Identification of diagnostic markers for tuberculosis by proteomic fingerprinting of serum

Correspondence to:
Prof Sanjeev Krishna
s.krishna@sgul.ac.uk

Webappendix: Supplementary methods

Supervised machine learning:

A dataset *D* is represented by a sample of input vectors, *X,* (i.e. individual spectra) with their corresponding sample of output labels, *Y*, such that *D* = [*X,Y*]. A sample input vector is represented by $\mathbf{x}$. The mass spectrum of the *i-th* sample is represented as an n-dimensional vector $\mathbf{x}_i$ with an associated class label y_i (+1 for TB, -1 for control) where n is the number of mass clusters or peaks, $i = 1,..,m$ and m is the number of samples. The spectrum vector elements are denoted by $x_{i,k}$ where $i = 1,..,m$ and $k = 1,..,n$. The classifier prediction of a sample class label y_i is denoted by $\hat{y}_i$.

A supervised learning algorithm is tasked to find a decision function capable of assigning the correct label for a set of input/output pairs of examples, called the training data. The ability of the decision function to predict correct labels for unseen samples (test data) is know as its generalization. Current machine learning methods such as support vector machines (SVM) aim to optimize this property.[1] The generalisation of a classifier is dependent on a set of parameters (model) that must be chosen to optimise performance. For this purpose we adopted a grid search strategy in which a range of parameter values are discretized and tested using cross-validation.

The SVM[2,3] maps its inputs to a high or even infinite, dimensional feature space. The output of the SVM is then a linear thresholded function of the mapped inputs in the feature space, which may be nonlinear in the original input space. The mapping is accomplished by a user-selected reproducing kernel function $\mathrm{K}(\mathbf{x},\mathbf{x}')$ where $\mathbf{x}$ and $\mathbf{x}'$ are input vectors. The kernel function must satisfy Mercer's conditions.[4] Well-known examples of kernels include the Gaussian $\mathrm{K}(\mathbf{x},\mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$ where the parameter σ determines the width; and the polynomial $\mathrm{K}(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}')^d$ where d determines the degree. When $d = 1$ it is called the linear kernel and corresponds to the identity map of the input data. A trained SVM classifier has the form $\mathrm{svm\_classifier}(\mathbf{x}) = \mathrm{sign}\left(\sum_{\mathrm{i}=1}^{\mathrm{m}} \alpha_{\mathrm{i}}\mathrm{K}(\mathbf{x}_{\mathrm{i}},\mathbf{x}) + \mathrm{b}\right)$ and training determines the values of α and b. Typically, many of the αs will be zero. Those that are non-zero are called support vectors and are used to define a separation hyperplane in the transformed feature space. Training a SVM is a convex (quadratic) optimisation problem not subject to local minima, unlike a multi-layer perceptron. We trained soft-margin SVMs which are practicable when data are noisy. In this case the algorithm also minimises the distance of incorrectly classified examples to the margin by adjusting a penalty value, *C,* called the soft-margin parameter.

The single layer perceptron[5] (SLP) is an artificial neural network with one output neuron that computes a linear combination of the values given by the input layer. The discrimination function is given by $\hat{\mathrm{y}} = sign\left(\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{w}_{\mathrm{i}}\mathrm{x}_{\mathrm{i}} + \mathrm{b}\right)$ where weights w are obtained by an iterative learning algorithm designed to reduce the total classification error $\sum_{i=1}^{m} | y_i - \hat{y}_i |$. In our study the SLP did not provide an optimal discriminative function, giving an accuracy of 86.5% in the independent test set (table 3 and fig 1A, blue square).

The multi-layer perceptron[6] (MLP) is a generalisation of the SLP with intermediate layers of hidden neurons. It tackles the problem of non-linearly separable classes by allowing the neurons to process their inputs with a sigmoid function on the activation level $f(\mathrm{a}) = \frac{1}{1+e^{-\mathrm{a}}}$. In this network the weights are learned by a back-propagation algorithm which is a gradient descent rule to minimise the error given by $\sum_{i=1}^{m} (y_i - \hat{y}_i)^2$. With our data the MLP showed similar generalisation performance to SLP, classifying with an accuracy of 86.5% (table 3 and figure 1A, orange diamond).

A decision tree learns to classify a dataset of samples *D*=[*X*,*Y*] by aggregating their features within a set of nodes organised in a binary tree structure. To find the tree structure, sample features are tested according to their discriminative power using a splitting criterion: for a given mass peak $x_{i,k}$ the test $x_{i,k} < T$ where T is any test that produces a binary partition of dataset *D*. In the C4.5 (ref. [7]) classifier the test thresholds are evaluated by an information-gain splitting criterion $\mathrm{Gain}(D,\mathrm{T}) = \mathrm{Info}(D) - \sum_{i=1}^{z} \frac{|D_i|}{|D|} \times \mathrm{Info}(D_i)$ where $\mathrm{Info}(D)$ is an entropy measure of the class to which the sample belongs and z is the number of outcomes of the test T. An iterative algorithm places nodes with increasing information gain from the root to the leaves of the tree. The final tree might be pruned in order to get a more compact representation of the classifier. A testing set sample can be classified by testing its mass peak values against those in the nodes of the tree following a path from the root to a leaf with a classification output. The C5.0 algorithm is an extended version of C4.5 that winnows irrelevant features and incorporates variable misclassification costs (http://www.rulequest.com/). The Alternating Decision Tree[8] (ADTree) is a tree with additional nodes for predicting values that are summed over a classification path and the final output is the sign of this sum. In the TB vs. control dataset (Table 2) the ADTree and the C4.5 classifiers achieved accuracies of 92.3% and 91.0%, respectively (Table 3 and Fig 1A), but relied on AdaBoost[9] boosting to achieve such levels of generalisation[9] (Table 3). We used AdaBoost with 100 iterations for the ADTree and C4.5 classifiers, and boosting with a maximum of 10 iterations for the non-commercial version of the C5.0 classifier.

Mass peak cluster selection

The Pearson correlation coefficient is defined as $R(\mathrm{k}) = \dfrac{\mathrm{covariance}(X_\mathrm{k}, Y)}{\sqrt{\mathrm{variance}(X_\mathrm{k})\mathrm{variance}(Y)}}$

where X_k is the random variable corresponding to the k^{th} component of sample input vectors $\mathbf{x}$ and Y is the random variable of output labels.

The estimate of $R(\mathrm{k})$ is given by $\hat{R}(\mathrm{k}) = \dfrac{\sum_{\mathrm{i}=1}^{\mathrm{m}} (\mathrm{x}_{\mathrm{i,k}} - \bar{\mathrm{x}}_\mathrm{k})(\mathrm{y}_\mathrm{i} - \bar{\mathrm{y}})}{\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{m}} (\mathrm{x}_{\mathrm{i,k}} - \bar{\mathrm{x}}_\mathrm{k})^2 \sum_{\mathrm{i}=1}^{\mathrm{m}} (\mathrm{y}_\mathrm{i} - \bar{\mathrm{y}})^2}}$ where

$\mathrm{x}_{\mathrm{i,k}}$ correspond to value *m/z* of the mass cluster k of sample i, y_i is the class label for sample i and m is the number of samples.

References

1 Cristianini N, Shawe-Taylor J. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge: Cambridge University Press, 2000.
2 Vapnik V. Statistical Learning Theory John Wiley & Sons Inc, 1998.
3 Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational Learning Theory 1992, Pittsburgh, Pennsylvania, United States: 144–152.
4 Aronszajn N. Theory of reproducing kernels. *Trans Amer Math Soc* 1950;**68:**337–404.
5 Rosenblatt F. Principles of Neurodynamics. New York: Spartan Books, 1962.
6 McClelland JL, Rumelhart DE. Parallel and Distributed Processing MIT Bradford Press, 1986.
7 Quinlan JR. C4.5: Programs for Machine Learning. San Francisco: Morgan Kaufmann, 1993.
8 Freund Y, Mason L. The alternating decision tree learning algorithm. In Proceedings of the Sixteenth International Conference on Machine Learning 1999: 124–133.
9 Freund Y, Schapire RE. Experiments with a New Boosting Algorithm. Thirteenth International Conference on Machine Learning 1996, Bari, Italy: 148–156.