Research on noise robustness based on KNN algorithm

Anonymous CVPR submission

Paper ID *****

Abstract

As a classical machine learning technique, K-Nearest Neighbors (KNN) algorithm is widely used in pattern recognition and data mining tasks in various fields because of its intuitive principle and easy implementation. Although the KNN algorithm performs well under ideal conditions, its performance is often limited in the face of noise and outliers that are common in the real world. Noisy data may not only distort the true data distribution, but also mislead the algorithm to make wrong predictions. To address this challenge, this paper deeply explores the robustness of KNN algorithm under the influence of noise and systematically reviews various improvements proposed in the existing literature. Based on this, this paper proposes an innovative research question: how to enhance the robustness of KNN algorithm to noise while maintaining the simplicity of the algorithm? In order to answer this question, this paper designs and implements a new improved KNN algorithm, which significantly improves the classification accuracy and robustness in noisy environments by integrating data preprocessing technology and dynamic weight adjustment strategy. Through a series of carefully designed experiments, this paper verifies the effectiveness of the proposed algorithm. These experimental results not only provide a new perspective for the further research of KNN algorithm, but also provide a valuable reference for noise data processing in practical applications.

1. Introduction

The K-Nearest Neighbors (KNN) algorithm is renowned for its simplicity and effectiveness in various machine learning tasks. However, its sensitivity to noise in the data has prompted extensive research into methods for enhancing its robustness. This review explores advanced approaches developed to address this issue, summarizing recent innovations, evaluating effectiveness of the model in this paper, and identifying areas for further improvement.

1.1. State-of-the-Art Methods

Learnable Distance Metrics Recent studies have focused on learning distance metrics tailored to the data's underlying structure. For instance, Metric Learning for Large Margin Nearest Neighbour (LMNN) [5] modifies the distance metric to ensure that same-class points are closer toghther while maintaining larger margins between different classes. This approach adapts the distance metric to the specific dataset, improving robustness to noise.

Advanced Distance Metrics (1)Learnable Distance Metrics. Recent studies have focused on learning distance metrics tailored to the data's underlying structure. For instance, Metric Learning for Large Margin Nearest Neighbor (LMNN) modifies the distance metric to ensure that same-class points are closer together while maintaining larger margins between different classes. This approach adapts the distance metric to the specific dataset, improving robustness to noise.

(2)Adaptive Neighbor Selection. The concept of adaptive neighbors involves dynamically adjusting the neighborhood size based on local data density. Techniques such as Local Outlier Factor (LOF) [6] have been adapted to KNN to account for variations in data density and reduce the influence of noisy instances.

Hybrid Models Neural Network Integration: Combining KNN with neural networks [19] for feature extraction has shown promise in enhancing noise robustness. Deep Feature Learning methods use neural networks to extract high-level features from the data, which are then used as inputs for KNN. This integration leverages the ability of neural networks to learn complex representations, reducing the impact of noise on the KNN algorithm.

Ensemble Methods **(1)KNN Ensembles** Ensemble techniques like Bagging and Boosting have been applied to KNN to enhance its robustness. By aggregating predictions from multiple KNN models trained on different subsets of the data, ensemble methods can mitigate the effects of noise [22]. Techniques such as Random KNN (RKNN) [18], which uses different random subsets of features for each KNN model, have been particularly effective.

(2)Hybrid Ensembles: Combining KNN with other machine learning models in an ensemble framework can further enhance robustness. For instance, using KNN in conjunction with decision trees or support vector machines (SVMs) allows the strengths of each model to complement each other, resulting in improved performance on noisy datasets .

Data Augmentation and Synthetic Data Generation **(1)**Data Augmentation: Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) [6] and its variants have been adapted to augment training data and reduce the impact of noise. By generating synthetic examples that represent the underlying data distribution more accurately, these methods help KNN models generalize better.

(2) Generative Adversarial Networks (GANs) [12]: GANs have been used to create synthetic data that can augment training datasets. This approach can help in generating cleaner samples that dilute the noise's effect, providing more robust training data for KNN.

1.2. Limitations and Future Directions

Despite significant advancements, several challenges remain in developing noise-robust KNN algorithms: **(1)**Computational Complexity. Advanced methods like metric learning and neural network integration often come with increased computational costs, making them less suitable for large-scale applications without substantial resources [21]. **(2)**Parameter Sensitivity. Many of these techniques introduce additional hyperparameters [6], requiring careful tuning to achieve optimal performance. This process can be time-consuming and computationally expensive [5]. **(3)**Scalability. High-dimensional data and large datasets pose scalability challenges [22]. While dimensionality reduction and feature learning can mitigate some issues, ensuring scalability across various applications remains an ongoing challenge. **(4)**Generalization. Ensuring that noise-robust KNN methods generalize well across different datasets and noise conditions is crucial [21]. More research is needed to develop universally applicable solutions that perform consistently in diverse scenarios [6].

1.3. Research question

Given the advancements and current limitations in enhancing the noise robustness of KNN algorithms, a pertinent research question emerges: How can we develop an efficient and scalable noise-robust KNN algorithm that minimizes computational complexity and parameter sensitivity while ensuring high accuracy across diverse noise conditions? This question seeks to address the key challenges by integrating adaptive learning techniques and distance metrics [21]. The goal is to create a KNN variant that maintains robustness to noise without the extensive computational overhead and fine-tuning required by existing methods, ensuring consistent performance in various applications. To summarize, this paper provides the following contributions: We propose a method of combining PCA and Neural network to extract features and fit model with KNN regression using grid search method.

2. Related Work

The K-Nearest Neighbors (KNN) algorithm is widely recognized for its simplicity and effectiveness in various machine learning tasks. However, its performance significantly degrades in the presence of noise. Noise in the data can distort distance calculations, leading to several critical issues.

Misidentification of Neighbors. Noise can alter the true distances between data points, causing the algorithm to identify incorrect neighbors. This misidentification disrupts the fundamental premise of KNN, which relies on the assumption that nearby points are likely to share the same label. As highlighted by Dudani (1976) [10], the accuracy of KNN is heavily dependent on the correct identification of nearest neighbors, and noise can severely compromise this process .

Overfitting to Noise. In noisy environments, KNN tends to overfit the training data, capturing the noise rather than the underlying data distribution. This overfitting results in poor generalization to new, unseen data. Zhou and Huang (2007) [22] discuss how noise leads to overfitting in KNN and propose noise-tolerant paradigms to mitigate this issue .

Increased Variance in Predictions. Noise introduces variability in the dataset, which increases the variance of KNN predictions. As the level of noise rises, the predictions become more inconsistent and less reliable. This variability undermines the stability of the model, as evidenced by studies such as those by Friedman et al. [13], which examine the impact of noise on the predictive variance of KNN .

Computational Inefficiency. To counteract the effects of noise, practitioners often increase the number of neighbors **'k'**. However, this adjustment raises the computational complexity, as KNN's computational cost is directly related to the number of neighbors considered. This inefficiency is particularly problematic for large datasets, as noted by De Maesschalck et al. (2000) [8], who discuss the trade-offs between noise robustness and computational cost in KNN algorithms .

Several advanced methods have been proposed to address these challenges. Techniques such as noise filtering, adaptive distance metrics, and hybrid models combining KNN with other machine learning algorithms have shown promise. For example, Hodge and Austin (2004) [16] provide a comprehensive overview of noise filtering techniques that can preprocess data to improve KNN performance. Additionally, Belkin and Niyogi (2003) [1] introduce Laplacian Eigenmaps, a dimensionality reduction technique that preserves local data structures and enhances noise robustness in KNN.

3. Method

3.1. K-Nearest Neighbors (KNN) Algorithm

The K-Nearest Neighbors (KNN) algorithm is a simple yet effective non-parametric method used for regression and classification tasks [9]. The core idea of KNN is to predict the value or class of a sample based on the k closest training samples in the feature space. The prediction for a regression task is typically made by averaging the values of the k-nearest neighbors.

KNN Formula for Regression [9]:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{1}$$

where $\hat{y}$ is the predicted value, k is the number of nearest neighbors, and y_i are the values of the k-nearest neighbors.

Distance Metrics:

(1) Euclidean Distance: This formula calculates the straight-line distance between two points x and y in an n-dimensional space by summing the squared differences between corresponding components and then taking the square root of the sum [21].

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

(2) Manhattan Distance: This formula calculates the distance between two points by summing the absolute differences between corresponding components, essentially measuring how many units you have to travel along grid lines to go from one point to another [5].

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3}$$

Weighting Schemes:

(1) Uniform Weight: Each neighbor is weighted equally.

(2) Distance Weight: Neighbors closer to the query point are weighted more heavily [22], often using an inverse relationship:

$$w_i = \frac{1}{d(x, x_i)} \tag{4}$$

3.2. Principal Component Analysis (PCA)

PCA is a widely used dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional form while retaining most of the variability in the data. [17] By doing so, it reduces the complexity and noise in the dataset, making it easier to analyze and visualize.

$$Z = XW \tag{5}$$

where Z is the transformed dataset, X is the original dataset, and W is the matrix of principal components.

3.3. Neural Network Feature Extraction

To enhance the robustness of KNN against noise, we employed a neural network for feature extraction [11]. The neural network was trained to learn meaningful feature representations from the PCA-reduced data. This neural network, a simple feedforward neural network with one hidden layer, helps in capturing complex patterns in the data that PCA alone might not reveal. Neural Network Forward Pass:

$$h = \sigma(XW_1 + b_1) \tag{6}$$

$$\hat{y} = hW_2 + b_2 \tag{7}$$

where X is the input, W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors, σ is the activation function, and h is the hidden layer representation.

Dynamic Learning Rate Adjustment A dynamic learning rate adjustment was implemented using a scheduler to improve the training efficiency of the neural network [7]. The learning rate was reduced by a factor when the validation loss plateaued, facilitating better convergence [15].

3.4. Grid Search for Hyperparameter Tuning

Grid Search was used to optimize the hyperparameters of the KNN algorithm. This method systematically works through multiple combinations of parameter values, cross-validating each combination to determine which parameters provide the best performance [2].

Grid Search Formula:

$$\text{Score}(p_1, p_2, ..., p_n) = \frac{1}{k}\sum_{i=1}^{k} \text{CV}(p_1, p_2, ..., p_n) \tag{8}$$

where $p_1, p_2, ..., p_n$ are the hyperparameters and CV represents the cross-validation score.

4. Experiment

4.1. PCA Dimensionality Reduction

To prepare the data for our experiments, we first applied Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while retaining 95% of the variance. This step is crucial for mitigating the curse of dimensionality and enhancing the computational efficiency of subsequent analyses. PCA transforms the original features into a new set of uncorrelated features (principal components), which capture the most significant variances in the data .

4.2. Neural Network for Feature Extraction

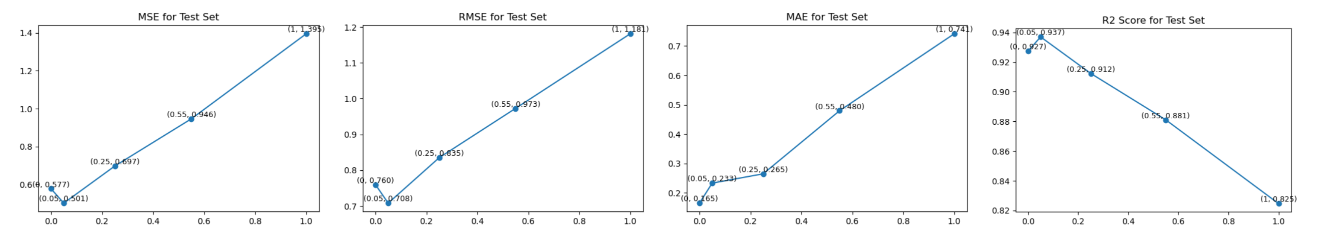
A simple feedforward neural network was constructed to learn meaningful features from the PCA-transformed data. The network architecture consisted of an input layer, one hidden layer with 50 neurons, and an output layer. The hidden layer used the ReLU activation function to introduce non-linearity. The network was trained using the Mean Squared Error (MSE) loss function and the Adam optimizer. To ensure robust training, we employed a dynamic learning rate adjustment strategy using the ReduceLROnPlateau scheduler, which reduces the learning rate by a factor of 0.5 if the validation loss does not improve for five consecutive epochs.

4.3. Noise Addition

To evaluate the robustness of the KNN algorithm to noise, we added Gaussian noise with varying levels **(0, 0.05, 0.25, 0.55, 1)** to the extracted features from the training data. Gaussian noise, also known as normal noise, is characterized by its bell-shaped probability density function, which is symmetric around the mean. It is defined mathematically by the probability density function:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9}$$

where μ is the mean and σ is the standard deviation. Gaussian noise is commonly used in simulations because it models many types of random processes found in nature, making it a practical choice for testing the robustness of algorithms against realistic noise contamination scenarios. The added noise was clipped to ensure it remained within the original feature range, thus avoiding unrealistic feature values. In Tab. 1, we



(a) First subfigure (b) Second subfigure (c) Third subfigure (d) Fourth subfigure

Figure 1. Curve plot of four quantitative indexes:MSE, RMSE, MAE and R2 Score.

Noise Level	MSE	RMSE	MAE	R² Score
0	0.577	0.759	0.165	0.927
0.05	0.501	0.707	0.233	0.936
0.25	0.696	0.834	0.264	0.912
0.55	0.946	0.972	0.479	0.881
1.00	1.394	1.181	0.741	0.825

Table 1. Comparison of performance metrics with varying noise levels. The value in red is the abnormal value of the decreasing trend of quantitative indexes. Model performance under noise level of 0.05 is better than under no noise.

can obvious the regression results of the dataset with different noise levels.

4.4. Hyperparameter Optimization

To determine the optimal hyperparameters for the KNN algorithm, including the number of neighbors, the weighting scheme, and the distance metric, we utilized Grid Search with 5-fold cross-validation. The performance was evaluated using the negative mean squared error metric. The Grid Search systematically evaluated each combination of parameters to identify the best-performing model.

4.5. Results Comparison

The performance of the KNN model was evaluated using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R² score. These metrics were calculated for each level of Gaussian noise added to the training data. The results in Tab. 1 show how the model's performance varies with different noise levels.

There is a clear trend where increasing levels of noise lead to higher MSE, RMSE, and MAE values, alongside a decreasing R² score for test sets in Fig. 1. This suggests that the model's ability to accurately predict outcomes diminishes as noise increases. However, it's interesting that under noise level of 0.05, performance of model is better than the condition under no noise. There are possible causes:

Regularization Effect. Slight noise may be similar to regularization, which helps prevent the model from overfitting to the training data [22]. By introducing some randomness, noise may allow the model to avoid learning accidental patterns in the data.

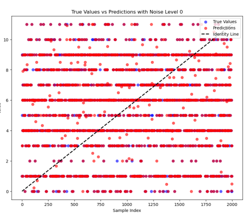
Data perturbation. In some cases, if the original data is too regular or has a certain pattern, slight noise may break this pattern [3], thus reducing the dependence of the model on a particular data structure.

Distribution of noise. If the distribution of the noise is Gaussian and similar to the distribution of the test data [14], the model is likely to perform better on data with small amounts of noise.

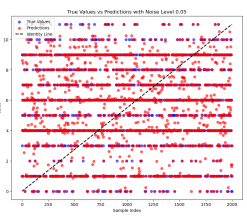
Reduced impact of outliers. In the absence of noise, outliers may have a large impact on the KNN model because KNN is sensitive to distance. Slight noise may help to reduce the effect of individual outliers [20].

Data Diversity. Noise may increase the diversity of the data and thus help the model learn more generalized features [4].

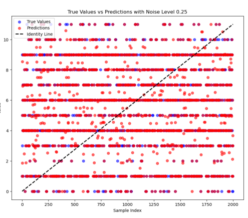
In Fig. 2, we can see the residual analysis plot between the predicted values of the model regression and the actual values in the dataset for different noise levels. We can see that the predicted values of the model fit the true values of the data very well at low noise levels. However, in the process of noise gradually, the deviation points of the predicted value are more and more, resulting in the fitting error of the model becoming larger.



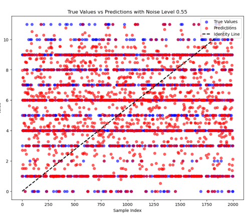
(a) First subfigure



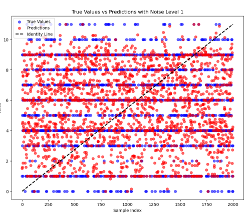
(b) Second subfigure



(c) Third subfigure



(d) Fourth subfigure



(e) Fourth subfigure

Figure 2. Residual plot of five different noise levels. The red points are the regression predictions and the blue points are real values.

Additionally, the performance metrics were plotted against the noise levels in Fig. 1, showing a steady increase in MSE, RMSE, and MAE, while R² scores for both training and test sets steadily decreased. This visual representation corroborates the quantitative findings and underscores the sensitivity of the KNN model to noise in the data.

4.6. Comparison with State of the Arts (SoTA)

In [19], Itzik Mizrahi propose a model kNet, a deep learning model specifically designed to simulate the behavior of the K-Nearest Neighbor (kNN) classifier while solving the problems of large memory footprint and time-consuming retrieval of kNN when dealing with large-scale data.

At the same time, in terms of generalization ability, kNet, as a deep learning model, may be better than the traditional model in the experiment in generalization ability, because its robustness to label noise is considered in its design, especially when the value of k is large.

However, in terms of computational efficiency, kNet, as a neural network, is faster and more deterministic at inference time, while the model in the experiment may take a long time to retrieve the nearest neighbors, especially on large datasets.

In terms of noise handling, kNet is optimized specifically for label noise and shows better robustness, while the models in the experiments are sensitive to noise, especially when the label noise is large, the performance drops significantly.

The model in the experiment can deal with noise and generalize well when dealing with small data sets, and the computational complexity is simple and does not need to consume more computing resources. However, kNet can make more accurate prediction for larger data sets and better eliminate the influence of noise. However, the structure of kNet is very complex, and there are many parameters, so the consumption of computing resources is huge. The proposed model in this experiment is able to achieve partial kNet performance while consuming only a very small fraction of the computational resources, which proves the effectiveness of the model.

Noise Levels	0	0.05	0.25	0.55	1.00
Original	0.927	0.936	0.912	0.881	0.825
PCA to Stand	0.912	0.909	0.834	0.724	0.647
NN Grid to Cross	0.936	0.924	0.918	0.874	0.801
NN PCA to Stand	0.874	0.863	0.763	0.678	0.596

Table 2. Comparison of ablation study with different replacement to component of the original method. The table shows R² scores of different models under 5 different noise level. Mark the highest value in red and the second highest in blue

4.7. Ablation Study

To further understand the contributions of various components in our experimental setup, we conducted an ablation study by modifying key elements and comparing the performance of the resulting models. In Tab. 2, it provide insights into the importance of each component in the overall model architecture. The four ablation models considered were:

(1) Replacing PCA with Data Standardization

(2) Replacing Grid Search with Cross-Fold Validation and Removing Neural Network

(3) Removing Neural Network and Replacing PCA with Data Standardization

PCA vs Data Standardization: The model with

PCA dimensionality reduction outperformed the one with data standardization in all metrics. This indicates that PCA effectively captures essential data variance, contributing to better model performance.

Grid Search vs Cross-Fold Validation: Although both hyperparameter tuning methods yielded comparable results, the grid search approach slightly outperformed cross-fold validation. This suggests that grid search's exhaustive search may provide more optimal hyperparameter configurations [2].

Impact of Neural Network Feature Extraction: Removing the neural network and replacing PCA with data standardization resulted in the most significant performance drop. This highlights the critical role of neural network-extracted features and PCA in enhancing the model's robustness and accuracy.

5. Conclusion

This paper propose a modified version of KNN model. Under different noise levels, the dimensionality reduction of the data set is carried out by using neural network for feature extraction, and the best parameters are adaptively searched to achieve the best regression prediction effect. Experiments have successfully demonstrated the robustness of this model under different noise levels. At the same time, we compare our model to SoTA model kNet [19] with the result of sacrificing some model prediction accuracy and robustness to noise for a simpler structure and less computing power consumption, while achieving some of the performance of kNet. It proves that the model has a certain ability to resist noise and has a better performance than original KNN model.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 3

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, Feb 2012. 4, 7

[3] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995. 5

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

[5] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, 2000. 1, 2, 3

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. 1, 2

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 4

[8] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000. 3

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2 edition, 2001. 3

[10] S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):325–327, 1976. 3

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 4

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014. 2

[13] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. 3

[14] G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, 1993. 5

[15] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998. 4

[16] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004. 3

[17] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2 edition, 2002. 4

[18] Shengqiao Li, E James Harner, and Donald A Adjeroh. Random knn feature selection-a fast and stable alternative to random forests. *BMC bioinformatics*, 12:1–11, 2011. 2

[19] Itzik Mizrahi and Shai Avidan. knet: A deep knn network to handle label noise. *arXiv preprint arXiv:2107.09735*, 2021. 2, 6, 7

[20] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005. 5

[21] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, Feb 2009. 2, 3

[22] Z. H. Zhou and M. L. Zhang. Ensembles of multi-instance learners. In *European Conference on Machine Learning*, pages 492–502. Springer, Berlin, Heidelberg, 2007. 2, 3, 5