Early Detection of Prostate Cancer with Classifier Ensembles

Joerg D. Wichard, Henning Cammann, Thomas Tolxdorff
Charité - Universitätsmedizin
Institute of Medical Informatics
Hindenburgdamm 30, 12200 Berlin
{joerg.wichard, henning.cammann, thomas.tolxdorff}@charite.de

Carsten Stephan
Charité - Universitätsmedizin
Department of Urology
Charitéplatz 1, 10098 Berlin
carsten.stephan@charite.de

Abstract

We investigate the performance of different classification models and their ability to recognize prostate cancer in an early state. We build ensembles of classification models in order to increase the classification performance. We measure the performance of our models in an extensive cross-validation procedure. The data sets come from clinical examinations and some of the classification models are already in use to support the urologists in their clinical work.

1. Introduction

Prostate cancer is one of the most common types of cancer among male patients in the western world. The number of expected new cases in the USA for the year 2006 was 235,000 with 27,000 expected deaths [17]. Early detection of prostate cancer improves the chances of a curative treatment and a lot of progress has been made in this field during the last decade. The early detection is considerably enhanced by the measurement of prostate-specific antigen (PSA) in conjunction with other clinically available data like age, digital rectal examination (DRE) and transrectal ultrasonography (TRUS) variables like prostate volume.

We compared several classification models and analyzed their performance on the clinical data set with an extended cross-validation procedure. The models were Linear Discriminant Analysis (LDA), Penalized Discriminant Analysis (PDA) [14], Logistic Regression [1], Classification and Regression Trees (CART) [6], Multi Layer Perceptron (MLP) [3], Support Vector Machines (SVM) [30, 4] and Nearest Neighbour Classifiers [21]. All these models are implemented in an OpenSource Matlab-toolbox that is available on the internet [32].

2. Data

We had access to the clinically available data of 506 patients with 313 cases of Prostate Cancer (PCa) and 193 non-PCa. The data entry for each patient included age, prostate-specific antigen (PSA), the ratio of free to total prostate-specific antigen (PSA-Ratio), the estimated prostate volume (TRUS) and the diagnostic finding from the digital rectal examination (DRE) which was a binary variable (suspicious or non-suspicious). The scatter plot of the variable is shown in Figure 1.

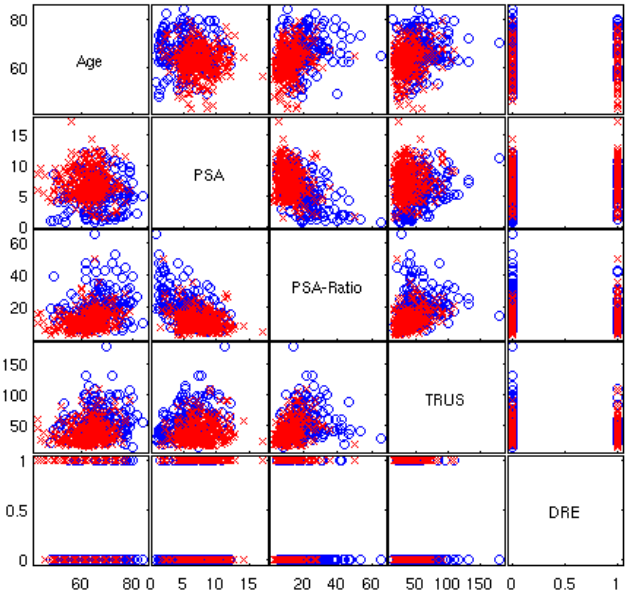

Figure 1. A scatterplot matrix of the PCa data. Each box shows a pair of variables and the cases are color coded, a red cross marks PCa and a blue circle non-PCa. The DRE is a binary variable (suspicious or non-suspicious).

3. Ensembles

Ensemble building is a common way to improve the performance of the resulting model for classification and regression tasks. An ensemble of individual models performs in general better than a single model in the average. This was first introduced in the neural networks community since it was discovered, that a combination of several Neural Networks can reduce the variance of the average regression model [12, 10, 24]. The extension to classification problems was straightforward after the formulation of a bias-variance decomposition for zero-one loss functions [18, 9]. The key feature of the ensemble approach is the introduction of model diversity [20, 23, 19] that helps to reduce the variance of the resulting ensemble model. There are several ways to achieve diverse models like the well known bootstrap aggregating or bagging (see Breiman [5]) where the models are trained on different subsets of the training data or heterogeneous ensembles, that consist of several different model classes like Neural Networks, nearest-neighbor models, decision trees, etc [27, 29].

If we consider a supervised learning problem with n training examples of the form $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ from an unknown function $y = f(\mathbf{x})$. The $\mathbf{x}$ values are usually d-dimensional vectors that are called *input-features* while the y values are continuous in the case of regression and discrete *class labels* in the case of classification. If $y \in \{-1, 1\}$ we call it a *binary classification problem*. A *classifier* is a hypothesis about the unknown function $y = f(\mathbf{x})$ in the sense, that given some new values $\mathbf{x}^*$ it predicts the corresponding class labels y^*. The average output of several different models $f_i(\mathbf{x})$ marks the ensemble model

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{K} \omega_i f_i(\mathbf{x}), \qquad (1)$$

where we assume that the model weights ω_i sum to one $\sum_{i=1}^{K} \omega_i = 1$. There are several suggestions concerning the choice of the model weights (see Perrone et al. [24] or Hashem et al. [13]). We decided to use uniform weights with $\omega_i = 1/K$ for the sake of simplicity and not to run into over-fitting problems as reported by Krogh et al. [19].

The central feature of the ensemble approach is the generalization ability of the resulting model. It is related to finding the right balance between model complexity and generalization in order to avoid overfitting as depicted in Figure 2. Our ensemble approach is based on the observation that the generalization error of an ensemble model could be improved if the models on which averaging is done disagree and if their fluctuations are uncorrelated, see Naftaly et al. [23] and the references therein for a detailed discussion.

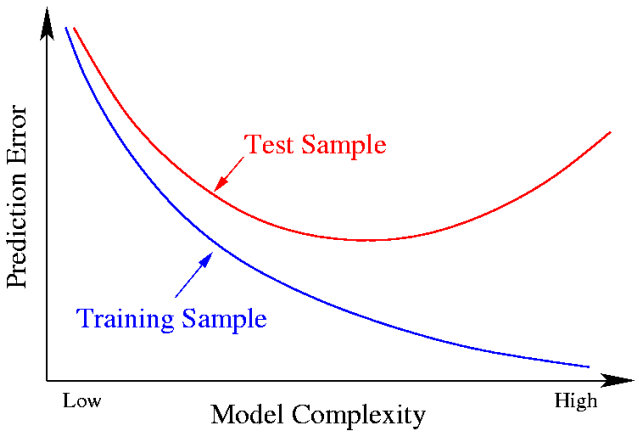

Figure 2. The balance between model complexity and generalization ability of the final classification model.

4. Cross-Validation and Model Selection

Our model selection scheme is a mixture of bagging [5] and cross-validation. *Bagging* or *Bootstrap aggregating* was proposed by Breiman [5] in order to improve the classification by combining classifiers trained on randomly generated subsets of the entire training sets. We extended this approach by applying a cross-validation scheme for model selection on each subset.

In K-fold cross-validation, the data set is partitioned into K subsets. Of these K subsets, a single subset is retained as the validation data for testing the model, and the remaining K - 1 subsets are used for model training. The cross-validation process is then repeated K times with each of the K subsets used only once as the validation data. The K results from the folds then can be averaged to produce a single estimation.

If we lack relevant problem-specific knowledge, cross-validation methods could be used to select a classification method empirically [28]. This is a common approach because it seems to be obvious that no classification method is uniformly superior, see for example Quinlan [25] for a detailed study. It is also a common approach to select the model parameters with cross-validation [11]. The idea to combine the models from the K cross-validation folds (stacking) was described by Wolpert [34].

We suggest to train several models on each CV-fold, to select the best performing model on the validation set and to combine the selected models from the K-folds. If we train models of one type but with different initial conditions (for example Neural Networks with different numbers of hidden neurons) then we find proper values for the free parameters of the model. We could extend that be combining

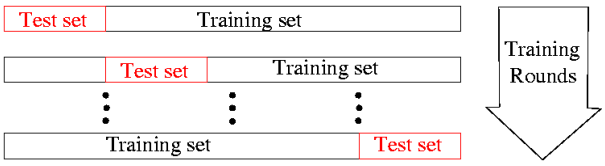

Figure 3. For every partition of the cross-validation, the data is divided in a training and a test set.

models from different classes in order to increase the model diversity. We call this a *heterogeneous ensemble* or *mixed ensemble* and applied this method effectively to regression problems [33] and classification tasks [27].

Our model selection scheme works as follows: For the K-fold CV the data is divided K-times into a *training set* and a *test set*, both sets containing randomly drawn subsets of the data without replications. The size of each test set was 25% of the entire data set. In every CV-fold we train several different models with a variety of model parameters (see Section 5 for an overview of the models and the related model parameters). In each fold we select only one model to become a member of the final ensemble (namely the best model with respect to the test set). This means, that all models have to compete with each other in a fair tournament because they are trained and validated on the same data set. The models with the lowest classification error in each CV-fold are taken out and added to the final ensemble, receiving the weight $\omega_i = \frac{1}{k}$ (see Equation 1). All other models in this CV-fold are deleted.

We can use this model selection scheme in two ways. If we have no idea or prior knowledge, which classification or regression method should be used to cope with a specific problem we could use this scheme in order to look for an empirical answer and to compare the performance of the different model classes. The other way is the estimation of model parameters for the different model classes described in Section 5.

5. Classification Models

In this section we give a short overview of the model classes that we used for ensemble building. All models belong to the canonical collection of machine learning algorithms for classification and regression tasks so details can be found in the textbooks like for instance Hastie et al. [14]. The implementation of these models[1] in an open source toolbox together with a more detailed description can be found in [32].

[1]The toolbox is an open source MATLAB Toolbox which allows the integration of existing implementations of classification algorithms and it contains more then the few model classes described in the text.

5.1 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) is a simple but useful classifier. If we assume that the two classes $k = \{0, 1\}$ have a Gaussian distribution with mean μ_k and they share the same covariance matrix Σ, then the *linear discriminant function* $\delta_k(\mathbf{x}), k = \{0, 1\}$ is given by

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log(\pi_k),$$

where π_k denotes the frequency of occurrence of the class labels. The predicted class labels are given by

$$f(\mathbf{x}) = \text{argmax}_{k=(0,1)} \left\{ \delta_k(\mathbf{x}) \right\} \ .$$

We also implemented two modifications: The Quadratic Discriminant Analysis (QDA) and the Penalized Discriminant Analysis (PDA) as described in detail in Hastie et al. [14]. Linear method are usually conceptually simple, robust and fast and in particular in high dimensional problems they could be very powerfull.

5.2 Logistic Regression Model

Logistic Regression (Log.Reg.) is a model for binomial distributed dependent variables and is used extensively in the medical and social sciences. Hastie et al. [14] pointed out, that the Logistic Regression model has the same form as the LDA, the only difference lies in the way, the linear coefficients are estimated. See Hosmer et al. for a detailed introduction [15]. We used a second order method to estimate the model coefficients which is a critical issue in high dimensional problems, because these calculations are time and memory consuming.

5.3 Nearest Neighbor Classifier

A k-Nearest-Neighbor Classifier takes a weighted average over the labels z_i of those observations $\mathbf{z}_i$ in the training set that are closest to the query point $\mathbf{x}$. This denotes as

$$f(\mathbf{x}) = \frac{1}{\sum w_i} \sum_{\mathbf{z}_i \in N_k(\mathbf{x})} w_i z_i,$$

where $N_k(\mathbf{x})$ denotes the k-element neighborhood of $\mathbf{x}$, defined in a given metric and w_i is the related distance. Common choices are the L_1, L_2 and the L_∞ metrics. The parameters of the model are the number of neighbors and the choice of the metric. Nearest Neighbor Classifiers offer a very intuitive approach to classification problems because they are based on the concept of similarity. This works fine in lower dimensions but leads to some trouble in higher dimensional probelms, known as the *curse of dimensionality* [2].

5.4 Trees

Trees are conceptually simple but powerful tools for classification and regression. For our purpose we use the *classification and regression trees* (CART) as described in Breiman et al. [6]. The main feature of the CART algorithm is the binary decision role that is introduced at each tree node with respect to the information content of the split. In this way the most discriminating binary splits are near the tree root building an hierarchical decision scheme. It is known, that trees have a high variance, so they benefit from the ensemble approach [5]. These trees ensembles are also know as *random forests*. The parameters of the tree models are related to splitting the tree nodes (the impurity measure and the split criterion, see [14] for a detailed description).

5.5 Artificial Neural Networks

We train a multilayer feed-forward Neural Network (MLP: Multi Layer Perceptron) with the a sigmoid activation function. The weights are initialized with Gaussian distributed random numbers having zero mean and scaled variances. The weights are trained with a gradient descend based on the Rprop Algorithm [26] with the improvements given in [16]. The MLP works with a common weight decay with the penalty term

$$P(\vec{w}) = \lambda \sum_{i=1}^{N} \frac{w_i^2}{1 + w_i^2} \ ,$$

where $\vec{w}$ denotes the N-dimensional weight vector of the MLP and a small regularization parameter λ. The number of hidden layers, the number of neurons and the regularization parameter are adjusted during the CV-training. We further applied the concept of an ϵ-insensitive error loss that we introduced in the context of CNN-learning [22].

5.6 Support Vector Machines

Over the last decade Support Vector Machines (SVMs) have become very powerful tools in machine learning. A SVM creates a hyperplane in a feature space that separates the data into two classes with the maximum-margin. The feature space can be a mapping of the original features $(\mathbf{x}, \mathbf{x}')$ into a higher dimensional space using a positive semi-definite function

$$(\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}').$$

The function $k(\cdot, \cdot)$ is called the *kernel function* and the so called *kernel trick* uses Mercer's condition, which states that any positive semi-definite kernel $k(\mathbf{x}, \mathbf{x}')$ can be expressed as a dot product in a high-dimensional space (see [8] for a detailed introduction). The theoretical foundations of this approach were given by Vapnik's *Statistical Learning Theory* [31, 30] and later extended to the nonlinear case [4]. We use an implementation of SVMs that is based on the libsvm provided by Chih-Jen Lin [7] with the standard kernels:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= (\mathbf{x} \cdot \mathbf{x}') && \text{linear} \\ &= (\mathbf{x} \cdot \mathbf{x}' + 1)^d && \text{polynomial} \\ &= \exp\left(-\tfrac{||\mathbf{x}-\mathbf{x}'||}{\sigma^2}\right) && \text{rbf} \end{aligned}$$

The parameters of the model are with respect to the kernel-type the polynomial degree d , the width of the rbf σ^2 and the value concerning the cost of constrain violation during the SVM training.

		Predicted	
		Class 1	Class −1
Real	Class 1	tp	fn
	Class −1	fp	tn

Figure 4. The confusion matrix for a binary classification problem. The notation: tp = true positive; tn = true negative; fp = false positive; fn = false negative.

6. Application to the clinical prostate data

We compared the model classes described above in a unified framework under fair conditions. Thus we trained an ensemble of each model class consisting of 11 ensembles members (11 CV-folds in the training scheme described in Section 4). The performance of each ensemble model was assessed on the 20% of data (validation set), which was initially removed and never included in model training. This procedure was independently repeated 20 times. This means that all model-building processes, that is, the random removal of 20% of the data, the construction of a classification model ensemble on the remaining 80% of the data as outlined in Section 4, and the final test on the unseen validation data were performed each time. Finally the mean average prediction values with respect to the validation set were calculated and are listed in Table 1. We used three different performance measures in order to compare the different classification models. Therefore we have to define the four possible outcomes of a classification that can be formulated

	Accuracy	F-score	AUC
LDA/PDA	0.776	0.823	0.863
Log.Regr.	0.778	0.823	0.868
MLP	0.791	0.823	0.863
SVM	0.795	0.833	0.825
CART	0.757	0.809	0.843
KNN	0.756	0.813	0.809
Mixed	0.783	0.828	0.860

Table 1. The average performance of several classifier ensembles with respect to the validation set which was initially removed and never included in model training. We show the mean values from 20 independent validation runs.

in a 2×2 confusion matrix, as shown in Figure 4. The

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

seems to be the canonical error measure for almost all classification problems if the data set is balanced. Other important measures are the specificity that quantifies how well a binary classification model correctly identifies the negative cases

$$\text{Specificity} = \frac{tn}{tn + fp}$$

and the sensitivity, which is the proportion of true positives of all diseased cases in the population

$$\text{Sensitivity} = \frac{tp}{tp + fn}.$$

A high sensitivity is required when early diagnosis and treatment is beneficial, which is the case in PCa.

The precision or positive predictive value (PPV) is given by

$$\text{PPV} = \frac{tp}{tp + fp}$$

and is the proportion of patients with positive test results who are correctly diagnosed. The F-Score is the harmonic mean of precision and sensitivity

$$\text{F-Score} = 2 \cdot \frac{\text{Sensitivity} \cdot \text{PPV}}{\text{Sensitivity} + \text{PPV}}$$

and it is usefull if the classes in the classification problem are not equally distributed. Another important measure is the area under curve (AUC) wherein the curve is the ROC-curve (Receiver Operating Characteristic). The ROC-curve is the graphical plot of the sensitivity versus the (1 - specificity) for a binary classifier as its discrimination threshold is varied.

If we compare the outcome of the statistical analysis of the model performance as listed in Table 1 we can state, that the differences between the different classifiers are marginal. Even the more sophisticated classification models (SVMs or Mixed Ensembles) could not outperform the robust linear candidates (LDA/PDA). The current version of the software package *ProstataClass*[2], which was developed at Charité uses an Artificial Neural Network as classification engine. One of the reasons for that is the growing use and acceptance of Artificial Neural Networks as tools for diagnostic support in the medical community.

7. Conclusions

We compared several classification models with respect to their ability to recognize prostate cancer in an early state. This was done in an ensemble framework in order to estimate proper model parameters and to increase classification performance. It turned out, that all models under investigation are performing very well with only marginal differences.

References

[1] Y. Bard. *Nonlinear Parameter Estimation*. Academic Press, New York, 1974.

[2] R. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *In Proc. of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. new edition 1993.

[7] C. C. Chang and C. Lin. Libsvm - A library for support vector machines, 2001.

[8] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[9] P. Domingos. A unified bias-variance decomposition for zero-one and squared loss. In *Proc. of the 17 th National Conference on Artifcial Inteligence*, pages 564–569, 2000.

[10] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[2]The Software is available under the URL: www.charite.de/ch/uro/de/html/patient_erkrankungen/prostatabiopsie.html

[12] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[13] T. Hashem and B. Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Trans. Neural Networks*, 6(3):792–794, 1995.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer-Verlag, 2001.

[15] D. Hosmer and S. Lemeshow. *Applied logistic regression*. John Wiley and Sons, 1989.

[16] C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In H. Bothe and R. Rojas, editors, *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, pages 115–121. ICSC Academic Press, 2000.

[17] A. Jemal, R. Siegel, E. Ward, and M. Thun. Cancer facts & figures. Technical report, Department of Epidemiology and Surveillance Research, American Cancer Society, Atlanta, Georgia, 2006.

[18] R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 275–283. Morgan Kaufmann, 1996.

[19] A. Krogh and P. Sollich. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811–825, 1997.

[20] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

[21] C. Merkwirth, U. Parlitz, and W. Lauterborn. Fast exact and approximate nearest neighbor searching for nonlinear signal processing. *Phys. Rev. E*, 62(2):2089–2097, 2000.

[22] C. Merkwirth, J. Wichard, and M. Ogorzałek. Stochastic gradient descent training of ensembles of DT-CNN classifiers for digit recognition. In *Proceedings of the European Conference on Circuit Theory and Design ECCTD'03*, pages 337–341, Krakøw, Poland, 2003.

[23] U. Naftaly, N. Intrator, and D. Horn. Optimal ensemble averaging of neural networks. *Network, Comp. Neural Sys.*, 8:283–296, 1997.

[24] M. P. Perrone and L. N. Cooper. When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall, 1993.

[25] J. Quinlan. *Comparing connectionist and symbolic learning methods*, volume I, pages 445–456. MIT Press, 1994.

[26] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Int. Conf. on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[27] A. Rothfuss, T. Steger-Hartmann, N. Heinrich, and J. Wichard. Computational prediction of the chromosome-damaging potential of chemicals. *Chemical Research in Toxicology*, 19(10):1313–1319, 2006.

[28] C. Schaffer. Selecting a classification method by cross-validation. In *Fourth Intl. Workshop on Artificial Intelligence & Statistics*, pages 15–25, January 1993.

[29] T. Taskaya-Temizel and M. Casey. A comparative study of autoregressive neural network hybrids. *Neural Networks*, 18(5-6):781–789, 2005.

[30] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1999.

[31] V. Vapnik and A. J. Tscherwonenkis. *Theorie der Zeichenerkennung*. Akademie Verlag Verlag, Berlin, 1979.

[32] J. Wichard and C. Merkwirth. ENTOOL - A Matlab toolbox for ensemble modeling. http://www.j-wichard.de/entool/, 2007.

[33] J. Wichard, M. Ogorzałek, and C. Merkwirth. Detecting correlation in stock markets. *Physica A*, 344:308–311, 2004.

[34] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.