IST-2004-027173

Eu*Resist*

Integration of viral genomics with clinical data
to predict response to anti-HIV treatment

Instrument: STREP
Thematic Priority 2: "Information Society Technologies"

D.3.1 - HIV resistance predictor engines specifications

Due date of deliverable: 15/01/07
Actual submission date: 17/01/07

Start date of project: Jan. 1, 2006 Duration: 30 months

Organisation name of lead contractor for this deliverable: IBM
Authors: Michal Rosen-Zvi and Ehud Aharoni
Main contributions: Hani Neuvirth, Andre Altmann, Fülöp Bazsó, Mattia Prosperi, Roberto D'Autilia and Yardena Peres
Revision [Final]

Table of Contents

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**x**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

List of Figures

Executive summary

This document provides the specification of the prediction engines that play a central role in the EuResist project and will be developed in the first half of 2007. Four groups including RMKI, IBM, MPI and UniSiena will develop different engines based on different algorithms.

The goal of this document is twofold. First, we describe the algorithmic design of each of the four engines. These designs are the result of research performed in each of the four groups during 2006. Second, we specify numerous details for various aspects of the design and development methodology. These details are aimed at directing the four separate development teams toward producing four interoperable and comparable engines that will fit within the overall architecture of the EuResist project.

To achieve the second goal, this document includes the following information:

- Overview of the overall system architecture and the role of the engines.
- Technical aspects of the engines' development, i.e., the technology used to implement the engines' algorithms and to communicate with them.
- Definitions regarding the engines' training and assessment.
- Detailed definition of the engines' interface.

1. Introduction

The EuResist project aims at developing a European integrated system for clinical management of antiretroviral drug resistance. At the heart of this effort lie four prediction engines. These prediction engines are being developed by four separate groups and will be combined together to achieve a single goal. This goal is to provide clinicians with a prediction of response to antiretroviral treatment in HIV patients.

The above fuzzy statement needs to take on a concrete form. This document is designed to provide detailed specifications of the engines. The specification includes several aspects of engine design and development methodology.

Chapter 2 starts with an overview of the system architecture and the engines' role within it. In Chapter 3, we describe the technical aspects of the engines' development, by detailing the technology used to implement the engines' algorithms and communicate with them. Chapter 4 outlines some definitions regarding the engines' training and assessment. This will simplify the combination of the engines and the evaluation that will be performed next year. The interface to the engines is described in Chapter 5; here we specify the exact format of valid queries to the engines and the format of the corresponding responses. The appendices include information regarding the internal algorithms used within each of the developed engines.

2. Architecture overview

The overall architecture of the project is described in **Figure 1**.

A web interface will be created so users can enter their queries. These queries will be submitted to the unit termed the Engine Combination, which will forward the query to an array of prediction engines. These engines will return their response and the Engine Combination unit will combine their responses into a single response, which will be returned to the web interface.

Two offline procedures will be implemented. One combines the available data sources into a single database termed the Integrated Database. The second procedure uses this Integrated Database to train the engines. Both procedures are expected to be a combination of automatic and manual operations. The web interface will have access to the Integrated Database, allowing the user to query it; however, the prediction engines will not have access to the database during runtime.

The Engine Combination and prediction engines unit will be implemented as web services, and communication with them will be performed using XML messages.

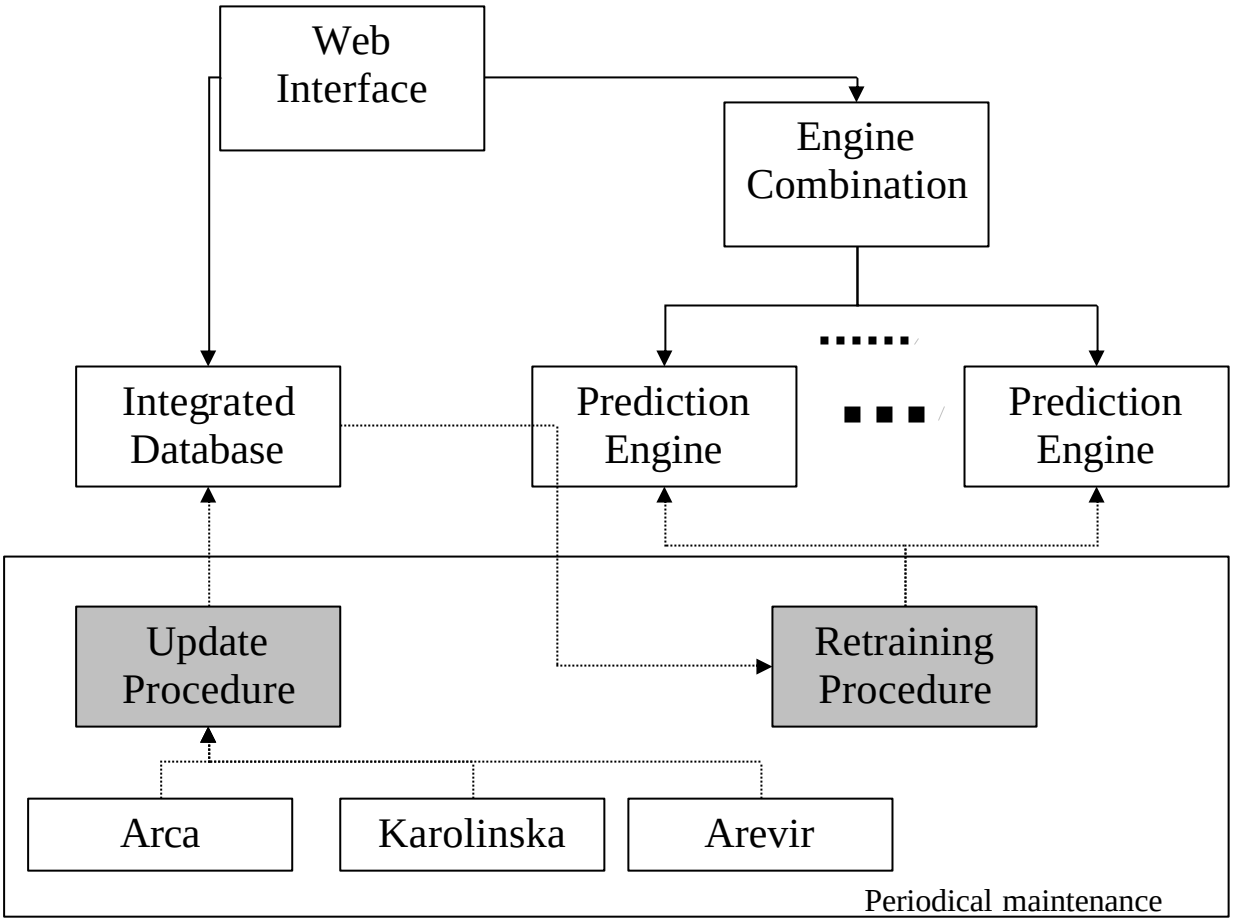

Figure 1 – Overview of the EuResist project planned architecture

3. Technical specification

3.1. Technology requirements

The engines will be implemented as a web service. These engines must comply with the WS-I standard to avoid interoperability problems (see www.ws-i.org).

Communication with the web services will be performed using the Simple Object Access Protocol (SOAP).

Each development team is free to choose its preferred platform and development tools.

The web service interface will be defined by a WSDL file. The final version of this file will be distributed by March 31, 2007.

3.2. Performance requirements

Each prediction engine must be designed to meet the following performance requirements:

- Process up to approximately 1000 queries per day.

- Process a single request within 60 seconds.
- Possible addition of a new feature that will allow the engine to process a large batch request while continuing the normal level of service to the on-line requests.

3.3. Integration tests schedule and requirements

To avoid technical difficulties when we combine and integrate the prediction engines, two integration tests are planned in advance of the actual integration.

The first integration test is will be carried out by December 31, 2006. Each development team will implement and run a web service executing some simple functionality that will be specified. IBM will test that a single request can be sent and processed by all prediction engines. This test will guarantee that the platform and tools used by all the partners allow the development of interoperable engines.

The second integration test will be carried out by March 31, 2007. By that time, the engines will be required to process a valid request in its final form and produce a valid response. The accuracy of the response will not yet be considered, just its form.

These tests are designed to overcome:

- Inability to connect to an engine because of communication problems.
- Data being distorted during transport, i.e., truncated or disordered arrays, date format problems, etc.

4. Engine training and assessment

The data used for training is described in the standard datum document (Deliverable D2.1). This chapter describes the queries that will be provided to the prediction engine in multiple possible forms. In the following sections we differentiate between a labelling scheme (Section 4.1), which is designed primarily to assess off-line engine performance, and Input Output definitions, which describe the actual on-line queries that will be submitted to the engines (Sections 4.2 and 4.3, respectively).

4.1. Labelling therapies

- The following rules determine which therapies can be used for training and how to label notes and notation for the rules:

  (1) START means the date of therapy start.

  (2) END means the date of therapy end.

  (3) A rule that specifies a time range for a measure also specifies the optimal date within this range. If several measures occur within the range, the measure closest to the optimal date is taken.

  (4) The rules for attaching sequences to therapies are considered separately for Pro (Protease) and RT (Reverse Transcriptase) sequences. Both must satisfy the rule, but not necessarily on the same date.

- Rules for treating overlapping therapies:

(1) A therapy that ends on the same day on which another therapy starts is not considered on overlap.

(2) For any other case of time overlap, all the therapies involved are discarded.

- Rules for therapy compounds:

(1) Therapies with no compounds are discarded.

(2) Therapies with any compound not appearing in Table 1 in the standard datum document (Deliverable D2.1) are discarded.

- Rules for the classic labelling:

(1) For the therapy to be labelled, it must have:

(i) Baseline Pro and RT sequences according to Rule 3 below.

(ii) Baseline Viral Load (VL) measure attached according to Rule 4 below.

(iii) Follow-up VL measure attached according to Rule 5 below.

(2) Let VL1 be the baseline VL measure and VL2 be the follow-up. The therapy is labelled 'success' if: VL2 <= 500 or (log10(VL1) - log10(VL2) >= 2). Otherwise it is labelled 'failure'.

(3) A baseline sequence taken on date SEQDAY is attached to a therapy if:

(i) START - 90 <= SEQDAY <= START + 2 and

(ii) The time range SEQDAY...START - 1 is safe according to Rule 6.

(iii) Optimal date: START.

(4) A baseline VL measure taken on date VLDAY is attached to a therapy, if VLDAY satisfies the same rules as in Rule 3 above.

(5) A follow-up VL measure taken on date VLDAY is attached to a therapy if:

(i) START + 28 <= VLDAY <= START + 84 and

(ii) VLDAY <= END + 2.

(iii) Optimal date: START + 56.

(6) A time range T1...T2 is considered safe for Rules (3), (4) if all the therapies that start within this time range are 14 days long or less.

- Rules for the alternative labelling:

(1) For the therapy to be labelled 'failure' the following must hold:

(i) Both Pro and RT sequences are attached to the therapy using Rule 4.

(2) For the therapy to be labelled 'success' the following must hold:

(i) Baseline Pro and RT sequences are attached to the therapy using Rule 5.

(ii) A low VL measure is attached to the therapy using Rule 6.

(3) If a therapy satisfies both the conditions for 'failure' and the conditions for 'success', it is split into two records. The two records have the same therapy

compounds, but each has different sequences according to Rules (1) and (2), respectively.

(4) A sequence taken on date SEQDAY is attached to a failing therapy if:

(i) START + 14 <= SEQDAY <= END + 2

(ii) Optimal date: START + 14

(5) A sequence taken on date SEQDAY is attached to a successful therapy if:

(i) START - 90 <= SEQDAY <= START + 2 and

(ii) A previous therapy exists starting on date START2 and ending on date END2 and START2 <= SEQDAY <= END2+2 and the time range END2...START-1 is safe according to rule (7).

(iii) Optimal date: START.

(6) A viral load measure taken on date VLDAY is attached to a therapy if:

(i) The viral load measure is 500 or less and

(ii) START + 7 <= VLDAY <= END + 2

(iii) Optimal date: START + 7

(7) A time range T1...T2 is considered safe for rule (5) if all the therapies that start within this time range are 14 days long or less.

4.2. Evaluating engine success

This section provides several assessment methods that will be used to evaluate the prediction engines' performance. The objective is to allow the engines to compare results 'on the fly'. Note that the evaluation of engine success is part of a separate work package scheduled for a later stage when the engines are already deployed. The final evaluation will likely include many more methods and might be somewhat different from the following methods.

- Binary prediction:
  - Misclassification rate: Number of wrongly predicted outcomes of therapies, out of the total number of therapies.
  - Receiver Operating Characteristic (ROC) and the Area Under the Curve (AUC).
- Continuous predictions
  - Mean squared error: The expected value of the square of the amount by which the estimator differs from the quantity to be estimated.

5. Input and output specification

This chapter describes in detail the interface of the prediction engines. Both the input queries and output responses of the engines are XML documents. In the following sections, we describe the structure of these documents.

Section 5.1 provides an overview of the query and how the engine should respond to it.

Section 5.2 describes the structure of the input query.

Section 5.3 describes the structure of the output response.

Section 5.4 specifies how sequences are coded within the input query.

Section 5.5 provides some examples.

5.1. Query and response overview

The input query has six major sections, of which only the first is mandatory:

- Sequences – Pro and RT sequences to be used for prediction
- Patient data – general details about the patient (age, gender, etc.)
- Clinical measures – viral load and CD4
- Historical data – details about past treatments
- Suggested treatments – a list of compound combinations the user wishes the engine to assess
- Restricted compounds – a list of compounds the user wishes the engine to avoid using

The engines mandatory role is to rank possible treatments from better to worse. If neither "suggested treatments" nor "restricted compounds" sections are specified, the engine is free to consider every possible treatment, and the response should include the top 20 treatments found.

If the "restricted compounds" section is specified, the engine should include in its response the top 20 treatments that do not use these compounds.

If the "suggested treatments" section is specified, the engine should rank just these treatments. Supplying both "restricted compounds" and "suggested treatments" in the same query is illegal.

A prediction engine may support the feature of predicting changes in clinical measures for a particular treatment. In order to get this prediction, the user must supply the relevant initial clinical measure in the "clinical measures" section. These predictions will be added to each of the ranked treatments included in the engine's response. Note that this is an optional feature. Each engine may support some subset of the possible predictions defined in the standard datum document (and below in Section 5.3).

This flexible design allows the user several ways to query an engine:

- Ask the engine for recommended treatments, with or without prediction of future viral load and CD4.
- Ask the engine to rank several user-defined treatments, with or without prediction of future viral load and CD4.

5.2. Input query

This section describes the structure of the input query XML. Fields (or elements) of the document are shown in italics, followed by a description of their content. Some fields are marked as "optional". Fields that can appear more than once are marked with "zero or more instances".

The meaning of the fields is only briefly explained. The names are the same as in the "standard datum" document, so it can be used as a reference.

- *sequences*

  (1) *pro* – see specification in Section 5.4.

  (2) *rt* – see specification in Section 5.4.

- *patientData* (optional) – containing various general details regarding the patient

  (1) *age* (optional) – integer

  (2) *gender* (optional) – integer value from the Genders table of the integrated database

  (3) *ethnic* (optional) – integer value from the EthnicGroups table of the integrated database.

  (4) *riskGroup* (optional) – integer value from the RiskGroups table of the integrated database.

- *clinicalMeasures* (optional) – current clinical measures of the patient

  (1) *viralLoad* – double

  (2) *cd4* – double

  (3) *cd4Precentage* - double

- historicalData (optional)

  (1) *suboptimalTreatment* (optional) – Boolean.

  (2) *numberOfPastTreatmentLines* (optional) – integer

  (3) *reasonForSwitch* (optional) – integer value from the StopTherapyCauses in the integrated database.

  (4) *steadyStateViralLoad* – double

  (5) *pastTreatmentsCategorical* (optional) – contains a list of Boolean fields indicating for each drug whether it has been used in the past. (See standard datum document for details.) Each field is named *compoundNNN* where NNN is an integer from the Compounds table of the Integrated Database.

  (6) *pastTreatmentsWeighted* (optional) – same as *pastTreatmentsCategorical* except with real values instead of Booleans.

  (7) *pastSequences* (zero or more instances)

    (i) *pro (*optional) - see specification in Section 5.4.

    (ii) *rt (optional)* - see specification in Section 5.4.

  (8) *aidsDefniningEvents* (optional) – Boolean

- *suggestedTreatments* (optional) – a list of compound combinations the user wants the engine to assess

  (1) *treatment* (zero or more instances)

    (i) *compound* (zero or more instances) – integer from the Compounds table of the integrated database

- *restrictedCompounds* (optional) – a list of compounds to avoid

  (1) *compound* (zero or more instances) – integer from the Compounds table of the integrated database

5.3. Output response

This section describes the structure of the output response XML. Fields (or elements) of the document are shown in italics, followed by a description of their content. Some fields are marked as "optional". Fields that can appear more than once are marked with "zero or more instances".

- *recommendedTreatment* (zero or more instances)

  (1) compounds

    (i) *compound* (zero or more instances) – integer from the Compounds table of the integrated database

  (2) *rank* – an integer value specifying the position of this treatment in the "best-first" order. This value can be deduced from the order of the treatments in the response but is added for robustness.

  (3) *score* (optional) – a real value in the range [0...1] specifying how well this treatment is expected to work. This is an optional value designed to give some absolute measure in addition to the relative rank.

  (4) *predictedClinicalMeasures* (optional) – a prediction of changes in clinical measures

    (i) *undetectableViralLoadProbability* – probability of achieving undetectable viral load after 3 months.

      (i) *value* – double [0...1]

      (ii) *confidence* – double [0...1]

    (ii) *predictedViralLoad3* – predicted viral load after 3 months

      (i) *value* – double

      (ii) *confidence* – double [0...1]

    (iii) *predictedViralLoad6* – predicted viral load after 6 months

      (i) *value* – double

      (ii) *confidence* – double [0...1]

    (iv) *predictedCd4* – predicted CD4 after 6 months

      (i) *value* – double

(ii) *confidence* – double [0...1]

(v) *cd4IncreaseProbability* – probability of achieving >50% increase in CD4 counts at 6 months

(i) *value* – double [0...1]

(ii) *confidence* – double [0...1]

5.4. Sequence representation

The sequences in the input query are represented by the following structure:

- *startPos* – integer specifying from which position the sequence starts
- *endPos* - integer specifying in which position the sequence ends
- *date* – when the sequence was extracted
- *mutation* (zero or more instances)

  (1) *position* – integer

  (2) *aa* – string representing list of possible amino acids using capital letters following the usual notation

- wildType (optional) – string specifying the wild type the mutations refer to. The string length must match the actual sequence length (i.e., 99 letters for Protease and 440 for reverse transcriptase). If it is not specified, the "Consensus-B" wild type will be assumed.

We chose this representation for easier reading and improved accuracy. Note that the web interface will have to make it clear to the user they must supply mutations in reference to the "Consensus-B" wild type or supply the wild type to which they refer.

Also note that *startPos* and *endPos* must be at least 10...95 for Protease and 41...219 for Reverse Transcriptase, as specified in the standard datum.

Currently, insertions and deletions cannot be specified. In the future, new fields will be added to support them.

Figure 2 shows an example of how a Protease sequence would appear in the input query.

```
<pro>
        <wildType>QITLWQRPI...</wildType>
         <startPos>5</startPos> <endPos>95</endPos>
        <mutation> // position 14 has mutated to amino acid "A"
                <position>14</position>
                <aa>A</aa>
        </mutation>
        <mutation> // position 20 has mutated to either "C" or "Y"
                <position>20</position>
                <aa>CY</aa>
        </mutation>
</pro>
```

Figure 2 – An example of a definition of a protease sequence in the input query

5.5. Examples

Figure 3 and **Figure 4** show examples of queries and responses.

```
<query>
   <patientData>
      <age>34</age>
      <gender>0</gender>
   </patientData>
   <sequences>
      <pro> <mutation>…</mutation> … </pro>
      <rt> <mutation> … </mutation> … </rt>
   </sequences>
    <restrictedCompounds>
      <compound>3</compound>
      <compound>5</compound>
    </restrictedCompounds>
</query>

<response>
   <recommendedTreatment>
      <compounds>
         <compound>4</compound>
         <compound>6</compound>
         <compound>7</compound>
      </compounds>
      <rank>1</rank>
   </recommendedTreatment>
   …
   <recommendedTreatment>
      <compounds>
         <compound>4</compound>
         <compound>6</compound>
         <compound>8</compound>
      </compounds>
      <rank>20</rank>
   </recommendedTreatment>
</response>
```

Figure 3 - Example of query and response. The query contains no suggested treatments. The engine responds with a list of the best 20 treatments that avoid the restricted compounds. No clinical measures are supplied so none are predicted.

```
<query>
   <sequences>
      <pro> <mutation>…</mutation> … </pro>
      <rt> <mutation> … </mutation> … </rt>
   </sequences>
   <clinicalMeasures>
      <viralLoad>1000</viralLoad>
      <cd4>150</cd4>
   </clinicalMeasures>
    <suggestedTreatments>
      <treatment>
         <compound>1</compound><compound>2</compound>
         <compound>3</compound>
      </treatment>
      <treatment>
         <compound>1</compound><compound>2</compound>
         <compound>4</compound>
      </treatment>
    </suggestedTreatments>
</query>

<response>
   <recommendedTreatment>
      <compounds>
         <compound>1</compound>
         <compound>2</compound>
         <compound>4</compound>
      </compounds>
      <rank>1</rank>
      <predictedClinicalMeasures>
         <predictedViralLoad3>
              <value>500</value>
               <confidence>0.8</condfidence>
         </predictedViralLoad3>
      </predictedClinicalMeasures>
   </recommendedTreatment>
   <recommendedTreatment>
      <compounds>
         <compound>1</compound>
         <compound>2</compound>
         <compound>3</compound>
      </compounds>
      <rank>2</rank>
      <predictedClinicalMeasures>
         <predictedViralLoad3>
              <value>900</value>
               <confidence>0.8</condfidence>
         </predictedViralLoad3>
      </predictedClinicalMeasures>
   </recommendedTreatment>
</response>
```

Figure 4 - Example of query and response. The query contains a list of two suggested treatments. The response ranks them from best to worse. Clinical measures are included in the query, so some clinical measure predictions are included in the response. Note that an engine may not support all possible predictions.

Appendix: Engines core technology

- **IBM**

Introduction

The IBM engine is based on a combined generative and discriminative approach. The engine is built to generate a response to input query as defined in Chapter 5 of this document. While training, the engine has access to rich information sources provided primarily from the Integrated Database (IB), along with access to in-vitro databases that are currently not part of the IB, the Geno2Pheno data of Arevir, and the Stanford database. (See Rhee et al.)

In general, there are two types of engines that are interconnected: the binary prediction engine and the continuous response engine. The binary prediction engine is trained on two distinct labeling schemes. For each scheme, given a new input (sequence and therapy), the engine generates a label as well as a continuous confidence value. The continuous engine generates probabilities for any viral load level at 3 and 6 months after the start of treatment. These probabilities are used to derive the various requirements in Section 5.3

Our testing approach involves training simple machine learning engines to serve as a baseline predictors. These include engines such as Naive Bayes and Linear SVM (binary prediction), and a Linear regressor (continuous prediction) with and without PCA.

We employ 10-fold cross validation testing for model selection, and the final model is tested against randomly chosen held-out data.

Building blocks of the prediction engine

Sequence representation

Sequences are represented by vectors of size 20*99 (Protease) and 20*440 (Reverse Transcriptase). The 20 amino acids are mapped into numbers from 1 to 20, if the ith Amino Acid appears at the mth position, then this entry Xim=1 and Xjm for all $j \neq i$ is zero. If there is more than one possible mutation, say the ith Amino Acid and the jth Amino Acid appear at the mth position, then Xim=1/2 and Xjm=1/2 and Xkm for all $k \neq i$ or j is zero. For positions at which the amino acid is not known, the histogram of the known amino acid is assumed. In practice, only a small fraction of the amino acids ever appear in a particular position and thus the vectors representing the amino acid are significantly reduced.

In-Vitro data

We employ SVR learners that have been used in previous studies to serve as a preprocessing feature extraction stage, with a Bayesian network built on top of them to impose the biologically relevant structure. The Bayesian network can be interpreted as a generating process of the sequence's features. The equivalence to the biological system is problematic, as the biologically-relevant order of things starts from a new mutated sequence showing a certain level of resistance. Nevertheless, this model is in some sense equivalent to the evolutionary selection pressure applied on the sequences by the drug. Specifically, at low resolutions the selection process distinguishes between resistant and susceptible sequences, each having a different probability of overcoming the drug, and becoming the dominant sequence. The following selection is at a higher resolution, and refines the resistance (and the probability of the sequence to dominate) according to

specific features of the sequence. We use the resultant prediction of fold-resistance obtained from SVR-net described above to generate the in-vivo prediction.

Dimensionality reduction

In the binary setting classic labeling approach, many of the therapies cannot be labelled due to missing data. However, the unlabeled data can be, and is, used to improve a dimensionality reduction scheme in which original sequences are mapped onto a smaller dimension. The approach we use rests on two important constructions:

(i) An information theoretic dimensionality reduction procedure and

(ii) A probabilistic kernel.

Both are linked through the assumption that the relevant metric for the instances provided is the Jensen Shannon (JS) metric.

Tailoring kernels

We learn binary prediction with kernel SVM - linear, Radial Basis Functions (RBF) and JS. The sequences are mandatory for some of the features and are optional for others, such as risk factor. We learn separate kernels per each of the feature groups and learn from the data how to tailor the kernels to achieve the best prediction results.

References:

Soo-Yon Rhee, Matthew J Gonzales, Rami Kantor, Bradley J Betts, Jaideep Ravela, and Robert W Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. Nucleic Acids Res, 31(1):298–303, Jan 2003

- **Max Planck**

Prediction of response to treatment (classification):

Input:

The prediction engine requires aligned sequences of Reverse Transcriptase and Protease. Sequences can be in nucleotide or amino acid form. This input is used to derive a number of features that are applied in the prediction.

Feature generation:

In the first step, the complexity of the genotypic information is reduced by using one indicator variable for each resistance-associated mutation [Johnson] instead of the complete sequence. Encoding the current therapy is done in a straightforward fashion by using one indicator variable for every drug under consideration. In the second step, more elaborate features are derived and extend this *indicator encoding*.

In the *activity representation*, the indicator vector is extended by the estimated activity of the drug cocktail against the virus population. The notion of single-drug activity is based on the distribution of predicted fold-changes and reflects the probability that the sample is phenotypically susceptible given the predicted fold-change. Evolutionary information is included by introducing mutations into the considered genotype and searching the sequence space by following mutants of least activity. The activities of the worst-case mutants at each level of search depth are then combined into a single activity score. We refer to [Beerenwinkel] for a detailed description of this procedure.

The *genetic barrier representation* also adds both phenotypic and evolutionary information to the indicator representation, and it can be regarded as an advancement

over the activity score. More precisely, we used mutagenetic trees, a family of probabilistic graphical models, to estimate the order and rate of occurrence of resistance mutations. Using the mtreemix software, for each drug, a mixture model of mutagenetic trees was learned from sequences derived under regimens comprising that drug. Based on these evolutionary models, we defined the genetic barrier as the probability that the virus will not escape from drug pressure by developing further mutations . Here, viral escape is approximated by exceeding a predefined level of phenotypic resistance. These levels are defined by the following cut-offs for the fold change in susceptibility: zidovudine 30.0; zalcitabine 2.2; didanosine 2.4; stavudine 2.0; lamivudine 15.4; abacavir 3.4; tenofovir DF 2.1; nevirapine 9.0; delavirdine 9.7; efavirenz 7.0; saquinavir 4.5; indinavir 4.6; ritonavir 2.6; nelfinavir 5.8; amprenavir 12.0; lopinavir 10.0; atazanavir 4.2. Unlike the sequence-space search for low-activity mutants, the genetic barrier accounts for the fact that not all mutations are equally likely to occur. This is also an advantage over simply counting resistance mutations, a frequently employed approximation to the genetic barrier.

Finally, the genetic *progression score (GPS)* involves only evolutionary information that is extracted from the mutagenetic tree models. The GPS of a genotype is defined as the expected waiting time for the mutational pattern to occur. Thus, the GPS also accounts for different probabilities of different mutations, but it does not include any phenotypic information. We emphasize that the GPS is not intended to estimate waiting times on the real time scale. Rather it provides a dimension-free measure of genetic progression that allows for comparing mutational patterns.

Statistical learning methods:

The different features define the input to several different machine learning techniques that are applied to predict treatment response. We selected several standard classification methods, including linear discriminant analysis (LDA), least-squares regression, linear support vector machines (SVMs), decision trees (namely the program C4.5), and logistic regression. As a more recent method, we also included logistic model trees (LMT), which combine decision trees with logistic regression in the leaves of the tree.

Output:

Output generated by the machine learning methods will be i) the predicted class label and ii) a confidence value.

Prediction of response to treatment (regression):

Input:

The prediction engine requires aligned (baseline) sequences for Reverse Transcriptase and Protease.

Sequences can be in nucleotide or amino acid form. In addition to the sequences a baseline viral load is required, and CD4 counts can be given as an optional feature.

Feature generation:

Features generated for the classification task are also applied in the regression task. The encodings described above are enhanced by the baseline viral load (required), the baseline CD4 counts (optional), and additional optional features as described in the standard datum document.

Statistical learning methods:

The different features define the input to several different machine learning techniques that are applied to predict the change in viral load (and CD4 counts) as response to a treatment. We selected several standard regression methods, including linear regression, support vector machine regression, and random forest regression.

Output:

The output is the level and change in viral load at 3 and 6 months, as computed by the regression methods. Additionally, the probability of achieving an undetectable viral load at 3 months is computed, along with the probability of achieving a >1 log decrease in viral load at 3 months.

If baseline CD4 counts are provided, the statistical learning methods are applied to compute the level and change in CD4 counts as well as the probability of achieving a >50% increase in CD4 counts at 6 months.

References:

Johnson VA, *et al.* Update of the drug resistance mutations in HIV-1: fall. Top HIV Med 2005; 13:125-131.

Beerenwinkel N, et. al. Methods for optimizing antiviral combination therapies. Bioinformatics 2003; 19 Suppl 1:i16-25.

- **RMKI**

Description of RMKI's Prediction Engine

Basic Description

The Prediction Engine operates on data provided by the Integrated Database of the EUResist consortium. It is designed for data extraction, data mining and the prediction of optimal drug treatment given the data of an AIDS patient in need of new therapy. RMKI's Prediction Engine operates under the assumption that the viral genome is well determined up to the known uncertainty and that the sequences are well-aligned in the Integrated Database.

Operation of the Prediction Engine developed by RMKI is based on results from combinatorics, graph and information theory. Viral genotypes are grouped based on therapy efficiencies using the data available from the Integrated Database. The grouping is performed in a way that minimizes the functionals depending on the data configuration. The RMKI's Prediction Engine uses functionals that quantify data roughness and data entropy. Depending on the quality of the available data, these functionals may be minimized separately, or simultaneously.

Detailed description

Data available from the Integrated Database is filtered and organised as a multidimensional array with periodic boundary conditions. The array is indexed with applied therapies, information derivable from viral genetic code and cumulative genotype, if available. Due to the high viral variability the filtered viral data are preprocessed and aggregated in a minimal way in order to decrease sparseness of the array. Array fields are filled with continuous indicators of therapy efficacies, which are derived from the viral concentration and concentration of $CD4^+$ cells, when available.

Smoothness and entropy functionals depending on the configuration of the array are introduced. Knowledge of the extrema of the functionals lead to coarsening partitions of the array indexing sets, while minimising the overall information loss. The method relies on the Szemerédi's regularity lemma in several contexts, which gives a guideline about simultaneous coarsening partitions and information loss minimisation. The query is processed as coloured hypergraph partition fuzzy membership determination.

Data Formats

RMKI's Prediction Engine uses data formats previously accepted by the Consortium. The data definitions used adhere to the standard datum document of the Consortium, and the query format adheres to the web services description language definition proposed by IBM, 26/10/06.

Input data

The RMKI's Prediction Engine operates on the most characteristic data of an individual patient, when there is a need to change AIDS therapy. The minimal input data includes viral genotype (sampled within 90 days). Data that may improve the prediction includes: viral load (count per unit volume, sampled within 90 days), concentration of $CD4^+$ cells, the dates of these measurements, cumulative genotype, and therapy history.

Output data

The suggested therapies are ranked according to their efficacies, where therapy is a combination of drugs used in AIDS treatment optimized for the individual patient.

Communication protocol

SOAP is used as the communication protocol. (SOAP is the successor of XML- RPC, a W3C Recommendation on June 24, 2003.) The WSDL module of the PHP5 language is used. The WSDL module processes SOAP requests, and passes serialized PHP objects to the Prediction Engine and back to the server. We have a system capable of processing SOAP requests to the full extent the PHP WSDL library allows.

Basic requirements and prediction accuracy

Successful prediction requires the actual viral genotype and viral load as minimal data. However, if more data is provided, this may lead to a prediction of higher accuracy. If the cumulative viral genotype is not provided, the information regarding the time elapsed since HIV infection was diagnosed may increase the efficacy of the suggested drug combination, and the time elapsed since the onset of the first therapy.

References:

J. Komlós, M. Simonovits, Szemerédi's regularity lemma and its applications in graph theory, *Combinatorics, Paul Erdos is eighty*, **2** (Keszthely, 1993), pp. 295-352, *Bolyai Soc. Math. Stud.*, 2, János Bolyai Math. Soc., Budapest, 1996.

J. N. Mordeson, P. S. **Nair**, Fuzzy Graphs and Fuzzy Hypergraphs, Studies in Fuzziness and Soft Computing, Vol. 46, Springer, Berlin-Heidelberg, 2000.

T. Nepusz, F. Bazsó, Likelihood based clustering of directed graphs, submitted

E. Szemerédi, Regular partitions of graphs, in *"Proc. Colloque Inter. CNRS"* (J.-C. Bermond, J.-C. Fournier, M. Las Vergnas, D. Sotteau, eds.) (1978), pp. 399 401.

T. Tao, Szemerédi's regularity lemma revisited, *Contributions to Discrete Mathematics*, (2006), **1** pp. 8-28.

- **UniSiena, Roma TRE, Informa**

Prediction of response to treatment (classification & regression):

Two approaches are under study: one based on Statistical learning methods, the other on Stochastic models.

Statistical Learning Methods:

The main effort is to develop an instance based learner, using k-Nearest Neighbour algorithm and local smoothing kernel methods [2]. In order to avoid the curse of dimensionality problem, a wide range of feature selection techniques will be explored, using either a Filter or a Wrapper [3] approach and statistical comparisons of cross validations. Another problem is the space density. The amount of the data projected onto the reduced feature set must be sufficiently dense and redundant to ensure good local approximation.

The local approach will be compared with feature selection enhanced eager methods (like SVMs, Neural Networks, Decision Trees, Rules, Fuzzy CANFIS): if the local approach is judged not providing sufficient performances, it will be replaced by the best model trained and validated.

Input:

Input to the engine is a standard datum instantiation. The engine will work starting from the simple input of a pol sequence, from which mutations are intended to be extracted (or more easily, a list of extracted mutations comparing viral sequence and wild type consensus B). Additional attributes that are included can significantly enhance prediction performance, but are not mandatory (for example drug history or baseline viral load) as requested in the standard datum definition. For the regression task, the viral load and CD4+ baseline counts will probably turn out to be mandatory in order to ensure sufficient performance.

Feature generation:

All the mutated positions are considered. From these a subset of literature-relevant position is also considered (from IAS/USA list [1]). Mutations will be coded as real values in [0,1] depending on the percentage in the total population in a codon. As a surrogate of the MPI's Genetic Barrier indicator, the number of accumulated resistant mutations will be calculated for each drug. Phenotype log_FoldChange values will be estimated through Multiple Linear Regression (first order) for each drug. Drug History will be coded in three different vectors: i) time of total exposure for each drug; ii) time passed from last exposure for each drug; iii) exponential decreasing function proposed by IBM (exp(-t1/t2)) for each drug. Therapies will be coded as binary vectors, indicating the presence of a certain drug in the therapy combination.

Risk Group, Subtype and similar nominal attributes will be considered, as for real valued attributes like Viral Load and CD4+.

Missing values will be handled, either using "Unknown" flags for nominal attributes, or mean-median values for numerical ones.

Clusters of data or Principal Component Analysis will be also taken into account, as candidate derived feature to improve prediction performances, calculated as the other indicators internally to the engine.

Preliminary Results:

A minimal (i.e. only baseline mutations extracted from the baseline pol sequences) Standard Datum view on the ARCA data base has been extracted and labeled with the 3-months undetectable viral load cutoff at 2.7 Logs (503 instances, balanced with success/failure).

The best local approach, made on adjusted comparisons of 100 independent runs of 10-fold cross validation on the misclassification error [4], was obtained with a wrapped forward selection on mutations. The set of mutations selected was resembling the well-known resistance associated list [1]. Prediction performances assessed to 68% correct (st.dev 0.8) and this was one of the best models compared to the naïve machine learning techniques (with little or no feature selection, also compared to boosted trees and similar learners).

However, differently from most feature selection enhanced methods, linear SVMs with Recursive Feature Elimination [5] provided up to 75% correct (st.dev 5.4). Using EM clustering on attribute space and then applying to the clustered space various techniques (k-NN, Rules, Trees, Neural Networks, SVM…) performances not satisfactory for most of learners. Mere PCA yielded similar mediocre results.

Even though not high, these results prove that mutations (along with therapies) are a good base to start with, but it seems there is an intrinsic limit which must be overcome using other information. It's important to point out here that the local and the SVM model were statistically better than a naïve model considering only therapy combination, i.e. genotype really has a crucial role in the resistance modeling.

Next step is to train learners on the entire integrated data base and to verify if other features (drug history…) and derived features (in-vitro phenotype…) improve performances. The hope is also that the instance based method gets better augmenting the DB size. The immediate goal is to achieve the 85% presented by MPI in [6].

Output:

Classification: the class label (success/failure as defined in the Standard Datum definition) and the confidence value.

Regression: n-weeks Viral Load Log (and presumably CD4+ counts), with either n fixed or variable and confidence values.

For both problems, a rank of therapy combinations will be calculated.

Stochastic modeling:

Together with the development of the statistical prediction methods, the time evolution for the HIV virus is also modeled. The goal is to predict the mutation probability in the virus population under the pressure of the external fields (drugs). The approach originated with the classical population dynamics works [7] [8] [9], and received new impulse from the recent development of the statistical mechanics of complex systems and its application to different research fields [10] [11] [12].

The stochastic model also gives a rigorous formal basis to the statistics and a biological interpretation of the quantities we analyze (for example the cross-entropy which makes use of theoretical and measured probability distribution). However the stochastic model is not realistic since does not take explicitly in account all the possible biological variables.

The standard datum represents in this framework the minimum number of relevant variables giving a good approximation of the phenomenon and rigorous bounds for the behavior of the real system (85%).

Taking the set S of all the possible values the standard datum, the subset G ? S gives the clinical or microscopic state of the patient. The G space is a coarse grain of the real

clinical state for the patient and is composed by measured, derived and averaged quantities.

A point q0 = q(t0) ? G is the clinical state of the patient at time t0 and q1 = q(t1) ? G is the clinical state at time t1. In fact in the G space is defined a dynamical flow which evolves the state from q0 to q1 . We suppose that the dynamical evolution is driven by three terms [13] [14]: a deterministic drift b(q(t), t) which is the natural evolution of the state in absence of drugs and mutation, an external field f (q(t), t) representing the drugs, and a stochastic term collecting all the stochastic state changes due for example to the virus mutation and to the environmental changes. The b(q(t), t) and c(q(t), t) functions depend only on q(t) and eventually on the time to satisfy the Markov assumption needed for the model to be coherent with the statistics. The time evolution of the clinical state is given by a stochastic differential equation.

By means of numerical integration technique and the theorems of control theory we simulate the probability distribution evolution in G and use this function to predict the mutation for the virus population.

References:

[1] www.iasusa.org

[2] Hastie T, Tibshirani R, Friedman J (2001) The Elements of Statistical Learning. Springer-Verlag New York Inc.

[3] Kohavi R, John G H (1997) Wrappers for Feature Subset Selection. Artificial Intelligence.

[4] Bengio Y and Nadeau C (2003) Inference for the Generalization Error. Machine Learning, 52, 239-281.

[5] Guyon I, Weston J, Barnhill S and Vapnik V (2002) Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning, 46,

389-422.

[6] I. Savenkov, N. Beerenwinkel, T. Sing, M. Daumer, S.Y. Rhee, M. Horberg, A. Scarsella, A. Zolopa, S.Y. Lee, L. Hurley, W.J. Fessel, R.W. Shafer, R. Kaiser, T. Lengauer (2005) Probabilistic modeling of genetic barriers enables reliable prediction of HAART outcome at below 15% error rate. BioSapiens-viRgil Workshop on Bioinformatics for Infectious Diseases, Bonn, Germany, September 21-23, 2005 (Abstract 24).

[7] Fisher, R. A. 1922. On the dominance ratio. Proc. R. Soc. Edinb. 42:321341.

[8] T Haldane, J. B. S. 1927. A mathematical theory of natural and artificial selection. V. Selection and mutation. Proc. Camb. Philos. Soc. 23:838844.

[9] Kimura, M. 1994. Population genetics, molecular evolution, and the neutral theory. Selected papers. The University of Chicago Press, Chicago, Ill.

[10] Spin Glass Theory and Beyond, ed. M. Mezard, G. Parisi and M. A. Virasoro (World

Scientific, Singapore, 1988).

[11] Amitrano C, Peliti L, Saber M. Population dynamics in a spin-glass model of chemical evolution. J Mol Evol. 1989 Dec;29(6):513-25.

[12] Monasson and R. Zecchina, Statistical mechanics of the random k-satisfiability model, Phys Rev E 56(2) (1997), 1357-1370.

[13] I. M. Rouzine, A. Rodrigoand J. M. Coffin, Transition between Stochastic Evolution and

Deterministic Evolution in the Presence of Selection: General Theory and Application to Virology, Microbiol. Mol. Biol. Rev., 65, 1 (2001), 151-185

[14] Rouzine, I. M., and J. M. Coffin. 1999. Search for the mechanism of genetic variation in the pro gene of human immunodeficiency virus. J. Virol. 73: 81678178.