core.Math |

DESCRIPTIONS

MODULE OLS

	OLS
(DATASET(NumericField) X=empty_data, DATASET(NumericField) Y=empty_data)	

Ordinary Least Squares (OLS) Linear Regression aka Ordinary Linear Regression Regression learns a function that maps a set of input data (independents) to one or more output variables (dependents). The resulting learned function is known as the model. That model can then be used repetitively to predict (i.e. estimate) the output value(s) based on new input data. Two major use cases are supported: 1) Learn and return a model 2) Use an existing (e.g. persisted) model to predict new values for Y Of course, both can be done in a single run. Alternatively, the model can be persisted and used indefinitely for prediction of Y values, as long as the record format has not changed, and the original training data remains representative of the population. OLS supports any number of independent variables (Multiple Regression) and multiple dependent variables (Multivariate Regression). In this way, multiple variables' values can be predicted from the same input (i.e. independent) data. Training data is presented as parameters to this module. When using a previously persisted model (use case 2 above), these parameters should be omitted. This module provides a rich set of analytics to assess the usefulness of the resulting linear regression model, and to determine the best subset of independent variables to include in the model. These include: For the whole model: - Analysis of Variance (ANOVA) - R-squared - Adjusted R-squared - F-Test - Akaike Information Criterion (AIC) For each coefficient: - Standard Error (SE) -

PARAMETER **X** The independent variable training data in DATASET(NumericField) format. Each observation (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number'). Omit this parameter when predicting from a persisted model.

PARAMETER **Y** The dependent variable training data in DATASET(NumericField) format. Each observation (e.g. record) is identified by 'id', and each feature is identified by field number. Omit this parameter when predicting from a persisted model.

Children

1. GetModel : GetModel Returns the learned model that maps X's to Y's
2. Betas : Return raw Beta values as numeric fields Extracts Beta values from the model
3. Predict : Predict the dependent variable values (Y) for any set of independent variables (X)
4. makeRSQ
5. RSquared : RSquared Calculate the R-Squared Metric used to assess the fit of the regression line to the training data
6. AnovaRec
7. calcAnova
8. Anova : ANOVA (Analysis of Variance) report Analyzes the sources of variance
9. SE : Standard Error of the Regression Coefficients Describes the variability of the regression error for each coefficient
10. TStat : T-Statistic The T-statistic identifies the significance of the value of each regression coefficient
11. AdjRSquared : Adjusted R2 Calculate Adjusted R Squared which is a scaled version of R Squared that does not arbitrarily increase with the number of features
12. AICRec
13. AIC : Akaike Information Criterion (AIC) Information theory based criterion for assessing Goodness of Fit (GOF)
14. RangeVec
15. DistributionBase
16. TDistribution
17. FDistribution
18. NormalDistribution

19. pVal : P-Value Calculate the P-value for each coefficient, which is the probability that the coefficient is insignificant (i.e

20. ConfintRec

21. ConfInt : Confidence Interval The Confidence Interval determines the upper and lower bounds of each estimated coefficient given a confidence level (level) that is required

22. FTestRec

23. FTest : F-Test Calculate the P-value for the full regression, which is the probability that all of the coefficients are insignificant (i.e

---

ATTRIBUTE GetModel

OLS \

DATASET(Layout_Model)	**GetModel**

GetModel Returns the learned model that maps X's to Y's. In the case of OLS, the model represents a set of Betas which are the coefficients of the linear model: Beta0 * 1 + Beta1 * Field1 + Beta2 * Field2 ... The ID of each model record specifies to which Y variable the coefficient applies. The Field Number ('number') indicates to which field of X the beta is to be applied. Field number 1 provides the intercept portion of the linear model and is always multiplied by 1. Note that if multiple work-items are provided within X and Y, there will be multiple models returned. The models can be separated by their work item id (i.e. 'wi'). A single model can be extracted from a myriad model by using e.g., model(wi=myWI_id). GetModel should not be called when predicting using a previously persisted model (i.e. when training data was not passed to the module.

RETURN Model in DATASET(Layout_Model) format

SEE ML_core/Types.Layout_Model

OVERRIDE True

FUNCTION Betas

OLS \

DATASET(NumericField)	**Betas**
(DATASET(Layout_Model) model=GetModel)	

Return raw Beta values as numeric fields Extracts Beta values from the model. Can be used during training and prediction phases. For use during training phase, the 'model' parameter can be omitted. GetModel will be called to retrieve the model based on the training data. For use during prediction phase, a previously persisted model should be provided. The 'number' field of the returned NumericField records specifies to which Y the coefficient applies. The 'id' field of the returned record indicates the position of the Beta value. ID = 1 provides the Beta for the constant term (i.e. the Y intercept) while subsequent values reflect the Beta for each correspondingly numbered X feature. Feature 1 corresponds to Beta with 'id' = 2 and so on. If 'model' contains multiple work-items, Separate sets of Betas will be returned for each of the 'myriad' models (distinguished by 'wi').

PARAMETER **model** Optional parameter provides a model that was previously retrieved using GetModel. If omitted, GetModel will be used as the model.

RETURN DATASET(NumericField) containing the Beta values.

FUNCTION Predict

OLS \

DATASET(NumericField)	**Predict**
(DATASET(NumericField) newX, DATASET(Layout_Model) model=GetModel)	

Predict the dependent variable values (Y) for any set of independent variables (X). Returns a predicted Y values for each observation (i.e. record) of X. This supports the 'myriad' style interface in that multiple independent work items may be present in 'newX', and multiple independent models may be provided in 'model'. The resulting predicted values will also be separable by work item (i.e. wi).

PARAMETER **newX** The set of observations of independent variables in DATASET(NumericField) format.

PARAMETER **model** Optional. A model that was previously returned from GetModel (above). Note that a model from a previous run will only be valid if the field numbers in X are the same as when the model was learned. If this parameter is omitted, the current model will be used.

RETURN An estimation of the corresponding Y value for each observation of newX. Returned in DATASET(NumericField) format with field number (i.e. 'number') indicating the dependent variable that is predicted.

OVERRIDE True

TRANSFORM makeRSQ

OLS \

R2Rec	**makeRSQ**
(CoCoRec coco)	

ATTRIBUTE RSquared

OLS \

DATASET(R2Rec)	**RSquared**

RSquared Calculate the R-Squared Metric used to assess the fit of the regression line to the training data. Since the regression has chosen the best (i.e. least squared error) line matching the data, this can be thought of as a measurement of the linearity of the training data. R Squared generally varies between 0 and 1, with 1 indicating an exact linear fit, and 0 indicating that a linear fit will have no predictive power. Negative values are possible under certain conditions, and indicate that the mean(Y) will be more predictive than any linear fit. Moderate values of R squared (e.g. .5) may indicate that the relationship of X -> Y is non-linear, or that the measurement error is high relative to the linear correlation (e.g. many outliers). In the former case, increasing the dimensionality of X, such as by using polynomial variants of the features, may yield a better fit. R squared always increases when additional independent variables are added, so it should not be used to determine the optimal set of X variables to include. For that purpose, use Adjusted R Squared (below) which penalizes larger numbers of variables. Note that the result of this call is only meaningful during training phase (use case 1 above) as it is an analysis based on the training data which is not provided during a prediction-only phase.

RETURN DATASET(R2Rec) with one record per dependent variable, per work-item. The number field indicates the dependent variable and coresponds to the number field of the dependent (Y) variable to which it applies.

RECORD AnovaRec

OLS \

	AnovaRec

TRANSFORM calcAnova

OLS \

AnovaRec	calcAnova
(tmpRec le)	

ATTRIBUTE Anova

OLS \

	Anova

ANOVA (Analysis of Variance) report Analyzes the sources of variance. Basic ANOVA equality: Model + Error = Total Determines how much of the variance of Y is explained by the regression model, versus how much is due to the error term (i.e. unexplained variance). This attribute is only meaningful during the training phase. Provides one record per work-item. Each record provides the following statistics: - Total_SS – Total Sum of Squares (SS) variance of the dependent data - Model_SS – The SS variance represented within the model - Error_SS – The SS variance not reflected by the model (i.e. Total_SS - Error_SS) - Total_DF – The total degrees of freedom within the dependent data - Model_DF – Degrees of freedom of the model - Error_DF – Degrees of freedom of the error component - Total_MS – The Mean Square (MS) variance of the dependent data - Model_MS – The Mean Square (MS) variance represented within the model - Error_MS – The MS variance not reflected by the model - Model_F – The F-Test statistic: Model_MS / Error_MS

RETURN DATASET(AnovaRec), one per work-item per dependent (Y) variable The number field indicates the dependent variable to which the analysis applies.

ATTRIBUTE SE

OLS \

DATASET(NumericField)	**SE**

Standard Error of the Regression Coefficients Describes the variability of the regression error for each coefficient. Only meaningful during the training phase.

RETURN DATASET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The 'number' field indicates the dependent variable to which the coefficient applies.

ATTRIBUTE TStat

OLS \

DATASET(NumericField)	**TStat**

T-Statistic The T-statistic identifies the significance of the value of each regression coefficient. Its calculation is simply the value of the coefficient divided by the Standard Error of the coefficient. A larger absolute value of the T-statistic indicates that the coefficient is more significant. Only meaningful during the training phase.

RETURN DATSET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable to which the coefficient applies.

ATTRIBUTE AdjRSquared

OLS \

DATASET(R2Rec)	**AdjRSquared**

Adjusted R2 Calculate Adjusted R Squared which is a scaled version of R Squared that does not arbitrarily increase with the number of features. Adjusted R2, rather than R2 should always be used when trying to determine the best set of features to include in a model. When adding features, R2 will always increase, whether or not it improves the predictive power of the model. Adjusted R2, however, will only increase with the predictive power of the model.

RETURN DATASET(R2Rec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD AICRec

OLS \

	AICRec

ATTRIBUTE AIC

OLS \

DATASET(AICRec)	AIC

Akaike Information Criterion (AIC) Information theory based criterion for assessing Goodness of Fit (GOF). Lower values mean better fit.

RETURN DATASET(AICRec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD RangeVec

OLS \

	RangeVec

MODULE DistributionBase

OLS \

	DistributionBase
`(t_Count Nranges = 10000)`	

Children

1. Low
2. High
3. Density
4. RangeWidth
5. DensityV
6. CumulativeV
7. Cumulative
8. NTile
9. InvDensity
10. Discrete

ATTRIBUTE Low

OLS \ DistributionBase \

	Low

ATTRIBUTE High

OLS \ DistributionBase \

	High

FUNCTION Density

OLS \ DistributionBase \

`t_FieldReal`	**Density**
`(t_FieldReal t)`	

ATTRIBUTE RangeWidth

OLS \ DistributionBase \

	RangeWidth

FUNCTION DensityV

OLS \ DistributionBase \

`DATASET(RangeVec)`	**DensityV**
`()`	

FUNCTION CumulativeV

OLS \ DistributionBase \

	CumulativeV
()	

FUNCTION Cumulative

OLS \ DistributionBase \

t_FieldReal	Cumulative
(t_FieldReal t)	

FUNCTION NTile

OLS \ DistributionBase \

t_FieldReal	NTile
(t_FieldReal Pc)	

FUNCTION InvDensity

OLS \ DistributionBase \

	InvDensity
(t_FieldReal delta)	

ATTRIBUTE Discrete

OLS \ DistributionBase \

	Discrete

MODULE TDistribution

OLS \

	TDistribution
	`(t_Discrete v_in,t_Count NRanges = 10000)`

Children

1. DensityV
2. NTile
3. Discrete
4. InvDensity
5. High
6. Low
7. RangeWidth
8. Density
9. CumulativeV
10. Cumulative

FUNCTION DensityV

OLS \ TDistribution \

DATASET(RangeVec)	DensityV
()	

OVERRIDE True

FUNCTION NTile

OLS \ TDistribution \

t_FieldReal	NTile
(t_FieldReal Pc)	

OVERRIDE True

ATTRIBUTE Discrete

OLS \ TDistribution \

	Discrete

INHERITED True

FUNCTION InvDensity

OLS \ TDistribution \

	InvDensity
(t_FieldReal delta)	

OVERRIDE True

ATTRIBUTE High

OLS \ TDistribution \

	High

OVERRIDE True

ATTRIBUTE Low

OLS \ TDistribution \

	Low

INHERITED True

ATTRIBUTE RangeWidth

OLS \ TDistribution \

	RangeWidth

OVERRIDE True

FUNCTION Density

OLS \ TDistribution \

t_FieldReal	Density
(t_FieldReal t)	

OVERRIDE True

FUNCTION CumulativeV

OLS \ TDistribution \

	CumulativeV
()	

OVERRIDE True

FUNCTION Cumulative

OLS \ TDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

OVERRIDE True

MODULE FDistribution

OLS \

	FDistribution
(t_Discrete d1_in, t_Discrete d2_in, t_Count NRanges = 10000)	

Children

1. DensityV
2. CumulativeV
3. Cumulative
4. NTile
5. InvDensity
6. Discrete
7. Low
8. High
9. RangeWidth
10. Density

FUNCTION DensityV

OLS \ FDistribution \

DATASET(RangeVec)	DensityV
()	

OVERRIDE True

FUNCTION CumulativeV

OLS \ FDistribution \

	CumulativeV
()	

OVERRIDE True

FUNCTION Cumulative

OLS \ FDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

OVERRIDE True

FUNCTION NTile

OLS \ FDistribution \

t_FieldReal	NTile
(t_FieldReal Pc)	

OVERRIDE True

FUNCTION InvDensity

OLS \ FDistribution \

	InvDensity
`(t_FieldReal delta)`	

INHERITED True

ATTRIBUTE Discrete

OLS \ FDistribution \

	Discrete

INHERITED True

ATTRIBUTE Low

OLS \ FDistribution \

	Low

INHERITED True

ATTRIBUTE High

OLS \ FDistribution \

	High

OVERRIDE True

ATTRIBUTE RangeWidth

OLS \ FDistribution \

	RangeWidth

OVERRIDE True

FUNCTION Density

OLS \ FDistribution \

`t_FieldReal`	**Density**
`(t_FieldReal t)`	

OVERRIDE True

MODULE NormalDistribution

OLS \

	NormalDistribution
`(t_Count NRanges)`	

Children

1. Low
2. High
3. RangeWidth
4. DensityV
5. CumulativeV
6. Cumulative
7. NTile
8. InvDensity
9. Discrete
10. Density

ATTRIBUTE Low

OLS \ NormalDistribution \

	Low

INHERITED True

ATTRIBUTE High

OLS \ NormalDistribution \

	High

INHERITED True

ATTRIBUTE RangeWidth

OLS \ NormalDistribution \

	RangeWidth

OVERRIDE True

FUNCTION DensityV

OLS \ NormalDistribution \

DATASET(RangeVec)	DensityV
()	

OVERRIDE True

FUNCTION CumulativeV

OLS \ NormalDistribution \

	CumulativeV
()	

OVERRIDE True

FUNCTION Cumulative

OLS \ NormalDistribution \

t_FieldReal	Cumulative
(t_FieldReal t)	

OVERRIDE True

FUNCTION NTile

OLS \ NormalDistribution \

t_FieldReal	NTile
(t_FieldReal Pc)	

OVERRIDE True

FUNCTION InvDensity

OLS \ NormalDistribution \

	InvDensity
(t_FieldReal delta)	

INHERITED True

ATTRIBUTE Discrete

OLS \ NormalDistribution \

	Discrete

INHERITED True

FUNCTION Density

OLS \ NormalDistribution \

t_FieldReal	**Density**
(t_FieldReal t)	

OVERRIDE True

ATTRIBUTE pVal

OLS \

	pVal

P-Value Calculate the P-value for each coefficient, which is the probability that the coefficient is insignificant (i.e. actually zero). A low P-value (e.g. .05) provides evidence that the coefficient is significant in the model. A high P-value indicates that the coefficient value should, in fact, be zero. P-value is related to the T-Statistic, and can be thought of as a normalized version of the T-Statistic. Only meaningful during the training phase.

RETURN DATSET(NumericField), one record per Beta coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD ConfintRec

OLS \

	ConfintRec

FUNCTION ConfInt

OLS \

	ConfInt
`(Types.t_fieldReal level)`	

Confidence Interval The Confidence Interval determines the upper and lower bounds of each estimated coefficient given a confidence level (level) that is required. For example, one could say that there is a 95% probability (level) that the coefficient of the first independent variable is between 2.05 and 3.62. This allows error margins to be determined with the desired confidence level. If the confidence interval spans zero, it implies that the coefficient may not be significant at the specified confidence level.

PARAMETER **level** The level of confidence required, expressed as a percentage from 0.0 to 100.0

RETURN DATASET(ConfintRec) with one record per coefficient per dependent variable per work-item. The 'id' field is the coefficient number, with 1 being the Y intercept, 2 being the coefficient for the first feature, etc. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

RECORD FTestRec

OLS \

	FTestRec

ATTRIBUTE FTest

OLS \

DATASET(FTestRec)	**FTest**

F-Test Calculate the P-value for the full regression, which is the probability that all of the coefficients are insignificant (i.e. actually zero). A low P-value (e.g. .05) provides evidence that at least one coefficient is significant. A high P-value indicates that all the coefficient values should in fact be zero, implying that the regression has no statistically significant predictive power. P-value is related to the ANOVA F-Statistic, and can be thought of as a standardized version of the ANOVA F-Statistic. The F-Test and T-Test are similar, except that the T-test is used to test the significance of each coefficient, while the F-Test is used to test the significance of the entire regression. For simple linear regression (i.e. only one independent variable, the T-Test and F-Test are equivalent.

RETURN DATASET(FTestRec), one record per dependent variable per work-item. The number field indicates the dependent variable and corresponds to the number field of the dependent (Y) variable to which it applies.

LogisticRegression

Go Up

Name	LogisticRegression
Version	1.0.0
Description	Logistic Regression implementation
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core, PBblas
Platform	6.2.0

Table of Contents

Extract the beta values form the model dataset
ExtractBeta_CI.ecl Extract the beta values form the model dataset
ExtractBeta_pval.ecl Extract the beta values form the model dataset
ExtractReport.ecl Extract Report records from model
LogitPredict.ecl Predict the category values with the logit function and the the supplied beta coefficients
LogitScore.ecl Calculate the score using the logit function and the the supplied beta coefficients
Model_Deviance.ecl Model Deviance
Null_Deviance.ecl Deviance for the null model, that is, a model with only an intercept
Types.ecl
IRLS
performance
Tests
validation

LogisticRegression/

BinomialConfusion

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION BinomialConfusion

DATASET(Types.Binomial_Confusion_Summary)	**BinomialConfusion**
(DATASET(Core_Types.Confusion_Detail) d)	

Binomial confusion matrix. Work items with multinomial responses are ignored by this function. The higher value lexically is considered to be the positive indication.

PARAMETER **d** confusion detail for the work item and classifier

RETURN confusion matrix for a binomial classifier

BinomialLogisticRegression

Go Up

IMPORTS

LogisticRegression | LogisticRegression.Constants | ML_Core.Interfaces | ML_Core.Types |

DESCRIPTIONS

MODULE BinomialLogisticRegression

	BinomialLogisticRegression
	`(UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)`

Binomial logistic regression using iteratively re-weighted least squares.

PARAMETER **max_iter** maximum number of iterations to try

PARAMETER **epsilon** the minimum change in the Beta value estimate to continue

PARAMETER **ridge** a value to populate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

Children

1. GetModel : Calculate the model to fit the observation data to the observed classes

2. Classify : Classify the observations using a model
3. Report : Report the confusion matrix for the classifier and training data

---

FUNCTION GetModel

BinomialLogisticRegression \

DATASET(Types.Layout_Model)	**GetModel**
(DATASET(Types.NumericField) observations, DATASET(Types.DiscreteField) classifications)	

Calculate the model to fit the observation data to the observed classes.

PARAMETER **observations** the observed explanatory values

PARAMETER **classifications** the observed classification used to build the model

RETURN the encoded model

OVERRIDE True

FUNCTION Classify

BinomialLogisticRegression \

DATASET(Types.Classify_Result)	**Classify**
(DATASET(Types.Layout_Model) model, DATASET(Types.NumericField) new_observations)	

Classify the observations using a model.

PARAMETER **model** The model, which must be produced by a corresponding getModel function.

PARAMETER **new_observations** observations to be classified

RETURN Classification with a confidence value

OVERRIDE True

FUNCTION Report

BinomialLogisticRegression \

DATASET(Types.Confusion_Detail)	**Report**
(DATASET(Types.Layout_Model) model, DATASET(Types.NumericField) observations, DATASET(Types.DiscreteField) classifications)	

Report the confusion matrix for the classifier and training data.

PARAMETER **model** the encoded model

PARAMETER **observations** the explanatory values.

PARAMETER **classifications** the classifications associated with the observations

RETURN the confusion matrix showing correct and incorrect results

OVERRIDE True

LogisticRegression/

Confusion

Go Up

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Confusion

DATASET(Confusion_Detail)	**Confusion**
(DATASET(DiscreteField) dependents, DATASET(DiscreteField) predicts)	

Detail confusion records to compare actual versus predicted response variable values.

PARAMETER **dependents** the original response values

PARAMETER **predicts** the predicted responses

RETURN confusion counts by predicted and actual response values.

Constants

Go Up

DESCRIPTIONS

MODULE Constants

	Constants

Children

1. limit_card
2. default_epsilon
3. default_ridge
4. local_cap
5. id_base
6. id_iters
7. id_delta
8. id_correct
9. id_incorrect
10. id_stat_set
11. id_betas
12. id_betas_coef
13. id_betas_SE
14. base_builder
15. base_max_iter

16. base_epsilon
17. base_ind_vars
18. base_dep_vars
19. base_obs
20. builder_irls_local
21. builder_irls_global
22. builder_softmax

ATTRIBUTE limit_card

Constants \

UNSIGNED2	**limit_card**

ATTRIBUTE default_epsilon

Constants \

REAL8	**default_epsilon**

ATTRIBUTE default_ridge

Constants \

REAL8	**default_ridge**

ATTRIBUTE local_cap

Constants \

UNSIGNED4	local_cap

ATTRIBUTE id_base

Constants \

	id_base

ATTRIBUTE id_iters

Constants \

	id_iters

ATTRIBUTE id_delta

Constants \

	id_delta

ATTRIBUTE id_correct

Constants \

	id_correct

ATTRIBUTE id_incorrect

Constants \

	id_incorrect

ATTRIBUTE id_stat_set

Constants \

	id_stat_set

ATTRIBUTE id_betas

Constants \

	id_betas

ATTRIBUTE id_betas_coef

Constants \

	id_betas_coef

ATTRIBUTE id_betas_SE

Constants \

	id_betas_SE

ATTRIBUTE base_builder

Constants \

	base_builder

ATTRIBUTE base_max_iter

Constants \

	base_max_iter

ATTRIBUTE base_epsilon

Constants \

	base_epsilon

ATTRIBUTE base_ind_vars

Constants \

	base_ind_vars

ATTRIBUTE base_dep_vars

Constants \

	base_dep_vars

ATTRIBUTE base_obs

Constants \

	base_obs

ATTRIBUTE builder_irls_local

Constants \

	builder_irls_local

ATTRIBUTE builder_irls_global

Constants \

	builder_irls_global

ATTRIBUTE builder_softmax

Constants \

	builder_softmax

LogisticRegression/

DataStats

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |
LogisticRegression.Constants |

DESCRIPTIONS

FUNCTION DataStats

DATASET(Types.Data_Info)	**DataStats**
(DATASET(Core_Types.NumericField) indep, DATASET(Core_Types.DiscreteField) dep, BOOLEAN field_details=FALSE)	

Information about the datasets. Without details the range for the x and y (independent and dependent) columns. Note that a column of all zero values cannot be distinguished from a missing column. When details are requested, the cardinality, minimum, and maximum values are returned. A zero cardinality is returned when the field cardinality exceeds the Constants.limit_card value.

PARAMETER **indep** data set of independent variables

PARAMETER **dep** data set of dependent variables

PARAMETER **field_details** Boolean directive to provide field level info

---

LogisticRegression/

Deviance_Analysis

Go Up

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Deviance_Analysis

DATASET(Types.AOD_Record)	**Deviance_Analysis**
(DATASET(Types.Deviance_Record) proposed, DATASET(Types.Deviance_Record) base)	

Compare deviance information for an analysis of deviance.

PARAMETER **proposed** the proposed model

PARAMETER **base** the base model for comparison

RETURN the comparison of the deviance between the models

LogisticRegression/

Deviance_Detail

Go Up

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Deviance_Detail

DATASET(Types.Observation_Deviance)	**Deviance_Detail**
(DATASET(Core_Types.DiscreteField) dependents, DATASET(Types.Raw_Prediction) predicts)	

Detail deviance for each observation.

PARAMETER **dependents** original dependent records for the model

PARAMETER **predicts** the predicted values of the response variable

RETURN the deviance information by observation and the log likelihood of the predicted result.

---

dimm

Go Up

IMPORTS

std.blas | std.BLAS.Types |

DESCRIPTIONS

EMBED dimm

Types.matrix_t	dimm
(BOOLEAN transposeA, BOOLEAN transposeB, BOOLEAN diagonalA, BOOLEAN diagonalB, Types.dimension_t m, Types.dimension_t n, Types.dimension_t k, Types.value_t alpha, Types.matrix_t A, Types.matrix_t B, Types.value_t beta=0.0, Types.matrix_t C=[])	

Matrix multiply when either A or B is a diagonal and is passed as a vector. alpha*op(A) op(B) + beta*C where op() is transpose

PARAMETER **transposeA** true when transpose of A is used

PARAMETER **transposeB** true when transpose of B is used

PARAMETER **diagonalA** true when A is the diagonal matrix

PARAMETER **diagonalB** true when B is the diagonal matrix

PARAMETER **m** number of rows in product

PARAMETER **n** number of columns in product

PARAMETER **k** number of columns/rows for the multiplier/multiplicand

PARAMETER **alpha** scalar used on A

PARAMETER **A** matrix A

PARAMETER **B** matrix B

PARAMETER **beta** scalar for matrix C

PARAMETER **C** matrix C or empty

Distributions

Go Up

IMPORTS

ML_Core.Constants | ML_Core.Math |

DESCRIPTIONS

MODULE Distributions

	Distributions

Children

1. Normal_CDF : Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean
2. Normal_PPF : Normal Distribution Percentage Point Function
3. T_CDF : Students t distribution integral evaluated between negative infinity and x
4. T_PPF : Percentage point function for the T distribution
5. Chi2_CDF : The cumulative distribution function for the Chi Square distribution
6. Chi2_PPF : The Chi Squared PPF function

FUNCTION Normal_CDF

Distributions \

REAL8	**Normal_CDF**
(REAL8 x)	

Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean. Taken from C/C++ Mathematical Algorithms for Scientists and Engineers, n. Shammas, McGraw-Hill, 1995

PARAMETER **x** the number of standard deviations

FUNCTION Normal_PPF

Distributions \

REAL8	**Normal_PPF**
(REAL8 x)	

Normal Distribution Percentage Point Function. Translated from C/C++ Mathematical Algorithms for Scientists and Engineers, N. Shammas, McGraw-Hill, 1995

PARAMETER **x** probability

FUNCTION T_CDF

Distributions \

REAL8	**T_CDF**
(REAL8 x, REAL8 df)	

Students t distribution integral evaluated between negative infinity and x. Translated from NIST SEL DATAPAC Fortran TCDF.f source

PARAMETER **x** value of the evaluation

PARAMETER **df** degrees of freedom

FUNCTION T_PPF

Distributions \

REAL8	**T_PPF**
(REAL8 x, REAL8 df)	

Percentage point function for the T distribution. Translated from NIST SEL DATAPAC Fortran TPPF.f source

---

FUNCTION Chi2_CDF

Distributions \

REAL8	**Chi2_CDF**
(REAL8 x, REAL8 df)	

The cumulative distribution function for the Chi Square distribution. the CDF for the specfied degrees of freedom. Translated from the NIST SEL DATAPAC Fortran subroutine CHSCDF.

FUNCTION Chi2_PPF

Distributions \

REAL8	**Chi2_PPF**
(REAL8 x, REAL8 df)	

The Chi Squared PPF function. Translated from the NIST SEL DATAPAC Fortran subroutine CHSPPF.

LogisticRegression/

ExtractBeta

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION ExtractBeta

	ExtractBeta
(DATASET(Core_Types.Layout_Model) mod_ds)	

Extract the beta values form the model dataset.

PARAMETER **mod_ds** the model dataset

RETURN a beta values as Model Coefficient records, zero as the constant term.

ExtractBeta_CI

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION ExtractBeta_CI

DATASET(Types.Confidence_Model_Coef)	**ExtractBeta_CI**
(DATASET(Core_Types.Layout_Model) mod_ds, REAL8 level)	

Extract the beta values form the model dataset.

PARAMETER **mod_ds** the model dataset

PARAMETER **level** the significance value for the intervals

RETURN the beta values with confidence intervals term.

LogisticRegression/

ExtractBeta_pval

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION ExtractBeta_pval

DATASET(Types.pval_Model_Coef)	**ExtractBeta_pval**
(DATASET(Core_Types.Layout_Model) mod_ds)	

Extract the beta values form the model dataset.

PARAMETER **mod_ds** the model dataset

RETURN the beta values with p-values as Model Coefficient records, zero as the constant term.

ExtractReport

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |
LogisticRegression.Constants |

DESCRIPTIONS

FUNCTION ExtractReport

DATASET(Types.Model_Report)	**ExtractReport**
(DATASET(Core_Types.Layout_Model) mod_ds)	

Extract Report records from model

PARAMETER **mod_ds** the model dataset

RETURN the model report dataset

LogisticRegression/

LogitPredict

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION LogitPredict

DATASET(Classify_Result)	**LogitPredict**
(DATASET(Model_Coef) coef, DATASET(NumericField) independents)	

Predict the category values with the logit function and the the supplied beta coefficients.

PARAMETER **coef** the model beta coefficients

PARAMETER **independents** the observations

RETURN the predicted category values and a confidence score

LogisticRegression/

LogitScore

Go Up

IMPORTS

ML_Core.Types | LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION LogitScore

DATASET(Raw_Prediction)	**LogitScore**
(DATASET(Model_Coef) coef, DATASET(NumericField) independents)	

Calculate the score using the logit function and the the supplied beta coefficients.

PARAMETER **coef** the model beta coefficients

PARAMETER **independents** the observations

RETURN the raw prediction value

LogisticRegression/

Model_Deviance

Go Up

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Model_Deviance

DATASET(Types.Deviance_Record)	Model_Deviance
(DATASET(Types.Observation_Deviance) od, DATASET(Types.Model_Coef) mod)	

Model Deviance.

PARAMETER **od** observation deviance record

PARAMETER **mod** model co-efficients

RETURN model deviance

LogisticRegression/

Null_Deviance

Go Up

IMPORTS

LogisticRegression | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION Null_Deviance

DATASET(Types.Deviance_Record)	**Null_Deviance**
(DATASET(Types.Observation_Deviance) od)	

Deviance for the null model, that is, a model with only an intercept.

PARAMETER **od** Observation Deviance record set.

RETURN a data set of the null model deviances for each work item and classifier.

Types

Go Up

IMPORTS

ML_Core.Types |

DESCRIPTIONS

MODULE Types

	Types

Children

1. t_Universe
2. Field_Desc
3. Data_Info
4. NumericField_U
5. DiscreteField_U
6. Layout_Column_Map
7. Classifier_Stats
8. Model_Report
9. Binomial_Confusion_Summary
10. Model_Coef

11. Confidence_Model_Coef
12. pval_Model_Coef
13. Raw_Prediction
14. Observation_Deviance
15. Deviance_Record
16. AOD_Record

ATTRIBUTE t_Universe

Types \

	t_Universe

RECORD Field_Desc

Types \

	Field_Desc

RECORD Data_Info

Types \

	Data_Info

RECORD NumericField_U

Types \

	NumericField_U

RECORD DiscreteField_U

Types \

	DiscreteField_U

RECORD Layout_Column_Map

Types \

	Layout_Column_Map

RECORD Classifier_Stats

Types \

	Classifier_Stats

RECORD Model_Report

Types \

	Model_Report

RECORD Binomial_Confusion_Summary

Types \

	Binomial_Confusion_Summary

RECORD Model_Coef

Types \

	Model_Coef

RECORD Confidence_Model_Coef

Types \

	Confidence_Model_Coef

RECORD pval_Model_Coef

Types \

	pval_Model_Coef

RECORD Raw_Prediction

Types \

	Raw_Prediction

RECORD Observation_Deviance

Types \

	Observation_Deviance

RECORD Deviance_Record

Types \

	Deviance_Record

RECORD AOD_Record

Types \

	AOD_Record

IRLS

Go Up

Table of Contents

GetModel

Go Up

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Constants | LogisticRegression.Types | LogisticRegression.IRLS |

DESCRIPTIONS

FUNCTION GetModel

DATASET(Layout_Model)	**GetModel**
(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)	

Generate logistic regression model from training data. The size of the inputs is used to determin which work items are processed with purely local operations (the data is moved once as necessary) or with global operations supporting a work item to use multiple nodes.

PARAMETER **independents** the independent values

PARAMETER **dependents** the dependent values.

PARAMETER **max_iter** maximum number of iterations to try

PARAMETER **epsilon** the minimum change in the Beta value estimate to continue

PARAMETER **ridge** a value to pupulate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

RETURN coefficient matrix plus model building stats

GetModel_global

Go Up

IMPORTS

ML_Core | ML_Core.Types | PBblas | PBblas.Types | LogisticRegression | LogisticRegression.Constants | LogisticRegression.Types |

DESCRIPTIONS

FUNCTION GetModel_global

DATASET(Layout_Model)	GetModel_global
(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)	

Internal function to determine values for the model coefficients and selected statistics from building the model.

PARAMETER **independents** the independent values

PARAMETER **dependents** the dependent values

PARAMETER **max_iter** maximum number of iterations to try

PARAMETER **epsilon** the minimum change in the Beta value estimate to continue

PARAMETER **ridge** a value to pupulate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

RETURN coefficient matrix plus model building statistics

GetModel_local

Go Up

IMPORTS

ML_Core | ML_Core.Types | LogisticRegression | LogisticRegression.Constants | LogisticRegression.Types | LogisticRegression.IRLS | std | std.blas |

DESCRIPTIONS

FUNCTION GetModel_local

DATASET(Layout_Model)	GetModel_local
(DATASET(NumericField) independents, DATASET(DiscreteField) dependents, UNSIGNED2 max_iter=200, REAL8 epsilon=Constants.default_epsilon, REAL8 ridge=Constants.default_ridge)	

Internal function to determine values for the model co-efficients and selected stats from building the model.

PARAMETER **independents** the independent values

PARAMETER **dependents** the dependent values.

PARAMETER **max_iter** maximum number of iterations to try

PARAMETER **epsilon** the minimum change in the Beta value estimate to continue

PARAMETER **ridge** a value to populate a diagonal matrix that is added to a matrix help assure that the matrix is invertible.

RETURN coefficient matrix plus model building stats

---

performance

Go Up

Table of Contents

RunBinomial

Go Up

IMPORTS

LogisticRegression | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE RunBinomial

	RunBinomial

Tests

Go Up

Table of Contents

Check_Dist

Go Up

IMPORTS

LogisticRegression.Distributions | ML_Core | python |

DESCRIPTIONS

ATTRIBUTE Check_Dist

	Check_Dist

---

validation

Go Up

Table of Contents

LogisticRegression/ validation/

BinomialRegression

Go Up

IMPORTS

LogisticRegression.validation | LogisticRegression | ML_Core.Types | ML_Core | LogisticRegression.Types |

DESCRIPTIONS

ATTRIBUTE BinomialRegression

	BinomialRegression

---

discrete_GermanDS

Go Up

IMPORTS

ML_Core.Types |

DESCRIPTIONS

MODULE discrete_GermanDS

	discrete_GermanDS

Children

1. content

ATTRIBUTE content

discrete_GermanDS \

	content

IrisDS

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE irisDS

	irisDS

LogisticRegression/ validation/

unit_test_dimm

Go Up

IMPORTS

LogisticRegression | std.BLAS.Types |

DESCRIPTIONS

ATTRIBUTE unit_test_dimm

	unit_test_dimm

ML_Core

Go Up

Name	ML_Core
Version	3.1.0
Description	Common definitions for Machine Learning
License	SeeLICENSE.TXT
Copyright	Copyright (C) 2017 HPCC Systems
Authors	HPCCSystems
Platform	6.2.0

Table of Contents

AppendID

Go Up

DESCRIPTIONS

MACRO AppendID

	AppendID
(dIn,idfield,dOut)	

ML_Core/

AppendSeqID

Go Up

DESCRIPTIONS

MACRO AppendSeqID

	AppendSeqID
(dIn,idfield,dOut)	

ML_Core/

Config

Go Up

DESCRIPTIONS

MODULE Config

	Config

Children

1. MaxLookup
2. Discrete
3. RoundingError

ATTRIBUTE MaxLookup

Config \

	MaxLookup

ATTRIBUTE Discrete

Config \

	Discrete

ATTRIBUTE RoundingError

Config \

	RoundingError

Constants

Go Up

DESCRIPTIONS

MODULE Constants

	Constants

Useful constants

Children

1. Pi : Constant PI
2. Root_2 : Constant square root of 2

ATTRIBUTE Pi

Constants \

	Pi

Constant PI

ATTRIBUTE Root_2

Constants \

	Root_2

Constant square root of 2

ML_Core/

FieldAggregates

Go Up

IMPORTS

ML_Core | ML_Core.Types | ML_Core.Utils | std.system.ThorLib |

DESCRIPTIONS

MODULE FieldAggregates

	FieldAggregates
(DATASET(Types.NumericField) d)	

Children

1. Simple
2. SimpleRanked
3. Medians
4. MinMedNext
5. Buckets
6. BucketRanges
7. Modes
8. Cardinality
9. RankedInput

10. NTiles
11. NTileRanges

ATTRIBUTE Simple

FieldAggregates \

	Simple

ATTRIBUTE SimpleRanked

FieldAggregates \

	SimpleRanked

ATTRIBUTE Medians

FieldAggregates \

	Medians

ATTRIBUTE MinMedNext

FieldAggregates \

	MinMedNext

FUNCTION Buckets

FieldAggregates \

	Buckets
(Types.t_Discrete n)	

FUNCTION BucketRanges

FieldAggregates \

	BucketRanges
(Types.t_Discrete n)	

ATTRIBUTE Modes

FieldAggregates \

	Modes

ATTRIBUTE Cardinality

FieldAggregates \

	Cardinality

ATTRIBUTE RankedInput

FieldAggregates \

	RankedInput

FUNCTION NTiles

FieldAggregates \

	NTiles
(Types.t_Discrete n)	

FUNCTION NTileRanges

FieldAggregates \

	NTileRanges
(Types.t_Discrete n)	

ML_Core/

FromField

Go Up

DESCRIPTIONS

MACRO FromField

	FromField
(dIn,lOut,dOut,dMap=")	

ML_Core/

Generate

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE Generate

	Generate

Children

1. tp_Method
2. MethodName
3. ToPoly

ATTRIBUTE tp_Method

Generate \

	tp_Method

FUNCTION MethodName

Generate \

	MethodName
(tp_Method x)	

FUNCTION ToPoly

Generate \

	ToPoly
(DATASET(Types.NumericField) seedCol, UNSIGNED maxN=6)	

ML_Core/

ToField

Go Up

DESCRIPTIONS

MACRO ToField

	ToField
(dIn,dOut,idfield=", wifield=", wivalue=",datafields=")	

Types

Go Up

DESCRIPTIONS

MODULE Types

	Types

Children

1. t_RecordID
2. t_FieldNumber
3. t_FieldReal
4. t_FieldSign
5. t_Discrete
6. t_Item
7. t_Count
8. t_Work_Item
9. AnyField
10. NumericField
11. DiscreteField
12. Layout_Model
13. Classify_Result
14. l_result
15. Confusion_Detail

16. ItemElement
17. t_node
18. t_level
19. NodeID

ATTRIBUTE t_RecordID

Types \

	t_RecordID

ATTRIBUTE t_FieldNumber

Types \

	t_FieldNumber

ATTRIBUTE t_FieldReal

Types \

	t_FieldReal

ATTRIBUTE t_FieldSign

Types \

	t_FieldSign

ATTRIBUTE t_Discrete

Types \

	t_Discrete

ATTRIBUTE t_Item

Types \

	t_Item

ATTRIBUTE t_Count

Types \

	t_Count

ATTRIBUTE t_Work_Item

Types \

	t_Work_Item

RECORD AnyField

Types \

	AnyField

RECORD NumericField

Types \

	NumericField

RECORD DiscreteField

Types \

	DiscreteField

RECORD Layout_Model

Types \

	Layout_Model

RECORD Classify_Result

Types \

	Classify_Result

RECORD l_result

Types \

	l_result

RECORD Confusion_Detail

Types \

	Confusion_Detail

RECORD ItemElement

Types \

	ItemElement

ATTRIBUTE t_node

Types \

	t_node

ATTRIBUTE t_level

Types \

	t_level

RECORD NodeID

Types \

	NodeID

Interfaces

Go Up

Table of Contents

IClassify

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE IClassify

	IClassify

Interface definition for Classification. Actual implementation modules will probably take parameters.

Children

1. GetModel : Calculate the model to fit the observation data to the observed classes
2. Classify : Classify the observations using a model
3. Report : Report the confusion matrix for the classifier and training data

---

FUNCTION GetModel

IClassify \

DATASET(Types.Layout_Model)	GetModel
(DATASET(Types.NumericField) observations, DATASET(Types.DiscreteField) classifications)	

Calculate the model to fit the observation data to the observed classes.

PARAMETER **observations** the observed explanatory values

PARAMETER **classifications** the observed classification used to build the model

RETURN the encoded model

FUNCTION Classify

IClassify \

DATASET(Types.Classify_Result)	Classify
(DATASET(Types.Layout_Model) model, DATASET(Types.NumericField) new_observations)	

Classify the observations using a model.

PARAMETER **model** The model, which must be produced by a corresponding getModel function.

PARAMETER **new_observations** observations to be classified

RETURN Classification with a confidence value

FUNCTION Report

IClassify \

DATASET(Types.Confusion_Detail)	Report
(DATASET(Types.Layout_Model) model, DATASET(Types.NumericField) observations, DATASET(Types.DiscreteField) classifications)	

Report the confusion matrix for the classifier and training data.

PARAMETER **model** the encoded model

PARAMETER **observations** the explanatory values.

PARAMETER **classifications** the classifications associated with the observations

RETURN the confusion matrix showing correct and incorrect results

---

IRegression

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE IRegression

	IRegression
(DATASET(NumericField) X=empty_data, DATASET(NumericField) Y=empty_data)	

Interface Definition for Regression Modules Regression learns a function that maps a set of input data to one or more output variables. The resulting learned function is known as the model. That model can then be used repetitively to predict (i.e. estimate) the output value(s) based on new input data.

PARAMETER **X** The independent data in DATASET(NumericField) format. Each statistical unit (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number').

PARAMETER **Y** The dependent variable(s) in DATASET(NumericField) format. Each statistical unit (e.g. record) is identified by 'id', and each feature is identified by field number (i.e. 'number').

Children

1. GetModel : Calculate and return the 'learned' model The model may be persisted and later used to make predictions using 'Predict' below
2. Predict : Predict the output variable(s) based on a previously learned model

ATTRIBUTE GetModel

IRegression \

DATASET(Layout_Model)	**GetModel**

Calculate and return the 'learned' model The model may be persisted and later used to make predictions using 'Predict' below.

RETURN DATASET(LayoutModel) describing the learned model parameters

---

FUNCTION Predict

IRegression \

DATASET(NumericField)	**Predict**
(DATASET(NumericField) newX, DATASET(Layout_Model) model)	

Predict the output variable(s) based on a previously learned model

PARAMETER **newX** DATASET(NumericField) containing the X values to b predicted.

RETURN DATASET(NumericField) containing one entry per observation (i.e. id) in newX. This represents the predicted values for Y.

Math

Go Up

Table of Contents

Beta

Go Up

IMPORTS

ML_Core.Math |

DESCRIPTIONS

FUNCTION Beta

	Beta
(REAL8 x, REAL8 y)	

Return the beta value of two positive real numbers, x and y

PARAMETER x the value of the first number

PARAMETER y the value of the second number

RETURN the beta value

Distributions

Go Up

IMPORTS

ML_Core.Constants | ML_Core.Math |

DESCRIPTIONS

MODULE Distributions

	Distributions

Children

1. Normal_CDF : Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean
2. Normal_PPF : Normal Distribution Percentage Point Function
3. T_CDF : Students t distribution integral evaluated between negative infinity and x
4. T_PPF : Percentage point function for the T distribution
5. Chi2_CDF : The cumulative distribution function for the Chi Square distribution
6. Chi2_PPF : The Chi Squared PPF function

FUNCTION Normal_CDF

Distributions \

REAL8	**Normal_CDF**
(REAL8 x)	

Cumulative Distribution of the standard normal distribution, the probability that a normal random variable will be smaller than x standard deviations above or below the mean. Taken from C/C++ Mathematical Algorithms for Scientists and Engineers, n. Shammas, McGraw-Hill, 1995

PARAMETER **x** the number of standard deviations

FUNCTION Normal_PPF

Distributions \

REAL8	**Normal_PPF**
(REAL8 x)	

Normal Distribution Percentage Point Function. Translated from C/C++ Mathematical Algorithms for Scientists and Engineers, N. Shammas, McGraw-Hill, 1995

PARAMETER **x** probability

FUNCTION T_CDF

Distributions \

REAL8	**T_CDF**
(REAL8 x, REAL8 df)	

Students t distribution integral evaluated between negative infinity and x. Translated from NIST SEL DATAPAC Fortran TCDF.f source

PARAMETER **x** value of the evaluation

PARAMETER **df** degrees of freedom

FUNCTION T_PPF

Distributions \

REAL8	T_PPF
(REAL8 x, REAL8 df)	

Percentage point function for the T distribution. Translated from NIST SEL DATAPAC Fortran TPPF.f source

---

FUNCTION Chi2_CDF

Distributions \

REAL8	Chi2_CDF
(REAL8 x, REAL8 df)	

The cumulative distribution function for the Chi Square distribution. the CDF for the specfied degrees of freedom. Translated from the NIST SEL DATAPAC Fortran subroutine CHSCDF.

FUNCTION Chi2_PPF

Distributions \

REAL8	Chi2_PPF
(REAL8 x, REAL8 df)	

The Chi Squared PPF function. Translated from the NIST SEL DATAPAC Fortran subroutine CHSPPF.

DoubleFac

Go Up

DESCRIPTIONS

EMBED DoubleFac

REAL8	**DoubleFac**
(INTEGER2 i)	

The 'double' factorial is defined for ODD n and is the product of all the odd numbers up to and including that number. We are extending the meaning to even numbers to mean the product of the even numbers up to and including that number. Thus DoubleFac(8) = 8*6*4*2 We also defend against i < 2 (returning 1.0)

PARAMETER **i** the value used in the calculation

RETURN the factorial of the sequence, declining by 2

Fac

Go Up

DESCRIPTIONS

EMBED Fac

REAL8	**Fac**
(UNSIGNED2 i)	

Factorial function

PARAMETER **i** the value used, (i)(i-1)(i-2)...(2)

RETURN the factorial i!

---

gamma

Go Up

DESCRIPTIONS

EMBED gamma

REAL8	**gamma**
(REAL8 x)	

Return the value of gamma function of real number x A wrapper for the standard C tgamma function.

PARAMETER **x** the input x

RETURN the value of GAMMA evaluated at x

---

log_gamma

Go Up

DESCRIPTIONS

EMBED log_gamma

REAL8	**log_gamma**
(REAL8 x)	

Return the value of the log gamma function of the absolute value of X. A wrapper for the standard C lgamma function. Avoids the race condition found on some platforms by taking the absolute value of the of the input argument.

PARAMETER **x** the input x

RETURN the value of the log of the GAMMA evaluated at ABS(x)

lowerGamma

Go Up

DESCRIPTIONS

EMBED lowerGamma

REAL8	**lowerGamma**
(REAL8 x, REAL8 y)	

Return the lower incomplete gamma value of two real numbers, x and y

PARAMETER **x** the value of the first number

PARAMETER **y** the value of the second number

RETURN the lower incomplete gamma value

NCK

Go Up

IMPORTS

ML_Core.Math |

DESCRIPTIONS

FUNCTION NCK

REAL8	NCK
(INTEGER2 N, INTEGER2 K)	

Poly

Go Up

DESCRIPTIONS

EMBED Poly

REAL8	**Poly**
(REAL8 x, SET OF REAL8 Coeffs)	

Evaluate a polynomial from a set of co-effs. Co-effs 1 is assumed to be the HIGH order of the equation. Thus for axˆ2+bx+c - the set would need to be Coef := [a,b,c];

PARAMETER **x** the value of x in the polynomial

PARAMETER **Coeffs** a set of coefficients forthe polynomial. The ALL set is considered to be all zero values

RETURN value of the polynomial at x

StirlingFormula

Go Up

IMPORTS

ML_Core.Math | ML_Core.Constants |

DESCRIPTIONS

FUNCTION StirlingFormula

	StirlingFormula
(REAL x)	

Stirling's formula

PARAMETER **x** the point of evaluation

RETURN evaluation result

---

upperGamma

Go Up

DESCRIPTIONS

EMBED upperGamma

REAL8	**upperGamma**
(REAL8 x, REAL8 y)	

Return the upper incomplete gamma value of two real numbers, x and y.

PARAMETER **x** the value of the first number

PARAMETER **y** the value of the second number

RETURN the upper incomplete gamma value

Tests

Go Up

Table of Contents

ML_Core/ Tests/

Check_Dist

Go Up

IMPORTS

ML_Core.Math.Distributions | ML_Core | python |

DESCRIPTIONS

ATTRIBUTE Check_Dist

	Check_Dist

---

field_aggregates

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE field_aggregates

	field_aggregates

---

generate

Go Up

IMPORTS

ML_Core |

DESCRIPTIONS

ATTRIBUTE generate

	generate

---

test_appends

Go Up

IMPORTS

ML_Core | std.system.thorlib |

DESCRIPTIONS

ATTRIBUTE test_appends

	test_appends

---

test_discrete

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE test_discrete

	test_discrete

---

to_from

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

ATTRIBUTE to_from

	to_from

ML_Core/ Tests/

Validate_Betas

Go Up

IMPORTS

ML_Core | ML_Core.Math | python |

DESCRIPTIONS

ATTRIBUTE Validate_Betas

	Validate_Betas

Validate_Gammas

Go Up

IMPORTS

ML_Core | ML_Core.Math | python |

DESCRIPTIONS

ATTRIBUTE Validate_Gammas

	Validate_Gammas

Utils

Go Up

Table of Contents

Fat

Go Up

IMPORTS

ML_Core.Types |

DESCRIPTIONS

FUNCTION Fat

DATASET(Types.NumericField)	**Fat**
(DATASET(Types.NumericField) d0, Types.t_FieldReal v=0)	

Will take a potentially sparse file d and fill in the missing with value v for Numeric Field datasets

PARAMETER **d0** They myriad format Numeric Field dataset to be filled

PARAMETER **v** The value to assign missing records

RETURN A full Numeric Field dataset with every field populated

---

FatD

Go Up

IMPORTS

ML_Core.Types |

DESCRIPTIONS

FUNCTION FatD

DATASET(Types.DiscreteField)	**FatD**
(DATASET(Types.DiscreteField) d0, Types.t_Discrete v=0)	

Will take a potentially sparse file d and fill in the missing with value v for Discrete Field datasets

PARAMETER **d0** They myriad format Discrete Field dataset to be filled

PARAMETER **v** The value to assign missing records

RETURN A full Discrete Field dataset with every field populated

---

Gini

Go Up

DESCRIPTIONS

MACRO Gini

	Gini
(infile, pivot, target, wi_name='wi')	

Creates a file of pivot/target pairs with a Gini impurity value.

PARAMETER **infile** the input file, any type with a work item field

PARAMETER **pivot** the name of the pivot field

PARAMETER **target** the name of the field used as the target

PARAMETER **wi_name** the name of the work item field, default is "wi" return A table by Work Item and Pivot value giving count and Gini impurity value

---

SequenceInField

Go Up

DESCRIPTIONS

MACRO SequenceInField

	SequenceInField
(infile,infield,seq,wi_name='wi')	

Given a file which is sorted by the work item identifier and INFIELD (and possibly other values), add sequence numbers within the range of each infield. Slighly elaborate code is to avoid having to partition the data to one value of infield per node and to work with very large numbers of records where a global count project would be inappropriate. This is useful for assigning rank positions with the groupings.

PARAMETER **infile** the input file, any type

PARAMETER **infield** field name of grouping field

PARAMETER **seq** name of the field to receive the sequence number

PARAMETER **wi_name** work item field name, default is wi

RETURN a file of the same type with sequence numbers applied

PBblas

Go Up

Name	PBblas
Version	3.0.1
Description	Parallel Block Basic Linear Algebra Subsystem
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2016, 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core
Platform	6.2.0

Table of Contents

Element-wise multiplication of X * Y
IElementFunc.ecl Function prototype for a function to apply to each element of the
MatUtils.ecl Provides various utility attributes for manipulating cell-based matrixes
potrf.ecl Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested
scal.ecl Scale a matrix by a constant Result is alpha * X This supports a ”myriad” style interface in that X may be a set of independent matrices separated by different work-item ids
tran.ecl Transpose a matrix and sum into base matrix
trsm.ecl Partitioned block parallel triangular matrix solver
Types.ecl Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes
Vector2Diag.ecl Convert a vector into a diagonal matrix

PBblas/

Apply2Elements

Go Up

IMPORTS

PBblas | PBblas.Types | std.blas |

DESCRIPTIONS

FUNCTION Apply2Elements

DATASET(Layout_Cell)	**Apply2Elements**
(DATASET(Layout_Cell) X, IElementFunc f)	

Apply a function to each element of the matrix Use PBblas.IElementFunc as the prototype function. Input and ouput may be a single matrix, or myriad matrixes with different work item ids.

PARAMETER **X** A matrix (or multiple matrices) in Layout_Cell form

PARAMETER **f** A function based on the IElementFunc prototype

RETURN A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/IElementFunc

SEE PBblas/Types.Layout_Cell

PBblas/

asum

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.blas |

DESCRIPTIONS

FUNCTION asum

DATASET(Layout_Norm)	**asum**
(DATASET(Layout_Cell) X)	

Absolute sum – the "Entrywise" 1-norm Compute SUM(ABS(X))

PARAMETER **X** Matrix or set of matrices in Layout_Cell format

RETURN DATASET(Layout_Norm) with one record per work item

SEE PBblas/Types.Layout_Cell

---

axpy

Go Up

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION axpy

DATASET(Layout_Cell)	**axpy**
(value_t alpha, DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Implements alpha*X + Y X and Y must have same shape

PARAMETER **alpha** Scalar multiplier for the X matrix

PARAMETER **X** X matrix in DATASET(Layout_Cell) form

PARAMETER **Y** Y matrix in DATASET(Layout_Call) form

RETURN Matrix in DATASET(Layout_Cell) form

SEE PBblas/Types.layout_cell

Constants

Go Up

DESCRIPTIONS

MODULE Constants

	Constants

Children

1. Block_Minimum
2. Block_NoSplit
3. Block_Maximum
4. Block_Vec_Rows
5. Dimension_Incompat
6. Dimension_IncompatZ
7. Distribution_Error
8. Distribution_ErrorZ
9. Not_Square
10. Not_SquareZ
11. Not_PositiveDef
12. Not_PositiveDefZ
13. Not_Single_Block
14. Not_Single_BlockZ
15. Not_Block_Vector

16. Not_Block_VectorZ

ATTRIBUTE Block_Minimum

Constants \

	Block_Minimum

ATTRIBUTE Block_NoSplit

Constants \

	Block_NoSplit

ATTRIBUTE Block_Maximum

Constants \

	Block_Maximum

ATTRIBUTE Block_Vec_Rows

Constants \

	Block_Vec_Rows

ATTRIBUTE Dimension_Incompat

Constants \

	Dimension_Incompat

ATTRIBUTE Dimension_IncompatZ

Constants \

	Dimension_IncompatZ

ATTRIBUTE Distribution_Error

Constants \

	Distribution_Error

ATTRIBUTE Distribution_ErrorZ

Constants \

	Distribution_ErrorZ

ATTRIBUTE Not_Square

Constants \

	Not_Square

ATTRIBUTE Not_SquareZ

Constants \

	Not_SquareZ

ATTRIBUTE Not_PositiveDef

Constants \

	Not_PositiveDef

ATTRIBUTE Not_PositiveDefZ

Constants \

	Not_PositiveDefZ

ATTRIBUTE Not_Single_Block

Constants \

	Not_Single_Block

ATTRIBUTE Not_Single_BlockZ

Constants \

	Not_Single_BlockZ

ATTRIBUTE Not_Block_Vector

Constants \

	Not_Block_Vector

ATTRIBUTE Not_Block_VectorZ

Constants \

	Not_Block_VectorZ

Converted

Go Up

IMPORTS

PBblas | PBblas.Types | ML_Core.Types |

DESCRIPTIONS

MODULE Converted

	Converted

Module to convert between ML_Core/Types Field layouts (i.e. NumericField and DiscreteField) and PBblas matrix layout (i.e. Layout_Cell)

Children

1. NFToMatrix : Convert NumericField dataset to Matrix
2. DFToMatrix : Convert DiscreteField dataset to Matrix
3. MatrixToNF : Convert Matrix to NumericField dataset
4. MatrixToDF : Convert Matrix to DiscreteField dataset

FUNCTION NFToMatrix

Converted \

DATASET(Layout_Cell)	**NFToMatrix**
(DATASET(NumericField) recs)	

Convert NumericField dataset to Matrix

PARAMETER **recs** Record Dataset in DATASET(NumericField) format

RETURN Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION DFToMatrix

Converted \

DATASET(Layout_Cell)	**DFToMatrix**
(DATASET(DiscreteField) recs)	

Convert DiscreteField dataset to Matrix

PARAMETER **recs** Record Dataset in DATASET(DiscreteField) format

RETURN Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

FUNCTION MatrixToNF

Converted \

DATASET(NumericField)	**MatrixToNF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to NumericField dataset

PARAMETER **mat** Matrix in DATASET(Layout_Cell) format

RETURN NumericField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION MatrixToDF

Converted \

DATASET(DiscreteField)	**MatrixToDF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to DiscreteField dataset

PARAMETER **mat** Matrix in DATASET(Layout_Cell) format

RETURN DiscreteField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

PBblas/

ExtractTri

Go Up

IMPORTS

PBblas | std.blas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted |

DESCRIPTIONS

FUNCTION ExtractTri

DATASET(Layout_Cell)	**ExtractTri**
(Triangle tri, Diagonal dt, DATASET(Layout_Cell) A)	

Extract the upper or lower triangle from the composite output from getrf (LU Factorization).

PARAMETER **tri** Triangle type: Upper or Lower (see Types.Triangle)

PARAMETER **dt** Diagonal type: Unit or non unit (see Types.Diagonal)

PARAMETER **A** Matrix of cells. See Types.Layout_Cell

RETURN Matrix of cells in Layout_Cell format representing a triangular matrix (upper or lower)

SEE Std.PBblas.Types

PBblas/

gemm

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.blas | PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION gemm

DATASET(Layout_Cell)	**gemm**
(BOOLEAN transposeA, BOOLEAN transposeB, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in, DATASET(Layout_Cell) C_in=emptyC, value_t beta=0.0)	

Extended Parallel Block Matrix Multiplication Module Implements: Result = alpha * op(A)op(B) + beta * C. op is No Transpose or Transpose. Multiplies two matrixes A and B, with an optional pre-multiply transpose for each Optionally scales the product by the scalar "alpha". Then adds an optional C matrix to the product after scaling C by the scalar "beta". A, B, and C are specified as DATASET(Layout_Cell), as is the Resulting matrix. Layout_Cell describes a sparse matrix stored as a list of x, y, and value. This interface also provides a "Myriad" capability allowing multiple similar operations to be performed on independent sets of matrixes in parallel. This is done by use of the work-item id (wi_id) in each cell of the matrixes. Cells with the same wi_id are considered part of the same matrix. In the myriad form, each input matrix A, B, and (optionally) C can contain many independent matrixes. The wi_ids are matched up such that each operation involves the A, B, and C with the same wi_id. A and B must therefore contain the same set of wi_ids, while C is optional for any wi_id. The same parameters: alpha, beta, transposeA, and transposeB are used for all work-items. The result will contain cells from all provided work-items. Result has same shape as C if provided. Note that matrixes are not explicitly

dimensioned. The shape is determined by the highest value of x and y for each work-item.

PARAMETER **transposeA** Boolean indicating whether matrix A should be transposed before multiplying

PARAMETER **transposeB** Same as above but for matrix B

PARAMETER **alpha** Scalar multiplier for alpha * A * B

PARAMETER **A_in** 'A' matrix (multiplier) in Layout_Cell format

PARAMETER **B_in** Same as above for the 'B' matrix (multiplicand)

PARAMETER **C_in** Same as above for the 'C' matrix (addend). May be omitted.

PARAMETER **beta** A scalar multiplier for beta * C, scales the C matrix before addition. May be omitted.

RETURN Result matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

---

getrf

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.blas | PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION getrf

DATASET(Layout_Cell)	**getrf**
(DATASET(Layout_Cell) A)	

LU Factorization Splits a matrix into Lower and Upper triangular factors Produces composite LU matrix for the diagonal blocks. Iterates through the matrix a row of blocks and column of blocks at a time. Partition A into M block rows and N block columns. The A11 cell is a single block. A12 is a single row of blocks with N-1 columns. A21 is a single column of blocks with M-1 rows. A22 is a sub-matrix of M-1 x N-1 blocks. | A11 A12 | | L11 0 | | U11 U12 | | A21 A22 | == | L21 L22 | * | 0 U22 | | L11*U11 L11*U12 | == | L21*U11 L21*U12 + L22*U22 | Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id). Note: The returned matrix includes both the upper and lower factors. This matrix can be used directly by trsm which will only use the part indicated by trsm's 'triangle' parameter (i.e. upper or lower). To extract the upper or lower triangle explicitly for other purposes, use the ExtractTri function. When passing the Lower matrix to the triangle solver (trsm), set the ”Diagonal” parameter to ”UnitTri”. This is necessary because both triangular matrixes returned from this function are packed into a square matrix with only one diagonal. By convention, The Lower triangle is assumed to be a Unit Triangle (diagonal all ones), so the diagonal

contained in the returned matrix is for the Upper factor and must be ignored (i.e. assumed to be all ones) when referencing the Lower triangle.

PARAMETER **A** The input matrix in Types.Layout_Cell format

RETURN Resulting factored matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE ExtractTri

PBblas/

HadamardProduct

Go Up

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types | PBblas.internal.Types | PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

FUNCTION HadamardProduct

DATASET(Layout_Cell)	**HadamardProduct**
(DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Element-wise multiplication of X * Y. Supports the "myriad" style interface – X and Y may contain multiple separate matrixes. Each X will be multiplied by the Y with the same work-item id. Note: This performs element-wise multiplication. For dot-product matrix multiplication, use PBblas.gemm.

PARAMETER **X** A matrix (or multiple matrices) in Layout_Cell form

PARAMETER **Y** A matrix (or multiple matrices) in Layout_Cell form

RETURN A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/Types.Layout_Cell

PBblas/

IElementFunc

Go Up

IMPORTS

PBblas |

DESCRIPTIONS

FUNCTION IElementFunc

value_t	**IElementFunc**
(value_t v, dimension_t r, dimension_t c)	

Function prototype for a function to apply to each element of the distributed matrix Base your function on this prototype:

PARAMETER **v** Input value

PARAMETER **r** Row number (1 based)

PARAMETER **c** Column number (1 based)

RETURN Output value

SEE PBblas/Apply2Elements

PBblas/

MatUtils

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims |

DESCRIPTIONS

MODULE MatUtils

	MatUtils

Provides various utility attributes for manipulating cell-based matrixes

SEE Std/PBblas/Types.Layout_Cell

Children

1. GetWorkItems : Get a list of work-item ids from a matrix containing one or more work items
2. InsertCols : Insert one or more columns of a fixed value into a matrix
3. Transpose : Transpose a matrix This attribute supports the myriad interface

FUNCTION GetWorkItems

MatUtils \

DATASET(Layout_WI_ID)	**GetWorkItems**
(DATASET(Layout_Cell) cells)	

Get a list of work-item ids from a matrix containing one or more work items

PARAMETER **cells** A matrix in Layout_Cell format

RETURN DATASET(Layout_WI_ID), one record per work-item

SEE PBblas/Types.Layout_Cell

SEE PBblas/Types.Layout_WI_ID

FUNCTION InsertCols

MatUtils \

DATASET(Layout_Cell)	**InsertCols**
(DATASET(Layout_Cell) M, UNSIGNED cols_to_insert=1, value_t insert_val=1)	

Insert one or more columns of a fixed value into a matrix. Columns are inserted before the first original column. This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER **M** the input matrix

PARAMETER **cols_to_insert** the number of columns to insert, default 1

PARAMETER **insert_val** the value for each cell of the new column(s), default 0

RETURN matrix in Layout_Cell format with additional column(s)

FUNCTION Transpose

MatUtils \

DATASET(Layout_Cell)	**Transpose**
(DATASET(Layout_Cell) M)	

Transpose a matrix This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER **M** A matrix represented as DATASET(Layout_Cell)

RETURN Transposed matrix in Layout_Cell format

SEE PBblas/Types.Layout_Cell

PBblas/

potrf

Go Up

IMPORTS

PBblas | PBblas.Types | std.blas | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION potrf

DATASET(Layout_Cell)	**potrf**
(Triangle tri, DATASET(Layout_Cell) A_in)	

Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested. The matrix A must be symmetric positive definite.

```
| A11   A12 |      |  L11   0   |    | L11**T     L21**T |
| A21   A22 |  ==  |  L21   L22 | *  |  0           L22  |

                   | L11*L11**T          L11*L21**T      |
               ==  | L21*L11**T  L21*L21**T + L22*L22**T |
```

So, use Cholesky on the first block to get L11. L21 = A21*L11**T**-1 which can be found by dtrsm on each column block A22' is A22 - L21*L21**T

Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra

This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id).

PARAMETER **tri** Types.Triangle enumeration indicating whether we are looking for the Upper or the Lower factor

PARAMETER **A_in** The matrix or matrixes to be factored in Types.Layout_Cell format

RETURN Triangular matrix in Layout_Cell format

SEE Std.PBblas.Types.Layout_Cell

SEE Std.PBblas.Types.Triangle

PBblas/

scal

Go Up

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION scal

DATASET(Layout_Cell)	scal
(value_t alpha, DATASET(Layout_Cell) X)	

Scale a matrix by a constant Result is alpha * X This supports a ”myriad” style interface in that X may be a set of independent matrices separated by different work-item ids.

PARAMETER **alpha** A scalar multiplier

PARAMETER **X** The matrix(es) to be scaled in Layout_Cell format

RETURN Matrix in Layout_Cell form, of the same shape as X

SEE PBblas/Types.Layout_Cell

---

tran

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

FUNCTION tran

DATASET(Layout_Cell)	**tran**
(value_t alpha, DATASET(Layout_Cell) A, value_t beta=0, DATASET(Layout_Cell) C=empty_c)	

Transpose a matrix and sum into base matrix result <== alpha * A**t + beta * C, A is n by m, C is m by n A**T (A Transpose) and C must have same shape

PARAMETER **alpha** Scalar multiplier for the A**T matrix

PARAMETER **A** A matrix in DATASET(Layout_Cell) form

PARAMETER **beta** Scalar multiplier for the C matrix

PARAMETER **C** C matrix in DATASET(Layout_Call) form

RETURN Matrix in DATASET(Layout_Cell) form alpha * A**T + beta * C

SEE PBblas/Types.layout_cell

trsm

Go Up

IMPORTS

PBblas | PBblas.Types | std.blas | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION trsm

DATASET(Layout_Cell)	**trsm**
(Side s, Triangle tri, BOOLEAN transposeA, Diagonal diag, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in)	

Partitioned block parallel triangular matrix solver. Solves for X using: AX = B or XA = B A is is a square triangular matrix, X and B have the same dimensions. A may be an upper triangular matrix (UX = B or XU = B), or a lower triangular matrix (LX = B or XL = B). Allows optional transposing and scaling of A. Partially based upon an approach discussed by MJ DAYDE, IS DUFF, AP CERFACS. A Parallel Block implementation of Level-3 BLAS for MIMD Vector Processors ACM Tran. Mathematical Software, Vol 20, No 2, June 1994 pp 178-193 and other papers about PB-BLAS by Choi and Dongarra This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. Corresponding A and B matrixes are related by a common work-item identifier (wi_id) within each cell of the matrix. The returned X matrix will contain cells for the same set of work-items as specified for the A and B matrices.

PARAMETER **s** Types.Side enumeration indicating whether we are solving AX = B or XA = B

PARAMETER **tri** Types.Triangle enumeration indicating whether we are solving an Upper or Lower triangle.

PARAMETER **transposeA** Boolean indicating whether or not to transpose the A matrix before solving

PARAMETER **diag** Types.Diagonal enumeration indicating whether A is a unit matrix or not. This is primarily used after factoring matrixes using getrf (LU factorization). That module produces a factored matrix stored within the same space as the original matrix. Since the diagonal is used by both factors, by convention, the Lower triangle has a unit matrix (diagonal all 1's) while the Upper triangle uses the diagonal cells. Setting this to UnitTri, causes the contents of the diagonal to be ignored, and assumed to be 1. NotUnitTri should be used for most other cases.

PARAMETER **alpha** Multiplier to scale A

PARAMETER **A_in** The A matrix in Layout_Cell format

PARAMETER **B_in** The B matrix in Layout_Cell format

RETURN X solution matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE Types.Triangle

SEE Types.Side

Types

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE Types

	Types

Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes.

Children

1. dimension_t : Type for matrix dimensions
2. partition_t : Type for partition id – only supports up to 64K partitions
3. work_item_t : Type for work-item id – only supports up to 64K work items
4. value_t : Type for matrix cell values
5. m_label_t : Type for matrix label
6. Triangle : Enumeration for Triangle type
7. Diagonal : Enumeration for Diagonal type
8. Side : Enumeration for Side type

9. t_mu_no : Type for matrix universe number
10. Layout_Cell : Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions
11. Layout_Norm : Layout for Norm results

ATTRIBUTE dimension_t

Types \

	dimension_t

Type for matrix dimensions. Uses UNSIGNED four as matrixes are not designed to support more than 4 B rows or columns.

ATTRIBUTE partition_t

Types \

	partition_t

Type for partition id – only supports up to 64K partitions

ATTRIBUTE work_item_t

Types \

	work_item_t

Type for work-item id – only supports up to 64K work items

ATTRIBUTE value_t

Types \

	value_t

Type for matrix cell values WARNING: type used in C++ attribute

---

ATTRIBUTE m_label_t

Types \

	m_label_t

Type for matrix label. Used for Matrix dimensions (see Layout_Dims) and for partitions (see Layout_Part)

ATTRIBUTE Triangle

Types \

	Triangle

Enumeration for Triangle type WARNING: type used in C++ attribute

---

ATTRIBUTE Diagonal

Types \

	Diagonal

Enumeration for Diagonal type WARNING: type used in C++ attribute

ATTRIBUTE Side

Types \

	Side

Enumeration for Side type WARNING: type used in C++ attribute

---

ATTRIBUTE t_mu_no

Types \

	t_mu_no

Type for matrix universe number Allow up to 64k matrices in one universe

---

RECORD Layout_Cell

Types \

	Layout_Cell

Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions. Matrixes are represented as DATASET(Layout_Cell), where each cell describes the row and column position of the cell as well as its value. Only the non-zero cells need to be contained in the dataset in order to describe the matrix since all unspecified cells are considered to have a value of zero. The cell also contains a work-item number that allows multiple separate matrixes to be carried in the same dataset. This supports the "myriad" style interface that allows the same operations to be performed on many different sets of data at once. Note that these matrixes do not have an explicit size. They are sized implicitly, based on the maximum row and column presented in the data. A matrix can be converted to an explicit dense form (see matrix_t) by using the utility module MakeR8Set. This module should only be used for known small matrixes (< 1M cells) or for partitions of a larger matrix. The Converted module provides utility functions to convert to and from a set of partitions (See Layout_parts).

FIELD **wi_id** Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD **x** 1-based row position within the matrix

FIELD **y** 1-based column position within the matrix

FIELD **v** Real value for the cell

SEE matrix_t

SEE Std/PBblas/MakeR8Set.ecl

SEE Std/PBblas/Converted.ecl WARNING: Used as C++ attribute. Do not change without corresponding changes to MakeR8Set.

RECORD Layout_Norm

Types \

	Layout_Norm

Layout for Norm results.

FIELD **wi_id** Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD **v** Real value for the norm

PBblas/

Vector2Diag

Go Up

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types | PBblas.internal.Types | PBblas.Constants |

DESCRIPTIONS

FUNCTION Vector2Diag

DATASET(Layout_Cell)	**Vector2Diag**
(DATASET(Layout_Cell) X)	

Convert a vector into a diagonal matrix. The typical notation is D = diag(V). The input X must be a 1 x N column vector or an N x 1 row vector. The resulting matrix, in either case will be N x N, with zero everywhere except the diagonal.

PARAMETER **X** A row or column vector (i.e. N x 1 or 1 x N) in Layout_Cell format

RETURN An N x N matrix in Layout_Cell format

SEE PBblas/Types.Layout_cell
