Data Preparation for Data Mining

Lesson 6

Lesson 6 Overview

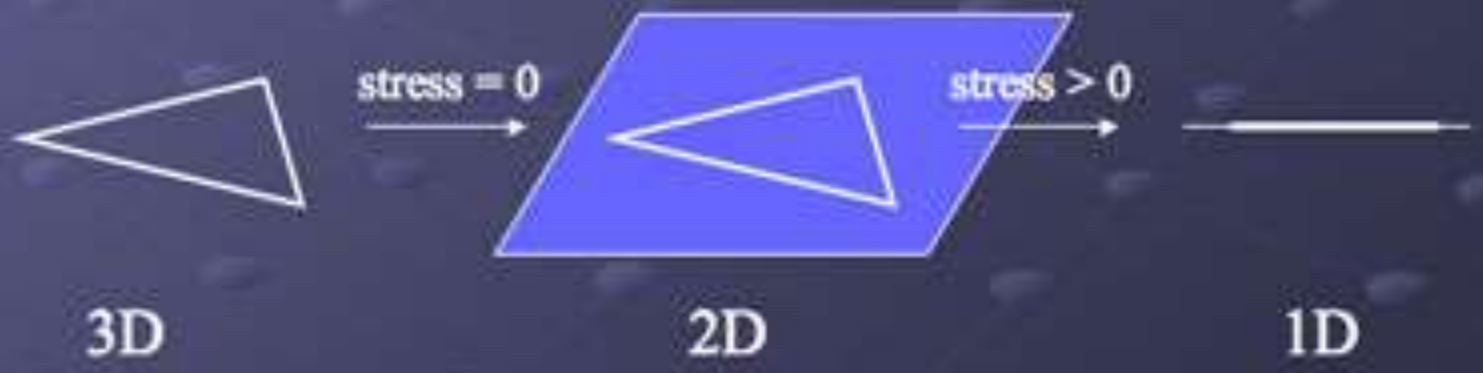
- Other Methods for Handling Alphas
  - Dimensionality Reduction
  - Multidimensional Scaling
- Let's Do It Ourselves Considerations
  - Preparing the Dataset
  - Hands-on exercises: Assignment III

Dimension reduction

- Some variables may be known to be correlated and do not need to be modeled
- Project data into fewer dimensions, avoiding significant loss of information
- Example: triangle

Dimension reduction

- Error measured by "stress" = change in perimeter of triangle
  - if no distortion occurs, stress = 0
- Generalization: change in the sum of distances between all pairs of points



Method suggested

- D_n := original data set , dim $D_n = n$, $k := n$
- Project D_k into D_{k-1} as follows:
  - rotate D_k randomly
  - project D_k into $k - 1$ dimensions and measure stress compared to D_n
  - repeat until required "degree of confidence" achieved
  - set D_{k-1} to be the projection causing the least stress
- Set $k := k - 1$ and repeat until stress gets too big

Dimension reduction: Questions

- Why not use principle component analysis (PCA)?
  - well understood, firm theoretical basis
    - minimizes the MSE
    - MSE more widely used than "stress"
  - no need for random iterations

Multidimensional Scaling

- Motivation
- Dissimilarity matrix
- Multidimensional scaling (MDS)
- Sammon's mapping
- Self-Organizing maps
- Comparison between MDS, Sammon's mapping, and SOM

Motivation

MDS attempts to

- Identify abstract variables which have generated the inter-object similarity measures
- Reduce the dimension of the data in a non-linear fashion
- Reproduce non-linear higher-dimensional structures on a lower-dimensional display

Dissimilarity Matrix

In MDS, the dissimilarities between every pair of observations are given

- Genuine distances (continuos data)
- Simple matching coefficients (categorical data)
- Scaled ranks (ordinal data)
- Gower's dissimilarity for mixed data

Multidimensional Scaling

Metric MDS:

- Distances between data items are given, a configuration of points which gives rise to those distances is sought
- Can be used for non-linear projection
- Objective function which is minimized:

$$E_M = \sum_{k,l} [d(k,l) - d'(k,l)]^2$$

Sammon's Mapping

- Closely related to metric MDS
- Tries to preserve pairwise distances
- Errors in distance preservation are normalized with the original distance
- Objective function:

$$E_S = \sum_{k \neq l} \frac{[d(k,l) - d'(k,l)]^2}{d(k,l)}$$

Self-Organizing Maps

- Algorithm that performs clustering and non-linear projection onto lower dimension at the same time
- Finds and orders a set of reference vectors located on a discrete lattice
- Learning rule:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]$$

- Objective function: $E_{SOM} = \sum_k \sum_i h_{ci} \left\| x_k - m_i \right\|$

(Discrete data, fixed neighborhood kernel)

Comparison Between MDS, Sammon's Mapping and SOM

- MDS tries to preserve the *metric* (ordering relations) of the original space, long distances dominate over the shorter ones
- SOM tries to preserve the *topology* (local neighborhood relations), items projected to nearby locations are similar
- Sammon's lies in the middle: it is like MDS but puts *more emphasis on small distances*

Next

- Let's Do It Ourselves Considerations
  - Preparing the Dataset
  - Hands-on exercises

Preparing the Data Set - Introduction

- focus now on the relationships between variables instead of values of a single variable
- goal: to make the information content most accessible to data mining tools
  - sparsely populated variables
  - problems with excessive dimensionality
  - determining, how many instances of data is needed
  - balancing the sample

Detailed Focus

-
- Using Sparsely Populated Variables
- Problems with High-Dimensional Data Sets
- Introducing the Neural Network
- Data Compression

Using Sparsely Populated Variables

- often discarded, threshold around 10-20% missing values
  - sometimes threshold has to be lowered to 80-90%
  - even this not always enough
    - sometimes only < 1% values present

Using Sparsely Populated Variables

- Why use them?
  - no other information sources available
- Example: a brokerage data set
  - over 700 variables
  - only few heavily populated
    - almost all below 10%
    - 1/2 under 2%, 1/3 under 1%
      - e.g. trades-in-corn-last-month

Using Sparsely Populated Variables

- the techniques introduced up to now inadequate to deal with sparse variables
  - example: the brokerage case, prediction of portfolio trading proclivity
    - Sparse variables excluded: correlation ~0.4
    - Standard methods: correlation under 0.5
    - Special sparse variable methods: over 0.7

Increasing Information Density

- missing value replacement not useful
  - single variable doesn't contain much information
  - working solution: collapse several variables into a single composite variable
    - numeric variables are reduced to categories
      - possibly grouping of several values ("binning")
    - categories may occur simultaneously

      => separate labels needed for every occurring combination

Binning of Sparse Numerical Variables

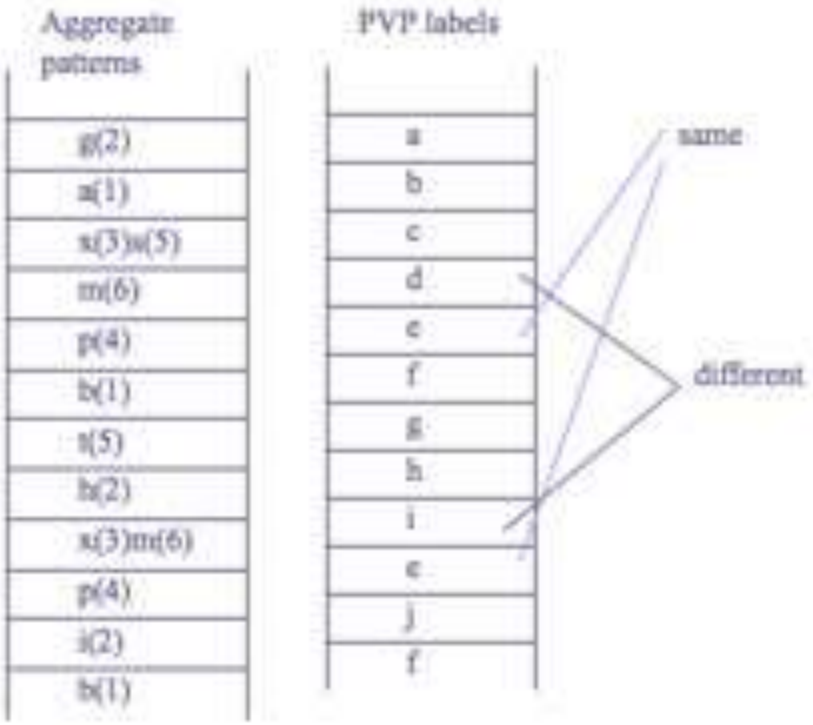
- divide range on variable to sub-ranges, assign a label (alpha) to each
  - example: coffee temperature classification (too hot, hot, mild, cool, cold)
  - bin boundaries, number of bins: domain knowledge needed
    - no rationale => equal counts to each bin

Present-Value Patterns (PVPs)

- Missing value patterns discussed earlier
  - PVPs almost reverse
  - nevertheless: the value in addition to its presence must be coded
  - every unique PVP gets an unique label
- PVPs used only for sparse variables
  - usually < 1% populated
  - (otherwise combinatorial explosion of # of labels)

Present-Value Patterns (PVPs)

Var 1	Var 2	Var 3	Var 4	Var 5	Var 6
	g				
a					
		x		s	
					m
			p		
b					
				t	
	h				
		x			m
			p		
	i				
b					



Present-Value Patterns (PVPs)

- too many PVPs (labels)
  - => divide to several variables
  - rule of thumb:
    - # of PVPs < 4 x # of sparse variables
- information is lost (e.g. in binning)
  - nevertheless: the remaining information can be used by a mining tool

Problems with High-Dimensional Data Sets

- dim. of state space ~ count of variables
  - computational load of mining tools grows rapidly w/ data dimension

    => comprehensive models no more possible
- volume of state space increases

  => low density sampling (=> e.g. overfitting)
  - in other words: explosion of number of possible state variable combinations

Problems with High-Dimensional Data Sets

- additional problem: nearly collinear variables very likely
  - problem to many mining tools

=> need to lower dimension

  - discarding variables poor solution
    - even then the question: which ones to discard ?

Information Representation

- much of the information in interrelationships between variables
- example: two variables carrying identical information
  - form may be different
- partial information sharing between variables
  - ex: height, weight, girth of a person
    - any two of them perfectly predict the third
    - 2D vs. 3D state space

Information Representation

- imperfect determination
  - noise
  - usually not interesting, increases dimensionality
- MDS (earlier in this lesson)
  - computationally intensive
    - but: all methods for reducing dim. are
    - all-or-nothing method
      - incremental method desired
    - specification: number of variables vs. confidence level

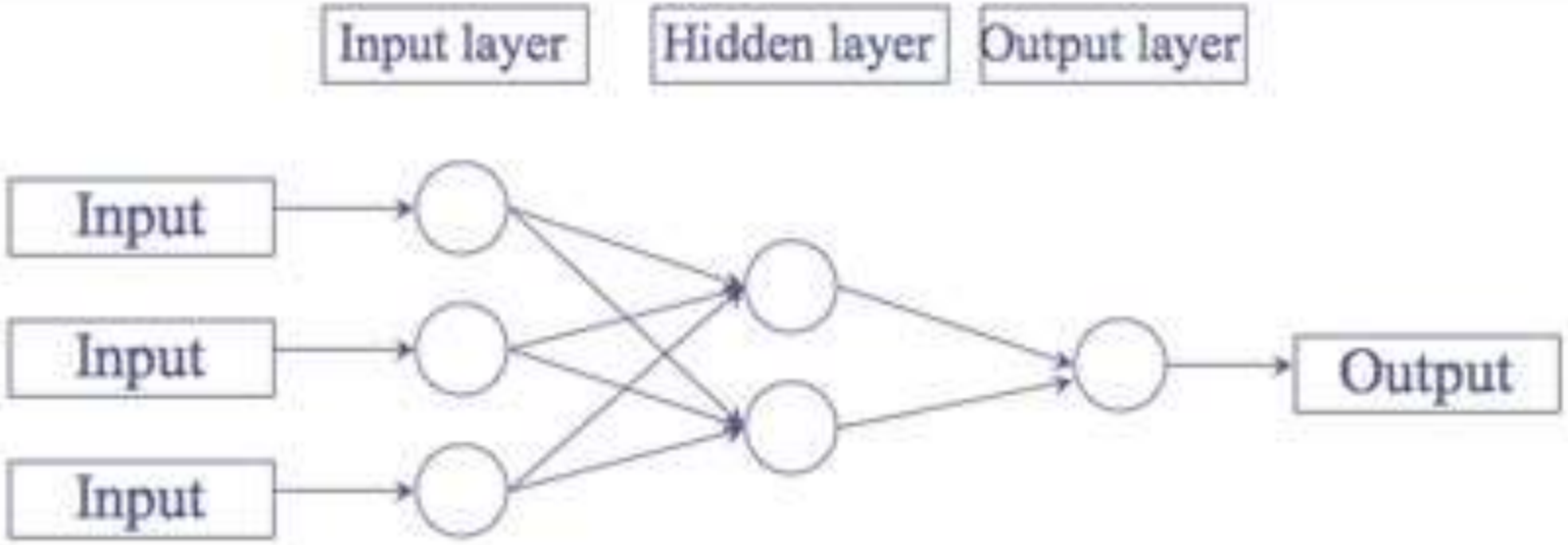
Representing High-Dimensional Data in Fewer Dimensions

- Principal Component Analysis and Factor Analysis work well for linear relationships
- PCA combines variables to orthogonal components
  - maximizes the variance of first components
  - components with small variability likely to be noise
    - even if not, benefits from dimension reduction may justify discarding these components

Introducing the Neural Network

- Goal:
  - squash data without destroying nonlinear relationships
  - choose least important data to discard
- standard tool, BP-ANN, is used
- idea: BP-ANN learns to associate input patterns with output patterns
- three layer MLP, full connectivity between layers

Introducing the Neural Network

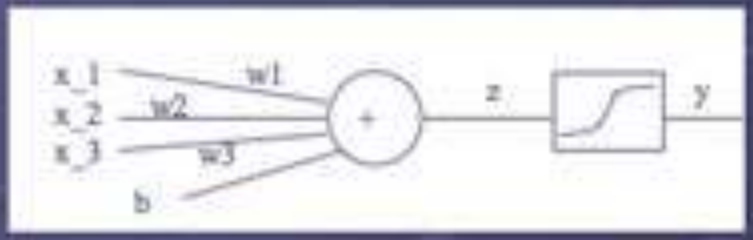


Training the Neural Network

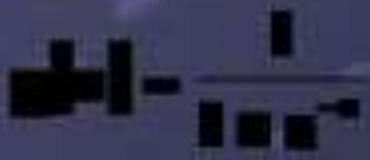
- two stages
  1. given inputs predict the output (forward pass)
  2. calculate the error between predicted and desired output, adjust network parameters to decrease the error
     - adjustment of weights only after each epoch
- after training only stage 1. is used
- any functional relationship between input and output can be learned

Neurons

- computational unit which replicates salient features of biological neurons
- neuron implements a nonlinear transfer function



- often g(z) is logistic function:



- two types of neuron weights:
  - input weights w
  - bias weight b

Networking Neurons to estimate a Function

- each neuron in hidden layer has a transition region in input space
  - there it implements a local mapping
    - elsewhere practically saturated
- each output layer unit nonlinearly combines the outputs of hidden layer to form a D to 1 mapping
  - separate output unit for each output variable

Network Learning

- unit weights must be initialized randomly
  - same weights would lead to learning of the same input space region by each neuron
    - avoided by sophisticated means in the learning algorithm
- network trained by set of data instances chosen by miner
  - epoch = go-through of the training set
- usually training takes several epochs
  - here error is accumulated during an epoch and weights are updated only after the whole epoch

Network Learning

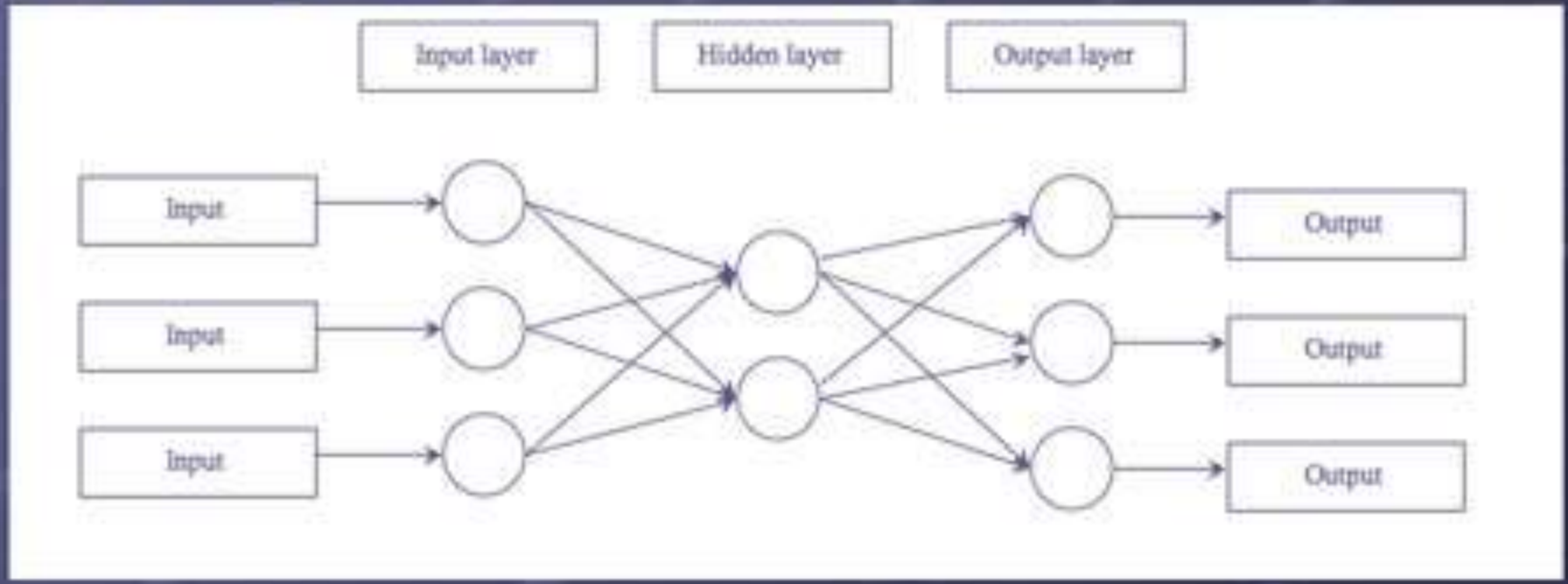
- training will (usually) never reach zero error
  - stopping condition has to be specified
    - target error level
    - maximum number of epochs
- often a separate test set is reserved
  - early stopping using test set error as criterion

Stochastic network performance

- with stochastic networks 100% good solution is not attainable
- often finding an exact solution to a problem is extremely difficult
  - ex: weekly coffee consumption
- often good approximate solution is enough
- stochastic techniques often work reasonably w/ incomplete and inaccurate inputs

Network architectures

- 1. The auto associative network
  - goal is to predict the input to the network based on the input
    - hidden layer forms a bottleneck



Network architectures

2. The Sparsely Connected network
   - # of connections = 2 x #(inputs) x #(hidden units)
   - most of the connections are usually not in use
     - need to prove occasionally during the training which connections are useful
   - graceful degradation of performance as connections are removed
     - typically 90% performance is retained if 10% of connections used

Compressing variables

- data compression tool we will use for the assignments is a sparsely connected auto associative neural network
- hidden layer must contain all the information needed to reconstruct the data
- hidden layer has less units than input layer

  => using hidden neurons as variables instead of input neurons results in dimensionality reduction

Using Compressed Dimensionality Data

- if the training data for compression is representative, compression works well
  - if the data to be compressed moves outside the training region, original values cannot be recovered from compressed
- great benefit from compression:
  - intractably large dimensionalities can be reduced and analyzed by the usual mining tools

Using Compressed Dimensionality Data

- in creating predictive model the variable(s) to be predicted must be kept out of compression
  - can be compressed separately if many of them
- compression very useful e.g. in modeling large industrial processes
  - lots of strongly redundant measurement values
- example: telecommunications
  - customer behavior changes only slowly
    => huge behavioral data sets

Preparing the Dataset - Continued

- Removing variables
  - What variables to remove?
  - SCANN
- How much data is enough?
  - Joint distribution
  - Joint variability
  - Degrees of freedom
- Beyond Joint distribution
  - Enhancing the data set

Removing Variables

- Use only as a last resort
- Ideal situation
  - Redundant variables
  - Highly nonlinear correlation
- Problems
  - Nonlinear correlation exists in some cases
    - Where to stop?
  - Single correlation or multiple correlation?
  - difficult task
- SCANN

Estimating Variable Importance Using SCANN

- Difficulties
  - Weights don't tell anything on themselves
    - High input weight -> low output weight
    - Low input weight -> high output weight
    - 2 weights up just for canceling each other
  - To REALLY understand SCANN all interactions between every neuron have to be evaluated
    - impossible
- Weights are important, but can't be directly analyzed

Using weights

- SCANN is relatively fast even for large data
  - if hidden layer is small
- Iteration starts with random weights
  - The algorithm always moves important weights
  - Unimportant variable weights are not moved during training
- Important weights are almost constant over training cycles
- Train network many times and compare results
  - Total distance weights have moved is a good measure of importance

Configuring and training the network

- Quality of final model is not important
- Network learning is important

Configuring and training the network

- Initially 5-10% hidden units of inputs
- Make sure pseudo-r^2 increases by 35-50%
- if it doesn't converge increase hidden layer units

Configuring and training the network

- Start training cycles that look for important variables
  - Train at least #inputs^0.6 cycles
- cut as many variables as needed up to 33%
- repeat if needed
- compress if possible
- do it again for rejected variables (test set)

Using the network

- By using these rules 7000+ variables were compressed to 700 while maintaining 90% confidence
  - 2000 variable collapsed to 5 as highly sparse
  - 5000 to 1500 trough reduction
  - 1500 to 700 by compression
- Test set produced almost identical results
- Better results than using domain experts

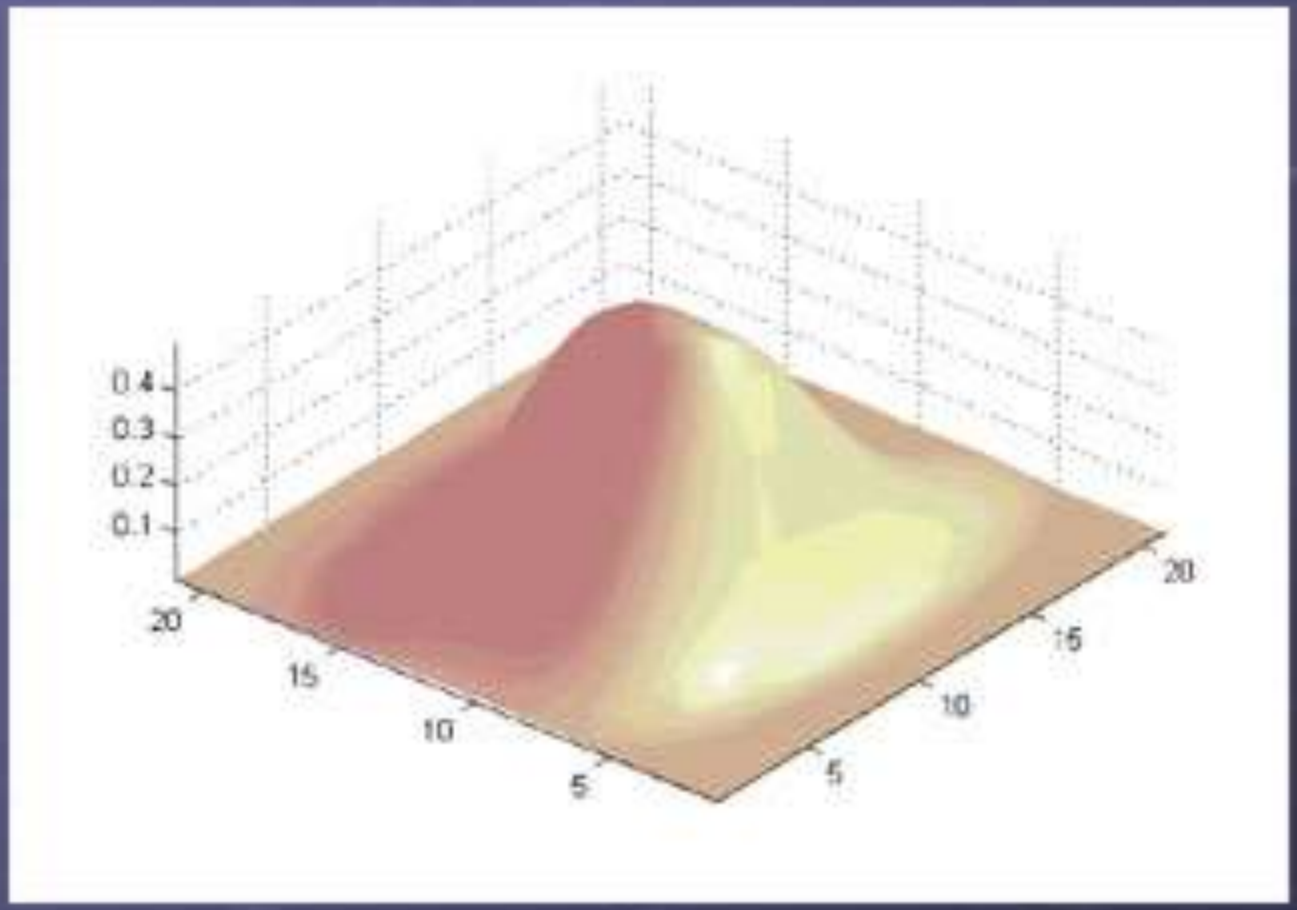
Using the network

- 1200 variables 6,000,000 records
- Reduced to 35 variables (no compression)
- The only effective model
  - includes 5 variables which are essential for the data set
  - without these variables other test models were not effective

How Much data is enough?

- Looking at distributions
- Joint distribution
- Joint variability
  - Density manifold stability
  - What probability to use?
    - Survey, not preparation
      - preparation changes variables
      - survey answers questions

Density Manifold



How Much data is enough?

- Degrees of freedom
  - Everything that can change
  - Can cause spurious patterns to appear
  - Use twice the preparation data in data mining

Beyond joint distribution

- Enhancing the dataset
  - Mailing problem
  - Chemical plant
- Feature enchantment
  - Plentiful data
    - Use representative sample of the problem
  - Limited data
    - Data multiplication
    - Add white noise
    - Add colored noise

Where Next?

- Data preparation is completed
- Data mining

Assignment III