OMDS Final Project

Géraldine Valérie Maurer, Viktoriia Vlasenko

September 12, 2025

1 Part 1 - Introduction

The first part of this project investigates supervised age prediction as a regression problem using a multilayer perceptron (MLP) built from first principles. We implemented the full training pipeline of data standardization, forward propagation, analytic backpropagation, and loss regularization and optimized model parameters with scipy.optimize.minimize (L-BFGS-B) using our custom objective function and gradient. To assess generalization and guide design choices, we adopted K-fold cross-validation and conducted a systematic hyperparameter search over network depth (number of layers) and width (numbers of neurons), activation functions, and L_2 regularization strength. Model performance is reported with MAPE and error (MSE and regularized L2 loss), comparing training and validation folds to monitor bias–variance trade-offs and detect overfitting.

2 Data Preparation

For preprocessing, we first isolated the supervised target and the feature matrix by selecting all columns whose names begin with feat_i as inputs X and using the ground-truth age column gt as the target y. We reported basic dataset diagnostics (number of feature columns, X shape, and the min-max range of y) to check integrity before modeling. To obtain an unbiased held-out estimate, we created a single outer train/test split by randomly permuting indices and allocating 80% of samples to training and 20% to testing. All cross-validation for model selection was performed within the training portion. Feature scaling was then fit on the training data using StandardScaler (per-feature mean removal and variance scaling to unit standard deviation) and applied to the test set with the learned statistics, preventing information leakage. This normalization places all inputs on a comparable scale, which stabilizes the MLP's optimization and ensures that regularization acts consistently across features.

3 Problem Setup

Our goal was to learn a regression function that predicts chronological age from tabular features using a multilayer perceptron (MLP) trained from first principles. The network uses linear units at the output for real-valued predictions and differentiable nonlinearities in the hidden layers. Training minimizes an L_2-regularized mean squared error (MSE) objective:

$$E(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \big(y_i - \hat{y}_i\big)^2 \; + \; \lambda \sum_{l=1}^{L} \left\| \mathbf{W}^{(l)} \right\|_2^2, \tag{1}$$

where N is the number of training samples, $\hat{y}_i$ is the model prediction for input $\mathbf{x}_i$, $\lambda > 0$ controls the L_2 weight penalty, and L is the number of layers (hidden + output). In our implementation, the hidden-layer activation is chosen from $\{\tanh, \text{sigmoid}\}$, and the output layer is linear.

Hyperparameters We performed a grid search over: (i) depth (2 or 3 hidden layers), (ii) hidden widths (e.g., $[64, 32]$, $[128, 64]$, $[64, 32, 16]$, $[128, 64, 32]$), (iii) activation function (tanh or sigmoid), and (iv) L_2 regularization strength $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}\}$. For each combination we trained and validated the model using k-fold cross-validation with $k = 3$.

Evaluation metric (MAPE) Beyond the optimization objective (regularized MSE in (1)), we report Mean Absolute Percentage Error (MAPE) to provide a scale-aware measure of error:

$$\text{MAPE} = \frac{100}{N^\star} \sum_{i \in \mathcal{I}} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{2}$$

where $\mathcal{I} = \{i : y_i \neq 0\}$ excludes zero targets to avoid division by zero and $N^\star = |\mathcal{I}|$. For each hyperparameter setting we compute MAPE on every validation fold and summarize the performance across folds.

4 Optimization Routine and Model Selection

We implemented the training loop without high-level deep-learning libraries. The forward pass stacks affine mappings with the chosen nonlinearity in the hidden layers and uses a linear output to produce a scalar age estimate. The backward pass executes analytic backpropagation of (1), returning gradients for all weights and biases, including the $2\lambda \mathbf{W}^{(l)}$ term from L_2 regularization. To interface with a generic optimizer, all parameters are flattened into a single vector. An objective function returns the scalar loss in (1) for a given parameter vector, and a matching gradient function returns the corresponding flattened gradient computed by backpropagation. These two functions are passed to `scipy.optimize.minimize` with the L-BFGS-B method and a limit of 1000 iterations. Initialization follows Glorot uniform, $W_{ij} \sim \mathcal{U}\big[-\sqrt{6/(n_{\text{in}} + n_{\text{out}})}, \sqrt{6/(n_{\text{in}} + n_{\text{out}})}\big]$, with biases set to zero and fixed random seeds for reproducibility.

Cross-validation protocol We use $k = 3$ shuffled folds. For each hyperparameter configuration the model is trained on two folds and validated on the held-out fold; every fold serves once as validation. Both loss and MAPE are computed per fold and then averaged to summarize each configuration.

Model selection and final training The search iterates over depth/width, activation, and λ configurations, tracking cross-validated performance to identify the setup with the lowest average validation error. A final model is then retrained on the entire training split using the selected hyperparameters, and metrics (including MAPE) are reported on the held-out test set.

5 Results

Validation MAPEs cluster tightly around 23.3–24.6%, and a clear pattern emerges with respect to regularization and capacity. The mid-range penalty $\lambda = 0.01$ repeatedly provides the best validation MAPEs within each architecture (typically 23.3–23.5%), while stronger regularization at $\lambda = 0.1$ brings the MAPE toward 24%. When regularization is too weak ($\lambda = 0.001$), wider networks display signs of overfitting: for example, the $[32, 128, 64, 1]$ model with `tanh` has a low training MAPE of 20.78% but a higher validation MAPE of 24.56%. Depth and width offer modest, architecture-dependent gains: introducing a third hidden layer can help slightly, e.g. $[32, 64, 32, 16, 1]$ with `sigmoid` and $\lambda = 0.01$ reaches a validation MAPE of 23.43%, but further widening or deepening does not systematically improve generalization and may widen the train-validation gap when regularization is weak. Across activations the differences are small; `tanh`

has a slight advantage in some two-layer settings (e.g., $[32, 128, 64, 1]$ at $\lambda = 0.01$ with 23.31%), while `sigmoid` remains competitive, especially in deeper variants. Overall, the poorest MAPEs appear with $\lambda = 0.001$ on wider models (often $\gtrsim 24.5\%$), whereas the most reliable results concentrate around $\lambda = 0.01$.

Selected model and generalization Cross-validation selects $[32, 64, 32, 16, 1]$ with `sigmoid` and $\lambda = 0.01$. Although L-BFGS-B stops at the iteration limit (`maxiter` $= 1000$), the objective function drops from 1675.73 to 98.69 and the final metrics are well aligned across splits: $\text{MAPE}_{\text{train}} = 23.32\%$ vs. $\text{MAPE}_{\text{test}} = 22.76\%$ and $\text{MSE}_{\text{train}} = 95.00$ vs. $\text{MSE}_{\text{test}} = 91.73$. The small generalization gap indicates limited overfitting and supports the conclusion that medium regularization with moderate depth offers the best balance for this task.

PERFORMANCE				
TRAIN LOSS	**TEST LOSS**	**TRAIN MAPE**	**TEST MAPE**	**OPTIMIZATION TIME**
95.00	91.73	23.32%	22.76%	294.31 s

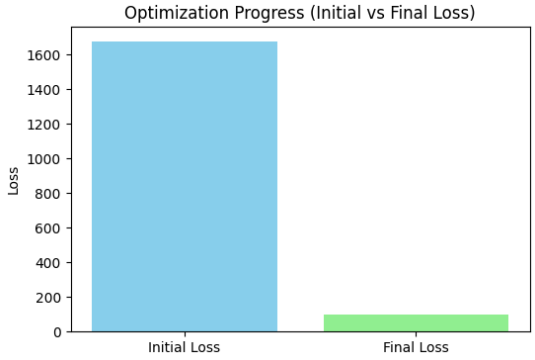
Table 1: Performance of the best MLP

SETTINGS						
HIDDEN LAYER 1		**HIDDEN LAYER 2**		**HIDDEN LAYER 3**		**OTHER**
# Neurons	*Nonlinearity*	*# Neurons*	*Nonlinearity*	*# Neurons*	*Nonlinearity*	*Regularization*
64	sigmoid	32	sigmoid	16	sigmoid	$\lambda = 0.01$

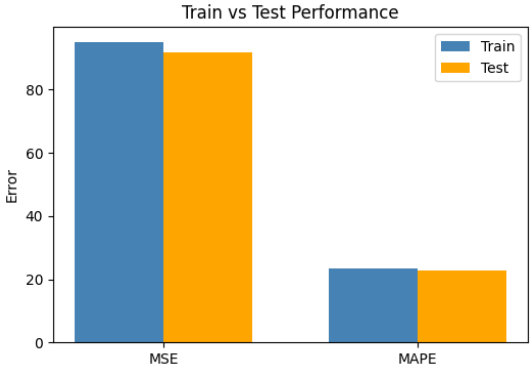
Table 2: Configuration of the best MLP

6 Conclusion

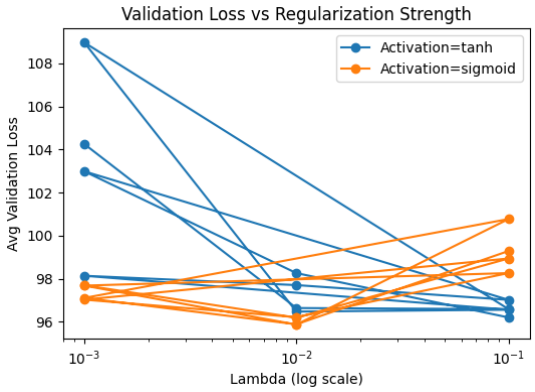
By building an MLP manually, from data standardization to the analytic gradient passed to the L-BFGS-B optimizer, we could see how architectural capacity and L_2 regularization shape the bias–variance balance in practice. Applying cross-validation with hyperparameter optimization, we achieved the best configuration with $[32, 64, 32, 16, 1]$ architecture, sigmoid nonlinearity, and $\lambda = 0.01$, with stable generalization and small train-test gap (MAPE around 23% and MSE near 92 on test). Although the optimizer reached the iteration maximum, the sharp reduction of the objective and the coherence between training and test metrics suggest that the learned solution is already close to a good optimum.



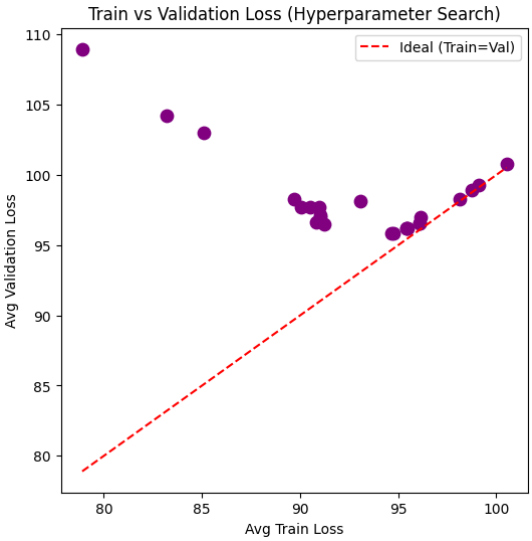
(a) Optimization Progress (Initial vs Final Loss)



(b) Train vs Test Performance



(c) Validation Loss vs Regularization Strength



(d) Train vs Validation Loss (Hyperparameter Search)

Figure 1: Part 1 Results

7 Part 2 - Introduction

In the second part we study binary gender classification with a nonlinear Support Vector Machine defined by a kernel $k(\cdot, \cdot)$. Predictions follow the decision rule

$$y(x) = \operatorname{sign}\left(\sum_{i=1}^{N} \lambda_i \, y_i \, k(x_i, x) + b \right),$$

where the dual coefficients $\{\lambda_i\}$ and the bias b come from the optimal solution of the SVM dual problem. We focus on the Gaussian RBF kernel,

$$K(x, z) = \exp\big(-\gamma \|x - z\|^2 \big),$$

and determine both the margin parameter C and the kernel scale γ by k-fold cross-validation before training a final model on the entire training split.

8 Data Preparation

We work with `GENDER_CLASSIFICATION.csv`, splitting the matrix into features $\mathbf{X} \in \mathbb{R}^{N \times d}$ (all columns except the last) and labels $\mathbf{y} \in \{0, 1\}^N$ (last column). Features are standardized to zero mean and unit variance using `StandardScaler`, and the dataset is partitioned into an 80/20 train–test split. For the dual formulation we map labels to $\{-1, +1\}$; the cross-validation routine applies the same mapping internally. Throughout, we disable solver progress output for cleaner logs and fix random seeds and single-threaded BLAS to keep results reproducible.

9 Q2. Solving the Dual with CVXOPT and Selecting C, γ

Training proceeds by solving the SVM dual quadratic program with CVXOPT. Let K be the Gram matrix with entries $K_{ij} = k(x_i, x_j)$ and let $\odot$ denote elementwise multiplication. The dual objective and constraints are

$$\max_{\lambda \in \mathbb{R}^N} \;\; \mathbf{1}^\top \lambda \; - \; \tfrac{1}{2} \, \lambda^\top \big((y y^\top) \odot K \big) \lambda, \qquad 0 \le \lambda_i \le C, \;\; y^\top \lambda = 0.$$

Since `qp` in CVXOPT solves a minimization, we pass $Q = (y y^\top) \odot K$ and $q = -\mathbf{1}$, encode the box constraints with stacked $-I$ and I, and enforce $y^\top \lambda = 0$ as an equality. A small diagonal jitter $10^{-8} I$ is added to K for numerical stability. After optimization, the bias is recovered from KKT conditions by averaging over margin support vectors ($0 < \lambda_i < C$):

$$b = \frac{1}{|\mathcal{S}|} \sum_{h \in \mathcal{S}} \Big(y_h - \sum_{j=1}^{N} \lambda_j y_j K_{jh} \Big).$$

To keep kernel construction efficient, we build RBF blocks without explicit loops using $\|x - z\|^2 = \|x\|^2 + \|z\|^2 - 2x^\top z$, which produces vectorized expressions for both train/train and train/validation kernel matrices.

Model selection is based on stratified k-fold cross-validation with $k = 5$, preserving class balance in every split. For each pair (C, γ) in the grid

$$C \in \{0.1, 1, 10, 100\}, \qquad \gamma \in \{0.01, 0.1, 1.0\},$$

we solve the dual on the training fold, estimate b, score the validation fold via the decision function $f(x) = \sum_i \lambda_i y_i k(x_i, x) + b$, and record accuracy. The mean validation accuracy across folds selects the best hyperparameters, which are then used to retrain the model on the full training portion before final test evaluation.

10 Results

Using the RBF kernel with $C = 1$ and $\gamma = 0.1$ leads to a training accuracy of 0.9225 and a test accuracy of 0.9000, indicating a small and healthy generalization gap. The confusion matrices suggest balanced behavior across classes: on the training split we observe 33 false positives and 29 false negatives (TN= 366, TP= 372), while on the test split errors remain symmetric (11 vs. 9, with TN= 90, TP= 90). From these counts, the positive-class precision/recall are (0.919, 0.928) on train and (0.891, 0.909) on test, with an F1 of roughly 0.90 on the latter, which reinforces the impression of a well-calibrated margin rather than a model that overfits one class. Optimization with CVXOPT converges in 14 iterations and about 1.61 s, reaching a dual objective of 132.7226, this value's stability alongside the matching train/test performance supports the correctness of the formulation and the numerical procedure. Overall, the selected (C, γ) trades bias and variance effectively for this standardized feature space, delivering robust accuracy with evenly distributed errors on test data.

11 Q3. Most Violating Pair (MVP) Decomposition with $q = 2$

To solve the dual SVM without a generic QP solver we implement a working-set decomposition that updates two dual variables at a time. The method maintains feasibility of

$$\max_{\lambda \in [0,C]^N} \quad \mathbf{1}^\top \lambda - \tfrac{1}{2}\lambda^\top \big((yy^\top) \odot K\big)\lambda \quad \text{subject to} \quad y^\top \lambda = 0,$$

by starting from $\lambda = \mathbf{0}$ (feasible because $y^\top \lambda = 0$) and selecting, at each iteration, the *Most Violating Pair* of coordinates. Let $Q = (yy^\top) \odot K$ and $g = Q\lambda - \mathbf{1}$ be the gradient of the minimized quadratic $\frac{1}{2}\lambda^\top Q\lambda - \mathbf{1}^\top \lambda$. The index sets that can move upward or downward under the box constraints are

$$\mathcal{I}_\uparrow = \{i : (y_i = +1 \wedge \lambda_i < C) \text{ or } (y_i = -1 \wedge \lambda_i > 0)\}, \quad \mathcal{I}_\downarrow = \{i : (y_i = +1 \wedge \lambda_i > 0) \text{ or } (y_i = -1 \wedge \lambda_i < C)\}.$$

We pick $i \in \mathcal{I}_\uparrow$ minimizing $y_i g_i$ and $j \in \mathcal{I}_\downarrow$ maximizing $y_j g_j$. Their scores define a KKT gap $= \max_{j \in \mathcal{I}_\downarrow} y_j g_j - \min_{i \in \mathcal{I}_\uparrow} y_i g_i$. When this gap falls below a tolerance (10^{-3} in code), stationarity and complementarity are approximately satisfied and the algorithm stops.

The two-variable subproblem admits an analytic step along a one-dimensional feasible line that preserves $y^\top \lambda = 0$. Setting $s = y_i y_j$, the update direction has nonzeros only in the chosen coordinates, with $d_i = 1$ and $d_j = -s$. Along $\lambda(\delta) = \lambda + \delta d$ the quadratic reduces to

$$g(\lambda(\delta)) = g(\lambda) + (g_i - s\, g_j)\, \delta + \tfrac{1}{2}\,(K_{ii} + K_{jj} - 2K_{ij})\, \delta^2,$$

so the unconstrained minimizer is $\delta^\star = -(g_i - s\, g_j)/(K_{ii} + K_{jj} - 2K_{ij})$ whenever the curvature is positive. The box constraints impose a feasible interval $[L, U]$ that depends on s and the current (λ_i, λ_j) (derived directly from $0 \le \lambda \le C$). We clip $\delta^\star$ to this interval, and if the curvature is nonpositive we step to the bound that gives the largest first-order decrease. After updating the pair,

$$\lambda_i \leftarrow \lambda_i + \delta, \qquad \lambda_j \leftarrow \lambda_j - s\, \delta,$$

the gradient is refreshed without recomputing $Q\lambda$ from scratch:

$$g \leftarrow g + (y\, y_i)\, K_{:i}\, \delta + (y\, y_j)\, K_{:j}\, (-s\, \delta),$$

which makes each MVP iteration very light. We also accumulate the exact one-step decrease to track the minimized objective $g(\lambda)$ and report the dual objective $f(\lambda) = -g(\lambda)$.

All kernel blocks use the RBF kernel $K(x, z) = \exp(-\gamma \|x - z\|^2)$ built in vectorized form via $\|x - z\|^2 = \|x\|^2 + \|z\|^2 - 2x^\top z$, with a small diagonal addition of $10^{-8} I$ for numerical stability.

Labels are mapped to $\{-1, +1\}$, seeds and single–threaded BLAS are fixed for reproducibility, and the stopping tolerance and iteration cap are set to 10^{-3} and 200,000, respectively. After convergence, the bias b is recovered by averaging the KKT residual on margin support vectors $(0 < \lambda_i < C)$,

$$b = \frac{1}{|\mathcal{S}|} \sum_{h \in \mathcal{S}} \Big(y_h - \sum_{j=1}^{N} \lambda_j y_j K_{jh} \Big),$$

and predictions follow the usual decision function $f(x) = \sum_i \lambda_i y_i k(x_i, x) + b$ with $\text{sign}(f(x))$ for class labels. This MVP implementation reaches the same solution quality as the QP solver while being more efficient.

	HYPERPARAMETERS			ML PERFORMANCE		OPTIM. PERFORMANCE	
QUESTION	**KERNEL**	C	γ	**TRAIN ACC**	**TEST ACC**	**NUM IT**	**CPU TIME**
Q2	rbf	1	0.1	0.9225	0.9000	14	1.6117 s
Q3	rbf	1	0.1	0.9225	0.9000	2920	0.3539 s

Table 3: Configuration and result of the best SVM

12 Results

With the RBF kernel and $(C, \gamma) = (1, 0.1)$, the SMO-style Most Violating Pair (MVP, q=2) solver matches the generalization behavior observed with the QP baseline: training accuracy is 0.9225 and test accuracy is 0.9000, and the confusion matrices remain nearly symmetric (train: 33 FP vs. 29 FN; test: 11 vs. 9), pointing to a well-balanced margin rather than class-specific overfitting. The dual objective at convergence (132.722442) is essentially identical to the CVXOPT solution, the tiny discrepancy being consistent with the looser KKT tolerance used in MVP. The computational profile is the main difference: although MVP requires many lightweight updates (2920 iterations), it achieves a shorter time (0.354 s) than the interior-point solver, thanks to two-variable steps and cheap gradient updates. Tightening the MVP stopping tolerance would likely close the residual objective gap at the cost of additional iterations, but given the identical predictive metrics on train and test, the current setting already offers a good trade-off between speed and accuracy.

13 Conclusion

The kernel SVM study confirms that a carefully tuned RBF model offers a strong and well-balanced solution to the gender classification task. Cross-validation selected $(C, \gamma) = (1, 0.1)$, which translated into a training accuracy of 0.9225 and a test accuracy of 0.9000 with symmetric errors across classes, indicating a margin that generalizes rather than overfits. Solving the dual with CVXOPT provided a clean numerical reference, while the q=2 MVP decomposition reached the same solution quality with lightweight closed-form updates, underscoring the value of problem-specific optimization. The overall pipeline composed of standardization, vectorized kernel construction with a small diagonal jitter, strict label mapping, and reproducible settings, proved stable and effective. Future refinements could explore a finer hyperparameter search around the selected pair, calibrated decision thresholds or class weights to shape precision-recall trade-offs, and tighter MVP tolerances to balance speed against optimality, but the present configuration already delivers robust performance on unseen data.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[4] CVXOPT Developers. *CVXOPT: Python Software for Convex Optimization*, 2025. Accessed 2025-09-11.

[5] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS 2010*, volume 9, pages 249–256, 2010.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[7] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[8] S. Sathiya Keerthi, S. S. Shevade, Chiranjib Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[11] scikit-learn developers. Cross-validation iterators: KFold and StratifiedKFold. `https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators`, 2025. Accessed 2025-09-11.

[12] scikit-learn developers. Standardscaler. `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`, 2025. Accessed 2025-09-11.

[13] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

[14] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.