FEATURES AND PREPROCESSING

Machine Learning for Autonomous Robots

Dr. Alexander Fabisch
DFKI, Robotics Innovation Center

October 25, 2022 – Bremen, Deutschland

University of Bremen

German Research Center for Artificial Intelligence

Features

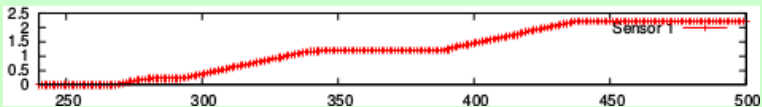
Features

Motivation

1. Data can be **simple and "ready to use"**.
2. More often available signals have to be **preprocessed** to be usable.
3. Or there is **no sensor existing** for the required information.
   Plus: later learning might need some help!

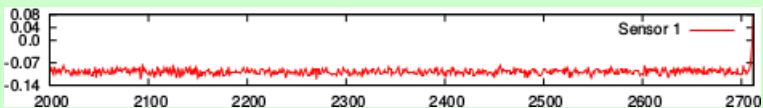
Examples

- Wheel speed measurement for a speed controller:



Examples

- Noisy signal of an Inertial Measurement Unit (IMU):



Features

Examples

- Sensor for
  “large rock in camera image”:

Spoiler:
This is called a “feature”!

What is a Feature?

- **In statistics:**
  Individual measurable property of the phenomena being observed.
- **In practice:**
  Input variable of a machine learning algorithm.

Vocabulary

Sample / instance

- Supervised learning: $(\boldsymbol{x}, y)$, $\boldsymbol{x} \in \mathbb{R}^m$
- Unsupervised learning: $\boldsymbol{x} \in \mathbb{R}^m$

$$\text{Feature vector} \rightarrow \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \begin{matrix} \leftarrow \text{Feature 1} \\ \leftarrow \text{Feature 2} \\ \\ \leftarrow \text{Feature } m \end{matrix}$$

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \leftarrow \text{One-hot encoding of a nominal feature}$$

Vocabulary

We often organize data sets / batches in a matrix:

$$\boldsymbol{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_m^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_m^{(n)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^{(1)\,T} \\ \boldsymbol{x}^{(2)\,T} \\ \vdots \\ \boldsymbol{x}^{(n)\,T} \end{pmatrix}$$

What is a Variable?

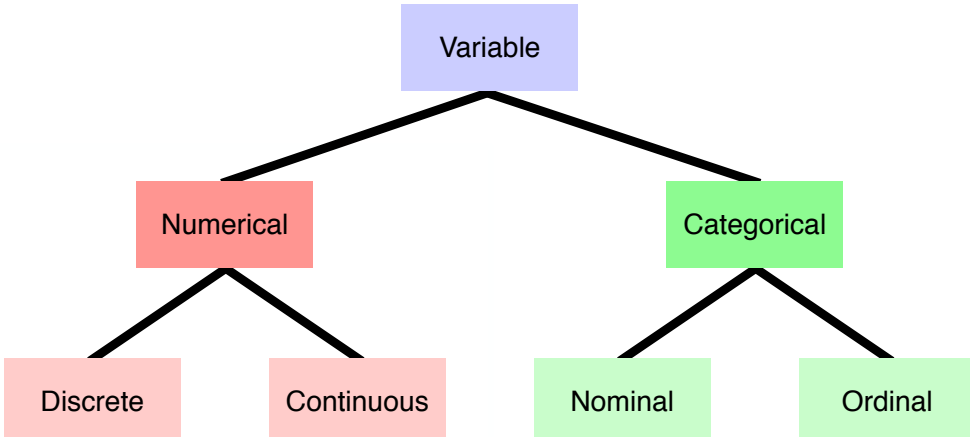
Any characteristic, number, or quantity that can be measured or counted

- Age (17, 39, 52, ...)
- Price (1.99$, 100 EUR, 1 000 000 EUR, ...)
- Vehicle vendor (Daimler, Volkswagen, Tesla...)

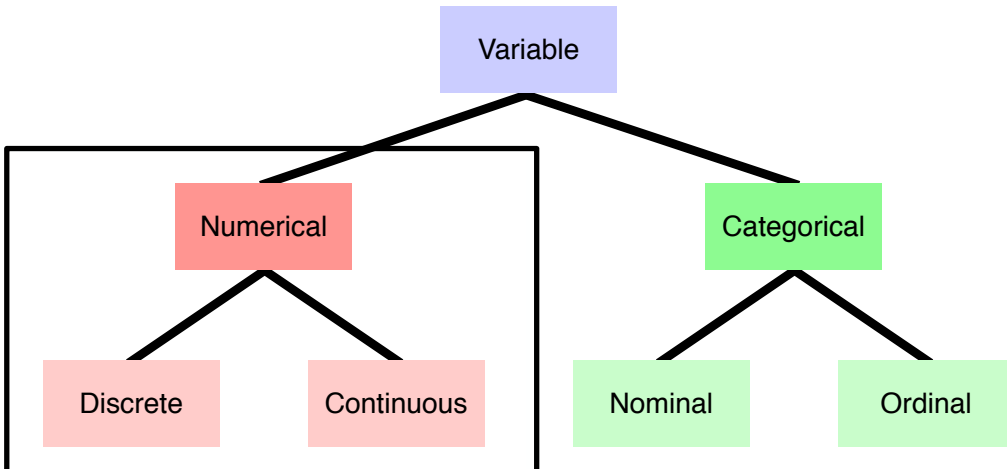
Variables in Robotics

- Temperature, current, voltage, ...
- Pixels in images (e.g., RGB)
- Objects in images (human, bottle, car ...)

Variable Types



Variable Types



Discrete Variables

Definition

A variable which values are integers or terms/words is called discrete

Examples

- Number of legs of a robot
- Number of people/bottles/cars/... in an image
- Number of events (e.g. Geiger counts) per unit of time
- ...

Continuous Variables

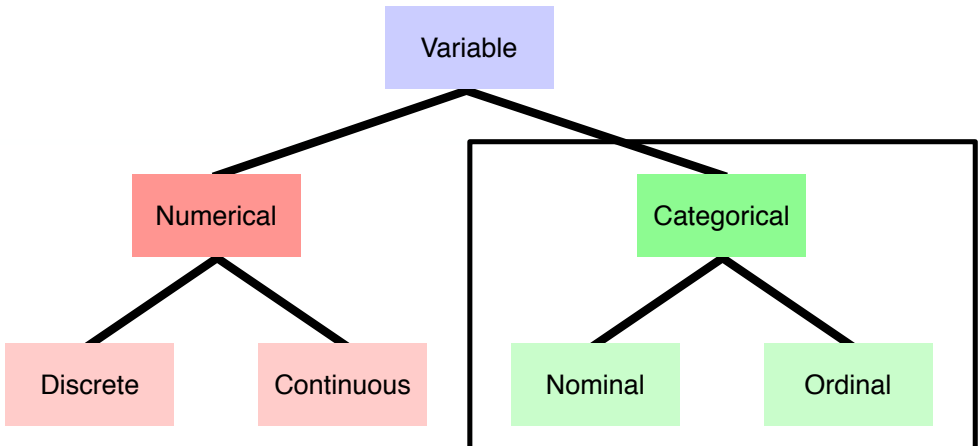
Definition

A variable that may contain any real/rational value within some range is called continuous

Examples

- Temperature
- Voltage
- Current
- Pressure
- ...

Variable Types



Ordinal Variables

Definition

A variable which values **can** be meaningful ordered

Examples

- Day of week
- Low, medium, high
- Age groups: 18-29, 30-49, 50-64, 65+
- ...

Nominal Variables

Definition

A variable which values **cannot** be meaningful ordered

Examples

- Mobile network provider (Vodafone, T-Mobile, ...)
- Object type (human, bottle, dog, . . .)
- Error codes
- . . .

Special cases

- Encoding of categories as numbers
- ID variables

Preprocessing

Preprocessing

What is Preprocessing?

1. "Preprocessing (PP)" and "Feature Generation (FG)" are often no separate method classes. FG can be seen as a PP method, sometimes.
2. In this lecture:
   PP: very general methods, no change of the dimensionality neither of the general behavior of the signal
   FG: (more) application-/task-specific, changes the behavior of the signal and/or its dimensionality.

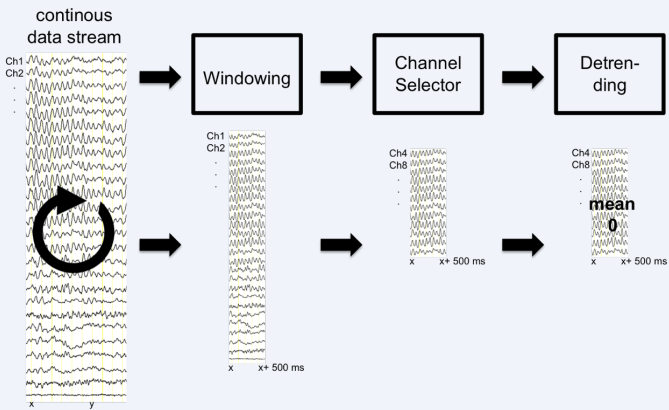
Problems with raw data

Typical problems

- Missing values for certain observations
- Nominal variables have to be transformed (cardinality?)
- Infrequent categories
- Outliers
- Noise

Example

EEG Data



Problems with raw data

Essential Preprocessing: Data Cleaning

Especially in robotics there might be problems of the sensor signal which need to be corrected before further processing. **Examples:**

- Missing samples/frames.
  Possible workaround: repetition of previous data, extrapolation based on previous data, delay and interpolation of previous and following data
- Asynchronous data channels.
  Possible workaround: buffering and alignment of data, inter-/extrapolation

Problems with raw data

Data/Feature Normalization

- Crucial for many learning algorithms
- Ensures comparability of components and samples
- Scale by norm: $x = \frac{x}{||x||_2}$ or $x = \frac{x}{||x||_{\max}}$
- Scale all features to interval $[0, 1]$: $x_i = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}$
- Standardize feature wise: $x_i = \frac{x_i - E[X]_i}{\sigma(X)_i}$

Feature Generation

Simple Features

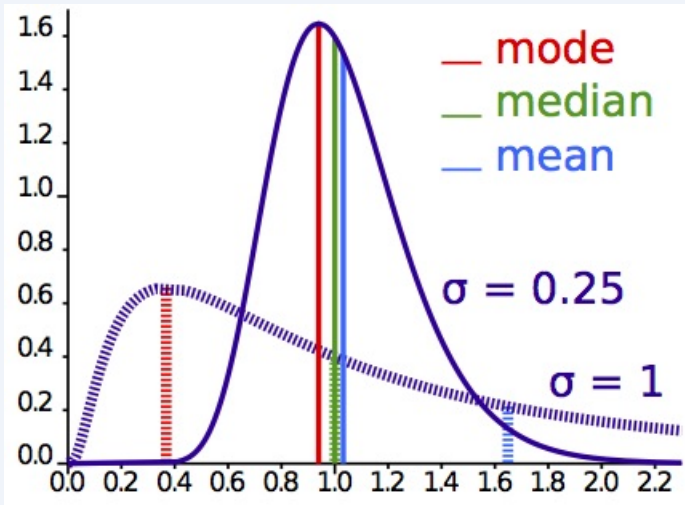
- Directly use data values (grayscale values, amplitudes)
- Thresholds (one dimensional or in image data)
- Sums, sum of squares
- Histogram, counts / bag of words (text documents, vision), tf-idf (term frequency-inverse document frequency)

"Pavlovsk Railing of bridge Yellow palace Winter", http://commons.wikimedia.org/wiki/

Statistical Features I

- A history of N samples needs to be buffered
- Mode: the value with most occurences
- Median: sort all values and take the middle
- Average / mean: $\mu = \frac{1}{n} \sum x_i$
- Variance
  - $Var(X) = E[(X - \mu)^2]$,
  - $(p_i = \frac{1}{n})$: $Var(X) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2$
  - Standard deviation: $\sigma = \sqrt{Var(X)}$
- Less frequently used as a feature
  - **skewness** (measure of asymmetry)
  - **kurtosis** (measure of the "peakedness")

Statistical Features II



"Comparison mean median mode" by Cmglee, http://commons.wikimedia.org/wiki/

Fourier Series

- Jean Baptiste Joseph Fourier: For any periodical (and integrable) function $f(t)$ a function series ("Fourier series") of sine/cosine functions can be developed
- General form ($\omega = 2\pi/T$):
  $f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cdot \cos(k\omega t) + b_k \cdot \sin(k\omega t))$
- Cancelation after n samples:
  $f_n(t) = \frac{a_0}{2} + \sum_{k=1}^{n}(a_k \cdot \cos(k\omega t) + b_k \cdot \sin(k\omega t))$
- In practice: we use a library like scipy to compute the Fourier transform

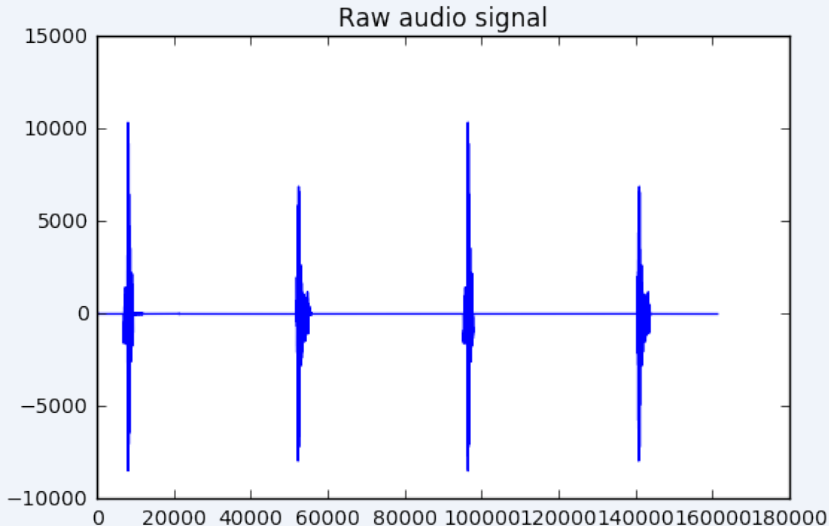
Example: Spectrum of Audio Signals

```
%matplotlib inline
from scipy.io import wavfile
rate, x = wavfile.read('test_song.wav')
plt.plot(x[:rate*15,0])
plt.title('Raw audio signal')
plt.show()
plt.specgram(x[:rate*15,0])
plt.title('Spectrogram')
```

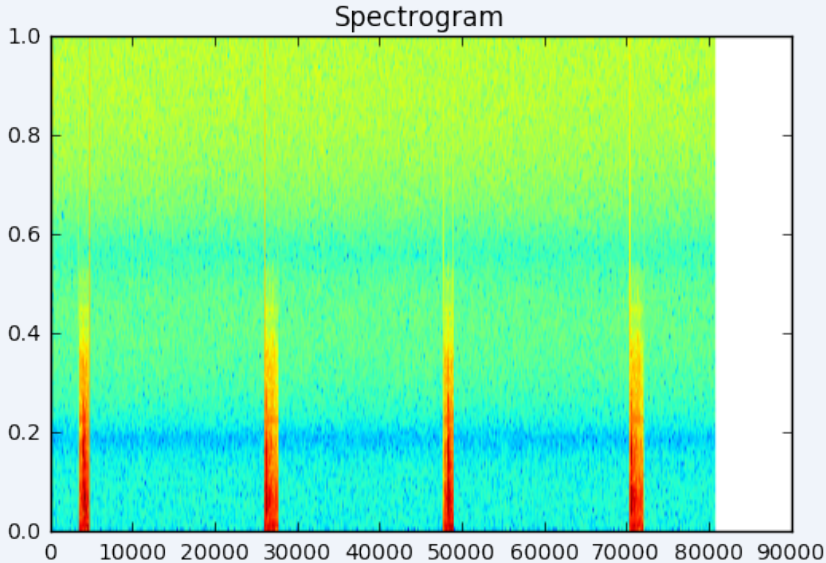
Result: spectrogram

- x-axis: time
- y-axis: frequency
- color: amplitude

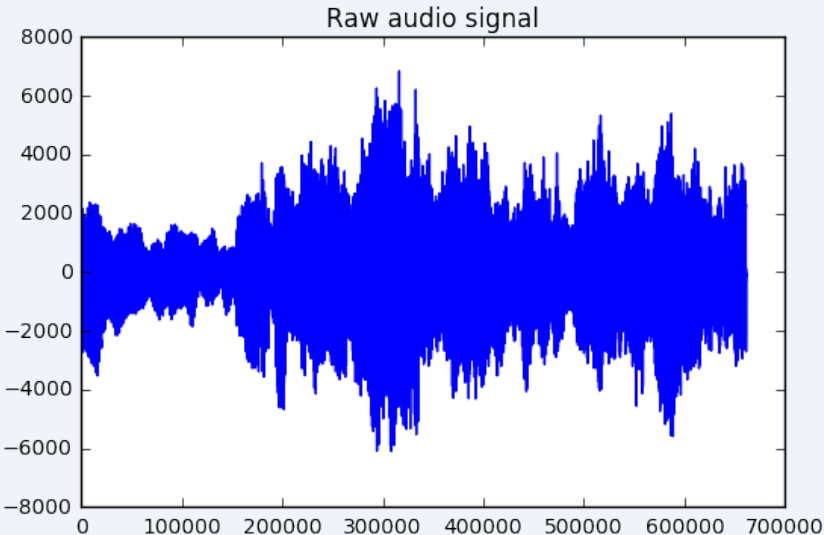
Example: Spectrum of Audio Signals



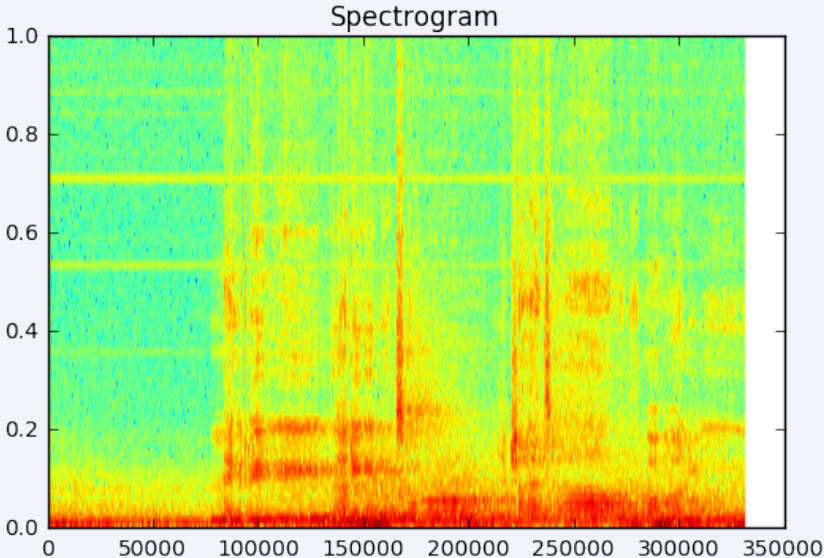
Example: Spectrum of Audio Signals



Example: Spectrum of Audio Signals



Example: Spectrum of Audio Signals



Overview of Visual Features

- In image processing:
  Feature = *interesting* part or local property in an image

  Methods:
- **Sobel Operator**
- Hough Transform
- SIFT and SURF
- ...

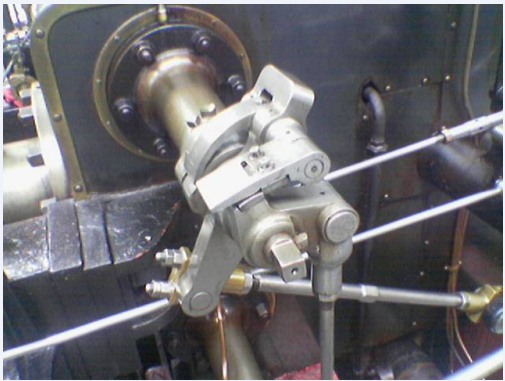
Sobel Operator

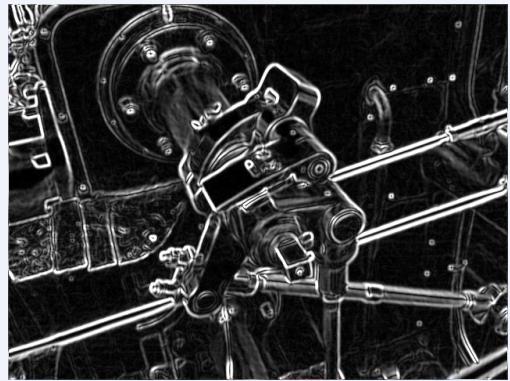
- Edge detection algorithm $\rightarrow$ image of emphasized edges/transitions
- Sobel Operators

$$S_x = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix} \text{ and } S_y = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

- Resulting images G_X, G_Y are a convolution of the Sobel operators with the original image A :
  $G_x = S_x * A \quad \text{and} \quad G_y = S_y * A$
- A direction independent result can be computed by the gradient magnitude
  $G = \sqrt{G_x^2 + G_y^2}$

Sobel Operator on Steam Engine Image

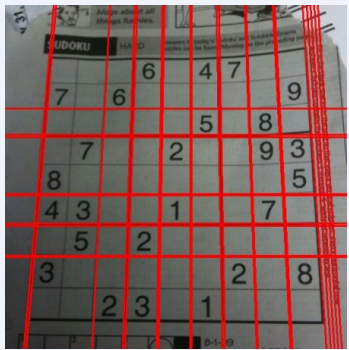




"Valve original (1)", "Valve sobel (3)" by Simpsons contributor, http://commons.wikimedia.org/wiki/File:Valve_original_(1).PNG, http://commons.wikimedia.org/wiki/File:Valve_sobel_(3).PNG

Further Visual Feature Generation Methods

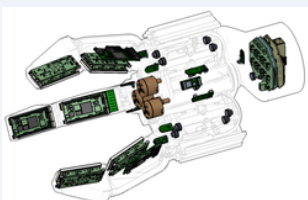
- Hough transform (parameter space of geometrical objects)
- SIFT (scale-invariant feature transform, US patent)
- SURF (Speeded Up Robust Features, faster than SIFT, US patent)
- ...

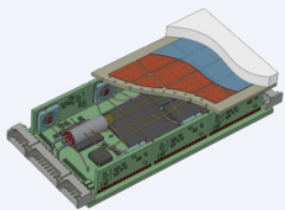


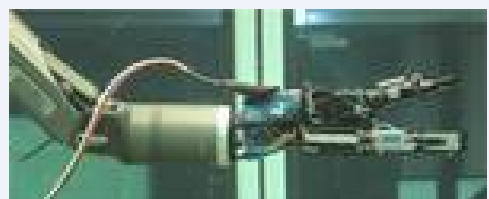
Feature Selection

Examples

- **Text categorization:** # of variables: 500,000, # of documents: 5,000 - 800,000 documents for research purposes
- **Computer vision:** # of variables: 3M, # of samples: 200,000
- **Seegrip Manipulator:** 2139 sensors







http://robotik.dfki-bremen.de/en/research/robot-systems/seegrip-manipulator.html

Motivation

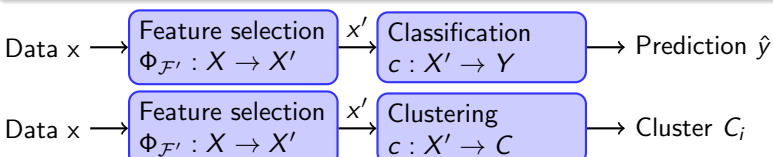
1. Desire in ML to **reduce the dimensionality** of the data (computation time, memory requirements, generalization)
2. Get some **insights** into problems instead of only solving them

Methods

- For both requirements, extracting a smaller subset of features with maximal information content is helpful
  ($\rightarrow$ **Feature Selection**)
- Mostly for the first requirement, combining a number of features into a smaller set of joint features is desirable, too
  ($\rightarrow$ **Dimensionality Reduction**)

Feature Selection

- Given a training set $\mathcal{D}_{train}$ consisting of n instances (and their corresponding class label / value)
- Let each instance x of $\mathcal{D}_{train}$ be a vector of m features ($\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$)
- Create a projection $\Phi_{\mathcal{F}'}$ in feature space, e.g. $\Phi_{\{1,3\}}((f_1, f_2, f_3)) = (f_1, f_3)$
- (Some FS algorithms only work in supervised problems)



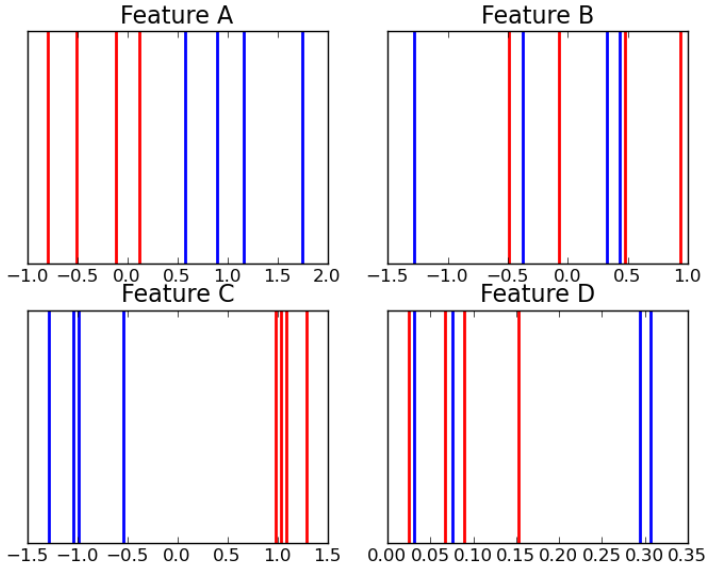
Feature Selection: Objective

- Objective: Find a subset of features $\mathcal{F}' \subsetneq \mathcal{F}$ with minimal cardinality $m' = |\mathcal{F}'|$ such that a classifier trained on $\mathcal{D}'_{train} = \{\Phi_{\mathcal{F}'}(x) | x \in \mathcal{D}_{train}\}$ achieves maximal performance.
- Two conflicting objectives:
  - m' as small as possible
  - Maximal predictive performance of classifier trained on $\mathcal{D}'_{train}$
- Often, one specifies m' and tries to find a set $\mathcal{F}'$ that allows for a maximal predictive performance.
- Determining the predictive performance of a classifier for a given feature set $\mathcal{F}'$ is often computationally expensive
- Are there computationally cheaper ways of determining the "merit" of a feature set $\mathcal{F}'$?

Some example data

Feat. A	Feat. B	Feat. C	Feat. D	class
-0.118	0.934	0.979	0.152	-1
-0.800	-0.489	1.082	0.024	-1
-0.511	0.475	1.029	0.089	-1
0.120	-0.076	1.291	0.067	-1
1.747	-1.276	-0.546	0.307	+1
0.897	0.327	-1.291	0.294	+1
1.157	0.432	-0.991	0.031	+1
0.573	-0.371	-1.037	0.076	+1

Visualization



Useful Features (informal)

- The features “A” and “C” seem to be **useful**, they “help to distinguish the classes”
- Feature “C” appears to be particularly useful since it has only small variance
- The features “B” and “D” seem to be less useful; it is not obvious how they could help in distinguishing the classes
- Without feature “C”, feature “A” would be very useful. But if feature “C” is known, feature “A” is less useful.
- There is some **redundancy** in the two features (which is not necessarily bad)

Recap: Multivariate Normal Distribution

Probability density function:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{n}{2}} \det(\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- $\boldsymbol{\mu} \in \mathbb{R}^n$ - mean vector
- $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ - covariance matrix

Covariance matrix:

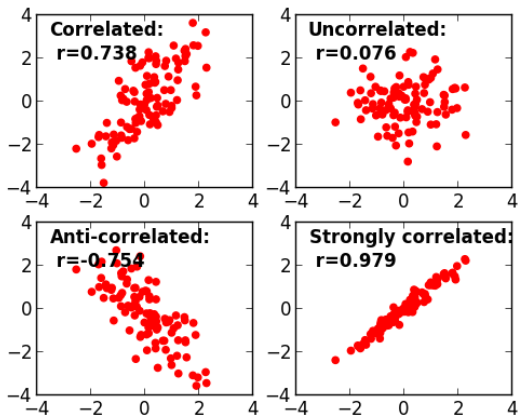
$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho_{1,2}\sigma_1\sigma_2 & \dots & \rho_{1,n}\sigma_1\sigma_n \\ \rho_{2,1}\sigma_2\sigma_1 & \sigma_2^2 & \dots & \rho_{2,n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n,1}\sigma_n\sigma_1 & \rho_{n,2}\sigma_n\sigma_2 & \dots & \sigma_n^2 \end{pmatrix}$$

Correlation coefficient (1/2)

Pearson product-moment correlation coefficient:

$$r_X(f_i, f_j) = \frac{\sum_{k=1}^{n}(X[k,i]-\overline{X[:,i]})(X[k,j]-\overline{X[:,j]})}{\sqrt{\sum_{k=1}^{n}(X[k,i]-\overline{X[:,i]})^2}\sqrt{\sum_{k=1}^{n}(X[k,j]-\overline{X[:,j]})^2}}$$

- Is an estimate for $\rho(f_i, f_j) = \frac{cov(X_i,X_j)}{\sigma_{X_i}\sigma_{X_j}}$
- Can be used to compute the **linear correlation** between
  - two features
  - a feature and the class label



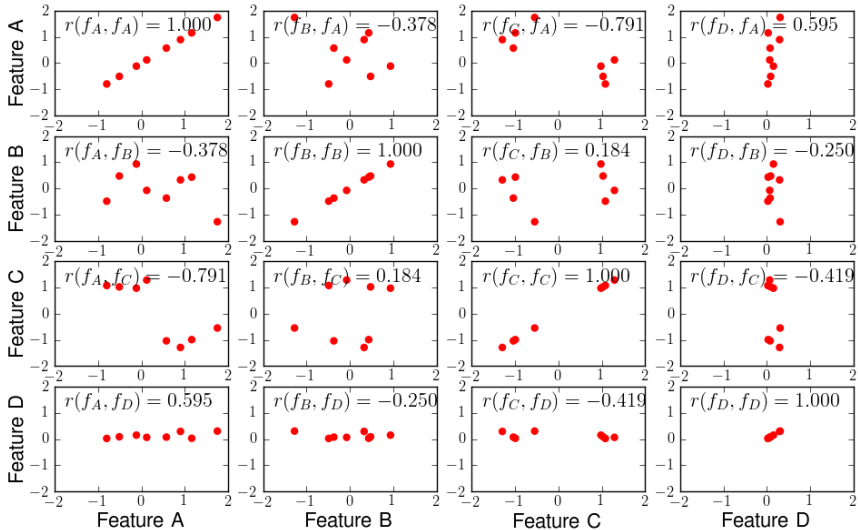
Correlation coefficient (2/2)

	Feat. A	Feat. B	Feat. C	Feat. D	Class
Feat. A	1.000	-0.378	-0.791	0.595	0.875
Feat. B	-0.378	1.000	0.184	-0.250	-0.332
Feat. C	-0.791	0.184	1.000	-0.419	-0.980
Feat. D	0.595	-0.250	-0.419	1.000	0.447

Table: Pearson correlation coefficient for the example data

- Pearson correlation coefficient can help to identify useful and redundant features
- It does only capture linear dependencies like $x_1 = \alpha x_2$, but not non-linear dependencies like $x_1 = x_2^2$

Visualisation



A Naive Feature Selector (1/3)

NaiveFS($\mathcal{F}, \mathcal{D}, k$):

```
/* Selects k features from F that have the strongest correlation
   with the class on training data D.                          */
```

$\mathcal{F}' \leftarrow \{\}$;

while $|\mathcal{F}'| < k$ **do**

  $feature \leftarrow \arg\max_{f \in \mathcal{F}} |r_{\mathcal{D}}(f, c)|$;

  $\mathcal{F}' \leftarrow \mathcal{F}' \cup feature$;

  $\mathcal{F} \leftarrow \mathcal{F} \setminus feature$;

end

return $\mathcal{F}'$;

A Naive Feature Selector (2/3)

Whats wrong with the "naive" algorithm?

- Consider the following dataset:

Feature A	Feature B	Feature C	Class
-0.890	-0.843	-0.424	-1
-1.083	-1.165	-0.071	-1
-0.392	-0.690	-1.756	-1
-1.063	-1.114	-0.678	-1
1.027	0.938	0.510	1
1.396	1.546	0.572	1
0.773	0.356	0.564	1
0.852	0.677	0.789	1

A Naive Feature Selector (3/3)

The naive feature selector would

- first select feature "A" with $|r_{\mathcal{D}}(f_A, c)| = 0.963$
- secondly select feature "B" with $|r_{\mathcal{D}}(f_B, c)| = 0.938$

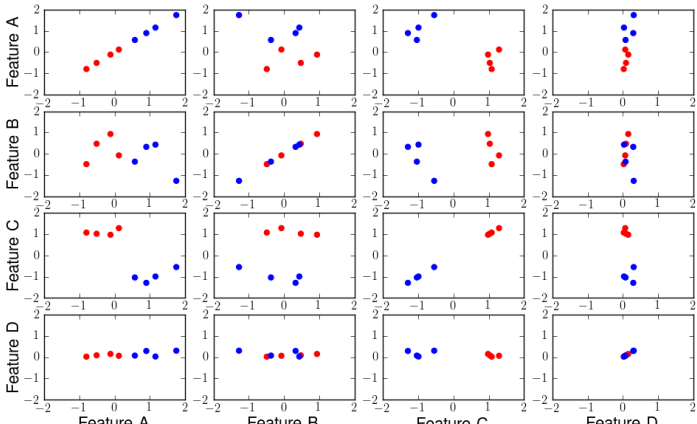
But:

- $|r_{\mathcal{D}}(f_A, f_B)| = 0.985$
- Often, two features that are too strongly correlated are not independent and thus potentially affected by the same errors
- Thus, it might be more desirable to select features that are only weakly correlated.

Correlation-based Feature Selection (CFS)

Objective:

- Good feature subsets contain features that are highly correlated with the class, yet uncorrelated with each other.



Merit

Definition with $\mathcal{F}' \subseteq \mathcal{F}$:

$$merit(\mathcal{F}') = \frac{|\mathcal{F}'|\overline{|r_{cf}|}}{\sqrt{|\mathcal{F}'| + |\mathcal{F}'|(|\mathcal{F}'|-1)\overline{|r_{ff}|}}}$$

- $\overline{|r_{cf}|}$: abs. val. average of feature-class correlation,
- $\overline{|r_{ff}|}$: abs. val. average of feature-feature correlation in $\mathcal{F}'$ on $\mathcal{D}$ (no auto-correlations, $\overline{|r_{\{S\}\{S\}}|} := 1$).
- Optimal feature subset is $\mathcal{F}'_{opt} = \arg\max_{\mathcal{F}' \subseteq \mathcal{F}} merit(\mathcal{F}')$

$\mathcal{F}'$	$\overline{\lvert r_{ff} \rvert}$	$\overline{\lvert r_{cf} \rvert}$	$merit$
$\{\}$	—	—	0.000
$\{A\}$	1.000	0.875	0.875
$\{B\}$	1.000	0.332	0.332
$\{C\}$	1.000	0.980	0.980
$\{D\}$	1.000	0.447	0.447
$\{A, B\}$	0.378	0.603	0.727
$\{A, C\}$	0.791	0.927	0.980
$\{A, D\}$	0.595	0.661	0.740
$\{B, C\}$	0.184	0.656	0.853
$\{B, D\}$	0.250	0.390	0.493
$\{C, D\}$	0.420	0.714	0.847
$\{A, B, C\}$	0.451	0.729	0.915
$\{A, B, D\}$	0.408	0.551	0.709
$\{A, C, D\}$	0.602	0.767	0.895
$\{B, C, D\}$	0.285	0.587	0.811
$\{A, B, C, D\}$	0.436	0.659	0.867

Merit

Definition with $\mathcal{F}' \subseteq \mathcal{F}$:

$$merit(\mathcal{F}') = \frac{|\mathcal{F}'|\overline{|r_{cf}|}}{\sqrt{|\mathcal{F}'| + |\mathcal{F}'|(|\mathcal{F}'|-1)\overline{|r_{ff}|}}}$$

	Feat. A	Feat. B	Feat. C
Feat. B	0.378		
Feat. C	0.791	0.184	
Feat. D	0.595	0.250	0.419

Table: Absolute correlation coefficients for the example data

	FClass
Feat. A	0.875
Feat. B	0.332
Feat. C	0.980
Feat. D	0.447

$\mathcal{F}'$	$\overline{\lvert r_{ff}\rvert}$	$\overline{\lvert r_{cf}\rvert}$	$merit$
$\{\}$	—	—	0.000
$\{A\}$	1.000	0.875	0.875
$\{B\}$	1.000	0.332	0.332
$\{C\}$	1.000	0.980	0.980
$\{D\}$	1.000	0.447	0.447
$\{A, B\}$	0.378	0.603	0.727
$\{A, C\}$	0.791	0.927	0.980
$\{A, D\}$	0.595	0.661	0.740
$\{B, C\}$	0.184	0.656	0.853
$\{B, D\}$	0.250	0.390	0.493
$\{C, D\}$	0.420	0.714	0.847
$\{A, B, C\}$	0.451	0.729	0.915
$\{A, B, D\}$	0.408	0.551	0.709
$\{A, C, D\}$	0.602	0.767	0.895
$\{B, C, D\}$	0.285	0.587	0.811
$\{A, B, C, D\}$	0.436	0.659	0.867

Searching the Feature-Subset Space

- If there are m features ($|\mathcal{F}| = m$), then there are 2^m subsets ($|2^{\mathcal{F}}| = 2^m$).
- For large m ($m \geq 20$), computing the merit for all subsets would be infeasible.
- Thus, only a subset of $2^{\mathcal{F}}$ can be tested.
- This problem can be framed into a heuristic search problem.
- Possible heuristics: "best first", "greedy hill climbing", or genetic algorithms

Greedy Hill-climbing Heuristic

GreedyCFS($\mathcal{F}, \mathcal{D}$):

```
/* Selects feature subset F' ⊆ F that maximizes merit(F') on D.
   Uses the greedy hill climbing heuristic.                    */
```

$\mathcal{F}' \leftarrow \{\}$;

while *True* **do**

  $last\_merit \leftarrow merit(\mathcal{F}')$;

  $f_{sel} \leftarrow \arg\max_{f \in \mathcal{F}} merit(\mathcal{F}' \cup \{f\})$;

  if $merit(\mathcal{F}' \cup \{f_{sel}\}) < last\_merit$ **then**

    return $\mathcal{F}'$;

  end

  $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{f_{sel}\}$;

  $\mathcal{F} \leftarrow \mathcal{F} \setminus f_{sel}$;

end

Feature Selection - Conclusion

Restrictions of the presented method:

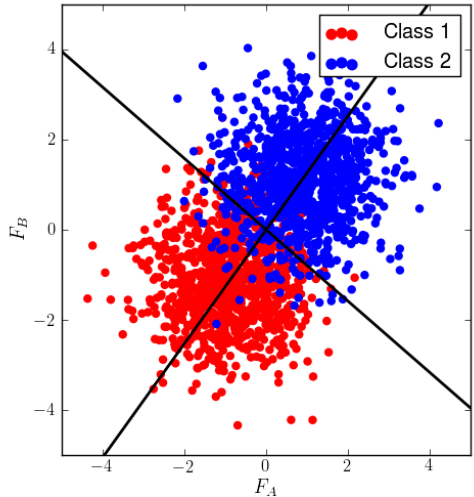
- Only *linear* correlations are detected (not $f_1 = f_2^2$)
- Only *pairwise* correlations are detected (not $f_1 + f_2 = f_3$)
- Features have to be real-valued!

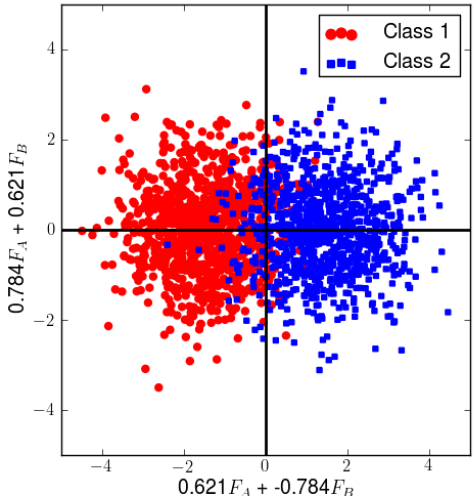
Dimensionality Reduction

Linearly Correlated Features

- Consider a setup where two features f_a, f_b are given that are strongly correlated and both affected by noise and we want to return only one feature
- Could we do better than just picking the one that looks (by chance) better?

- We could create a new feature $f_m = w_a f_a + w_b f_b$ that captures the essence of the two original features



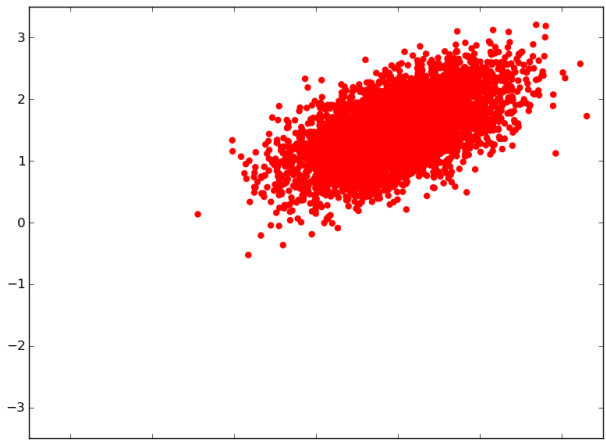


Feature	F_A	F_B	$0.621F_A$-$0.784F_B$	$0.784F_A$+$0.621F_B$
Correlation to class	0.656	0.747	0.817	0.002

Principal Component Analysis

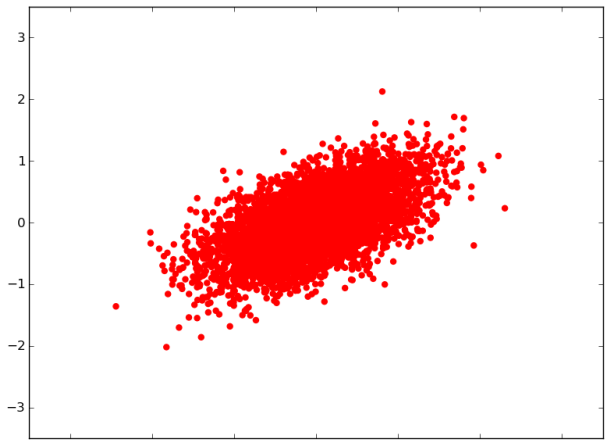
- One way to choose the new base is the "Principal Component Analysis" (PCA)
- Objective: detect correlations in data and reveal the internal structure of data
- Formally: find some orthonormal matrix A such that $\Sigma_Y = \frac{1}{n-1} YY^T$ is diagonalized (where $Y = AX$).
- Intuitively: PCA finds the axes of maximum variance
- Does not use class labels (unsupervised)

PCA Visualisation



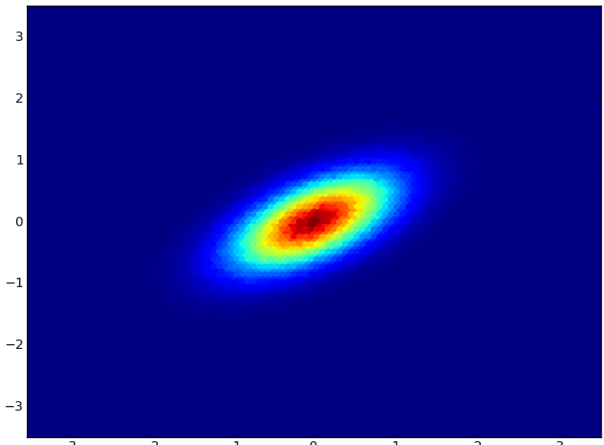
Some normally distributed data

PCA Visualisation



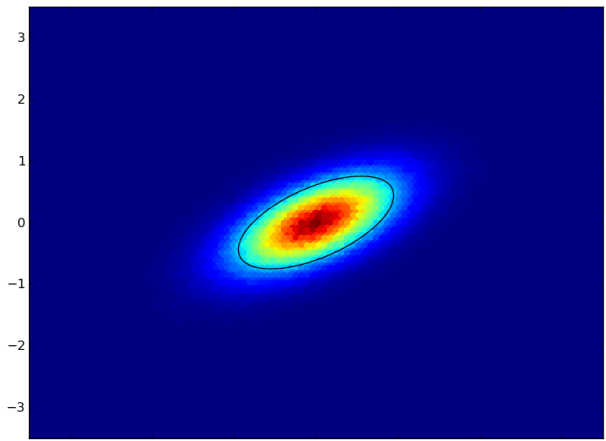
Subtracting the empirical mean $X \leftarrow X - \bar{X}$

PCA Visualisation



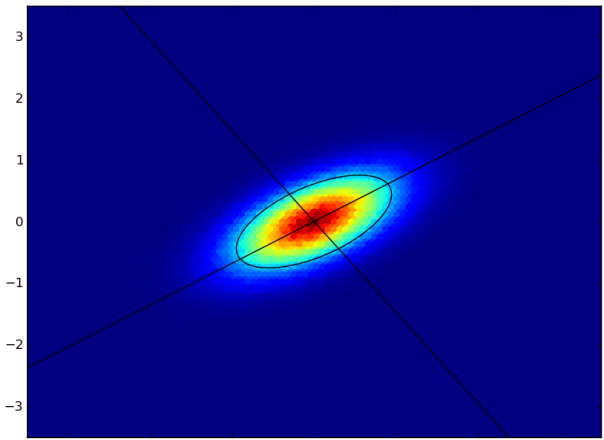
Fitting a multivariate Gaussian probability density function $p(x) = const * exp(-0.5x^T \Sigma^{-1} x)$

PCA Visualisation



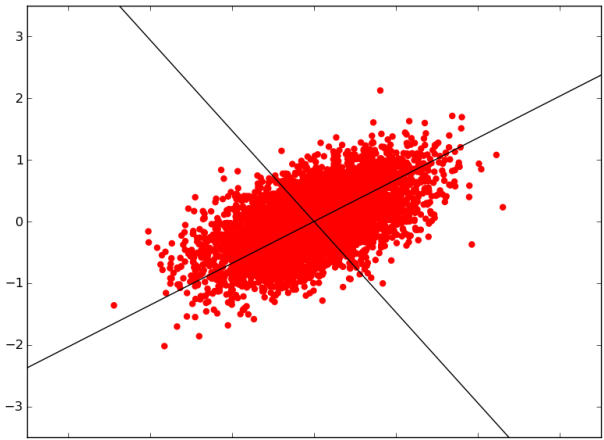
Isoprobability lines of gaussians are ellipses

PCA Visualisation



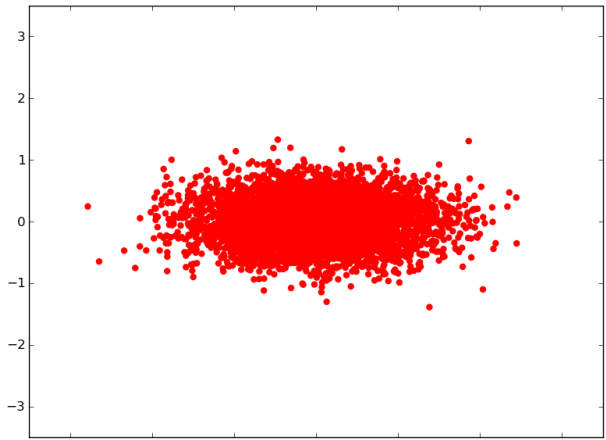
Computing principal axes of gaussian's ellipse (eigendecomposition)

PCA Visualisation



New coordinate frame

PCA Visualisation



Data in rotated coordinate frame

PCA Pseudocode

PCA(X, k):

/* Projects data matrix X with n samples and m dims onto its first k principal components. */

// Subtract emp. mean $\bar{X} = n^{-1} \sum_{i=1}^{n} X[i, :]$

$X \leftarrow X - \bar{X}$;

// Compute emp. $m \times m$ covariance matrix

$\Sigma \leftarrow (n-1)^{-1} X^T \cdot X$;

// Compute eigendecomposition ($\Sigma W = W\Lambda$) sorted by size of eigenvalues $\lambda \in \Lambda$

$\Lambda, W = \text{eig}(\Sigma)$;

// Project onto first k principal components

$Y \leftarrow X \cdot W[:, 0 : k]$;

return Y;

Thank You!
Please feel free to ask questions in the forums.