InterfaceConsiderations

Vector

```
/**
 * Basic vector interface.
 */
public interface Vector {

  /**
   * Size of the vector
   *
   * @return size of the vector
   */
  public int size();

  /**
   * Gets the value of index
   *
   * @param index
   * @return v(index)
   */
  public double get(int index);

  /**
   * Sets the value of index
   *
   * @param index
   * @param value
   */
  public void set(int index, double value);

  /**
   * Sets the vector
   *
   * @param v
   * @return x = v
   */
  public Vector set(Vector v);

  /**
   * x = alpha * v
   *
   * @param alpha
   * @param v
   * @return x = alpha * v
   */
  public Vector set(double alpha, Vector v);

  /**
   * Adds the value to v(index)
   *
   * @param index
   * @param value
   */
  public void add(int index, double value);

  /**
   * x = alpha*v + x
   *
   * @param alpha
   * @param v
   * @return x = alpha*v + x
   */
  public Vector add(double alpha, Vector v);

  /**
   * x = v + x
```

```
   *
   * @param v
   * @return x = v + x
   */
  public Vector add(Vector v);

  /**
   * x dot v
   *
   * @param v
   * @return x dot v
   */
  public double dot(Vector v);

  /**
   * v = alpha*v
   *
   * @param alpha
   * @return v = alpha*v
   */
  public Vector scale(double alpha);

  /**
   * Returns a sub-vector.
   *
   * @param i0 the index of the first element
   * @param i1 the index of the last element
   * @return v[i0:i1]
   */
  public Vector subVector( int i0, int i1 );

  /**
   * Computes the given norm of the vector
   *
   * @param type
   * @return norm of the vector
   */
  public double norm(Norm type);

  /**
   * Supported vector-norms.
   */
  enum Norm {

    /** Sum of the absolute values of the entries */
    One,

    /** The root of sum of squares */
    Two,

    /** The robust norm of the vector */
    TwoRobust,

    /** Largest entry in absolute value */
    Infinity
  }

  /**
   * Returns an iterator
   *
   * @return iterator
   */
  public Iterator<Writable> iterator();

  /**
   * Returns the {@link org.apache.hadoop.io.MapWritable}
   *
   * @return the entries of vector
   */
  public MapWritable getEntries();
}
```

Matrix

```
/**
 * Basic matrix interface.
 */
public interface Matrix {

  /**
   * Gets the double value of (i, j)
   *
   * @param i ith row of the matrix
   * @param j jth column of the matrix
   * @return the value of entry
   * @throws IOException
   */
  public double get(int i, int j) throws IOException;

  /**
   * Gets the vector of row
   *
   * @param i the row index of the matrix
   * @return the vector of row
   * @throws IOException
   */
  public Vector getRow(int i) throws IOException;

  /**
   * Gets the vector of column
   *
   * @param j the column index of the matrix
   * @return the vector of column
   * @throws IOException
   */
  public Vector getColumn(int j) throws IOException;

  /**
   * Get the number of row of the matrix from the meta-data column
   *
   * @return a number of rows of the matrix
   * @throws IOException
   */
  public int getRows() throws IOException;

  /**
   * Get the number of column of the matrix from the meta-data column
   *
   * @return a number of columns of the matrix
   * @throws IOException
   */
  public int getColumns() throws IOException;

  /**
   * Gets the label of the row
   *
   * @throws IOException
   */
  public String getRowLabel(int i) throws IOException;

  /**
   * Gets the label of the column
   *
   * @throws IOException
   */
  public String getColumnLabel(int j) throws IOException;

  /**
   * Return the matrix path.
   * (in hbase, path is the tablename. in filesystem, path may be a file path.)
```

```
     *
     * @return the name of the matrix
     */
    public String getPath();

    /**
     * Sets the label of the row
     *
     * @param i
     * @param name
     * @throws IOException
     */
    public void setRowLabel(int i, String name) throws IOException;

    /**
     * Sets the label of the column
     *
     * @param j
     * @param name
     * @throws IOException
     */
    public void setColumnLabel(int j, String name) throws IOException;

    /**
     * Sets the double value of (i, j)
     *
     * @param i ith row of the matrix
     * @param j jth column of the matrix
     * @param value the value of entry
     * @throws IOException
     */
    public void set(int i, int j, double value) throws IOException;

    /**
     * A=alpha*B
     *
     * @param alpha
     * @param B
     * @return A
     * @throws IOException
     */
    public Matrix set(double alpha, Matrix B) throws IOException;

    /**
     * A=B
     *
     * @param B
     * @return A
     * @throws IOException
     */
    public Matrix set(Matrix B) throws IOException;

    /**
     * Set the row of a matrix to a given vector
     *
     * @param row
     * @param vector
     * @throws IOException
     */
    public void setRow(int row, Vector vector) throws IOException;

    /**
     * Set the column of a matrix to a given vector
     *
     * @param column
     * @param vector
     * @throws IOException
     */
    public void setColumn(int column, Vector vector) throws IOException;

    /**
```

```
 * Sets the dimension of matrix
 *
 * @param rows the number of rows
 * @param columns the number of columns
 * @throws IOException
 */
public void setDimension(int rows, int columns) throws IOException;

/**
 * A(i, j) += value
 *
 * @param i
 * @param j
 * @param value
 * @throws IOException
 */
public void add(int i, int j, double value) throws IOException;

/**
 * A = B + A
 *
 * @param B
 * @return A
 * @throws IOException
 */
public Matrix add(Matrix B) throws IOException;

/**
 * A = alpha*B + A
 *
 * @param alpha
 * @param B
 * @return A
 * @throws IOException
 */
public Matrix add(double alpha, Matrix B) throws IOException;

/**
 * C = A*B
 *
 * @param B
 * @return C
 * @throws IOException
 */
public Matrix mult(Matrix B) throws IOException;

/**
 * C = alpha*A*B + C
 *
 * @param alpha
 * @param B
 * @param C
 * @return C
 * @throws IOException
 */
public Matrix multAdd(double alpha, Matrix B, Matrix C) throws IOException;

/**
 * Computes the given norm of the matrix
 *
 * @param type
 * @return norm of the matrix
 * @throws IOException
 */
public double norm(Norm type) throws IOException;

/**
 * Supported matrix-norms.
 */
enum Norm {
  /** Maximum absolute row sum */
```

```
    One,

    /** The root of sum of the sum of squares */
    Frobenius,

    /** Largest entry in absolute value */
    Infinity,

    /** Largest entry in absolute value.  */
    Maxvalue
  }

  /**
   * Save to a table or file
   *
   * @param path
   * @return true if saved
   * @throws IOException
   */
  public boolean save(String path) throws IOException;

  /**
   * Returns the matrix type
   *
   * @return the matrix type
   */
  public String getType();

  /**
   * Returns the sub matrix formed by selecting certain rows and
   * columns from a bigger matrix. The sub matrix is a in-memory operation only.
   *
   * @param i0 the start index of row
   * @param i1 the end index of row
   * @param j0 the start index of column
   * @param j1 the end index of column
   * @return the sub matrix of matrix
   * @throws IOException
   */
  public SubMatrix subMatrix(int i0, int i1, int j0, int j1) throws IOException;

  /**
   * Close current matrix.
   *
   * @throws Exception
   */
  public void close() throws IOException;
}
```