ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

Computational Models of Intracytoplasmic Sperm Injection Prognosis

Hui Liu[1], Ash Kshirsagar[1], Jessica Ku[2], Dolores Lamb[2],
Craig Niederberger[1]

[1]University of Illinois at Chicago, Department of Urology,
840 South Wood Street M/C 955, Chicago, Illinois 60612 USA
craign@uic.edu

[2]Baylor College of Medicine, Scott Department of Urology,
One Baylor Plaza Room N730, Houston, Texas 77030 USA

Abstract. Intracytoplasmic sperm injection (ICSI), a procedure in which a single sperm is microinjected into an ovum, presents a difficult modeling problem highly sensitive to both under- and overfitting. We modeled an ICSI data set of 528 outcomes with a variety of clinical and laboratory variates using linear and non-linear (neural computational) logistic regression methods and with linear and radial basis function support vector machines. Interestingly, depending on the threshold chosen to determine a positive outcome, investigated neural computational methods yielded similar or lower ROC AUCs than logistic regression in the test set, whereas support vector machines yielded similar or higher ROC AUCs than those of logistic regression and the investigated neural computational models.

1 Introduction

Van Steirteghem's 1992 report that a single sperm injected into the cytoplasm of the egg resulted in pregnancies in cases where in-vitro fertilization (IVF) had failed revolutionized the treatment of the infertile couple.[5] The technique was termed intracytoplasmic sperm injection, or ICSI. Whereas traditional IVF required at minimum 100,000 and preferably greater than 1 million motile sperm, ICSI required only 1 immotile sperm to render conception and pregnancy. Pregnancy was possible even with necrospermia, a condition in which vital processes within the sperm have ceased.

Prognosticating ICSI outcomes confers a unique modeling problem. As a certain number of eggs are injected with sperm, the embryologist is highly interested in predicting when fertilization will fail entirely, an infrequent but highly significant case. As fertilization failure is rare, models which predict all eggs will fertilize yield high classification accuracy but low ROC AUCs. Overfitting in models prognosticating fertilization failure presents significant difficulty. Likewise, embryologists are interested in predicting whether the fertilization rate will exceed 25%, below which fertilization dysfunction is suggested, and above 50%, indicating high yield.

Supported in part by Grant P01 HD36289 from the National Institutes of Health. 2S

We sought to model fertilization outcomes in ICSI with logistic regression, support vector machines and neural computation to obtain models with high accuracy and to minimize overfitting.

2 Data sets

Variates tracked for ICSI outcomes include maternal and paternal age, number of ova retrieved and in various grades (immature, post-mature, degenerated, unused, fractured,) sperm origin (donor, husband,) source (testicular extraction, electroejaculation, epididymal aspiration,) whether fresh or frozen, semen analysis (volume, density, motility, forward progression, total motile count,) micromanipulation time, ova manipulation parameters (grainy, dark center, killed,) antisperm antibodies, DNA damage, endocrine parameters (FSH, LH, testosterone, estrogen, prolactin, free testosterone, IGF, DHEA-S,) reactive oxygen parameters, leukocytes and SRY. Fertilization was defined as the number of embryos reaching 2 pronuclei after incubation. Three binary modeling outcomes were chosen to represent if any, if $\geq$ 25%, and if $\geq$ 50% of ova fertilized. Of a database of 2177 cases, a fully populated data set of 528 exemplars was generated which was randomly divided into modeling set of 328 and a cross-validation set of 200 in $N1/N2$ fashion (the cross-validation set was independent of the modeling set) with outcome frequency preserved in both subsets. The proportion of positive outcomes in the cross validation set was 4% for if any eggs fertilized, 7% for if $\geq$ 25% fertilized, and 29% for if $\geq$ 50% fertilized.

3 Algorithms

3.1 Logistic Regression

Logistic regression may be formally considered as a neural network with a single output node with a sigmoidal transfer function, no hidden layers and cross-entropy error function

$$e(\mathbf{t}_k, \mathbf{a}) = -\mathbf{t}_k. * \mathbf{log}[\mathbf{a}_k] - (\mathbf{1} - \mathbf{t}_k). * \mathbf{log}[1 - \mathbf{a}_k]. \tag{1}$$

where $\mathbf{a}$ denotes the activation and $\mathbf{t}$ the target for the k_{th} training pattern.[4] Error minimization was achieved using the gradient descent algorithms described in the neural computation section.

3.2 Support Vector Machines

A support vector machine is a supervised learning algorithm developed over the past decade by Vapnik and others.[1, 7]

Given a labeled training set of ℓ examples $(\vec{x}_i, y_i)$, $i = 1, \ldots, \ell$, where $\vec{x}_i$ is the input pattern and y_i is the labeled class, the vectors $\vec{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function Φ. The support vector machine constructs a maximal margin linear classifier in this high dimensional feature space. A positive definite kernel function, $k(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_j').\Phi(\vec{x}_i)$, computes inner products in the feature space, A common kernel is the Gaussian radial basis function (RBF): $k(\vec{x}_i, \vec{x}_j) = e^{-||\vec{x}_i - vecx_j||^2/2\sigma^2}$. Another common kernel is the linear kernel: $k(\vec{x}_i, \vec{x}_j) = \vec{x}_j'\vec{x}_i$. The SVM with a linear kernel is thus a special case of a SVM with a RBF kernel.

The function implemented by a support vector machine is given by

$$f(\vec{x}) = \left\{ \sum_{i=1}^{\ell} \alpha_i y_i k(\vec{x}_i, \vec{x}) \right\} - b. \tag{2}$$

with b a bias parameter. To find the optimal coefficients, $\vec{\alpha}$, of this expansion it is sufficient to maximize the function,

$$W(\vec{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j), \tag{3}$$

in the non-negative quadrant,

$$0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, \ell, \tag{4}$$

subject to the constraint,

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{5}$$

C is a regularization parameter, controlling a compromise between maximizing the margin and minimizing the number of training set errors. The Karush-Kuhn-Tucker (KKT) conditions can be stated as follows:

$$\alpha_i = 0 \implies y_i f(\vec{x}_i) \geq 1, \tag{6}$$

$$0 < \alpha_i < C \implies y_i f(\vec{x}_i) = 1, \tag{7}$$

$$\alpha_i = C \implies y_i f(\vec{x}_i) \leq 1. \tag{8}$$

These conditions are satisfied for the set of feasible Lagrange multipliers, $\vec{\alpha}^0 = \{\alpha_1^0, \alpha_2^0, \ldots, \alpha_\ell^0\}$, maximizing the objective function given by equation 3. The bias parameter, b, is selected to ensure that the second KKT condition is satisfied for all input patterns corresponding to non-bound Lagrange multipliers. Note that in general only a limited number of Lagrange multipliers, $\vec{\alpha}$, will have non-zero values; the corresponding input patterns are known as support vectors. Let $\mathcal{I}$ be the set of indices of patterns corresponding to non-bound Lagrange multipliers, $\mathcal{I} = \{i \ : \ 0 < \alpha_i^0 < C\}$, and similarly, let $\mathcal{J}$ be the set of indices of patterns with Lagrange multipliers at the upper bound C, $\mathcal{J} = \{i \ : \ \alpha_i^0 = C\}$. Equation 2 can then be written as an expansion over support vectors,

$$f(\vec{x}) = \left\{ \sum_{i \in \{\mathcal{I}, \mathcal{J}\}} \alpha_i^0 y_i k(\vec{x}_i, \vec{x}) \right\} - b. \tag{9}$$

SVMs elegantly address the dual difficulties of incurring computational and modeling costs by translating the training set into higher-dimensional space and exposing the learning system to the risk of finding trivial solutions that overfit the data. SVMs avoid overfitting by choosing the maximum margin separating hyperplane from among the many that can separate the 2 classes in the feature space. With this modeling technique, the decision function for classifying points with respect to the separating hyperplane only involves dot products between points in the feature space. Because the algorithm that finds a separating hyperplane in the feature space can be stated entirely in terms of vectors in the input space and dot products in the feature space, a support vector machine can locate the hyperplane without ever representing the space explicitly simply by defining a function, the kernel function, that models the dot product in the feature space. This technique avoids the computational burden of explicitly representing the feature vectors.

For some data sets, the SVM may not be able to find a separating hyperplane in feature space, either because the kernel function is inappropriate for the training data or because the data contains mislabeled examples. The latter problem can be addressed by using a soft margin that allows some training examples to fall on the wrong side of the separating hyperplane. Completely specifying a support vector machine therefore requires specifying two parameters: the kernel function and the magnitude of the penalty for violating the soft margin. The settings of these parameters depend on the specific data to model.

3.3 Neural Computation

Canonical single hidden layer backpropagation neural networks were implemented with hidden node number chosen to minimize overfitting by optimizing ROC AUC in the independent cross-validation set. Final hidden node number over which ROC AUC decayed indicating overfitting was 3. Transfer functions in all nodes were sigmoidal with cross-entropy error as the output error function.1

The error function E_k was statistically paramaterized (weight decay) according to:[3]

$$E_k(\mathbf{w}) = e(t_k, \mathbf{a}_k) + \frac{1}{2\sigma_w^2} \sum_{i=1}^{m} |\mathbf{w}|^2. \tag{10}$$

where $\mathbf{w}$ represents the weight vector. $\sigma_w = 100$ for all learning trials. The learning algorithm was classic off-line (classic batch) backpropagation alone or with Shanno's algorithm or conjugate gradient descent optimization.[3] Learning was specified to be complete when the maximum weight gradient was $\leq 10^{-6}$.

4 Analysis

ROC AUC was computed statistically according to the method of Wickens'.[8] Briefly, hit and false rates were transformed to Gaussian coordinates and linearly fit with errors in both coordinates according to the procedure described in

Numerical Recipes in C.[6] Goodness-of-fit p was calculated as chi-square, with $p \to 1$ indicating better fit.

The non-parametric method according to DeLong was used to compare model ROC AUCs.[2]

5 Results

ROC AUC results are displayed in the following tables. 'FR' indicates fertilization rate outcome modeled, and goodness-of-fit $p-$values for ROC AUCs are displayed in parentheses. Cross-validation set ROC AUCs for logistic regression and neural computation are shown in Table 1, and ROC AUCs for linear and radial basis function SVMs are shown in Table 2.

FR	Logistic Regression		Neural Computation	
>0	0.849	(0.99)	0.757	(0.67)
≥25	0.804	(0.85)	0.775	(0.81)
≥50	0.851	(0.53)	0.849	(0.99)

Table 1: Cross-validation set ROC AUCs for logistic regression and neural computation.

FR	Linear SVM		RBF SVM	
>0	0.751	(0.40)	0.759	(0.61)
≥25	0.790	(0.55)	0.805	(0.48)
≥50	0.867	(0.40)	0.869	(0.23)

Table 2: Cross-validation set ROC AUCs for linear and RBF SVMs.

ROC AUC comparisons by DeLong's method are displayed in Table 3.

FR	LR-LSVM	NNET-LSVM	LR-RSVM	NNET-RSVM	LR-NNET
>0	3.05e-01	3.61e-01	3.80e-01	3.24e-01	2.48e-01
≥25	2.71e-03	3.28e-01	8.51e-04	2.51e-01	2.58e-03
≥50	1.62e-01	3.62e-03	1.27e-01	9.71e-04	9.15e-02

Table 3: $p-$values for ROC AUC comparisons by DeLong's method. LR = logistic regression, NNET = neural computation, LSVM = linear SVM and RSVM = radial basis function SVM.

6 Conclusion

Prognosticating ICSI outcomes presents a difficult and interesting modeling problem, highly sensitive both to under- and overfitting. In our investigations of linear and non-linear (neural computational) logistic regression methods, an ICSI outcomes data set was modeled with ROC AUC varying between 0.751 and 0.869 depending on the threshold chosen for fertilization outcome. In general, neural computational ROC AUCs were similar to or less than those of logistic regression, indicating that although the decision space surface may be complex, it may be possible to describe it with a single hyperplane.

Investigating support vector machines, a linear statistical method with highly adaptive decision space modeling, we achieved lower ROC AUCs than with logistic regression when the threshold chosen was if any ova fertilized with ICSI, indicating overfitting. If we specified the threshold to be if $\geq 25\%$ of ova fertilized, SVM performance was similar to that of logistic regression. However, given a threshold of if $\geq 50\%$ of ova fertilized, SVMs produced higher ROC AUCs than either logistic regression or the investigated neural computational algorithms, with an ROC AUC of 0.867 for a linear kernel, and 0.869 for an RBF kernel.

ICSI outcomes modeling may thus present an application uniquely suited to highly adaptive linear modeling methods such as support vector machines.

References

[1] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–97, 1995.

[2] E.R. DeLong, D.M. DeLong, and Clarke-Pearson D.L. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44:837–45, 1988.

[3] R.M. Golden. *Mathematical Methods for Neural Network Analysis and Design.* The MIT Press, Cambridge, Massachusetts, 1996.

[4] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression.* John Wiley and Sons, Inc., New York, New York, second edition, 2000.

[5] G. Palermo, H. Joris, P. Devroey, and A.C. Van Steirteghem. Pregnancies after intracytoplasmic injection of single spermatozoon into an oocyte. *Lancet*, 340:17–8, 1992.

[6] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*, chapter 15. Cambridge University Press, Cambridge, England, second edition, 1992.

[7] Vladimir N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, Inc., New York, New York, 1998.

[8] T.D. Wickens. *Elementary Signal Detection Theory.* Oxford University Press, New York, New York, 2002.