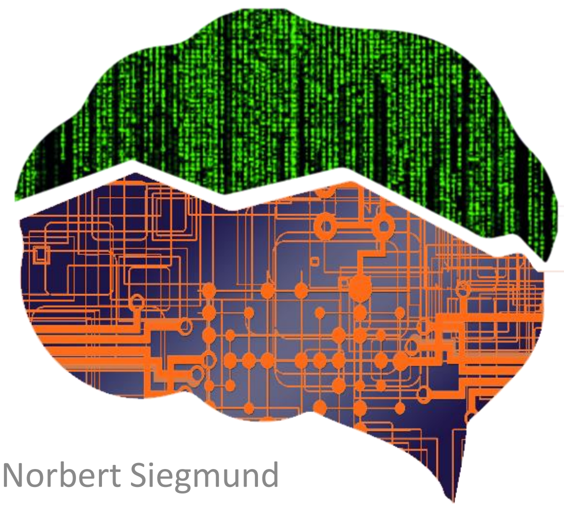
Machine Learning for Software Engineering

Dimensionality Reduction



Prof. Dr.-Ing. Norbert Siegmund

Intelligent Software Systems

Bauhaus-Universität Weimar

Exam Info

- Scheduled for Tuesday 25th of July
- 11-13h (same time as the lecture)
- Karl-Haußknecht Str. 7 (HK7)

- Project submissions are due Monday 17th of July
  - Submit your Name and Matrikelnummer (Student-ID) along
- Second pack of models (DIMACS format) are coming this week!

Recap I

- Constraint Satisfaction Problems
  - Variables with domains
  - Constraints:
    - Implicit (represented as code snippets)
    - Explicit (a set of all legal tuples)
    - Unary, binary, n-ary
  - Goal: Find any solution, which is a complete assignment of variables from their domains without breaking a constraint

- Backtracking
  - DFS + fixed order of variable assignment + constraint checking after each assignment

Recap II

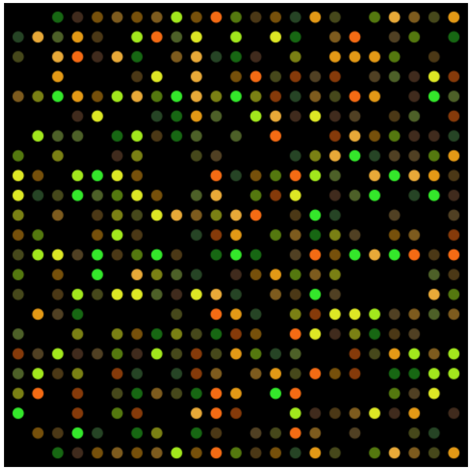
- Improvements of backtracking:
- Filtering:
  - Forward checking
  - Constraint propagation using arc consistency

- What is arc consistency?

- Ordering:
  - Which variable should be assigned next? (MRV)
  - In what order should its values be tried? (LCV)

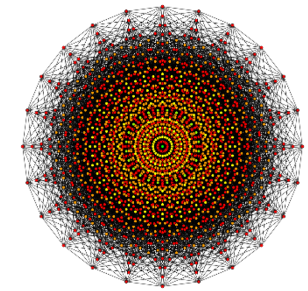
Curse of Dimensionality

- What is dimensionality?
  - Number of random variables in a dataset, also denoted as features in machine learning or columns in a csv file
- What is high dimensional data?
  - Microarrays (genes) with >20,000 features
  - Text with words as features
  - Images with pixels as features



- What is the curse?

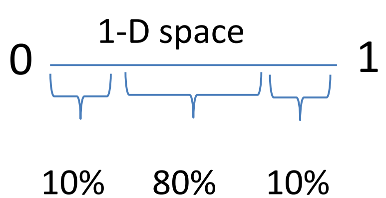
The Curse!



- The distance to the closest neighbor is nearly equal to the distance to any neighbor

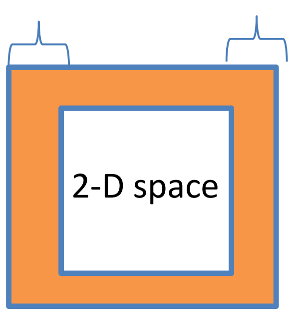
$$\lim_{d \to \infty} \frac{dist_{max} - dist_{min}}{dist_{min}} = 0$$

- Probability of data points being at the edge of the configuration space is exponentially increasing with the dimensions



$$P = 1^1 = 1$$
$$P_{inner} = 0.8^1 = 0.8$$
$$P_{outer} = 1 - 0.8 = 0.2$$

$$P = 1^3 = 1$$
$$P_{inner} = 0.8^3 = 0.512$$
$$P_{outer} = 1 - 0.512 = 0.488$$



$$P = 1^2 = 1$$
$$P_{inner} = 0.8^2 = 0.64$$
$$P_{outer} = 1 - 0.64 = 0.36$$

$$P = 1^{20} = 1$$
$$P_{inner} = 0.8^4 = 0.0115$$
$$P_{outer} = 1 - 0.0115 = 0.988$$

And The Curse Continues!

- Higher dimensions need exponentially more data points to draw any conclusions
  - 100 observations for 1-D space in the range 0 and 1, we get and good impression of the space of real numbers
  - 100 observations for 10-D space tell us nothing! We would need $100^{10}=10^{20}$ observations
- So, in higher dimensions the volume increases, such that all data points become sparse
  - Every distance increases
  - Every observation becomes dissimilar
  - This effect is especially strong when dimensions do not have an effect on the observation!

Do We Need all these Dimensions?

- NO! Often, only a subset of all features are relevant
  - Features might have **no effect** at all on observation
  - Features might strongly **correlate** with each other

- What means correlation?
  - Measure describing how much related two variables are
  - Described in the range -1 to 1
    - Positive correlation: If one variable increases, the other increases, too
    - Negative correlation: If one variable increase, the other decreases
  - The higher the absolute value, the higher the relation
  - Example: Predict fastest car based on two features: kW and horsepower -> both correlate with 1 -> only one is needed

How to Find the Actually Needed Features?

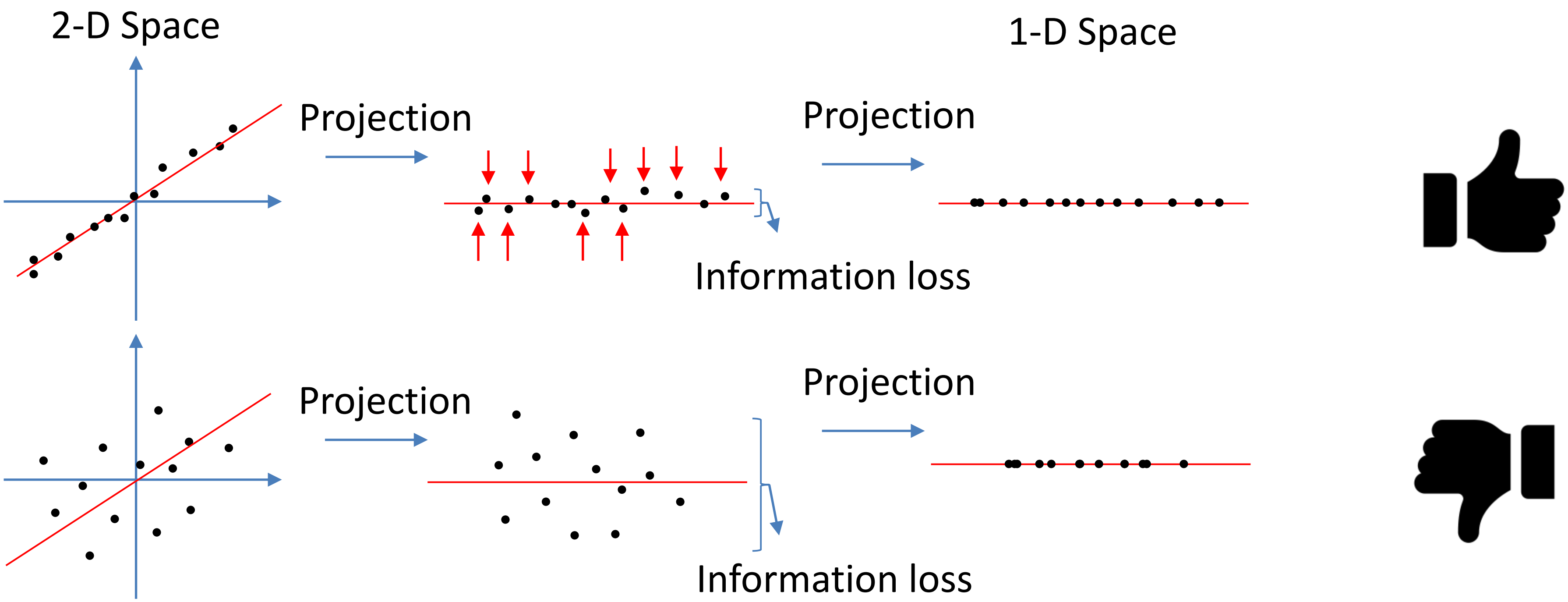
- **Feature extraction:** transformation of the data to a lower dimensionality with small loss of information
  - Principal component analysis (PCA) using a linear transformation
  - Kernel PCA using a kernel function
  - Autoencoder via neural networks
  - Linear discriminant analysis (LDA)
- **Feature selection:** techniques finding subsets of features mainly based on observations and what-if analyses
  - Filter (e.g., via information gain)
  - Wrapper (using search algorithms)
  - Embedded (during model building)

Principal Component Analysis

Goal of PCA

- Identify patterns in data to reduce the dimensionality of the dataset without sacrificing too much information
- Idea: Project the feature space to a smaller subspace that still represents our data good enough
  - PCA tries to find the features that correlate most
  - Highly correlating features can be combined such that the dimensionality can be reduced
- Approach: Find the dimensions of maximum variance and project the data onto a smaller dimensional space

PCA Visually Explained



1. Center data points around 0
2. Find eigenvector

3. Eigenvector minimizes least square error
4. Repeat step 2 with orthogonal vector for remaining dimensions

5. Keep principal components with highest variation and remove the least important PCs

PCA Algorithm Overview

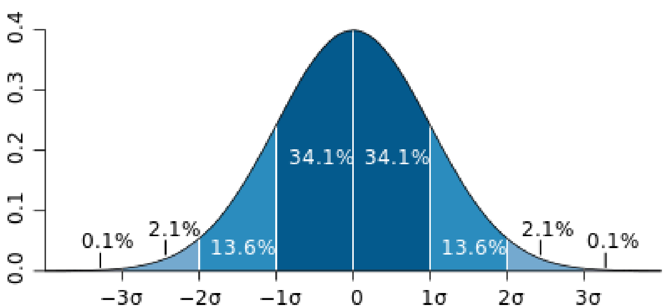
- Standardize the data: $x_i = x_i - E(x) = x_i - \frac{\sum_j^{||x||} x_j}{||x||}$
- Obtain the **eigenvectors** and **eigenvalues** from the **covariance matrix** or **correlation matrix** (alternatively applied Singular Vector Decomposition)
- Sort eigenvalues in descending order and select the k eigenvectors that correspond to the k largest eigenvalues
  - Where k is the number of features we want to keep
- Construct **projection matrix** W from the k eigenvectors
- Transform the original dataset x via W to obtain y (the k-dim subspace)

Preliminaries: Statistic I

- Mean: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
- Standard deviation: A measure of the spread of the data around the mean $s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}}$

Use n if you calculate the standard deviation of the whole population (i.e., when you have all possible data points).

- Variance: s^2
- What about more dimensions?



About 68% of the data are within the interval $[\bar{x} - s, \bar{x} + s]$

Preliminaries: Statistic II

- Covariance: Measure for describing the relationship between two dimensions (very similar to correlation)
  - $cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$
  - What is the covariance between a dimension and itself?
  - It is the variance!
  - $cov(x, x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{(n-1)} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)} = var(x) = s^2$
  - So, when both variables behave in a linked way (e.g., when x increases, y increases also), we can observe this
- Correlation: $corr(x, y) = \frac{cov(x,y)}{\sqrt{var(x) * var(y)}}$

Preliminaries: Statistic III

- Covariance of n-dimensional data points
  - Since covariance is a pair-wise measure, we have to compute the covariance of all pairs of dimensions
  - $C^{n \times n} = \left(c_{i,j} \mid c_{i,j} = cov(Dim_i, Dim_j)\right)$
  - Example: $C^{3 \times 3} = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$
  - Properties:
    - $cov(x,y) = cov(y,x)$
    - $cov(x,x) = s^2(x)$
    - So, covariance matrix is symmetrical about the main diagonal

Preliminaries: Statistic IV

- What are eigenvectors?
  - An eigenvector is a vector when multiplied with a (transformation) matrix, it results in a vector that is a multiple of the original vector
  - The multiple is called the **eigenvalue**
  - Example:
    - $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 2*1+3*3 \\ 2*1+1*3 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$ no eigenvector
    - $\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} * \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 2*3+3*2 \\ 2*3+1*2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \underbrace{4 * \begin{pmatrix} 3 \\ 2 \end{pmatrix}}_{\text{eigenvector with 4 as eigenvalue}}$
  - Properties:
    - Only square matrices: $n \times n$ has no or n eigenvectors
    - All eigenvectors are orthogonal to each other

1. Step in PCA: Subtract the Mean

- PCA finds patterns in data to describe their similarity and differences
- When similar, we can reduce the corresponding dimensions
- Running example:

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

$\xrightarrow{\text{Subtract mean}}$

$\bar{x} = 1.81$
$\bar{y} = 1.91$

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

2. Step in PCA: Calculate Covariance Matrix

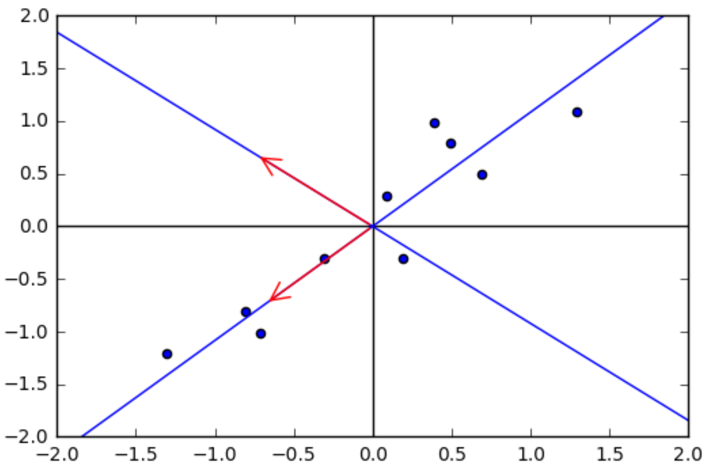
- $cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$
- $cov(x,x) = var(x) = s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}$
- $C(x,y) = \begin{pmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{pmatrix} =$

$$\begin{pmatrix} \frac{\sum_{i=1}^{10}(x_i - 1.81)^2}{9} & \frac{\sum_{i=1}^{10}(x_i - 1.81)(y_i - 1.91)}{9} \\ \frac{\sum_{i=1}^{10}(y_i - 1.91)(x_i - 1.81)}{9} & \frac{\sum_{i=1}^{10}(y_i - 1.91)^2}{9} \end{pmatrix} =$$

$$\begin{pmatrix} 0.61655556 & 0.61544444 \\ 0.61544444 & 0.71655556 \end{pmatrix}$$

3.Step in PCA: Calculate the eigenvectors and eigenvalues

- How to compute these is out of scope here…
- Eigenvalues= $\begin{pmatrix} 0.0490834 \\ 1.28402771 \end{pmatrix}$
- Eigenvectors= $\begin{pmatrix} -0.73517866 & -0.6778734 \\ 0.6778734 & -0.73517866 \end{pmatrix}$
- Vectors are *unit vectors*, meaning that there lengths are both normalized to 1



4.Step: Choose Components

- Approach:
  - Order eigenvalues from highest to lowest
  - Take only components with the largest eigenvalue as they contain the most information
  - If you want to remove k dimensions, remove the k dimensions with the lowest eigenvalues
- Eigenvector of the corresponding eigenvalue is the principle component

- Next, build a feature vector (which is a matrix!) using the eigenvectors that we keep, where each eigenvector is a column in the matrix

Feature Vector = Reduction Step

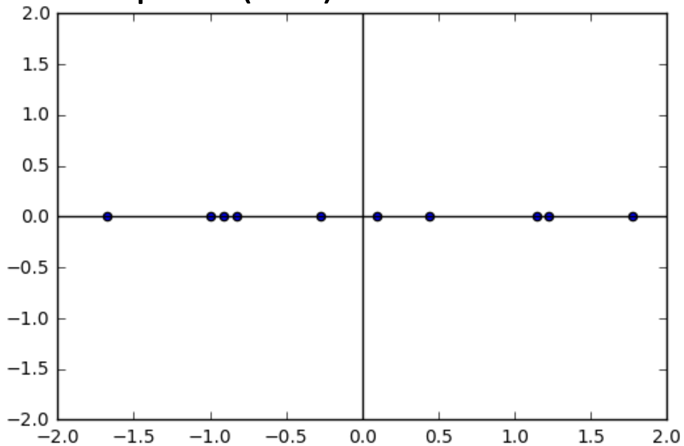
- $FeatureVector = (eigenvector_1, \dots, eigenvector_n)$
  - Where n is the number of dimensions that remain after PCA
  - And all eigenvectors were previously sorted according to their eigenvalues
- Example:
  - Eigenvalues= $\begin{pmatrix} 0.0490834 \\ 1.28402771 \end{pmatrix}$ Highest eigenvalue, so keep this vector
  - Eigenvectors= $\begin{pmatrix} -0.73517866 & -0.6778734 \\ 0.6778734 & -0.73517866 \end{pmatrix}$
  - FeatureVector= $\begin{pmatrix} -0.6778734 \\ -0.73517866 \end{pmatrix}$

5.Step: Transforming the Data

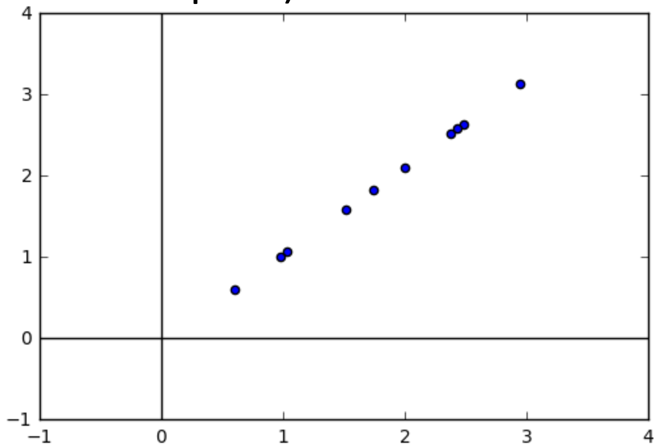
- To transform the data to the reduced sub space:
  - Transpose the feature vector and multiply it (on the left) with the transposed adjusted data set
  - $FinalData = RowFeatureVector \times RowDataAdjusted = FeatureVector^T \times DataAdjusted^T$
  - $FinalData$ is composed based on one dimension per row and the data points in the columns
- What is actually shown by this matrix?
  - Our original data, described solely by the vectors we selected
  - Originally, the data was expressed by the axes x and y (or z if we had 3-D data)
  - Now, our data is expressed by eigenvector 1, 2, etc.

After Transformation

Reduced dimensionality in the transformed and rotated subspace (1-D)



Scale back to the original coordination system (1-D data in 2-D space)



$$FinalData = RowFeatureVector \times RowDataAdjusted$$
$$RowDataAdjusted = RowFeatureVector^{-1} \times FinalData$$

If we take all principle components / feature vectors, then

$$RowFeatureVector^{-1} = RowFeatureVector^{T}$$
$$RowDataAdjusted = RowFeatureVector^{T} \times FinalData$$
$$RowOriginalData = (RowFeatureVector^{T} \times FinalData) + OriginalMean$$

Open Questions

- How to interpret the eigenvectors?
  – Where is the correlation between dimensions/variables?

- Where to define the line between components to keep and components that can be removed?
  – How much variance / information do I want to keep?

- Can we use the principal component scores in further analyses?
  – What are the limitations of this technique?

A More Complex Example

Place Rated Almanac
(Boyer and Savageau)

Rating 329 communities based on 9 criteria

- Climate and Terrain
- Housing
- Health Care & Environment
- Crime
- Transportation
- Education
- The Arts
- Recreation
- Economics

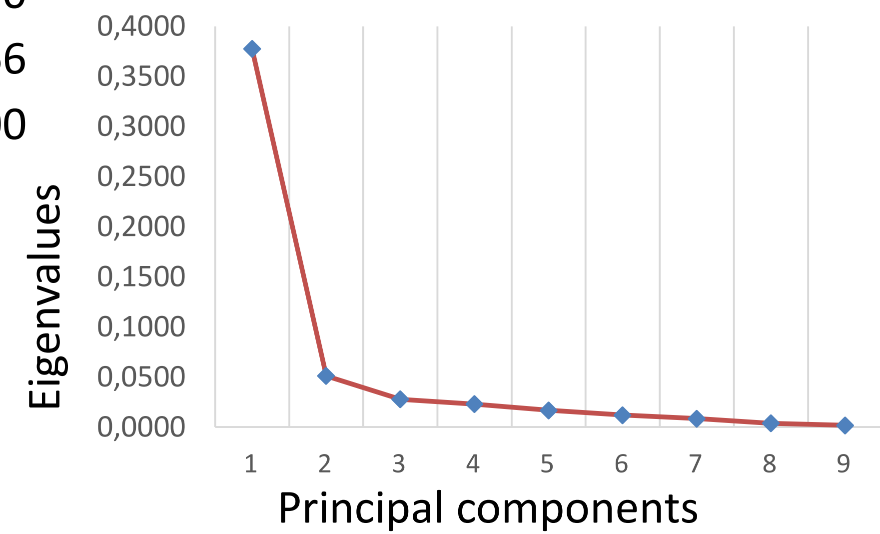
521	6200	237	923	4031	2757	996	1405	7633	1
575	8138	1656	886	4883	2438	5564	2632	4350	2
468	7339	618	970	2531	2560	237	859	5250	3
476	7908	1431	610	6883	3399	4655	1617	5864	4
659	8393	1853	1483	6558	3026	4496	2612	5727	5
520	5819	640	727	2444	2972	334	1018	5254	6
559	8288	621	514	2881	3144	2333	1117	5097	7
537	6487	965	706	4975	2945	1487	1280	5795	8
561	6191	432	399	4246	2778	256	1210	4230	9
609	6546	669	1073	4902	2852	1235	1109	6241	10
885	16047	2025	983	3954	2843	5632	3156	6220	11
195	12175	601	1223	5091	2414	2346	3000	7668	12
530	5704	580	878	2865	2469	430	838	3370	13
				…					
597	7927	1445	1115	4532	3112	4545	1923	6174	323
564	6858	1099	1423	2904	2876	1077	2668	5390	324
562	8715	1805	680	3643	3299	1784	910	5040	325
535	6440	317	1106	3731	2491	996	2140	4986	326
540	8371	713	440	2267	2903	1022	842	4946	327
570	7021	1097	938	3374	2920	2797	1327	3894	328
608	7875	212	1179	2768	2387	122	918	4694	329

Applying PCA on the Data

$= 0.3775/0.5225 = 72\%$ of variation explained

Component	Eigenvalue	Proportion	Cumulative
1	0.3775	0.7227	0.7227
2	0.0511	0.0977	0.8204
3	0.0279	0.0535	0.8739
4	0.0230	0.0440	0.9178
5	0.0168	0.0321	0.9500
6	0.0120	0.0229	0.9728
7	0.0085	0.0162	0.9890
8	0.0039	0.0075	0.9966
9	0.0018	0.0034	1.0000
Total	0.5225		

First 3 components explain 87% of the variation



Computing Principal Component Scores

- Remember: Eigenvalues are connected with eigenvectors
  - So, use the eigenvector of the largest eigenvalue to compute the principal component score for that component
- $Y_1 = 0.0351 \times (climate) + 0.0933 \times (housing) + 0.4078 \times (health) + 0.1004 \times (crime) + 0.1501 \times (transportation) + 0.0321 \times (education) 0.8743 \times (arts) + 0.1590 \times (recreation) + 0.0195 \times (economy)$
- Coefficients are the elements of the eigenvector of the first principal component
- Plug in the concrete values for the variables to obtain the value for each community

Interpreting Eigenvectors and Principal Components

	Principal Component		
Variable	**1**	**2**	**3**
Climate	0.190	0.017	0.207
Housing	**0.544**	0.020	0.204
Health	**0.782**	**-0.605**	0.144
Crime	0.365	0.294	**0.585**
Transportation	**0.585**	0.085	0.234
Education	0.394	-0.273	0.027
Arts	**0.985**	0.126	-0.111
Recreation	**0.520**	0.402	**0.519**
Economy	0.142	0.150	0.239

Compute the correlations between the original data for each variable and each principal component

PC1: correlation with 5 variables: if house, health, transportation, arts, and recreation increases, so does the PC1. So, these five variables vary together

PC2: if health decreases, PC1 will increase. Measure of how unhealthy a location is

PC3: correlation with crime and recreation: locations with high crime have higher recreation facilities

PCA in Software Engineering

Measuring Programming Experience

Janet Feigenspan, University of Magdeburg
Christian Kästner, Philipps University Marburg
Jörg Liebig and Sven Apel, University of Passau
Stefan Hanenberg, University of Duisburg-Essen

***Abstract*—Programming experience is an important confounding parameter in controlled experiments regarding program comprehension. In literature, ways to measure or control programming experience vary. Often, researchers neglect it or do not specify how they controlled it. We set out to find a well-defined understanding of programming experience and a way to measure it. From published comprehension experiments, we extracted questions that assess programming experience. In a controlled experiment, we compare the answers of 128 students to these questions with their performance in solving program-comprehension tasks. We found that self estimation seems to be a reliable way to measure programming experience. Furthermore, we applied exploratory factor analysis to extract a model of programming experience. With our analysis, we initiate a path toward measuring programming experience with a valid and reliable tool, so that we can control its influence on program comprehension.**

I. Introduction

In software-engineering experiments, program comprehension is frequently measured, for example, for the evaluation of programming-language constructs or software-development tools [3], [7], [13], [16], [26]. Program comprehension is an internal cognitive process that we cannot observe directly

timated programming experience compared to class mates and self estimated experience with logical programming. Furthermore, we present a five-factor model that describes programming experience using exploratory factor analysis. The contributions of this paper are the following:

- Literature review about the state of the art of measuring and controlling the influence of programming experience.
- A questionnaire that contains the common questions to measure programming experience.
- Reusable experimental design to evaluate the questionnaire.
- Initial evaluation of this questionnaire with sophomores.
- Proposal toward two relevant questions and a five-factor model of programming experience.

II. Literature Review

To get an overview of whether and how researchers measure programming experience, we conducted a literature review based on the guidelines for systematic literature reviews

Why PCA?

- Building a model for program comprehension
- Reducing the data

Source	Question	Scale	Abbreviation
Self estimation	On a scale from 1 to 10, how do you estimate your programming experience?	1: very inexperienced to 10: very experienced	s.PE
	How do you estimate your programming experience compared to experts with 20 years of practical experience?	1: very inexperienced to 5: very experienced	s.Experts
	How do you estimate your programming experience compared to your class mates?	1: very inexperienced to 5: very experienced	s.ClassMates
	How experienced are you with the following languages: Java/C/Haskell/Prolog	1: very inexperienced to 5: very experienced	s.Java/s.C/s.Haskell/ s.Prolog
	How many additional languages do you know (medium experience or better)?	Integer	s.NumLanguages
	How experienced are you with the following programming paradigms: functional/imperative/logical/object-oriented programming?	1: very inexperienced to 5: very experienced	s.Functional/s.Imperative/ s.Logical/s.ObjectOriented
Years	For how many years have you been programming?	Integer	y.Prog
	For how many years have you been programming for larger software projects, e.g., in a company?	Integer	y.ProgProf
Education	What year did you enroll at university?	Integer	e.Years
	How many courses did you take in which you had to implement source code?	Integer	e.Courses
Size	How large were the professional projects typically?	NA, <900, 900-40000, >40000	z.Size
Other	How old are you?	Integer	o.Age

Integer: Answer is an integer; Nominal: Answer is a string. The abbreviation of each question encodes also the category to which it belongs.

TABLE I
OVERVIEW OF QUESTIONS TO ASSESS PROGRAMMING-EXPERIENCE.

Data Analysis

- Programming tasks were solved by >120 students
- What did the students answer in the questionnaire that had answered the tasks correctly?
- See correlations table

No.	Question	ρ	N
1	s.PE	.539	70
2	s.Experts	.292	126
3	s.ClassMates	.403	127
4	s.Java	.277	124
5	s.C	.057	127
6	s.Hasekll	.252	128
7	s.Prolog	.186	128
8	s.NumLanguages	.182	118
9	s.Functional	.238	127
10	s.Imperative	.244	128
11	s.Logical	.128	126
12	s.ObjectOriented	.354	127
13	y.Prog	.359	123
14	y.ProgProf	.004	127
15	e.Years	-.058	126
16	e.Courses	.135	123
17	z.Size	-.108	128
18	o.Age	-.116	128

ρ: Spearman correlation; N: number of subjects; gray cells denote significant correlations ($p < .05$).

TABLE IV
SPEARMAN CORRELATIONS OF NUMBER OF CORRECT ANSWERS WITH ANSWERS IN QUESTIONNAIRE.

No.	Question	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Number of subjects
1	s.PE	-.279	-.417	-.042	.004	-.002	.016	.014	-.182	.071	.085	68 – 27
2	s.Experts	-.300	-.177	.047	-.026	.006	-.075	-.217	-.004	.206	.131	122 – 40
3	s.ClassMates	-.189	-.401	-.084	-.065	-.053	-.059	-.163	-.061	.161	.100	123 – 40
4	s.Java	.029	-.066	-.154	-.022	-.066	.003	-.040	.145	-.170	-.222	105 – 34
5	s.C	-.175	-.124	.018	.027	.126	-.108	-.056	-.052	.043	.108	123 – 40
6	s.Haskell	-.171	-.109	-.144	-.113	-.014	-.216	-.153	-.183	.019	.158	124 – 40
7	s.Prolog	-.174	-.141	-.079	-.104	-.027	-.039	.076	-.239	-.047	.146	124 – 40
8	s.NumLanguages	-.295	-.339	-.131	-.121	-.027	-.103	-.035	-.090	.232	.168	115 – 34
9	s.Functional	-.148	-.150	-.150	-.004	-.017	-.204	-.120	-.217	.027	.175	123 – 40
10	s.Imperative	-.283	.331	-.033	-.089	-.06	-.129	-.296	-.156	.126	.043	124 – 40
11	s.Logical	-.209	-.105	-.158	-.136	-.022	-.014	.058	-.257	-.191	.108	122 – 40
12	s.ObjectOriented	-.084	-.232	-.008	.012	-.093	-.034	.025	-.060	.156	.082	123 – 40
13	y.Prog	-.241	-.379	-.144	-.071	.010	-.113	-.258	-.159	.273	.180	120 – 38
14	y.ProgProf	-.217	-.196	-.012	-.119	-.130	.071	-.274	-.022	.044	-.010	123 – 39
15	e.Years	-.032	.001	.018	-.152	.059	.047	-.119	.037	-.092	-.173	122 – 40
16	e.Courses	-.146	-.088	-.040	-.062	.071	.028	-.053	-.004	.268	.058	120 – 38
17	z.Size	-.155	-.160	-.057	-.134	.059	.003	-.201	.046	.000	-.023	124 – 40
18	o.Age	.036	.014	.110	.082	.131	.102	-.081	.090	.059	.010	124 – 40

Gray cells denote significant correlations ($p < .05$).

TABLE V
SPEARMAN CORRELATIONS OF RESPONSE TIMES FOR EACH TASK WITH ANSWERS IN QUESTIONNAIRE.

- 28 out of 180 were significant

PCA for Finding Latent Factors

Variable	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
s.C	.723				
s.ObjectOriented	.700			.403	
s.Imperative	.673	.333		.303	
s.Experts	.600	.326			
s.Java	.540		.427		
y.ProgProf		.859			
z.Size		.764			
s.NumLanguages	.335	.489		.403	
s.ClassMates		.449	.403	.424	
s.Functional			.880		
s.Haskell			.879		
e.Courses				.795	
e.Years			-.460	.573	
y.Prog		.493		.554	
s.Logical					.905
s.Prolog					.883

Gray cells denote main factor loadings.

TABLE VII
FACTOR LOADINGS OF VARIABLES IN QUESTIONNAIRE.

Experience with mainstream prog. languages

Courses are taught at the university. Paradigms and languages make sense to correlate.

Professional experience

The longer a subject is programming, the larger the projects are and the more languages he/she encountered.

Functional and logical programming

The paradigms and the corresponding languages correlate.

Experience from education

Courses, years, and languages indicate the experience gained by education.

Feature (Subset) Selection

Goal of Feature Selection

- Find a minimal subset of features (variables, etc.) that represents the data without (substantial) information loss such that it is sufficient for data analysis, machine learning, etc.

- How to select the subset?
  - PCA: Use variance of the data in an unsupervised fashion
  - Feature Selection: Use a predictor (e.g., via information gain)

- Idea: Throw away features that will not influence a dependent variable (observation, prediction) -> supervised learning

Three Objectives for Feature Selection

- The subset with a specified size that optimizes an evaluation criteria
- The subset of smaller size that satisfies a restriction on an evaluation criteria
- The subset with the best tradeoff between its size and its corresponding result of the evaluation criteria

- General: Improve a learner by learning speed, generalization error, or understanding
- Idea: differentiate between relevant, irrelevant, and redundant features

Types of Algorithms

- Continuous feature selection
  - Assignment of weights to each feature in such a way that the order corresponds to its theoretical relevance
- Binary feature selection
  - Assignment of binary weights, meaning filtering the set of features

- Type of problem affects learning algorithm
- There are 2^n potential subsets of features

- In essence, we have a search problem to find a suitable subset

Composition of an Algorithm

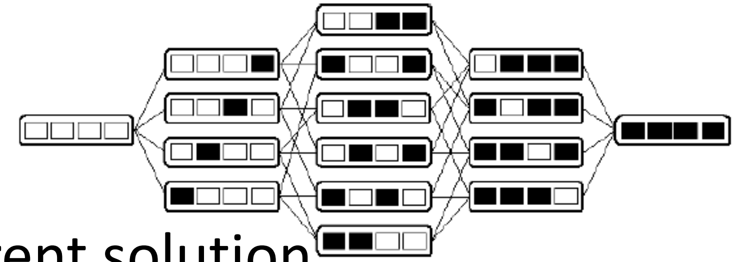
- Search organization
  – General strategy to explore the search space
- Generation of successors
  – Defines the successor state based on the current search state
- Evaluation measure
  – Mechanism by which successor candidate states are evaluated, allowing to decide where to go next

Search Organization

- Exponential search
  - Exhaustive algorithm that is guaranteed to find the optimal solution
- Sequential search
  - Selects exactly one candidate solution to be the successor state out of all possible candidate solutions
  - Iteratively searches the space (backward not possible) where the number of steps must be linear, but the complexity can be $O(n^{k+1})$, where k is the number of evaluated candidate solution at each step
- Random search
  - Randomness avoids local optima

Generation of Successors I

- Five operators are possible
  - Forward, backward, compound, weighting, and random
  - Operators modify the weights of the features



- Forward selection:
  - Operator adds a feature to the current solution
  - The feature must improve the evaluation measure
  - Linear in number of steps
  - Cannot account for interactions (e.g., if two features individually do not improve the evaluation measure, but do so when combined, they will never be selected)

Generation of Successors II

- Backward operator
  - Starts from the full set of features and removes in each step a single feature that does not degrade the evaluation more than a specified threshold
  - Also linear in effort, but is usually more cost intensive in practice as more features need to be considered in each step for computation
- Compound operator
  - Apply k consecutive steps forward and r steps backward, where $k > r$
  - Allows discovering also interactions of features

Generation of Successors III

- Weighting operator
  - The search space is continuous and all features are considered in a solution, but only to a certain degree
  - A successor candidate solution has a different weighting on the features
- Random operator
  - Used to generate potentially any other solution in a single step
  - Still the solution need to be better for the evaluation criteria

Evaluation Measures I

- Evaluate the fitness of a candidate solution
- Probability of error
  - Used when the learner is a classifier
  - Counts the number of falsely classified data based on the current feature set
- Divergence
  - Goal is to have more diversely classified data
- Dependence
  - Quantifies how strongly a feature is associated with the to be predicted class (i.e., knowing the value of the feature, is it possible to predict the value of the class?)

Evaluation Measures II

- Information or uncertainty
  - Measures how much information do we gain when adding a feature (e.g., used in decision trees)
  - If all classes become equally probable, the information gain is minimal and the uncertainty (entropy) is max
- Inconsistency
  - Remove or avoid features that do not agree on classifying a data point to the same class

General Algorithm

$S \leftarrow \{data\ sample\}$ with features X
$candidates \leftarrow getInitialSet(X)$
$solution \leftarrow getBest\big(assessFitness(candidates, S)\big)$
repeat
  $candidates \leftarrow searchStrategy(candidates, successorOperator(solution), X)$
  $candidate \leftarrow getBest\big(assessFitness(candidates, S)\big)$
  if $fitness(candidate) > fitness(solution)$ **or** $(fitness(candidate) == fitness(solution)$ **and** $|candidate| < |solution|)$ **then**
    $solution \leftarrow candidate$
until stop criteria or out of time
return $solution$

Take Home Message:

- Dimensionality reduction is useful when too many dimensions complicate learning and understanding
- Unsupervised reduction via PCA
  - Removes correlated variables
  - Finds the latent components that are behind the data
- Supervised reduction via feature subset selection
  - Finds a subset of features that satisfies a certain evaluation criteria by assessing the fitness of intermediate solutions

Next Lecture

- Developing our own neuronal network

Literature

- http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
- Feature Selection Algorithms: A Survey and Experimental Evaluation
  - http://www.lsi.upc.edu/~belanche/Publications/OLDresearch/R02-62.pdf